Oracle® HTML DB

User's Guide

Release 1.6

Part No. B14303-02

March 2005

ORACLE®

Oracle HTML DB User's Guide, Release 1.6

Part No. B14303-02

Primary Author: Terri Winters

Contributors: Carl Backstrom, Christina Cho, Michael Hichwa Joel Kallman, Sharon Kennedy, Syme Kutz, Sergio Leunissen, Raj Mattamal, Tyler Muth, Kris Rice, Marc Sewtz, Scott Spadafore, Scott Spendolini, and Jason Straub

Contents

3 Running a Demonstration Application

Part II Using Oracle HTML DB

4 Managing Data with Data Workshop

5 Managing Database Objects with SQL Workshop

6 Application Builder Concepts

7 Using Application Builder

8 Building an Application

9 Controlling Page Layout and User Interface

10 Adding Navigation

11 Debugging an Application

12 Deploying an Application

13 Managing a Development Workspace

14 Managing Security

15 Advanced Programming Techniques

16 Managing Globalization

17 Oracle HTML DB APIs

Part III Administration

18 Managing an Oracle HTML DB Hosted Service

Send Us Your Comments

Oracle HTML DB User's Guide, Release 1.6

Part No. B14303-02

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227 Attn: Server Technologies Documentation Manager
- Postal service:

  Oracle Corporation
  Oracle Server Technologies Documentation
  500 Oracle Parkway, Mailstop 4op11
  Redwood Shores, CA 94065
  U.S.A.

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle HTML DB User's Guide describes how to use the Oracle HTML DB development environment to build and deploy database-centric Web applications. Oracle HTML DB turns a single Oracle database into a shared service by enabling multiple workgroups to build and access applications as if they were running in separate databases.

This preface contains these topics:

Audience

Oracle HTML DB User's Guide is intended for application developers who are building database-centric Web applications using Oracle HTML DB. The guide describes how to use the Oracle HTML DB development environment to build, debug, manage, and deploy applications.

To use this guide, you need to have a general understanding of relational database concepts as well as an understanding of the operating system environment under which you are running Oracle HTML DB.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

This document contains:

Part I, "Getting Started with Oracle HTML DB"

Part I provides an introduction to Oracle HTML DB by introducing you to basic Oracle HTML DB concepts.

Chapter 1, "What is Oracle HTML DB?"

This section offers a general description of Oracle HTML DB and the components you can use it to develop database-centric Web applications.

Chapter 2, "Quick Start"

This section offers a quick introduction to using Oracle HTML DB.

Chapter 3, "Running a Demonstration Application"

This section describes how to run a demonstration application and defines fundamental concepts that are unique to Oracle HTML DB.

Part II, "Using Oracle HTML DB"

Part II describes how to use Data Workshop, SQL Workshop, and Application Builder to develop database-driven applications.

Chapter 4, "Managing Data with Data Workshop"

This section describes how to use Data Workshop to import data into and export data using a Web Browser.

Chapter 5, "Managing Database Objects with SQL Workshop"

This section provides information on how to use SQL Workshop to view and manage database objects, create and manage user interface defaults, and browse the data dictionary.

Chapter 6, "Application Builder Concepts"

This section provides basic conceptual information about Application Builder, the core component within Oracle HTML DB that enables you to build database centric Web applications.

Chapter 7, "Using Application Builder"

This section provides important background information about using Application Builder.

Chapter 8, "Building an Application"

This section describes how to use Application Builder to build an application and application components.

Chapter 9, "Controlling Page Layout and User Interface"

This section describes different approaches to customizing an application's user interface and page layout including customizing regions, editing item attributes, customizing templates, and incorporating cascading style sheets and images.

Chapter 10, "Adding Navigation"

This section describes how to implement navigation in your application using different types of navigation controls, including navigation bars, tabs, menus, lists, and trees.

Chapter 11, "Debugging an Application"

This section describes a approaches to debugging your application including viewing Debug Mode, enabling SQL tracing, viewing page reports, and how to manually remove a controls to isolate a problem.

Chapter 12, "Deploying an Application"

This section describes how move an application from one Oracle HTML DB instance to another.

Chapter 13, "Managing a Development Workspace"

This section describes different user roles and common tools and reports available to developers and Workspace administrators.

Chapter 14, "Managing Security"

This section describes how to provide security for your application through authentication and authorization.

Chapter 15, "Advanced Programming Techniques"

This section provides information about advanced programming techniques including establishing database links, using collections, running background SQL, implementing Web Services and managing user preferences.

Chapter 16, "Managing Globalization"

This section describes how to translate an application created in Oracle HTML DB.

Chapter 17, "Oracle HTML DB APIs"

This chapter describes available Oracle HTML DB APIs.

Part III, "Administration"

Part III describes the tasks associated with administering Oracle HTML DB, including creating and managing workspaces, translating an application, and managing activities, log files, and sessions.

Chapter 18, "Managing an Oracle HTML DB Hosted Service"

This section describes tasks an Oracle HTML DB administrator performs when administering an Oracle HTML DB hosted service.

Appendix A, "Available Conditions"
Provides a listing of conditions available in Oracle HTML DB.

Related Documents

For more information, see these Oracle resources:

- *Oracle HTML DB 2 Day Developer*
- *Oracle Database Concepts*
- *Oracle Database Application Developer's Guide - Fundamentals*
- *Oracle Database Administrator's Guide*
- *Oracle Database SQL Reference*

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/membership/`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/documentation/`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.

Convention	Meaning	Example
UPPERCASE monospace (fixed-width) font	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, Recovery Manager keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a NUMBER column. You can back up the database by using the BACKUP command. Query the TABLE_NAME column in the USER_TABLES data dictionary view. Use the DBMS_STATS.GENERATE_STATS procedure.
lowercase monospace (fixed-width) font	Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names and connect identifiers, user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. *Note:* Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter sqlplus to start SQL*Plus. The password is specified in the orapwd file. Back up the datafiles and control files in the /disk1/oracle/dbs directory. The department_id, department_name, and location_id columns are in the hr.departments table. Set the QUERY_REWRITE_ENABLED initialization parameter to true. Connect as oe user. The JRepUtil class implements these methods.
lowercase italic monospace (fixed-width) font	Lowercase italic monospace font represents placeholders or variables.	You can specify the *parallel_clause*. Run *old_release*.SQL where *old_release* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Anything enclosed in brackets is optional.	DECIMAL (*digits* [, *precision*])
{ }	Braces are used for grouping items.	{ENABLE | DISABLE}
|	A vertical bar represents a choice of two options.	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	Ellipsis points mean repetition in syntax descriptions. In addition, ellipsis points can mean an omission in code examples or text.	CREATE TABLE ... AS *subquery*; SELECT *col1*, *col2*, ... , *coln* FROM employees;
Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

Convention	Meaning	Example

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case-sensitive, you can use them in either UPPERCASE or lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

lowercase	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose **Start** > *menu item*	How to start a program.	To start the Database Configuration Assistant, choose **Start** > **Programs** > **Oracle** - *HOME_NAME* > **Configuration and Migration Tools** > **Database Configuration Assistant**.
File and directory names	File and directory names are not case-sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the filename begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32
C:\>	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\> exp HR/HR TABLES=emp QUERY=\"WHERE job='REP'\"`

Convention	Meaning	Example
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory. The default for Windows NT was `C:\orant`. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle\product\10.1.0`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\product\10.1.0\db_`*n*, where *n* is the latest Oracle home number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Installation Guide for 32-Bit Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

What's New in Oracle HTML DB?

This section describes new features of Oracle HTML DB release 1.6 and provides pointers to additional information.

Oracle HTML DB Release 1.6 New Features

- Streamlined user interface

  Release 1.6 features a redesigned and simpler user interface that reduces the number of clicks needed to edit an application or application components. Users now navigate by first selecting an application from the Applications list and then drilling down to specific components.

  See Also: "About the Oracle HTML DB User Interface" on page 2-4 and "Accessing Application Builder" on page 7-1

- Master Detail Form Wizard

  The new Master Detail Form Wizard automates the process of building a master detail form. This wizard creates a form which displays a master row and multiple detail rows within a single HTML form. With this form, users can query, insert, update, and delete values from two tables or views.

  See Also: "Building a Master Detail Form" on page 8-20

- Improved Web services support

  Developers can incorporate calls to Web services within an Oracle HTML DB application. Users can now create a reusable Web Service reference by either browsing a Universal Description, Discovery, and Integration (UDDI) registry, or by specifying a Web Services Description Language (WSDL). New wizards enable developers to create a from or report on a Web service.

  See Also: "Implementing Web Services" on page 15-14

- Improved Quick Application Wizard

  The new Create Application Wizard enables developers to create an application directly from spreadsheet data, or an existing database table. The resulting application contains more summary reports and charts as well as additional user interface options.

  See Also: "Creating a New Application" on page 8-2 and "Creating an Application Using the Create Application Wizard" on page 2-5

- Themes

  In release 1.6, application templates are organized into named collections called themes. Each theme contains templates for every type of application component and page control. Developers can change the look an entire application by selecting and applying a new theme.

  See Also: "Managing Themes" on page 9-8

- Enhanced calendar capability

  Developers can quickly generate a monthly calendar based on a specific table and then create drill down links to information stored in specific columns.

  See Also: "Creating Calendars" on page 8-43

- Tabular forms enhancements

  The Tabular Form Wizard creates a form to perform update, insert, and delete operations on multiple rows in a database table. With release 1.6, tabular forms now supports the addition of a row selector for bulk operations, row highlighting, date picker functionality, check boxes, declarative add row, and support for the data entry using a format mask (for example, $1234,89).

  See Also: "Creating a Tabular Form" on page 8-18

- Charting enhancements

  In release 1.6, Oracle HTML DB now includes stacked bar charts, cluster bar charts, and dial charts. Additionally, all SVG (Scalable Vector Graphics) charts support an asynchronous refresh of source data as well as support for a custom cascading style sheets.

  See Also: "Creating Charts" on page 8-24

- Page groups

  Developers can make the pages within their application even easier to access by organizing them into page groups.

  See Also: "Grouping Pages" on page 8-5

- Page locking

  Prevent conflicts during application development by locking application pages. Locking a page, prevents other developers from editing it.

  See Also: "Locking and Unlocking Page" on page 8-6

Part I

Getting Started with Oracle HTML DB

Part I provides an introduction to Oracle HTML DB. These chapters introduce you to basic Oracle HTML DB concepts.

Part I contains the following chapters:

1

What is Oracle HTML DB?

The section offers a general description of Oracle HTML DB and the components you can use to develop database-centric Web applications.

This section contains the following topics:

About Oracle HTML DB

Oracle HTML DB is a hosted declarative development environment for developing and deploying database-centric Web applications. Oracle HTML DB turns a single Oracle database into a shared service by enabling multiple workgroups to build and access applications as if they were running in separate databases. Thanks to built-in features such as design themes, navigational controls, form handlers, and flexible reports, Oracle HTML DB accelerates the application development process.

The HTML DB engine renders applications in real time from data stored in database tables. When you create or extend your application, Oracle HTML DB creates or modifies metadata stored in database tables. When the application is run, the HTML DB engine then reads the metadata and displays the application.

Oracle HTML DB automatically maintains session state without requiring any coding. To provide stateful behavior within an application, Oracle HTML DB transparently manages session state in the database. Application developers can get and set session state using simple substitutions as well as standard SQL bind variable syntax.

The Oracle HTML DB development platform consists of the following components:

- Application Builder
- SQL Workshop
- Data Workshop

About Application Builder

You use Application Builder to assemble an HTML interface (or application) on top of database objects such as tables and procedures. An application is a collection of database-driven Web pages linked together using tabs, buttons, or hypertext links.

Once you create an application, the HTML DB engine renders the application using the templates and user interface elements you specify.

A page is the basic building block of an application. Each page can have buttons and fields and can include application logic (or processes). You can branch from one page to the next using conditional navigation, perform calculations, run validations (such as edit checks), and display reports, forms, and charts.

> **See Also:** "Application Builder Concepts" on page 6-1 and "Using Application Builder" on page 7-1

About SQL Workshop

You use SQL Workshop to view and manage database objects from a Web browser. Using SQL Workshop you can store and retrieve data, execute SQL commands, and perform the following tasks:

- Run SQL commands
- Upload and run SQL scripts
- Maintain a history of the executed SQL
- Create or modify database objects
- Query data by example
- Browse the data dictionary
- Enable database browsing with drill-up and drill-down

> **See Also:** "Managing Database Objects with SQL Workshop" on page 5-1

About Data Workshop

You use Data Workshop to import data into and export data from the hosted database. Supported import formats include text (such as comma or tab delimited data), XML documents, and spreadsheets. Supported export formats include text (such as comma or tab delimited data) and XML documents.

For example, you can quickly share data with multiple users by converting a spreadsheet into a database table using the Import Spreadsheet Data Wizard. Running this wizard creates a new table and loads the data without requiring any SQL knowledge. Once the data is loaded into a database table, you can build a application on top of it just like you would on any other database table.

> **See Also:** "Managing Data with Data Workshop" on page 4-1and "Importing Spreadsheet Data" on page 4-2

2

Quick Start

This section offers a quick introduction to using Oracle HTML DB. This section assumes you have already completed the installation process.

This section contains the following topics:

- Understanding Oracle HTML DB User Roles
- Logging in to Oracle HTML DB
- About the Oracle HTML DB User Interface
- Creating an Application Using the Create Application Wizard

See Also:

- "Running a Demonstration Application" on page 3-1
- "Application Builder Concepts" on page 6-1
- "Using Application Builder" on page 7-1

Understanding Oracle HTML DB User Roles

In the Oracle HTML DB development environment, users log in to a shared work area called a workspace. Users are divided into three primary roles:

- Developer
- Workspace administrator
- Oracle HTML DB administrator

A **developer** can create and edit applications. A **Workspace administrator** performs administrator tasks specific to their workspace. An **Oracle HTML DB administrator** manages an entire Oracle HTML DB development environment instance.

See Also:

- "Application Builder Concepts" on page 6-1
- "Managing a Development Workspace" on page 13-1
- "Managing an Oracle HTML DB Hosted Service" on page 18-1

Logging in to Oracle HTML DB

When you log in to Oracle HTML DB you log in to a workspace. A workspace is an area within the Oracle HTML DB development environment where multiple developers can create applications.

Topics in this section include:

- Requesting a Workspace
- Logging in to a Workspace
- Resetting Your Password
- Logging Out of Your Workspace

Note: Before users can request a workspace or change their passwords, an Oracle HTML DB administrator must configure Oracle HTML DB environment preferences.

See Also: "Managing Environment Preferences" on page 18-17

Requesting a Workspace

Note: This section only applies if your Oracle HTML DB administrator has configured Oracle HTML DB to support workspace requests.

See Also: "Specifying a Provisioning Mode" on page 18-5 for more information on enabling workspace requests

Before you can log in to Oracle HTML DB, an administrator must grant you access to a workspace. Each workspace has a unique ID and name. Only an administrator with the appropriate credentials can create a new workspace.

Whether you should use an existing workspace or request a new one depends upon your development goals. Consider the following general criteria:

- **Existing Workspace** - Use an existing workspace if you are building an application on data that already exists in the database.
- **New Workspace** - Request a new workspace if you are building an application on data that does not yet exist in the database (for example, data from a spreadsheet or desktop database).

To request a workspace:

1. In a Web browser, navigate to the Oracle HTML DB Login page. By default, Oracle HTML DB installs to the following location:

   ```
   http://hostname:port/pls/htmldb/htmldb
   ```

   Where:

   - *hostname* is the name of the system where Oracle HTTP Server is installed.
   - *port* is the port number assigned to Oracle HTTP Server. In a default installation, this number is 7777. You can find information about your Oracle HTTP Server installation's port number from either of the following files:
     - *ORACLE_BASE**ORACLE_HOME*\install\portlist.ini
     - *ORACLE_BASE**ORACLE_HOME*\Apache\Apache\conf\httpd.conf
   - *htmldb* is the database access descriptor (DAD) defined in the mod_plsql configuration file.

The Login page appears.

2. Under Tasks, click **Request a Workspace**.

   The Request Service Wizard appears.

3. Click **Continue** and follow the on-screen instructions.

See Also: "Creating a Workspace" on page 18-4

Logging in to a Workspace

Once your workspace request has been approved, an Oracle HTML DB administrator provides you with a workspace name, username, and password.

To log in to Oracle HTML DB:

1. In a Web browser, navigate to the Oracle HTML DB Login page. By default, Oracle HTML DB installs to the following location:

   ```
   http://hostname:port/pls/htmldb/htmldb
   ```

   The Login page appears.

2. Under Login, type the following:
   - In Workspace, type the name of your workspace
   - In Username, type your username
   - In Password, type your case-sensitive password
3. Click **Login**.

Resetting Your Password

You can reset your password by clicking the Change Password link on the Oracle HTML DB home page.

To reset your password:

1. Log in to Oracle HTML DB. (See "Logging in to Oracle HTML DB" on page 2-1.)
2. Under Workspace Administration, click **Change Password**.
3. In Change Password, type the following:
   - In the Password field, type your new password
   - In the Confirm Password field, type your new password again
   - Click **Apply Changes**

Note: All users (developers and administrators) can use the Change Password link on the Oracle HTML DB home page to reset their password.

See Also: "Changing a User Password" on page 13-8

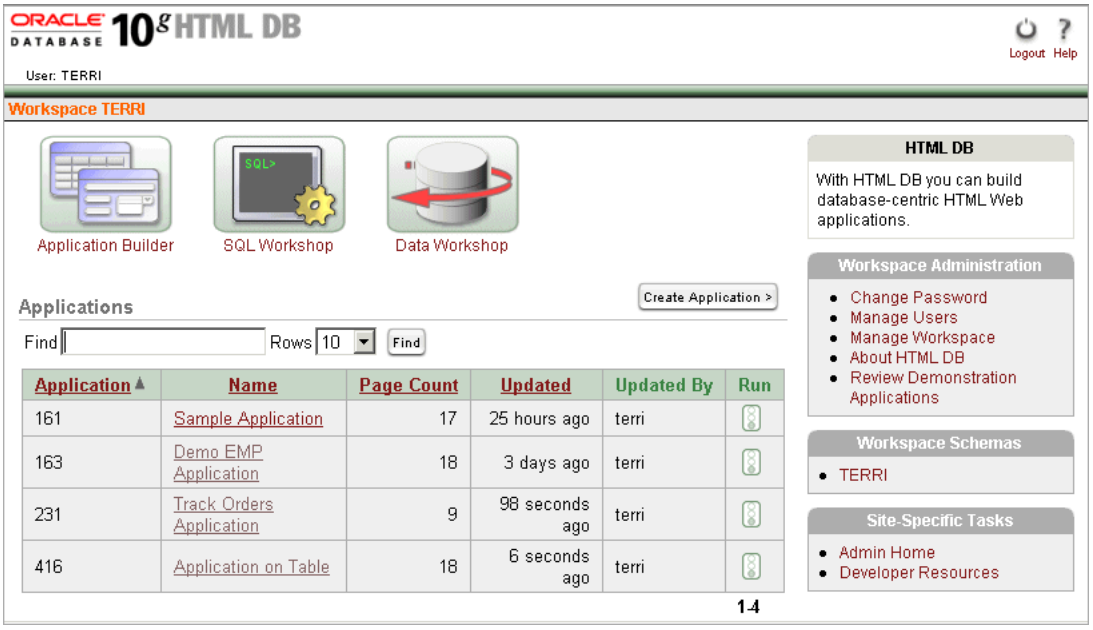
Logging Out of Your Workspace

To logout of Oracle HTML DB, click the **Logout** icon in the upper right corner of the window.

About the Oracle HTML DB User Interface

Once you log in to Oracle HTML DB, the Workspace home page appears as shown in Figure 2–1.

Figure 2–1 Workspace Home Page



The Workspace home page consists of five primary components:

- **Application Builder**. Use Application Builder to assemble an HTML interface (or application) on top of a database objects such as tables and procedures.
- **SQL Workshop**. Use SQL Workshop to view and manage database objects from a Web browser.
- **Data Workshop**. Use Data Workshop to import data into and export data from the hosted database.
- **Applications List**. Select an existing application name to open it in Application Builder.
- **Workspace Administration**. Use this list to manage user accounts, monitor workspace activity, review log files, manage session state, view reports, and manage development services.
- **Workspace Schemas**. Use this list to view details about the schemas accessible to the current workspace.

To access Application Builder, SQL Workshop, or Data Workshop, click the large icons in the center of the page.

See Also:

- "Using Application Builder" on page 7-1
- "Managing Database Objects with SQL Workshop" on page 5-1
- "Managing Data with Data Workshop" on page 4-1

Navigating Using Breadcrumb Menus

As shown in Figure 2–2, a breadcrumb menu appears at the top of every page in Oracle HTML DB. Each menu entry indicates where the current page is relative to other pages in the Oracle HTML DB development environment. Additionally, you can click a specific entry to instantly link to a previous page.

Figure 2–2 Breadcrumb menu

Accessing Online Help

Most pages in Oracle HTML DB include page level help. Page level help displays in a text box on the right side of the page and offers a brief description of the page functionality. Oracle HTML DB also includes two other forms of online help:

- **Procedural online help.** You can access an HTML-based online help system by clicking the Help icon in the upper right corner of the window.
- **Field level help.** Most lists of values, select lists, check boxes, and fields in Oracle HTML DB include item help. When item help is available, the item label appears highlighted when you pass your cursor over it. Clicking the item label displays a description in a separate window.

Creating an Application Using the Create Application Wizard

A quick way to make data in the Oracle database accessible to an end user is to run the Create Application Wizard. This wizard creates a basic application based upon a table and contains the following pages:

- Standard report
- Insert form
- Update form
- Success form (indicates when a record is successfully inserted)
- Analysis menu page
- Analysis reports
- Analysis charts
- Login page

The Create Application Wizard assumes you have a single table for which you want to create a report and update and insert data. Once the application is generated, you can modify it using Application Builder.

To create an application based on an existing table:

1. Log in to Oracle HTML DB. (See "Logging in to Oracle HTML DB" on page 2-1.)
2. When the Workspace home page appears, click the **Create Application** button.
3. On Select Creation Method, click **Based on an Existing Table**.
4. On Identify Table/View Owner, select the table or view owner on which your application will be based and click **Next**.

   Each application is based on a table or view owned by a specific database schema.

5. On Identify Table/View Name, select the table or view on which your application will be based and click **Next**.
6. On Table User Interface Defaults:
   a. In Singular Name, enter a singular name of the table. This name becomes the form region title.
   b. In Plural Name, enter the plural name of the table. This name becomes the report region title and tab label.
   c. If this is the first time you are creating an application on a given table, go to step e.
   d. For Save user interface defaults, specify whether to save user interface defaults.

      Selecting **Yes**, saves table properties, column properties, summary columns and aggregate by columns selections to user interface defaults. User interface defaults enable you to assign default user interface properties to a table, column, or view within a specified schema.
   e. For Column User Interface Defaults, edit the column labels (optional).
   f. Click **Next**.
7. On Summary By Column, select the columns that should have a detail report and chart and click **Next**.
8. On Aggregate By Column:
   a. From Columns to Aggregate, select the columns for which values will be aggregated.
   b. For Aggregate Function to Use, select aggregate method (Sum, Average, or both).
   c. Click **Next**.
9. On Application Options:
   a. For Application Name, enter a name for the application.
   b. For a Create Mode, select **Read and Write**.

      Read Only prevents users from changing the data in the underlying table. **Read and Write** enables users to change the data in the underlying table (that is, perform inserts and updates).
   c. For Chart Type, select a chart type.

      The wizard creates a chart page for each summary column you selected in the previous step.
   d. Click **Next**.
10. On Identify Primary Key, enter the name of the primary key column for the table and click **Next**.
11. On Primary Key Source, select a source type for the primary key column. Valid options include:
    - **Existing Trigger** - Select this option if a trigger is already defined for the table. You do not need to define a primary key column source for existing columns. Existing columns already have a primary key.

- **Custom PL/SQL Function** - Select this option if the source is a PL/SQL function returning the key value.
- **Existing Sequence** - Select this option if an existing sequence is defined for the table. If that is the case, the sequence next value would be used.

12. On Select Theme, select a theme for this application and click **Next**.

    A theme is a named collection of templates used to define the user interface of an application.

13. Confirm your selections and click **Create**.

    A confirmation page appears, displaying two icons:

    - Run Application
    - Edit Application

    See Also:

    - "Managing User Interface Defaults" on page 5-12 for more information on assigning default user interface properties to a table, column, or view
    - "Editing Application Attributes" on page 7-4 for more information on application attributes

Running Your Application

You can run your application by clicking the Run Application icon on the Create Application Success page.

To run your application from the Quick Application Confirmation page:

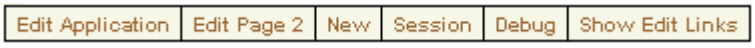
1. Click **Run Application**.

   The Login page appears.

2. Log in to your application by typing your workspace username and password and clicking **Login**.

   Your application appears. Note the Developer toolbar at the bottom on the page. (See Figure 2–3.)

Figure 2–3 Application Builder Developer Toolbar



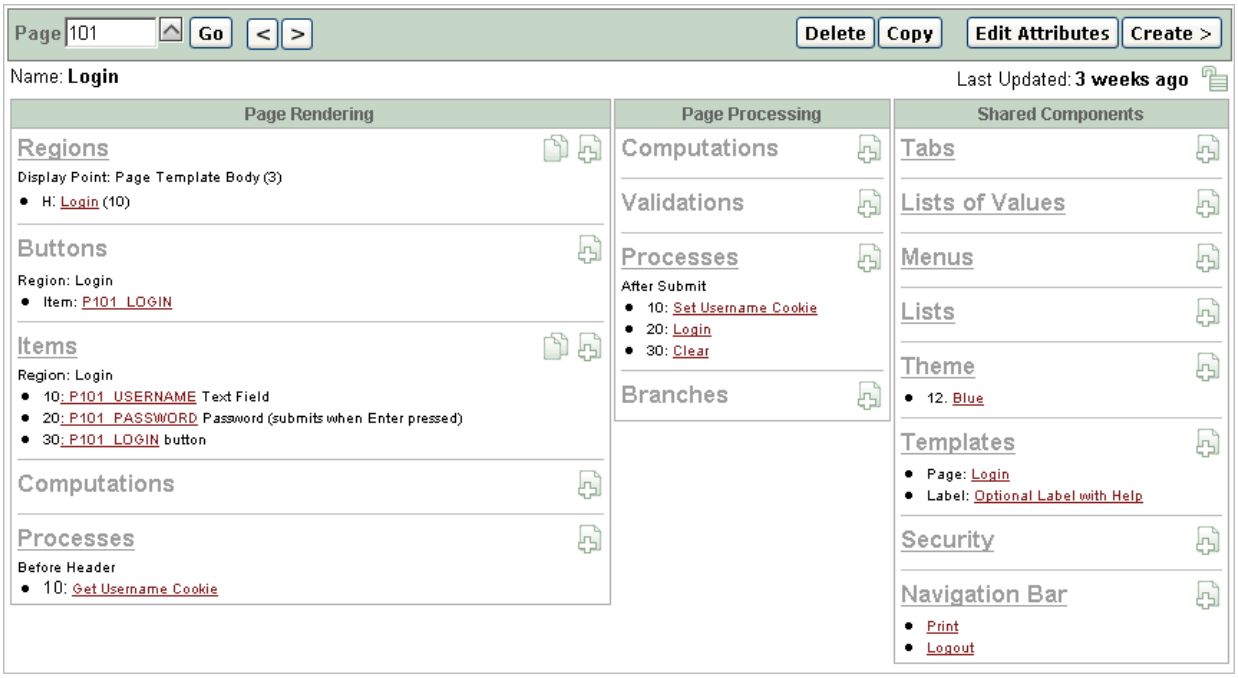
   The Developer toolbar offers a quick way to edit the current page, create a new page, control, or component, view session state, or toggle edit links on an off.

3. Explore your application.

4. To exit your application and return to Application Builder, click **Edit Page** on the Developer toolbar.

   As shown in Figure 2–4, the Page Definition appears.

Figure 2–4 Page Definition



A page is the basic building block of an application. You use the Page Definition to view, create, and edit the controls and components that define a page.

5. To return to Application Builder home page, select the Application Builder tab.

See Also:

- "Application Builder Concepts" on page 6-1
- "Accessing Application Builder" on page 7-1
- "Viewing a Page" on page 7-11

3

Running a Demonstration Application

This section describes how to run and modify the demonstration applications that install with Oracle HTML DB. Running and analyzing how an application works is an effective way to better understand how you can use Oracle HTML DB to build your own applications.

This section contains the following topics:

- Viewing and Installing a Demonstration Application
- Running a Demonstration Application
- Understanding Sample Application
- Modifying a Demonstration Application
- Viewing Underlying Database Objects

See Also:

- "What is Oracle HTML DB?"
- "Quick Start"
- "Application Builder Concepts"
- "Using Application Builder"

Viewing and Installing a Demonstration Application

Oracle HTML DB installs with a number of demonstration applications. Use these applications to learn more about the different types of functionality you can include in your applications.

To view the demonstration applications included with Oracle HTML DB:

1. Log in to Oracle HTML DB as described in "Logging in to Oracle HTML DB" on page 2-1.

   The Workspace home page appears.

2. From the Workspace Administration list, select **Review Demonstration Applications**.

   The Demonstration Applications page appears, displaying links to the following applications:

   - *Sample Application* offers a working demonstration that highlights basic design concepts
   - *Collection Showcase* demonstrates shopping cart concepts

- *Web Services* serves an example of how you can use Web Services
- *Presidential Inaugural Addresses* demonstrates Oracle Text

See Also: "Implementing Web Services" on page 15-14

The Status column indicates whether or not an application is currently installed.

To install a demonstration application:

1. Navigate to the Demonstration Applications page as described in the previous procedure.
2. Scroll down to the application you wish to install, click **Install**.
3. Follow the on-screen instructions.

Running a Demonstration Application

Oracle HTML DB installs with a number of demonstration applications. Once you have installed a demonstration application you can run it from the Demonstration Application page or from the Workspace home page.

By default, installing Sample Application creates two accounts, `demo` and `admin`. The default password for both accounts is the name of the current workspace in lowercase letters.

Running an Application from Demonstration Applications

The simplest way to run a demonstration application is navigate to the Demonstration Applications page.

To run a demonstration application from the Demonstration Applications page:

1. Log in to Oracle HTML DB as described in "Logging in to Oracle HTML DB" on page 2-1.

   The Workspace home page appears.
2. From the Workspace Administration list, select **Review Demonstration Applications**.
3. On the Demonstration Applications page, locate the application you wish to run.
4. In the Action column, click **Run**.
5. Enter the appropriate username and password and click **Login**
   - For User Name, enter either `demo` or `admin`
   - For Password, enter the name of the current workspace using all lowercase letters

Running an Application from the Workspace Home Page

Once you have installed a demonstration application, you can run it from Workspace home page.

To run a demonstration application from the Workspace home page:

1. Log in to Oracle HTML DB as described in "Logging in to Oracle HTML DB" on page 2-1.

   The Workspace home page appears.

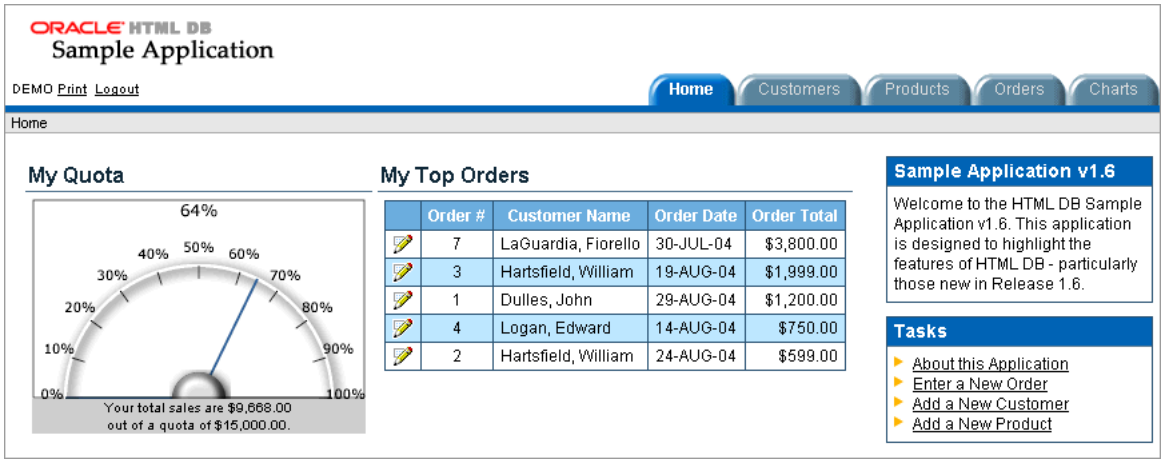
2. Locate the application you wish to run in the Applications list.
3. Click the **Run** icon to the right of the application name.
4. Enter the appropriate username and password and click **Login**
   a. For User Name, enter one of the following:
      - `demo`
      - `admin`
   b. For Password, enter the name of the current workspace using all lowercase letters.

Understanding Sample Application

Each demonstration application features a different set of functionality. This section describes the demonstration application, *Sample Application*.

As shown in Figure 3–1, Sample Application features an easy-to-use interface for viewing, updating, and searching order and customer information for electronic and computer products. Users can navigate between the pages using the Home, Customers, Products, Orders, and Charts tabs.

Figure 3–1 Sample Application, Home Page



Sample Application demonstrates the following functionality:

- Examples of ways to display summary information, including a dial chart and summary reports
- Reports for viewing, updating, and adding customers, products, and orders
- A Calendar report
- SVG charts available in Oracle HTML DB including cluster bar, pie chart, and stacked bar
- Printer friendly mode

The sections that follow describe specific functionality available on each page.

See Also: "What is a Page?" on page 6-3

About the Home Page

As shown in Figure 3–1, the Home page contains four regions:

- My Quota
- My Top Orders
- Sample Application 1.6
- Tasks list

My Quota demonstrates the use of a new SVG chart called a Dial Chart. This chart displays a value based on an underlying SQL statement. Although not demonstrated in this example, you can enable an asynchronous refresh by editing the attributes of any SVG chart.

My Top Orders is a simple report based on a SQL query. This report displays a subset of the information that appears on the Orders page. Users can link to order details by selecting the **Edit** icon.

Sample Application 1.6 is a simple HTML region that displays static text. You can create this type a region to display explanatory information to users.

Tasks list contains an Oracle HTML DB list with links to other pages in *Sample Application*. Links available on the Home page Tasks list include:

- **About this Application** links to an informational page that describes this application.
- **Enter a New Order** links to a wizard for creating a new order.
- **Add a New Customer** links to a form for entering new customer information.
- **Add a New Product** links to a form for adding a new product.

See Also:

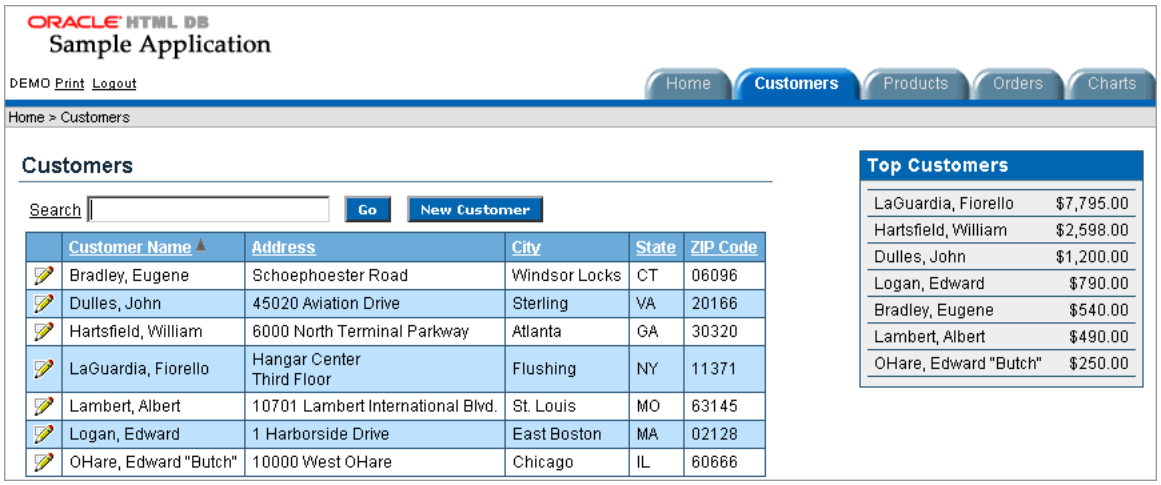
- "Creating Charts" on page 8-24
- "Creating a Report Using a Wizard" on page 8-9
- "Creating a Region" on page 9-4
- "Creating Lists" on page 10-11

About the Customers Page

As shown in Figure 3–2, the Customers page enables users to view and edit customer information. The Customers page consists of two main regions:

- Customers
- Top Customers

Figure 3–2 Sample Application, Customers Page



Customers is an updatable report for tracking customer information. This region is also based on a SQL query. To search for a customer, type a customer name in the Search for field and click **Go**. To sort by customer name, click the column heading. A Sort icon appears to the right of the heading, Customer Name. To update existing customer information, click the Edit icon.

Top Customers ranks customers by order amount. This report is based on a SQL query which returns top customers based on their orders.

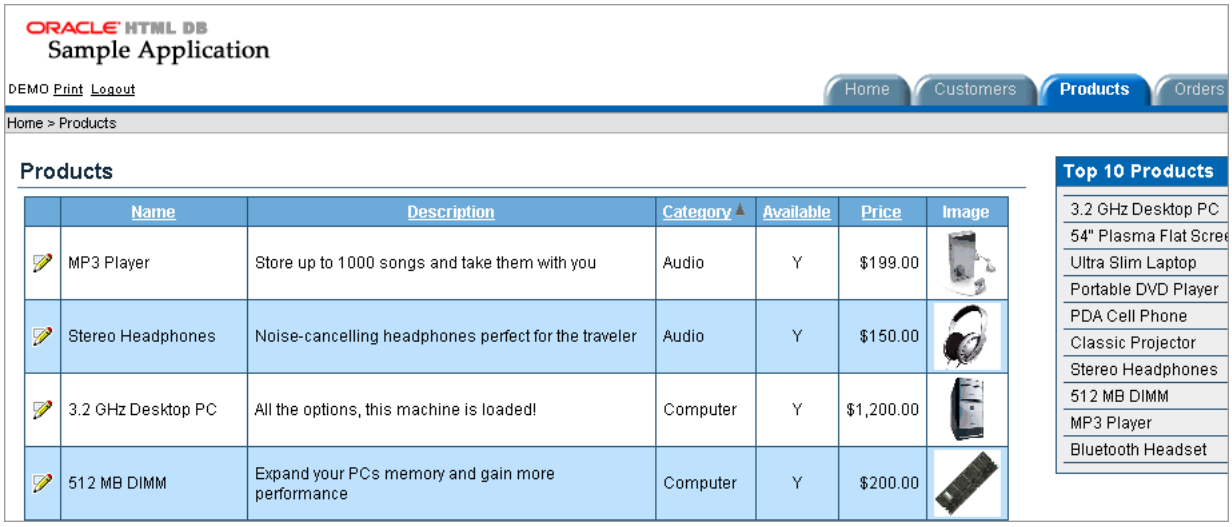
See Also: "Creating Reports" on page 8-8

About the Products Page

As shown in Figure 3–3, the Products page enables users to view and edit product information. The Products page consists of two main regions:

- Products
- Top 10 Products

Figure 3–3 Sample Application, Products Page



Products displays an updatable report for tracking product information. This region is based on a SQL query which makes use of a custom function for displaying images stored in the database. To sort by product category, click the column heading. A Sort icon appears to the right of the heading. To edit a product description, click the Edit icon. To add a new product, click the **Create Product** button at the bottom of the page. Users can export the data in the Products report to a spreadsheet, by clicking **Export to Spreadsheet**.

Top 10 Products is also a SQL report. This report outlines the top ten products based on quantities sold.

See Also: "Creating Reports" on page 8-8

About the Orders Page

The Orders page (see Figure 3–4) enables users to view and edit customer orders. The Orders page contains two regions:

- My Orders
- Order by Day

Figure 3–4 Sample Application, Orders Page

My Orders is an Easy Report which summarizes the current orders in the system. To sort a column, click the column heading. A Sort icon appears next to column heading. To edit an existing order, click the Edit icon. To add a new order, click the **Enter New Order** button.

Order by Day is a Calendar report. This report displays the amount of an order on its corresponding date in a calendar. Users can select a calendar entry to view order details.

See Also: "Creating Calendars" on page 8-43 and "Creating Reports" on page 8-8

About the Charts Page

The Charts page illustrates three of the several types of SVG charts available in Oracle HTML DB: cluster bar, pie chart, and stacked bar. To view a chart, select a chart type.

See Also: "Creating Charts" on page 8-24

About the Admin Page

The Admin page only displays if you log in to *Sample Application* using the user name `admin`. Sample Application makes use of a custom authentication scheme that stores user names and obfuscated passwords in a table. The Manage Users page enables you to manage additional users.

Note the this custom authentication scheme does not utilize any user names or passwords associated with Oracle HTML DB developers.

Viewing Pages in Printer Friendly Mode

Clicking Print in the upper right corner of the page displays the current page in Printer Friendly mode. When in Printer Friendly mode, the HTML DB engine displays all text within HTML form fields as text.

To enable your application to display in Printer Friendly mode, you need to create and then specify a Print Mode Page Template on the Edit Application Attributes page.

See Also: "Optimizing a Page for Printing" on page 9-39

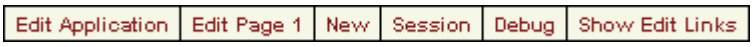
Modifying a Demonstration Application

Once you understand the type of functionality available in a demonstration application, the next step is to learn more about how each page is constructed. You edit an application using Application Builder. Using Application Builder you can edit existing pages in an application, add pages to an application, or create entirely new applications.

About the Developer Toolbar

When you log in to Oracle HTML DB having developer privileges and run an application, a Developer toolbar displays at the bottom of every page. As shown in Figure 3–5, the Developer toolbar offers a quick way to edit the currently running page, create a new page, control, or component, view session state, or turn edit links on or off.

Figure 3–5 Developer Toolbar in Sample Application



The Developer toolbar consists of the following links:

- **Edit Application** links you to the Application Builder home page. (See "Viewing a Page" on page 7-11.)
- **Edit Page** accesses the Page Definition for the currently running page. (See "Viewing a Page" on page 7-11.)

- **New** links to a wizard that enables you to create a new blank page, a component (report, chart, or form), a page control (region, button, or item), or a shared component (menu, list, or tab).
- **Session** links you to session state information for the current page. (See "Viewing Session State" on page 6-9.)
- **Debug** runs the current page in debug mode. (See "Accessing Debug Mode" on page 11-2.)
- **Show Edit Links** toggles between **Show Edit Links** and **Hide Edit Links**. Clicking **Show Edit Links** displays edit links next to each object on the page that can be edited. Each edit link resembles two colons (::) and appears to the right of navigation bar items, tabs, region titles, buttons, and items. Clicking on the link displays another window in which to edit the object.

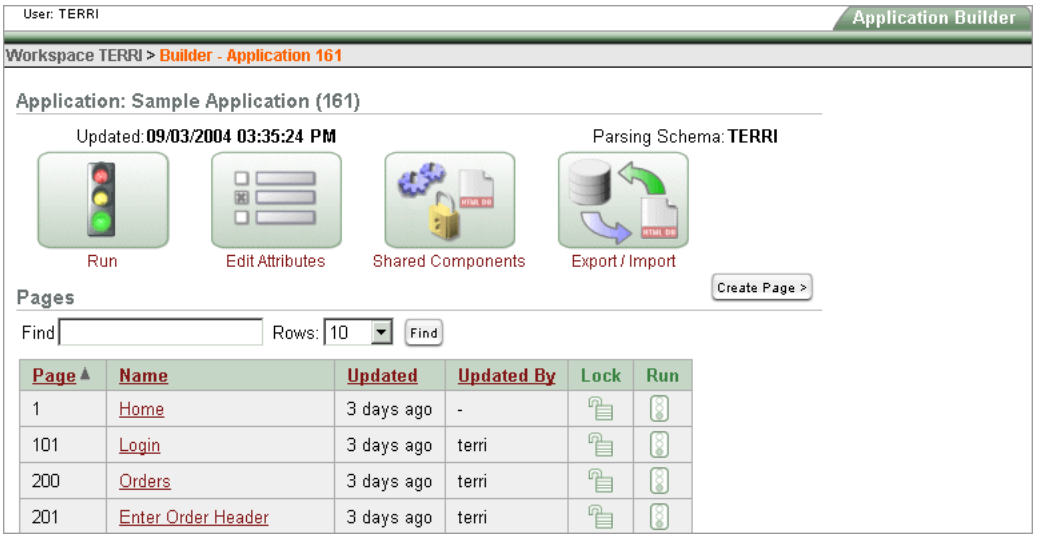
Editing a Demonstration Application

There are two common ways to edit a demonstration application:

- From Demonstration Applications page, click **Edit** next to the desired application.
- If you are running an application, click **Edit Application** on the Developer toolbar.

The Application Builder home page appears. As shown in Figure 3–6, the Application Builder home page displays the current application name and application ID, last update date, and parsing schema.

Figure 3–6 Application Builder Home Page



You can run the current application, edit application attributes, create shared components, export and import information, or create a new page by clicking the one of the following:

- **Run** submits the pages in the current application to the HTML DB engine to render viewable HTML
- **Edit Attributes** displays the Edit Application Attributes page
- **Shared Components** links to a new page for building shared application components and user interface controls
- **Export/Install** links you to the Export Import Wizard

- **Create Page** links to a wizard for creating a new page

The Pages list displays at the bottom of the page. To access a specific page, select the page name. To search for a specific page number, enter a page number in the Find field and click **Find**.

See Also:

- "Accessing Application Builder" on page 7-1 for more information on using the Application Builder home page
- "Viewing a Page" on page 7-11 for more information on viewing, creating, and editing the components and controls that define a page

Viewing Underlying Database Objects

The HTML DB engine renders applications in real time based on data stored in database tables. You can view the database objects for any demonstration application in SQL Workshop.

See Also: "Managing Database Objects with SQL Workshop" on page 5-1

To view the database objects used for an application:

1. Navigate to the Workspace home page.
2. Click the click the **SQL Workshop** icon.

   You view by schema and type and then by name. Under Database Browser, you can selecting existing database objects by selecting a database object type.
3. Under Database Browser, select **Tables**.
4. To create a search:
   - In Schema, select your workspace
   - In Type, select **Table**
   - In Search, type **DEMO**
   - Click **Go**

   All tables having names that contain the string `DEMO` appear.
5. To view table details, click the **View** icon adjacent to the appropriate table name.

   The Object Detail page appears.
6. Optionally, select a task from Tasks list on the right side of the page.

Part II

Using Oracle HTML DB

Part II describes how to use Data Workshop, SQL Workshop, and Application Builder to develop database-driven applications.

Part II contains the following chapters:

4

Managing Data with Data Workshop

This section describes how to use Data Workshop to import data and export data using a Web Browser.

This section contains the following topics:

- About Data Workshop
- Importing Data
- Exporting Data

See Also:

- "What is Oracle HTML DB?"
- "Quick Start"

About Data Workshop

Oracle HTML DB renders information stored in an Oracle database to create a collection of database-driven Web pages called an **application**. Using Data Workshop you can import data into and export data from the hosted database. Supported import formats include:

- Text such as comma or tab delimited data
- XML documents
- Spreadsheets

Supported export formats include:

- Text such as comma or tab delimited data
- XML documents

To access Data Workshop:

1. Click the **Data Workshop** icon on the Workspace home page. (See Figure 0–1.)
2. Under Data Import and Data Export, click the appropriate link.

See Also:

- "Importing a Text File" on page 4-2
- "Importing an XML Document" on page 4-2
- "Importing Spreadsheet Data" on page 4-2
- "Exporting to a Text File" on page 4-3
- "Exporting to an XML Document" on page 4-3

Importing Data

You can use Data Workshop to import text files, XML documents, and data stored in a spreadsheet into an Oracle database.

Topics in this section include:

- Importing a Text File
- Importing an XML Document
- Importing Spreadsheet Data

Importing a Text File

For files less than 30KB, you can copy and paste tab delimited data directly into the Import Text Wizard. For files larger than 30KB, you must upload a separate file.

To load a text file:

1. Click **Data Workshop** on the Workspace home page.
2. Under Data Import, click **Import Text Data**.

   The Data Import Wizard appears.
3. Under Import To, select either **Existing table** or **New table**.
4. Under Import from, select either **Upload file** or **Copy and paste**.
5. Follow the on-screen instructions.

Importing an XML Document

Data Workshop supports the import of XML documents adhering to the Canonical XML specification.

To import an XML document:

1. Click **Data Workshop** on the Workspace home page.
2. Under Data Import, click **XML Import**.

   The Import XML Wizard appears.
3. Follow the on-screen instructions.

Importing Spreadsheet Data

You can load spreadsheet data by either copying and pasting text, or by importing a file. To copy and paste text, the spreadsheet file must be less than 30KB. For files larger than 30KB, you can import the file in a delimited format (such as comma delimited

(.csv) or tab delimited), upload the file, and then load the data into a new or existing table.

To import spreadsheet data:

1. Click **Data Workshop** on the Workspace home page.
2. Under Data Import, click **Import Spreadsheet Data**.

   The Import Spreadsheet Data Wizard appears.
3. Under Import to, select either **Existing table** or **New table**.
4. Under Import from, select either **Upload file** or **Copy and paste**.
5. Follow the on-screen instructions.

Exporting Data

You can also use Data Workshop to export the contents of a table to a text file or XML document.

Topics in this section include:

- Exporting to a Text File
- Exporting to an XML Document

Exporting to a Text File

Use the Export Text Data Wizard to export the contents of a table to a text file. For example, you could export an entire table to a comma delimited file (.csv).

To export a table to a text file:

1. Click **Data Workshop** on the Workspace home page.
2. Under Data Export, click **Export Text Data**.

   The Export Text Data Wizard appears.
3. Follow the on-screen instructions.

You select the schema and choose the table and columns to be exported. You can also specify the type of separator to be used to separate column values as well as whether column text strings are identified using single or double quotation marks.

Exporting to an XML Document

Use the Export XML Wizard to export the contents of a table to an XML document adhering to the Canonical XML specification.

To export a table to an XML document:

1. Click **Data Workshop** on the Workspace home page.
2. Under Data Import, click **XML Export**.

   The Export XML Wizard appears.
3. Follow the on-screen instructions.

You select the schema and choose the table and columns to be exported.

5

Managing Database Objects with SQL Workshop

This section provides information on how to use SQL Workshop to view and manage database objects, create and manage user interface defaults, and browse the data dictionary.

This section contains the following topics:

- About SQL Workshop
- Viewing Database Objects
- Managing Database Objects
- Managing User Interface Defaults
- Browsing the Data Dictionary

See Also:

- "What is Oracle HTML DB?"
- "Quick Start"

About SQL Workshop

You can use SQL Workshop to view and manage database objects from a Web browser.

To access the SQL Workshop home page, click the **SQL Workshop** icon on the Workspace home page. (See Figure 5–1.)

Figure 5–1 SQL Workshop Icon

The SQL Workshop home page is divided into four primary sections:

- **SQL Workshop**. Offers quick access to the SQL Command Processor, User Interface Defaults, and Create Database Object wizards.
- **Database Browser.** Offers a view of existing database objects by type.

- **SQL Scripts**. Manage uploaded script files and control files as well as generate DDL (data definition language) statements.
- **Task list**. Available options include:
  - **Manage Recycle Bin**. View and restore dropped database objects.
  - **View SQL History**. View recent commands and scripts run in the SQL Command Processor.
  - **Manage SQL Archive**. Save frequently used SQL commands in the SQL Archive.
  - **Drop Database Object**. Drop database objects using the Drop Database Object Wizard.
  - **Explain Plan**. View the "plan" the Oracle Optimizer uses to run your SQL command.
  - **Query Data Dictionary**. Browse the Oracle data dictionary.

About Transaction Support

Oracle HTML DB is a browser-based development environment which communicates over HTTP. Because HTTP is a stateless protocol, any command you issue using SQL Workshop is automatically followed by a database COMMIT. There is no support for transactions that span multiple pages in the SQL Workshop. For example, you cannot issue an `UPDATE` statement on one page in the SQL Workshop and then revert it on a subsequent page using a `ROLLBACK` command.

Since the commands `COMMIT`, `ROLLBACK` and `SAVEPOINT` are executed as one transaction, you can include these commands in SQL Workshop by using scripts.

> **See Also:** "Using the SQL Script Repository" on page 5-6 for information on running scripts

About Unsupported SQL*Plus Commands

SQL Workshop does not support SQL*Plus commands. If you attempt to enter a SQL*Plus command in SQL Workshop an error message displays. The following are examples of unsupported SQL*Plus commands:

```
SET ECHO OFF
SET ECHO ON
SET VERIFY ON
SET LONG 600
COLUMN dummy NOPRINT
COLUMN name FORMAT A20
DEFINE
ACCEPT
PROMPT
REMARK
SHOW
```

Viewing Database Objects

You can use SQL Workshop to view database objects. For example, you can view details about database objects by querying the Oracle dictionary. You can also run SQL commands and SQL scripts in the SQL Command Processor or view database objects in the Database Browser.

Topics in this section include:

- Using the SQL Command Processor
- Using Explain Plan
- Viewing Database Objects
- Creating Database Objects

Using the SQL Command Processor

You use the SQL Command Processor to run SQL commands and SQL scripts on any Oracle database schema for which you have privileges.

To access the SQL Command Processor:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, click **SQL Command Processor**.

   The SQL Command Processor appears.
3. Select a schema from the list and follow the on-screen instructions.
4. To run entered commands, click **Run SQL**.

   For example, to describe a database object you would type `DESC MY_OBJECT`. For example:

   ```
   DESC DEMO_ABOUT
   ```

   Alternatively, if you are using Internet Explorer, you can also press `CTRL+Enter` to execute your SQL Statement.
5. To save entered commands, click **Save**.

> **See Also:** "Accessing Saved Commands in the SQL Archive" on page 5-9 for more information on viewing saved commands and queries

About Command Termination

You can terminate commands in the Command Processor using either a semicolon (;) or forward slash (/). Consider the following examples:

```
INSERT INTO emp
      (50,'John Doe','Developer',10,SYSDATE,1000,10);

INSERT INTO emp
      (50,'John Doe','Developer',10,SYSDATE,1000,10)
/
```

The first example demonstrates the use of a semicolon (;). The second example demonstrates the use of forward slash (/).

Using Explain Plan

Use Explain Plan to view the plan the Oracle Optimizer uses to run your SQL Command.

To view the Explain Plan:

1. Click **SQL Workshop** on the Workspace home page.

2. From the Tasks list on right side of the page, select **Explain Plan**.

   Explain Plan appears.

3. Enter a command in the field provided and click **Explain Plan**.

Viewing Database Objects

You can use the Database Browser to view database objects. To find a database object, select the schema you would like to view. The values available in the schema depend upon your resource privileges.

To browse database objects:

1. Click **SQL Workshop** on the Workspace home page.
2. Under Database Browser, select the type of database object you would like to view.
3. To search for an object, select a schema, an object type, type a search string in the Search field, and click **Go**. Searches are case insensitive and no wildcards or quotes are necessary.
4. To view object details, click the **View** icon adjacent to the appropriate name.
5. Optionally, select a task from the Tasks list on the right side of the page.

Querying by Example

Once you have located a specific table you can declaratively create a report to display the data in the table.

To Query by Example:

1. Click **SQL Workshop** on the Workspace home page.
2. Under Database Browser, select **Tables**.

   The Database Browser appears.

3. Click the **View** icon to view details about a specific object.

   The Object Detail appears.

4. From the Tasks list, select **Query by Example**.
5. Follow the on-screen instructions.

Managing Database Objects

You can use SQL Workshop to manage database objects. For example, you can create new database objects, manage script files and control files, or alter a table.

Topics in this section include:

- Creating Database Objects
- Dropping Database Objects
- Restoring Dropped Database Objects
- Using the SQL Script Repository
- Accessing Saved Commands in the SQL Archive
- Accessing the SQL Command History
- Generating DDL

- Managing Control Files
- Managing Tables

See Also: "Viewing Database Objects" on page 5-4

Creating Database Objects

You can create new database objects using the Create Database Object Wizard.

To create new database objects in SQL Workshop:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, select **Create Object**.

   The Create Database Object Wizard appears.
3. Select the type of database object you would like to create.
4. Follow the on-screen instructions.

Dropping Database Objects

You can drop database objects using the Drop Database Object Wizard. When you drop a table using this wizard, you also remove all related triggers and indexes.

To drop a database object:

1. Click **SQL Workshop** on the Workspace home page.
2. From the Tasks list on the right side of the page, select **Drop Database Object**.

   The Drop Database Object Wizard appears.
3. Select a schema and then an object type.
4. Follow the on-screen instructions.

Restoring Dropped Database Objects

If you are running Oracle HTML DB with Oracle Database 10g release 1 (10.1), you can use the Recycle Bin to view and restore dropped database objects. When you drop a table, the space associated with the table is not immediately removed. The Oracle database renames the table and places it and any associated objects in the Recycle Bin where it can be recovered at a later time.

Note: The Recycle Bin feature is only available if you are running Oracle HTML DB with an Oracle 10g database.

To use the Recycle Bin:

1. Click **SQL Workshop** on the Workspace home page.
2. From the Tasks list on the right side of the page, select **Manage Recycle Bin**.

   The Recycle Bin appears.
3. To search for an object, select a schema, an object type, type a search string in the Search field, and click **Go**.
4. To view object details, click the **View** icon adjacent to the appropriate name.

   On the Object Summary page, you can:

- Click **Restore Object** to restore the current object
- Click **Purge** to permanently delete the current object

To empty the Recycle Bin without viewing the objects:

1. Click **SQL Workshop** on the Workspace home page.
2. From the Tasks list on the right side of the page, select **Manage Recycle Bin**.

   The Recycle Bin appears.
3. From the Tasks list on the right side of the page, select **Purge Recycle Bin**.

Using the SQL Script Repository

You can use the SQL Script Repository to view, edit, and run uploaded script files. For example, you can upload new script files as well as create and edit your create table, create index, and create PL/SQL package scripts.

Topics in this section include:

- Managing Script Files in the SQL Script Repository
- Uploading and Creating Script Files
- About Script Termination
- Using Parameters in a Script
- Including SQL Queries in a Script
- Exporting a Script File

Managing Script Files in the SQL Script Repository

Use the SQL Script Repository to view, edit, run, and delete uploaded script files.

To view scripts in the SQL Script Repository:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Scripts, click **Scripts**.

   SQL Script Repository appears. Scripts are stored based on your workspace username.
3. To search for a script, select a username from the Show list, enter a search string in the Find field (optional), and click **Go**.

   While in the Script Repository you can:

   - Reorder a list by clicking the column heading
   - View details about a specific file by clicking the **View** icon
   - Edit a script by clicking the **Edit** icon
   - Run a script by clicking **Run** in the Actions column
   - Delete a script by selecting it and clicking **Delete Checked**
   - Upload a script by clicking **Upload**
   - Create a script by clicking **Create**

To view script details:

1. In the Script Repository, click the **View** icon.

The Files Details page appears.

2. Under View Links, you can:
   - Click **Native file format** to download the file locally
   - Click **View document as text** to view the file in your Web Browser
   - Click **Parse this script** to parse the script to run

To run a script in the Script Repository:

1. In the Actions column, click **Run**.

   The Run page appears.

   If you have parameters in your script you must define them. You can define up to ten different parameters in each script.

2. Enter a parameter name and value in the fields provided.
3. To view the script file, click **View File**.
4. To run the script file, click **Run Script**.

   The Run Results page appears displaying the number of success, failure and the elapsed time. RED indicates that errors occurred while executing the file.

5. To view the script file source, click **View Source**.
6. To run the file again, click **Run Script** again.

   Once you have run a script file, you can view a history of previous executions by clicking **Previous Runs**.

To delete a script file from the Script Repository:

1. In the Script Repository, select the script to be deleted.
2. Click **Delete Checked**.

Uploading and Creating Script Files

To upload a script file into the Script Repository:

1. In the Script Repository, click **Upload**.

   The Upload Script page appears.

2. Follow the on-screen instructions.

If the script file you upload has a valid file extension, SQL Workshop recognizes the file as a script and automatically parses it. Table 5–1 on page 5-7 describes the file extensions SQL Workshop considers to be valid for script files.

Table 5–1 Valid Script File Extensions

Extension	Description
pkh	Package headers
plb	Package bodies
sql	Scripts
con	Constraints
ind	Indexes
sqs	Sequences

Table 5–1 (Cont.) Valid Script File Extensions

Extension	Description
tab	Tables
trg	Triggers
pkb	Package bodes
pks	Package specs

To create a script file while in the Script Repository:

1. In the Script Repository, click **Create**.

   The Create Script page appears.

2. Follow the on-screen instructions.

About Script Termination

You terminate a statement in a script by adding a carriage return and a forward slash (/) at the end of each statement. Consider the following examples:

```
INSERT INTO emp values(10,'aaa')
/
INSERT INTO emp values(20,'bbb')
/
INSERT INTO emp  values(30,'ccc')
/
```

Using Parameters in a Script

You can parameterize a script using a pound sign (#) or ampersand (&). The following two examples demonstrate valid parameter syntax.

```
CREATE TABLE #OWNER#.xyz (X INT)
/
CREATE TABLE #OWNER#.abc (Y NUMBER)
/

CREATE TABLE &OWNER.xyz (X INT)
/
CREATE TABLE &OWNER.abc (Y NUMBER)
/
```

Including SQL Queries in a Script

If you include SELECT statements in a script, the script will run without errors, but the result set will not display.

Exporting a Script File

You can export SQL Script Repository scripts using the Export Import Wizard in Application Builder.

To export a script from SQL Workshop:

1. Navigate to the Workspace home page.
2. Click **Application Builder**.

3. When Application Builder appears, click **Export/Import**.

   The Export Import Wizard appears.

4. Select **Export** and click **Next**.

5. Click the **Script Files** button and follow the on-screen instructions.

If you export a UNIX format, the wizard generates a file with rows delimited by `CHR (10)` (that is, line feeds). If you export a DOS format then each row is terminated with `CHR(13)||CHR(10)` (that is, `CR LF` or carriage return line feed).

> **See Also:** "Exporting an Application and Related Files" on page 12-5

Accessing Saved Commands in the SQL Archive

When you click Save in SQL Command Processor, SQL Workshop saves entered commands and scripts to the SQL Archives.

The SQL Archives is different from the SQL Script Repository. By saving frequently used SQL commands in the SQL Archives, you can run the commands again without retyping. When you save a SQL command to the SQL Archives, the saved command does not appear in the Script Repository.

To view the SQL Archives:

1. Click **SQL Workshop** on the Workspace home page.
2. From the Tasks list on the right side of the page, select **Manage SQL Archive**.

   SQL Archives appears. Scripts are stored based on your workspace username.

3. Follow the on-screen instructions.

Accessing the SQL Command History

SQL Command History displays the 200 most recent commands and scripts run in the SQL Command Processor.

To view the SQL Command History:

1. Click **SQL Workshop** on the Workspace home page.
2. From the Tasks list on the right side of the page, select **View SQL History**.

   SQL Command History appears.

3. To run a command, click the appropriate SQL link.

   The selected SQL command or the script displays in the SQL Command Processor.

Generating DDL

If you are running Oracle HTML DB with Oracle Database 10g release 1 (10.1), you can use DDL statements to create, alter, and drop schema objects when they are no longer needed. You can also use DDL statements to grant and revoke privileges and roles, analyze table, index, or cluster information, establish auditing options, or add comments to the data dictionary.

To generate DDL statement in SQL Workshop:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Scripts, click **Generate DDL**.

The Generate DDL Wizard appears.

3. Follow the on-screen instructions.

See Also:

- *Oracle Database SQL Reference* for more information on DDL statements
- *Oracle Database Concepts* for more information on the data dictionary

Managing Control Files

Control files enable you to run a series of scripts in a predefined order. From the Control Files Repository, you can create, edit, delete, or run control files.

To access the Control Files Repository:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Scripts, click **Script Control**.

   Control Files Repository appears.
3. To search for a script, select a username from the Show list, enter a search string in the Find field (optional), and click **Go**.

   While in the Control Files Repository you can:

   - Reorder a list by clicking the column heading
   - Edit a file by clicking the **Edit** icon
   - Run a file by clicking **Run**
   - Delete a script by selecting it and clicking **Delete Checked**

To create a control file:

1. In the Control Files Repository, click **Create**.

   The Control File Create page appears.
2. Enter a name for the control file, select the script files you would like to include, and click **Create**.
3. Specify the execution order and click **Done**.

To edit a control file:

1. In the Control Files Repository, click the **Edit** icon.

   The Edit Control File page appears.
2. On the Edit Control File page you can:

   - Change the order in which files are executed by clicking **Edit Execution Order**
   - Add additional script files by clicking **Add More Files**
   - Delete a script by selecting it and clicking **Delete Checked**

To run a control file in the Control Files Repository:

1. In the Action Column, click **Run**.

   The Run File page appears.
2. Select an Oracle Schema from the Parse As list.

If your script files include parameters you must define them. You can define up to ten different parameters in each script.

3. Enter a parameter name and value in the fields provided.

   If you wish to run the control file in the background, select **Run in Background**. Running a control file in the background means that Oracle HTML DB submits it as a job. The advantage of this approach is that you do not have to wait for it to finish to continue using SQL Workshop.

4. Click **Run File**.

   The Run Results page appears displaying the number of success, failures, and the elapsed time. RED indicates that errors occurred while executing the file.

5. To run the file again, select **Run File** again

Viewing the Control File Run History

Once you have run a file, you can view a history of previous executions by clicking **Previous Runs** on the Run File page.

To view the Control File Run History:

1. Run the control file as described in the previous procedure.
2. Select **Previous Runs**.

Viewing Control File Job Status

You can view control file job status from either the Edit File or Run File page.

To view the status of a job:

1. Run the control file as described in the previous section with the **Run in Background** option selected.
2. From the Tasks list, select **View Job Status**.

Managing Tables

You can also use SQL Workshop to create new tables or edit existing tables.

To create a new table:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, click **Create Object**.

   The Create Database Object Wizard appears.

3. Select **Table** and click **Next**.
4. Follow the on-screen instructions.

   Note: To edit an existing table, you must first navigate to it using the Database Browser.

To edit an existing table:

1. Click **SQL Workshop** on the Workspace home page.
2. Under Database Browser, select **Tables**.
3. To search for a table, select a schema, table type, type a search string in the Search field, and click **Go**.

4. To view table details, click the **View** icon.

   The Table Definition appears.

5. To edit the table or generate a report, make selection from the Tasks list. Table 5–2 describes the options available on the Tasks list.

Table 5–2 Table Tasks

Task	Description
Count Rows	Counts all rows in the currently selected table.
Create Lookup Table	Creates a lookup table for the column you select. The selected column becomes a foreign key for the lookup table.
Drop this object	Drops the current table.
Generate Create Table Script	Generates a create table script.
Insert Row	Inserts a row into the current table.
Manage Table	Displays a wizard for performing a variety of tasks on the current table, including: ▪ table (copy, drop, rename, analyze, truncate) ▪ columns (add, drop, modify, and rename) ▪ constraints (add, disable, drop) ▪ trigger (create, disable, drop, enable) ▪ index (create and drop
Popup Table Columns	Displays a popup window that lists the columns in the current table.
Query by Example	Displays a form so you can create a report and view the data in selected columns.
Report Dependent Objects	Displays a report that describes database objects that have dependencies on the selected table.
View Data Model	Displays a report that shows the referential integrity constraints for the current table.

6. To view user interface defaults, click the **Manage User Interface Defaults** button.

Managing User Interface Defaults

User interface defaults enable developers to assign default user interface properties to a table, column, or view within a specified schema. When a developer creates a form or report using a wizard, the wizard uses this information to create default values for region and item properties.

Since user interface defaults are associated with a table, you can use them with applications created using the form and report wizards.

Topics in this section include:

- Viewing Tables Utilizing User Interface Defaults
- Editing a Column Attributes
- Applying User Interface Defaults to a Table or View
- Comparing User Interface Defaults Across Applications
- About Exporting and Importing User Interface Defaults

See Also: "Application Builder Concepts" on page 6-1 and "Using Application Builder" on page 7-1 for more information on regions and item properties

Viewing Tables Utilizing User Interface Defaults

To view tables that utilize user interface defaults:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, select **User Interface Defaults**.

   A list of tables for which user interface defaults are defined displays.
3. Select a specific table by clicking the **Edit** icon adjacent to the table name.

Editing a Column Attributes

You define user interface defaults for a specific column by editing column attributes on the Column Definition page.

To edit column attributes:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, select **User Interface Defaults**.

   A list of tables for which user interface defaults are defined displays.
3. Select a specific table by clicking the **Edit** icon adjacent to the table name.

   The View Details page appears. The following information displays at the top of the page:

   - **Schema** is the schema that owns the table.
   - **Table/View Name** is the name of the selected table or view.
   - **Report Region Title** and **Form Region Title** becomes the default title for all report or form regions. Both of these names a modified versions of Table/View Name in which the first letter is capitalized and any underscores are replaced with spaces.

   Column-level User Interface Defaults appear next. You can edit select attributes for all displayed columns, by clicking **Grid Edit**.
4. To select a specific column, click the **Edit** icon adjacent to the column name.

Editing Column-level Defaults

Edit Column-level Defaults is divided into two pages: Column Definition and List of Values. The topics that follow describes how to edit specific attributes on the Column Definition and List of Values pages.

Edit Column-level User Interface Defaults The top of the page displays the selected schema, table or view name, and column name.

Default for Label (used in Reports and Forms) Use Label to specify default label text for items in a form and the heading for columns in reports.

Defaults for Reports Available Display for Reports attributes include:

- **Display** - Indicates whether the column displays in a report. The default is **Yes**.

- **Display Sequence** - Specifies the display sequence of items in a report. The default value is based on the column ID, which is based on the order of the columns in the table.
- **Display As** - Specifies how the column should be displayed in reports
- **Mask** - Indicates if a mask should be applied against the data. This attribute is not applicable for character- based items.
- **Alignment** - Specifies report alignment (Left, Center, or Right). If the column is a number, the default is **Right**. Otherwise, the default is **Left**.
- **Searchable** - Indicates whether the column should be searchable in reports. If the column is VARCHAR2 or CHAR, the default is **Yes**. If not, the default is **No**.
- **Group By** - Indicates whether the column should be used for Group By and then the sequence of the grouping. The default is **Yes**.
- **Aggregate By** - Indicates whether the column should be used for aggregation in reports and charts.

Defaults for Tabular Forms Use **Display As** to specify how an item should display in a tabular form.

Defaults for Forms Available Defaults for Forms attributes include:

- **Display** - Indicates whether the column displays in a form. The default is **Yes**.
- **Display Sequence** - Specifies the sequence of items in a form. The default is based on the column ID, which is based on the order of the columns in the table.
- **Display As** - Indicates how items in a form display. The default selection is **Text Field**.
- **Mask** - Indicates a mask to be applied against the data in a form. Not used for character-based items.
- **Default Value** - Specifies the default value associated with this column.
- **Width** - Specifies the display width.
- **maxWidth** - Specifies the maximum string length a user is allowed to enter in this item.
- **Height** - Specifies the display height of an item.
- **Required** - Used to generate a validation in which the resulting item must not be null. If resulting item is not null, select **Yes**.
- **Help Text** - Becomes Item help. By default, this text is pulled from the column hint (if applicable).

List of Values To access List of Values defaults, click the **List of Values** tab. Once you select the type, you are prompted to enter either Display Value, Return Value pairs, or a list of values query.

The top of the page displays the selected schema, table or view name, and column name. Use the List of Values Type list to specify whether the selected column will include a static or dynamic list of values.

Viewing the Column Definition Report

You can view details about a specific column by accessing the Column Definition report. The Column Definition report displays the data type, data length, nullable,

default, comment as well as any check constraints, primary and unique keys, and foreign keys that reference the column.

To view the Column Definition report:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, select **User Interface Defaults**.
3. Select a table by clicking the **Edit** icon adjacent to the table name.
4. Select a column.
5. From the Tasks list, select **View Column Definition**.

Applying User Interface Defaults to a Table or View

You can view a listing of tables without user interface defaults on the Tables without Defaults page.

To apply user interface defaults to a table or view:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, select **User Interface Defaults**.
3. From the Tasks list, select **Tables without User Interface Defaults**.
4. To apply user interface defaults to a table or view, select the **Default** link to the left of the table or view name.

Comparing User Interface Defaults Across Applications

Use the Compare Defaults report to monitor consistency in user interface design across all pages in a single application or multiple applications in the current workspace. Running the Compare Defaults report compares currently defined user interface defaults (or column attributes) against the column attributes set for forms, reports, and tabular forms.

See Also: "Editing a Column Attributes" on page 5-13

To run the Compare Defaults report:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, select **User Interface Defaults**.
3. From the Tasks list, select **Compare User Interface Defaults**.

   The Compare Defaults page appears.
4. Select a schema, table name, and column.
5. From Display, select the attributes you want to compare.
6. Optionally select an application.
7. Click **Go**.

About Exporting and Importing User Interface Defaults

You export user interface defaults in the same way you export any related application file. Exporting user interface defaults from one Oracle HTML DB development instance to another involves the following steps:

- Export the user interface defaults using the Export User Interface Defaults utility
- Import the exported file into the target Oracle HTML DB instance
- Install the exported file from Export Repository

When you export user interface defaults, all user interface defaults for the selected schema are exported to a single script. The file contains an API call to create table hints by making calls to the application PL/SQL API. You can use this file to import user interface defaults to another database and Oracle HTML DB instance.

> **See Also:** "Exporting User Interface Defaults" on page 12-9 and "Importing Export Files" on page 12-10

Browsing the Data Dictionary

Each Oracle database has a data dictionary. An Oracle data dictionary is a set of tables and views that are used as a read-only reference about the database. For example, a data dictionary stores information about both the logical and physical structure of the database. A data dictionary also stores information about valid Oracle database users, integrity constraints for tables in the database, and the amount of space allocated for a schema object as well as how much of it is being used.

To browse the data dictionary:

1. Click **SQL Workshop** on the Workspace home page.
2. From the Tasks list, select **Query Data Dictionary**.

   The Data Dictionary Browser appears.
3. Click the **View** icon to display the Query By Example (QBE) form. Use this form to query the Oracle data dictionary for details about database objects.

> **See Also:** *Oracle Database Concepts* for more information on the data dictionary

To view data dictionary reports:

1. Click **SQL Workshop** on the Workspace home page.
2. From the Tasks list, select **Query Data Dictionary**.

   The Data Dictionary Browser appears.
3. From the Tasks list, select **Data Dictionary Reports**.

6

Application Builder Concepts

This section provides basic conceptual information about Application Builder. Application Builder is the core component within Oracle HTML DB that enables you to build database centric Web applications.

This section contains the following topics:

- About the Workspace Home Page
- What is Application Builder?
- What is a Page?
- How Page Processing and Page Rendering Work
- Understanding Session State Management
- Managing Session State Values
- Understanding URL Syntax
- Using Substitution Strings

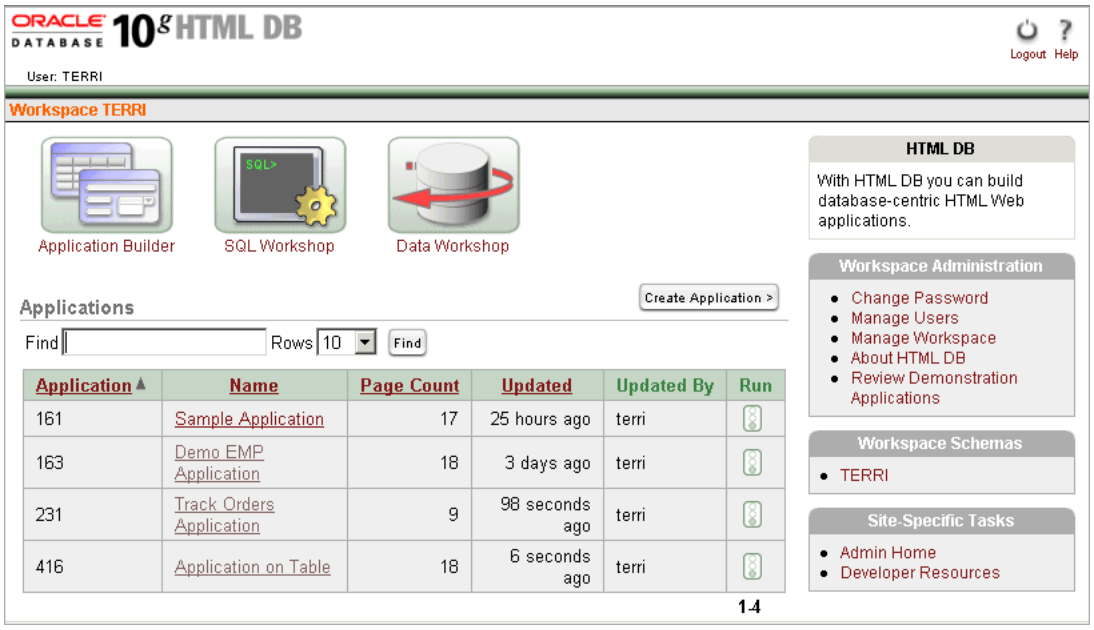
See Also:

- "What is Oracle HTML DB?"
- "Using Application Builder"
- "Building an Application"

About the Workspace Home Page

When you log in to Oracle HTML DB, a Workspace home page appears. A workspace is shared work area within the Oracle HTML DB development environment where multiple developers can create applications.

Figure 6–1 Workspace Home Page



As shown in Figure 6–1, your user name displays in the upper left corner of the page, directly above the workspace name. Three large icons (Application Builder, SQL Workshop, Data Workshop) appear next followed by a list of current applications.

The right side of the page contains two lists:

- **Workspace Administration.** Contains links to tools and reports designed to help workspace administrators manage the current workspace.
- **Workspace Schemas.** Contains links details about the schemas accessible to the current workspace. Each workspace has access to one or more database schemas.

> **See Also:** "About the Workspace Administration List" on page 13-2 for more information on using the Workspace Administration list

What is Application Builder?

In Oracle HTML DB you use Application Builder to build dynamically rendered applications. Each application is a collection of pages linked together using tabs, buttons, or hypertext links.

To access Application Builder:

1. Log in to Oracle HTML DB.

   The Workspace home page appears.

2. To access Application Builder, you can either:

   - Select an application name from the Applications list
   - Create a new application by clicking **Create Application**

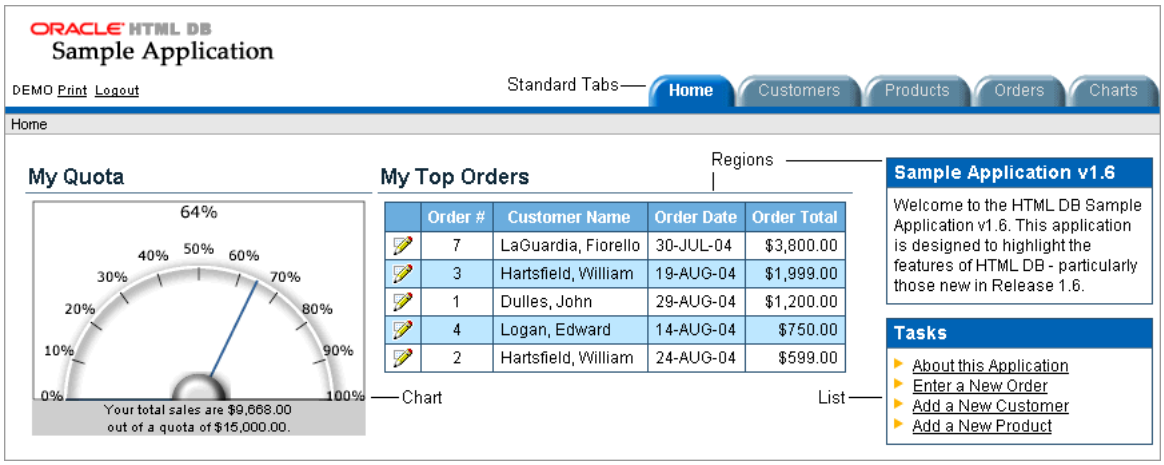
   Application Builder appears.

> **See Also:** "About the Application Builder Home Page" on page 7-2

What is a Page?

A page is the basic building block of an application. When you build an application in Application Builder, you create pages that contain user interface elements, such tabs, lists, buttons, items, and regions. Figure 6–2 illustrates the use of some of these elements in a running application.

Figure 6–2 Sample Application



You add components and layout controls to a page on the Page Definition.

To view the Page Definition of an existing page:

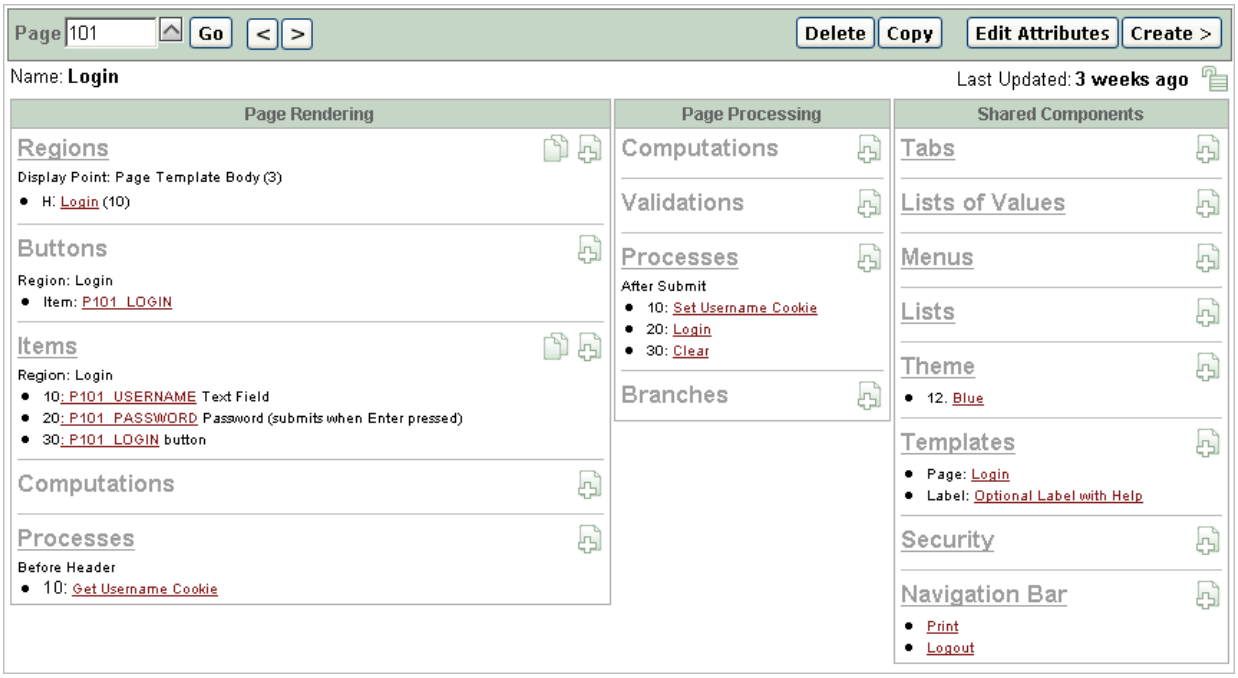
1. On the Workspace home page, select an application name from the Applications list.

   Application Builder appears. A list of pages appears at the bottom of the page.

2. To edit a specific page, select a page name, or enter the page number in the Find field and click **Go**.

   The Page Definition appears. (See Figure 6–3.)

Figure 6–3 Page Definition



The Page Definition is divided into three sections:

- Page Rendering
- Page Processing
- Shared Components

The left section, **Page Rendering**, lists user interface controls and logic execute when the page is rendered. The middle section, **Page Processing**, lists logic controls (such as computations and processes) that are evaluated and executed when the page is processed. The far right section, **Shared Components**, lists common components that display on every page within the application.

See Also:

- "Editing a Page Definition" on page 7-15
- "About Shared Components" on page 6-5

About Page Rendering

The following list briefly describes Page Rendering controls:

- **Regions**. A region is an area of a page that uses a specific template to generate HTML content. Each page can have any number of regions. You can use regions to group other controls, such as buttons and items. You can create simple regions that do not generate additional HTML, or create elaborate regions that frame content within HTML tables or images. The HTML DB engine displays regions in sequence within columns. You can choose whether a region displays conditionally.
- **Buttons**. Buttons are used to submit a page. When you submit a page, the HTML DB engine processes it, or redirects users to another page without any processing. A button can be implemented as an HTML button, an image, or by using a template.

- **Items**. Lists the items grouped by region. Items are HTML form elements such as text fields, select lists and check boxes with an associated session state.
- **Computations**. Lists the computations that are executed at the time the page is rendered. Computations are units of logic used to assign session state to items.
- **Processes**. Lists the processes that are executed at the time the page is rendered. Processes are logic controls used to execute data manipulation language (DML) or PL/SQL. For example, you can use a process to populate session state at the time the page is rendered.

See Also: "Controlling Page Layout and User Interface" on page 9-1

About Page Processing

The following list describes Page Processing controls:

- **Computations**. Lists the computations that are executed at the time the page is processed. Computations are units of logic used to assign session state to items.
- **Validations**. Enables you to create logic controls to verify whether user input is valid. For example, a validation can check whether a value has been typed into a mandatory field.
- **Processes**. Lists the processes that are executed after the page is submitted. Processes are logic controls used to execute data manipulation language (DML) or PL/SQL.
- **Branching**. Enables you to create logic controls that determine how the user navigates through the application.

About Shared Components

Shared components are common elements that can display on every page within an application. Examples of shared layout controls include:

- Lists
- Lists of Values
- Menus
- Navigation Bars
- Tabs
- Templates

Lists

A list is a collection of links that is rendered using a template. For each list entry, you specify display text, a target URL, and other attributes that control when and how the list entry displays. You control the display of the list and the appearance of all list entries by linking it to a template.

See Also: "Creating Lists" on page 10-11

Lists of Values

A list of values (LOV) is a static or dynamic definition used to display a specific type of page item, such as a radio group, check box, or select list. LOVs can be static (that is, based on a set of predefined display and return values) or dynamic (based on SQL queries that select values from tables).

You define LOVs at the application level by running the LOV Wizard and adding them to the Named List of Values repository.

> **See Also:** "Creating Lists of Values" on page 8-41

Menus

A menu is a hierarchical list of links that is rendered using a template. For example, you can display menus as a list of links or as a breadcrumb path.

> **See Also:** "Creating Menus" on page 10-7

Navigation Bars

Use navigation bars to link users to various pages within an application. Typically navigation bars are used to enable users to log in and log out or link to help text. The location of a navigation bar depends upon the associated page template. A navigation bar icon enables you to display a link from an image or text. When you create a navigation bar icon you can specify an image name, text, a display sequence, and a target location (a URL or page).

> **See Also:** "Creating a Navigation Bar" on page 10-1 for more information on creating navigation bars

Tabs

Tabs are an effective way to navigate users between pages of an application. Application Builder includes two different types of tabs: standard tabs and parent tabs

An application having only one level of tabs uses a standard tab set. A standard tab set is associated with a specific page. You can use standard tabs to link users to other pages within your application. A parent tab set functions as a container to hold a group of standard tabs. Parent tabs give users another level of navigation as well as a context (or sense of place) within the application.

> **See Also:** "Creating Tabs" on page 10-4

Templates

Templates control the look and feel of the pages in your application. As you create your application you specify templates for pages, regions, reports, lists, labels, menus, buttons, and popup list of values. Groups of templates are organized into named collections called themes.

> **See Also:** "About Themes and Templates" on page 7-29

How Page Processing and Page Rendering Work

When you create an application in Oracle HTML DB, you link pages together using tabs, buttons, or hypertext links. Each page can have buttons and items and can include application logic. You can branch from one page to the next using conditional navigation, perform calculations, validations, and display reports, calendars, and charts. You can generate reports, charts, and forms using built-in wizards, static HTML, or deliver more custom rendering with PL/SQL programming.

The HTML DB engine dynamically renders and process pages based on data stored in database tables. To view a rendered version of your application, you run or submit it to the HTML DB engine. When you run an application, the HTML DB engine relies on two processes:

- **Show Page** is the page rendering process. It assembles all the page attributes (including regions, items, and buttons) into a viewable HTML page.
- **Accept Page** takes care of page processing. It performs any computations, validations, processes, and branching.

When you call a page using a URL, the HTML DB engine is running the Show page or page rendering process. When you submit a page, the HTML DB engine saves the submitted values in the session cache and then performs any computations, validations, or processes.

Understanding Conditional Rendering and Processing

A condition is a small unit of logic that helps you control the display of regions, items, buttons, and tabs as well the execution of processes, computations, and validations. For example, when you apply a condition to a button, the rendering engine evaluates the condition during the rendering (or Show page) process. Whether the condition passes or fails determines whether the page control (such as a button) displays.

You specify a condition by selecting a condition type when you create the control or component (for example, the region, item, button, or tab) or by making a selection from the conditional display attribute. (See Figure 6–4 on page 6-7.) The condition evaluates to true or false based on the values you enter in the Expression fields.

Figure 6–4 Conditional Display Attribute

To view a complete listing of all available conditions for a given component or control click the View icon to the right of the Condition Type list. Shortcuts to common selections appear directly beneath the list. If your condition requires an expression, type it in the appropriate field.

The sections that follow offer examples of some commonly used condition types.

> **See Also:** Appendix A, "Available Conditions" on page A-1 for a detailed listing of condition types available in Oracle HTML DB

Current Page In Expression 1

Current page = Expression 1 evaluates to true if the current page matches the page number listed in the Expression 1 field. For example:

```
100,101,102
```

If the current page is 100, 101, or 102, then this condition evaluates to true and the condition passes.

Exists

Exists (SQL query returns at least one row) is expressed as a SQL query. If the query returns at least one row, then the condition evaluates as true. For example:

```
SELECT 1 FROM emp WHERE deptno = :P101_DEPTNO
```

This example references item `P101_DEPTNO` as a bind variable. You can use bind variables within application processes and SQL query regions to reference item session state. If one or more employees are in the department identified by the value of `P101_DEPTNO`, then the condition evaluates as true.

> **See Also:** "About Bind Variables" on page 6-13 for more information

PLSQL Expression

Use **PL/SQL Expression** to specify an expression in valid PL/SQL syntax that evaluates to true or false. For example:

```
NVL(:MY_ITEM,'NO') = 'YES'
```

If the value of `:MY_ITEM` is YES, then the condition evaluates as true. Otherwise it evaluates as false.

Verifying User Identity

Authentication is the process of establishing users' identities before they can access an application. Authentication may require a user enter a username and password or may involve the use of a digital certificate or a secure key.

Oracle HTML DB supports modular authentication, making it easy to switch authentication methods when needed. You can establish a user's identity by selecting from a number of built-in authentication methods, or by using a wizard to create your own custom authentication approach.

> **See Also:** "Establishing User Identity Through Authentication" on page 14-3 for more information

Controlling Access to Controls and Components

While conditions control the rendering and processing of specific controls or components on a page, authorization schemes control user access. Authorization is a broad term for controlling access to resources based on user privileges.

Authorization schemes extend the security of your application's authentication scheme. You can specify an authorization scheme for an entire application, a page, or specific page control such as a region, item, or button. For example, you could use an authorization scheme to selectively determine which tabs, regions, or navigations bars a user sees.

> **See Also:** "Providing Security Through Authorization" on page 14-9

Understanding Session State Management

HTTP, the protocol over which HTML pages are most often delivered, is a stateless protocol. A Web browser is only connected to the server for as long as it takes to download a complete page. In addition, each page request is treated by the server as

an independent event, unrelated to any page requests that have happened previously or may occur in the future. This means that to access form values entered on one page on a subsequent page, some form of session state management needs to occur. Typically, when a user enters values into a form on one page, those values are not accessible on later pages. Oracle HTML DB transparently maintains session state and provides developers with the ability to get and set session state values from any page in the application.

A **session** is a logical construct that establishes persistence (or stateful behavior) across page views. Each session is assigned a unique identifier within the Oracle HTML DB installation. The HTML DB engine uses this identifier (or session ID) to store and retrieve an application's working set of data (or session state) before and after each page view.

Because sessions are entirely independent of one another, any number of sessions can exist in the database at the same time. Since sessions persist in the database until purged by an administrator, a user can return to an old session and continue running an application long after first launching it. A user can also run multiple instances of an application simultaneously in different browser sessions.

Oracle HTML DB sessions are logically and physically distinct from the Oracle database sessions used to service page requests. A user runs an application in a single Oracle HTML DB session from log in to log out with a typical duration measured in minutes or hours. Each page requested during that session results in the HTML DB engine creating or reusing an Oracle database session to access database resources. Each of these sessions lasts just a fraction of a second.

Understanding Session IDs

The HTML DB engine establishes the identity (or anonymity) of the user for each page request and the session ID in order to fetch session state from the database. The most visible location of the session ID is in the URL for a page request. Another visible location is in the page's HTML POST structures or in a session cookie sent by the HTML DB engine during authentication and maintained for the life of the application (or browser) session.

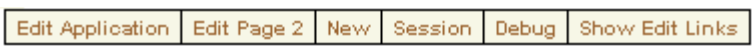
Oracle HTML DB assigns new session IDs during authentication processing, records the authenticated user's identity with the session ID, and continually checks the session ID in each page request's URL or POST data with the session cookie and the session record in the database. These checks provide users with both flexibility and security.

While the session ID is the key to session state, the session cookie (where applicable) and the session record safeguard the integrity of the session ID and the authentication status of the user.

Viewing Session State

The behavior of an HTML DB application is usually driven by values in session state. For example, a button may display conditionally based on the value of an item session state. You can view the session state for a page by clicking **Session** on the Developer toolbar.

Figure 6–5 Developer Toolbar



The Session State page provides valuable information about the page. Table 6–1 describes the various types of information available on the Session State page.

Table 6–1 Information Available on the Session State Page

Heading	Description
Application Environment	Identifies the session ID, current user, security ID, and browser language.
Application Items	Application items are items that do not reside on a page. Application items are session state variables without the associated user interface properties. **See Also:** "Using Substitution Strings" on page 7-15 for more information on referencing item values
Application, Page, Session	Identifies the application name, page number, and session ID.
Page Items	Identify attributes of the page, including the item name, how the item displays (hidden, popup, button, display only HTML), the state or session ID, and status. The Status column indicates the status of the session state. Available values include: ■ I - Inserted ■ U - Updated ■ R - Reset
Session State	Indicates the user's entire session state. The section at the top indicates the state for the current page.

See Also: "Using the Developer Toolbar" on page 7-14 for more information about the Developer toolbar

Managing Session State Values

When building interactive, data driven Web applications, the ability to access and manage session state values with ease is critical. In Oracle HTML DB, session state is automatically managed for every page and easily referenced in static HTML or logic controls such as processes or validations.

Topics in this section include:

- Referencing Session State
- Setting Session State
- Clearing Session State
- About Bind Variables

See Also: "Items" on page 7-21 and "Referencing Item Values" on page 8-38 for more information

Referencing Session State

Referencing the value of an item is one of the most common examples of referencing session state. In Oracle HTML DB, an item can be a field, a text area, a password, a select list, or check box. Table 6–2 describes the supported syntax for referencing *`item values`*.

Table 6–2 Syntax for Referencing Item Values

Type	Syntax	Description
SQL	`:MY_ITEM`	Standard bind variable syntax for items no longer than 30 characters. Use this syntax for references within a SQL query and within PL/SQL.
PL/SQL	`V('MY_ITEM')`	PL/SQL syntax referencing the item value using the `V` function. **See Also:** "Oracle HTML DB APIs" on page 17-1
PL/SQL	`NV('MY_NUMERIC_ITEM')`	Standard PL/SQL syntax referencing the numeric item value using the `NV` function. **See Also:** "Oracle HTML DB APIs" on page 17-1
Static Text (exact)	`&MY_ITEM.`	Static text. Exact Substitution.

Setting Session State

When a user submits a page in Oracle HTML DB, the HTML DB engine automatically stores values typed into fields (items) in session state. For example, suppose you have an application containing two pages. The first page of the application contains a form in which a user can enter a phone number. You have defined this form by creating an item named *P2_PhoneNo*. On the second page you want to display the information the user enters in the form.

When the page is submitted, Oracle HTML DB captures the value typed in the phone number field and stores the value for future use. The phone number typed by the user can then be retrieved from session state by referencing the item associated with the field on the page.

Clearing Session State

As you develop your applications, you may find it useful to clear the cached value for specific items, all items on a page, all pages in an application, or the current user session. Clearing a cached value resets the value to null. The topics that follow offer specific examples of clearing session state.

Clearing Cache by Item

Clearing cache for a single item resets the value of the item to null. For example, you might use this approach to make sure a specific item's value is null when a page is prepared for rendering.

The following example uses standard `f?p` syntax to clear the cache for an item. This example calls page 5 of application 100. Placing `MY_ITEM` in the `ClearCache` position of the `f?p` syntax resets the value of `MY_ITEM` to `NULL`.

```
f?p=100:5:&SESSION.::NO:MY_ITEM
```

The following example resets the value of the items *THE_EMPNO* and *THE_DEPTNO*.

```
f?p=100:5:&SESSION.::NO:THE_EMPNO,THE_DEPTNO
```

Clearing Cache by Page

Caching application items provides a very effective way to maintain session state. However, there are occasions when you may want to clear the cache for all items on a page. For example, suppose you needed to clear all fields on page when a user clicks a

link the creates a new order. By clearing cache for an entire page you set the value of all items on the page to null.

Clearing Session Cache for Two Pages While Resetting Pagination This example clears the session cache for two pages and resets pagination.

```
f?p=6000:6003:&SESSION.::NO:RP,6004,6014
```

This example:

- Runs page 6003 of application 6000 and uses the current session ID
- Indicates to not show debug information (`NO`)
- Clears all values maintained by the current session's cache for items of pages 6004 and 6014
- Resets region pagination (`RP`) on page 6003 (the requested page)

> **See Also:** Controlling Report Pagination on page 8-10

Clearing Session Cache on a Page and Passing an Item Value This example demonstrates a good way to implement an update form. It clears existing information and sets the item's value (typically a primary key).

```
f?p=6000:6003:&SESSION.::NO:6003:MY_ITEM:1234
```

This example:

- Runs page 6003 of application 6000 and use the current session ID
- Indicates to not show debug information (`NO`)
- Clears all values maintained by the current session's cache for items on page 6003
- Sets the session state of an item called `MY_ITEM` to the value `1234`

Clearing Session Cache on a Page and Passing Values to Multiple Items This example is similar to the previous one except it passes values to multiple items.

```
f?p=6000:6004:&SESSION.::NO:6003:MY_ITEM1,MY_ITEM2,MY_ITEM3:1234,,5678
```

This example:

- Runs page 6004 of application 6000 and use the current session ID
- Clears the current session's cache for items on page 6003
- Indicates debug information should be hidden (`NO`)
- Sets the value of `MY_ITEM1` to 1234, sets the value of `MY_ITEM2` to null (indicated by the comma used as placeholder), and sets the value of `MY_ITEM3` to 5678

Clearing Cache for an Entire Application

You can also clear application cache by using `f?p` syntax by creating a `REQUEST` argument using the keyword *APP* using the following syntax:

```
f?p=App:Page:Session::NO:APP
```

Note: Resetting the cache for an entire application does not actually restore the application to a completely reset state. For example, if an application includes on-new instance computations or on-new instance processes, the HTML DB engine runs these computations and processes when the application session is created. Then, it processes the clear cache request and displays the requested page.

The only way to reset the application completely without a session ID (if no cookie is used to track the session ID), is to request it using a URL without a session ID, or by calling `HTMLDB_APPLICATION.CLEAR_APP_CACHE` from another application. If the session ID is tracked using a cookie, you will need to logout in order to reset the state.

Clearing Cache for the Current User Session

You can also clear application cache by using `f?p` syntax. Create a `REQUEST` argument using the keyword `SESSION`. For example:

```
f?p=6000:6004:12507785108488427528::NO:SESSION
```

About Bind Variables

You can use bind variables within an application process or SQL query to reference session state of a specified item. For example:

```
SELECT * FROM emp WHERE name like '%' || :SEARCH_STRING || '%'
```

In this example, the search string is a page item. If the region type is defined as SQL Query, you can reference the value using standard SQL bind variable syntax. Using bind variables ensures that parsed representations of SQL queries are reused by the database, optimizing memory usage by the server.

When using bind variable syntax remember the following rules:

- Bind variable names must correspond to an item name
- Bind variable names are not case-sensitive
- Bind variable names cannot be longer than 30 characters (that is, they must be a valid Oracle identifier)

  Although application items can be up to 255 characters, if you intend to use an application item within SQL using bind variable syntax, the item name must be 30 characters or less.

Using Bind Variables in Regions Based on a SQL Query or LOV

If your region type is defined as a SQL Query, SQL Query (plsql function body returning SQL query), or list of values (LOV) you can reference session state using the syntax:

```
:MY_ITEM
```

One common way to do this is to incorporate a session state variable in a WHERE clause. The following example demonstrates how to bind the value of the item `THE_DEPTNO` into a region defined from a SQL Query.

```
SELECT ename, job, sal
FROM emp
WHERE deptno = :THE_DEPTNO
```

> **See Also:** "Customizing Regions" on page 9-3 for more information on creating different types of regions

Using Bind Variables in PL/SQL Procedures

For region types defined as a PL/SQL Procedure, regions are constructed using PL/SQL anonymous block syntax. In other words, the beginning and ending are added automatically around the PL/SQL. For example:

```
INSERT INTO emp (empno, ename, job)
VALUES (:P1_empno, :P1_name, :P1_job);
```

In this example, the values of the `empno`, `ename`, and `job` are populated by the values of `P1_empno`, `P1_name`, and `P1_job`.

Understanding URL Syntax

Each application has a number (called an application ID) or alphanumeric alias which uniquely identifies it. Similarly, each page also has a unique number (called a page number) or an alphanumeric alias. When you run an application, the HTML DB engine generates a session number that serves as a key to the user's session state.

The URL that displays for each page indicates the location of Oracle HTML DB and identifies the application ID, page number, and session ID. For example:

```
http://htmldb.oracle.com/pls/otn/f?p=4350:1:220883407765693447
```

This example indicates that:

- The address of Oracle HTML DB is:

  ```
  http://htmldb.oracle.com/pls/otn/
  ```

- The application ID is `4350`
- The page number is `1`
- The session ID is `220883407765693447`

Using f?p Syntax to Link Pages

You can create links between pages in your application using the following syntax:

```
f?p=App:Page:Session:Request:Debug:ClearCache:itemNames:itemValues:PrinterFriendly
```

Table 6–3 describes the possible arguments you can pass when using `f?p` syntax.

Table 6–3 f?p Syntax Arguments

Syntax	Description
`App`	Indicates an application ID or alphanumeric alias.
`Page`	Indicates a page number or alphanumeric alias.

Table 6–3 (Cont.) f?p Syntax Arguments

Syntax	Description
`Session`	Identifies a session ID. You can reference a session ID to create hypertext links to other pages that maintain the same session state by passing the session number. You can reference the session ID using the syntax: ■ Short substitution string: `&SESSION.` ■ PL/SQL: `V('SESSION')` ■ Bind variable: `:APP_SESSION`
`Request`	Sets the value of `REQUEST`. Each application button sets the value of `REQUEST` to the name of the button. This enables accept processing to reference the name of the button when a user clicks it. You can reference `REQUEST` using the syntax: ■ Substitution string: `&REQUEST.` ■ PL/SQL: `V('REQUEST')` ■ Bind variable: `:REQUEST`
`Debug`	Displays application processing details. Valid values for the DEBUG flag are `YES` or `NO`. Setting this flag to `YES` displays details about application processing. You can reference the Debug flag using the following syntax: ■ Short substitution string: `&DEBUG.` ■ PL/SQL: `V('DEBUG')` ■ Bind variable: `:DEBUG`
`ClearCache`	Clears cache. Clearing cache for a single item simply sets the value of the list of names to null. To clear cached items, use a comma delimited list of page numbers. Comma delimited lists can also contain collections to be reset or the keyword `RP`, which resets region pagination on the requested page.
`itemNames`	Comma delimited list of item names used to set session state with a URL.
`itemValues`	List of item values used to set session state within a URL. Item values may not include colons, but may contain commas if enclosed with baskslashes. To pass a comma in an item value, enclose the characters with backslashes. For example: `\123,45\`
`PrinterFriendly`	Determines whether the page is being rendered in printer friendly mode. If PrinterFriendly is set to Yes, then the page is rendered in printer friendly mode. The value of PrinterFriendly can be used in rendering conditions to remove elements such as regions from the page to optimize printed output.You can reference the printer friendly preference by using the syntax: `V('PRINTER_FRIENDLY')` When referenced, the HTML DB engine will not display tabs or navigation bars and all items will be displayed as text and not as form elements.

Although it is important to understand how `f?p` syntax works, you rarely have to construct this syntax yourself. Oracle HTML DB includes many wizards that automatically create these references for you. The sections that follow describe a number of specific instances that utilize `f?p` syntax to link pages.

Calling a Page Using an Application and Page Alias

Application and page aliases must consist of valid Oracle identifiers, cannot contain any whitespace, and are not case-sensitive. The following example calls a page using an application and a page alias from within an Oracle HTML DB application. It runs the page *home* of the application *myapp* and uses the current session ID.

```
f?p=myapp:home:&SESSION.
```

Application aliases must be unique within a workspace. If applications in different workspaces within the same Oracle HTML DB instance have the same application alias, use the &C argument to specify the workspace name. For example:

```
f?p=common_alias:home:&session.&c=WORKSPACE_A
```

Calling a Page from a Button URL

When you create a button, you can specify a URL to redirect to when the user clicks the button. This example runs page 6001 of application 6000 and uses the current session ID.

```
f?p=6000:6001:&SESSION.
```

Note that this is only one approach to using a button in Oracle HTML DB. This method bypasses page submission and acts as a hyperlink on the page. Another method is to submit the page first. In that approach, clicking the button submits the page for processing, allowing forms to be submitted and session state to be saved.

See Also: "Creating Buttons" on page 8-32

Using Substitution Strings

You can use substitution strings within a page template or region source to replace a character string with another value. As you design your application and enable users to edit items, you will need to use substitution strings in order to pass information.

You can use substitution strings in Oracle HTML DB in the following ways.

- Include a substitution string within a template
- Create an item at the application or page level
- Use built-in substitution strings to achieve a specific type of functionality

Substitution strings used within a template contain the pound symbol (#). For example:

```
#ABC#
```

Substitution strings created at the application level do not display, but are used as variables to maintain session state. You can define page items as an attribute of a page. You can use this type of session state substitution at the application or page level. For example:

```
&MY_ITEM.
```

Built-in Substitution Strings

Application Builder supports a number of built-in substitution strings. You may need to reference these values in order to achieve specific types of functionality.

The sections that follow describe these substitution strings, when to use them, and what supported syntax is currently available. Note that no short bind variable exists for *SESSION* or *USER* since both are Oracle reserved words.

Topics in this section include:

- APP_ALIAS
- APP_ID
- APP_IMAGES
- APP_PAGE_ID
- APP_SESSION
- APP_UNIQUE_PAGE_ID
- APP_USER
- AUTHENTICATED_URL_PREFIX
- BROWSER_LANGUAGE
- CURRENT_PARENT_TAB_TEXT
- DEBUG
- HOME_LINK
- IMAGE_PREFIX
- HTML DB SCHEMA OWNER
- PRINTER_FRIENDLY
- LOGOUT_URL
- PROXY SERVER
- PUBLIC_URL_PREFIX
- REQUEST
- SQLERRM
- SYSDATE_YYYYMMDD
- WORKSPACE_IMAGES

See Also:

- "Static Substitution Strings" on page 7-9 for more information on defining static substitution strings as an application attribute
- "Providing Security Through Authorization" on page 14-9 for more information on authentication

APP_ALIAS

APP_ALIAS is alphanumeric name for the current application. APP_ALIAS is different from the APP_ID in that the APP_ID must be unique over all companies and all applications hosted in one database. In contrast, APP_ALIAS must be unique within a workspace. Using the same APP_ALIAS you can create an application called ABC in two workspaces. You can use APP_ALIAS almost anywhere a APP_ID can be used. For example, f?p Syntax can use an APP_ALIAS or an application ID as demonstrated in this example:

```
f?p=ABC:1:&SESSION.
```

This example runs application ABC, page 1 using the current session.

Table 6–4 describes the supported syntax for referencing APP_ALIAS.

Table 6–4 APP_ALIAS Syntax

Reference Type	Syntax
Bind variable	`:APP_ALIAS`
PL/SQL	`V('APP_ALIAS')`
Substitution string	`&APP_ALIAS.`

The following is an HTML example:

```
Click me to go to page 1 <a href="f?p=&APP_ALIAS.:1:&SESSION."> of the current
application</a>
```

APP_ID

`APP_ID` identifies the application ID of the currently executing application. Table 6–5 describes the supported syntax for referencing `APP_ID`.

Table 6–5 APP_ID Syntax

Reference Type	Syntax
Bind variable	`:APP_ID`
Direct PL/SQL	`HTMLDB_APPLICATION.G_FLOW_ID (A NUMBER)`
PL/SQL	`NV('APP_ID')`
Substitution string	`&APP_ID.`

The following is an example of a substitution string reference:

```
f?p=&APP_ID.:40:&SESSION.
```

APP_IMAGES

Use this substitution string to reference uploaded images, JavaScript, and cascading style sheets that are specific to a given application and are not shared over many applications. If you upload a file and make it specific to an application, then you must use this substitution string, or bind variable. Table 6–6 describes the supported syntax for referencing `APP_IMAGES`.

Table 6–6 APP_IMAGES Syntax

Reference Type	Syntax
Bind variable	`:APP_IMAGES`
Direct PL/SQL	Not available.
PL/SQL	`V('APP_IMAGES')`
Substitution string	`&APP_IMAGES.`
Template Substitution	`#APP_IMAGES#`

See Also: "IMAGE_PREFIX" on page 6-22, "WORKSPACE_IMAGES" on page 6-26, and "Uploading Images" on page 9-41

APP_PAGE_ID

APP_PAGE_ID is the current application page number. For example, if your application was on page 3 then the result would be 3. Using this syntax is useful when writing application components that need to work generically in multiple applications. Table 6–7 describes the supported syntax for referencing APP_PAGE_ID.

Table 6–7 APP_PAGE_ID Syntax

Reference Type	Syntax
Bind variable	`:APP_PAGE_ID`
Direct PL/SQL	`HTMLDB_APPLICATION.G_FLOW_STEP_ID (A NUMBER)`
Direct PL/SQL	`:APP_PAGE_ID`
PL/SQL	`NV('APP_PAGE_ID')`
Substitution string	`&APP_PAGE_ID.`

The following is an example of a substitution string reference:

```
f?p=&APP_ID.:&APP_PAGE_ID.:&SESSION.
```

APP_SESSION

APP_SESSION is one of the most commonly used built-in substitution strings. You can use this substitution string to create hypertext links between application pages that maintain a session state by passing the session number. Table 6–8 describes the supported syntax for referencing APP_SESSION.

Table 6–8 APP_SESSION Syntax

Reference Type	Syntax
Bind variable	`:APP_SESSION`
PL/SQL	`V('APP_SESSION')`
Short PL/SQL	`V('SESSION')`
Short substitution string	`&SESSION.`
Substitution string	`&APP_SESSION.`

Consider the following examples:

- From within an HTML Region:

  ```
  <a href="f?p=100:5:&SESSION.">click me</a>
  ```

- Using PL/SQL:

  ```
  htf.anchor('f?p=100:5:'||V('SESSION'),'click me');
  ```

- Using a SQL query:

  ```
  SELECT htf.anchor('f?p=100:5:'||:app_session,'clickme') FROM DUAL;
  ```

APP_UNIQUE_PAGE_ID

APP_UNIQUE_PAGE_ID is an integer generated from an Oracle sequence which is unique for each page view. This number is used by applications to prevent duplicate page submissions and can be used for other purposes. For example, if you wish to make a unique URL to avoid browser caching issues, you can embed this number in the request or debug column in calls to the f procedure. Table 6–9 describes the supported syntax for referencing APP_UNIQUE_PAGE_ID.

Table 6–9 APP_UNIQUE_PAGE_ID Syntax

Reference Type	Syntax
Bind variable	:APP_UNIQUE_PAGE_ID
PL/SQL	V('APP_UNIQUE_PAGE_ID')
Substitution string	&APP_UNIQUE_PAGE_ID.

The following is an HTML example:

```
SELECT 'f?p=100:1:'||:APP_SESSION||':'||:APP_UNIQUE_PAGE_ID||
    ':::P1_EMPNO:'||empno,
   ename,
    job
FROM emp
```

Note the use of the APP_UNIQUE_PAGE_ID in the request column. This makes this URL unique and may avoid excessive browser caching problems.

APP_USER

APP_USER is the current user running the application. Depending upon your authentication model, the value of the user is set differently. If the application is running using database authentication, then the value of the user is the same as the database pseudo column USER. If the application uses an authentication scheme that requires the user to authenticate, the value of APP_USER is set by the authentication scheme, usually to the user name used during authentication. Table 6–10 describes the supported syntax for referencing APP_USER.

Table 6–10 APP_USER Syntax

Reference Type	Syntax
Bind variable	:APP_USER
PL/SQL	V('APP_USER')
Short PL/SQL	V('USER')
Substitution string	&USER.

Consider the following examples:

- From within an HTML Region:

  ```
  Hello you are logged in as &USER.
  ```

- Using PL/SQL:

  ```
  htp.p('Hello you are logged in as'||V('USER'));
  ```

- As a bind variable:

```
SELECT * FROM some_table WHERE user_id = :app_user
```

AUTHENTICATED_URL_PREFIX

This application level attribute identifies a valid authenticated prefix (that is, a logged in URL prefix). You can use a relative path or a full path beginning with `http`. This item is useful if your application can be run in both authenticated (logged in) and public (not logged in) modes. You can use `AUTHENTICATED_URL_PREFIX` to construct a link to an authenticated page. This item is most useful when using basic database authentication since changes to the URL can require authentication. Table 6–11 describes the supported syntax for referencing `AUTHENTICATED_URL_PREFIX`.

Table 6–11 AUTHENTICATED_URL_PREFIX Syntax

Reference Type	Syntax
Bind variable	`:AUTHENTICATED_URL_PREFIX`
PL/SQL	`V('AUTHENTICATED_URL_PREFIX')`
Substitution string	`&AUTHENTICATED_URL_PREFIX.`

BROWSER_LANGUAGE

`BROWSER_LANGUAGE` refers to the Web browser's current language preference. Table 6–12 describes the supported syntax for referencing `BROWSER_LANGUAGE`.

Table 6–12 BROWSER_LANGUAGE Syntax

Reference Type	Syntax
Bind variable	`:BROWSER_LANGUAGE`
Direct PL/SQL	`HTMLDB_APPLICATION.G_BROWSER_LANGUAGE`
PL/SQL	`V('BROWSER_LANGUAGE')`
Substitution string	`:BROWSER_LANGUAGE.`
Substitution string	`&BROWSER_LANGUAGE.`

CURRENT_PARENT_TAB_TEXT

`CURRENT_PARENT_TAB_TEXT` is most useful in page templates, but is only relevant for applications that use two level tabs (that is, parent and standard tabs). Use this string to reference the parent tab label. This substitution string enables you to repeat the currently selected parent tab within the page template. Table 6–13 describes the supported syntax for referencing `CURRENT_PARENT_TAB_TEXT`.

Table 6–13 CURRENT_PARENT_TAB_TEXT Syntax

Reference Type	Syntax
Bind variable	Not Available.
Substitution string	`&CURRENT_PARENT_TAB_TEXT.`

DEBUG

Valid values for the `DEBUG` flag are YES or NO. Turning debug on shows details about application processing. If you write your own custom code, you may wish to generate

debug information only if the debug mode is set to YES. Table 6–14 describes the supported syntax for referencing DEBUG.

Table 6–14 DEBUG Syntax

Reference Type	Syntax
Bind variable	:DEBUG
Direct PL/SQL	HTMLDB_APPLICATION.G_DEBUG
PL/SQL	V('DEBUG')
Substitution string	&DEBUG.

The following is an example of a substitution string reference that preserves the current value of DEBUG:

```
f?p=100:1:&SESSION.::&DEBUG
```

HOME_LINK

HOME_LINK is the home page of an application. The HTML DB engine will redirect to this location if no page is given and if no alternative page is dictated by the authentication scheme's logic. You define the Home Link on Application Attributes page.

Table 6–11 describes the supported syntax for referencing HOME_LINK.

Table 6–15 HOME_LINK Syntax

Reference Type	Syntax
Direct PL/SQL	HTMLDB_APPLICATION.G_HOME_LINK
PL/SQL	V('HOME_LINK')
Template Reference	#HOME_LINK#
Substitution String	&HOME_LINK.

> **See Also:** "Session Management" on page 7-7 for more information on the Home Link attribute

IMAGE_PREFIX

The value of IMAGE_PREFIX determines the virtual path the Web server uses to point to the images directory distributed with Oracle HTML DB. If you wish to reference uploaded images, use WORKSPACE_IMAGES and APP_IMAGES. Table 6–16 describes the supported syntax for referencing IMAGE_PREFIX.

> **See Also:** "APP_IMAGES" on page 6-18,"WORKSPACE_IMAGES" on page 6-26, and "Editing Application Attributes"

Table 6–16 IMAGE_PREFIX Syntax

Reference Type	Syntax
Bind variable	:IMAGE_PREFIX
Direct PL/SQL	HTMLDB_APPLICATION.G_IMAGE_PREFIX
PL/SQL	V('IMAGE_PREFIX')

Table 6–16 (Cont.) IMAGE_PREFIX Syntax

Reference Type	Syntax
Substitution string	`&IMAGE_PREFIX.`
Template Substitution	`#IMAGE_PREFIX#`

HTML DB SCHEMA OWNER

If you are generating calls to applications from within your PL/SQL code, you may need to reference the owner of the Oracle HTML DB schema. The following describes the correct syntax for a direct PL/SQL reference:

```
HTMLDB_APPLICATION.G_FLOW_SCHEMA_OWNER
```

You may also use `#FLOW_OWNER#` to reference this value in SQL queries and PL/SQL (for example, in a region or a process).

PRINTER_FRIENDLY

The value of `PRINTER_FRIENDLY` determines whether the HTML DB engine is running in print view mode. This setting can be referenced in conditions to eliminate elements not desired in a printed document from a page. Table 6–17 describes the supported syntax for referencing `PRINTER_FRIENDLY`.

Table 6–17 PRINTER_FRIENDLY Syntax

Reference Type	Syntax
Direct PL/SQL	`HTMLDB_APPLICATION.G_PRINTER_FRIENDLY (VARCHAR2 DATATYPE)`
PL/SQL	`V('PRINTER_FRIENDLY')`
Substitution string	`&PRINTER_FRIENDLY.`

LOGOUT_URL

`LOGOUT_URL` is application level attribute used to identify the logout URL. This is a URL that navigates the user to a logout page or optionally directly logs a user out. To create a logout navigation bar icon, use `&LOGOUT_URL` for the navigation bar link. If you are coding a page template use `#LOGOUT_URL#`. Table 6–18 describes the supported syntax for referencing `LOGOUT_URL`.

Table 6–18 LOGOUT_URL Syntax

Reference Type	Syntax
Bind variable	`:LOGOUT_URL`
PL/SQL	`V('LOGOUT_URL')`
Substitution string	`&LOGOUT_URL.`
Template Substitution	`#LOGOUT_URL#`

PROXY SERVER

`PROXY SERVER` is an application attribute. The attribute may be used by regions whose source comes from a URL. The following is the correct syntax for a direct PL/SQL reference used when you are writing PL/SQL to access remote Web servers from within the database (for example, when using the `utl_http package` shipped with the database).

```
HTMLDB_APPLICATION.G_PROXY_SERVER
```

PUBLIC_URL_PREFIX

PUBLIC_URL_PREFIX is an application level attribute that identifies a URL to toggle out of a logged in mode to a public view. Table 6–19 describes the supported syntax for referencing PUBLIC_URL_PREFIX.

Table 6–19 PUBLIC_URL_PREFIX Syntax

Reference Type	Syntax
Bind variable	:PUBLIC_URL_PREFIX
PL/SQL	V('PUBLIC_URL_PREFIX')
Substitution string	&PUBLIC_URL_PREFIX.
Template Substitution	#PUBLIC_URL_PREFIX#

REQUEST

Each application button sets the value of REQUEST to the name of the button. This enables accept processing to reference the name of the button when a user clicks it. In the f?p syntax, REQUEST may be set using the fourth argument.

Referencing the Value of REQUEST REQUEST is typically referenced during Accept processing (that is, the processing that occurs when you post a page). Table 6–20 describes the supported syntax for referencing REQUEST.

Table 6–20 REQUEST Syntax

Reference Type	Syntax
Bind variable	:REQUEST
Direct PL/SQL	HTMLDB_APPLICATION.G_REQUEST
PL/SQL	V('REQUEST')
Substitution string	&REQUEST
	&REQUEST. (exact syntax match)

Scope and Value of REQUEST for Posted Pages When you post a page, you initiate Accept processing. Accept processing consists of computations, validations, processes, and branches. The value of request is available during each phase of the Accept processing. Once Oracle HTML DB branches to a different page then REQUEST is set to NULL.

The value of REQUEST is the name of the button the user clicks, or the name of the tab the user selects. For example, suppose you have a button with a name of "CHANGE", and a label of "Apply Change." When a user clicks the button the value of REQUEST will be CHANGE.

Referencing REQUEST Using Declarative Conditions It is common to reference REQUEST using conditions. For example, you may wish to reset pagination when a user clicks **Go** on a report page. You can reset pagination by creating a on-submit page process. The page process can be made conditional using the condition Request = Expression 1.

To create an on-submit page process:

1. Under Conditional Display, select the condition type **Request = Expression 1**.

2. In Expression 1, enter **GO**.

Using REQUEST for Show Processing You can also use `REQUEST` for Show processing when navigating to a page using `f?p` syntax. For example:

```
f?p=100:1:&SESSION.:GO
```

Remember that the fourth argument is in the `f?p` syntax is `REQUEST`. This example goes to application 100, page 1 for the current session and sets the value of `REQUEST` to GO. Any process or region can reference the value of `REQUEST` using Show processing.

The following is a similar example using PL/SQL:

```
IF V ('REQUEST') = 'GO' THEN
   htp.p('hello');
END IF;
```

Note that `htp.p('hello')` is a call to a PL/SQL Web Toolkit package in order to print out the specified text string.

See Also:

- *Oracle Database Application Developer's Guide - Fundamentals* for more information on developing Web applications with PL/SQL
- *PL/SQL Packages and Types Reference* for more information on htp packages

SQLERRM

`SQLERRM` is template substitution only available in the Applications Region Error Message. The following describes the correct syntax for a region template substitution reference:

```
#SQLERRM#
```

SYSDATE_YYYYMMDD

`SYSDATE_YYYYMMDD` represents the current date on the database server, with the *`YYYYMMDD`* format mask applied. You may use this value instead of repeated calls to the SYSDATE() function. The following list describes the supported syntax for referencing `SYSDATE_YYYYMMDD`.

- Bind variable:

  ```
  :SYSDATE_YYYYMMDD
  ```

- PL/SQL:

  ```
  V('SYSDATE_YYYYMMDD')
  ```

- Direct PL/SQL:

  ```
  HTMLDB_APPLICATION.G_SYSDATE (DATE DATATYPE)
  ```

Table 6–21 SYSDATE_YYYYMMDD Syntax

Reference Type	Syntax
Bind variable	`:SYSDATE_YYYYMMDD`
Direct PL/SQL	`HTMLDB_APPLICATION.G_SYSDATE (DATE DATATYPE)`

Table 6–21 (Cont.) SYSDATE_YYYYMMDD Syntax

Reference Type	Syntax
PL/SQL	`V('SYSDATE_YYYYMMDD')`

WORKSPACE_IMAGES

Use this substitution string to reference uploaded images, JavaScript, and cascading style sheets that are shared over many applications within a workspace. Table 6–22 describes the supported syntax for referencing `WORKSPACE_IMAGES`.

Table 6–22 WORKSPACE_IMAGES Syntax

Reference Type	Syntax
Bind variable	`:WORKSPACE_IMAGES`
Direct PL/SQL	Not available.
PL/SQL	`V('WORKSPACE_IMAGES')`
Substitution string	`&WORKSPACE_IMAGES.`
Template Substitution	`#WORKSPACE_IMAGES#`

See Also: "APP_IMAGES" on page 6-18 and "IMAGE_PREFIX"

7

Using Application Builder

This section provides important background information about how to use Application Builder. You use Application Builder to build dynamically rendered applications in Oracle HTML DB. An application is a collection of database-driven Web pages linked together using tabs, buttons, or hypertext links.

This section contains the following topics:

- Accessing Application Builder
- Editing Application Attributes
- Viewing a Page
- Using the Developer Toolbar
- Editing a Page Definition
- Working with Shared Components
- About Themes and Templates
- Viewing Application Reports

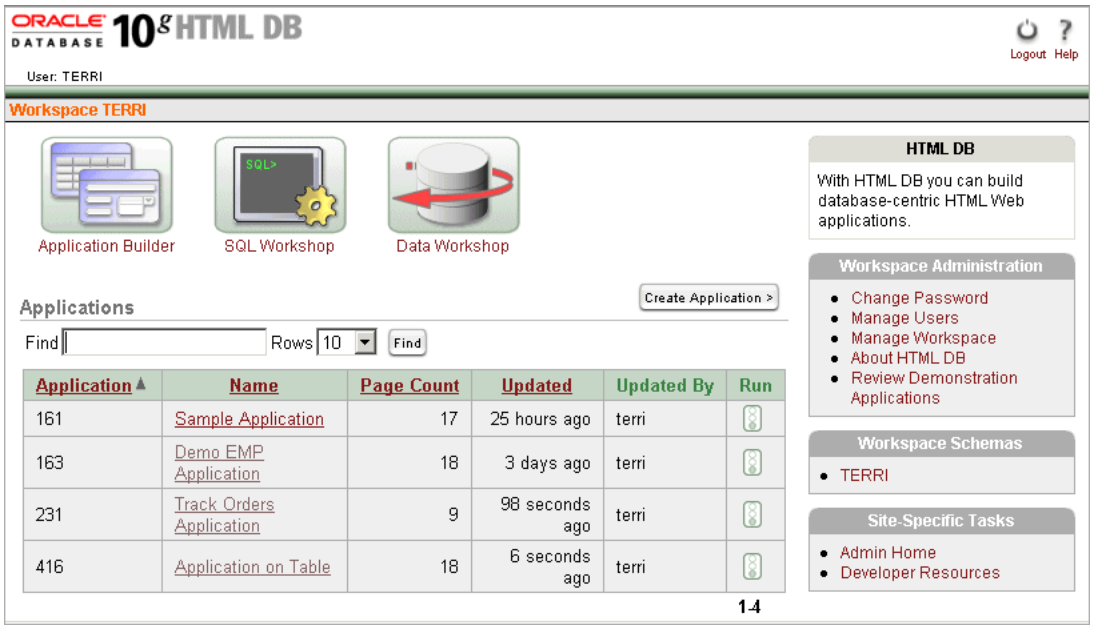
See Also:

- "Quick Start"
- "Application Builder Concepts"
- "Building an Application"
- "Controlling Page Layout and User Interface"
- "Adding Navigation"

Accessing Application Builder

An application is a collection of pages that share a common session state definition and authentication method. Application Builder is the tool you use to build the pages that comprise an application. You access Application Builder from the Workspace home page. (See Figure 7–1 on page 7-2.)

Figure 7–1 Workspace Home Page



To access Application Builder:

1. Log in to Oracle HTML DB.

   The Workspace home page appears.

2. To access Application Builder, you can either:

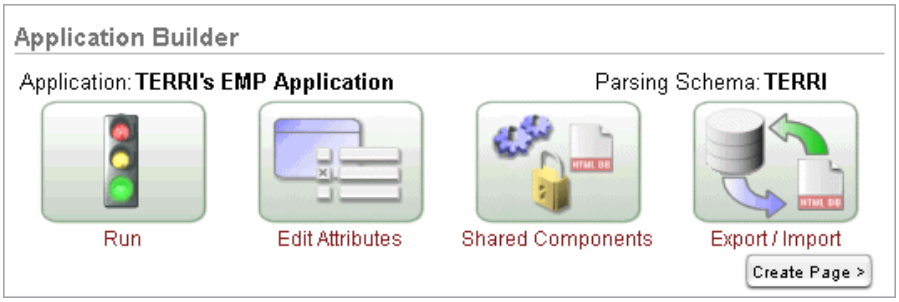
   - Select an application name from the Applications list
   - Create a new application by clicking **Create Application**

     The Application Builder home page appears.

About the Application Builder Home Page

As shown in Figure 7–2, the top of the Application Builder home page displays the current application name and application ID, last update date, and parsing schema.

Figure 7–2 Application Builder Home Page (Top)



Click the one of the following to run the current application, edit application attributes, create shared components, export and import information, or create a new page:

- **Run** submits the pages in the current application to the HTML DB engine to render viewable HTML

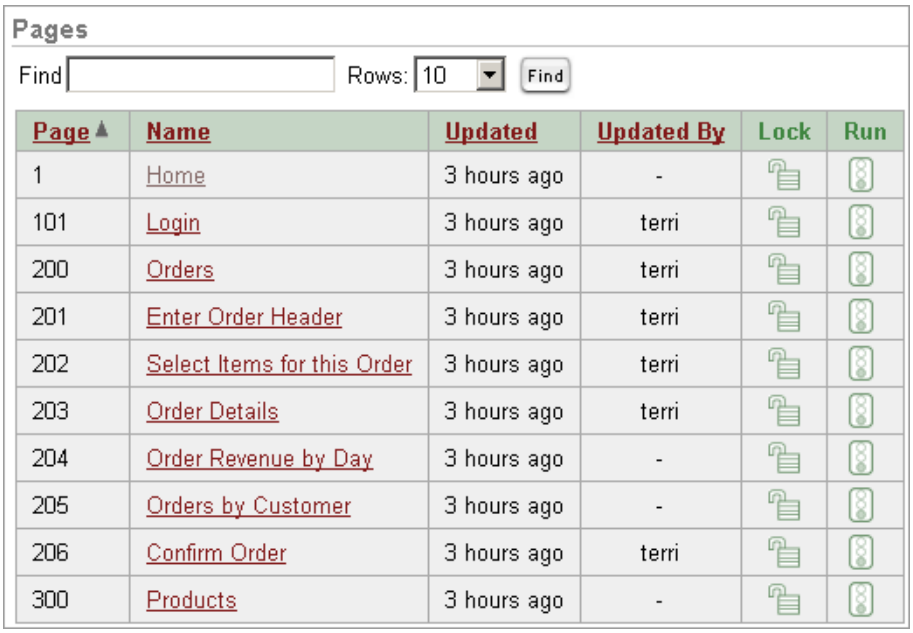
- **Edit Attributes** displays the Edit Application Attributes page
- **Shared Components** links to a new page for building shared application components and user interface controls
- **Export/Install** links you to the Export Import Wizard
- **Create Page** links to a wizard for creating a new page

See Also:

- "Running a Page" on page 8-4
- "Editing Application Attributes" on page 7-4
- "Working with Shared Components" on page 7-26
- "Deploying an Application to Another Oracle HTML DB Instance" on page 12-4
- "Adding Additional Pages" on page 8-3

As shown in Figure 7–3 on page 7-3, the Pages list displays at the bottom of the page. To access a specific page, select the page name. To search for a specific page number, enter a page number in the Find field and click **Find**. To view all pages in an application, leave the Find field blank and click **Find**.

Figure 7–3 Pages List



A Tasks list displays on the right side of the page. It contains the following links:

- **Delete this Application** enables you to delete the current application. (See "Deleting an Application" on page 8-2.)
- **Manage Page Groups** links to the Page Groups page. Make the pages within your application easier to access by organizing them into page groups. (See "Grouping Pages" on page 8-5.)

- **Manage Page Locks** links to the Locked Pages page. Locking pages in an application, prevents conflicts during application development. (See "Locking and Unlocking Page" on page 8-6.)
- **View Application Reports** displays links to summary application reports. (See "Viewing Application Reports" on page 7-29.)

Editing Application Attributes

Application attributes apply to an entire application. Once you create an application the next step is to specify application attributes.

See Also: "Creating an Application" on page 8-1

To edit application attributes:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears.
3. Click the **Edit Attributes** icon. (See Figure 7–4.)

Figure 7–4 Edit Attributes Icon

Edit Application Attributes page appears. Oracle HTML DB creates a unique application ID when you create a new application. The application ID displays at the top of the page. Beneath the application ID are links to various sections of the page. Required values are marked with a red asterisk (*).

The topics that follow describe the specific sections of the Edit Application Attributes page, including:

- Application Definition
- Authorization
- Session Management
- Theme
- Globalization
- Application Availability
- Global Notifications
- Virtual Private Database (VPD)
- Static Substitution Strings
- Logo
- Build Options
- Application Template Defaults

- Wizard Template Defaults
- Application Comments

Application Definition

Use Application Definition attributes to define basic characteristics of your application, including the application name, an optional alphanumeric alias, a version number, and the application owner. Table 7–1 on page 7-5 describes all Application Definition attributes.

Table 7–1 Application Definition Attributes

Attribute	Description
Name	Provides a short descriptive name for the application to distinguish it from other applications in the Oracle HTML DB development environment.
Application Alias	Assigns an alternate alphanumeric application identifier. You can use this identifier in place of the application ID. For example, suppose you create an alias of `myapp` for application 105. Using `f?p` syntax, you could call application 105 as either: ▪ `f?p=105:1` ▪ `f?p=myapp:1`
Version	Includes the application's version number on a page. You can also automatically tie the version to the date of last modification using the following format masks: ▪ `YYYY.MM.DD` ▪ `MM.DD.YYYY` ▪ `DD.MM.YYYY` If your application version uses YYYY.MM.DD then Oracle HTML DB replaces this format mask with the date of last modification of any application attribute.
Image Prefix	Determines the virtual path the Web server uses to point to the images directory distributed with Oracle HTML DB. During installation, the virtual path is configured as `/i/`. When embedding an image in static text (for example, in page or region headers or footers) you can reference an image using the substitution string `#IMAGE_PREFIX#`. For example, to reference the `image go.gif` you would use the following syntax: `<img src="#IMAGE_PREFIX#go.gif">` **See Also:** "IMAGE_PREFIX" on page 6-22, "Uploading Images" on page 9-41, and "Referencing Images" on page 9-41

Table 7–1 (Cont.) Application Definition Attributes

Attribute	Description
Proxy Server	Use this field to specify a proxy server. For example, Application Builder may require a proxy server when using a region source type of URL. The URL region source embeds the results of the URL (that is, the page returned by navigating to the URL) as the region source. If you use a firewall and the target of a URL is outside the firewall relative to Oracle HTML DB, you may need to specify a proxy server. You can reference values entered into this field from PL/SQL using the PL/SQL package variable `HTMLDB_APPLICATION.G_PROXY_SERVER`.
Logging	Determines whether user activity is recorded in the Oracle HTML DB activity log. When set to **Yes**, every page view will be logged, allowing a Workspace administrator to monitor user activity for each application. Disabling logging may be advisable for high volume applications.
Parsing Schema	Specifies the schema that all SQL and PL/SQL in the application will be parsed as. You may use `#OWNER#` to reference this value in SQL queries and PL/SQL (for example, in a region or a process).

Exact Substitutions	Select whether or not only exact substitutions will be supported. For optimal runtime performance, it is recommended you use exact substitutions.
	Exact substitutions use the following sytnax:
	`&ITEM.`
	Non-exact substitutions use the following sytnax:
	`&ITEM`

See Also:

- "Using Substitution Strings" on page 6-16
- "Using f?p Syntax to Link Pages" on page 6-14 for more information on linking pages

Authorization

Use Authorization to specify an authorization scheme for your application. You may assign only one authorization to an entire application. However, you may assign an authorization scheme to individual pages, page controls (such as a region, a button, or an item), or a shared component (such as a menu, a list, or tab).

An authorization scheme is a binary operation that either succeeds (equals true) or fails (equals false). If it succeeds then the component or control can be viewed, if it fails then the component or control cannot be viewed or processed. When you attach an authorization scheme to a page and it fails, an error message displays instead of the page. However, when you attach an authorization scheme to a page control (for example, a region, a button, or an item) and it fails, no error page displays. Instead, the control either does not display or is not processed or executed.

See Also: "Creating an Authentication Scheme" on page 14-4

Session Management

Use Session Management when establishing your authentication and session management infrastructure. Table 7–2 describes all session management attributes.

Table 7–2 Session Management Attributes

Attribute	Description
Home Link	This is the relative URL used to display the home page of your application. For example, `f?p=6000:600` could be for application 6000 with a home page number of 600. The value you enter here replaces `#HOME_LINK#` substitution string in application templates. You can also use this attribute to specify a custom procedure to function as a home link. For example, you could create a custom procedure named `personal_calendar` to call an HTML page to serve as the application home.
Login URL	Specifies the location of the application login page. **See Also:** "Using Substitution Strings" on page 6-16 and "Creating an Authentication Scheme" on page 14-4
Public User	Identifies the Oracle schema (that is, the user) used to connect to the database when generating unprotected pages. When a user logs in as this user, the HTML DB engine considers this to be a "public user" session. The HTML DB engine supports the following built -in display conditions: ▪ `USER_IS_PUBLIC_USER` ▪ `USER_IS_NOT_PUBLIC_USER` If the current application user (or `V('USER')`) equals the value of this attribute, then the user is considered to be a public user. Some applications have public (not logged in) and a private (logged in) modes. By determining if the user is a public user, you can conditionally display or hide information. For example you can show a login button if the user is a public user and a logout link if the user is not the public user. The public user (if null) defaults to "`PUBLIC_USER`". Reference this value using `HTMLDB_APPLICATION.G_PUBLIC_USER`. The HTML DB engine also has built-in condition types `USER_IS_PUBLIC_USER` and `USER_IS_NOT_PUBLIC`. **See Also:** "Establishing User Identity Through Authentication" on page 14-3

To view details about a selected authentication scheme, click **mange** next to **Authentication: SCHEME**.

Theme

Themes are collections of templates that can be used to define the layout and style of an entire application. Each theme provides a complete set of templates that accommodate every user interface pattern that may be needed in an application.

See Also: "Managing Themes" on page 9-8

Globalization

Use Globalization attributes to specify globalization options such as the primary application language. Table 7–3 describes theses attributes.

Table 7–3 Globalization Attributes

Attribute	Description
Application Primary Language	Identifies the language in which an application is developed. This language is the base language from which all translations are made. For example, suppose application 100 was authored in English, translated it into French, and published as application 101. The application ID would be transparent to the end user. All modifications to the application should be made to the primary language specified here.
Application Language Derived From	Specifies how Oracle HTML DB determines or derives the application language. The application primary language can be static (that is, derived from the Web browser language) or determined from a user preference or item. The database language setting determines date display and sorting characteristics. This option enables you to disable browser derived language support. You also have the option of having the application language derived from an application preference.

See Also: "About Translating an Application and Globalization Support" on page 16-1

Application Availability

Use these attributes to manage your application by defining an application status and build status. For example, if you select the status **Restricted Access**, you can specify which users who have access and can run the application. Table 7–4 describes these attributes.

Table 7–4 Application Availability Attributes

Attribute	Description
Status	Specifies if the application is available or unavailable for use.
Build Status	Identifies the build status of the current application: ▪ **Run and Build Application** - Developers can both run and develop the application. ▪ **Run Application Only** - Developers can only run the application.
Message for unavailable application	If Status is set to **Unavailable** this text displays. This text will not display if Status is set to **Available**.
Restrict to comma separated user list (Status must equal Restricted Access)	Specifies users who can run the application if the Status is set to Restricted Access. To use this attribute: 1. From the Status list, select **Restricted Access**. 2. Enter a comma delimited list of users who can run the application in the field provided.

See Also: "Managing Application Build Status" on page 18-20

Global Notifications

You can use the Global Notifications attribute to communicate system status to application users. For example, you can use this attribute to notify users of scheduled

downtime or communicate other messages regarding application availability. If your page template contains a #GLOBAL_NOTIFICATION# substitution string, then the text entered here displays on each page.

To create a global notification:

1. Include the #GLOBAL_NOTIFICATION# substitution string in your page template.
2. Navigate to the Edit Application Attributes page and enter a message in the Global Notifications attribute.
3. Click **Apply Changes**.

See Also: "Using Substitution Strings" on page 6-16

Virtual Private Database (VPD)

VPD provides an application programmatic interface (API) which enables developers to assign security policies to database tables and views. Using PL/SQL, developers can create security policies with stored procedures and bind the procedures to a table or view by means of a call to an RDBMS package. Such policies are based on the content of application data stored within the database, or are based on context variables provided by the Oracle database. In this way, VPD permits access security mechanisms to be removed from applications and centralized.

The PL/SQL you enter in this field is executed immediately after the user is authenticated. V('USER') is accessible from this function. Session state for the current call is not yet initialized when this call is made. If your application does not need to employ VPD to support multiple customers in the same database, leave this attribute null.

See Also:

- "Providing Security Through Authorization" on page 14-9
- *Oracle Label Security Administrator's Guide*

Static Substitution Strings

Use these fields to define static substitution strings for your application. You can use static substitution string for phrases or labels that occur in many places within an application. Defining static substitution strings centrally enables you to change text strings in multiple places in your application by making a single change to the Substitution Value defined on this page.

See Also: "Using Substitution Strings" on page 6-16

Logo

Use these attributes to identify an image to be used as the logo for this application. In Image identify the image name. If you identify an image in the Image attribute and include the #LOGO# substitution string in your page template, the HTML DB engine generates an image tag. Use Logo Image Attributes to identify specific image attributes for the logo image. For example:

```
width="100" height="20" alt="Company Logo"
```

See Also: "Uploading Images" on page 9-41, "Customizing Templates" on page 9-16, and "Page Templates" on page 9-24

Build Options

Use Build Options to enable or disable functionality. Most application attributes have a build option attribute. Do not specify a build option unless you plan on excluding that object from specific installations. Build Options have two possible values: `INCLUDE` and `EXCLUDE`. If you specify an attribute as being included then the HTML DB engine considers it at runtime. However, if you specify an attribute as being excluded then the HTML DB engine treats it as if it does not exist.

See Also: "Using Build Options to Control Configuration" on page 12-13

Application Template Defaults

Application Template Defaults list the default templates for this application. To specify a new template at the application level, you can either:

- Select a new theme
- Select a new default page template on the Define Theme page

You can also override this default by making a selection from the Page Template list on the Page Attributes page.

Table 7–5 describes Application Template Defaults for the current application.

Table 7–5 Application Template Defaults Attributes

Attribute	Description
Default Page Template	Indicates the default page template for displaying pages. You can override this selection by making a selection from the Page Template list on the Page Attributes page. **See Also:** "Editing Page Attributes" on page 7-16
Print Mode Page Template	Identifies the template to be used when the HTML DB engine is in printer friendly mode. When calling the HTML DB engine to render a page, you have the option of specifying whether or not the page should be displayed in a printer friendly mode. If you specify YES, then the page displays using a printer friendly template. The HTML DB engine displays all text within HTML Form Fields as text. The printer friendly template does not need to have the `#FORM_OPEN#` or `#FORM_CLOSE#` substitution string. **See Also:** "Optimizing a Page for Printing" on page 9-39
Error Page Template	Optional. Specifies a page template to use for errors that display on a separate page as opposed to those that display inline.

See Also: "Changing Default Templates in a Theme" on page 9-9 and "Customizing Templates" on page 9-16

Wizard Template Defaults

Wizard Template Defaults identify default templates Application Builder uses when running wizards. You can override these settings on the attributes page for each control or component. Table 7–6 on page 7-11 describes Wizard Template Defaults for the current application.

Table 7–6 Wizard Template Defaults Attributes

Attribute	Description
Calendar	Default calendar template used when creating a new calendar.
Label	Default label template used when you create new page items.
Report	Default report template used when you create new report.
List	Default template used when you create a list.
Menu	Default template used when you create a menu.
Button	Default template to be used when you create new buttons that are template controlled.
Region	Default region template used when you create a new region.
Chart Region	Default region template used when you create a chart.
Form Region	Default region template used when you create a form.
Report Region	Default region template used when you create a report.
Tabular Form Region	Default region template used when you create a tabular form.
Wizard Region	Default region template used when you create a new wizard component
Menu Region	Default region template used when you create a new menu.
List Region	Default region template used when you create a new list.

See Also: "Changing Default Templates in a Theme" on page 9-9 and "Customizing Templates" on page 9-16

Application Comments

Use this attribute to record developer comments about the current application.

Viewing a Page

A page is the basic building block of an Oracle HTML DB application. Each page can have buttons and fields (called items) and application logic (or processes). You can branch from one page to the next using conditional navigation, perform calculations (called computations), perform validations (such as edit checks), and display reports, calendars, and charts.

Topics in this section include:

- Viewing a Page Definition
- Understanding the Page Definition
- Additional Page Definition Features

Viewing a Page Definition

You can view, create, and edit the controls and components that define a page by accessing the Page Definition.

To view the Page Definition for an existing page:

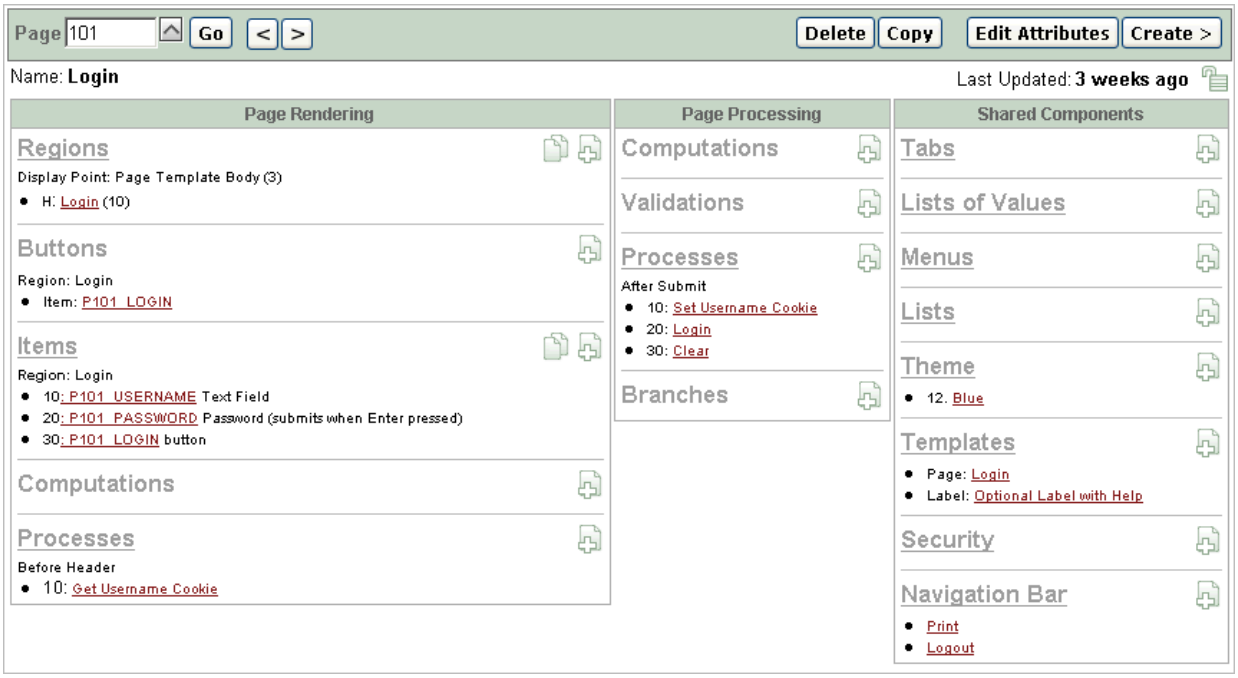
1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

Application Builder appears. The list of pages in the current application appears at the bottom of the page.

3. From the Pages list, select a page.

   As shown in Figure 7–5, the Page Definition appears.

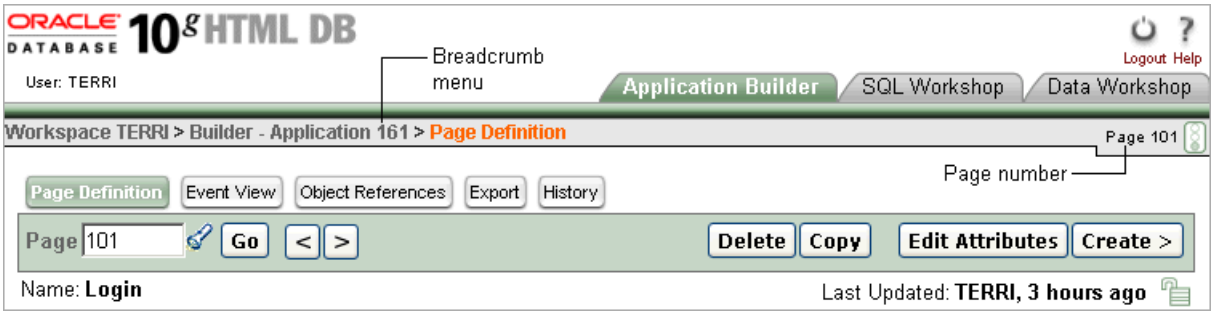
Figure 7–5 Page Definition



Understanding the Page Definition

As shown in Figure 7–6 on page 7-12, a breadcrumb menu displays at the top of each Page Definition. Breadcrumb menus appear on every page in Oracle HTML DB. Each menu entry indicates your location relative to other pages in the current application and functions as a navigation path. You can instantly link to another page by clicking a breadcrumb menu entry.

Figure 7–6 Page Definition (Top)



The current page number displays on the far right side of the page, next to a small button that resembles a traffic light. Clicking this button to runs the current page and renders it into viewable HTML.

A page navigation bar appears next. (See Figure 7–6.) Options available on the page navigation bar include:

- **Page.** This field displays the current page number. To access another page directly, enter a page number and click **Go**. To access the previous or next page number, click the arrow buttons.
- **Delete**. Deletes the currently selected page.
- **Copy**. Creates a copy of the selected page within the current application.
- **Edit Attributes**. Links to Page Attributes. Use this page to edit high-level page attributes such as the page name, an optional name alias, and view information about defined tab sets, specified templates, and authorization schemes.
- **Create**. Links to a wizard for creating a new page.

The page name and last update date appear next.

See Also:

- "Editing Page Attributes" on page 7-16
- "Adding Additional Pages" on page 8-3
- "Running a Page" on page 8-4

The center of every Page Definition is divided into three sections as described in Table 7–7.

Table 7–7 Divisions of the Page Definition

Section	Description
Page Rendering	Defines all attributes for the page, regions, buttons, items, page rendering computations and page level processes. **See Also:** "Understanding Page Rendering Controls" on page 7-20
Page Processing	Specifies page level application logic such as computations, validations, processes, and branching. **See Also:** "Understanding Page Processing Controls" on page 7-21
Shared Components	Displays application components that can be shared among multiple pages. **See Also:** "Working with Shared Components" on page 7-26, "Creating Tabs" on page 10-4, "Creating Lists of Values" on page 8-41, "Creating Menus" on page 10-7, "Creating Lists" on page 10-11, "Customizing Templates" on page 9-16, "Understanding Security" on page 14-2, and "Creating a Navigation Bar" on page 10-1

Additional Page Definition Features

Each Page Definition contains the following row of buttons at the top of the page:

- Event View links to a report that details currently defined page controls and processes
- Object References displays a list of database objects referenced by the current page
- Export enables you to export the current page
- History displays a history of recently changed pages

See Also: "Viewing a Page" on page 7-11

Event View

Clicking **Event View** displays the Page Application View report. This report details all currently defined page controls and processes. It provides a chronological view of how and in what order the HTML DB engine renders the page, invokes logic, and runs processes. You can control the amount of information that displays by selecting one of the following view options:

- **Show All** displays all possible page controls and processes, including those not currently defined
- **Show Used** displays currently used page controls and processes (Default)

To view details about a specific page control or process, click the appropriate hypertext link. Alternately, you can create new page controls and processes by clicking the small icons to the left of each entry.

To run the current page, click **Run**. To create a new page, click **Create**.

Object References

Clicking **Object References** displays a list of database objects referenced by the current page. Click the Referenced Name to view object details. Click the application number to view all database objects referenced by the current application.

Export

Click **Export** to export the current page. Remember that some pages may reference shared components. To export all pages within an application, you need to complete an application export.

See Also: "Deploying an Application to Another Oracle HTML DB Instance" on page 12-4 and "Exporting a Page in an Application" on page 12-5

History

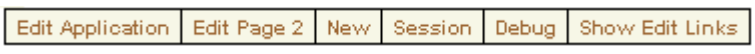
Clicking **History** displays the Recent Changes report. This report displays a history of recent changes to the currently selected page by developer, application, page number, modification date, attribute, and action.

Using the Developer Toolbar

Users who log in to Oracle HTML DB having developer privileges have access to the Developer toolbar. The Developer toolbar offers a quick way to edit the currently selected page, create a new page, control, or component, view session state, or turn edit links on an off.

The Developer toolbar (see Figure 7–7) displays at the bottom of every page in a running application.

Figure 7–7 Developer Toolbar



The Developer toolbar consists of the following links:

- **Edit Application** links you to the Application Builder home page. (See "About the Application Builder Home Page" on page 7-2.)
- **Edit Page** accesses the Page Definition for the currently running page. (See "Editing a Page Definition" on page 7-15.)
- **New** links to a wizard for creating a new blank page, component (report, chart, or form), page control (region, button, or item), or a shared component (menu, list, or tab). (See "Creating a Page from the Developer Toolbar" on page 8-4.
- **Session** displays a new window containing session state information for the current page. (See "Viewing Session State" on page 6-9.)
- **Debug** runs the current page in debug mode. (See "Accessing Debug Mode" on page 11-2.)
- **Show Edit Links** toggles between **Show Edit Links** and **Hide Edit Links**. Clicking **Show Edit Links** displays edit links next to each object on the page that can be edited. Each edit link resembles two colons (::) and appears to the right of navigation bar items, tabs, region titles, buttons, and items. Clicking on the link displays another window in which to edit the object.

Editing a Page Definition

A page is the basic building block of an application. Each page has page number, a name, and typically some text attributes such as a header, title and footer. You add content to your page by creating page controls (regions, items, and buttons). Page templates and page region templates control the exact look and feel of each page.

Topics in this section include:

- Accessing a Page Definition
- Accessing a Summary View of Controls, Components, and Application Logic
- Copying or Creating a New Control or Component
- Editing Page Attributes
- Understanding Page Rendering Controls
- Understanding Page Processing Controls
- Creating a Page Computation
- Creating a Page Process

See Also: "Understanding the Page Definition" on page 7-12

Accessing a Page Definition

You can view, create, and edit the controls and components that define a page by accessing the Page Definition.

To view the Page Definition for an existing page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears. The list of pages for the selected application appears at the bottom of the page.
3. From the Pages list, select a page name.

The Page Definition appears.

Accessing a Summary View of Controls, Components, and Application Logic

Each Page Definition serves a a central navigation point for all the controls, components, and application logic that defines a page.

You can access a summary view of all defined controls or components by selecting the title (for example, Regions, Button, Items, Computations, Processes, and so on). For example, selecting **Regions** displays a summary report of all currently defined regions on the current page. Use this summary view to:

- Edit the multiple attributes at once by making new selections from the available fields and select lists
- Link to a definition page by clicking the **Edit** icon

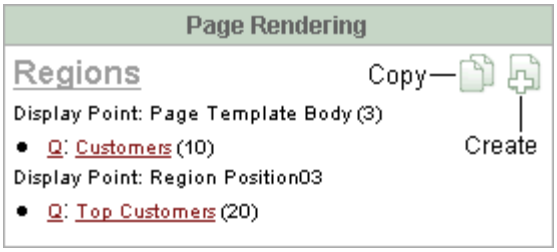
You can access additional summary views by clicking the buttons at the top of each page. To save your edits to any summary view, click **Apply Changes**.

You can also view the attributes of a specific control or component by selecting its name on the Page Definition. For example, if your Page Definition contained a region named *Customers*, clicking the region name displays an attribute page for that region.

Copying or Creating a New Control or Component

You can copy or create new controls or components by clicking the Copy and Create icons. (See Figure 7–8.) The Copy icon resembles two small overlapping pages. The Create icon consists of a plus (+) sign overlapping a small page. Click the Copy icon to make a copy of the current control or component. Click the Create icon to create a new control or component.

Figure 7–8 Copy and Create Icons



Editing Page Attributes

Page attributes only apply to a specific page. You access page attributes from the Page Definition.

To edit page attributes:

1. Navigate to the Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. From the navigation bar at the top of the page, click **Edit Attributes**.

   The Page Attributes page appears. Required values are marked with a red asterisk (*).

The topics that follow describe the specific sections of the Page Attributes page, including:

- Page Identification

- Primary Display Attributes
- HTML Header
- Page Header, Footer and Text Attributes
- On Load JavaScript
- Security
- Duplicate Page Submission Checks
- On Error Text
- Page Help Text
- Comments

> **See Also:** "Adding Additional Pages" on page 8-3 for more information on creating a new page

Page Identification

Use these attributes to define general attributes for the current page such as a page name, an optional alphanumeric alias, and associated page groups. Table 7–9 describes these attributes.

Table 7–8 Page Identification Attributes

Attributes	Descriptions
Name	Identifies the name of the current page for application developers. This name is used in numerous Oracle HTML DB pages and reports, along with the page number and page title.
Page Alias	Enter an alphanumeric alias for this page. This alias must be unique within the current application. For example, if you were working on page 1 of application 100, you could create an alias called `home`. You could then access this page from other pages using the following `f?p` syntax: `f?p=100:home`
Group	Identify the page group you would like to associate with this page. **See Also:** "Grouping Pages" on page 8-5

Primary Display Attributes

Use these attributes to define general display attributes for the current page such as the selected page template, standard tab set, title, and cursor focus. Table 7–9 on page 7-17 describes these attributes.

Table 7–9 Primary Display Attributes

Attributes	Descriptions
Page Template	Select a page template to control the appearance of this page. Making a selection here overrides the use of the default page template selected for the application when rendering this page.
Standard Tab Set	Select a standard tab set to be used for this page. A standard tab set is associated with a specific page and page number. You can use standard tabs to link users to a specific page. **See Also:** "Creating Tabs" on page 10-4

Table 7–9 (Cont.) Primary Display Attributes

Attributes	Descriptions
Title	Enter a title to display in the title bar of the browser window. The HTML DB engine uses the title you specify here in place of the #TITLE# substitution string used in the page template. This title is inserted between the HTML tag <TITLE></TITLE>.
Cursor Focus	Select whether you want the cursor focus to be placed in the first field on the page. Select **Do not focus cursor** if you do not want to include JavaScript.

HTML Header

Use this attributes to replace the #HEAD# substitution string in the page template header. The values entered here are inserted after the HTML <HEAD> tag. Common uses of these attributes:

- Code page specific inline cascading style classes
- Add additional style sheets for a specific page
- Code page specific JavaScript
- Code page specific meta tag page refresh

Page Header, Footer and Text Attributes

Use these attributes to define page header, body header, body footer, and page footer text. Table 7–10 describes these attributes.

Table 7–10 Page Header, Footer and Text Attributes

Attribute	Description
Header Text	Enter text of HTML to display after the page template header and before page template body.
Body Header	Enter text of HTML to display before showing regions. Displays before the page template #BOX_BODY# substitution string.
Footer	Enter text of HTML to display after page template body and before page template footer.

On Load JavaScript

Use this attribute to add onload events such as calls to JavaScript. In the Page HTML Body Attribute, enter JavaScript or text to be substituted for your page template's #ONLOAD# substitution string. To use this feature, your page template must include the #ONLOAD# substitution string.

You can use the Page HTML Body Attribute to write into the contents of the opening BODY tag. A typical page template might use #ONLOAD# within the opening <body> tag as shown in the following example:

```
<html>
<head>
...
</head>
<body #ONLOAD# >
```

See Also: "Incorporating JavaScript into an Application" on page 8-50

Security

Use these attributes to specify an authorization scheme for the current page and indicate whether the page requires an authentication method.

From the Authorization Scheme list, select an authorization scheme to be applied to the page. Authorization schemes are defined at the application level and can be applied to many elements within the application. A given authorization scheme is evaluated either once for each application session (at session creation), or once for each page view. If the selected authorization scheme evaluates to true, then the page displays and is subject to other defined conditions. If it evaluates to false, then the page will not display and an error message displays.

From the Authentication list, specify whether this page has been defined as public or requires authentication. If a page is identified as public, the page can be viewed before authentication. This attribute only applies if the application uses SCHEME authentication. The application's page sentry function may access this page attribute to identify pages that do not require prior authentication to view. The implementation of the authentication scheme's page sentry function determines whether this attribute has any effect.

See Also: "Understanding Security" on page 14-2

Duplicate Page Submission Checks

Use the Allow duplicate page submissions list to specify whether Oracle HTML DB enables users to process a page multiple times in a row. Set this attribute to No to prevent duplicate page submissions from being processed multiple times.

Examples of duplicate page submissions include:

- A user clicks the Submit button multiple times
- You create a branch of type Branch to Page and the user clicks the browser reload button

Configuration Management

Build options allow you to enable or disable functionality. Most application attributes have a build option attribute. Do not specify a build option unless you plan on excluding that object from specific installations.

Build options have two possible values: `INCLUDE` and `EXCLUDE`. If you specify an attribute as being included, then the HTML DB engine considers it part of the application definition at runtime. Conversely, if you specify an attribute as being excluded, then the HTML DB engine treats it as if it does not exist

See Also: "Using Build Options to Control Configuration" on page 12-13

On Error Text

Use this attribute to specify the error text that displays in the `#NOTIFICATION_MESSAGE#` substitution string in the event of an error occurring on the page.

See Also: "Page Templates" on page 9-24

Page Help Text

Use this attribute to enter help text for the current page. Page level help supports shortcuts using the following syntax:

```
"SHORTCUT_NAME"
```

Help text is displayed using a help system that you must develop. To show the help for a specific page, call the `HTMLDB_APPLICATION.HELP` procedure from a page that you create for displaying help text. For example, you could use a navigation bar icon similar to:

```
f?p=4000:4600:&SESSION::&DEBUG::LAST_STEP:&APP_PAGE_ID
```

See Also: "Creating a Help Page" on page 8-53

Comments

Use this attribute to record developer comments about the current page. These comments never display when the application is running.

Understanding Page Rendering Controls

Use the Page Rendering section of the Page Definition to specify attributes for defined regions, buttons, items, page rendering computations, and page processes.

Topics in this section include:

- Regions
- Buttons
- Items
- Page Computations
- Page Processes

Regions

A region is a section of a page that serves as a container for content within a page. The content of a region is determined by the region source. For example, a region may contain a report based on a SQL query you define, or it may contain static HTML.

See Also:

- "Customizing Regions" on page 9-3 for more information on creating specific types of regions
- *Oracle Database Application Developer's Guide - Fundamentals* for more information on developing Web applications with PL/SQL
- *PL/SQL Packages and Types Reference* for more information on htp packages

Buttons

As you design your application you can use buttons to direct users to a specific page or URL, or to post or process information. Buttons can be placed in predefined region template positions or among items in a form.

See Also: "Creating Buttons" on page 8-32

Items

An item can be a text field, text area, password, combobox, check box, and so on. Item attributes affect the display and behavior of items on a page. For example, these attributes can impact where a label displays, how large an item will be, and whether the item will display next to, or below, the previous item.

There are two types of items, page items and application items. Page items are placed on a page and have associated user interface properties, such as Display As, Label, and Label Template. Application items are not associated with a page and therefore have no user interface properties. An application item can be used as a global variable.

See Also: "Creating Items" on page 8-35

Page Computations

You can use computations to assign a value to an identified item when a page is submitted or displayed. You can also use application level computations to assign values to items. Most application level computations are performed for every page in an application. In contrast, computations created at the page level only execute when that page is rendered or processed.

See Also: "Creating a Page Computation" on page 7-23

Page Processes

You create a page process to execute some logic (for example, using PL/SQL), or to make a call to the rendering engine. Typically a process performs an action. For example, a process may be hand coded PL/SQL, or the invocation of a predefined process available within Oracle HTML DB.

A page process is a unit of logic that runs when a specific event occurs, such as loading or submitting page. From a functional perspective, there is no difference between page level and application level processes. The difference lies in the point at which the process occurs.

See Also: "Creating a Page Process" on page 7-24

Understanding Page Processing Controls

Use the Page Processing section of the Page Definition to specify application logic such as computations, validations, processes, and branching. In general, the HTML DB engine runs this logic in the order it appears on the Page Definition.

Understanding Validations

You can define a validation declaratively by selecting a built-in validation type or by entering custom SQL or PL/SQL. You enter the actual validation edit check in the Validation Messages field. Be aware that if a validation fails, subsequent page processing does not occur. Also remember that the validation you enter must be consistent with validation type you selected. For more information on validation types, see online help.

See Also: "Validating User Input in Forms" on page 8-23

Creating a Validation To create a new validation:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)

2. Under Validations, click the **Create** icon.
3. When the Create Validations Wizard appears, follow the on-screen instructions.

   Validations Types are divided into two categories:

   - **Item.** These validations start with the phrase "Item" and provide common checks you may want to perform on the item that the validation is associated with.
   - **Code.** These validations require you provide either PL/SQL code or a SQL query that defines the validation logic. Use this type of validation to perform custom validations that require verifying values of more than one item or accessing additional database tables.
4. Follow the on-screen instructions.

Note: Validations may not contain more than 3,950 characters.

Defining How Validation Error Messages Display You can choose to have validation error messages display inline (that is, on the page where the validation is performed) or on a separate error page.

To define how a validation error message displays:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page" on page 7-11.)
2. Under Validations, select the appropriate validation.

   The attributes page for the validation appears.
3. Scroll down to Error Messaging.
4. In Error Message, enter your error message text.
5. From Error message display location, select a display location.

   This attribute identifies where a validation error message displays. Validation error messages can display on an error page or inline within the existing page. Inline error messages can display in a notification area (defined as part of the page template) or within the field label.

   To create a hard error that stops processing of any remaining validations, you must display the error on an error page.
6. If you select Inline with Field or Inline with Field and in Notification, you need to associate an item with the error message. To associate an item with the error message, select the item from the Associated Item list.
7. Click **Apply Changes**.

Processing Validations Conditionally You can control when and if a validation is performed under Conditional Validation Processing. To have a validation performed when a user clicks a button, make a selection from the When Button Pressed list.

You can add other conditions by making a selection from the Condition Type list and entering text in the expression fields.

Understanding Branching

A branch is an instruction to go to a specific page, procedure, or URL. For example you can branch from page 1 to page 2 after page 1 is submitted.

You create a new branch by running the Create Page Branch Wizard and specifying Branch Point and Branch Type. The Branch Type defines the type of branch you are creating. For more information on Branch Types, see online help.

Defining a Branch Point and Branch Action You specify when a branch executes by making a selection from the Branch Point list. Valid options include:

- **On Submit: Before Computation** - Occurs before computations, validations, or processing. Use this option for buttons that do not need to invoke any processing, for example, a Cancel button.
- **On Submit: Before Validation** - Occurs after computations, but before validations or processing. If a validation fails, page processing stops, a rollback is issued, and the page displays the error. Because of this default behavior, you do not need to create branches to accommodate validations. However, you may want to branch based on the result of a computation (for example, to a previous branch point).
- **On Submit: Before Processing** - Occurs after computations and validations, but before processing. Use this option to branch based on a validated session state, but before performing any page processing.
- **On Submit: After Processing** - Occurs after computations, validations, and processing. This option branches to a URL or page after performing computations, validations, and processing. When using this option, remember to sequence your branches if you have multiple branches for a given branch point.
- **On Load: Before Header** - Occurs before a page is rendered. This option displays another page instead of the current page or redirects the user to another URL or procedure.

Depending upon the Branch Type you select, you can specify the following additional information in Branch Action attributes:

- The page number of the page you wish to branch to
- PL/SQL procedure which ultimately renders a branch target page
- A URL address

Branching Conditionally Like other controls, branches can be made conditional. To create a conditional branch, make a selection from the Condition Type list and enter text in the expression fields to implement the condition type you choose.

See Also: "Controlling Flow Using Branches" on page 10-6

Creating a Page Computation

You create a page computation by running the Create Page Computation Wizard. For each computation, specify the item for which you are creating the computation as well as a computation type.

See Also: "Page Computations" on page 7-21

To create a page computation:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Computations, click the **Create** icon.
3. Select a category.

4. Follow the on-screen instructions.

Editing Computation Attributes

You can edit how a computation works on the Edit Page Computation page. To access the Edit Page Computation page, navigate to the Page Definition and select the computation name.

Defining a Computation Point and Computation Source

You control when a computation executes under Firing Points attributes by specifying a sequence number and a computation point. The computation point `On New Instance` executes the computation when a new session (or instance) is generated.

Under Computation Source, enter an expression or query to compute an item's value. In the event a computation fails, you can optionally define an error message in Computation Error Message field.

Creating Conditional Computations

You can make a computation conditional by making a selection from the Condition Type list and entering text in the expression fields.

Creating a Page Process

You create a process by running the Create Process Wizard. During the wizard, you define a process name, specify a sequence, the point at which process will execute, and select a process category. You can change nearly all of these attributes on the Edit Page Process page.

> **See Also:** "Page Processes" on page 7-21

To create a new process:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Processes, click the **Create** icon.
3. Select a category. Table 7–11 describes available page process categories.

Table 7–11 Process Categories

Process Category	Description
Data Manipulation	Oracle HTML DB supports the following declarative data manipulation processes: ▪ Select **Automatic Row Fetch** and **Automatic Row Processing (DML)** to create an automatic DML (Data Manipulation Language) process ▪ Use **Multi Row Update** and **Multi Row Delete** in conjunction with tabular forms. ▪ Use **Add Rows to Tabular Form** in conjunction with a tabular form
Form Pagination	Implements pagination through the detail records associated with a master detail form. This process check the master table to determine which set of detail records you are in and determines what the next detail record should be. **See Also:** "Building a Master Detail Form" on page 8-20

Table 7–11 (Cont.) Process Categories

Process Category	Description
On Demand	Creates an application level process that can only be executed when called from a specific page. When you create this process type at the page level, it creates a procedure that calls a previously defined On Demand procedure from the application level.
PL/SQL	Runs the PL/SQL you provide. Use this process type to execute a block of PL/SQL entered directly into the process or to simply call an existing API.
Report Pagination	Resets pagination back to the first result set. The HTML DB engine keeps track of where the user is within a given result set. This process category returns the user to the beginning result set. In other words, this category resets the counters associated with the report region to return the first part of the result set the next time the result set displays.
Session State	Sets the values of existing session state items to null. Select this process type to clear the cache for applications, sessions, or items as well as to clear existing user preferences. **See Also:** "Managing Session State Values" on page 6-10 and "Managing User Preferences" on page 15-22
Web Services	Implements a Web Service as a process on a page. Running the process submits a request to the service provider. **See Also:** "Invoking a Web Service as a Process" on page 15-20

4. Follow the on-screen instructions.

Editing Process Attributes

Once you create a process you can control when the process executes and what the process does by editing attributes on the Edit Page Process page.

To edit an existing page process:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Select the process name.

   The Edit Page Process page appears.

Changing Processing Points and Source You control when a process executes by specifying a sequence number and a process point under Process Firing Point. You can prevent a process from running during subsequent visits to a page by selecting one of the following options under Run Process:

- Once for each page visit
- Once for each session or when reset

Enter the appropriate code for PL/SQL process types. For PL/SQL anonymous block processes, enter the appropriate code under **Process**. For Clear Cache processes, enter the appropriate code under **Source**. In the event a process fails, you can optionally define an error message in the Process Error Message field.

Creating Conditional Processes You can make a process conditional by selecting a condition type and entering an expression under Conditional Processing.

Additionally, you can also make a selection from the When Button Pressed attribute. When you select a button from this list, the process only executes if a user clicks the selected button.

Working with Shared Components

You can use the tools and wizards on the Shared Components page either at the application level, or on specific pages. Examples of shared components include logic controls (build options, computations, processes), shared navigation and user interface elements (menus, lists, tabs, lists of values, templates) and authentication and authorization schemes.

Accessing Shared Components

To access the Shared Components page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears. Shared components are grouped into the following categories:

   - Logic
   - Navigation
   - Security
   - User Interface
   - Translations
   - Files
4. To create a shared component, select the appropriate link and follow the on-screen instructions.

Logic

Logic contains links to tools for creating and managing application level logic controls such as build options, computations, items, processes, and Web services. Table 7–12 describes the options available under Logic.

Table 7–12 Shared Components - Logic

Shared Component	Description
Build Options	Use build options to conditionally display or process specific functionality within an application. You can use build options to control which features of an application are turned on for each application deployment. If you specify a build option at the application level, you do not need to specify it for each component (for example, for each page, branch, button, item, or tab). **See Also:** "Using Build Options to Control Configuration" on page 12-13

Table 7–12 (Cont.) Shared Components - Logic

Shared Component	Description
Computations	Use application level computations to assign values to application and page level items for each page displayed or for a new page instance. You can also create an application level computation and execute it conditionally on multiple pages.
Items	Application level items do not display, but are used as global variables to the application. You set the value of a page level item using an application or page computations. Use the `ON_NEW_INSTANCE` computation to set a value. **See Also:** "Creating Items" on page 8-35
Processes	Use application processes to run PL/SQL logic: ■ At specific points for each page in the application ■ As defined by the conditions under which the process is set to execute ■ Upon the create of a new session Note that **On Demand** processes executes only when called from specific pages.
Web Services	Web services in Oracle HTML DB are based on SOAP (the Simple Object Access Protocol). You can create a reference to a Web service and then incorporate it into an application to process data submitted by a form, or to render output in the form. **See Also:** "Implementing Web Services" on page 15-14

Navigation

Contains links to tools for creating and managing lists, menus, navigation bars, tabs, and trees. Table 7–13 describes the available options under Navigation.

Table 7–13 Shared Components - Navigation

Shared Component	Description
Lists	A list is a shared collection of links. You control the appearance of a list through list templates. Each list element has a display condition which enables you to control when it displays. **See Also:** "Creating Lists" on page 10-11
Menus	A menu is a hierarchical list of links that displays using templates. You can display menus as a list of links, or as a breadcrumb path. See Also: "Creating Menus" on page 10-7
Navigation Bar	Navigation bars offer users a simple navigation path for moving between pages in an application. The location of a navigation bar depends upon the associated page template. A navigation bar icons can display as a link from an image or text. A navigation bar entry can be an image, an image with text beneath it, or text. **See Also:** "Creating a Navigation Bar" on page 10-1

Table 7–13 (Cont.) Shared Components - Navigation

Shared Component	Description
Tabs	Tabs are an effective way to navigate users between pages in an application. You can create two types of tabs: standard tabs or parent tabs. A standard tab set is associated with a specific page and page number. A parent tab set functions as a container to hold a group of standard tabs. **See Also:** "Creating Tabs" on page 10-4
Trees	A tree is an effective way to communicate hierarchical or multiple level data. See Also: "Creating Trees" on page 10-14

Security

Contains links to the Authentication and Authorization pages. You provide security for your application through authentication and authorization. **Authentication** is the process of establishing users' identities before they can access an application. **Authorization** controls user access to specific controls or components based on predefined user privileges.

See Also: "Understanding Security" on page 14-2

User Interface

Contains links to tools for creating and managing lists of values, shortcuts, and templates. Table 7–14 describes the available options under User Interface.

Table 7–14 Shared Components - User Interface

Shared Component	Description
Lists of Values	A list of values (LOV) is a static or dynamic definition used to display a popup list of values, select list, check box, or radio group. **See Also:** "Creating Lists of Values" on page 8-41
Shortcuts	When you create a shortcut, you define frequently used HTML in a central repository so you can later references it in various locations within your application. **See Also:** "Utilizing Shortcuts" on page 8-48
Themes and Templates	Templates control the look and feel of the page in your application. **See Also:** "About Themes and Templates" on page 7-29 and "Customizing Templates" on page 9-16

Translations

Contains links to the tools for translating applications developed in Oracle HTML DB. You can develop applications in Oracle HTML DB that can run concurrently in different languages. A single Oracle database and Oracle HTML DB instance can support multiple database sessions customized to support different languages. Table 7–15 describes the available options under Translations.

Table 7–15 Shared Components - Translations

Shared Component	Description
Translation Services	Translate an application. To translate an application, you must map the primary and target application IDs, seed and export text to a translation file, translate the text, and then apply and publish the translation file.
Manage Messages	Translate messages. Messages are named text strings that can be called from PL/SQL code you write. This PL/SQL can be anonymous blocks within page processes and page regions, or in packages and procedures.

See Also: "About Translating an Application and Globalization Support" on page 16-1

Files

Contains links to repositories for uploading and managing style sheets, images, and static files.

See Also: "Using Custom Cascading Style Sheets" on page 9-39. "Uploading Images" on page 9-41, and "Uploading Static Files" on page 9-42

About Themes and Templates

The HTML DB engine constructs the appearance of an application using themes. A theme is a named collection of templates that defines the application user interface.

Each theme contains templates for every type of application component and page control, including individual pages, regions, reports, lists, labels, menus, buttons, and list of values. Templates contain HTML and variables that the HTML DB engine substitutes with dynamic values at runtime. The primary advantage of themes is that you can manage the appearance of components and controls in one central location.

See Also: "Managing Themes" on page 9-8 and "Customizing Templates" on page 9-16

How Themes and Page Templates Effect User Interface

The HTML DB engine creates the user interface of an application based on a named collection of templates called a theme. Each templates contains HTML and variables that the HTML DB engine substitutes at runtime with dynamic values. By using themes, you can manage the appearance of your application user interface in one central location.

You can also override the default page level template on a page by page basis. An application can have any number of page templates. If you do not specify a template, the HTML DB engine uses the default template specified in the current theme.

See Also: "Changing Default Templates in a Theme" on page 9-9 and "Selecting a Default Page Template" on page 9-16

Viewing Application Reports

Application Builder includes a number of reports to help you better manage your application.

To access application reports:

1. Navigate to the Workspace home page.
2. Select an application name from the Applications list.
3. From the Tasks list, select **View Application Reports**.

   Application Reports page appears. Reports are categorized as follows:

   - Shared Component Reports
   - Page Component Reports
   - Activity Reports
4. Select a report to review.

See Also: "Viewing Application and Schema Reports" on page 13-4 for information on other reports that are available to all developers

About the Database Object Dependencies Report

The Database Object Dependencies report identifies database objects referenced by the current application.

To view the Database Object Dependencies report:

1. Navigate to the Workspace home page.
2. Select an application name from the Applications list.
3. From the Tasks list, select **View Application Reports**.

   Application Reports page appears.
4. Under Shared Component Reports, select **Database Object Dependencies**.
5. To view the components that reference a specific database object, select the Reference Count number.

About the Region Source Report

Use the Region Source report to search through all region source in your application.

To view the Region Source report:

1. Navigate to the Workspace home page.
2. Select an application name from the Applications list.
3. From the Tasks list, select **View Application Reports**.

   Application Reports page appears.
4. Page Component Reports, select **Region Source**.
5. To view the components that reference a specific database object, select the Reference Count number.

8

Building an Application

This section describes how to use Application Builder to build an application and application components. It includes instructions for creating an application and adding pages as well as adding components (reports, charts, or forms), page controls (buttons, items, list of values), or a shared components (menus, lists, or tabs).

This section contains the following topics:

- Creating an Application
- Adding Additional Pages
- Creating Reports
- Creating Forms
- Creating Charts
- Creating Buttons
- Creating Items
- Creating Lists of Values
- Creating Calendars
- Utilizing Shortcuts
- Incorporating JavaScript into an Application
- Creating Dependent Select Lists
- Creating a Help Page

See Also:

- "Using Application Builder"
- "Controlling Page Layout and User Interface"
- "Adding Navigation"

Creating an Application

You create a new application in Oracle HTML DB using the Create Application Wizard. You delete an application from the Application Builder home page.

Topics in this section include:

- Creating a New Application
- Deleting an Application

Creating a New Application

You can use the Create Application Wizard to create a new application having up to nine pages.

To create an application using the Create Application Wizard:

1. Navigate to the Workspace home page.
2. Click the **Create Application** button.
3. Choose the method by which you want to create your application:
   - **From Scratch.** Enables you to define tabs, select a user interface (UI), and many other options.
   - **Based on an Existing Application.** Creates a copy of another application, including any authentication settings, but without any the pages. Select this option to create an application using the same user interface templates as an existing application.
   - **Based on an Existing Table.** Creates a complete application based on an existing table in your workspace. The resulting application includes a standard report, an insert form, an update form, a success form (indicates when a record is successfully inserted), an analysis menu page, analysis reports, analysis charts, and a login page.
   - **Demonstration Application.** Installs a demonstration application included with Oracle HTML DB.
   - **Based on Spreadsheet**. Creates an easily deployable application from a spreadsheet.
   - **From an Application Export File**. Uploads an application export file.
4. Follow the on-screen instruction.

   Once you create an application the next step is to specify application attributes.

   See Also: "Creating an Application Using the Create Application Wizard" on page 2-5, "Editing Application Attributes" on page 7-4, and "Importing Export Files" on page 12-10

Deleting an Application

You can delete an application from the Application Builder home page, or while editing application attributes. If you delete an application you also delete all defined components (reports, charts, or forms), page controls (buttons, items, list of values), or a shared components (menus, lists, or tabs).

To delete an application from Application Builder home page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. When Application Builder appears, verify the application name and the application ID at the top of the page.
4. From the Tasks list, select **Delete this Application**.
5. Follow the on-screen instructions.

To delete an application while editing application attributes:

1. Navigate to the Workspace home page.

2. From the Applications list, select an application.
3. When Application Builder appears, verify the application name and the application ID at the top of the page.
4. Click the **Edit Attributes** icon.

   The Edit Application Attributes page appears.
5. Verify the application ID and name.
6. Click **Delete** at the top of the page.

Adding Additional Pages

You can add a new page or add a component to an existing page by running the Create Page Wizard. You can access this wizard on the Application Builder home page, the Page Definition, or by selecting the **New...** link on the Developer toolbar.

When you run the Create Page Wizard, you choose whether to create a blank page or a page with a component. Select Page with Component to create a page containing a report, a chart, a form, a wizard, a calendar, or a tree.

Note: You can add a component (that is, a report, chart, form, wizard, a calendar, or tree) to an existing page using the Create Page Wizard. When prompted, specify an existing page number.

Topics in this section include:

- Creating a Page from Application Builder
- Creating a Page from the Page Definition
- Creating a Page from the Developer Toolbar
- Running a Page
- Grouping Pages
- Locking and Unlocking Page
- Deleting a Page

See Also: "Creating Reports" on page 8-8, "Creating Charts" on page 8-24, "Creating Forms" on page 8-17, "Creating Calendars" on page 8-43, and "Creating Trees" on page 8-43

Creating a Page from Application Builder

To create a new page from the Application Builder home page:

1. Navigate to Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Create Page**.
4. Select the type of page you wish to create:
   - **Blank Page**
   - **Page with Component**

Select **Page with Component** to create a page containing a report, a chart, a form, a wizard, a calendar, or a tree. Selecting this option creates complete pages containing multiple attributes. For example, the Report wizard generates one page containing one region and multiple buttons.

5. Follow the on-screen instructions.

Creating a Page from the Page Definition

To create a new page while viewing a Page Definition:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. From the navigation bar at the top of the page, click the **Create** button.
3. Select the type of page you wish to create:
   - **Blank Page**
   - **Page with Component**

     Select **Page with Component** to create a page containing a report, a chart, a form, a wizard, a calendar, or a tree. Selecting this option creates complete pages containing multiple attributes. For example, the Report wizard generates one page containing one region and multiple buttons.
4. Follow the on-screen instructions.

See Also: Editing a Page Definition on page 7-15

Creating a Page from the Developer Toolbar

Users who log in to Oracle HTML DB having developer privileges have access to the Developer toolbar. The Developer toolbar displays at the bottom every page and offers a quick way create a new page.

To create a new page from the Developer toolbar:

1. On the Developer toolbar, select **New**.

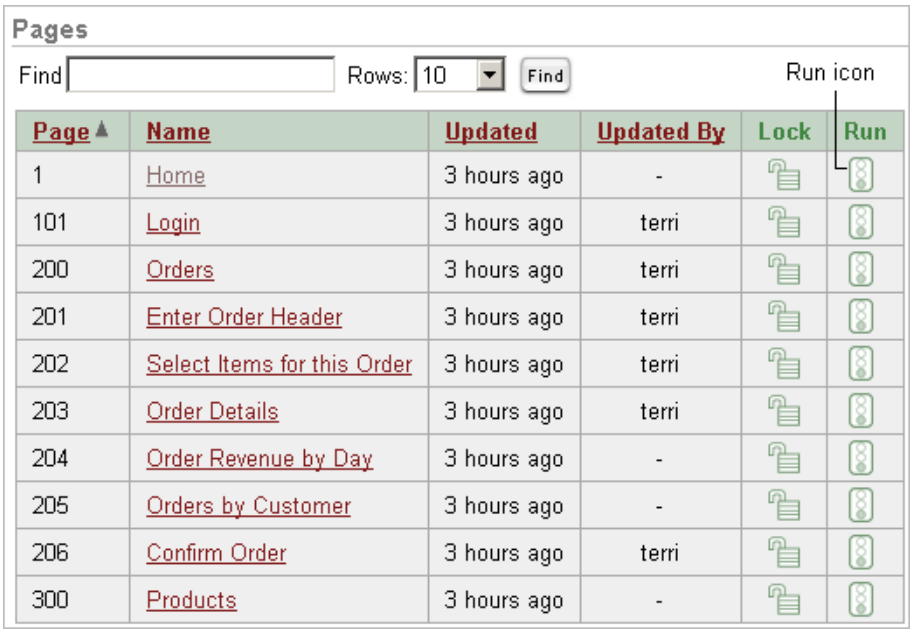
   The New Component Wizard appears.
2. Select the type of page you want to create:
   - Page (Blank Page)
   - Component (Report, Form, Chart)
   - Page Control (Region, Item, Button)
   - Shared Control (Menu, List, Tab)
3. Follow the on-screen instructions.

See Also: "Using the Developer Toolbar" on page 7-14

Running a Page

The HTML DB engine dynamically renders and process pages based on data stored in database tables. To view a rendered version of your application, you run or submit it to the HTML DB engine. As you create new pages you can run them individually, or run an entire application. You can run a page from numerous locations within Application Builder by clicking the Run icon. (See Figure 8–1.)

Figure 8–1 Run Icon on the Pages List



To run a specific page from the Pages list:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears. The list of pages for the selected application appears at the bottom of the page.
3. From the Pages list, locate the page you wish to run and click the **Run** button in the far right column.

To run a specific page from the Page Definition:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Click the **Run** button in the upper right corner of the page.

To run an entire application:

1. Navigate to the Workspace home page.
2. Locate the application in the Applications list.
3. Click the **Run** button in the far right column.

Grouping Pages

You can make the pages within your application easier to access by organizing them into page groups.

Topics in this section include:

- Creating a Page Group
- Assigning Pages to a Page Group
- Viewing the Page Group Report

Creating a Page Group

To create a page group:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. From the Tasks List on the right side of the page, select **Manage Page Groups**.
4. On the Page Groups page, click **Create**.
5. Enter a name, a description (optional), and click **Create**.

Assigning Pages to a Page Group

To assign pages to page group:

1. Navigate to the Workspace home page.
2. From the Applications list, select the application.
3. From the Tasks List on the right side of the page, select **Manage Page Groups**.
4. From the Tasks list, select **Report Unassigned Pages**.

   The Unassigned Pages page appears.
5. From Page Group, select a group to assign pages to.
6. Select the pages to be assigned.
7. Click **Assigned Checked**.

   Selecting the page number takes you to the Page Attributes page. Selecting the Page Name links to the Page Definition.

Viewing the Page Group Report

The Page Group Report offers a comprehensive list of which pages in an application are assigned to a group and which pages are unassigned.

To view the Page Group Report:

1. Navigate to the Workspace home page.
2. From the Applications list, select the application.
3. From the Tasks List on the right side of the page, select **Manage Page Groups**.
4. On the Page Groups page, click **Report Page Groups**.

Locking and Unlocking Page

You can prevent conflicts during application development by locking pages of your application. By locking a page, you prevent other developers from editing it.

To lock a page of your application:

1. Navigate to the Workspace home page.
2. From the Applications list, select the application.
3. To lock pages from the Locked Pages page:
   a. From the Tasks List on the right side of the page, select **Manage Page Locks**.
   b. Select the appropriate pages and click **Lock**.
   c. Enter a comment in the Comment field.

d. Click **Lock Pages**.

4. To lock pages from the Pages list:
   a. Click the **Lock** icon in the Pages list.
   b. Enter a comment in the Comment field.
   c. Click **Lock Pages**.
5. To lock pages from the Page Definition:
   a. Navigate to the Page Definition. (See "Viewing a Page Definition" on page 7-11.)
   b. Click the **Lock** icon in the upper right corner above Shared Components.

      The Locked Pages page appears.
   c. Select the appropriate pages and click **Lock**.
   d. Enter a comment in the Comment field.
   e. Click **Lock Pages**.

Only the user who locked a page can unlock it.

To unlock a page of your application:

1. Navigate to the Workspace home page.
2. From the Applications list, select the application.
3. To unlock pages from the Locked Pages page:
   a. From the Tasks List on the right side of the page, select **Manage Page Locks**.
   b. Select the appropriate pages.
   c. Click **UnLock**.
4. To unlock pages from the Pages list:
   a. Click the **Lock** icon in the Pages list.

      The Edit Page Lock page appears.
   b. Click **UnLock**.
5. To unlock pages from the Page Definition:
   a. Navigate to the Page Definition. (See "Viewing a Page Definition" on page 7-11.)
   b. Click the **Lock** icon in the upper right corner above Shared Components.

      The Locked Pages page appears.
   c. Select the appropriate pages.
   d. Click **UnLock**.

Note the Lock icon changes to reflect the lock status of a given page.

Accessing Alternative Locked Pages Views

You can access a number of different views of Locked Pages on the Locked Pages page.

To access different views of locked pages:

1. Navigate to the Workspace home page.

2. From the Applications list, select the application.
3. From the Tasks List on the right side of the page, select **Manage Page Locks**.
4. From the Tasks list, select one of the following:
   - **Show Locked Pages** displays only locked pages within the current application
   - **Show All Pages** displays all pages within the current application
   - **Show Unlocked Pages** display only unlocked pages within the current application
   - **Administer Locks** enables workspace administrators to unlock any pages locked by a developer

Deleting a Page

You can delete a page from the Page Definition or while editing page attributes.

To delete a page from the Page Definition:

1. Navigate to the Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Verify the page name.
3. From the navigation bar at the top of the page, click **Delete**.
4. Follow the on-screen instructions.

> **See Also:** "Editing a Page Definition" on page 7-15 for more information on editing page attributes

To delete a page while editing page attributes:

1. Navigate to the Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. From the navigation bar at the top of the page, click **Edit Attributes**.
3. Verify the application ID and page name.
4. Click **Delete**.
5. Follow the on-screen instructions.

Creating Reports

In Oracle HTML DB a report is the formatted result of a SQL query. You can generate reports by selecting and running a built-in query, or by defining a report region based on a SQL query.

Topics in this section include:

- Creating a Report Using a Wizard
- Editing Report Attributes
- Controlling Report Pagination
- Enabling Column Sorting
- Exporting a Report as a CSV or XML File
- Creating a Column link
- Defining an Updatable Column

- Defining a Column as a List of Values
- Controlling When Columns Display
- Controlling Column Breaks

Creating a Report Using a Wizard

Oracle HTML DB includes the number of built-in wizards for generating reports.

To create a report using a wizard:

1. Navigate to the Workspace home page.
2. From the Applications list, select the application.

   Application Builder appears.
3. Click **Create Page**.
4. Select **Page with Component**.
5. Select **Report**.
6. Select one of the following report types:
   - **Easy Report** - Does not require any SQL knowledge. Simply select the appropriate schema, table, columns, and result set display.
   - **Report with Form** - Builds a two page report and form combination. The first page enables users to specify the row to be updated. The second page includes a form for updating the selected table or view.
   - **SQL Report** - Creates a report based on a custom SQL SELECT statement or a PL/SQL function returning a SQL SELECT statement that you provide.
7. Follow the on-screen instructions.

Editing Report Attributes

You can use the Report Attributes and Column Attributes pages to precisely control the look and feel of report pages. For example, you can use these attributes to alter column heading text, change column positioning, hide a column, create a sum of a column, or select a sort sequence.

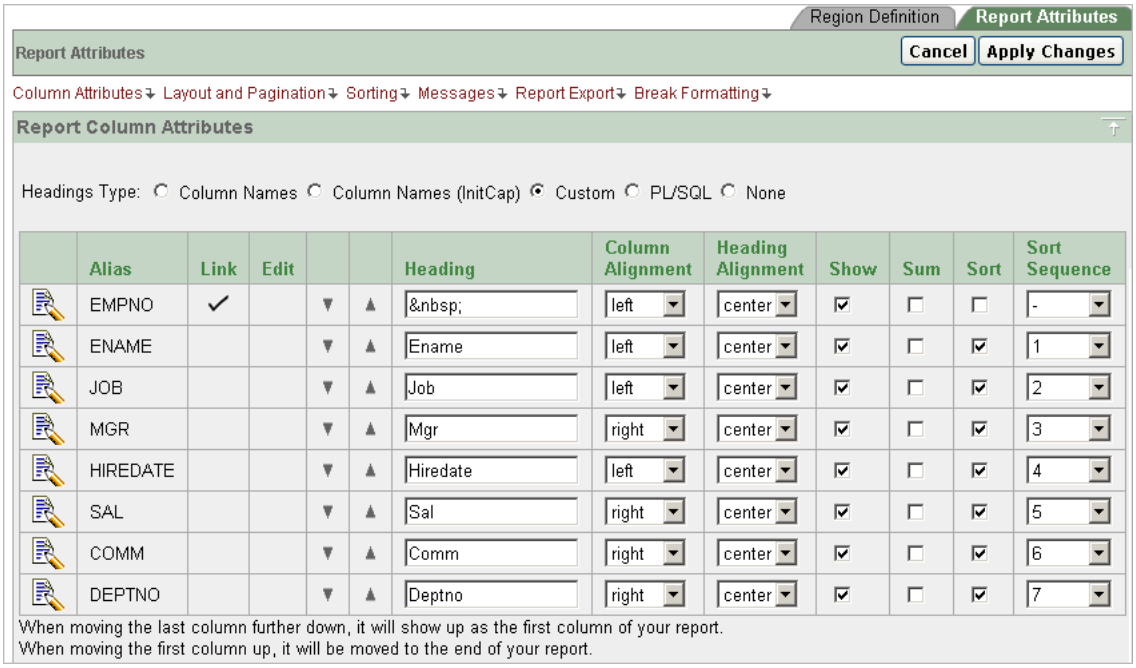
On the Page Definition, you can access the Report Attributes page by clicking either Q or RPT adjacent to the report region you want to edit. **Q** indicates the report is a regular report and **RPT** indicates the report is an easy report. You can also navigate to the Report Attributes page by clicking the region name then selecting the Report Attributes tab.

To access the Report Attributes page:

1. Navigate to the Workspace home page.
2. From the Applications list, select the application.
3. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
4. Under Regions, click **Q** next to the name of the report region you want to edit.

   The Report Attributes page appears.

Figure 8–2 Report Attributes Page



Heading Type identifies how the heading was generated for the report. Use the Report Column Attributes section to control report column appearance and functionality. The Link column indicates whether a column link is currently defined. The Edit column indicates whether or not a column is currently updatable. The Edit column.

Table 8–1 describes common report column edits.

Table 8–1 Common Report Column Edits

Description	Developer Action
Alter column display sequence.	Click the up and down arrows.
Alter heading alignment.	Under Column Alignment, select a new column alignment.
Change column heading text.	Under Heading, enter a different heading text.
Control which columns display.	Click **Show** to indicate a column should display.
Enable an unique sort sequence.	Click **Sort** and select a sequence number from Sort Sequence.
Enable the sum of a column.	Click **Sum** to enable the sum of a column.

You can further refine the attributes of a specific column on the Column Attributes page.

5. To access the Column Attributes page, click the **Edit** icon adjacent to the appropriate column name.

   See online help for more information on a specific attribute.

Controlling Report Pagination

You control report pagination by:

- Including a pagination substitution string in the report template

- Making selections from Layout and Pagination on the Report Attributes page

You control how pagination displays by making selections from the Layout and Pagination attributes on the Report Attributes page.

To access the Layout and Pagination section of the Report Attributes page:

1. Create a report. (See "Creating a Report Using a Wizard" on page 8-9.)
2. Under Regions, click the appropriate report attributes link (**Q** or **RPT**).

   The Report Attributes page appears.
3. Scroll down to Layout and Pagination.

   You use the Layout and Pagination attributes to select a pagination style, determine where pagination displays, and specify the number of rows that display on each page. Table 8–2 describes the most commonly used Layout and Pagination attributes.

Table 8–2 Layout and Pagination Attributes

Attribute	Description
Report Template	Specifies a template to be applied to this report. Report templates provide control over the results of a row from your SQL query. You can choose from a number of default templates, or pick a custom build template.
Pagination Scheme	Specifies a pagination scheme for this report. Pagination provides the user with information about the number of rows and the current position within the result set. Pagination also defines the style of links or buttons used to navigate to the next or previous page. For more information, see the help for this item.
Display Position	Defines where pagination displays. If you choose to display pagination above a report, the selected report template needs to support that type of display.
Number of Rows	Defines the maximum number of rows to display on each page.
Strip HTML	Specifies whether or not to remove HTML tags from the original column values for HTML expressions and column links. If you select values from the database that already contain HTML tags, then those tags can cause conflicts with the HTML generated for your columns links or HTML expressions. When this option is enabled, only the actual data portion of your column value is used.

Including Pagination After the Rows in a Report

To include pagination after the rows in a report:

1. Create a report. (See "Creating a Report Using a Wizard" on page 8-9.)

   Next, select the appropriate Layout and Pagination attributes.
2. Navigate to the Report Attributes page:
   a. Navigate to the Page Definition.
   b. Under Regions, click the appropriate report attributes link (**Q** or **RPT**)

      The Report Attributes page appears.
3. Under Layout and Pagination, select the following:

a. Report Template - Select a report template (optional).
b. Pagination Scheme - Select a pagination scheme.
c. Display Position - Select a display position.
d. Number of Rows - Specify how many rows display on each page.
e. Click **Apply Changes**.

4. Edit the report template.
   a. Navigate to the Page Definition.
   b. Under Templates, select the report template name.
   c. Include the `#PAGINATION#` substitution string in the After Rows attribute.
   d. Click **Apply Changes**.
5. Run the page.

Including Pagination Before the Rows in a Report

To include pagination after the rows in a report:

1. Create a report. (See "Creating a Report Using a Wizard" on page 8-9.)

   Next, select the appropriate Layout and Pagination attributes.
2. Navigate to the Report Attributes page:
   a. Navigate to the Page Definition.
   b. Under Regions, click the appropriate report attributes link (**Q** or **RPT**).

      The Report Attributes page appears.
3. Under Layout and Pagination:
   a. Report Template - Select a report template (optional).
   b. Pagination Scheme - Select a pagination scheme.
   c. Display Position - Select a position that contains the word "top."
   d. Number of Rows - Specify how many rows display on each page.
   e. Click **Apply Changes**.
4. Edit the report template.
   a. Navigate to the Page Definition.
   b. Under Templates, select the report template name.
   c. Include the `#TOP_PAGINATION#` substitution string in the Before Rows attribute.
   d. Click **Apply Changes**.
5. Run the page.

Enabling Column Sorting

You enable column sorting on the Report Attributes page.

To enable column sorting:

1. Navigate to the Report Attributes page. (See "Editing Report Attributes" on page 8-9.)

2. Under Report Column Attributes, select the **Sort** check box adjacent to the columns to be sorted.
3. From Sort Sequence, select a sequence number.

   Sort Sequence is optional. However, if there are one or more sort enabled columns, then at least one column needs a defined Sort Sequence.
4. Scroll down to Sorting.
5. Specify ascending and descending image attributes or click **set defaults**.

Exporting a Report as a CSV or XML File

You can create a link within a report that enables users to export the report as either a comma delimited file (.csv) or XML file. You specify an export format by selecting a report template.

To specify an export report template:

1. Navigate to the appropriate Report Attributes page. (See "Editing Report Attributes" on page 8-9.)
2. Scroll down to Layout and Pagination.
3. From the Report Template list, select one of the following:
   - **export: CSV** exports the report as a CSV file
   - **export: XML** exports the report as a XML file

   Selecting either option prevents the HTML DB engine from rendering the page and dumps the content to either a CSV or XML file.

   You can use the options under CSV Output to create a link that downloads the content of the report.
4. Scroll down Report Export.
5. From Enable SCV output, select **Yes**.
6. (Optional) In the Separator and Enclosed By fields, define the separator and delimiter characters.

   The default Enclosed By by characters are a double quotes (" "). The default delimiter character is either a comma or a semicolon depending upon your current NLS settings.
7. In the Link Label field, enter link text. This text will display in your report and enable users to invoke a download.
8. (Optional) To specify a default export filename, enter a name in the Filename field.

   By default, the HTML DB engine creates an export filename by taking the region name and adding the appropriate filename extension (`.csv` or `.xml`).
9. Click **Apply Changes**.

Creating a Column link

Use the Column Link attributes to create a link from a report to another page in your application or to a URL.

To create a column link to another page:

1. Navigate to the appropriate Report Attributes page. (See "Editing Report Attributes" on page 8-9.)
2. Under Report Column Attributes, locate the column to contain the link.
3. Click the **Edit** icon adjacent to the column name.

   The Column Attributes page appears.
4. Scroll down to Column Link.
5. To create a Column Link to another page:
   a. From Target, select **Page in this Application**.
   b. (Optional) In Link Attributes, specify additional column link attributes that will be included in the `<a href= >` tag (for example, a link target, classes, or styles).
   c. In Link Text, enter text to be displayed as a link, specify an image tag, or pick from the list of default images.
   d. In Page, specify the target page number. To reset pagination for this page, select **Reset Pagination**.
   e. In Request, specify the request to be used.
   f. In Clear Cache, specify the pages (that is, the page numbers) on which to clear cache. You can specify multiple pages by listing the page numbers in a comma delimited list.
   g. Use the Name and Value fields to specify session state for a specific item.
6. Click **Apply Changes**.

To create a column link to a URL:

1. Navigate to the appropriate Report Attributes page. (See "Editing Report Attributes" on page 8-9.)
2. Access the Column Attributes page by clicking the **Edit** icon adjacent to the appropriate column.

   The Column Attributes page appears.
3. Scroll down to Column Link.
4. Under Column Link, make the following selections
   a. From Target Type, select **URL**.
   b. In Link Text, enter text to be displayed as a link, select a substitution string.
   c. (Optional) In Link Attributes, specify additional column link attributes that will be included in the `<a href= >` tag (for example, a link target, classes, or styles).
   d. In URL, enter the appropriate address.
5. Click **Apply Changes**.

Defining an Updatable Column

You can make a column updatable by editing Tabular Form Element attributes on the Column Attributes page. Note that the HTML DB engine can only perform updates if a multi row update is defined, or PL/SQL process is implemented to process updated data.

To define updatable column attributes:

1. Navigate to the appropriate Report Attributes page. (See "Editing Report Attributes" on page 8-9.)
2. Access the Column Attributes page by clicking the **Edit** icon adjacent to the appropriate column.

   The Column Attributes page appears.
3. Scroll down to Tabular Form Element.
4. Under Tabular Form Element, make the following selections:
   a. From Display As, select a type of updatable column.

      Use this option to make a column updatable. Updates can only be performed if a multi row update is defined, or PL/SQL process is implemented to process updated data.
   b. From Date Picker Format Mask, make a selection if you selected the Display As type of **Date Picker**.
   c. In Element Width, specify the width of the form item.
   d. In Number of Rows, specify the height of a form item (applicable to text areas).
   e. In Element Attributes, define a style or standard form element attribute.
   f. In Element Option Attributes, specify form element attributes for items in a radio group or check box.
   g. From Default Type, identify the default type.
   h. In Default, identify the default source.

      Use Default Type and Default Source to add new rows to a tabular form. The HTML DB engine uses this setting to determine the default value for the column. Otherwise, the column is defined as NULL.
   i. From Reference Table Owner, identify the owner of the referenced table. Use this attribute to build User Interface Defaults for reports.
   j. In Reference Table Name, identify the table or view that contains the current report column.
   k. In Reference Column Name, identify the column name that this report column references
5. Click **Apply Changes**.

Defining a Column as a List of Values

Report columns may be rendered as lists of values. For example, a column can be rendered using a select list or a popup list of values. Or, a column can also be rendered as read-only text based on a list of values.

This last approach is an effective strategy when creating display lookup values and is particularly useful in regular, non-updatable reports. This approach enables you to display the value of a column without having to write a SQL `JOIN`.

To render a report column as a list of values:

1. Navigate to the appropriate Report Attributes page. (See "Editing Report Attributes" on page 8-9.)

2. Access the Column Attributes page by clicking the **Edit** icon adjacent to the appropriate column.

   The Column Attributes page appears.

3. Scroll down to List of Values.
4. From Named LOV, make a selection from the Named List of Values repository.
5. To include a null value in a list of values:
   a. In Display Null, select **Yes**.
   b. In Null Text, specify the value that displays.

   A column may also have a value that does not display in its list of values.
6. To define a value that does not display in the list of values:
   a. From Display Extra Value, select **Yes**.

      The extra value is used if the actual column value is not part of the LOV. In that situation, the actual value is shown. If you do not display extra values, you may end up with the wrong value and unintentionally and update your data incorrectly.
   b. In Null Value, specify the value that displays.
   c. If you have not selected a Named LOV, enter the query used to display a select list in the LOV Query field.
7. If you have not selected a Named LOV, enter the query used to display a select list in LOV Query.
8. Click **Apply Changes**.

See Also: "Creating Lists of Values" on page 8-41

Controlling When Columns Display

You can use the Authorization and Conditional Display column attributes to control when a column displays.

Authorization enables you to control access to resources (such as a report column) based on predefined user privileges. For example, you could create an authorization scheme in which only managers can view a specific report column. Before you can select an authorization scheme, you must first create it.

A condition is a small unit of logic that enables you to control the display of a column based on a predefined condition type. The condition evaluates to true or false based on the values you enter in the Expressions fields.

To specify Authorization and Conditional Display attributes:

1. Navigate to the appropriate Report Attributes page. (See "Editing Report Attributes" on page 8-9.)
2. Access the Column Attributes page by clicking the **Edit** icon adjacent to the appropriate column.

   The Column Attributes page appears.
3. Under Authorization, make a selection from the Authorization Scheme list.
4. Under Conditional Display, make a selection from the Condition Type list and depending upon your selection, enter an expression or value in the appropriate Expression fields.

If the authorization is successful and the condition type display evaluates to true, the column displays.

See Also:

- "Providing Security Through Authorization" on page 14-9
- "Understanding Conditional Rendering and Processing" on page 6-7
- Appendix A, "Available Conditions" on page A-1

Controlling Column Breaks

You can control whether a specific column repeats and how column breaks appear when printed using Break Formatting attributes. For example, suppose your report displays employee information by department number. If multiple employees are members of the same department, you can increase the readability by specifying the department number only appears once.

To create this type of column break:

1. Navigate to the appropriate Report Attributes page. (See "Editing Report Attributes" on page 8-9.)
2. Scroll down to Break Formatting.
3. Make a selection from the Breaks list.

Creating Forms

You can include a variety of different types of forms in your applications. You can include forms that enable users to update just a single row in a table or multiple rows at once. Oracle HTML DB includes a number of wizards you can use to create forms automatically, or you can create forms manually.

Topics in this section include:

- Creating a Form Using a Wizard
- Creating a Tabular Form
- Building a Master Detail Form
- Creating a Form Manually
- Validating User Input in Forms

Creating a Form Using a Wizard

The easiest way to create a form is to use a wizard. For example, the Form on Table or View Wizard creates one item for each column in a table. It also includes the necessary buttons and processes required to insert, update, and delete rows from the table using a primary key. Each region has a defined name and display position all other attributes are items, buttons, processes, and branches.

To create a form using a wizard:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears.

3. Click **Create Page**.
4. Select **Page with Component**.
5. Select **Form**.
6. Under Forms, select a wizard described in Table 8–3.

Table 8–3 Forms Wizards

Wizard	Description
Form on a Procedure	Builds a form based on stored procedure arguments. Use this approach when you have implemented logic or DML (Data Manipulation Language) in a stored procedure or package.
Form on a SQL Query	Creates a form based on the columns returned by a SQL query such as an equijoin.
Form on a Table or View	Creates a form that enables users to update a single row in a database table.
Form on a Table with Report. 2 pages	Creates two pages. One page displays a report. Each row provides a link to the second page to enable users to update each record. **Note:** This wizard does not support tables having more than 127 columns. Selecting more than 127 columns will generate an error.
Form on Web Service	Creates a page with items based on a Web service definition. This wizard creates a user input form, a process to call the Web service, and a submit button. **See Also:** "Creating a Form on a Web Service" on page 15-19
Form and Report on Web Service	Creates a page with items based on a Web service definition. This wizard creates a user input form, a process to call the Web service, a submit button, and displays the results returned in a report. See Also: "Creating an Input Form and Report on a Web Service" on page 15-17
Master Detail Form	Creates a form which displays a master row and multiple detail rows within a single HTML form. With this form, users can query, insert, update, and delete values from two tables or views. **See Also:** "Building a Master Detail Form" on page 8-20
Summary Page	Creates a read-only version of a form. Typically used to provide a confirmation page at the end of a wizard.
Tabular Form	Creates a form in which users can update multiple rows in a database. **See Also:** "Creating a Tabular Form" on page 8-18

7. Follow the on-screen instructions.

Creating a Tabular Form

A tabular form enables users to update multiple rows in a table. The Tabular Form Wizard creates a form to perform update, insert, and delete operations on multiple rows in a database table.

To create a tabular form:

1. From the Workspace home page, select the appropriate application.

2. Click **Create Page**.
3. Select **Page with Component**.
4. Select **Form**.
5. Select **Tabular Form**.

   The Tabular Form Wizard appears.
6. On Identify Table/View Owner:
   a. Specify the table or view owner on which you want to base your tabular form.
   b. Select the type of tabular form you want to create.
   c. Select a table.
7. On Identify Table/View Name, select a table.
8. On Identify Columns to Display:
   a. Specify whether to use user interface defaults.

      User interface defaults enable you to assign default user interface properties to a table, column, or view within a specified schema.
   b. Select the columns (updatable and nonupdatable) to include in the form.

      You can modify the column order or your SQL query after you have created the page.
9. On Identify Primary Key, select the Primary Key column and a foreign key column (if applicable).
10. On Defaults for Primary and Foreign Keys, select a source type for the primary key column. Valid options include:
    - **Existing trigger** - Select this option if a trigger is already defined for the table. You do not need to define a primary key column source for existing columns. Existing columns already have a primary key.
    - **Custom PL/SQL function** - Select this option if the source is a PL/SQL function returning the key value.
    - **Existing sequence** - Select this option if an existing sequence is defined for the table. If that is the case, the sequence next value would be used.
11. On Updatable Columns, select which columns should be updatable.
12. On Identify Page and Region Attributes.
    a. Specify page and region information.
    b. Select a region template.
    c. Select a report template.
13. On Identify Tab, specify a tab implementation for this page.
14. On Button Labels, enter the display text to appear on the Cancel, Submit, Delete, and Add Row buttons.
15. On Identify Branching, specify the pages to branch to after the user clicks the Submit and Cancel buttons.
16. Click **Finish**.

Note: Do not modify the select list of a SQL statement of a tabular form after it has been generated. Doing so can result in a checksum error when you alter data in the form.

See Also: "Managing User Interface Defaults" on page 5-12

Building a Master Detail Form

A master detail form reflects a one-to-many relationship between two tables in a database. Typically a master detail form displays a master row and multiple detail rows within a single HTML form. With this form, users can insert, update, and delete values from two tables or views.

To create a master detail form:

1. From the Workspace home page, select the appropriate application.
2. Click **Create Page**.
3. Select **Page with Component**.
4. Select **Form**.
5. Select **Master Detail Form**.

   The Master Detail Wizard appears.
6. On Define Master Table:
   a. Select a table or view owner
   b. Select a table or view name
   c. Select the columns to display
7. On Define Detail Table:
   a. Specify to show only related tables
   b. Select the table or view owner
   c. Select the table or view name
   d. Select the columns to display
8. On Define Primary Key, select the primary key column for the master table and then the primary key column for the detail table.
9. On Define Master and Detail, define the relationships between master and detail tables.
10. Specify the source for the master table and detail table primary key columns.
11. On Define Master Options, specify whether to include master row navigation.

    If you include master row navigation, define navigation order columns. If a navigation order column is not defined, the master update form will navigate by the primary key column.
12. On Choose Layout, specify the layout of the master detail pages.

    You can include the master detail as a tabular form on the same page, or add the master detail on a separate page.
13. On Page Attributes, review and edit the master page and detail page information.

14. On Identify Tabs, specify whether to include an optional tab set.
15. Click **Create**.

Creating a Form Manually

You can also create a form manually by performing the following steps:

- Create an HTML region (to serve as a container for your page items)
- Create items to display in the region
- Create processes and branches

To create a form manually:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Create an HTML region:
   - Under Regions, click **Create**
   - Select the region type **HTML**
   - Follow the on-screen instructions
3. Start adding items to the page:
   - Under Items, click **Create**
   - Follow the on-screen instructions

Processing a Form

Once you create a form, the next step is to process the data a user types by inserting into or updating the underlying database tables or views. There are three ways to process a form:

- Creating an Automatic Row (DML) Processing Process
- Creating a Process that Contains One or More Insert Statements
- Using a PL/SQL API to Process Form Values

Creating an Automatic Row (DML) Processing Process

One common way to implement a form is to manually create an Automatic Row Processing (DML) process. This approach offers three advantages. First, you are not required to provide any SQL coding. Second, Oracle HTML DB performs DML processing for you. Third, this process automatically performs lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

In order to implement this approach you need to:

- Add items and define the Item Source Type as Database Column and specify a case-sensitive column name
- Select the option **Always overrides the cache value**

To create an Automatic Row Processing (DML) process:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Processes, click the **Create** icon.

3. Select the process type **Data Manipulation**.
4. Select the process category **Automatic Row Processing (DML)**.
5. Specify the following process attributes:
   a. In the Name field, type a name to identify the process.
   b. In the Sequence field, specify a sequence number.
   c. From the Point list, select the appropriate processing point. In most instances, select **Onload - After Header**.
   d. From the Type list, select **Automated Row Processing (DML)**.
6. Follow the on-screen instructions.

Creating a Process that Contains One or More Insert Statements

In this approach to form handling, you create one or more processes to handle insert, update and delete actions. Instead of having the HTML DB engine handling everything transparently, you are in complete control.

For example, suppose you have a form with three items:

- `P1_ID` - A hidden item to store the primary key of the currently displayed row in a table.
- `P1_FIRST_NAME` - A text field for user input.
- `P1_LAST_NAME` - A text field for user input.

Assume also there are three buttons labeled Insert, Update, and Delete. Also assume you have a table T which contains the columns `id`, `first_name`, and `last_name`. The table has a trigger which automatically populates the `ID` column when there is no value supplied.

To process the insert of a new row, you create a conditional process of type PL/SQL that executes when the user clicks the Insert button. For example:

```
BEGIN
  INSERT INTO T ( first_name, last_name )
     VALUES  (:P1_FIRST_NAME, :P1_LAST_NAME);
END;
```

To process the update of a row, you create another conditional process of type PL/SQL. For example:

```
BEGIN
    UPDATE T
       SET first_name = :P1_FIRST_NAME,
           last_name = :P1_LAST_NAME
    WHERE ID = :P1_ID;
END;
```

To process the deletion of a row, you create a conditional process that executes when the user clicks the Delete button. For example:

```
BEGIN
    DELETE FROM T
    WHERE ID = :P1_ID;
END;
```

Using a PL/SQL API to Process Form Values

For certain types of applications it is appropriate to centralize all access to tables in a single or few PL/SQL packages. If you have created a package to handle DML operations, you can call procedures and functions within this package from a After Submit PL/SQL process to process insert, update and delete requests.

See Also: "Oracle HTML DB APIs" on page 17-1

Populating Forms

Oracle HTML DB populates a form on load, or when the HTML DB engine renders the page. You can populate a form in the following ways:

- Create a process and define the type as Automated Row Fetch
- Populate the form manually by referencing a hidden session state item

To create an Automated Row Fetch process:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Processes, click **Create**.
3. Select the process type **Data Manipulation**.
4. Select the process category **Automatic Row Fetch**.
5. Specify the following process attributes:
   a. In the Name field, type a name to identify the process.
   b. In the Sequence field, specify a sequence number.
   c. From the Point list, select the appropriate processing point.
   d. From the Type list, select **Automated Row Fetch**.
6. Follow the on-screen instructions.

You can also populate a form manually by referencing a hidden session state item. For example, the following code in an Oracle HTML DB process of type PL/SQL would set the values of `ename` and `sal`. The example also demonstrates how to manually populate a form by referencing a hidden session state item named `P2_ID`.

```
FOR C1 in (SELECT ename, sal
FROM emp WHERE ID=:P2_ID)
LOOP
     :P2_ENAME := C1.ename;
     :P2_SAL := C1.sal;
END LOOP;
```

In this example:

- `C1` is an implicit cursor
- The value of `P2_ID` has already been set
- The process point for this process would be set to execute (or fire) on or before **Onload - Before Regions**

Validating User Input in Forms

You can use validations to check data a user types prior to processing. Once you create a validation and the associated error message, you can associate it with a specific item.

You can choose to have validation error messages display inline (that is, on the page where the validation is performed) or on a separate error page.

Creating an inline error message involves these steps:

- Create a new validation and specify error message text
- Associate the validation with a specific item

To create a new validation:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Validations, click the **Create** icon.
3. When the Create Validations Wizard appears, follow the on-screen instructions.

   Validations Types are divided into two categories:

   - **Item.** These validations start with the phrase "Item" and provide common checks you may want to perform on the item that the validation is associated with.
   - **Code.** These validations require you provide either a piece of PL/SQL code or SQL query that defines the validation logic. Use this type of validation to perform custom validations that require verifying values of more than one item or accessing additional database tables.
4. Follow the on-screen instructions.

   Note: Validations may not contain more than 3,950 characters.

To associate an item with a validation and specify error message text:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Validations, select the validation item you want to associate.

   The attributes page for the validation appears.
3. Scroll down to Error Messaging:
   - In Error message display location, verify the display location
   - In Associated Item, select the item you want to associate with this validation
4. Click **Apply Changes**.

About Error Messaging

Error message display location identifies where a validation error message displays. Validation error messages can display on an error page or inline within the existing page. Inline error messages can display in a notification area (defined as part of the page template) or within the field label.

If you want to create a hard error that stops processing of any remaining validations, you must display the error on an error page.

Creating Charts

Oracle HTML DB includes built-in wizards for generating HTML and Scalable Vector Graphics (SVG) charts. Oracle HTML DB supports two types of graphical charts:

- HTML
- SVG

SVG is an XML-based language for Web graphics from the World Wide Web Consortium (W3C). SVG charts are defined using an embed tag. When evaluating whether a SVG chart is the appropriate chart type for your application remember that:

- Some Web browsers do not support SVG charts
- Most Web browsers that support SVG charts require users download an SVG plug-in

Topics in this section include:

- About SVG Plug-in Support
- About Creating Charts
- Creating a New Chart
- Editing Chart Attributes
- Understanding Chart CSS Classes
- Referencing a Custom Cascading Style Sheet
- Specifying Custom CSS Styles Inline
- Enabling Asynchronous Updates
- Displaying Charts in Other Languages

About SVG Plug-in Support

The Adobe SVG plug-in can properly handle data encoded in `UTF-8`, `UTF-16`, `ISO-8859-1`, and `US-ASCII`. Encoding of an SVG chart is determined by the database access descriptor (DAD) database character set. If the DAD character set is not `UTF8`, `AL32UTF8`, `AL16UTF16`, `WE8ISO8859P1`, or `US7ASCII`, SVG charts may not render properly in the Adobe SVG plug-in.

About Creating Charts

You define a chart in Oracle HTML DB using a wizard. For most chart wizards, you select a chart type and provide a SQL query using the following syntax:

```
SELECT link, label, value
FROM   ...
```

Where:

- `link` is a URL
- `label` is the text that displays in the bar
- `value` is the numeric column that defines the bar size

For example:

```
SELECT null, ename, sal
FROM   scott.emp
WHERE  deptno = :P101_DEPTNO
```

To create a dial chart, select a dial chart type and provide a SQL query using the following syntax:

```
SELECT value , maximum_value [ ,low_value [ ,high_value] ]
FROM   ...
```

Where:

- `value` is the starting point on the dial
- `maximum_value` is the possible highest point on the dial
- `low_value` and `high_value` are the historical low and high values

For example:

```
select dbms_random.value(500, 1200), 1300, dbms_random.value(100, 200)
FROM DUAL
```

Table 8–4 describes the chart types available in Oracle HTML DB.

Table 8–4 Available Chart Types

Chart Type	Description
Bar (HTML)	Bar chart showing one data series with each data point represented by a bar. HTML-based. Does not require a plug-in.
Bar, Horizontal	Single series based bar chart oriented horizontally with each data point in series represented by a bar. SVG-based. Requires a SVG plug-in.
Bar, Vertical	Single series based bar chart oriented vertically with each data point in series represented by a bar. SVG-based. Requires a SVG plug-in.
Cluster Bar, Horizontal	Multiple series based bar chart oriented horizontally and clustered by a common variable with each data point in series represented by a bar (for example, *Department sales total clustered by month of year*). SVG-based. Requires a SVG plug-in.
Cluster Bar, Vertical	Multiple series based bar chart oriented vertically clustered by a common variable with each data point in series represented by a bar (for example, *Department sales total clustered by month of year*). SVG-based. Requires a SVG plug-in.
Dial - Sweep	Also known as an angular gauge, this chart shows either percentage of maximum value or absolute value compared to a maximum value represented as a solid area. SVG-based. Requires a SVG plug-in.
Dial	Also known as angular gauge, this chart shows either percentage of maximum value or absolute value compared to maximum value represented as a line. SVG-based. Requires a SVG plug-in.
Line	Multiple series based line chart oriented with each line representing all data points in the series. SVG-based. Requires a SVG plug-in.
Pie	Single series based pie chart with each slice representing a data point in the series. SVG-based. Requires a SVG plug-in.

Table 8–4 (Cont.) Available Chart Types

Chart Type	Description
Stacked Bar, Horizontal	Multiple series based bar chart oriented horizontally with each data point being an absolute value in series representing a segment of a single bar. SVG-based. Requires a SVG plug-in.
Stacked Bar, Vertical	Multiple series based bar chart oriented vertically with each data point being an absolute value in series representing a segment of a single bar. SVG-based. Requires a SVG plug-in.
Stacked Percentage Bar, Horizontal	Multiple series based bar chart oriented horizontally with each data point being an percentage of 100% of the series represented by a segment of a single bar. SVG-based. Requires a SVG plug-in.
Stacked Percentage Bar, Vertical	Multiple series based bar chart oriented vertically with each data point being an percentage of 100% of the series represented by a segment of a single bar SVG-based. Requires a SVG plug-in.

Note: Do not change the Type of an existing chart. Instead, delete the existing chart and then re-create it.

Creating a New Chart

How you create a chart depends upon whether you are adding the chart to an existing page, or adding a chart on a new page.

Adding a Chart to an Existing Page

To add a chart to an existing page:

1. Navigate to the Page Definition.
2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.
3. Select **Chart**.
4. Select the type of chart you wish to create. (See Table 8–4 on page 8-26.)
5. Follow the on-screen instructions.

Adding a Chart to a New Page

To create a chart on a new page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. Click **Create Page**.
4. When prompted for the type of page to create, select **Page with Component**.
5. Select the component type **Chart**.
6. Select the type of chart you wish to create. (See Table 8–4 on page 8-26.)

7. Follow the on-screen instructions.

See Also: "Adding Additional Pages" on page 8-3

Editing Chart Attributes

Once you have created a chart you can alter its display by editing chart attributes on the chart attributes page.

To access the Chart Attributes page:

1. Navigate to the Page Definition.
2. Under Regions, click **Chart** next to the name of the chart region you wish to edit.

   The Chart Attributes page appears.

   Note that removing the chart title (that is, the Chart Title attribute) may negatively impact the location and display of the chart legend.

Understanding Chart CSS Classes

When you create a new chart, Oracle HTML DB renders it based on cascading style sheet (CSS) classes associated with the current theme. You can change the appearance of a chart by referencing another CSS or by overriding individual classes in the CSS section of the Edit Attributes page

The following sample contains the CSS classes for the dial chart in *Sample Application*. This example contains all the available CSS classes. Class names appear in boldface.

```
text{font-family:Verdana, Geneva, Arial, Helvetica, sans-serif;fill:#000000;}
tspan{font-family:Verdana, Geneva, Arial, Helvetica, sans-serif;fill:#000000;}
text.title{font-weight:bold;font-size:14;fill:#000000;}
text.moredatafound{font-size:12;}
rect.legend{fill:#EEEEEE;stroke:#000000;stroke-width:1;}
text.legend{font-size:10;}
#background{fill:#FFFFFF;stroke:none;}
rect.chartholderbackground{fill:#ffffff;stroke:#000000;stroke-width:1;}
#timestamp{text-anchor:start;font-size:9;}
text.tic{stroke:none;fill:#000000;font-size:12}
line.tic{stroke:#000000;stroke-width:1px;fill:none;}
#dial{stroke:#336699;stroke-width:2px;fill:#336699;fill-opacity:.5;}
#dial.alert{fill:#FF0000;fill-opacity:.5;}
#dialbackground{stroke:#000000;stroke-width:none;fill:none;filter:url(#MyFilter);}
#dialcenter{stroke:none;fill:#111111;filter:url(#MyFilter);}
#dialbackground-border{stroke:#DDDDDD;stroke-width:2px;fill:none;filter:url
(#MyFilter);}#low{stroke-width:3;stroke:#336699;}
#high{stroke-width:3;stroke:#FF0000;}
#XAxisTitle{letter-spacing:2;kerning:auto;font-size:14;fill:#000000;text-anchor:mi
ddle;}
#YAxisTitle{letter-spacing:2;kerning:auto;font-size:14;fill:#000000;text-anchor:mi
ddle;writing-mode:tb;}
.XAxisValue{font-size:8;fill:#000000;}
.YAxisValue{font-size:8;fill:#000000;text-anchor:end;}
.nodatafound{stroke:#000000;stroke-width:1;font-size:12;}
.AxisLine{stroke:#000000;stroke-width:2;fill:#FFFFFF;}
.GridLine{stroke:#000000;stroke-width:0.3;stroke-dasharray:2,4;fill:none;}
g.dataholder rect{stroke:#000000;stroke-width:0.5;}
.legenditem rect{stroke:#000000;stroke-width:0.5;}
```

Table 8–5 describes all supported CSS class. Note that certain classes only apply to specific chart types.

Table 8–5 Available Chart CSS Classes

Class	Description
`text`	Defines the appearance of text that displays in a chart.
`tspan`	Defines the appearance of text that displays in a chart. `tspan` should match the definition of `text`.
`text.title`	Overrides the default chart text. Use this class for title text.
`text.moredatafound`	Defines the appearance of more datafound text.
`rect.legend`	Creates the rectangular box that holds the chart legend. For a remove the legend border, change `rect.legend` to the following: `rect.legend{fill:#CCCC99;stroke:none;}`
`text.legend`	Defines the text that appears in the chart legend.
`#background`	Creates the entire background for the SVG plug-in. For a solid white background with no border, change `#background` to the following: `#background{fill:#FFFFFF;stroke:#FFFFFF;stroke-width:2;}`
`rect.chartholderbackground`	(Not applicable to pie and dial charts.) Creates the background of the rectangle that holds the chart data. For a clear background, change `rect.chartholderbackground` to the following: `rect.chartholderbackground(display:none;)`
`#timestamp`	Only applicable if the Asynchronous Update chart attribute is set to Yes. Controls the appearance of the update timestamp test. To disable the display of the timestamp, use defines `#timestamp` as follows in the Custom CSS, Inline attribute. `"#timestamp{display:none;}"` **See Also:** "Enabling Asynchronous Updates" on page 8-31
`text.tic`	(Dial charts only.) Defines the numbers on a dial chart.
`line.tic`	(Dial charts only.) Defines the graduation mark that displays directly beneath the number on a dial chart.
`#dial`	(Dial charts only.) Defines the value that displays on the dial chart.
`#dial.alert`	(Dial charts only.) Defines a value (called an alert value) that renders on in a dial chart using a different display.
`#dialbackground`	(Dial charts only.) Creates the background of a dial chart.
`#dialcenter`	(Dial charts only.) Creates the center of the dial on a dial chart.
`#dialbackground-border`	(Dial charts only.) Works in conjunction with `#dialbackground` to create specific graphic effect.
`#low`	(Dial charts only.) Defines a historical low watermark of the data being displayed on a chart.

Table 8–5 (Cont.) Available Chart CSS Classes

Class	Description
`#high`	(Dial charts only.) Defines historical high watermark of the data being displayed on a chart.
`#XAxisTitle`	Defines the title that appears on the X axis
`#YAxisTitle`	Defines the title that appears on the Y axis.
`.XAxisValue`	Defines the value that appears on the X axis.
`.YAxisValue`	Defines the value that appears on the Y axis.
`.AxisLabel`	Similar to axis value.
`.nodatafound`	Defines the text element that displays if no information is available.
`.AxisLine`	Indicates zero on charts that have negative values.
`.GridLine`	Creates the horizontal and vertical lines on the chart.
`g.dataholder rect`	Applies a blanket style to all data that displays in the chart.
`.legenditem rect`	Applies a blanket style to all rectangular items in the legend.

Referencing a Custom Cascading Style Sheet

You can reference a custom cascading style sheet for a chart using the CSS section of the Chart Attributes page. When you reference an external CSS, you can reference it entirely or simply override specific styles.

To reference a custom chart CSS:

1. Upload the CSS to Oracle HTML DB. (See "Uploading Cascading Style Sheets" on page 9-40).
2. Create a chart. (See "Creating a New Chart" on page 8-27.)
3. Navigate to the Page Definition.
4. Under Regions, click **Chart** next to the region name.

   The Chart Attributes page appears.
5. Scroll down to the CSS section.
6. From Use Custom CSS, select **Yes**.
7. To reference an external CSS exclusively:
   a. In Custom CSS, Link enter a link to a custom CSS. For example:

      ```
      #IMAGE_PREFIX#themes/theme_4/svg.css
      ```

   b. Specify that the CSS should be used exclusively. In Custom CSS, Inline enter the following:

      ```
      /**/
      ```

8. To reference a custom CSS and override specific styles:
   a. In Custom CSS, Link enter a link to a custom style sheet. For example:

      ```
      #IMAGE_PREFIX#themes/theme_4/svg.css
      ```

b. In Custom CSS, Inline, enter the custom CSS styles you want to override.

Specifying Custom CSS Styles Inline

You can override specific styles within the default CSS, using the Custom CSS, Inline attribute on the Chart Attributes page.

To override specific styles within the default CSS:

1. Create a chart. (See "Creating a New Chart" on page 8-27.)
2. Navigate to the Page Definition.
3. Under Regions, click **Chart** next to the region name.

   The Chart Attributes page appears.
4. Scroll down to CSS.
5. From Use Custom CSS, select **Yes**.
6. In Custom CSS, Inline, enter the custom CSS styles you want to override.

Enabling Asynchronous Updates

You can create charts that monitor information by enabling the Asynchronous Update attribute on the Chart attributes page. Enabling this attribute updates the chart to reflect changes in the underlying data within a specified time interval.

To enable asynchronous updates:

1. Create a chart. (See Creating a New Chart on page 8-27.)
2. Navigate to the Page Definition.
3. Under Regions, click **Chart** next to the region name.

   The Chart Attributes page appears.
4. Scroll down to Refresh.
5. From Asynchronous Update, select **Yes**.
6. In Update Interval (Seconds), enter the interval in seconds between chart updates. For optimal performance, select an interval that is greater than 2 seconds.

When Asynchronous Update is enabled, a timestamp displays on the chart indicating that last update.

To disable the Asynchronous Update timestamp:

1. Navigate to the Chart Attributes page.
2. Locate the CSS section.
3. From Use Custom CSS, select **Yes**.
4. In Custom CSS, Inline edit #timestamp as follows:

```
#timestamp{display:none;}
```

Displaying Charts in Other Languages

To display a chart in another language, you edit the `text` and `tspan` classes to reflect the correct language.

To display a chart in another language:

1. Navigate to the Chart Attributes page. (See Editing Chart Attributes on page 8-28.)
2. Scroll down to CSS.
3. From Use Custom CSS, select **Yes**.
4. In Custom CSS, Inline, edit the `text` and `tspan` classes to reflect the correct language. The following example demonstrates how to change a chart to Korean:

```
text{font-family:Batang;fill:#000000;}
tspan{font-family:Batang;fill:#000000;}
```

Creating Buttons

As you design your application you can use buttons to direct users to a specific page or URL, or to post or process information (for example, by creating Create, Cancel, Next, Previous, or Delete buttons).

Buttons can perform two different types of actions. A button can submit the page and then redirect to a URL. Alternately, a button can simply branch to a URL without submitting the page.

Topics in this section include:

- Creating a Button Using a Wizard
- Creating Multiple Buttons
- Understanding the Relationship Between Button Names and REQUEST
- About Branching with Buttons
- Displaying Buttons Conditionally

Creating a Button Using a Wizard

You create a button by running the Create Button Wizard from the Page Definition. Each button resides in a region. A region is a area on a page that serves as a container for content.

To create a new button:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. If necessary, create an HTML region. (See "Customizing Regions" on page 9-3.)
3. Under Buttons, click the **Create** icon.

   The Create Button Wizard appears.
4. Select a region to contain the button.
5. Select a position for the button:
   - **Create a button displayed among this region's items**
   - **Create a button in a region position**

   Select **Create a button displayed among this region's items** to add a button to a region as if it was an item (for example, to add a button directly to the right of a form field).
6. If you select **Create a button in a region position**:
   a. Specify the Button Name and Label.

b. Select a Button Type: **HTML Button (Default)**, **Image**, or **Template Driven**

   Select **Button is Reset** to create an Undo button. When enabled, this type of button resets the page values to the state they were in when the page was initially rendered.

c. Select an Action.

   Selecting **Submit page and redirect to URL** submits the current page to the HTML DB engine whenever a user clicks the button.

   Selecting **Redirect to URL without submitting page** avoids submitting the page. Choose this action when submitting the page for processing is not necessary (for example, a Cancel button). This action avoids processing in the database and therefore reduces the load.

7. If you select **Create a button displayed among this region's items**:

   a. Specify the Button Name and Sequence.

   b. Specify whether the button displays at the beginning of a new line or new field.

   c. Specify a Label.

   d. Enter the value of Request.

   e. Select the Button Style.

8. Follow the on-screen instructions

See Also: "Understanding the Relationship Between Button Names and REQUEST" on page 8-34

Creating an HTML Button

Buttons can be placed in predefined region template position or among items in a form. To create an HTML button, select one of the following while running the Create Button Wizard:

- Under Task, select **Create a button in a region position**
- Under Button Type, select a button type and then **HTML Button (default)**

Creating Multiple Buttons

You can create multiple buttons within the same region at once using the Create Multiple Buttons Wizard.

To create multiple buttons at once:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. If necessary, create an HTML region. (See "Customizing Regions" on page 9-3.)
3. Under Buttons, click the **Create** icon.

   The Create Button Wizard appears.

4. From the Tasks list, select **Create Multiple Buttons**.

   The Create Multiple Button Wizard appears.

5. From Place Buttons in Region, select the region to contain the buttons.
6. From Template, select a template.

7. In HTML Attributes, Specify HTML attributes for these buttons. This text will be added to the HTML element definition. For example, you could set the class of a text button as follows:

   ```
   class="myclass"
   ```

8. To quickly populate the remaining fields, select make a selection from the Quick Button list on the right side of the page.
9. Click **Create Buttons**.

Understanding the Relationship Between Button Names and REQUEST

The name you give a button determines the value of the built-in attribute `REQUEST`. You can reference the value of `REQUEST` from within PL/SQL using the bind variable `:REQUEST`. By using this bind variable, you can conditionally process, validate, or branch based on which button the user clicks. You can also create processes that execute when the user clicks a button. You can also use a more complex condition as demonstrated in the following examples:

```
If :REQUEST in ('EDIT','DELETE') then ...
If :REQUEST != 'DELETE' then ...
```

These examples assume the existence of buttons named EDIT and DELETE. You can also use this syntax in PL/SQL Expression conditions. Be aware, however, that the button name case is preserved. In other words, if you name a button "LOGIN" then a request looking for the name "Login" will fail. For example:

```
<input type="BUTTON" value="Finish" onClick="javascript:doSubmit('Finish');">
```

Note that in this example *`Finish`* is the name of the `REQUEST` and this example is case-sensitive.

About Branching with Buttons

Each page can include any number of branches. A branch links to another page in your application or to a URL. The HTML DB engine considers branching at different times during page processing. You can choose to branch before processing, before computation, before validation, and after processing. Like any other control in Application Builder, branching can be conditional. For example, you can branch when a user clicks a button. When you create a branch, you associate it with a specific button. The branch will only be considered if a user clicks the button.

See Also: "Controlling Flow Using Branches" on page 10-6

Displaying Buttons Conditionally

You can choose to have a button display conditionally by editing attributes on the Edit Pages Button page.

To have a button display conditionally:

1. Create the button. (See "Creating a Button Using a Wizard" on page 8-32.)
2. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
3. Under Buttons, select the button name.

   The attributes page for the button appears.

4. Scroll down to Conditional Button Display.
5. Make a selection from the Condition Type list.
6. Enter an expression in the fields provided.
7. Click **Apply Changes**.

See Also: "About Bind Variables" on page 6-13

Creating Items

An item is part of an HTML form. An item can be a text field, text area, password, combobox, check box, and so on. Item attributes affect the display of items on a page. For example, these attributes can impact where a label displays, how large an item will be, and whether the item will display next to or below the previous item.

There are two types of items, page items and application items. Page items are placed on a page and have associated user interface properties, such as Display As, Label and Label Template. Application items are not associated with a page and therefore have no user interface properties. You can use an application item as a global variable.

Topics in this section include:

- Creating a Page Level Item
- Referencing Item Values
- Displaying Conditional or Read-only Items
- Creating an Application Level Item
- Populating an Alternative Date Picker Format

See Also: "How Item Attributes Affect Page Layout" on page 9-2 and "Using Substitution Strings" on page 6-16

Creating a Page Level Item

You create a page level item by running the Create Item Wizard from the Page Definition.

To create a new page level item:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. If necessary, create an HTML region. (See "Customizing Regions" on page 9-3.)
3. Under Items, click the **Create** icon.
4. Select an item type. (See "About Item Types" on page 8-36.)
5. Follow the on-screen instructions

About Item Naming Conventions

When specifying an item name, remember the following rules. Item names must:

- Not use quotes
- Begin with a letter or a numeral, and subsequent characters can be letters, numerals, or underscore characters
- Be case insensitive

- Should not exceed 30 characters
- Cannot contain letters outside the base ASCII character set

About Item Types

When you create an item, you specify an item type. Once you create an item, these types appear on the Display As list on the Edit Page Item page. Table 8–6 describes available item types.

Table 8–6 Available Item Types

Item Type	Description
Button	Used to build forms in Oracle HTML DB. Use an Item Button when you want to place a button among other fields (or items) in a form. When clicked, this type of button will automatically switch the HTML DB engine to processing mode, enabling you to perform validations, execute processes, or branch the user to another page.
Display Only	Oracle HTML DB uses HTML tables to render items. Use this item to control the layout of items in forms by closing a table and starting a new one.

Check box	Displayed using a list of values. A list of values is required for items displayed as check boxes. The value corresponding to a checked box is returned in a single colon (:) delimited string.
	The following example demonstrates how to create a single check box that returns `YES`. This example would display both a check box and a field label.
	`SELECT NULL display_text, 'YES' return_value FROM DUAL;`
	This example includes the additional text "Click to select."
	`SELECT 'Click to select' display_text, 'YES' return_value FROM DUAL;`
	See Also: "HTMLDB_UTIL" on page 17-1 for information on breaking up returned values

Date Picker	Displays a text field with a Calendar icon next to it. When clicked, this icon displays a small calendar from which the user can select a date and a time (optional). If the format you need is not included in the Display As list, select **Date Picker (use application format mask)**. When using a format mask, your application looks for the format in an item called `PICK_DATE_FORMAT_MASK`. Note that you need to populate this item before this item type will work. **See Also:** "Populating an Alternative Date Picker Format" on page 8-40
File Browse	Displays a text field with a "Browse..." button. This enables the user to locate a file on a local file system and upload it. Oracle HTML DB provides a table for these files to be uploaded to as well as an API to retrieve the files.
Hidden	Renders an HTML hidden form element. Session state can be assigned and referenced just like a text field.
List Managers	Based on a list of values. This item enables you to manage a list of items by selecting and adding to a list. The list of values display as a popup.
Multiple Select	Renders as a multiselect HTML form element. When submitted, selected values are returned in a single colon delimited string. You can break up the values using the `HTMLDB_UTIL API`. **See Also:** "HTMLDB_UTIL" on page 17-1
Password	Renders as an HTML password form element.

Table 8–6 (Cont.) Available Item Types

Item Type	Description

Popup List of Values	Renders as a text field with an icon next to it from which a user can select a value from a popup window. The list in the popup window is driven by a list of values. There are two types of Popup LOVs, one that fetches a set of rows when the window pops up and one that does not.
	Popup LOV values must select two columns. For example:
	`SELECT ename, empno FROM emp`
	If one of the columns is an expression, remember to use an alias. For example:
	`SELECT ename||' '||job display_value, empno FROM emp`

Radio	Renders as an HTML radio group form element, based on a list of values. Choose **Radiogroup with Submit** to have the page submitted when the radio button is selected. The following example displays employee names (`ename`), but returns employee numbers (`empno`): `SELECT ename, empno FROM emp`
Select List	Displays using a list of values. A list of values is required for items displayed as a select list. Select lists are rendered using the HTML form element `<select>`. The values in a select list are determined using a named list of values or a list of values defined at the item level. You may specify the NULL display value and NULL return value. The following example would return employee names (`ename`) and employee numbers (`empno`) from the `emp` table. Note that column aliases are not required and are included in this example for clarity. `SELECT ename display_text, empno return_value FROM emp` Oracle HTML DB provides additional enhancements to a standard HTML select list: ■ **Select List with Submit** - Submits the page when the user changes its selected value. Upon submit, the REQUEST will be set to the name of the item that represents the select list, allowing you to execute conditional computations, validations, processes, and branches. ■ **Select List with Redirect** - Redirects the user back to the same page, setting ONLY the newly selected value of the select list in session state. ■ **Select List Returning URL redirect** - Based on a list of values with URLs as the return values. Changing the value of the select list causes the browser to redirect to the corresponding URL. ■ **Select List with Branch to Page** - Based on list of values with page numbers as return values. Changing the selected value in the select list causes the HTML DB engine to branch to the corresponding page.

Table 8–6 (Cont.) Available Item Types

Item Type	Description
Text	Displays as an HTML text field containing a maximum of 30,000 bytes of text. You control the maximum length and display width by editing the Height and Width item attribute.
Text Area	Renders as an HTML text area. This is no maximum length for an item displayed as a text area. You control the height and width by editing the Height and Width item attribute. Additional available Text Area Display As options include: ■ **Text Area (auto height)** - Varies the height based on the amount of text. Use this option to have a large text area when you have a lot of data and a smaller text area when you have little or no data. ■ **Text Area with Counter** - Includes a counter that displays the number of bytes entered in the field. ■ **Text Area with Spell Checker** - Provides a popup English language spell checker. ■ **Text Area with HTML Editor** - Provides basic text formatting controls. Note that these controls may not work in all Web browsers.
Text	Available Text Control display options include: ■ **Text Field** - Renders as a text field. ■ **Text Field (Disabled, does not save state)** - Displays a read-only version of a display value from a list of values by using the item's value in session state to look up the corresponding display value in the associated list of values. The value displayed on the screen is not saved in session state upon submit. ■ **Text Field (Disabled, saves state)** - Displays a read-only version of a display value from a list of values by using the item's value in session state to look up the corresponding display value in the associated list of values. ■ **Text Field (always submits page when Enter pressed)** - Displays a read-only version of the value in session state. Upon submit, the value displayed is saved in session state. ■ **Text Field with Calculator Popup** - Renders as a text field with an icon next to. When clicked, the icon displays a small window containing a calculator. Calculations are placed back in the text field.

Referencing Item Values

You can reference item values stored in session state in regions, computations, processes, validation, and branches. Table 8–7 describes the supported syntax for referencing item values.

See Also: "Managing Session State Values" on page 6-10

Table 8–7 Syntax for Referencing Item Values

Type	Syntax	Description
SQL	`:MY_ITEM`	Standard bind variable syntax for items no longer than 30 bytes. Use this syntax for references within a SQL query and within PL/SQL.
PL/SQL	`V('MY_ITEM')`	PL/SQL syntax referencing the item value using the `V` function. **See Also:** "Oracle HTML DB APIs" on page 17-1
PL/SQL	`NV('MY_NUMERIC_ITEM')`	Standard PL/SQL syntax referencing the numeric item value using the `NV` function. **See Also:** "Oracle HTML DB APIs" on page 17-1
Static Text	`&MY_ITEM`	Static text.
Static Text (exact)	`&MY_ITEM.`	Static text. Exact Substitution.

You can set the value of an item in your application using any of the following methods:

- For page items, use the Source Attribute to set the item value.

  From the Page Definition, select the item name to view the Edit Page Item page. Scroll down to Source and edit the appropriate fields.

  You can also set the value of an item in any region based on PL/SQL or a process using the following syntax:

  ```
  BEGIN
   :MY_ITEM :=  'new value';
  END;
  ```

- Pass the value on a URL reference using `f?p` syntax. For example:

  ```
  f?p=100:101:10636547268728380919::NO::MY_ITEM:ABC
  ```

- Set the value using a computation. Computations are designed to set item values. For example:

  ```
  TO_CHAR(SYSDATE,'Day DD Month, YYYY');
  ```

- Use the PL/SQL API to set an item value within a PL/SQL context. For example:

  ```
  HTMLDB_UTIL.SET_SESSION_STATE('MY_ITEM',SYSDATE);
  ```

See Also:

- "Clearing Session State" on page 6-11
- "Oracle HTML DB APIs" on page 17-1

Displaying Conditional or Read-only Items

You can choose to have an item display conditionally or as read-only by editing attributes on the Edit Pages Item page.

To display a conditional or read-only item:

1. Create the item. (See "Creating a Page Level Item" on page 8-35.)

2. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
3. Under Items, select the item name.

   The attributes page for the item appears.
4. To display an item conditionally:
   a. Scroll down to Conditional Display.
   b. Make a selection from the Condition Type list.
   c. Enter an expression in the fields provided.
5. To make an item read-only:
   a. Scroll down to Read Only Display Settings.
   b. Make a selection from the Read Only Condition Type list.
   c. Enter an expression in the fields provided.
6. Click **Apply Changes**.

Creating an Application Level Item

Application level items do not display, but are used as global variables to the application. To reference an application level item you must set a value by using an `ON_NEW_INSTANCE` computation.

To create a new application level item:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application name.
3. When Application Builder appears, click **Shared Components**.
4. Under Logic, select **Items**.

   The Application Items page appears.
5. To create a new application item, click the **Create** icon.
6. Follow the on-screen instructions.

Accessing Application Item History

You can view history of changes to application items by clicking **History** at the top of the Application Items page.

Populating an Alternative Date Picker Format

If you need to create a Date Picker item, but the format you need does not appear in the Display As list, you should select **Date Picker (use application format mask)**. When an application uses this type of date picker, the HTML DB engine derives the date format from an item named `PICK_DATE_FORMAT_MASK`. You can populate this item in two ways:

- By defining an application substitution string named `PICK_DATE_FORMAT_MASK`
- By creating an application level item named `PICK_DATE_FORMAT_MASK`

Defining PICK_DATE_FORMAT_MASK as an Application Substitution String

One approach to populating PICK_DATE_FORMAT_MASK is to create an application substitution string. You define application level substitution strings on the Edit Application Attributes page. Remember that an application level substitution string is a static value and cannot be altered at runtime.

To define a new application substitution string named PICK_DATE_FORMAT_MASK:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears.
3. Click the **Edit Attributes** icon.
4. Scroll down to Static Substitution Strings.
5. Create a new static substitution string named PICK_DATE_FORMAT_MASK:
   a. In Substitution String, enter the name PICK_DATE_FORMAT_MASK.
   b. In Substitution Value, enter a value for your for your date format (for example, Month DD, YYYY).

Defining an Application Level Item Named PICK_DATE_FORMAT_MASK

Another approach to populating PICK_DATE_FORMAT_MASK is to create an application level item named PICK_DATE_FORMAT_MASK. This approach enables you to control any items rendered as **Date Picker (use application format mask)** by simply setting the value of this item. Plus, you can set the value of PICK_DATE_FORMAT_MASK using a computation from anywhere within your application.

If you wish to provide the user with a list of date formats as preferences, you will need to create an application level item named PICK_DATE_FORMAT_MASK and then use a computation to set the value of this item based upon the user's selection.

See Also: "Creating an Application Level Item" on page 8-40

Creating Lists of Values

A list of values (LOV) is a static or dynamic definition used to display a specific type of page item, such as popup lists of values, a select list, a check box, a radio group, or multiple select lists.

Topics in this section include:

Creating Named LOVs at the Application Level

You define named (or shared) LOVs at the application level by running the Create LOV Wizard and adding them to the Named List of Values repository. All LOVs can be defined as static or dynamic. Static lists are based on predefined pairs of display and

return values. Dynamic lists are based on a SQL query you write that selects values from a table.

To create a named LOV:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application name.
3. When Application Builder appears, click **Shared Components**.
4. Under User Interface, select **Lists of Values**.
5. To create a new LOV, click the **Create** icon.
6. Follow the on-screen instructions.

   New named LOVs are added to the Named List of Values repository.

About Static LOVs

Static LOVs are based on a static list of display and return values you specify when you run the Create LOV Wizard. To create a static LOV you run the Create LOV Wizard and select the LOV type Static. Oracle HTML DB stores the display values, return values, and sort sequence you specify in the Named List of Values repository. Once you add a static LOV to the repository you can create an item and display it as a check box, radio group, select list, or popup list based on this definition.

About Popup LOVs

Using a popup LOV is a good choice for lists of values that are too large to return on a single page. Popup LOVs create an icon to the right of a standard text field. When the user clicks this icon, a popup window appears with a list of values represented as a series of links. When the user selects from this searchable list, the selected value will be placed in the text field. You control popup LOVs through templates. You can only specify one popup LOV template for each application.

Popup LOVs must be based on a query that selects two columns with different column aliases. For example:

```
SELECT ename name, empno id
   FROM emp
```

Referencing Session State within a LOV

You can reference session state by using bind variables. In the following example, this LOV only works if the item called *my_deptno* contains a valid department number.

```
SELECT ename, empno FROM emp WHERE deptno = :my_deptno
```

Accessing LOV Reports

Application Builder includes a number of reports designed to help you better manage LOVs, including:

- Bulk Update LOV Data Values
- Subscribed LOVs
- LOV Utilization

- LOV Change History

To access LOV reports:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application name.
3. When Application Builder appears, click **Shared Components**.
4. Under User Interface, select **Lists of Values**.
5. Select one of the following buttons at the top of the page:
   - **Edit all lists of values entries**
   - **Subscription**
   - **Utilization**
   - **History**
6. Follow the on-screen instructions.

Bulk Update LOV Data Values

Click **Edit all lists of values entries** to display the Bulk Update LOV Data Values page. Use this page to edit the display values of all static LOVs at once. To save your changes, click **Apply Changes**. To search for a specific name, enter keywords at the top of the page and click **Go**.

Subscribed LOVs

Click **Subscription** to display the Subscribed LOVs page. This page displays all subscribed LOVs in your application.

LOV Utilization

Click **Utilization** to display the LOV Utilization page. This page displays LOVs used in the current application. To edit an LOV, click the LOV name.

LOV Change History

Click **History** to display the LOV Change History page. This page displays a history of recently changed LOVs by date.

Creating Calendars

Oracle HTML DB includes a built-in wizard for generating a calendar. Once you specify the table on which the calendar is based you can create drill down links to information stored in specific columns.

Topics in this section include:

- About Creating Calendars
- Creating a New Calendar
- Converting an Easy Calendar to a SQL Calendar
- Editing a Calendar Title
- Editing Calendar Attributes

About Creating Calendars

Oracle HTML DB supports two calendar types:

- **Easy Calendar** creates a calendar based on schema, table, and columns you specify. The wizard prompts you to select a date column and display column.
- **SQL Calendar** creates a calendar based on a SQL query you provide. The SQL SELECT statement you provide must include at least two columns, a date column and display column.

The date column determines which days on the calendar will contain entries. The display column defines a specific row that will display a calendar date.

See Also: "Calendar Display" on page 8-46

Supported Calendar Substitution Strings

Oracle HTML DB supports a number of date format substitution strings. You can view a complete list of supported substitution strings on the Calendar Templates page.

To view a list of supported substitution strings for calendars:

1. Navigate to the appropriate calendar template.
2. Expand the Substitution Stings list on the right side of the page.

See Also: See "Viewing Templates" on page 9-17

Creating a New Calendar

How you create a calendar depends upon whether you are adding a calendar to an existing page, or adding a calendar on a new page.

Adding a Calendar to an Existing Page

To add a calendar to an existing page:

1. Navigate to the Page Definition.
2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.
3. Select **Calendar**.
4. Select the type of calendar you wish to create:
   - **Easy Calendar** creates a calendar based on the date column and display column you specify.
   - **SQL Calendar** creates a calendar based on a SQL query you provide.
5. Follow the on-screen instructions.

Adding a Calendar to a New Page

To create a calendar on a new page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. Click **Create Page**.
4. When prompted for the type of page to create, select **Page with Component**.

5. Select the component type **Calendar**.
6. Select the type of calendar you wish to create:
   - **Easy Calendar** creates a calendar based on the date column and display column you specify.
   - **SQL Calendar** creates a calendar based on a SQL query you provide.
7. Follow the on-screen instructions.

See Also: "Adding Additional Pages" on page 8-3

Converting an Easy Calendar to a SQL Calendar

Creating an Easy Calendar is the simplest way to create a calendar in Oracle HTML DB. However, if you find the resulting calendar does not meet your needs, you can quickly convert it to SQL Calendar.

To convert an Easy Calendar to a SQL Calendar:

1. Navigate to the Page Definition.
2. Under Regions, click **CAL** next to the region name.

   The Calendar Attributes page appears.
3. From the Tasks list, select **Convert to SQL Based calendar**.

   Converting an Easy Calendar to a SQL Calendar, adds a Region Source section to the Region Definition. The Region Source contains the original SQL query that creates the calendar. By having accessing the Region Source, you can easily edit the query to meet your needs.

Editing a Calendar Title

The title that appears at the top of calendar corresponds to the region title.

To alter the region title:

1. Navigate to the Page Definition.
2. Under Regions, select the region name.

   The Region Definition appears.
3. Under Region, enter a new title.
4. Click **Apply Changes**.

Editing Calendar Attributes

Once you have created a calendar you can alter how it appears by editing its attributes.

Topics in this section include:

- Accessing the Calendar Attributes Page
- Calendar Display
- Calendar Interval
- Column Link
- Day Link

Accessing the Calendar Attributes Page

To access the Calendar Attributes page:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Regions, click **CAL** next to the region name.

   The Calendar Attributes page appears. The topics that follow describe the specific sections of the Calendar Attributes page.

Calendar Display

Use Calendar Display to specify a calendar template, date columns, and general calendar formatting.

Calendar Template determines what template is used when the HTML DB engine renders a calendar. **Date Column** defines the column from the table or query containing the dates to be placed on the calendar. **Display Column** defines a specific row that displays on a calendar date.

To select another Display Column:

1. Navigate to the appropriate Calendar Attributes page.
2. Locate the Calendar Display section.
3. To specify another display column, make a selection from the Display Column list.
4. Click **Apply Changes**.

To specify a custom Display Column:

1. Navigate to the appropriate Calendar Attributes page.
2. Locate the Calendar Display section.
3. From Display Type, select **Custom**.
4. In Column Format enter a custom column format. You can use an HTML expression and supported substitution strings.
5. Click **Apply Changes**.

> **See Also:** "Supported Calendar Substitution Strings" on page 8-44

Calendar Interval

Use Calendar Interval to define the dates which are included in the calendar.

Begin At Start Of Interval determines when the calendar should start. Selecting this option creates a calendar that spans an entire interval (for example, a month). For example:

- If **Begin at start of interval** is selected, the date is June 15th, and the display is monthly, the resulting calendar spans from June 1st to June 30th.
- If **Begin at start of interval** is not selected, the date is June 15th, and the display is monthly, the resulting calendar spans from June 15th to June 30th.

Date Item holds the date on which the calendar is based.

The next two attributes define which items hold the calendar start date and end date. You can use these attributes to create calendars that span multiple months at a time. **Item Containing Start Date** points to an item that holds the start date of the calendar.

Item Containing End Date points to an item that holds the end date of calendar. Note that format of the date of either item must be `YYYMMDD`.

Start of Week determines the day of the week on which the calendar starts.

Column Link

Use Column link to create a link on the column in the calendar.

To create a column link to another page:

1. Navigate to the appropriate Calendar Attributes page.
2. Scroll down to Column Link.
3. From Target is a, select **Page in this Application**.
4. In Page, specify the target page number. To reset pagination for this page, select **reset pagination for this page**.
5. In Request, specify the request to be used.
6. In Clear Cache, specify the pages (that is, the page numbers) on which to clear cache. Specify multiple pages by listing the page numbers in a comma delimited list.

   You can set session state (that is, give a listed item a value) using the next two attributes:
7. To set session state:
   a. Set these items - Enter a comma delimited list of item names for which you would like to set session state.
   b. With these values - Enter a comma delimited list of values for the items specified in the previous step.

      You can specify static values or substitution syntax (for example, `&APP_ITEM_NAME.`). Note that item values passed to `f?p=` in the URL may not contain a colon (:). Additionally, item values may not contain commas unless you enclose the entire value in backslash characters (for example, `\1234,56\`).
8. Click **Apply Changes**.

See Also: "Supported Calendar Substitution Strings" on page 8-44

To create a column link to a URL:

1. Navigate to the appropriate Calendar Attributes page.
2. Scroll down to Column Link.
3. From Target is a, select **URL**.
4. In URL, enter the appropriate address.
5. Click **Apply Changes**.

Day Link

Use Day link to create link on a day in the calendar. This attribute creates a link on an actual numeral (or day) on the calendar.

To create a day link to another page:

1. Navigate to the appropriate Calendar Attributes page.

2. Scroll down to Day Link.
3. From Target is a, select **Page in this Application**.
4. In Page, specify the target page number.

   To reset pagination for this page, select **reset pagination for this page**.
5. In Request, specify the request to be used.
6. In Clear Cache, specify the pages (that is, the page numbers) on which to clear cache. Specify multiple pages by listing the page numbers in a comma delimited list.

   You can set session state (that is, give a listed item a value) using the next two attributes:
7. To set session state:
   a. Set these items - Enter a comma delimited list of item names for which you would like to set session state.
   b. With these values - Enter a comma delimited list of values for the items specified in the previous step.

      You can specify static values or substitution syntax (for example, `&APP_ITEM_NAME.`). Note that item values passed to `f?p=` in the URL may not contain a colon (:). Additionally, item values may not contain commas unless you enclose the entire value in backslash characters (for example, `\1234,56\`).
8. Click **Apply Changes**.

To create a day link to a URL:

1. Navigate to the appropriate Calendar Attributes page.
2. Scroll down to Day Link.
3. From Target is a, select **URL**.
4. In URL, enter the appropriate address.
5. Click **Apply Changes**.

Utilizing Shortcuts

By using shortcuts you can avoid repetitive coding of HTML or PL/SQL functions. You can use a shortcut to define a page control such as a button, HTML text, a PL/SQL procedure, or HTML. Once defined, you can invoke a shortcut using specific syntax unique to the location in which the shortcut is used. Shortcuts can be referenced many times, thus reducing code redundancy.

This section contains the following topics:

About Shortcut Types

When you create a new shortcut, you must specify the type of shortcut you wish to create. Oracle HTML DB supports the following shortcut types:

- PL/SQL Function Body

- HTML Text
- HTML Text with Escaped Special Characters
- Image
- Text with JavaScript Escaped Single Quotes
- Message
- Message with JavaScript Escaped Special Quotes

Text with JavaScript Escaped Single Quotes

Use this type of shortcut to reference a shortcut inside of a JavaScript literal string. This shortcut defines a text string. When the shortcut is referenced, it escapes the single quotes required for JavaScript.

Message

Use this type of shortcut to reference a translatable message at runtime. Note that since this shortcut does not have a shortcut body, the name of the shortcut must match the corresponding message name. At runtime, the name of the shortcut expands to the text of the translatable message for the current language.

Message with JavaScript Escaped Single Quotes

Use this type of shortcut to reference a shortcut inside of JavaScript literal string and reference a translatable message at runtime.

See Also: "About Translating an Application and Globalization Support" on page 16-1

Defining Shortcuts

Before you can incorporate a shortcut in your application, you must define it and add it to the Shortcuts repository. You reference new shortcuts using the following syntax:

```
"MY_SHORTCUT"
```

Note that the shortcut name must be capitalized and enclosed in quotes.

To define a new shortcut:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application name.
3. When Application Builder appears, click **Shared Components**.
4. Under User Interface, select **Shortcuts**.
5. Click **Create**.
6. Select one of the following creation methods:
   - From Scratch
   - As a Copy of an Existing Shortcut
7. Follow the on-screen instructions.

New shortcuts are added to the Shortcut repository and are available for use within the following locations:

- The Region Source attribute of regions defined as HTML Text (with shortcuts). (See "Customizing Regions" on page 9-3.)
- Region Header and Footer Text attribute (See "Specifying a Region Header and Footer" on page 9-6.)
- Item Label attributes and Item Default Value attribute. (See "Items" on page 7-21.)
- Region Templates attributes. (See "Editing Templates" on page 9-19.)

Accessing Shortcut Reports

Application Builder includes a number of reports designed to help you better manage LOVs, including:

- Subscribed Shortcuts
- Shortcut History

To access of report detailing recent shortcut changes:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application name.
3. When Application Builder appears, click **Shared Components**.
4. Under User Interface, select **Shortcuts**.
5. Click **Create**.
6. Follow the on-screen instructions.

Subscribed Shortcuts

Click **Subscription** to display the Subscribed Shortcuts page. This page displays all subscribed shortcuts in your application.

Shortcut History

Click **History** to display the Shortcut History page. This page displays a history of recently changed shortcuts by date.

Incorporating JavaScript into an Application

Adding JavaScript to a Web applications is a great way to add features that mimic those found client/server applications without sacrificing all of the benefits of Web deployment. Oracle HTML DB includes multiple built-in interfaces (or "hooks") especially designed for adding JavaScript.

Remember that JavaScript is not appropriate for data intensive validations. For example, to verify that a name is contained within a large database table, you would need to pull down every record to the client, creating a huge HTML document. In general, complex operations are much better suited for server-side HTML DB validations instead of JavaScript.

This section contains the following topics:

- About Referencing Items Using JavaScript
- How to Incorporate JavaScript Functions
- Calling JavaScript from a Button

See Also: "Understanding Validations" on page 7-21

About Referencing Items Using JavaScript

When you reference an item, the best approach is to reference by ID. If you view the HTML source of an Oracle HTML DB page in a Web browser, you would notice that all items have an ID attribute. This ID corresponds to name of the item, not the item label. For example, if you create an item with the name `P1_FIRST_NAME` and a label of `First Name`, the ID will be P1_FIRST_NAME.

Knowing the item ID enables you to use the JavaScript function `getElementById` to get and set item attributes and values. The following example demonstrates how to reference an item by ID and display its value in an alert box.

```
<script language="JavaScript1.1" type="text/javascript">
  function firstName(){
    alert('First Name is ' + document.getElementById('P1_FIRST_NAME').value );
  }
 // or a more generic version would be
  function displayValue(id){
    alert('The Value is ' + document.getElementById(id).value );
  }
</script>

  // Then add the following to the "Form Element Attributes" Attribute of the
item:
  onChange="javascript:displayValue('P1_FIRST_NAME');"
```

While it is possible to reference items using the HTML form element attribute name (represented in the previous example as form `p_t01`), this is strongly discouraged. Remember that the name attribute is dynamic and is determined at when a page is rendered. For example, suppose the first element rendered on a page is `p_t01` and the second is `p_t02`. If you write a function that references `p_t01` and then create a new item on the page that appears first, the item that was `p_t01` will now be `p_t02` and your function will break.

How to Incorporate JavaScript Functions

There are two primary places to include JavaScript functions:

- In the HTML Header attribute of the page
- In a .js file in the page template.

See Also: "Text with JavaScript Escaped Single Quotes" on page 8-49 for information on referencing a shortcut inside of a JavaScript literal string

Incorporating JavaScript in the HTML Header Attribute

One way to include JavaScript into your application is add it to the HTML Header attribute of the page. This is good approach for functions that are very specific to a page as well as a convenient way to test a function before you include it in the `.js` file.

You can add JavaScript functions to a page by simply entering the code into the HTML Header attribute of the Page Attributes page. For example, adding following would `test` function accessible from anywhere on the current page.

```
<script language="JavaScript1.1" type="text/javascript>
  function test(){
```

```
    alert('This is a test.');
  }
</script>
```

See Also: "HTML Header" on page 7-18

Including JavaScript in a .js File in the Page Template

In a `.js` file in the page template. This is the most efficient approach since a `.js` file loads on the first page view of your application and is then cached by the browser.

The following demonstrates how to include a .js file in the header section of a page template.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>#TITLE#</title>
    #HEAD#
    <script src="#APP_IMAGES#custom.js" type="text/javascript"></script>
</head>
<body #ONLOAD#>#FORM_OPEN#
```

See Also: "Page Templates" on page 9-24

Calling JavaScript from a Button

Calling a JavaScript from a button is a great way to confirm a request. Oracle HTML DB actually uses this technique for the delete operation of most objects. For example, when you delete a button, a JavaScript Alert Box appears asking you to confirm your request. Consider the following example:

```
<script language="JavaScript1.1" type="text/javascript">
  function deleteConfirm(msg)
  {
var confDel = msg;
if(confDel ==null)
  confDel= confirm("Would you like to perform this delete action?");
else
  confDel= confirm(msg);

if (confDel== true)
  doSubmit('Delete');
  }
</script>
```

This example creates a function to confirm a delete action and then calls that function from a button. Note that the function optionally submits the page and sets the value of the internal variable `:REQUEST` to `Delete`, thus performing the delete using a process that conditionally executes based on the value of request.

Note that when you create the button you would need to select **Action Redirect to URL without submitting page**. Then, you would specify a URL target such as the following:

```
javascript:confirmDelete('Would you like to perform this delete action?');
```

See Also: "Creating a Button Using a Wizard" on page 8-32

Creating Dependent Select Lists

You may use a select list to determine the range of values of another select list on the same page. You can achieve this functionality by having a driving select list submit values to a subsequent select list. You incorporate these values in the subsequent select list as a bind variable in the WHERE clause of its query.

You can have one LOV drive another LOV by:

- Creating a basic form.
- Defining two list of values. Note that the driving LOV must submit the page after a value is chosen.
- Defining a branch that branches back to the current page.

Consider the following example. The first LOV enables the user to pick a state.

```
SELECT state_name d, state_id v
FROM states
```

The second LOV selects the country name and country ID based on the state selected in the first LOV.

```
SELECT county_name d, county_id v
  FROM counties
WHERE state_id = :Px_STATE_ID
```

See Also:

- "Creating Forms" on page 8-17
- "Creating Lists of Values" on page 8-17
- "Controlling Flow Using Branches" on page 10-6

Creating a Help Page

Oracle HTML DB includes built-in attributes to make creating help for your application quick and easy. Creating help for your application involves the following steps:

- Create a dedicated help page and help region
- Define page help text
- Define item help text
- Create a navigation bar icon to link to your help page

Help created in Oracle HTML DB displays on a dedicated help page. To access help, users click a link that takes them to a dedicated help page. This help page displays page and item help topics specific to the page they are viewing.

Topics in this section include:

- Creating a Help Page and Region
- Defining Help Text
- Creating a Help Navigation Bar Icon

Creating a Help Page and Region

The first step in creating a help for your application it to create a dedicated page and Help Text region.

To create a new Help Text region:

1. Create new page for your help. (See "Adding Additional Pages" on page 8-3.)
2. Navigate to the Page Definition of your help page. (See "Viewing a Page" on page 7-11.)
3. Under Regions, the **Create** icon.
4. When prompted to select a region type, select **Help Text**.
5. Follow the on-screen instructions.

Defining Help Text

You define help text for a page or single item by editing attributes. Ideally, you would define these attributes as you create your application. For simplicity, however, the following procedures describe how to define this text after the fact.

To define page help text:

1. Navigate to the Page Definition for the page for which you want to add page help. (See "Viewing a Page" on page 7-11.)
2. Click **Edit Attributes** to view the existing page attributes.
3. Scroll down to **Page Help Text**.
4. Enter your help text in the field provided.
5. Click **Apply Changes**.

Repeat the previous procedure for each page requiring page help text.

To define item help text:

1. Navigate to the Page Definition for the page for which you want to add item help.
2. Under Items, click name of the item you want to edit.
3. Scroll down to **Help Text**.
4. Enter your help text in the field provided.
5. Click **Apply Change**.

Repeat the previous procedure for each item requiring help text.

About the Item Help Text Report

If you are including item help in your application, you can edit multiple item help topics at once using the Bulk Edit Item Help report.

To view the Bulk Edit Item Help report:

1. Navigate to the Workspace home page.
2. Select an application name from the Applications list.
3. From the Tasks list, select **View Application Reports**.

   Application Reports page appears.
4. Select **Item Help Text**.

5. Under Item Help Text column, make the appropriate edits.
6. Click **Apply Changes**.

Creating a Help Navigation Bar Icon

Once you have created your help, the next step is to create a navigation bar icon so users can link to it.

To create a navigation bar icon:

1. Navigate to the Page Definition. (See "Viewing a Page" on page 7-11.)
2. Under Navigation Bar, click the **Create** icon.
3. Specify the appropriate NavBar entry attributes:
   - Sequence
   - Alt Tag Text
   - Icon Image Name
   - Image Height and Image Width
   - Text

   Specify the target location.
4. To specify the target location:
   - From Target type, select **Page in this application**.
   - In Page, specify the page number.
   - In Request, type:

     ```
     &APP_PAGE_ID.
     ```

     By specifying substitution string *&APP_PAGE_ID* as the Request, you are instructing the HTML DB engine to display help text for the current page when the user clicks this icon.

9

Controlling Page Layout and User Interface

This section describes different ways you can customize your application's user interface and page layout including customizing regions, editing item attributes, customizing templates, and incorporating cascading style sheets and images.

This section contains the following topics:

> **See Also:** "Adding Navigation" on page 10-1 for more information on creating navigation bars, tabs, menus, lists, and trees

Understanding Page Layout

Oracle HTML DB renders pages by combining templates with application components defined by the developer and data in the database.

The overall framework or structure of a page is determined by the page template. For example, the page templates controls whether a page uses tabs and a navigation bar. It may also define whether a page includes a bar on the left side that serves as a placeholder for navigation or secondary content. Finally, a page template may include definitions of region positions, which enable precise control over placement of regions using HTML tables or style sheet definitions. The page template itself is composed of HTML combined with substitution strings which will be substituted with the appropriate Oracle HTML DB components at runtime.

As a developer, you add content on a page by creating regions. Each region may be a report based on a SQL query or a list of navigation links provided by an Oracle HTML DB list. You position a region either relative to other regions (that is, based on its sequence number and column), or by using a region position defined in the page template. The style of the region is also controlled by the region template. Like the page template, the region template defines the structure of the area that the region takes up on a page. It defines whether the region title is displayed and where it is displayed relative to the main content, or the body. A region may also define absolute positions for buttons.

Displaying Components on Every Page of an Application

Page zero of your application functions as a master page. The HTML DB engine renders all components you add to page zero on every page within your application. You can further control whether the HTML DB engine renders a component or runs a computation, validation, or process by defining conditions.

To create a page zero:

1. Create a new page.
2. Specify the page number as zero (0).

See Also:

- "Adding Additional Pages" on page 8-3
- "Understanding Conditional Rendering and Processing" on page 6-7
- "Available Conditions" on page A-1

How Item Attributes Affect Page Layout

An item is part of an HTML form and can be a text field, text area, password, combobox, check box, and so on. You can alter the appearance of a page by changing the item attributes. For example, these attributes can impact where a label displays, how large an item will be, whether the item will display next to, or below, the previous item.

See Also: "Creating Items" on page 8-35

To edit item attributes:

1. Navigate to the Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Items, click the item name.

   The Item Definition appears.

Table 9–1 describes how item attributes affect the layout of a page.

Table 9–1 Item Attributes Affecting Page Layout

Heading	Attribute	Description
Displayed	Sequence	Determines the order in which items are rendered within a region.
Displayed	Region	Defines the region in which the item displays. All items must be in a region.
Displayed	Begin On New Line	Determines whether this item displays on the same line as the previous item or whether it displays on the next line.
Displayed	Begin On New Field	Determines whether this item displays in the next column or in the same column as the previous item.
Displayed	ColSpan	Items are laid out in HTML tables. Defines the value to be used for the COLSPAN attribute of the table cell containing an item.
Displayed	RowSpan	Items are laid out in HTML tables. Defines the value to be used for the ROWSPAN attribute in the table cell that the item displays in.
Label	Label	Enter the label for this item. You may include HTML, JavaScript, and shortcuts. You can also use the substitution string `#CURRENT_ITEM_NAME#` to obtain the name of the item associated with this label.
Label	Horizontal/Vertical Alignment	Controls the placement as well as the horizontal and vertical alignment of the label. Labels can be displayed above, below or to the left of the item.
Label	Template	Specifies the label template. Use label templates to apply a consistent appearance to labels in your application.
Label	HTML Table Cell Attributes	Defines additional attributes for the cell containing this item's label (for example, `nowrap="nowrap"`).
Label	Post Element Texts	Specifies additional attributes for the HTML table cell used to display each individual option in a radio group or set of check boxes. May include HTML, JavaScript, and shortcuts. You may reference the following substitution strings: ▪ `#CURRENT_FORM_ELEMENT#` obtains the name of the HTML form element that this post element text is associated with. ▪ `#CURRENT_ITEM_NAME#` obtains the name of the item that this post element text is associated with.
List of Values	Columns	Applies to radio groups and check boxes. Defines the number of columns to use to display the values defined in the List of Values. By default, all values display in one column.
Conditional Display	Condition Type and Expressions	Defines conditions and appropriate expressions that determine whether an item displays. **See Also:** "Understanding Conditional Rendering and Processing" on page 6-7
Read Only Display Settings	Read Only Condition Type	Defines conditions and expressions that determine whether the item will be display as read-only. Use this attribute to display certain items to a set of users as updatable, while displaying that same set of items to others users as non-updatable. Reduces the need to code duplicate interfaces for different users.

Customizing Regions

A region is a area on a page that serves as a container for content. Each page can have any number of regions. You control the appearance of a region through a specific region template. The region template controls the look of the region, the size, determines whether there will be a border, a background color, and what type of fonts display. A region template also determines the standard placement for any buttons placed in region positions.

You can use regions to group page controls (such as items or buttons). You can create simple regions that do not generate additional HTML, or create elaborate regions that frame content within HTML tables or images.

Regions display in sequence within HTML table columns. You can also explicitly place regions in positions defined in the page template. You can also choose to display regions conditionally.

Topics in this section include:

- Creating a Region
- How Region Attributes Affect Page Layout
- Incorporating Content from Other Web Sites
- Rendering HTML Using Custom PL/SQL

Creating a Region

You create new regions by running the Create Region Wizard.

To create a new region:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.
3. Select a region type and follow the on-screen instructions.

When you create a region you select a region type. The HTML DB engine interprets a region differently based the type you select. Table 9–2 describes available region types.

Table 9–2 Region Types

Region Type	Description
HTML	When you select HTML, the wizard prompts you to select one of the following: ▪ **HTML** - Functions as containers for items and contain the HTML you provide. Any HTML you type may contain substitution strings. ▪ **Other, HTML Text (escape special characters)** - Same as HTML region, but the HTML DB engine escapes special characters before they are rendered. ▪ **Other, HTML Text (with shortcuts)** - Same as HTML region, but with support for shortcuts. ▪ **Other, Help Text** - Enable you to provide page level help. **See Also:** "Utilizing Shortcuts" on page 8-48 and "Creating a Help Page" on page 8-53
Report	Report regions can be defined by a SQL query you write, or by using a wizard to guide you through the steps needed to write a query. **See Also:** "Creating Reports" on page 8-8
Calendar	Calendar regions are used to contain a monthly calendar. **See Also:** "Creating Calendars" on page 8-43

Table 9–2 (Cont.) Region Types

Region Type	Description
Chart	Chart regions contain line, bar, or pie charts based on SQL queries. **See Also:** "Creating Charts" on page 8-24
List	List regions are used for navigation and may consist of links or images. Individual list entries can be conditionally displayed. **See Also:** "Creating Lists" on page 10-11
Menu	Menu regions are navigational controls consisting of predefined, hierarchically organized links. **See Also:** "Creating Menus" on page 10-7
PL/SQL Dynamic Content	Regions based on PL/SQL enable you to render any HTML or text using the PL/SQL Web Toolkit.
URL	URL based regions obtain their content by calling a Web server using a predefined URL. **See Also:** "Incorporating Content from Other Web Sites" on page 9-7
Tree	Trees are a hierarchical navigational control based on a SQL query executed at runtime. It enables the user to expand an collapse nodes. **See Also:** "Creating Trees" on page 10-14

See Also:

- *Oracle Database Application Developer's Guide - Fundamentals* for more information on developing Web applications with PL/SQL
- *PL/SQL Packages and Types Reference* for more information on htp packages

How Region Attributes Affect Page Layout

You can alter the appearance of a page by changing the region attributes.

To edit region attributes:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Regions, click the region name.

   The Region Definition appears.

Table 9–3 describes region attributes that affect the layout of a page.

Table 9–3 Region Attributes Affecting Page Layout

Attribute	Description
Conditional Display, Condition Type and Expressions	Defines conditions and appropriate expressions that determine whether the region displays. Conditions may reference session state, the currently logged in user, or HTML DB environment preferences (such as whether or not a page is in Print View mode). **See Also:** "Understanding Conditional Rendering and Processing" on page 6-7 and "Optimizing a Page for Printing" on page 9-39
Header and Footer Text, Region Header	Specifies HTML text to be displayed at the top of the region (just before the `#BODY#` content).
End User Customization	Enables end user customization. To utilize this feature, you must include the `#CUSTOMIZE#` substitution string in the Header, Body, or Footer section of the page template. **See Also:** "Enabling Users to Customize a Page" on page 9-16
User Interface Attributes, Column	Determines which column the region displays in. If two regions are in the same display point, you can place them next to one other by setting the second region to display in column 2. Many regions can display in each column and the display order of the regions within the region display point and column is controlled by the region display sequence number.
User Interface Attributes, Template	Determines the look of the region. Select from the region templates defined in the application. To view template attributes clicking on the template name on the Page Definition. **See Also:** "About Themes and Templates" on page 7-29
User Interface Attributes, Sequence	Specifies the display order of the regions within the page.
User Interface Attributes, Display Point	Identifies where within the page the region displays. Regions are rendered in order of sequence number within a Display Point. Click the View icon to see the page layout and select a position. The possible display points for a region are determined by the page level template (which is a page attribute). If no page level template is selected, the default page level template, defined in the Application Definition is used.
User Interface Attributes, Region HTML table cell attributes	Defines additional attributes to be used in the HTML table cells when regions display in multiple columns. The attributes control the cells in the table used to lay out a regions in multiple columns.

Controlling Region Positioning

When you create a region, you must specify its position (or Display Point) on the page. You can choose either a default position (such as Page Template Body) or a user defined position in the template (such as Page Template Region Position 1.)

In addition to Display Point, you can specify which column the region will be placed in. When you place regions in multiple columns, Oracle HTML DB automatically renders the necessary HTML to produce a multiple column layout.

Specifying a Region Header and Footer

In addition to the body content of a region (which can be a report, a chart, or HTML with form elements), you can specify additional HTML to be placed above and below a region or in its header and footer. The region footer supports the following substitution strings:

- `#TIMING#` shows the elapsed time in seconds used when rendering a region. You may use this for debugging purposes.

- #ROWS_FETCHED# shows the number of rows fetched by the Oracle HTML DB reporting engine (the page size). You may use these substitution strings to display customized messages to the user. For example:

  ```
  Fetched #ROWS_FETCHED# rows in #TIMING# seconds.
  ```

- #TOTAL_ROWS# displays the total number of rows that satisfy a SQL query used for a report.
- #FIRST_ROW_FETCHED# and #LAST_ROW_FETCHED# displays the range of rows displayed. For example:

  ```
  Row(s) #FIRST_ROW_FETCHED# through #LAST_ROW_FETCHED# of #ROWS_FETCHED#
  displayed
  ```

Enabling Users to Customize a Page

You can use the End User Customization attribute to enable users to turn regions on and off in a running application.

To enable end user customization:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Regions, click the region name.

   The Region Definition appears.
3. Scroll down to End User Customization and select one of the following:
   - Customizable and Not Shown By Default
   - Customizable and Shown By Default
4. In Customized Option Name, enter the label that represents this region on the page to the user.
5. Include the #CUSTOMIZE# substitution string in the Header, Body, or Footer section of the page template.

To utilize this feature, you must include the #CUSTOMIZE# substitution string in the Header, Body, or Footer section of the page template.

If at least one region supports end user customization, a link called Customize appears wherever you include the #CUSTOMIZE# substitution string in the page template. When users click this link, a window displays enabling them to turn on and off regions on the page.

> **See Also:** "Customizing Templates" on page 9-16

Incorporating Content from Other Web Sites

Typically, pages in an application are based on data stored in an Oracle database. To incorporate content from other servers, you can create a region based on a URL to display content. For example, suppose you wanted to reference the current Oracle stock price. You could create a region of type URL based on a URL such as the following:

```
http://quote.yahoo.com/q?d=b&s=ORCL
```

You could then create a item called STOCK_SYMBOL and base your region on a stock price entered by the user. For example:

```
http://quote.yahoo.com/q?d=b&s=&STOCK_SYMBOL.
```

Sometimes (as is the case with the previous example) the HTML returned to the region is more than is needed. To restrict the HTML displayed you can use the following region attributes:

- URL (discard until but not including this text)
- URL (discard after and including this text)

Note that the previous example may require that you set the Proxy Server application attribute. If you do not set the Proxy Server application attribute, you will get an error message. Oracle HTML DB uses the Oracle `utl_http.request_pieces` function to obtain the HTML generated from the given URL.

> **See Also:** "Editing Application Attributes" on page 7-4 for more information on setting the Proxy Server application attribute

Managing Themes

Themes are collections of templates that can be used to define the layout and style of an entire application.The idea behind a theme is to provide a complete set of templates that accommodate every UI pattern that may be needed in an application. Templates are organized first by type (button, calendar, label, list, menu, page, popup LOV, region, and report) and then by template classes, identifying the purpose of the each template within that type. Each template type provides a group of standard classes and eight custom classes. These classifications enable Oracle HTML DB to map templates between themes, making it easy to quickly change the entire look and feel of an application.

Topics in this section include:

- Accessing the Themes Page
- Changing Default Templates in a Theme
- Creating a New Theme
- Switching an Active Theme
- Copying a Theme
- Deleting a Theme
- About Exporting and Importing Themes
- Changing a Theme Identification Number
- Viewing Theme Reports

> **See Also:** "Customizing Templates" on page 9-16

Accessing the Themes Page

You manage themes on the Themes page.

To access the Themes page from Shared Components:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **Themes and Templates**.

The Themes page appears. The currently selected theme displays a check mark in the current column.

To access the Themes page from the Page Definition:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Select the theme title.

Changing Default Templates in a Theme

Each theme contains templates for every type of application component and region type. You can change the selected default templates for a theme on the Define Theme page.

You can override these defaults, by either selecting another template when you create new components or regions, or by changing the template on the component or region attributes page.

To review of change the default templates in a theme:

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.
2. In the Themes list, click the **Edit** icon adjacent to the theme name.

   The theme name displays at the top of the page.
3. To change the theme name, enter a new name in the Name field.
4. To change a default template, make a new selection from the appropriate list.

Table 9–4 describes the default templates available under the section Default Templates by Component.

Table 9–4 Default Templates by Component

Attribute	Description
Page	Identifies the default template for displaying pages. If a developer does not explicitly choose a template then the HTML DB engine uses the template specified here. Once defined, this default template appears on the Edit Application Attributes page under the heading Application Template Defaults.
Error Page	Optional. Specifies a page template to use for errors that display on a separate page as opposed to those that display inline. Leave this attribute blank if you do not wish to use a template designed to display errors. Once defined, this default template appears on the Edit Application Attributes page under the heading Application Template Defaults.

Table 9–4 (Cont.) Default Templates by Component

Attribute	Description
Printer Friendly Page	Identifies the template to be used when the HTML DB engine is in printer friendly mode. When calling the HTML DB to render a page, you have the option to identify a printer friendly attribute with values of YES or NO. If you select `YES`, then the page displays using a printer friendly template. The HTML DB engine displays all text within HTML form fields as text. The printer friendly template does not need to have the `#FORM_OPEN#` or `#FORM_CLOSE#` tags. The objective is to be able to display information with few tables and in a format suitable for printing. Once defined, this default template appears on the Edit Application Attributes page under the heading Application Template Defaults. **See Also:** "Optimizing a Page for Printing" on page 9-39
Button	Identifies the default button template used when creating a new button.
Calendar	Specifies the default calendar template used when you create new calendar.
Label	Identifies the default label template used when you create new label.
List	Specifies the default list template used when you create new list.
Menu	Identifies the default menu template used when you create new menu.
Region	Specifies the default region template used when you create new region.
Report	Identifies the default region template used when you create a report.

Table 9–5 describes the default templates available under the section Default Templates by Component.

Table 9–5 Region Templates by Region Type

Attribute	Description
Charts	Default chart template used when creating a chart.
Forms	Default form template used when creating a form.
Lists	Default region template used when creating a list.
Menus	Default region template used when creating a menu.
Reports	Default region template used when creating a report.
Tabular Forms	Default region template used when creating a tabular form.
Wizards	Default region template used when creating a new wizard component.

Creating a New Theme

You can create a new theme from scratch or select an existing theme from the HTML DB repository.

To create a new theme:

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.

b. On the Application Builder home page, click **Shared Components**.

c. Under User Interface, select **Themes and Templates**.

2. Click **Create Theme**.
3. Specify whether to select a theme from the HTML DB repository, or create a theme from scratch.
4. If you select **From the HTML DB Repository**:
   a. Select a theme from the repository.
   b. Click **Create**.
5. If you select **From Scratch**:
   a. On the Define Theme page, select default templates for the new theme.
   b. Click **Create**.

Switching an Active Theme

When you switch to a new theme, all components that are assigned a template are assigned to a corresponding template in the new theme. Application Builder accomplishes template mapping through the assignment of a template class identifiers.

To apply a theme to an application:

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.
2. Click **Switch Theme**.

   A Theme Compatibility report appears. When you switch to a new theme, Application Builder maps all components currently using a template to a template in the new theme.
3. From the Switch to Theme list, select a new theme.
4. Review the Status column to identify problematic mappings:
   - **Check** indicates the mapping was successful.
   - **Warning** indicates there are more than one template in the theme you are switching to with the identified class. The warning provides a select list to choose the appropriate template.
   - **Error** indicates that Application Builder was unable to map the class between the themes. Ensure that a class is identified for the templates in both themes.
5. Click **Next** to continue.
6. Click **Switch Theme**.

See Also: "Editing Templates" on page 9-19

Copying a Theme

Each theme is identified by a numeric identification number (ID). When you create a copy of theme you specify a new theme ID. Copying a theme is useful if you wish experiment with editing a theme or to export a theme with a different ID.

To copy a theme:

1. Navigate to the Themes page:
    a. Navigate to the Workspace home page and select an application from the Application list.
    b. On the Application Builder home page, click **Shared Components**.
    c. Under User Interface, select **Themes and Templates**.
2. From the Tasks list, select **Copy**.
3. From Copy From Theme, select the theme you wish to copy.
4. From Copy to this Theme Identification Number, enter a new ID for the theme.
5. Click **Copy Theme ID**.

Deleting a Theme

You can only delete inactive themes. When you delete a theme, Application Builder only removes inactive templates.

To delete a theme:

1. Navigate to the Themes page:
    a. Navigate to the Workspace home page and select an application from the Application list.
    b. On the Application Builder home page, click **Shared Components**.
    c. Under User Interface, select **Themes and Templates**.
2. From the Tasks list, select **Delete**.
3. From Remove Theme, select the theme you wish to delete.
4. Click **Remove Theme**.

About Exporting and Importing Themes

You export a theme in the same way you export any related application file. Exporting a theme from one Oracle HTML DB development instance to another involves the following steps:

- Export the theme using the Export Theme utility.
- Import the exported file into the target Oracle HTML DB instance.
- Install the exported file from Export Repository

See Also: "Exporting Themes" on page 12-9 and "Importing Export Files" on page 12-10

Changing a Theme Identification Number

Each theme has a numeric identification number (or ID). You can use the Change Theme ID utility to change a theme ID to another number. Changing a theme ID is

useful when you wish to export a theme with a different number and then import it into another application.

To change a theme identification number:

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.
2. From the Tasks list, select **Change Identification Number**.
3. On the Change Theme ID page:
   a. Select a theme.
   b. Specify a new identification number.
   c. Click **Next**.

Viewing Theme Reports

Application Builder includes a number of reports designed to help you manage themes and templates.

Topics in this section include:

- Viewing All Templates in a Theme
- Viewing Theme Template Counts
- Viewing File References
- Viewing Class References
- Viewing Template Substitution Strings

Viewing All Templates in a Theme

To view all templates that comprise a theme:

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.
2. On the Themes page click **Theme Reports**.
3. From Report, select **Application Templates**.
4. From the Theme list, select a theme.
5. Click **Go**.

   A listing of templates displays listing the template type, template name, the associated theme, and template class.
6. To edit a template, select the template name.

Viewing Theme Template Counts

Accessing the Theme Template Count report lists which template classes currently have templates created for them.

To view the Theme Template Count report

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.
2. On the Themes page click **Theme Reports**.
3. From Report, select **Theme Template Counts**.
4. From the Theme list, select a theme.
5. If you are using custom classifications, select **Show Custom**.
6. Click **Go**.

Viewing File References

Accessing the File References report displays a listing of all files associated with templates, shared components, or page components in the current application.

To view the File References report:

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.
2. On the Themes page, click **Theme Reports**.
3. On the File References page:
   a. From Show, select the type of component to include in the report. If you do not make a selection, no results are returned.
   b. From Show Files, select one of the following:
      – **With context** displays the component, the theme identification number, the component name, the image (if applicable), and the page number. Select the page number to link to a Page Definition.
      – **Without context** displays only the file name and the image (if applicable).
   c. From File Extensions, select the type of extensions to search for.
   d. Click **Go**.
4. To download a comma delimited file (.csv) version of this report, click **Download CSV**.

Viewing Class References

Accessing the Class References report displays a listing of classes associated with templates, shared components, or page components in the current application.

To view the Class References report:

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.
2. On the Themes page click **Theme Reports**.
3. On the Theme Reports page:
   a. From Report, select **Class References**.
   b. From Theme, select a theme.
   c. Click **Go**.
4. On the Class References page:
   a. From Show, select the components to check for a class reference. If you do not make a selection, no results are returned.
   b. From Show Class Names, select one of the following:
      – **With context** displays the component, the theme identification number, the component name, the image (if applicable), and the page number.
      – **Without context** displays only the referenced class.
   c. Click **Go**.
5. To download a comma delimited file (`.csv`) version of this report, click **Download CSV**.

Viewing Template Substitution Strings

Use the Template Substitution Strings report to view all supported substitution strings by component.

To view the Substitution String report:

1. Navigate to the Themes page:
   a. Navigate to the Workspace home page and select an application from the Application list.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.
2. On the Themes page click **Theme Reports**.
3. On the Theme Reports page:
   a. From the Report list, select **Template Substitution Strings**.
   b. From the Theme list, select which themes to include in the report.
   c. Click **Go**.
4. To link to a template definition, select the template name.

See Also: "Using Substitution Strings" on page 6-16

Customizing Templates

The HTML DB engine creates the user interface of an application based on a named collection of templates called a theme. Templates control the look and feel of these components in your application. If you need to create a custom template, it is generally simplest to start with an existing template and then modify it. Once you have created one or more default templates, you can modify those templates to fit your specific needs.

Topics in this section include:

- About Cascading Style Sheets
- Selecting a Default Page Template
- Viewing Templates
- Creating a New Template
- Viewing Template Reports
- Editing Templates
- Button Templates
- Calendar Templates
- Label Templates
- List Templates
- Menu Templates
- Page Templates
- Popup LOV Templates
- Region Templates
- Report Templates

See Also: About Themes and Templates on page 7-29

About Cascading Style Sheets

A cascading style sheet (CSS) provides a way to control the style of a Web page without changing its structure. When used properly, a CSS separates visual attributes such as color, margins, and fonts from the structure of the HTML document. Oracle HTML DB includes themes that contain templates that reference their own CSS. The style rules defined in each CSS for a particular theme also determine the way reports and regions display.

See Also: "Using Custom Cascading Style Sheets" on page 9-39

Selecting a Default Page Template

You can specify a default page template in two ways:

- Select a default page template within a specific theme
- Select a specific page template on a page by page basis.

By default the HTML DB engine uses the Page template specified on the Themes page.

Selecting a Page Level Template within a Theme

To specify a default page template within a theme:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **Themes and Templates**.

   The Themes page appears.
5. In the Themes list, click the **Edit** icon adjacent to the theme name.
6. Under Default Templates by Component, make a selection from the Page list.

See Also: "Changing Default Templates in a Theme" on page 9-9

Selecting a Page Level Template for a Specific Page

To specify an page level template for a specific page:

1. Navigate to the Workspace home page
2. From the Applications list, select an application.
3. Select a page name from the Pages list.
4. Click **Edit Attributes**.
5. Locate the first section Primary Page Attributes.
6. Make a selection from the Page Template list.

Viewing Templates

You can view all available templates on the Templates page. Alternatively, you can view the templates used on a specific page on the Page Definition.

See Also: "Viewing All Templates in a Theme" on page 9-13

To view existing templates:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **Themes and Templates**.

   The Themes page appears.
5. To view templates you can:
   - View all templates by selecting **View Templates** from the Tasks list.
   - View templates in a specific theme by selecting a theme name.

   The Templates page appears.
6. You can narrow your search by selecting:
   - A specific theme from the Theme list
   - A template type from the Show list
   - A template status from the View list

7. To view a template definition, click the template name. Or, to see a preview of a template, click **Run** in the Preview column.

   Note: Not all template types have the preview capability.

To view existing templates from the Page Definition:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, select a page from the Pages list.

   The Page Definition appears. Templates associated with the current page display under the Templates heading in the far right column.
4. To view attributes of an existing template, click the template name.

Creating a New Template

If you need to create a custom template, it is generally simplest to start with an existing template and then modify it. Once you have created one or more default templates, you can modify those templates to fit your specific needs.

To create a custom template:

1. Navigate to the Templates page.
   a. From the Workspace home page, select an application.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.

      The Themes page appears.
   d. From the Tasks list, select **View Templates**.

      The Templates page appears.
2. Click **Create**.
3. Select the type of template you would like to create.
4. Select a creation method:
   - **From Scratch**
   - **As a Copy of an Existing Template**
5. Follow the on-screen instructions, being careful to associate your template with the correct theme.

Viewing Template Reports

Oracle HTML DB includes reports describing template utilization, subscriptions, and edit history.

To view template reports for the current application:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

4. Under User Interface, select **Themes and Templates**.

   The Themes page appears.

5. To view templates you can:
   - View all templates by selecting **View Templates** from the Tasks list.
   - View templates in a specific theme by selecting a theme name.

6. To view template reports click the following buttons:
   - **Utilization** displays template utilization in the current application for all template types (page, report, region, label and list).
   - **Subscriptions** displays subscribed templates in your application.
   - **History** details recent changes to templates by developer and last update date.

Editing Templates

Once you create a custom template, you can quickly edit it from either the Templates page or from the Page Definition.

To edit an existing template from the Templates page:

1. Navigate to the Templates page.
   a. From the Workspace home page, select an application.
   b. On the Application Builder home page, click **Shared Components**.
   c. Under User Interface, select **Themes and Templates**.

      The Themes page appears.

   d. From the Tasks list, select **View Templates**.

      The Templates page appears.

2. Locate the template you wish to edit and select the template name. You can narrow your search by selecting:
   - A template type from the Show list
   - A template status from the View list

3. Follow the on-screen instructions.

As you edit templates, you can make changes in one window and run your application in another by selecting **Return to Page**. Selecting this check box, keeps the page you are editing current after you click **Apply Changes**.

Button Templates

Button templates enable application developers to customize the look and feel of a button. To build a button, you can use multiple images or HTML tags. Using button templates is optional.

Button Template Attributes

This section describes specific sections of the Button Template page.

Button Template **Template Name** identifies the name that identifies the template. Use the **Translatable** check box to indicate if the template contains text strings which require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use Template Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Template Text Defines the button template that displays. You have the option of including standard application substitutions. For example, `&ITEM_NAME` values can be substituted at rendering time. Button templates support the following substitution strings:

- `#LABEL#` is replaced with a button label.
- `#LINK#` is replaced with a URL. The URL then calls a #doSubmit# or a redirect JavaScript which submits the page (that is, setting the request value), or simply redirects it to the supplied URL.

Comments Use this attribute to record developer comments.

Calendar Templates

Calendar templates control the appearance and placement of a calendar. Calendar templates frequently use HTML tables to arrange dates. You place calendar attributes using substitution strings such as `#DD#` and `#MONTH#`. A list of supported substitution strings appears on the right side of Calendar Template Attributes page. Note that template substitution strings must be in upper case and begin and end with a pound sign (#).

See Also: "Creating Calendars" on page 8-43

Calendar Template Attributes

This section describes specific sections of the Calendar Template page.

Calendar Template Identification **Name** identifies the name that identifies the template. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use Template Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Month Attributes Enter formats for the month title, day of the week, month open, and month close.

Week Attributes Enter HTML to open and close a week.

Day Attributes Enter HTML to format the days which occur during the work week (that is, Monday through Friday).

Non-Day Attributes A non-day is not part of the current month. For example, suppose the first of a month is a Monday, but the week starts on a Sunday. Since Sunday is not be part of the current month, Sunday would be a non-day. Use these attributes to format a non-days.

Weekend Attributes Enter HTML used to format days which occur on the weekend.

Label Templates

Label templates are designed to centrally manage HTML markup of page item labels. Each item can have an optional label. You can control how these labels display using label templates. For example, you could create a label template called "Required Field" which references an image (such as an asterisk) to indicate to the user that the field is required.

Label templates enable you to define a before and after text string that gets prepended and appended to the item.

Label Template Attributes

This section describes specific sections of the Label Template page.

Label Template Attributes **Template Name** identifies the name that identifies the template. Use the **Translatable** check box to indicate that the template contains text strings which require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Template Definition (normal display) In **Before Label** enter HTML to display before the item label. Before Label supports the substitution strings `#CURRENT_FORM_ELEMENT#`; `#CURRENT_FORM_ID#`, and `#CURRENT_ITEM_NAME#`.

In **After Label**, enter HTML to display after the item label.

Label Template Attributes (error display) In **On Error Before Label** enter HTML to precede the item label when a application displays an inline validation error message for the item.

In **On Error After Label** enter HTML to be appended to the item label when a application displays an inline validation error message for the item. This attribute supports the substitution strings `#CURRENT_FORM_ELEMENT#`, `#CURRENT_FORM_ID#`, and `#CURRENT_ITEM_NAME#`.

Template Comments Use this attribute to record developer comments.

List Templates

A list is a shared collection of links. You control the appearance of a list through list templates. Using template attributes, you can also define a list element to be either current or non current for a specific page.

See Also:

- Online help for more information on using specific sections of the Edit List Template page
- "Creating Lists" on page 10-11

List Template Attributes

This section describes specific sections of the List Template page.

Template Identification **Name** identifies the name that identifies the template. Use the **Translatable** check box to indicate that the template contains text strings which require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Before List Elements Enter HTML that displays before any list elements. You can use this attribute to open an HTML table or HTML table row.

List Element Display Defines current and noncurrent list templates.

- **List Template Current.** Enter HTML or text to be substituted for the selected (or current) list template.
- **List Template Noncurrent.** Enter HTML or text to be substituted for the unselected (or noncurrent) list template.

After List Elements Enter HTML that displays after displaying all list elements. You can use this attribute to close a HTML table opened in the Before List Elements attribute.

Comments Use this attribute to record developer comments.

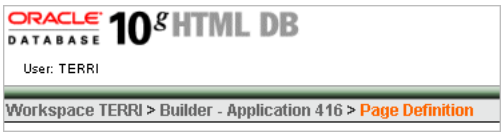
Menu Templates

A menu template controls the display of menus. You select a menu template when you create a region.

Breadcrumb Style Menu Navigation

Breadcrumb style menus usually indicate where the current page is relative to other pages in the application. In addition, users can click a specific page to instantly view it. Oracle HTML DB includes breadcrumb paths beneath the standard tabs (or second level navigation tabs) at the top of each page as shown in the following example:

Figure 9–1 Breadcrumb Style Menu



See Also:

- Online help for more information on using specific sections of the Edit Menu Template page
- "Creating Menus" on page 10-7

Menu Template Attributes

This section describes specific sections of the Menu Template page.

Menu Template Identification **Name** identifies the name that identifies the template. Use the **Translatable** check box to indicate that the template contains text strings which require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Menu Template Subscription Use Template Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Menu Template Style) Select one of the following menu template styles:

- **Parent to Leaf (breadcrumb style)** displays the current page menu option, its parent to the left, and so on until the root node is reached.
- **Parent Menu** displays all menu options for the current pages parent menu (that is, one level up from current menu).
- **Current Menu** displays all menu options in sequence with a common parent.
- **Child Menu** displays all menu options that are children of the current page parent menu (that is, peers of the current menu).

Menu Option Control Table 9–6 describes available Menu Option Control attributes.

Table 9–6 Menu Option Control attributes

Attribute	Description
Before First	Defines text that displays before the first menu option.
Current Page Menu Option	Defines the look of a menu option that corresponds to the current page. This attribute supports the following substitution strings: ■ `#NAME#` specifies the short name of the menu option ■ `#LINK#` specifies the anchor target of the menu option ■ `#LONG_NAME#` specifies the long name of the menu option

Table 9–6 (Cont.) Menu Option Control attributes

Attribute	Description
Non Current Page Menu Option	Defines the look of a menu option that does not correspond to the current page. This attribute supports the following substitution strings: ▪ #NAME# specifies the short name of the menu option ▪ #LINK# specifies the anchor target of the menu option ▪ #LONG_NAME# specifies the long name of the menu option
After Last	Defines text that displays after the last menu option.

Template Attributes Use **Menu Link Attributes** to specify hypertext link attributes for a menu option.

Use **Between Levels** to specify text that displays between each level of a breadcrumb menu. For example, suppose your breadcrumb menus has three levels this text would be displayed at the "X" in the example that follows:

```
main X cars X porsche X 911
```

In **Max Levels** specify the number of levels when displaying menus in a breadcrumb style.

Comments Use this attribute to record developer comments.

Page Templates

Page templates define the appearance of a page. Each template consists of a header template, a body template, a footer template, and a number of subtemplate definitions. If you do not specify a page template as a page level attribute, then the HTML DB engine uses the default page template defined on the Define Theme page.

Page templates combine static HTML with substitution strings which are replaced at runtime. You use substitution strings to indicate the existence and placement of a component within a page template. You can further specify how a component should display using subtemplates.

Topics in this section include:

- Supported Page Template Substitution Strings
- Page Template Attributes

Supported Page Template Substitution Strings

Table 9–7 describes the available page template substitution strings. Note that all template substitution strings must be in upper case and begin and end with a pound sign (#).

Table 9–7 Page Template Substitution Strings

Substitution String	Description
#APP_VERSION#	May be used in the Header or Footer sections of the page template. You define the value of #APP_VERSION# in the Version attribute on the Edit Application Attributes page **See Also:** "Application Definition" on page 7-5
#BOX_BODY#	Identifies where the Body displays. If the Body is null, then #BOX_BODY# will be used instead.

Table 9–7 (Cont.) Page Template Substitution Strings

Substitution String	Description
#CUSTOMIZE#	May be used in the Header, Body, or Footer sections of the page template. The End User Customization section of the Region Definition enables you to turn on end user customization. To utilize this feature, you must also include the #CUSTOMIZE# substitution string in the page template. If at least one region supports end user customization, a link called **Customize** appears wherever the #CUSTOMIZE# substitution string appears in the page template. When users click this link, a window displays enabling them to turn on and off regions on the page. **See Also:** "How Region Attributes Affect Page Layout" on page 9-5
#FORM_CLOSE#	If a #FORM_OPEN# is included, then you must include a #FORM_CLOSE# in the header, body, or footer template. #FORM_OPEN# must appear before the #BOX_BODY# and #FORM_CLOSE# must appear after the #BOX_BODY# substitution string.
#FORM_OPEN#	Specifies where the HTML open form tag <form> is placed. You must include this substitution string in order to submit a form. You do not need to code your own form open, the HTML DB engine does it for you.
#GLOBAL_NOTIFICATION#	Displays the Global Notification attribute. Global notifications are intended to communicate system status, such as a pending system downtime. You can also use HTMLDB_APPLICATION.G_GLOBAL_NOTIFICATION to set this value if you wish to set it programmatically **See Also:** "Global Notifications" on page 7-8 for more information on the Global Notification attribute
#HEAD#	Used after the <head> open tag, but before the </head> close tag. You can optionally define the contents of #HEAD# for each page (for example, to reference additional style sheets or JavaScript libraries).
#LOGO#	Identifies an application logo. In the Logo section of the Edit Application Attributes page, you can identify an image and image attributes for an application logo. To utilize this feature, you must also include the #LOGO# substitution string in the Header or Body the page template. **See Also:** "Logo" on page 7-9
#NAVIGATION_BAR#	Defines the existence of navigation bar icons. A navigation bar will appear on every page in your application that uses a template which includes this substitution string.You can expand this substitution string using the Navigation bar subtemplate. **See Also:** "Subtemplate Definitions" on page 9-27 for more information on Navigation Bar subtemplate
#NOTIFICATION_MESSAGE#	Enables developers to communicate messages to the user. Defines where a summary of inline error messages is displayed. Inline error message that can be displayed next to a field, or inline in the notification area or both.

Table 9–7 (Cont.) Page Template Substitution Strings

Substitution String	Description
#ONLOAD#	May be used in the Header and Footer section of the page template and should be placed inside the <body> html tag. For example: `<body #ONLOAD#>` Use this string as substitute in a JavaScript call to be executed when a page is loaded by the Web browser. The JavaScript to be called can vary for each page.
#PARENT_TAB_CELLS#	Identifies the display of parent tabs. Parent tabs require standard tabs. If your application only has one level tabs, you do not need this substitution string. **See Also:** "Standard Tab Attributes" on page 9-27 for more information on defining Parent Tab Attributes
#REGION_POSITION_NN#	Identifies the exact placement of regions within a page. If no region is specified (for example, #REGION_POSITION_01#) then #REGION_POSITION_01# will be replaced with nothing.
#SUCCESS_MESSAGE#	Defines where in the page success and error message appear. If the page process runs without raising an error, then this text displays. You can customize the display of the success message for each template by adding HTML to be displayed before and after the success message.
#TAB_CELLS#	Identifies the display of standard tabs. **See Also:** "Standard Tab Attributes" on page 9-27 for more information on defining Standard Tab Attributes
#TITLE#	Defines the page title. Typically included within HTML title tags.

See Also:

- "Using Substitution Strings" on page 6-16
- "Adding Additional Pages" on page 8-3

Page Template Attributes

This section describes specific sections of the Page Template page.

Template Identification **Name** identifies the name that identifies the template. Use the **Translatable** check box to indicate that the template contains text strings which require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, select **Refresh**.

Header, Body, and Footer Definitions Each template consists of a header, a body, a footer, and subtemplate definitions. Use substitution strings to include dynamic content. All template substitution strings must be in upper case and begin and end with a pound sign (#). See online help for specific information as to which substitution strings are supported.

Header is the first section of the page template. Enter HTML that defines the `<Head>` section of the HTML document. Regions that display or processes and computations that execute AFTER HEADER, will display or execute immediately after this section in the template is rendered.

Body is the second section in the page template and is rendered after the header and before the footer section. Enter HTML that defines the `<Body>` section of the HTML document. At a minimum, you must include the `#BOX_BODY#` substitution string.It is strongly recommended you also include `#FORM_OPEN#` and `#FORM_CLOSE#`.

Footer is the third section in the page template that displays after the body.

Default Display Points **Breadcrumb Display Point** applies to generated components that use Breadcrumb style menus and defines where these Breadcrumb style menus are placed on the page. **Sidebar Display Point** applies to generated components that use Sidebars and defines where Sidebars are placed on the page.

Subtemplate Definitions Use **Subtemplate Definitions** to specify how a component should display. Available subtemplate definitions include:

- **Success Message.** Expands the `#SUCCESS_MESSAGE#` substitution string. You can define a success message either programmatically, or as an attribute of a process. If a success message exists and if the page template includes the `#SUCCESS_MESSAGE#` substitution string, then this subtemplate is used to render the message.
- **Navigation Bar.** Controls the display of navigation bar icons. Enter HTML or text that to be substituted when the `#NAVIGATION_BAR#` substitution string is referenced in the template header, body or footer. Use the `#BAR_BODY#` substitution string to identify where each navigation bar icon should display.
- **Navigation Bar. Entry** Enter HTML or text that to be substituted into the navigation bar `#BAR_BODY#` substitution string for each navigation bar entry. Use the following substitution strings to compose the navigation bar entry sub template.
- **Notification.** Enter HTML or text that to be substituted when the `#NOTIFICATION_MESSAGE#` substitution string is referenced in the template header, body or footer. Use the substitution string `#MESSAGE#` to indicate where in the Notification Message the body of the message will appear.

Standard Tab Attributes You must populate this attribute if your application includes standard tabs. Standard tabs may be placed in the header, body, or footer sections of the page template using the `#TAB_CELLS#` substitution string. The page template Header/Body/Footer defines the HTML table and rows. This subtemplate defines how these tabs display by defining the specific cell. Available attributes include:

- **Current Tab.** Enter HTML or text that to be substituted for the currently selected standard tab. Whether or not a tab is current is determined by standard tab attributes. Use the `#TAB_TEXT#` substitution string to position a tab's label and link within the template.

- **Non Current Standard Tab.** Enter HTML or text that will be substituted for the unselected standard tabs. Use #TAB_TEXT# to position a tab's label and link within the template.
- **Non Current Tab Font Attributes.** This attribute is part of the Standard Tab subtemplate. This value replaces the #TAB_FONT_ATTRIBUTES# substitution string.

See Also: "Creating Tabs" on page 10-4

Parent Tab Attributes You must populate this attribute if your application includes two levels of tabs. Enter HTML or text that will be substituted for the selected parent tabs. Parent tabs may be placed in the header, body, or footer section of the page template using the #PARENT_TAB_CELLS# substitution string. Parent tabs only display in conjunction with standard tabs. Available attributes include:

- **Current Parent Tab.** Enter HTML or text that will be substituted for the selected parent tabs. Whether or not a tab is current is determined by the page that displays and the standard tab set the page uses. Use #TAB_TEXT# to position a tab's label and link within the template.
- **Parent Current Tab Font Attributes.** This attribute is part of the Parent Tab subtemplate and expands the #PARENT_TAB_CELLS# substitution string. A "current tab" is a tab that is current for the current page. This attribute controls the font used to display the parent tab text.
- **Non Current Parent Tab.** Enter HTML or text that will be substituted for the unselected parent tabs. Use #TAB_TEXT# to position a tab's label and link within the template.
- **Parent Non Current Tab Font Attributes.** This attribute is part of the Parent Tab subtemplate and expands the #PARENT_TAB_CELLS# substitution string. A "non current tab" is a tab that is not current for the current page. This attribute controls the font used to display the parent tab text.

See Also: "Creating Tabs" on page 10-4

Image Based Tab Attributes Use this subtemplate for tabs that are entirely based on images. Available attributes include:

- **Current Image Tab.** Enter HTML to be used to indicate that an image based tab is currently selected. Include the #TAB_TEXT# substitution string to show the displayed name of the tab.
- **Non Current Image Tab.** Enter the HTML to be used to indicate that an image tab is not currently selected. Include the #TAB_TEXT# substitution string to show the displayed name of the tab.

Multi Column Region Table Attribute If the HTML DB engine displays regions in multiple columns in the same region position then HTML DB will render an HTML table. This attribute enables you to control the attributes of the <table> tag.

Error Page Template Control Use this attribute only when a page template will be designated as an error template. Use #MESSAGE# to place the error message and #BACK_LINK# to display a link back to the previous page. A template can be designated as an error template by editing the application attributes. For example:

```
#MESSAGE#
```

```
<br>

<a href="#BACK_LINK#">back</a>
```

Configuration Management You can use build options to enable or disable functionality. Most application attributes have a build option attribute.

Build Options have two possible values: INCLUDE and EXCLUDE. A component that is excluded is not considered part of the application definition at runtime.

> **See Also:** "Using Build Options to Control Configuration" on page 12-13

Comments Use this attribute to record developer comments.

Popup LOV Templates

Popup LOV template controls how popup lists display for all items defined as POPUP. You can only specify one popup LOV template for each theme.

> **See Also:** "Creating Lists of Values" on page 8-41

Popup List of Values Template Attributes

This section describes specific sections of the Popup List of Values Template page.

Application Identification **Theme** indicates the theme to which the template is a member. **Template Class** identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class. Use the **Translatable** check box to indicate that the template contains text strings which require translation.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Form Icon Use **Popup Icon** to specify an icon to display to the right of a form field for items of type POPUP. By default, the HTML DB engine uses a `list.gif` image. Use **Popup Icon Attr** to defines image attributes (such as height and width) for the Popup Icon

Search Field Use these attributes to specify how a Search field displays. Table 9–8 describes available Search Field attributes.

Table 9–8 Search Field Attributes

Attribute	Description
Before Field Text	Defines text to display before the popup list of values search field displays.
Filter Width	Display the text field using this width.
Filter Max Width	Display the text field widget using this maximum width.

Table 9–8 (Cont.) Search Field Attributes

Attribute	Description
Filter Text Attribute	Display the text field using these attributes. This will be included within the HTML input tag.
After Field Text	Display this text after displaying the search field, the search button, and the close button.

Buttons Use these attributes to define the button name and attributes for Find, Close, Next, and Previous button.

Window Popup lists of values are executed using JavaScript. Use these attribute to control the values of `scrollbars=`, `resizable=`, `width=`, and `height=`. For more information on default values see online help.

Pagination Defines how row count results display.

Result Set Use these attributes to define text or HTML to display before and after a result set.

Page Attributes Use these attributes to define popup pages. For more information see online help.

Region Templates

Region templates control the appearance and placement of region attributes. Region templates frequently use HTML tables to arrange content.

Region templates apply style elements to regions. Region templates display substitution strings. The only required substitution string, `#BODY#`, identifies where the source of the region should be placed. All other substitution strings are optional. You can use these substitution strings to indicate the existence and placement of a page control (such as a button) within the region.

Region Template Attributes

This section describes specific sections of the Region Template page.

Region Template Identification **Name** identifies the name that identifies the template. Use the **Translatable** check box to indicate that the template contains text strings which require translation. **Theme** indicates the theme to which the template is a member.

Template Class identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use **Template Subscription** to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Region Template `##BODY#` is the only required substitution string. It identifies where the source of the region should be placed. All other substitution strings are optional. The following are valid substitution strings:

- #TITLE#
- #EXPAND#
- #CHANGE#
- #BODY#
- #FORM_OPEN#
- #FORM_CLOSE#

When you create a button in a region position, the positions you have defined will appear in a select list. Use the following substitution strings to define positions for placement of buttons in a region:

- #EDIT#
- #CLOSE#
- #CREATE#
- #EXPAND#
- #HELP#
- #DELETE#
- #COPY#
- #NEXT#
- #PREVIOUS#

See Also: "Using Substitution Strings" on page 6-16

Form Table Attributes Page items display within regions. Items are rendered as HTML form elements in an HTML table. With this template property, you can define attributes that will be placed in the <table> tag. For example:

```
class="instructiontext"
```

Comments Use this attribute to record developer comments.

Report Templates

Report column templates provide you with control over the results of a row from a SQL query. This type of template defines a cell not an entire row

Each report template identifies column names using the syntax #1#, #2#, #3# and so on. You can also name columns using column name substitution syntax such as #ENAME# or #EMPNO#. You can reference any item from your application within your template. For example, to reference an item called *ABC.* in your template, you could include the exact substitution string *&ABC.*. The actual value of ABC. would be provided by an end user editing an item in your application named *ABC*.

Topics in this section include:

- About Generic Column Templates and Named Column Templates
- Report Column Template Attributes for Generic Column Templates
- Report Column Template Attributes for Named Column Templates
- About Using JavaScript in Column Templates

About Generic Column Templates and Named Column Templates

Oracle HTML DB includes two types of report templates:

- generic column templates
- named column templates

Generic Column Templates A generic column template determines the appearance of a report by defining the look of the column once. This look is then repeated as many times as is necessary based on the number of columns specified in the report's definition. This type of templates is limited to reports that have a standard row and column structure. Additional style may be applied to a report using this type of template through use of conditions.

The following example demonstrates how to have each column use a specific style:

```
<td class="tabledata" align="#ALIGN#">#COLUMN_VALUE#</td>
```

This example assumes your page template includes a CSS containing the class `tabledata`. This example also demonstrates the use the substitution strings `#ALIGN#` and `#COLUMN_VALUE#`. If you actually ran this report, these substitution strings would be replaced with values generated by the results of a SQL query.

If your query uses an expression in the select list, it is a good idea to create an alias for the columns to avoid runtime errors. For example, suppose your query was as follows:

```
SELECT ename, (sal + comm) * 12 FROM emp
```

You could rewrite the query to alias the columns as follows:

```
SELECT ename, (sal + comm) * 12 yearly_comp FROM emp
```

Named Column Templates Named column templates allow for more flexibility in report design. However, because they reference columns by name, they can only be used by reports that are based on those columns. For example:

```
<tr><td>#ENAME#</td><td>#SAL#</td></tr>
```

Although named column templates offer a great deal of flexibility, you may need to create a new template for each query. You can also include a position notation. The following example demonstrates how to use following HTML and substitution strings:

```
<tr><td>#ENAME#</td><td>#SAL#</td></tr>

<tr><td>#1#</td><td>#2#</td></tr>
```

Report Column Template Attributes for Generic Column Templates

This section describes specific sections of the Report Column Template page for Generic Column Templates.

Report Template Identification **Template Name** identifies the name of the template. Use the **Translatable** check box to indicate the template contains text strings which require translation. **Template Type** indicates the type of template. Named Column templates reference column names in the template. Generic Column Templates reference `#COLUMN_VALUE#` in the template.

Theme indicates the theme to which the template is a member. **Template Class** identifies a specific usage for the template. When you switch to a new theme, all

templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Template Subscription Use Template Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Before Rows In **Before Rows** enter HTML that displays once at the beginning of a report template. Opening an HTML table is a common use of this attribute as shown in the following example:

```
<table>
```

You can identify column headers using the syntax #1#, #2#, #3#. For example:

```
<th>#1#</th><th>#2#</th><th>#3#</th>
```

You can include pagination above a report by including the substitution string #TOP_PAGINATION#. This substitution string generates HTML which starts with an opening <tr> tag and ends with a closing </tr> tag. For example, to include an open table tag and #TOP_PAGINATION# substitution string you would enter the following:

```
<table>#TOP_PAGINATION#
```

You can also include the substitution string #CSV_LINK# to include support for exporting your report to CSV format, a format compatible with most spreadsheet programs.

Column Headings Use **Column Heading Template** to colorize each column header cell. Note that the text of this attribute must indicate where the cell heading text will be colorized. For example:

```
 <th #ALIGNMENT#>#COLUMN_HEADER#</th>
```

If you do not want any column headings, enter the following:

```
OMIT
```

If you do use this attribute, HTML DB engine applies the default column heading template.

Before Each Row In **Before Each Row** enter text to display before all columns in the report. Use this attribute to open a new HTML row. Before Each Row supports the following substitution strings:

- #ROWNUM#

  Use this substitution strings to specify the current row

- #COLCOUNT#

  Use this substitution strings to specify the number of columns

- #HIGHLIGHT_ROW#

  Use this substitution strings to specify the number of highlighted rows.

Column Templates Column templates define the look of each column. You can define up to four column templates, each of which can conditional. For example, you can have

different background colors for even and odd rows, or highlight rows which meet a PL/SQL defined condition.

In each Column Template, you define the look of each column. Column Templates support the substitution strings described in Table 9–9.

Table 9–9 Column Template Substitution Strings

Substitution String	Description
#ALIGNMENT#	Determines the column alignment. Specified by the user.
#COLCOUNT#	Count of the number of columns.
#COLNUM#	Defines the current column number.
#COLUMN_HEADER#	Defines the column header.
#COLUMN_VALUE#	Replaced with the value of the column.
#ROWNUM#	Specifies the current row number.

Consider the following example:

```
<td #ALIGNMENT#>#COLUMN_VALUE#</td>
```

If you actually ran this report, these substitution strings would be replaced with values generated by the results of a SQL query.

By creating conditions, you can create a report that displays columns differently depending on whether the specified condition is met. To specify a column template be used conditionally, select a condition type from the Column Template Condition list. Valid values include:

- **Use Based on PL/SQL Expression.** Conditionally format columns based on data in that row.
- **Use for Even Numbered Rows.** Conditionally format even numbered rows.
- **Use for Odd Numbered Rows.** Conditionally format odd numbered row.

If you select **Use Based on PL/SQL Expression**, the next step is to enter a PL/SQL expression in Column Template Expression field. For example, the following expression displays a value in bold if the value is greater than 2000:

```
#SAL# > 2000
```

Note that you could also use the substitution string #ROWNUM#. For example:

```
#ROWNUM# > 2000
```

After Each Row In **After Each Row** enter HTML that displays after all columns in the report display. This attribute is often used to close an HTML table row. For example:

```
</tr>
```

After Rows Use **After Rows** to specify text that should display after the last row. A common use of this attribute is to close the HTML table tag. For example:

```
</table>
```

The After Rows attribute supports the following substitution strings:

- #PAGINATION#

Replaced with a pagination attribute.

- #COLCOUNT#

  Substituted at runtime with the number of columns defined in the report.

Row Highlighting Use **Background color for checked row** to control the background color of a report row when the row selector is checked. Use **Background color for current row** to control the background color of a report row when the user moves the mouse over the row.

Pagination Subtemplate The Pagination Subtemplate section contains attributes for editing the Pagination Template, Next Page Template, Previous Page Template, Next Set Template, and Previous Template. Pagination Subtemplates support the substitution strings #PAGINATION_NEXT#, #PAGINATION_NEXT_SET#, #PAGINATION_PREVIOUS# and #PAGINATION_PREVIOUS_SET#. Table 9–12 describes these templates.

Table 9–10 Pagination Subtemplate Attribute

Pagination Subtemplate Attribute	Description
Pagination Template	Applies to the entire pagination subtemplate. For example: `<span class="instructiontext">#TEXT#</span>` You can use the substitution string #TEXT# to specify where you want the pagination subtemplate to display. Use the other Pagination Subtemplate attributes to modify individual items.
Next Page Template	Enter HTML to modify how the Next Page portion of the pagination subtemplate appears. For example: `<a href="#LINK#">next</a>`
Previous Page Template	Enter HTML to modify how the Previous Page portion of the pagination subtemplate appears. For example: `<a href="#LINK#">previous</a>`
Next Set Template	Enter HTML to modify how the Next Set portion of the pagination subtemplate appears. For example: `<a href="#LINK#">next set</a>`
Previous Set Template	Enter HTML to modify how the Previous Set portion of the pagination subtemplate appears. For example: `<a href="#LINK#">previous set</a>`

Comments Use this attribute to record developer comments.

Report Column Template Attributes for Named Column Templates

This section describes specific sections of the Report Column Template page for Named Column Templates.

Identification **Template Name** identifies the name of the template. Use the **Translatable** check box to indicate the template contains text strings which require translation.

Template Type indicates the type of template. Named Column templates reference column names in the template. Generic Column Templates reference #COLUMN_VALUE# in the template.

Theme indicates the theme to which the template is a member. **Template Class** identifies a specific usage for the template. When you switch to a new theme, all templates in one theme are mapped to corresponding templates in another theme. Application Builder accomplishes this template mapping through the assignment of a template class.

Subscription Use Subscription to apply an existing template to the current application. When you select an existing template, you become a subscriber to that template.

To load a new copy of a master template, click **Refresh Template**.

Row Templates Row templates define the look of each column. You can define up to four row templates, each of which can conditional.

In each Row Template, you define the look of each row. Row Templates support the substitution strings described in Table 9–11.

Table 9–11 Row Template Substitution Strings

Substitution String	Description
#ALIGNMENT#	Determines the row alignment. Specified by the user.
#COLCOUNT#	Count of the number of columns.
#COLNUM#	Defines the current column number.
#COLUMN_HEADER#	Defines the column header.
#COLUMN_VALUE#	Replaced with the value of the column.
#ROWNUM#	Specifies the current row number.

By creating conditions, you can create a report that displays rows differently depending on whether the specified condition is met. To specify a row template be used conditionally, select a condition type from the Column Template Condition list. Valid values include:

- **Use Based on PL/SQL Expression.** Conditionally format columns based on data in that row.
- **Use for Even Numbered Rows.** Conditionally format even numbered rows.
- **Use for Odd Numbered Rows.** Conditionally format odd numbered row.

If you select **Use Based on PL/SQL Expression**, the next step is to enter a PL/SQL expression in Column Template Expression field. For example, the following expression displays a value in bold if the value is greater than 2000:

```
#SAL# > 2000
```

Note that you could also use the substitution string #ROWNUM#. For example:

```
#ROWNUM# > 2000
```

Column Headings Use this template to colorize each column header cell. The text of this attribute must include help to indicate where the cell heading text should be colorized. If you do not enter a Column Heading Template, then a default column header template is applied. If you do not want any column headings, then enter OMIT.

For example:

```
<th #ALIGNMENT#>#COLUMN_HEADER#</th>
```

Before first and after last row text In **Before Rows** enter HTML that displays once at the beginning of a report template. Opening an HTML table is a common use of this attribute as shown in the following example:

```
<table>
```

You can identify column headers using the syntax #1#, #2#, #3#. For example:

```
<th>#1#</th><th>#2#</th><th>#3#</th>
```

You can include pagination above a report by including the substitution string #TOP_PAGINATION#. This substitution string generates HTML which starts with an opening <tr> tag and ends with a closing </tr> tag. For example, to include an open table tag and #TOP_PAGINATION# substitution string you would enter the following:

```
<table>#TOP_PAGINATION#
```

You can also include the substitution string #CSV_LINK# to include support for exporting your report to CSV format, a format compatible with most spreadsheet programs.

Use **After Rows** to specify text that should display after the last row. A common use of this attribute is to close the HTML table tag. For example:

```
</table>
```

The After Rows attribute supports the following substitution strings:

- #PAGINATION#

  Replaced with a pagination attribute.

- #COLCOUNT#

  Substituted at runtime with the number of columns defined in the report.

Row Highlighting Use **Background color for checked row** to control the background color of a report row when the row selector is checked. Use **Background color for current row** to control the background color of a report row when the user moves the mouse over the row.

Pagination The Pagination section contains attributes for editing the Pagination Template, Next Page Template, Previous Page Template, Next Set Template, and Previous Template. Pagination Subtemplates support the substitution strings #PAGINATION_NEXT#, #PAGINATION_NEXT_SET#, #PAGINATION_PREVIOUS# and #PAGINATION_PREVIOUS_SET#. Table 9–12 describes these templates.

Table 9–12 Pagination Subtemplate Attribute

Pagination Subtemplate Attribute	Description
Pagination Template	Applies to the entire pagination subtemplate. For example: `<span class="instructiontext">#TEXT#</span>` You can use the substitution string #TEXT# to specify where you want the pagination subtemplate to display. Use the other Pagination Subtemplate attributes to modify individual items.
Next Page Template	Enter HTML to modify how the Next Page portion of the pagination subtemplate appears. For example: `<a href="#LINK#">next</a>`
Previous Page Template	Enter HTML to modify how the Previous Page portion of the pagination subtemplate appears. For example: `<a href="#LINK#">previous</a>`
Next Set Template	Enter HTML to modify how the Next Set portion of the pagination subtemplate appears. For example: `<a href="#LINK#">next set</a>`
Previous Set Template	Enter HTML to modify how the Previous Set portion of the pagination subtemplate appears. For example: `<a href="#LINK#">previous set</a>`

Comments Use this attribute to record developer comments.

About Using JavaScript in Column Templates

You can conditionally display HTML depending upon values in the database using JavaScript. The following example displays an HTML row only if the GROUP_DESC query column is not null.

```
<script language="javascript">
IF ( "#GROUP_DESC#" != "" )
document.writeln( "<TR>;
<TD BGCOLOR=#336699>;</TD>
</TR>
</TR>
<TD>#GROUP_DESC#</TD>
          </TR>" );
</TR>" );
```

See Also:

- Online help for more information on using specific sections of the Edit Report Template page
- "Customizing Regions" on page 9-3

Optimizing a Page for Printing

You can optimize a page for printing by creating a specific Print Mode template and specifying that template in the User Template Defaults section of the Edit Application Attributes page. Generally, a Print Mode template optimizes a page for printing. For example, this template might:

- Not display tabs or navigation bars
- Have items display as text instead of form elements

If the theme you select does not include a printer friendly template, you can create a Print Mode template by creating a new page template.

See Also: "Creating a New Template" on page 9-18

Setting a Print Mode Template for an Application

You enable your Print Mode template by selecting it in Default Templates by Component section of Define Theme page.

To enable Print Mode mode:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **Themes and Templates**.

   The Themes page appears.
5. In the Themes list, click the **Edit** icon adjacent to the theme name.
6. Make a new selection from Printer Friendly Page.
7. Click **Apply Changes**.

See Also: "Changing Default Templates in a Theme" on page 9-9

Using f?p Syntax to Toggle to Print Mode

Once you create a Print Mode template and select it as an application attribute, you can use `f?p` syntax to toggle to Print Mode. Including the ninth `f?p` syntax argument (`PrinterFriendly`) renders the page in printer friendly mode (optimize printed output). For example, you could include this argument when coding a link, or creating navigation bar icon.

See Also: "Using f?p Syntax to Link Pages" on page 6-14

Using Custom Cascading Style Sheets

A cascading style sheet (CSS) provides a way to control the style of a Web page without changing its structure. When used properly, a CSS separates visual attributes such as color, margins, and fonts from the structure of the HTML document. Oracle HTML DB includes themes that contain templates that reference their own CSS. The style rules defined in each CSS for a particular theme also determine the way reports and regions display.

Topics in this section include:

- Uploading Cascading Style Sheets

Uploading Cascading Style Sheets

You upload cascading style sheets to your workspace using the Cascading Style Sheet (CSS) Repository. Uploaded cascading style sheets (CSS) are available to any application created in your workspace. The cascading style sheets are written to the file system, so you can reference them in your HTML source code.

To upload cascading style sheets to your workspace:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application name.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Cascading Style Sheets**.

   The CSS Repository appears.
5. To upload a new CSS, click **Create** and follow the on-screen instructions.
6. To edit an existing CSS, click the **Edit** icon in the CSS Repository.

Referencing an Uploaded Cascading Style Sheet in the Page Template

You can reference an uploaded cascading style sheets within the Header section of the page template. You use the Header section to enter the HTML that makes up the <HEAD> section of the HTML document.

To reference an uploaded cascading style sheets:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **Themes and Templates**.

   The Themes page appears.
5. From the Tasks list, select View Templates.
6. Select the name of the page template you wish to edit.
7. The Header, use a `<link>` tag within the head section to reference the appropriate style sheet.

   To reference an uploaded file that is associated with a specific application, you would use the substitution string `#APP_IMAGES#`. For example:

   ```
   <html>
   <head>
       <title>#TITLE#</title>
       #HEAD#
       <link rel="stylesheet" href="#APP_IMAGES#sample2.css" type="text/css">
   </head>
   ...
   ```

   To reference an uploaded file that is associated with a specific workspace, you would use the substitution string `#WORKSPACE_IMAGES#`. For example:

   ```
   <html>
   <head>
   ```

```
    <title>#TITLE#</title>
    #HEAD#
    <link rel="stylesheet" href="#WORKSPACE_IMAGES#sample3.css"
type="text/css">
</head>
...
```

See Also: "Creating a New Template" on page 9-18, "Editing Templates" on page 9-19, "Page Templates" on page 9-24, "APP_IMAGES" on page 6-18, and "WORKSPACE_IMAGES" on page 6-26

Uploading Images

You can upload images to your workspace using the Image Repository.

To upload images to your workspace:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application name.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Images**.

   The Image Repository appears.
5. To upload a new image, click **Create**.
6. Follow the on-screen instructions.

Referencing Images

When you install Oracle HTML DB, the installer creates a virtual directory for images. This virtual directory points to the actual path on the file system that contains uploaded images. By default, you reference this virtual directory using the prefix:

```
/i/
```

When you create an application, you need to verify this prefix on the Edit Application Attributes page:

To verify the Image Prefix for an application:

1. Navigate to the Workspace home page.
2. From the Applications list, select the application name.
3. On the Application Builder home page, click **Edit Attributes**.
4. When the Edit Application Attributes page appears, locate the Image Prefix field.

   By default, this attribute is defined as `/i/`. Contact your administrator for more information on the name of this virtual directory for your installation.

When you embed an image in static text (for example, in page or region headers or footers) you can reference the image using the substitution string `#IMAGE_PREFIX#`. For example, to reference the image `go.gif` you would use the following syntax:

```
<img src="#IMAGE_PREFIX#go.gif">
```

Alternatively, you can also reference an image using a fully qualified URL. For example:

```
<img
src="http://g-images.amazon.com/images/G/01/associates/navbar2000/logo-no-border(1
).gif">
```

> **See Also:** "Built-in Substitution Strings" on page 6-16, "IMAGE_PREFIX" on page 6-22, "APP_IMAGES" on page 6-18, and "WORKSPACE_IMAGES" on page 6-26

Uploading Static Files

You can upload static files to your workspace using the Static File Repository.

To upload static files to your workspace:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application name.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Files, select **Static Files**.

   The Static Files Repository appears.
5. To upload a file, click **Create**.
6. Follow the on-screen instructions.

Creating a Multiple Column Layout

A region is an area of a page that uses a specific template to format HTML content. You use regions to group page controls. To create a multiple column layout, you create two regions that display in adjacent cells of the same table.

You can create a multiple column layout by either:

- Manually creating the two adjacent regions
- Defining a page template that contains a multiple column table

Creating Regions in Multiple Columns

You create new regions using the Create Region Wizard. To create a two column page, you create two regions. Oracle HTML DB replaces `#BOX_BODY#` within a two column table and displays the regions in two separate cells.

To create a two column page by creating regions:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. From the Pages list, select a page.

   The Page Definition appears.
4. Create the first region:
   - Under Regions, click **Create**.

     The Create Region Wizard appears.
   - Select a region type.

- From the Column field, select **1**.
- Follow the on-screen instructions.

5. Create the second region:
   - Under Regions, click **Create**.

     The Create Region Wizard appears.
   - Select a region type.
   - From the Column field, select **2**.
   - Follow the on-screen instructions.

Creating a Multiple Column Page Template

Page templates define the appearance of individual pages, including the placement of page controls and components. Each page template is divided into three sections: Header, Body, and Footer. The most basic template must include the substitution string #BOX_BODY# in the Body attribute. When the page is rendered, the HTML DB engine replaces #BOX_BODY# with HTML to display the regions on that page.

You can create a multiple column page by defining a page template that contains a multiple column table. You then explicitly place regions within specific table cells.

The following example demonstrates how to create a two column page and specify a region position using the #REGION_POSITION_XX# substitution string in each column. You would enter this code in the Body section of the page level template.

```
<body #ONLOAD#>
  #FORM_OPEN#
  <table style="width:100%">
    <tr>
      <td style="width:50%;padding:5px;">#REGION_POSITION_01#</td>
      <td style="width:50%; border-left:2px #bbbbbb dashed; padding:5px;">#REGION_
POSITION_02#</td>
    </tr>
  <br />
  #BOX_BODY#
  #FORM_CLOSE#
</body>
```

Once you create this page level template, the newly defined positions would be available as Display Point options when you run the Create Region Wizard.

Rendering HTML Using Custom PL/SQL

If you need to generate specific HTML content not handled by Oracle HTML DB forms, reports, and charts, you can use the region type PL/SQL. To generate HTML in this type of region, you need to use the PL/SQL Web Toolkit. You can reference session state using bind variable syntax. Keep in mind that when you generate HTML in this way you do not get the same consistency and control provided with templates.

See Also:

- *Oracle Database Application Developer's Guide - Fundamentals* for more information on developing Web applications with PL/SQL
- *PL/SQL Packages and Types Reference* for more information on htp packages

To give you more control over HTML dynamically generated within a region, you can use PL/SQL. For example, to print the current date you could create a region with the following source:

```
htp.p(TO_CHAR(SYSDATE,'Day Month DD, YYYY'));
```

This next example accesses tables:

```
DECLARE
   l_max_sal NUMBER;
BEGIN
   SELECT max(sal) INTO l_max_sal FROM emp;
   htp.p('The maximum salary is: '||TO_CHAR(l_max_sal,'999,999.00'));
END;
```

10

Adding Navigation

When you build an application you can include a number of different types of navigation controls, including navigation bars, tabs, menus, lists, and trees. This section describes how to implement navigation in your application.

Navigation controls are shared components. Once you create them, you can add them to any page within your application. You add a specific type of navigation control at the application level on the Shared Components page.

This section contains the following topics:

- Creating a Navigation Bar
- Creating Tabs
- Controlling Flow Using Branches
- Creating Menus
- Creating Lists
- Creating Trees

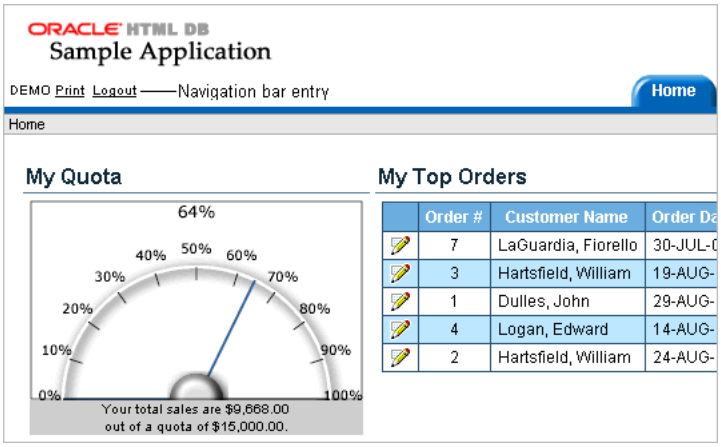
See Also:

- "Working with Shared Components" on page 7-26
- "Viewing a Page" on page 7-11
- "Controlling Page Layout and User Interface" on page 9-1

Creating a Navigation Bar

Navigation bars (see Figure 10–1) offer an easy way for users to move between pages in an application. The location of a navigation bar depends upon the associated page template. A navigation bar icon enables you to display a link from an image or text. A navigation bar entry can be an image, an image with text beneath it, or text. You must supply navigation bar entry images and text. When you create a navigation bar entry, you can specify an image, text, a display sequence, or a URL.

Figure 10–1 Navigation Bar Entries



Navigation bars are different from other shared components in that you not need to reference them on a page by page basis. If your page template includes the `#NAVIGATION_BAR#` substitution string, the HTML DB engine automatically includes any defined navigation bars when it renders the page.

> **See Also:** "Supported Page Template Substitution Strings" on page 9-24 on using the `#NAVIGATION_BAR#` substitution string

Creating a Navigation Bar Entry

Before you can add a navigation bar, you must create a navigation bar entry on the Navigation Bar page. You can access the Navigation Bar page from either the Page Definition or from the Shared Components page.

> **See Also:** "Working with Shared Components" on page 7-26

To create a navigation bar entry referencing an icon:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)

   The Page Definition appears.
2. Under Shared Components, scroll down to **Navigation Bar**.
3. Under Navigation Bar, click **Create**.

   The Create NavBar Entry Wizard appears.
4. Specify the following NavBar entry attributes:
   - Sequence
   - Alt Tag Text
   - Icon Image Name
   - Image Height and Image Width
   - Text

   Specify the target location.
5. If the target location is a URL:

a. From Target is a, select **URL**.

b. In URL Target, type a URL.

6. If the target location is a page:

a. From Target is a, select **Page in this application**.

b. In Page, specify the page number.

7. If the navigation bar entry will display conditionally, specify the appropriate conditional information and click **Create**.

To create a navigation bar entry without icons:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)

   The Page Definition appears.

2. Under Shared Components, scroll down to **Navigation Bar**.

3. Under Navigation Bar, click **Create**.

   The Create NavBar Entry Wizard appears.

4. Specify the following icon attributes:

   - Sequence
   - Text

5. If the target location is a URL:

a. From Target is a, select **URL**.

b. In URL Target, enter an URL. For example:

```
http://www.yahoo.com
```

6. If the target location is a page:

a. From Target is a, select **Page in this application**

b. In Page, specify the page number

7. If the navigation bar entry will display conditionally, specify the appropriate conditional information and click **Create**.

Managing Navigation Bar Entries

To manage navigation bar entries:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)

2. Select the heading **Navigation Bar**.

   The Navigation Bar page appears.

3. On the Navigation Bar page you can:

   - View details about a specific entry by clicking the **Edit** icon
   - Access an editable summary view, by clicking **Edit selected attributes**.
   - View navigation bar entries that are subscribed to by clicking **Subscriptions**.
   - View a history of recently changed navigation bar entries by clicking **History**.

- Create a new navigation bar entry, by clicking **Create**

Creating Tabs

Tabs are an effective way to navigate users between pages of an application. You can create a tabbed application look by using parent tabs, standard tabs, and Oracle HTML DB lists.

Application Builder includes two different types of tabs:

- Standard tabs
- Parent tabs

An application having only one level of tabs uses a standard tab set. A standard tab set is associated with a specific page and page number. You can use standard tabs to link users to a specific page. A parent tab set functions as a container to hold a group of standard tabs. Parent tabs give users another level of navigation as well as a context (or sense of place) within the application. You can use parent tabs link users to a specific URL associated with a specific page.

Topics in this section include:

- About Template Support
- Using Tab Manager
- Editing Multiple Tabs at Once
- Accessing Tab Reports

Note: When running the Create Application Wizard, you have the option of creating an application with tabs. The following procedures assume you have already created an application that does not have any tabs.

See Also: "Creating an Application" on page 8-1

About Template Support

Before you can create parent and standard tabs, you need to check that your default template has positions defined for both standard and parent tabs using the appropriate substitution strings. You also need to make sure you do not override this template at the page level.

See Also:

- "Application Template Defaults" on page 7-10 for more information on setting a default page template at the application level
- "Primary Display Attributes" on page 7-17 for more information on setting a template at the page level

Using Tab Manager

You manage tab information using Tab Manager. You can access Tab Manager from the Shared Components page, or by clicking either the Parent Tabs or Standard Tabs heading on the Page Definition.

Accessing Tab Manager

To access Tab Manager from the Shared Components page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, click **Tabs**.

   Tab Manager appears displaying a graphical representation of the tabs defined in your application.
5. To make another tab current, click the tab.

   Notice the two Add buttons. Use the Add button on the upper right side of the graphic to add Parent tabs. Use the Add button at the lower left side of the graphic to add standard tabs.
6. To add a new tab, click **Add** adjacent to the appropriate tab type.

   Think of parent tabs as a container to hold standard tabs. For example, in order to add two levels of tabs you first create a parent tab and then add standard tabs to it.

To access Tab Manager from the Page Definition:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)

   The Page Definition appears.
2. Under Shared Components, select one of the following headings:

   - Parent Tabs
   - Standard Tabs

   Tab Manager appears displaying a graphical representation of the tabs defined in your application. The currently selected standard or parent tab is highlighted.
3. To make another tab current, click the tab.

   Notice the two Add buttons. Use the Add button on the upper right side of the graphic to add Parent tabs. Use the Add button at the lower left side of the graphic to add standard tabs.
4. To add a new tab, click **Add** adjacent to the appropriate tab type.

   Think of parent tabs as a container to hold standard tabs. For example, in order to add two levels of tabs you first create a parent tab and then add standard tabs to it.

Creating a New Tab from the Page Definition

To create a new tab from the Page Definition:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Tabs, click the Create icon.

   Create Tab Wizard appears.
3. Follow the on-screen instructions.

Editing Multiple Tabs at Once

You can edit multiple tabs at once by clicking **Edit all standard tabs** and **Edit all parent tabs** on the Tab Manager page.

To edit multiple tabs at once:

1. Navigate to Tab Manager. (See "Using Tab Manager" on page 10-4.)
2. Click one of the following to edit tab attributes in a summary view:
   - **Edit all standard tabs**
   - **Edit all parent tabs**

Accessing Tab Reports

Application Builder includes the following tab reports

- Standard Tab Utilization
- Standard and Parent Tab History

You can access these reports by clicking the appropriate button at the top of the Tab Manager page.

Standard Tab Utilization

Click **Utilization** to access the Standard Tab Utilization report. This report lists the standard tabs used in the current application.

Standard and Parent Tab History

Click History to view the Standard Tab History and Parent Tab History reports. These reports display a history of changes to tab attributes for the current application.

Controlling Flow Using Branches

A branch is an instruction to link to a specific page, procedure, or URL. For example you can branch from page 1 to page 2 after page 1 is submitted.

Note: If a page has a select list and a submit button, it can submit itself. However, you must create a branch to call the page or the submit will fail.

You create a new branch by running the Create Branch Wizard and specifying Branch Point and Branch Type. The Branch Type defines the type of branch you are creating.

To create a branch:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Branching, click the **Create** icon to run the Create Branch Wizard.
3. Select a Branch Point:
   - **On Submit: Before Computation** - Occurs before computations, validations, or processing. Use this option for a Cancel button.
   - **On Submit: Before Validation** - Occurs after computations, but before validations or processing. Typically not used. If a validation fails, page

processing stops, a rollback is issued, and the page displays the error. Because of this default behavior, you do not need to create branches to accommodate validations. However, you might want to branch based on the result of a computation (for example, to the previous branch point).

- **On Submit: Before Processing** - Occurs after computations and validations, but before processing. Use this option to branch based on a validated session state, but before performing any page processing.
- **On Submit: After Processing** - Occurs after computations, validations, and processing. This option branches to a URL or page after performing computations, validations, and processing. When using this option, remember to sequence your branches if you have multiple branches for a given branch point.
- **On Load: Before Header** - Occurs before a page rendered. This option displays another page instead of the current page or redirects the user to another URL or procedure.

4. Select a Branch Type.

   Depending upon the Branch Type, specify the following types of information on the pages that follows:

   - A page number of the page you wish to branch to
   - PL/SQL code
   - A URL address

5. Follow the on-screen instructions.

Creating Menus

Menus provide users with hierarchical navigation. A menu is a hierarchical list of links that display using templates. You can display menus as a list of links, or as a breadcrumb path.

Topics in this section include:

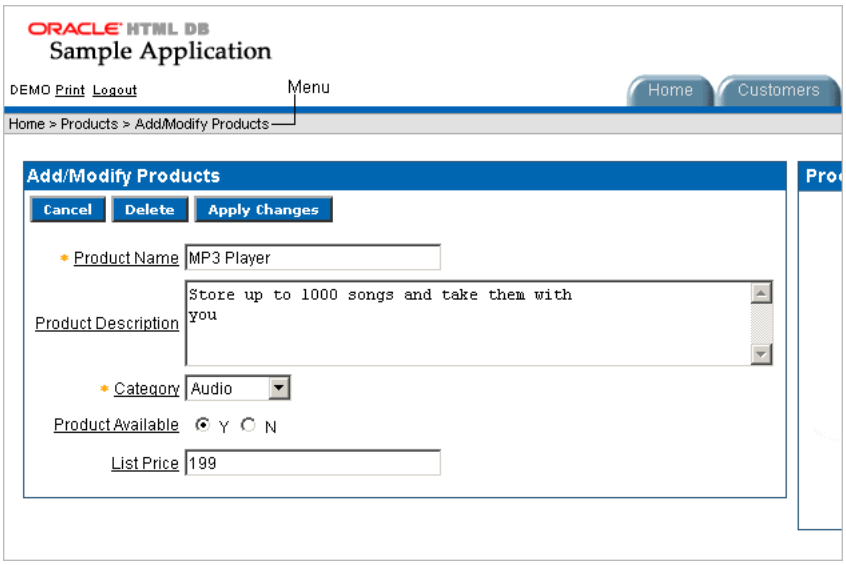
- About Breadcrumb Menus
- Creating a Menu
- Editing Multiple Menu Names Simultaneously
- Accessing Menu Reports

> **See Also:** "Creating a New Template" on page 9-18 and "Menu Templates" on page 9-22 for more information on changing menu display

About Breadcrumb Menus

As shown in Figure 10–2, a breadcrumb style menu indicates where the user is within the application from a hierarchical perspective. In addition, users can click a specific page to instantly view it. You can include a breadcrumb menu that functions as second level of navigation and displays beneath the standard tabs at the top of each page.

Figure 10–2 Breadcrumb Style Menu



Creating a Menu

To add a menu to a page in your application you must:

- Create the menu by running the Create Menu Wizard.
- Add options to it
- Add the menu to a page by creating a menu region

To create a menu from the Shared Components page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, click **Menus**.

   The Menus page appears.
5. To create a new menu, click **Create**.
6. Follow the on-screen instructions.

To create a menu from a Page Definition:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Shared Components, scroll down to **Menu** and click **Create**.

   The Menus page appears.
3. To create a new menu, click **Create**.
4. Follow the on-screen instructions.

Once your menu has been created, you need to add options to it.

Adding Options to a Menu

To add options to a menu:

1. Navigate to the Menus page.
2. Select a menu and click **Create Menu Option**.
3. Under Menu Identification, specify the page on which this menu will be current
4. Under Menu Option, specify the following:
   - **Sequence** - Indicate the order in which menu options appear.
   - **Parent Menu Option** - Identify the parent of this menu entry.
   - **Short Name** - Specify the short name of this menu option (referenced in the menu template).
   - **Long Name** - Specify the long name of this menu option (referenced in the menu template).
5. Specify a target location.

   If the target location is a URL:

   a. From Target is a, select **URL**.

   b. In URL Target, type a URL.

   If the target location is a page:

   a. From Target is a, select **Page in this Application**

   b. In Page, specify the page number
6. To make the menu conditional:

   a. Make a selection from the Condition Type list.

   b. Enter an expression in the fields provided.
7. When you are finished defining menu attributes, click **Create** at the top of the page.

Repeat these procedures for each menu option you need to create.

Adding a Menu to a Page

Once you create a menu and a menu template, the next step is to add it a page by creating a region and specifying the region type as Menu.

> **See Also:** "Creating a New Template" on page 9-18 and "Menu Templates" on page 9-22 for more information on changing menu display

To add a menu to a page:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Regions, click **Create**.

   The Create Region Wizard appears.
3. Under Regions, click **Create**.
4. While running the Create Region Wizard:
   - Select **Menu** for the region type.
   - Enter a title.

- Select a menu and menu template.

5. Click **Create Menu Region**.

Repeat these procedures for each page on which you would like to add a menu.

About Creating a Dynamic Menu

To give users a more exact context, you may include session state in a menu, making your menus dynamic. For example, suppose a page in your application displays a list of orders for a particular company and you want to include the following breadcrumb menu:

```
Home > Orders > Orders for ACME Inc
```

In this example, ACME Inc not only indicates the page a user is on, but also the navigation path. The HTML DB engine stores the value of ACME Inc. in session state.

To create this type of dynamic menu, you must include a reference to a session state item in the menu's short name or long name, for example:

```
&COMPANY_NAME.
```

Editing Multiple Menu Names Simultaneously

You can edit multiple menu entries at once by clicking **Edit multiple menu entries text** at the top of the Menus page.

To edit menu entries at once:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, click **Menus**.

   The Menus page appears.
5. Click **Edit multiple menu entries text**.
6. Edit the appropriate menu name and click **Apply Changes**.

Accessing Menu Reports

Application Builder includes the following menu reports:

- Menu Utilization Report
- Recent Menu Option Changes

You can access these reports by clicking the appropriate button at the top of the Menus page.

Menu Utilization Report

Click **Utilization** to access the Menu Utilization Report. This report lists menus by page. Click the page number to link to a specific page.

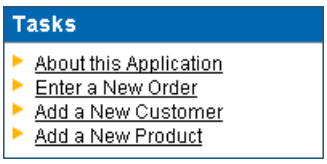
Recent Menu Option Changes

Click **History** to view the Recent Menu Option Changes report. This report lists recent changes to menus.

Creating Lists

As shown in Figure 10–3, a list is a shared collection of links. You control the appearance of a list through list templates. Each list element has a display condition which enables you to control when it displays. You can define a list element to be either current or non current for a specific page. You further specify what current looks like using template attributes. You add a list to a page by creating a region and specifying the region type as List.

Figure 10–3 List



Topics in this section include:

- Creating a List
- Adding a List to a Page
- Editing Multiple List Entries Simultaneously
- Accessing List Reports

> **See Also:** "Creating a New Template" on page 9-18 and "List Templates" on page 9-22 for more information on altering list display

Creating a List

To add a list to a page in your application you must:

- Create the list by running the Create Lists Wizard.
- Add items to the list.
- Add the list to a page by creating a List region.

To create a list from the Shared Components page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, click **Lists**.

   The Lists page appears.
5. To create a new list, click **Create List**.
6. Follow the on-screen instructions.

To create a list from a Page Definition:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Shared Components, scroll down to Lists and click **Create**.

   The Create / Edit List Object page appears.
3. To create a new list, click **Create**.

4. Follow the on-screen instructions.

Once your list has been created, you need to add items to it.

Adding Items to a List

To add items to a list:

1. Navigate to the Lists page.
2. Select a list
3. Click **Create List Entry** to add entries to it.

   The List Entry Attributes page appears.
4. Under Label and Sequence, edit the appropriate attributes. In List Entry Label (required), enter the label text for this link.

   Use Target to specify a target location.
5. If the target location is a page:
   a. From Target Type, select **Page in this Application**.
   b. In Page, specify the target page number.

      To reset pagination for this page, select **reset pagination for this page**.
   c. In Request, specify the request to be used.
   d. In Clear Cache, specify the pages (that is, the page numbers) on which to clear cache. Specify multiple pages by listing the page numbers in a comma delimited list.

      You can set session state (that is, give a listed item a value) using the next two attributes:
   e. To set session state:
      – In Set these items, enter a comma delimited list of item names for which you would like to set session state.
      – In With these values, enter a comma delimited list of values for the items specified in the previous step.

        You can specify static values, or substitution syntax (for example, `&APP_ITEM_NAME.`). Note that item values passed to `f?p=` in the URL may not contain a colon (:). Additionally, item values may not contain commas unless you enclose the entire value in backslash characters (for example, `\1234,56\`).

   If the target location is a URL:
   a. From Target type, select **URL**.
   b. In URL Target, type a URL.

   If the target location is a page:
   a. From Target type, select **Page in this Application**.
   b. In Page, specify the page number.
6. If the target location is a URL:
   a. From Target type, select **URL**.
   b. In URL Target, type a URL.

7. To make the list entry conditional:
   a. Make a selection from the Condition Type list.
   b. Enter an expression in the fields provided.
8. When you are finished defining list attributes, click **Create** or **Create and Create Another**.

Adding a List to a Page

Once you created a list, the next step is to add it a page by creating a region and specifying the region type as List.

> **See Also:** "Creating a New Template" on page 9-18"List Templates" on page 9-22 for more information on altering list display

To add a list to a page:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Regions, click **Create** to run the Create Region Wizard.
3. Select **List** as the region type.
4. Specify region attributes:
   - Enter a title
   - Select a region template
   - Specify a display point
   - Specify a sequence
5. From List, select the list you want to add.
6. Click **Create List Region**.

Repeat these procedures for each page on which you would like to add a list.

Editing Multiple List Entries Simultaneously

You can edit multiple list entries at once by clicking **Grid Edit** on the List Entries page.

To edit menu entries at once:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, click **Lists**.

   The Lists page appears.
5. Select a list name.

   The List Entries page appears.
6. Click **Grid Edit**.
7. Edit the appropriate items and click **Apply Changes**.

Accessing List Reports

Application Builder includes the following list reports:

- Menu Utilization Report
- Recent Menu Option Changes

You can access these reports by clicking the appropriate button at the top of the Lists page.

List Utilization

Click **Utilization** to access the Lists Utilization report. This report displays all lists included in the current application. To edit list entries, select the list name. To view the pages on which the list appears, click the number in the Pages column.

Unused lists

Click **Unused Lists** to identify lists that are not used in the current application.

List Definition and List Entry History

Click **History** to view changes to list definitions and list entries by developer and date.

Creating Trees

You can use a tree in your application to effectively communicate hierarchical or multiple level data.

To create a tree:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under Navigation, click **Trees**.

   The Trees page appears.
5. To create a new tree, click **Create**.
6. Enter basic page information.
7. Enter a Tree Name and specify the Default Expanded Levels.
8. Specify how tabs should be implemented.
9. Identify tree attributes:
   a. Enter a tree name.
   b. Under Start Tree, specify how the starting tree node is created:
      - **Based on new item with popup list of values** - Creates a tree based on a new item with a popup list of values (LOV). Requires an LOV query that selects, displays, and returns values from a tree table. Use this method to select a different starting point each time you visit a page.
      - **Based on a SQL Query** - Creates a tree based on a SQL query. Requires a SQL query that selects a primary key from a tree table.
      - **Static value** - Creates a tree based on a static value.

Keep in mind that the option **Based on new item with popup list of values** enables you to select a different starting point each time you visit a page. The starting point for the last two options always remains the same.

You build a tree based on a table that contains the node data. This base table must have an ID (a primary key) and a parent ID that functions as a foreign key of the table. These IDs determine the number of tree levels.

10. Select a tree template.
11. Follow the on-screen instructions and specify the owner and name of the table on which the tree will be based.
12. Under Identify Query:
    a. In ID, enter the column you want to use as the ID.
    b. In Parent ID, enter the Parent ID.
    c. In Leaf Node Text, specify the text that should appear on the leaf nodes.
    d. Under Link Option, select Existing Application Item to make the leaf node text a link.

Accessing Tree Reports

Application Builder includes the following tree reports:

- Tree Utilization
- Tree History

You can access these reports by clicking the appropriate button at the top of the Lists page.

Tree Utilization

Click **Utilization** to access the Tree Utilization report. This report displays all trees included in the current application. To edit a tree, select the tree name.

Tree History

Click **History** to view changes to trees by developer and date.

11

Debugging an Application

This section describes a number of approaches to debugging your application including viewing Debug Mode, enabling SQL tracing, viewing page reports, and how to manually remove a control or a component to isolate a problem.

This section contains the following topics:

About Tuning Performance

For applications having a large number of concurrent users, maintaining optimal performance is critical. To optimize your application's performance, remember to utilize the following Oracle HTML DB features:

- Use bind variables within your application whenever possible. You can reference session state values using bind variable syntax in SQL queries and application logic such as PL/SQL executed from processes and validations. Accessing session state using bind variables is the most efficient way to reference session state.
- Include a `#TIMING#` substitution string in the region footer so that you can view the timing of each region.

See Also:

- "About Bind Variables" on page 6-13
- "Using Substitution Strings" on page 6-16

Reviewing Session State

Many application are based on data contained within application controls. For example, buttons may display conditionally based on a value stored in session state. You can view current session state for your application by clicking the Session link on the Developer Toolbar.

See Also:

- "Using the Developer Toolbar" on page 7-14
- "Viewing Session State" on page 6-9
- "Managing Session State Values" on page 6-10
- "Managing Session State and User Preferences" on page 13-5

Accessing Debug Mode

Viewing a page in Debug Mode is effective way to track what the HTML DB engine is doing as it renders a page. You access Debug mode by clicking the **Debug** link in the Developer Toolbar.

See Also: "Using the Developer Toolbar" on page 7-14

Debug Mode displays time codes that correspond to specific HTML DB engine actions. This can be useful if you want to determine when the engine is setting session state. The bottom of the page displays an augmented version of the Page Definition. In addition to enabling you to link to page and component attributes, you can view additional details about item names and computation and processing points. To exit Debug mode, click **No Debug** in the Developer Toolbar.

You can also use `f?p` syntax run an application in Debug mode. Simply call the page and set the Debug argument to `YES`. For example:

```
f?p=100:1:&SESSION::YES
```

See Also: "Using f?p Syntax to Link Pages" on page 6-14

Enabling SQL Tracing and Using TKPROF

Tracing your session can be a very effective way to debug an application. From a database perspective, each page request is a single database session. If you enable SQL tracing, then Oracle HTML DB creates a temporary file you can then analyze using the TKPROF utility.

You enable SQL tracing in Oracle HTML DB by using `f?p` syntax to set the argument `p_trace=YES`. For example, to trace the display of page 1 in application 100 you would use the syntax:

```
http:/.../f?p=100:1&p_trace=YES
```

To use the TKPROF utility:

- Navigate to the directory in which the trace file is created.
- Type the following to view instructions about using TKPROF utility:

  ```
  tkprof help=yes
  ```

See Also: *Oracle Database Performance Tuning Guide* for more information on using the TKPROF program or contact your database administrator

Monitoring Application and Page Resource Use

Oracle HTML DB facilitates the monitoring of resources used by applications and pages by calling the package `DBMS_APPLICATION_INFO`. Whenever the HTML DB engine renders or processes a page, the module is set to `HTML DB` and includes the application ID and page number. Once set, you can query the `V$SESSION` and `V$SQLAREA` views to monitor transactions.

Viewing Oracle HTML DB Reports

When isolating an issue within a page, it is important to clearly understand the functions it is performing. To accomplish this goal, Oracle HTML DB includes a number of page and application reports.

To view page reports:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Click one of the following buttons at the top of the Page Definition:
   - **Event View** links to a report that details currently defined page controls and processes.
   - **Object References** displays a list of database objects referenced by the current page.
   - **History** displays a history of recently changed pages.

     See Also: "Event View" on page 7-14, "Object References" on page 7-14, and "History" on page 7-14

To view application reports:

1. Navigate to the Workspace home page.
2. Select an application name from the Applications list.
3. From the Tasks list, select **View Application Reports**.

   Application Reports page appears.
4. Select a report to review.

   See Also: "Viewing Application Reports" on page 7-29

Debugging Problematic SQL Queries

If your query does not seem to be running correctly, try running it in SQL Plus or in SQL Workshop. Either approach will test your query outside of the context of your application, making it easier to determine what the problem is.

Removing Controls and Components to Isolate a Problem

If you have problems running a page, try removing controls and components one at time. Using this approach, you can quickly determine which control or component may be the source of your problem. You can disable a control or component by selecting the conditional display attribute NEVER.

To remove a control or component using conditional attributes:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Select the name of the control or component you wish to disable.

   The appropriate attributes page appears.
3. Scroll down to the Conditional Display attribute and select **NEVER** from the Condition Type list.
4. Click **Apply Changes** and return to the Page Definition.
5. Try running the page again.
6. Continue to remove controls or components until the page runs correctly.

See Also:

- "Viewing a Page" on page 7-11
- "Editing Page Attributes" on page 7-16
- "Understanding Conditional Rendering and Processing" on page 6-7
- "Running a Page" on page 8-4

12

Deploying an Application

This section describes how to deploy an application.

This section contains the following topics:

- About the Oracle HTML DB Application Development Life Cycle
- About Deploying an Application in Oracle HTML DB
- Deploying an Application to Another Oracle HTML DB Instance
- About Publishing the Application URL
- Using Build Options to Control Configuration

See Also: "Advanced Programming Techniques" on page 15-1

About the Oracle HTML DB Application Development Life Cycle

When developing applications using Oracle HTML DB, you need to find a balance between two dramatically different development methodologies:

- Iterative, rapid application development
- Planned, linear style development

The first approach offers so much flexibility, you run the risk of never completing your project. In contrast, the second approach can yield applications that do not meet the needs of end users even if they meet the stated requirements on paper.

System Development Life Cycle Methodologies to Consider

The System Development Life Cycle (SDLC) is the overall process of developing software using a series of defined steps. There are a number of SDLC models that work well for developing applications in Oracle HTML DB.

The **SDLC waterfall** is probably the best known model. In this methodology, the development process is broken down into the following stages:

1. Project Planning
2. Requirements Definition
3. Design
4. Development
5. Integration and Testing
6. Installation and Acceptance

7. Maintenance

This methodology is referred to as a waterfall since the output from one stage is the input for the next stage. One of the primary problems with this approach is that it is assumed that all requirements can be establish in advanced. Unfortunately, in the real world requirements often change and evolve during the development process.

The Oracle HTML DB development environment enables developers to take a more iterative approach to development. Unlike many other development environments, creating prototypes is easy. With Oracle HTML DB, developers can:

- Use built-in wizards to quickly design an application user interface
- Easily make protoypes available to users and gather feedback
- Implement changes in real time, creating new prototypes instantly

Other methodologies what work well with Oracle HTML DB include:

- **Spiral** - This approach is actually a series of short waterfall cycles. Each waterfall cycles yields new requirements and enables the development team to create a robust series of prototypes.
- **Rapid Application Development (RAD) Life Cycle** - This approach has a heavy emphasis on creating a prototype that closely resembles the final product. Essentially the prototype is an essential part of the requirements phase. One disadvantage of this model is that the emphasis on prototyping can lead to scope creep. Developers can lose sight of their initial goals in the attempt to create the perfect application.

About Deploying an Application in Oracle HTML DB

Deploying an application from one Oracle HTML DB instance to another is a two part process:

- First, you move the supporting database objects.
- Second, you export the application definition and all associated files.

See Also: "Deploying an Application to Another Oracle HTML DB Instance" on page 12-4

Deployment Options to Consider

When you develop an application in Oracle HTML DB, you create the application within a specific workspace. Each workspace has an unique ID and name. A common scenario is to create the application in a development instance and then deploy it to a production instance.

During the deployment process you would need to decide whether to use the existing application ID, the existing workspace, the existing database, or the existing Oracle HTTP Server, or create new ones. Deployment options to consider include:

1. **Do nothing.** Send the URL and login information to users. This approach works well for applications with a small and tolerant user population.
2. **Same workspace and same schema.** Export and then import the application and install it using a different application ID. The approach works well when there are few changes to the underlying objects, but frequent changes to the application functionality.

3. **Different workspace and same schema.** Export and then import the application into a different workspace. This is an effective way to prevent a production application from being modified by developers.
4. **Different workspace and different schema.** Export and then import the application into a different workspace and install it using a different schema.
5. **Different database with all its variations.** Export and then import the application into a different Oracle HTML DB instance and install it using a different schema and database.

Whether to Copy the Workspace

Whether to copy an existing workspace really is a matter of preference. Keep in mind that the production version must have access to all the appropriate objects. For example, you might want to copy a workspace in the following situations:

- When the application subscribes to other objects within the workspace.
- When the application relies on Oracle HTML DB authentication. Copying the workspace would automatically migrate all the required user data.

Whether to Copy the Database

When determining whether to copy the database, remember that the schema against which the application runs must have access to the same objects as the development instance. The actual name of the schema is unimportant. You can change it during the import process.

> **See Also:** "Importing Export Files" on page 12-10

About the Application ID

It is not necessary to have matching application IDs for a development version and production version of an application. In fact, as a best practice never hard code the application ID. Instead use the application alias (defined on the Edit Application Attributes page), or use a built-in substitution string (such as `APP_ID` and `APP_ALIAS`). Using a substitution string is the best approach since it enables you to change the application ID without impacting any application functionality.

> **See Also:** "Application Definition" on page 7-5 for information on defining an application alias and "Built-in Substitution Strings" on page 6-16 for information on using `APP_ID` and `APP_ALIAS`

Whether to Install a New Oracle HTTP Server

Installing Oracle HTML DB loads a new Oracle HTTP Server in a separate Oracle home. Additionally, the installer properly configures Oracle HTTP Server with a `mod_plsql` database access descriptor (DAD) and creates all virtual directory mappings.

Using a different Oracle HTTP Server configuration requires additional configuration. For example, you might want to:

- Use a different Oracle HTTP Server from the one that installs with Oracle HTML DB
- Use the Oracle HTTP Server that installs with Oracle Application Server Release 10*g*
- Use the Oracle HTTP Server that installs with Oracle*9i* Application Server

All of these scenarios require you manually configure the `mod_plsql` DAD and map the directory from which Oracle HTML DB retrieves images.

You can also have a single Oracle HTTP Server serve pages for multiple Oracle HTML DB instances. In this configuration, all Oracle HTML DB instances must be the same version, map to the same image directory, and have a unique `mod_plsql` DAD.

> **See Also:** *Oracle HTML DB How To Documents* section of Oracle Technology Network for more information on implementing these configurations

Deploying an Application to Another Oracle HTML DB Instance

To move an application and related files from one instance of Oracle HTML DB to another, you must export the application definition and all associated files. Exporting your application definition is the first step toward deploying it outside of your development environment.

Topics in this section include:

- How Exporting an Application Works
- About Managing Database Objects
- Exporting an Application and Related Files
- Importing Export Files
- Installing Files from the Export Repository

How Exporting an Application Works

Whether you want to move an application to another workspace or just make a copy of it, the export process involves the following steps:

- Export the application and all related files (See "Exporting an Application and Related Files" on page 12-5)
- Import the exported files into the target Oracle HTML DB instance. (See "Importing Export Files" on page 12-10)

  Note that if the target instance is a different schema, you also need to export and import any required database objects.
- Install the exported files from Export Repository (See "Installing Files from the Export Repository" on page 12-11)

You can import an application into your workspace regardless of the workspace in which it was developed.

About Managing Database Objects

Before you export an application and the appropriate related files, you need to determine if you also need to migrate the database objects referenced by the application.

If the target Oracle HTML DB instance is different from the development environment, you will need to migrate the database objects referenced by the application. In many cases this process can be as simple as using Oracle database export and import utilities to copy the application schema from the development environment to target Oracle HTML DB instance. The following are two common scenarios where this approach will not work:

- When the object development schema refers to tablespaces to which the target instance schema does not have access
- When the development instance schema has sample data that you do not to want migrate to the target instance schema

If a database administrator or an Oracle HTML DB administrator is the person responsible for exporting Oracle HTML DB applications, be sure to clearly communicate if he or she:

- Should include all data when exporting your application
- Should not include data from specific tables you identify

See Also: "Importing Data" on page 4-2 and "Exporting Data" on page 4-3

Exporting an Application and Related Files

You export and import application definitions and all associated files using the Workspace, Application, CSS, Images, Script Files, Themes, and User Interface Defaults buttons located at the top the Export page.

Topics in this section include:

- Exporting an Application
- Exporting a Page in an Application
- Exporting Cascading Style Sheets
- Exporting Images
- Exporting Static Files
- Exporting Script Files
- Exporting Themes
- Exporting User Interface Defaults

Exporting an Application

When you export a application, Oracle HTML DB generates a text file containing PL/SQL API calls.

To export an application:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. From the Application list, select an application
6. From File Format, select how rows in the export file will be formatted:
   - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
   - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. From Build Status Override, select one of the following:
   - **Run Application Only** - Developers can only run an application

- **Run and Build Application** - Developers can both run and edit an application

Selecting **Run Application Only** is an effective way to protect an application from modifications from other developers. Note that if you select **Run Application Only** you cannot set the argument `p_trace` to `Yes`. Also, be aware that once you override the Build Status, you can only change it in Oracle HTML DB Administration Services.

8. Use the **As of** field to export your application as it was previously defined. Specify the number of minutes in the field provided.

   This utility uses the `DBMS_FLASHBACK` package. Because the timestamp to System Change Number (SCN) mapping is refreshed approximately every five minutes, you may have to wait that amount of time to locate the version you are looking for. The time undo information is retained and influenced by the startup parameter `UNDO_RETENTION` (the default is three hours). However, this only influences the size of the undo tablespace. While two databases may have the same `UNDO_RETENTION` parameter, you will be able to go back further in time on a database with fewer transactions since it is not filling the undo tablespace, forcing older data to be archived.

9. Click **Export Application**.

In addition to exporting the actual application file, you may also need to export other related files such cascading style sheets, images, and script files.

> **See Also:** "Managing Application Build Status" on page 18-20 and "Enabling SQL Tracing and Using TKPROF" on page 11-2

Exporting a Page in an Application

You can export a specific page within an application by clicking the Export button on the Page Definition. When exporting a page, remember that some pages may reference shared components. To export all pages within an application as well as application shared components, you need to export the entire application.

> **See Also:** "Exporting an Application" on page 12-5

To export a page in an application:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. From the Applications list, select an application.

   Application Builder appears. The list of pages in the current application appears at the bottom of the page.

3. From the Pages list, select a page.

   The Page Definition appears.

4. On the Page Definition, click **Export** at the top of the page.
5. From Page, select the page to be exported.
6. From File Format, select how rows in the export file will be formatted:
   - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
   - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.

7. Use the **As of** field to export a page as it was previously defined. Specify the number of minutes in the field provided.

   This utility uses the `DBMS_FLASHBACK` package. Because the timestamp to System Change Number (SCN) mapping is refreshed approximately every five minutes, you may have to wait that amount of time to locate the version you are looking for. The time undo information is retained and influenced by the startup parameter `UNDO_RETENTION` (the default is three hours). However, this only influences the size of the undo tablespace. While two databases may have the same `UNDO_RETENTION` parameter, you will be able to go back further in time on a database with fewer transactions since it is not filling the undo tablespace, forcing older data to be archived.

8. Click **Export Page**.

Exporting Cascading Style Sheets

Use the Export Cascading Style Sheets utility to export cascading style sheets you imported.

To export related cascading style sheets:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **CSS** at the top of the page.
6. On the Export Cascading Style Sheets page:
   a. Select the cascading style sheets
   b. From File Format, select how rows in the export file will be formatted:
      – Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      – Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. Click **Export Style Sheets**.

Exporting Images

Use the Export Images utility to export images you have imported.

To export related images:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **Images** at the top of the page.
6. On the Export Images page:
   a. Select application from which to export images.

      Be aware that selecting **Workspace Images** only exports those images in your repository that are not associated with a specific application. If all of your

images are associated with specific applications then the workspace image export file will be empty.

b. From File Format, select how rows in the export file will be formatted:
   - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
   - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.

7. Click **Export Images**.

Exporting Static Files

Use the Export Static Files utility to export static files you have imported.

To export related static files:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **Files** at the top of the page.
6. On the Export Static Files page:
   a. Select the files to be exported.
   b. From File Format, select how rows in the export file will be formatted:
      - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      - Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. Click **Export File(s)**.

Exporting Script Files

Use the Export Script Files utility to export script files from one Oracle HTML DB development instance to another.

To export script files:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **Script Files** at the top of the page.
6. On the Export Script Files page:
   a. Select the files to be exported.
   b. From File Format, select how rows in the export file will be formatted:
      - Choose **UNIX** to have the resulting file contain rows delimited by line feeds.

- Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.

7. Click **Export Script Files**.

Exporting Themes

Use the Export Theme utility to export themes from one Oracle HTML DB development instance to a file.

To export an application theme from the Export page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **Themes** at the top of the page.
6. From Export Theme, select the theme to be exported.
7. Click **Export Theme**.

To export an application theme from the Themes page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **Themes and Templates**.

   The Themes page appears.
5. From the Tasks list, select **Export**.

   The Export page appears.
6. Click **Themes** at the top of the page.
7. From Export Theme, select the theme to be exported.
8. Click **Export Theme**.

> **See Also:** "Managing Themes" on page 9-8

Exporting User Interface Defaults

Exporting User Interface Defaults is useful when you plan to develop on the target machine.

When you export User Interface Defaults, all User Interface Defaults for the selected schema are exported to a single SQL*Plus script. When prompted by your browser, save this file to your hard drive. The file contains an API call to create table hints by making calls to the application PL/SQL API. You can use this file to import User Interface Defaults to another database and Oracle HTML DB instance.

> **See Also:** "Deploying an Application to Another Oracle HTML DB Instance" on page 12-4 and "Managing User Interface Defaults" on page 5-12

To export User Interface Defaults from SQL Workshop:

1. Click **SQL Workshop** on the Workspace home page.
2. Under SQL Workshop, select **User Interface Defaults**.
3. From the Tasks list, select **Import/Export User Interface Defaults**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **User Interface Defaults** at the top of the page.
6. On the User Interface Defaults page:
   a. From Schema, select the schema that owns the table associated with the User Interface Defaults.
   b. From File Format, select how rows in the export file will be formatted:
      – Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      – Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. Click **Export**.

To export User Interface Defaults from the Export page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Export** and click **Next**.
5. Click **User Interface Defaults** at the top of the page.
6. On the User Interface Defaults page:
   a. From Schema, select the schema that owns the table associated with the User Interface Defaults.
   b. From File Format, select how rows in the export file will be formatted:
      – Choose **UNIX** to have the resulting file contain rows delimited by line feeds.
      – Choose **DOS** to have the resulting file contain rows delimited by carriage returns and line feeds.
7. Click **Export**.

See Also: "Managing User Interface Defaults" on page 5-12

Importing Export Files

Once you export an application and any related files, you need to import them into the target Oracle HTML DB instance before you can install them. Always import application first and then the related files.

To import an application and related files:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. When prompted to select a task, select **Import** and click **Next**.

5. In Import file, specify the file you are importing.
6. From File Type, select the type of file you are importing and click **Next**.

   Once you have imported a file, you have the option to install it. You can also install it later from the Export Repository.
7. To install an imported file, click **Install**. On the Application Install page:
   a. From Parse As Schema, select a schema.

      This is the schema against which all of the application's SQL and PL/SQL will be parsed.
   b. From Build Status Override, select one of the following:
      - **Run Application Only** - Developers can only run an application
      - **Run and Build Application** - Developers can both run and edit an application

      Selecting **Run Application Only** is an effective way to protect an application from modifications from other developers. Be aware that once you override the Build Status, you can only change it in Oracle HTML DB Administration Services.
   c. From Install As Application, select one of the following:
      - **Reuse Application ID from Export File**
      - **Auto Assign New Application ID**
      - **Change Application ID**

      Use these options to avoid application ID conflicts. These options come in handy when you need to have two versions of the same application in the same workspace. For example, if you are migrating an application to a production instance but still need to maintain development version.
   d. Click **Install Application**.

      See Also: "Managing Application Build Status" on page 18-20

Installing Files from the Export Repository

Once you have imported files into the target Oracle HTML DB instance, you must install them before they can become active in Application Builder.

To install files stored in the Export Repository:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Export/Import**.
4. From the Navigate menu on the right side of the page, select **View Repository**.
5. Select the file to be installed and click **Install** in the Action column adjacent to the appropriate file.
   a. From Parse As Schema, select a schema.

      This is the schema against which all of the application's SQL and PL/SQL will be parsed.
   b. From Build Status, select one of the following:

- **Run Application Only**
- **Run and Build Application**

Select **Run Application Only** to run the application in the target instance, but make it inaccessible to developers.

c. From Install As Application, select one of the following:

- **Reuse Application ID from Export File**
- **Auto Assign New Application ID**
- **Change Application ID**

Use these options to avoid application ID conflicts. These options come in handy when you need to have two versions of the same application in the same workspace. For example, if you are migrating an application to a production instance but still need to maintain development version.

d. Click **Install Application**.

In addition to installing files, you can also use this page to delete a file from the Export Repository.

To delete a file from the Export Repository:

1. Navigate to the Export Repository.
2. Select the file to be deleted **Delete Checked**.

About Publishing the Application URL

Once you have deployed your application, loaded data, and created users, you can publish your production URL.

You can determine the URL to your application by positioning the mouse over the **Run** icon on the Application Builder home page. The URL displays in the status bar at the bottom of the page.

The Run icon gets its value from the Home link found under the Session Management section of the Application Attributes page. This link is only referenced by this icon and by applications that do not use the Oracle HTML DB Login API. Consider the following example:

```
http://htmldb.oracle.com/pls/otn/f?p=11563:1:3397731373043366363
```

Where:

- `htmldb.oracle.com` is the URL of the server
- `pls` is the indicator to use the `mod_plsql` cartridge
- `otn` is the DAD name
- `f?p=` is a prefix used by Oracle HTML DB
- `11563` is application being called
- `1` is the page within the application to be displayed
- `3397731373043366363` is the session number

To run this example application, you would use the URL:

```
http://htmldb.oracle.com/pls/otn/f?p=11563:1
```

When each user logs in, he or she will receive an unique session number.

Using Build Options to Control Configuration

Build options enable you to conditionally display specific functionality within an application.

Build options have two possible values: INCLUDE and EXCLUDE. If you specify an attribute as being included, then the HTML DB engine considers it part of the application definition at runtime. Conversely, if you specify an attribute as being excluded then the HTML DB engine treats it as if it does not exist.

Topics in this section include:

- Creating Build Options
- Viewing Build Option Reports

Creating Build Options

To create a build option:

1. Navigate to the Workspace home page.
2. Select an application name from the Applications list.
3. On the Application Builder home page, click **Shared Components**.
4. Under Logic, select **Build Options**.
5. To create a new build option, click **Create**.
6. Follow the on-screen instructions.

   Once you create a build option, click the **Edit** icon to change it.

You can choose to enable or disable a build option on the appropriate attributes page. Most attributes pages contain a Configuration Management section where you can select defined build options.

> **See Also:** "Editing Application Attributes" on page 7-4 and "Editing Page Attributes" on page 7-16 for more information on specifying build options

Viewing Build Option Reports

Oracle HTML DB includes a report detailing build option utilization in the current application.

To view a report of build option utilization:

1. Navigate to the Workspace home page.
2. Select an application name from the Applications list.
3. On the Application Builder home page, click **Shared Components**.
4. Under Logic, select **Build Options**.
5. On the Build Options page, click **Utilization**.

   This report displays build option utilization in the current application.

13

Managing a Development Workspace

In the Oracle HTML DB development environment, developers log in to a shared work area called a workspace. Users are divided into two primary roles: *developer* and *workspace administrator*.

Developers can create and edit applications as well as view developer activity, session state, workspace activity, application, and schema reports. Workspace administrators additionally can create and edit user accounts, manage groups, manage development services. This section describes how to access many of these reports and perform Workspace administrator tasks.

This section contains the following topics:

Understanding Administrator Roles

In an Oracle HTML DB development environment there are two different administrator roles:

- Workspace administrator
- Oracle HTML DB administrator

A Workspace administrator uses the Workspace Administration list, located on the Workspace home page, to manage their workspace. In contrast, an Oracle HTML DB administrator is a superuser that manages the entire hosted instance. In order to become a Workspace administrator, an existing administrator must give the developer administrator privileges on the Edit User Page.

> **See Also:** "Managing an Oracle HTML DB Hosted Service" on page 18-1 for more information administering a workspace as an Oracle HTML DB administrator

About the Workspace Administration List

The Workspace Administration list displays on right side of the Workspace home page. Both developers and workspace administrators have access to the following links:

- **Change Password** links to a page where users can change their workspace password.
- **Manage Workspace** links to the Manage Workspace page. Users can click the icons on this page to view the Developer Activity Log, access tools and reports for managing sessions state, view workspace activity reports as well as view application and schema reports
- **About HTML DB** links to an About page. The About page lists basic product information including the product version, schema compatibility, application owner, workspace information, current user name, language preference, and database version.
- **Review Demonstration Applications** links to the Demonstration Applications page. Demonstration Applications contains links to sample applications that install with Oracle HTML DB. Users can install and run these demonstration applications to learn more about the different types of functionality you can include in Oracle HTML DB applications.

Additionally, workspace administrators have access to the Manage Users and Manage Service links.

See Also:

- "Changing Your Password" on page 13-2
- "Monitoring Workspace and User Activity" on page 13-3
- "Viewing Application and Schema Reports" on page 13-4
- "Managing Session State and User Preferences" on page 13-5
- "Creating New User Accounts" on page 13-7 and "Editing Existing User Accounts" on page 13-8 for more information on specifying a developer as an administrator
- "Managing an Oracle HTML DB Hosted Service" on page 18-1 for more information administering a workspace as an Oracle HTML DB administrator

Changing Your Password

All users can change their password using the Change Password page.

To change your password:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Change Password**.
3. Type a new password in the Password field and then retype the password in the Confirm Password field.
4. Click **Apply Changes**.

Note: All users (developers and administrators) can use the Change Password link on the Oracle HTML DB home page to reset their password.

See Also: "Changing a User Password" on page 13-8

Monitoring Workspace and User Activity

Both developers and workspace administrators can monitor workspace utilization and user activity by selecting reports on the Monitor page. To view counts of application changes by developer users can access the Developer Activity Log.

This section contains the following topics:

- Viewing Workspace and User Activity Reports
- Viewing Application Changes by Developer and Day
- Purging Log Files

Viewing Workspace and User Activity Reports

The Monitor page displays links to a variety of page and application reports, including top views, page view statistics, top application changes, and application change statistics.

To access the Monitor page:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Workspace Reporting, click **Monitor Activity**.

   The Monitor page is divided into the following sections:

   - Top Views
   - Page View Statistics
   - Top Changes
   - Application Change Statistics
4. Select a report to review

Viewing Application Changes by Developer and Day

The Developer Activity Log displays counts of application changes by developer, by day, and application.

To view Developer Activity Log:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Logs**.
4. Click **Monitor Developer Activity Log**.
5. Specify a time frame, the appropriate number of rows, and click **Go**.

   By default, the Developer Activity by Developer by Day report appears.

6. To view additional details, select a user ID.
7. To view additional activity reports, click one of the following buttons at the top of the page:
   - Application Changes by Month
   - Application Changes by Day Report
   - Application Changes by Day Link Chart

Purging Log Files

The Developer Activity Log track changes to applications within the current workspace. Log entries older then one month are automatically deleted. You can manually purge developer logs and the External Count Clicks log on the Log files page.

> **See Also:** "Managing Logs" on page 18-13 for more information deleting log files as an Oracle HTML DB administrator

Purging the Developer Activity Log

To purge the Developer Activity Log:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Logs**.
4. Click **Purge Dev. Log**.

Purging the External Clicks Log

The External Clicks Log counts clicks from an Oracle HTML DB application to an external site. You can implement this functionality using `COUNT_CLICK` procedure.

> **See Also:** "COUNT_CLICK Procedure" on page 17-4

To purge the External Clicks Log:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Logs**.
4. Click **Purge Click Log**.

Viewing Application and Schema Reports

Application and schema reports are available to all users. Application Reports offer useful information about the size, scope, and content of the applications being developed in a workspace. Schema Reports offer summaries of database privileges by schema as well as a list of all database schemas available in the current workspace.

To view application and schema reports:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.

   The Administration Services page appears.

3. From the Tasks list on the right side of the page, select **Application Reports**.

   The Administrative Reports page appears.

4. Select a report to review.

Managing Session State and User Preferences

A session is a logical construct that establishes persistence (or stateful behavior) across page views. Each session is assigned a unique ID which the HTML DB engine uses to store and retrieve an application's working set of data (or session state) before and after each page view. Sessions persist in the database until purged by an administrator.

Workspace administrators can purge session state or user preferences within their workspace on the Session State Management page. Developers can purge their session state and user preferences for the current session.

Topics in this section include:

- Managing Session State and User Preferences for the Current Session
- Purging Recent Sessions by Age
- Viewing Session Details Prior to Removing Session State
- Viewing Preferences for a Specific User
- Purging Preferences for a Specific User

See Also:

- "Understanding Session State Management" on page 6-8
- "Managing User Preferences" on page 15-22
- "Managing Session State" on page 18-16

Managing Session State and User Preferences for the Current Session

Workspace administrators and developers can use the Session State Management page to manage session state and user preferences for the current session.

To manage session state and user preferences for the current session:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Session State**.
4. When the Session State Management page appears, click **Report, with an option to purge your current session**.
5. Under Session State you can:
   - Reset the session state for the current session by clicking **Purge Session State**.
   - View information about the current session by clicking **View Session State**.
6. Under User Preferences, you can:
   - View preferences for the current user, by clicking **View Preferences**.
   - Reset user preferences for the current user by clicking **Reset Preferences**.

See Also: "Viewing Session State" on page 6-9

Purging Recent Sessions by Age

Sessions are used to maintain user state. Workspace administrators can purge existing sessions by age.

To purge existing session by age:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Session State**.
4. On the Session State Management page, select **Purge existing sessions by age**.
5. Make a selection from the Sessions older than list.
6. Click one of the following buttons:
   - **Report Session**. Generates a report detailing the total number of sessions for the workspace, the number of users, and the number of old sessions.
   - **Purge Sessions**. Purges existing sessions by age.

See Also: "Viewing Session State" on page 6-9

Viewing Session Details Prior to Removing Session State

Workspace administrators can determine whether to remove existing sessions by first reviewing session details on the Session State page.

To view session details prior to removing session state:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Session State**.
4. On the Session State Management page, select **Report recent sessions with drilldown to session details**.
5. Select a session ID.
6. When Session Information appears, click either:
   - **Remove State**
   - **Remove Session**

Viewing Preferences for a Specific User

Workspace administrators view preferences for a specific user on the Preferences Report.

To view the Preferences Report:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Session State**.
4. On the Session State Management page, select **Report preferences for users**.

   The Preferences Report page appears.
5. To search for a specific user, enter a user name in the User field and click **Go**.

Purging Preferences for a Specific User

Workspace administrators purge preferences for a specific user on the Purge Preferences page.

To purge preferences for a specific user:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Session State**.
4. On the Session State Management page, select **Purge preferences for a selected user**.

   The Purge Preferences page appears.
5. Select a specific user in the User field and click **Report**.

   The User Preferences for the selected user appear.
6. To purge the displayed user preferences, click **Purge User Preferences**.

Managing Users

Workspace administrators can create new user accounts, manage existing user accounts, and change user passwords. User accounts are particularly useful if you are using HTML DB Authentication. HTML DB Authentication checks the username and password against the Oracle HTML DB account repository. The Oracle HTML DB account repository contains account information that developers and administrators when logging in to Oracle HTML DB applications.

Topics in this section include:

- Creating New User Accounts
- Editing Existing User Accounts
- Changing a User Password
- Managing Groups

> **See Also:** "About HTML DB Account Credentials" on page 14-6 for more information on implementing HTML DB Authentication

Creating New User Accounts

Workspace administrators create new user accounts on the Create User page.

To create a new user account:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Users**.

   The Manage Users page appears.
3. Under Manage Users, click **Create New User**.

   The Create User page appears.
4. Under User Identification, enter the appropriate information. Required fields are marked with a red asterisk (*).
5. Under Developer Privileges, specify whether the user is a developer or an administrator.

- **User is a developer** - These users can create and edit applications as well as view developer activity, session state, workspace activity, application, and schema reports.
- **User is an administrator** - Workspace administrators additionally can create and edit user accounts, manage groups, alter passwords of users within the same workspace, and manage development services.

6. Under User Groups, select an optional user group.

   You can use groups to restrict access to various parts of an application. Groups are primarily useful when using HTML DB Authentication.

7. Click **Create User** or **Create and Create Another**.

See Also: "Managing Groups" on page 13-9 and "Adding Users to and Removing Users from a Group" on page 13-10

Editing Existing User Accounts

Workspace administrators edit existing user accounts on the Edit User page.

To edit an existing a user account:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Users**.

   The Manage Users page appears.

3. Under Manage Users, click **Edit Users**.

   The Edit Users page appears. On the Edit Users page, you can:

   - Create a new user by clicking **Create**
   - Search for an existing user by entering a search condition in the Find field and clicking **Go**

4. To edit a user account, click the **Edit** icon.

   The Edit User page appears.

5. Under Developer Privileges, specify whether the user is a developer or an administrator.

   Developers having administrator privilege have access to all tools and reports available on the Workspace Administration list. These users can also alter passwords of users within the same workspace.

6. Under User Groups, select an optional user group.

   You can use groups to restrict access to various parts of an application. Groups are primarily useful when using HTML DB Authentication.

7. Follow the on-screen instructions.

See Also: "Adding Users to and Removing Users from a Group" on page 13-10

Changing a User Password

Workspace administrators can change the password of any user in their workspace.

To change a user password:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Users**.

   The Manage Users page appears.
3. Under Manage Users, click **Edit Users**.

   The Edit Users page appears.
4. Locate the appropriate user and click the **Edit** icon. You can search for an existing user by entering a search condition in the Find field and clicking **Go**.

   The Edit User page appears.
5. Scroll down to Password, type a new password in the Password and Confirm Password fields, and click **Apply Change**s.

Managing Groups

Workspace administrators can create groups to restrict access to various parts of an application. Keep in mind, however, that groups are not portable over different authentication schemes. Groups are primarily useful when using HTML DB Authentication.

Topics in this section include:

- Creating and Editing Groups
- Viewing Group Assignment Reports
- Adding Users to and Removing Users from a Group

> **See Also:** "About HTML DB Account Credentials" on page 14-6 for more information on implementing HTML DB Authentication

Creating and Editing Groups

To create a new group:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Users**.
3. Under Manage Groups, click **Create New Group**.

   The Create User Groups page appears.
4. Specify a group name, description, and click **Create Group**.

To edit an existing group:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Users**.
3. Under Manage Groups, click **Edit Groups**.

   The Create User Groups page appears.
4. Click the **Edit** icon next to the appropriate group.
5. Make the appropriate change and click **Apply Changes**.

Viewing Group Assignment Reports

To view a report of user group assignments:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Users**.
3. Under Manage Groups, click **Report User Group Assignments**.

   The User Groups Assignments Report appears.

Adding Users to and Removing Users from a Group

To add a user from a group:

1. Navigate to the Edit User page:
   a. Navigate to the Workspace home page.
   b. From the Workspace Administration list, select **Manage Users**.
   c. Under Manage Users, click **Edit Users**.

      The Edit Users page appears.
   d. To edit a user account, click the **Edit** icon.

      The Edit User page appears.
2. To add a user to a group:
   a. Scroll down to User Groups.
   b. Select a group from the Groups list.
3. To remove a user to a group:
   a. Scroll down to User Groups.
   b. Deselect the selected group in the Groups list.

Managing Development Services

Workspace administrators can use the Provisioning Services section of the Administration Services page to:

- View information and reports describing the current workspace
- Submit a request to the Oracle HTML DB administrator for a new database schema, additional storage, or to terminate workspace service

Topics in this section include:

Viewing Current Workspace Status

Workspace administrators can view current workspace status on the Manage Development Services page.

To view current workspace status:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.

3. Under Administration Services, click **Manage Service**.

   The Manage Development Services page appears.

4. Select **Report Utilization**.
5. Follow the on-screen instructions.

Requesting a Database Schema

To submit a request to the Oracle HTML DB administrator for a new database schema:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Manage Service**.

   The Manage Development Services page appears.

4. Select **Request Schema**.
5. Follow the on-screen instructions.

Requesting Additional Storage

To submit a request to the Oracle HTML DB administrator for additional storage space for your workspace:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Manage Service**.

   The Manage Development Services page appears.

4. Select **Request Storage**.
5. Follow the on-screen instructions.

Requesting Service Termination

To submit a request to the Oracle HTML DB administrator to terminate workspace service:

1. Navigate to the Workspace home page.
2. From the Workspace Administration list, select **Manage Workspace**.
3. Under Administration Services, click **Manage Service**.

   The Manage Development Services page appears.

4. Select **Terminate Service**.
5. Follow the on-screen instructions and click **Request Termination**.

14

Managing Security

You can provide security for your application through authentication and authorization. **Authentication** is the process of establishing users' identities before they can access an application. **Authorization** controls user access to specific controls or components based on user privileges.

This section contains the following topics:

Escaping Special Characters Rendered from Session State

Cross site scripting (also referred to as XSS) is a security breach that takes advantage of dynamically generated Web pages. In a XSS attack, a Web application is sent a script that activates when it is read by a user's browser. Once activated, these scripts can steal data or even session credentials and return the information to the attacker. If malicious code was introduced into an Oracle HTML DB application, it could be rendered into HTML regions and other places within the application during normal page rendering. To prevent the introduction of malicious code into session state, the HTML DB engine escapes characters in certain cases as outlined below.

Specifically, Oracle HTML DB escapes special character:

- For &MY_ITEM. substitutions performed within rendered HTML regions. For example, the rendered character of the less than sign (<) is represented in HTML as `<`.

  Note that this rule does not apply to many static areas such as region and page headers and footers, region titles, item labels, and other places. Item substitution is performed without escaping special characters in those sections.
- For `&MY_ITEM.` substitutions performed in report output, including column headings.
- In other situations where `&MY_ITEM.` substitutions take place. For example:
  - During the replacement of branch targets (such as `&LAST_PAGE.`) in direct page branches (not redirects)
  - In branches that use an item name to contain the branch target
- In displayed form field values.

For example, the rendered character of the less than sign (<) is represented in HTML as `<`. Items of type **Display as text (does not save state)** are not escaped when they are rendered. However, items of this type have their values escaped when passed into a page in the URL and subsequently stored in session state. This filter on input behavior also applies to application level items (or items that are not associated with a particular page).

Additionally, form field values submitted to be saved in session state are passed in `POSTDATA` so that they are received by the `ACCEPT` procedure and saved in session state in unescaped format. Oracle HTML DB does not attempt to filter output sent to the browser. As a developer, you have full control of any dynamic content sent to the browser, including PL/SQL regions or any other developer provided code.

As an added precaution, always follow best practices to guard against cross site scripting attacks. For example, during the development process think about whether or not an item could have malicious content stored in it by URL tampering or another means. Avoid referencing form items using `&MY_ITEM.` for rendering in most types of static areas since it can lead to security vulnerabilities. Instead, use application level items or page items of type **Display as text (does not save state)**. Another easy safeguard is to escape less than (<), greater than (>), and ampersands (&).

Understanding Security

You can provide security for your application through authentication and authorization. **Authentication** is the process of establishing users' identities before they can access an application. **Authorization** controls user access to specific controls or components based on user privileges.

To access either the Authentication Schemes or Authorization Schemes pages:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select either **Authentication** or **Authorization**.

See Also:

- "Establishing User Identity Through Authentication" on page 14-3
- "Providing Security Through Authorization" on page 14-9

Using the Security Navigation List

The Security Navigation list displays on the right side of the Authentication Schemes and Authorization Schemes pages. This list offers quick access to relevant security homes pages as well as related utilization reports.

To access the Security Navigation list:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.

4. Under Security, select either **Authentication** or **Authorization**.

   The Security Navigation list displays on the right side of the page and contains the following links:

   - Security Home
   - Authentication Schemes
   - Manage Authentication Scheme Subscriptions
   - Authorization Schemes
   - Manage Authorization Scheme Subscriptions
   - Authorization Scheme Utilization

Establishing User Identity Through Authentication

Authentication is the process of establishing each user's identify before they can access your application. Authentication may require a user identify a username and password or could involve the use of digital certificates or a secure key.

When you create an authentication scheme, you have the option of choosing from a number of preconfigured authentication schemes, copying an authentication scheme from an existing application, or creating your own custom authentication scheme.

Topics in this section include:

- Understanding How Authentication Works
- Creating an Authentication Scheme
- Using the Authentication Scheme Repository
- Viewing the Current Authentication Scheme for an Application
- About Preconfigured Authentication Schemes
- About Creating an Authentication Scheme from Scratch

Understanding How Authentication Works

You determine how your application interacts with users. If all users have the same rights and privileges they are referred to as public users. However, if your application needs to track each user individually, you need to specify an authentication method.

Authentication establishes the identity of each user who accesses your application. Many authentication processes require a user provide some type of credentials such as a username and password. These credentials are then evaluated and they either pass or fail. If the credentials pass, the user has access to the application. Otherwise, access is denied.

Once a user has been identified, the HTML DB engine keeps track of each user by setting the value of a built-in substitution string. As a user navigates from page to page, the HTML DB engine sets the value of APP_USER to identify who they are. The HTML DB engine uses APP_USER as one component of a key for tracking each user's session state.

From a programming perspective, you can access APP_USER using the following syntax:

- From PL/SQL:

  ```
  V('APP_USER')
  ```

- As a bind variable from either PL/SQL or SQL:

  ```
  :APP_USER
  ```

You can use APP_USER to perform your own security checks and conditional processing. For example, suppose you created the following table:

```
CREATE TABLE my_security_table (
  user_id   VARCHAR2(30),
  privilege VARCHAR2(30));
```

Once created, you could populate this table with user privilege information and then use it to control the display of pages, tabs, navigation bars, buttons, regions, or any other control or component.

See Also: "Using Substitution Strings" on page 6-16

Creating an Authentication Scheme

As you create your application, you need to determine whether to include authentication. You can:

- **Choose to not require authentication.**

  Oracle HTML DB does not check any user credentials. All pages of your application are accessible to all users.

- **Select a built-in authentication scheme.**

  Create an authentication method based on available preconfigured authentication schemes. Depending on which scheme you choose, you may also have to configure the corresponding components of Oracle 9*i*AS, Oracle Internet Directory, or other external services.

- **Create custom authentication scheme.**

  Create a custom authentication method, giving you complete control over the authentication interface. To implement this approach, you must provide a PL/SQL function the HTML DB engine executes before processing each page request. This function's Boolean return value determines whether the HTML DB engine processes the page normally or displays a failure page.

To create an authentication scheme:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select either **Authentication**.

   The Authentication Schemes page appears.
5. To create a new authentication scheme, click **Create Scheme**.
6. Specify how the scheme should be created by selecting one of the following:
   - **Based on preconfigured scheme**
   - **As a copy of an existing scheme**
   - **From Scratch**

7. Follow the on-screen instructions

See Also:

- "About Preconfigured Authentication Schemes" on page 14-5
- "About Creating an Authentication Scheme from Scratch" on page 14-7

Using the Authentication Scheme Repository

Once created, available authentication schemes display in the Authentication Schemes Repository.

To navigate to the Authentication Schemes Repository:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select either **Authentication**.

   The Authentication Schemes page appears.

From the Authentication Schemes repository, you can:

- Make an authentication scheme current by selecting the **make current** link
- Edit an authentication scheme by clicking the **Edit** icon
- View a flow chart explanation of an authentication scheme by clicking the **View** icon
- Create a new authentication scheme by clicking **Create Scheme** and following the on-screen instructions

Viewing the Current Authentication Scheme for an Application

To view the current authentication scheme for an application:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. Click the **Edit Attributes** icon.

   The Edit Application Attributes page appears.
4. Scroll down to the Session Management attribute and click **manage**.

   The Authentication Schemes page appears. Available authentication schemes display in the Authentication Schemes Repository. You apply an authentication scheme to an application by designating it as current.
5. To apply an authentication scheme to the current application, select the **make current** link

About Preconfigured Authentication Schemes

When you select a preconfigured authentication scheme, Oracle HTML DB creates an authentication scheme for your application that follows a standard behavior for

authentication and session management. The following list describes available preconfigured authentication schemes:

- **Open Door Credentials** enables anyone to access your application using a built-in login page which captures a username. This can be useful during application development.
- **HTML DB Account Credentials** refers to the internal user accounts (also known as "cookie user" accounts) created and managed in the Oracle HTML DB user repository. Using this scheme authentication method, your application can easily authenticate against these accounts.
- **LDAP Credentials Verification** requires you specify configuration parameters about the external LDAP directory you will be using.
- **No Authentication (using DAD)** gets the username from the Database Access Descriptor (DAD), either as the value stored in the DAD configuration or, if the account information is not stored in the DAD configuration, as the username captured using the basic authentication challenge.
- **Oracle Application Server Single Sign-On (HTML DB engine as Partner App)** delegates authentication to the Oracle AS Single Sign-On (SSO) Server. To you use authentication scheme, your site must have already been registered as a partner application with the SSO server. For more information, contact your administrator.
- **Oracle Application Server Single Sign-On (My application as Partner App)** delegates authentication to the SSO server. Requires you register an application with SSO as a partner application.

About DAD Credentials Verification

Database Access Descriptor (DAD) database authentication uses the Oracle database native authentication and user mechanisms to authenticate users using a basic authentication scheme. To use DAD credentials verification:

- Each application user must have a user account in the Oracle database.
- You must configure a PL/SQL DAD for basic authentication (without account information).

  This results in one username/password challenge for browser session for your application users. The user identity token is then made available in the APP_USER item.

DAD database authentication is useful when you need to implement an authentication method that requires minimal setup for a manageable number of users. Ideally these users would already have self-managed accounts in the database and your use of this authentication method would be short lived (for example, during the demonstration or prototyping stages of development).

The main drawback of this approach is burdensome account maintenance, especially if users do not administer their own passwords, or if their database accounts exist only to facilitate authentication to your application.

About HTML DB Account Credentials

HTML DB Account Credentials authentication uses internal user accounts (also known as "cookie user" accounts) created and managed in the Oracle HTML DB user repository. Workspace administrators can create and edit user accounts using the Manage Users page. HTML DB Account Credentials is a good solution when:

- You want control of the user account repository

- Username and password based approach to security is sufficient
- You do not need to integrate into a single sign-on framework

This is an especially good approach when you need to get a group of users up and running on a new application quickly.

> **See Also:** "Managing Users" on page 13-7 for more information on creating and managing user accounts

About LDAP Credentials Verification

Any authentication scheme that uses a login page may be configured to use Lightweight Directory Access Protocol (LDAP) to verify the username and password submitted on the login page. Application Builder includes wizards and edit pages that explain how to configure this option. These wizards assume that an LDAP directory accessible to your application for this purpose already exists and that it can respond to a `SIMPLE_BIND_S` call for credentials verification. When you create a LDAP Credentials authentication scheme, the wizard requests and saves the LDAP host name, LDAP port, and the DN string. An optional pre-processing function can be specified to adjust formatting of the username passed to the API.

About Single Sign-On Server Verification

Oracle HTML DB applications can operate as partner applications with Oracle Application Server's Single Sign-On (SSO) infrastructure. To accomplish this, you must register your application (or register the HTML DB engine) as the partner application. To register your application or the HTML DB engine as a partner application, follow the Oracle Application Server instructions for registering partner applications and install the Oracle 9iAS SSO Software Developer Kit (SDK).

If you choose this approach, your application will not use an integrated login page. Instead, when a user accesses your application in a new browser session, the HTML DB engine redirects to the Single Sign-On login page. After the user is authentication by SSO, the SSO components redirect back to your application, passing the user identity and other information to the HTML DB engine. The user can then continue to use the application until they log off, terminate their browser session, or until some other session-terminating event occurs.

About Creating an Authentication Scheme from Scratch

Creating an authentication scheme from scratch gives you complete control over your authentication interface. This is the best approach for applications when any of the following is true:

- Database authentication, or other methods are not adequate.
- You want to develop your own login form and associated methods.
- You want to delegate all aspects of user authentication to external services such as Oracle 9*i*AS Single Sign-On.
- You want to control security aspects of Oracle HTML DB session management.
- You want to record or audit activity at the user or session level.
- You want to enforce session activity or expiry limits.
- You want to program conditional n-way redirection logic before Oracle HTML DB page processing.

- You want to integrate your application with non-Oracle HTML DB applications using a common session management framework.
- Your application consists of multiple applications that operate seamlessly (for example, more than one Oracle HTML DB application ID).

> **See Also:** "HTMLDB_CUSTOM_AUTH" on page 17-54 for more information

About Session Management Security

When running custom authentication, Oracle HTML DB attempts to prevent two improper situations:

- Intentional attempts by a user to access session state belonging to someone else. However, users can still type in an arbitrary application session ID into the URL.
- Inadvertent access to a stale session state (probably belonging to the same user from an earlier time). This would commonly result from using bookmarks to application pages.

Oracle HTML DB checks that the user identity token set by the custom authentication function matches the user identity recorded when the application session was first created. If the user has not yet been authenticated and the user identity is not yet known, the session state being accessed does not belong to someone else. These checks determine whether the session ID in the request can be used. If not, the HTML DB engine redirects back the same page using an appropriate session ID.

Building a Login Page

When you create a new application in Oracle HTML DB, a login page is created. The alias for the page is `'LOGIN'`. You can use this page as the 'invalid session page' in an authentication scheme. The page is constructed with processes that call the Oracle HTML DB login API to perform credentials verification and session registration.

You can also build your own login pages using the pre-built pages as models and tailoring all of the user interface and processing logic to your requirements.

To create a login page for your application:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. Click **Create Page**.
4. From the Tasks list on the right side of the page, select **Create Login Page**.
5. Follow the on-screen instructions.

About Deep Linking

Deep linking refers to the ability to link to an Oracle HTML DB page out of context (for example, from a hyperlink in an e-mail or workflow notification). When you link to a page out of context and the application requires the user be authenticated, the user will be taken to the login page. After credentials verification, the HTML DB engine automatically displays the page that was referenced in the original link. Deep linking is supported for applications that use authentication schemes.

Providing Security Through Authorization

Authorization is a broad term for controlling access to resources based on user privileges. While conditions control the rendering and processing of specific page controls or components, authorization schemes control user access to specific controls or components.

Topics in this section include:

- How Authorization Schemes Work
- Creating an Authorization Scheme
- Attaching an Authorization Scheme to an Application, Page, or Components
- Viewing Authorization Scheme Utilization Reports

How Authorization Schemes Work

An authorization scheme extends the security of your application's authentication scheme. You can specify an authorization scheme for an entire application, a page, or specific control such as a region, item, or button. For example, you could use an authorization scheme to selectively determine which tabs, regions, or navigation bars a user sees.

An authorization scheme either succeeds or fails. If a component or control level authorization scheme succeeds, the user can view the component or control. If it fails, the user cannot view the component or control. If an application or page level authorization scheme fails, then Oracle HTML DB displays a previously defined message.

When you define an authorization scheme you give it a unique name. Once defined, you can attach it to any component or control in your application. To attach an authorization scheme to a component or control in your application, simply navigate to the appropriate attributes page and select an authorization scheme from the Authorization Scheme list.

Creating an Authorization Scheme

Before you can attach an authorization scheme to an application or an application component or control, you must first create it.

To create an authorization scheme:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select **Authorization**.
5. Click **Create**.
6. Specify how to create an authorization scheme by selecting one of the following:
   - **From Scratch**
   - **As a Copy of an Existing Authorization Scheme**
7. Follow the on-screen instructions.

To edit attributes of an existing authorization scheme:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select **Authorization**.

   Existing Authorization Schemes display at the bottom of the page.
5. To edit attributes for an existing authorizations scheme, click the **Edit** icon.
6. Follow the on-screen instructions.

About the Evaluation Point Attribute

You can specify when your authorization scheme is validated in the Evaluation Point attribute. You can choose to have your authorization scheme validated once for each session or once for each page view.

Keep in mind, that if you specify that an authorization scheme should be evaluated once for each session and the authorization scheme passes, the underlying code, test, or query will not be executed again for the duration of the application session. If your authorization scheme consists of a test whose results might change if evaluated at different times during the session, then you should specify that the evaluation point be once for each page view.

About Resetting Authorization Scheme State

If an authorization scheme is validated once for each session, Oracle HTML DB caches the validation results in each user's session cache. You can reset a session's authorization scheme state by calling the `HTMLDB_UTIL.RESET_AUTHORIZATIONS` API:

Calling this procedure nulls out any previously cached authorization scheme results for the current session. Be aware that this procedure takes no arguments and is part of the publicly executable `HTMLDB_UTIL` package.

See Also: "RESET_AUTHORIZATIONS Procedure" on page 17-21

Attaching an Authorization Scheme to an Application, Page, or Components

Once you have created an authorization scheme you can attach it to an entire application, page, control, or component.

Topics in this section include:

- Attaching an Authorization Scheme to an Application
- Attaching an Authorization Scheme to a Page
- Attaching an Authorization Scheme to a Control or Component

Attaching an Authorization Scheme to an Application

To attach an authorization scheme to an application:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. Click the **Edit Attributes** icon.

The Edit Application Attributes page appears.

4. Scroll down to Authorization and make a selection from the Authorization Scheme list.

Attaching an Authorization Scheme to a Page

To attach an authorization scheme to a page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. From the Pages list, select a page name.
4. Click **Edit Attributes**.
5. Scroll down to Security and make a selection from the Authorization Scheme list.

Attaching an Authorization Scheme to a Control or Component

To attach an authorization scheme to a page component or control:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. From the Pages list, select a page name.

   The Page Definition appears.
4. Click the name of the component or control to which you want to apply the authorization scheme.
5. Scroll down to Authorization and make a selection from the Authorization Scheme list.

Viewing Authorization Scheme Utilization Reports

You can use the Authorization Scheme Utilization Report to view details about authorization schemes included in your application.

To view Authorization Scheme Utilization Report:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.

   The Shared Components page appears.
4. Under Security, select either **Authorization**.
5. From the Navigation list, select either **Authorization Scheme Utilization**.
6. Make a selection from the Authorization Scheme list and click **Go**.
7. To view additional reports indicating which pages having authorization schemes and which do not, make a selection from the Tasks list.

15

Advanced Programming Techniques

This section provides information about advanced programming techniques including establishing database links, using collections, running background SQL, utilizing Web Services and managing user preferences.

This section contains the following topics:

- Sending E-mail from an Application
- Accessing Data with Database Links
- Using Collections
- Running Background PL/SQL
- Implementing Web Services
- Managing User Preferences

> **See Also:** "Oracle HTML DB APIs" on page 17-1 and "Deploying an Application" on page 12-1

Sending E-mail from an Application

You can send an e-mail from an Oracle HTML DB application by:

- Creating a background job to periodically send all mail messages stored in the active mail queue
- Calling the PL/SQL package `HTMLDB_MAIL`

Topics in this section include:

- Sending E-mail Using a Background Job
- Sending E-mail Manually by Calling HTMLDB_MAIL

> **See Also:** "Configuring Oracle HTML DB to Send Mail" on page 18-18 and "Managing E-mail" on page 18-21 for more information on viewing the mail queue and the mail log

Sending E-mail Using a Background Job

Oracle HTML DB stores unsent e-mail messages in a table named `HTMLDB_MAIL_QUEUE`. A `DBMS_JOB` background process is automatically created when you install Oracle HTML DB. This background process pushes the mail queue every 15 minutes. The package that is executed by the background process has two parameters:

- p_smtp_host is the hostname of your SMTP gateway. The default value is localhost.
- p_smtp_portno is the port number of your SMTP gatway. The default value is 25.

The most efficient approach to sending e-mail is to create a background job (using a DBMS_JOB package) to periodically send all mail messages stored in the active mail queue.

> **See Also:** "Configuring Oracle HTML DB to Send Mail" on page 18-18

Sending E-mail Manually by Calling HTMLDB_MAIL

You can send an e-mail from an Oracle HTML DB application by calling a PL/SQL package called HTMLDB_MAIL. This package is built on top of the Oracle supplied UTL_SMTP package. Because of this dependence, in order to use HTMLDB_MAIL, the UTL_SMTP package must be installed and functioning.

> **See Also:** *PL/SQL Packages and Types Reference* for more information

HTMLDB_MAIL contains two procedures for manually sending e-mail:

- Use the HTMLDB_MAIL.SEND procedure to manually send an outbound e-mail message from your application
- Use HTMLDB_MAIL.PUSH_QUEUE to deliver mail messages stored in HTMLDB_MAIL_QUEUE

Oracle HTML DB stores unsent e-mail messages in a table named HTMLDB_MAIL_QUEUE. You can deliver mail messages stored in this queue to the specified SMTP gateway by calling the procedure HTMLDB_MAIL.PUSH_QUEUE. This procedure requires two input parameters:

- p_smtp_hostname defines the hostname of your SMTP gateway
- p_smtp_portno defines port number of your SMTP gateway (for example, 25)

Oracle HTML DB logs successfully submitted messages in the table HTMLDB_MAIL_LOG with the timestamp reflecting your server's local time.

The following example demonstrates the use of the HTMLDB_MAIL.PUSH_QUEUE procedure using a shell script. This example only applies to UNIX/LINUX installations. In this example, the SMTP gateway hostname is defined as smtp01.oracle.com and the SMTP gateway port number is *25*.

```
SQLPLUS / <<EOF
FLOWS_010600.HTMLDB_MAIL.PUSH_QUEUE('smtp01.oracle.com','25');
DISCONNECT
EXIT
EOF
```

> **See Also:** "HTMLDB_MAIL" on page 17-27 for more information on using the HTMLDB_MAIL

Accessing Data with Database Links

Since Oracle HTML DB runs on top of an Oracle database, you have access to all distributed database capabilities. Typically, you perform distributed database operations using database links.

To create a database link:

1. Navigate to the Workspace home page.
2. Click **SQL Workshop**.
3. Under SQL Workshop, select **Create Object**.

   The Create Database Object Wizard appears.
4. Select **Database Link**.
5. Follow the on-screen instructions.

   Note that Database Link names must conform to Oracle naming conventions and cannot contain spaces, or start with a number or underscore.

See Also: *Oracle Database Administrator's Guide*

Using Collections

Collections enable you to temporarily capture one or more non-scalar values. You can use collections to store rows and columns currently in session state so they can be accessed, manipulated, or processed during a user's specific session. Think of a collection as a bucket in which you can temporarily store and name rows of information.

Examples of when you might use collections include:

- When you are creating a data-entry wizard in which multiple rows of information first need to be collected within a logical transaction. You can use collections to temporarily store the contents of the multiple rows of information, prior to performing the final wizard step when both the physical and logical transactions are completed.
- When your application includes an update page on which a user updates multiple detail rows on one page. They can make many updates, apply these updates to a collection, then call a final process to apply the changes to the database.
- When you are building a wizard where you are collecting an arbitrary number of attributes. At the end of the wizard the user then performs a task that takes the information temporarily stored in the collection and applies it to the database.

Topics in this section include:

- Merging Collections
- Managing Collections
- Clearing Collection Session State

About the HTMLDB_COLLECTION API

Every collection contains a named list of data elements (or members) which can have up to 50 attributes (or columns). You insert, update, and delete collection information using the PL/SQL API `HTMLDB_COLLECTION`.

About Collection Naming

When you create a new collection, you must give it a name that cannot exceed 255 characters. Note that collection names are not case-sensitive and will be converted to upper case.

Once named, you can access the values in a collection by running a SQL query against the view `HTMLDB_COLLECTION`.

Creating a Collection

Every collection contains a named list of data elements (or members) which can have up to 50 attributes (or columns). Use the following methods to create a collection:

- `CREATE_COLLECTION`
- `CREATE_OR_TRUNCATE_COLLECTION`
- `CREATE_COLLECTION_FROM_QUERY`

`CREATE_COLLECTION` raises an exception if the named collection already exists. For example:

```
HTMLDB_COLLECTION.CREATE_COLLECTION(
    p_collection_name => collection name );
```

`CREATE_OR_TRUNCATE_COLLECTION` creates a new collection if the named collection does not exist. If the named collection already exists, this method truncates it. Truncating a collection empties it, but leaves it in place. For example:

```
HTMLDB_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
    p_collection_name => collection name );
    p_generate_md5    => YES or NO );
```

`CREATE_COLLECTION_FROM_QUERY` creates a collection and then populates it with the results of a specified query. For example:

```
HTMLDB_COLLECTION.CREATE_COLLECTION_FROM_QUERY(
    p_collection_name => collection name,
    p_query           => your query );
    p_generate_md5    => YES or NO );
```

`CREATE_COLLECTION_FROM_QUERY_B` also creates a collection and then populates it with the results of a specified query. For example:

```
HTMLDB_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B(
    p_collection_name => collection name,
    p_query           => your query );
```

CREATE_COLLECTION_FROM_QUERY_B offers significantly faster performance than CREATE_COLLECTION_FROM_QUERY by performing bulk SQL operations, but has the following limitations:

- No column value in the select list of the query can be more than 2,000 bytes. If a row is encountered that has a column value of more than 2,000 bytes, an error will be raised during execution.
- The MD5 checksum will not be computed for any members in the collection.

About the Parameter p_generate_md5

Use the p_generate_md5 flag to specify if the message digest of the data of the collection member should be computed. By default, this flag is set to NO. Use this parameter to check the MD5 of the collection member (that is, compare it with another member or see if a member has changed).

> **See Also:** "Determining Collection Status" on page 15-8 for more information on using the function GET_MEMBER_MD5

Truncating a Collection

Truncating a collection removes all members from the specified collection, but leaves the named collection in place. For example:

```
HTMLDB_COLLECTION.TRUNCATE_COLLECTION(
    p_collection_name => collection name );
```

Deleting a Collection

Deleting a collection deletes the collection and all of its members. Be aware that if you do not delete a collection, it will eventually be deleted when the session is purged. For example:

```
HTMLDB_COLLECTION.DELETE_COLLECTION (
    p_collection_name => collection name );
```

Deleting All Collections for the Current Application

Use the method DELETE_ALL_COLLECTIONS to delete all collections defined in the current application. For example:

```
HTMLDB_COLLECTION.DELETE_ALL_COLLECTIONS;
```

Deleting All Collections in the Current Session

Use the method DELETE_ALL_COLLECTIONS_SESSION to delete all collections defined in the current session. For example:

```
HTMLDB_COLLECTION.DELETE_ALL_COLLECTIONS_SESSION;
```

Adding Members to a Collection

When data elements (or members) are added to a collection, they are assigned a unique sequence ID. As you add members to a collection, the sequence ID will change in increments of 1 with the newest members having the largest ID.

You add new member to a collection using the function ADD_MEMBER. Calling this method returns the sequence ID of the newly added member. The following example demonstrates how to use the procedure ADD_MEMBER.

```
HTMLDB_COLLECTION.ADD_MEMBER(
    p_collection_name => collection name,
    p_c001          => [member attribute 1],
    p_c002          => [member attribute 2],
    p_c003          => [member attribute 3],
    p_c004          => [member attribute 4],
    p_c005          => [member attribute 5],
    p_c006          => [member attribute 6],
    p_c007          => [member attribute 7],
  ...
    p_c050          => [member attribute 50]);
    p_clob001       => [CLOB member attribute 1],
    p_generate_md5  => YES or NO);
```

The next example demonstrates how to use the function ADD_MEMBER. This function returns the sequence number assigned to the newly created member.

```
l_id := HTMLDB_COLLECTION.ADD_MEMBER(
    p_collection_name => collection name,
    p_c001          => [member attribute 1],
    p_c002          => [member attribute 2],
    p_c003          => [member attribute 3],
    p_c004          => [member attribute 4],
    p_c005          => [member attribute 5],
    p_c006          => [member attribute 6],
    p_c007          => [member attribute 7],
  ...
    p_c050          => [member attribute 50]);
    p_clob001       => [CLOB member attribute 1],
    p_generate_md5  => YES or NO);
```

You can also add new members (or an array of members) to a collection using the method ADD_MEMBERS. This method raises an exception if the specified collection does not exist with the specified name of the current user and in the same session. Also any attribute exceeding 4,000 characters will be truncated to 4,000 characters. The number of members added is based on the number of elements in the first array. For example:

```
HTMLDB_COLLECTION.ADD_MEMBERS(
    p_collection_name => collection name,
    p_c001          => member attribute array 1,
    p_c002          => member attribute array 2,
    p_c003          => member attribute array 3,
    p_c004          => member attribute array 4,
    p_c005          => member attribute array 5,
    p_c006          => member attribute array 6,
    p_c007          => member attribute array 7,
    ...
    p_c050          => member attribute array 50);
    p_generate_md5  => YES or NO);
```

About the Parameters p_generate_md5 and p_clob001

Use the p_generate_md5 flag to specify if the message digest of the data of the collection member should be computed. By default, this flag is set to NO. Use this

parameter to check the MD5 of the collection member (that is, compare it with another member or see if a member has changed).

Use `p_clob001` for collection member attributes which exceed 4,000 characters.

> **See Also:** "Determining Collection Status" on page 15-8 for more information on using the function `GET_MEMBER_MD5`

Updating Collection Members

You can update collection members by calling `UPDATE_MEMBER` and referencing the desired collection member by its sequence ID. This procedure replaces an entire collection member, not individual member attributes. This procedure raises an exception if the named collection does not exist. For example:

```
HTMLDB_COLLECTION.UPDATE_MEMBER (
    p_collection_name => collection name,
    p_seq             => member sequence number,
    p_c001            => member attribute 1,
    p_c002            => member attribute 2,
    p_c003            => member attribute 3,
    p_c004            => member attribute 4,
    p_c005            => member attribute 5,
    p_c006            => member attribute 6,
    p_c007            => member attribute 7,
    ...
    p_c050            => member attribute 50);
    p_clob001       => [CLOB member attribute 1],
```

Use `p_clob001` for collection member attributes which exceed 4,000 characters.

If you wish to update a single attribute of a collection member, use `UPDATE_MEMBER_ATTRIBUTE`. Calling this procedure raises an exception if the named collection does not exist. For example:

```
HTMLDB_COLLECTION.UPDATE_MEMBER_ATTRIBUTE(
    p_collection_name => collection name,
    p_seq             => member sequence number,
    p_c001            => member attribute 1,
    p_c002            => member attribute 2,
    p_c003            => member attribute 3,
    p_c004            => member attribute 4,
    p_c005            => member attribute 5,
    p_c006            => member attribute 6,
    p_c007            => member attribute 7,
    ...
    p_c050            => member attribute 50);
    p_clob_number     => number of CLOB attribute to be updated, <-- Only valid
value for now is 1
    p_clob_value      => new CLOB attribute value);
```

Deleting a Collection Member

You can delete a collection member by calling `DELETE_MEMBER` and referencing the desired collection member by its sequence ID. For example:

```
HTMLDB_COLLECTION.DELETE_MEMBER(
    p_collection_name => collection name,
    p_seq             => member sequence number);
```

Be aware that this procedure leaves a gap in the sequence IDs in the specified collection. Also, calling this procedure results in an error if the named collection does not exist.

You can also delete all members from a collection by when an attribute matches a specific value. For example:

```
HTMLDB_COLLECTION.DELETE_MEMBERS(
    p_collection_name => collection name,
    p_attr_number     => number of attribute used to match for the specified
                         attribute value for deletion,
    p_attr_value      => attribute value of the member attribute used to
                         match for deletion);
```

Be aware that this procedure also leaves a gap in the sequence IDs in the specified collection. Also, this procedure raises an exception if:

- The named collection does not exist
- The specified attribute number is outside the range of 1 to 50, or is in invalid

If the supplied attribute value is null, then all members of the named collection will deleted.

Determining Collection Status

The `p_generate_md5` parameter determines whether the MD5 message digests are computed for each member of a collection. The collection status flag is set to `FALSE` immediately after you create a collection. If any operations are performed on the collection (such as add, update, truncate, and so on), this flag is set to `TRUE`.

You can reset this flag manually by calling `RESET_COLLECTION_CHANGED`. For example:

```
HTMLDB_COLLECTION.RESET_COLLECTION_CHANGED (
    p_collection_name => collection name)
```

Once this flag has been reset, you can determine if a collection has changed by calling `COLLECTION_HAS_CHANGED`. For example:

```
l_changed := HTMLDB_COLLECTION.COLLECTION_HAS_CHANGED(
  p_collection_name => collection_name);
```

When you add a new member to a collection, an MD5 message digest is computed against all 50 attributes and the CLOB attribute if the `p_generated_md5` parameter is set to `YES`. You can access this value from the MD5_ORIGINAL column of the view `HTMLDB_COLLECTION` using the function `GET_MEMBER_MD5`. For example:

```
HTMLDB_COLLECTION.GET_MEMBER_MD5 (
    p_collection_name => collection name,
    p_seq             => member sequence number );
    RETURN VARCHAR2;
```

Merging Collections

You can merge members of collection with values passed in a set of arrays. By using the argument `p_init_query`, you can create a collection from the supplied query. For example:

```
HTMLDB_COLLECTION.MERGE_MEMBERS
p_collection_name => collection_name
```

Be aware, however, that if the collection exists, the following occurs:

- Rows in the collection (not in the arrays) will be deleted
- Rows in the collection and in the arrays will be updated
- Rows in the array and not in the collection will be inserted

Any attribute value exceeding 4,000 characters will be truncated to 4,000 characters. Table 15–1 describes the available arguments you can use when merging collections.

Table 15–1 Available Arguments for Merging Collections

Argument	Description
`p_c001`	Array of first attribute values to be merged. Maximum length can be 4,000 characters. If the maximum length is greater, it will be truncated to 4,000 characters. The count of elements in the P_C001 PL/SQL table is used as the total number of items across all PL/SQL tables. For example, if P_C001.count = 2 and P_C002.count = 10, only 2 members will be merged. Be aware that if P_C001 is null, an application error will be raised.
`p_c0xx`	Attribute of XX attributes values to be merged. Maximum length can be 4,000 characters. If the maximum length is greater, it will be truncated to 4,000 characters.
`p_collection_name`	Name of the collection. **See Also:** "About Collection Naming" on page 15-4
`p_null_index`	Use this argument to identify rows the merge function should ignore. This argument identifies an row as null. Null rows are automatically removed from the collection. Use `p_null_index` in conjunction with.
`p_null_value`	Use this argument in conjunction with the `p_null_index`. Identifies the null value. If used this value cannot be null. A typical value for this argument is 0.
`p_init_query`	Use the query defined by this argument to create a collection if the collection does not exist.

Managing Collections

You can use the following utilities to manage collections.

Obtaining a Member Count

Use `COLLECTION_MEMBER_COUNT` to return the total count of all members in a collection. Be aware that this count does not imply the highest sequence in the collection. For example:

```
l_count := HTMLDB_COLLECTION.COLLECTION_MEMBER_COUNT (
  p_collection_name => collection name );
```

Resequencing a Collection

Use `RESEQUENCE_COLLECTION` to resequence a collection to remove any gaps in sequence IDs while maintaining the same element order. For example:

```
HTMLDB_COLLECTION.RESEQUENCE_COLLECTION (
   p_collection_name => collection name )
```

Verifying Whether a Collection Exists

Use COLLECTION_EXISTS to determine if a collection exists. For example:

```
l_exists := HTMLDB_COLLECTION.COLLECTION_EXISTS  (
  p_collection_name => collection name );
```

Adjusting Member Sequence ID

You can adjust the sequence ID of a specific member within a collection by moving the ID up or down. When you adjust a sequence ID, the specified ID is exchanged with another one. For example, if you were to move the ID 2 up, 2 would become 3 and 3 would become 2.

Use MOVE_MEMBER_UP to adjust a member sequence ID up by one. Alternately, use MOVE_MEMBER_DOWN to adjust a member sequence ID down by one. For example:

```
HTMLDB_COLLECTION.MOVE_MEMBER_DOWN(
    p_collection_name => collection name,
    p_seq             => member sequence number);
```

Be aware that while using either of these methods an application error displays:

- If the named collection does not exist for the current user in the current session
- If the member specified by sequence ID p_seq does not exist

However, an application error will not be returned if the specified member already has the highest or lowest sequence ID in the collection (depending on whether you are calling MOVE_MEMBER_UP or MOVE_MEMBER_DOWN).

Sorting Collection Members

Use SORT_MEMBERS to reorder members of a collection by the column number. This method not only sorts the collection by a particular column number, but it also reassigns the sequence IDs for each member to remove gaps. For example:

```
HTMLDB_COLLECTION.SORT_MEMBERS(
    p_collection_name       => collection name,
    p_sort_on_column_number => column number to sort by);
```

Clearing Collection Session State

By clearing the session state of a collection, you remove the collection members. A shopping cart is a good example of when you might need to clear collection session state. When a user requests to empty his or her cart and start again, you would need to clear the session state for a collection. You can remove session state of a collection by calling the CREATE_OR_TRUNCATE_COLLECTION method or by using f?p syntax.

Calling CREATE_OR_TRUNCATE_COLLECTION deletes the existing collection and then recreates it. For example:

```
HTMLDB_COLLECTION.CREATE_OR_TRUNCATE_COLLECTION(
    p_collection_name       => collection name,
```

You can also use the sixth f?p syntax argument to clear session state. For example:

```
f?p=App:Page:Session::NO:1,2,3,collection name
```

See Also: "Understanding URL Syntax" on page 6-14

Running Background PL/SQL

You can use the HTMLDB_PLSQL_JOB package to run PL/SQL code in the background of your application. This is an effective approach for managing long running operations that do not need to complete in order for a user to continue working with your application.

Topics in this section include:

- Understanding the HTMLDB_PLSQL_JOB Package
- About System Status Updates
- Using a Process to Implement Background PL/SQL

Understanding the HTMLDB_PLSQL_JOB Package

HTMLDB_PLSQL_JOB is a wrapper package around DBMS_JOB functionality offered in the Oracle database. Be aware that the HTMLDB_PLSQL_JOB package only exposes that functionality which is necessary to run PL/SQL in the background. The following is a description of the HTMLDB_PLSQL_JOB package.

```
SQL> DESC HTMLDB_PLSQL_JOB
FUNCTION JOBS_ARE_ENABLED RETURNS BOOLEAN
PROCEDURE PURGE_PROCESS
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_JOB                          NUMBER                  IN
FUNCTION SUBMIT_PROCESS RETURNS NUMBER
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_SQL                          VARCHAR2                IN
 P_WHEN                         VARCHAR2                IN     DEFAULT
 P_STATUS                       VARCHAR2                IN     DEFAULT
FUNCTION TIME_ELAPSED RETURNS NUMBER
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_JOB                          NUMBER                  IN
PROCEDURE UPDATE_JOB_STATUS
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 P_JOB                          NUMBER                  IN
 P_STATUS                       VARCHAR2                IN
 P_DESC
```

Table 15–1 describes the functions available in the HTMLDB_PLSQL_JOB package.

Table 15–2 HTMLDB_PLSQL_JOB Package Available Functions

Function	Description
SUBMIT_PROCESS	Use this procedure to submit background PL/SQL. This procedure returns a unique job number. Since you can use this job number as a reference point for other procedures and functions in this package, it may be useful to store it in your own schema.
UPDATE_JOB_STATUS	Call this procedure to update the status of the currently running job. This procedure is most effective when called from the submitted PL/SQL.
TIME_ELAPSED	Use this function to determine how much time has elapsed since the job was submitted.

Table 15–2 (Cont.) HTMLDB_PLSQL_JOB Package Available Functions

Function	Description
JOBS_ARE_ENABLED	Call this function to determine whether or not that database is currently in a mode which supports submitting jobs to the HTMLDB_PLSQL_JOB package.
PURGE_PROCESS	Call this procedure to clean up submitted jobs. Submitted jobs stay in the HTMLDB_PLSQL_JOBS view until either Oracle HTML DB cleans out those records, or you call PURGE_PROCESS to manually remove them.

You can view all jobs submitted to the HTMLDB_PLSQL_JOB package using the HTMLDB_PLSQL_JOBS view. The following is the description of HTMLDB_PLSQL_JOBS view.

```
SQL> DESCRIBE HTMLDB_PLSQL_JOBS
 Name                              Null?    Type
 --------------------------------- -------- ----------------------------
 ID                                         NUMBER
 JOB                                        NUMBER
 FLOW_ID                                    NUMBER
 OWNER                                      VARCHAR2(30)
 ENDUSER                                    VARCHAR2(30)
 CREATED                                    DATE
 MODIFIED                                   DATE
 STATUS                                     VARCHAR2(100)
 SYSTEM_STATUS                              VARCHAR2(4000)
 SYSTEM_MODIFIED                            DATE
 SECURITY_GROUP_ID                          NUMBER
```

Table 15–3 describes the columns available in HTMLDB_PLSQL_JOBS view.

Table 15–3 HTMLDB_PLSQL_JOBS View Columns

Name	Description
ID	An unique identifier for each row.
JOB	The job number assigned to each submitted PL/SQL job. The HTMLDB_PLSQL_JOB.SUBMIT_PROCESS function returns this value. This is also the value you pass into other procedures and functions in the HTMLDB_PLSQL_JOB package.
FLOW_ID	The application from which this job was submitted.
OWNER	The database schema that owns the application. This identifies what schema will parse this code when DBMS_JOB runs it.
ENDUSER	The end user (that is, who logged into the application) that caused this process to be submitted.
CREATED	The date when the job was submitted.
MODIFIED	The date when the status was modified.
STATUS	The user defined status for this job. Calling HTMLDB_PLSQL_JOB.UPDATE_JOB_STATUS updates this column.
SYSTEM_STATUS	The system defined status for this job.
SYSTEM_MODIFIED	The date when the system status was modified.

Table 15–3 (Cont.) HTMLDB_PLSQL_JOBS View Columns

Name	Description
SECURITY_GROUP_ID	The unique ID assigned to your workspace. Developers can only see jobs submitted from their own workspace.

About System Status Updates

Submitted jobs can contain any of the following system status settings:

- **SUBMITTED.** Indicates the job has been submitted, but has not yet started. DBMS_JOB does not guarantee immediate starting of jobs.
- **IN PROGRESS.** Indicates that DBMS_JOB has started the process.
- **COMPLETED.** Indicates the job has finished.
- **BROKEN (sqlcode) sqlerrm.** Indicates there was a problem in your job that resulted in an exception. The SQL code and SQL Error Message for the exception should be included in the system status. Review this information to determine what went wrong.

Using a Process to Implement Background PL/SQL

The simplest way to implement the HTMLDB_PLSQL_JOB package is to create a page process that specifies the process type PLSQL DBMS JOB. By selecting this process type, Application Builder will submit the PL/SQL code you specify as a job. Since you are not calling the function directly, you can use the built-in substitution item APP_JOB to determine the job number of any jobs you submit.

The following example runs a PL/SQL job in the background for testing and explanation.

```
001  BEGIN
002    FOR i IN 1 .. 100 LOOP
003      INSERT INTO emp(a,b) VALUES (:APP_JOB,i);
004      IF MOD(i,10) = 0 THEN
005        HTMLDB_PLSQL_JOB.UPDATE_JOB_STATUS(
006          P_JOB     => :APP_JOB,
007          P_STATUS  => i || 'rows inserted');
008      END IF;
009      HTMLDB_UTIL.PAUSE(2);
010    END LOOP;
011  END;
```

In this example, note that:

- Lines 002 to 010 run a loop that inserts 100 records into the emp table.
- APP_JOB is referenced as a bind variable inside the VALUES clause of the INSERT, and specified as the P_JOB parameter value in the call to UPDATE_JOB_STATUS.
- APP_JOB represents the job number which will be assigned to this process as it is submitted to HTMLDB_PLSQL_JOB. By specifying this reserved item inside your process code, it will be replaced for you at execution time with the actual job number.
- Notice this example calls to UPDATE_JOB_STATUS every ten records, INSIDE the block of code. Normally, Oracle transaction rules dictate updates made inside code blocks will not be seen until the entire transaction is committed. The HTMLDB_PLSQL_JOB.UPDATE_JOB_STATUS procedure, however, has been implemented

in such a way that the update will happen regardless of whether or not the job succeeds or fails. This last point is important for two reasons:

1. Even if your status reads "100 rows inserted," it does not mean the entire operation was successful. If an exception occurred at the time the block of code tried to commit, the `user_status` column of `HTMLDB_PLSQL_JOBS` would not be affected since status updates are committed separately.
2. These updates are performed autonomously. You can view the job status before the job has completed. This gives you the ability to display status text about ongoing operations in the background as they are happening.

Implementing Web Services

Web services enable applications to interact with one another over the Web in a platform-neutral, language independent environment. In a typical Web services scenario, a business application sends a request to a service at a given URL by using the protocol over HTTP. The service receives the request, processes it, and returns a response. You can incorporate calls with external Web services in application developed in Oracle HTML DB

Web services in Oracle HTML DB are based on SOAP (the Simple Object Access Protocol). SOAP is a World Wide Web Consortium (W3C) standard protocol for sending and receiving requests and responses across the Internet. SOAP messages can be sent back and forth between a service provider and a service user in SOAP envelopes.

SOAP offers two primary advantages:

- SOAP is based on XML and therefore easy to use.
- SOAP messages are not blocked by firewalls since this protocol uses simple transport protocols such as HTTP.

Topics in this section include:

- Understanding Web Service References
- Creating a Web Service Reference
- Using the Web Service Reference Repository
- Creating an Input Form and Report on a Web Service
- Creating a Form on a Web Service
- Invoking a Web Service as a Process
- Editing a Web Service Process
- Viewing a Web Service Reference History

Note: The SOAP 1.1 specification is a W3C note. (The W3C XML Protocol Working Group has been formed to create a standard that will supersede SOAP.)

For more information on Simple Object Access Protocol (SOAP) 1.1 see:

`http://www.w3.org/TR/SOAP/`

Understanding Web Service References

To utilize Web services in Oracle HTML DB, you create a Web service reference using a wizard. Each Web service reference is based on a Web Services Description Language (WSDL) document that describes the target Web service. When you create a Web service reference, the wizard analyzes the WSDL and collects all the necessary information to create a valid SOAP message, including:

- The URL used to post the SOAP request over HTTP
- An Uniform Resource Identifier (URI) identifying the SOAP HTTP request
- Operations of the Web Service
- Input parameters for each operation
- Output parameters for each operation

Accessing the Web Service References Page

You manage Web service references on the Web Service References page.

To access the Web Service References page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Logic, select **Web Service References**.

   The Web Service References page appears.

Specifying an Application Proxy Server Address

If your environment requires a proxy server to access the Internet, you must specify a proxy server address on the Application Attributes page before you can create a Web service reference.

To specify a proxy address for an application:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears.
3. Click **Edit Attributes**.
4. Under Application Definition, enter the proxy server in Proxy Server.

Creating a Web Service Reference

When you create a Web service reference you need to decide how to locate the WSDL. You can locate a WSDL in two ways:

- By searching a UDDI registry
- by entering the URL to the WSDL document

A Universal Description, Discovery, and Integration (UDDI) registry is a directory where businesses register their Web services.

Creating a Web Service Reference by Searching a UDDI Registry

To create a new Web service by searching a UDDI registry:

1. Navigate to the Web Service References page. (See "Accessing the Web Service References Page" on page 15-15.)
2. Click **Create**.
3. When prompted whether to search a UDDI registry to find a WSDL, click **Yes**.
4. For UDDI Location you can either:
   - Enter a URL endpoint to a UDDI registry.
   - Click the **List** icon and select a UDDI registry.
5. Under Search for Services, specify whether to search for a business name or a service name.
   a. For Search Type, specify whether to search for a business name or a service name. You cannot search for both.
   b. In Name, enter the business name or service name to search for.
   c. Optionally indicate whether the search should be case-sensitive or an exact match. Use the percent (%) symbol as a wildcard character.
   d. Click **Search**.
   e. When the search results appears, make a selection and click **Next**.

   A summary page appears describing the selected Web service.
6. Review your selection and click **Next** to continue.

   The URL to the WSDL document displays in the WSDL Location field.
7. Click **Finish**.

The Web service reference is added to the Web Service References Repository.

Creating a Web Service Reference by Specifying a WSDL Document

To create a new Web service by specifying a URL to a specific WSDL document:

1. Navigate to the Web Service References page. (See "Accessing the Web Service References Page" on page 15-15.)
2. Click **Create**.
3. When prompted whether to search a UDDI registry to find a WSDL, click **No**.
4. In WSDL Location, enter the URL to the WSDL document.
5. Click **Finish**.

The Web service reference is added to the Web Service References Repository.

Using the Web Service Reference Repository

Web service references are stored in the Web Service Reference Repository.

To access the Web Service References Repository:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears.

3. Click **Shared Components**.

   The Shared Components page appears.

4. Under Logic, select **Web Service References**.

From the Web Service References Repository you can:

- Edit a reference by clicking the **Edit** icon.
- Test a reference by clicking the **Run** icon.
- View details about a reference, by clicking the reference name.

Testing a Web Service Reference

Once you have created a Web service reference you can test it on the Test Web Service Reference.

To test a Web service reference:

1. Navigate to the Web Service Repository. (See "Using the Web Service Reference Repository" on page 15-16.)
2. Click **Run** adjacent to the Web Service reference name.

   The Test Web Service Reference page appears. The Web service name and URL endpoint display at the top of the page.
3. From Operation, select an operation (that is, the method to be executed).
4. Under Input Parameters, enter the appropriate value.
5. Click **Test**.

   The message request and response appear at the bottom of the page.

Creating an Input Form and Report on a Web Service

The Create Form and Report on Web Service Wizard creates an input form, a submit button, and a report for displaying results. You can execute this wizard directly after creating the Web service reference, or by adding a new page.

Use this wizard when you expect a non-scalar result from the Web service. The Amazon Web service is a good example. This Web service returns many results based on the search criteria entered in an input form.

Creating a Form and Report Directly After Creating a Reference

To create a form and report directly after creating a Web Service Reference:

1. Create the Web service reference. (See "Creating a Web Service Reference" on page 15-15.)
2. Once the Web service references has been added, select **Create Form and Report on Web Service**.
3. For Web Service Reference and Operation, select the Web service reference and operation (that is, the method to be executed).
4. For Identify Page and Region Attributes, review the page and region attributes. If the page you specify does not exist, the wizard creates the page for you.
5. For Items for Input Parameters:

a. Identify which items to add to the form. To include an item, select **Yes** in the Create column. Otherwise, select **No**.

b. If necessary, edit the item label.

6. For Base Node:

a. In Temporary Result Set Name, enter a name for the collection that stores the Web service result.

b. For Result Tree to Report On, select the portion of the resulting XML document that contains the information you want to include in the report.

7. For Result Parameters to Display, select the parameters to be included in the report.

8. Click **Finish**.

Creating a Form and Report by Adding a New Page

If you have an existing Web service reference, you can create an input form and report by adding a new page.

To create a form and report by adding a new page:

1. Create the Web service reference. (See "Creating a Web Service Reference" on page 15-15.)

2. Create a new page. (See "Adding Additional Pages" on page 8-3.)

   In the Create Page Wizard:

a. Select **Page with Component**.

b. For Select Component Type, select **Form**.

c. For Create Page, select **Form and Report on Web Service**.

3. For Web Service Reference and Operation, select the Web service reference and operation (that is, the method to be executed).

4. For Identify Page and Region Attributes, review the page and region attributes. If the page you specify does not exist, the wizard creates the page for you.

5. For Items for Input Parameters:

a. Identify which items to add to the form. To include an item, select **Yes** in the Create column. Otherwise, select **No**.

b. If necessary, edit the item label.

6. For Base Node:

a. In Temporary Result Set Name, enter a name for the collection that stores the Web service result.

b. In Result Tree to Report On, select the portion of the resulting XML document that contains the information you want to include in the report.

7. For Result Parameters to Display, select the parameters to be included in the report.

8. Click **Finish**.

Creating a Form on a Web Service

The Create Form on Web Service wizard creates a form and a submit button. You can execute this wizard directly after creating the Web service reference, or from the Page Definition.

Use this wizard when you expect a scalar result from the Web service. A Web service that looks up a stock price is a good example since the input is a stock symbol and the output is the scalar value price.

Creating a Form Directly After Creating a Reference

To create a form directly after creating a Web Service Reference:

1. Create the Web service reference. (See "Creating a Web Service Reference" on page 15-15.)
2. Once the Web service references has been added, select **Create Form on Web Service**.
3. For Web Service Reference and Operation, select the Web service reference and operation (that is, the method to be executed).
4. For Identify Page and Region Attributes, review the page and region attributes. If the page you specify does not exist, the wizard creates the page for you.
5. For Items for Input Parameters:
   a. Identify which items to add. To include an item, select **Yes** in the Create column. Otherwise, select **No**.
   b. If necessary, edit the item label.
6. For Items for Output Parameters:
   a. Identify which items need to be added. To include an item, select **Yes** in the Create column. Otherwise, select **No**.
   b. If necessary, edit the item label.
7. Click **Finish**.

Creating a Form by Adding a New Page

If you have an existing Web service reference, you can create form by adding a new page.

To create a form by adding a new page:

1. Create the Web service reference. (See "Creating a Web Service Reference" on page 15-15.)
2. Create a new page. (See "Adding Additional Pages" on page 8-3.)

   In the Create Page Wizard:

   a. Select **Page with Component**.
   b. For Select Component Type, select **Form**.
   c. For Create Page, select **Form on Web Service**.
3. For Web Service Reference and Operation, select the Web service reference and operation (that is, the method to be executed).
4. For Identify Page and Region Attributes, review the page and region attributes. If the page you specify does not exist, the wizard creates the page for you.

5. For Items for Input Parameters:
   a. Identify which items need to be added. To include an item, select **Yes** in the Create column. Otherwise, select **No**.
   b. If necessary, edit the item label.
6. For Items for Output Parameters:
   a. Identify which items need to be added. To include an item, select **Yes** in the Create column. Otherwise, select **No**.
   b. If necessary, edit the item label.
7. Click **Finish**.

Invoking a Web Service as a Process

You can also implement a Web Service as a process on the page. Running the process submits the request to the service provider. You can then display the request results in report.

To invoke a Web Service as a process:

1. Create a new page. (See "Adding Additional Pages" on page 8-3.)

   In the Create Page Wizard:

   a. Select **Blank Page**.
   b. When prompted whether to use tabs, select **No**.
2. Navigate to the Page Definition. (See "Viewing a Page Definition" on page 7-11.)
3. Under Page Rendering, Processes, click the **Create** icon.

   The Create Page Processes Wizard appears.
4. From the process category, select **Web Services**.
5. Specify a process name, sequence, and processing point.
6. Select the Web service reference and operation (that is, the method to be executed).
7. Define the process. You can store the results in a collection or in items on the page by selecting options under Web Service Output Parameters.
   a. To store the results in a collection:
      – For Store Result in, select **Collection**.
      – Enter a name for the collection in the value field.
   b. To store the results in items on the page:
      – For Store Result in, select **Items**.
      – Enter the appropriate items value in the fields provided.
8. Click **Create Process**.

Displaying Web Service Results in a Report

To create a report in which to display Web Service request results:

1. Navigate to the Page Definition.
2. Under Regions, click the **Create** icon.

   The Create Region Wizard appears.

3. For the region type, select **Report**.
4. For the report implementation, select **Report on collection containing Web service result**.
5. On Identify Region Attributes, enter a region title and optionally edit the region attributes.
6. For Web Service Reference and Operation, select a Web service reference and an operation (that is, the method to be executed).
7. For Result Tree to Report On, select the portion of the resulting XML document that contains the information you want to include in the report.
8. For Result Parameters:
   a. In Temporary Result Set Name, enter a name for the collection that stores the Web service result.
   b. Select and deselect the appropriate parameters.
9. Click **Create SQL Report**.

Editing a Web Service Process

Once you create a process of type Web service, you can map input parameters to a static value (for example to pass a key) by editing the Web service process.

To edit a Web service process:

1. Create a Web service process. (See "Invoking a Web Service as a Process" on page 15-20.)
2. Navigate to the Page Definition containing the Web service process.
3. Select the process name.

   The Edit Page Process page appears.
4. Scroll down to Web Service Input Parameters.
5. To map an input parameter to a static value:
   a. Scroll down to Web Service Input Parameters.
   b. Enter a value in the Value field adjacent to the appropriate parameter name.
6. Click **Apply Changes**.

Viewing a Web Service Reference History

The Web Services History displays changes to Web service references for the current application by application ID, Web service references name, developer, and date.

To view a history of Web service reference changes:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.

   Application Builder appears.
3. Click **Shared Components**.

   The Shared Components page appears.
4. Under Logic, select **Web Service References**.

5. Click **History**.

Managing User Preferences

You can use preferences to set the session state for a specific user. Once set, these preferences can only be removed by an Oracle HTML DB administrator. You can set user preferences by creating a page process, by the calculation of a preference Item Source Value, or programatically using the PL/SQL API.

Topics in this section include:

- Viewing User Preferences
- Setting User Preferences
- Resetting User Preferences Manually
- Resetting Preferences Using a Page Process

Viewing User Preferences

You view user preferences for a specific user on the Session State Management page.

To view user preferences for a specific user:

1. From the Oracle HTML DB home page select the **Administration** tab.
2. Under Administration Services, click **Manage Users** and then **Session State**.

   The Session State Management page appears.
3. Click **Report preferences for users**.
4. Type a username in the field provided and click **Go**.

> **See Also:** "Managing Session State and User Preferences" on page 13-5 for more information on using the Session State Management page

Setting User Preferences

You can set user preferences within your application through the creation of a page process, by creating a preference item, or programatically.

Topics in this section include:

- Setting User Preferences Using a Page Process
- Setting the Source of an Item Based on a User Preference
- Setting User Preferences Programatically

Setting User Preferences Using a Page Process

To set user preference values by creating a page process:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Page Processes, click the **Create** icon.

   The Create Page Computation Wizard appears.
3. Specify a process name, sequence, and processing point.
4. From Type, select one of the following:

- **Set Preference to value of item**
- **Set Preference to value of item if item is not NULL**

5. Specify the preference value in the field provided using the format:

   ```
   PreferenceName:Item
   ```

6. Click **Page Items** to see a list of available items.
7. Follow the on-screen instructions

Setting the Source of an Item Based on a User Preference

You can set the source of an item based on a user preference by defining the item source type as Preference.

To define the source of item based on a user preference:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Item, click the **Create** icon.

   The Create Page Computation Wizard appears.
3. Specify the Item Name and Display Position Attributes and click **Next**.
4. Specify the Item Attributes click **Next**.
5. From the Item Source list, select **Preferences**.
6. In Item Source Value, enter the name of the preference.
7. Follow the on-screen instructions

Setting User Preferences Programatically

To set or reference user preferences programatically, you must use a PL/SQL API. User level caching is available programmatically. You can use the `set_preference` function to set a user level preference called `NAMED_PREFERENCE`. For example:

```
HTMLDB_UTIL.SET_PREFERENCE(
 p_preference=>'NAMED_PREFERENCE',
 p_value =>:ITEM_NAME);
```

You can reference the value of a user preference using the function `GET_PREFERENCES`. For example:

```
NVL(HTMLDB_UTIL.GET_PREFERENCE('NAMED_PREFERENCE'),15)
```

In the previous example, the preference would default to the value 15 if the preference contained no value.

> **See Also:** "GET_PREFERENCE Function" on page 17-16 and "SET_PREFERENCE Procedure" on page 17-24

Resetting User Preferences Manually

You can manually purge user preferences for a specific user.

To manually purge preferences for a specific user:

1. From the Oracle HTML DB home page select **Administration** tab.
2. Under Administration Services, click **Manage Users** and then **Session State**.

The Session State Management page appears.

3. Click **Purge preferences for a selected user**.
4. Specify a user and follow the on-screen instructions.

> **See Also:** "Managing Session State and User Preferences" on page 13-5 for more information on using the Session State Management page

Resetting Preferences Using a Page Process

You can reset user preferences by creating a page process and selecting the process type Reset Preferences.

To reset user preferences using a page process:

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Page Processes, click the **Create** icon.

   The Create Page Computation Wizard appears.
3. Specify a process name, sequence, and processing point.
4. From Type, select **Reset Preferences**.
5. Follow the on-screen instructions

16

Managing Globalization

This section describes how to translate an application built-in Oracle HTML DB.

This section contains the following topics:

- About Translating an Application and Globalization Support
- Specifying the Primary Language for an Application
- Understanding the Translation Process
- Translating Messages Used in PL/SQL Procedures
- Translating Data that Supports List of Values
- About Oracle HTML DB Globalization Codes

About Translating an Application and Globalization Support

In Oracle HTML DB you can develop applications that can run concurrently in different languages. A single Oracle database instance and Oracle HTML DB can support multiple database sessions customized to support different language.

In general, translating an Oracle HTML DB application involves the following steps:

- Map primary and target application IDs
- Seed and export text to a file for translation
- Translate the text in the file
- Apply and publish the translated file

See Also: "Understanding the Translation Process" on page 16-5

Topics in this section include:

- About Language Identification
- Rule for Translating Applications in Oracle HTML DB
- How Translated Applications Are Rendered
- About Translatable Components

About Language Identification

After you create an application, you specify a language preference on the Edit Application Attributes page. Under Globalization, you select a primary application language and select how the HTML DB engine determines the application language.

You can specify to have the application language based on the user's browser language preference, an application preference, or an item preference.

> **See Also:** "Specifying the Primary Language for an Application" on page 16-4

Rule for Translating Applications in Oracle HTML DB

When using translated applications in Oracle HTML DB, use the following rules to determine which translated version to use:

- Look for an exact match between the user language preference and the language code of the translated application
- Look for a truncated match. That is, see if the language and locale exist. For example, if the user language preference is `en-us` and the translated version of `en-us` does not exist, look for a translated application that has the language code `en`
- Use the primary application

For example, suppose you create an application with the primary language of German, `de`, and you create a translated version of the application with a language code of `en-us`. Users accessing this application with a browser language of `en-us` execute the English `en-us` version of the application. Users accessing the application with a browser language of `en-gb` view the application in the application primary language. In this example, these users see the application in German, which is the application's primary language. For this example, you should create the translated English version using language code `en` to encompass all variations of `en`.

How Translated Applications Are Rendered

Once Oracle HTML DB determines the language for an application, the HTML DB engine alters the database language for a specific page request. It then looks for a translated application in the appropriate language. If the HTML DB engine finds that language, it render the application using that definition. Otherwise, it renders the application in the base (or primary) application language.

Note that the text that displays within an application is not translated on the fly. Oracle HTML DB dynamically collects page attributes from either a base language application definition or an alternative application definition.

> **See Also:** "About Dynamic Translation Text Strings" on page 16-3 and "Translating Data that Supports List of Values" on page 16-11

About Translatable Components

When you build an application in Oracle HTML DB, you define a large number of declarative attributes such as field labels, region headings, page header text, and so on. Using the steps described in this chapter, you can make all the application definition attributes within your application translatable.

About Shortcuts that Support Translatable Messages

Oracle HTML DB includes two shortcuts type that enable you to reference translatable messages:

- **Message** - Use this shortcut to reference a translatable message at runtime. Note that the name of the shortcut must match the corresponding message name. At

runtime, the name of the shortcut expands to the text of the translatable message for the current language.

- **Message with JavaScript Escaped Single Quotes** - Use this shortcut to reference a shortcut inside of JavaScript literal string and reference a translatable message at runtime. This shortcut defines a text string. When the shortcut is referenced, it escapes the single quotes required for JavaScript.

See Also: "Utilizing Shortcuts" on page 8-48

About Messages

If your application includes PL/SQL regions or PL/SQL processes, you may need to translate any generated HTML or text. Within Oracle HTML DB these types of generated HTML and text are called "messages." You can define all messages and translate them on the Translatable Messages page. You can use the `HTMLDB_LANG.MESSAGE` API to translate text strings from PL/SQL stored procedures, functions, triggers, packaged procedures and functions.

See Also: "Translating Messages Used in PL/SQL Procedures" on page 16-10

About Dynamic Translation Text Strings

Dynamic translations are used for database data that needs to be translated at runtime. For example, you might use a dynamic translation to translate a list of values based on a database query. A dynamic translation consists of a "translate-from" language string, a language code, and a "translate-to" string. You can also use the `HTMLDB_LANG.LANG` API to retrieve dynamic translations programmatically.

See Also: "Translating Data that Supports List of Values" on page 16-11

About Translating Templates

By default, templates in Oracle HTML DB are not translatable and therefore not included in the generated translation file. Generally, templates do not and should not contain translatable text. However, if you need to mark a template as translatable, mark the Translatable check box on the Edit Page Template page.

To identify a template as translatable:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. On the Application Builder home page, click **Shared Components**.
4. Under User Interface, select **Templates**.

   The Templates page appears.
5. Locate the template you wish to edit and select the template name.
6. Under Template Identification, select **Translatable**.

One way to include translatable text at the application level is to define the translatable text using static substitution strings. Since application level attributes are translated any text defined as a static substitution strings will be included in the generated translation file.

See Also:

- "Editing Templates" on page 9-19
- "Static Substitution Strings" on page 7-9

Specifying the Primary Language for an Application

Globalization attributes specify how the HTML DB engine determines the primary language of an application.

To edit Globalization attributes:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. Click the **Edit Attributes** icon.

   The Edit Application Attributes page appears.
4. Scroll down to Globalization.
5. From **Application Primary Language**, select the language in which the application is being developed.
6. From **Application Language Derived From**, specify how the HTML DB engine determines (or derives) the application language. Available options are described in Table 16–1.

Table 16–1 Application Language Derived From Options

Option	Description
No NLS (Application not translated)	Select this option if the application will not be translated.
Use Application Primary Language	Determines the application primary language based on the Application Primary Language attribute (see step 5).
Browser (use browser language preference)	Determines the application primary language based on the user's browser language preference.
Application Preference (use FSP_LANGUAGE_PREFERENCE)	Determines the application primary language based a value defined using the `HTMLDB_UTIL.SET_PREFERENCE` API. Select this option to maintain the selected language preference across multiple log ins. **See Also:** "SET_PREFERENCE Procedure" on page 17-24
Item Preference (use item containing preference)	Determines based on an application level item called `FSP_LANGUAGE_PREFERENCE`. Using this option requires Oracle HTML DB to determine the appropriate language preference every time the user logs in.

See Also:

- "Editing Application Attributes" on page 7-4
- "Globalization" on page 7-7
- "About Oracle HTML DB Globalization Codes" on page 16-13

Using Format Masks for Items

The HTML DB engine applies Globalization settings for each rendered page. This default behavior can impact the display of certain items such as numbers and dates.

For example, suppose your application determines the application language based on the user's browser language preference. If the HTML DB engine determines the users's browser language preference is French, it displays dates and numbers in a format that conforms to French standards. You can override this default behavior and explicitly control how items display by applying a format mask. You apply a format mask by making a selection from the Display As list:

- When you create the item
- After you create the item by editing the item attributes

To edit item attributes

1. Navigate to the appropriate Page Definition. (See "Viewing a Page Definition" on page 7-11.)
2. Under Items, drill down on the item name.

   The Edit Page Item page appears.
3. Under Identification, make a select from the Display As list.

> **See Also:** "Items" on page 7-21 for more information on item attributes.

Translating Applications for Multibyte Languages

If your application needs to run in several languages (such as Chinese or Japanese) simultaneously, you should consider configuring your database with a character set to support all of the languages. The same character set has to be configured in the corresponding DAD (Data Access Description) in mod_plsql. UTF8 and AL32UTF8 are the character sets you can use to support almost all languages around the world.

> **See Also:** *Oracle Database Globalization Support Guide*

Understanding the Translation Process

To translate an application developed in Oracle HTML DB, you must map the primary and target application IDs, seed and export text to a translation file, translate the text, and then apply and publish the translation file.

Topics in this section include:

- Navigating to the Translate Application Page
- Mapping Primary and Target Application IDs
- Seeding and Exporting Text to a Translation File
- Translating the XLIFF File
- Uploading and Publishing a Translated XLIFF Document

Navigating to the Translate Application Page

You perform the translation process on the Translate Application page.

To navigate to the Translate Application page:

1. Navigate to the Workspace home page.
2. From the Applications list, select an application.
3. Click the **Shared Components** icon.
4. Under Translations, select **Translation Services**.

   The Translate Application page appears.

Mapping Primary and Target Application IDs

The first step in translating an application is to map the primary and target application IDs. The primary application is the application to be translated. The target application is the resulting translated application.

To map the primary and target application IDs:

1. Navigate to the Translate Application page. (See "Navigating to the Translate Application Page" on page 16-5.)
2. On the Translate Application page, select **Map your primary language application to a translated application ID**.

   The Application Mappings page appears.
3. Click **Create**.
4. On the Translation Application Mapping page:
   - In Translation Application, type a numeric application ID to identify the target application. The translated application ID must be an integer and cannot end in zero.
   - From Translation Application Language Code, select the language you are translating to.
   - In Image Directory, enter the directory where images will be obtained.

     This attribute determines the virtual path for translated images. For example, if your primary language application had an image prefix of `'/images/'`, you could define additional virtual directories for other languages such as `'/images/de/'` for German or `'/images/es/'` for Spanish.
5. Click **Create**.

Seeding and Exporting Text to a Translation File

The second step in translating an application is to seed the translation table and then export the translation text to a translation file.

Seeding Translatable Text

To translate a application, you must seed the translations. Seeding the translation copies all translatable text into a translation text repository. Once you have seeded the application and specific language in the translation text repository, you can then generate and export an XLIFF file for translation.

The seeding process keeps your primary language application synchronized with the translation text repository. You should run the seed process any time your primary language application changes.

To seed translatable text:

1. Navigate to the Translate Application page. (See "Navigating to the Translate Application Page" on page 16-5.)
2. On the Translate Application page, select **Seed and export the translation text of your application into a translation file**.
3. From Language Mapping, select the appropriate primary and target application ID map.
4. Click **Seed Translatable Text**.

   The XLIFF Export page appears.

> **Note:** XML Localization Interchange File Format (XLIFF) is a XML-based format for exchanging localization data. For more information on the XLIFF, or to view the XLIFF specification see:
>
> `http://www.xliff.org`

Exporting Text to a Translation File

Once you have seeded translatable text, a status box displays at the top of the XLIFF Export page indicating the total number of attributes that may require translation as well as the number of:

- Existing updated attributes that may require translation
- New attributes that may require translation
- Purged attributes that no longer require translation

You can use this information to determine whether you need to export translatable text for an entire application or just a specific page.

The XLIFF Export page is divided into two sections. Use the upper half of the page to export translatable text for an entire application (that is, all pages, lists of values, messages, and so on). Use the lower section to export translatable text for a specific page.

To export translatable text for an entire application:

1. Seed the translatable text as described in the previous procedure, "Seeding Translatable Text" on page 16-6.
2. Under **Step 2, Export XLIFF**:
   - From Application, select the appropriate primary and target application ID map
   - Specify whether to include XLIFF target elements
   - Under Export, specify what translation text is included in your XLIFF file
   - Click **Export XLIFF for Application**
3. Follow the on-screen instructions.

To export translatable text for a specific page:

1. Seed translatable text as described in "Seeding Translatable Text" on page 16-6.
2. Under **Export XLIFF for specific Page**:
   - From Application, select the appropriate primary and target application ID map

- Specify whether to include XLIFF target elements
- Under Export, specify what translation text is included in your XLIFF file
- Click **Export XLIFF for Page**

3. Follow the on-screen instructions.

About Include XLIFF Target Elements When Oracle HTML DB generates an XLIFF document, each document contains multiple translation units. Each translation unit consists of a source element and a target element. The XLIFF document can be generated with both the source and target elements for each translation unit. You have the option of generating a file containing only source elements. The updated translations will be applied from the target elements of translation units.

About Export Use **Export** to specify what translation text is included in your XLIFF file. Select **All translatable elements** to include all translation text for an application. In contrast, select **Only those elements requiring translation** to include only new elements that have not yet been translated. **Only those elements requiring translation** produces an XLIFF file containing new or modified translation units. Also, if translation units were intentionally not previously translated (that is, the source of the translation element equals the target of the translation element), those translation units will also be included in the file.

Translating the XLIFF File

After you export a translatable file to XLIFF format, you can translate it into the appropriate languages. Since XLIFF is an open standard XML file for exchanging translation, most translation vendors should support it. Oracle HTML DB only supports XLIFF files encoded in UTF-8 character sets. In other words, it exports XLIFF files for translation in UTF-8 and assumes that the translated XLIFF files will be in the same character set.

Translation is a time consuming task. Oracle HTML DB supports incremental translation so that application development can be done in parallel with the translation. An XLIFF file can be translated and uploaded to Oracle HTML DB even when only part of the XLIFF file is translated. For strings that have no translation in the corresponding translated application, Oracle HTML DB uses the corresponding ones in the primary language.

> **See Also:** For more information on the XLIFF, or to view the XLIFF specification see:
>
> `http://www.xliff.org`

Uploading and Publishing a Translated XLIFF Document

Once your XLIFF document has been translated, the next step is to upload it back into Oracle HTML DB.

To upload a translated XLIFF document:

1. Navigate to the Translate Application page. (See "Navigating to the Translate Application Page" on page 16-5.)
2. On the Translate Application page, select **Apply your translation file and publish**.
3. Click **Upload XLIFF**.

4. On the XLIFF Upload page:
   - Specify a title
   - Enter a description
   - Click **Browse** and locate the file to be uploaded
   - Click **Upload XLIFF File**

   The uploaded document appears in the XLIFF Files repository.

Once you upload an XLIFF document, the next step is to apply the XLIFF document and then publish the translated application. When you apply an XLIFF document, the HTML DB engine parses the file and then updates the translation tables with the new translatable text.

Publishing your application creates a copy of the base language application, substituting the translated text strings from your translations table. This published application can then be used to render your application in alternate languages.

Remember that in order to run an application in an alternative language, you need to run it with Globalization settings that will cause an alternative language version to display. For example, if the language is derived from the browser language, you must set the browser language to the same language as the translated application.

> **See Also:** "Specifying the Primary Language for an Application" on page 16-4

To apply and publish a translated XLIFF document:

1. Navigate to the Translate Application page. (See "Navigating to the Translate Application Page" on page 16-5.)
2. On the Translate Application page, select **Apply your translation file and publish**.
3. In the XLIFF Files repository, click the **View** icon adjacent to the document you wish to publish.
4. From Apply to, select the appropriate primary and target application ID map.
5. Click **Apply XLIFF Translation File**.
6. Click **Publish Application**.

To delete an uploaded XLIFF document:

1. Navigate to the Translate Application page. (See "Navigating to the Translate Application Page" on page 16-5.)
2. On the Translate Application page, select **Apply your translation file and publish**.
3. In the XLIFF Files repository, select the check box to the left of the document title.
4. Click **Delete Checked**.

You should verify the existence of the translated application once it is published. Translated applications do not display in the Available Applications list on the Application Builder home page. Instead, use the Application Navigate list on the left side of the page.

Note that in order for a translated application to appear in Application Builder, you need to make sure the you have correctly configured the application Globalization attributes.

See Also: "Specifying the Primary Language for an Application" on page 16-4

Translating Messages Used in PL/SQL Procedures

If your application includes PL/SQL regions or PL/SQL processes or calls PL/SQL package, procedures, or functions, you may need to translate generated HTML. First, you define each message on the Translatable Messages page. Second, you use the `HTMLDB_LANG.MESSAGE` API to translate the messages from PL/SQL stored procedures, functions, triggers, or packaged procedures and functions.

Defining Translatable Messages

You create translatable messages on the Translate Messages page.

To define a new translation message:

1. Navigate to the Translate Application page. (See "Navigating to the Translate Application Page" on page 16-5.)
2. On the Translate Application page, select **Optionally translate messages which are used by PL/SQL procedures and functions**.
3. On the Translate Messages page, click **Create**.
4. On the Identify Text Message page:
   - In Name, type a name to identify the text message
   - In Language, select the language for which the message would be used
   - In text, type the text to be returned when the text message is called.

     For example, you could define the message `GREETING_MSG` in English as:

     ```
     Good morning %0
     ```

     Or, you could define the message `GREETING_MSG` in German as:

     ```
     Guten Tag %0
     ```
5. Click **Create**.

HTMLDB_LANG.MESSAGE API

Use the `HTMLDB_LANG.MESSAGE` API to translate text strings (or messages) generated from PL/SQL stored procedures, functions, triggers, packaged procedures and functions.

Syntax

```
HTMLDB_LANG.MESSAGE (
    p_name    IN    VARCHAR2 DEFAULT NULL,
    p0        IN    VARCHAR2 DEFAULT NULL,
    p1        IN    VARCHAR2 DEFAULT NULL,
    p2        IN    VARCHAR2 DEFAULT NULL,
    ...
    p9        IN    VARCHAR2 DEFAULT NULL,
    p_lang    IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 16–2 describes the parameters available in the HTMLDB_LANG.MESSAGE.

Table 16–2 HTMLDB_LANG.MESSAGE Parameters

Parameter	Description
p_name	Name of the message as defined in Oracle HTML DB.
p0 ... p9	Dynamic substitution value. p0 corresponds to 0% in the message. p1 corresponds to 1% in the message. p2 corresponds to2% in the message and so on.
p_lang	Language code for the message to be retrieved. If not specified, Oracle HTML DB uses the current language for the user as defined in the Application Language Derived From attribute. **See Also:** "Specifying the Primary Language for an Application" on page 16-4

Example

The following example assumes you have defined a message called GREETING_MSG in your application in English as Good morning%0 and in German as Guten Tag%1. The following example demonstrates how you could invoke this message from PL/SQL:

```
BEGIN
    --
    -- Print the greeting
    --
    HTMLDB_LANG.MESSAGE('GREETING_MSG', V('APP_USER'));
END;
```

How p_lang attribute is defined depends on how the HTML DB engine derives the Application Primary Language. For example, if you are running the application in German and the previous call is made HTMLDB_LANG.MESSAGE, the HTML DB engine first looks for a message called GREETING_MSG with a LANG_CODE of de. If it does not find anything, then it will revert to the Application Primary Language attribute. If it still does not find anything, the HTML DB engine looks for a message by this name with a language code of en-us.

> **See Also:** "Specifying the Primary Language for an Application" on page 16-4 for more information on the Application Primary Language attribute

Translating Data that Supports List of Values

You create a dynamic translation to translate dynamic pieces of data. For example, you might use a dynamic translation on a list of values based on a database query.

Dynamic translations differ from messages in that you query a specific string rather then a message name. You define dynamic translations on the Dynamic Translations page. You then use the HTMLDB_LANG.LANG API to return the dynamic translation string identified by the parameter p_primary_text_string.

Defining a Dynamic Translation

You define dynamic translations on the Dynamic Translations page. A dynamic translation consists of a "translate-from" language string, a language code, and a "translate-to" string.

To define a dynamic translation:

1. Navigate to the Translate Application page. (See "Navigating to the Translate Application Page" on page 16-5.)
2. On the Translate Application page, select **Optionally identify any data that needs to be dynamically translated to support SQL based lists of values**.
3. On the Dynamic Translations page, click **Create** and specify the following:
   - In Language, select a target language
   - In Translate From Text, type the source text to be translated
   - In Translate To, type the translated text
4. Click **Create**.

HTMLDB_LANG.LANG API

Syntax

```
HTMLDB_LANG.LANG (
    p_primary_text_string    IN    VARCHAR2 DEFAULT NULL,
    p0                       IN    VARCHAR2 DEFAULT NULL,
    p1                       IN    VARCHAR2 DEFAULT NULL,
    p2                       IN    VARCHAR2 DEFAULT NULL,
    ...
    p9                       IN    VARCHAR2 DEFAULT NULL,
    p_primary_language       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 16–3 describes the parameters available in the `HTMLDB_LANG.LANG`.

Table 16–3 HTMLDB_LANG.LANG Parameters

Parameter	Description
`p_primary_string`	Text string of the primary language. This will be the value of the Translate From Text in the dynamic translation.

`p0`
...
`p9`
	Dynamic substitution value. `p0` corresponds to 0% in the in the translation string. `p1` corresponds to 1% in the in the translation string. `p2` corresponds to 2% in the in the translation string and so on.

`p_primary_language`	Language code for the message to be retrieved. If not specified, Oracle HTML DB uses the current language for the user as defined in the Application Language Derived From attribute. **See Also:** "Specifying the Primary Language for an Application" on page 16-4

Example

Suppose you have a table that defines all primary colors. You could define a dynamic message for each color and then apply the LANG function to the defined values in a query. For example:

```
SELECT HTMLDB_LANG.LANG(color)
  FROM my_colors
```

For example, suppose you were running the application in German andRED was a value for the color column in my_colors table. If you defined the German word for red, the previous example would return ROT.

About Oracle HTML DB Globalization Codes

If you are building a multilingual application, it is important to understand how Oracle HTML DB globalization codes impact the way in which your application runs. These codes are set automatically based on the application level Globalization attributes you select.

See Also: "Specifying the Primary Language for an Application" on page 16-4

NLS_LANGUAGE and NLS_TERRITORY determine the default presentation of number, dates, and currency.

Table 16–4 describes the globalization codes in Oracle HTML DB.

Table 16–4 Oracle HTML DB Globalization Codes

Language Name	Language Code	NLS_LANGUAGE	NLS_TERRITORY
Arabic	ar	ARABIC	
Assamese	as	ASSAMESE	INDIA
Bengali	bn	BANGLA	
Bulgarian	bg	BULGARIAN	BULGARIA
Catalan	ca	CATALAN	CATALONIA
Chinese (China)	zh-cn	SIMPLIFIED CHINESE	CHINA
Chinese (Hong Kong SAR)	zh-hk	TRADITIONAL CHINESE	HONG KONG
Chinese (Singapore)	zh-sg	SIMPLIFIED CHINESE	SINGAPORE
Chinese (Taiwan)	zh-tw	TRADITIONAL CHINESE	TAIWAN
Chinese	zh	SIMPLIFIED CHINESE	CHINA
Croatian	hr	CROATIAN	CROATIA
Czech	cs	CZECH	CZECH REPUBLIC
Danish	da	DANISH	DENMARK
Dutch (Netherlands)	nl	DUTCH	THE NETHERLANDS
English (United States)	en-us	AMERICAN	AMERICA
English	en	ENGLISH	
English	en-gb	ENGLISH (UNITED KINGDOM)	

Table 16–4 (Cont.) Oracle HTML DB Globalization Codes

Language Name	Language Code	NLS_LANGUAGE	NLS_TERRITORY
Estonian	et	ESTONIAN	ESTONIA
Finnish	fi	FINNISH	FINLAND
French (Canada)	fr-ca	CANADIAN FRENCH	CANADA
French (France)	fr	FRENCH	FRANCE
German (Germany)	de	GERMAN	GERMANY
Greek	el	GREEK	GREECE
Gujarati	gu	GUJARATI	
Hebrew	he	HEBREW	ISRAEL
Hindi	hi	HINDI	INDIA
Hungarian	hu	HUNGARIAN	HUNGARY
Icelandic	is	ICELANDIC	ICELAND
Indonesian	id	INDONESIAN	INDONESIA
Italian (Italy)	it	ITALIAN	ITALY
Japanese	ja	JAPANESE	JAPAN
Kannada	kn	KANNADA	INDIA
Korean	ko	KOREAN	KOREA
Latvian	lv	LATVIAN	LATVIA
Lithuanian	lt	LITHUANIAN	LITHUANIANA
Malay (Malaysia)	ms	MALAY	MALAYSIA
Malayalam	ml	MALAYALAM	
Marathi	mr	MARATHI	
Norwegian	no	NORWEGIAN	NORWAY
Oriya	or	ORIYA	
Polish	pl	POLISH	POLAND
Portuguese (Brazil)	pt-br	BRAZILIAN PORTUGUESE	BRAZIL
Portuguese (Portugal)	pt	PORTUGUESE	PORTUGAL
Punjabi	pa	PUNJABI	
Romanian	ro	ROMANIAN	ROMANIA
Russian	ru	RUSSIAN	
Slovak	sk	SLOVAK	SLOVAKIA
Slovenian	sl	SLOVENIAN	SLOVENIA
Spanish	es	SPANISH	SPAIN
Spanish (Mexico)	es-mx	MEXICAN SPANISH	MEXICO
Swedish	sv	SWEDISH	SWEDEN
Tamil	ta	TAMIL	
Telugu	te	TELUGU	

Table 16–4 (Cont.) Oracle HTML DB Globalization Codes

Language Name	Language Code	NLS_LANGUAGE	NLS_TERRITORY
Thai	th	THAI	THAILAND
Turkish	tr	TURKISH	TURKEY
Ukrainian	uk	UKRAINIAN	UKRAINE
Vietnamese	vi	VIETNAMESE	VIETNAM

17

Oracle HTML DB APIs

This section describes the APIs available in Oracle HTML DB.

This section contains the following topics:

HTMLDB_UTIL

The `HTMLDB_UTIL` package provides utilities you can use when programming in the Oracle HTML DB environment. You can use `HTMLDB_UTIL` to get and set session state, get files, check authorizations for users, reset different states for users, and also to get and set preferences for users.

Topics in this section include:

CHANGE_CURRENT_USER_PW Procedure

This procedure changes the password of the currently authenticated user, assuming HTML DB user accounts are in use.

Syntax

```
HTMLDB_UTIL.CHANGE_CURRENT_USER_PW(
    p_new_password IN VARCHAR2);
```

Parameters

Table 17–1 describes the parameters available in the CHANGE_CURRENT_USER_PW procedure.

Table 17–1 CHANGE_CURRENT_USER_PW Parameters

Parameter	Description
p_new_password	The new password value in clear text.

Example

```
BEGIN
HTMLDB_UTIL.CHANGE_CURRENT_USER_PW ('secret99');
END;
```

CLEAR_APP_CACHE Procedure

This procedure removes session state for a given application for the current session.

Syntax

```
HTMLDB_UTIL.CLEAR_APP_CACHE (
    p_app_id    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

p_app_id is the ID of the application for which session state will be cleared for current session.

Example

```
BEGIN
        HTMLDB_UTIL.CLEAR_APP_CACHE('100');
END;
```

CLEAR_PAGE_CACHE Procedure

This procedure removes session state for a given page for the current session.

Syntax

```
HTMLDB_UTIL.CLEAR_PAGE_CACHE (
    p_page_id IN NUMBER DEFAULT NULL);
```

Parameters

p_page_id is the ID of the page in the current application for which session state will be cleared for current session.

Example

```
BEGIN
HTMLDB_UTIL.CLEAR_PAGE_CACHE('10');
END;
```

CLEAR_USER_CACHE Procedure

This procedure removes session state and application system preferences for the current user's session. Run this procedure if you reuse session IDs and want to run applications without the benefit of existing session state.

Syntax

```
HTMLDB_UTIL.CLEAR_USER_CACHE;
```

Example

```
BEGIN
      HTMLDB_UTIL.CLEAR_USER_CACHE;
END;
```

COUNT_CLICK Procedure

This procedure counts clicks from an Oracle HTML DB application to an external site. You can also use the shorthand version procedure `Z` in place of `HTMLDB_UTIL.COUNT_CLICK`.

Syntax

```
HTMLDB_UTIL.COUNT_CLICK (
    p_url         IN    VARCHAR2,
    p_cat         IN    VARCHAR2,
    p_id          IN    VARCHAR2    DEFAULT NULL,
    p_user        IN    VARCHAR2    DEFAULT NULL,
    p_workspace   IN    VARCHAR2    DEFAULT NULL);
```

Parameters

Table 17–2 describes the parameters available in the `COUNT_CLICK` procedure.

Table 17–2 COUNT_CLICK Parameters

Parameter	Description
`p_url`	The URL to redirect to.
`p_cat`	A category to classify the click.
`p_id`	Secondary ID to associate with the click (optional).
`p_user`	The application user ID (optional).
`p_workspace`	The workspace associated with the application (optional).

Example

```
BEGIN
htp.p('<a
href=HTMLDB_UTIL.COUNT_CLICK?p_url=http://yahoo.com&p_cat=yahoo&p_workspace=NNN>
Click</a>');
end;
```

Where `NNN` equals your workspace ID.

See Also: "Purging the External Clicks Log" on page 13-4

CREATE_USER Procedure

This procedure creates a new account record in the HTML DB user account table. To execute this procedure, the current user must have administrative privileges.

Syntax

```
HTMLDB_UTIL.CREATE_USER(
    P_USER_ID                      NUMBER                 IN     DEFAULT NULL
    P_USER_NAME                    VARCHAR2               IN
    P_FIRST_NAME                   VARCHAR2               IN     DEFAULT NULL
    P_LAST_NAME                    VARCHAR2               IN     DEFAULT NULL
    P_DESCRIPTION                  VARCHAR2               IN     DEFAULT NULL
    P_EMAIL_ADDRESS                VARCHAR2               IN     DEFAULT NULL
    P_WEB_PASSWORD                 VARCHAR2               IN
    P_WEB_PASSWORD_FORMAT          VARCHAR2               IN     DEFAULT NULL
    P_GROUP_IDS                    VARCHAR2               IN     DEFAULT NULL
    P_DEVELOPER_PRIVS              VARCHAR2               IN     DEFAULT NULL
    P_DEFAULT_SCHEMA               VARCHAR2               IN     DEFAULT NULL
    P_ALLOW_ACCESS_TO_SCHEMAS      VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_01                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_02                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_03                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_04                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_05                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_06                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_07                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_08                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_09                 VARCHAR2               IN     DEFAULT NULL
    P_ATTRIBUTE_10                 VARCHAR2               IN     DEFAULT NULL)
```

Parameters

Table 17–3 describes the parameters available in CREATE_USER procedure.

Table 17–3 CREATE_USER Procedure Parameters

Parameter	Description
p_user_id	Numeric primary key of user account.
p_user_name	Alphanumeric name used for login.
p_first_name	Informational.
p_last_name	Informational.
p_description	Informational.
p_email_address	E-mail address.
p_web_address	Clear text password.
p_group_ID	Colon separated list of numeric group IDs.
p_developer_privs	Colon separated list of developer privileges (only applies to Oracle HTML DB administrators).
p_default_schema	A database schema assigned to user's workspace used by default for browsing.
p_allow_access_to_ schemas	A list of schemas assigned to user's workspace to which user is restricted.

Table 17–3 (Cont.) CREATE_USER Procedure Parameters

Parameter	Description
p_attribute_01	Arbitrary text accessible with API.
...	
p_attribute_10	

Example

```
BEGIN
HTMLDB_UTIL.CREATE_USER
    P_USER_NAME    => 'NEWUSER1',
    P_WEB_PASSWORD => 'secret99');
END;
```

CREATE_USER_GROUP Procedure

This procedure changes the password of the currently authenticated user, assuming HTML DB user accounts are in use. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
HTMLDB_UTIL.CREATE_USER_GROUP(
    P_ID                    NUMBER                  IN
    P_GROUP_NAME            VARCHAR2                IN
    P_SECURITY_GROUP_ID     NUMBER                  IN
    P_GROUP_DESC            VARCHAR2                IN);
```

Parameter

Table 17–4 describes the parameters available in the CREATE_USER_GROUP procedure.

Table 17–4 CREATE_USER_GROUP Parameters

Parameter	Description
p_id	Primary key of group.
p_group_name	Arbitrary name.
p_security_group_id	Workspace ID.
p_group_desc	Descriptive text.

Example

```
BEGIN
HTMLDB_UTIL.CREATE_USER_GROUP (
    p_id               => 0 - trigger will assign PK,
    p_group_name       => 'Managers',
    p_security_group_id => null, -- defaults to current workspace ID
    p_group_desc       => 'text');
END;
```

CURRENT_USER_IN_GROUP Function

This function returns a boolean result based on whether the current user is a member of the specified group. You may use the group name or group ID to identify the group.

Syntax

```
HTMLDB_UTIL.CURRENT_USER_IN_GROUP(
    p_group_name    IN VARCHAR2)
RETURN BOOLEAN;

HTMLDB_UTIL.CURRENT_USER_IN_GROUP(
    p_group_id    IN NUMBER)
RETURN BOOLEAN;
```

Parameters

Table 17–5 describes the parameters available in CURRENT_USER_IN_GROUP function.

Table 17–5 CURRENT_USER_IN_GROUP Parameters

Parameter	Description
p_group_name	Identifies the name of an existing group in the workspace.
p_group_id	Identifies the numeric ID of an existing group in the workspace.

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := HTMLDB_UTIL.CURRENT_USER_IN_GROUP(p_group_name=>'Managers');
END;
```

EDIT_USER Procedure

This procedure enables a user account record to be altered. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
EDIT_USER (
    P_USER_ID                    NUMBER                 IN
    P_USER_NAME                  VARCHAR2               IN
    P_FIRST_NAME                 VARCHAR2               IN     DEFAULT
    P_LAST_NAME                  VARCHAR2               IN     DEFAULT
    P_WEB_PASSWORD               VARCHAR2               IN     DEFAULT
    P_NEW_PASSWORD               VARCHAR2               IN     DEFAULT
    P_EMAIL_ADDRESS              VARCHAR2               IN     DEFAULT
    P_START_DATE                 VARCHAR2               IN     DEFAULT
    P_END_DATE                   VARCHAR2               IN     DEFAULT
    P_EMPLOYEE_ID                VARCHAR2               IN     DEFAULT
    P_ALLOW_ACCESS_TO_SCHEMAS    VARCHAR2               IN     DEFAULT
    P_PERSON_TYPE                VARCHAR2               IN     DEFAULT
    P_DEFAULT_SCHEMA             VARCHAR2               IN     DEFAULT
    P_GROUP_IDS                  VARCHAR2               IN     DEFAULT
    P_DEVELOPER_ROLES            VARCHAR2               IN     DEFAULT
    P_DESCRIPTION                VARCHAR2               IN     DEFAULTIN);
```

Parameters

Table 17–6 describes the parameters available in EDIT_USER procedure.

Table 17–6 EDIT_USER Parameters

Parameter	Description
p_user_id	Numeric primary key of user account.
p_user_name	Alphanumeric name used for login.
p_first_name	Informational.
p_last_name	Informational.
p_web_password	Clear text password,
p_start_date	Unused.
p_end_date	Unused.
p_employee_id	Unused.
p_allow_access_to_schemas	A list of schemas assigned to user's workspace to which user is restricted.
p_person_type	Unused.
p_default_schema	A database schema assigned to user's workspace used by default for browsing.
p_group_ids	Colon separated list of numeric group IDs.
p_developer_privs	Colon separated list of developer p.rivileges (only ADMIN: has meaning to HTML DB)
p_description	Informational.

EXPORT_USERS Procedure

When called from an Oracle HTML DB page, this procedure produces an export file of the current workspace definition, workspace users, and workspace groups. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
HTMLDB_UTIL.EXPORT_USERS(
    p_export_format in varchar2 default 'UNIX');
```

Parameters

Table 17–7 describes the parameters available in EXPORT_USERS procedure.

Table 17–7 EXPORT_USERS Parameters

Parameter	Description
p_export_format	Indicates how rows in the export file will be formatted. Specify 'UNIX' to have the resulting file contain rows delimited by line feeds. Specify 'DOS' to have the resulting file contain rows delimited by carriage returns and line feeds.

Example

```
BEGIN
  HTMLDB_UTIL.EXPORT_USERS;
END;
```

FETCH_APP_ITEM Function

This function fetches session state for the current or specified application in the current or specified session.

Syntax

```
HTMLDB_UTIL.FETCH_APP_ITEM(
    p_item    IN VARCHAR2,
    p_app     IN NUMBER DEFAULT NULL,
    p_session IN NUMBER DEFAULT NULL)
RETURN VARCHAR2;
```

Parameters

Table 17–8 describes the parameters available in the FETCH_APP_ITEM function.

Table 17–8 FETCH_APP_ITEM Parameters

Parameter	Description
p_item	The name of an application level item (not a page item) whose current value is to be fetched.
p_app	The ID of the application that owns the item (leave null for current application).
p_session	The session ID from which to obtain the value (leave null for current session)

Example

```
DECLARE VAL VARCHAR2(30);
BEGIN
VAL := HTMLDB_UTIL.FETCH_APP_ITEM (p_item=>'F300_NAME',p_app=>300);
END;
```

FETCH_USER Procedure

This procedure fetches a user account record. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
FETCH_USER (
    P_USER_ID                   NUMBER              IN
    P_WORKSPACE                 VARCHAR2            OUT
    P_USER_NAME                 VARCHAR2            OUT
    P_FIRST_NAME                VARCHAR2            OUT
    P_LAST_NAME                 VARCHAR2            OUT
    P_WEB_PASSWORD              VARCHAR2            OUT
    P_EMAIL_ADDRESS             VARCHAR2            OUT
    P_START_DATE                VARCHAR2            OUT
    P_END_DATE                  VARCHAR2            OUT
    P_EMPLOYEE_ID               VARCHAR2            OUT
    P_ALLOW_ACCESS_TO_SCHEMAS   VARCHAR2            OUT
    P_PERSON_TYPE               VARCHAR2            OUT
    P_DEFAULT_SCHEMA            VARCHAR2            OUT
    P_GROUPS                    VARCHAR2            OUT
    P_DEVELOPER_ROLE            VARCHAR2            OUT);
```

Parameters

Table 17–9 describes the parameters available in the FETCH_USER procedure.

Table 17–9 Fetch_User Parameters

Parameter	Description
p_user_id	Numeric primary key of user account.
p_workspace	The name of the workspace
p_user_name	Alphanumeric name used for login.
p_first_name	Informational.
p_last_name	Informational.
p_description	Informational.
p_email_address	E-mail address.
p_start_date	Unused.
p_end_date	Unused.
p_employee_id	Unused.
p_allow_access_to_schemas	A list of schemas assigned to user's workspace to which user is restricted.
p_person_type	Unused.
p_default_schema	A database schema assigned to user's workspace used by default for browsing.
p_groups	Unused.
p_developer_role	Unused.

FIND_SECURITY_GROUP_ID Function

This function returns the numeric security group ID of the named workspace.

Syntax

```
HTMLDB_UTIL.FIND_SECURITY_GROUP_ID(
    p_workspace    IN VARCHAR2
RETURN NUMBER;
```

Parameters

p_workspace is the name of the workspace.

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_UTIL.FIND_SECURITY_GROUP_ID (p_workspace=>'DEMOS');
END;
```

FIND_WORKSPACE Function

This function returns the workspace name associated with a security group ID.

Syntax

```
HTMLDB_UTIL.FIND_WORKSPACE(
```

```
    p_security_group_id    IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

`p_security_group_id` is the security group ID of a workspace.

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_UTIL.FIND_ FIND_WORKSPACE (p_security_group_id =>'20');
END;
```

GET_ATTRIBUTE Function

This function returns the value of one of the attribute values (1 through 10) of a named user in the HTML DB accounts table.

Syntax

```
HTMLDB_UTIL.GET_ATTRIBUTE(
    P_USERNAME                IN VARCHAR2
    P_ATTRIBUTE_NUMBER        IN NUMBER)
RETURN VARCHAR2;
```

Parameters

Table 17–10 describes the parameters available in the GET_ATTRIBUTE function.

Table 17–10 GET_ATTRIBUTE Parameters

Parameter	Description
`p_username`	User name in the account.
`p_attribute_number`	Number of attributes in the user record (1 through 10).

Example

```
DECLARE VAL VARCHAR2(30);
BEGIN
  VAL := HTMLDB_UTIL.GET_ATTRIBUTE (
                          p_username => 'SCOTT',
                          p_attribute_number => 1);
END;
```

GET_CURRENT_USER_ID Function

This function returns the numeric user ID of the current user.

Syntax

```
HTMLDB_UTIL.GET_CURRENT_USER_ID;
RETURN NUMBER;
```

Example

```
DECLARE VAL NUMBER;
BEGIN
```

```
  VAL := HTMLDB_UTIL.GET_CURRENT_USER_ID;
END;
```

GET_DEFAULT_SCHEMA Function

This function returns the default schema name associated with the current user.

Syntax

```
HTMLDB_UTIL.GET_DEFAULT_SCHEMA;
RETURN VARCHAR2;
```

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL. GET_DEFAULT_SCHEMA;
END;
```

GET_EMAIL Function

This function returns the e-mail address associated with the named user.

Syntax

```
HTMLDB_UTIL.GET_EMAIL(
   P_USERNAME IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

`p_username` is the user name in the account.

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_EMAIL(p_username => 'SCOTT');
END;
```

GET_FILE Procedure

This procedure downloads files from the Oracle HTML DB file repository.

Syntax

```
HTMLDB_UTIL.GET_FILE (
    p_file_id    IN   VARCHAR2,
    p_mime_type  IN   VARCHAR2 DEFAULT NULL,
    p_inline     IN   VARCHAR2 DEFAULT 'NO');
```

Parameters

Table 17–11 describes the parameters available in `GET_FILE` procedure.

Table 17–11 GET_FILE Parameters

Parameter	Description

p_file_id	ID in HTMLDB_APPLICATION_FILES of the file to be downloaded. HTMLDB_APPLICATION_FILES is a view on all files uploaded to your workspace. The following example demonstrates how to use HTMLDB_APPLICATION_FILES:
	`DECLARE`
	`  l_file_id NUMBER;`
	`BEGIN`
	`  SELECT id INTO l_file_id FROM HTMLDB_APPLICATION_FILES`
	`WHERE filename = 'myxml';`
	`  --`
	`  HTMLDB_UTIL.GET_FILE(`
	`  p_file_id => l_file_id,`
	`  p_mime_type => 'text/xml',`
	`  p_inline => 'YES');`
	`END;`

p_mime_type	Mime type of the file to download.
p_inline	Valid values include YES and NO. YES to display inline in a browser. NO to download as attachment.

Example

```
BEGIN
        HTMLDB_UTIL.GET_FILE(
              p_file_id   => '8675309',
              p_mime_type => 'text/xml',
              p_inline    => 'YES');
END;
```

GET_FILE_ID Function

This function obtains the primary key of a file in the Oracle HTML DB file repository.

Syntax

```
HTMLDB_UTIL.GET_FILE_ID (
    p_fname    IN   VARCHAR2)
RETURN NUMBER;
```

Parameters

p_fname is NAME in HTMLDB_APPLICATION_FILES of the file to be downloaded. HTMLDB_APPLICATION_FILES is a view on all files uploaded to your workspace

Example

```
DECLARE
        l_name VARCHAR2(255);
        l_file_id NUMBER;
BEGIN
        SELECT name INTO l_name FROM HTMLDB_APPLICATION_FILES
        WHERE filename = 'F125.sql';
--
        l_file_id := HTMLDB_UTIL.GET_FILE_ID(p_fname => );
END;
```

GET_FIRST_NAME Function

This function returns the FIRST_NAME field stored in the named user account record.

Syntax

```
HTMLDB_UTIL.GET_FIRST_NAME
   P_USERNAME IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

p_username identifies the user name in the account.

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_FIRST_NAME(p_username => 'SCOTT');
END;
```

GET_GROUPS_USER_BELONGS_TO Function

This function returns a colon separated list of group names to which the named user is a member.

Syntax

```
HTMLDB_UTIL.GET_GROUPS_USER_BELONGS_TO(
   P_USERNAME IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

p_username identifies the user name in the account.

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_GROUPS_USER_BELONGS_TO(p_username => 'SCOTT');
END;
```

GET_GROUP_ID Function

This function returns the numeric ID of a named group in the workspace.

Syntax

```
HTMLDB_UTIL.GET_GROUP_ID(
   P_GROUP_NAME);
RETURN VARCHAR2;
```

Parameters

p_group_name identifies the user name in the account.

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_UTIL.GET_GROUP_ID(p_group_name => 'Managers');
END;
```

GET_GROUP_NAME Function

This function returns the name of a group identified by a numeric ID.

Syntax

```
HTMLDB_UTIL.GET_GROUP_NAME(
   P_GROUP_ID);
RETURN NUMBER;
```

Parameters

p_group_id identifies a numeric ID of a group in the workspace.

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_GROUP_NAME(p_group_id => 8922003);
END;
```

GET_LAST_NAME Function

This function returns the LAST_NAME field stored in the named user account record.

Syntax

```
HTMLDB_UTIL.GET_LAST_NAME(
   P_USERNAME IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

p_username is the user name in the user account record.

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_LAST_NAME(p_username => 'SCOTT');
END;
```

GET_USERNAME Function

This function returns the user name of a user account identified by a numeric ID.

Syntax

```
HTMLDB_UTIL.GET_USERNAME(
   P_USERID);
RETURN NUMBER;
```

Parameters

p_userid identifies the numeric ID of a user account in the workspace.

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_USERNAME(p_userid => 228922003);
END;
```

GET_NUMERIC_SESSION_STATE Function

This function returns a numeric value for a numeric item. You can use this function in Oracle HTML DB applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function NV, in place of HTMLDB_UTIL.GET_NUMERIC_SESSION_STATE.

Syntax

```
HTMLDB_UTIL.GET_NUMERIC_SESSION_STATE (
    p_item     IN VARCHAR2)
    RETURN NUMBER;
```

Parameters

p_item is the case insensitive name of the item for which you wish to have the session state fetched.

Example

```
DECLARE
      l_item_value    Number;
BEGIN
      l_item_value := HTMLDB_UTIL.GET_NUMERIC_SESSION_STATE('my_item');
END;
```

GET_PREFERENCE Function

This function retrieves the value of a previously saved preference for a given user.

Syntax

```
HTMLDB_UTIL.GET_PREFERENCE (
    p_preference  IN    VARCHAR2 DEFAULT NULL,
    p_user        IN    VARCHAR2 DEFAULT V('USER'))
    RETURN VARCHAR2;
```

Parameters

Table 17–12 describes the parameters available in the GET_PREFERENCE function.

Table 17–12 GET_PREFERENCE Parameters

Parameter	Description
p_preference	Name of the preference to retrieve the value.
p_value	Value of the preference.

Table 17–12 (Cont.) GET_PREFERENCE Parameters

Parameter	Description
p_user	User for whom the preference is being retrieved.

Example

```
DECLARE
      l_default_view    VARCHAR2(255);
BEGIN
      l_default_view := HTMLDB_UTIL.GET_PREFERENCE(
                  p_preference => 'default_view',
                  p_user       => :APP_USER);
END;
```

GET_SESSION_STATE Function

This function returns the value for an item. You can use this function in your Oracle HTML DB applications wherever you can use PL/SQL or SQL. You can also use the shorthand, function V, in place of HTMLDB_UTIL.GET_SESSION_STATE.

Syntax

```
HTMLDB_UTIL.GET_SESSION_STATE (
    p_item    IN   VARCHAR2)
    RETURN VARCHAR2;
```

Parameters

p_item is the case insensitive name of the item for which you wish to fetch session state.

Example

```
DECLARE
      l_item_value  VARCHAR2(255);
BEGIN
      l_item_value := HTMLDB_UTIL.GET_SESSION_STATE('my_item');
END;
```

GET_USER_ID Function

This function returns the numeric ID of a named user in the workspace.

Syntax

```
HTMLDB_UTIL.GET_USER_ID(
   P_USERNAME);
RETURN VARCHAR2;
```

Parameters

p_username identifies the name of a user in the workspace.

Example

```
DECLARE VAL NUMBER;
BEGIN
```

```
  VAL := HTMLDB_UTIL.GET_USER_ID(p_username => 'Managers');
END;
```

GET_USER_ROLES Function

This function returns the DEVELOPER_ROLE field stored in the named user account record.

Syntax

```
HTMLDB_UTIL.GET_USER_ROLES(
   P_USERNAME IN VARCHAR2);
RETURN VARCHAR2;
```

Parameters

p_username identifies a user name in the account.

Example

```
DECLARE VAL VARCHAR2;
BEGIN
  VAL := HTMLDB_UTIL.GET_USER_ROLES(p_username=>'SCOTT');
END;
```

IS_LOGIN_PASSWORD_VALID Function

This function returns a boolean result based on the validity of the password for a named user account in the current workspace. Returns true if the password matches and false if the password does not match.

Syntax

```
HTMLDB_UTIL.IS_LOGIN_PASSWORD_VALID(
P_USERNAME IN VARCHAR2,
   P_PASSWORD IN VARCHAR2);
RETURN BOOLEAN;
```

Parameters

Table 17–13 describes the parameters available in the IS_LOGIN_PASSWORD_VALID function.

Table 17–13 IS_LOGIN_PASSWORD_VALID Parameters

Parameter	Description
p_username	User name in account
p_password	Password to be compared with password stored in the account.

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := HTMLDB_UTIL. IS_LOGIN_PASSWORD_VALID (
             p_username=>'SCOTT'
             p_password=>'tiger');
END;
```

IS_USERNAME_UNIQUE Function

This function returns a boolean result based on whether the named user account is unique in the workspace.

Syntax

```
HTMLDB_UTIL.IS_USERNAME_UNIQUE(
   P_USERNAME IN VARCHAR2);
RETURN BOOLEAN;
```

Parameters

p_username identifies the user name to be tested.

Example

```
DECLARE VAL BOOLEAN;
BEGIN
  VAL := HTMLDB_UTIL.IS_USERNAME_UNIQUE(
             p_username=>'SCOTT');
END;
```

PUBLIC_CHECK_AUTHORIZATION Function

Given the name of a security scheme, this function determines if the current user passes the security check.

Syntax

```
HTMLDB_UTIL.PUBLIC_CHECK_AUTHORIZATION (
    p_security_scheme    IN    VARCHAR2)
    RETURN BOOLEAN;
```

Parameters

p_security_name is the name of the security scheme that determines if the user passes the security check.

Example

```
DECLARE
      l_check_security  boolean;
BEGIN
      l_check_security := HTMLDB_UTIL.PUBLIC_CHECK_AUTHORIZATION('my_auth_
scheme');
END;
```

REMOVE_PREFERENCE Procedure

This function removes the preference for the supplied user.

Syntax

```
HTMLDB_UTIL.REMOVE_PREFERENCE(
    p_preference    IN    VARCHAR2 DEFAULT NULL,
    p_user          IN    VARCHAR2 DEFAULT V('USER'));
```

Parameters

Table 17–14 describes the parameters available in the REMOVE_PREFERENCE procedure.

Table 17–14 REMOVE_PREFERENCE Parameters

Parameter	Description
p_preference	Name of the preference to remove.
p_user	User for whom the preference is for.

Example

```
BEGIN
       HTMLDB_UTIL.REMOVE_PREFERENCE(
                    p_preference => 'default_view',
                    p_user       => :APP_USER);
END;
```

REMOVE_SORT_PREFERENCES Procedure

This procedure removes the user's column heading sorting preference value.

Syntax

```
HTMLDB_UTIL.REMOVE_SORT_PREFERENCES (
    p_user  IN VARCHAR2 DEFAULT V('USER'));
```

Parameters

p_user identifies the user for whom sorting preferences will be removed.

Example

```
BEGIN
      HTMLDB_UTIL.REMOVE_SORT_PREFERENCES(:APP_USER);
END;
```

REMOVE_USER Procedure

This procedure removes the user account identified by the primary key or a user name. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
HTMLDB_UTIL.REMOVE_USER(
    p_user_id IN NUMBER,
    p_user_name IN VARCHAR2);
```

Parameters

Table 17–15 describes the parameters available in the REMOVE_USER procedure.

Table 17–15 REMOVE_USER Parameters

Parameter	Description
p_user_id	The numeric primary key of the user account record.
p_user_name	The the user name of the user account.

Example

```
BEGIN
HTMLDB_UTIL.REMOVE_USER(p_user_id=>'99997');
END;

BEGIN
HTMLDB_UTIL.REMOVE_USER(p_user_name => 'SCOTT');
END;
```

RESET_PW Procedure

This procedure resets the password for a named user and emails it to them with a message. To execute this procedure, the current user must have administrative privilege in the workspace.

Syntax

```
HTMLDB_UTIL.RESET_PW(
    p_user IN VARCHAR2,
    p_msg  IN VARCHAR2);
```

Parameters

Table 17–16describes the parameters available in the RESET_PW procedure.

Table 17–16 RESET_PW Parameters

Parameter	Description
p_user	The user name of the user account
p_msg	Message text to be emailed to user.

Example

```
BEGIN
HTMLDB_UTIL.REMOVE_USER(
    p_user => 'SCOTT',
    p_msg => 'Contact help desk at 555-1212 with questions');
END;
```

RESET_AUTHORIZATIONS Procedure

To increase performance, Oracle HTML DB caches security checks. You can use this procedure to undo caching thus requiring all security checks be revalidated for the current user. Use this procedure if you wish users to have the ability to change their responsibilities (their authorization profile) within your application.

Syntax

```
HTMLDB_UTIL.RESET_AUTHORIZATIONS;
```

Example

```
BEGIN
HTMLDB_UTIL.RESET_AUTHORIZATIONS;
END;
```

SET_EMAIL Procedure

This procedure updates a user account with a new e-mail address. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
HTMLDB_UTIL.SET_EMAIL(
    p_userid IN NUMBER,
    p_email  IN VARCHAR2);
```

Parameters

Table 17–17 describes the parameters available in the SET_EMAIL procedure.

Table 17–17 SET_EMAIL Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_email	The e-mail address to be saved in user account.

Example

```
BEGIN
HTMLDB_UTIL.SET_EMAIL(
    p_userid  => '888883232',
    P_email   => 'scott.scott@oracle.com');
END;
```

SET_FIRST_NAME Procedure

This procedure updates a user account with a new FIRST_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
HTMLDB_UTIL.SET_FIRST_NAME(
    p_userid      IN NUMBER,
    p_first_name  IN VARCHAR2);
```

Parameters

Table 17–19 describes the parameters available in the SET_FIRST_NAME procedure.

Table 17–18 SET_FIRST_NAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_first_name	FIRST_NAME value to be saved in user account.

Example

```
BEGIN
HTMLDB_UTIL.SET_FIRST_NAME(
    p_userid       => '888883232',
    P_first_name   => 'Scott');
END;
```

SET_LAST_NAME Procedure

This procedure updates a user account with a new LAST_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
HTMLDB_UTIL.SET_LAST_NAME(
    p_userid      IN NUMBER,
    p_last_name  IN VARCHAR2);
```

Parameters

Table 17–19 describes the parameters available in the SET_LAST_NAME procedure.

Table 17–19 SET_LAST_NAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_last_name	LAST_NAME value to be saved in the user account.

Example

```
BEGIN
HTMLDB_UTIL.SET_LAST_NAME(
    p_userid       => '888883232',
    p_last_name   => 'SMITH');
END;
```

SET_USERNAME Procedure

This procedure updates a user account with a new USER_NAME value. To execute this procedure, the current user must have administrative privileges in the workspace.

Syntax

```
HTMLDB_UTIL.USERNAME(
    p_userid      IN NUMBER,
    p_username    IN VARCHAR2);
```

Parameters

Table 17–20 describes the parameters available in the SET_USERNAME procedure.

Table 17–20 SET_USERNAME Parameters

Parameter	Description
p_userid	The numeric ID of the user account.
p_user_name	USER_NAME value to be saved in the user account.

Example

```
BEGIN
HTMLDB_UTIL.SET_USERNAME(
    p_userid       => '888883232',
    P_username   => 'USER-XRAY');
END;
```

SET_PREFERENCE Procedure

This procedure sets a preference that will persist beyond the user's current session.

Syntax

```
HTMLDB_UTIL.SET_PREFERENCE (
    p_preference   IN    VARCHAR2 DEFAULT NULL,
    p_value        IN    VARCHAR2 DEFAULT NULL,
    p_user         IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 17–21 describes the parameters available in the SET_PREFERENCE procedure.

Table 17–21 SET_PREFERENCE Parameters

Parameter	Description
p_preference	Name of the preference (case-sensitive).
p_value	Value of the preference.
p_user	User for whom the preference is being set.

Example

```
BEGIN
       HTMLDB_UTIL.SET_PREFERENCE(
             p_preference => 'default_view',
             p_value      => 'WEEKLY',
             p_user       => :APP_USER);
END;
```

SET_SESSION_STATE Procedure

This procedure sets session state for a current Oracle HTML DB session.

Syntax

```
HTMLDB_UTIL.SET_SESSION_STATE (
    p_name     IN    VARCHAR2 DEFAULT NULL,
```

```
    p_value    IN    VARCHAR2 DEFAULT NULL);
```

Parameters

Table 17–22 describes the parameters available in the SET_SESSION_STATE procedure.

Table 17–22 SET_SESSION_STATE Parameters

Parameter	Description
p_name	Name of the application or page level item for which you are setting sessions state.
p_value	Value of session state to set.

Example

```
BEGIN
HTMLDB_UTIL.SET_SESSION_STATE('my_item','myvalue');
END;
```

STRING_TO_TABLE Function

Given a string, this function returns a PL/SQL array of type HTMLDB_APPLICATION_GLOBAL.VC_ARR2. This array is a VARCHAR2(32767) table.

Syntax

```
HTMLDB_UTIL.STRING_TO_TABLE (
    p_string       IN VARCHAR2,
    p_separator    IN VARCHAR2 DEFAULT ':')
    RETURN HTMLDB_APPLICATION_GLOBAL.VC_ARR2;
```

Parameters

Table 17–23 describes the parameters available in the STRING_TO_TABLE function.

Table 17–23 STRING_TO_TABLE Parameters

Parameter	Description
p_string	String to be converted into a PL/SQL table of type HTMLDB_APPLICATION_GLOBAL.VC_ARR2.
p_separator	String separator. The default is a colon.

Example

```
DECLARE
       l_vc_arr2    HTMLDB_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
       l_vc_arr2 := HTMLDB_UTIL.STRING_TO_TABLE('One:Two:Three');
       FOR z IN 1..l_vc_arr2.count LOOP
             htp.p(l_vc_arr2(z));
       END LOOP;
END;
```

TABLE_TO_STRING Function

Given a a PL/SQL table of type HTMLDB_APPLICATION_GLOBAL.VC_ARR2, this function returns a delimited string separated by the supplied separator, or by the default separator, a colon (:).

Syntax

```
HTMLDB_UTIL.TABLE_TO_STRING (
    p_table       IN     HTMLDB_APPLICATION_GLOBAL.VC_ARR2,
    p_string      IN     VARCHAR2 DEFAULT ':')
    RETURN VARCHAR2;
```

Parameters

Table 17–24 describes the parameters available in the TABLE_TO_STRING function.

Table 17–24 TABLE_TO_STRING Parameters

Parameter	Description
p_string	String separator. Default separator is a colon (:).
p_table	PL/SQL table that is to be converted into a delimited string.

Example

```
DECLARE
       l_string     VARCHAR2(255);
       l_vc_arr2    HTMLDB_APPLICATION_GLOBAL.VC_ARR2;
BEGIN
       l_vc_arr2 := HTMLDB_UTIL.STRING_TO_TABLE('One:Two:Three');

       l_string := HTMLDB_UTIL.TABLE_TO_STRING(l_vc_arr2);
END;
```

URL_ENCODE Function

This function encodes (into HEX) all special characters that include spaces, question marks, and ampersands.

Syntax

```
HTMLDB_UTIL.URL_ENCODE (
    p_url   IN    VARCHAR2)
    RETURN VARCHAR2;
```

Parameters

p_string is the string you would like to have encoded.

Example

```
DECLARE
      l_url  VARCHAR2(255);
BEGIN
      l_url := HTMLDB_UTIL.URL_ENCODE('http://www.myurl.com?id=1&cat=foo');
END;
```

HTMLDB_MAIL

You can use the HTMLDB_MAIL package to send an e-mail from an Oracle HTML DB application. This package is built on top of the Oracle supplied UTL_SMTP package. Because of this dependence, the UTL_SMTP package must be installed and functioning in order to use HTMLDB_MAIL.

> **See Also:** *PL/SQL Packages and Types Reference* for more information about the UTL_SMTP package

HTMLDB_MAIL contains two procedures. Use HTMLDB_MAIL.SEND to send an outbound e-mail message from your application. Use HTMLDB_MAIL.PUSH_QUEUE to deliver mail messages stored in HTMLDB_MAIL_QUEUE.

Topics in this section include:

- SEND Procedure
- PUSH_QUEUE Procedure

> **Note:** The most efficient approach to sending e-mail is to create a background job (using a DBMS_JOB package) to periodically send all mail messages stored in the active mail queue.

> **See Also:** "Sending E-mail from an Application" on page 15-1

SEND Procedure

This procedure sends an outbound e-mail message from an application. Although you can use this procedure to pass in either a VARCHAR2 or a CLOB to p_body and p_body_html, the data types must be the same. In other words, you cannot pass a CLOB to P_BODY and a VARCHAR2 to p_body_html.

When using HTMLDB_MAIL.SEND, remember the following:

- **No single line may exceed 1000 characters.** The SMTP/MIME specification dictates that no single line shall exceed 1000 characters. To comply with this restriction, you must add a carriage return or line feed characters to break up your p_body or p_body_html parameters into chunks of 1000 characters or less. Failing to do so will result in erroneous e-mail messages, including partial messages or messages with extraneous exclamation points.
- **Plain text and HTML e-mail content.** Passing a value to p_body, but not p_body_html results in a plain text message. Passing a value to p_body and p_body_html yields a multi-part message that includes both plain text and HTML content. The settings and capabilities of the recipient's email client determine what displays. Although most modern e-mail clients can read a HTML formatted email, remember that some users disable this functionality to address security issues.
- **Avoid images.** When referencing images in p_body_html using the <img /> tag, remember that the images must be accessible to the recipient's e-mail client in order for them to see the image.

  For example, suppose you reference an image on your network called hello.gif as follows:

```
<img src="http://someserver.com/hello.gif" alt="Hello" />]
```

In this example, the image is not attached to the email, but is referenced by the e-mail. For the recipient to see it, they must be able to access the image using a Web browser. If the image is inside a firewall and the recipient is outside of the firewall, the image will not display. For this reason, avoid using images. If you must include images, be sure to include the ALT attribute to provide a textual description in the event the image is not accessible.

Syntax

```
HTMLDB_MAIL.SEND(
    p_to                        IN    VARCHAR2,
    p_from                      IN    VARCHAR2,
    p_body                      IN  [ VARCHAR2 | CLOB ],
    p_body_html                 IN  [ VARCHAR2 | CLOB ] DEFAULT,
    p_subj                      IN    VARCHAR2 DEFAULT)
    p_cc                        IN    VARCHAR2 DEFAULT)
    p_bcc                       IN    VARCHAR2 DEFAULT);
```

Parameters

Table 17–25 describes the parameters available in the SEND procedure.

Table 17–25 Send Parameters

Parameter	Description
p_to	Valid e-mail address to which the e-mail will be sent (required). For multiple e-mail addresses, use a comma separated list.
p_from	E-mail address from which the e-mail will be sent (required). This e-mail address must be a valid address. Otherwise, the message will not be sent.
p_body	Body of the e-mail in plain text, not HTML (required). If a value is passed to p_body_html, then this is the only text the recipient sees. If a value is not passed to p_body_html, then this text only displays for e-mail clients that do not support HTML or have HTML disabled. A carriage return or line feed (CRLF) must be included every 1000 characters.
p_body_html	Body of the e-mail in HTML format. This must be a full HTML document including the <html> and <body> tags. A single line cannot exceed 1000 characters without a carriage return or line feed (CRLF).
p_subj	Subject of the e-mail.
p_cc	Valid e-mail addresses to which the e-mail is copied. For multiple e-mail addresses, use a comma separated list.
p_bcc	Valid e-mail addresses to which the e-mail is blind copied. For multiple e-mail addresses, use a comma separated list.

Examples

The following example demonstrates how to use HTMLDB_MAIL.SEND to send a plain text e-mail message from an application.

```
-- Example One: Plain Text only message
DECLARE
    l_body      CLOB;
BEGIN
    l_body := 'Thank you for your interest in the HTMLDB_MAIL
package.'||utl_tcp.crlf||utl_tcp.crlf;
    l_body := l_body ||'  Sincerely,'||utl_tcp.crlf;
```

```
    l_body := l_body ||'  The HTMLDB Dev Team'||utl_tcp.crlf;
    htmldb_mail.send(
        p_to       => 'some_user@somewhere.com',   -- change to your email address
        p_from     => 'some_sender@somewhere.com', -- change to a real senders
email address
        p_body     => l_body,
        p_subj     => 'HTMLDB_MAIL Package - Plain Text message');
END;
/
```

The following example demonstrates how to use `HTMLDB_MAIL.SEND` to send a HTML e-mail message from an application. Remember, you must include a carriage return or line feed (CRLF) every 1000 characters. The example that follows uses `utl_tcp.crlf`.

```
-- Example Two: Plain Text / HTML message
DECLARE
    l_body      CLOB;
    l_body_html CLOB;
BEGIN
    l_body := 'To view the content of this message, please use an HTML enabled
mail client.'||utl_tcp.crlf;

    l_body_html := '<html>
                      <head>
                        <style type="text/css">
                          body{font-family: Arial, Helvetica, sans-serif;
                               font-size:10pt;
                               margin:30px;
                               background-color:#ffffff;}

                          span.sig{font-style:italic;
                                   font-weight:bold;
                                   color:#811919;}
                        </style>
                      </head>
                      <body>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'<p>Thank you for your interest in the
<strong>HTMLDB_MAIL</strong> package.</p>'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  Sincerely,<br />'||utl_tcp.crlf;
    l_body_html := l_body_html ||'  <span class="sig">The HTMLDB Dev
Team</span><br />'||utl_tcp.crlf;
    htmldb_mail.send(
     p_to        => 'some_user@somewhere.com',   -- change to your email address
     p_from      => 'some_sender@somewhere.com', -- change to a real senders email
address
     p_body      => l_body,
     p_body_html => l_body_html,
     p_subj      => 'HTMLDB_MAIL Package - HTML formatted message');
END;
/
```

PUSH_QUEUE Procedure

Oracle HTML DB stores unsent e-mail messages in a table named `HTMLDB_MAIL_QUEUE`. You can manually deliver mail messages stored in this queue to the specified SMTP gateway by invoking the `HTMLDB_MAIL.PUSH_QUEUE` procedure. This procedure requires two input parameters:

- p_smtp_hostname defines the hostname of your SMTP gateway
- p_smtp_portno defines port number of your SMTP gateway (for example, 25)

Oracle HTML DB logs successfully submitted message in the table HTMLDB_MAIL_LOG with the timestamp reflecting your server's local time. Keep in mind, the most efficient approach to sending e-mail is to create a background job (using a DBMS_JOB package) to periodically send all mail messages stored in the active mail queue.

See Also: "Sending E-mail Using a Background Job" on page 15-1

Syntax

```
HTMLDB_MAIL.PUSH_QUEUE(
    p_smtp_hostname             IN    VARCHAR2 DEFAULT,
    p_smtp_portno               IN    NUMBER   DEFAULT;
```

Parameters

Table 17–26 describes the parameters available in the HTMLDB_MAIL procedure.

Table 17–26 PUSH_QUEUE Parameters

Parameters	Description
p_smtp_hostname	SMTP gateway hostname.
p_smtp_portno	SMTP gateway port number.

Example

The following example demonstrates the use of the HTMLDB_MAIL.PUSH_QUEUE procedure using a shell script. This example only applies to UNIX/LINUX installations. In this example, the SMTP gateway hostname is defined as smtp01.oracle.com and the SMTP gateway port number is 25.

```
SQLPLUS / <<EOF
FLOWS_010600.HTMLDB_MAIL.PUSH_QUEUE('smtp01.oracle.com','25');
DISCONNECT
EXIT
EOF
```

See Also: "Sending E-mail from an Application" on page 15-1

HTMLDB_ITEM

You can use the HTMLDB_ITEM package to create form elements dynamically based on a SQL query instead of creating individual items page by page.

Topics in this section include:

CHECKBOX Function

This function creates check boxes.

Syntax

```
HTMLDB_ITEM.CHECKBOX(
    p_idx                       IN    NUMBER,
    p_value                     IN    VARCHAR2 DEFAULT,
    p_attributes                IN    VARCHAR2 DEFAULT,
    p_checked_values            IN    VARCHAR2 DEFAULT,
    p_checked_values_delimitor  IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 17–27 describes the parameters available in the `CHECKBOX` function.

Table 17–27 CHECKBOX Parameters

Parameter	Description
`p_idx`	Number which determines which `HTMLDB_APPLICATION` global will be used. Valid range of values is 1 to 50. For example 1 creates `F01` and 2 creates `F02`.
`p_value`	Value of a check box, hidden field, or input form item.
`p_attributes`	Controls HTML tag attributes (such as disabled).
`p_checked_values`	Values to be checked by default.
`p_checked_values_delimitor`	Delimits the values in the previous parameter, `p_checked_values`.

Examples of Default Check Box Behavior

The following example demonstrates how to create a selected check box for each employee in the emp table.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,empno,'CHECKED') " ",
       ename,
       job
FROM   emp
ORDER BY 1
```

The next example demonstrates how to have all check boxes for employees display without being selected.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,empno) " ",
       ename,
       job
FROM   emp
ORDER BY 1
```

The next example demonstrates how to select the check boxes for employees who work in department 10.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,empno,DECODE(deptno,10,'CHECKED',null)) " ",
       ename,
       job
FROM   emp
ORDER BY 1
```

The next example demonstrates how to select the check boxes for employees who work in department 10 or department 20.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,deptno,NULL,'10:20',':') " ",
       ename,
       job
FROM   emp
ORDER BY 1
```

Creating an On-Submit Process

If you are using check boxes in your application, you might need to create an On Submit process to perform a specific type of action on the selected rows. For example, you could have a Delete button that utilizes the following logic:

```
SELECT HTMLDB_ITEM.CHECKBOX(1,empno) " ",
       ename,
       job
FROM   emp
ORDER  by 1
```

Consider the following sample on-submit process:

```
FOR I in 1..HTMLDB_APPLICATION.G_F01.COUNT LOOP
    DELETE FROM emp WHERE empno = to_number(HTMLDB_APPLICATION.G_F01(i));
END LOOP;
```

DATE_POPUP Function

Use this function with forms that include date fields. DATE_POPUP dynamically generates a date field that has popup calendar button.

Syntax

```
HTMLDB_ITEM.DATE_POPUP(
    p_idx          IN    NUMBER,
    p_row          IN    NUMBER,
    p_value        IN    VARCHAR2 DEFAULT,
    p_date_format  IN    DATE DEFAULT,
    p_size         IN    NUMBER DEFAULT,
    p_maxlength    IN    NUMBER DEFAULT,
    p_attributes   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 17–28 describes the parameters available in the DATE_POPUP function.

Table 17–28 DATE_POPUP Parameters

Parameter	Description
p_idx	Number which determines which HTMLDB_APPLICATION global will be used.Valid range of values is 1 to 50. For example 1 creates F01 and 2 creates F02.
p_row	p_row is deprecated. Anything specified for this value will be ignored.
p_value	Value of a field item.
p_date_format	Valid database date format.
p_size	Controls HTML tag attributes (such as disabled).
p_maxlength	Determine the maximum number of enterable characters. Becomes the maxlength attribute of the <input > HTML tag.
p_attributes	Extra HTML parameters you wish to add.

> **See Also:** *Oracle Database SQL Reference* for more information on the TO_CHAR or TO_DATE functions

Example

The following example demonstrates how to use HTMLDB_ITEM.DATE_POPUP to create popup calendar buttons for the hiredate column.

```
SELECT
  empno,
  HTMLDB_ITEM.HIDDEN(1,empno)||
  HTMLDB_ITEM.TEXT(2,ename) ename,
  HTMLDB_ITEM.TEXT(3,job) job,
  mgr,
  HTMLDB_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hd,
  HTMLDB_ITEM.TEXT(5,sal) sal,
  HTMLDB_ITEM.TEXT(6,comm) comm,
  deptno
FROM emp
ORDER BY 1
```

DISPLAY_AND_SAVE Function

Use this function to display an item as text, but save its value to session state.

Syntax

```
HTMLDB_ITEM.DISPLAY_AND_SAVE(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 17–29 describes the parameters available in the DISPLAY_AND_SAVE.

Table 17–29 DISPLAY_AND_SAVE Parameters

Parameter	Description
p_idx	Number which determines which HTMLDB_APPLICATION global will be used.Valid range of values is 1 to 50. For example 1 creates F01 and 2 creates F02.
p_value	Current value.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the text field item.

Example

The following example demonstrates how to use HTMLDB_ITEM.DISPLAY_AND_SAVE.

```
SELECT HTMLDB_ITEM.DISPLAY_AND_SAVE(10,empno) c FROM emp
```

HIDDEN Function

This function dynamically generates hidden form items.

Syntax

```
HTMLDB_ITEM.HIDDEN(
    p_idx     IN    NUMBER,
    p_value   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 17–30 describes the parameters available in the HIDDEN function.

Table 17–30 HIDDEN Parameters

Parameter	Description
p_idx	Number to identify the item you wish to generate. The number will determine which G_FXX global is populated. **See Also:** "HTMLDB_APPLICATION" on page 17-52

Table 17–30 (Cont.) HIDDEN Parameters

Parameter	Description
p_value	Value of the hidden input form item.

Example

Typically, the primary key of a table is stored as a hidden column and used for subsequent update processing. Consider the following sample SLQ query:

```
SELECT
  empno,
  HTMLDB_ITEM.HIDDEN(1,empno)||
  HTMLDB_ITEM.TEXT(2,ename) ename,
  HTMLDB_ITEM.TEXT(3,job) job,
  mgr,
  HTMLDB_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
  HTMLDB_ITEM.TEXT(5,sal) sal,
  HTMLDB_ITEM.TEXT(6,comm) comm,
  deptno
FROM emp
ORDER BY 1
```

The previous query could use the following page process to process the results:

```
BEGIN
  FOR i IN 1..HTMLDB_APPLICATION.G_F01.COUNT LOOP
    UPDATE emp
    SET
      ename=HTMLDB_APPLICATION.G_F02(i),
      job=HTMLDB_APPLICATION.G_F03(i),
      hiredate=to_date(HTMLDB_APPLICATION.G_F04(i),'dd-mon-yyyy'),
      sal=HTMLDB_APPLICATION.G_F05(i),
      comm=HTMLDB_APPLICATION.G_F06(i)
    WHERE empno=to_number(HTMLDB_APPLICATION.G_F01(i));
  END LOOP;
END;
```

Note that the G_F01 column (which corresponds to the hidden EMPNO) is used as the key to update each row.

MD5_CHECKSUM Function

This function passes values to HTMLDB_ITEM.MULTI_ROW_UPDATE and is used for lost update detection. Lost update detection ensures data integrity in applications where data can be accessed concurrently.

Syntax

```
HTMLDB_ITEM.MD5_CHECKSUM(
    p_value01   IN    VARCHAR2 DEFAULT,
    p_value02   IN    VARCHAR2 DEFAULT,
    p_value03   IN    VARCHAR2 DEFAULT,
    ...
    p_value50   IN    VARCHAR2 DEFAULT,
    p_col_sep   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 17–32 describes the parameters available in the MD5_CHECKSUM function.

Table 17–31 MD5_HIDDEN Parameters

Parameter	Description
p_value01	Fifty available inputs. Parameters that are not supplied default to null.
...	
p_value50	
p_col_sep	String used to separate p_value inputs. Defaults to the pipe symbol (|).

Example

```
SELECT HTMLDB_ITEM.MD5_CHECKSUM(ename,job,sal)
FROM emp
```

MD5_HIDDEN Function

This function is used for lost update detection which ensures data integrity in applications where data can be accessed concurrently.

This function produces a hidden form field and includes 50 inputs. HTMLDB_ITEM.MD5_HIDDEN also produces an MD5 checksum using the Oracle database DBMS_OBFUSCATION_TOOLKIT:

```
UTL_RAW.CAST_TO_RAW(DBMS_OBFUSCATION_TOOLKIT.MD5())
```

An MD5 checksum provides data integrity through hashing and sequencing to assure that data is not altered or stolen as it is transmitted over a network

Syntax

```
HTMLDB_ITEM.MD5_HIDDEN(
    p_idx       IN    NUMBER,
    p_value01   IN    VARCHAR2 DEFAULT,
    p_value02   IN    VARCHAR2 DEFAULT,
    p_value03   IN    VARCHAR2 DEFAULT,
    ...
    p_value50   IN    VARCHAR2 DEFAULT,
    p_col_sep   IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 17–32 describes the parameters available in the MD5_HIDDEN function.

Table 17–32 MD5_HIDDEN Parameters

Parameter	Description
p_idx	Indicates the form element to be generated. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.
p_value01	Fifty available inputs. Parameters not supplied default to null.
...	
p_value50	

Table 17–32 (Cont.) MD5_HIDDEN Parameters

Parameter	Description
p_col_sep	String used to separate p_value inputs. Defaults to the pipe symbol (|).

Example

p_idx specifies the FXX form element to be generated. In the following example, 7 generates F07. Also note that an HTML hidden form element will be generated.

```
SELECT HTMLDB_ITEM.MD5_HIDDEN(7,ename,job,sal), ename, job, sal FROM emp
```

MULTI_ROW_UPDATE Procedure

Use this procedure within a Multi Row Update process type. This procedure takes a string containing a multiple row update definition in the following format:

```
OWNER:TABLE:pk_column1,pk_idx:pk_column2,pk_idx2|col,idx:col,idx...
```

Syntax

```
HTMLDB_ITEM.MULTI_ROW_UPDATE(
    p_mru_string    IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Example

To use this procedure indirectly within application level process, you need to create a query to generate a form of database data. The following example demonstrates how to create a multiple row update on the emp table.

```
SELECT
empno,
HTMLDB_ITEM.HIDDEN(1,empno),
HTMLDB_ITEM.HIDDEN(2,deptno),
HTMLDB_ITEM.TEXT(3,ename),
HTMLDB_ITEM.SELECT_LIST_FROM_QUERY(4,job,'SELECT DISTINCT job FROM emp'),
HTMLDB_ITEM.TEXT(5,sal),
HTMLDB_ITEM.TEXT(7,comm),
HTMLDB_ITEM.MD5_CHECKSUM(ename,job,sal,comm),
deptno
FROM emp
WHERE deptno = 20
```

Note the call to HTMLDB_ITEM.MD5_CHECKSUM instead of HTMLDB_ITEM.MD5_HIDDEN. Since HTMLDB_ITEM.MULTI_ROW_UPDATE gets the checksum from HTMLDB_APPLICATION.G_FCS, you need to call HTMLDB_ITEM.MD5_CHECKSUM in order to populate HTMLDB_APPLICATION.G_FCS when the page is submitted. Additionally, the columns in HTMLDB_ITEM.MD5_CHECKSUM must be in the same order those in the MULTI_ROW_UPDATE process. These updates can then processed (or applied to the database) using an after submit page process of Multi Row Update in a string similar to the following:

```
SCOTT:emp:empno,1:deptno,2|ename,3:job,4:sal,5:comm,7:,:,:,:,
```

SELECT_LIST Function

This function dynamically generates a static select list. Similar to other functions available in the HTMLDB_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT,
    p_list_values   IN   VARCHAR2 DEFAULT,
    p_attributes    IN   VARCHAR2 DEFAULT,
    p_show_null     IN   VARCHAR2 DEFAULT,
    p_null_value    IN   VARCHAR2 DEFAULT,
    p_null_text     IN   VARCHAR2 DEFAULT,
    p_item_id       IN   VARCHAR2 DEFAULT,
    p_item_label    IN   VARCHAR2 DEFAULT,
    p_show_extra    IN   VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 17–33 describes the parameters available in the SELECT_LIST function.

Table 17–33 SELECT_LIST Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the P_IDX parameter is constant for a given column.
p_value	Current value. This value should be a value in the P_LIST_VALUES parameter.
p_list_values	List of static values separated by commas. Display values and return values are separated by semicolons. Note that this is only available in the SELECT_LIST function.
p_attributes	Extra HTML parameters you wish to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the null option. Only relevant when P_SHOW_NULL equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the select list.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following example demonstrates a static select list that displays Yes, returns Y, defaults to Y, and generates a F01 form item.

```
SELECT HTMLDB_ITEM.SELECT_LIST(1,'Y','Yes;Y,No;N')
FROM emp
```

SELECT_LIST_FROM_LOV Function

This function dynamically generates select lists from a shared list of values (LOV). Similar to other functions available in the HTMLDB_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST_FROM_LOV(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT,
    p_show_null     IN   VARCHAR2 DEFAULT,
    p_null_value    IN   VARCHAR2 DEFAULT,
    p_null_text     IN   VARCHAR2 DEFAULT,
    p_item_id       IN   VARCHAR2 DEFAULT,
    p_item_label    IN   VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 17–34 describes the parameters available in the SELECT_LIST_FROM_LOV function.

Table 17–34 SELECT_LIST_FROM_LOV Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_list_values parameter.
p_lov	Text name of a application list of values. This list of values must be defined in your application. This parameter is used only by the select_list_from_lov function.
p_attributes	Extra HTML parameters you wish to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the null option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the null option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the select list.

Example

The following demonstrates a select list based on a LOV defined in the application.

```
SELECT HTMLDB_ITEM.SELECT_LIST_FROM_LOV(2,job,'JOB_FLOW_LOV')
FROM emp
```

SELECT_LIST_FROM_LOV_XL Function

This function dynamically generates very large select lists (greater than 32K) from a shared list of values (LOV). Similar to other functions available in the HTMLDB_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST_FROM_LOV_XL(
    p_idx           IN   NUMBER,
    p_value         IN   VARCHAR2 DEFAULT,
    p_lov           IN   VARCHAR2,
    p_attributes    IN   VARCHAR2 DEFAULT,
    p_show_null     IN   VARCHAR2 DEFAULT,
    p_null_value    IN   VARCHAR2 DEFAULT,
    p_null_text     IN   VARCHAR2 DEFAULT,
    p_item_id       IN   VARCHAR2 DEFAULT,
    p_item_label    IN   VARCHAR2 DEFAULT)
    RETURN CLOB;
```

Parameters

Table 17–35 describes the parameters available in the SELECT_LIST_FROM_LOV_XL function.

Table 17–35 SELECT_LIST_FROM_LOV_XL Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_list_values parameter.
p_lov	Text name of a list of values. This list of values must be defined in your application. This parameter is used only by the select_list_from_lov function.
p_attributes	Extra HTML parameters you wish to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the null option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the null option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the select list.

Example

The following demonstrates a select list based on a LOV defined in the application.

```
SELECT HTMLDB_ITEM.SELECT_LIST_FROM_LOV_XL(2,job,'JOB_FLOW_LOV')
FROM emp
```

SELECT_LIST_FROM_QUERY Function

This function dynamically generates a select list from a query. Similar to other functions available in the HTMLDB_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST_FROM_QUERY(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT,
    p_show_null     IN    VARCHAR2 DEFAULT,
    p_null_value    IN    VARCHAR2 DEFAULT,
    p_null_text     IN    VARCHAR2 DEFAULT,
    p_item_id       IN    VARCHAR2 DEFAULT,
    p_item_label    IN    VARCHAR2 DEFAULT,
    p_show_extra    IN    VARCHAR2 DEFAULT)
    RETURN VARCHAR2;
```

Parameters

Table 17–36 describes the parameters available in the SELECT_LIST_FROM_QUERY function.

Table 17–36 SELECT_LIST_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_list_values parameter.
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example: `SELECT dname, deptno FROM dept` Note that this is used only by the SELECT_LIST_FROM_QUERY function.
p_attributes	Extra HTML parameters you wish to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the null option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the null option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the select list.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following demonstrates a select list based on a SQL query.

```
SELECT HTMLDB_ITEM.SELECT_LIST_FROM_QUERY(3,job,'SELECT DISTINCT job FROM emp')
```

```
FROM emp
```

SELECT_LIST_FROM_QUERY_XL Function

This function dynamically generates very large select lists (greater than 32K) from a query. Similar to other functions available in the HTMLDB_ITEM package, these select list functions are designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.SELECT_LIST_FROM_QUERY_XL(
    p_idx           IN    NUMBER,
    p_value         IN    VARCHAR2 DEFAULT,
    p_query         IN    VARCHAR2,
    p_attributes    IN    VARCHAR2 DEFAULT,
    p_show_null     IN    VARCHAR2 DEFAULT,
    p_null_value    IN    VARCHAR2 DEFAULT,
    p_null_text     IN    VARCHAR2 DEFAULT,
    p_item_id       IN    VARCHAR2 DEFAULT,
    p_item_label    IN    VARCHAR2 DEFAULT,
    p_show_extra    IN    VARCHAR2 DEFAULT)
    RETURN CLOB;
```

Parameters

Table 17–37 describes the parameters available in the SELECT_LIST_FROM_QUERY_XL function.

Table 17–37 SELECT_LIST_FROM_QUERY_XL Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically the p_idx parameter is constant for a given column.
p_value	Current value. This value should be a value in the p_list_values parameter.
p_query	SQL query that is expected to select two columns, a display column, and a return column. For example: `SELECT dname, deptno FROM dept` Note that this is used only by the SELECT_LIST_FROM_QUERY_XL function.
p_attributes	Extra HTML parameters you wish to add.
p_show_null	Extra select option to enable the NULL selection. Range of values is YES and NO.
p_null_value	Value to be returned when a user selects the null option. Only relevant when p_show_null equals YES.
p_null_text	Value to be displayed when a user selects the null option. Only relevant when p_show_null equals YES.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the select list.
p_show_extra	Show the current value even if the value of p_value is not located in the select list.

Example

The following demonstrates a select list based on a SQL query.

```
SELECT HTMLDB_ITEM.SELECT_LIST_FROM_QUERY_XL(3,job,'SELECT DISTINCT job FROM emp')
FROM emp
```

TEXTAREA

This function creates text areas

Syntax

```
HTMLDB_ITEM.TEXTAREA(
    p_idx         IN    NUMBER,
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_rows        IN    NUMBER DEAULT 40,
    p_cols        IN    NUMBER DEFAULT 4
    p_attributes  IN    VARCHAR2 DEFAULT,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 17–39 describes the parameters available in the TEXT function.

Table 17–38 TEXTAREA Parameters

Parameter	Description
p_idx	Number to identify the item you wish to generate. The number will determine which G_FXX global is populated. **See Also:** "HTMLDB_APPLICATION" on page 17-52
p_value	Value of a textarea item.
p_rows	Height of the textarea (HTML rows attribute)
p_cols	Width of the textarea (HTML cols attribute).
p_attributes	Extra HTML parameters you wish to add.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the text textarea item.

Example

The following example demonstrates a textarea based on a SQL query.

```
SELECT HTMLDB_ITEM.TEXTAREA(3,ename,5,80) a
FROM emp
```

TEXT Function

This function generates text fields (or text input form items) from a SQL query.

Syntax

```
HTMLDB_ITEM.TEXT(
    p_idx         IN    NUMBER,
```

```
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_size        IN    NUMBER DEFAULT NULL,
    p_maxlength   IN    NUMBER DEFAULT NULL,
    p_attributes  IN    VARCHAR2 DEFAULT NULL,
    p_item_id     IN    VARCHAR2 DEFAULT NULL,
    p_item_label  IN    VARCHAR2 DEFAULT NULL)
```

Parameters

Table 17–39 describes the parameters available in the TEXT function.

Table 17–39 TEXT Parameters

Parameter	Description
p_idx	Number to identify the item you wish to generate. The number will determine which G_FXX global is populated. **See Also:** "HTMLDB_APPLICATION" on page 17-52
p_value	Value of a text field item.
p_size	Controls HTML tag attributes (such as disabled).
p_maxlength	Maximum number of characters that can be entered in the text box.
p_attributes	Extra HTML parameters you wish to add.
p_item_id	HTML attribute ID for the <input> tag.
p_item_label	Label of the text field item.

Example

The following sample query demonstrates how to generate one update field for each row. Note that the ename, sal, and comm columns use the HTMLDB_ITEM.TEXT function to generate an HTML text field for each row. Also, notice that each item in the query is passed an unique p_idx parameter to ensure that each column is stored in its own array.

```
SELECT
  empno,
  HTMLDB_ITEM.HIDDEN(1,empno)||
  HTMLDB_ITEM.TEXT(2,ename) ename,
  HTMLDB_ITEM.TEXT(3,job) job,
  mgr,
  HTMLDB_ITEM.DATE_POPUP(4,rownum,hiredate,'dd-mon-yyyy') hiredate,
  HTMLDB_ITEM.TEXT(5,sal) sal,
  HTMLDB_ITEM.TEXT(6,comm) comm,
  deptno
FROM emp
ORDER BY 1
```

TEXT_FROM_LOV Function

Use this function to display an item as text, deriving the display value of the named LOV.

Syntax

```
HTMLDB_ITEM.TEXT_FROM_LOV (
```

```
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_lov         IN    VARCHAR2,
    p_null_text   IN    VARCHAR2 DEFAULT '%')
    RETURN VARCHAR2;
```

Parameters

Table 17–40 describes the parameters available in the TEXT_FROM_LOV function.

Table 17–40 TEXT_FROM_LOV Parameters

Parameter	Description
p_value	Value of a field item.
p_lov	Text name of a shared list of values. This list of values must be defined in your application.
p_null_text	Value to be displayed when the value of the field item is null or a corresponding entry is not located for the value p_value in the list of values.

Example

The following example demonstrates how to derive the display value from a named LOV (EMPNO_ENAME_LOV).

```
SELECT HTMLDB_ITEM.TEXT_FROM_LOV(empno,'EMPNO_ENAME_LOV') c FROM emp
```

TEXT_FROM_LOV_QUERY Function

Use this function to display an item as text, deriving the display value from a list of values query.

Syntax

```
HTMLDB_ITEM.TEXT_FROM_LOV_QUERY (
    p_value       IN    VARCHAR2 DEFAULT NULL,
    p_query       IN    VARCHAR2,
    p_null_text   IN    VARCHAR2 DEFAULT '%')
    RETURN VARCHAR2;
```

Parameters

Table 17–40 describes the parameters available in the TEXT_FROM_LOV_QUERY function.

Table 17–41 TEXT_FROM_LOV_QUERY Parameters

Parameter	Description
p_value	Value of a field item.
p_query	SQL query that is expected to select two columns, a display column and a return column. For example: `SELECT dname, deptno FROM dept`
p_null_text	Value to be displayed when the value of the field item is null or a corresponding entry is not located for the value p_value in the list of values query.

Example

The following how to derive the display value from a query.

```
SELECT HTMLDB_ITEM.TEXT_FROM_LOV_QUERY(empno,'SELECT ename, empno FROM emp') c
from emp
```

RADIOGROUP Function

This function generates a radio group from a SQL query.

Syntax

```
HTMLDB_ITEM.RADIOGROUP(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT,
    p_selected_value   IN    VARCHAR2 DEFAULT,
    p_display          IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_onblur           IN    VARCHAR2 DEFAULT,
    p_onchange         IN    VARCHAR2 DEFAULT,
    p_onfocus          IN    VARCHAR2 DEFAULT,)
    RETURN VARCHAR2;
```

Parameters

Table 17–42 describes the parameters available in the RADIOGROUP function.

Table 17–42 RADIOGROUP Parameters

Parameter	Description
p_idx	Number which determines which HTMLDB_APPLICATION global will be used. Valid range of values is 1 to 50.For example 1 creates F01 and 2 creates F02.
p_value	Value of the radio group.
p_selected_value	Value that should be "on", or selected.
p_display	Text to display next to the radio option.
p_attributes	Extra HTML parameters you wish to add.
p_onblur	JavaScript to execute in the onBlur event.
p_onchange	JavaScript to execute in the onChange event.
p_onfocus	JavaScript to execute in the onFocus event.

Example

The following example demonstrates how to select department 20 from the emp table as a default in a radio group.

```
SELECT HTMLDB_ITEM.CHECKBOX(1,deptno,'20',dname) dt
FROM   dept
ORDER  BY 1
```

POPUP_FROM_LOV Function

This function generates an HTML popup select list from an application list of values (LOV). Like other available functions in the HTMLDB_ITEM package, POPUP_FROM_LOV is designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.POPUP_FROM_LOV(

    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 17–43 describes the some parameters in the POPUP_FROM_LOV function.

Table 17–43 POPUP_FROM_LOV Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.
p_value	Form element current value. This value should be one of the values in the p_lov_name parameter.
p_lov_name	Named LOV used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field which posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle HTML DB must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV which passes a value back to a form element.
p_escape_html	Replacements for special characters that require an escaped equivalent. ▪ <lt; for < ▪ > for > ▪ & for & Range of values is YES and NO. If YES, special characters will be escaped. This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a more narrow set of results.
p_attributes	Additional HTML attributes to use for the form item.

Table 17–43 (Cont.) POPUP_FROM_LOV Parameters

Parameter	Description
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup from a LOV named DEPT.

```
SELECT HTMLDB_ITEM.POPUP_FROM_LOV (1,deptno,'DEPT_LOV') dt
FROM emp
```

POPUP_FROM_QUERY Function

This function generates an HTML popup select list from a query. Like other available functions in the HTMLDB_ITEM package, POPUP_FROM_QUERY is designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.POPUP_FROM_QUERY(

    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 17–44 describes the parameters in the POPUP_FROM_QUERY function.

Table 17–44 POPUP_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column.
p_value	Form element current value. This value should be one of the values in the p_lov_query parameter.
p_lov_query	SQL query that is expected to select two columns (a display column and a return column). For example: `SELECT dname, deptno FROM dept`
p_width	Width of the text box.

Table 17–44 (Cont.) POPUP_FROM_QUERY Parameters

Parameter	Description
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field which posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle HTML DB must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV which passes a value back to a form element.
p_escape_html	Replacements for special characters that require an escaped equivalent. ■ < for < ■ > for > ■ & for & Range of values is YES and NO. If YES, special characters will be escaped. This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a more narrow set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates a sample query the generates a popup select list from the emp table.

```
SELECT HTMLDB_ITEM.POPUP_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt
FROM emp
```

POPUPKEY_FROM_LOV Function

This function generates a popup key select list from a shared list of values (LOV). Like other available functions in the HTMLDB_ITEM package, POPUPKEY_FROM_LOV is designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.POPUPKEY_FROM_LOV(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT,
    p_lov_name         IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
```

```
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    RETURN VARCHAR2;
```

Although the text field associated with the popup displays in the first column in the LOV query, the actual value is specified in the second column in the query.

Parameters

Table 17–45 describes the some parameters in the POPUPKEY_FROM_LOV function.

Table 17–45 POPUPKEY_FROM_LOV Parameters

Parameter	Description
p_idx	Identifies a form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example: SELECT HTMLDB_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt, HTMLDB_ITEM.HIDDEN(3,empno) eno
p_value	Indicates the current value. This value should be one of the values in the P_LOV_NAME parameter.
p_lov_name	Identifies a named LOV used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field which posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle HTML DB must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV which passes a value back to a form element.
p_escape_html	Replacements for special characters that require an escaped equivalent. ▪ < for < ▪ > for > ▪ & for & This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a more narrow set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.

Example

The following example demonstrates how to generate a popup key select list from a shared list of values (LOV).

```
SELECT HTMLDB_ITEM.POPUPKEY_FROM_LOV (1,deptno,'DEPT') dt
FROM emp
```

POPUPKEY_FROM_QUERY Function

This function generates a popup key select list from a SQL query. Like other available functions in the HTMLDB_ITEM package, POPUPKEY_FROM_QUERY is designed to generate forms with F01 to F50 form array elements.

Syntax

```
HTMLDB_ITEM.POPUPKEY_FROM_QUERY(
    p_idx              IN    NUMBER,
    p_value            IN    VARCHAR2 DEFAULT,
    p_lov_query        IN    VARCHAR2,
    p_width            IN    VARCHAR2 DEFAULT,
    p_max_length       IN    VARCHAR2 DEFAULT,
    p_form_index       IN    VARCHAR2 DEFAULT,
    p_escape_html      IN    VARCHAR2 DEFAULT,
    p_max_elements     IN    VARCHAR2 DEFAULT,
    p_attributes       IN    VARCHAR2 DEFAULT,
    p_ok_to_query      IN    VARCHAR2 DEFAULT,
    p_item_id          IN    VARCHAR2 DEFAULT NULL,
    p_item_label       IN    VARCHAR2 DEFAULT NULL)
    RETURN VARCHAR2;
```

Parameters

Table 17–46 describes the some parameters in the POPUPKEY_FROM_QUERY function.

Table 17–46 POPUPKEY_FROM_QUERY Parameters

Parameter	Description
p_idx	Form element name. For example, 1 equals F01 and 2 equals F02. Typically, p_idx is a constant for a given column. Because of the behavior of POPUPKEY_FROM_QUERY, the next index value should be p_idx + 1. For example: `SELECT HTMLDB_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept') dt, HTMLDB_ITEM.HIDDEN(3,empno) eno`
p_value	Form element current value. This value should be one of the values in the P_LOV_QUERY parameter.
p_lov_query	LOV query used for this popup.
p_width	Width of the text box.
p_max_length	Maximum number of characters that can be entered in the text box.

Table 17–46 (Cont.) POPUPKEY_FROM_QUERY Parameters

Parameter	Description
p_form_index	HTML form on the page in which an item is contained. Defaults to 0 and rarely used. Only use this parameter when it is necessary to embed a custom form in your page template (such as a search field which posts to a different Web site). If this form comes before the #FORM_OPEN# substitution string, then its index is zero and the form opened automatically by Oracle HTML DB must be referenced as form 1. This functionality supports the JavaScript used in the popup LOV which passes a value back to a form element.
p_escape_html	Replacements for special characters that require an escaped equivalent. ▪ < for < ▪ > for > ▪ & for & This parameter is useful if you know your query will return illegal HTML.
p_max_elements	Limit on the number of rows that can be returned by your query. Limits the performance impact of user searches. By entering a value in this parameter, you force the user to search for a more narrow set of results.
p_attributes	Additional HTML attributes to use for the form item.
p_ok_to_query	Range of values is YES and NO. If YES, a popup returns first set of rows for the LOV. If NO, a search is initiated to return rows.
p_item_id	ID attribute of the form element.
p_item_label	Invisible label created for the item.

Example

The following example demonstrates how to generate a popup select list from a SQL query.

```
SELECT HTMLDB_ITEM.POPUPKEY_FROM_QUERY (1,deptno,'SELECT dname, deptno FROM dept')
dt
FROM emp
```

HTMLDB_APPLICATION

The HTMLDB_APPLICATION package is a PL/SQL package that implements the Oracle HTML DB rendering engine. You can use this package to take advantage of a number of global variables. Table 17–47 describes the global variables available in HTMLDB_APPLICATION.

Table 17–47 Global Variables Available in HTMLDB_APPLICATION

Global Variable	Description
G_USER	Specifies the currently logged in user.
G_FLOW_ID	Specifies the ID of the currently running application.
G_FLOW_STEP_ID	Specifies the ID of the currently running page.
G_FLOW_OWNER	Specifies the schema to parse for the currently running application.

Topics in this section include:

- Referencing Arrays
- Referencing Values Within an On Submit Process
- Converting an Array to a Single Value

Referencing Arrays

Items are typically HTML form elements such as text fields, select lists and check boxes. When you create a new form item using a wizard, the wizard uses a standard naming format. The naming format provides a handle so you can retrieve the value of the item later on.

If you need to create your own items, you can access them after a page is submitted by referencing `HTMLDB_APPLICATION.G_F01` to `HTMLDB_APPLICATION.G_F50` arrays. You can create your own HTML form fields by providing the input parameters using the format `F01`, `F02`, `F03` and so on. You can create up to 50 input parameters ranging from `F01` to `F50`. Consider the following example:

```
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="some value">

<TEXTAREA NAME="F02" ROWS=4 COLS=90 WRAP="VIRTUAL">this is the example of a text
area.</TEXTAREA>

<SELECT NAME="F03" SIZE="1">
<OPTION VALUE="abc">abc
<OPTION VALUE="123">123
</SELECT>
```

Since the `F01` to `F50` input items are declared as PL/SQL arrays, you can have multiple items named the same value. For example:

```
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 1">
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" ALUE="array element 2">
<INPUT TYPE="text" NAME="F01" SIZE="32" MAXLENGTH="32" VALUE="array element 3">
```

Note that following PL/SQL produces the same HTML as show in the previous example.

```
FOR i IN 1..3 LOOP
HTMLDB_ITEM.TEXT(P_IDX        => 1,
 p_value      =>'array element '||i ,
 p_size       =>32,
 p_maxlength  =>32);
END LOOP;
```

Referencing Values Within an On Submit Process

You can reference the values posted by an HTML form using the PL/SQL variable `HTMLDB_APPLICATION.G_F01` to `HTMLDB_APPLICATION.G_F50`. Since this element is an array you can reference values directly. For example:

```
FOR i IN 1.. HTMLDB_APPLICATION.G_F01.COUNT LOOP
    htp.p('element '||I||' has a value of '||HTMLDB_APPLICATION.G_F01(i));
END LOOP;
```

Converting an Array to a Single Value

You can also use Oracle HTML DB public utility functions to convert an array into a single value. For example:

```
htp.p(HTMLDB_UTIL.TABLE_TO_STRING(HTMLDB_APPLICATION.G_F01));
```

This function is enables you to reference `G_F01` to `G_F50` values in an application process that performs actions on data. The following sample process demonstrates the insertion of values into an table:

```
FOR i IN 1..HTMLDB_APPLICATION.G_F01.COUNT LOOP
    INSERT INTO my_table (my_column) VALUES HTMLDB_APPLICATION.G_F01(i);
END LOOP;
```

HTMLDB_CUSTOM_AUTH

You can use `HTMLDB_CUSTOM_AUTH` to perform various operations related to authentication and session management.

Topics in this section include:

APPLICATION_PAGE_ITEM_EXISTS Function

This function checks for the existence of page level item within an application. This function requires the parameter `p_item_name`. This function returns a boolean value (true or false).

Syntax

```
FUNCTION APPLICATION_PAGE_ITEM_EXISTS(
    p_item_name   IN    VARCHAR2)
RETURN BOOLEAN;
```

CURRENT_PAGE_IS_PUBLIC Function

This function checks whether the current page's authentication attribute is set to **Page Is Public** and returns a boolean value (true or false)

> **See Also:** "Editing Page Attributes" on page 7-16 and "Security" on page 7-19 for information on setting this page attribute

Syntax

```
FUNCTION CURRENT_PAGE_IS_PUBLIC
RETURN BOOLEAN;
```

DEFINE_USER_SESSION Procedure

This procedure combines the SET_USER and SET_SESSION_ID functions to create one call.

Syntax

```
PROCEDURE DEFINE_USER_SESSION(
    p_user         IN    VARCHAR2)
    p_session_id   IN    NUMBER);
```

GET_COOKIE_PROPS Procedure

This procedure obtains the properties of the session cookie used in the current authentication scheme for the specified application. These properties can be viewed directly in the Application Builder by viewing the authentication scheme attributes.

Syntax

```
HTMLDB_CUSTOM_AUTH.GET_COOKIE_PROPS(
 p_app_id                      IN  NUMBER,
 p_cookie_name                 OUT VARCHAR2,
 p_cookie_path                 OUT VARCHAR2,
 p_cookie_domain               OUT VARCHAR2);
```

Parameters

Table 17–48 describes the parameters available in the GET_COOKIE_PROPS procedure.

Table 17–48 GET_COOKIE_PROPS Parameters

Parameter	Description
p_app_id	An application ID in the current workspace.
p_cookie_name	The cookie name.
p_cookie_path	The cookie path.
p_cookie_domain	The cookie domain.

Example

```
DECLARE
    l_cookie_name   varchar2(256);
    l_cookie_path   varchar2(256);
    l_cookie_domain varchar2(256);
BEGIN
HTMLDB_CUSTOM_AUTH.GET_COOKIE_PROPS (
    p _cookie_name   => l_cookie_name,
    p _cookie_path   => l_cookie_path,
    p _cookie_domain => l_cookie_domain);
END;
```

GET_LDAP_PROPS Procedure

This procedure obtains the LDAP attributes of the current authentication scheme for the current application. These properties can be viewed directly in Application Builder by viewing the authentication scheme attributes.

Syntax

```
HTMLDB_CUSTOM_AUTH.GET_LDAP_PROPS(
 p_ldap_host                OUT VARCHAR2,
 p_ldap_port                OUT NUMBER,
 p_ldap_dn                  OUT VARCHAR2,
 p_ldap_edit_function       OUT VARCHAR2);
```

Parameters

Table 17–49 describes the parameters available in the GET_LDAP_PROPS procedure.

Table 17–49 GET_LDAP_PROPS Parameters

Parameter	Description
p_ldap_host	LDAP host name.
p_ldap_port	LDAP port number.
p_ldap_host	LDAP DN string.
p_ldap_host	LDAP host name.
p_ldap_edit_function	LDAP edit function name.

Example

```
DECLARE
    l_ldap_host          varchar2(256);
    l_ldap_port          number;
    l_ldap_dn            varchar2(256);
    l_ldap_edit_function varchar2(256);
BEGIN
HTMLDB_CUSTOM_AUTH.GET_LDAP_PROPS (
    p_ldap_host       => l_ldap_host,
    p_ldap_port       => l_ldap_port,
    p_ldap_dn         => l_ldap_dn,'
    p_ldap_edit_function => l_ldap_edit_function);
END;
```

GET_NEXT_SESSION_ID Function

This function generates the next session ID from the Oracle HTML DB sequence generator. This function returns a number.

Syntax

```
FUNCTION GET_NEXT_SESSION_ID
RETURN NUMBER;
```

GET_SESSION_ID_FROM_COOKIE Function

This function returns the Oracle HTML DB session ID located by the session cookie in the context of a page request in the current browser session.

Syntax

```
HTMLDB_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE;
RETURN NUMBER;
```

Example

```
DECLARE VAL NUMBER;
BEGIN
  VAL := HTMLDB_CUSTOM_AUTH.GET_SESSION_ID_FROM_COOKIE;
END;
```

GET_USERNAME Function

This function returns user name registered with the current Oracle HTML DB session in the internal sessions table. This user name is usually the same as the authenticated user running the current page.

Syntax

```
HTMLDB_CUSTOM_AUTH.GET_USERNAME;
RETURN VARCHAR2;
```

Example

```
DECLARE VAL VARCHAR2(256);
BEGIN
  VAL := HTMLDB_CUSTOM_AUTH.GET_USERNAME;
END;
```

GET_SECURITY_GROUP_ID Function

This function returns a number with the value of the security group ID that identifies the workspace of the current user.

Syntax

```
FUNCTION GET_SECURITY_GROUP_ID
RETURN NUMBER;
```

GET_SESSION_ID Function

This function returns HTMLDB_APPLICATION.G_INSTANCE global variable. GET_SESSION_ID returns a number.

Syntax

```
PROCEDURE GET_SESSION_ID
RETURN NUMBER;
```

GET_USER Function

This function returns the HTMLDB_APPLICATION.G_USER global variable (VARCHAR2).

Syntax

```
FUNCTION GET_USER
RETURN VARCHAR2;
```

IS_SESSION_VALID Function

This function is a boolean result obtained from executing the current application's authentication scheme to determine if a valid session exists. This function returns the boolean result of the authentication scheme's page sentry.

Syntax

```
HTMLDB_CUSTOM_AUTH.IS_SESSION_VALID;
RETURN BOOLEAN;
```

Example

```
DECLARE VAL VARCHAR2(256);
BEGIN
  VAL := HTMLDB_CUSTOM_AUTH.IS_SESSION_VALID;
END;
```

LOGIN Procedure

Also referred to as the "Login API," this procedure performs authentication and session registration.

Syntax

```
HTMLDB_CUSTOM_AUTH.LOGIN(
 p_uname                    IN  VARCHAR2,
 p_password                 IN  VARCHAR2,
 p_session_id               IN  VARCHAR2,
 p_app_page                 IN  VARCHAR2,
 p_entry_point              IN  VARCHAR2,
 p_preserve_case            IN  BOOLEAN);
```

Parameter

Table 17–50 describes the parameters available in the LOGIN procedure.

Table 17–50 LOGIN Parameters

Parameter	Description
p_uname	Login name of the user.
p_password	Clear text user password.
p_session_id	Current Oracle HTML DB session ID.
p_app_page	Current application ID. After login page separated by a colon (:).
p_entry_point	Internal use only.
p_preserve_case	If true, do not upper p_uname during session registration

Example

```
BEGIN
HTMLDB_CUSTOM_AUTH.LOGIN (
    p_uname       => 'SCOTT',
    p_password    => 'secret99',
    p_session_id  => V('APP_SESSION'),
    p_app_page    => :APP_ID||':1');
END;
```

Note: :Do not use bind variable notations for p_session_id argument.

LOGOUT Procedure

This procedure effects a logout from the current session by unsetting the session cookie and redirecting to a new location.

Syntax

```
HTMLDB_CUSTOM_AUTH.LOGOUT(
 p_this_app                    IN VARCHAR2,
 p_next_app_page_sess          IN VARCHAR2,
 p_next_url                    IN VARCHAR2);
```

Parameter

Table 17–51 describes the parameters available in the LOGOUT procedure.

Table 17–51 LOGOUT Parameters

Parameter	Description
p_this_app	Current application ID.
p_next_app_page_sess	Application and page ID to redirect to. Separate multiple pages using a colon (:) and optionally followed by a colon (:) and the session ID (if control over the session ID is desired).
p_next_url	URL to redirect to (use this instead of p_next_app_page_sess).

Example

```
BEGIN
HTMLDB_CUSTOM_AUTH.LOGOUT (
```

```
    p_this_app            => '1000',
    p_next_app_page_sess  => '1000:99');
END;
```

POST_LOGIN Procedure

This procedure performs session registration, assuming the authentication step has been completed. It can be called only from within an Oracle HTML DB application page context.

Syntax

```
HTMLDB_CUSTOM_AUTH.POST_LOGIN(
 p_uname                    IN  VARCHAR2,
 p_session_id               IN  VARCHAR2,
 p_app_page                 IN  VARCHAR2,
 p_preserve_case            IN  BOOLEAN);
```

Parameter

Table 17–52 describes the parameters available in the `POST_LOGIN` procedure.

Table 17–52 POST_LOGIN Parameters

Parameter	Description
`p_uname`	Login name of user.
`p_session_id`	Current Oracle HTML DB session ID.
`p_app_page`	Current application ID and after login page separated by a colon (:).
`p_preserve_case`	If true, do not include `p_uname` in upper case during session registration.

Example

```
BEGIN
HTMLDB_CUSTOM_AUTH.POST_LOGIN (
    p_uname       => 'SCOTT',
    p_session_id  => V('APP_SESSION'),
    p_app_page    => :APP_ID||':1');
END;
```

SESSION_ID_EXISTS Function

This function returns a boolean result based on the global package variable containing the current Oracle HTML DB session ID. Returns true if the result is a positive number. returns false if the result is a negative number.

Syntax

```
FUNCTION SESSION_ID_EXISTS
RETURN BOOLEAN;
```

Example

```
DECLARE VAL BOOLEAN;
```

```
BEGIN
  VAL := HTMLDB_CUSTOM_AUTH.SESSION_ID_EXISTS;
END;
```

SET_USER Procedure

This procedure sets the HTMLDB_APPLICATION.G_USER global variable. SET_USER requires the parameter P_USER (VARCHAR2) which defines a user ID.

Syntax

```
PROCEDURE SET_USER(
    p_user   IN    VARCHAR2)
```

SET_SESSION_ID Procedure

This procedure sets HTMLDB_APPLICATION.G_INSTANCE global variable. SET_SESSION_ID returns a number. This procedure requires the parameter P_SESSION_ID (NUMBER) which specifies a session ID.

Syntax

```
PROCEDURE SET_SESSION_ID(
    p_session_id    IN    NUMBER)
```

SET_SESSION_ID_TO_NEXT_VALUE Procedure

This procedure combines the operation of GET_NEXT_SESSION_ID and SET_SESSION_ID in one call.

Syntax

```
PROCEDURE SETsN_ID_TO_NEXT_VALUE;
```

Part III

Administration

Part III describes all tasks performed by an Oracle HTML DB administrator. An Oracle HTML DB administrator manages an entire Oracle HTML DB development environment instance through the Oracle HTML DB Administration Services application. Common Oracle HTML DB administrator tasks include creating and managing workspaces, translating an application, and managing activities, log files, and sessions.

Part III contains the following chapter:

- "Managing an Oracle HTML DB Hosted Service"

18

Managing an Oracle HTML DB Hosted Service

This section describes tasks an Oracle HTML DB administrator performs when administering an Oracle HTML DB hosted service.

This section contains the following topics:

What is an Oracle HTML DB Administrator?

In the Oracle HTML DB development environment, users log in to a shared work area called a workspace. Users are divided into three primary roles:

- **Developer**

  A developer can create and edit applications.

- **Workspace administrator**

A Workspace administrator performs administrator tasks specific to their workspace such as managing user accounts, monitoring workspace activity, and viewing log files.

- **Oracle HTML DB administrator**

  An Oracle HTML DB administrator is a superuser that manages the entire hosted instance. To perform these tasks, an Oracle HTML DB administrator logs into the Oracle HTML DB Administration Services application.

  > **See Also:** Refer to appropriate installation guide for your platform for more information on installing Oracle HTML DB

Logging in to Oracle HTML DB Administration Services

Oracle HTML DB administrators are responsible for managing an entire Oracle HTML DB instance. To perform these tasks, an Oracle HTML DB administrator logs into the Oracle HTML DB Administration Services application.

To log into Oracle HTML DB Administration Services:

1. In a Web browser, navigate to the Oracle HTML DB Administration Services application:

   ```
   http://hostname:port/pls/htmldb/htmldb_admin
   ```

   Where:

   - *hostname* is the name of the system where Oracle HTTP Server is installed.
   - *port* is the port number assigned to Oracle HTTP Server. In a default installation, this number is 7777. You can find information about your Oracle HTTP Server installation's port number from either of the following files:
     - *ORACLE_BASE**ORACLE_HOME*\install\portlist.ini
     - *ORACLE_BASE**ORACLE_HOME*\Apache\Apache\conf\httpd.conf
   - *htmldb* is the database access descriptor (DAD) defined in the mod_plsql configuration file.

   The Login page appears.
2. In Username, enter `admin`.
3. In Password, enter the Oracle HTML DB administrator account password you specified when you installed Oracle HTML DB.
4. Click **Login**.

   Oracle HTML DB Administration Services appears.

   > **See Also:** Refer to the appropriate installation guide for more information on installing Oracle HTML DB

Determining the HTML DB Engine Schema

Oracle HTML DB administrators may need to perform certain actions within the HTML DB engine schema. For example, in order for an Oracle HTML DB administrator to have the ability to assign Oracle default schemas, the database administrator (DBA) must explicitly grant the privilege by running the `HTMLDB_SITE_ADMIN_PRIVS.ALLOW` procedure within the HTML DB engine schema.

> **See Also:** "Understanding Oracle Default Schema Restrictions" on page 18-3 for more information on the HTMLDB_SITE_ADMIN_PRIVS.ALLOW procedure

To determine the current HTML DB engine schema for your Oracle HTML DB instance:

1. Use SQL*Plus to connect to the database.
2. Run the following query in a schema with DBA privileges (for example, SYSTEM).

```
SELECT TABLE_OWNER FROM all_synonyms
WHERE SYNONYM_NAME = 'WWV_FLOW' and OWNER = 'PUBLIC'
```

Managing the Schemas Associated with a Workspace

When a user logs into the Oracle HTML DB they log in to a shared work area called a workspace. Each workspace can have multiple associated schemas. By associating a workspace with a schema, developers in that workspace can:

- Build applications that interact with the database objects in that schema.
- Create new database objects in that schema.

To view the schema associated with a workspace:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Workspaces, select **Manage Schema to Workspace Assignments**.

   The Schemas Provisioned by Workspace page appears.
3. To view information about an existing schema, click the **Edit** icon.
4. To create a new schema association, click **Create**.
5. Follow the on-screen instructions.

Understanding Oracle Default Schema Restrictions

When Oracle HTML DB installs, the Oracle HTML DB administrator does not have the ability to assign Oracle default schemas to workspaces. Default schemas (such as SYS, SYSTEM, and RMAN) are reserved by Oracle for various product features and for internal use. Access to a default schema can be a very powerful privilege. For example, a workspace with access to the default schema SYSTEM can run applications that parse as the SYSTEM user.

In order for an Oracle HTML DB administrator to have the ability to assign Oracle default schemas, the database administrator (DBA) must explicitly grant the privilege.

The DBA can remove this restriction and grant the privilege using SQL*Plus to run the HTMLDB_SITE_ADMIN_PRIVS.ALLOW procedure within the HTML DB engine schema. For example:

```
EXEC HTMLDB_SITE_ADMIN_PRIVS.ALLOW;
```

The DBA can reimpose this restriction and revoke this privilege using the HTMLDB_SITE_ADMIN_PRIVS.RESTRICT procedure within the HTML DB engine schema. For example:

```
EXEC HTMLDB_SITE_ADMIN_PRIVS.RESTRICT;
```

The DBA can determine the current status of the privilege using SQL*Plus to run the `HTMLDB_SITE_ADMIN_PRIVS.REPORT` procedure within the HTML DB engine schema. For example:

```
SET SERVEROUTPUT ON
EXEC HTMLDB_SITE_ADMIN_PRIVS.REPORT;
```

`EXEC HTMLDB_SITE_ADMIN_PRIVS.REPORT` returns number of rows in the `wwv_flow_restrict_admin` table. `0` rows means that the Oracle HTML DB administrator is restricted and may not assign Oracle default schemas to workspaces. One or more rows means that the Oracle HTML DB administrator may assign Oracle default schemas to workspaces.

> **See Also:** "Determining the HTML DB Engine Schema" on page 18-2

Creating a Workspace

When a user logs into the Oracle HTML DB they log in to a shared work area called a workspace. Each workspace is an area within the Oracle HTML DB development environment where multiple developers can create applications. Each workspace has a unique ID and name. In order to make changes to their workspace, Workspace administrators submit change request to an Oracle HTML DB administrator. Only an Oracle HTML DB administrator can create a new workspace.

Topics in this section include:

- About Workspace Provisioning
- Specifying a Provisioning Mode
- Managing Service and Change Requests
- Creating a Workspace Without a Request
- Viewing Workspace Reports

> **See Also:** "Managing Development Services" on page 13-10, "Viewing Workspace Reports" on page 18-6, and "Removing a Workspace" on page 18-12

About Workspace Provisioning

When an Oracle HTML DB administrator creates a new workspace with a new tablespace, the datafile for the new tablespace is managed by Oracle-managed files.

Oracle-managed files simplifies the administration of the Oracle database and eliminates the need for the database administrator (DBA) to directly manage the operating system files that comprise the database. Using Oracle-managed files the DBA specifies operations in terms of database objects rather than filenames. The datafile for the any new tablespaces will be named according to the Oracle-managed files conventions and the placement of these files will be determined by the database initialization parameter `DB_CREATE_FILE_DEST`.

If the Oracle-Managed Files is not enabled, the datafile will be created in the same directory as the first datafile of the tablespace in which Oracle HTML DB was installed.

> **See Also:** *Oracle Database Administrator's Guide* for more information on Oracle-managed files

Specifying a Provisioning Mode

As an Oracle HTML DB administrator, you determine how the process of creating (or provisioning) a workspace works for your Oracle HTML DB development environment.

In **manual** provision mode, an Oracle HTML DB administrator creates new workspaces and notifies the Workspace administrator of the login information. In **request** provision mode, users request workspaces directly in a self-service fashion. In this scenario, users use a link on the login page to access a request form. Once the workspace request has been granted, they are e-mailed the appropriate login information.

To specify a provisioning mode:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Toggle Provisioning Status**.
3. Select one of the following:
   - **Manual**
   - **Request**
4. Click **Apply Changes**.

Note: Before users can request a workspace or change their passwords, an Oracle HTML DB administrator must configure environment preferences to facilitate the workflow component.

See Also: "Managing Environment Preferences" on page 18-17 and "About SERVICE_REQUEST_FLOW" on page 18-18

Creating a Workspace Without a Request

Administrators can create a workspace manually by running the Provision Workspace Wizard. You can access this wizard from either the Oracle HTML DB Administration Services home page or the Workspace Administration tab.

To create a workspace manually:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Workspaces, select **Create New Workspace**.

   The Provision Workspace Wizard appears.
3. Specify a workspace name and description and click **Next**.
4. Select a schema, or enter the name for a new schema, followed by a password, and initial disk space quota and click **Next**.
5. Specify a Workspace administrator by providing a username, password, and e-mail address and click **Next**.
6. Confirm your selections and click **Provision**.

Viewing Workspace Reports

Oracle HTML DB administrators can view detailed information about a specific workspace by viewing the Workspace Utilization Report.

To view a workspace report:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Workspaces, select **Report Workspace Attributes**.
3. Select a workspace from the Workspace list and click **Go**.

   The Workspace Utilization Report appears. Table 18–1 describes the various sections of the Workspace Utilization Report.

Table 18–1 Workspace Utilization Report

Report	Description
Workspace Information	Displays high level information about the current workspace.
Workspace Schemas	Manage workspace to schema mappings. **See Also:** "Managing the Schemas Associated with a Workspace" on page 18-3
Workspace Schema Space Utilization	Displays a report that details tablespace utilization.
Workspace Applications	Displays a report that lists all applications within the current workspace.
Developers	Displays a report that lists all application developers within the current workspace.
Workspace Users	Enables administrators to manage user accounts. **See Also:** "Managing Users in an Oracle HTML DB Instance" on page 18-9
Workspace Database Objects	Displays a report that lists objects used in the current workspace.
Service Change Requests	Enables administrators to manage change requests for the current workspace, or to view a report of all change requests in an Oracle HTML DB development instance. **See Also:** "Managing Service and Change Requests" on page 18-6
Developer Activity	Displays a report that details developer activity by date.

See Also: "Creating a Workspace" on page 18-4 and "Removing a Workspace" on page 18-12

Managing Service and Change Requests

Oracle HTML DB administrators can make modifications to a workspace (such as such as adding an additional schema or increasing the disk space limit) by approving a change request from a Workspace administrator.

Topics in this section include:

- Viewing a Pending Service or Change Request

Viewing a Pending Service or Change Request

You can view existing service requests and change requests from the Notifications list on the Administration home page, or from the Service Requests or Change Requests pages.

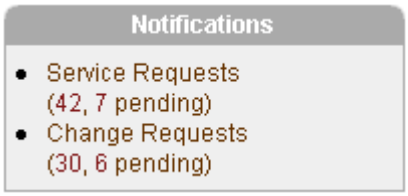
Topics in this section include:

Viewing a Pending Request from the Notifications List

To view pending service and change requests from the Notifications list:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Locate the Notifications list on the lower right side of the page. (See Figure 18–1.)

Figure 18–1 Notifications List



   The Notifications list displays a summary of total and pending service and change requests.

3. To view additional details, click the appropriate service request or change request number.

   The appropriate Change Request page appears.

Viewing a Request from the Workspace Utilization Report

To view pending requests from the Workspace Utilization Report:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under **Manage Workspaces**, select **Manage Workspaces**.
3. Locate a workspace as follows:
   - To locate a specific workspace, type the workspace name in the Search field and click **Go**.
   - To view all workspaces, leave the Search field blank and click **Go**.
4. To view details about a specific workspace, click the **View** icon to the left of the workspace name.

The Workspace Utilization Report appears.

5. Under Available Reports, click **Service Change Requests**.
6. Select a specific request, or click **View All Change Requests**.

Viewing Requests from the Service Requests Page

To view service requests from the Service Requests page:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Service Requests**.
3. From Status, select the type of requests you wish to view.
4. To view request details, click the **Edit** icon associated with the appropriate request.

Viewing Requests from the Change Requests Page

To view change requests from the Service Requests page:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Change Requests**.
3. From Status, select the type of requests you wish to view.
4. To view request details, click the **Edit** icon associated with the appropriate request.

Approving a Service or Change Request

To approve a pending service request:

1. Navigate to the appropriate request page as described in "Viewing a Pending Service or Change Request" on page 18-7.
2. Click **Adjust**.

   The Adjust Request page appears.
3. From Project Status, select one of the following:
   - To approve the request, click **Approved**.
   - To decline the request, click **Declined**.
4. Follow the on-screen instructions.

> **Note:** Be cautious when setting the Project Status **Requested**. Although **Requested** enables you to reprovision a workspace, it could result in data corruption due to the manner in which accounts are provisioned. The provisioning system assumes Requested service requests do not have the corresponding schemas and dictionary entries for a workspace administrator or developers. If you need to change the Project Status for an **Approved** workspace to **Requested**, terminate the service first and then change the status to Requested.

To approve a pending change request:

1. Navigate to the Service Change Request page as described in "Viewing a Request from the Workspace Utilization Report" on page 18-7.

2. Click **View Request**.

   The Process Change Request page appears.

3. Review the displayed reports.
4. Select one of the following:
   - To approve a request for a schema, click **Create Schema**.
   - To approve a request for additional disk space, click **Provision Space**.
   - To approve a request to terminate the service, click **Terminate Service**
   - To deny a request, click **Deny Request**.
5. Follow the on-screen instructions.

Deleting an Existing Request

To delete an existing service or change request:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Navigate to the appropriate request page. (See "Viewing a Pending Service or Change Request" on page 18-7.)
3. From Status, select the type of request you wish to delete.
4. Click the **Edit** icon associated with the request you wish to delete.
5. When the request appears, click **Terminate or Delete** if the request is still active, or click **Delete** if the request has already been terminated.

Managing Users in an Oracle HTML DB Instance

Oracle HTML DB administrators can manage all user accounts within an Oracle HTML DB instance on the Manage Application Developers and Users page. User accounts are particularly useful if a workspace utilizes HTML DB Authentication.

See Also:

- "Managing a Development Workspace" on page 13-1
- "About HTML DB Account Credentials" on page 14-6 for more information on implementing HTML DB Authentication

To create a new user account:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Workspaces, select **Manage Application Developers**.

   The Manage Application Developers and Users page appears.

3. Click **Create**.
4. Under User Attributes, enter the appropriate information. Fields marked with a red asterisk (*) are required.
5. Under Password, type a case-sensitive password for this account.
6. Under Developer Privileges, specify the developer's privileges:

- **User is a developer** - These users can create and edit applications as well as view developer activity, session state, workspace activity, application, and schema reports.
- **User is an administrator** - Workspace administrators additionally can create and edit user accounts, manage groups, alter passwords of users within the same workspace, and manage development services as described in "Managing a Development Workspace" on page 13-1.

7. Click **Create** or **Create and Create Another**.

To edit an existing user account:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Workspaces, select **Manage Application Developers**.

   The Manage Application Developers and Users page appears.
3. Locate a user as follows:
   - To locate a specific user, type a username or partial string in the Search field and click **Go**.
   - To view all users, leave the Search field blank and click **Go**
4. Click the **Edit** icon adjacent to appropriate username.
5. Follow the on-screen instructions.

Purging Inactive Workspaces

If you are managing a large hosted Oracle HTML DB instance, periodically purging inactive workspaces can free up resources for other users. The process of purging inactive workspaces consists of the following steps:

- Identify inactive workspaces
- Remove the resources associated with each inactive workspace
- Delete the inactive workspaces

Topics in this section include:

- Identifying Inactive Workspaces
- Removing the Resources Associated with Inactive Workspaces
- Deleting Inactive Workspaces

Identifying Inactive Workspaces

The first step in determining if a workspace is inactive is to establish some basic rules. A common approach is to base the rules on the Oracle HTML DB activity records found in the current HTML DB engine schema.

> **See Also:** "Determining the HTML DB Engine Schema" on page 18-2

The following DDL (data definition language) creates a table of all workspaces requested before June 28, 2004 but that have been inactive since June 10, 2004. In this example, inactivity is determined by checking a key within the HTML DB engine schema for the most recent updates by each workspace.

```
CREATE TABLE ws_to_purge AS
 SELECT c.security_group_id, c.company_name, c.admin_email, c.request_date,
 SYSDATE last_updated_on, 'Y' ok_to_delete
   FROM wwv_flow_provision_company c
  WHERE
c.request_date <= to_date('20040628','YYYYMMDD') AND
     (  not exists
 (SELECT NULL /* Activity Log */
        FROM wwv_flow_activity_log l
       WHERE l.security_group_id = c.security_group_id
         AND l.time_stamp > to_date('20040610','YYYYMMDD'))
 )
    AND NOT EXISTS
     (SELECT NULL /* workspace applications */
        FROM wwv_flows f
       WHERE f.security_group_id = c.security_group_id
         AND f.last_updated_on > to_date('20040610','YYYYMMDD'))
    AND NOT EXISTS
     (SELECT NULL /* Pages */
        FROM wwv_flow_steps s
       WHERE s.security_group_id = c.security_group_id
         AND s.last_updated_on > to_date('20040610','YYYYMMDD'))
    AND NOT EXISTS
     (SELECT NULL /* Regions */
        FROM wwv_flow_page_plugs p
       WHERE p.security_group_id = c.security_group_id
         AND p.last_updated_on > to_date('20040610','YYYYMMDD'))
    AND NOT EXISTS
     (SELECT NULL /* Items */
        FROM wwv_flow_step_items i
       WHERE i.security_group_id = c.security_group_id
         AND i.last_updated_on > to_date('20040610','YYYYMMDD'))
    AND NOT EXISTS
     (SELECT NULL /* Templates */
        FROM wwv_flow_templates t
       WHERE t.security_group_id = c.security_group_id
         AND t.last_updated_on > to_date('20040610','YYYYMMDD'))
    AND NOT EXISTS
     (SELECT NULL /* Files uploaded */
        FROM wwv_flow_file_objects$ o
       WHERE o.security_group_id = c.security_group_id
         AND o.created_on > to_date('20040610','YYYYMMDD'))
    AND NOT EXISTS
     (SELECT NULL /* SQL Workshop history */
        FROM wwv_flow_sw_sql_cmds s
       WHERE s.security_group_id = c.security_group_id
         AND s.created_on > to_date('20040610','YYYYMMDD'));
```

Once you identify inactive workspaces, you can purge them. Purging inactive workspaces is a two step process:

- First, remove the resources (that is, the database schemas, tablespaces, and data files) associated with each inactive workspace
- Second, drop the inactive workspaces from Oracle HTML DB

Removing the Resources Associated with Inactive Workspaces

Once you have identified inactive workspaces in a single table, the next step is to remove them.

> **Note:** Before removing the schemas, tablespaces, or data files associated with inactive workspaces, make sure these resources are not being used in by any other workspace or application

To remove the resources associated with inactive workspaces:

1. Identify the schemas used by the workspaces to be deleted by joining the table containing the identified inactive workspaces to `wwv_flow_company_schemas.`
2. Drop the schemas, tablespaces, and data files used exclusively by the inactive workspaces from the database by running a query similar to the following.

```
SELECT s.schema
  FROM ws_to_purge ws,
       wwv_flow_company_schemas s
WHERE s.security_group_id = ws.security_group_id
   AND ws.ok_to_delete = 'Y';
```

Deleting Inactive Workspaces

Once you remove the resources associated with an inactive workspace, you can delete it. You can delete inactive workspaces manually using the Oracle HTML DB Administration Services application. Or, you can delete them programmatically as shown in the following example.

```
BEGIN
     FOR c1 IN (SELECT security_group_id
                FROM ws_to_purge
                WHERE ok_to_delete = 'Y')
     LOOP
         WWV_FLOW_PROVISIONING.TERMINATE_SERVICE_BY_SGID(c1.security_group_id);
     END LOOP;
 END;
```

Removing a Workspace

Removing a workspace does not remove any of the associated database objects. To remove the associated schemas, a database administrator (DBA) must use a standard database administration tool such as Oracle Enterprise Manager or SQL*Plus.

Sees Also:

- *Oracle Enterprise Manager Administrator's Guide*
- *SQL*Plus User's Guide and Reference*
- "Viewing Workspace Reports" on page 18-6
- "Creating a Workspace" on page 18-4

To remove a workspace:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Workspaces, select **Remove Workspace**.
3. Select a workspace name and click **Next**.

4. Follow the on-screen instructions.

Exporting and Importing a Workspace

To move a workspace and all associated users to a new Oracle HTML DB instance, you must export the workspace. When you export a workspace, Oracle HTML DB generates a text file. This file contains information about your workspace, all the users in your workspace, and any groups in your workspace (if applicable). You can use this file to import your workspace into another Oracle HTML DB instance.

Keep in mind, this method only imports workspace, users, and groups. This file does not contain:

- The schemas associated with this workspace, or the objects in those schemas.
- Any applications, images, cascading style sheets and static text files.

All of these items must be exported separately.

See Also:

- "Deploying an Application to Another Oracle HTML DB Instance" on page 12-4
- "About Managing Database Objects" on page 12-4
- "Using Custom Cascading Style Sheets" on page 9-39

To export a workspace:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Workspaces, select **Export Workspace**.
3. Select a workspace name and click **Export**.
4. To export the selected workspace, click **Save File**.
5. Follow the on-screen instructions.

To import a workspace:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Workspaces, select **Import Workspace**.
3. Select a workspace name and click **Next**.
4. To install the workspace, click **Install**.
5. Follow the on-screen instructions.

Managing Logs

Oracle HTML DB administrators can manage the following log files on the Manage Logs and Files page:

- SQL Workshop logs
- Page View Activity logs
- Developer activity logs
- External click counting log

Topics in this section include:

- Deleting SQL Workshop Logs
- Deleting Page View Activity Log Entries
- Deleting Developer Activity Log Entries
- Deleting Click Counting Log Entries
- Deleting the HTML DB Mail Log Entries

Deleting SQL Workshop Logs

The SQL Workshop logs maintain a history of recent commands and scripts run in the SQL Command Processor.

To delete log files entries:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Logs**.

   The Manage Logs page appears.
3. Select **Review SQL Workshop logs**.
4. Select one of the following:
   - Script File executions log entries
   - Control File execution log entries
   - SQL Command Processor history log entries
   - SQL Archives entries
5. To delete entries by age:
   - Specify the age of the entries to deleted.
   - Click **Delete Entries**.
6. To delete all entries, click **Truncate Log**.

See Also: "Accessing the SQL Command History" on page 5-9

Deleting Page View Activity Log Entries

Page view activity logs track user activity for an application. Developers enable logging within their application on the Edit Application Attributes page.

The HTML DB engine actually uses two logs to track user activity. At any given time, one log is designated as current. For each rendered page view, the HTML DB engine inserts one row into the log file. A log switch occurs at the interval listed on the Manage Activity Logs page. At that point, the HTML DB engine removes all entries in the noncurrent log and designates it as current.

To truncate the activity logs manually:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Logs**.

   The Manage Logs page appears.

3. Select **Review page view activity log, with option to truncate**.
4. Click **Truncate Logs**.
5. Click either **Truncate Log 1** or **Truncate Log 2**.

See Also:

- "Application Definition" on page 7-5 for information on enabling logging
- "Viewing Workspace and User Activity Reports" on page 13-3 for information on page view activity logs

Deleting Developer Activity Log Entries

The Developer Activity Log tracks changes to applications within an individual workspace. Log entries older than one month are automatically deleted.

To delete Developer Activity Log entries:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Logs**.

   The Manage Logs page appears.
3. Select **Developer activity logs, review with option to delete entries**.
4. On the Developer Activity Logs page, click **Manage**.
5. Specify the age of the entries to be deleted and click **Delete Entries**.

See Also: "Viewing Application Changes by Developer and Day" on page 13-3 for more information on the Developer Activity Log

Deleting Click Counting Log Entries

The External Clicks Log counts clicks from an Oracle HTML DB application to an external site. You can implement this functionality using `COUNT_CLICK` procedure.

See Also: "COUNT_CLICK Procedure" on page 17-4

To delete click counting log entries:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Logs**.

   The Manage Logs page appears.
3. Select **External click counting log, review with option to truncate**.
4. On the Click Counting Log page, click **Manage**.
5. Specify the age of the entries to be deleted and click **Delete Entries**.

Deleting the HTML DB Mail Log Entries

The HTML DB Mail Log records message header information and send date of successfully sent mail message.

See Also: "Managing E-mail" on page 18-21

To truncate the mail log:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Logs**.

   The Manage Logs page appears.
3. Select **Manage mail log**.
4. On the Manage Mail Log page, click **Truncate Log**.

Managing Session State

A session is a logical construct that is used to establish persistence (or stateful behavior) across page views. Each session is assigned a unique ID which the HTML DB engine uses to store and retrieve an application's working set of data (or session state) before and after each page view. An automatic process clears sessions older than 24 hours every eight hours. As an Oracle HTML DB administrator, you can also purge them manually.

An Oracle HTML DB administrator can view session state statistics and purge session state on the Session State Management page.

Topics in this section include:

- Purging Sessions by Age
- Viewing Session Details Before Purging
- Viewing Session Statistics Before Purging

> **See Also:** "Understanding Session State Management" on page 6-8

Purging Sessions by Age

Using the Purge Session page, administrators can purge sessions by age.

To view specific session details:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Select the **Manage Session State**.
3. Select **Purge old sessions by age**.
4. On the Purge Session page, specify:
   - The maximum number of sessions to be purged
   - The age of session to be purged
5. To view a report of session statistics, click **Count Sessions**.
6. To purge the selected sessions, click **Purge Sessions**.

Viewing Session Details Before Purging

Before purging sessions, administrators can use the Recent Sessions page to first view a listing of recent sessions and then drill down on session details.

To purge sessions by age:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Select the **Manage Session State**.
3. Select **Report recent sessions with drill down to session details**.
4. On the Recent Sessions page, you can:
   - Click a session ID to view additional details.
   - Click Purge Session to delete the displayed sessions.

Viewing Session Statistics Before Purging

On the Session State Statistics page, administrators can view statistics about current sessions prior to purging.

To view session state statistics:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Select the **Manage Session State**.
3. Select **Report session counts**.
4. Click **Purge Sessions** to delete the current sessions.

Monitoring Activities

Oracle HTML DB administrators can monitor user activity by accessing a number of charts and reports on the Monitoring page.

To monitor user activity:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Monitor Activity, select a chart or page to review.

Managing Environment Preferences

HTML DB Environment Preferences are named substitution value pairs defined by an Oracle HTML DB administrator. Oracle HTML DB uses these preferences internally to determine a provisioning mode, configure the HTML DB engine to send mail, and restrict user access by IP address.

Topics in this section include:

Accessing the HTML DB Environment Preferences Page

To access Oracle HTML DB environment preferences:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Environment Preferences**.

   The HTML DB Environment Preferences page appears.

About SERVICE_REQUEST_FLOW

`SERVICE_REQUEST_FLOW` is a preference managed by Oracle HTML DB that determines the provisioning status of an Oracle HTML DB instance. Do not edit this preference.

See Also: "Specifying a Provisioning Mode" on page 18-5

Configuring Oracle HTML DB to Send Mail

If you enable request provision mode or enable users to reset their passwords using a link on the login page, you must configure Oracle HTML DB to send mail. In order to enable Oracle HTML DB to send mail, you must configure a number of settings on the Environment Preferences page.

To configure Oracle HTML DB to send mail:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Environment Preferences**.

   The HTML DB Environment Preferences page appears. Configure the following Run settings:

   - `DEVELOPMENT_SERVICE_URL` - If you are running in request provisioning mode, the value of this setting is used in the e-mail when the request is approved. This setting defines the URL for the service. If this setting is not present, the URL will be derived from your environment.
   - `SMTP_HOST_ADDRESS` - Defines the server address of the SMTP server. On installation, this will be set to `localhost`. If you are using another server for SMTP relaying, change `localhost` to that server's address.
   - `SMTP_HOST_PORT` - Defines the port the SMTP server listens to for mail requests. By default, this setting will be set to 25 at the time of installation.
   - `SMTP_FROM` - Defines the "from" address when an administrative tasks such as approving a provision request, or resetting a password generates an e-mail.

Restricting User Access by IP Address

Oracle HTML DB administrators can restrict user access to an Oracle HTML DB instance by creating a Runtime setting named `RESTRICT_IP_RANGE`.

To restrict user access by IP address:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Environment Preferences**.

   The HTML DB Environment Preferences page appears.
3. Click **Create**.
4. On the Runtime Settings page:

a. In Name, enter RESTRICT_IP_RANGE.

b. In Value, enter an IP address. Use an asterisk (*) to specify a wildcard.

   – Enter an IP address from one to four levels. For example:

   ```
   141, 141.* ...
   192.128.23.1 ...
   ```

   – Enter a comma delimited list of IP addresses.

Note: When using wildcards, do not include additional numeric values after wildcard characters. For example, `138.*.41.2`.

Disabling Access to Oracle HTML DB Administration Services

Oracle HTML DB administrators can restrict user access to Oracle HTML DB Administration Services by creating a Runtime setting named `DISABLE_ADMIN_LOGIN`. Creating this setting in production environments prevents unauthorized users from logging in to Oracle HTML DB Administration Services and possibly compromising user login credentials.

To disable user access to Oracle HTML DB Administration Services:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Environment Preferences**.

   The HTML DB Environment Preferences page appears.
3. Click **Create**.
4. On the Runtime Settings page:

   a. In Name, enter `DISABLE_ADMIN_LOGIN`

   b. In Value, enter `YES` in upper case letters

Setting this value and logging out, prevents anyone for logging in to Oracle HTML DB Administration Services.

To reverse this setting, connect in SQL*Plus as the HTML DB engine schema and execute the following:

```
UPDATE WWV_FLOW_PLATFORM_PREFS set value = 'NO' WHERE name = 'DISABLE_ADMIN_LOGIN'
   /
commit
   /
```

Disabling Access to Oracle HTML DB Internal Applications

Oracle HTML DB administrators can restrict user access to Oracle HTML DB Internal applications by creating a Runtime setting named `DISABLE_WORKSPACE_LOGIN`. Creating this setting in production environments prevents unauthorized users from running applications in the Internal workspace (that is, Application Builder, SQL Workshop, and Data Workshop) and possibly compromising login credentials. Administrators who use this feature should also consider disabling user access to Oracle HTML DB Administration Services.

See Also: "Disabling Access to Oracle HTML DB Administration Services" on page 18-19

To disable user access to the Internal workspace:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Environment Preferences**.

   The HTML DB Environment Preferences page appears.
3. Click **Create**.
4. On the Runtime Settings page:
   a. In Name, enter `DISABLE_WORKSPACE_LOGIN`
   b. In Value, enter `YES` in upper case letters

Setting this value and logging out, prevents anyone for logging in to the Internal workspace.

To reverse this restriction, log in to Oracle HTML DB Administration Services and remove the `DISABLE_WORKSPACE_LOGIN` Runtime, or change the Value to `NO`.

Managing Application Build Status

Every Oracle HTML DB application has an application level attribute called Build Status. You can use this attribute to prevent an application from being modified by other developers. Build Status has two settings:

- **Run and Build Application** - Developers can both run and edit an application
- **Run Application Only** - Developers can only run an application

Setting the Build Status to **Run Application Only** is an effective way to prevent other developers from modifying it. You can change the Build Status by:

- Changing the Build Status attribute on the Edit Application Attributes page. (See "Application Availability" on page 7-8.)
- Changing the Build Status during the deployment process. (See "Deploying an Application to Another Oracle HTML DB Instance" on page 12-4.)

Deploying an application from one Oracle HTML DB instance to another is a three step process:

1. Export the application and all related files from the development Oracle HTML DB instance
2. Import the exported files into the target Oracle HTML DB instance
3. Install the exported files from Export Repository

During steps 1 and 2, you have the option of setting the Build Status to **Run Application Only**. Be aware that if you set the Build Status to **Run Application Only** during deployment, you can only change it in Oracle HTML DB Administration Services.

To change a Build Status set during deployment:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage Applications, select **Manage Build Status**.
3. Locate an application by making selections from the Build Status, Workspace, and Application lists and clicking **Go**.

4. Click the **Edit** icon adjacent to the appropriate application.

   The Edit Build Status page appears.

5. Select the alternate build status and click **Apply Changes**.

Managing E-mail

Oracle HTML DB administrators can manage e-mail sent from an application by accessing the HTML DB Mail Queue and HTML DB Mail Log.

See Also: "Sending E-mail from an Application" on page 15-1

Topics in this section include:

- Viewing the Mail Queue
- Viewing the HTML DB Mail Log

Viewing the Mail Queue

Oracle HTML DB administrators can use the Manage Mail Queue page to monitor e-mail messages in the mail queue.

To monitor messages in the mail queue:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Mail Queue**.

   The Mail Queue page appears.

3. To send e-mail messages, select the messages to be send and click **Send All Mail**.
4. To delete e-mail messages, select the messages to be deleted and click **Delete**.

Viewing the HTML DB Mail Log

The HTML DB Mail Log records message header information and send date of successfully sent mail message.

To view the mail log:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Logs**.

   The Manage Logs page appears.

3. Select **Manage mail log**.

   Tip: To delete mail log entries, click **Truncate Log**.

Creating a Site-Specific Tasks List

You can create a list of links called Site-Specific Tasks on the Workspace Home page and Workspace Login page. To include these task lists in an Oracle HTML DB instance, an Oracle HTML DB administrator must create the tasks on the Site-Specific Tasks page.

Topics in this section include:

Adding a New Task

To add new task to a Site-Specific Tasks lists:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Site-Specific Task Lists**.

   The Site-Specific Tasks page appears.
3. To create a new link, click **Create**.
4. On the Site-Specific Tasks page you can specify the following:
   a. In Display Sequence, indicate the relative order of this task within the list.
   b. In Display Location, indicate the page on which the task should display (that is, the Workspace Login page or Workspace Home page).
   c. In Task Name, enter a name for this task.
   d. In Tasks Link, enter the link target for this task using `f?p` syntax.

      See Also: "Using f?p Syntax to Link Pages" on page 6-14

Editing an Existing Task

To edit an existing task:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Site-Specific Task Lists**.

   The Site-Specific Tasks page appears.
3. Click the **Edit** icon adjacent to the appropriate link.
4. On the Site-Specific Tasks page, edit the appropriate attributes.
5. Click **Apply Changes**.

Deleting a Task

To delete an existing task:

1. Log in to Oracle HTML DB Administration Services. (See "Logging in to Oracle HTML DB Administration Services" on page 18-2.)
2. Under Manage HTML DB Service, select **Manage Site-Specific Task Lists**.

   The Site-Specific Tasks page appears.
3. Click the **Edit** icon adjacent to the appropriate link.
4. Click **Delete**.

A

Available Conditions

A condition is a small unit of logic that helps you control the display of regions, items, buttons, and tabs as well execute processes, computations and validations. When you apply a condition to a control or component, the condition is evaluated. Whether a condition passes or fails determines whether a control or component displays, or page processing executes.

You can specify conditions by selecting a condition type when you create the control (region, button, or item) or component (tab, list, or navigation bar), or by making a selection under the conditional display attribute.

See Also: "Understanding Conditional Rendering and Processing" on page 6-7

Conditions Available in Oracle HTML DB

The following table describes some commonly used conditions. To view a complete listing of all available conditions for a given control or component, click the View icon to the right of the Conditional Display Type list. Shortcuts to common selections appear directly beneath the Type list. If your condition requires an expression, type it in the appropriate field.

Table A–1 describes the conditions available in Oracle HTML DB.

Table A–1 Available Conditions

Condition	Description
Always	Always returns true. Used primarily for the read-only conditions of a page item
Current Language != Expression 1	Verifies the language setting in which the client browser is not currently running. Evaluates to true if the current language is contained within the string entered in Expression 1.
Current Language = Expression	Verifies the language setting in which the client browser is currently running. Evaluates to true if the current language matches the value entered in Expression 1.
Current Language is contained within Expression 1	Determines whether the browser current language is contained within a string. Evaluates to true if the current language matches the string entered in Expression 1. For example, to check if the current language is either en-US or en-GB, choose this condition and enter the following string in Expression 1: `en-us,en-gb`
Current Language is not contained within Expression 1	Verifies the application's current language is not contained within a specified string. Evaluates to true if the current language is not contained within the string entered in Expression 1.
Current page != Expression 1	Evaluates to true if the current page does not equal the page you specify in Expression 1.

Table A–1 (Cont.) Available Conditions

Condition	Description
Current Page != Page Submitted (this page was not the page posted)	Determines if the specified page was not posted. Evaluates to true if the current page does not match the value entered in Expression 1.
Current page = Expression 1	Evaluates to true if the current page equals the page you specify in Expression 1.
Current Page = Page Submitted (this page was posted)	Verifies the whether the specified page was posted. Evaluates to true if the current page matches the value entered in Expression 1.
Current Page is contained within Expression 1 (comma delimited list of pages)	Verifies if the current page is part of the list of pages you specify in Expression 1. To check if the current page is in either page 1, 2, 3 or 4, select this condition type and enter the following string in Expression 1: `1,2,3,4`
Current page is in Printer Friendly mode	Only displays certain page control or components when the user has selected printer friendly mode. If the current page is in printer friendly mode, then the condition evaluates to true. Use `f?p` syntax to specify printer friendly mode.
Current page is not in Printer Friendly mode	Hides page controls or components when printer friendly mode is selected. Use `f?p` syntax to specify printer friendly mode. **See Also:** "Using f?p Syntax to Link Pages" on page 6-14 for more information on `f?p` syntax
Current Page not in Expression 1 (comma delimited list of pages)	Verifies if the current page is not part of the comma separated list of pages specified in Expression 1.
Exists (SQL query returns at least one row)	This condition is expressed as a SQL query. If the query returns at least one row then the condition evaluates as true. For example: `select 1 from emp where deptno = :P101_DEPTNO` This example references item `P101_DEPTNO` as a bind variable. You can use bind variables a within an application processes and SQL query regions to reference item session state. If one or more employees are in the department identified by the value of `P101_DEPTNO` then the condition evaluates as true. **See Also:** "About Bind Variables" on page 6-13 for more information on bind variables
Never	This condition type is hard wired to always fail. It is useful in temporarily preventing controls or components (such as regions, buttons, or items) from being rendered on a page, or to prevent processes, computations and validations from running.
NOT Exists (SQL query returns no rows)	This condition is expressed as a SQL query. If the query does not return any rows, it evaluates as true.
PLSQL Expression	A PL/SQL expression is any expression in valid PL/SQL syntax that evaluates to true or false. For example: `nvl(:MY_FLOW_ITEM,'NO') = 'YES'` If the value of `MY_FLOW_ITEM` is YES then the condition evaluates as true. Otherwise it evaluates as false.

PLSQL Function Body returning a boolean	The body of a PL/SQL function that returns true or false. For example:
	`BEGIN`
	`IF :P1_DAY = 'MONDAY' THEN`
	`  RETURN TRUE;`
	`ELSE`
	`  RETURN FALSE;`
	`END;`

Table A–1 (Cont.) Available Conditions

Condition	Description
Request != Expression 1	`REQUEST` is an internal attribute that tracks of how a page is submitted. By default, when a page is submitted, the value of the application attribute `REQUEST` is set according the name of the object that caused the page to be submitted. For a regular button, `REQUEST` is set as the name of the button (such as CANCEL or SAVE) not the label of the button. You can also set request using `f?p` syntax. For example, the event could be when a user clicks a button or selects a tab menu. Depending upon the event, you can perform different operations by referencing the value of the `REQUEST` application attribute. This condition evaluates as true if `REQUEST` does not equal the value entered in Expression 1. **See Also:** "Understanding URL Syntax" on page 6-14, "REQUEST" on page 6-24, and "Understanding the Relationship Between Button Names and REQUEST" on page 8-34
Request = Expression 1	This condition is the opposite of `Request != Expression 1`. This condition evaluates as true if `REQUEST` equals the value entered in Expression 1. From PL/SQL you can also reference the application attribute using the following syntax: `V('REQUEST')` **See Also:** "Understanding URL Syntax" on page 6-14, "REQUEST" on page 6-24, and "Understanding the Relationship Between Button Names and REQUEST" on page 8-34
Request is contained within Expression 1	`REQUEST` is an internal application attribute that tracks of how a page is submitted. By default, when a page is submitted, the value of `REQUEST` is set according to the event that caused the page to be submitted. For example, the event could be when a user clicks a button or selects a tab. Depending upon the event, you can perform different operations by referencing the value of the `REQUEST` application attribute. Use this condition to specify a list of allowed requests (such as SAVE or UPDATE) in Expression 1. The condition evaluates to true if the value of `REQUEST` is contained in the list. **See Also:** "REQUEST" on page 6-24, and "Understanding the Relationship Between Button Names and REQUEST" on page 8-34
Request is not contained within Expression 1	This condition is the opposite of `Request is contained within Expression 1`. Evaluates to true if the value of the `REQUEST` is not contained within Expression 1. **See Also:** "REQUEST" on page 6-24, and "Understanding the Relationship Between Button Names and REQUEST" on page 8-34
SQL Expression	SQL Expressions are evaluated as a WHERE clause in a SQL statement. For example suppose your expression is `:MY_ITEM = 'ABC'`. The HTML DB engine processes the following: `select 1 from dual where :MY_ITEM = 'ABC'` This condition evaluates to true if a row is returned.
SQL Reports (OK to show the back button)	Use this condition for reports having pagination. It automatically determines when it is appropriate to include a button that pages back in the result set.
SQL Reports (OK to show the forward button)	Use this condition for reports having pagination. It automatically determines when it is appropriate to include a button that pages forward in the result set.
Text in Expression 1 != Expression 2 (includes &ITEM substitutions)	Use this expression to compare two expressions containing strings. Either expression may contain references to session state using `&MY_ITEM` syntax. **See Also:** "Using Substitution Strings" on page 6-16 for more information on `&MY_ITEM` syntax

Table A–1 (Cont.) Available Conditions

Condition	Description
Text in Expression 1 = Expression 2 (includes &ITEM substitutions)	This condition is the opposite of `Text in Expression 1 != Expression 2` (includes `&ITEM` substitutions). Compares two expressions containing strings. Either expression may contain references to session state using the `&ITEM.` syntax. To check if the item `F100_P2_DAY_DATE` equals "Wednesday", select this condition enter the following in Expression 1 and Expression 2: ■ Expression 1: `F100_P2_DAY_DATE` ■ Expression 2: `Wednesday` **See Also:** "Using Substitution Strings" on page 6-16 for more information on `&MY_ITEM` syntax
User is authenticated (not public)	Verifies whether the current user was authenticated using one of the built-in authentication schemes or a custom authentication scheme. See Also: "Providing Security Through Authorization" on page 14-9 for more information on authentication
User is the public user (user has not authenticated)	The public user is defined as an application attribute. To set the public user for a specific application, navigate to the Application Builder home page and click the edit link corresponding to your application. A public user is a user used for multiple users. Sometimes applications have pages that are public and thus require authentication and log in. This condition returns true if the user is the public user (that is, the user is authenticated as themselves or some other user not equal to the public user identified in the application attribute Public User. **See Also:** "Session Management" on page 7-7
Value of Item in Expression 1 != zero	Verifies if the value of the item in Expression 1 does not equal zero.
Value of item in Expression 1 = Expression 2	Compares the value of an item with a specific string. Comparisons using this condition are case-sensitive. For example, to verify whether the value of an item `F100_P2_WORD` is contained within the string "the quick brown fox", enter the following in the Expression 1 and Expression 2 fields: ■ Expression 1: `F100_P2_WORD` ■ Expression 2: `the quick brown fox`
Value of Item in Expression 1 = zero	Verifies if the value of the item in Expression 1 does equal zero.
Value of item in Expression 1 contains no spaces	Evaluate to true if the value of the item specified in Expression 1 contains no spaces.
Value of Item in Expression 1 is alphanumeric	Evaluates to true when the string in Expression 1 contains only alphanumeric characters.
Value of Item in Expression 1 is contained within colon delimited list in Expression 2	Use this condition type to check whether a certain string is contained within the value of a session state item. Verifies whether the string specified in Expression 1 is contained in the value of the item specified in Expression 2.
Value of Item in Expression 1 is NOT contained within colon delimited list in Expression 2	Evaluates to true when the value specified in Expression 1 contains a string that lists elements delimited by colons. To check if the item `P1_TODAY` is either "Monday", "Tuesday", or "Wednesday", select this condition and enter the following in Expression 1 and Expression 2: ■ Expression 1: `P1_TODAY` ■ Expression 2: `Monday: Tuesday:Wednesday`
Value of Item in Expression 1 is NOT NULL	In Expression 1, enter the name (upper case) of the application or page item. Evaluates as true, if the current cache value of the item is not null and has a value. If not, the condition evaluates as false.
Value of Item in Expression 1 is NULL	Evaluates as true if the item in Expression 1 has no value.
Value of Item in Expression 1 is NULL or zero	Evaluates as true if the item in Expression is either NULL or zero.
Value of item in Expression 1 is numeric	Evaluates to true if the value of the Item in Expression 1 is numeric.
Value of user preference in Expression 1 != Expression 2	This condition is the opposite of `Value of user preference in Expression 1 = Expression 2`. Evaluates to true if the name of the user preference specified in Expression 1 is not equal to the string in Expression 2.

Table A–1 (Cont.) Available Conditions

Condition	Description
Value of user preference in Expression 1 = Expression 2	Verifies the value of a user preferences. Evaluates to true if the name of the user preference specified in Expression 1 is equal to the string in Expression 2.
When any item in comma delimited list of items has changed	Evaluates to true when the value of any non NULL session state item in the list of items specified in Expression 1 has changed.
When any item in comma delimited list of pages has changed	Evaluates to true when the value of any non NULL session state item in the list of pages specified in Expression 1 has changed.
When any item in current application has changed	This condition passes when the value of any non NULL session state item in the current application has changed.
When any item in current page has changed	Evaluate to true when the value of any non NULL session state item in the current page has changed.
When any item in current session has changed	Evaluates to true when the value of any non NULL session state item in the current session has changed.
When cgi_env DAD_NAME != Expression 1	This condition is the opposite of `When cgi_env DAD_NAME = Expression 1`. Checks for the DAD (Database Access Descriptor) that is being used in the URL to call the current page in the application and compares it to Expression 1. Evaluate to true, when the DAD is not the same as Expression 1.
When cgi_env DAD_NAME = Expression 1	Checks for the DAD (Database Access Descriptor) that is being used in the URL to call the current page in the application and compares it to Expression 1. Evaluate to true, when the DAD is the same as Expression 1.
When cgi_env HTTP_HOST != Expression 1	This condition is the opposite of `When cgi_env HTTP_HOST = Expression 1`. Checks for the value of the CGI environment variable `HTTP_HOST` that is the value returned by `owa_util.get_cgi_env ('HTTP_HOST')`. Evaluate to true, when this value is not equal to the string in Expression 1.
When cgi_env HTTP_HOST = Expression 1	Checks for the value of the CGI environment variable `HTTP_HOST` that is the value returned by `owa_util.get_cgi_env ('HTTP_HOST')`. Evaluate to true, when this value is equal to the string in Expression 1.
When cgi_env SERVER_NAME != Expression 1	This condition is the opposite of `When cgi_env SERVER_NAME = Expression 1`. This condition checks for the value of the CGI environment variable `SERVER_NAME`, that is the value returned by `owa_util.get_cgi_env ('SERVER_NAME')`. Evaluate to true, when this value is not equal to the string in Expression 1.
When cgi_env SERVER_NAME = Expression 1	This condition checks for the value of the CGI environment variable `SERVER_NAME`, that is the value returned by `owa_util.get_cgi_env ('SERVER_NAME')`. Evaluate to true, when this value is equal to the string in Expression 1.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X