R12.x Extend Oracle Applications: Building OA Framework Applications

Student Guide – Volume II

D61636GC10
Edition 1.0
November 2010
D69574

ORACLE

Author

Lauren Cohn

Technical Contributors and Reviewers

Mike Waddoups, Phil Cannon

Curriculum Manager

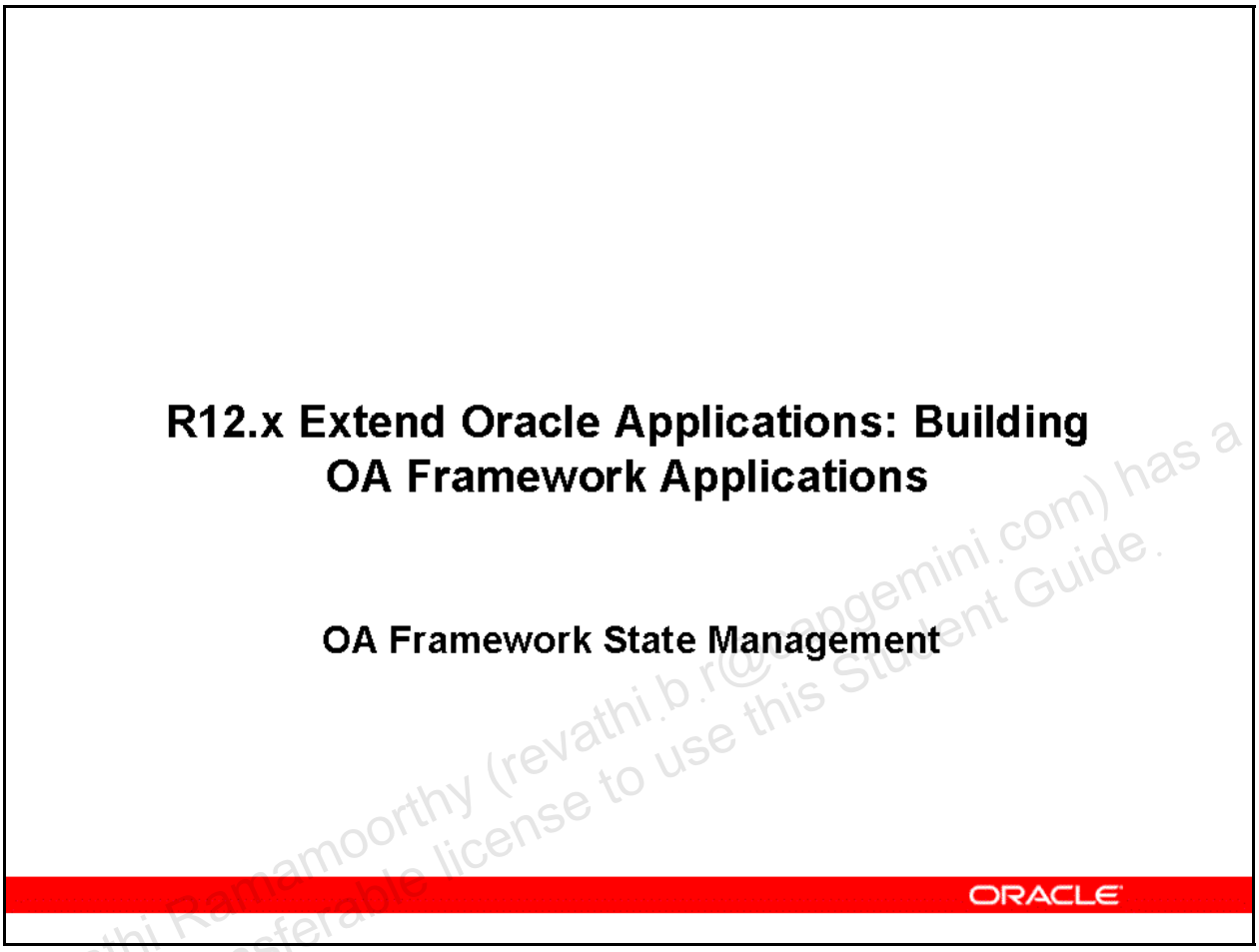
Bill Sawyer

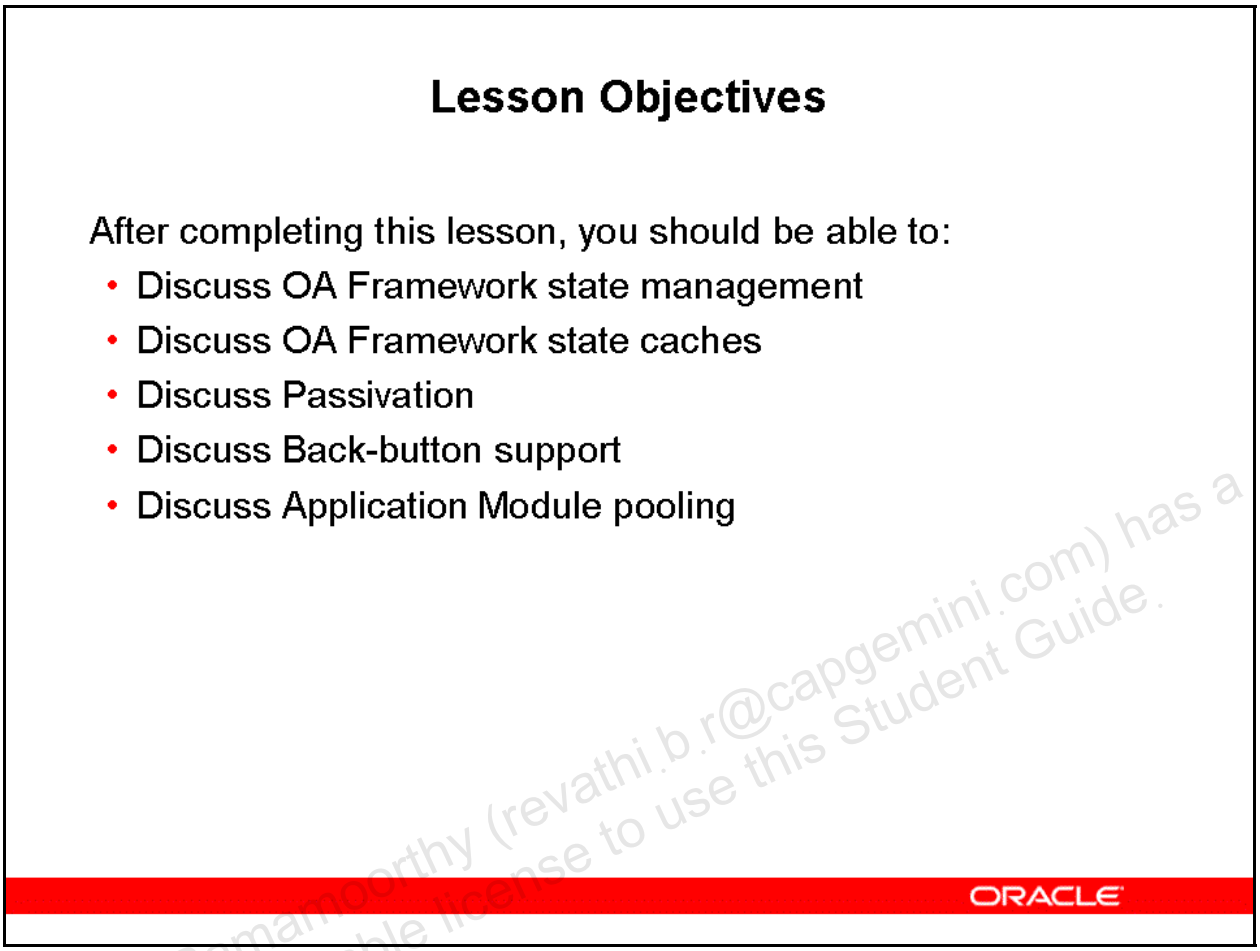
This book was published using: Oracle Tutor

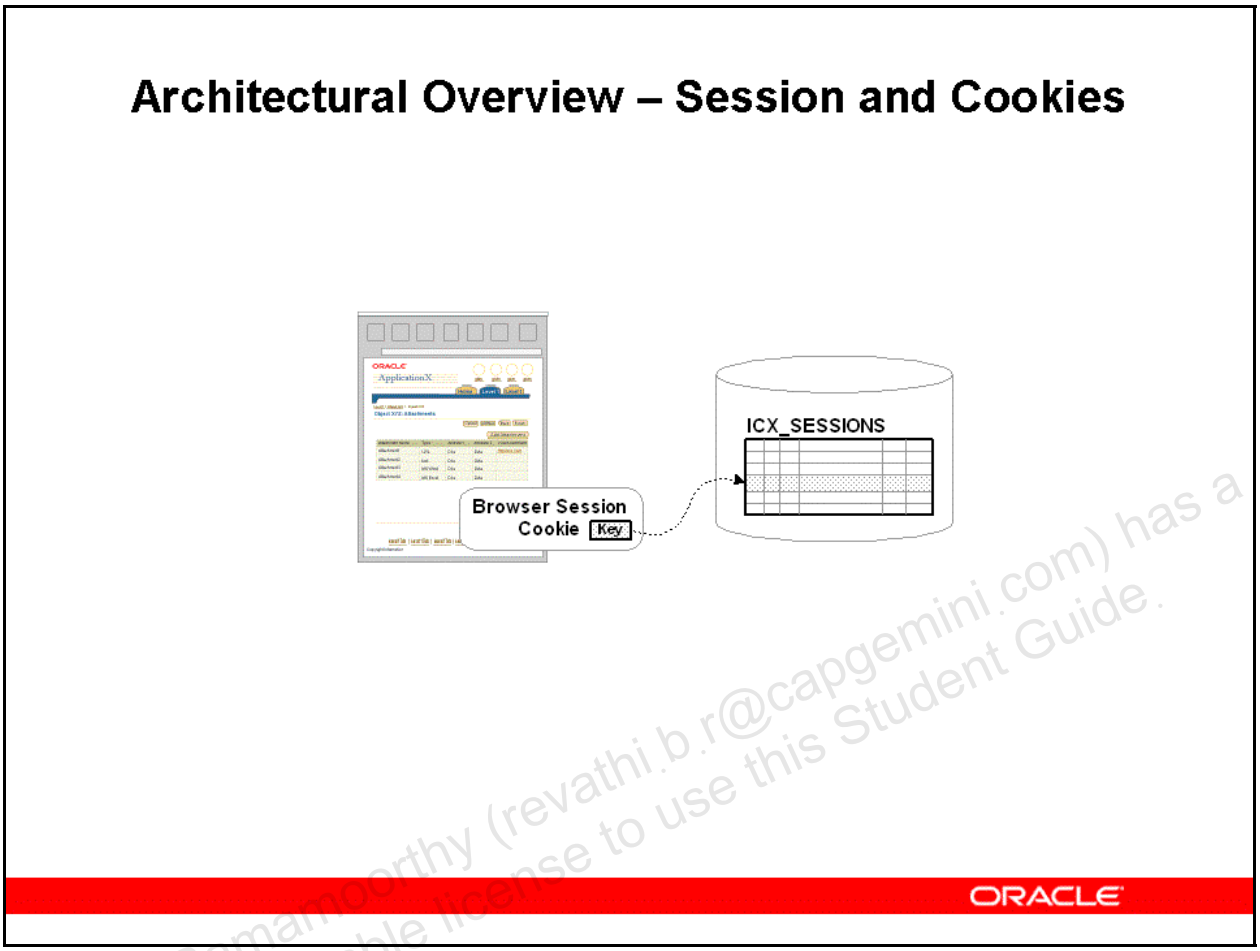
Table of Contents

OA Framework State Management

Chapter 7

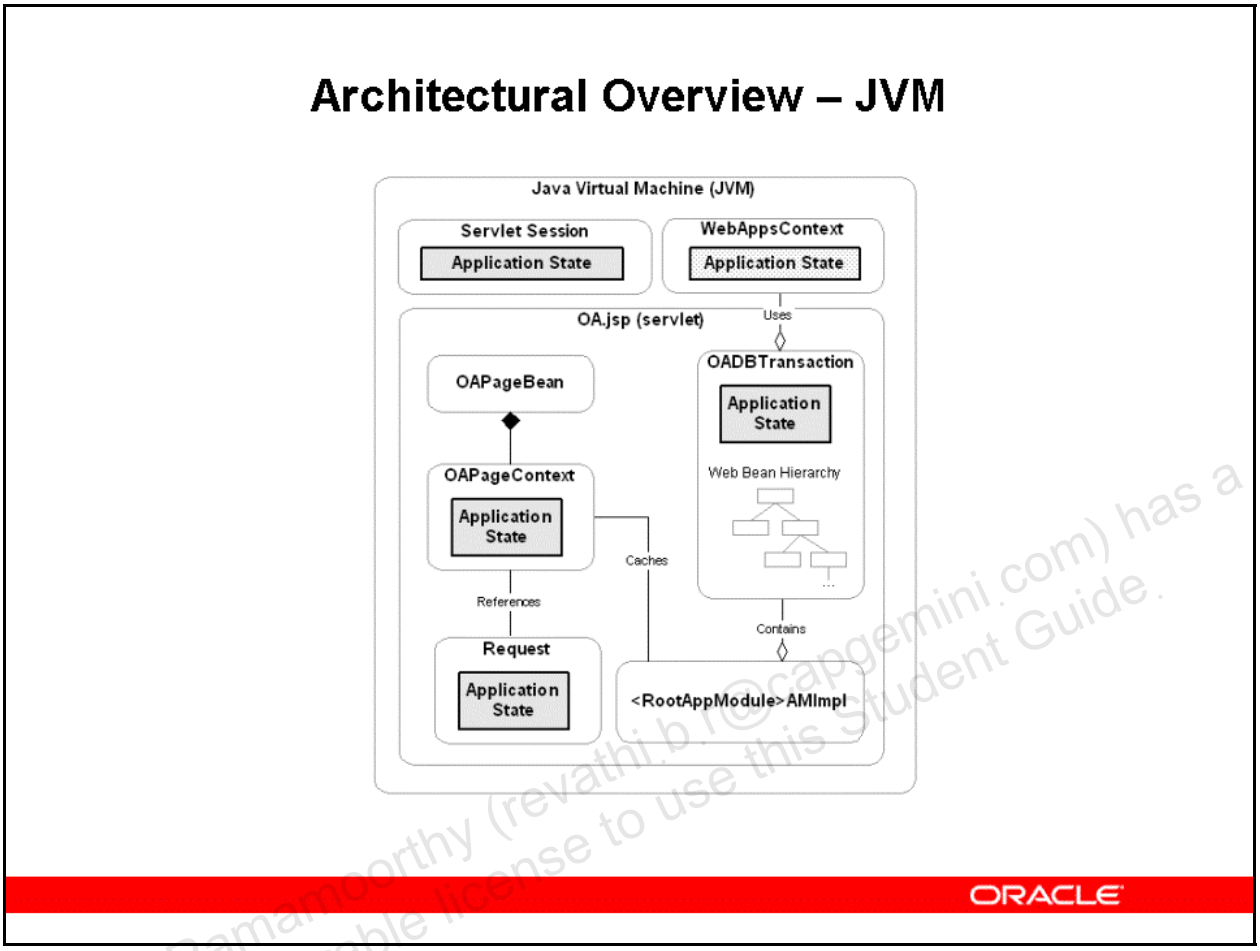




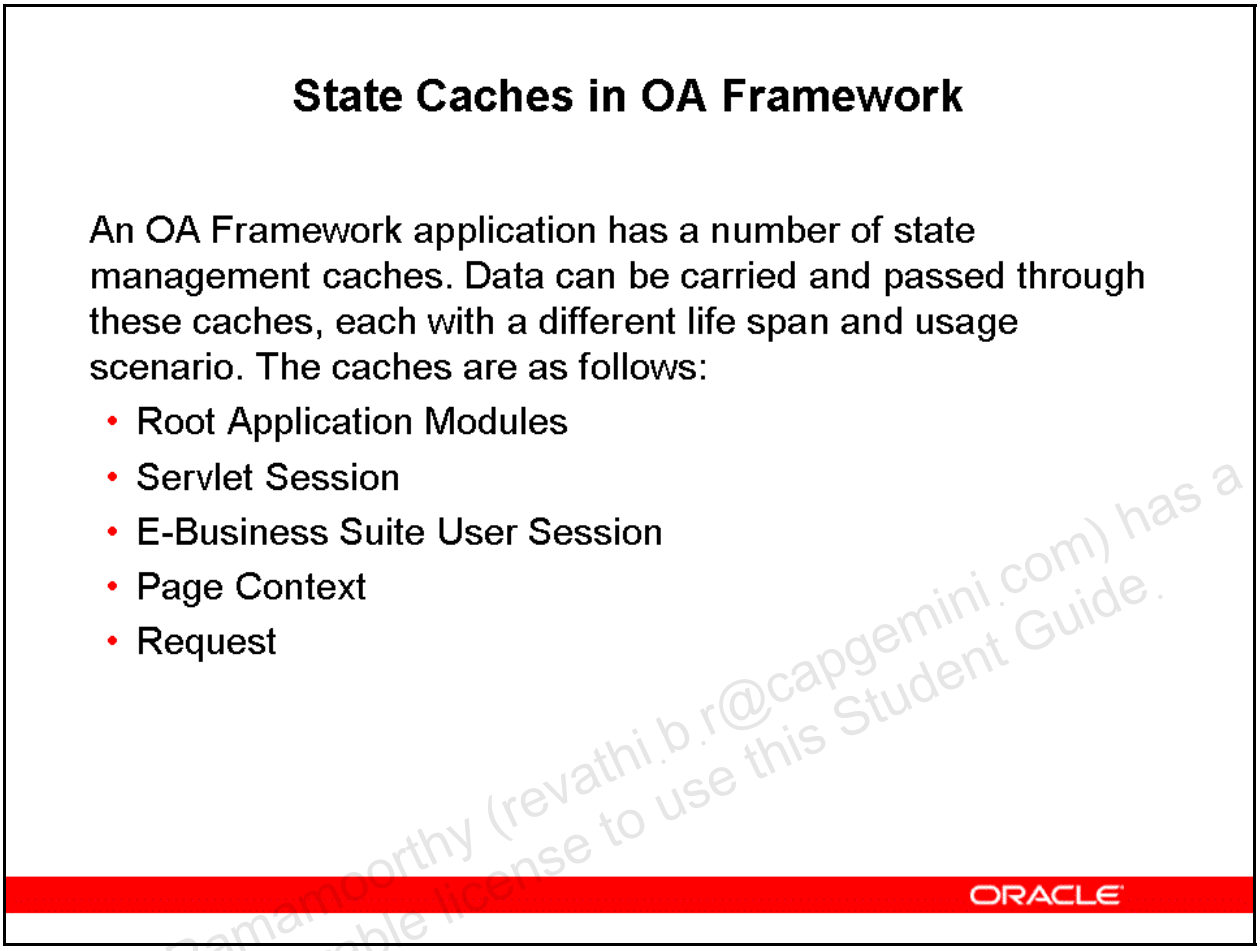


Everything starts with the browser. When the user enters the URL for the E-Business Suite instance, and they log in, a session is created in the database, and a session cookie is stored on the browser for that session. This is a security mechanism to prevent a cut-n-paste of URLs, and subsequent passage of those URLs to other users who may not have the proper security rights to that part of the application.

Architectural Overview – JVM

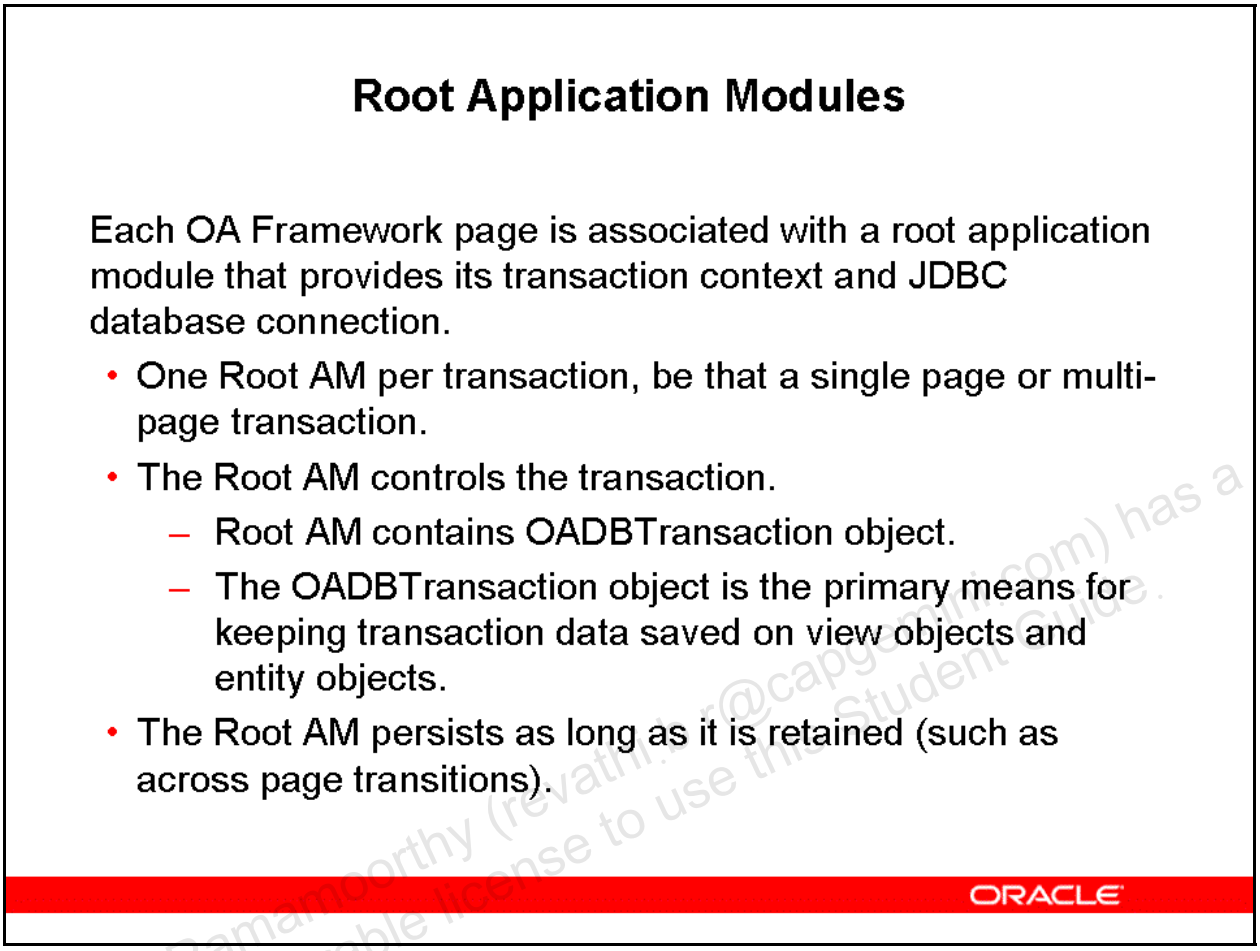


Everything in Java runs inside a Java Virtual Machine (JVM), and OA Framework pages are no exception to that rule.



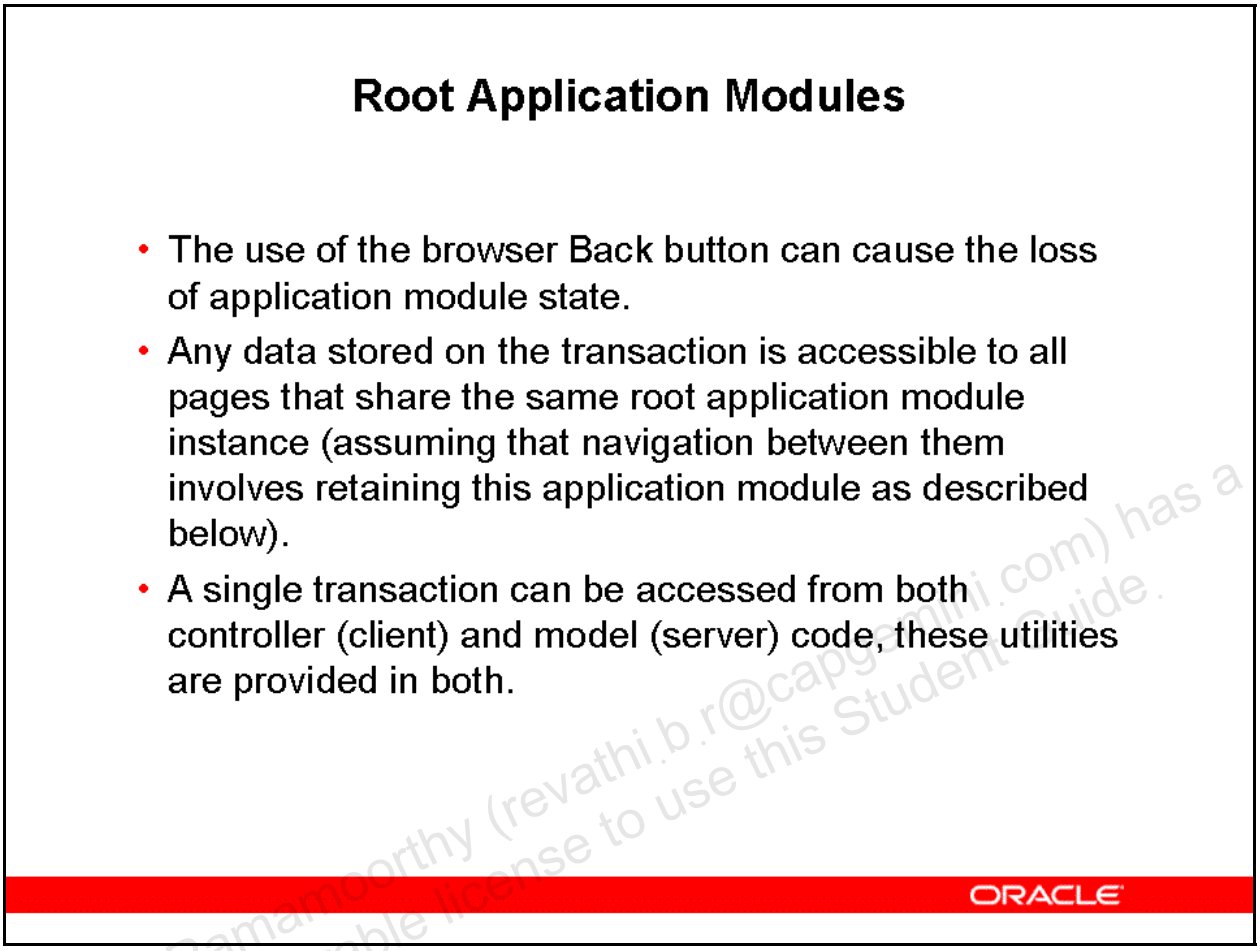
Additionally, as all pages are run from within OA.jsp, there is a JSP cache that is part of the caching mechanism. However, the JSP cache is outside the scope of an OA Framework page and is outside the scope of this course.

Root Application Modules



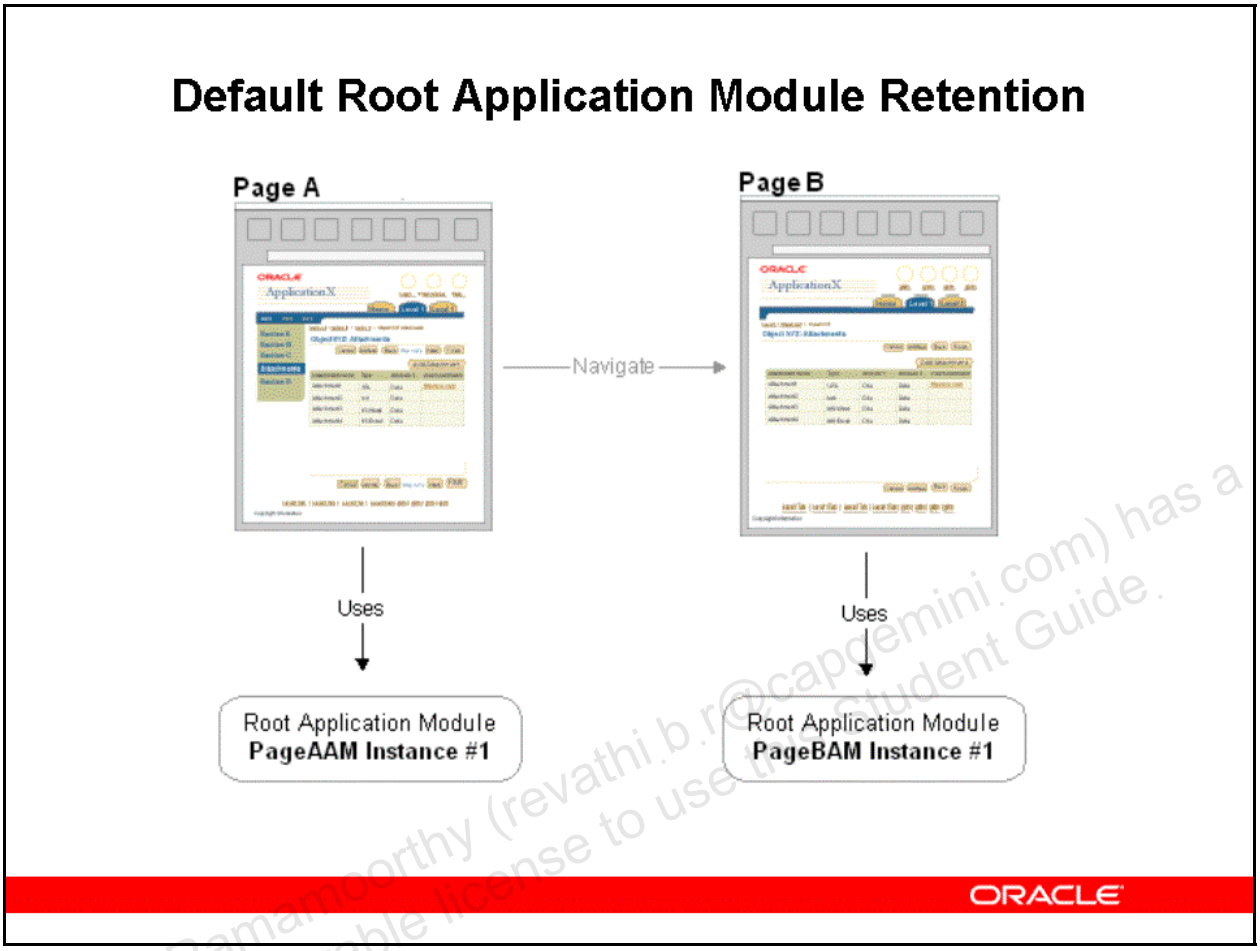
The Root AM is simply the first AM that is loaded. Initially, the Root AM is without connections or a transaction context. Once instantiated, the Root AM establishes the connections and creates the transaction context. The Root AM, shares the connection and context through the OA Framework transaction context for that user on that page (or set of pages).

Root Application Modules



Example: Imagine a typical multi-page ordering flow common to most e-Commerce sites. At the end of the transaction, the user is given a confirmation page and a summary page finalizing the transaction. The final page is the summary. In the transition from the confirmation page to the summary page, the user record is committed to the database as the transaction is final at that point. Whatever order/transaction the user has been performing is now finished. At the summary page, if the Back button were allowed, what would be the desired behavior? Should the transaction be uncommitted (rolled back)? The nature of web applications and the browser in which those applications run, makes it difficult to anticipate user behavior. The back button within the browser was originally created as a “Window History – 1” and was designed around static HTML pages. Transactional contexts were never built into Web Browsers and because of that, the back button’s use is sometimes problematic.

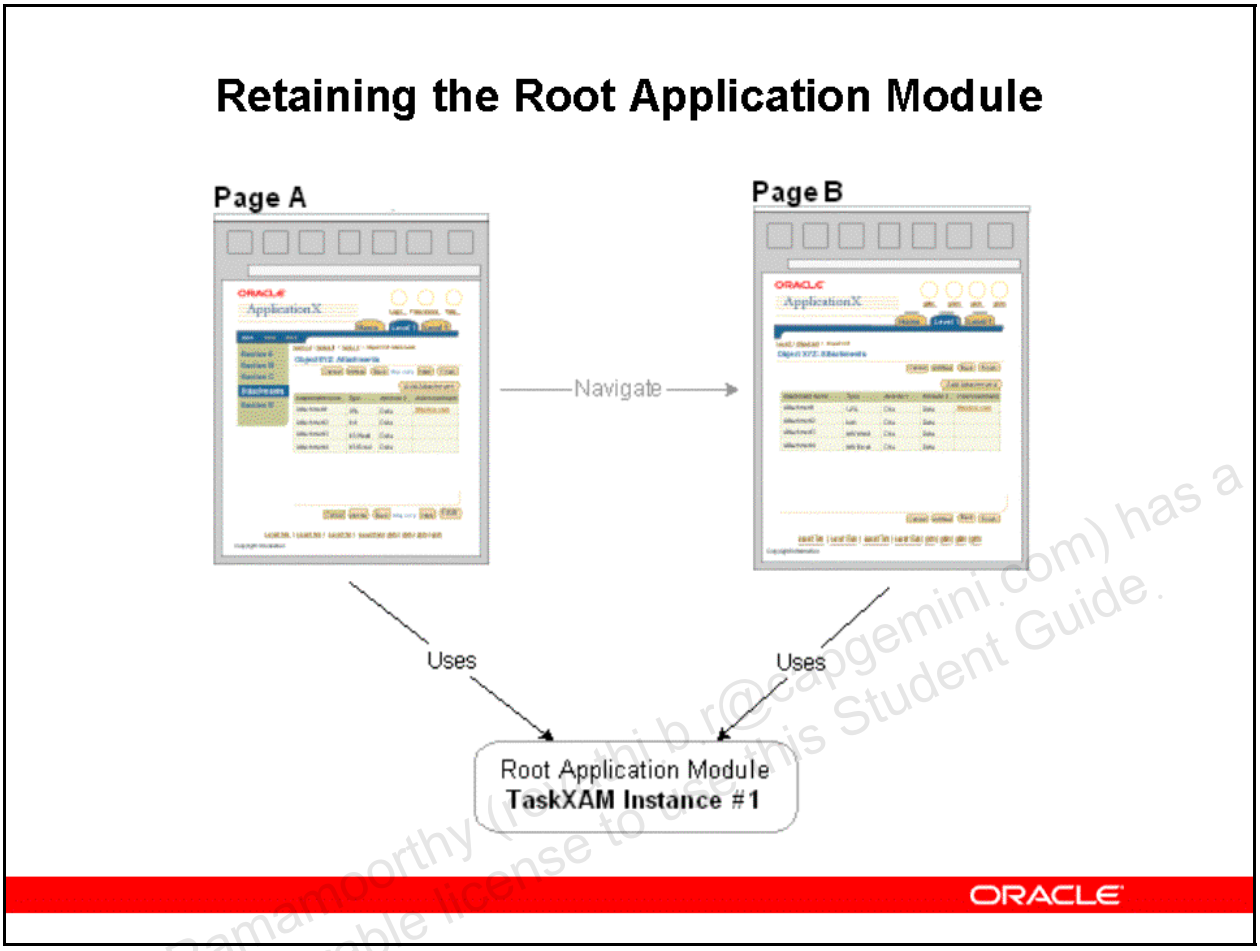
Default Root Application Module Retention



By default, when the user navigates from one page to the next (such as with a GET request or a JSP forward), and OA Framework renders the new page, the application module instance associated with the previous page is "released," and a new instance is requested from an application module pool.

Note: OA Framework never releases application modules during form submit (POST) requests unless you explicitly release the application module in a controller. For example, if a user sorts a table or navigates the result set within a table -- two actions that implicitly submit the page form -- the page's root application module instance is automatically retained.

Retaining the Root Application Module



The default behavior as described above is desirable for individual pages that comprise an isolated, complete task. However, it is not appropriate for a multi-page flow that implements a single task, or a series of related pages participating in a virtual transaction. In these cases, the different pages should be associated with the same root application module instance.

To achieve this, you must do the following:

Declaratively associate the same root application module type with each page in the multi-page flow.

Set the application module retention flag for a page by specifying the URL parameter retainAM=Y. For GET requests. This flag is evaluated when a new page is rendered (as mentioned above, OA Framework always retains the application module for POST requests regardless of the retainAM parameter value). If set to "Y," the previous page's application module instance will be retained. If set to "N" (or not specified, which implies "N"), OA Framework releases all application modules -- including any that might have been explicitly retained before reaching this point. You also set this parameter when calling JSP forward OAPageContext methods

Warning: It is not enough to simply associate the same root application module with each page. If you forget to set the retainAM flag, each page will have a different

application module instance -- and transaction -- even though they are associated with the same application module type.

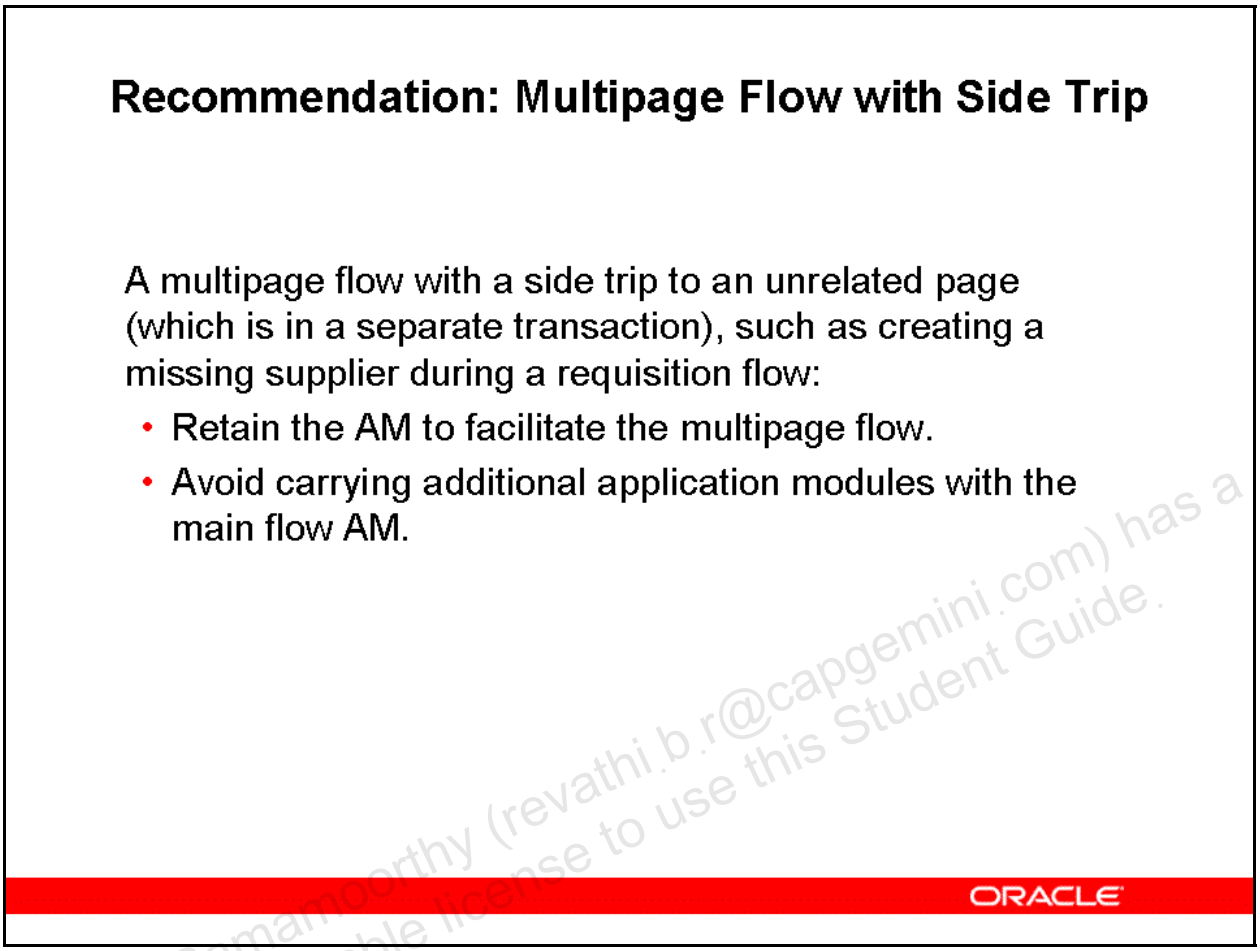
Note: Technically, depending on the state of the application module pool, Page B could get a reference to the same physical application module instance that Page A used. However, the object's state will be completely reset as if created anew. For the purposes of this discussion, consider it a "new instance."

Recommendation: Multipage Flow

A flow of related pages (query, update, insert, delete pages in the same virtual transaction):

- Retain the AM
  - Everything hanging off the same menu item should usually use and retain the same AM (includes navigation through trains, links, and so on, but not another tab). These would be considered the same virtual transaction.
  - Be careful to test for and handle back button problems.

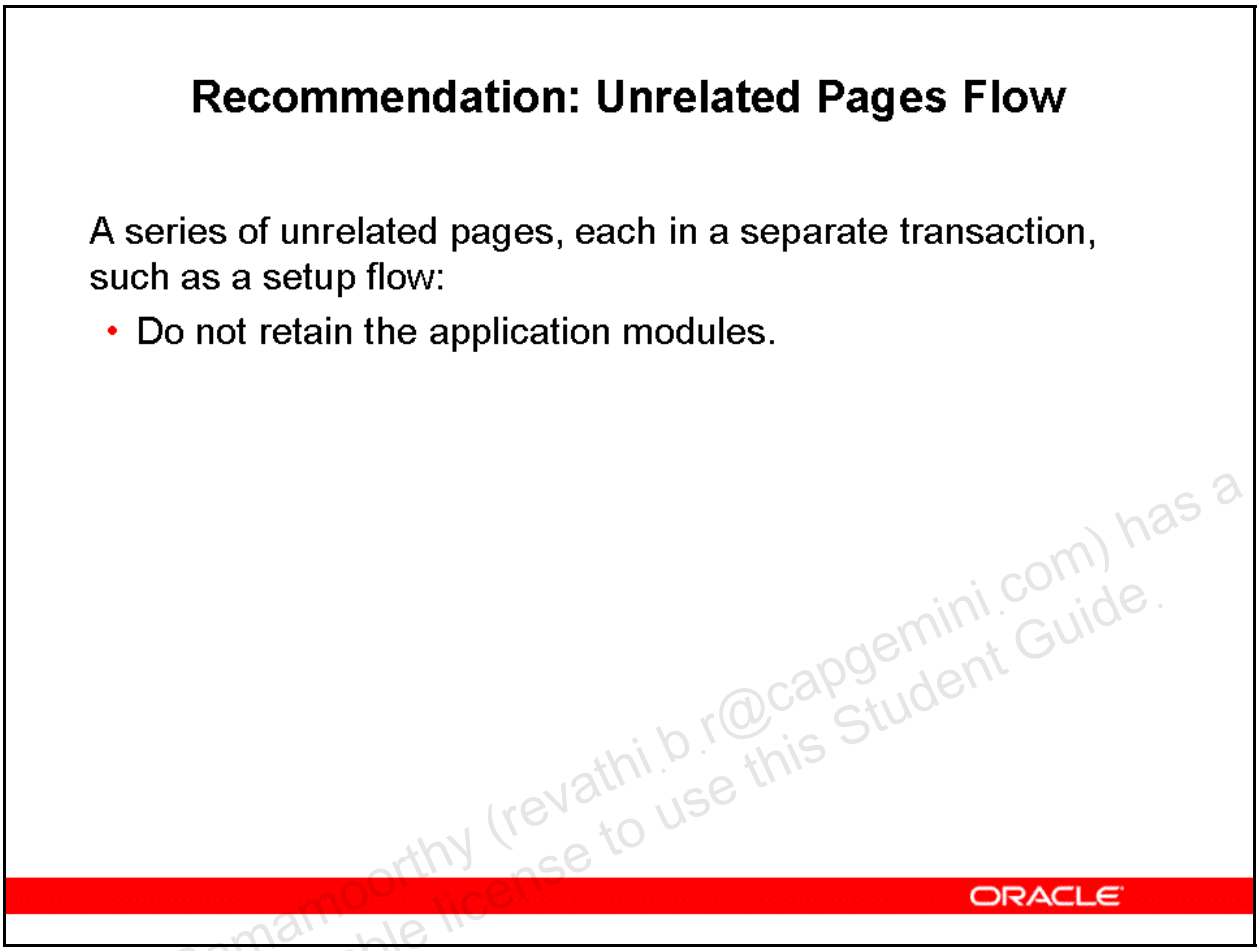
ORACLE



Recommendation: Side Trip with Extended Page

- Incorporate the side-trip flow into the main flow by using an "extended page" with the same retained AM as the main flow.
  - Create a new page with the main AM.
  - At the pageLayout region, extend the pageLayout region of the side-trip page into the new page. The side-trip page AM will be nested under the main AM.

ORACLE



Servlet Session

Servlet Session

A servlet session is a mechanism for maintaining state between HTTP requests during a period of continuous interaction between a browser and a web application. A session usually corresponds to an application login/logout cycle, but that is not strictly true in the case of OA Framework applications.

- Servlet Session persists until it times out or user logout.
- It is used for the following:
  - Expensive-to-fetch data that is used across pages and transactions
  - Only small serializable objects of Number, String and Date data types allowed
  - Accessed through the OAPageContext

ORACLE

There are two hard-coded timeouts, based on idle-time and total connection time, which the user cannot override. These are done for security reasons.

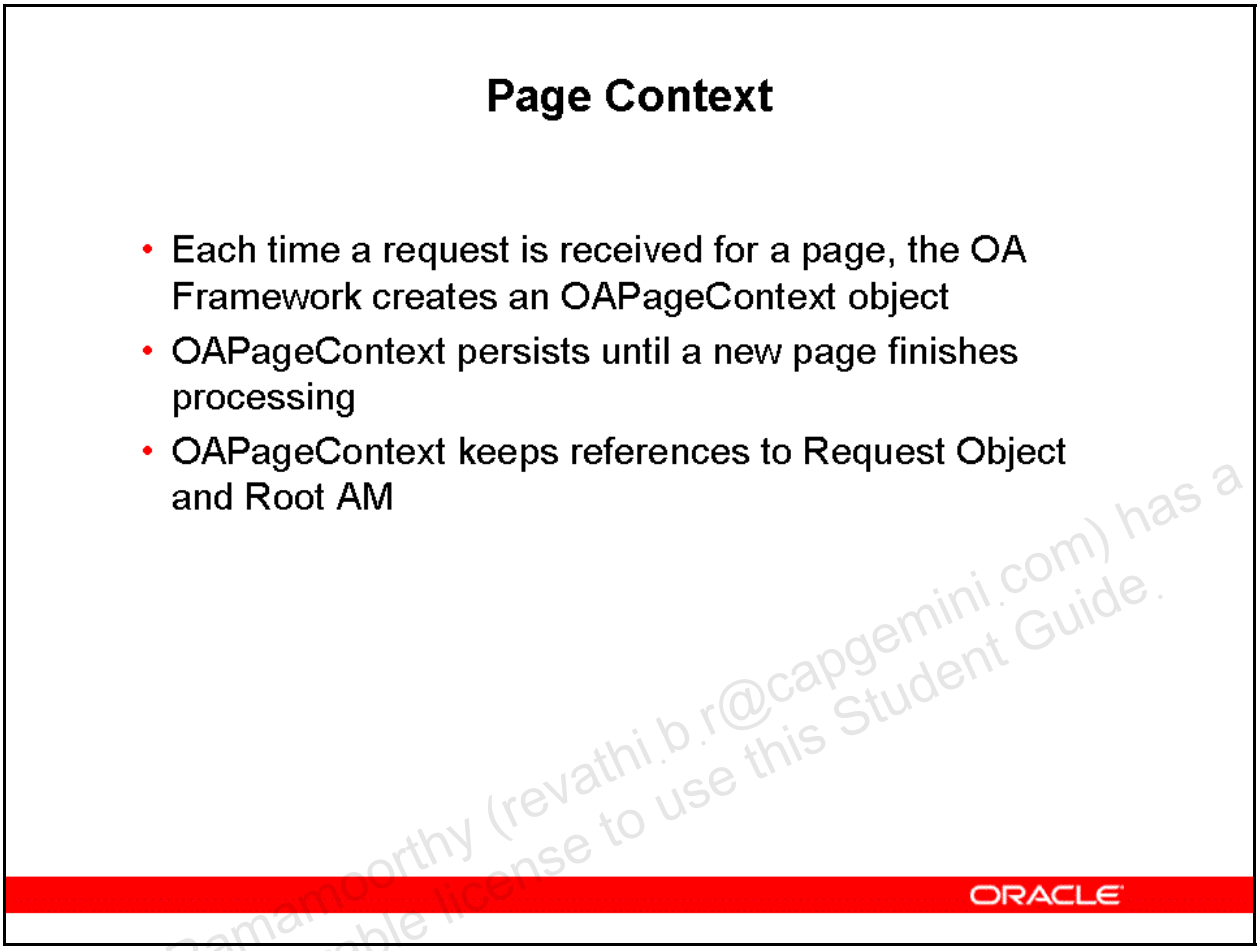
Applications User Session (ICX Session)

When the user logs in to an OA Framework application, the OA Framework creates WebAppsContext object and a browser session-based cookie that track key E-Business Suite context, responsibility, organization id, user name, user id, employee id and so on.

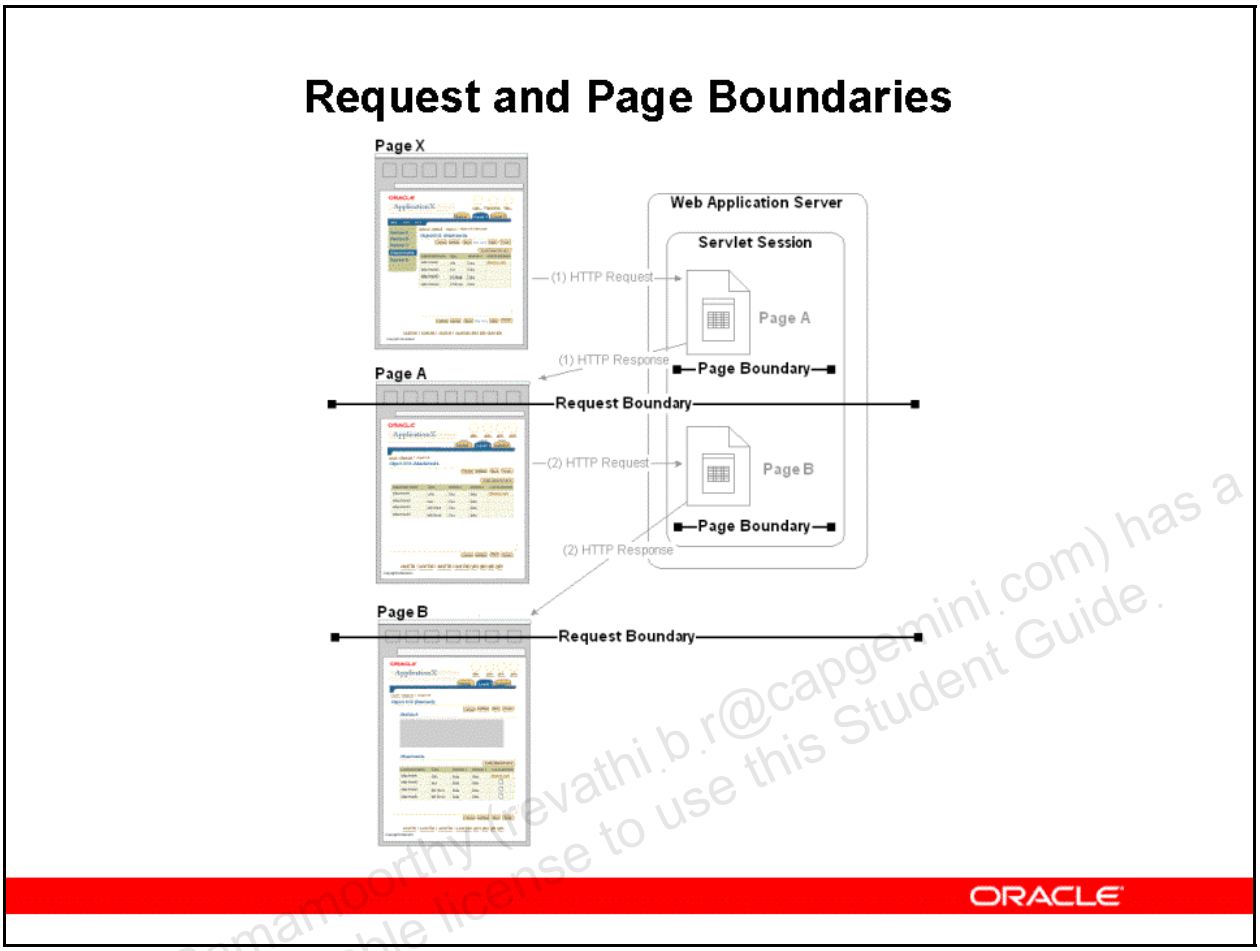
- The cookie contains an encrypted key identifier for a session row stored in the Applications database.
- User session times out based on profile settings.
  - Typically lasts longer than the Servlet Session.
  - An E-Business Suite user session might be associated with multiple servlet sessions. For example, the servlet session times out while the user takes a phone call, then resumes work before the E-Business Suite user session times out.

ORACLE

Since cookies are used, the user's browser must be set to allow cookies, or at the very least allow cookies from the EBS servers.

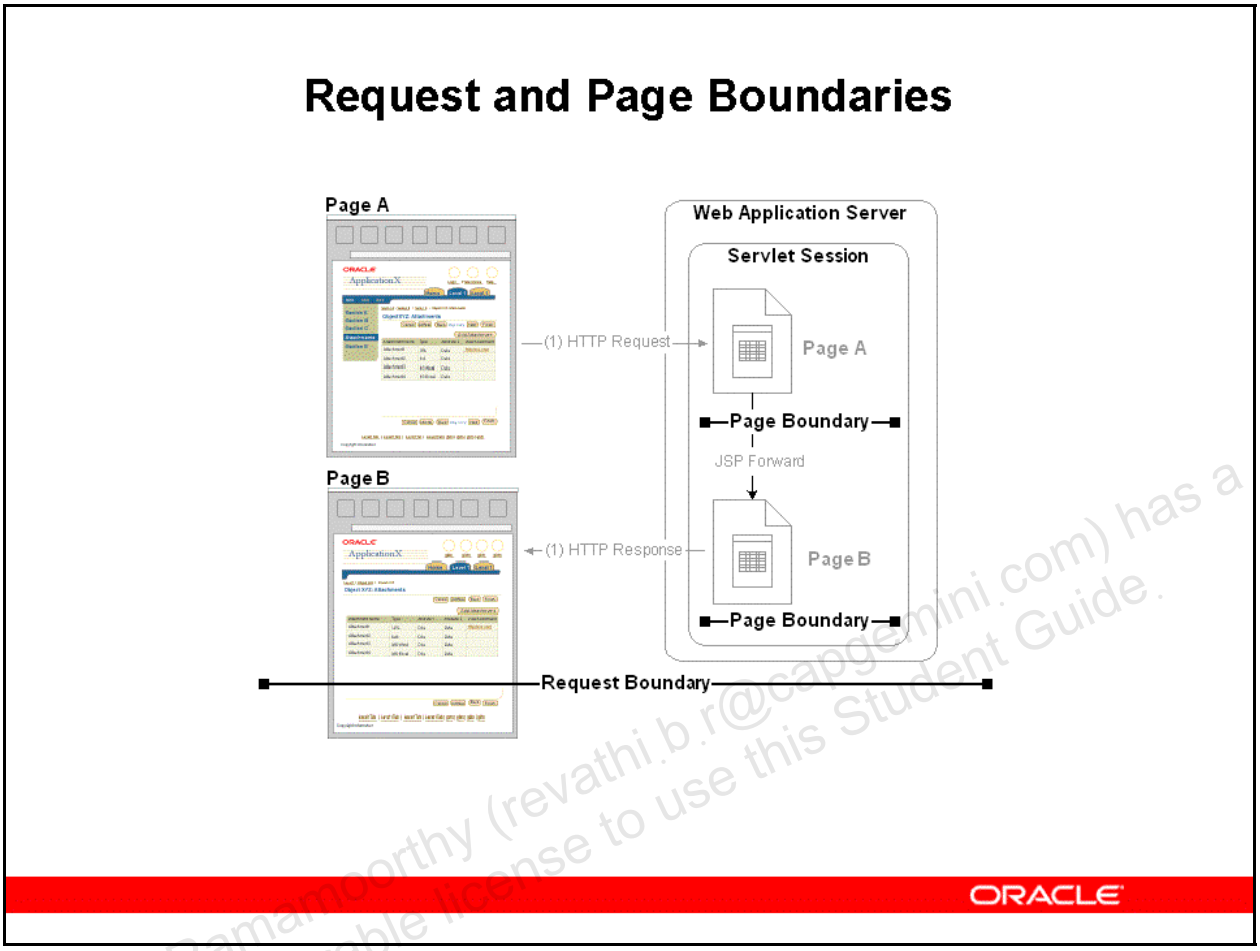


Request and Page Boundaries



A web application's unit of work is a request/response pair. The browser submits a request; the servlet processes the request and returns a response. The transmission of a response signifies the end of a single request, or the "boundary" between the completed request and a new one. Similarly, when the OAPageBean finishes processing a page, this is the "boundary" between the current page and a new one. So, in the following simple scenario where a user navigates from Page X to Page A and then to Page B, there are two request boundaries: the first is between Page X and Page A, the second is between Page A and Page B. There are also two page boundaries in the same conceptual location between Page X and Page A, and Page A and Page B.

Request and Page Boundaries



In some situations, however, the request and page boundaries are not the same. Consider the following JSP Forward case:

The user navigates from Page X to Page A as illustrated on the previous slide.

While on Page A, the user selects a control that Page A code must evaluate before deciding which page to display in response. The browser issues a request to Page A, which OA Framework processes, including creating an OAPageContext for the page. Once Page A finishes processing, the first page boundary is reached as illustrated above.

Within the Page A code, the developer evaluates which control the user selected and issues a JSP Forward to Page B.

Instead of providing an HTTP response at this point , since we don't want to redisplay Page A, OA Framework begins processing for Page B, including creating a new OAPageContext for this page.

Once Page B finishes processing, the second page boundary is reached.

Since Page B must now be displayed to the user, an HTTP response it sent to the browser.

The request boundary is now reached.

It is important to understand this distinction for several reasons:

Request parameters exist throughout the life span of the request -- which can span multiple page boundaries. This can be somewhat surprising for new OA Framework developers who simply assume that a request and a page are the same thing, and therefore do not account for request parameters "hanging around" after performing a JSP Forward.

Consider the following example:

A user selects a link in Page X that navigates to Page A. The Page A URL includes the parameter foo=bar.

Page A issues a JSP Forward to Page B. Now, even though we are in a new page, the request still includes the value foo=bar.

If you don't want a parameter value on the request after doing a JSP Forward, you must explicitly replace it. For example, in this case, simply reset the value to something like foo=X when you call the OAPageContext's setForward*() method.

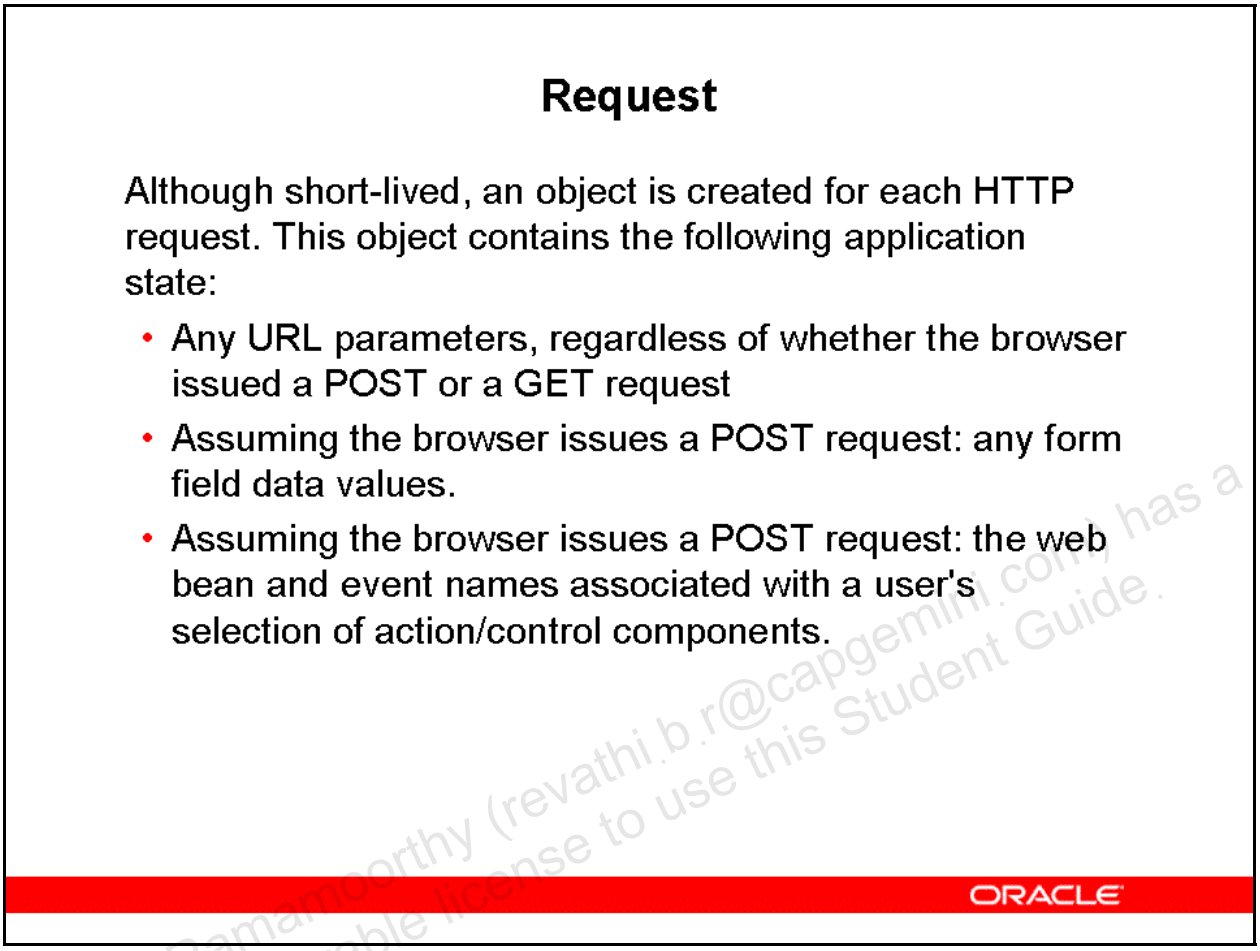
Note: You cannot actually remove parameters from a request.

Tip: It is preferable to replace the unwanted parameter value with a new one that your code can use as an "ignore" value. Do not simply set the value to "".

Since there isn't a one-to-one mapping between the page context and the request, some people find it a bit confusing that you access request parameters from the OAPageContext. Just remember that each page is a distinct entity, and from its "point of view," the OAPageContext represents the request.

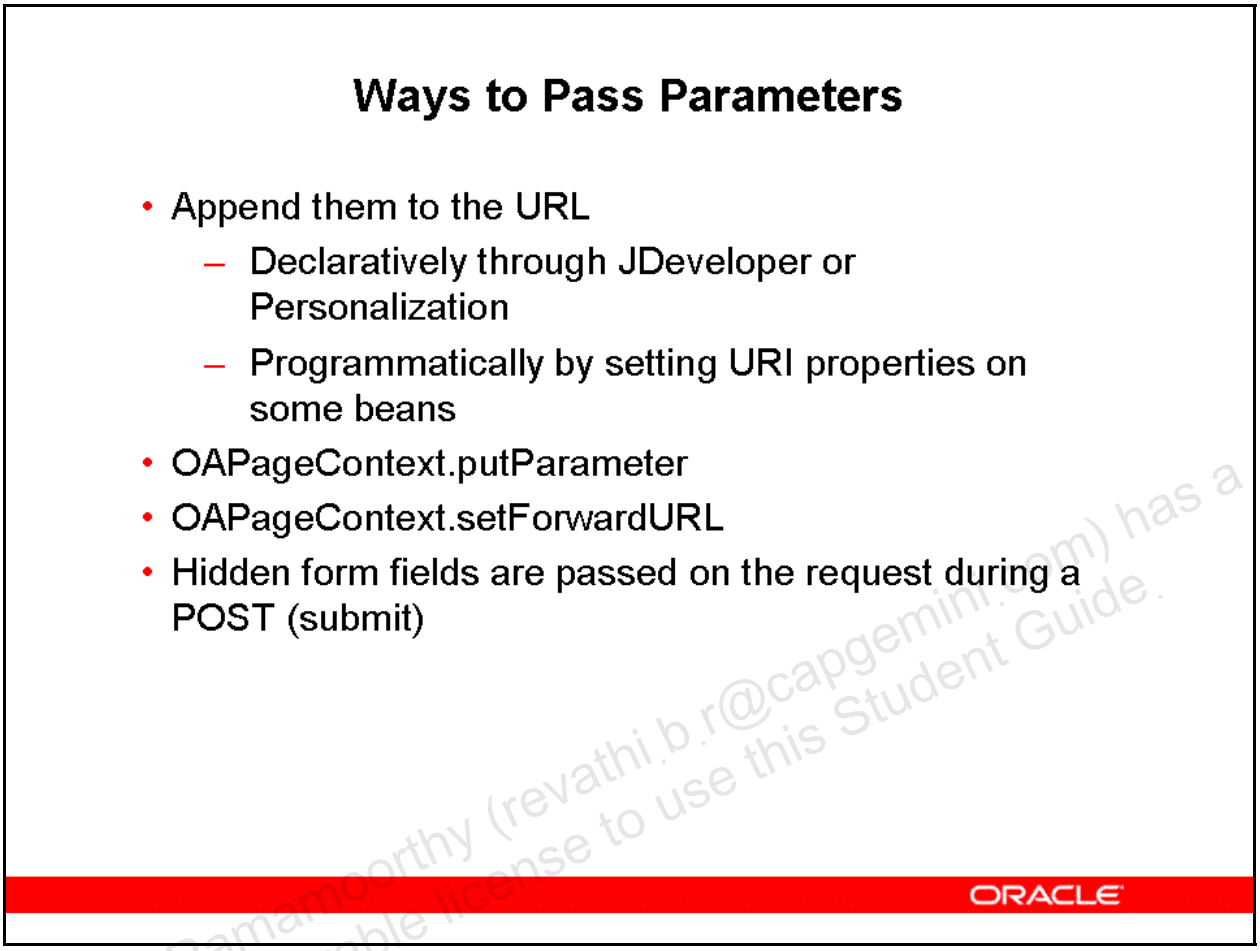
When you get into the details of passivation in Chapter 6, you'll see that page and request boundaries are distinct event points with different passivation implications.

Request



To access any of these request values, use OAPageContext getParameter*() methods. You will not interact directly with the request itself.

Ways to Pass Parameters



To put values on the request (the preferred way of communicating between pages) you can do any supported method listed below.

Note: The following is a general description of the request-passing mechanisms; there are browser Back button considerations related to each of these techniques that you should fully understand before building your pages.

Use Hidden Fields

A "hidden" field is a tool for developers to get/set values on a form that can't be accessed by the user. Just as the user's field values are added to the request during a form submit, so are the developer's field values -- assuming your page issues a POST.

You can create hidden fields declaratively in JDeveloper by selecting the formValue item style. At runtime, these are instantiated as an oracle.apps.fnd.framework.webui.beans.form.OAFormValueBean.

Specify Values During JSP Forward/Client Redirect

When you explicitly forward to a new page using the OAPageContext setForward*() methods or issue a client redirect by calling OAPageContext.sendRedirect(), you can optionally set request parameter values.

For example, Page A includes a submit button. When this button is selected, the user navigates to Page B using a JSP Forward. Page A needs to pass a "mode" value to Page B, which can be accessed several different ways so it knows how to behave.

The user selects the submit button.

In the Page A controller that handles this button press, we call OAPageContext.setForwardURL() in the processFormRequest() method. As part of this method call, we pass a request parameter named queryMode with a value of automatic.

In a Page B controller we check for the queryMode parameter value in the processRequest() method by calling getParameter("queryMode").

Page B's controller then responds to the fact that the queryMode value is automatic by immediately querying the data the page should display.

Specify Values by Calling OAPageContext.putParameter()

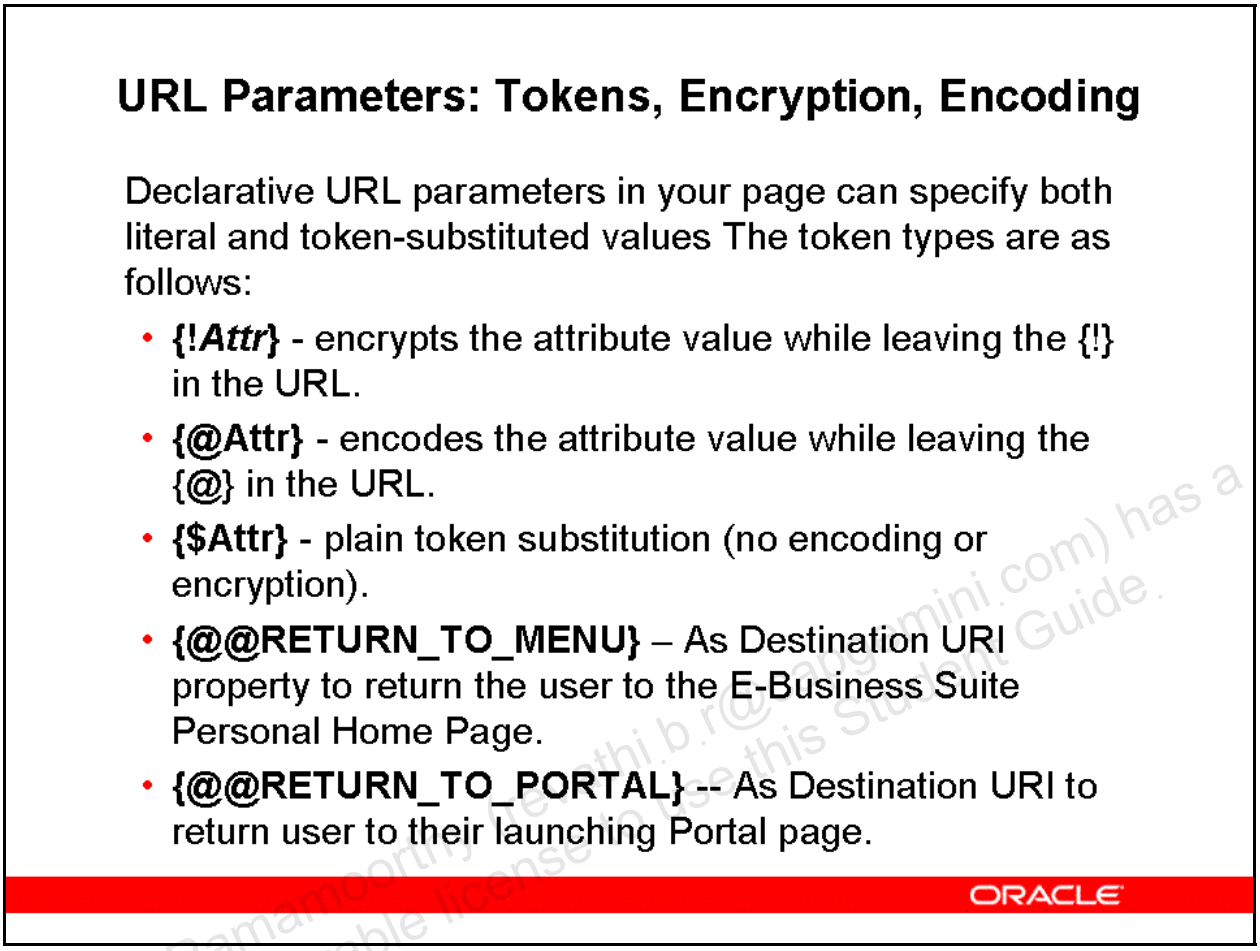
OAPageContext includes a putParameter() method that is used for passing values down the web bean hierarchy during page processing. Values specified with a call to putParameter() are not technically added to the request, but are stored in a special page cache.

Tip: For those familiar with the HttpServletRequest.setAttribute() method in the Java servlet 2.1 API, which is simply a way of stashing some information on the HTTP request, consider this its equivalent.

Set URL Parameters Declaratively

Specify request parameter values when defining URLs declaratively in JDeveloper or by setting the URL programmatically on web beans that have associated URLs.

Warning: The URL is space-constrained; be cautious about adding numerous URL parameters, particularly if they are lengthy. Since the URL is visible to users, encrypt any sensitive values.



Token Substitution

Example (using the view object attribute name "OrderNum"):

OA.jsp?OAFunc=FWK_TBX_T_PO_PAGE&order={@OrderNum}

Literal Example:

OA.jsp?OAFunc=FWK_TBX_T_PO_PAGE&order=123

Token Types

Tokens use a special character prefix to tell the OA Framework how to resolve the value at runtime (note that the concepts of "encoding " and "encryption " are described below):

{!Attr} - encrypts the attribute value while leaving the {!} in the URL

{@Attr} - encodes the attribute value while leaving the {@} in the URL

{$Attr} - plain token substitution (no encoding or encryption)

{@@RETURN_TO_MENU} - Can be used exactly as shown to specify the Destination URI property of an application component if you want it to return the user to the E-Business Suite Personal Home Page. If you need to specify this when performing a JSP forward, the corresponding constant for this is OAWebBeanValues.RETURN_TO_MENU_URL.

{@@RETURN_TO_PORTAL} -- Can be used exactly as shown to specify the Destination URI property of an application component if you want it to return the user to a launching Portal page. If you need to specify this when performing a JSP forward, the corresponding constant for this is OAWebBeanValues.RETURN_TO_PORTAL_URL.

Encoding

Any value that you specify for a request parameter must conform to HTTP syntax rules. For example, you can't pass a URL parameter value with a blank space ; the following parameter value would cause a runtime error when the corresponding URL is accessed:buyerName=John Doe. To fix this, we encode these values, meaning that the encoding routine replaces problematic characters with standard substitutions as shown in this example:buyerName=John%20Doe.

When the OA Framework adds parameters to the request (form field values, for example), it automatically encodes them.

When you put parameters on the request during a call to a setForward* method, the OA Framework automatically encodes these values.

When you put parameters on a URL that you assemble yourself (if, for example, you set a bean's URL by calling its setDestination method), you must encode any part of the String that could include invalid characters. To do this, you pass the String to an encode method on the oracle.apps.fnd.framework.webui.OAUrl utility class.

Tip: If you manually set a URL parameter value that can't include invalid characters (for example, "value=Y") then you don't need to bother with the encoding step.

When you put values on the request using OAPageContext.putParameter, you must encode the String if necessary.

The OA Framework automatically decodes parameter values when you call the OAPageContext.getParameter* methods, except for the following cases:

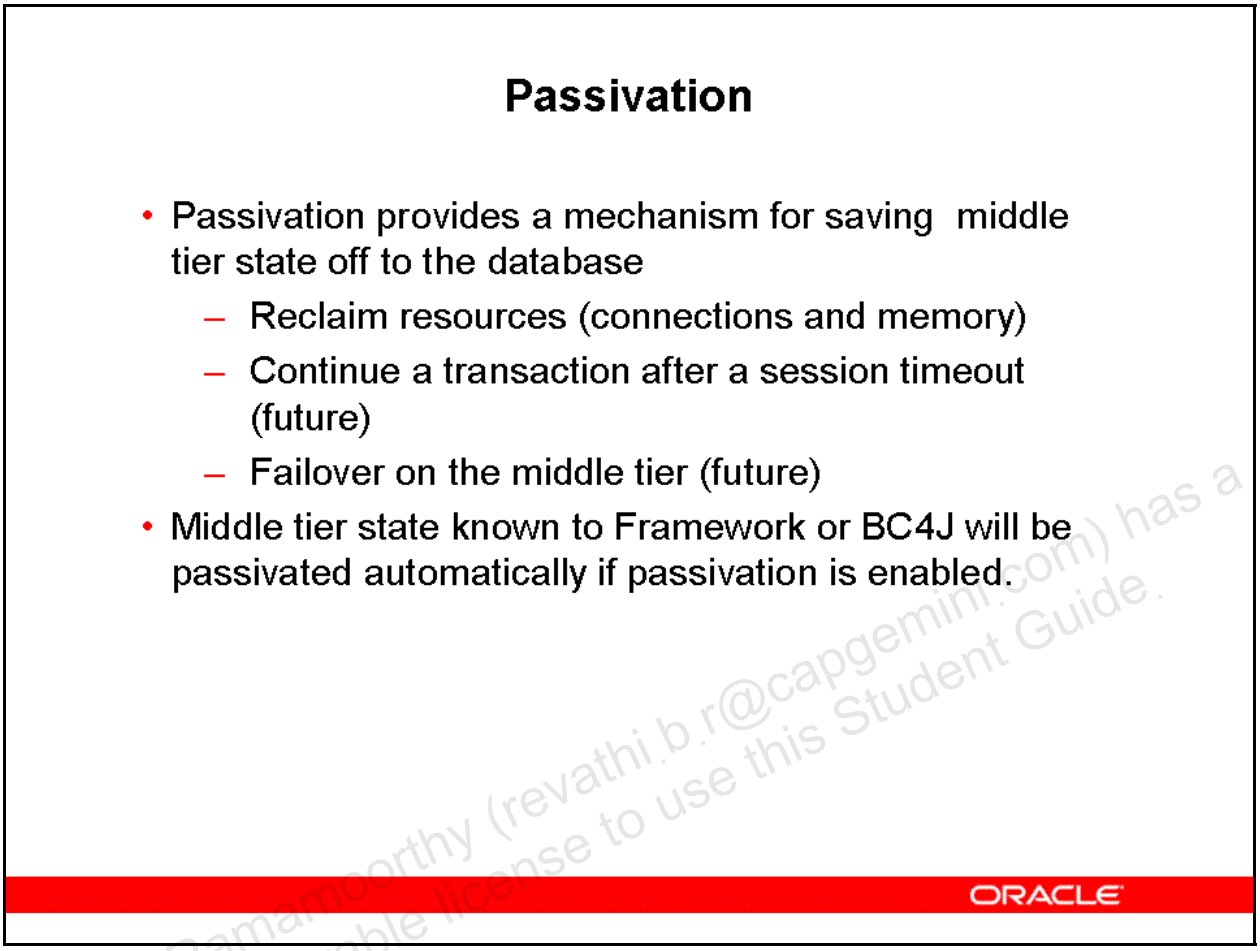
When you use the "#" character for Javascript function tokens, the OA Framework encodes the token values, but it does NOT automatically decode them when you call pageContext.getParameter("<tokenName>"). To do this yourself, you'll need to use the OAUrl decode method on the value that getParameter returns.

When you call putParameter with an encoded value, the OA Framework does not decode it. You must also use the OAUrl decode method in this case on the value the getParameter returns.

Encryption

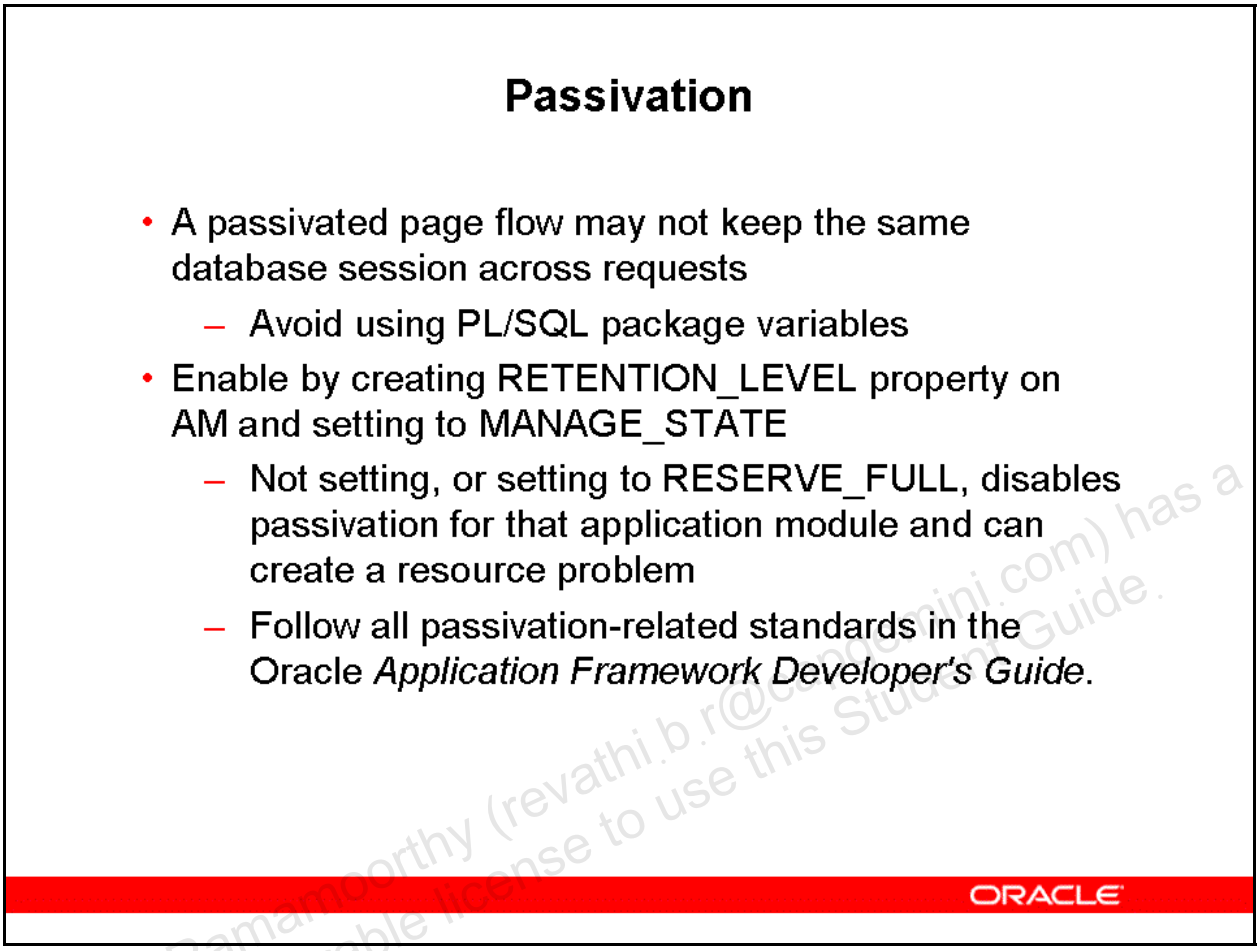
Encryption is the process of obfuscating data to make it illegible. Since URL request parameter values may be visible to the user (and hidden form field values if the user opts to view the HTML page source), you should always encrypt sensitive data if stored in a URL parameter or a hidden field. In addition to the declarative, token-based encryption described above, the OA Framework also provides methods in oracle.apps.fnd.framework.webui.OAPageContext for manually encrypting and decrypting any parameter values that you put on the request programmatically.

Passivation



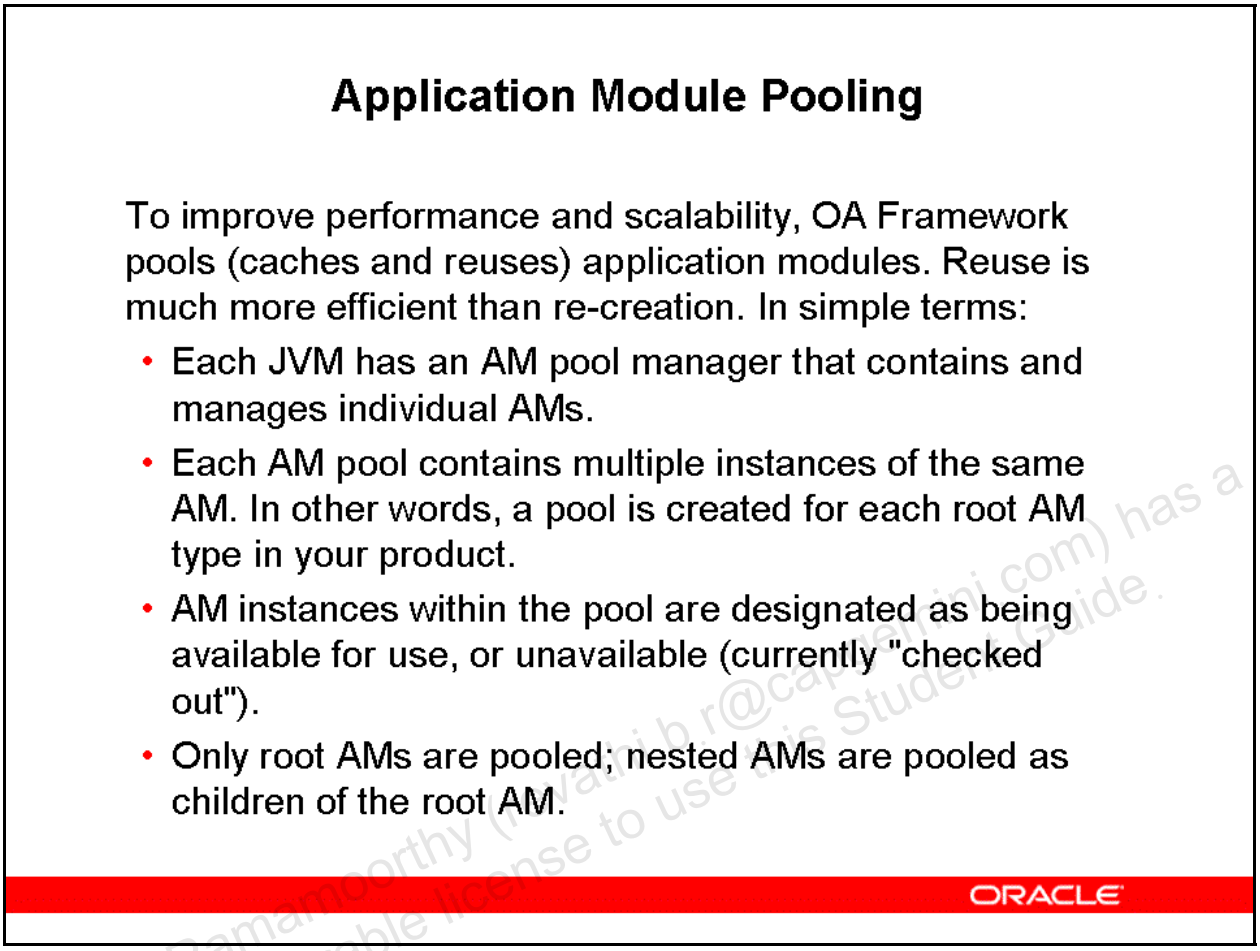
Note: Passivation is not being released in R12, but there are already actions that you can take to allow passivation (such as setting the RETENTION_LEVEL property on your AMs to MANAGE_STATE) and the Transaction Undo feature, which leverages passivation and is supported.

Passivation



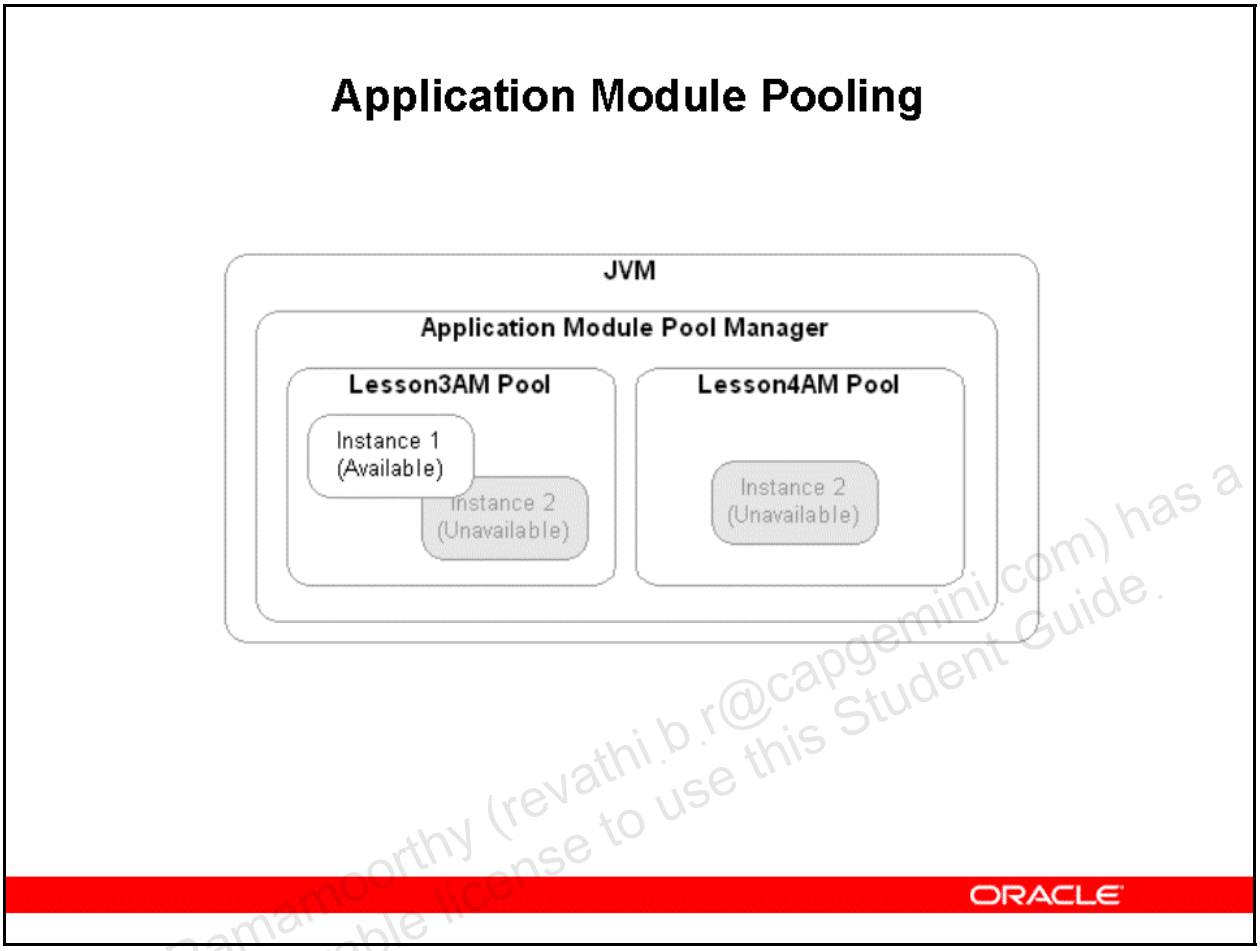
Passivation is not fully implemented in most systems due to the significant resources requirements. However, such common features as “Save for Later” are artifacts of the Passivation technology.

Application Module Pooling

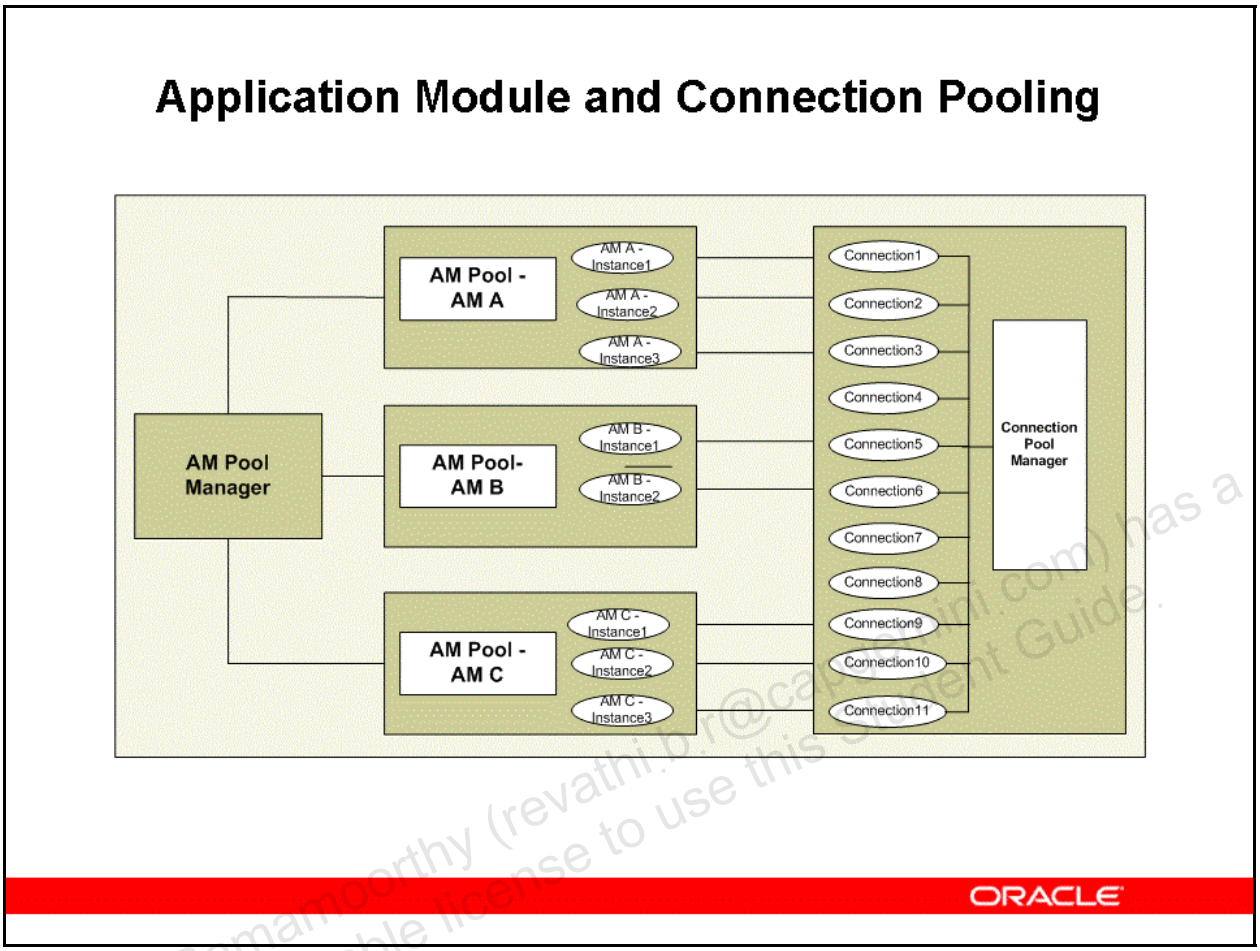


Without connection pooling, it would be both possible and probable for there to be connection leaks. Given that opening and closing connections are expensive instructions, it significantly impacts middle-tier performance if there is no AM pooling.

Application Module Pooling



Application Module and Connection Pooling



Pooling is a mechanism where cached objects kept in a runtime object pool can be used and reused as needed by an application. Instead of creating an object when requested by an application and destroying it when released by the application, you can pool these objects and reuse them when required. This saves the creation time associated with the object and improves performance and scalability.

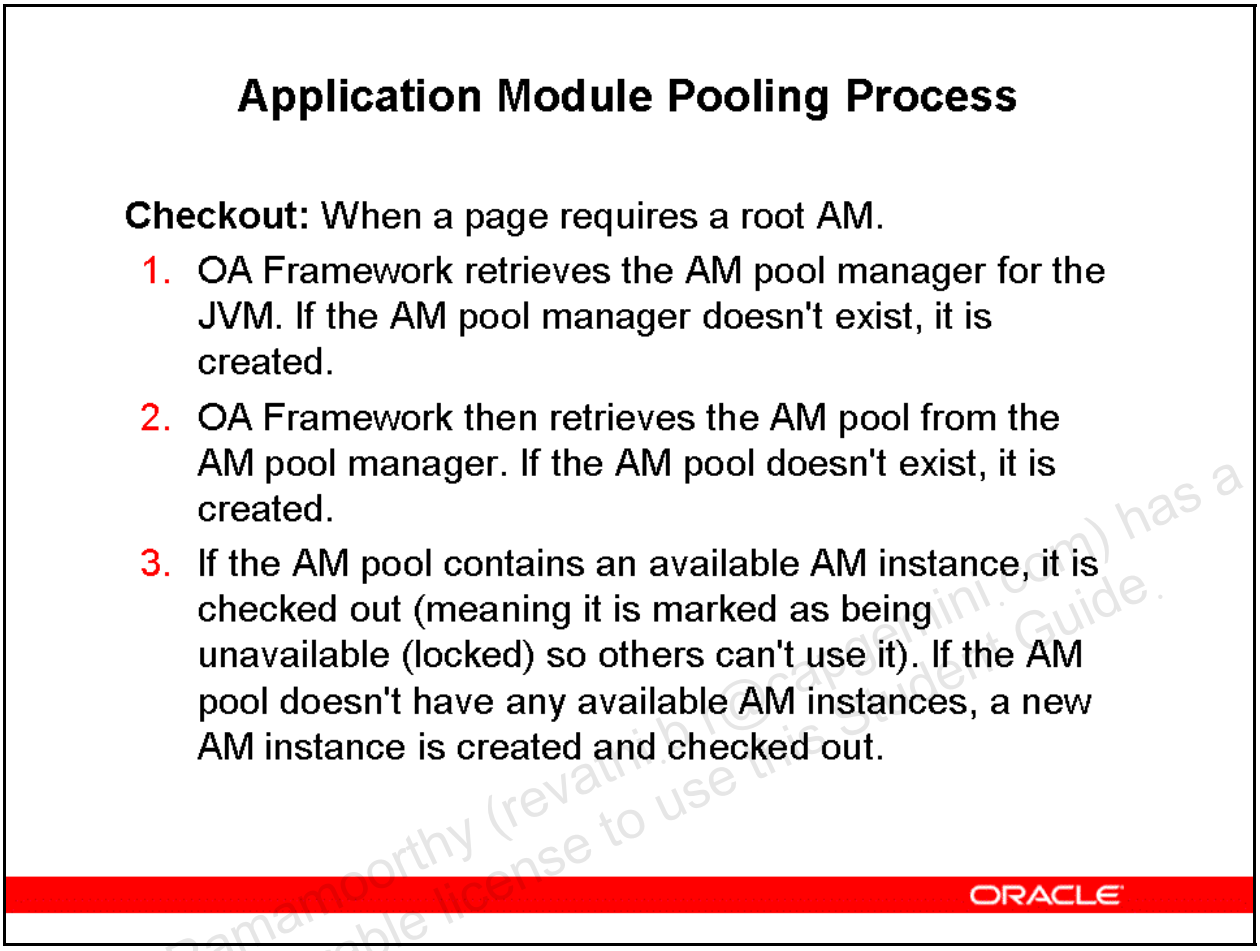
A pool manager manages the objects in the pool. When an object is requested, it looks for an available object in the pool. If there isn't any, it creates one, up to the maximum allowable objects. If it can't create an object, it waits for one to be released and depending on the implementation may eventually display an error indicating that it was unable to retrieve the object.

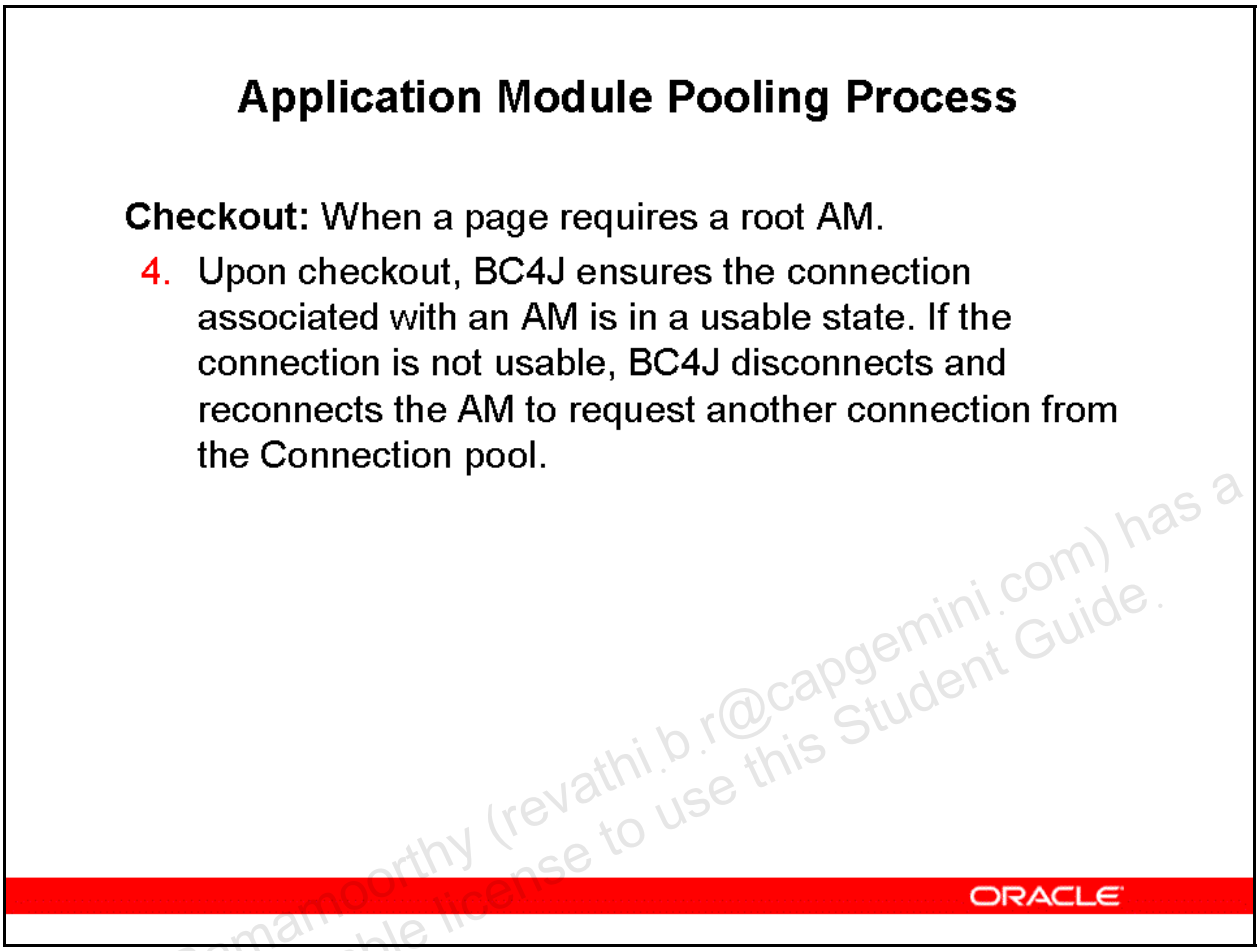
The OA Framework pools the application modules and the connections used by application modules. This chapter describes these two pools and the various options that configure each pool. You will also learn how to diagnose and troubleshoot performance and scalability issues associated with pooling in your application.

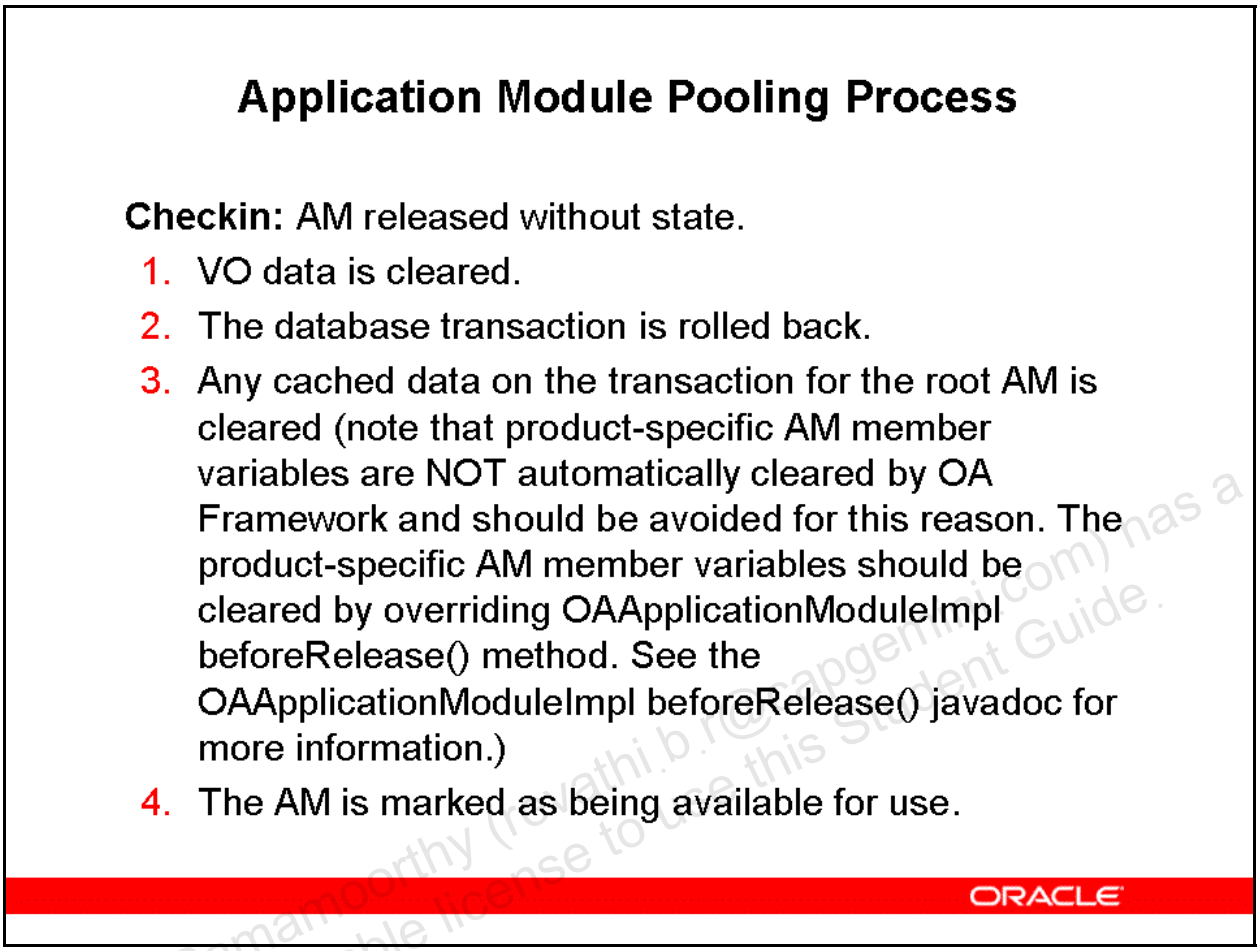
The Application Module (AM) instances are pooled in an Application Module Pool (also known as AM Pool). Each JVM has an application module pool manager that contains and manages individual application module pools. Each application module pool contains multiple instances of the same application module definition. In other words, a pool is created for each root application module definition (type) in your

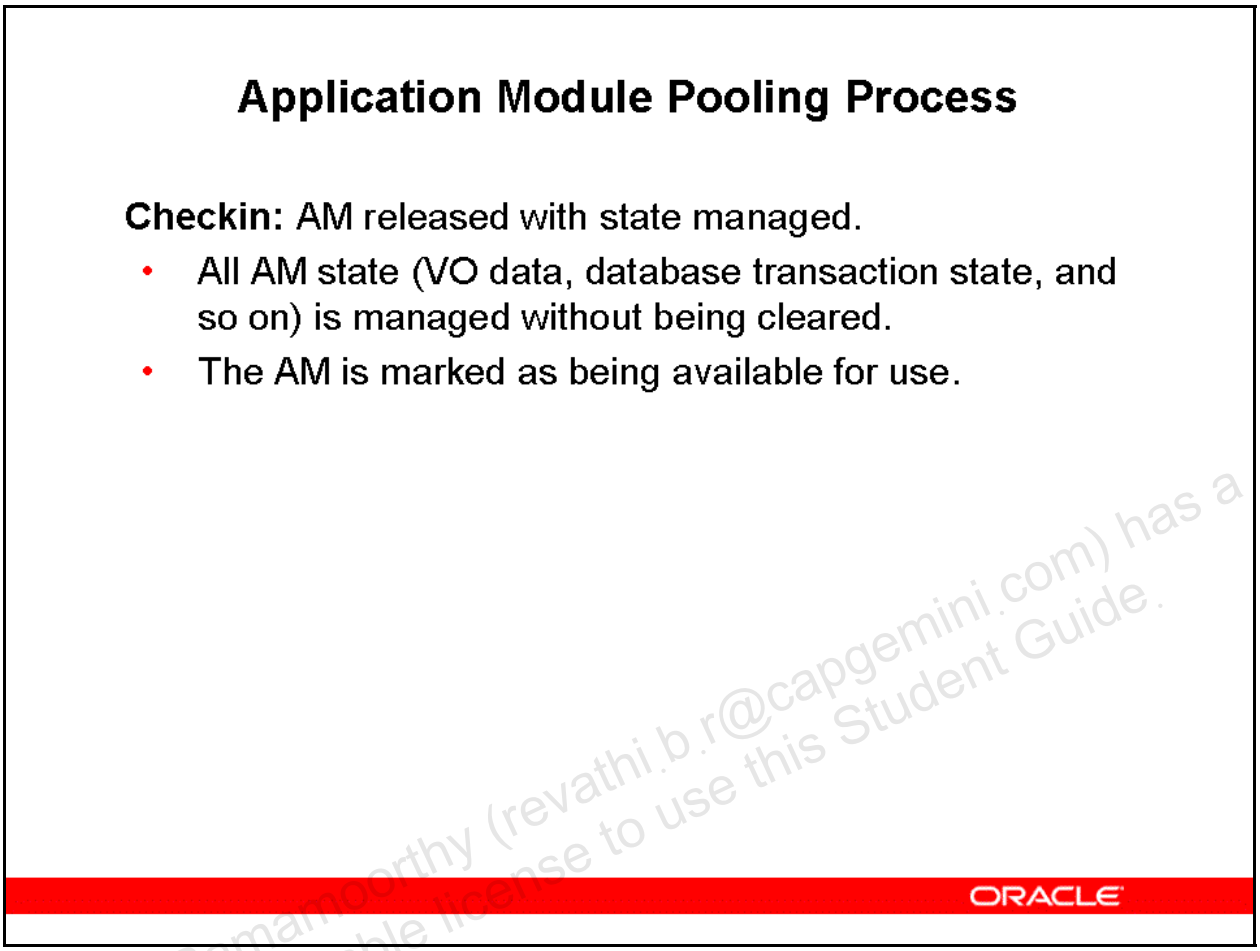
product. Application module instances within the pool are designated as being available for use, or unavailable (currently "checked out"). Only root application modules are pooled. Nested application modules are pooled as children of the root application module; they cannot be checked out or checked in independently of the parent.

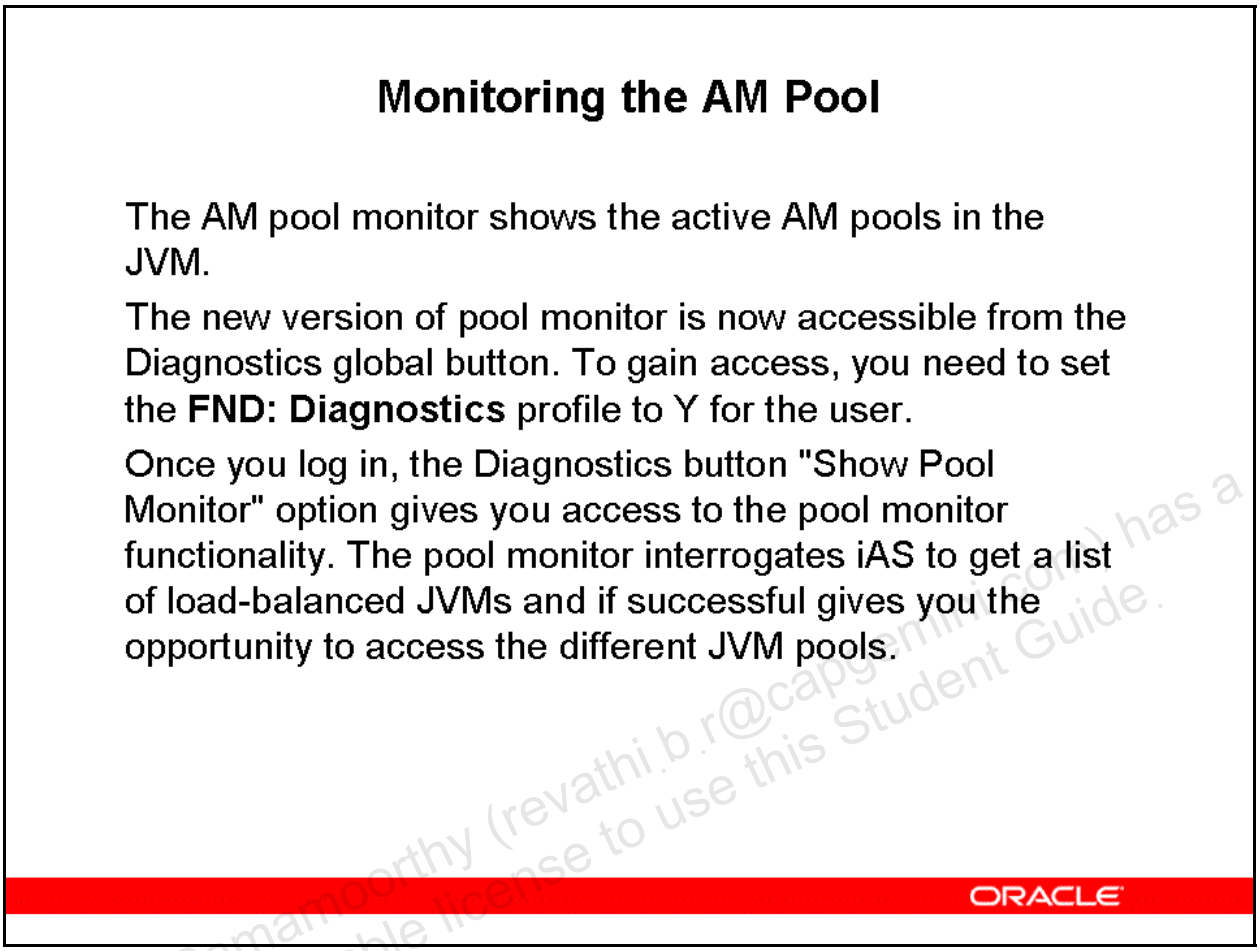
In addition to pooling the AMs, the OA Framework relies upon AOL/J to pool the connections used by them. The AMs reuse a pool of connections instead of creating a JDBC connection for every new instance and destroying it when the instance disconnects. A connection pool is assigned for each JDBC connection URL; currently in OA Framework, only one DBC file (and hence, only one JDBC connection URL) is allowed per JVM.











The functionality to access all JVMs is only available with iAS 1.0.2.2.2 or higher. If you are using an earlier version or the JVM list can not be obtained, then you are only able to inspect the pool in the current JVM you are connected to.

JDBC Connection Pooling Process

Checkout:

1. Every active AM has a dedicated JDBC connection. When the AM is created, it requests a JDBC connection from the JDBC Connection pool.
2. The JDBC Connection Pool Manager tries to select a usable connection from the list of available connections and returns one if found.
3. If the selected connection is not usable, or if there are no available connections, then the pool checks to see if it has already reached its maximum size. If the pool is not at maximum size, it tries to create a new connection for the client.

ORACLE

JDBC Connection Pooling Process

Checkout:

4. If the pool is already at maximum size, the JDBC connections are detached (harvested) from the inactive AMs and released back to the pool for reuse. If it cannot create a new connection, it records an error and returns null.

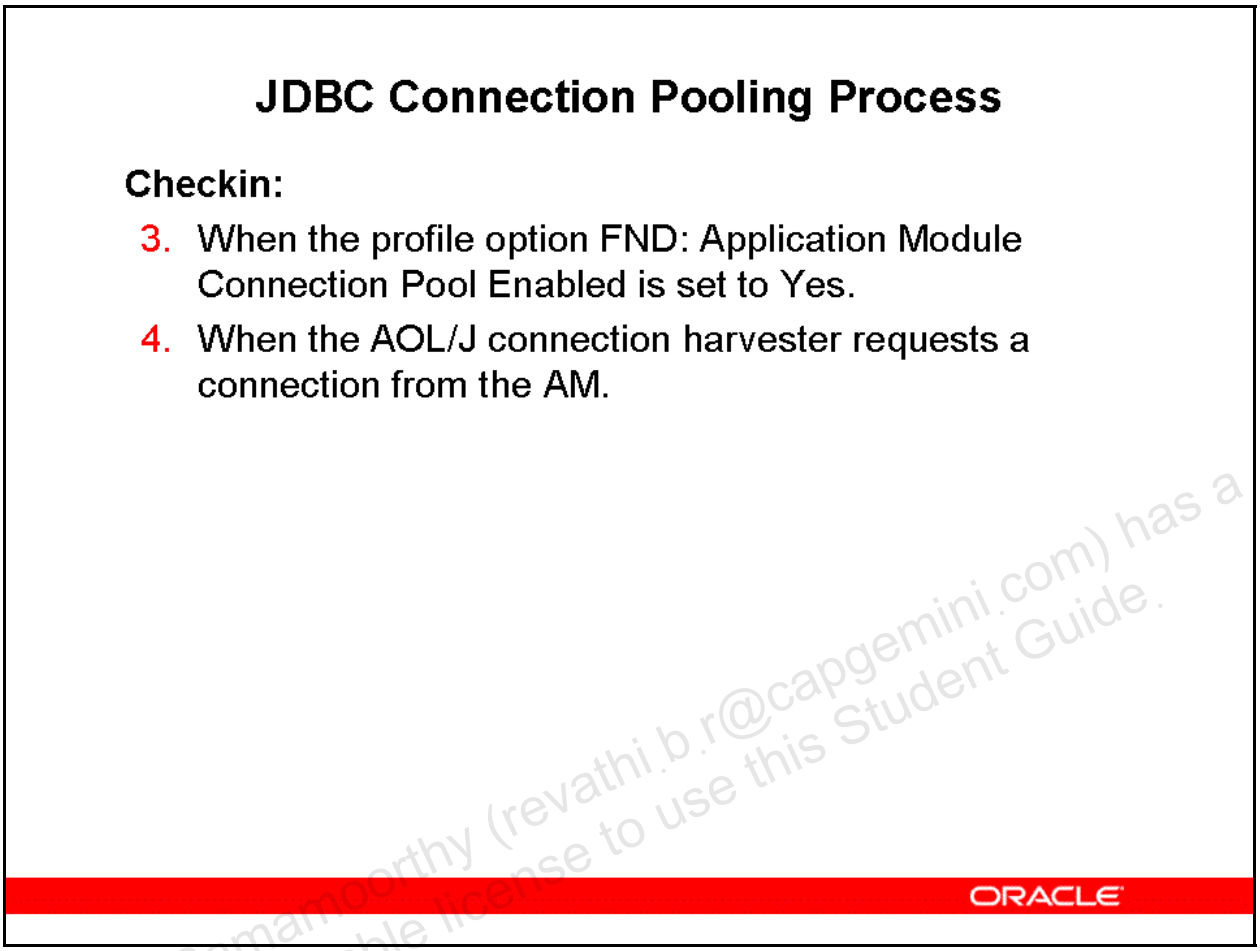
JDBC Connection Pooling Process

Checkin:

Once a connection is checked out by the AM, it remains locked. Note that checking an AM into the AM pool does not check in the connection back to the connection pool by default. Instead, connections are lazily checked back into the connection pool in cases as below:

1. When the AM is destroyed by the AM Pool Monitor cleanup thread.
2. When the AM needs to disconnect and reconnect upon checkout to acquire a usable connection in place of an unusable (closed or timed out) connection.

ORACLE



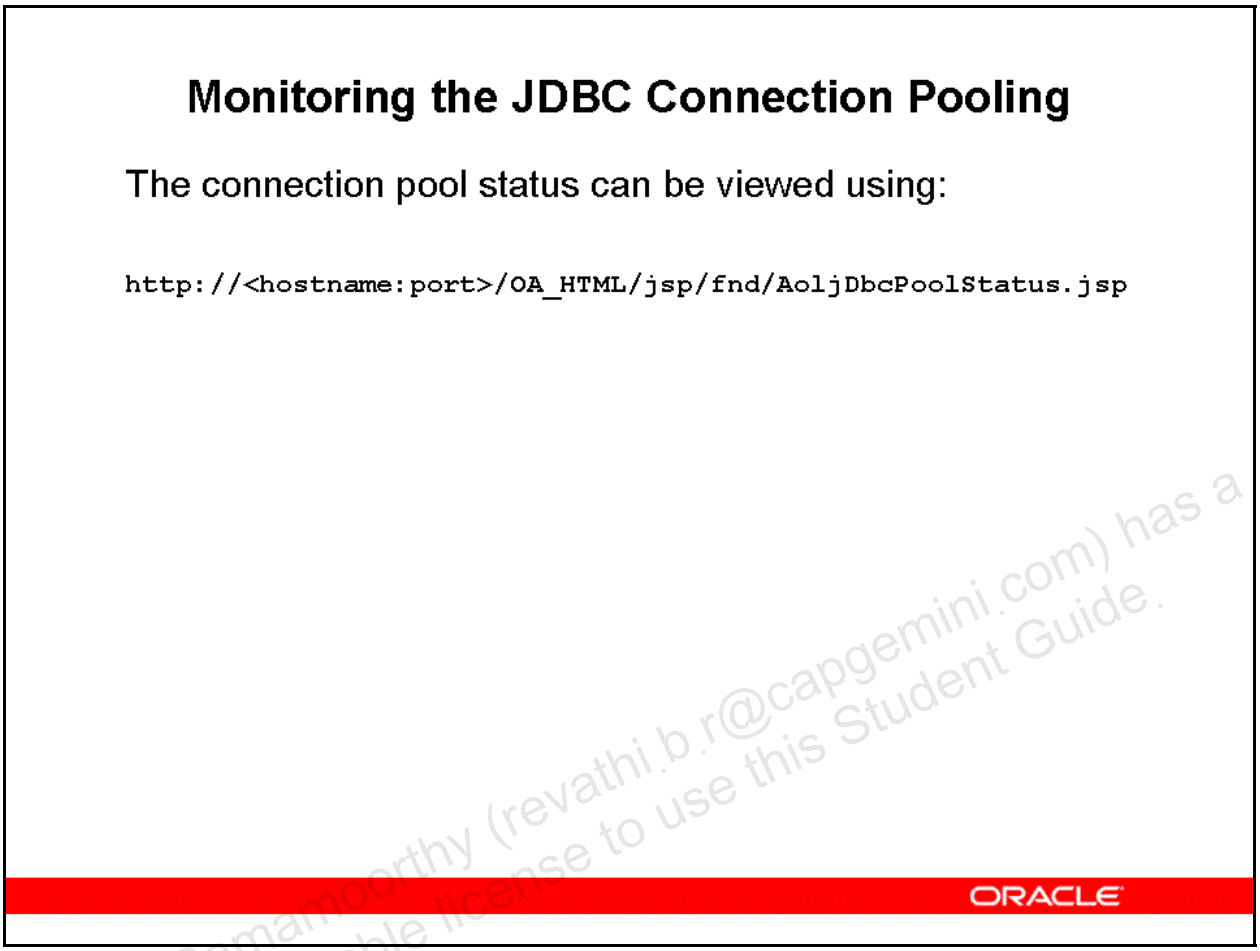
JDBC Connection Pooling Process

Cleanup:

The connection pool tries to maintain a number of available connections between the buffer min and buffer max values. When the usage is high, the pool size may grow considerably, and hence, the cleanup thread removes (destroys) connections. When the usage is low, the pool shrinks down below the buffer max size, and hence, the cleanup thread creates connections.

The cleanup happens over a period of time controlled by the decay interval and the decay size. The decay size is set to 1 by default, and you can increase this to 5 to speed up the decaying process. You should however leave the decay interval at its default 60 seconds.

ORACLE



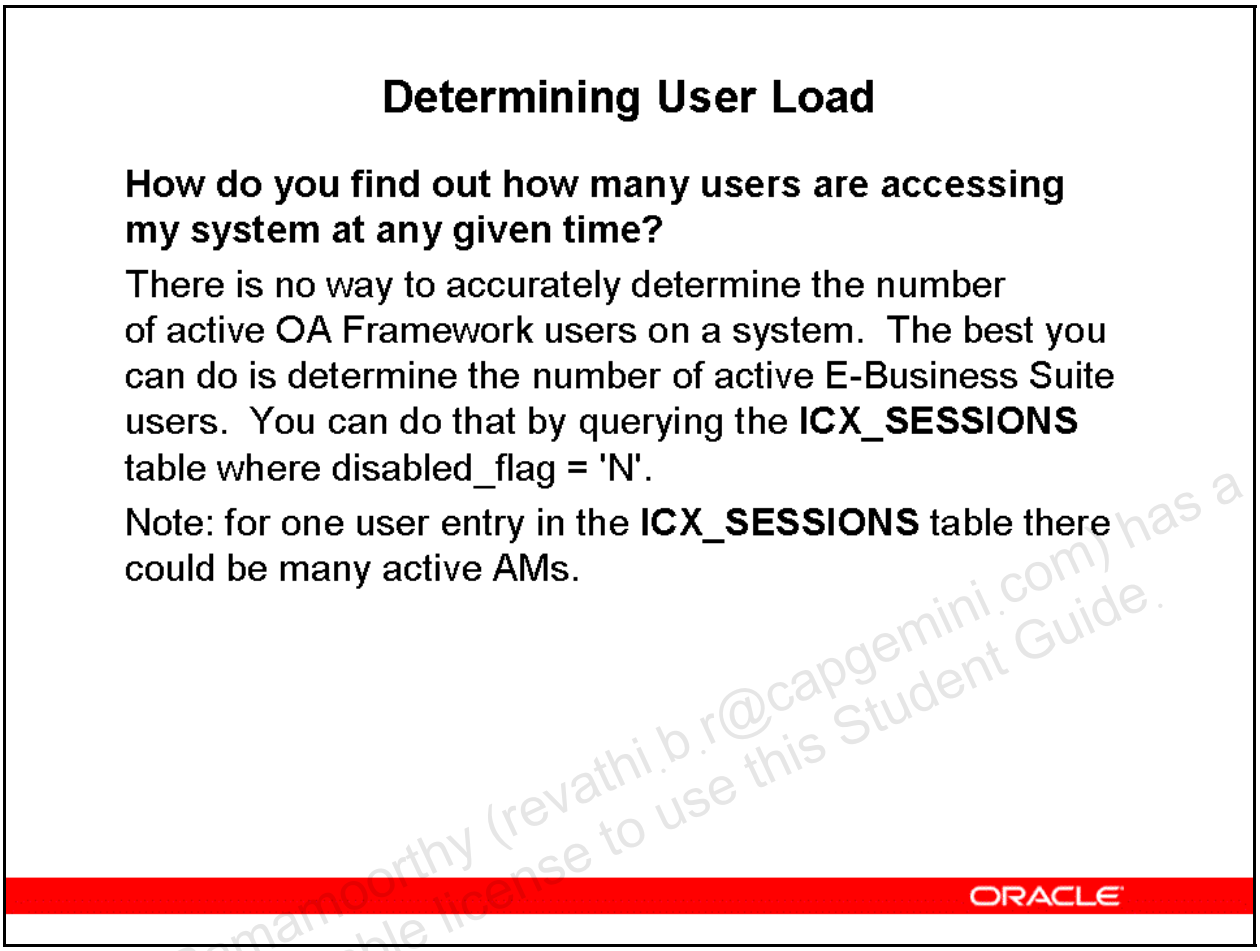
Monitoring the JDBC Connection Pooling

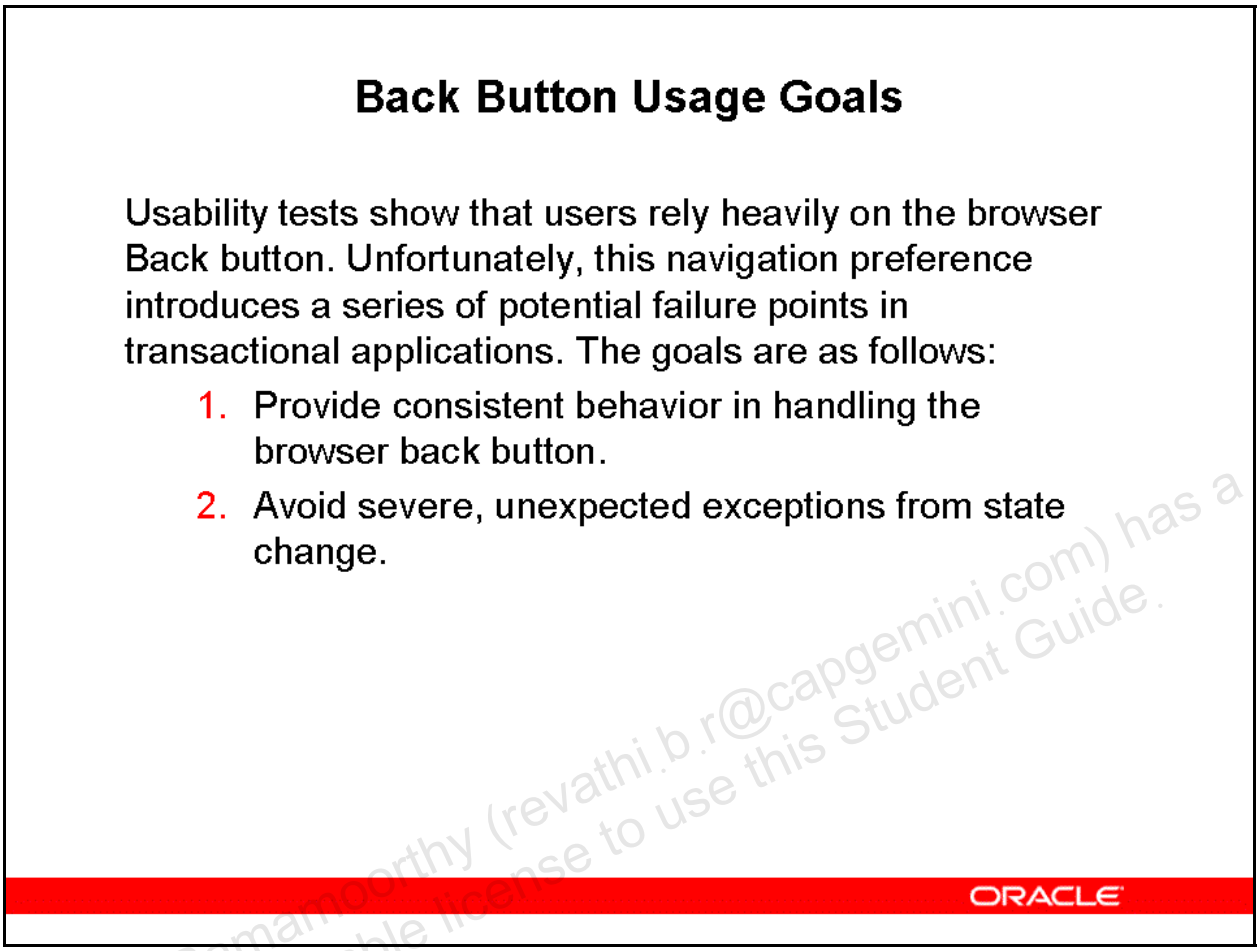
Inspecting the Connection Pool Monitor

You can inspect the connection pool monitor for the following:

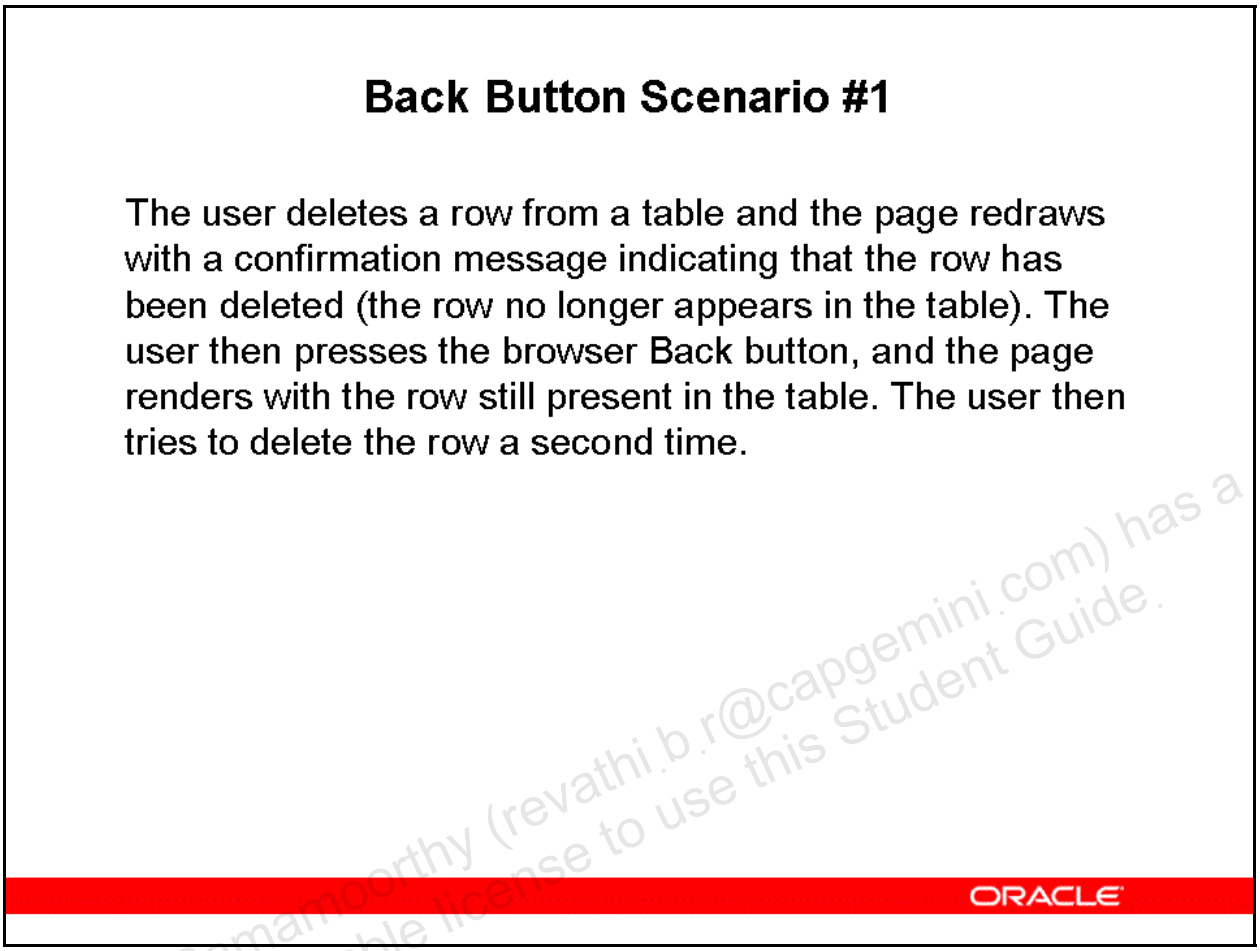
- Available Connections: The number of connections that are available for checking out.
- Locked Connections: The number of connections that were checked out by clients that use the JDBC Connection Pool. This includes the OA Framework AMs, other JSPs, servlets and so on. By clicking on locked connections, you can get a list of all users and the connections held by them.
- Pool Size Counter: Total number of available and used connections in the pool.

ORACLE





Back Button Scenario #1



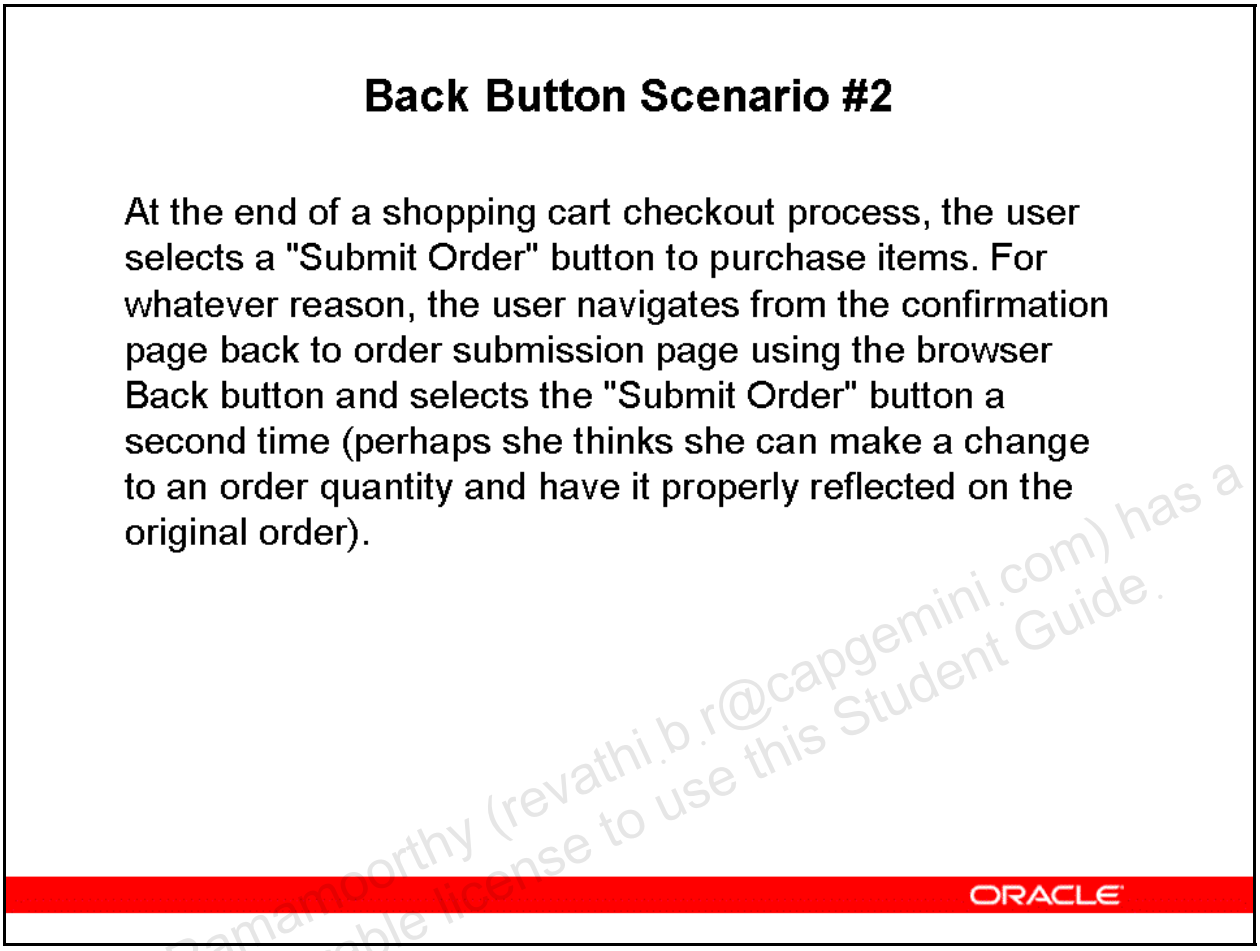
Problem

The browser caches page contents. If the user performs an action that changes data state, and then uses the browser Back button, the browser's page cache doesn't reflect the correct middle tier state (in this case, that a row no longer exists).

An attempt by the user to delete, or otherwise transact, this deleted row in the cached page would likely result in an ugly runtime application error.

A Back button compliant page would detect the attempt to transact a deleted row and fail gracefully by displaying a user-friendly message indicating that the row has already been deleted.

Back Button Scenario #2

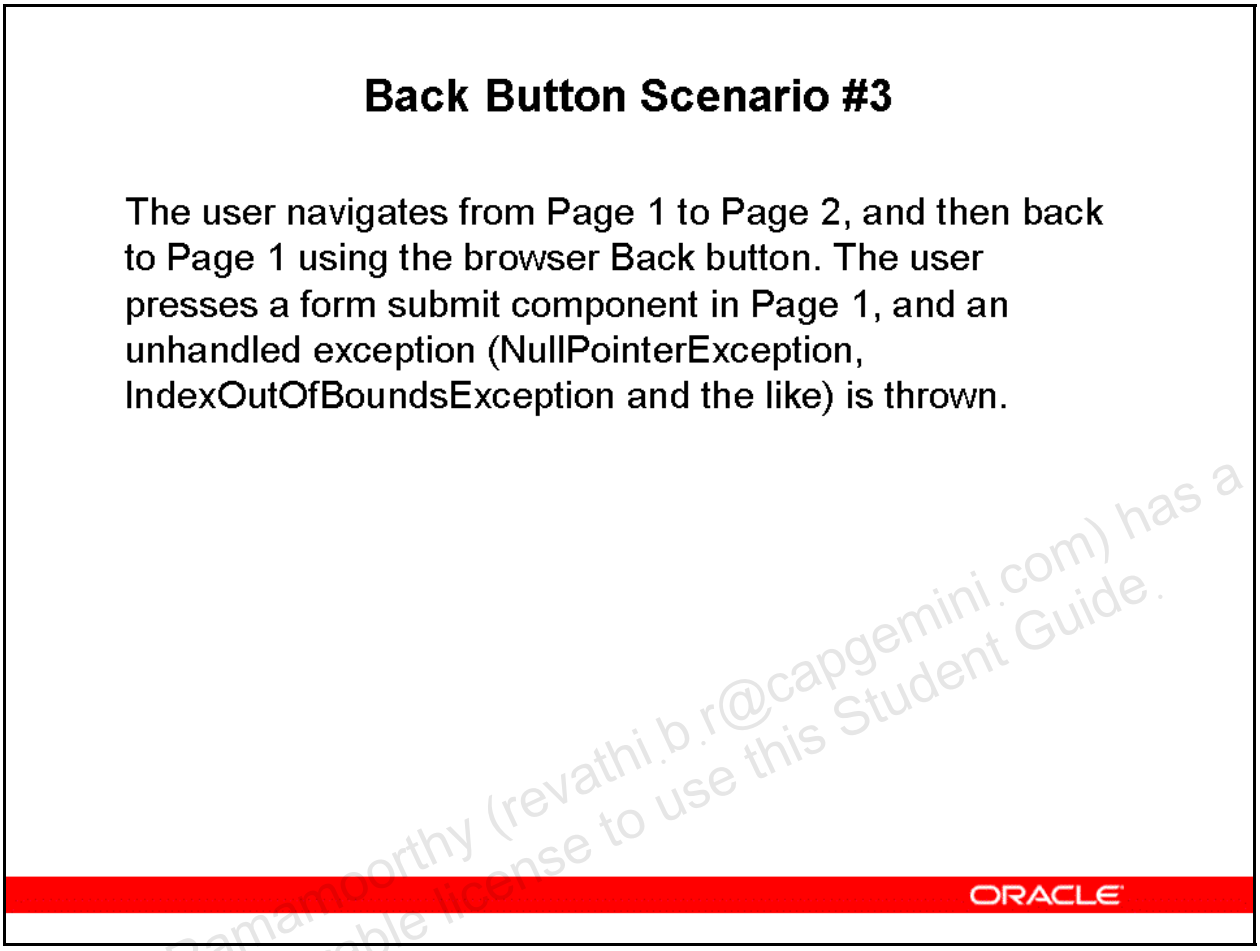


Problem

This scenario is similar to the one described above, however, the unguarded action could result in "successful" duplicate transaction (the order might be created twice, which is unlikely to be the user's expectation).

A Back button compliant page would detect an attempt to submit the same order twice and fail gracefully by displaying a user-friendly error message indicating that the order had already been placed.

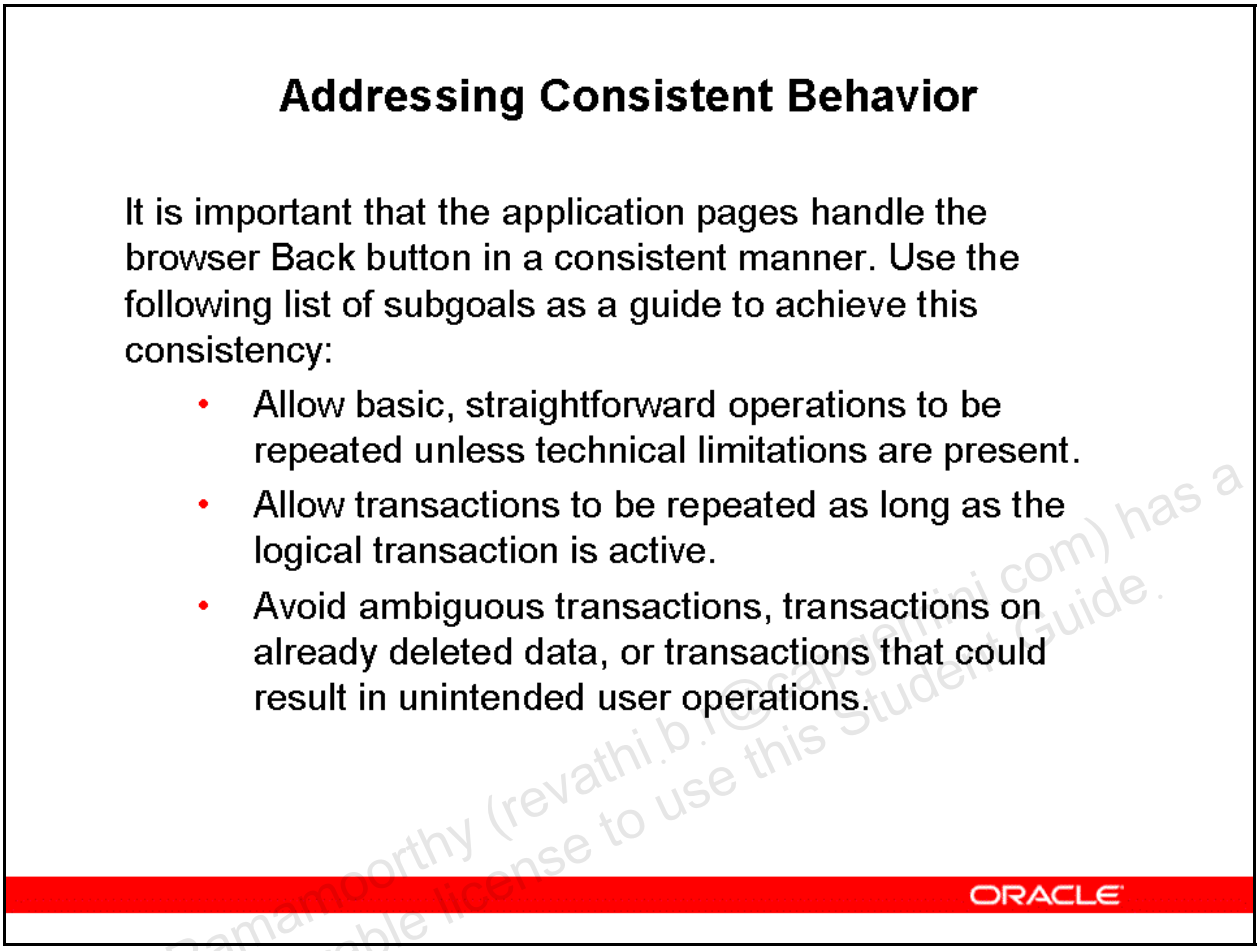
Back Button Scenario #3

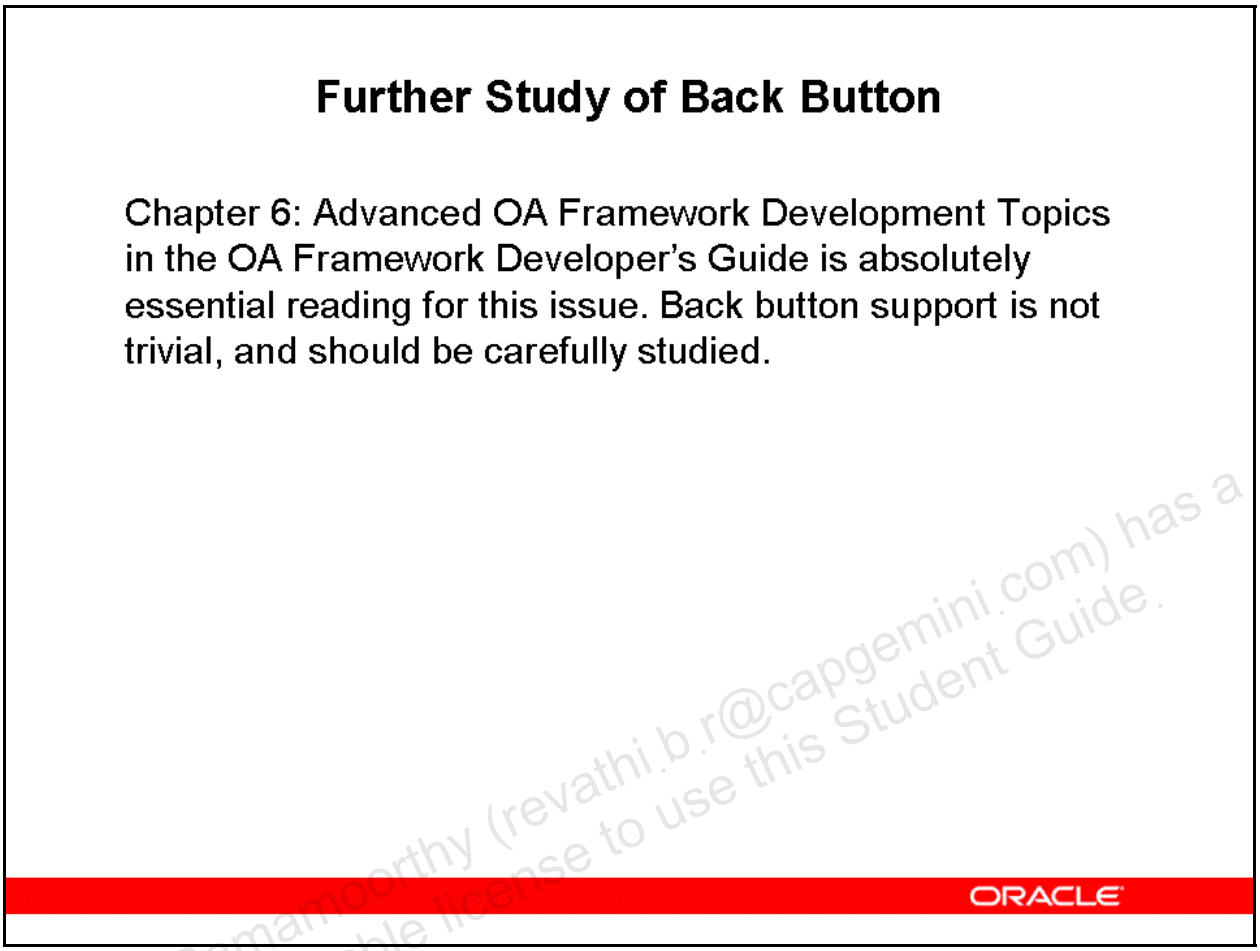


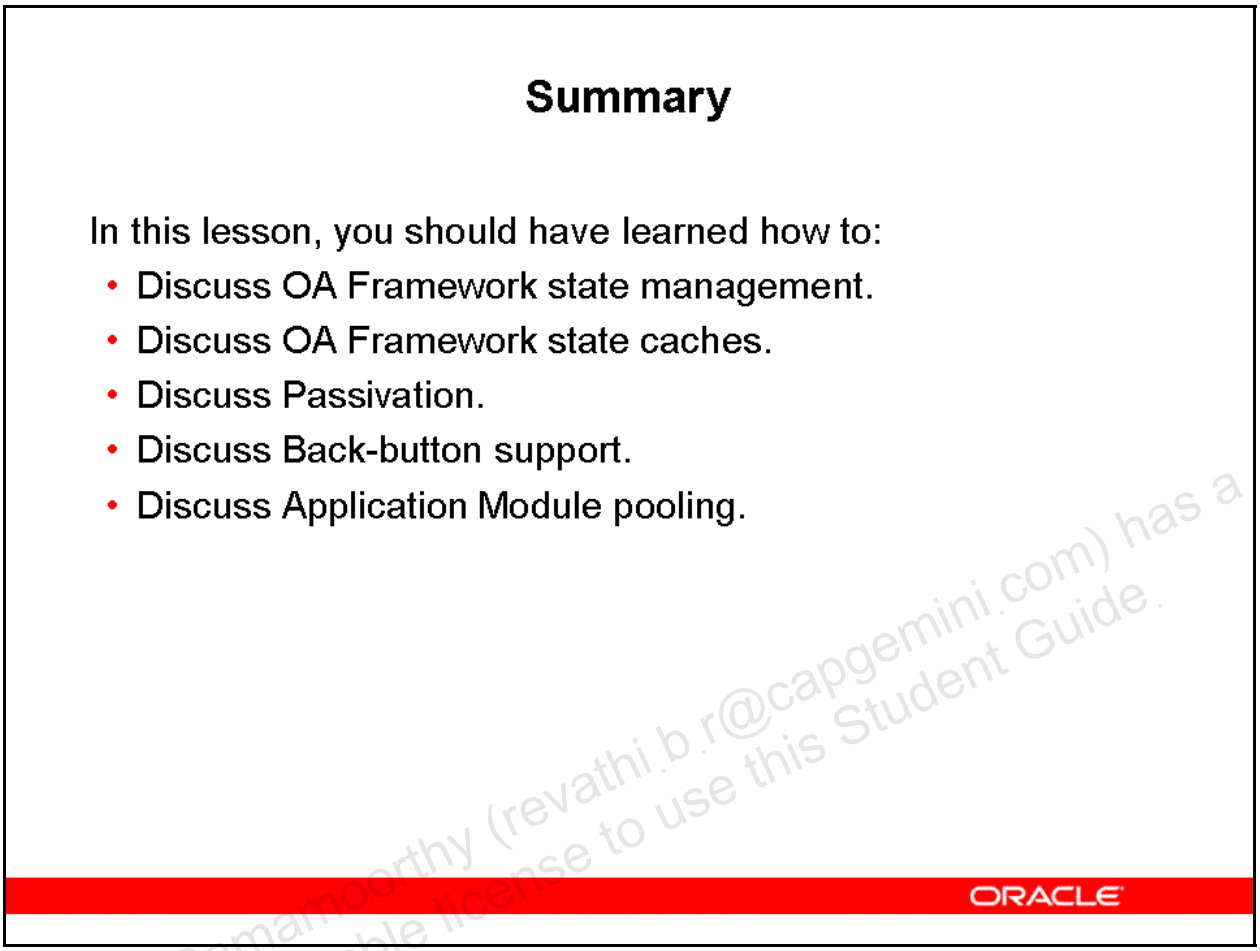
Problem

The OA Framework "Page 1" expects the web bean hierarchy to be in a certain state, and/or it expects transaction, session or BC4J object state values to be present. When navigating with the browser Back button, this state may be lost. If the page doesn't anticipate this, unhandled exceptions may display to the user.

A Back button compliant page would anticipate the potential loss of state behind the scenes, and would either be capable of recreating the state so it can continue functioning normally, or fail gracefully by displaying a user-friendly error message indicating that the page cannot be accessed after navigating with the browser Back button.

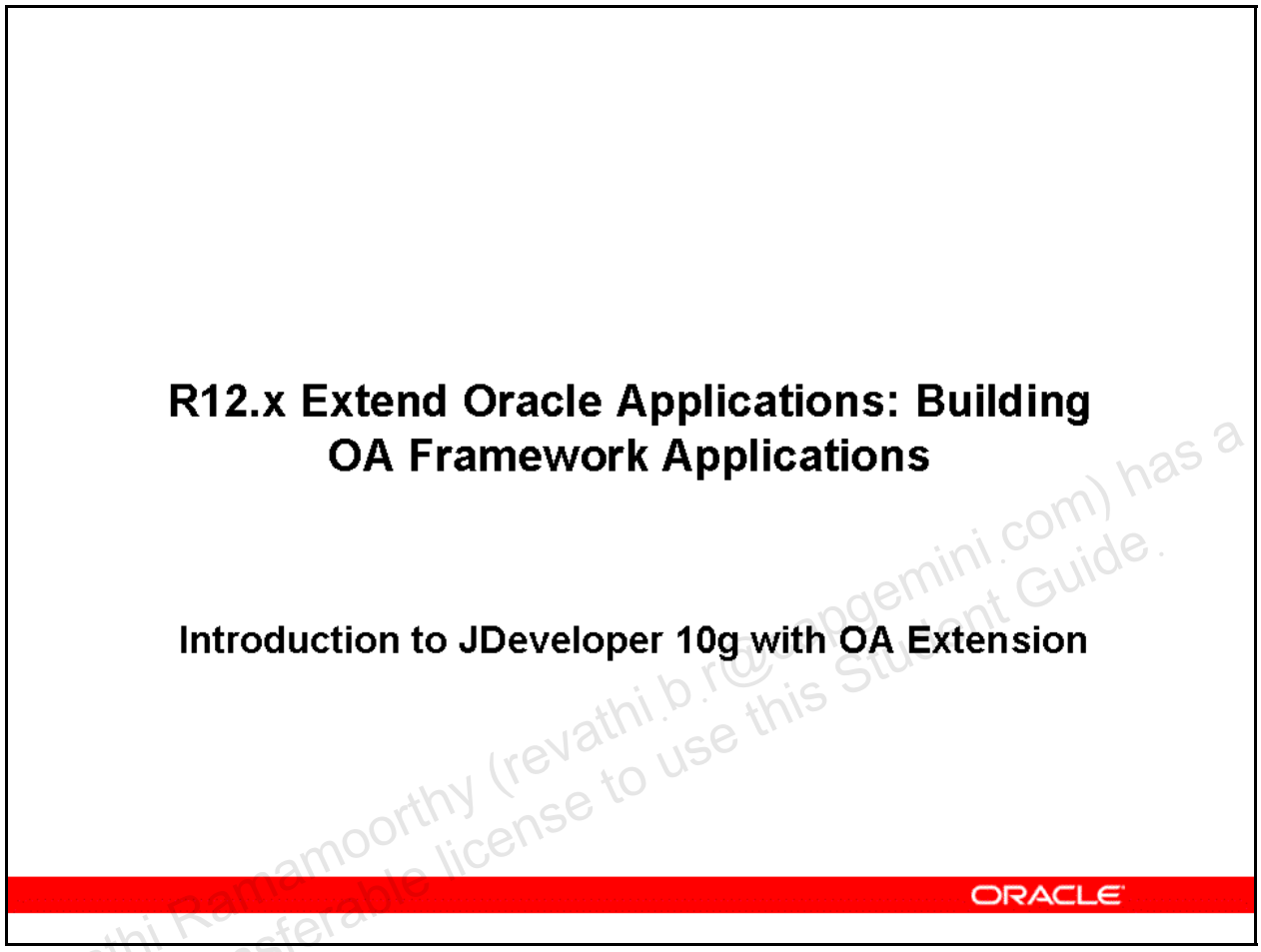




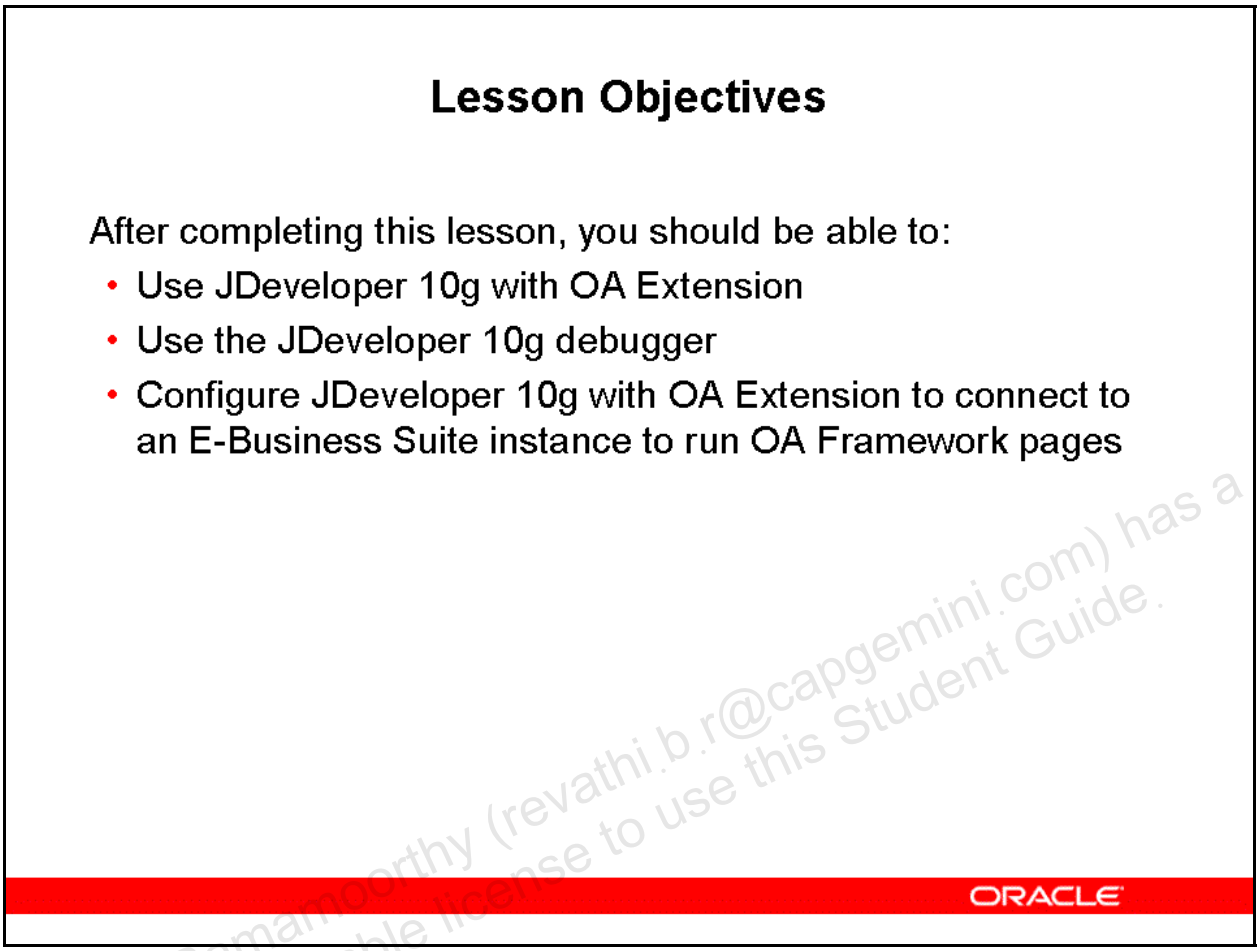


Introduction to JDeveloper 10g with OA Extension

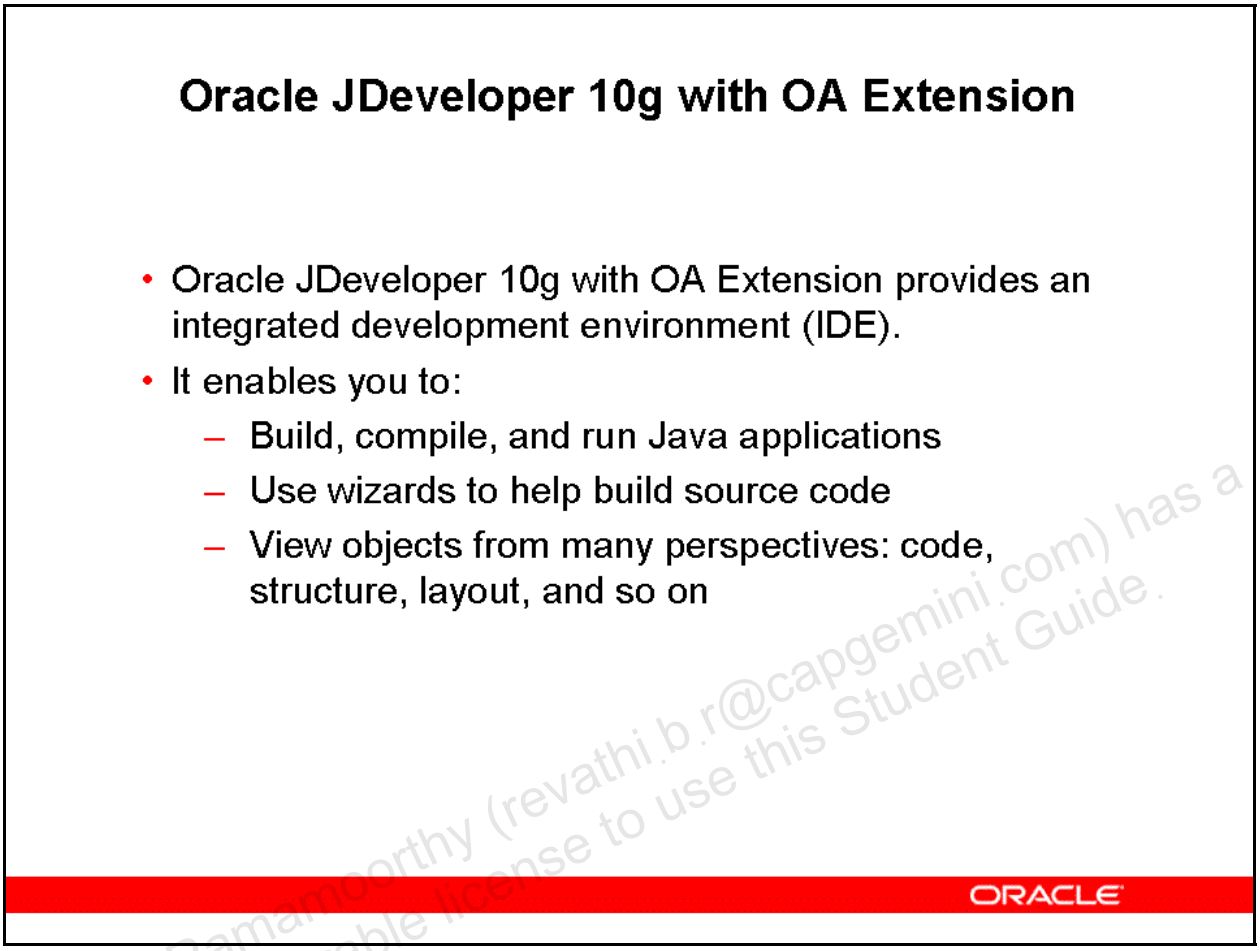
Chapter 8



Lesson Objectives



The debugger section in this course is a very brief overview of what can be debugged in JDeveloper. There are many techniques and methods to debugging and the more advanced topics are beyond the scope of this class. Using the Debugger in the classroom is usually a very network intensive process and your instructor will probably opt to go over the debugging or demonstrate it for you, in lieu of every student using the debugger. If you use the debugger, you will cause performance to suffer for all of your classmates.



Oracle JDeveloper

You can use Oracle JDeveloper to build many different types of Java components. This lesson focuses on using the JDeveloper IDE for building OAF applications. Many of the components that you will create are Wizard driven.

A Wizard is a graphical tool that provides step-by-step guidance through the process of defining a new element in the IDE. Oracle JDeveloper provides many contextual Wizards, including:

Application Wizard: Defines a new application and associated projects

Applet Wizard: Defines a new Java applet and adds it to the specified project

EJB Wizard: Defines a new Enterprise JavaBean (EJB) and adds it to the specified project

JSP Wizard: Defines a new JavaServer Page (JSP) and adds it to the specified project

HTTP Servlet Wizard: Defines a new servlet and adds it to the specified project

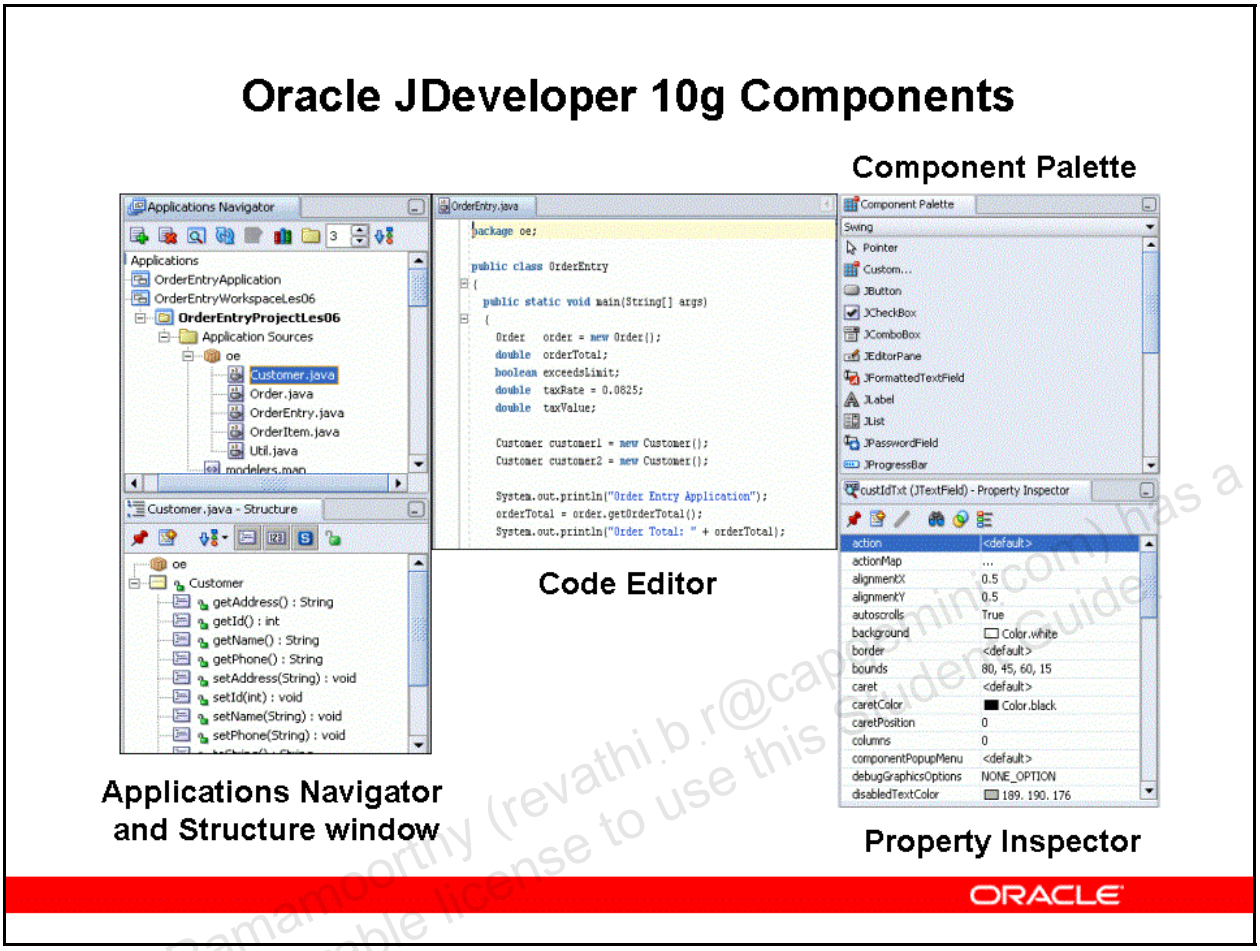
OA Framework Contextual Wizards

Note: Some of the wizards above are not used when building OAF Applications. They are part of JDeveloper.

Oracle JDeveloper helps you specify the following features of your user interface:

- Size and position of controls
- Properties for each control, such as labels, enabled or disabled status, font, etc.
- Event-handler methods

Oracle JDeveloper 10g Components



Oracle JDeveloper 10g Environment

Oracle JDeveloper 10g contains four major user interface components. These components are what you use to edit code, design and manage the user interface, and navigate your program.

Component Palette

The Component Palette displays the components available to build user interfaces, models, navigation diagrams, and so on.

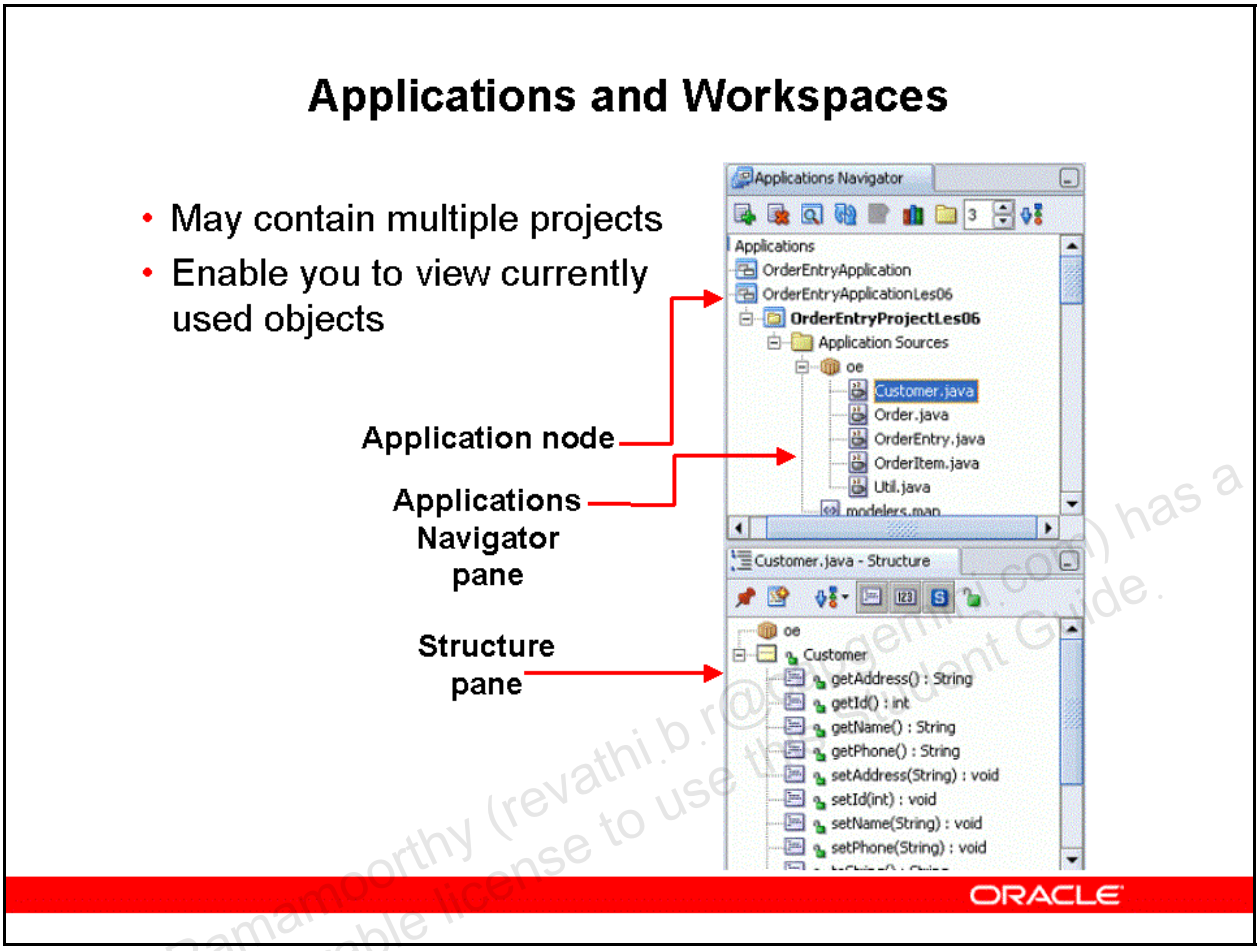
Applications Navigator and Structure window

The Applications Navigator displays a list of files or classes in a project. The files may be Java source files, .class files, graphics files, HTML, XML documents, and so on. The associated Structure window shows the detailed structure of the object selected in the Navigator.

Code Editor

The Code (Design) Editor is where most of the work takes place; this is where you write code and modify user interfaces. You can open the editor by double-clicking the name of the file in the Navigator that you want to edit or view.

Note: The interface components windows are movable and you have control over which ones are displayed.



Application Organization

Oracle JDeveloper 10g with OA Extension uses a well-defined structure to manage Java programming applications. The structure is hierarchical and supports applications, projects, images, .html files, and so on.

Applications

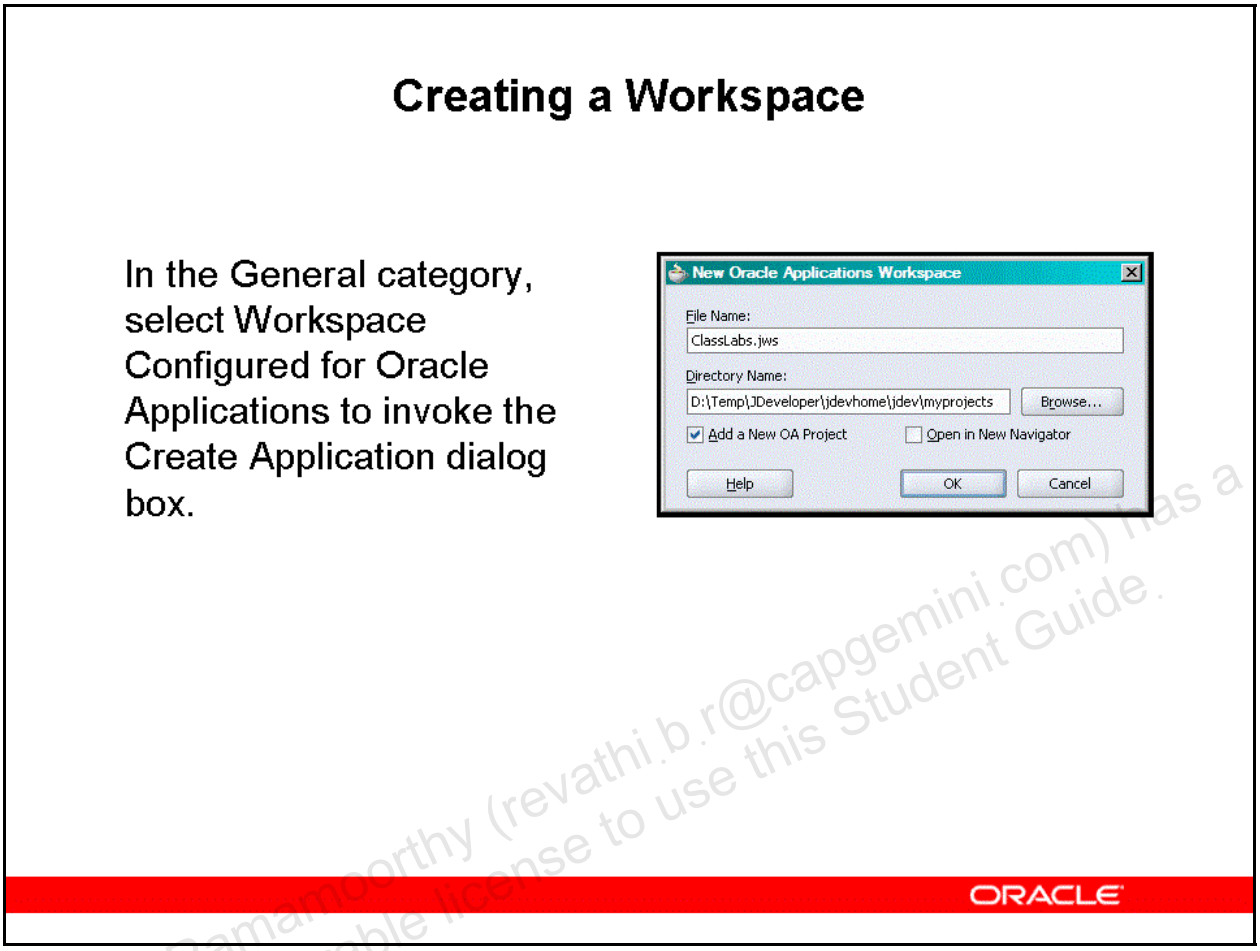
An Application is the highest level in the control structure. It is a view of all the objects you currently need while you are working. An application keeps track of the projects you use and the environment settings while you are developing your application. When you open JDeveloper, the last application used is opened by default so that you can resume your work.

Applications are stored in files with the extension .jws. You do not edit an application file directly. Whenever you save your application, you are prompted to save all the files currently open. To save the open and modified files, click the Save option (or the Save All option) from the File menu.

Note: You can view the content of an application file by using any text editor.

Workspaces

For E-Business Suite development, there is additional information and restrictions. For example, an OA Framework application needs to know the E-Business Suite user, password, application short name, and responsibility key in order to run. Additionally, unlike typical J2EE development, E-Business Suite doesn't run Applications, it runs pages (XML files). So, in JDeveloper, an Application is used as the starting template that is modified to meet E-Business Suite needs, and is then renamed as a Workspace.



Creating a Workspace - .jws files in the file system

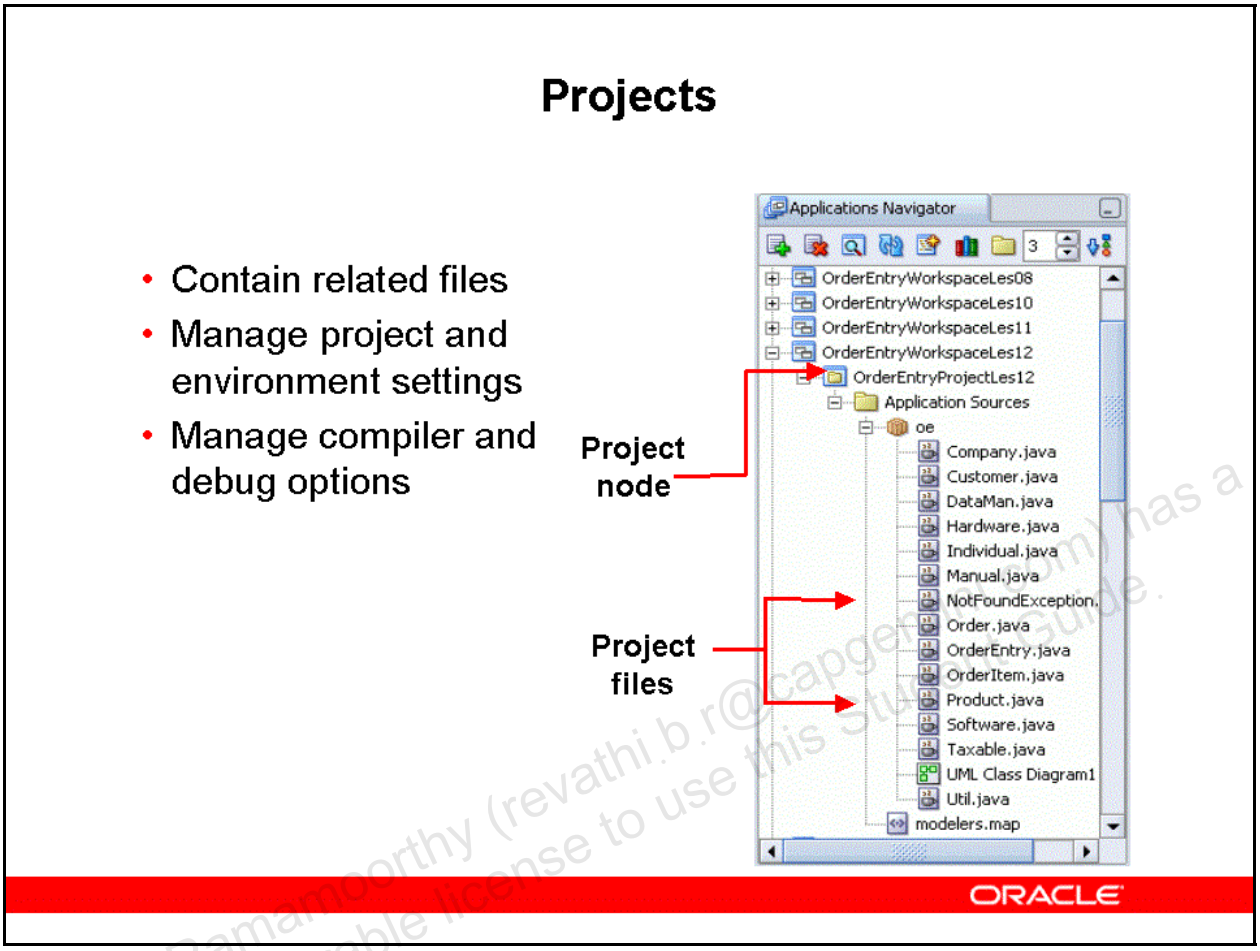
The first step is to create a new workspace that acts as a container for all the files that are going to be part of your project. Workspaces contain projects. When you create a workspace, you will enter a name for the workspace.

When creating a Workspace you will designate the top-level directory for the application files that are going to be part of your project. The top level directory will be the physical location for your files. By default, the location is:

```
JDEV_USER_HOME/myprojects.
```

When you create a Workspace, you immediately can create a new OA Project

Projects



Projects - .jpr files in the file system

JDeveloper projects organize the file elements that are used to create your program. A project file has the file extension .jpr and keeps track of the source files, packages, classes, images, and other elements that may be needed for your program. You can add multiple projects to your application and workspace to easily access, modify, and reuse your source code. You can view the content of a project file by using any text editor.

Projects manage environment variables, such as the source and output paths used for compiling and running your program. Projects also maintain compiler, run-time, and debugging options so that you can customize the behavior of those tools for each project.

In the Navigator pane, projects are displayed as the second level in the hierarchy under the application.

When you select a .java or .html file in the Applications Navigator, the Structure pane displays the elements of the file in a tree format. For example, when you select a .java source file, the classes, interfaces, methods, and variables are displayed.

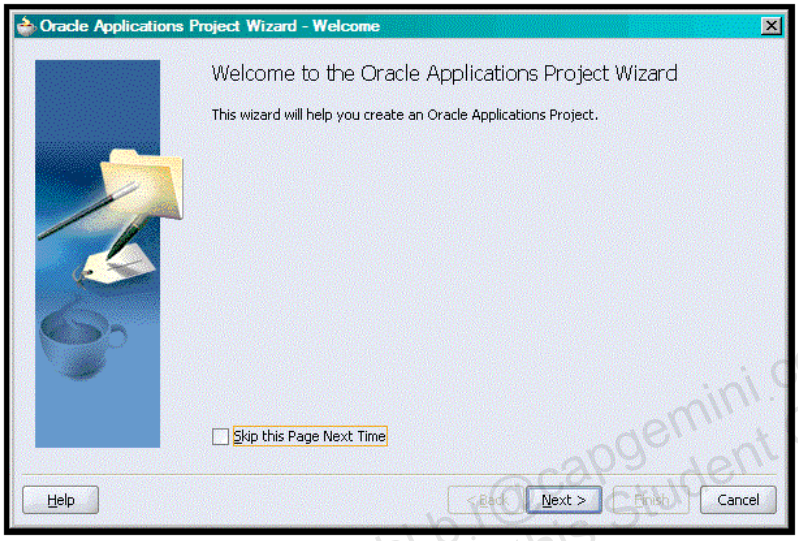
To edit source code, double-click the file in the navigation list to display the contents in the appropriate editor. The Structure pane can be used to quickly locate specific areas of code in your Java source files and to browse the class hierarchy.

When you are working with the visual designer, the Structure pane displays the components of your user interface and their associated event-handling methods in a hierarchical tree format.

Note: Italic style is used to indicate file names that have not yet been saved.

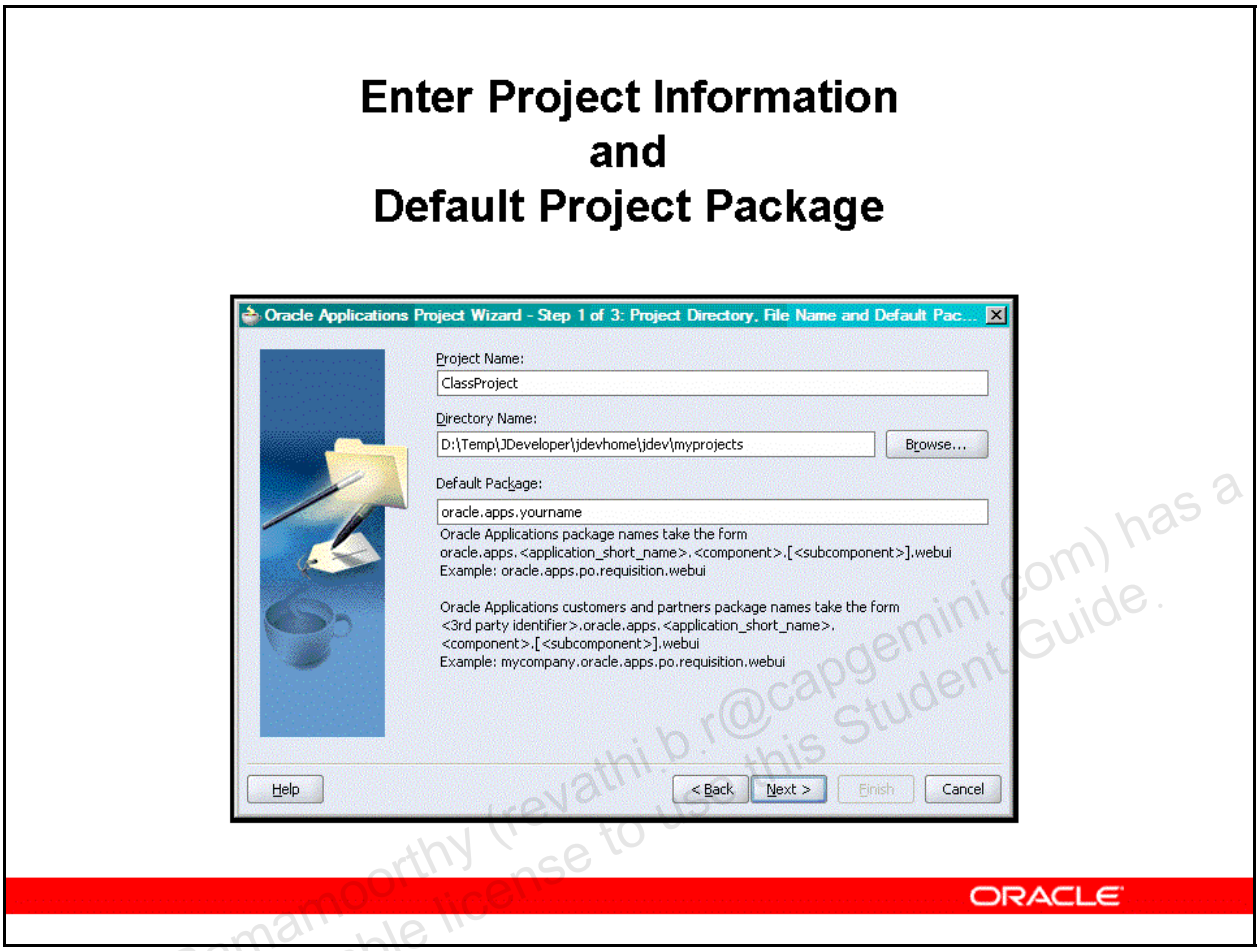
Creating Project Wizard Start Screen

Creating Project Wizard Start Screen



ORACLE

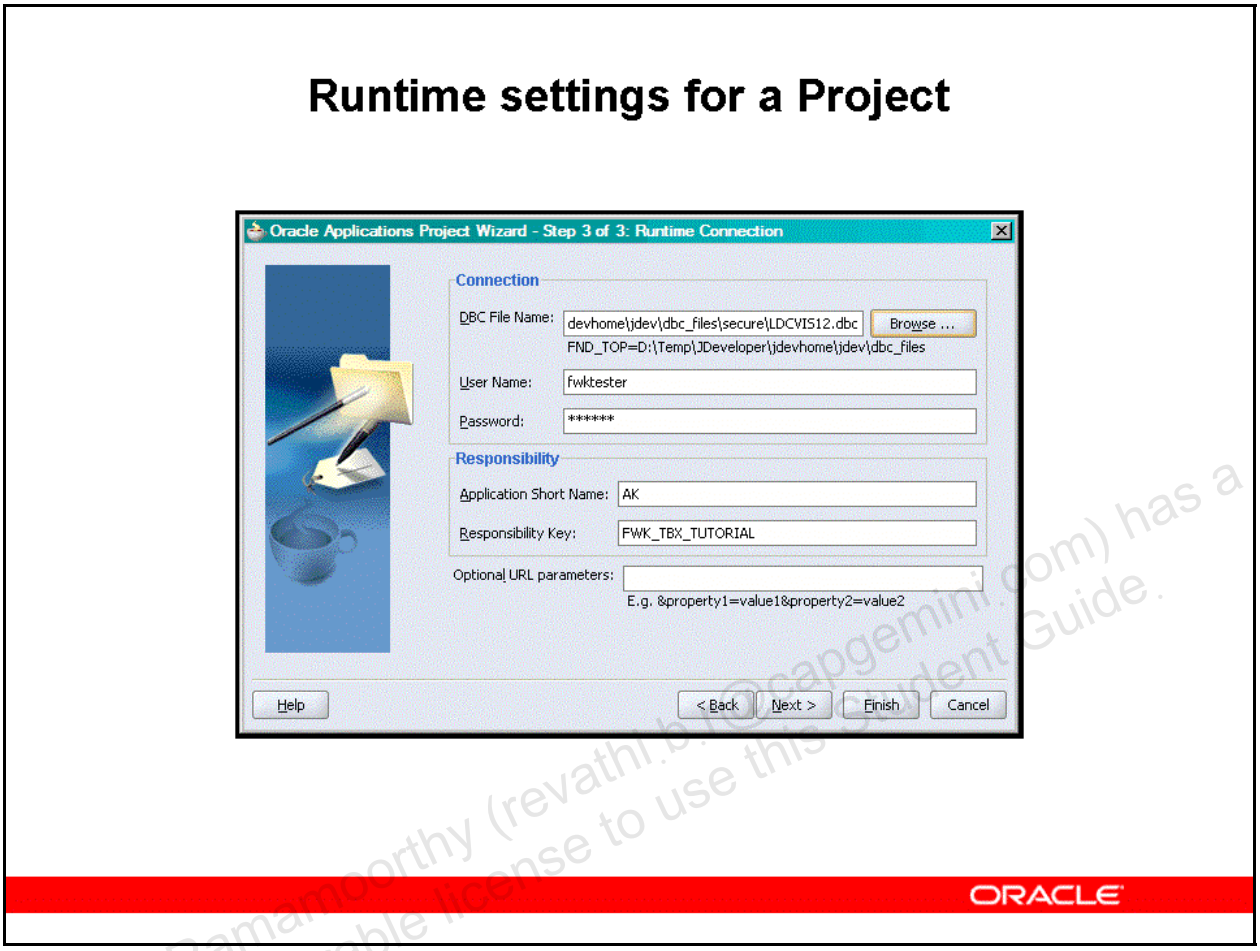
Enter Project Information and Default Project Package



Default Package:

This is simply a default starting point for future files. It is not a limit. You can change the package of any object you create to point to the appropriate path.

Runtime settings for a Project



DBC File Name:

You can either copy it to your own ./secure directory, or you can link to the directory where you instructor has stored the original.

User Name:

Default for this course is fwktester or FWKTESTER.

Password:

Default for this course is fwkdev. or FWKDEV

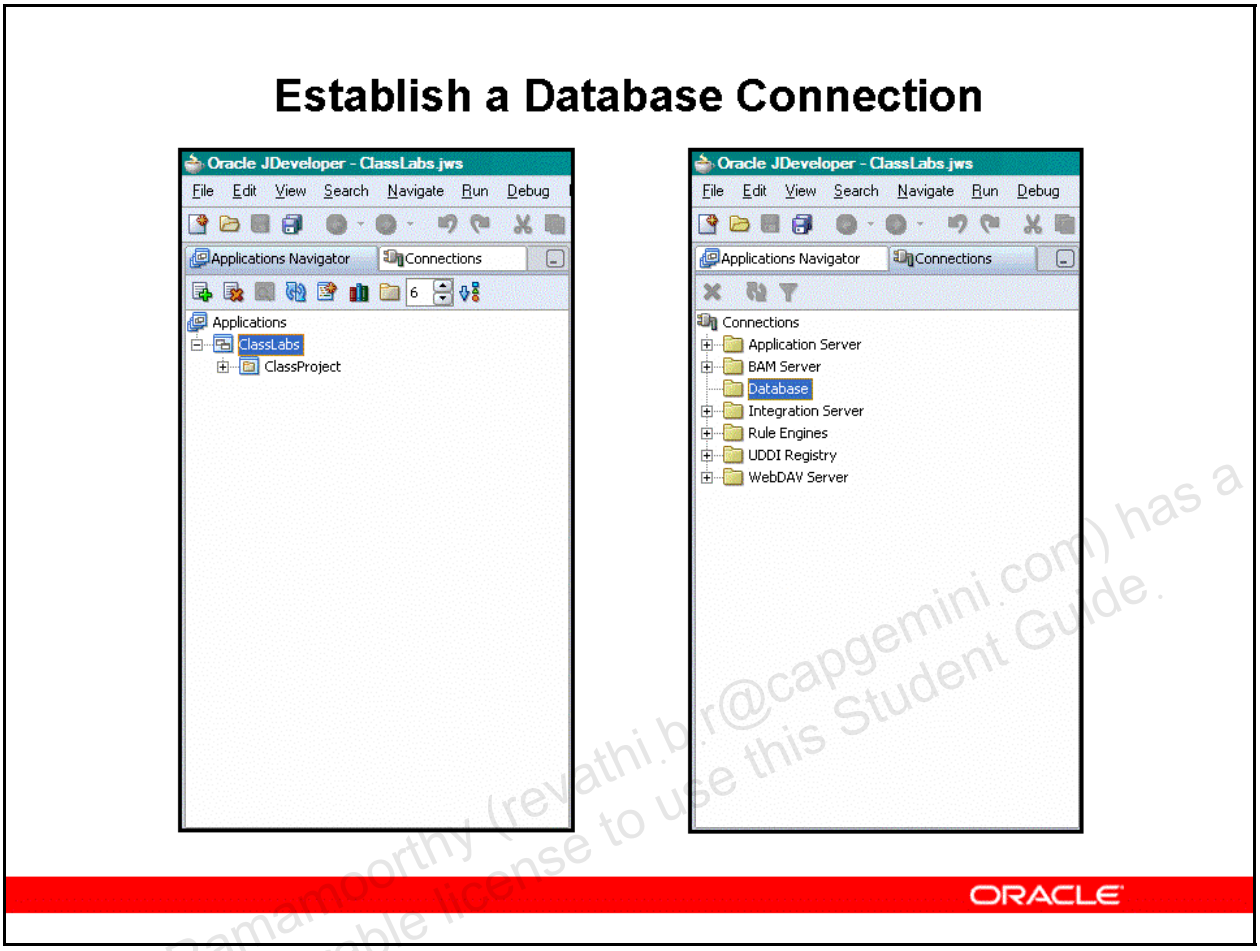
Application Short Name:

Default for this course is AK.

Responsibility Key:

Default for this course is FWK_TBX_TUTORIAL.

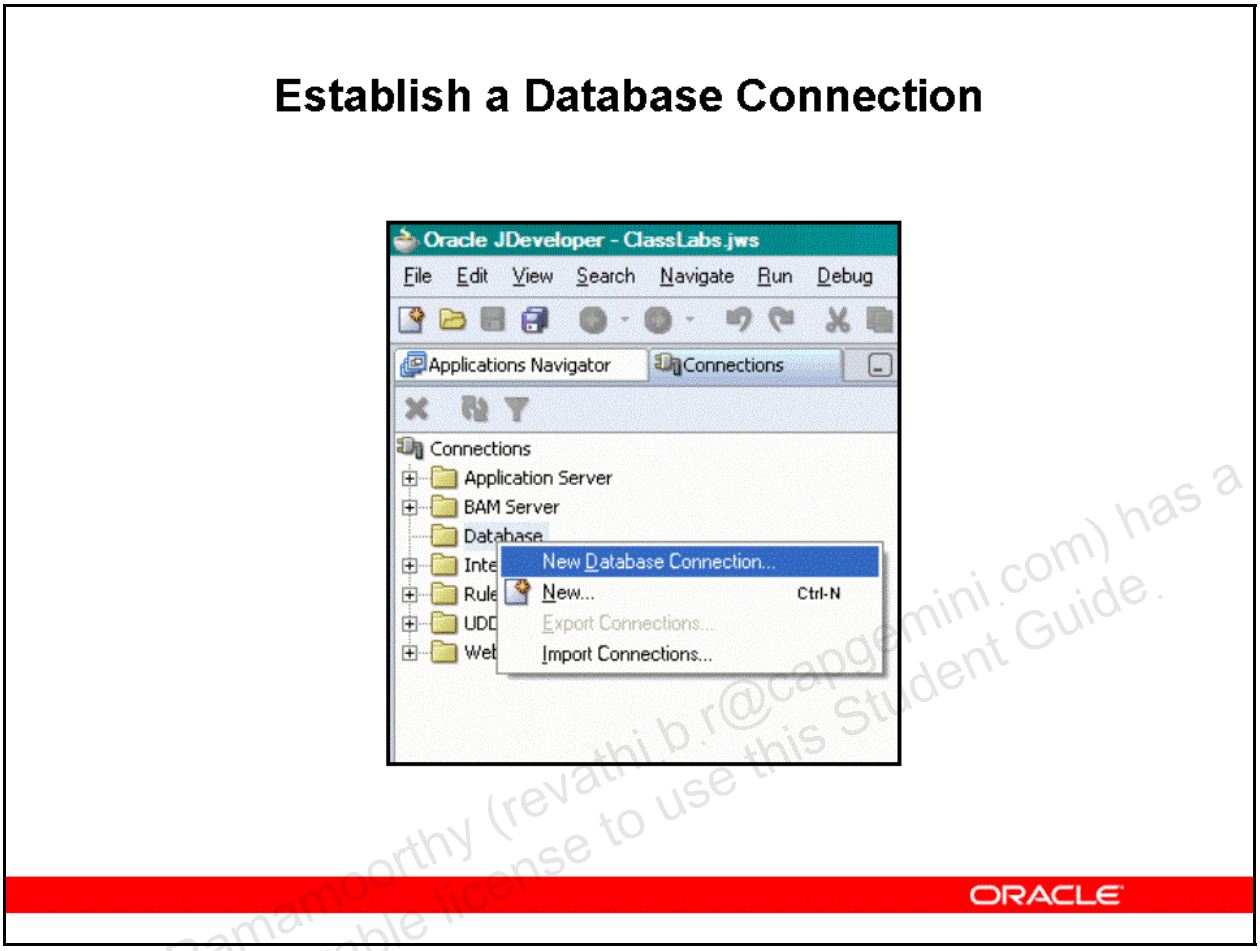
Establish a Database Connection



Once you have a workspace and a project, most developers make sure there is a working connection to their test instance and define the required parameters. It is important to change the seeded values as you work through the setup wizards. The seeded values that appear will not work. The values you see are populated to give you an idea of what you need to know about your instance.

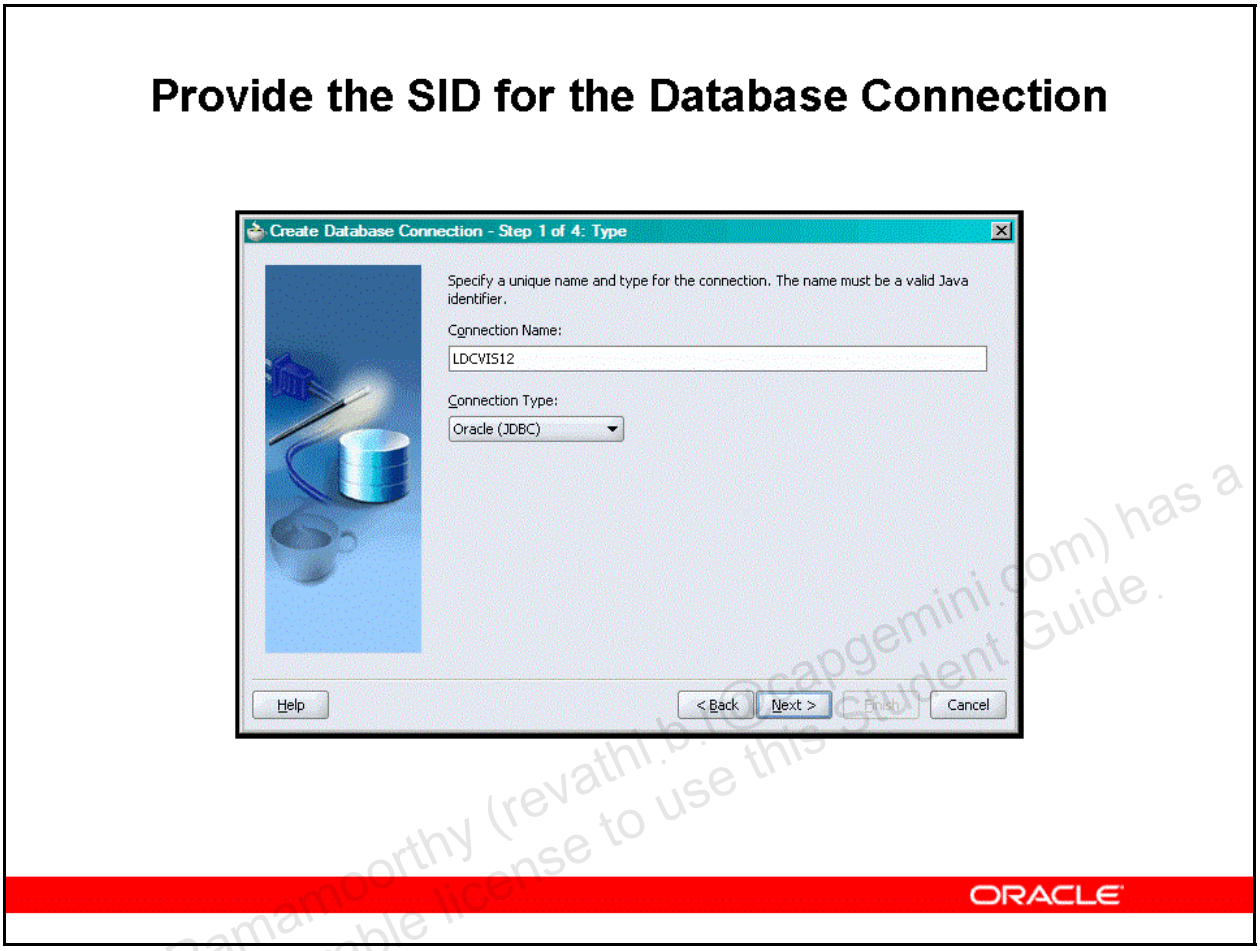
From the Applications Navigator panel, you can click the Connections tab, and choose the Database folder to configure the correct setting based on what your instructor provides.

Establish a Database Connection



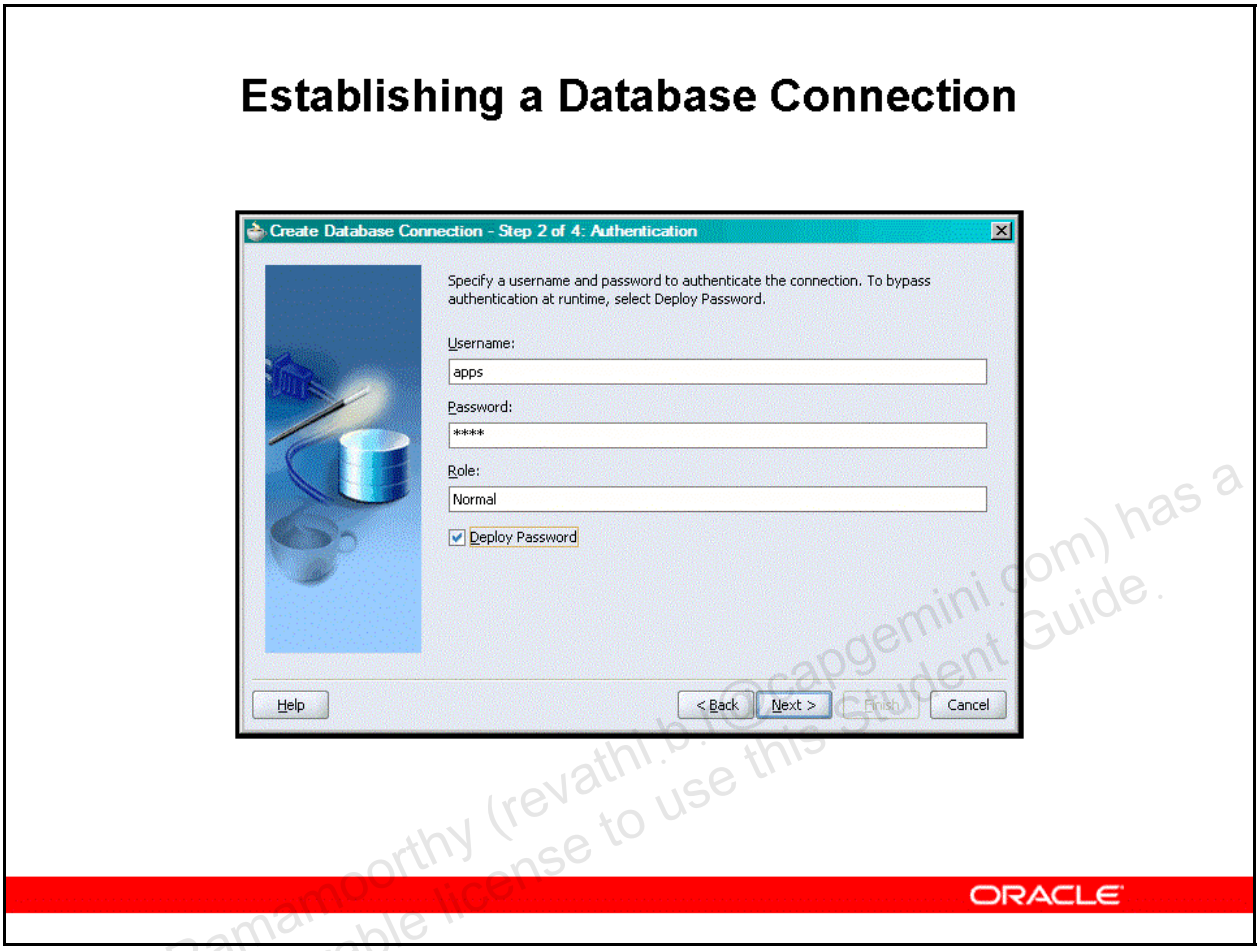
Right-click the Database folder, and choose New Database Connection.

Provide the SID for the Database Connection



Generally, it is recommended to give the connection the same name as the SID of the E-Business Suite database to which you are connecting. The slide above is NOT what you are supposed to fill in.

Establishing a Database Connection



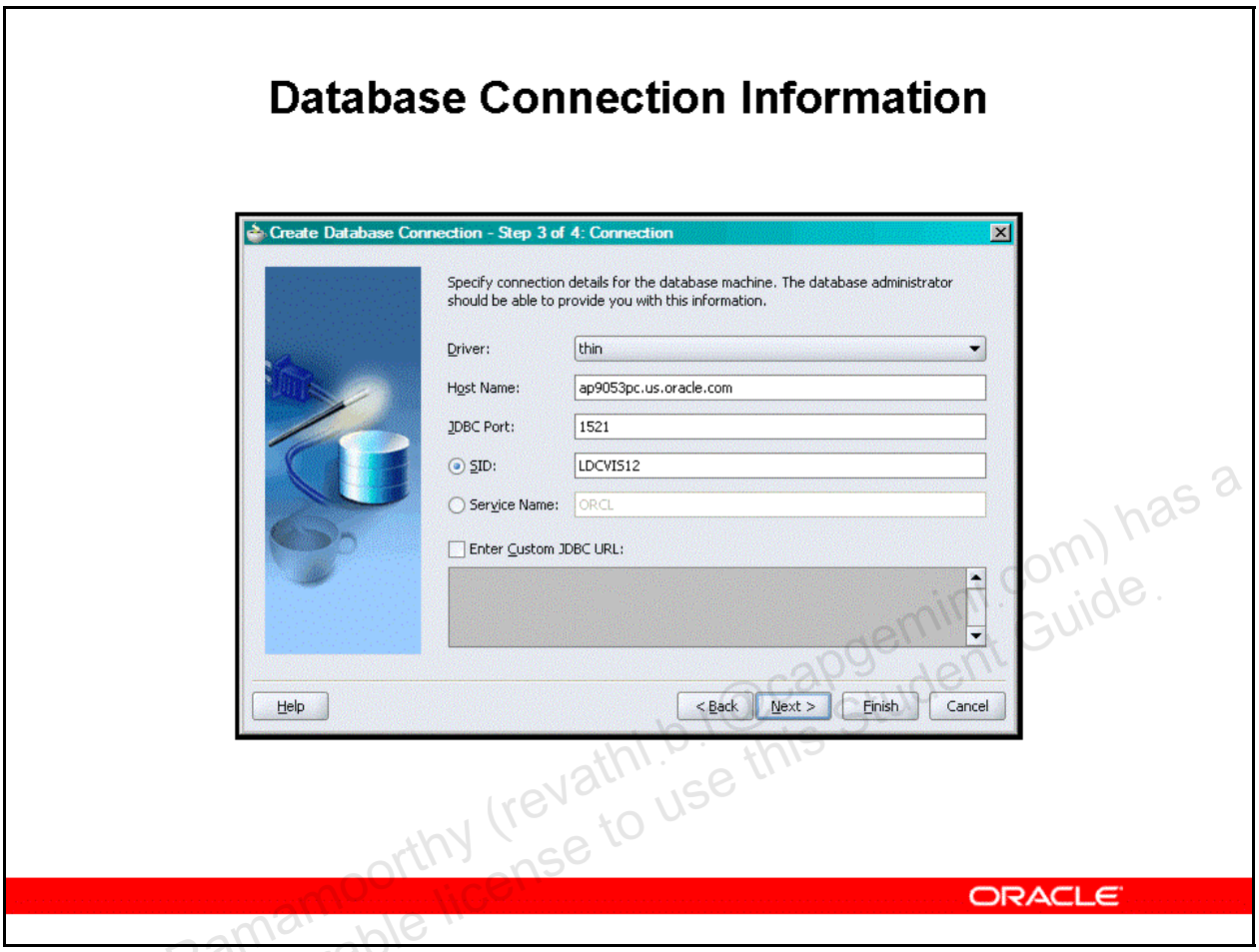
Username:

Default for this course is **apps**.

Password:

Default for this course is **apps**.

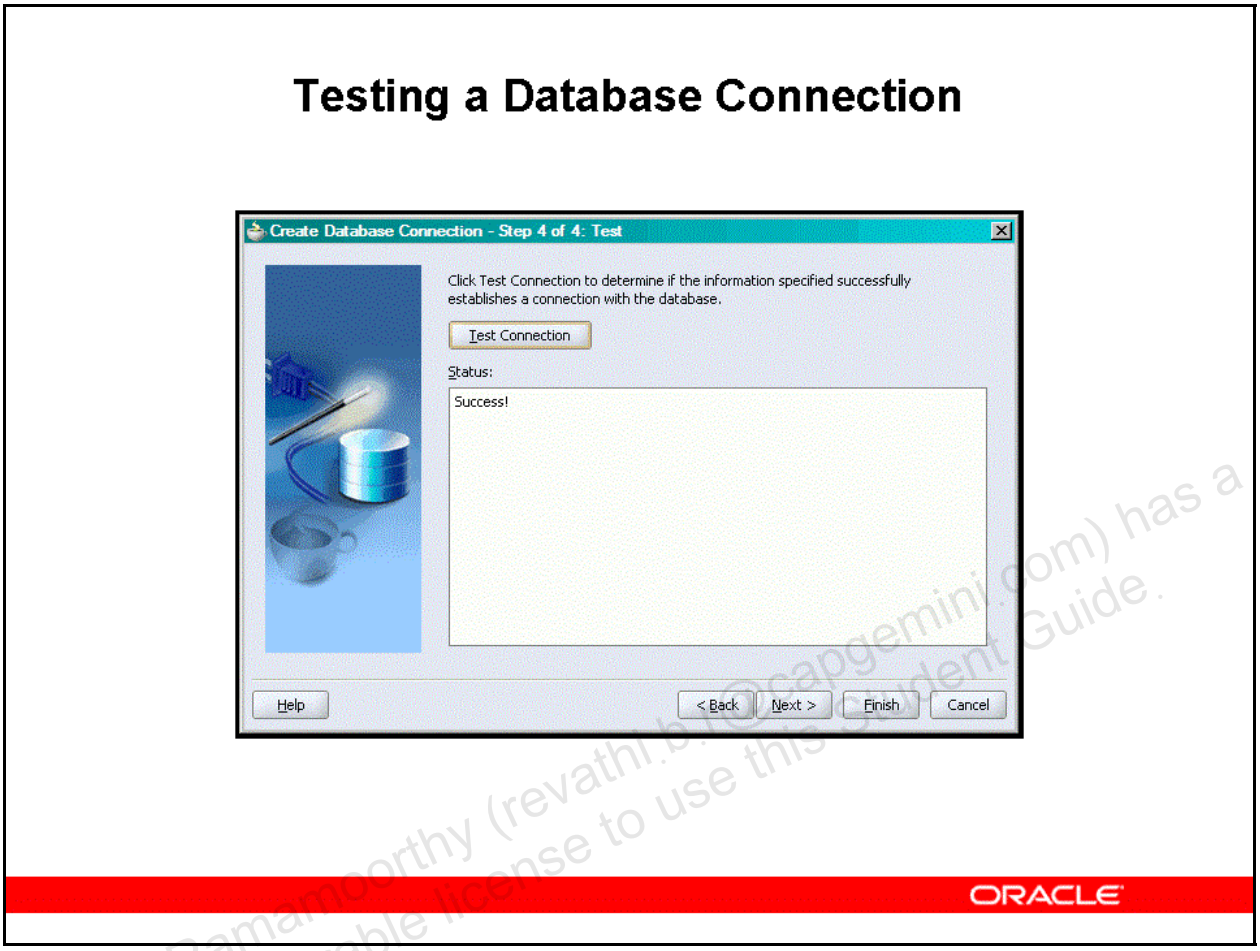
Database Connection Information



If you don't know this information, you can ask your instructor, or you can examine the DBC file they provided. All of the connection information is contained in the DBC.

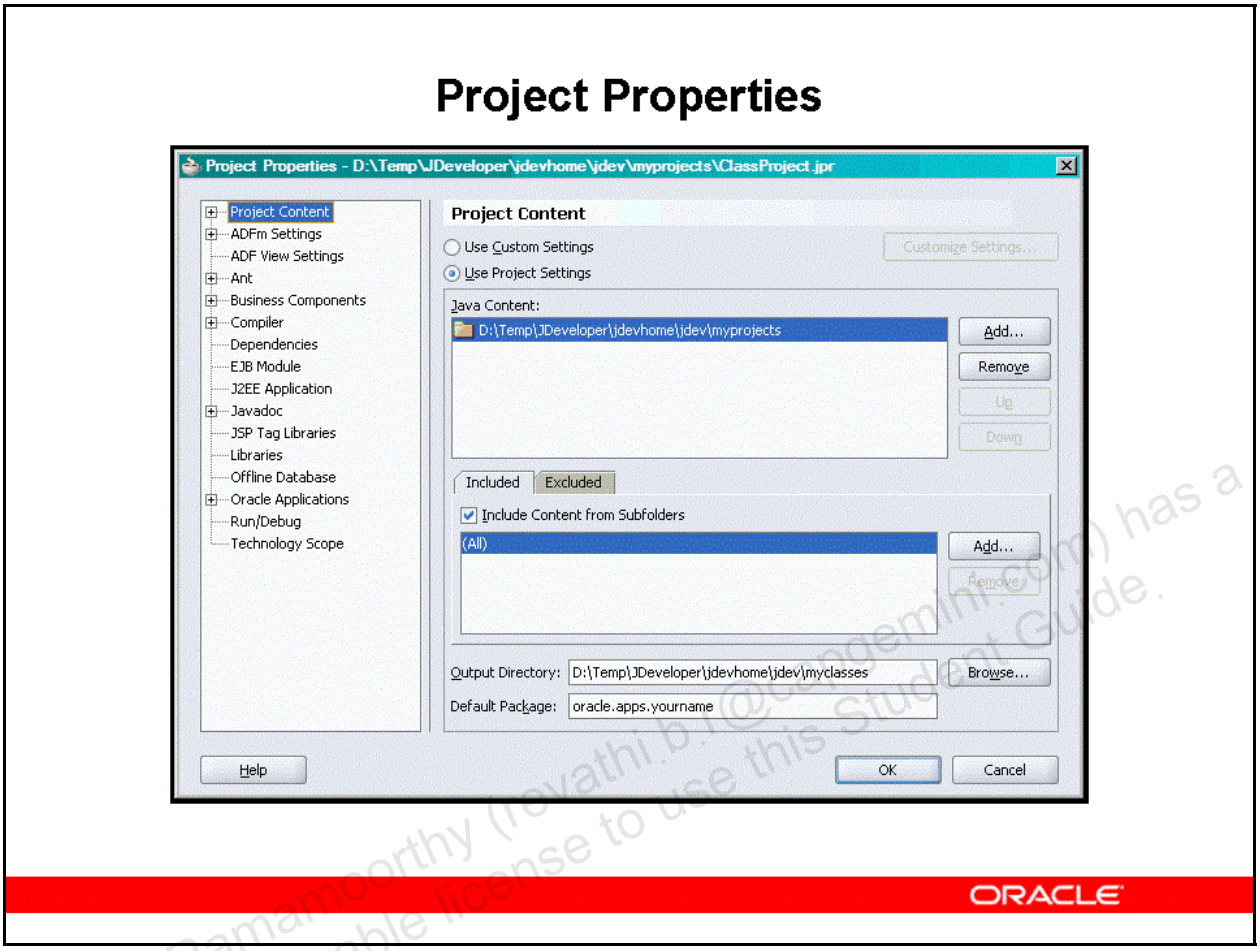
Again, the slide above is NOT what you are supposed to fill in.

Testing a Database Connection



Always test the connection.

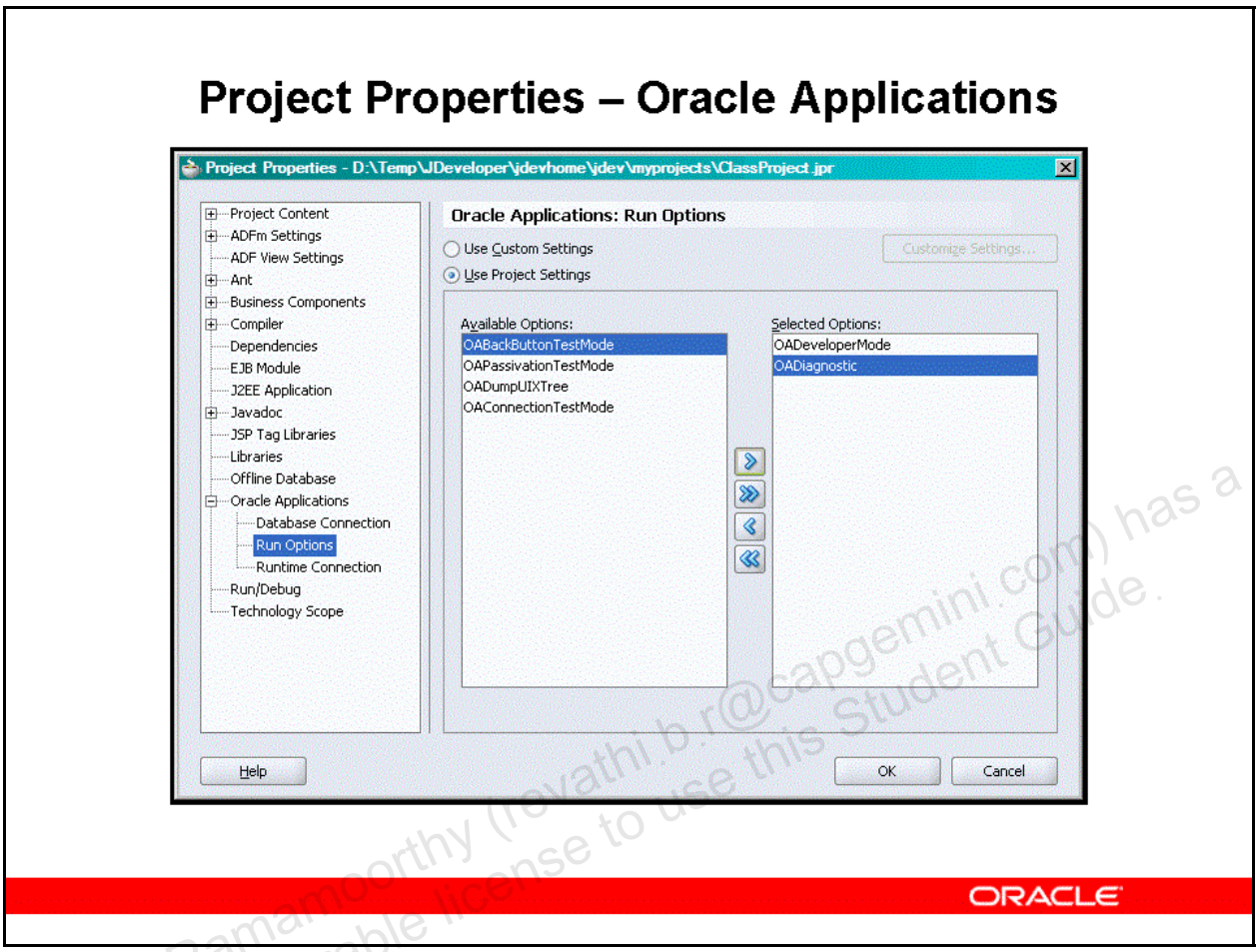
Project Properties



Project Properties: Specifying Project Details

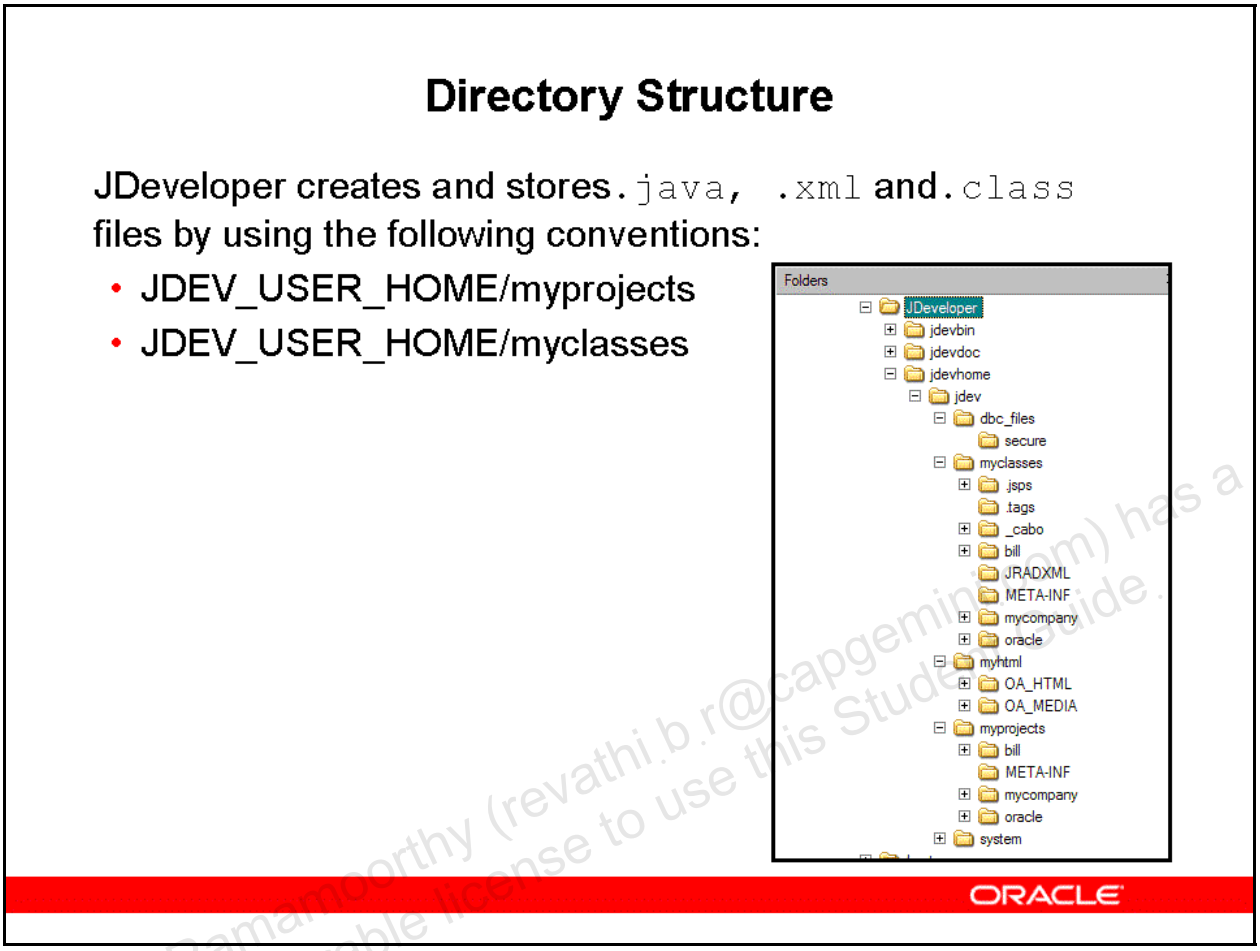
Once the database connection is working, most developers go ahead and set up other parameters as part of their project and workspace. If you need to change any of the connection information, or any other project setting, simply double-click the project in the Applications Navigator panel to open the Project Properties window.

Project Properties – Oracle Applications



In order to see complete diagnostic errors and stack dumps, we will add OADiagnostic to the Run Options of every project.

Directory Structure

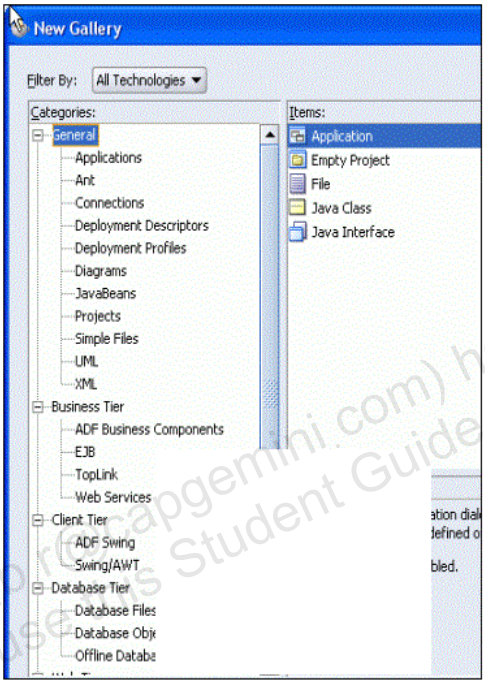


When JDeveloper was unzipped, it created a file structure on the local machine.

`myprojects`:	Holds .java and .xml files
`myclasses`:	Holds .class and .xml files
`myhtml`:	Holds deployed objects, like shared images, and .jsps
`system`:	Holds JDeveloper configuration information
`dbc_files/secure`:	Holds DBC files

Creating JDeveloper Items

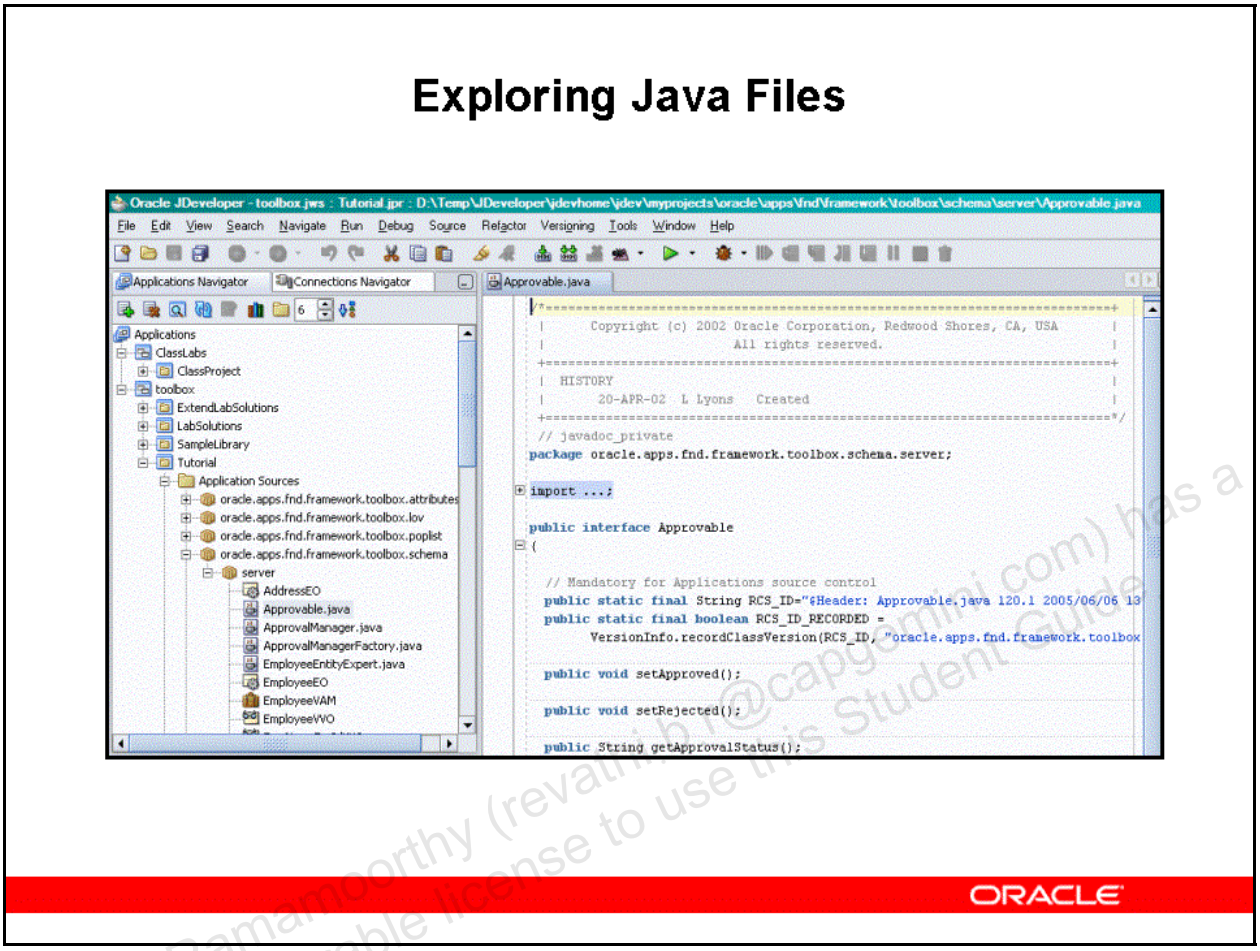
- JDeveloper items are invoked by selecting File > New.
- They are categorized by type:
  - General
  - Business Tier
  - Client Tier
  - Database Tier
  - Integration Tier
  - Web Tier
- Create any JDeveloper element.



Creating JDeveloper Items

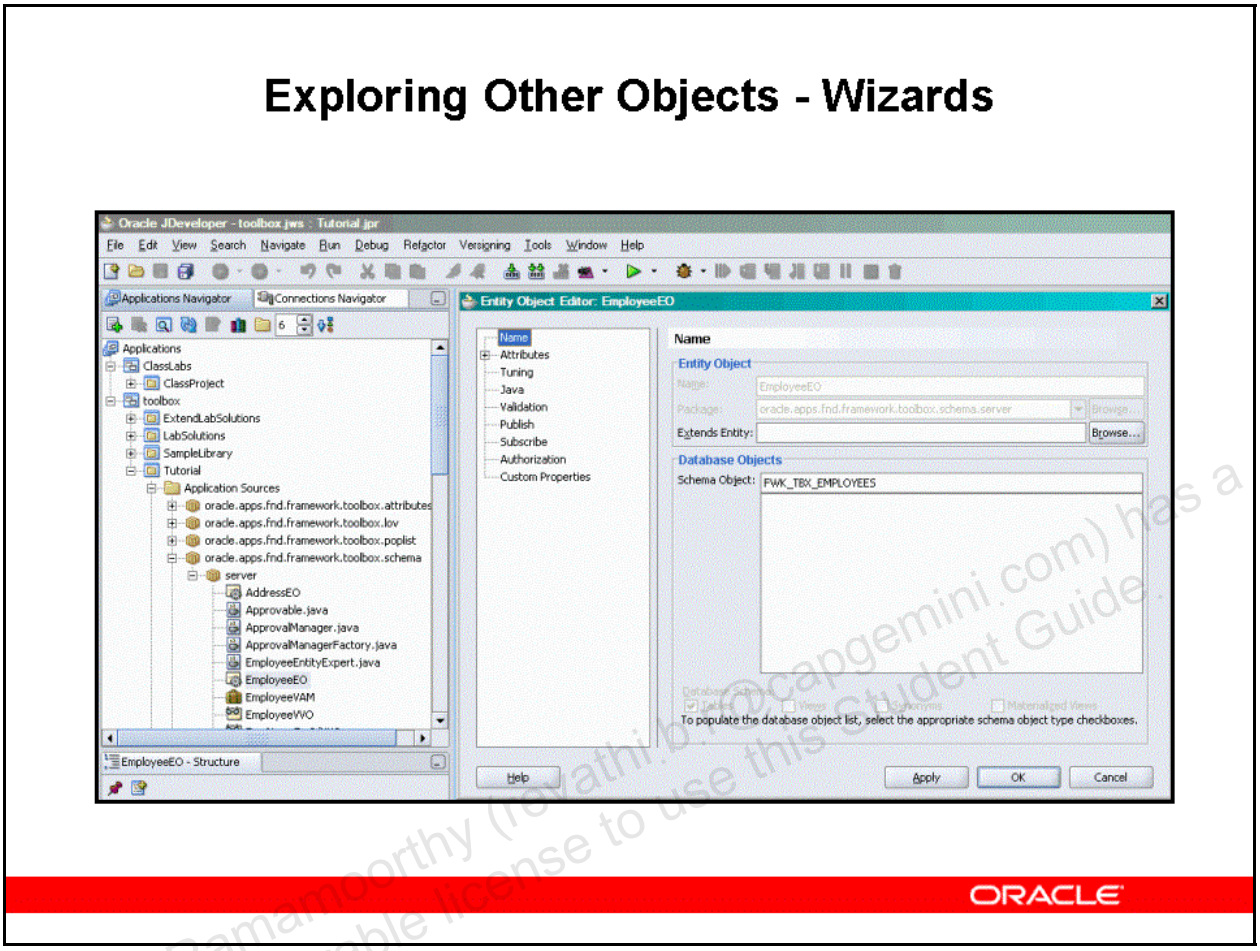
Once past JDeveloper set up, you can start to create objects. You can create any JDeveloper item from the New Gallery window. You can select new and work your way down the categories to create what you would like to create or use a contextual based creation mechanism. The contextual mechanism for creating an item is based on what was highlighted when you clicked the right mouse button or selected new. You must have the correct element selected in the category column to contextually create an appropriate item. The functionality to contextually create something is based on the OAF extension. Use the Filter By drop-down list to view specific types of elements.

Exploring Java Files



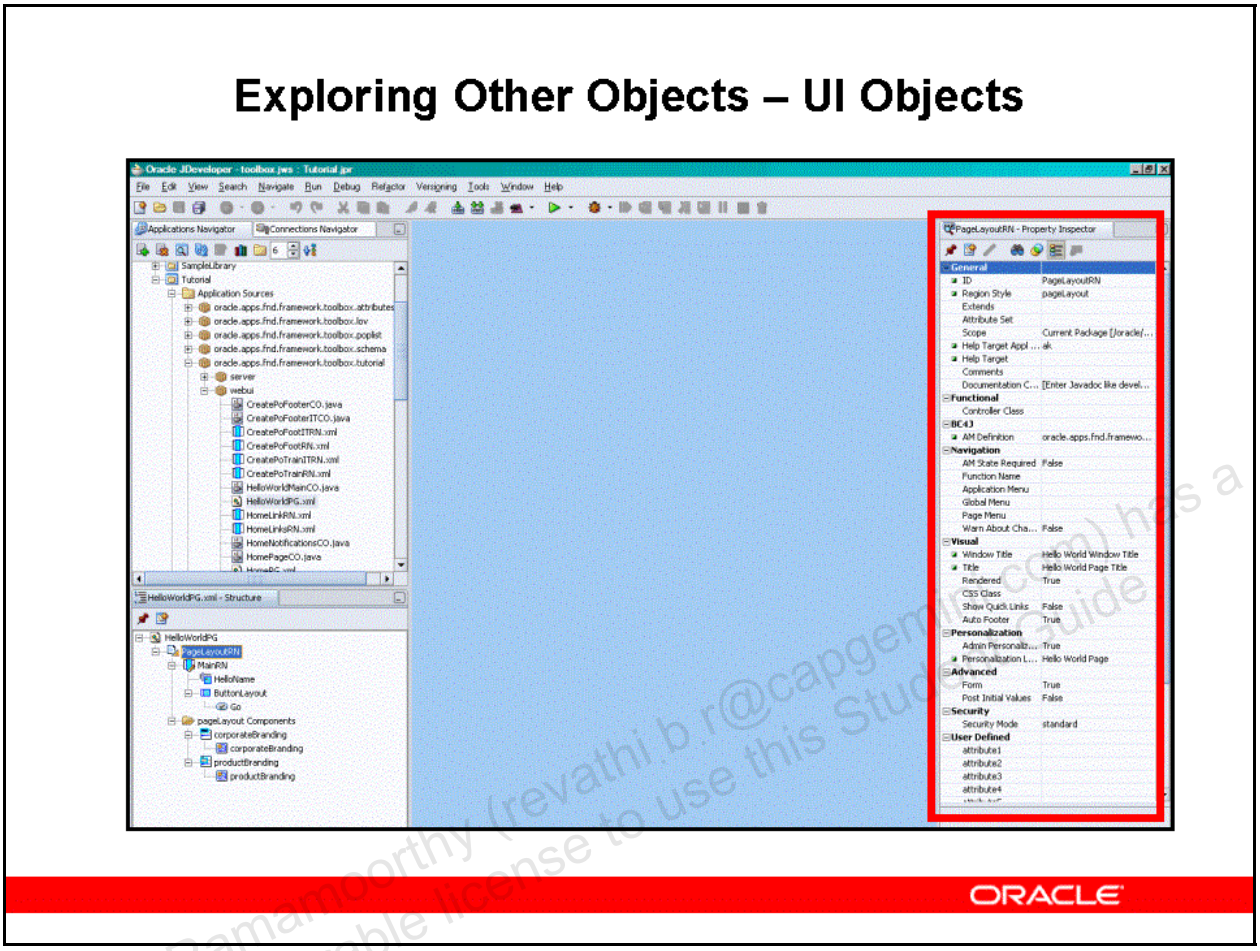
You can edit the Java files that are part of your application.

Exploring Other Objects - Wizards

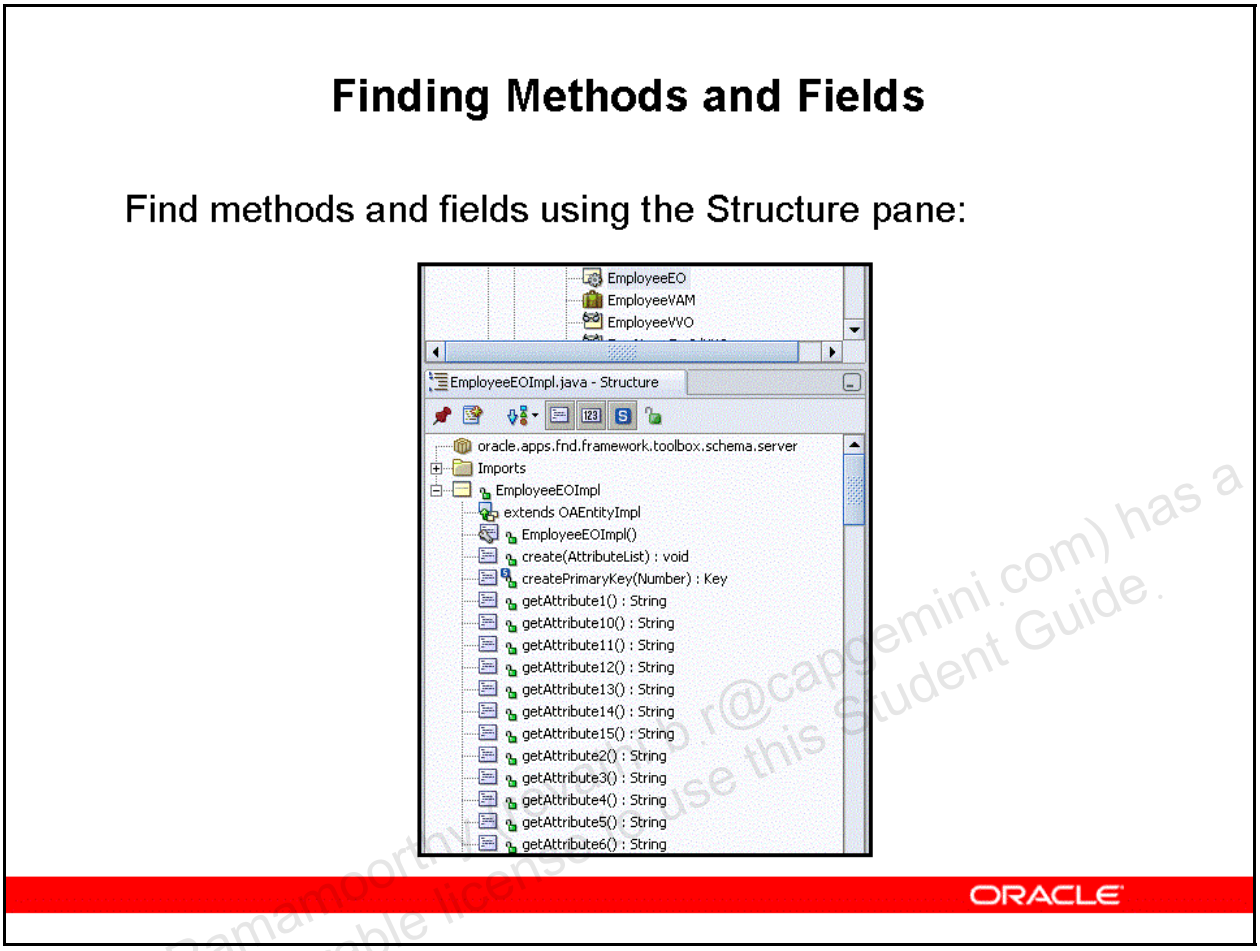


Some objects, notably any BC4J objects (AM, EO, VO, AO, VL), have Wizards associated with them. When you double-click the object, it opens the object wizard by default.

Exploring Other Objects – UI Objects



UI (View-layer) objects, like pages (PG) and regions (RN), aren't written in Java and they don't have object wizards. These objects are edited using the Property Inspector. The UI objects that you create are actually XML files that are used declaratively to render the pages. You can't edit the XML files for your pages from within JDeveloper. There is an enforced hierarchy of the page structure via the OAF framework.



Finding Methods and Fields

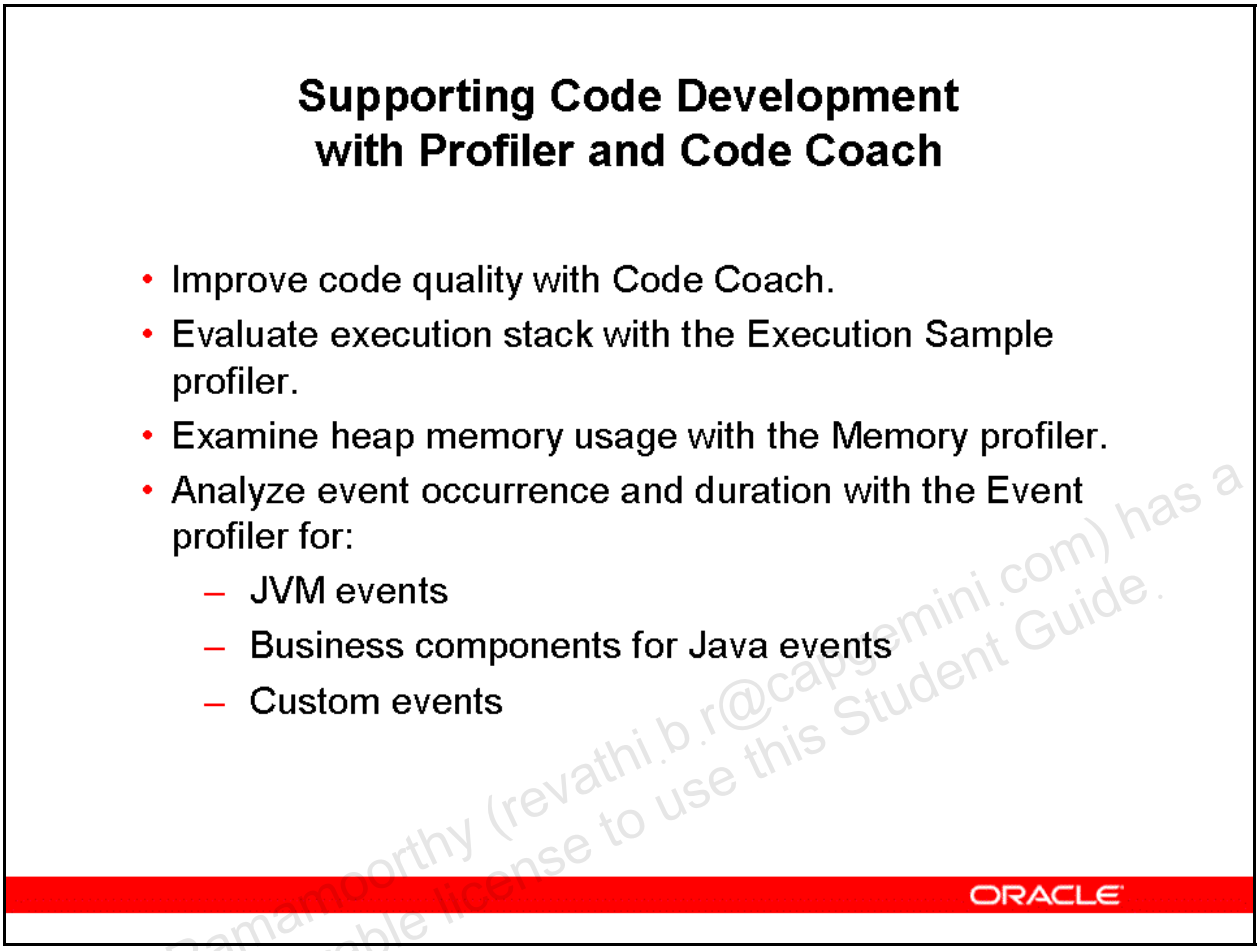
As projects evolve, classes can become quite large, containing several methods and fields. To help you find your way around complex classes, JDeveloper provides the Structure pane, which is the bottom pane in the Applications Navigator.

The Structure pane lists all the methods and fields for the currently selected class. If you double-click an item in the Structure pane, JDeveloper takes you to the definition of that item in the source code, displaying and highlighting it in the Code Editor. For example, if you double-click a method in the Structure pane, the start of the method you clicked is highlighted in the Code Editor.

You can also search the Navigator and Structure pane components for strings by using a [letter]. The search is a hierarchical search based on the first letter of each component. As you enter subsequent letters, the structure list highlights the first component that begins with that set of letters. If there is more than one occurrence, use the up and down arrow keys to scroll through the result set.

Note: JDeveloper 10g with OA Extension provides some new features to facilitate navigation through your Java code. One of these is navigation between members, which allows you to quickly navigate between fields and methods in the Code Editor

by using the Previous Member ([Alt] + [Up]) and Next Member ([Alt] + [Down]) accelerators.



Supporting Code Development with Profiler and Code Coach

Code Coach

Code Coach creates more efficient Java programs by helping you write higher-quality, better-performing code. You run Code Coach on a class to obtain advice on how to make your code better.

Profilers

Profilers gather statistics on your program, enabling you to more easily diagnose performance issues. With profilers, you can examine and analyze your data.

Code Editor

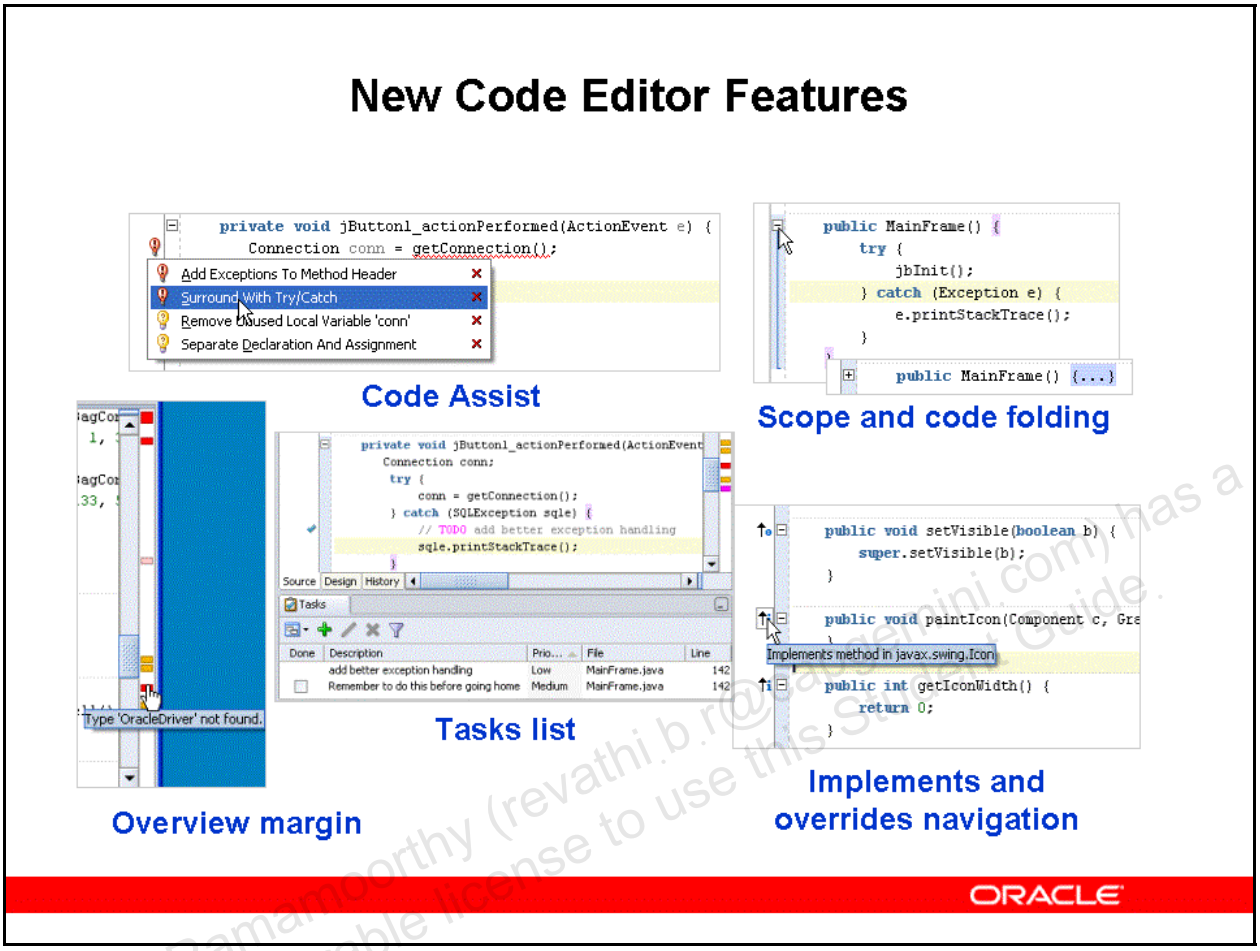
When you pause momentarily while entering code in the Code Editor, JDeveloper automatically scans your code to look for syntax errors. If there are any, you see them represented in the Structure pane or in the Log window.

Code Insight

If you pause after entering a "." (period), JDeveloper invokes Code Insight. Code Insight displays valid methods and members from which you can select. JDeveloper

also provides Code Insight for parameters when you pause after entering a left parenthesis.

New Code Editor Features



New Code Editor Features

Code Assist

Code Assist is a new feature in JDeveloper 10g with OA Extension. It deals with adherence to coding standards rather than syntactical correctness.

It examines your code in the Java Source Editor and offers suggestions to fix coding problems or breaches of coding standards. A light bulb icon appears in the margin beside a line where JDeveloper has a suggestion for a code change. You click the icon to display the suggestion. You can accept the suggestion by clicking it and amending the code; or you can reject the suggestion and suppress the light bulb by closing the suggestion. In many cases, if you accept the suggestion, JDeveloper makes the appropriate code modifications for you automatically.

You can choose which rules you want Code Assist to use, or you can disable the feature altogether by selecting Tools > Preferences > Audit: Profiles.

Code Folding

Code folding allows you to shrink sections of code, making large programs more readable and more manageable. You use the + and – buttons in the blue vertical bar on the left of the code to expand or contract classes, respectively.

Tasks List

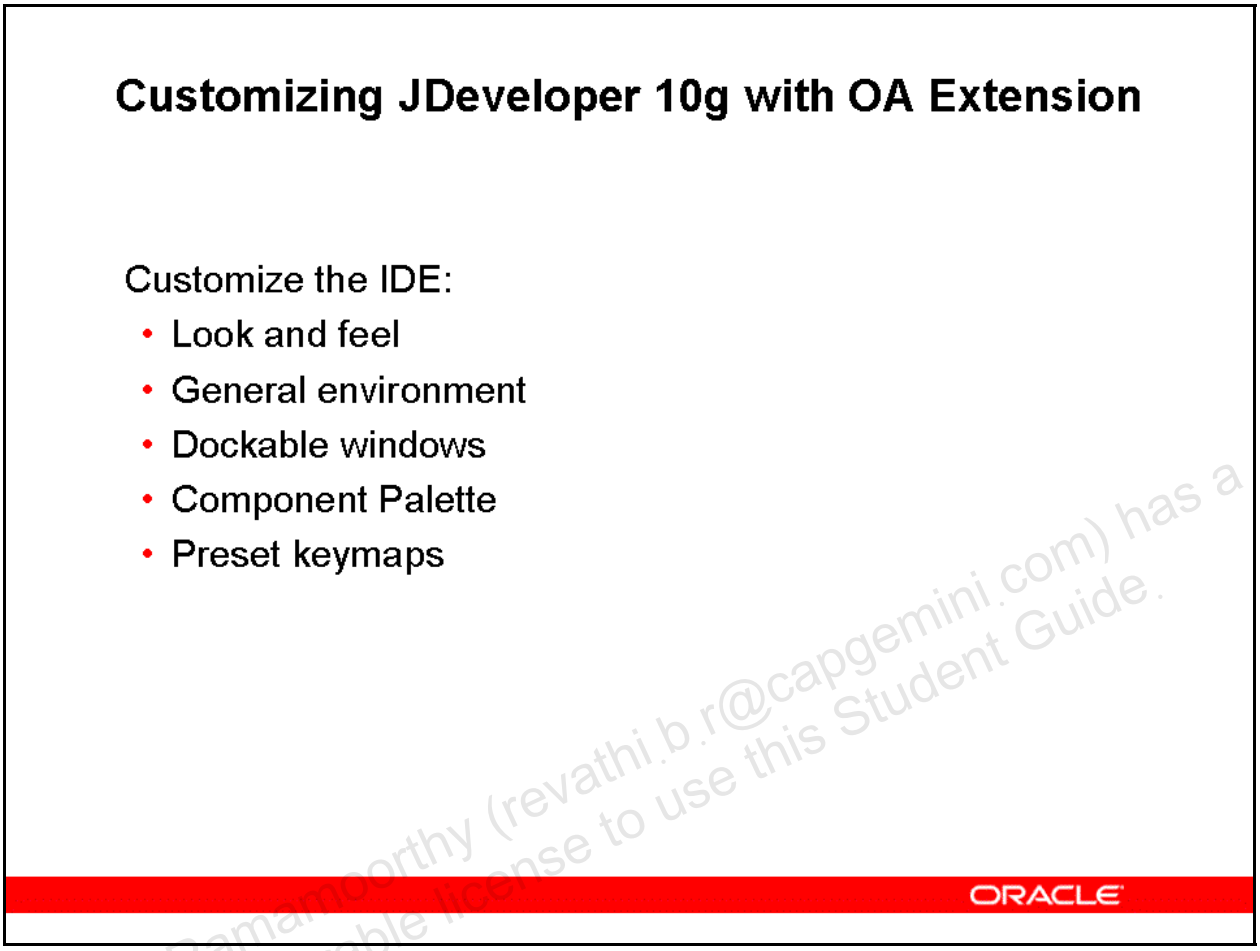
By using the Tasks list, you can create and keep track of tasks. The tasks can be of any nature, but the feature has been provided primarily for tasks connected with the process of software development. For this reason, a direct link has been provided between the Code Editor and the Tasks list: When writing code, a task is created whenever you enter
//TODO. A check mark is displayed in the margin to the left of the line, indicating the presence of a task.

Overview Margin

The Overview margin is displayed vertically on the right side of the Code Editor. If there are no problems with the code, a small green marker appears at the top of the margin. An orange marker indicates a warning, and a red marker denotes an error. When you position the cursor over an orange or red marker, you see a brief displayed description of the error or warning.

Overridden and Implemented Method Definitions

When working in the Java Source Editor, you can identify methods that override superclass definitions or implement interface declarations. Overriding definitions are marked with an upward-pointing arrow and the letter o in the left margin; clicking this letter takes you to the overridden definition. Similarly, an upward-pointing arrow and the letter i in the margin identifies a method that implements an interface; clicking the letter takes you to the implemented method declaration.



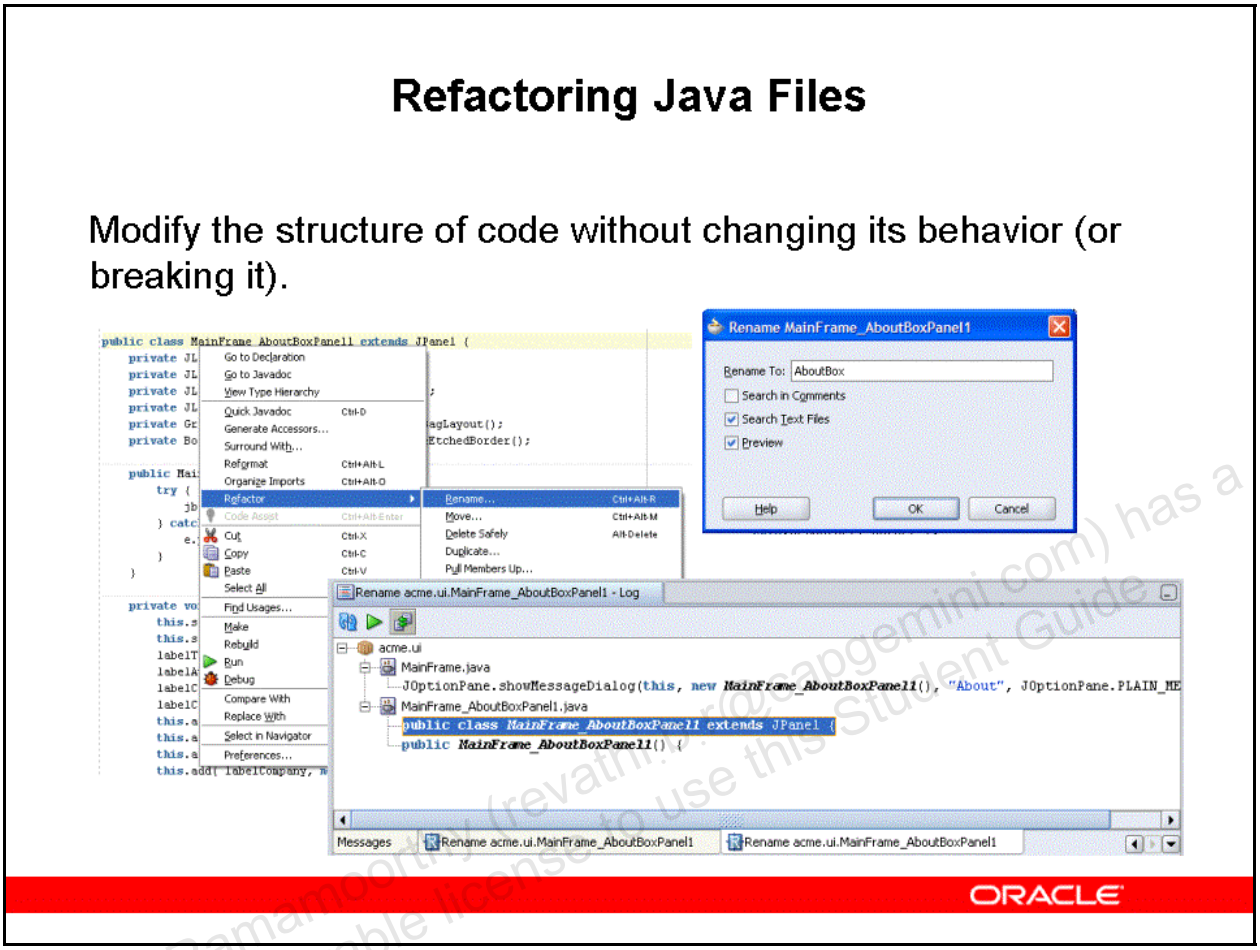
Customizing JDeveloper 10g with OA Extension

Customizing the IDE

You can customize JDeveloper's default display options (for example, whether the splash screen is displayed at startup and whether dockable windows are always on top) as well as other general behavior (for example, whether JDeveloper automatically reloads externally modified files and whether output to the Log window is automatically saved to a file).

You can do all of the following:

- Customize the general environment for the IDE.
- Customize dockable windows for the IDE.
- Customize the Component Palette.
- Load preset keymaps and customize individual accelerators.You can take advantage of the several existing keymap sets in JDeveloper or begin with an existing keymap and then customize it to suit your own coding style by changing which keyboard shortcuts, or accelerators, map to which actions.
- Customize options for the Code Editor.
- Add external tools to JDeveloper.



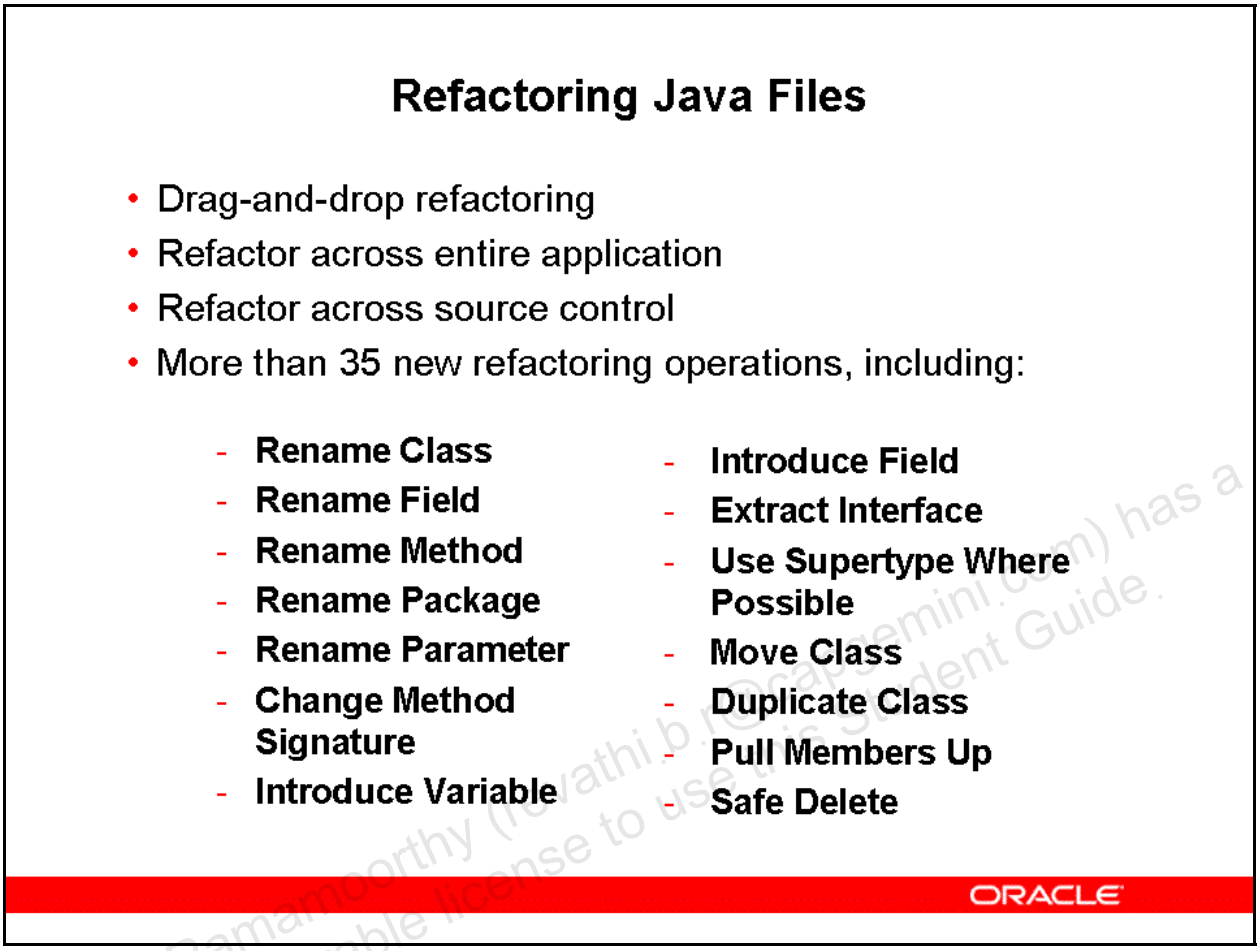
Refactoring

Refactoring is the process of improving an application by reorganizing its internal structure without changing its external behavior. In JDeveloper, you can make these changes without breaking any dependent files on which your project relies because these files are automatically updated for you.

Oracle JDeveloper 10g with OA Extension provides an even more powerful refactoring framework; as a result, performing refactoring actions is faster and much smoother. You can now right-click a method or field in the Java Structure window (or in the Code Editor) to initiate refactoring. When refactoring, JDeveloper searches your entire project and any projects that are listed in Tools > Project Properties, Dependencies.

Before proceeding with a refactoring action, you have the option of previewing the occurrences that will be updated (as the slide shows). You can then choose to continue with the refactoring action or cancel it. You can even undo refactoring if needed.

Refactoring Java Files



Refactoring (continued)

You can refactor across an entire application. In particular, you can refactor across multiple projects (via project dependencies).

Some of the available refactoring operations are:

Rename Class: Renames a class, its constructors, and its source file and updates all references to the class in the project

Rename Field: Renames a field and updates all references to the field in the project

Rename Method: Renames a method and updates all references to the method in the project

Rename Package: Renames a package and updates all references to the package in the project (including organization of subpackages)

Rename Local Variable: Renames a local variable and updates all references to the variable

Rename Parameter: Renames a parameter and updates all references to the parameter

Introduce Variable/Field/Constant/Parameter: Replaces the selected expression with a new expression of the same type

Extract Method: Creates a new method from the selected code, setting up parameters for any variables that need to be passed to the new method

Extract Interface: Creates a new interface from any of the public methods in the current type definition and implements that interface for the current type

Use Supertype Where Possible: Replaces all occurrences of a type with one of its supertypes if applicable

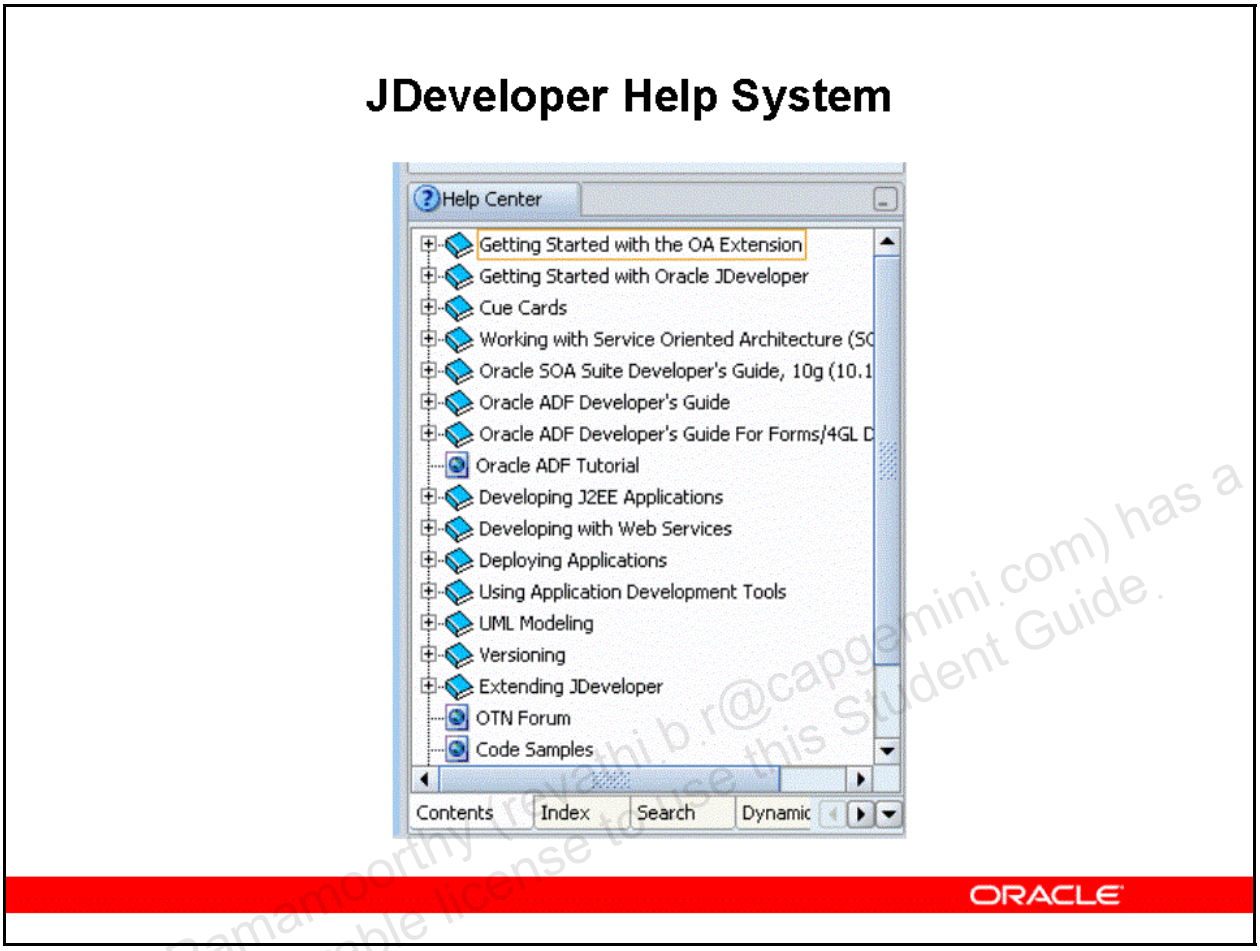
Pull Members Up: Promotes the declaration of methods or fields to the supertype and updates references accordingly

Push Members Down: Moves the declaration of methods or fields to all the subtypes of the current type and updates references accordingly

Delete Safely: Checks to make sure that the element you are attempting to delete is not actually being used in your code before allowing the delete operation to proceed. If references are found, you are warned and given the option to cancel the delete operation. Delete Safely can be used when deleting types, methods, and fields.

You can also Reformat Java files so that the indents and spaces are compliant with standards. The Reformat command in on the same pop out window as Refactor. Reformat changes the immediate file and not the entire project.

JDeveloper Help System



JDeveloper Help System

To make the best use of JDeveloper tools and libraries, and of Java itself, you can use a comprehensive help system that covers all aspects of Java development.

To access the help system:

1. Select Help > Table of Contents in the main menu. The help system appears.
2. Select one of the topics from the content navigator at the left of the window. After you select a topic, the topic expands to display subtopics.
3. Select the topic you are interested in and right-click it to display the help text in a window.

Use hypertext links to navigate within a topic or to related topics.

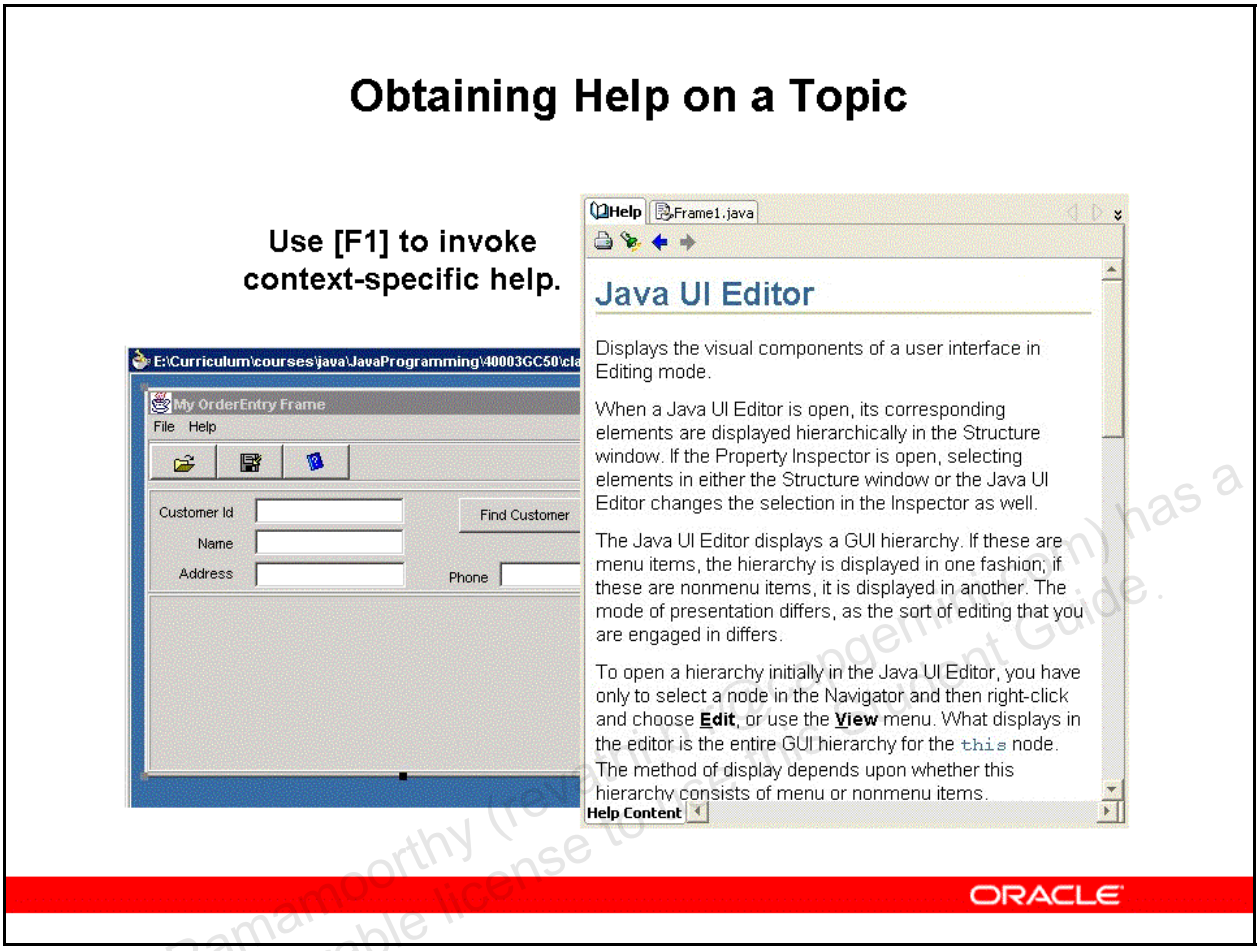
Note: The JDeveloper 10g with OA Extension contains both the JDeveloper 10g help and the specific help for the additional components provided by OA Extension and OA Framework.

Contained within the Help System, you will find several Tutorials. The Tutorials in the help system are legacy and may not work with the E-Business suite release 12.x. The Tutorials will contain obsolete examples or non functioning examples. Do not use any of the code snippets from the Tutorials for this class. Remember, portions of the Help

system come from the generic 10g JDeveloper and do not contain or pertain to the OAF extensions.

Note: The Oracle University course you are taking represents the most up-to-date version of the legacy based Tutorial and contains examples that are not within the legacy Tutorials.

Obtaining Help on a Topic

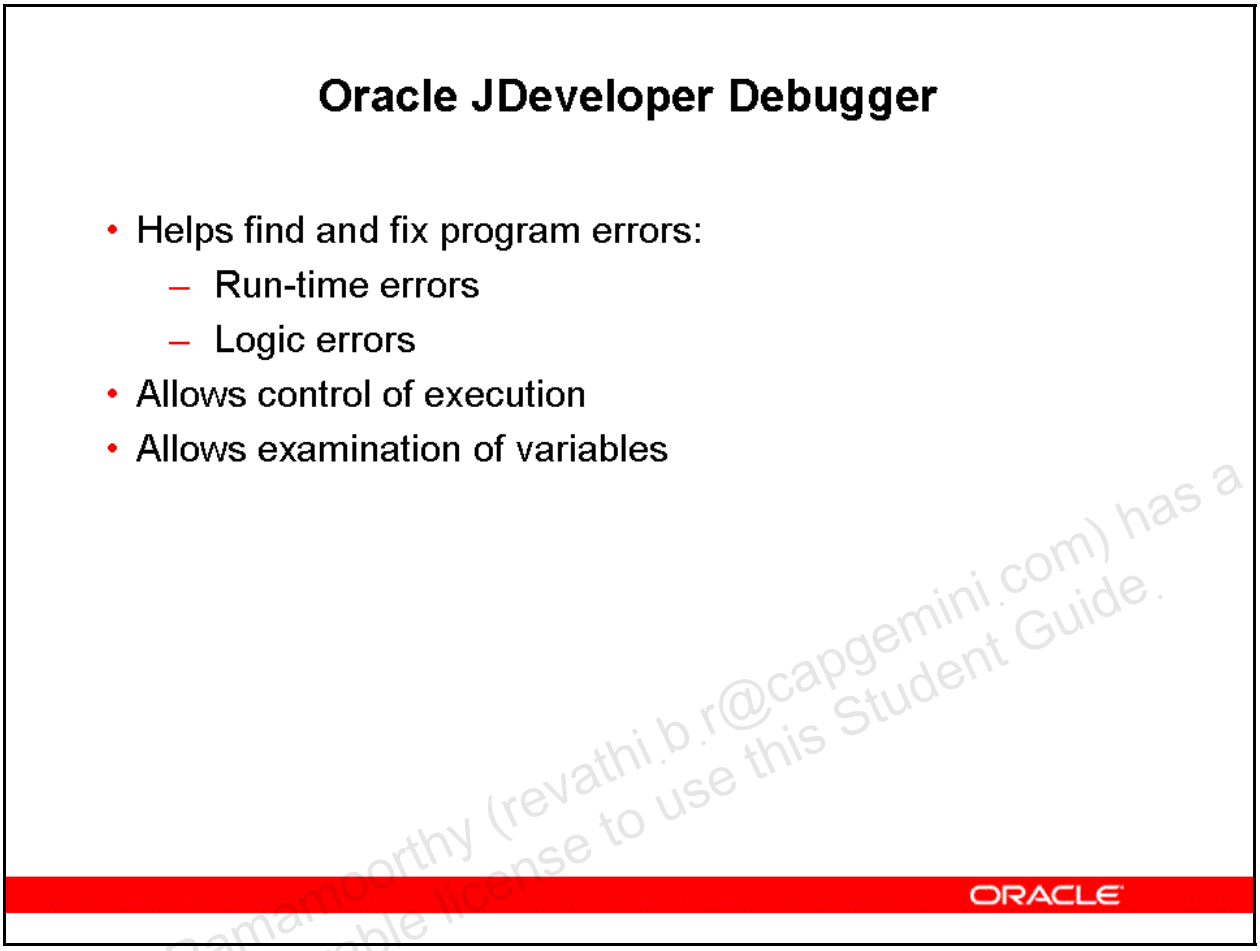


Obtaining Help on a Topic

Pressing the [F1] key invokes a context-sensitive Help topic window.

Javadoc is a tool that parses the declarations and documentation comments in a set of source files and produces a set of HTML pages describing the classes, inner classes, interfaces, constructors, methods, and fields.

When working in the Code Editor, you can quickly access the specific javadoc entry for any element in the source file. To browse the specific javadoc entry for a given class, member, or method, select the appropriate code element, right-click, and choose Quick Javadoc. A pop-up window with javadoc for just that element appears.



Oracle JDeveloper Debugger

Debugging is the process of looking for program errors that prevent your program from doing what you intended. There are two basic types of program errors: run-time errors and logic errors. Remember that the compiler catches any syntax problems.

If your program successfully compiles but gives run-time exceptions or hangs, then you have a run-time error. That is, your program contains valid statements but is encountering errors when they are executed. For example, you may be trying to open a file that does not exist or you may be trying to divide by zero.

Logic errors are errors in the design and implementation of your program. That is, your program statements are valid and do something, but the results are not what you intended them to be. This type of error is usually the most difficult to find.

The debugger enables you to control the execution of your program. It provides the ability to execute parts of the code step by step. You can also set breakpoints that pause the program execution when it reaches the line of code you want to examine.

While the program is paused, you can inspect and even modify program variables. This helps you examine loops and other control structures to make sure that what is happening is what you intended.

Breakpoints

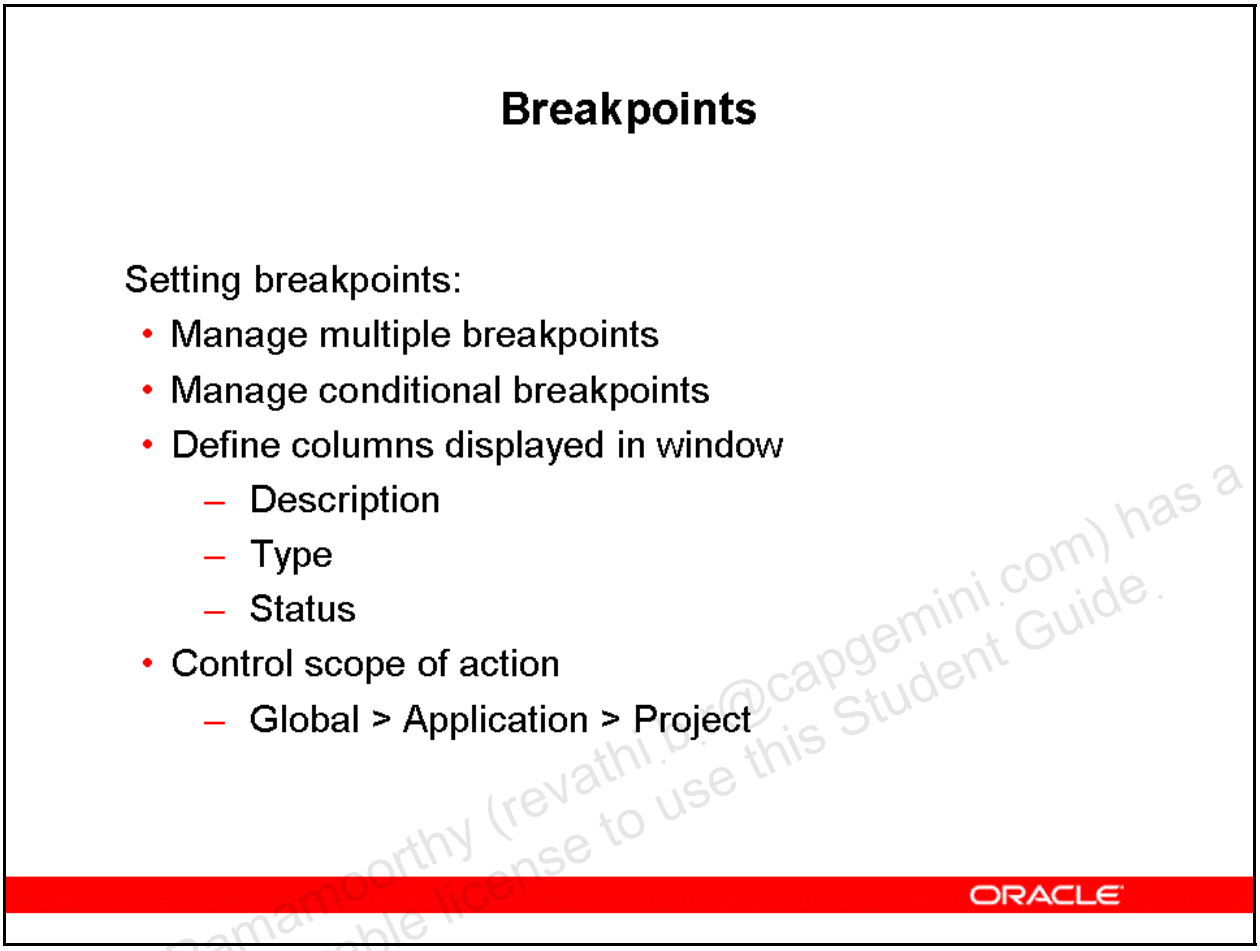
Breakpoints are a convenient way of tracing the cause of a problem in a program. When the debugger encounters a breakpoint, it pauses program execution. You can resume execution, stepping through the code line by line and examining variables and conditions. Or you can simply stop the program. You can set as many breakpoints as you want.

Watchpoints

A watchpoint is a new type of breakpoint in JDeveloper 10g with OA Extension. A watchpoint is a breakpoint that breaks on a value change.

Watchpoints enable you to pause the debugger when the value of a specified field is accessed or modified. You set a watchpoint by right-clicking a variable in the Code Editor and choosing Toggle Watchpoint from the shortcut menu.

Breakpoints



Breakpoints

Setting Breakpoints

To set a breakpoint, you select a line of code in the source code window, right-click, and then select Toggle Breakpoint. You can click in the left margin to set a new breakpoint. After you start debugging, breakpoints that are known to be valid have a check mark in the breakpoint icon. A breakpoint without a check mark may mean that this line does not represent code where the debugger can stop. However, it might simply mean that the debugger does not yet know whether the breakpoint is valid or invalid.

Viewing Breakpoints

To view all the currently enabled breakpoints, select View > Breakpoints from the menu bar. A window appears showing all the breakpoints that are set in the program. To disable or remove a breakpoint, right-click the breakpoint and select an action from the shortcut menu.

Conditional Breakpoints

To set the conditions on which you want a breakpoint to be activated, right-click a breakpoint and select Edit Breakpoint. On the Breakpoint Conditions tab, you can specify information about how and when the breakpoint is activated (including valid Java conditional statements and thread-specific conditions).

Display Settings

To select the columns that are displayed in the breakpoints window, right-click in the breakpoints window and select Settings. In the dialog box, select Debugger > Breakpoints in the navigator tree on the left, and then select the columns to display.

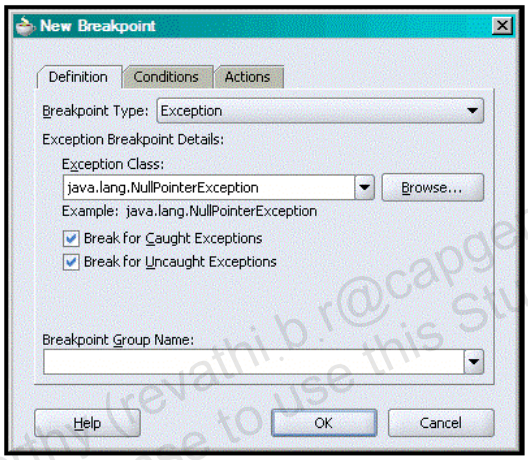
Scope Settings

To select the scope for debugging, right-click in the breakpoints window, select Change Scope, and then select the appropriate value.

Breaking on Exceptions

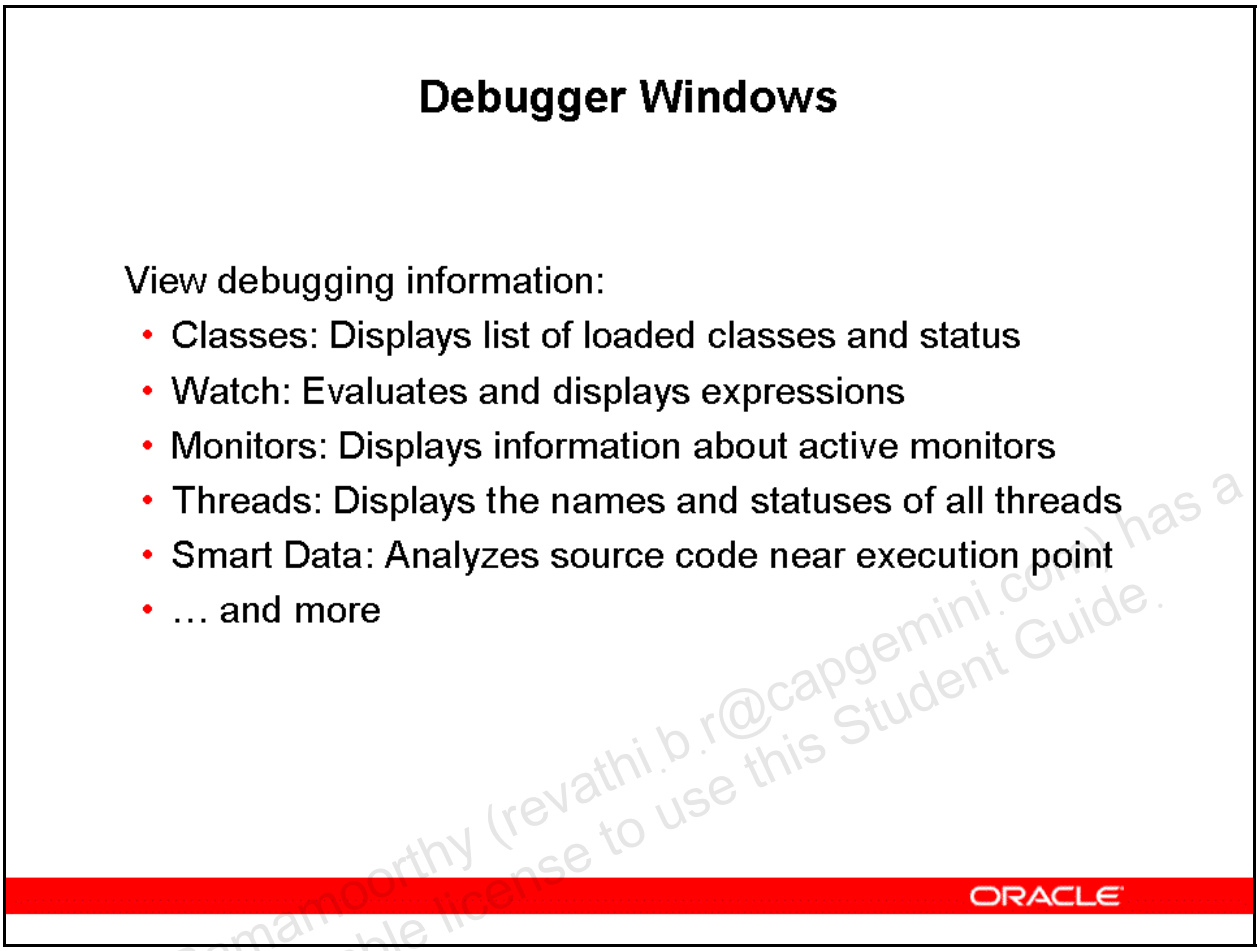
Breaking on Exceptions

- When you get an exception, create a new breakpoint with the Exception breakpoint type.
- Copy your exception from the error message exception stack and paste it into your new breakpoint.



ORACLE

Debugger Windows



Debugger Windows

Make sure that the project is selected, and then click the Debug icon. Alternatively, in the menu bar you can select Debug > Debug <Project Name>.jpr. This causes any required files to be compiled and then starts your program in debug mode.

Debugging Windows

As soon as you start the debugger, three tabs are added to a new window at the lower-right side of JDeveloper: the Smart Data tab, the Data tab, and the Watch tab. A new tab is added to the existing message window. You can use this tab to monitor the code as it executes. Fields for each window can be modified in the Tools > Preferences menu in the Debugger node.

Smart Data tab:

Displays only the data that appears to be relevant to the source code you are stepping through

Data tab:

Displays all the arguments, local variables, and static fields for the current method

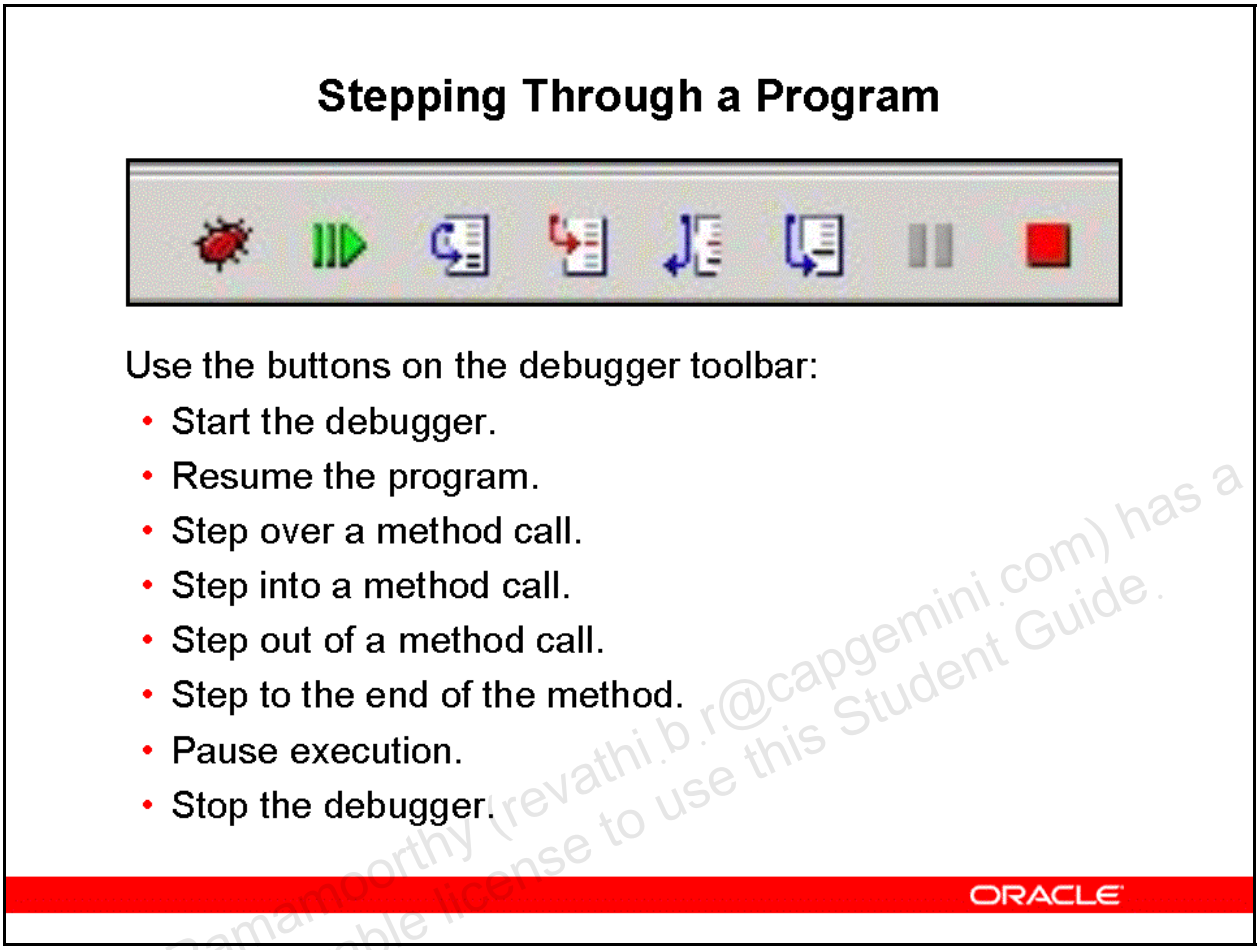
Watch tab:

Displays the current value of an expression that you have flagged to appear during execution of the program

Remote Debugging:

You can manually launch the program you want to debug and then start the debugger. In the Host list, select the name of the machine where the program has started. After the debugger is attached to the running program, remote debugging is similar to local debugging.

Stepping Through a Program



Stepping Through a Program

Click the following buttons in the debugger toolbar:

Start the debugger:

Executes the program in debug mode. The program is paused when it encounters a breakpoint. If no breakpoints are set, you can pause the program by clicking the "Pause execution" button.

Resume the program:

Resumes the program after stopping at a breakpoint

Step over a method call:

Executes the method at the current position in the program at full speed rather than tracing the method line by line

Step into a method call:

Traces a method line by line. This is useful when you suspect that the method may be the one that is causing the problem.

Step out of a method call:

Enables you to step out of the current method and return to the next instruction of the calling method

Step to the end of the method:

Jumps to the end of the method

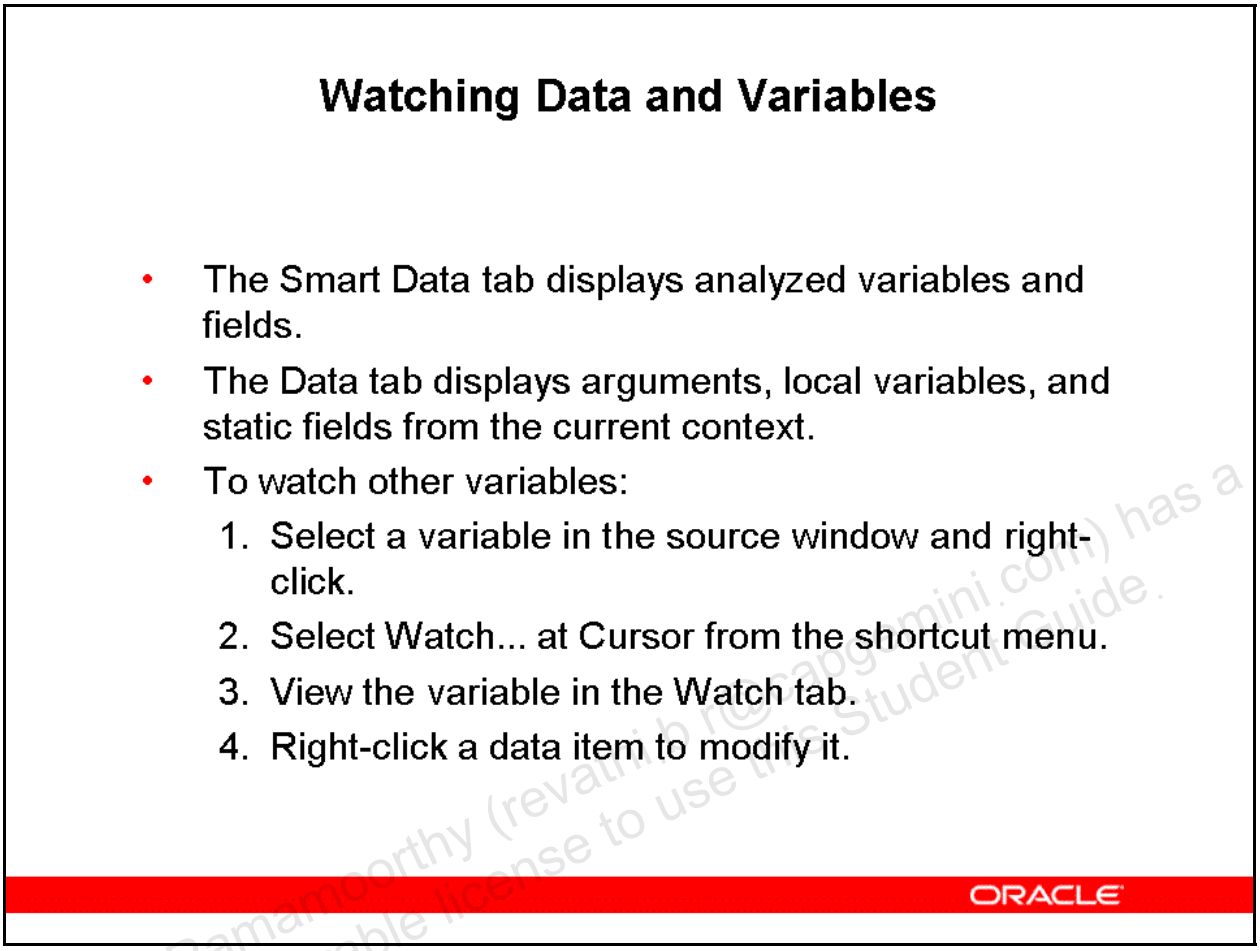
Pause execution:

Pauses a running program at its current position

Stop the debugger:

Stops the execution of a running program. This is a useful way of killing the program.

Watching Data and Variables



Watching Data and Variables

Viewing Analyzed Data on the Smart Data Tab

The debugger analyzes the source code near the execution point, looking for variables and fields expressions that are used in the lines of code. By default, the debugger analyzes only one line of code for each location.

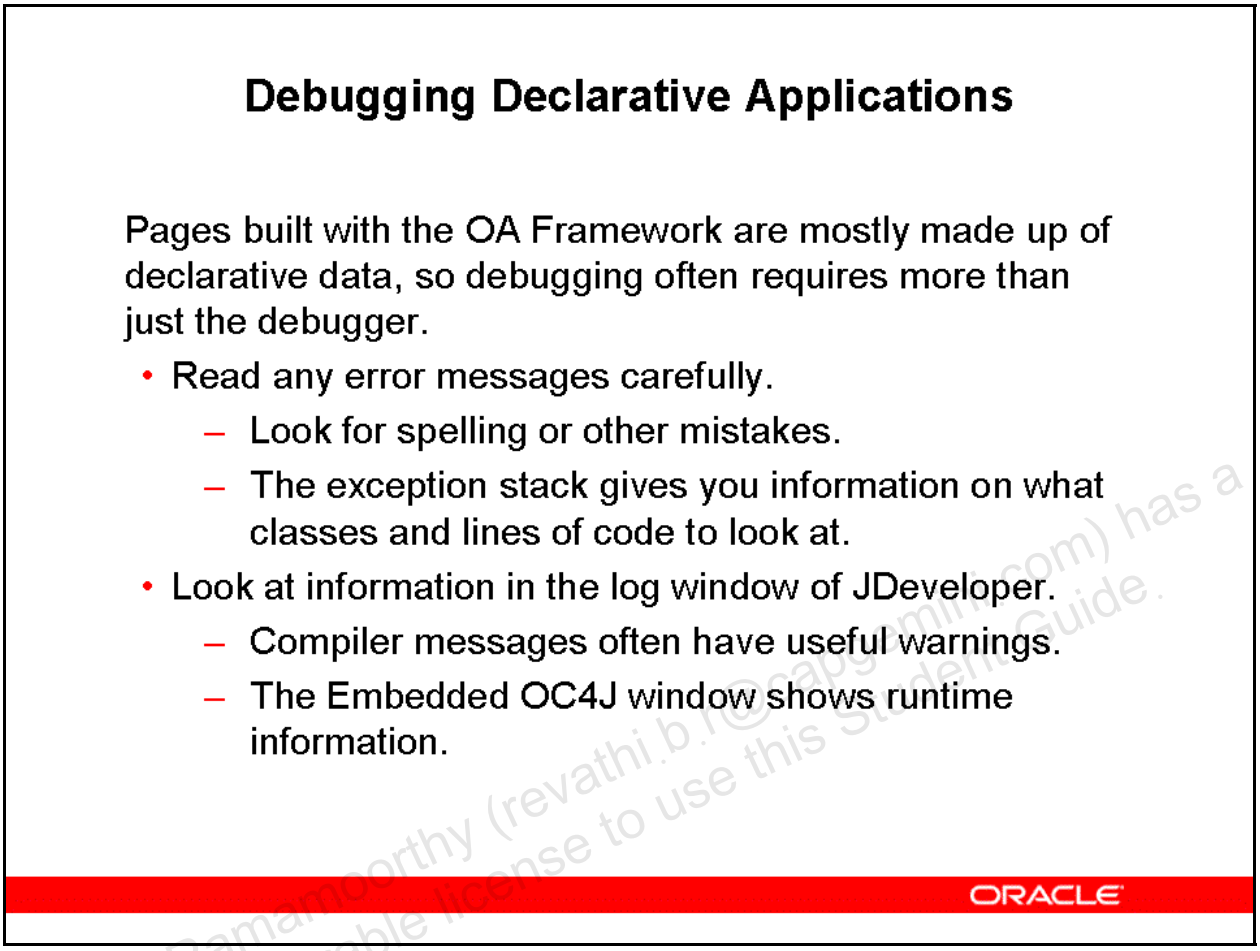
Viewing Local Variables on the Data Tab

The Data tab is the lower window that is displayed when you click the Debug tab at the bottom of the Applications Navigator. The Data tab automatically displays a list of local variables, static fields, and arguments that are in scope. As you jump from one method to the next, the list of variables displayed on the Data tab changes.

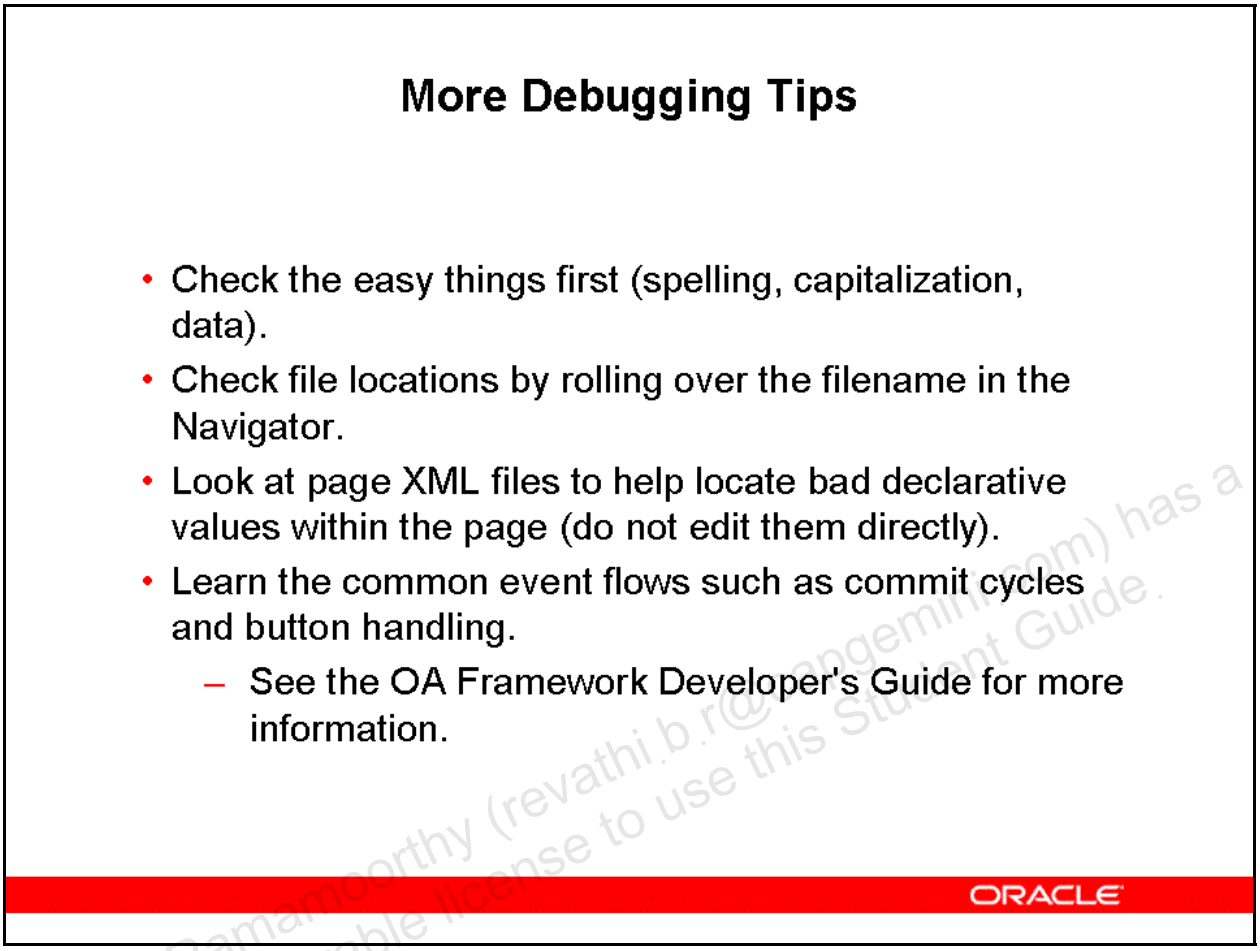
Selecting Other Variables and Expressions to Watch

Other variables and expressions can be viewed by following the steps described in the slide. Select a variable or expression such as age+10. Right-click the variable or expression and select “Watch… at Cursor” from the shortcut menu. A dialog box displays the selected variable or expression; click OK to accept it and add it to the

Watch tab. View the variables that you have selected to monitor on the Watch tab. To modify a data value, right-click it and select Modify Value from the shortcut menu.



For example, compiler messages can warn you that you forgot to provide a setting for important properties that would affect how the page runs. The Embedded OC4J window shows runtime information such as SQL statements being executed as well as their bind variables.



Understand BC4J Interactions

- Follow BC4J interactions from an item (field) to the underlying BC4J objects.
  - Root AM, View Instance, and View Attribute properties for a pageLayout region, tables, and items
    - Are these set correctly?
  - View object definitions and code
    - Is the query correct? Does it return data?
    - Are the attribute settings and mappings correct?
  - Entity object definitions and code
  - Entity associations and view links
  - Application module definitions and code
    - Are the view objects included correctly?

ORACLE

Understand BC4J Interactions

- Check any redirections, links, and forwards to other pages or the same pages, either in Destination URI property or in code.
  - Are view attribute names correct?
  - Are you passing correct parameter names and values?
  - Are page URL paths and function names correct?
- Check if a page is getting the AM state information it expects.
  - Does the page have AM State Required set correctly?
  - Is the RetainAM flag set correctly for the transition to the page?

ORACLE

Debugging Validation and Commits

If you are using entity experts, validation VOs, and validation AMs and getting errors:

- Is the ExpertClass property set correctly on the owning entity object?
- Is the VAMDef property set correctly on the owning entity object?
- Are the validation VOs included in the VAM?
- Does the validation code reference objects such as VO instances correctly?
- If you are writing data back to the entity object (in processFormData, through the VO), was the entity object actually created properly?

ORACLE

Summary

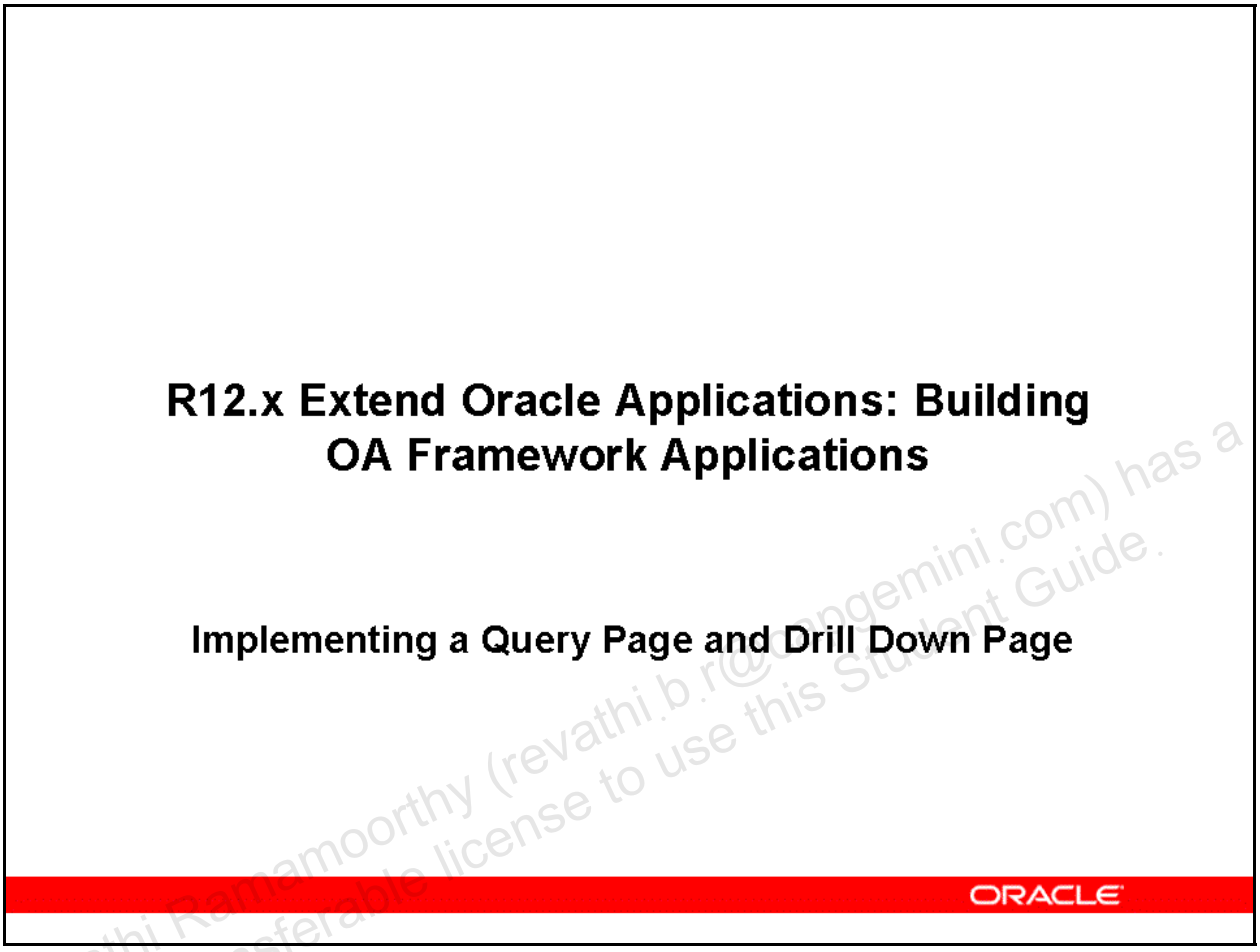
In this lesson, you should have learned how to:

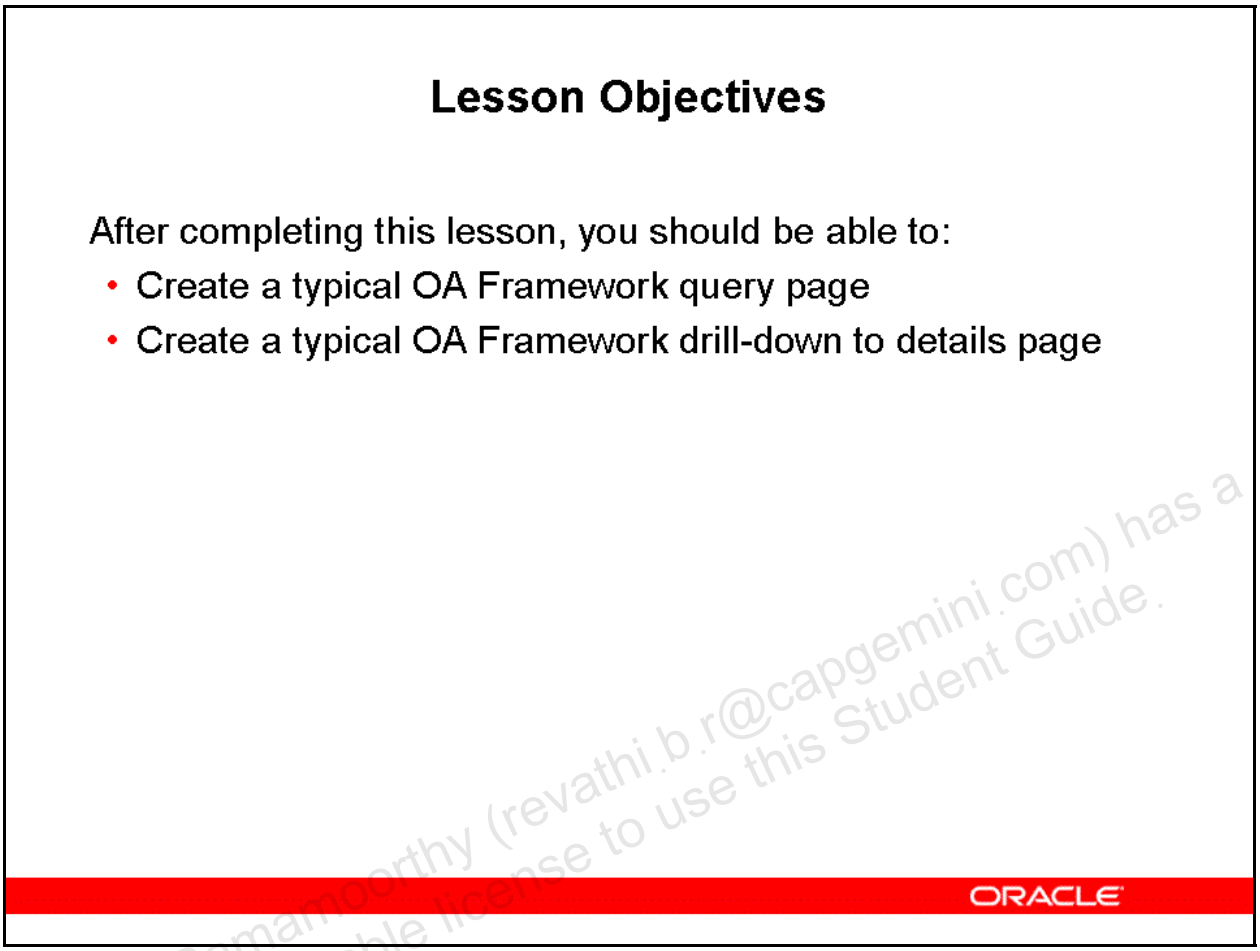
- Use JDeveloper 10g with OA Extension.
- Use the JDeveloper 10g debugger.
- Configure JDeveloper 10g with OA Extension to connect to an E-Business Suite instance to run OA Framework pages.

ORACLE

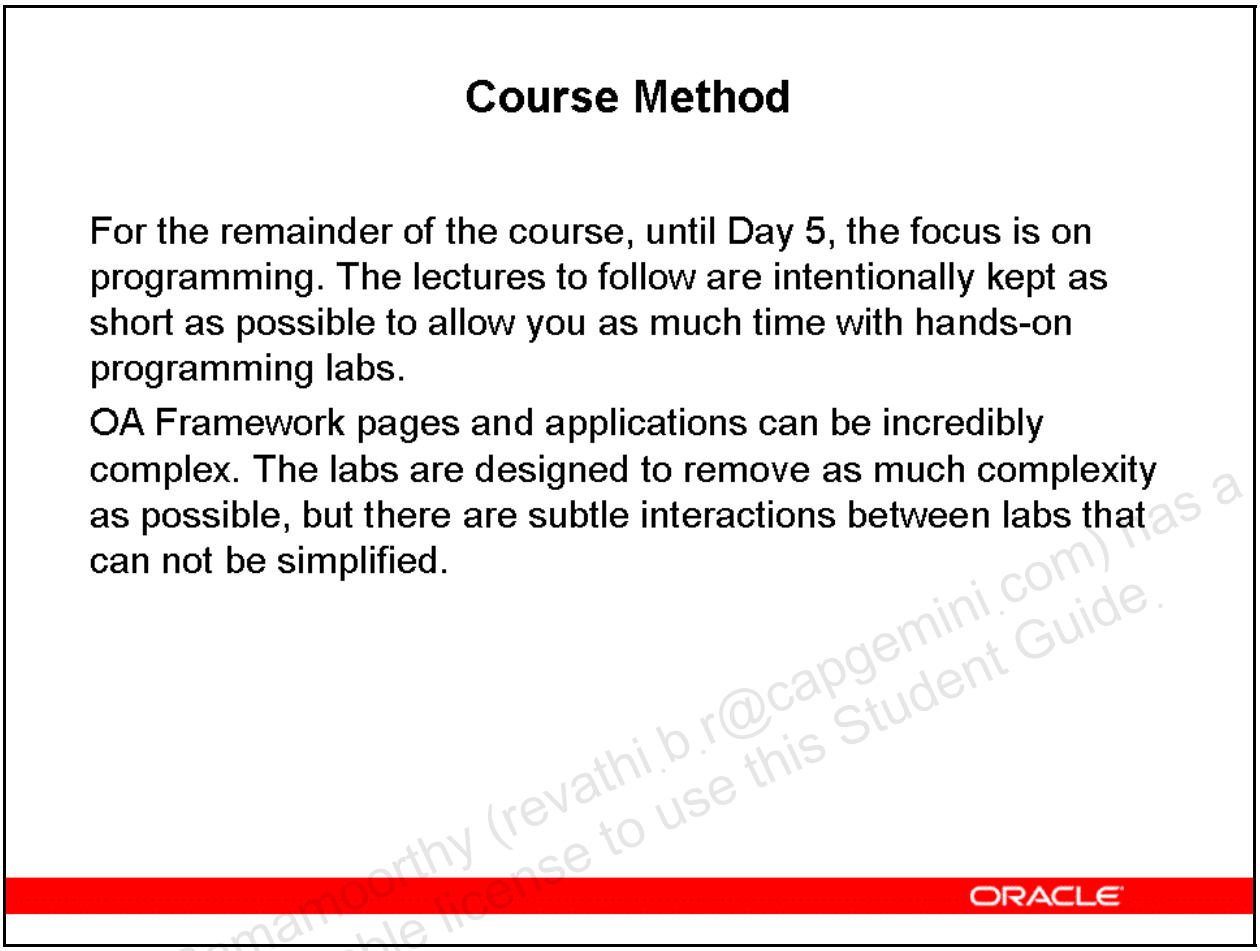
Implementing a Query Page and Drill Down Page

Chapter 9





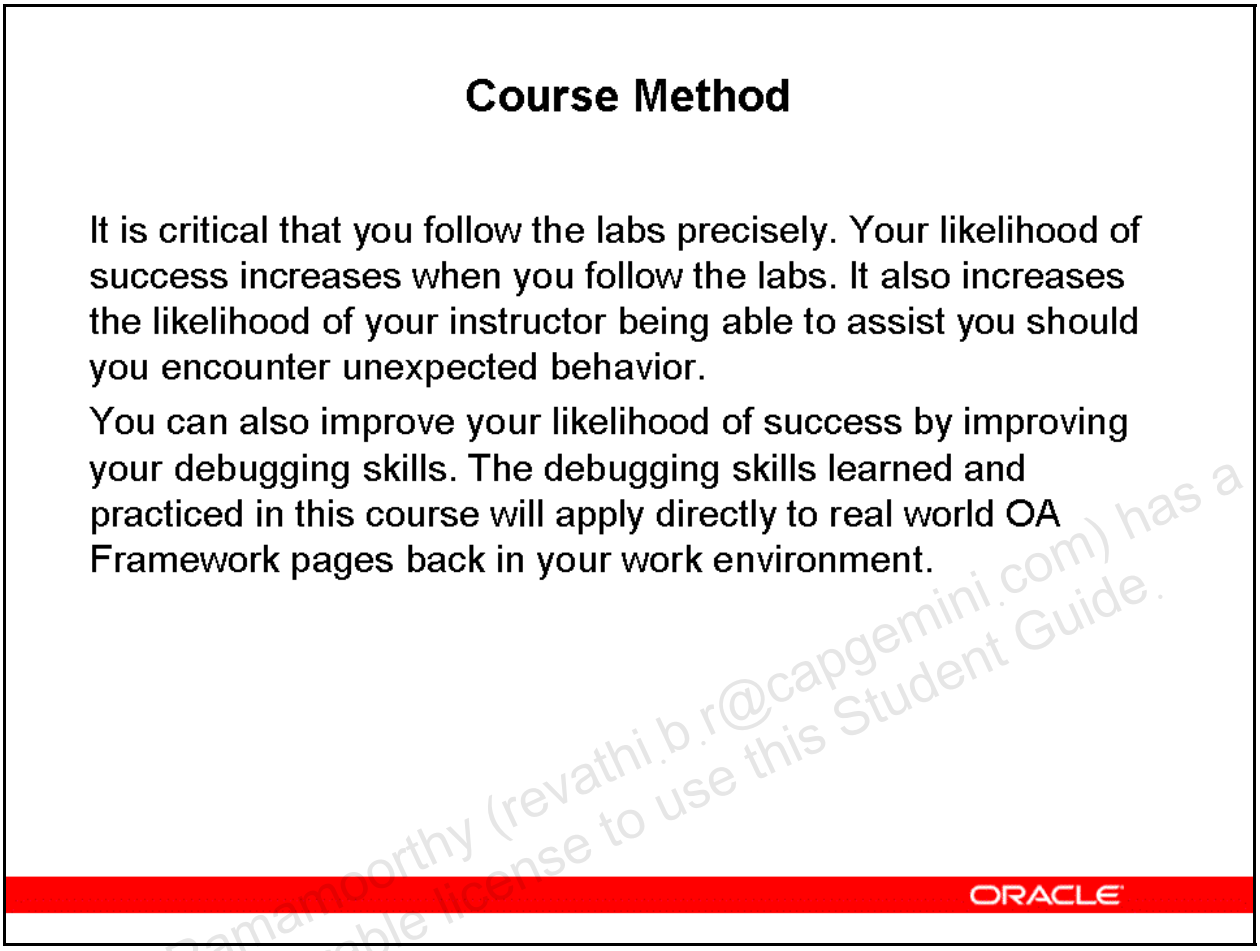
Course Method



Things to remember.

1. Java is case sensitive
2. SPEL is case sensitive
3. Select the correct attributes from your attribute sets.
4. Save your project and workspace frequently.
5. The labs are sequential and cumulative.
6. Do not change order or skip sections unless your instructors say you can.
7. Take your time and take frequent breaks.
8. Sometimes have another set of eyes to review your work. Looking at the same error over and over will hide something that is apparent to another set of eyes.
9. This is a programming course, so give the instructor a little time to come up to speed on exactly where you are in your steps and process. There is no magic code snippet that the instructor can use to correct days of incorrect steps.
10. The initial labs and setting up the BC4J objects are the backbone for all that is to come.

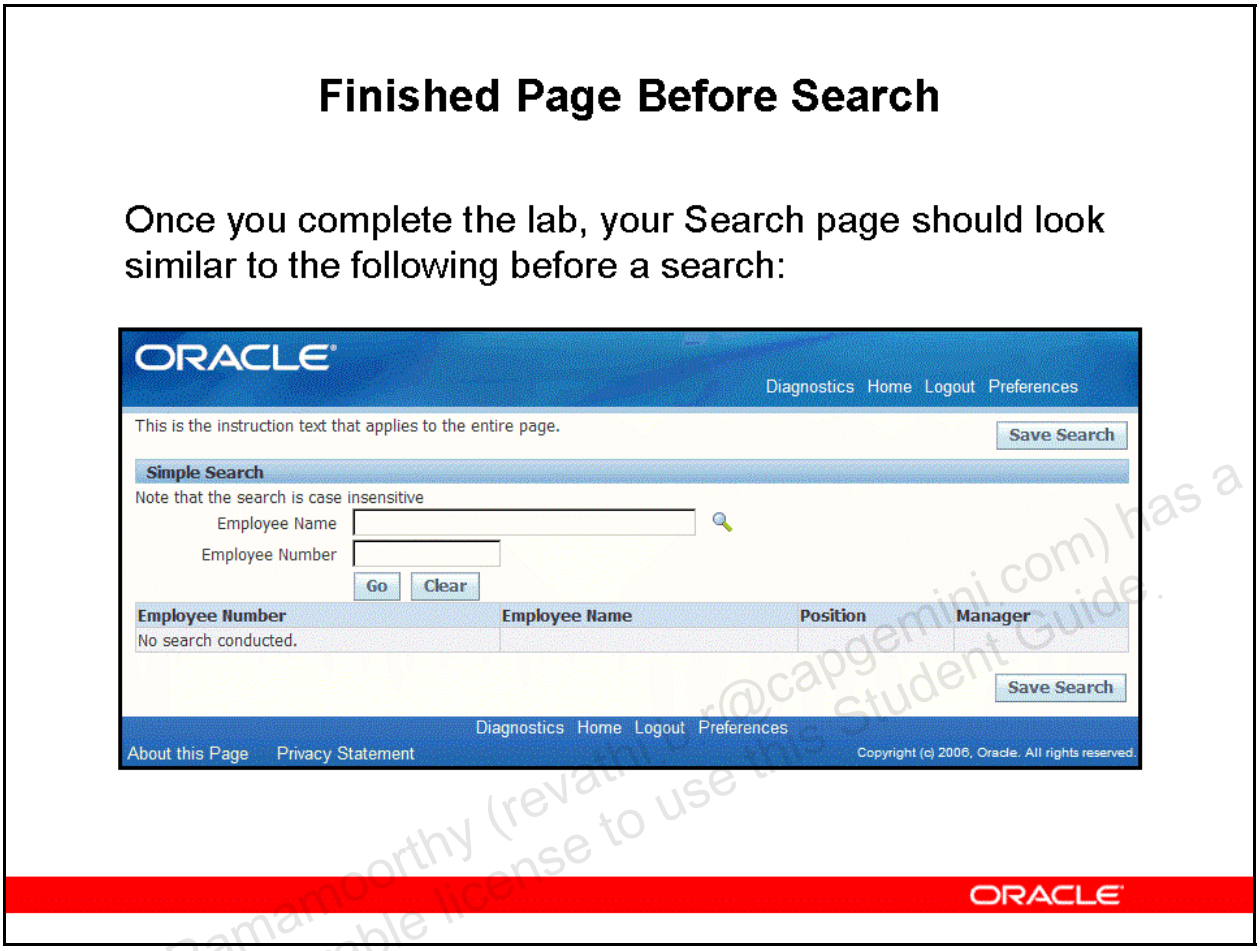
Course Method



We want to convey, that if you understand the steps to make the error, your success in finding out what is not working when you get back to your company will be much greater.

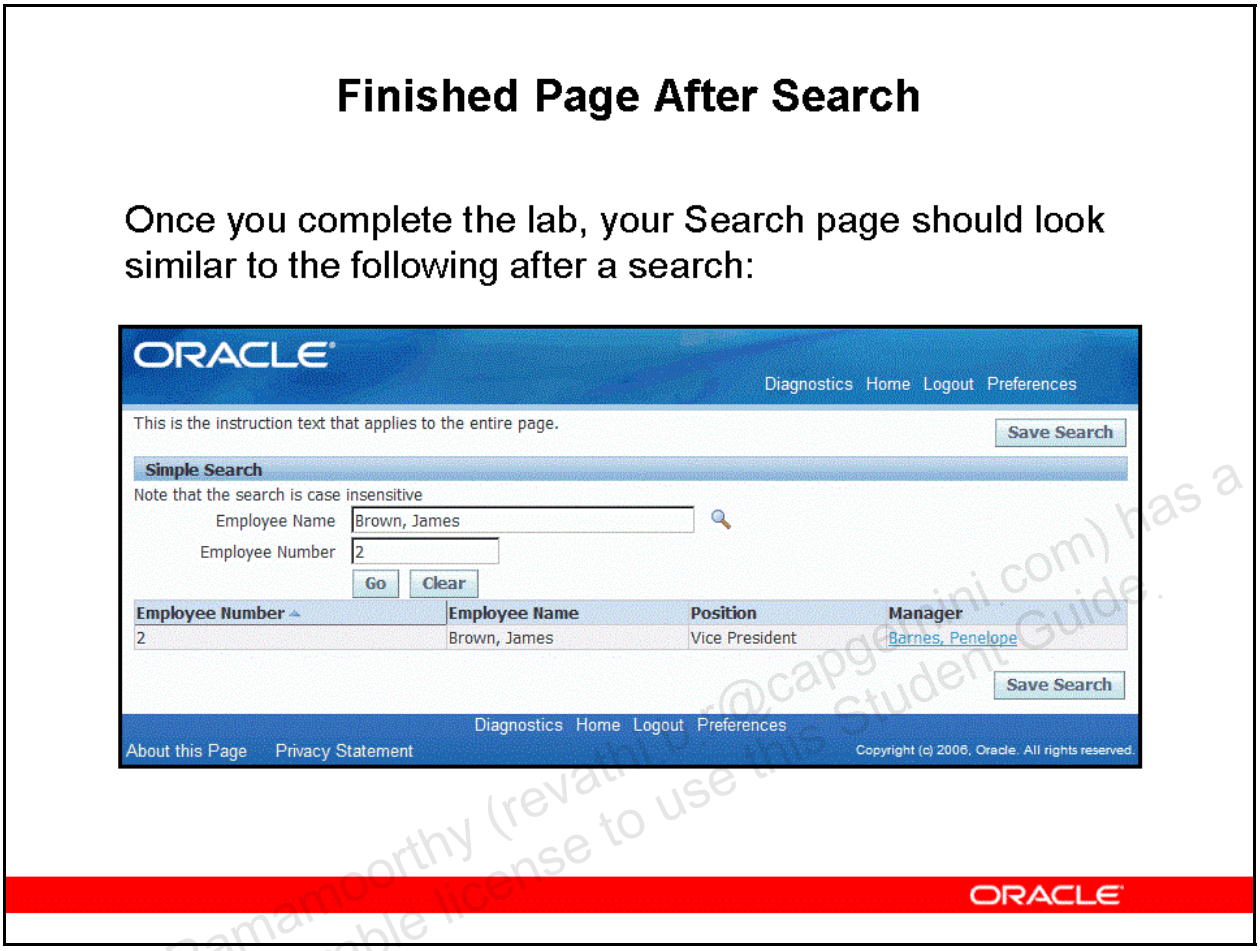
Remember that OAF is a J2EE framework. The Model, View, Controller are all components that make of your application. When you have something that doesn't work right, think about the component that is directly responsible and how the linkage to the MVC creates pages and how they are interrelated. Think about what you see and don't see in both the errors, and what is displayed when you run your test pages.

Finished Page Before Search



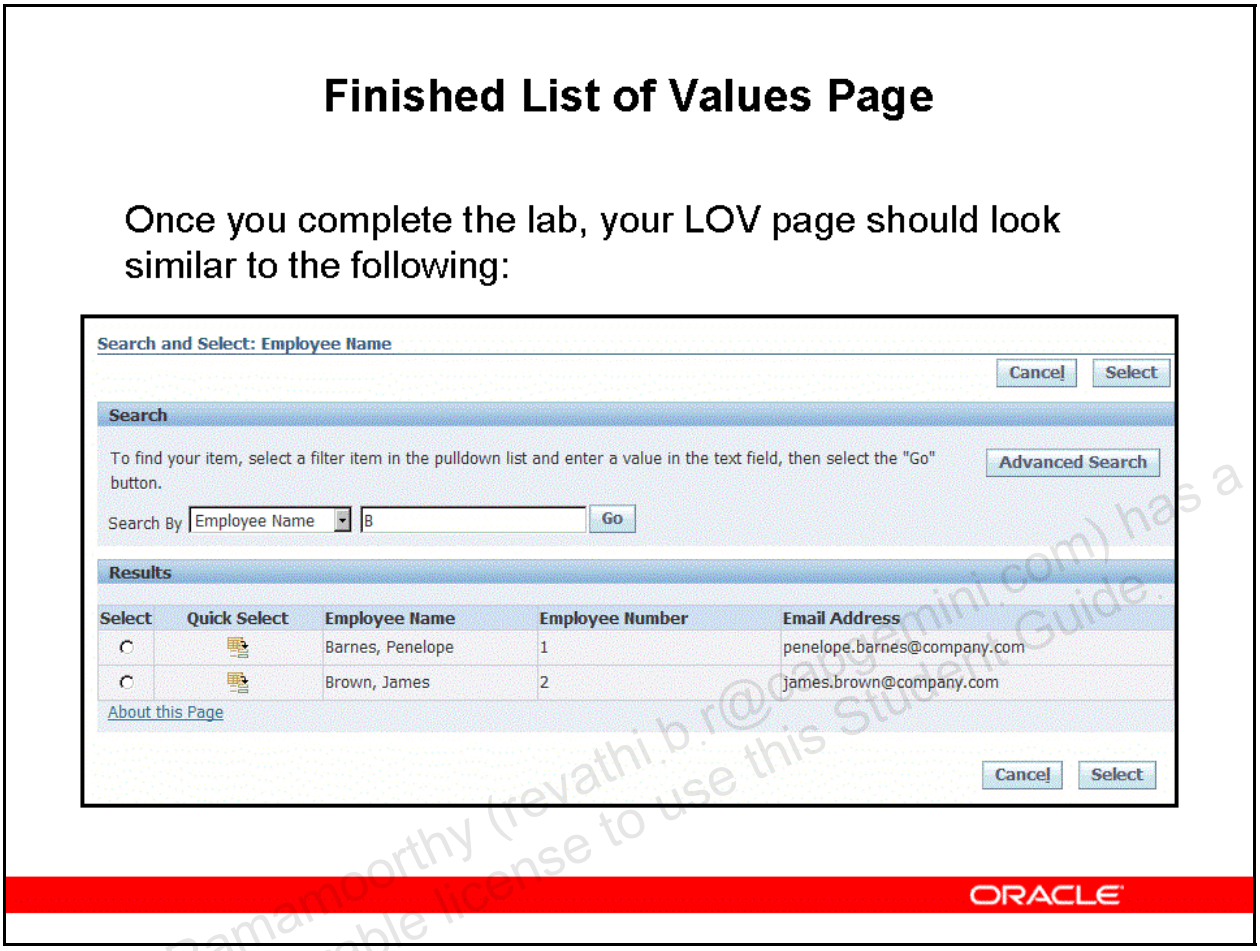
The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Finished Page After Search



The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Finished List of Values Page



The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

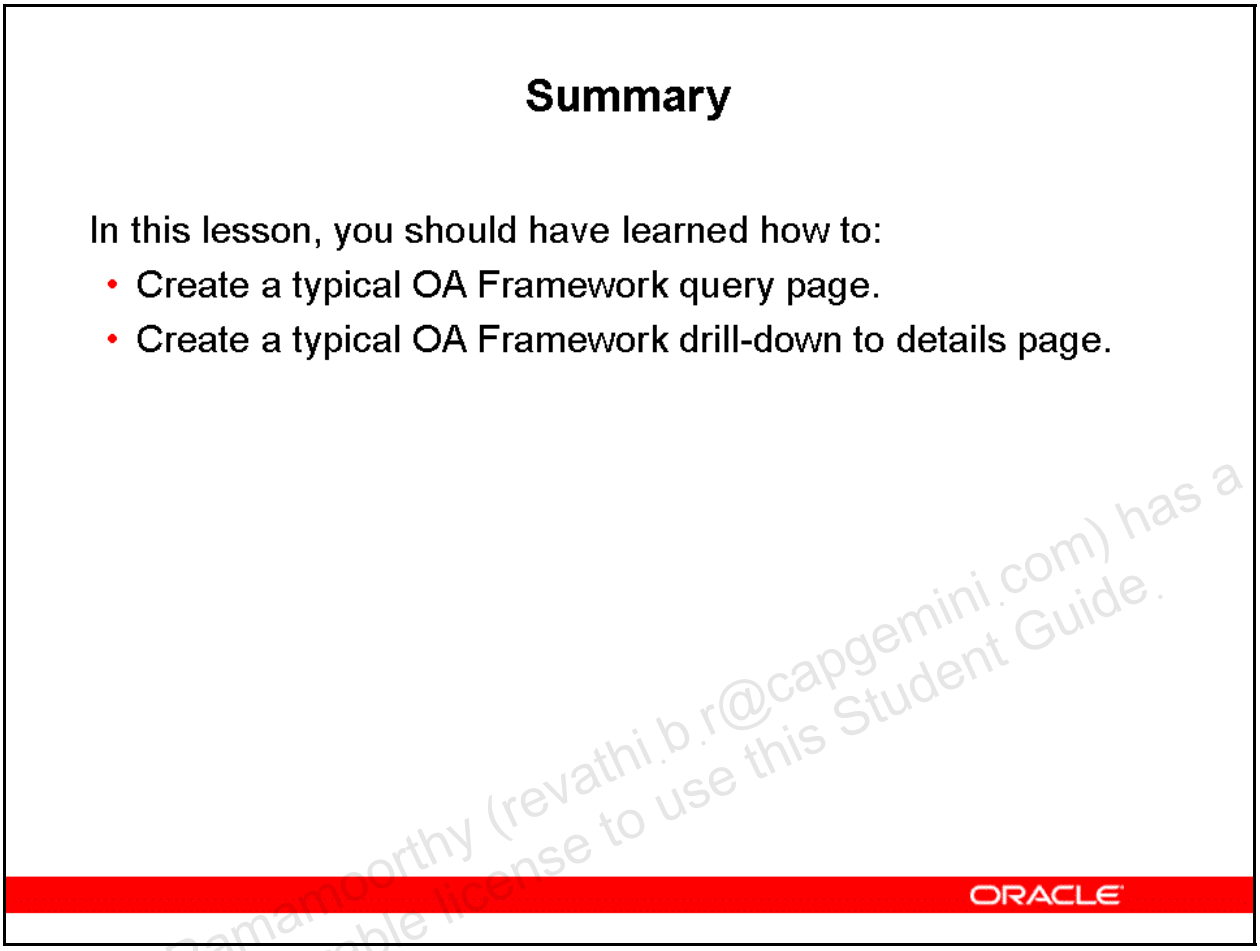
Finished Drilldown-to-Details Page

Finished Drilldown-to-Details Page

Once you complete the lab, your Details page should look similar to the following:

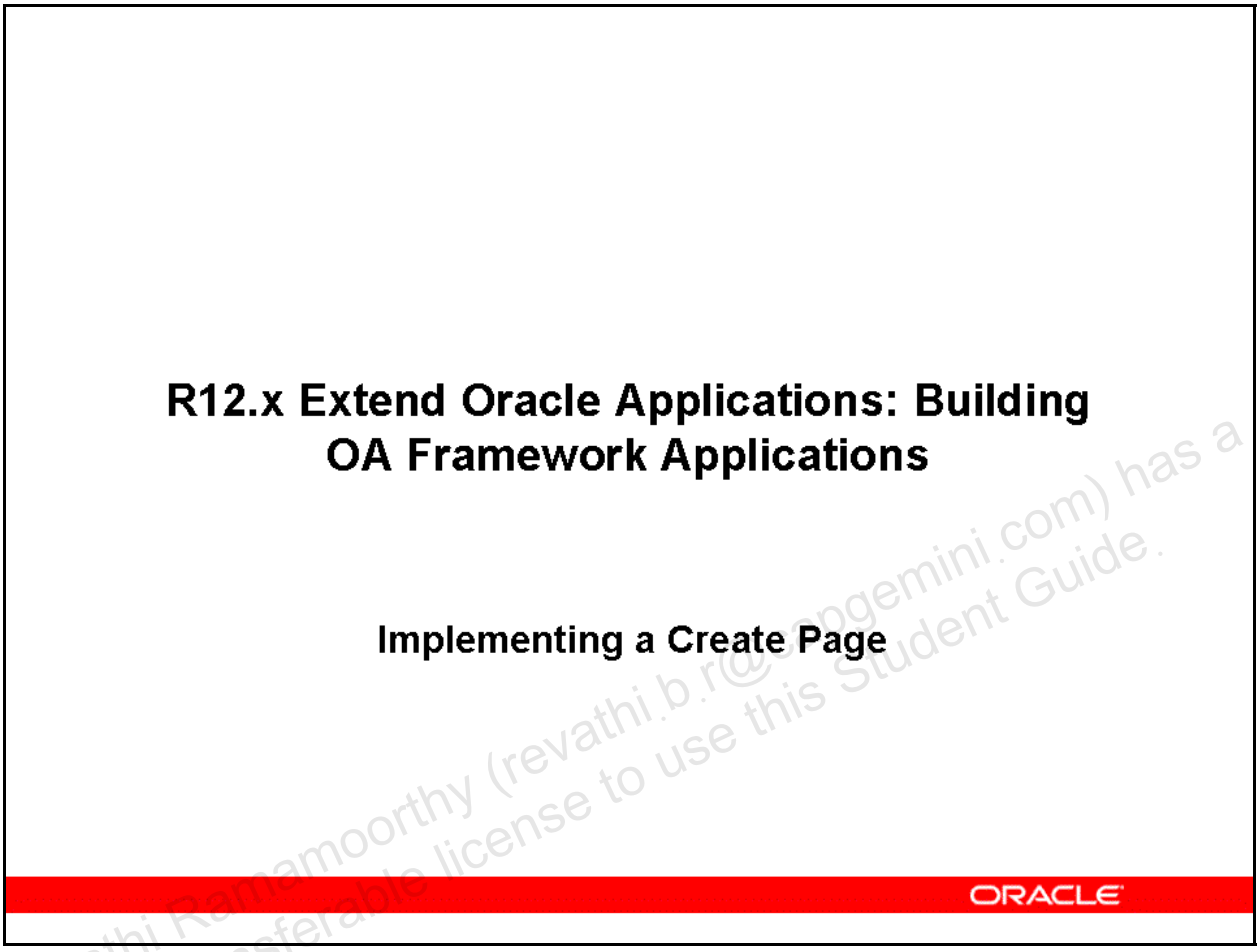
ORACLE

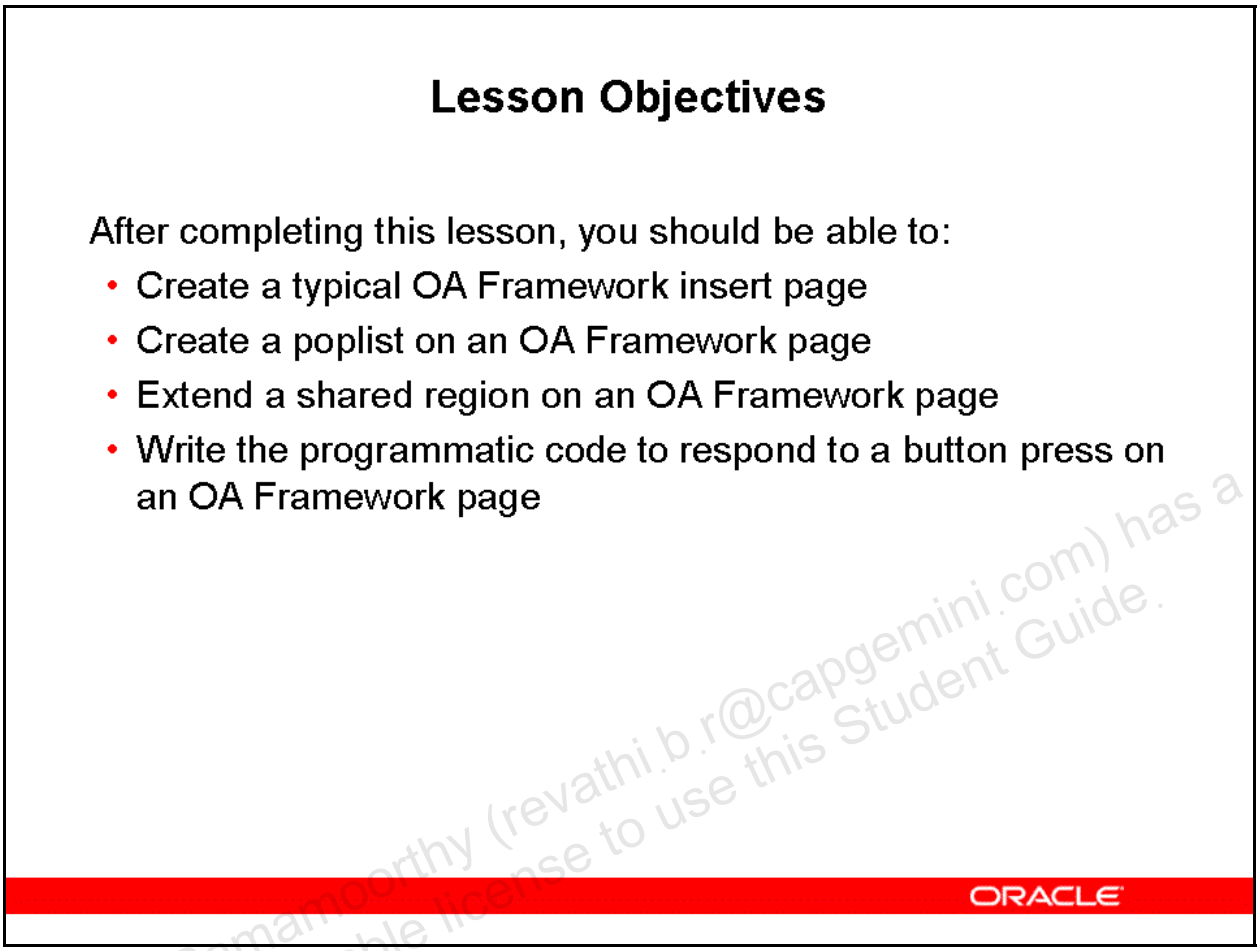
The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.



Implementing a Create Page

Chapter 10





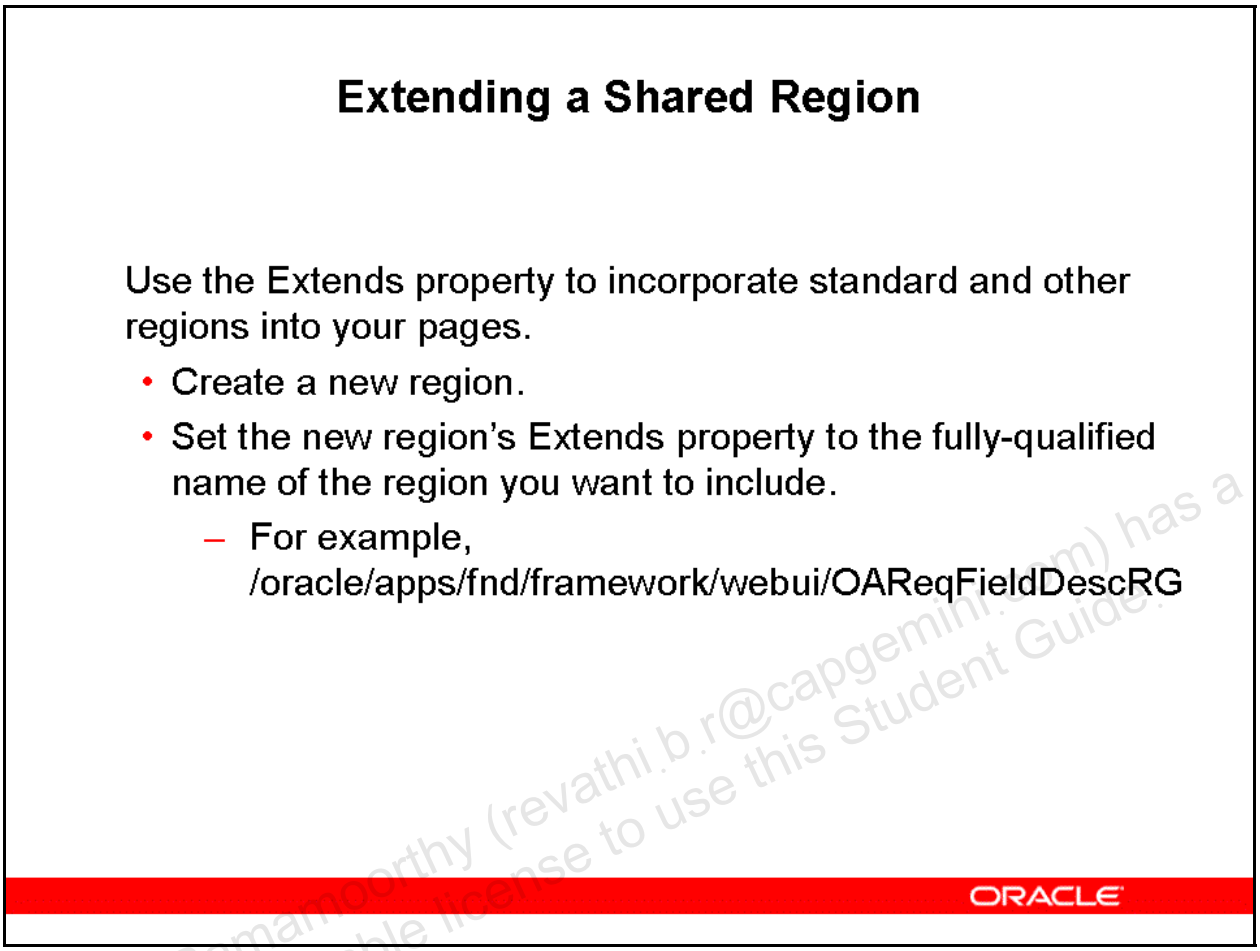
Implementing a Poplist

Poplists use two view objects:

- VO for the base page item
  - View Instance
  - View Attribute
- VO for the values in the list
  - Definition
  - Value
  - Meaning

ORACLE

Extending a Shared Region



Whenever you need a BLAF-standard "Indicates Required Field" key in your page, you can simply extend the common OA Framework region by extending /oracle/apps/fnd/framework/webui/OAReqFieldDescRG.

Note: For the Extends property, you may need to deselect the "Show Components With Valid Scope Only" check box in the dialog window if the shared region is outside your application packages or otherwise restricted in scope. In other cases you may want to restrict the choices to components with a valid scope, such as in your particular application.

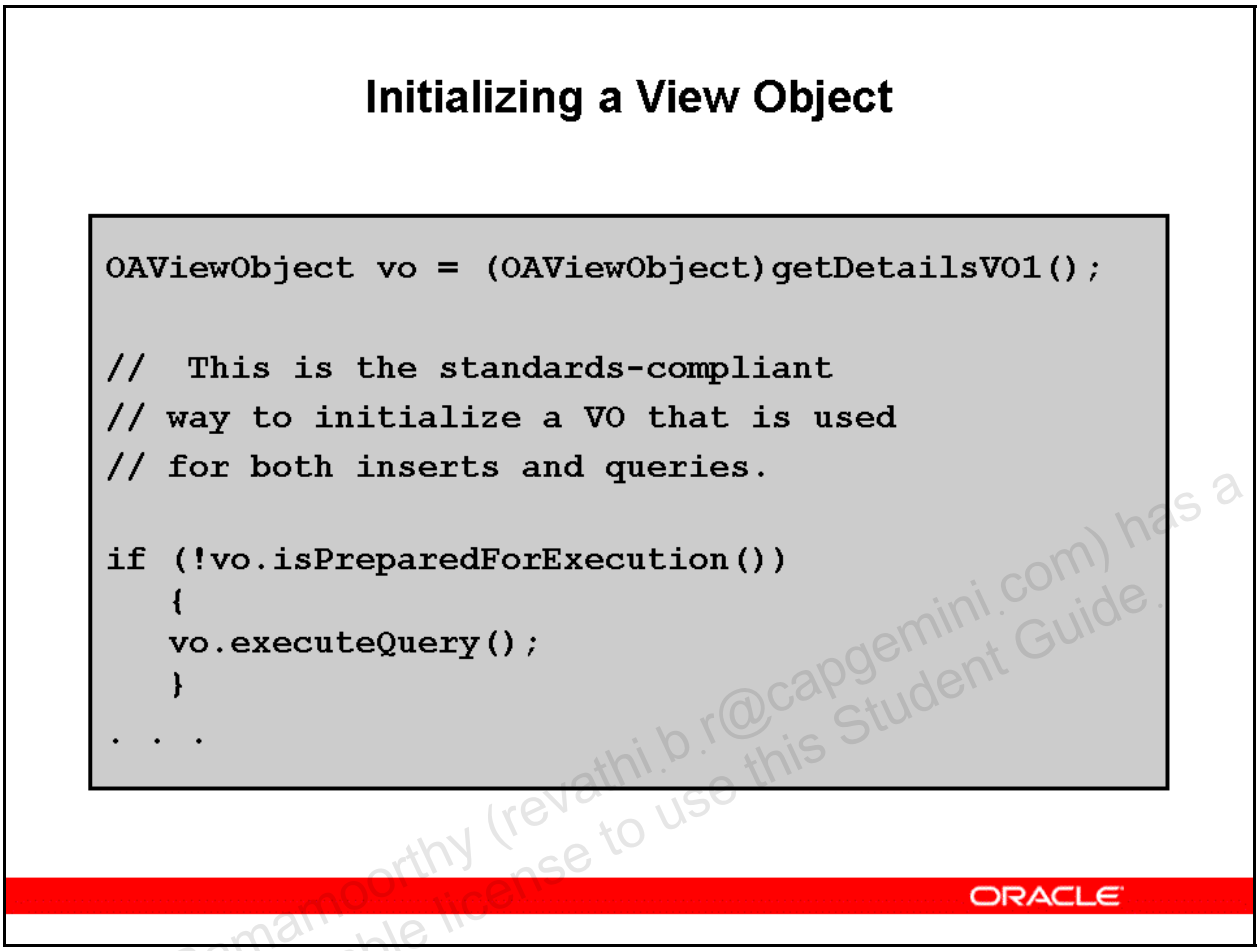
Creating a New Row

Before the user can enter a new row, the middle tier must be prepared to accept the data (usually during page initialization):

- Get and initialize the view object.
- Create an empty view object row, and corresponding entity object if any, on the middle tier.

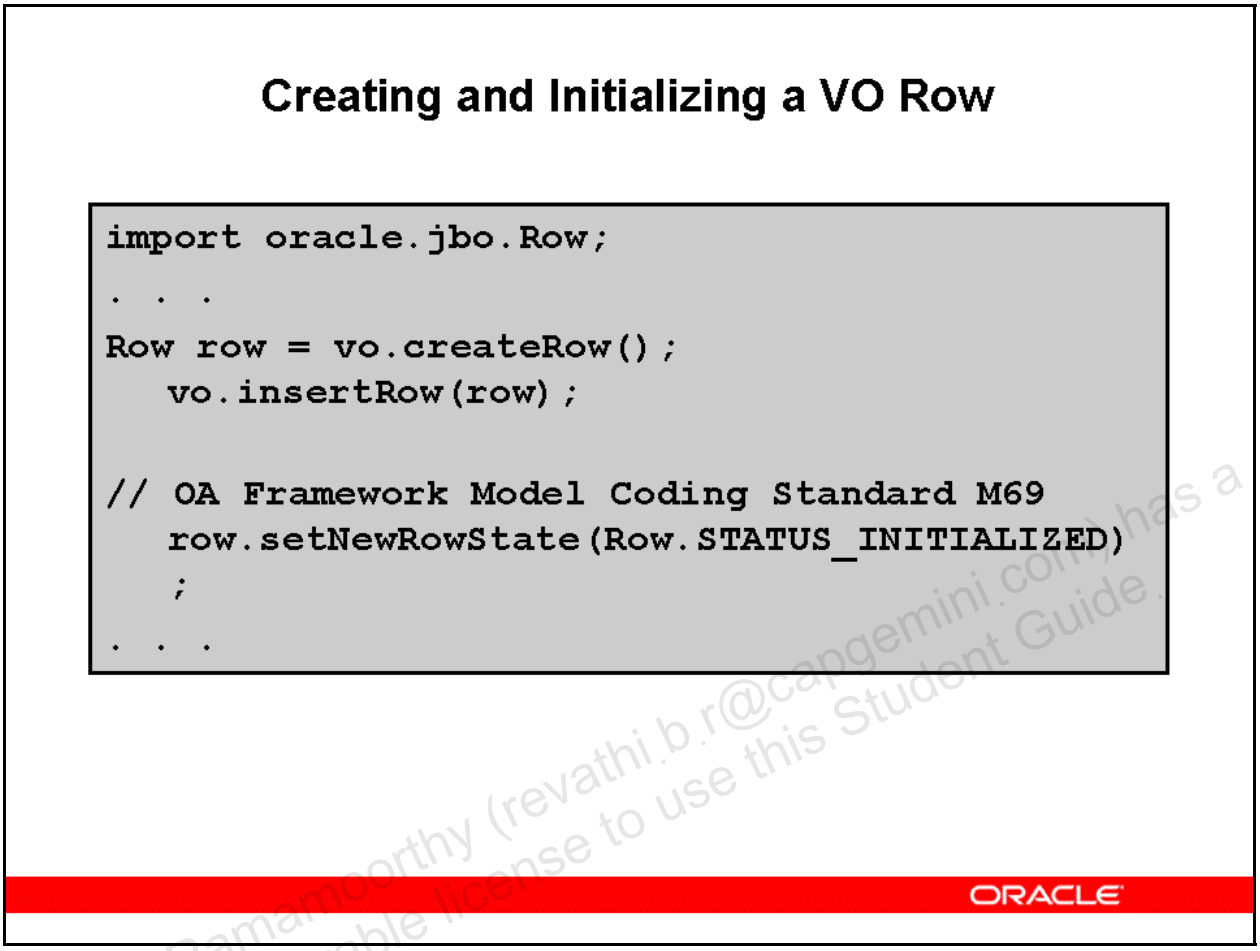
ORACLE

Initializing a View Object



This code would typically be in a create() method in the application module for the page (such as a createEmployee() method in an EmployeeAMImpl.java file). The method would be invoked from the client-side controller code, either from processRequest() or processFormRequest().

Creating and Initializing a VO Row



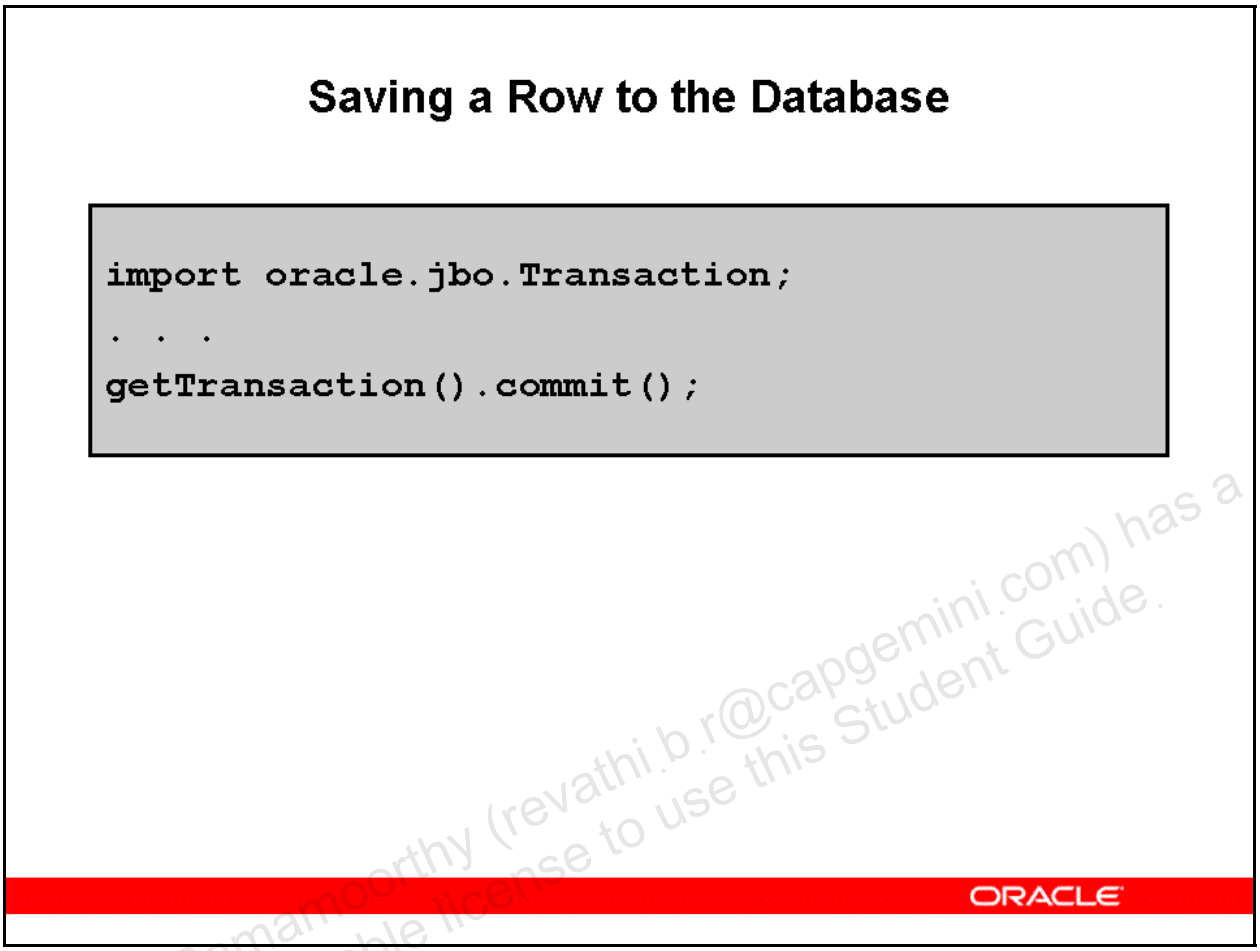
This code would be right after the view object initialization in the create() method on the application module.

The view object internally delegates to the entity object, if there is one, to execute entity object methods such as create(). The create() method provides any default values for the new row.

Getting the Data

- The user enters values and does an action, such as pressing an Apply button, to cause a form submit.
- The processFormData method writes the user's data to the middle tier into the view object row and triggers validation.
- If the validation is successful, commit the row to the database.

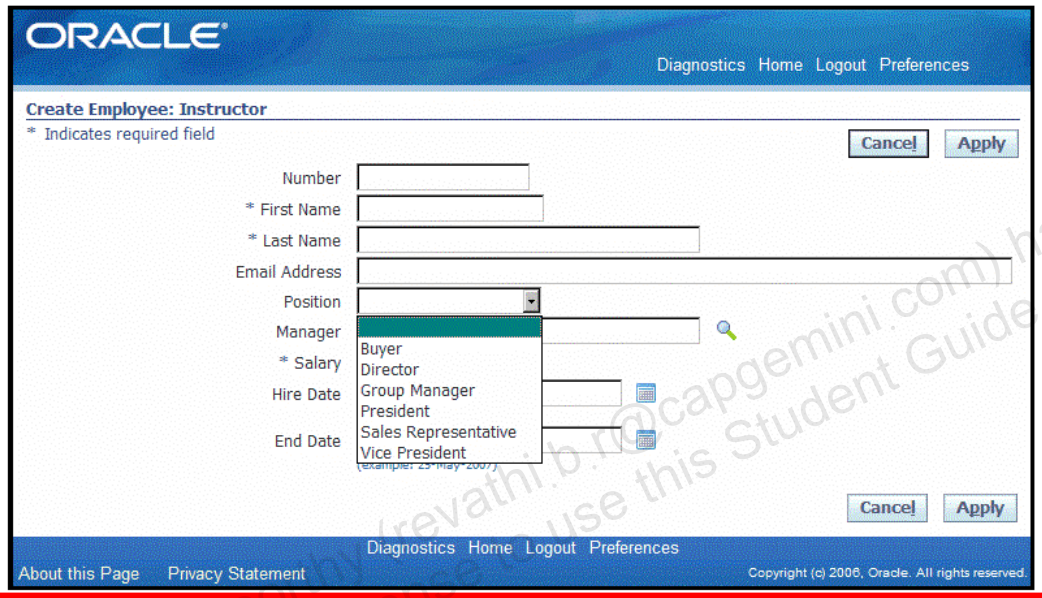
ORACLE



Lab – After Create Basics

Lab – After Create Basics

After the first part of the Lab, Create Basics, your progress should look similar to the following:

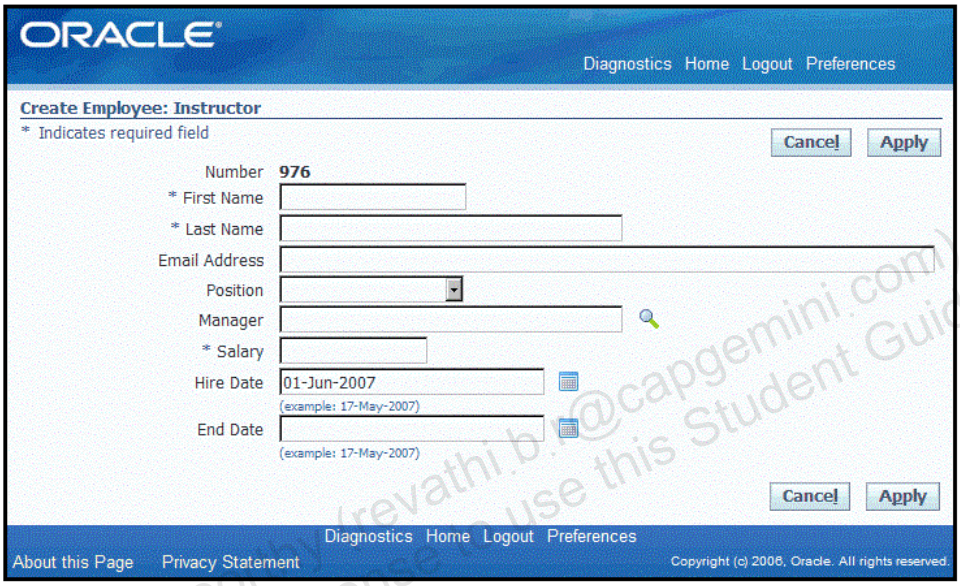


The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Lab – After Validations

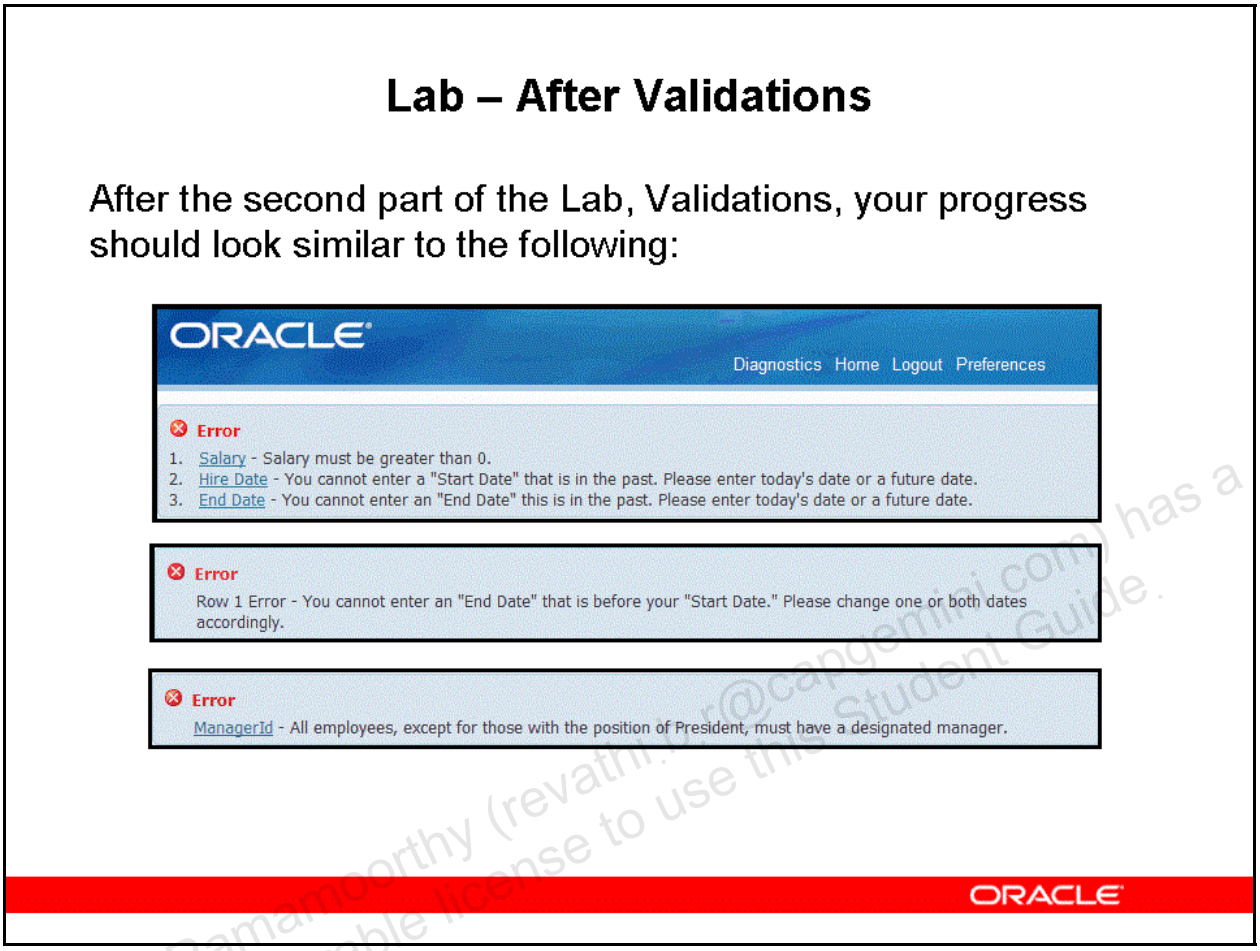
Lab – After Validations

After the second part of the Lab, Validations, your progress should look similar to the following:



The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Lab – After Validations

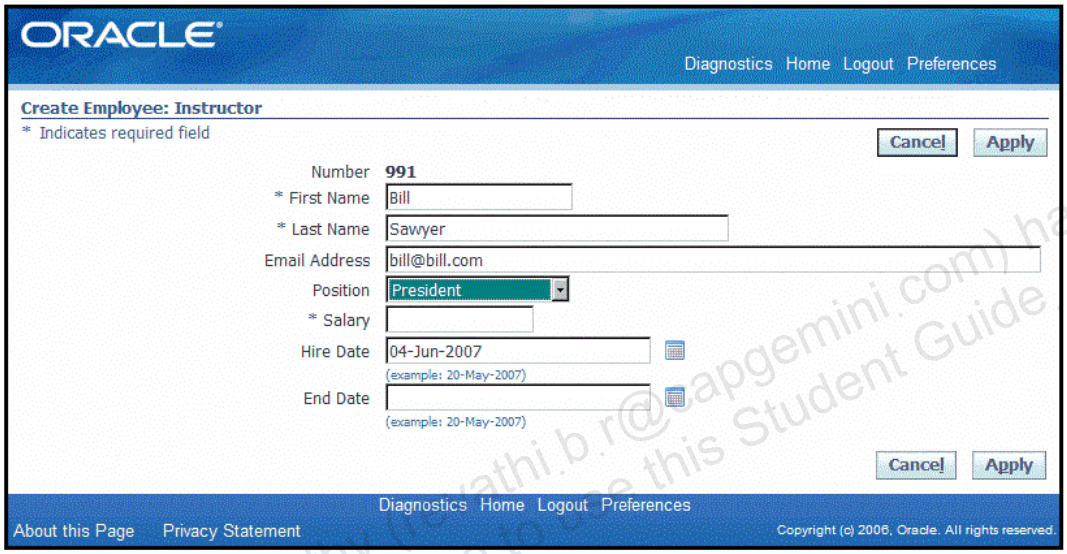


The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Lab – After Partial Page Rendering

Lab – After Partial Page Rendering

After the third part of the Lab, Partial Page Rendering, your progress should look similar to the following:



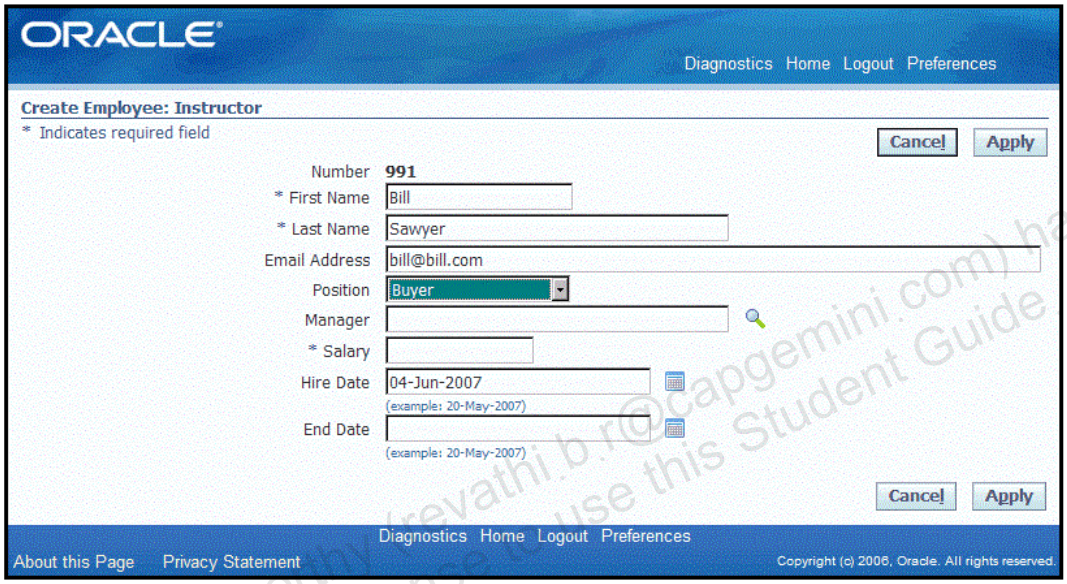
Note: This sample shows the Position field set to President. Notice that when the Position is President, there is no Manager field presented in the UI.

The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Lab – After Partial Page Rendering

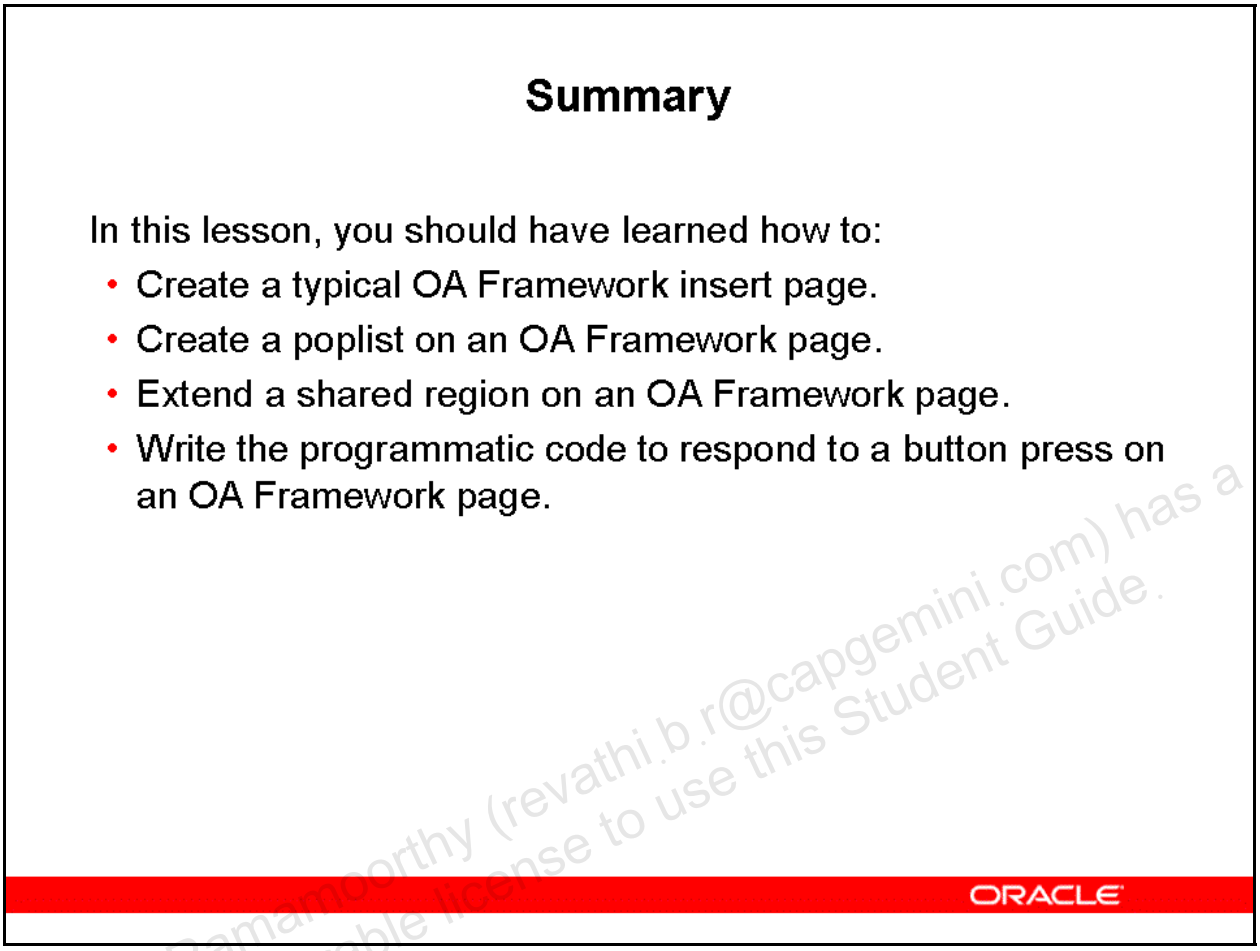
Lab – After Partial Page Rendering

After the third part of the Lab, Partial Page Rendering, your progress should look similar to the following:



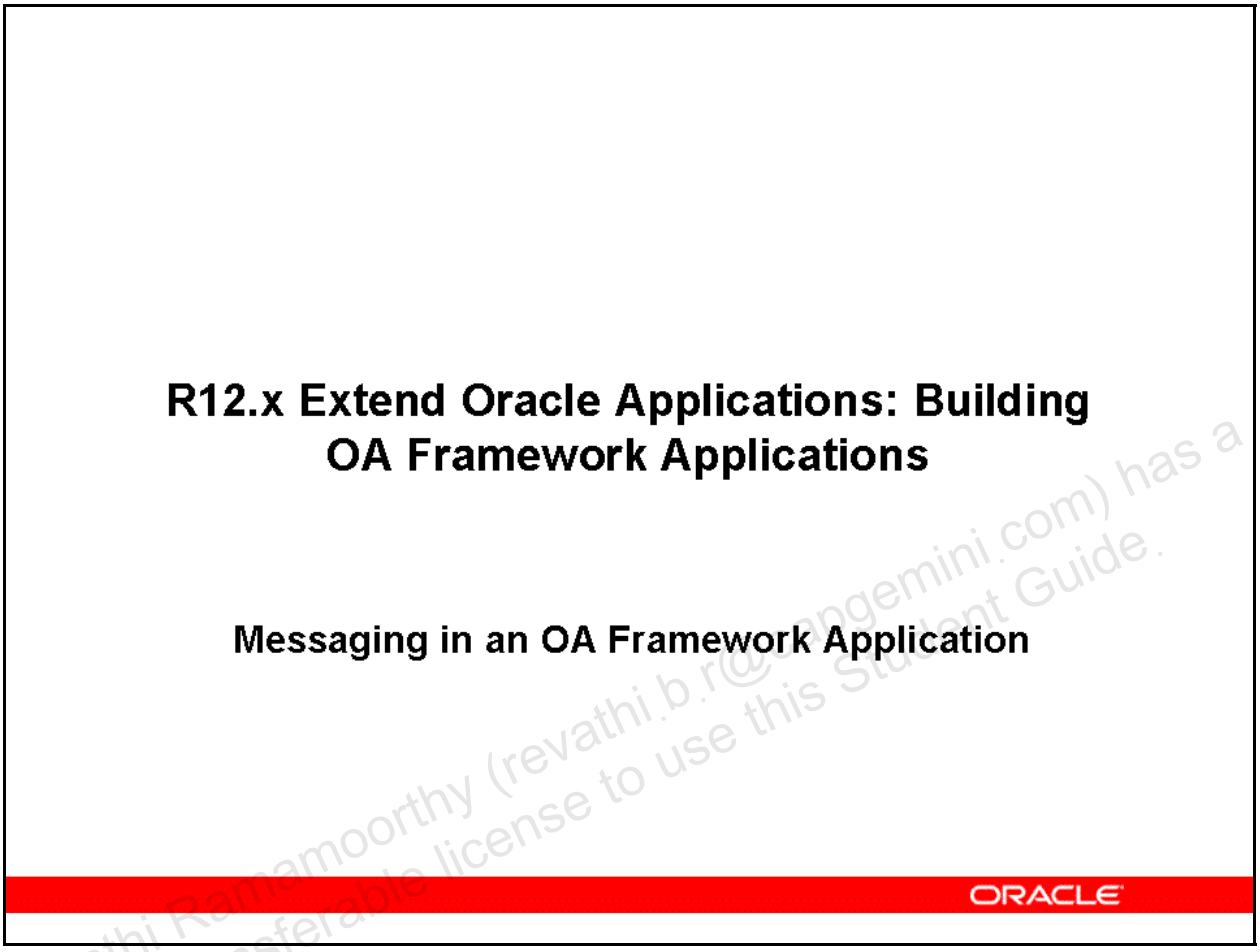
Note: This sample shows the Position field set to something other than President (Buyer in this case). Notice that when the Position is not President, there is a Manager field presented in the UI.

The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

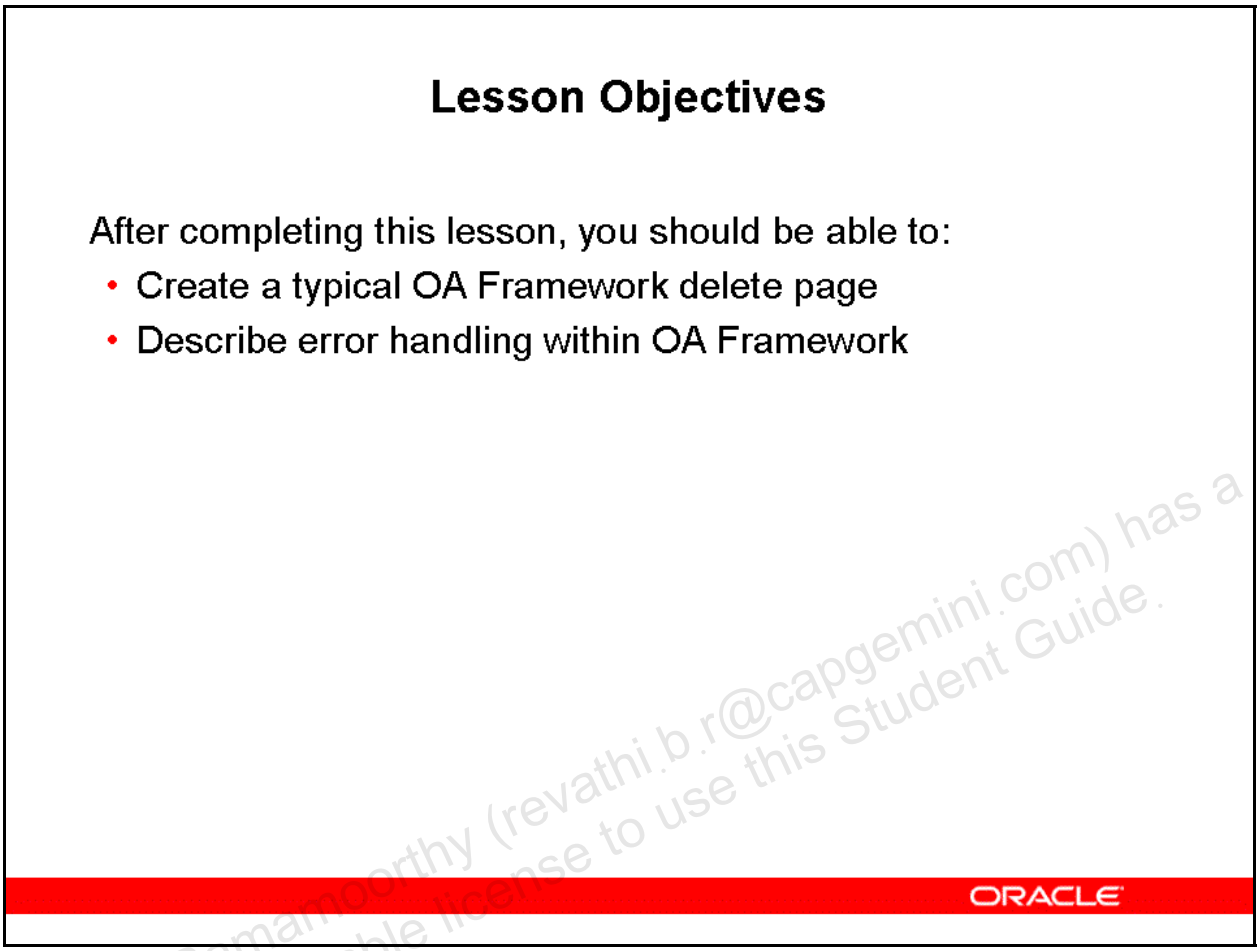


Implementing a Delete Page

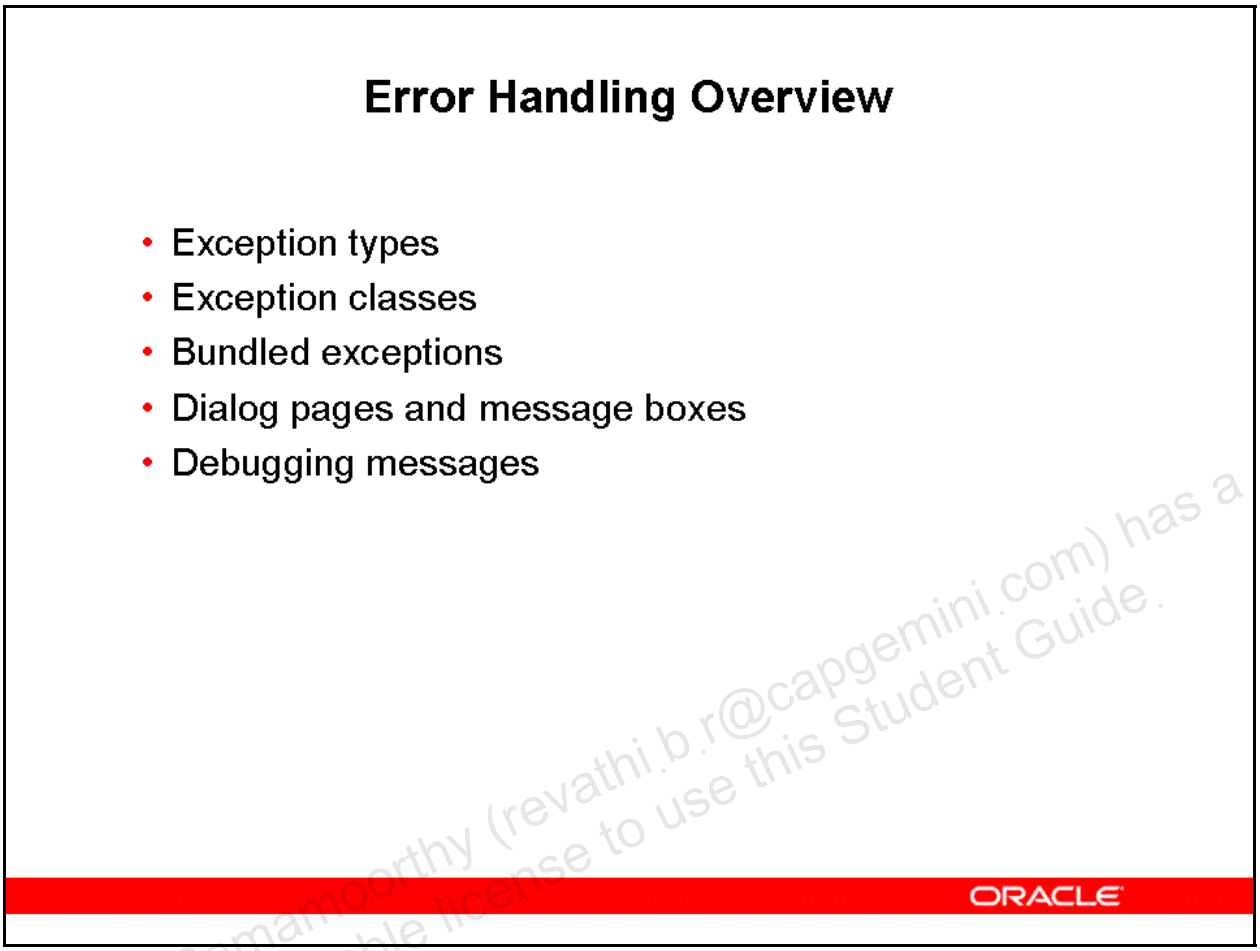
Chapter 11



Lesson Objectives

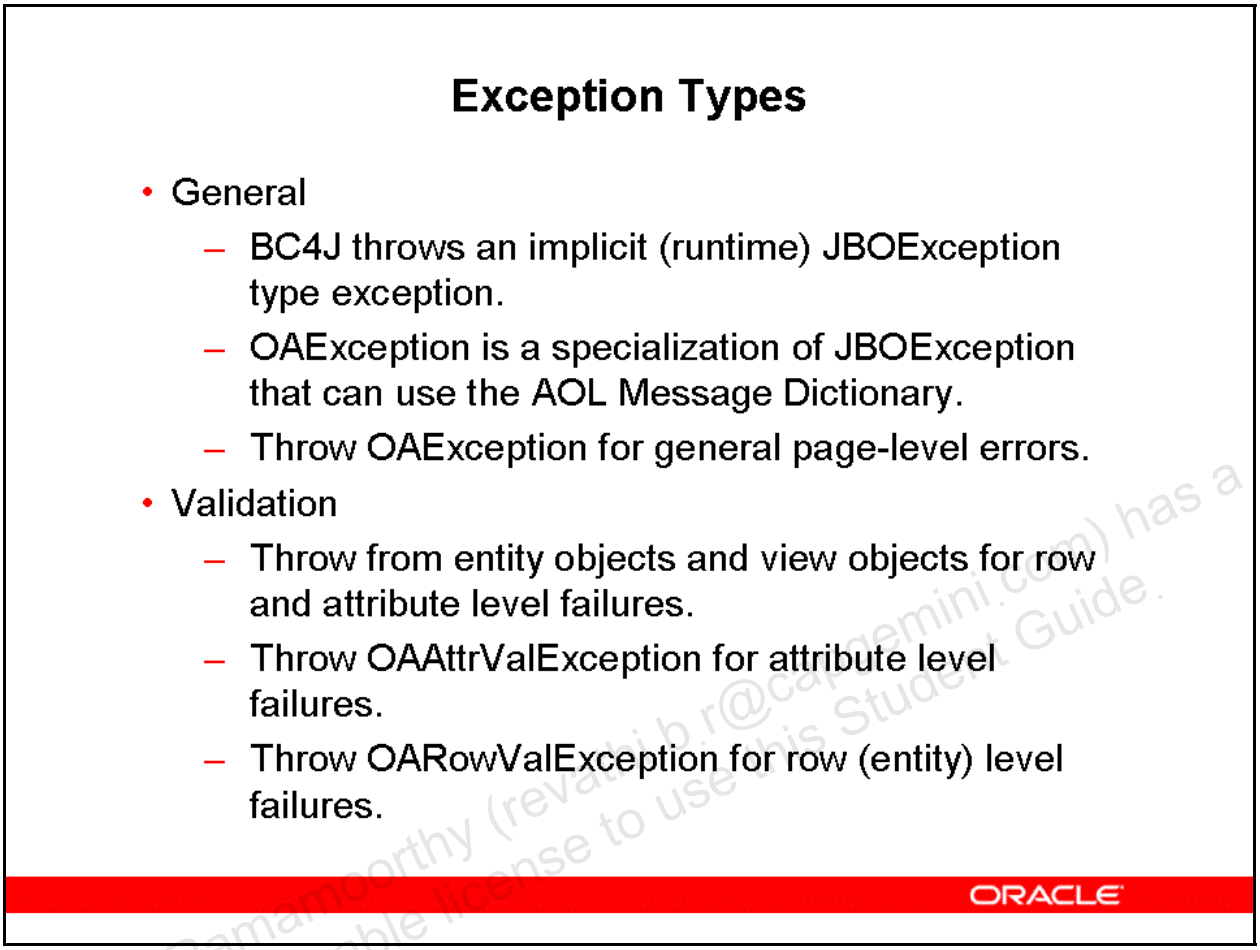


Error Handling Overview



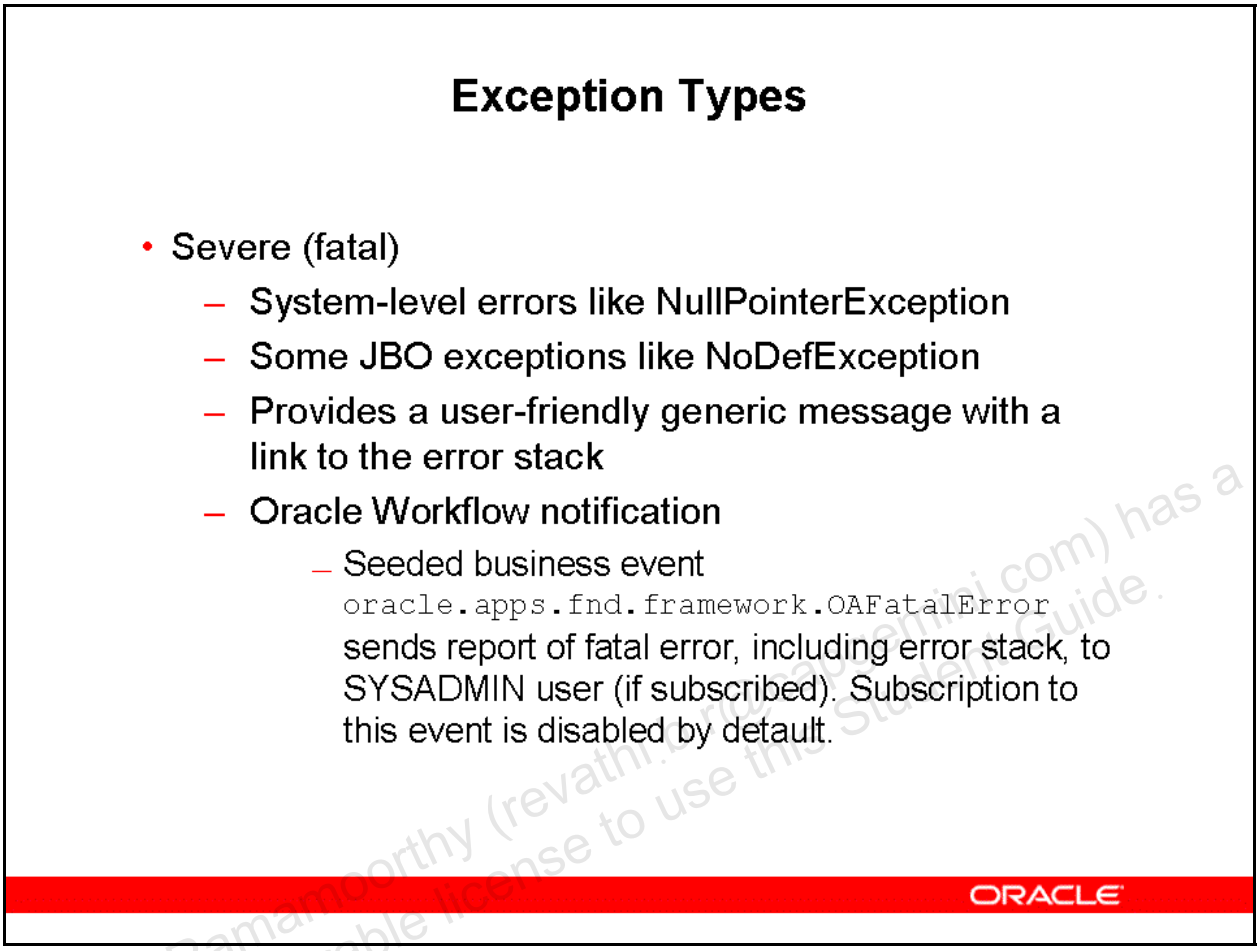
Note: It should be noted that the very nature of error handling is different in OA Framework pages, and a new programmer should have a good grasp of exception handling in Java as a foundation to this material.

Exception Types

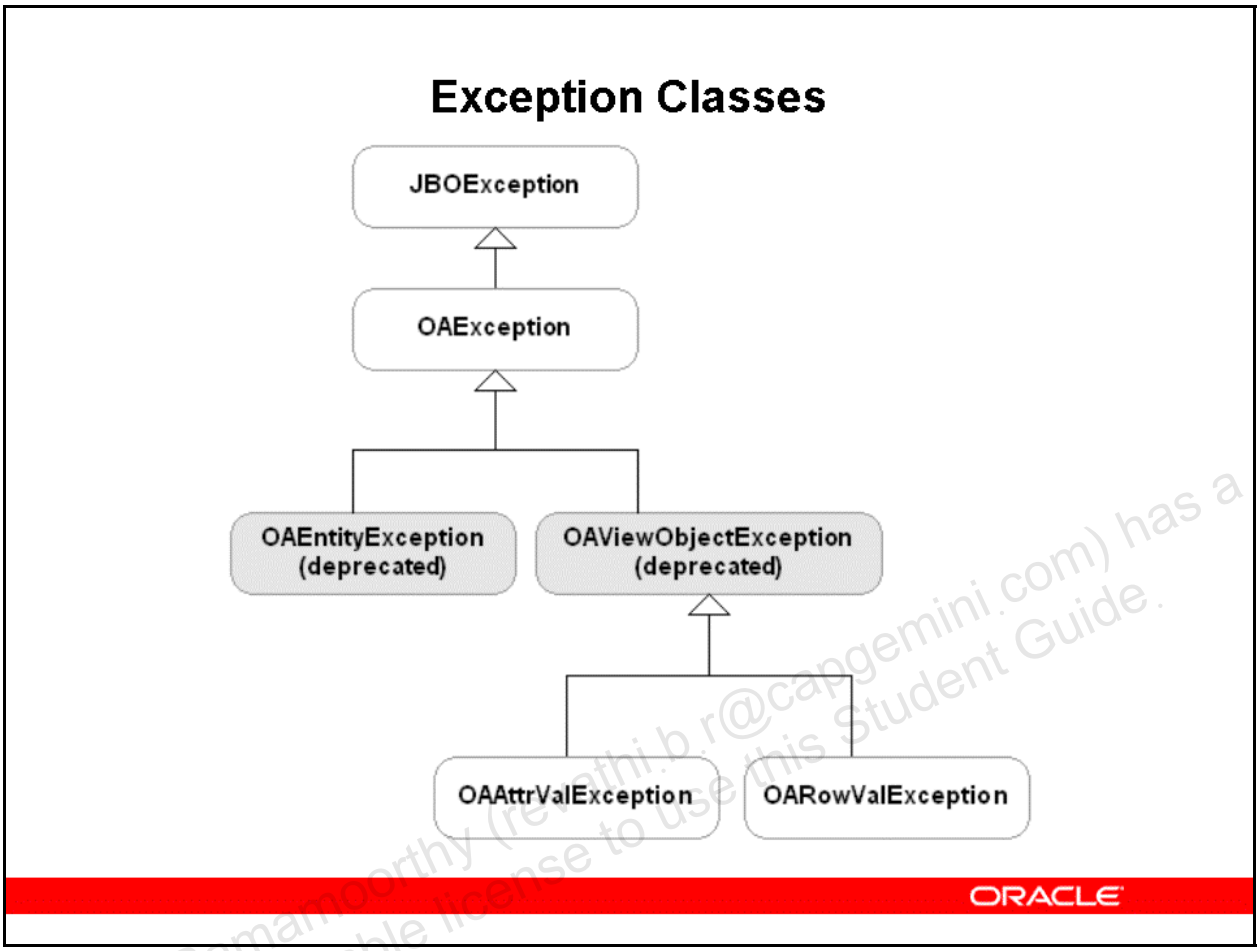


In earlier versions of OA Framework, what is now called ADFm, ADFBC, or BC4J, was called Java Business Objects or JBO. In the case of any exception that you encounter within OA Framework pages, if the exception contains jbo, it is a BC4J or model-layer problem. Likewise, the earlier versions of UIX and OA Framework web beans were called Java Rapid Application Development or JRAD. So, any exceptions with jrad in the name refer to view-layer, UIX, or UI-related problems.

Exception Types



Exception Classes

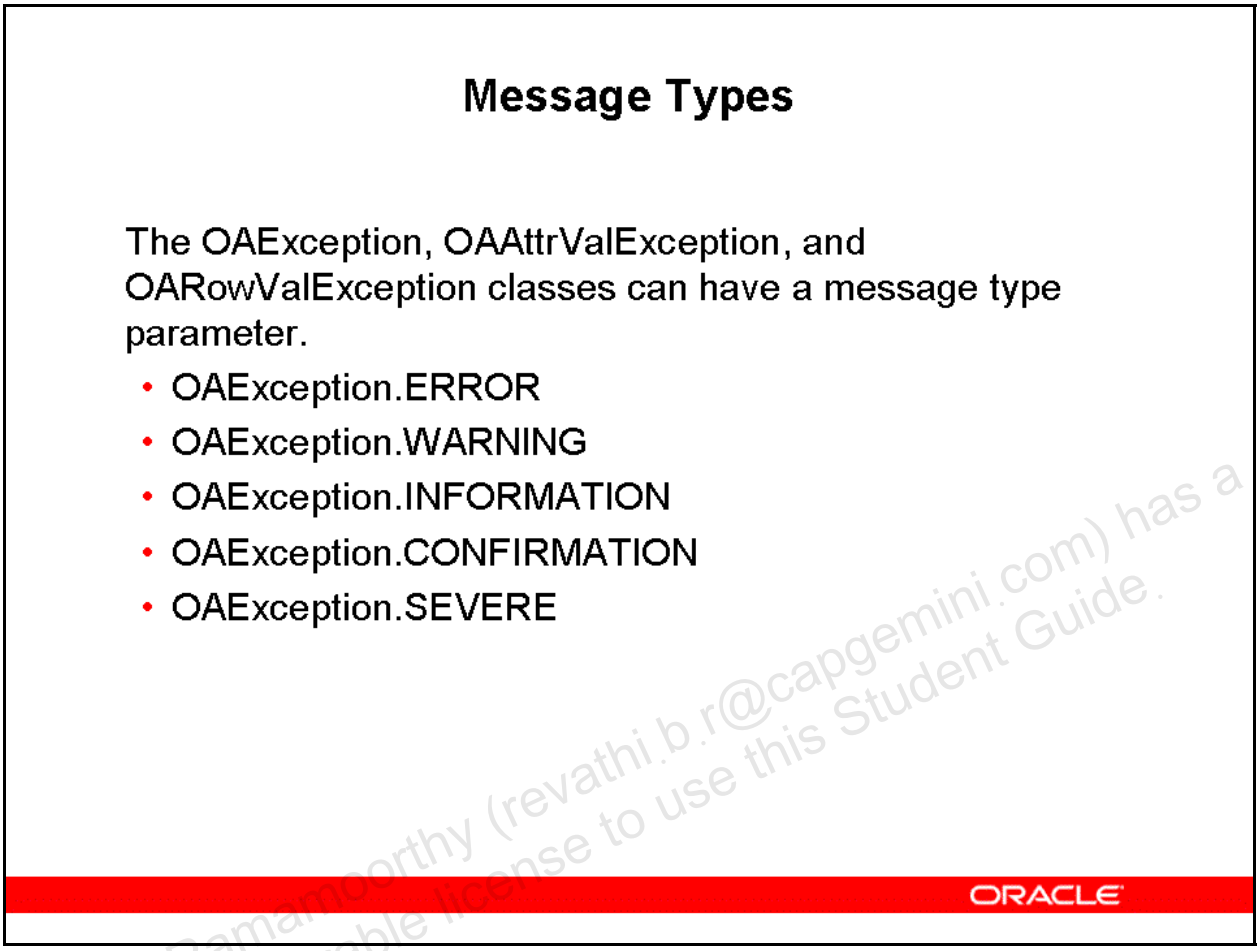


OAException is for generic error cases.

OAAttrValException is for any failed attribute-level validation in the VO row or EO.

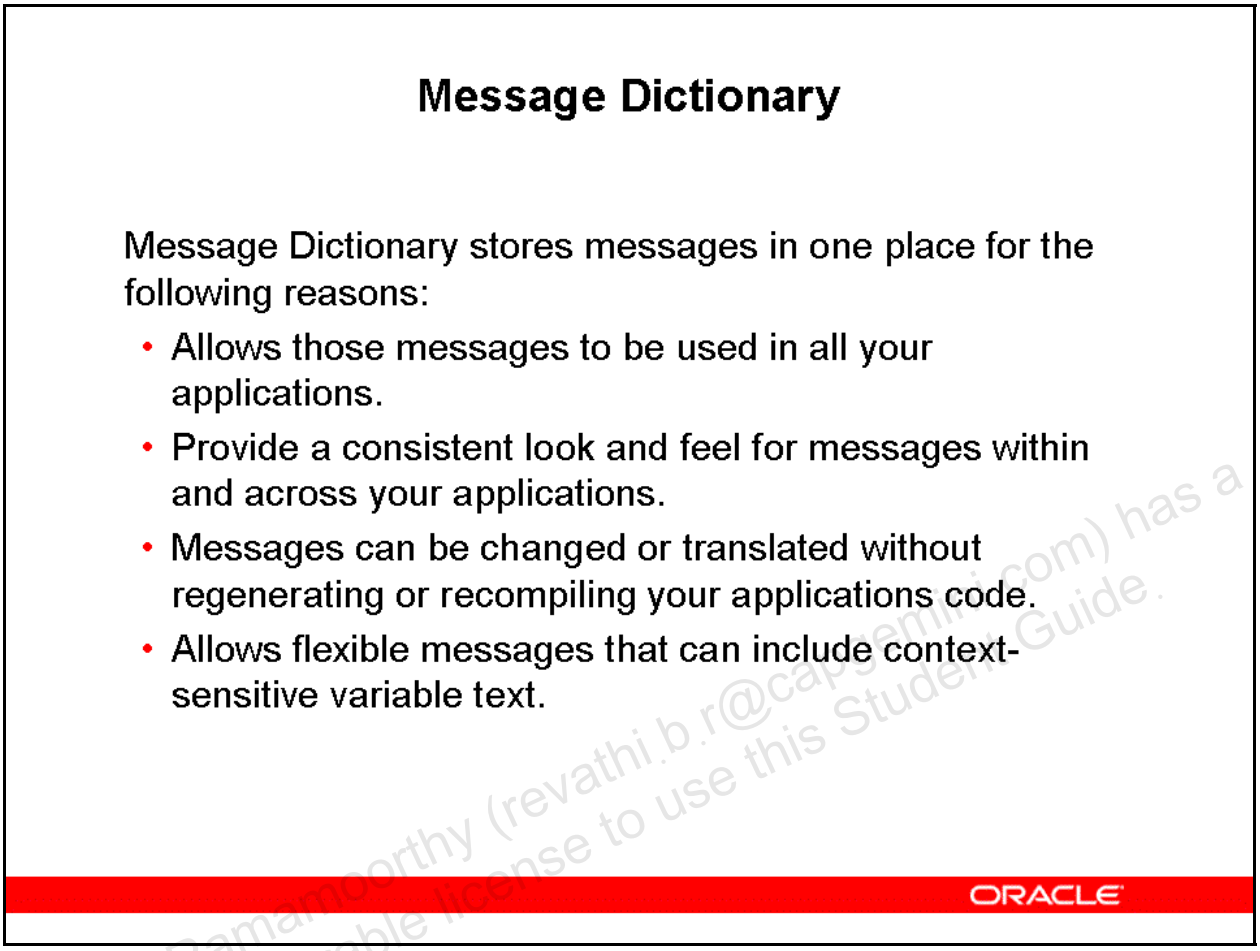
OARowValException is for any failed row-level validation in the VO row or EO.

Message Types



The OAException, OAAttrValException, and OARowValException classes all include constructors that accept a message type (byte) parameter. The message type parameter tells OA Framework the type of message to display to a user. Any of the list message types are valid for any of these exception classes.

Message Dictionary



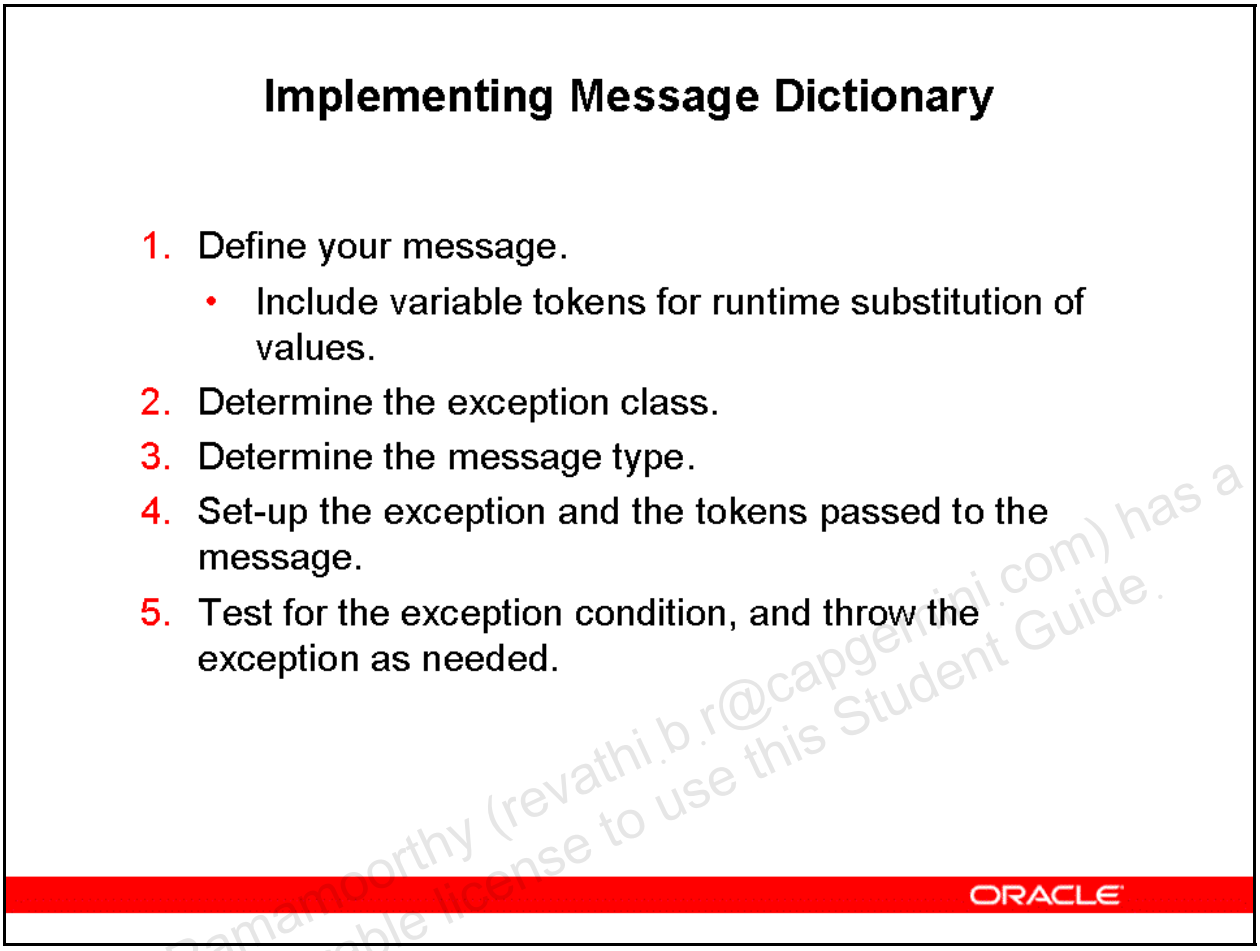
Create a message using the Messages form (Application Developer responsibility) or the Messages page (Functional Administrator responsibility)

Optionally include tokens for runtime substitution of values

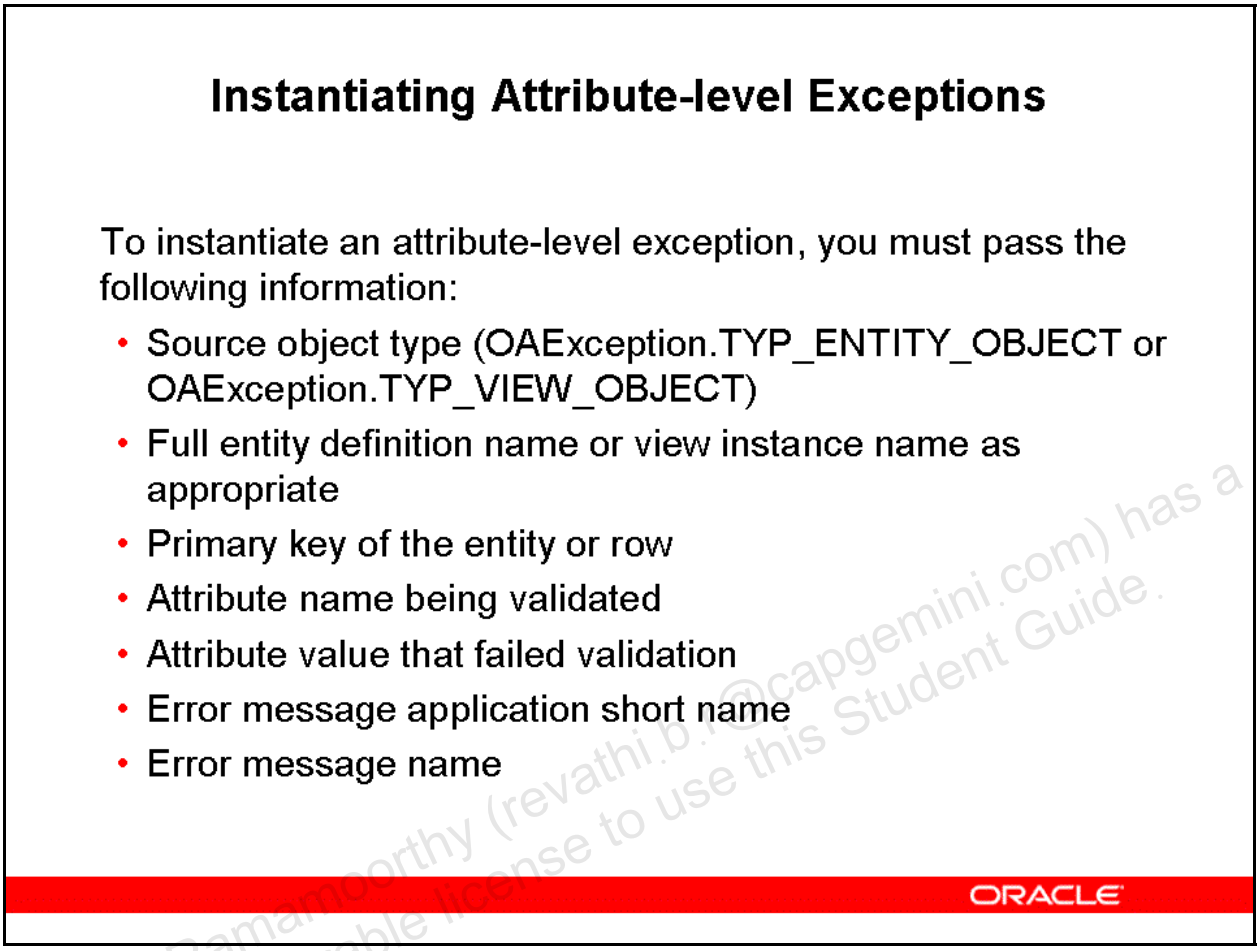
Identify the message in code by its application short name and message name

Set up and pass any tokens and their values

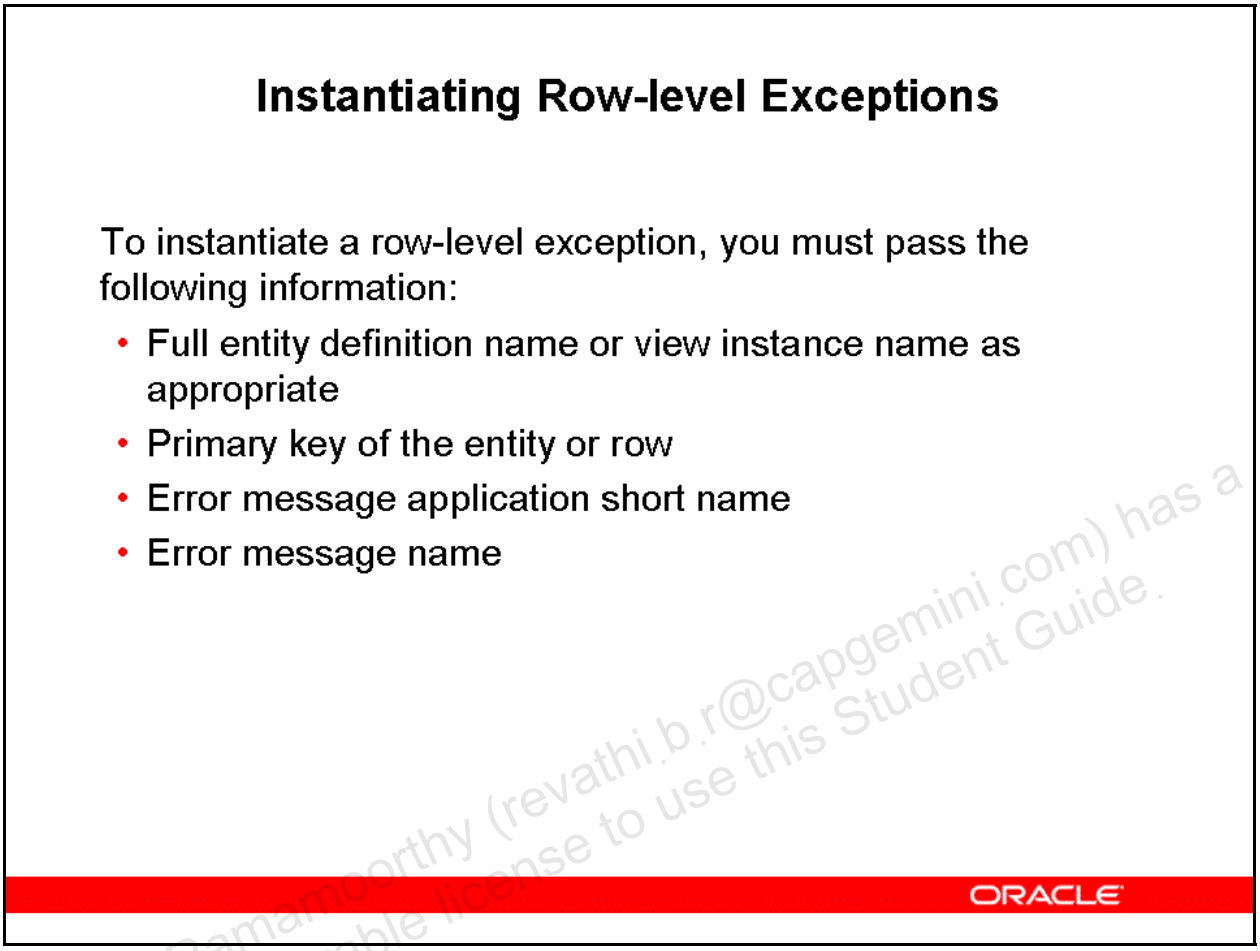
Display the message



The Message Dictionary allows us to centrally store, manage, and reuse messages. One major benefit is reduced translation costs, which may be a consideration for your company and project.

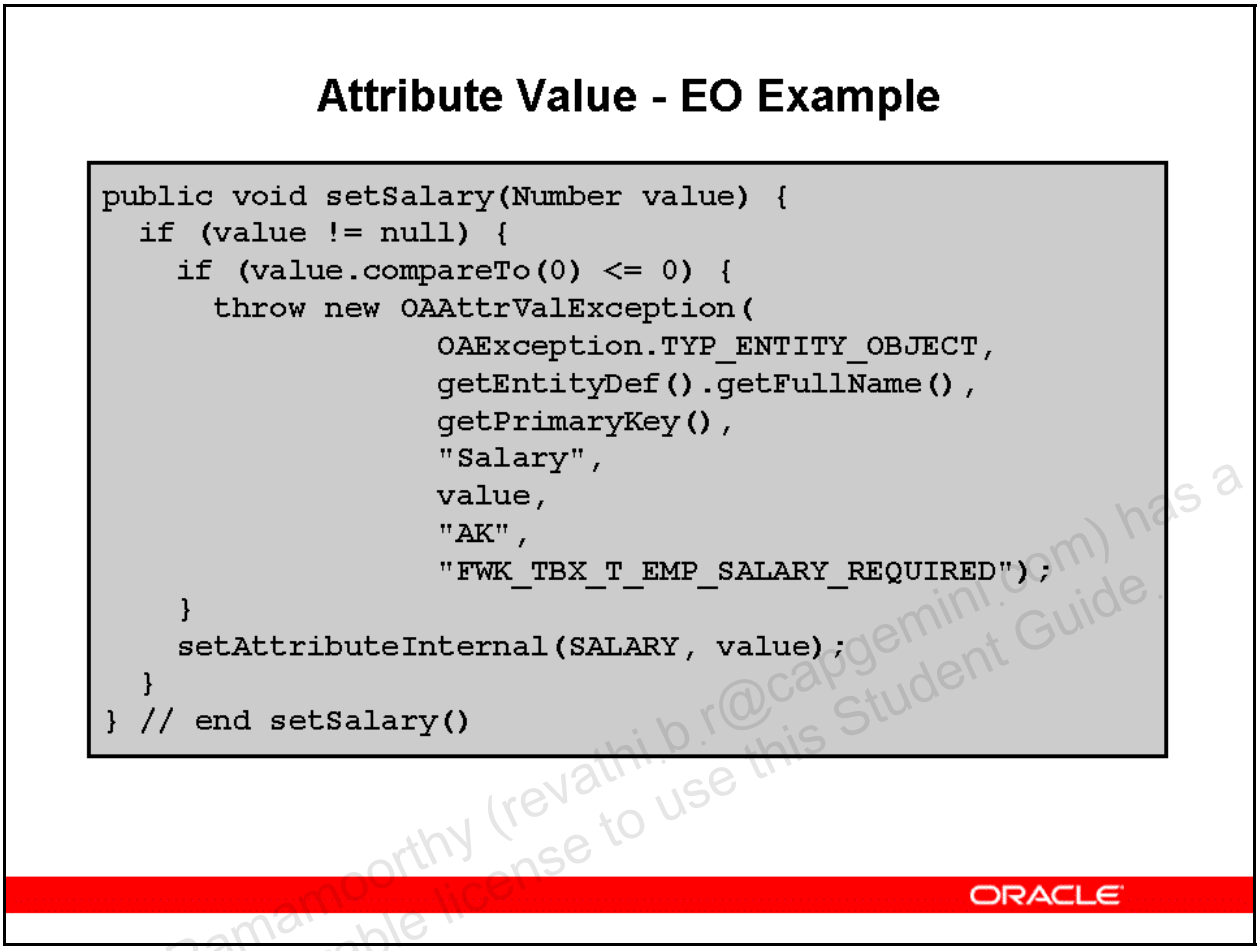


Attribute Exceptions are the messages on the individual fields of the page.



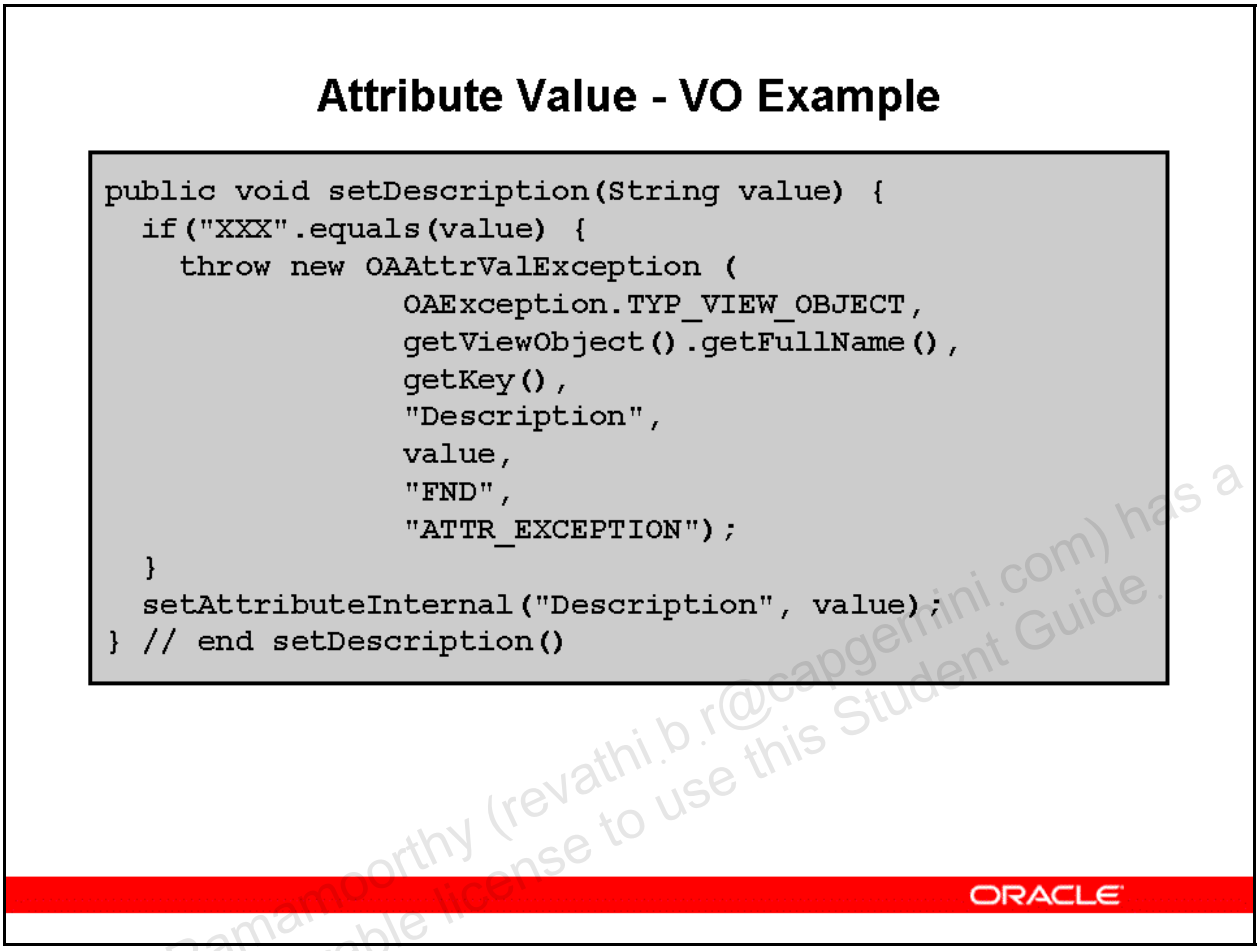
Row-level exceptions are the messages derived for validation of two or more fields on the page.

Attribute Value - EO Example



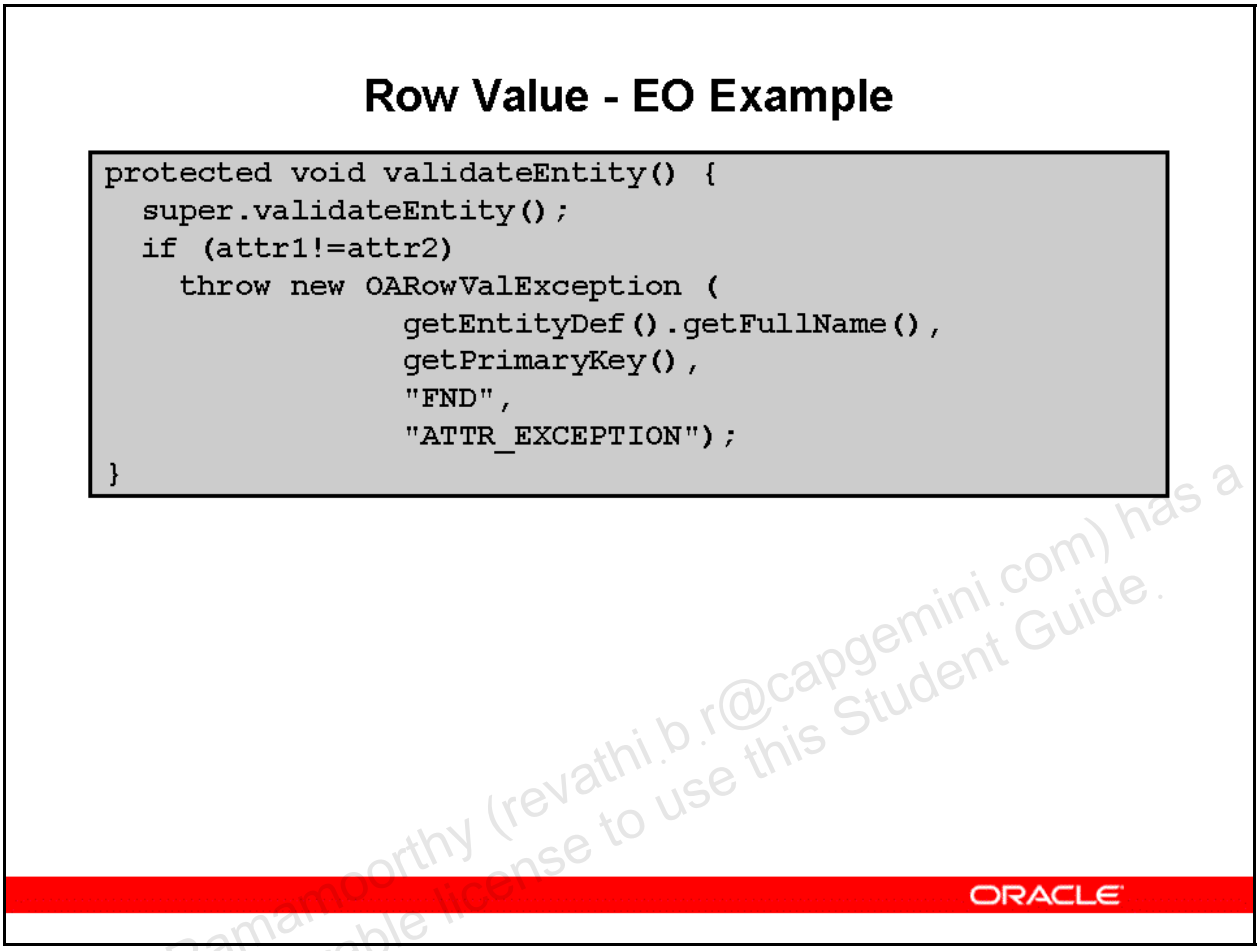
This code would be added to the EOImpl.java class, into the setSalary method.

Attribute Value - VO Example



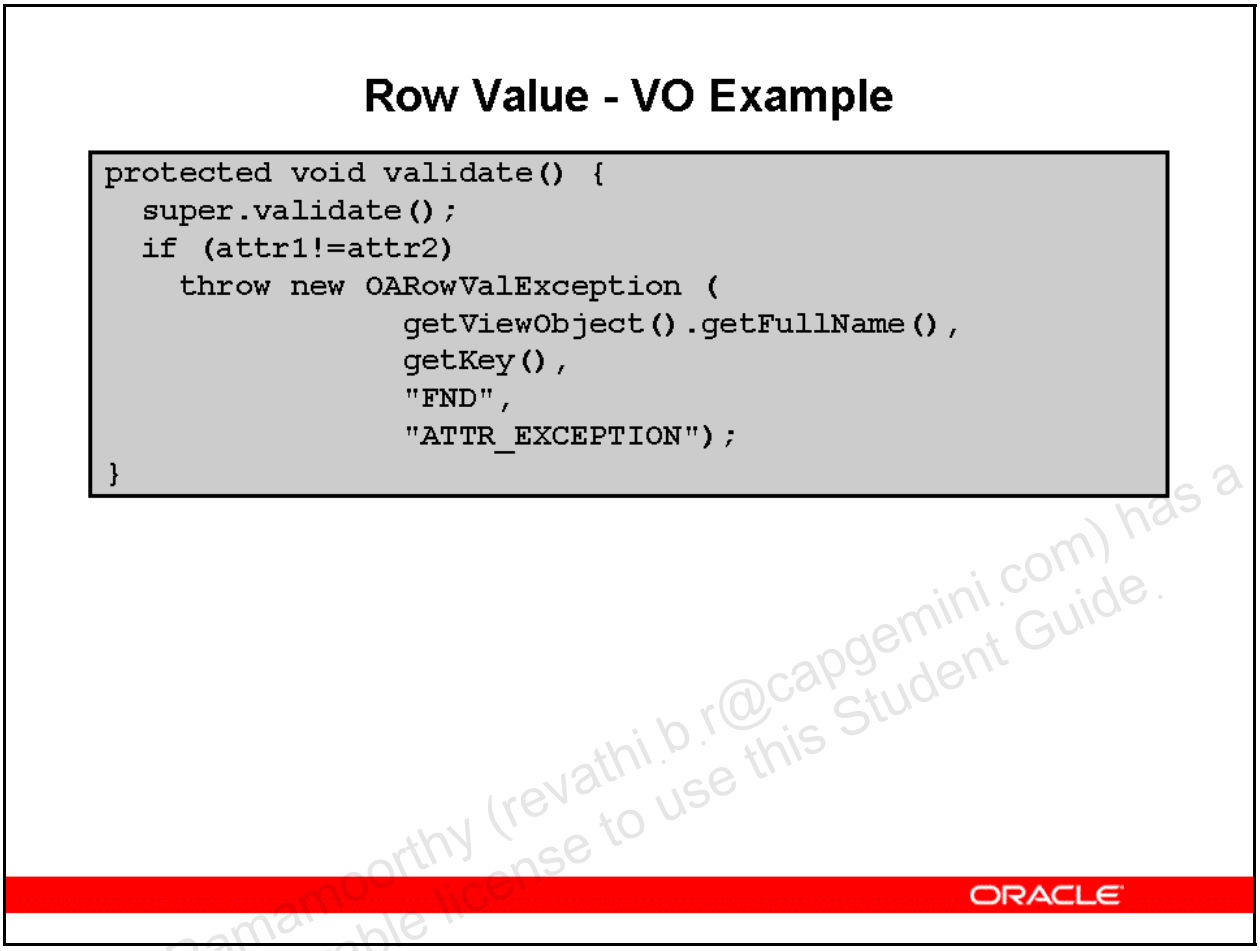
This code would be added to the VORowImpl.java class, into the setDescription method. You should NEVER use a VO set method when you have an EO-based VO. You should defer all the business logic (set methods) to the EOImpl.java class.

Row Value - EO Example

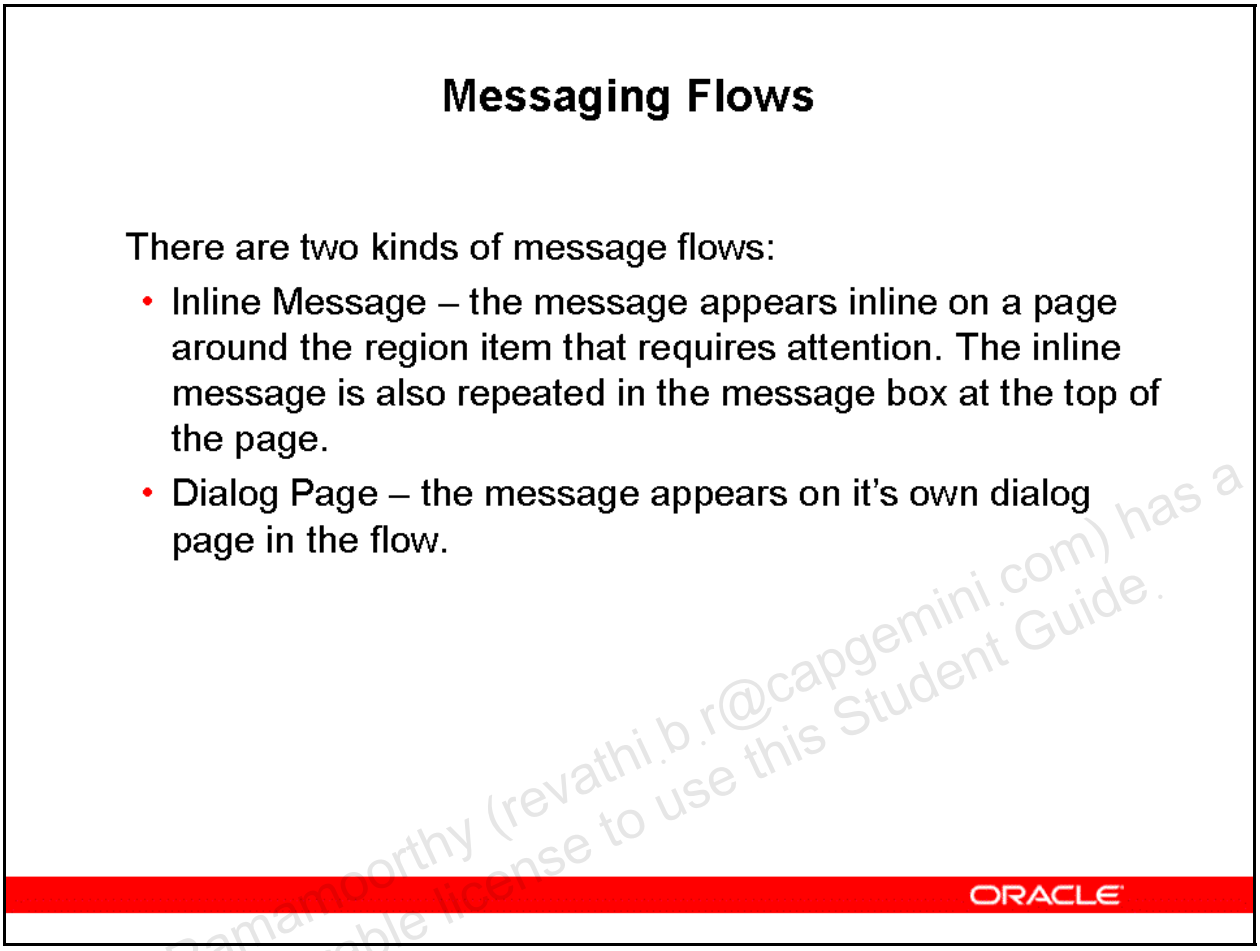


This code would be added to the EOImpl.java class. The validateEntity method is a default method provided for cross-attribute validation. This method is called after all the individual set methods have been called.

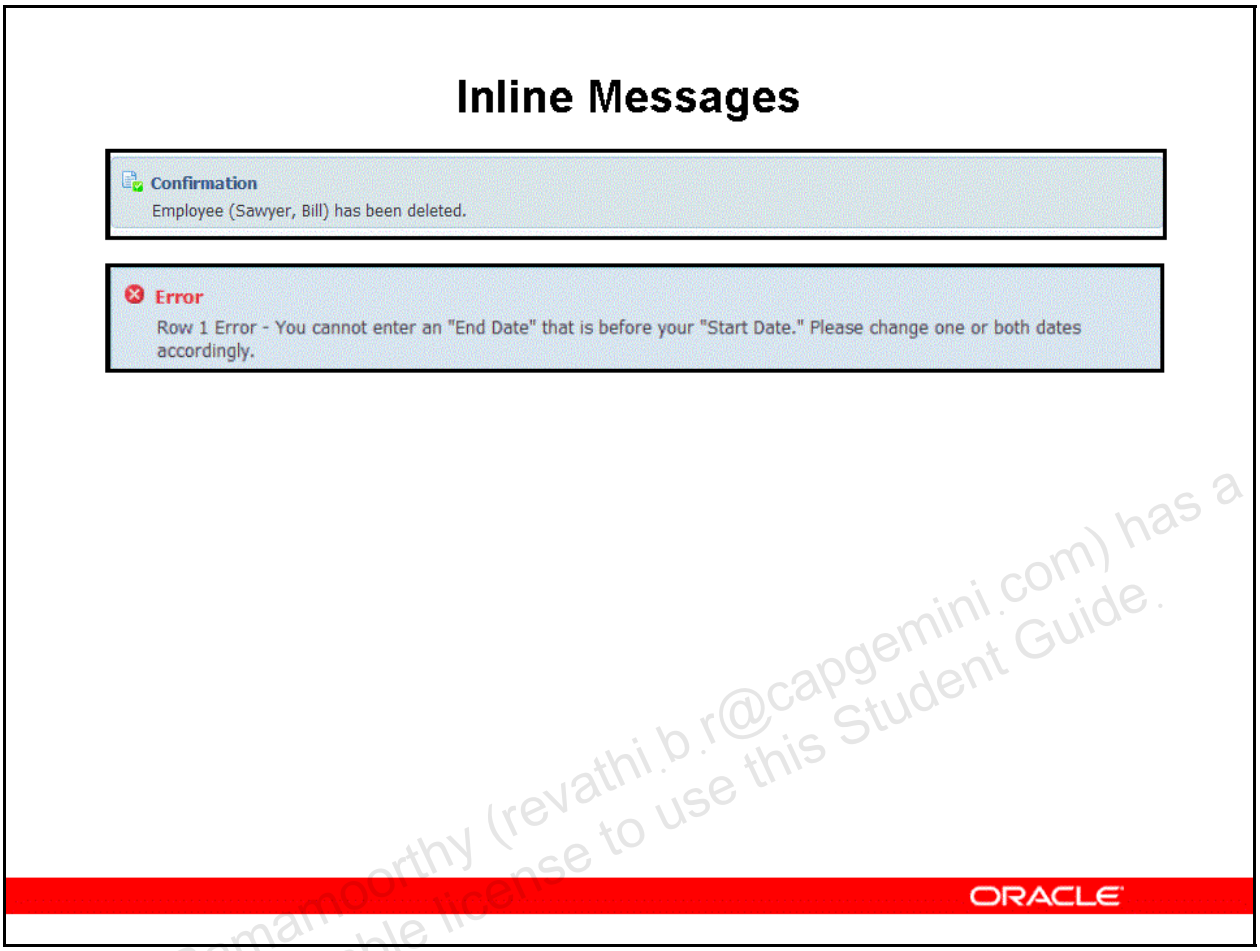
Row Value - VO Example



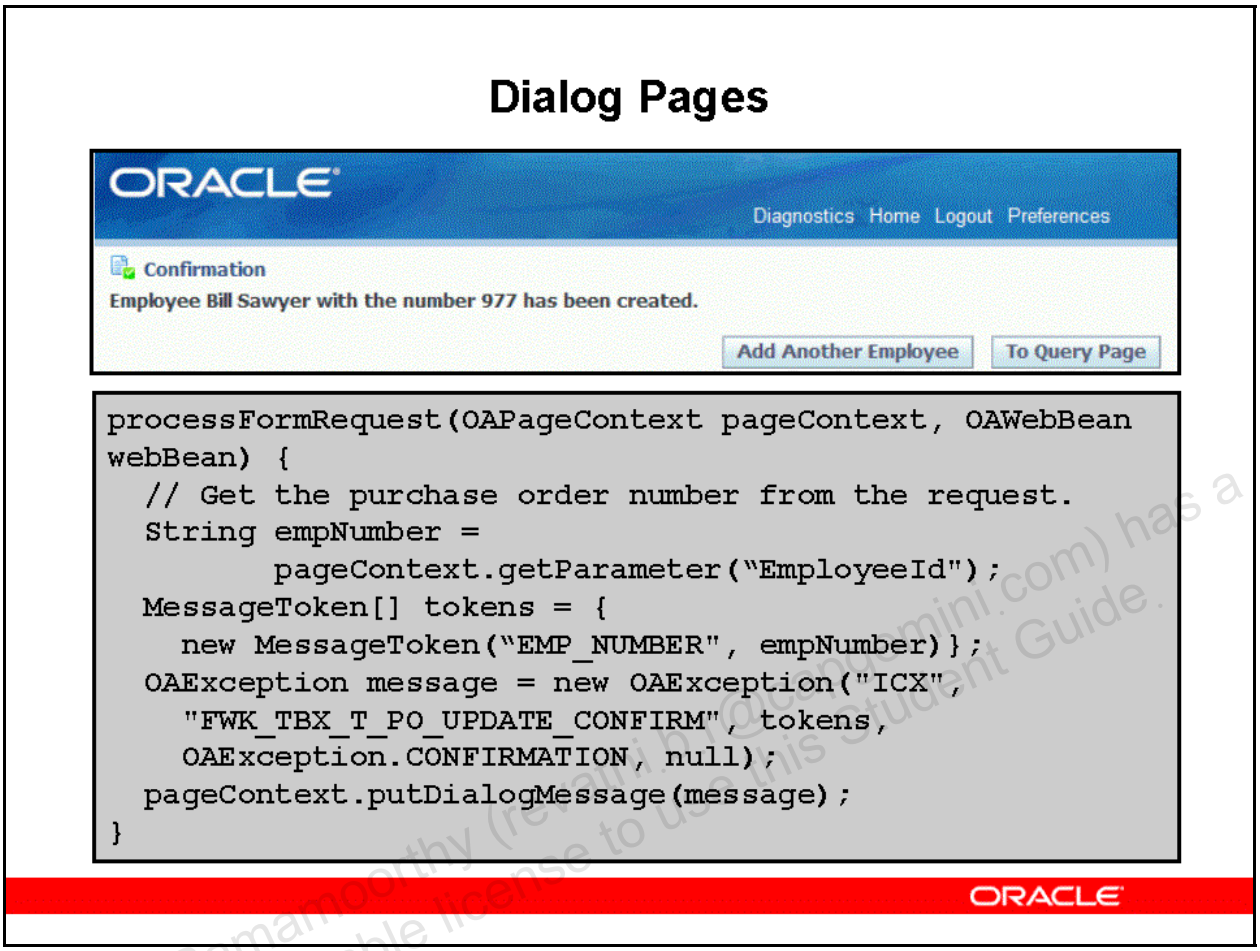
This code would be added to the VORowImpl.java class. The validate method is a default method provided for cross-attribute validation. This method is called after all the individual set methods have been called. You should NEVER use the VO validate method when you have an EO-based VO. You should defer all the business logic to the validateEntity method of the EOImpl.java class.



Inline Messages



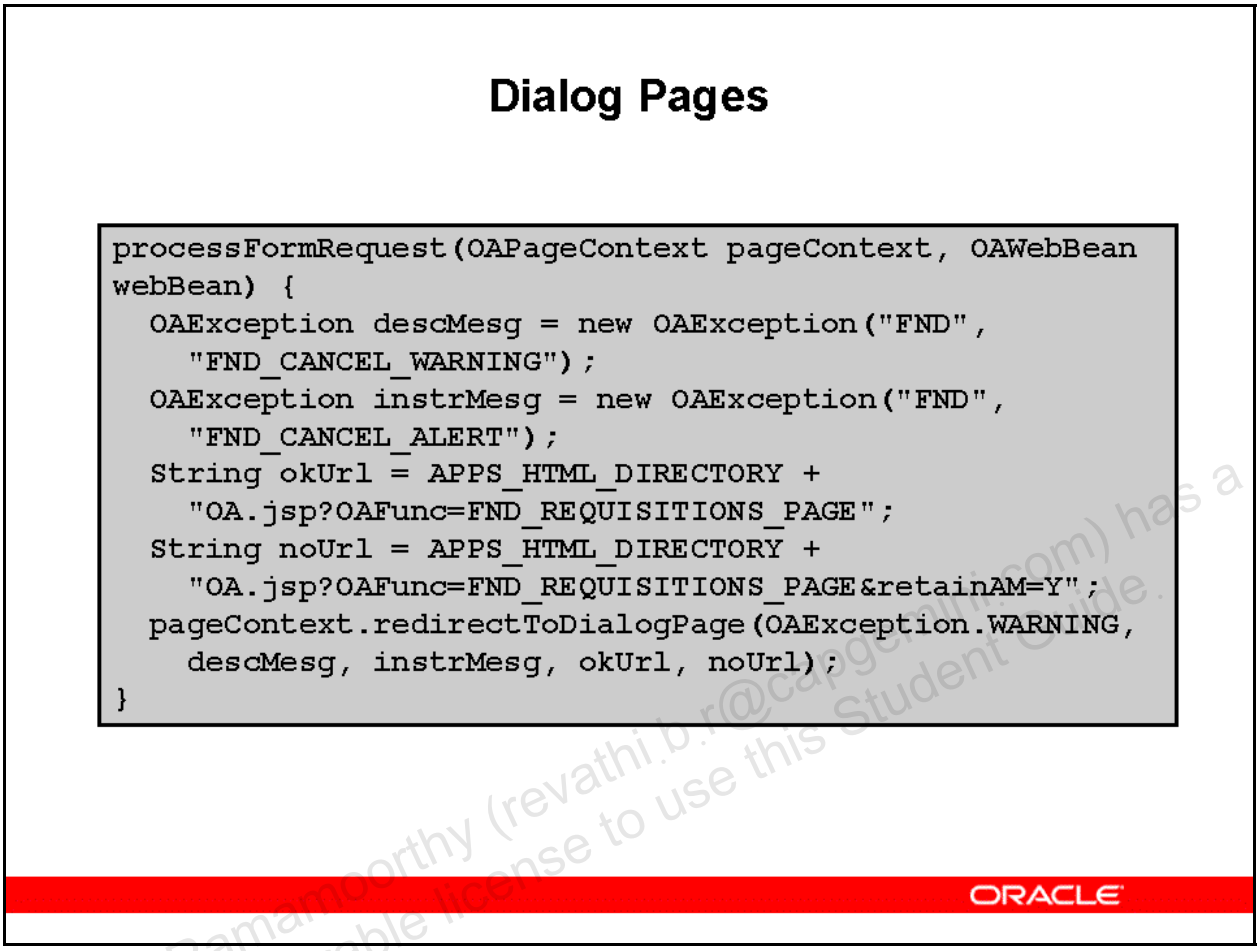
Dialog Pages



Browser-based Dialog pages are not true dialog pages. Dialog pages in your operating system cannot be overridden, and the user must respond to that window. In browser-based dialog pages, the dialog page is just like any other page. It can be closed without the user responding to it. As such, unexpected behaviors can occur. This is the nature of web-based applications, and must be anticipated by the programmer.

This code is contained within the CO.java (Controller) class, in the processFormRequest method.

Dialog Pages



This code is contained within the CO.java (Controller) class, in the ProcessFormRequest method.

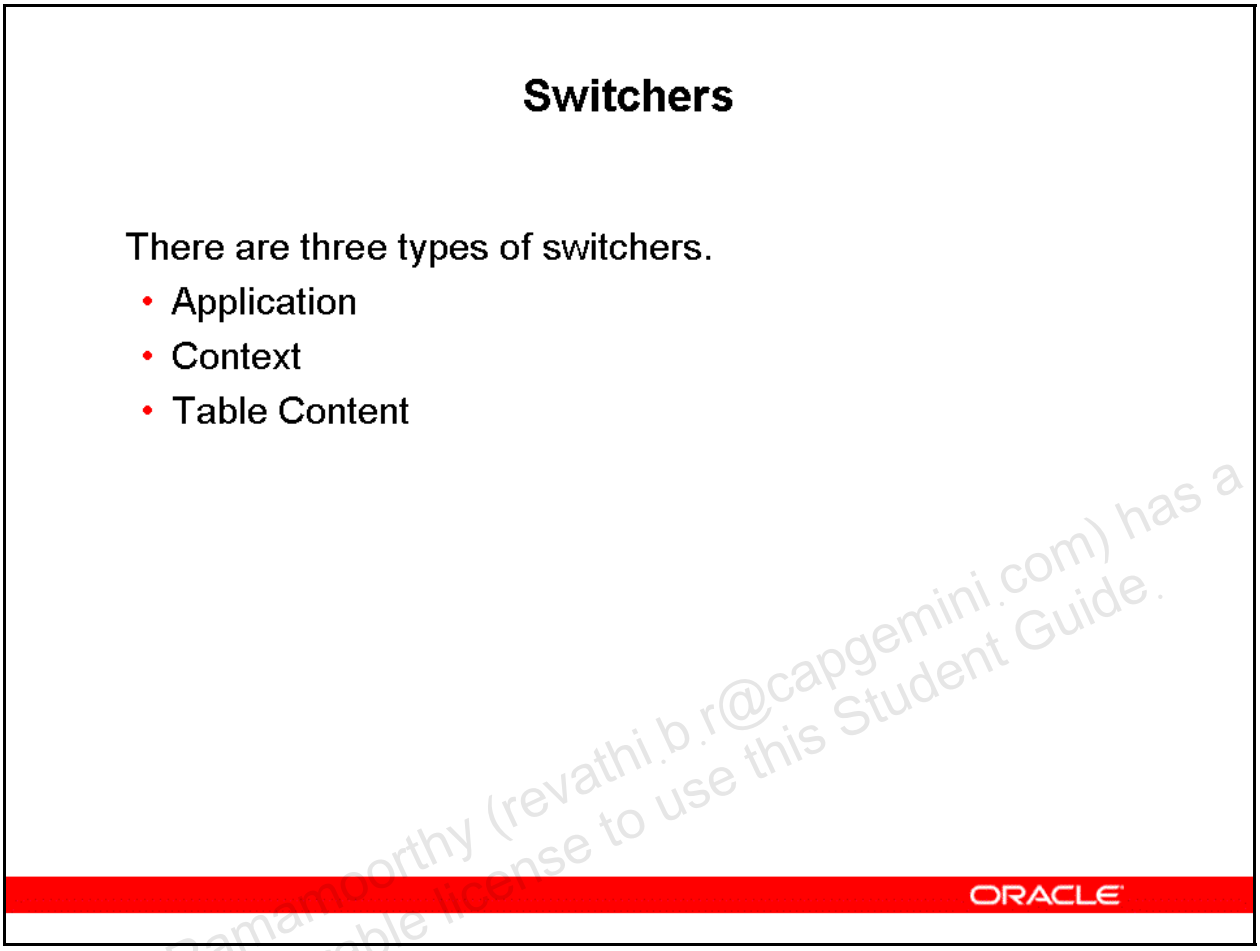
descMesg - concise message describing complex error, warning, information, or confirmation

instrMesg - further concise instructions suggesting how to recover and proceed with the task at hand

okUrl - URL value for the OK (or Yes) action button

noUrl - URL value for the NO action button

Switchers



Applications switchers allow you to switch from the home/default page of one application to the home/default page of a different application. It is a rapid navigation aid.

Context switchers allow you to quickly change the parameters of an LOV. For example, assume you are a VP, and you have several groups that report to you. You are looking at the headcount of one of your groups (Group A) presented in a table. A context switcher provides a quick poplist-type UI element to allow you to switch the context to another group (Group B). When you select the different context, the data presented in the table is immediately changed.

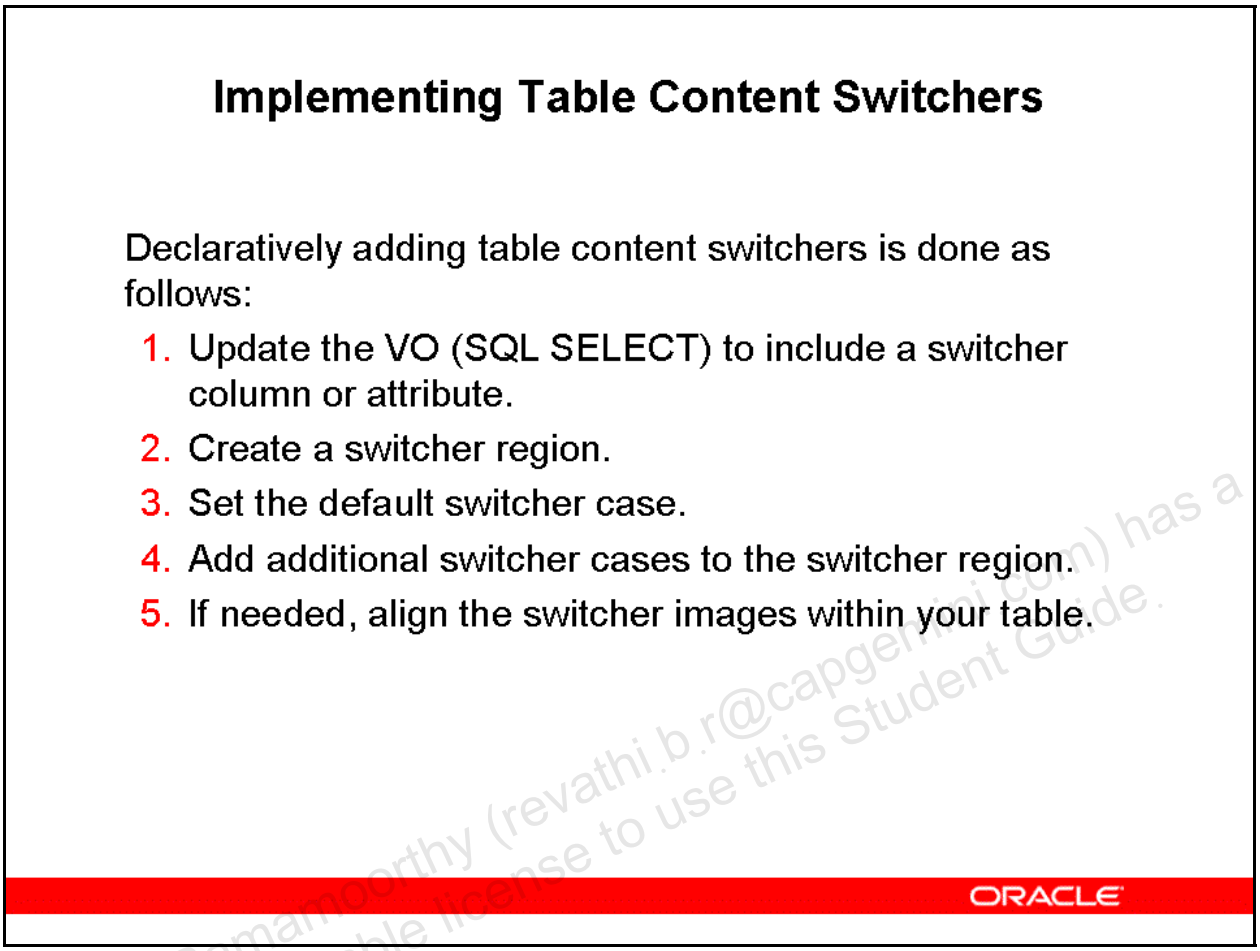
Table content switchers are the focus on the lab for this lesson. A table content switcher is a region with two or more display alternatives. The display alternatives are predefined items of which only one is selectively rendered at any given time.

Table Content Switcher Abilities and Limits

Table content switchers can have the following:

- Column header labels set by modifying the Prompt property for each Switcher nested region item.
- Sorting for the item set by modifying the Initial Sort Sequence property for each Switcher nested region item.
- Limit the use of Switchers to within tables, advanced tables, or HGrids, particularly when you want to switch between different kinds of web beans, such as different images.
- Switchers can be used outside a table, but it is recommended to use SPEL binding for the Rendered property of the content that you want to conditionally display.

ORACLE



Step 1: The Switcher attribute must return the name of the conditionally displayed item or region to render. Remember that a Switcher region can contain two or more nested region items as display alternatives. You can add this "Switcher" attribute to your view object by including a DECODE statement in your SELECT statement. The DECODE statement determines which child item name to return.

For example, the underlying query includes the following DECODE statement to determine whether an employee can be deleted based on the employee's status:

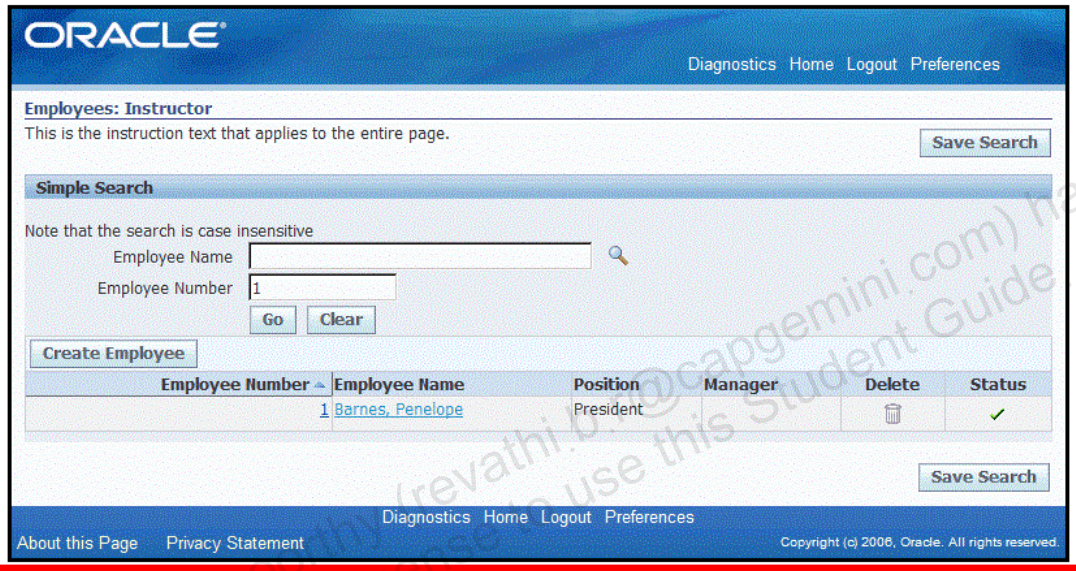
```
decode(nvl(to_char(EmployeeEO.END_DATE), 'N'), 'N','DeleteDisabled',
    'DeleteEnabled') AS DELETE_SWITCHER
```

Step 5: You only need to manually align images when you are using a classic table. Most tables are implement using advanced tables.

Query Page with Non-Deleteable Employee

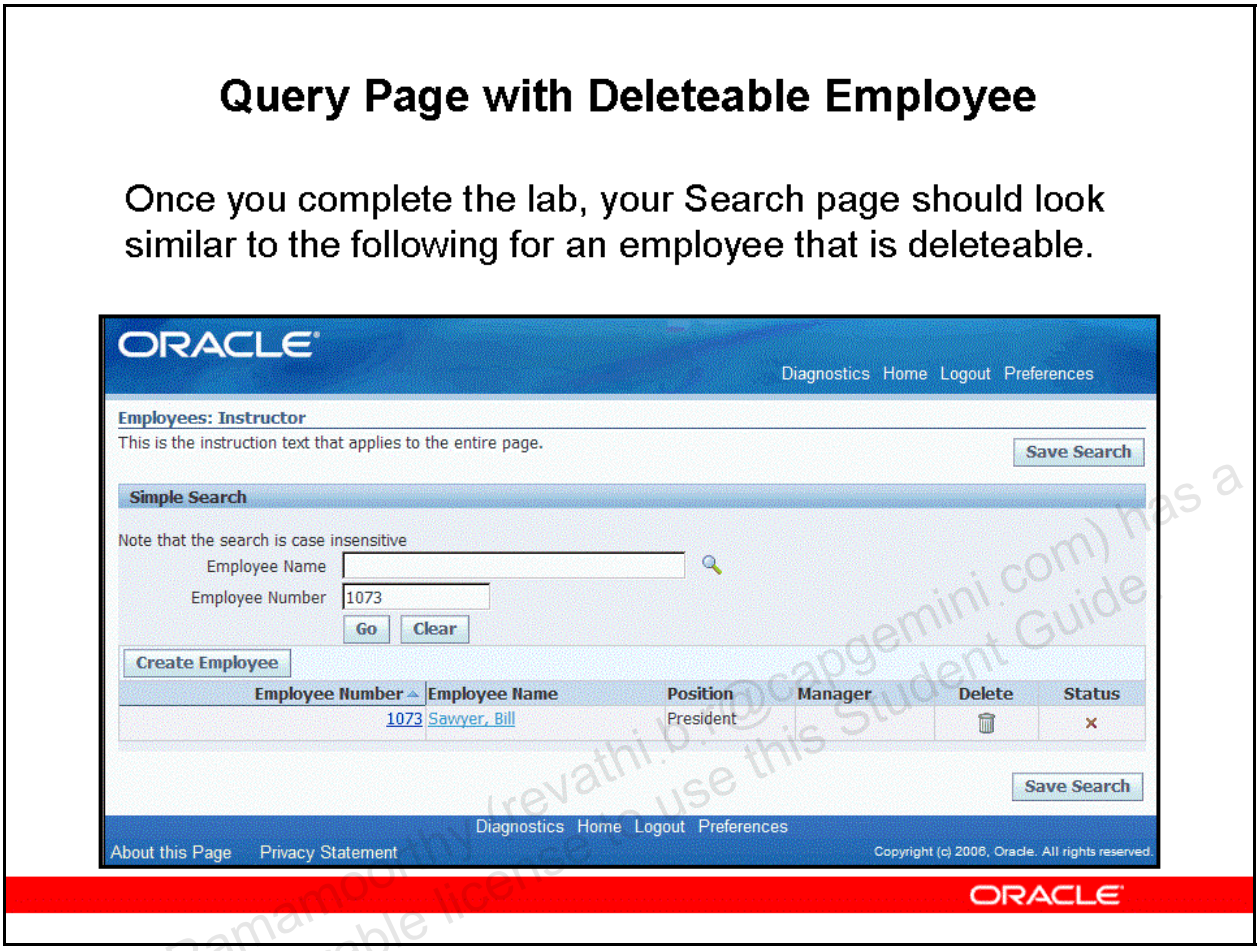
Query Page with Non-Deleteable Employee

Once you complete the lab, your Search page should look similar to the following for an employee that is not deleteable.



The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Query Page with Deleteable Employee



The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

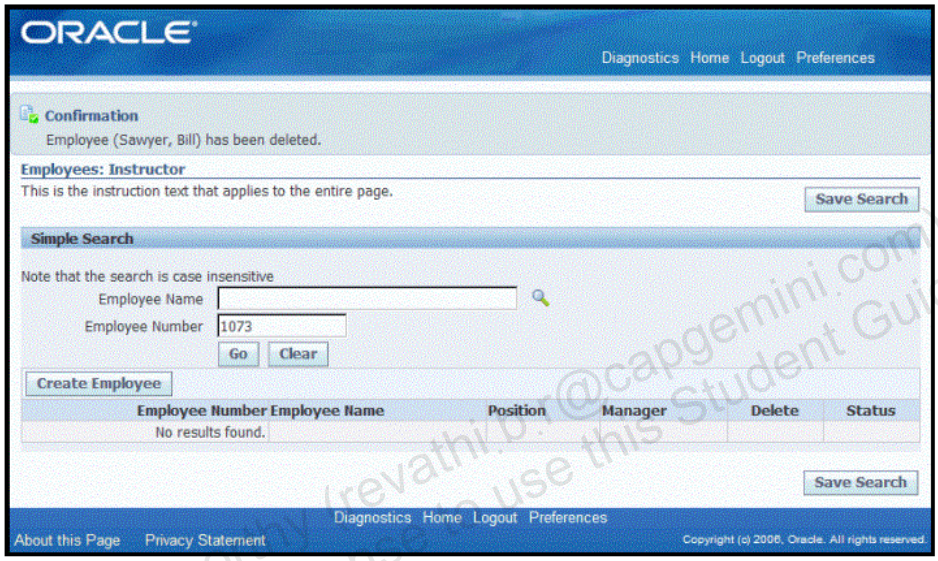
Warning Dialog

The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

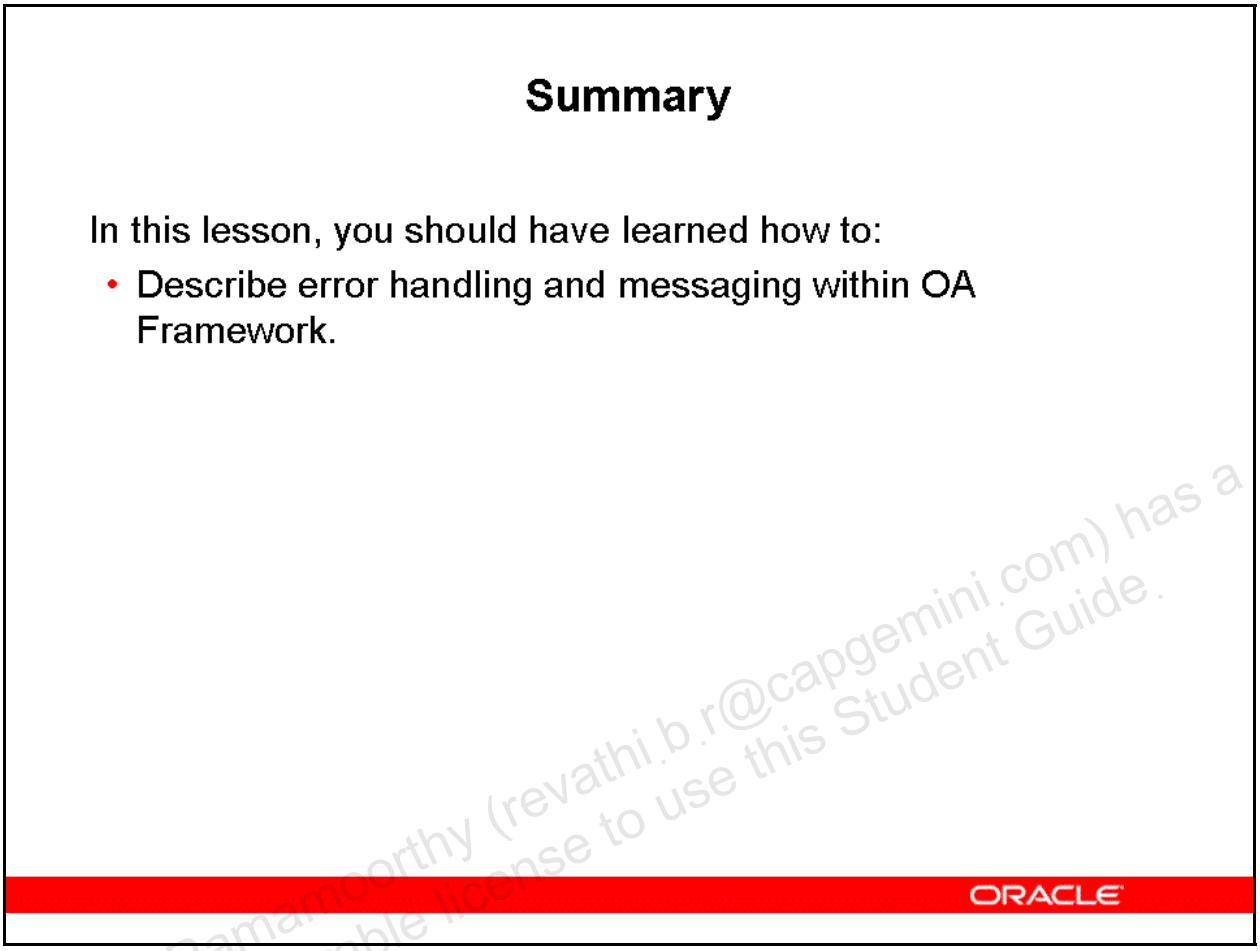
Confirmation Message

Confirmation Message

Once you complete the lab, your Search page should provide a confirmation similar to the following of a success deletion.

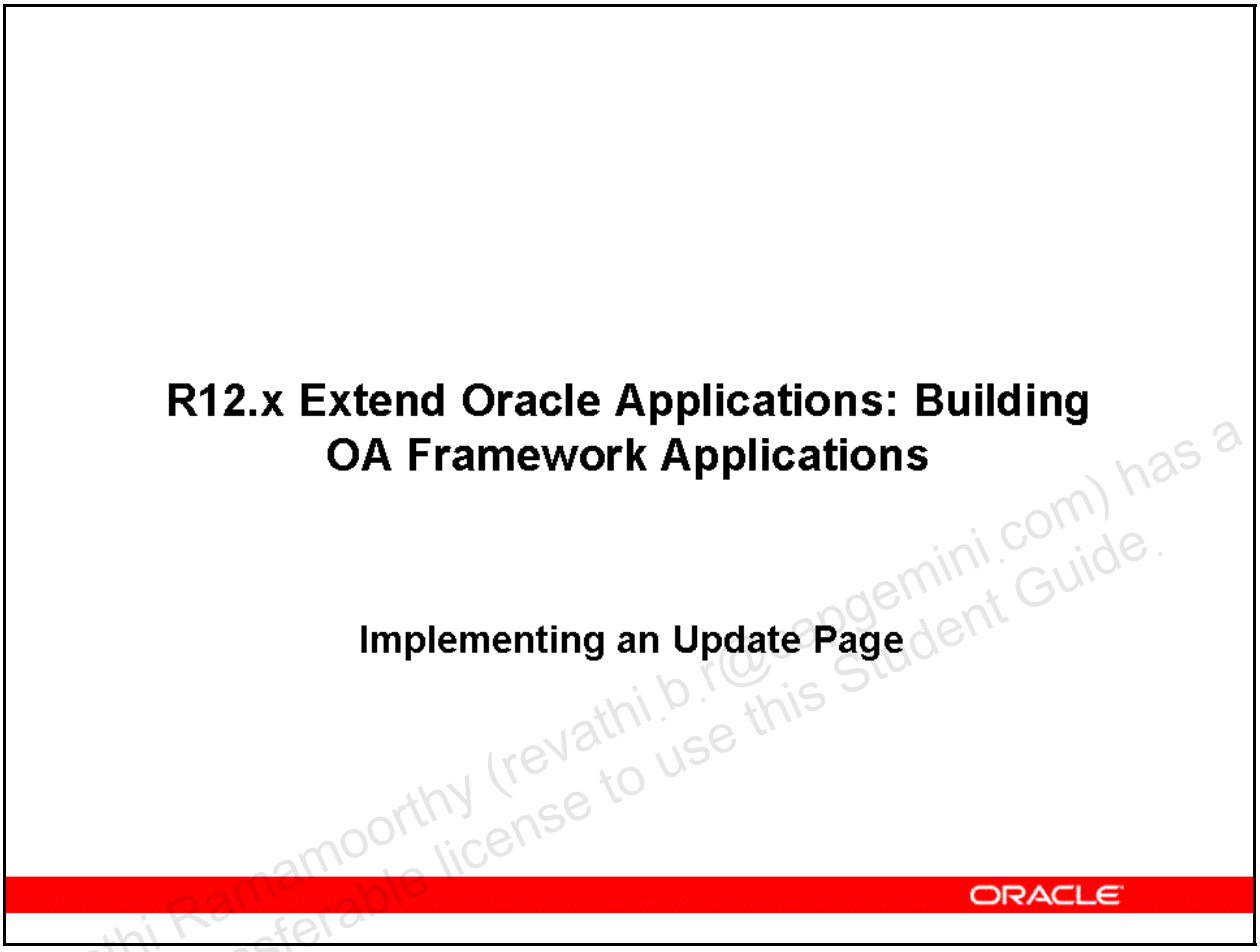


The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

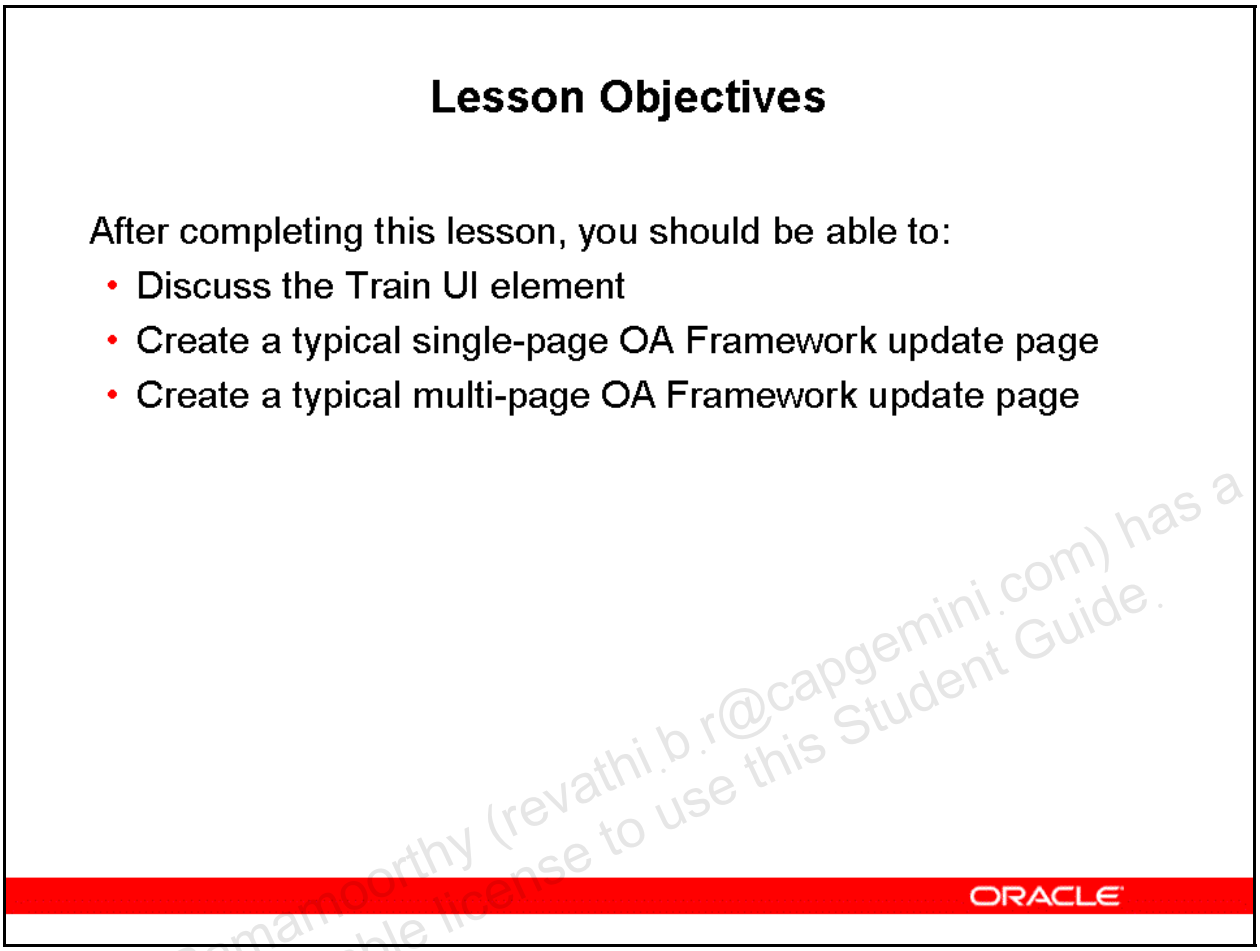


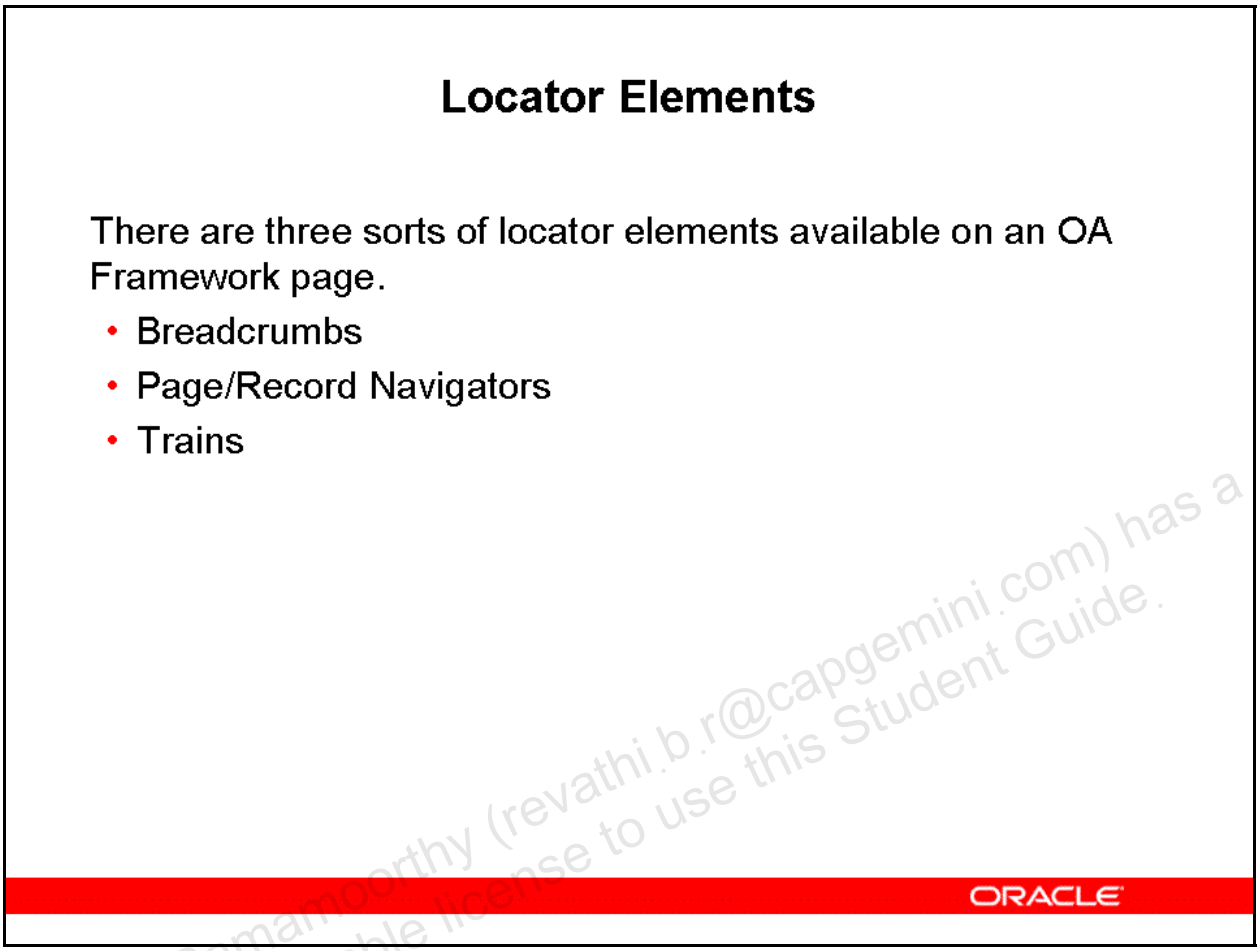
Implementing an Update Page

Chapter 12

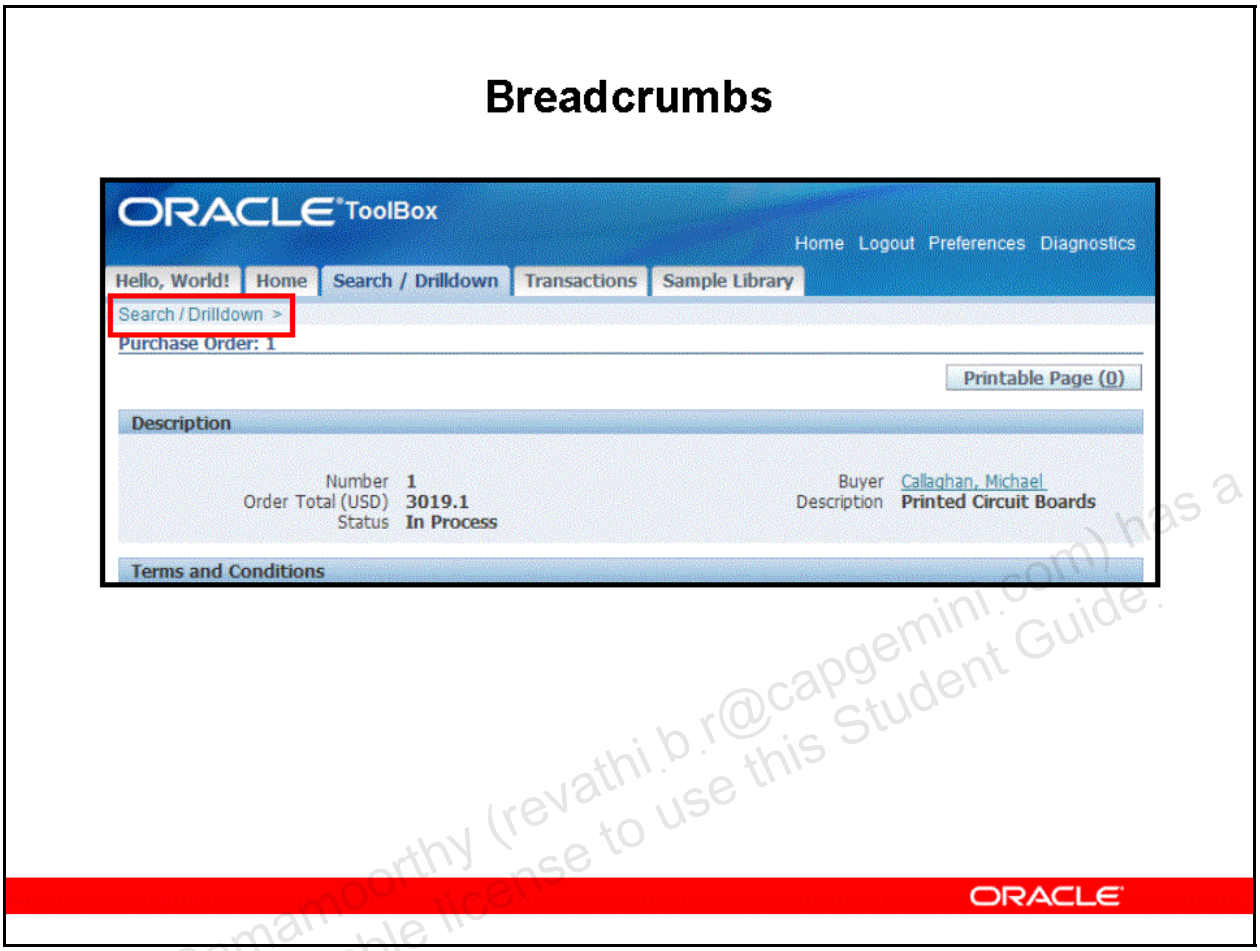


Lesson Objectives



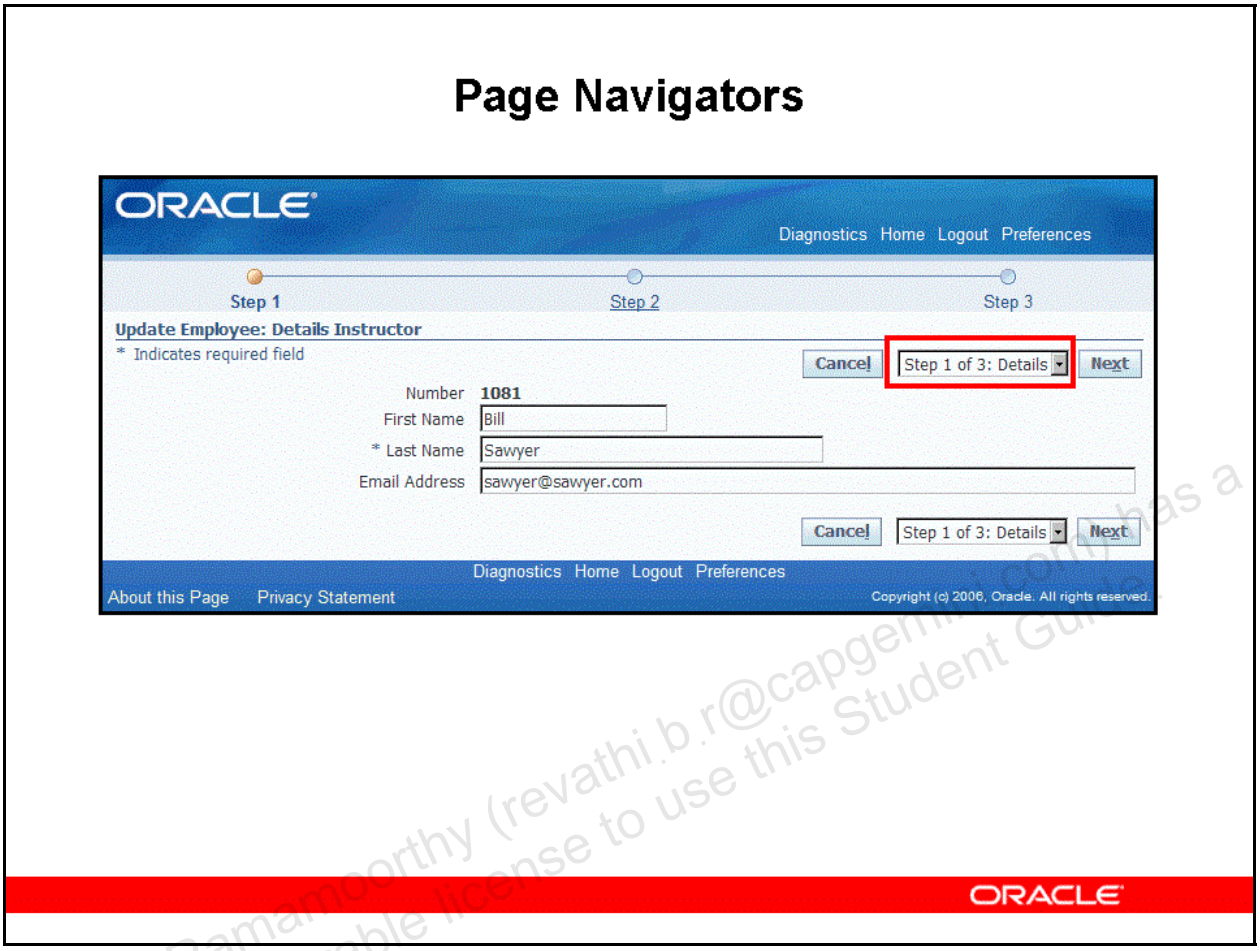


Breadcrumbs



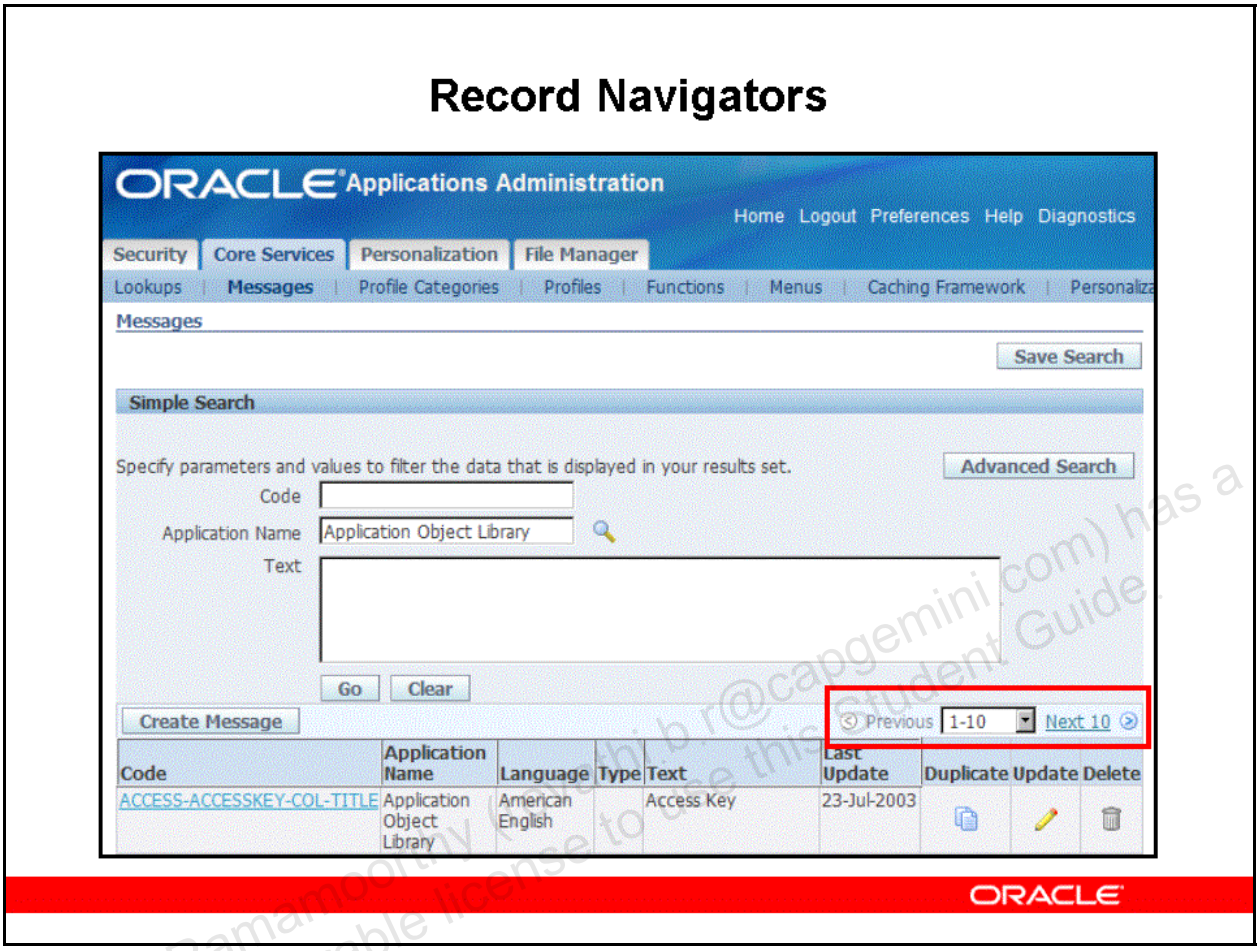
Breadcrumbs are a series of links that display the user's current location within the application page hierarchy.

Page Navigators



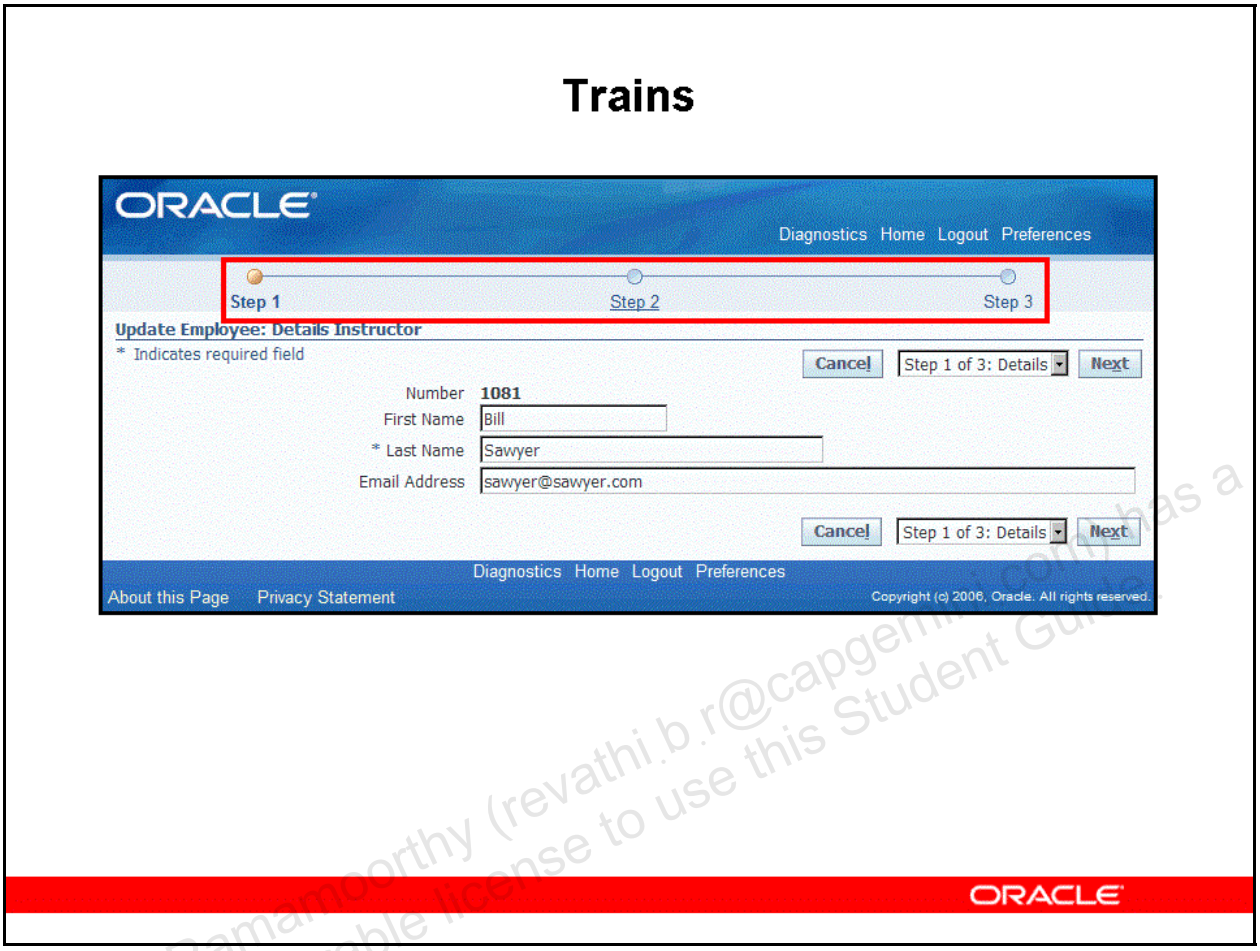
A single navigation bar control that can be configured to allow page or record navigation. Page navigation allows quick movement between specific pages in a sequence.

Record Navigators

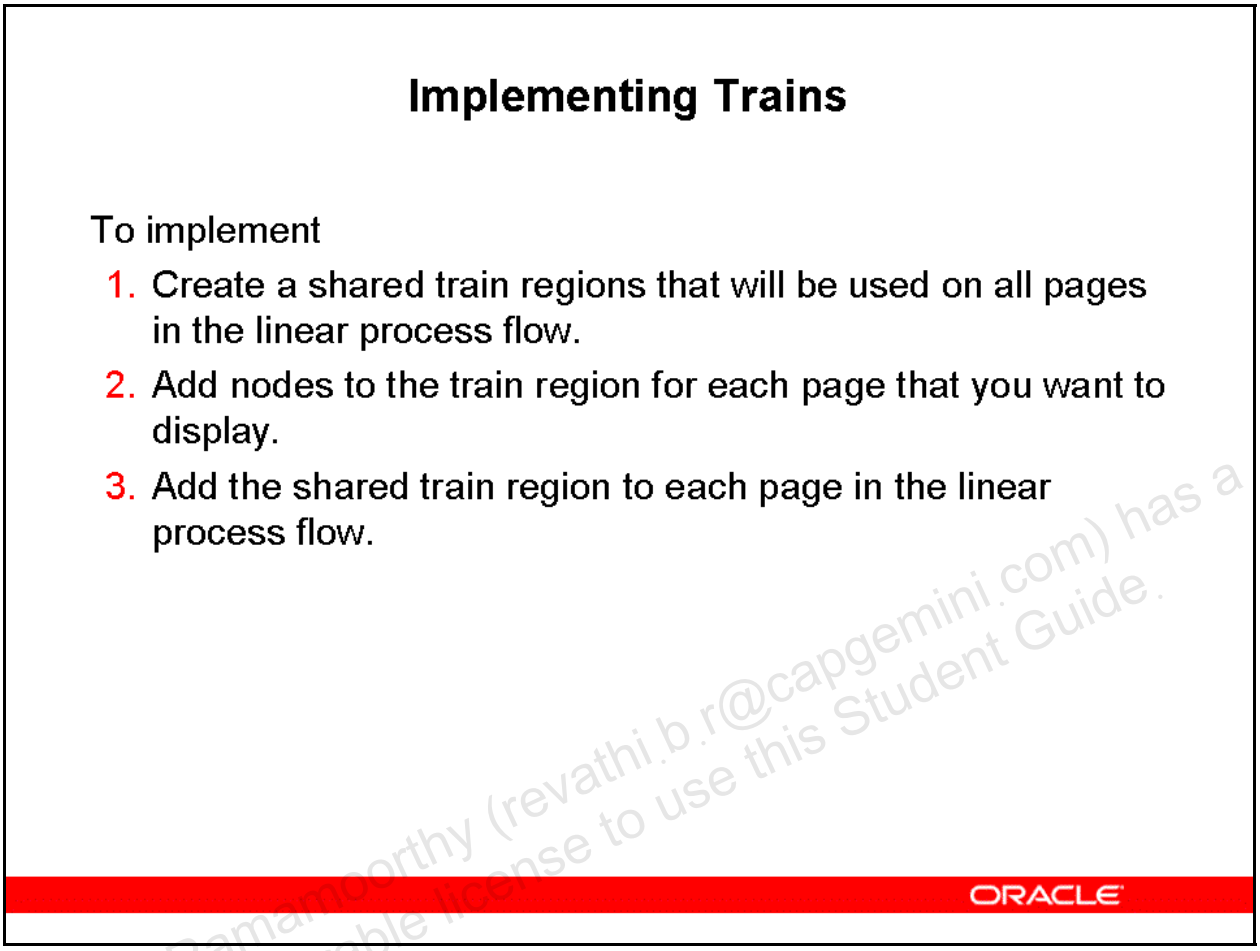


Record navigation allows quick navigation to sets of data in a table or advanced table. Unlike page navigation, you don't have to table any specific actions to implement record navigation. It is automatically implemented for you on table and advanced table regions.

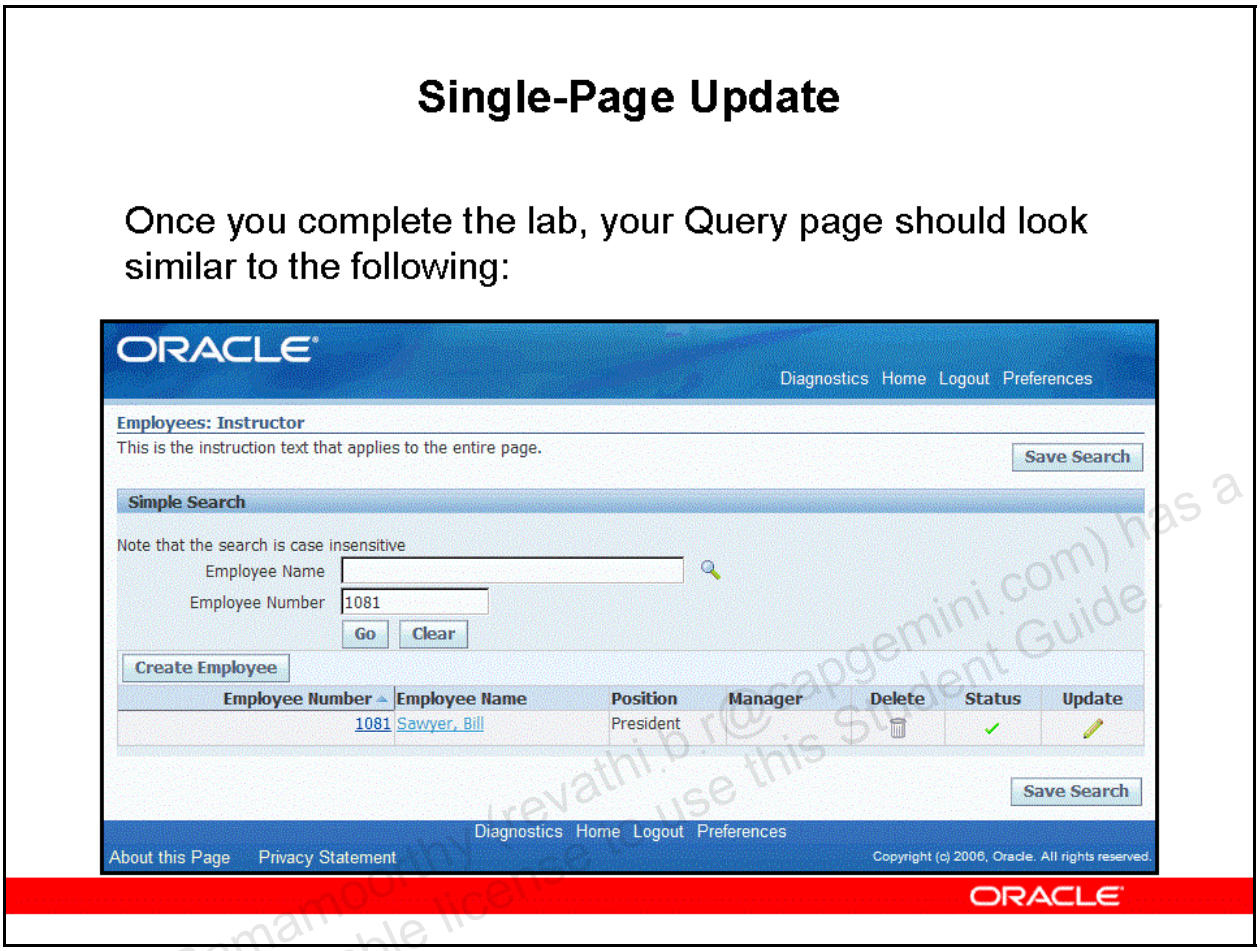
Trains



A train is a graphical component used to show a user's current location in a linear process flow.

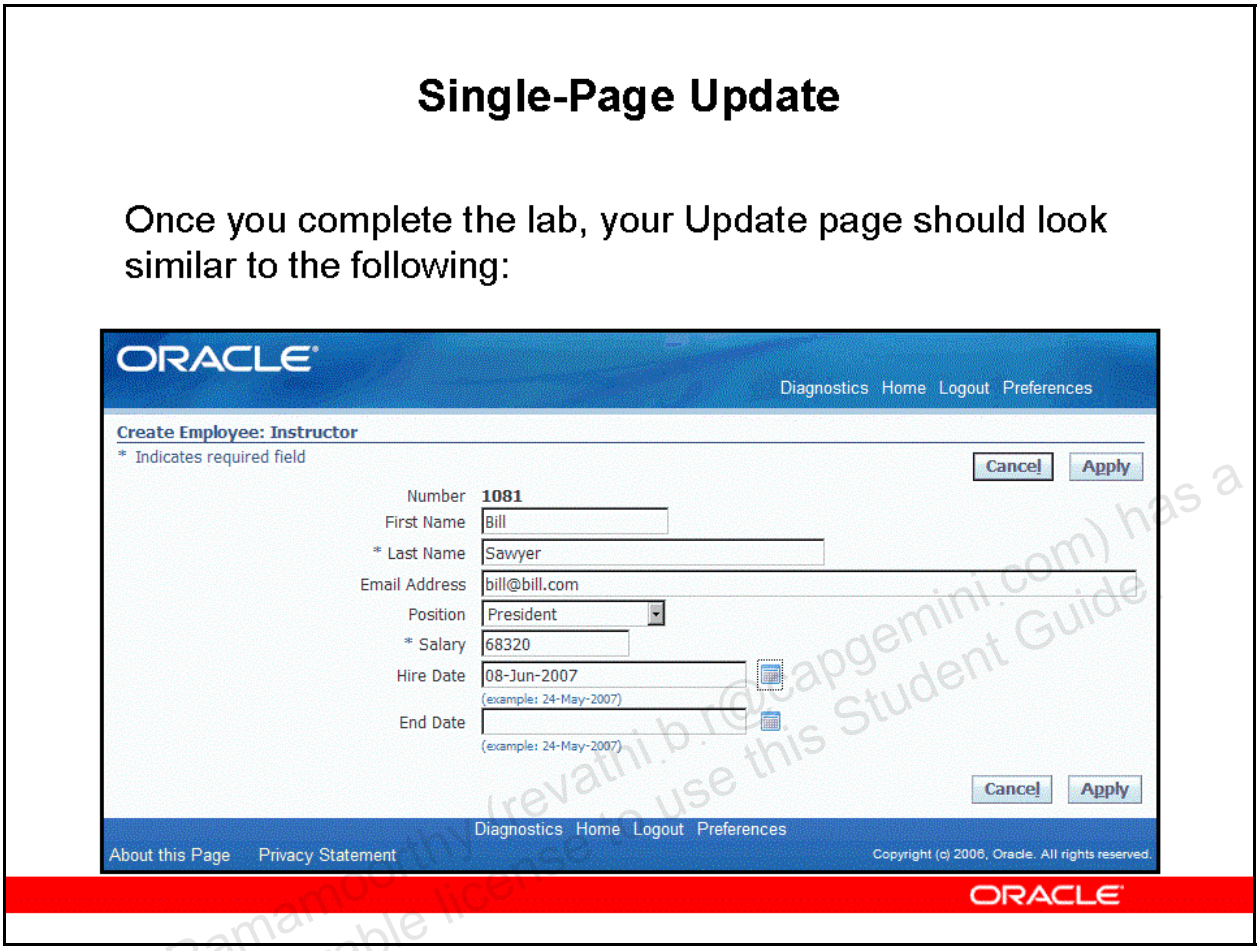


Single-Page Update



The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Single-Page Update

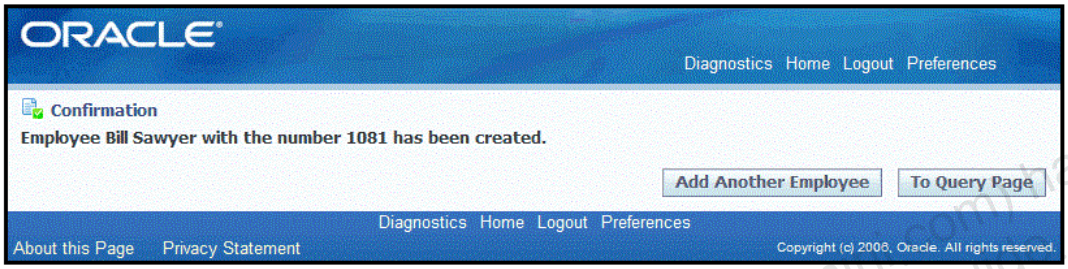


The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Update Confirmation

Update Confirmation

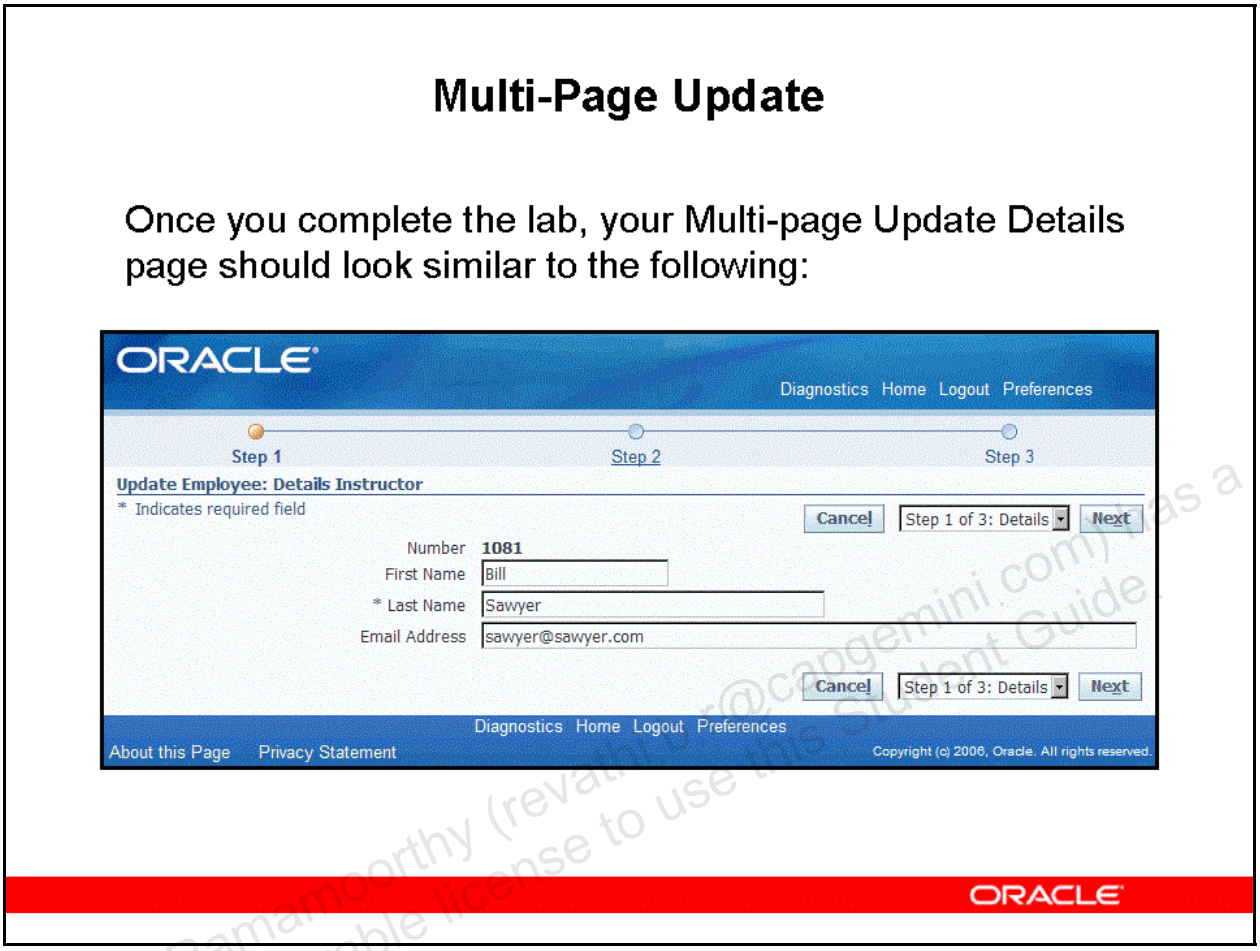
Once you complete the lab, your Update Confirmation dialog page should look similar to the following:



ORACLE

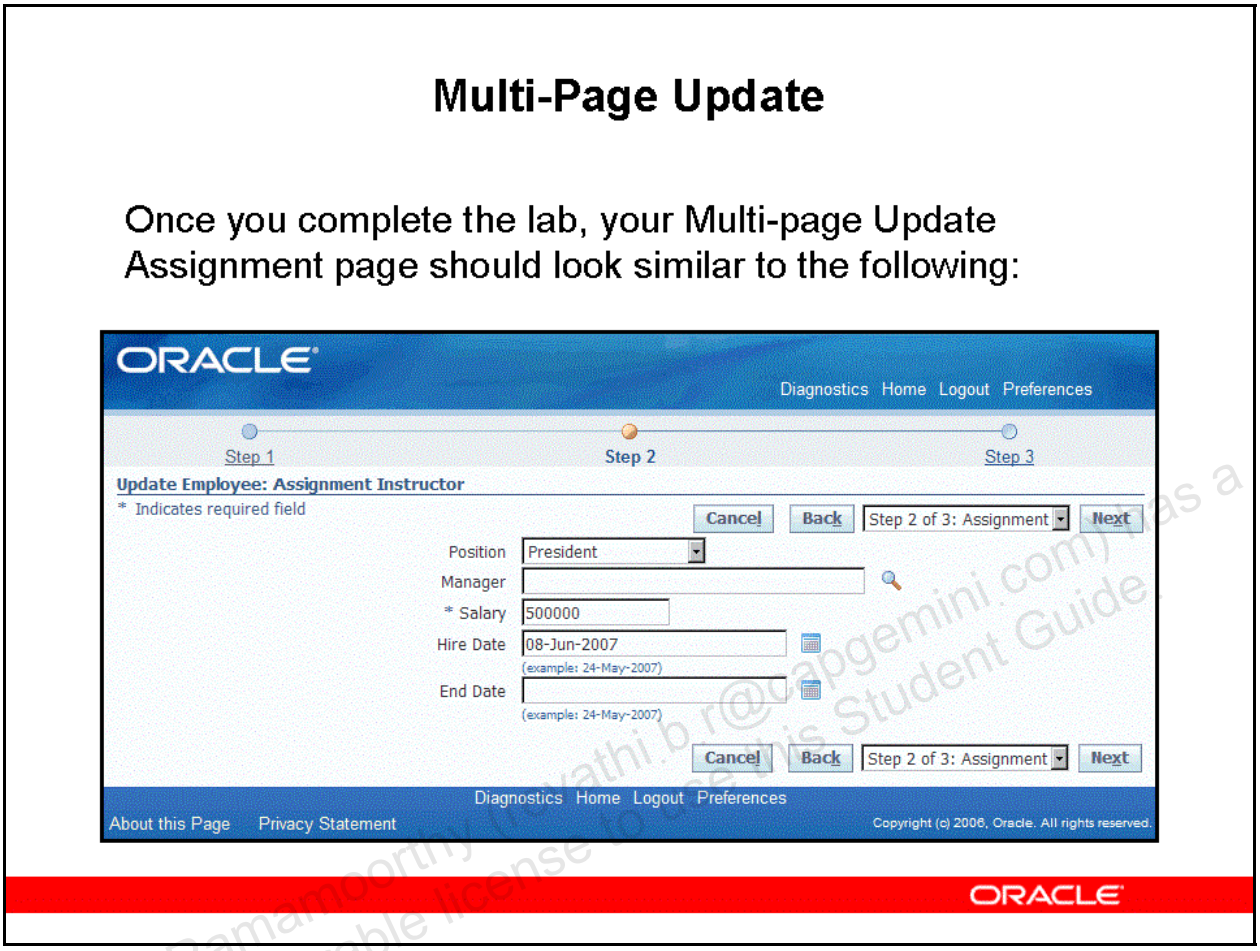
The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Multi-Page Update



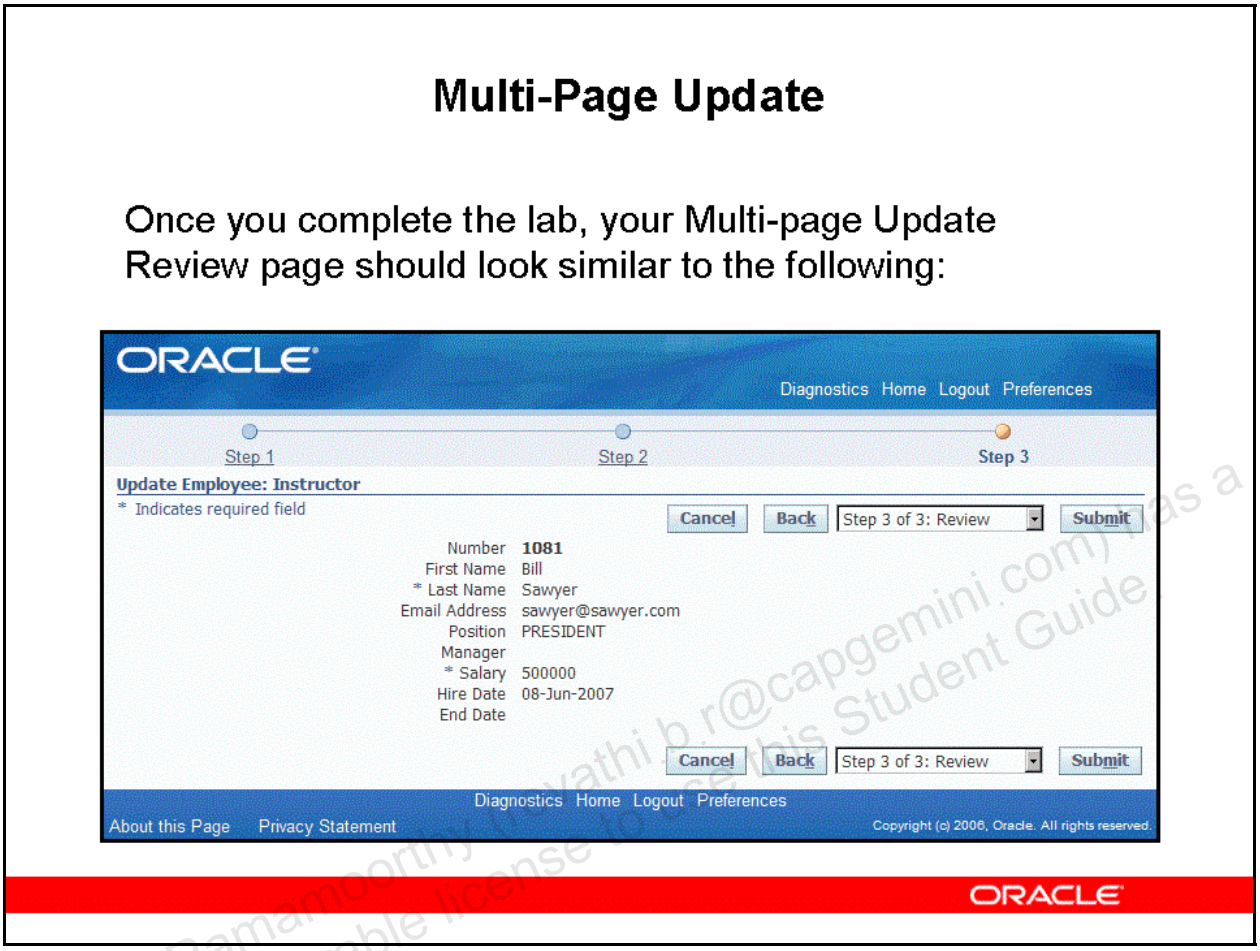
The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Multi-Page Update



The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Multi-Page Update

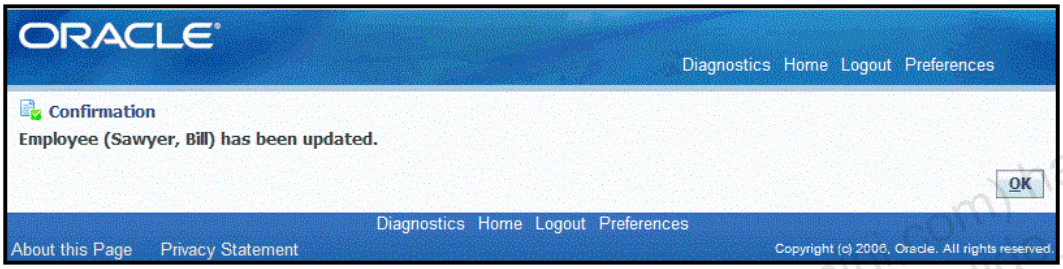


The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

Multi-Page Update Confirmation

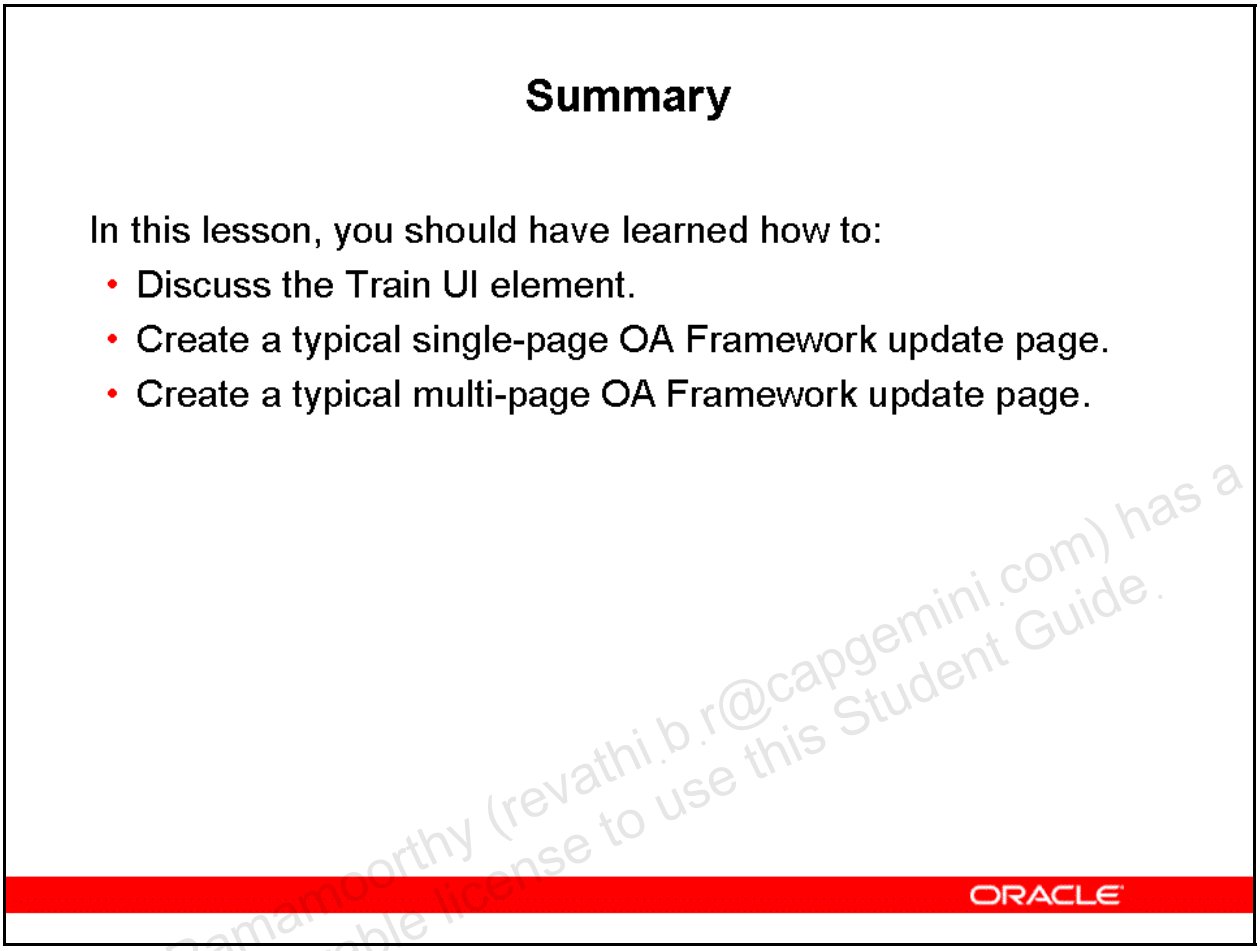
Multi-Page Update Confirmation

Once you complete the lab, your Multi-page Update Confirmation dialog page should look similar to the following:



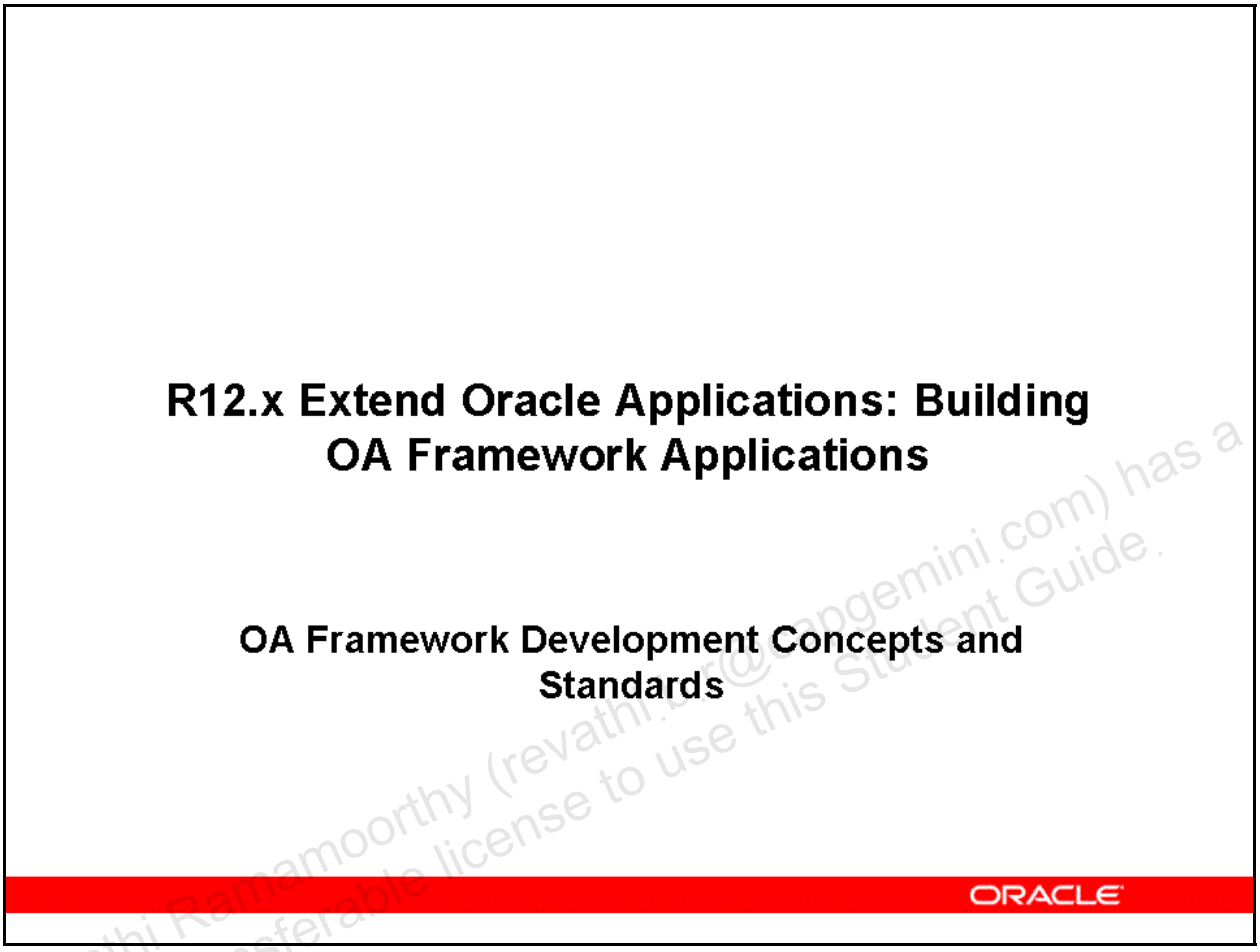
ORACLE

The actual labs will have the most up-to-date screen captures. The above screen capture is only an example.

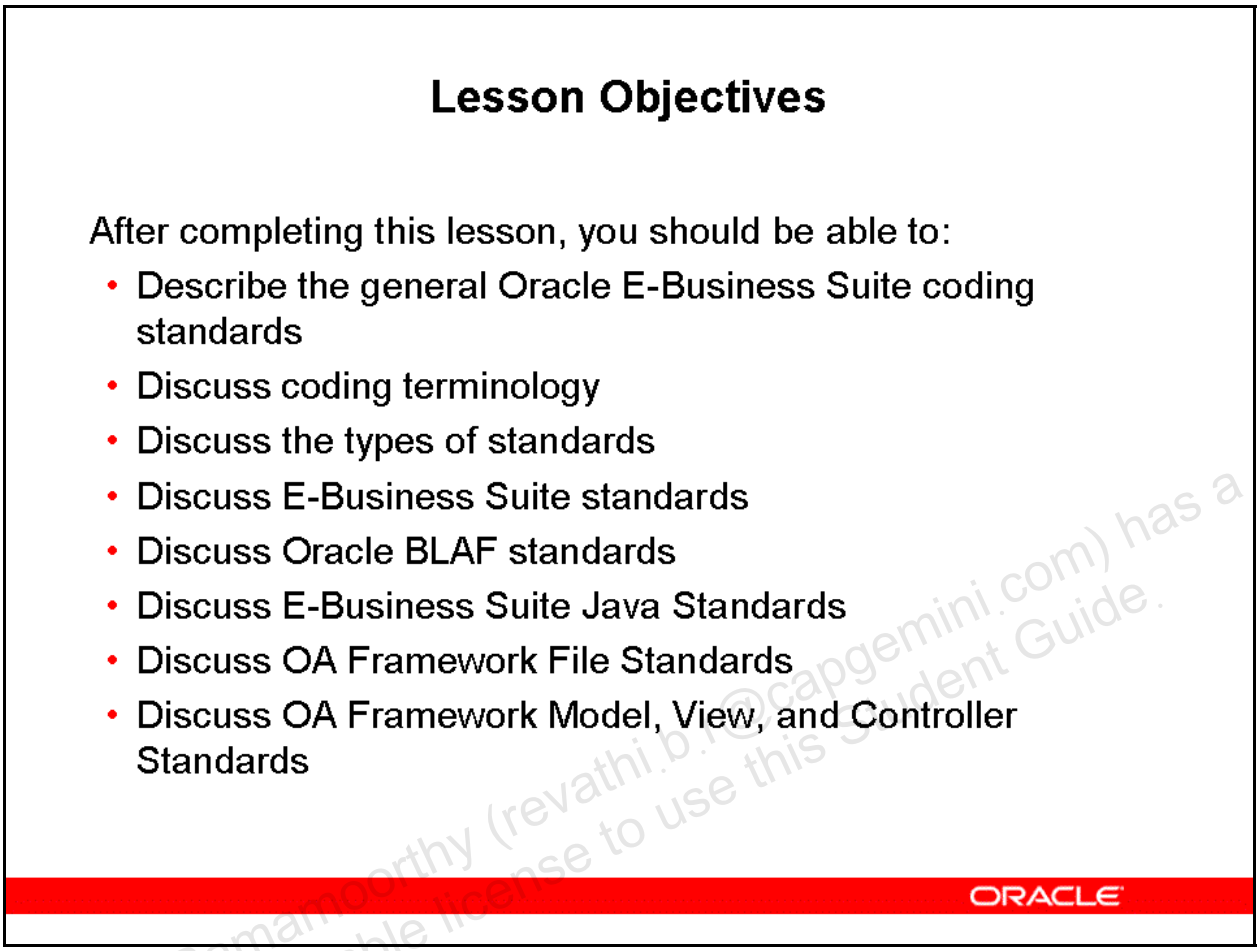


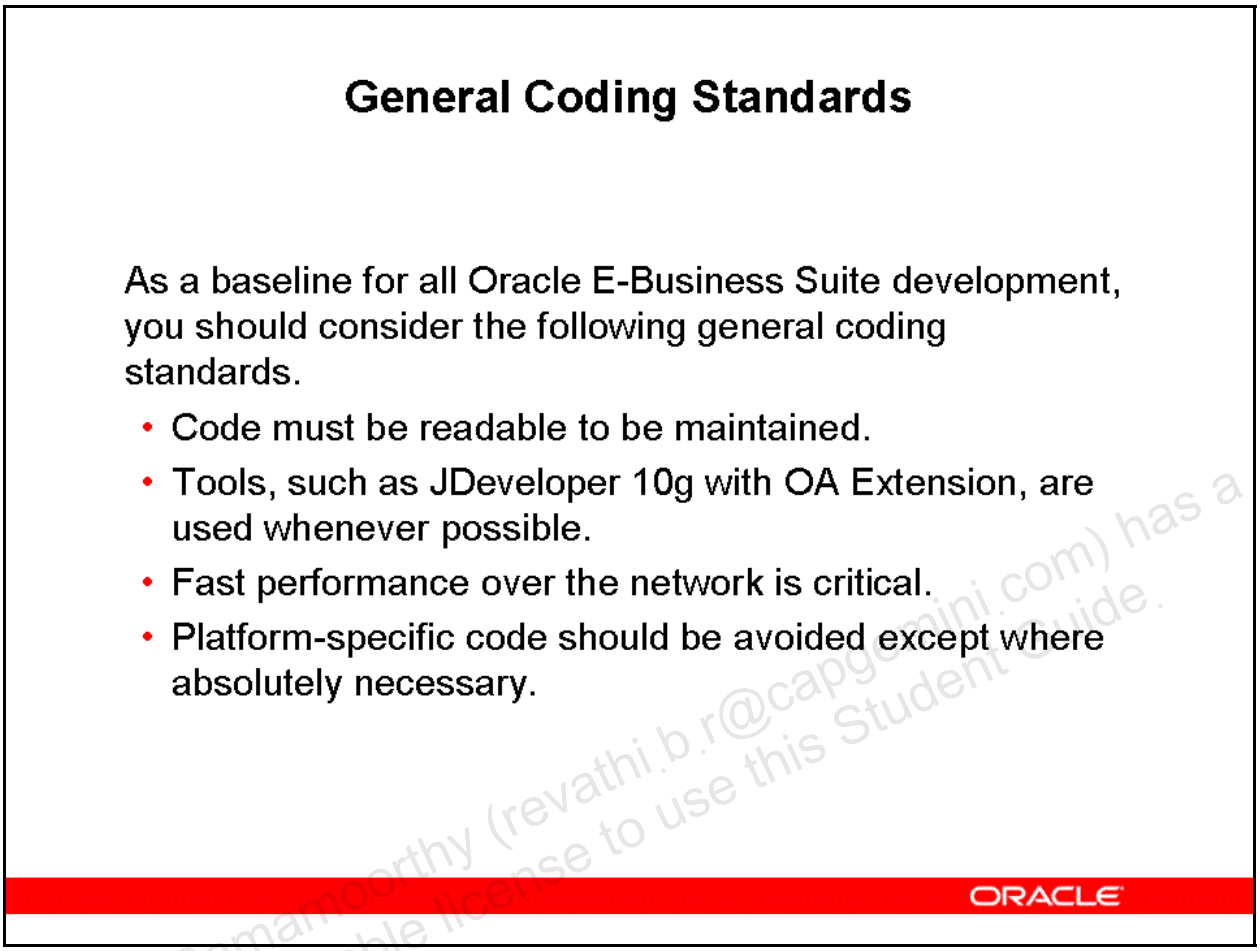
OA Framework Development Concepts and Standards

Chapter 13



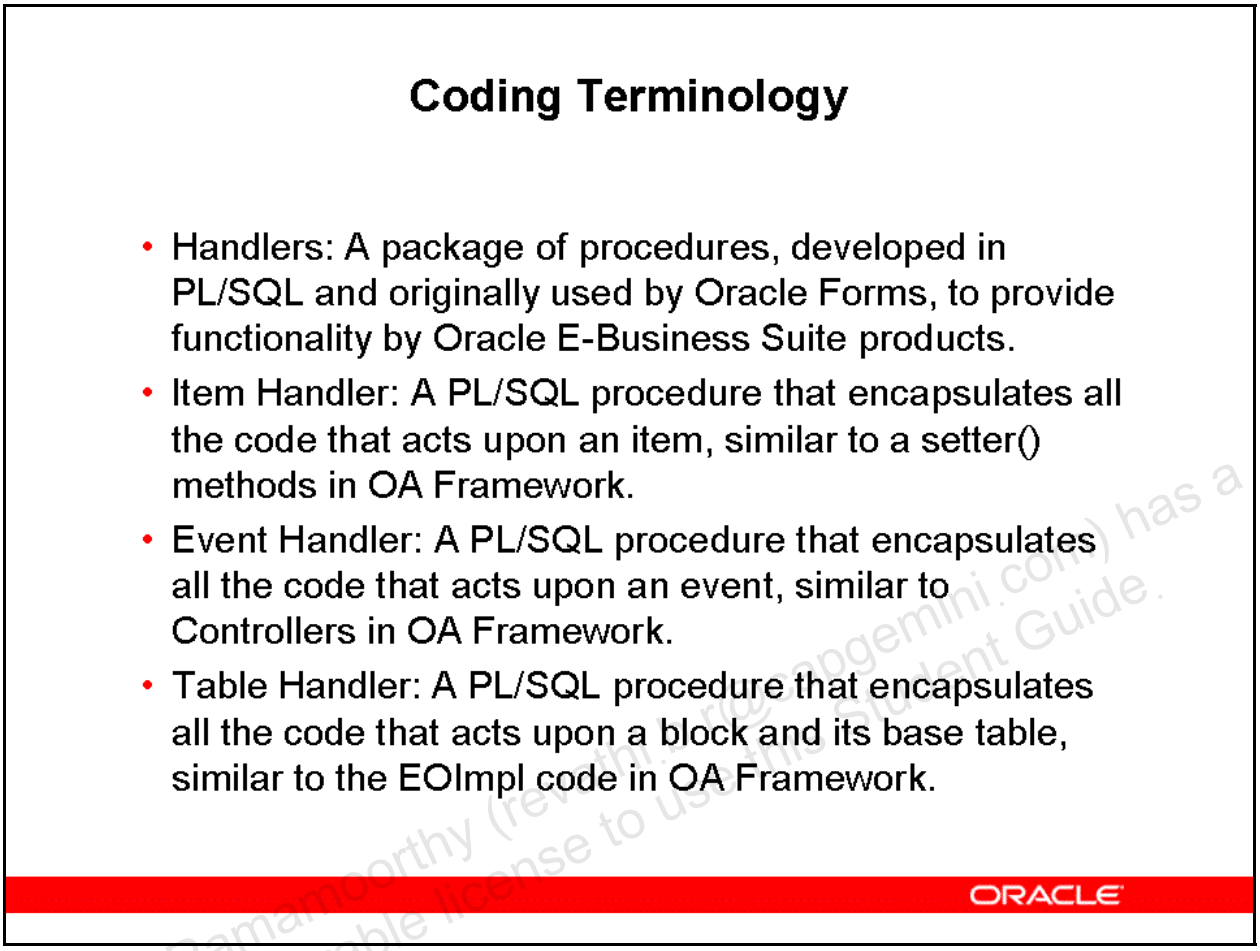
Lesson Objectives





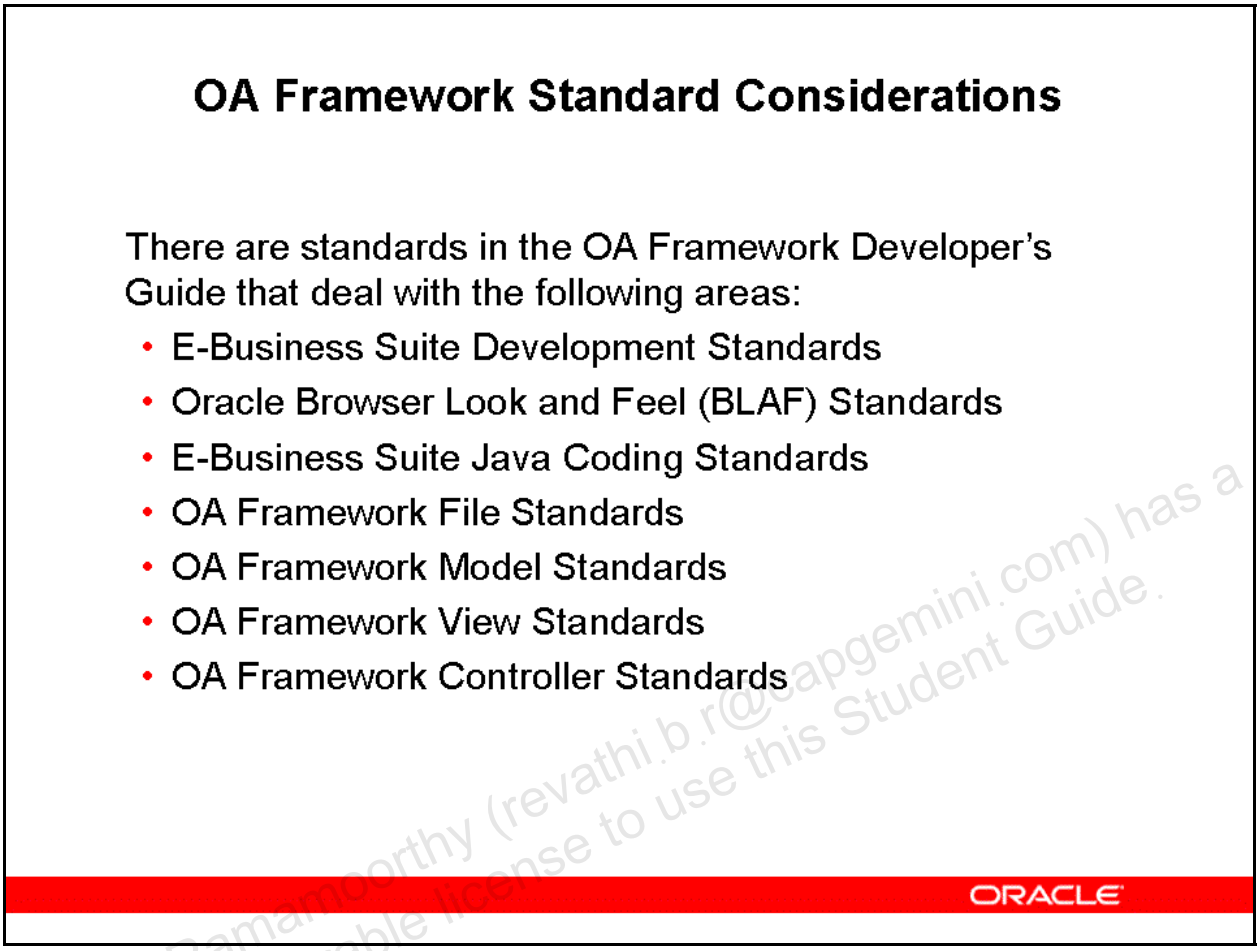
Note: If you are developing OA Framework pages, JDeveloper 10g with OA Extension is not optional. JDeveloper is only optional when you are using technologies other than OA Framework.

Coding Terminology

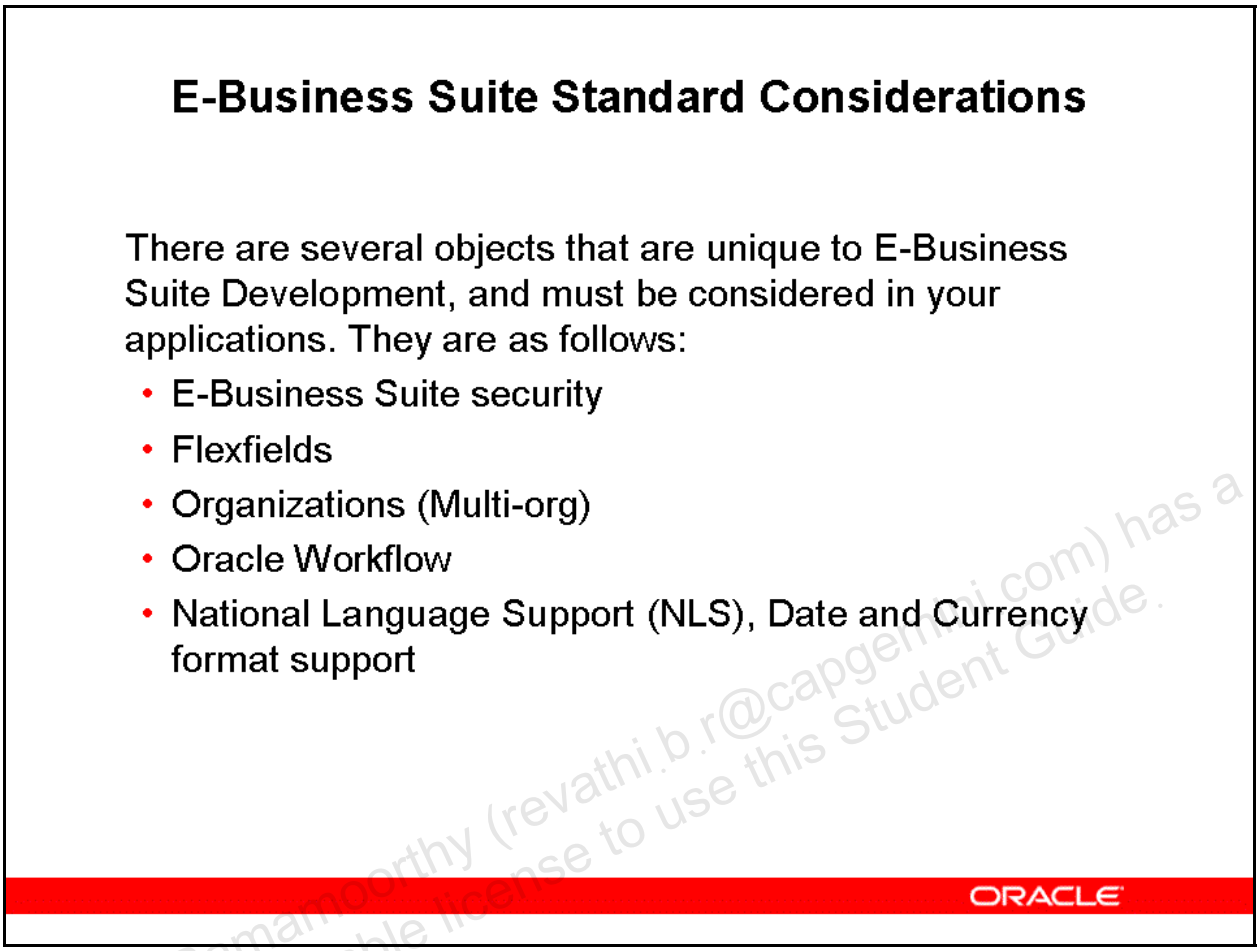


Why discuss technology originally used in Oracle Forms? Many of the E-Business Suite products still use PL/SQL-based Handlers instead of the corresponding objects from OA Framework. By knowing that these objects exist, it will help you understand OA Framework pages that you might have to examine.

Chapter 5: Implementing Server-Side Features of the OA Framework Developer's Guide covers the issues around using PL/SQL procedures and functions in OA Framework pages.



E-Business Suite Standard Considerations



For more information on E-Business Suite security, refer to the System Administrator's Guide – Security in your R12 Documentation Library.

For general development information, there are three sources. The Oracle Applications Developer's Guide in the R12 Documentation Library, and the OA Framework Developer's Guide available with your JDeveloper 10g with OA Extension download are your general programming references.

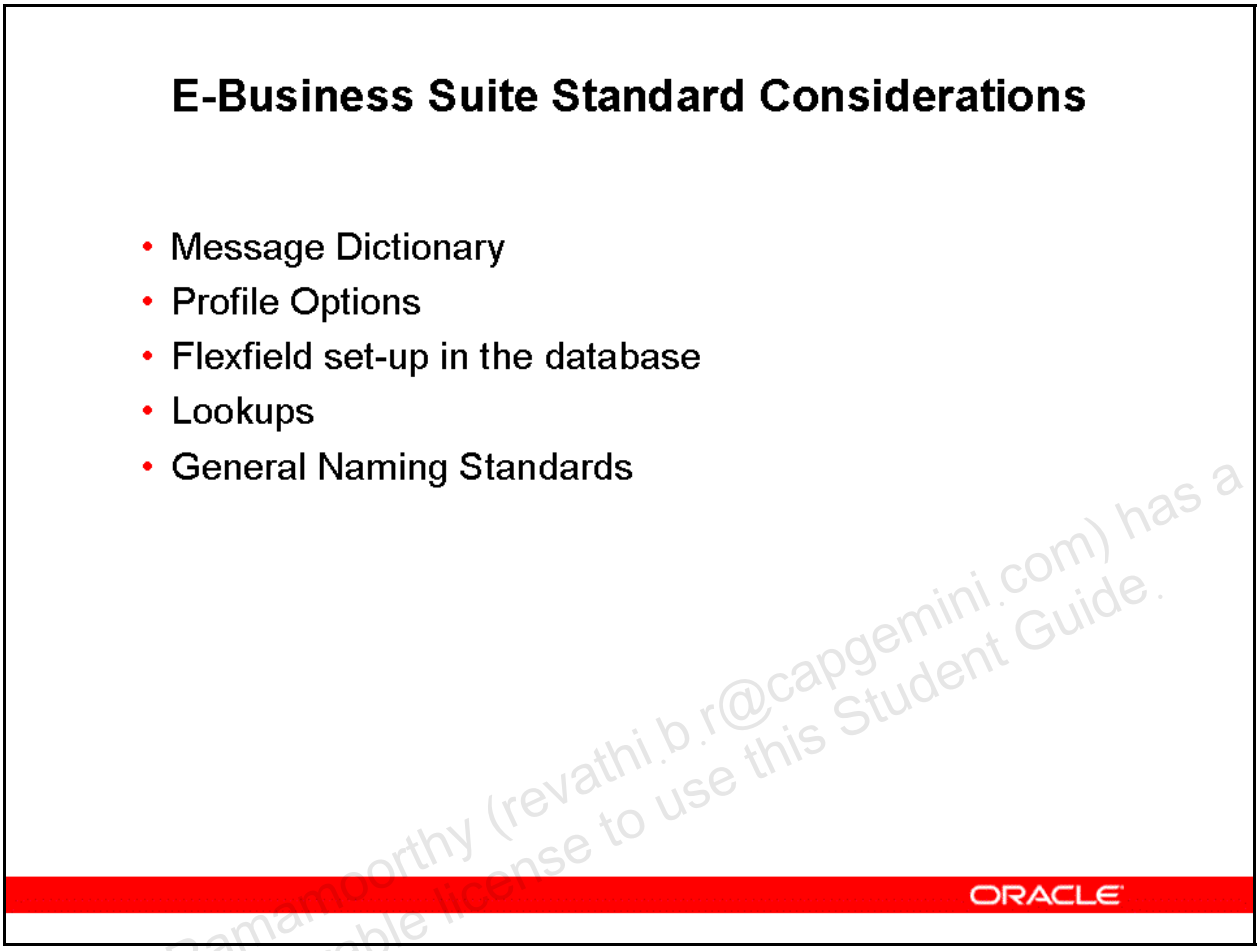
For more information on Flexfields, use the Oracle Applications Flexfields Guide.

For more information on Multiple Organizations, use the Oracle Applications Multiple Organizations Implementation Guide.

For more information on Workflow, use the Oracle Workflow guides. Note: There are several Oracle Workflow guides covering various aspects of this product.

Finally, one of the most important resources is an evolving document called the Oracle Application Framework Documentation Resources, Release 12 document in My Oracle Support (formerly MetaLink). The Knowledge Document is #391554.1. It has links to all the other important OA Framework docs and resources, and is a critical resource for successful OA Framework development.

E-Business Suite Standard Considerations



For more information on the Message Dictionary, refer to Chapter 12 of the Oracle Applications Developer's Guide.

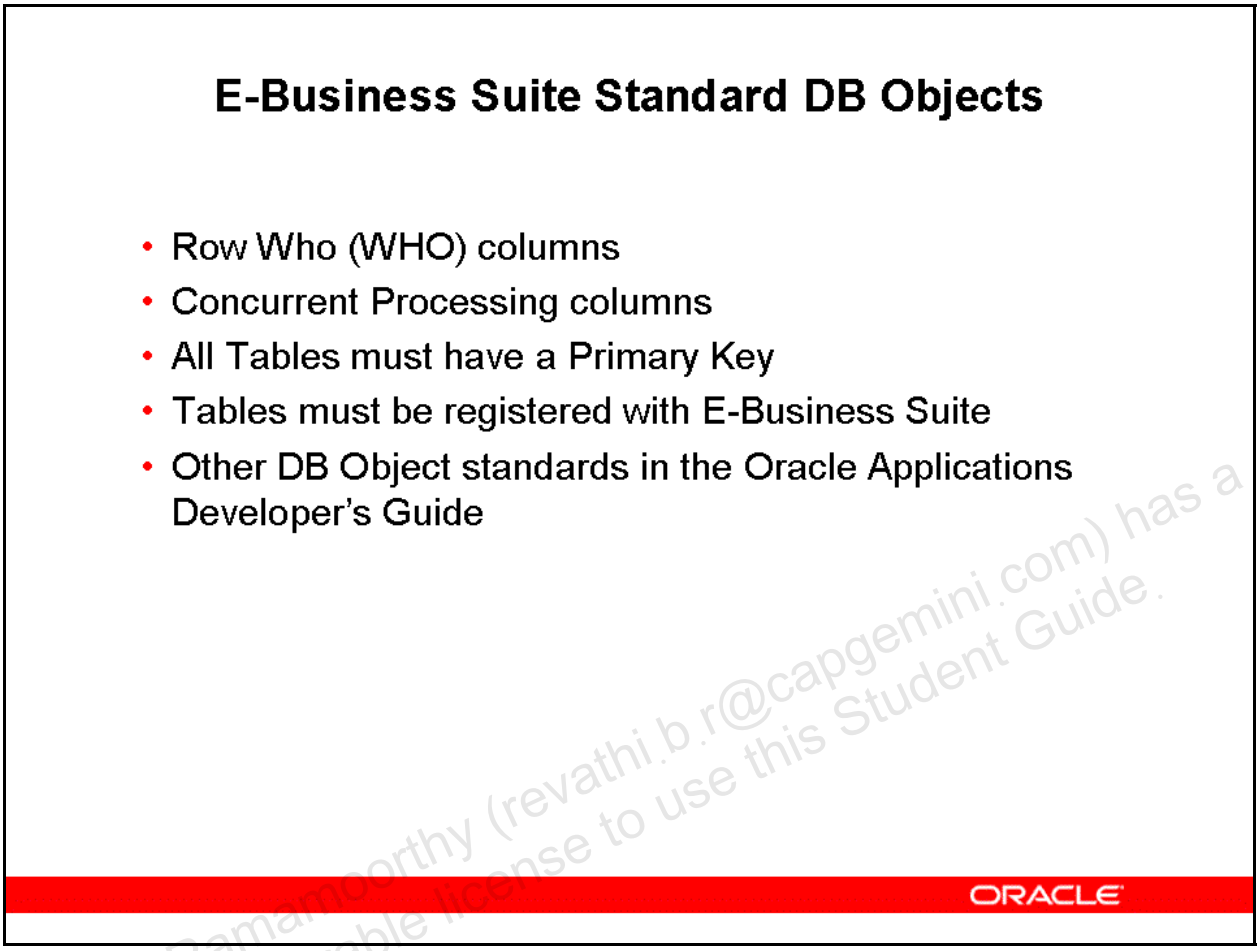
For more information on Profile Options, refer to Chapter 13 of the Oracle Applications Developer's Guide.

For more information of setting up flexfields in your database objects, refer to Chapter 14 of the Oracle Applications Developer's Guide.

For more information Lookups, refer to Appendix A of the Oracle Applications Developer's Guide.

Additionally, the E-Business Suite product follow naming standards for most of their objects, both database and file system related. This lesson will cover the naming standards for OA Framework related objects, but for other standards, like table names, view names, etc., see Chapter 31 of the Oracle Applications Developer's Guide.

E-Business Suite Standard DB Objects



Row Who (WHO) columns are provided as an audit feature in an E-Business Suite Applications. OA Framework EOs are aware of this functionality, but you need to create tables that take advantage of this functionality. WHO needs the following columns in the database table:

CREATED_BY – Number(15) – Not Null

CREATION_DATE – Date – Not Null

LAST_UPDATED_BY – Number(15) – Not Null

LAST_UPDATE_DATE – Date – Not Null

LAST_UPDATE_LOGIN – Number(15)

If a table can be updated by a Concurrent Processing, it needs these additional columns:

REQUEST_ID – Number(15)

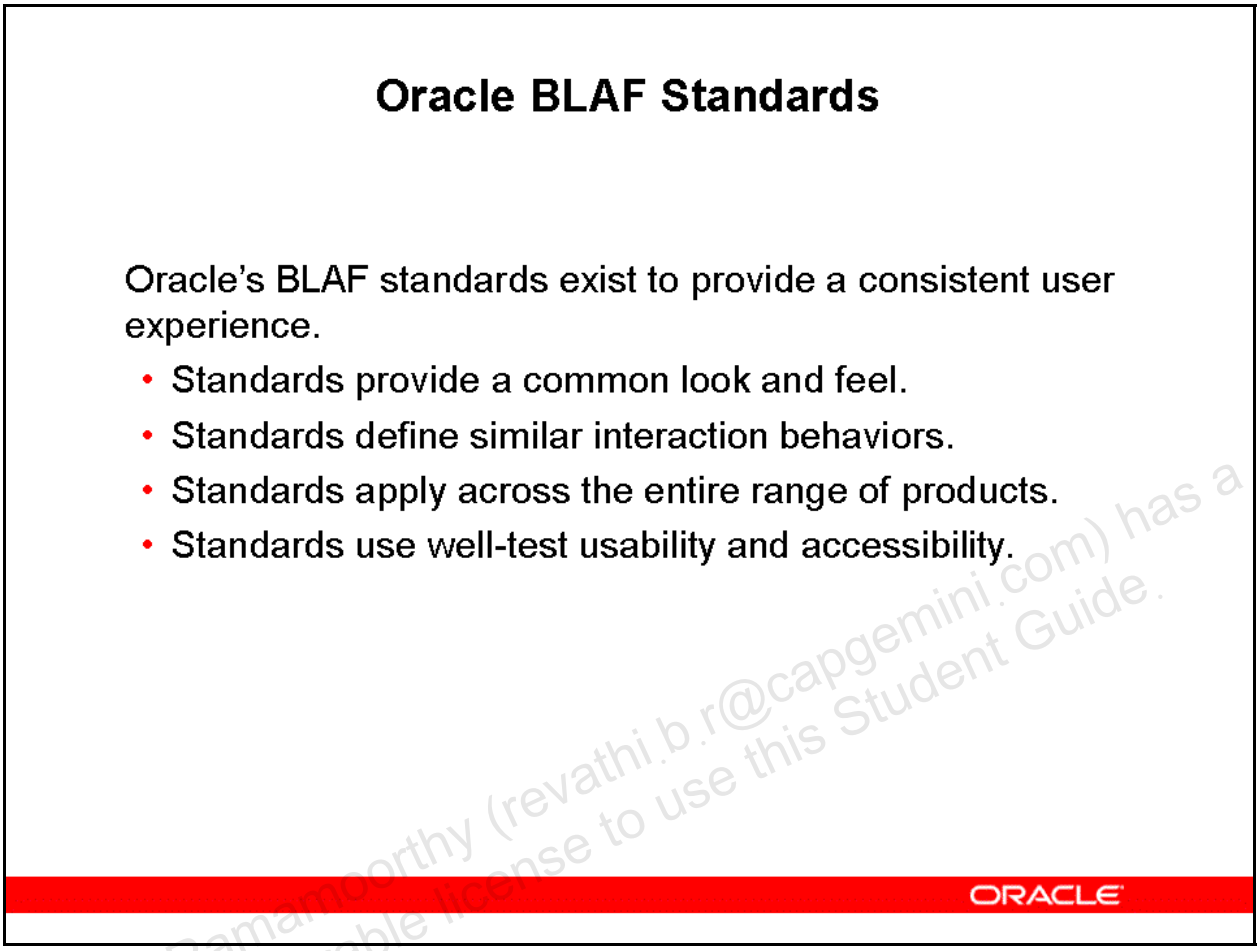
PROGRAM_APPLICATION_ID – Number(15)

PROGRAM_ID – Number(15)

PROGRAM_UPDATE_DATE – Date

Tables are registered using the Table Registration API, which is documented in the Oracle Applications Developer's Guide.

Oracle BLAF Standards



The latest Oracle BLAF standards are available online at the Oracle website at:
http://www.oracle.com/technology/tech/blaf/specs/index.html

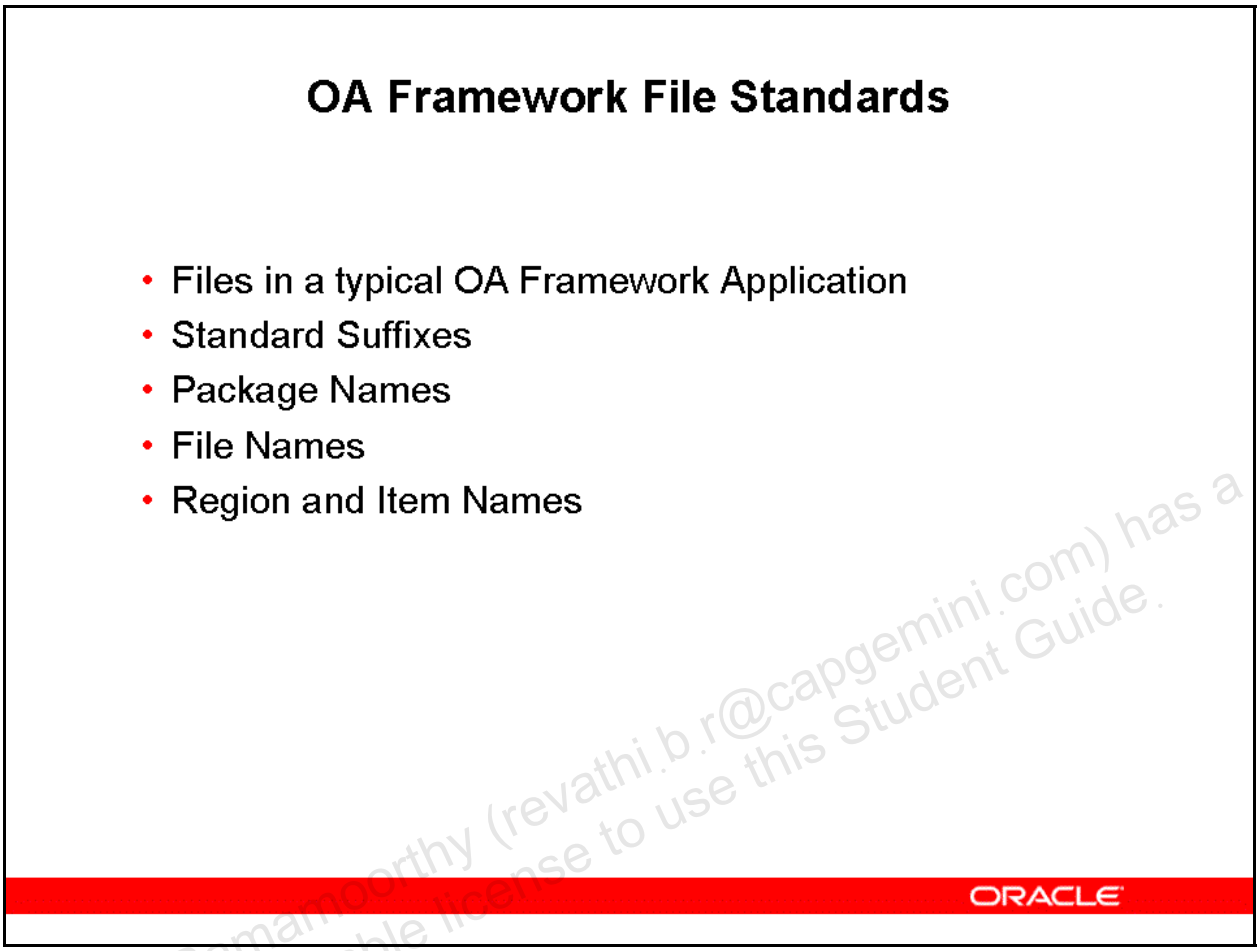
Why is this important to you? You are writing OA Framework pages and applications, either a new customized add-ons or modifications to existing pages. Either way, for ease of maintenance, reductions in training costs, improved usability, and a host of other considerations, your pages and applications should look and behave like any other Oracle product.

E-Business Suite Java Standards

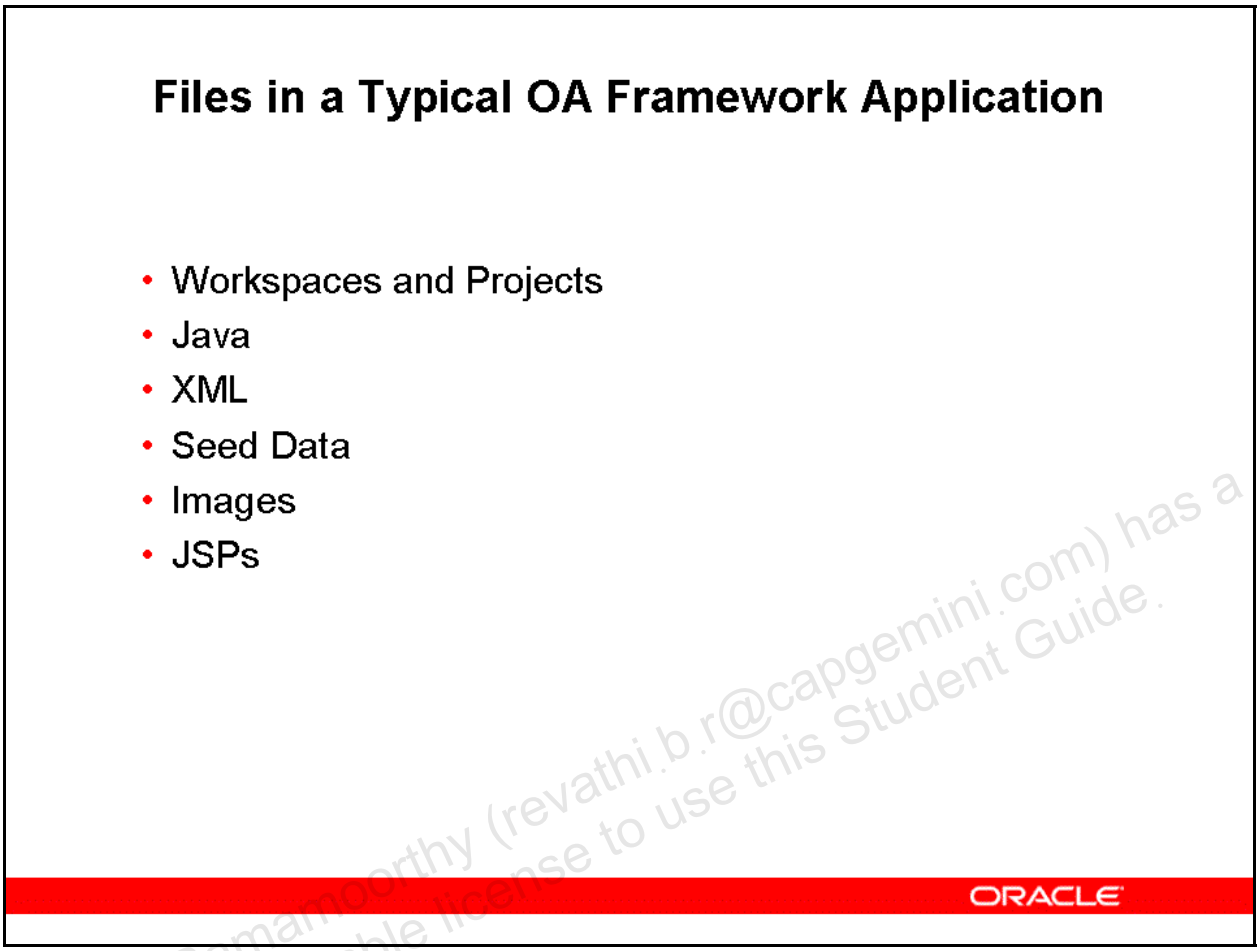
There are numerous E-Business Suite Java Standards. Some of the more important are:

- Avoid client-side Java (Swing, EWT, applets, etc.).
- Never use Java in the database.
- Do not use browser cookies to store state information.
- Do not use resource bundles, use the Message Dictionary.
- Do not use Java APIs that rely on the platform encoding or locale.

ORACLE



Files in a Typical OA Framework Application



Workspaces and Projects:

JDeveloper files:

*.jws	Workspace XML definition
*.jpr	Project XML definition
*.jpx	BC4J Substitution XML definition
*.xcfg	JDeveloper configuration file

Java/Class Files:

CO	Controller files
AMImpl	Application Module
VOImpl	View Objects
VORowImpl	Generated VO Accessors
EOImpl	Entity Objects

EntityExpert	Entity Experts for Validation

XML Files:

PG/RN	UI Page and Regions
Attribute Sets	Reusable UI definitions
AM.xml	Application Module definitions
VO.xml	View Object definitions
EO.xml	Entity Object definitions
AO.xml	Association Object definitions
VL.xml	View Link definitions

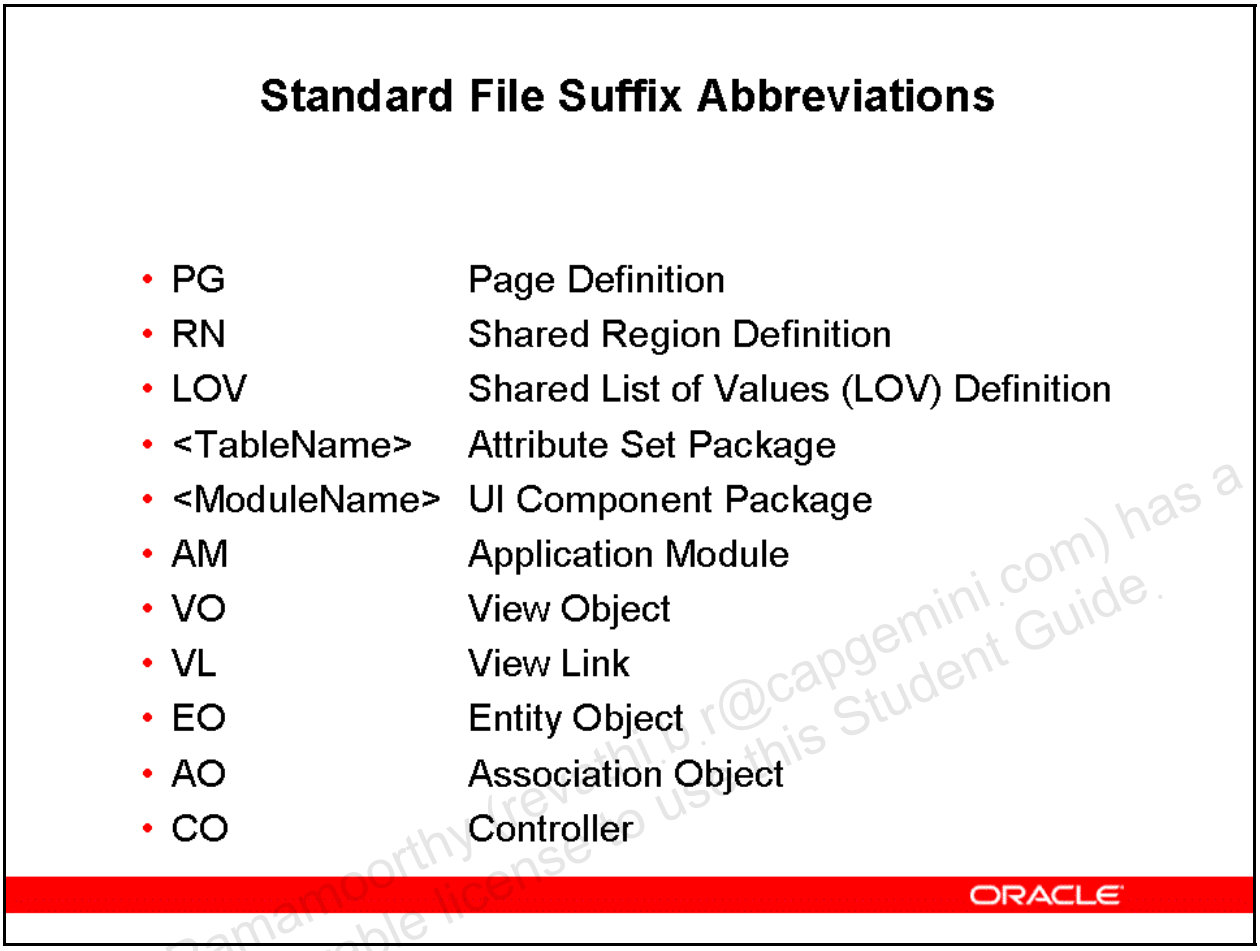
Seed Data:

Menu (.ldt)	Seeded menu definitions
Message (.ldt)	Translatable messages for the Message Dictionary
*.ldt	General seed data, like users, responsibilities, concurrent programs, profiles options, lookups, etc.

Images:

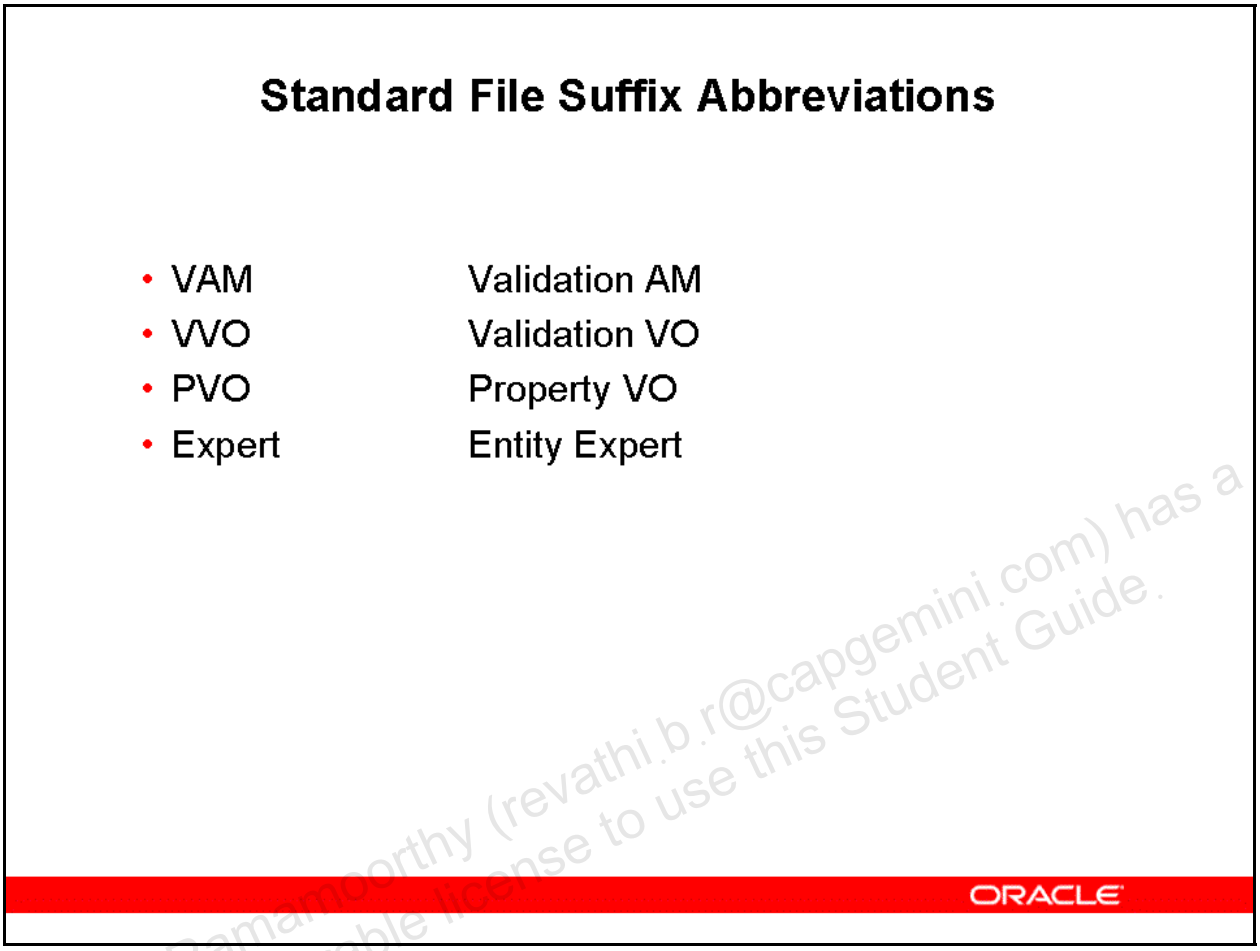
*.gif	Put into $OA_MEDIA directory on the server.
JSPs	**Note**: All OA Framework pages are routed through a single, central JSP (OA.jsp). There is no need for OA Framework applications to create additional JSPs.

Standard File Suffix Abbreviations



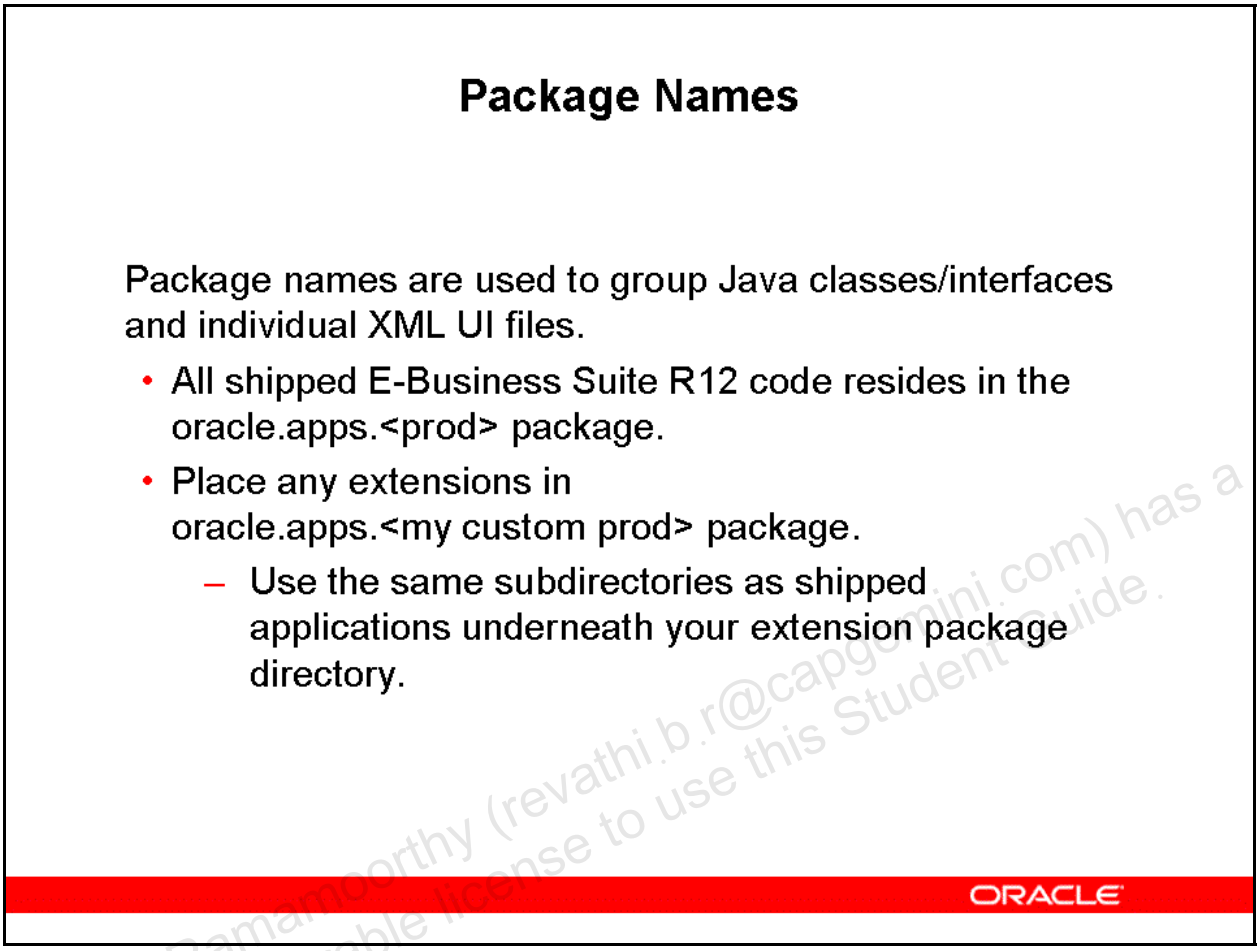
This slide could have been presented earlier in the course but would have created too much confusion. Some of the objects above you have already encountered.

Standard File Suffix Abbreviations



This slide could have been presented earlier in the course but would have created too much confusion. Some of the objects above you have already encountered.

Package Names

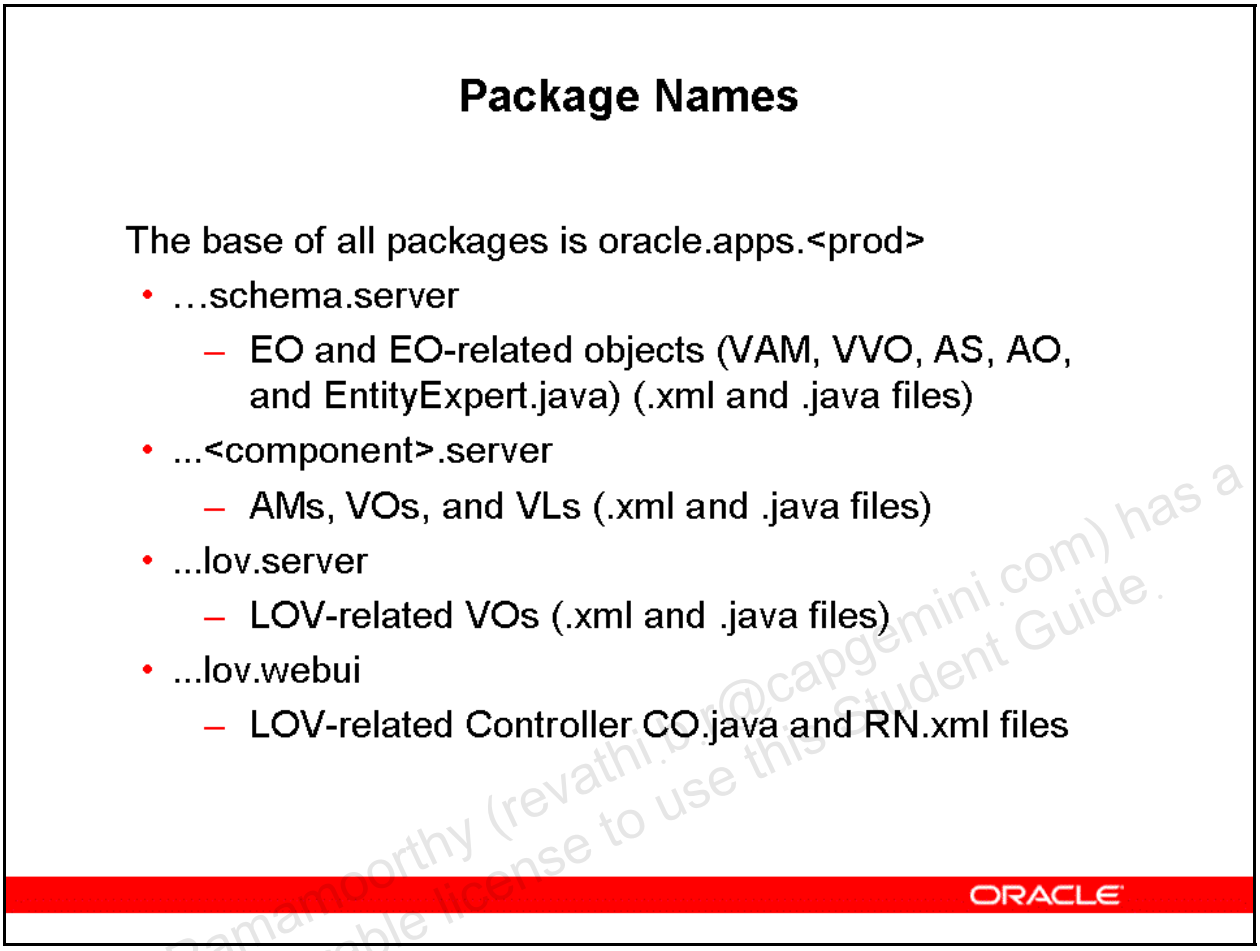


Package names are used to group Java classes/interfaces and individual XML UI files. The Oracle corporate standard on package names requires that they begin with a lower case letter and use initial capitals rather than underscores for each subsequent word in the name (for example: oracle.apps.fnd.wf.statusMonitor).

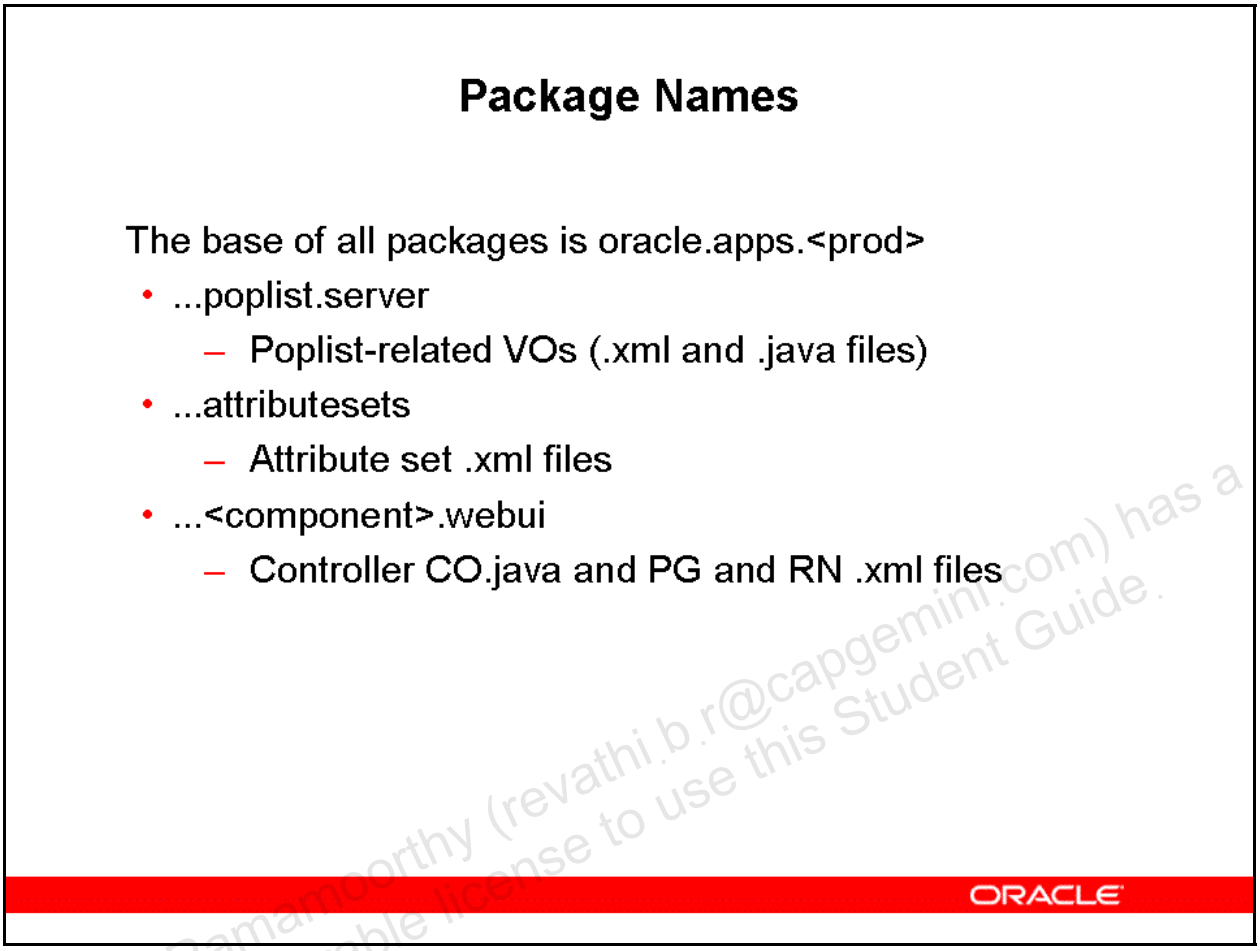
At the highest level, all Oracle E-Business Suite code goes into the oracle.apps package. You should NOT create your code under the following restricted packages:

- oracle.apps.fnd.framework
- oracle.jrad
- oracle.mds
- oracle.jbo
- oracle.cabo
- oracle.jdbc

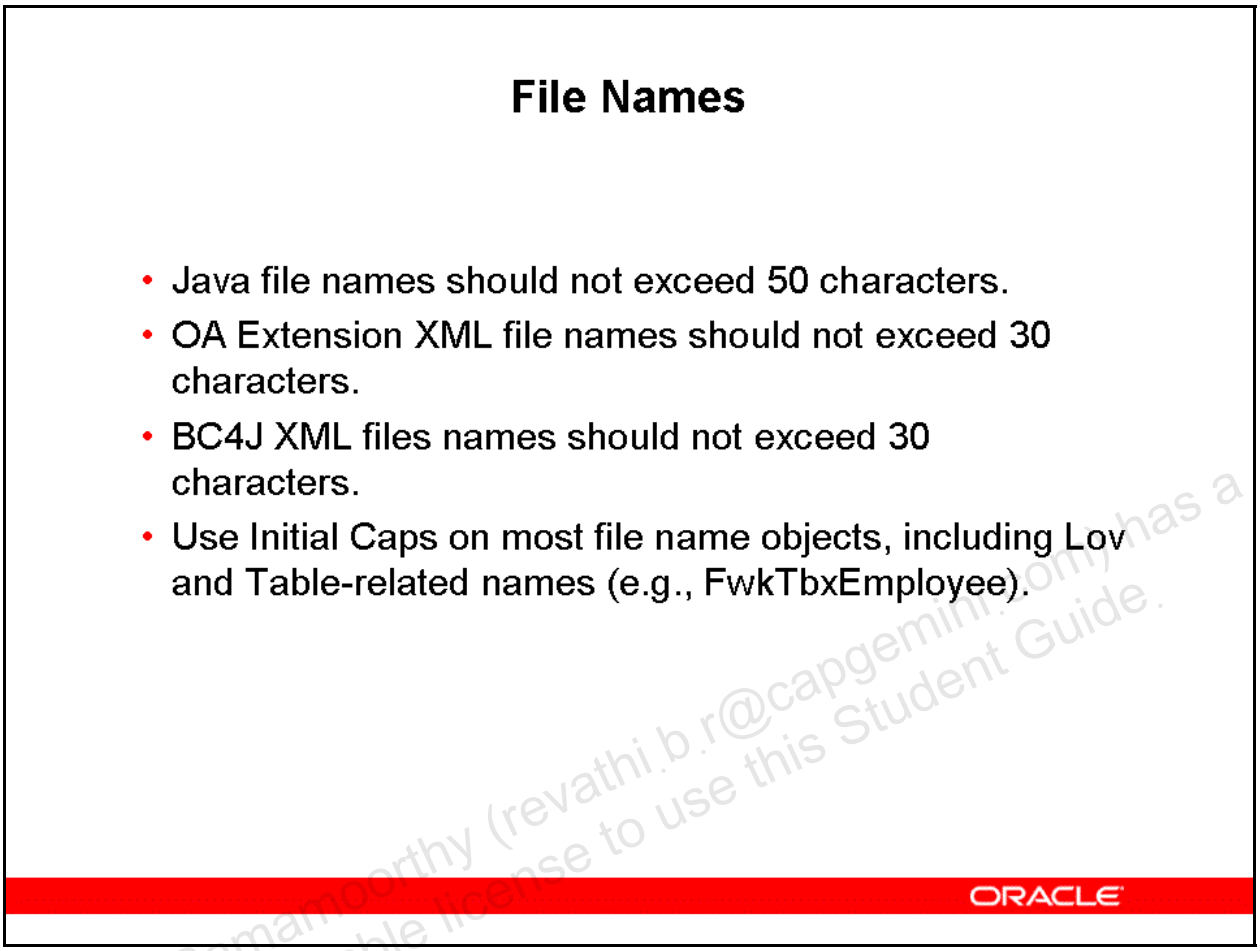
Package Names



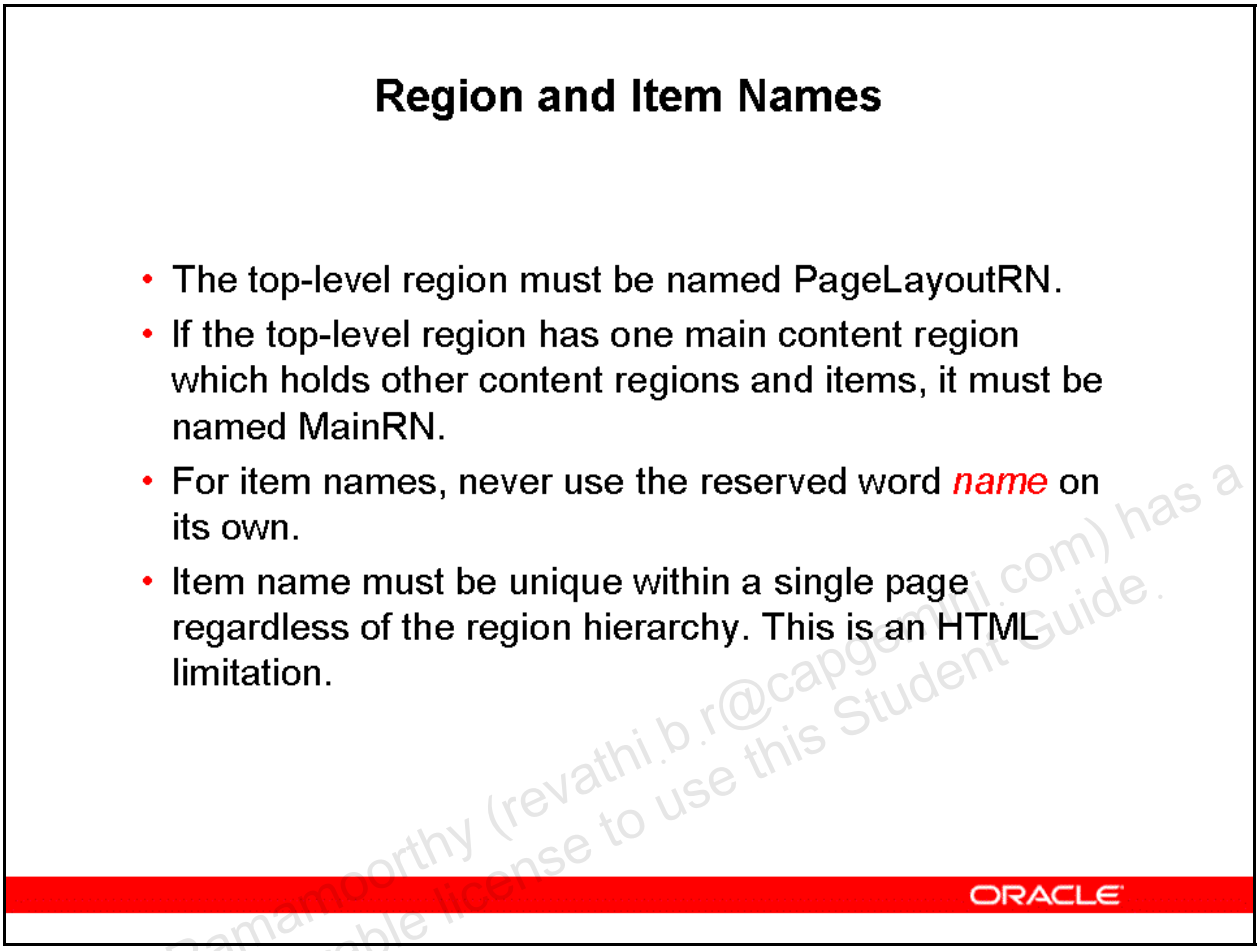
Do you have to follow these standards? No, but you should know where E-Business Suite developers are going to put their files, and you can reduce maintenance costs by following the established standards.

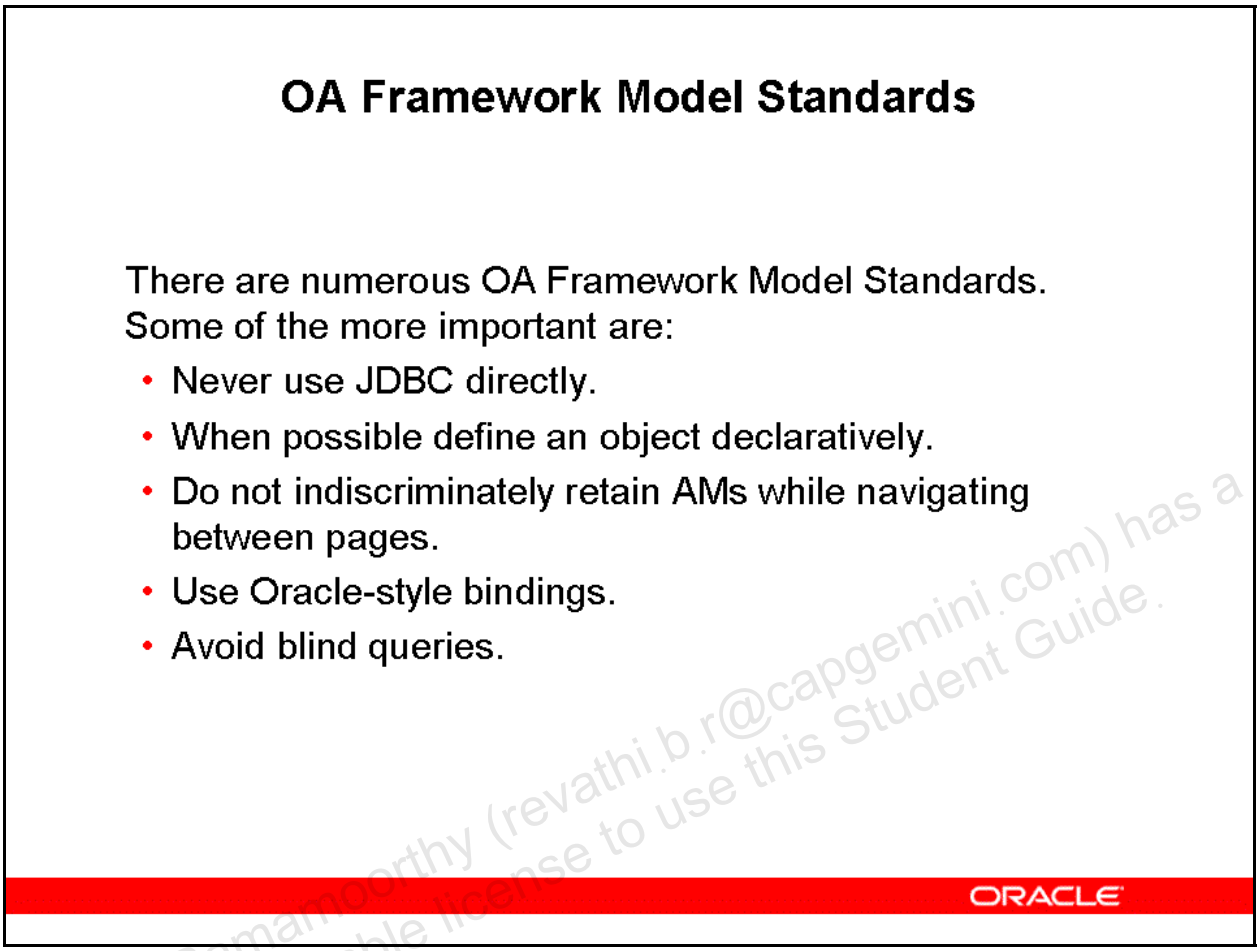


File Names

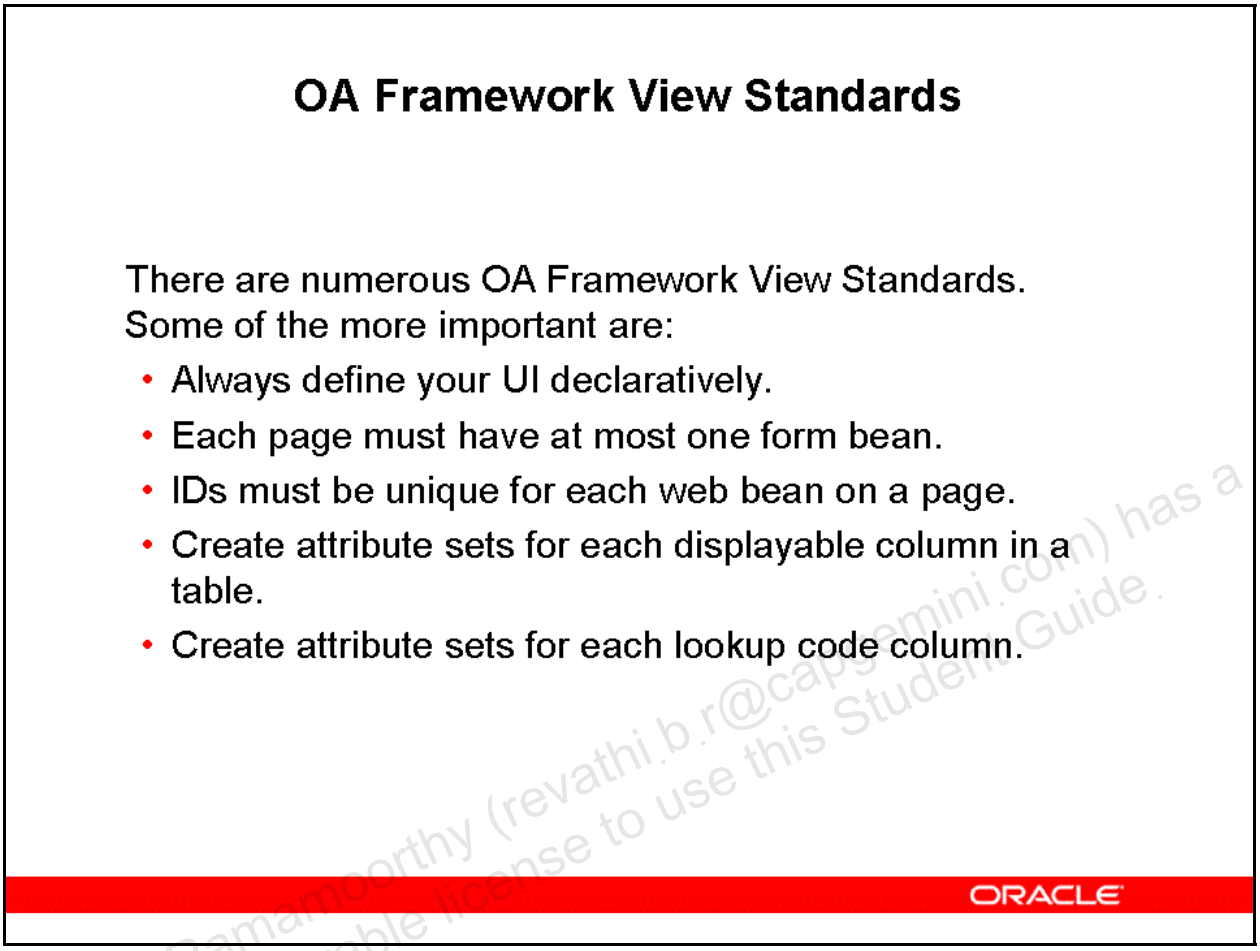


Note: XML names must be unique within the first 30 characters. If they are not unique within the first 30 characters, you will encounter unexpected behavior.

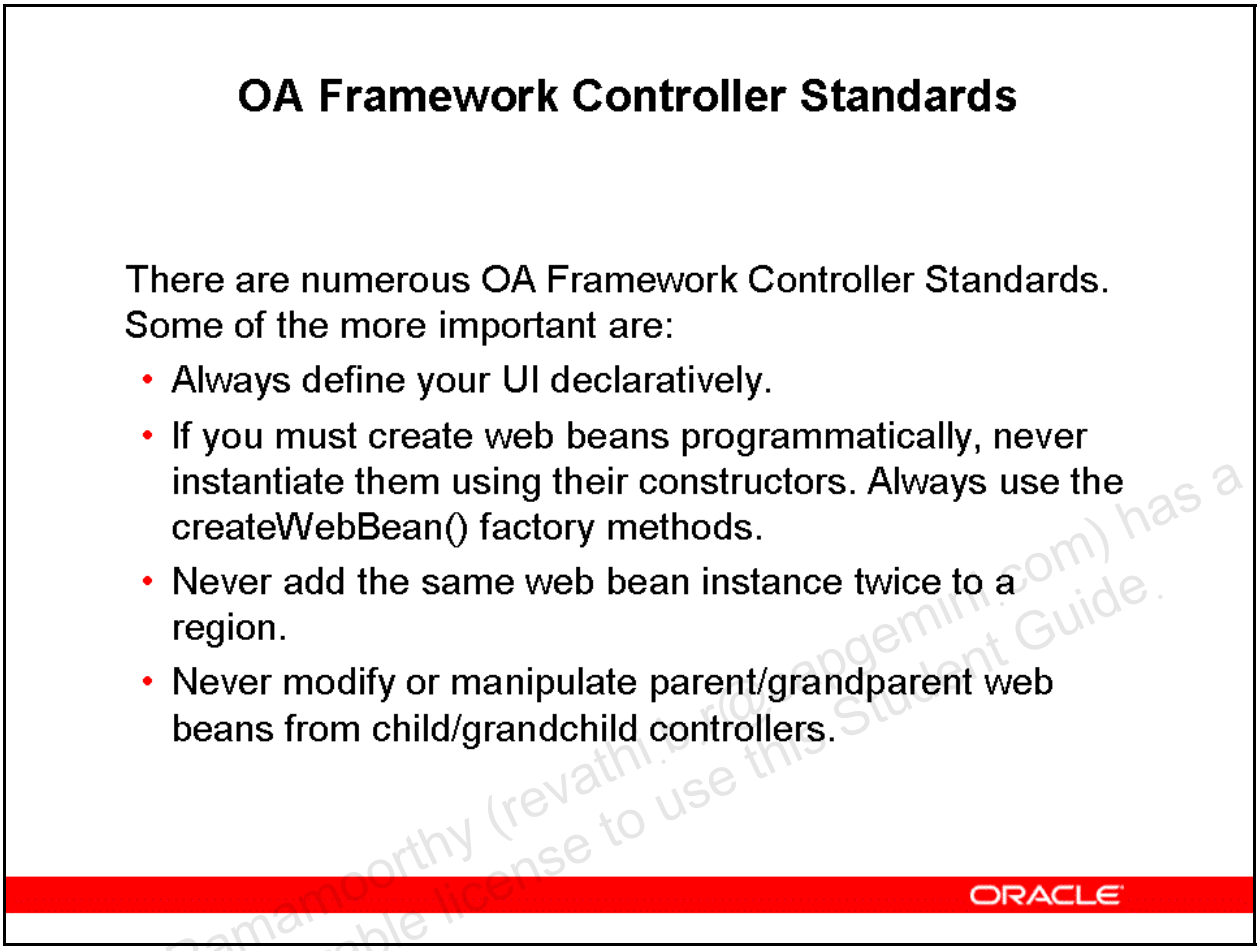




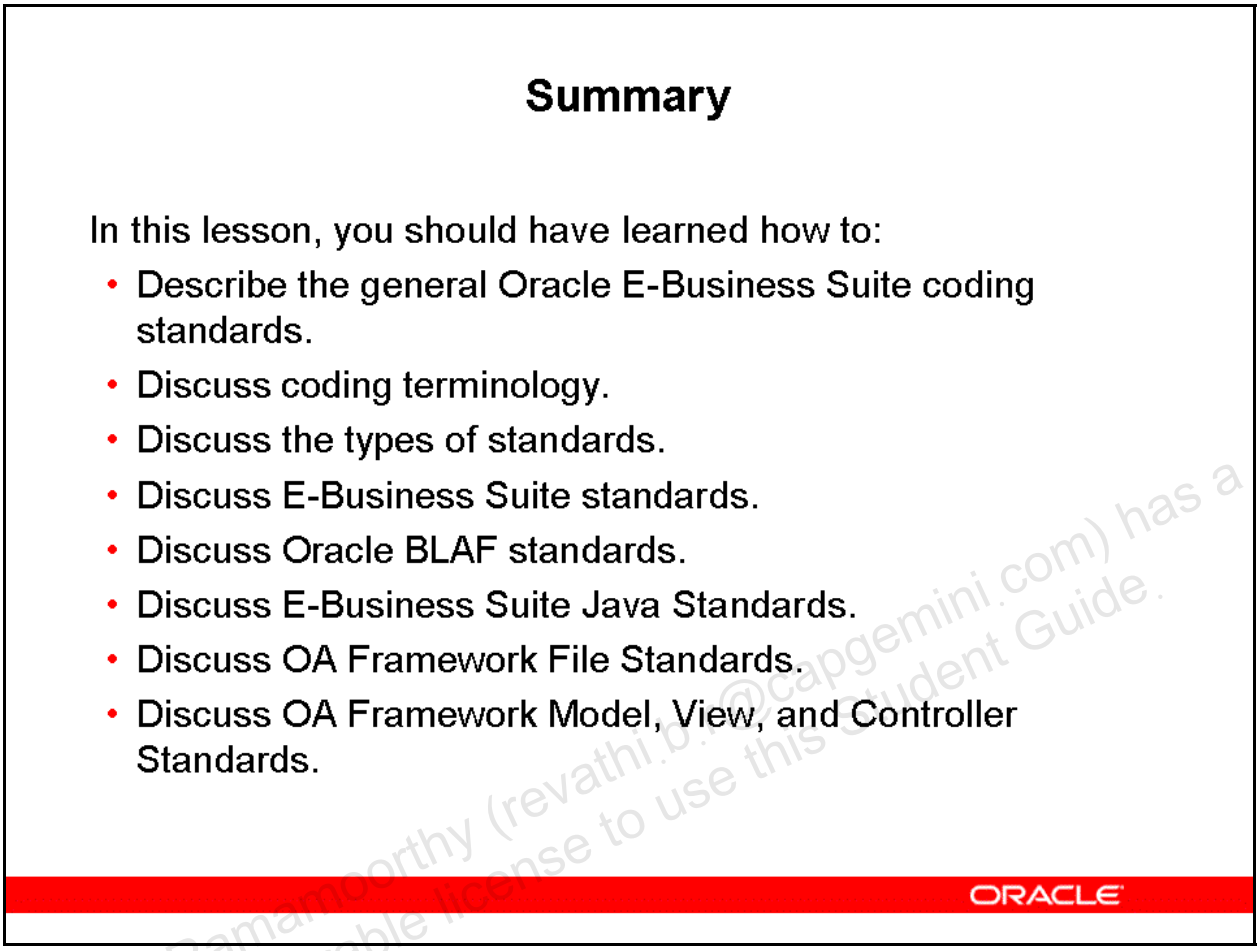
For more information, refer to Chapter 8: OA Framework Model Coding Standards in the OA Framework Developer's Guide.



For more information, refer to Chapter 8: OA Framework View Coding Standards in the OA Framework Developer's Guide.

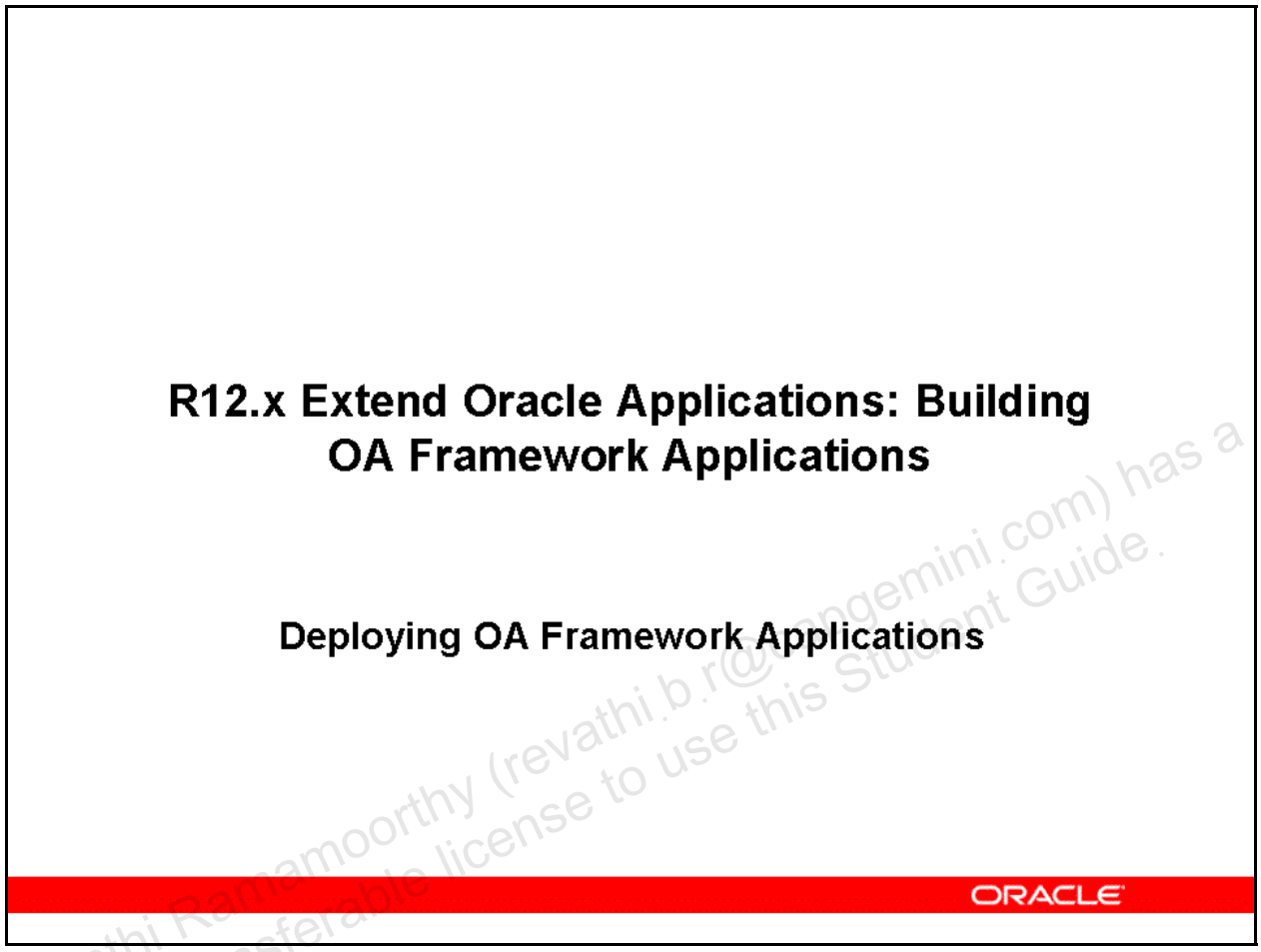


For more information, refer to Chapter 8: OA Framework Controller Coding Standards in the OA Framework Developer's Guide.

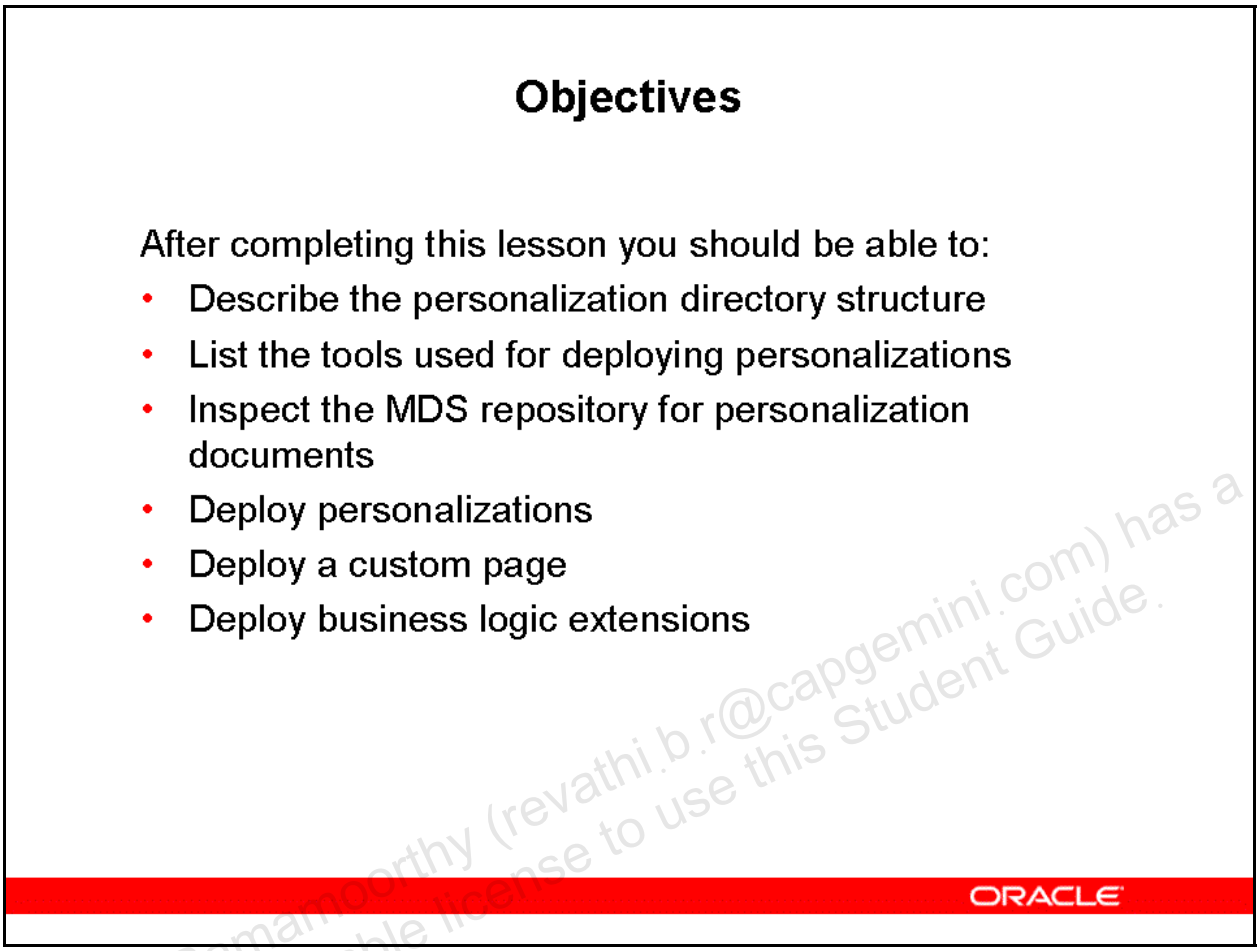


Deploying OA Framework Applications

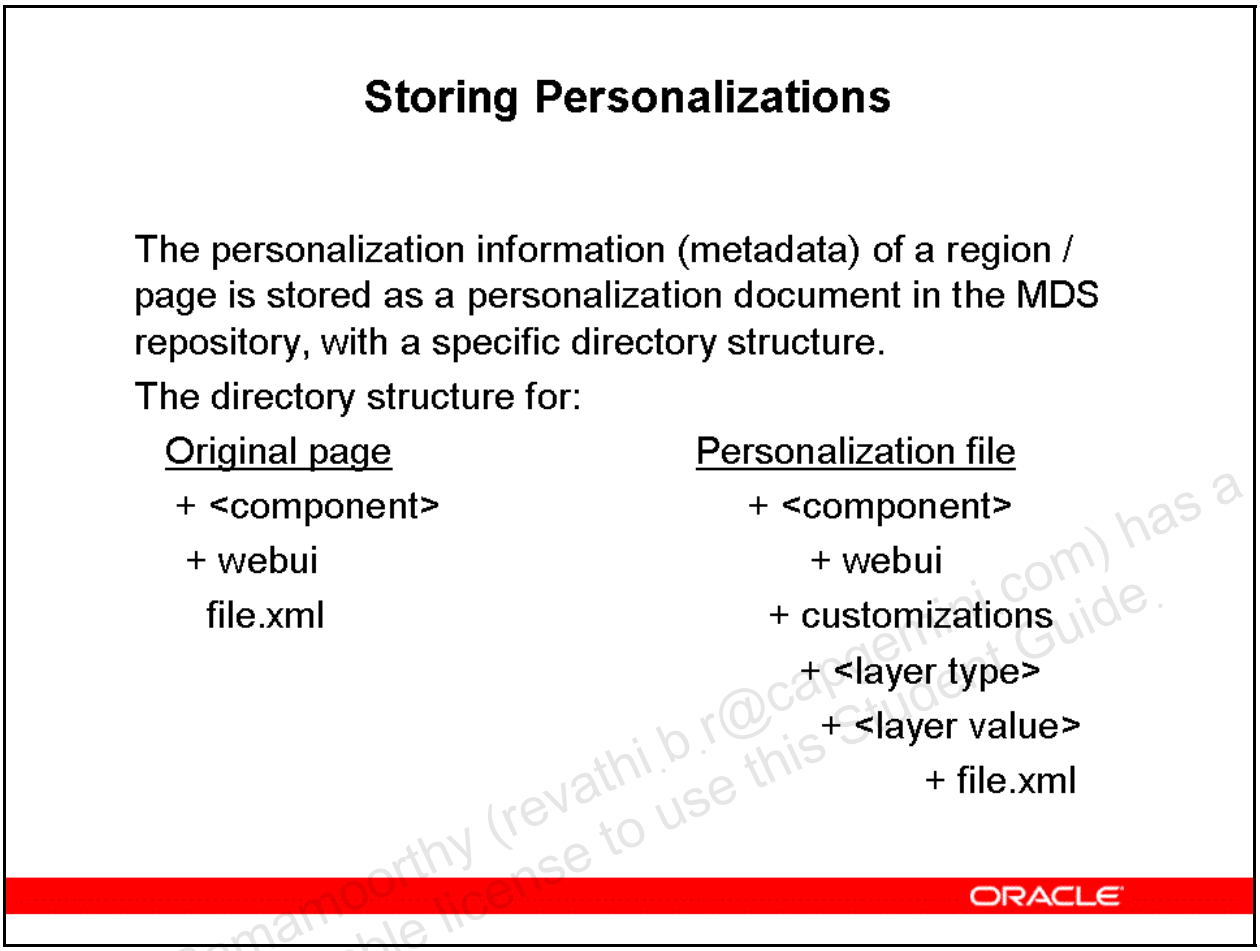
Chapter 14



Objectives



Storing Personalizations



MDS Repository Documents

The MDS Repository is the database that stores the metadata of your OA Framework-based pages and regions, as well as information on any personalizations that have been created in your system. Each metadata definition of a region, page or personalization is called a document.

This document is stored in JDR_PATHS, JDR_COMPONENTS and JDR_ATTRIBUTES tables.

JDR_PATHS – has columns that store document path.

JDR_COMPONENTS – has columns that store component information.

JDR_ATTRIBUTES – has columns that store attribute information.

Directory Structure

To understand the storage structure in the MDS let's look at the following example:

If the following page is personalized at site level:

```
/oracle/apps/fnd/wf/worklist/webui/AdvancWorklistRG
```

The page will be stored in the MDS repository with the following structure:

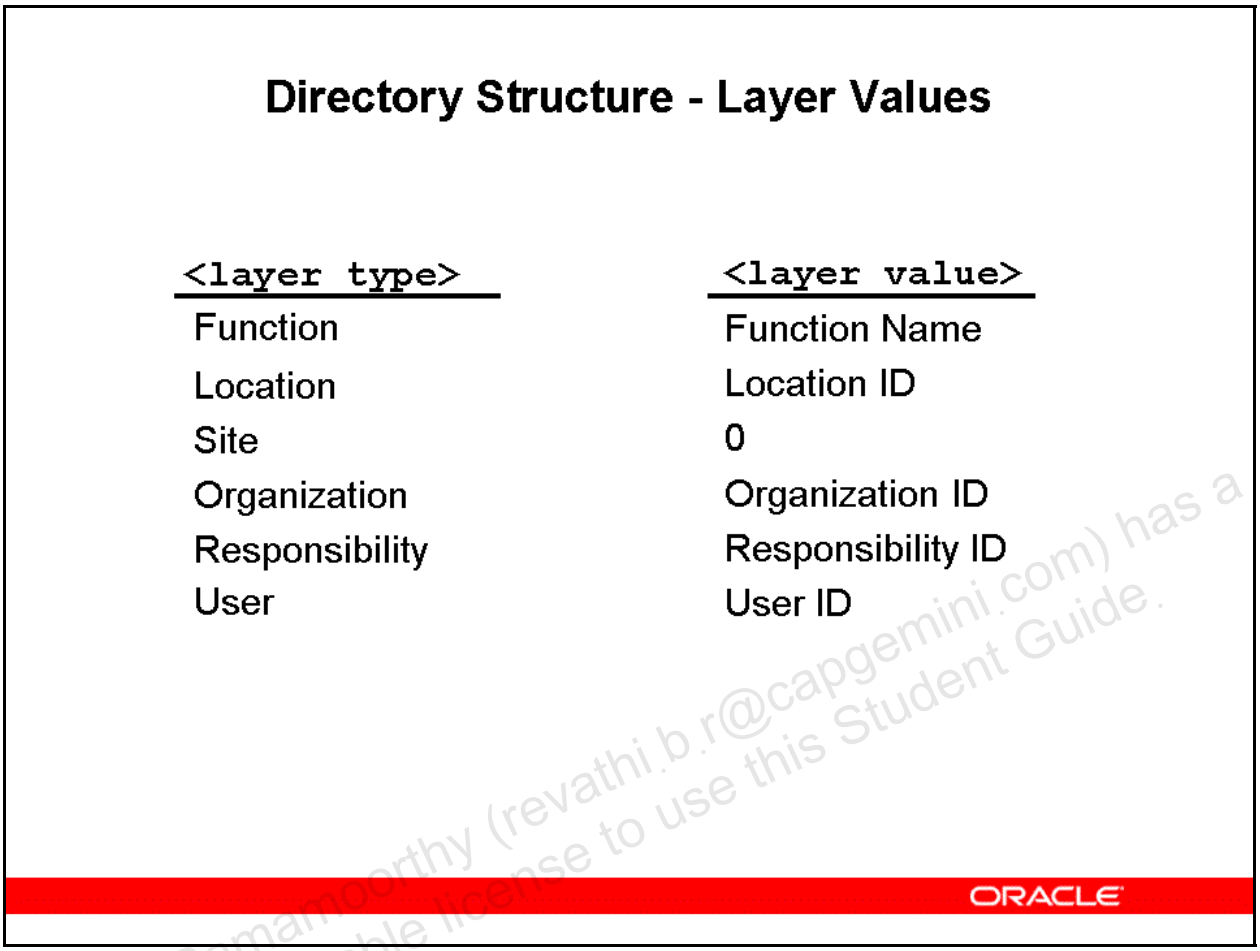
```
/oracle/apps/fnd/wf/worklist/webui/customizations/
site/0/AdvancWorklistRG
```

site - customization layer type

0 – customization layer value

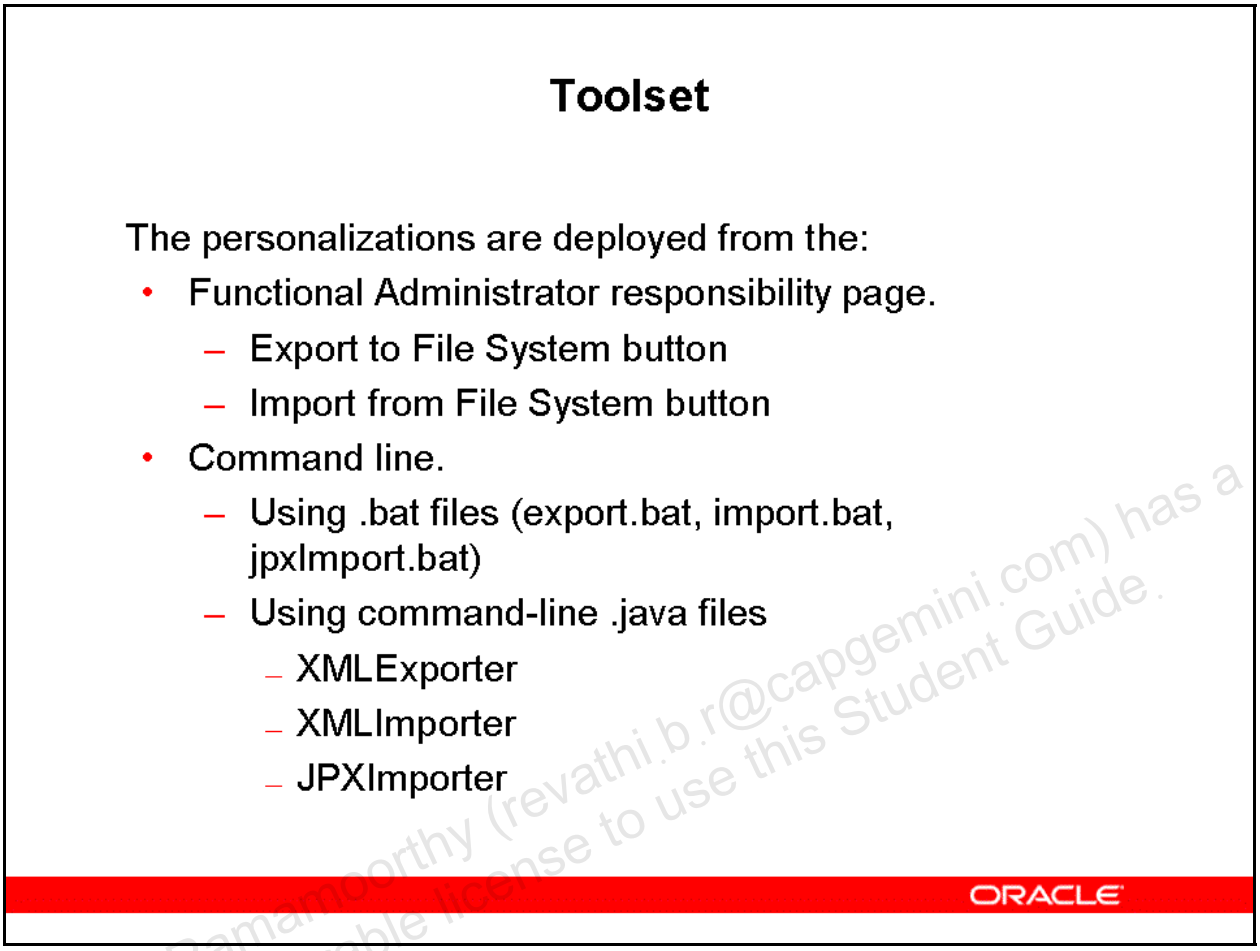
ORACLE

Directory Structure - Layer Values



Note: One quirk of interest. Please notice that the Location, Organization, Responsibility and User all use the ID. There may be cases where IDs exist on your Development system and not in Production, and vice-versa. You must test this carefully when deploying.

Toolset

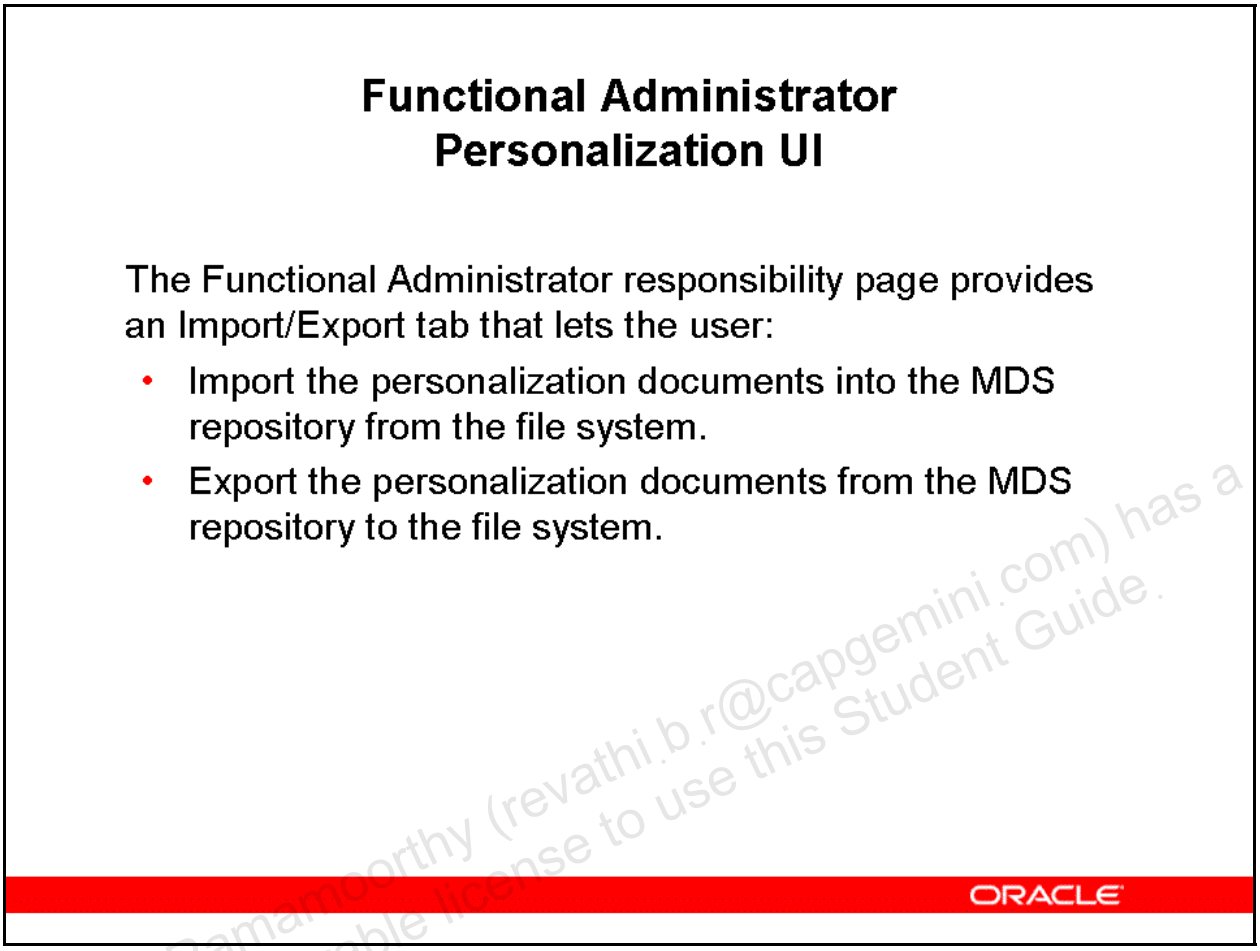


XMLExporter – exports personalization from MDS repository to .xml file.

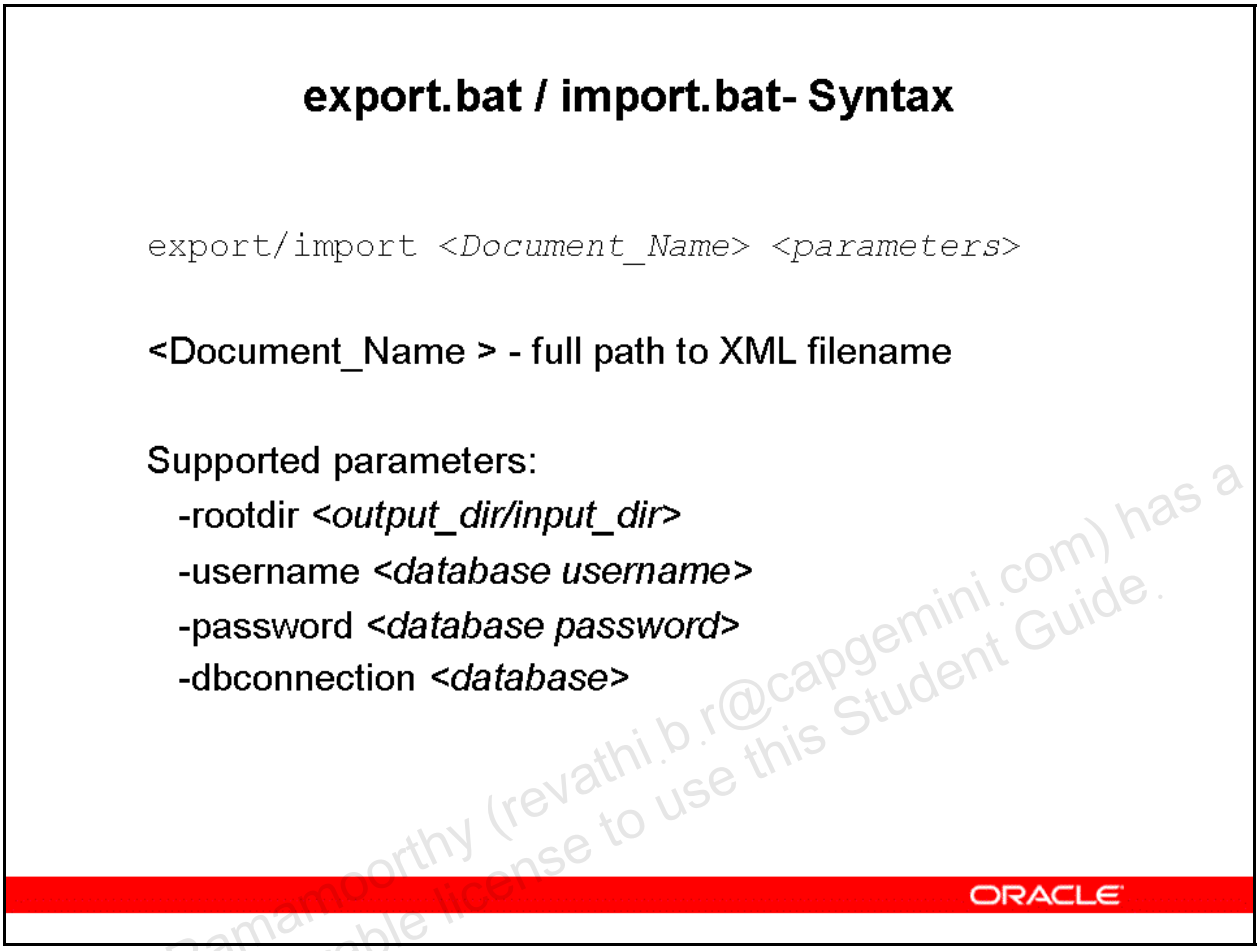
XMLImporter – imports personalizations into MDS repository.

JPXImporter – imports substitutions specified in the .jpx file to the MDS repository.

The export/import batch files are shipped along with OA Extension to Jdeveloper. These batch files have the script to run the XMLExporter and XMLImporter java files.



You can only work with Personalizations from this UI. It does not allow the import/export of OA Framework pages or BC4J substitutions.



-rootdir = directory where the exported xml file needs to be stored
-username = username for the database to export from
-password = password for the database to export from
-dbconnection = connection for the MDS

The –dbconnection information can actually be in one of two formats:

Format 1: The long format

You can put the entire TNS Names entry. For example, your entry might read as follows:

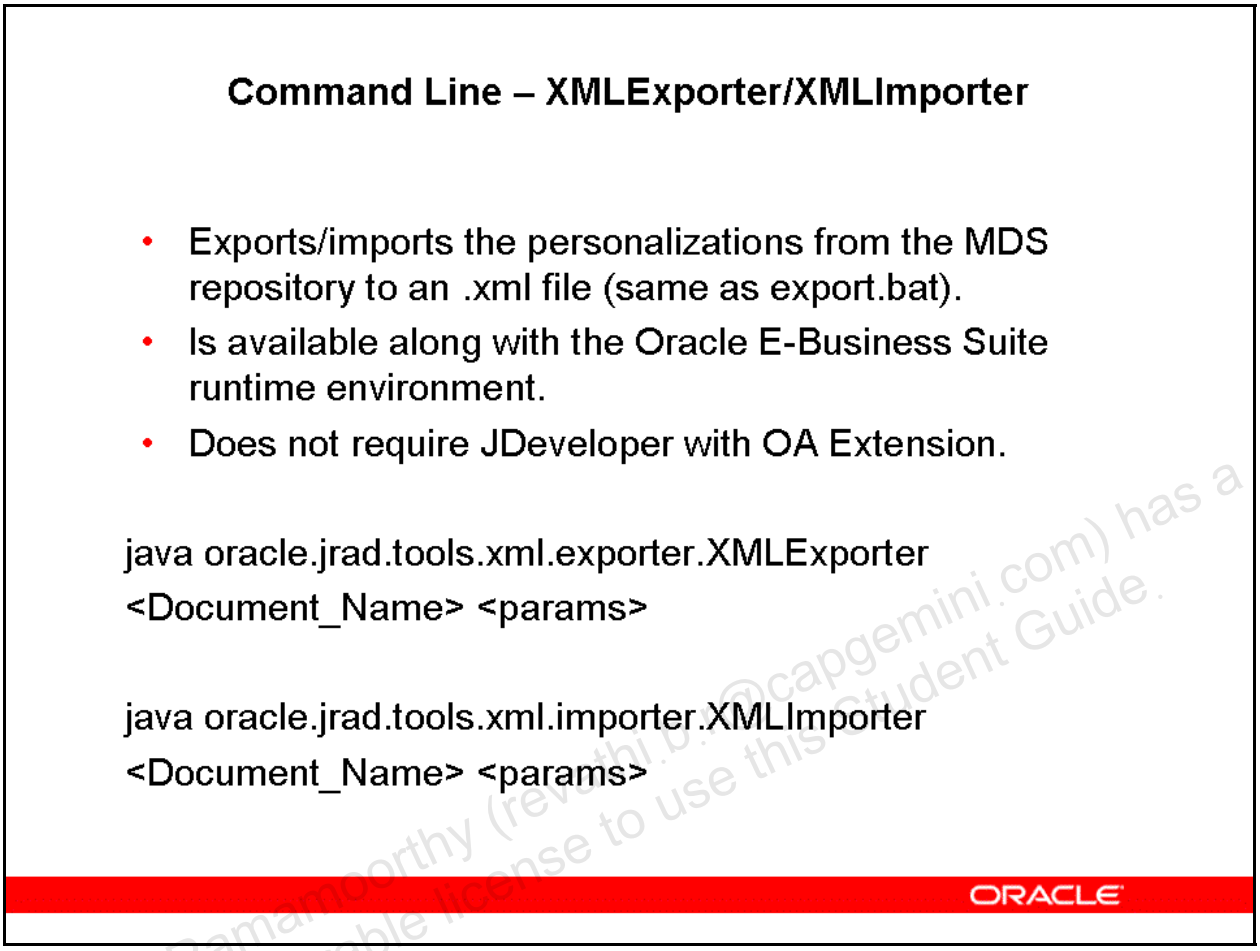
```
-dbconnection "(description = (address_list = (address =
   (community = tcp.world)(protocol = tcp) (host
   =es0006.oracle.com)(port = 1521))) (connect_data = (sid =
   es0006)))"
```

Format 2: The short format (preferred)

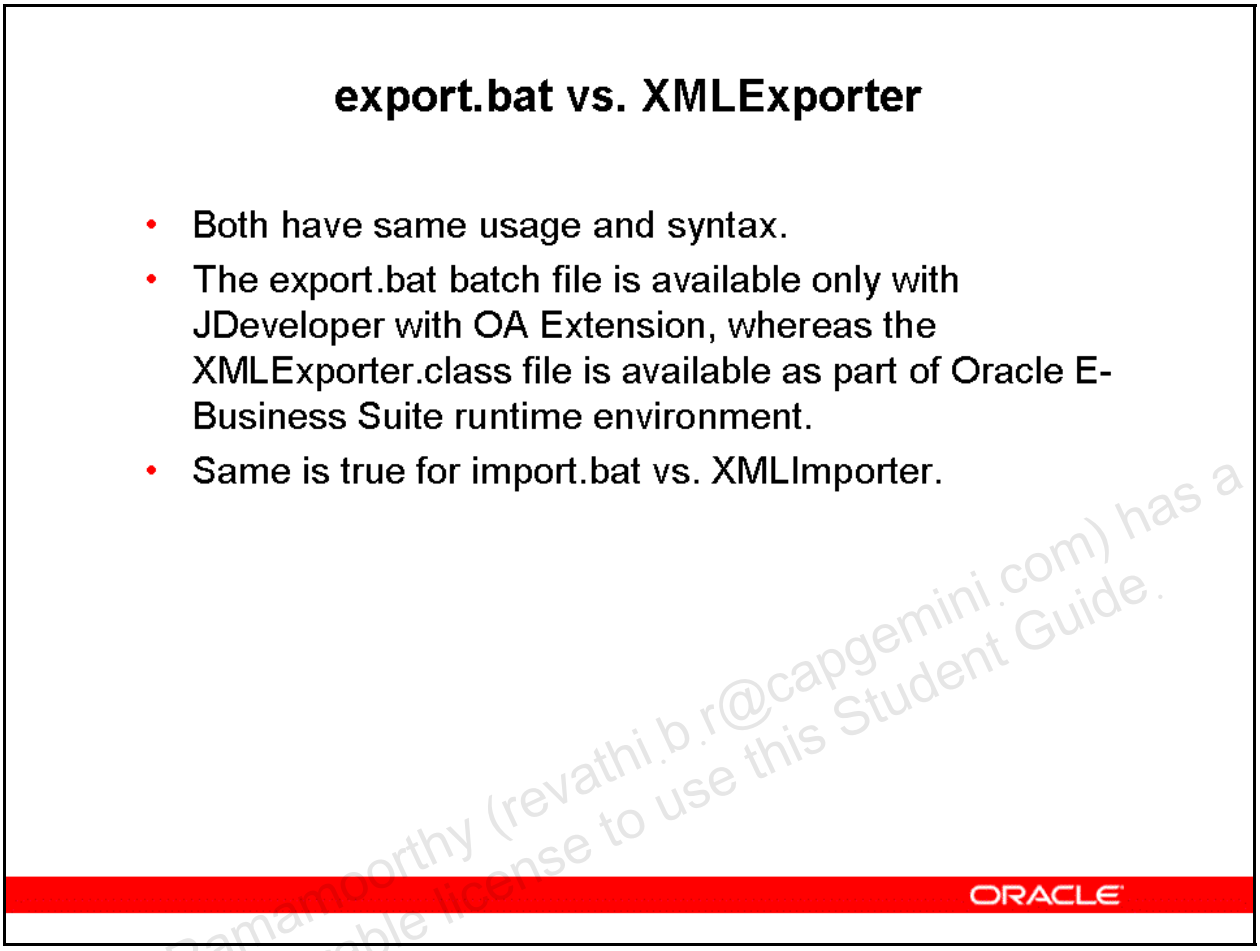
You can put an entry similar to the following:

```
-dbconnection es0006.oracle.com:1521:es0006
```

Command Line – XMLExporter/XMLImporter



This tool can be run from either the JDeveloper with OA Extension install, or it can be run on the Admin Server where the files will be stored.

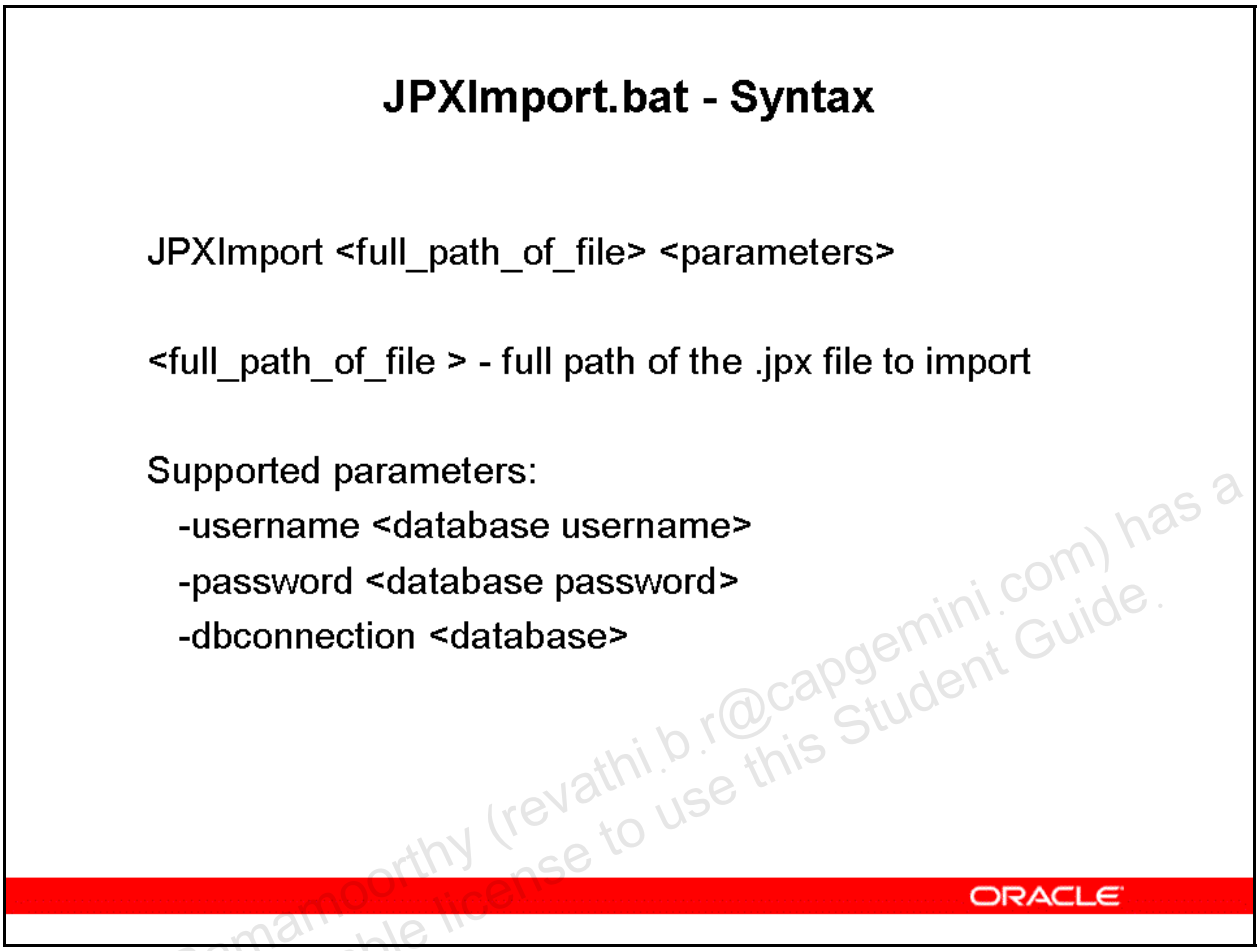


Import Substitutions – JPXImport.bat

- JPXImport.bat is used to import the BC4J substitution documents into the MDS repository.
- Is available as part of JDeveloper with OA Extension and has the script that runs the JPXImporter.java file.
- JPXImport:
  - Parses the .jpx file for each BC4J substitution defined in the file.
  - Transforms each substitution into a separate site level customization document.
  - Imports the customization document into the MDS repository.

ORACLE

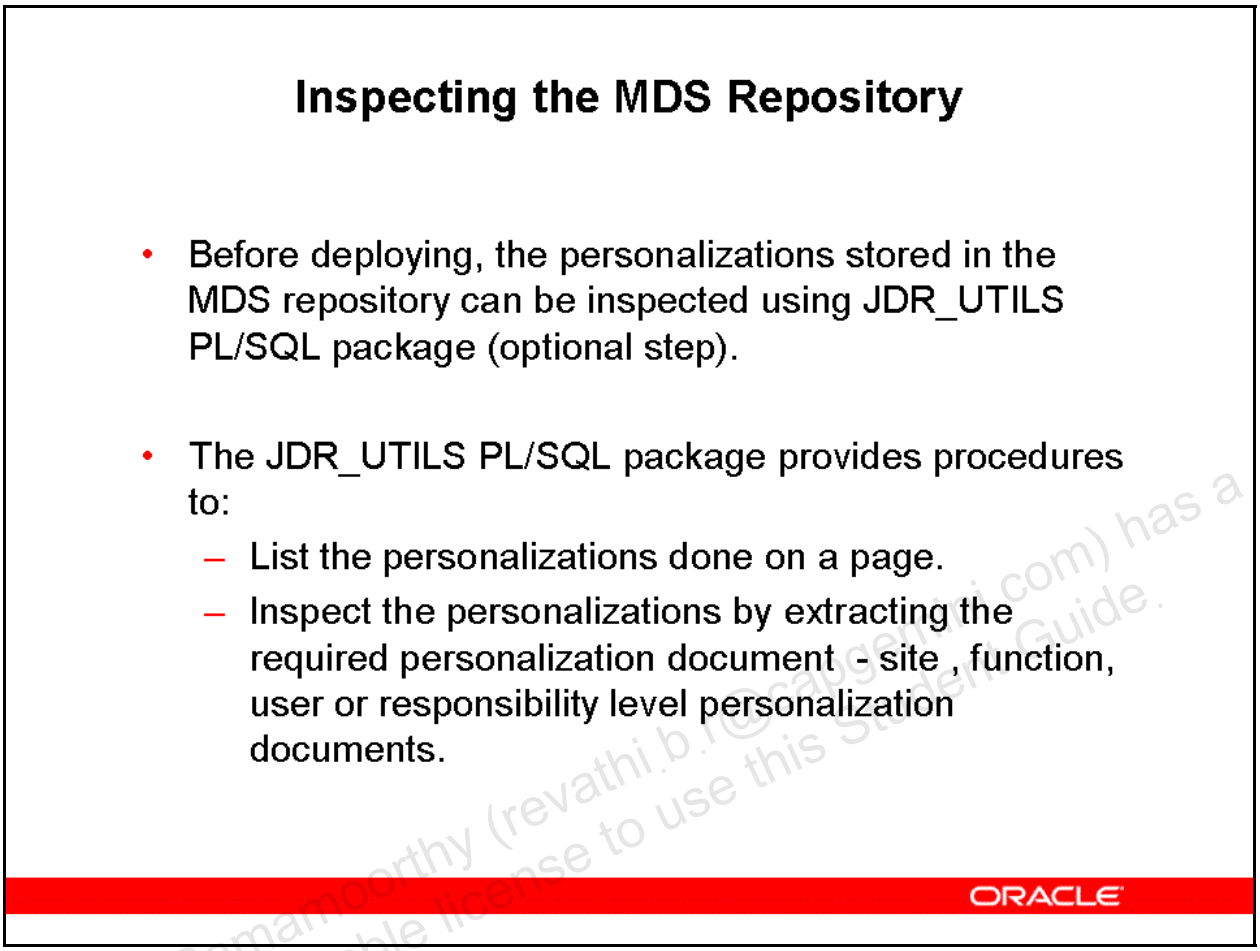
JPXImport.bat - Syntax



-username = username for the database to import to
-password = password for the database to import to
-dbconnection = connection for the database to import to, in tnsnames format

Import Substitutions – JPXImporter

- JPXImporter is used to import the BC4J substitution documents into the MDS repository (same as the JPXImport.bat).
- Is available as part Oracle E-Business Suite runtime environment.
- Does not require JDeveloper with OA Extension.



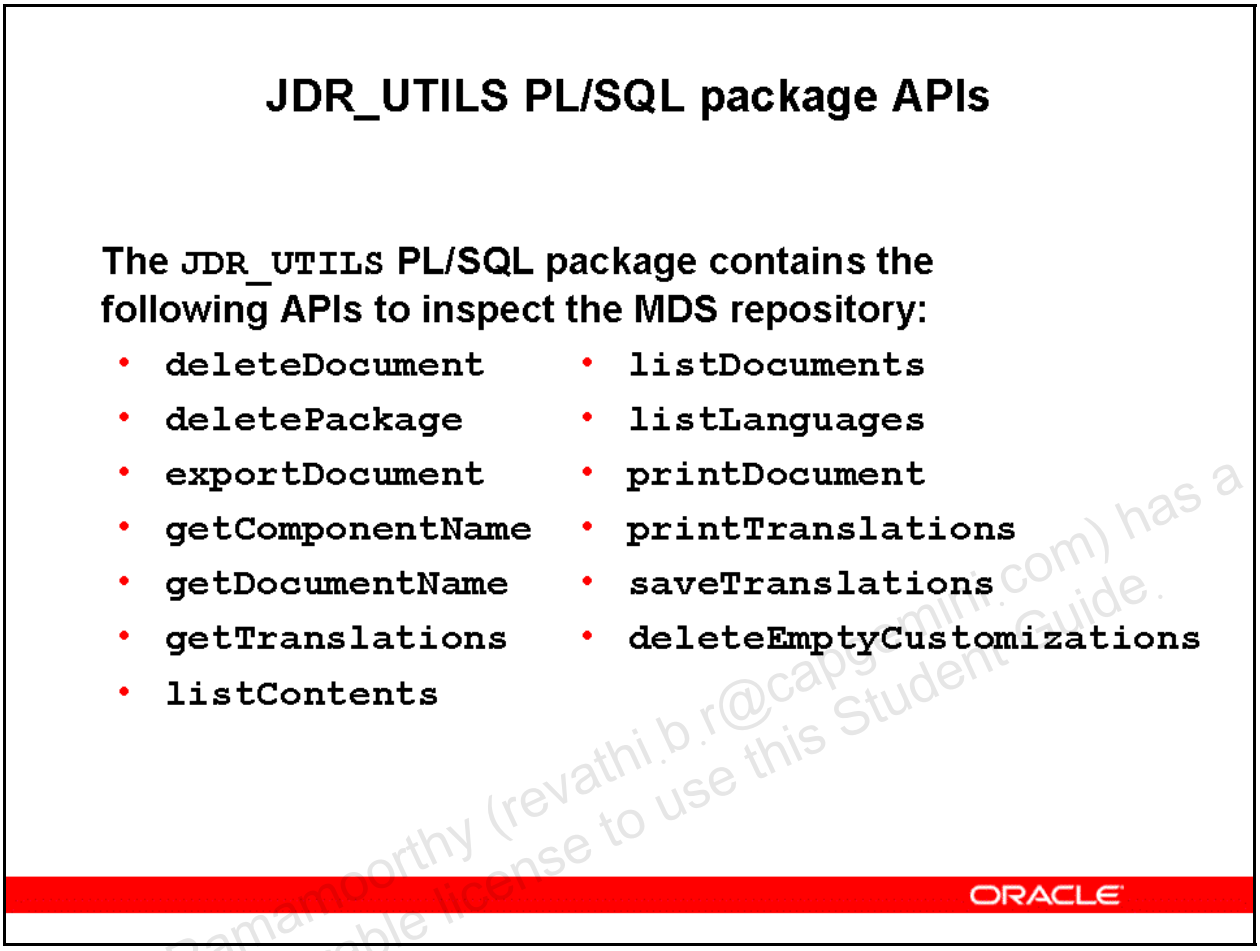
Using JDR_UTILS in SQL*Plus

You can use SQL*Plus to review all the personalizations for a given base document. JDR_UTILS is a PL/SQL package that allows you to evaluate the list of personalization documents that are in your MDS repository. Included in this package is a procedure called JDR_UTILS.listcustomizations( ); which allows you to see the personalization document path names that are currently defined in the MDS repository.

See "Inspecting the MDS Repository Content" in the OA Framework Developer's Guide for more information on the JDR_UTILS package.

This package should never be called from any program code.

JDR_UTILS PL/SQL package APIs




```
PROCEDURE deleteDocument(p_document VARCHAR2);
  -- Deletes all empty customization documents from the repository. An empty
  -- customization document is a customization document that does not
specify
  -- any modifications to the base metadata.
  --
  -- Example 1: /oracle/apps/hr/customizations/localization/US/page1
  -- <customization customizes="/oracle/apps/hr/page1"
  --                xmlns="http://xmlns.oracle.com/jrad"
  --                xmlns:ui="http://xmlns.oracle.com/uix/ui"
  --                xmlns:oa="http://xmlns.oracle.com/oa"
  --                xmlns:user="http://xmlns.oracle.com/jrad/user"
  --                file-version="$Header: JDRUTEXS.pls 120.3 2005/10/26
06:16:00 akbansal noship $" version="9.0.3.6.6_557"
  --                xml:lang="en-US">
  --    <modifications/>
  -- </customization>
  --
  -- Example 2: /oracle/apps/hr/customizations/user/100/page1
```

```
  -- <customization customizes="/oracle/apps/hr/page1"
  --                xmlns="http://xmlns.oracle.com/jrad"
  --                xmlns:ui="http://xmlns.oracle.com/uix/ui"
  --                xmlns:oa="http://xmlns.oracle.com/oa"
  --                xmlns:user="http://xmlns.oracle.com/jrad/user"
  --                file-version="$Header: JDRUTEXS.pls 120.3 2005/10/26
06:16:00 akbansal noship $" version="9.0.3.6.6_557"
  --                xml:lang="en-US">
  --   <views>
  --     <view name="MyTest10" description="my view"
  --           id="view1" element="Region1">
  --       <modifications/>
  --     </view>
  --   <views/>
  -- <customization/>

PROCEDURE deleteEmptyCustomizations;
  -- Deletes the package from the repository if the package is empty.  If
the
  -- package is not empty (i.e. it contains either documents or packages),
then
  -- an error will be issued indicated that non-empty packages can not be
  -- deleted.
  --
  -- Parameters:
  --  p_package    - the fully qualified package name
  --                 (i.e.  '/oracle/apps')

PROCEDURE deletePackage(p_package VARCHAR2);
  -- Export the XML for a document and pass it back in 32k chunks.  This
  -- function will return XML chunks, with a maximum size of 32k.
  --
  -- Specifying a document name will initiate the export.  Thereafter, a
NULL
  -- document name should be passed in until the export is complete.
  -- That is, to export an entire document, you should do:
  --
  -- firstChunk := jdr_utils.exportDocument('/oracle/apps/fnd/mydoc',
isDone);
  -- WHILE (NOT isDone) LOOP
  --   nextChunk := jdr_utils.exportDocument(NULL, isDone);
  -- END LOOP;
  --
  -- Parameters:
```

```
  --   p_document       - the fully qualified name of the document.
However,
  --                      after the first chunk of text is exported, a NULL
  --                      value must be passed in to retrieve the next
  --                      chunks.
  --
  --   p_exportFinished - OUT parameter which indicates whether or not the
export
  --                      is complete.  TRUE indicates the entire document
is
  --                      exported, FALSE indicates that there are more
chunks
  --                      remaining.
  --
  --   p_formatted      - TRUE indicates that the XML is formatted nicely
  --                      (i.e. whether or not the elements are indented).
  --                      This is defaulted to TRUE.
  --
  --
  -- Returns:
  --   The exported XML, in 32k chunks.
  --
  -- Notes:
  --   As this function relies on package state, it is not possible to
export
  --   multiple documents at the same time.  A document must be finished
  --   exporting before a new document can be exported.

  FUNCTION exportDocument(
    p_document           VARCHAR2,
    p_exportFinished OUT NOCOPY /* file.sql.39 change */ BOOLEAN,
    p_formatted          BOOLEAN DEFAULT TRUE) RETURN VARCHAR2;
  -- Gets the fully qualified name of the component.
  --
  -- Parameters:
  --  p_docid       - the ID of the document which contains the component
  --
  --  p_compid      - the ID of the component (from comp_id in the
  --                  jdr_components table

FUNCTION getComponentName(
    p_docid  jdr_paths.path_docid%TYPE,
    p_compid jdr_components.comp_id%TYPE) RETURN VARCHAR2;
  -- Gets the fully qualified name of the document.
```

```
  --
  -- Parameters:
  --  p_docid       - the ID of the document

FUNCTION getDocumentName(
    p_docid  jdr_paths.path_docid%TYPE) RETURN VARCHAR2;
  -- Gets all of the translations of the specified document.
  --
  -- Parameters:
  --  p_document    - the fully qualified document name
  --
  -- Raises NO_SUCH_DOCUMENT exception if the document does not exist.

FUNCTION getTranslations(
    p_document VARCHAR2) RETURN translationList;
  -- Prints the contents of a package.
  --
  -- For the non-recursive case, this will list the documents,
  -- package files and package directories.
  --
  -- For the recursive case, this will list the document, package files
  -- and empty package directories (i.e. packages which contain no documents
  -- or child packages).
  --
  -- In order to diferentiate documents from package directories, package
  -- directories will end with a '/'.
  --
  -- Parameters:
  --  p_path       - The path in which to list the documents.  To specify
  --                 the root directory, use '/'.
  --
  --  p_recursive  - If TRUE, recursively lists the contents of
  --                 sub-directories.  Defaults to FALSE.
  --
  -- To use this from SQL*Plus, do:
  --
  -- (1) set serveroutput on
  --     execute jdr_utils.listContents('/oracle/apps/ak');
  --     This will list the contents of the ak directory, without showing
  --     the contents of the sub-directories.
  --
  -- (2) set serveroutput on
  --     execute jdr_utils.listContents('/', TRUE);
```

```
  --     This will list the contents of the entire repository.
  --     sub-directories.

PROCEDURE listContents(p_path VARCHAR2, p_recursive BOOLEAN DEFAULT FALSE);
  -- List the customizations for the specified document.
  --
  -- Parameters:
  --  p_document    - the fully qualified document name, which can represent
  --                  either a document or package file.
  --                  (i.e.  '/oracle/apps/ak/attributeSets')

PROCEDURE listDocuments(p_path VARCHAR2, p_recursive BOOLEAN DEFAULT FALSE);
  -- Lists the supported languages for the specified document.
  --
  -- Parameters:
  --  p_document    - the fully qualified document name, which can represent
  --                  either a document or package file.
  --                  (i.e.  '/oracle/apps/ak/attributeSets')

PROCEDURE listLanguages(p_document VARCHAR2);
  -- Prints the contents of a JRAD document to the console.
  --
  -- Parameters:
  --  p_document    - the fully qualified document name, which can represent
  --                  either a document or package file.
  --                  (i.e.  '/oracle/apps/ak/attributeSets')
  --
  -- p_maxLineSize - the maximum size of line.  This defaults to 255 which
is
  --                  the maximim allowable size of a line (the 255 limit is
  --                  a limitation of the DBMS_OUPUT package).
  --
  -- Limitations:
  --  Documents larger than 1000000 bytes will fail as DBMS_OUPUT's maximim
  --  buffer is 1000000 bytes.
  --
  -- To use this from SQL*Plus, do:
  --   set serveroutput on format wrapped (this is needed for leading
spaces)
  --   set linesize 100
  --   execute jdr_utils.printDocument('/oracle/apps/ak/attributeSets',
100);
  --
  -- To create an XML file, you can create the following SQL file:
```

```
  --   set feedback off
  --   set serveroutput on format wrapped
  --   set linesize 100
  --   spool (parameter 1)
  --   execute jdr_utils.printDocument('(parameter 2)', 100);
  --   spool off
  --
  -- and call the file with:
  --   sqlplus scott/tiger @export.sql myxml.xml
/oracle/apps/ak/attributeSets

PROCEDURE printDocument(p_document    VARCHAR2,
                        p_maxLineSize NUMBER DEFAULT MAX_LINE_SIZE);
  -- Prints the translations for the document in XLIFF format.
  --
  -- Parameters:
  --  p_document    - the fully qualified document name, which can represent
  --                  either a document or package file.
  --                  (i.e.  '/oracle/apps/ak/attributeSets')
  --
  --  p_language    - the language to use for the translations
  --
  --  p_maxLineSize - the maximum size of line.  This defaults to 255 which
is
  --                  the maximim allowable size of a line (the 255 limit is
  --                  a limitation of the DBMS_OUPUT package).
  --
  -- To use this from SQL*Plus, do:
  --   set serveroutput on format wrapped (this is needed for leading
spaces)
  --   set linesize 100
  --   execute jdr_utils.printTranslations('/oracle/apps/ak/attributeSets',
  --                                       'mylanguage', 100);

PROCEDURE printTranslations(p_document    VARCHAR2,
                        p_language    VARCHAR2,
                        p_maxLineSize NUMBER DEFAULT MAX_LINE_SIZE);
  -- Saves the specified translations for the specified document.
  --
  -- This procedure will do the following:
  --  (1) Lock the document so as to prevent multiple users attempting
  --      to modify translations at the same time
  --  (2) Delete all of the translations for the specified document
  --  (3) Insert the new translations
```

```
  --  (4) Commit the data unless p_commit set to FALSE
  --
  -- Please use this with care as it will delete all of the
  -- translations for the specified document.
  --
  -- Parameters:
  --  p_document     - the fully qualified document name
  --
  --  p_translations - the list of translations to insert
  --
  --  p_commit       - if TRUE, the data is committed.  Default is TRUE
  --
  -- NOTE: If p_commit is set to FALSE, then the document will remain locked
  -- after the call to saveTranslations.  In order to prevent a deadlock, a
  -- commit (or rollback) must occur to unlock the document.
  --
  -- Raises NO_SUCH_DOCUMENT exception if the document does not exist.

PROCEDURE saveTranslations(
                      p_document     VARCHAR2,
                      p_translations translationList,
                      p_commit       BOOLEAN := TRUE);
```

List the Personalizations Done on a Page

List the Personalizations Done on a Page

Use the PL/SQL procedure JDR_UTILS.listContents to list the personalizations done on a page.

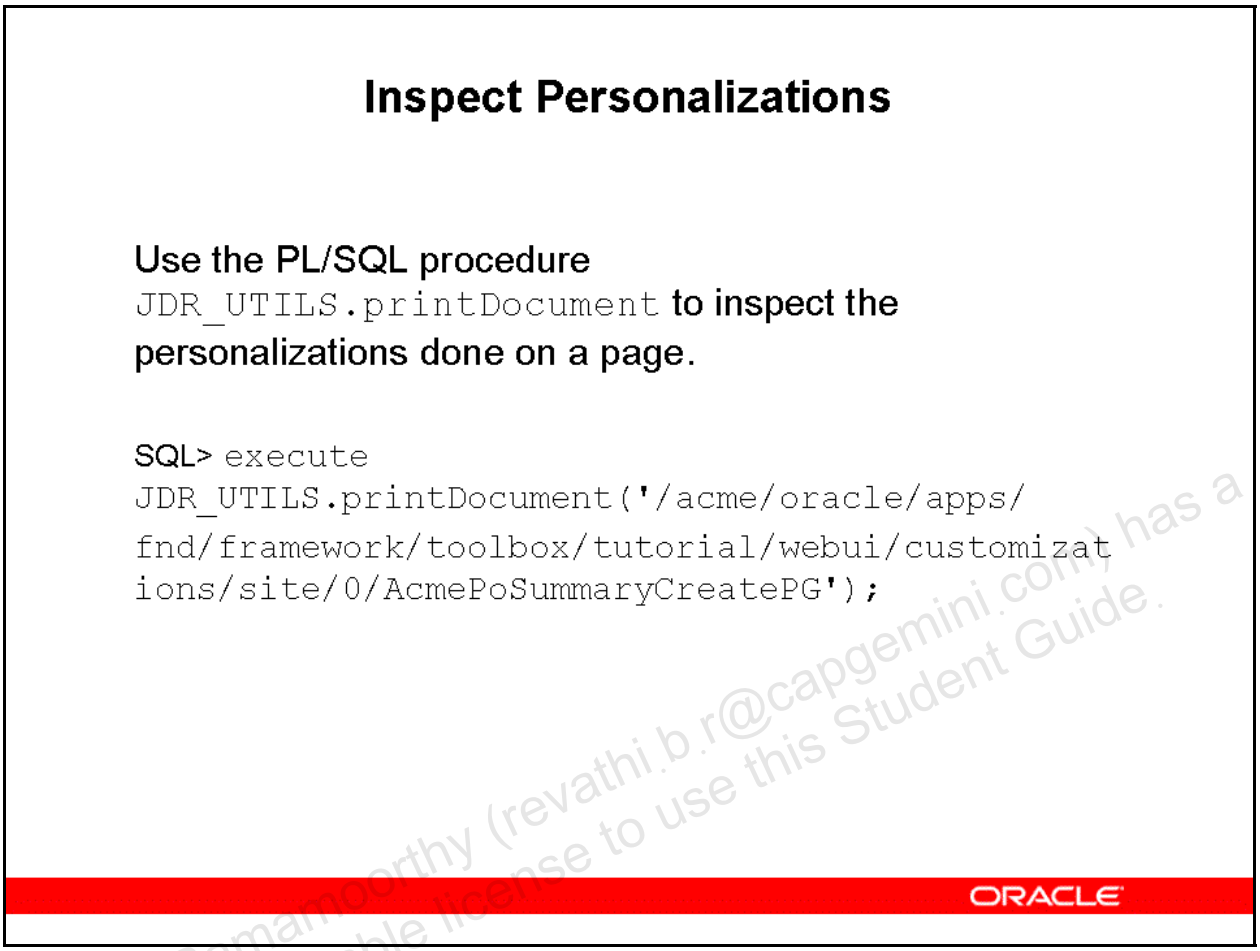
```
SQL> set serveroutput on;
SQL> execute JDR_UTILS.listContents('/acme',TRUE);

Printing contents of /acme recursively
/acme/oracle/apps/fnd/framework/toolbox/tutorial/
webui/customizations/site/0/AcmePoSummaryCreatePG

/acme/oracle/apps/fnd/framework/toolbox/tutorial/
webui/acmePoSummaryCreatePG
PL/SQL procedure successfully completed.
```

ORACLE

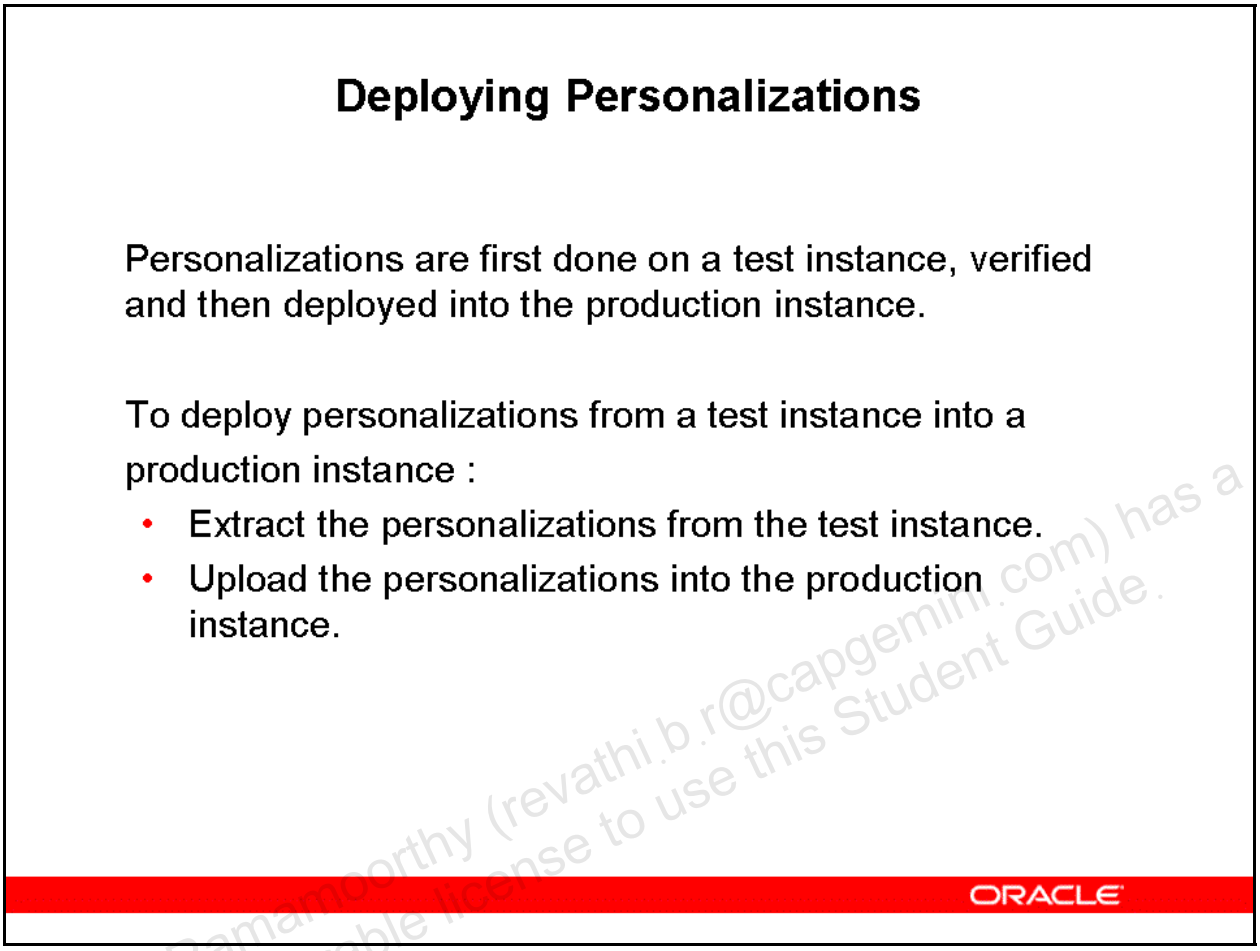
Inspect Personalizations

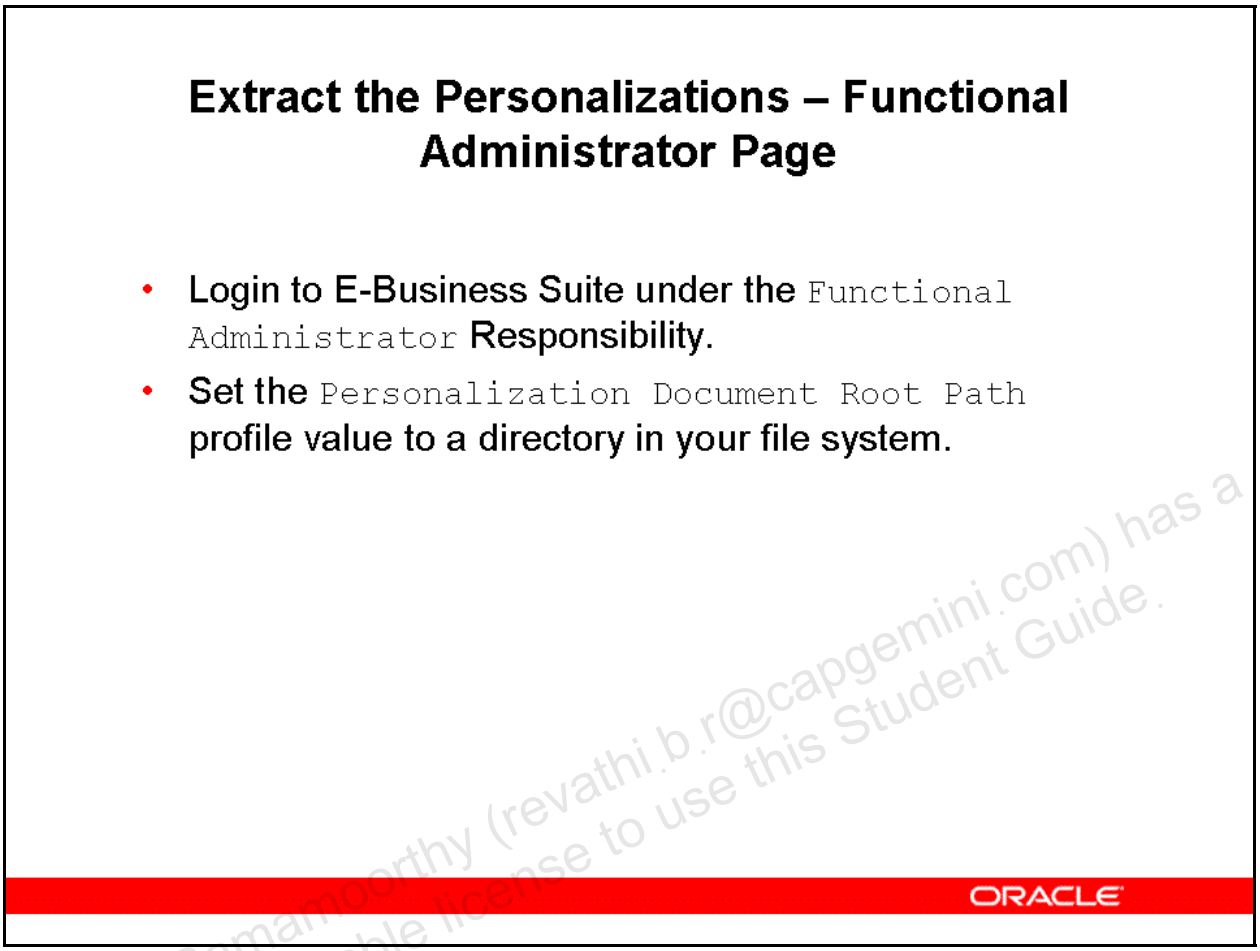


The above mentioned command will produce the following output.

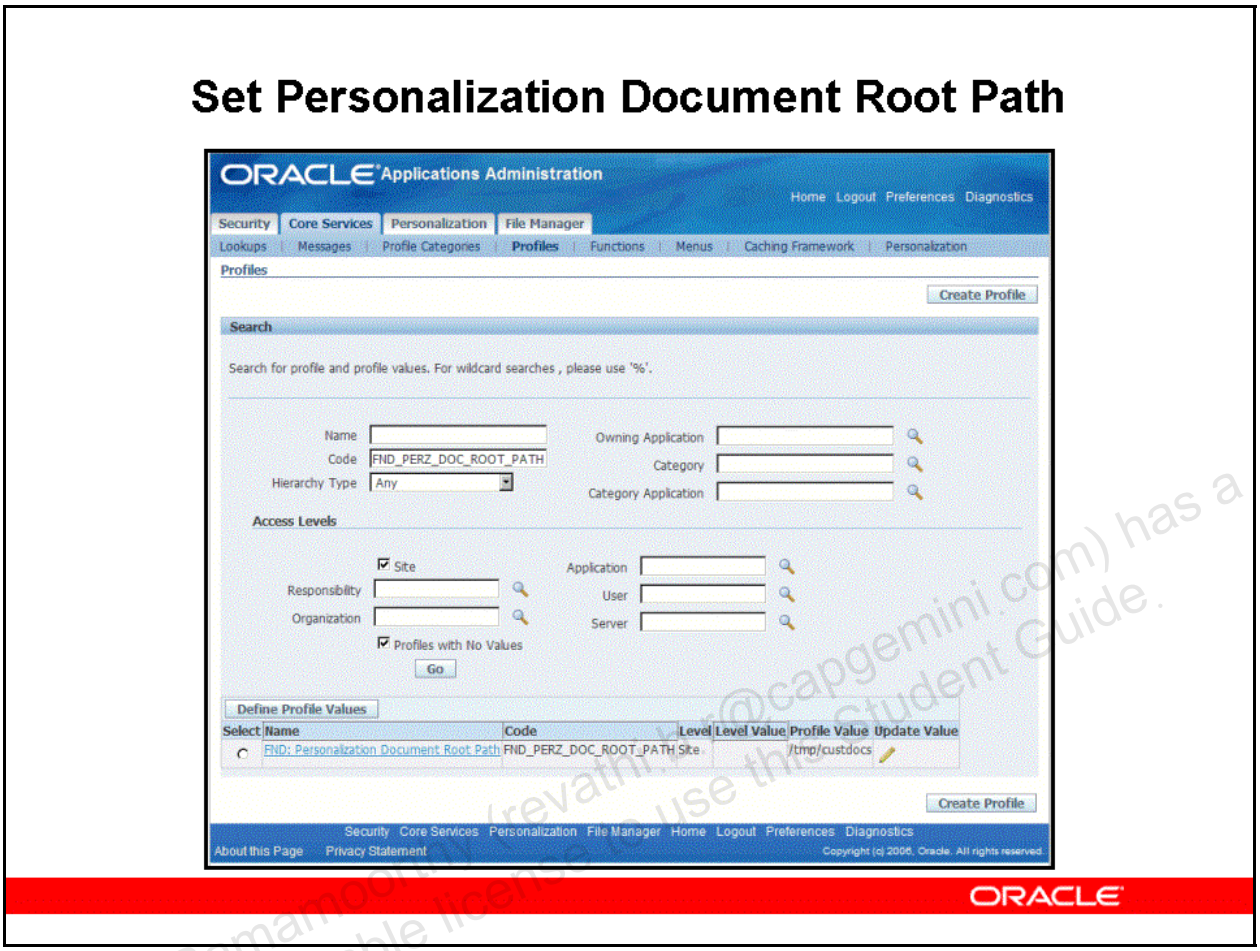
```
<?xml version='1.0' encoding='UTF-8'?>
<customization xmlns="http://xmlns.oracle.com/jrad" version="9.0.5.4.79_479"
xml:lang="en-US"
customizes="/acme/oracle/apps/fnd/framework/toolbox/tutorial/webui/AcmePoSum
maryCreatePG"
xmlns:oa="http://xmlns.oracle.com/oa">
<modifications>
<move element="OrdersRN.Status" after="OrdersRN.OrderNum"/>
…
<move element="OrdersRN.Description" after="OrdersRN.Status"/>
<move element="OrdersRN.Buyer" after="OrdersRN.Description"/>
<move element="OrdersRN.Supplier" after="OrdersRN.Buyer"/>
<move element="OrdersRN.Currency" after="OrdersRN.Supplier"/>
<insert before="OrdersRN.Currency">
<oa:messageStyledText id="SiteName" adminCustomizable="true"
cellNoWrapFormat="false"
dataType="VARCHAR2" initSortSeq="none" prompt="Supplier Site"
queryable="false" rendered="true"
```

```
required="no" scope="." selectiveSearchCriteria="false"
serverUnvalidated="false" sortState="no" tipType="none" totalValue="false"
userCustomizable="false"
vAlign="middle" viewAttr="SiteName" viewName="PoSummaryVO1"
warnAboutChanges="true"/>
…
</insert>
<move element="OrdersRN.OrderTotal" after="OrdersRN.Currency"/>
<move element="OrdersRN.DeleteSwitcher" after="OrdersRN.OrderTotal"/>
      <move
element="OrdersRN.UpdateSwitcher" after="OrdersRN.DeleteSwitcher"/>
</modifications>
</customization>
```





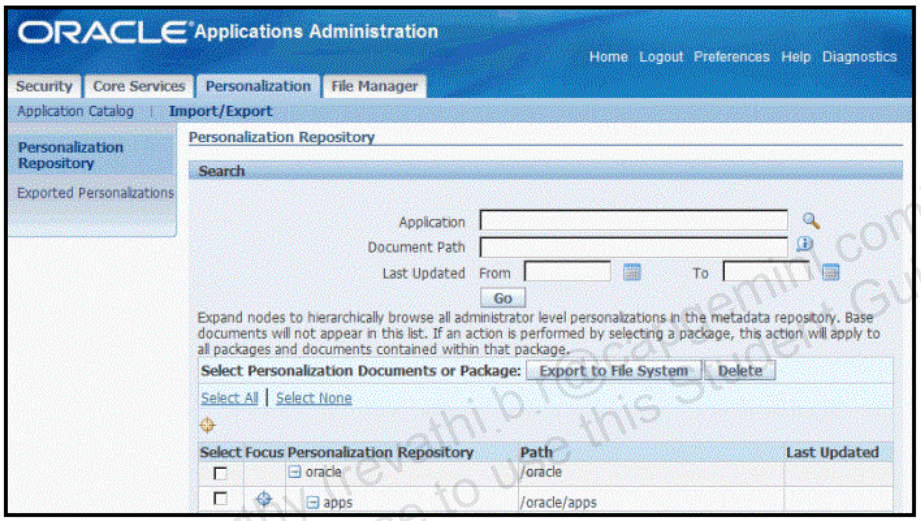
Set Personalization Document Root Path



The full Internal Profile Code is FND_PERZ_DOC_ROOT_PATH.

Import/Export Personalizations

- From the Personalizations tab, select the `Import/Export` subtab.



Extract the Personalizations – Select the Page

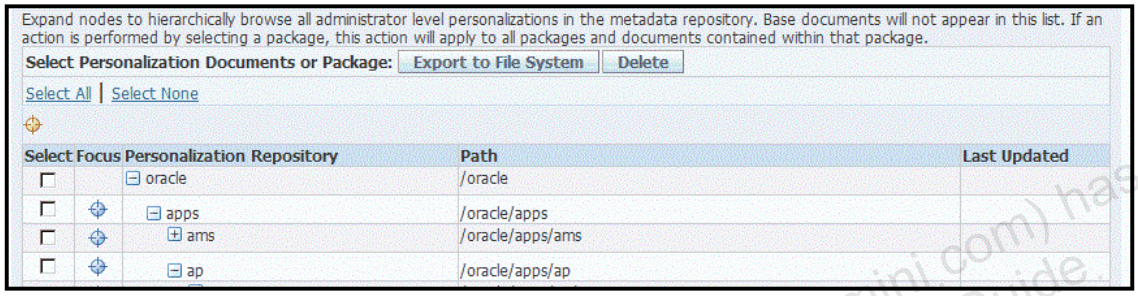
- Expand the `Personalization Repository` view and select the page that needs to be exported to the file system.

Select	Focus	Personalization Repository	Path	Last Updated
☐		⊟ oracle	/oracle	
☐	✥	⊟ apps	/oracle/apps	
☐	✥	⊞ ams	/oracle/apps/ams	
☐	✥	⊟ ap	/oracle/apps/ap	
☐	✥	⊞ customizations	/oracle/apps/ap/customizations	
☐	✥	⊟ oie	/oracle/apps/ap/oie	
☐	✥	⊞ audit	/oracle/apps/ap/oie/audit	
☐	✥	⊞ entry	/oracle/apps/ap/oie/entry	
☐	✥	⊟ webui	/oracle/apps/ap/oie/webui	
☐	✥	⊟ customizations	/oracle/apps/ap/oie/webui/customizations	
☐	✥	⊞ function	/oracle/apps/ap/oie/webui/customizations/function	
☐	✥	⊞ responsibility	/oracle/apps/ap/oie/webui/customizations/responsibility	
☐	✥	⊟ site	/oracle/apps/ap/oie/webui/customizations/site	
☐	✥	⊟ 0	/oracle/apps/ap/oie/webui/customizations/site/0	
☐		TrackReportsTblRN	/oracle/apps/ap/oie/webui/customizations/site/0/TrackReportsTblRN	27-September-2004

ORACLE

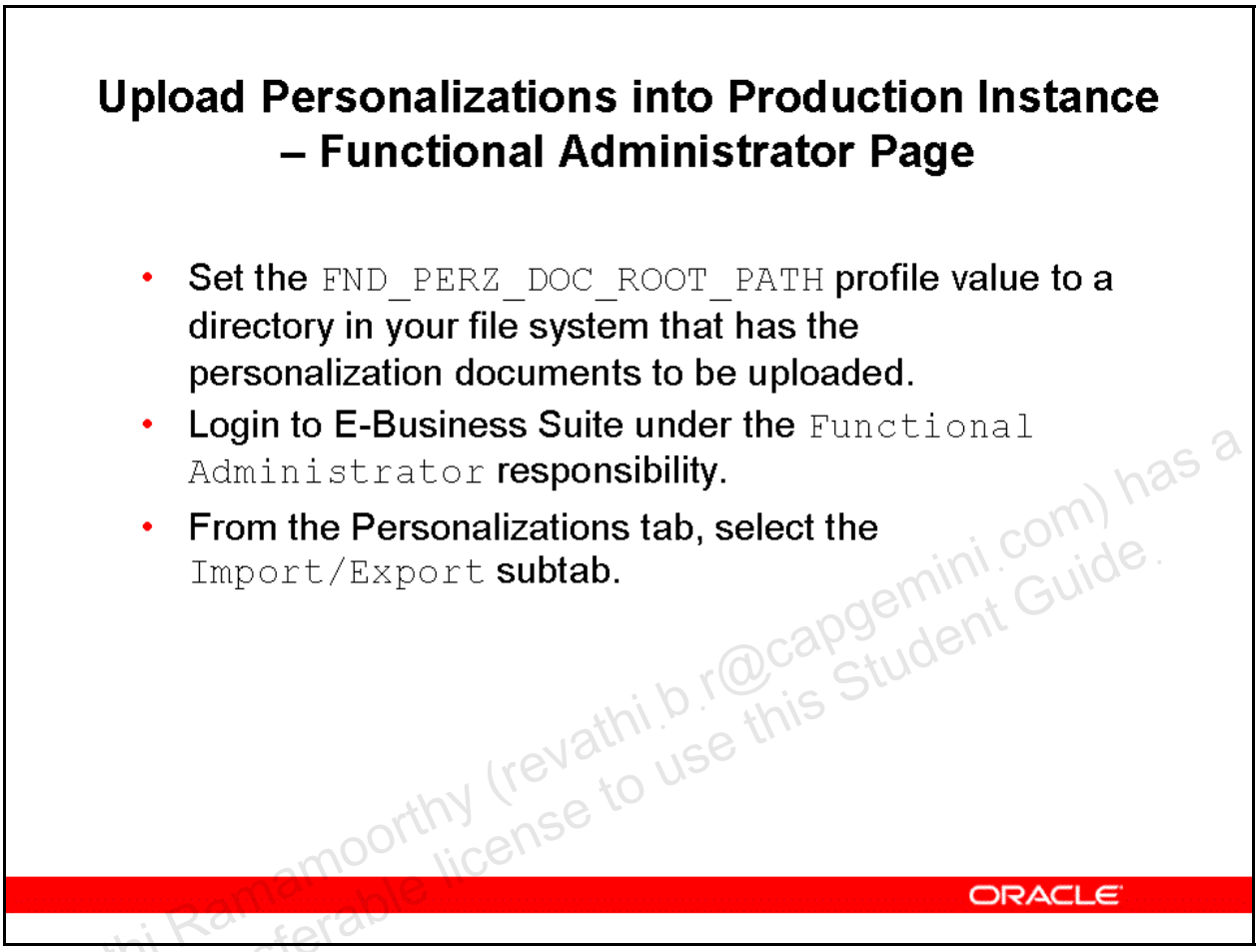
Extract the Personalizations – Export to File System

- Click on `Export to File System` button.



ORACLE

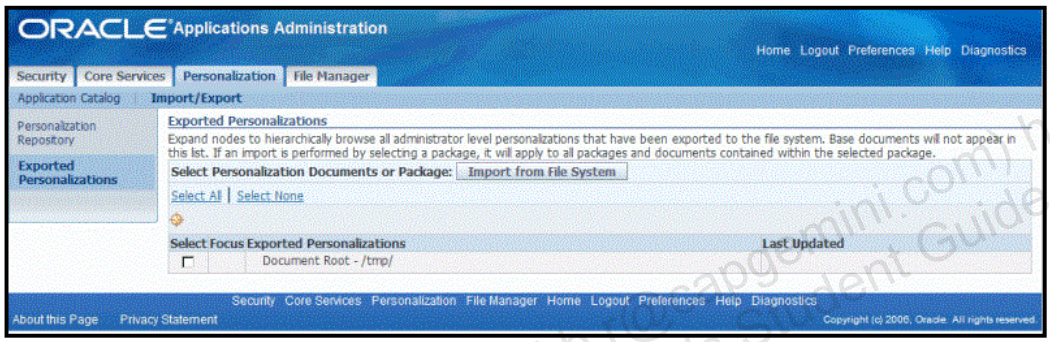
Upload Personalizations into Production Instance – Functional Administrator Page



The profile option should be set at the User level to ensure security.

Upload Personalizations into Production Instance – Exported Personalizations

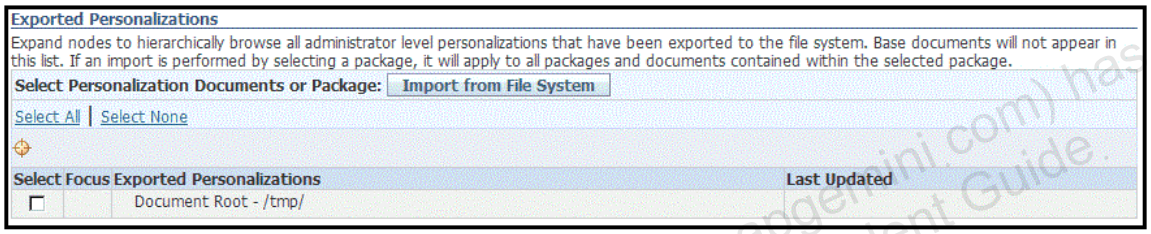
- Select `Exported Personalizations` from the side navigation menu.



Upload Personalizations into Production Instance – Import from File System

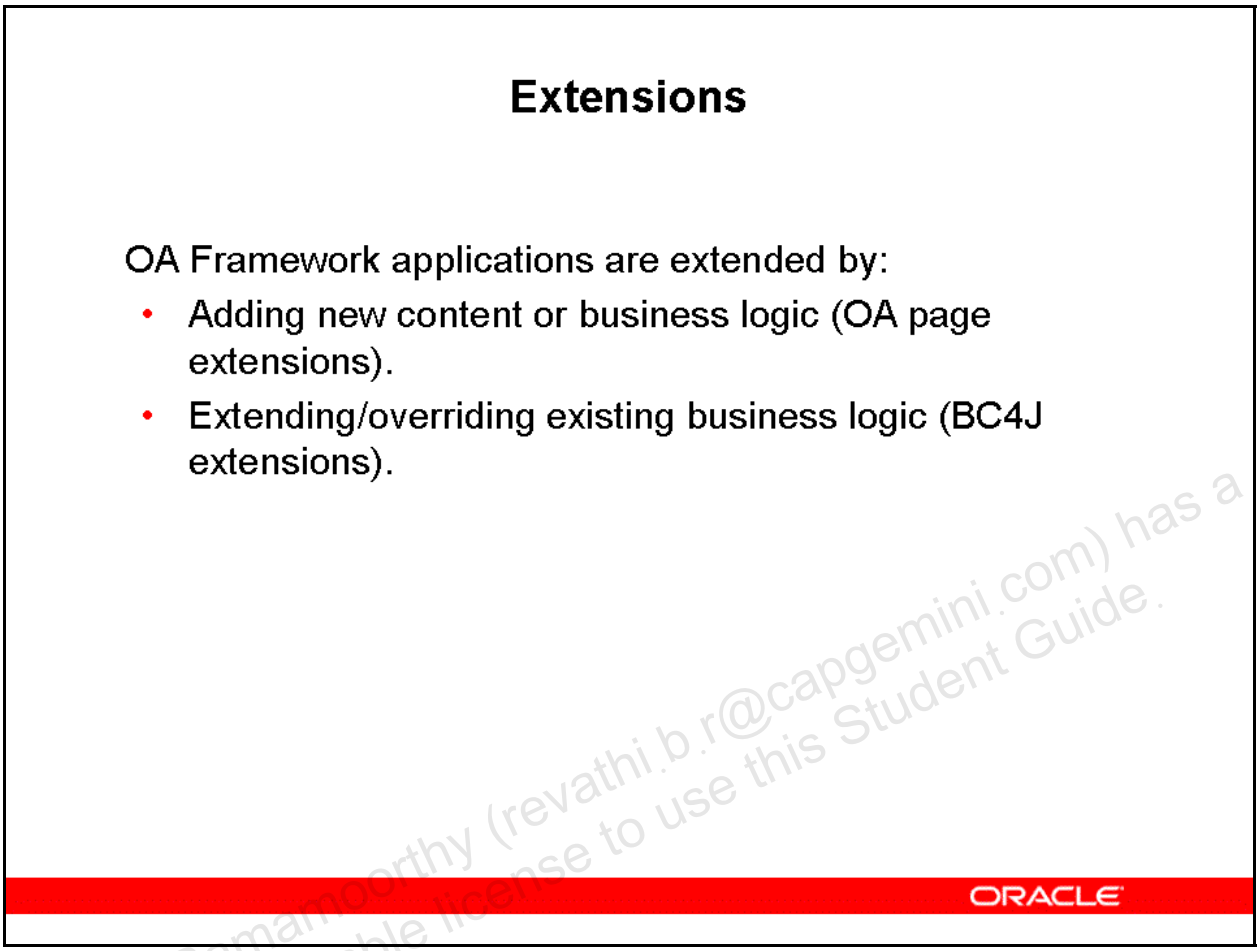
Upload Personalizations into Production Instance – Import from File System

- Expand the Exported Personalizations and select the page that needs to be imported from the file system.
- Click on `Import from File System` button.

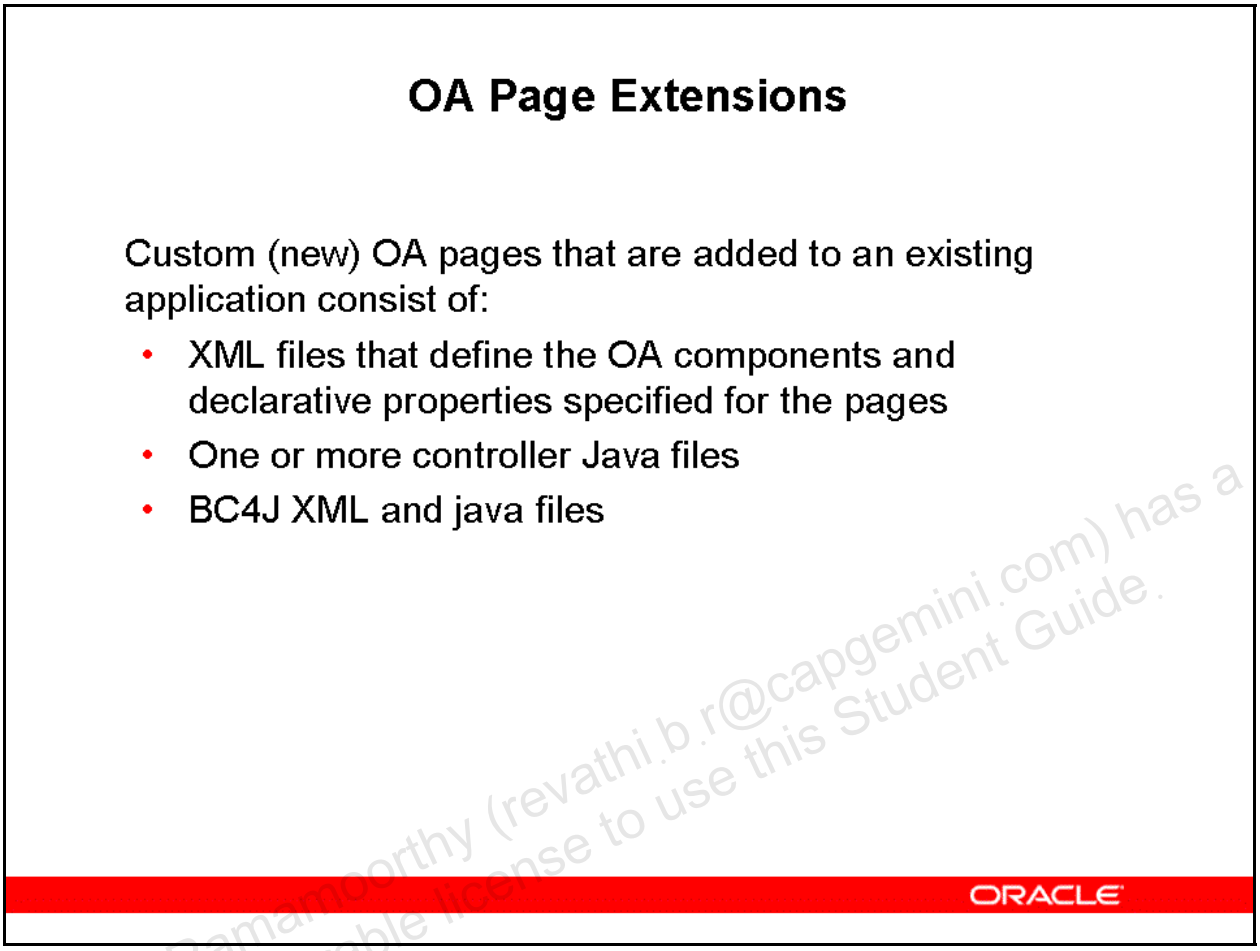


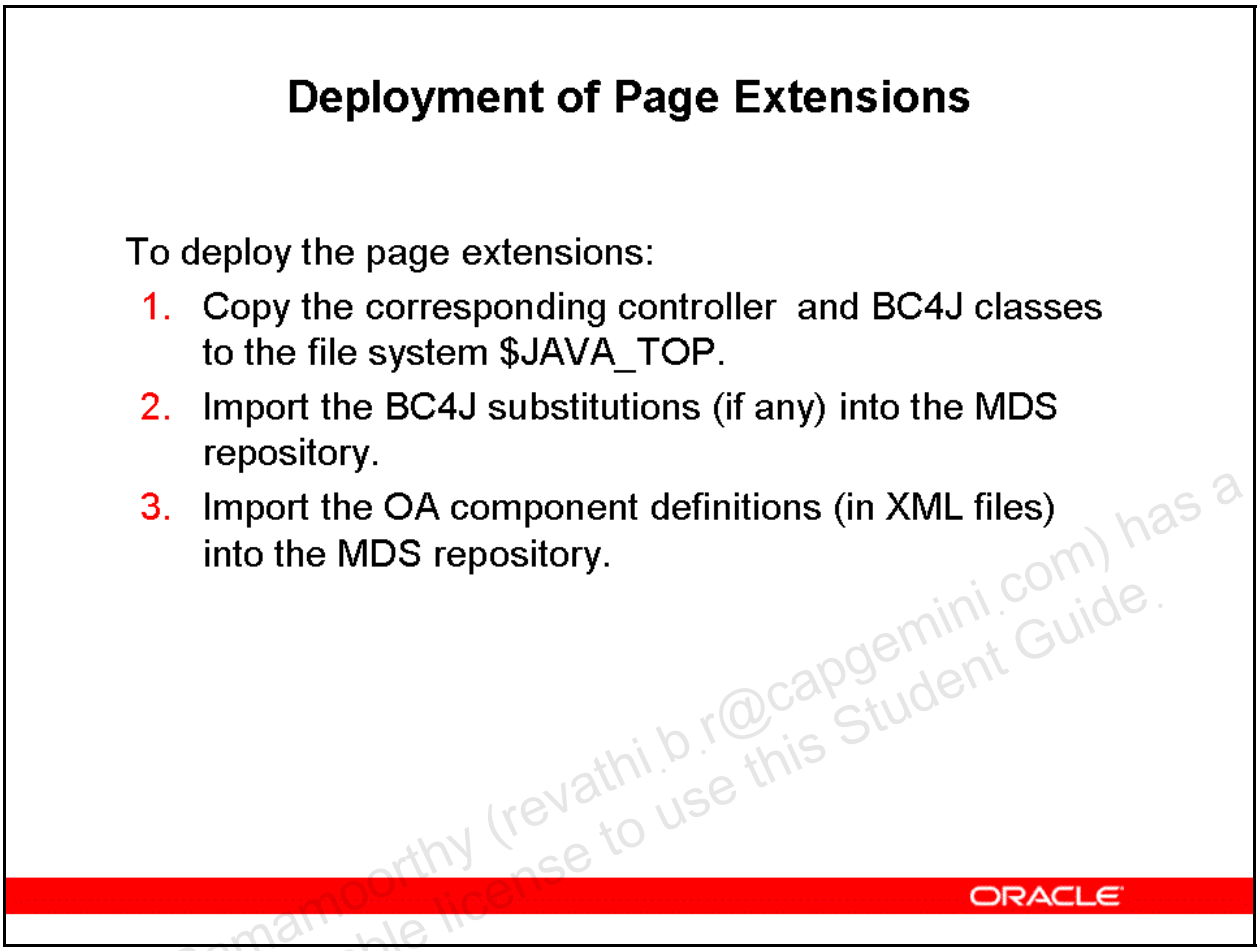
ORACLE

Extensions

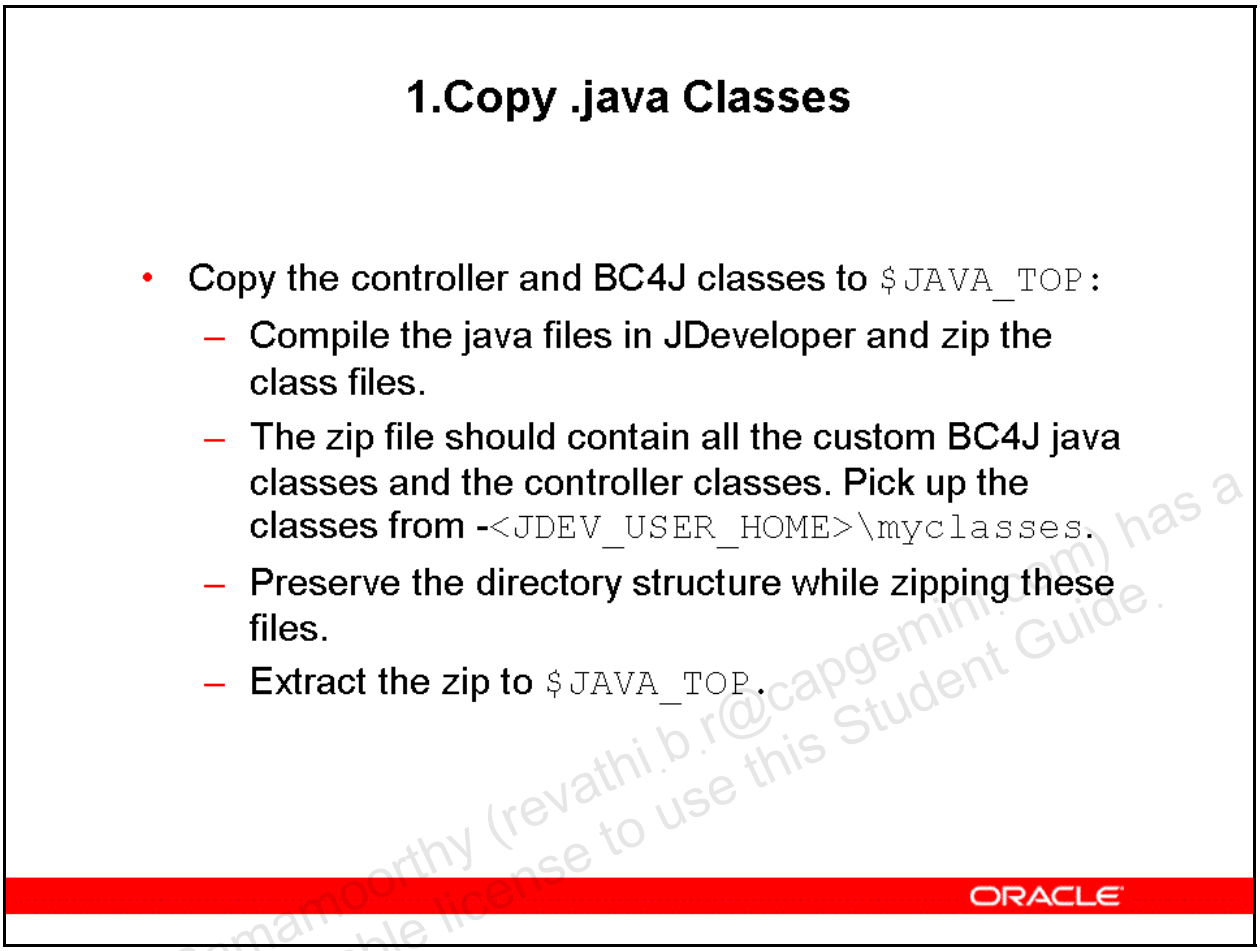


Extensions require either changes to Java code or changes to XML files that can not be done via personalizations. Since personalizations deal with the UI objects, the XML files needed by extensions are BC4J-based XML files.

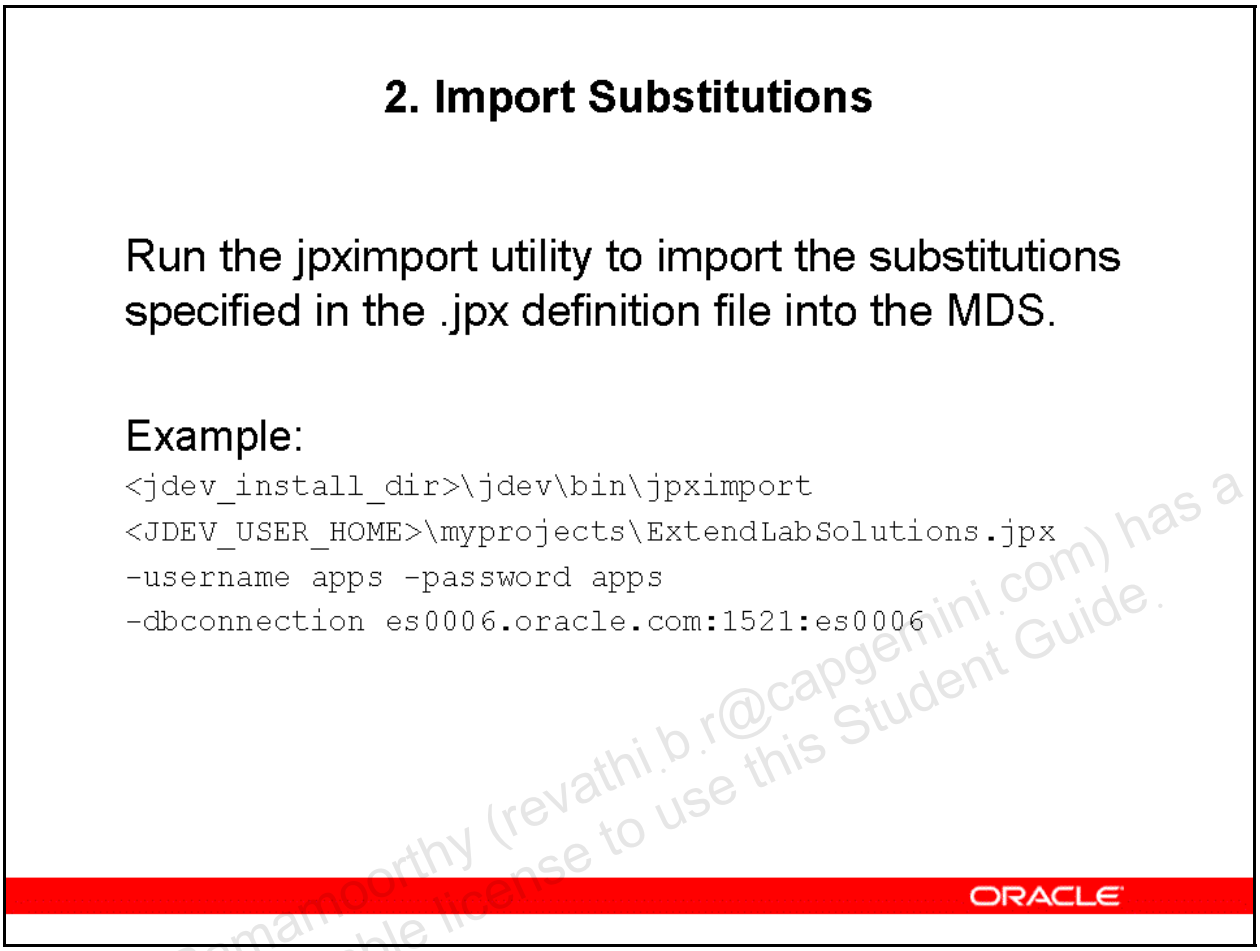


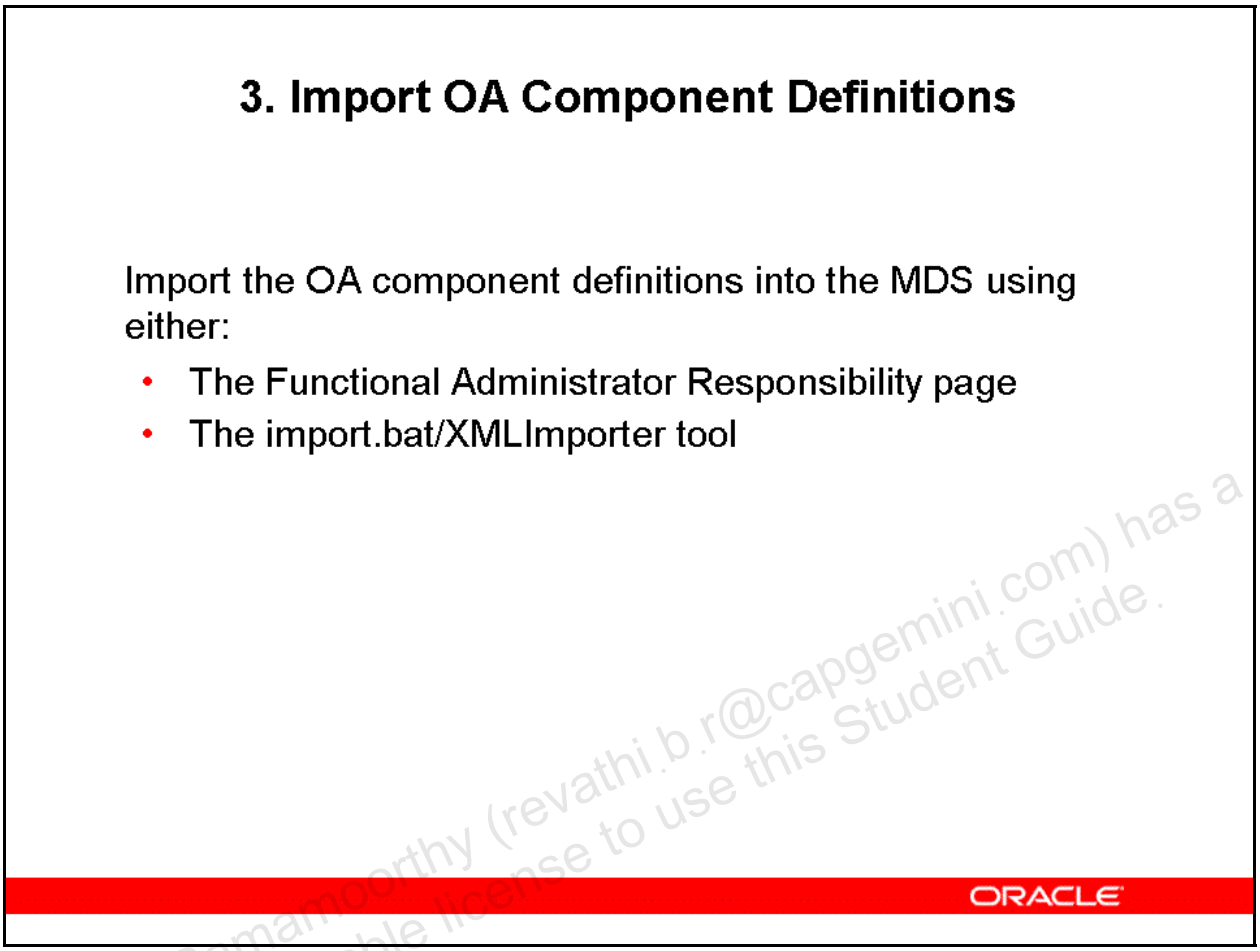


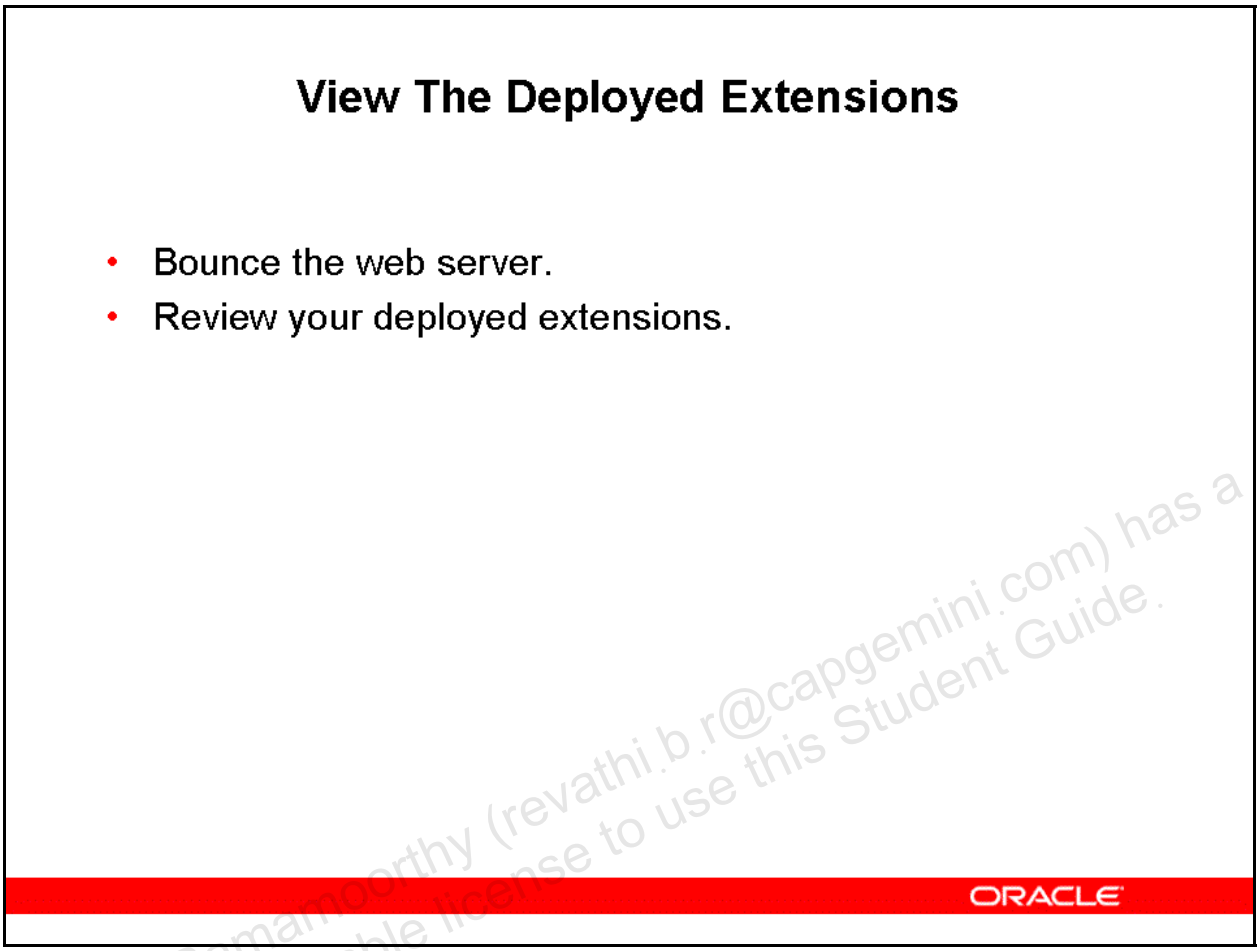
1.Copy .java Classes



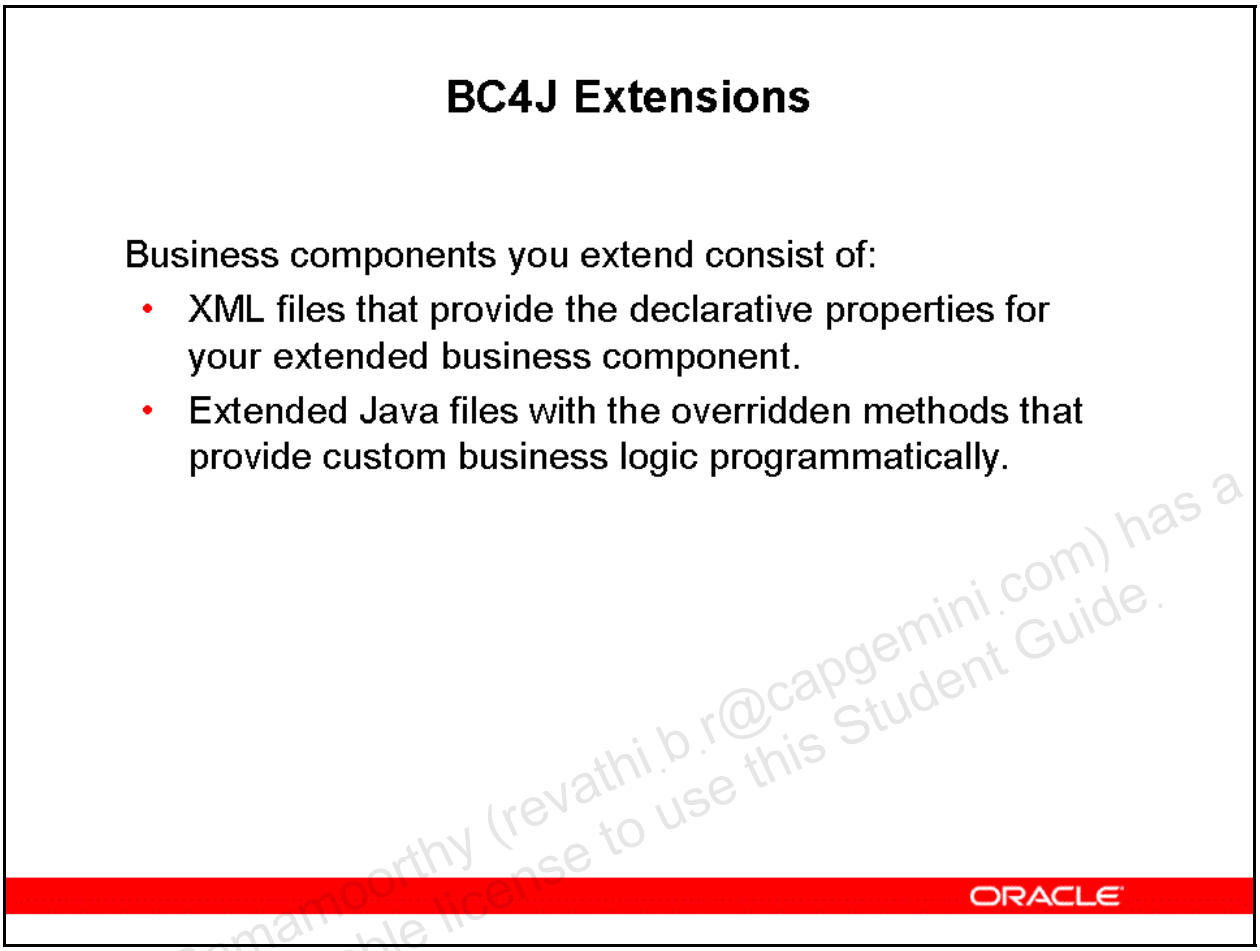
2. Import Substitutions







BC4J Extensions



Note: Extending BC4J objects is supported. Extending Controllers (CO) is not. Any work done to extend an Oracle-supplied Controller is by definition unsupported.

Deployment of Business Logic Extensions

To deploy business logic extensions:

1. Compile your Java in JDeveloper and zip up your Java classes.
2. Extract the .zip to `$JAVA_TOP`.
3. Run the jpx import utility to import substitutions specified in the .jpx definition file to the MDS repository.
4. Bounce the web server (if required).
5. Review your deployed extensions.

ORACLE

Summary

In this lessons, you should have learned to:

- Describe the personalization directory structure.
- List the tools used for deploying Personalizations.
- Inspect the MDS repository for personalization documents.
- Deploy Personalizations.
- Deploy a custom page.
- Deploy business logic extensions.

ORACLE