R12.x Extend Oracle Applications: Building OA Framework Applications

Student Guide – Volume I

D61636GC10
Edition 1.0
November 2010
D69573

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

Preface

Profile

Before You Begin This Course

- Working experience with *Java, JDeveloper and J2EE technologies*

Prerequisites

- There are no prerequisites for this course.

How This Course Is Organized

This is an instructor-led course featuring lecture and hands-on exercises. Online demonstrations and written practice sessions reinforce the concepts and skills introduced.

Related Publications

Oracle Publications

Title	Part Number

Additional Publications

- System release bulletins
- Installation and user's guides
- Read-me files
- International Oracle User's Group (IOUG) articles
- *Oracle Magazine*
- My Oracle Support related documentation

Typographic Conventions

Typographic Conventions in Text

Convention	Element	Example
Bold italic	Glossary term (if there is a glossary)	The ***algorithm*** inserts the new key.
Caps and lowercase	Buttons, check boxes, triggers, windows	Click the Executable button. Select the Can't Delete Card check box. Assign a When-Validate-Item trigger to the ORD block. Open the Master Schedule window.
Courier new, case sensitive (default is lowercase)	Code output, directory names, filenames, passwords, pathnames, URLs, user input, usernames	Code output: `debug.set ('I", 300);` Directory: `bin` (DOS), `$FMHOME` (UNIX) Filename: Locate the `init.ora` file. Password: User `tiger` as your password. Pathname: Open `c:\my_docs\projects` URL: Go to `http://www.oracle.com` User input: Enter `300` Username: Log on as `scott`
Initial cap	Graphics labels (unless the term is a proper noun)	Customer address (*but* Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do *not* save changes to the database. For further information, see *Oracle7 Server SQL Language Reference Manual.* Enter `user_id@us.oracle.com`, where *user_id* is the name of the user.
Quotation marks	Interface elements with long names that have only initial caps; lesson and chapter titles in cross-references	Select "Include a reusable module component" and click Finish. This subject is covered in Unit II, Lesson 3, "Working with Objects."
Uppercase	SQL column names, commands, functions, schemas, table names	Use the SELECT command to view information stored in the LAST_NAME column of the EMP table.
Arrow	Menu paths	Select File > Save.
Brackets	Key names	Press [Enter].
Commas	Key sequences	Press and release keys one at a time: [Alternate], [F], [D]
Plus signs	Key combinations	Press and hold these keys simultaneously: [Ctrl]+[Alt]+[Del]

Typographic Conventions in Code

Convention	Element	Example

Caps and lowercase	Oracle Forms triggers	`When-Validate-Item`
Lowercase	Column names, table names	`SELECT last_name` `FROM s_emp;`
	Passwords	`DROP USER scott` `IDENTIFIED BY tiger;`
	PL/SQL objects	`OG_ACTIVATE_LAYER` `(OG_GET_LAYER ('prod_pie_layer'))`
Lowercase italic	Syntax variables	`CREATE ROLE` *`role`*
Uppercase	SQL commands and functions	`SELECT userid` `FROM emp;`

Typographic Conventions in Oracle Application Navigation Paths

This course uses simplified navigation paths, such as the following example, to direct you through Oracle Applications.

(N) Invoice > Entry > Invoice Batches Summary (M) Query > Find (B) Approve

This simplified path translates to the following:

1. (N) From the Navigator window, select **Invoice** then **Entry** then **Invoice Batches Summary**.
2. (M) From the menu, select **Query** then **Find**.
3. (B) Click the **Approve** button.

Notations:

(N) = Navigator

(M) = Menu

(T) = Tab

(B) = Button

(I) = Icon

(H) = Hyperlink

(ST) = Sub Tab

Typographical Conventions in Oracle Application Help System Paths

This course uses a "navigation path" convention to represent actions you perform to find pertinent information in the Oracle Applications Help System.

The following help navigation path, for example—

(Help) General Ledger > Journals > Enter Journals

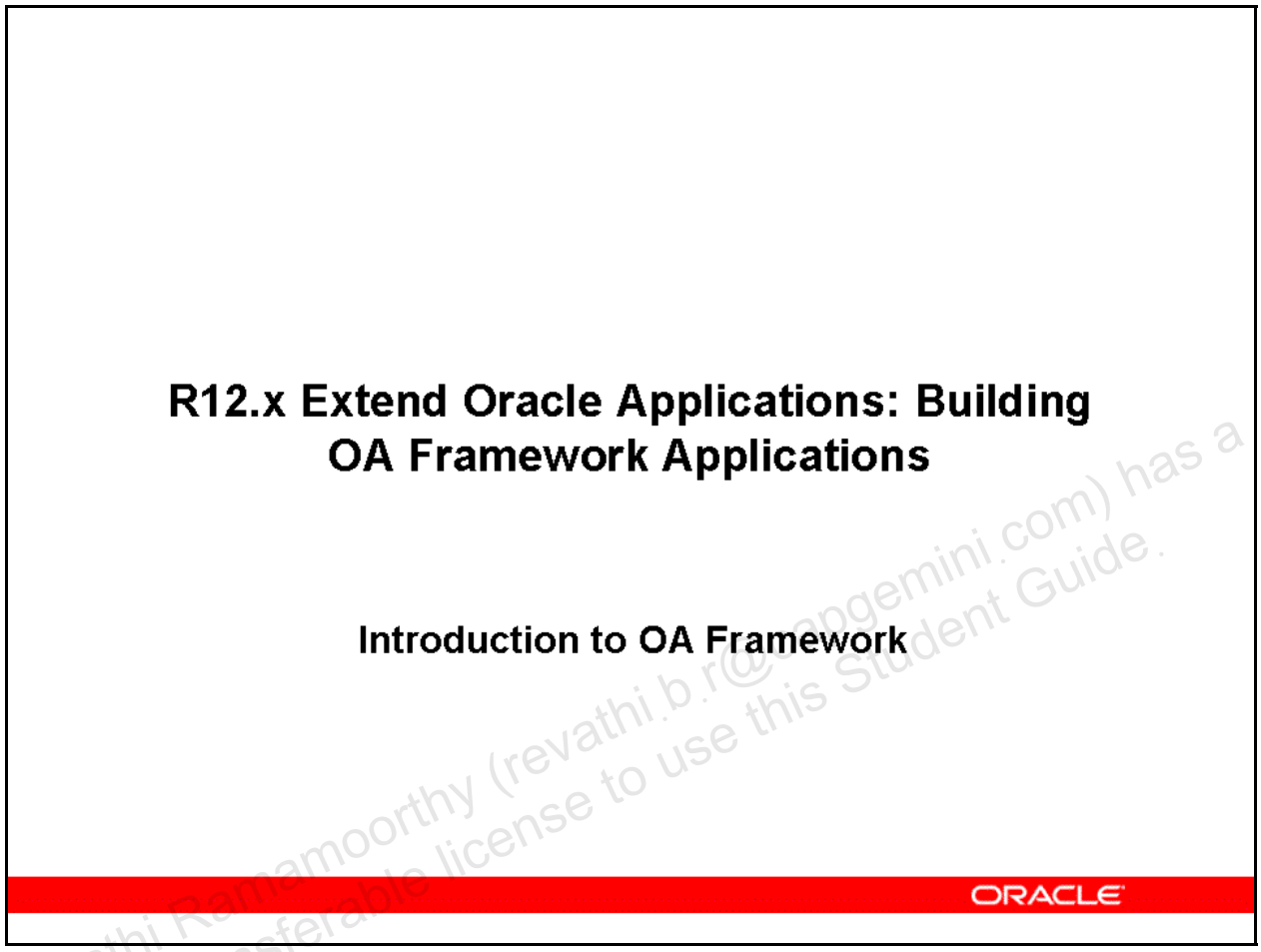
—represents the following sequence of actions:

1. In the navigation frame of the help system window, expand the General Ledger entry.
2. Under the General Ledger entry, expand Journals.

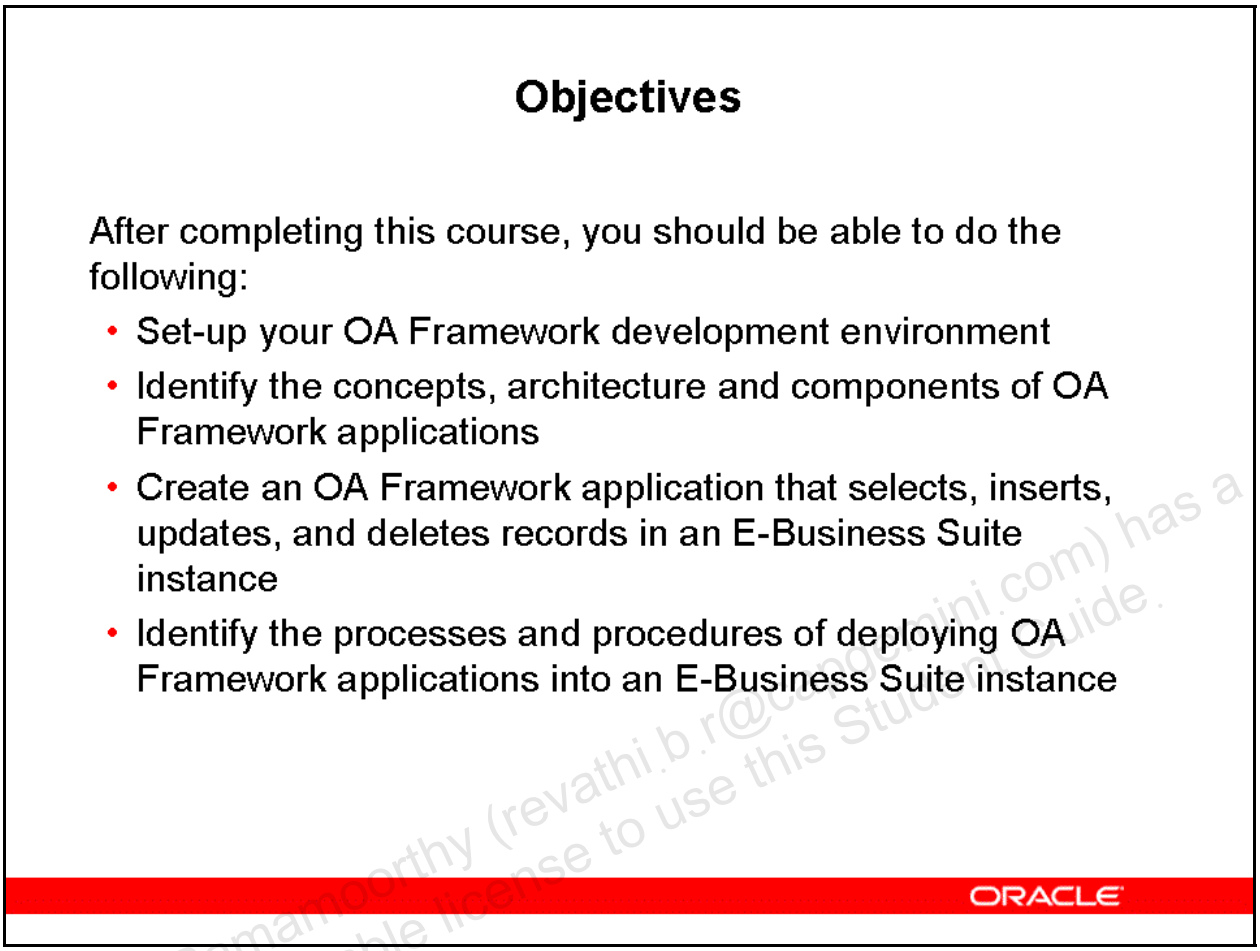
3. Under Journals, select Enter Journals.
4. Review the Enter Journals topic that appears in the document frame of the help system window.

Introduction to OA Framework

Chapter 1



Objectives



Note: It is equally important to note a few of the things you will not be doing or specifically learning in this course.

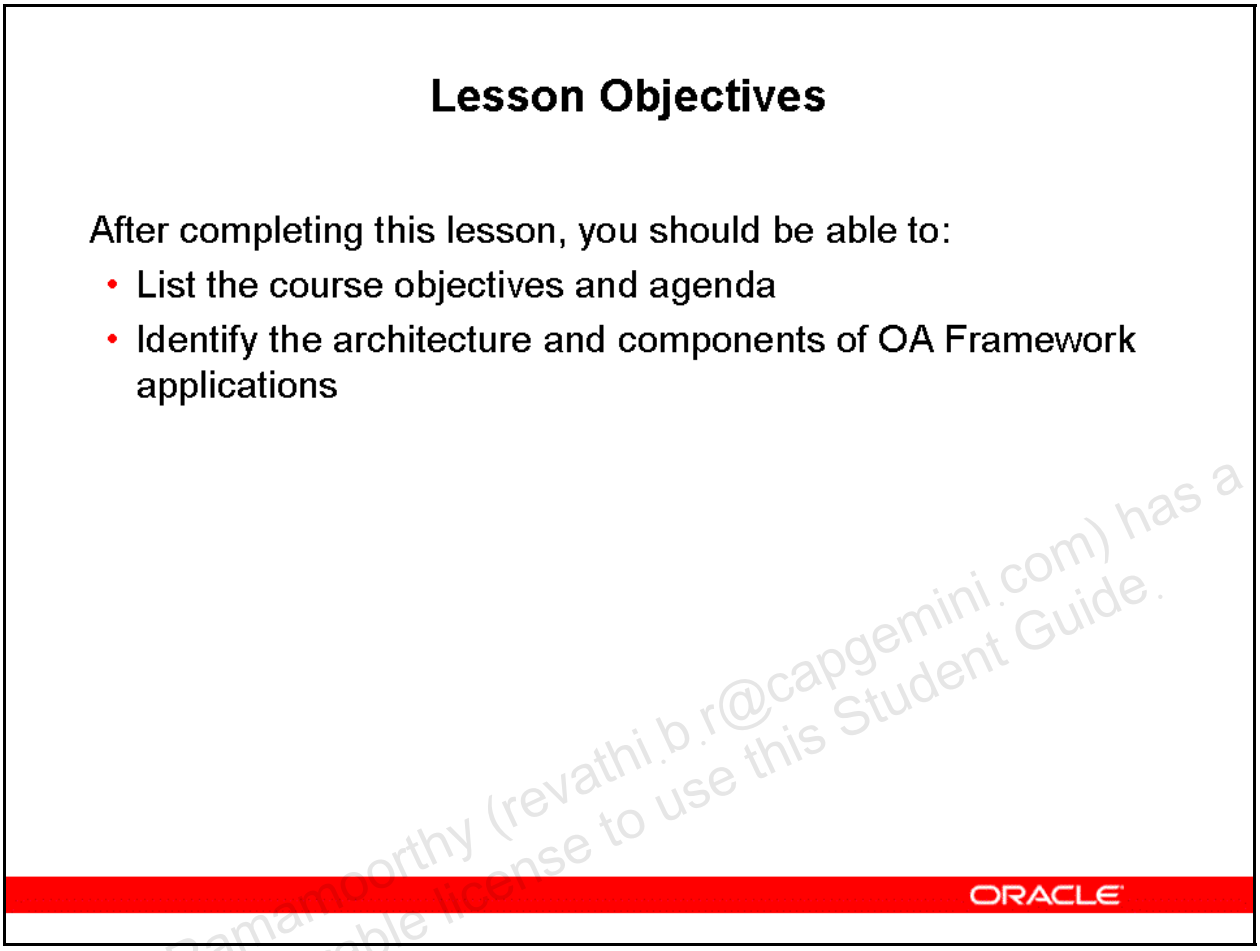
You will not be learning or doing OA Framework personalizations.

You will not be extending or modifying existing OA Framework pages.

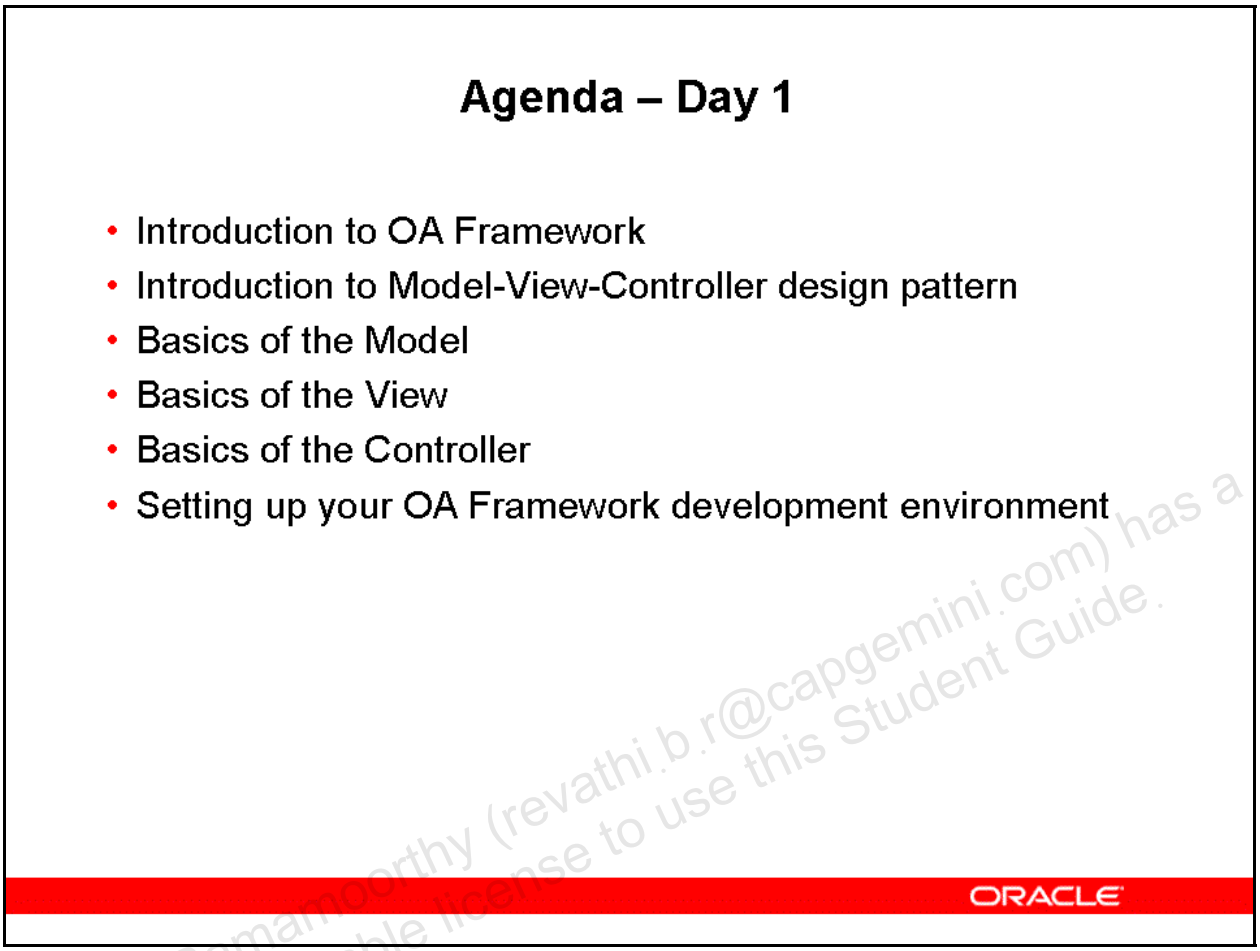
You will not perform an actual deployment. The process will be discussed.

Deployment details are outside the scope of this course.

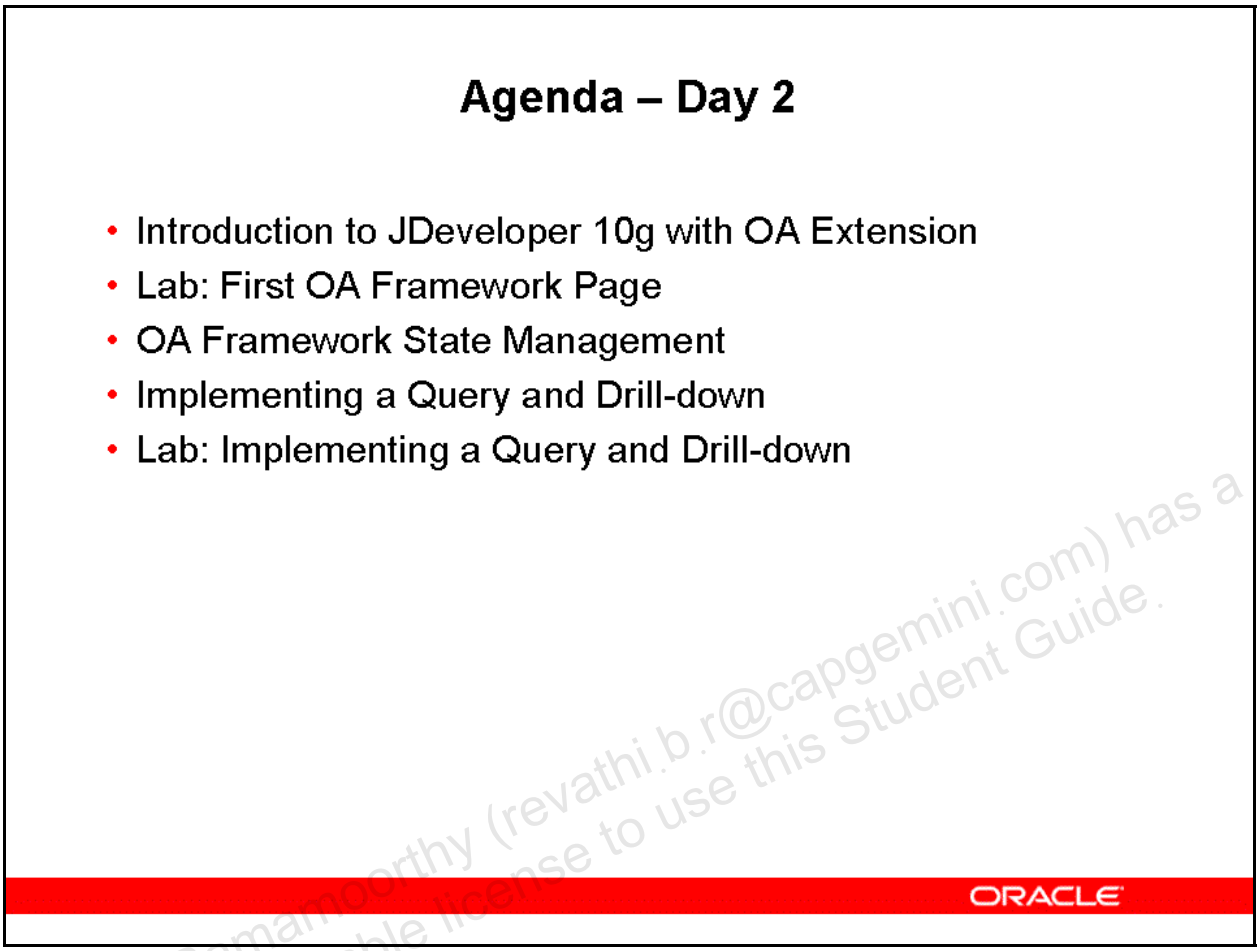
For more details on this see the R12 Extending OA Framework Pages course.

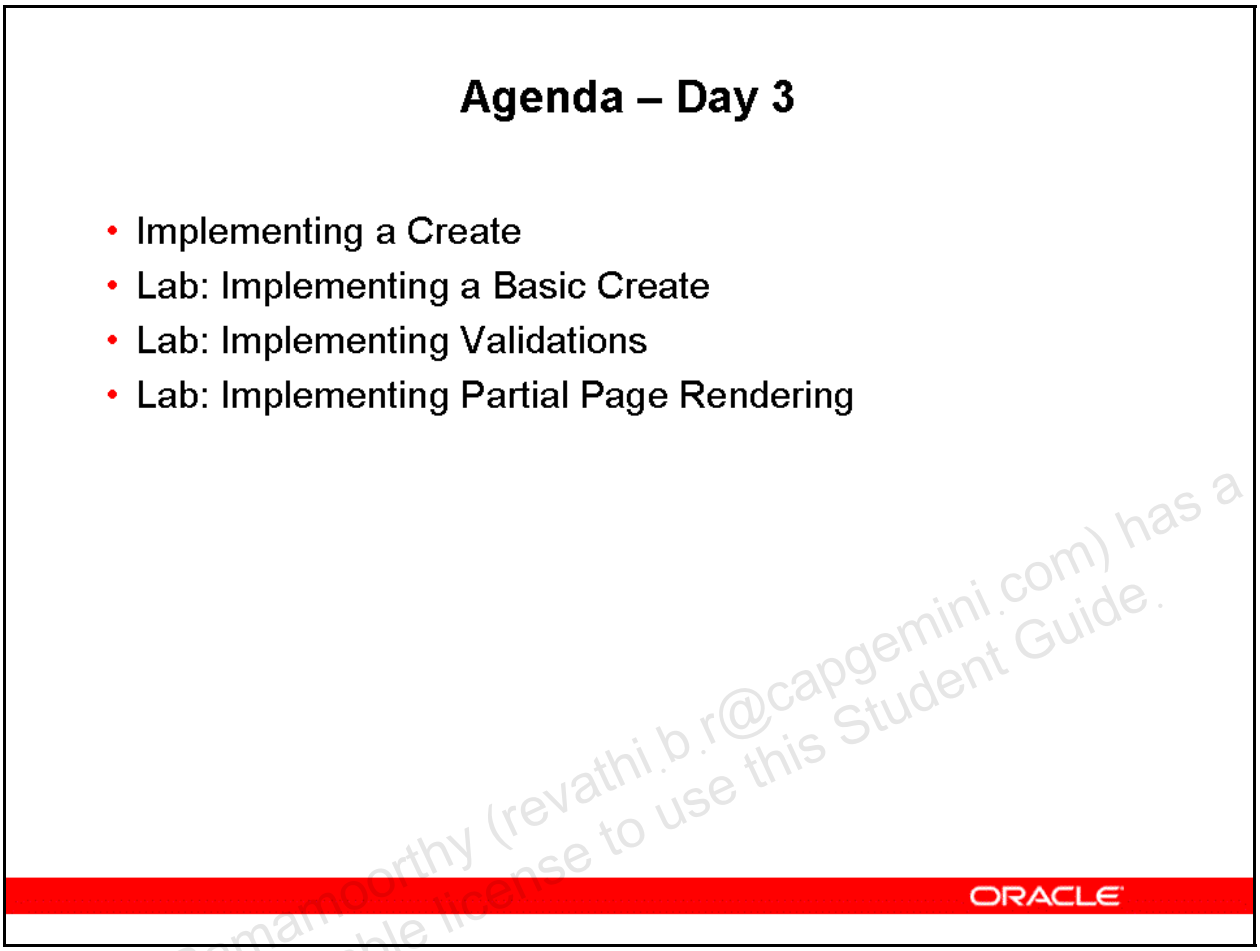


Agenda – Day 1

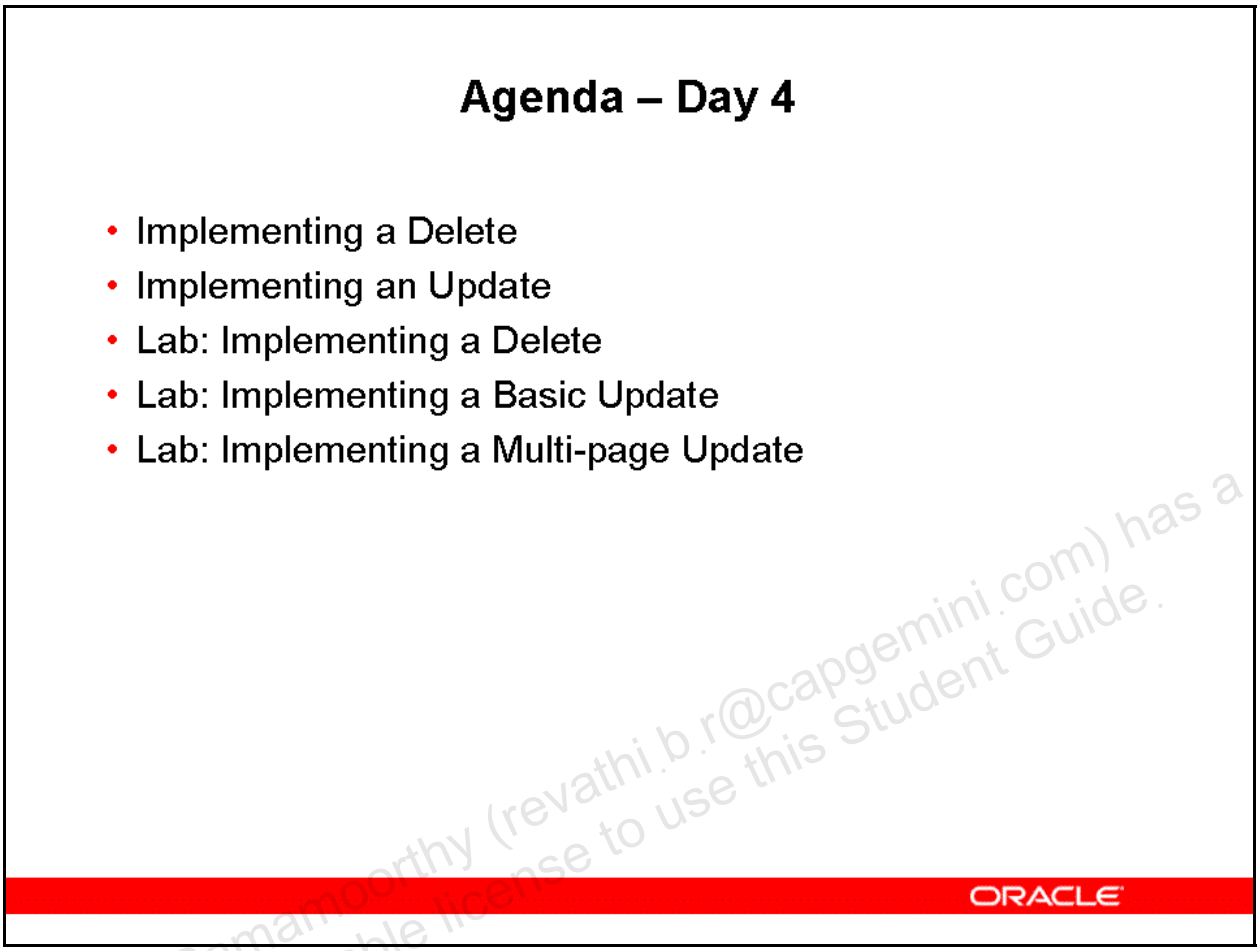


Agenda – Day 2

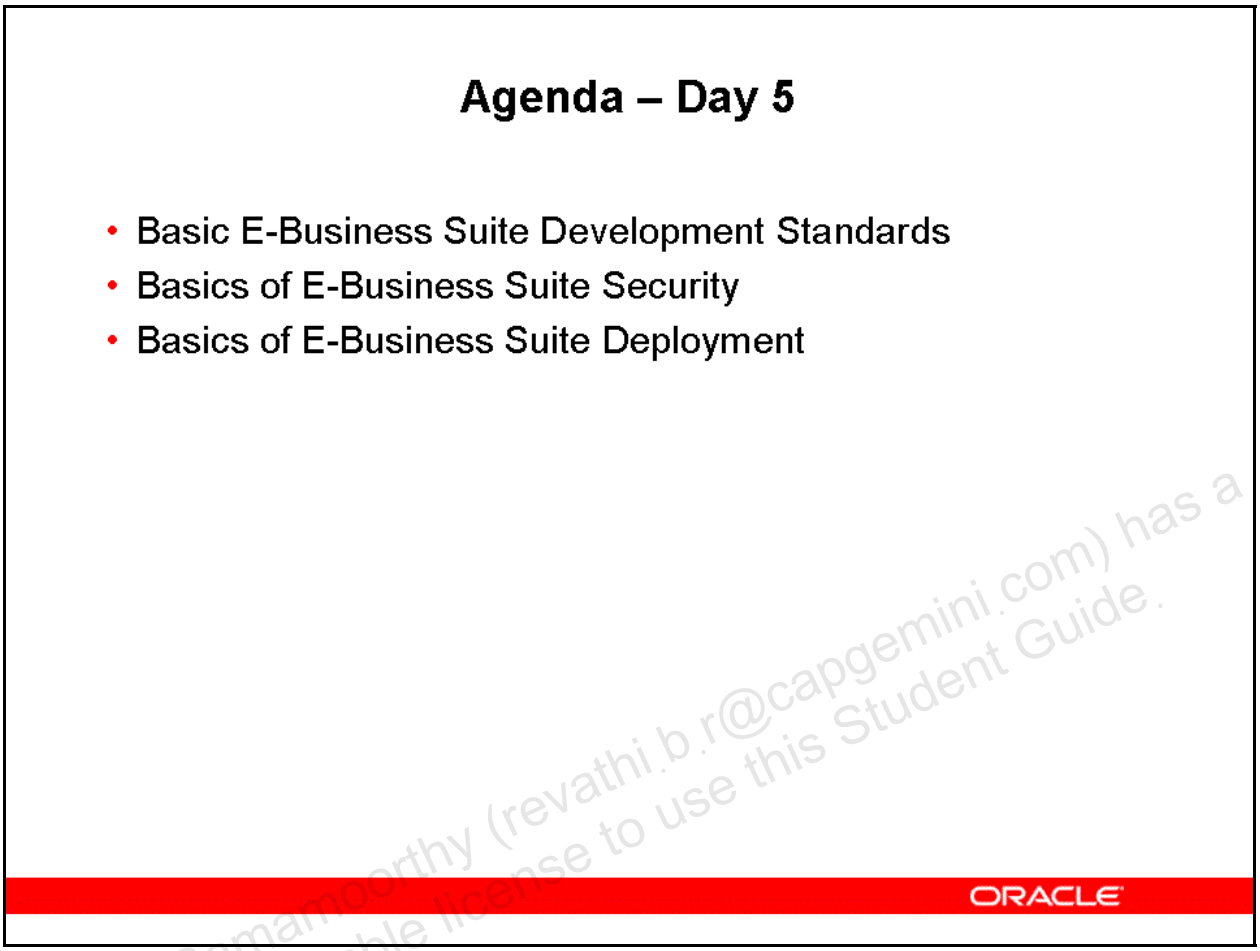




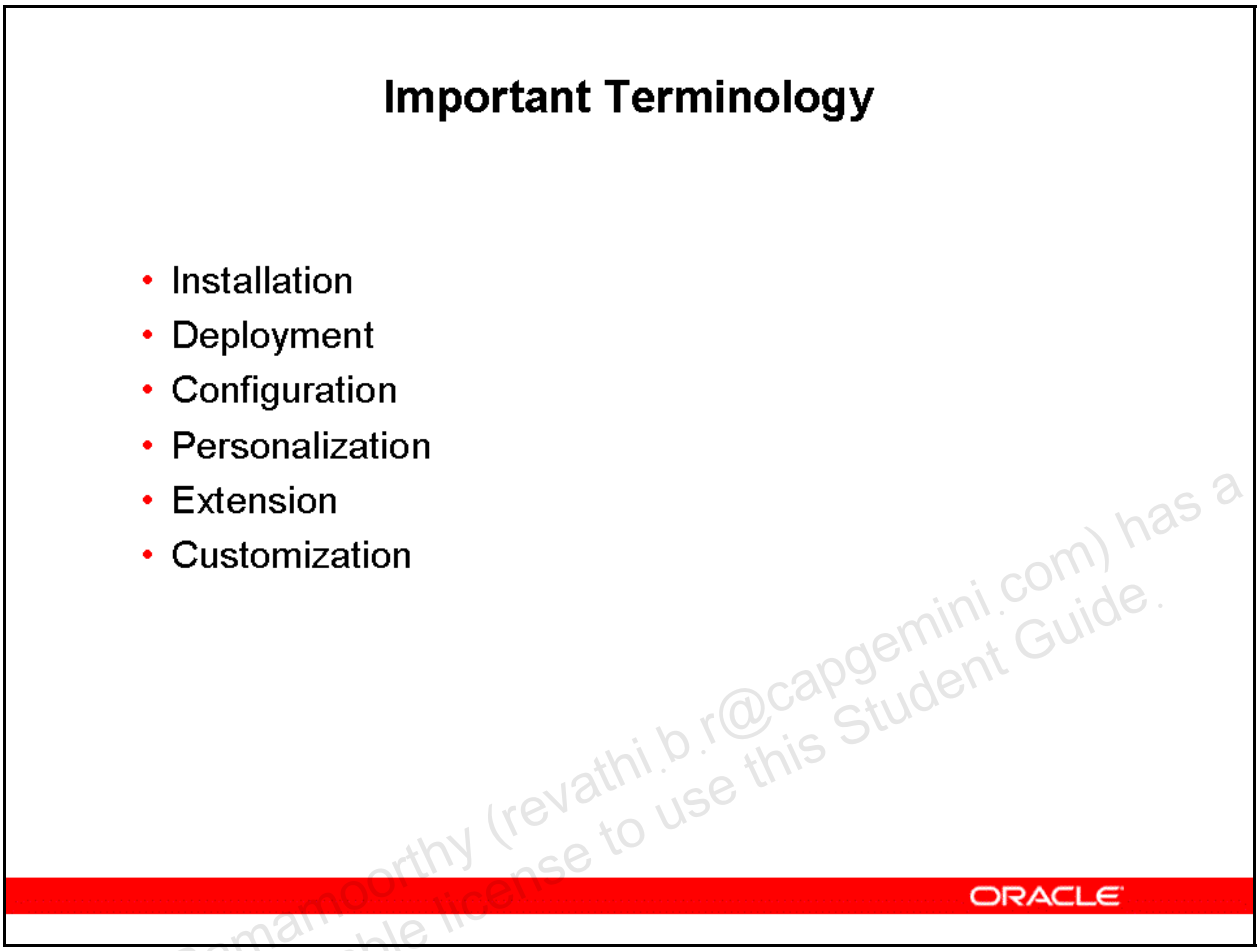
Agenda – Day 4



Agenda – Day 5



Important Terminology



In any development, customization, extension, configuration, or installation process, there are words that can be used interchangeably. The usage of the words are defined by their context. The context of the usage of a word is fine during a conversation, but it doesn't always make for a clear understanding while learning. For this course, there are six critically important terms that you must understand and use properly: installation, deployment, configuration, personalization, extension and customization.

Installation is the process of moving the E-Business Suite from distribution media (i.e., CD or DVD) to the host system. The complexity of the process and the tasks to complete the move are based on the operating system (i.e., Windows, Linux, etc.) of the host system. Installation is only the first step of bringing up a fully functional E-Business Suite instance. The software still needs to be configured to the specific needs of the business or business unit for which it is installed. The software may also need to be installed on multiple servers if a multi-node installation is desired.

Deployment is the process of moving personalizations, extensions, or customizations from the development systems or servers to testing/production servers.

Configuration is the process of setting the proper parameters and customer-specific information for a fully-functional E-Business Suite instance. The requirements of configuration are set by each of the product teams. This is the most time consuming part of the process of bringing an E-Business Suite instance to full production. This

process is not trivial, and will involve a multiple of subject matter experts covering the gamut of your business.

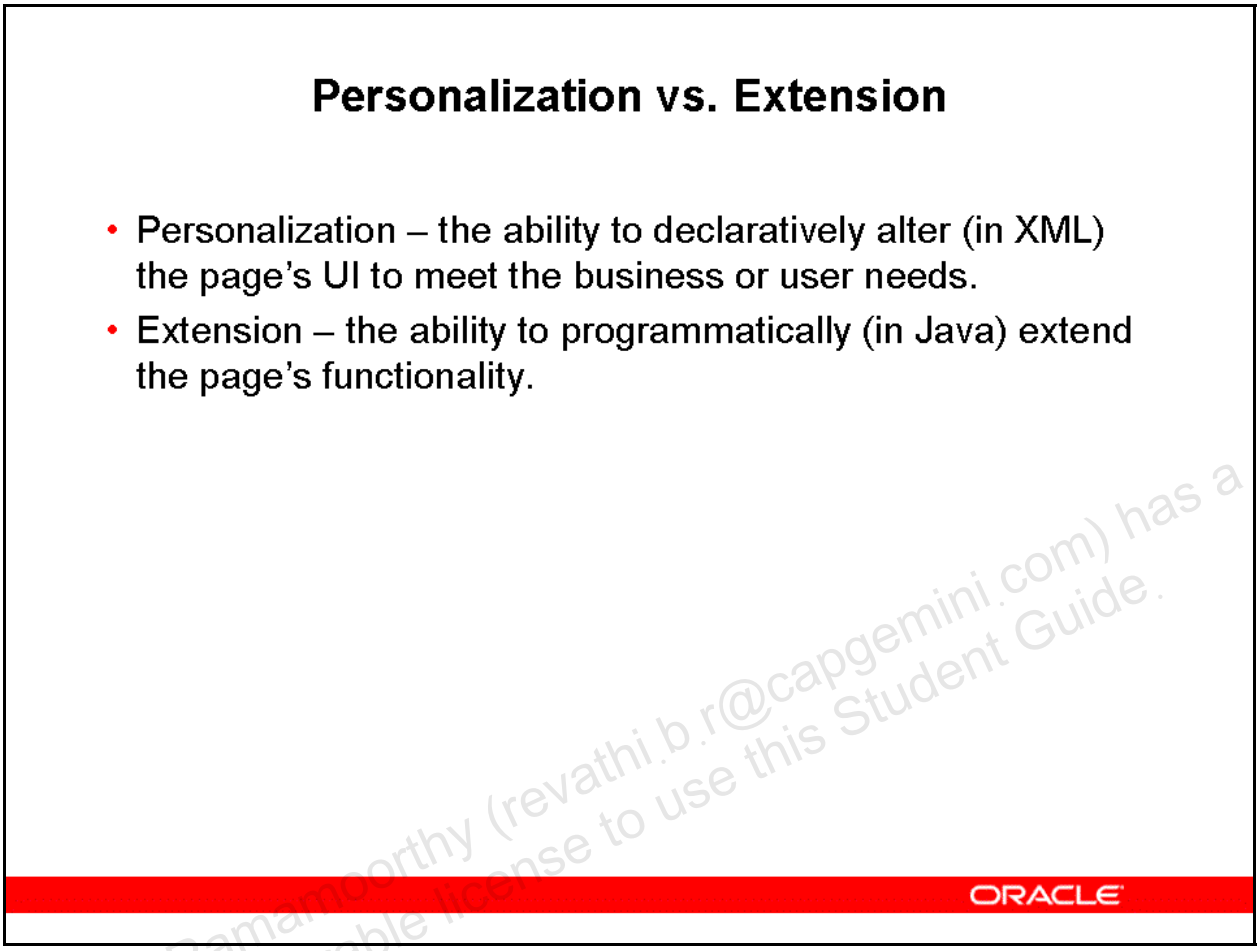
Personalizations, Extensions and Customizations are all supported. But, it is critical to understand the scope and limitations of that support. Oracle supports the mechanics of each of these techniques. It also provides the tools for accomplishing such actions along with documentation on the tool. The implementation specifics for any given personalization, extension, or customization as required by a specific customer installation, are not supported beyond the mechanics, tools, and documentation support.

Personalization is the process of making changes to the User Interface (UI) from within an E-Business Suite form/page. It is possible to make personalizations to both Forms-based and OA Framework-based pages.

Extension is the process of making changes to the programmatic (i.e., PL/SQL or Java) elements of an E-Business Suite form/page. It is possible to extend both Forms-based and OA Framework-based pages.

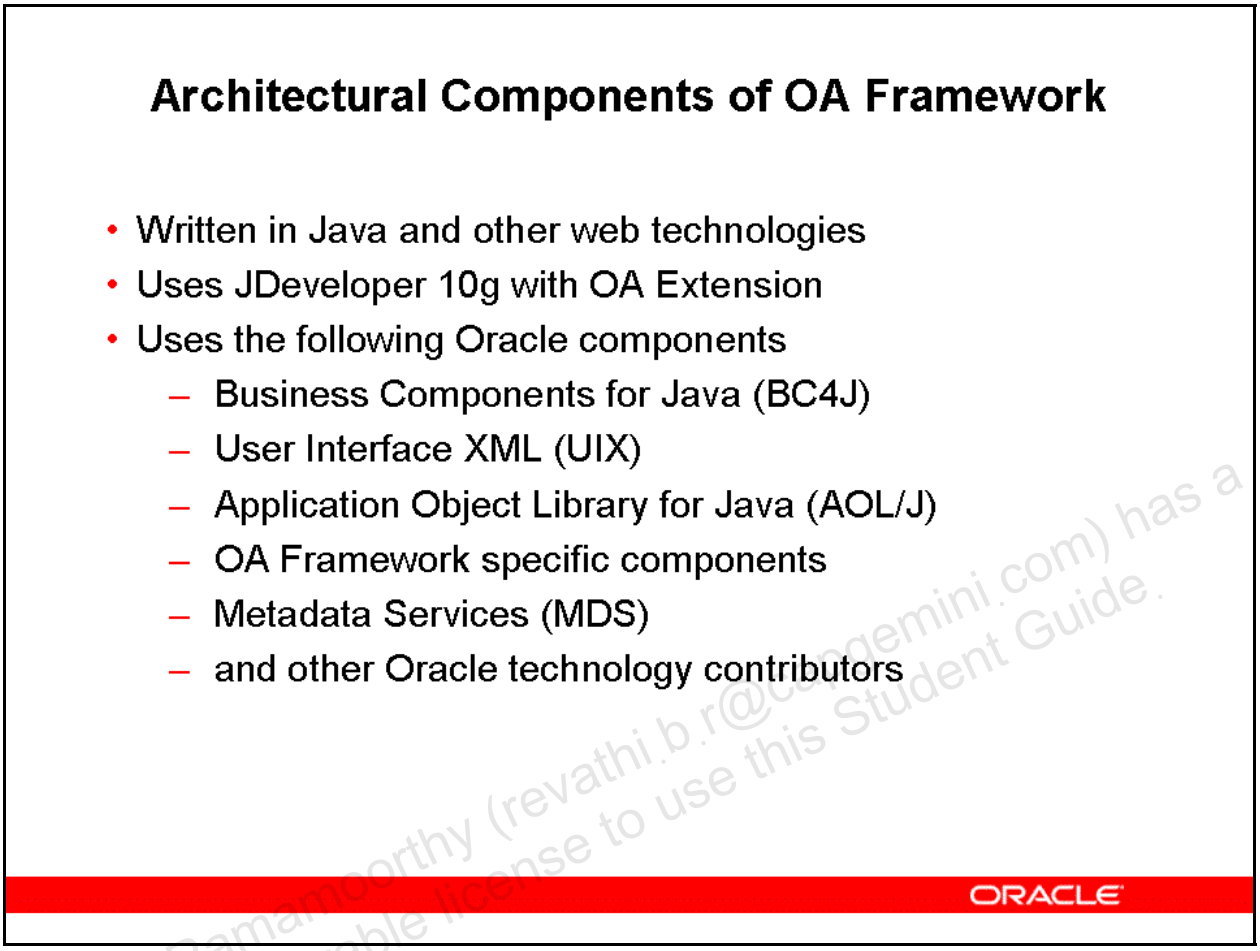
Customization is the process of creating new forms/pages. While Oracle does provide tools to do this (i.e., Oracle Forms and JDeveloper 10g with OA Extension), this is the least supported option.

Personalization vs. Extension



From this page forward, we will discuss Personalizations and Extensions within the context of OA Framework pages and applications. While it is possible to personalize and extend Forms-based pages and applications, it is beyond the scope of this course.

Architectural Components of OA Framework



The smaller contributors are:

Caching Framework

Oracle JDBC 10g

To establish a common understanding, here are some basic definitions of the components:

Business Components for Java (**BC4J**) is a component that meshes SQL database concepts like views, tables, and transactions into the Java world via View Objects (VOs), Entity Objects (EOs), and Application Modules (AMs).

User Interface XML (**UIX**) is the component that allows creation of user interface (UI) objects through use of XML. This removes the need for the programmer to know various implementation details, like HTML/DHTML or WML. At run-time, the generic XML can be transformed into the necessary output. UIX is also component-based, with each UI element defined as a JavaBean.

Application Object Library for Java (**AOL/J**) provides a number of E-Business Suite specific services, especially security. OA Framework pages implement that same user security model used in E-Business Suite.

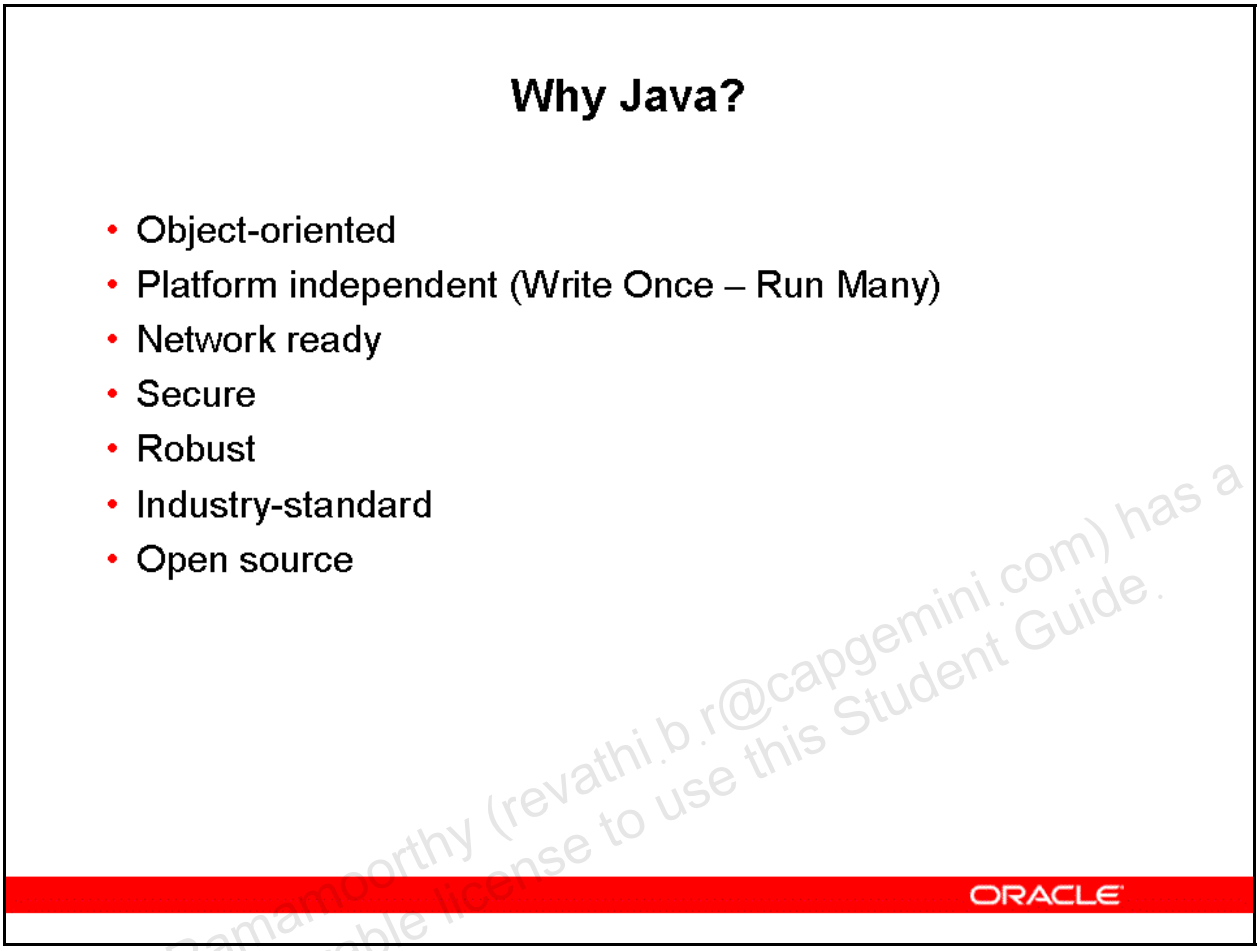
Metadata Services (**MDS**) is the delivery of XML on demand. OA Framework page, personalizations, and BC4J substitutions are not stored in the file system of the server. These XML components are stored and managed in MDS (tables in the database).

HTML is hypertext markup language, and is a standard maintained by the World Wide Web Consortium (W3C) (http://www.w3.org). It is the language supported by browsers.

WML is wireless markup language, and is supported by most cell phones and other wireless devices.

XML is extensible markup language, and is the generic standard underneath technologies like HTML and WML.

Why Java?



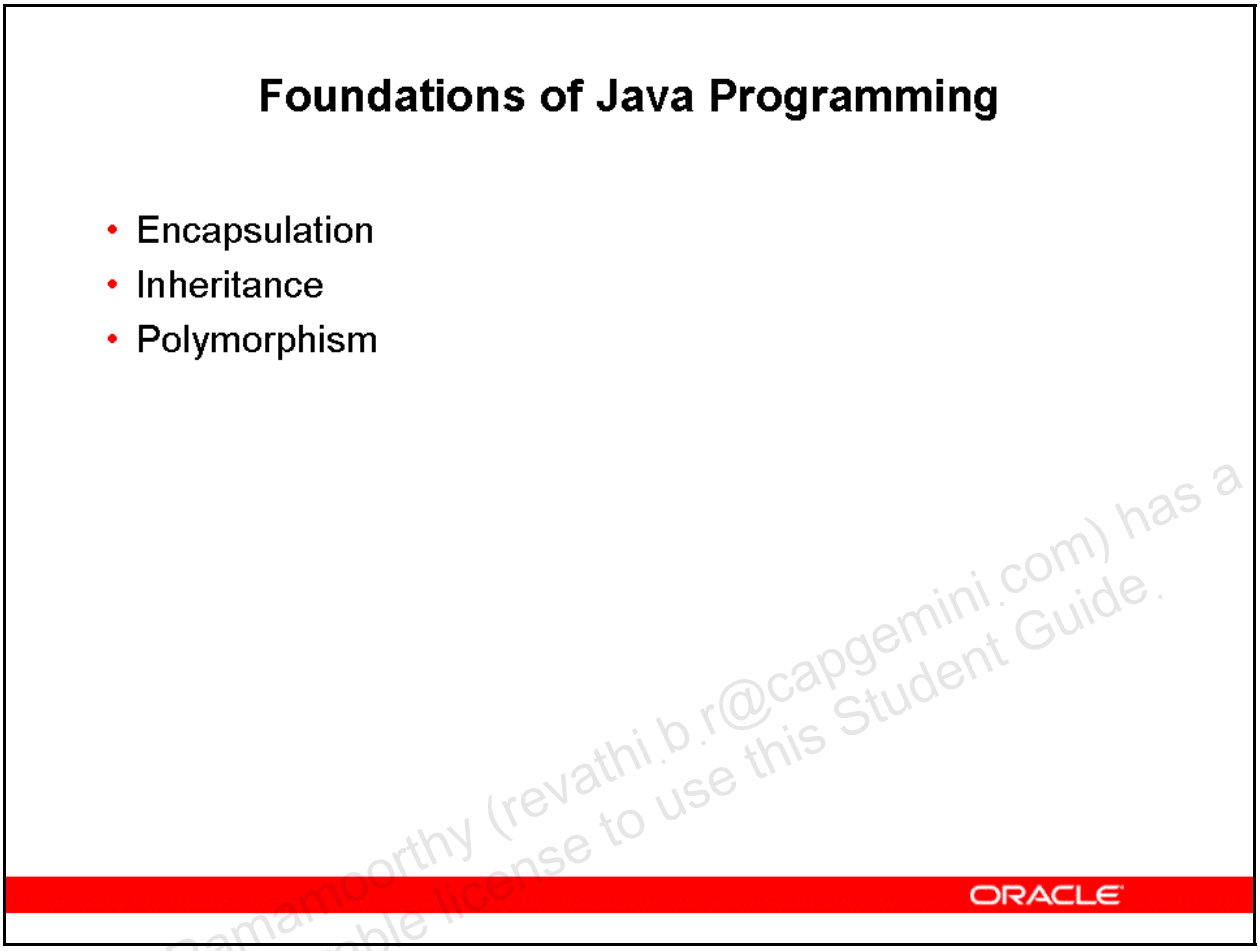
Java is a programming language that shares numerous features with other programming languages, especially C and C++. Java usage eases the learning curve for developers. Java is also an industry standard. Java is not a proprietary development language like PL/SQL or Oracle Forms. Most of Java is now open source, and Oracle has contributed many components, as open source, to the Java community.

Object-oriented programming (OOP) allows us to create the highly-sophisticated, distributed, networked applications that are in demand today. OOP allows us to do so by implementing 3 common pillars, encapsulation, inheritance, and polymorphism.

Java is robust in that it includes numerous compile-time and run-time checks to ensure that the programmer has followed good programming habits. The most common practical application of this is in Java's memory management. Java provides all the benefits of C and C++ programmer-defined data types, while avoiding the programming errors that are common within C and C++.

Security within Java is built-in at its core. As such, applications created in Java can't be invaded from outside.

Finally, Java code is portable. Java code is compiled into an intermediate bytecode. That bytecode is distributed, and it is the responsibility of an individual Java Virtual Machine (JVM) to run that bytecode. Because JVMs exist for numerous platforms, the bytecode can run on each of those platforms without any programmer intervention.



Encapsulation

Encapsulation in Java is essentially a shield around the object. That shield protects the object from other objects looking directly into its code. In practice, this means that if someone wants to use an object they communicate to that object via its methods. What encapsulation does is to hide the implementation from the client who interacts with the object via its interface. In theory, this should allow implementations to change without adversely affecting the client. Encapsulation also allows the building of modular systems without many tightly coupled interdependencies.

As an example, in a business application we might allow customers to place orders for books. The customer doesn't care how we get them their book. They only care that they get it. Internally, we might have our own warehouse, or we might drop-ship directly from the printer. What we've done is "encapsulate" the delivery mechanism of the book from the customer.

Inheritance

The ability to extend a class so the new class inherits the behavior and data of its parent while adding its own differentiating characteristics. Inheritance is often described as an "is-a" relationship. Subclasses (child classes) are specialized

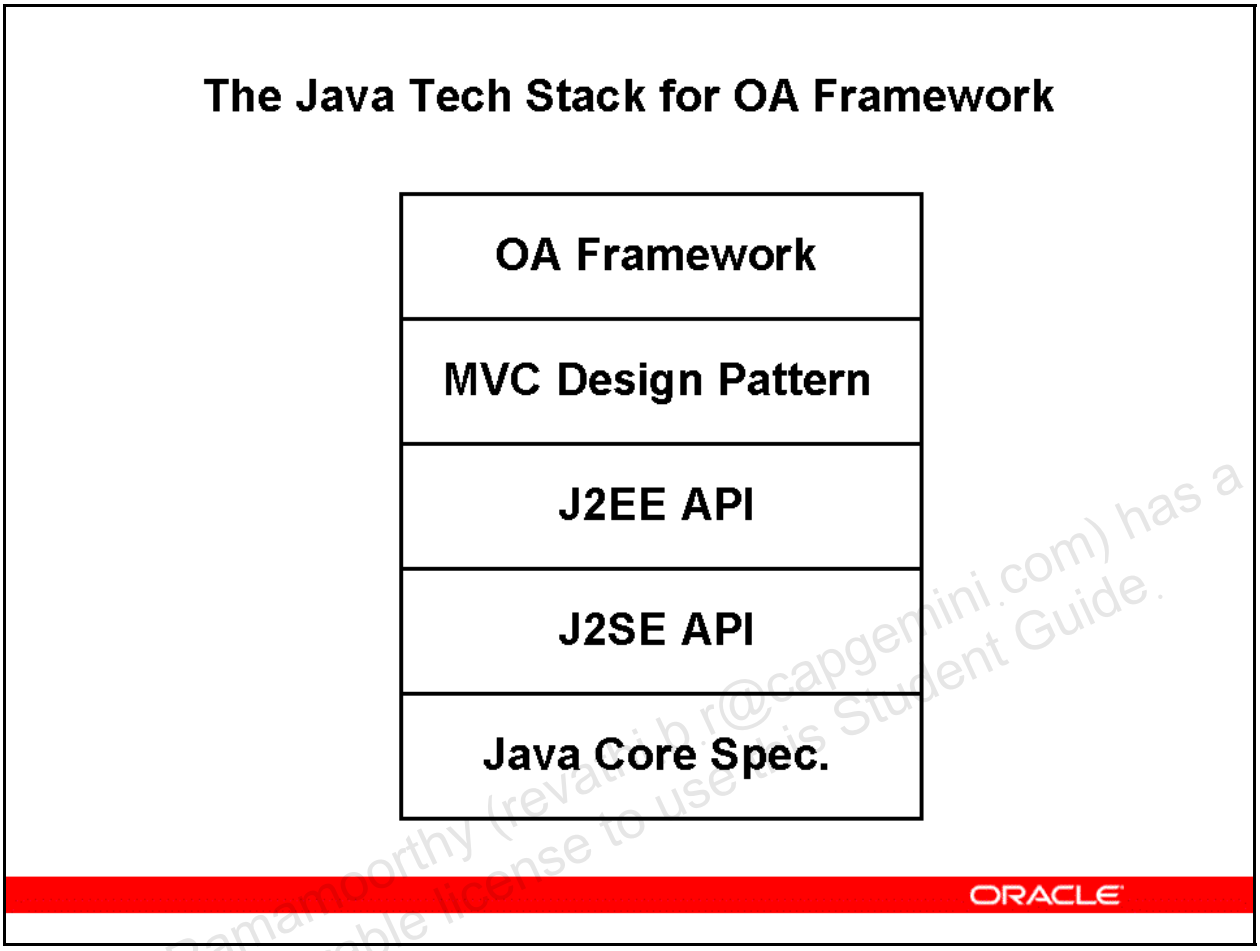
implementations of their more generic superclasses (parent classes). For example, assume you have a generic base class called Mammal with two subclasses: LandMammal and MarineMammal. You then subclass LandMammal to create the new class Cat, and subclass MarineMammal to create the new subclass Whale. A cat "is a" mammal the same way that a whale "is a" mammal (so they share many behaviors and characteristics), but they also have their own behaviors and characteristics: cats purr and whales swim.

Polymorphism

In general terms, "polymorphism" refers to a component's ability to take on many forms. In the inheritance hierarchy described above, we have a Whale class that extends MarineMammal, which in turn extends Mammal. The single class -- Whale -- is capable of assuming three forms: Whale, MarineMammal and Mammal. Similarly, the Cat class is a Cat, a LandMammal and a Mammal. If you were to create an array of Mammals, for example, you could include both Cat and Whale objects since they are both Mammals.

Taking this idea one step farther, you can also send the same message to these related classes which may result in distinctive class-specific behavior. Suppose we were to define an abstract method on our Mammal base class called move( ) ("abstract" means there is no implementation for this method at this level; the implementation must be supplied by a subclass). Then, we implement this method for our Mammal subclasses as appropriate: cats are quadrupeds that run, trot and walk while whales swim. Finally, we access the array of Mammals that we created above, and call the method move( ) on each object. This moves the animal from point A to point B using its species-specific method of locomotion.

The Java Tech Stack for OA Framework



All Java programs share a basic platform. Within Java development, we call these shared basics the core specification.

Java itself is a compact programming language. It is the APIs that account for the seemingly endless list of objects and methods. While there are other APIs within Java, these are the ones that are critical to us.

- The Java 2 Platform, Standard Edition (J2SE) 1.4 API. (You can learn more about this at: http://java.sun.com/j2se/corejava/index.jsp)
- The Java 2 Platform, Enterprise Edition (J2EE) 1.4 API (You can learn more about this at: http://java.sun.com/j2ee/index.jsp)

It is outside the scope of this course to teach you the fundamentals of Java. If you need additional exposure to Java, the New to Java center (new2java) is a free resource available at:

http://java.sun.com/learning/new2java/index.html

Java development puts together the methods and instance variables provided in the core specification APIs into common formats called J2EE Design Patterns.

Note: Design Patterns are not specific to the language (Java, for example) used to implement them. Design Patterns can be implemented in numerous programming languages, but for OA Framework we choose to implement them in Java, and the

primary pattern we use is the Model-View-Controller (MVC) pattern. MVC will be discussed in more detail later in the course.

Why patterns?

- Patterns work. Since the core requirement of any application is that it works, having a pattern to follow that already works is a tremendous boost.
- Patterns are reusable. Templates decrease the time it takes to type in large quantities of code. At best, it will reduce errors and cut the time it takes to develop new applications.
- Patterns are shared. Java development patterns allow a common shared understanding amongst various developers.

For examples of design patterns, see the book: Design Patterns -- Elements of Reusable Object-Oriented Software by Gamma, Helm, Johnson, and Vlissides (Addison-Wesley, 1995)

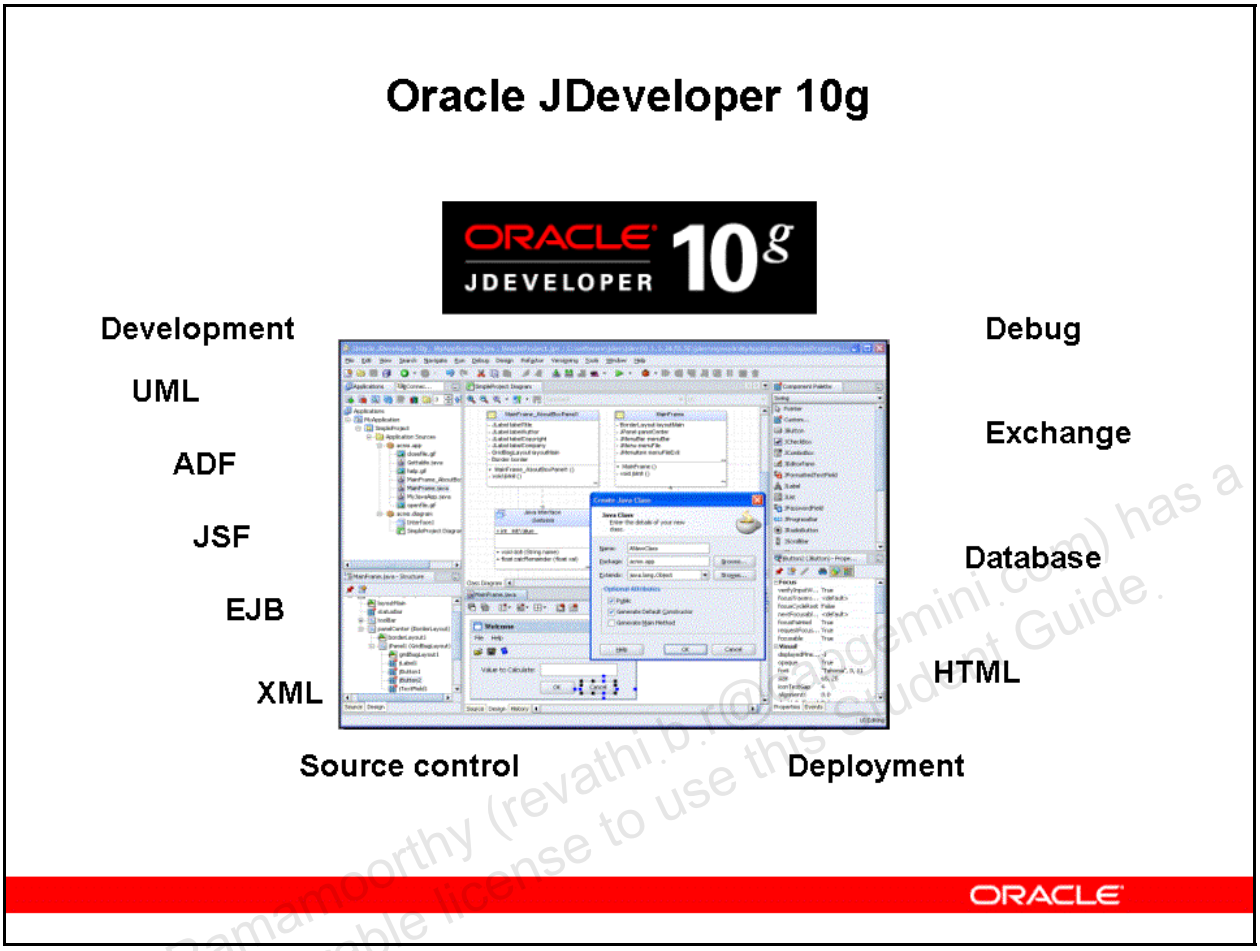
You may have heard of other Java frameworks like Struts or JFC/Swing. In the E-Business Suite, we use a framework called OA Framework. OA Framework is a Java framework that excels at creating 3-tier web-based applications that link to an E-Business Suite instance while maintaining all the security features of that instance.

A framework, then, is a specialized set of related classes designed to make application development easier.

In effect, a framework implements part of an application so developers don't have to write all of the application code from scratch. Developers use the framework as the basis for their work, so they can focus on the additional code required to implement their specific application requirements. Frameworks can also cooperate. For example, the OA Framework cooperates with the UIX and BC4J frameworks to provide a rich, comprehensive foundation for building web-based applications. BC4J handles database interaction, UIX handles web beans and HTML rendering, and the OA Framework "glues" these technologies together into a seamless technology stack including application security, personalization support and other Oracle E-Business Suite features.

Without these frameworks, a web application developer would have to implement all of this functionality in addition to her application's behavior. Not only do frameworks make development more efficient, but they also improve product quality. Each module is designed, developed and tested by developers with specific areas of expertise and focus.

Oracle JDeveloper 10g



Integrated Development Environment

The add-in API architecture of the Oracle JDeveloper integrated development environment (IDE) means that all the tool components (for example, navigator, editor, and modeler) share memory models and event systems. In this way, an update in one tool is communicated to another tool so that the latter can refresh its image or take other appropriate actions.

In Oracle 10g, the JDeveloper IDE is developed in pure Java. Synchronization between model and code can be set so that you can decide to work using one or the other user interface.

Customizable Environment

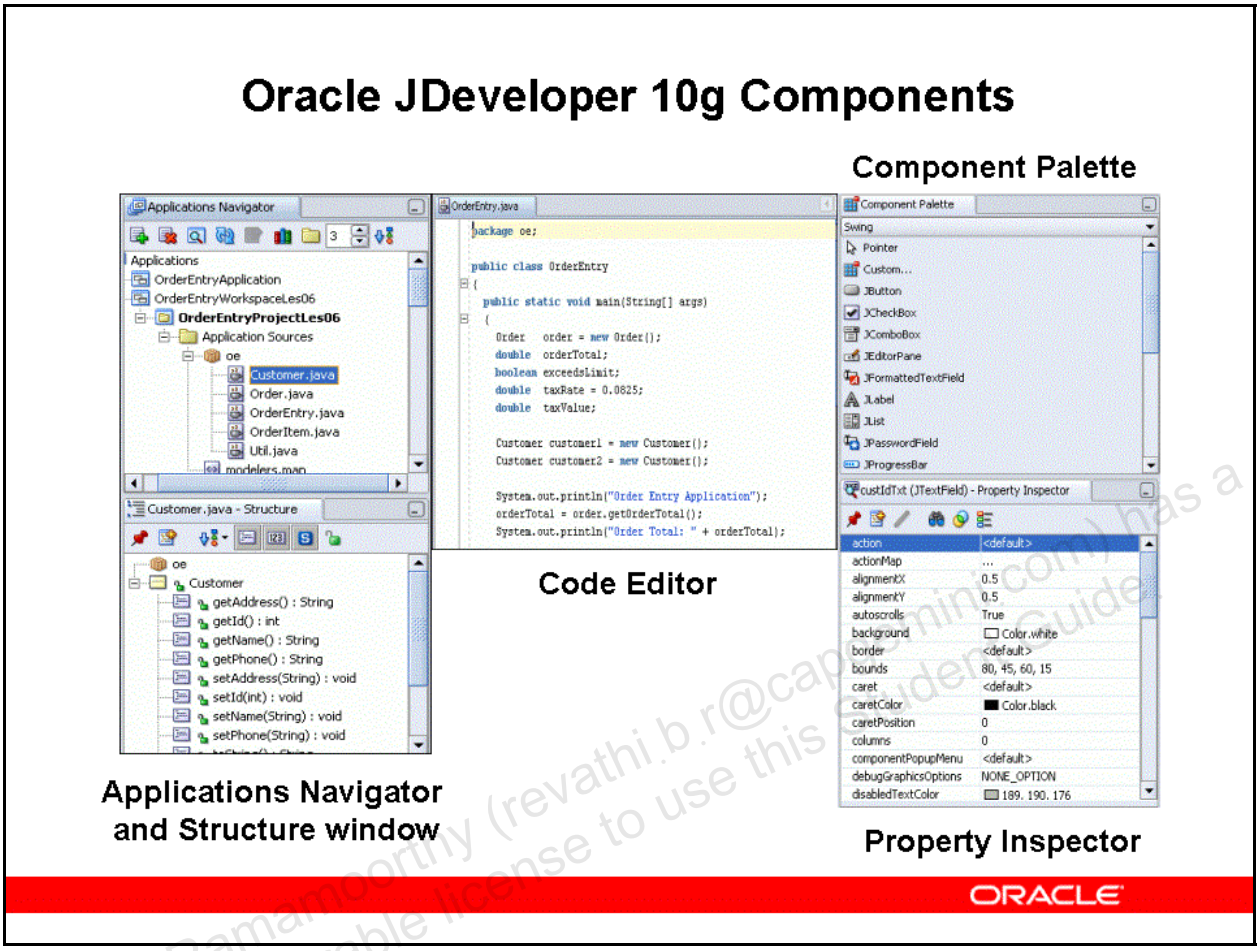
You can customize the JDeveloper IDE and arrange its look to better suit your project needs and programming style. To suit the JDeveloper IDE to your individual taste, you can:

- Change its look and feel
- Create and manipulate custom navigators
- Customize the Component Palette
- Customize the IDE environment

Select JDeveloper's embedded Java EE server

Arrange the windows in the IDE

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. The Component Palette in the slide displays Swing components that you use later in the course to build the user interface to a Java application.

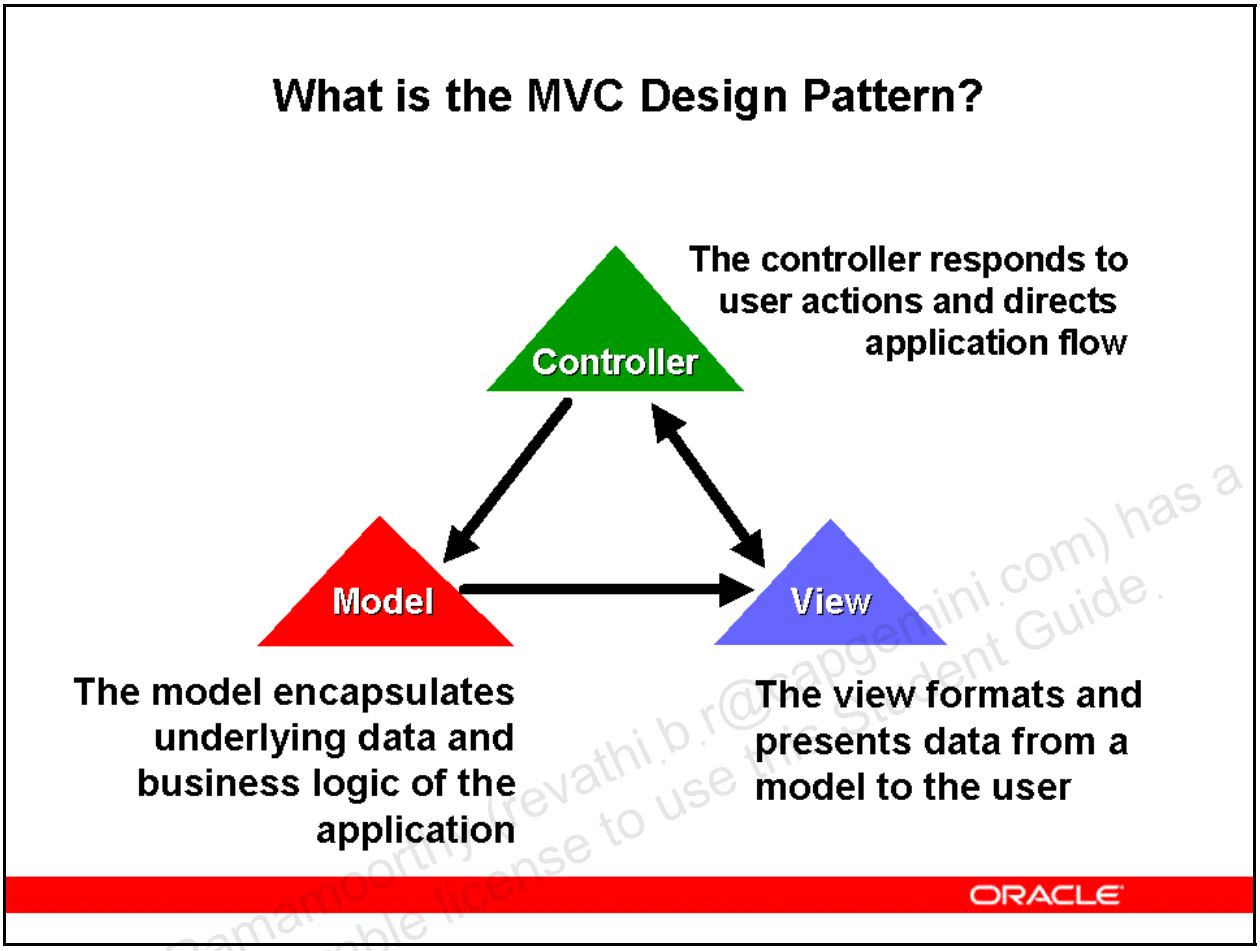
Applications Navigator and Structure window

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

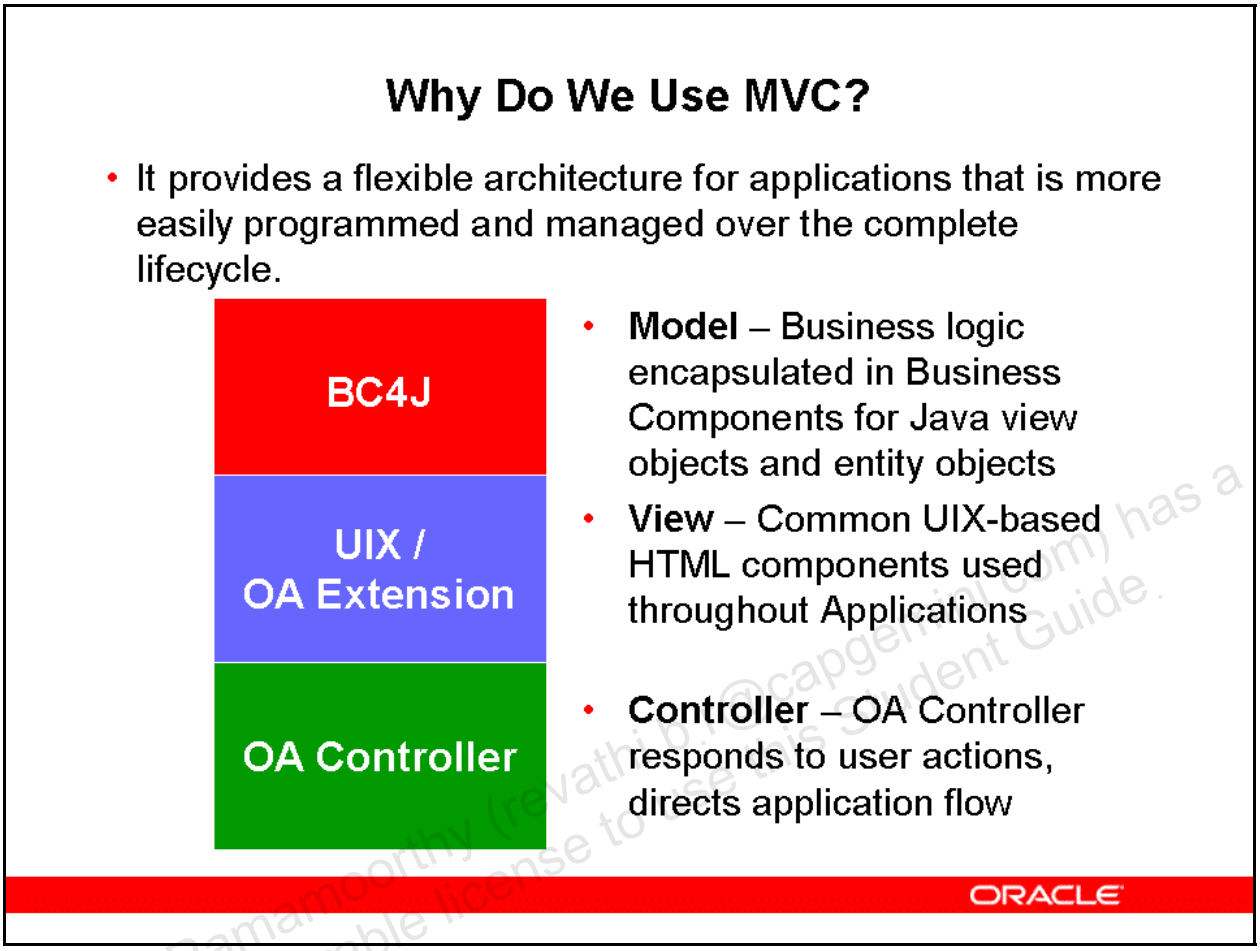
The Code (Design) Editor is where most of the work takes place; this is where you write code and design user interfaces. Open the editor by double-clicking the name of the file in the Navigator that you want to edit or view. The appropriate editor, code or design, will open based on your selection.

What is the MVC Design Pattern?



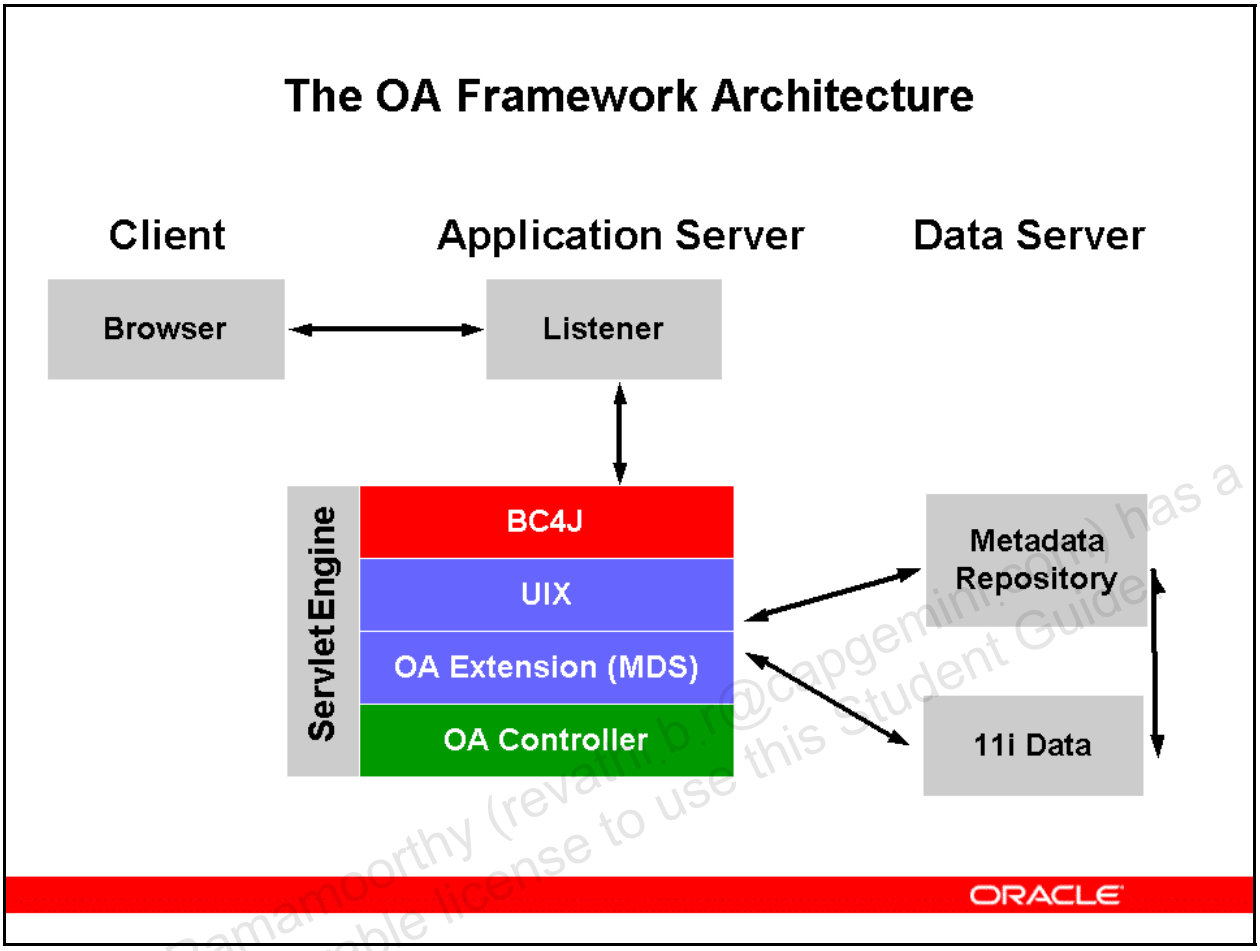
The term "view" has several meanings for Oracle-based applications (such as database view). In the case of the Model-View-Controller architecture, it helps to think of the view as the "end user's view of the data".

Why Do We Use MVC?



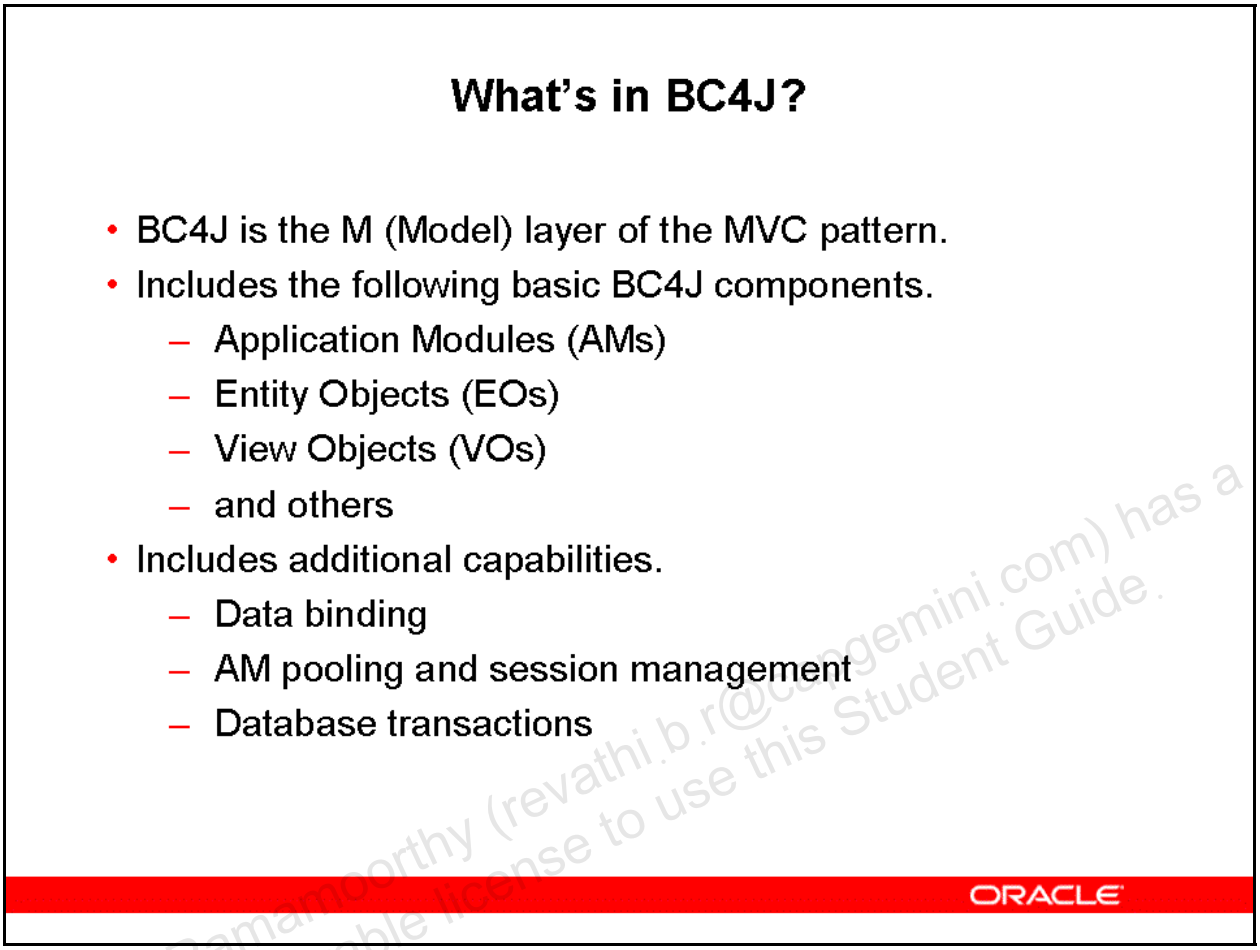
It is also the most popular and consistently used pattern for this type of development. While not a standard, it is still widely used and adopted, and Oracle followed with its own implementation of this pattern.

The OA Framework Architecture



In OA Framework, there is often a discussion of Client-Server. If you see this, it is not the same as the older client-server computing with which you might be familiar. The UIX and OA Extension components, on the middle-tier, are referred to as the Client. The BC4J components, on the middle-tier, are referred to as the Server. The OA Controller spans the two components, and provides communication between them.

What's in BC4J?

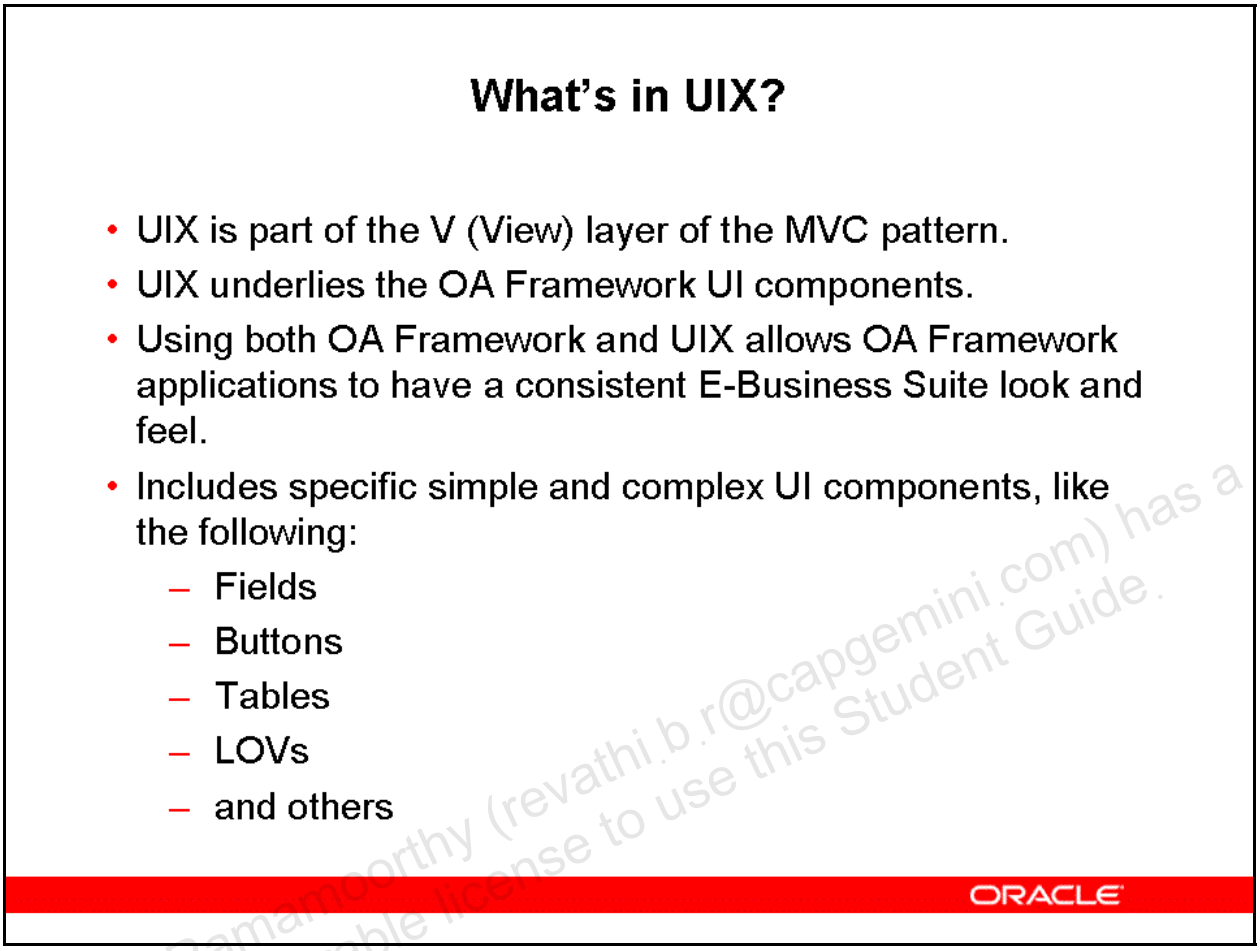


Note: In many of the error messages generated by BC4J, you will see a reference to oracle.jbo

Why?

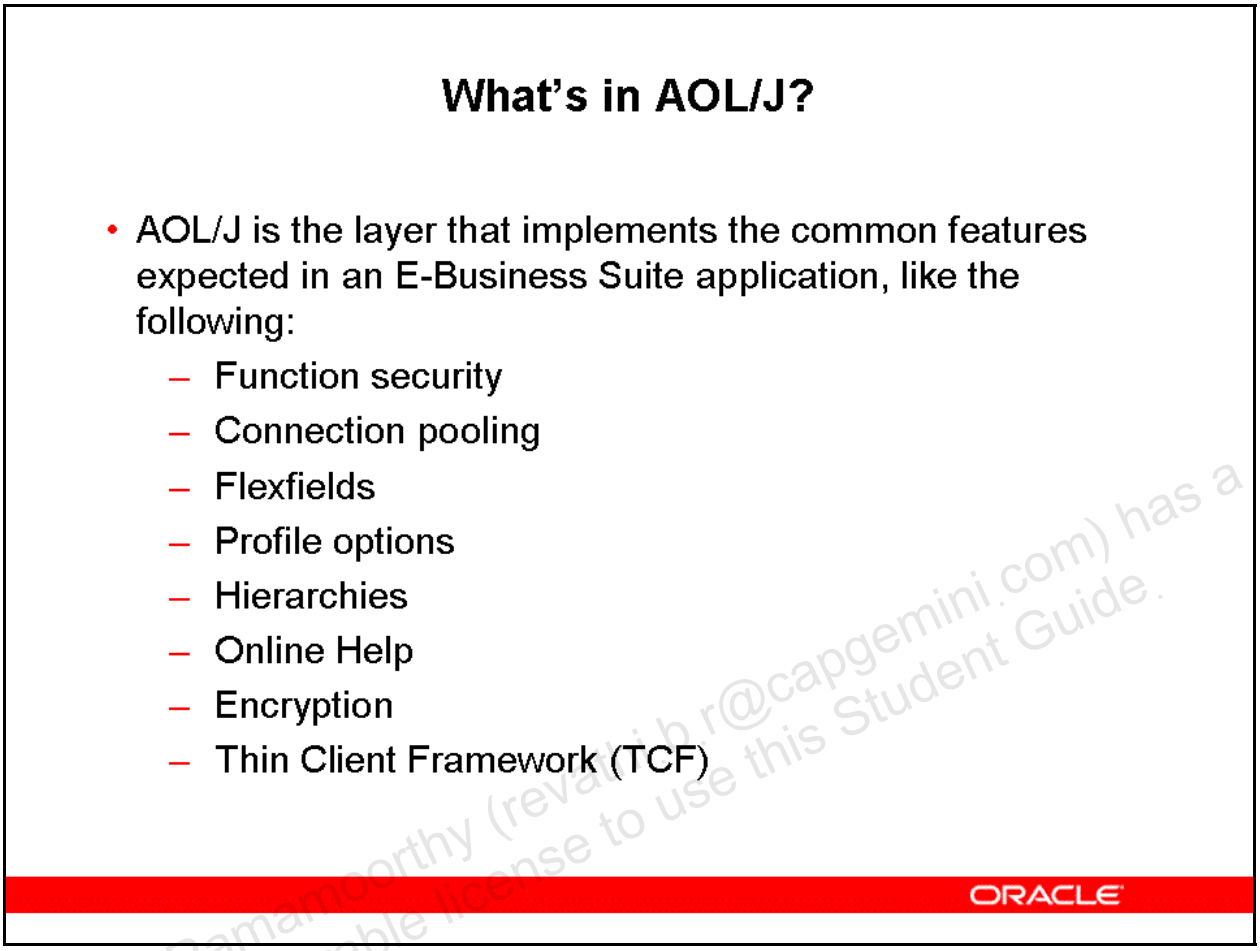
Originally, BC4J was named JBO (Java Business Objects). To maintain code, those APIs have not been refactored into a newer BC4J naming structure.

What's in UIX?



UIX is an Oracle standard. But, it allows a migration path to future technologies. For example, Java Server Faces (JSF) was chosen and developed as the standard over UIX. So, Oracle has taken UIX, implemented it as a JSF Implementation Library, and open sourced the code. So, all the UI components migrate very easily to JSF.

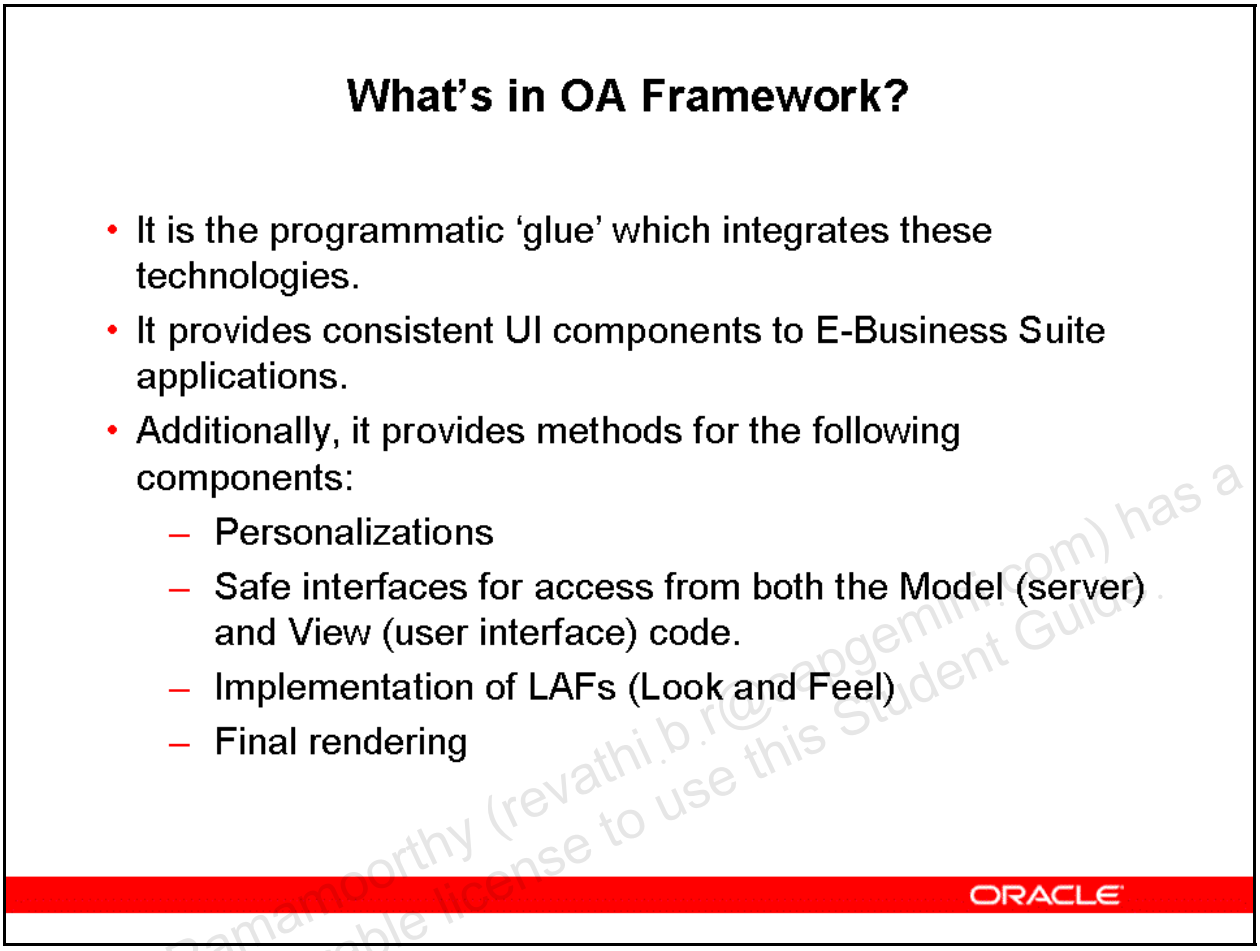
What's in AOL/J?



AOL/J is also responsible for managing and maintaining the application context (state) as you navigate within and on OA Framework pages.

Note: The Thin Client Framework (TCF) was a means to allow an older, Oracle Forms-based interface to run newer java-based components from within the Oracle Forms UI.

What's in OA Framework?



Note: In many of the error messages generated by OA Framework, you will see a reference to jrad….

Why?

Originally, this UI-level was called Java Rapid Application Development (JRAD). To maintain code, those APIs have not been refactored into a newer naming structure.

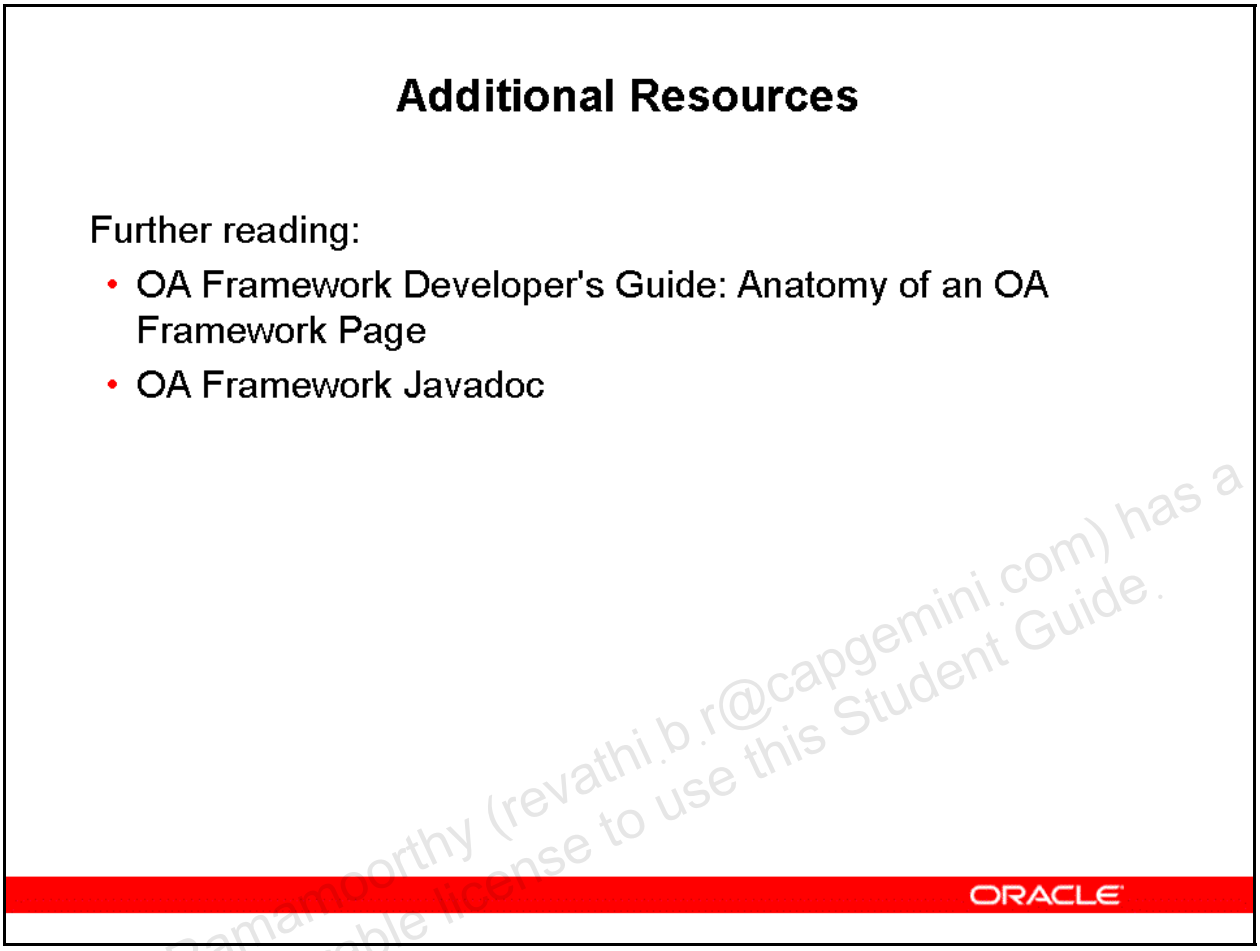
What's in the Metadata Services?

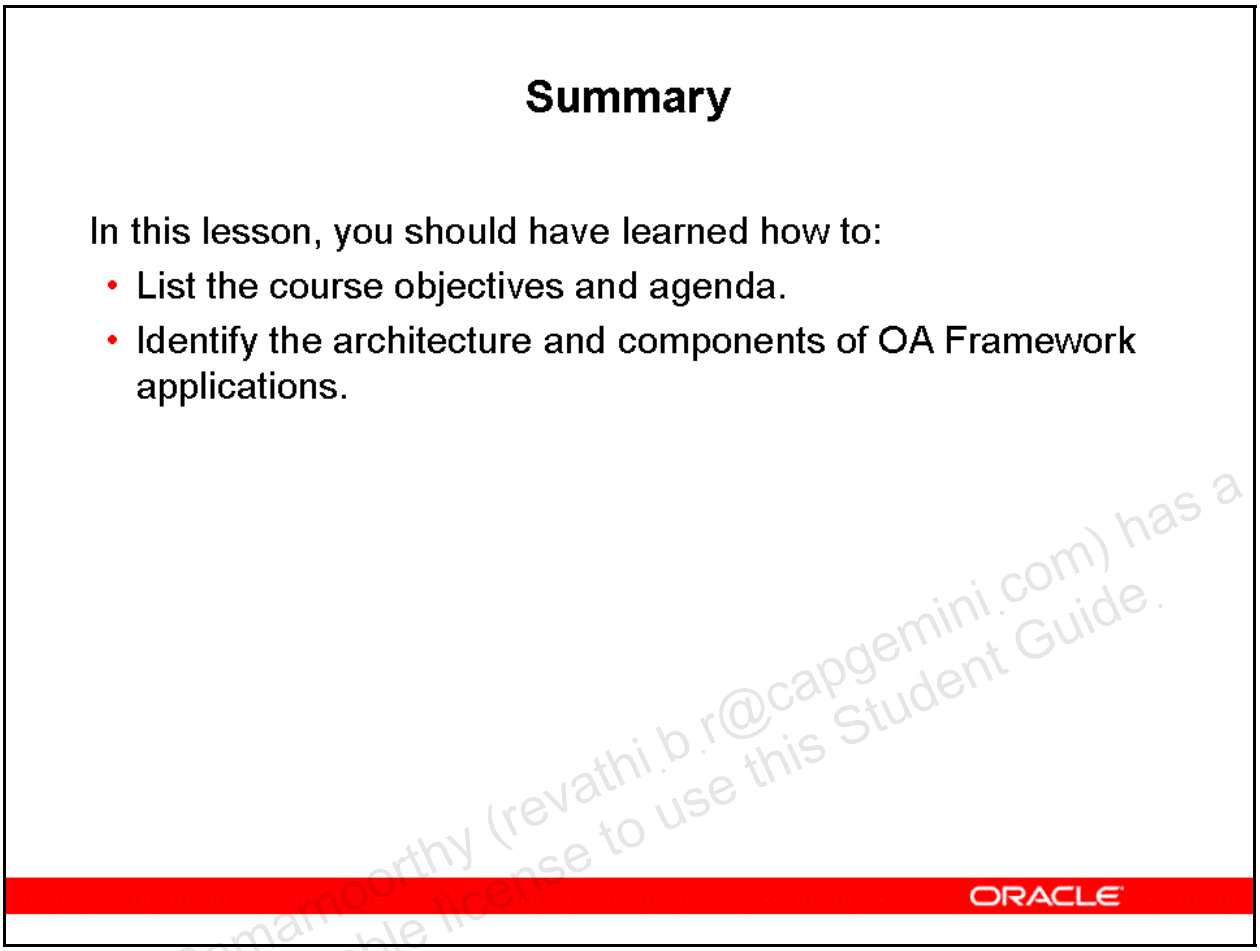
What's in the Metadata Services?

- OA Framework encourages the use of declarative objects.
- The declarative objects are stored in XML.
- XML has the advantage over Java in that it does not need to be compiled to be used.
- XML has limitations, and Java code is still needed to overcome those limitations.
- OA Framework stores most of the XML in the database. That database is called, the metadata services (MDS), and includes the following objects:
  - OA Framework Pages and Regions (PGs and RNs)
  - Personalizations
  - BC4J substitutions

ORACLE

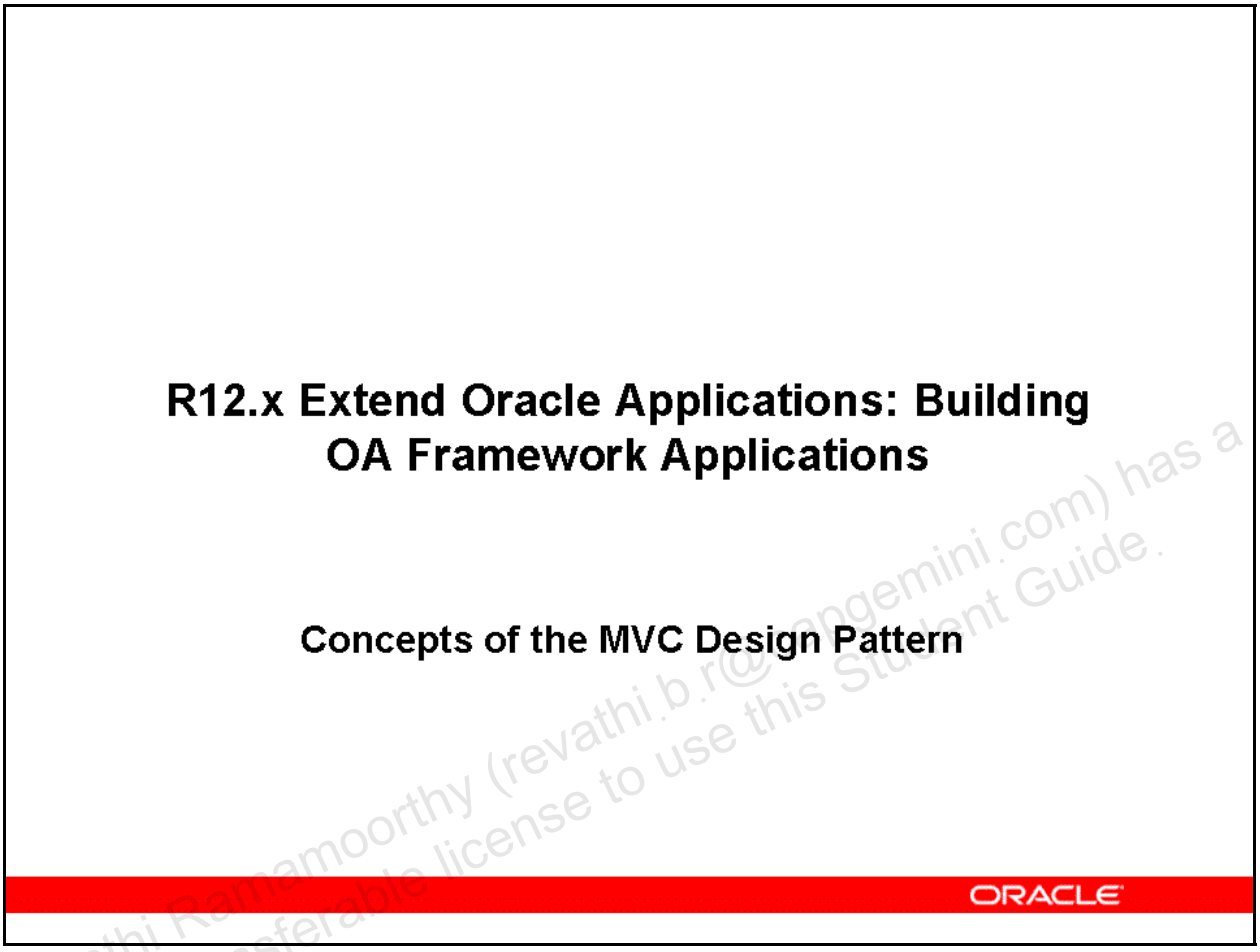
In can not be over emphasized, unlike most objects in E-Business Suite, the OA Framework declarative components are retrieved from the MDS. Even though the XML files may exist in the EBS file system, those are not the objects that are being executed.

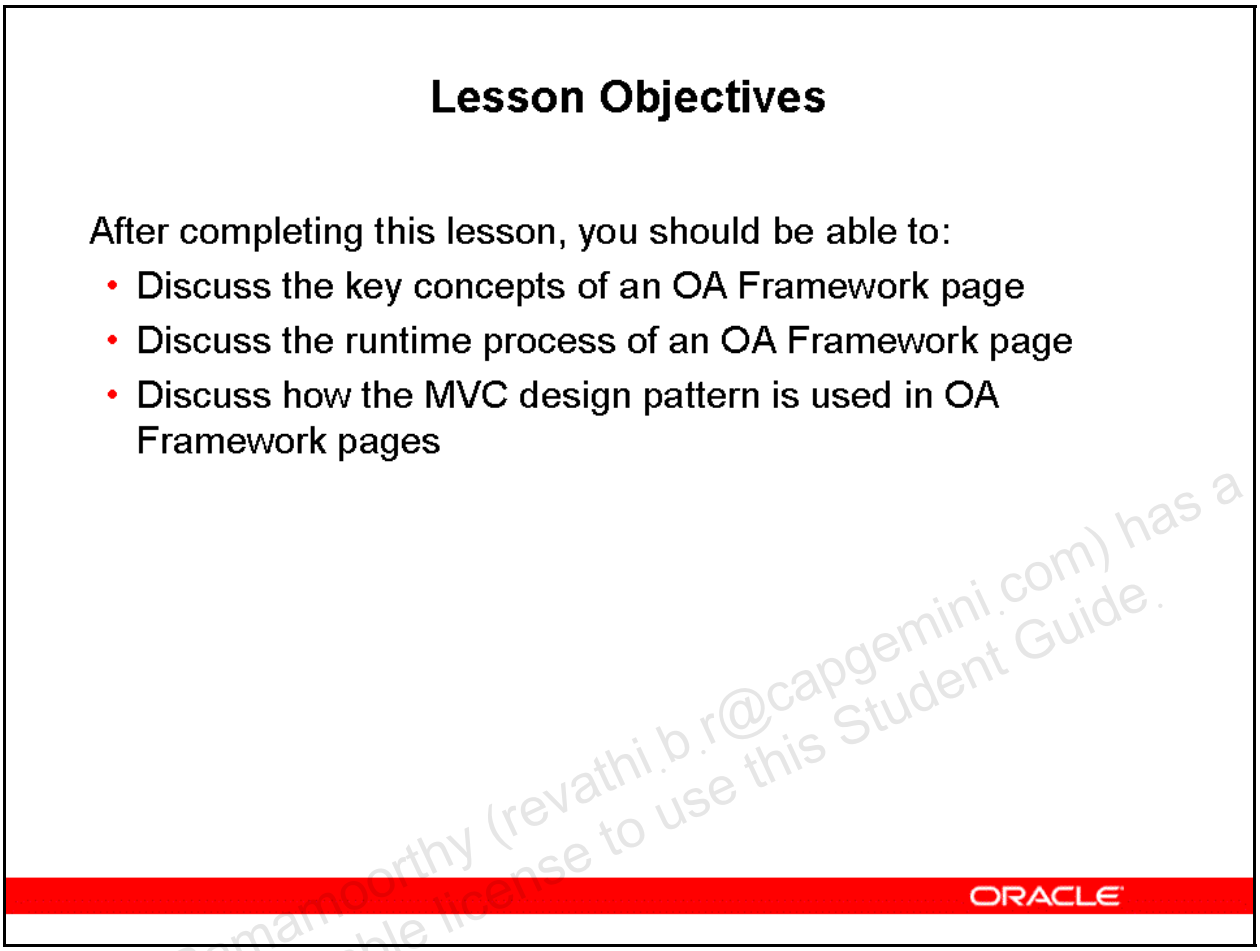




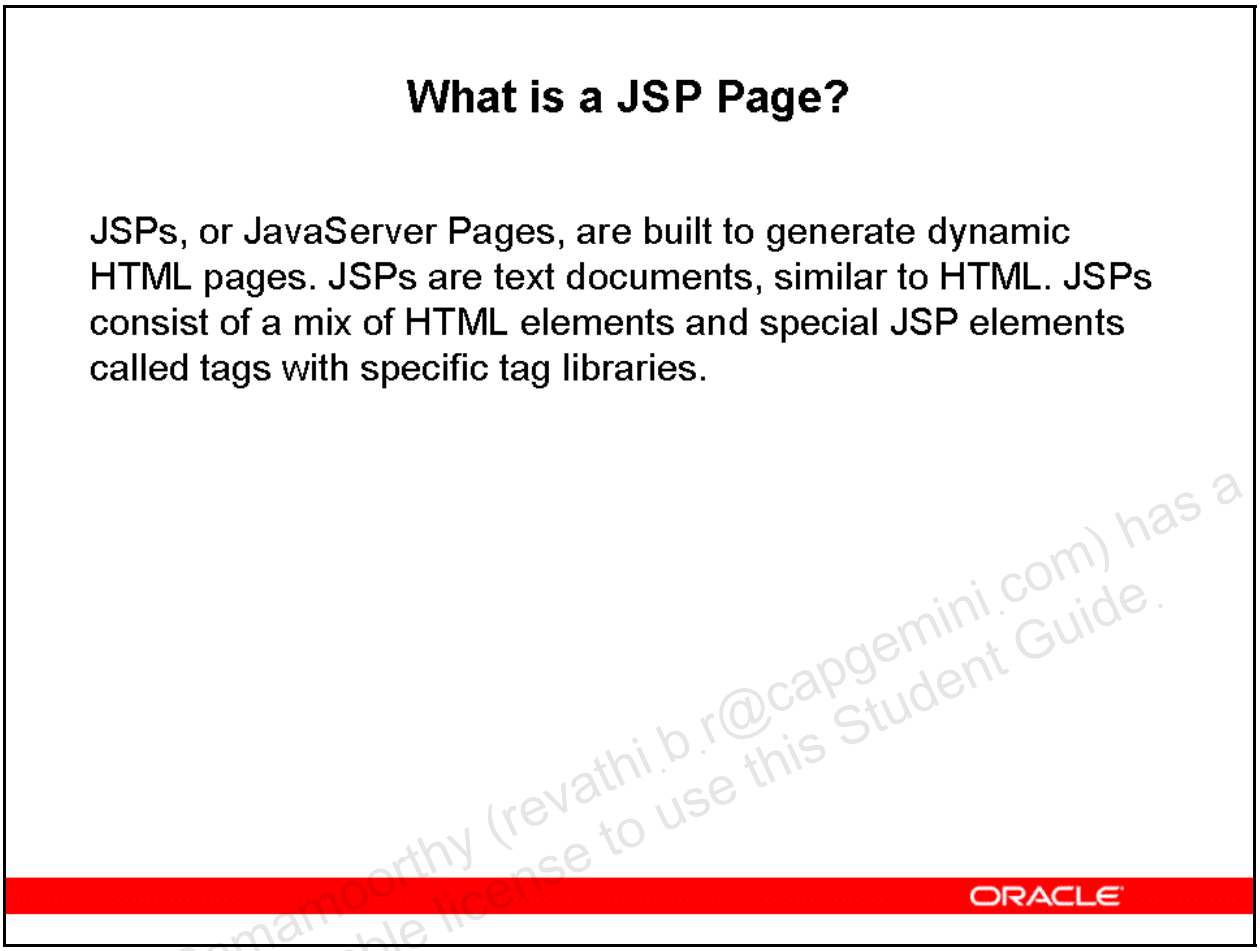
Concepts of the MVC Design Pattern

Chapter 2





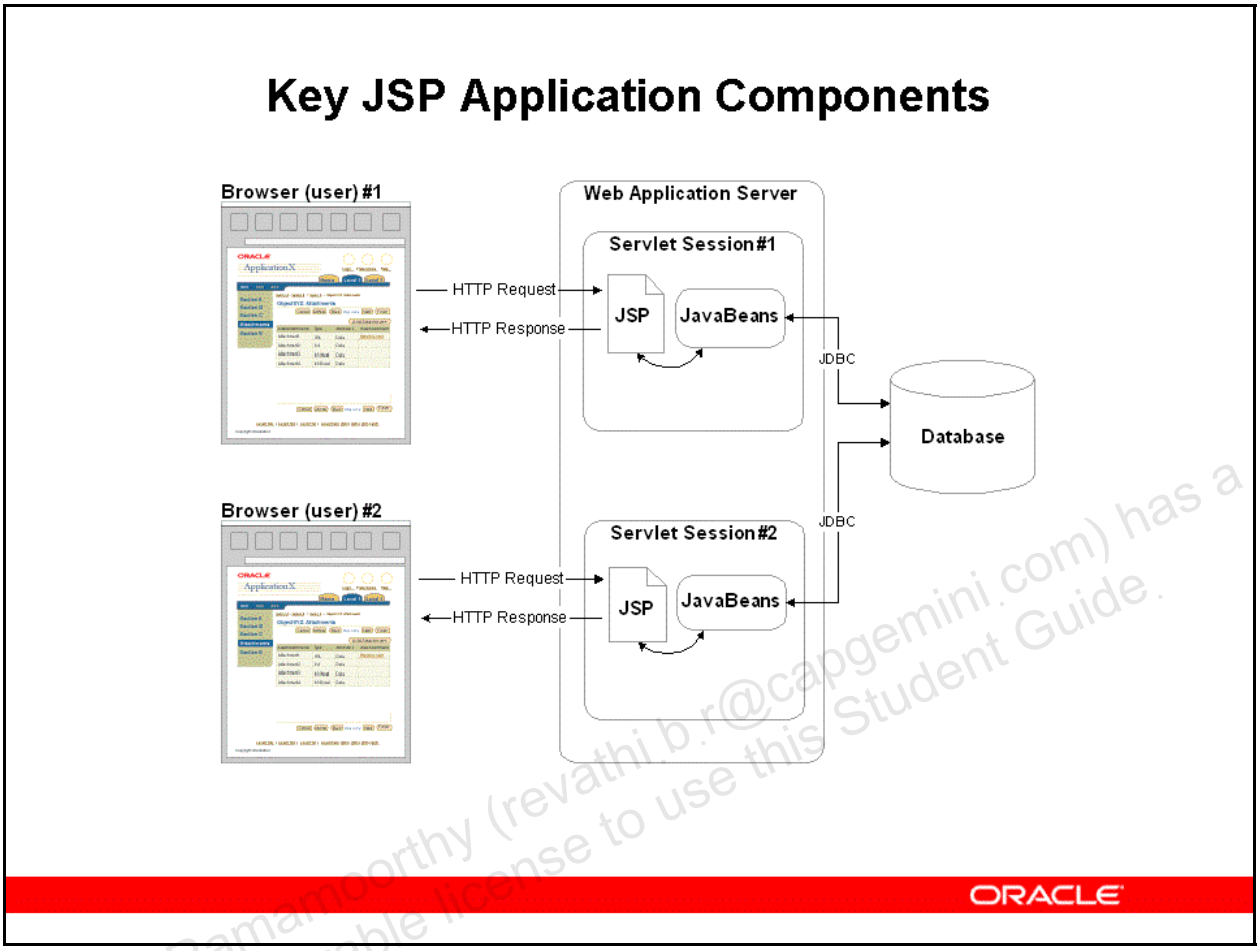
What is a JSP Page?



Typical JSP programming is plagued by two significant drawbacks. One, the UI is only as good as the programmer's ability to generate the proper HTML to generate that UI. Two, as a follow-on, making global changes to the UI via JSPs is difficult and labor-intensive.

Example: In typical JSP programming, you might have a global footer that is included on each page. If that footer is changed, all the pages that reference that footer must be changed. Typically, there are no tools for doing this globally. Programmers must update each page manually. OA Framework JSPs differ substantially from this model. The major difference is the use of UIX JavaBeans to implement the UI.

Key JSP Application Components

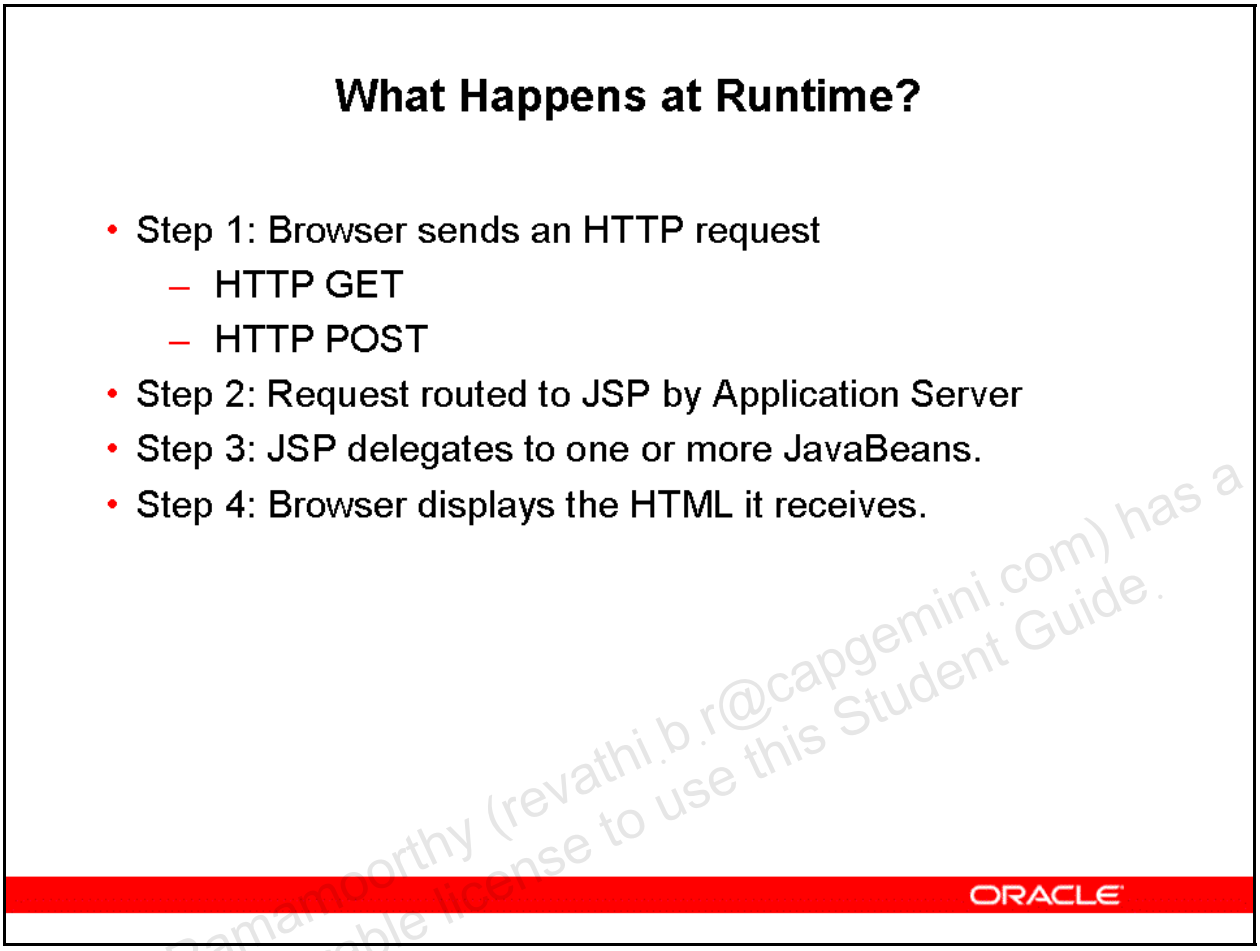


A typical JSP application involves the following components: a browser for client access, a database for enterprise data and a web application server ("middle tier") where the application objects live.

1. The browser communicates with the middle tier using Hyper Text Transfer Protocol (HTTP), which involves sending a request message to which the middle tier replies with a response message.
2. A JSP is a file with some HTML and Java code that executes top to bottom. At runtime, it is compiled into a Java class that is actually a servlet.
3. A servlet is a Java-based web application server extension program that implements a standard API.
4. 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.
5. A session may be initiated at any time by the application and terminated by the application, by the user closing the browser, or by a period of user inactivity.
6. A session usually corresponds to an application login/logout cycle
7. A JavaBean (or "bean" for short) is simply a reusable component that implements specific design patterns to make it easy for programmers and development tools to discover the object's properties and behavior.

8. Any objects in the middle tier that communicate with the database use a JDBC (Java Database Connectivity) driver.

What Happens at Runtime?



Step 1

When the user selects a link, a button or an active image, the browser sends an HTTP request to the web application server for processing. For the purposes of this introduction, we will focus on the two primary HTTP request methods (POST and GET) that are relevant for an OA Framework application.

HTTP GET

Whenever the user clicks a link or an image with an associated URL (like http://www.yahoo.com) the browser submits a GET request.

You can think of a GET as a postcard: both the address (URL) and any information the sender wants to convey (URL parameters) are written on the card itself (which is inherently space-constrained; how much can you write on a postcard?). This means that all the information for the communication is visible externally (and in an HTTP GET, all the data sent to the server is visible as parameters on the URL).

HTTP POST

Whenever the user clicks a button, image or link that performs a form submit in an OA Framework application (see What is a Form? below), the browser submits a POST request to the server (technically, a form can be submitted with a GET, but for the

purposes of working with the OA Framework, you can assume a form submit is a POST).

You can think of a POST as an envelope: the address (URL) is written on the outside, but the content within has the information the sender wants to convey. There's no limit to the amount of information that can be stored inside the envelope. Furthermore, the submitted data is not visible on the URL -- just as the contents of an envelope are not externally visible (although the metaphor isn't absolutely accurate: a developer could also add some parameters to the URL when doing a form submit).

What is a "Form?"

In simple terms, a "form" lets you collect data entered by users into "form fields," and send that data to the server for processing.

A form is an HTML construct that groups data entry controls like fields (both hidden and visible), poplists and so on with action controls (like buttons) that are capable of "submitting the form." When the user selects a submit button, for example, the browser issues a POST request which sends the form's data to the server. The term “form” is one of those definitions that is “overloaded” concerning Oracle. In the context above, we are using the word “form” in the generic sense of an HTML form.

Tip: People often use the terms "POST" and "submit form" interchangeably when talking about the OA Framework.

Step 2

The HTTP listener in the web application server routes the incoming request to the JSP. The developer's code does not know or care whether the browser issued a POST or a GET. All it does is read request values to determine what to do. So, for example, one of the request values might tell the JSP that a "Go" button had been pressed, which means it must execute a query.

Step 3

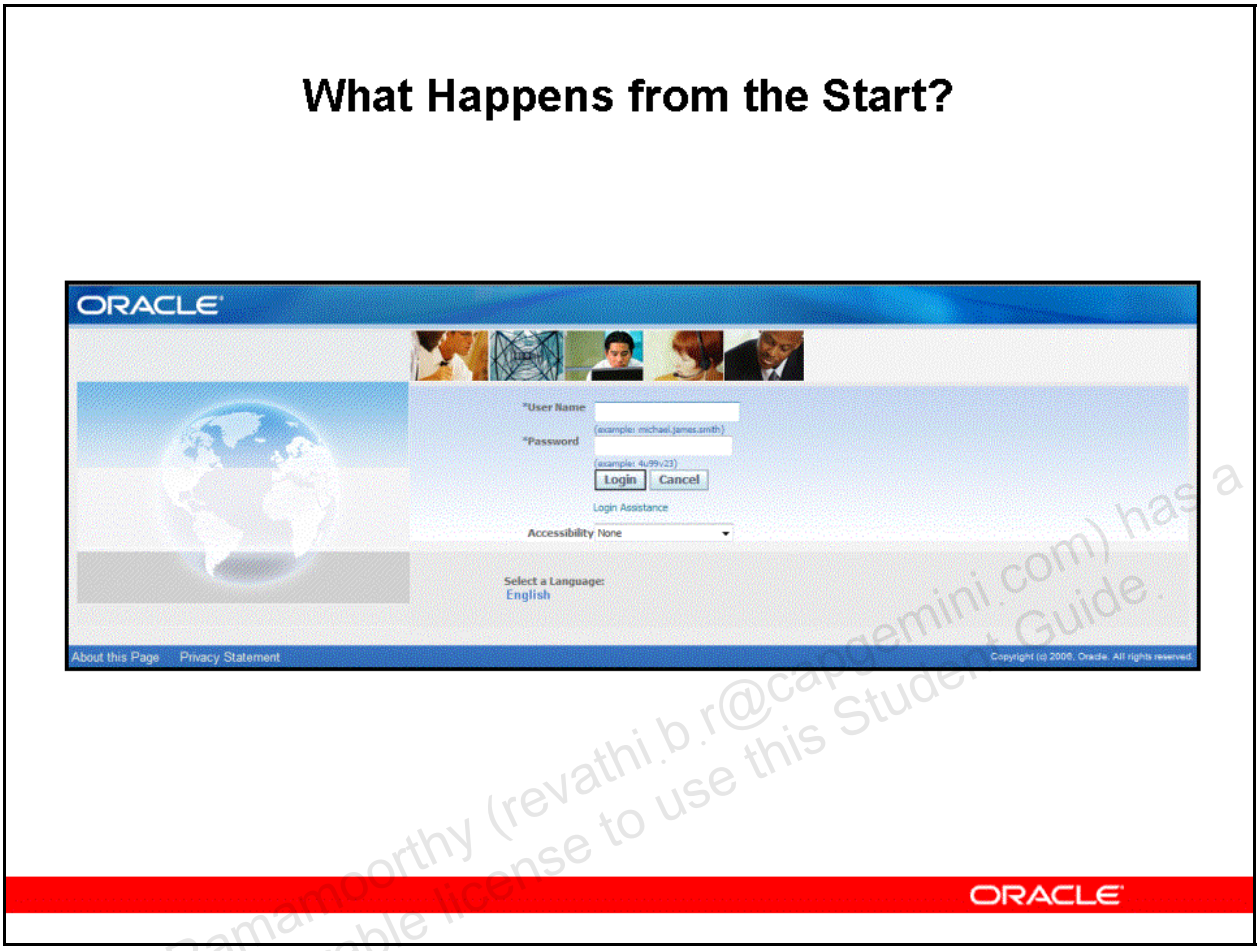
The JSP delegates to one or more JavaBeans that implement various behaviors including database interaction. Once they have completed their work, the JSP prepares the appropriate HTML content to send back to the browser in the response.

Note: We included the JavaBeans in this example just to make the point that in an application of any complexity -- and modularity -- the JSP does not do the application work on its own since you should not combine model, view and controller code in the same file. However, there is no absolute technical requirement for the JSP to work with any other Java classes, and if it does, there is no requirement that these classes be JavaBeans.

Step 4

The browser displays the HTML it received in the response.

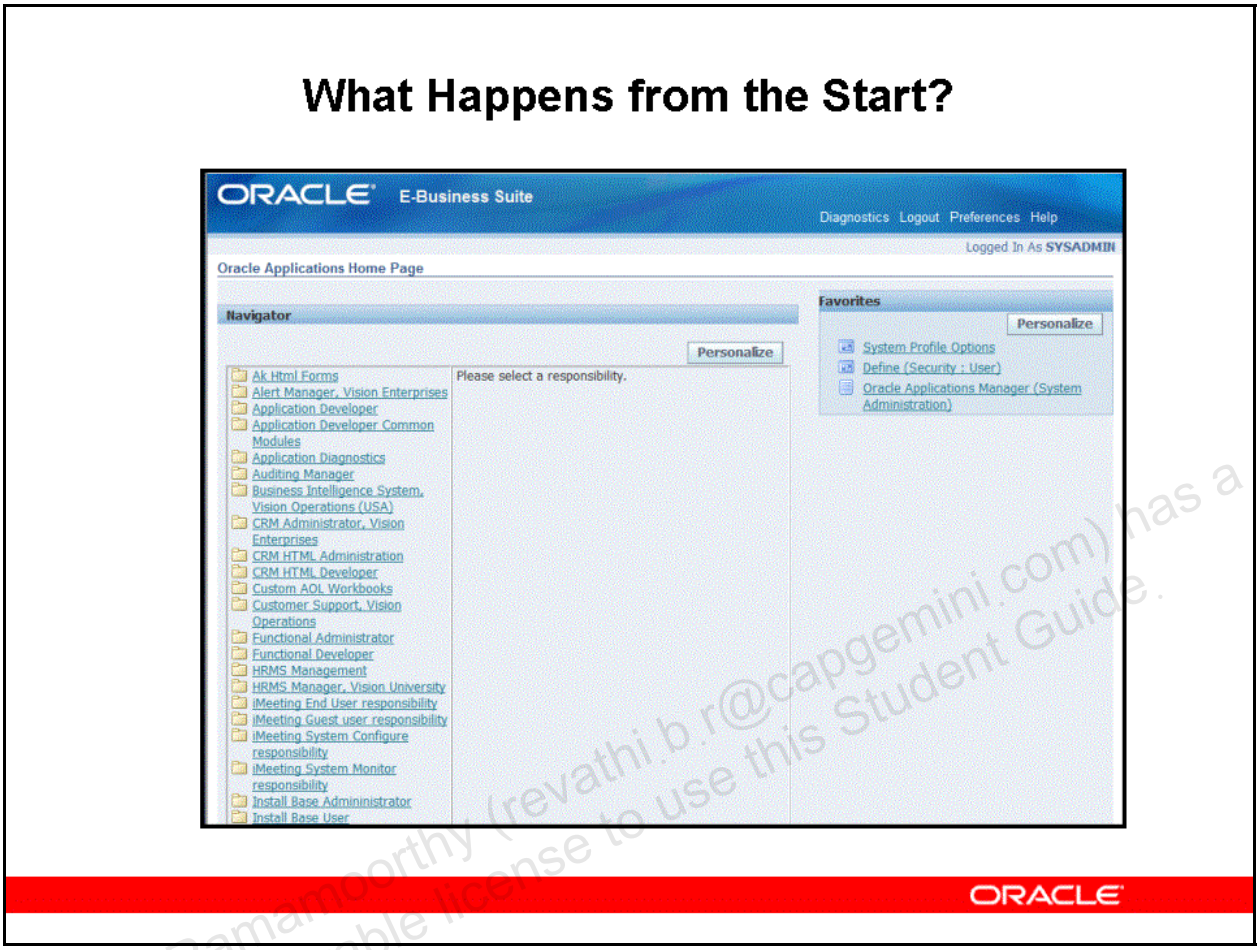
What Happens from the Start?



While it may seem trivial, just seeing the Login page indicates a lot of very good information. If you see this page, it means the following:

1. The database for your Applications instance is UP.
2. The TNS listener for your Applications instance is UP.
3. All of the critical Applications services, especially middle-tier services like HTTP Listeners,OC4J Servers, and other processes are UP.
4. Most importantly, all of those processes listed above are properly communicating with each other.

What Happens from the Start?



Assuming you have correctly logged in to the Applications instance various aspects are now in place:

1. When you logged in with your username and password, the username and password you entered was checked against valid users and passwords in the Applications instance. All of the communication of this sensitive data was encrypted. You would not see the Navigator (or Personal Home Page) if you did not enter a proper username and password.
2. Once your username and password were checked and authorized, a SESSION was created for you in the database. A special item, called a SESSION COOKIE, was passed to your browser. The SESSION and SESSION COOKIE combination ensure that you are who you say you are. Why is this important? It prohibits Applications users from cut and pasting URLs or bookmarking URLs to ensure proper security. From now on, every time you do something within Applications, the SESSION COOKIE on your browser will check itself against the SESSION in the database just to make sure your are secure. As an additional note on security, the SESSION and SESSION COOKIE have timeouts. They are only valid for a specific period of time. There are two timeouts that should interest you. The first timeout controls how long you can remain IDLE. In other words, how long can you go between interactions with the server? This is important for

performance of the servers, not security. The second timeout is a maximum limit, called a SESSION LIMIT. It simply says that after the session limit has expired (the default is 4 hours, but various systems can change this to longer or shorter periods of time), you MUST login again to the system. This timeout is important for security reasons.

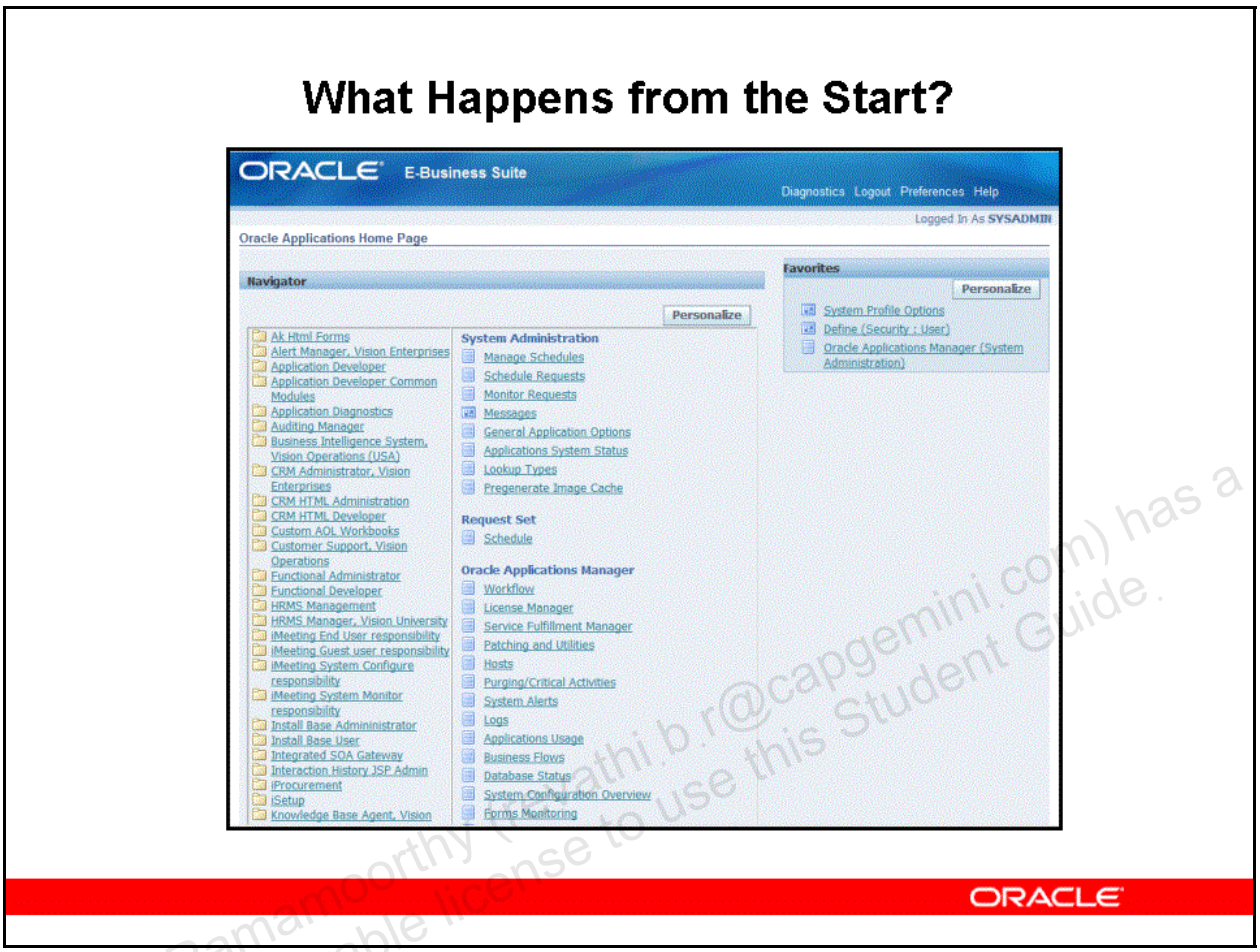
3. What has happened to see the Application Navigator? The Application looked at the username you entered and did the following:

4.

It read all the responsibilities that were assigned to your username. For each responsibility, it looked at the Menu that was assigned to that responsibility and any menu items that were specifically excluded from the menu assigned to you. For each menu, it read all of the individual submenus and functions that were assigned to that menu. After reading all of this data for your username, the Applications knows precisely what data and pages you can and can not see. In other words, it can enforce proper security to limit what you can do.

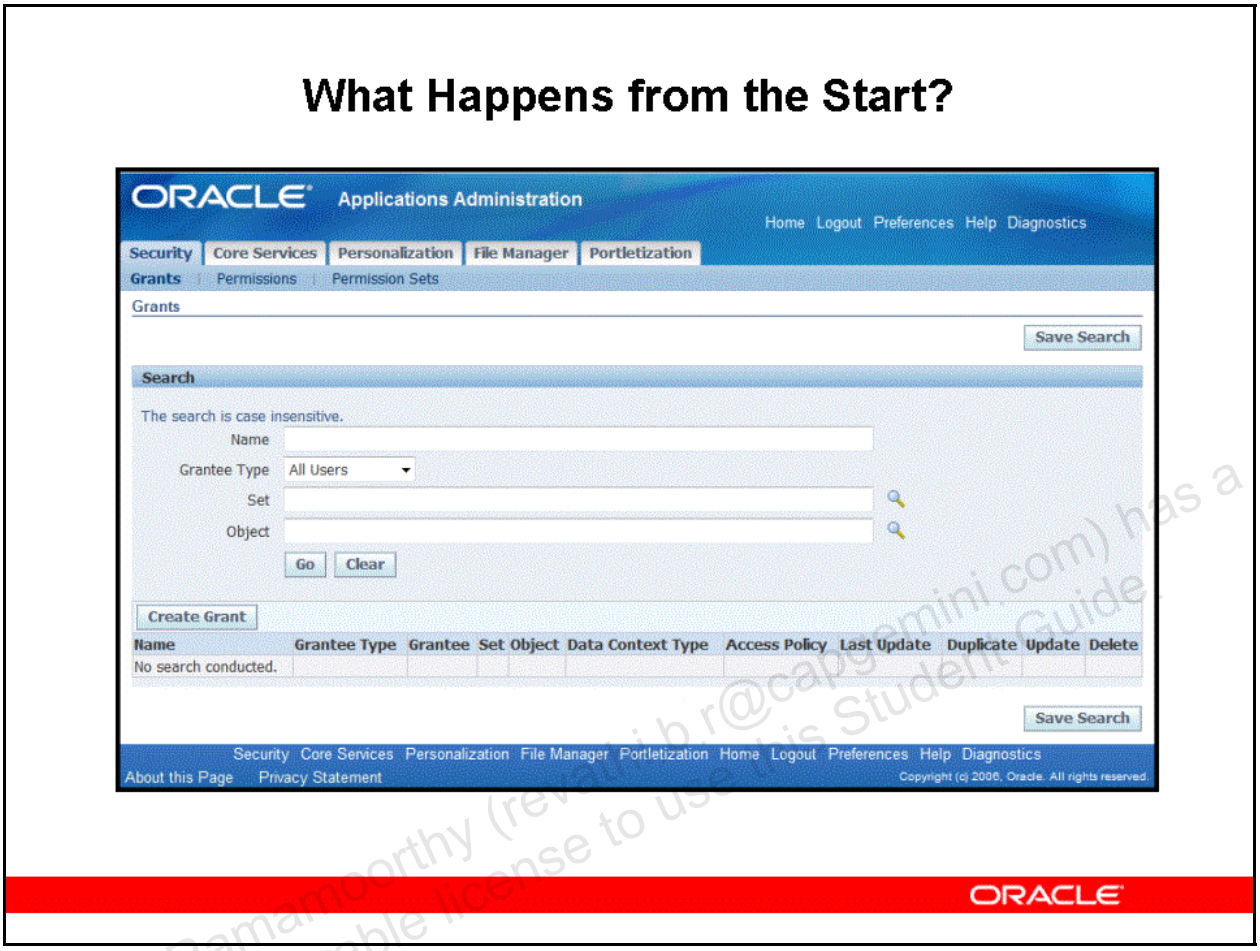
Finally, now that the Applications understand what you can and cannot do, it puts up a list of all the functionality to which you have access. The user, SYSADMIN, which is shown in the image above, has quite a broad range of functionality assigned to them.

What Happens from the Start?



Let's assume that you pick the System Administration responsibility. This responsibility has a complex menu made up of numerous submenus and individual pages.

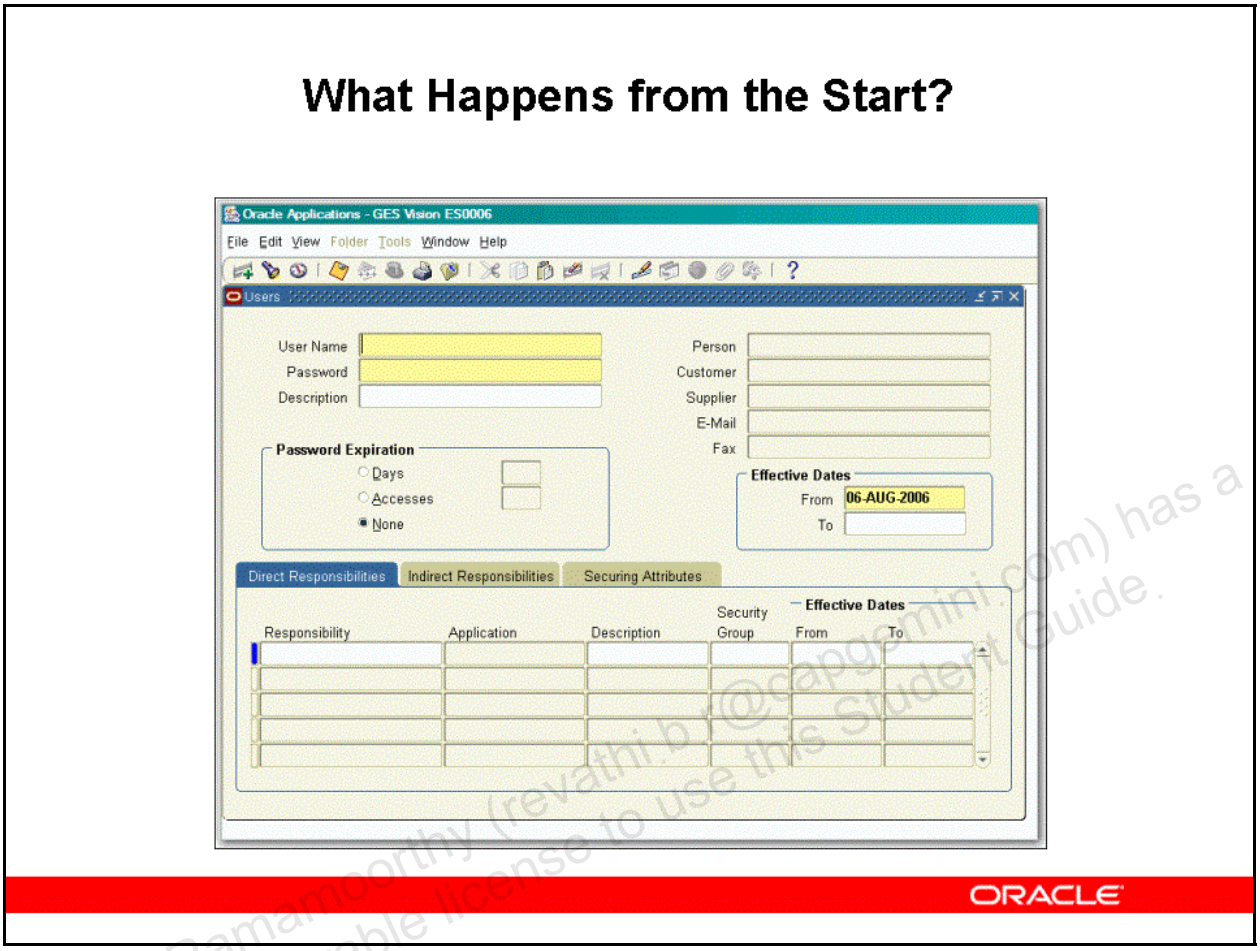
What Happens from the Start?



Some responsibilities have simple menus that include only one page. In this case, the example above assumes you selected the Functional Administrator responsibility.

Whether the menu is simple or complex, the menu will eventually lead to a individual page.

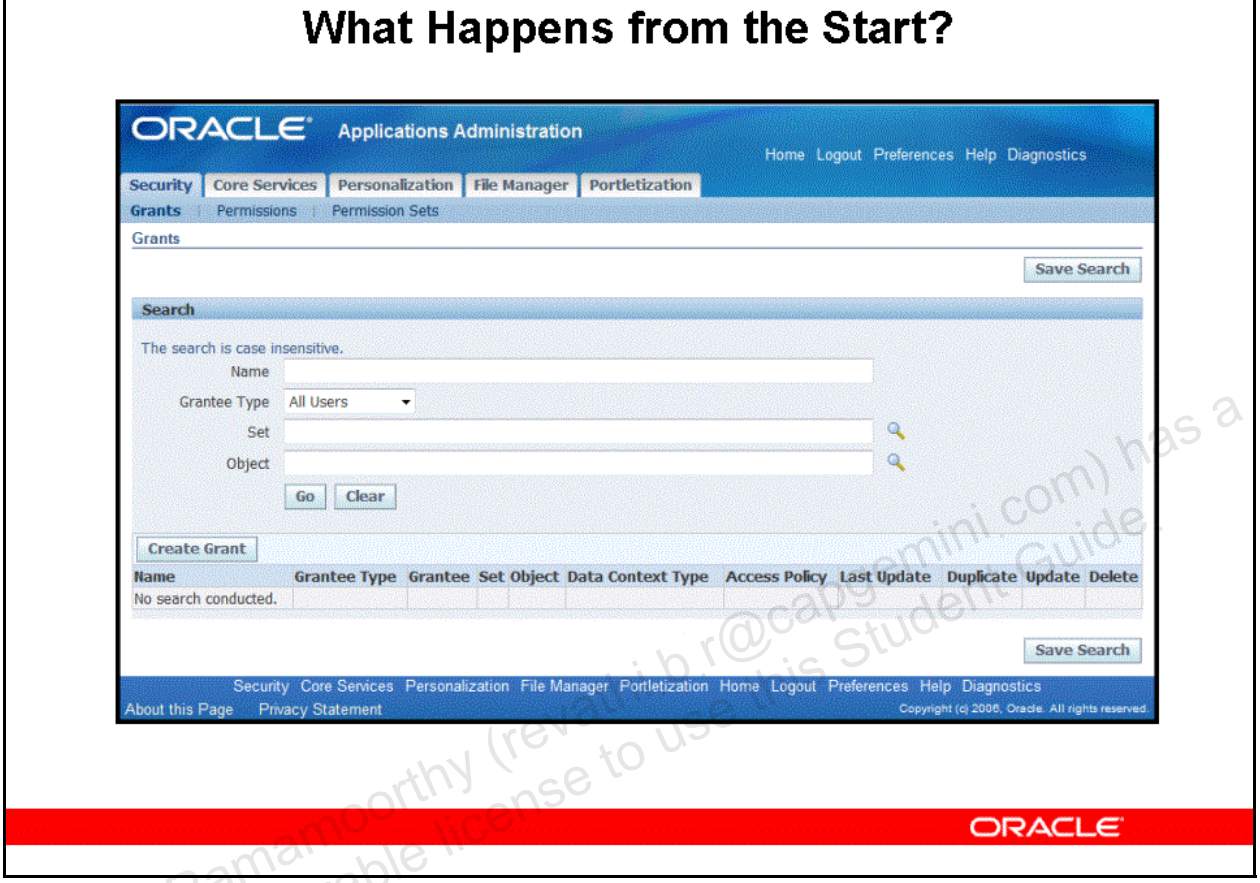
What Happens from the Start?



In some cases, the responsibility selected from the home page will utilize Oracle Forms-based pages.

Again, here the word “form” is overloaded and we use the term to describe an older technology that was used in Oracle Applications. Forms technology is beyond the scope of this course. It is also a technology that is being replaced with OA Framework-based pages.

What Happens from the Start?



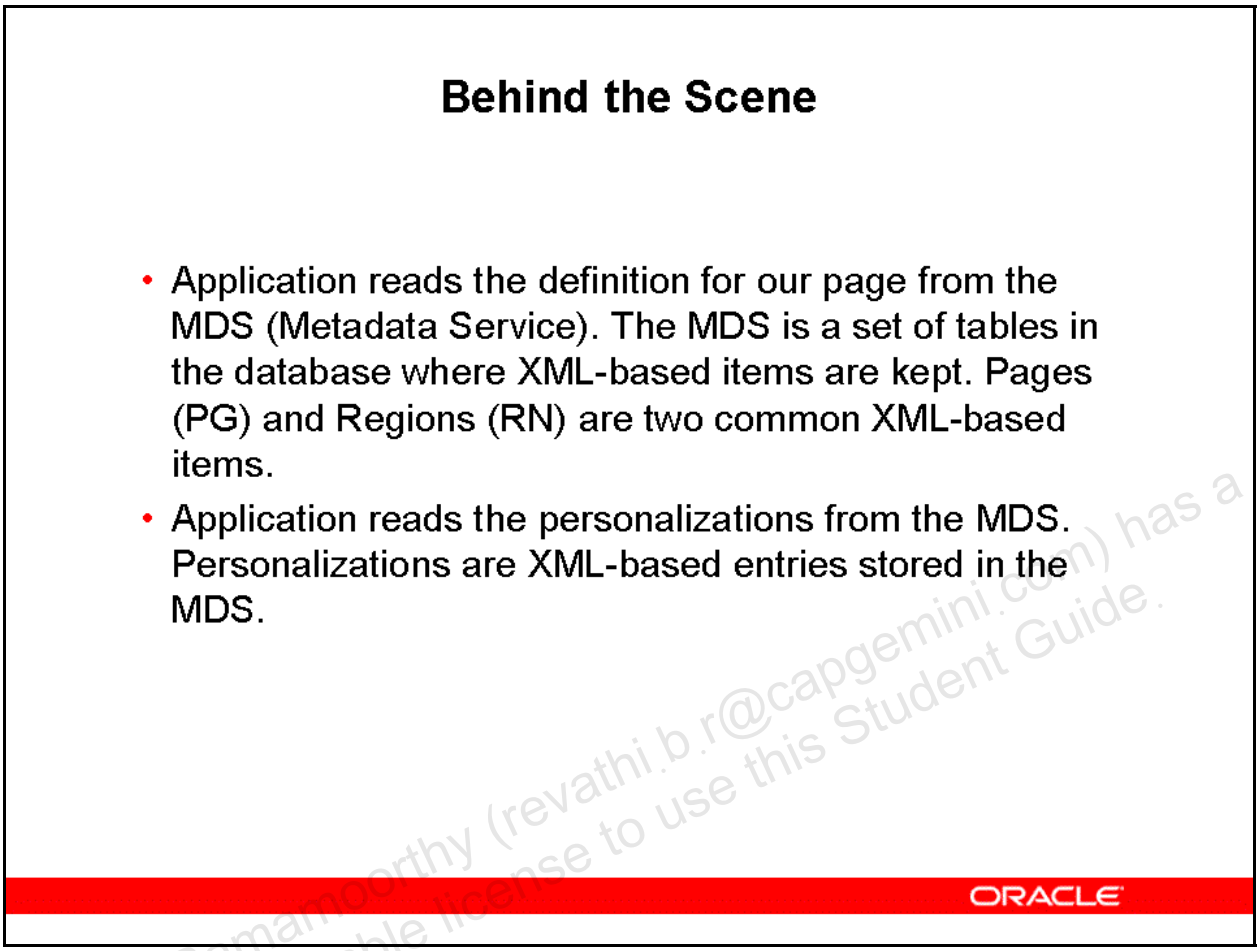
This is an OA Framework-based page. If you compare that with the Forms-based page on the previous slide, you will see that there are obvious visual differences in the two types of pages. While the visual differences are dramatic, the functionality that you can achieve with either is comparable.

Behind the Scene

- The user selected an item from the menu that pointed to a individual page to which that user was allowed to access. (Security first!)
- In the definition of that page (within Applications, this is called a Function), the function's definition called a specific page (e.g., GrantSummaryPG)
- By default, OA Framework pages are named with PG on the end of the name. Sometimes individual objects on a page are used in multiple places, those items are called Regions and named (by default) with names ending in RN.

ORACLE

Behind the Scene

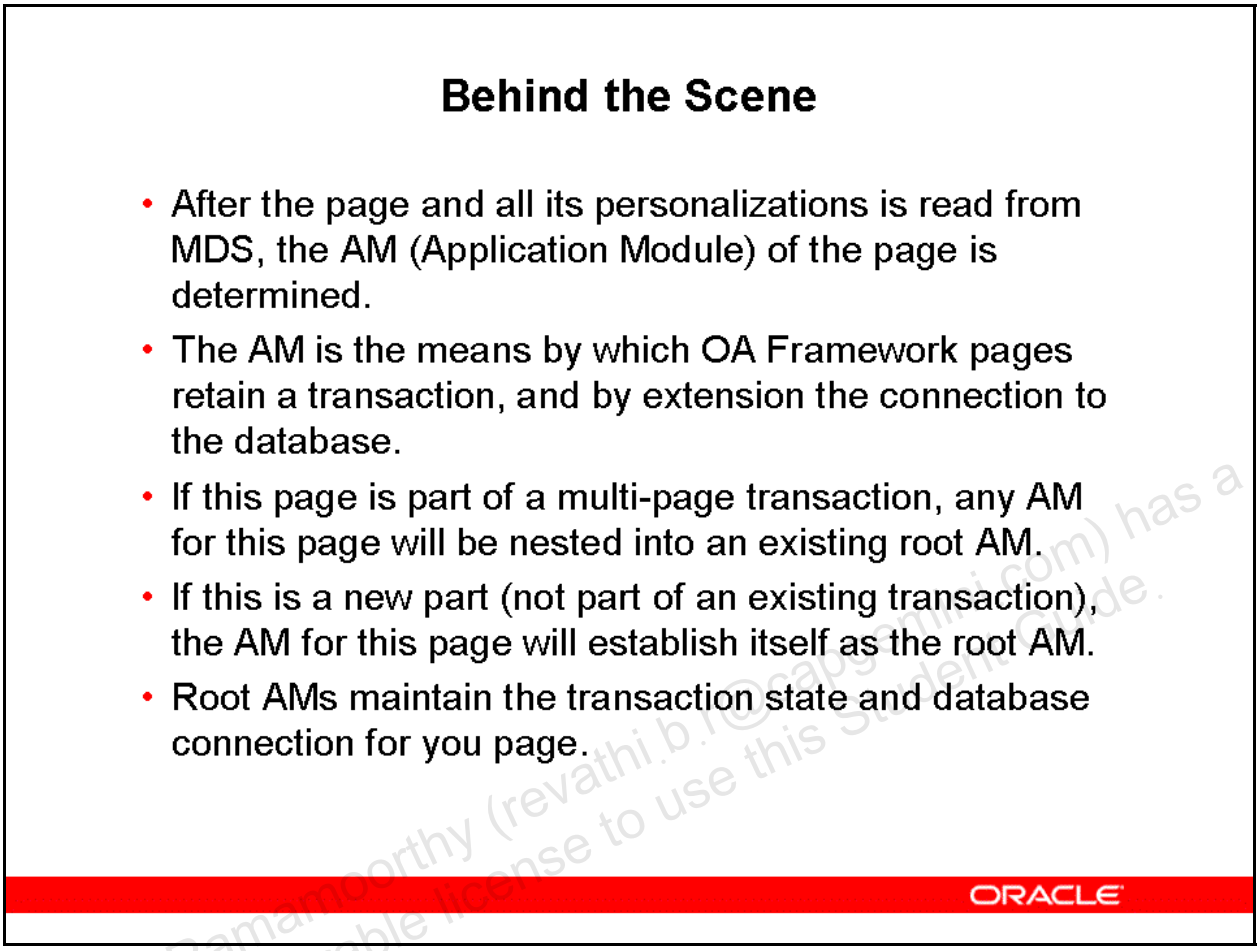


Personalizations add themselves like onion-layers to a page. The order in which those layers of personalization are applied are as follows:

1. Base page
2. Function-level
3. Location-level
4. Site-level
5. Organization-level
6. Responsibility-level

Each progress layer up the hierarchy includes all the changes beneath it, plus any changes that it adds to the mix. Understanding this hierarchy is important to debugging unusual behavior.

Behind the Scene

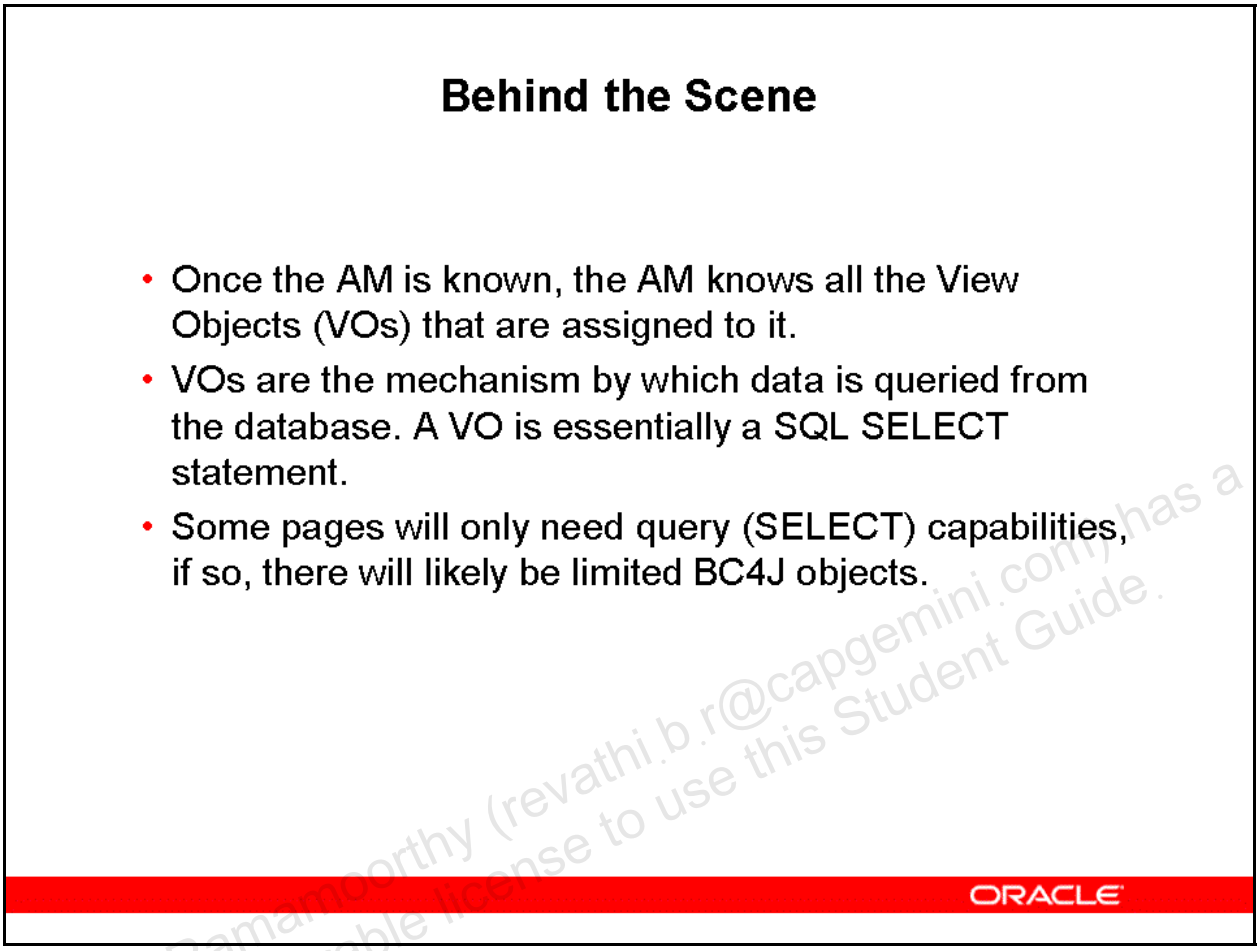


In order to read the AM, there are a couple of files the page will need to find. Those files will be stored on the server in a location referred to as $JAVA_TOP. In this case, the AM related files are:

<AMName>AM.xml (this is the XML-based definition of the AM. While it is XML-based, it is not stored in the MDS.)

<AMName>AMImpl.class (this is the Java-based programmatic capability of the AM. The XML-based capabilities are limited. In most cases, the needed capabilities of the AM have to be programmed in Java to accomplish their task.)

Behind the Scene



The VOs are, like the AM, stored in $JAVA_TOP. The VO related files are:

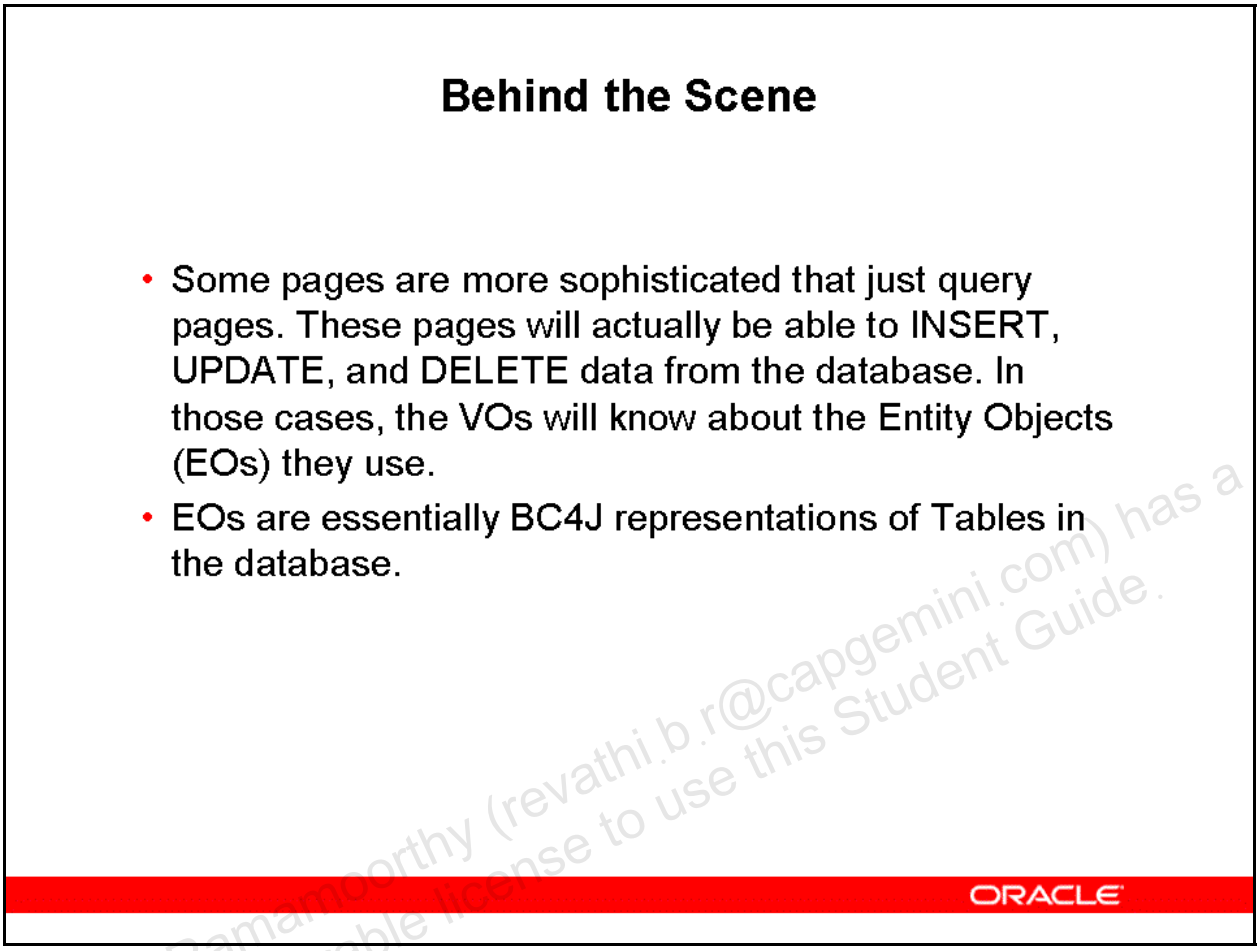
<VOName>VO.xml (this is the XML-based definition of the VO. While it is XML-based, it is not stored in the MDS.)

<VOName>VOImpl.class (this is the Java-based programmatic capability of the VO. The XML-based capabilities are limited. In most cases, the needed capabilities of the VO have to be programmed in Java to accomplish their task.)

<VOName>VORowImpl.class (this is a pre-generated Java-based file. It needs to be present, but rarely, if ever, is anything actually done with this pre-generated file.)

There may be multiple VOs, each with their set of files. There are also view-level BC4J objects that may be referenced called View Links (VLs). VLs are the means by which VOs can be joined.

Behind the Scene



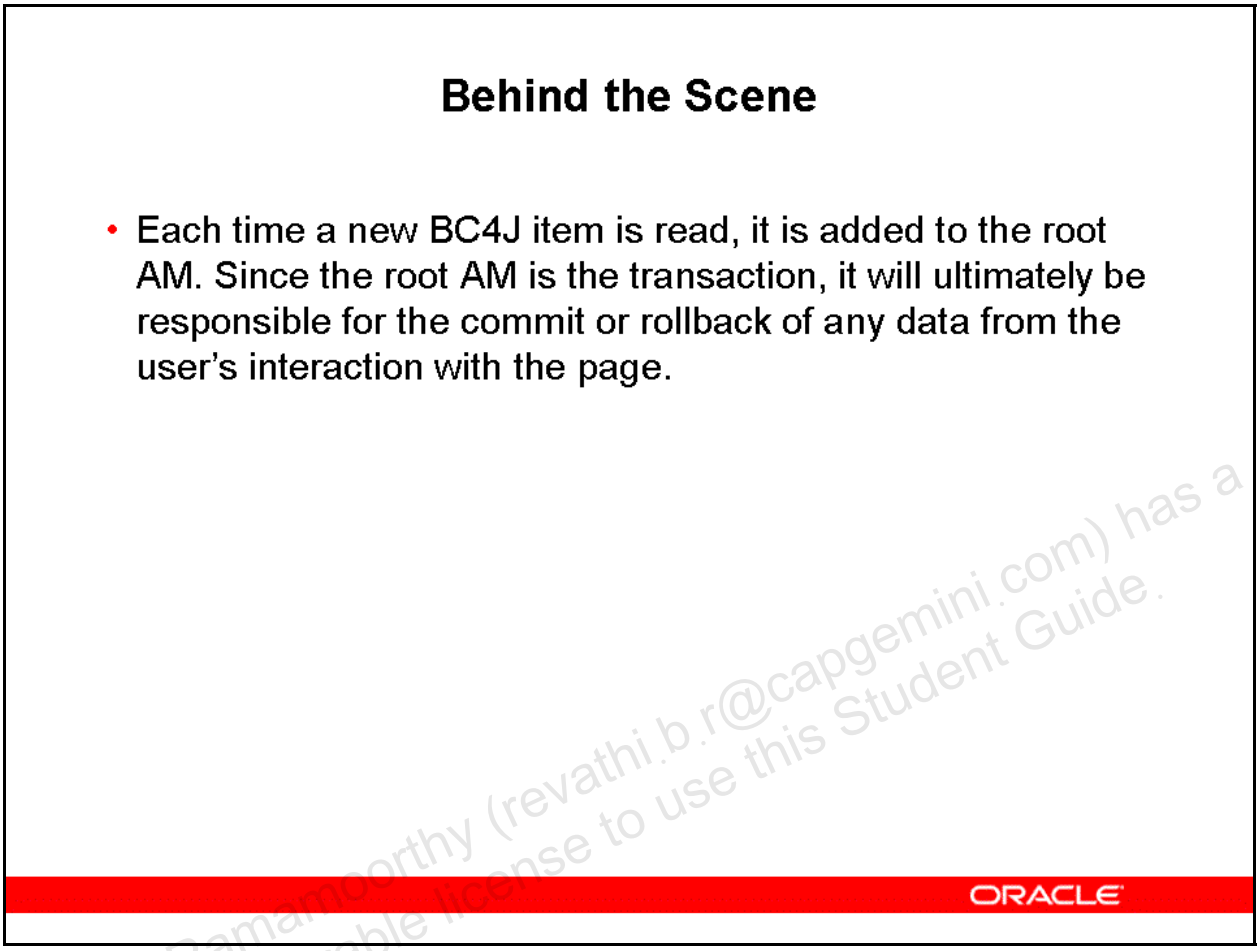
The EOs are stored in $JAVA_TOP. The EO related files are:

<EOName>EO.xml (this is the XML-based definition of the EO. While it is XML-based, it is not stored in the MDS.)

<EOName>EOImpl.class (this is the Java-based programmatic capability of the EO. The XML-based capabilities are limited. In most cases, the needed capabilities of the EO have to be programmed in Java to accomplish their task. Specifically, any data INSERTs, UPDATEs, DELETEs, or unique business logic validations (individual fields or multi-field) must be in this code.)

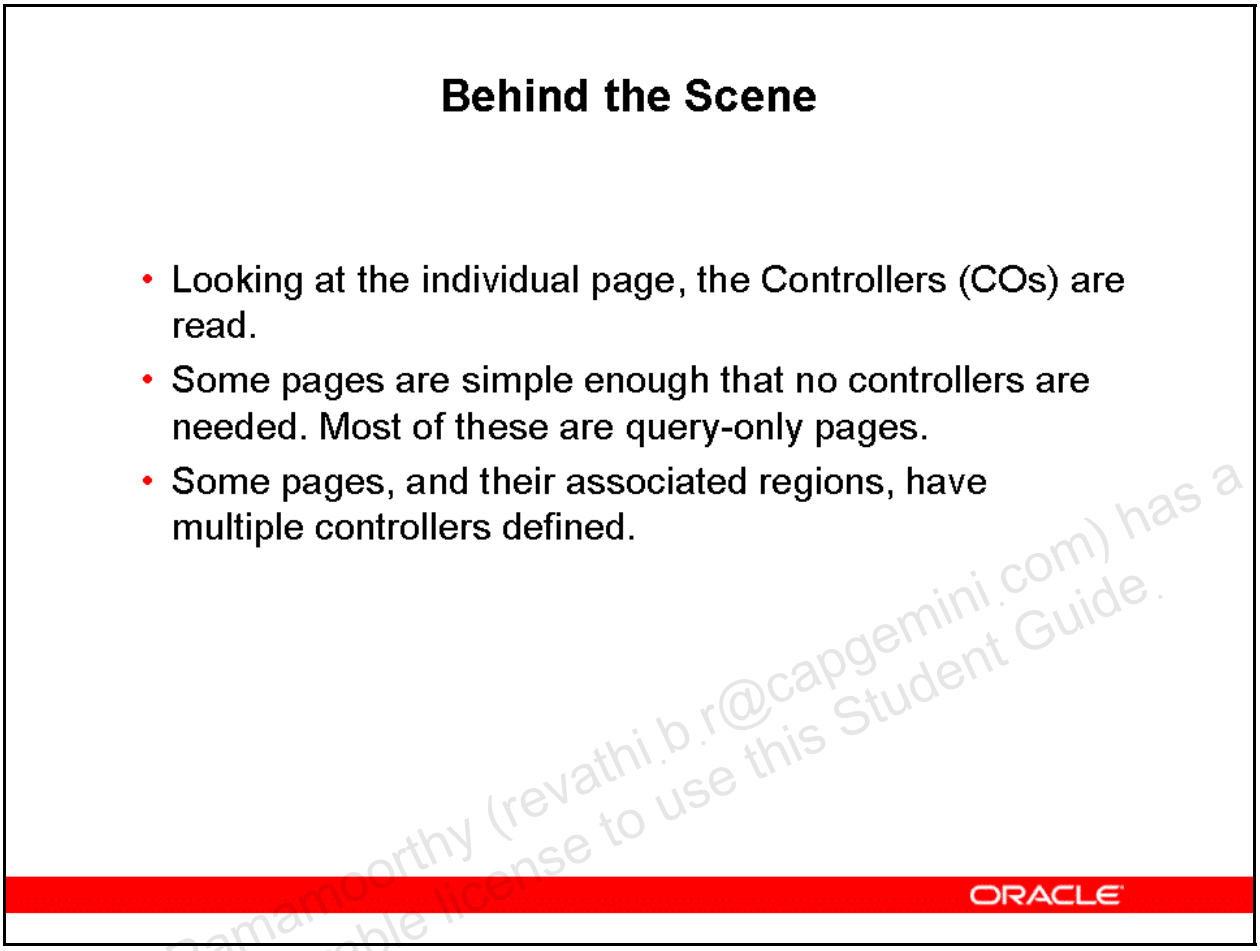
There may be multiple EOs, each with their set of files. There are also entity-level BC4J objects that may be referenced called Association Objects (AOs). AOs are the means by which EOs can be joined. There are also objects like Entity Experts, Validation AMs (VAMs), and Validation VOs (VVOs) to name some others.

Behind the Scene



There is a much more in-depth discussion on the details of page processing later in the course. Additionally, a detailed discussion of State Management (i.e., the overall management of the transaction) also occurs later.

Behind the Scene



The COs are, like the AM, stored in $JAVA_TOP. The CO related file is:

<COName>CO.class (this is the Java-based programmatic capability of the CO. For controllers, there is no XML-based definition. Any UI-capabilities beyond the things accomplishable by the Pages themselves will require the CO to accomplish the task.

The most common such tasks are:

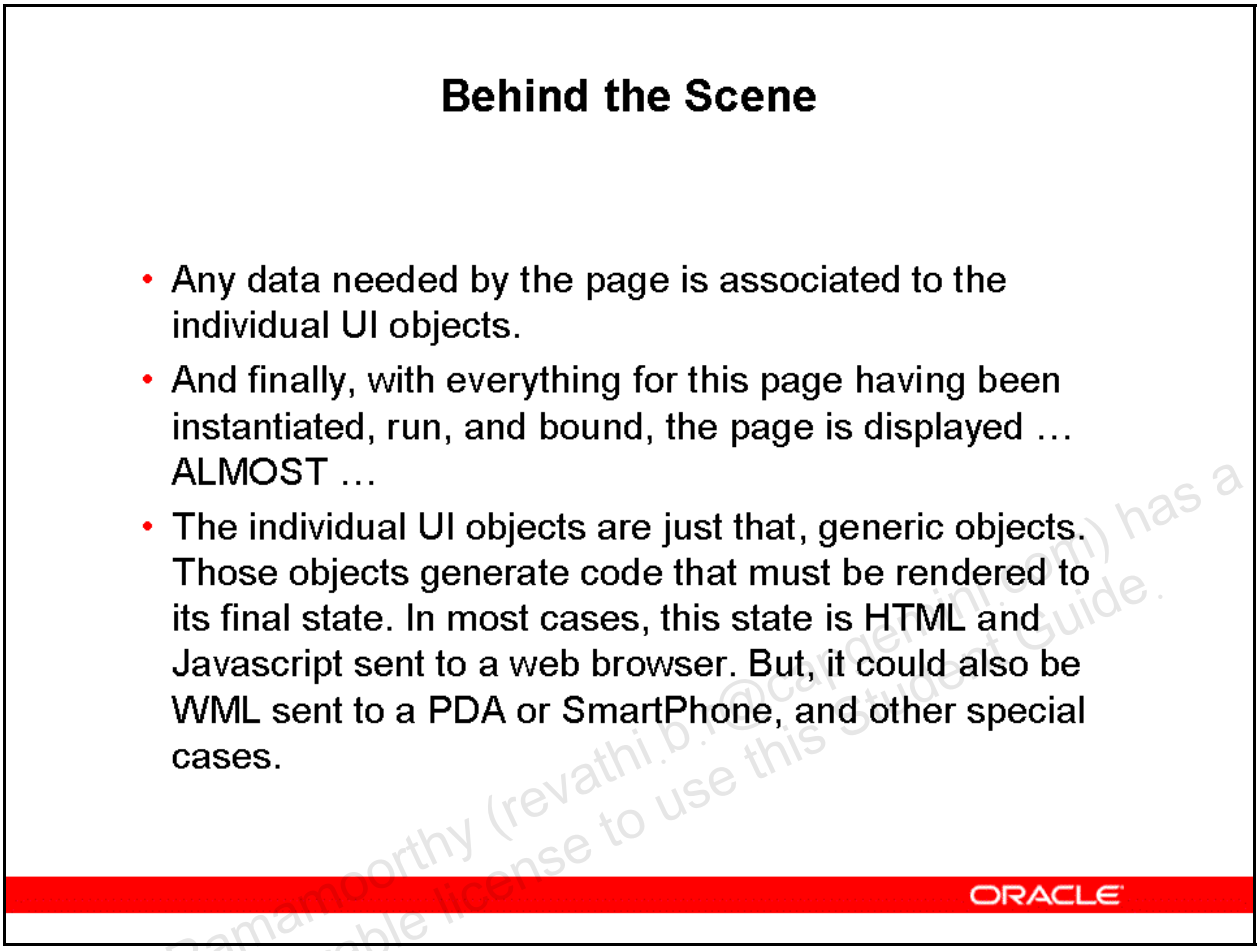
Generation of dynamic UI items (like dynamic window-titles).

Initiating data queries.

Responses to button-clicks and icon-clicks by the user.

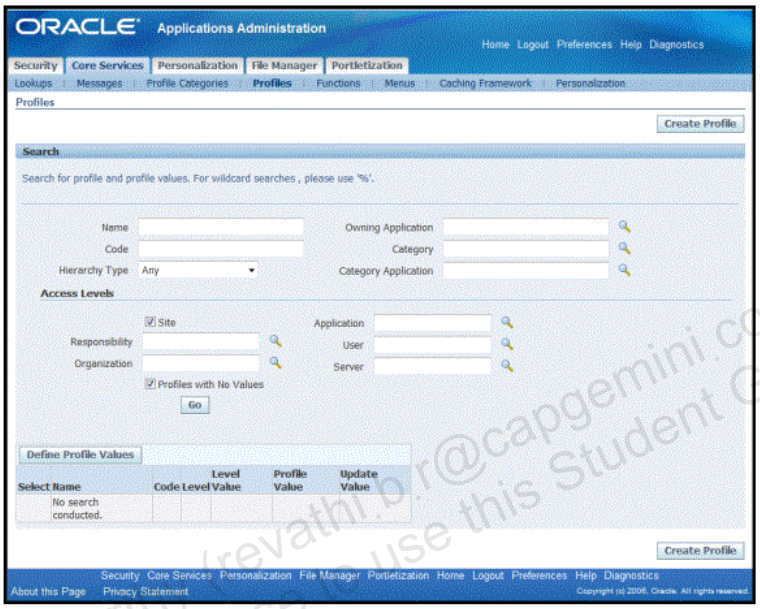
Note: There are two methods in the CO. One method is run when the page is loading (processRequest()). The other method is run when the page is responding to an event (processFormRequest()).

There may be multiple COs, each with a class file. While discussed in this course, COs are not extensible objects within OA Framework pages. Any deeper discussion of COs is beyond the scope of this course.



Behind the Scene

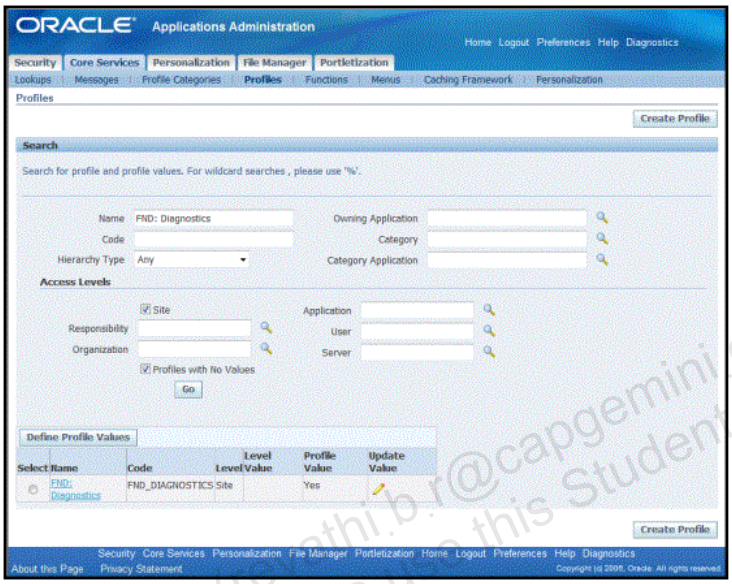
- The following is a page without data associated with it:



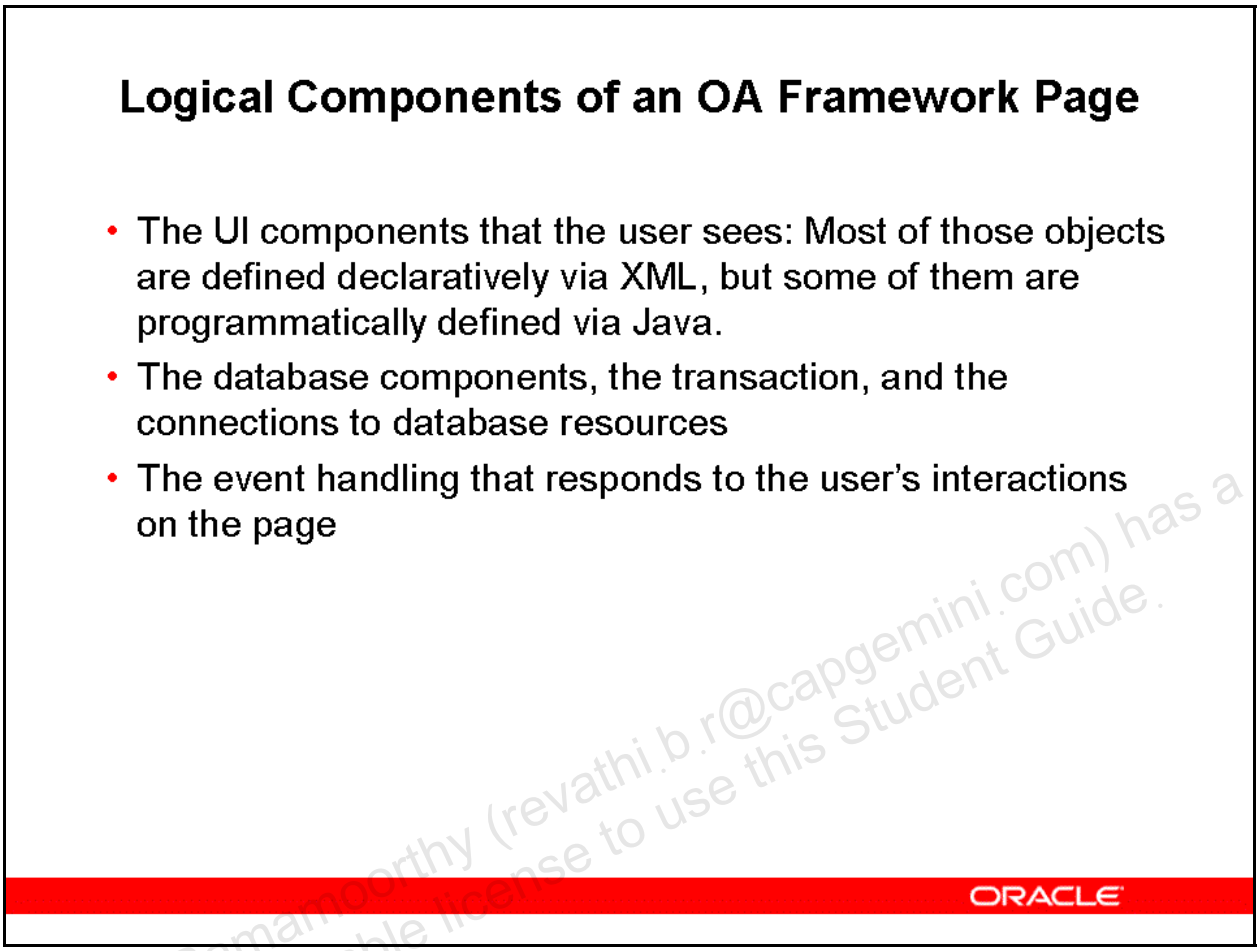
ORACLE

Behind the Scene

- The following is a page with data associated with it :



Logical Components of an OA Framework Page



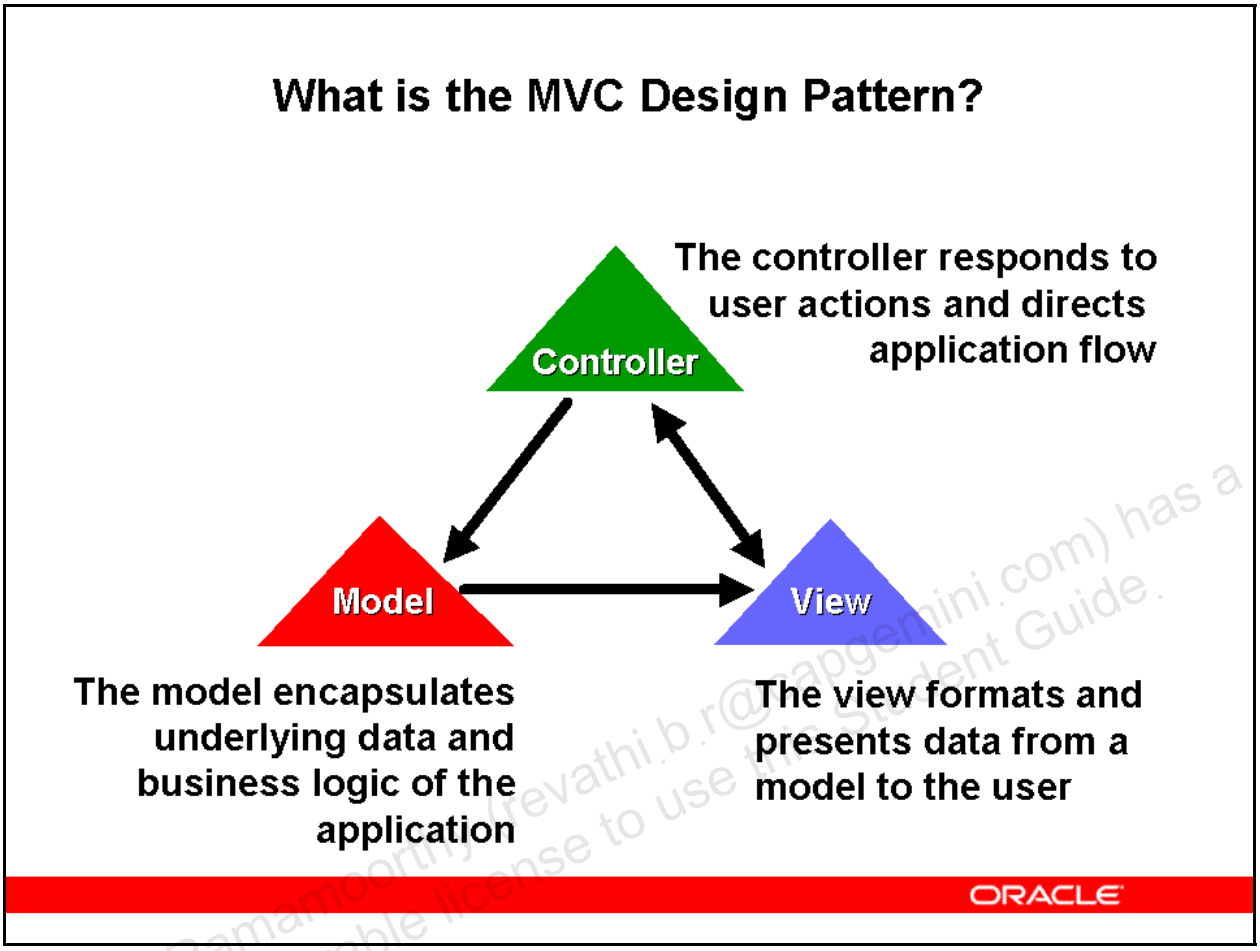
Event Handling:

In traditional client/server applications, you have the option of handling events ranging in granularity from very low-level mouse movements to field, region and finally, window-level scope. Furthermore, when you communicate from the client to the server, you can send a single value to be validated back to the server while expecting a single validation result back. You can then modify the user interface accordingly, which allows for a highly interactive experience for the user.

In a web application, you essentially handle "page-level" events (unless you are using JavaScript extensively to create a more interactive experience, and since the OA Framework Coding Standards and Oracle Browser Look and Feel (BLAF) UI Guidelines prohibit this, we will not consider it here). In this case, as users navigate from field to field and enter data, there are no events for you as a developer to handle.

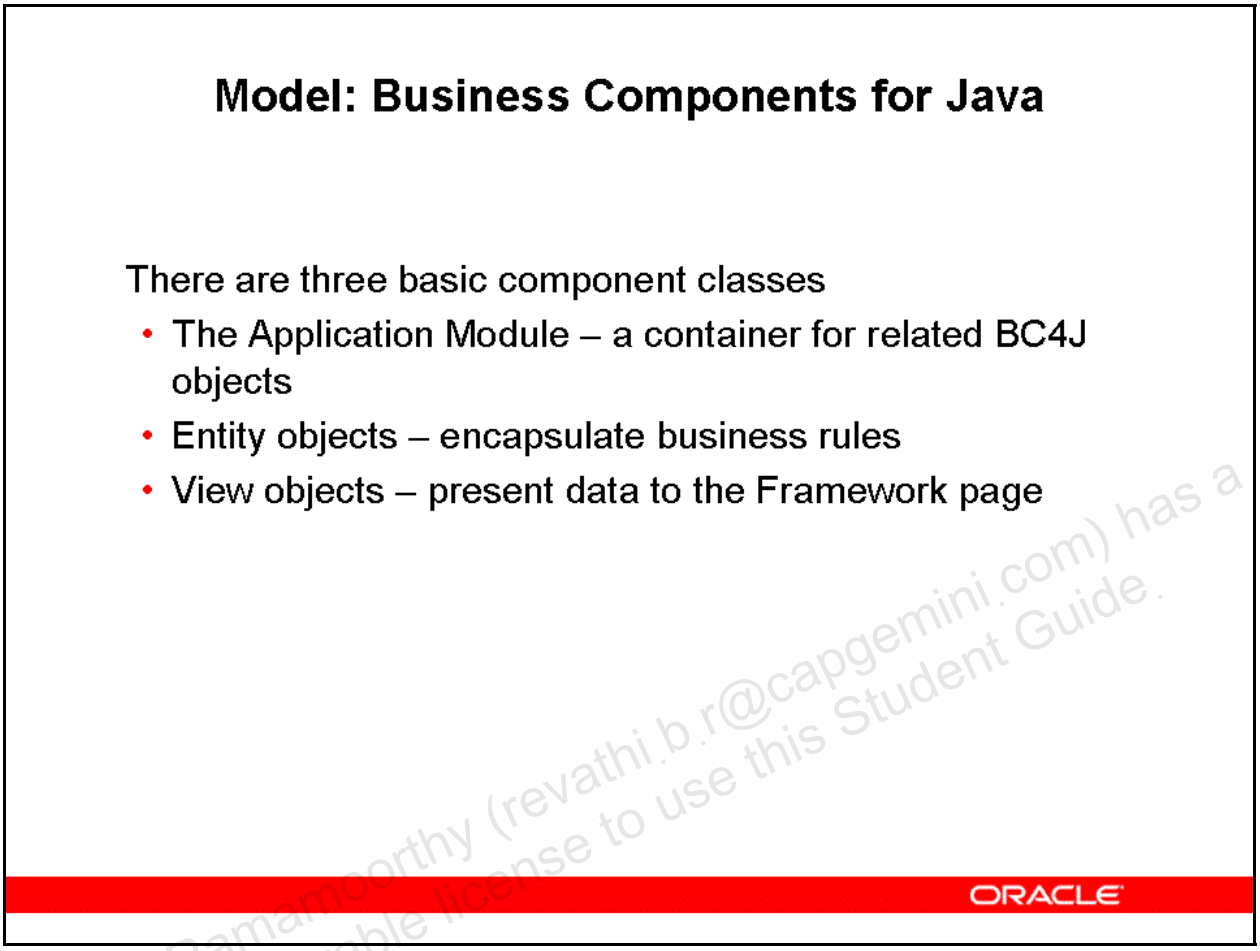
When the browser finally sends a request as described above, all the page data is sent in that single communication -- including any user-entered values and information about what actions the user wants to perform. The developer reads request values to determine what happened (if the user pressed a button, which one was it?), does whatever work is required by the selected action, and then transmits a new HTML page back to the browser.

What is the MVC Design Pattern?



The term "view“ is another overloaded term at Oracle and has several meanings for Oracle-based applications (such as database view). In the case of the Model-View-Controller architecture, it helps to think of the view as the "end user's view of the data".

Model: Business Components for Java



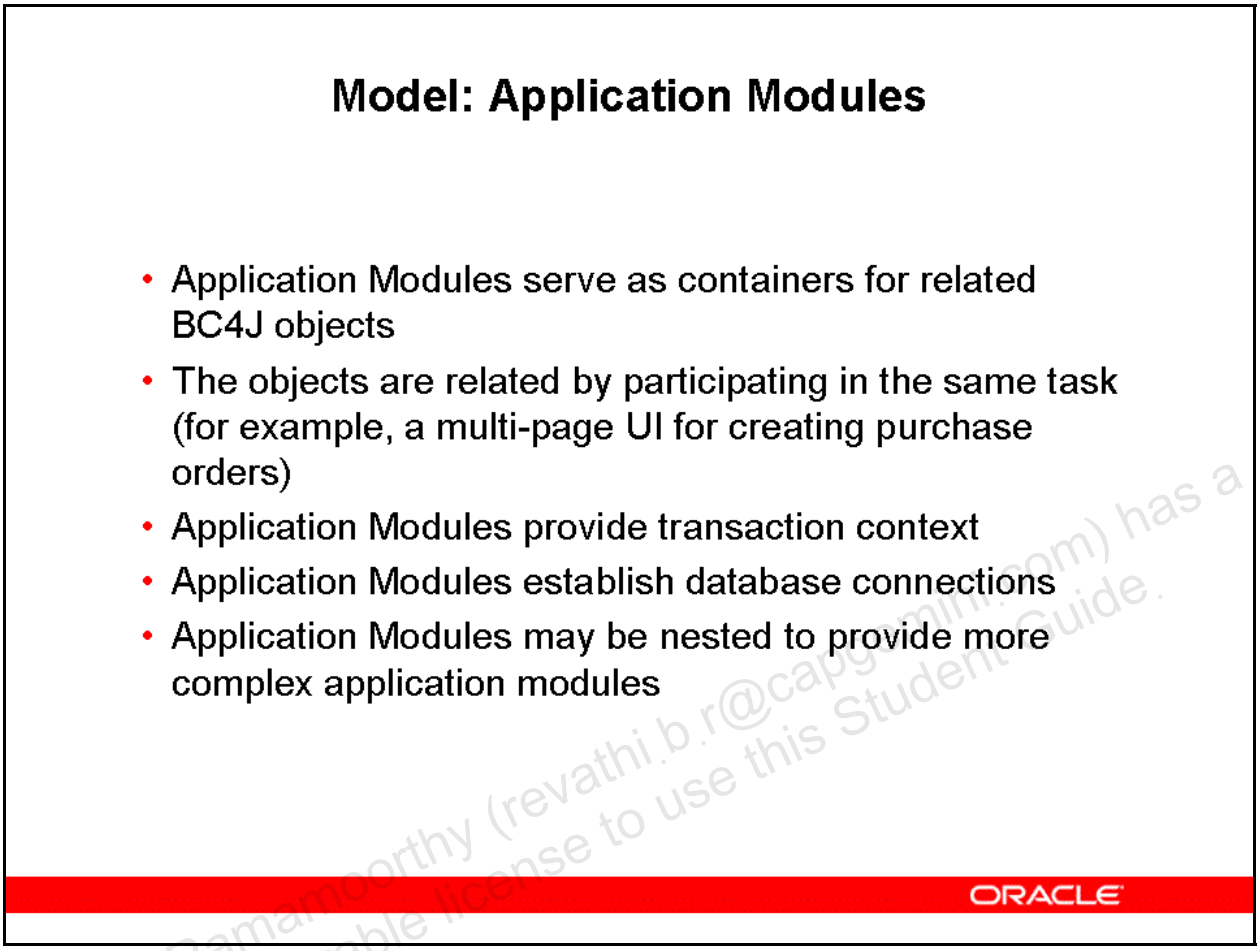
In general, the following definitions are useful as quick references to BC4J components.

The AM is the transaction container.

EOs are the tables and the business rules.

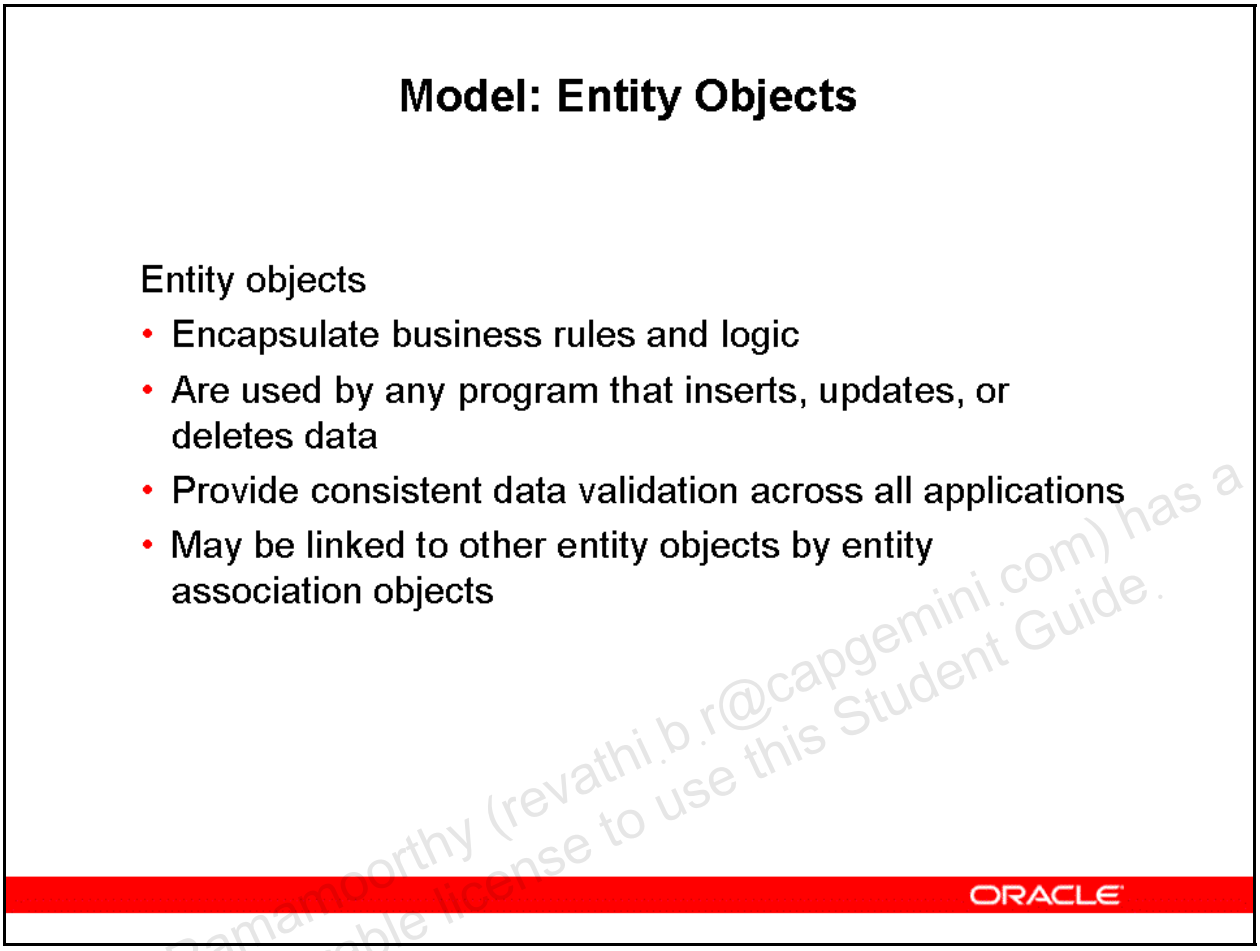
VOs are the queries / SELECT statements.

Model: Application Modules



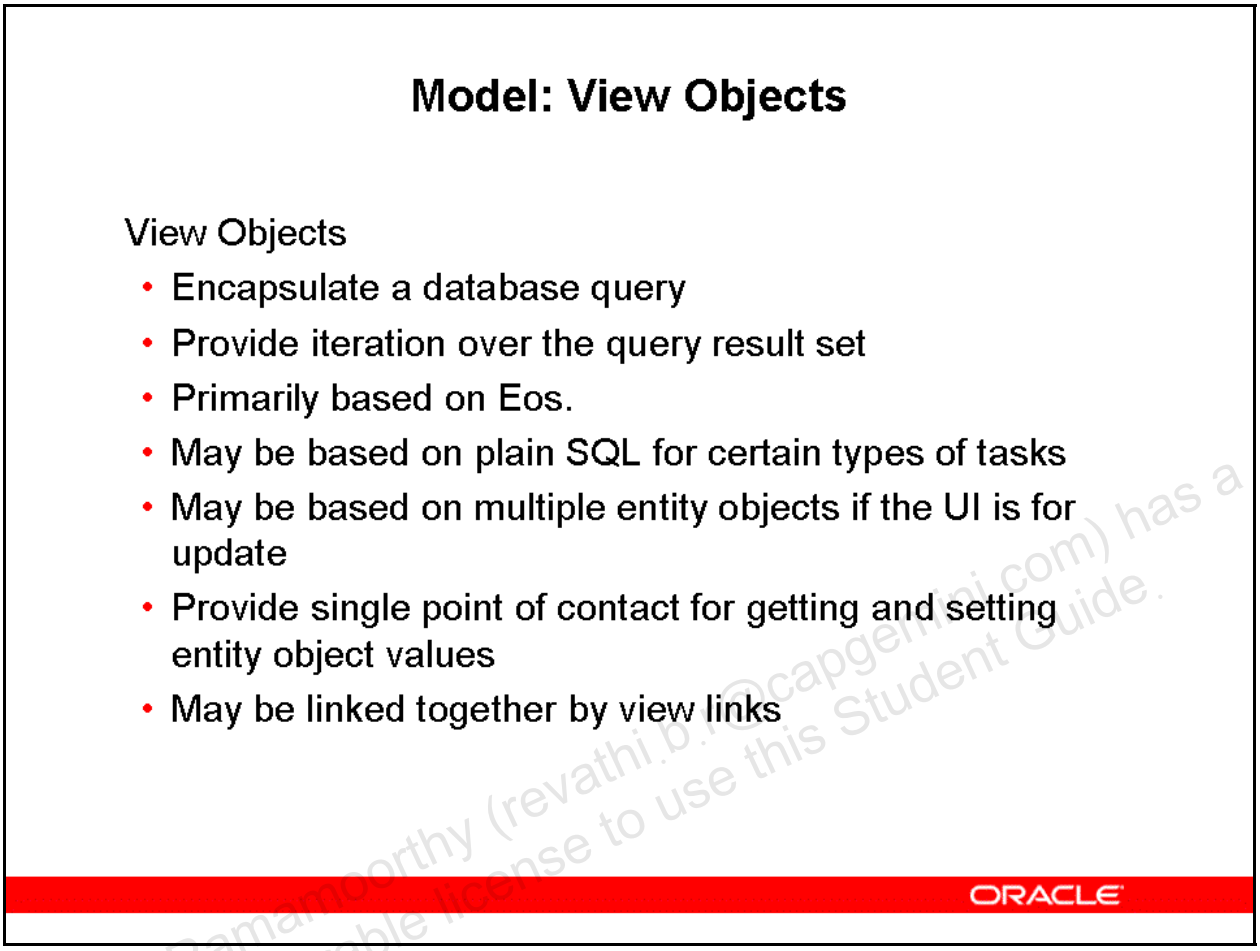
An AM must explicitly contain a VO. But, EOs and other objects are indirectly contained in the AM via their association to the VO and the objects the VO needs.

Model: Entity Objects



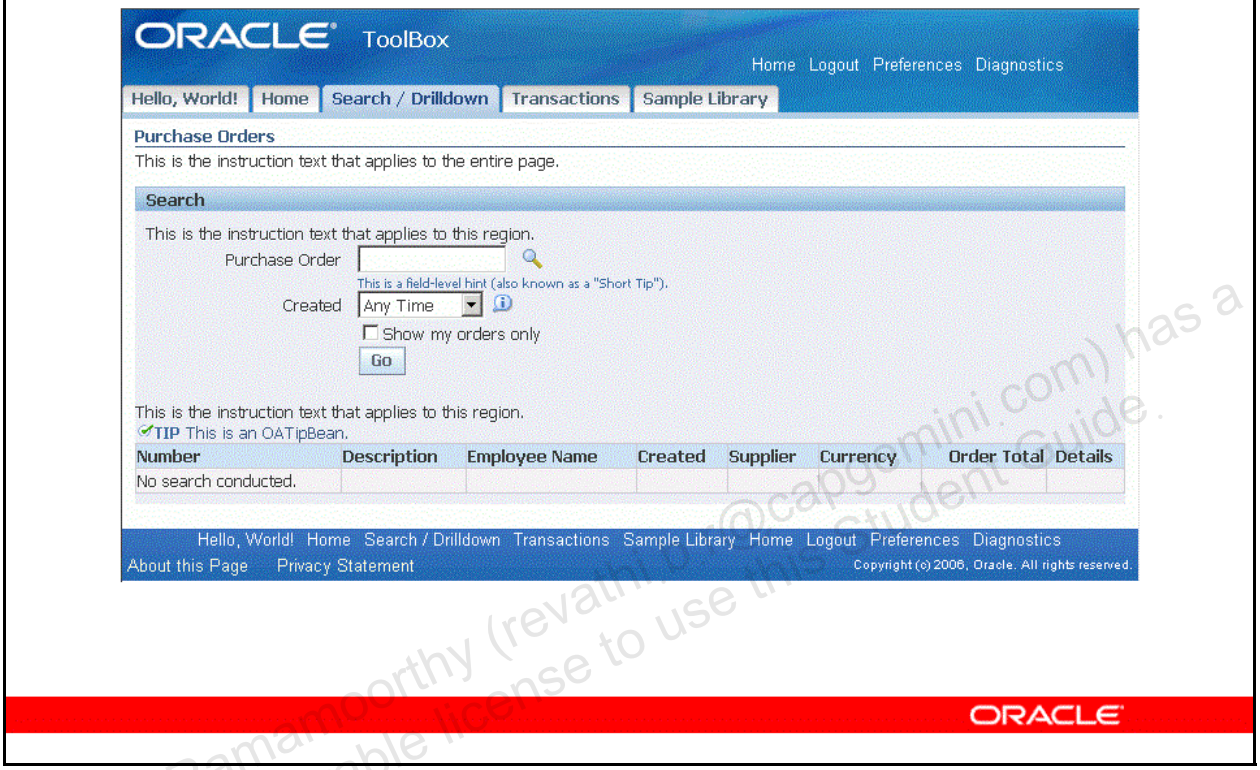
There should be one EO per table, and that EO should be the means by which all business rules are implemented on that table.

Model: View Objects

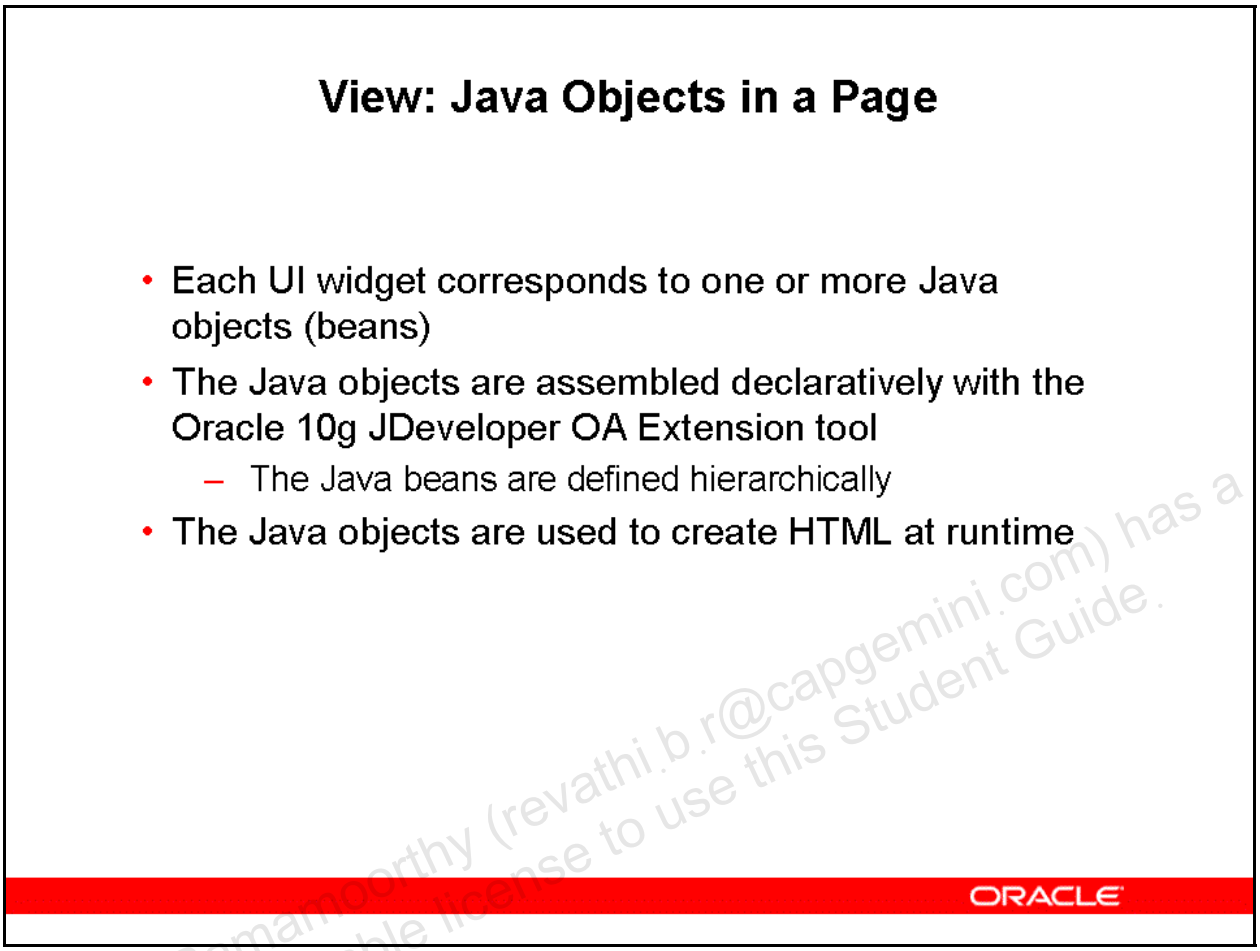


Here is a quick rule to remember. While code reuse for most objects is a desirable trait. It is far more important for VOs to run quickly and minimize their memory impact to the middle tier. So, in general, you should develop specific VOs for specific needs, and avoid VO reuse.

View: OA Framework-Based Page

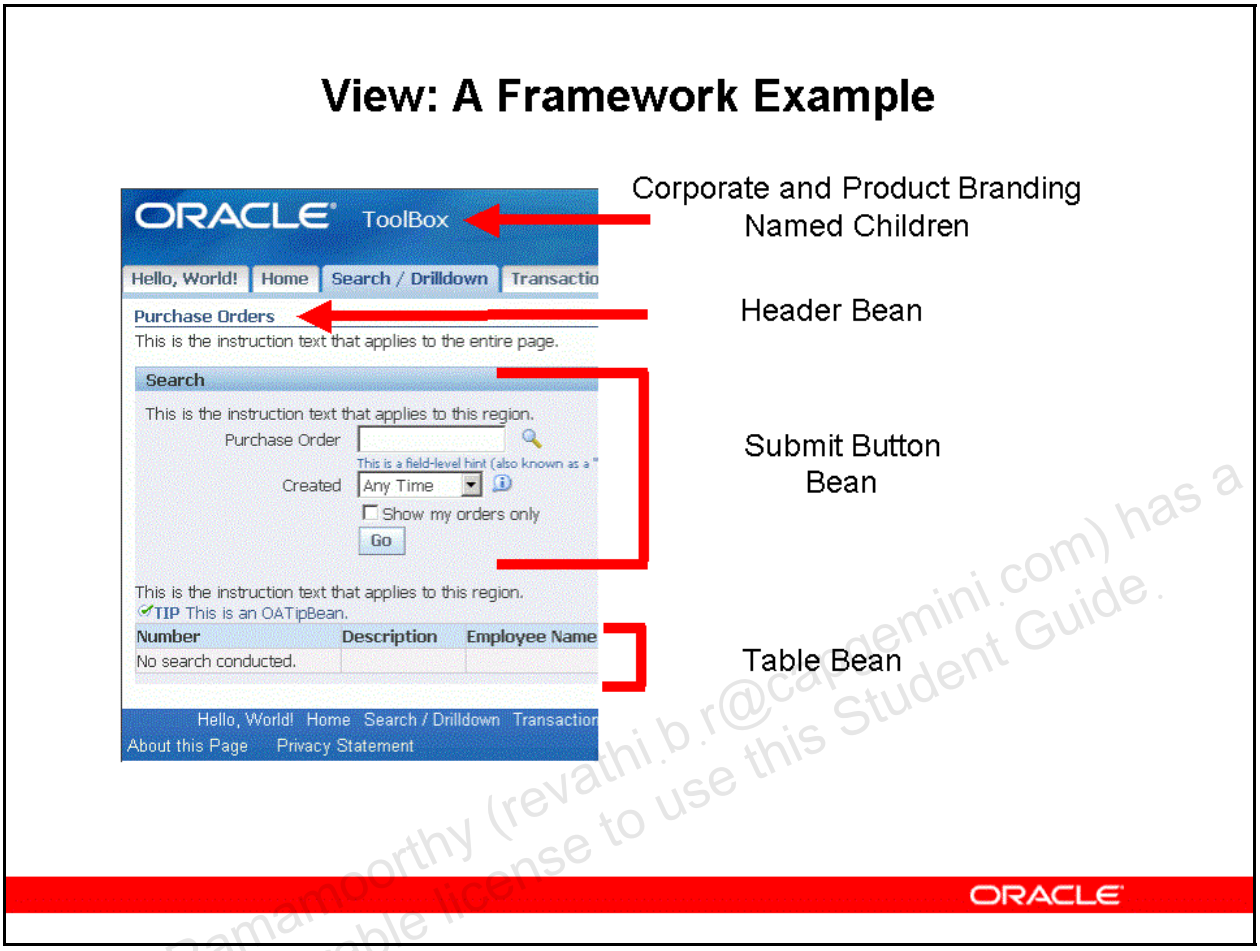


View: Java Objects in a Page



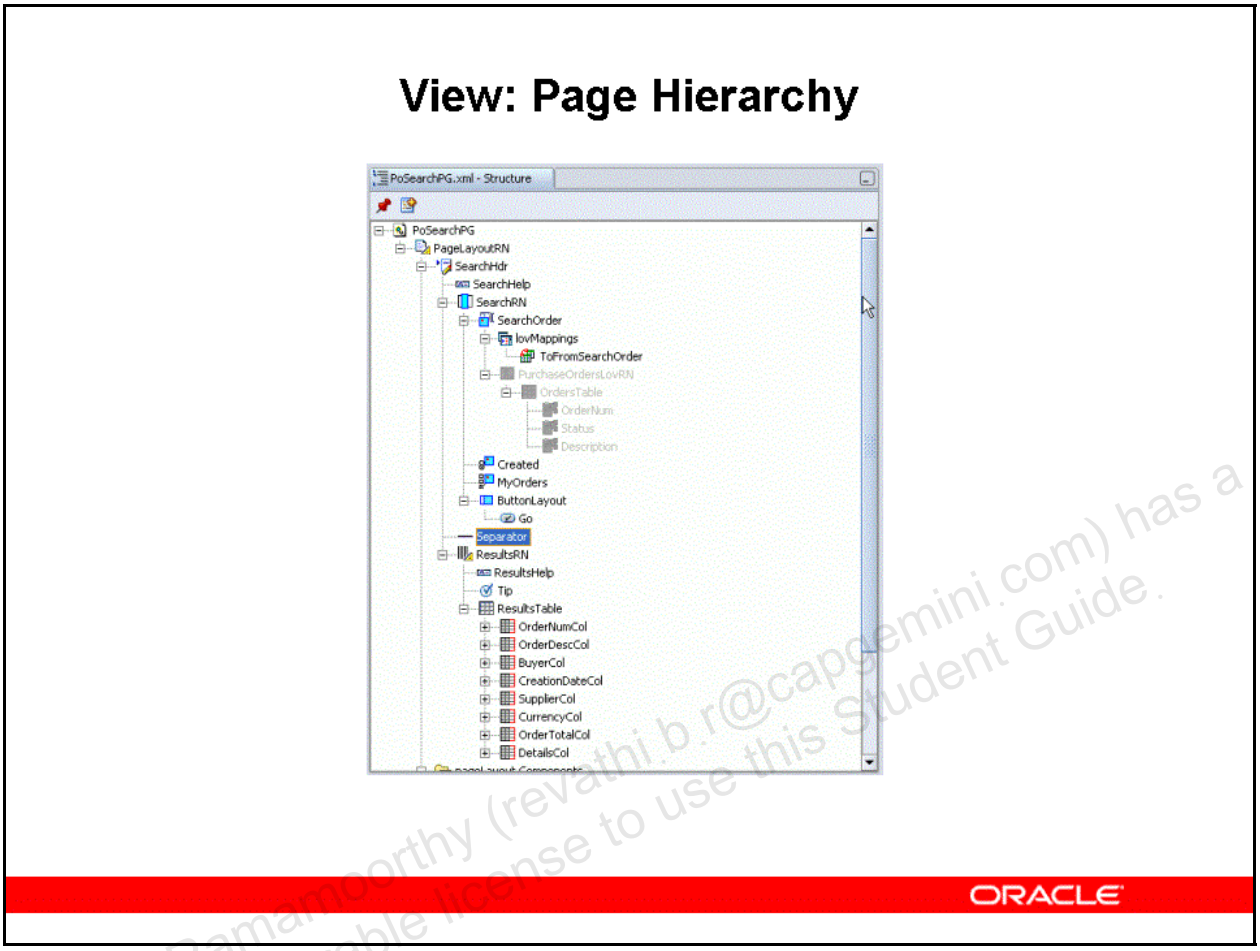
By referring to UI objects, instead of the code to create those UI objects, the UI objects can change in their underlying implementation without requiring changes to multiple pages of OA Framework code.

View: A Framework Example



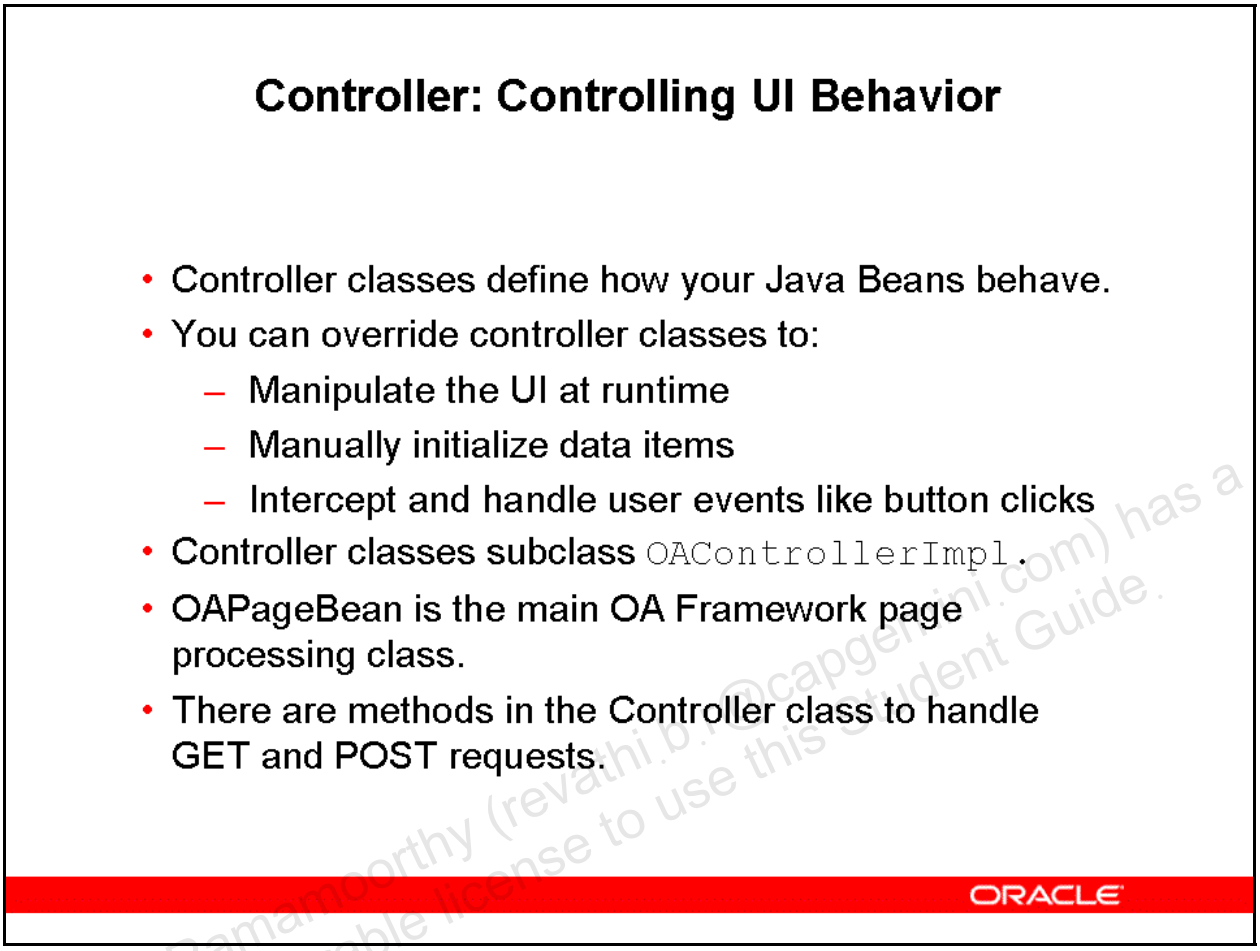
These are examples of some of the JavaBeans that construct the UI for this page.

View: Page Hierarchy



The screen capture above, was taken from within JDeveloper. It shows the structure of a page. OA Framework pages use a hierarchical layout. The order in which an object appears in the hierarchy is the order in which it appears on the page. Sometimes, you may have to look at the page XML to verify that JDeveloper created the page in the correct order. You can't edit the page XML in JDeveloper and you should not edit it outside either.

Controller: Controlling UI Behavior

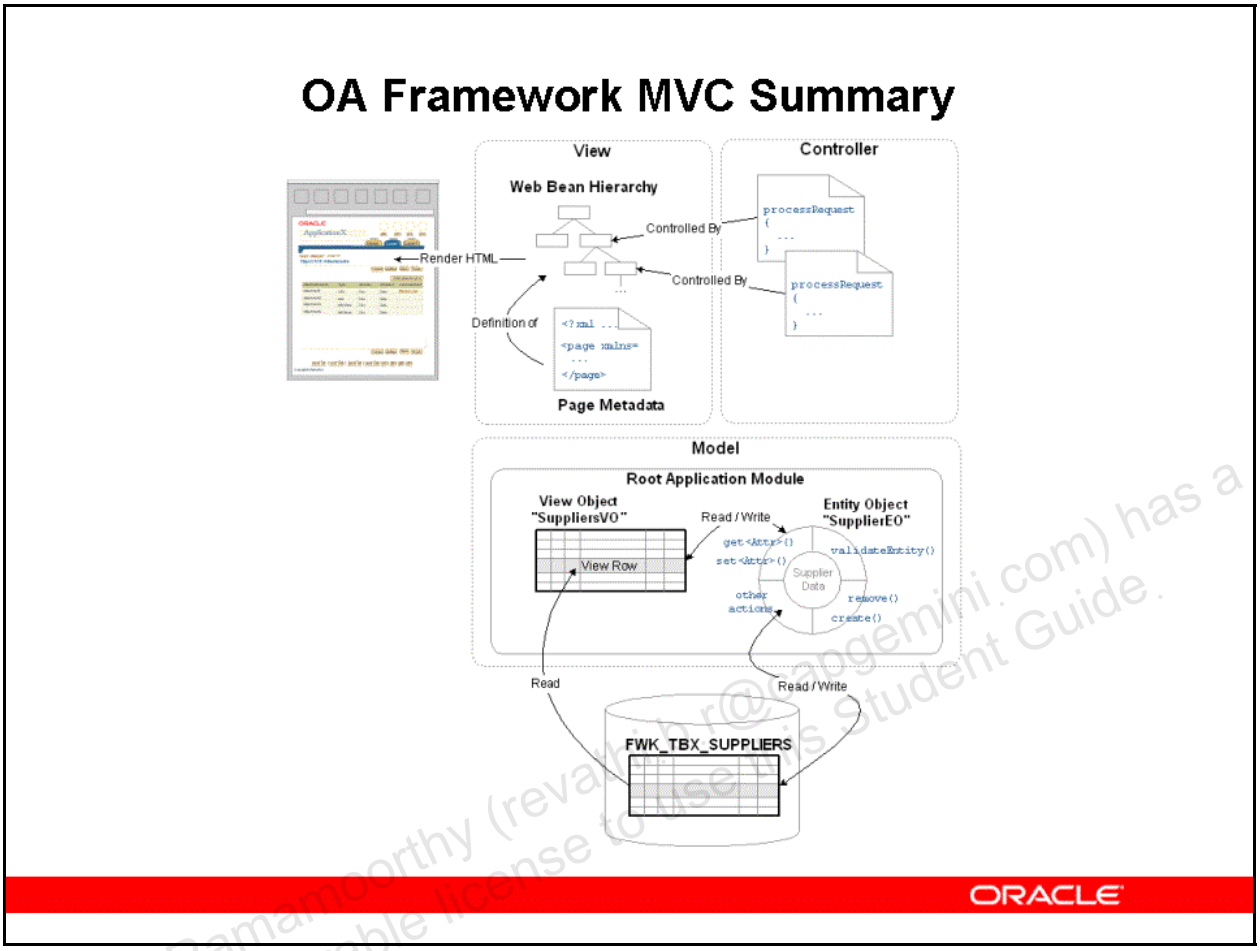


In general, a controller does two things.

Responds to the drawing of the page (GET).

Responds to events (like button clicks) on the page (POST).

OA Framework MVC Summary



At the browser level, an OA Framework page, like any other web page, renders as standard HTML.

In the middle tier, however, this page is implemented in memory as a hierarchy of Java beans -- very much like a classical Java client UI. Each UI widget, such as buttons, a table, the tabs, the application branding image and so on, that renders in the page corresponds to one or more web beans in the hierarchy.

When the browser issues a request for a new page, OA Framework reads the page's declarative metadata definition to create the web bean hierarchy. For each bean with an associated UI controller, OA Framework calls code that you write to initialize the page. When page processing completes, OA Framework hands the web bean hierarchy to the UIX framework so it can generate and send HTML to the browser.

When the browser issues a form submit (if, for example, the user selects a search region's Go button), OA Framework recreates the web bean hierarchy if necessary (the hierarchy is cached between requests, and typically needs to be recreated only in exceptional cases that we'll discuss in detail later), and then calls any event handling code that you've written for the page beans. When page processing completes, the page HTML is generated again and sent to the browser.

Summary

Summary

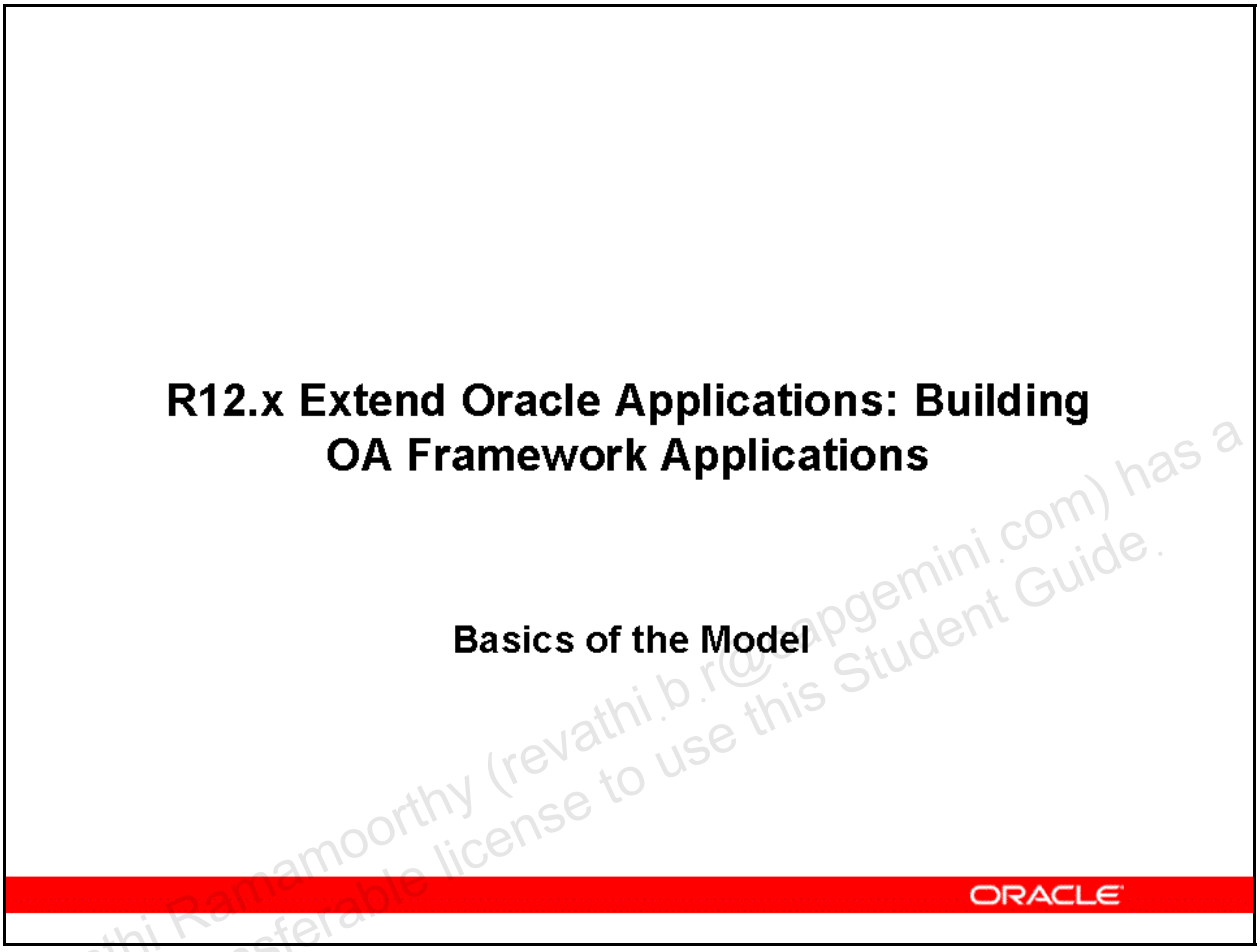
In this lesson, you should have learned how to:

- Discuss the key concepts of an OA Framework page.
- Discuss the runtime process of an OA Framework page.
- Discuss how the MVC design pattern is used in OA Framework pages.

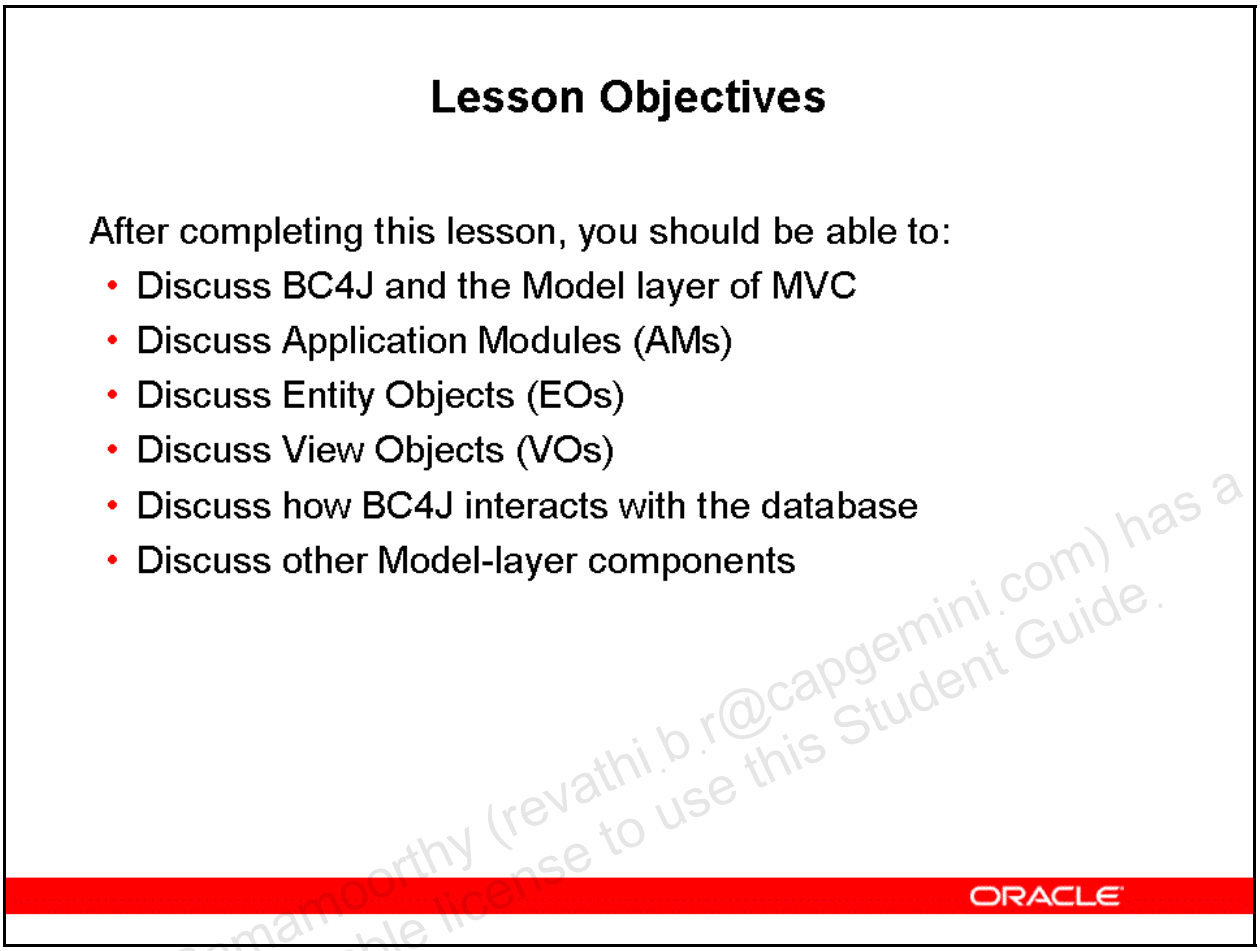
ORACLE

Basics of the Model

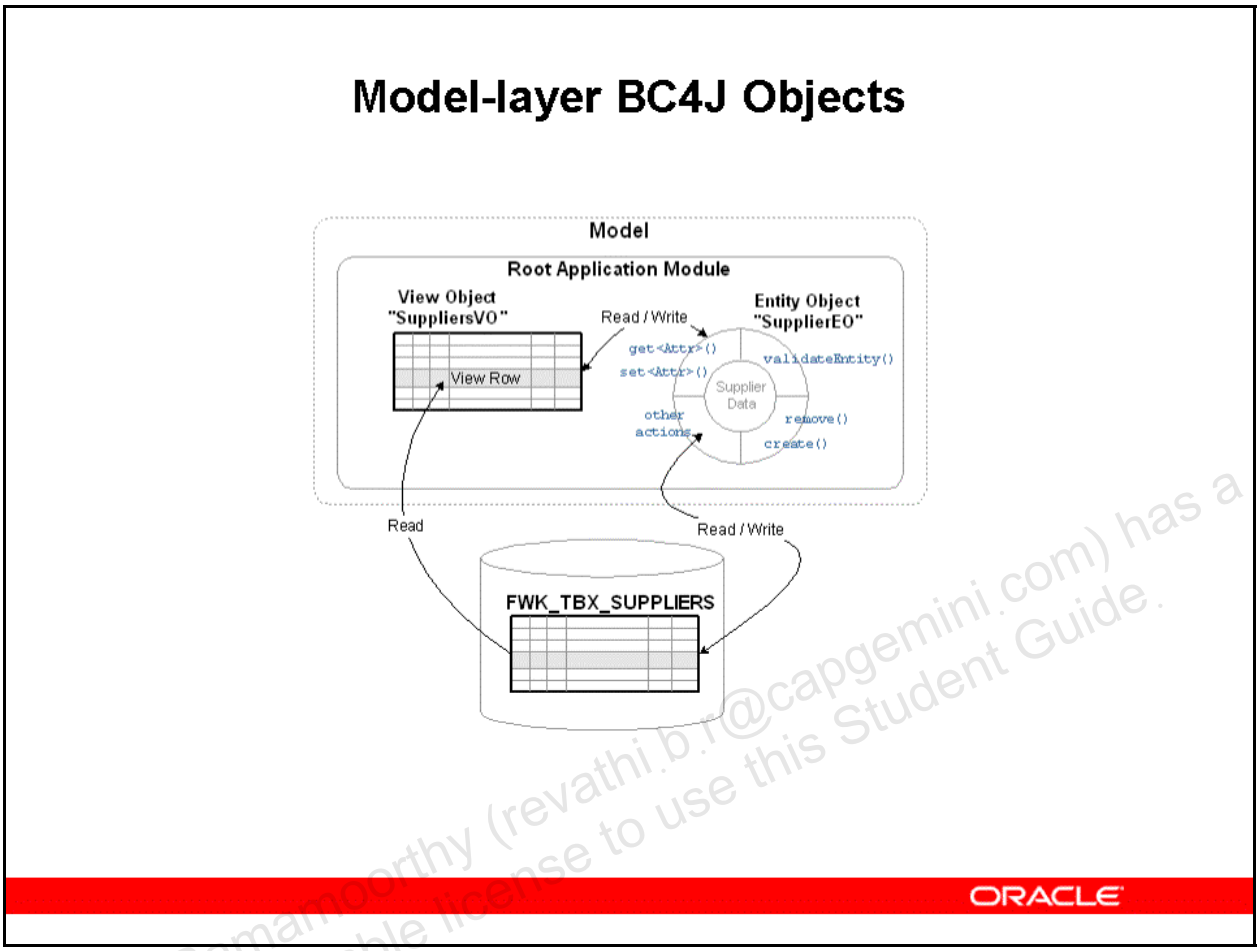
Chapter 3



Lesson Objectives



Model-layer BC4J Objects

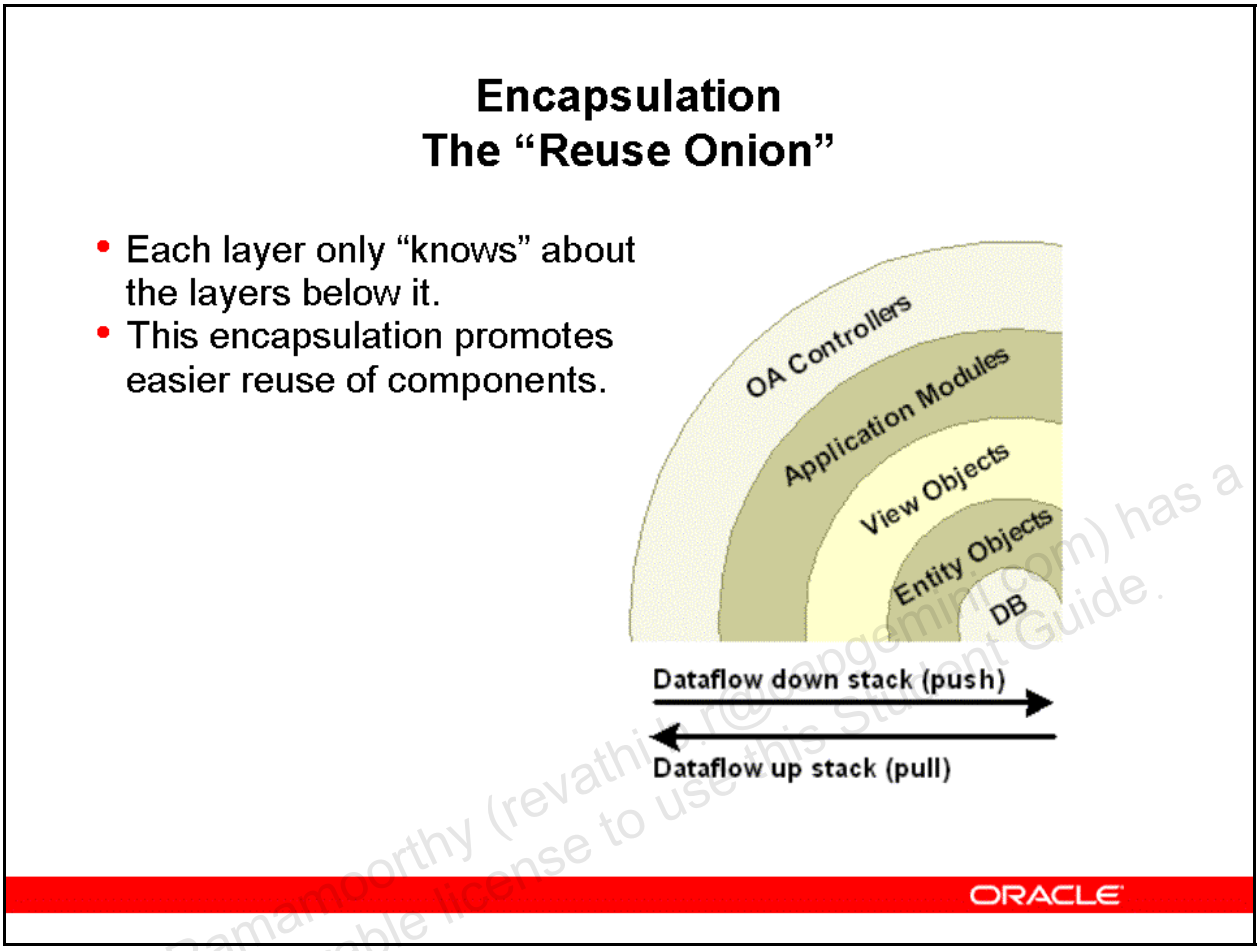


Remember two things:

The Root AM is just the first AM for a page. It contains the transaction and all the connection objects.

The Root AM explicitly contains the VOs. All other objects are contained via their association to the VOs.

Encapsulation The “Reuse Onion”



'Client' / 'Server' Code Separation

Within the Model-View-Controller architecture, OA Framework draws a clear distinction between client and server classes, a distinction that on the surface may seem to conflict with JSP application architecture. A typical JSP application has 3 physical tiers:

The browser (the client where our HTML renders and users interact with the UI)

The web application server (the middle tier where our UI web bean hierarchy is constructed and our application business logic executes)

The database server

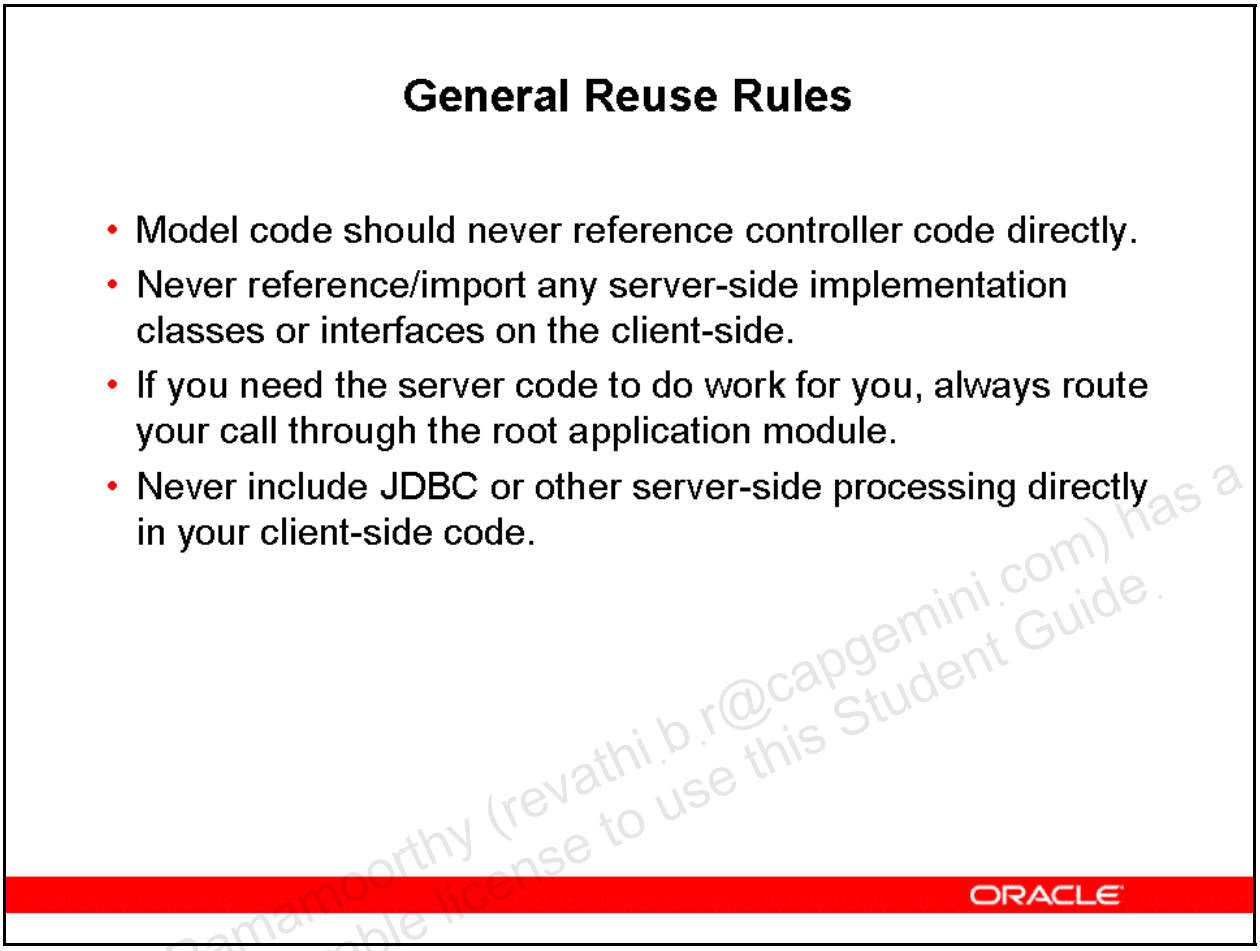
Within the middle tier, OA Framework draws a further distinction between client and server classes as follows:

Client classes (View and Controller code) drive the HTML user interface

- Server classes (Model code) supports any client (not just OA Framework) user interfaces

This distinction is important because it preserves the ability to use server code with different clients.

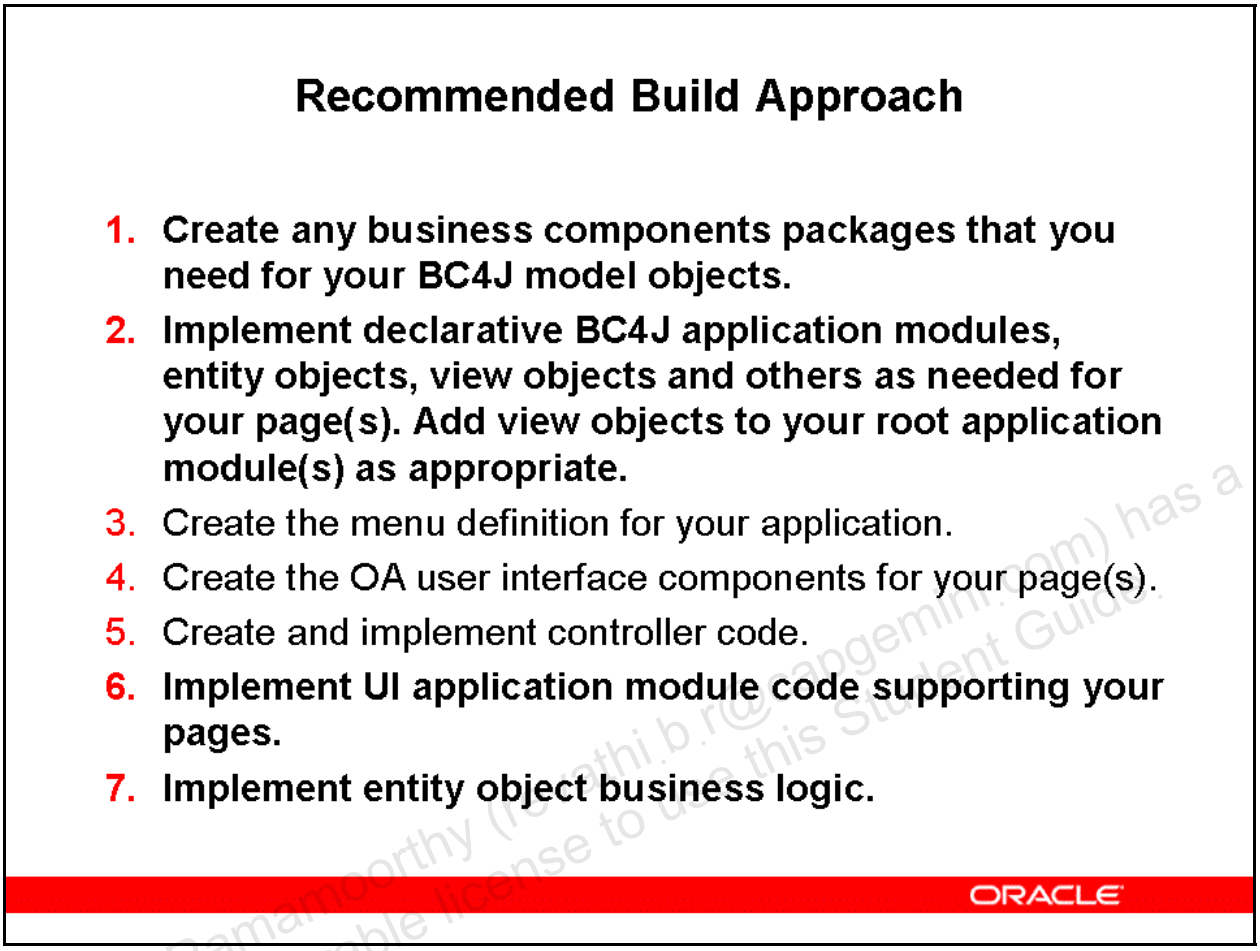
General Reuse Rules



Note about JDBC: It is not recommended or supported to included JDBC calls in OA Framework because of connection pooling. BC4J, through the Root AM, pools its connections to the database, and manages those connections to provide consistent performance on the middle tier. If you call JDBC directly, your code can cause memory and connection leaks, which will ultimately produce bad performance and server crashes.

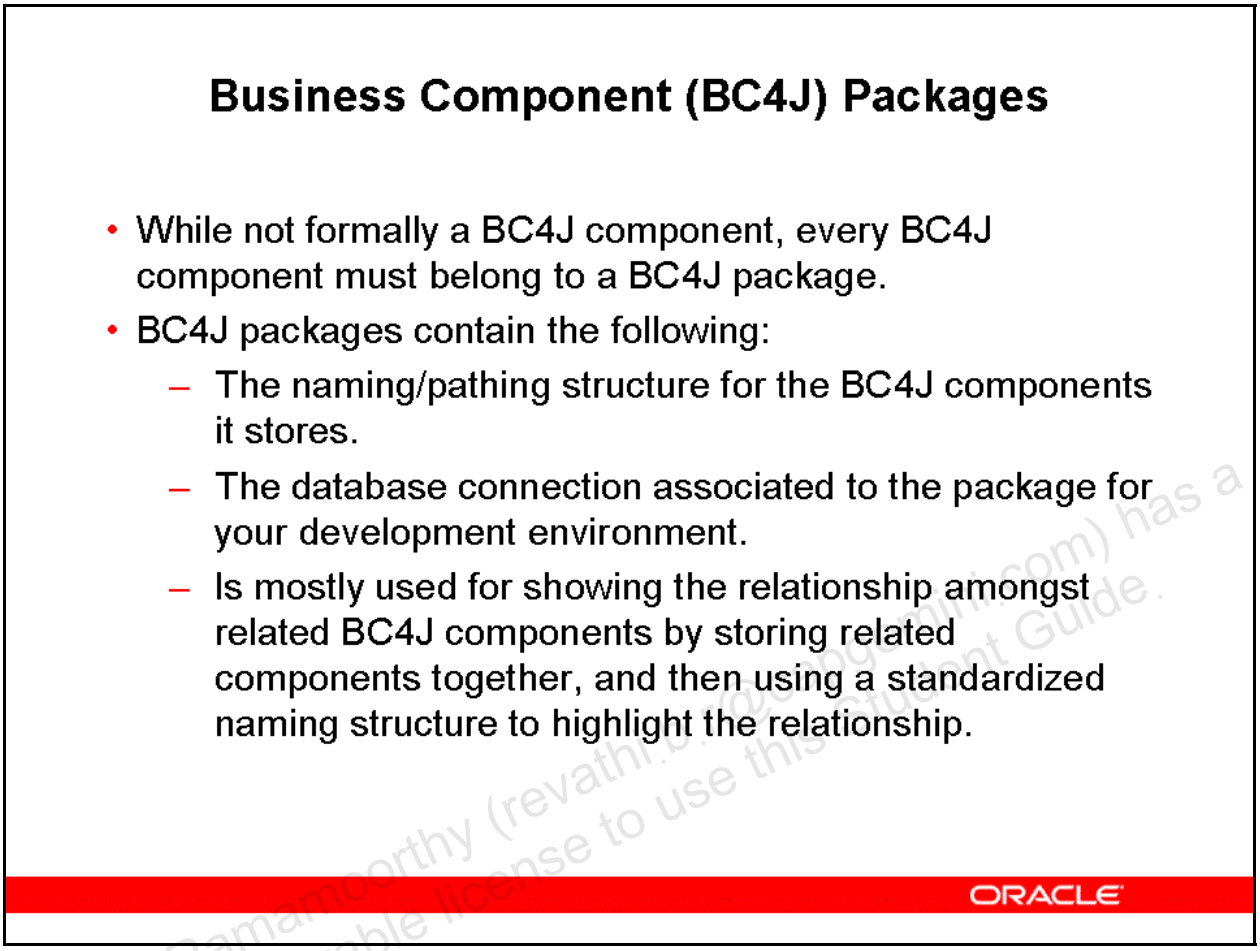
JDBC is the Java Database Connectivity API. It is a J2EE API that provides a way for Java programs to access one or more sources of data.

Recommended Build Approach



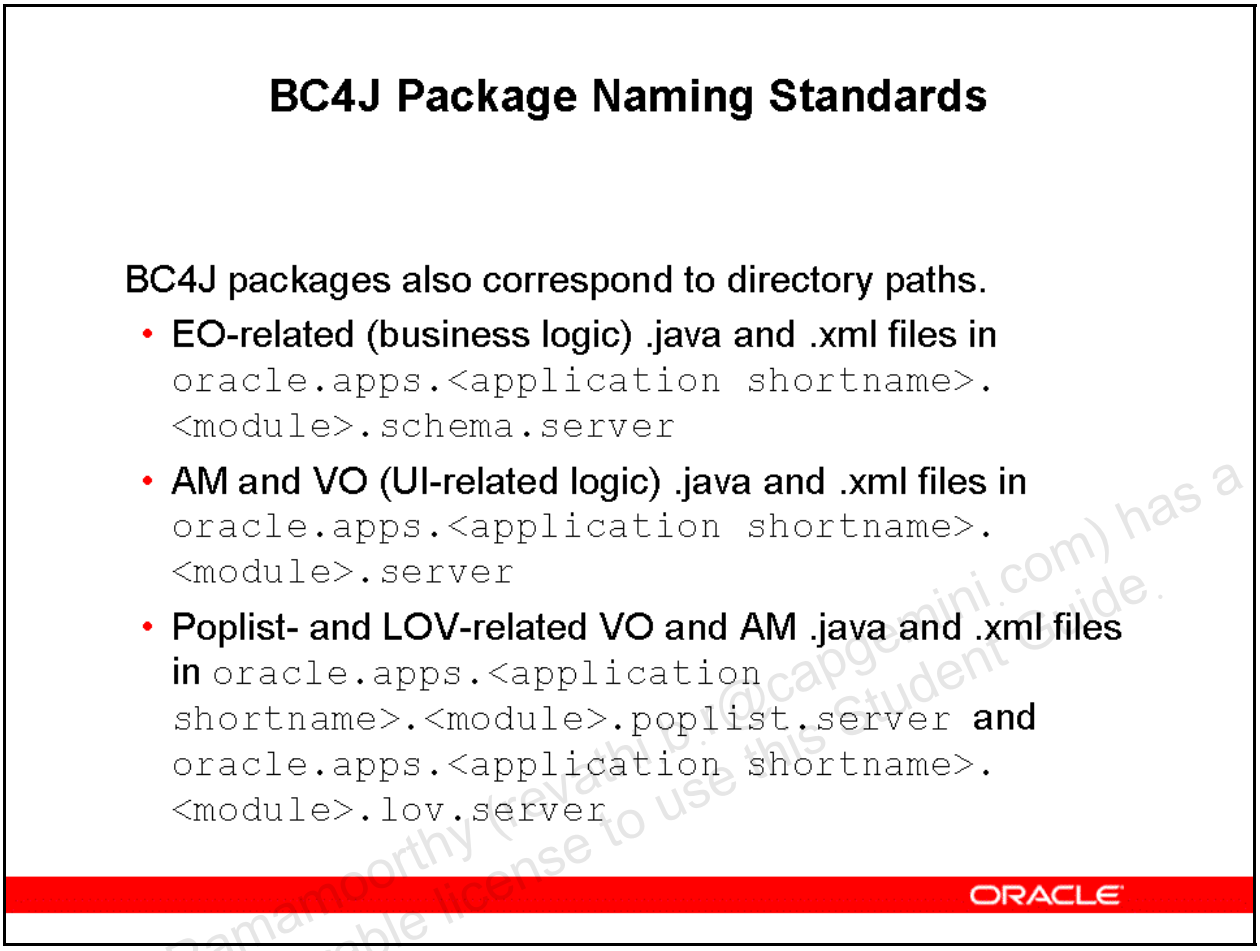
Steps 1, 2, 6, and 7 are the steps that deal directly with Model layer components.

Business Component (BC4J) Packages

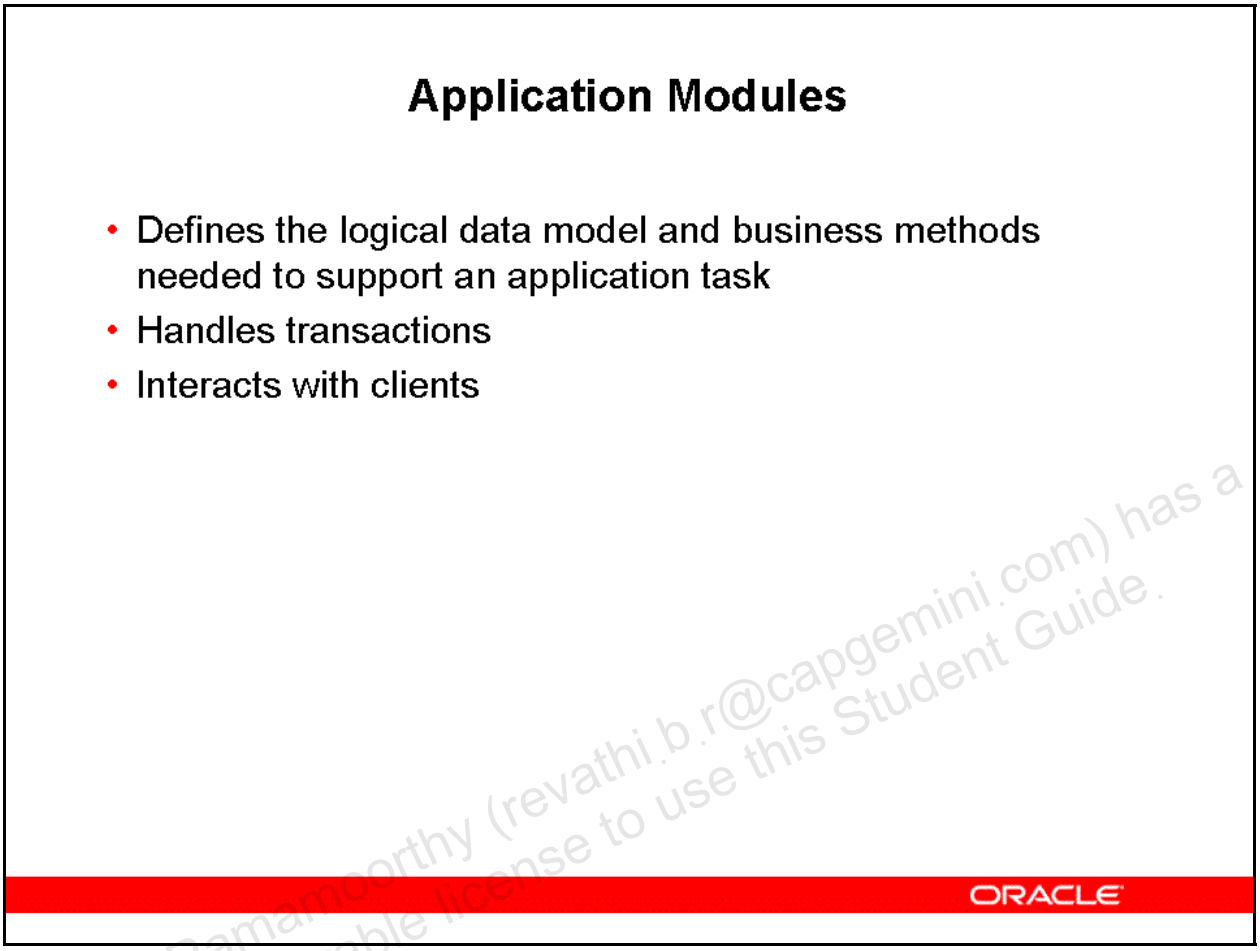


The JDeveloper 10g process changed from the JDeveloper 9i process. In 9i, you explicitly created BC4J packages. In 10g, BC4J packages are create inherently from the BC4J object creation.

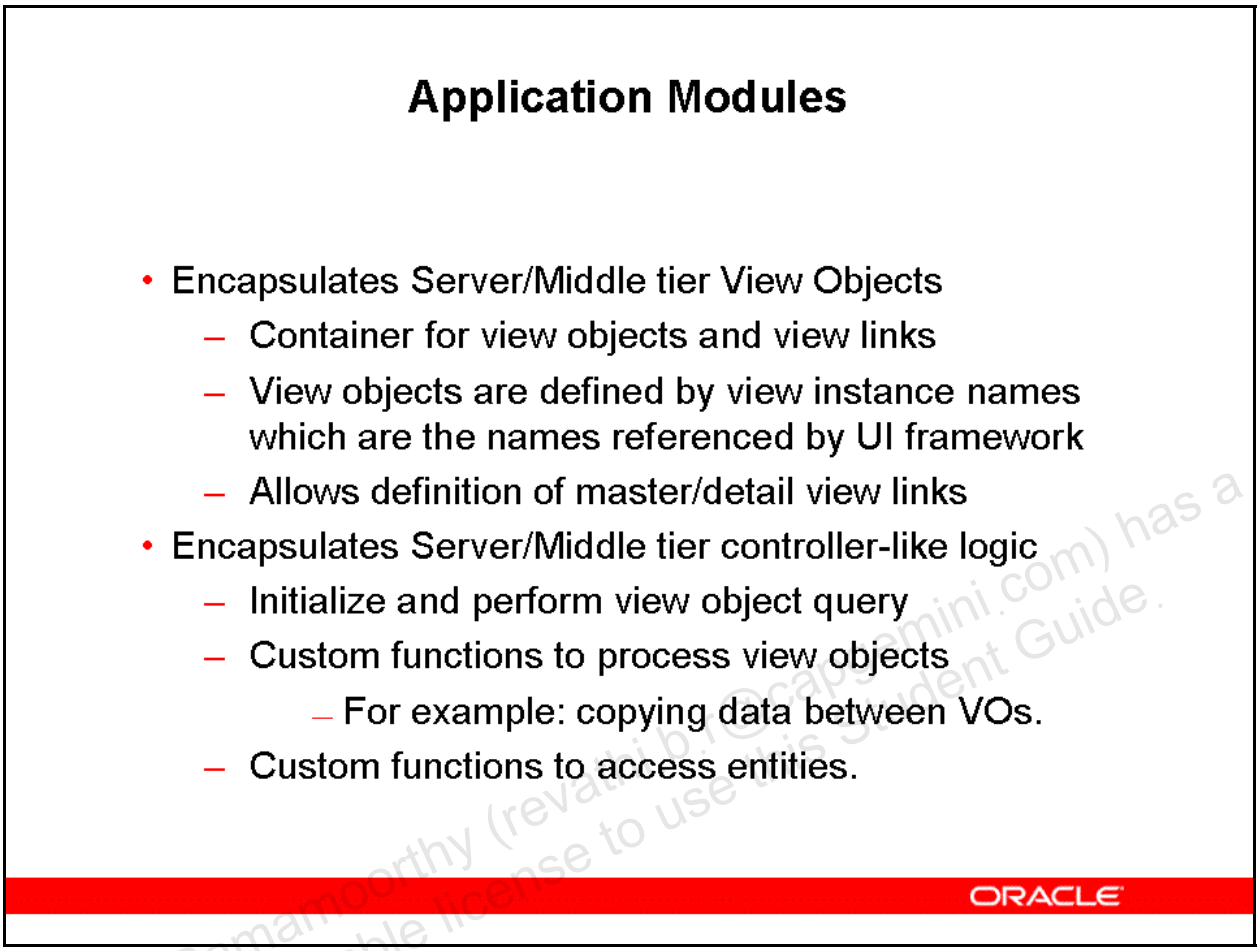
BC4J Package Naming Standards



Naming Standards are not enforced on customers. These standards are simply what Oracle uses. Like any standards, they evolve over time. If you encounter objects that don't adhere to any given naming standard, chances are those are older objects adhering to an older naming standard that is no longer used.

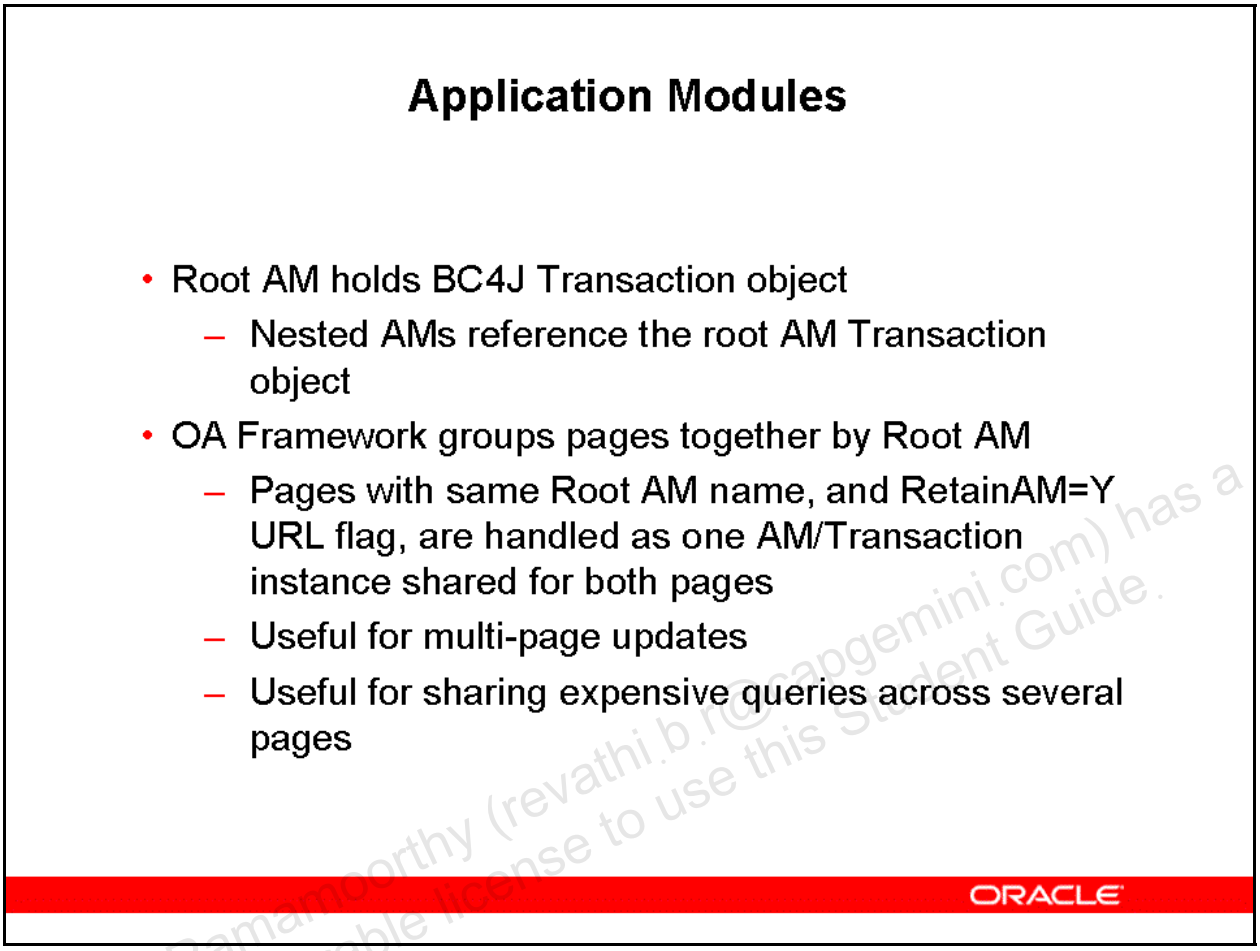


Application Modules



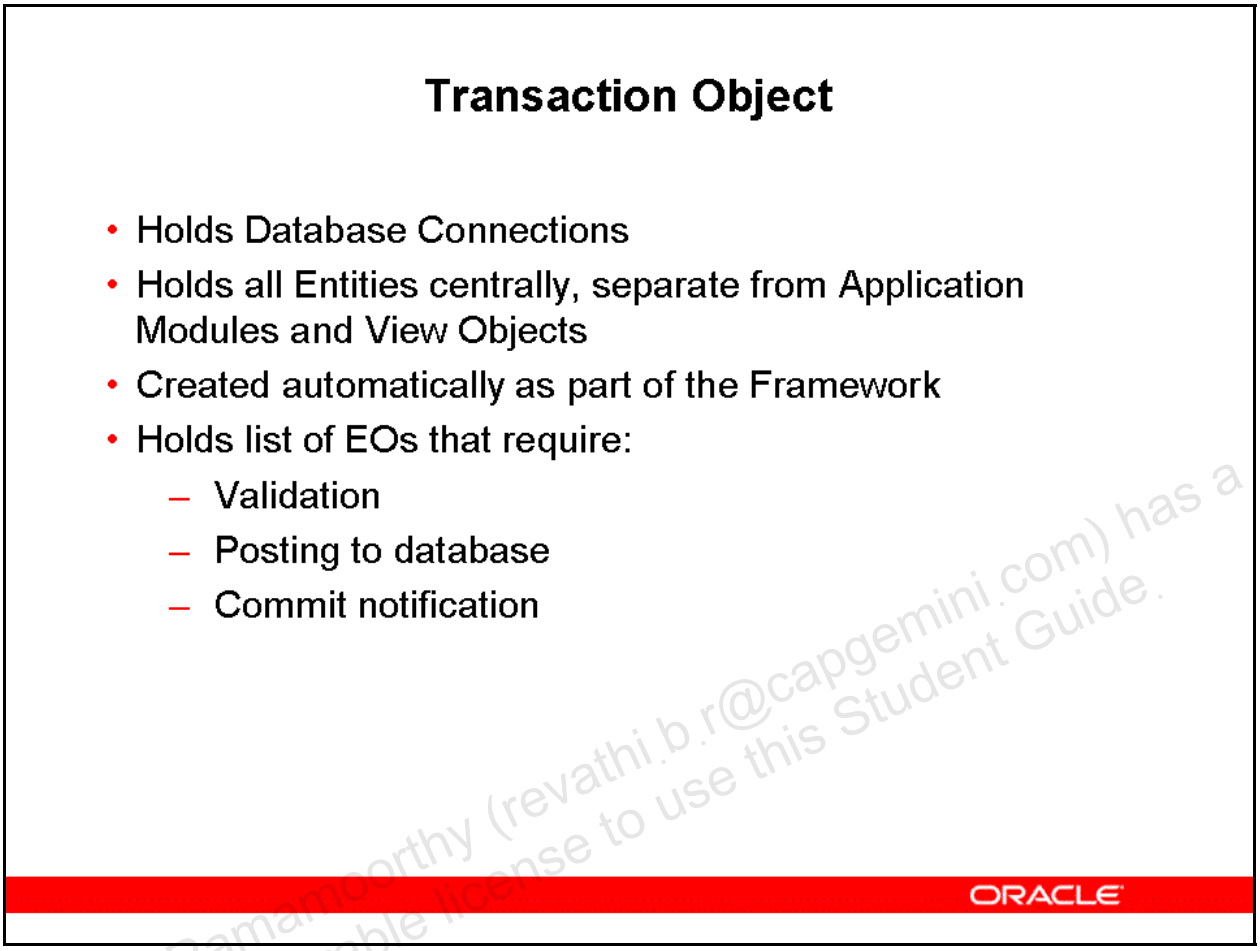
AMs communicate with Controllers and VOs.

Application Modules



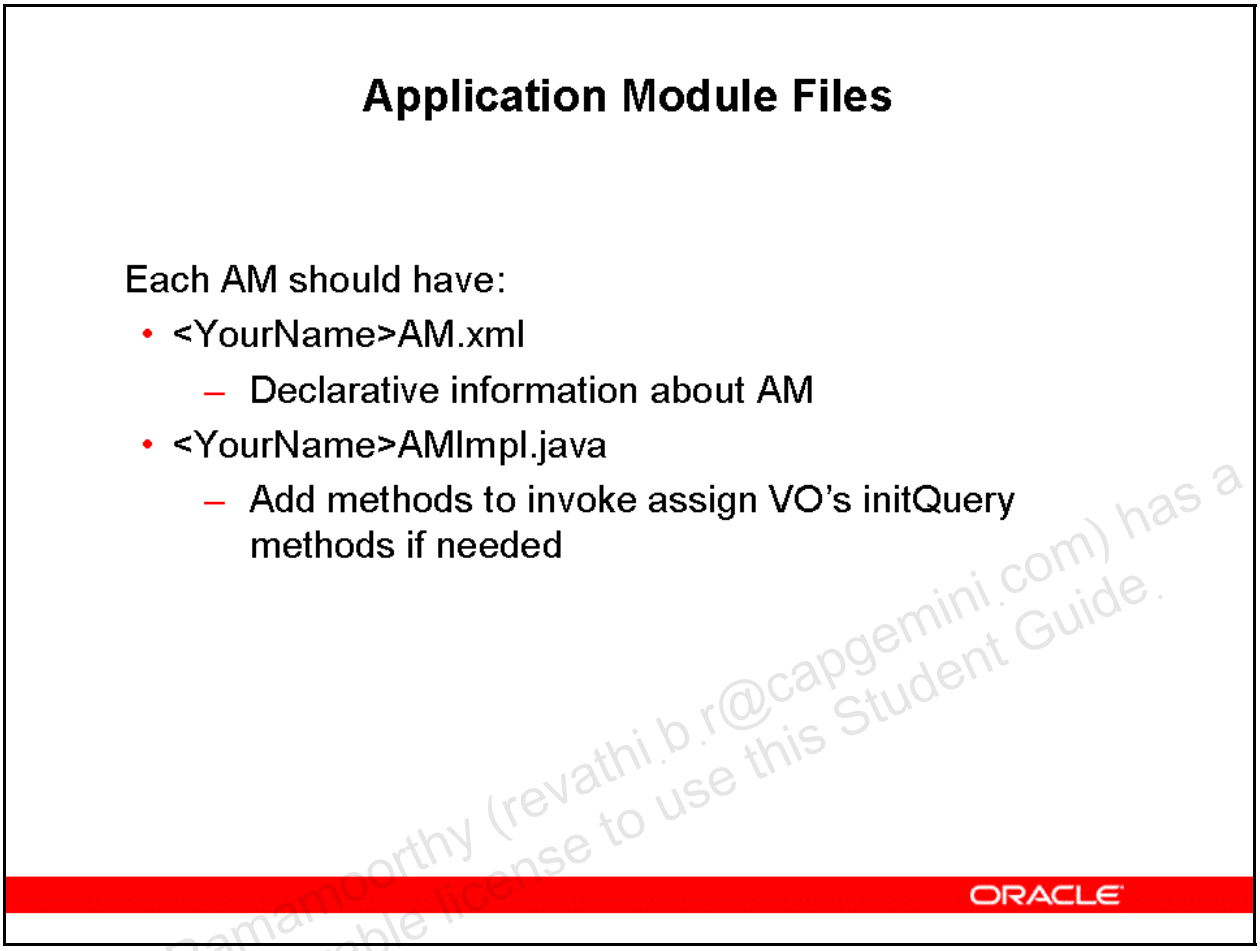
Transactions then get connections from a connection pool, and those connections are reused and recycled to increase performance and reduce resource requirements on the middle tier.

Transaction Object



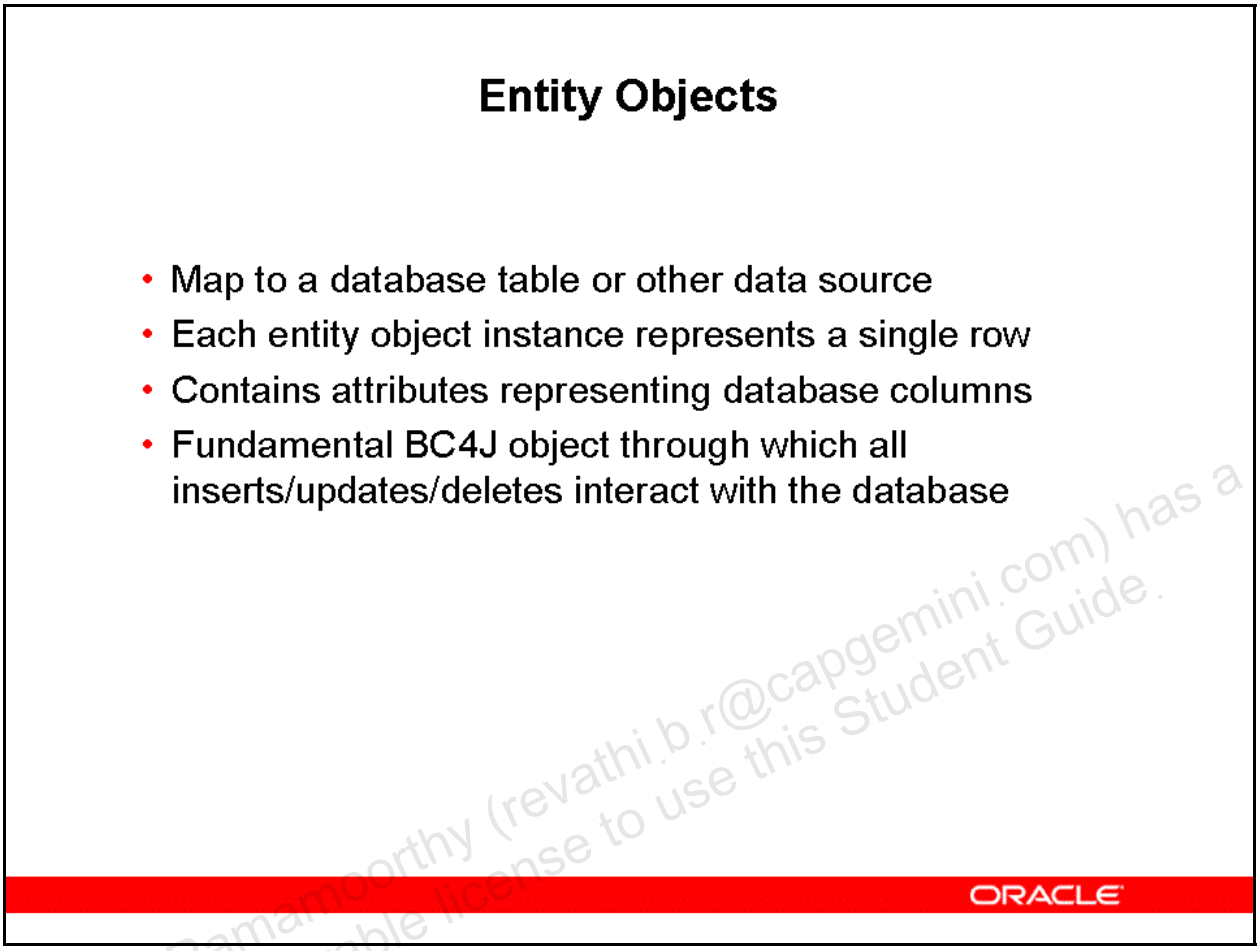
While it holds all the EOs, the EO and EO-related objects are not explicitly declared as belonging to the AM, like VOs are explicitly declared.

Application Module Files



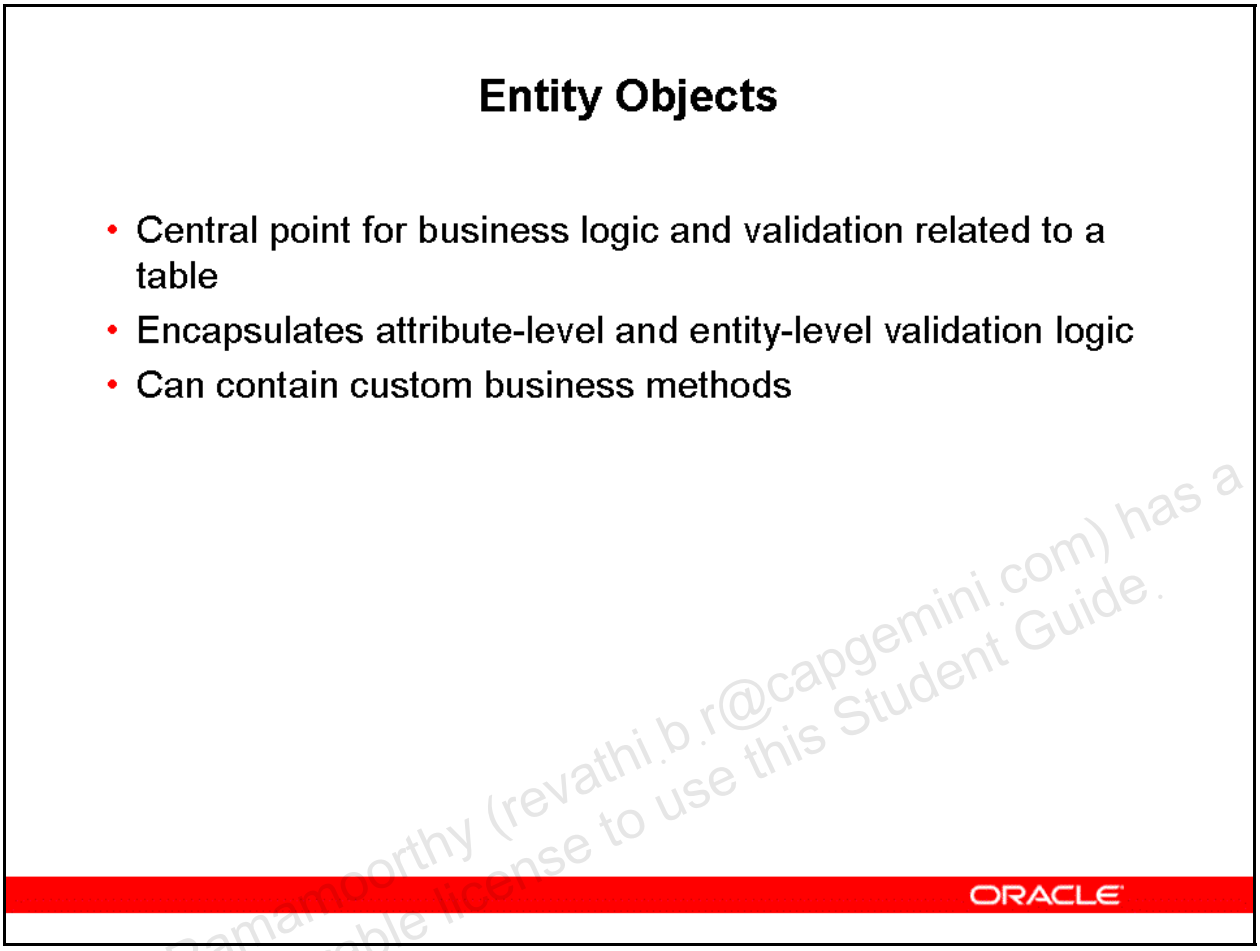
Most BC4J objects follow this format, a declarative (XML) component with a programmatic (Java) component. It is worth stating that the XML component for BC4J objects is not stored in the MDS.

Entity Objects

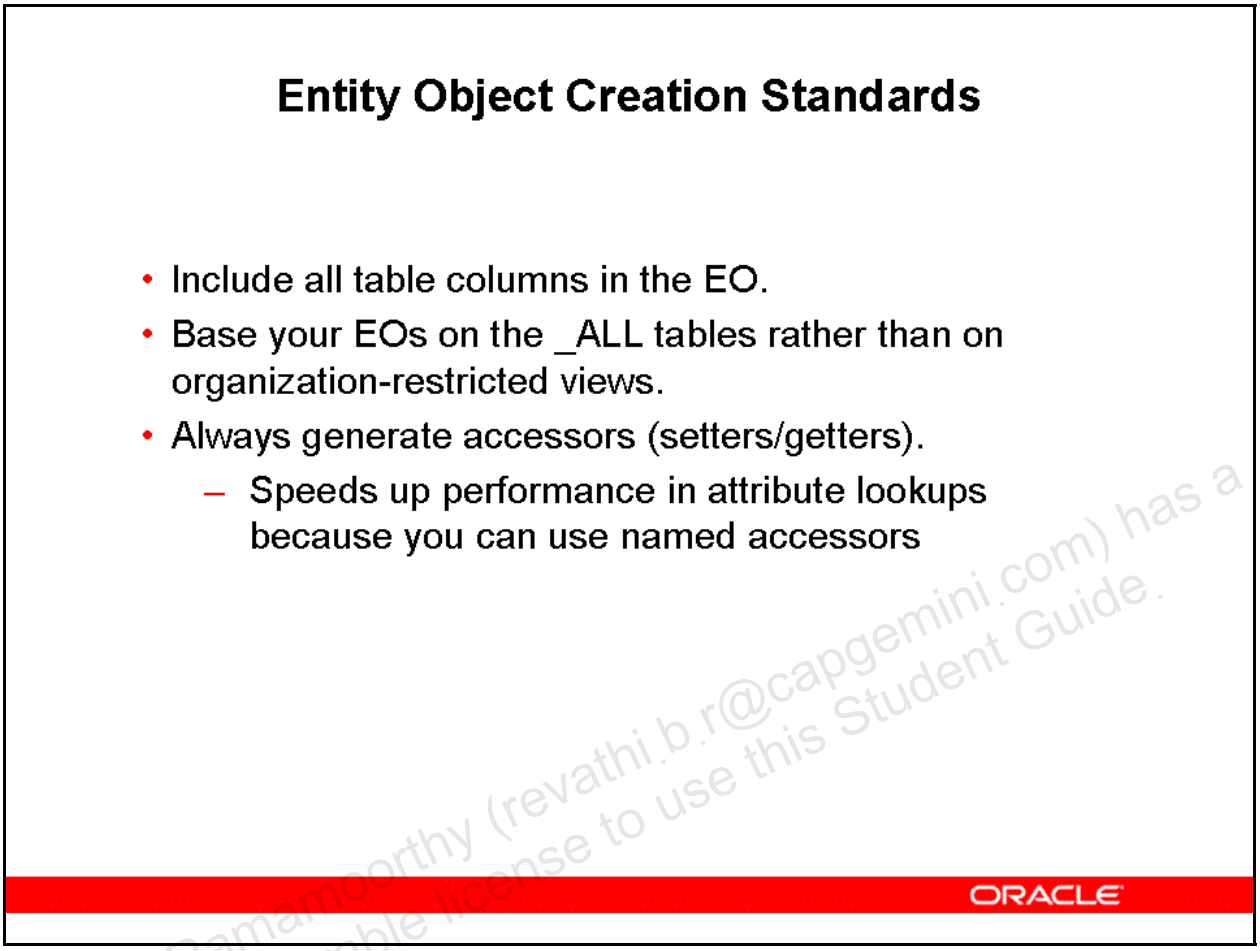


The general rule is one EO per table. Although in some cases, one EO may represent several tables. But, those are unusual circumstances.

Entity Objects

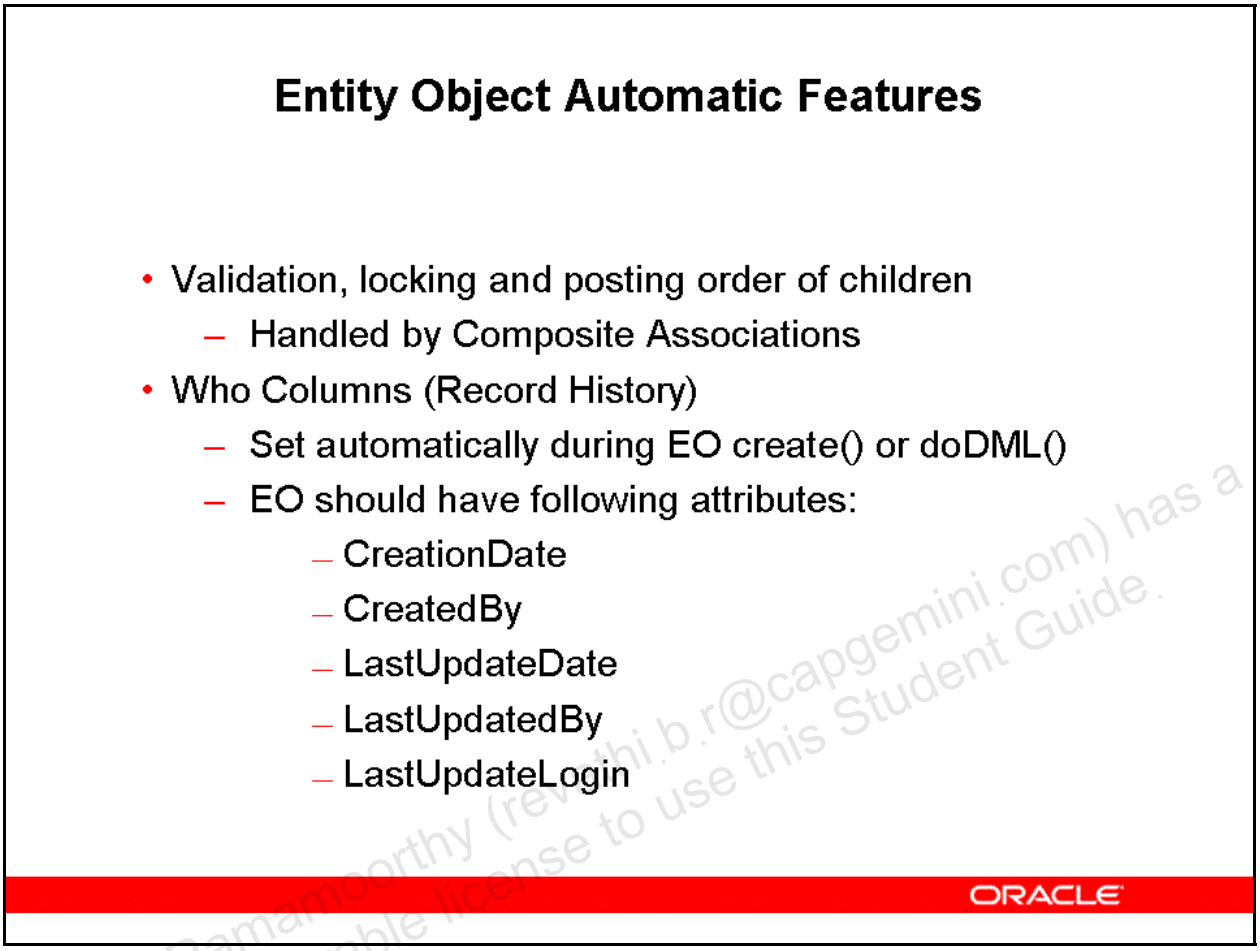


No Oracle created object will contain custom methods. And, the preferred method of implementing custom methods is to extend an EO.



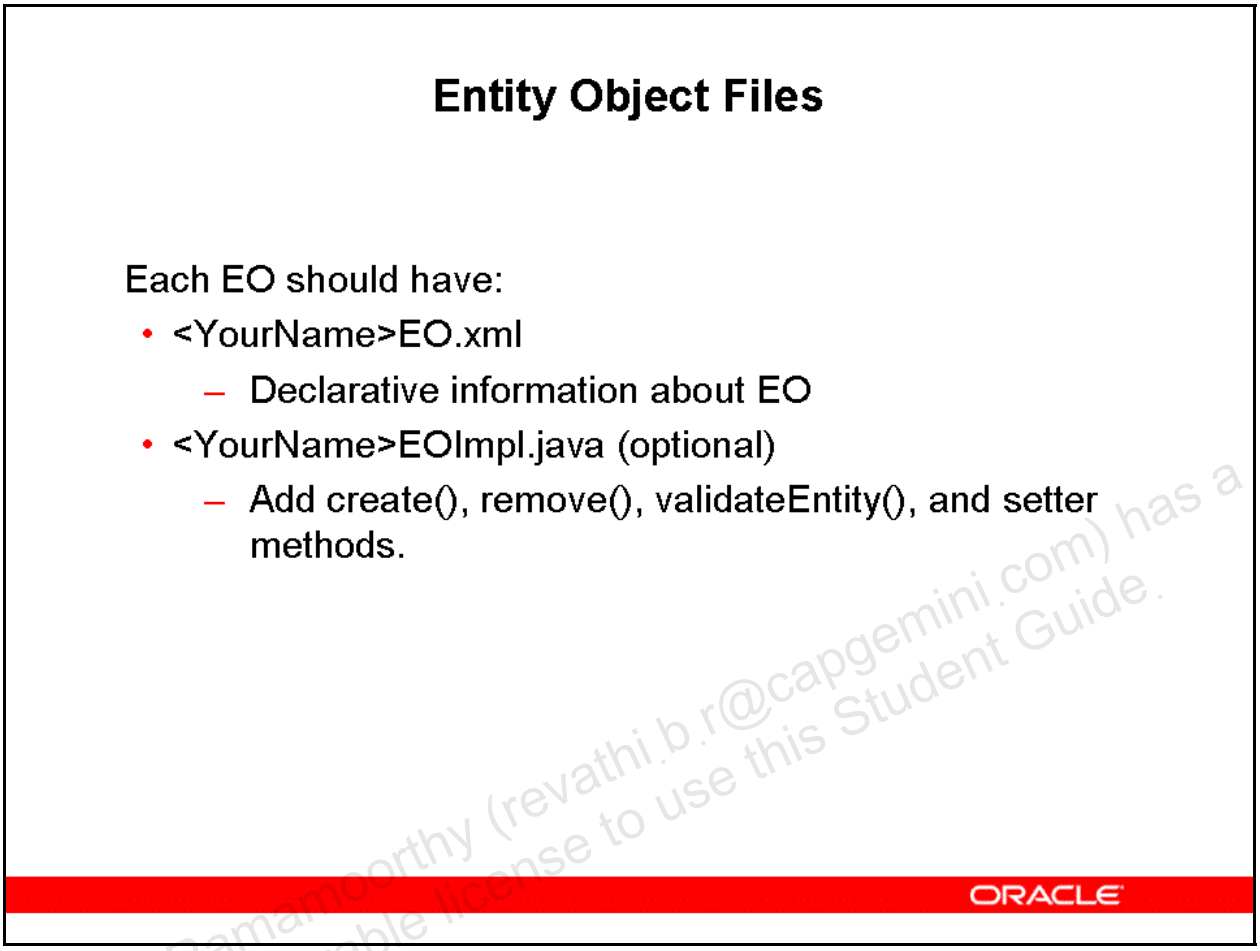
Why all columns? Simply because you only want one EO per table.

Entity Object Automatic Features



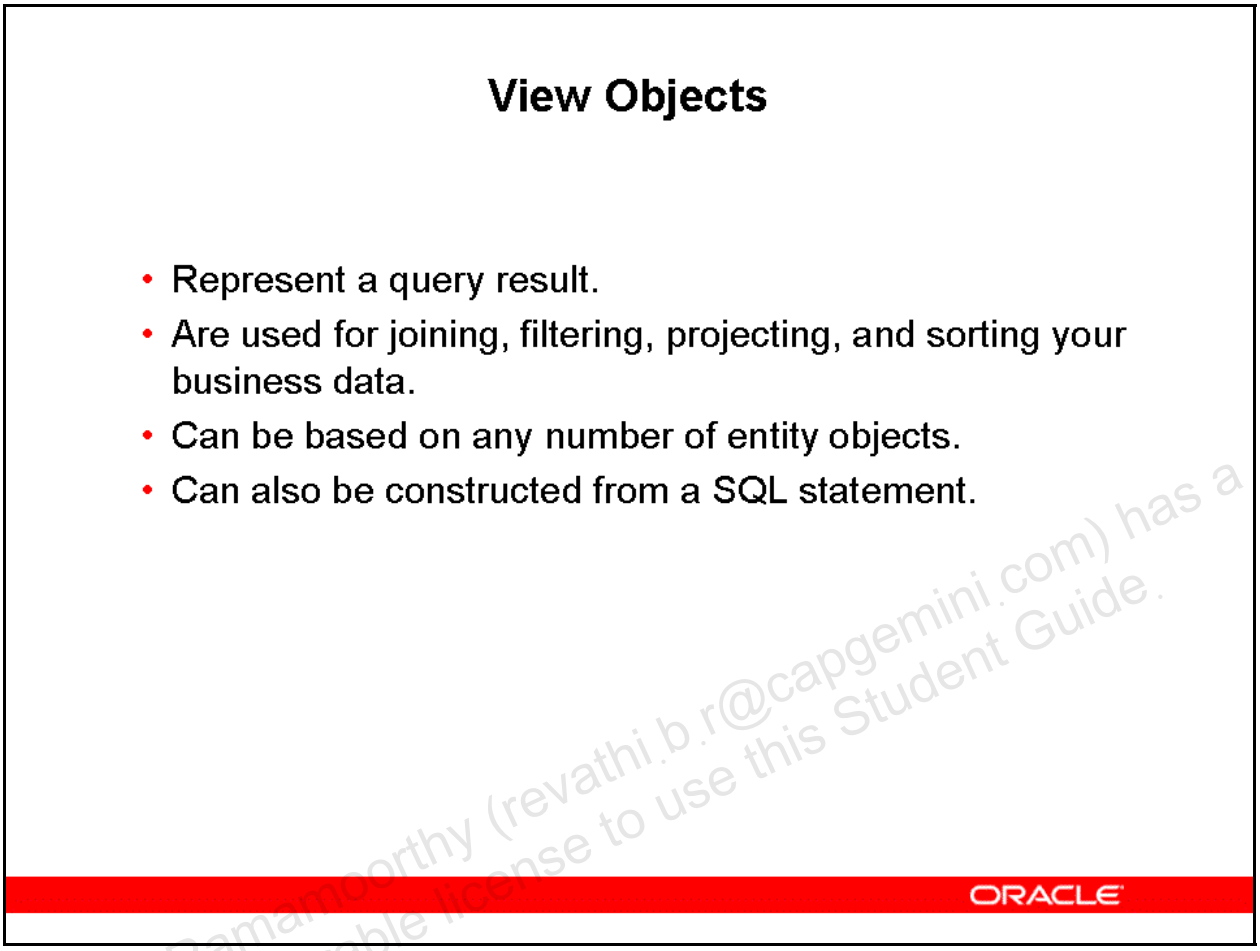
Composite associations control when validation business logic (declarative validation or program code in the entity objects) fires for the parent as opposed to the child.

Entity Object Files



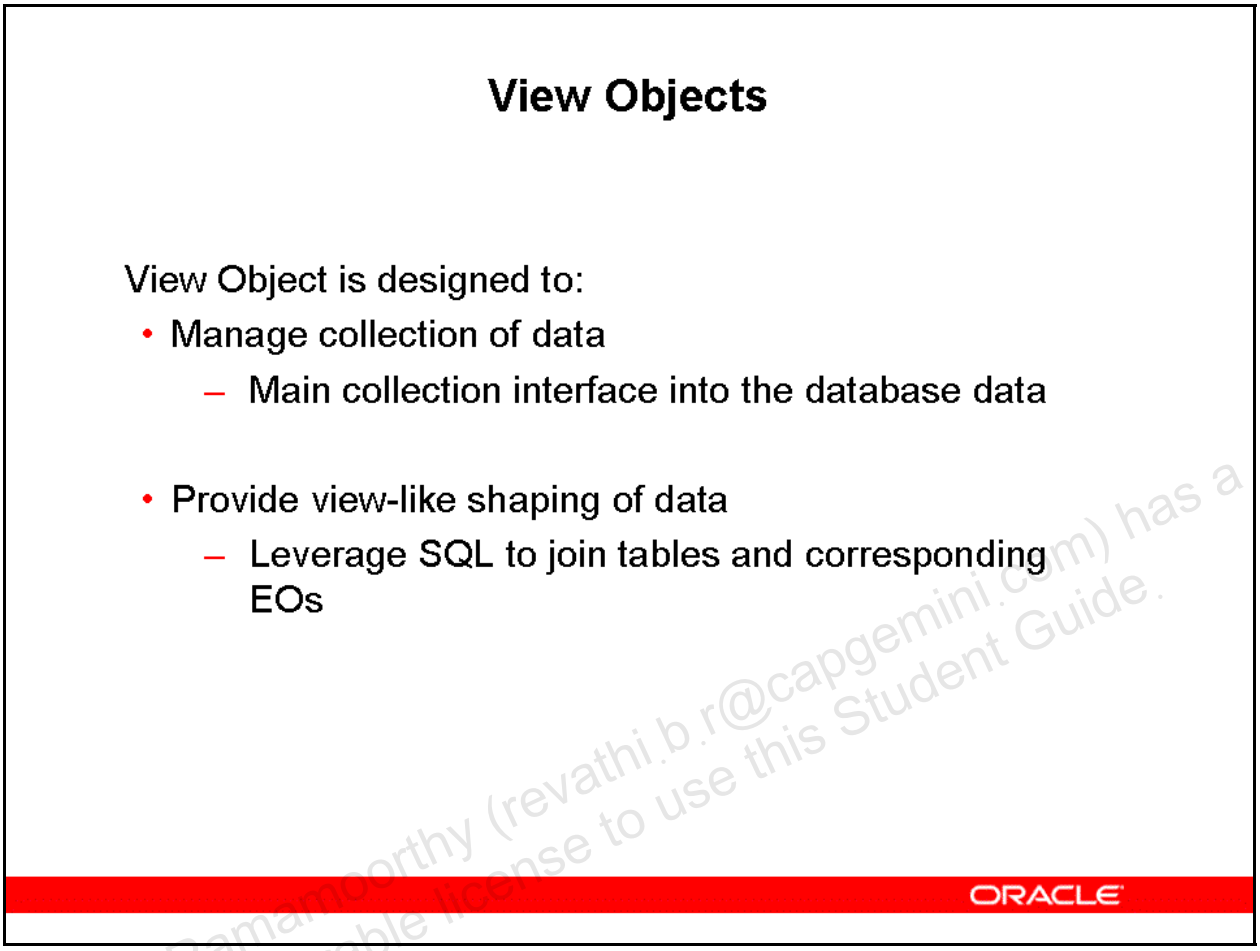
Most BC4J objects follow this format, a declarative (XML) component with a programmatic (Java) component. It is worth stating that the XML component for BC4J objects is not stored in the MDS.

View Objects



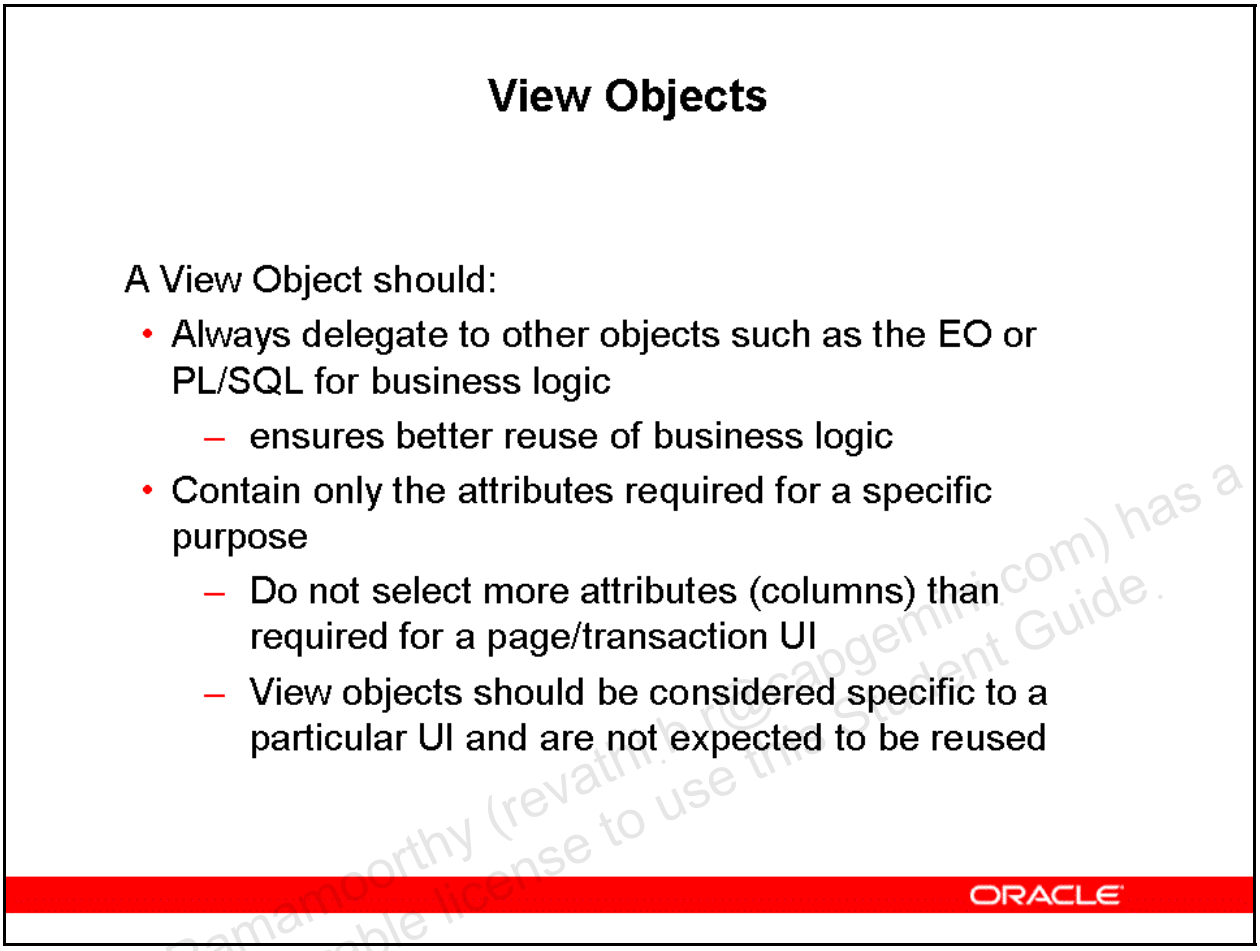
It is an oversimplification to state that a VO is a SELECT statement. A VO is an object. The object represents both the data (the SELECT), and the operations (methods) you can perform on that data (gets/sets/validations). Most VOs are defined declaratively, but there are even methods to programmatically create and alter VOs at run-time.

View Objects

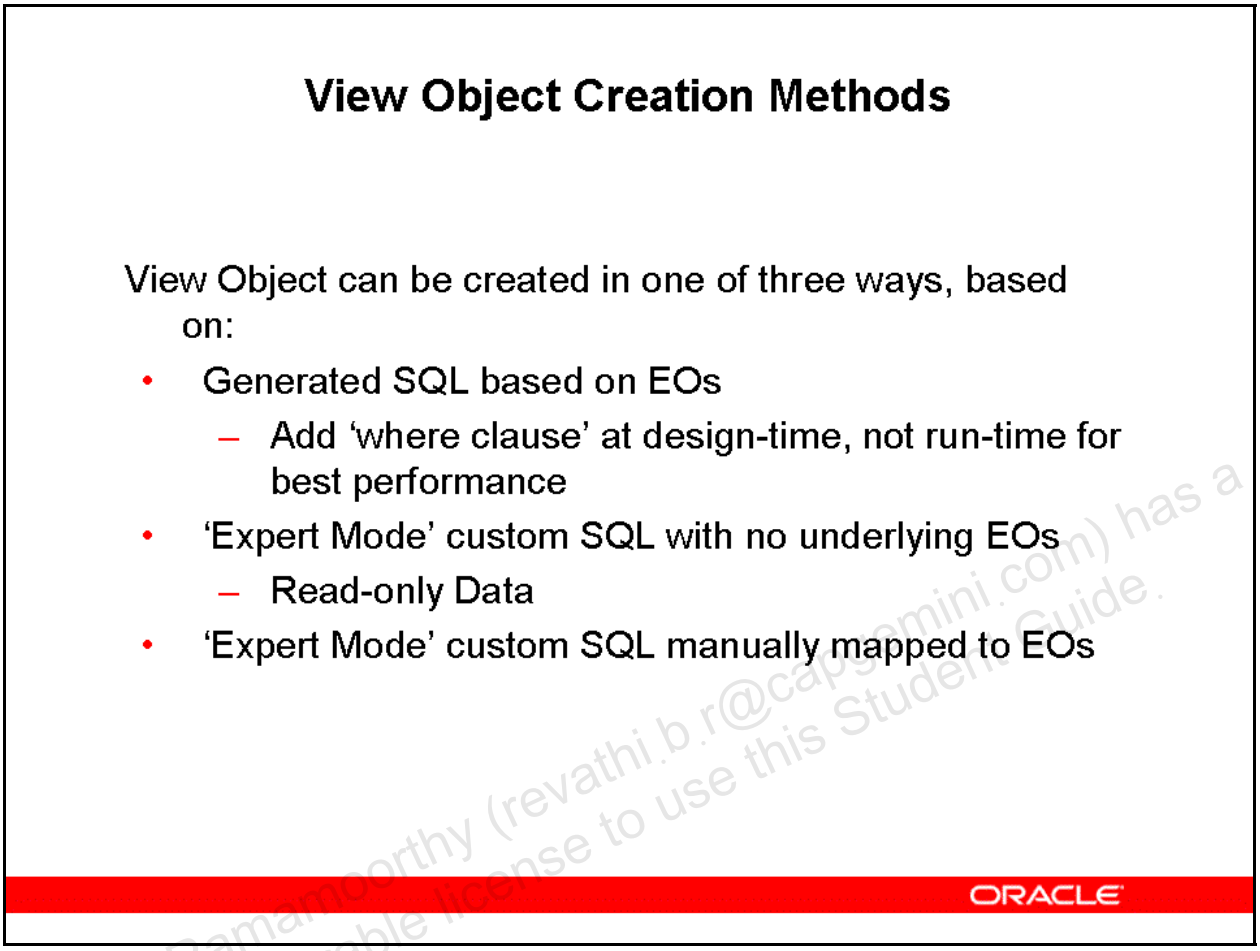


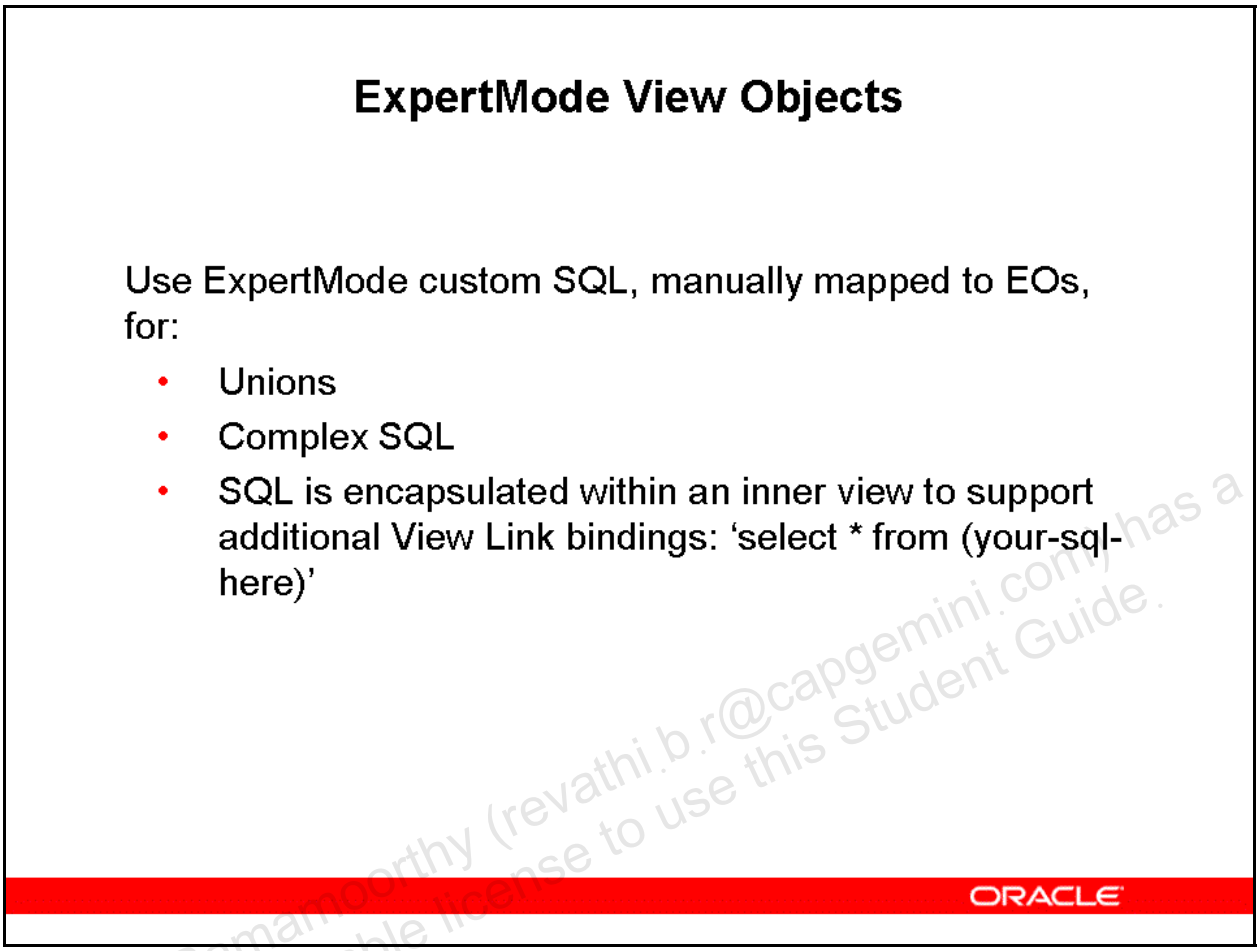
The VO allows us to do the action most often performed, a query, without the overhead of a full EO. This significantly increases the performance of OA Framework pages.

View Objects

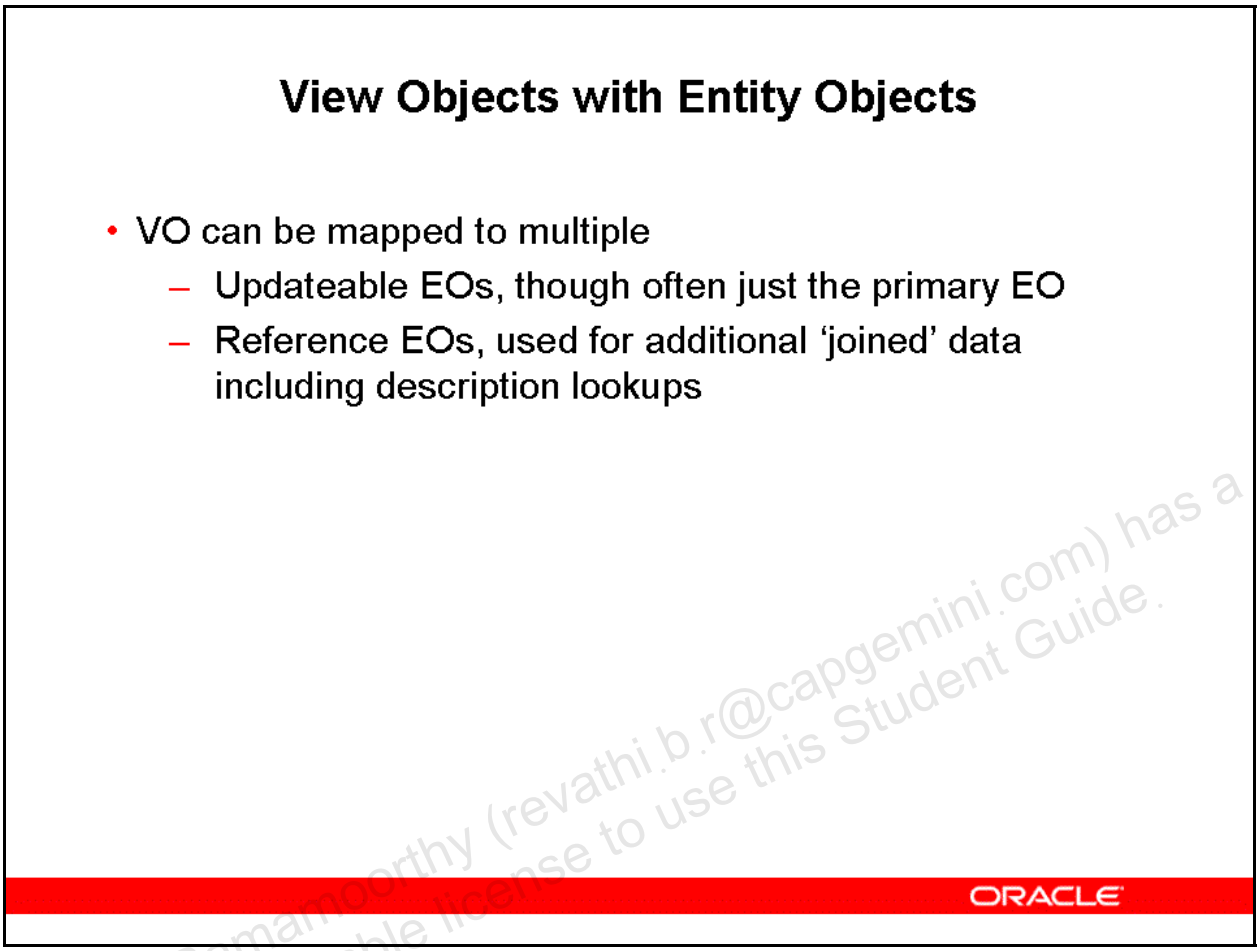


You should never put business logic in your view object.



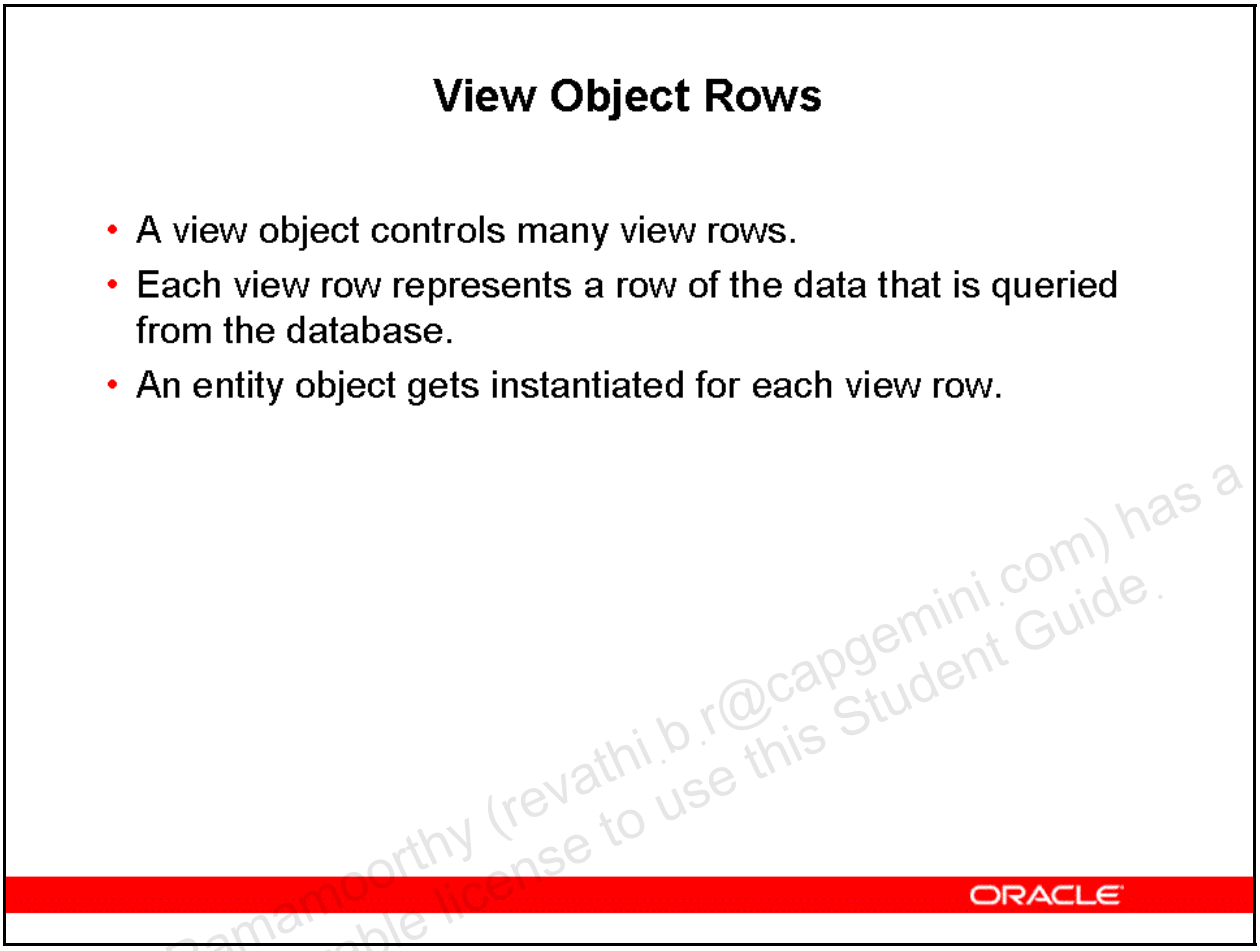


View Objects with Entity Objects



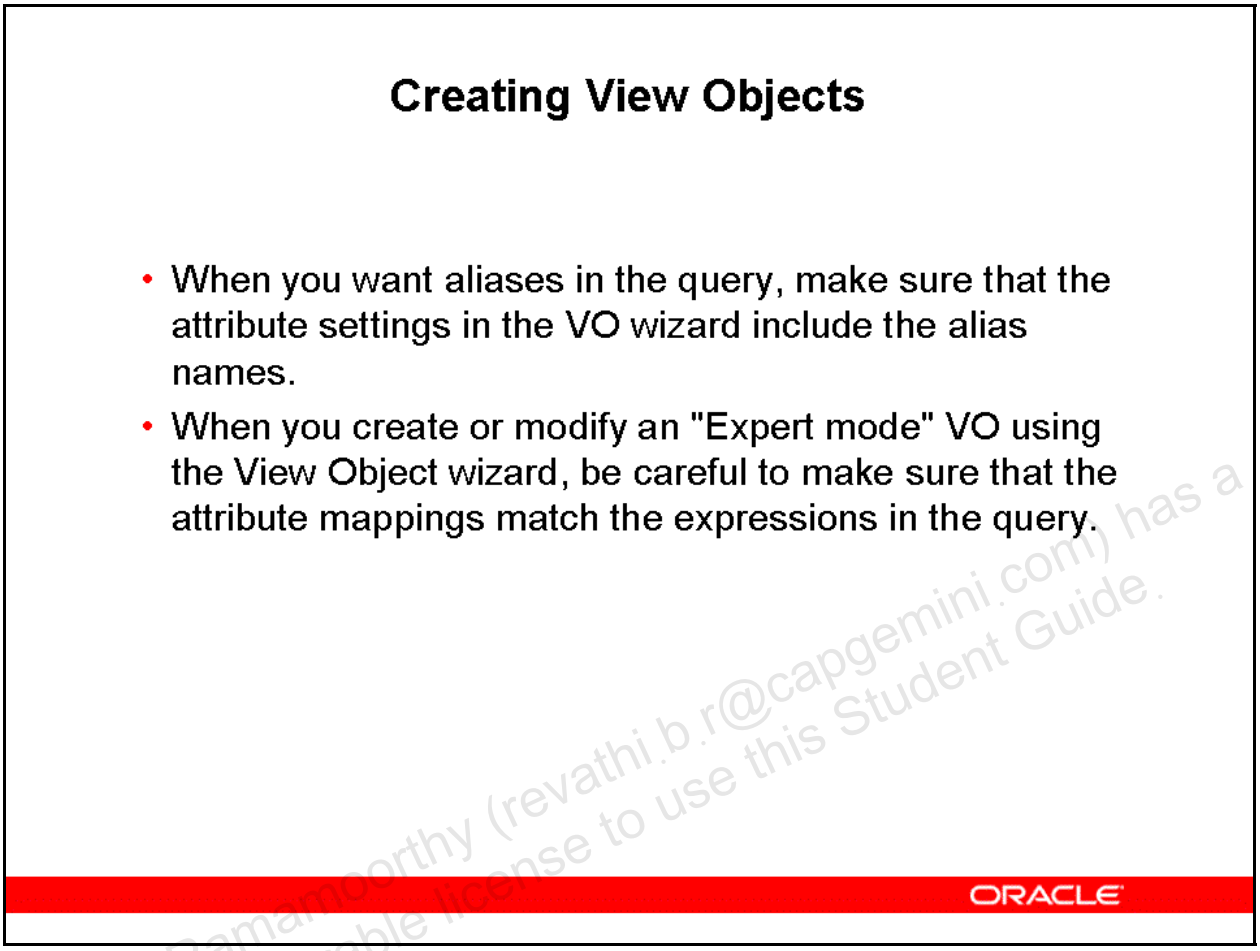
EOs are the objects through which we insert, update, and delete data. VOs are the objects through which we query (select) data. This causes a natural disconnect of sorts. If you have a VO based on an EO, what is that VO capable of doing? A VO mapped to updateable EOs allows the actions on the EO to occur through the VO. VOs mapped to reference EOs are using those EOs just for query purposes, generally a foreign key of some sort.

View Object Rows



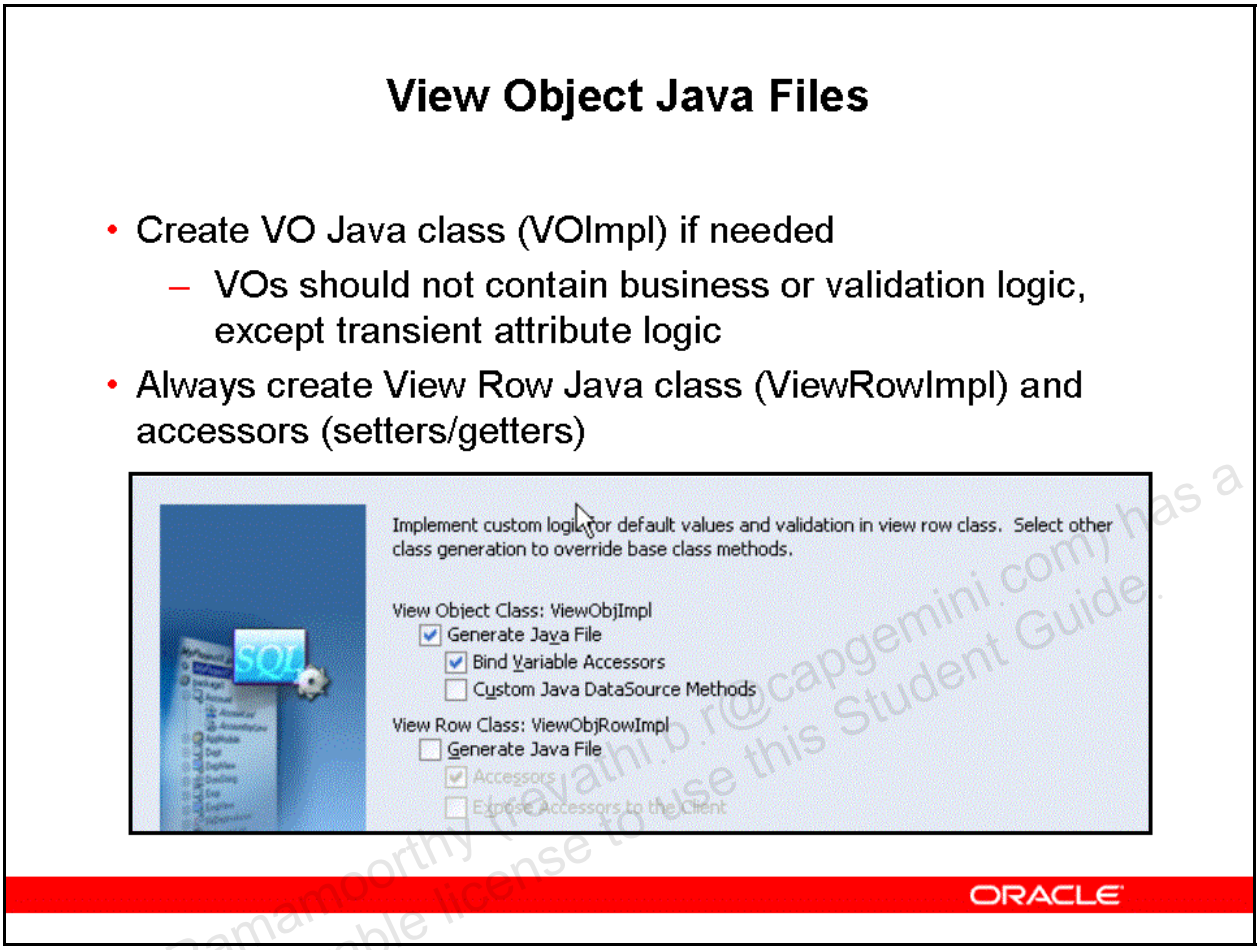
Unlike the SELECT statement itself, VOs have a means by which to iterate through each returned row. But, there are performance implications, and your VO should be as precise as possible to return the fewest number of records. And, you should consider a limit beyond which you VO should not return any additional rows. The default in OA Framework is 200 records.

This default behavior is controlled by a profile option, FND: View Object Max Fetch Size (VO_MAX_FETCH_SIZE). Changing this profile option should not be done lightly. If this value is increased, it will increase the memory requirements of the JVMs running on the middle tier, and it will impact the performance of the servers. The business requirements of your application may require you to increase the value. But, be aware of the consequences.



Note: Also be aware that if you create a query in Expert Mode, and then toggle out of Expert Mode, all of your SQL changes are lost.

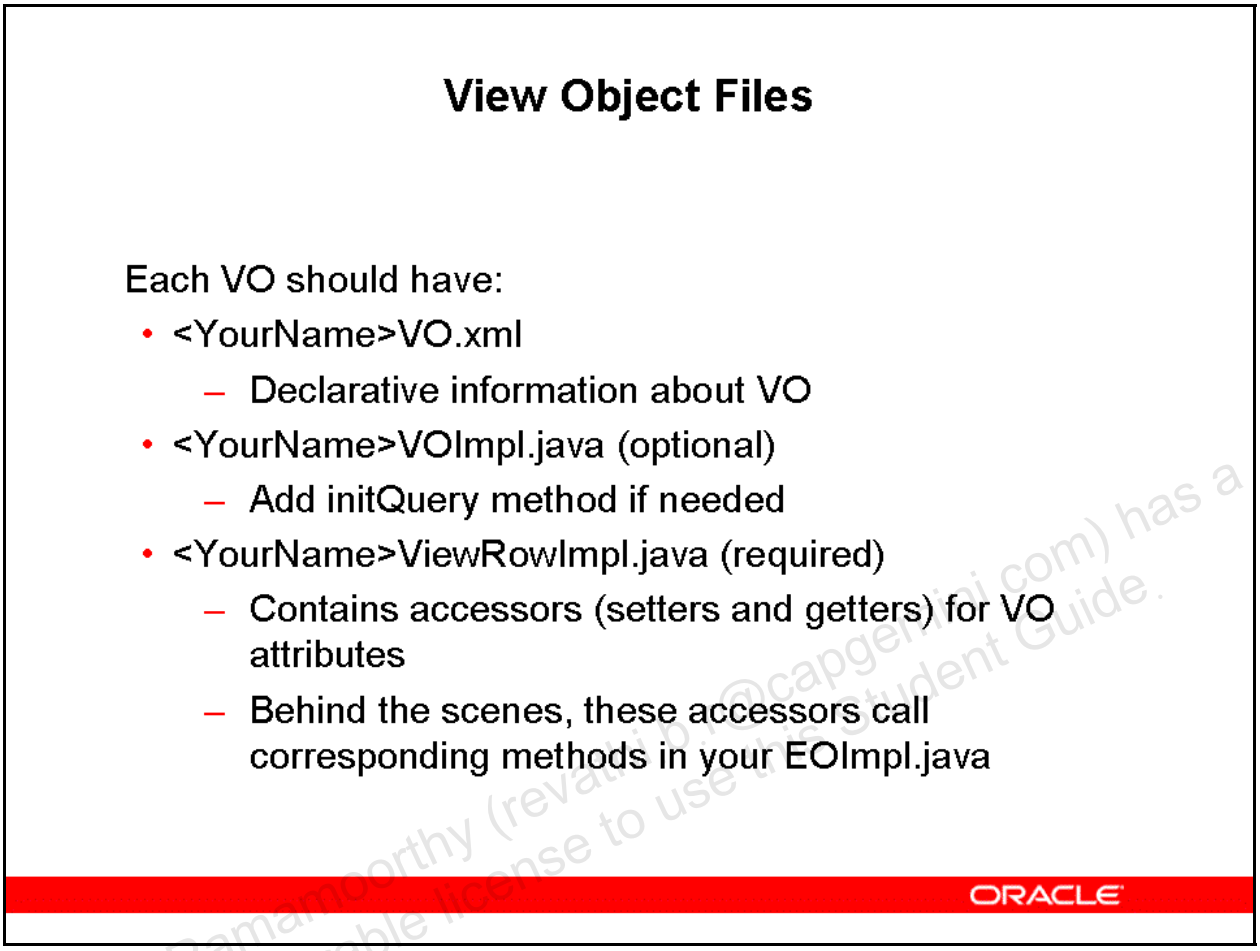
View Object Java Files



The ViewRowImpl class should always be created for two reasons. One, it is the iterator that allows you to step through the individual rows of the VO. Two, it generates the get/set methods for the VO. Merely creating the get/set methods doesn't gain anything, but it does allow others to extend those methods should the need arise.

Note: On this same page of the wizard, there is an Object Definition class (ViewObjDefImpl) that can also be selected (created). What is this? An Object Definition class allows for the run-time creation of the definition of a VO object. If this sounds indirect, it is. While it preserves a pure object-oriented model, it doesn't perform or scale properly. These objects are not used in OA Framework pages. Even in non-OA Framework pages, they are seldom used because of their poor performance characteristics.

View Object Files



Most BC4J objects follow this format, a declarative (XML) component with a programmatic (Java) component. It is worth stating that the XML component for BC4J objects is not stored in the MDS.

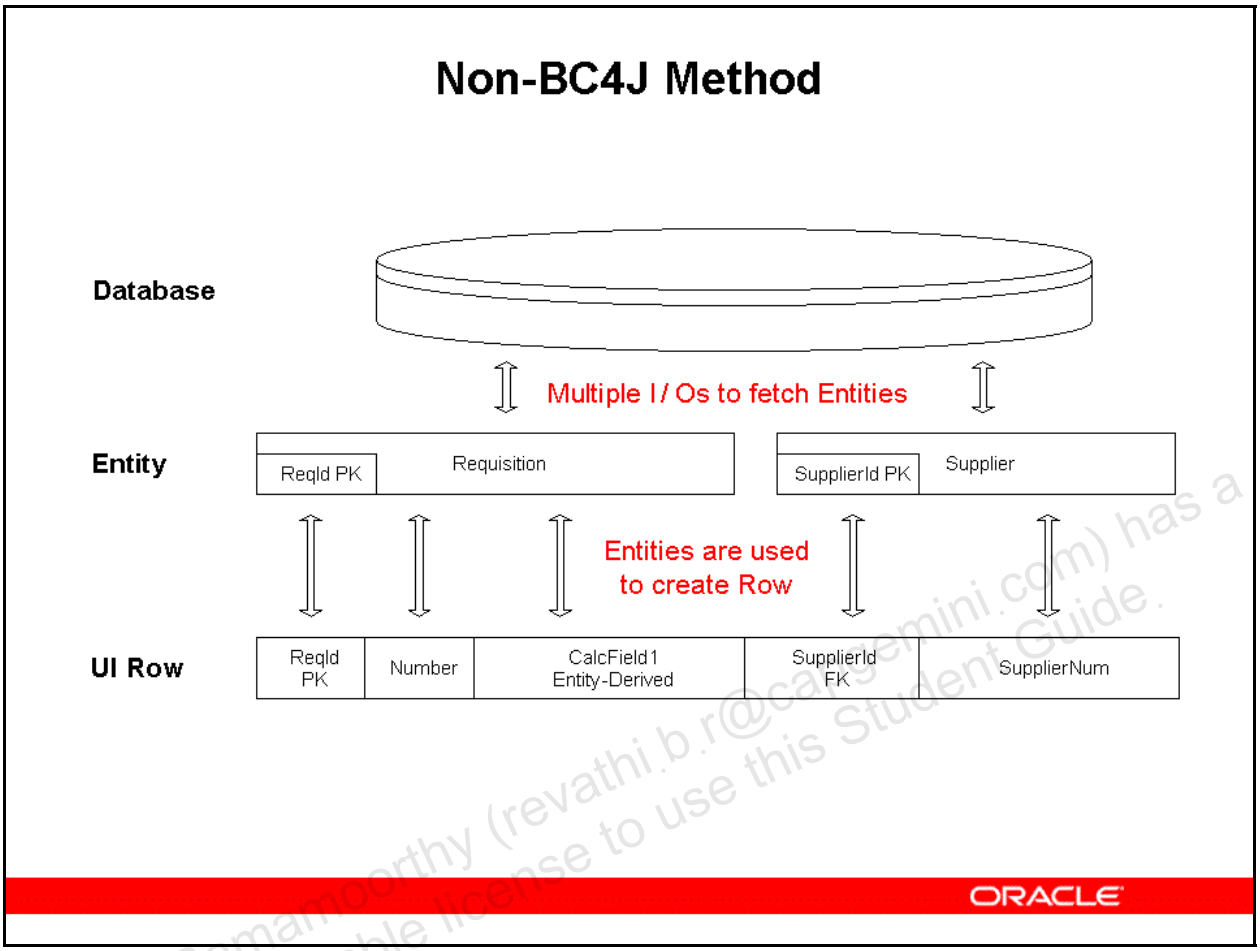
BC4J Database Interactions

Examine the following scenarios:

- Pure object-oriented (non-BC4J) method
- View Objects without Entity Objects (read-only queries)
- View Objects with Entity Objects
  - Step 1: View Object initial query
  - Step 2: Entity Object population
  - Step 3: Entity Object reuse
  - Step 4: Entity-derived attributes
  - Step 5: Entity Object fault-in
  - Step 6: Entity Object references
- EO/VO merge

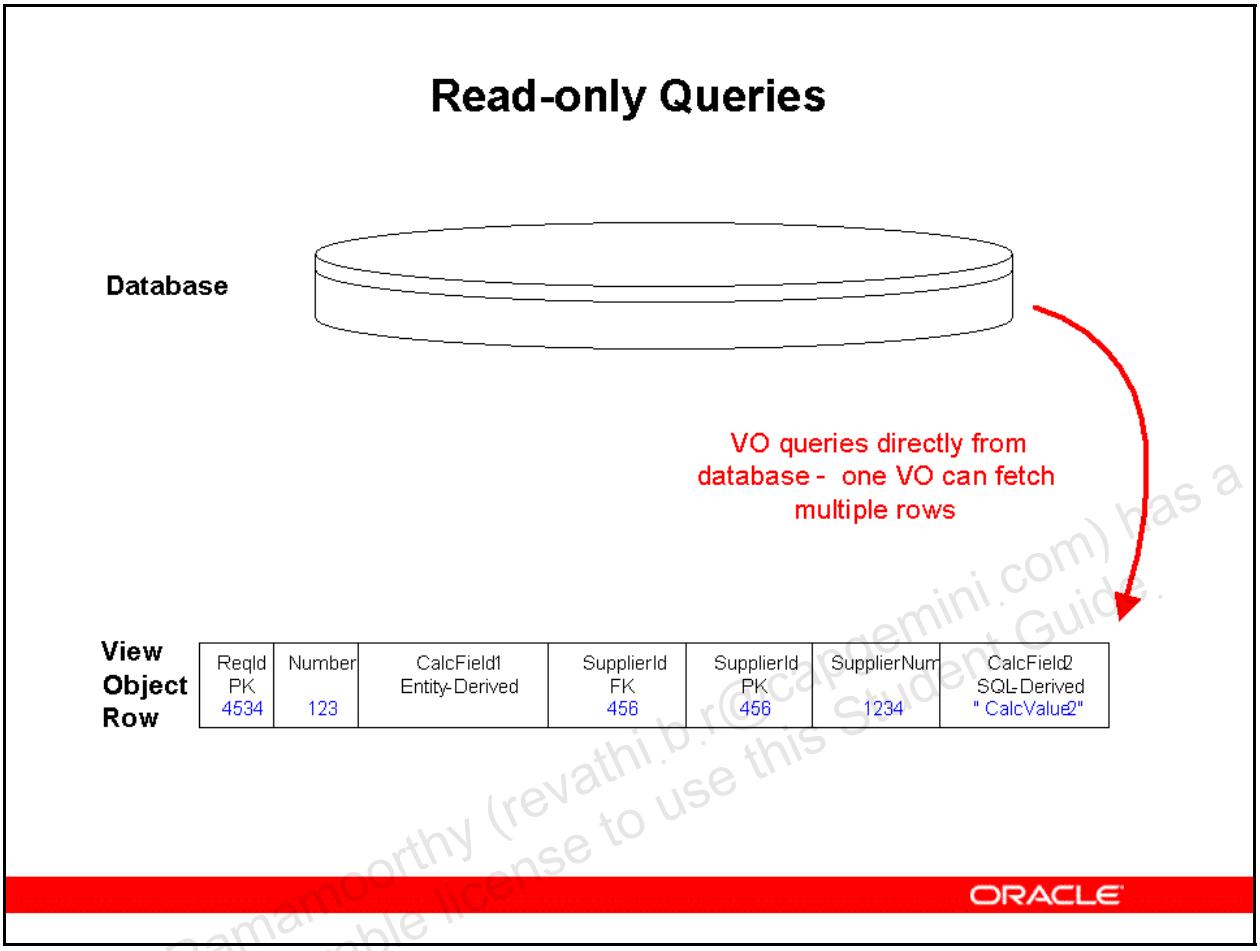
ORACLE

Non-BC4J Method



This method chooses object-oriented purity over performance. While ideal in a theoretical sense, performance is always, in the eyes of the end-user, a priority over programming methodology.

Read-only Queries

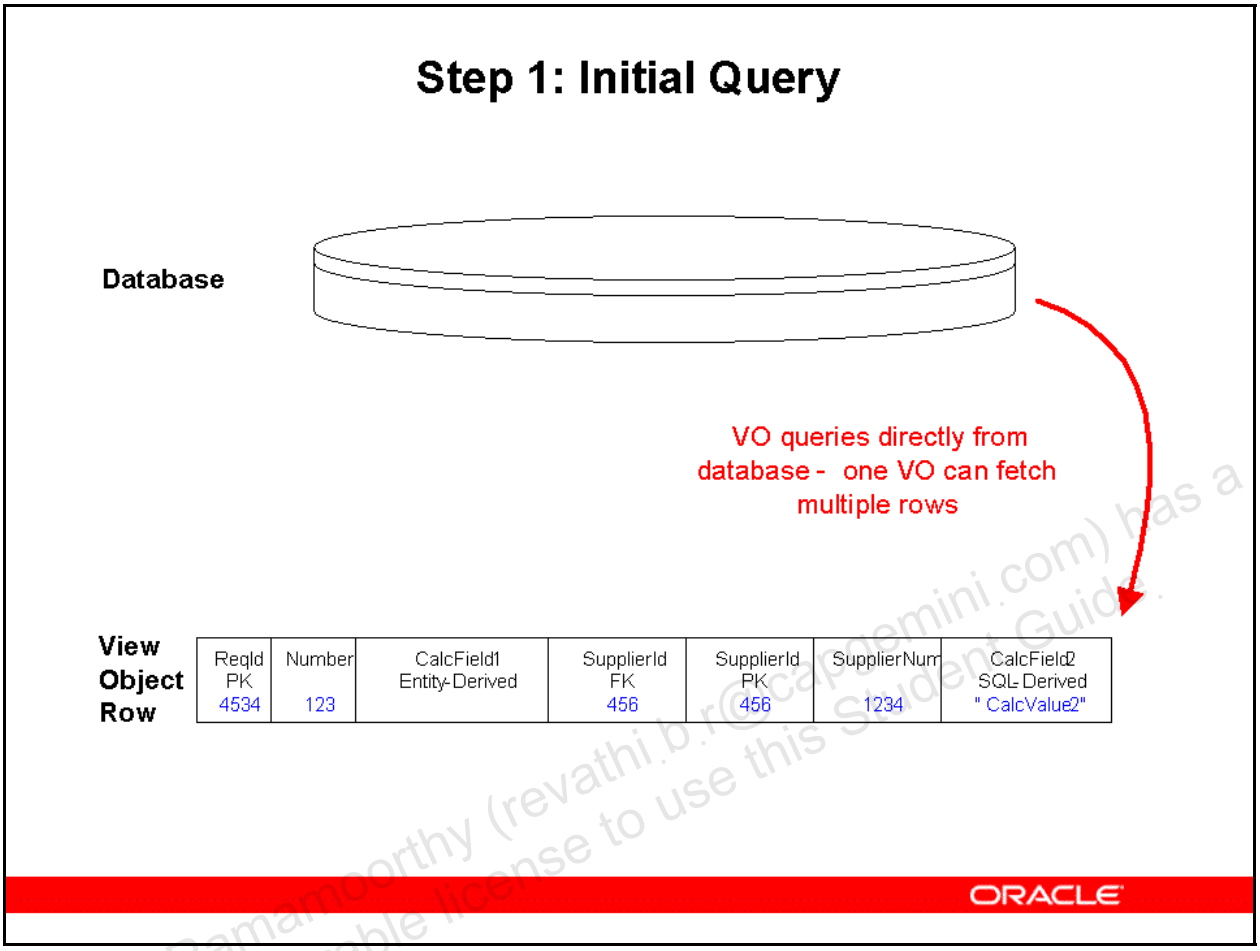


Entity objects are not required for read-only queries. So, BC4J breaks the Entity encapsulation for View Objects. This is done for significant performance improvement reasons.

Note: While the diagram shows one row, one VO can potentially fetch multiple rows during its query.

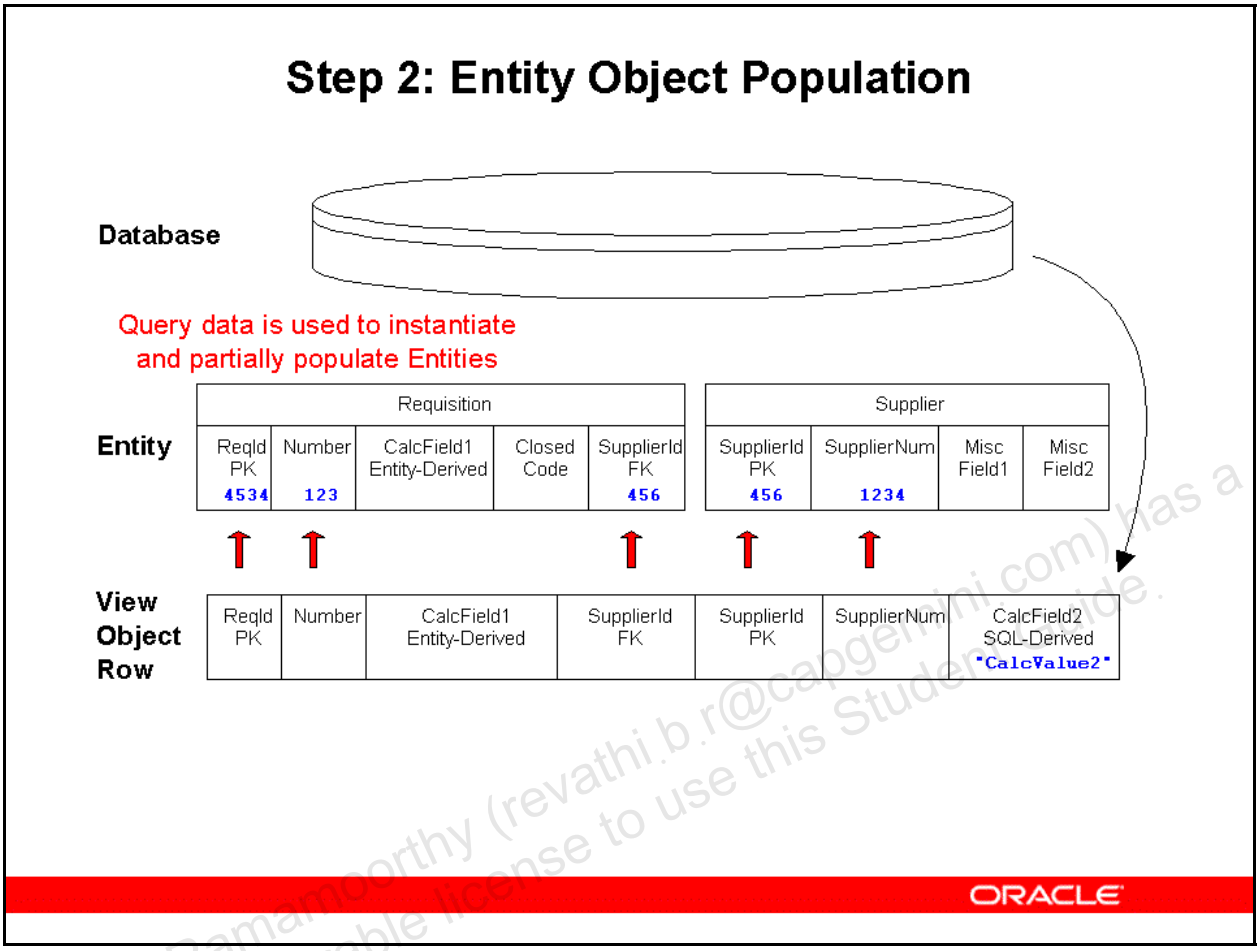
Note: The difference between a read-only query and an initial query is nothing at this point. Underlying the process is the definition of the VO. If the VO has no associated EO, it is read-only. If the VO has been defined with an associated EO, then the first step is an initial query.

Step 1: Initial Query



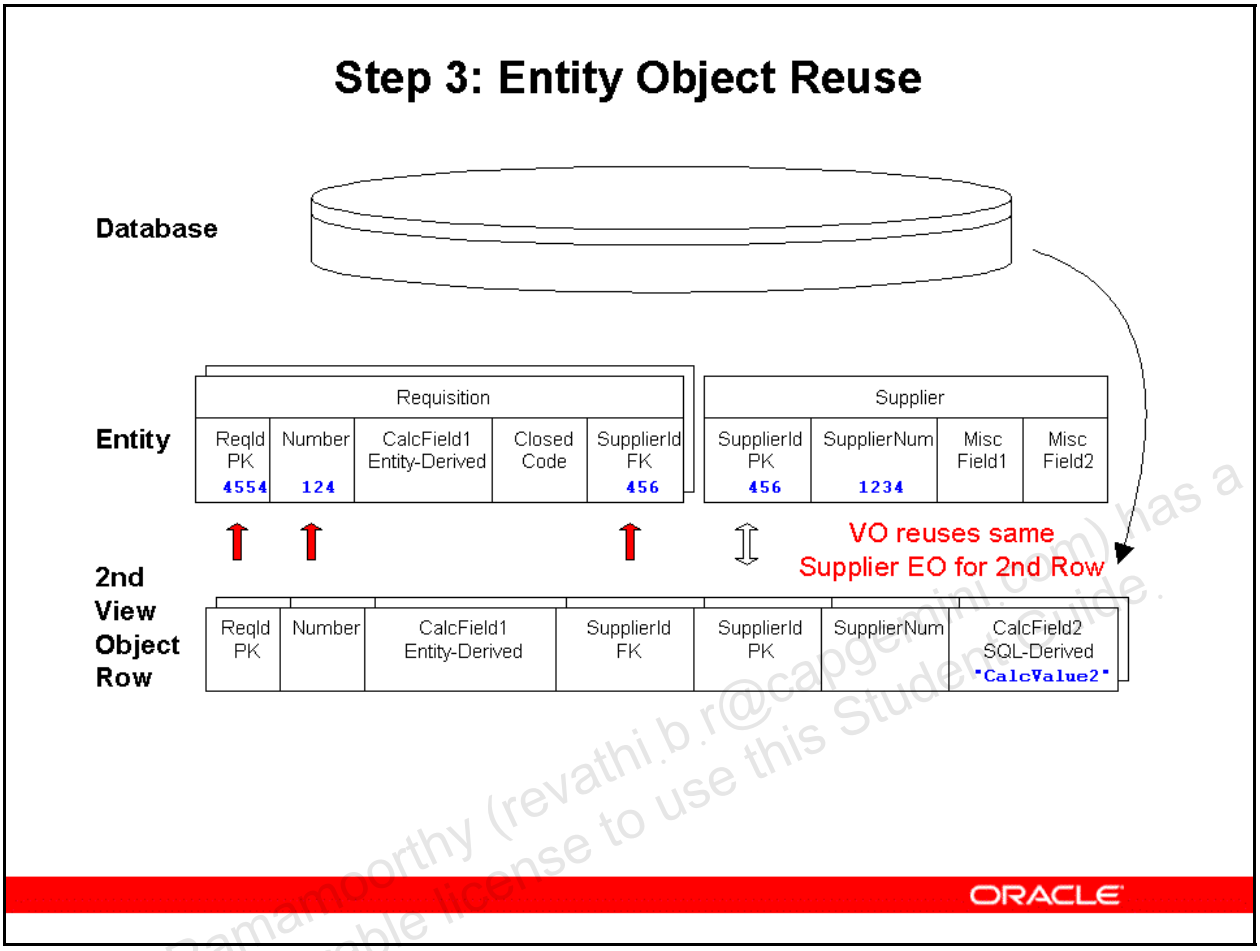
Note: The difference between a read-only query and an initial query is nothing at this point. Underlying the process is the definition of the VO. If the VO has no associated EO, it is read-only. If the VO has been defined with an associated EO, then the first step is an initial query.

Step 2: Entity Object Population



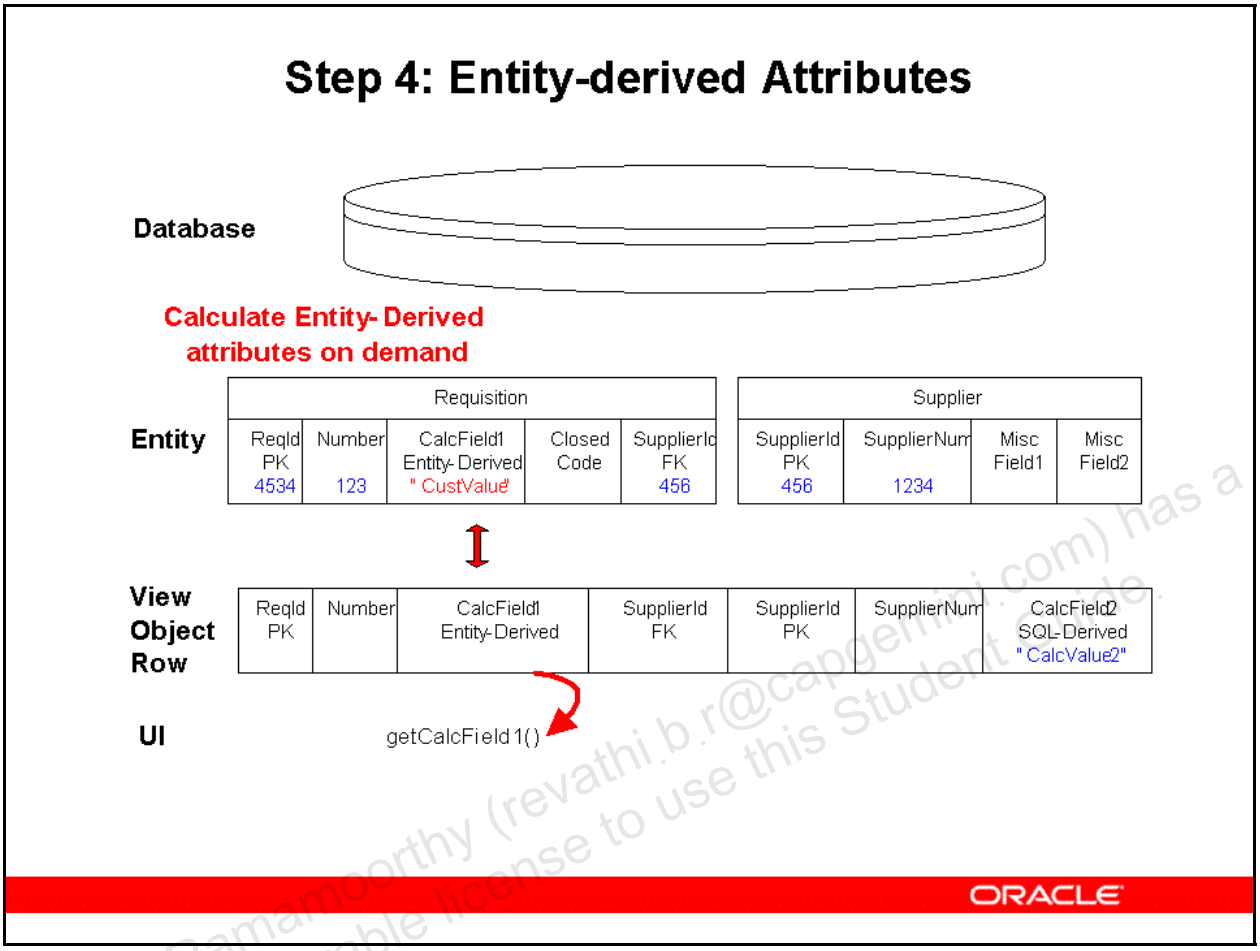
The query can bring back many view object rows (the result set or row set) in a single fetch from the database. Each view object row corresponds to an entity object (entity object instance). If the view object is based on an entity object, entity objects are found or instantiated by primary key (PK) for each VO row. No data is preserved in the VO-layer except for those columns that are SQL-derived or transient, because those objects are only found in the SELECT and/or VO definition.

Step 3: Entity Object Reuse



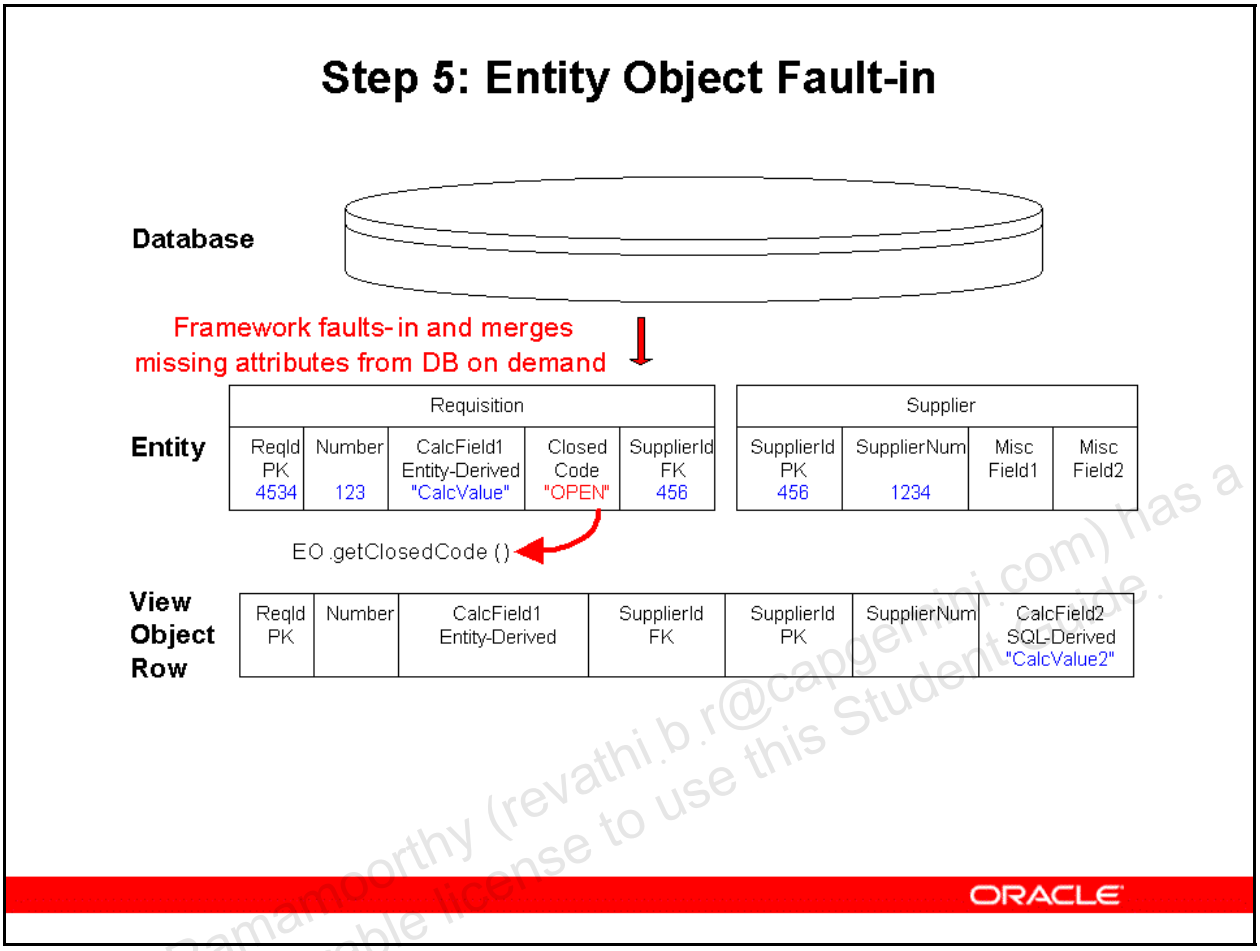
The 2nd VO Row uses less memory since it references and reuses the same Supplier EO

Step 4: Entity-derived Attributes



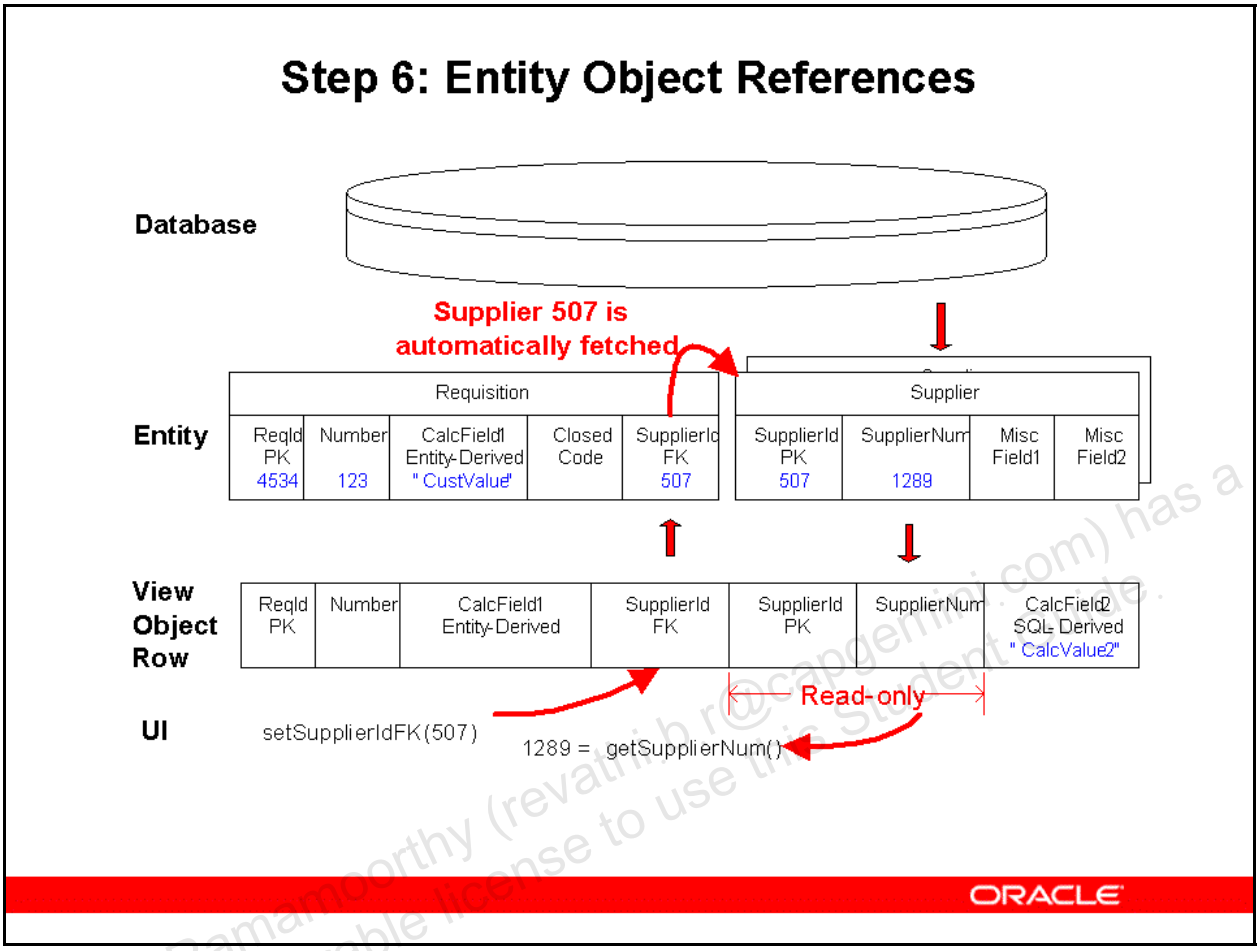
Entity-Derived attributes are useful for calculations owned by the Entity and shared by multiple VOs. But, entity-derived attributes are only calculated when demanded. When is that? When the VO attempts to bind the entity-derived attribute into the UI for rendering to the browser.

Step 5: Entity Object Fault-in



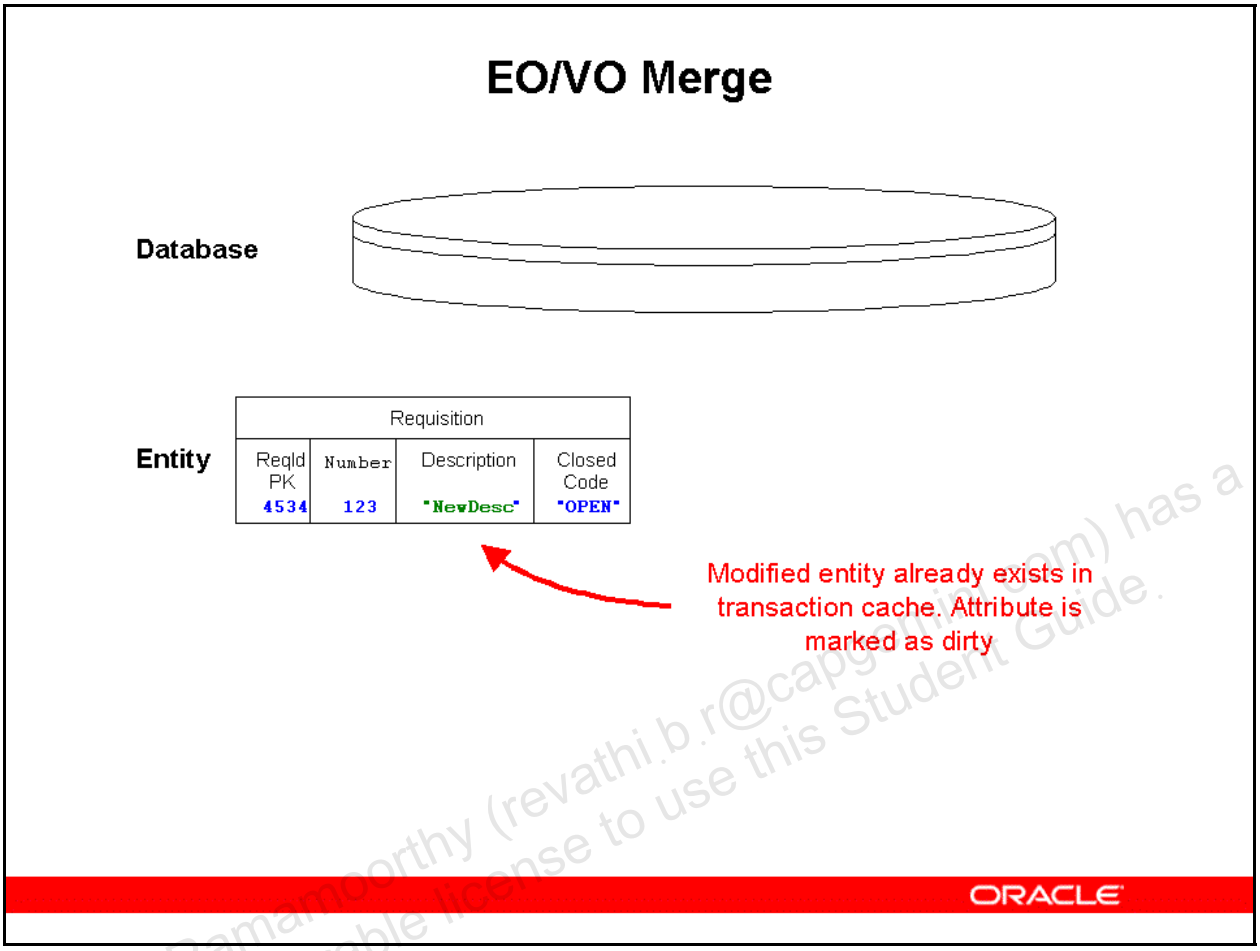
Missing attributes are faulted-in on demand during EO validation.

Step 6: Entity Object References



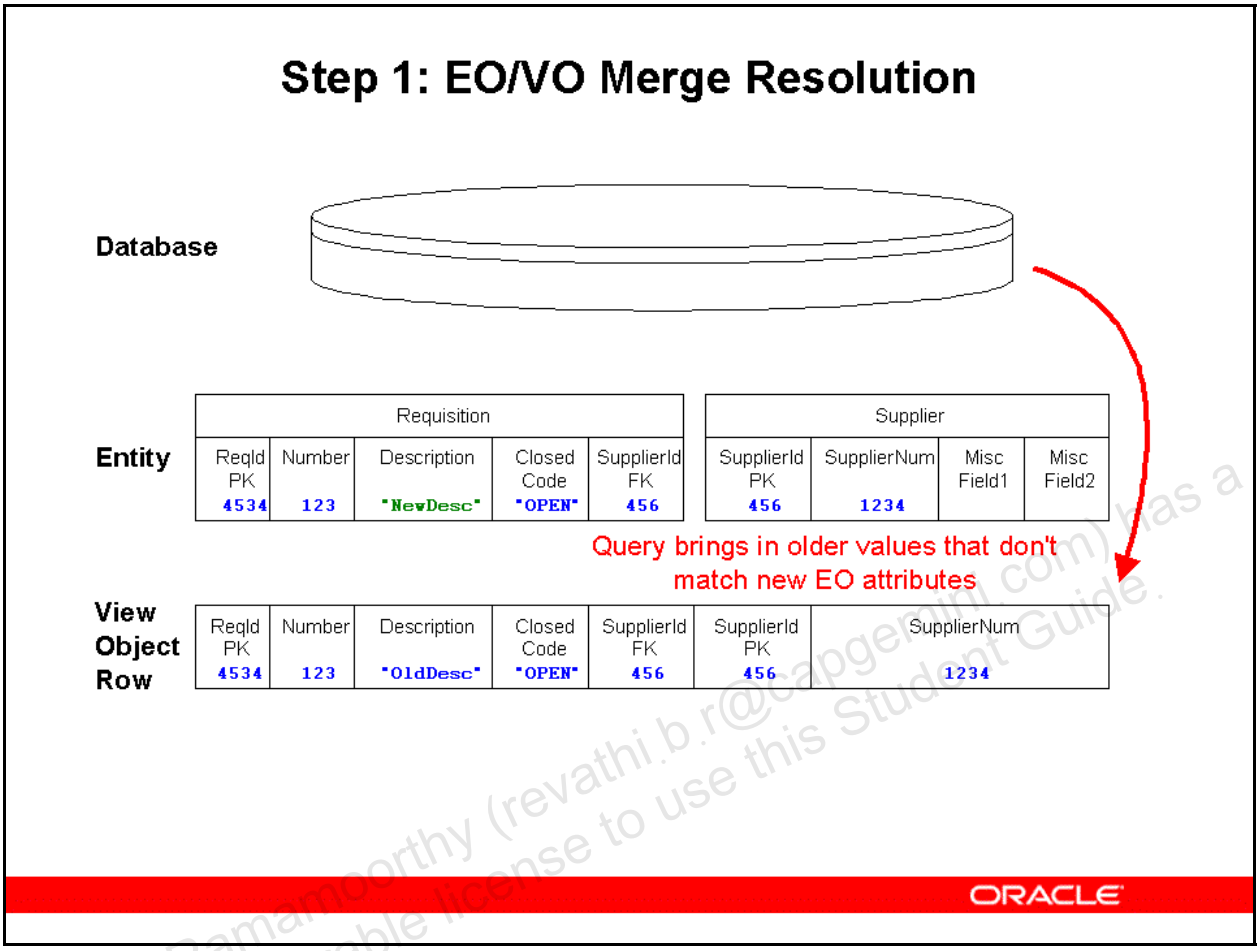
Here is the scenario. Assume you have an Association Object that connects the two Entity Objects. Initially, the VO queries the data, transfers it to the EO. Then, during interaction with the end-user, they assign a different SupplierId. This requires that a new Entity Object row in the cache that currently is not there. In these scenarios, the VO marks all reference EO (Supplier) attributes as read-only. And, you cannot change the primary key of the reference EO. You must change the foreign key attributes, which will result in the fault-in of the EO-layer data from the database.

EO/VO Merge



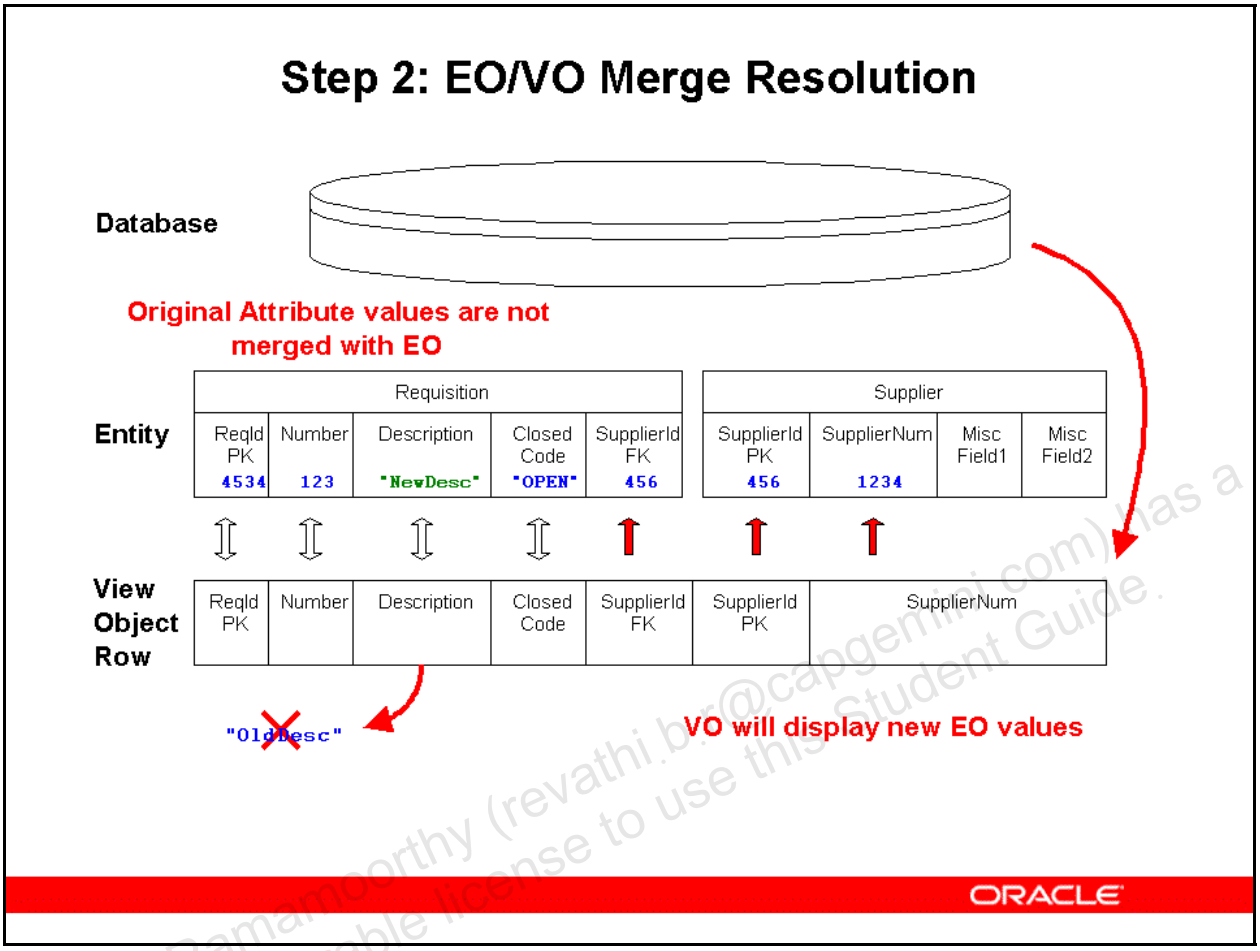
An EO/VO merge occurs whenever an EO already exists in memory, containing original or modified values, and a new VO is queried containing data for the EO. The EO is the transaction cache, and the values need to be resolved before the VO can properly bind the correct value to the UI.

Step 1: EO/VO Merge Resolution



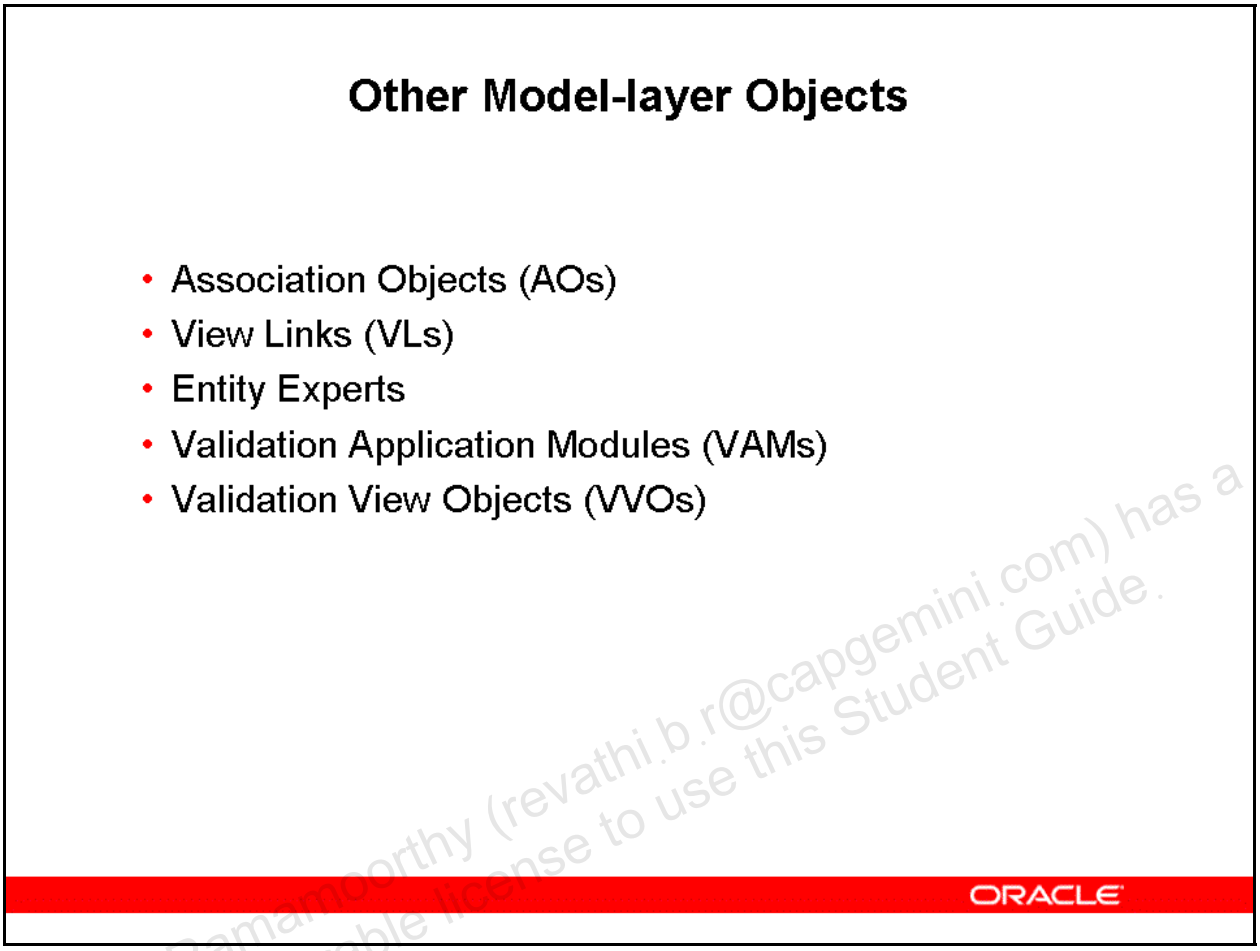
Note: Other users do not see the changed value until it is saved to the database. The EO/VO merge only applies to data within that user's current session.

Step 2: EO/VO Merge Resolution



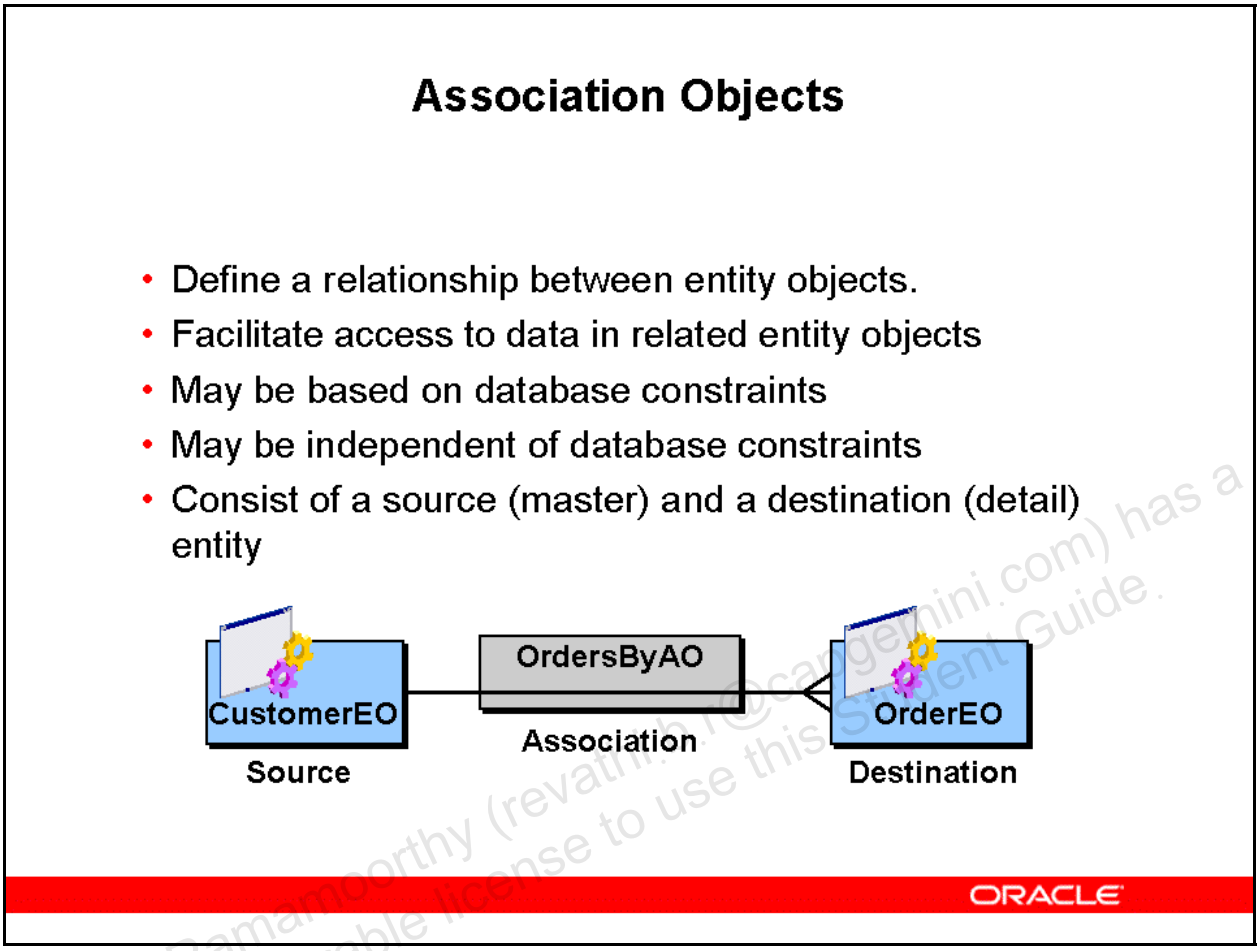
After query and the EO/VO merge, the VO will then bind the correct value to the UI during rendering.

Other Model-layer Objects



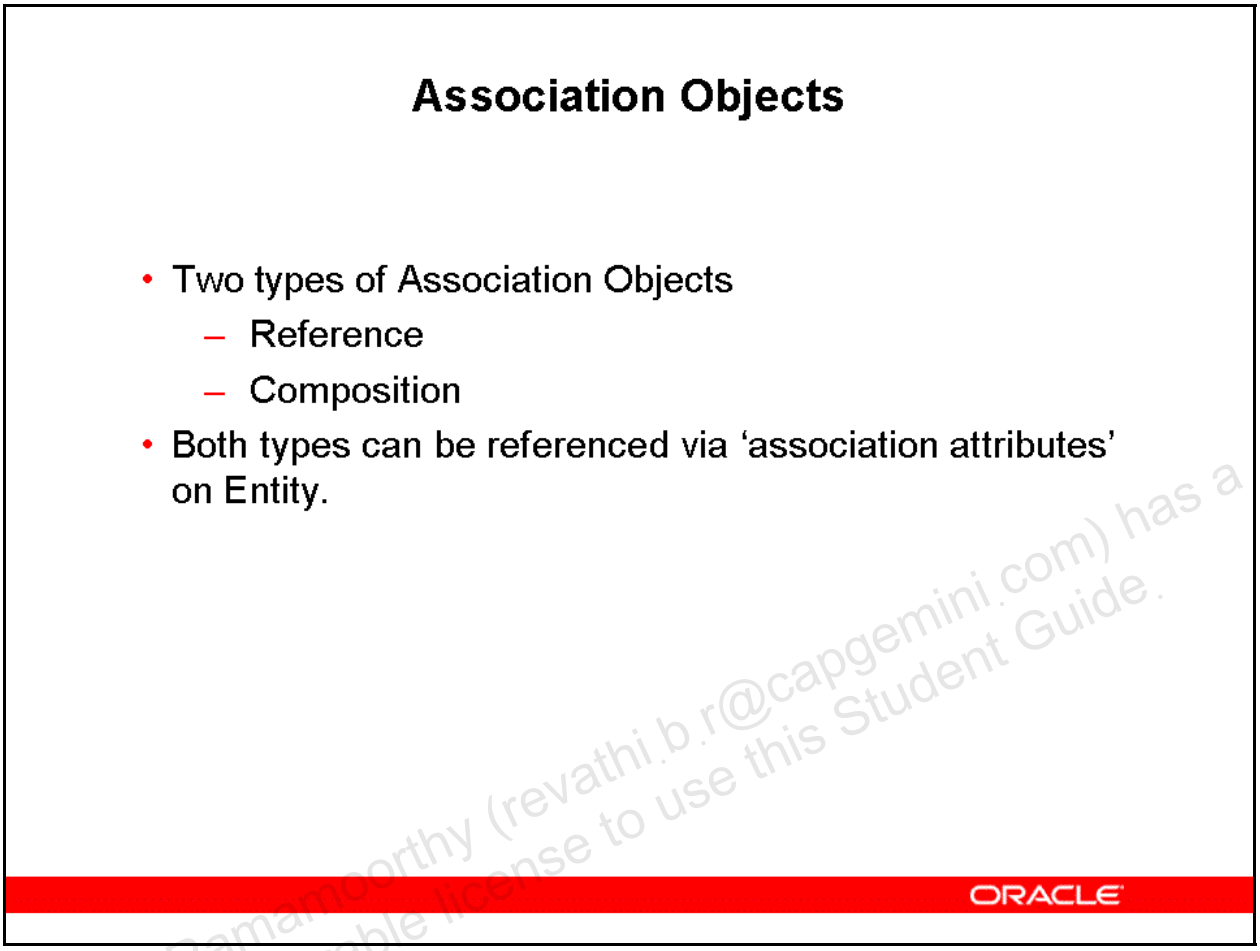
We will cover Association Objects (AOs) and View Links (VL) in this lesson. Entity Experts, Validation Application Modules (VAMs), and Validation View Objects (VVOs) will be defined later in the course. At this point, it is sufficient to simply know that these objects exist.

Association Objects



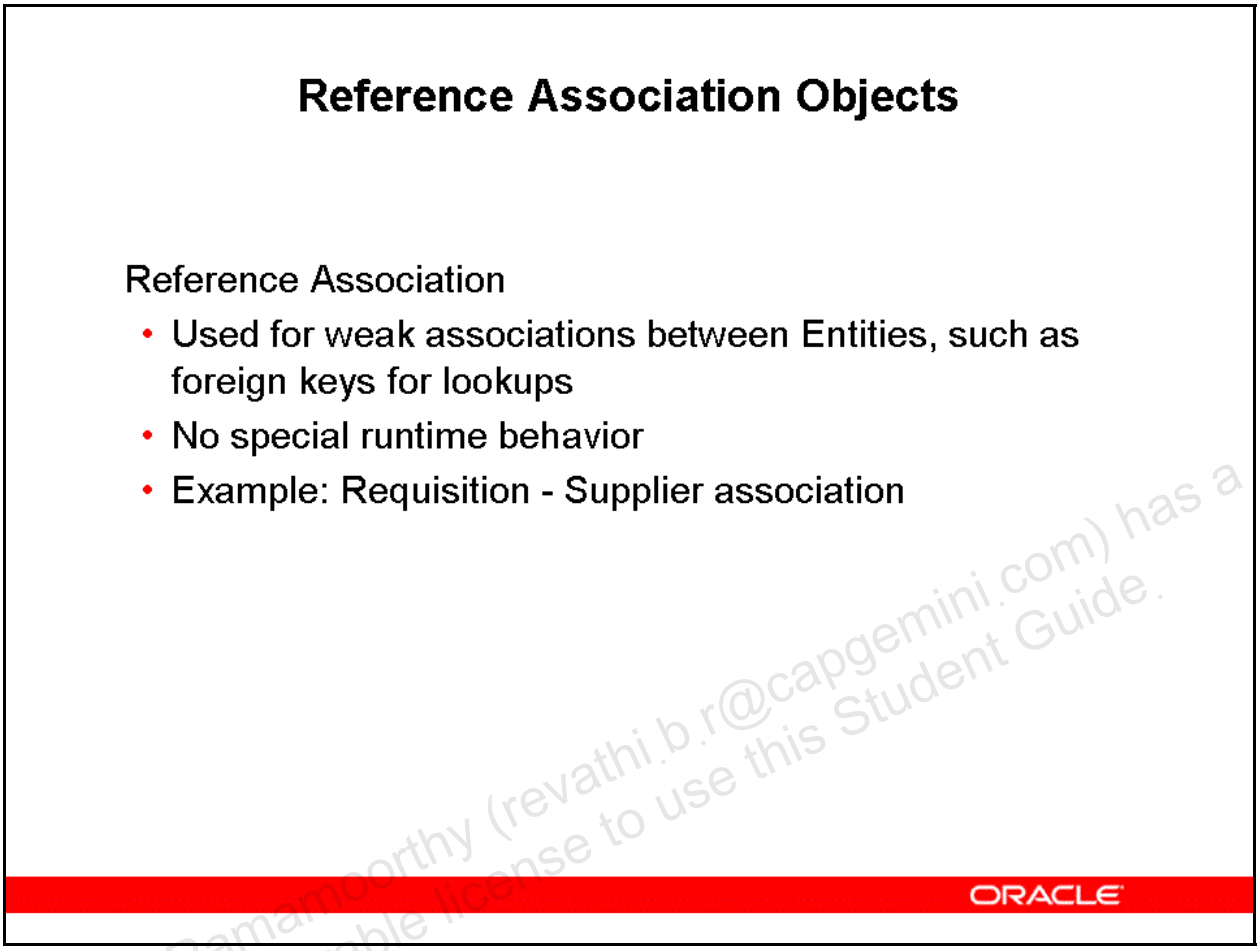
Association Objects are the way to implement foreign keys at the EO-layer.

Association Objects

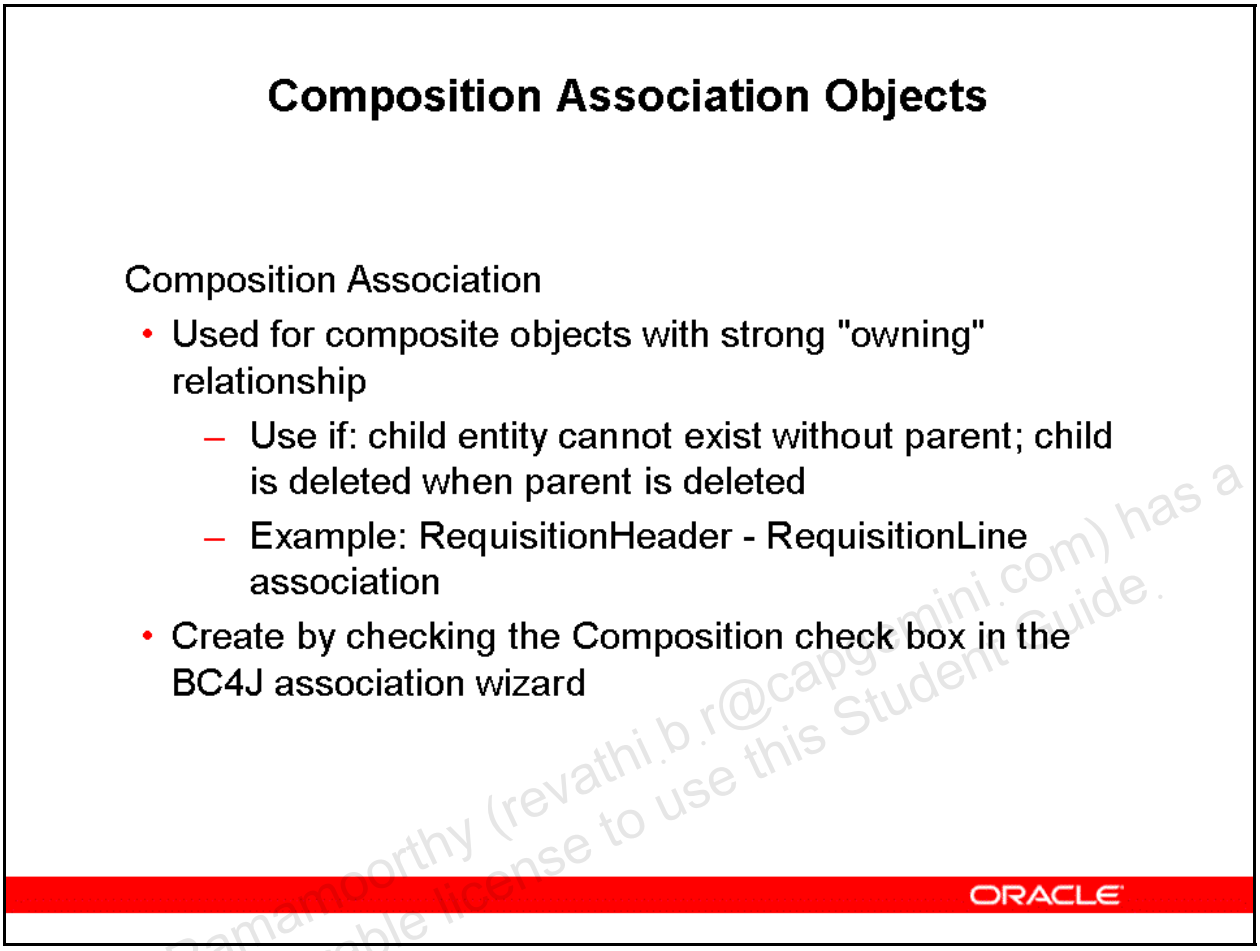


Composition - A strong association where the source entity object owns the destination entity object. In other words, the destination cannot exist independent of its source. For example, a purchase order header is comprised of purchase order lines, which have no meaning or life span outside the context of their header.

Reference - A weak association where the source entity object only references the destination entity object. For example, a purchase order header references a supplier, but the supplier can still exist regardless of whether a purchase order references it or not.



This is the default behavior for AOs.



This has to be specifically indicated in the EO.

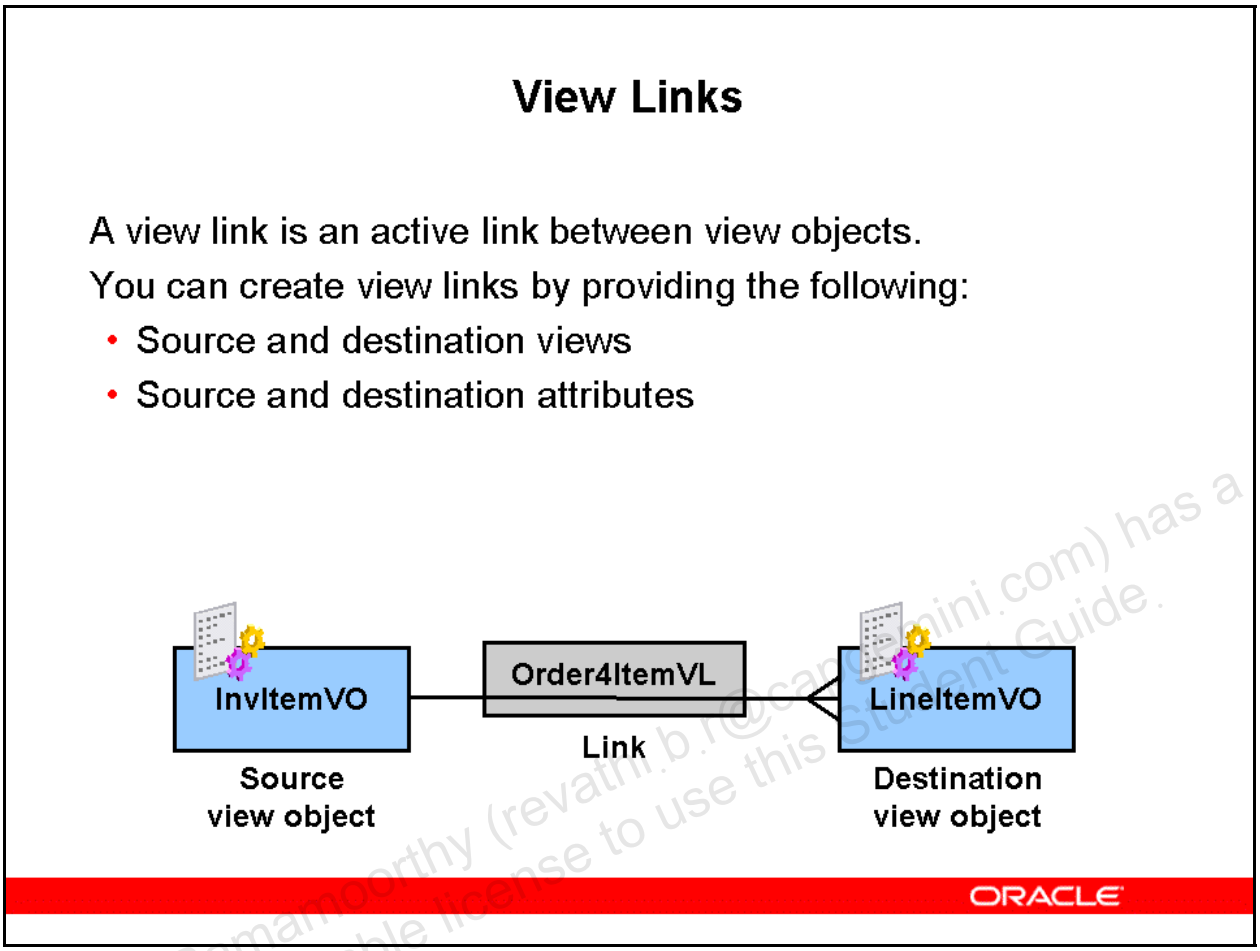
Composition Association Object Behavior

Behavior of Composition Association

- When child is dirtied, parent is dirtied automatically
- When child is locked, parent is locked first automatically
- Parent is brought into memory automatically if not already in memory
- Validation order is child first, parent second - parent has final veto power on child modifications
- Insert/Update order is parent first, child second
- Delete order is child first, parent second

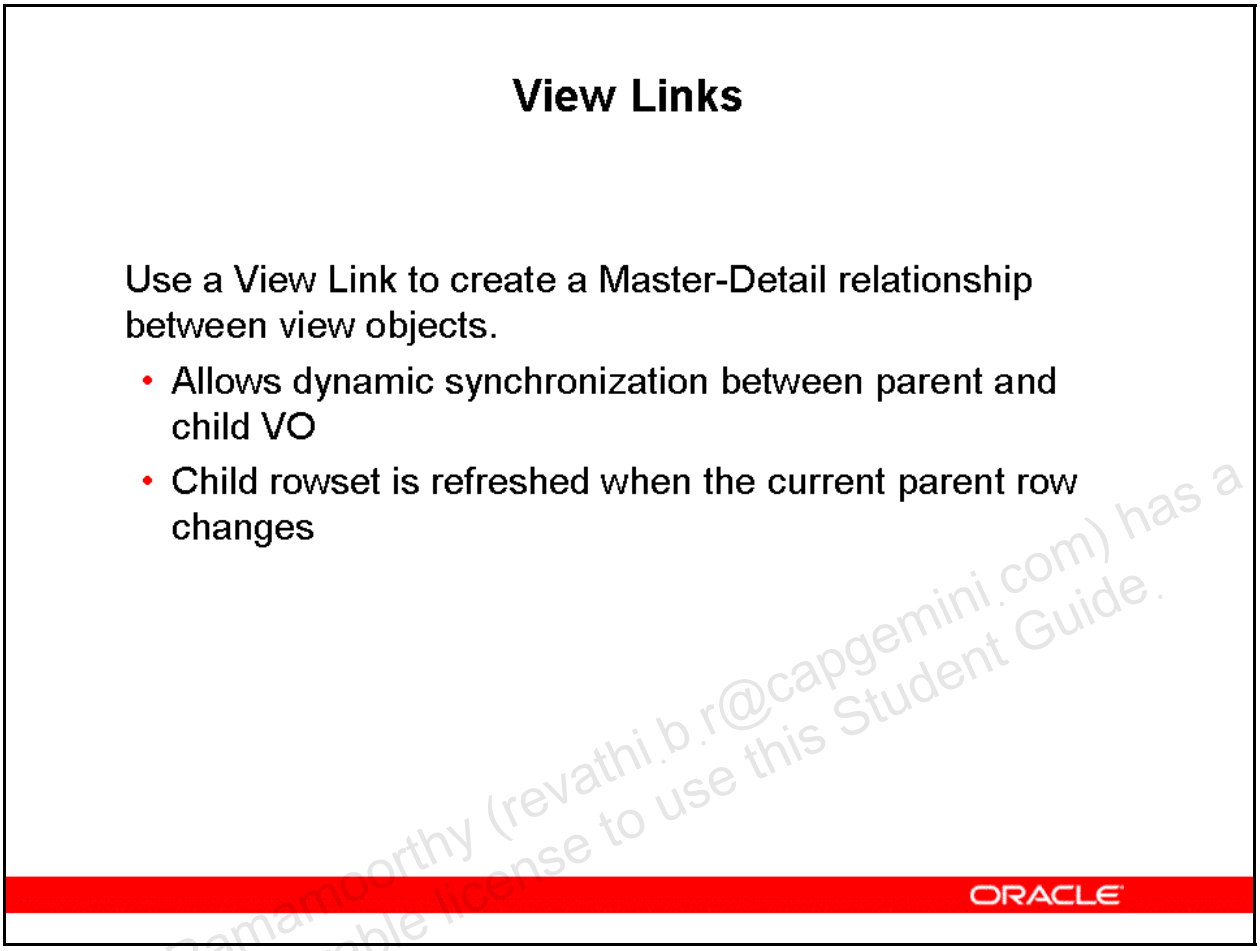
ORACLE

View Links



Note: While it is possible to create View Links (VLs), they are seldom-used objects. Why? If you application is doing any data manipulation, you will be using Association Objects (AOs) instead of VLs. If you are doing read-only, it is far easier to either put the join into the SELECT statement that defines the VO. Or, create a database view that has your join, and base you VO on that database view.

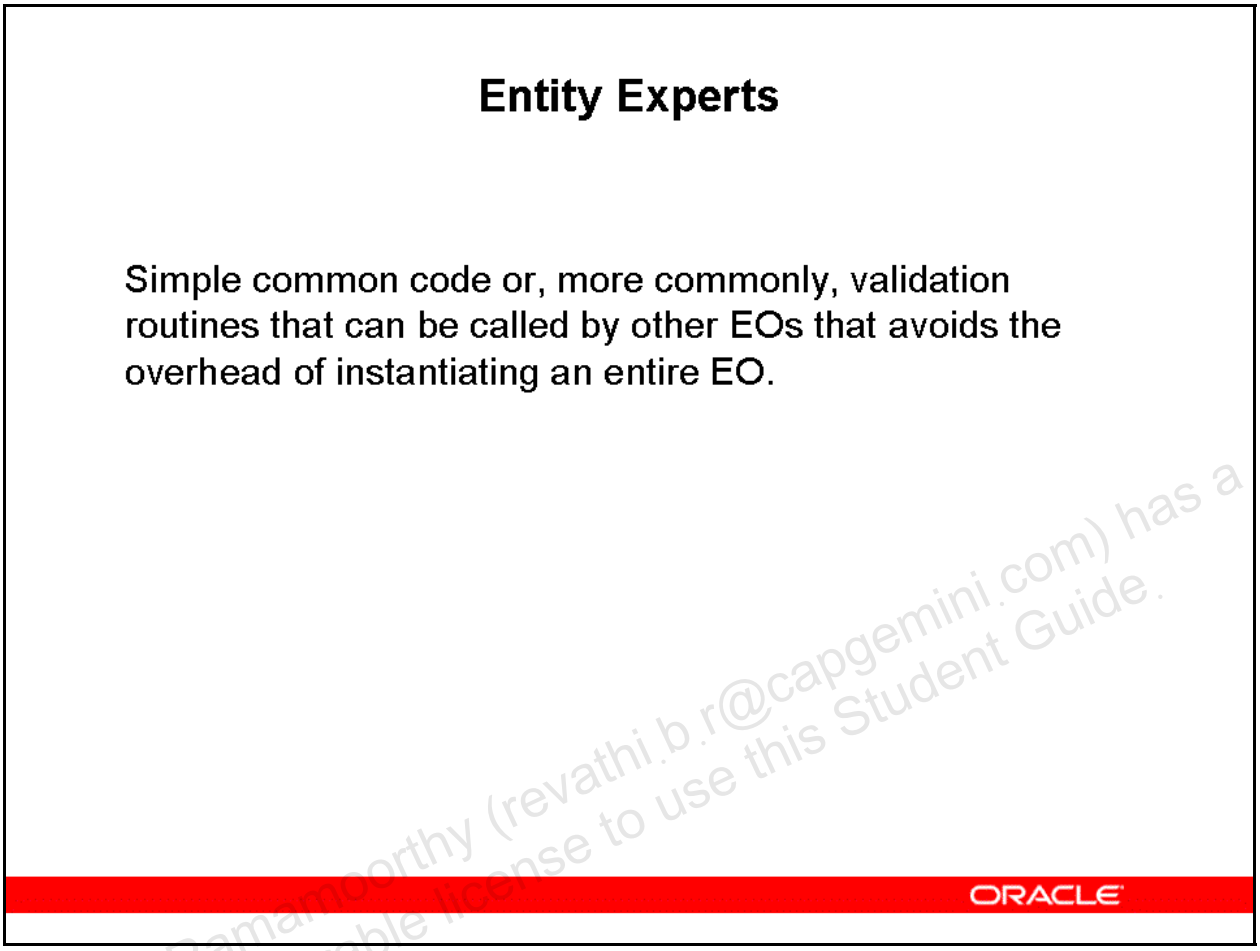
View Links



View Links

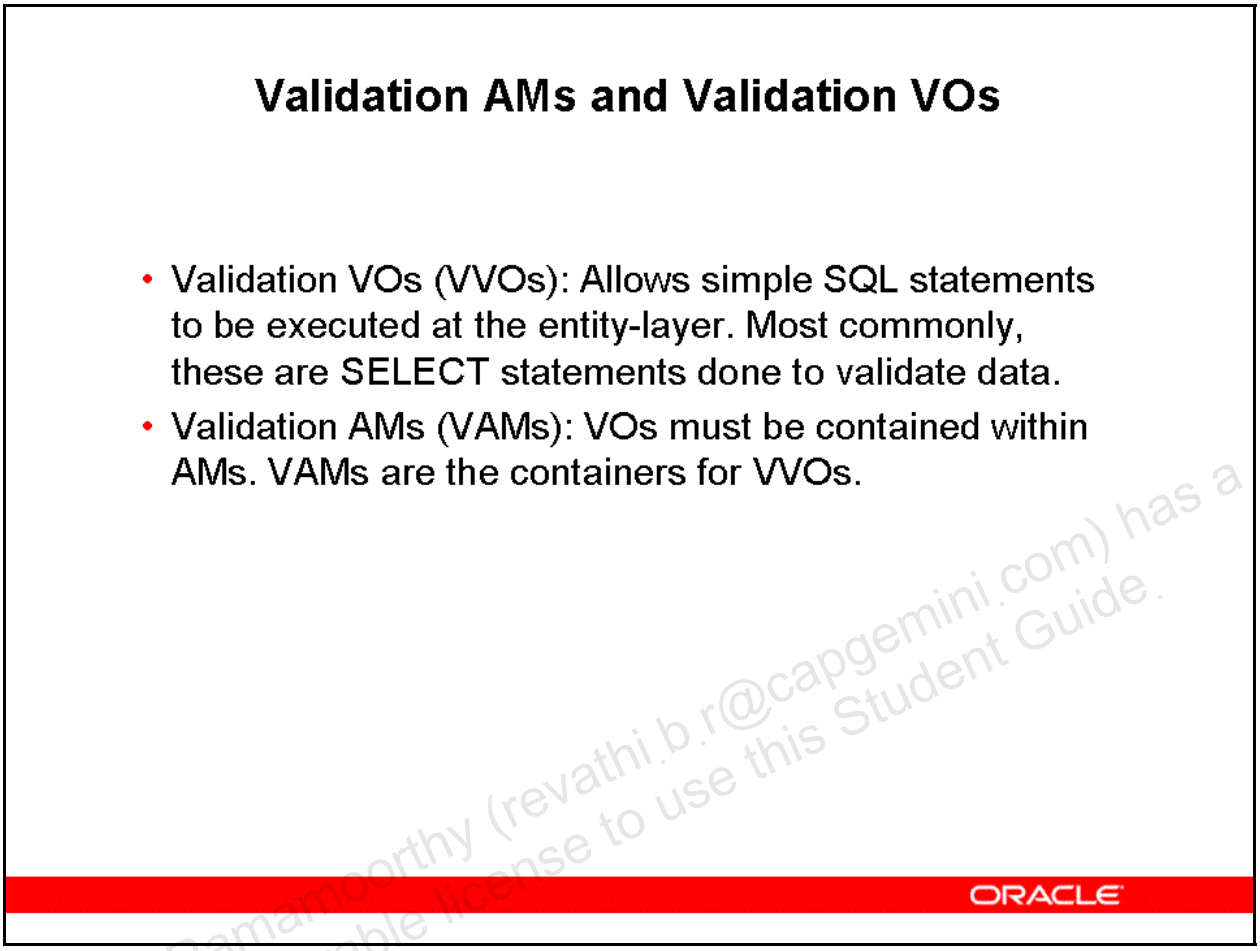
Use view links sparingly. See the OA Framework Developer's Guide for more information.

Entity Experts



The entity expert is a singleton defined to be a special affiliate of a business object (either the top entity object in a composition, or a standalone entity object). It includes common code called by the owning business object, or simple validation routines called by other entity objects that don't want the cost of instantiating the entity object itself. For example, a PurchaseOrderHeaderEOImpl class doesn't want to instantiate a whole SupplierEOImpl class just to find out if the supplierId foreign key it's about to set is valid. Instead, it calls an isSupplierIdValue(Number supplierId) method on the supplier's entity expert singleton -- a much lighter weight operation.

Validation AMs and Validation VOs



When you implement business logic in your entity objects, you will frequently find that you need to execute some simple SQL statements, and not just for pure validation purposes. For example, a purchase order header has many lines. Each line is assigned a unique line number. This number is defined as the current maximum line number for the entire purchase order + 1. At runtime, we need to query the database to find out what the maximum line number is for a given purchase order header:

```
SELECT MAX(LINE_NUMBER) FROM FWK_TBX_PO_LINES WHERE HEADER_ID
  = :1;
```

Whenever you need to execute SQL like this, you can create a view object dynamically from a SQL statement, or you can predefine a declarative view object for it. That being said, OA Framework Coding Standards require that you use the declarative strategy in this case since it is more performant: a view object is cached in its respective application module, which allows entity object code to reuse it (and the underlying JDBC prepared statement) by simply rebinding and re-execute the query. This is an important performance benefit since validation routines are called repeatedly.

Predefined view objects must be assigned to an application module so that they can be accessed at runtime. In other words, view objects do not exist outside the context of an application module.

Since entity objects (and their associated validation view objects) can be shared by multiple UI clients (and the root application modules should be considered UI-specific), it is not appropriate to use the root application module for a particular page to hold your validation view objects. Instead, to group these utility view objects into meaningful, reusable units, create a validation application module per business object to hold them. A business object is defined the top-level entity object in a composition, or a single entity object if it stands alone.

Summary

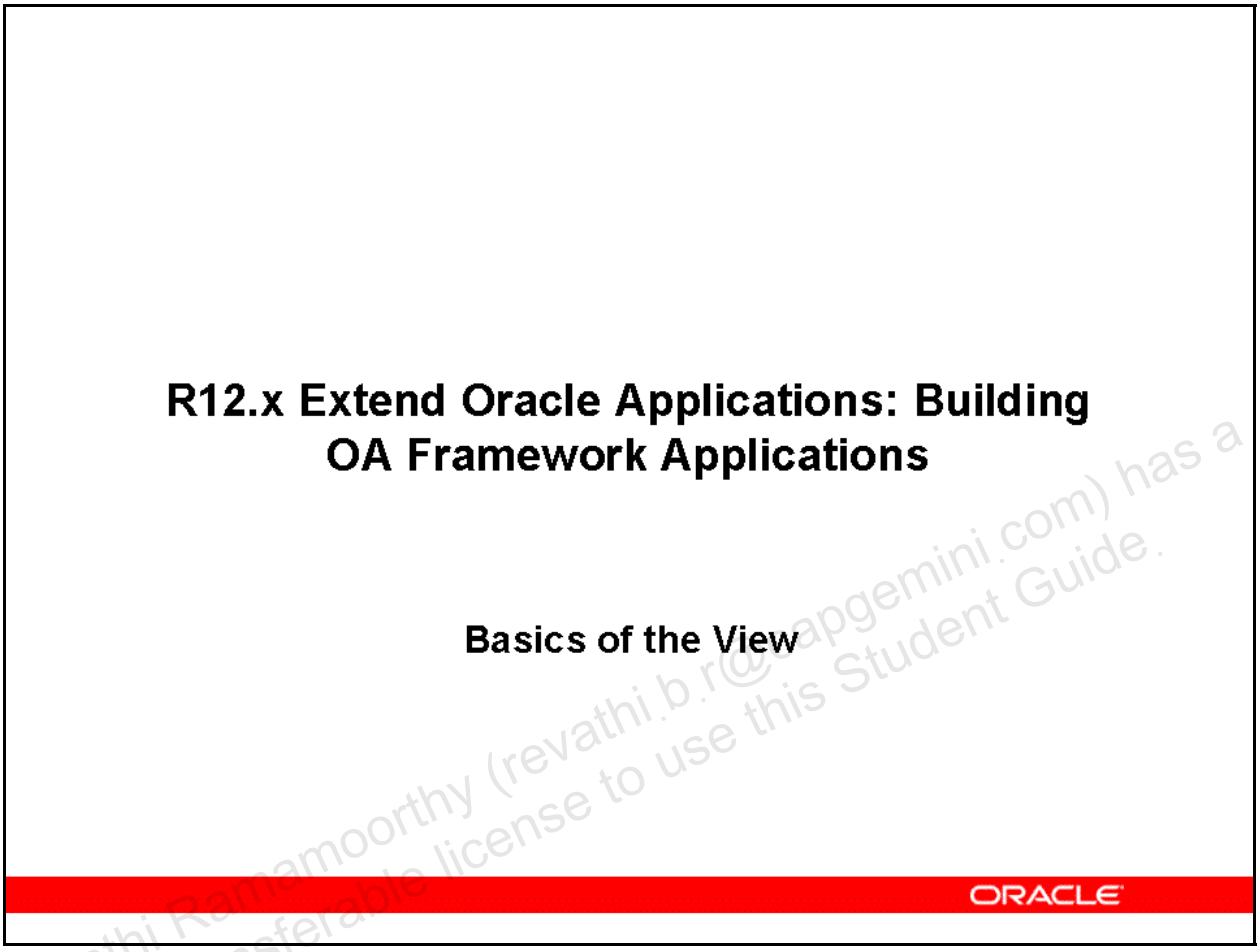
In this lesson, you should have learned how to:

- Discuss BC4J and the Model layer of MVC.
- Discuss Application Modules (AMs).
- Discuss Entity Objects (EOs).
- Discuss View Objects (VOs).
- Discuss how BC4J interacts with the database.
- Discuss other Model-layer components.

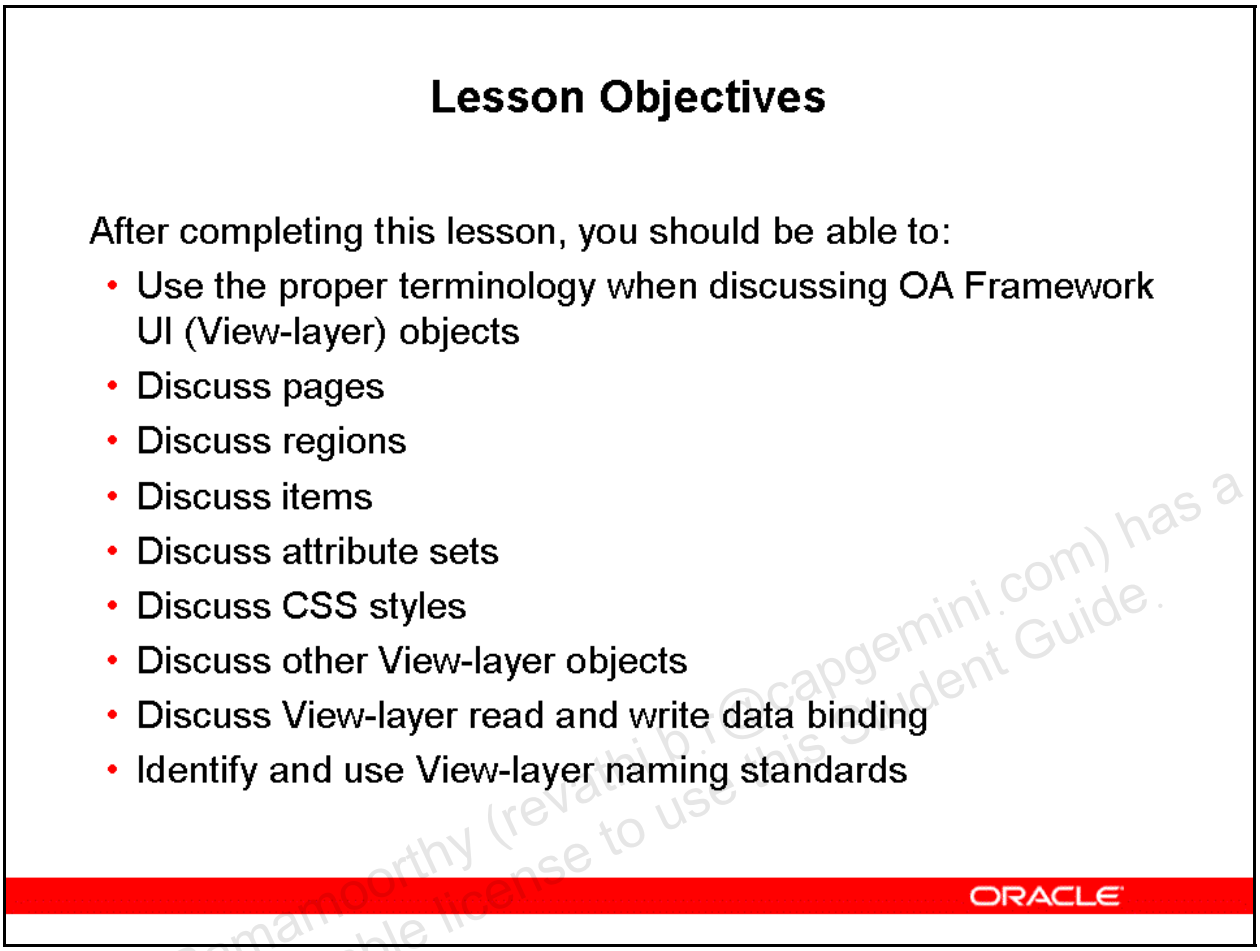
ORACLE

Basics of the View

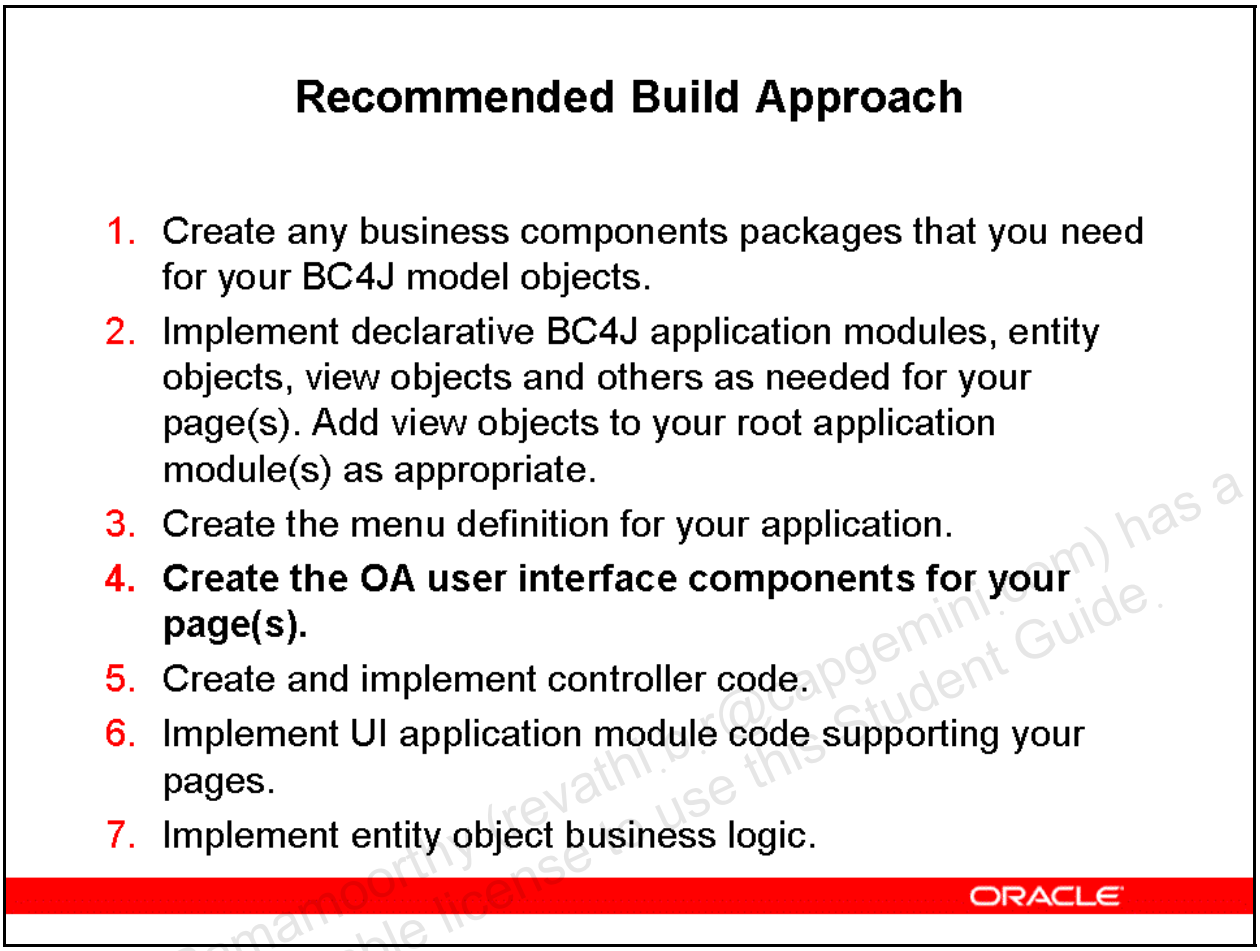
Chapter 4



Lesson Objectives

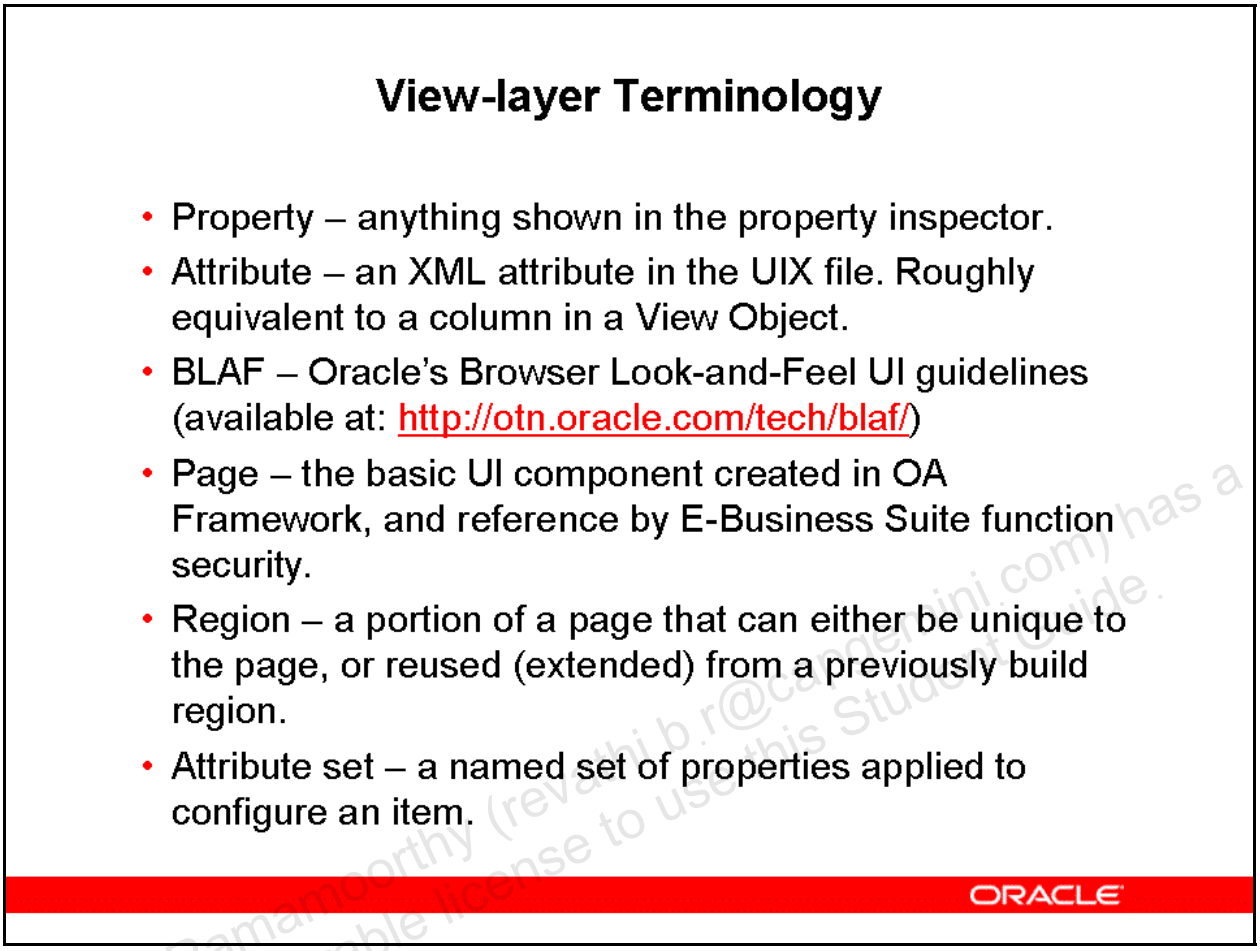


Recommended Build Approach



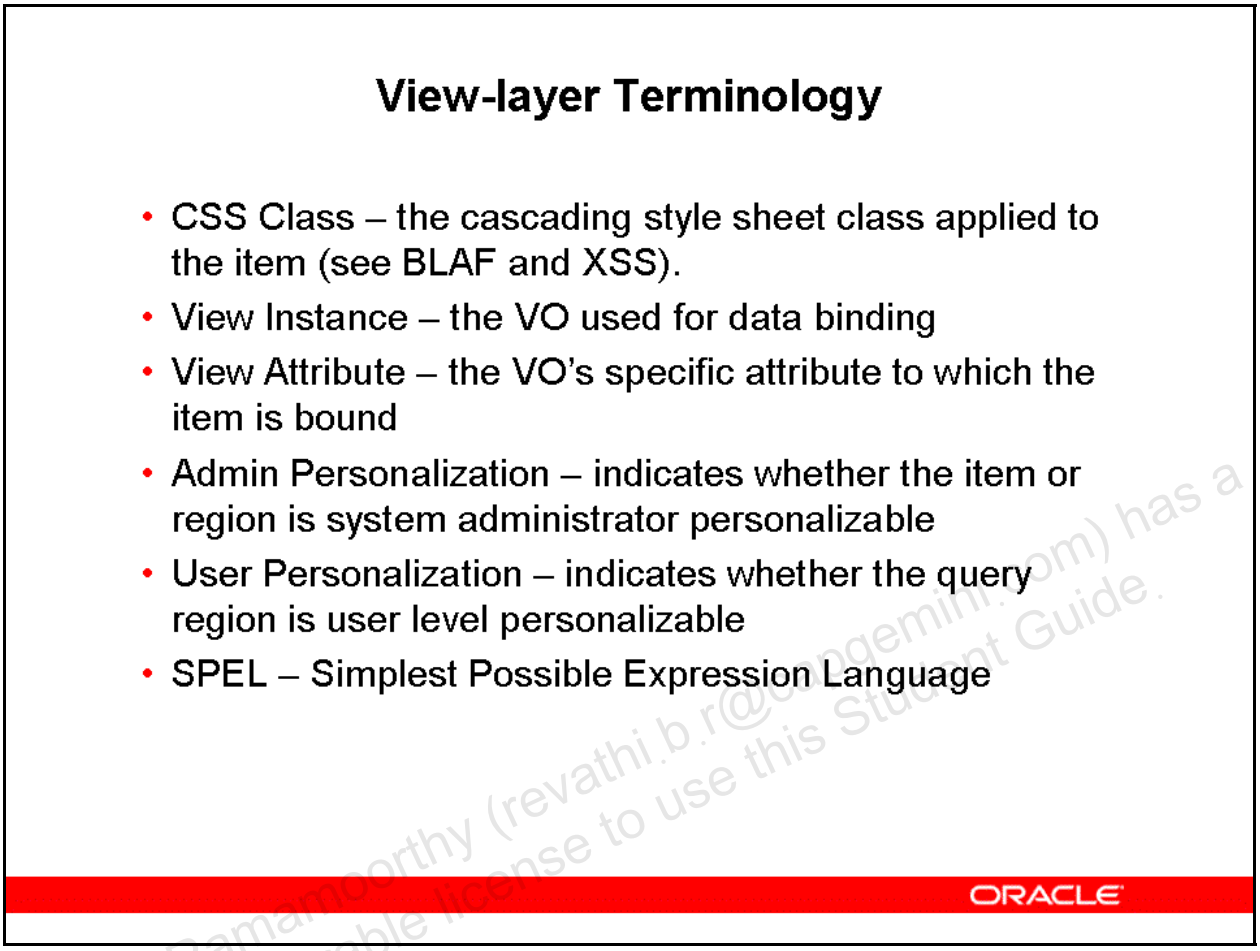
Steps 4 deals directly with View layer components.

View-layer Terminology



Be careful with terminology, and always try to understand the context. As an example, a property in one context may or may not mean the same thing as a property in a differing context. A Property is one of many context sensitive attributes used to define items (Beans) on the page. For each item there is a collections of attributes shown in the property inspector. Setting these attributes define the display properties of the item on the page.

View-layer Terminology



Simplest Possible Expression Language (SPEL)

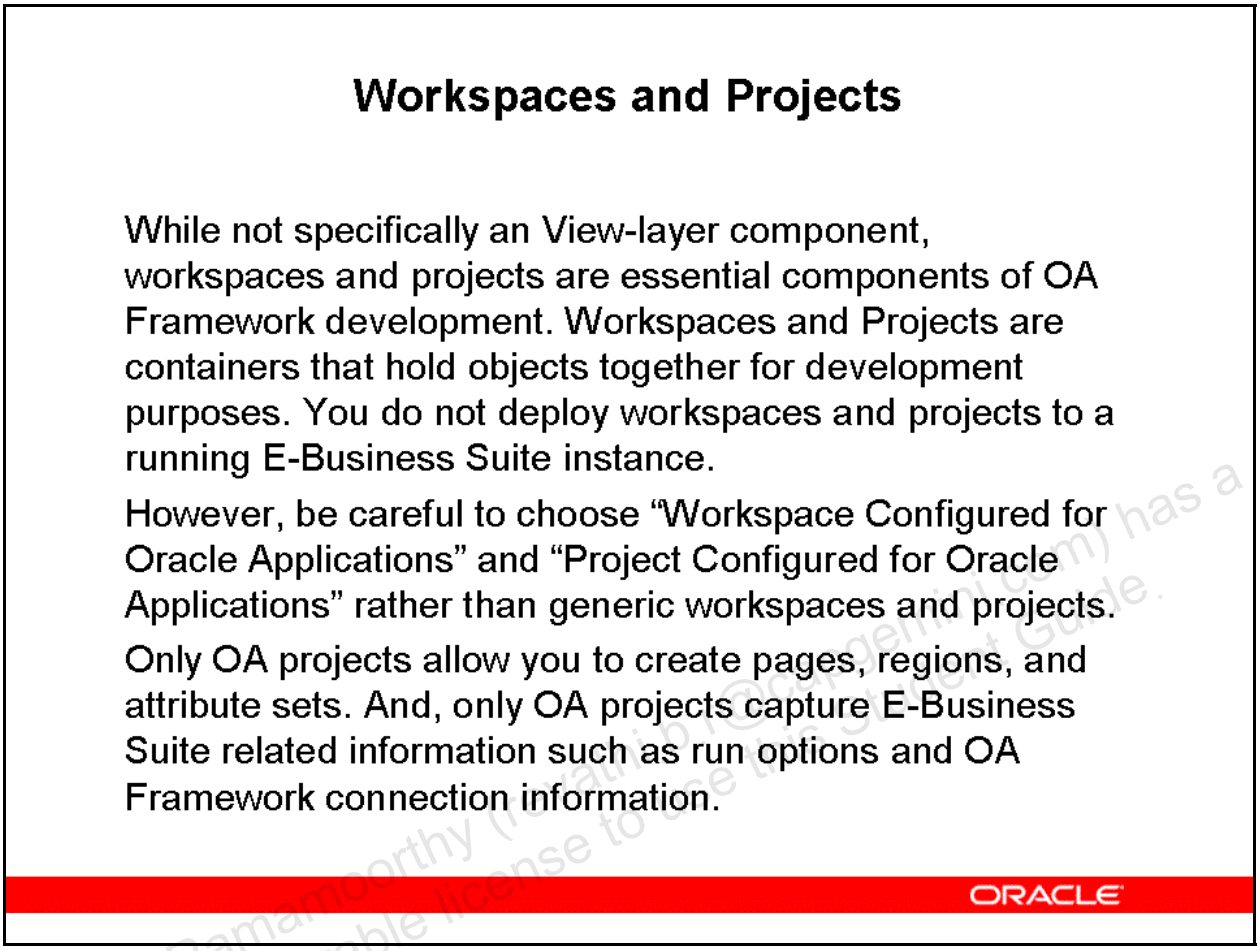
For selected properties, the OA Framework supports the use of SPEL expressions to quickly bind the property to an underlying data source that provides the property's value. For example, you could bind the Rendered property of a button to a view object attribute to ascertain whether it should be hidden or shown when the page renders. The SPEL syntax for this property looks like:

${oa.<ViewInstanceName>.<ViewAttributeName>}

${oa.FunctionSecurity.<FunctionName>}

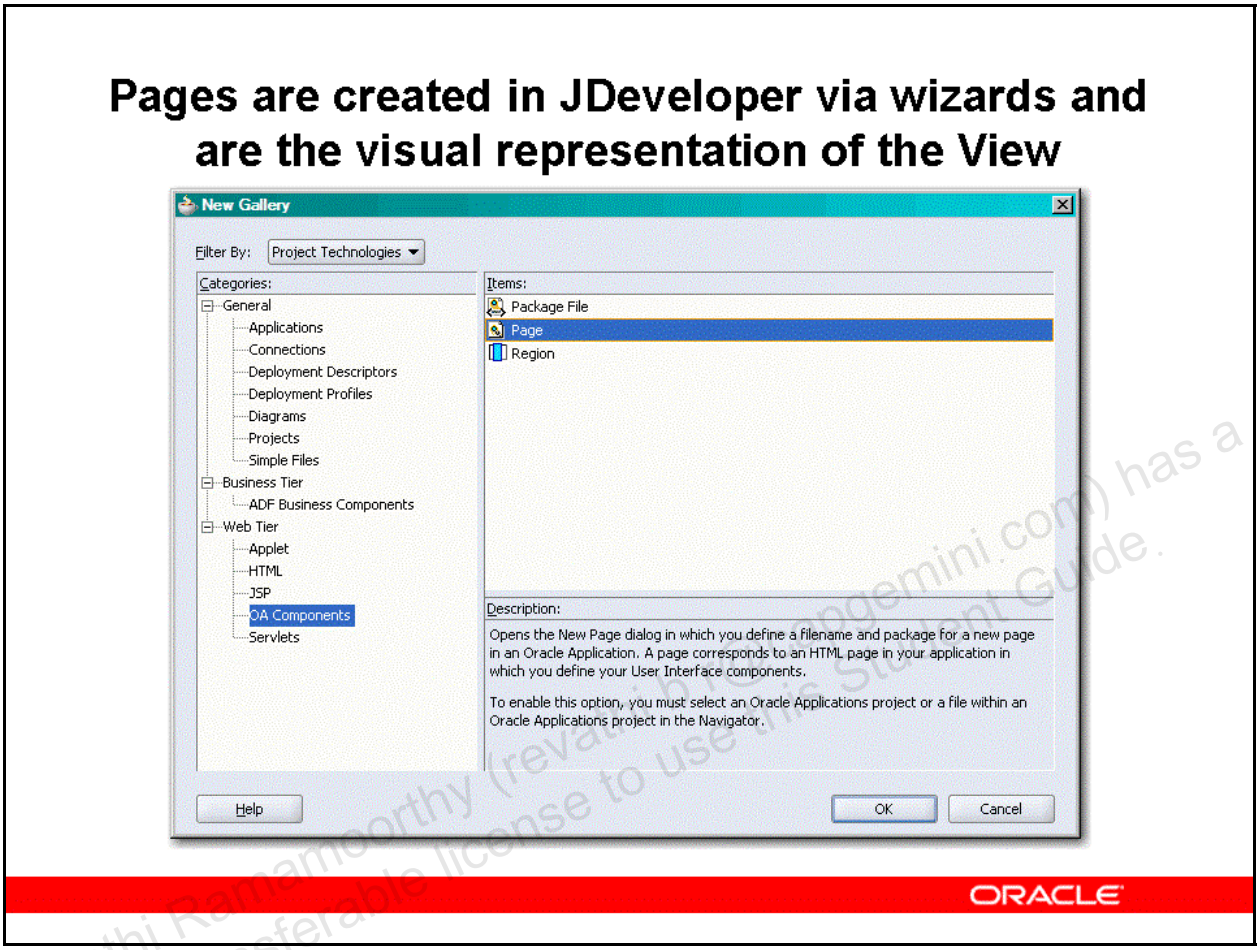
Note: SPEL is defined in JSR-52, which is maintained by the JCP (Java Community Process). You can download more information on SPEL, and the types of expressions possible in SPEL at:
http://jcp.org/aboutJava/communityprocess/final/jsr052/index2.html

Workspaces and Projects



Workspaces and Projects are just containers for JDeveloper, and are not needed to run OA Framework pages.

Pages are created in JDeveloper via wizards and are the visual representation of the View

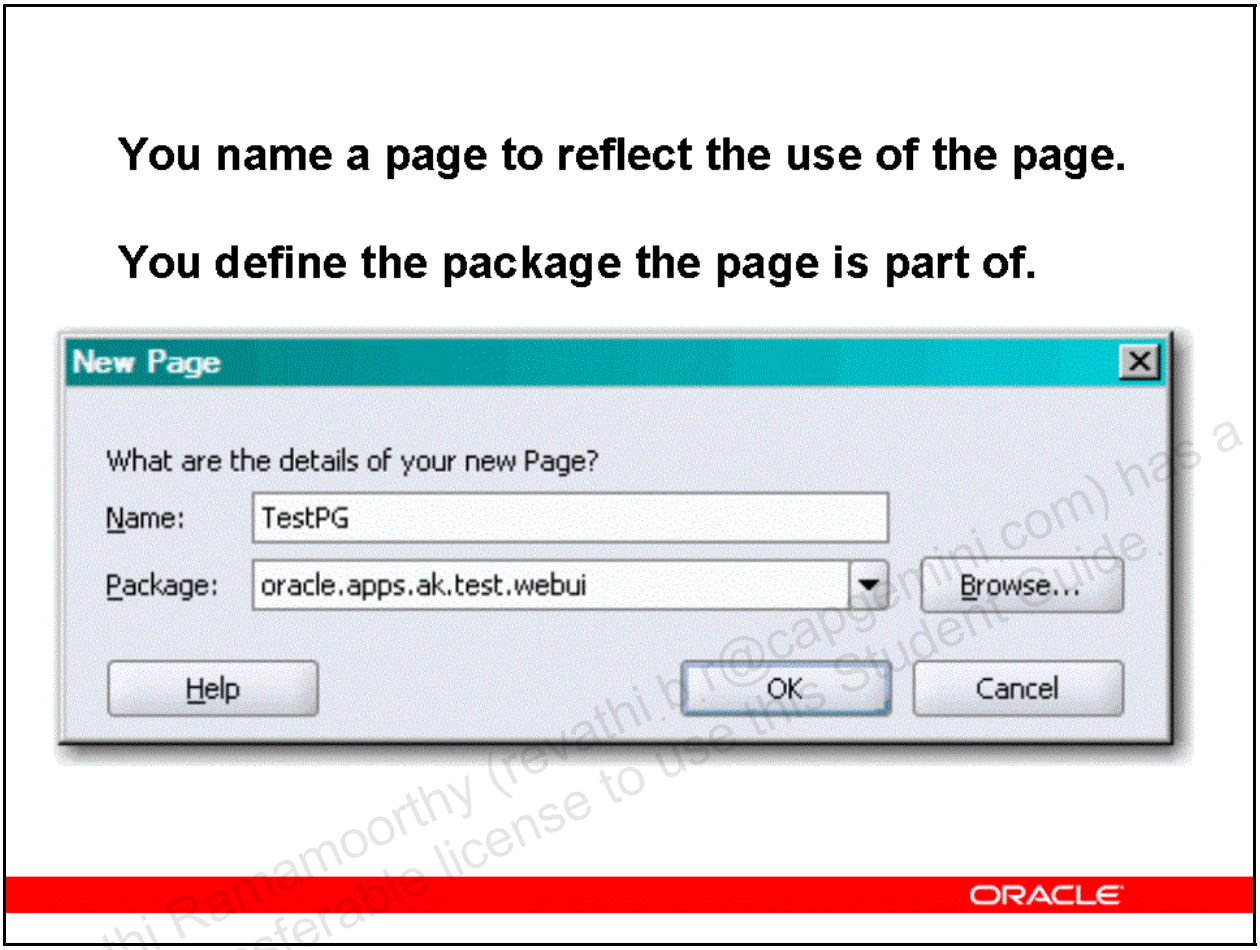


The creation of a page, and the view, starts in JDeveloper.

(Menu) File > New > Web Tier > OA Components > Page

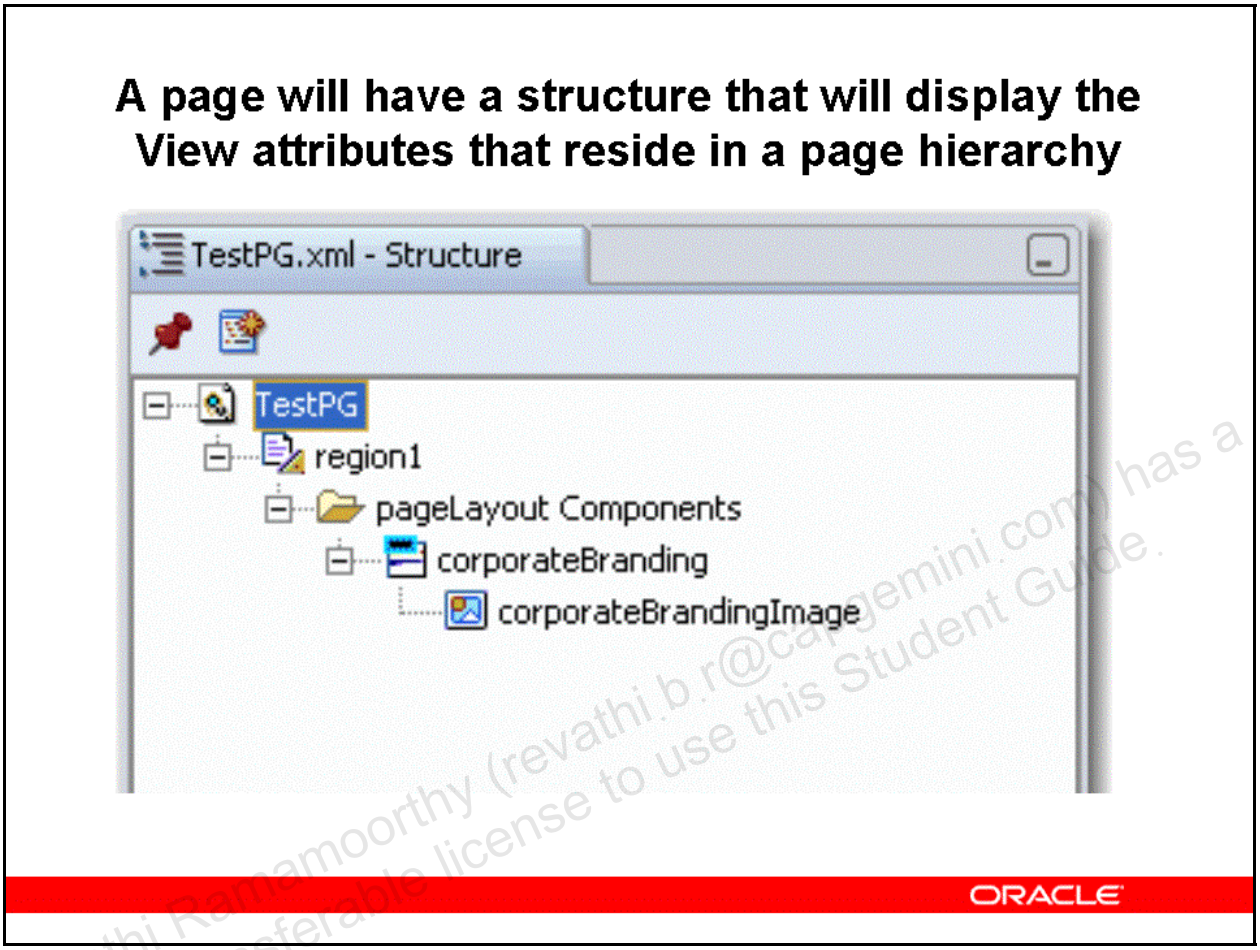
The above screen capture is an overly simplistic overview of the specific steps to create a page. The labs that are to follow will show you the detailed steps and show you the techniques within JDeveloper.

You name a page to reflect the use of the page. You define the package the page is part of.



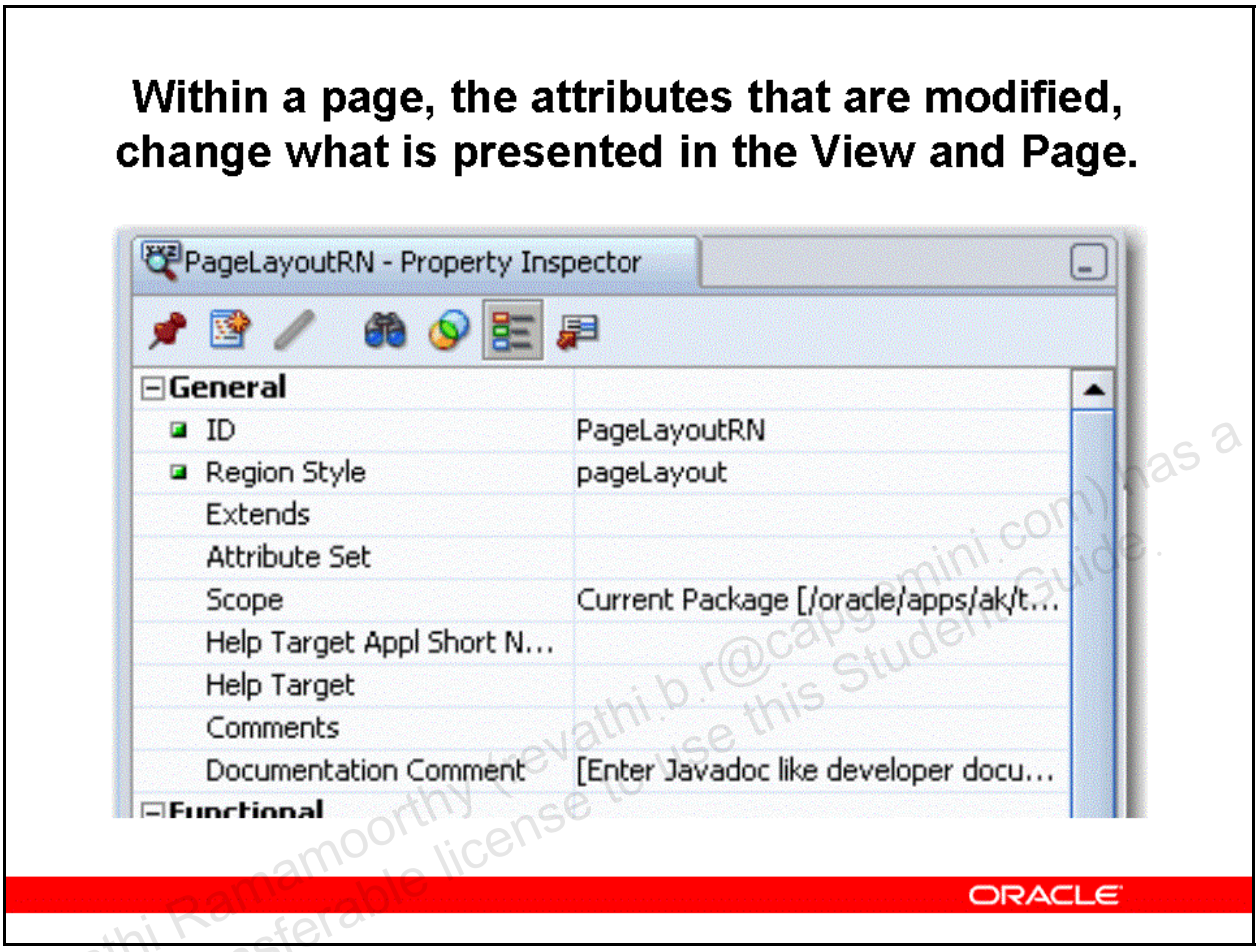
By E-Business Suite Development standards, all pages end in PG. While not mandatory for your pages, it is strongly recommended that you follow all of the E-Business Suite Development naming standards to ensure interoperability and maintainability of your pages. The above screen capture was taken as the Page creation wizard was being used.

A page will have a structure that will display the View attributes that reside in a page hierarchy



The structure of a page is shown in the Structure panel within JDeveloper. By default, the Structure panel is in the lower-left corner of the JDeveloper window. In the screen capture above, some basic components and naming are created automatically as part of the OA Framework plug-in for JDeveloper.

Within a page, the attributes that are modified, change what is presented in the View and Page.



By E-Business Suite Development standards, the first region, currently named region1, should be modified to set it's properties as follows:

ID must be PageLayoutRN. (This is an E-Business Suite Development naming standard.)

Region Style must be pageLayout.

The pageLayout region must be the first region of a page. (The page is created by default to adhere to this.)

AM Definition must be set to a defined AM.

Window Title must be set.

Title must be set.

Autofooter must be set to True.

Form must be set to True.

Note: Both XML and Java are case-sensitive.

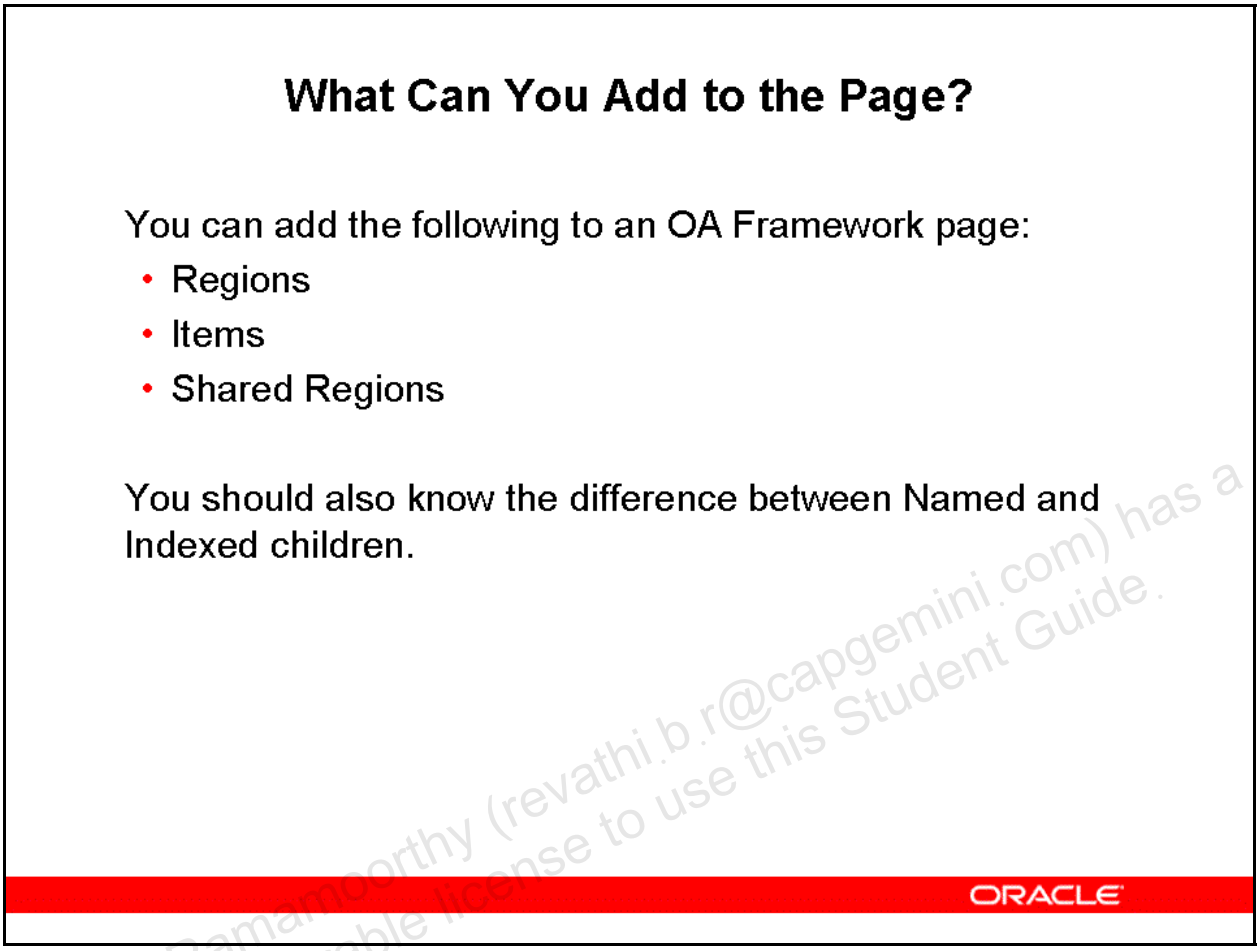
A Page is tested from within JDeveloper to verify the view is rendering what you expect it to.

If you ran a basic page such as was created in the previous screen captures, you would have created nothing more than a blank page. Since OA Framework is built in Java, and one of Java's core principles is inheritance, the objects that you create will inherit certain added components as part of the Framework. For example, in the page, note the following:

- The page has the R12 look and feel, especially the color scheme.
- There is a corporate branding image, the Oracle logo in the upper left.
- There is a header line with options, Home, Logout, and Preferences.
- There is a footer link with options, Home, Logout, and Preferences.
- There is a Privacy Statement link.
- There is a Copyright statement

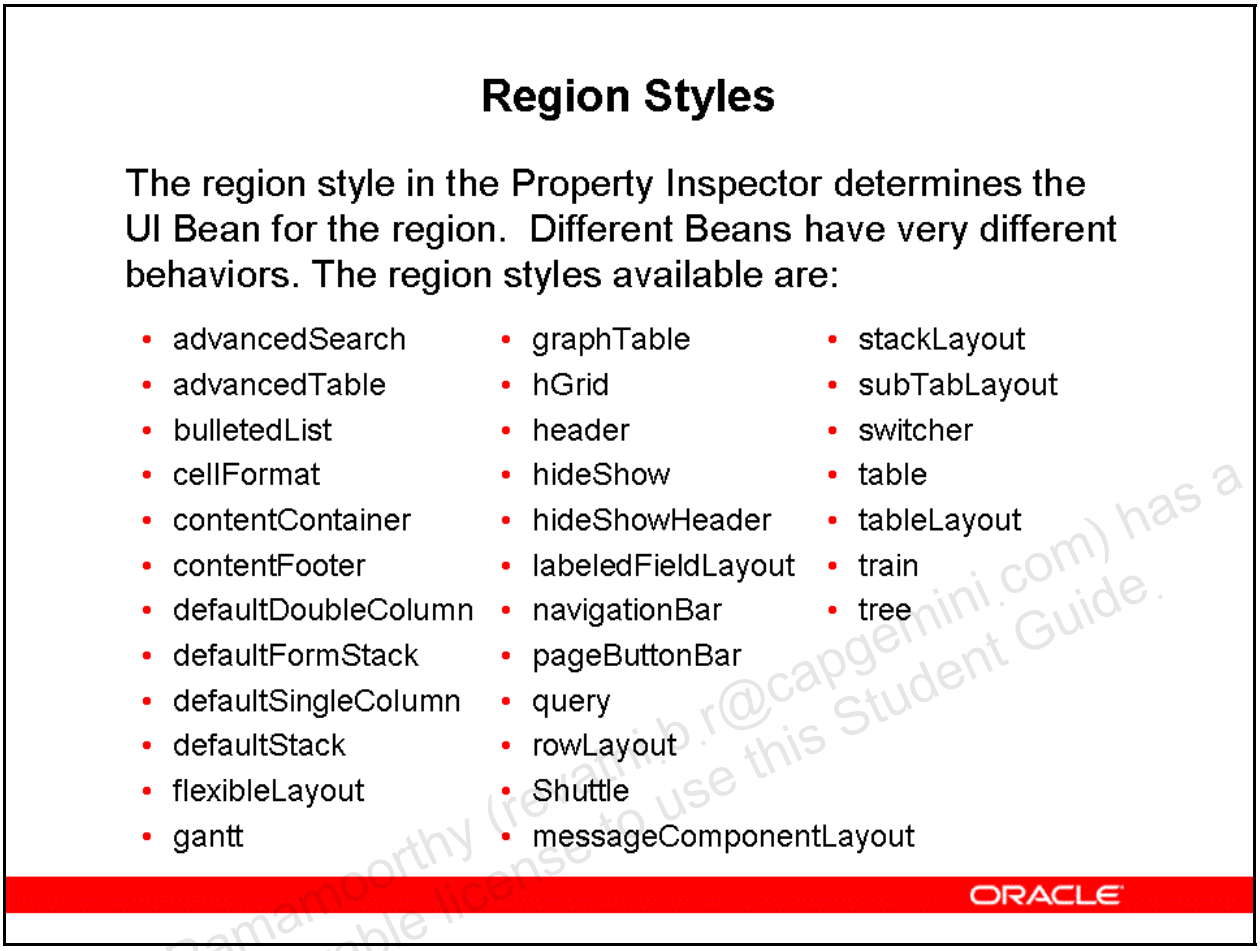
All of these defaults come from the OAF plug-in as part of JDeveloper and through inheritance.

What Can You Add to the Page?



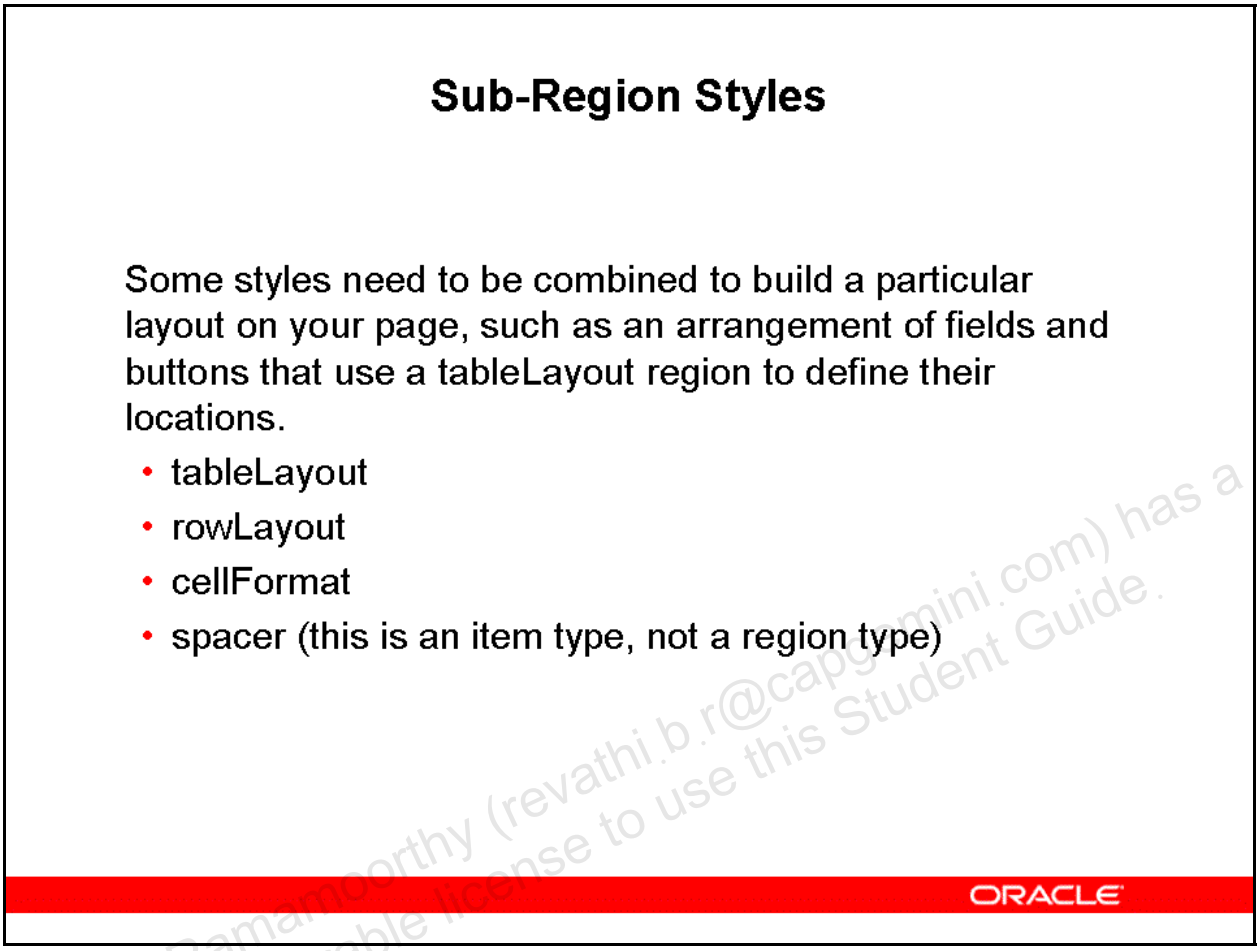
A Shared Region is simply a region with the Extends property set to the name of the Shared Region object.

Region Styles



It is absolutely critical that you begin to familiarize yourself with the OA Framework Developer's Guide (OAFDG). The OAFDG comes in the patch that contains the JDeveloper with OA Extension version that you are using for your development. In Chapter 4: Implementing Specific UI Features of the OAFDG, there is a detailed discussion of each of these region types, along with other UI features. The OAFDG is also available on My Oracle Support – formerly Metalink. Each updated version of JDeveloper will have a specific Developer's guide.

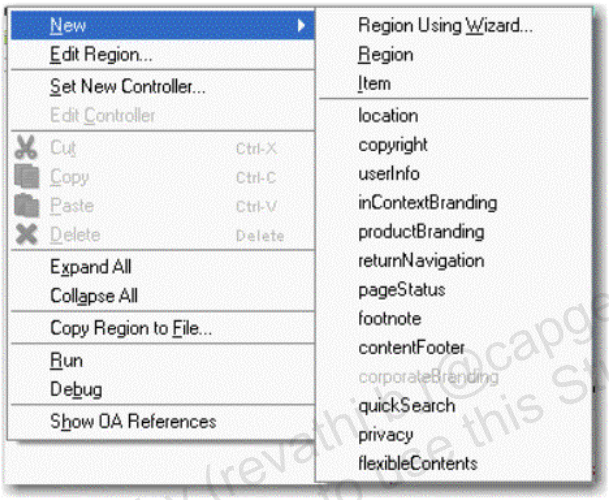
Sub-Region Styles



Named Children vs. Indexed Children

Named Children vs. Indexed Children

- Named children of a Bean have a specific name, specific purpose, and a specific location in the UI.
- Indexed children of a Bean depend on their order under the Bean



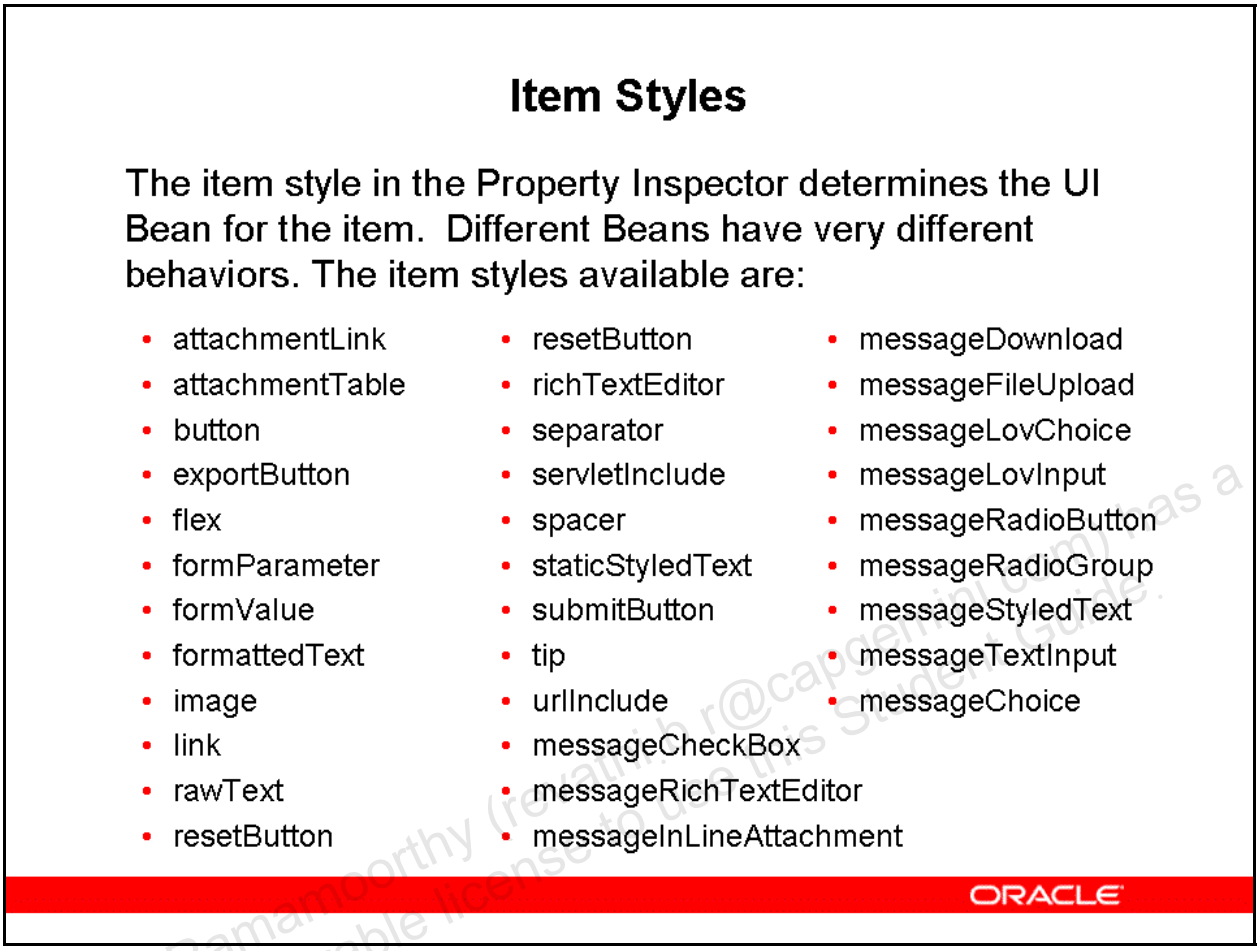
ORACLE

The named children are as follows:

location
copyright
userInfo
inContextBranding
productBranding
returnNavigation
pageStatus
footnote
contentFooter
corporateBranding
quickSearch
privacy
flexibleContents

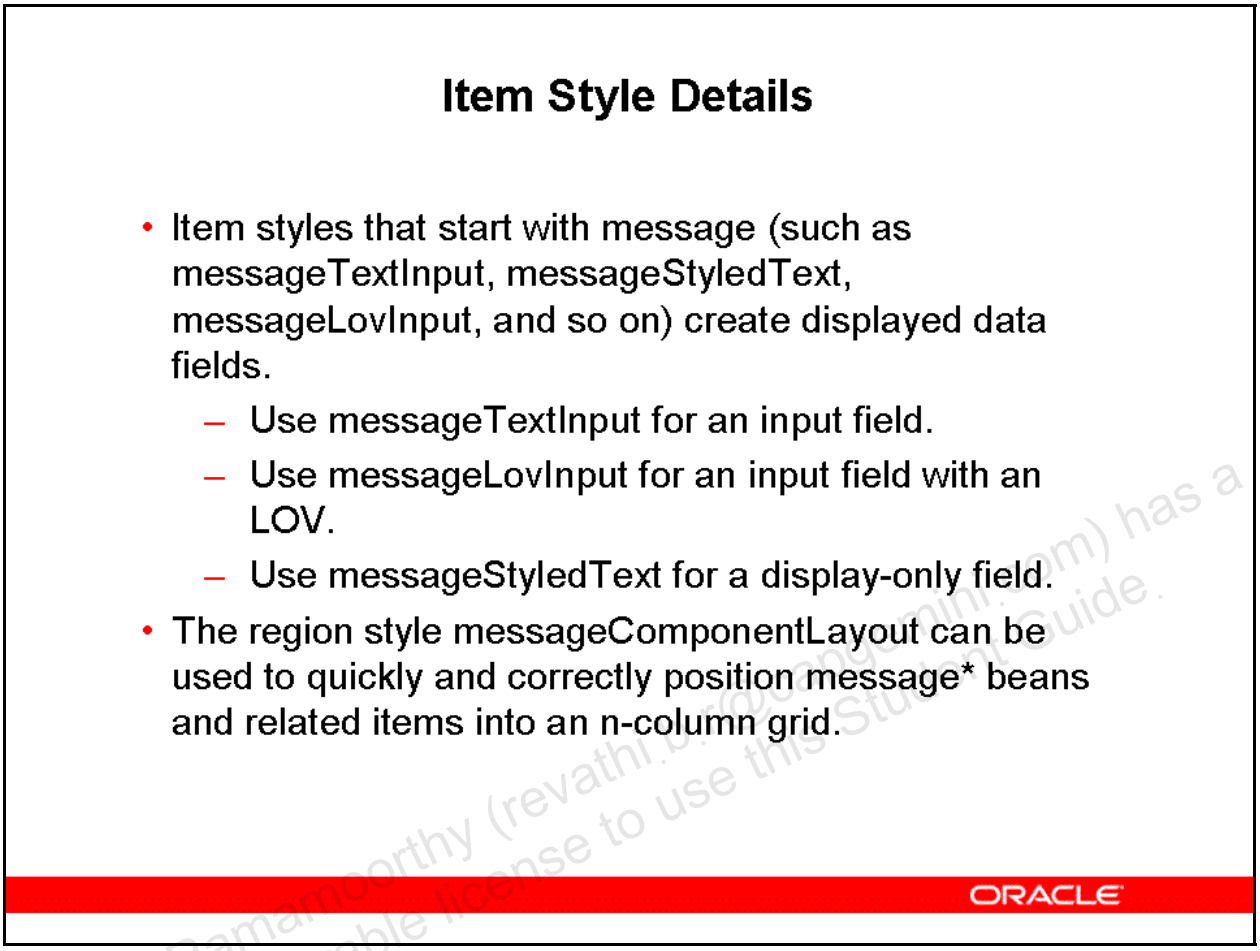
By default, the corporateBranding named children is set automatically when the page is created. The corporateBranding named child is set with default characteristics, but you can change this as desired.

Item Styles



It is absolutely critical that you begin to familiarize yourself with the OA Framework Developer's Guide (OAFDG). The OAFDG comes in the patch that contains the JDeveloper with OA Extension version that you are using for your development. In Chapter 4: Implementing Specific UI Features of the OAFDG, there is a detailed discussion of each of these region types, along with other UI features. The OAFDG is also available on My Oracle Support – formerly Metalink. Each updated version of JDeveloper will have a specific Developer's guide.

Item Style Details

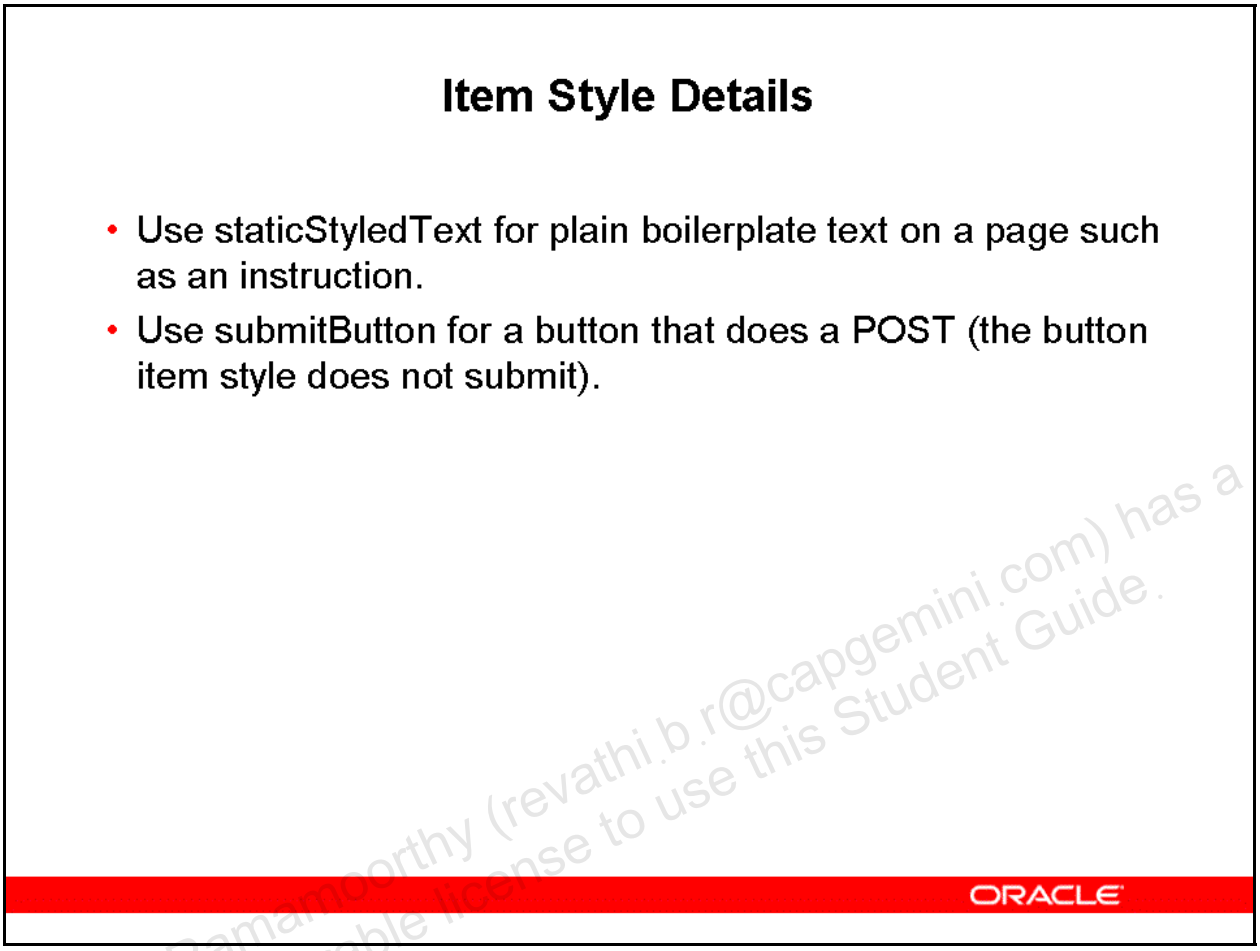


Message beans are components that are capable of displaying associated error, warning or information icon(s) with an explanatory message (for example, if a user enters the wrong value in a text input field an error icon renders next to its prompt). Typically message beans either display or accept data values (as opposed to boilerplate text, for example).

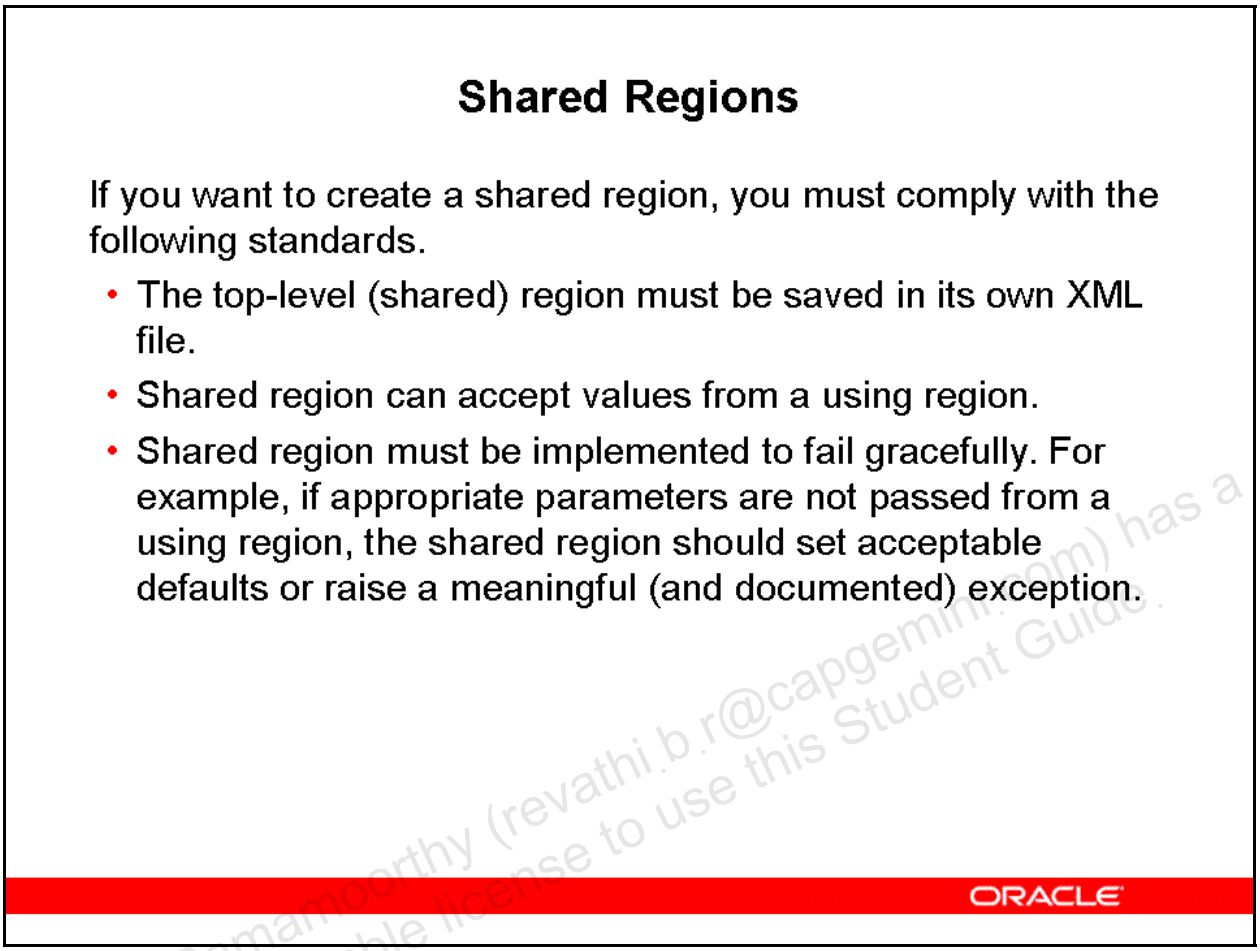
If you use the region style messageComponentLayout to contain your message beans, and you want to add some other type of bean such as a submitButton item, you must insulate or encapsulate the non-message bean in a messageLayout region first.

For example:

```
messageComponentLayout
    messageTextInput
    messageStyledText
    messageLovInput
    messageLayout
        submitButton
    messageTextInput
```

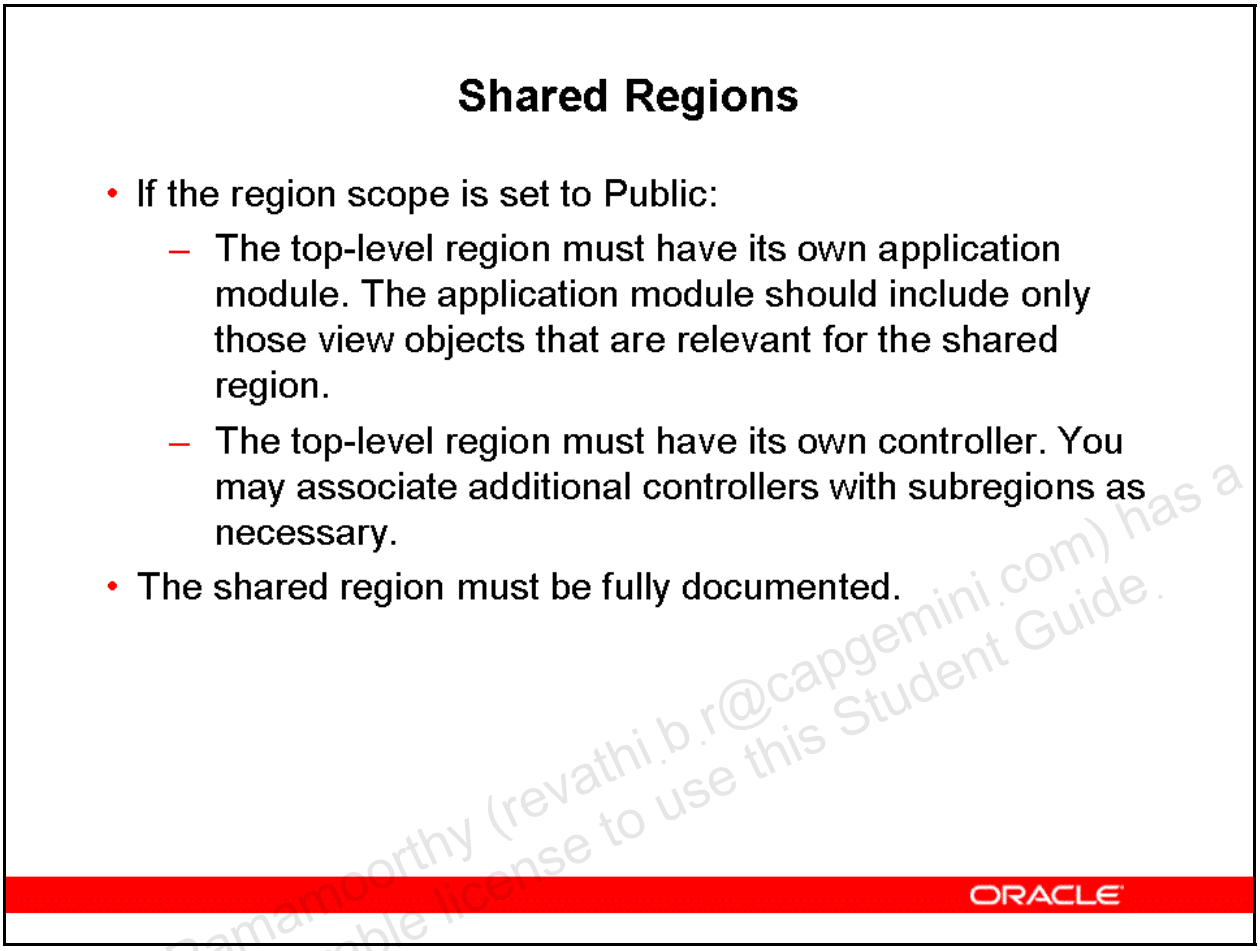


Shared Regions



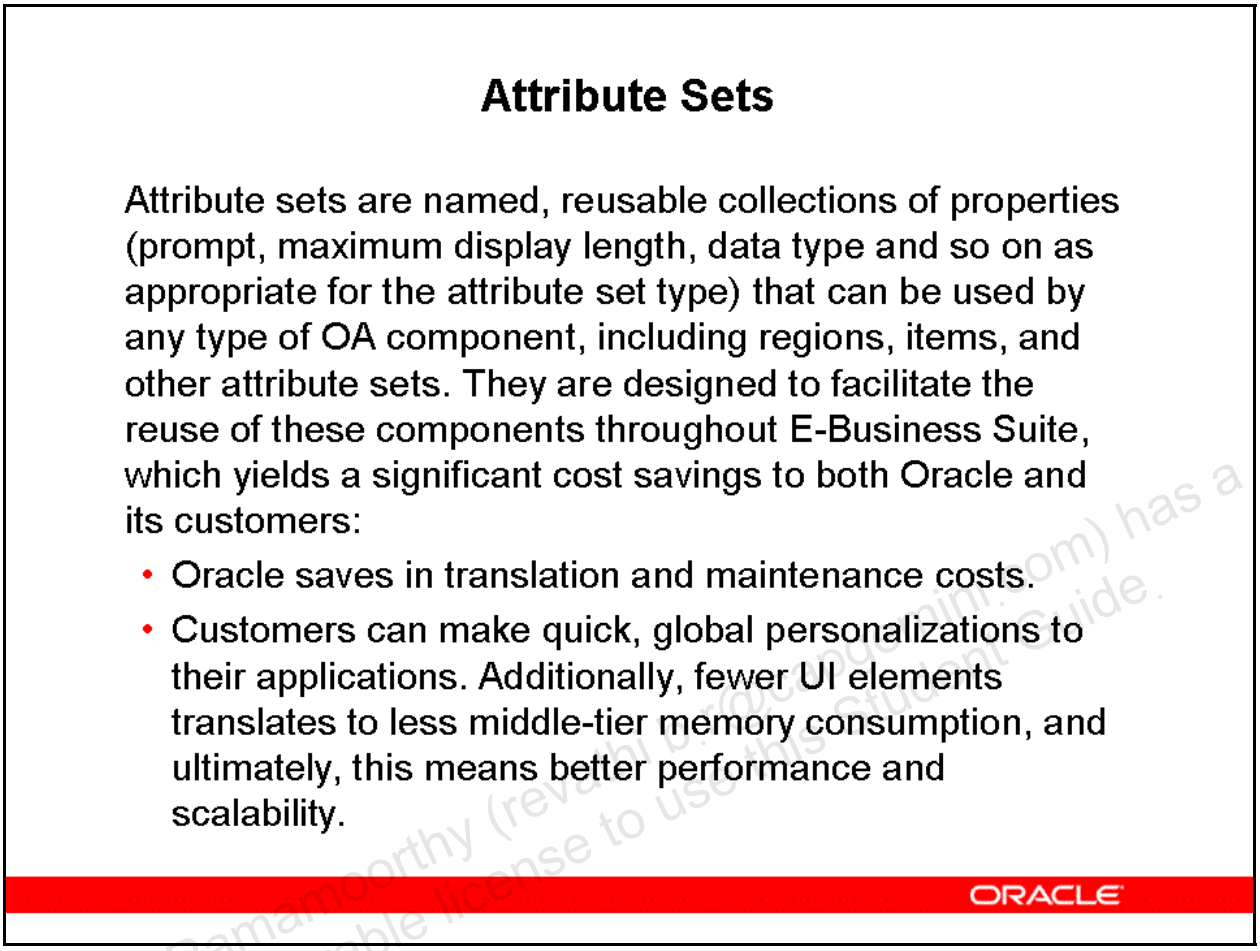
Shared Regions are an important tool to ensure a consistent look and feel, while minimizing the cost and time to make changes to a page. Shared regions can be used in multiple pages and shared across an application provided that the AM and associated BC4J objects are shared too.

Shared Regions



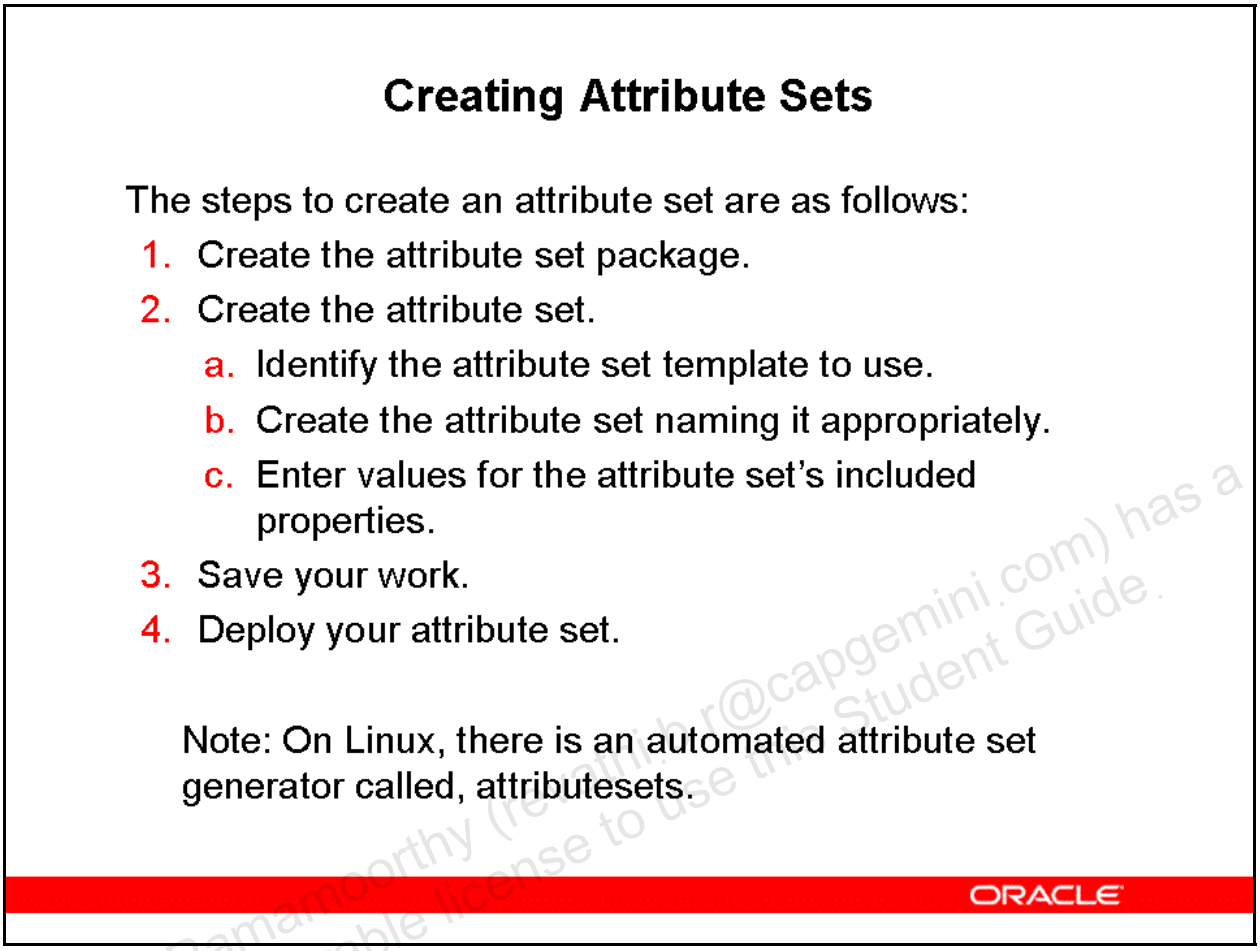
Shared Regions are an important tool to ensure a consistent look and feel, while minimizing the cost and time to make changes to a page.

Attribute Sets



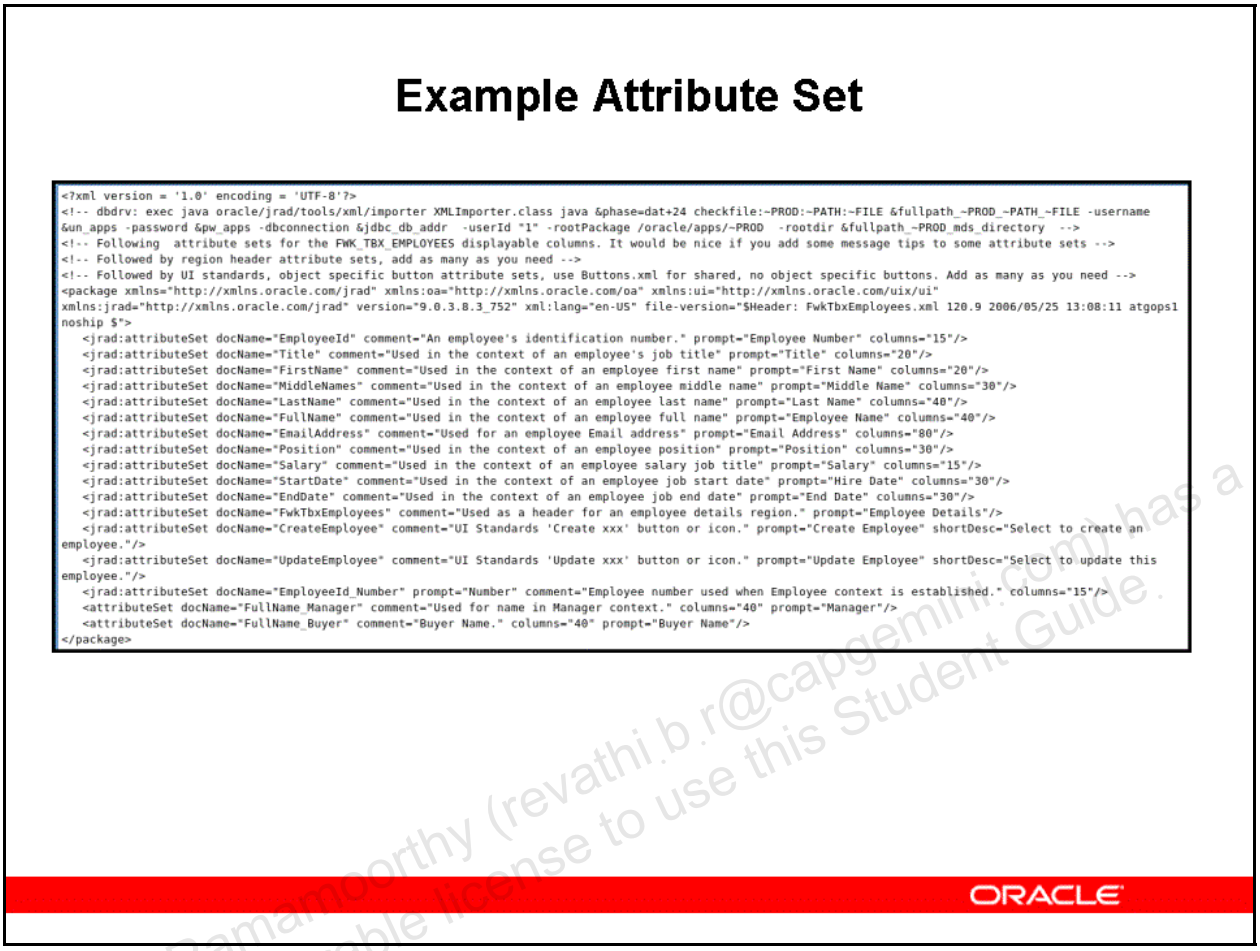
Property values can be inherited from either attribute sets or extended objects (Extends property). Inherited values are designated by a blue arrow in the Windows version of JDeveloper. When you download JDeveloper you will find some default Attribute sets. These attribute sets are defaults that are used so that a developer doesn't have to enter all the specifically different attributes when they are creating components. Again, the word "Attribute" is an overloaded term in Oracle and here it is used as "property" or "Setting" of an item.

Creating Attribute Sets



Not all properties should be set via an attribute set. Check the OA Framework Developer's Guide for standards and guidelines. Attribute sets are created in XML and should be easy for a developer to modify.

Example Attribute Set



As you can see in the above screen capture and below example, the “FwkTbxEmployees.xml” attribute set has certain properties that are pre-set and allow quick and consistent development of redundant components.

<?xml version = '1.0' encoding = 'UTF-8'?>

<!-- dbdrv: exec java oracle/jrad/tools/xml/importer XMLImporter.class java &phase=dat+24 checkfile:~PROD:~PATH:~FILE &fullpath_~PROD_~PATH_~FILE -username &un_apps -password &pw_apps -dbconnection &jdbc_db_addr -userId "1" -rootPackage /oracle/apps/~PROD -rootdir &fullpath_~PROD_mds_directory -->

<!-- Following attribute sets for the FWK_TBX_EMPLOYEES displayable columns. It would be nice if you add some message tips to some attribute sets -->

<!-- Followed by region header attribute sets, add as many as you need -->

<!-- Followed by UI standards, object specific button attribute sets, use Buttons.xml for shared, no object specific buttons. Add as many as you need -->

<package xmlns="http://xmlns.oracle.com/jrad" xmlns:oa="http://xmlns.oracle.com/oa" xmlns:ui="http://xmlns.oracle.com/uix/ui" xmlns:jrad="http://xmlns.oracle.com/jrad" version="9.0.3.8.3_752" xml:lang="en-US" file-version="$Header: FwkTbxEmployees.xml 120.9 2006/05/25 13:08:11 atgops1 noship $">

<jrad:attributeSet docName="EmployeeId" comment="An employee's identification number." prompt="Employee Number" columns="15"/>

<jrad:attributeSet docName="Title" comment="Used in the context of an employee's job title" prompt="Title" columns="20"/>

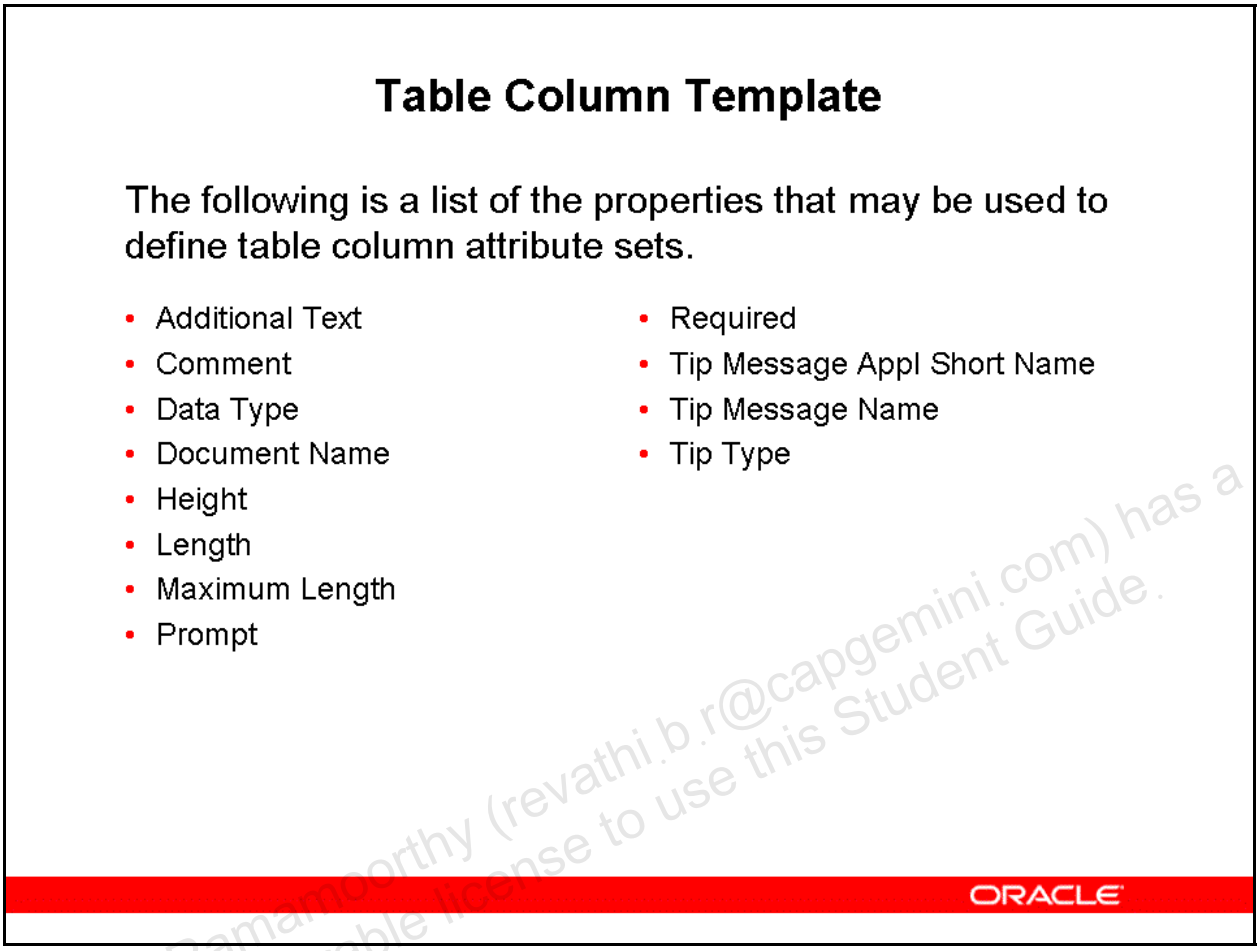
<jrad:attributeSet docName="FirstName" comment="Used in the context of an employee first name" prompt="First Name" columns="20"/>

<jrad:attributeSet docName="MiddleNames" comment="Used in the context of an employee middle name" prompt="Middle Name" columns="30"/>

<jrad:attributeSet docName="LastName" comment="Used in the context of an employee last name" prompt="Last Name" columns="40"/>

```
  <jrad:attributeSet docName="FullName" comment="Used in the context of an employee full name" prompt="Employee Name"
columns="40"/>
  <jrad:attributeSet docName="EmailAddress" comment="Used for an employee Email address" prompt="Email Address"
columns="80"/>
  <jrad:attributeSet docName="Position" comment="Used in the context of an employee position" prompt="Position" columns="30"/>
  <jrad:attributeSet docName="Salary" comment="Used in the context of an employee salary job title" prompt="Salary"
columns="15"/>
  <jrad:attributeSet docName="StartDate" comment="Used in the context of an employee job start date" prompt="Hire Date"
columns="30"/>
  <jrad:attributeSet docName="EndDate" comment="Used in the context of an employee job end date" prompt="End Date"
columns="30"/>
  <jrad:attributeSet docName="FwkTbxEmployees" comment="Used as a header for an employee details region."
prompt="Employee Details"/>
  <jrad:attributeSet docName="CreateEmployee" comment="UI Standards 'Create xxx' button or icon." prompt="Create Employee"
shortDesc="Select to create an employee."/>
  <jrad:attributeSet docName="UpdateEmployee" comment="UI Standards 'Update xxx' button or icon." prompt="Update Employee"
shortDesc="Select to update this employee."/>
  <jrad:attributeSet docName="EmployeeId_Number" prompt="Number" comment="Employee number used when Employee
context is established." columns="15"/>
  <attributeSet docName="FullName_Manager" comment="Used for name in Manager context." columns="40" prompt="Manager"/>
  <attributeSet docName="FullName_Buyer" comment="Buyer Name." columns="40" prompt="Buyer Name"/>
</package>
```

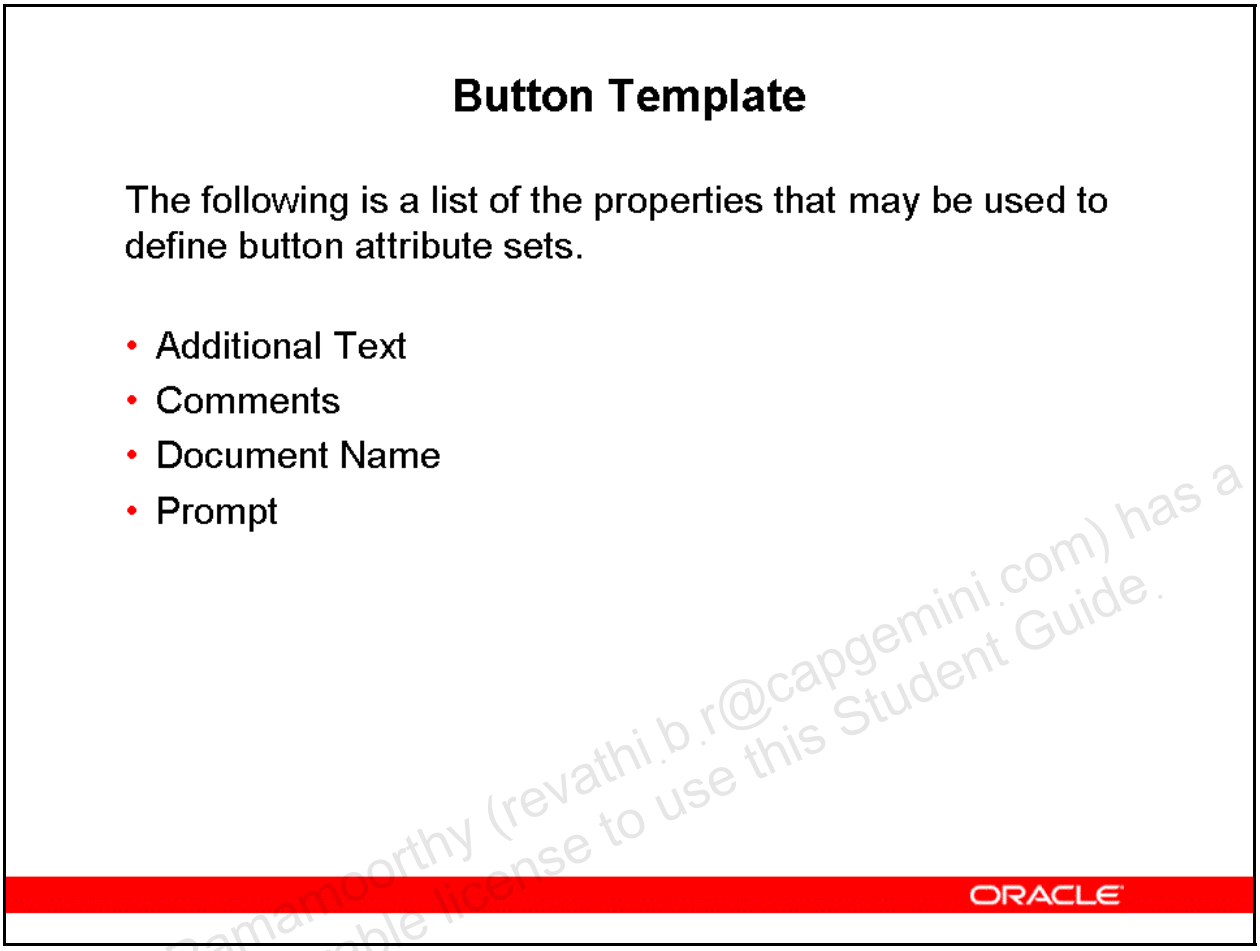
Table Column Template



Property values can be inherited from either attribute sets or extended objects (Extends property). Inherited values are designated by a blue arrow in the Windows version of JDeveloper.

Note: Just because JDeveloper allows you to set all the properties, it does not mean that you could or should do so. Please note that the attribute set template specifically limits the properties to the item type.

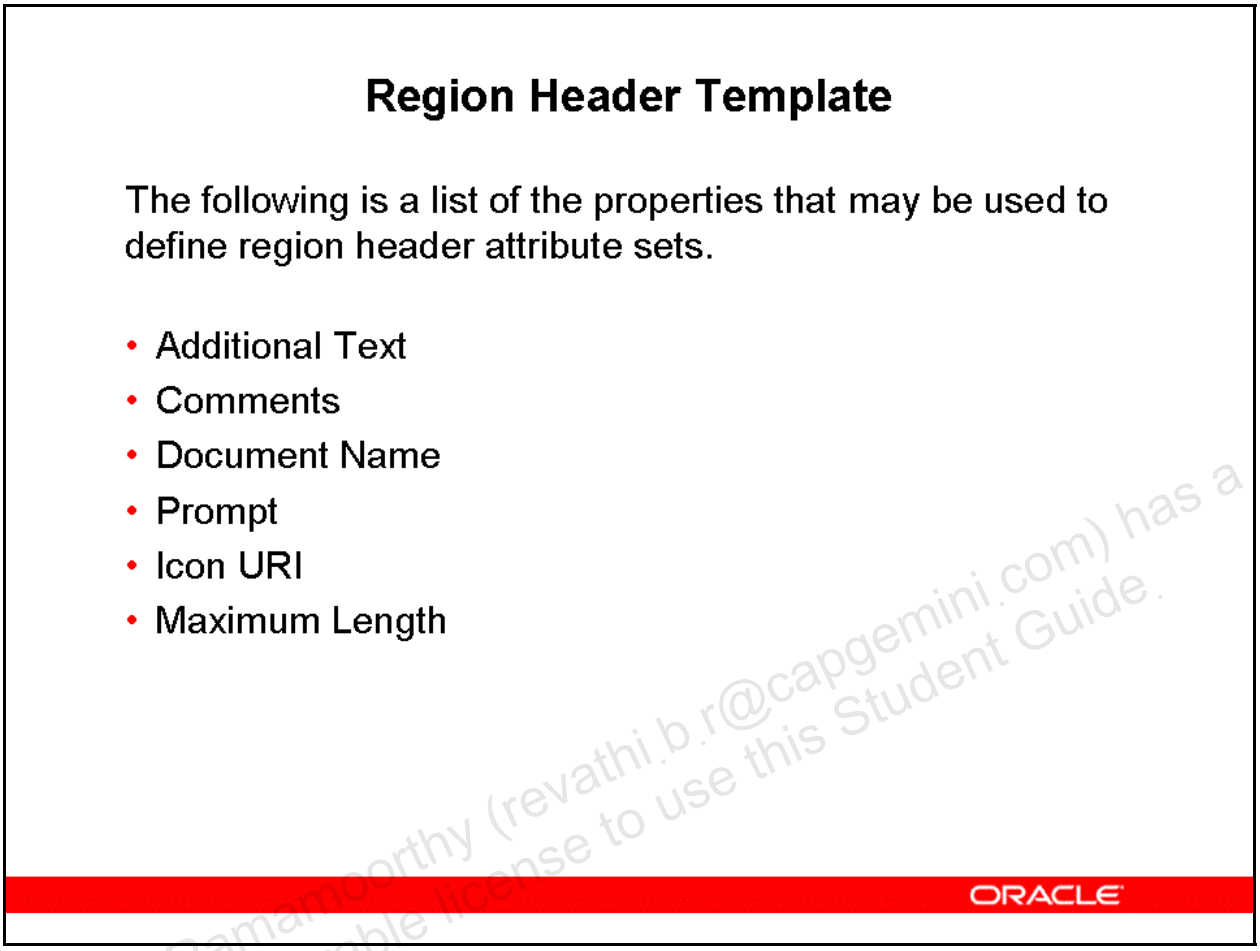
Button Template



Property values can be inherited from either attribute sets or extended objects (Extends property). Inherited values are designated by a blue arrow in the Windows version of JDeveloper.

Note: Just because JDeveloper allows you to set all the properties, it does not mean that you could or should do so. Please note that the attribute set template specifically limits the properties to the item type.

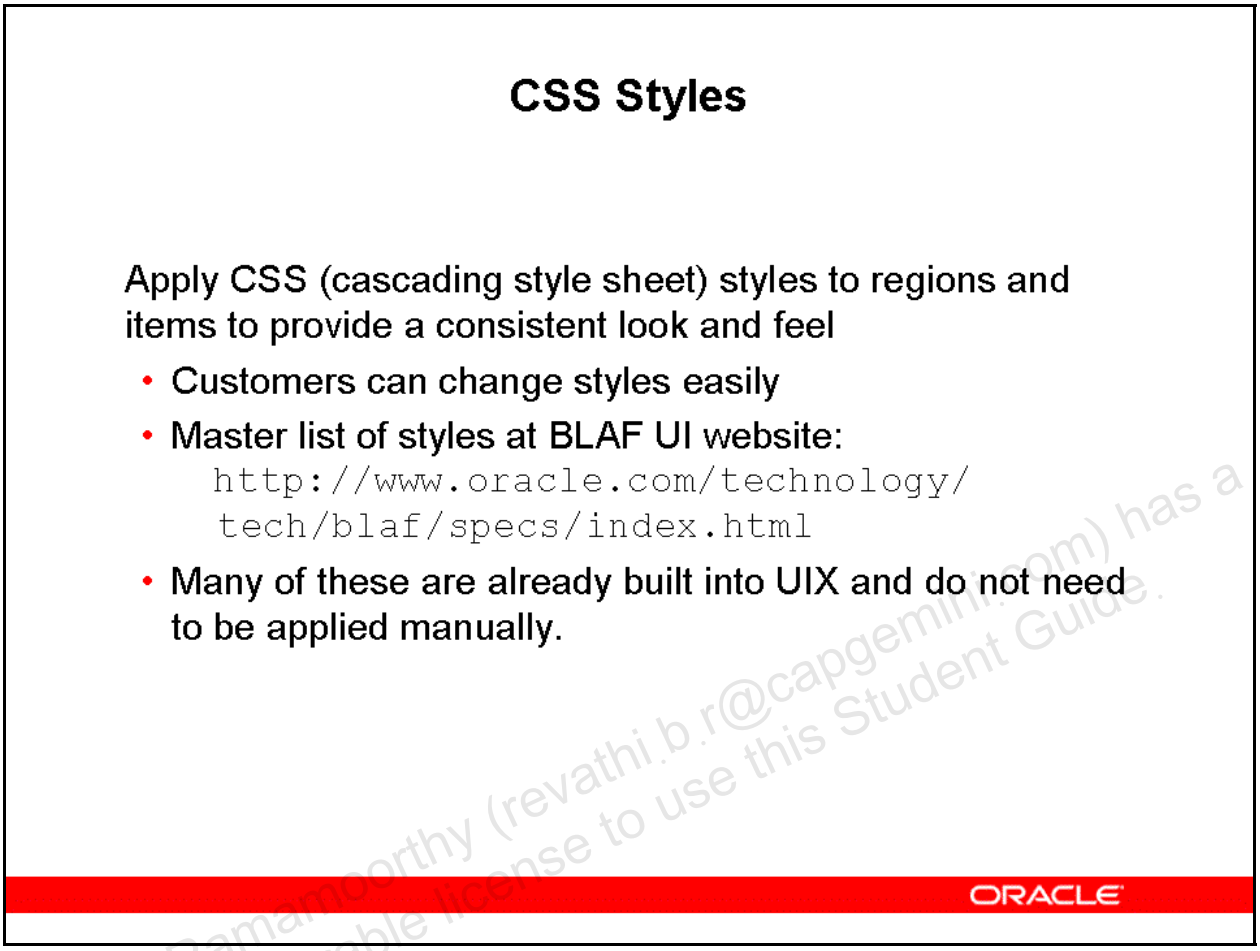
Region Header Template



Property values can be inherited from either attribute sets or extended objects (Extends property). Inherited values are designated by a blue arrow in the Windows version of JDeveloper.

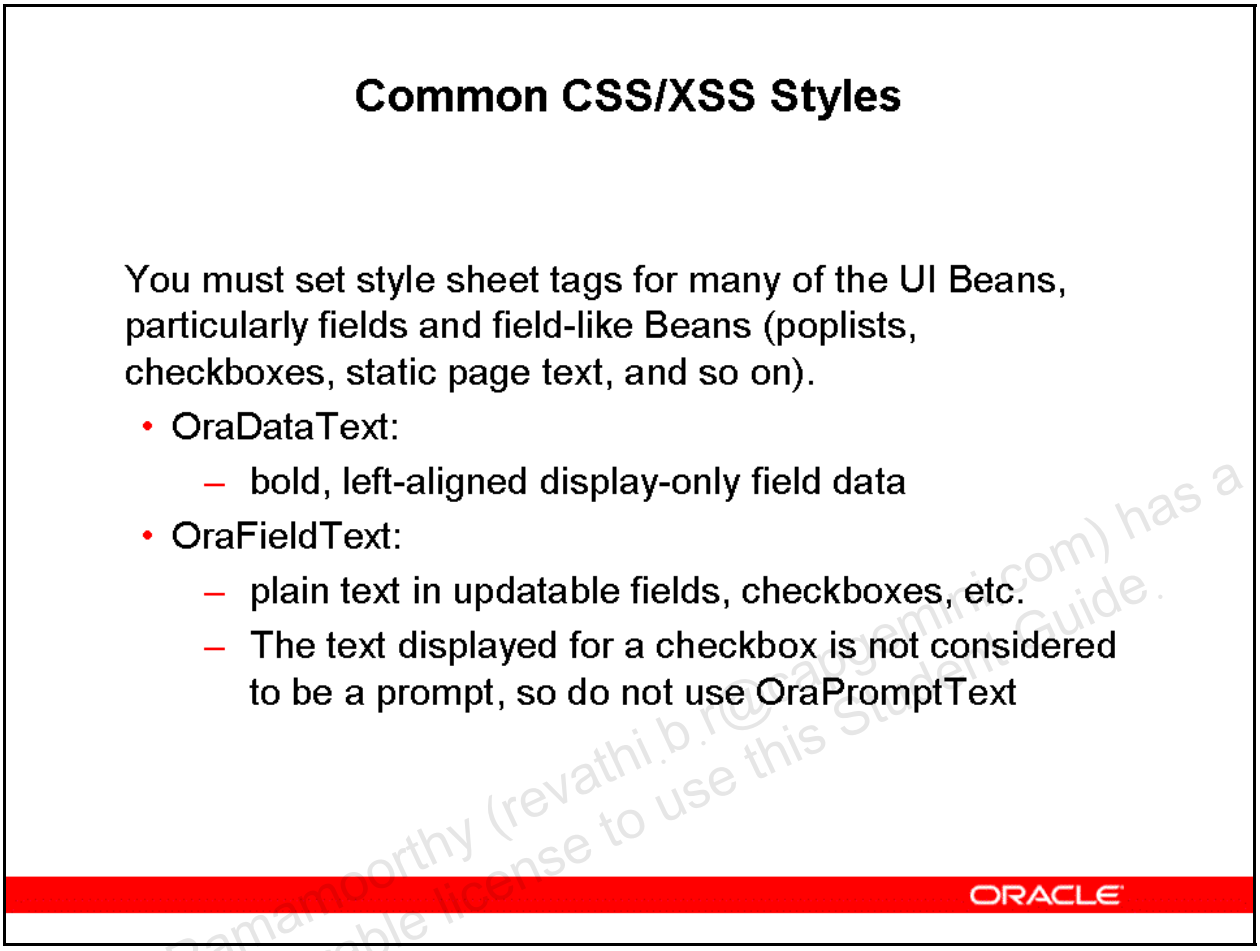
Note: Just because JDeveloper allows you to set all the properties, it does not mean that you could or should do so. Please note that the attribute set template specifically limits the properties to the item type.

CSS Styles



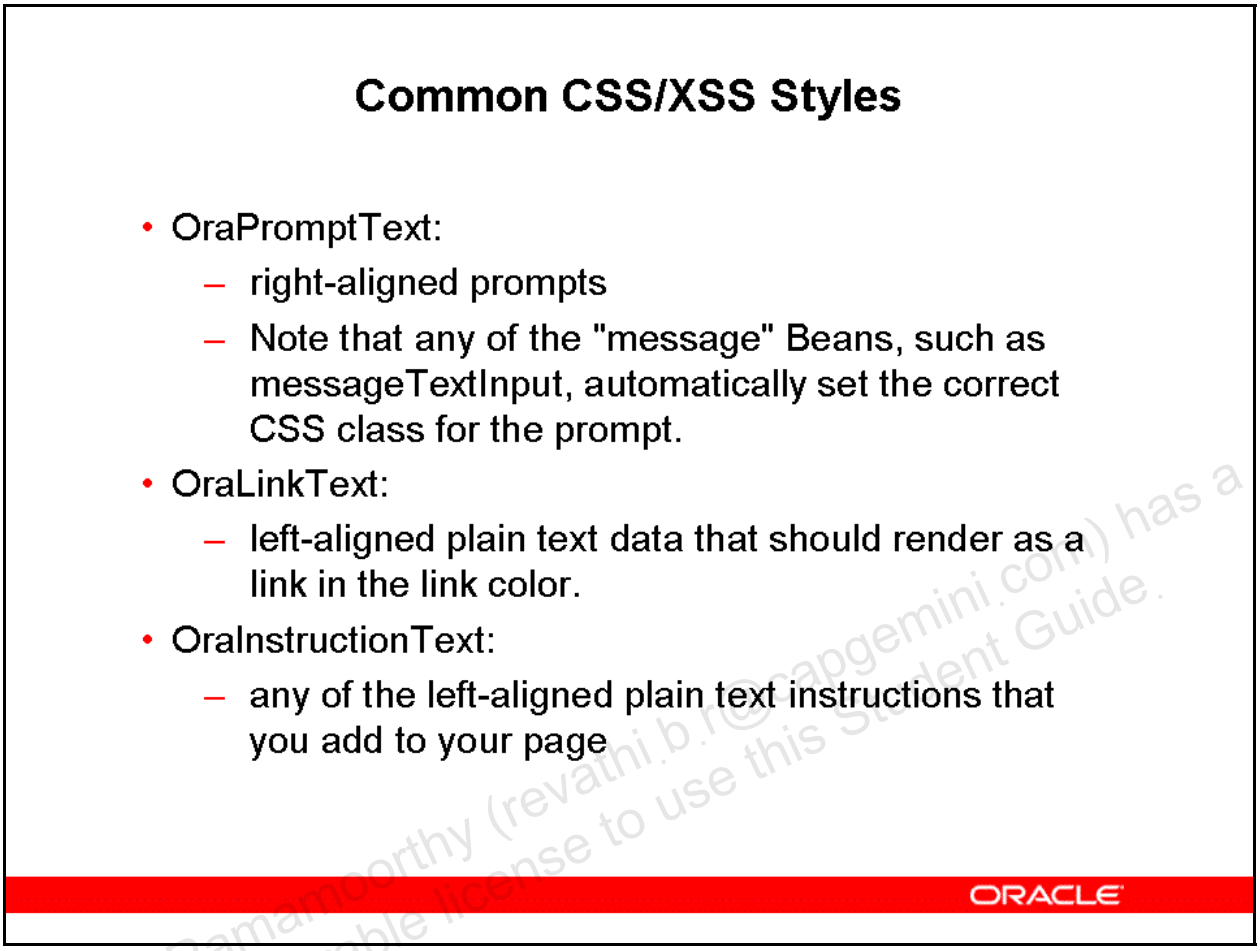
Be careful with CSS changes. One small CSS change, because it is shared and used across all OA Framework pages, can have a dramatic impact on the page. If not planned and executed properly, that change can have unintended consequences.

Common CSS/XSS Styles



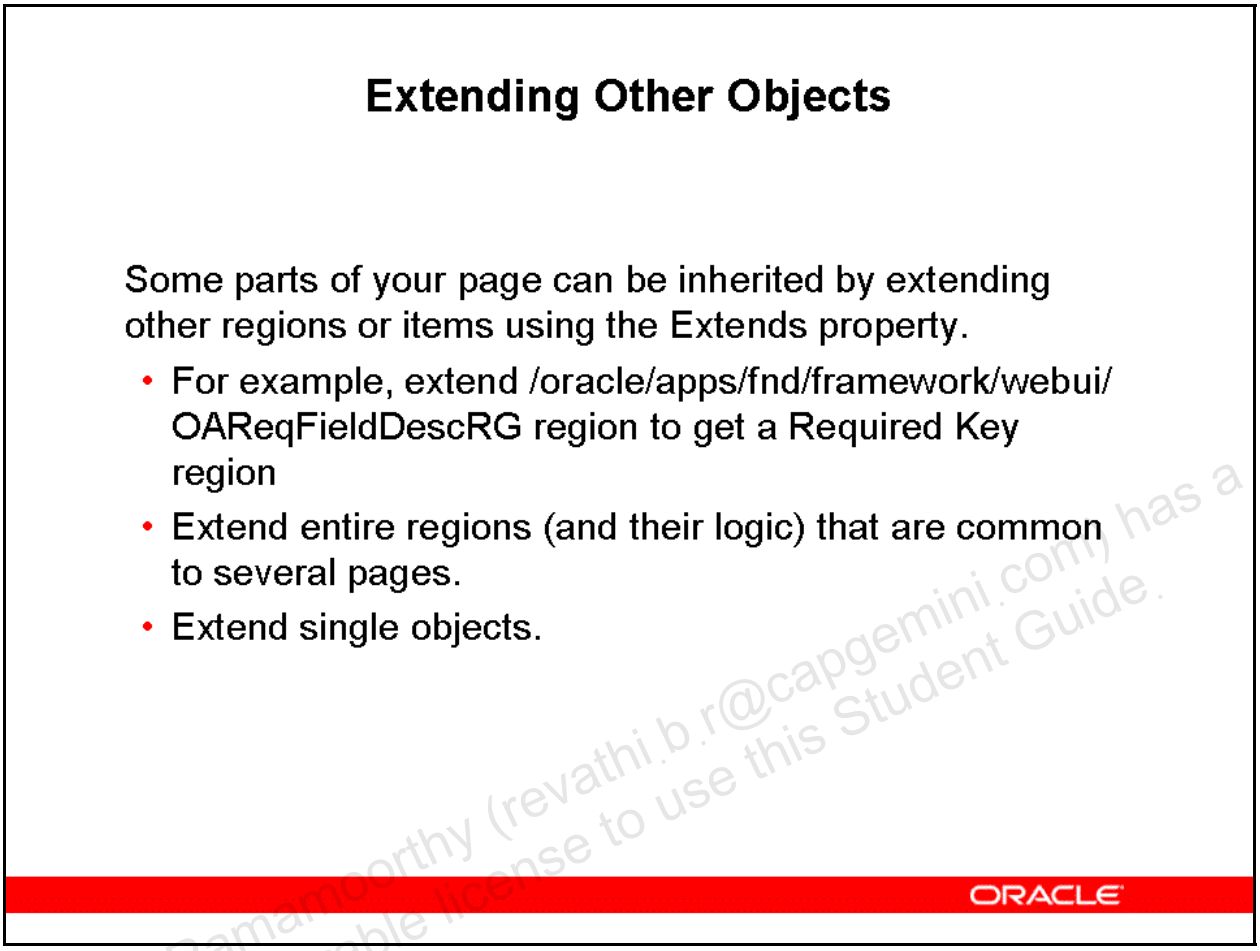
See the OA Framework Developer's Guide for more on these common CSS styles.

Common CSS/XSS Styles



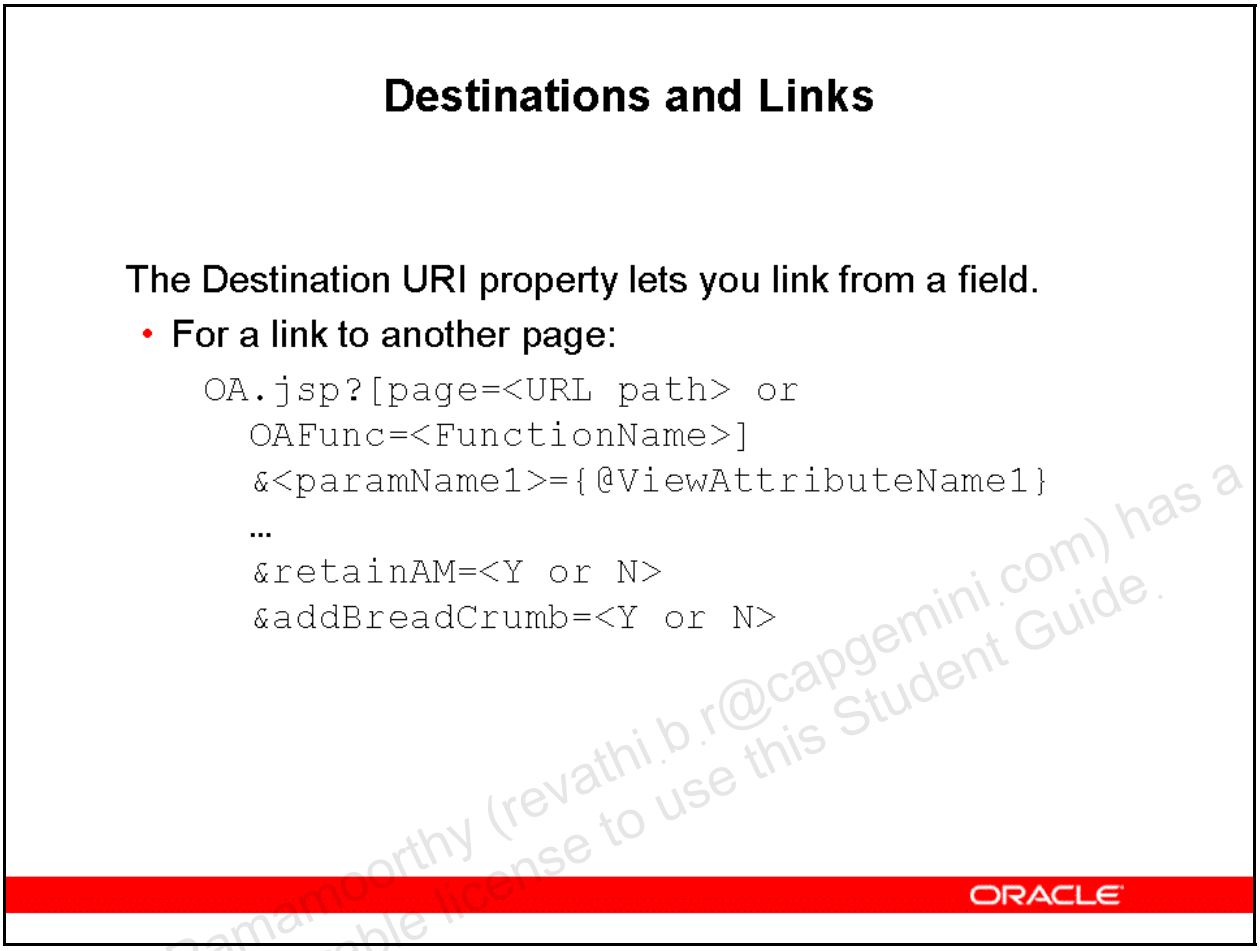
See the OA Framework Developer's Guide for more on these common CSS styles.

Extending Other Objects



When you extend an UI object, you are setting the Extends property of that UI object. This uses the same terminology as Java uses when it extends a java object. While identical in concept, the implements are quite different.

Destinations and Links

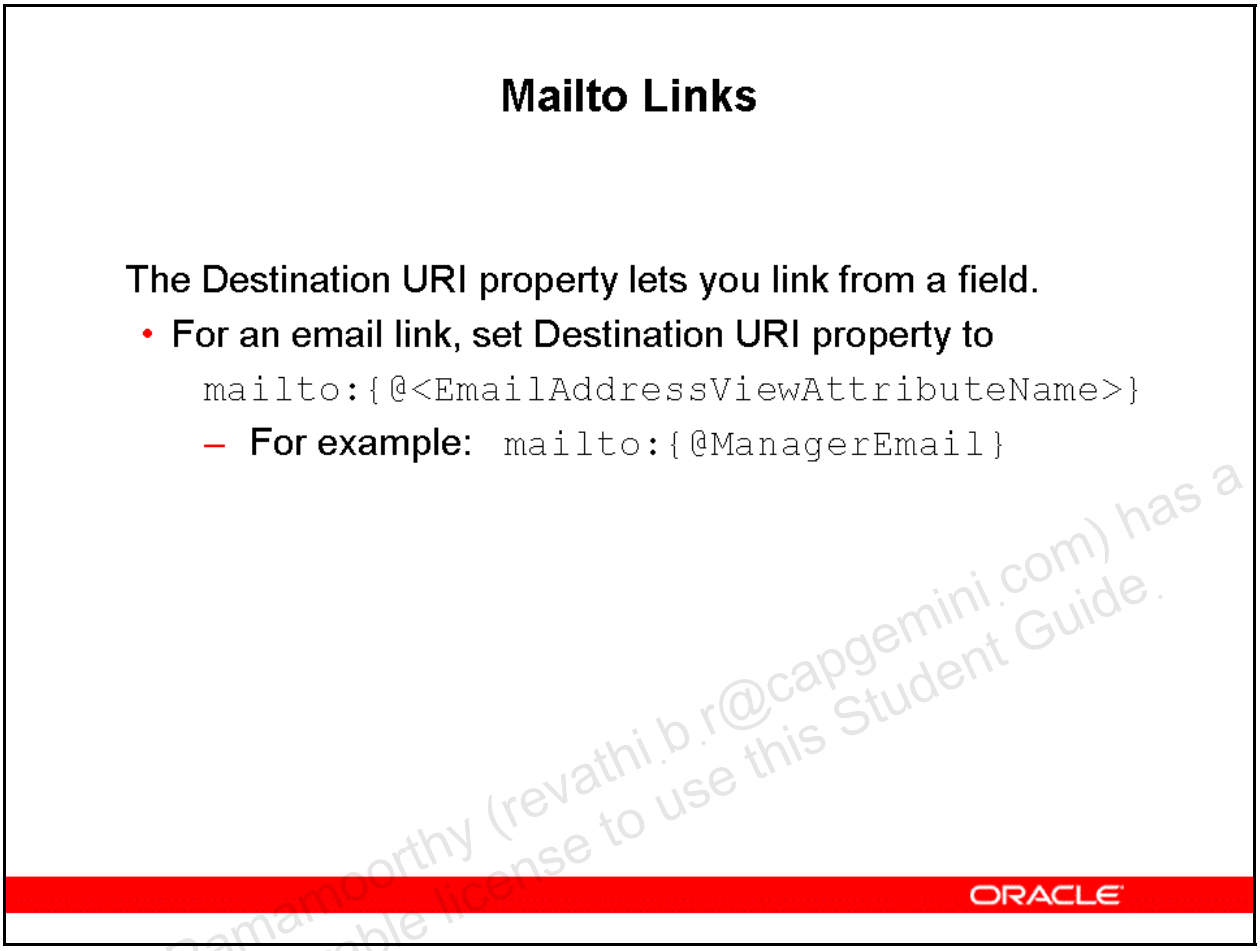


Whenever you programmatically or declaratively navigate to a page that is not included in your navigation menu, you should use the page name (page=/...) syntax instead of a function (OAFunc=...). Whenever you navigate to page whose navigation function IS included in the navigation menu, you may use the navigation function as a shorthand.

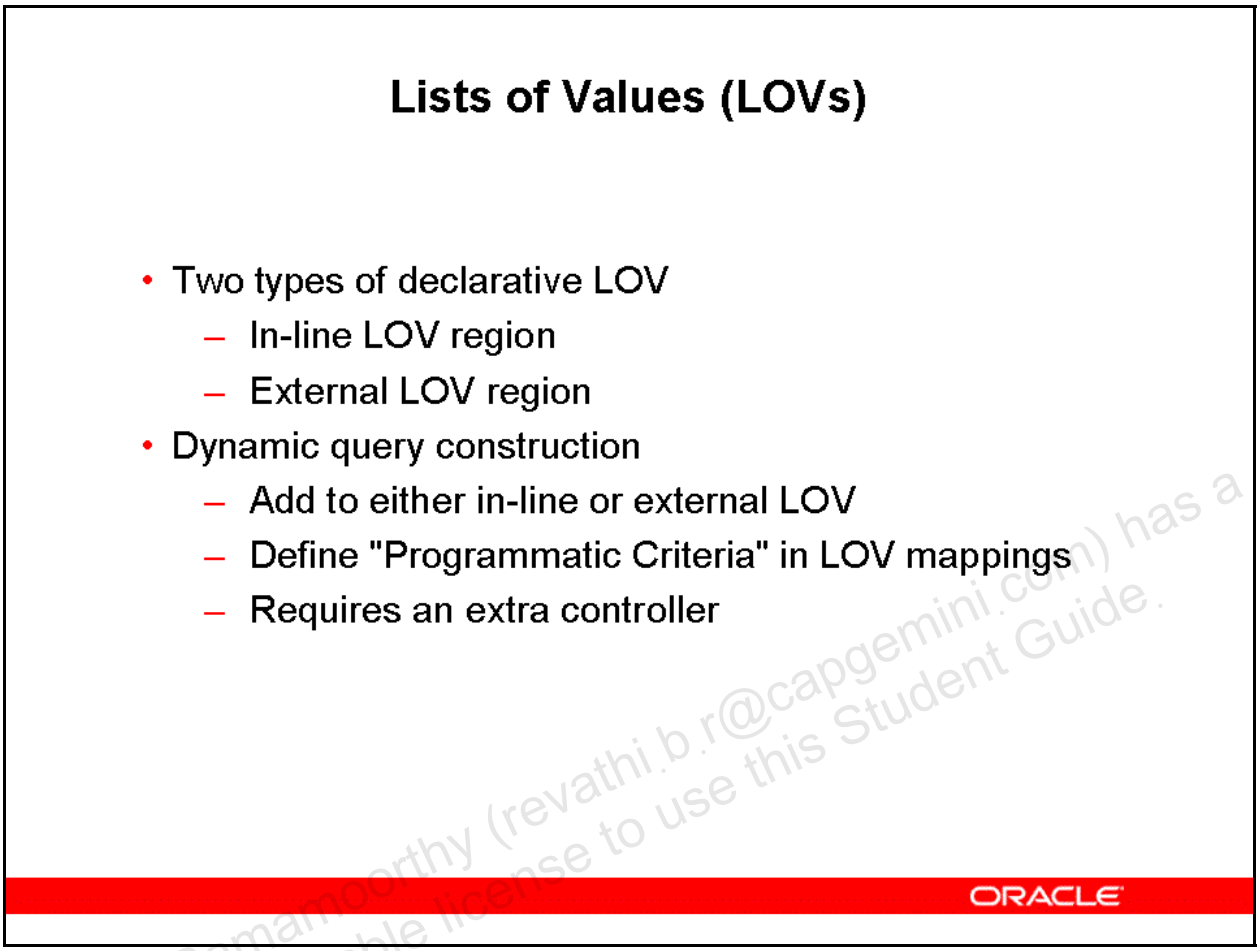
We have opted to consistently use the page=/... syntax in the labs.

We are going in this direction to help underscore the important difference between a "navigation" function, and a "securing" function. The page=/... syntax has nothing to do with page security, and the OAFunc=... syntax has nothing to do with navigation. The function syntax is the older way of doing navigation and security combined.

Note that the Destination URI property is only available on some item types, such as messageStyledText, but not on others such as messageTextInput.

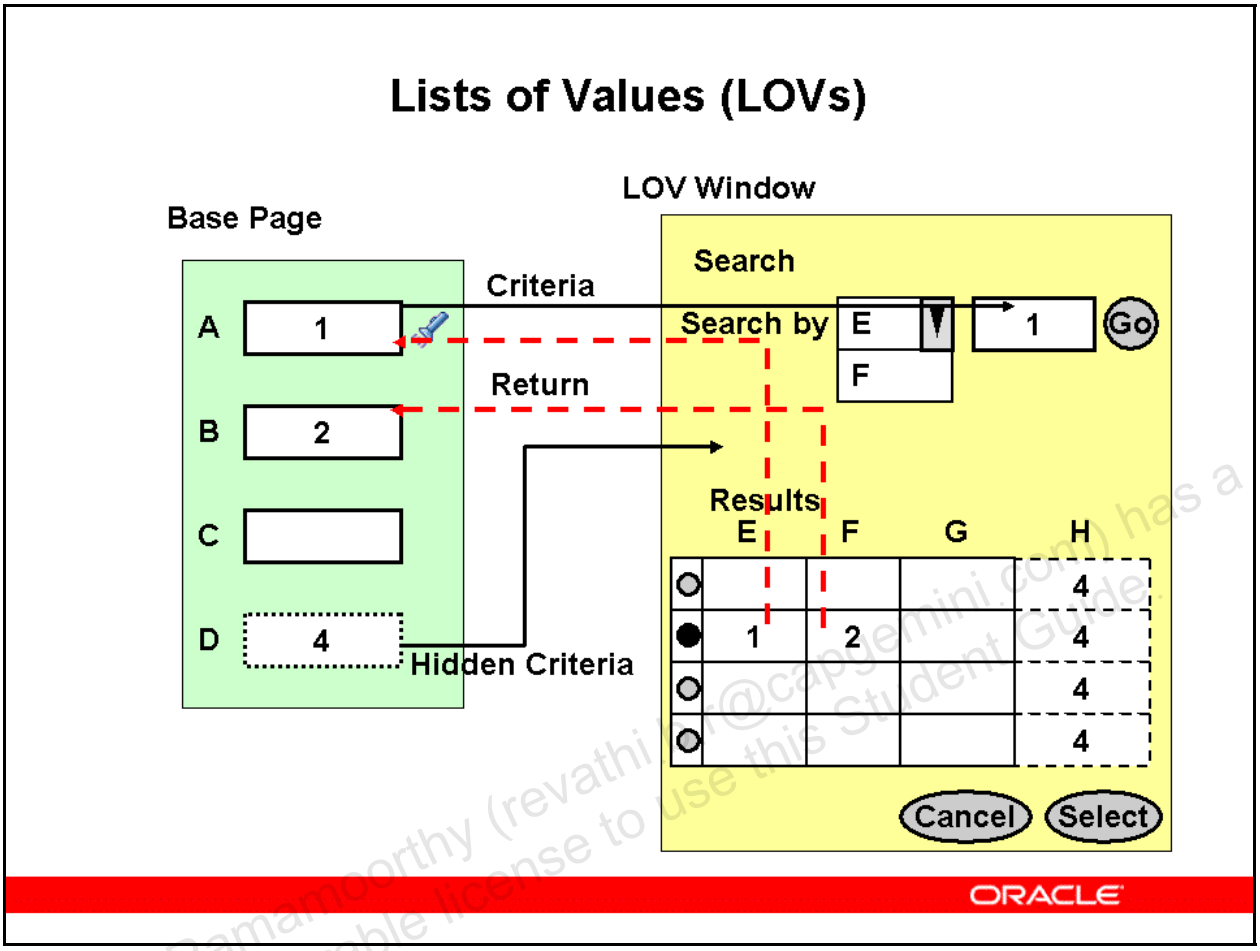


Lists of Values (LOVs)



Be aware of the limits of LOVs and other UI objects. Not every region type is capable of containing an LOV, even if you can add that LOV within JDeveloper. OA Framework may simply refuse to display the object. See the OA Framework Developer's Guide for more information.

Lists of Values (LOVs)



Fields E, F, and H have their Search Allowed property set to True, so they appear in the poplist in the LOV window.

Field A (with the search icon) on the base page is the LOV field. The LOV field must have both criteria item and return item LOV mappings. The other fields in the base page can only be either criteria or return items. If a field other than the LOV field provides criteria, that criteria must be an exact match to a column in the LOV results table. While the diagram shows the criteria field D and the results column H as hidden fields, they can be hidden or displayed fields.

Defining an External LOV

- Build the LOV region in a region file:
  - Create an LOV region of style ListofValues.
  - Create a table region inside the LOV region.
  - Items that users can choose for entering query criteria in LOV window must have Search Allowed property set to True.
- Attach the LOV region to the base page using External List of Values property on LOV field
  - Set to qualified name of region file.
  - Base page LOV field must have Item Style property set to messageLovInput.

ORACLE

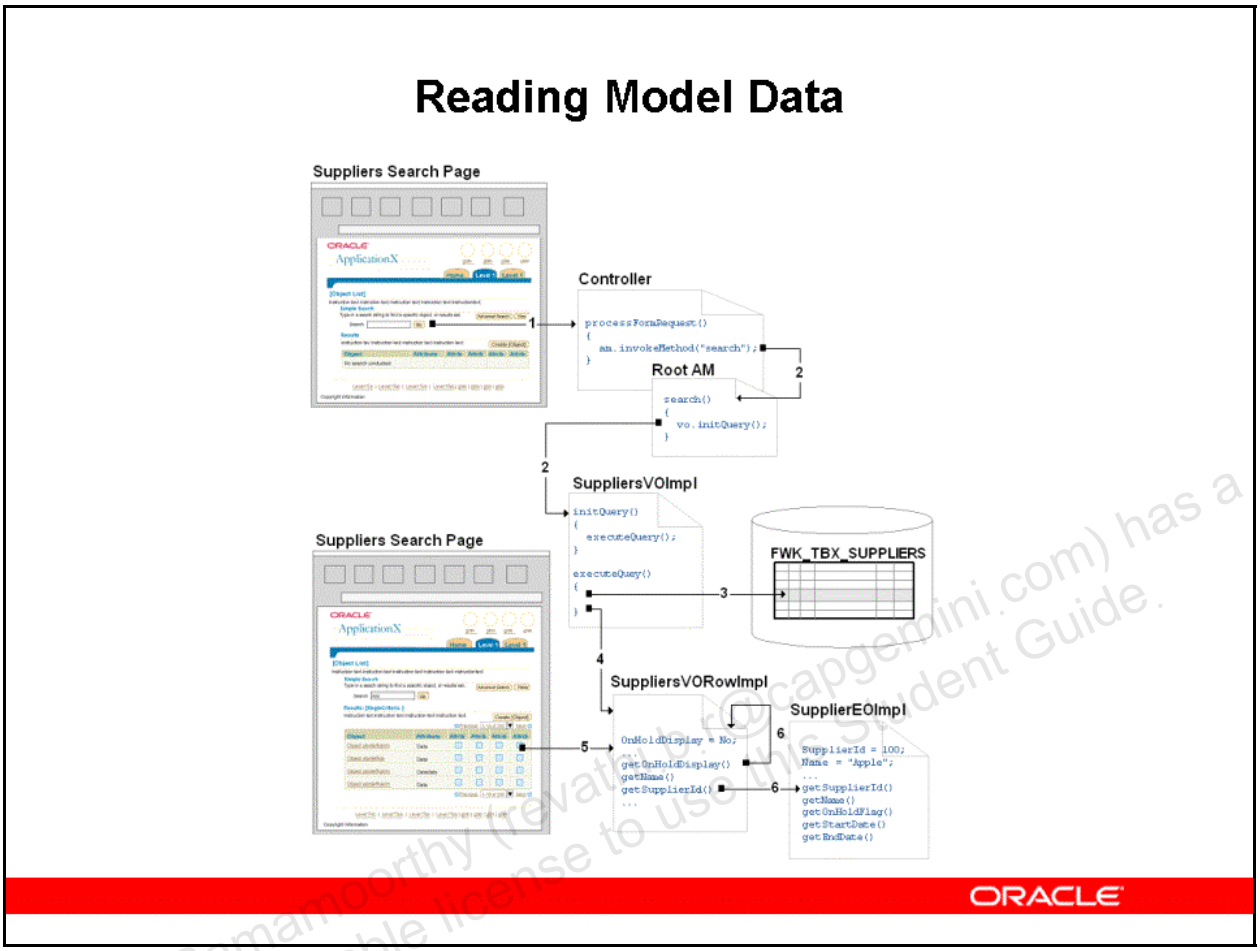
Defining an External LOV

Create the LOV mappings:

- LOV field on base page must be specified for both Criteria From and Return To properties.
- Other fields can only be used for one direction only (criteria item or return item).
- Other criteria fields provide hidden criteria that is always applied to the search in the LOV window.

ORACLE

Reading Model Data



Assuming you have specified the appropriate data source bindings, the OA Framework automatically reads data from the model for display in the view, and writes user-entered data in the view back to the model. You don't need to write a single line of code (except for any validation you want to perform in the underlying entity objects, of course).

In simple terms, every time the OA Framework renders a page, it calls the current view object row's get<AttributeName> method for each web bean's associated view object attribute. Consider an example page with a "Suppliers" table, which binds to a SuppliersVO view object. The SuppliersVO maps to an underlying SupplierEOImpl, although it also includes one "calculated" transient attribute ("OnHoldDisplay") that does not have a corresponding entity object attribute.

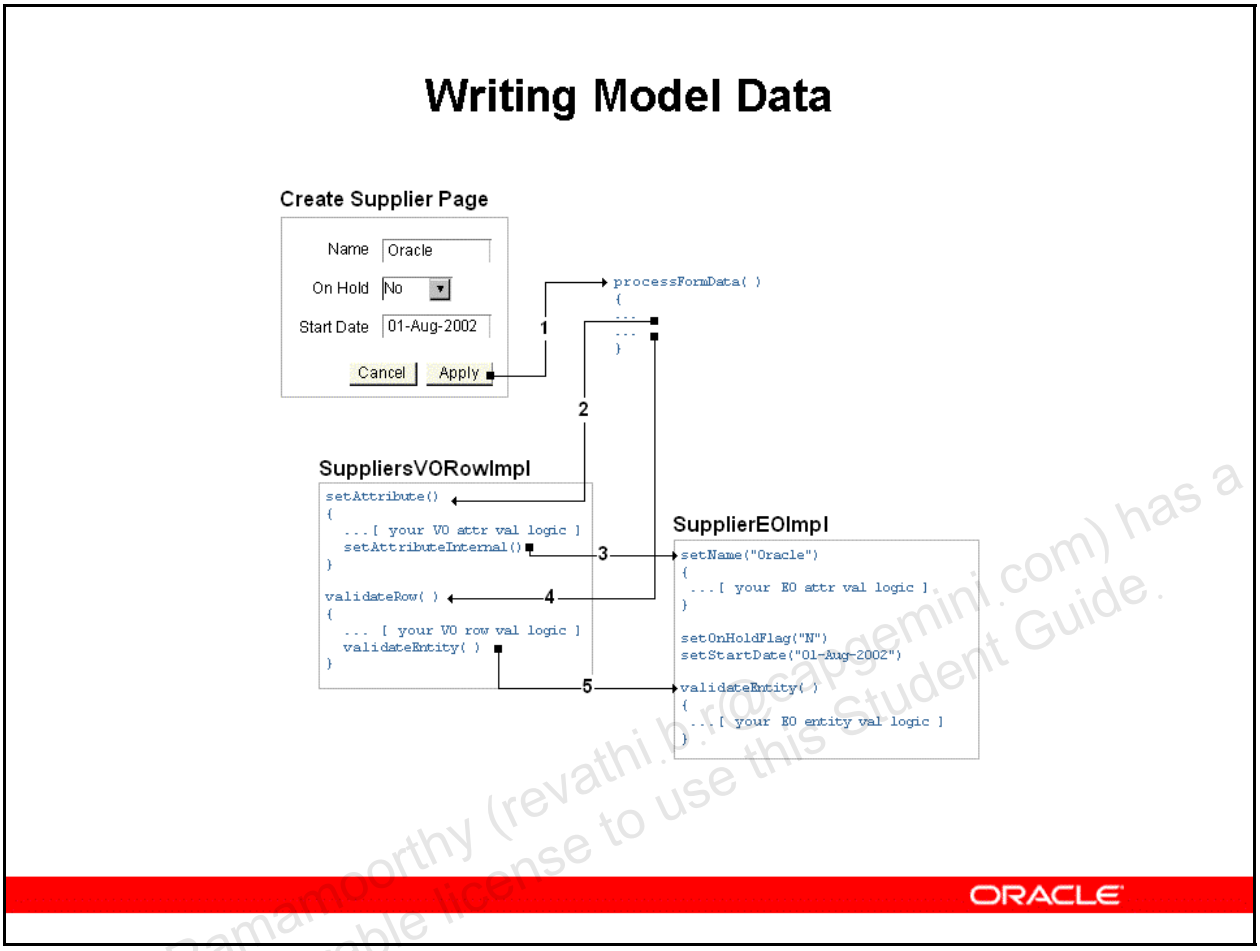
1. The user selects the "Search" region's "Go" button to populate search results in the "Suppliers" table.
2. The "Search" region's controller handles the button press by invoking a search method in the root application module, which in turn delegates to the SuppliersVOImpl class so it can query itself.
3. Within the executeQuery method, the SuppliersVOImpl view object performs its SQL SELECT in the database.

4. For each row returned in our example result set, the view object instantiates a SupplierEOImpl entity object and sets its attribute values with the query results.

Note: Entity object-based attribute values aren't actually stored anywhere in the view object. They "live" in the entity object, and are retrieved as needed by the view object. "Calculated" (meaning the values are simply selected from a SQL statement and have no relationship to an entity object) or "Transient" view object attribute values are stored on the SuppliersVORowImpl object.

5. During page rendering (after all the query processing), the OA Framework uses the view object data bindings defined for each web bean to call the corresponding SuppliersVORowImpl object's getAttribute("<attributeName">) which in turns calls its get<AttributeName> method.
6. The SuppliersVORowImpl get<Attribute Name> method in turn calls the corresponding SupplierEOImpl get<AttributeName> method to retrieve the value. For the "calculated" OnHoldDisplay attribute, the view object row retrieves the value from its own cache.

Writing Model Data



Whenever the browser issues a POST request , the OA Framework automatically writes any user-entered form values back to the underlying view objects, which in turn update any corresponding entity objects as shown below.

Note: The following steps assume that the entity object for the row has already been successfully instantiated and initialized (such as in the create method on the underlying view object for the page when the user originally comes into a Create page. The view object create method calls the corresponding create method on the entity object behind the scenes).

1. UIX performs onSubmit Javascript validation (required fields, data types, formats) and issues the POST request only if this validation succeeds.
2. The browser sends a POST request and the OA Framework calls the processFormData methods on all the web beans in the hierarchy.
3. Within processFormData, the OA Framework automatically calls setAttribute(String name, Object value) on the current row of the underlying view object for each bean. This executes any attribute-level validation that you've written in the view object row.

4. Within this setAttribute method, the view object row automatically calls the corresponding set<AttributeName> method in the underlying entity object. This executes any associated attribute-level validation in the entity object.
5. Once all the attribute values have been set, the OA Framework calls the view object validate for each row it modified to execute any associated row-level validation.
6. Finally, within the validate method, the view object row calls validateEntity for the underlying entity object which executes any associated entity-level validation.

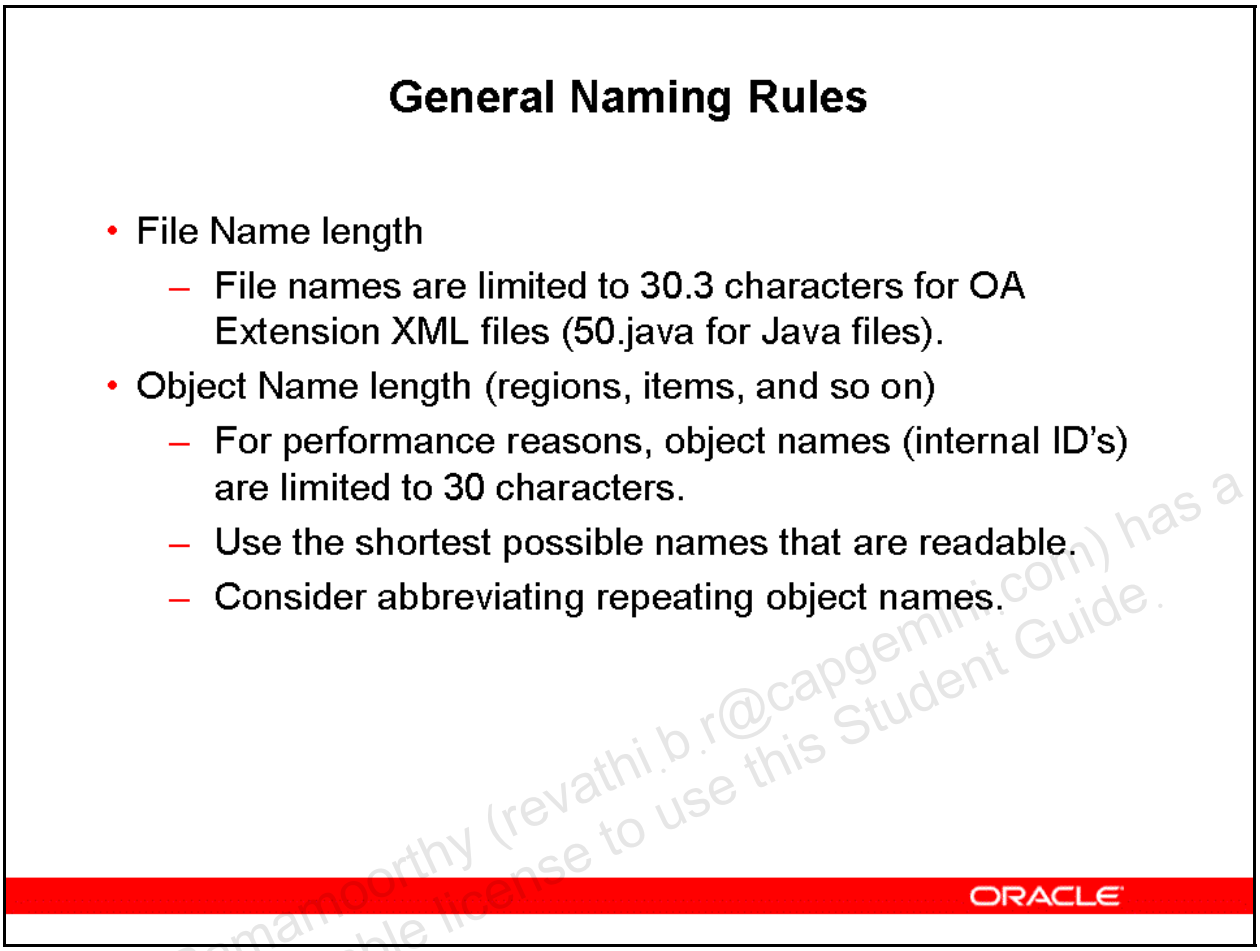
Binding Items to Data

In your declarative data, binding a specific item to data from an existing VO is simple.

- Add the VO instance to the root AM.
- Specify the root AM for the pageLayout.
- Specify the View Instance in the Property Inspector for each item.
  - Also called view object instance or view usage
- Specify the View Attribute in the Property Inspector for that item.
  - Also called view object attribute

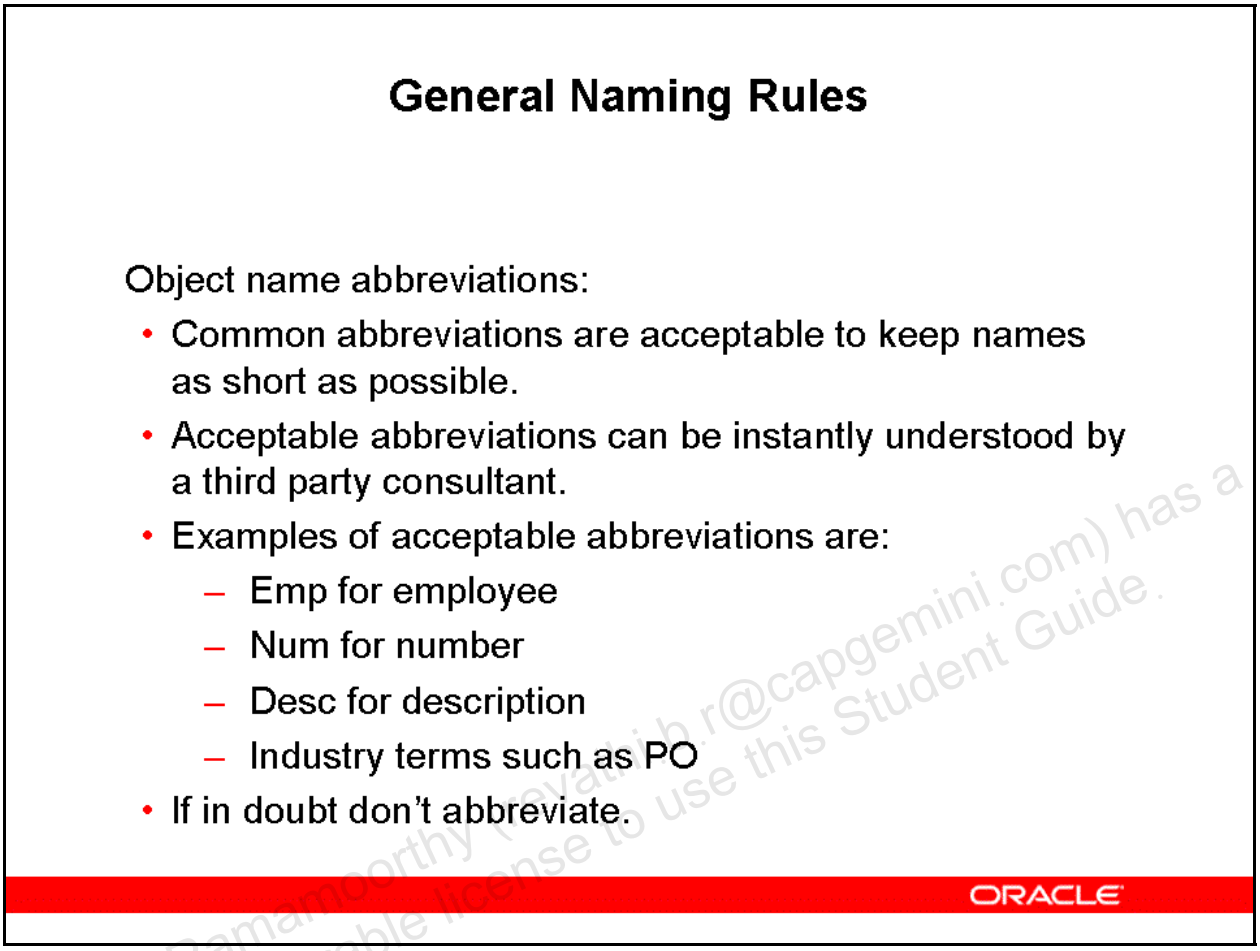
ORACLE

General Naming Rules



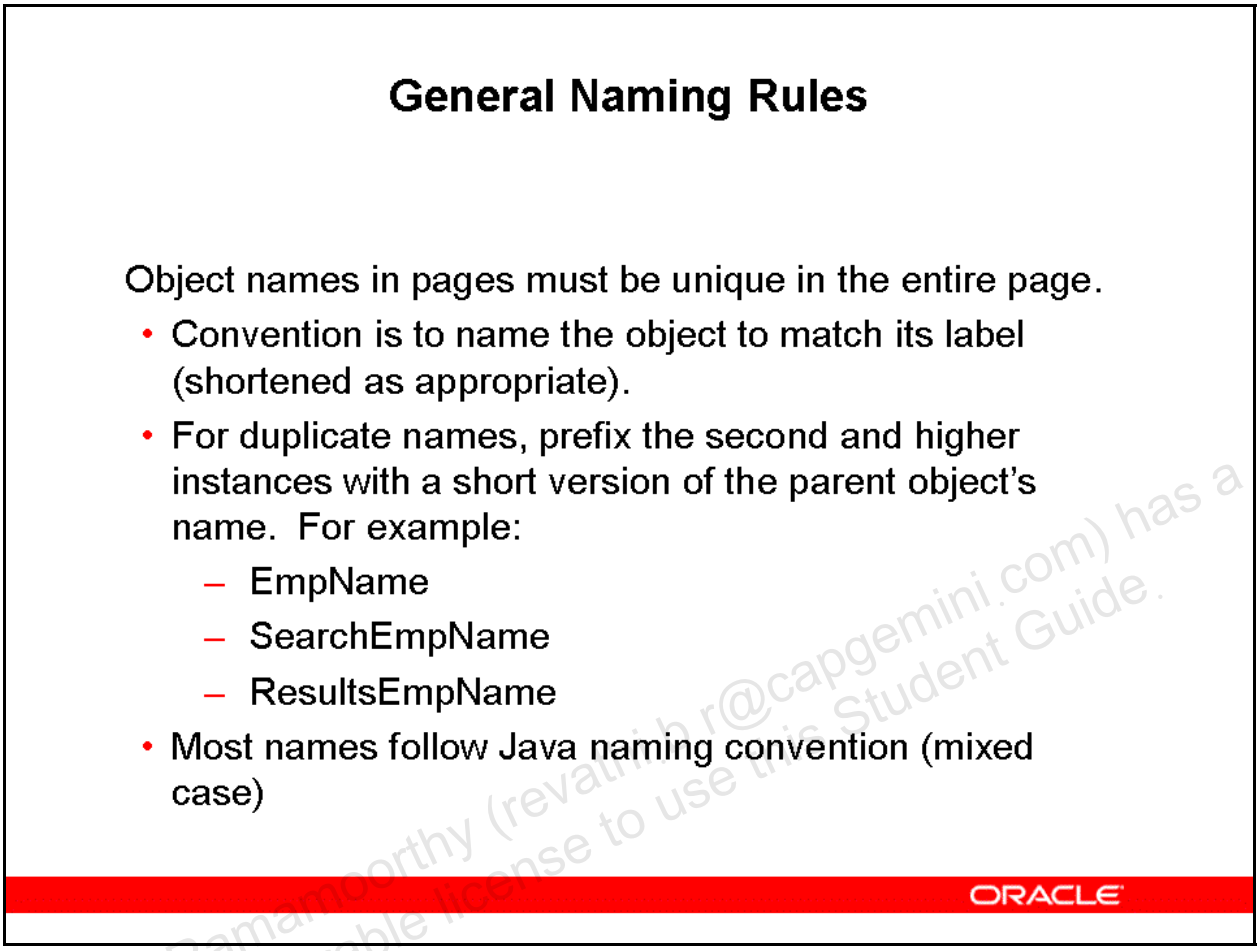
These naming rules are guidelines, and are meant to show the standards to which Oracle's E-Business Suite developers adhere.

General Naming Rules



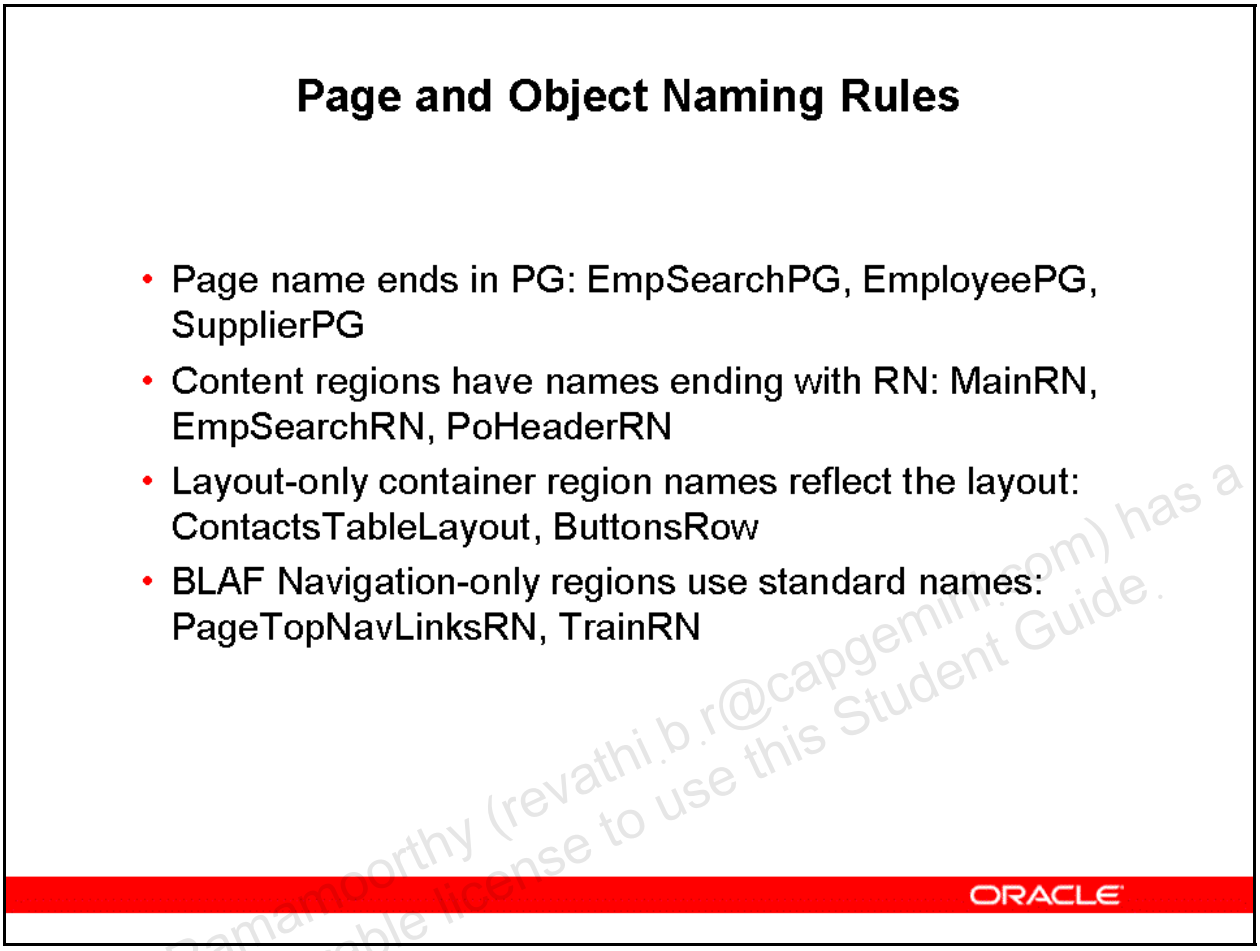
Short names are important because they reduce the size of the generated HTML file sent back to the browser. This have middle tier and network performance issues.

General Naming Rules



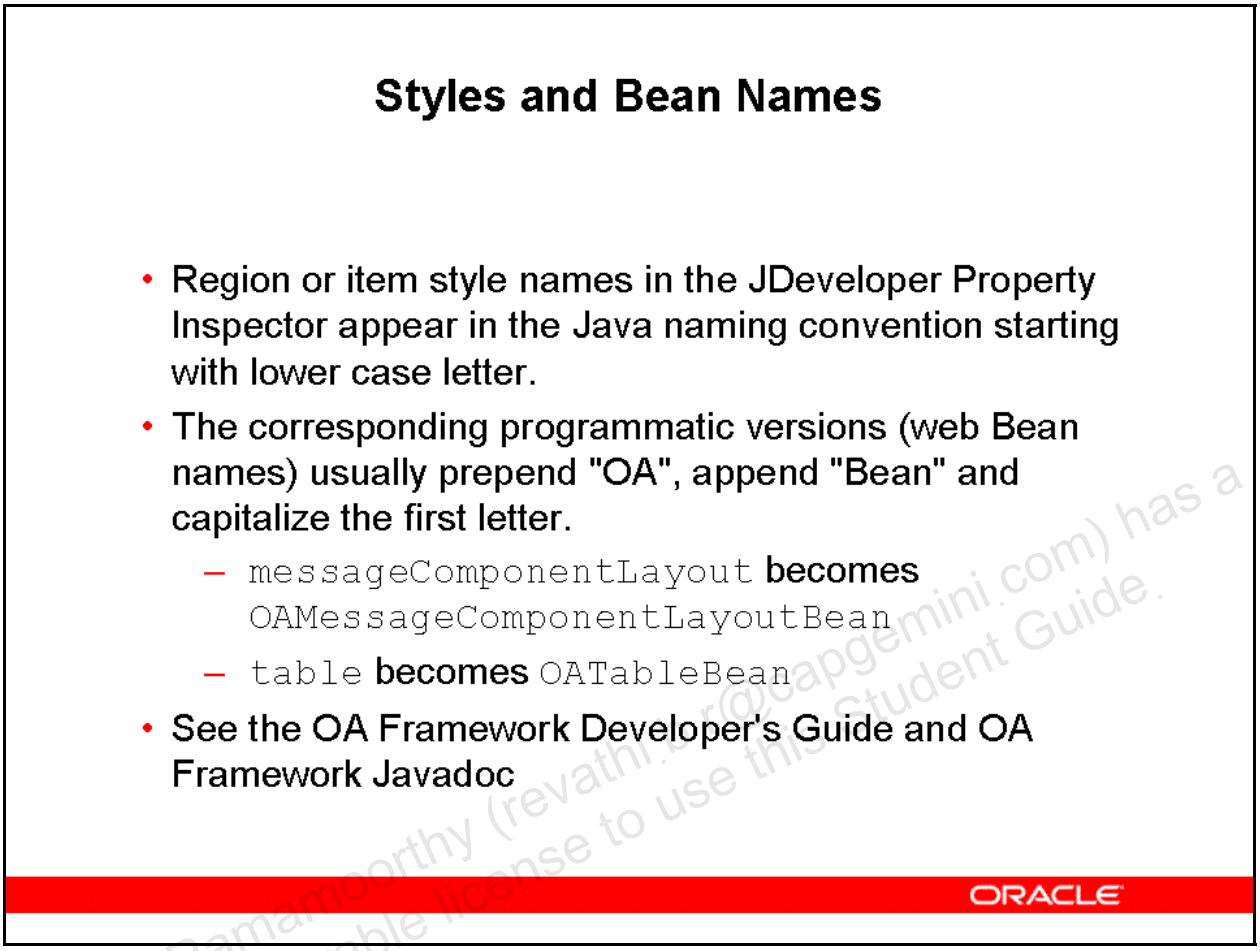
Because of OA Framework's hierarchy, names must be unique within the entire page.

Page and Object Naming Rules



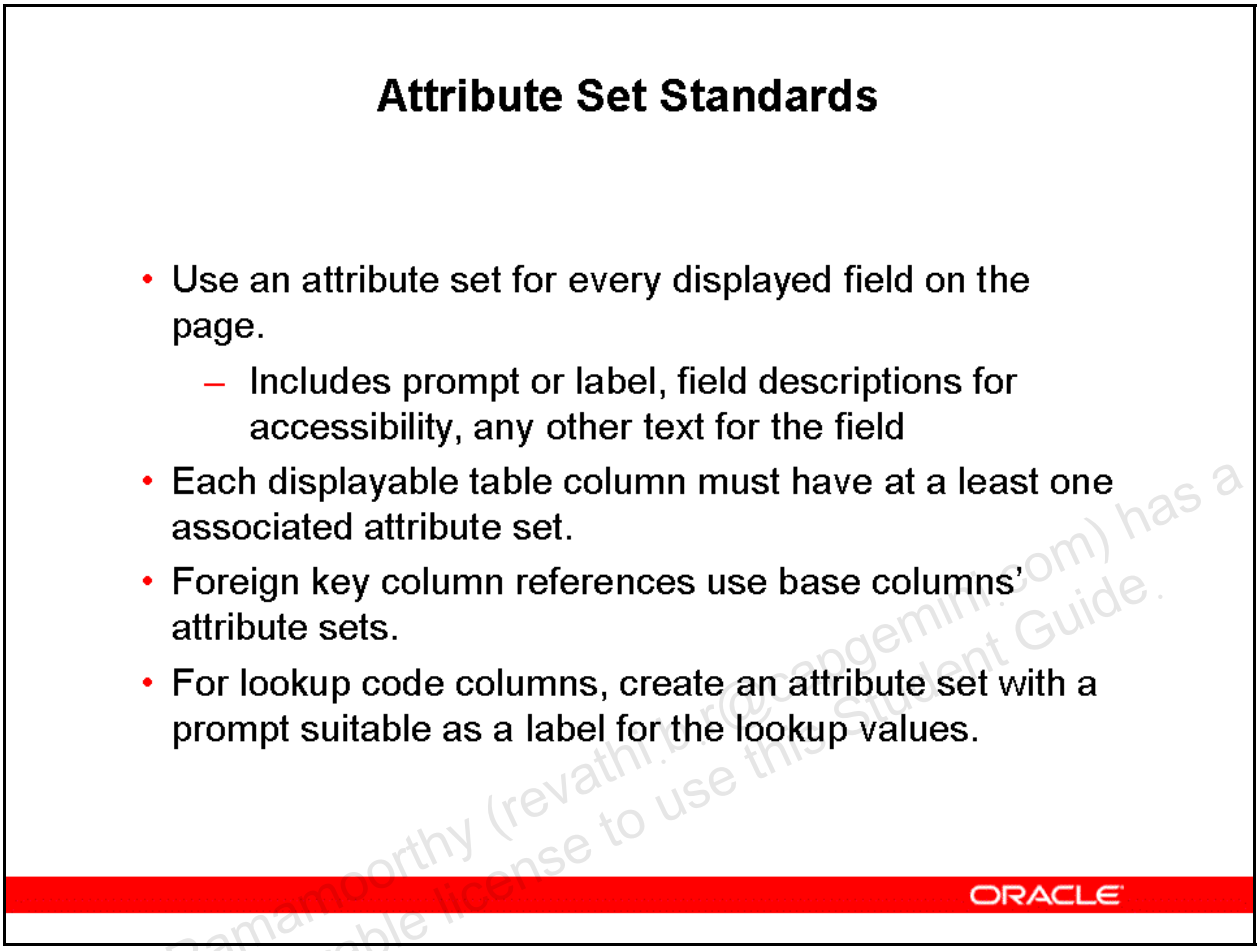
BLAF stands for Oracle's Browser-Look-And-Feel specification, and it specifies the appearance and behavior of OA Framework-based applications.

Styles and Bean Names



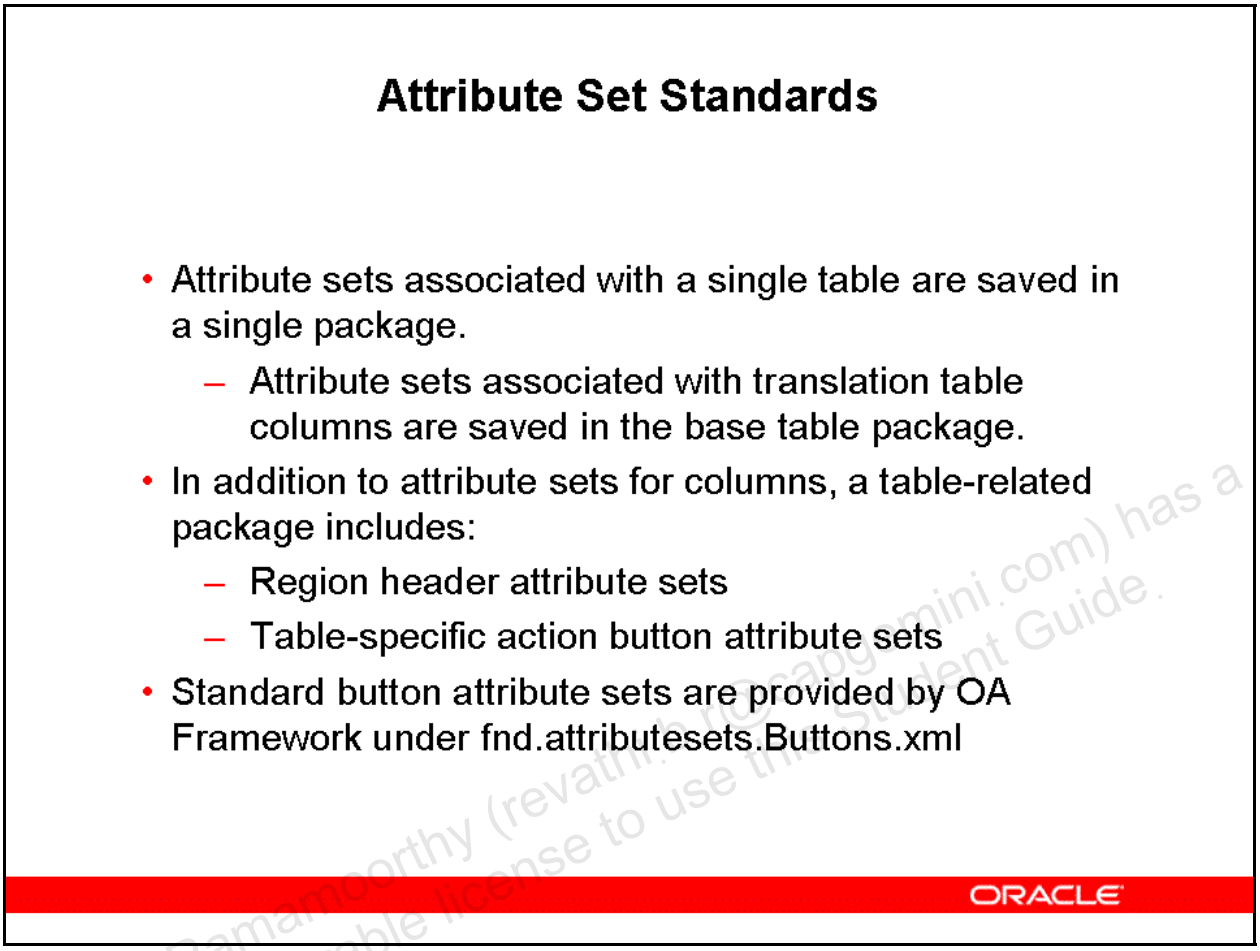
Exceptions to the rule: attachmentLink becomes OAMessageAttachmentLinkBean

Attribute Set Standards



By using Attribute Sets, Oracle significantly reduces development and translation costs.

Attribute Set Standards



Creating Attribute Set Packages Manually

Attribute sets are created into the context of an OA Component package file. To create an attribute set package in JDeveloper:

1. In the JDeveloper Navigator, select the OA Project where you want to create your package.
2. From the main menu, choose File > New to open the New Object Gallery.
3. In the Categories tree, expand the Web Tier node, and select OA Components.
4. In the Items list, select Package File to open the New Package File dialog.
5. Enter a Name and a Package. Select OK to save create your <Package>.<Name>.xml OA Component document.
6. Save your work.

Creating Attribute Sets Manually

To create and edit individual attribute sets in JDeveloper:

1. Identify which Attribute Set template you should be using based on the kind of attribute set you need to create. For example, if you are creating a button attribute set, you need to use the Button Template.

2. In the Oracle10g JDeveloper System Navigator pane, select the attribute set package to display its contents in the Structure pane.
3. In the Structure pane, select the Attribute Set folder, right-click and select New > Attribute Set to open the New Attribute Set dialog.
4. In the New Attribute Set dialog, enter an Attribute Set Name. Per the attribute set template that you identify in Step 1, select all the appropriate properties in the Available Properties list and shuttle them to the Included Properties list. Select OK to create the attribute set.
5. Enter values for the attribute set's included properties in the Property Inspector.
6. Save your work.

Generating Attribute Sets Automatically (Only on Linux)

In order to facilitate the process of creating table-related attribute set packages and attribute sets, you should automatically generate these using a command-line utility.

Prerequisites

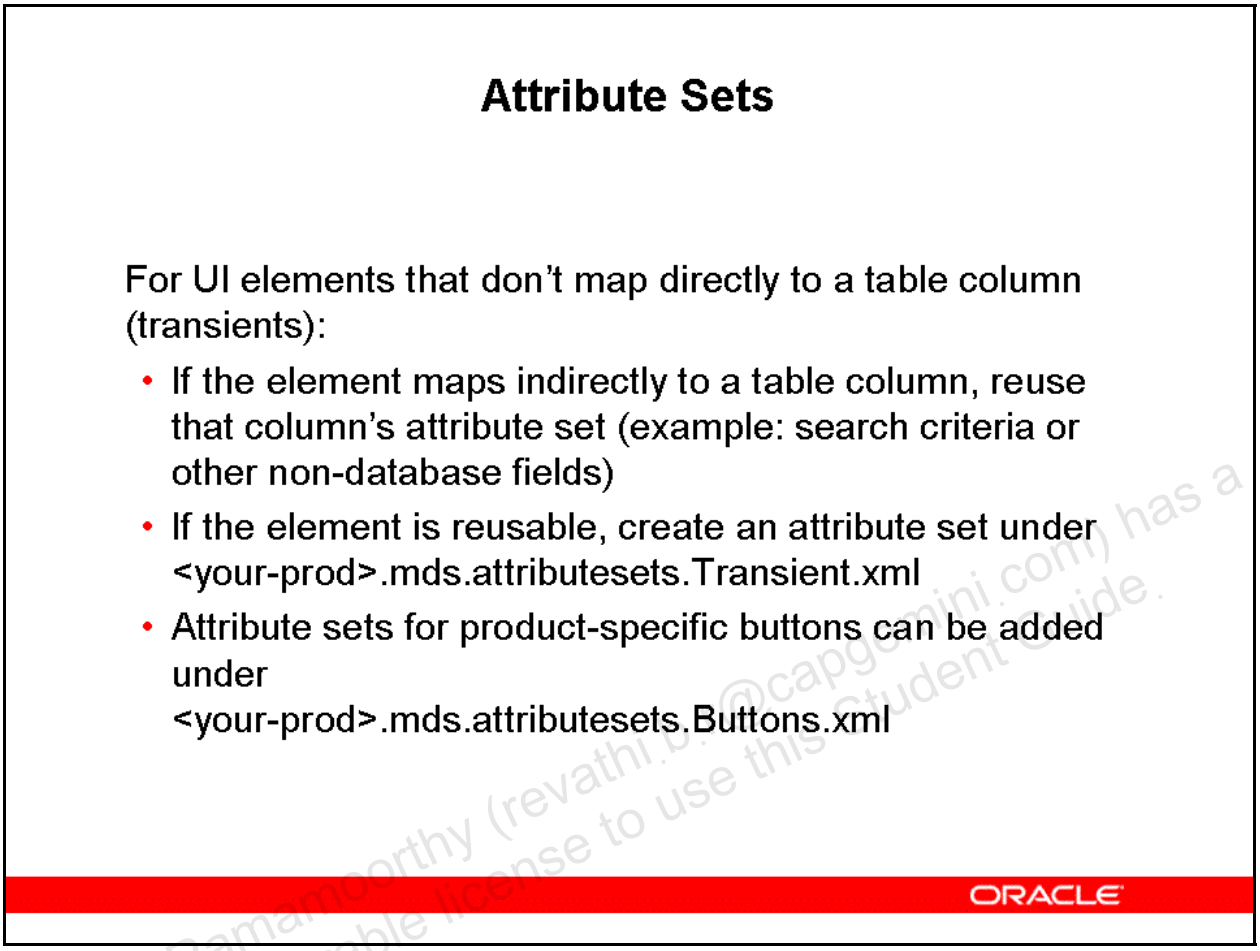
1. Before proceeding, make sure that any tables for which you want to generate attribute sets are fully described in FND_TABLES and FND_COLUMNS (to do this, you need to load properly defined XDF table definitions). The table name, column name and column descriptions defined in these entities become the basis for the generated attribute sets.
2. You should check out any preexisting attribute set packages from source control that you want to update with the generator. Note the physical location of the XML files in your working area.

Running the Attribute Set Generator

To automatically generate new and updated existing attribute sets, run the Attribute Set Generator from the Linux command line as shown below. This utility will create new attribute sets, and update preexisting attribute set Comments property values with the corresponding column descriptions in FND_COLUMNS. You can elect to generate attribute sets for an entire product, or for specific tables within a product. Once you generate your attribute sets, you can maintain all other properties in JDeveloper as described above (the generator never overrides any other property values).

Note: For persistent attributes, the Comments property value is always copied from FND_COLUMNS (the column description value in FND_COLUMNS should be treated as the single source of truth). If you need to make changes to the Comments property value, you need to change the underlying column description.

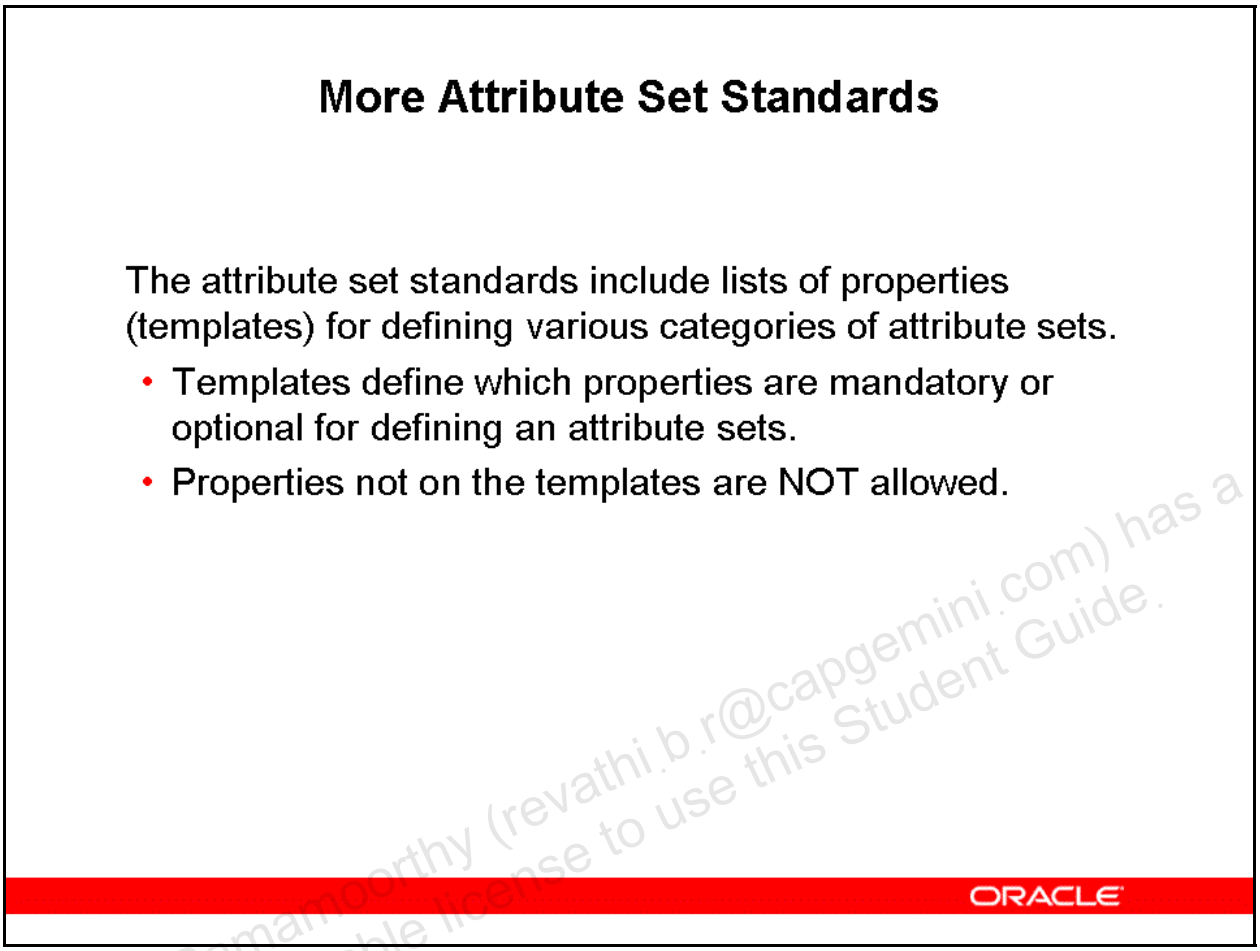
The resulting attribute set packages and attribute sets are named according to standards. Correct naming and packaging is essential for service creation since the OA Framework automatically detects table.column attribute sets using these naming conventions, and defaults them when you create view object attributes that map to persistent entity attributes.



Attribute Set Naming Conventions

- For column-related attribute sets, change column name to use Java class naming style.
  - For example, PARTY_NAME column name becomes PartyName.
- When a column is used in different contexts, use multiple attribute sets.
  - 1st attribute set name is derived from column name and considered the default (such as PartyName, which contains the most common prompt, Organization)
  - 2nd and higher attribute sets add an *_OtherPrompt* suffix (such as PartyName_Partner)

ORACLE



Summary

In this lesson, you should have learned how to:

- Use the proper terminology when discussing OA Framework UI (View-layer) objects.
- Discuss pages.
- Discuss regions.
- Discuss items.
- Discuss attribute sets.
- Discuss CSS styles.
- Discuss other View-layer objects.
- Discuss View-layer read and write data binding.
- Identify and use View-layer naming standards.

ORACLE

Basics of the Controller

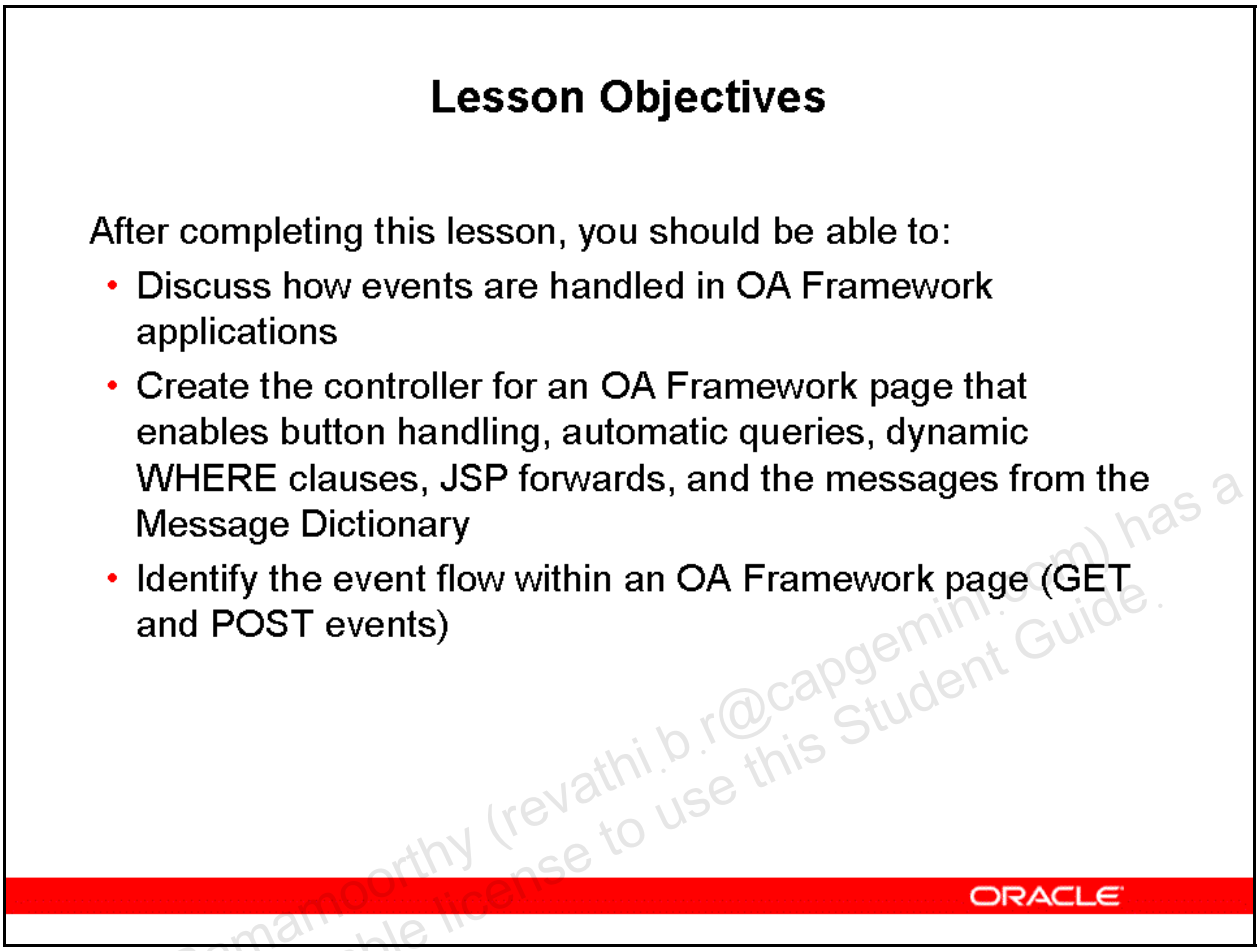
Chapter 5

R12.x Extend Oracle Applications: Building OA Framework Applications

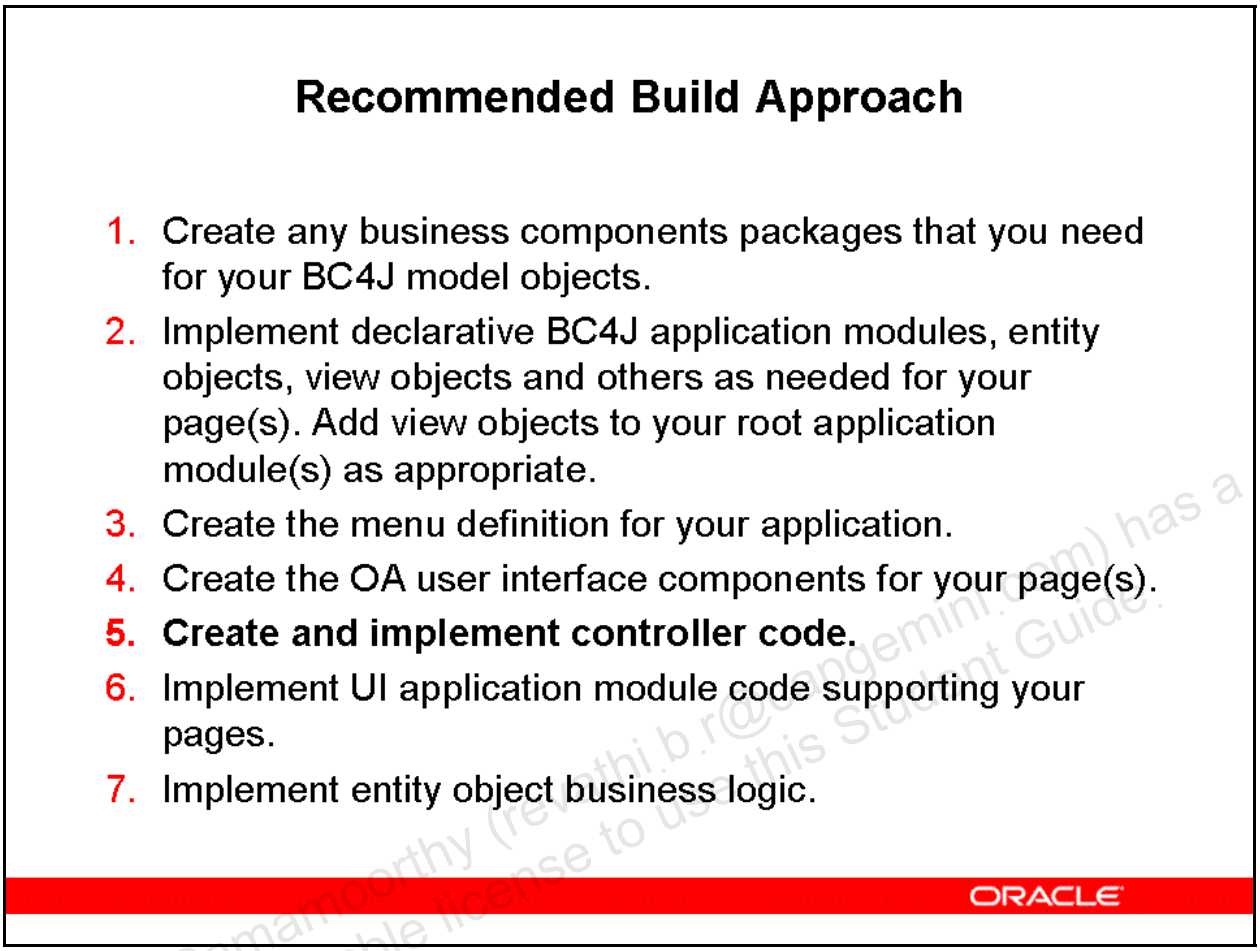
Basics of the Controller

ORACLE

Lesson Objectives

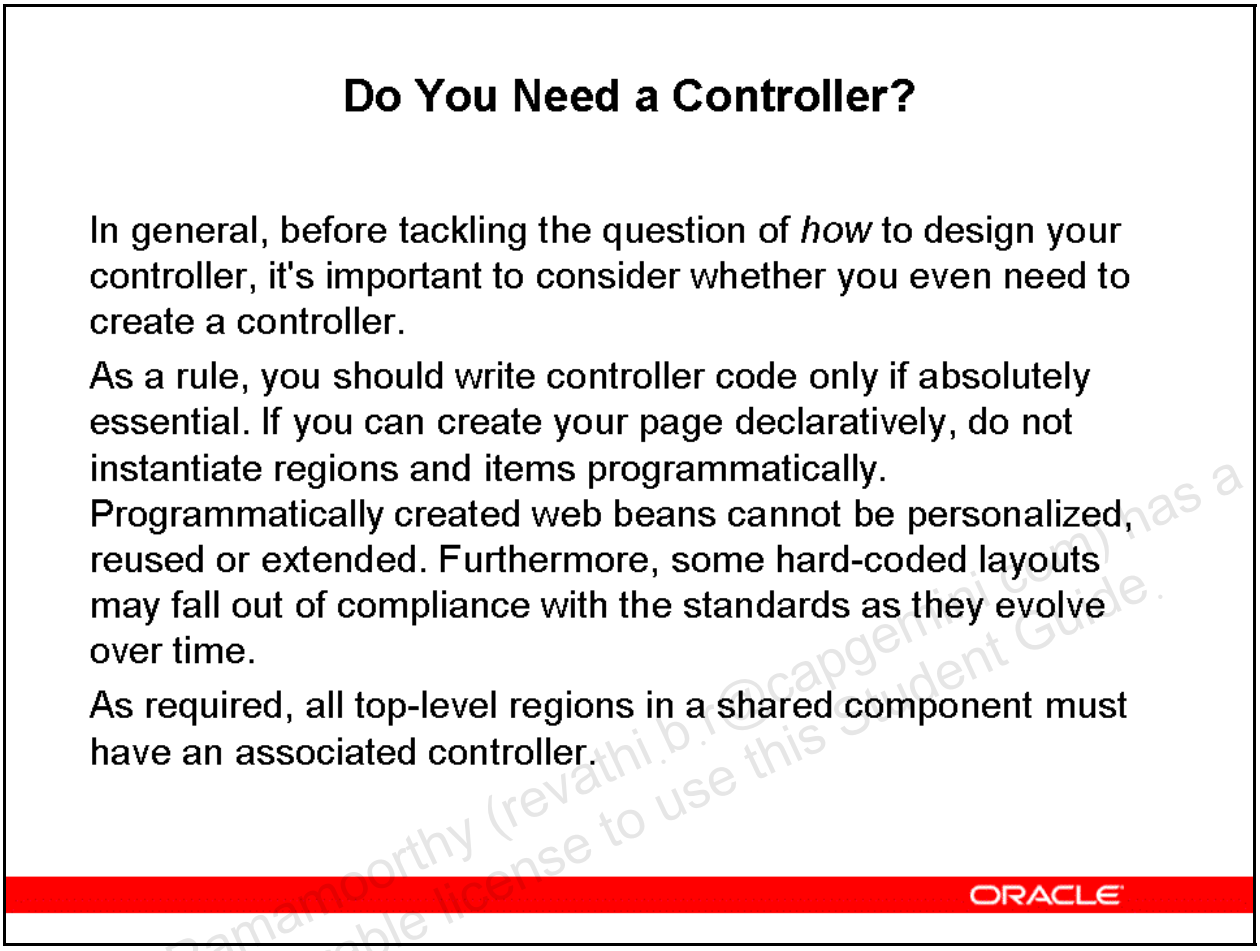


Recommended Build Approach



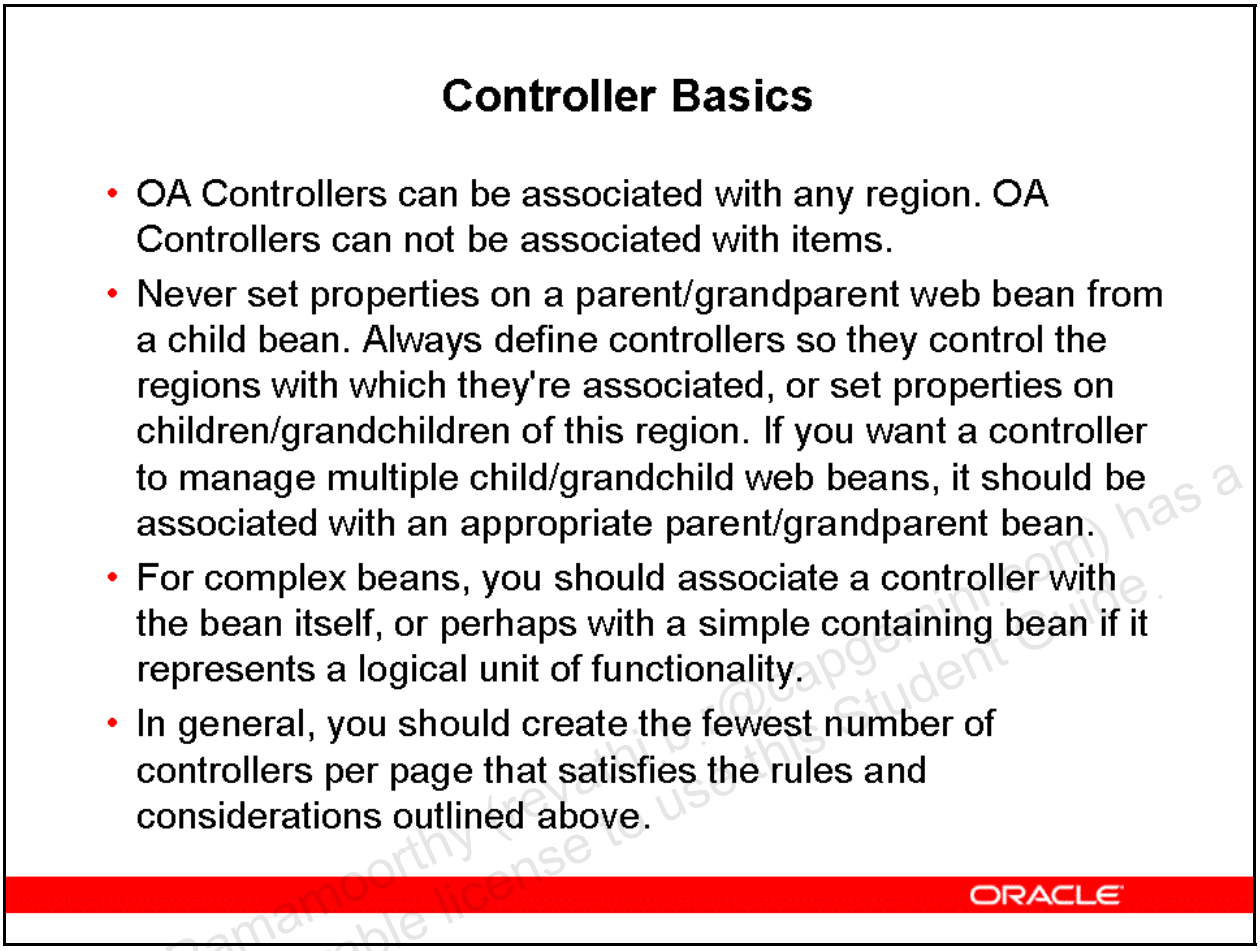
Steps 5 deals directly with Controller layer components.

Do You Need a Controller?

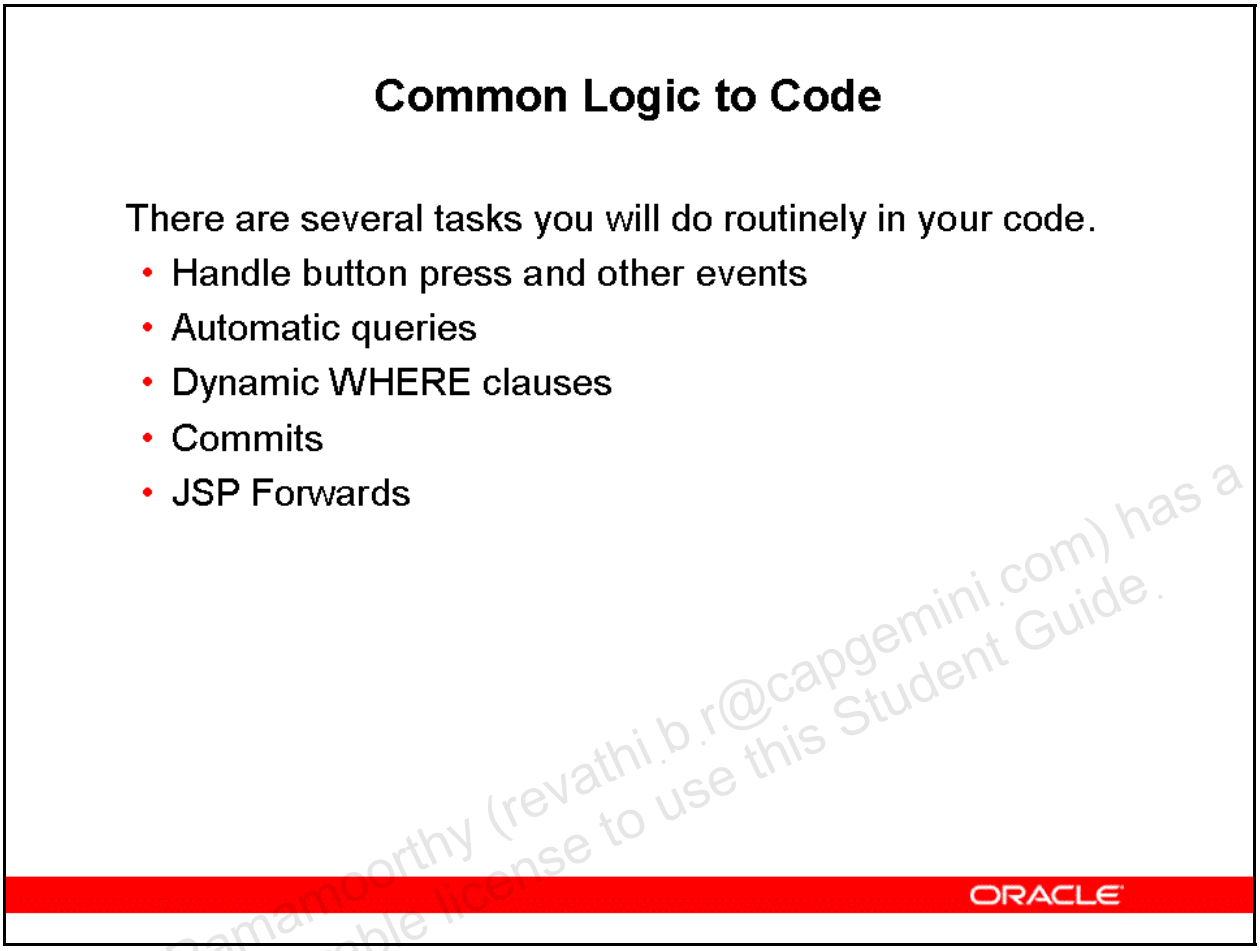


In a J2EE design pattern, the controller is designed for application flow. In Oracle Application Framework (OAF), controller code is primarily used in that manner. However, UI components can be instantiated through the controller. Components built within a controller are not personalizable. There are no errors or visual cues that a component has been created via a controller. Because of this, it is not advisable or supported to use a controller for anything other than application flow or transaction - based functionality.

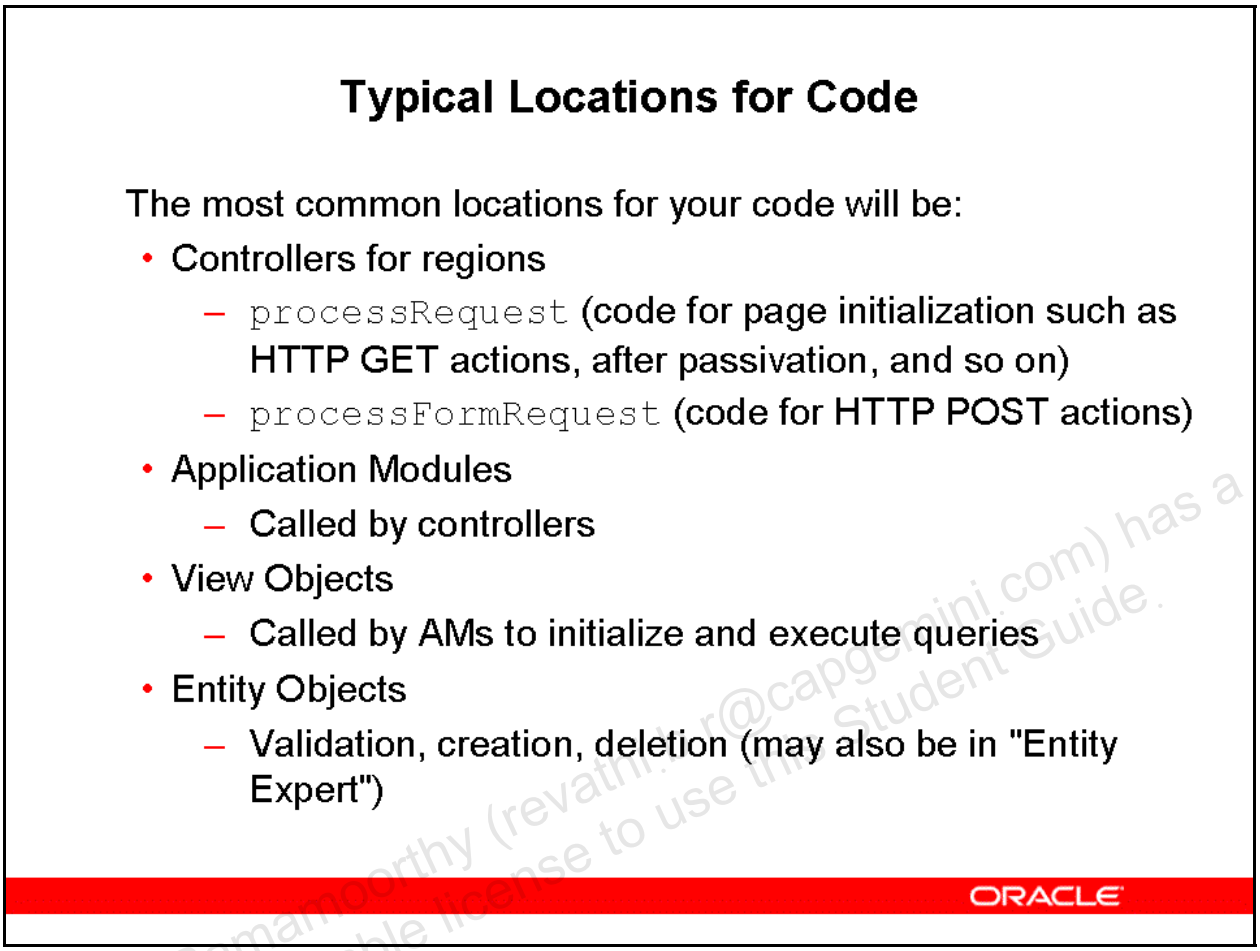
Controller Basics



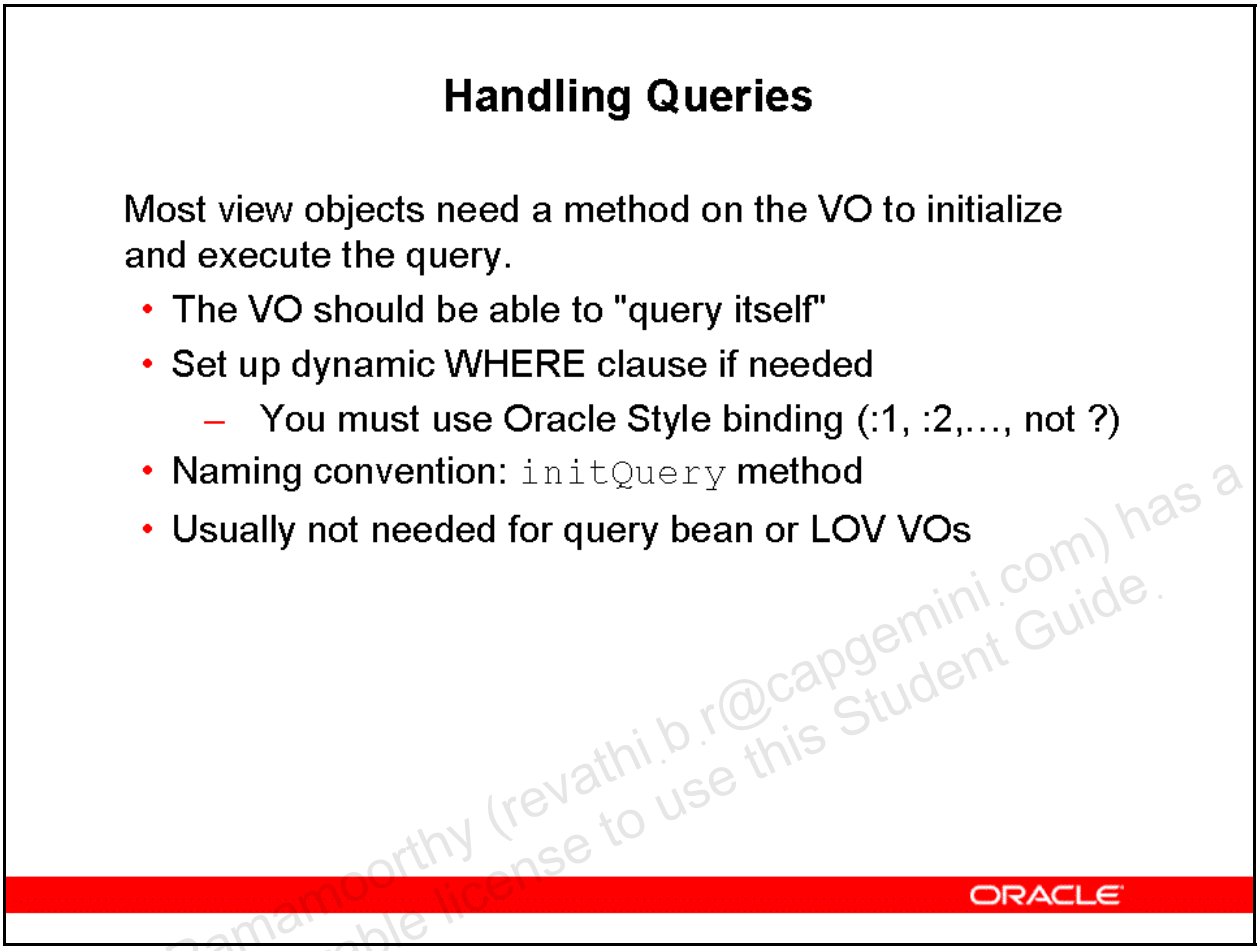
A general rule is to minimize the number of controllers. While it is possible for every region or object to have a controller, it is not advisable to do so. Minimizing regions with controllers, can simplify the code for your OA Framework page and make the page more maintainable.



Typical Locations for Code

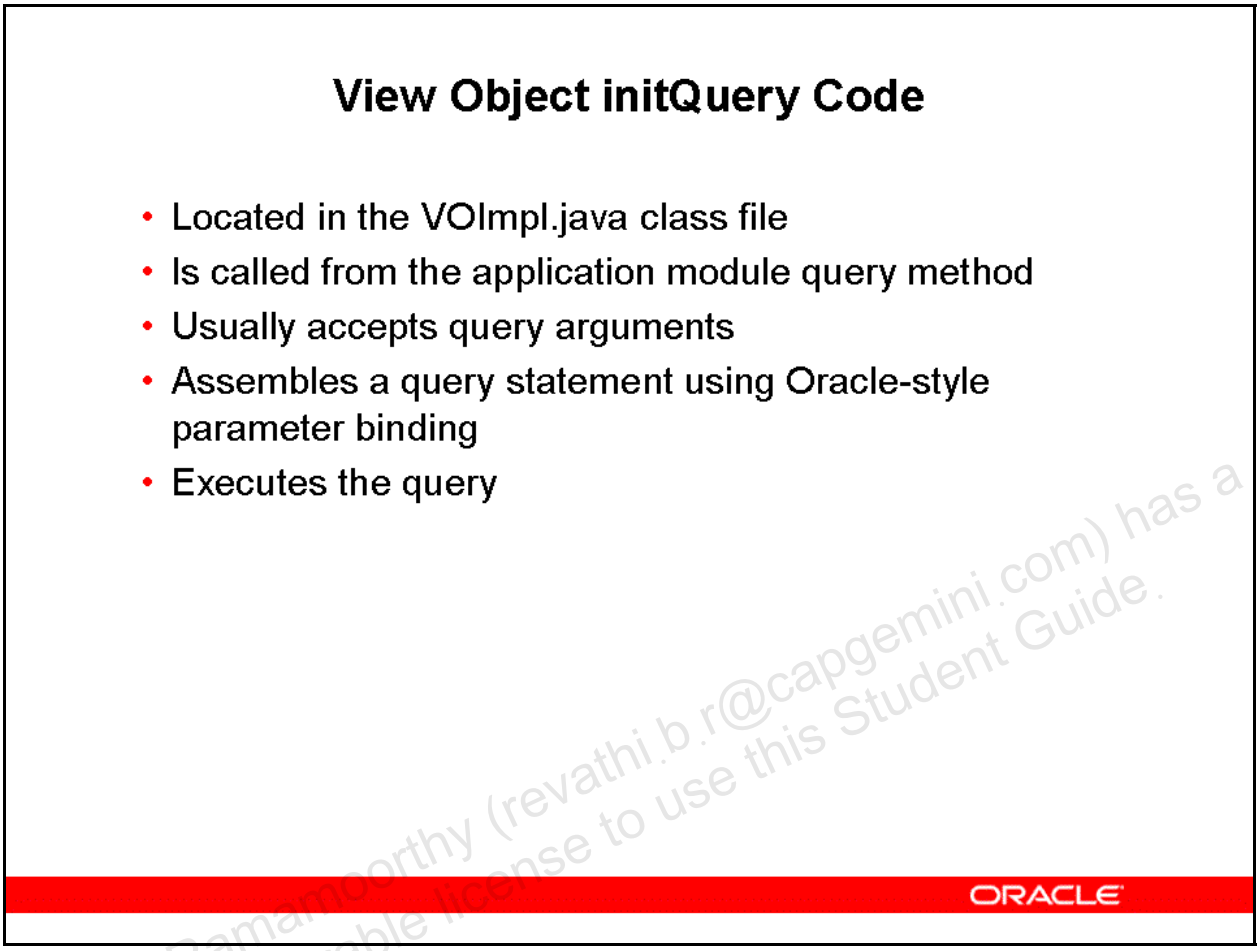


The code itself will typically be in the <AM/VO/EO>Impl.java file or a <controller name>CO.java file.



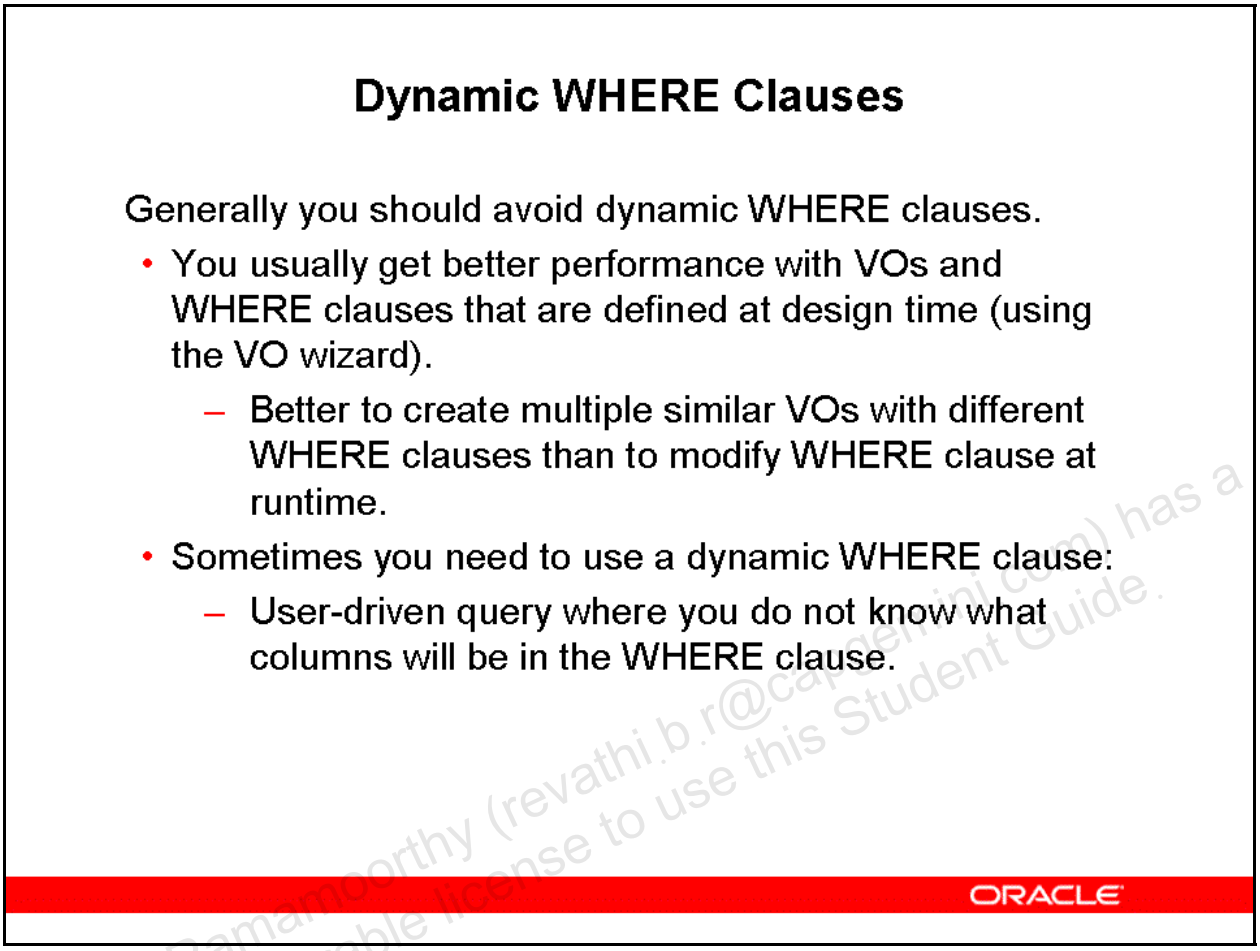
This is the most common use of a controller.

View Object initQuery Code



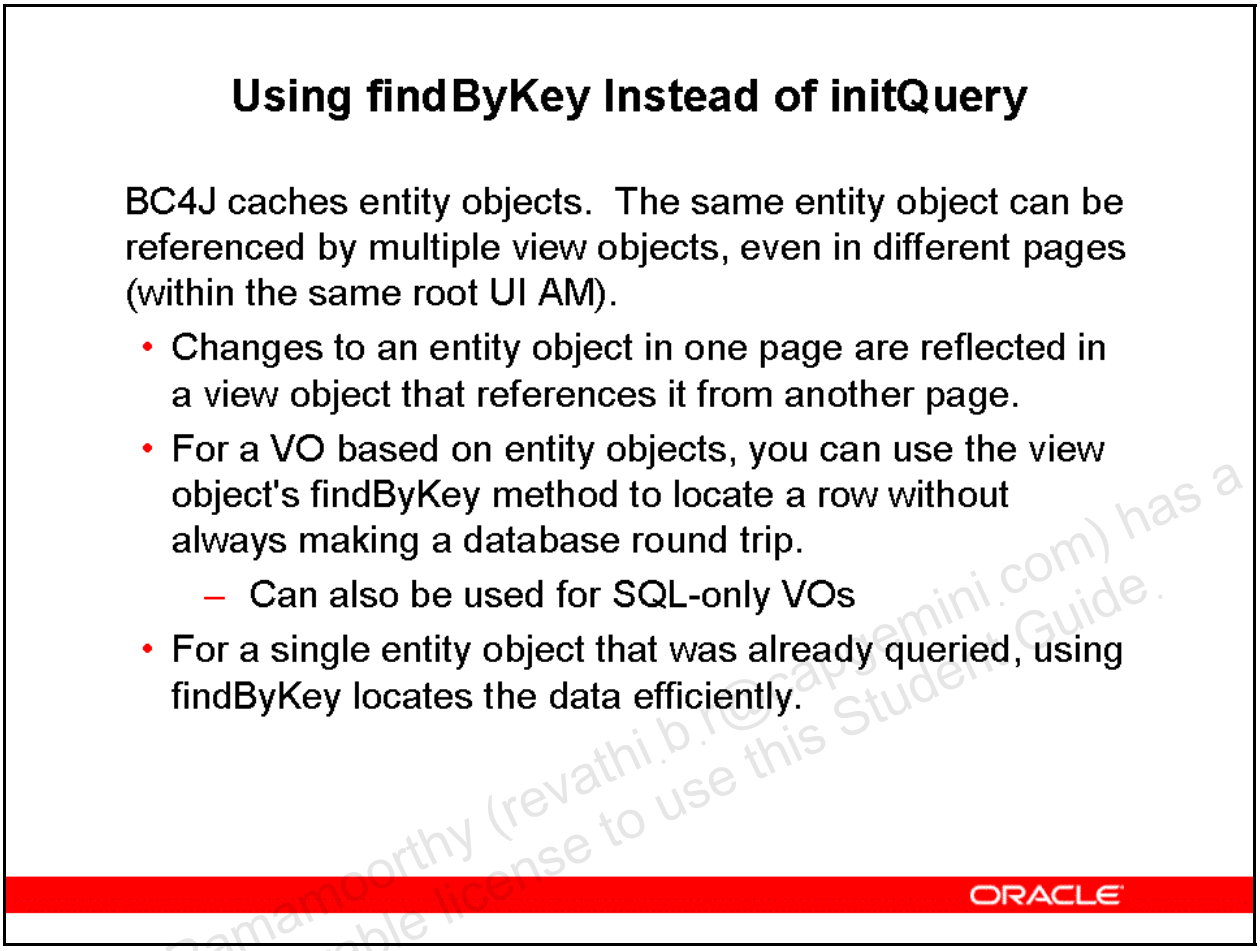
Remember to keep to the encapsulation. The controller should call the AM, and defer to the AM to call the necessary VO.

Dynamic WHERE Clauses



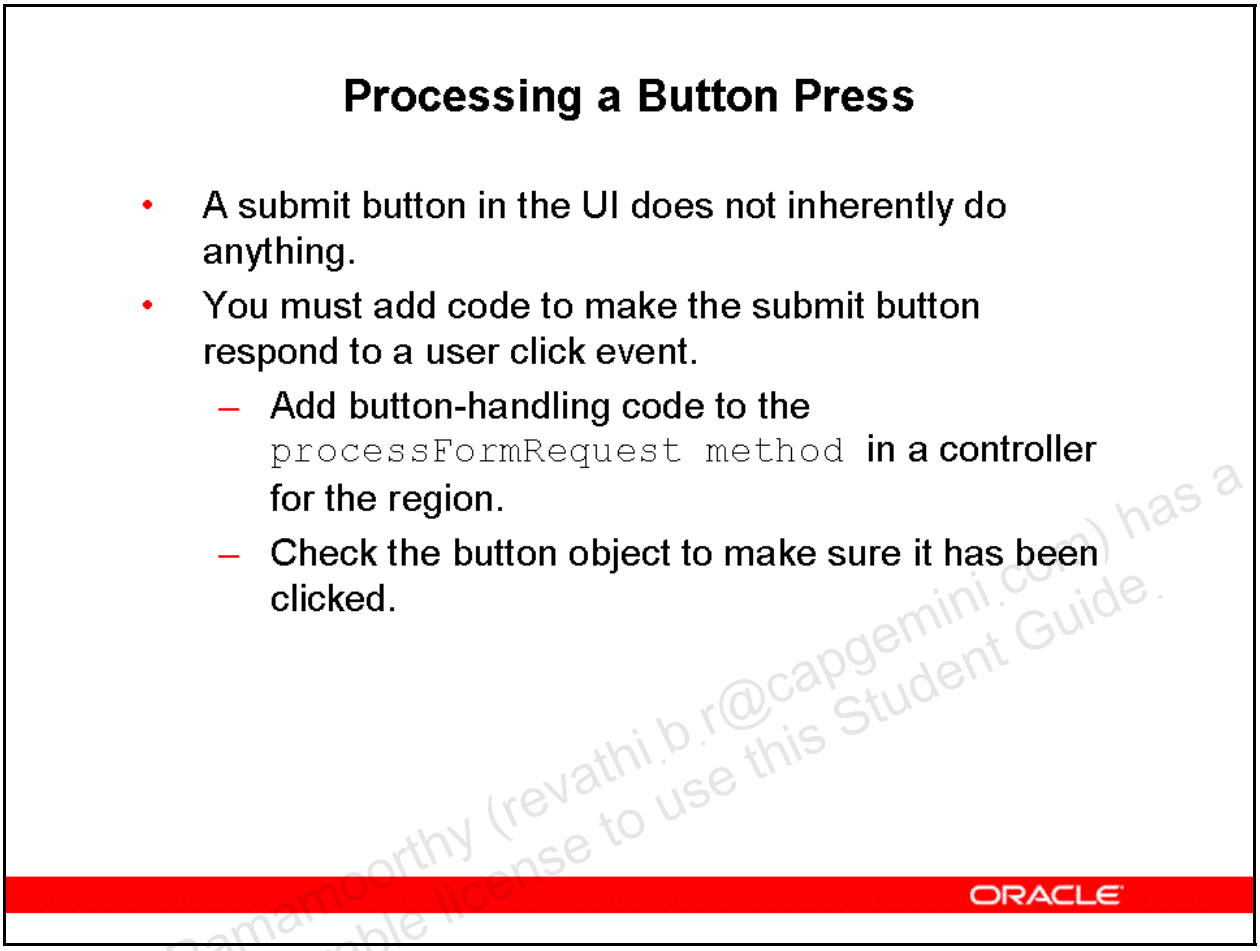
This is a good general rule. Where possible, keep things defined at design time (declarative) instead of at run time (programmatic). Declarative objects have much better performance.

Using findByKey Instead of initQuery

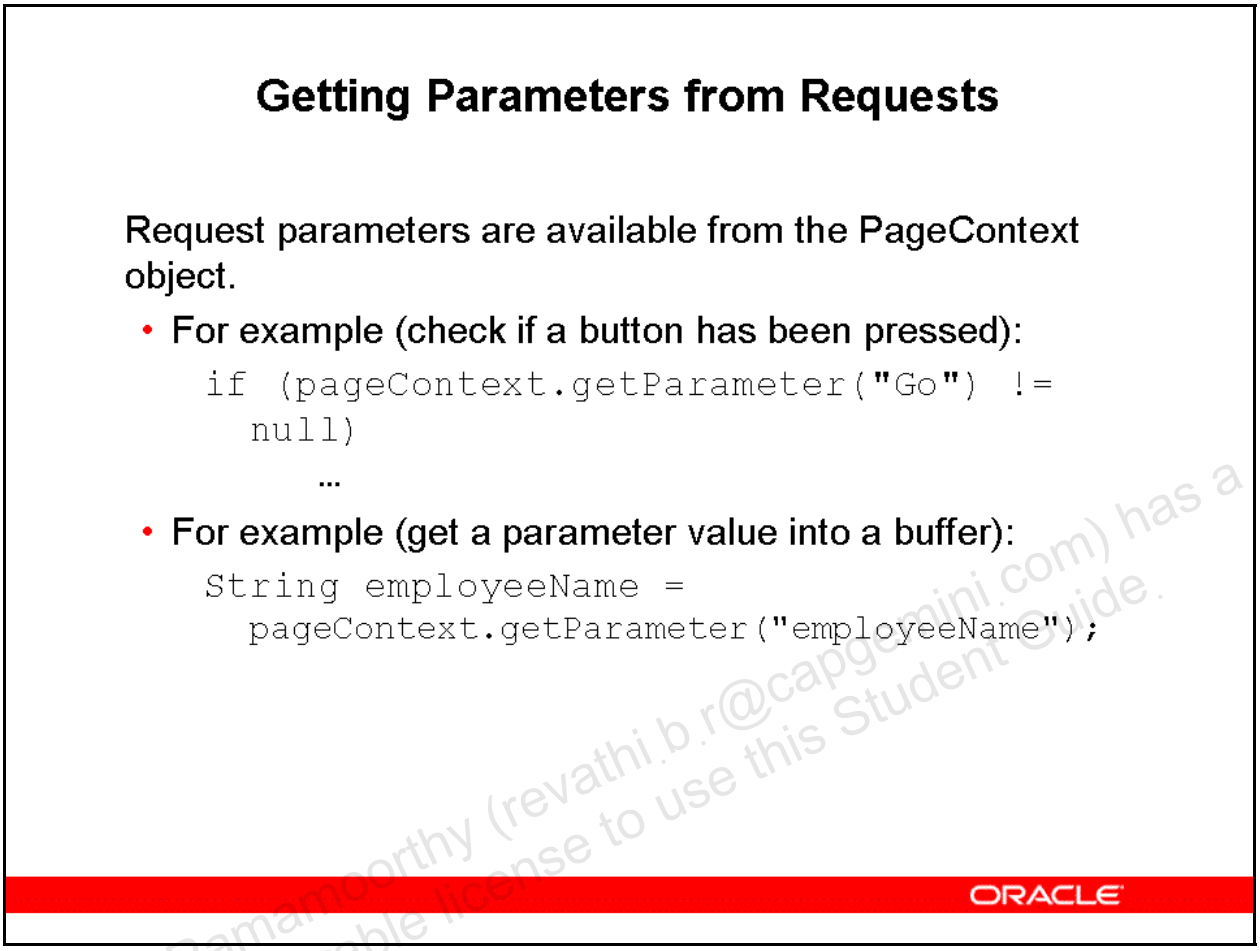


The findByKey is useful when you are looking for a single row whose entity objects are likely to already be in the cache (BC4J will pluck them from the cache instead of doing a round trip to the database). This is a very useful tool for developers and we want to encourage them to use it.

Processing a Button Press

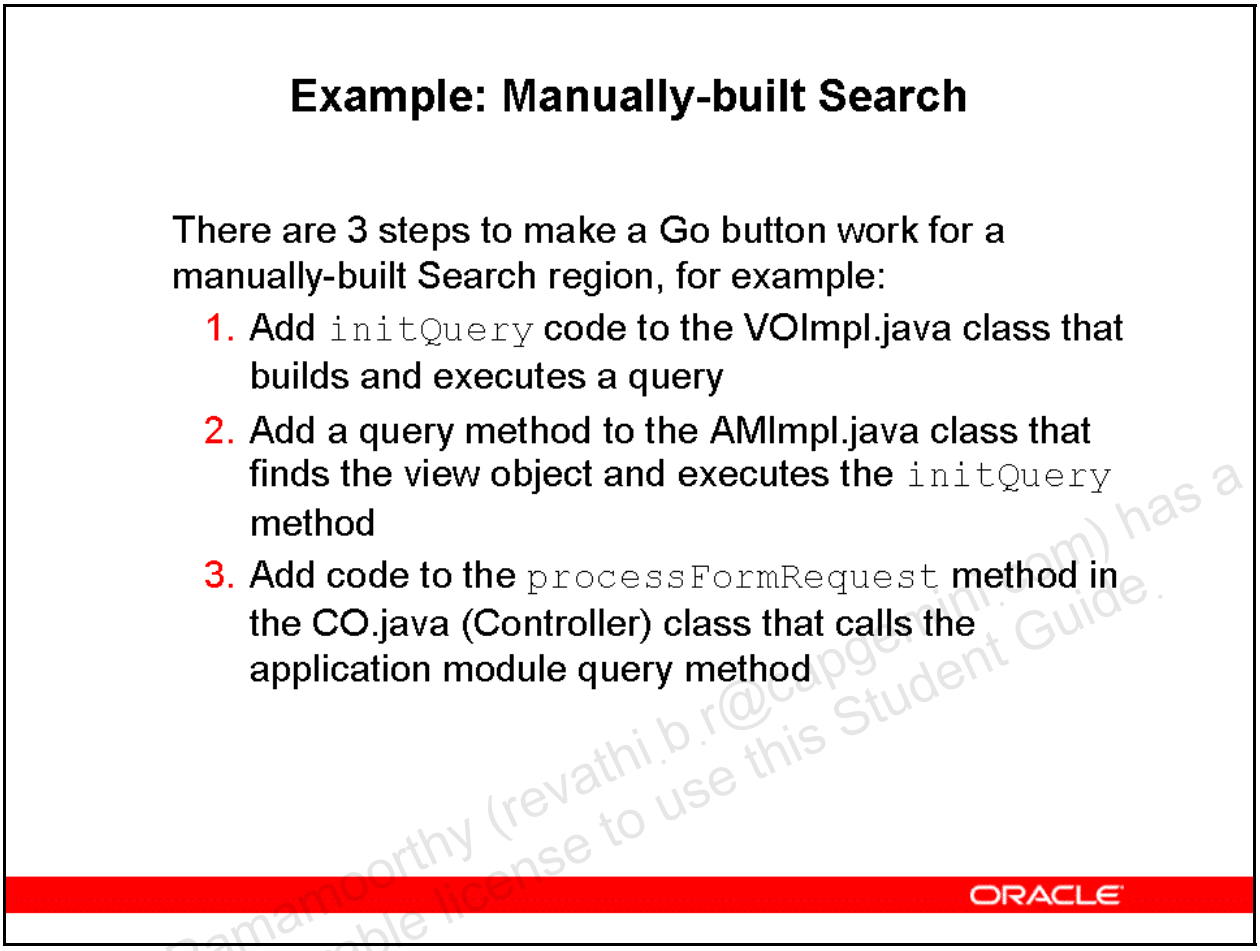


Button presses and other events are part of the POST cycles spoken about earlier.



Note: Java and XML are case sensitive. This is a common error in OA Framework pages.

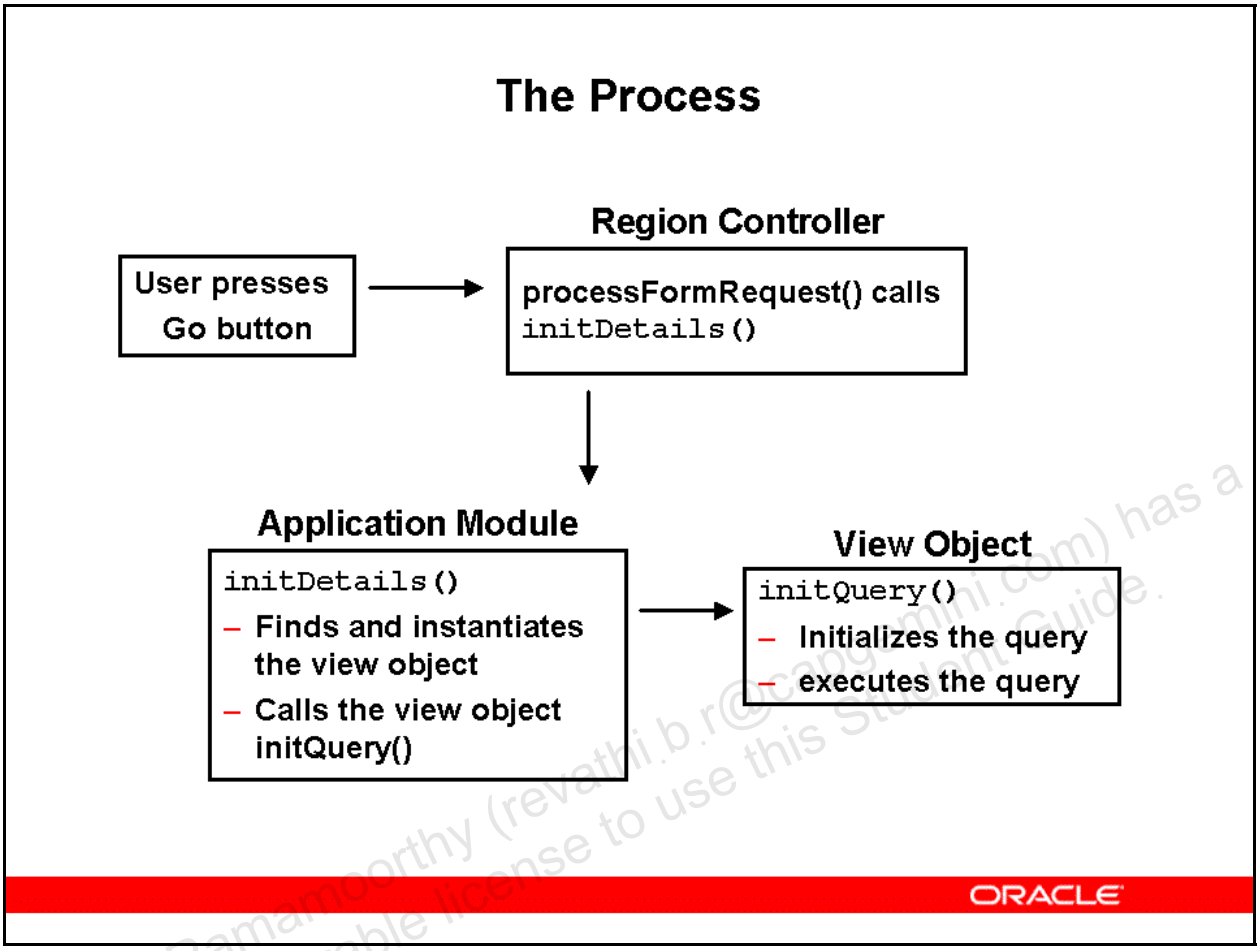
Example: Manually-built Search



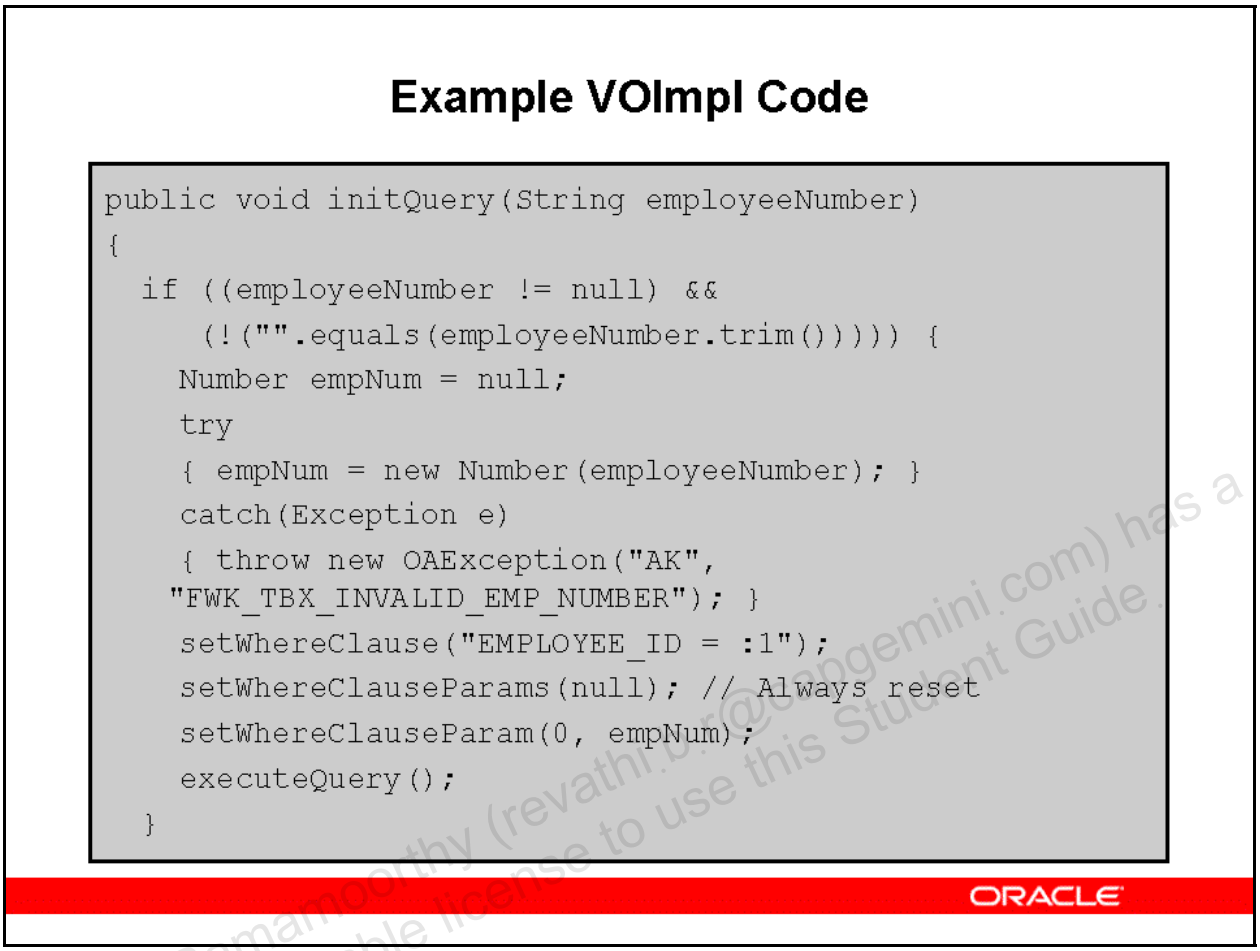
The processFormRequest in the controller should never call methods on the VO directly; it should always delegate to the AM.

This same code, added to the processRequest method instead of the processFormRequest method, can be used to do an automatic query when the user comes into the page.

The Process



Example VOImpl Code



The above code gives an example of code that would be put into the VOImpl.java class. Note the following characteristics:

The method is named initQuery(). This is a naming standard.

The parameter passed to the method (employeeNumber) is not assumed to be valid number. There is a try/catch to typecast the String object to a Number object.

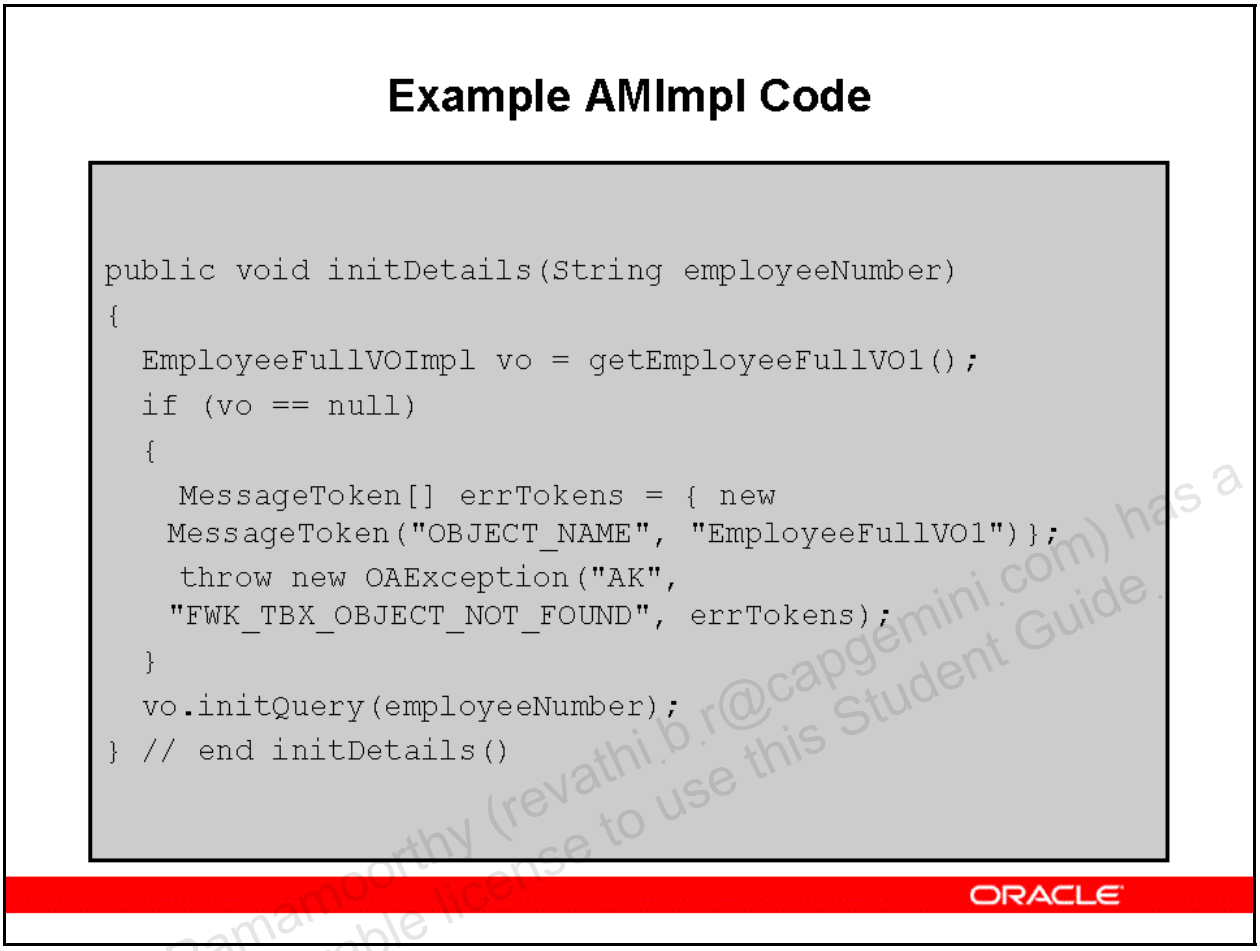
If the typecast fails, an OAException is thrown from the message dictionary.

The remainder of the code sets up the dynamic WHERE clause for the VO, and then executes the query.

If you do not have the executeQuery(), the VO will not query your data. This is a common error. Sometimes, you will, when creating the VO, accidentally or unintentionally check the generate VOImpl.java class. The VOImpl.java class file is

just an empty stub. Without an initQuery method with a call to executeQuery, the data will not be retrieved.

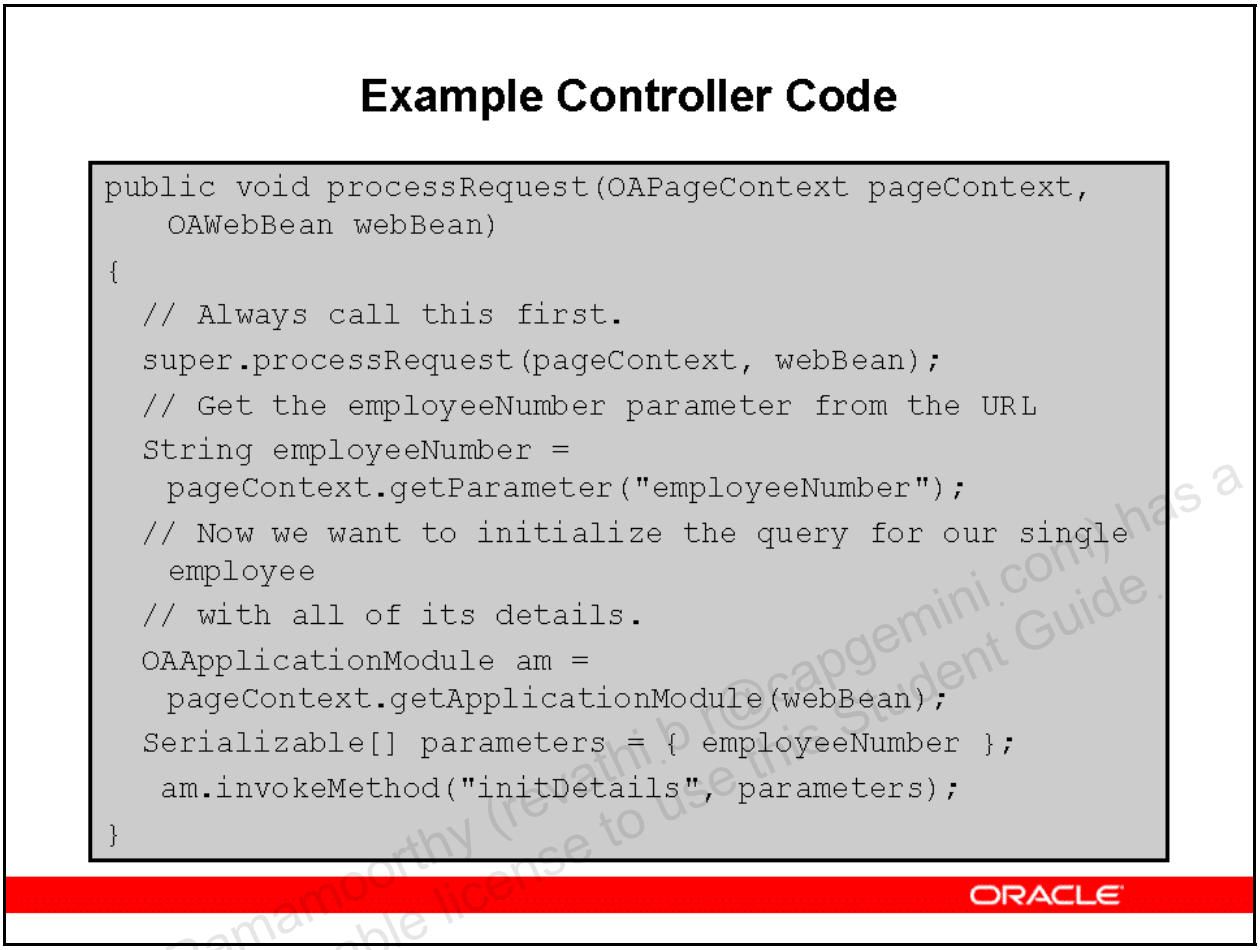
Example AMImpl Code



The above code gives an example of the code that would be put into the AMImpl.java class. Note the following characteristics:

- The method checks to see if the VO is assigned to the AM.
- If the VO is not known to the AM, an OAException is thrown.
- The AM then invokes the initQuery method on the VO passing any parameters needed.

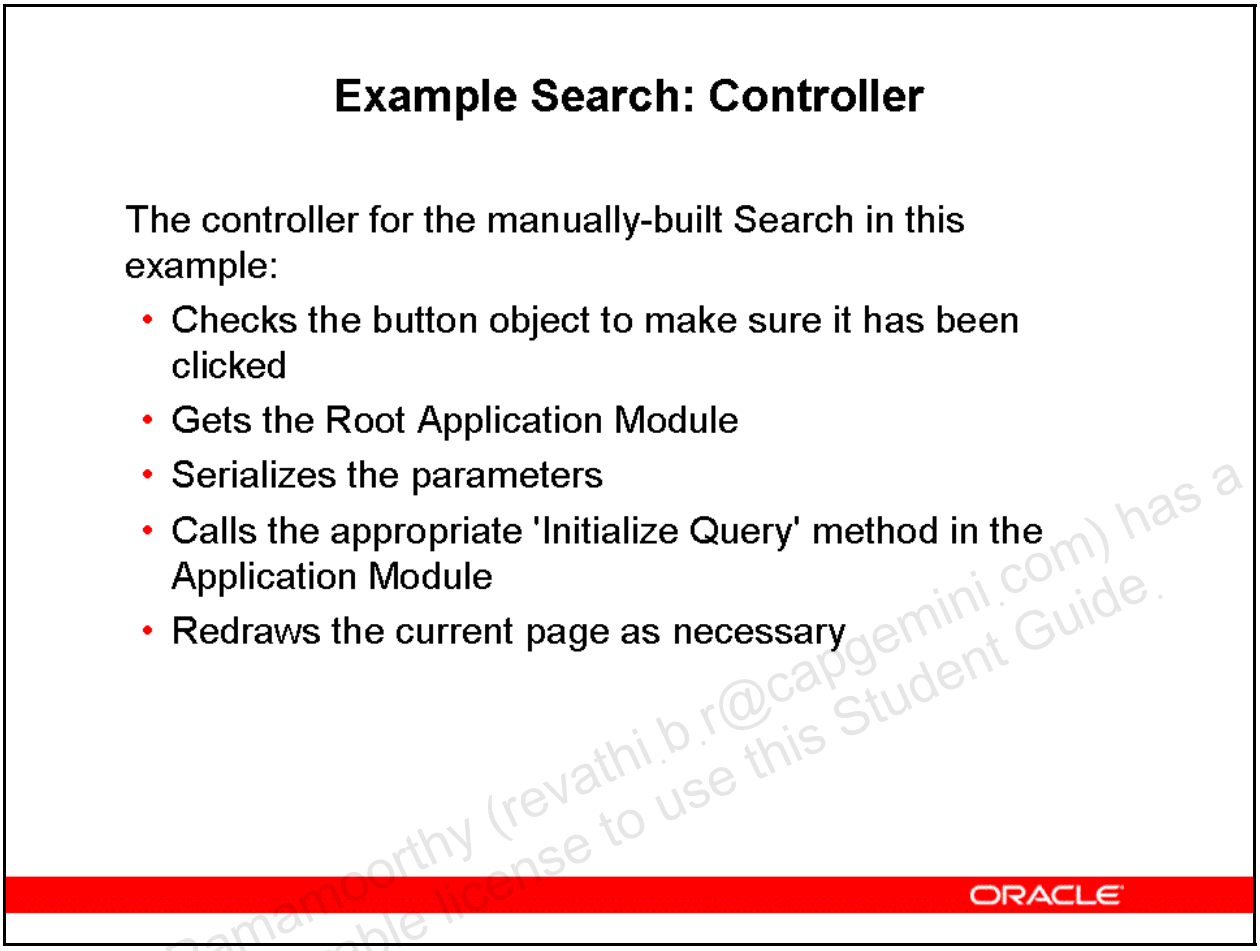
Example Controller Code



The above code gives an example of the code that would be put into the CO.java (Controller) class. Note the following characteristics:

1. The first line of the method is a call to super.
2. The controller knows the UI objects, and gets their values from the pageContext.
3. The pageContext knows what its root AM is. So, the controller asks for the root AM from the pageContext.
4. Parameters are serialized. For this example, this is simple. Serialization is the process of converting an object into a string representation. Since the parameter is already a string, this is simple. In some cases, more complex objects may be passed as parameters to the method.
5. Finally, the AM method is invoked. Since the AM can contain multiple VOs, it is not possible, like initQuery for VOs, to call a standard AM method.

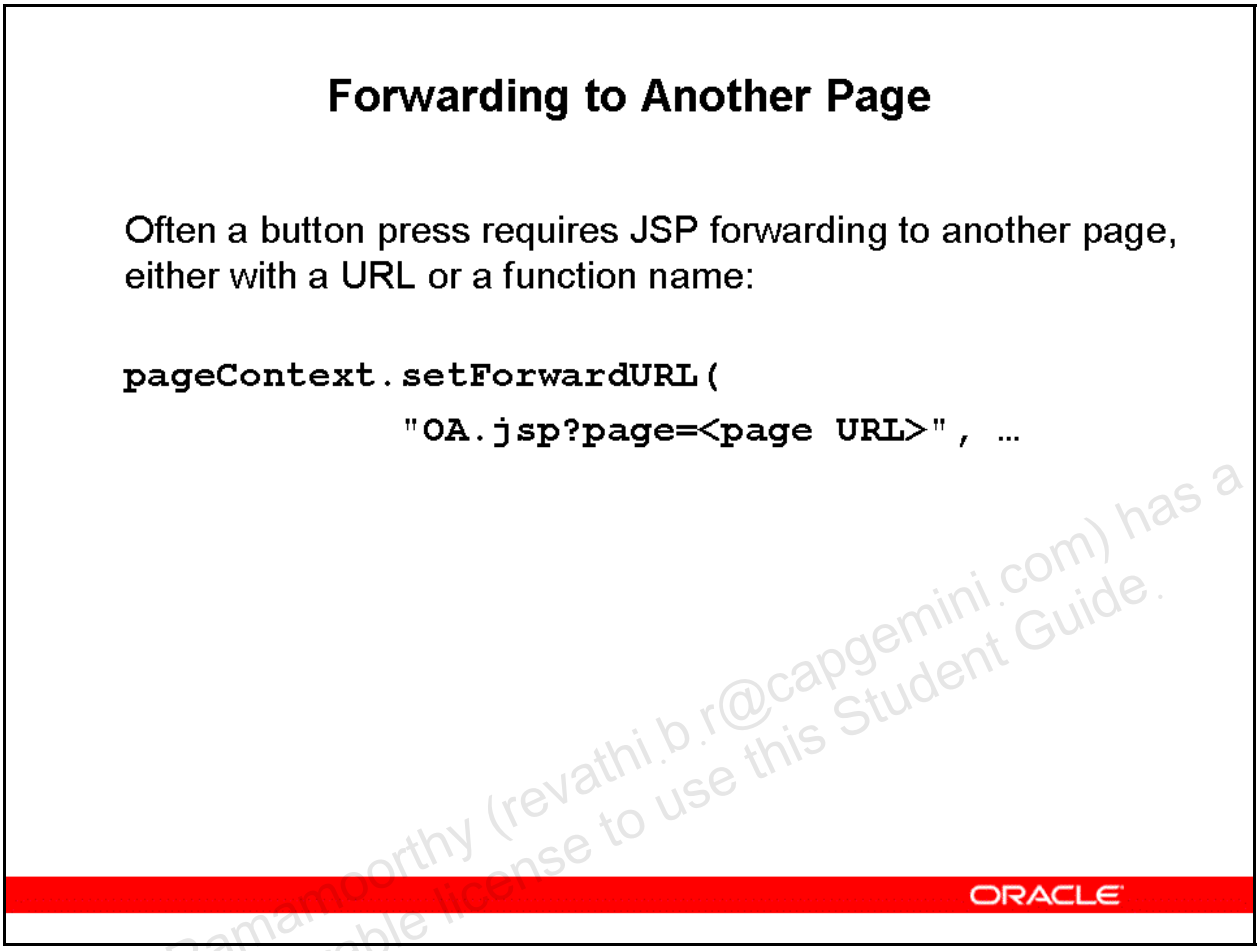
Example Search: Controller



There are two methods of redrawing the page.

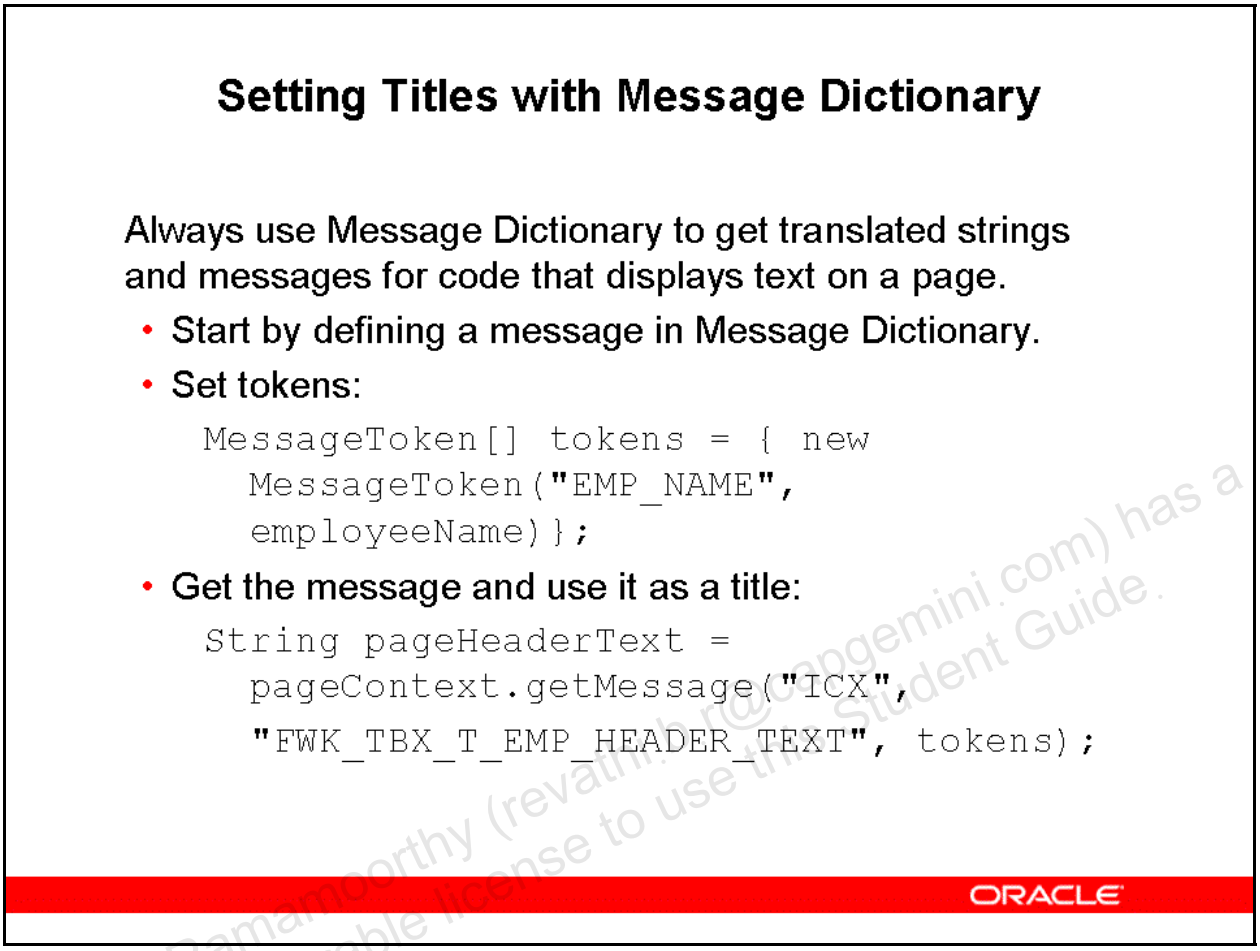
1. The primary way is now to use partial page rendering (PPR) to redraw relevant sections of the page without redrawing the entire page.
2. The older method is to forward the current page back to itself if there is some UI change that requires going back through the processRequest logic for the page.
3. The use of forwarding back to the current page to display UI changes, such as a data-dependent title, is no longer necessary for most pages, which now use partial page rendering (PPR) instead.

Forwarding to Another Page



A forward is used to direct to a new page without having to interact with the browser. The redirection is done on the middle tier server.

Setting Titles with Message Dictionary



Declarative values set in the pages (text, prompts, and so on, set in the Property Inspector) are also translatable, but do not use Message Dictionary.

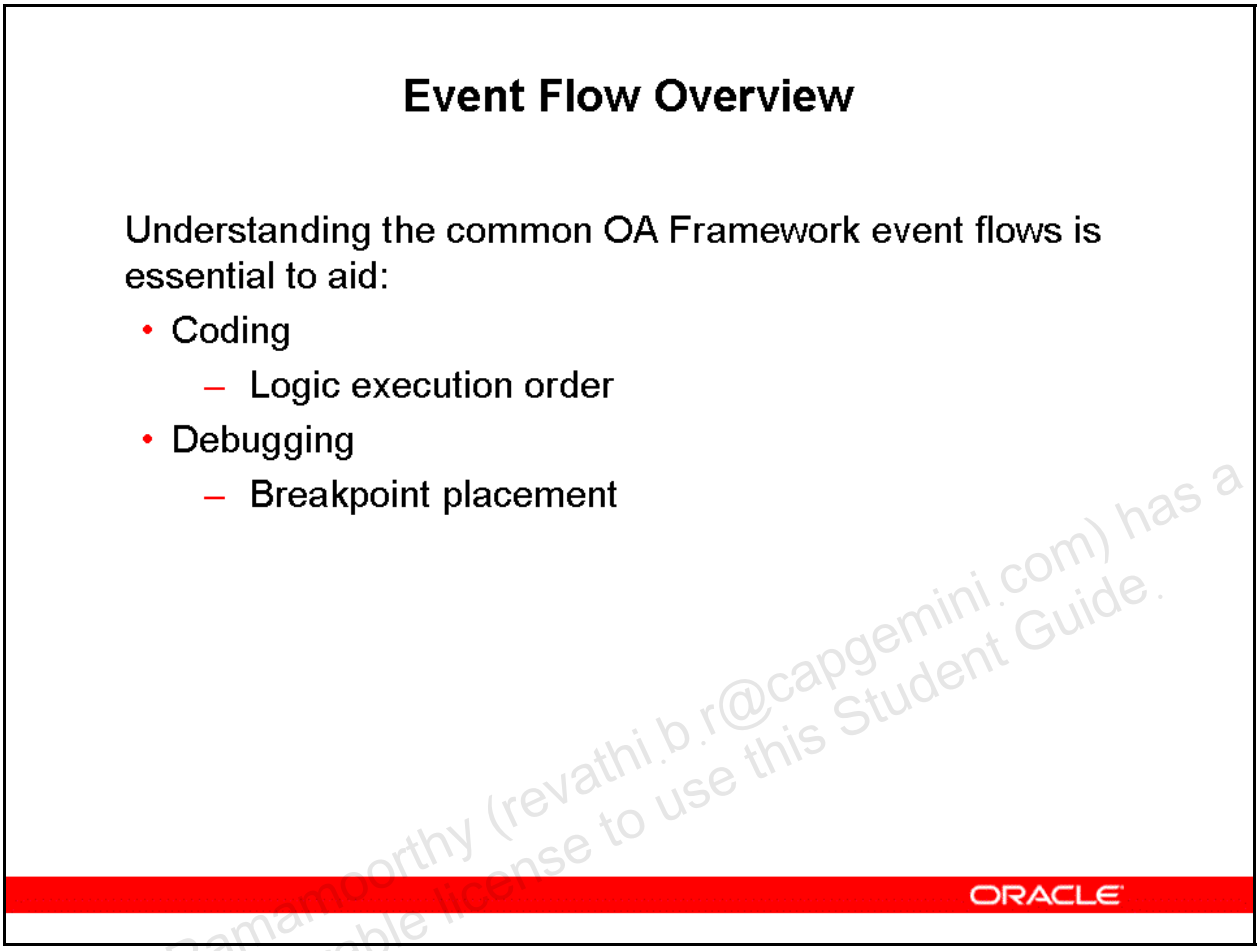
You should never hardcode text into your page logic. Use the Message Dictionary to ensure that the text can be translated and reused.

Do not use tokens for single words, partial words, or sentence fragments because they do not translate into languages that have a different grammatical structure. For example:

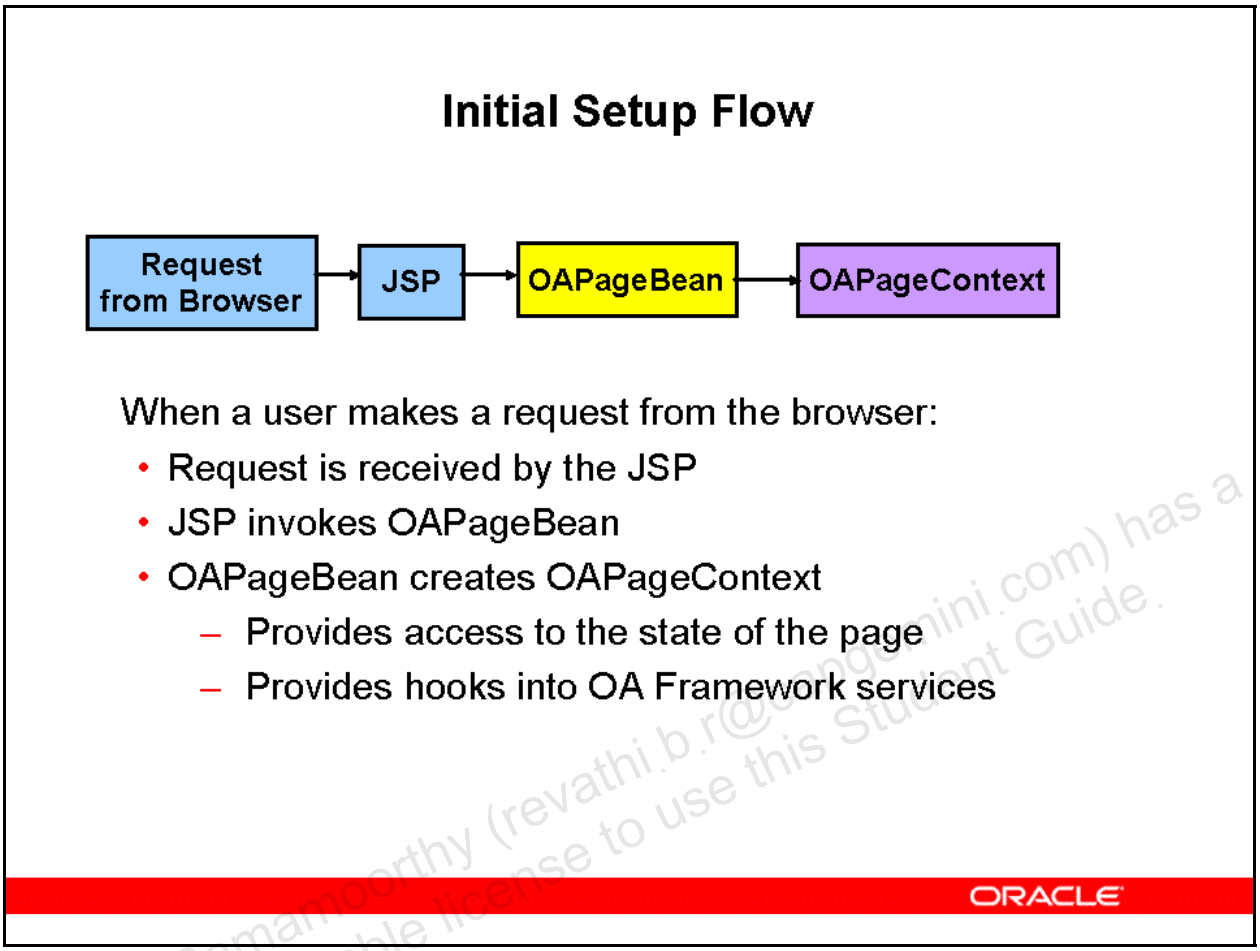
"This &DOCUMENT cannot be &ACTION."

DOCUMENT might be purchase order or invoice, and ACTION might be approved, denied, and so on. These would not translate to a Romance language that uses gender, because there may be a mismatch between the noun and the verb.

"&NUMBER of row&PLURAL saved." where the value of PLURAL can be either "s" or nothing, to result in "row" or "rows". This does not translate to any language that does not use "s" to designate a plural.

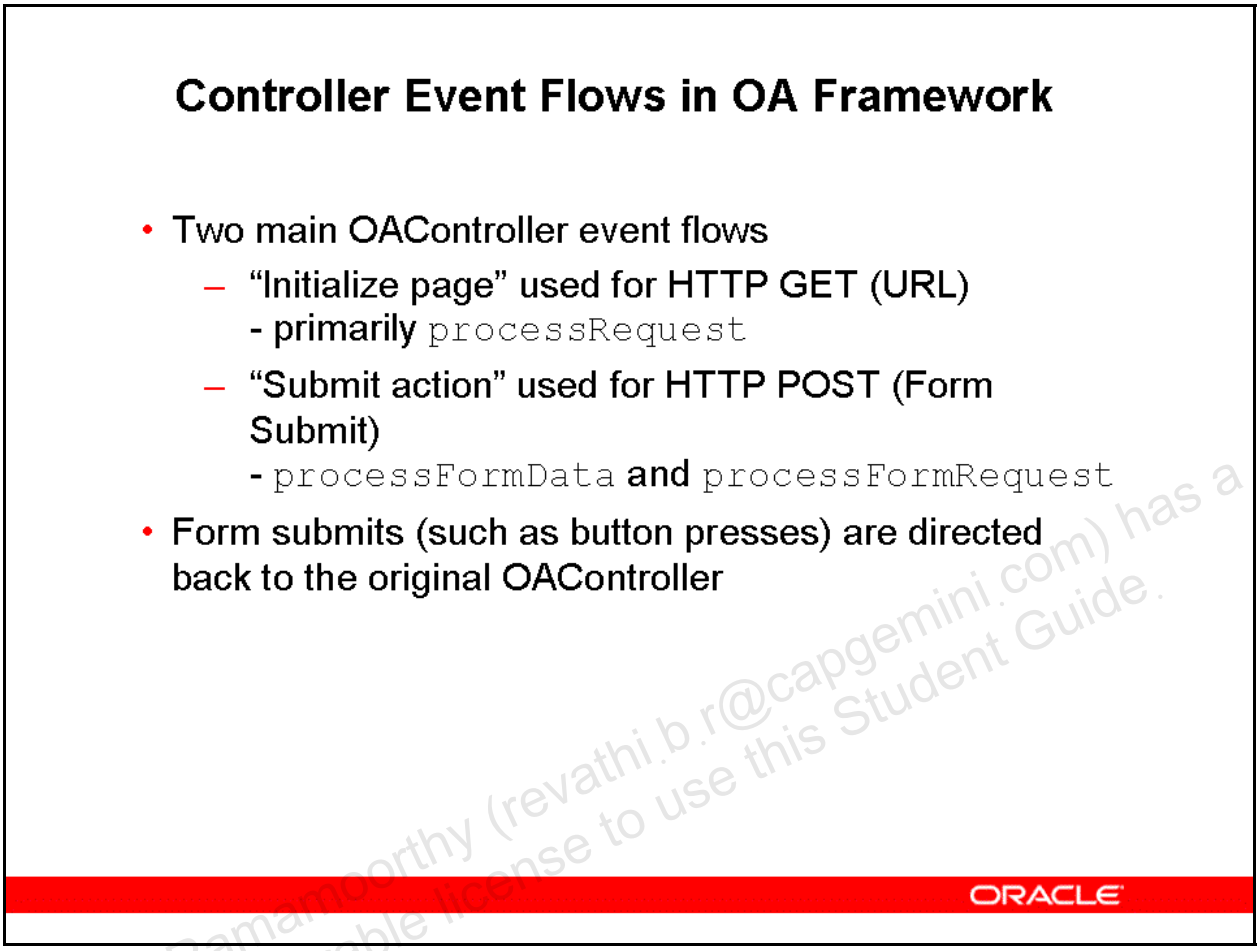


Initial Setup Flow

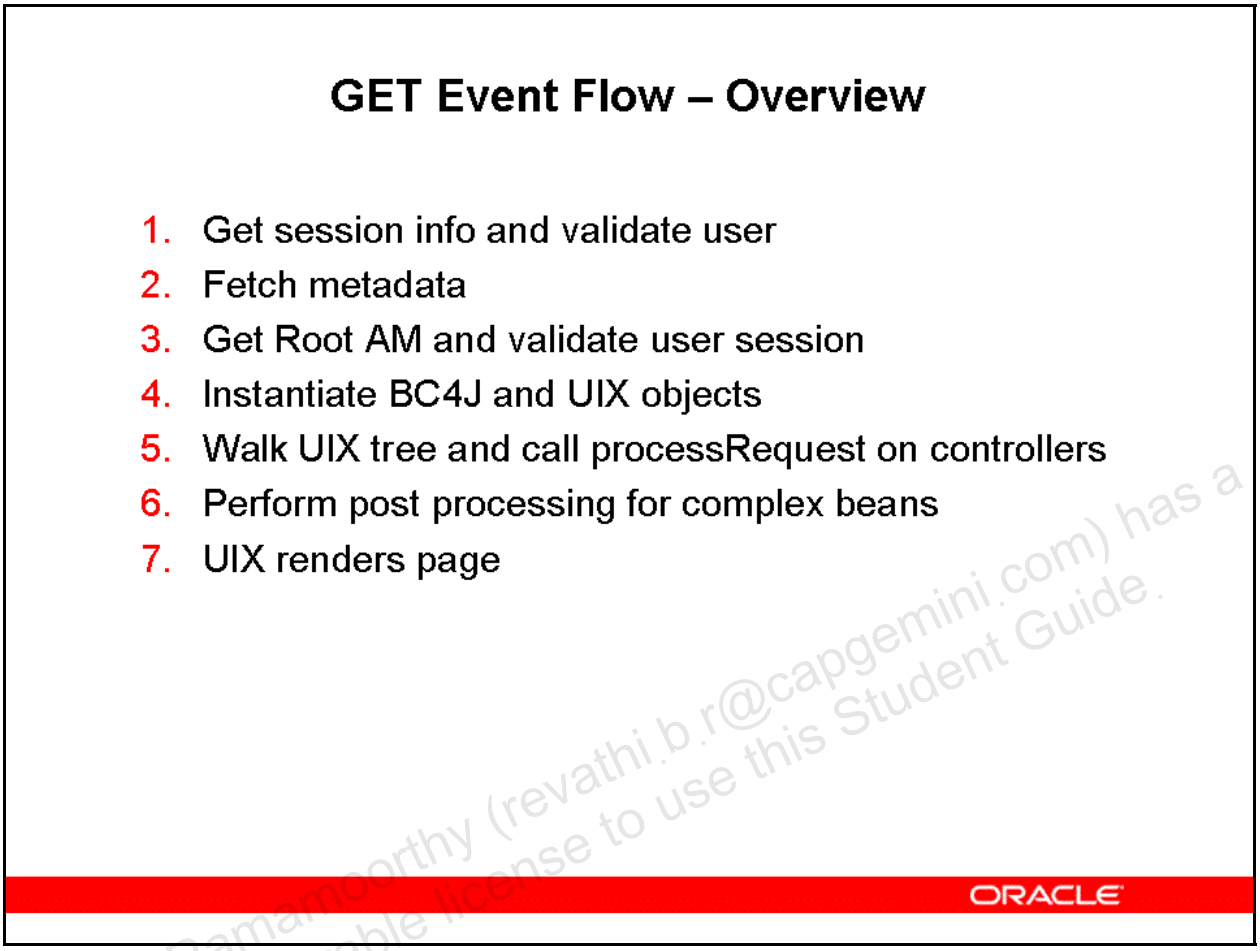


OAPageBean does many “housekeeping” tasks.

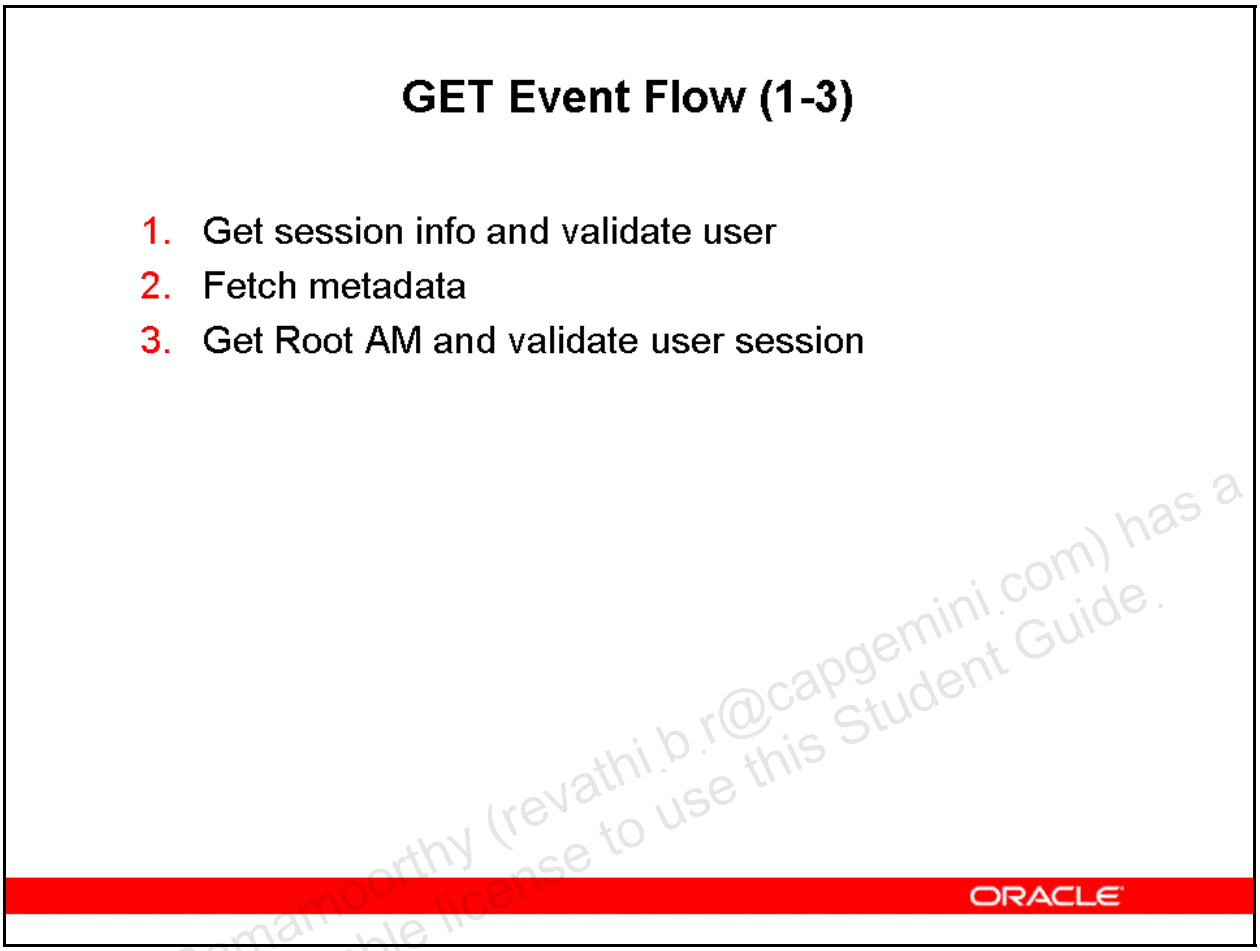
OAPageContext is passed to all the web beans when they are called. It can be considered the beans communication link to the outside world. "OA Framework services" mostly refers to services from AOL/J, but also to convenience methods for common tasks.



"Submit action" is an oversimplification. The flow for this action also includes drawing or redrawing a page.



GET Event Flow (1-3)



1. Get the ICX Session Cookie

The URL provides:
Database (DBC) information
Page name
Other parameters

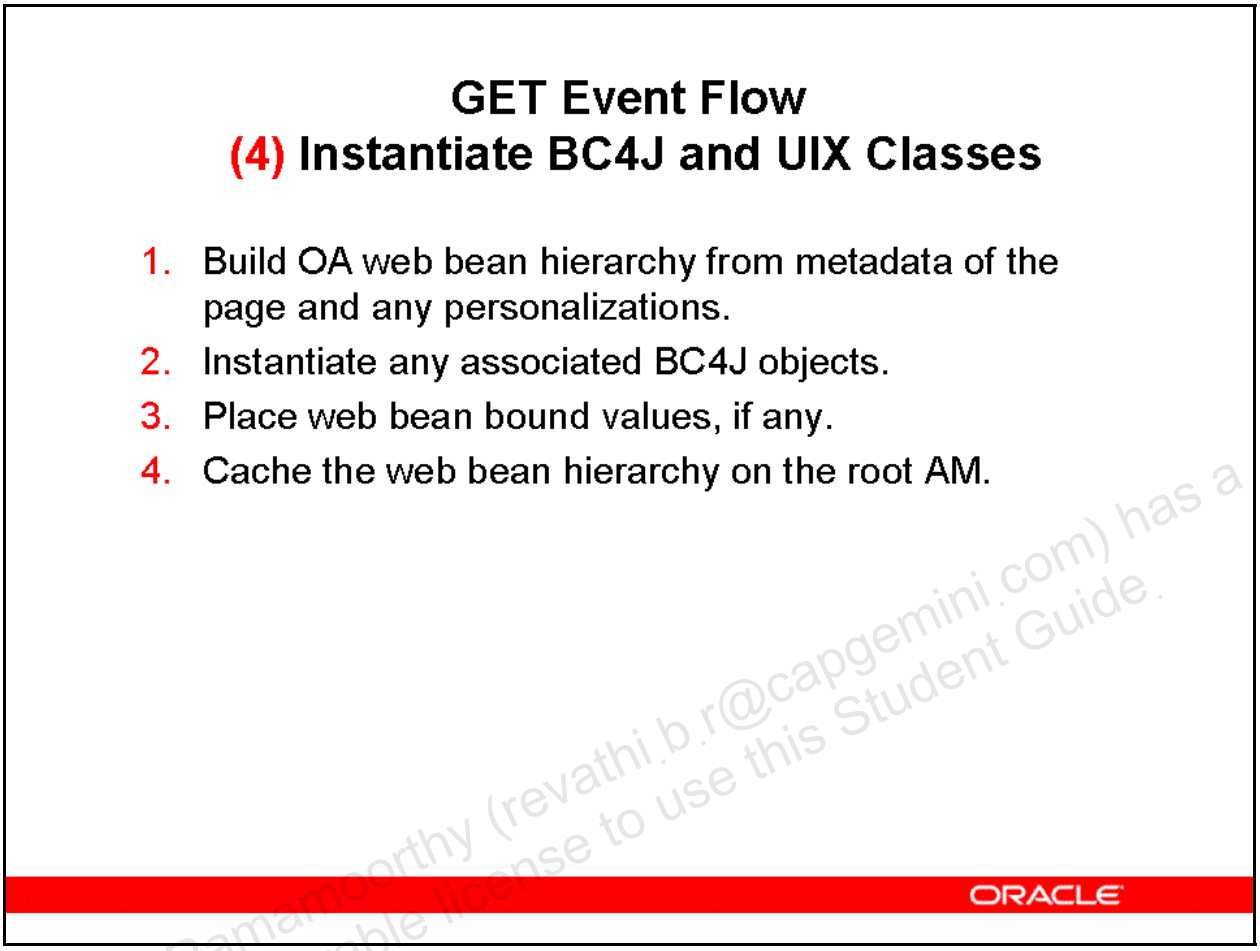
2. Check to see if metadata is in the cache.
3. If metadata not in cache, fetch the metadata. Metadata fetched through separate static connection. At customer sites metadata will be in the database. In JDeveloper you can work against XML files, the repository, or both.

Note: Separate static connection for fetching metadata means that: 1) It's a different commit cycle so rollbacks in their own pages will not affect the metadata (though that would be read-only anyhow) and 2) there is a separate AM in the AM pool for fetching the metadata.

4. Apply personalizations.
5. Get root AM for session.
6. Validate session on root AM
7. Validate the function (function security) associated with the page, the user, and the responsibility.

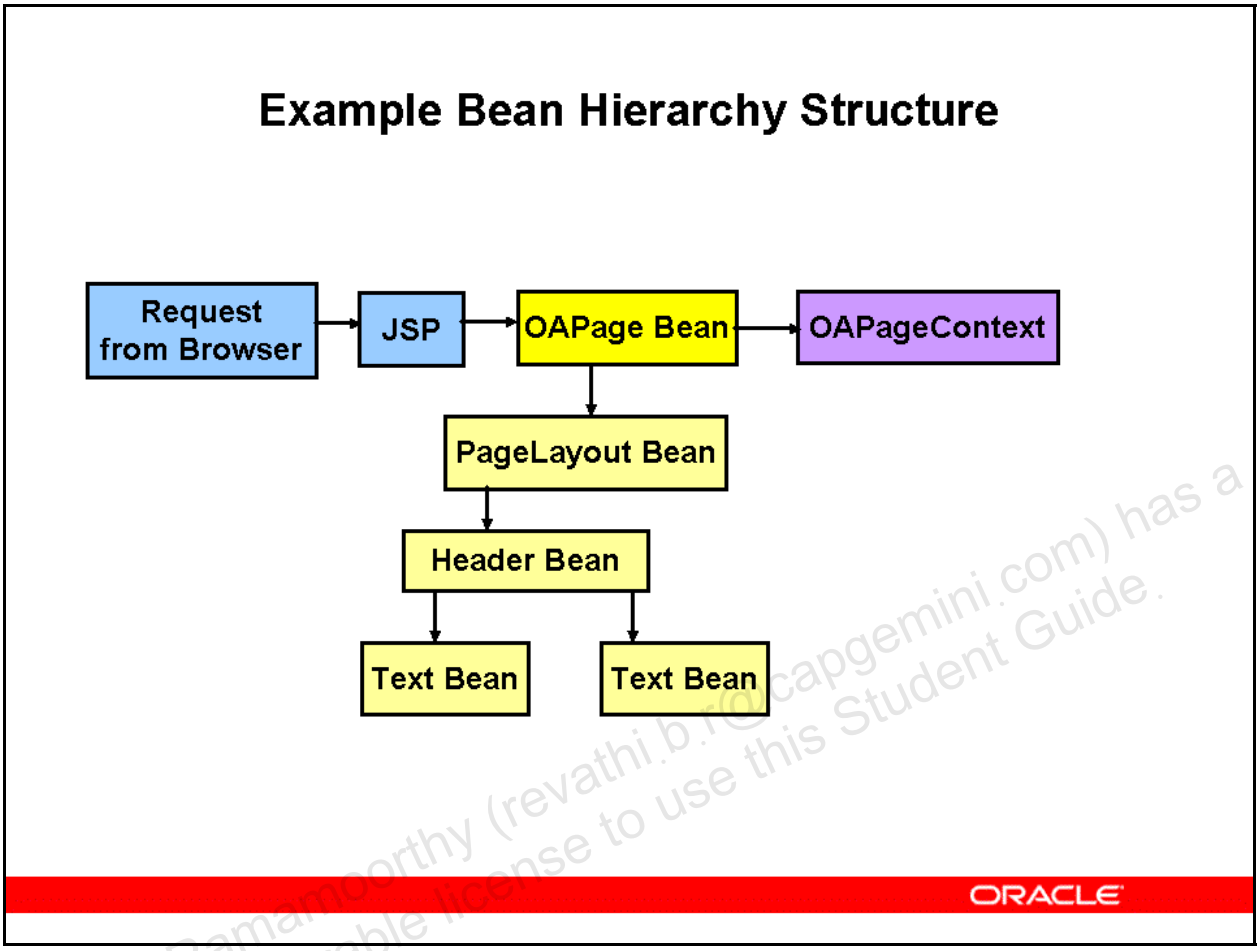
The AM associated with page is the root AM. The root AM holds the database connections. One connection is used for data. The other connection, the shadow, is used to fetch metadata and validate SQL.

Note: “Validate function associated with page” refers to going through function security to see if user has access to the function. Function is (at least) associated with the page using a property at the pageLayout (top) region of a page.

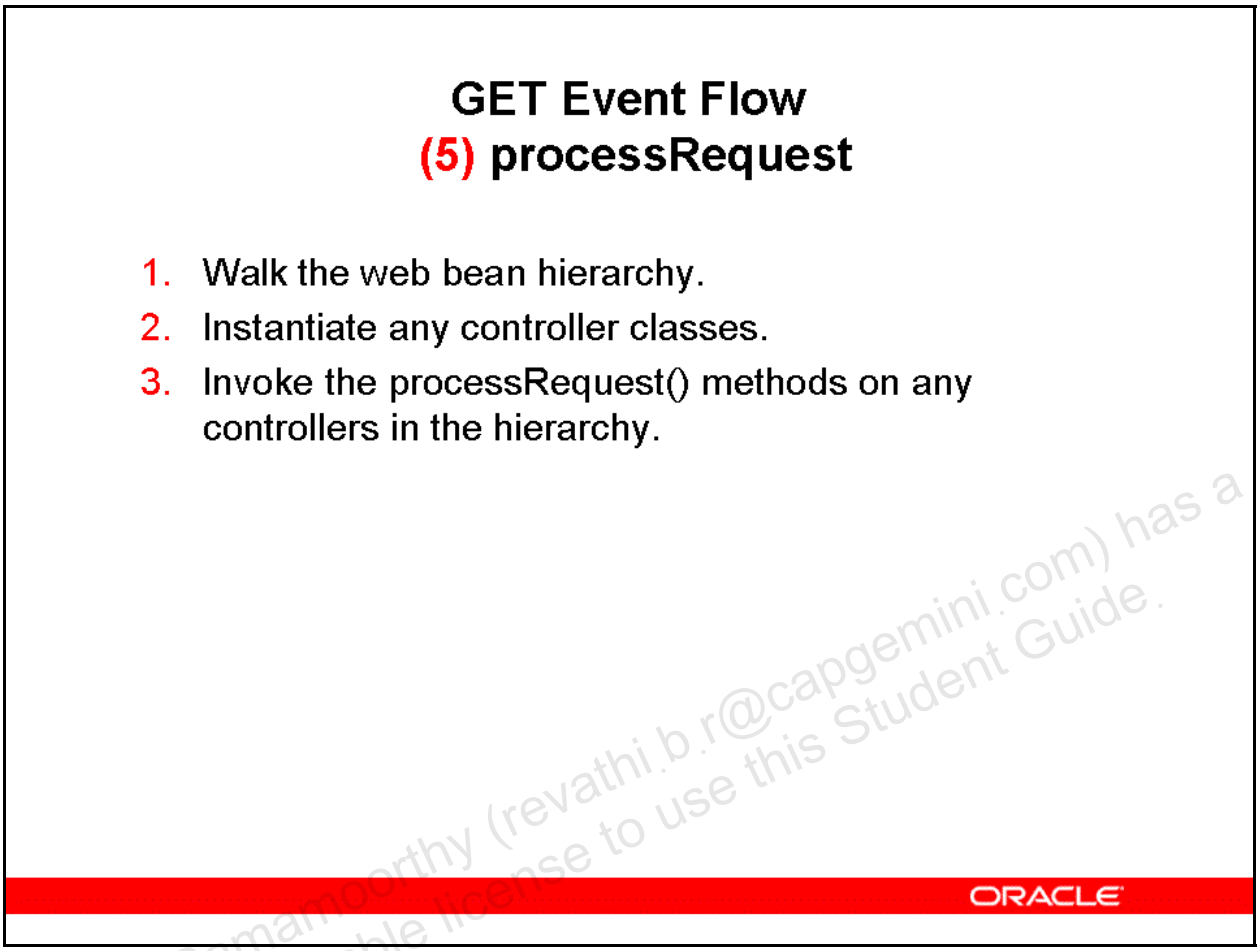


At runtime, UI regions and items map to web beans (JavaBeans). OABeans are extensions of UIX beans.

Example Bean Hierarchy Structure



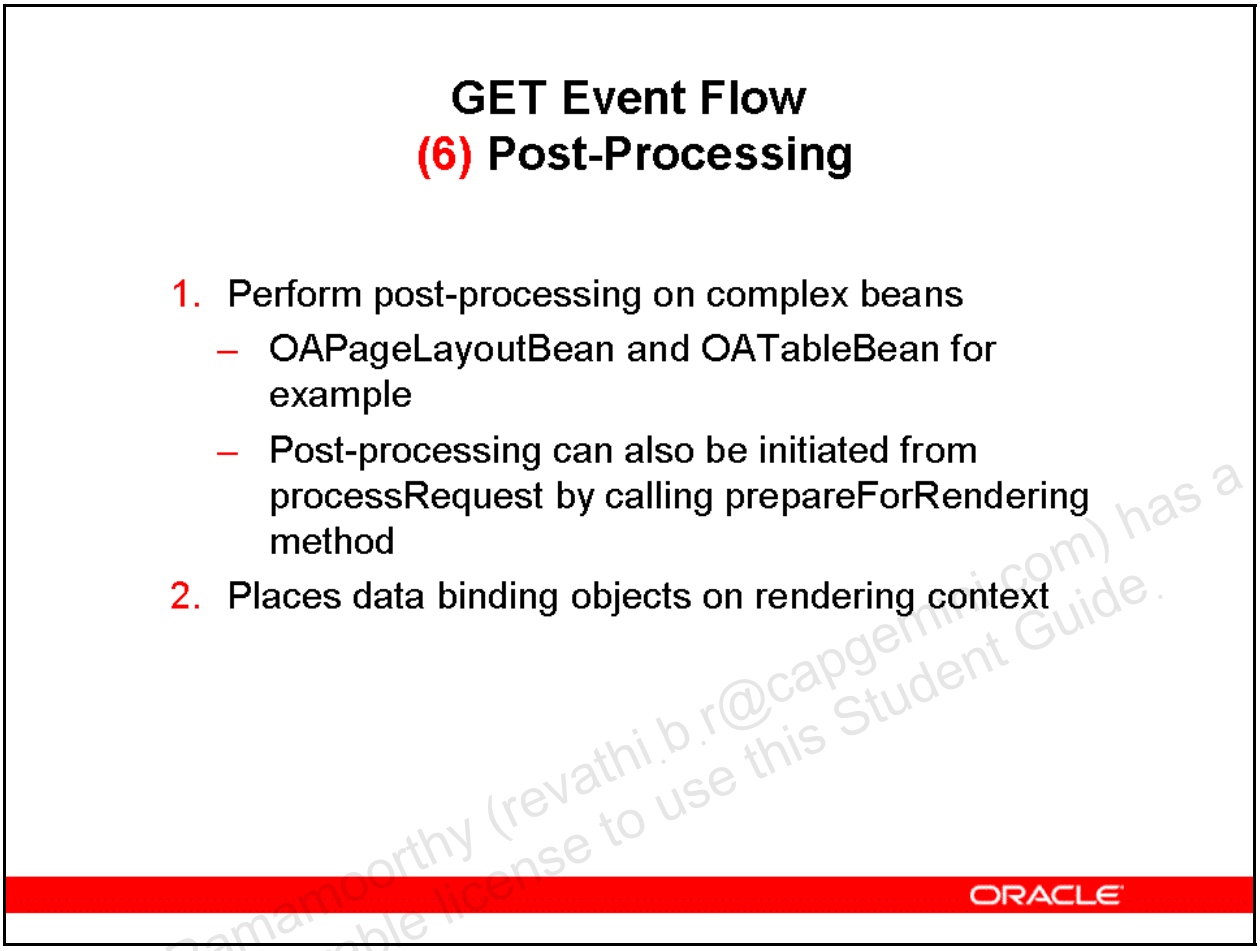
GET Event Flow (5) processRequest



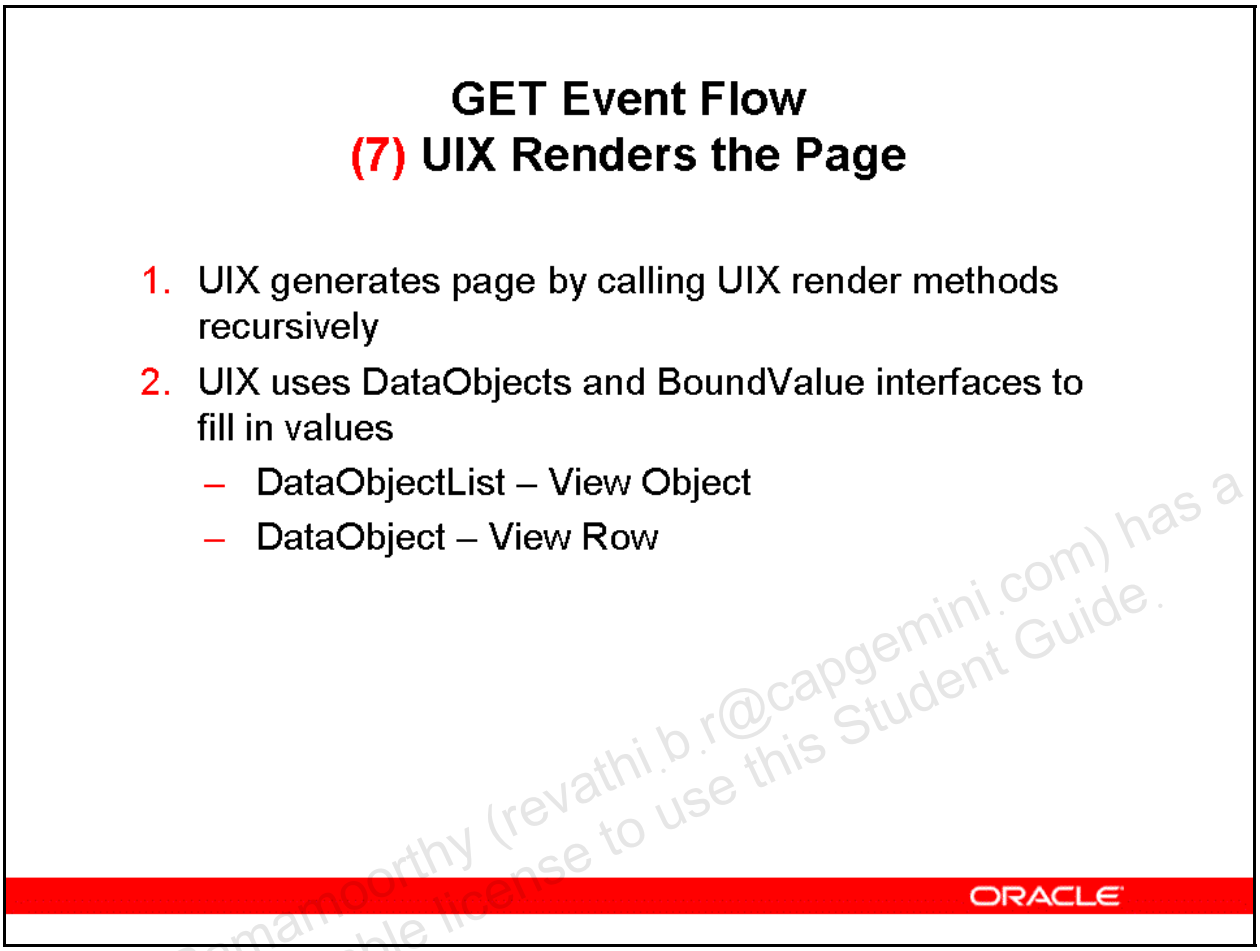
In the processRequest() method, developers can query data (using vo.executeQuery) and set bean properties programmatically.

Note: Programmatically setting, and even creating, a UI object is possible, but is highly discouraged. Programmatic UI creation or manipulation has the following drawbacks:

1. Adverse performance impact
2. Inability to apply personalizations
3. Difficult to maintain and debug
4. Programmatically created pages cannot use important OA Framework abilities, like partial page refresh



GET Event Flow (7) UIX Renders the Page

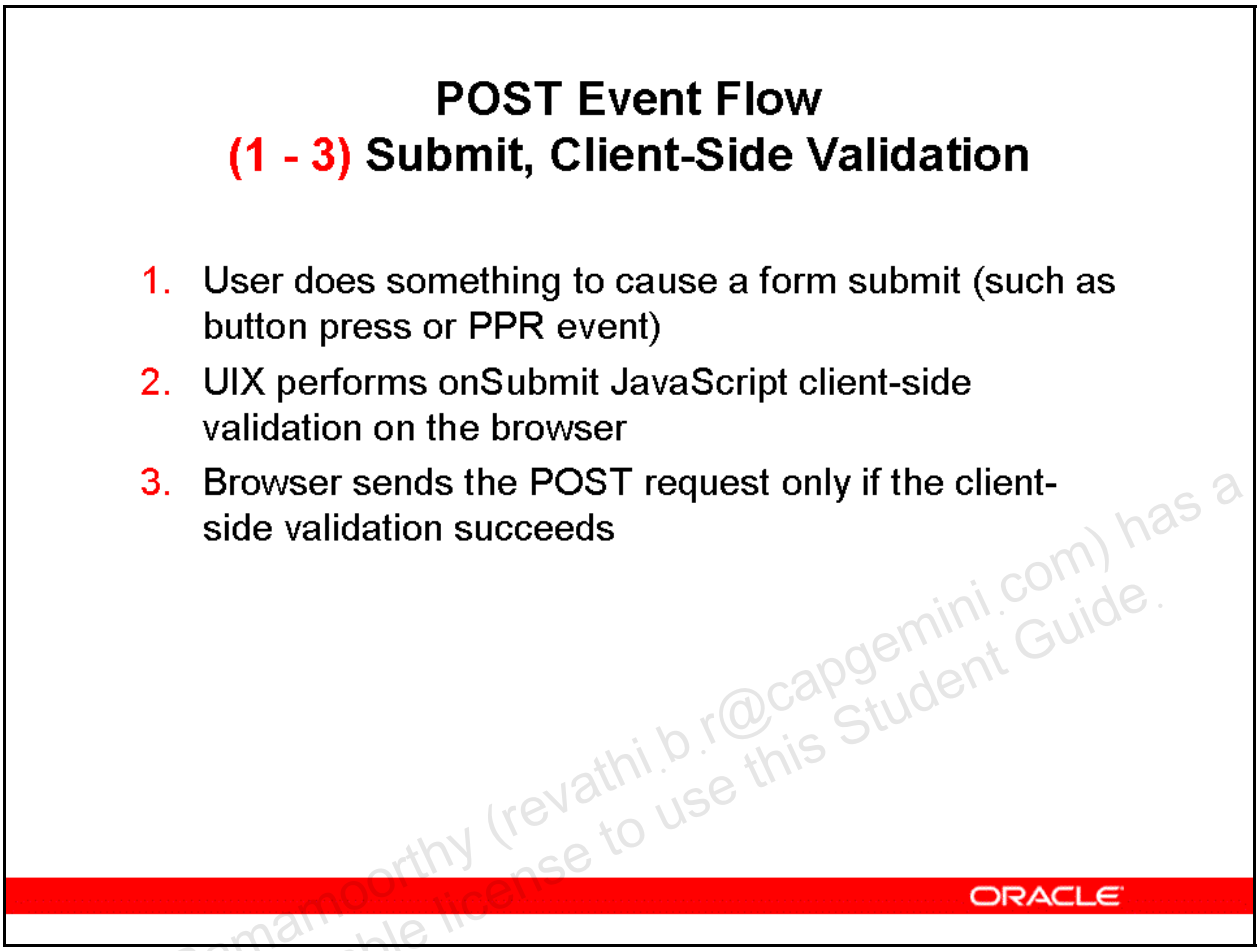


POST Event Flow – Overview

1. User does something to cause a form submit
2. UIX performs client-side validation on the browser
3. Browser sends the POST request
4. Validate user
5. Retrieve AM and bean hierarchy if saved
6. Walk UIX tree and apply form data to data objects in `processFormData`
7. Walk UIX tree and call `processFormRequest` on controllers
8. If no redirect then refresh the page

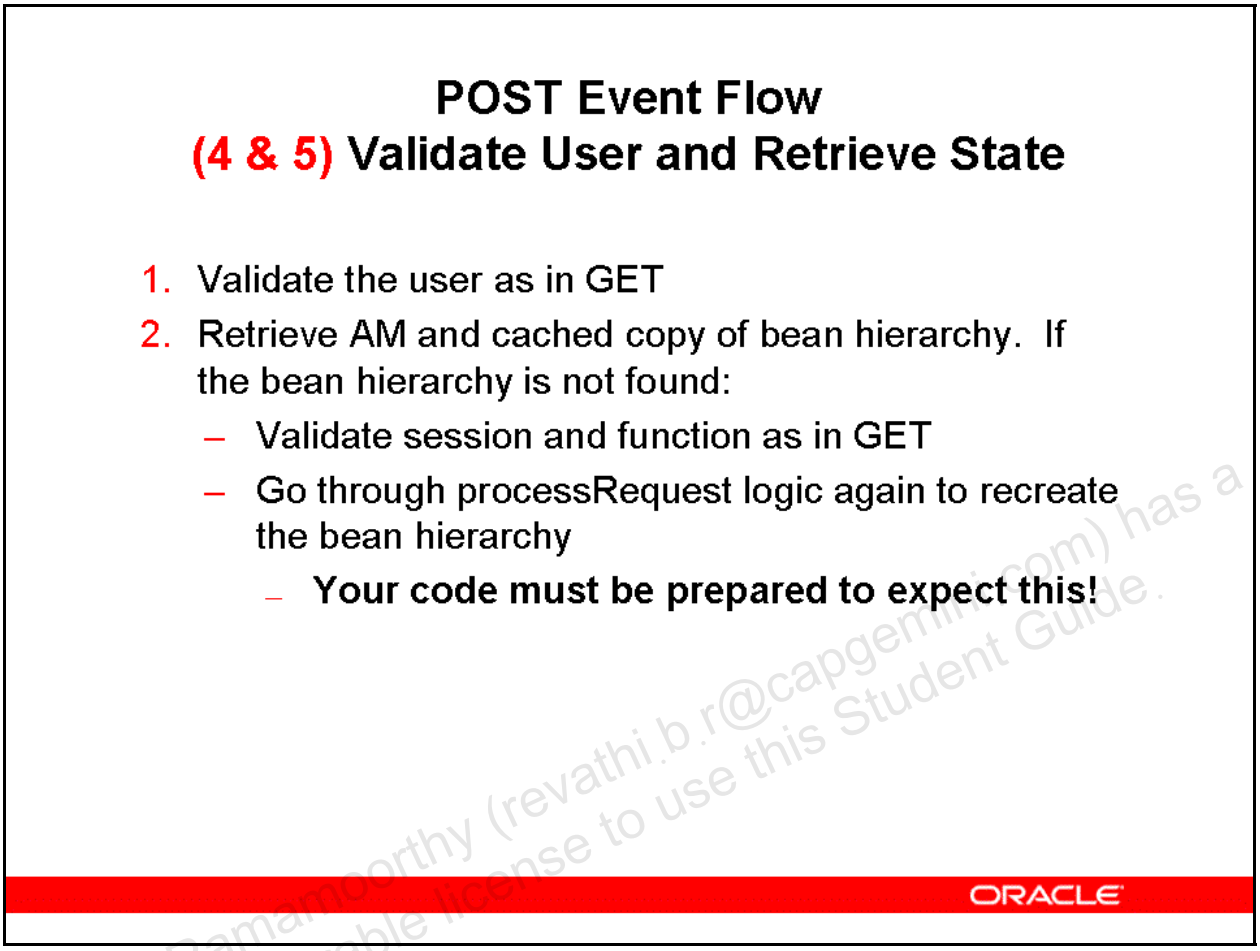
ORACLE

POST Event Flow (1 - 3) Submit, Client-Side Validation



UIX performs browser-side validation using JavaScript as UIX builds JavaScript validation. However, OA Framework based pages and their related code must not use JavaScript in the controller or in any other mechanism. UIX JavaScript is not the same as custom JavaScript.

POST Event Flow (4 & 5) Validate User and Retrieve State



Cases where the bean hierarchy might not exist or might be out of synch include: Back button use, passivation, and failover to another JVM. Failover is not yet supported.

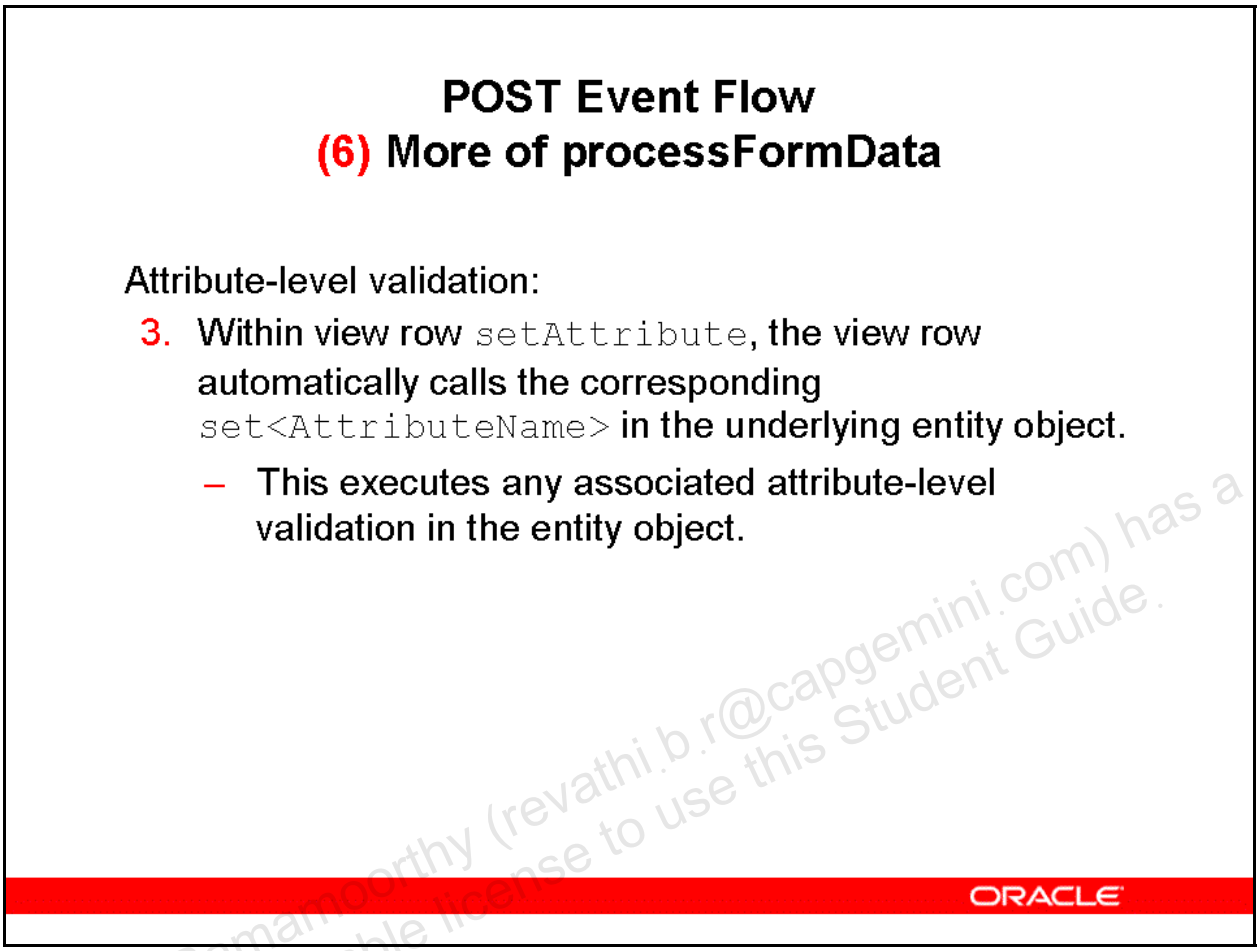
POST Event Flow (6) Apply Form Data

POST Event Flow
(6) Apply Form Data

First pass through the bean hierarchy:

1. The `processFormData` method applies form data to the underlying objects. If a primary key is defined for the object, it validates that the data is being applied to the correct object.
   - Throws state error if data is out of synch
2. Within `processFormData`, OA Framework calls `setAttribute` on the current row of the underlying VO for each bean.
   - Executes any attribute-level validation you've written for the view object row (ViewRowImpl)

ORACLE



POST Event Flow (6) More of processFormData

POST Event Flow
(6) More of processFormData

Row-level validation:

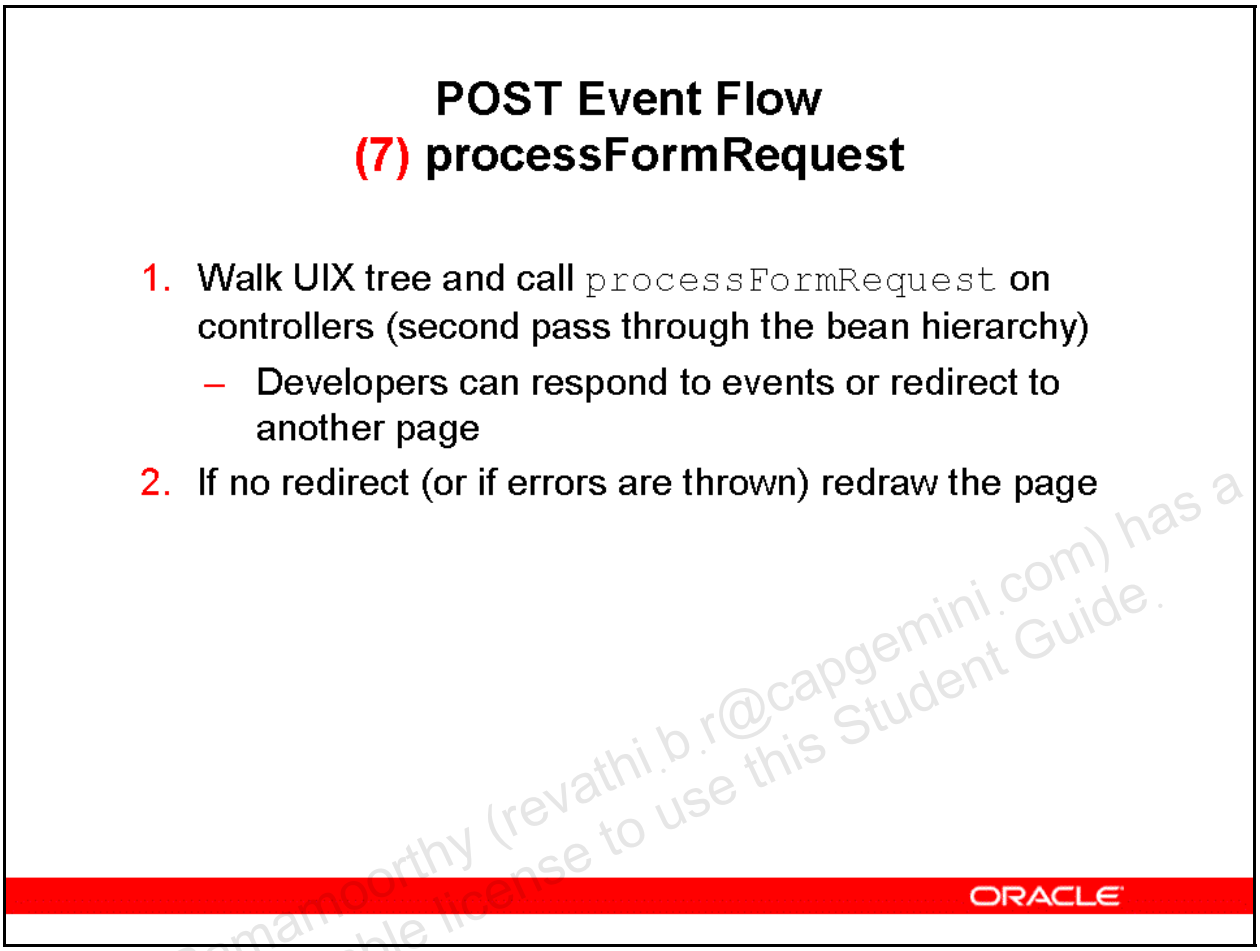
4. Once all the attribute values have been set, the OA Framework calls the VO `validate` for each row it modified to execute any associated row-level validation.
   - Within `validate`, the view row calls `validateEntity` for the underlying EO, which executes any entity-level validation.
     - Debugging Tip: Any declarative BC4J validation (such as Update While New specified in BC4J wizards) fires after validation in your Impl.java files.

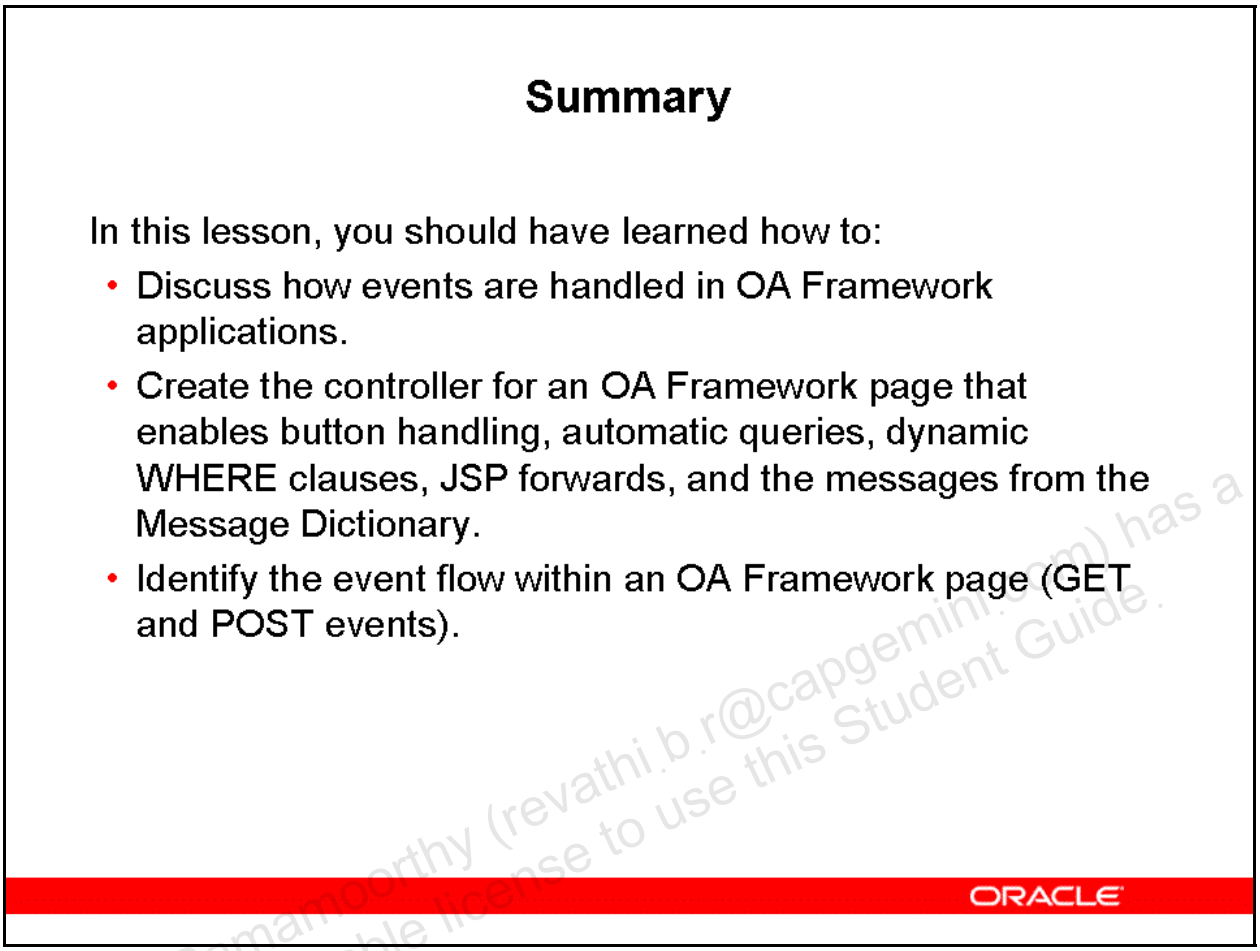
ORACLE

POST Event Flow (6) More of processFormData

5. OA Framework automatically displays error messages for any exceptions thrown by the model layer during `processFormData`.
   - Bad attribute values are maintained in the `OAAttrValException` object
   - If there are errors, code does not proceed to the next phase of calling `processFormRequest`
     - Debugging Tip: If your code never gets to a debugger breakpoint in `processFormRequest`, it probably had an error in `processFormData`.

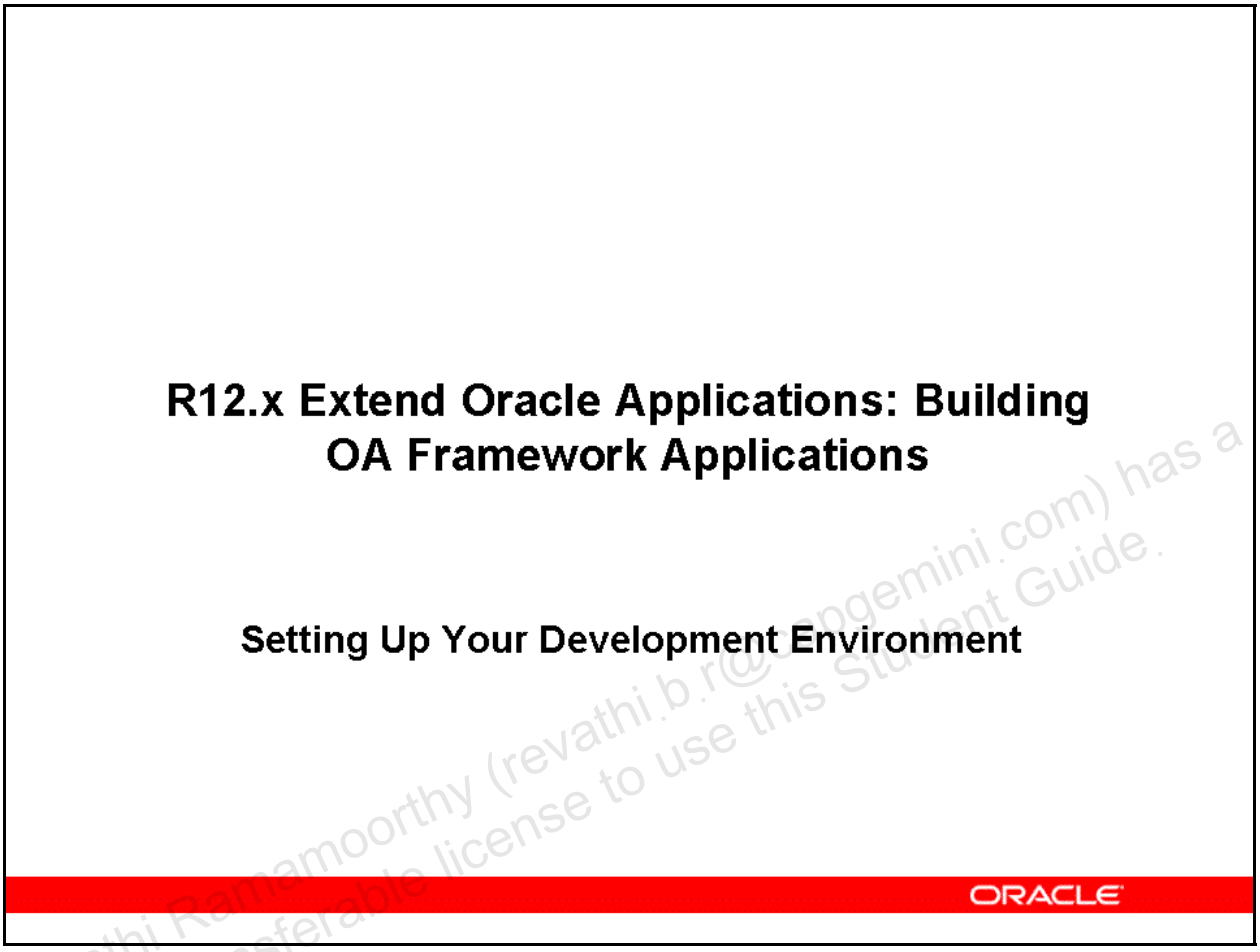
ORACLE



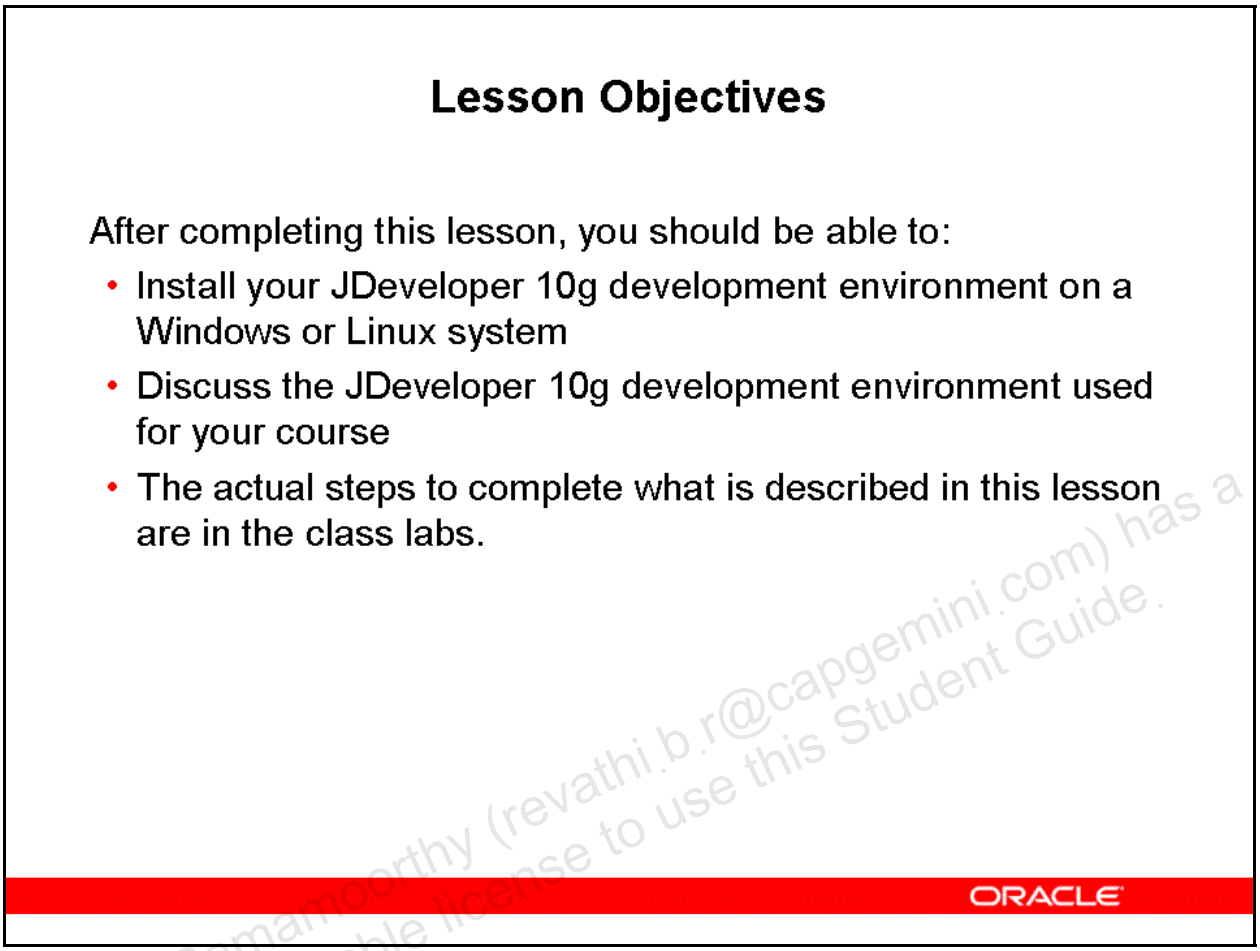


Setting Up Your Development Environment

Chapter 6



Lesson Objectives

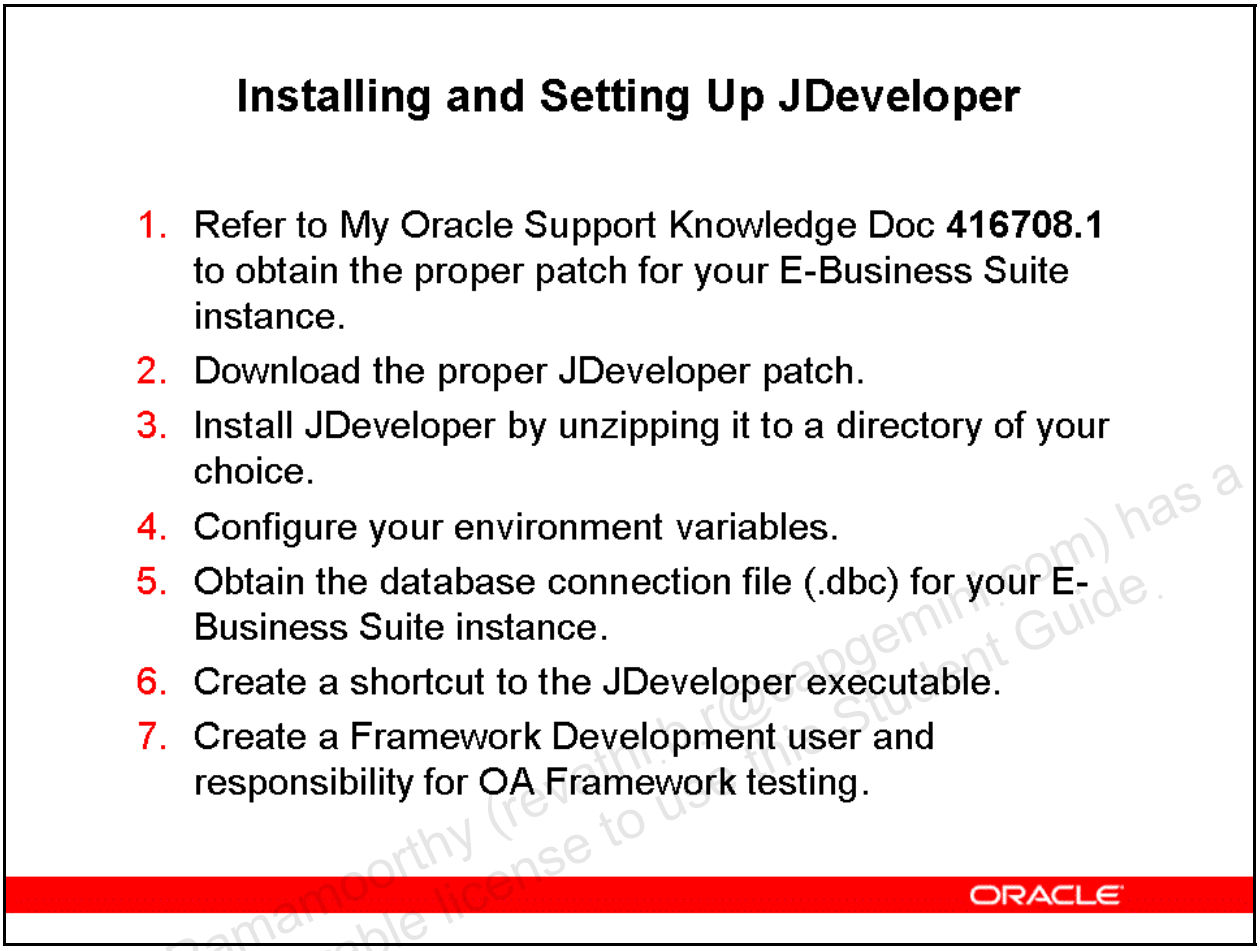


While not specifically part of this course, it is critically important that an OA Framework developer obtains the specific version of JDeveloper 10g with OA Extension for their version of E-Business Suite. JDeveloper 10g with OA Extension is used with R12 E-Business Suite instances. For 11.5 (11i) E-Business Suite instances, you will need to use JDeveloper 9i. The techniques for OA Framework development in both versions of JDeveloper are similar, but there are some important differences between the two versions of JDeveloper.

It would be possible to include a table of the patch numbers that contain the specific versions of JDeveloper 10g with OA Extension. The problem with that is that this course material would quickly become out of date. There is a very specific My Oracle Support knowledge article (Formerly Oracle MetaLink Note) that contains all the latest information on the released patches.

How to find the correct version of JDeveloper to use with E-Business Suite 11i or Release 12 – My Oracle Support Knowledge Document: 416708.1

Installing and Setting Up JDeveloper



When you download the proper patch for your E-Business Suite instance, the patch will be named similar to the following:

```
pXXXXXX_RXX_GENERIC.zip
```

JDeveloper 10g with OA Extension does not contain a installation routine. The installation process consists of unzipping the patch to a directory of your choice. For example, you could use the following:

On Linux – `/Jdev`

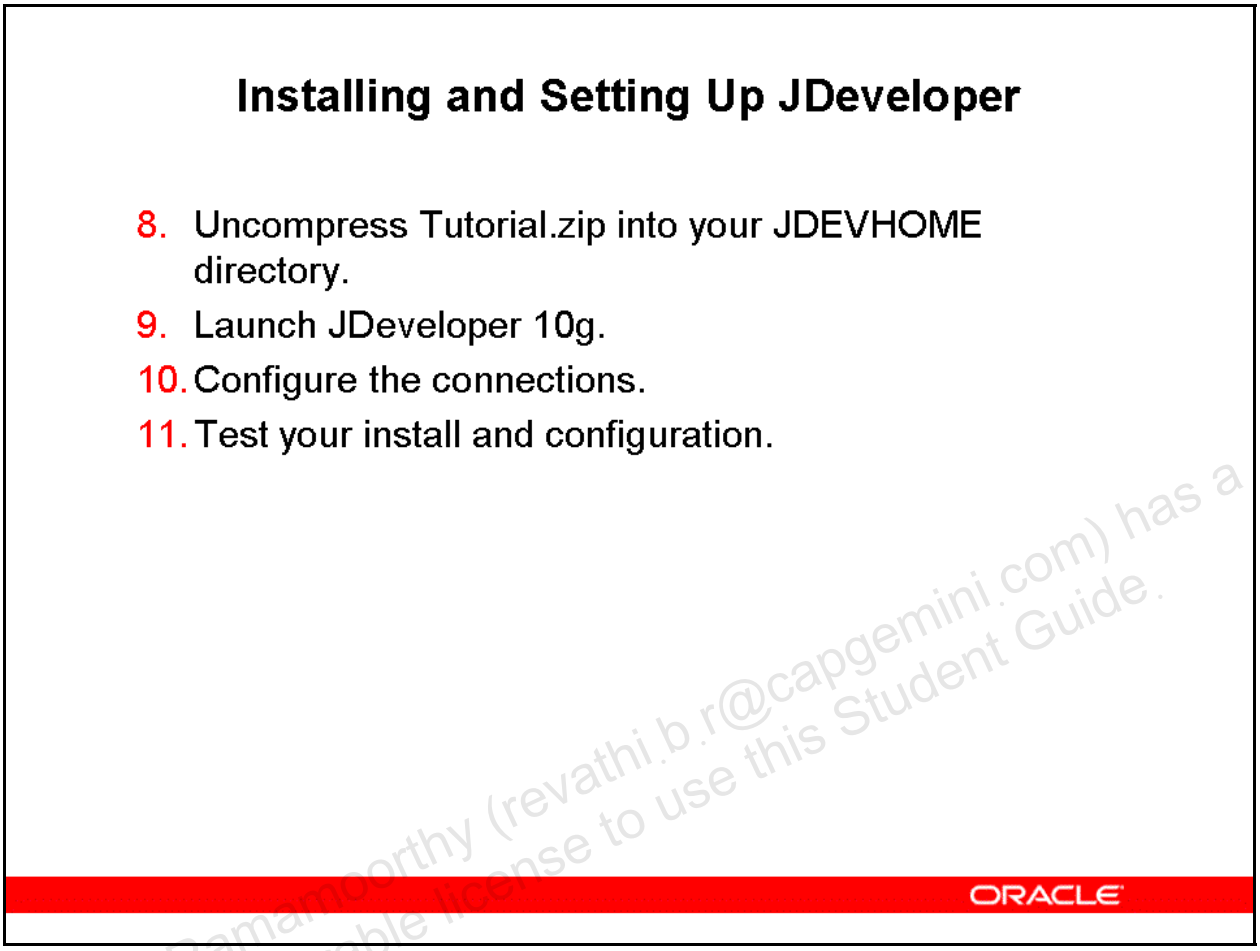
On Windows – `D:/JDev`

When you unzip the patch, it will create three (3) subdirectories as follows:

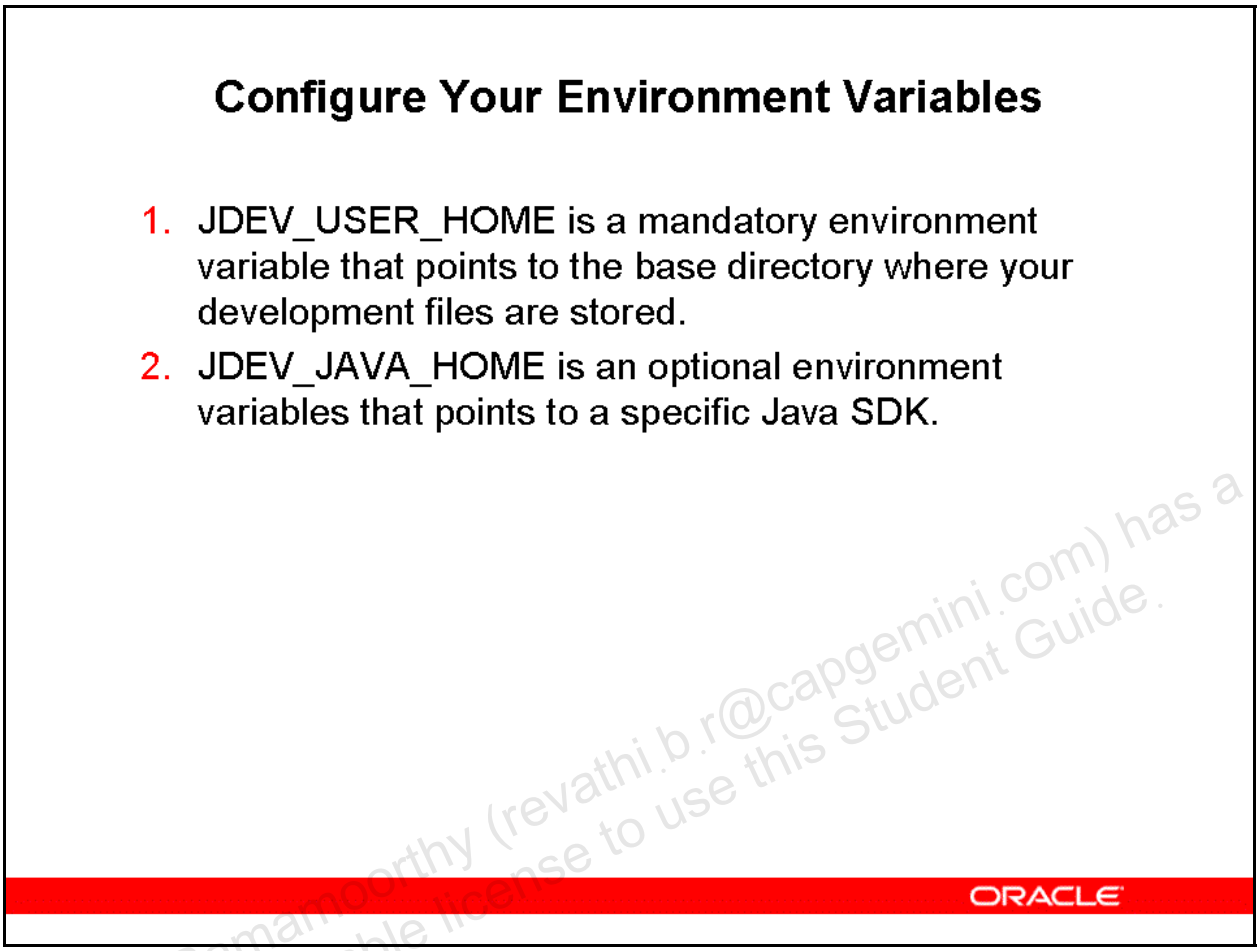
`jdevbin` – This subdirectory contains all the executables, utilities, and supporting files for JDeveloper 10g with OA Extension.

`jdevdoc` – This subdirectory contains the OA Framework Developer's Guide (OAFDG) along with the Javadoc files for the APIs.

`jdevhome` – This subdirectory is empty. But, it can be used as the base directory for your OA Framework files.



Configure Your Environment Variables



Setting environment variables is specific to both the operating system, operating system version, and command shell being used. It is not possible to list every single variation. Here are some specific examples.

Setting the JDEV_USER_HOME environment variable on Windows 2000/XP

1. Right-click My Computer on your desktop, select Properties.
2. In the System Properties dialog, select the Advanced tab.
2. On the Advanced tab, select the Environment Variables button.
4. Select the New button at the User variables for <username> box.
5. In the New User Variable dialog, enter JDEV_USER_HOME in the Variable Name field.
6. Set the Variable Value field to the location of your JDEV_USER_HOME subdirectory (for example, D:\JDev\jdevhome\jdev)
7. Select OK in each of the dialogs you opened to save the new user environment variable.

Setting the JDEV_USER_HOME environment variable on Linux

There are a multitude of shells that can be run for Linux operating systems. Examples of these shells include, bash, csh, tcsh, and others. In most cases, one of these two methods will work.

Open a terminal window on your Linux machine, and type the commands listed in the method that works for you.

Method 1:

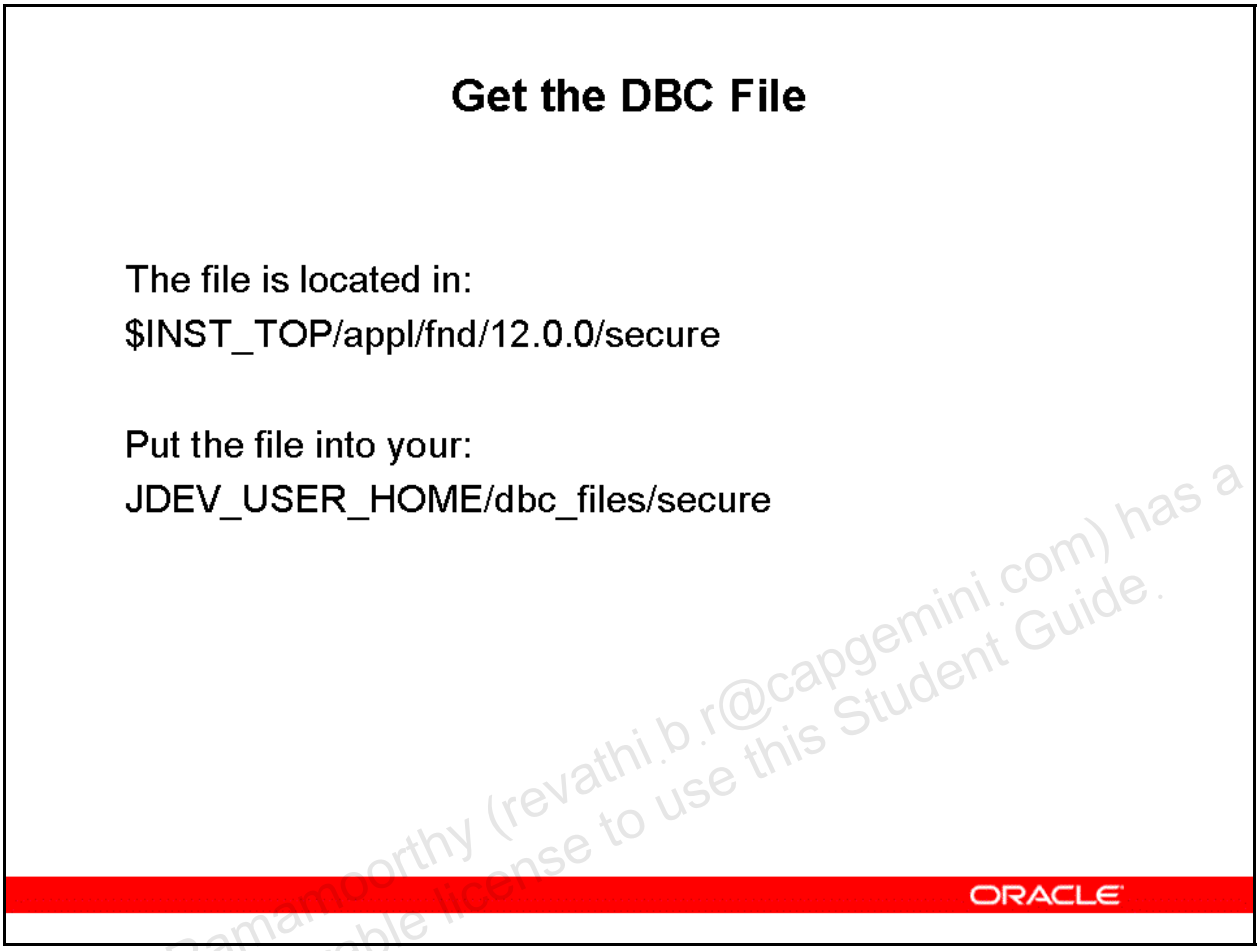
```
set JDEV_USER_HOME = /JDev/jdevhome/jdev
export JDEV_USER_HOME
```

Method 2:

```
export JDEV_USER_HOME = /JDev/jdevhome/jdev
```

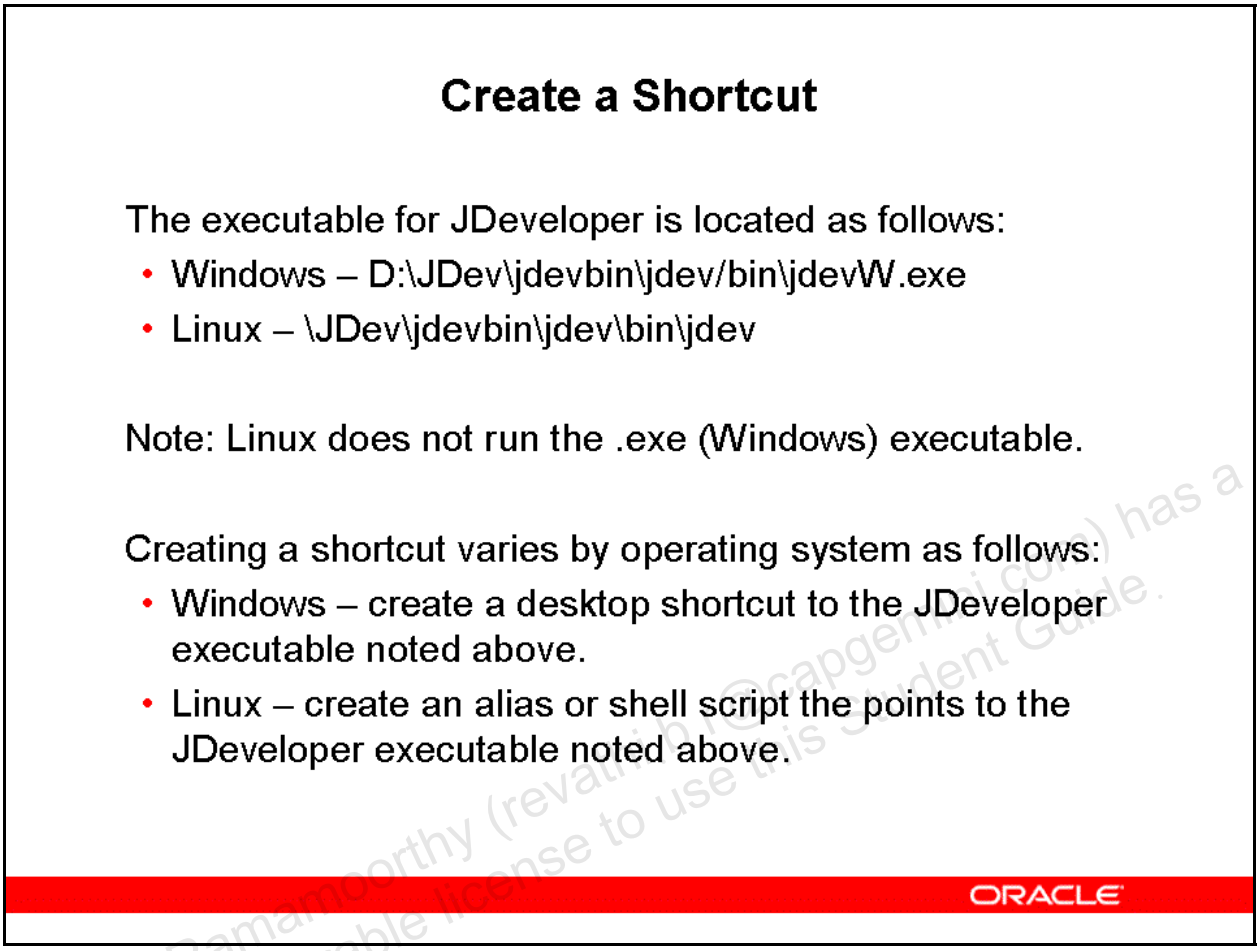
JDEV_JAVA_HOME points to the appropriate Java SDK for your development, and certified by Oracle for your Operating System and E-Business Suite version. In most cases, you will not need to set this on either Linux or Windows servers. The appropriate values are already set-up for you. In rare cases almost always on Linux servers, this needs to be set.

Get the DBC File



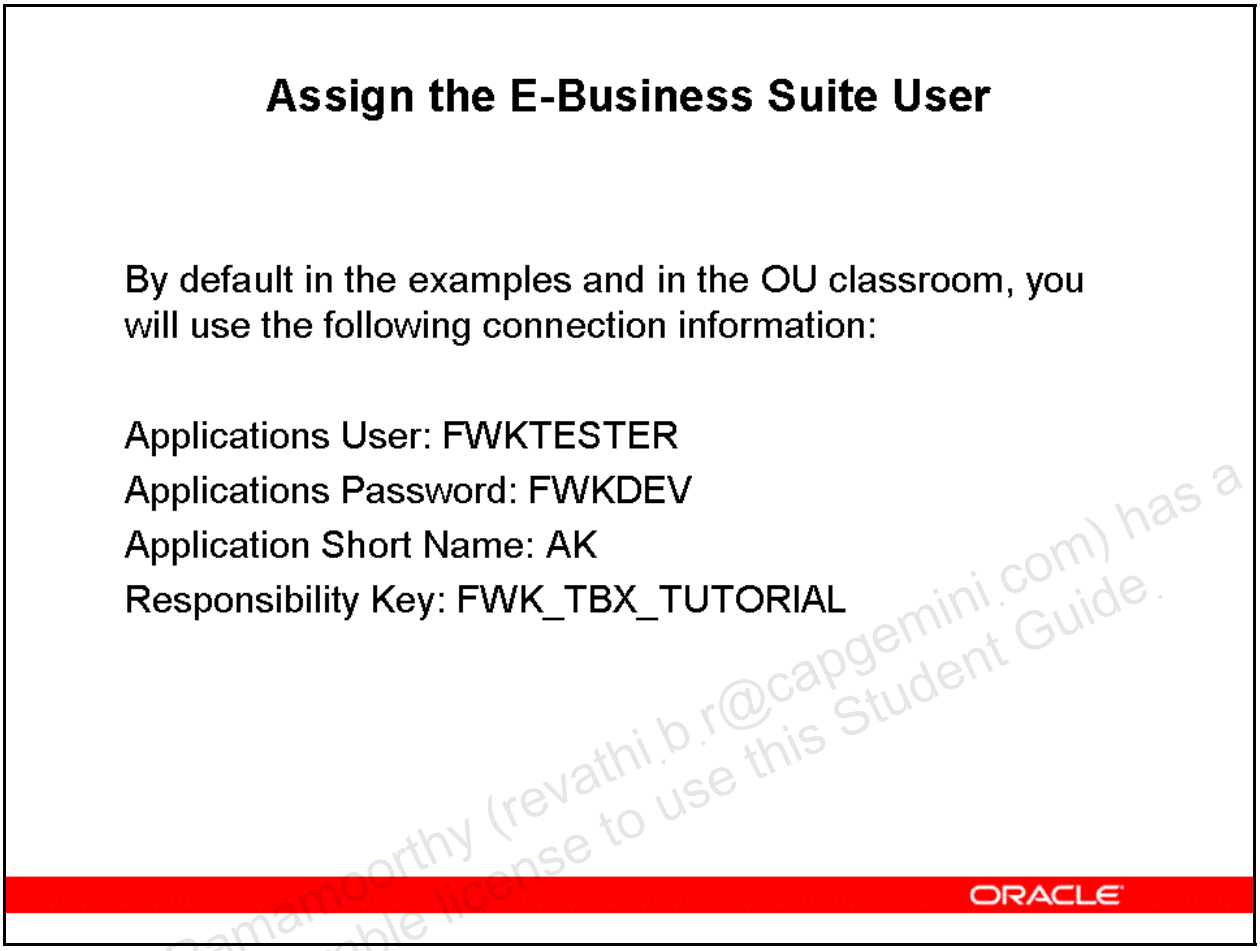
On your course server, the value of the environment variable, INST_TOP will not be initially set. You will need to type the entire path. Your instructor will have the locations for this course.

Create a Shortcut



The example above should be verified with your instructor.

Assign the E-Business Suite User



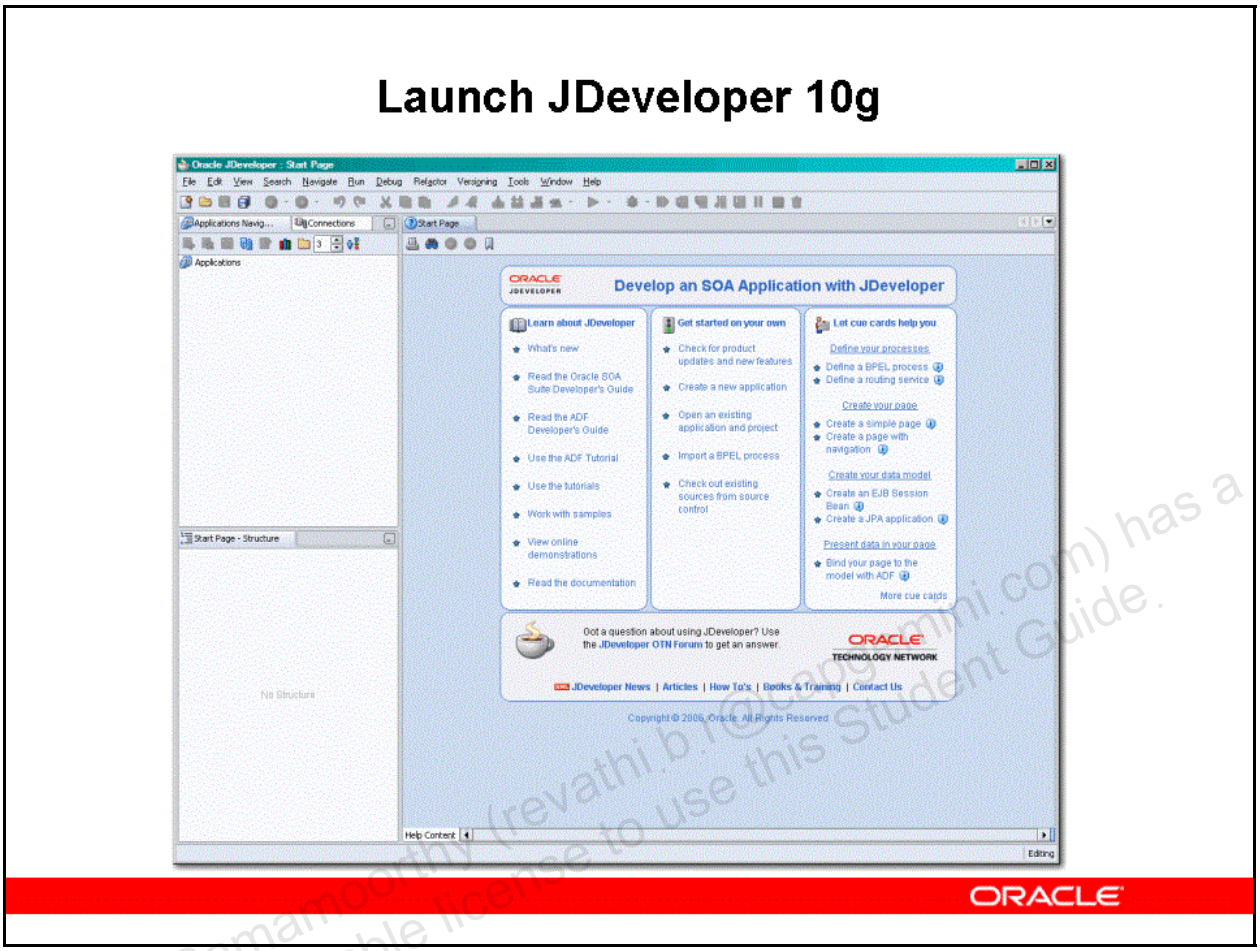
This is the user, password, and responsibility that will be used for the OA Framework course. In your actual development environment, these values can be changed to appropriate values for your development/testing server. The user FWKTESTER is a seeded user. It is advisable that the responsibilities are set correctly for FWKTESTER via the E-Business Suite Functional administrator or System Administration responsibilities. If you get run time errors when trying to run a test page in JDeveloper, look to see that FWKTESTER is set up correctly.

Uncompress Tutorial.zip

In the JDEVBIN directory, there is a file named, Tutorial.zip. Uncompress this file into your JDEVHOME directory.

Because the .zip file contains the ./jdev/… and other directories, it will create the proper structure for you as it is uncompressed.

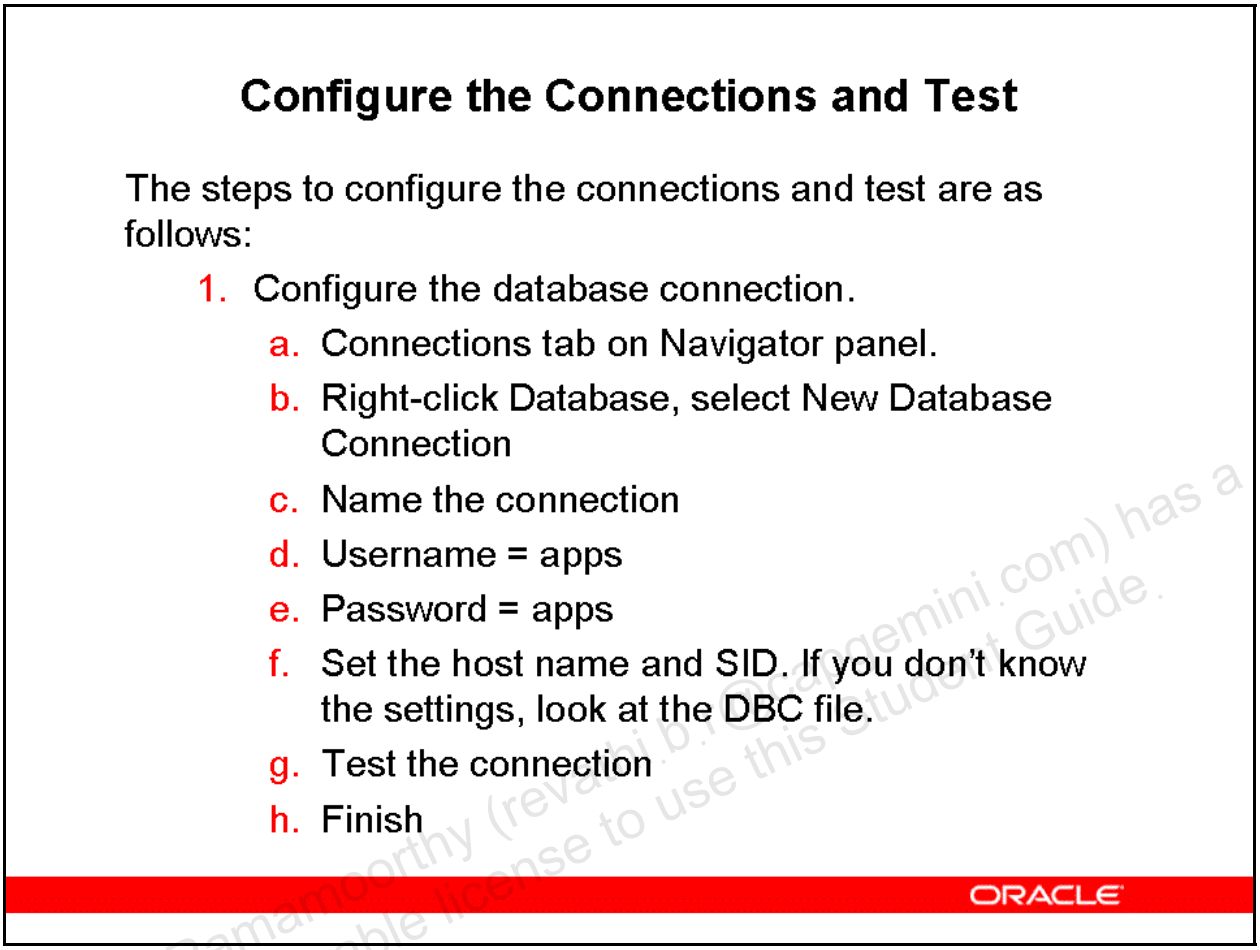
Launch JDeveloper 10g



The method to launch JDeveloper with vary depending on operating system, operating system version, and the shell.

Note: The first time JDeveloper 10g is launched you see the Start Page. The start page can be closed.

Configure the Connections and Test



Username: This is the database user to which you are going to connect. Within the E-Business Suite, the master database user that has access to all the E-Business Suite database objects is **apps**.

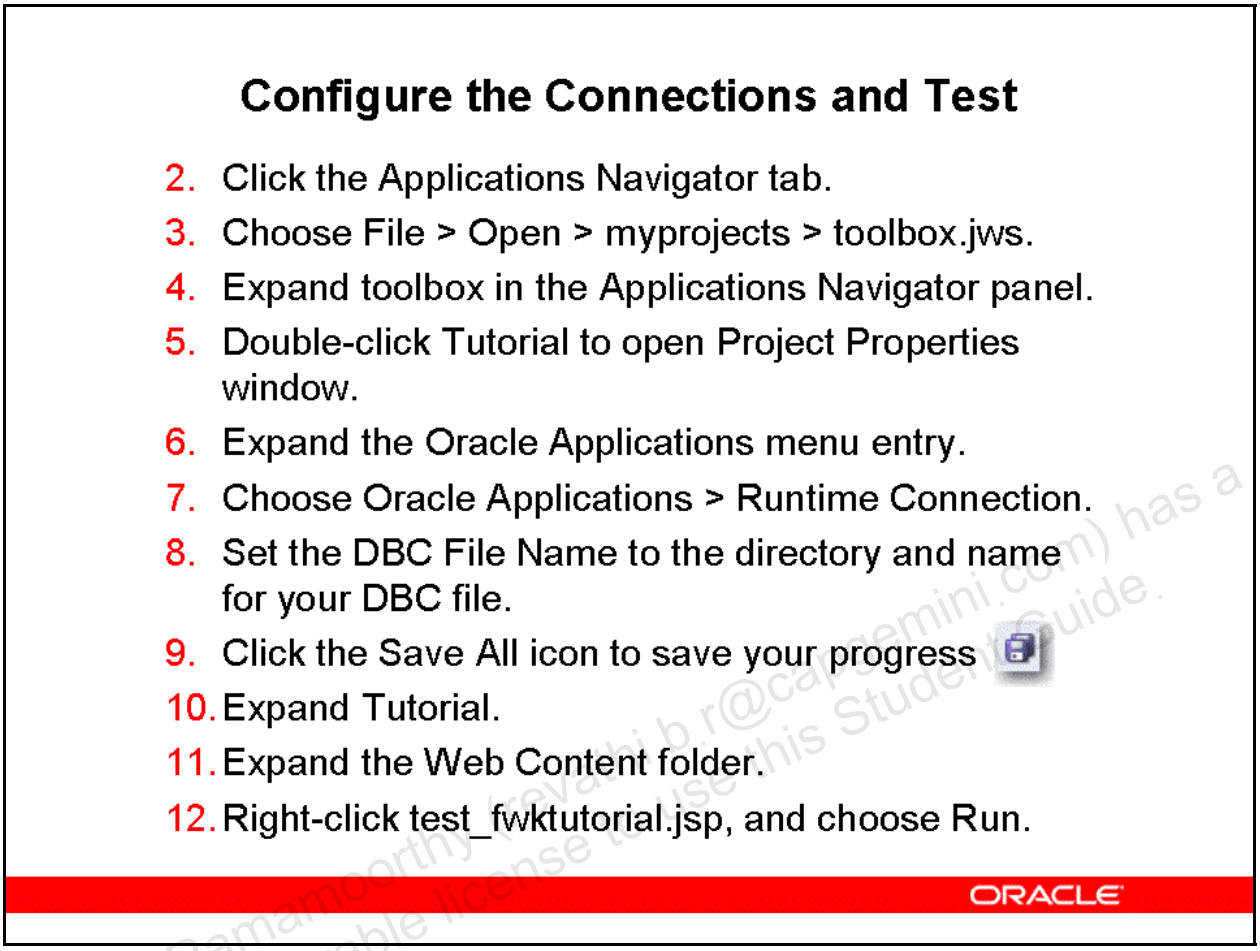
Password: The default password for the apps database user is **apps**.

Note: On production servers or any server requiring security, it is important that this password has been changed.

Role: In defining database privileges, there are three manners in which a particular user might (depending upon their granted security) access a database, Normal, SYSOPER, and SYSDBA. SYSOPER and SYSDBA are expanded privilege roles. Normal is the default method. Since connecting to the database is only done to select, insert, update, and delete tables, Normal is adequate privilege for use in JDeveloper.

Deploy Password is used for classic J2EE deployment and testing. OA Framework does not use a classic J2EE deployment model; so, this setting is not used by OA Framework developers.

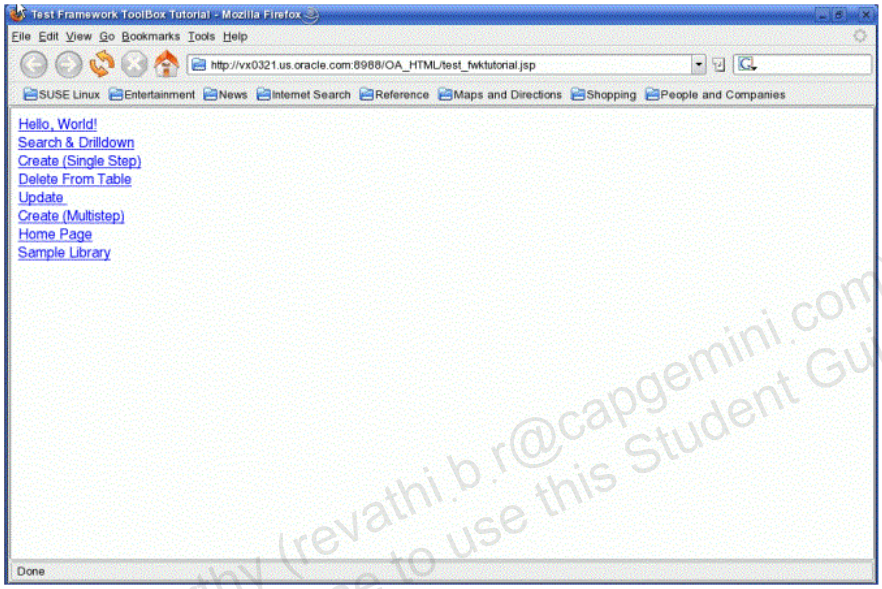
Configure the Connections and Test



The toolbox.jws workspace has many legacy files. Do not use this workspace or any of the files contained for any solutions or code snippets for this course. The toolbox.jws is included for legacy support only.

Configure the Connection and User

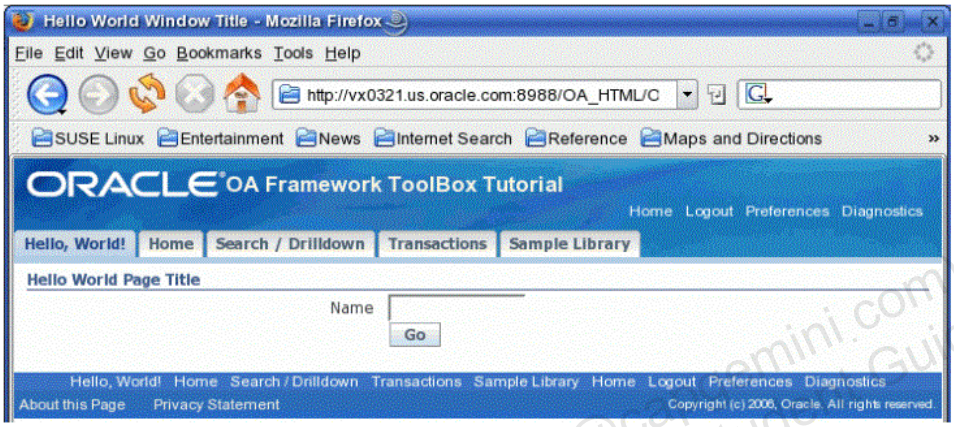
13. Test the test_fwktutorial.jsp Page



ORACLE

Configure the Connection and User

14. You should see the Hello, World! Page as follows:



At this point, if you have seen both the test_fwktutorial.jsp page and the Hello, World! page, you know several things:

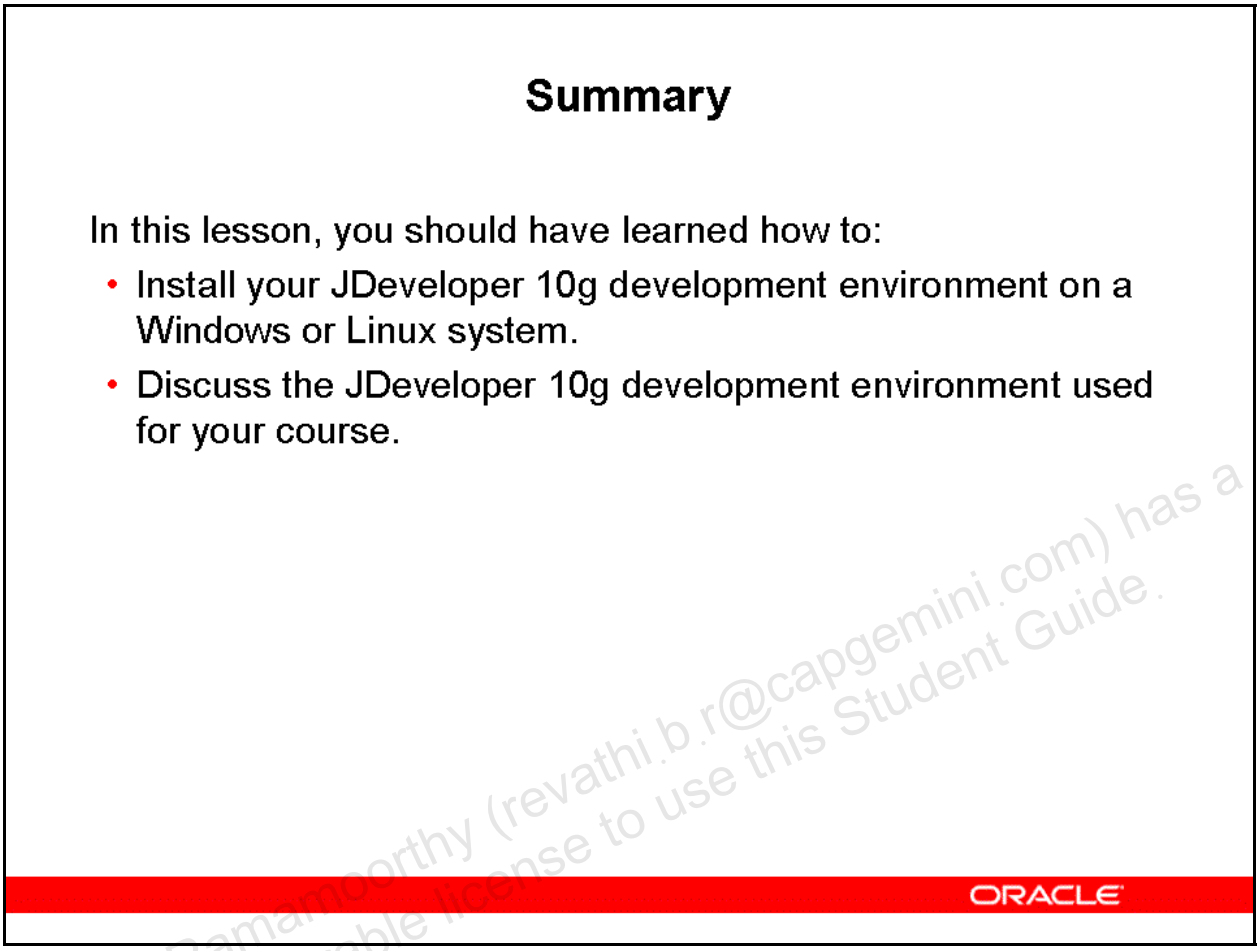
You have connected to a database and tested test_fwktutorial.jsp.

You have connected to an R12 E-Business Suite applications instance, or else Hello, World! would not have worked.

You have a user on that E-Business Suite instance named, FWKTESTER, with a password of FWKDEV, assigned to the AK product, and given the FWK_TBX_TUTORIAL responsibility.

Your installation and configuration is complete.

Summary



Note: The actual steps are contained within the Lab Exercises