

R12.x Extend Oracle Applications: Building OA Framework Applications

Student Guide – Volume I

D61636GC10

Edition 1.0

November 2010

D69573

ORACLE®

Copyright © 2000, 2010, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

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

Introduction to OA Framework	1-1
R12.x Extend Oracle Applications: Building OA Framework Applications.....	1-3
Objectives	1-4
Lesson Objectives	1-5
Agenda – Day 1	1-6
Agenda – Day 2	1-7
Agenda – Day 3	1-8
Agenda – Day 4	1-9
Agenda – Day 5	1-10
Important Terminology	1-11
Personalization vs. Extension	1-13
Architectural Components of OA Framework.....	1-14
Why Java?	1-16
Foundations of Java Programming	1-18
The Java Tech Stack for OA Framework	1-20
Oracle JDeveloper 10g	1-22
Oracle JDeveloper 10g Components	1-24
What is the MVC Design Pattern?.....	1-26
Why Do We Use MVC?.....	1-27
The OA Framework Architecture.....	1-28
What's in BC4J?.....	1-29
What's in UIX?	1-30
What's in AOL/J?	1-31
What's in OA Framework?	1-32
What's in the Metadata Services?	1-33
Additional Resources	1-34
Summary.....	1-35
Concepts of the MVC Design Pattern.....	2-1
R12.x Extend Oracle Applications: Building OA Framework Applications.....	2-3
Lesson Objectives	2-4
What is a JSP Page?	2-5
Key JSP Application Components.....	2-6
What Happens at Runtime?	2-8
What Happens from the Start?	2-10
Behind the Scene.....	2-17
Logical Components of an OA Framework Page	2-27
What is the MVC Design Pattern?	2-28
Model: Business Components for Java	2-29
Model: Application Modules	2-30
Model: Entity Objects	2-31
Model: View Objects	2-32
View: OA Framework-Based Page	2-33
View: Java Objects in a Page.....	2-34
View: A Framework Example	2-35
View: Page Hierarchy.....	2-36
Controller: Controlling UI Behavior.....	2-37

OA Framework MVC Summary.....	.2-38
Summary.....	.2-39
Basics of the Model	3-1
R12.x Extend Oracle Applications: Building OA Framework Applications.....	3-3
Lesson Objectives.....	3-4
Model-layer BC4J Objects	3-5
Encapsulation The "Reuse Onion".....	3-6
General Reuse Rules.....	3-8
Recommended Build Approach.....	3-9
Business Component (BC4J) Packages	3-10
BC4J Package Naming Standards.....	3-11
Application Modules	3-12
Transaction Object	3-15
Application Module Files	3-16
Entity Objects	3-17
Entity Object Creation Standards.....	3-19
Entity Object Automatic Features.....	3-20
Entity Object Files	3-21
View Objects	3-22
View Object Creation Methods.....	3-25
ExpertMode View Objects.....	3-26
View Objects with Entity Objects.....	3-27
View Object Rows	3-28
Creating View Objects.....	3-29
View Object Java Files.....	3-30
View Object Files	3-31
BC4J Database Interactions	3-32
Non-BC4J Method.....	3-33
Read-only Queries	3-34
Step 1: Initial Query.....	3-35
Step 2: Entity Object Population.....	3-36
Step 3: Entity Object Reuse	3-37
Step 4: Entity-derived Attributes.....	3-38
Step 5: Entity Object Fault-in	3-39
Step 6: Entity Object References	3-40
EO/VO Merge.....	3-41
Step 1: EO/VO Merge Resolution	3-42
Step 2: EO/VO Merge Resolution	3-43
Other Model-layer Objects	3-44
Association Objects.....	3-45
Reference Association Objects	3-47
Composition Association Objects.....	3-48
Composition Association Object Behavior	3-49
View Links	3-50
Entity Experts	3-52
Validation AMs and Validation VOs.....	3-53
Summary.....	3-55
Basics of the View.....	4-1

R12.x Extend Oracle Applications: Building OA Framework Applications	4-3
Lesson Objectives.....	4-4
Recommended Build Approach.....	4-5
View-layer Terminology.....	4-6
Workspaces and Projects	4-8
Pages are created in JDeveloper via wizards and are the visual representation of the View.....	4-9
You name a page to reflect the use of the page. You define the package the page is part of.	4-10
A page will have a structure that will display the View attributes that reside in a page hierarchy.....	4-11
Within a page, the attributes that are modified, change what is presented in the View and Page.....	4-12
A Page is tested from within JDeveloper to verify the view is rendering what you expect it to.....	4-14
What Can You Add to the Page?	4-15
Region Styles	4-16
Sub-Region Styles	4-17
Named Children vs. Indexed Children.....	4-18
Item Styles	4-20
Item Style Details	4-21
Shared Regions	4-23
Attribute Sets.....	4-25
Creating Attribute Sets	4-26
Example Attribute Set	4-27
Table Column Template	4-29
Button Template	4-30
Region Header Template	4-31
CSS Styles.....	4-32
Common CSS/XSS Styles	4-33
Extending Other Objects	4-35
Destinations and Links	4-36
Mailto Links	4-37
Lists of Values (LOVs)	4-38
Defining an External LOV	4-40
Reading Model Data	4-42
Writing Model Data.....	4-44
Binding Items to Data.....	4-46
General Naming Rules	4-47
Page and Object Naming Rules	4-50
Styles and Bean Names.....	4-51
Attribute Set Standards	4-52
Attribute Sets.....	4-55
Attribute Set Naming Conventions	4-56
More Attribute Set Standards	4-57
Summary.....	4-58
Basics of the Controller.....	5-1
Lesson Objectives.....	5-4
Recommended Build Approach.....	5-5
Do You Need a Controller?	5-6
Controller Basics	5-7
Common Logic to Code	5-8
Typical Locations for Code.....	5-9
Handling Queries	5-10

View Object initQuery Code5-11
Dynamic WHERE Clauses5-12
Using findByKey Instead of initQuery5-13
Processing a Button Press5-14
Getting Parameters from Requests5-15
Example: Manually-built Search5-16
The Process5-17
Example VOImpl Code5-18
Example AMImpl Code5-20
Example Controller Code5-21
Example Search: Controller5-22
Forwarding to Another Page5-23
Setting Titles with Message Dictionary5-24
Event Flow Overview5-25
Initial Setup Flow5-26
Controller Event Flows in OA Framework5-27
GET Event Flow – Overview5-28
GET Event Flow (1-3)5-29
GET Event Flow (4) Instantiate BC4J and UIX Classes5-31
Example Bean Hierarchy Structure5-32
GET Event Flow (5) processRequest5-33
GET Event Flow (6) Post-Processing5-34
GET Event Flow (7) UIX Renders the Page5-35
POST Event Flow – Overview5-36
POST Event Flow (1 - 3) Submit, Client-Side Validation5-37
POST Event Flow (4 & 5) Validate User and Retrieve State5-38
POST Event Flow (6) Apply Form Data5-39
POST Event Flow (6) More of processFormData5-40
POST Event Flow (7) processFormRequest5-43
Summary5-44
Setting Up Your Development Environment6-1
R12.x Extend Oracle Applications: Building OA Framework Applications6-3
Lesson Objectives6-4
Installing and Setting Up JDeveloper6-5
Configure Your Environment Variables6-7
Get the DBC File6-9
Create a Shortcut6-10
Assign the E-Business Suite User6-11
Uncompress Tutorial.zip6-12
Launch JDeveloper 10g6-13
Configure the Connections and Test6-14
Configure the Connection and User6-16
Summary6-18
OA Framework State Management7-1
R12.x Extend Oracle Applications: Building OA Framework Applications7-3
Lesson Objectives7-4
Architectural Overview – Session and Cookies7-5
Architectural Overview – JVM7-6

State Caches in OA Framework7-7
Root Application Modules7-8
Default Root Application Module Retention.....	.7-10
Retaining the Root Application Module7-11
Recommendation: Multipage Flow7-13
Recommendation: Multipage Flow with Side Trip7-14
Recommendation: Side Trip with Extended Page7-15
Recommendation: Unrelated Pages Flow.....	.7-16
Servlet Session7-17
Applications User Session (ICX Session)7-18
Page Context7-19
Request and Page Boundaries7-20
Request.....	.7-23
Ways to Pass Parameters.....	.7-24
URL Parameters: Tokens, Encryption, Encoding7-26
Passivation.....	.7-29
Application Module Pooling7-31
Application Module and Connection Pooling7-33
Application Module Pooling Process7-35
Monitoring the AM Pool.....	.7-39
JDBC Connection Pooling Process7-40
Monitoring the JDBC Connection Pooling7-45
Determining User Load7-47
Back Button Usage Goals7-48
Back Button Scenario #1.....	.7-49
Back Button Scenario #2.....	.7-50
Back Button Scenario #3.....	.7-51
Addressing Consistent Behavior7-52
Further Study of Back Button7-53
Summary.....	.7-54
Introduction to JDeveloper 10g with OA Extension.....	.8-1
R12.x Extend Oracle Applications: Building OA Framework Applications.....	.8-3
Lesson Objectives8-4
Oracle JDeveloper 10g with OA Extension8-5
Oracle JDeveloper 10g Components8-7
Applications and Workspaces8-9
Creating a Workspace.....	.8-11
Projects8-12
Creating Project Wizard Start Screen8-14
Enter Project Information and Default Project Package8-15
Runtime settings for a Project8-16
Establish a Database Connection8-17
Provide the SID for the Database Connection8-19
Establishing a Database Connection8-20
Database Connection Information.....	.8-21
Testing a Database Connection8-22
Project Properties.....	.8-23
Project Properties – Oracle Applications8-24
Directory Structure8-25

Creating JDeveloper Items.....	8-26
Exploring Java Files.....	8-27
Exploring Other Objects - Wizards	8-28
Exploring Other Objects – UI Objects	8-29
Finding Methods and Fields	8-30
Supporting Code Development with Profiler and Code Coach.....	8-32
New Code Editor Features	8-34
Customizing JDeveloper 10g with OA Extension	8-36
Refactoring Java Files.....	8-37
JDeveloper Help System.....	8-40
Obtaining Help on a Topic.....	8-42
Oracle JDeveloper Debugger.....	8-43
Breakpoints	8-45
Breaking on Exceptions	8-47
Debugger Windows.....	8-48
Stepping Through a Program	8-50
Watching Data and Variables.....	8-52
Debugging Declarative Applications.....	8-54
More Debugging Tips.....	8-55
Understand BC4J Interactions	8-56
Debugging Validation and Commits	8-58
Summary.....	8-59
Implementing a Query Page and Drill Down Page	9-1
R12.x Extend Oracle Applications: Building OA Framework Applications.....	9-3
Lesson Objectives.....	9-4
Course Method.....	9-5
Finished Page Before Search	9-7
Finished Page After Search	9-8
Finished List of Values Page	9-9
Finished Drilldown-to-Details Page	9-10
Summary.....	9-11
Implementing a Create Page	10-1
R12.x Extend Oracle Applications: Building OA Framework Applications.....	10-3
Lesson Objectives.....	10-4
Implementing a Poplist	10-5
Extending a Shared Region	10-6
Creating a New Row	10-7
Initializing a View Object	10-8
Creating and Initializing a VO Row	10-9
Getting the Data	10-10
Saving a Row to the Database.....	10-11
Lab – After Create Basics	10-12
Lab – After Validations	10-13
Lab – After Partial Page Rendering.....	10-15
Summary.....	10-17
Implementing a Delete Page.....	11-1
R12.x Extend Oracle Applications: Building OA Framework Applications.....	11-3
Lesson Objectives.....	11-4

Error Handling Overview	11-5
Exception Types	11-6
Exception Classes	11-8
Message Types	11-9
Message Dictionary	11-10
Implementing Message Dictionary	11-11
Instantiating Attribute-level Exceptions	11-12
Instantiating Row-level Exceptions	11-13
Attribute Value - EO Example	11-14
Attribute Value - VO Example	11-15
Row Value - EO Example	11-16
Row Value - VO Example	11-17
Messaging Flows	11-18
Inline Messages	11-19
Dialog Pages	11-20
Switchers	11-22
Table Content Switcher Abilities and Limits	11-23
Implementing Table Content Switchers	11-24
Query Page with Non-Deleteable Employee	11-25
Query Page with Deleteable Employee	11-26
Warning Dialog	11-27
Confirmation Message	11-28
Summary	11-29
Implementing an Update Page	12-1
R12.x Extend Oracle Applications: Building OA Framework Applications	12-3
Lesson Objectives	12-4
Locator Elements	12-5
Breadcrumbs	12-6
Page Navigators	12-7
Record Navigators	12-8
Trains	12-9
Implementing Trains	12-10
Single-Page Update	12-11
Update Confirmation	12-13
Multi-Page Update	12-14
Multi-Page Update Confirmation	12-17
Summary	12-18
OA Framework Development Concepts and Standards	13-1
R12.x Extend Oracle Applications: Building OA Framework Applications	13-3
Lesson Objectives	13-4
General Coding Standards	13-5
Coding Terminology	13-6
OA Framework Standard Considerations	13-7
E-Business Suite Standard Considerations	13-8
E-Business Suite Standard DB Objects	13-10
Oracle BLAF Standards	13-11
E-Business Suite Java Standards	13-12
OA Framework File Standards	13-13

Files in a Typical OA Framework Application	13-14
Standard File Suffix Abbreviations	13-16
Package Names	13-18
File Names	13-21
Region and Item Names	13-22
OA Framework Model Standards	13-23
OA Framework View Standards	13-24
OA Framework Controller Standards	13-25
Summary	13-26
Deploying OA Framework Applications	14-1
R12.x Extend Oracle Applications: Building OA Framework Applications	14-3
Objectives	14-4
Storing Personalizations	14-5
Directory Structure	14-6
Directory Structure - Layer Values	14-7
Toolset	14-8
Functional Administrator Personalization UI	14-9
export.bat / import.bat- Syntax	14-10
export.bat/import.bat - Example	14-11
Command Line – XMLExporter/XMLImporter	14-12
export.bat vs. XMLExporter	14-13
Import Substitutions – JPXImport.bat	14-14
JPXImport.bat - Syntax	14-15
Import Substitutions – JPXImporter	14-16
Inspecting the MDS Repository	14-17
JDR_UTILS PL/SQL package APIs	14-18
List the Personalizations Done on a Page	14-25
Inspect Personalizations	14-26
Deploying Personalizations	14-28
Extract the Personalizations – Functional Administrator Page	14-29
Set Personalization Document Root Path	14-30
Import/Export Personalizations	14-31
Extract the Personalizations – Select the Page	14-32
Extract the Personalizations – Export to File System	14-33
Upload Personalizations into Production Instance – Functional Administrator Page	14-34
Upload Personalizations into Production Instance – Exported Personalizations	14-35
Upload Personalizations into Production Instance – Import from File System	14-36
Extensions	14-37
OA Page Extensions	14-38
Deployment of Page Extensions	14-39
1.Copy .java Classes	14-40
2. Import Substitutions	14-41
3. Import OA Component Definitions	14-42
View The Deployed Extensions	14-43
BC4J Extensions	14-44
Deployment of Business Logic Extensions	14-45
Summary	14-46

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 (<i>but</i> Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do <i>not</i> save changes to the database. For further information, see <i>Oracle7 Server SQL Language Reference Manual</i> . Enter user_id@us.oracle.com, where <i>user_id</i> 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
------------	---------	---------

Copyright © Oracle, 2008. All rights reserved.

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 <i>role</i>
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.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

Introduction to OA Framework

Chapter 1

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

R12.x Extend Oracle Applications: Building OA Framework Applications

R12.x Extend Oracle Applications: Building OA Framework Applications

Introduction to OA Framework

ORACLE

Objectives

Objectives

After completing this course, you should be able to do the following:

- Set-up your OA Framework development environment
- Identify the concepts, architecture and components of OA Framework applications
- Create an OA Framework application that selects, inserts, updates, and deletes records in an E-Business Suite instance
- Identify the processes and procedures of deploying OA Framework applications into an E-Business Suite instance

ORACLE

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.

Lesson Objectives

Lesson Objectives

After completing this lesson, you should be able to:

- List the course objectives and agenda
- Identify the architecture and components of OA Framework applications

ORACLE®

Agenda – Day 1

Agenda – Day 1

- Introduction to OA Framework
- Introduction to Model-View-Controller design pattern
- Basics of the Model
- Basics of the View
- Basics of the Controller
- Setting up your OA Framework development environment

ORACLE

Agenda – Day 2

Agenda – Day 2

- Introduction to JDeveloper 10g with OA Extension
- Lab: First OA Framework Page
- OA Framework State Management
- Implementing a Query and Drill-down
- Lab: Implementing a Query and Drill-down

ORACLE

Agenda – Day 3

Agenda – Day 3

- Implementing a Create
- Lab: Implementing a Basic Create
- Lab: Implementing Validations
- Lab: Implementing Partial Page Rendering

ORACLE

Agenda – Day 4

Agenda – Day 4

- Implementing a Delete
- Implementing an Update
- Lab: Implementing a Delete
- Lab: Implementing a Basic Update
- Lab: Implementing a Multi-page Update

ORACLE

Agenda – Day 5

Agenda – Day 5

- Basic E-Business Suite Development Standards
- Basics of E-Business Suite Security
- Basics of E-Business Suite Deployment

ORACLE

Important Terminology

Important Terminology

- Installation
- Deployment
- Configuration
- Personalization
- Extension
- Customization

ORACLE

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

Personalization vs. Extension

- Personalization – the ability to declaratively alter (in XML) the page's UI to meet the business or user needs.
- Extension – the ability to programmatically (in Java) extend the page's functionality.

ORACLE®

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

Architectural Components of OA Framework

- Written in Java and other web technologies
- Uses JDeveloper 10g with OA Extension
- Uses the following Oracle components
 - Business Components for Java (BC4J)
 - User Interface XML (UIX)
 - Application Object Library for Java (AOL/J)
 - OA Framework specific components
 - Metadata Services (MDS)
 - and other Oracle technology contributors

ORACLE®

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?

Why Java?

- Object-oriented
- Platform independent (Write Once – Run Many)
- Network ready
- Secure
- Robust
- Industry-standard
- Open source

ORACLE®

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.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

Foundations of Java Programming

Foundations of Java Programming

- Encapsulation
- Inheritance
- Polymorphism

ORACLE®

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.

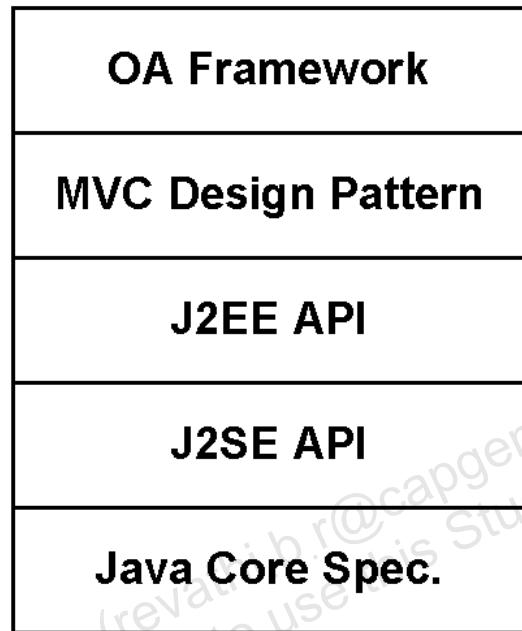
Polymorphism

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

The Java Tech Stack for OA Framework



ORACLE®

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

Oracle JDeveloper 10g

Development

UML

ADF

JSF

EJB

XML



Source control

Debug

Exchange

Database

HTML

Deployment

ORACLE

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

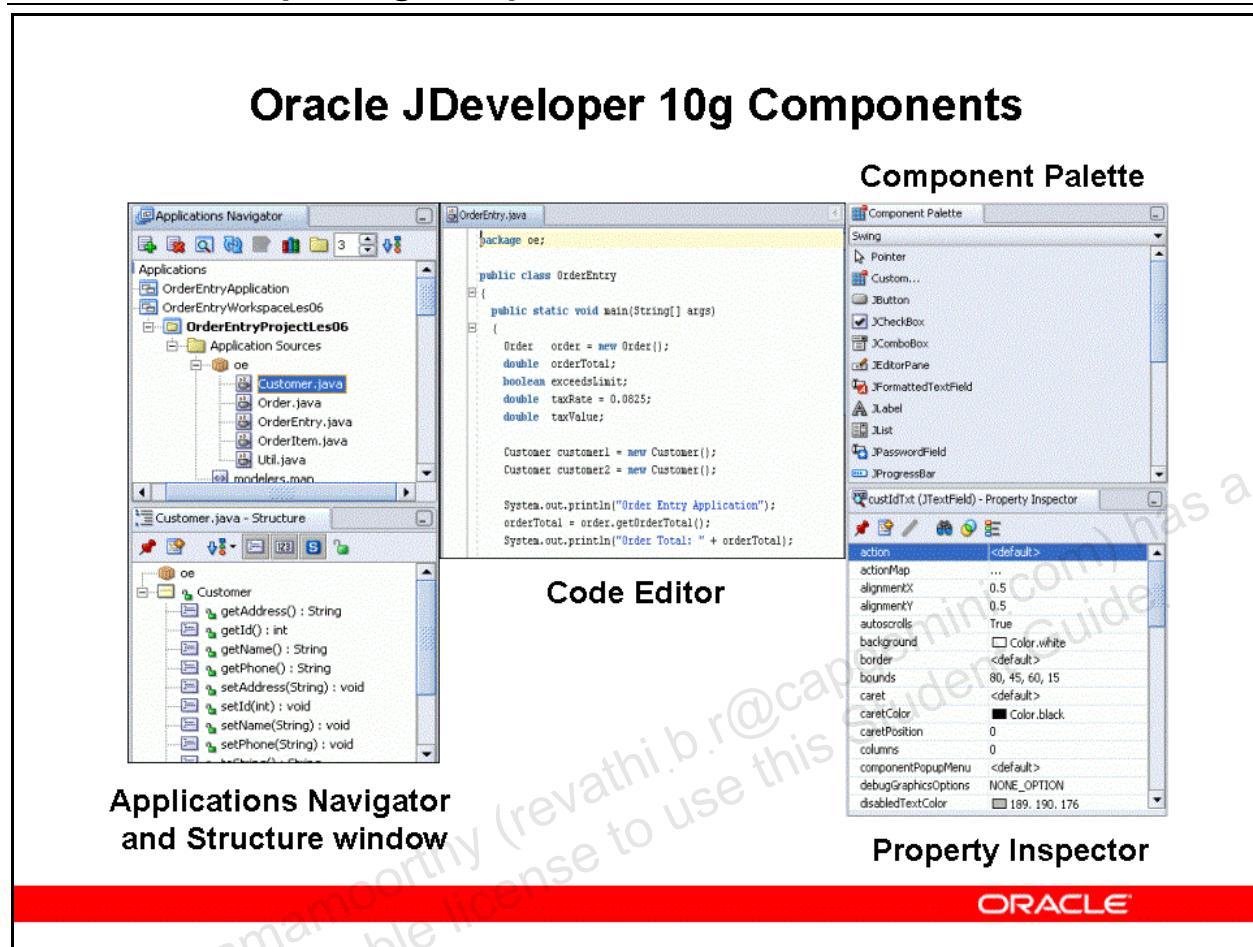
Customize the Component Palette

Customize the IDE environment

Select JDeveloper's embedded Java EE server
Arrange the windows in the IDE

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

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.

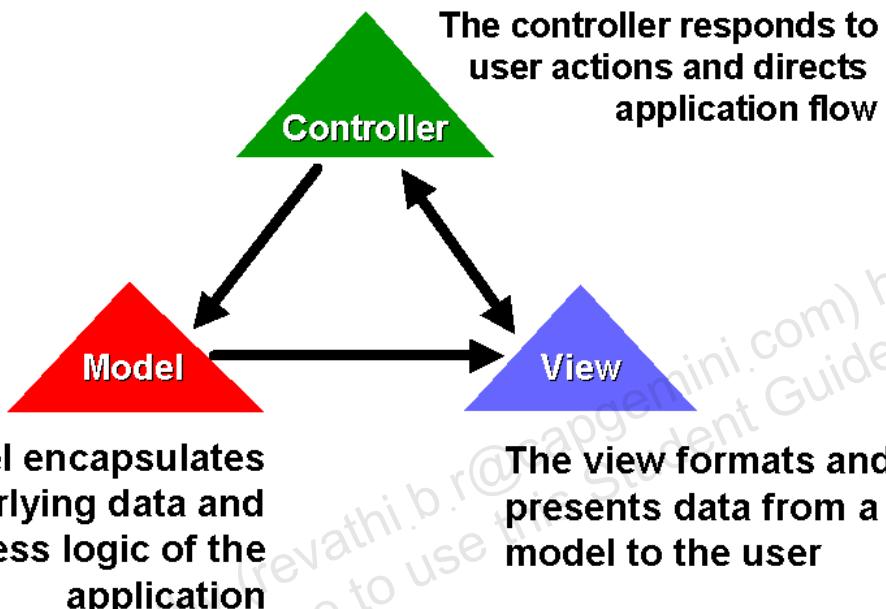
Code Editor

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.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

What is the MVC Design Pattern?

What is the MVC Design Pattern?



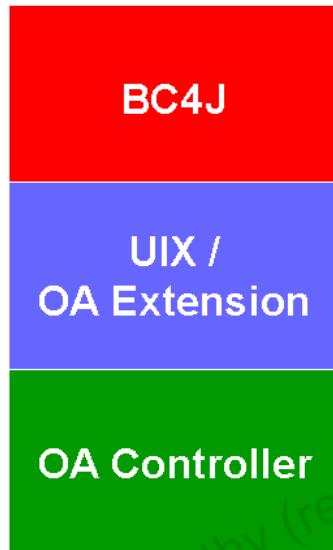
ORACLE®

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?

Why Do We Use MVC?

- It provides a flexible architecture for applications that is more easily programmed and managed over the complete lifecycle.

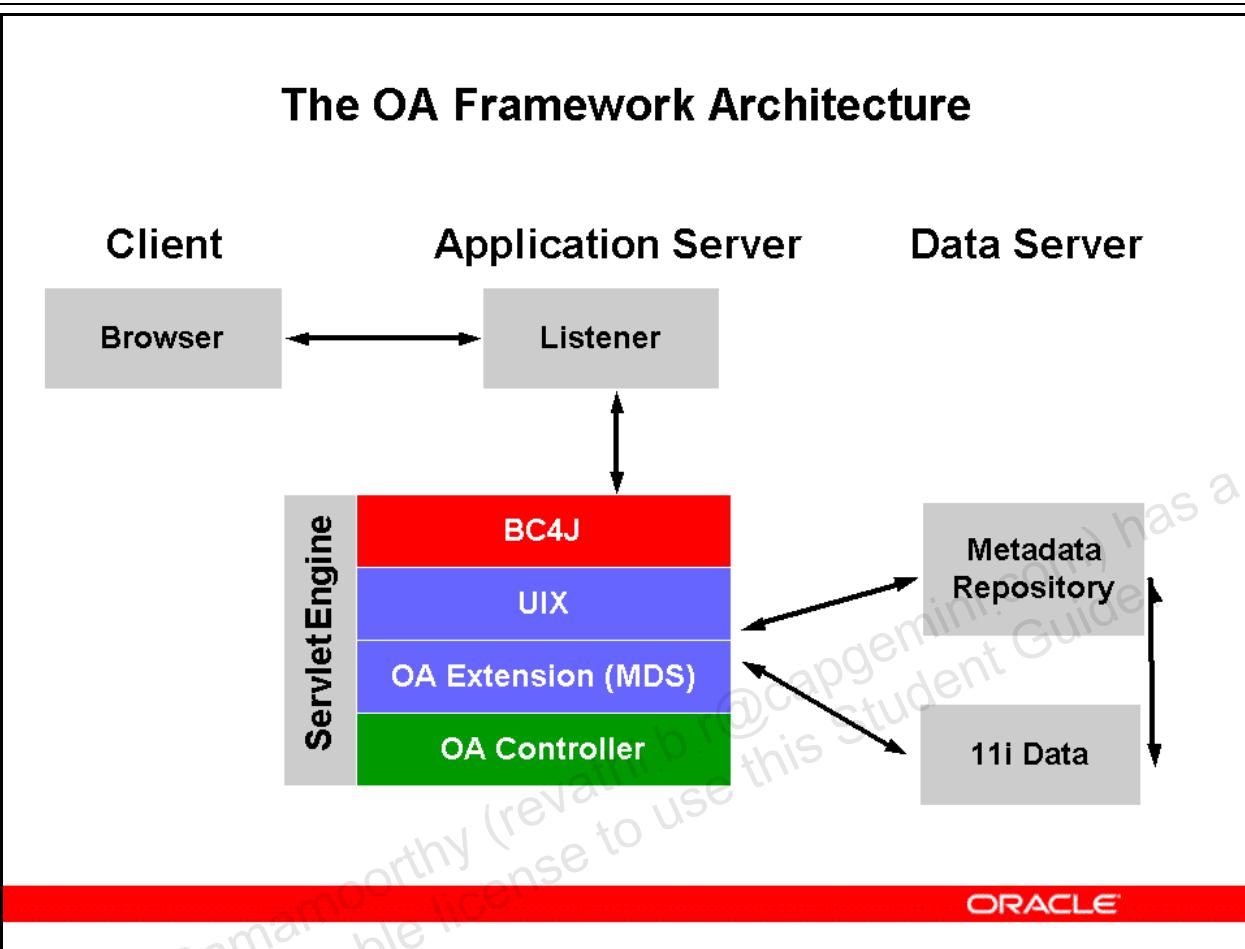


- **Model** – Business logic encapsulated in Business Components for Java view objects and entity objects
- **View** – Common UIX-based HTML components used throughout Applications
- **Controller** – OA Controller responds to user actions, directs application flow

ORACLE®

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?

What's in BC4J?

- BC4J is the M (Model) layer of the MVC pattern.
- Includes the following basic BC4J components.
 - Application Modules (AMs)
 - Entity Objects (EOs)
 - View Objects (VOs)
 - and others
- Includes additional capabilities.
 - Data binding
 - AM pooling and session management
 - Database transactions

ORACLE®

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?

What's in UIX?

- UIX is part of the V (View) layer of the MVC pattern.
- UIX underlies the OA Framework UI components.
- Using both OA Framework and UIX allows OA Framework applications to have a consistent E-Business Suite look and feel.
- Includes specific simple and complex UI components, like the following:
 - Fields
 - Buttons
 - Tables
 - LOVs
 - and others

ORACLE®

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?

What's in AOL/J?

- AOL/J is the layer that implements the common features expected in an E-Business Suite application, like the following:
 - Function security
 - Connection pooling
 - Flexfields
 - Profile options
 - Hierarchies
 - Online Help
 - Encryption
 - Thin Client Framework (TCF)

ORACLE®

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?

What's in OA Framework?

- It is the programmatic ‘glue’ which integrates these technologies.
- It provides consistent UI components to E-Business Suite applications.
- Additionally, it provides methods for the following components:
 - Personalizations
 - Safe interfaces for access from both the Model (server) and View (user interface) code.
 - Implementation of LAFs (Look and Feel)
 - Final rendering

ORACLE®

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.

Additional Resources

Additional Resources

Further reading:

- OA Framework Developer's Guide: Anatomy of an OA Framework Page
- OA Framework Javadoc

ORACLE®

Summary

Summary

In this lesson, you should have learned how to:

- List the course objectives and agenda.
- Identify the architecture and components of OA Framework applications.

ORACLE

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

Concepts of the MVC Design Pattern

Chapter 2

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

R12.x Extend Oracle Applications: Building OA Framework Applications

R12.x Extend Oracle Applications: Building OA Framework Applications

Concepts of the MVC Design Pattern

ORACLE

Lesson Objectives

Lesson Objectives

After completing this lesson, you should be able 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

What is a JSP Page?

What is a JSP Page?

JSPs, or JavaServer Pages, are built to generate dynamic HTML pages. JSPs are text documents, similar to HTML. JSPs consist of a mix of HTML elements and special JSP elements called tags with specific tag libraries.

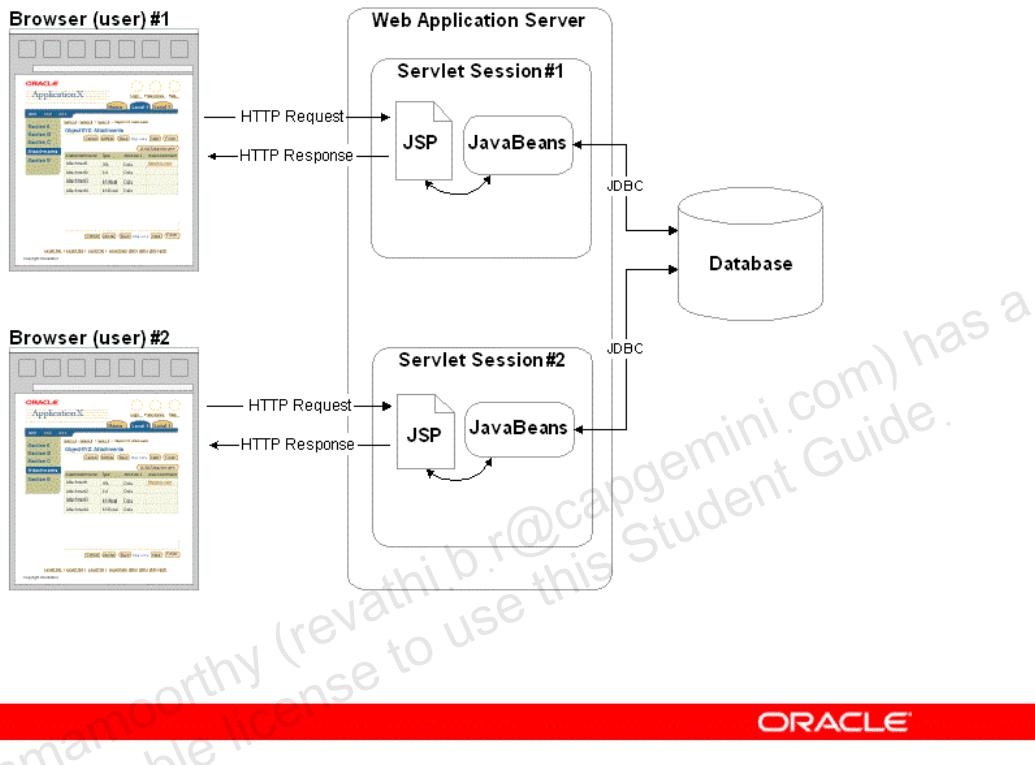
ORACLE

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

Key JSP Application Components



ORACLE®

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.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

What Happens at Runtime?

What Happens at Runtime?

- Step 1: Browser sends an HTTP request
 - HTTP GET
 - HTTP POST
- Step 2: Request routed to JSP by Application Server
- Step 3: JSP delegates to one or more JavaBeans.
- Step 4: Browser displays the HTML it receives.

ORACLE®

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?

What Happens from the Start?

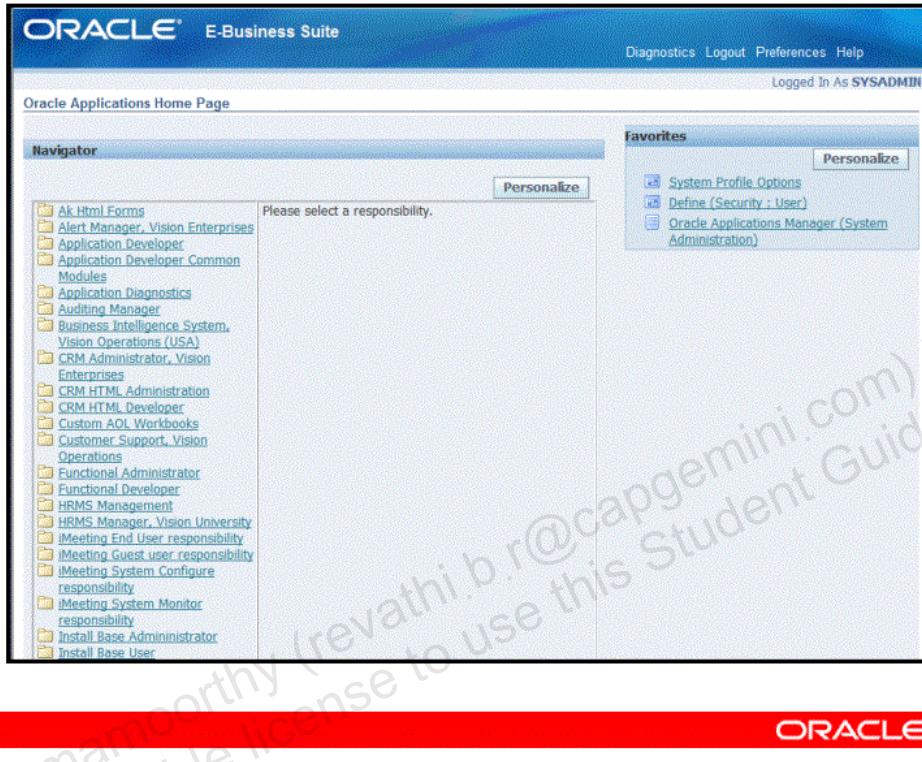


While it may seem trivial, just seeing the Login page indicates a lot of very good information. If you see this page, it means the following:

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?

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 you 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:

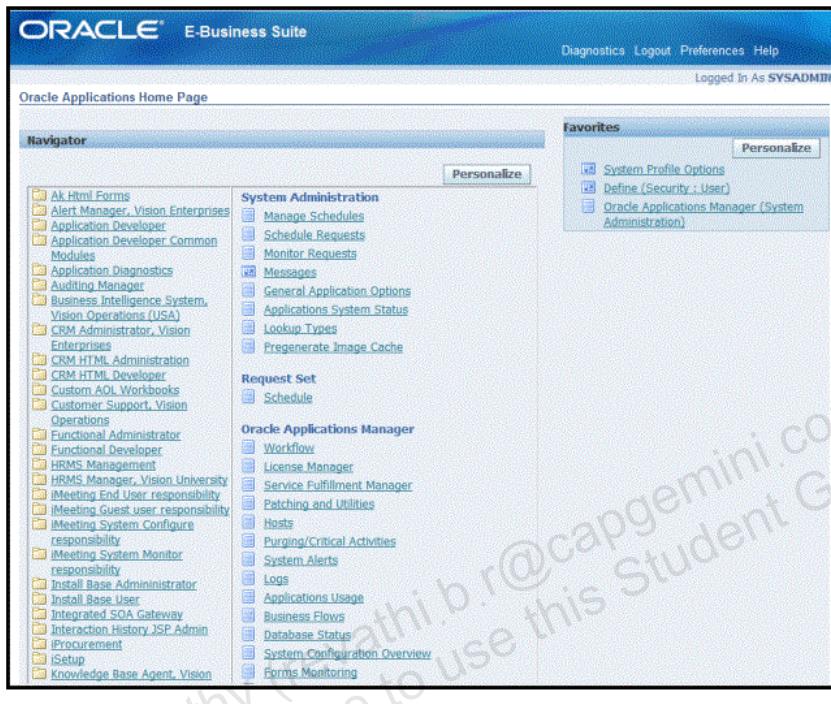
- 4.

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?

What Happens from the Start?



ORACLE®

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?

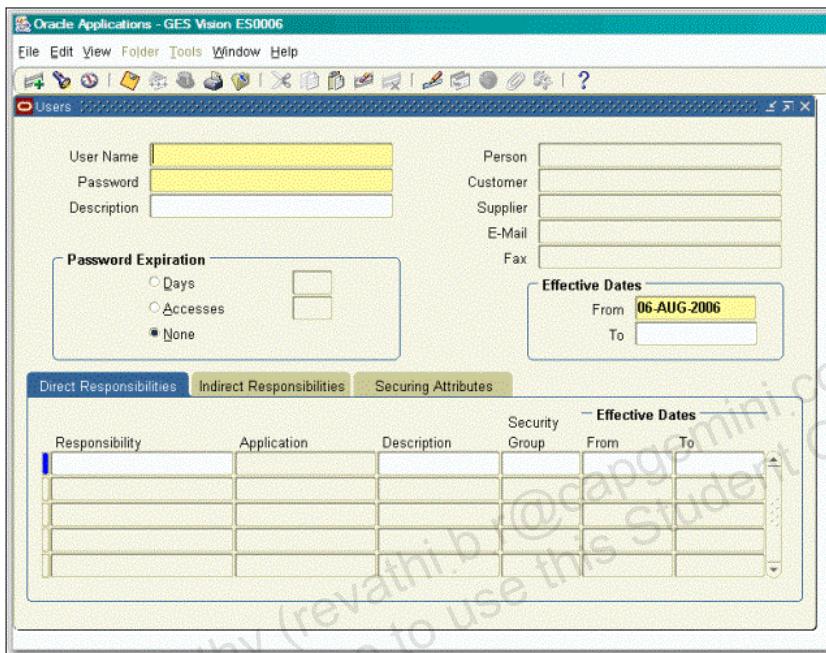
What Happens from the Start?

The screenshot shows the Oracle Applications Administration interface. The top navigation bar includes links for Home, Logout, Preferences, Help, and Diagnostics. Below this, a secondary menu bar has tabs for Security, Core Services, Personalization, File Manager, Portletization, Grants, Permissions, and Permission Sets. The Grants tab is selected. A search bar at the top right contains the text "Save Search". The main content area is titled "Search" and includes fields for Name, Grantee Type (set to All Users), Set, and Object, each with a search icon. Below these fields are "Go" and "Clear" buttons. A "Create Grant" button is located above a table header. The table header includes columns for Name, Grantee Type, Grantee, Set, Object, Data Context Type, Access Policy, Last Update, Duplicate, Update, and Delete. The message "No search conducted." is displayed below the table. At the bottom of the page, there are links for Security, Core Services, Personalization, File Manager, Portletization, Home, Logout, Preferences, Help, Diagnostics, About this Page, and Privacy Statement. The copyright notice "Copyright (c) 2006, Oracle. All rights reserved." is also present.

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?

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?

What Happens from the Start?

The screenshot shows the Oracle Applications Administration interface. The top navigation bar includes links for Home, Logout, Preferences, Help, and Diagnostics. Below this, a secondary navigation bar has tabs for Security, Core Services, Personalization, File Manager, Portletization, Grants, Permissions, and Permission Sets. The 'Grants' tab is selected. A search bar at the top right contains the text 'Save Search'. The main content area is titled 'Search' and includes fields for Name, Grantee Type (set to All Users), Set, and Object, each with a search icon. Below these are 'Go' and 'Clear' buttons. A 'Create Grant' button is located above a table header. The table header includes columns for Name, Grantee Type, Grantee, Set, Object, Data Context Type, Access Policy, Last Update, Duplicate, Update, and Delete. The message 'No search conducted.' is displayed below the table. At the bottom of the page, there are links for Security, Core Services, Personalization, File Manager, Portletization, Home, Logout, Preferences, Help, Diagnostics, About this Page, and Privacy Statement. The copyright notice 'Copyright (c) 2006, Oracle. All rights reserved.' is also present.

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.

ORACLE®

Behind the Scene

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

Behind the Scene

- Application reads the definition for our page from the MDS (Metadata Service). The MDS is a set of tables in the database where XML-based items are kept. Pages (PG) and Regions (RN) are two common XML-based items.
- Application reads the personalizations from the MDS. Personalizations are XML-based entries stored in the MDS.

ORACLE

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

Behind the Scene

- After the page and all its personalizations is read from MDS, the AM (Application Module) of the page is determined.
- The AM is the means by which OA Framework pages retain a transaction, and by extension the connection to the database.
- If this page is part of a multi-page transaction, any AM for this page will be nested into an existing root AM.
- If this is a new part (not part of an existing transaction), the AM for this page will establish itself as the root AM.
- Root AMs maintain the transaction state and database connection for you page.

ORACLE®

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

Behind the Scene

- Once the AM is known, the AM knows all the View Objects (VOs) that are assigned to it.
- VOs are the mechanism by which data is queried from the database. A VO is essentially a SQL SELECT statement.
- Some pages will only need query (SELECT) capabilities, if so, there will likely be limited BC4J objects.

ORACLE®

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

Behind the Scene

- Some pages are more sophisticated than just query pages. These pages will actually be able to INSERT, UPDATE, and DELETE data from the database. In those cases, the VOs will know about the Entity Objects (EOs) they use.
- EO_s are essentially BC4J representations of Tables in the database.

ORACLE®

The EO_s 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 EO_s, 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 EO_s can be joined. There are also objects like Entity Experts, Validation AMs (VAMs), and Validation VOs (VVOs) to name some others.

Behind the Scene

Behind the Scene

- Each time a new BC4J item is read, it is added to the root AM. Since the root AM is the transaction, it will ultimately be responsible for the commit or rollback of any data from the user's interaction with the page.

ORACLE®

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

Behind the Scene

- Looking at the individual page, the Controllers (COs) are read.
- Some pages are simple enough that no controllers are needed. Most of these are query-only pages.
- Some pages, and their associated regions, have multiple controllers defined.

ORACLE®

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

Behind the Scene

- Any data needed by the page is associated to the individual UI objects.
- And finally, with everything for this page having been instantiated, run, and bound, the page is displayed ... **ALMOST ...**
- The individual UI objects are just that, generic objects. Those objects generate code that must be rendered to its final state. In most cases, this state is HTML and Javascript sent to a web browser. But, it could also be WML sent to a PDA or SmartPhone, and other special cases.

ORACLE®

Behind the Scene

Behind the Scene

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

The screenshot shows the Oracle Applications Administration Profiles search interface. At the top, there's a navigation bar with links like Security, Core Services, Personalization, File Manager, Portletization, Lookups, Messages, Profile Categories, Profiles (which is selected), Functions, Menus, Caching Framework, and Personalization. Below the navigation is a sub-menu for Profiles. A large search section titled 'Search' contains fields for Name, Code, Owning Application, Category, and Category Application. It also includes dropdowns for Hierarchy Type (set to Any) and Access Levels (with Site, Application, User, and Server checkboxes). There are search buttons for each category and a 'Go' button. Below the search area is a 'Define Profile Values' section with columns for Select Name, Level, Profile Value, and Update Value. A note says 'No search conducted.' At the bottom, there's a 'Create Profile' button and a red footer bar with the Oracle logo.

Behind the Scene

Behind the Scene

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

The screenshot shows the Oracle Applications Administration Profiles page. At the top, there's a search bar with fields for Name (FND: Diagnostics), Code, Hierarchy Type (Any), and various category and application filters. Below the search is a section titled 'Access Levels' with fields for Responsibility, Organization, Application (User/Server), and a checkbox for 'Profiles with No Values'. A 'Go' button is present. At the bottom, there's a table titled 'Define Profile Values' with columns: Select Name, Code, LevelValue, Profile Value, and Update Value. One row is selected: FND: Diagnostics, FND_DIAGNOSTICS, Site, Yes, and a pencil icon. A 'Create Profile' button is at the bottom right. The footer includes links for Security, Core Services, Personalization, File Manager, Portletization, Home, Logout, Preferences, Help, and Diagnostics, along with a copyright notice for Oracle 2008.

ORACLE®

Logical Components of an OA Framework Page

Logical Components of an OA Framework Page

- The UI components that the user sees: Most of those objects are defined declaratively via XML, but some of them are programmatically defined via Java.
- The database components, the transaction, and the connections to database resources
- The event handling that responds to the user's interactions on the page

ORACLE

Event Handling:

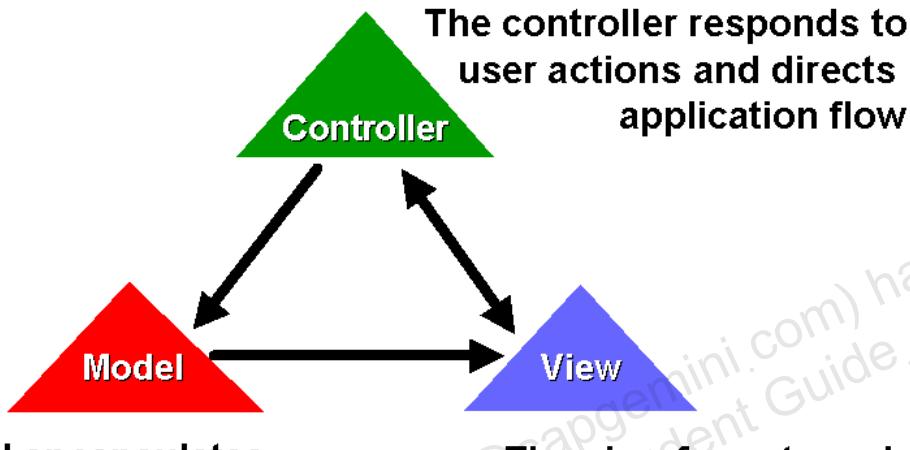
In traditional client/server applications, you have the option of handling events ranging in granularity from very low-level mouse movements to field, region and finally, window-level scope. Furthermore, when you communicate from the client to the server, you can send a single value to be validated back 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?

What is the MVC Design Pattern?



The model encapsulates underlying data and business logic of the application

The controller responds to user actions and directs application flow

The view formats and presents data from a model to the user

ORACLE®

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

There are three basic component classes

- The Application Module – a container for related BC4J objects
- Entity objects – encapsulate business rules
- View objects – present data to the Framework page

ORACLE®

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

Model: Application Modules

- Application Modules serve as containers for related BC4J objects
- The objects are related by participating in the same task (for example, a multi-page UI for creating purchase orders)
- Application Modules provide transaction context
- Application Modules establish database connections
- Application Modules may be nested to provide more complex application modules

ORACLE®

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

Model: Entity Objects

Entity objects

- Encapsulate business rules and logic
- Are used by any program that inserts, updates, or deletes data
- Provide consistent data validation across all applications
- May be linked to other entity objects by entity association objects

ORACLE®

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

Model: View Objects

View Objects

- Encapsulate a database query
- Provide iteration over the query result set
- Primarily based on Eos.
- May be based on plain SQL for certain types of tasks
- May be based on multiple entity objects if the UI is for update
- Provide single point of contact for getting and setting entity object values
- May be linked together by view links

ORACLE®

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: OA Framework-Based Page

The screenshot shows a web application interface for 'Purchase Orders'. At the top, there's a blue header bar with the 'ORACLE® ToolBox' logo on the left and navigation links: Home, Logout, Preferences, and Diagnostics on the right. Below the header is a toolbar with buttons for Hello, World!, Home, Search / Drilldown, Transactions, and Sample Library. A main content area starts with a section titled 'Purchase Orders' containing some instruction text. Below it is a 'Search' region with a form for entering a Purchase Order number and a dropdown for 'Created' date. There's also a checkbox for 'Show my orders only' and a 'Go' button. Further down, another section of instruction text is followed by a table with columns for Number, Description, Employee Name, Created, Supplier, Currency, Order Total, and Details. The table currently shows 'No search conducted.' At the bottom of the page is a footer bar with links for Hello, World!, Home, Search / Drilldown, Transactions, Sample Library, Home, Logout, Preferences, and Diagnostics, along with links for About this Page and Privacy Statement. The footer also includes a copyright notice: Copyright(c) 2006, Oracle. All rights reserved.

View: Java Objects in a Page

View: Java Objects in a Page

- Each UI widget corresponds to one or more Java objects (beans)
- The Java objects are assembled declaratively with the Oracle 10g JDeveloper OA Extension tool
 - The Java beans are defined hierarchically
- The Java objects are used to create HTML at runtime

ORACLE

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

View: A Framework Example

Corporate and Product Branding
Named Children

Header Bean

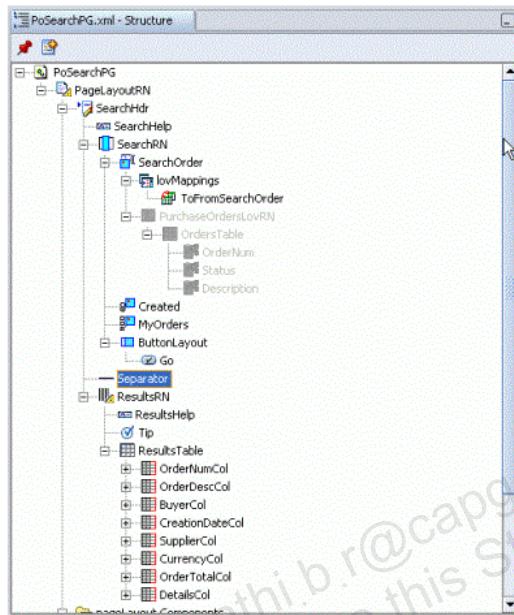
Submit Button
Bean

Table Bean

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

View: Page Hierarchy

View: Page Hierarchy



ORACLE®

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

Controller: Controlling UI Behavior

- Controller classes define how your Java Beans behave.
- You can override controller classes to:
 - Manipulate the UI at runtime
 - Manually initialize data items
 - Intercept and handle user events like button clicks
- Controller classes subclass `OACControllerImpl`.
- `OAPageBean` is the main OA Framework page processing class.
- There are methods in the Controller class to handle GET and POST requests.

ORACLE®

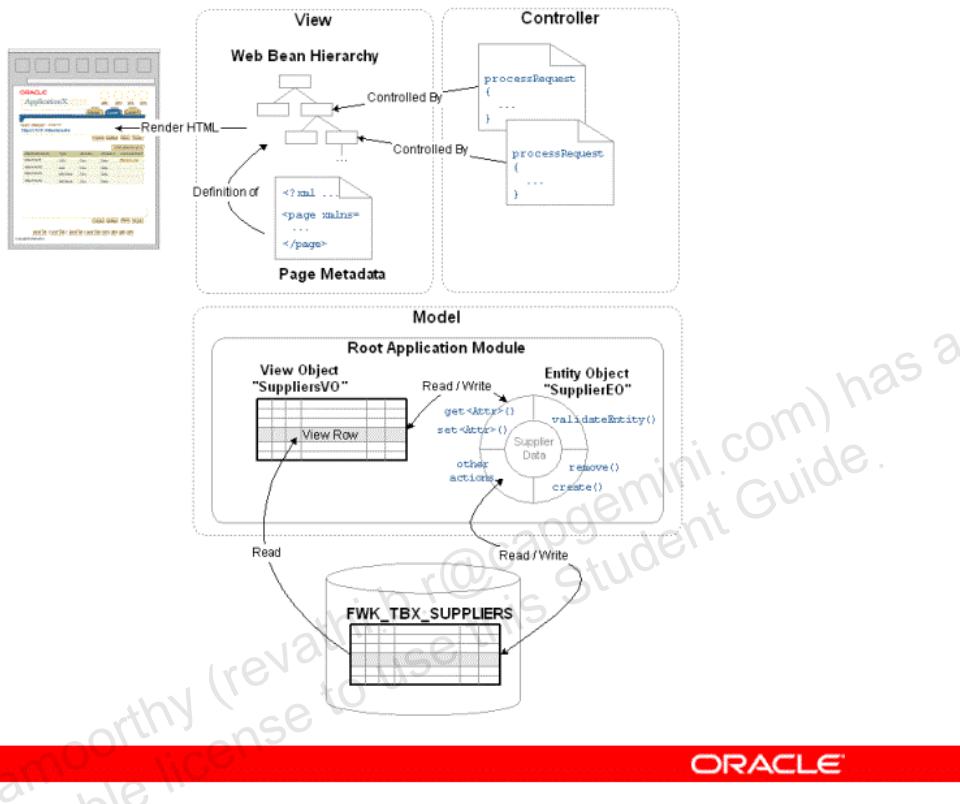
In general, a controller does two things.

Responds to the drawing of the page (GET).

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

OA Framework MVC Summary

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®

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

Basics of the Model

Chapter 3

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

R12.x Extend Oracle Applications: Building OA Framework Applications

R12.x Extend Oracle Applications: Building OA Framework Applications

Basics of the Model

ORACLE

Lesson Objectives

Lesson Objectives

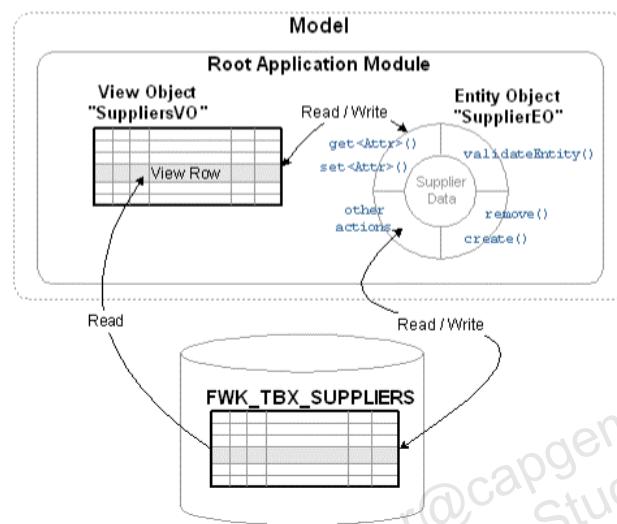
After completing this lesson, you should be able to:

- 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

Model-layer BC4J Objects

Model-layer BC4J Objects



ORACLE

Remember two things:

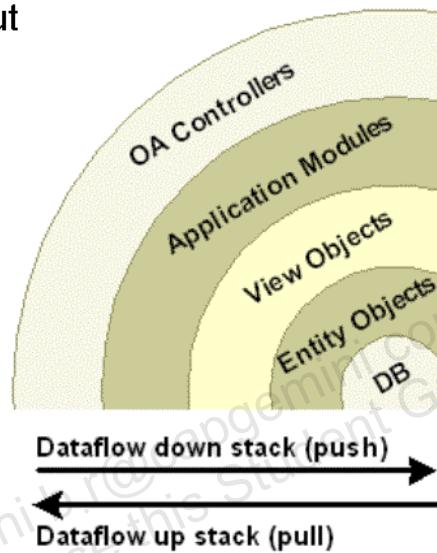
The Root AM is just the first AM for a page. It contains the transaction and all the connection objects.

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

Encapsulation The “Reuse Onion”

Encapsulation The “Reuse Onion”

- Each layer only “knows” about the layers below it.
- This encapsulation promotes easier reuse of components.



ORACLE®

'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

General Reuse Rules

- Model code should never reference controller code directly.
- Never reference/import any server-side implementation classes or interfaces on the client-side.
- If you need the server code to do work for you, always route your call through the root application module.
- Never include JDBC or other server-side processing directly in your client-side code.

ORACLE®

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

Recommended Build Approach

- 1. Create any business components packages that you need for your BC4J model objects.**
- 2. Implement declarative BC4J application modules, entity objects, view objects and others as needed for your page(s). Add view objects to your root application module(s) as appropriate.**
- 3. Create the menu definition for your application.**
- 4. Create the OA user interface components for your page(s).**
- 5. Create and implement controller code.**
- 6. Implement UI application module code supporting your pages.**
- 7. Implement entity object business logic.**

ORACLE®

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

Business Component (BC4J) Packages

Business Component (BC4J) Packages

- While not formally a BC4J component, every BC4J component must belong to a BC4J package.
- BC4J packages contain the following:
 - The naming/pathing structure for the BC4J components it stores.
 - The database connection associated to the package for your development environment.
 - Is mostly used for showing the relationship amongst related BC4J components by storing related components together, and then using a standardized naming structure to highlight the relationship.

ORACLE®

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

BC4J Package Naming Standards

BC4J packages also correspond to directory paths.

- EO-related (business logic) .java and .xml files in oracle.apps.<application shortname>. <module>.schema.server
- AM and VO (UI-related logic) .java and .xml files in oracle.apps.<application shortname>. <module>.server
- Poplist- and LOV-related VO and AM .java and .xml files in oracle.apps.<application shortname>. <module>.poplist.server and oracle.apps.<application shortname>. <module>.lov.server

ORACLE®

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

Application Modules

- Defines the logical data model and business methods needed to support an application task
- Handles transactions
- Interacts with clients

ORACLE®

Application Modules

Application Modules

- Encapsulates Server/Middle tier View Objects
 - Container for view objects and view links
 - View objects are defined by view instance names which are the names referenced by UI framework
 - Allows definition of master/detail view links
- Encapsulates Server/Middle tier controller-like logic
 - Initialize and perform view object query
 - Custom functions to process view objects
 - For example: copying data between VOs.
 - Custom functions to access entities.

ORACLE

AMs communicate with Controllers and VOs.

Application Modules

Application Modules

- Root AM holds BC4J Transaction object
 - Nested AMs reference the root AM Transaction object
- OA Framework groups pages together by Root AM
 - Pages with same Root AM name, and RetainAM=Y URL flag, are handled as one AM/Transaction instance shared for both pages
 - Useful for multi-page updates
 - Useful for sharing expensive queries across several pages

ORACLE®

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

Transaction Object

- Holds Database Connections
- Holds all Entities centrally, separate from Application Modules and View Objects
- Created automatically as part of the Framework
- Holds list of EO_s that require:
 - Validation
 - Posting to database
 - Commit notification

ORACLE®

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

Application Module Files

Application Module Files

Each AM should have:

- <YourName>AM.xml
 - Declarative information about AM
- <YourName>AMImpl.java
 - Add methods to invoke assign VO's initQuery methods if needed

ORACLE®

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

Entity Objects

- Map to a database table or other data source
- Each entity object instance represents a single row
- Contains attributes representing database columns
- Fundamental BC4J object through which all inserts/updates/deletes interact with the database

ORACLE®

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

Entity Objects

- Central point for business logic and validation related to a table
- Encapsulates attribute-level and entity-level validation logic
- Can contain custom business methods

ORACLE®

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

Entity Object Creation Standards

Entity Object Creation Standards

- Include all table columns in the EO.
- Base your EO's on the _ALL tables rather than on organization-restricted views.
- Always generate accessors (setters/getters).
 - Speeds up performance in attribute lookups because you can use named accessors

ORACLE®

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

Entity Object Automatic Features

Entity Object Automatic Features

- Validation, locking and posting order of children
 - Handled by Composite Associations
- Who Columns (Record History)
 - Set automatically during EO create() or doDML()
 - EO should have following attributes:
 - CreationDate
 - CreatedBy
 - LastUpdateDate
 - LastUpdatedBy
 - LastUpdateLogin

ORACLE®

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

Entity Object Files

Each EO should have:

- <YourName>EO.xml
 - Declarative information about EO
- <YourName>EOImpl.java (optional)
 - Add create(), remove(), validateEntity(), and setter methods.

ORACLE®

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

View Objects

- Represent a query result.
- Are used for joining, filtering, projecting, and sorting your business data.
- Can be based on any number of entity objects.
- Can also be constructed from a SQL statement.

ORACLE®

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

View Objects

View Object is designed to:

- Manage collection of data
 - Main collection interface into the database data
- Provide view-like shaping of data
 - Leverage SQL to join tables and corresponding EO

ORACLE®

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

View Objects

A View Object should:

- Always delegate to other objects such as the EO or PL/SQL for business logic
 - ensures better reuse of business logic
- Contain only the attributes required for a specific purpose
 - Do not select more attributes (columns) than required for a page/transaction UI
 - View objects should be considered specific to a particular UI and are not expected to be reused

ORACLE®

You should never put business logic in your view object.

View Object Creation Methods

View Object Creation Methods

View Object can be created in one of three ways, based on:

- Generated SQL based on EO^s
 - Add ‘where clause’ at design-time, not run-time for best performance
- ‘Expert Mode’ custom SQL with no underlying EO^s
 - Read-only Data
- ‘Expert Mode’ custom SQL manually mapped to EO^s

ORACLE

ExpertMode View Objects

ExpertMode View Objects

Use ExpertMode custom SQL, manually mapped to EO^s, for:

- Unions
- Complex SQL
- SQL is encapsulated within an inner view to support additional View Link bindings: 'select * from (your-sql-here)'

ORACLE

View Objects with Entity Objects

View Objects with Entity Objects

- VO can be mapped to multiple
 - Updateable EO_s, though often just the primary EO
 - Reference EO_s, used for additional ‘joined’ data including description lookups

ORACLE

EO_s are the objects through which we insert, update, and delete data. VO_s 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 EO_s allows the actions on the EO to occur through the VO. VO_s mapped to reference EO_s are using those EO_s just for query purposes, generally a foreign key of some sort.

View Object Rows

View Object Rows

- A view object controls many view rows.
- Each view row represents a row of the data that is queried from the database.
- An entity object gets instantiated for each view row.

ORACLE®

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 your 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.

Creating View Objects

Creating View Objects

- When you want aliases in the query, make sure that the attribute settings in the VO wizard include the alias names.
- When you create or modify an "Expert mode" VO using the View Object wizard, be careful to make sure that the attribute mappings match the expressions in the query.

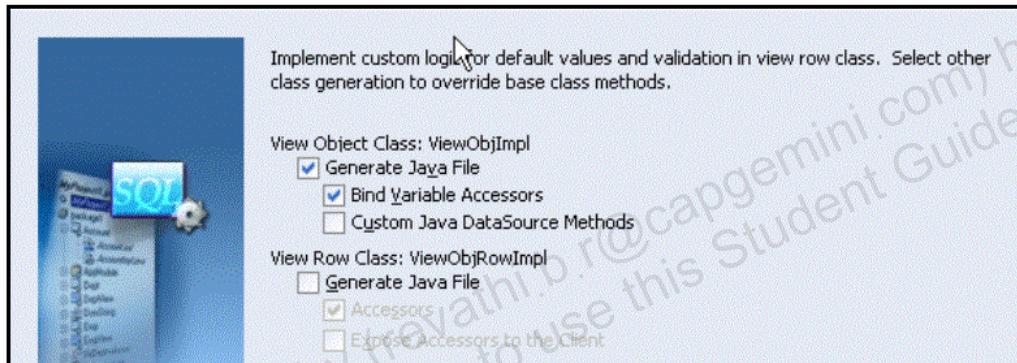
ORACLE®

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

View Object Java Files

- Create VO Java class (VOImpl) if needed
 - VOs should not contain business or validation logic, except transient attribute logic
- Always create View Row Java class (ViewRowImpl) and accessors (setters/getters)



ORACLE®

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

View Object Files

Each VO should have:

- <YourName>VO.xml
 - Declarative information about VO
- <YourName>VOImpl.java (optional)
 - Add initQuery method if needed
- <YourName>ViewRowImpl.java (required)
 - Contains accessors (setters and getters) for VO attributes
 - Behind the scenes, these accessors call corresponding methods in your EOImpl.java

ORACLE®

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

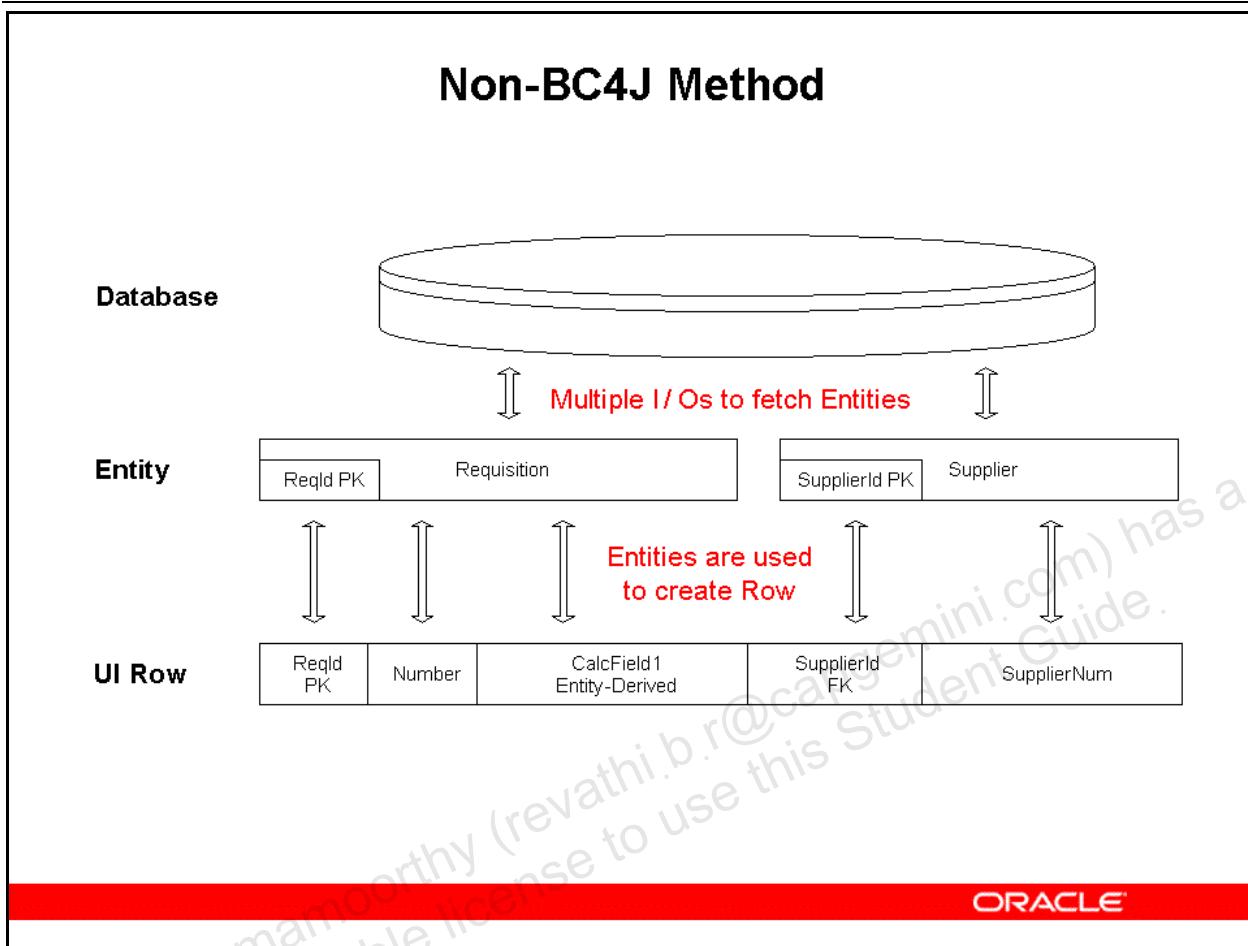
BC4J Database Interactions

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

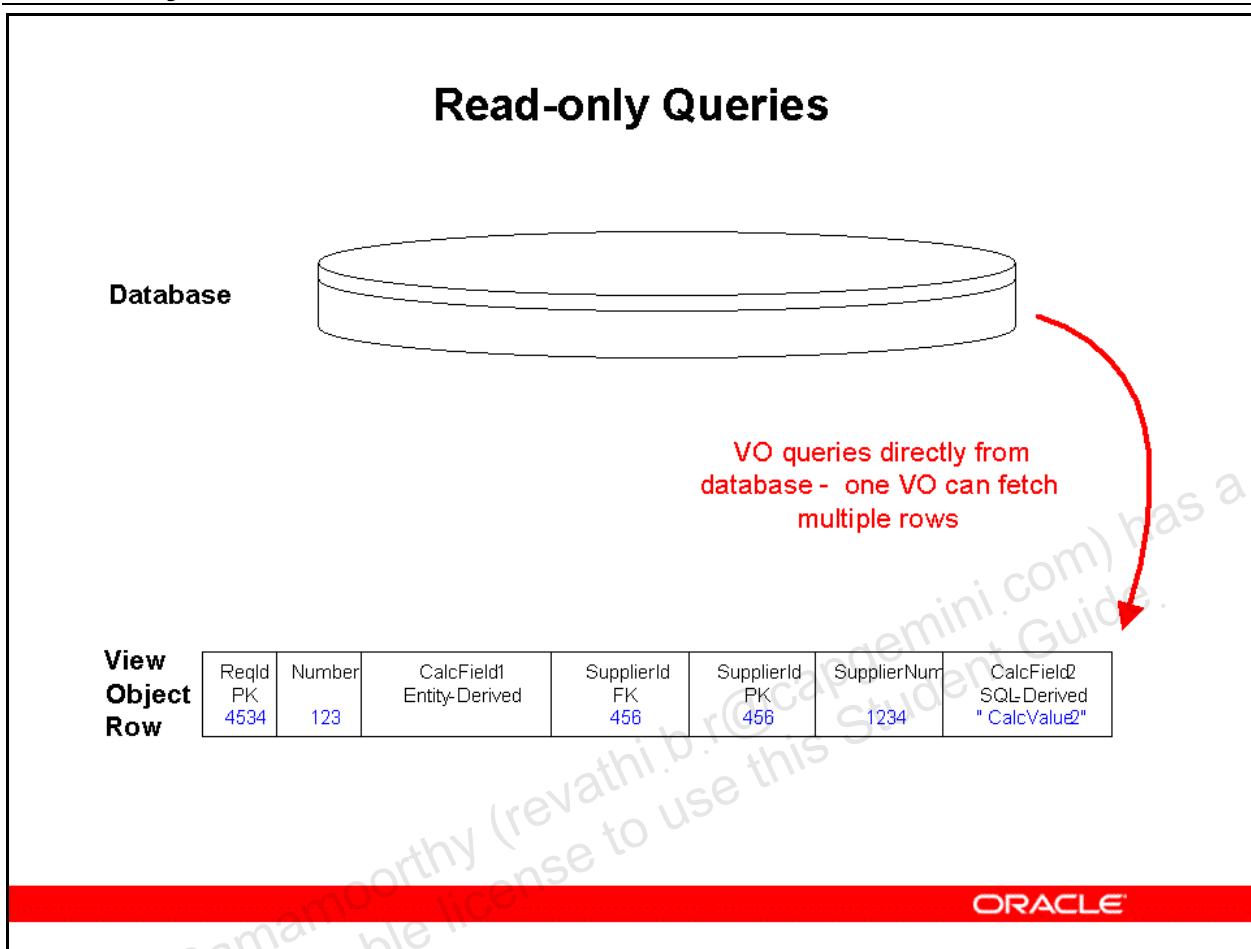
ORACLE®

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

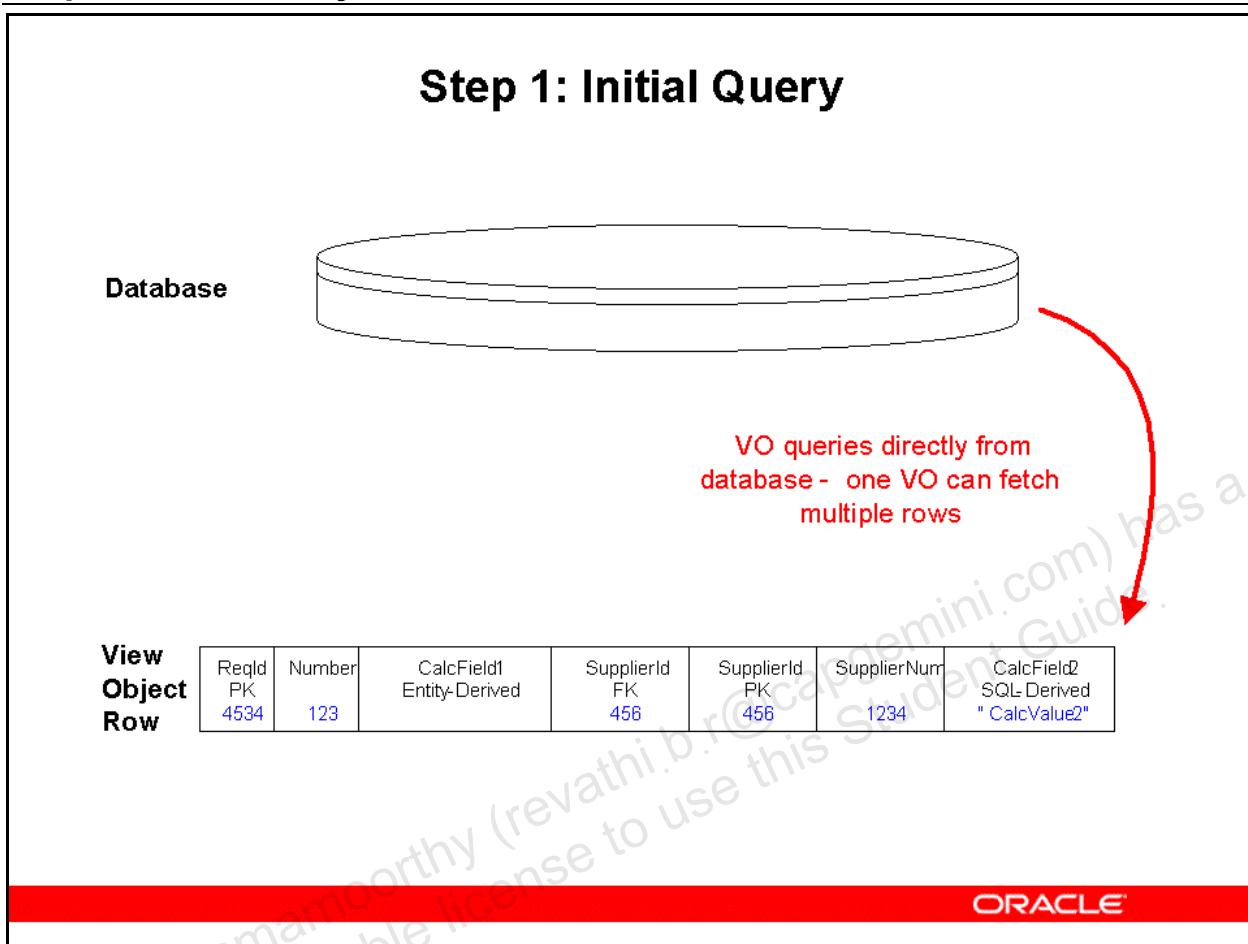


Entity objects are not required for read-only queries. So, BC4J breaks the Entity encapsulation for View Objects. This is done for significant performance improvement reasons.

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

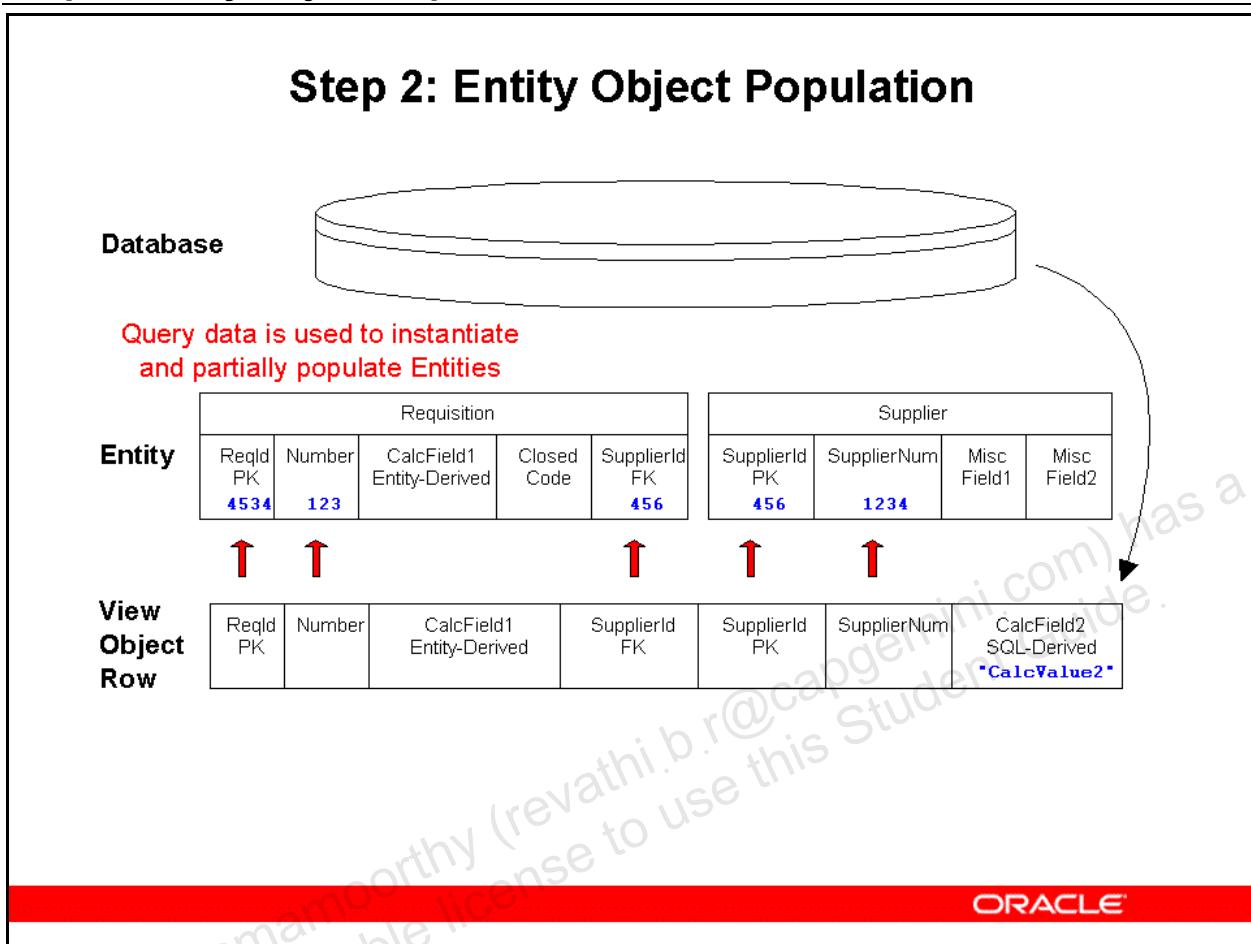
Note: The difference between a read-only query and an initial query is nothing at this point. Underlying the process is the definition of the VO. If the VO has no associated EO, it is read-only. If the VO has been defined with an associated EO, then the first step is an initial query.

Step 1: Initial Query



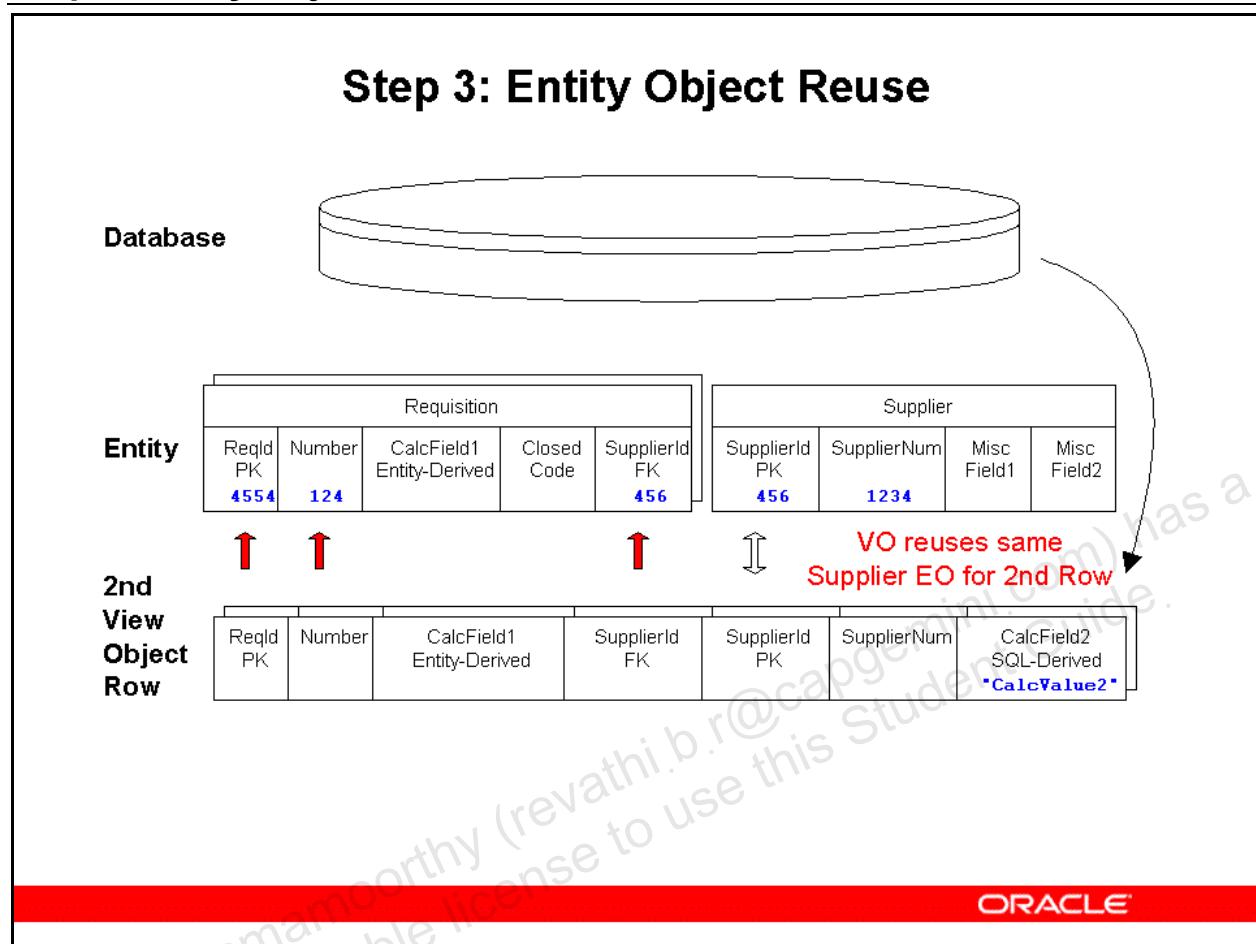
Note: The difference between a read-only query and an initial query is nothing at this point. Underlying the process is the definition of the VO. If the VO has no associated EO, it is read-only. If the VO has been defined with an associated EO, then the first step is an initial query.

Step 2: Entity Object Population



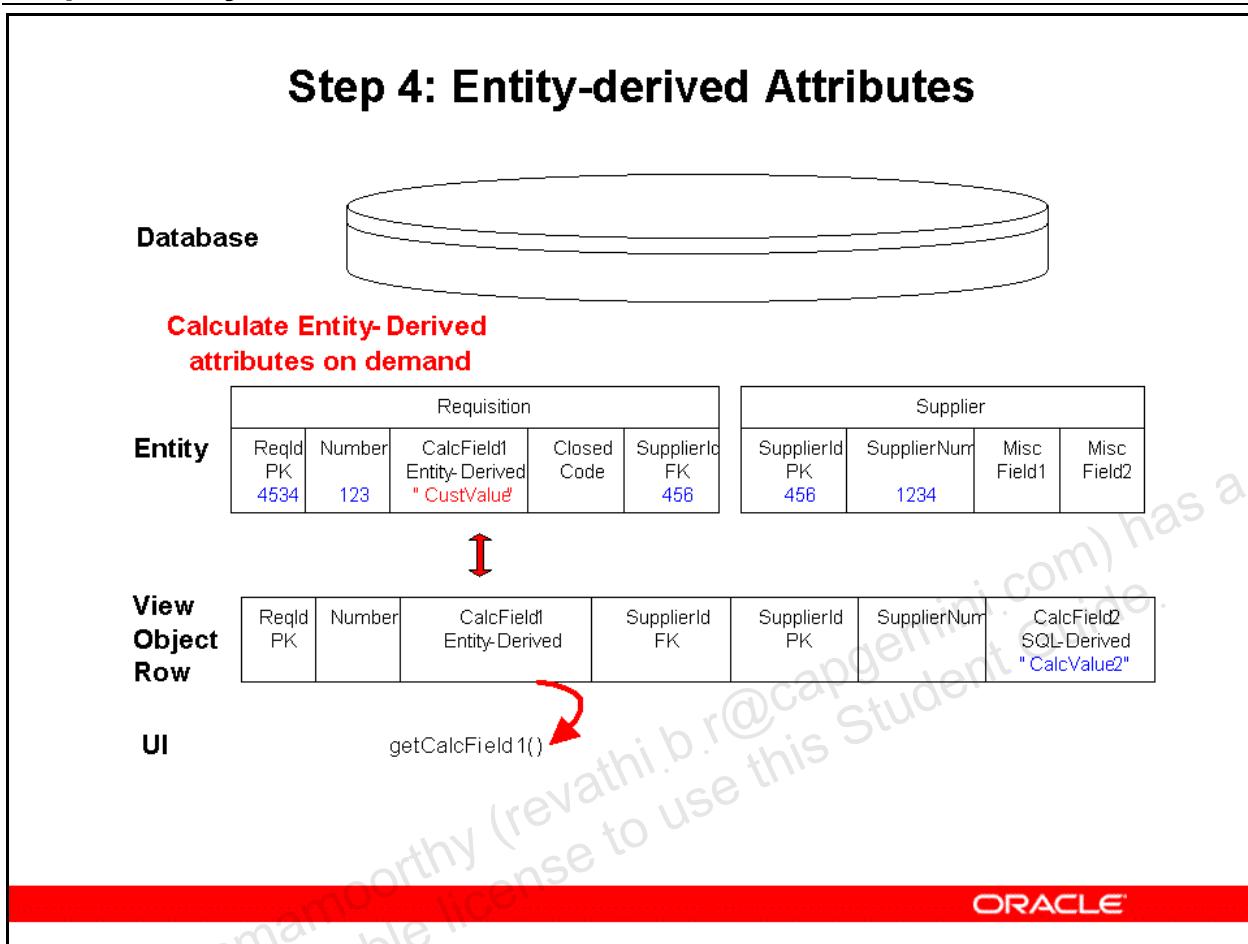
The query can bring back many view object rows (the result set or row set) in a single fetch from the database. Each view object row corresponds to an entity object (entity object instance). If the view object is based on an entity object, entity objects are found or instantiated by primary key (PK) 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



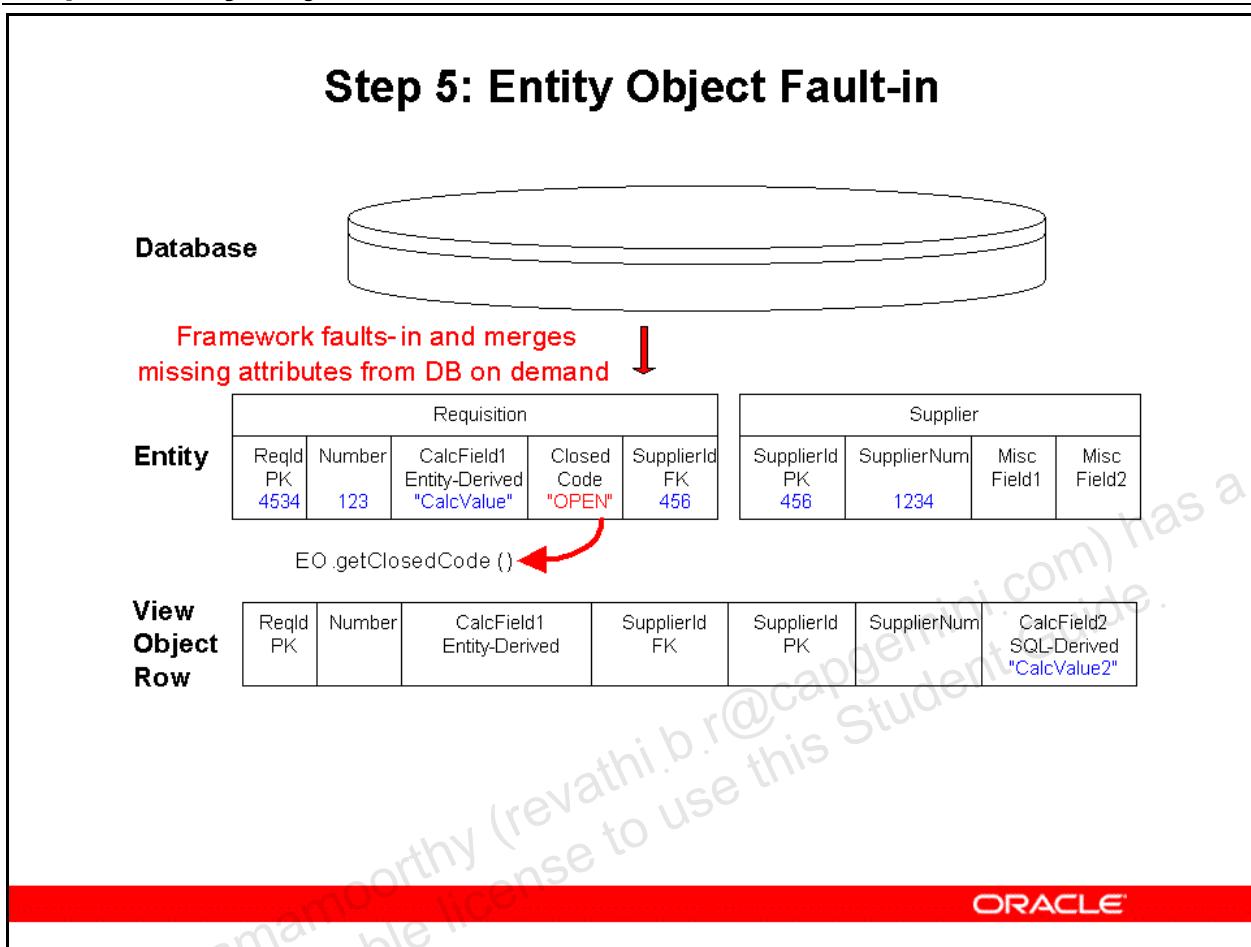
The 2nd VO Row uses less memory since it references and reuses the same Supplier EO.

Step 4: Entity-derived Attributes



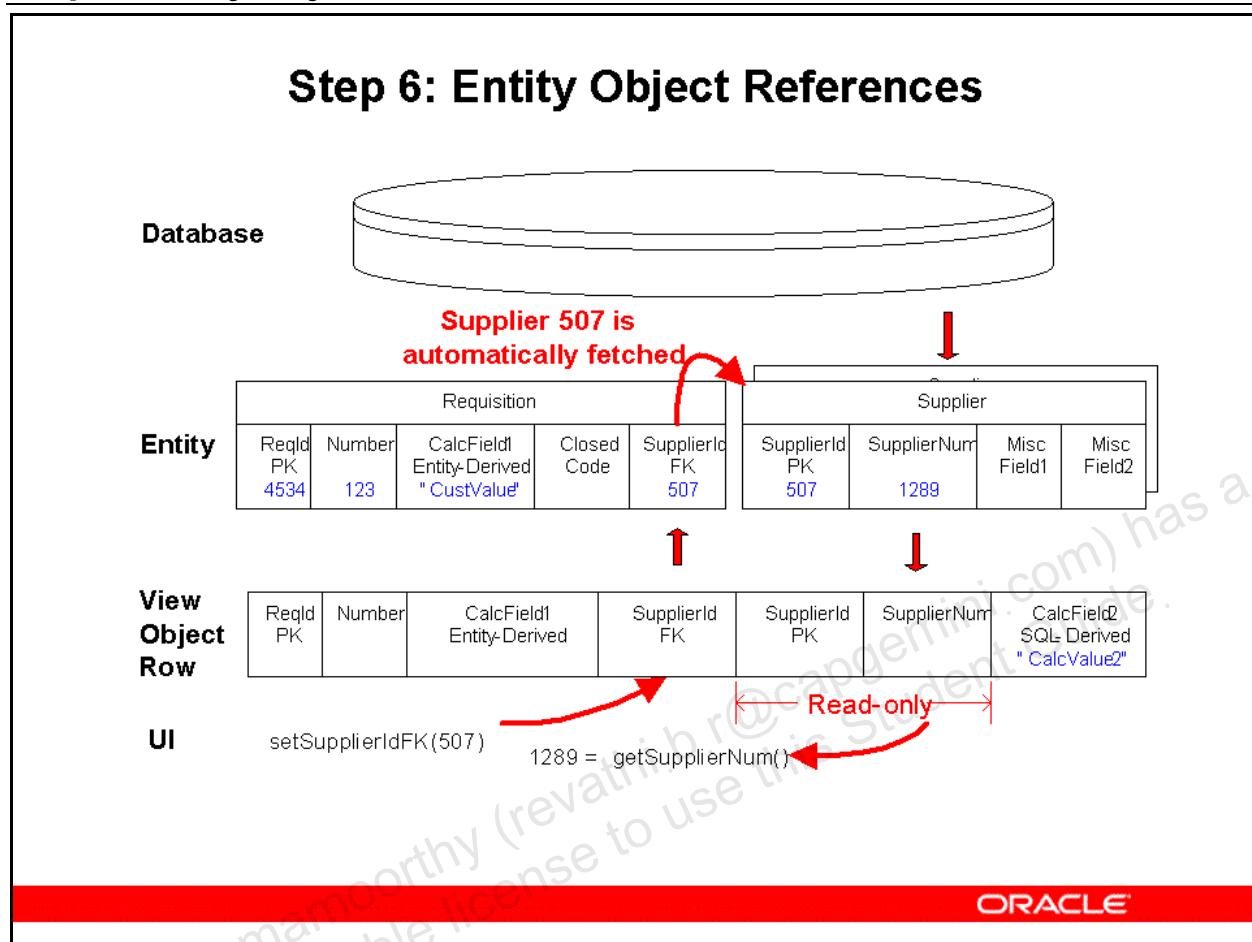
Entity-Derived attributes are useful for calculations owned by the Entity and shared by multiple VOs. But, entity-derived attributes are only calculated when demanded. When is that? When the VO attempts to bind the entity-derived attribute into the UI for rendering to the browser.

Step 5: Entity Object Fault-in



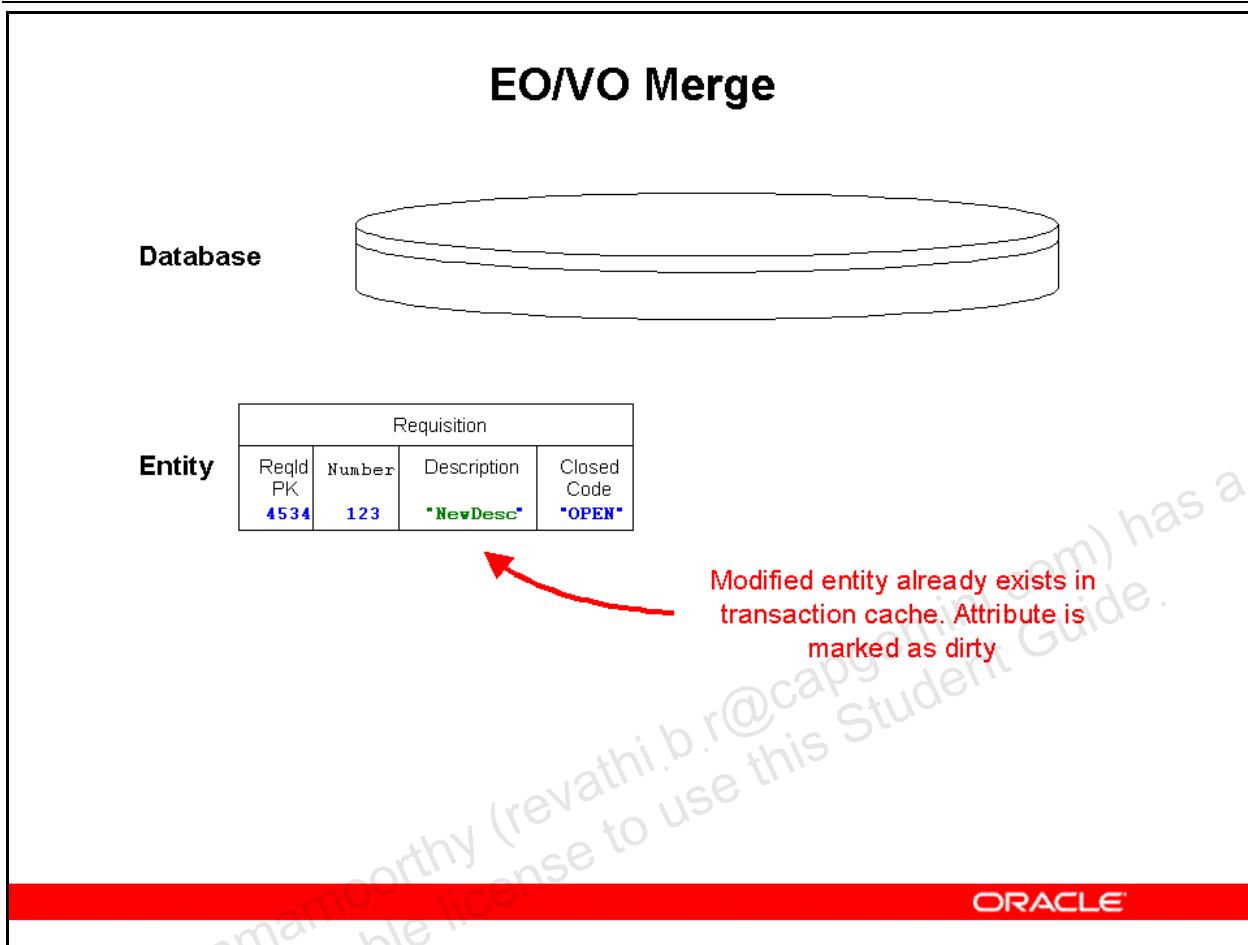
Missing attributes are faulted-in on demand during EO validation.

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

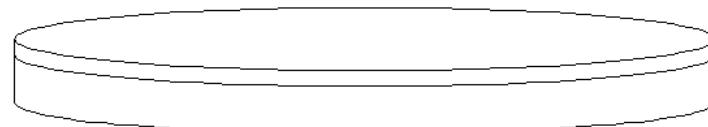


An EO/VO merge occurs whenever an EO already exists in memory, containing original or modified values, and a new VO is queried containing data for the EO. The EO is the transaction cache, and the values need to be resolved before the VO can properly bind the correct value to the UI.

Step 1: EO/VO Merge Resolution

Step 1: EO/VO Merge Resolution

Database



Entity

Requisition					Supplier			
ReqId PK	Number	Description	Closed Code	SupplierId FK	SupplierId PK	SupplierNum	Misc Field1	
4534	123	"NewDesc"	"OPEN"	456	456	1234		

Query brings in older values that don't
match new EO attributes

View
Object
Row

ReqId PK	Number	Description	Closed Code	SupplierId FK	SupplierId PK	SupplierNum
4534	123	"OldDesc"	"OPEN"	456	456	1234

ORACLE®

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

Step 2: EO/VO Merge Resolution



Original Attribute values are not merged with EO

Entity	Requisition					Supplier			
	ReqId PK 4534	Number 123	Description "NewDesc"	Closed Code "OPEN"	SupplierId FK 456	SupplierId PK 456	SupplierNum 1234	Misc Field1	Misc Field2
	↔	↔	↔	↔	↑	↑	↑		

View
Object
Row

ReqId PK	Number	Description	Closed Code	SupplierId FK	SupplierId PK	SupplierNum
		"OldDesc"				

~~"OldDesc"~~ VO will display new EO values

ORACLE®

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

Other Model-layer Objects

Other Model-layer Objects

- Association Objects (AOs)
- View Links (VLs)
- Entity Experts
- Validation Application Modules (VAMs)
- Validation View Objects (VVOs)

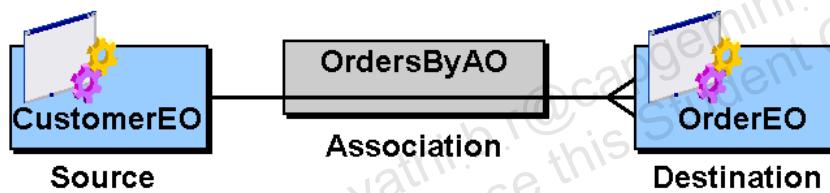
ORACLE®

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

- Define a relationship between entity objects.
- Facilitate access to data in related entity objects
- May be based on database constraints
- May be independent of database constraints
- Consist of a source (master) and a destination (detail) entity



ORACLE®

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

Association Objects

Association Objects

- Two types of Association Objects
 - Reference
 - Composition
- Both types can be referenced via 'association attributes' on Entity.

ORACLE®

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.

Reference Association Objects

Reference Association Objects

Reference Association

- Used for weak associations between Entities, such as foreign keys for lookups
- No special runtime behavior
- Example: Requisition - Supplier association

ORACLE®

This is the default behavior for AOs.

Composition Association Objects

Composition Association Objects

Composition Association

- Used for composite objects with strong "owning" relationship
 - Use if: child entity cannot exist without parent; child is deleted when parent is deleted
 - Example: RequisitionHeader - RequisitionLine association
- Create by checking the Composition check box in the BC4J association wizard

ORACLE®

This has to be specifically indicated in the EO.

Composition Association Object Behavior

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®

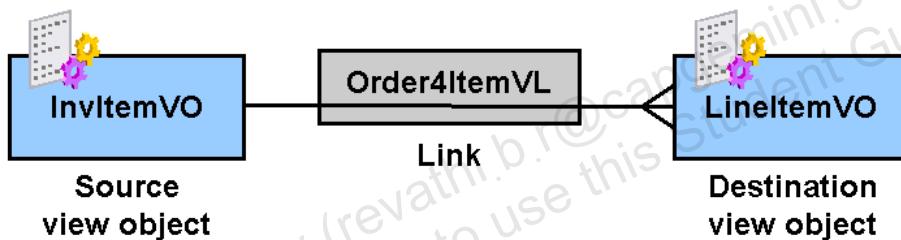
View Links

View Links

A view link is an active link between view objects.

You can create view links by providing the following:

- Source and destination views
- Source and destination attributes



ORACLE

Note: While it is possible to create View Links (VLs), they are seldom-used objects. Why? If your 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 your VO on that database view.

View Links

View Links

Use a View Link to create a Master-Detail relationship between view objects.

- Allows dynamic synchronization between parent and child VO
- Child rowset is refreshed when the current parent row changes

ORACLE®

View Links

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

Entity Experts

Entity Experts

Simple common code or, more commonly, validation routines that can be called by other EO's that avoids the overhead of instantiating an entire EO.

ORACLE®

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

Validation AMs and Validation VOs

- Validation VOs (VVOs): Allows simple SQL statements to be executed at the entity-layer. Most commonly, these are SELECT statements done to validate data.
- Validation AMs (VAMs): VOs must be contained within AMs. VAMs are the containers for VVOs.

ORACLE®

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

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®

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

Basics of the View

Chapter 4

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

R12.x Extend Oracle Applications: Building OA Framework Applications

R12.x Extend Oracle Applications: Building OA Framework Applications

Basics of the View

ORACLE

Lesson Objectives

Lesson Objectives

After completing this lesson, you should be able 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

Recommended Build Approach

Recommended Build Approach

1. Create any business components packages that you need for your BC4J model objects.
2. Implement declarative BC4J application modules, entity objects, view objects and others as needed for your page(s). Add view objects to your root application module(s) as appropriate.
3. Create the menu definition for your application.
4. **Create the OA user interface components for your page(s).**
5. Create and implement controller code.
6. Implement UI application module code supporting your pages.
7. Implement entity object business logic.

ORACLE

Steps 4 deals directly with View layer components.

View-layer Terminology

View-layer Terminology

- Property – anything shown in the property inspector.
- Attribute – an XML attribute in the UIX file. Roughly equivalent to a column in a View Object.
- BLAF – Oracle's Browser Look-and-Feel UI guidelines (available at: <http://otn.oracle.com/tech/blaf/>)
- Page – the basic UI component created in OA Framework, and reference by E-Business Suite function security.
- Region – a portion of a page that can either be unique to the page, or reused (extended) from a previously build region.
- Attribute set – a named set of properties applied to configure an item.

ORACLE®

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

View-layer Terminology

- CSS Class – the cascading style sheet class applied to the item (see BLAF and XSS).
- View Instance – the VO used for data binding
- View Attribute – the VO's specific attribute to which the item is bound
- Admin Personalization – indicates whether the item or region is system administrator personalizable
- User Personalization – indicates whether the query region is user level personalizable
- SPEL – Simplest Possible Expression Language

ORACLE®

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

While not specifically a View-layer component, workspaces and projects are essential components of OA Framework development. Workspaces and Projects are containers that hold objects together for development purposes. You do not deploy workspaces and projects to a running E-Business Suite instance.

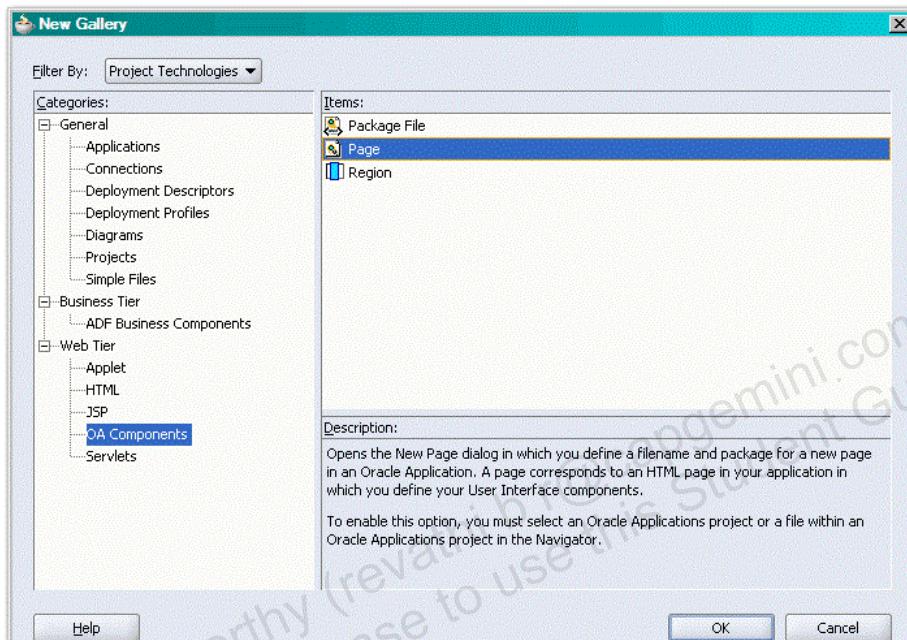
However, be careful to choose “Workspace Configured for Oracle Applications” and “Project Configured for Oracle Applications” rather than generic workspaces and projects. Only OA projects allow you to create pages, regions, and attribute sets. And, only OA projects capture E-Business Suite related information such as run options and OA Framework connection information.

ORACLE®

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

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



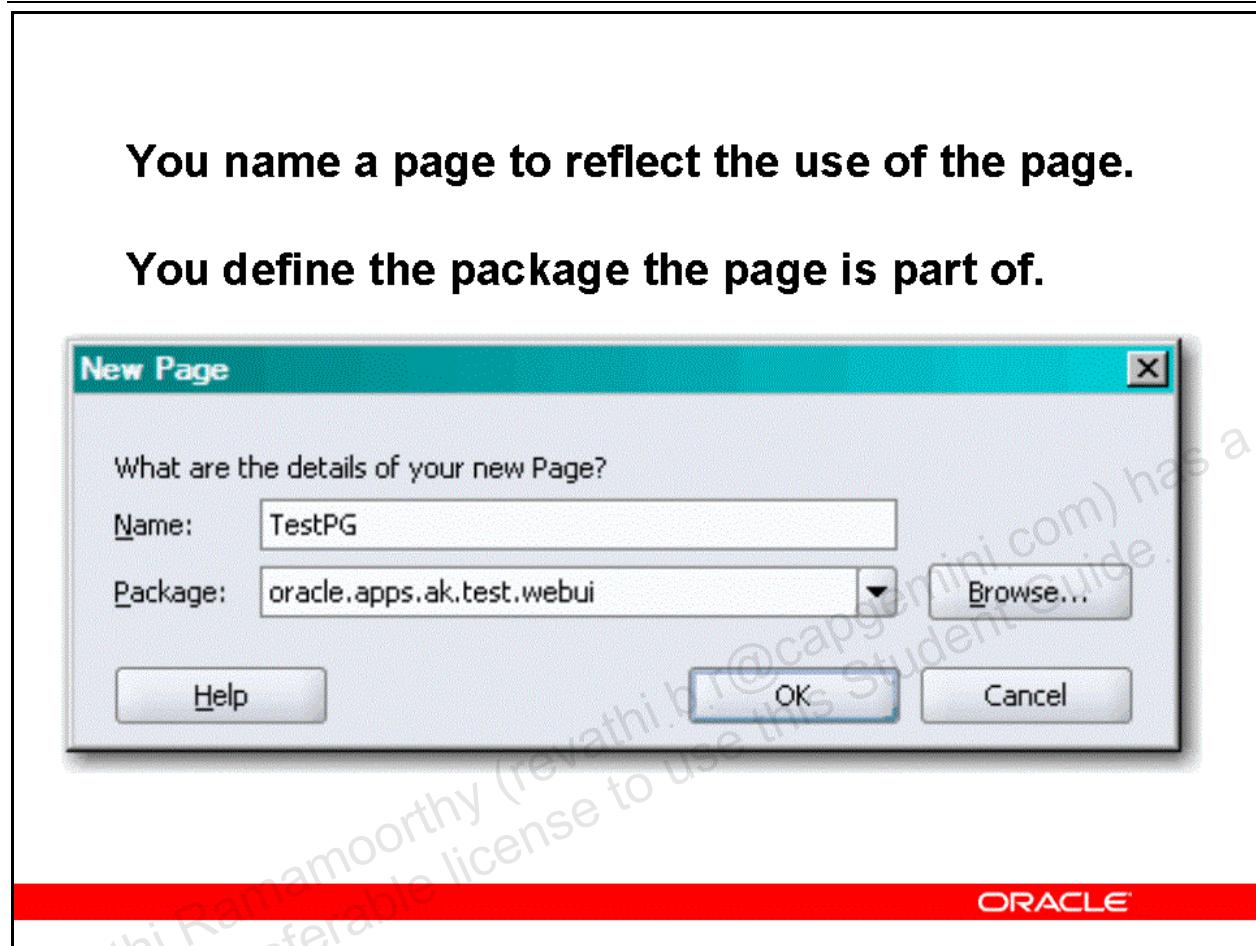
ORACLE

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.

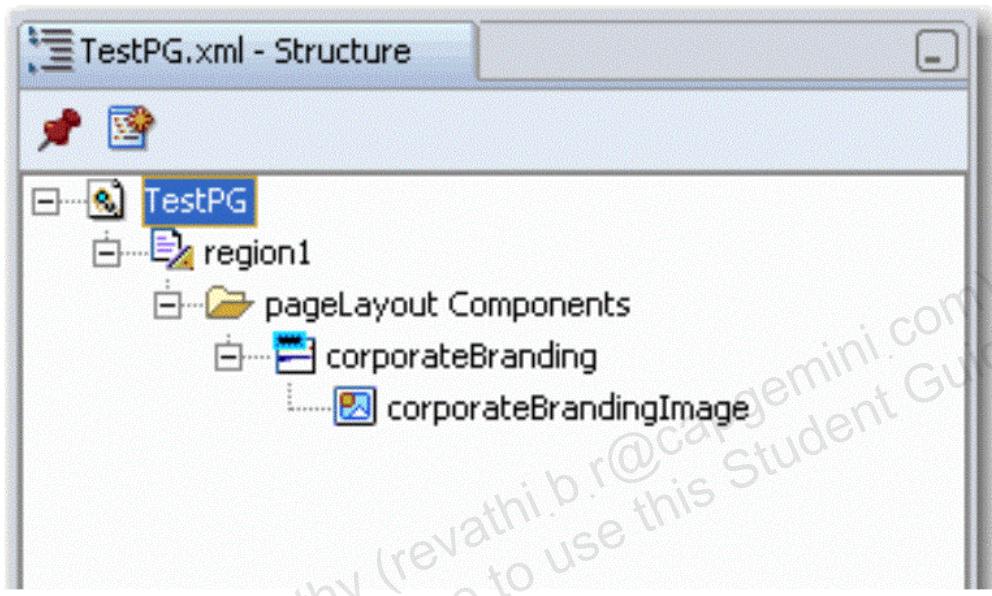
You name a page to reflect the use of the page. You define the package the page is part of.



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

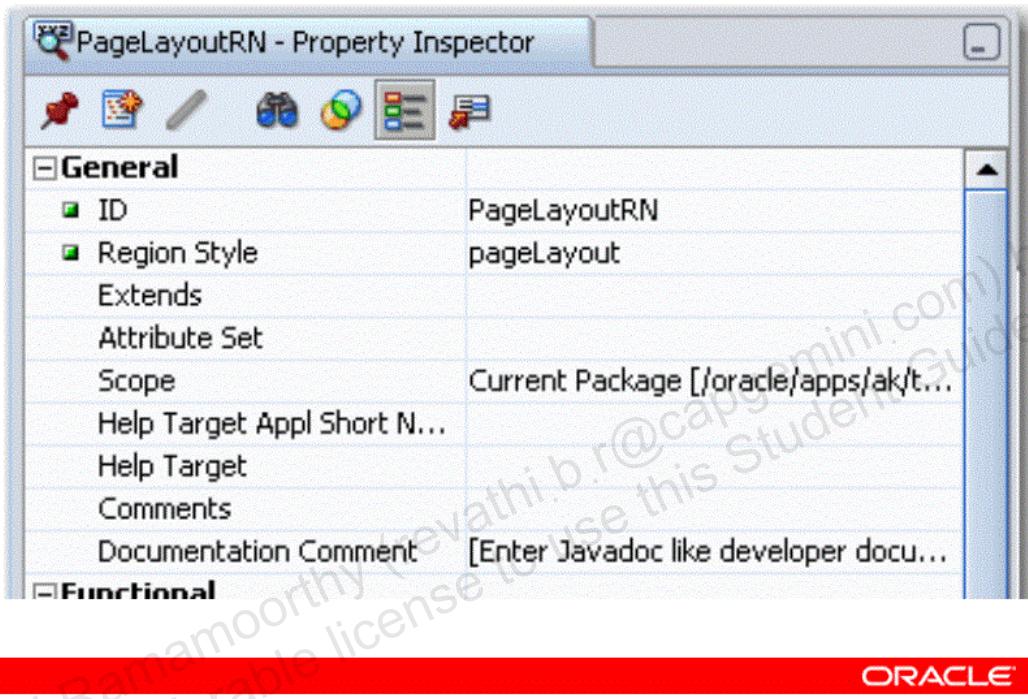
A page will have a structure that will display the View attributes that reside in a page hierarchy



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.

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.

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?

What Can You Add to the Page?

You can add the following to an OA Framework page:

- Regions
- Items
- Shared Regions

You should also know the difference between Named and Indexed children.

ORACLE®

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

Region Styles

Region Styles

The region style in the Property Inspector determines the UI Bean for the region. Different Beans have very different behaviors. The region styles available are:

- advancedSearch
- advancedTable
- bulletedList
- cellFormat
- contentContainer
- contentFooter
- defaultDoubleColumn
- defaultFormStack
- defaultSingleColumn
- defaultStack
- flexibleLayout
- gantt
- graphTable
- hGrid
- header
- hideShow
- hideShowHeader
- labeledFieldLayout
- navigationBar
- pageButtonBar
- query
- rowLayout
- Shuttle
- stackLayout
- subTabLayout
- switcher
- table
- tableLayout
- train
- tree

ORACLE®

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

Sub-Region Styles

Some styles need to be combined to build a particular layout on your page, such as an arrangement of fields and buttons that use a tableLayout region to define their locations.

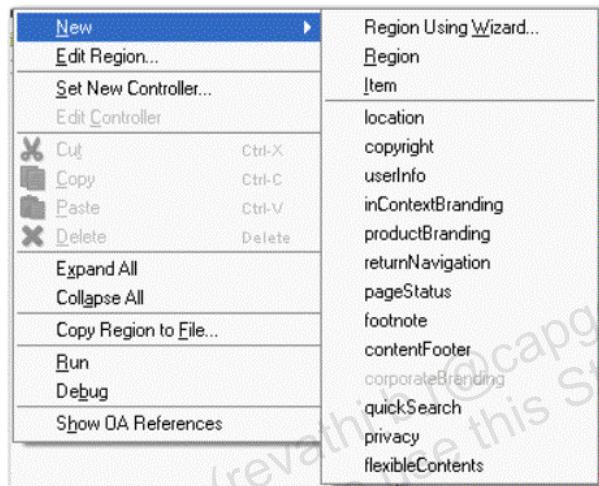
- tableLayout
- rowLayout
- cellFormat
- spacer (this is an item type, not a region type)

ORACLE®

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

Item Styles

The item style in the Property Inspector determines the UI Bean for the item. Different Beans have very different behaviors. The item styles available are:

- attachmentLink
- attachmentTable
- button
- exportButton
- flex
- formParameter
- formValue
- formattedText
- image
- link
- rawText
- resetButton
- richTextEditor
- separator
- servletInclude
- spacer
- staticStyledText
- submitButton
- tip
- urlInclude
- messageCheckBox
- messageRichTextEditor
- messageInLineAttachment
- messageDownload
- messageFileUpload
- messageLovChoice
- messageLovInput
- messageRadioButton
- messageRadioGroup
- messageStyledText
- messageTextInput
- messageChoice

ORACLE®

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

Item Style Details

- Item styles that start with message (such as messageTextInput, messageStyledText, messageLovInput, and so on) create displayed data fields.
 - Use messageTextInput for an input field.
 - Use messageLovInput for an input field with an LOV.
 - Use messageStyledText for a display-only field.
- The region style messageComponentLayout can be used to quickly and correctly position message* beans and related items into an n-column grid.

ORACLE®

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
```

Item Style Details

Item Style Details

- Use staticStyledText for plain boilerplate text on a page such as an instruction.
- Use submitButton for a button that does a POST (the button item style does not submit).

ORACLE®

Shared Regions

Shared Regions

If you want to create a shared region, you must comply with the following standards.

- The top-level (shared) region must be saved in its own XML file.
- Shared region can accept values from a using region.
- Shared region must be implemented to fail gracefully. For example, if appropriate parameters are not passed from a using region, the shared region should set acceptable defaults or raise a meaningful (and documented) exception.

ORACLE

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

- If the region scope is set to Public:
 - The top-level region must have its own application module. The application module should include only those view objects that are relevant for the shared region.
 - The top-level region must have its own controller. You may associate additional controllers with subregions as necessary.
- The shared region must be fully documented.

ORACLE®

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

Attribute Sets

Attribute sets are named, reusable collections of properties (prompt, maximum display length, data type and so on as appropriate for the attribute set type) that can be used by any type of OA component, including regions, items, and other attribute sets. They are designed to facilitate the reuse of these components throughout E-Business Suite, which yields a significant cost savings to both Oracle and its customers:

- Oracle saves in translation and maintenance costs.
- Customers can make quick, global personalizations to their applications. Additionally, fewer UI elements translates to less middle-tier memory consumption, and ultimately, this means better performance and scalability.

ORACLE®

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

Creating Attribute Sets

The steps to create an attribute set are as follows:

1. Create the attribute set package.
2. Create the attribute set.
 - a. Identify the attribute set template to use.
 - b. Create the attribute set naming it appropriately.
 - c. Enter values for the attribute set's included properties.
3. Save your work.
4. Deploy your attribute set.

Note: On Linux, there is an automated attribute set generator called, attributesets.

ORACLE®

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

Example Attribute Set

```
<?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 -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 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>
```



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 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>
```

```
<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

Table Column Template

The following is a list of the properties that may be used to define table column attribute sets.

- Additional Text
- Comment
- Data Type
- Document Name
- Height
- Length
- Maximum Length
- Prompt
- Required
- Tip Message Appl Short Name
- Tip Message Name
- Tip Type

ORACLE®

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

Button Template

The following is a list of the properties that may be used to define button attribute sets.

- Additional Text
- Comments
- Document Name
- Prompt

ORACLE®

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

Region Header Template

The following is a list of the properties that may be used to define region header attribute sets.

- Additional Text
- Comments
- Document Name
- Prompt
- Icon URI
- Maximum Length

ORACLE®

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

CSS Styles

Apply CSS (cascading style sheet) styles to regions and items to provide a consistent look and feel

- Customers can change styles easily
- Master list of styles at BLAF UI website:
<http://www.oracle.com/technology/tech/blaf/specs/index.html>
- Many of these are already built into UIX and do not need to be applied manually.

ORACLE®

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

Common CSS/XSS Styles

You must set style sheet tags for many of the UI Beans, particularly fields and field-like Beans (poplists, checkboxes, static page text, and so on).

- OraDataText:
 - bold, left-aligned display-only field data
- OraFieldText:
 - plain text in updatable fields, checkboxes, etc.
 - The text displayed for a checkbox is not considered to be a prompt, so do not use OraPromptText

ORACLE®

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

Common CSS/XSS Styles

Common CSS/XSS Styles

- OraPromptText:
 - right-aligned prompts
 - Note that any of the "message" Beans, such as messageTextInput, automatically set the correct CSS class for the prompt.
- OraLinkText:
 - left-aligned plain text data that should render as a link in the link color.
- OralInstructionText:
 - any of the left-aligned plain text instructions that you add to your page

ORACLE®

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

Extending Other Objects

Extending Other Objects

Some parts of your page can be inherited by extending other regions or items using the Extends property.

- For example, extend /oracle/apps/fnd/framework/webui/OAReqFieldDescRG region to get a Required Key region
- Extend entire regions (and their logic) that are common to several pages.
- Extend single objects.

ORACLE®

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

Destinations and Links

The Destination URI property lets you link from a field.

- For a link to another page:

```
OA.jsp?[page=<URL path> or  
OAFunc=<FunctionName>]  
&<paramName1>{@ViewAttributeName1}  
...  
&retainAM=<Y or N>  
&addBreadCrumb=<Y or N>
```

ORACLE

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.

Mailto Links

Mailto Links

The Destination URI property lets you link from a field.

- For an email link, set Destination URI property to

`mailto:{@<EmailAddressViewAttributeName>}`

— For example: `mailto:{@ManagerEmail}`

ORACLE®

Lists of Values (LOVs)

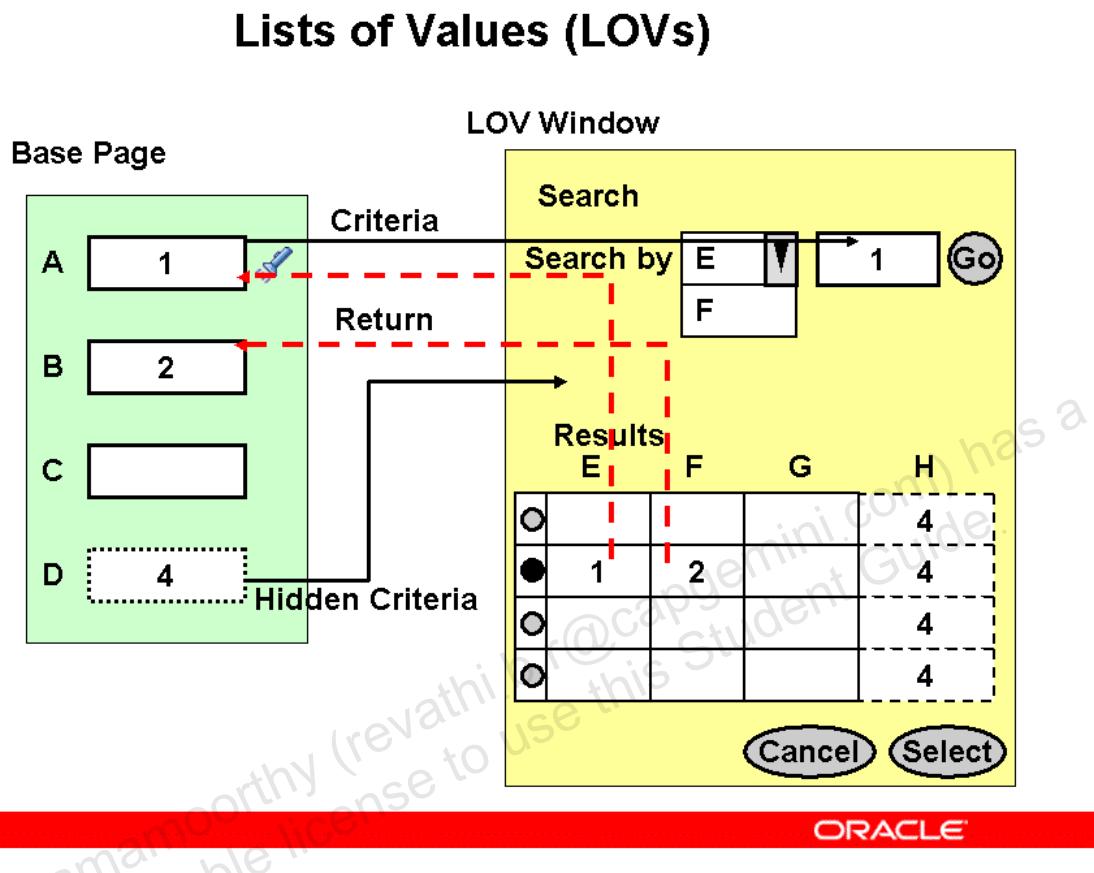
Lists of Values (LOVs)

- Two types of declarative LOV
 - In-line LOV region
 - External LOV region
- Dynamic query construction
 - Add to either in-line or external LOV
 - Define "Programmatic Criteria" in LOV mappings
 - Requires an extra controller

ORACLE

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

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

Defining an External LOV

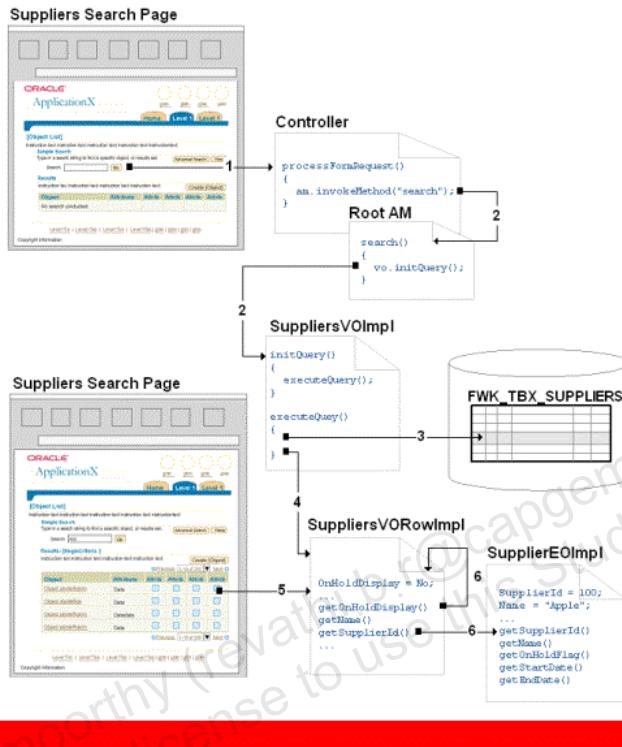
Create the LOV mappings:

- 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

Reading Model Data



ORACLE

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.

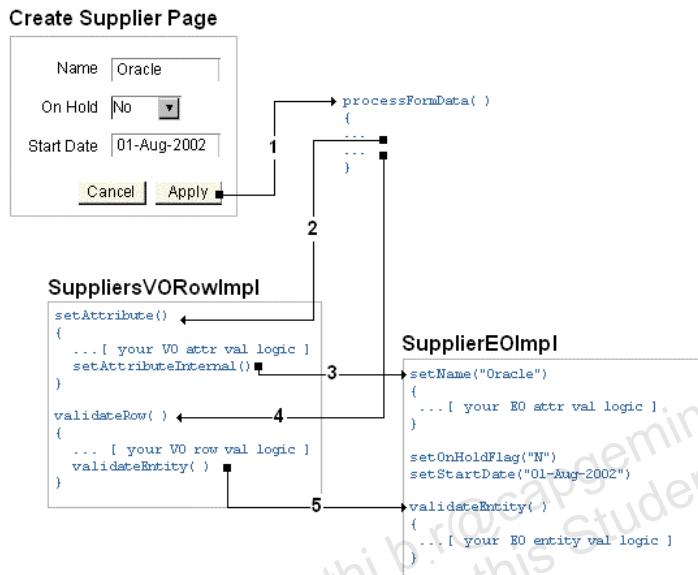
4. For each row returned in our example result set, the view object instantiates a SupplierEOImpl entity object and sets its attribute values with the query results.

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

Writing Model Data



ORACLE®

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.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

Binding Items to Data

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

General Naming Rules

- File Name length
 - File names are limited to 30.3 characters for OA Extension XML files (50.java for Java files).
- Object Name length (regions, items, and so on)
 - For performance reasons, object names (internal ID's) are limited to 30 characters.
 - Use the shortest possible names that are readable.
 - Consider abbreviating repeating object names.

ORACLE

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

General Naming Rules

General Naming Rules

Object name abbreviations:

- Common abbreviations are acceptable to keep names as short as possible.
- Acceptable abbreviations can be instantly understood by a third party consultant.
- Examples of acceptable abbreviations are:
 - Emp for employee
 - Num for number
 - Desc for description
 - Industry terms such as PO
- If in doubt don't abbreviate.

ORACLE®

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

General Naming Rules

Object names in pages must be unique in the entire page.

- Convention is to name the object to match its label (shortened as appropriate).
- For duplicate names, prefix the second and higher instances with a short version of the parent object's name. For example:
 - EmpName
 - SearchEmpName
 - ResultsEmpName
- Most names follow Java naming convention (mixed case)

ORACLE®

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

Page and Object Naming Rules

Page and Object Naming Rules

- Page name ends in PG: EmpSearchPG, EmployeePG, SupplierPG
- Content regions have names ending with RN: MainRN, EmpSearchRN, PoHeaderRN
- Layout-only container region names reflect the layout: ContactsTableLayout, ButtonsRow
- BLAF Navigation-only regions use standard names: PageTopNavLinksRN, TrainRN

ORACLE®

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

Styles and Bean Names

- Region or item style names in the JDeveloper Property Inspector appear in the Java naming convention starting with lower case letter.
- The corresponding programmatic versions (web Bean names) usually prepend "OA", append "Bean" and capitalize the first letter.
 - messageComponentLayout **becomes** OAMessageComponentLayoutBean
 - table **becomes** OATableBean
- See the OA Framework Developer's Guide and OA Framework Javadoc

ORACLE®

Exceptions to the rule: attachmentLink becomes OAMessageAttachmentLinkBean

Attribute Set Standards

Attribute Set Standards

- Use an attribute set for every displayed field on the page.
 - Includes prompt or label, field descriptions for accessibility, any other text for the field
- Each displayable table column must have at least one associated attribute set.
- Foreign key column references use base columns' attribute sets.
- For lookup code columns, create an attribute set with a prompt suitable as a label for the lookup values.

ORACLE®

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

Attribute Set Standards

Attribute Set Standards

- Attribute sets associated with a single table are saved in a single package.
 - Attribute sets associated with translation table columns are saved in the base table package.
- In addition to attribute sets for columns, a table-related package includes:
 - Region header attribute sets
 - Table-specific action button attribute sets
- Standard button attribute sets are provided by OA Framework under fnd.attributesets.Buttons.xml

ORACLE®

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 Sets

Attribute Sets

For UI elements that don't map directly to a table column (transients):

- If the element maps indirectly to a table column, reuse that column's attribute set (example: search criteria or other non-database fields)
- If the element is reusable, create an attribute set under <your-prod>.mds.attributesets.Transient.xml
- Attribute sets for product-specific buttons can be added under <your-prod>.mds.attributesets.Buttons.xml

ORACLE®

Attribute Set Naming Conventions

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

More Attribute Set Standards

More Attribute Set Standards

The attribute set standards include lists of properties (templates) for defining various categories of attribute sets.

- Templates define which properties are mandatory or optional for defining an attribute sets.
- Properties not on the templates are NOT allowed.

ORACLE

Summary

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

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

R12.x Extend Oracle Applications: Building OA Framework Applications

Basics of the Controller

ORACLE®

Lesson Objectives

Lesson Objectives

After completing this lesson, you should be able to:

- Discuss how events are handled in OA Framework applications
- Create the controller for an OA Framework page that enables button handling, automatic queries, dynamic WHERE clauses, JSP forwards, and the messages from the Message Dictionary
- Identify the event flow within an OA Framework page (GET and POST events)

ORACLE

Recommended Build Approach

Recommended Build Approach

1. Create any business components packages that you need for your BC4J model objects.
2. Implement declarative BC4J application modules, entity objects, view objects and others as needed for your page(s). Add view objects to your root application module(s) as appropriate.
3. Create the menu definition for your application.
4. Create the OA user interface components for your page(s).
- 5. Create and implement controller code.**
6. Implement UI application module code supporting your pages.
7. Implement entity object business logic.

ORACLE

Steps 5 deals directly with Controller layer components.

Do You Need a Controller?

Do You Need a Controller?

In general, before tackling the question of *how* to design your controller, it's important to consider whether you even need to create a controller.

As a rule, you should write controller code only if absolutely essential. If you can create your page declaratively, do not instantiate regions and items programmatically.

Programmatically created web beans cannot be personalized, reused or extended. Furthermore, some hard-coded layouts may fall out of compliance with the standards as they evolve over time.

As required, all top-level regions in a shared component must have an associated controller.

ORACLE®

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

Controller Basics

- OA Controllers can be associated with any region. OA Controllers can not be associated with items.
- Never set properties on a parent/grandparent web bean from a child bean. Always define controllers so they control the regions with which they're associated, or set properties on children/grandchildren of this region. If you want a controller to manage multiple child/grandchild web beans, it should be associated with an appropriate parent/grandparent bean.
- For complex beans, you should associate a controller with the bean itself, or perhaps with a simple containing bean if it represents a logical unit of functionality.
- In general, you should create the fewest number of controllers per page that satisfies the rules and considerations outlined above.

ORACLE®

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.

Common Logic to Code

Common Logic to Code

There are several tasks you will do routinely in your code.

- Handle button press and other events
- Automatic queries
- Dynamic WHERE clauses
- Commits
- JSP Forwards

ORACLE®

Typical Locations for Code

Typical Locations for Code

The most common locations for your code will be:

- Controllers for regions
 - processRequest (code for page initialization such as **HTTP GET actions, after passivation, and so on**)
 - processFormRequest (code for **HTTP POST actions**)
- Application Modules
 - Called by controllers
- View Objects
 - Called by AMs to initialize and execute queries
- Entity Objects
 - Validation, creation, deletion (may also be in "Entity Expert")

ORACLE

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

Handling Queries

Handling Queries

Most view objects need a method on the VO to initialize and execute the query.

- The VO should be able to "query itself"
- Set up dynamic WHERE clause if needed
 - You must use Oracle Style binding (:1, :2,..., not ?)
- Naming convention: `initQuery` method
- Usually not needed for query bean or LOV VOs

ORACLE®

This is the most common use of a controller.

View Object initQuery Code

View Object initQuery Code

- Located in the VOImpl.java class file
- Is called from the application module query method
- Usually accepts query arguments
- Assembles a query statement using Oracle-style parameter binding
- Executes the query

ORACLE®

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

Dynamic WHERE Clauses

Generally you should avoid dynamic WHERE clauses.

- You usually get better performance with VOs and WHERE clauses that are defined at design time (using the VO wizard).
 - Better to create multiple similar VOs with different WHERE clauses than to modify WHERE clause at runtime.
- Sometimes you need to use a dynamic WHERE clause:
 - User-driven query where you do not know what columns will be in the WHERE clause.

ORACLE®

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

Using findByKey Instead of initQuery

BC4J caches entity objects. The same entity object can be referenced by multiple view objects, even in different pages (within the same root UI AM).

- Changes to an entity object in one page are reflected in a view object that references it from another page.
- For a VO based on entity objects, you can use the view object's findByKey method to locate a row without always making a database round trip.
 - Can also be used for SQL-only VOs
- For a single entity object that was already queried, using findByKey locates the data efficiently.

ORACLE®

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

Processing a Button Press

- A submit button in the UI does not inherently do anything.
- You must add code to make the submit button respond to a user click event.
 - Add button-handling code to the `processFormRequest` method **in a controller for the region**.
 - Check the button object to make sure it has been clicked.

ORACLE®

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

Getting Parameters from Requests

Getting Parameters from Requests

Request parameters are available from the PageContext object.

- For example (check if a button has been pressed):

```
if (pageContext.getParameter("Go") !=  
    null)  
    ...
```

- For example (get a parameter value into a buffer):

```
String employeeName =  
    pageContext.getParameter("employeeName");
```

ORACLE

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

Example: Manually-built Search

Example: Manually-built Search

There are 3 steps to make a Go button work for a manually-built Search region, for example:

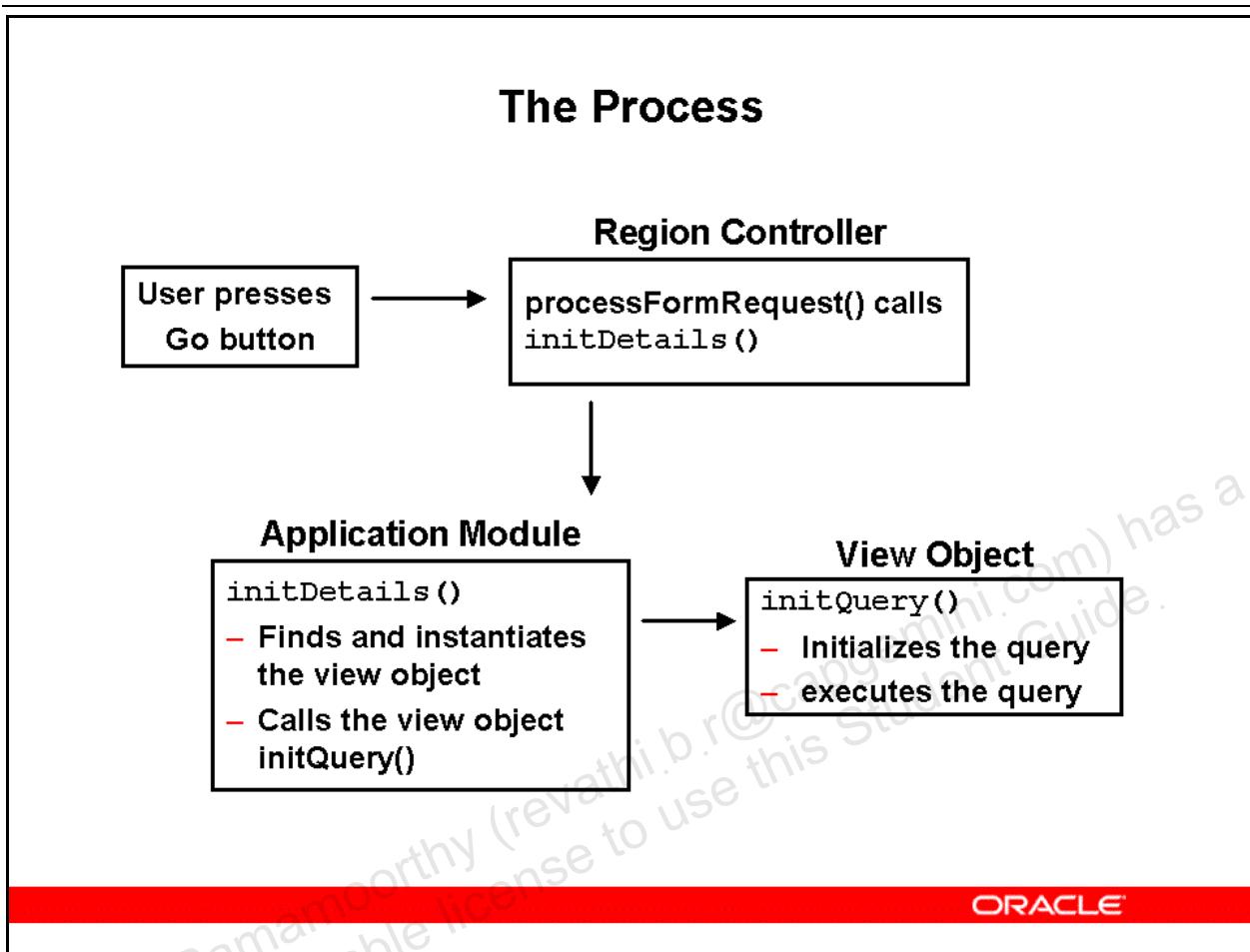
1. Add `initQuery` code to the `VOImpl.java` class that builds and executes a query
2. Add a query method to the `AMImpl.java` class that finds the view object and executes the `initQuery` method
3. Add code to the `processFormRequest` method in the `CO.java` (Controller) class that calls the application module query method

ORACLE®

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

Example VOImpl Code

```
public void initQuery(String employeeNumber)
{
    if ((employeeNumber != null) &&
        (!"".equals(employeeNumber.trim())))) {
        Number empNum = null;
        try {
            { empNum = new Number(employeeNumber); }
        catch(Exception e)
            { throw new OAException("AK",
"FWK_TBX_INVALID_EMP_NUMBER"); }
        setWhereClause("EMPLOYEE_ID = :1");
        setWhereClauseParams(null); // Always reset
        setWhereClauseParam(0, empNum);
        executeQuery();
    }
}
```

ORACLE

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.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

Example AMImpl Code

Example AMImpl Code

```
public void initDetails(String employeeNumber)
{
    EmployeeFullVOImpl vo = getEmployeeFullVO1();
    if (vo == null)
    {
        MessageToken[] errTokens = { new
        MessageToken("OBJECT_NAME", "EmployeeFullVO1") };
        throw new OAException("AK",
        "FWK_TBX_OBJECT_NOT_FOUND", errTokens);
    }
    vo.initQuery(employeeNumber);
} // end initDetails()
```

ORACLE

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

Example Controller Code

```
public void processRequest(OAPageContext pageContext,
    OAWebBean webBean)
{
    // Always call this first.
    super.processRequest(pageContext, webBean);
    // Get the employeeNumber parameter from the URL
    String employeeNumber =
        pageContext.getParameter("employeeNumber");
    // Now we want to initialize the query for our single
    // employee
    // with all of its details.
    OAApplicationModule am =
        pageContext.getApplicationModule(webBean);
    Serializable[] parameters = { employeeNumber };
    am.invokeMethod("initDetails", parameters);
}
```

ORACLE

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

Example Search: Controller

The controller for the manually-built Search in this example:

- Checks the button object to make sure it has been clicked
- Gets the Root Application Module
- Serializes the parameters
- Calls the appropriate 'Initialize Query' method in the Application Module
- Redraws the current page as necessary

ORACLE®

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

Forwarding to Another Page

Often a button press requires JSP forwarding to another page, either with a URL or a function name:

```
pageContext.setForwardURL(  
    "OA.jsp?page=<page URL>" , ...
```

ORACLE

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

Setting Titles with Message Dictionary

Always use Message Dictionary to get translated strings and messages for code that displays text on a page.

- Start by defining a message in Message Dictionary.
- Set tokens:

```
MessageToken[] tokens = { new  
    MessageToken("EMP_NAME",  
    employeeName) };
```

- Get the message and use it as a title:

```
String pageHeaderText =  
    pageContext.getMessage("ICX",  
    "FWK_TBX_T_EMP_HEADER_TEXT", tokens);
```

ORACLE

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.

Event Flow Overview

Event Flow Overview

Understanding the common OA Framework event flows is essential to aid:

- Coding
 - Logic execution order
- Debugging
 - Breakpoint placement

ORACLE®

Initial Setup Flow

Initial Setup Flow



When a user makes a request from the browser:

- Request is received by the JSP
- JSP invokes OAPageBean
- OAPageBean creates OAPageContext
 - Provides access to the state of the page
 - Provides hooks into OA Framework services

ORACLE

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.

Controller Event Flows in OA Framework

Controller Event Flows in OA Framework

- Two main OAController event flows
 - “Initialize page” used for HTTP GET (URL)
 - primarily `processRequest`
 - “Submit action” used for HTTP POST (Form Submit)
 - `processFormData` **and** `processFormRequest`
- Form submits (such as button presses) are directed back to the original OAController

ORACLE®

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

GET Event Flow – Overview

1. Get session info and validate user
2. Fetch metadata
3. Get Root AM and validate user session
4. Instantiate BC4J and UIX objects
5. Walk UIX tree and call processRequest on controllers
6. Perform post processing for complex beans
7. UIX renders page

ORACLE®

GET Event Flow (1-3)

GET Event Flow (1-3)

1. Get session info and validate user
2. Fetch metadata
3. Get Root AM and validate user session

ORACLE

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.

GET Event Flow (4) Instantiate BC4J and UIX Classes

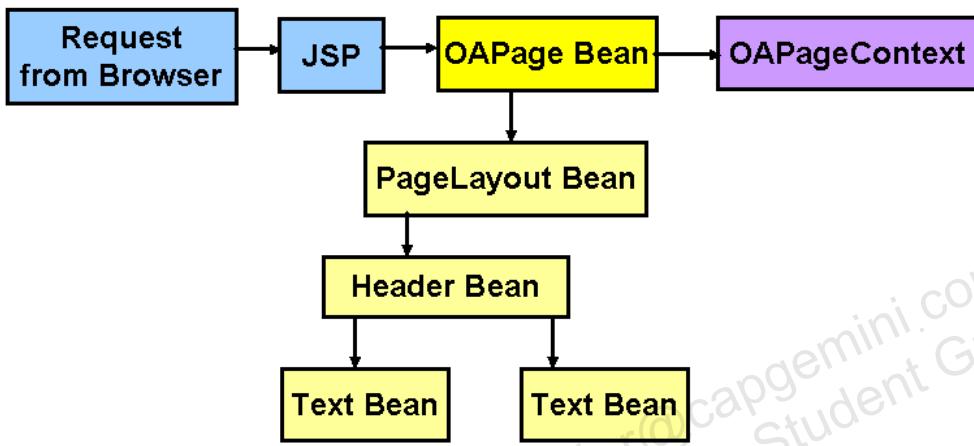
- 1.** Build OA web bean hierarchy from metadata of the page and any personalizations.
- 2.** Instantiate any associated BC4J objects.
- 3.** Place web bean bound values, if any.
- 4.** Cache the web bean hierarchy on the root AM.

ORACLE®

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

Example Bean Hierarchy Structure

Example Bean Hierarchy Structure



ORACLE®

GET Event Flow (5) processRequest

GET Event Flow (5) processRequest

1. Walk the web bean hierarchy.
2. Instantiate any controller classes.
3. Invoke the processRequest() methods on any controllers in the hierarchy.

ORACLE

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 (6) Post-Processing

GET Event Flow (6) Post-Processing

1. Perform post-processing on complex beans
 - OAPageLayoutBean and OATableBean for example
 - Post-processing can also be initiated from processRequest by calling prepareForRendering method
2. Places data binding objects on rendering context

ORACLE®

GET Event Flow (7) UIX Renders the Page

GET Event Flow **(7) UIX Renders the Page**

1. UIX generates page by calling UIX render methods recursively
2. UIX uses DataObjects and BoundValue interfaces to fill in values
 - DataObjectList – View Object
 - DataObject – View Row

ORACLE

POST Event Flow – Overview

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

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

- 1.** User does something to cause a form submit (such as button press or PPR event)
- 2.** UIX performs onSubmit JavaScript client-side validation on the browser
- 3.** Browser sends the POST request only if the client-side validation succeeds

ORACLE

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

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

- 1. Validate the user as in GET**
- 2. Retrieve AM and cached copy of bean hierarchy. If the bean hierarchy is not found:**
 - Validate session and function as in GET**
 - Go through processRequest logic again to recreate the bean hierarchy**
 - Your code must be prepared to expect this!**

ORACLE

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

Attribute-level validation:

3. Within view row `setAttribute`, the view row automatically calls the corresponding `set<AttributeName>` in the underlying entity object.
 - This executes any associated attribute-level validation in the entity object.

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

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 `OAAtrrValException` 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

POST Event Flow (7) processFormRequest

POST Event Flow (7) processFormRequest

- 1.** Walk UIX tree and call `processFormRequest` on controllers (second pass through the bean hierarchy)
 - Developers can respond to events or redirect to another page
- 2.** If no redirect (or if errors are thrown) redraw the page

ORACLE

Summary

Summary

In this lesson, you should have learned how to:

- Discuss how events are handled in OA Framework applications.
- Create the controller for an OA Framework page that enables button handling, automatic queries, dynamic WHERE clauses, JSP forwards, and the messages from the Message Dictionary.
- Identify the event flow within an OA Framework page (GET and POST events).

ORACLE

Setting Up Your Development Environment

Chapter 6

Unauthorized reproduction or distribution prohibited. Copyright© 2017, Oracle and/or its affiliates.

Revathi Ramamoorthy (revathi.b.r@capgemini.com) has a
non-transferable license to use this Student Guide.

R12.x Extend Oracle Applications: Building OA Framework Applications

R12.x Extend Oracle Applications: Building OA Framework Applications

Setting Up Your Development Environment

ORACLE

Lesson Objectives

Lesson Objectives

After completing this lesson, you should be able to:

- Install your JDeveloper 10g development environment on a Windows or Linux system
- Discuss the JDeveloper 10g development environment used for your course
- The actual steps to complete what is described in this lesson are in the class labs.

ORACLE

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

Installing and Setting Up JDeveloper

1. Refer to My Oracle Support Knowledge Doc **416708.1** to obtain the proper patch for your E-Business Suite instance.
2. Download the proper JDeveloper patch.
3. Install JDeveloper by unzipping it to a directory of your choice.
4. Configure your environment variables.
5. Obtain the database connection file (.dbc) for your E-Business Suite instance.
6. Create a shortcut to the JDeveloper executable.
7. Create a Framework Development user and responsibility for OA Framework testing.

ORACLE®

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.

Installing and Setting Up JDeveloper

8. Uncompress Tutorial.zip into your JDEVHOME directory.
9. Launch JDeveloper 10g.
10. Configure the connections.
11. Test your install and configuration.

ORACLE

Configure Your Environment Variables

Configure Your Environment Variables

1. JDEV_USER_HOME is a mandatory environment variable that points to the base directory where your development files are stored.
2. JDEV_JAVA_HOME is an optional environment variables that points to a specific Java SDK.

ORACLE

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

Get the DBC File

The file is located in:

\$INST_TOP/appl/fnd/12.0.0/secure

Put the file into your:

JDEV_USER_HOME/dbc_files/secure

ORACLE

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

Create a Shortcut

The executable for JDeveloper is located as follows:

- Windows – D:\JDev\jdevbin\jdev\bin\jdevW.exe
- Linux – \JDev\jdevbin\jdev\bin\jdev

Note: Linux does not run the .exe (Windows) executable.

Creating a shortcut varies by operating system as follows:

- Windows – create a desktop shortcut to the JDeveloper executable noted above.
- Linux – create an alias or shell script the points to the JDeveloper executable noted above.

ORACLE®

The example above should be verified with your instructor.

Assign the E-Business Suite User

Assign the E-Business Suite User

By default in the examples and in the OU classroom, you will use the following connection information:

Applications User: FWKTESTER

Applications Password: FWKDEV

Application Short Name: AK

Responsibility Key: FWK_TBX_TUTORIAL

ORACLE®

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

Uncompress Tutorial.zip

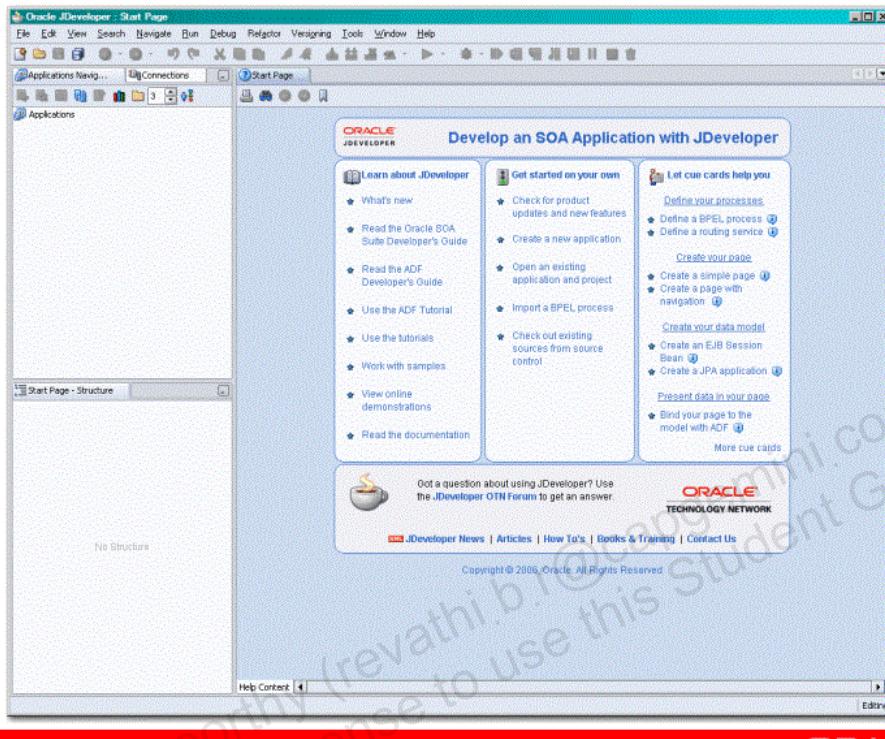
In the JDEVBIN directory, there is a file named, Tutorial.zip. Uncompress this file into your JDEVHOME directory.

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

ORACLE®

Launch JDeveloper 10g

Launch JDeveloper 10g



The method to launch JDeveloper will 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

Configure the Connections and Test

The steps to configure the connections and test are as follows:

1. Configure the database connection.
 - a. Connections tab on Navigator panel.
 - b. Right-click Database, select New Database Connection
 - c. Name the connection
 - d. Username = apps
 - e. Password = apps
 - f. Set the host name and SID. If you don't know the settings, look at the DBC file.
 - g. Test the connection
 - h. Finish

ORACLE

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

- ### Configure the Connections and Test
2. Click the Applications Navigator tab.
 3. Choose File > Open > myprojects > toolbox.jws.
 4. Expand toolbox in the Applications Navigator panel.
 5. Double-click Tutorial to open Project Properties window.
 6. Expand the Oracle Applications menu entry.
 7. Choose Oracle Applications > Runtime Connection.
 8. Set the DBC File Name to the directory and name for your DBC file.
 9. Click the Save All icon to save your progress 
 10. Expand Tutorial.
 11. Expand the Web Content folder.
 12. Right-click test_fwtutorial.jsp, and choose Run.

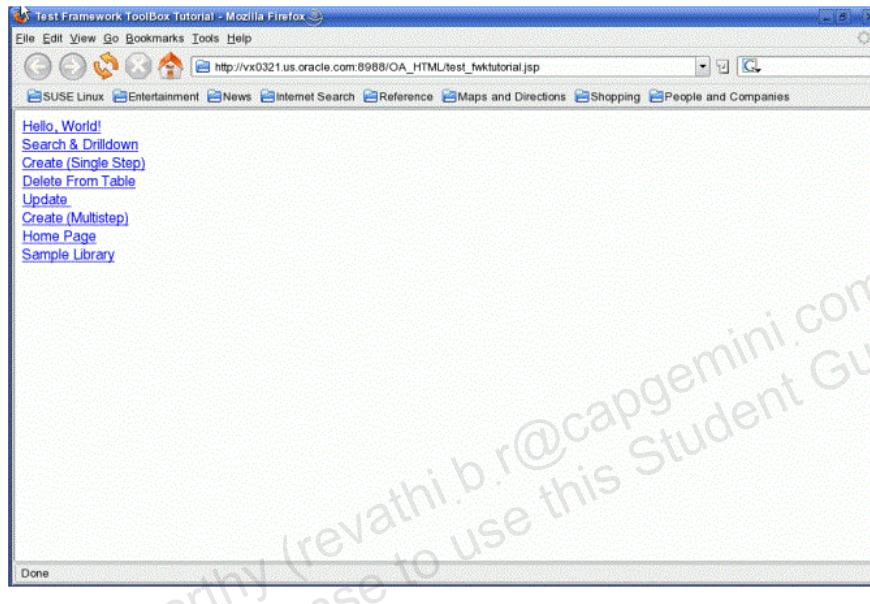
ORACLE

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

Configure the Connection and User

13. Test the test_fwtutorial.jsp Page

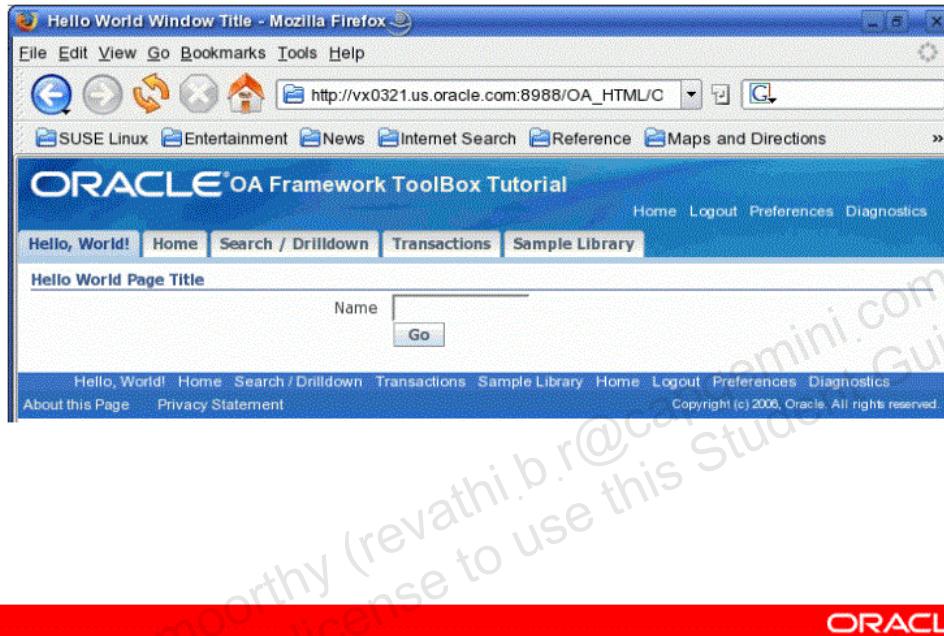


ORACLE®

Configure the Connection and User

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_fwtutorial.jsp page and the Hello, World! page, you know several things:

You have connected to a database and tested test_fwtutorial.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

Summary

In this lesson, you should have learned how to:

- Install your JDeveloper 10g development environment on a Windows or Linux system.
- Discuss the JDeveloper 10g development environment used for your course.

ORACLE®

Note: The actual steps are contained within the Lab Exercises