Oracle ADF & JDeveloper for Forms Developers

Presented by: John Jay King
King Training Resources - john@kingtraining.com

Download this paper from: http://www.kingtraining.com

Objectives

- Be aware of Oracle's Application Development Framework (ADF) standards-based applications
- Understand ADF development using advanced graphical editing and declarative techniques
- Know ADF's Java and XML underpinnings
- See how JDeveloper creates ADF applications
- Grasp ADF BC's reusable data source support
- Learn how to build UIs graphically with drag and drop components, and declarative properties

Who Am I?

- John King – Partner, King Training Resources
- Providing training to Oracle and IT community for over 20 years – http://www.kingtraining.com
- "Techie" who knows Oracle, SQL, Java, and PL/SQL pretty well (along with many other topics)
- Leader in Service Oriented Architecture (SOA) design and implementation
- Member of ODTUG (Oracle Development Tools User Group) Board of Directors
- Moved to Scottsdale, Arizona last year (we miss Colorado!)
- Active member of Rocky Mountain Oracle Users Group (RMOUG)

Who Are You?

- Forms Developer
- 4GL Developer
- Java Developer
- All of the above
- None of the above

Why Oracle ADF?

- Oracle Application Development Framework (ADF) is a Java-based development tool (much like Oracle Forms is a PL/SQL-based tool) designed to take full advantage of Java Enterprise Edition or Java EE
- Java EE is one of the most widespread application environments today
- ADF's 4GL features make application development much easier than normal Java "coding"
- Oracle is rewriting their ERP stack as "Fusion Applications" using ADF; the already rich toolset gets richer every day

Do I Need to Know Java ?

- Probably not well
  - Someone with very basic Java and Web Skills can easily create applications with ADF (much the same as someone with basic PL/SQL could create very basic Oracle Forms)
  - Someone on your team needs to know Java very well
  - Someone on your team needs to know Groovy
- Someone on your team needs to understand ADF and its available components very well

Is Forms Going Away?

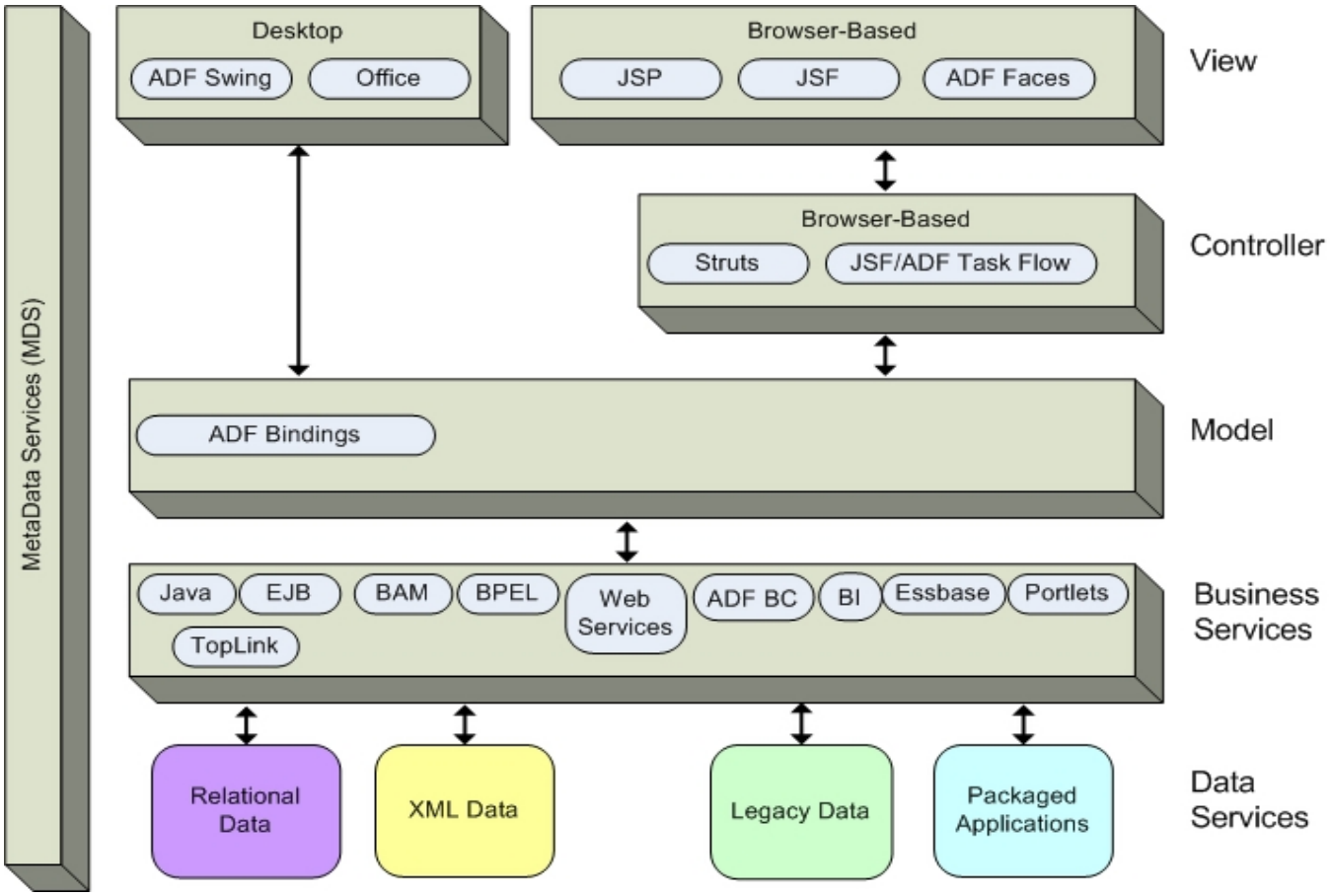
- NO, NO, NO, NO, NO
- Oracle is committed to supporting Oracle Forms for many years to come
- A new version of Oracle Forms (12c) is on the way!

What is ADF?

- ADF is a "meta-Framework" interacting with a variety of underlying software components (including Frameworks) to provide:
  - Database connectivity and transfer
  - Mapping of application views to data sources
  - Database interaction: constraints, keys, data types, master/detail, null handling
  - Data caching via entity objects
  - Transaction management (locks, commit, rollback, etc...)
  - Declarative validation
  - Business logic and event handling
  - User Interface (UI) logic, flow, look & feel
  - Data-bound UI Components
  - UI properties including: formatting, colors, defaults, visual components, LOVs, etc...

ADF Technology Stack

- ADF Technology simplifies interaction with "Java" EE and Oracle's Fusion Middleware



ADF: Two Major Pieces

- ADF has many parts but two are central to creating applications
  - ADF BC Business Components (data)
  - ADF Faces Graphical User Interface

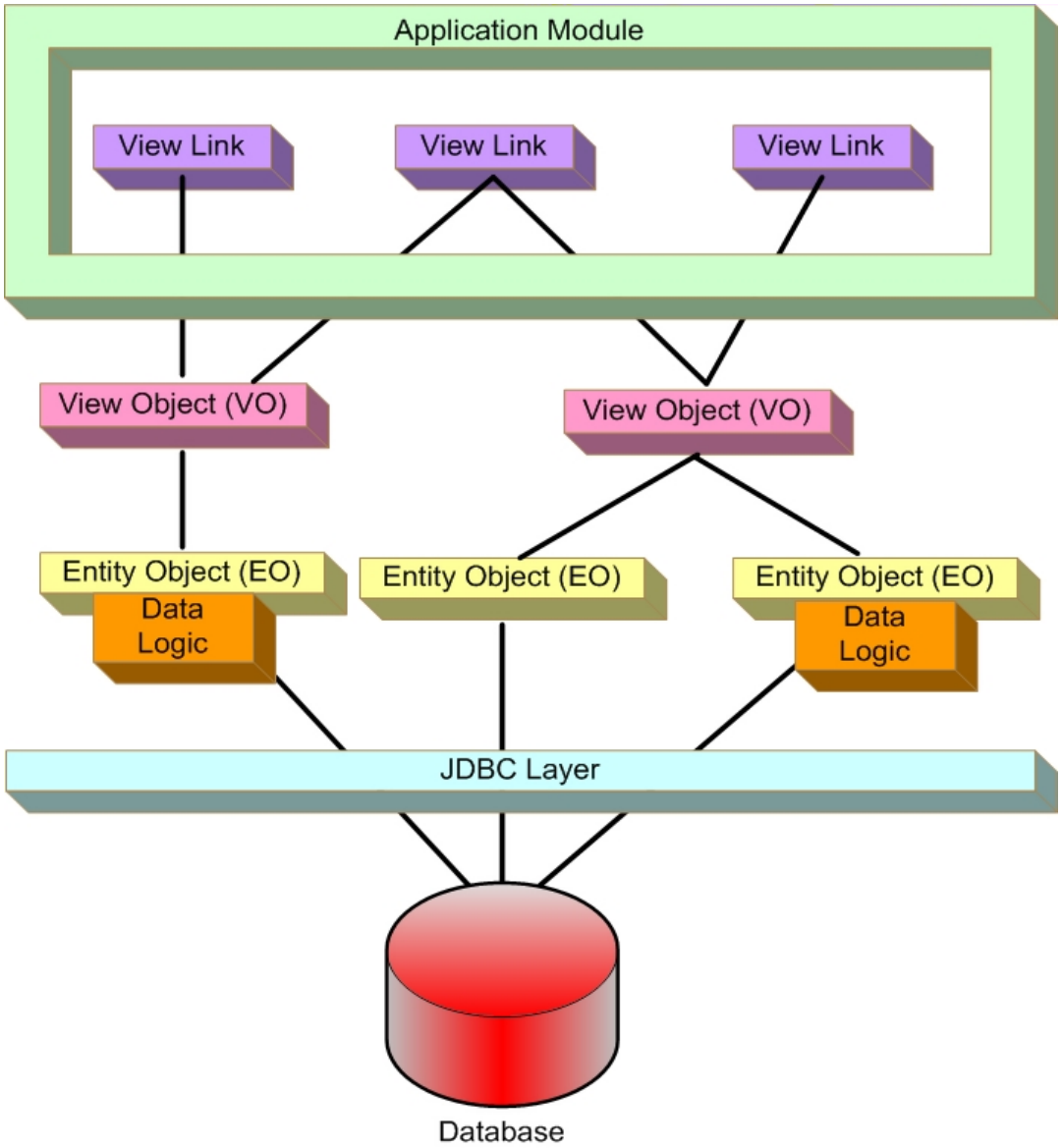
ADF Business Components

- ADF Business Components (ADF BC) is a framework that simplifies developing Java EE business services
- ADF BC is part of the ADF Business Services layer and is used to provide:
  - Persistence and data retrieval with SQL using data views
  - Object-Relational Mapping (ORM) between Java classes and database data
  - Simplified data access, validation, and business logic
  - Transactional infrastructure
  - Implementation of best practices

ADF BC Objects

- ADF BC is implemented using a variety of objects to:
  - Define Insert-Update-Delete views to perform queries and data manipulation
  - Define query views (read-only)
  - Define links between queries

ADF BC - Component Structure



ADF BC Components

- ADF BC uses a variety of object types to represent data:
  - Database tables and views — Application Base Data
  - Entity Objects — Business rules, validations, defaults for a table (or view)
  - View Objects — SQL output to query, filter, join,modify, or sequence data
  - Application Modules — Use View Objects to access/modify data acting as a back-end data service
  - Appl. Module Data Model — Describes actual View Object uses
- Objects may be reused in multiple Application Modules

ADF Data Binding

- After identifying Entity Objects and View Objects two additional ADF Data Model components are used
  - Data Controls — Java objects used to abstract View Object Business Services
  - Binding Containers — Java object; provides data access to a single ADF application page, fragment, or activity

Java Server Faces (JSF)

- Java Server Faces (JSF) is a Web-tier framework of JSP technology and JSP Tag libraries to create and use User Interface components
- JSF is extended by components of Oracle ADF Faces
- JSF includes:
  - Runtime architecture
  - Library of JSF components
  - JSF "Life Cycle"
  - Many JSF-Oriented Files

ADF Faces

- Even though JSF sought to simplify user interface; it is often felt to be too complex
- Oracle has extended JSF as "ADF Faces" providing a set of libraries and tags that include enhanced UI components and easier use
- Oracle has presented ADF Faces to the Open Source community where it is part of the Apache Foundation Trinidad MyFaces project

http://myfaces.apache.org/trinidad/index.html

Using ADF

- Using ADF Faces is simple using JDeveloper:
  - Add Application layout containers to describe user interface
  - Add ADF Faces components to layout containers
  - All UI is done with ADF Faces; no HTML coding
- Features added by ADF Faces:
  - Pop-ups and Dialog boxes
  - Data Visualization Tools: Charts, graphics, etc...
  - Declarative AJAX support
  - More…

ADF Controller

- The ADF Controller extends the JSF controller and controls ADF's MVC (Model-View-Controller) in ADF
- ADF Controller features include:
  - Sequence of page displays (may be conditional)
  - Allows partial-page processing in the same way as full page processing; only the necessary part of a page is rendered, the rest is unchanged
  - Allows reuse of page parts
  - Provides conditional control of page flow

ADF Faces "Rich-Client"

- ADF Faces is designed to create "rich-client" (RC) interfaces; full-featured and declarative including:
  - Complete JDeveloper support graphic development (screen-painter) and property palettes
  - Visual Editor
  - Property Inspector
  - Changeable "skins" to easily alter look-and-feel
  - Modifiable look-and-feel properties (declarative)
  - Layout control

Oracle JDeveloper

- JDeveloper provides a world-class, easy to use IDE
- Oracle has extended JDeveloper beyond Java to include:
  - Oracle ADF modeling, business services, and GUI design
  - XML edit including Syntax Checking & Schema Validation
  - SQL development including debugging of stored PL/SQL
  - UML Modeling and MDA (Model Driven Architecture)
  - Web Services development
  - ESB design
  - BPEL design
  - Portlets

Downloading JDeveloper

- JDeveloper is Free!
- To learn more about JDeveloper, see Oracle's website:

http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html

Oracle WebLogic Server

- Oracle WebLogic Server is Oracle's preferred platform to provide both a standard Java EE environment and an environment specifically tailored to Oracle Fusion Middleware; providing:
  - Complete Java EE 5 compatibility
  - Complete Java SE 6 compatibility
  - Web Services support
  - Integration with Oracle's Fusion Middleware tools

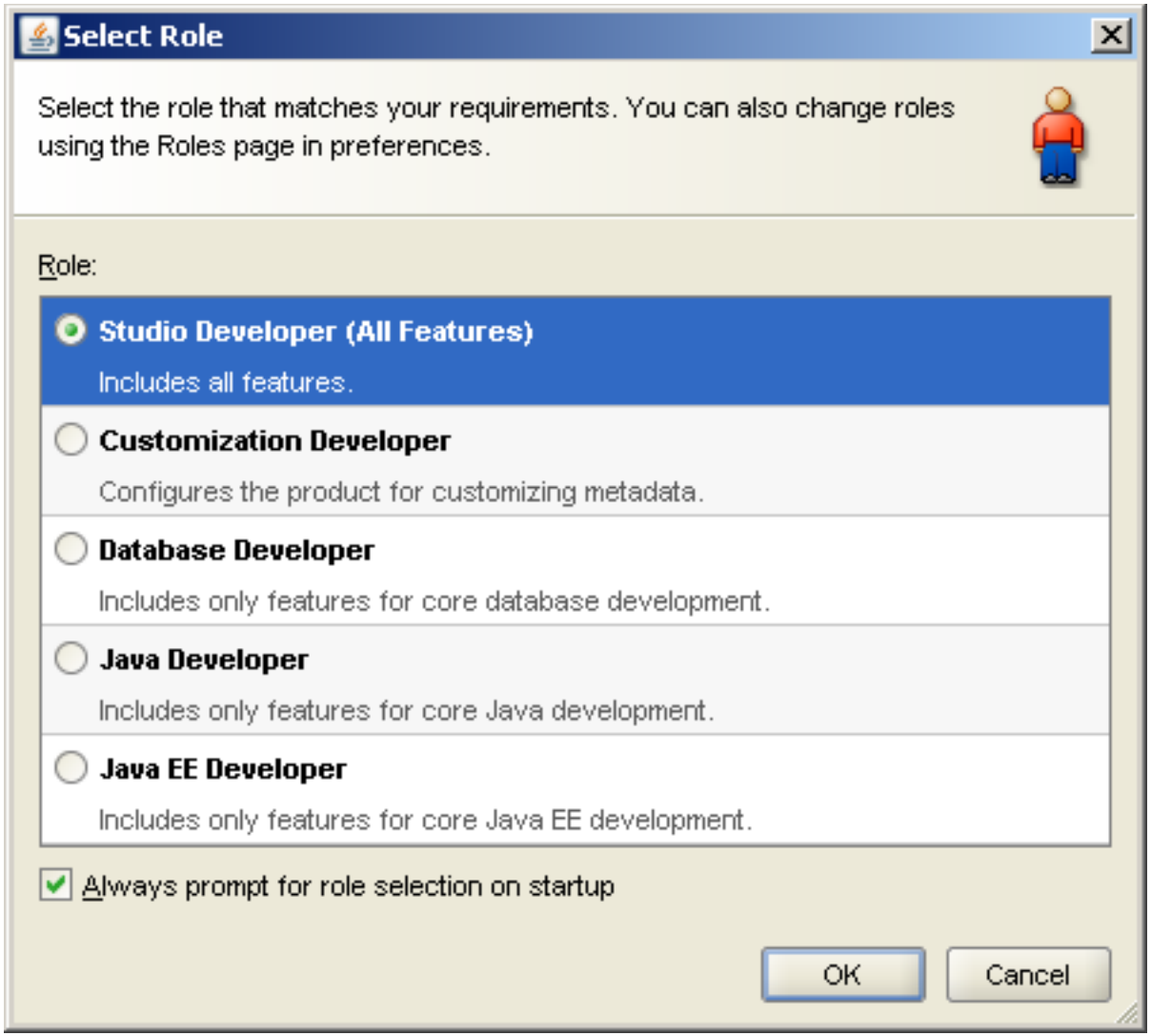
Oracle AS and OC4J?

- Oracle WebLogic Server is the replacement for Oracle Application Server (OAS) and OC4J
- OAS and OC4J are still supported and may be used instead of WebLogic if desired but ADF requires at least Java 1.5 / Java 5 (needed for ADF)
- To learn more about Oracle WebLogic Server see Oracle's website:

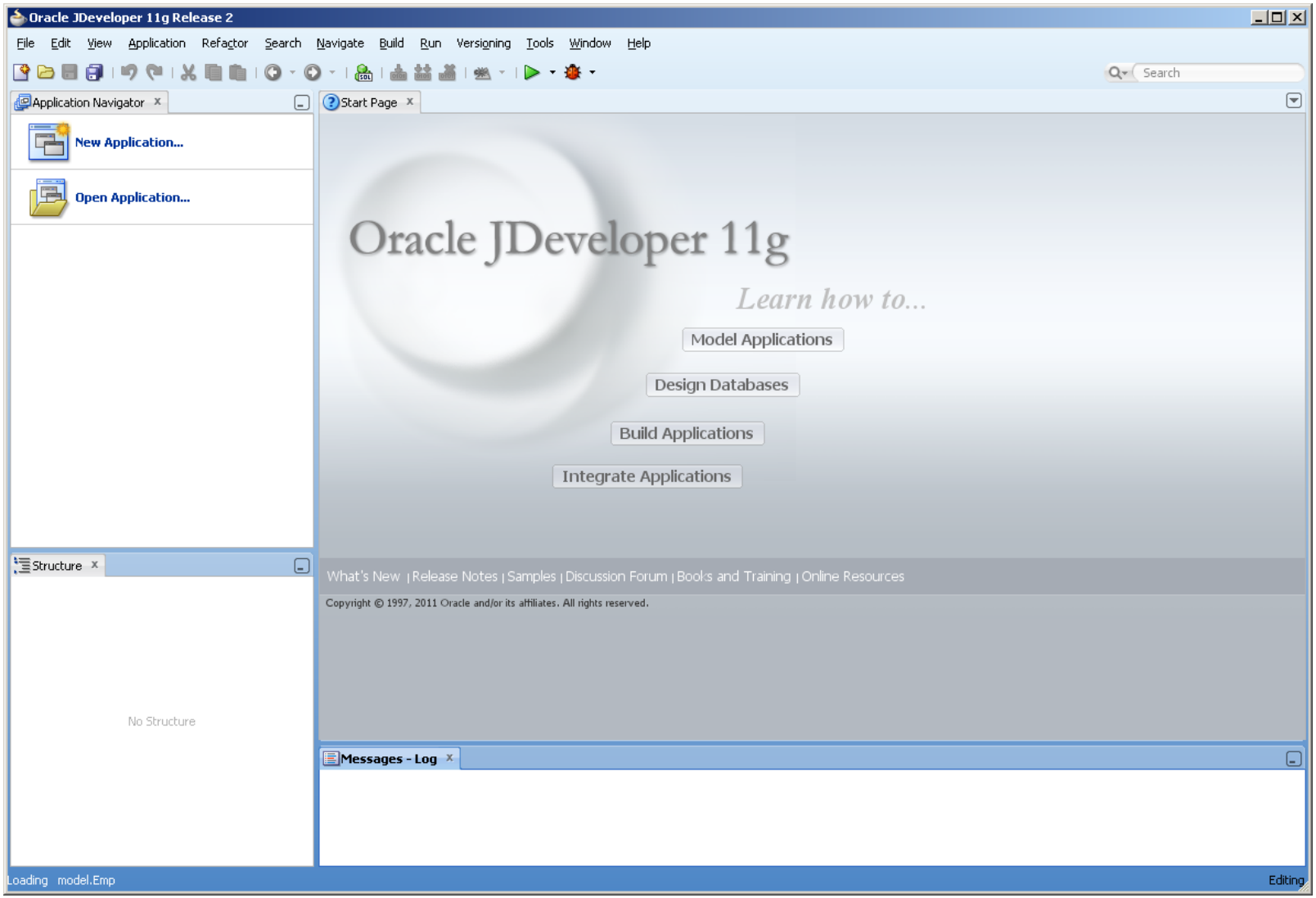
  http://www.oracle.com/us/products/middleware/application-server/index.htm

Starting JDeveloper

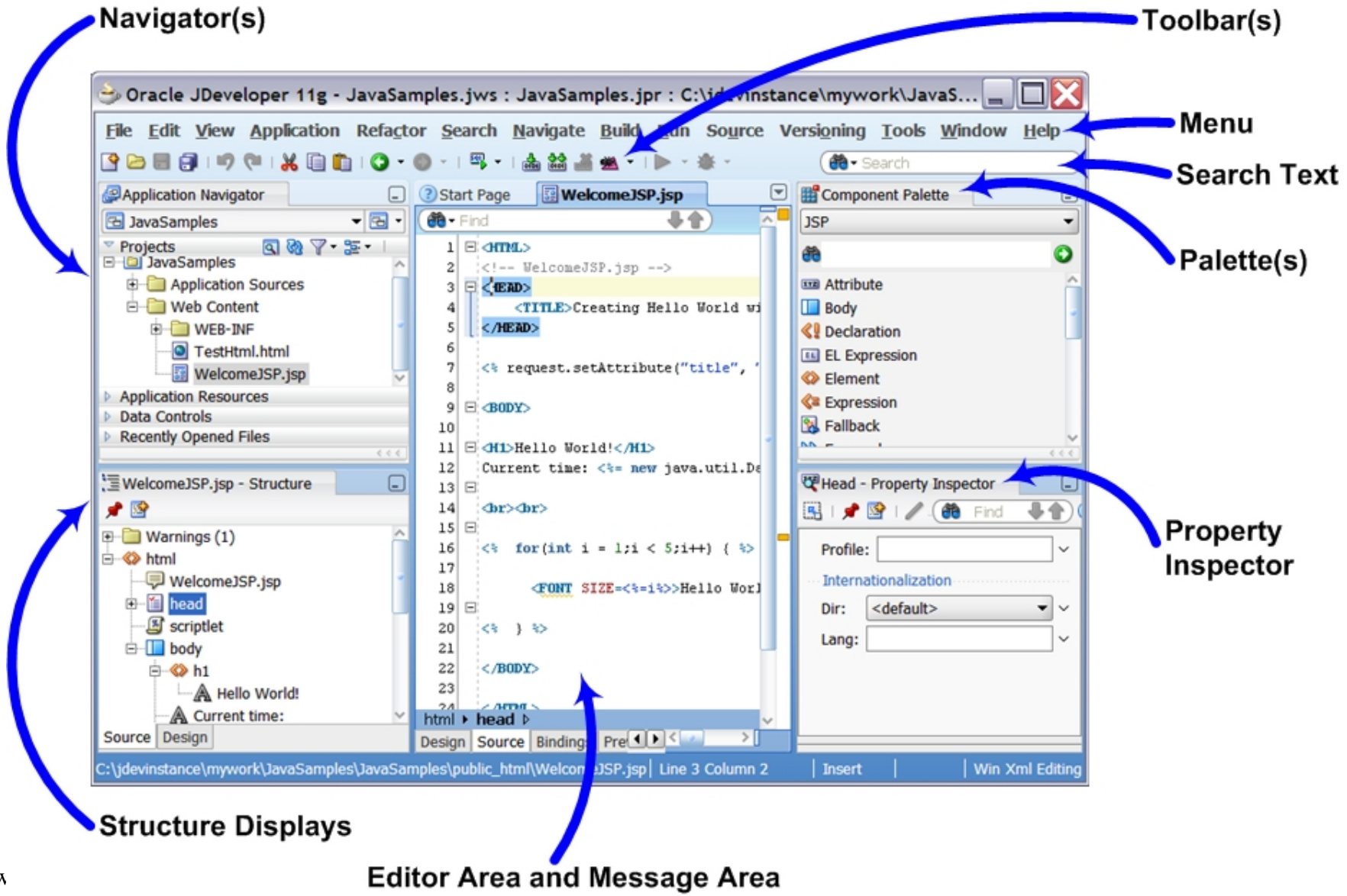
JDeveloper - Select Role



JDeveloper - Start Page



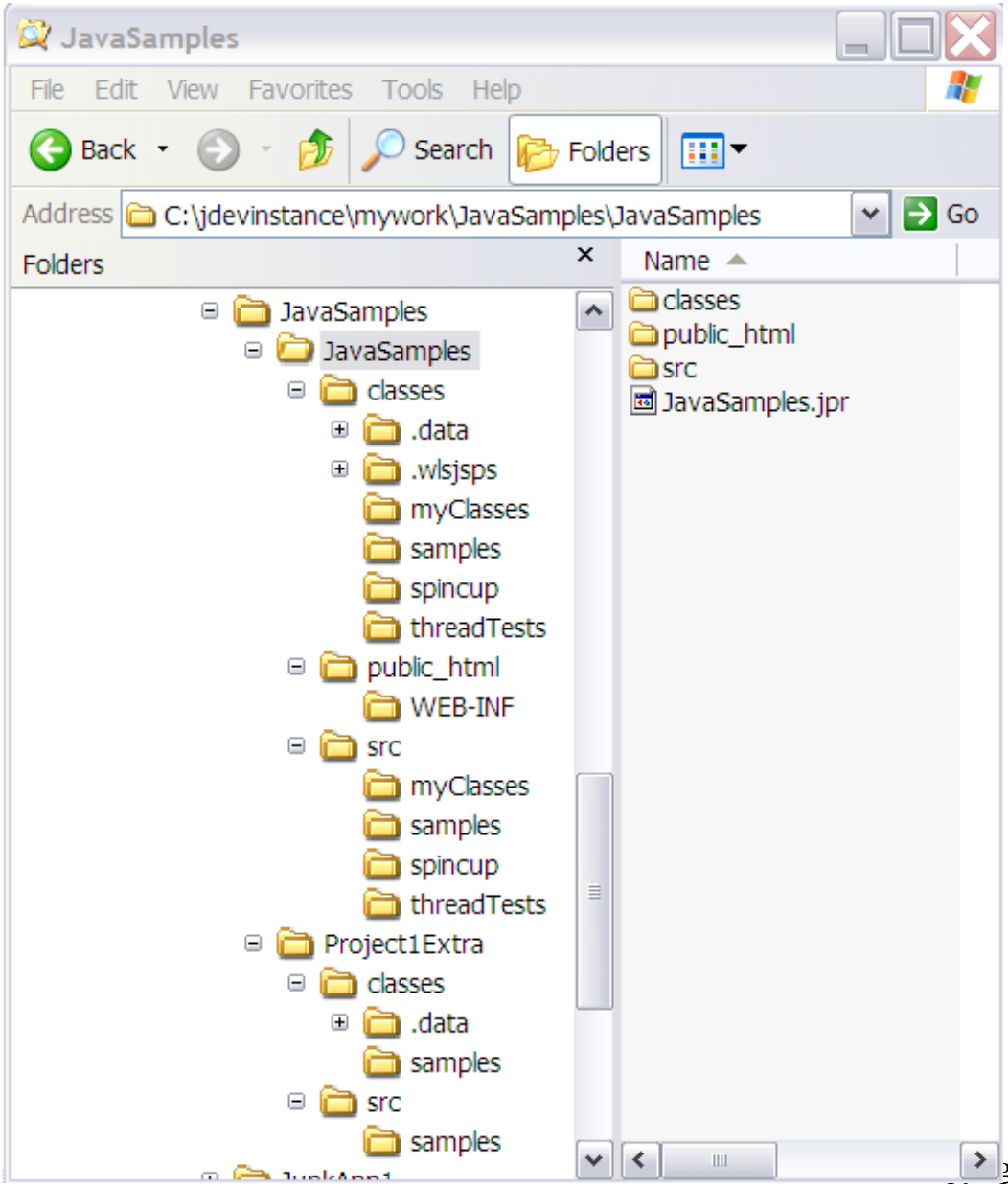
Exploring JDeveloper



Applications and Projects

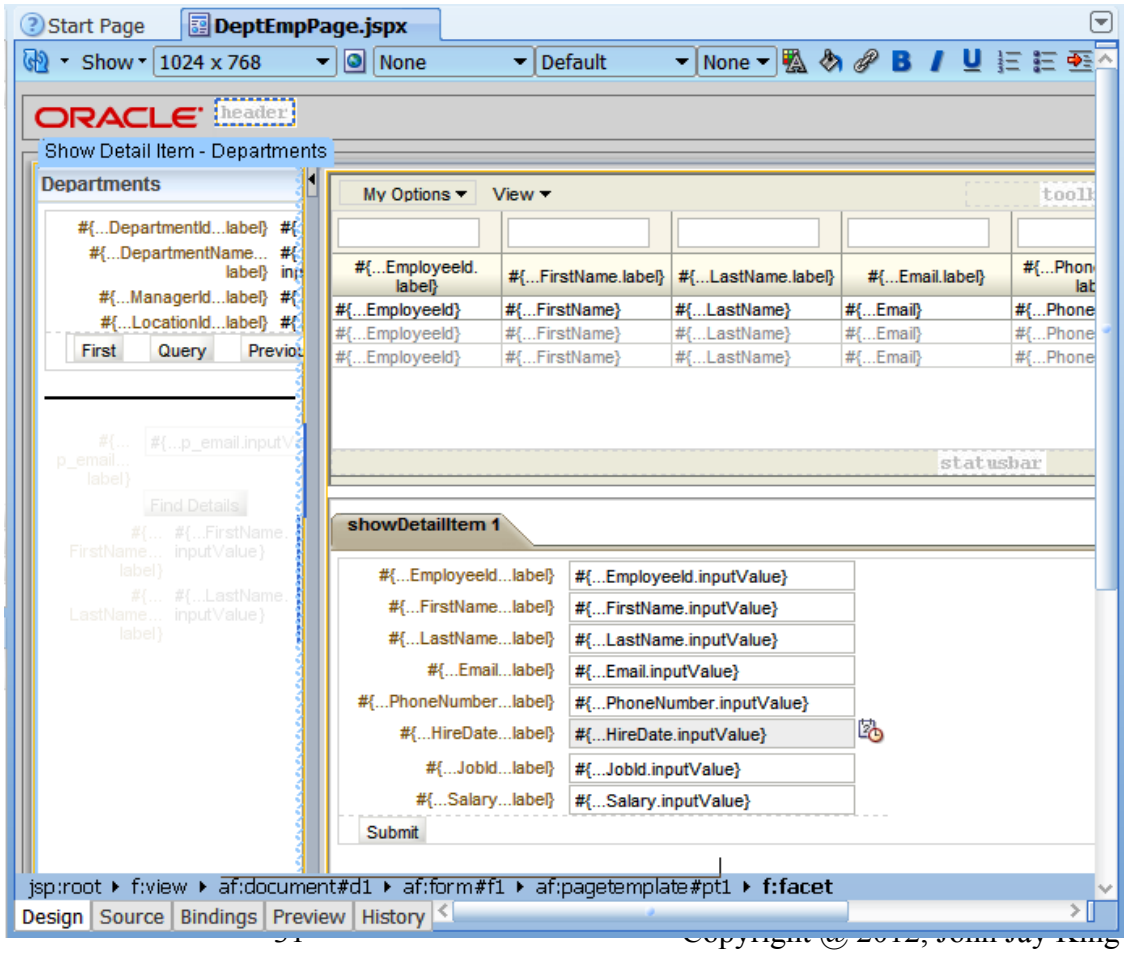
- JDeveloper uses a non-standard, Oracle-specific "Application" to group a collection of "Projects" (similar to how it is done by other IDEs)
- All files representing an "Application" share a common root directory (folder) on a disk
- Many Applications may be open at once in JDeveloper; but only one at a time will be visible in the Application Navigator

JDeveloper Directory Structure



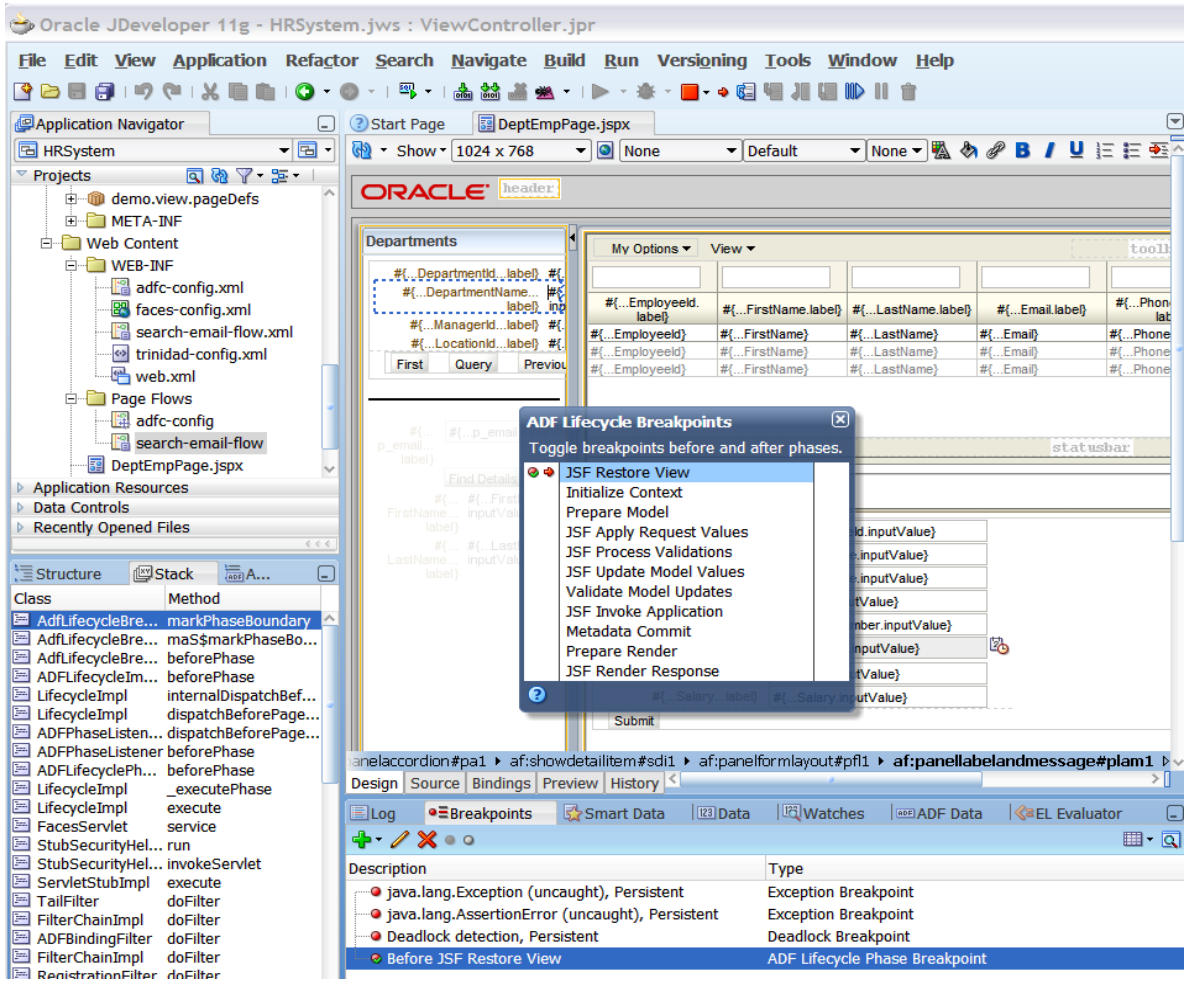
JDeveloper Editing

- JDeveloper has many Code Editors & Visual Editors including: Java, XML, HTML, JSP, JSF/ADF Faces, BPEL, & more



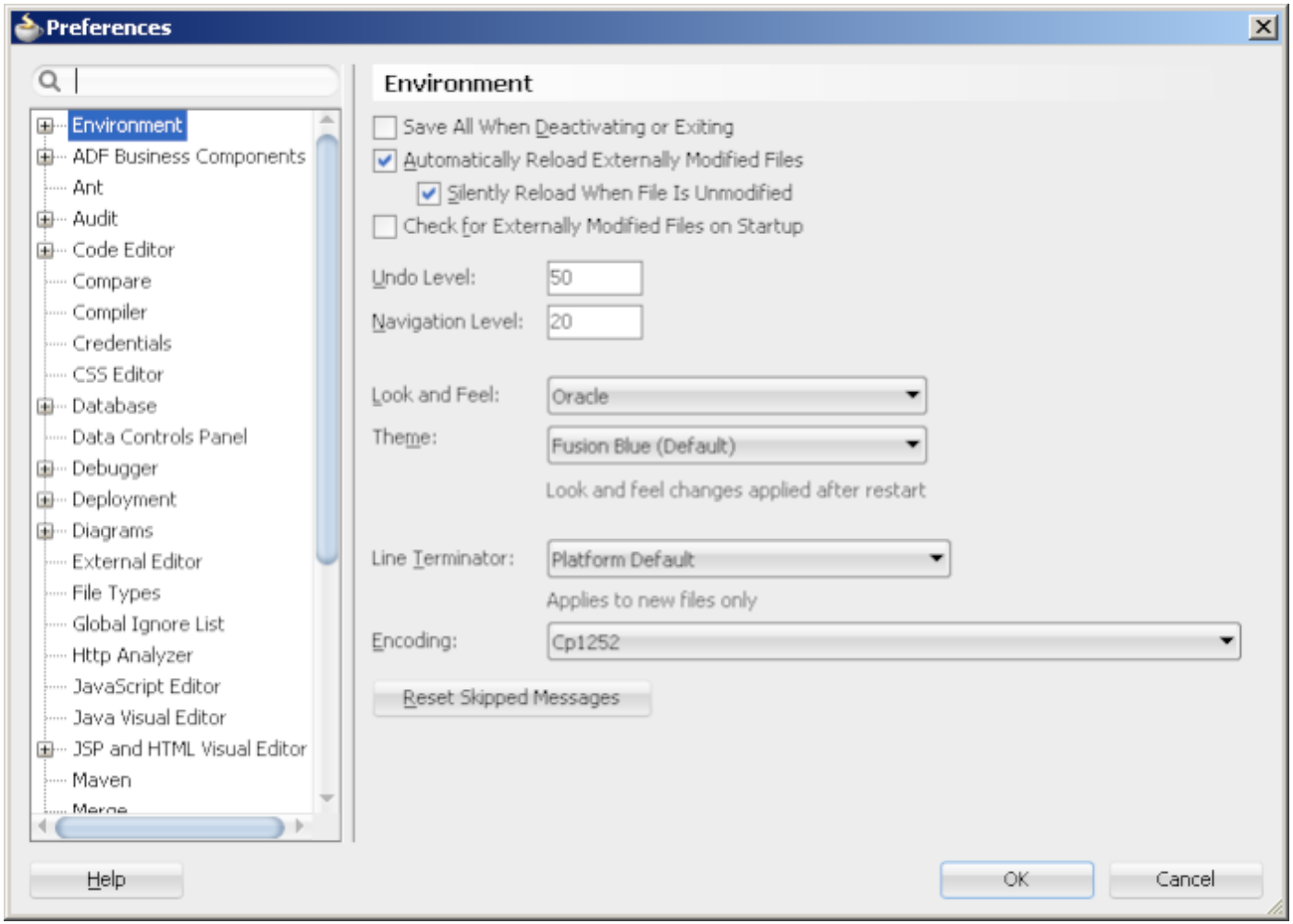
JDeveloper Debugging

- JDeveloper allows local and remote debugging



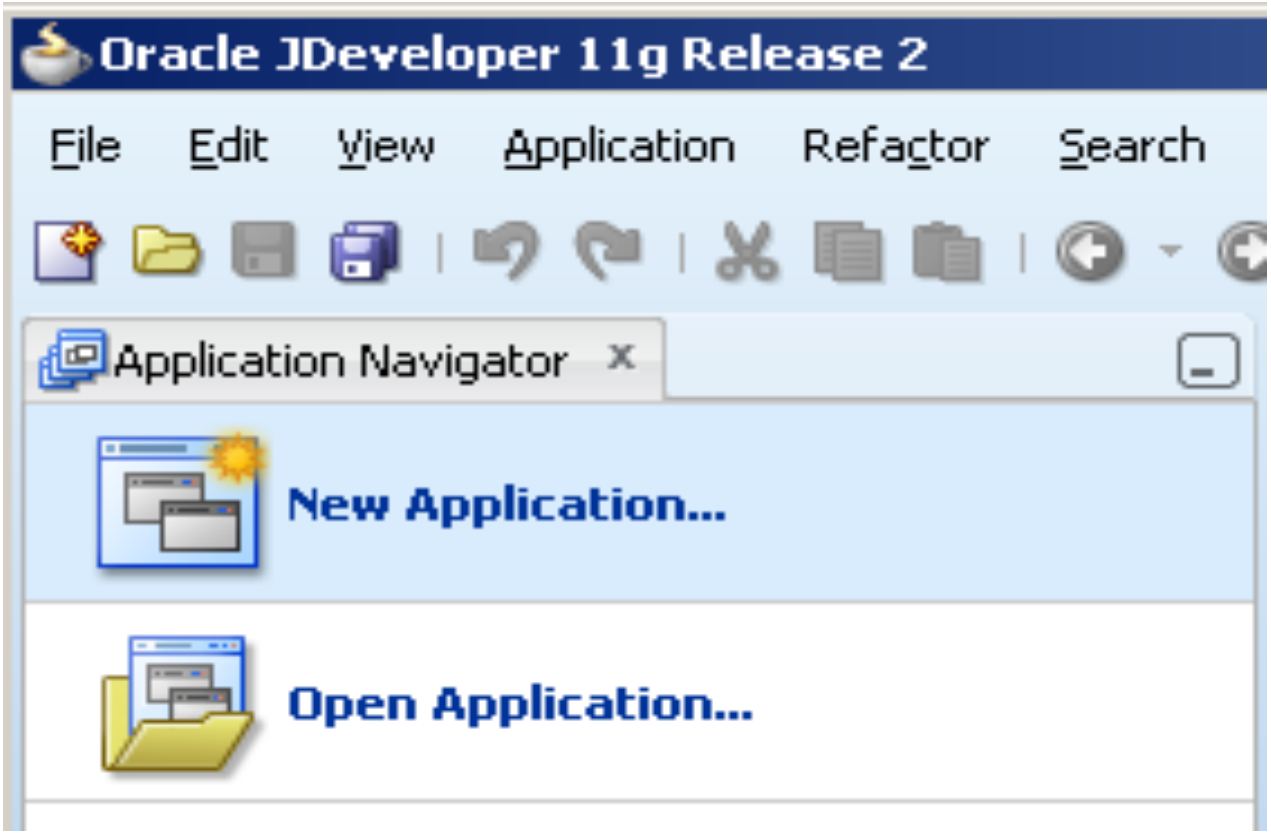
JDeveloper Preferences

- JDeveloper is customizable; preferences may be viewed/modified using Tools->Preferences

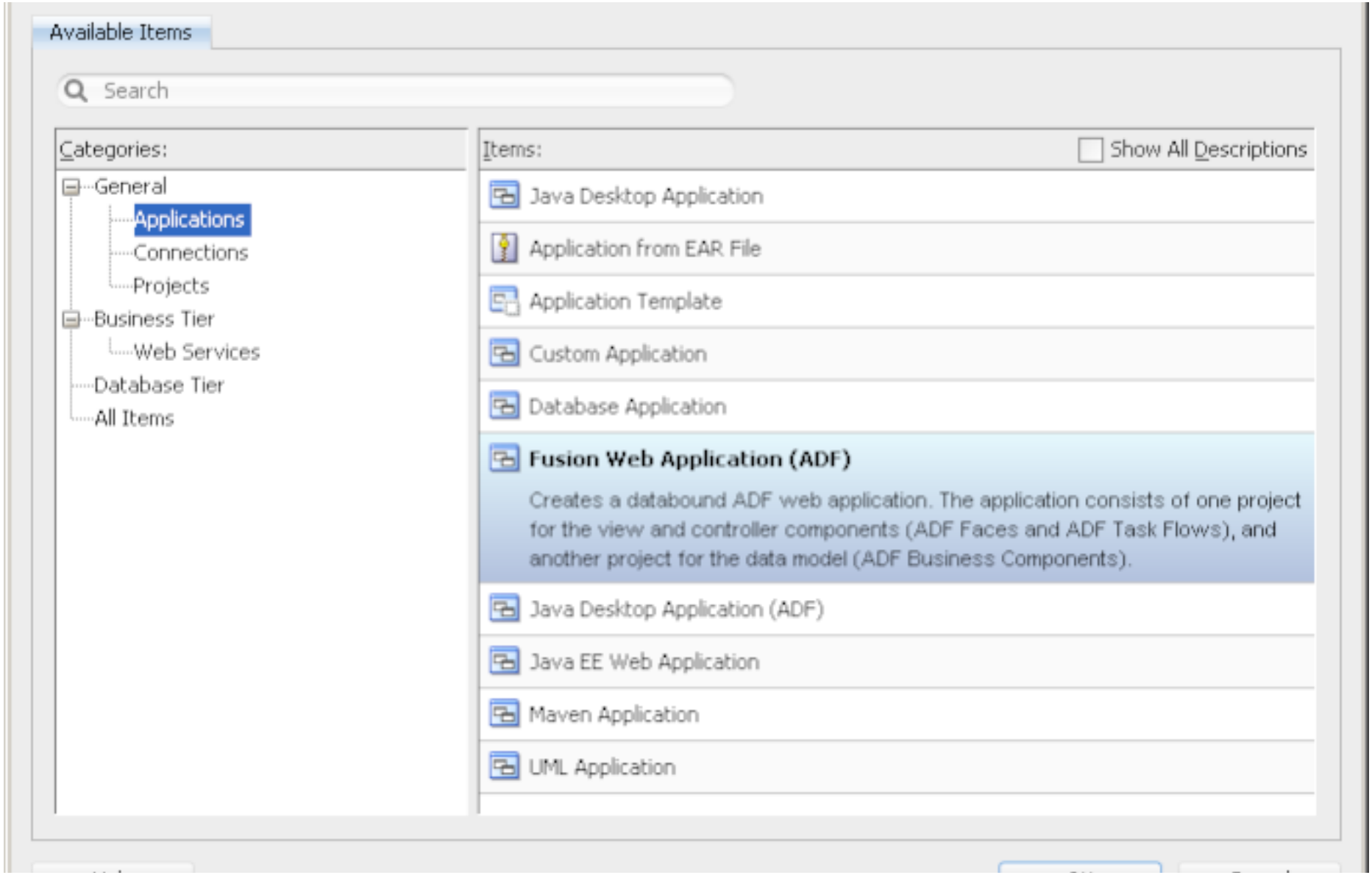


New Application

- To create a new application use the JDeveloper menu's File->New->General->Applications option, or click Application Navigator -> New Application



New Gallery

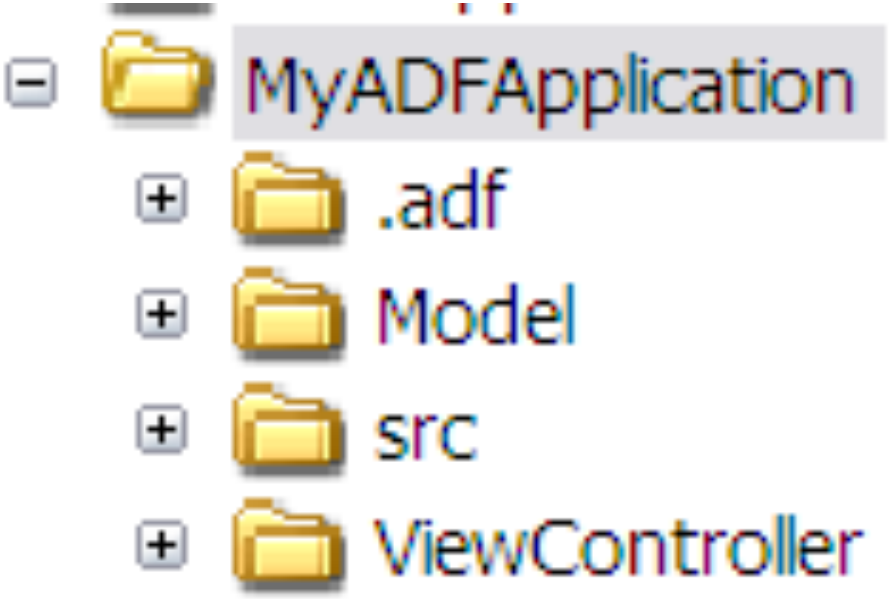


Application Structure

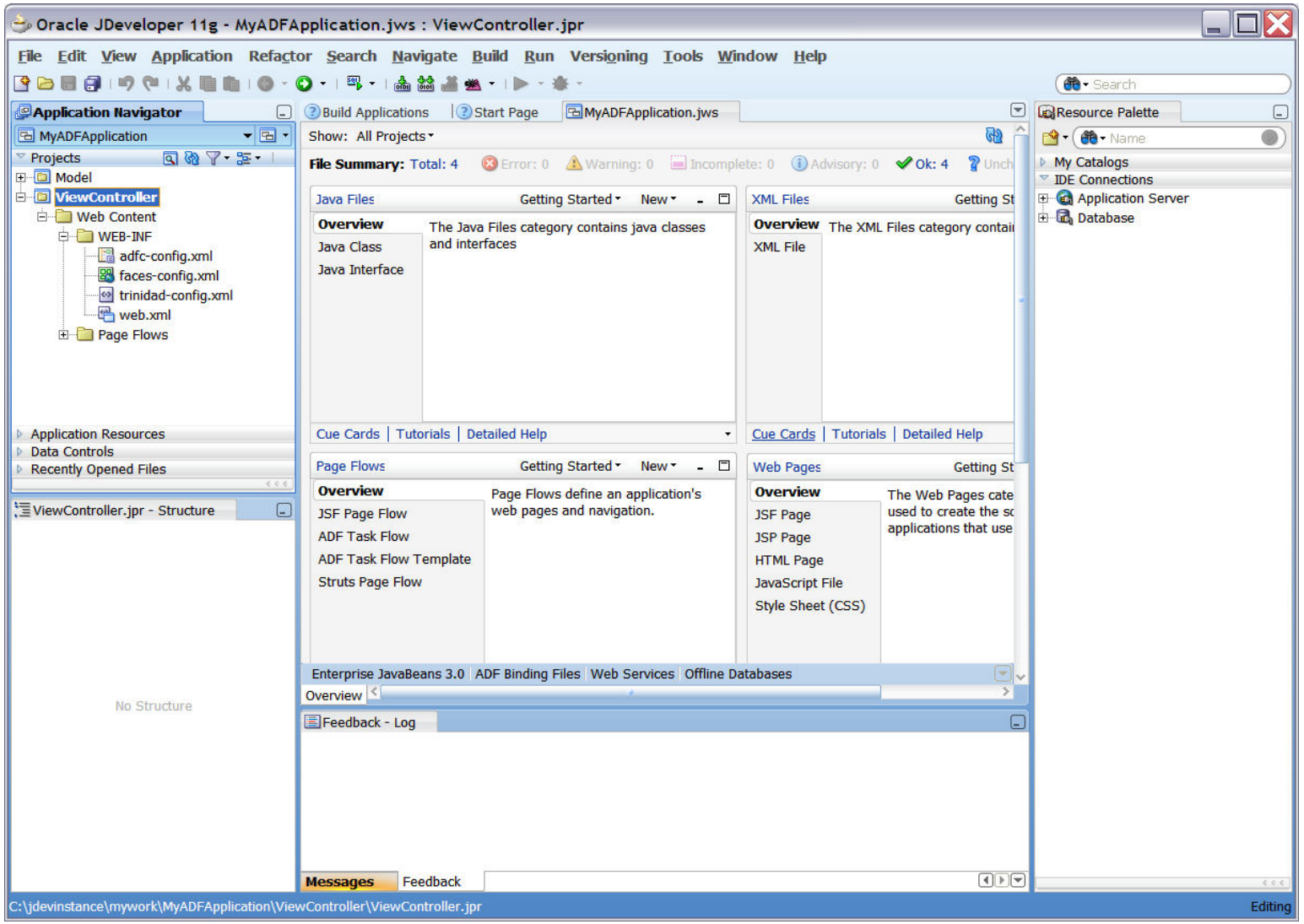
- When a JDeveloper ADF Web Application is created ADF uses the MVC (Model-View Controller) pattern
- JDeveloper creates two subordinate projects
  - Model — Data and Business Rules
  - ViewController — User Interface
- ADF provides the "Controller"

File Structure

- Review the directory structure created to support the application and the associated projects



How It Looks In JDeveloper



Create ADF BC Objects

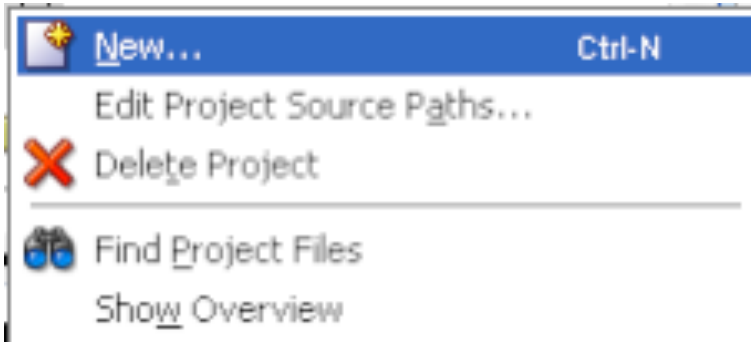
- The following pages show how to create ADF BC objects using the Wizards provided by JDeveloper
- Each object created may be created individually using JDeveloper's features or by coding them manually rather than using the Wizards
- JDeveloper's database modeling capabilities are shown to good effect by the use of Database Connections and Wizards

Wizard-Based Development

- The "Create Business Objects from Tables" Wizard follows a few simple steps:
  - Create Business Component, select type of Business Component to be built
  - Select Database Connection to be used (may create Database Connection via Wizard)
  - Build Entity Objects using database Tables/Views
  - Build Updateable View Objects (if desired)
  - Build Read-Only View Objects (if desired)
  - Save Application Module

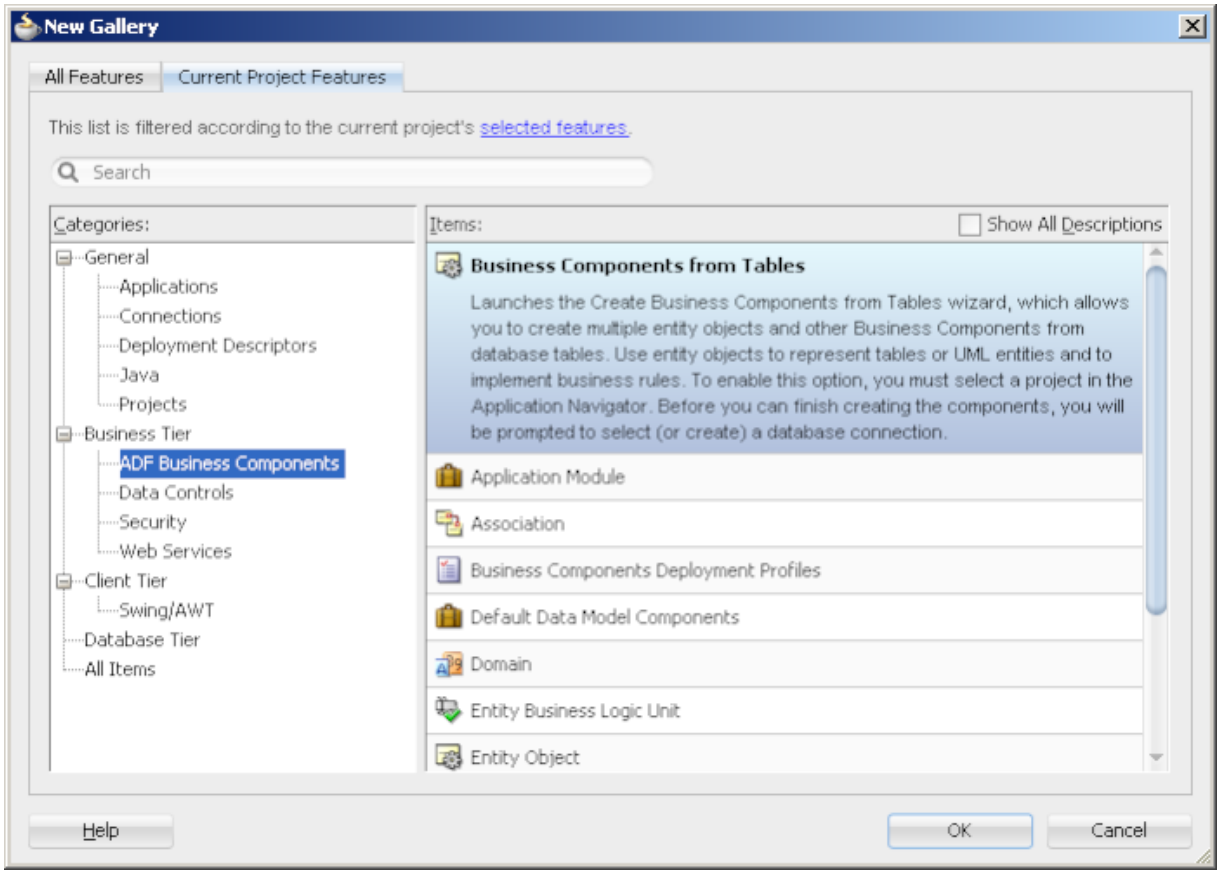
New ADF BC Object: 1

- Start building new components as follows:
- Right-click on the application's “Model” project and choose “New”



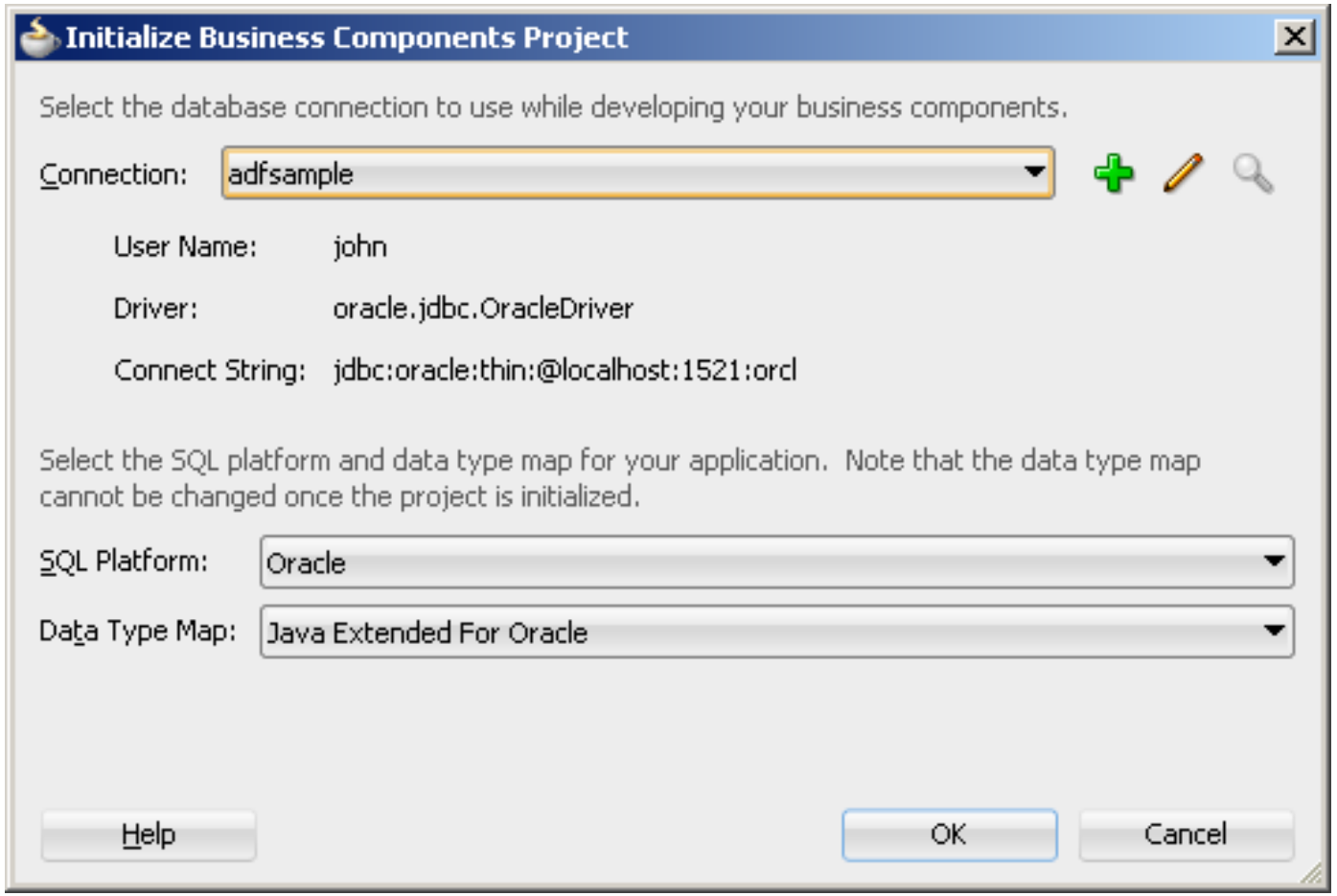
New ADF BC Object: 2

- Choose **Business Tier -> ADF Business Components -> Business Components from Tables** from the "New Gallery"

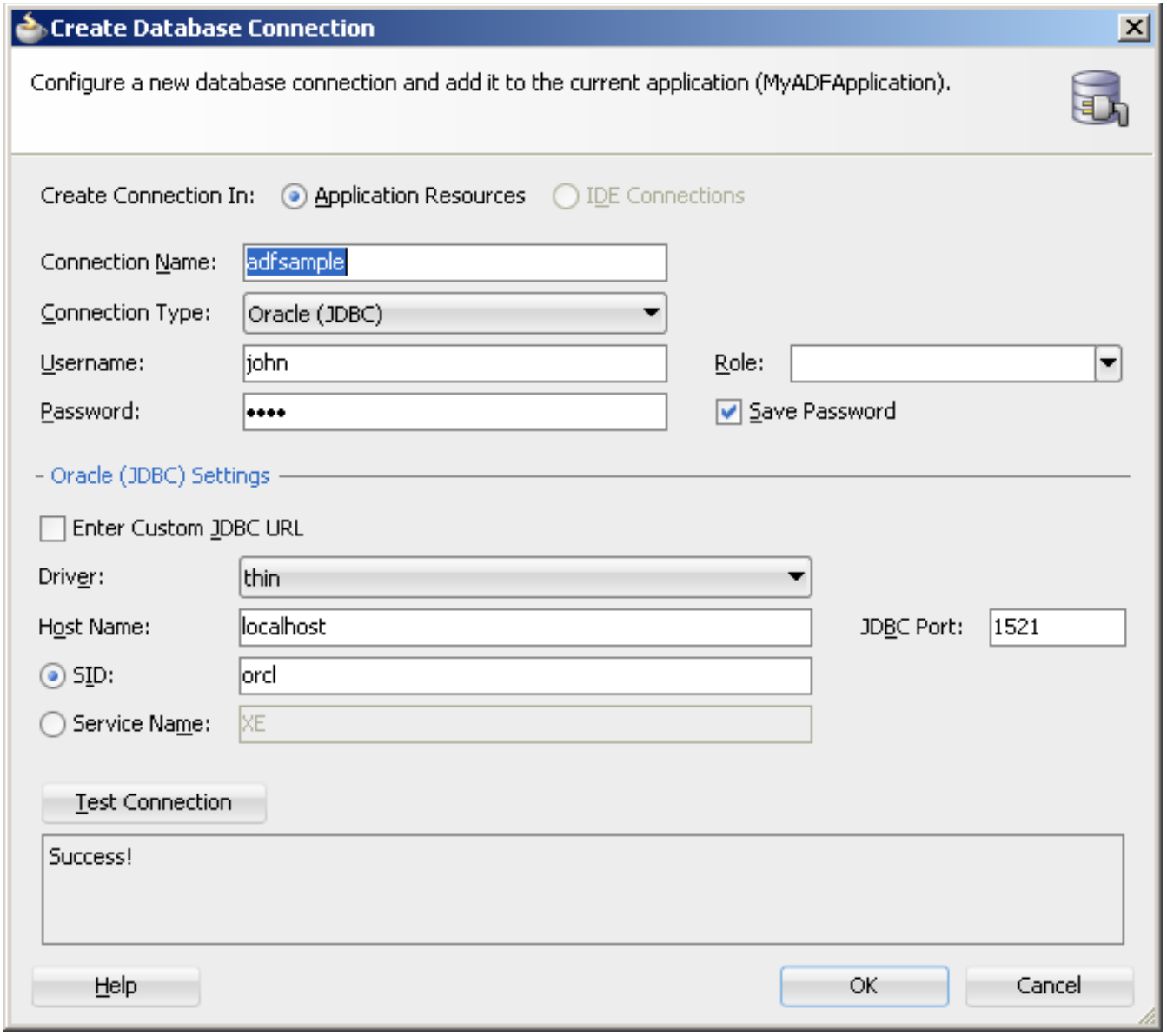


Choosing Database Connection

- Choose an existing Database Connection from the drop-down list or build a new one by clicking the green plus sign (Oracle client and tnsnames not required!)

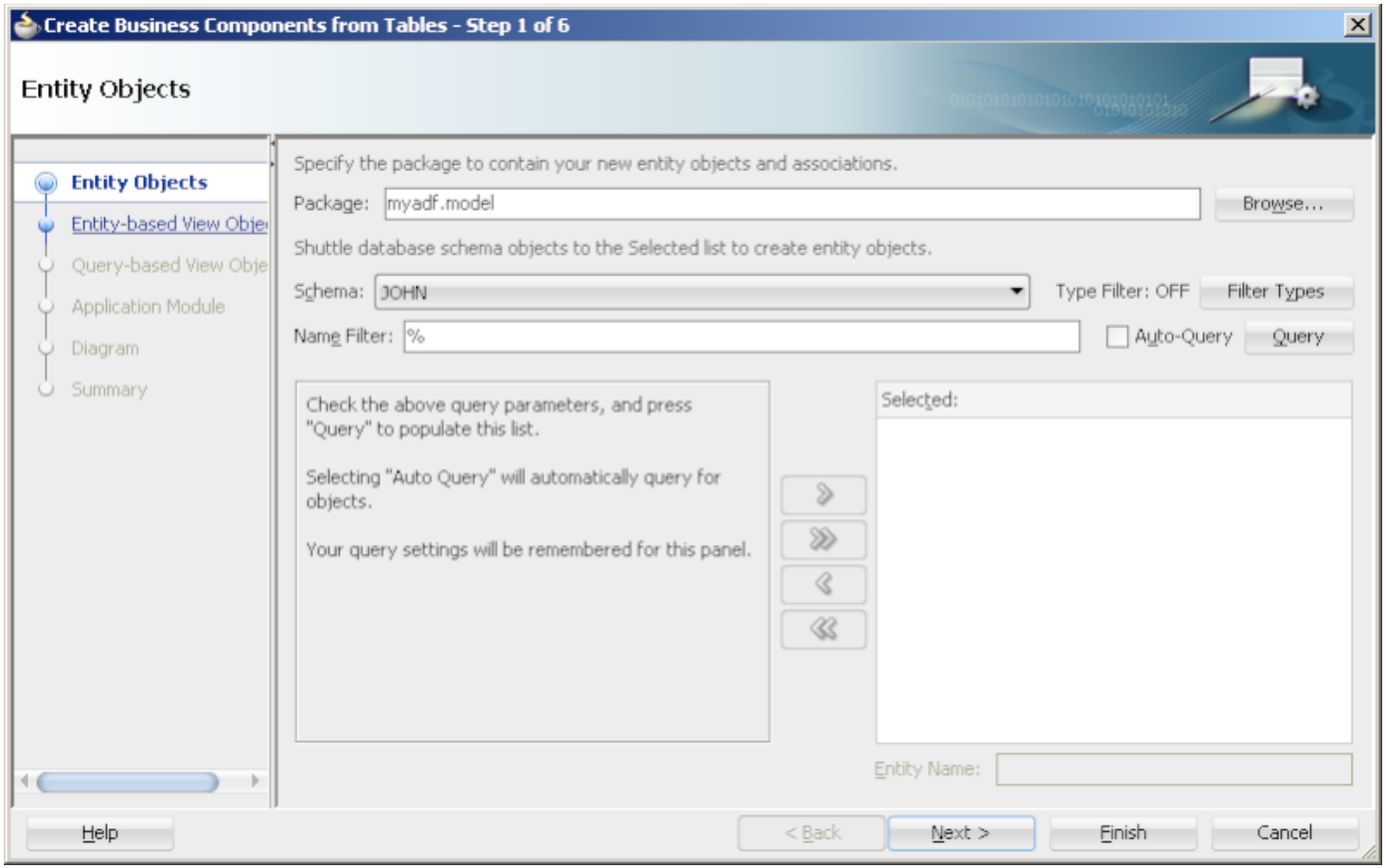


Create Database Connection



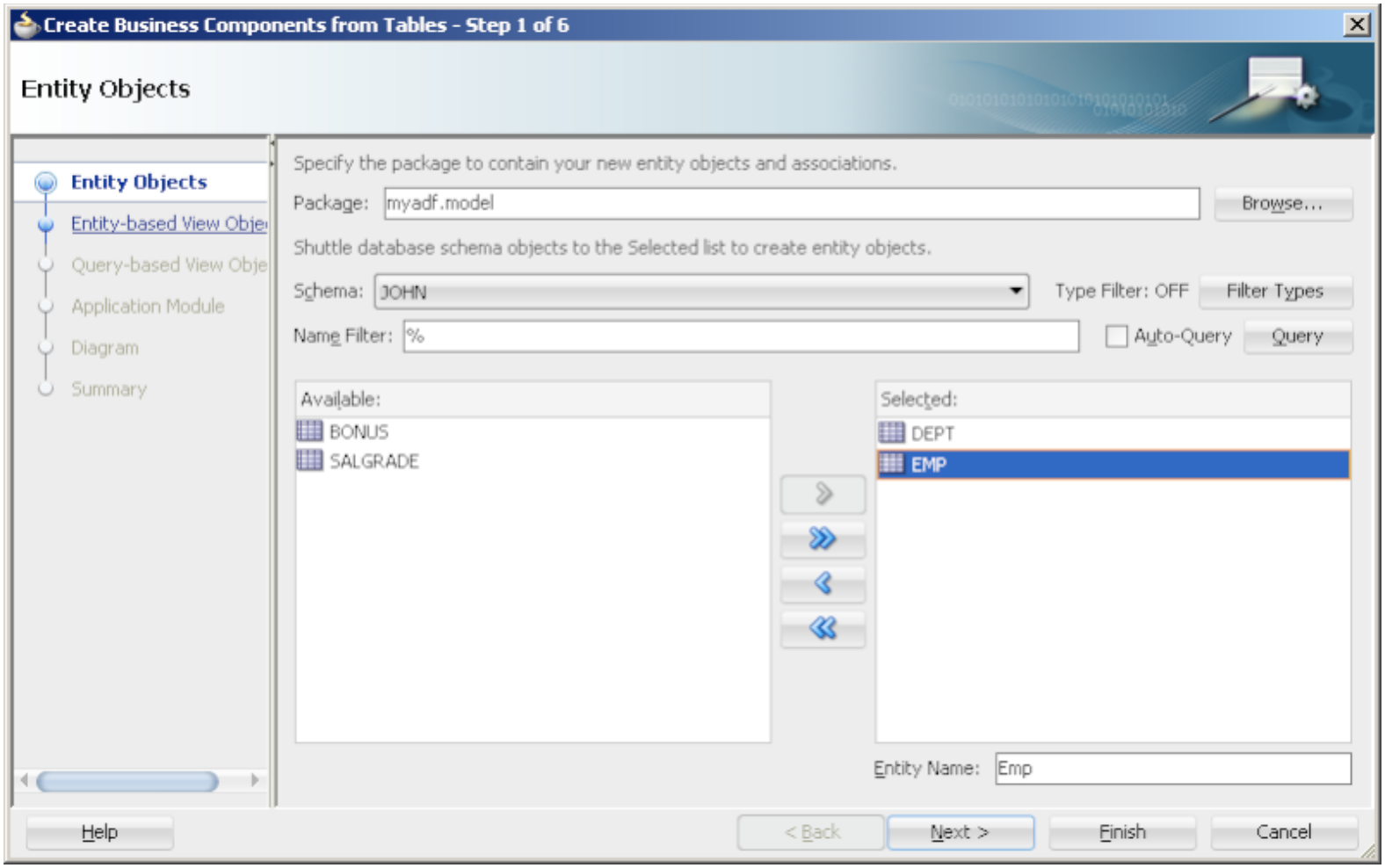
Create BC from Tables, 1

- Add, verify, or alter package name as desired; verify Schema to be used; modify filter (if desired) using SQL "LIKE" wild cards; click "Query" to view accessible database objects



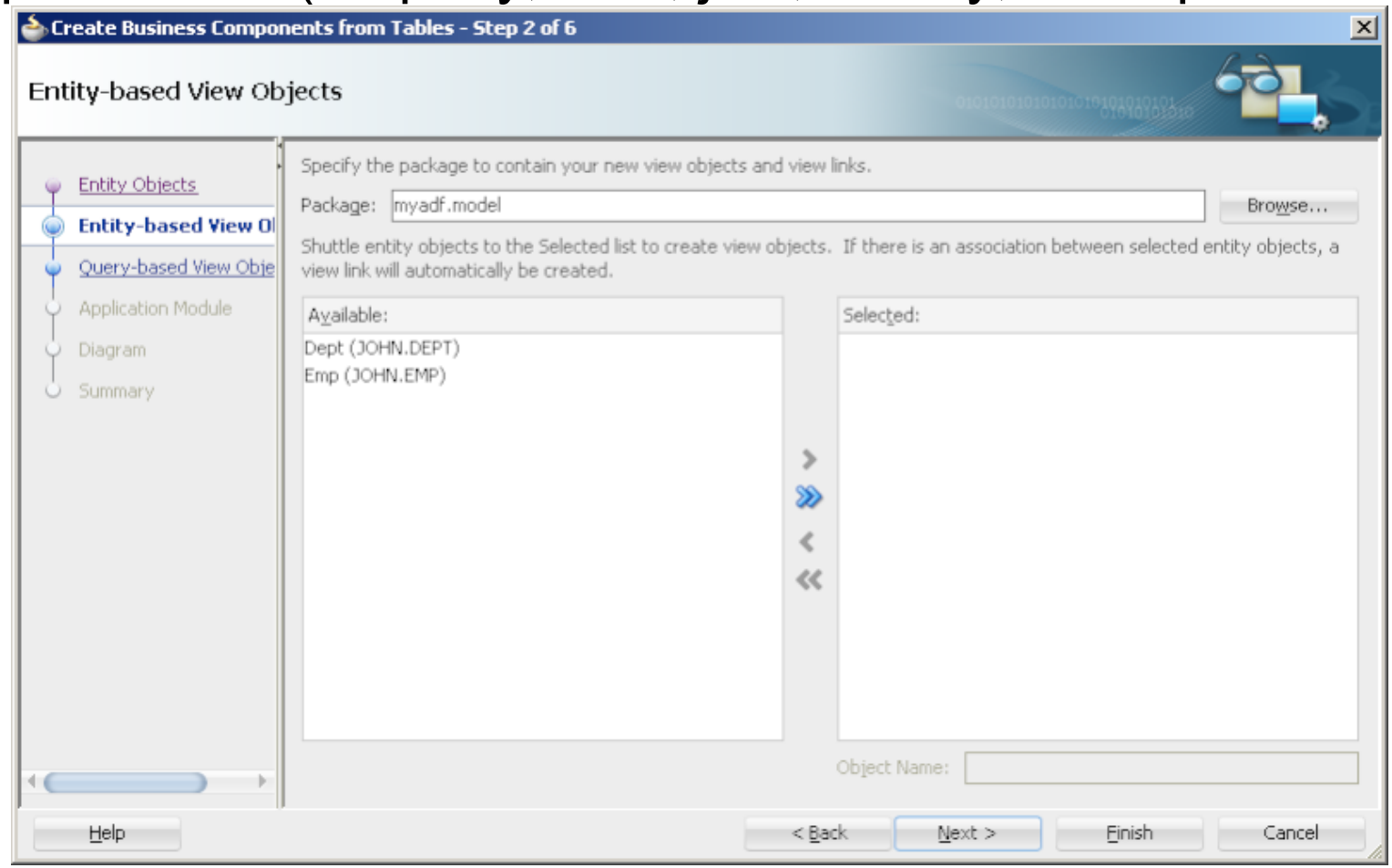
Create BC from Tables, 2

- Choose the tables and/or views to be part of the Entity Object and move them to the "Selected" side of the wizard



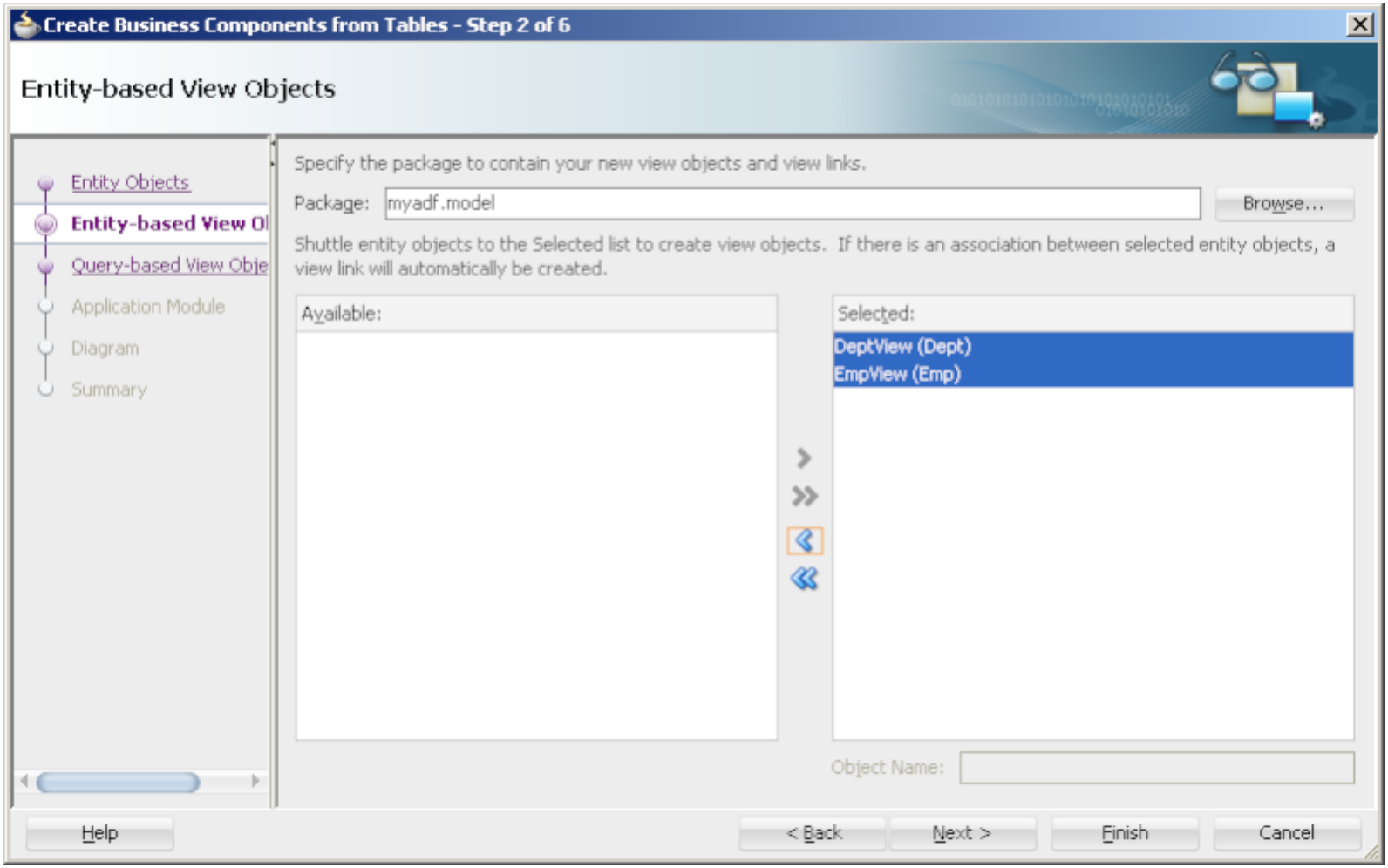
Create BC from Tables, 3

- After creating Entity Objects; the wizard offers to create Updateable Entity-based View Objects representing the output of SQL (to query, filter, join, modify, or sequence data)



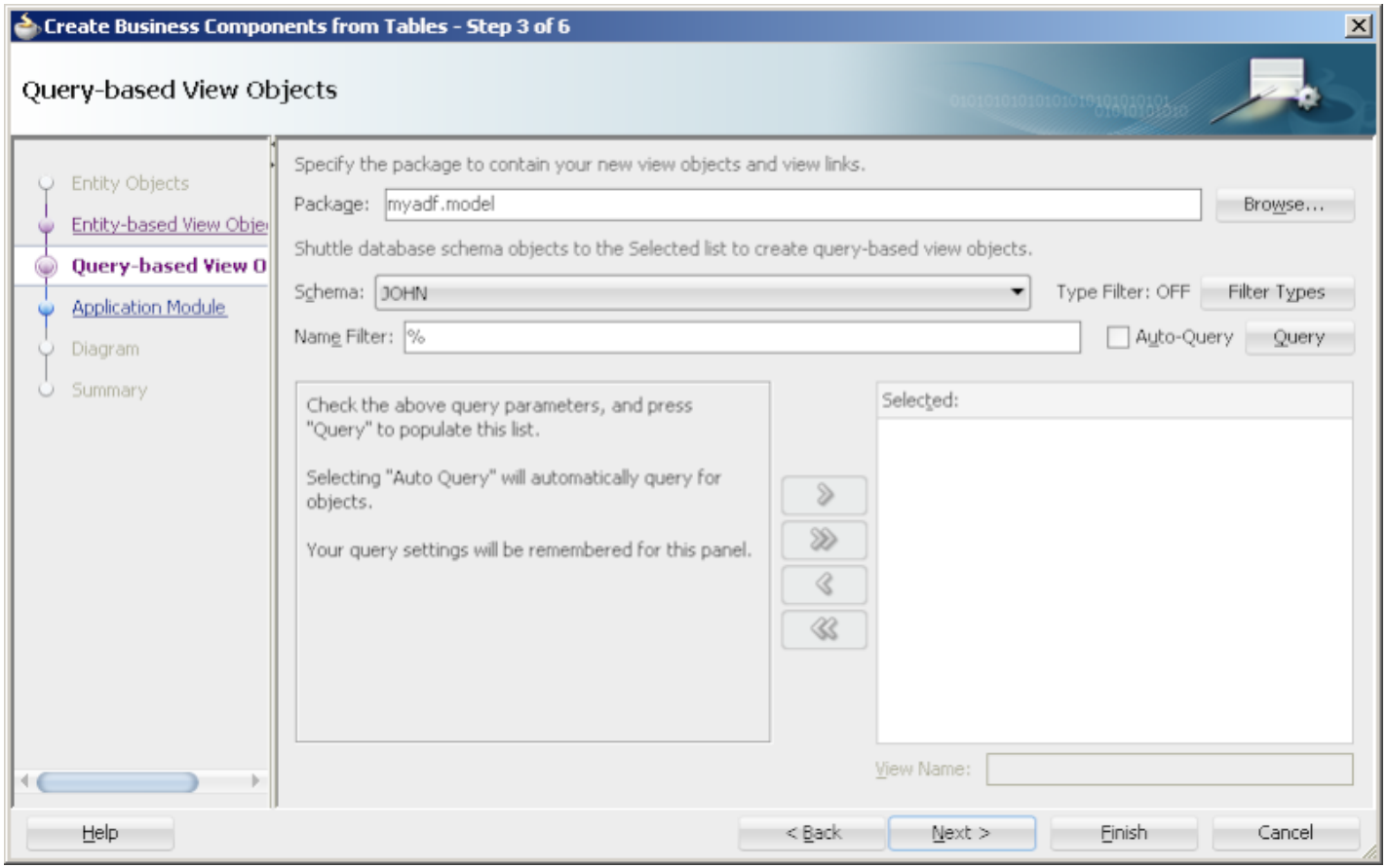
Create BC from Tables, 4

- Select Entity Objects to be used by the view being created; move them to the "Selected" side of panel



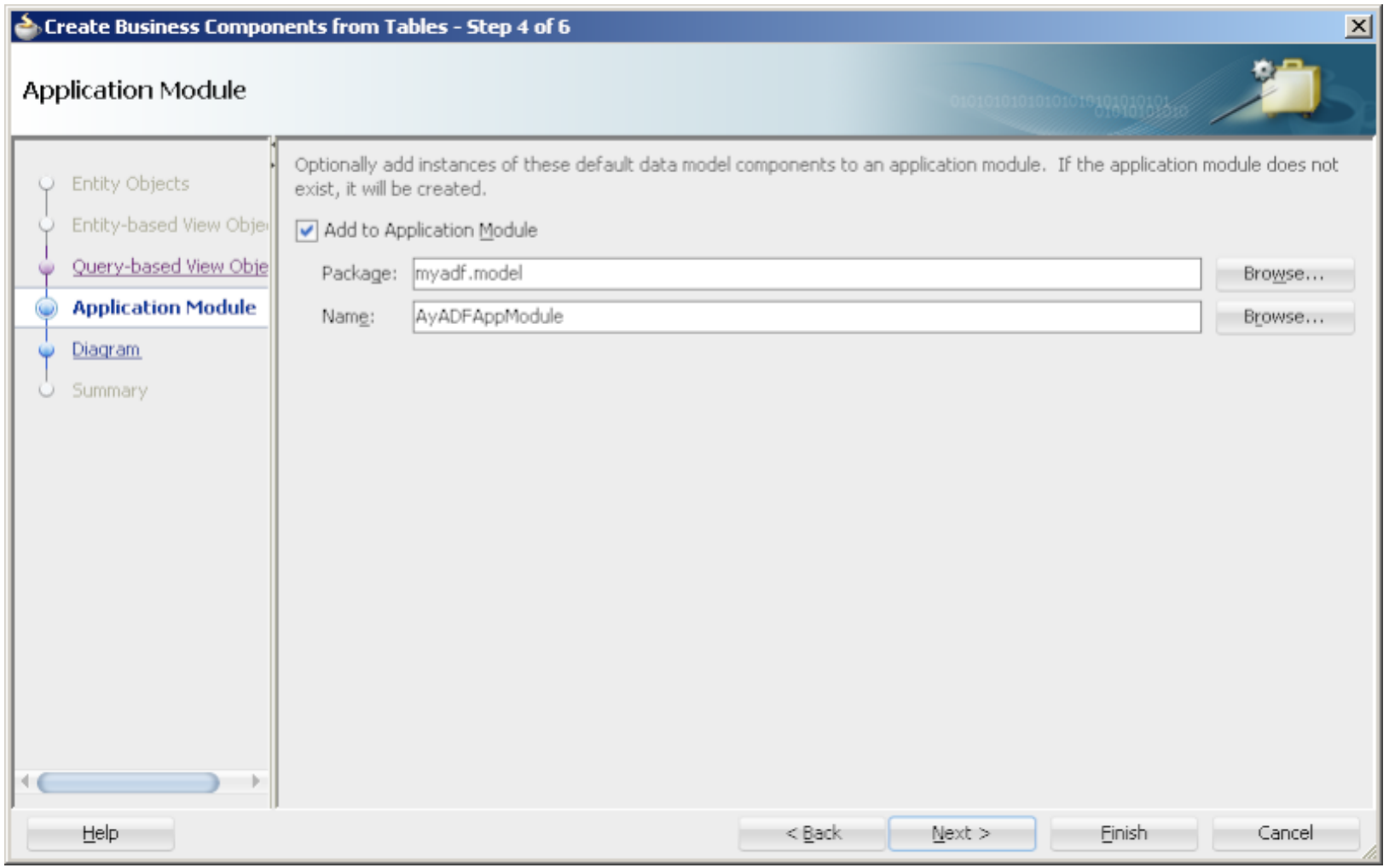
Create BC from Tables, 5

- After creating Updateable Entity-Based View Objects; the wizard goes on to create Read-Only Query-based View Objects (used for List-of-Values)



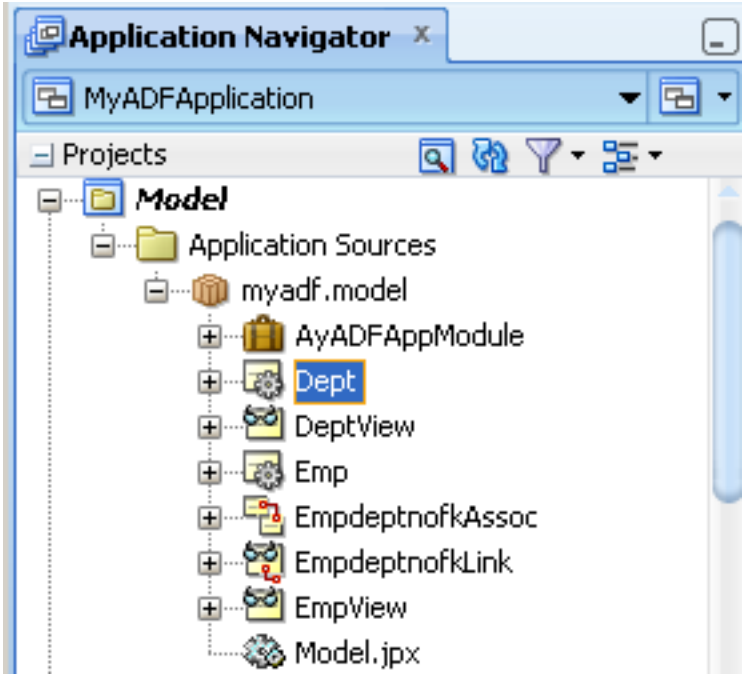
Create BC from Tables, 6

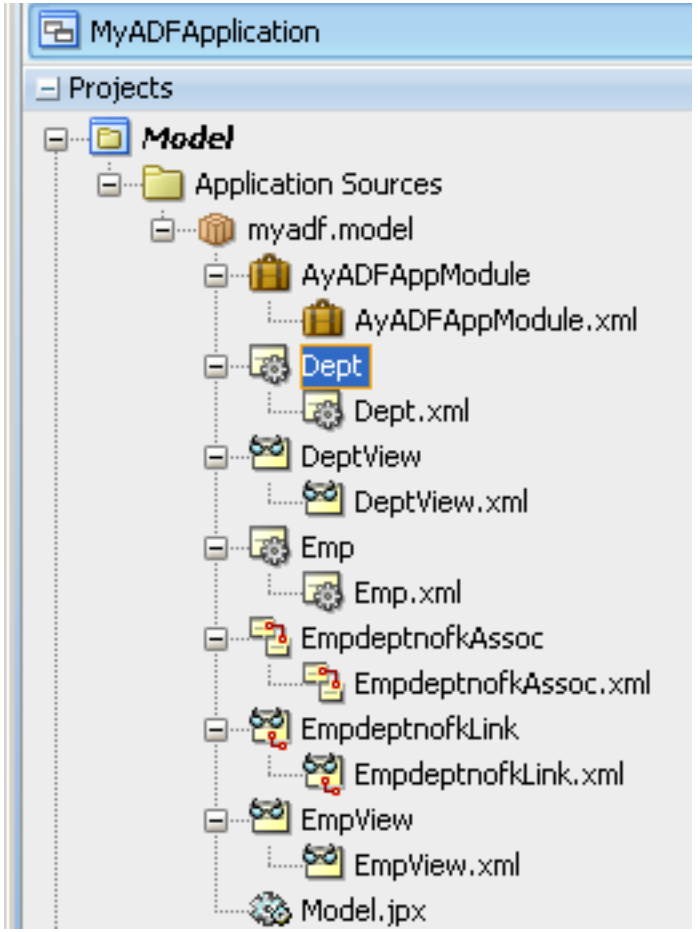
- Name the Application Module and click Finish



Business Component Files

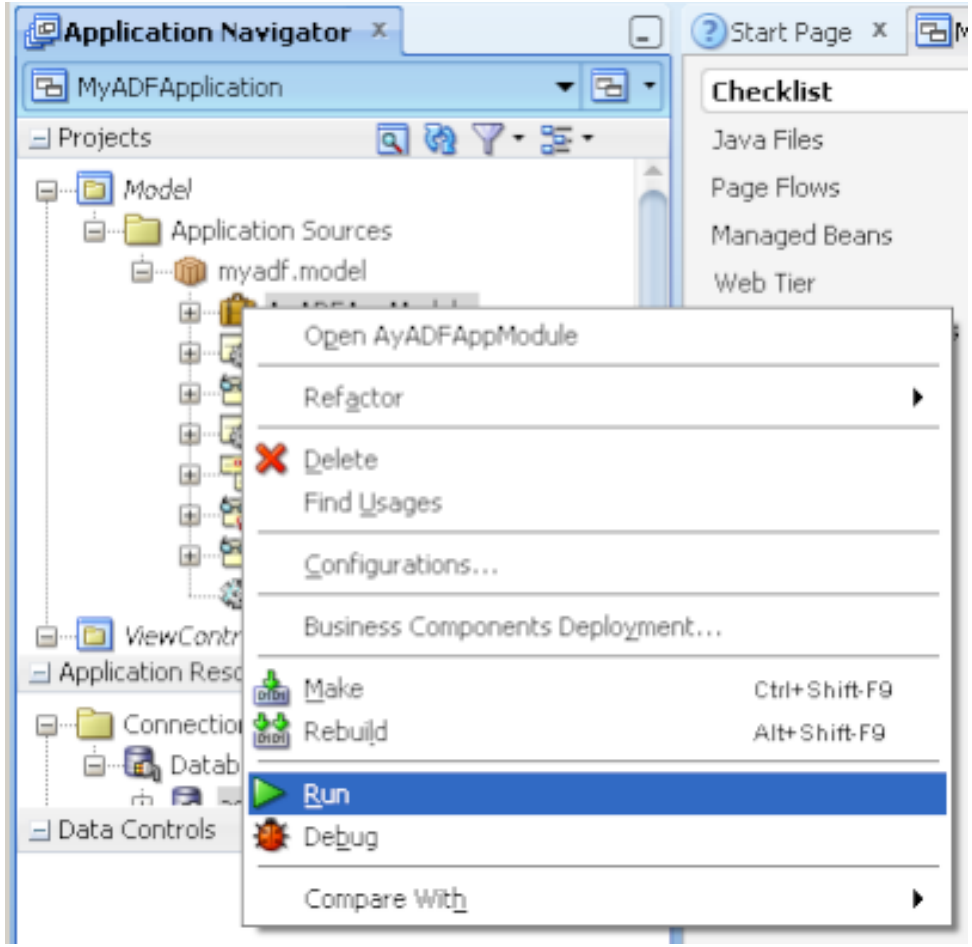
- Note the use of XML to declaratively support ADF BC





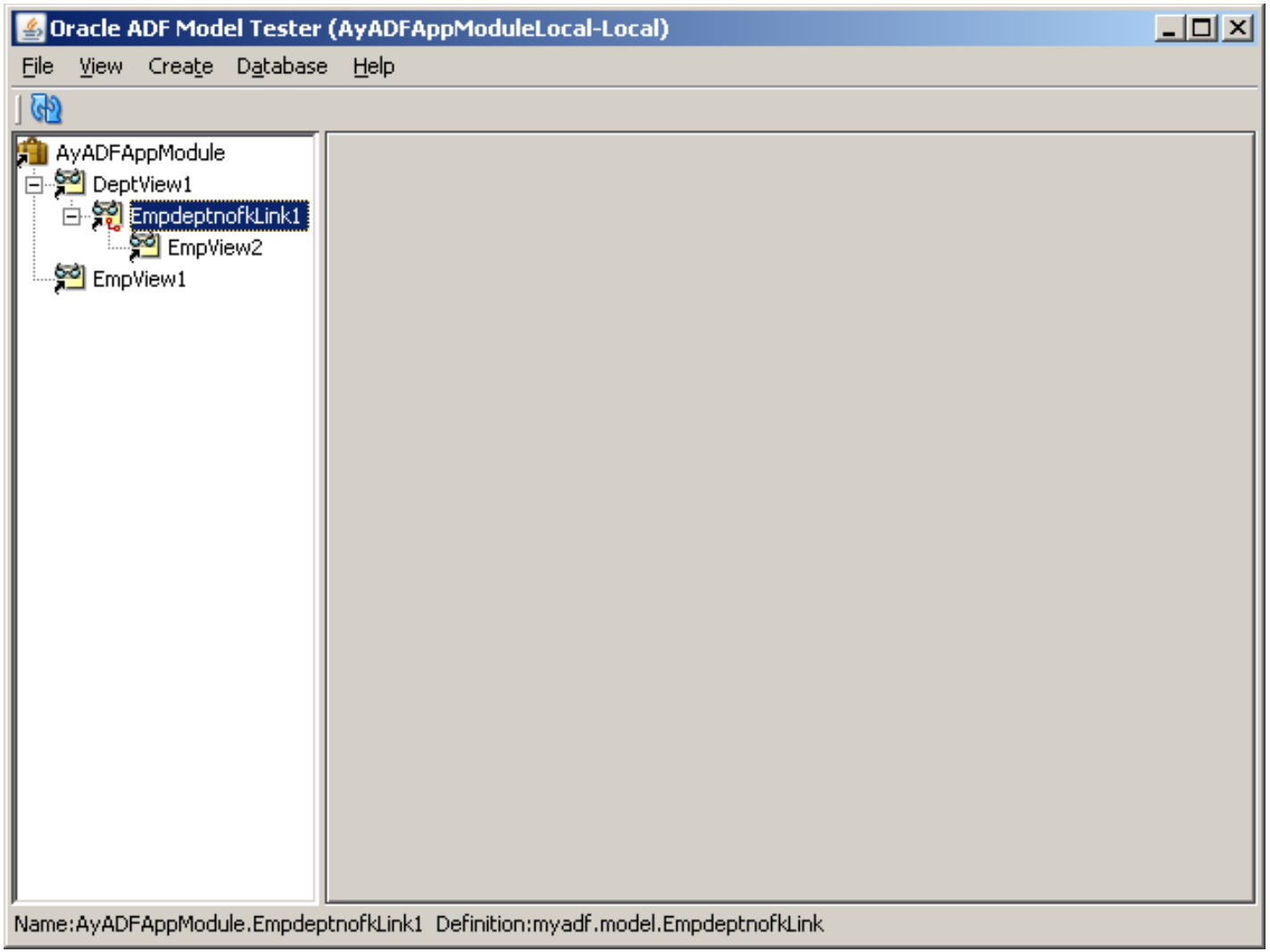
Business Component Browser

- JDeveloper provides a tool to “browse” ADF BC Application Module objects graphically; using the Application Navigator, find the Application Module to be viewed; right-click and choose “Run” to start



Component Browser Choices

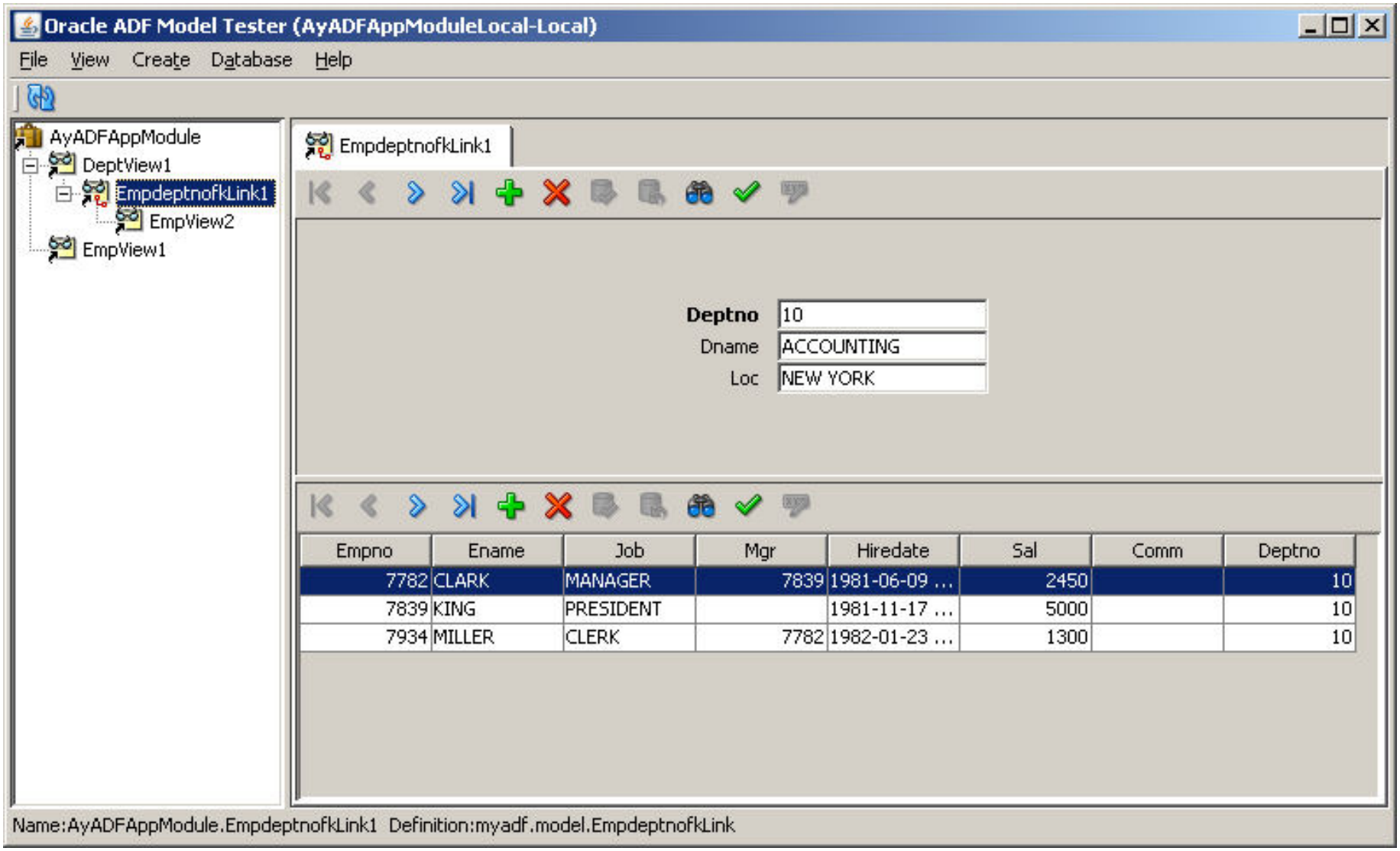
- Choose the Business Component to be tested



Component Browser – Display, 1

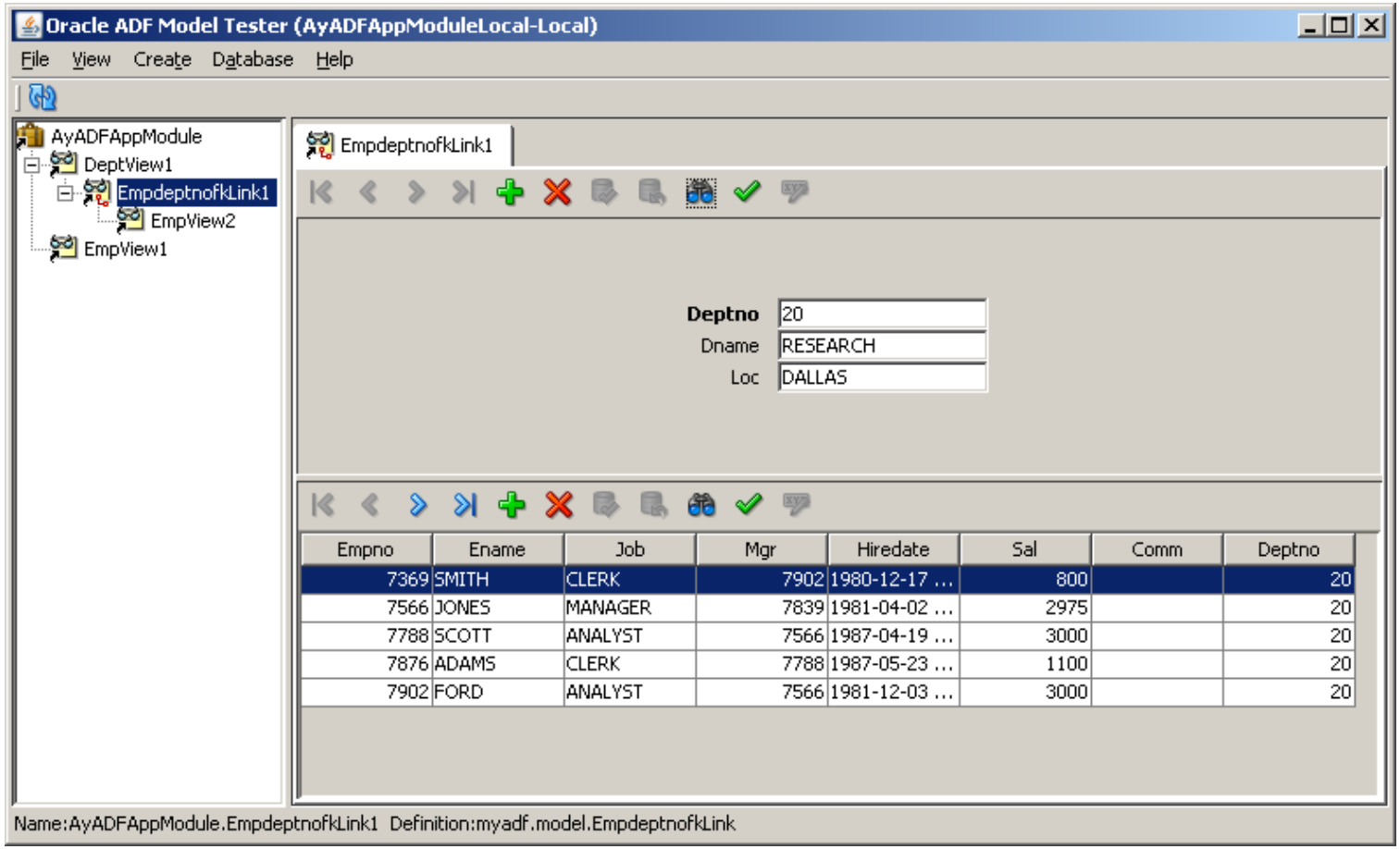
- Oracle's Business Component Browser displays data from the underlying database objects (screen should look familiar to Oracle Forms users)
- If referential keys are defined in the database (Primary Keys and Foreign Keys) the ADF BC Wizard automatically arranges the tables into a Master-Detail relationship

Component Browser – Display, 2



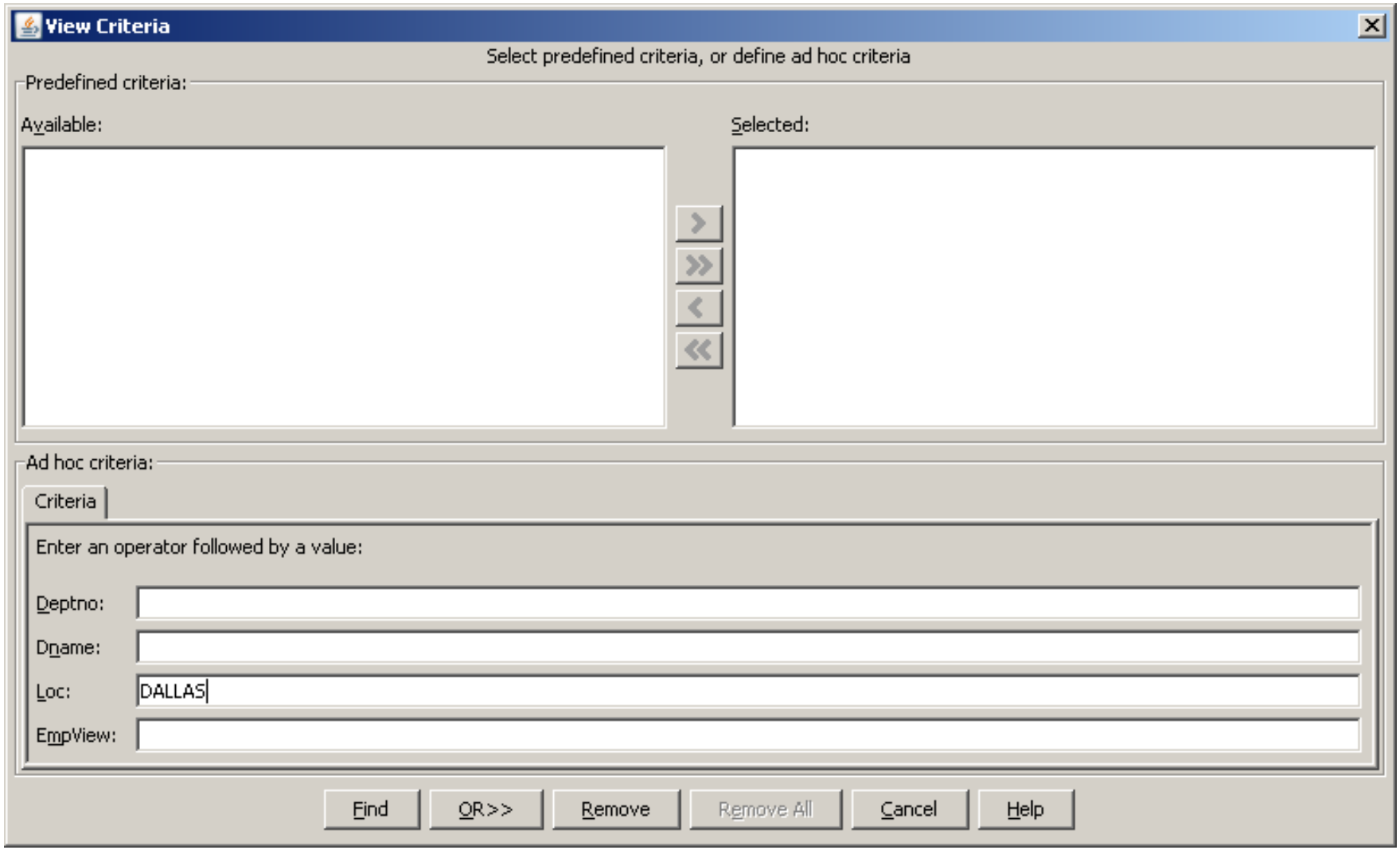
Searching Data

- Use the "Specify View Criteria" (Binocular) icon to Search

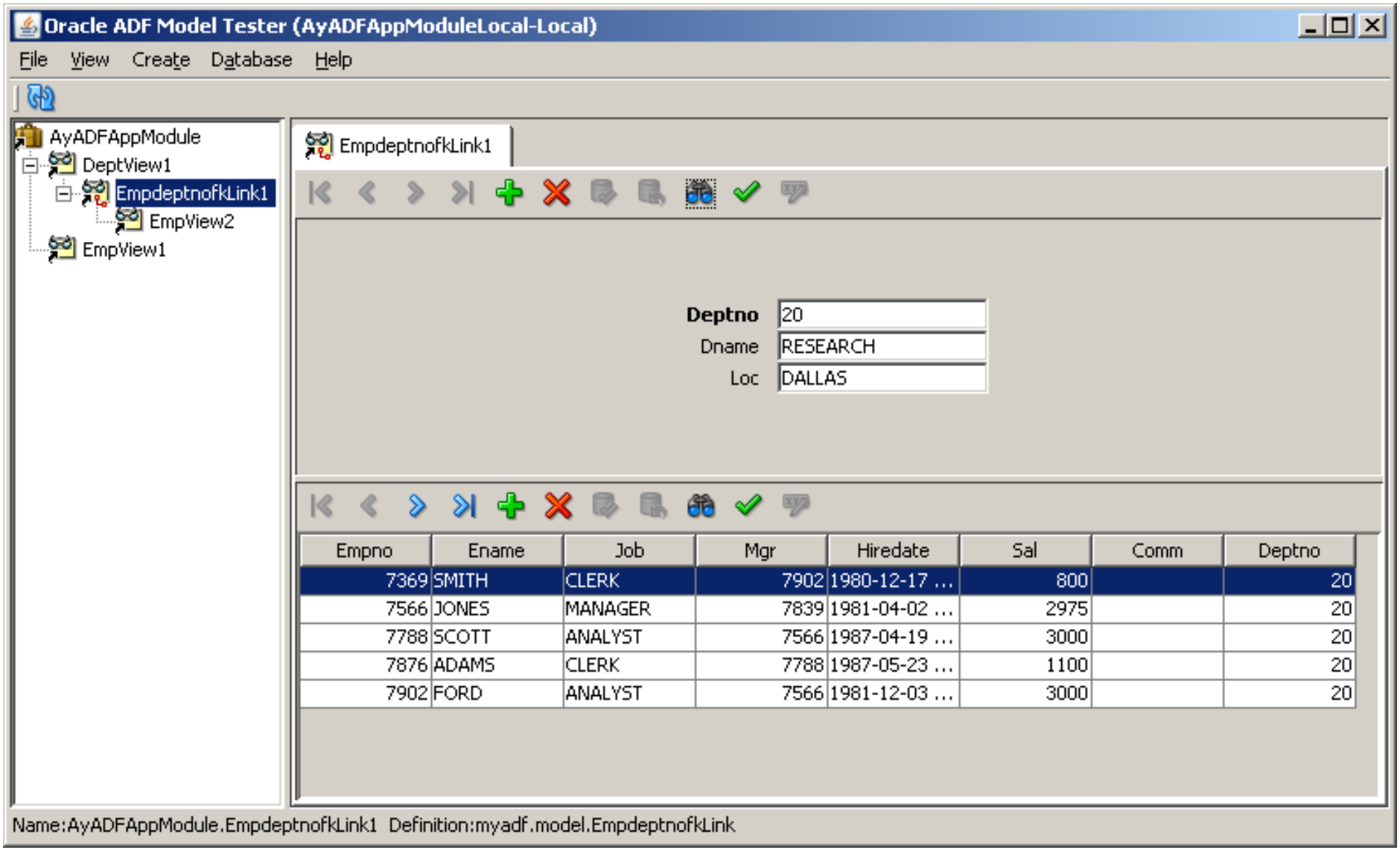


Search View Criteria

- Enter Search criteria and click “Find”

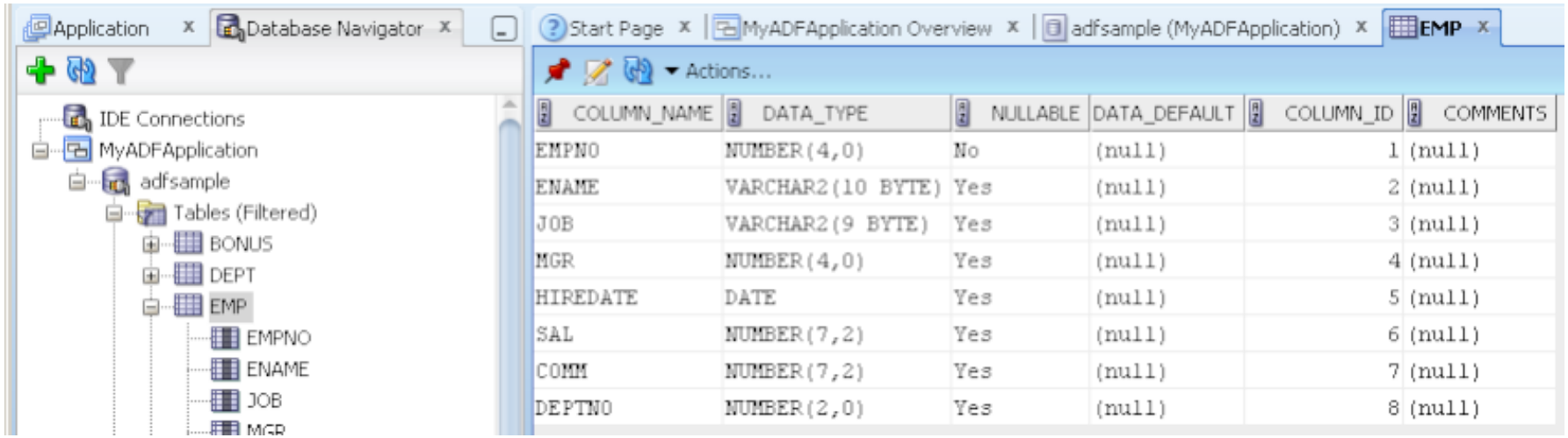


Search Results



Browsing Database Objects

- JDeveloper's Database Navigator allows browsing of database objects
  (parts of Oracle's SQL Developer tool have been incorporated into JDeveloper)

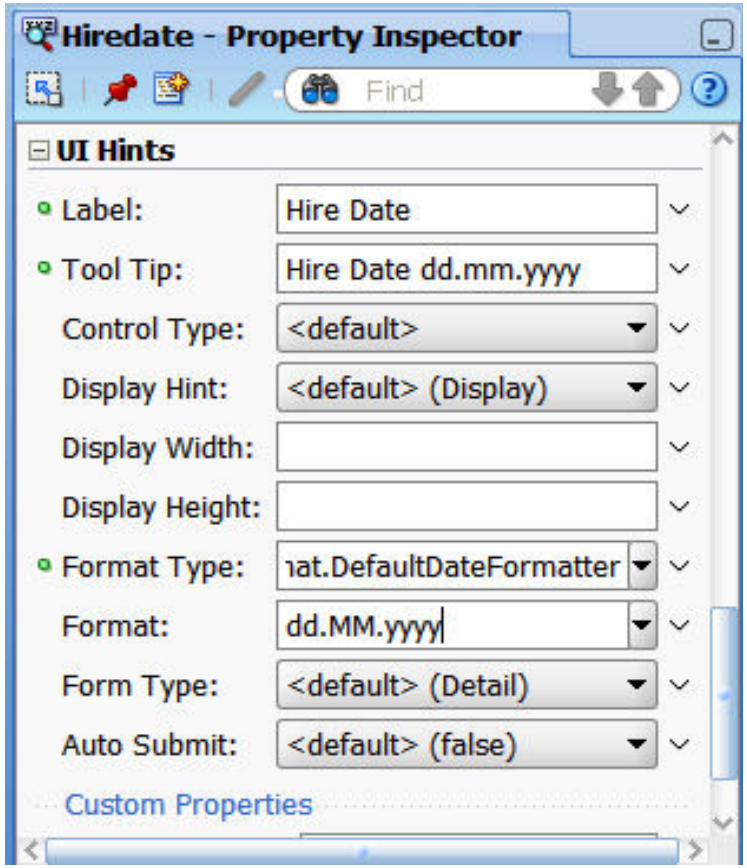


Modification of Application

- Once the initial Business Components are created in the application, it might be useful to:
  - Set default values
  - Define formatting
  - Validate data

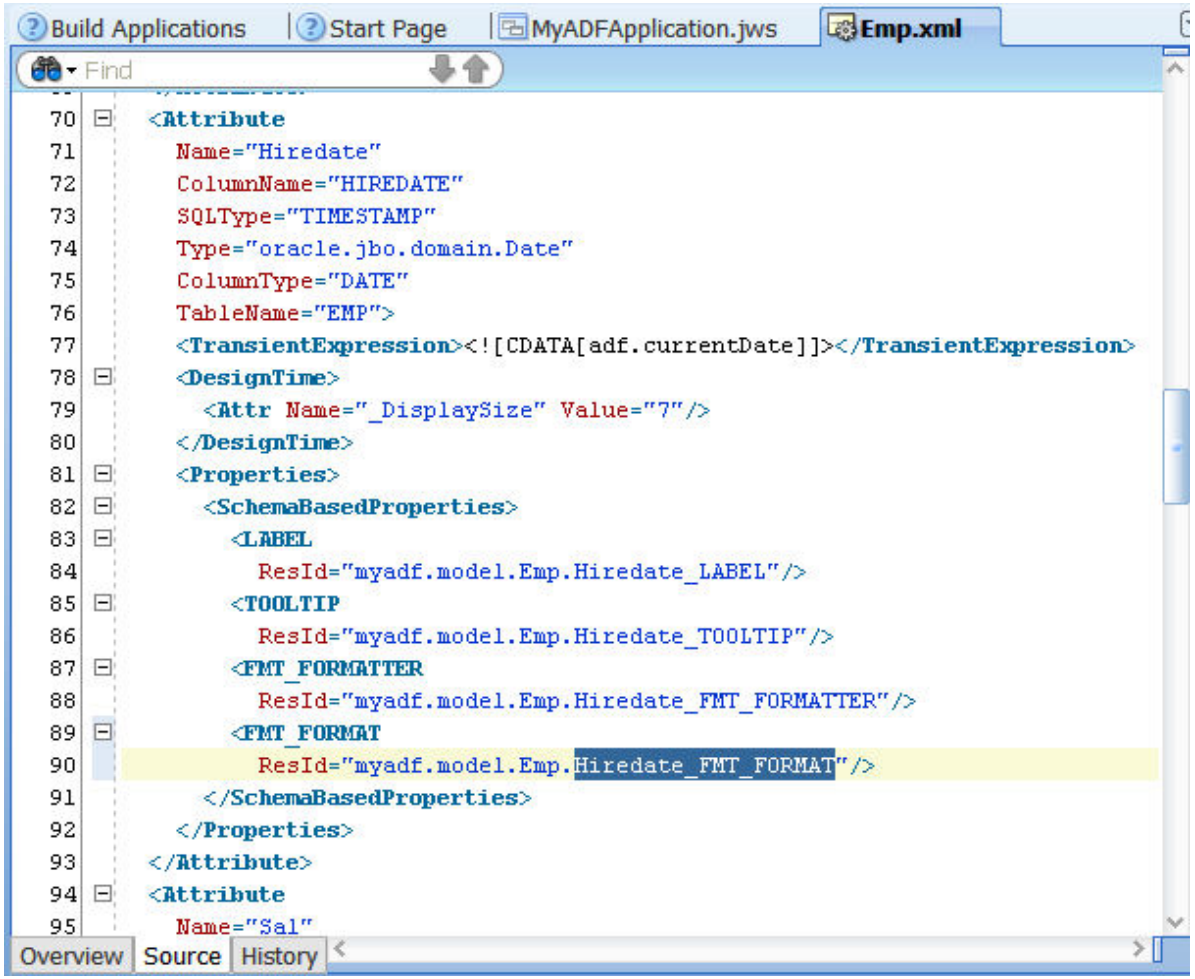
Object Properties

- Like Oracle Forms (and other 4GLs) properties are listed



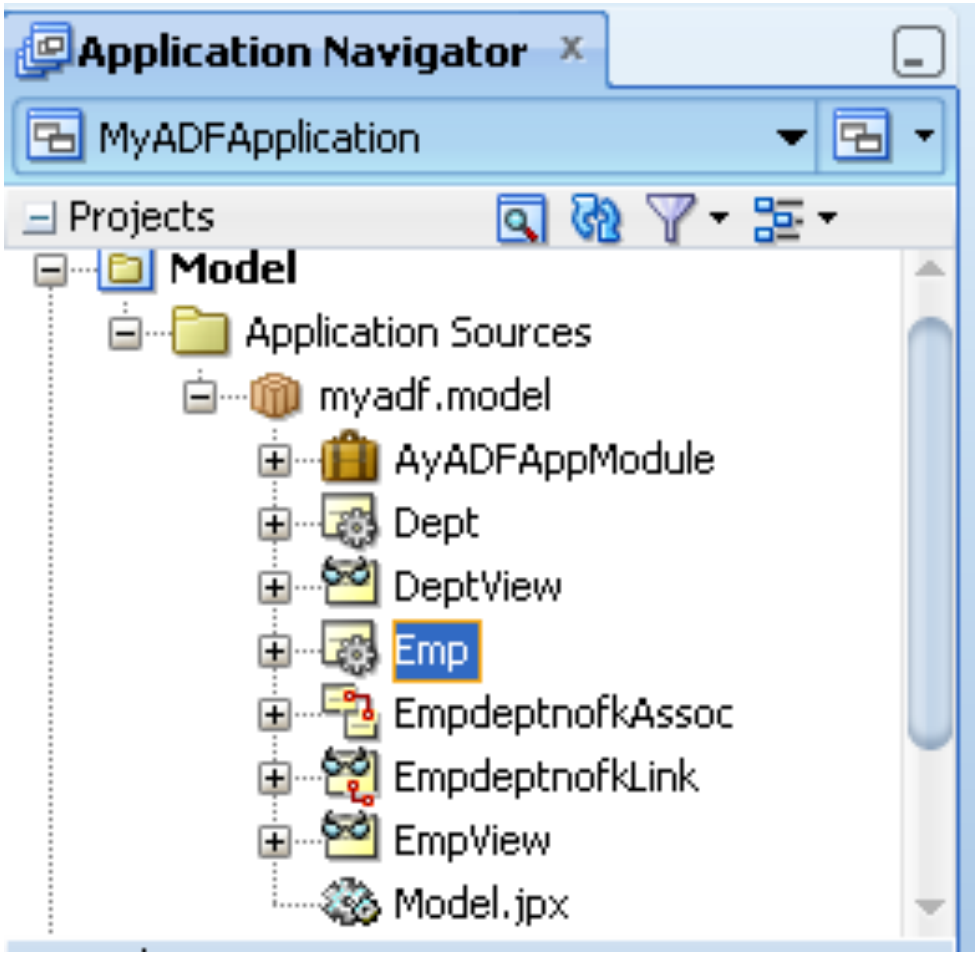
Properties in XML Files

- ADF uses XML files to store declared definitions

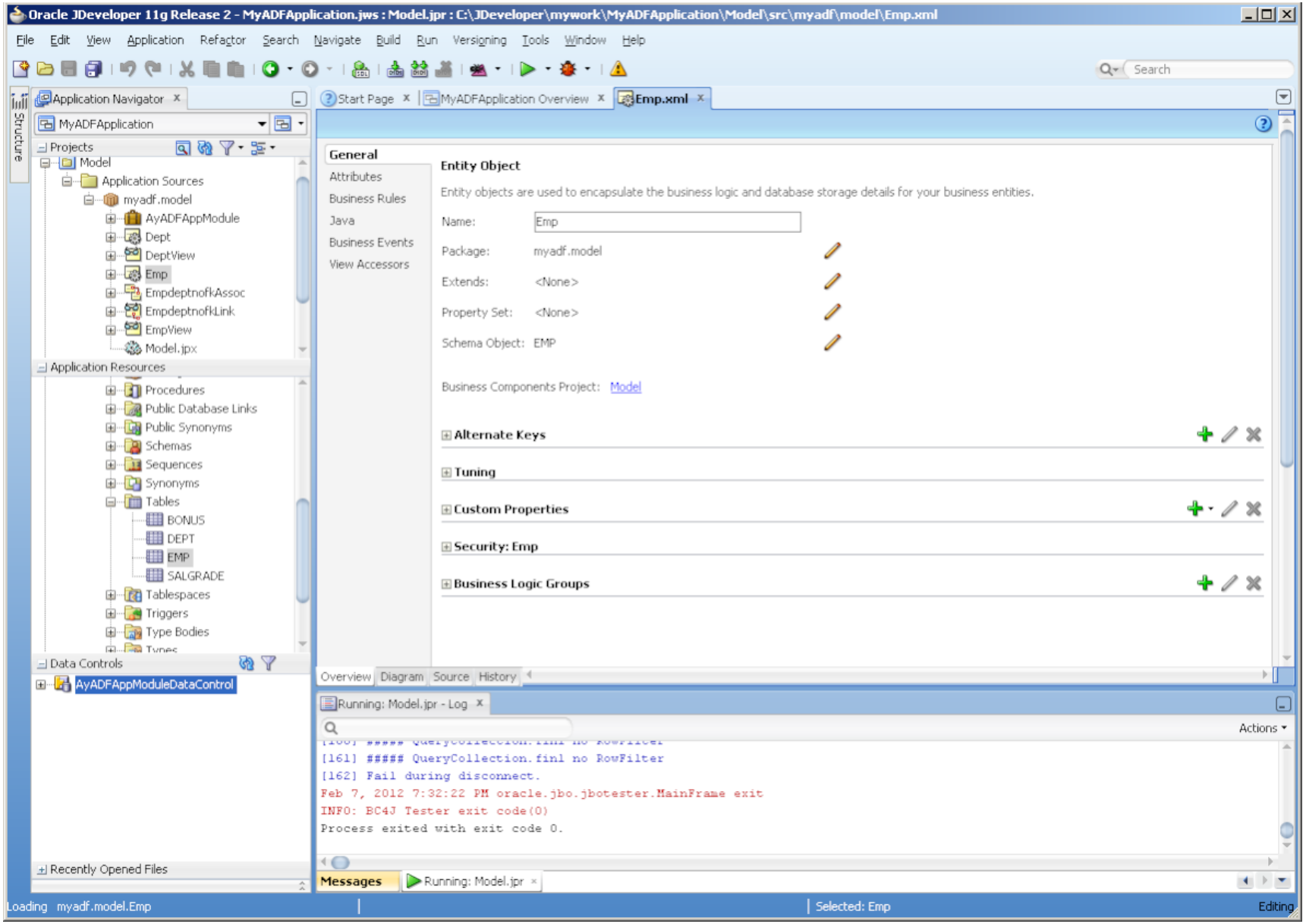


Modify Appearance & Formatting

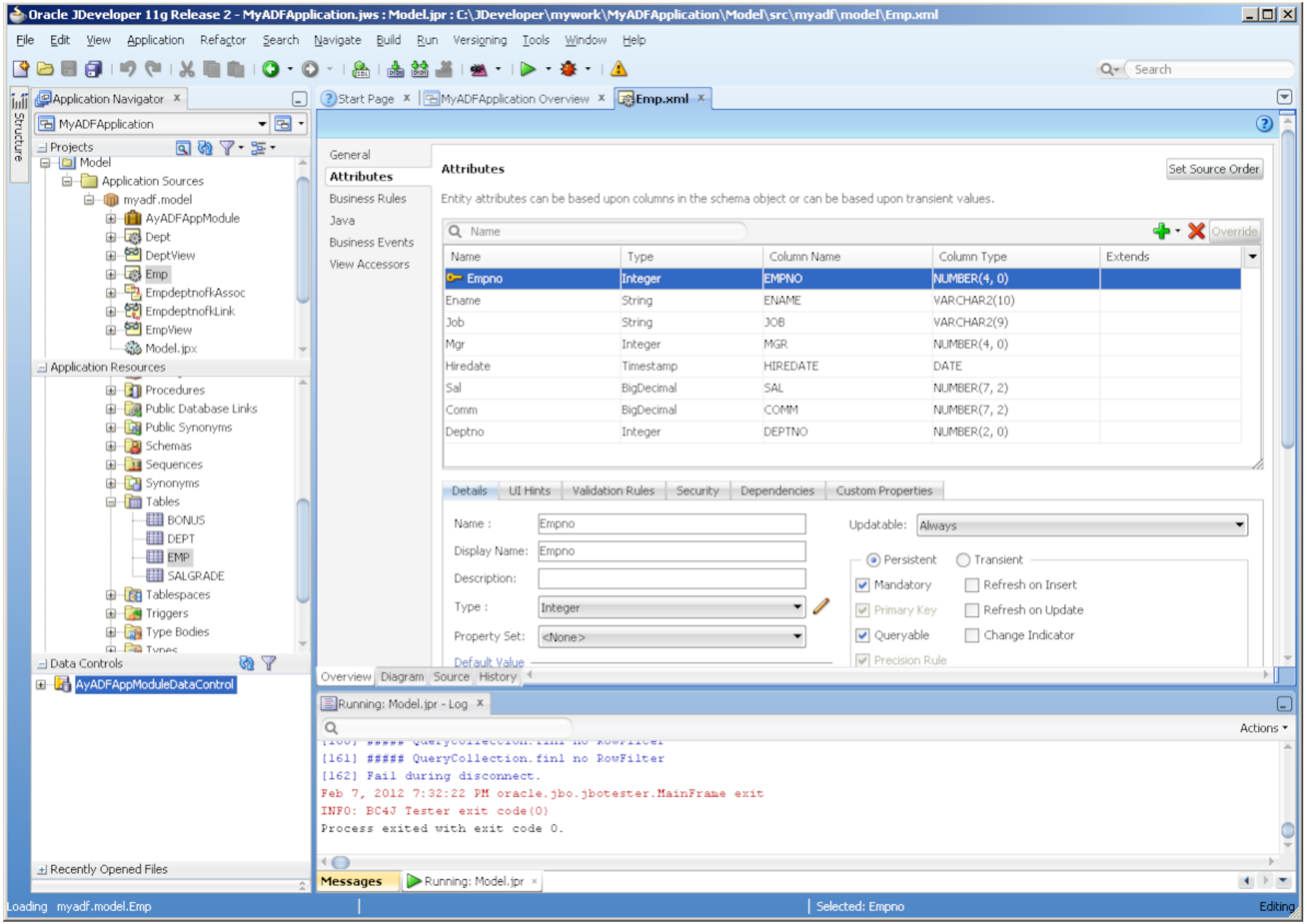
- Use JDeveloper to modify appearance of database column values by double-clicking an Entity Object



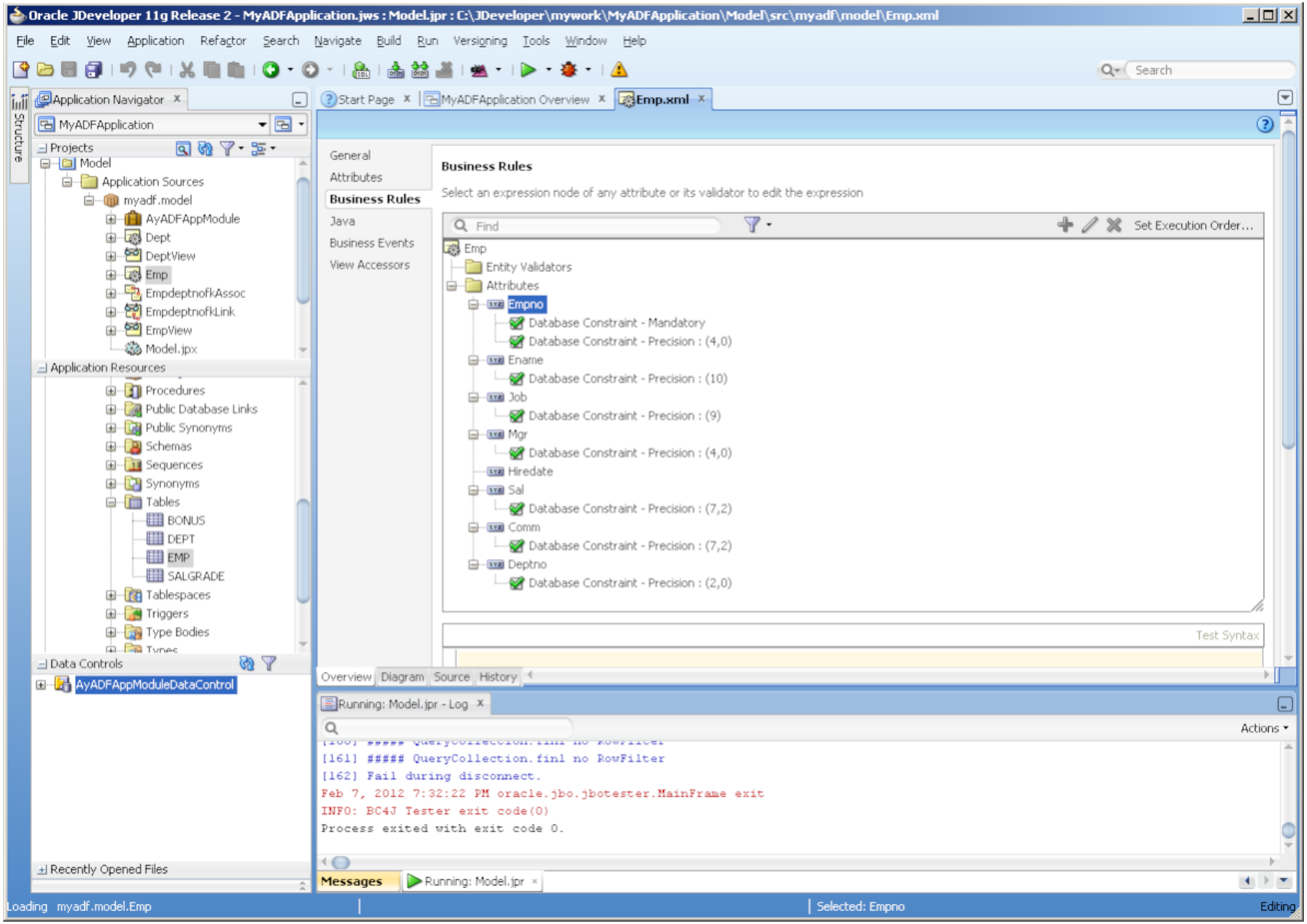
Entity Object Edit Panel



Entity Object Attributes



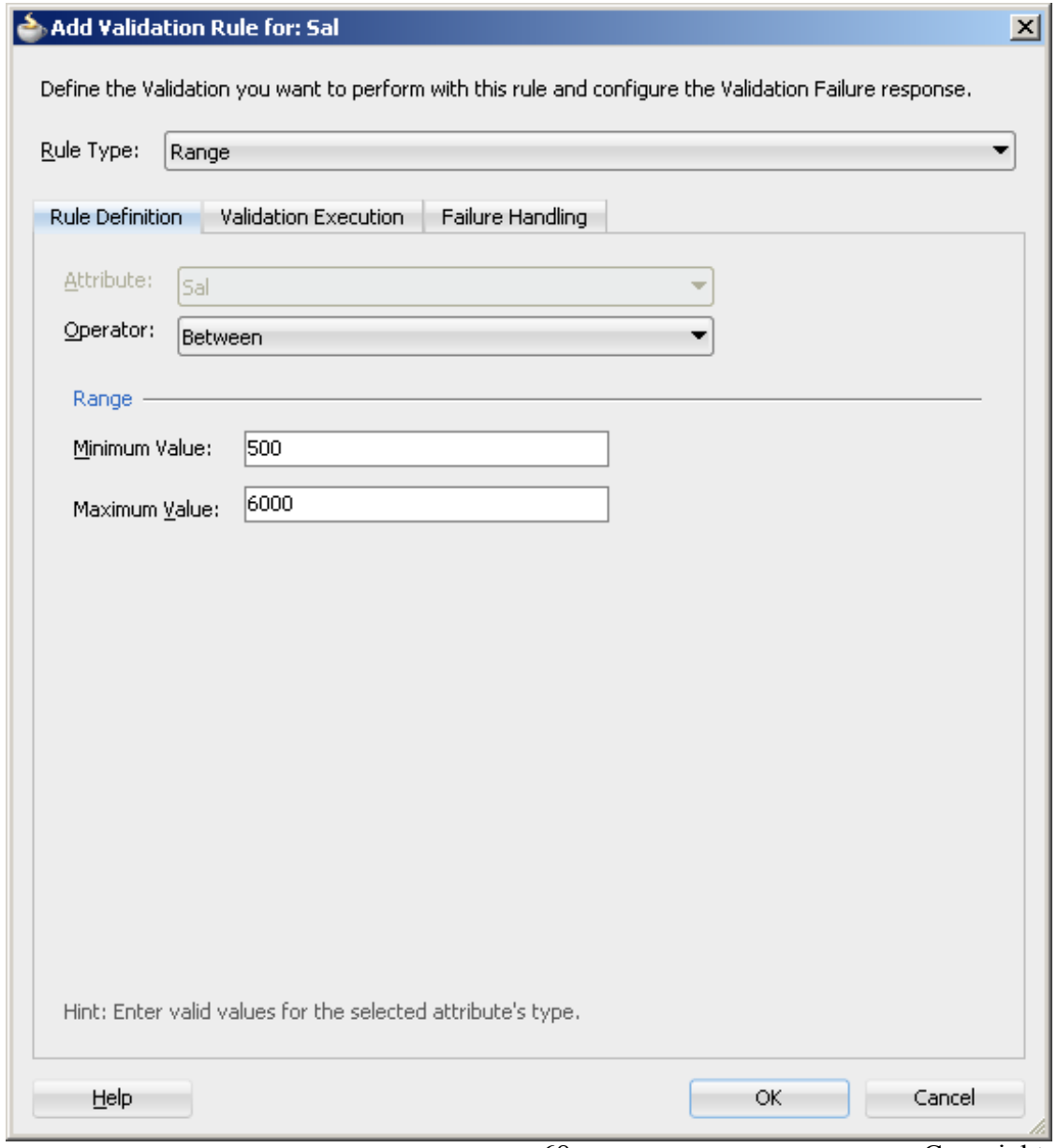
Entity Object Business Rules



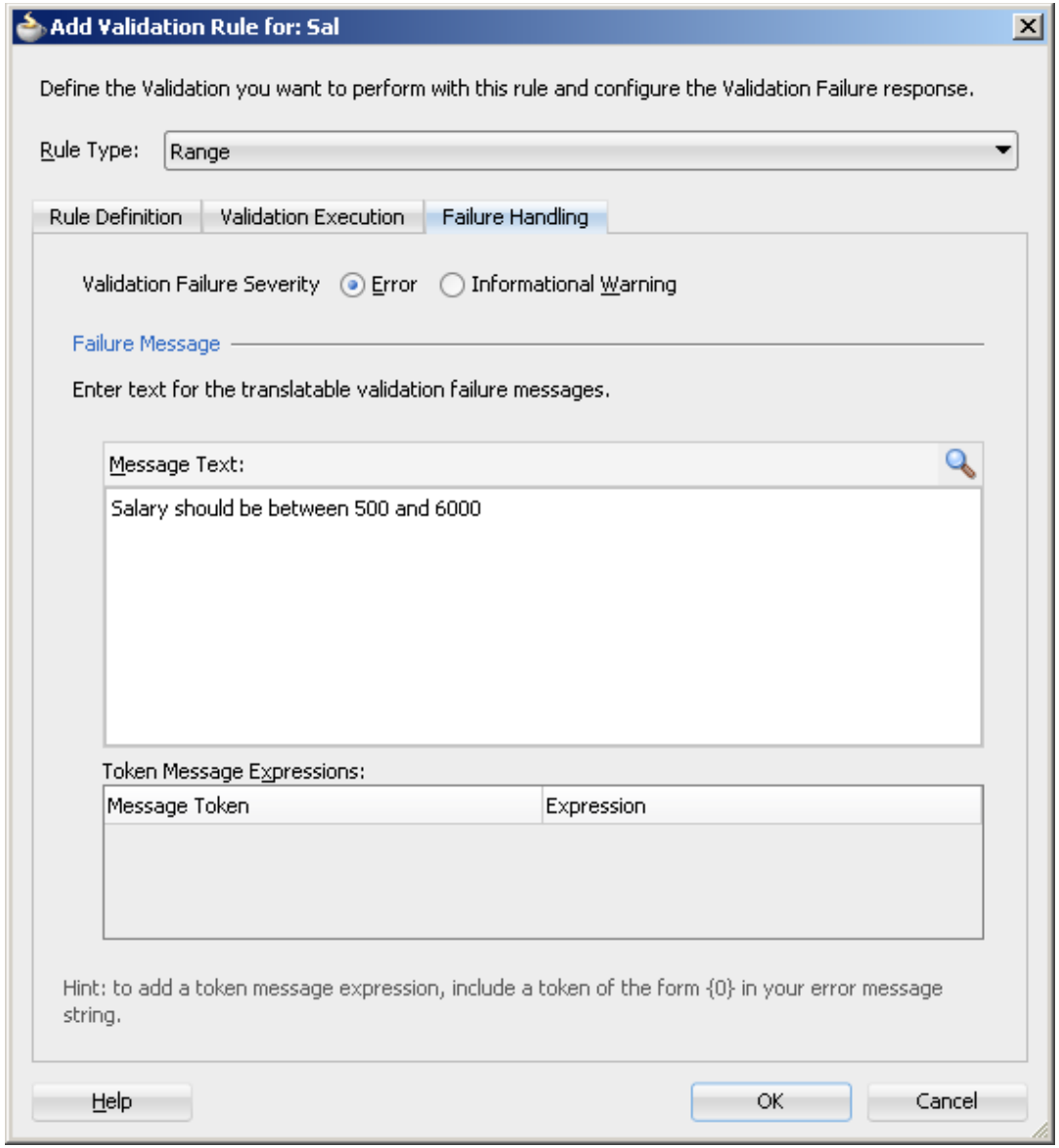
Validations and Business Logic

- Validations and Business Logic may be added including:
  - Client-side validation
  - Format masks
  - Default Values
  - Declarative Range (and other) Validation
  - CSS (Visual Attributes)
  - List of Values
  - Calculated field
  - Code Validation
  - Extensible for complex application validation
  - Transactional Triggers

Validation Rules

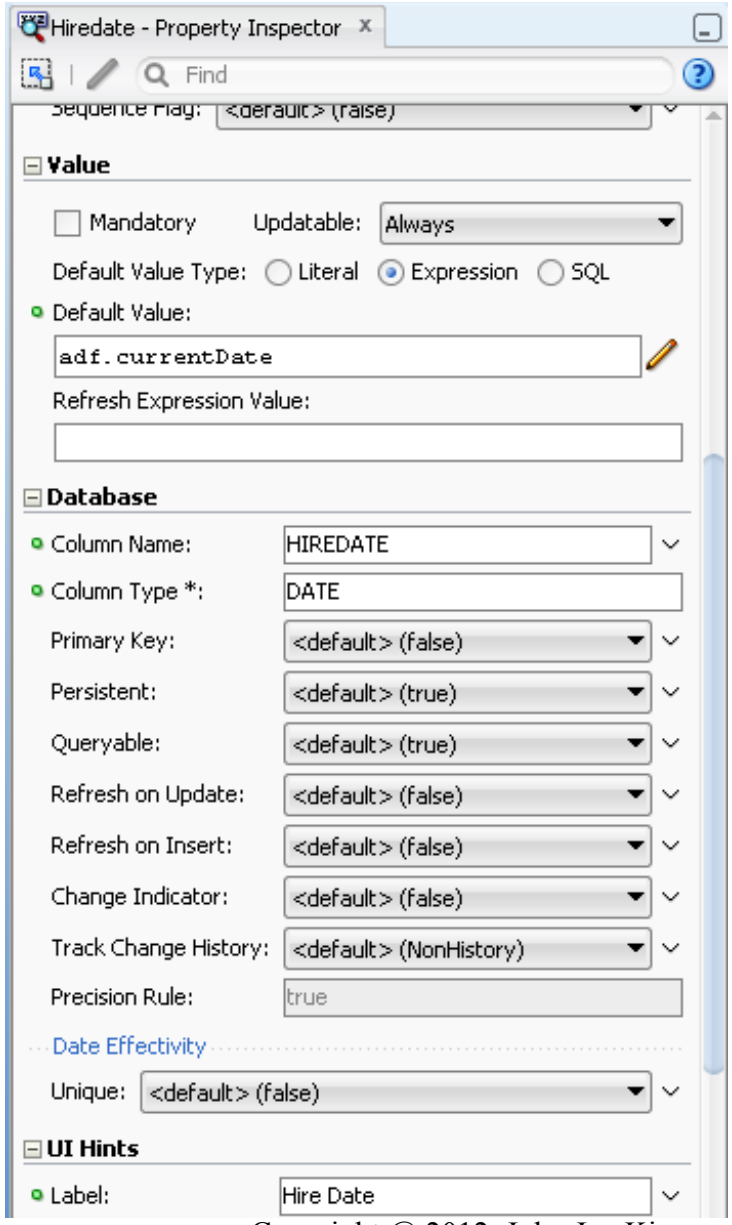


Validation Error Messages



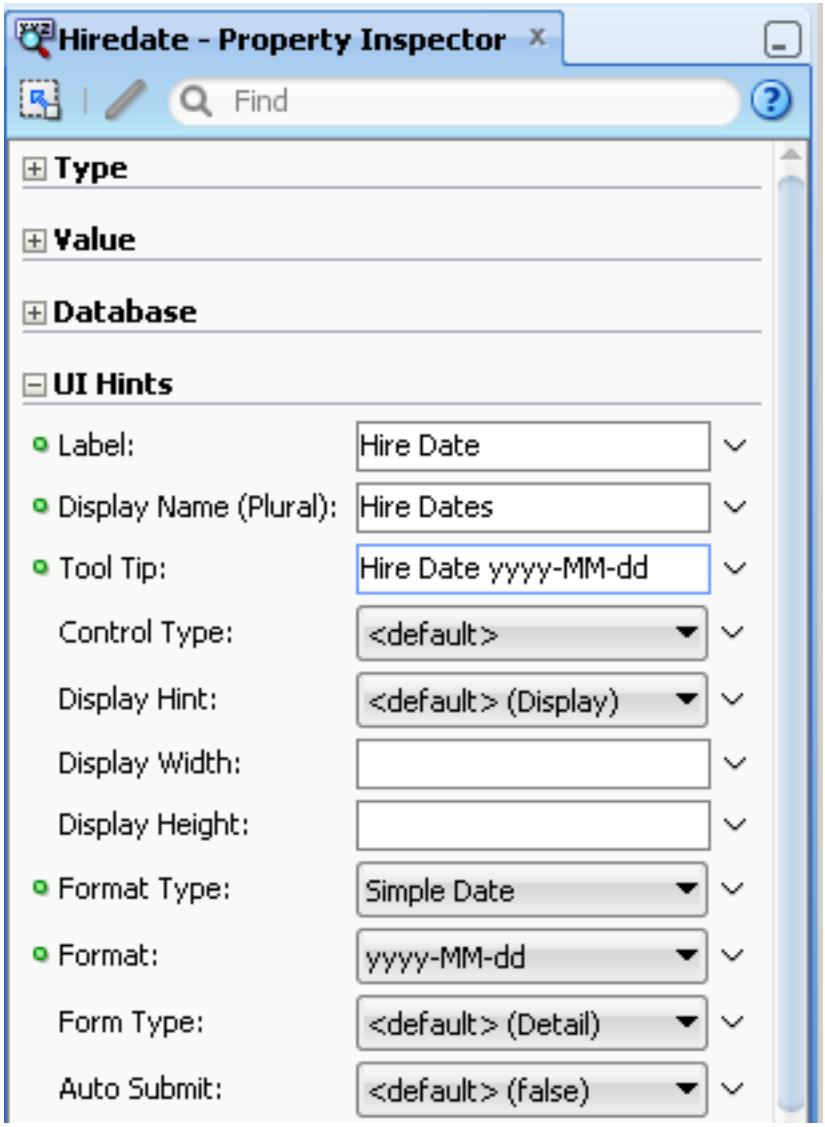
Attribute Defaults

- Using the Property Inspector, open the "Value" properties and set the default value (in this case "adf.currentDate" using ADF's "Groovy" support)



Attribute Formatting

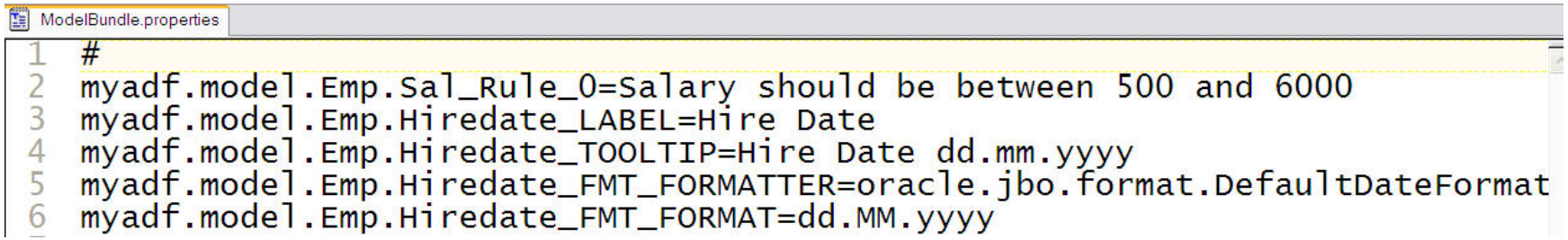
- Use an Attribute's Property Palette “UI Hints” section to control formatting, label, tool tip, etc… (note this formatting uses Java SimpleDateFormat options)



What Does the XML Look Like?

```
<Attribute
  Name="Hiredate"
  ColumnName="HIREDATE"
  SQLType="DATE"
  Type="oracle.jbo.domain.Timestamp"
  ColumnType="DATE"
  TableName="EMP"
  DefaultValue="adf.currentDate">
  <Properties>
    <SchemaBasedProperties>
      <LABEL
        ResId="myadf.model.Emp.Hiredate_LABEL"/>
      <LABEL_PLURAL
        ResId="myadf.model.Emp.Hiredate_LABEL_PLURAL"/>
      <TOOLTIP
        ResId="myadf.model.Emp.Hiredate_TOOLTIP"/>
      <FMT_FORMATTER
        ResId="myadf.model.Emp.Hiredate_FMT_FORMATTER"/>
      <FMT_FORMAT
        ResId="myadf.model.Emp.Hiredate_FMT_FORMAT"/>
    </SchemaBasedProperties>
  </Properties>
</Attribute>
```

Date Mask Properties File



Comparison to Oracle Forms

- In Oracle Forms we defined “data blocks” that represented tables and views that would be used in our forms
- ADF BC components do that and more, plus they may be shared by many applications
- In Oracle Forms once the “data block” was created we would then use it to create the presentation
- With ADF we use ADF Faces to accomplish the same thing and more
  (again creating components that may be reused by other applications)

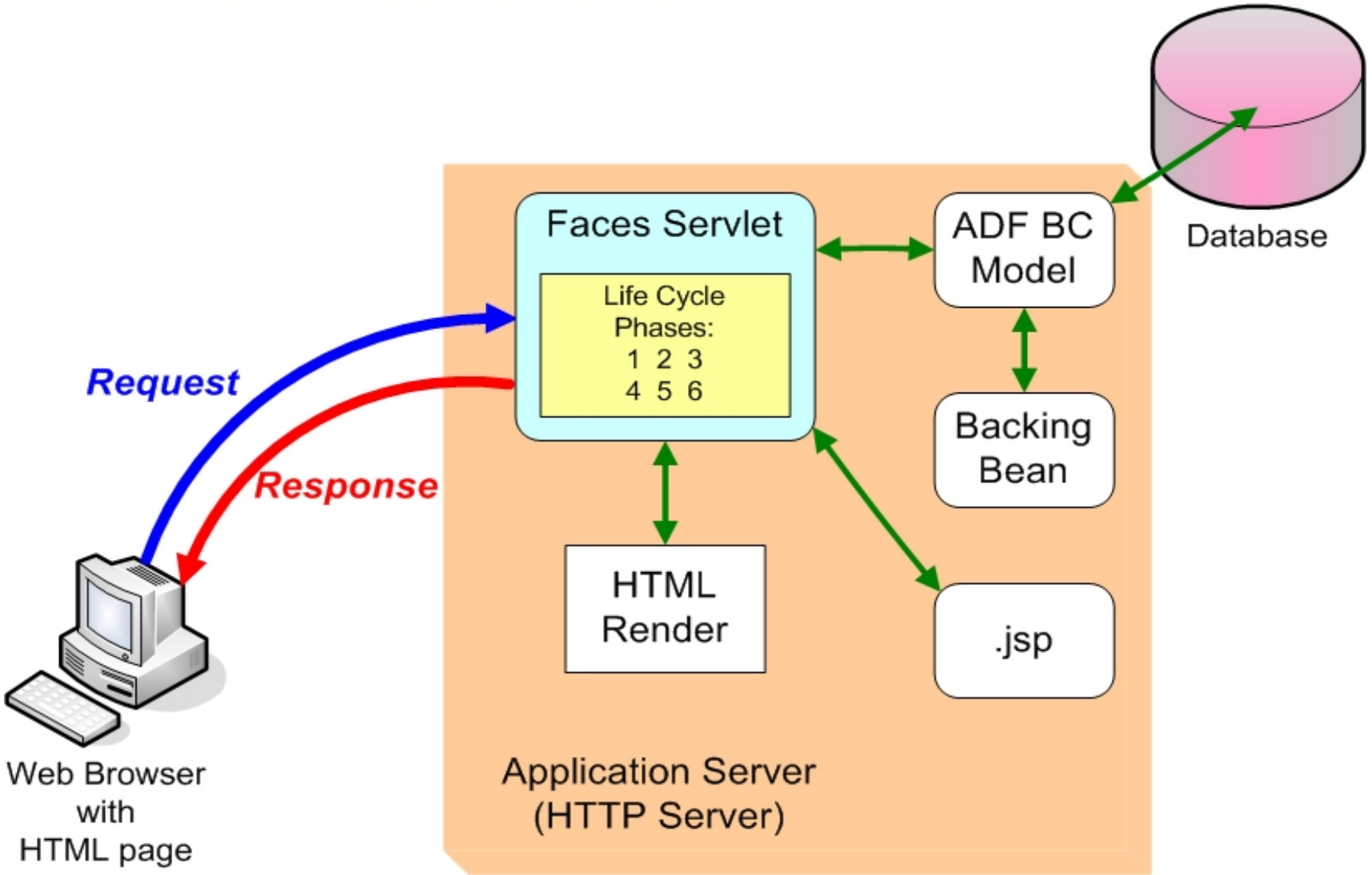
Comparison to Typical 4GLs

- Most 4GLs offer some type of "Data Object" or "Data Access Object" capability
  - Usually include wizard-based development
  - Usually work with relational database; do not usually support procedure-based data
  - Sometimes provide ability to find and link data objects using database dictionary
  - Sometimes provide stand-alone reusable data objects
  - Sometimes linked to GUI development via "drag-and-drop" capability

Creating Web Applications

- Oracle's Business Component Browser is impressive, but hardly a customer-facing interface
- ADF Faces extends the Java Server Faces (JSF) framework using XML tags to describe the UI
- ADF Faces provides a Rich-Client Interface that uses JavaScript and AJAX components; users must have a reasonably up-to-date browser (Internet Explorer 7.0 or higher, Mozilla Firefox 2.0 or higher, Safari 3.0 or higher) to use all of its features
- ADF Faces is designed to make creation of "rich-client" (RC) interfaces full-featured and declarative where possible

Review of Web Processing



HTML, CSS, and Forms

- Even though the ultimate page delivered to the Client Browser is HTML; with JDeveloper's Visual Editor and the combination of ADF Faces and JSF Faces it uses to create .jspx pages there is little need for ADF Developers to code HTML or CSS

- Yield to JDeveloper's declarative mechanism and refrain from coding

ADF Controller

- The ADF Controller extends the standard JSF controller and controls the MVC in ADF
- ADF Controller features include:
  - Sequence of page displays (may be conditional)
  - Allows partial-page processing in the same way as full page processing; only the necessary part of a page is rendered, the rest is unchanged (makes page processing faster)
  - Allows reuse of page parts
  - Provides conditional control of page flow

JSF Life Cycle

- JSF (and ADF Faces) follows a predictable cycle:

    1. Restore Components
    2. Apply Request Values
    3. Process Validations
    4. Update Model Values
    5. Invoke Application
    6. Render Response

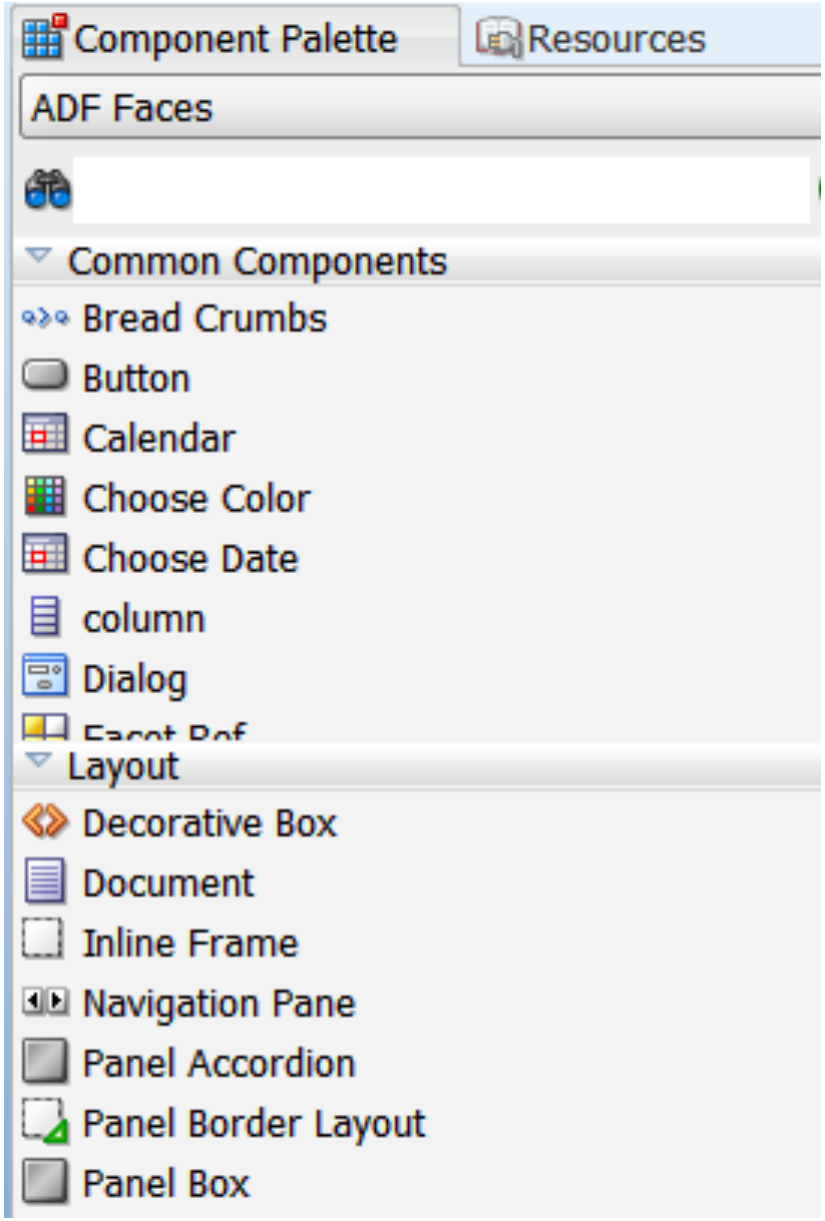
- This Life Cycle is normally transparent; however, it is useful to understand it when debugging

JDeveloper Visual Designer

- JDeveloper's Visual Designer may be used to "paint" a User Interface using the Component Palette
- The JDeveloper Visual Designer is intended to be WYSIWYG (What You See Is What You Get); however the nature of the web and HTML is that it's really WYSIKOWYG (What You See Is Kind-Of What You Get; thank you Peter Koletzke…)

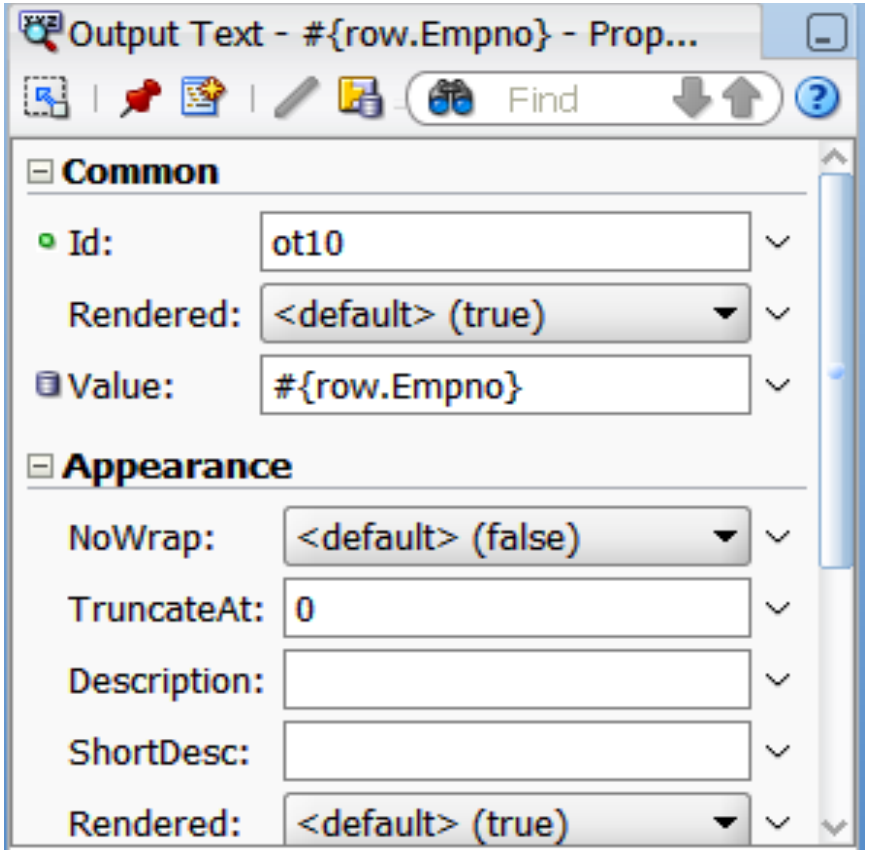
ADF Faces Component Palette

- The ADF Faces Component Palette includes icons representing various User Interface objects
- Drag-and-drop desired components into the position desired



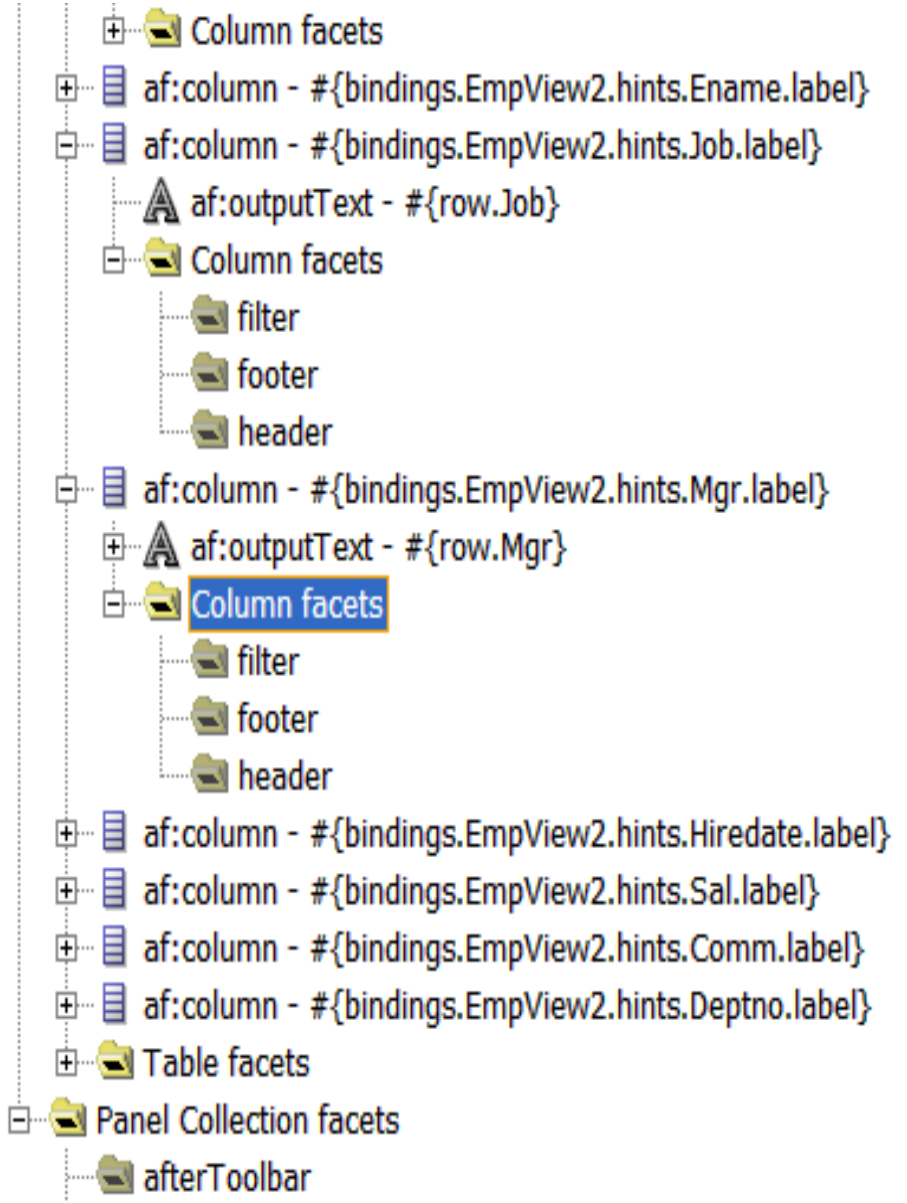
Property Inspector

- When editing Web Pages, the Property Inspector shows properties for the various “facets” and components displayed upon the page



Facets in Structure Window

- The “facets” are components that are used to contain groups of other components
- JDeveloper’s “Structure” Window” lists facets in the current page



Panel and Panel Splitter

- Pages in ADF are sometimes divided by Panels; pre-existing templates exist to help create the number of desired Panels
- Each Panel in turn may be divided into smaller areas using a Panel Splitter
  - By default Panel Splitters split an area horizontally
  - Panel Splitters have an “Orientation” property that allow the split to be vertical

Container Objects

- ADF provides many “container” objects specifically designed to hold data objects including:
  - Panel Collections are facets that contain other objects
  - Panel Accordions are facets that contain other objects but shrink-and-grow depending upon mouse movement
  - Tabbed Panels are facets that allow components to be placed into a tabbed structure

User Interface (UI) Components

- UI Components provided by ADF Faces include:
  - Buttons
  - Calendars
  - Choose Color
  - Forms
  - Input Text
  - Output Text
  - Panel Collection
  - Submit
  - Tables
  - more…

Binding Data

- JDeveloper's interface allows the creation of web components using drag-and-drop processing
  - Drag-and-drop may also be used to associate View Objects with UI Components
  - This has the effect of “binding” the data to the data control object

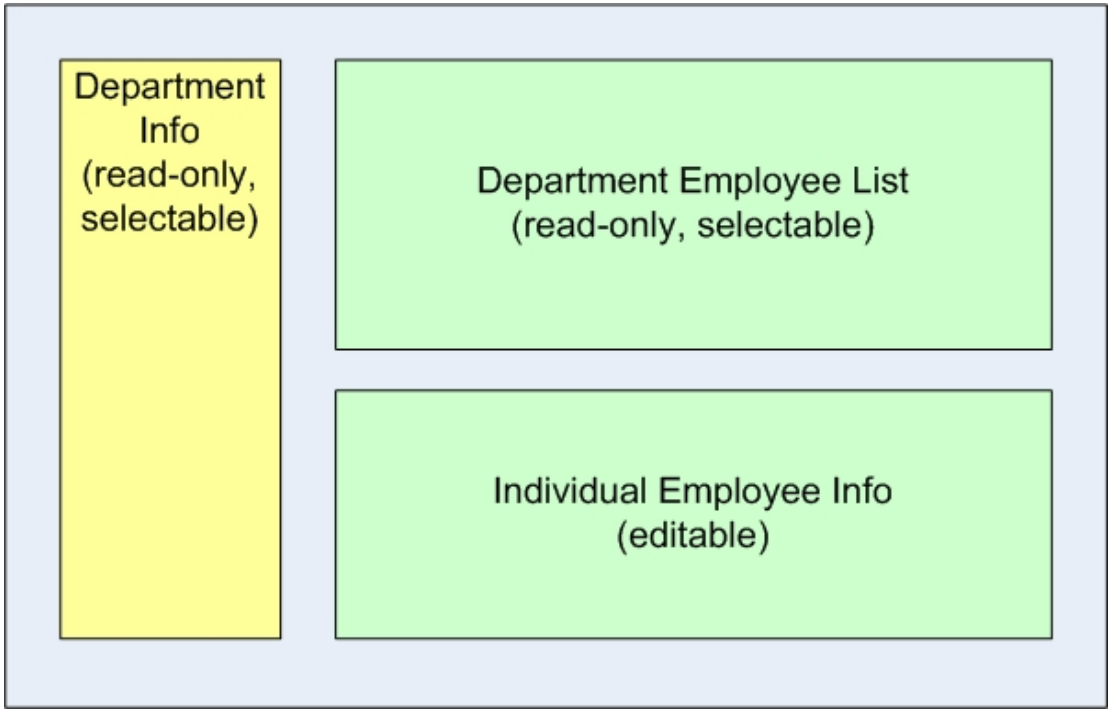
Creating ADF/JSF Faces Pages

- The following pages walk through the creation of a simple Web Application using ADF Faces and ADF BC objects as follows:

1. Design Web Page
2. Create new JSF Page using JDeveloper
3. Add Visual Components to JSF Page
4. Bind Visual Components to ADF BC Objects

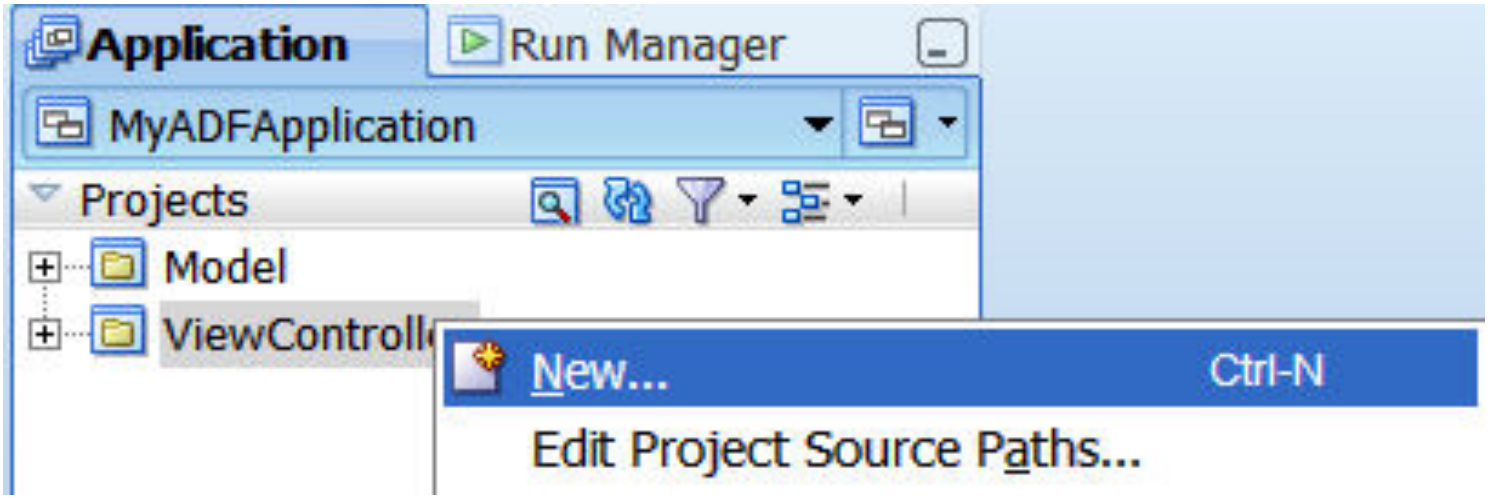
Target Screen Layout

- Rough design: Department info on the left, list of Department Employees (for selected department) in the upper-right, and the information for a single employee on the lower-right (selected from list)

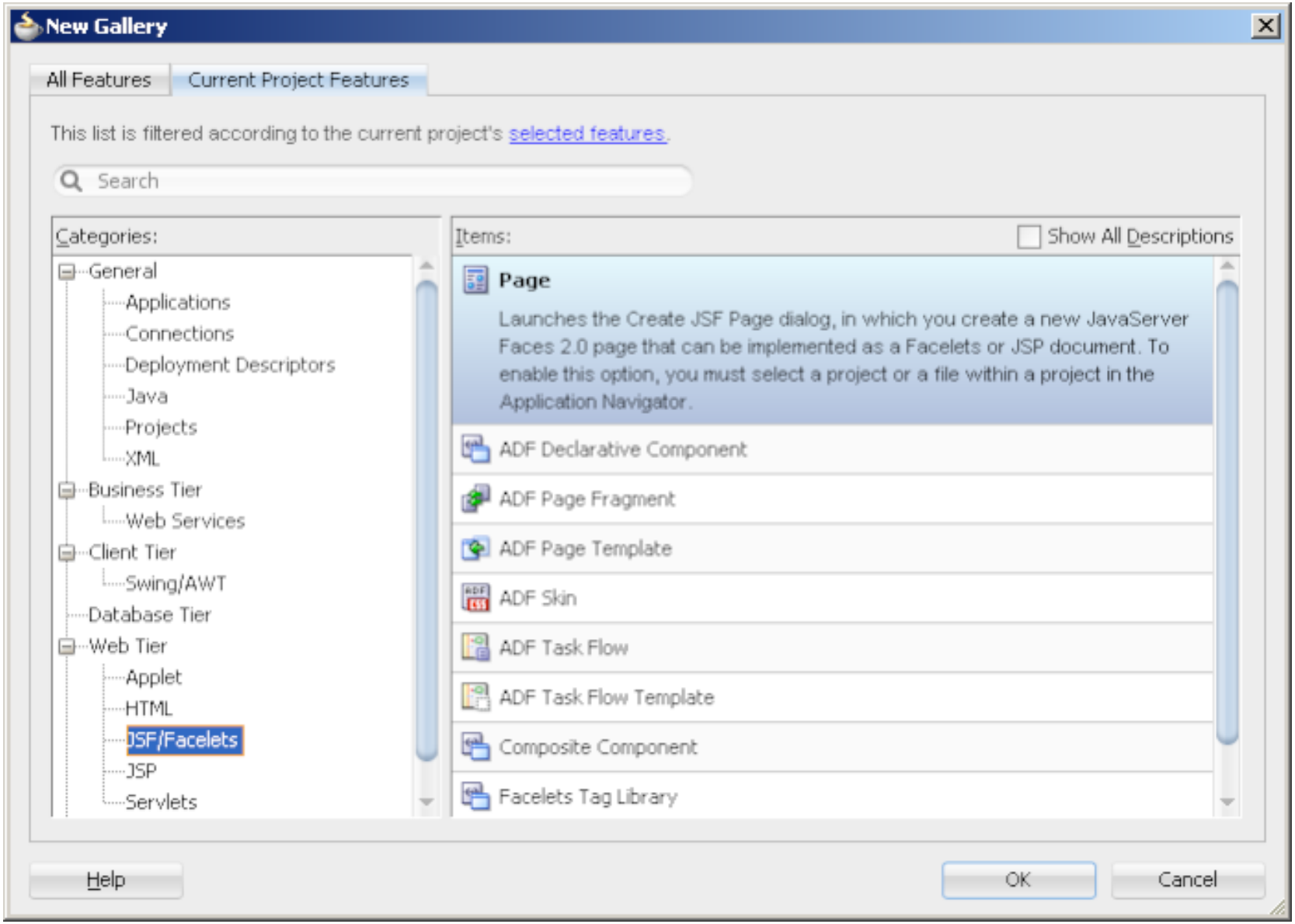


Create ADF Faces Page

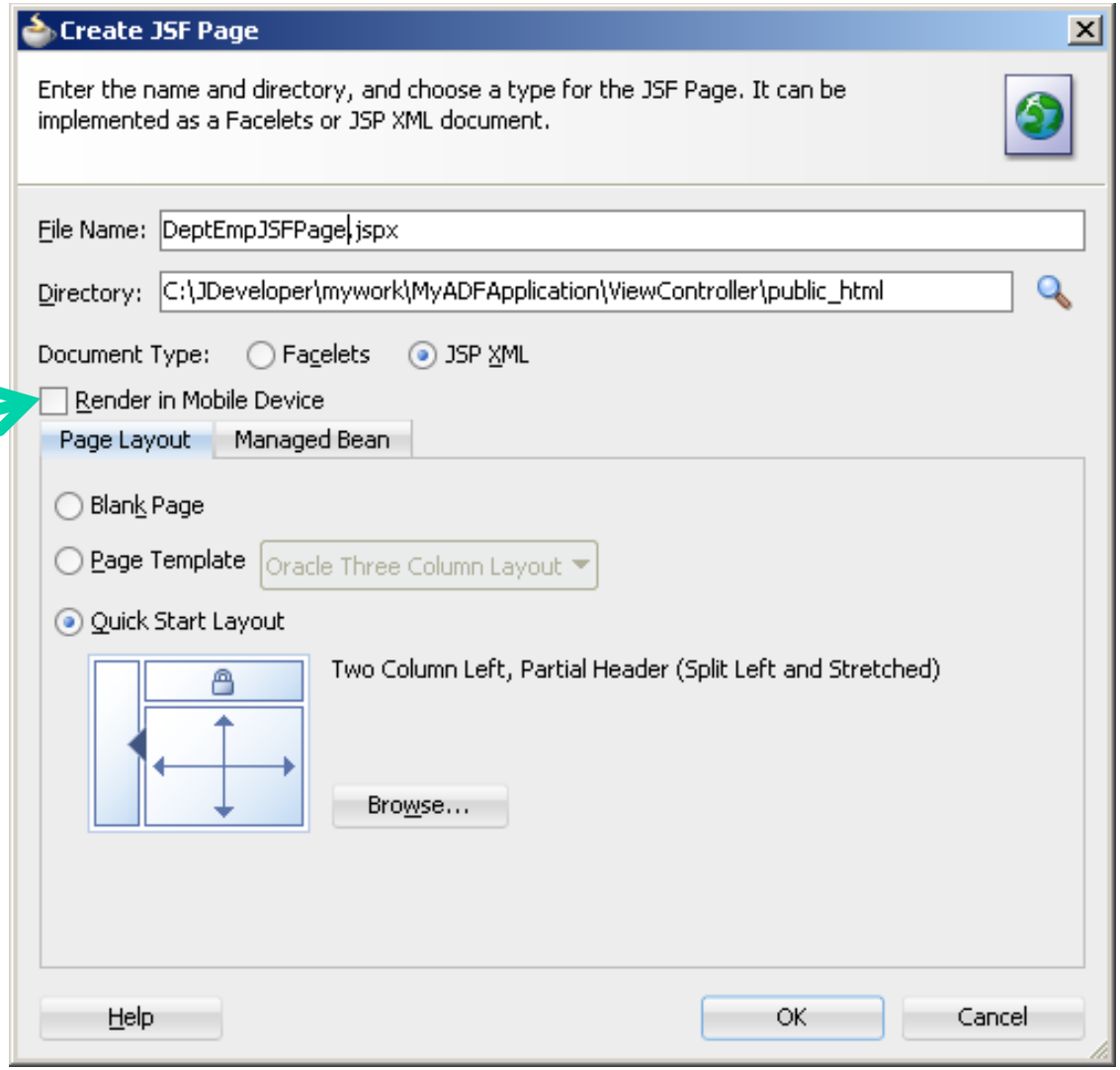
- To create an ADF Faces page, right-click on an Application's ViewController Project and choose “New” to display the “New Gallery” dialog



New Gallery



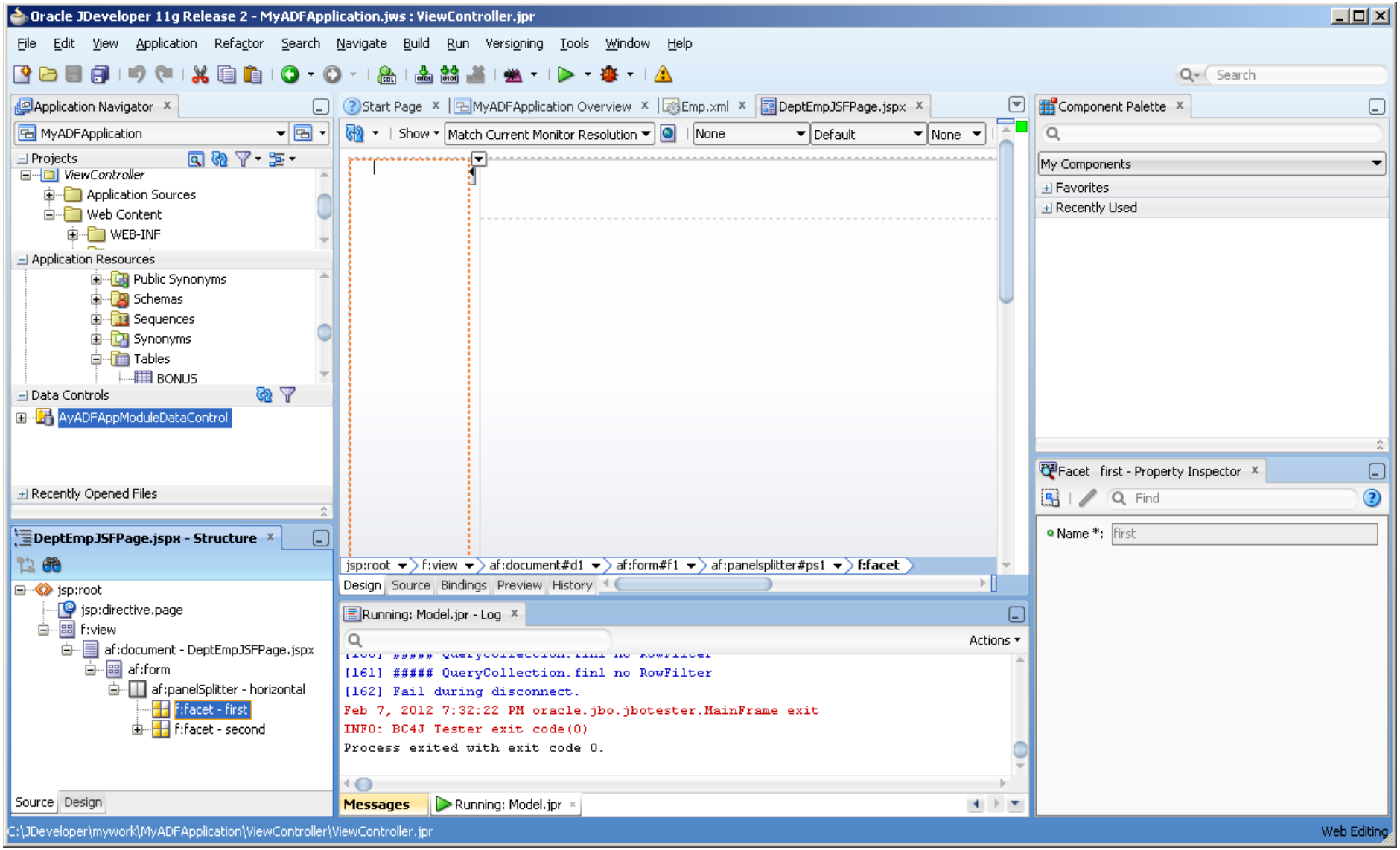
Naming Web Page



Not used in demo but pretty cool!

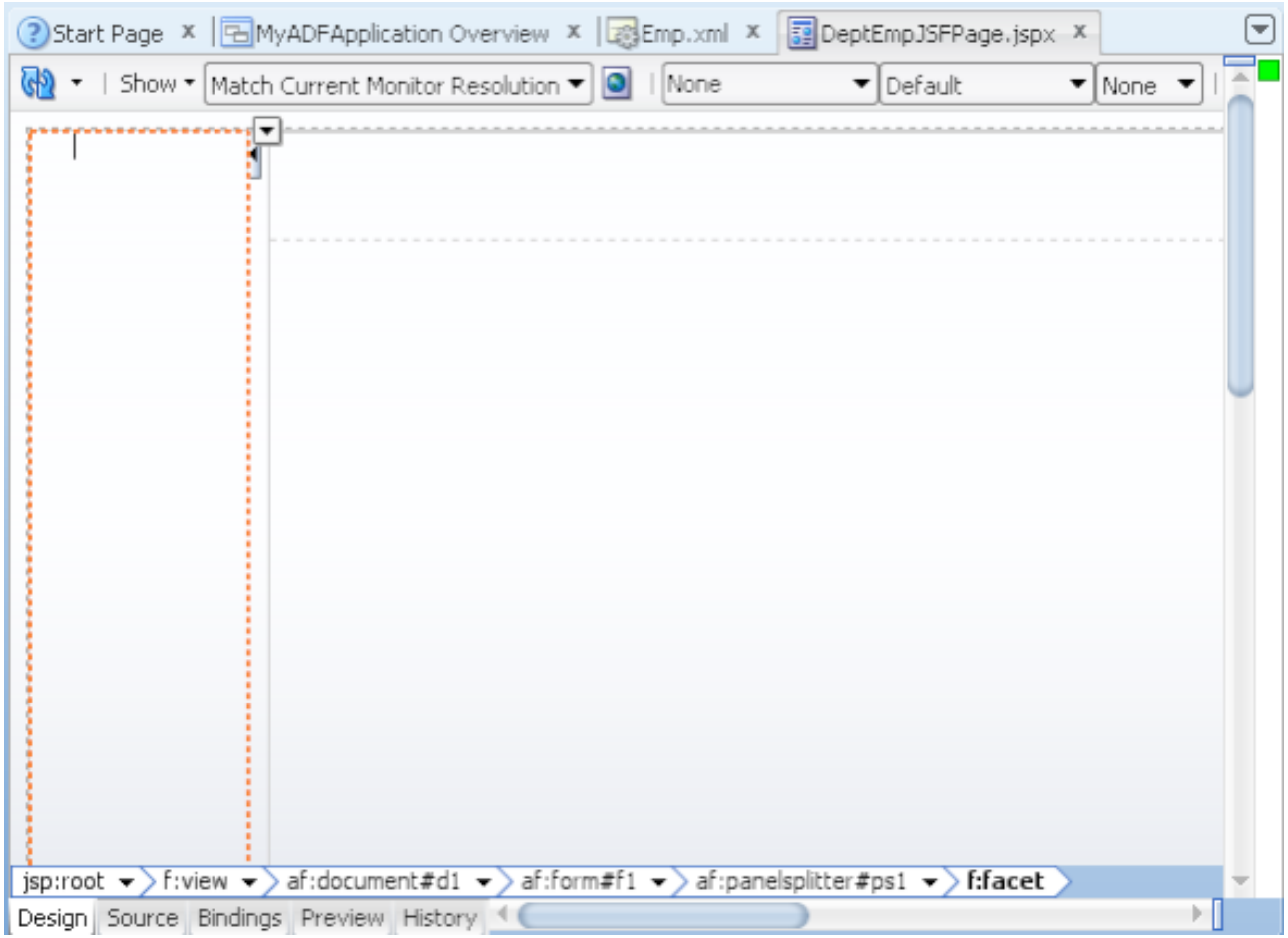
– Note the "JSP XML" Document Type

Visual Display with Initial Screen



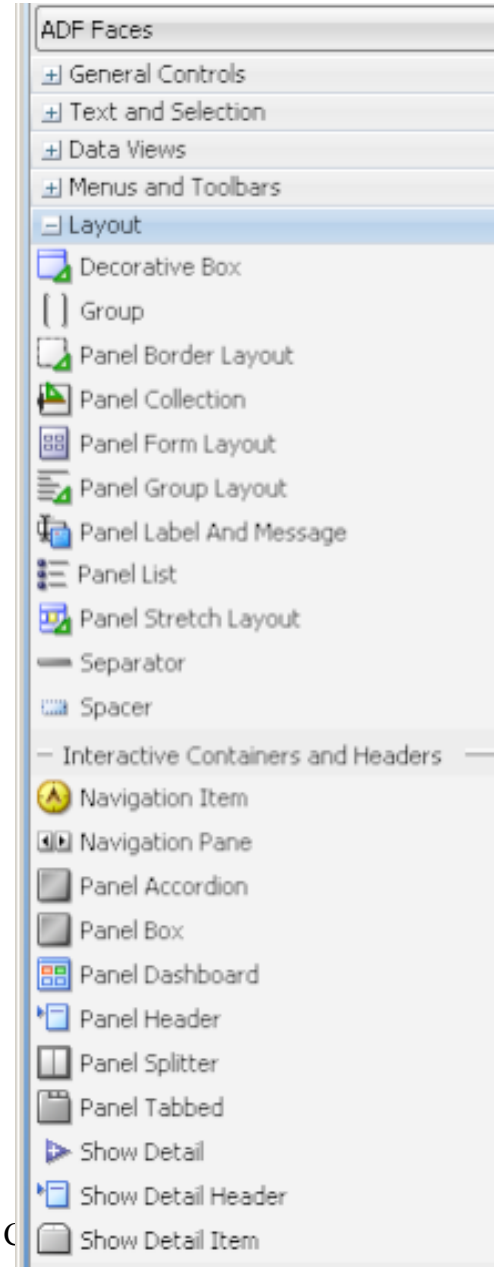
Quick-Start Layout

- The supplied quick start layout is ready to have objects dropped into it (resizing might be needed)



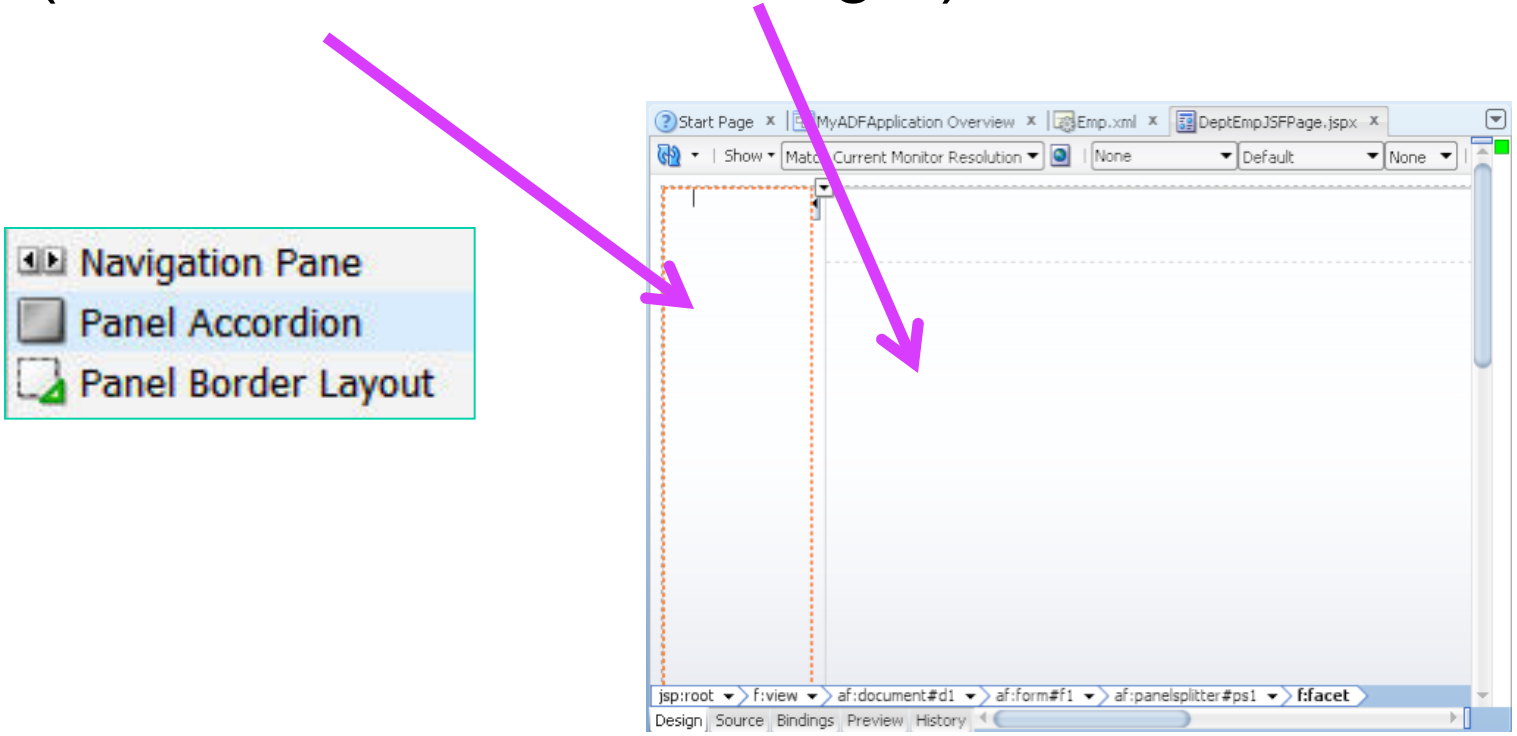
Layout Components

- ADF provides several “container” objects to hold data including:
  - Panel Accordian
  - Panel Tabbed
  - Panel Collection
  - Panel Splitter
  - more …



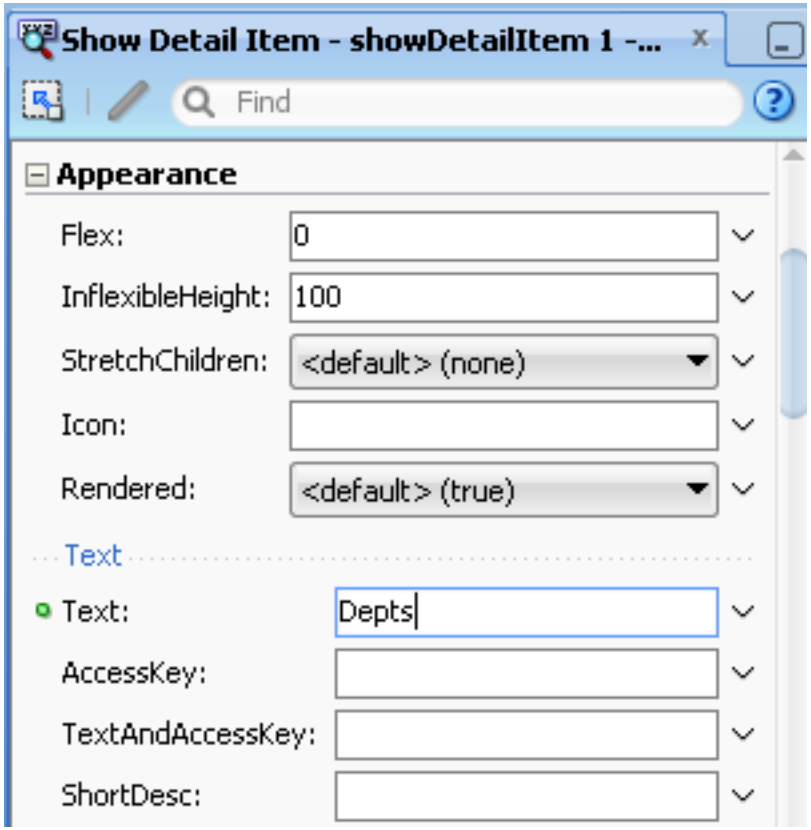
Adding Accordion Component

- To add an Accordion Component to the web page where the individual Dept values will appear and where the individual Emp values will appear (left-column and lower-right)



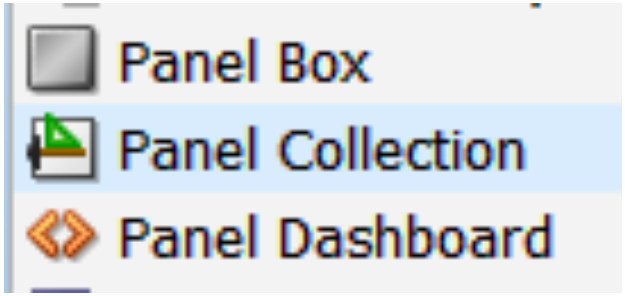
Change Accordion Title Property

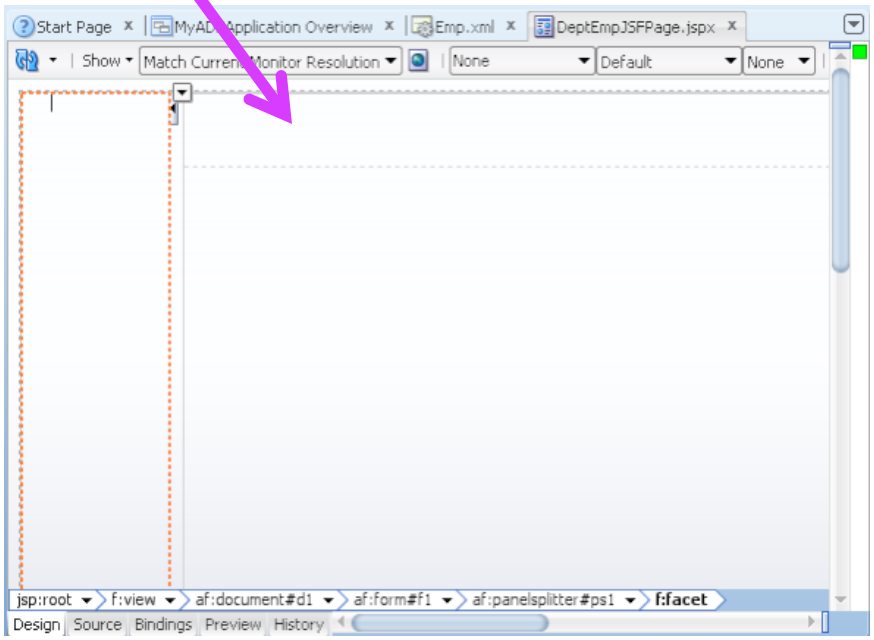
- To alter the Accordion's title, click on the Accordion and modify its Property Inspector Text item (changed to “Depts”)



Adding Collection Container

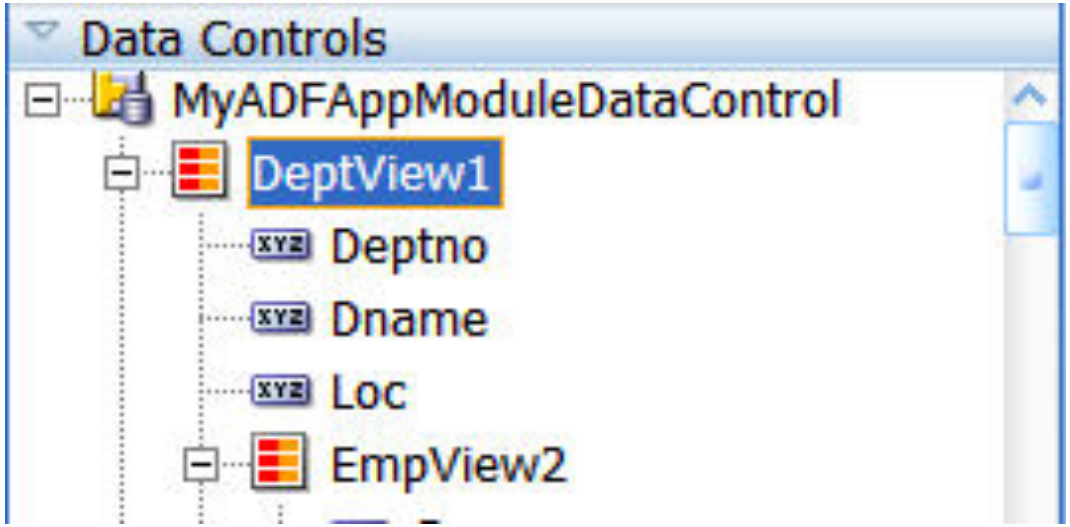
- Find the "Panel Collection" component in the Layout components and drag it to the "first" (top) part of the Splitter area





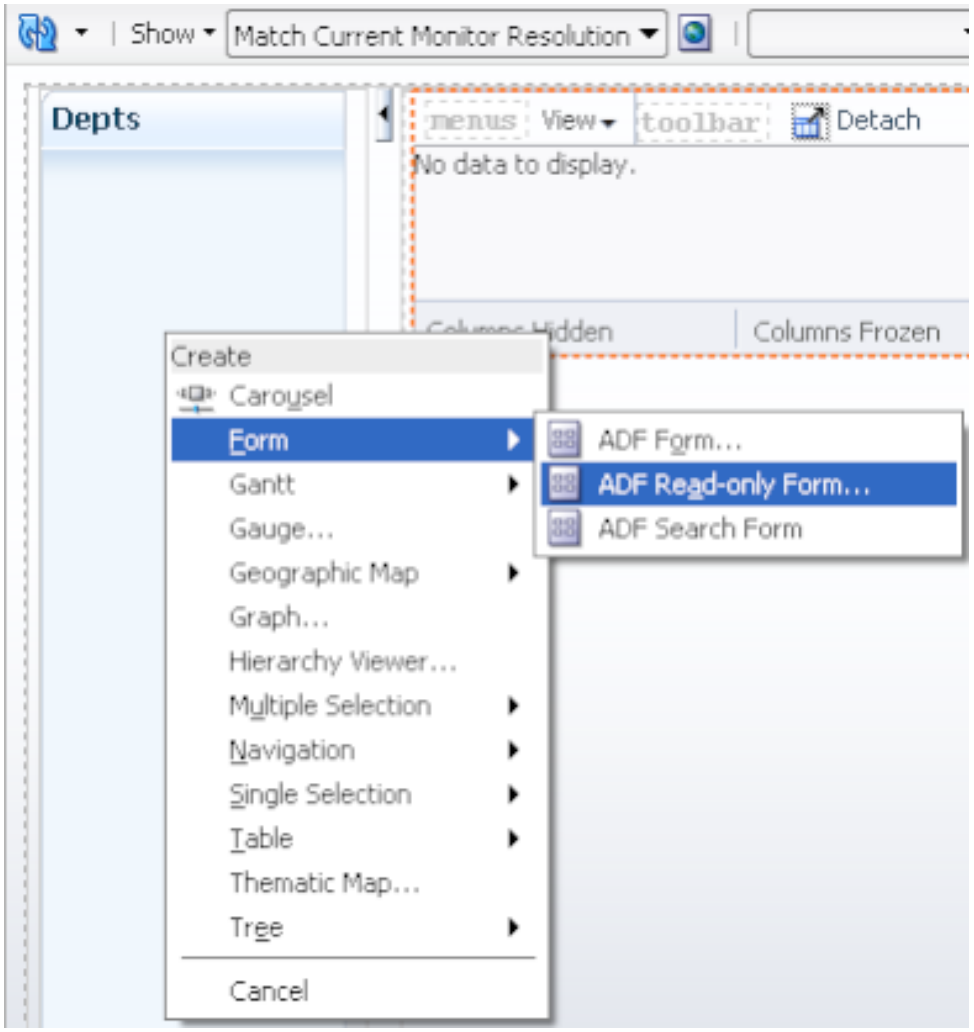
Data Binding: Adding Data, 1

- To “bind” data to web page components, simply drag ADF BC data objects to the Visual Editor
- Open the “Application Navigator” and expand the “Data Controls” accordion to see the ADF BC components created earlier then drag “DeptView1” to the “Depts” accordion



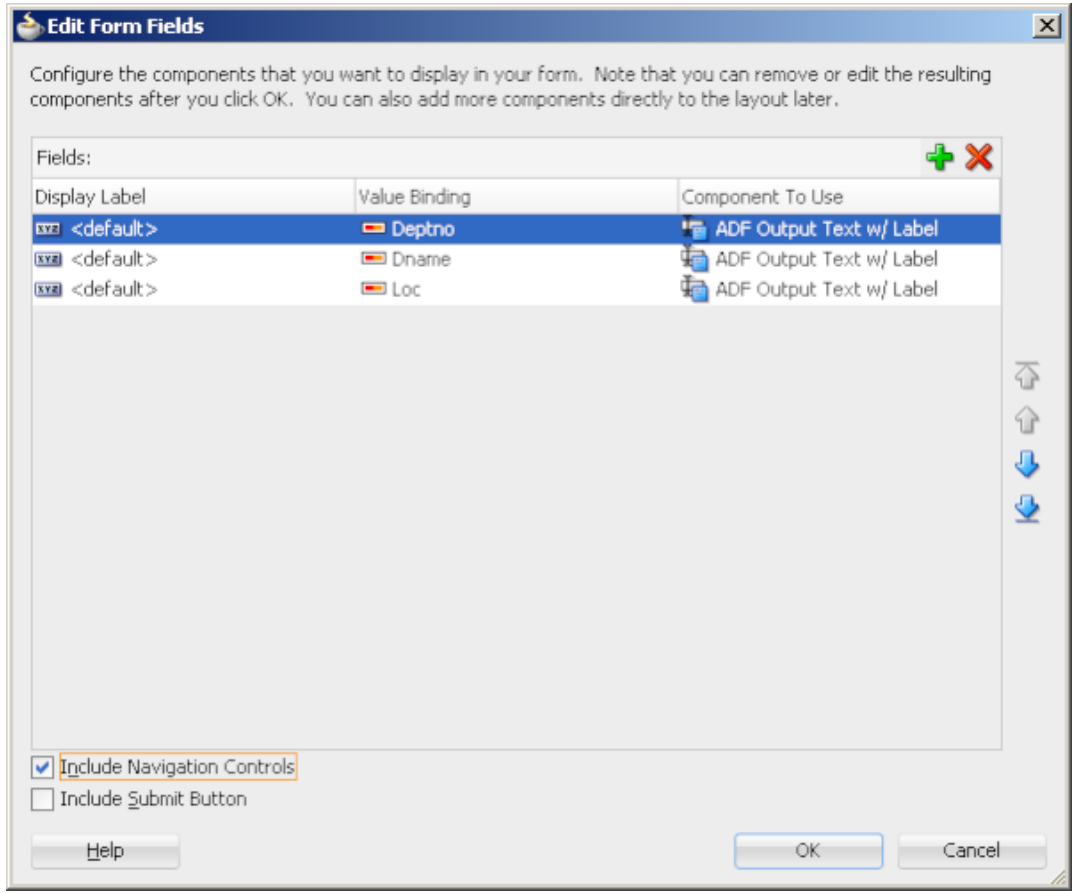
Data Binding: Adding Data, 2

- When prompted; choose "Create Forms -> ADF Read-Only Form" to populate the Department data display



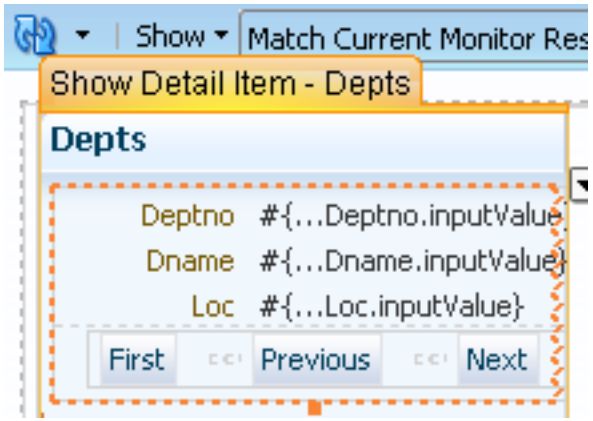
Adding Navigation Controls

- Check the "Include Navigation Controls" box
- You may also modify display labels and add, delete, or reorganize the values displayed



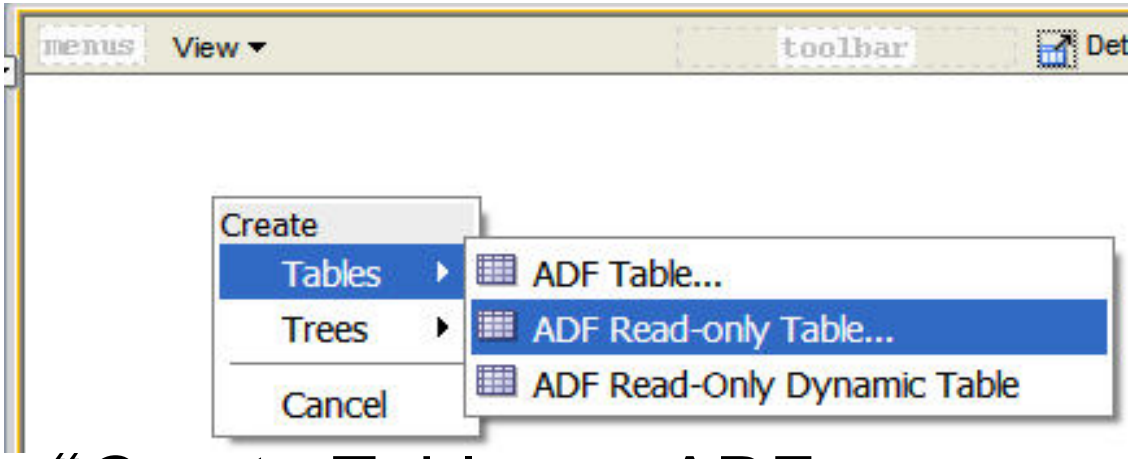
Department Display Area

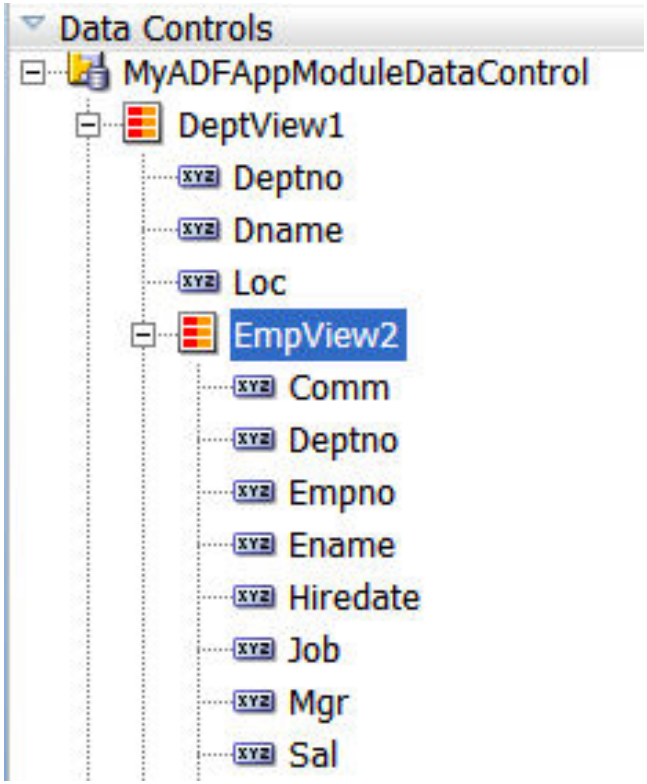
- After adding the Department information; the "Depts" accordion should look like the following



Adding Department Employees

- Next, to add Department Employees to the page, drag the EmpView2 data control

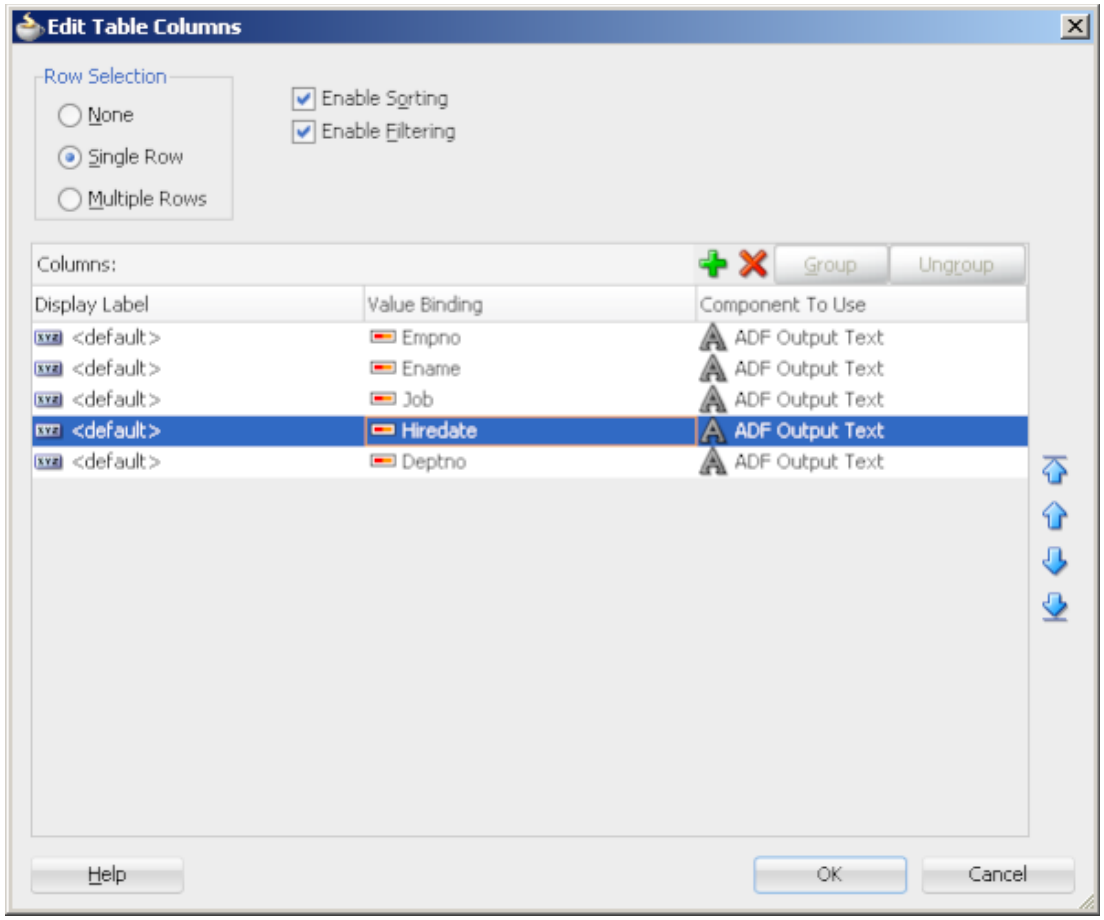




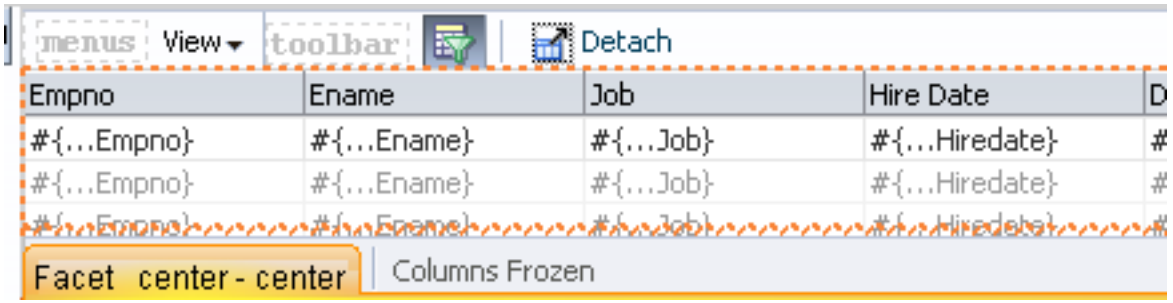
"Create Tables -> ADF Read-Only Table"

Add Employee Navigation Controls

- Check all three navigation controls: Row Selection, Sorting, and Filtering; as before columns may be relabeled, added, deleted, reorganized

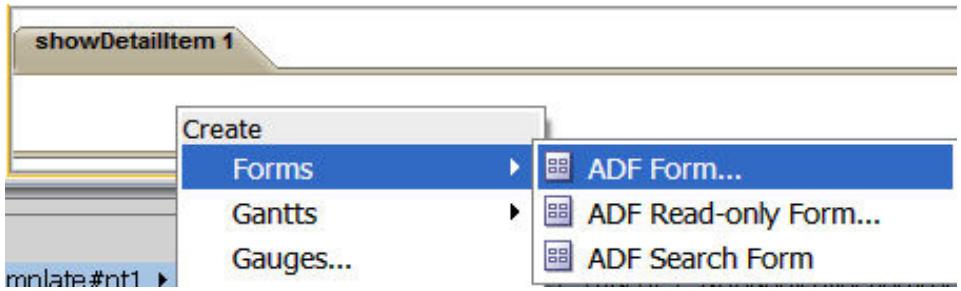


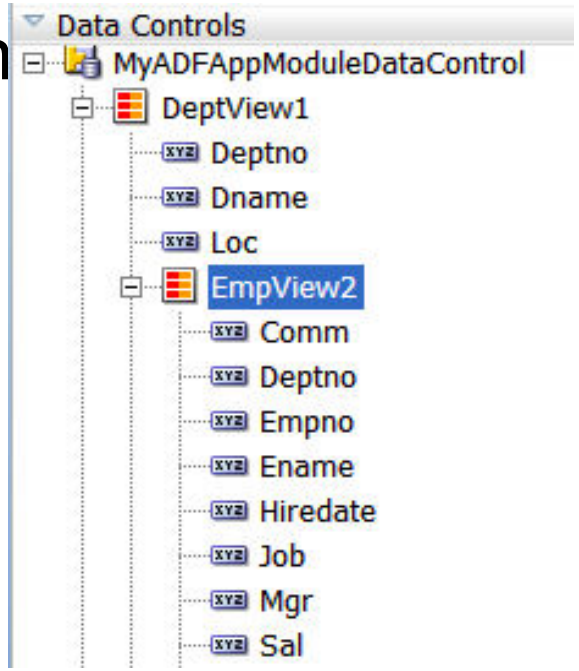
Department Employee Area



Adding Individual Employee

- Finally, add the individual Employee display to the Accordian area at the bottom of the page

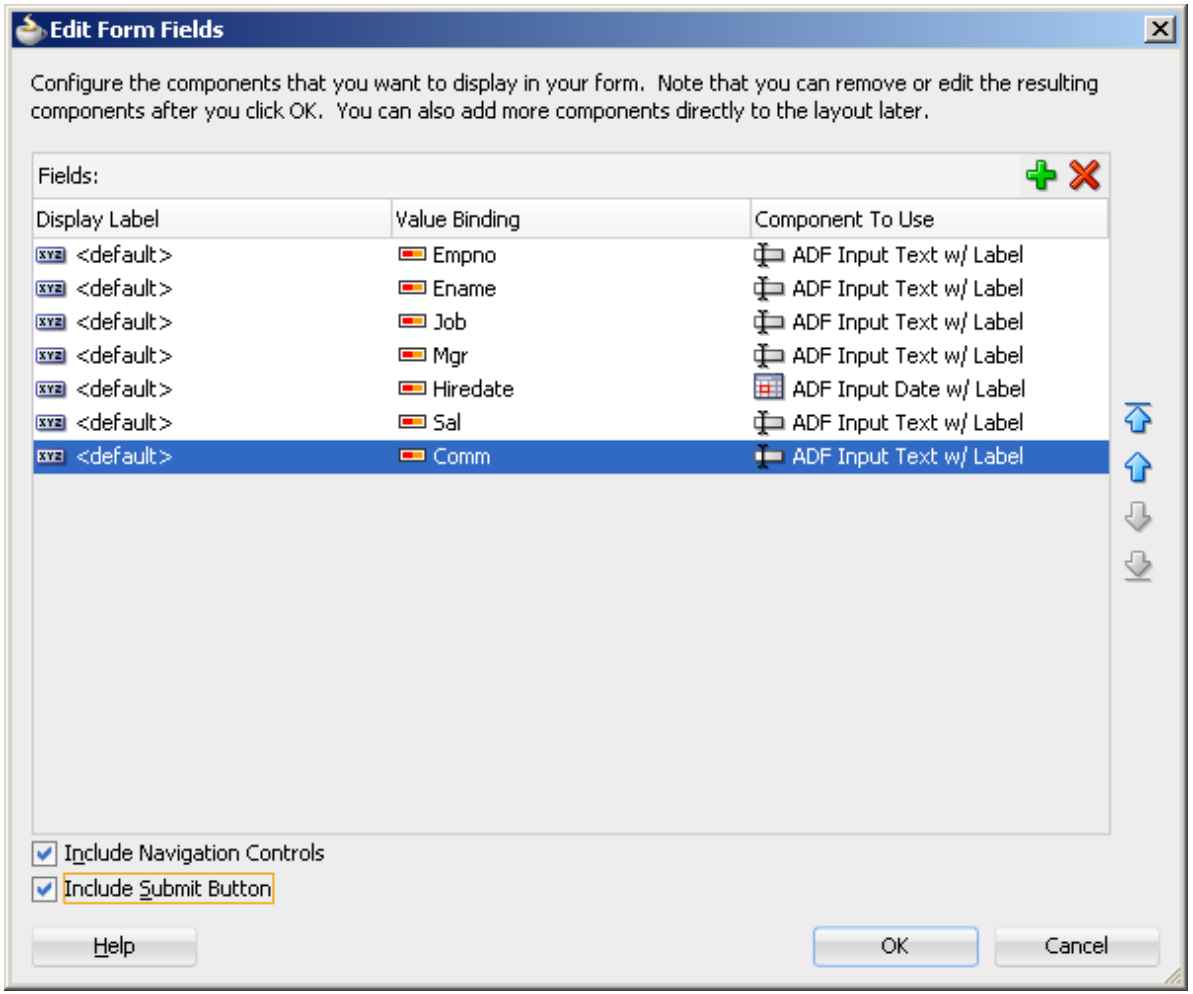




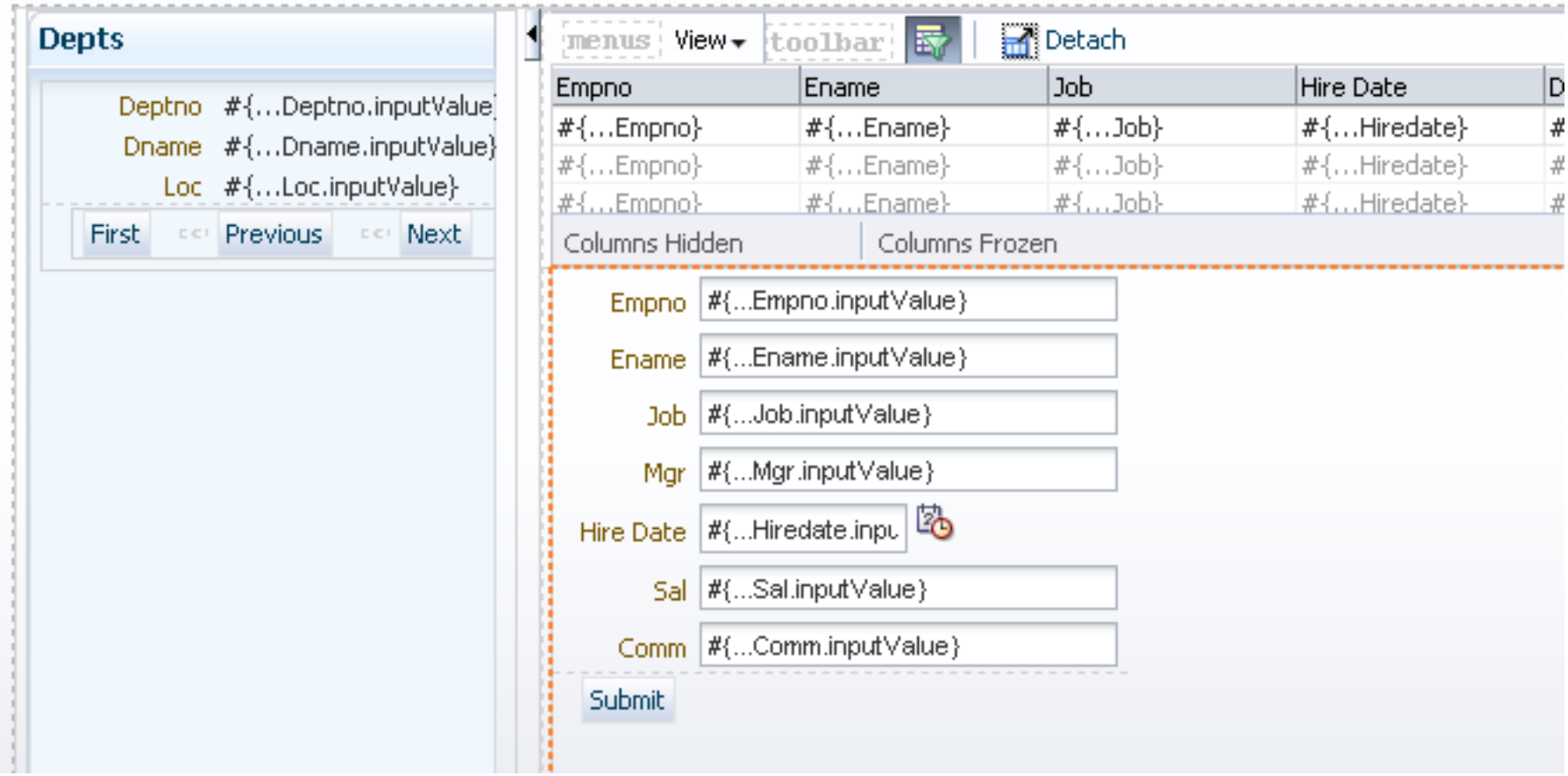
- When prompted, choose "Create Forms -> ADF Form" to select the display format (this part of the form will be editable)

Add Employee Navigation

- Delete the DEPTNO from the display (highlight & click X); check “Include Submit Button”

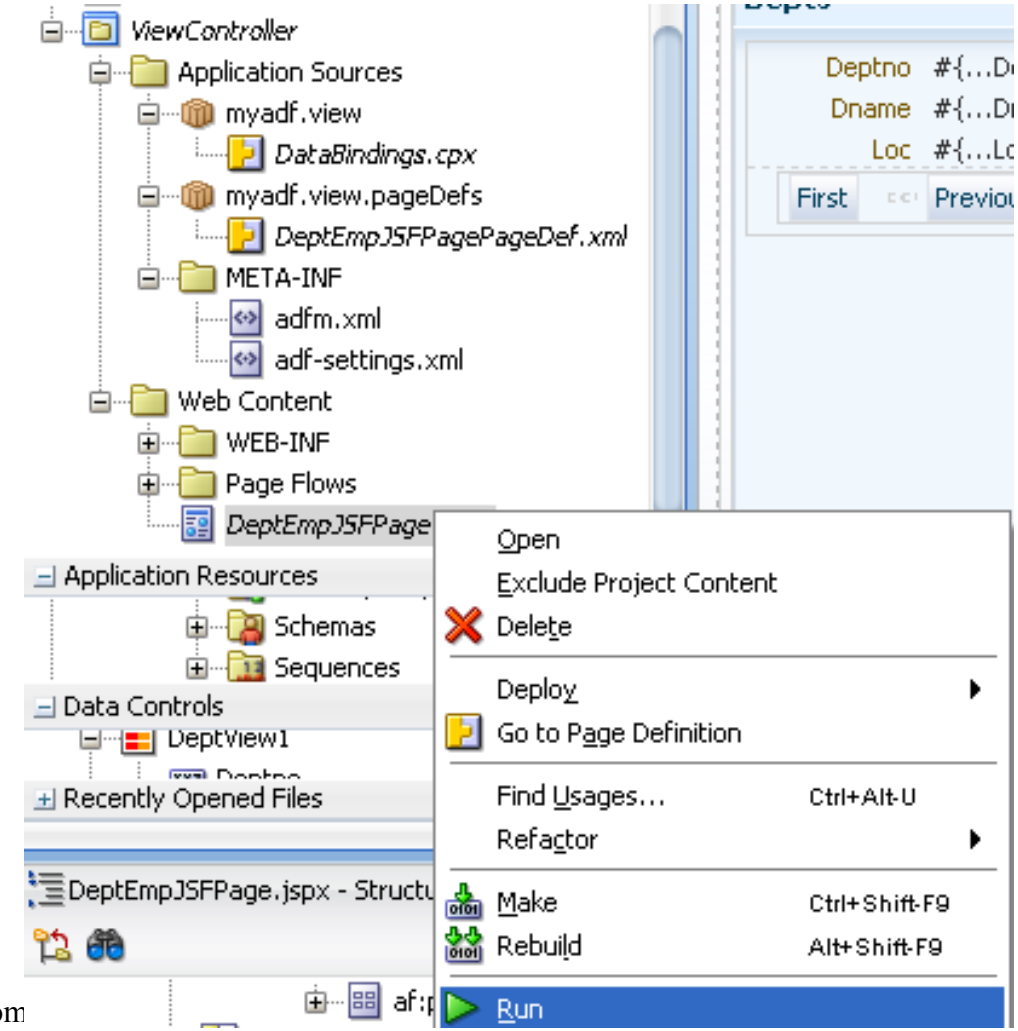


Completed Web Application Page



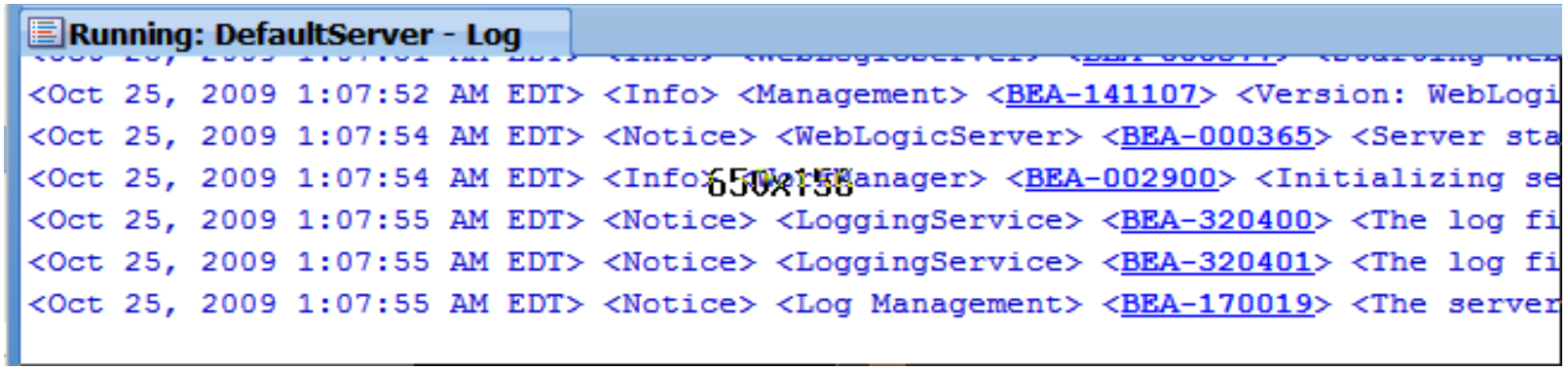
Testing the Web Application

- To test the Web Application; right-click the ".jspx" file in the ViewController project and choose "Run"



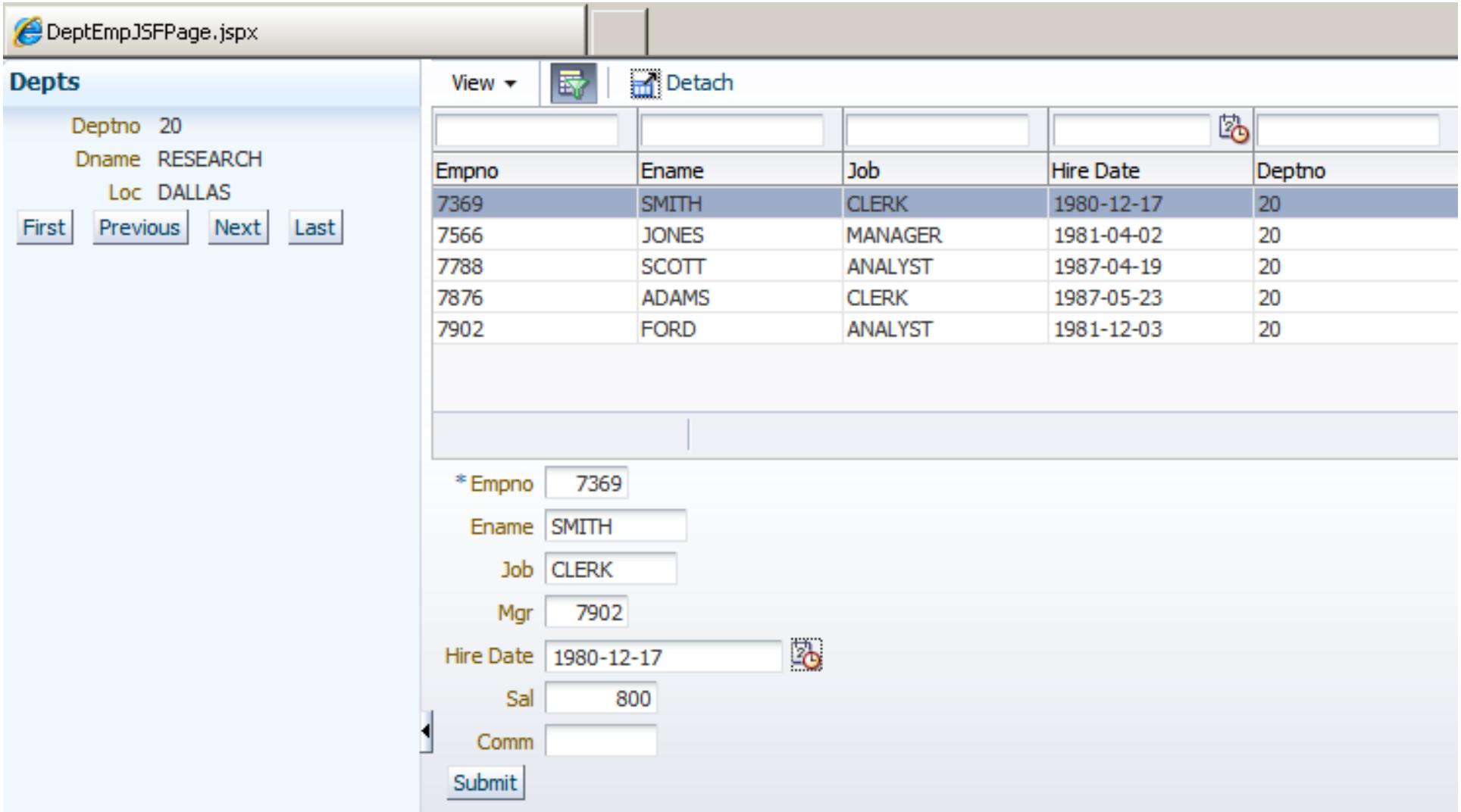
Be Patient!

- The first time you execute a Web application JDeveloper starts its built-in WebLogic Application Server; this takes a while
- You can track the progress of the Server's startup in JDeveloper's DefaultServer Log



- Once the Server is "up" your web page should be displayed in a browser (again, please be patient!)

Web Page in Browser



Files Supporting Application

- Several files make up the typical ADF Web Application
  - A .jspx file is used to define each web page
  - Web pages reference a page definition XML file (.xml)
  - Bindings are described in another XML file (.cpx)

JSF .jspx File

- ADF defines a web page using an XML .jspx file

DeptEmpJSFPage.jspx | Emp.xml

Find

```
 7  <f:view>
 8    <af:document id="d1">
 9      <af:messages id="m1"/>
10      <af:form id="f1">
11        <af:pageTemplate viewId="/oracle/templates/threeColumnTemplate.jspx"
12                         id="pt1">
13          <f:facet name="center">
14            <af:panelSplitter id="ps1" orientation="vertical">
15              <f:facet name="first">
16                <af:panelCollection id="pc1">
17                  <f:facet name="menus"/>
18                  <f:facet name="toolbar"/>
19                  <f:facet name="statusbar"/>
20                  <af:table value="#{bindings.EmpView2.collectionModel}"
21                            var="row" rows="#{bindings.EmpView2.rangeSize}"
22                            emptyText="#{bindings.EmpView2.viewable ? 'No data to display
23                            fetchSize="#{bindings.EmpView2.rangeSize}"
24                            rowBandingInterval="0"
25                            filterModel="#{bindings.EmpView2Query.queryDescriptor}"
26                            queryListener="#{bindings.EmpView2Query.processQuery}"
27                            filterVisible="true" varStatus="vs"
28                            selectedRowKeys="#{bindings.EmpView2.collectionModel.selected
29                            selectionListener="#{bindings.EmpView2.collectionModel.makeCu
```

cument#d1 ▸ af:form#f1 ▸ af:pagetemplate#pt1 ▸ f:facet ▸ af:panelaccordion#pa1 ▸ **af:showdetailitem#sdi1** ▸

Design | Source | Bindings | Preview | History

ADF Web Page Definition file (.xml)

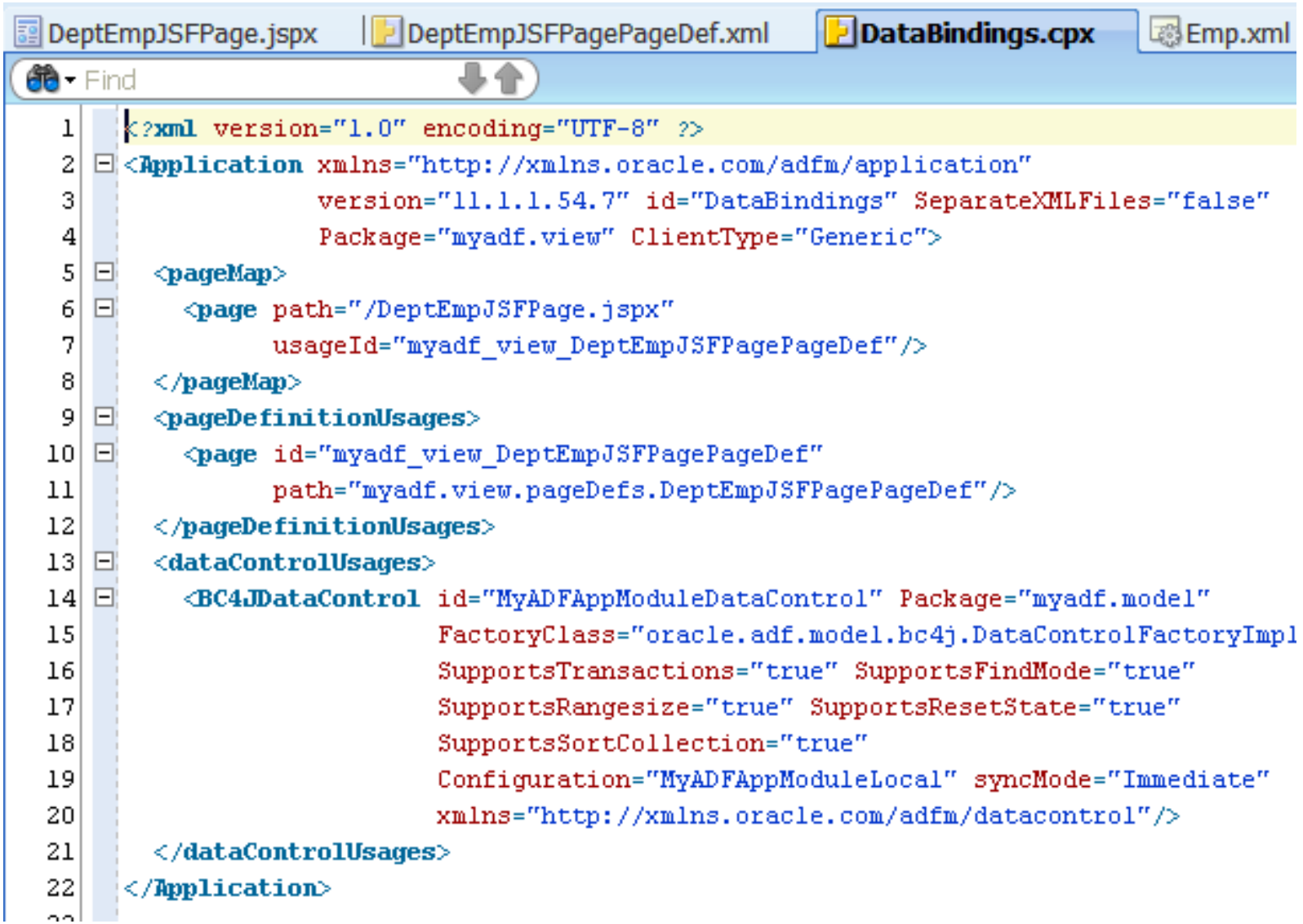
DeptEmpJSFPage.jspx | DeptEmpJSFPagePageDef.xml | Emp.xml

Find

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
3                  version="11.1.1.54.7" id="DeptEmpJSFPagePageDef"
4                  Package="myadf.view.pageDefs">
5    <parameters/>
6    <executables>
7      <variableIterator id="variables"/>
8      <iterator Binds="DeptView1" RangeSize="25"
9                DataControl="MyADFAppModuleDataControl" id="DeptView1Iterator"
10               ChangeEventPolicy="ppr"/>
11     <iterator Binds="EmpView2" RangeSize="25"
12               DataControl="MyADFAppModuleDataControl" id="EmpView2Iterator"
13               ChangeEventPolicy="ppr"/>
14     <searchRegion Binds="EmpView2Iterator" Criteria=""
15                   Customizer="oracle.jbo.uicli.binding.JUSearchBindingCustomizer"
16                   id="EmpView2Query"/>
17   </executables>
18   <bindings>
19     <attributeValues IterBinding="DeptView1Iterator" id="Deptno">
20       <AttrNames>
21         <Item Value="Deptno"/>
22       </AttrNames>
23     </attributeValues>
24     <attributeValues IterBinding="DeptView1Iterator" id="Dname">
```

Overview | Source | History

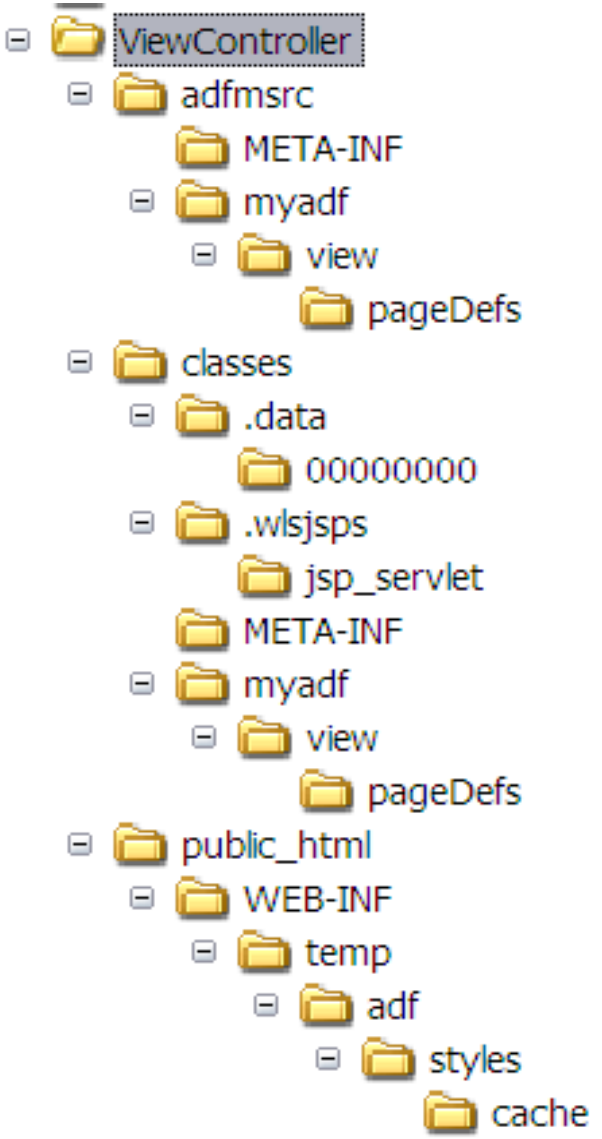
ADF Bindings XML file (.cpx)




```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <Application xmlns="http://xmlns.oracle.com/adfm/application"
3               version="11.1.1.54.7" id="DataBindings" SeparateXMLFiles="false"
4               Package="myadf.view" ClientType="Generic">
5    <pageMap>
6      <page path="/DeptEmpJSFPage.jspx"
7            usageId="myadf_view_DeptEmpJSFPagePageDef"/>
8    </pageMap>
9    <pageDefinitionUsages>
10     <page id="myadf_view_DeptEmpJSFPagePageDef"
11           path="myadf.view.pageDefs.DeptEmpJSFPagePageDef"/>
12   </pageDefinitionUsages>
13   <dataControlUsages>
14     <BC4JDataControl id="MyADFAppModuleDataControl" Package="myadf.model"
15                      FactoryClass="oracle.adf.model.bc4j.DataControlFactoryImpl
16                      SupportsTransactions="true" SupportsFindMode="true"
17                      SupportsRangesize="true" SupportsResetState="true"
18                      SupportsSortCollection="true"
19                      Configuration="MyADFAppModuleLocal" syncMode="Immediate"
20                      xmlns="http://xmlns.oracle.com/adfm/datacontrol"/>
21   </dataControlUsages>
22 </Application>
```

ADF Faces ViewController Files

- The XML files representing the ViewController project are distributed using a directory structure



ADF vs. Forms

Feature	Forms	ADF
Declarative database access	Yes	Yes
Reuse of database access	Some	Yes
Declarative user interface development	Yes	Yes
Automatic screen generation	Yes	Some
Reuse of user interface	Some	Yes
Web Deployment	Yes	Yes
Client-Server Deployment	No	Yes
Fusion Applications development tool	No	Yes
Customizable	Yes	Yes
Built with open standards	No	Yes

Available Books

- Quick Start Guide to Oracle Fusion Development
  - Grant Ronald
  - Oracle Press
- Oracle JDeveloper 11g Handbook
  - Duncan Mills, Peter Koletzke, Dr. Avrom Roy-Federman
  - Oracle Press
- Oracle Fusion Developer's Guide
  - Frank Nimphius, Lynn Munsinger
  - Oracle Press

Wrapping It Up

- Oracle Forms is not going anywhere; it is not necessary to "convert" things to ADF (third-party tools are available to help if you insist)
- Oracle's design emphasis and new features will support the Java-based ADF mechanism; Oracle Forms probably won't see much in the way of new functionality
- JDeveloper and ADF allow me to create simple forms almost as easily as in forms except:
  - ADF BC (data) creates reusable components
  - ADF Faces (view) creates reusable components
- I did not write a single line of Java in this demo!

Training Days 2013

http://www.rmoug.org

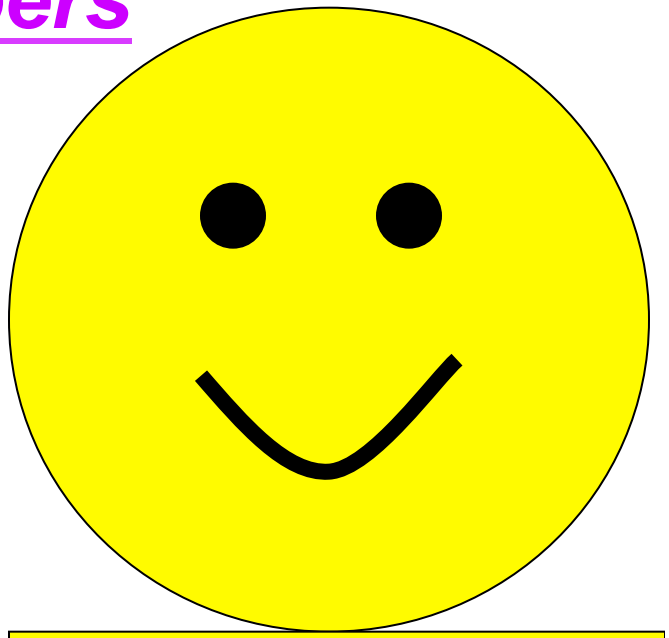
Watch for the Date!

MARK YOUR CALENDARS!

COLLABORATE 12
April 22-26, 2012
Mandalay Bay Convention Center
Las Vegas, Nevada

Please Fill Out Session Evaluations

Oracle ADF for Forms Developers



Thanks for your attention!

To contact the author:

John King
King Training Resources
PO Box 1780
Scottsdale, AZ 85252 USA
1.800.252.0652 - 1.303.798.5727
Email: john@kingtraining.com

Today's slides and examples are on the web:

http://www.kingtraining.com