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

- Learn how JDeveloper may be used to create ADF-based applications
- Become familiar with ADF Faces and how it is used to create user interfaces
- Use ADF BC to model data and address business rules

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
- Home is Centennial, Colorado – I love it here!
- Member of ODTUG (Oracle Development Tools User Group) Board of Directors
- Active member of Rocky Mountain Oracle Users Group (RMOUG)

Who Are You?

- Forms Developer
- Java Developer
- Both
- Neither

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 (12g) is on the way!

Why ADF?

- Oracle Application Development Framework (ADF) is a Java-based development tool (much like 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
- 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?

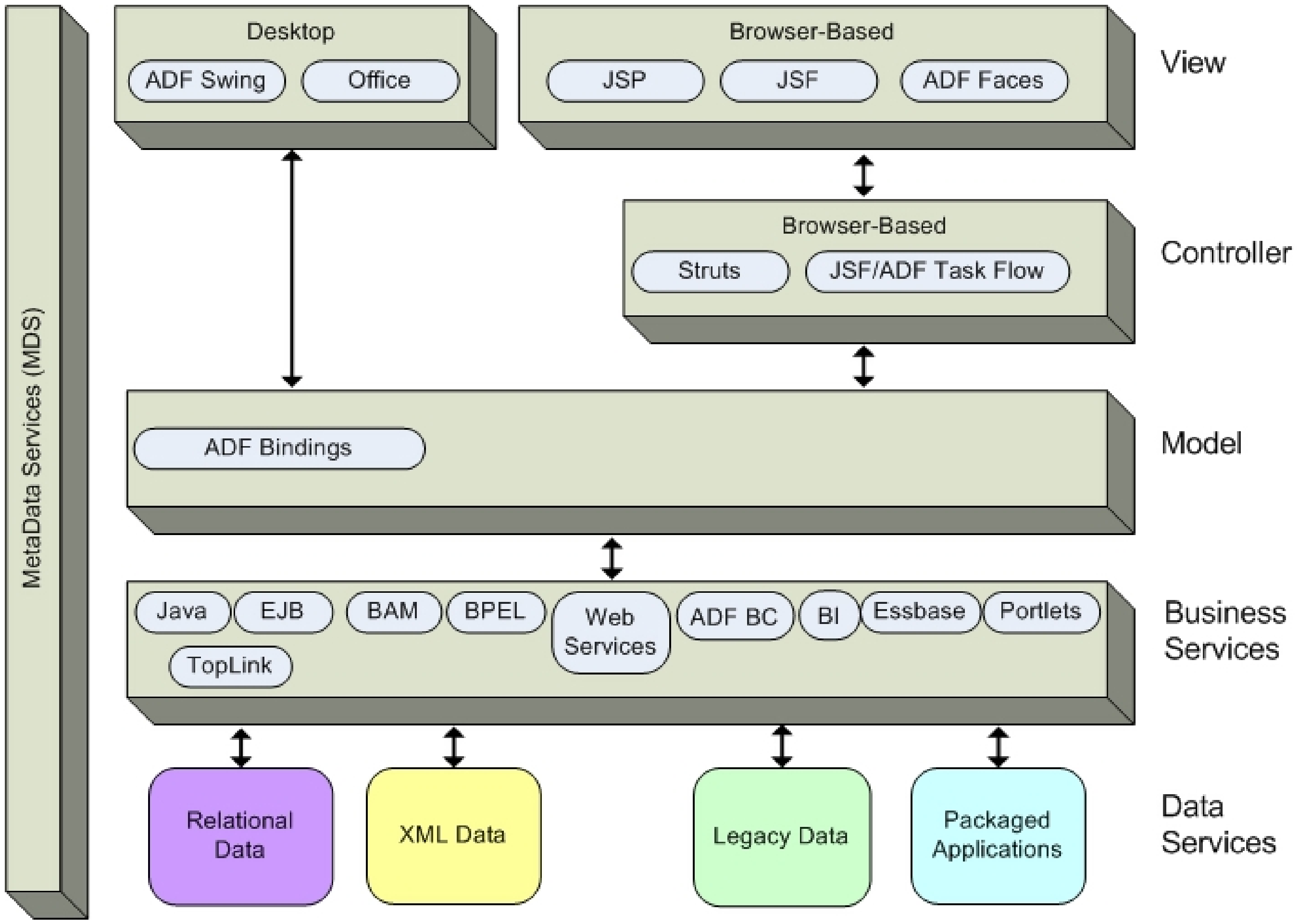
- Probably not well
  - Much the same as someone with basic PL/SQL could create very basic Oracle Forms
  - Someone with very basic Java and Web Skills can easily create applications with ADF
- Someone on your team needs to know Java very well
- Someone on your team needs to understand ADF and its available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

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

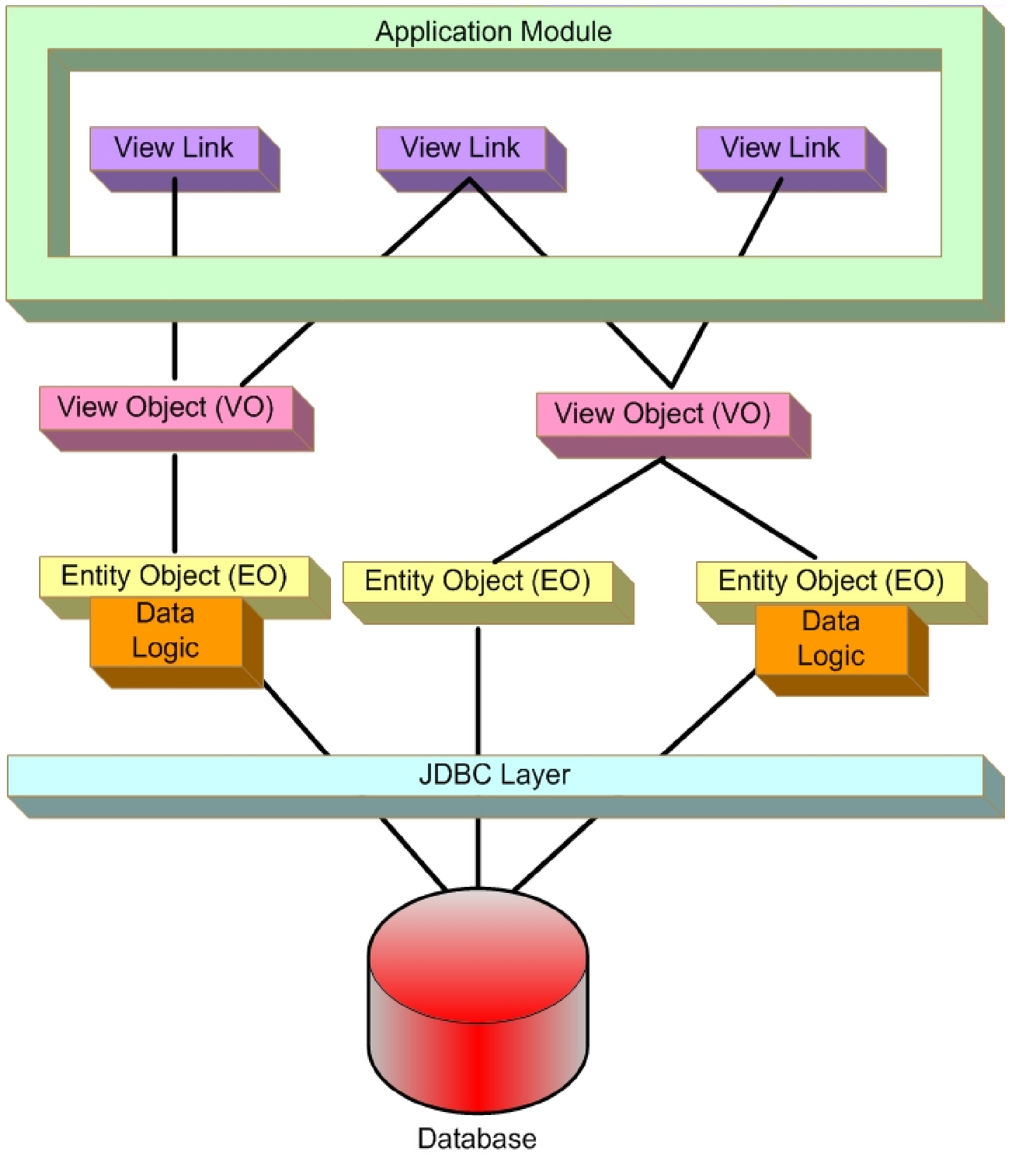
ADF Business Components (ADF BC)

- ADF Business Components 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
  - Provide transactional infrastructure
  - Implement best practices

ADF BC Objects

- ADF BC is implemented using a variety of objects to:
  - Define query views (read-only)
  - Define Insert-Update-Delete views to perform data manipulation
  - 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:
  - Application layout containers
  - 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” Features

- 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
- JDeveloper 11g is Oracle's latest release
- 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/technology/products/jdev/index.html

Oracle WebLogic Server

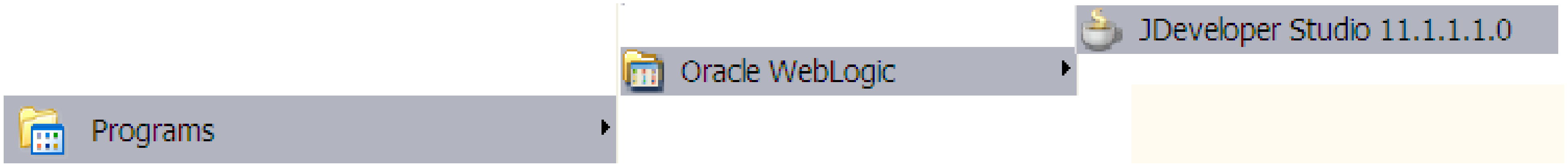
- Oracle WebLogic Server is Oracle's preferred platform to provide both a standard Java EE environment and an environment specifically tailored to Oracle Fusion Middleware; providing:
  - Complete Java EE 5 compatibility
  - Complete Java SE 6 compatibility
  - Web Services support
  - Integration with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

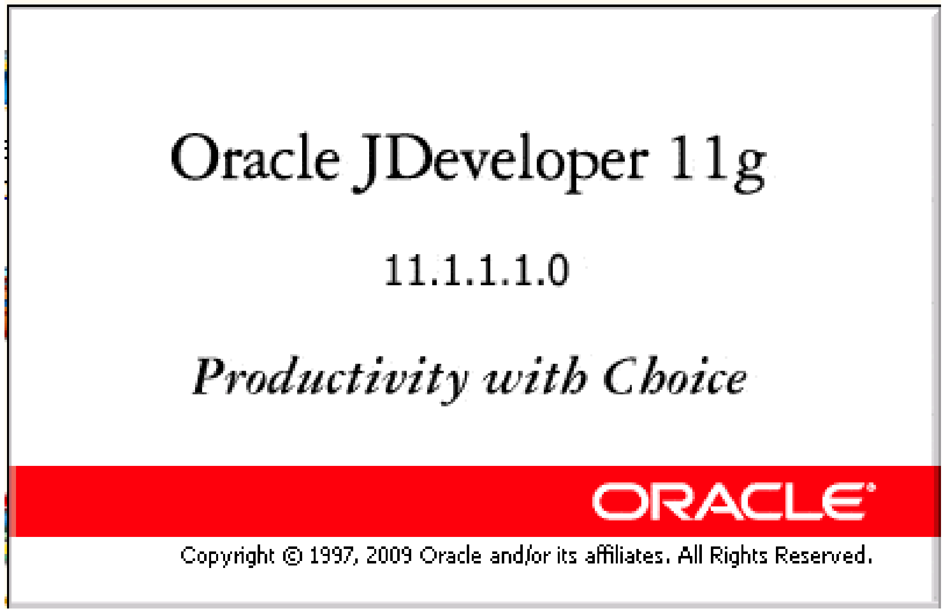
Oracle AS and OC4J?

- 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
- To learn more about Oracle WebLogic Server see Oracle's website:

  http://www.oracle.com/appserver/index.html

Starting JDeveloper




Oracle JDeveloper 11g

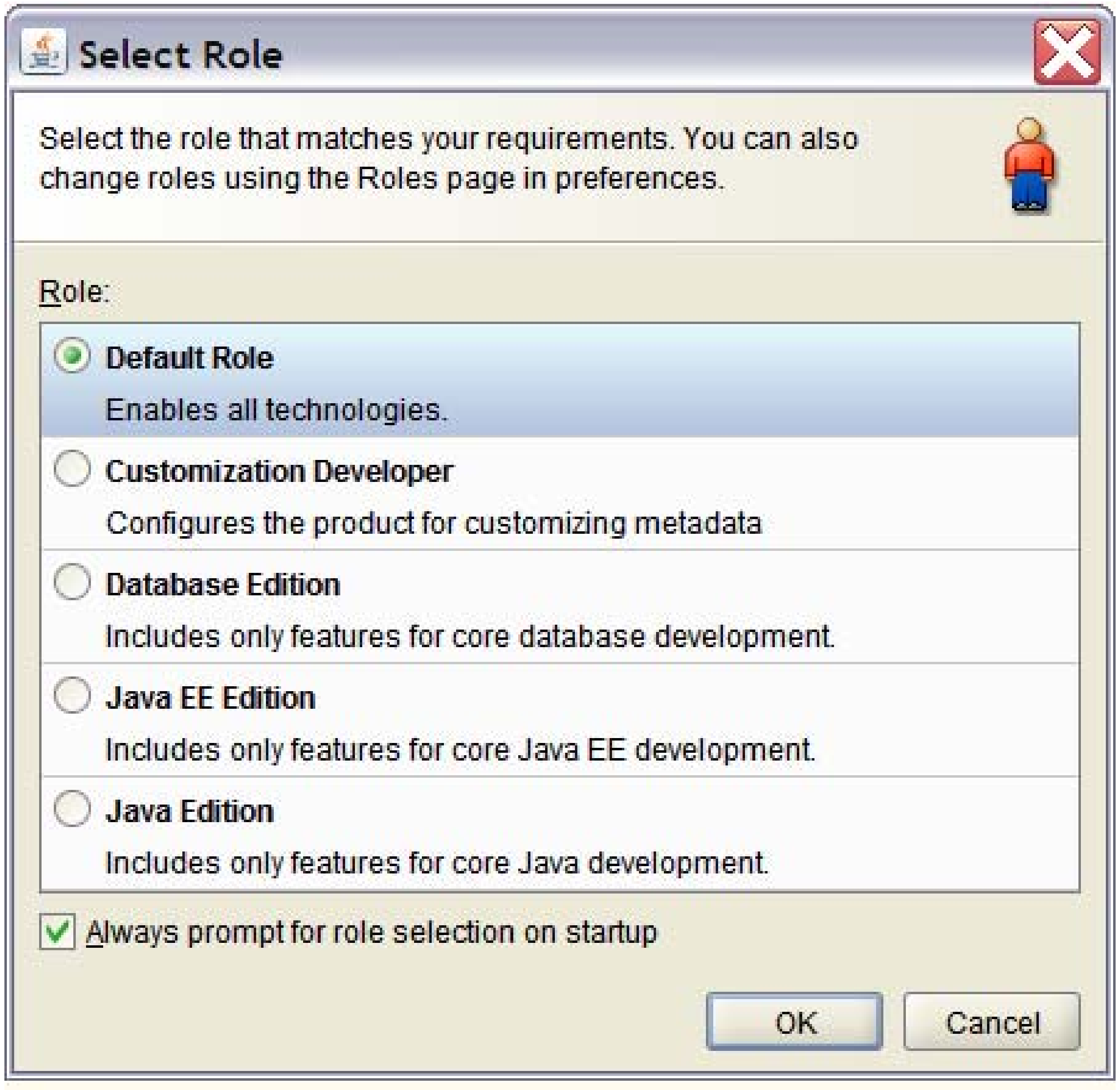
11.1.1.1.0

Productivity with Choice

ORACLE

Copyright © 1997, 2009 Oracle and/or its affiliates. All Rights Reserved.

JDeveloper - Select Role



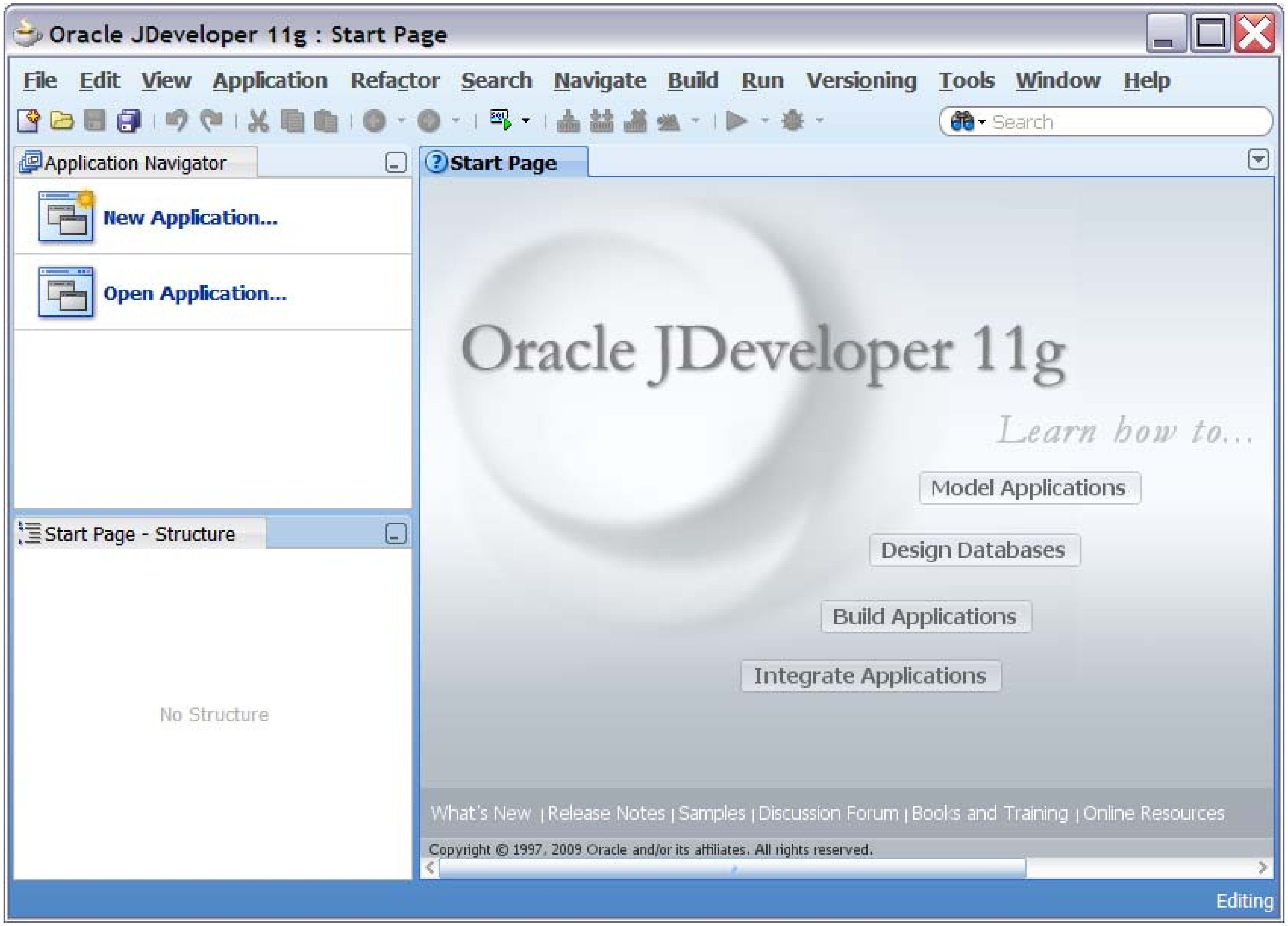
Select the role that matches your requirements. You can also change roles using the Roles page in preferences.

Role:

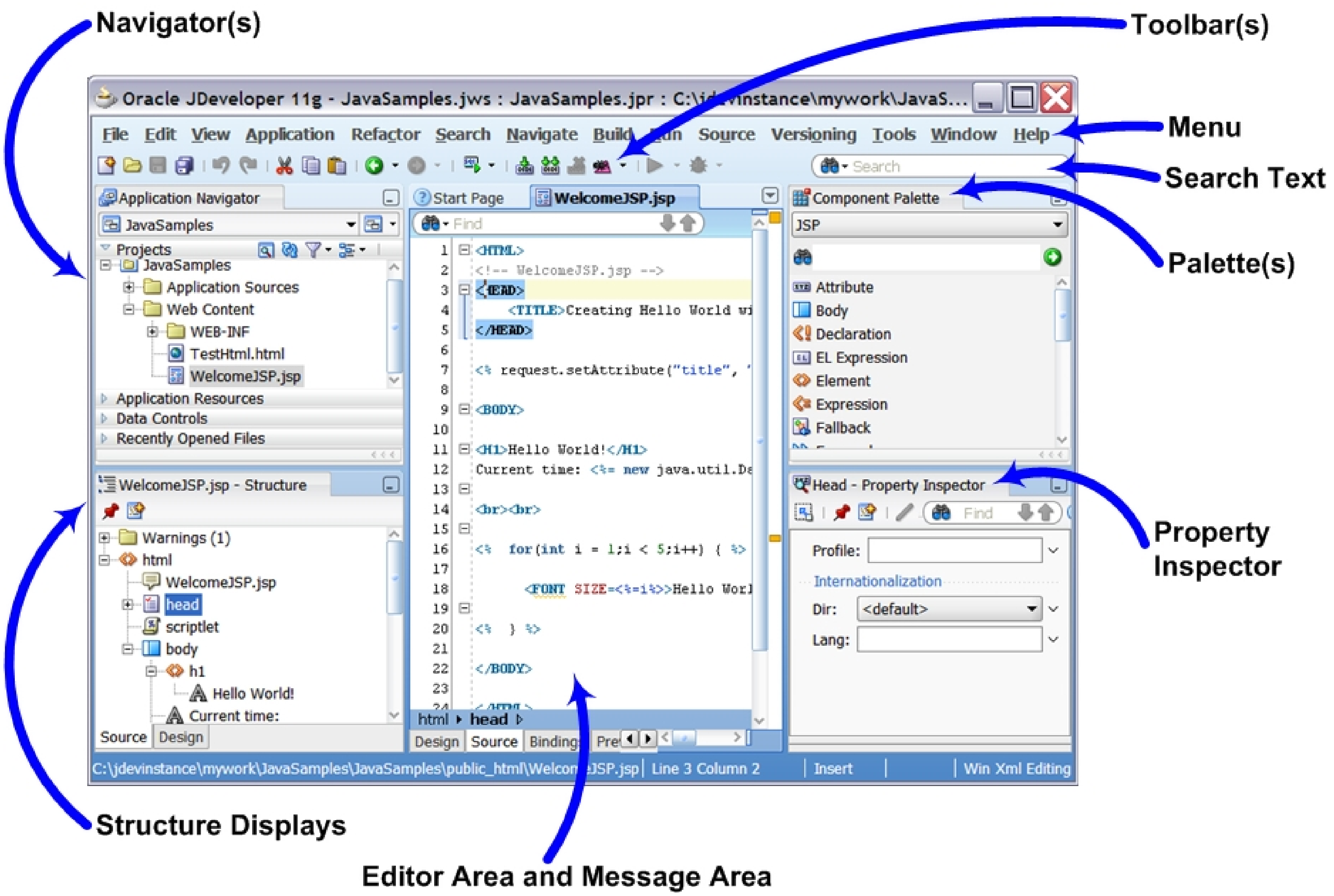
- [✓] **Default Role** Enables all technologies.
- **Customization Developer** Configures the product for customizing metadata
- **Database Edition** Includes only features for core database development.
- **Java EE Edition** Includes only features for core Java EE development.
- **Java Edition** Includes only features for core Java development.

☑ Always prompt for role selection on startup

JDeveloper - Start Page


Oracle JDeveloper 11g : Start Page
File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help
Search
Application Navigator
New Application...
Open Application...
Start Page - Structure
No Structure
Start Page
Oracle JDeveloper 11g
Learn how to...
Model Applications
Design Databases
Build Applications
Integrate Applications
What's New | Release Notes | Samples | Discussion Forum | Books and Training | Online Resources
Copyright © 1997, 2009 Oracle and/or its affiliates. All rights reserved.
Editing

Exploring JDeveloper

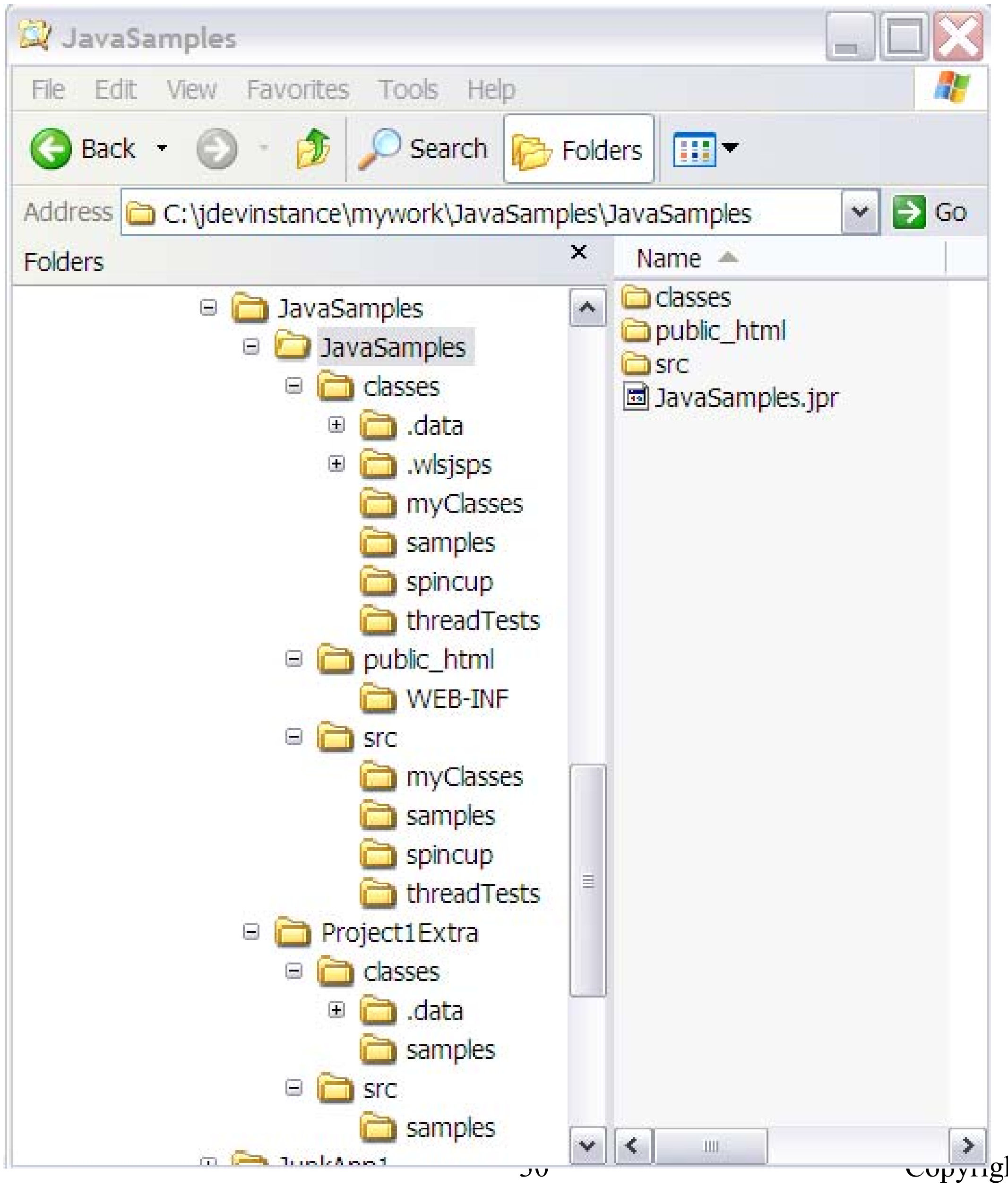


```
<HTML>
<!-- WelcomeJSP.jsp -->
<HEAD>
    <TITLE>Creating Hello World wi
</HEAD>

<% request.setAttribute("title", '

<BODY>

<H1>Hello World!</H1>
Current time: <%= new java.util.De

<br><br>

<%  for(int i = 1;i < 5;i++) { %>

      <FONT SIZE=<%=i%>>Hello Worl

<%  } %>

</BODY>

</HTML>
```

Applications and Projects

- JDeveloper uses a non-standard, Oracle-specific "Application" to group a collection of "Projects"
- 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



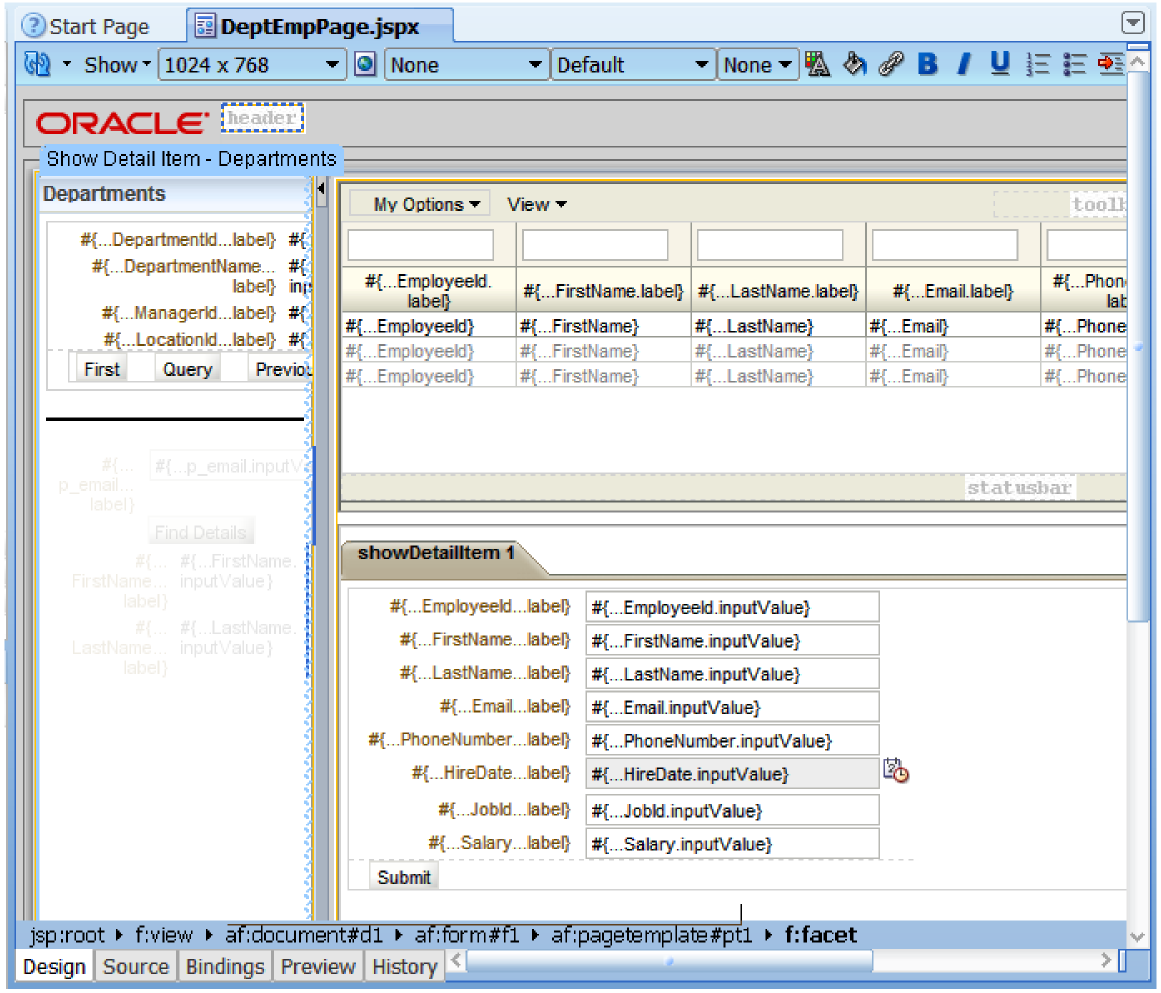
Address C:\jdevinstance\mywork\JavaSamples\JavaSamples

```
⊟ JavaSamples
    ⊟ JavaSamples
        ⊟ classes
            ⊞ .data
            ⊞ .wlsjsps
              myClasses
              samples
              spincup
              threadTests
        ⊟ public_html
              WEB-INF
        ⊟ src
              myClasses
              samples
              spincup
              threadTests
    ⊟ Project1Extra
        ⊟ classes
            ⊞ .data
              samples
        ⊟ src
              samples
  ⊞ JunkApp1
```

```
classes
public_html
src
JavaSamples.jpr
```

JDeveloper Editing

- JDeveloper has a variety of Code Editors and Visual Editors; including: Java, XML, HTML, JSP, JSF/ADF Faces, BPEL, and more



#{...DepartmentId...label} #{
#{...DepartmentName... label} #{ inp
#{...ManagerId...label} #{
#{...LocationId...label} #{

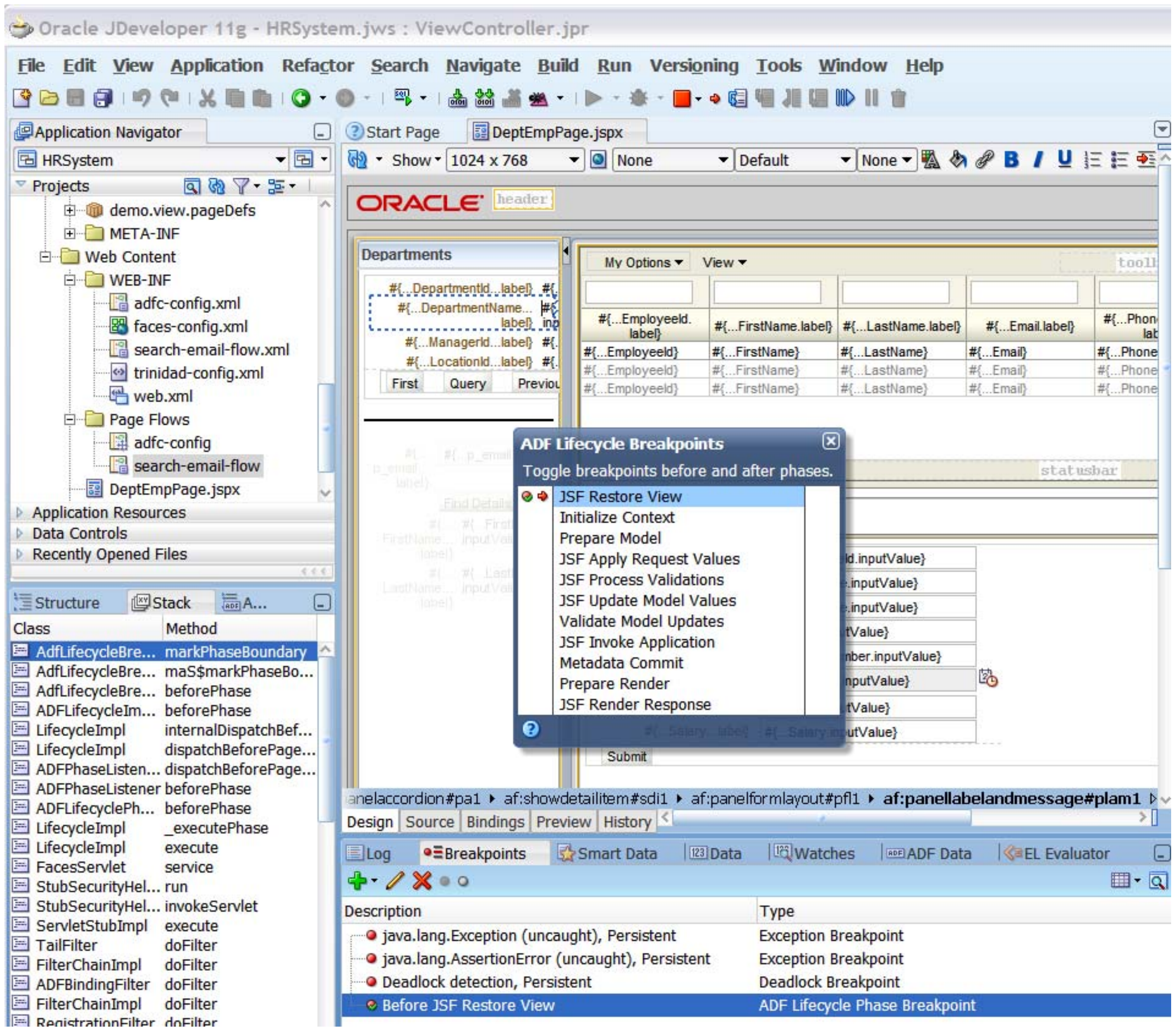
#{...EmployeeId. label}	#{...FirstName.label}	#{...LastName.label}	#{...Email.label}	#{...Phon lab
#{...EmployeeId}	#{...FirstName}	#{...LastName}	#{...Email}	#{...Phone
#{...EmployeeId}	#{...FirstName}	#{...LastName}	#{...Email}	#{...Phone
#{...EmployeeId}	#{...FirstName}	#{...LastName}	#{...Email}	#{...Phone

#{...EmployeeId...label}	#{...EmployeeId.inputValue}
#{...FirstName...label}	#{...FirstName.inputValue}
#{...LastName...label}	#{...LastName.inputValue}
#{...Email...label}	#{...Email.inputValue}
#{...PhoneNumber...label}	#{...PhoneNumber.inputValue}
#{...HireDate...label}	#{...HireDate.inputValue}
#{...JobId...label}	#{...JobId.inputValue}
#{...Salary...label}	#{...Salary.inputValue}

jsp:root ▸ f:view ▸ af:document#d1 ▸ af:form#f1 ▸ af:pagetemplate#pt1 ▸ **f:facet**

JDeveloper Debugging

- JDeveloper allows both local and remote debugging



ADF Lifecycle Breakpoints

Toggle breakpoints before and after phases.

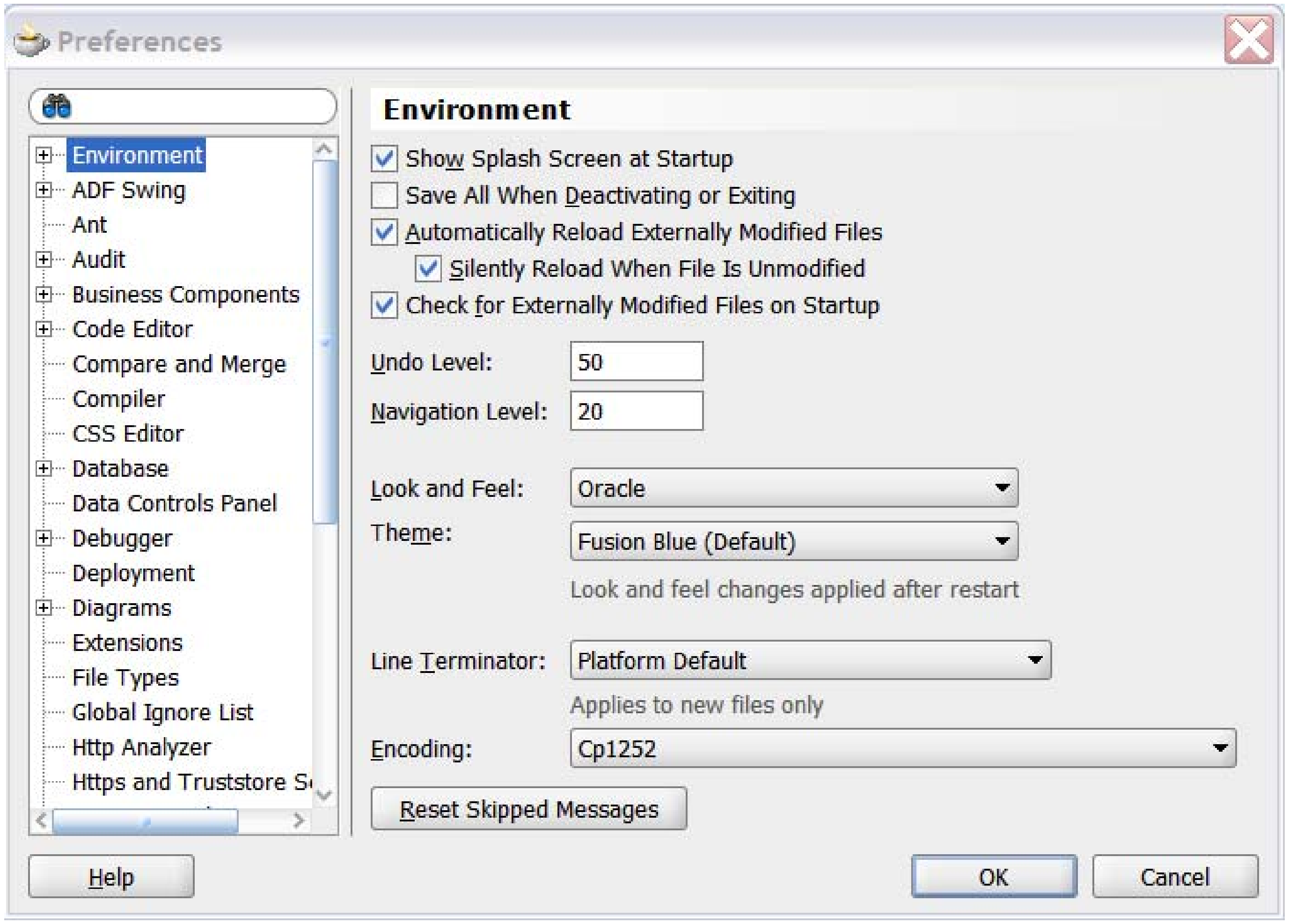
- JSF Restore View
- Initialize Context
- Prepare Model
- JSF Apply Request Values
- JSF Process Validations
- JSF Update Model Values
- Validate Model Updates
- JSF Invoke Application
- Metadata Commit
- Prepare Render
- JSF Render Response

Class	Method
AdfLifecycleBre...	markPhaseBoundary
AdfLifecycleBre...	maS$markPhaseBo...
AdfLifecycleBre...	beforePhase
ADFLifecycleIm...	beforePhase
LifecycleImpl	internalDispatchBef...
LifecycleImpl	dispatchBeforePhase...
ADFPhaseListen...	dispatchBeforePage...
ADFPhaseListener	beforePhase
ADFLifecyclePh...	beforePhase
LifecycleImpl	_executePhase
LifecycleImpl	execute
FacesServlet	service
StubSecurityHel...	run
StubSecurityHel...	invokeServlet
ServletStubImpl	execute
TailFilter	doFilter
FilterChainImpl	doFilter
ADFBindingFilter	doFilter
FilterChainImpl	doFilter
RegistrationFilter	doFilter

Description	Type
java.lang.Exception (uncaught), Persistent	Exception Breakpoint
java.lang.AssertionError (uncaught), Persistent	Exception Breakpoint
Deadlock detection, Persistent	Deadlock Breakpoint
Before JSF Restore View	ADF Lifecycle Phase Breakpoint

JDeveloper Preferences

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



Environment

- [✓] Show Splash Screen at Startup
- [ ] Save All When Deactivating or Exiting
- [✓] Automatically Reload Externally Modified Files
  - [✓] Silently Reload When File Is Unmodified
- [✓] Check for Externally Modified Files on Startup

Undo Level: 50

Navigation Level: 20

Look and Feel: Oracle

Theme: Fusion Blue (Default)

Look and feel changes applied after restart

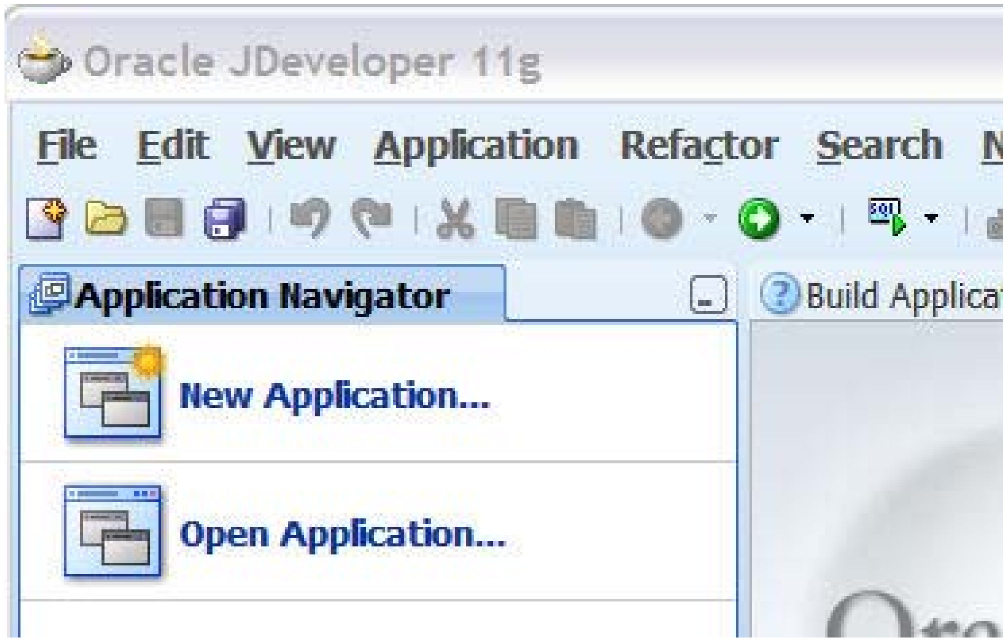
Line Terminator: Platform Default

Applies to new files only

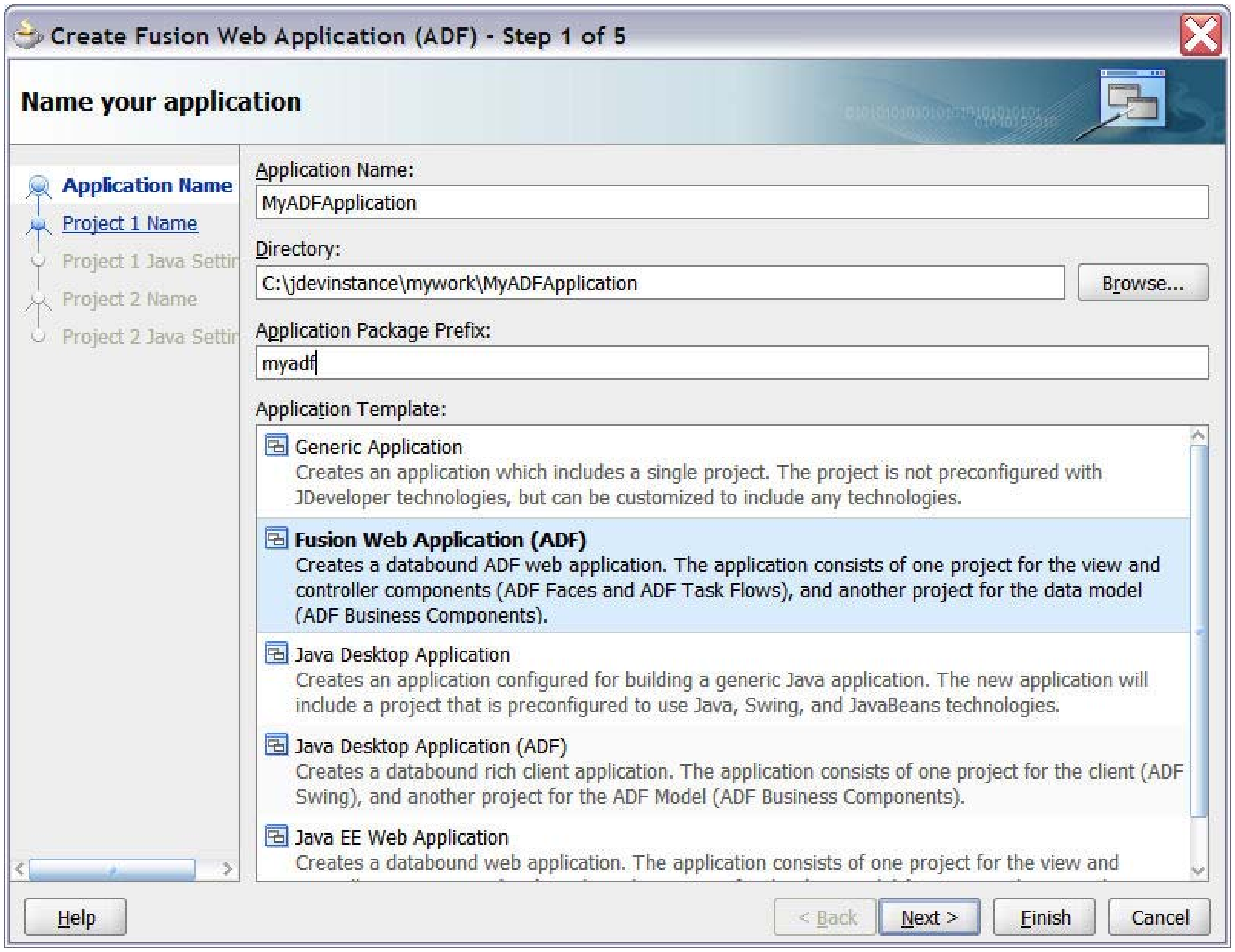
Encoding: Cp1252

New Application

- To create a new application use the JDeveloper menu's File->New->General->Applications option



New Gallery



Create Fusion Web Application (ADF) - Step 1 of 5

Name your application

Application Name: MyADFApplication

Directory: C:\jdevinstance\mywork\MyADFApplication

Application Package Prefix: myadf

Application Template:

Generic Application
Creates an application which includes a single project. The project is not preconfigured with JDeveloper technologies, but can be customized to include any technologies.

Fusion Web Application (ADF)
Creates a databound ADF web application. The application consists of one project for the view and controller components (ADF Faces and ADF Task Flows), and another project for the data model (ADF Business Components).

Java Desktop Application
Creates an application configured for building a generic Java application. The new application will include a project that is preconfigured to use Java, Swing, and JavaBeans technologies.

Java Desktop Application (ADF)
Creates a databound rich client application. The application consists of one project for the client (ADF Swing), and another project for the ADF Model (ADF Business Components).

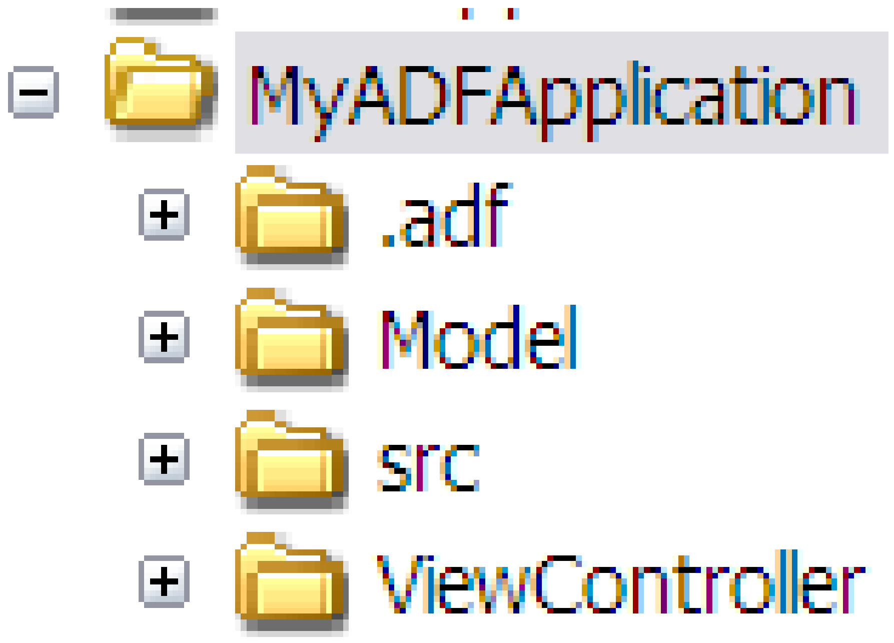
Java EE Web Application
Creates a databound web application. The application consists of one project for the view and

Application Structure

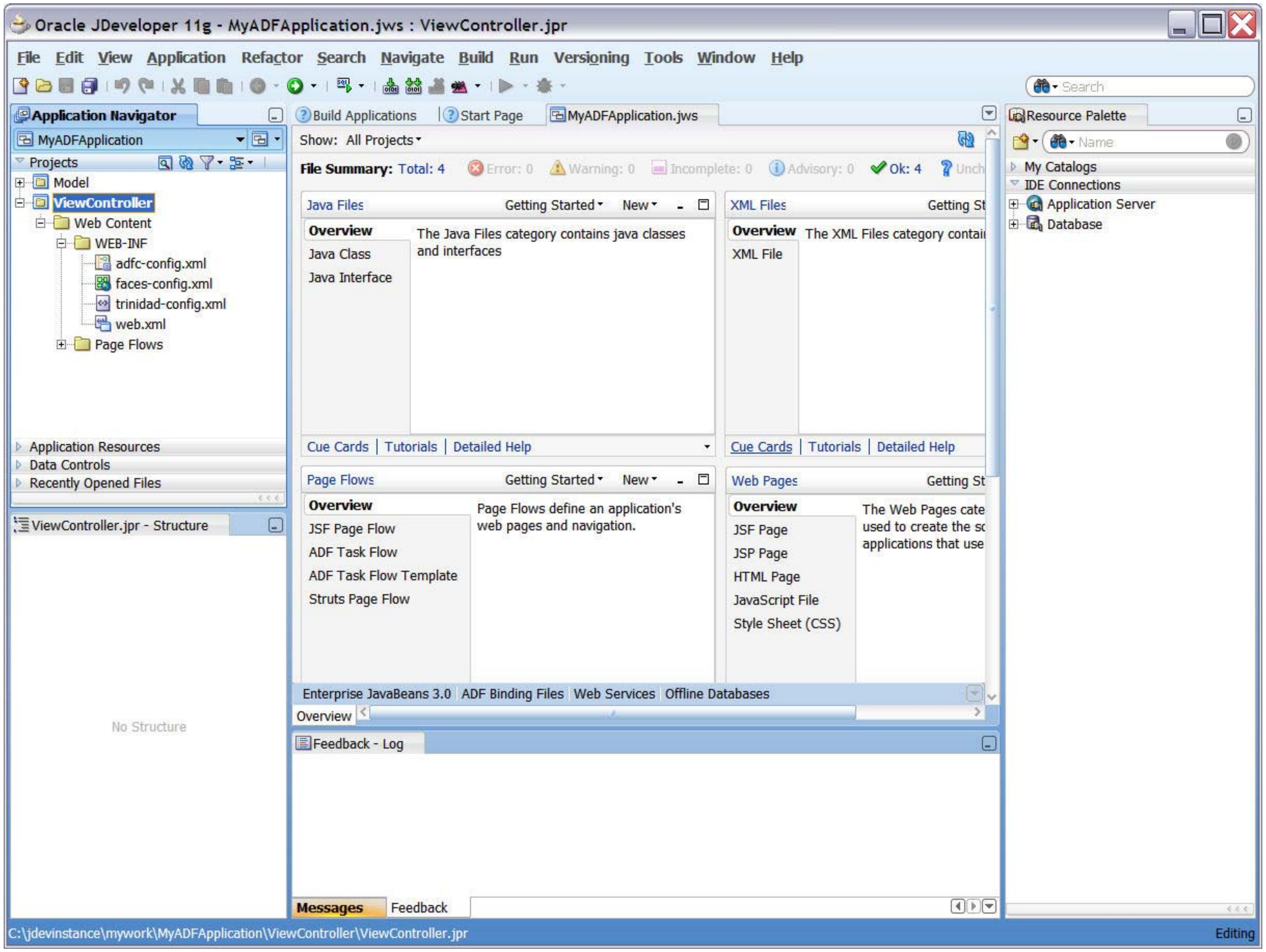
- When a JDeveloper ADF Web Application is created ADF uses the MVC (Model-View Controller) pattern
- JDeveloper creates two subordinate projects
  - Model — Data and Business Rules
  - ViewController — User Interface
  - ADF provides the "Controller"

File Structure

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



How It Looks In JDeveloper



Create ADF BC Objects

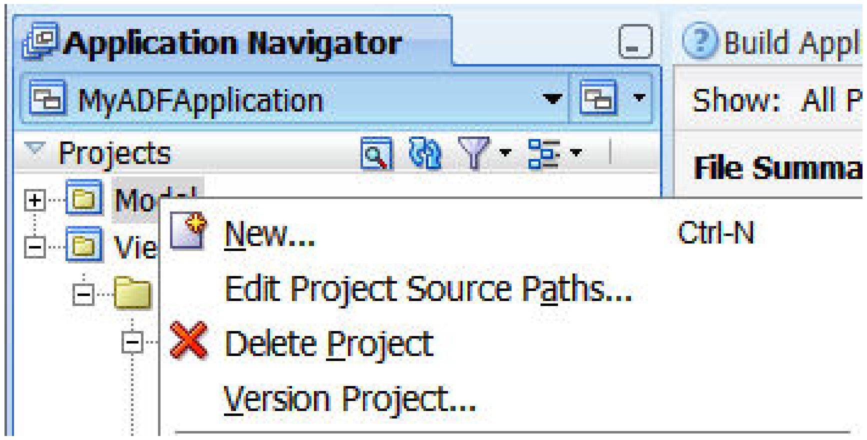
- The following pages show how to create ADF BC objects using the Wizards provided by JDeveloper
- Each object created may be created individually using JDeveloper's features or by coding them manually rather than using the Wizards
- JDeveloper's database modeling capabilities are shown to good effect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

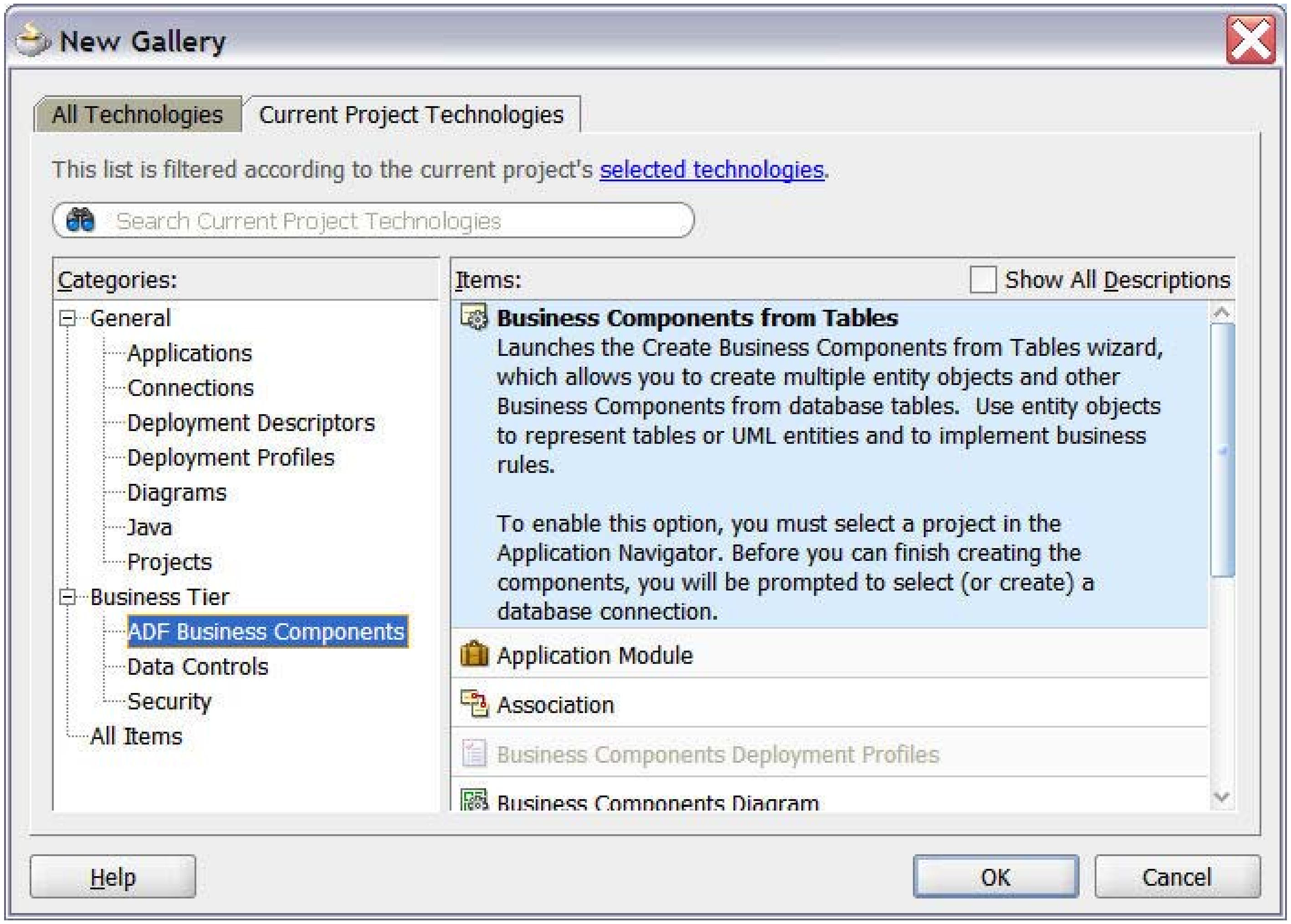
New ADF BC Object: 1

- 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



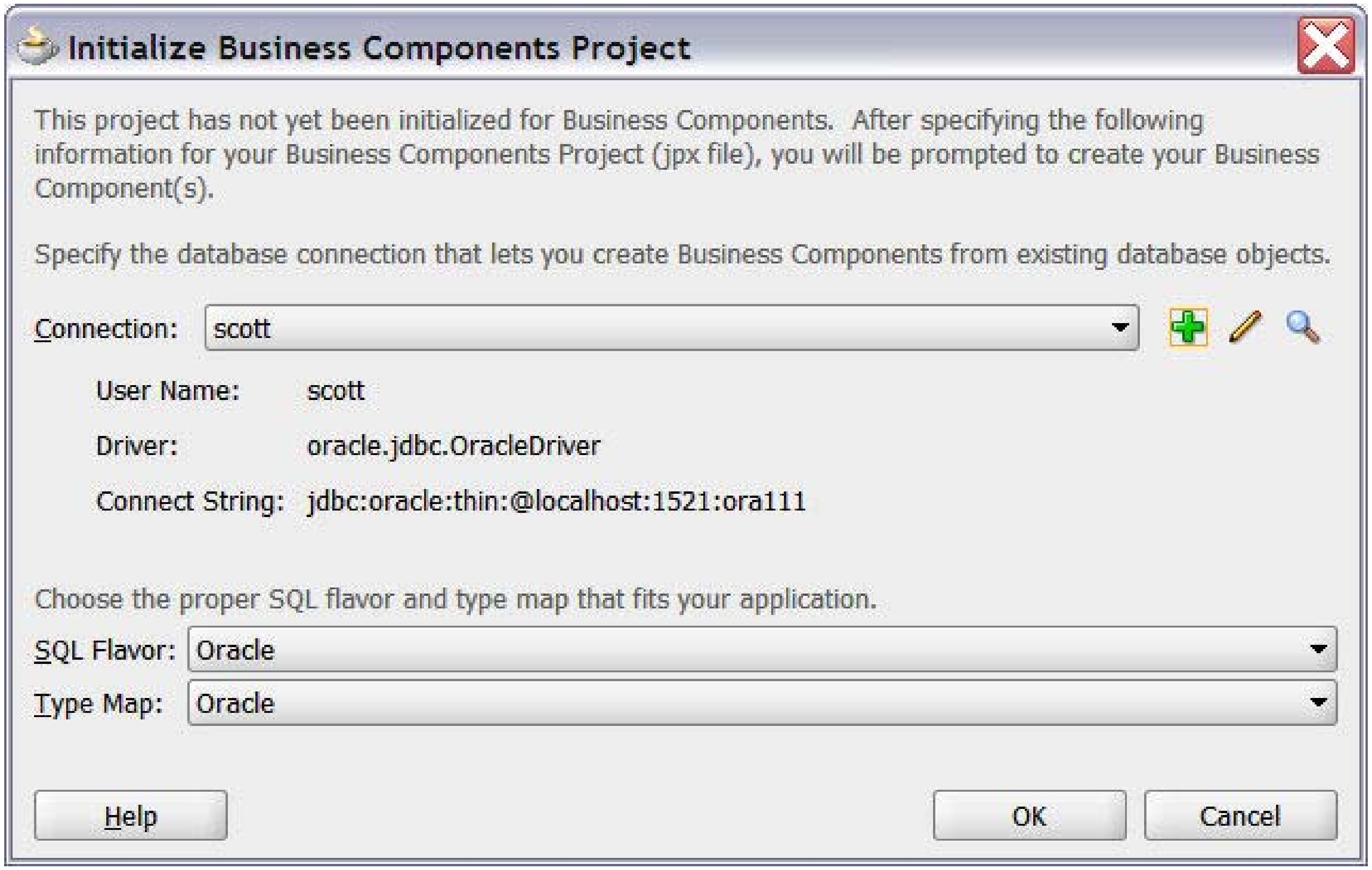
This list is filtered according to the current project's selected technologies.

Business Components from Tables
Launches the Create Business Components from Tables wizard, which allows you to create multiple entity objects and other Business Components from database tables. Use entity objects to represent tables or UML entities and to implement business rules.

To enable this option, you must select a project in the Application Navigator. Before you can finish creating the components, you will be prompted to select (or create) a database connection.

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 tnsname.or not required!)



This project has not yet been initialized for Business Components. After specifying the following information for your Business Components Project (jpx file), you will be prompted to create your Business Component(s).

Specify the database connection that lets you create Business Components from existing database objects.

Connection: scott

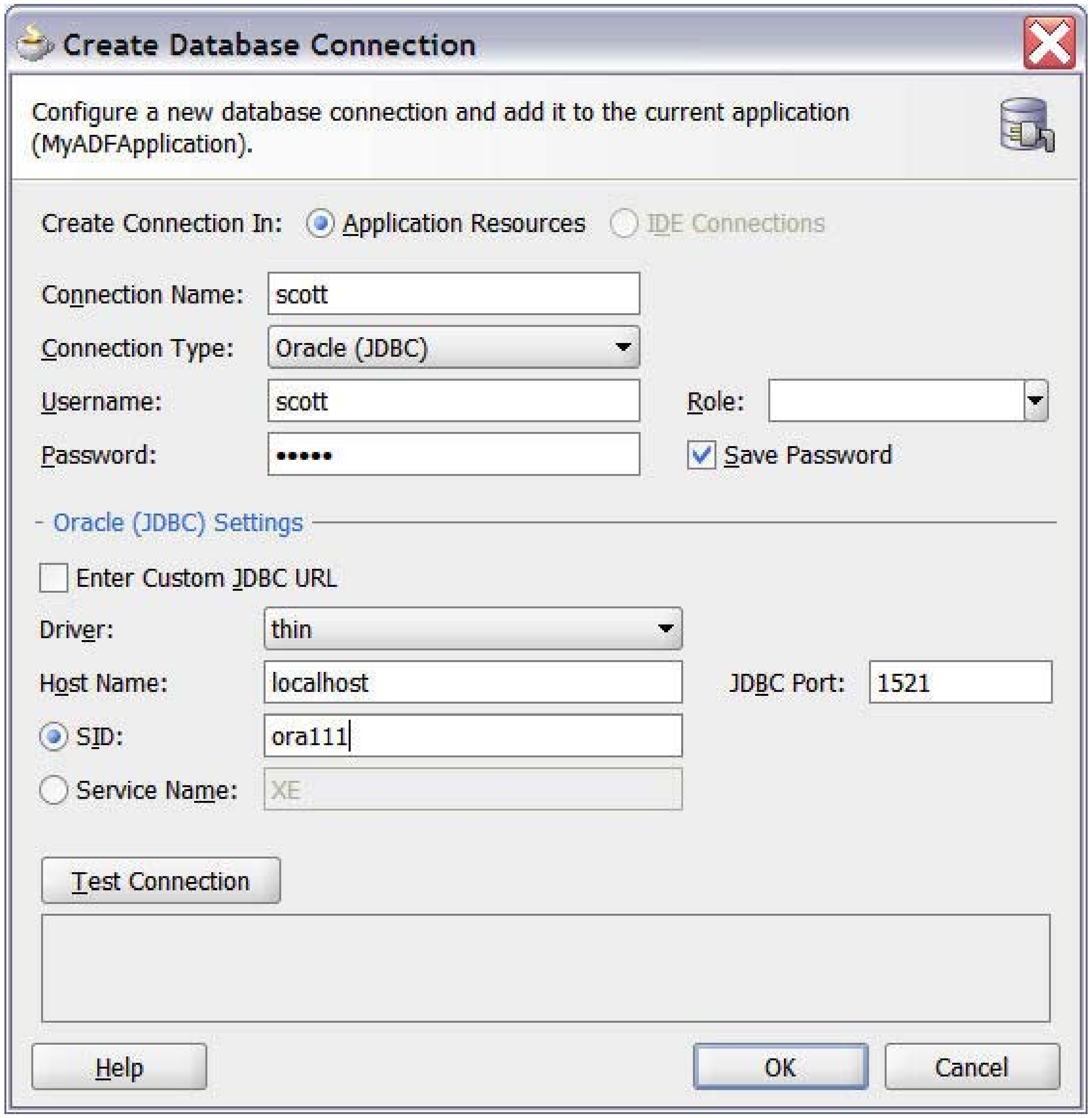
User Name:	scott
Driver:	oracle.jdbc.OracleDriver
Connect String:	jdbc:oracle:thin:@localhost:1521:ora111

Choose the proper SQL flavor and type map that fits your application.

SQL Flavor: Oracle

Type Map: Oracle

Create Database Connection



Create Database Connection

Configure a new database connection and add it to the current application (MyADFApplication).

Create Connection In: [✓ Application Resources] IDE Connections

Connection Name:	scott		
Connection Type:	Oracle (JDBC)		
Username:	scott	Role:	
Password:	•••••	✓ Save Password	

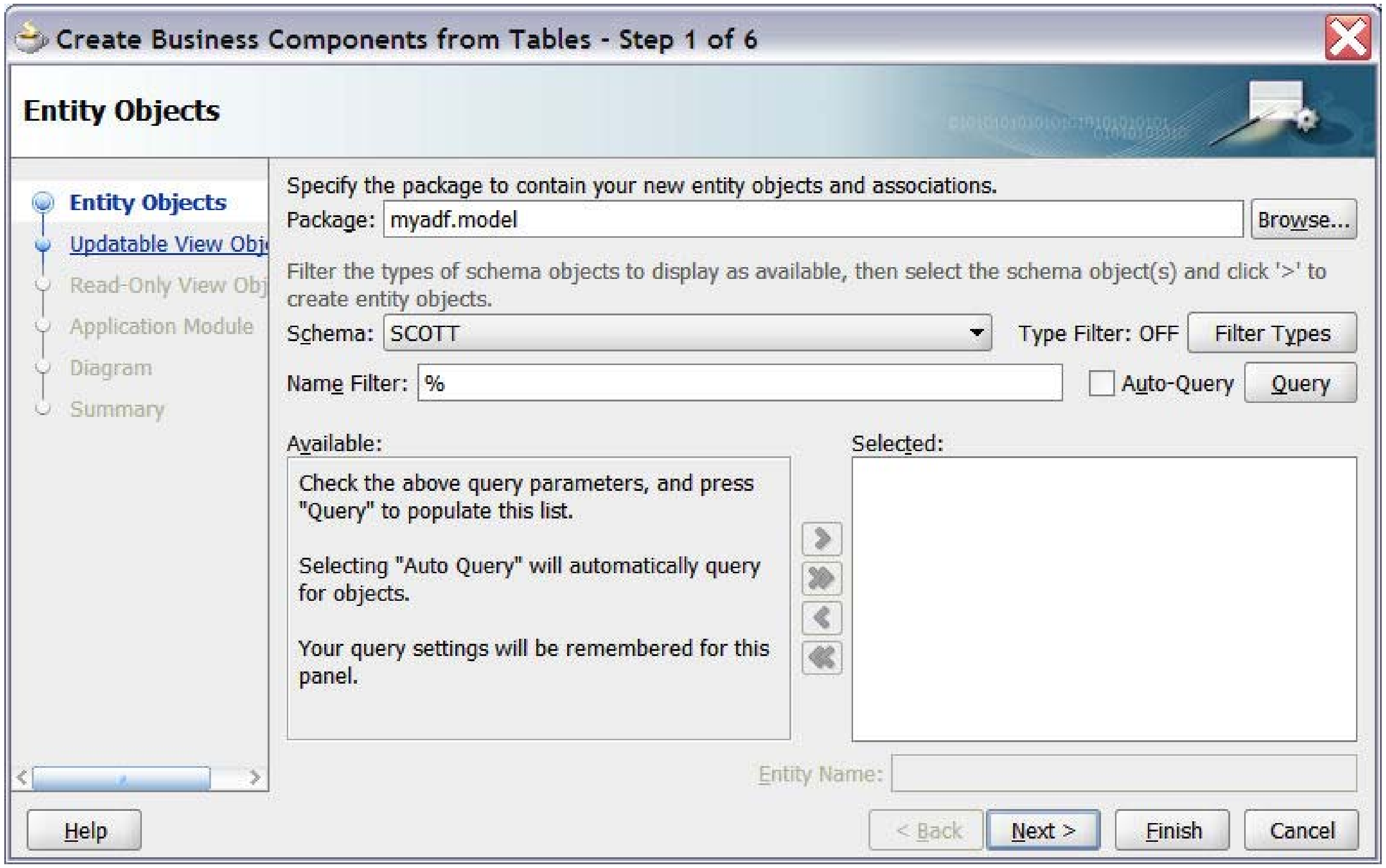
Oracle (JDBC) Settings

☐ Enter Custom JDBC URL

Driver:	thin		
Host Name:	localhost	JDBC Port:	1521
[✓ SID:]	ora111		
Service Name:	XE		

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



Specify the package to contain your new entity objects and associations.
Package: myadf.model

Filter the types of schema objects to display as available, then select the schema object(s) and click '>' to create entity objects.
Schema: SCOTT
Type Filter: OFF
Name Filter: %

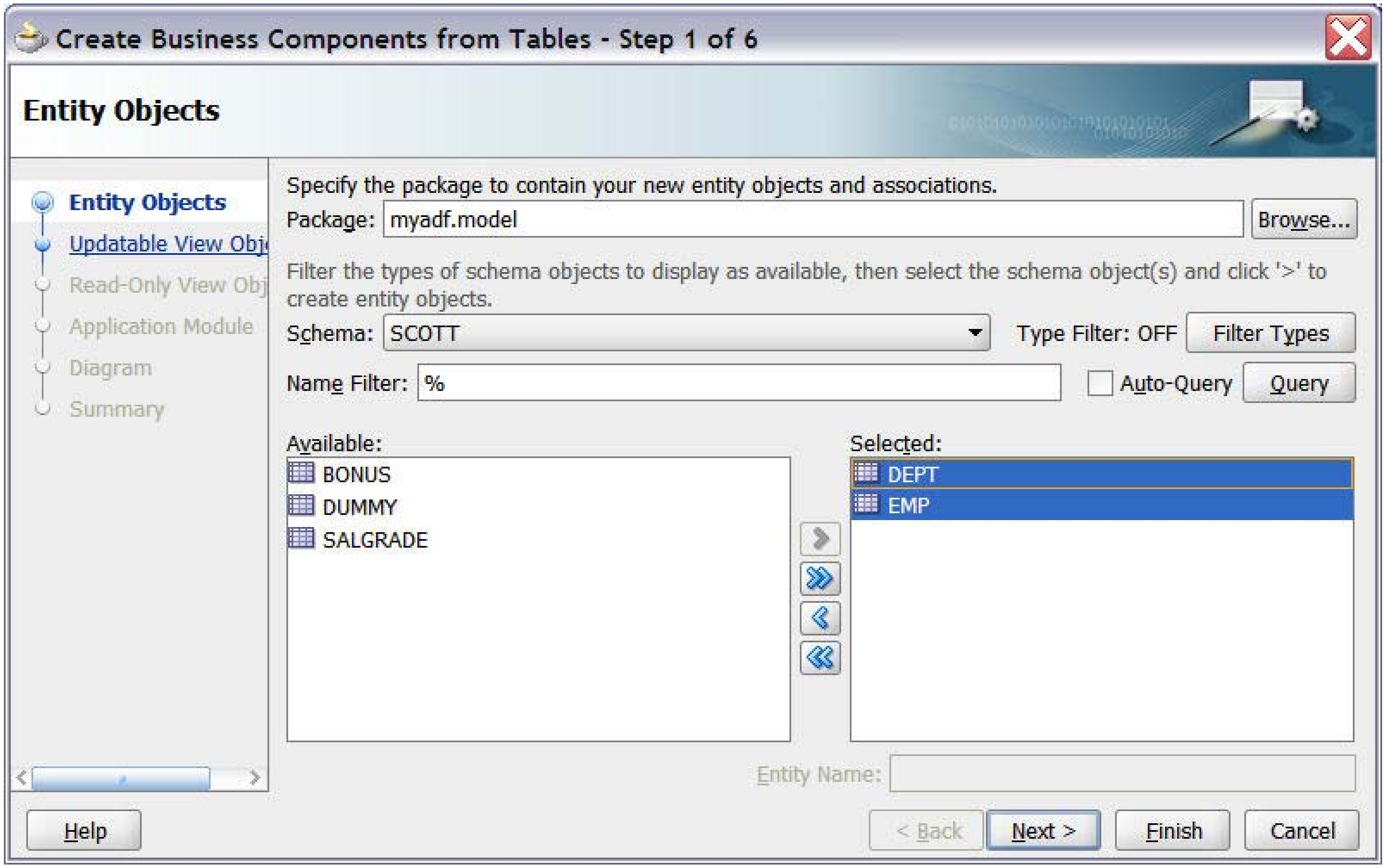
Check the above query parameters, and press "Query" to populate this list.

Selecting "Auto Query" will automatically query for objects.

Your query settings will be remembered for this panel.

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 display



Create Business Components from Tables - Step 1 of 6

Entity Objects

Specify the package to contain your new entity objects and associations.

Package: myadf.model

Filter the types of schema objects to display as available, then select the schema object(s) and click '>' to create entity objects.

Schema: SCOTT

Type Filter: OFF

Name Filter: %

Auto-Query

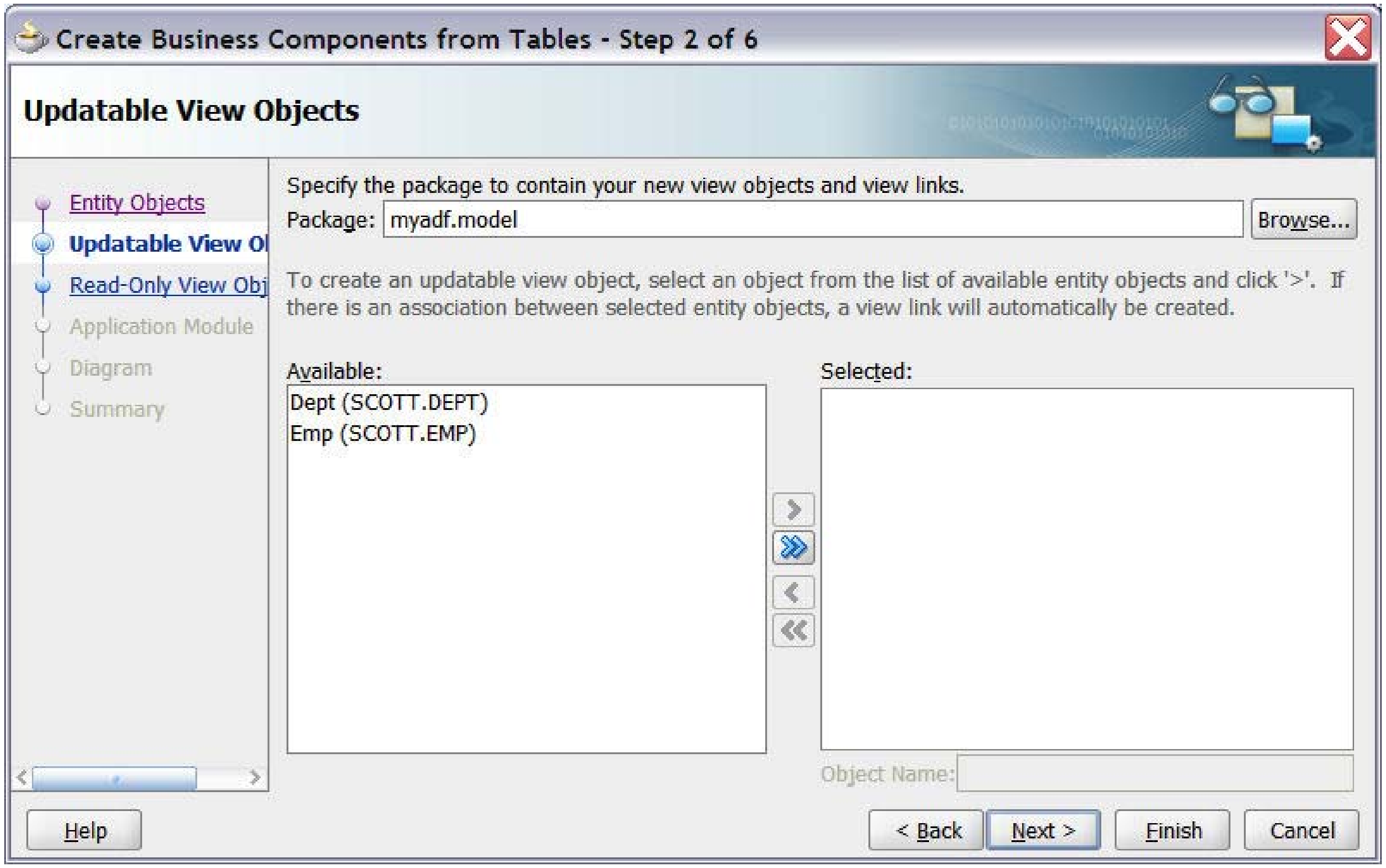
Available:
BONUS
DUMMY
SALGRADE

Selected:
DEPT
EMP

Entity Name:

Create BC from Tables, 3

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



Create Business Components from Tables - Step 2 of 6

Updatable View Objects

Specify the package to contain your new view objects and view links.

Package: myadf.model

To create an updatable view object, select an object from the list of available entity objects and click '>'. If there is an association between selected entity objects, a view link will automatically be created.

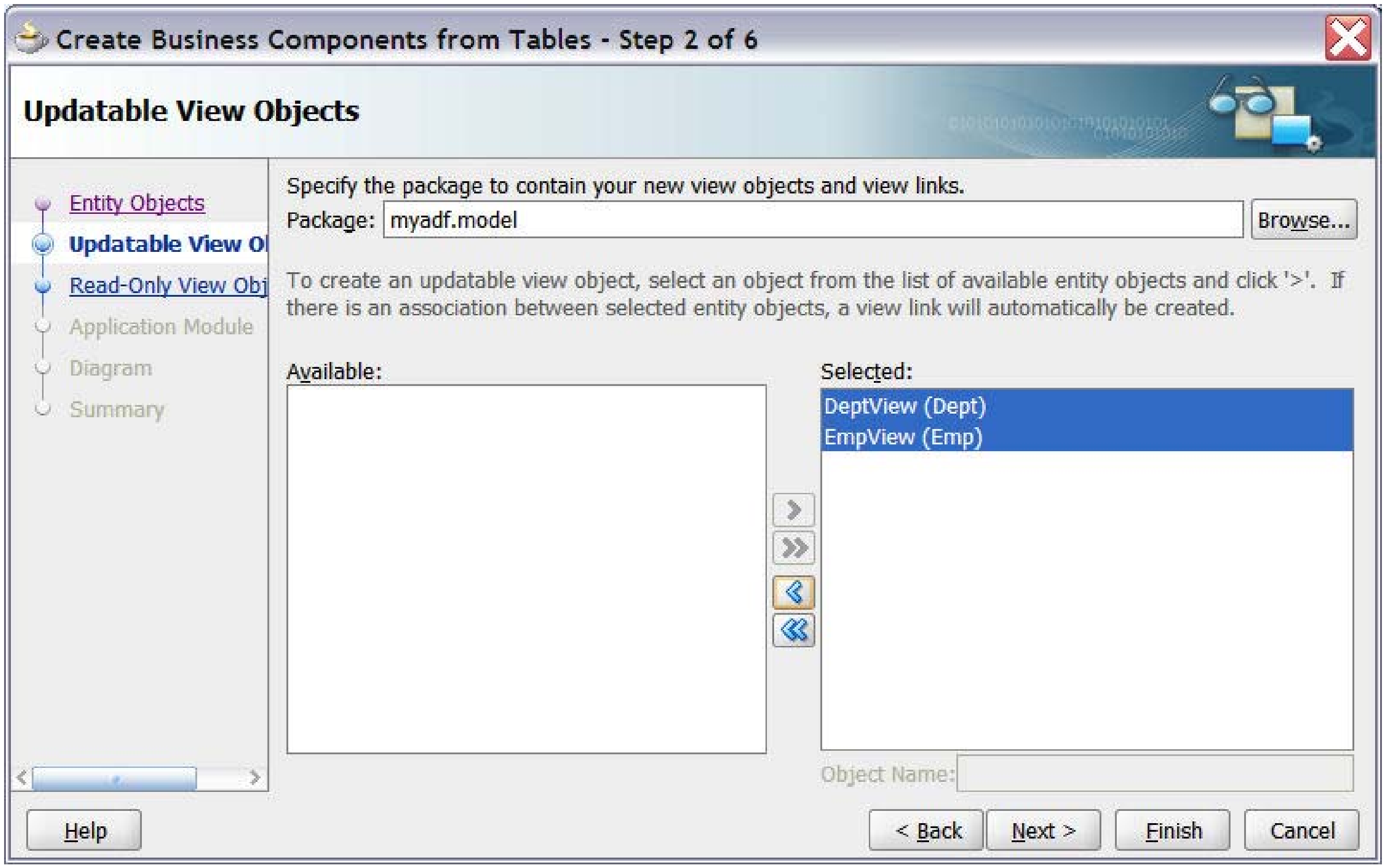
Available:
Dept (SCOTT.DEPT)
Emp (SCOTT.EMP)

Selected:

Object Name:

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 Business Components from Tables - Step 2 of 6

Updatable View Objects

Specify the package to contain your new view objects and view links.

Package: myadf.model

To create an updatable view object, select an object from the list of available entity objects and click '>'. If there is an association between selected entity objects, a view link will automatically be created.

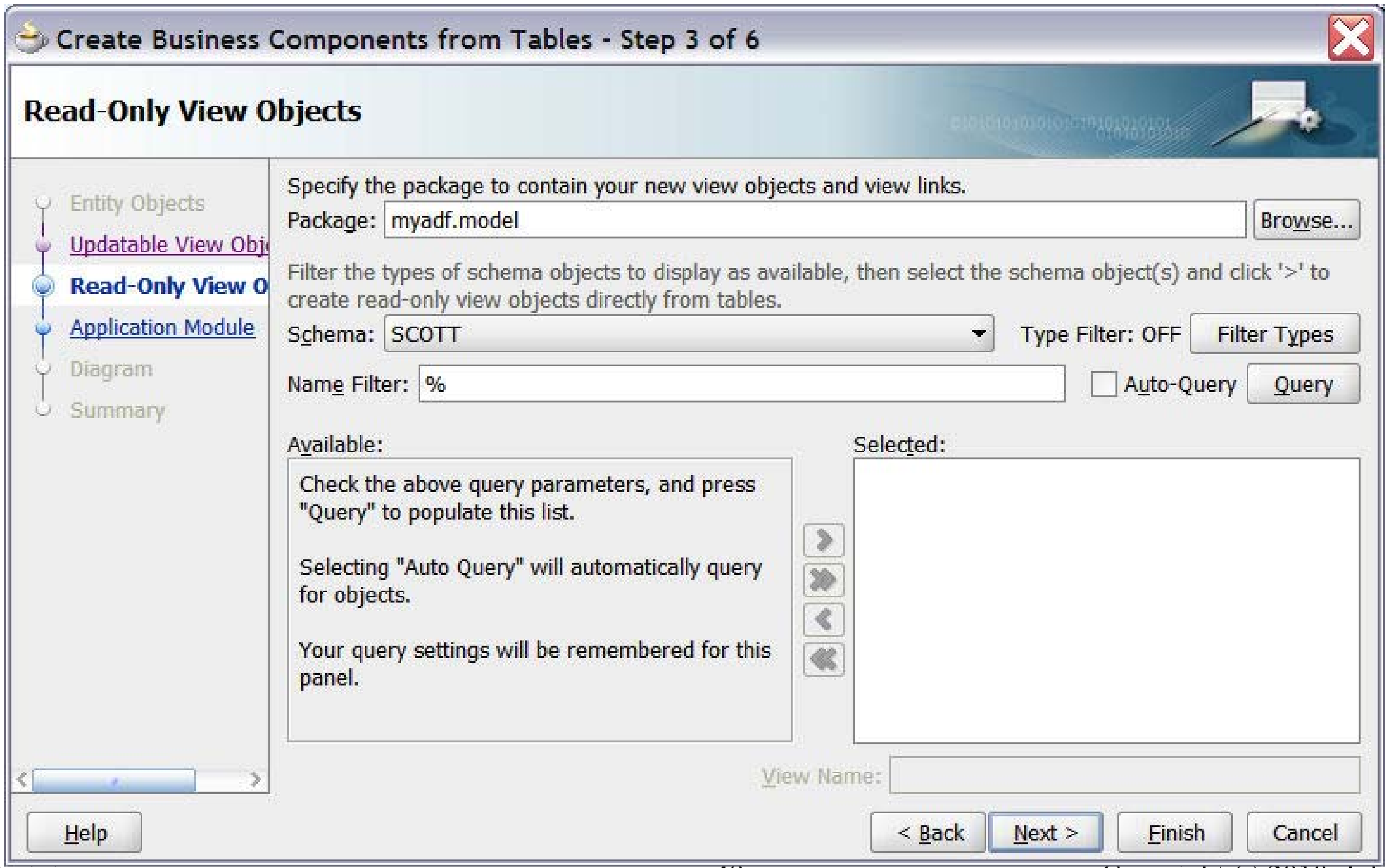
Available:

Selected:
DeptView (Dept)
EmpView (Emp)

Object Name:

Create BC from Tables, 5

- After creating Updateable View Objects; the wizard goes on to create Read-Only View Objects (might be useful to support an LOV (List-of-Values))


Create Business Components from Tables - Step 3 of 6

Read-Only View Objects

Specify the package to contain your new view objects and view links.

Package: myadf.model

Filter the types of schema objects to display as available, then select the schema object(s) and click '>' to create read-only view objects directly from tables.

Schema: SCOTT

Type Filter: OFF

Name Filter: %

Available:

Check the above query parameters, and press "Query" to populate this list.

Selecting "Auto Query" will automatically query for objects.

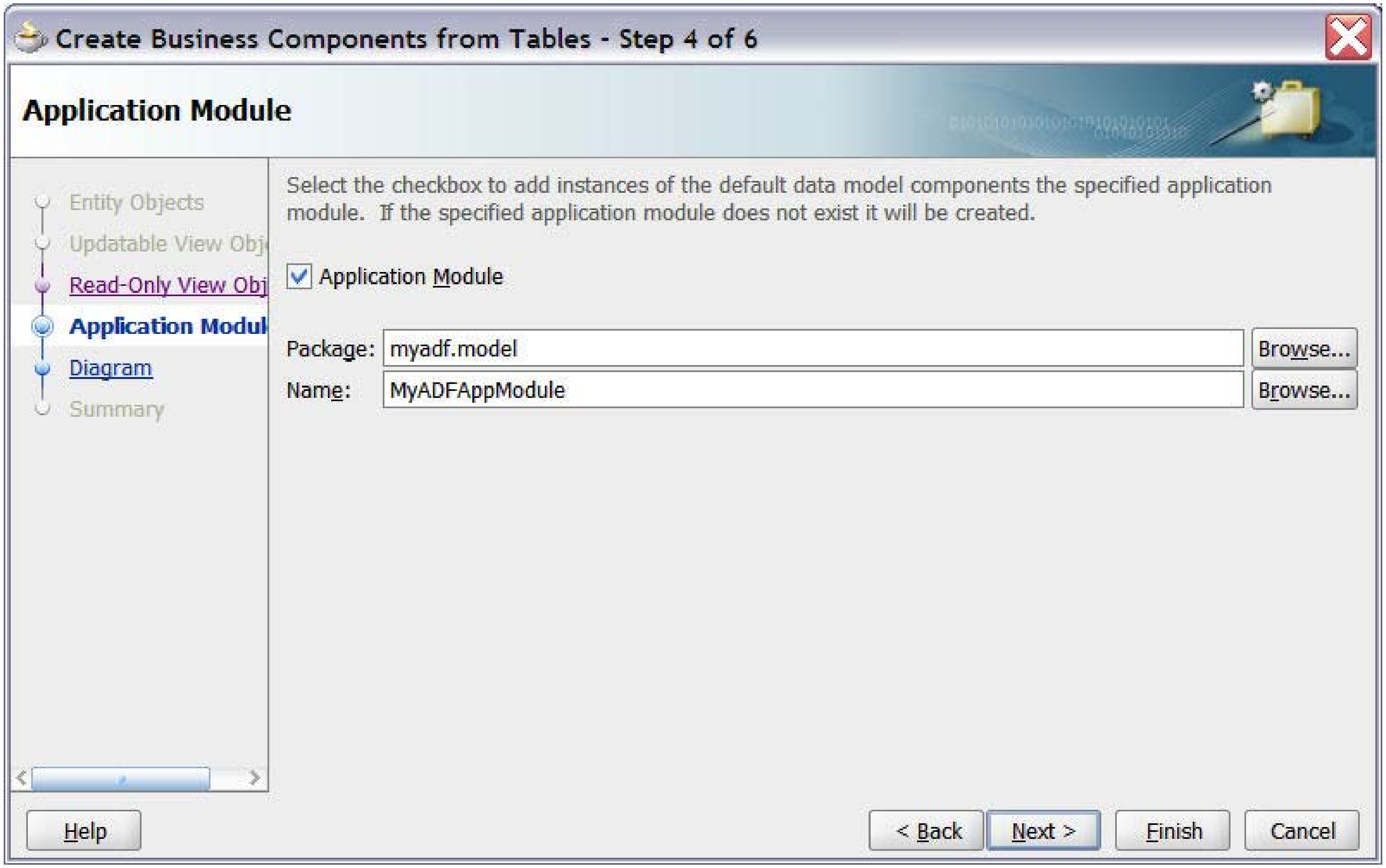
Your query settings will be remembered for this panel.

Selected:

View Name:

Create BC from Tables, 6

- Name the Application Module and save it; click Finish



Create Business Components from Tables - Step 4 of 6

Application Module

Select the checkbox to add instances of the default data model components the specified application module. If the specified application module does not exist it will be created.

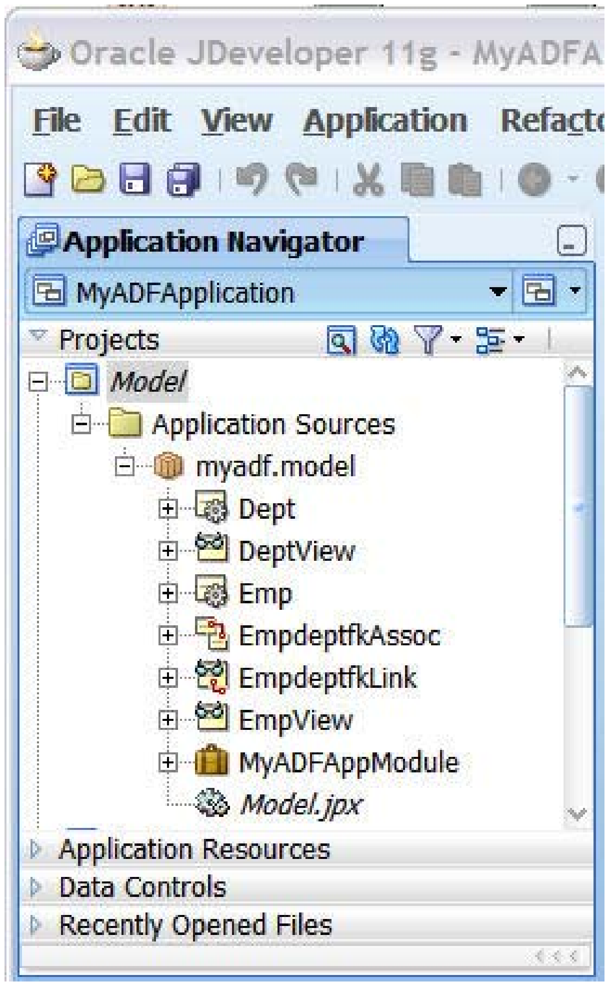
☑ Application Module

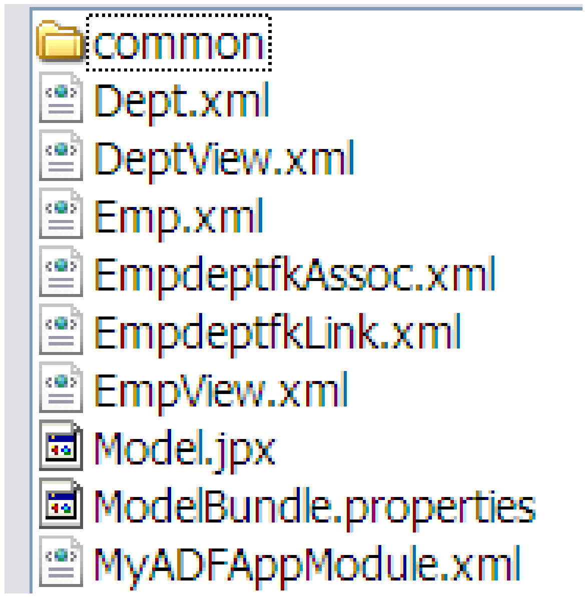
Package: myadf.model

Name: MyADFAppModule

Business Component Files

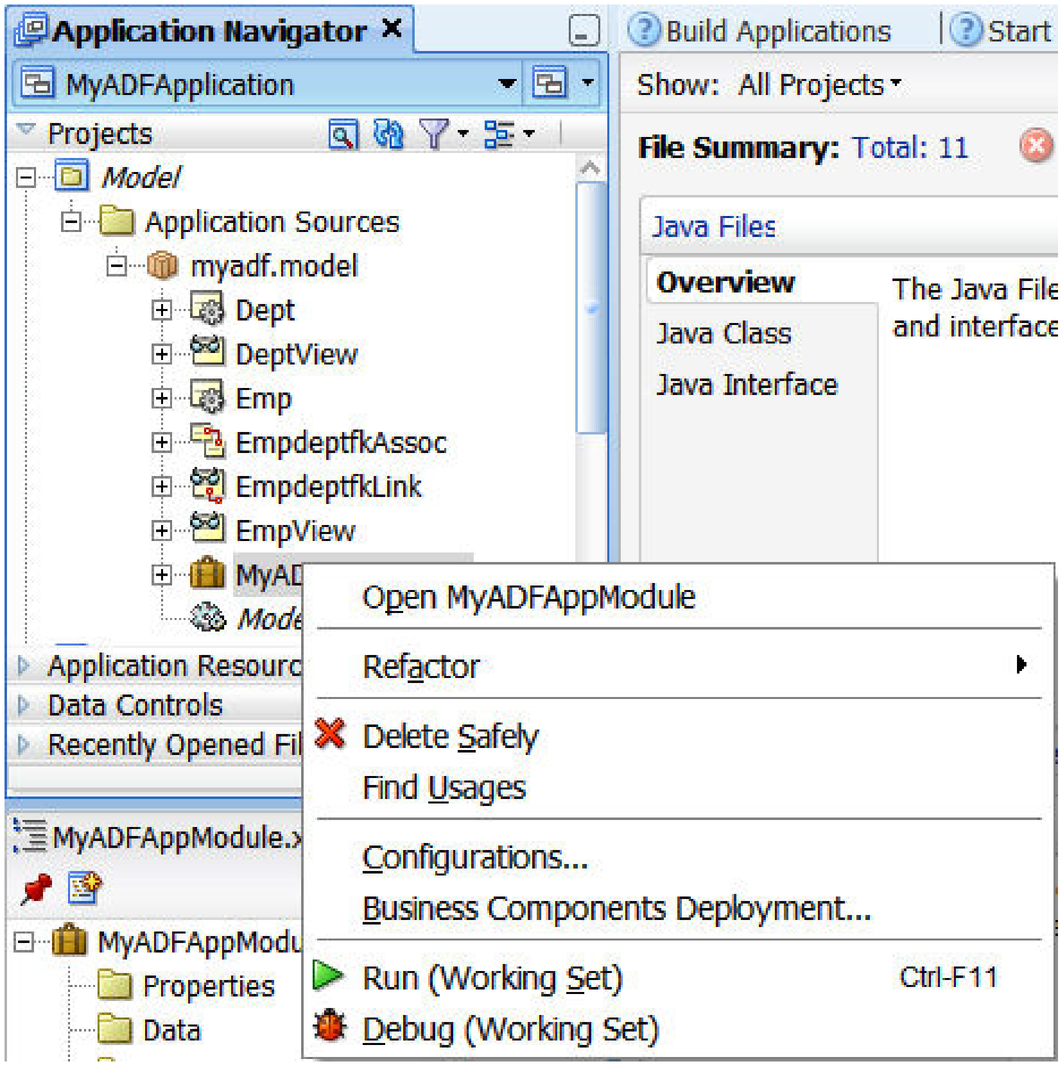
- Note the use of XML to declaratively support ADF BC




common
Dept.xml
DeptView.xml
Emp.xml
EmpdeptfkAssoc.xml
EmpdeptfkLink.xml
EmpView.xml
Model.jpx
ModelBundle.properties
MyADFAppModule.xml

Business Component Browser

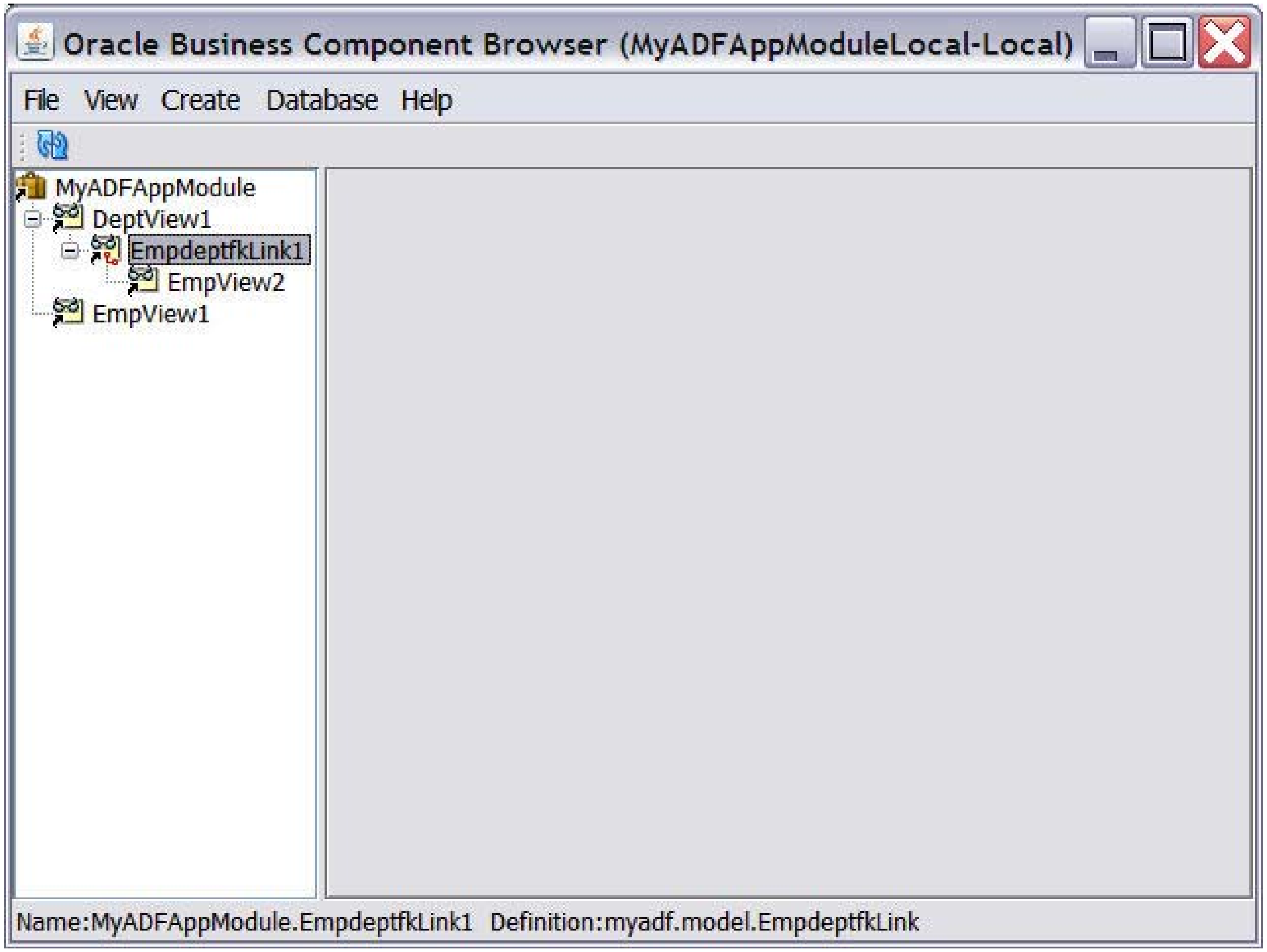
- JDeveloper provides a tool to “browse” ADF BC Application Module objects graphically; using the Application Navigator, find the Application Module to be viewed; right-click and choose “Run” to start



Training Resources

Component Browser Choices

- Choose the Business Component to be tested

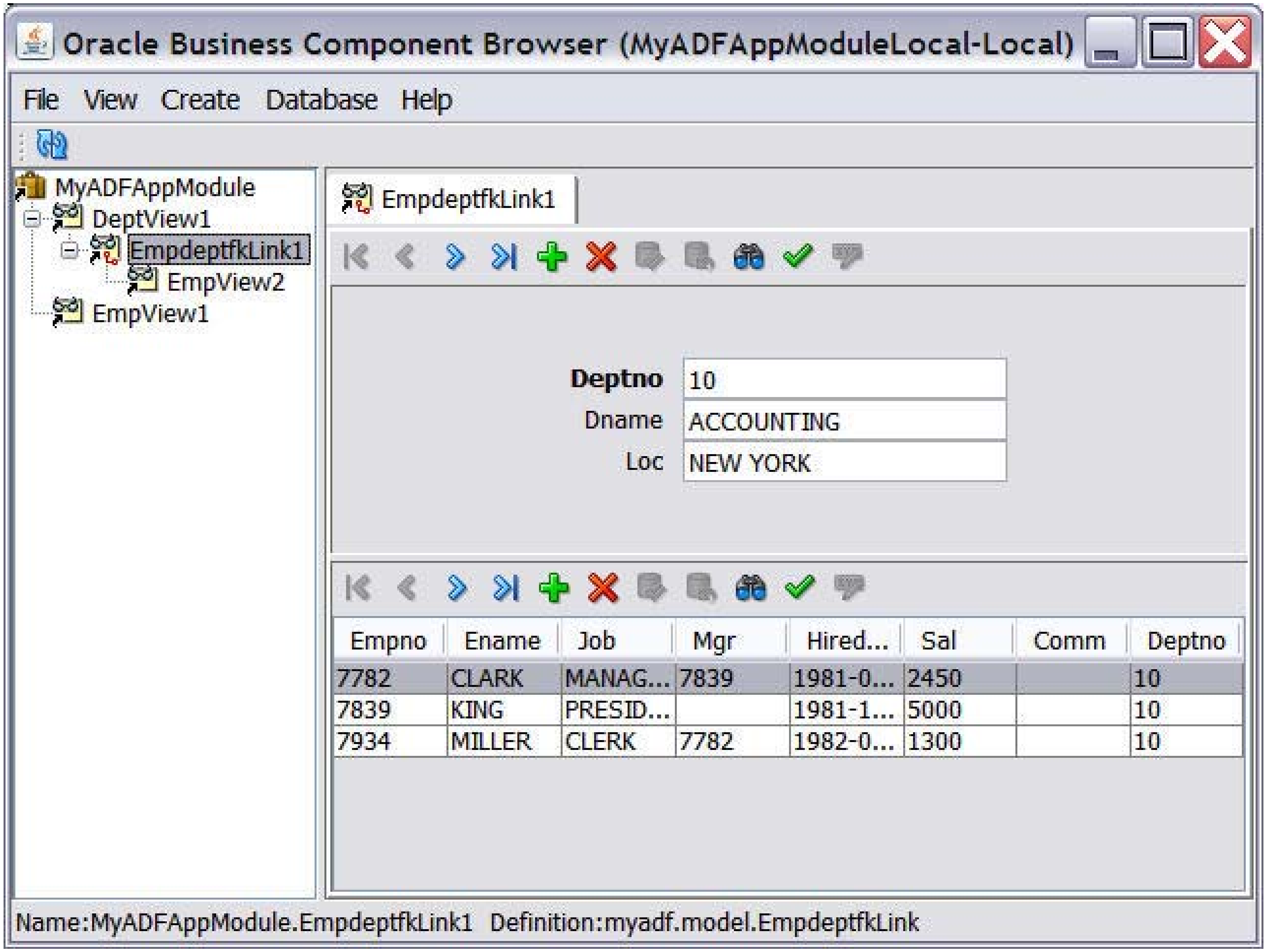

Oracle Business Component Browser (MyADFAppModuleLocal-Local)
File View Create Database Help
MyADFAppModule
DeptView1
EmpdeptfkLink1
EmpView2
EmpView1

Name:MyADFAppModule.EmpdeptfkLink1 Definition:myadf.model.EmpdeptfkLink

Component Browser – Display, 1

- Oracle's Business Component Browser displays data from the underlying database objects (this should look familiar to 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


Oracle Business Component Browser (MyADFAppModuleLocal-Local)
File View Create Database Help
MyADFAppModule
DeptView1
EmpdeptfkLink1
EmpView2
EmpView1
EmpdeptfkLink1

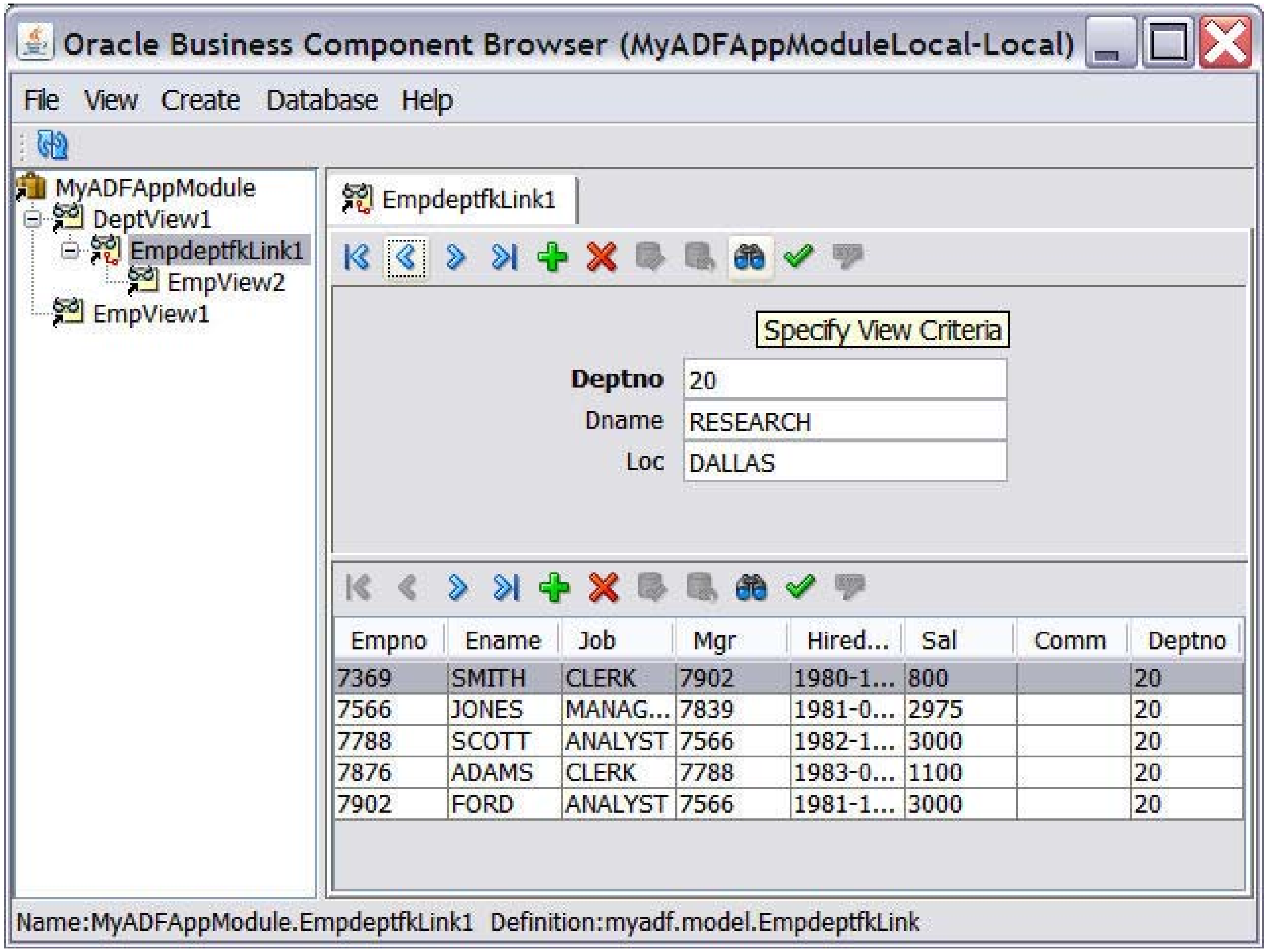
Deptno	10
Dname	ACCOUNTING
Loc	NEW YORK

Empno	Ename	Job	Mgr	Hired...	Sal	Comm	Deptno
7782	CLARK	MANAG...	7839	1981-0...	2450		10
7839	KING	PRESID...		1981-1...	5000		10
7934	MILLER	CLERK	7782	1982-0...	1300		10

Name:MyADFAppModule.EmpdeptfkLink1 Definition:myadf.model.EmpdeptfkLink

Searching Data

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



Oracle Business Component Browser (MyADFAppModuleLocal-Local)
File View Create Database Help
MyADFAppModule
DeptView1
EmpdeptfkLink1
EmpView2
EmpView1
EmpdeptfkLink1
Specify View Criteria

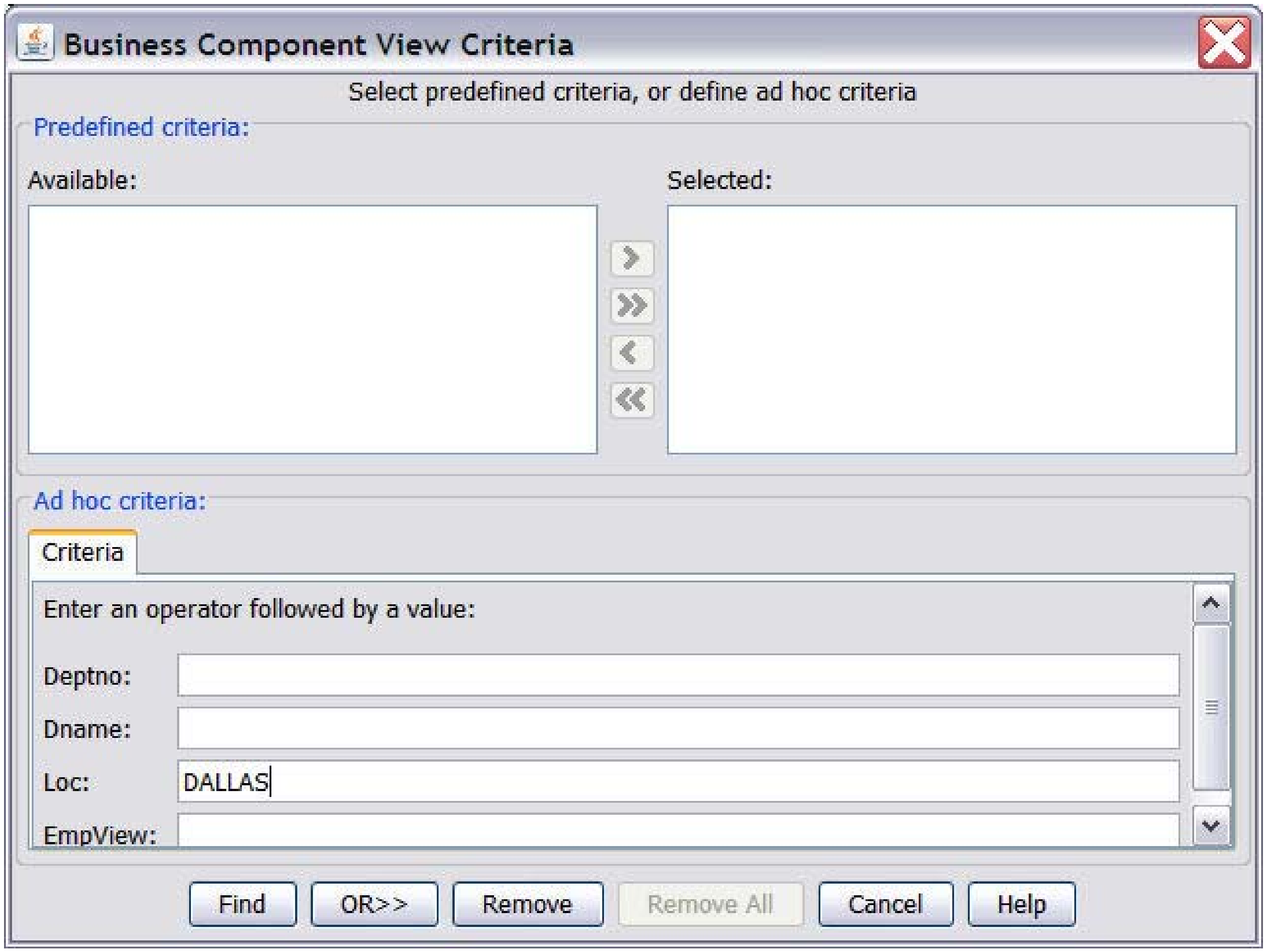
Deptno	20
Dname	RESEARCH
Loc	DALLAS

Empno	Ename	Job	Mgr	Hired...	Sal	Comm	Deptno
7369	SMITH	CLERK	7902	1980-1...	800		20
7566	JONES	MANAG...	7839	1981-0...	2975		20
7788	SCOTT	ANALYST	7566	1982-1...	3000		20
7876	ADAMS	CLERK	7788	1983-0...	1100		20
7902	FORD	ANALYST	7566	1981-1...	3000		20

Name:MyADFAppModule.EmpdeptfkLink1 Definition:myadf.model.EmpdeptfkLink

Search View Criteria

- Enter Search criteria and click "Find"



Business Component View Criteria

Select predefined criteria, or define ad hoc criteria

Predefined criteria:

Available:

Selected:

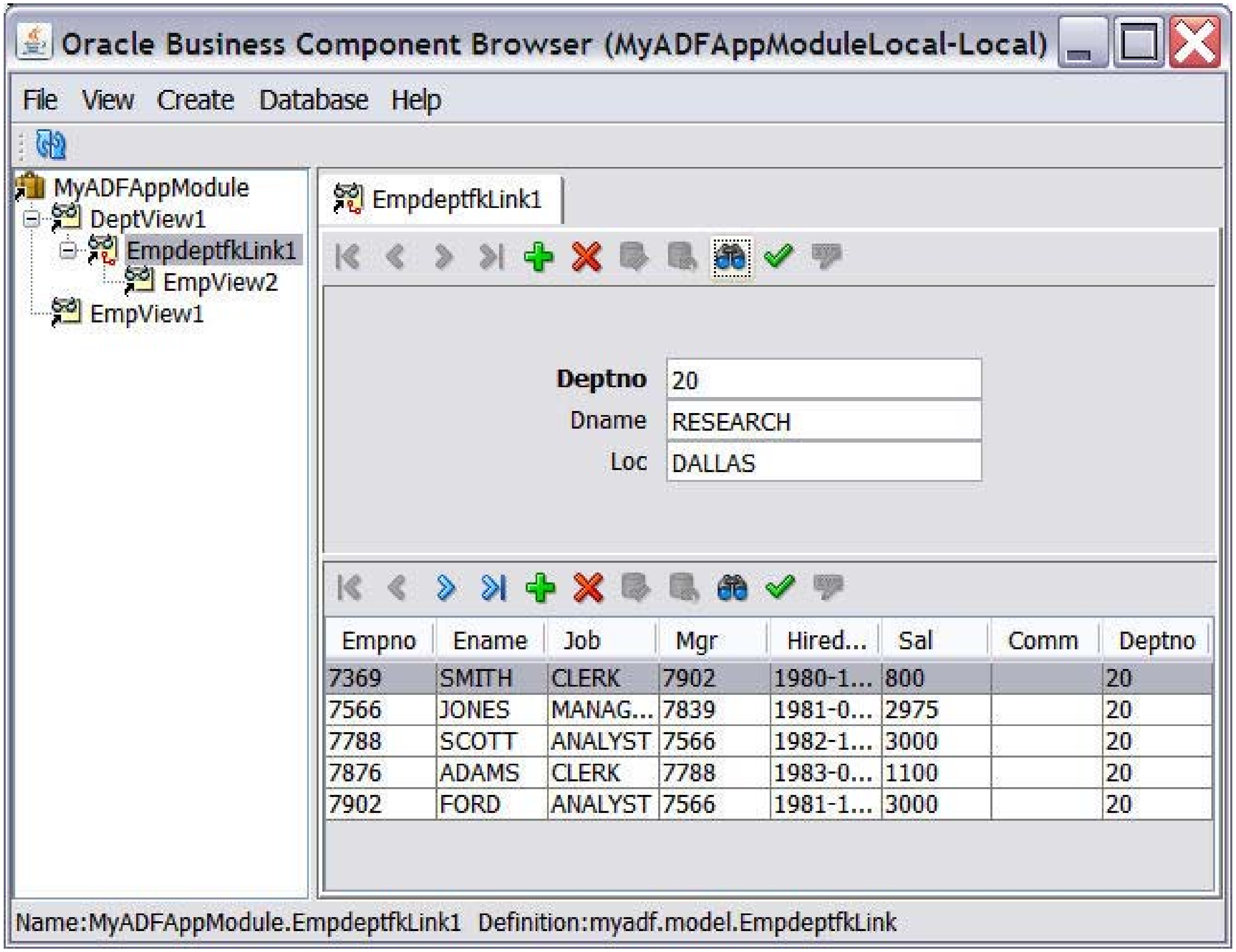
Ad hoc criteria:

Criteria

Enter an operator followed by a value:

Deptno:	
Dname:	
Loc:	DALLAS
EmpView:	

Search Results


Oracle Business Component Browser (MyADFAppModuleLocal-Local)
File View Create Database Help
MyADFAppModule
DeptView1
EmpdeptfkLink1
EmpView2
EmpView1
EmpdeptfkLink1

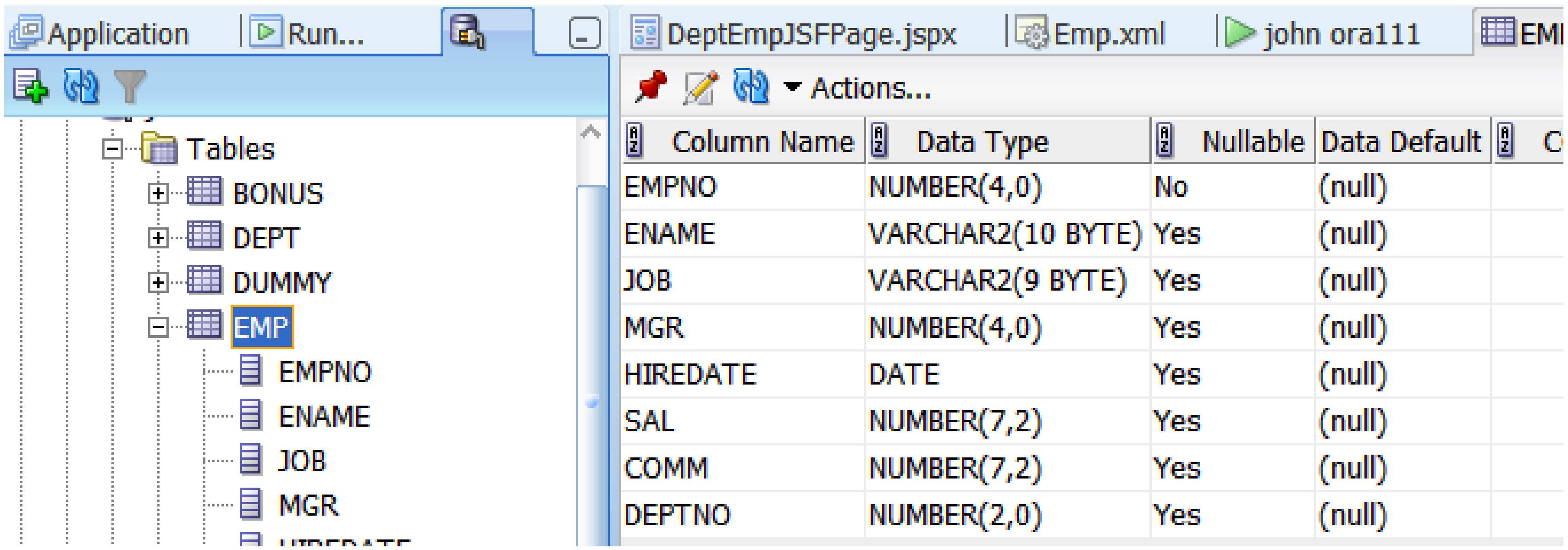
Deptno	20
Dname	RESEARCH
Loc	DALLAS

Empno	Ename	Job	Mgr	Hired...	Sal	Comm	Deptno
7369	SMITH	CLERK	7902	1980-1...	800		20
7566	JONES	MANAG...	7839	1981-0...	2975		20
7788	SCOTT	ANALYST	7566	1982-1...	3000		20
7876	ADAMS	CLERK	7788	1983-0...	1100		20
7902	FORD	ANALYST	7566	1981-1...	3000		20

Name:MyADFAppModule.EmpdeptfkLink1 Definition:myadf.model.EmpdeptfkLink

Browsing Database Objects

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



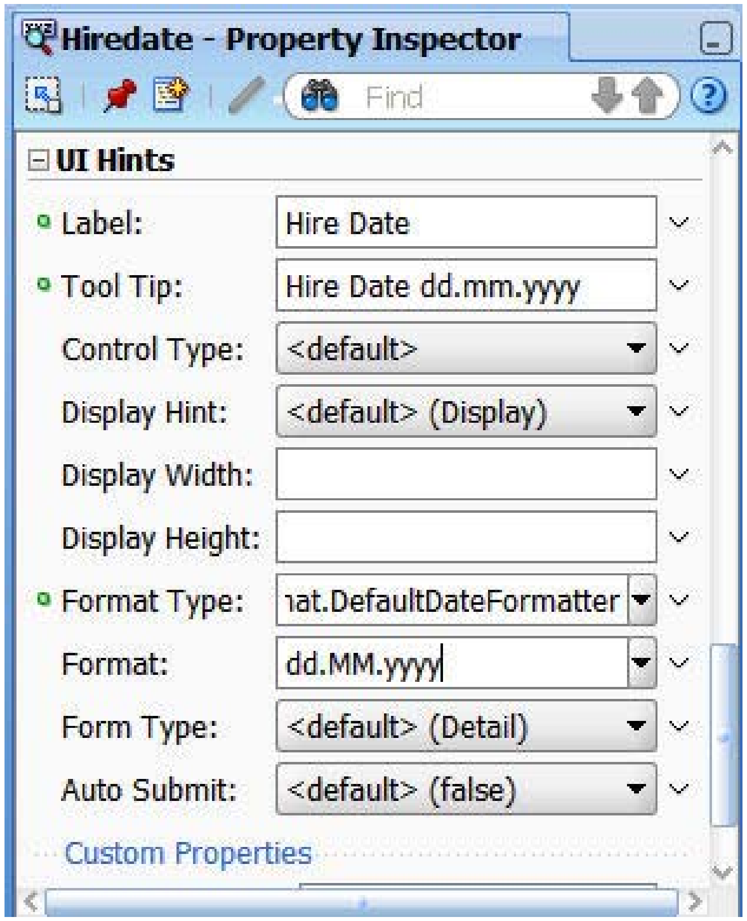
Column Name	Data Type	Nullable	Data Default	C
EMPNO	NUMBER(4,0)	No	(null)	
ENAME	VARCHAR2(10 BYTE)	Yes	(null)	
JOB	VARCHAR2(9 BYTE)	Yes	(null)	
MGR	NUMBER(4,0)	Yes	(null)	
HIREDATE	DATE	Yes	(null)	
SAL	NUMBER(7,2)	Yes	(null)	
COMM	NUMBER(7,2)	Yes	(null)	
DEPTNO	NUMBER(2,0)	Yes	(null)	

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



Hiredate - Property Inspector

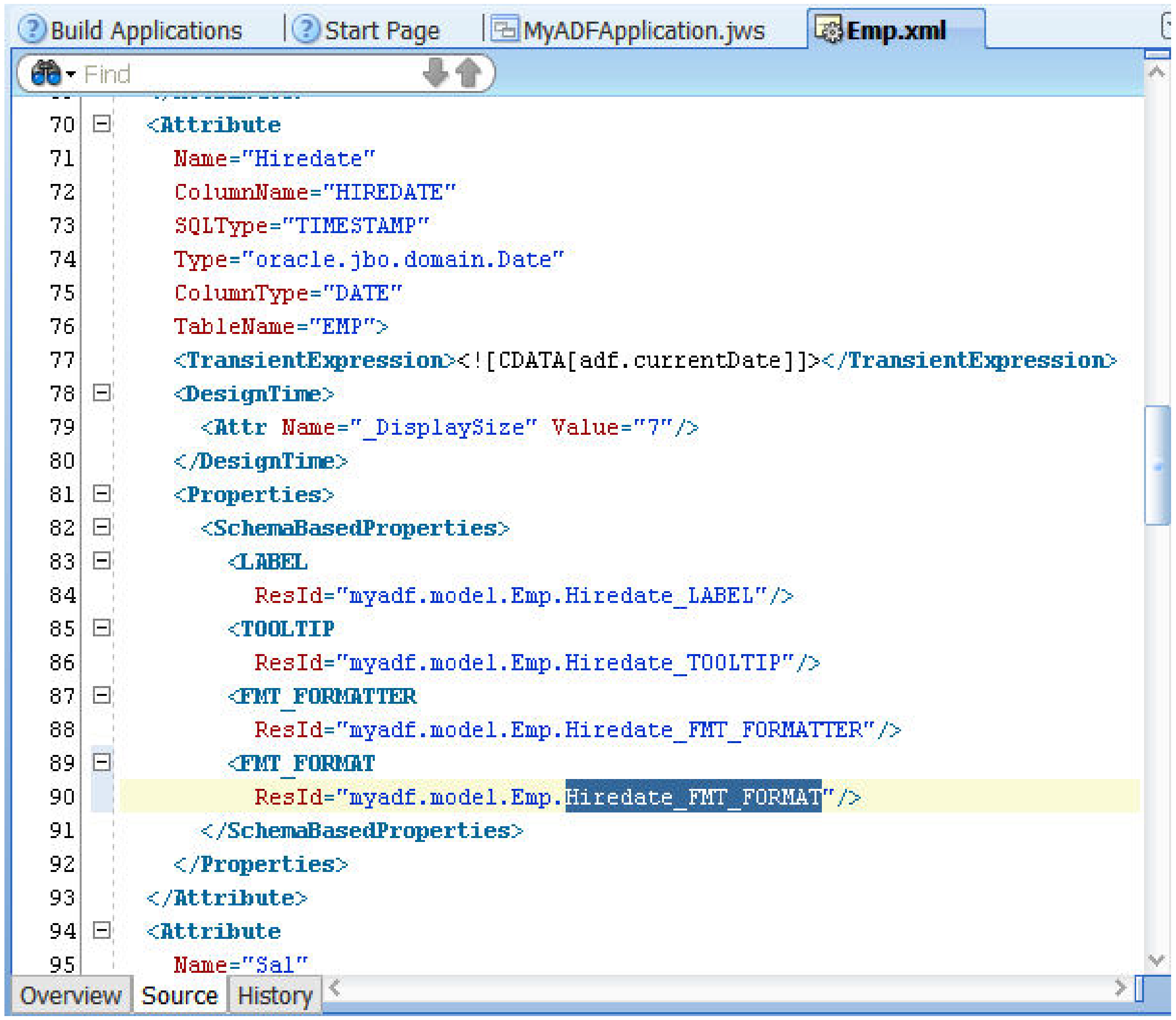
UI Hints

Label:	Hire Date
Tool Tip:	Hire Date dd.mm.yyyy
Control Type:	<default>
Display Hint:	<default> (Display)
Display Width:	
Display Height:	
Format Type:	ıat.DefaultDateFormatter
Format:	dd.MM.yyyy
Form Type:	<default> (Detail)
Auto Submit:	<default> (false)

Custom Properties

Properties in XML Files

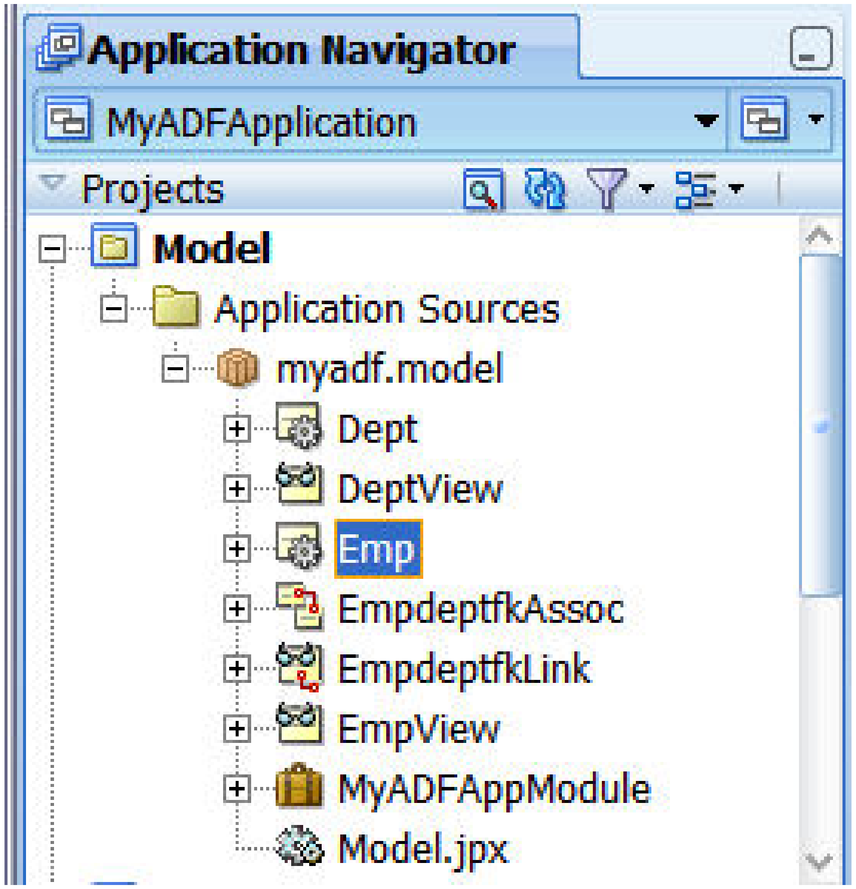
- ADF uses XML files to store declared definitions



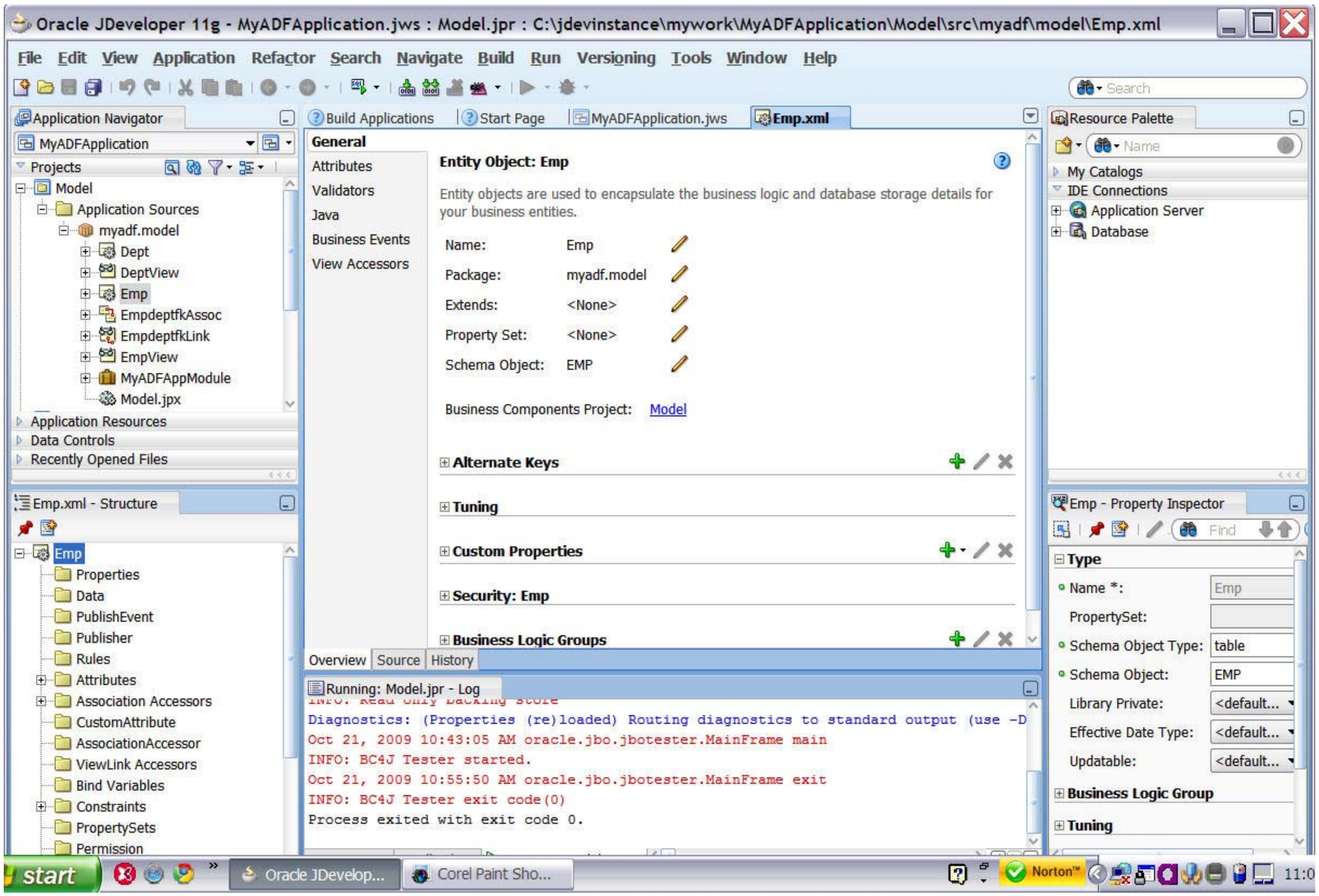
```
<Attribute
  Name="Hiredate"
  ColumnName="HIREDATE"
  SQLType="TIMESTAMP"
  Type="oracle.jbo.domain.Date"
  ColumnType="DATE"
  TableName="EMP">
  <TransientExpression><![CDATA[adf.currentDate]]></TransientExpression>
  <DesignTime>
    <Attr Name="_DisplaySize" Value="7"/>
  </DesignTime>
  <Properties>
    <SchemaBasedProperties>
      <LABEL
        ResId="myadf.model.Emp.Hiredate_LABEL"/>
      <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>
<Attribute
  Name="Sal"
```

Modify Appearance and Formatting

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



Entity Object Edit Panel



Entity Object: Emp

Entity objects are used to encapsulate the business logic and database storage details for your business entities.

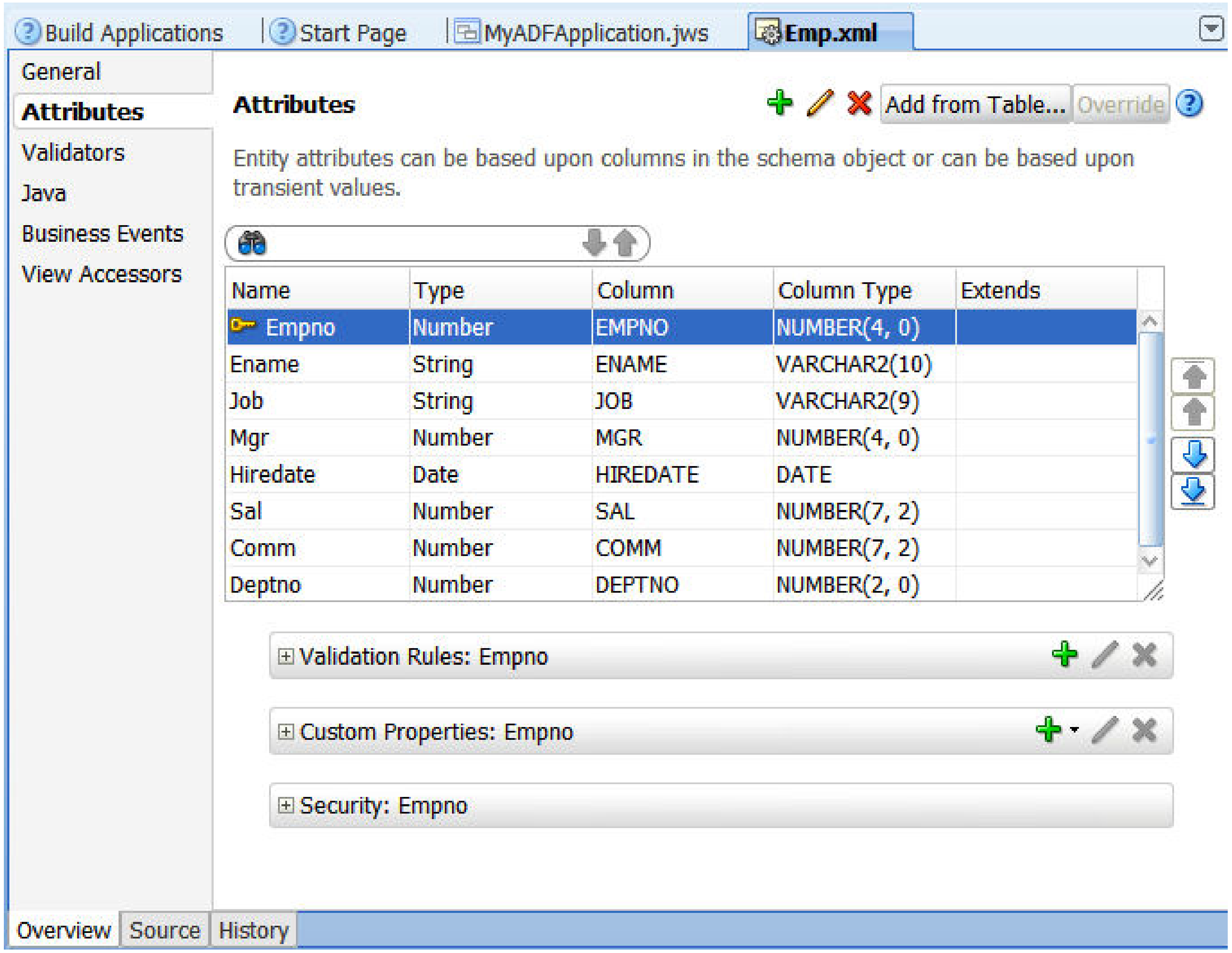
Name:	Emp
Package:	myadf.model
Extends:	<None>
Property Set:	<None>
Schema Object:	EMP

Business Components Project: Model

```
Diagnostics: (Properties (re)loaded) Routing diagnostics to standard output (use -D
Oct 21, 2009 10:43:05 AM oracle.jbo.jbotester.MainFrame main
INFO: BC4J Tester started.
Oct 21, 2009 10:55:50 AM oracle.jbo.jbotester.MainFrame exit
INFO: BC4J Tester exit code(0)
Process exited with exit code 0.
```

Name *:	Emp
PropertySet:	
Schema Object Type:	table
Schema Object:	EMP
Library Private:	<default...
Effective Date Type:	<default...
Updatable:	<default...

Entity Object Attributes



Attributes

Entity attributes can be based upon columns in the schema object or can be based upon transient values.

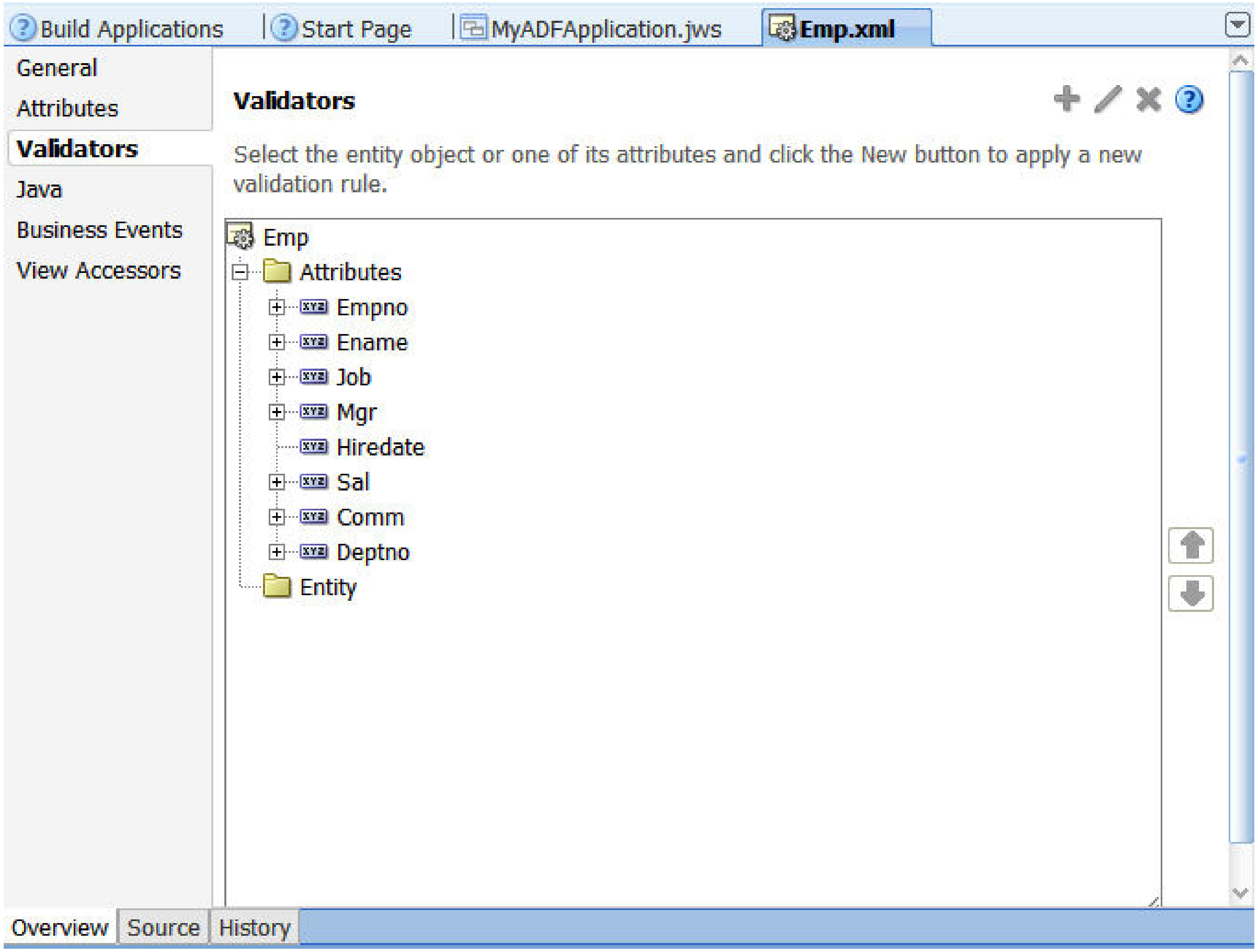
Name	Type	Column	Column Type	Extends
Empno	Number	EMPNO	NUMBER(4, 0)	
Ename	String	ENAME	VARCHAR2(10)	
Job	String	JOB	VARCHAR2(9)	
Mgr	Number	MGR	NUMBER(4, 0)	
Hiredate	Date	HIREDATE	DATE	
Sal	Number	SAL	NUMBER(7, 2)	
Comm	Number	COMM	NUMBER(7, 2)	
Deptno	Number	DEPTNO	NUMBER(2, 0)	

⊞ Validation Rules: Empno

⊞ Custom Properties: Empno

⊞ Security: Empno

Entity Object Validators



Validators

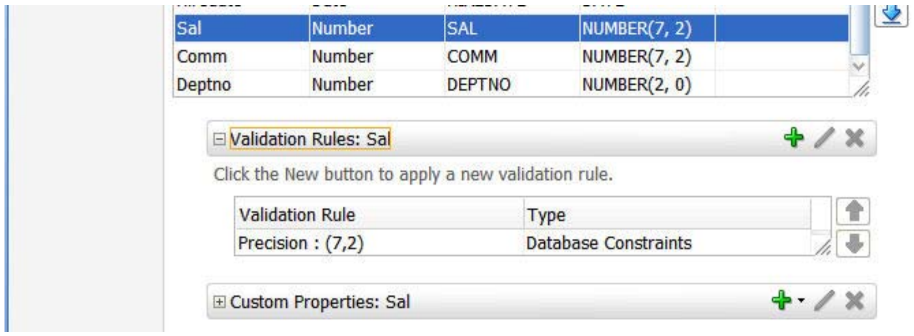
Select the entity object or one of its attributes and click the New button to apply a new validation rule.

```
Emp
  Attributes
    Empno
    Ename
    Job
    Mgr
    Hiredate
    Sal
    Comm
    Deptno
  Entity
```

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



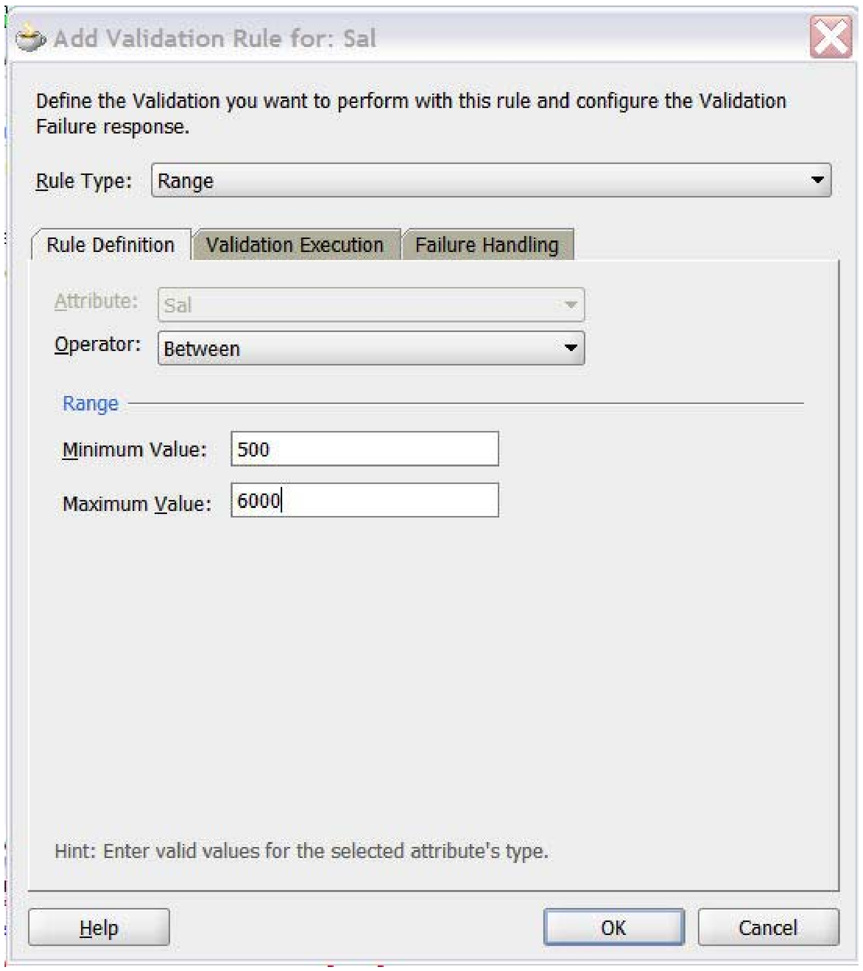
Sal	Number	SAL	NUMBER(7, 2)
Comm	Number	COMM	NUMBER(7, 2)
Deptno	Number	DEPTNO	NUMBER(2, 0)

Validation Rules: Sal

Click the New button to apply a new validation rule.

Validation Rule	Type
Precision : (7,2)	Database Constraints

Custom Properties: Sal



Add Validation Rule for: Sal

Define the Validation you want to perform with this rule and configure the Validation Failure response.

Rule Type: Range

Attribute: Sal

Operator: Between

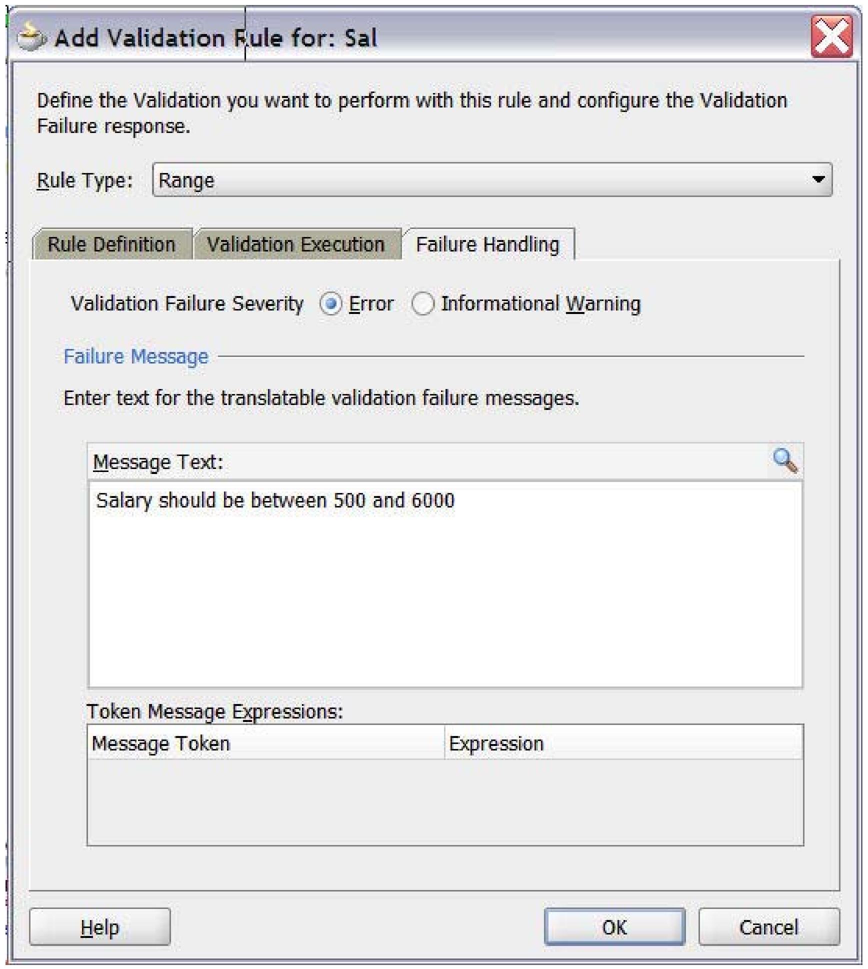
Range

Minimum Value: 500

Maximum Value: 6000

Hint: Enter valid values for the selected attribute's type.

Validation Error Messages



Add Validation Rule for: Sal

Define the Validation you want to perform with this rule and configure the Validation Failure response.

Rule Type: Range

Validation Failure Severity [✓ Error] Informational Warning

Failure Message

Enter text for the translatable validation failure messages.

Message Text:

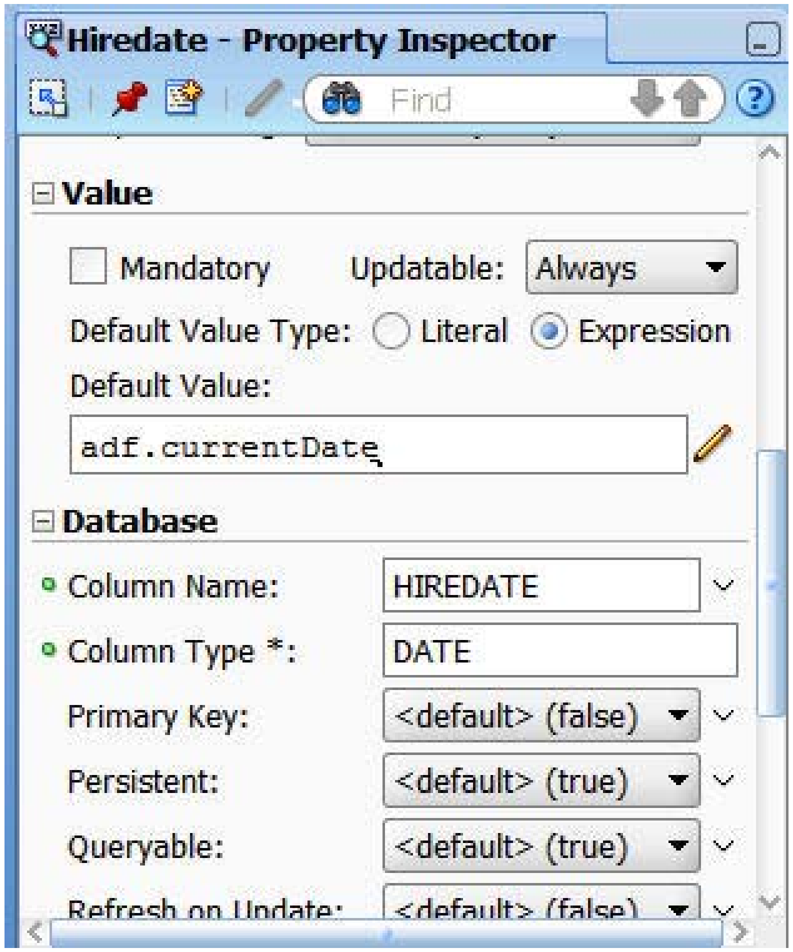
Salary should be between 500 and 6000

Token Message Expressions:

Message Token	Expression

Attribute Defaults

- Using the Property Palette, open the “Value” properties and set the default value (in this case “adf.currentDate” using ADF’s “Groovy” support)



Value

Mandatory [ ] Updatable: Always

Default Value Type: Literal [✓ Expression]

Default Value:

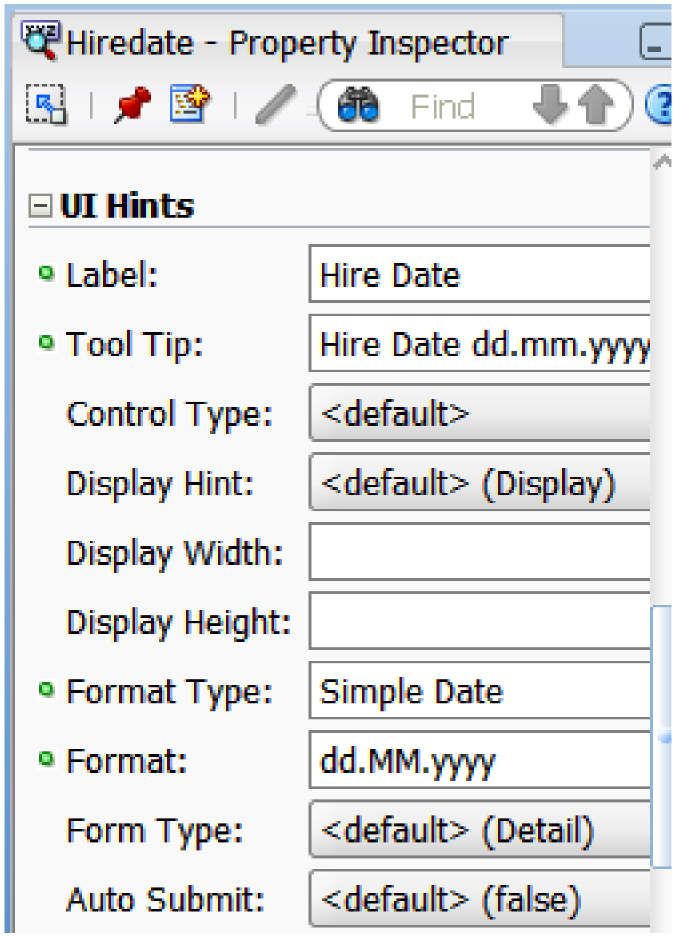
adf.currentDate

Database

Column Name:	HIREDATE
Column Type *:	DATE
Primary Key:	<default> (false)
Persistent:	<default> (true)
Queryable:	<default> (true)
Refresh on Update:	<default> (false)

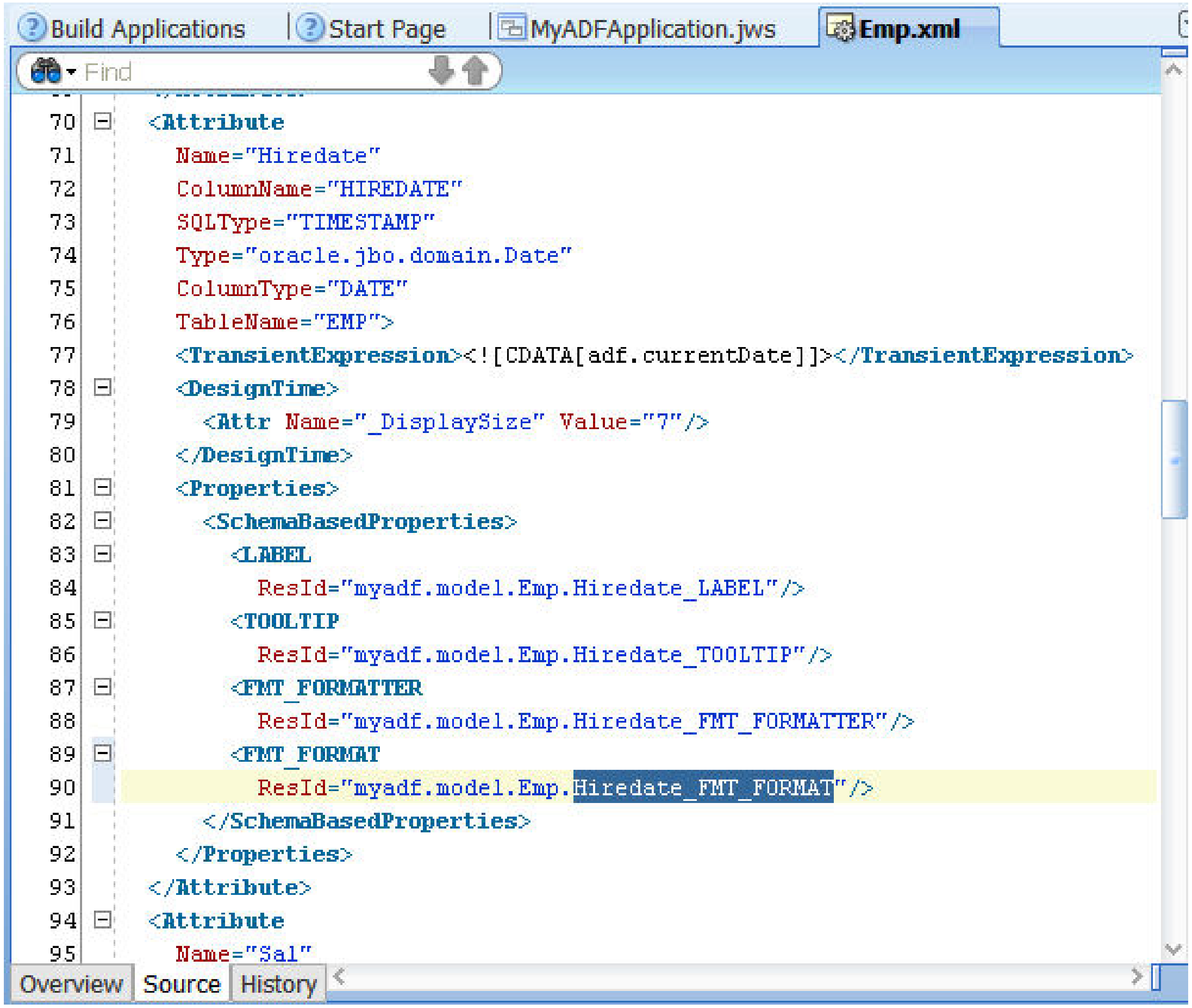
Attribute Formatting

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



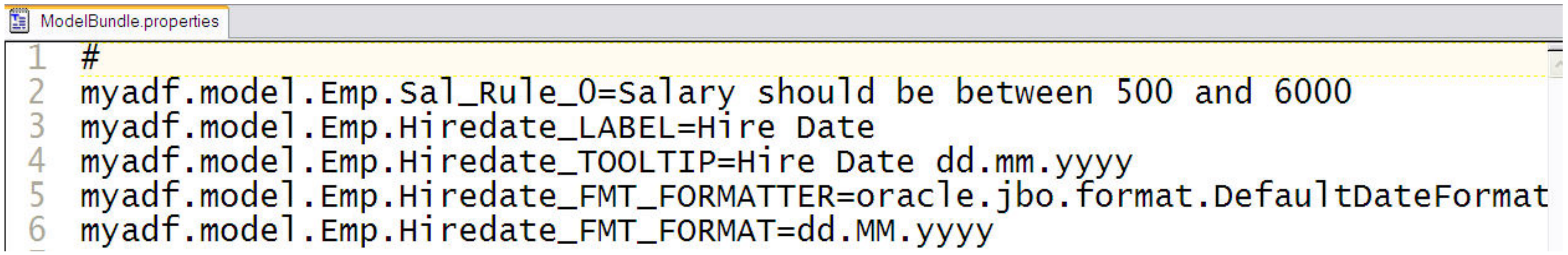
Label:	Hire Date
Tool Tip:	Hire Date dd.mm.yyyy
Control Type:	<default>
Display Hint:	<default> (Display)
Display Width:	
Display Height:	
Format Type:	Simple Date
Format:	dd.MM.yyyy
Form Type:	<default> (Detail)
Auto Submit:	<default> (false)

What Does the XML Look Like?



```
<Attribute
  Name="Hiredate"
  ColumnName="HIREDATE"
  SQLType="TIMESTAMP"
  Type="oracle.jbo.domain.Date"
  ColumnType="DATE"
  TableName="EMP">
  <TransientExpression><![CDATA[adf.currentDate]]></TransientExpression>
  <DesignTime>
    <Attr Name="_DisplaySize" Value="7"/>
  </DesignTime>
  <Properties>
    <SchemaBasedProperties>
      <LABEL
        ResId="myadf.model.Emp.Hiredate_LABEL"/>
      <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>
<Attribute
  Name="Sal"
```

Date Mask Properties File



ModelBundle.properties

```
#
myadf.model.Emp.Sal_Rule_0=Salary should be between 500 and 6000
myadf.model.Emp.Hiredate_LABEL=Hire Date
myadf.model.Emp.Hiredate_TOOLTIP=Hire Date dd.mm.yyyy
myadf.model.Emp.Hiredate_FMT_FORMATTER=oracle.jbo.format.DefaultDateFormat
myadf.model.Emp.Hiredate_FMT_FORMAT=dd.MM.yyyy
```

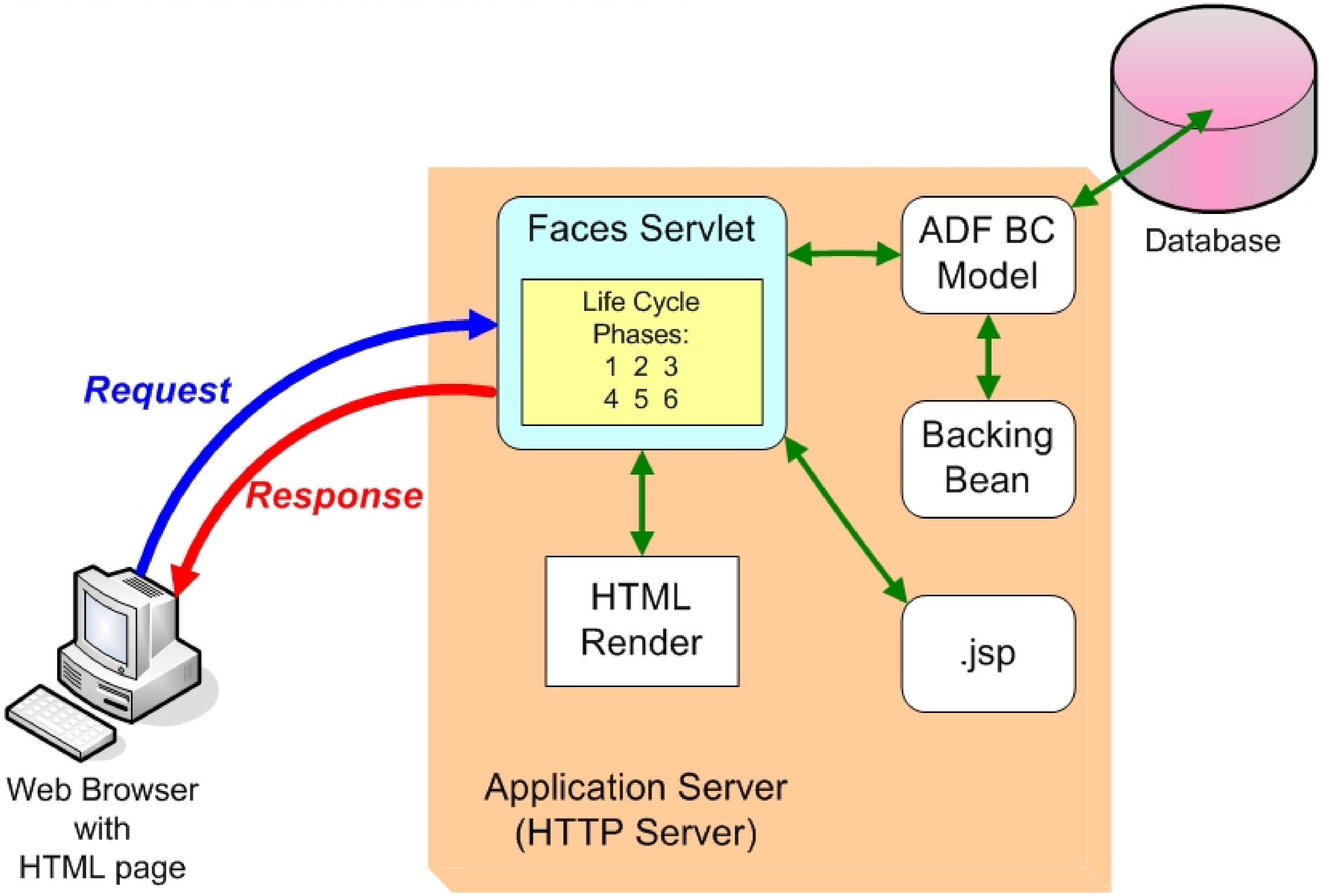
Comparison to Oracle Forms

- 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)

Creating Web Applications

- Oracle’s Business Component Browser is impressive, but it's hardly a customer-facing user interface
- ADF Faces extends the Java Server Faces (JSF) framework using XML tags to describe the user interface
- ADF Faces provides a Rich-Client Interface that uses JavaScript and AJAX components; therefore 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


Database
Faces Servlet
Life Cycle Phases: 1 2 3 4 5 6
ADF BC Model
Backing Bean
HTML Render
.jsp
Request
Response
Web Browser with HTML page
Application Server (HTTP Server)

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) perform a predictable cycle as follows:

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

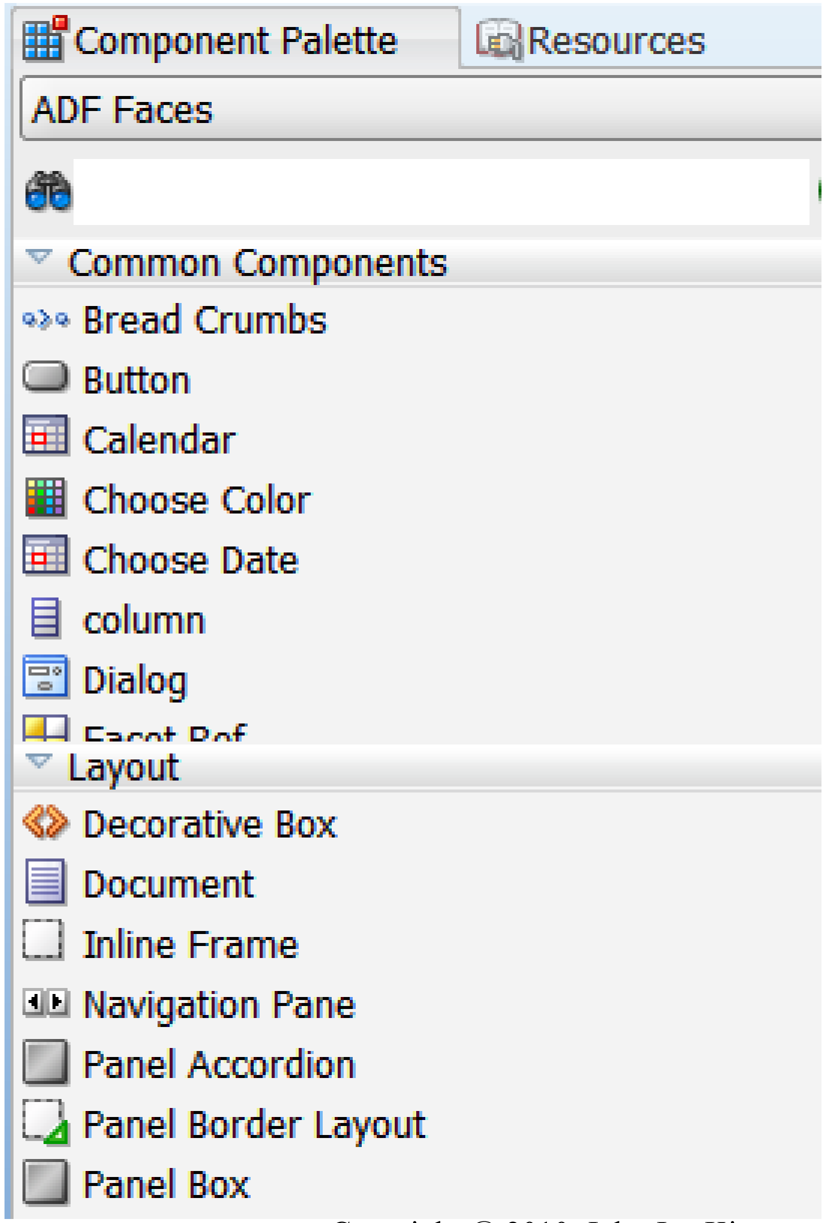
- This Life Cycle is normally transparent; however, it is useful to understand it when debugging

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)

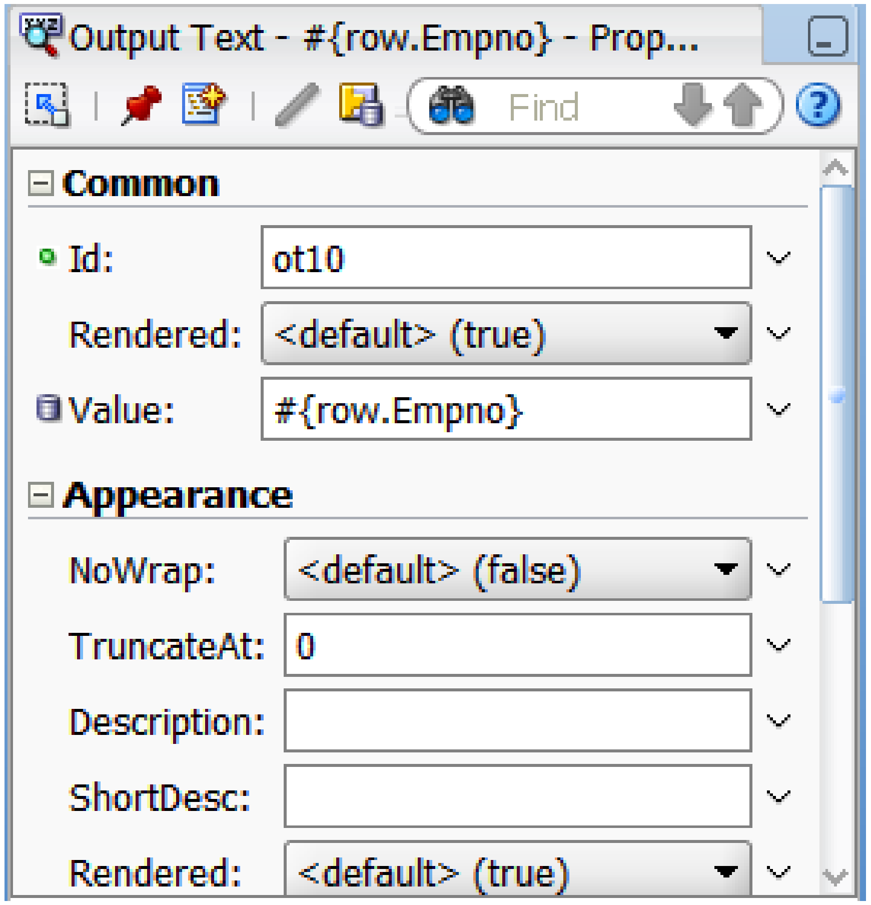
ADF Faces Component Palette

- 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



Output Text - #{row.Empno} - Prop...

Common

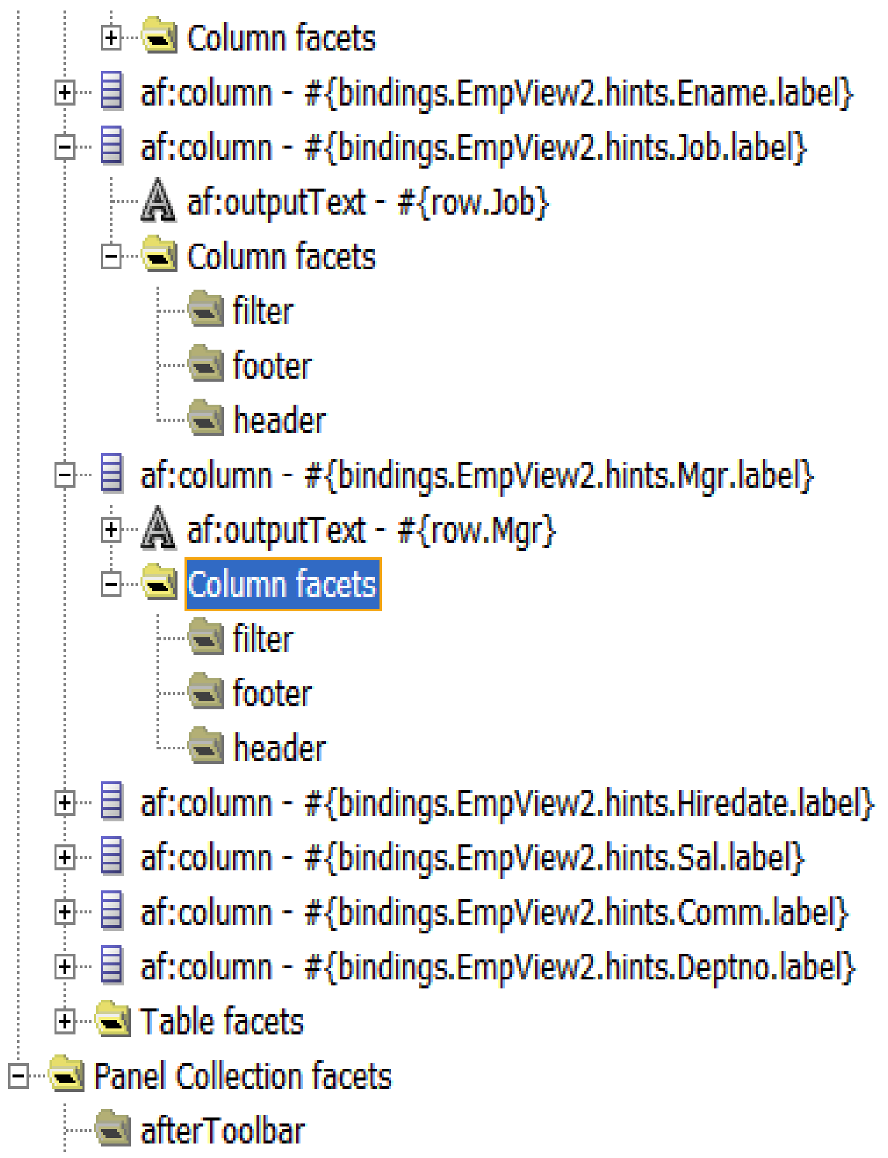
Id:	ot10
Rendered:	<default> (true)
Value:	#{row.Empno}

Appearance

NoWrap:	<default> (false)
TruncateAt:	0
Description:	
ShortDesc:	
Rendered:	<default> (true)

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



```
    ⊞ Column facets
⊞ af:column - #{bindings.EmpView2.hints.Ename.label}
⊟ af:column - #{bindings.EmpView2.hints.Job.label}
    af:outputText - #{row.Job}
  ⊟ Column facets
      filter
      footer
      header
⊟ af:column - #{bindings.EmpView2.hints.Mgr.label}
  ⊞ af:outputText - #{row.Mgr}
  ⊟ Column facets
      filter
      footer
      header
⊞ af:column - #{bindings.EmpView2.hints.Hiredate.label}
⊞ af:column - #{bindings.EmpView2.hints.Sal.label}
⊞ af:column - #{bindings.EmpView2.hints.Comm.label}
⊞ af:column - #{bindings.EmpView2.hints.Deptno.label}
⊞ Table facets
⊟ Panel Collection facets
    afterToolbar
```

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

Panel Collections, Accordions, Tabbed Panels

- 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 will allow not only 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

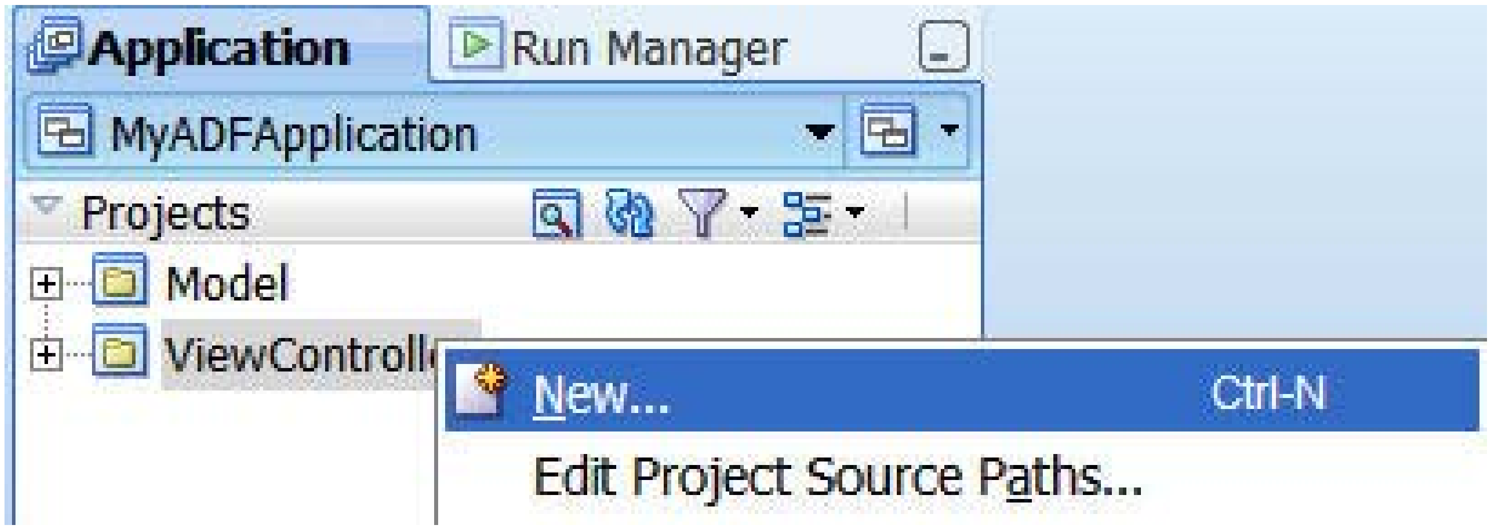
Creating ADF Faces / JSF Pages

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

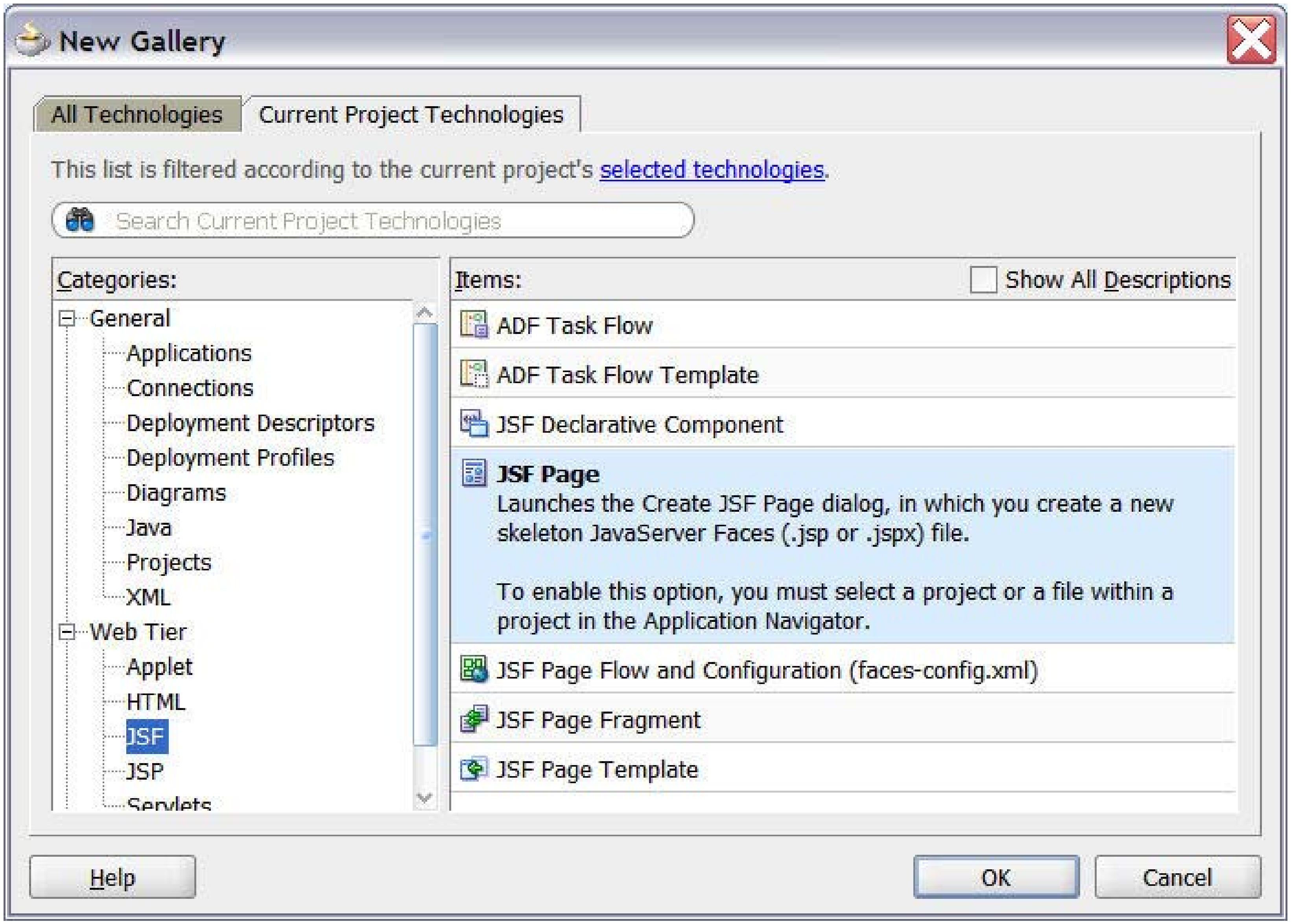
1. Design Web Page
2. Create new JSF Page using JDeveloper
3. Add Visual Components to JSF Page
4. Bind Visual Components to ADF BC Objects

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



New Gallery

All Technologies | Current Project Technologies

This list is filtered according to the current project's selected technologies.

Search Current Project Technologies

Categories:

- General
  - Applications
  - Connections
  - Deployment Descriptors
  - Deployment Profiles
  - Diagrams
  - Java
  - Projects
  - XML
- Web Tier
  - Applet
  - HTML
  - JSF
  - JSP
  - Servlets

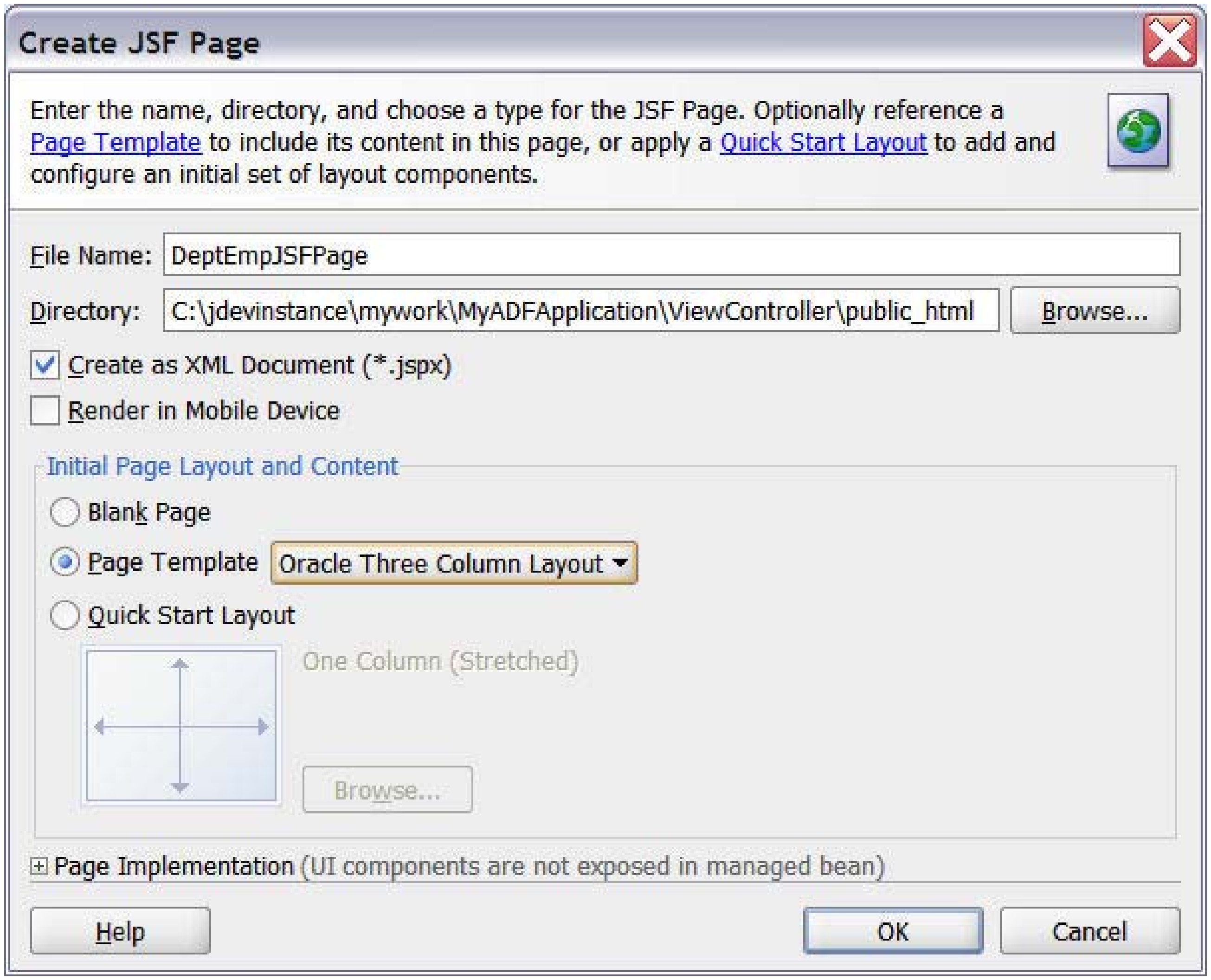
Items: ☐ Show All Descriptions

- ADF Task Flow
- ADF Task Flow Template
- JSF Declarative Component
- **JSF Page**
  Launches the Create JSF Page dialog, in which you create a new skeleton JavaServer Faces (.jsp or .jspx) file.

  To enable this option, you must select a project or a file within a project in the Application Navigator.
- JSF Page Flow and Configuration (faces-config.xml)
- JSF Page Fragment
- JSF Page Template

Training Resources

Naming Web Page


Create JSF Page

Enter the name, directory, and choose a type for the JSF Page. Optionally reference a Page Template to include its content in this page, or apply a Quick Start Layout to add and configure an initial set of layout components.

File Name: DeptEmpJSFPage

Directory: C:\jdevinstance\mywork\MyADFApplication\ViewController\public_html

[✓] Create as XML Document (*.jspx)

[ ] Render in Mobile Device

Initial Page Layout and Content

( ) Blank Page

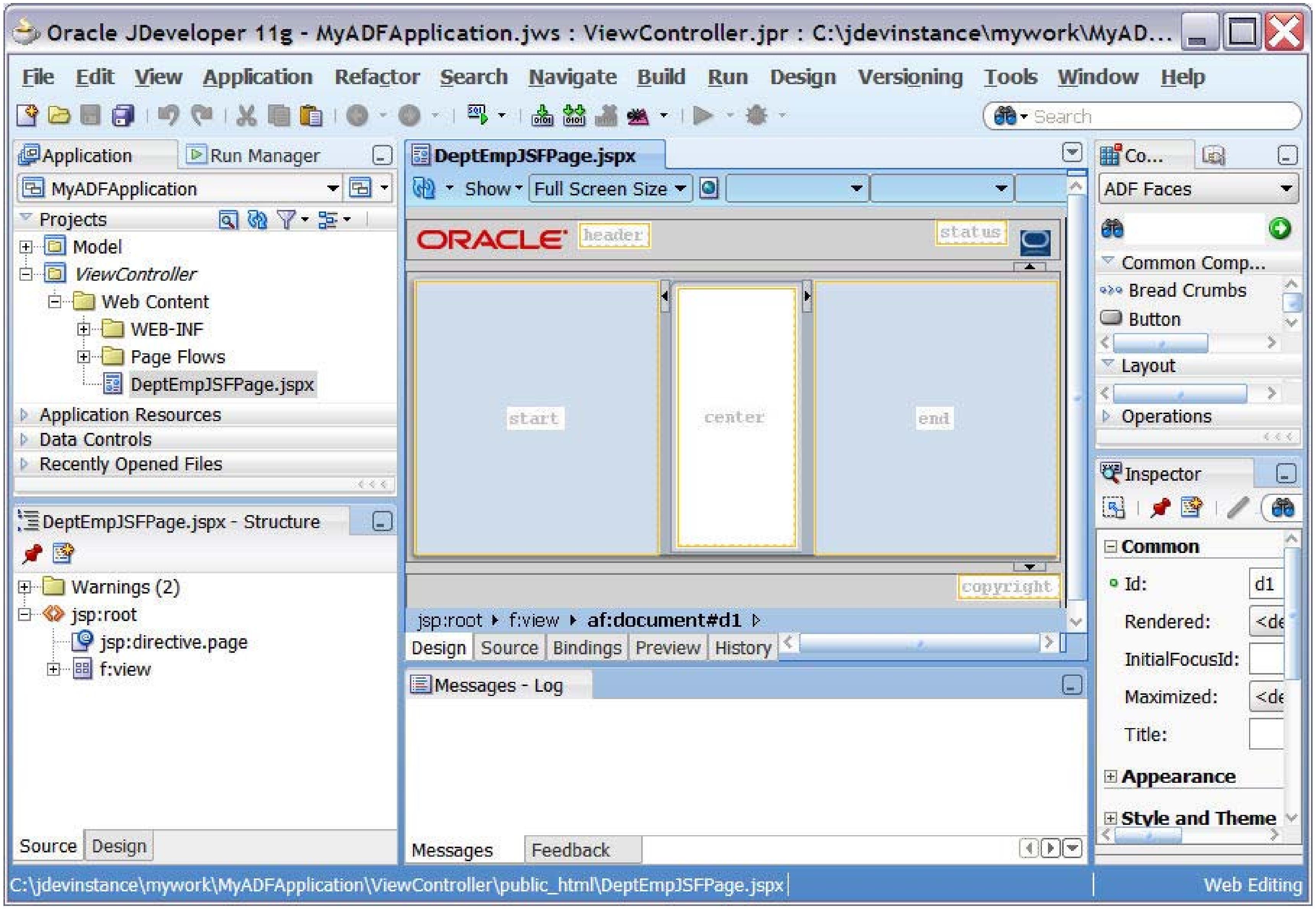
(●) Page Template: Oracle Three Column Layout

( ) Quick Start Layout: One Column (Stretched)

⊞ Page Implementation (UI components are not exposed in managed bean)

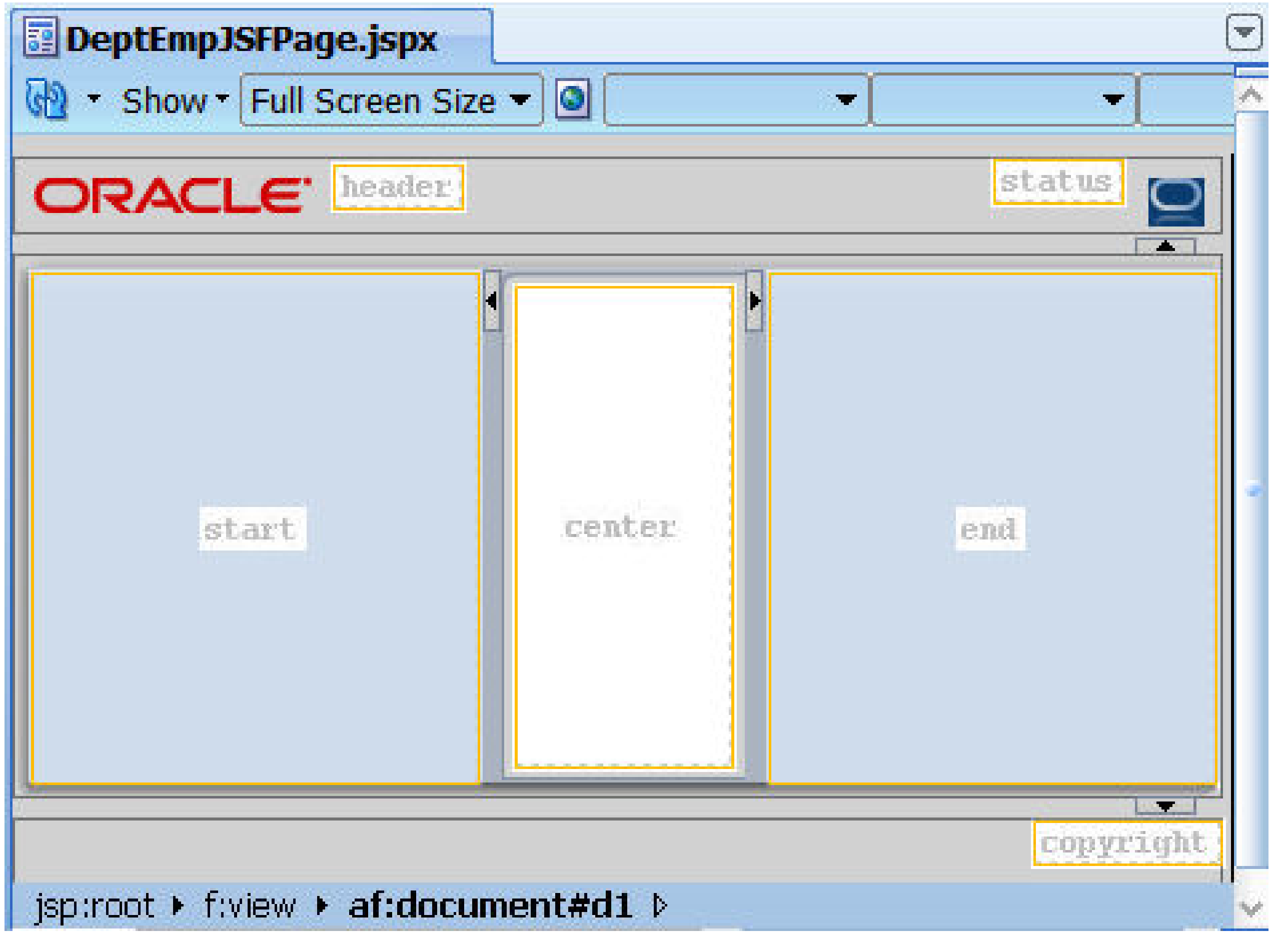
– Note the “Create as XML Document (*.jspx)” box

Visual Display with Initial Screen



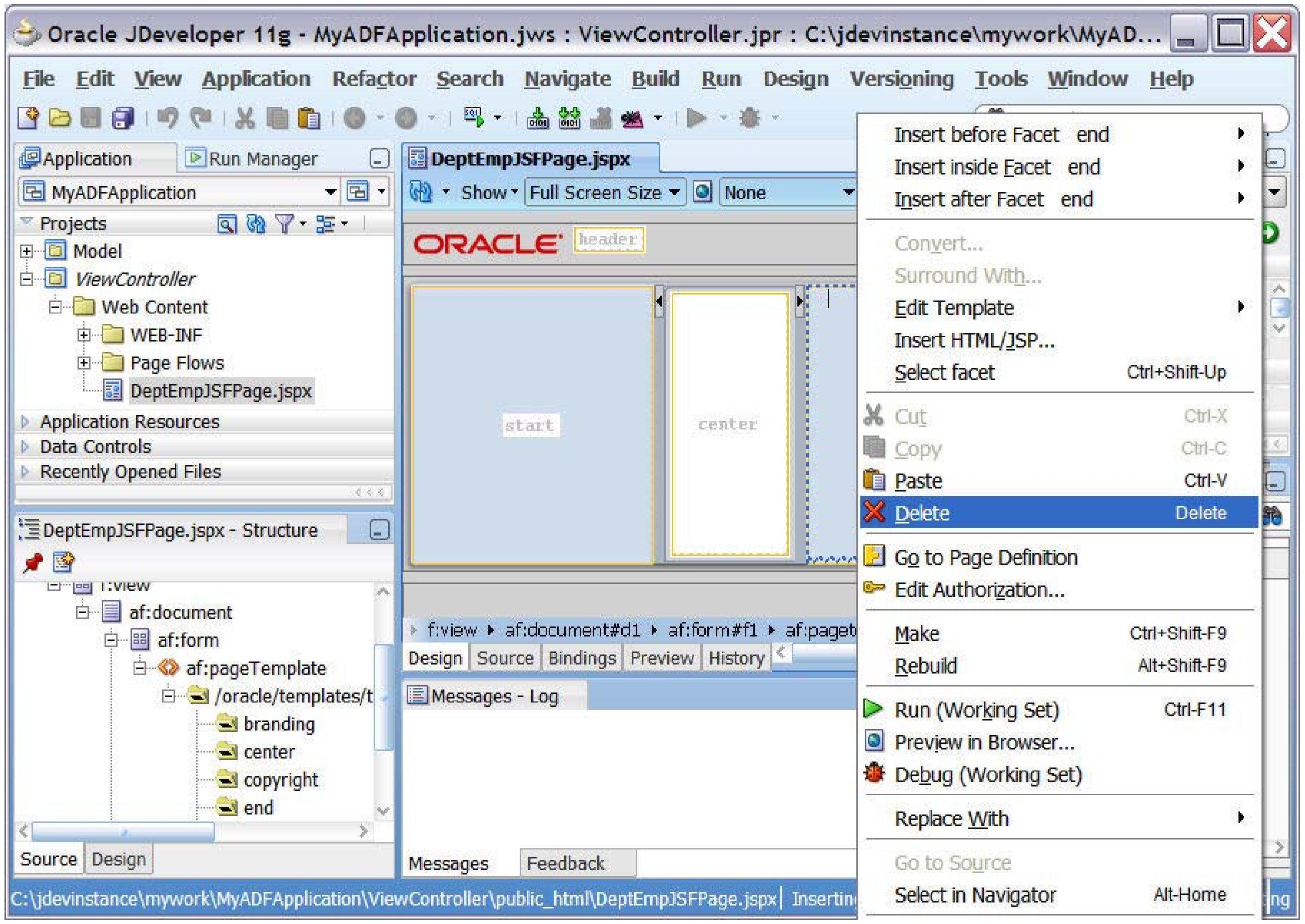
Three-Column Layout

- The supplied three-column layout is ready to have objects dropped into it

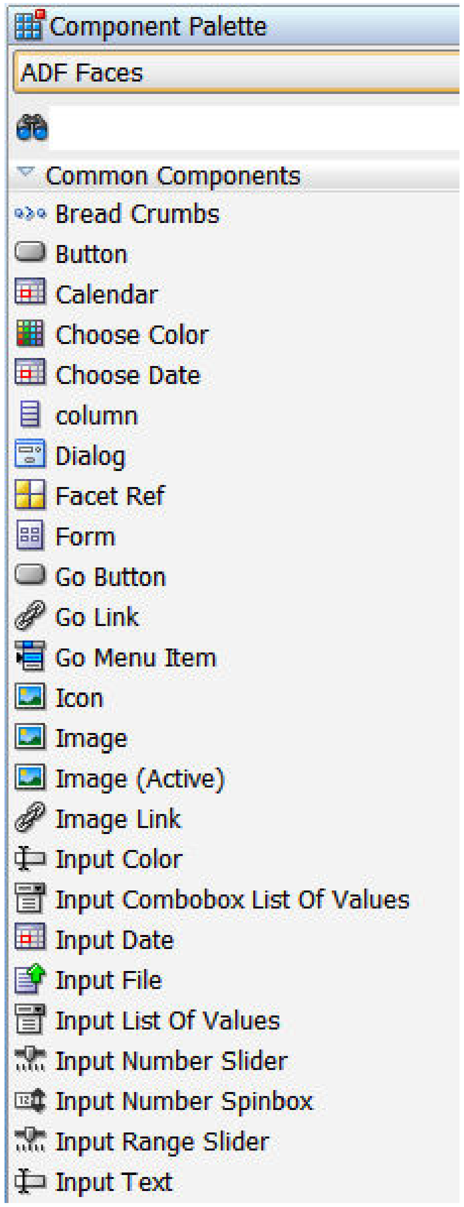


Deleting Panel from Layout

- Sometimes you want a two-column layout rather than three; just select the panel you don't want and delete



Common Components

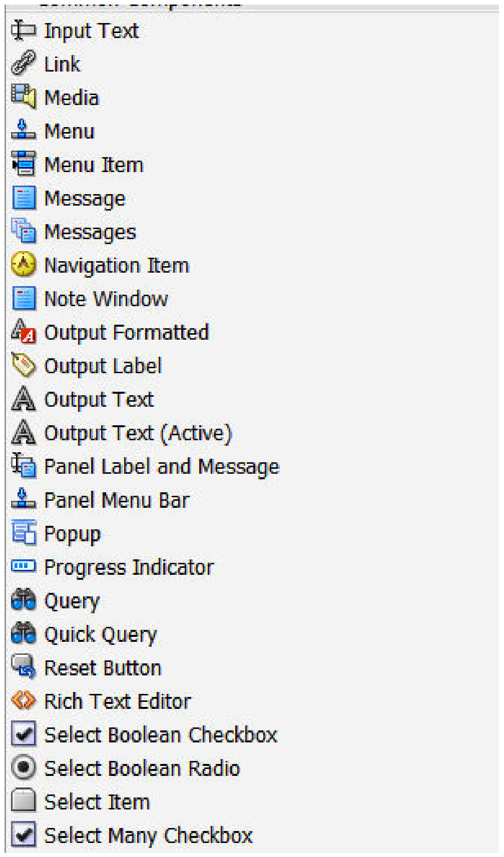


Component Palette

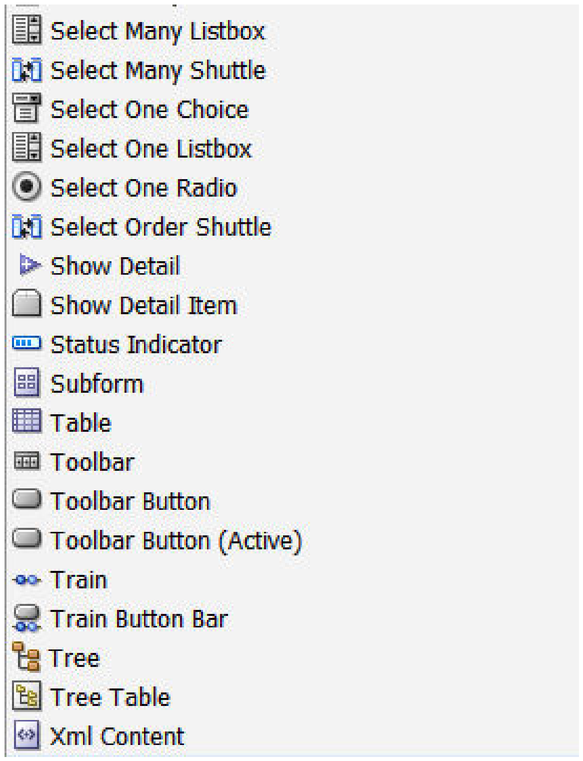
ADF Faces

Common Components

- Bread Crumbs
- Button
- Calendar
- Choose Color
- Choose Date
- column
- Dialog
- Facet Ref
- Form
- Go Button
- Go Link
- Go Menu Item
- Icon
- Image
- Image (Active)
- Image Link
- Input Color
- Input Combobox List Of Values
- Input Date
- Input File
- Input List Of Values
- Input Number Slider
- Input Number Spinbox
- Input Range Slider
- Input Text

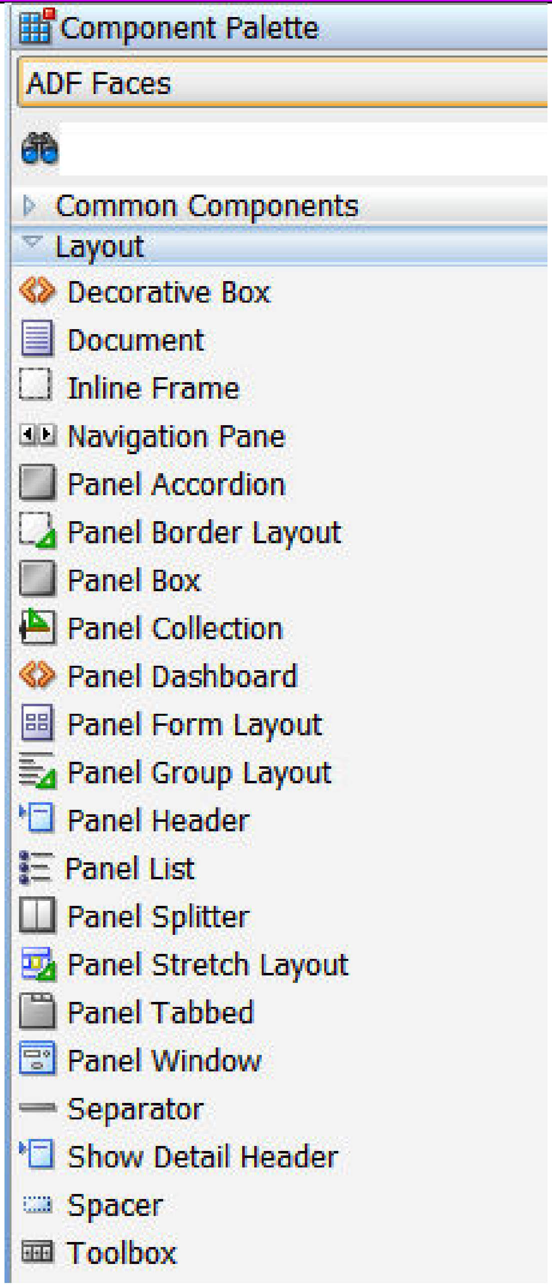


- Input Text
- Link
- Media
- Menu
- Menu Item
- Message
- Messages
- Navigation Item
- Note Window
- Output Formatted
- Output Label
- Output Text
- Output Text (Active)
- Panel Label and Message
- Panel Menu Bar
- Popup
- Progress Indicator
- Query
- Quick Query
- Reset Button
- Rich Text Editor
- Select Boolean Checkbox
- Select Boolean Radio
- Select Item
- Select Many Checkbox



- Select Many Listbox
- Select Many Shuttle
- Select One Choice
- Select One Listbox
- Select One Radio
- Select Order Shuttle
- Show Detail
- Show Detail Item
- Status Indicator
- Subform
- Table
- Toolbar
- Toolbar Button
- Toolbar Button (Active)
- Train
- Train Button Bar
- Tree
- Tree Table
- Xml Content

Layout Components


Component Palette

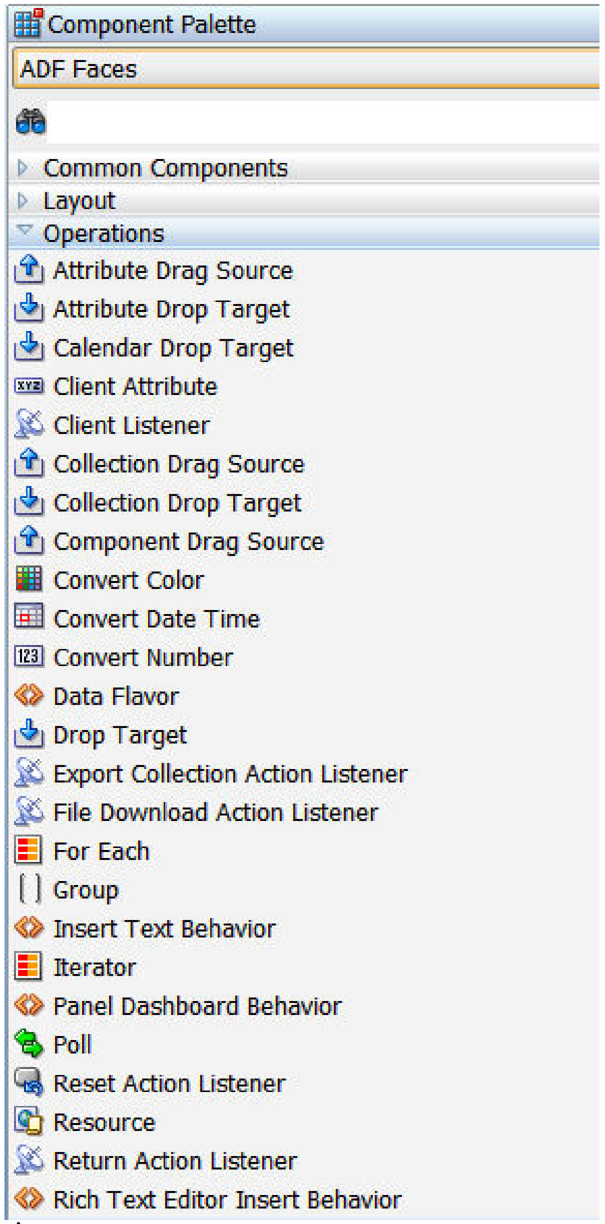
ADF Faces

Common Components

Layout

- Decorative Box
- Document
- Inline Frame
- Navigation Pane
- Panel Accordion
- Panel Border Layout
- Panel Box
- Panel Collection
- Panel Dashboard
- Panel Form Layout
- Panel Group Layout
- Panel Header
- Panel List
- Panel Splitter
- Panel Stretch Layout
- Panel Tabbed
- Panel Window
- Separator
- Show Detail Header
- Spacer
- Toolbox

Operations Components



Component Palette

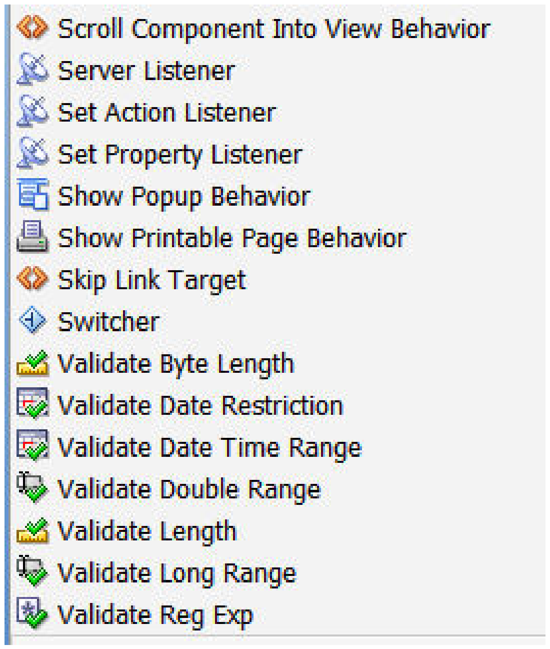
ADF Faces

Common Components

Layout

Operations

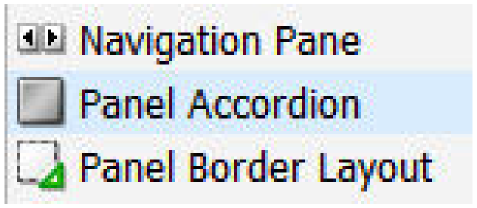
- Attribute Drag Source
- Attribute Drop Target
- Calendar Drop Target
- Client Attribute
- Client Listener
- Collection Drag Source
- Collection Drop Target
- Component Drag Source
- Convert Color
- Convert Date Time
- Convert Number
- Data Flavor
- Drop Target
- Export Collection Action Listener
- File Download Action Listener
- For Each
- Group
- Insert Text Behavior
- Iterator
- Panel Dashboard Behavior
- Poll
- Reset Action Listener
- Resource
- Return Action Listener
- Rich Text Editor Insert Behavior

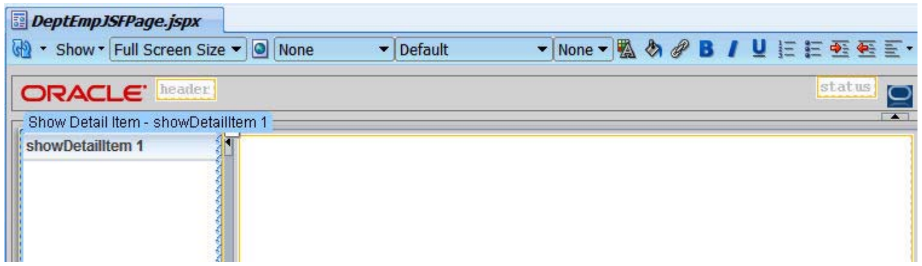


- Scroll Component Into View Behavior
- Server Listener
- Set Action Listener
- Set Property Listener
- Show Popup Behavior
- Show Printable Page Behavior
- Skip Link Target
- Switcher
- Validate Byte Length
- Validate Date Restriction
- Validate Date Time Range
- Validate Double Range
- Validate Length
- Validate Long Range
- Validate Reg Exp

Adding Accordion Component

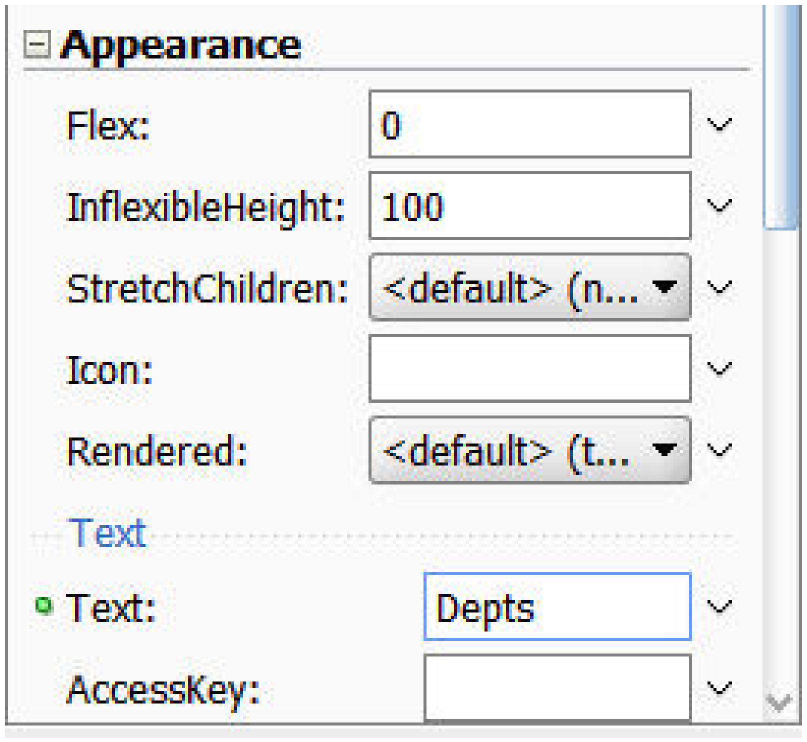
- To add an Accordion Component to the web page; Panel Accordion component from the pallet to the desired column ("start")





Change Accordion Title Property

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



Appearance

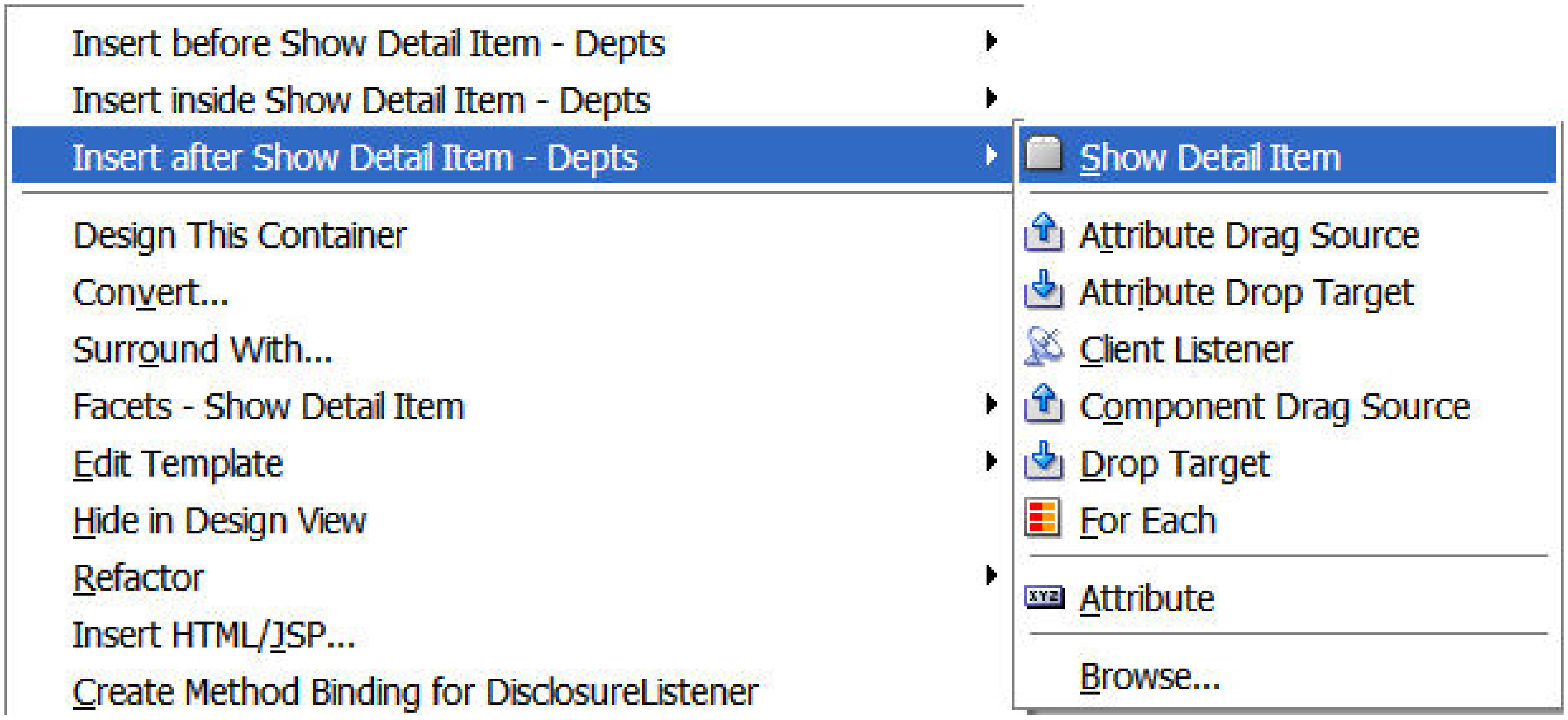
Flex:	0
InflexibleHeight:	100
StretchChildren:	<default> (n...
Icon:	
Rendered:	<default> (t...

Text

Text:	Depts
AccessKey:	

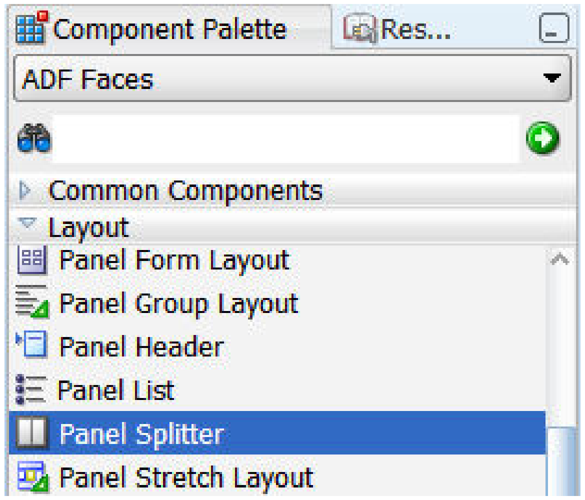
Add Data Component

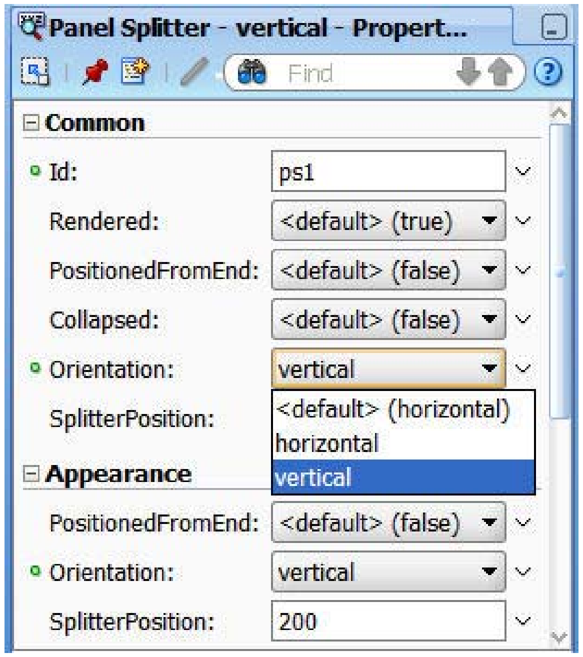
- Right-click in the “Depts” Accordion; when prompted choose “Insert After Show Details Item - Depts -> Show Detail Item” to add another Accordion to the page (not used further in this demo…)



Using Panel Splitter

- To divide the right-part of the page (Center) into two parts; find the “Panel Splitter” component then drag it to the column marked “Center”
- Next, change it's “Orientation” property to “Vertical”




Panel Splitter - vertical - Propert...

Common

Id:	ps1
Rendered:	<default> (true)
PositionedFromEnd:	<default> (false)
Collapsed:	<default> (false)
Orientation:	vertical
SplitterPosition:	

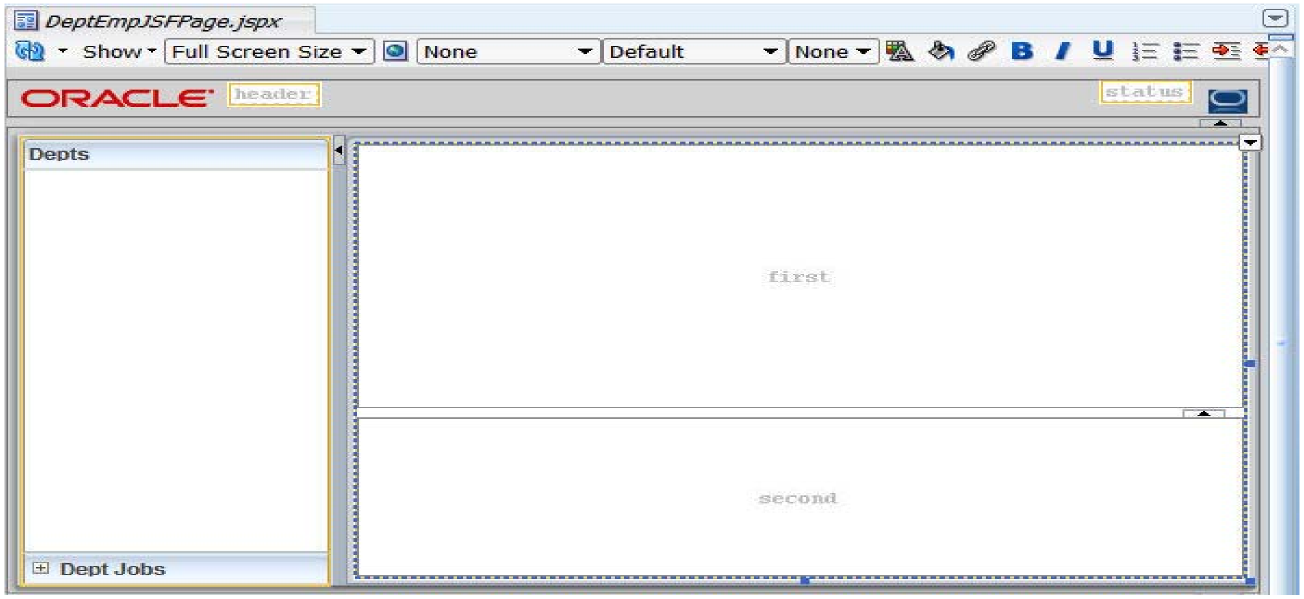
<default> (horizontal)
horizontal
vertical

Appearance

PositionedFromEnd:	<default> (false)
Orientation:	vertical
SplitterPosition:	200

Page After Splitting

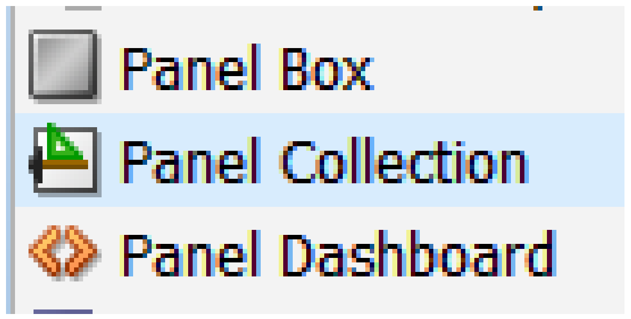
- After splitting; the page looks something like this:

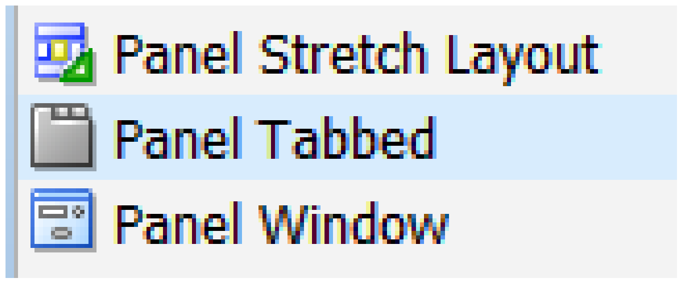


  - The "Depts" part will hold specifics for one Department
  - The "first" part will hold a list of Department employees
  - The "second" part of the page will hold details for an individual Employee

Adding Collection and Tabbed Area

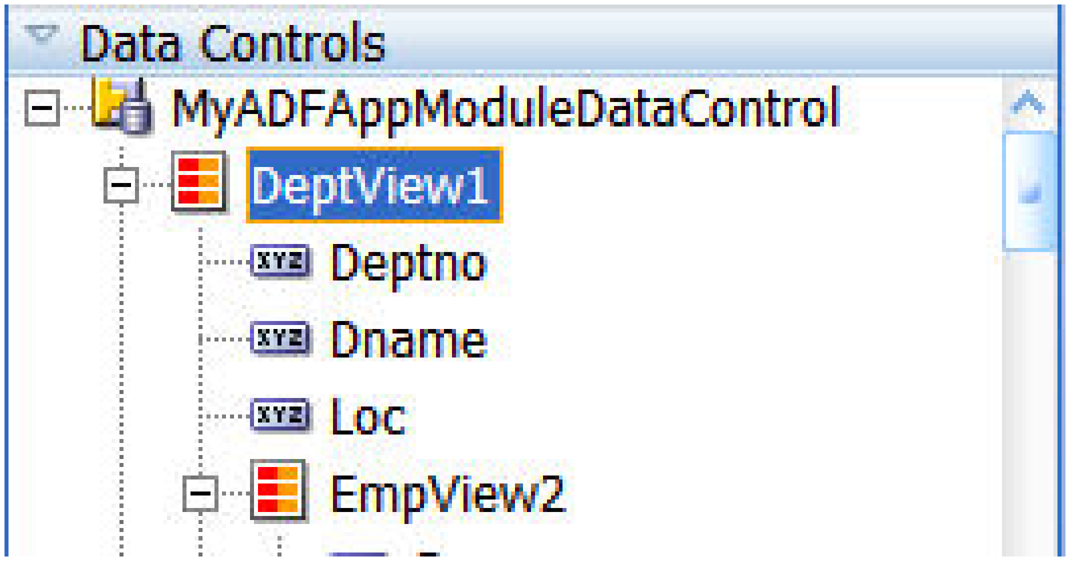
- Find the “Panel Collection” component in the Layout components and drag it to the “first” (top) part of the Splitter area
- Find the “Panel Tabbed” component in the Layout components and drag it to the “second” (bottom) part of the Splitter area





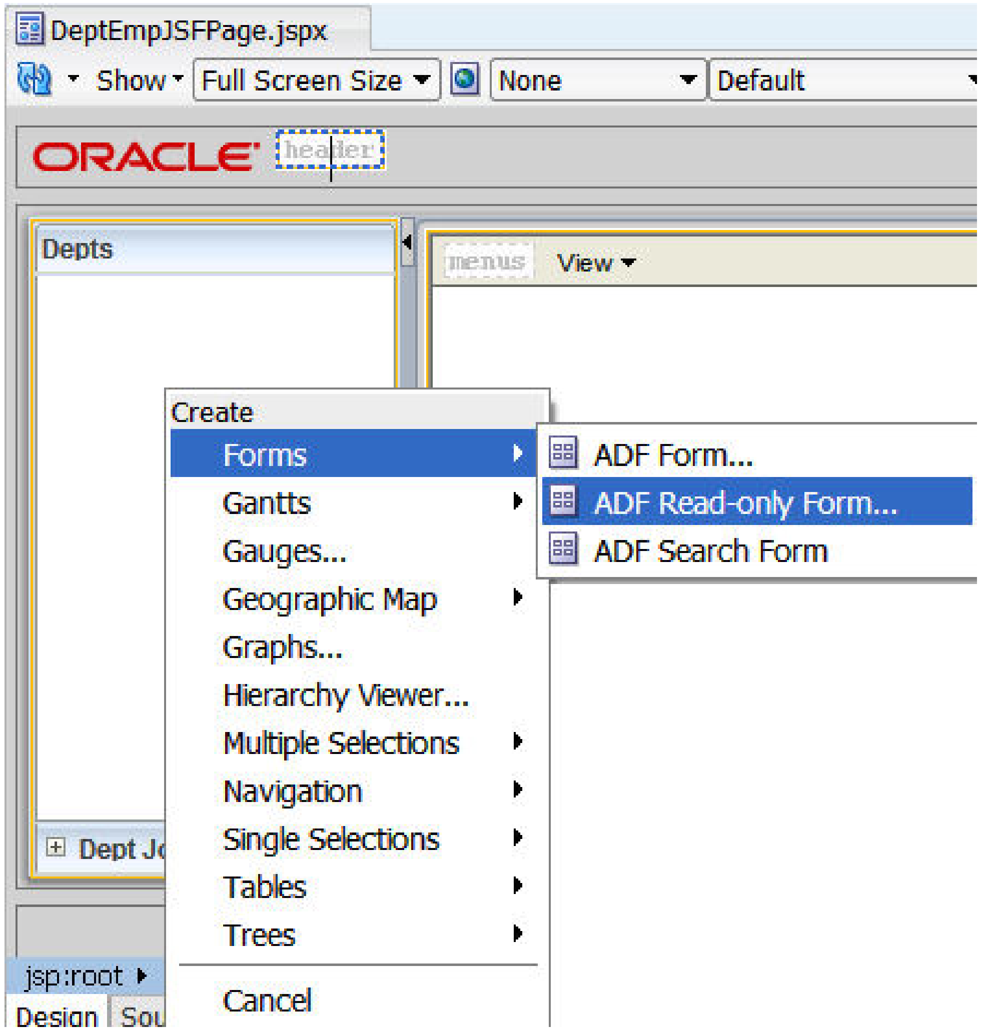
Data Binding: Adding Data, 1

- 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



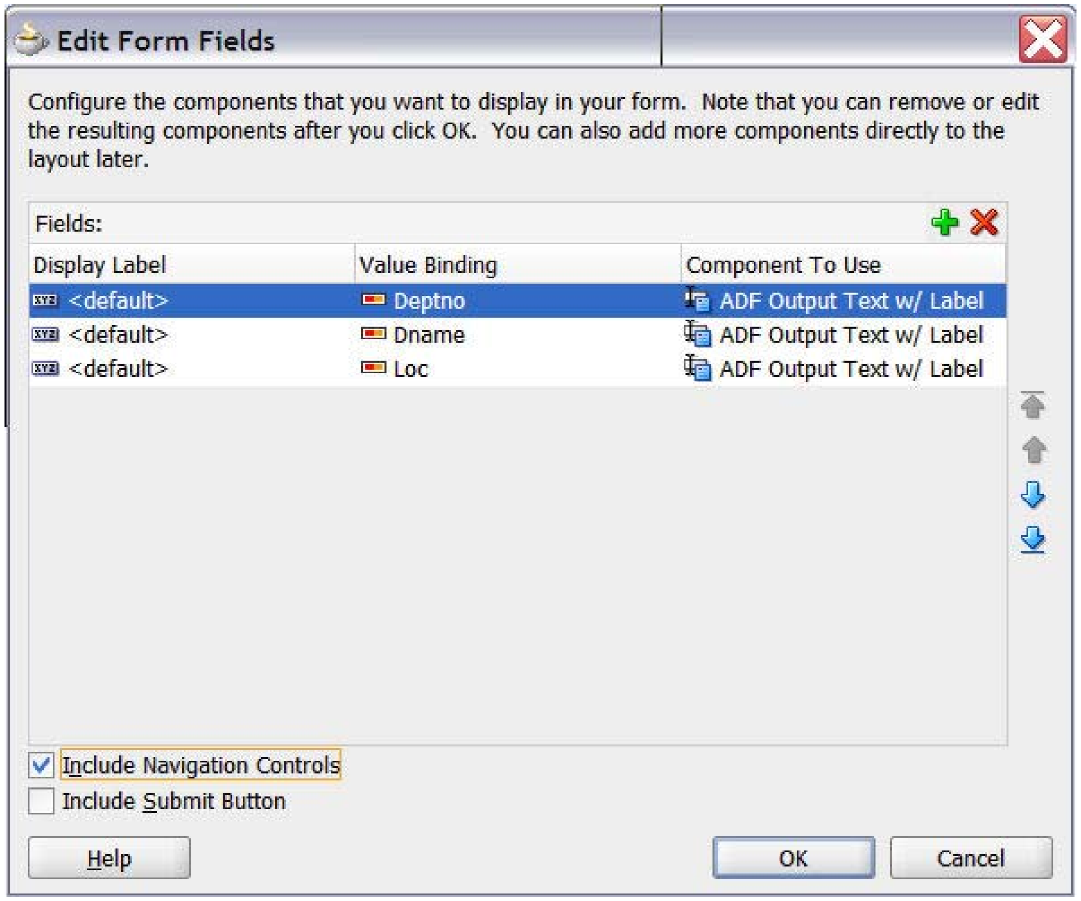
Data Binding: Adding Data, 2

- 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



Edit Form Fields

Configure the components that you want to display in your form. Note that you can remove or edit the resulting components after you click OK. You can also add more components directly to the layout later.

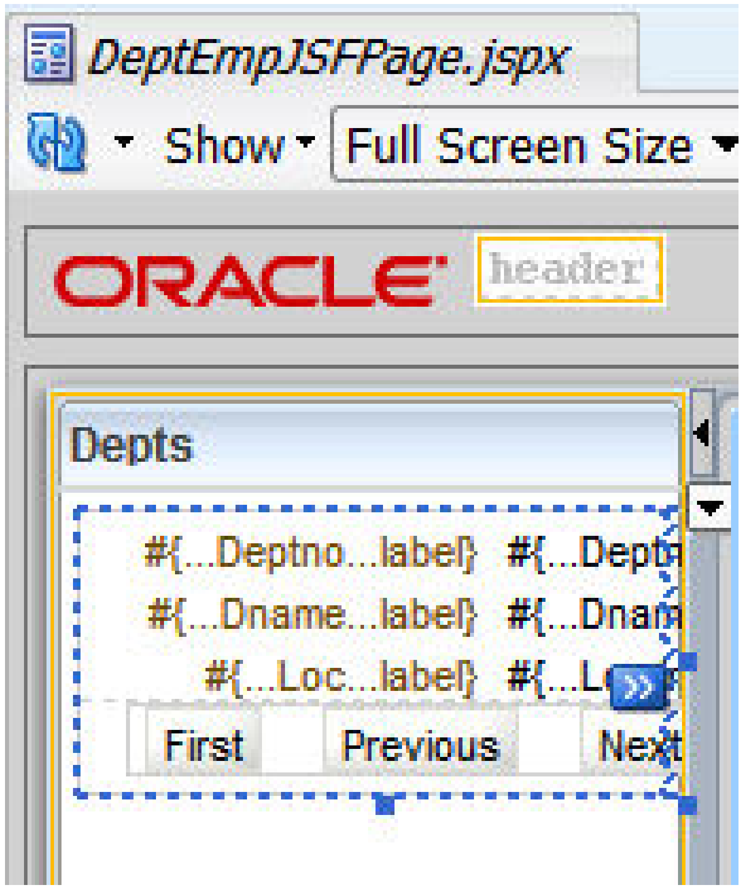
Fields:

Display Label	Value Binding	Component To Use
<default>	Deptno	ADF Output Text w/ Label
<default>	Dname	ADF Output Text w/ Label
<default>	Loc	ADF Output Text w/ Label

[✓] Include Navigation Controls
[ ] Include Submit Button

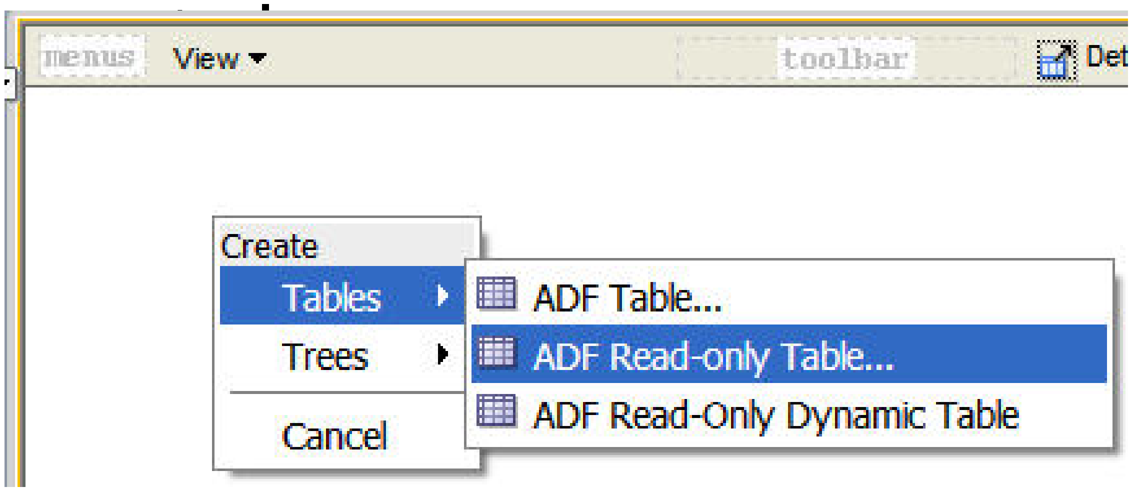
Department Display Area

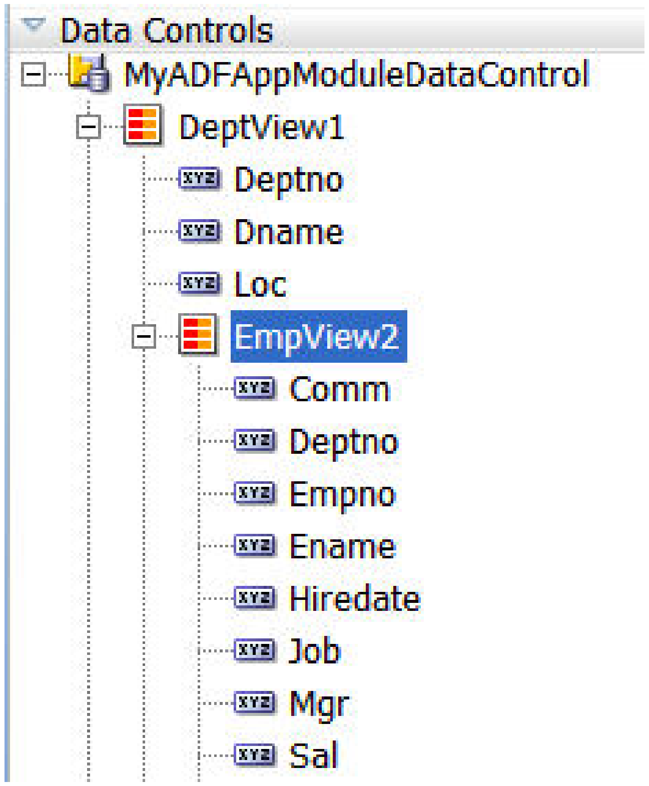
- After adding the Department information; the “Depts” accordion should look like the following



Adding Department Employees

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

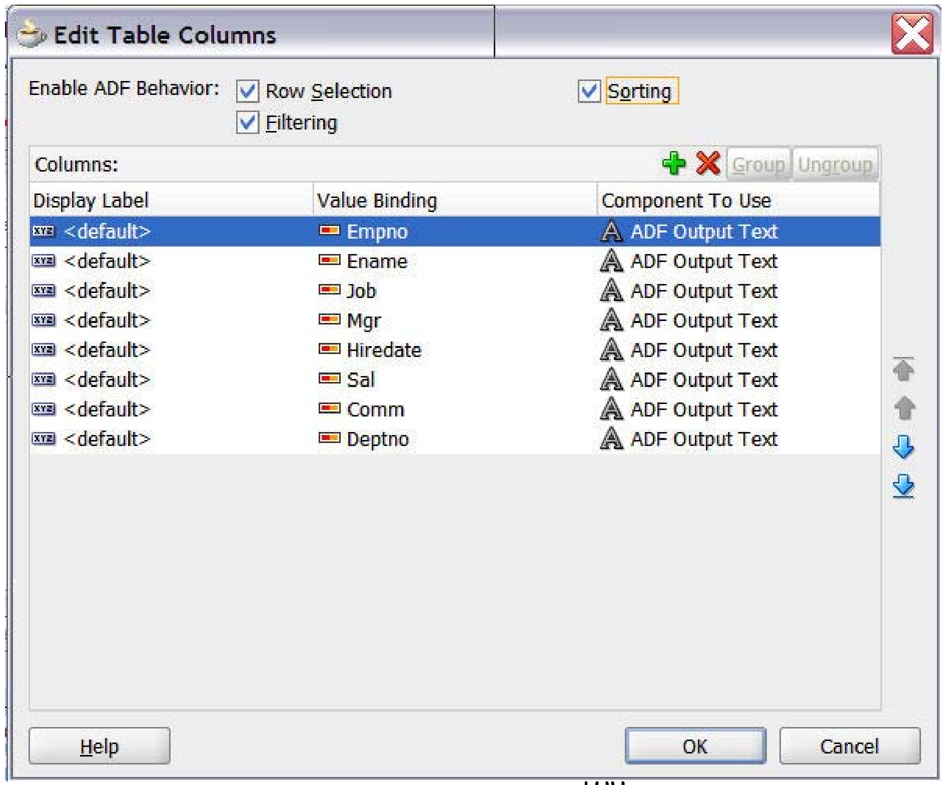




- When prompted, choose “Create Tables -> ADF Read-Only Table” again

Add Employee Navigation Controls

- Check all three navigation controls:
- Row Selection (user may select), filtering (user may search), and sort; as before columns may be relabelled, added, deleted, reorganized



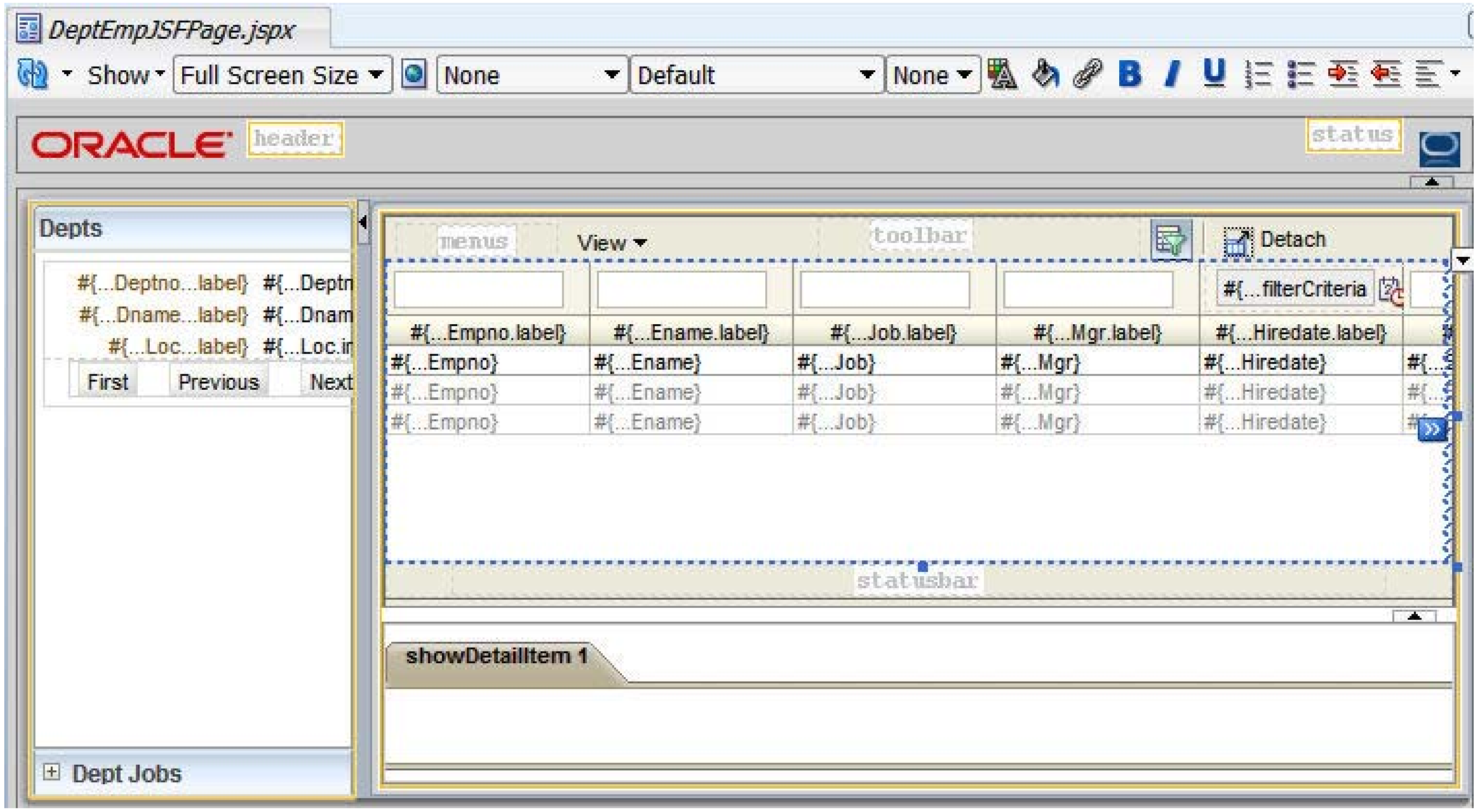
Edit Table Columns

Enable ADF Behavior: [✓ Row Selection] [✓ Filtering] [✓ Sorting]

Columns:

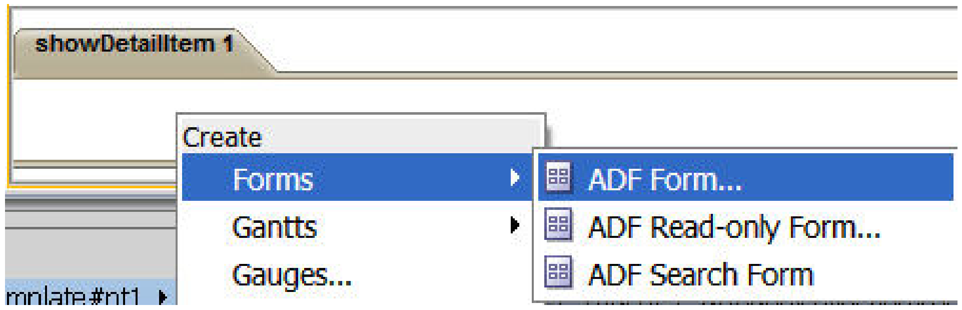
Display Label	Value Binding	Component To Use
<default>	Empno	ADF Output Text
<default>	Ename	ADF Output Text
<default>	Job	ADF Output Text
<default>	Mgr	ADF Output Text
<default>	Hiredate	ADF Output Text
<default>	Sal	ADF Output Text
<default>	Comm	ADF Output Text
<default>	Deptno	ADF Output Text

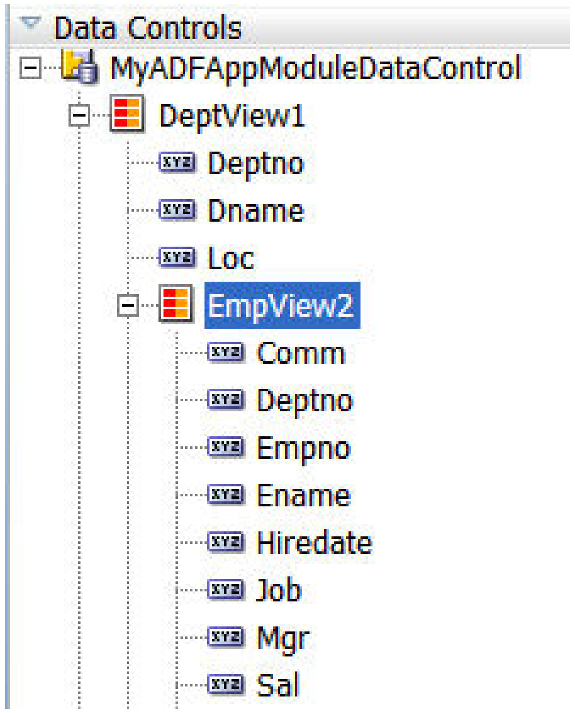
Department Employee Area



Adding Individual Employee

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

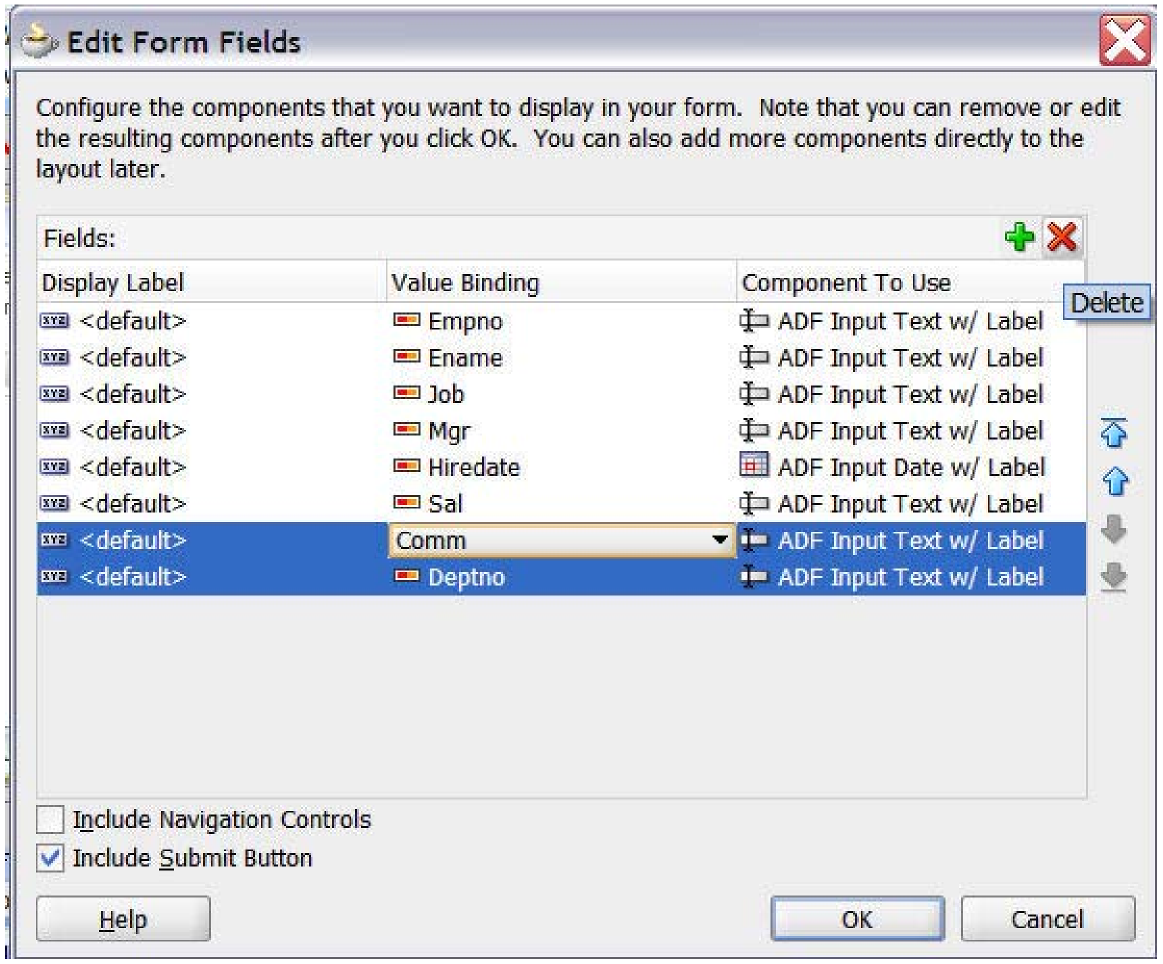




- When prompted, choose “Create Forms -> ADF Form” to select the display format (this part of the form will be editable)

Add Employee Navigation

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



Edit Form Fields

Configure the components that you want to display in your form. Note that you can remove or edit the resulting components after you click OK. You can also add more components directly to the layout later.

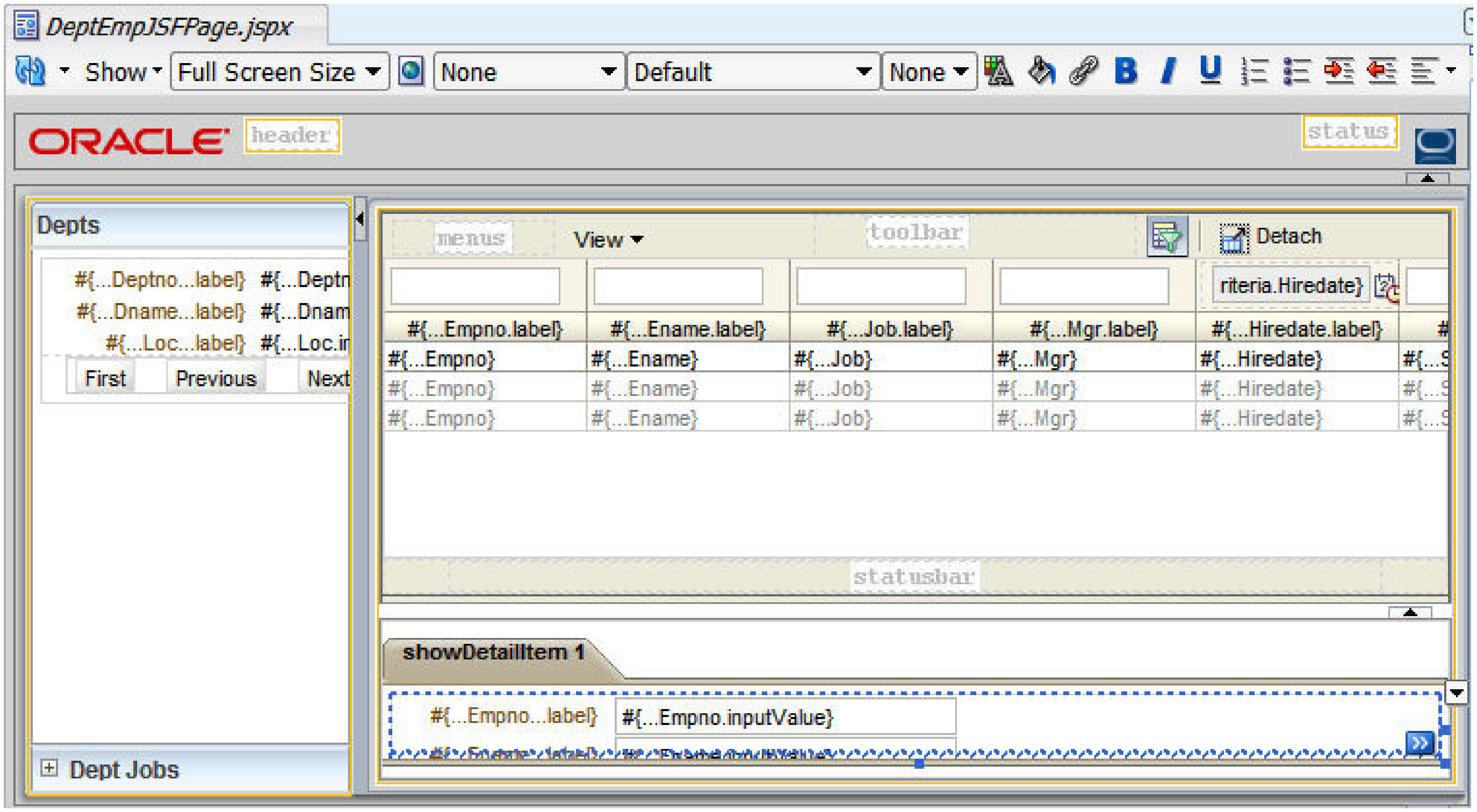
Fields:

Display Label	Value Binding	Component To Use
<default>	Empno	ADF Input Text w/ Label
<default>	Ename	ADF Input Text w/ Label
<default>	Job	ADF Input Text w/ Label
<default>	Mgr	ADF Input Text w/ Label
<default>	Hiredate	ADF Input Date w/ Label
<default>	Sal	ADF Input Text w/ Label
<default>	Comm	ADF Input Text w/ Label
<default>	Deptno	ADF Input Text w/ Label

☐ Include Navigation Controls

☑ Include Submit Button

Completed Web Application Page



DeptEmpJSFPage.jspx

Show · Full Screen Size · None · Default · None

ORACLE header status

Depts

#{...Deptno...label}	#{...Deptn
#{...Dname...label}	#{...Dnam
#{...Loc...label}	#{...Loc.ir

First · Previous · Next

Dept Jobs

menus · View · toolbar · Detach

#{...Empno.label}	#{...Ename.label}	#{...Job.label}	#{...Mgr.label}	#{...Hiredate.label}	#
#{...Empno}	#{...Ename}	#{...Job}	#{...Mgr}	#{...Hiredate}	#{...S
#{...Empno}	#{...Ename}	#{...Job}	#{...Mgr}	#{...Hiredate}	#{...S
#{...Empno}	#{...Ename}	#{...Job}	#{...Mgr}	#{...Hiredate}	#{...S

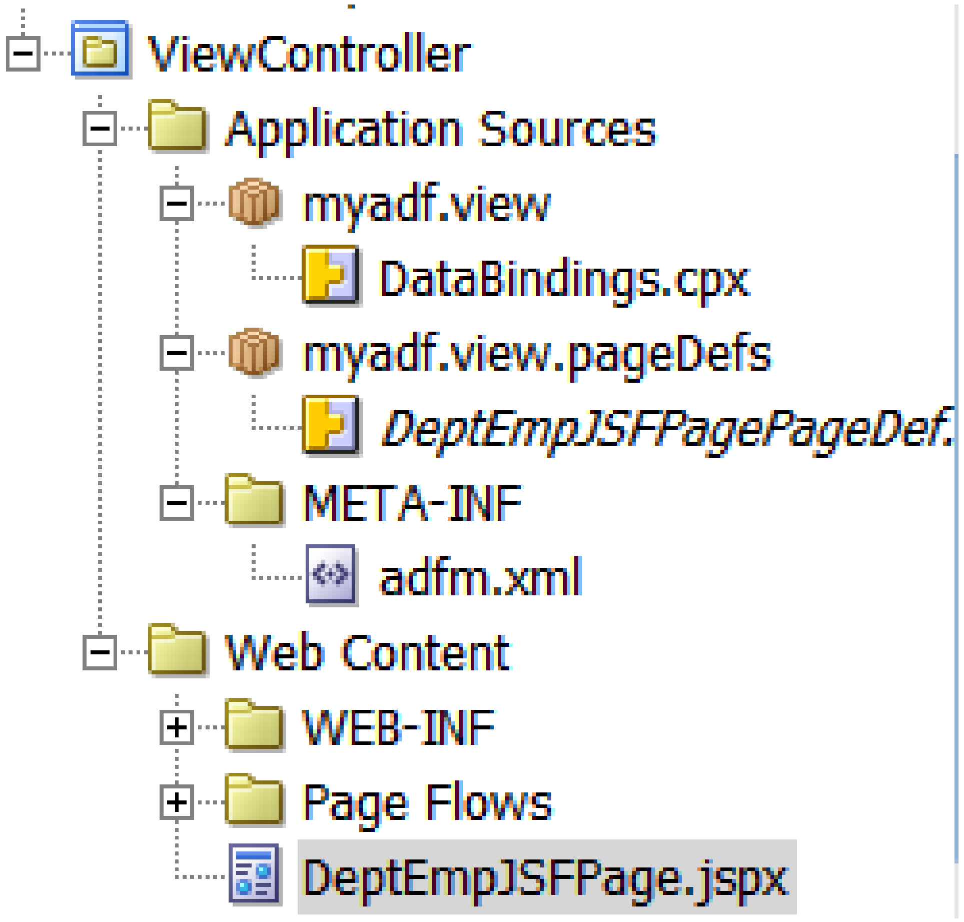
statusbar

showDetailItem 1

#{...Empno...label} #{...Empno.inputValue}

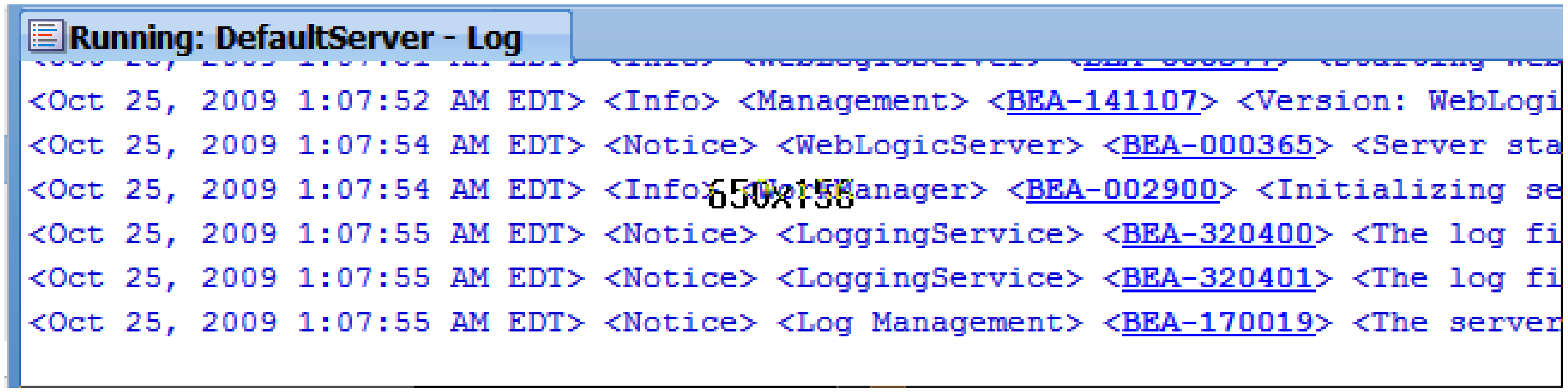
Testing the Web Application

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



Be Patient!

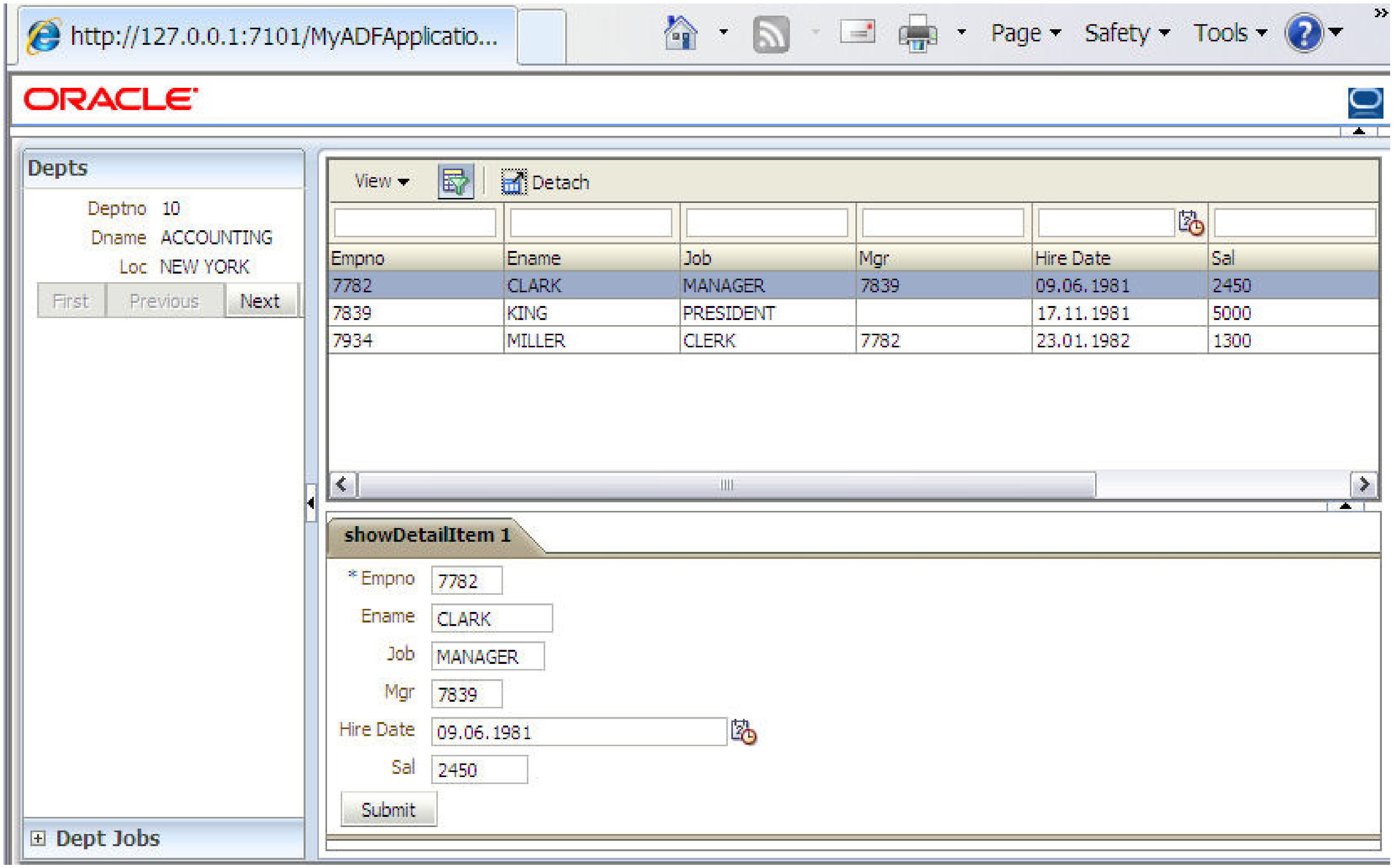
- The first time you execute a Web application JDeveloper will start its built-in WebLogic Application Server; this takes a while
- You can track the progress of the Server's startup in JDeveloper's DefaultServer Log



```
<Oct 25, 2009 1:07:52 AM EDT> <Info> <Management> <BEA-141107> <Version: WebLogi
<Oct 25, 2009 1:07:54 AM EDT> <Notice> <WebLogicServer> <BEA-000365> <Server sta
<Oct 25, 2009 1:07:54 AM EDT> <Info> <WorkManager> <BEA-002900> <Initializing se
<Oct 25, 2009 1:07:55 AM EDT> <Notice> <LoggingService> <BEA-320400> <The log fi
<Oct 25, 2009 1:07:55 AM EDT> <Notice> <LoggingService> <BEA-320401> <The log fi
<Oct 25, 2009 1:07:55 AM EDT> <Notice> <Log Management> <BEA-170019> <The server
```

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

Web Page in Browser



Depts

Deptno	10
Dname	ACCOUNTING
Loc	NEW YORK

Empno	Ename	Job	Mgr	Hire Date	Sal
7782	CLARK	MANAGER	7839	09.06.1981	2450
7839	KING	PRESIDENT		17.11.1981	5000
7934	MILLER	CLERK	7782	23.01.1982	1300

showDetailItem 1

* Empno	7782
Ename	CLARK
Job	MANAGER
Mgr	7839
Hire Date	09.06.1981
Sal	2450

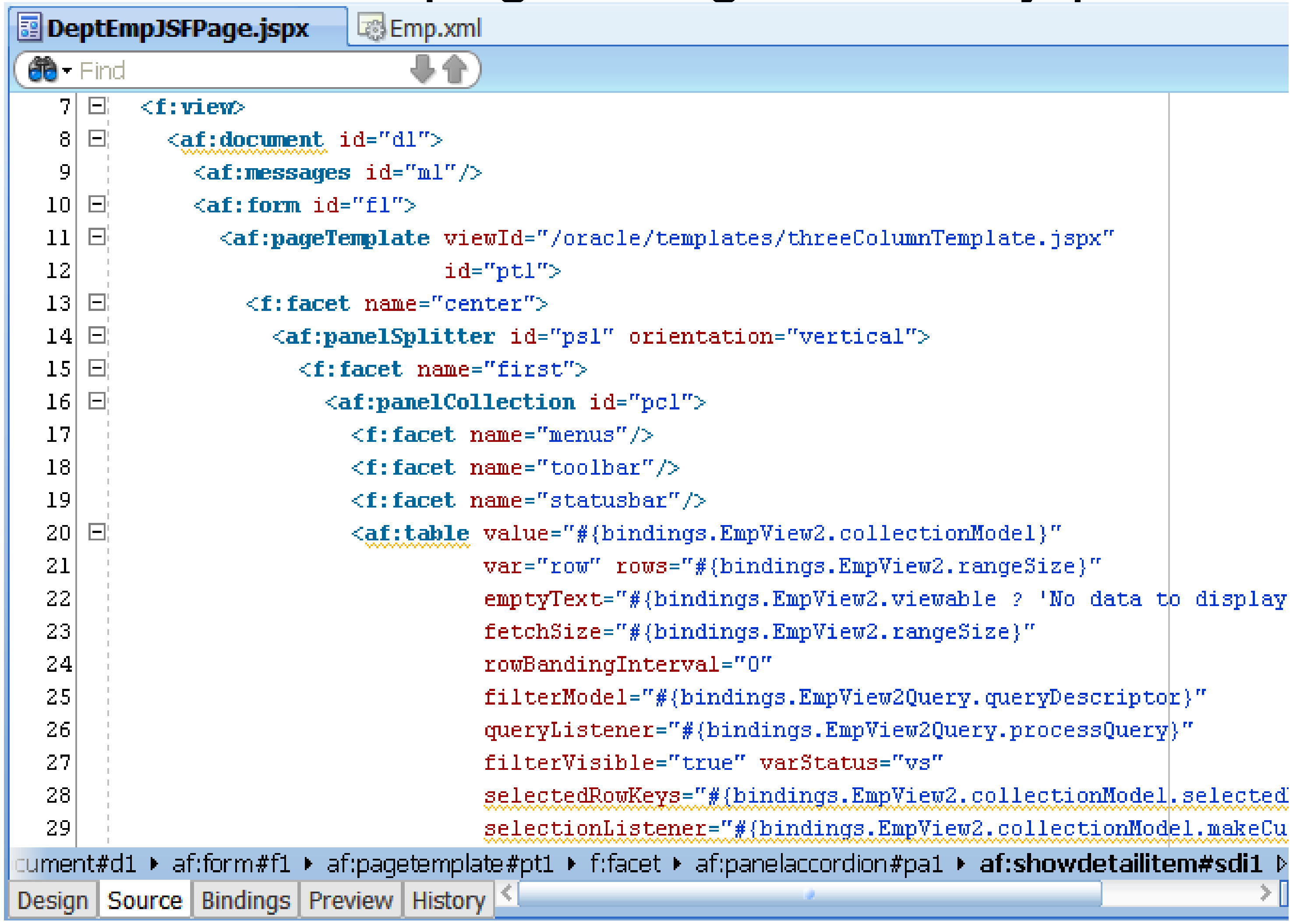
Dept Jobs

Files Supporting Web 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

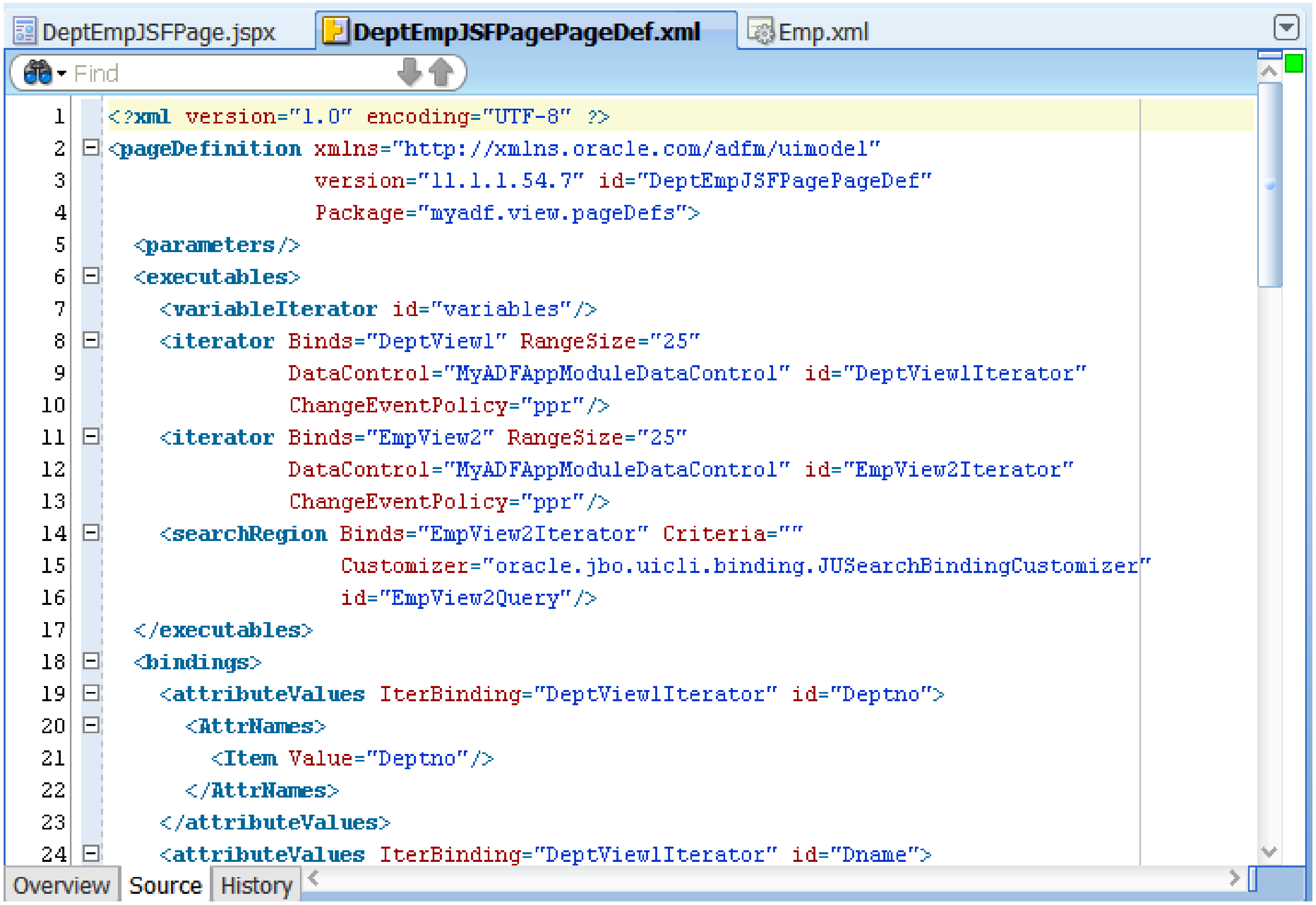
- ADF defines a web page using an XML .jspx file



```
<f:view>
  <af:document id="d1">
    <af:messages id="m1"/>
    <af:form id="f1">
      <af:pageTemplate viewId="/oracle/templates/threeColumnTemplate.jspx"
                       id="pt1">
        <f:facet name="center">
          <af:panelSplitter id="ps1" orientation="vertical">
            <f:facet name="first">
              <af:panelCollection id="pc1">
                <f:facet name="menus"/>
                <f:facet name="toolbar"/>
                <f:facet name="statusbar"/>
                <af:table value="#{bindings.EmpView2.collectionModel}"
                          var="row" rows="#{bindings.EmpView2.rangeSize}"
                          emptyText="#{bindings.EmpView2.viewable ? 'No data to display
                          fetchSize="#{bindings.EmpView2.rangeSize}"
                          rowBandingInterval="0"
                          filterModel="#{bindings.EmpView2Query.queryDescriptor}"
                          queryListener="#{bindings.EmpView2Query.processQuery}"
                          filterVisible="true" varStatus="vs"
                          selectedRowKeys="#{bindings.EmpView2.collectionModel.selected
                          selectionListener="#{bindings.EmpView2.collectionModel.makeCu
```

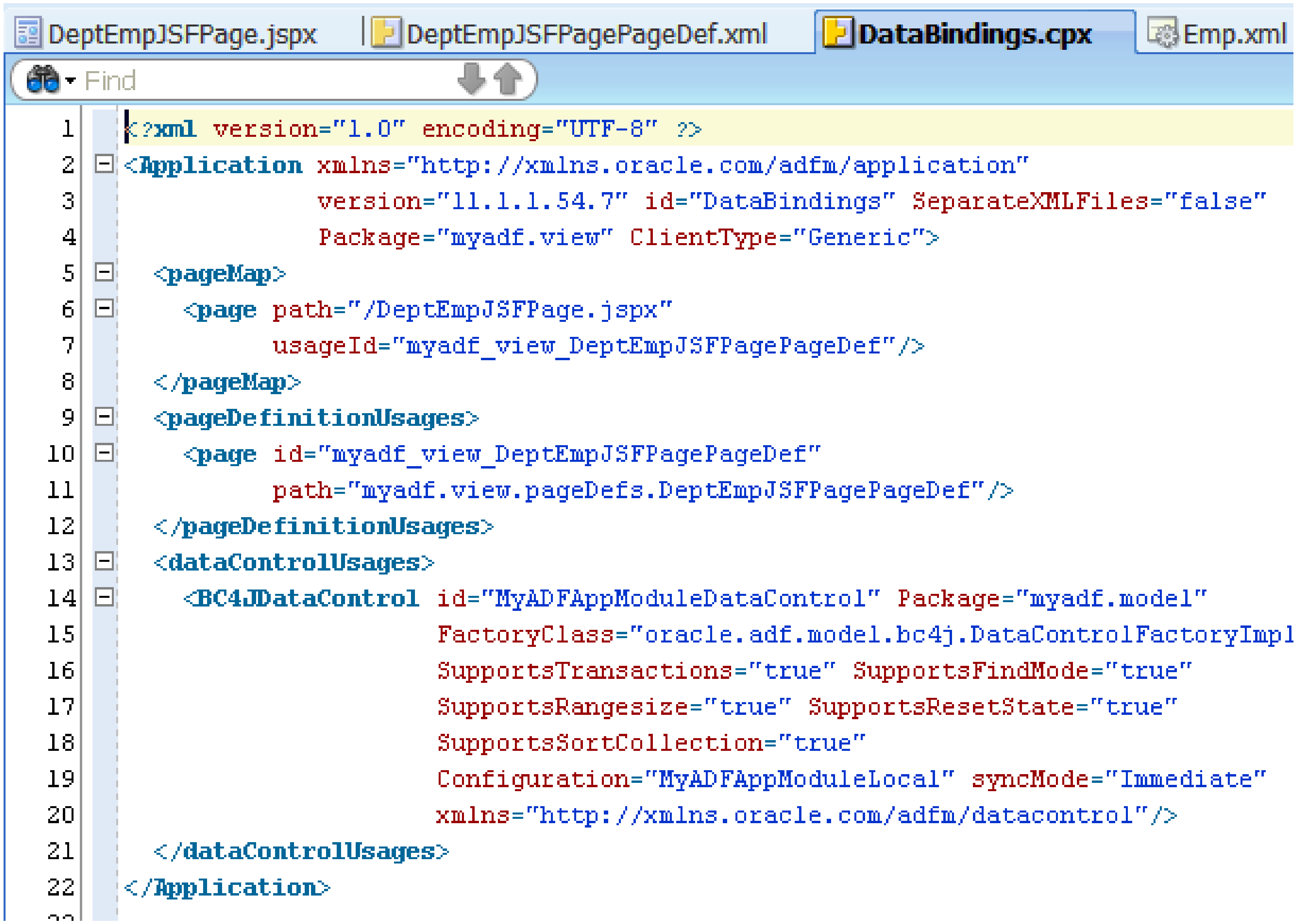
cument#d1 ▸ af:form#f1 ▸ af:pagetemplate#pt1 ▸ f:facet ▸ af:panelaccordion#pa1 ▸ af:showdetailitem#sdi1
Design
Source
Bindings
Preview
History

ADF Web Page Definition file (.xml)



```
<?xml version="1.0" encoding="UTF-8" ?>
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
                version="11.1.1.54.7" id="DeptEmpJSFPagePageDef"
                Package="myadf.view.pageDefs">
  <parameters/>
  <executables>
    <variableIterator id="variables"/>
    <iterator Binds="DeptView1" RangeSize="25"
              DataControl="MyADFAppModuleDataControl" id="DeptView1Iterator"
              ChangeEventPolicy="ppr"/>
    <iterator Binds="EmpView2" RangeSize="25"
              DataControl="MyADFAppModuleDataControl" id="EmpView2Iterator"
              ChangeEventPolicy="ppr"/>
    <searchRegion Binds="EmpView2Iterator" Criteria=""
                  Customizer="oracle.jbo.uicli.binding.JUSearchBindingCustomizer"
                  id="EmpView2Query"/>
  </executables>
  <bindings>
    <attributeValues IterBinding="DeptView1Iterator" id="Deptno">
      <AttrNames>
        <Item Value="Deptno"/>
      </AttrNames>
    </attributeValues>
    <attributeValues IterBinding="DeptView1Iterator" id="Dname">
```

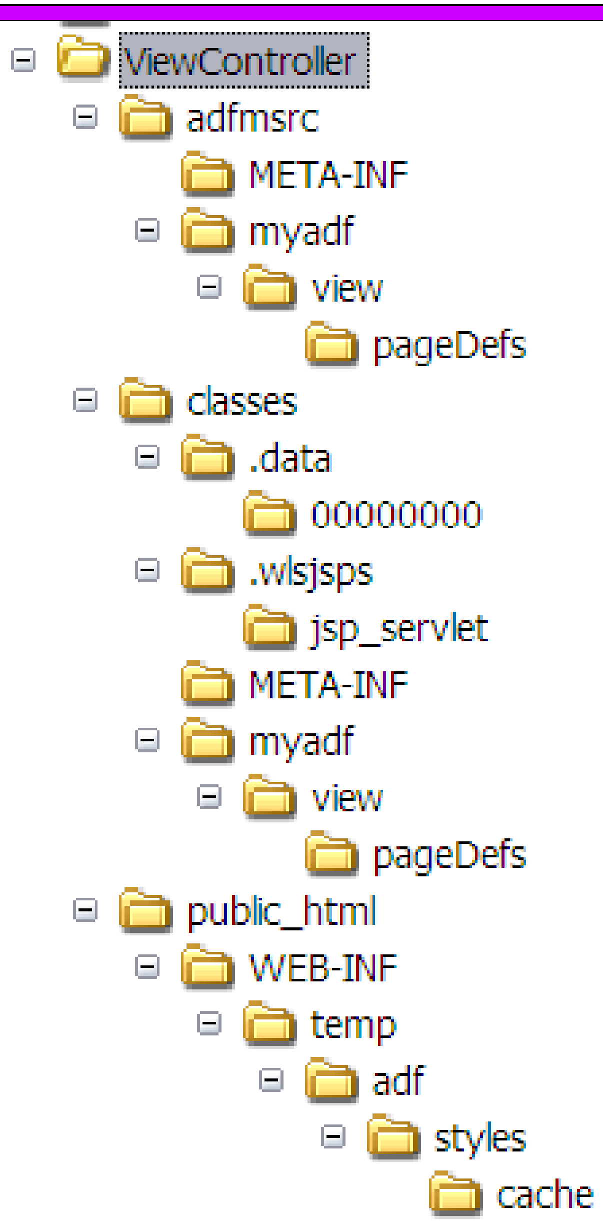
ADF Bindings XML file (.cpx)



```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Application xmlns="http://xmlns.oracle.com/adfm/application"
             version="11.1.1.54.7" id="DataBindings" SeparateXMLFiles="false"
             Package="myadf.view" ClientType="Generic">
  <pageMap>
    <page path="/DeptEmpJSFPage.jspx"
          usageId="myadf_view_DeptEmpJSFPagePageDef"/>
  </pageMap>
  <pageDefinitionUsages>
    <page id="myadf_view_DeptEmpJSFPagePageDef"
          path="myadf.view.pageDefs.DeptEmpJSFPagePageDef"/>
  </pageDefinitionUsages>
  <dataControlUsages>
    <BC4JDataControl id="MyADFAppModuleDataControl" Package="myadf.model"
                     FactoryClass="oracle.adf.model.bc4j.DataControlFactoryImpl
                     SupportsTransactions="true" SupportsFindMode="true"
                     SupportsRangesize="true" SupportsResetState="true"
                     SupportsSortCollection="true"
                     Configuration="MyADFAppModuleLocal" syncMode="Immediate"
                     xmlns="http://xmlns.oracle.com/adfm/datacontrol"/>
  </dataControlUsages>
</Application>
```

ADF Faces ViewController Files

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



```
⊟ ViewController
  ⊟ adfmsrc
      META-INF
    ⊟ myadf
      ⊟ view
          pageDefs
  ⊟ classes
    ⊟ .data
        00000000
    ⊟ .wlsjsps
        jsp_servlet
      META-INF
    ⊟ myadf
      ⊟ view
          pageDefs
  ⊟ public_html
    ⊟ WEB-INF
      ⊟ temp
        ⊟ adf
          ⊟ styles
              cache
```

Available Books

- Oracle JDevloper 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
- 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 for data creates reusable components
  - ADF Faces for view creates reusable components
- I did not write a single line of Java in this demo!

RMOUG

ROCKY MOUNTAIN ORACLE USERS GROUP

Training Days 2011

Save the dates!

February 15-17 2011

ODTUG Kaleidoscope 2010

ODTUG
2010
KALEIDOSCOPE
Washington, D.C. ▪ June 27-July 1 ▪ Marriott Wardman Park Hotel

June 27-July 1 2010
Washington, DC

Please fill out session Evaluations

Oracle ADF & JDeveloper for Forms Developers

To contact the author:

John King

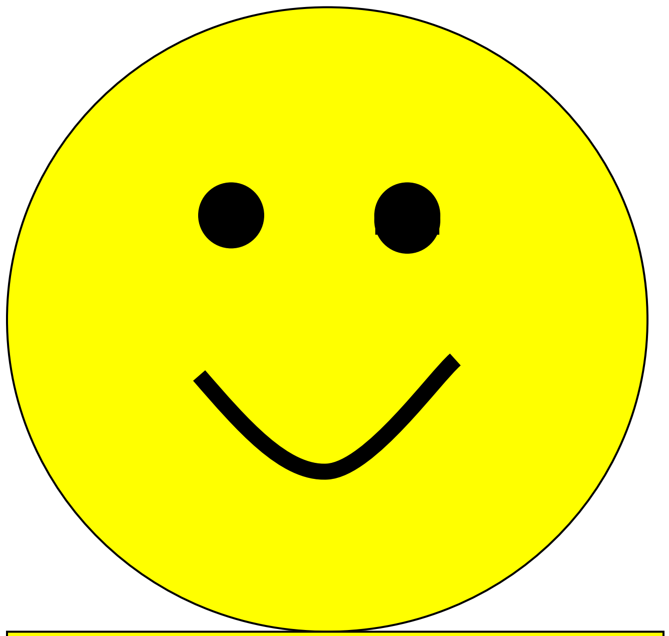
King Training Resources

6341 South Williams Street

Littleton, CO 80121-2627 USA

1.800.252.0652 - 1.303.798.5727

Email: john@kingtraining.com



Thanks for your attention!

Today’s slides are on the web:

http://www.kingtraining.com