

# ADF 101- in a Nutshell

## Overview and Hands-on Lab

Peter Koletzke

Technical Director & Principal Instructor



quovera



On the Positive Side...

If we do not find  
anything pleasant, at least  
we shall find something new.

Si nous ne trouvons pas des choses  
agréables, nous trouverons du  
moins des choses nouvelles.

—Voltaire (1694-1778), *Candide*

quovera

2

## Objectives

- Understanding of the ADF development process
  - Lecture sections
  - Declarative and visual
- Experience with ADF 12c (current version)
  - Hands-on lab
  - Labs do not explain “why” or “what”
    - Lectures and book should help
- About the book
  - All 11g concepts in it apply to 12c
  - Much longer hands-on
  - You may need to adjust some steps

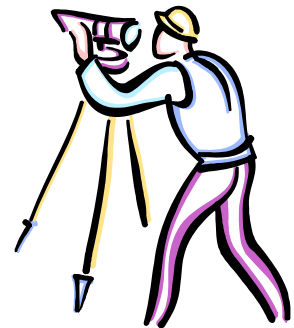


quovera

3

## Survey

- Languages?
  - PL/SQL
  - Java
  - Other
- Tools?
  - APEX
  - JDeveloper
  - Eclipse, NetBeans
  - Developer Forms/Reports
  - Other



quovera

4

# Agenda

- Intros: ADF, sample app, JDev
- ADF Business Components
- ADF Faces
- ADF Model and ADF Controller

Slides will be available on the NoCOUG and Quovera websites.

Screenshots are 11g unless marked as 12c

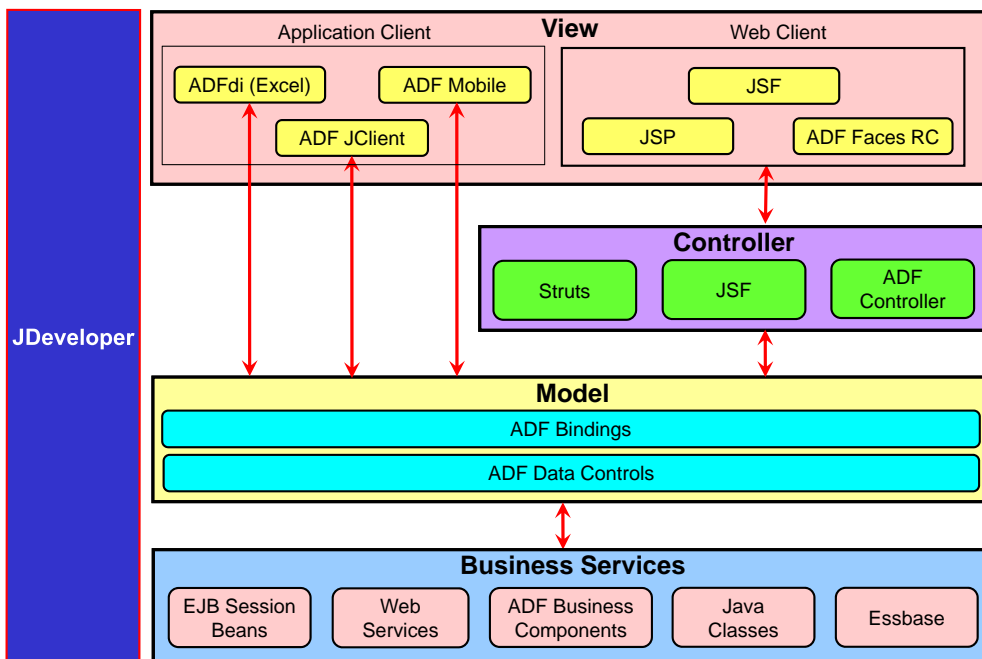


# Oracle Application Development Framework (ADF)

- A *framework* is a prebuilt service for solving a particular problem – like access to the database
  - Code libraries and standards support the framework
  - Implements code reuse and best practices
  - An architecture with code libraries
- ADF is a *meta-framework*
  - A wrapper for other frameworks
  - Available starting in JDeveloper 10g
  - Provides a consistent developer experience
- Pre-ADF available in OAF
  - Oracle Application Framework (UIX/MVC)
- Based on Model-View-Controller design pattern (Java EE, SmallTalk)



# ADF Architecture



# Which ADF Technologies to Use?

- An example: Fusion Applications (Oracle applications suite) uses ADF 11g
- *Core technology stack* used for Fusion Apps is:
  - ADF Business Components
  - ADF Faces Rich Client
  - ADF Model
  - ADF Controller
- Other *high-level technologies* or strategies also used
  - SOA, ESB, Business Rules, WebCenter, BPM, BPA, BAM
  - Need to consider those, too, **Out of scope** at the architectural level



# ADF Essentials

- No-license-fee version of ADF
  - Runs on the public domain app server, Glassfish, not WebLogic Server
- Works in JDeveloper
- Works in Eclipse
  - Through Oracle Enterprise Pack for Eclipse
- More information on OTN
  - <http://www.oracle.com/technetwork/developer-tools/adf/overview/adfessentials-1719844.html>

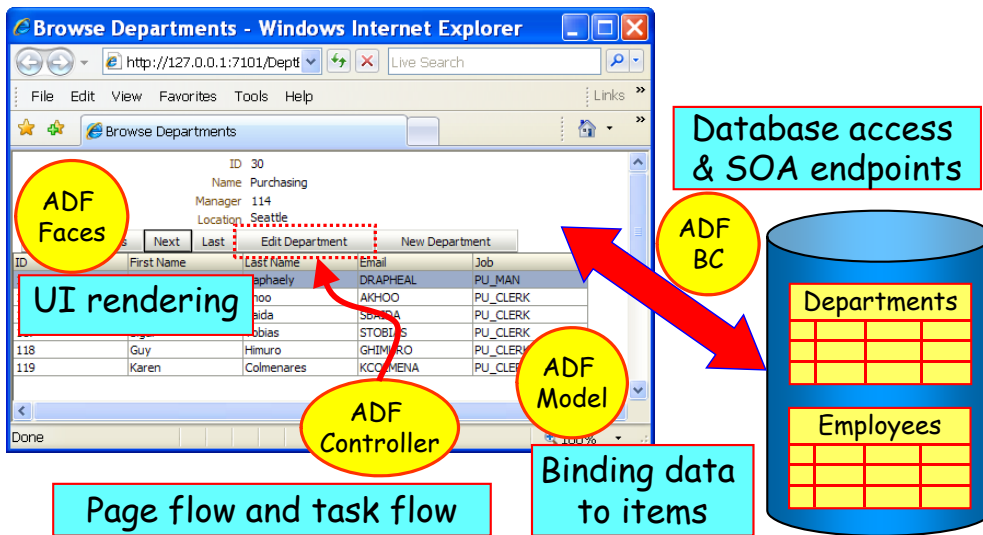


# ADF Mobile

- Similar ADF principles and development methods
  - Business services
  - Pages with components (amx)
  - Task flows
- Supports iOS and Android
- Use local services like camera & contacts
- OTN demo video and ADF Academy video
  - [www.oracle.com/technetwork/developer-tools/adf-mobile](http://www.oracle.com/technetwork/developer-tools/adf-mobile)

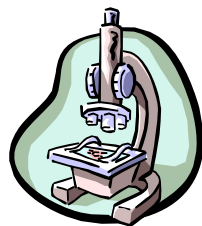


# Where Do The Core Technologies End Up?

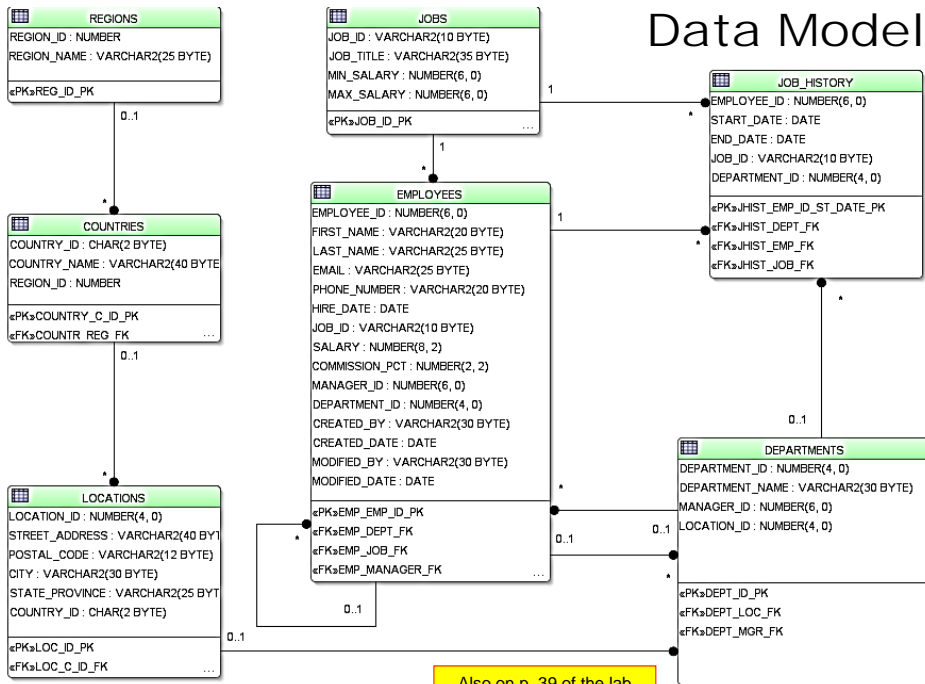


# About the Hands-on Lab

- Sample app
  - Required sections (A – L)
    - Basic functions and techniques
  - Extra credit sections (M – R)
    - More practice, more techniques
- Work together
- How to succeed
  - Take your time
  - Ask questions
  - Read carefully *Use the sticky arrows!*
  - Watch white board for corrections
  - Use sample solutions – compare files

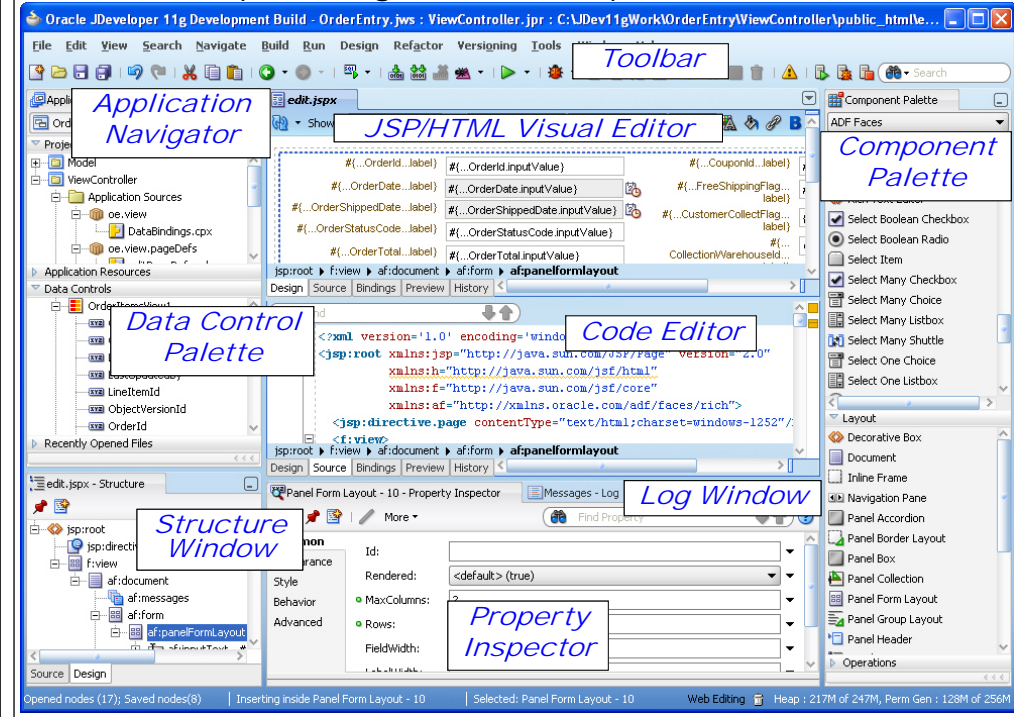


## Data Model



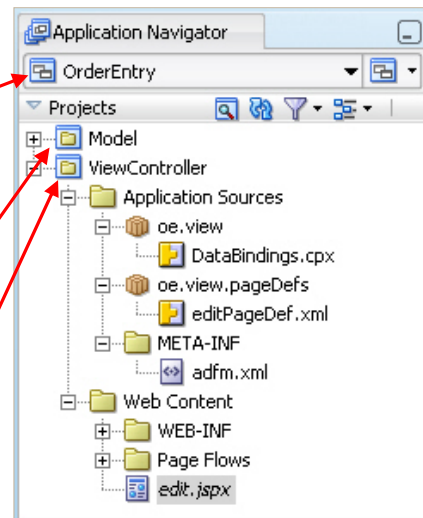
Also on p. 39 of the lab handout and in the sample app's UML project.

## JDeveloper Integrated Development Environment



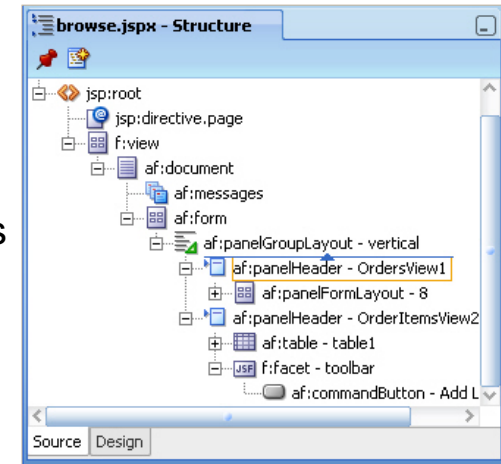
## Application Navigator

- Shows files that are part of your work session
  - Application
    - Consists of a number of project files
    - Opening the application opens a number of files
    - Open multiple applications at a time
  - Project
    - A number of files
    - The deployment unit



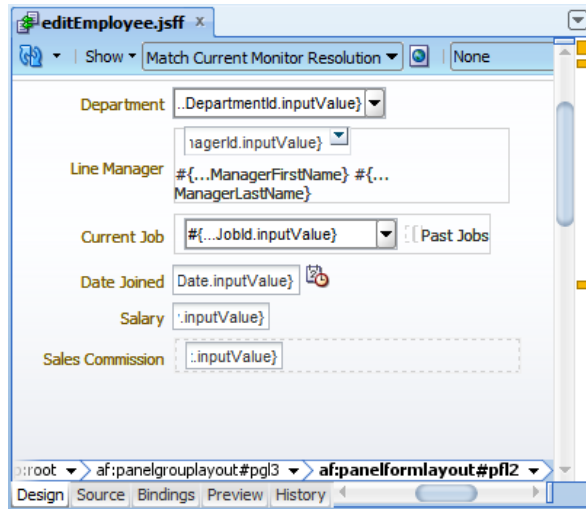
## Structure Window

- Shows details of selected code
  - Shows HTML and XML structures
  - Shows Java elements
  - Drag and drop components to it
  - Use right-click menu to add components to the page



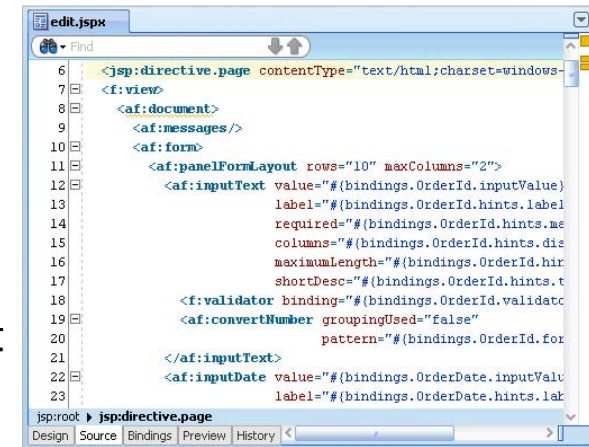
# JSP/HTML Visual Editor

- WYSIKOWYG development
- Shows visual aspects
- Drag and drop components to the editor
- Drag and drop components on the editor



# XML Code Editor

- You can type in code if needed
- Different views of the file
- Many, many features to assist with code entry
  - Code insight
  - Javadoc
  - Auto import

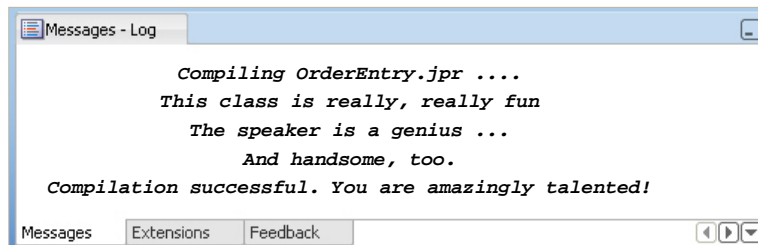


Not BMW, Isuzu, Audi, Acura



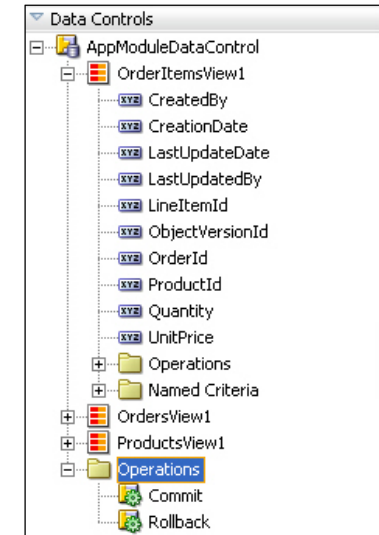
# Log Window

- Displays the Java “console” messages
  - Displays compile messages
  - System print statements
  - Debug messages
  - Configure in **Tools | Preferences**



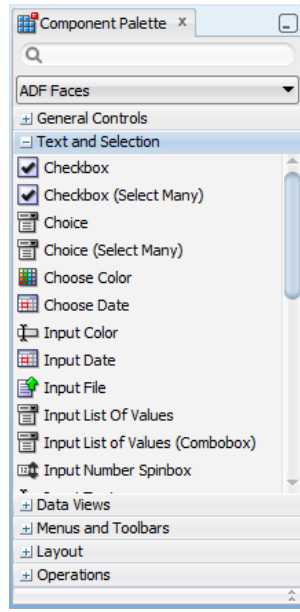
# Data Controls Panel

- Part of the navigator window
- A mechanism for dragging components into the visual editor
  - Dragged controls are fully bound to data in the Business Services layer
- Uses same basic components as the Component Palette



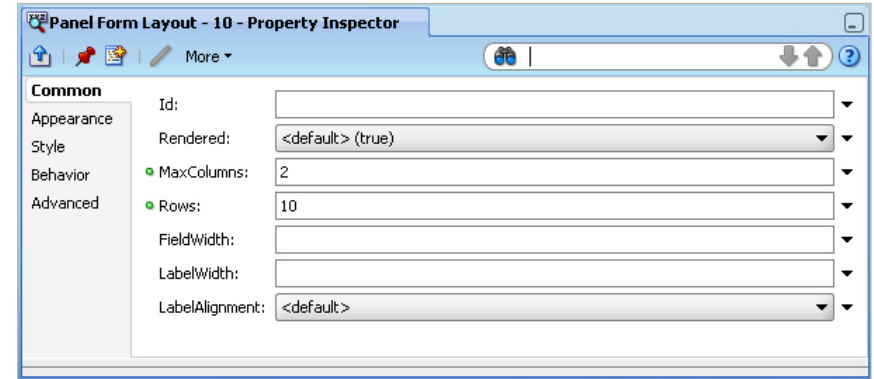
# Component Palette

- Drag from this palette into the visual editor
- Creates code in the Code Editor
- Appropriate palettes for different types of files
  - For example, JSP files have HTML, JSP, JSF palettes



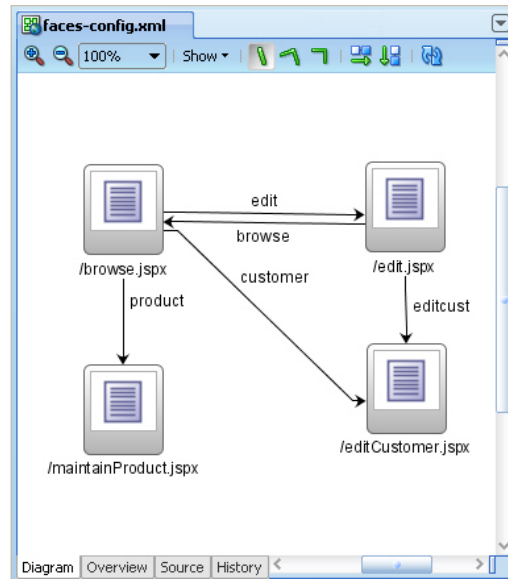
# Property Inspector

- Standard way to code property values
- Entries here show in Code Editor
- Properties accessible for any element



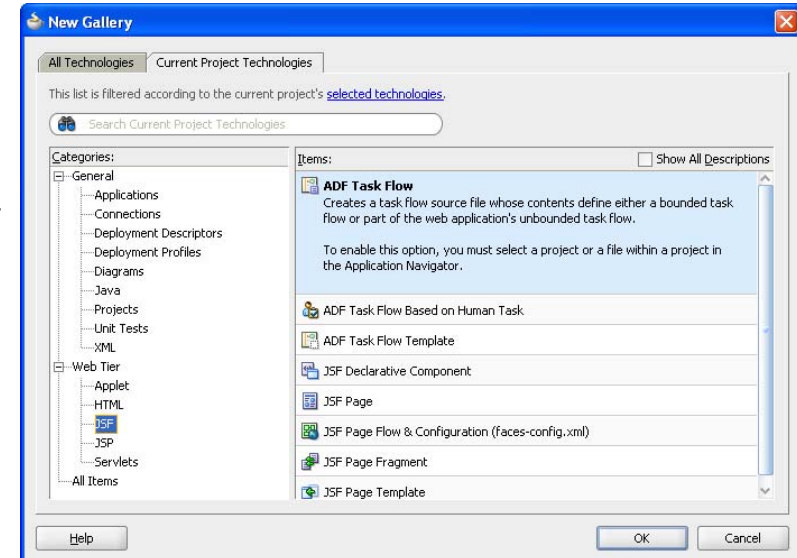
# JSF Navigation/Task Flow Diagram

- Draw JSF pages and notes
- Creates and edits the `faces-config.xml` or `adf-config.xml` file
- This defines which pages are displayed based on user actions



# New Gallery

- **File | New**
- Starts up a wizard
- Use for all new files



## Some Other UI Features

- Database Utilities
- Debugger
- UML Modelers
  - Class Diagram
    - For ADF BC, Java Classes, Web Services, Database objects
  - Activity Modeler
  - Use Case Diagram
  - Sequence Diagram
- XML Schema Diagram
- Integrated WebLogic Server
- Software Configuration Management



## Agenda

- Intros: ADF, JDev, sample app
- ADF Business Components
- ADF Faces
- ADF Model and ADF Controller



## ADF Business Components

- ADF BC: an option in the Business Services layer of ADF
- **Persistence**: storing data in a database
- **O/R mapping**: Translates relational (database) things to object-oriented (Java) whatsits
- Handles JDBC mechanics
  - Creates SQL and handles results
- Primarily declarative
  - XML source code to define the use of framework classes



## More About ADF BC

- Various component types, e.g.:
  - View objects: define queries
  - View links: view object relationships
  - Entity objects: define insert-update-delete (“DML”)
  - Application modules: Define the data models and the database transaction
- It does not create user interfaces



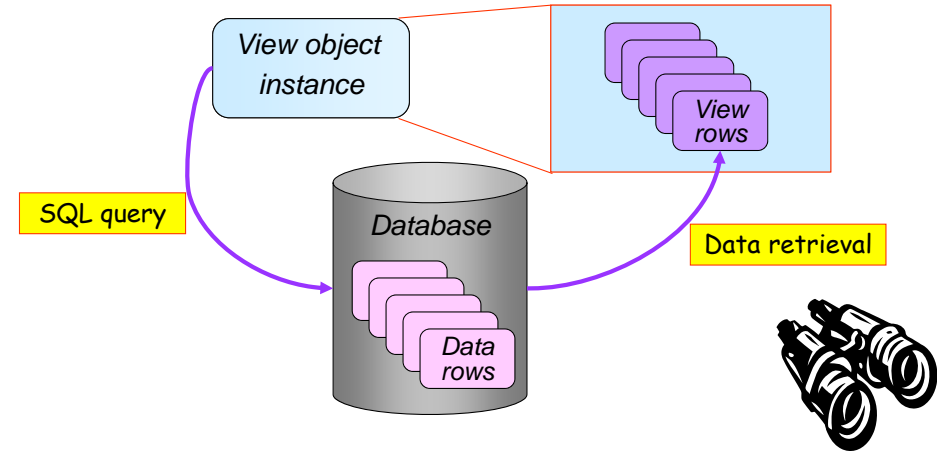
## View Objects = SELECT

- Retrieve data from the database
- Manage caches of data
- *View object definitions*
  - Contain SQL queries
  - Act as templates for *view object instances*
- *View object instances*
  - The “thing” used in code
  - Technically, an object of the class



## Read-Only View Objects

- Each view object instance can execute a SQL query and cache the results.



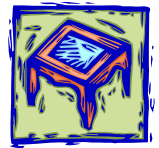
## ADF View Links

- *View link definitions* relate a pair of view object definitions
- Specifies a master/detail relationship between the SQL queries
- Has a source and destination attribute
  - Like foreign key/primary key relationship



## Entity Objects = INSERT, UPDATE, DELETE

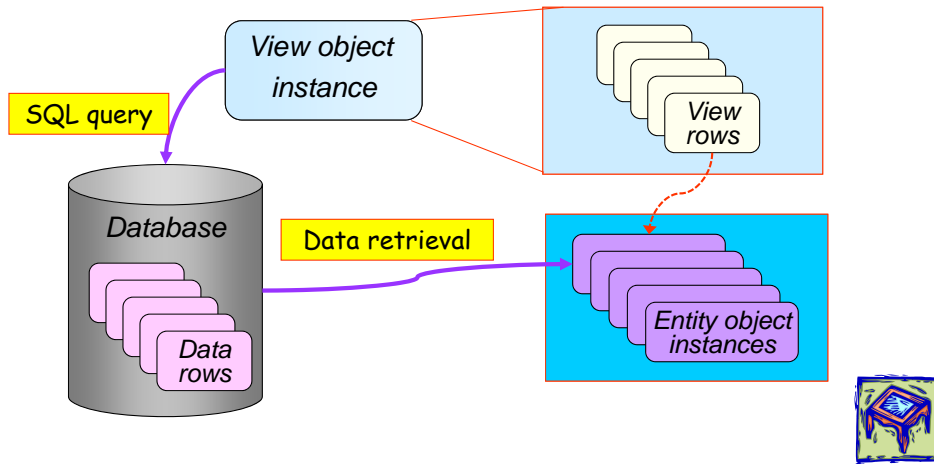
- *Entity object definitions*
  - Correspond to database tables (or views)
  - Templates for *entity object instances*, which correspond to table rows
- *Entity object instances*
  - Write data to the database
  - Validate data before updates and inserts
  - Supply default values





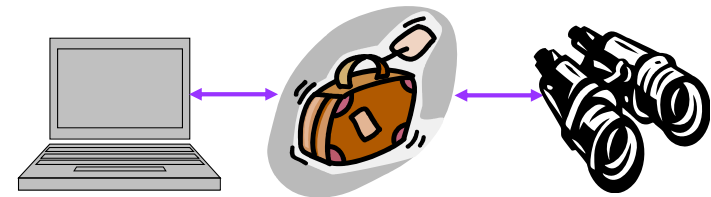
# Entity Object Instances

- View object instances stores data in **entity object instances** - view rows point to it

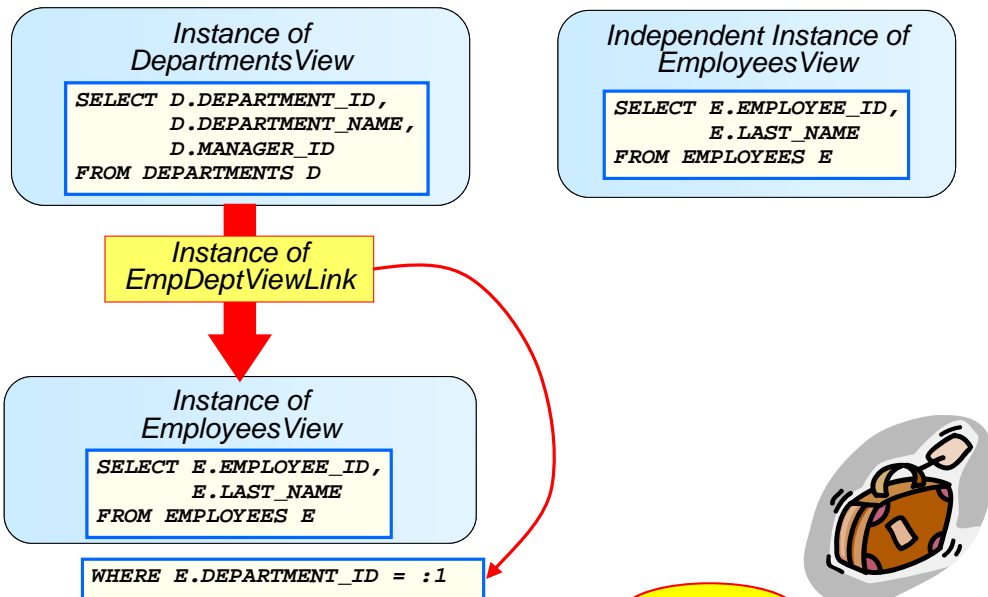


# Application Modules

- Application modules represent the interface from business components to the application
  - The application always references the application module for data
  - Each instance (user) of the application will get its own application module instance
- The application module instance contains **the data model**
  - View object instances
  - View link instances for master-detail relationships



# A Data Model

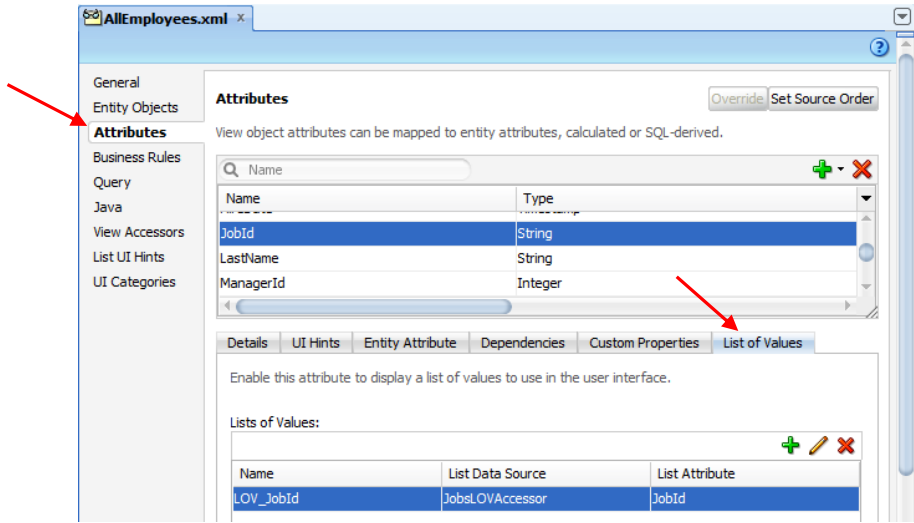


# List-of-Values (LOV)

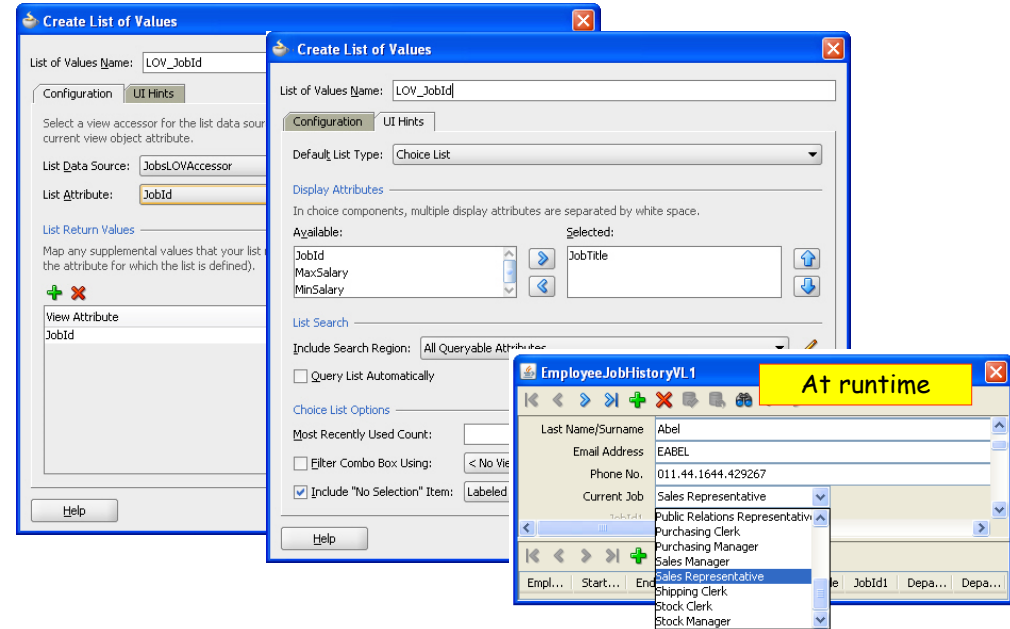
- Defined for an view object attribute
- Associate a query (view object) with the attribute
- Set up is declarative
  - Start in the view object editor
- UI code will display this attribute specially
  - Pulldown list item loaded with values
  - Popup list-of-values dialog



# VO Editor – LOV Subtab

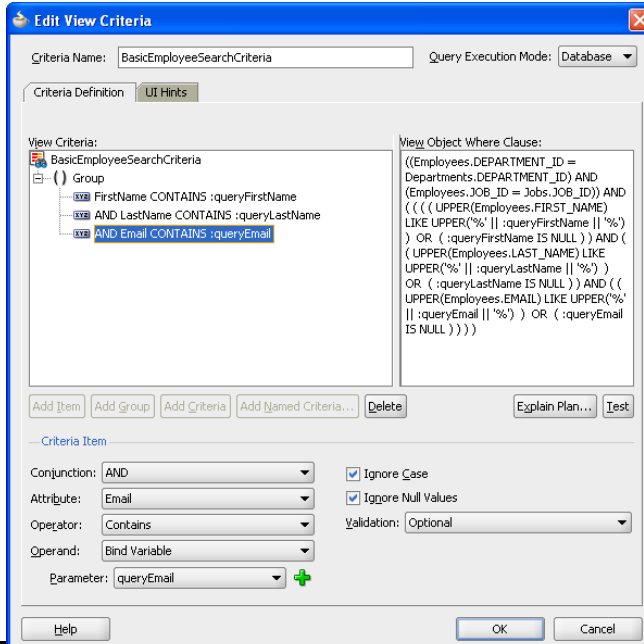


# Create List of Values



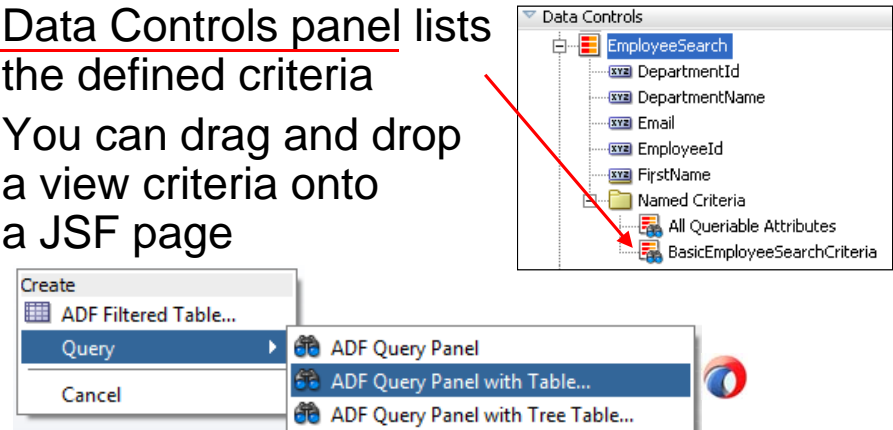
# View Criteria

- Used for queries
- View Object Editor
  - 11g Query page
  - 12c View Criteria page
- View Criteria dialog
  - Defines the WHERE clause declaratively
  - Bind variables are optional



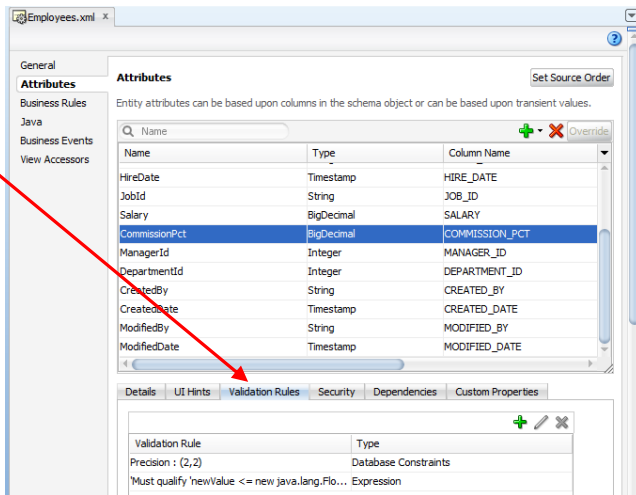
# View Criteria in the DCP

- Data Controls panel lists the defined criteria
- You can drag and drop a view criteria onto a JSF page
- Components are added for each attribute in the view criteria



# Declarative Validation Rules

- Simple rules written on the entity object
- Method 1
  - Attributes tab
  - Validation Rules subtab
- Method 2
  - Business Rules tab
  - Attribute or entity level



# Declarative Validation Benefits

- Quick validation on the app server side
- No Java coding!
- Add a friendlier message for ADF BC-level errors (e.g., length)
- Messages stored in a message bundle file that you can internationalize
- All UIs built from the entity object will contain the same validation
  - Like a trigger in the database
- You need to decide where to place business rules code



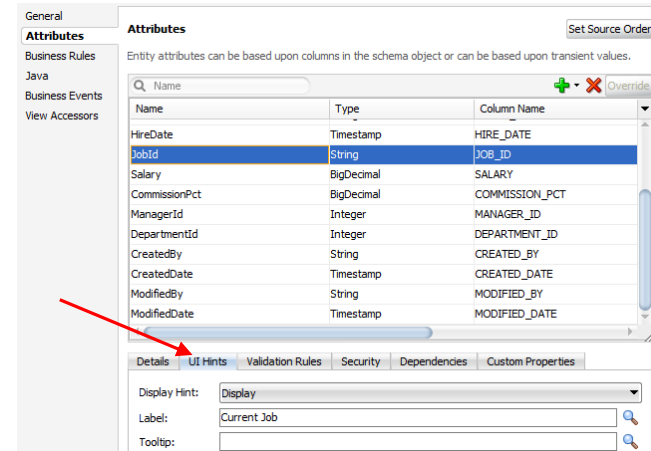
# UI Hints

- A.k.a., *control hints*
- UI definitions stored in the entity object or view object definitions
  - Universal to all UIs created from those EOs or VOs
  - Provides consistency
  - “Set it and forget it”
- Can put the hints on either EO or VO
  - Rule of thumb: use the entity object UI hints whenever possible
  - Override on view object level if needed



# Setting UI Hints

- In the editor, Attributes tab, select attribute and click the UI Hints tab in the Property Inspector
- **OR** Use the UI Hints subtab



# Actual California State Laws

- California State
  - It is a misdemeanor to shoot at any kind of game from a moving vehicle, unless the target is a whale.
  - Women may not drive in a house coat.
  - No vehicle without a driver may exceed 60 miles per hour.
- Baldwin Park, CA
  - Nobody is allowed to ride a bicycle in a swimming pool.
- San Francisco, CA
  - It is illegal to pile horse manure more than six feet high on a street corner.
- Carmel, CA
  - Women may not wear high heels while in the city limits.
- Palm Springs, CA
  - It is illegal to walk a camel down Palm Canyon Drive between the hours of four and six PM.

*Sunshine is guaranteed to the masses.*



[www.dumblaws.com](http://www.dumblaws.com)

quovera

45

# Agenda

- Intros: ADF, JDev, sample app
- ADF Business Components
- ADF Faces
- ADF Model and ADF Controller



quovera

46

# ADF Faces Rich Client Overview

- Fits into the View layer of ADF – ADF Faces RC
- Evolution:
  - ADF UIX → ADF Faces → Apache Trinidad
  - ADF Faces → ADF Faces RC
- Built on top of JSF APIs
- Deployable on any 1.2 implementation of JSF
- Support for pop-ups and dialogs
- ADF model support out-of-the-box
- Data Visualization Tools (DVT) components
  - Charts, Gantt, Pivot, Maps, Hierarchy

Really rich!

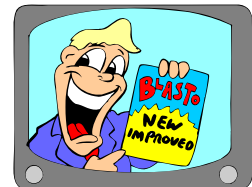


quovera

47

# ADF Faces RC Features

- Solid development support in JDeveloper
- Changeable “skins”
  - Common look-and-feel characteristics
  - Skin editor in JDev 11.1.2 and beyond
- Layout management features
- Extensive set of properties
  - Declarative access to application metadata
  - Properties can reference dynamic values using Expression Language
- Template support



quovera

48

# Some Components

\* ID

**af:inputText**

Department

**af:inputListOfValues**

Help ▾  
Using the application  
About TUHRA

**af:commandMenuItem**

\* Hire Date

1999

SUN	MON	TUE	WED	THU	FRI	SAT
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3

**af:inputDate**


**af:selectBooleanCheckbox**

\* Job

**af:link (with icon)**

President  
Administration Vice President  
Administration Assistant  
Purchasing Clerk  
Stock Manager  
Stock Clerk  
Shipping Clerk  
Programmer  
Marketing Manager  
Marketing Representative  
Human Resources Representative  
Public Relations Representative  
<Select A Job>

**af:selectOneChoice**



# Component Categories

- Atomic
  - User-facing components
    - af:inputText
    - af:selectOneChoice
  - Operation components
    - Action-oriented components
    - af:showPopupBehavior
- Layout
  - Organize the atomic components
  - Offer various *facets*
  - Normal prefix is “panel” and suffix is “Layout”
    - e.g., af:panelFormLayout
- Data Visualization Tools
  - Graphs, Maps, Hierarchy Viewers, etc.
  - dvt prefix, for example, dvt:graph



# Component Properties

- All components have properties
  - id, binding, rendered
  - accessKey, disabled, readOnly
- Some are component-specific
  - hAlign, columns, rows
  - title, selected
- Example:
 

```
<af:commandButton text="No"
            action="GlobalHome" />
```
- Available in the Property Inspector



# Deep Dive Into af:table

With af:panelCollection

First/Given Name	Last Name	JobTitle	Contact Information		Department Name
			Email Address	Phone No.	
Ellen	Abels	Sales Manager	EABEL	011.44.1644.429267	Sales
Mozhe	Atkinson	Stock Clerk	MATKINSO	650.124.6234	Shipping
David	Austin	Programmer	DAUSTIN	590.423.4569	IT
Elizabeth	Bates	Sales Representative	EBATES	011.44.1343.529268	Sales
David	Bernstein	Sales Representative	DBERNSTE	011.44.1344.345268	Sales
Laura	Bissot	Stock Clerk	LBISSOT	650.124.5234	Shipping
Karen	Colmenares	Purchasing Clerk	KCOLMENA	515.127.4566	Purchasing
Curtis	Davies	Stock Clerk	CDAVIES	650.121.2994	Shipping
Bruce	Ernst	Programmer	BERNSTE	590.423.4568	IT
Timothy	Gates	Shipping Clerk	TGATES	650.505.3876	Shipping
Michael	Hartstein	Marketing Manager	MHARTSTE	515.123.5555	Marketing
Shelley	Higgins	Accounting Manager	SHIGGINS	515.123.8080	Accounting
Charles	Johnson	Sales Representative	CJOHNSON	011.44.1644.429262	Sales
Vance	Jones	Shipping Clerk	VJONES	650.501.4876	Shipping
Jack	Livingston	Sales Representative	JLIVINGS	011.44.1644.429264	Sales
Mattea	Marvins	Sales Representative	MMARVINS	011.44.1346.329268	Sales
Randall	Matos	Stock Clerk	RMATOS	650.121.2874	Shipping

- Column grouping
- Row highlighting
- Column moving
- Export to Excel
- Detach option
- Column sorting & filtering

# Some Interactivity Features

- Appear mainly in the user interface components (such as text fields)

*ConvertNumber sub-component*  
 Pattern = "###,###.##"

- Examples:

- Converters; e.g., number format

- Validators; some are built in, e.g., required

- Messages appear next to items

*Required = true*  
*RequiredMessageDetail = "You must enter a value for {0}."*

# AJAX in ADF Faces RC

- Asynchronous JavaScript and XML
- **Partial Page Rendering (PPR)** in ADF Faces
  - “Declarative AJAX”
- Much AJAX in ADF Faces is transparent
  - Built into the components
  - Nothing special needs to be done
- You can setup non-default AJAX behavior using properties
  - *partialSubmit* – used by command items
  - *autoSubmit* – used by input items/lists, etc.
  - *partialTriggers* – all components, sets up the “viewer” (listener)



AJAX provides a cleaner user interface!



# AJAX Interactions - Total Pay

Recalculate

Id	Raise
Value	#{bindings.Raise.inputValue}
AutoSubmit	true

Id	TotalPay
Value	#{bindings.Salary.inputValue + bindings.Raise.inputValue}
AutoSubmit	false
partialTriggers	Salary Raise

Id	Salary
Value	#{bindings.Salary.inputValue}
AutoSubmit	true

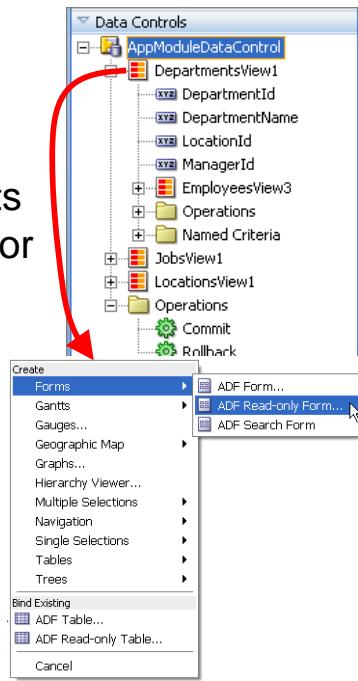
# Agenda

- Intros: ADF, JDev, sample app
- ADF Business Components
- ADF Faces
- ADF Model and ADF Controller



# ADF Model

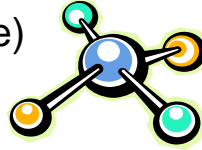
- ADF Data Controls
  - Provides list of components or groups of components for a node in the data model
  - “Drop as” options
- ADF Bindings
  - Prebuilt connection from the ADF BC to the UI
  - Drag and drop action above does the work



# ADF Data Controls

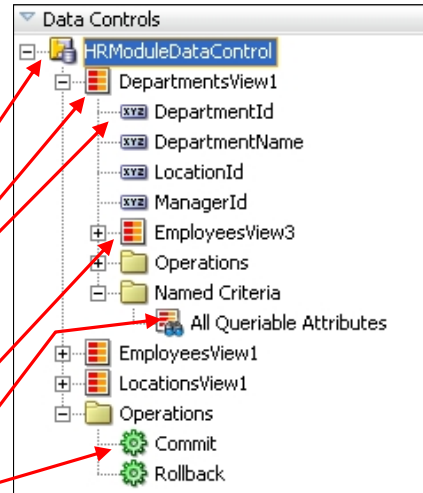
- Business Services abstraction
  - Makes Model components available to ViewController
  - Automatically created with ADF BC
  - Can be created for other business services
  - For non-ADF BC, defined in DataControls.cpx
- Provide list of “Drop as” options that create pre-bound components
  - Collection level (view object instance)
  - Attribute level (view attribute)

The Good News:  
You don't normally write data controls



# Data Controls in JDeveloper

- Exposed in the Data Controls panel (DCP)
  - Automatically appears when editing a JSF JSP
  - OR CTRL-SHIFT-D
- Nodes for
  - Data control
  - Data collection
  - Attribute
  - Nested data collection
  - Named Criteria
  - Operation

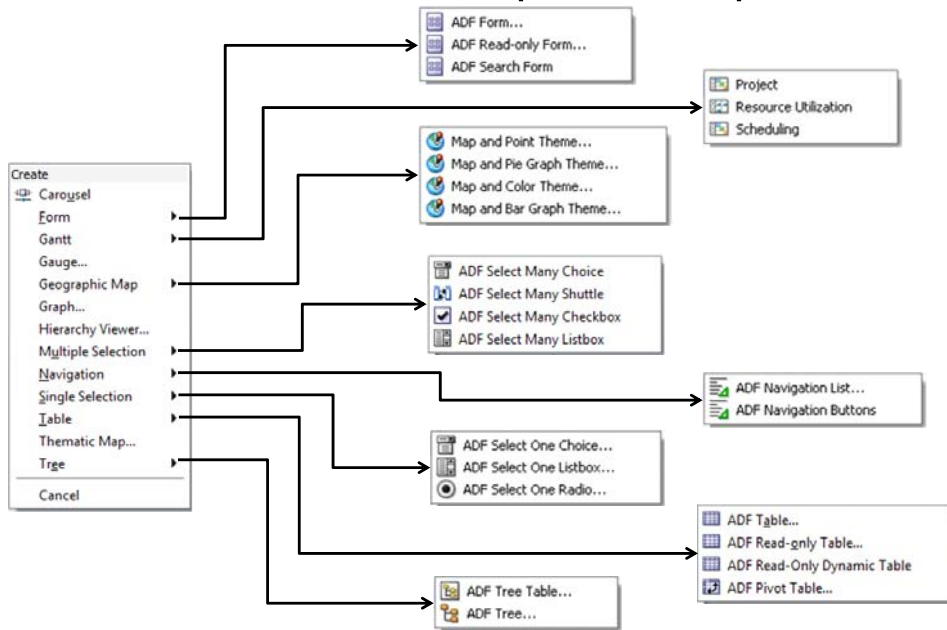


# Adding Bound Components

Type, paste, or page, or drag at

1. Drag-and-drop from the Data Controls panel
2. Select a “drop as” option
3. Bound components will be added to the page

# Collection "Drop As" Options

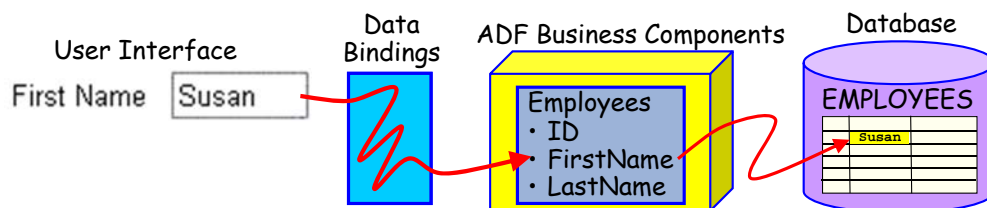


# Drop As Examples: Form and Table

Note the binding expressions

## ADF Bindings

- Association of a business service data element or action with a UI component
  - Relatively automatic in Oracle Forms
  - Definitely not automatic in native Java EE
- Binding normally takes a lot of coding
  - One-off solution is not the answer
  - Need a framework to assist



## Common Binding Types

- Attribute
  - For single attribute in a collection
- List
  - For data-bound list elements
- List of Values
  - For view object lists of values
- Tree
  - Hierarchical controls (master detail), tables, list views
- Table (or range)
  - For table components bound to collections (not used much now)
- Action
  - For standard operations like Commit
- Method
  - For custom methods
- Boolean
  - For checkboxes

"Drop As" options add these to the page automatically





## Attribute Bindings and JSF

- You can access attribute bindings from any component attribute

```
<af:outputText value="#{bindings.DepartmentId.inputValue}" />
```

- Other attributes access properties on the Model level object
  - Control hints or attribute properties (for example, label and width):

```
<af:inputText  
value="#{bindings.PhoneNumber.inputValue}"  
label="#{bindings.PhoneNumber.label}"  
columns="#{bindings.PhoneNumber.displayWidth}" />
```



## Binding File

- The *page definition* (or *PageDef*) XML file stores binding definitions for the page
  - One PageDef file for each JSP
  - Called *filenamePageDef*
  - For example; editEmployeePageDef.xml
- Created when you drag a data control to the page the first time
  - You can create the file manually
- Maintained as you drag or delete components
  - You can edit manually



## Binding Code

### In the JSF page file

```
<af:inputText value="#{bindings.DepartmentId.inputValue}"  
label="#{bindings.DepartmentId.hints.label}"  
required="#{bindings.DepartmentId.hints.mandatory}"  
columns="#{bindings.DepartmentId.hints.displayWidth}"  
maxLength="#{bindings.DepartmentId.hints.precision}"  
shortDesc="#{bindings.DepartmentId.hints.tooltip}"  
id="it1">  
</af:inputText>
```

### In the bindings PageDef file

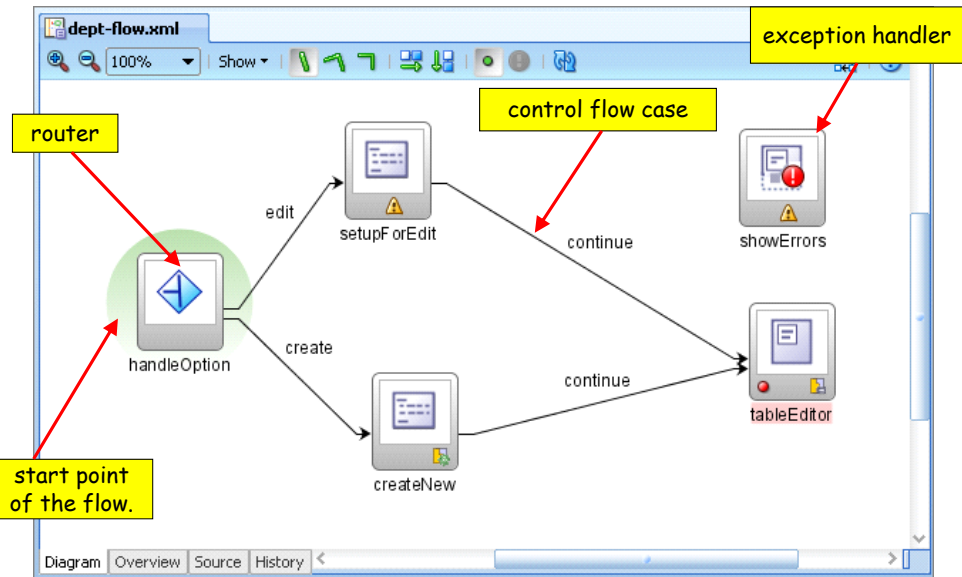
```
<bindings>  
<attributeValues IterBinding="DepartmentsView1Iterator"  
id="DepartmentId">  
<AttrNames>  
<Item Value="DepartmentId" />  
</AttrNames>  
</attributeValues>
```

## ADF Controller

- Extension to standard JSF Controller functionality
- Defines *task flows*
  - Logic and page fragment components
  - Embedded on the page in a region component
- Benefits
  - Page fragment re-use
  - Executing code in a logic-defined flow
    - "Task flow" not "page flow"
  - Security
  - Exception handling and transaction management
- Defined in a diagram
  - Like JSF but more components available



# Sample ADF Controller Development

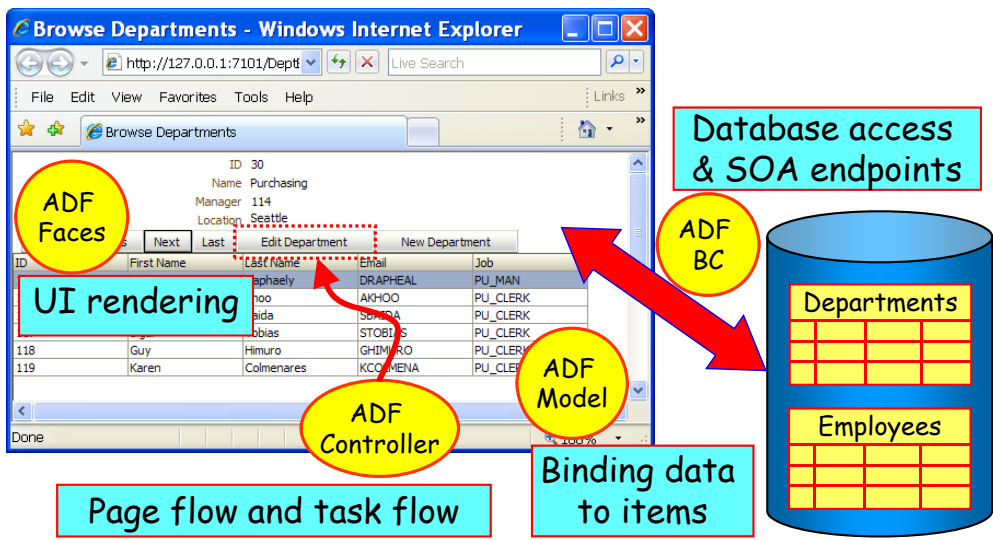


# Sample ADF Controller Code

```

<task-flow-definition id="dept-flow">
<default-activity>deptBrowse</default-activity>
<view id="deptBrowse">
  <page>/deptBrowse.jsp</page>
</view>
<view id="deptEdit">
  <page>/deptEdit.jsp</page>
</view>
<control-flow-rule>
  <from-activity-id>deptBrowse</from-activity-id>
  <control-flow-case>
    <from-outcome>toEdit</from-outcome>
    <to-activity-id>deptEdit</to-activity-id>
  </control-flow-case>
</control-flow-rule>
<router id="checkForExplicitID">
  <case id="_6">
    <expression>#{!empty pageFlowScope.employeeId}
    </expression>
    <outcome>byId</outcome>
  </case>
  <default-outcome>currentUser</default-outcome>
</router>
<method-call id="queryEmployeeById">
  <method>#{bindings.queryEmployeeById.execute}</method>
  <outcome>
    <fixed-outcome>queryEmployeeById</fixed-outcome>
  </outcome>
</method-call>
  
```

# Summary: ADF Core Technologies



# Final Voltaire Wisdom

The secret of being a bore is to tell everything.

---

Le secret d'ennuyer est celui de tout dire.

—Voltaire (1694-1778), *Sept Discours en Vers sur l'Homme*

# Summary

- Oracle is building Fusion Applications with an ADF core technology stack
- ADF offers a consistent developer experience regardless of the technologies
- ADF Business Components provide access to the database and other data sources
- ADF Model connects ADF BC to ADF Faces
- ADF Faces provide 150+, feature-rich item and container components for JSF pages
- ADF Controller manages page flow and task flow



- Books co-authored with Dr. Paul Dorsey, Avrom Roy-Faderman, & Duncan Mills



[www.quovera.com](http://www.quovera.com)

- Founded in 1995 as Millennia Vision Corp.
- Profitable without outside funding
- Consultants each have 10+ years industry experience
- Strong High-Tech industry background
- 200+ clients/300+ projects
- JDeveloper Partner
- More technical white papers and presentations on the web site