

The Tie That Binds: An Introduction to ADF Bindings

Peter Koletzke

Technical Director &
Principal Instructor



quovera



Bindings Are Like This



quovera

2

Survey

- “Traditional” Oracle development (Forms, Reports, Designer, PL/SQL)
 - 1-2 years?
 - More than 2 years?
- Java development
 - 1-3 years?
 - 4-17 years?
 - 17+ years?
- JDeveloper
 - 1-2 years?
 - 2+ years?



quovera

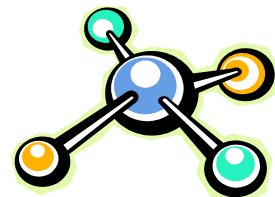
3

Agenda

- Overview
- Working with Bindings
- Binding Examples

Slides will be available on the Quovera and UTOUG websites.

Additional information in the appendixes.

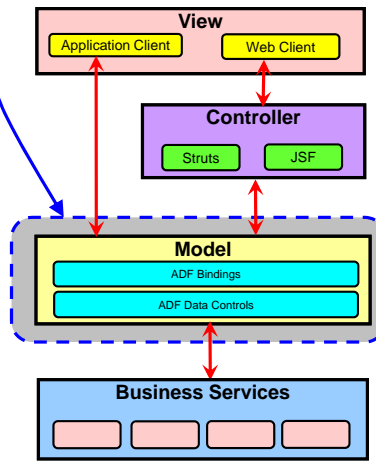


quovera

4

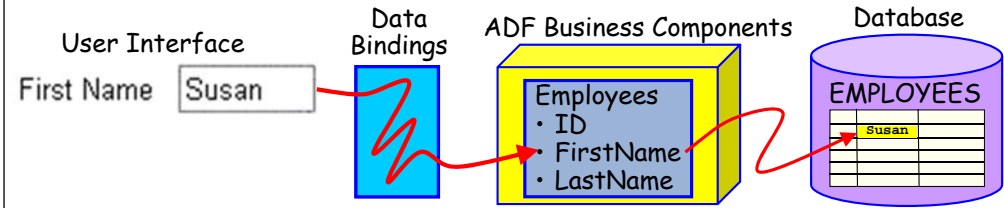
Review: ADF Model (ADFm)

- A layer of ADF
 - ADF Bindings
 - Bindings provide objects to link to components
 - ADF Data Controls
 - Automatically bound sets of components
- Communication from Business Services to View and Controller layers
 - One common layer for all types of business services
 - E.g., EJB, ADF BC, web services
 - Code to access any business service works the same way



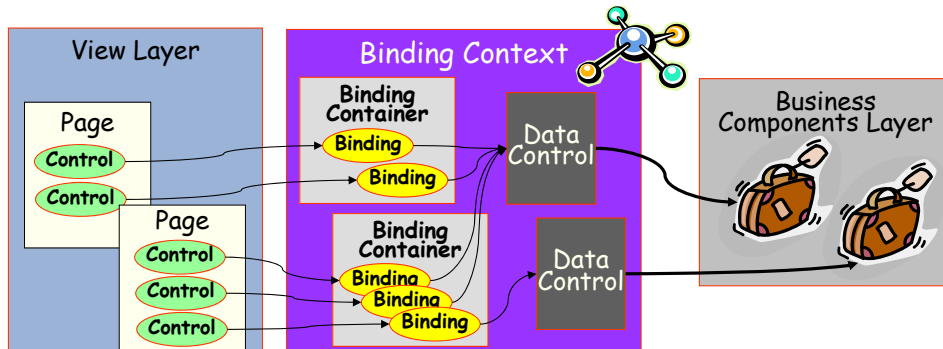
ADF Bindings

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



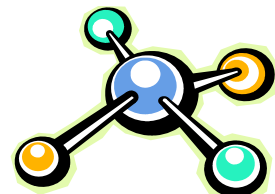
ADF Model Components

- Each control on the page can use bindings
- Bindings expose actions to buttons and links
 - Commit, Rollback, First, Last
- Bindings expose data to input items
 - FirstName, LastName, Email, HireDate



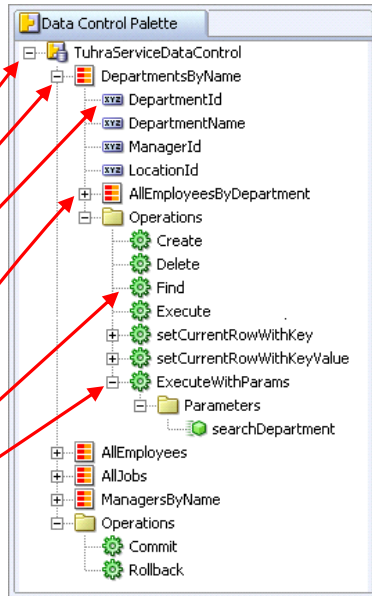
Agenda

- Overview
- Working with Bindings
 - Creating/Accessing PageDef file
 - DataBindings.cpx
- Binding Examples



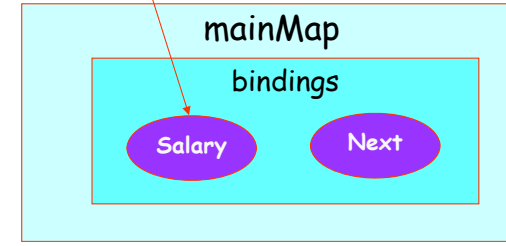
Creating Bindings Automatically

- Drag and drop control from the Data Control panel
 - Automatically appears when editing a JSF JSP
 - OR Ctrl-Shift-D
 - This creates and binds UI items
- Nodes for
 - Data control
 - Data collection
 - Attribute
 - Nested data collection
 - Operation
 - Method



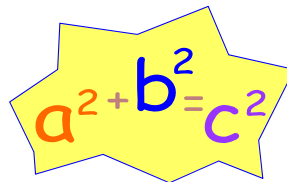
Accessing Bindings Manually

- Programmatically, you use `java.util.Map`
 - An interface for organizing data
 - Stores *elements* – data of any Object type
- The **bindings** map contains all the bindings in the current page's binding container
- You can access these bindings using Expression Language `#{bindings.Salary}`



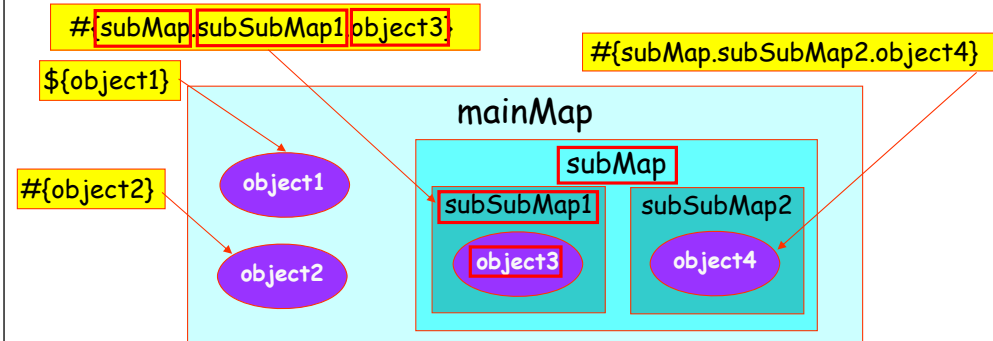
Expression Language

- A.k.a.:
 - “JSP Expression Language”
 - “EL”
- Part of JavaServer Pages Standard Tag Language (JSTL)
 - Procedural language within tags
 - `forEach`; `if`; `choose`; `set`; `when`
- Many other technologies can use it
 - JSF, UIX, Struts, Swing
- Can be used to refer to elements stored in maps
 - Collections of objects



EL Syntax

- All EL expressions have the form `${...}` or `#{...}`
 - JSF uses the # variation for component properties
- Refer to map elements by specifying the path to the element within the map, separated by “.”
- The main map is determined by context



EL for ADF

- ADF's submap is "bindings"
- You can use EL expressions that refer to this map in attribute values, e.g.

ID 203

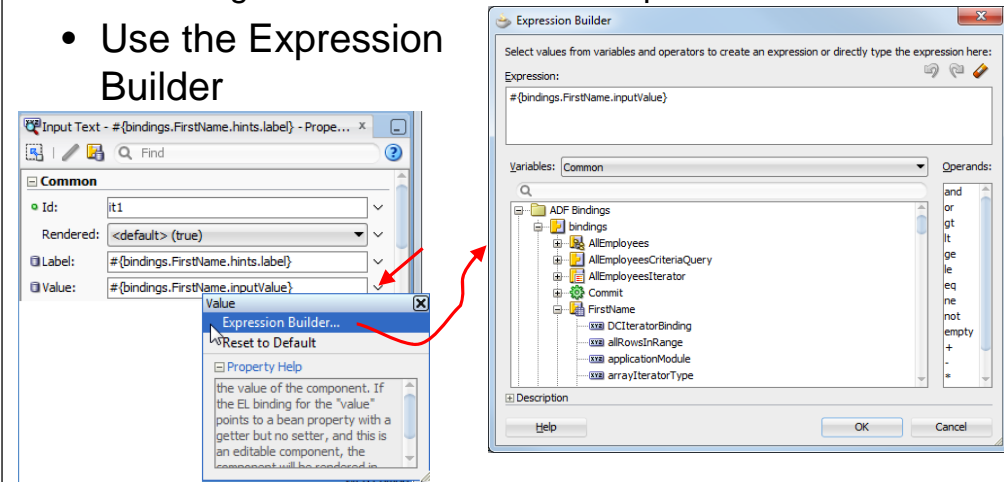
```
<af:inputText  
  value="#{bindings.EmployeeId.inputValue}"  
  label="#{bindings.EmployeeId.label}"/>
```

- The af:inputText label and value are derived from the bindings context
 - EmployeeId – a submap referring to the ADF BC view object instance
 - In Forms, those EL expressions are like this:

```
GET_ITEM_PROPERTY('EMP.EMPLOYEE_ID',DATABASE_VALUE);  
GET_ITEM_PROPERTY('EMP.EMPLOYEE_ID',PROMPT_TEXT);
```

Creating the Expression

- JSF expressions use the "#" prefix
 - Distinguishes them from JSTL expressions
- Use the Expression Builder



Programmatic Access to Bindings

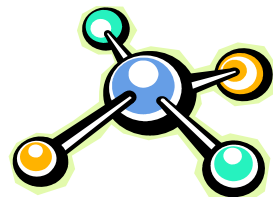
- Double click a bound button to generate the backing bean binding code, like this:

```
public String saveAction()  
{  
  BindingContainer bindings = getBindings();  
  OperationBinding operationBinding =  
    bindings.getOperationBinding("Commit");  
  Object result = operationBinding.execute();  
  if (!operationBinding.getErrors().isEmpty())  
  {  
    return null;  
  }  
  return null;  
}
```

Agenda

- Overview
- Working with Bindings
- Binding Examples

Creating/Accessing
PageDef file
DataBindings.cpx



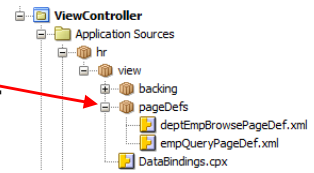
The Bindings File

- 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



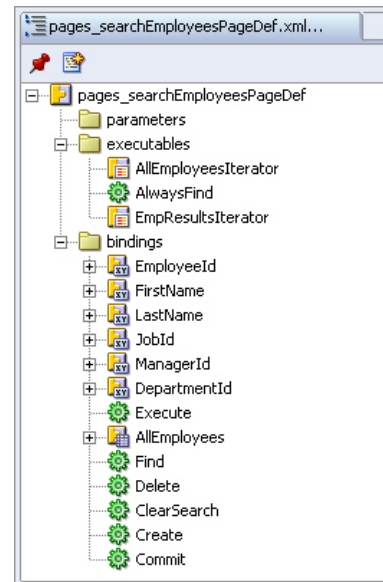
Viewing/Editing the PageDef

- Navigator
- **OR** Bindings tab on the JSF page



Contents of the PageDef File

- Executables
 - Definitions of actions that will be run when the PageDef file is loaded
 - Like trigger code in Forms
- Bindings
 - Values and operations required on the page



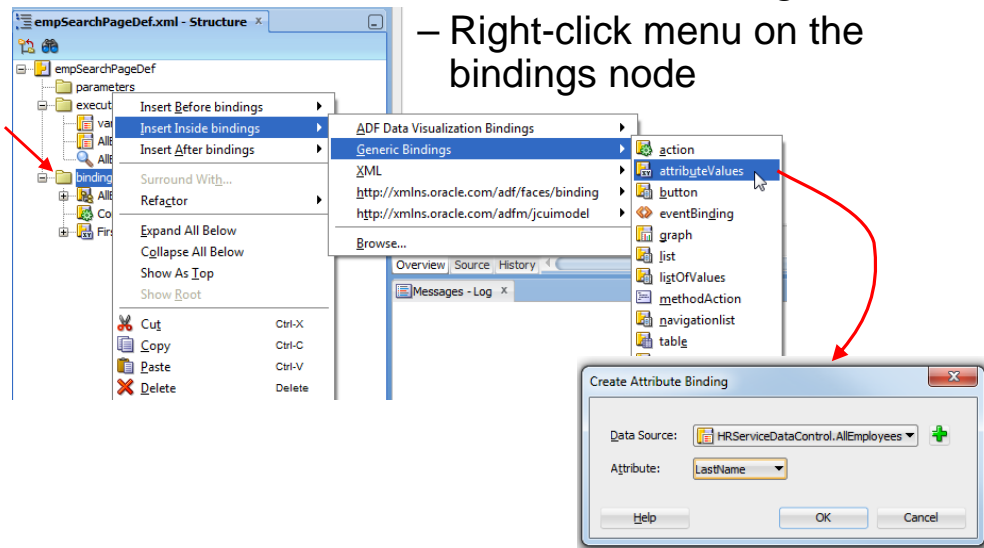
PageDef Snippet: Bindings

```
<pageDefinition xmlns=http://xmlns.oracle.com/adfm/uimodel ...
<bindings>
  <attributeValues id="EmployeeId" IterBinding="AllEmployeesIterator">
    <AttrNames>
      <Item Value="EmployeeId"/>
    </AttrNames>
  </attributeValues>
  <action id="Commit" InstanceName="TuhraServiceDataControl"
    DataControl="TuhraServiceDataControl"
    RequiresUpdateModel="true" Action="100"/>
  <list id="AllEmployeesJobId" IterBinding="AllEmployeesIterator"
    StaticList="false" ListOperMode="0" ListIter="AllJobsIterator"
    NullValueFlag="1" NullValueId="AllEmployeesJobId_null">
    <AttrNames>
      <Item Value="JobId"/>
    </AttrNames>
    <ListAttrNames>
      <Item Value="JobId"/>
    </ListAttrNames>
    <ListDisplayAttrNames>
      <Item Value="JobTitle"/>
    </ListDisplayAttrNames>
  </list>
</bindings>
```

Creating Bindings–Alternative 1

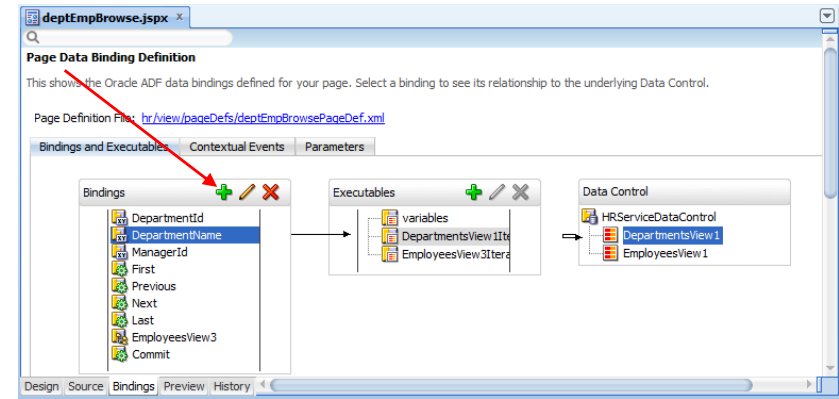
- Use the Structure window for the PageDef file

– Right-click menu on the bindings node



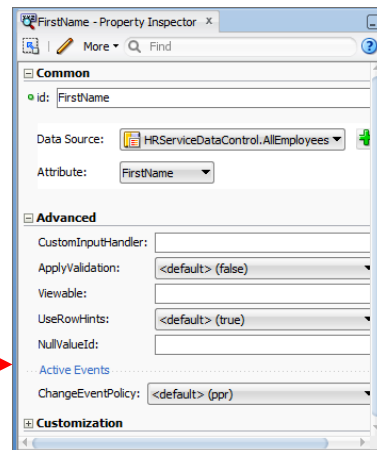
Creating Bindings–Alternative 2

- Bindings tab on the JSPX page
- Or Overview tab in the PageDef file



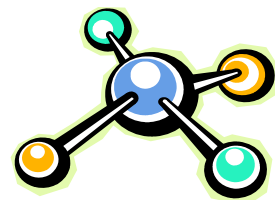
Editing Bindings–the Alternatives

- PageDef (Overview tab) or page (Bindings tab)
- Code Editor for the PageDef file
- Or select **Go to Binding** from a UI component
- Or select **Go to Properties** from the right-click menu on a binding in the Structure window
- Or select the binding in the Structure window and use the Property Inspector

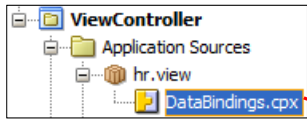


Agenda

- Overview
 - Working with Bindings
 - Binding Examples
- Creating/Accessing PageDef file
DataBindings.cpx



DataBindings.cpx



Data Binding Registry

This file defines the Oracle ADF binding context for your application. JDeveloper creates this file the first time you data bind a UI component.

Page Mappings	
path	usageId
/deptEmpBrowse.ispx	hr_view_deptEmpBrowsePageDef
/empQuery.ispx	hr_view_empQueryPageDef

Page Definition Usages	
id	path
hr_view_empQueryPageDef	hr_view_pageDefs.empQueryPageDef
hr_view_deptEmpBrowsePageDef	hr_view_pageDefs.deptEmpBrowsePageDef

Data Control Usages	
id	
HRServiceDataControl	

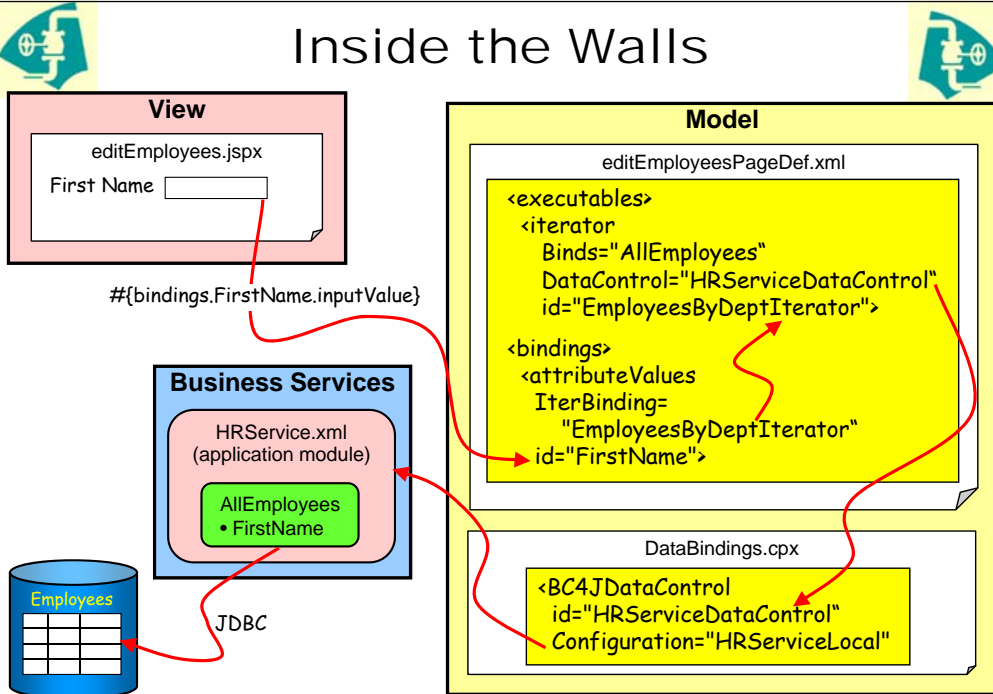
- PageDef file mappings
- PageDef locations
- Data Control usages

DataBindings.cpx DC Usages

```
<dataControlUsages>
  <BC4JDataControl id="HRServiceDataControl"
    Package="hr.model"
    FactoryClass="oracle.adf.model.bc4j.DataControlFactoryImpl"
    SupportsTransactions="true"
    SupportsFindMode="true"
    SupportsRangesize="true"
    SupportsResetState="true"
    SupportsSortCollection="true"
    Configuration="HRServiceLocal"
    syncMode="Immediate"
    xmlns="http://xmlns.oracle.com/adfm/datacontrol"/>
</dataControlUsages>
```

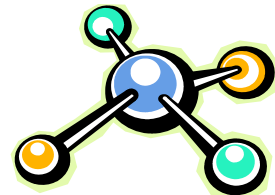
- Last section in overview editor
- ADF BC data control (BC4J)
- ID for data control used

Inside the Walls



Agenda

- Overview
- Working with Bindings
- Binding Examples



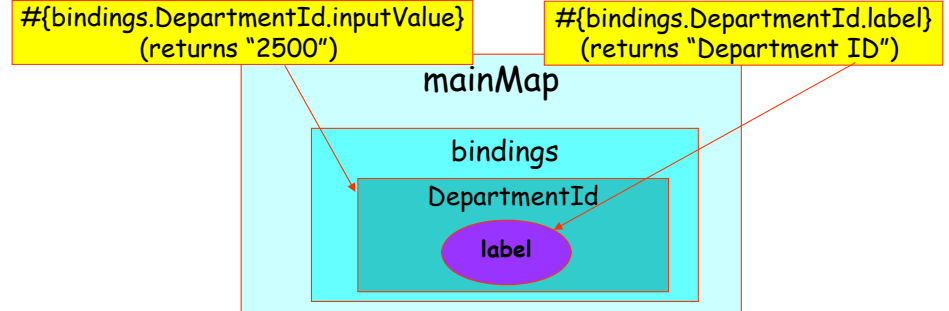
Common Binding Types

- **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) and tables
- Table (or range)
 - For table components bound to collections (not used much now)
- Boolean
 - For checkboxes
- Action
 - For standard operations like Commit
- Method
 - For custom methods



Attribute Bindings

- A single view attribute value on the iterator binding's current view row
- Example: DepartmentId



Attribute Bindings and JSF

- You can access attribute bindings from any component attribute

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

- Other component 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}"/>
```

Code Snippets

JSF

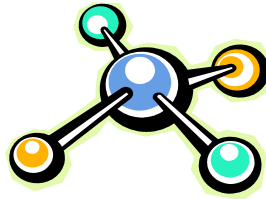
```
<af:inputText value="#{bindings.DepartmentId.inputValue}" label="#{bindings.DepartmentId.hints.label}" required="#{bindings.DepartmentId.hints.mandatory}" columns="#{bindings.DepartmentId.hints.displayWidth}" maxLength="#{bindings.DepartmentId.hints.precision}" shortDesc="#{bindings.DepartmentId.hints.tooltip}"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

PageDef

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

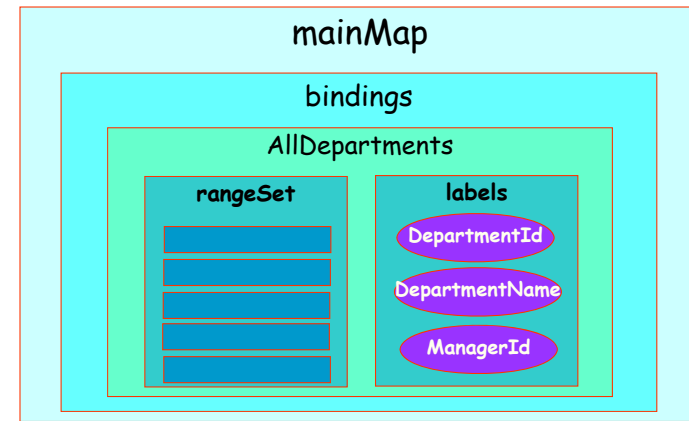

Tree Bindings

- Used for tables, tree-tables and trees
- Expose data from all rows in the iterator binding's current range
- Can also be used for master-detail
- Expose some or all columns
- Iterator's *collectionModel* property is used as the table's value
- A variable is assigned to the table binding's data
 - Usually "row"



Tree Binding Map

- Example: AllDepartments
 - Contains rangeSet and labels



PageDef Snippets

```
<bindings>
  <table
    id="AllEmployees"
    IterBinding=
      "EmpResultsIterator">
    <AttrNames>
      <Item Value="EmployeeId"/>
      <Item Value="FirstName"/>
      <Item Value="LastName"/>
      <Item Value="Email"/>
      <Item Value="PhoneNumber"/>
      <Item Value="HireDate"/>
      <Item Value="JobId"/>
      <Item Value="Salary"/>
      <Item Value="ManagerId"/>
    </AttrNames>
  </table>
```

Table Binding

```
<executables>
  <iterator
    id="EmpResultsIterator"
    RangeSize="10"
    Binds="AllEmployees"
    DataControl=
      "TuhraServiceDataControl"/>
  <iterator
    id="AllJobsIterator"
    RangeSize="-1"
    Binds="AllJobs"
    DataControl=
      "TuhraServiceDataControl"/>
</executables>
```

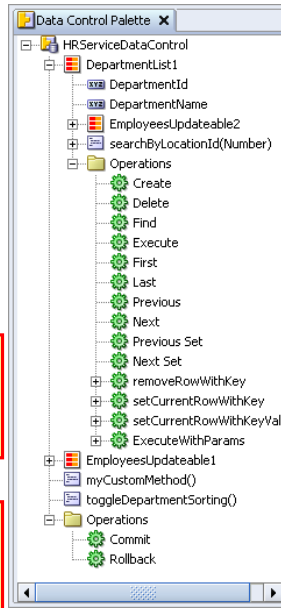
Iterator

JSF Snippet: af:table

```
<af:table
  value="#{bindings.AllEmployees.collectionModel}" var="row"
  rows="#{bindings.AllEmployees.rangeSize}"
  first="#{bindings.AllEmployees.rangeStart}"
  selectionState=
    "#{bindings.AllEmployees.collectionModel.selectedRow}"
  selectionListener=
    "#{bindings.AllEmployees.collectionModel.makeCurrent}"
  width="100%">
  <af:column
    headerText="#{bindings.AllEmployees.labels.EmployeeId}"
    sortProperty="EmployeeId" sortable="true">
    <af:outputText value="#{row.EmployeeId}"/>
  </af:column>
  <af:column
    headerText="#{bindings.AllEmployees.labels.FirstName}"
    sortProperty="FirstName" sortable="true">
    <af:outputText value="#{row.FirstName}"/>
  </af:column>
```

Action Bindings

- Added when buttons or links are dropped into the UI
- Run *operations*
 - Collection operations
 - Create, Delete, Find, First, Last, etc.
 - Data control operations
 - Commit, Rollback



JSF

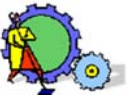
```
<af:commandButton
  text="Save"
  actionListener="#{bindings.Commit.execute}"
  disabled="#{!bindings.Commit.enabled}"/>
```

PageDef

```
<action id="Commit" RequiresUpdateModel="true"
  Action="commitTransaction"
  DataControl="HRServiceDataControl"/>
```

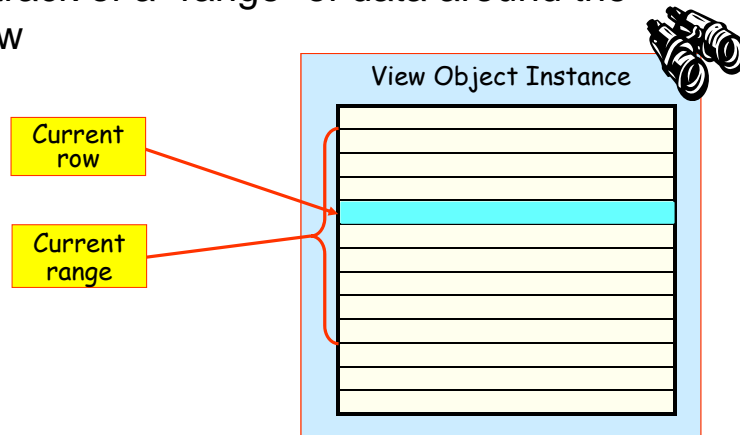
What is an Iterator?

- A pointer to a row in a collection
 - Allows manipulation of attribute values in a row
- Used to navigate a set of rows
 - Much like a cursor navigating a query in PL/SQL
- You implicitly refer to these in binding expressions:
 - `#{bindings.EmployeeId.inputValue}`
 - “The employee ID of whatever row is currently pointed to by the iterator”
- Can define more than one iterator per collection
 - To have different pointers to the same collection
 - E.g., if the same collection were used for two purposes like a search form and a read-only results table



Iterator – An Executable

- These keep track of the current view row in a view object instance’s query result
- Keeps track of a “range” of data around the view row



I'd Hammer in the Morning

All parts should go together
without forcing...
By all means,
do not use a hammer.

— 1925 IBM Maintenance Manual



Summary

- ADF Model layer connects the ADF Controller or View and Business Services layers
- ADF Data Bindings provide association of UI components and business services
- Simple coding in EL (and XML) tap into the ADF Model framework code
- XML code in the PageDef file defines the bindings
- EL in the UI properties access the binding by name
- Different types of bindings accomplish different tasks



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



www.quovera.com

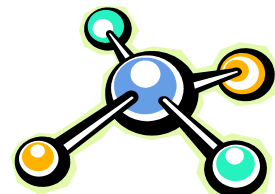
- Founded in 1995 as Millennia Vision Corp.
- Profitable without outside funding
- Consultants each have 10+ years industry experience
- Strong High-Tech industry background
- 200+ clients/300+ projects
- JDeveloper Partner
- More technical white papers and presentations on the web site

Appendixes

- A: More Binding Examples
- B: More About Iterators and Executables

Table Bindings

- A.k.a., *range bindings*
- Expose data from all rows in the iterator binding's current range
- Expose some or all columns
- Iterator's *collectionModel* property is used as the table's value
- A variable in the table UI component is assigned to the table binding's data
 - Usually "row"



List Bindings

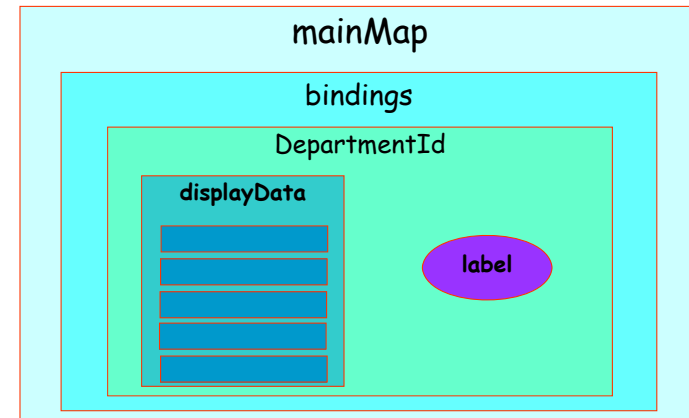
- Populate a single attribute in the current row, or navigate between rows
 - Dynamic or static
- Generally used as the **value** attribute for a pulldown list or similar element

```
<af:selectOneChoice  
  value="#{bindings.AllEmployeesJobId.inputValue}"  
>
```



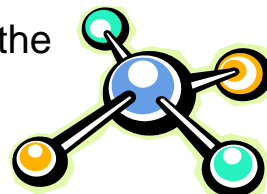
List Bindings and Maps

- Example: a list for DepartmentId



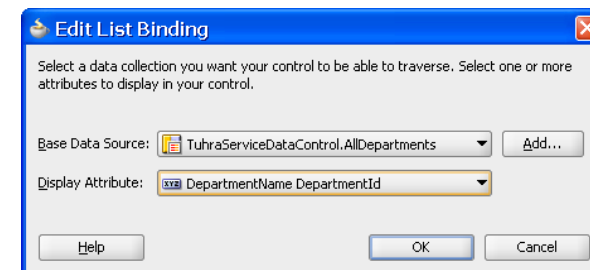
List Binding Modes

- For populating an attribute, a list binding can be in static or dynamic list
 - Static allows you to hard code the list
 - Dynamic lets you specify a collection as the source
- For navigating, a list binding uses Navigation mode
 - The iterator binding will jump to the row selected in the list



Navigation List Bindings

- Changes the current row in the iterator
- Selecting a value from a navigation list sets that row as current
- This List Binding Editor appears when you drop a collection as a navigation list



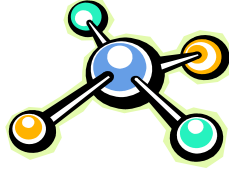
List of Values Bindings

- Used for ADF LOV input list or ADF LOV choice list
 - Popup LOV dialog
- List elements are displayed from the view object
 - Any number of columns
- Provides search capabilities
 - More functionality than a standard list binding
 - Can also display a *quick selection list*
 - Values displayed in a pulldown not popup



Method Bindings

- Used for custom code
 - Otherwise, identical in results to action bindings
- Drop the method from the Data Controls panel as a parameter form or button or link
 - Creates a methodAction binding type
- Can also create the binding manually

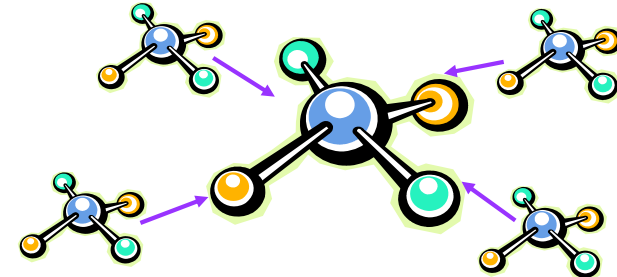


Appendixes

- A: More Binding Examples
- B: More About Iterators and Executables

Iterators Are Necessary

- Almost all other kinds of bindings must be associated with an iterator
- Exceptions: are bindings to actions
 - E.g., “Commit” and “Rollback”,
 - Do not apply to a particular query result



Iterator Details

- Consider this binding:
`#{bindings.EmployeeId.inputValue}`
 - The employee ID of whatever record is currently pointed to by the iterator
- Iterators are defined in the PageDef file executables section
 - The query associated with the iterator's view object is run when the page loads
 - If more than one iterator per collection, the query executes once
- Important property: *rangeSize*
 - Number of rows fetched (-1 is all)



InvokeAction - Another Executable

- Calls an existing action or method binding defined in action bindings
- Example: Create binding will add a row to the collection
 - invokeAction on the binding will execute this operation
- Important property: *Refresh*
 - Specifies when this invokeAction occurs
 - ifNeeded – framework decides
 - always – every time the page is invoked



Action and ActionListener Properties

- Used on action components
 - af:commandButton, af:commandLink
- An ActionListener executes before the main action of the component, e.g.,

```
<af:commandButton  
  actionListener="#{bindings.Rollback.execute}"  
  text="Cancel"  
  action="cancelEditEmp">
```

- The main action returns "cancelEditEmp" to the framework to cause page navigation
- The ActionListener executes a Rollback operation before the main action
 - Two actions, one component
 - No Java!

