

**ORACLE®**




ORACLE  
**OPEN**  
WORLD

# Your. Open. World.

## ADF Bindings – Understand what you are Building

Duncan Mills  
Sr. Director of Product Management

ORACLE®



The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Questions You Should Have Asked Already

- How is the Data Control Palette populated?
- What controls the options offered when I drag and drop?
- What actually happens when I drag and drop onto a page?
- Can I do more?
- What's different in ADF 11?

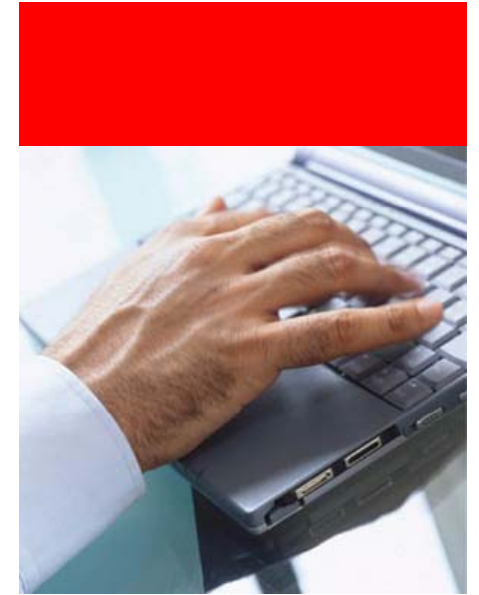
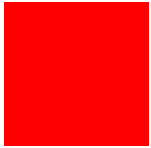




# The Data Control Palette

## Where Does That Come From?

- We know it describes services
  - Attributes
  - Collections
  - Methods
  - Operations
- Description for ADF BC in the workspace is automatic
  - no extra metadata required
- Other Service types
  - Explicit “Create Data Control” step
  - Describes the service factory and object types



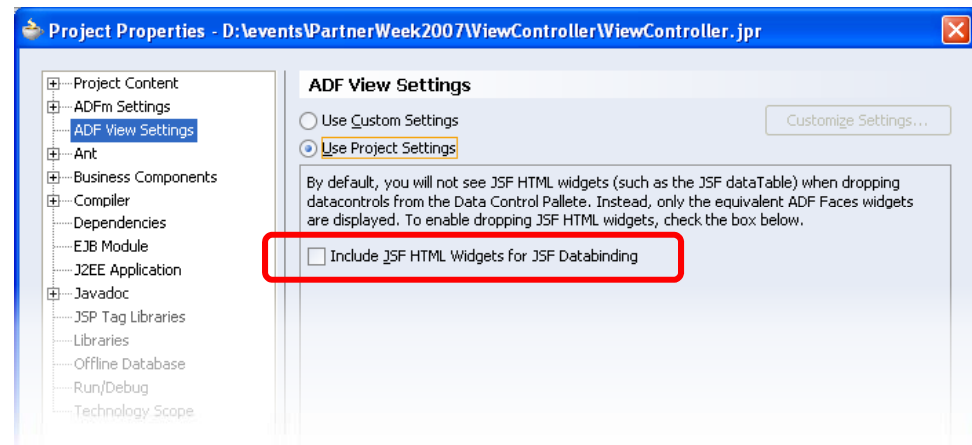
# Demonstration

---

## Data Control Artifacts

# Drag and Drop Options

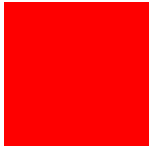
- Controlled by several factors
  - Page technology
  - The project property “ADF View Settings”
  - Templates
- Customize the markup created on drop
- Or, add your own...



# Custom Templates

- /jdev/system/oracle.adfm.dt.faces.10.1.3.(n)
- For JSF: **faces\_creator\_configuration.xml**

```
<creator
  name="outputlabel_adf_faces"
  bindingsTypes="primitiveText"
  rank=".45f"
  namespace="http://xmlns.oracle.com/adf/faces"
  representativeNamespace="http://java.sun.com/jsf/html"
  localName="outputLabel"
  binderClass="oracle.adfdtinternal.view.faces.binding.binder.LabelBinder"
  properties="readOnly rebindOnly">
  <![CDATA[
    <af:outputLabel
      #if ($properties.get("primitive:suggestedLabel"))
        value="{properties.get("primitive:suggestedLabel")}"
      #else
        value="{model.getLabelExpression()}"
      #end
    />
  ]]>
</creator>
```

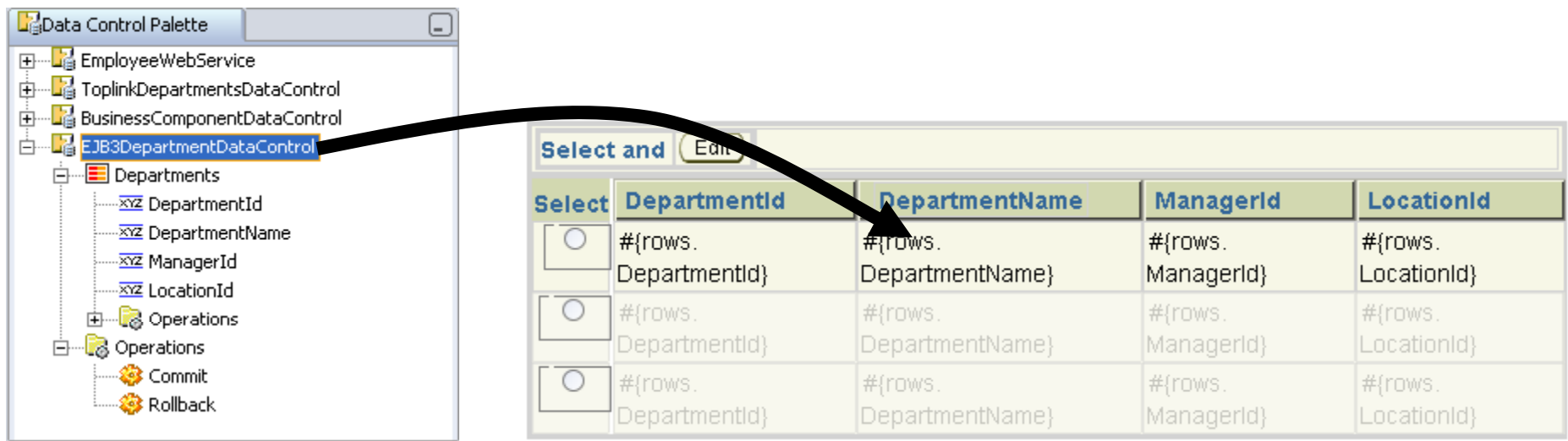


# Voyage into the PageDef



# So What Exactly is Going On?

- This is what we do...

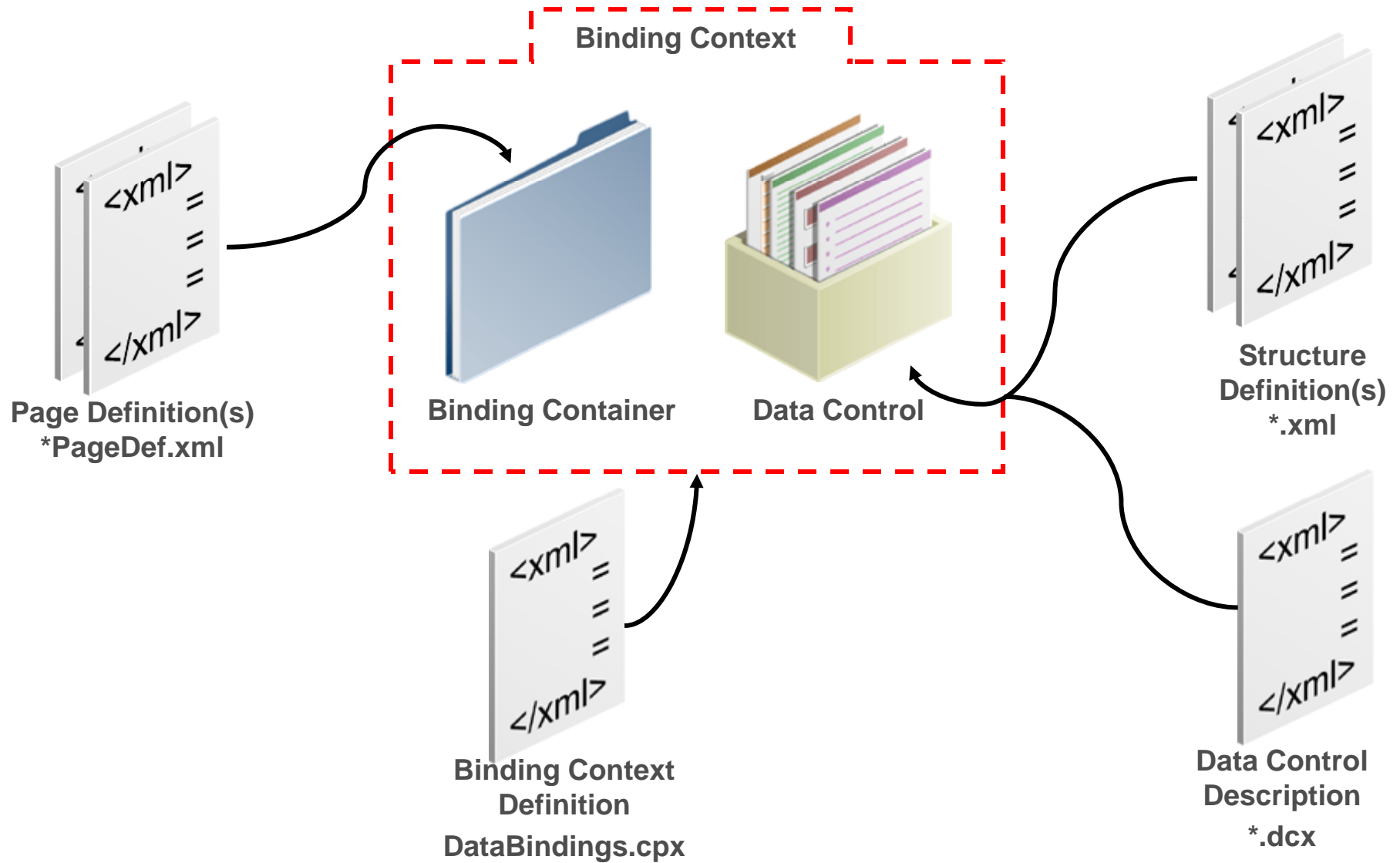


The screenshot illustrates the configuration of a data control. On the left, the 'Data Control Palette' shows a tree view with 'EJB3DepartmentDataControl' selected, containing 'Departments' with fields 'DepartmentId', 'DepartmentName', 'ManagerId', and 'LocationId'. An arrow points from 'DepartmentName' in the palette to the 'DepartmentName' column in the table editor on the right. The table editor shows a table with columns 'DepartmentId', 'DepartmentName', 'ManagerId', and 'LocationId', and three rows of data.

Select	DepartmentId	DepartmentName	ManagerId	LocationId
<input type="radio"/>	{rows. DepartmentId}	{rows. DepartmentName}	{rows. ManagerId}	{rows. LocationId}
<input type="radio"/>	{rows. DepartmentId}	{rows. DepartmentName}	{rows. ManagerId}	{rows. LocationId}
<input type="radio"/>	{rows. DepartmentId}	{rows. DepartmentName}	{rows. ManagerId}	{rows. LocationId}

- But what's behind the drag and drop?

# Bindings MetaData Summary




# DataBindings.cpx

- Locates the correct PageDef for a page
- Defines the Data Control usages of the project

```
<?xml version="1.0" encoding="UTF-8" ?>
  <Application xmlns="http://xmlns.oracle.com/adfm/application"
    version="10.1.3.40.33" id="DataBindings" SeparateXMLFiles="false"
    Package="oracle.demo.view" ClientType="Generic">
    <pageMap>
      <page path="/home.jspx" usageId="homePageDef" />
      <page path="..." />
    </pageMap>
    <pageDefinitionUsages>
      <page id="homePageDef" path="oracle.demo.view.pageDefs.homePageDef" />
      <page id="..." />
    </pageDefinitionUsages>
    <dataControlUsages>
      <BC4JDataControl id="AppModuleDataControl" Package="oracle.demo.model"
        FactoryClass="oracle.adf.model.bc4j.DataControlFactoryImpl"
        SupportsTransactions="true" SupportsFindMode="true"
        SupportsRangeSize="true" SupportsResetState="true"
        SupportsSortCollection="true"
        Configuration="AppModuleLocal" syncMode="Immediate"
        xmlns="http://xmlns.oracle.com/adfm/datacontrol" />
    </dataControlUsages>
  </Application>
```

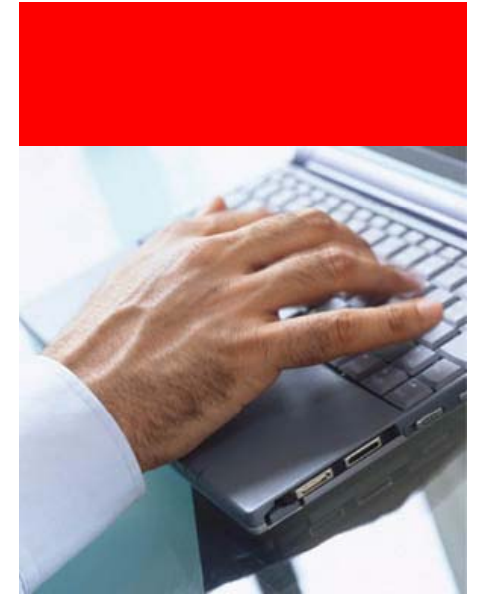
# And So To The PageDef...

- The key data binding artifact
  - Accessible via the VE context menu:  Go to Page Definition
  - Or via the PI quick-link
  - Describes the service usage of a particular page / panel / region
- XML File
  - Default location controlled by project property  
*ADFm Settings* → *PageDef* sub-package
  - Runtime location determined by the DataBindings.cpx file  
(see <context-param> CpxFileName in web.xml)
- Three sections (in order of interest not appearance!):
  1. Bindings
  2. Executables
  3. Parameters



# What You Need to Know About Bindings

- They may or may not be used in a UI
- Accessed via the "bindings" object on the HTTP Request
  - Conventionally via Expression Language (EL)  
`#{bindings.EmpNo}`
  - `#{bindings}` provides access to everything defined in the Binding Container / PageDef
  - Individual bindings e.g. Table Bindings
    - Are objects
    - Have attributes \*other\* than value
    - All accessible via EL
  - `#{bindings.Employees.rangesize}`



# Demonstration

---

**Turning the Tables...**



# Executables

- Iterators
  - Link other bindings to a Data Control (mostly)
    - Either Attributes directly or collections
  - Types:
    - **Iterator** – basic link data to binding
    - **Method iterator** – link the results of a method call to a binding
    - **Accessor Iterator** – link to detail collections for nested objects (not used with ADF BC)
    - **Variable Iterator** – access to local data defined in the pagedef
      - Most often used for method arguments
      - Not linked to a Data Control



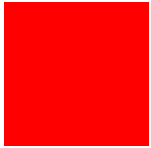
# Executables Cont.

- InvokeAction
  - Allows the invocation of a method binding automatically within the lifecycle
    - An alternative to invocation via user action e.g. button press
  - Controlled using:
    - Refresh attribute: When in the lifecycle it fires
    - RefreshCondition: EL expression to refine execution
  - Examples:
    - `RefreshCondition="$ {!adfFacesContext.postback}"/>`
    - `RefreshCondition="`  
 `"${(userState.currentSvrId != bindings.svrId.inputValue)}`  
 `|(userState.editRefresh)}"`
- Important!
  - Pay attention to side effects, for example: Double Invocation when Bindings based on a method iterator will implicitly invoke the method



# Refresh Settings

Value	Meaning
always	Invoke on page entry or postback
deferred	Only invoke if the binding is used (e.g. table may or may not be rendered based on some other condition)
ifNeeded	Let ADF decide – often used with ADF BC where the model is "self aware"
never	Use when only invoking from code
prepareModel	See lifecycle next – also has ifNeeded variant
renderModel	See lifecycle next – also has ifNeeded variant

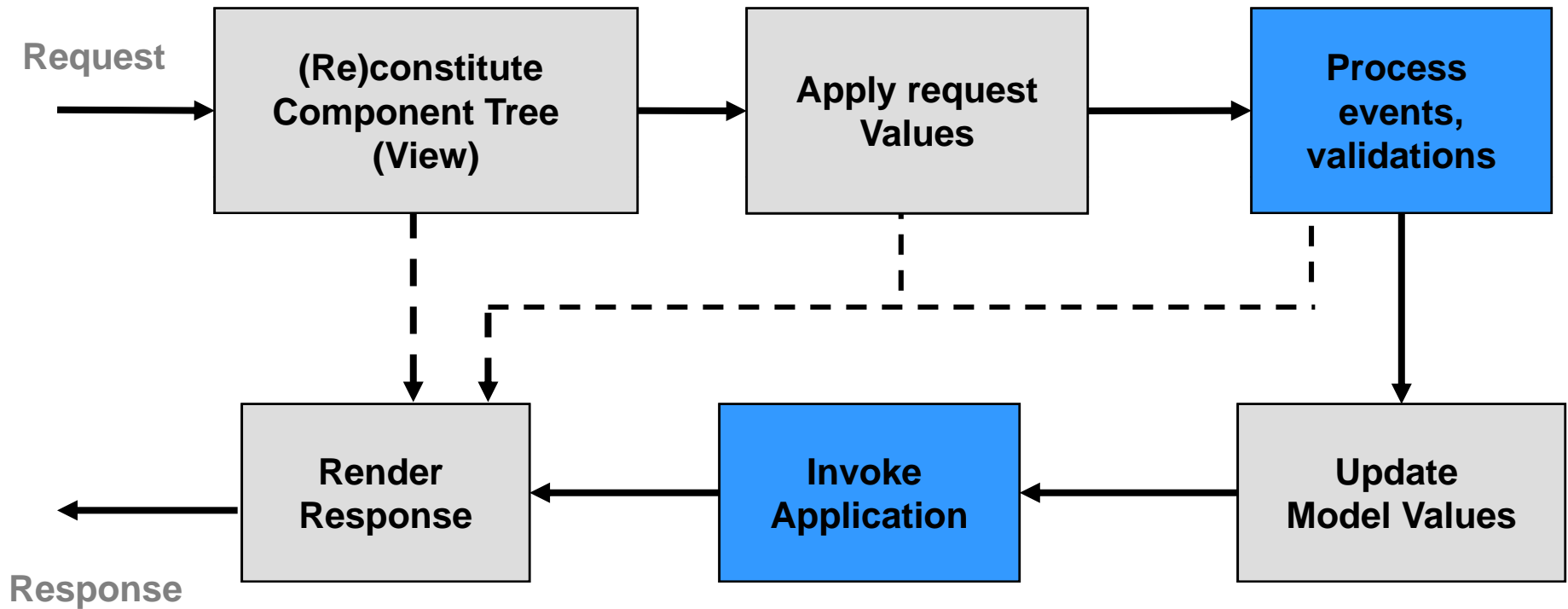


# The LifeCycle

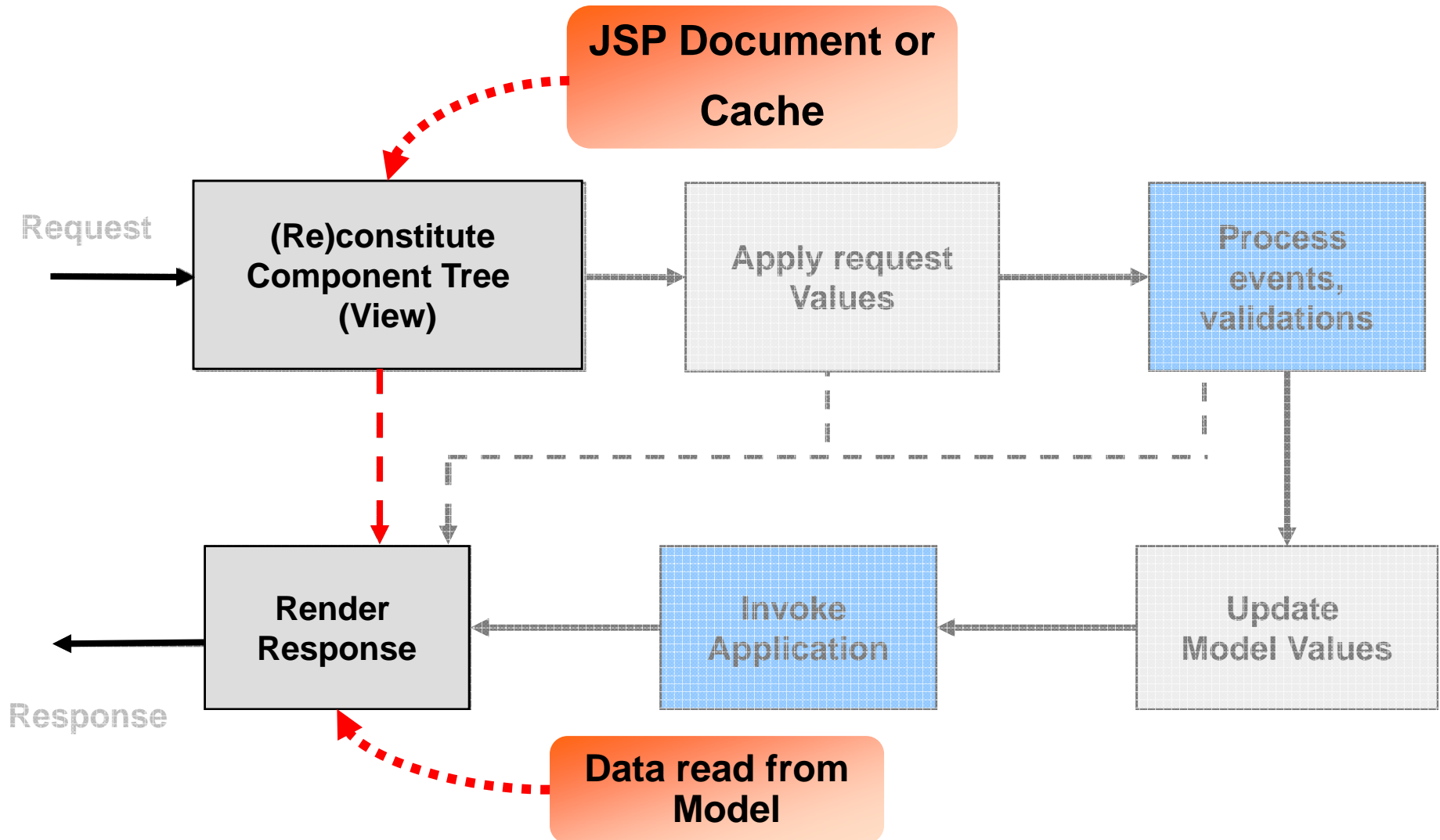




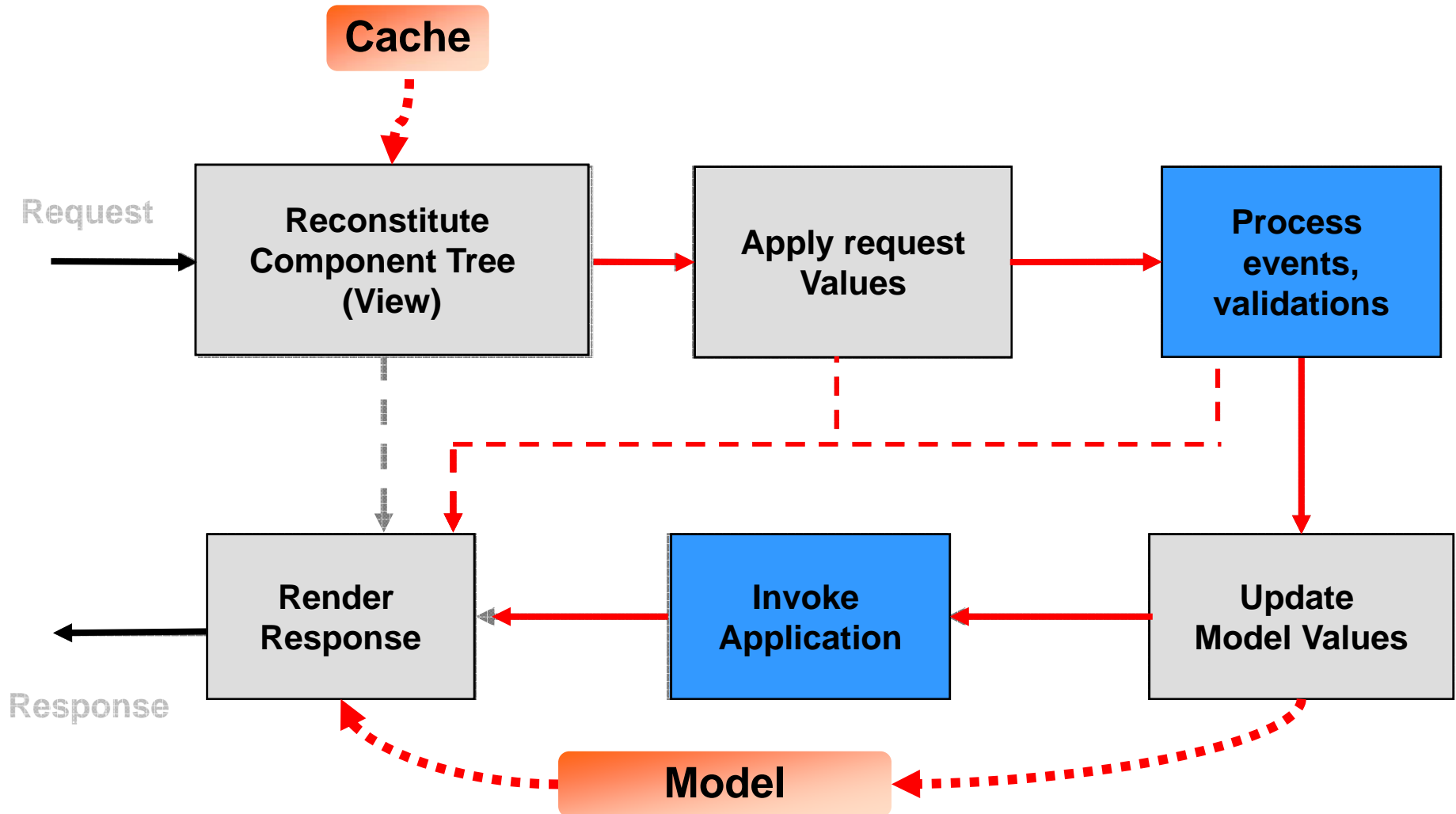
# The Core JSF Lifecycle



# Core JSF Displaying a New Page

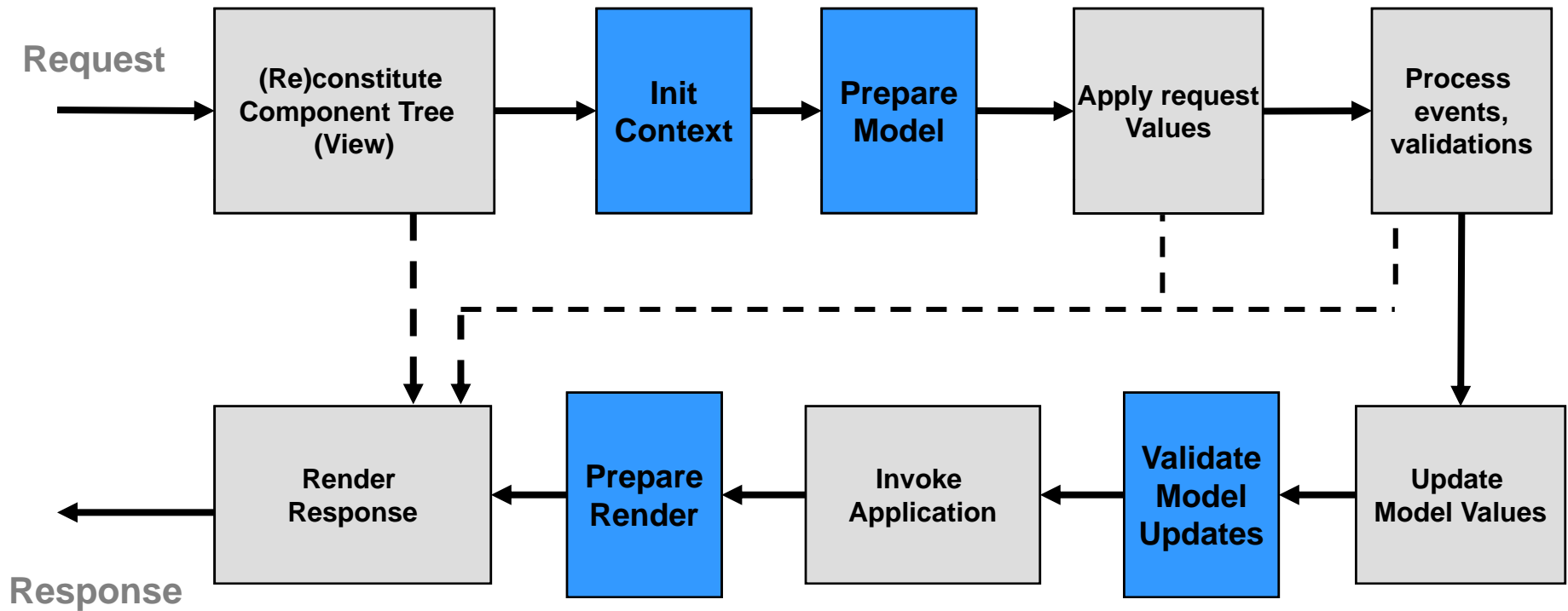


# Core JSF Postback to a Page





# The ADFm Lifecycle in Context





# Using the Lifecycle

- Declarative
  - Control execution via executables "Refresh" attribute
    - prepareModel
    - renderModel
- Code
  - PagePhaseListener – Decorates the lifecycle
  - Custom page controller - subclass and extend the lifecycle for a page
    - Use with caution remember to call the super-class!
  - Complete custom lifecycle



# The LifeCycle in 11g

11g

- Prepare Render should no longer be used
  - Use Method Activites on TaskFlows instead
- Deferred is now the default refresh setting



# Defining Listener / Custom Controllers

- Listeners
  - Implement `oracle.adf.controller.v2.lifecycle.PagePhaseListener`
  - Very similar to JSF phase listeners
- Page Controller
  - Extend `oracle.adf.controller.v2.lifecycle.PageController`
  - Methods for prepareRender etc.
- Set in the PageDef
  - ControllerClass attribute of the PageDef <pageDefinition> element
  - Auto-detects listener / controller



# Custom Listener Example

- Code

```
public class CookieSettingListener implements PagePhaseListener {
    public void beforePhase(PagePhaseEvent event) { }
    public void afterPhase(PagePhaseEvent event) {
        if (event.getPhaseId() == Lifecycle.PREPARE_RENDER_ID)
        {
            setCookie();
        }
    }
}
```

- Plugged in

```
<pageDefinition xmlns="http://xmlns.oracle.com/adfm/uimodel"
    version="..."
    id="homePageDef"
    Package="groundside.com.view.pageDefs"
    ControllerClass="groundside.com.fwkext.CookieSettingListener">
```



# Defining a Completely Custom Lifecycle

- Controlled (in JSF) by a Phase Listener

```
<faces-config xmlns="http://java.sun.com/JSF/Configuration">
  <lifecycle>
    <phase-listener>
      oracle.adf.controller.faces.lifecycle.ADFPhaseListener
    </phase-listener>
  </lifecycle>
</faces-config>
```

...

- Can be replaced

```
public class SRDemoADFPhaseListener extends ADFPhaseListener {
  protected PageLifecycle createPageLifecycle() {
    return new SRDemoPageLifecycle();
  }
}
```

# Custom Error Handling in 11g

- Previously required a whole custom lifecycle
- In 11g extend  
`oracle.adf.model.binding.DCErrorHandlerImpl`
- Register `ErrorHandlerClass` in `DataBindings.cpx`



# Writing Code that Interacts with Bindings

1. Getting access to the binding in code – best practice
  - Inject the `#{bindings}` object into the managed bean

```
<managed-bean>
  <managed-bean-name>EditPageBean</managed-bean-name>
  <managed-bean-class>view.backing.EditPageBean</managed-bean-class>
  <managed-bean-scope>request</managed-bean-scope>
  <managed-property>
    <property-name>bindings</property-name>
    <value>#{bindings}</value>
  </managed-property>
</managed-bean>
```

- Add getters and setters for `oracle.binding.BindingContainer`
- JDeveloper will do this automatically if you let it maintain backing beans for you

# Changes in 11g

## 1. Getting access to the binding in code

- Lookup the object in the getter

```
public BindingContainer getBindings() {
    if (this.bindings == null) {
        FacesContext fc = FacesContext.getCurrentInstance();
        this.bindings =
            (BindingContainer)fc.getApplication().evaluateExpressionGet(fc,
                "#{bindings}", BindingContainer.class);
    }
    return this.bindings;
}
```

- JDeveloper will do this automatically if you let it maintain backing beans for you



# Writing Code that Interacts with Bindings

## 2. Using the bindings object

```
AttributeBinding deptBinding =  
    (AttributeBinding)getBindings().getControlBinding("DepartmentName");  
String departmentName = (String)deptBinding.getInputValue();
```

---

```
BindingContainer bindings = getBindings();  
OperationBinding operationBinding = bindings.getOperationBinding("First");  
Object result = operationBinding.execute();
```

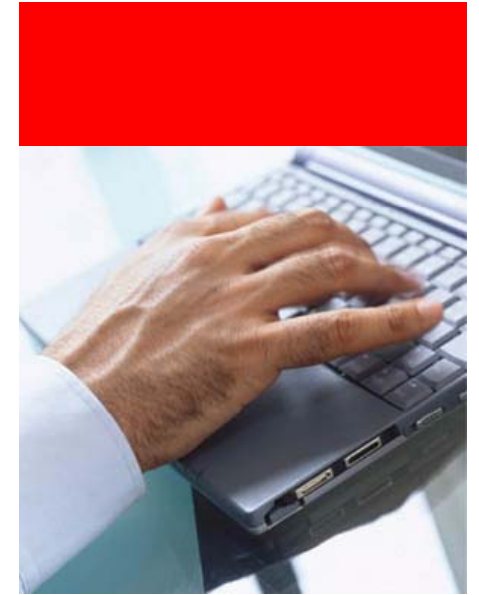
---

```
OperationBinding operationBinding =  
getBindings().getOperationBinding("findDepartmentManagerId");  
Map params = operationBinding.getParamsMap();  
params.put("searchTerm", "Sales");  
Number deptManager = operationBinding.execute();
```



# Summary – The Secrets of Binding

- Understand the files involved
  - Don't be afraid to edit them
  - Understand the abstraction / separation between the binding and the UI
- Understand the lifecycle and how to change it
  - Listen / extend / replace
- Debugging!
  - -Djbo.debugoutput=console
  - New Debugger **11g**
- There is a lot of information out there...



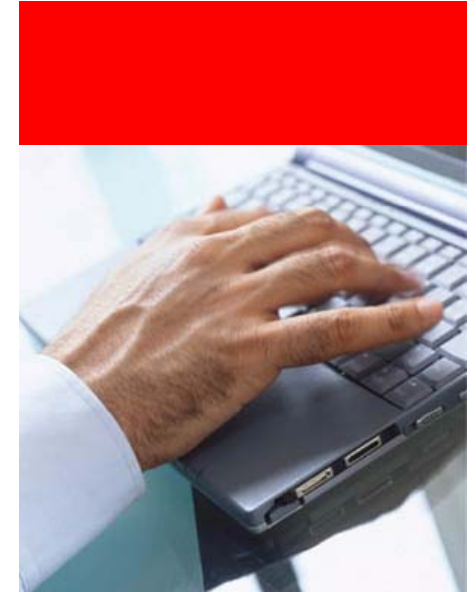
# Demonstration

---

## Debugging Bindings in 11g

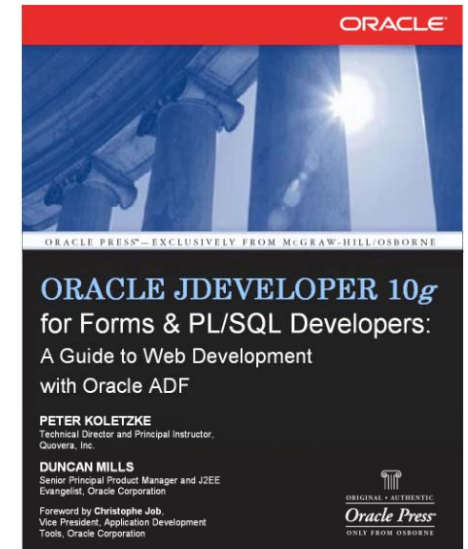
# More at OOW

- Upcoming Related Sessions
  - MS Office Front Ends for Oracle Application Development Framework Applications. 4pm today
  - RIAs and Web 2.0 Development Made Simple 1:30pm Thursday
- Hands On Labs
  - Data Visualization for Web Applications 4pm today
- Oracle Demoground – Moscone South
  - A28 - Oracle JDeveloper and Oracle ADF
  - A25 – RIA - Oracle ADF Faces



# Learn More

- [Oracle.com/technology/jdev](http://Oracle.com/technology/jdev)
  - Download
  - Tutorials
  - Discussion forum
  - Blogs
  - Samples
- Oracle JDeveloper 10g from Oracle Press
  - Chapters 7 & 8
  - R11 version next year!



# Participate in Usability Activities

**Win a Wii**

**Help us build tools that help you.**

*and*

**Receive a Gift and a chance to win a Wii.**



**Find us at the OTN Lounge in Moscone West**

**ORACLE®**