

ORACLE®

# **Automating the Implementation of User Interface Design Patterns in JSF**

**Steven Davelaar  
JHeadstart Team  
Oracle Consulting**

# User Interface Patterns

Select	EmployeeId	FirstName	LastName	Email	Salary
<input type="radio"/>	100	Steven	King	SKING	24000
<input checked="" type="radio"/>	101	Neena	Kochhar	NKOCHHAR	17000
<input type="radio"/>	115	Alexander	Khoo	AKHOO	3100
<input type="radio"/>	122	Payam	Kaufling	PKAUFLIN	7900
<input type="radio"/>	156	Janette	King	JKING	10000
<input type="radio"/>	173	Sundita	Kumar	SKUMAR	6100

Summary  
View

Record Browsing

⏪ [2 / 6] ⏩ ⏪ ⏩

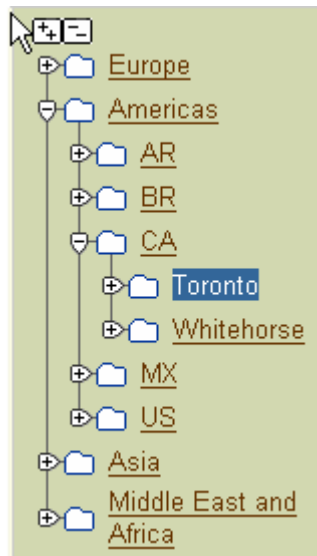
* EmployeeId	<input type="text" value="101"/>	* JobId	<input type="text" value="AD_VP"/>
FirstName	<input type="text" value="Neena"/>	Salary	<input type="text" value="17000"/>
* LastName	<input type="text" value="Kochhar"/>	CommissionPct	<input type="text" value=""/>
* Email	<input type="text" value="NKOCHHAR"/>	ManagerId	<input type="text" value="King"/>
PhoneNumber	<input type="text" value="515.123.4568"/>	Department	<input type="text" value="Executive"/>
* HireDate	<input type="text" value="21-Sep-1989"/>		

Detail  
View

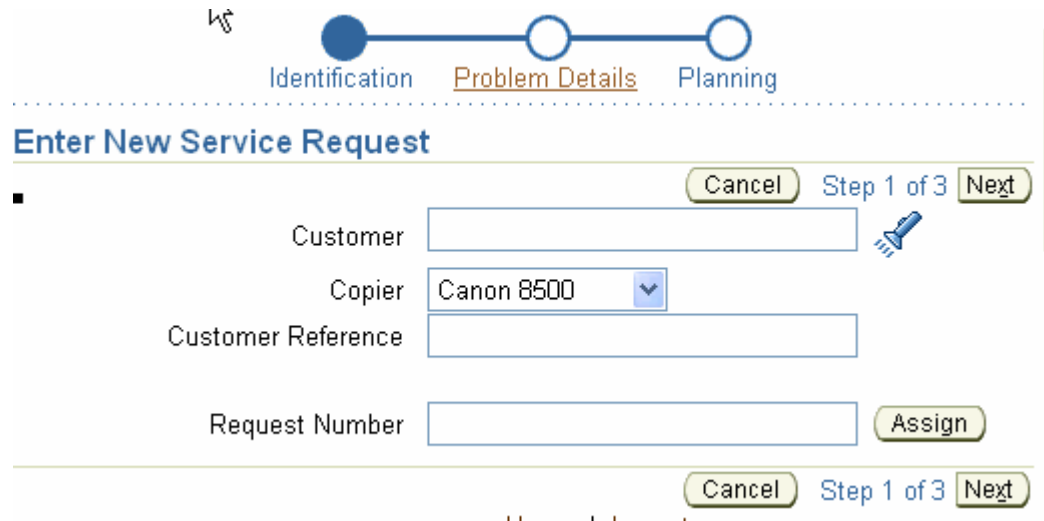
# User Interface Patterns



## Menu-based Navigation



Tree-based Navigation




Wizard-style Navigation

# User Interface Patterns

Filter By JobTitle

## Simple Search

Result matches all conditions  
 Result matches any condition  
Case Sensitive?

EmployeeId   
FirstName   
LastName starts with   
Email   
PhoneNumber   
HireDate  

## Advanced Search

http://localhost:8992 - Search and Select Manager - Mo...

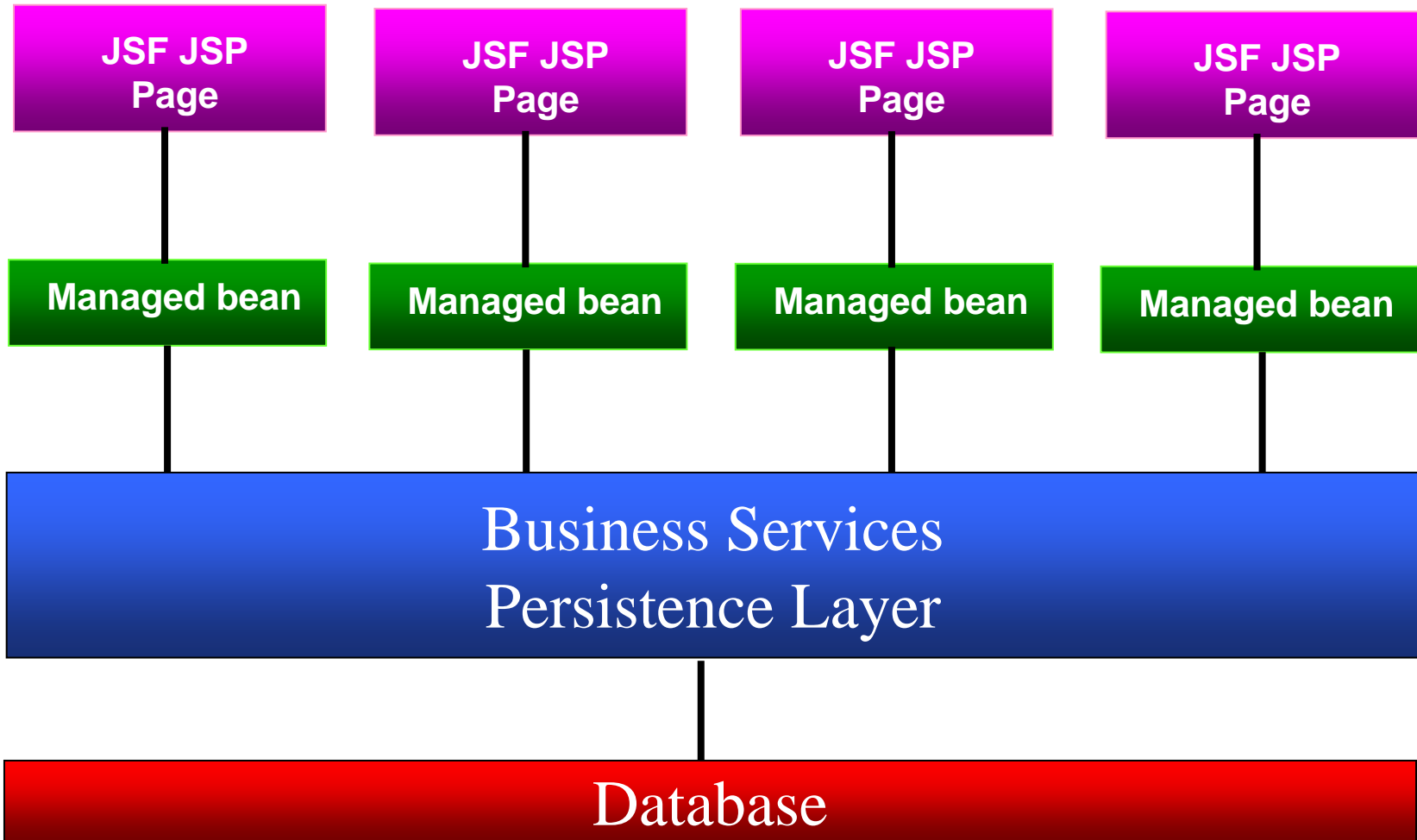
Search and Select Manager

Filter By LastName

Select	EmployeeId	FirstName	LastName	Email
<input checked="" type="radio"/>	106	Valli	Pataballa	VPATABAL
<input type="radio"/>	113	Luis	Popp	LPOPP
<input type="radio"/>	136	Hazel	Philtanker	HPHILTAN
<input type="radio"/>	140	Joshua	Patel	JPATEL
<input type="radio"/>	146	Karen	Partners	KPARTNER
<input type="radio"/>	191	Randall	Perkins	RPERKINS

## List of Values

# Building JSF Web Apps



**ORACLE®**

D E M O N S T R A T I O N

**JSF Web App**

**“the Struts Way”**

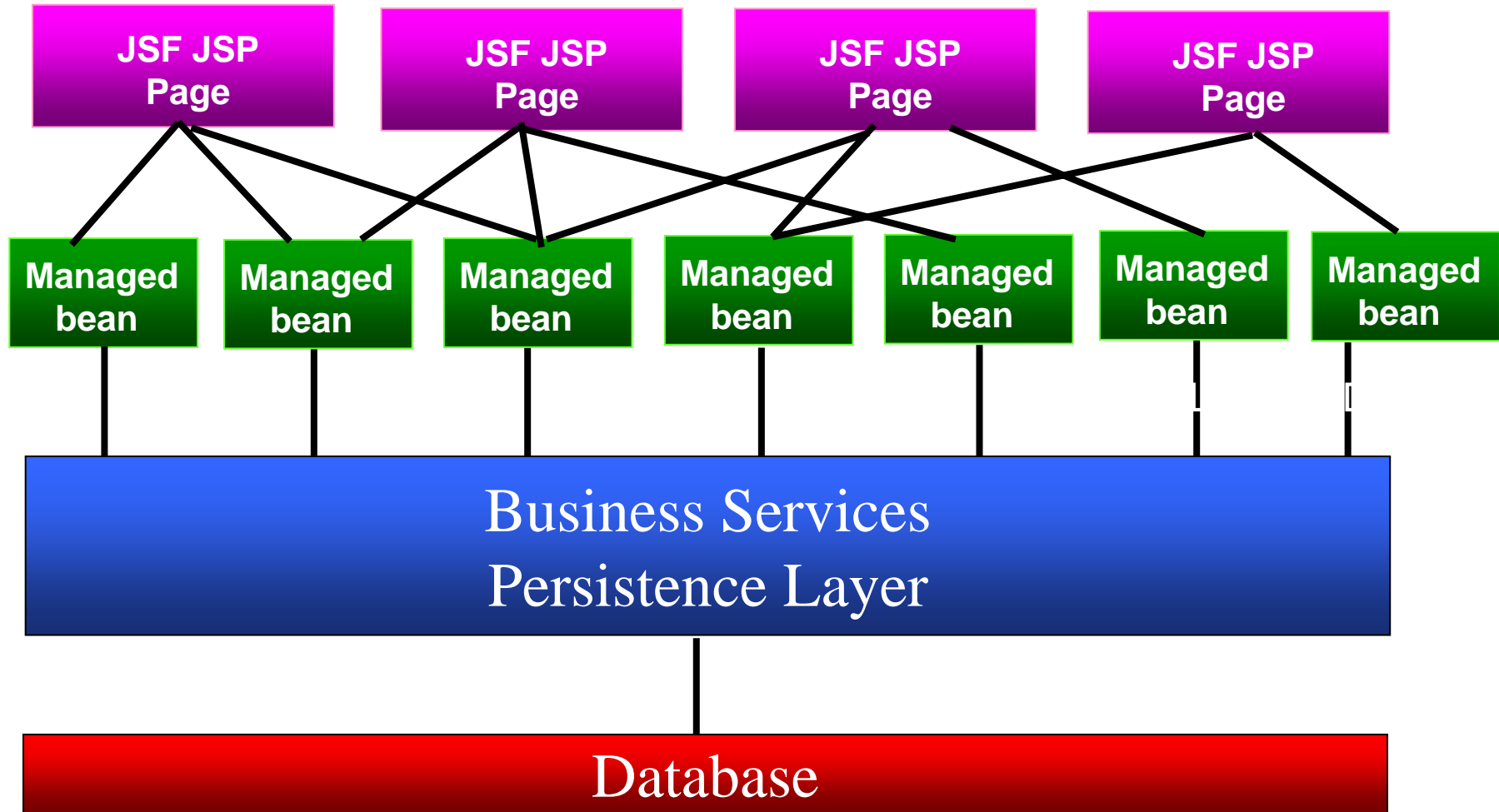
# JSF Apps “the Struts Way”

- One backing bean per page
  - Includes code to support all user interface patterns present in the associated page
  - Extends “Base” Backing Bean with utility methods
- Often encouraged by IDE support:
  - Auto-creation of page backing beans
- Recurring code patterns in all backing beans.
- How to Improve Productivity
  - Copy and Paste of Backing Beans?
  - Generation of backing bean classes?

# A Better Solution for Increased Productivity

- One page can use multiple managed beans!
- Create separate managed beans for each User Interface Pattern (UIP), e.g.
  - Search Bean, Collection Bean, Shuttle Bean, etc.
- Managed beans for each UIP reuse the same Java Class!
  - Configured for specific usage in page in faces-config: Dependency Injection

# Building Smart JSF Web Apps



# Generic Search Bean Class

```
<managed-bean>
  <managed-bean-name>DeptSearchBean</managed-bean-name>
  <managed-bean-class>view.SearchBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
  <managed-property>
    <property-name>handler</property-name>
    <value>#{DepartmentHandler}</value>
  </managed-property>
  <managed-property>
    <property-name>collectionBean</property-name>
    <value>#{DeptCollectionBean}</value>
  </managed-property>
</managed-bean>
```

```
<af:panelGroup layout="horizontal">
  <af:inputText label="Filter by name"
    value="#{DeptSearchBean.searchText}"/>
  <af:commandButton text="Go"
    actionListener="#{DeptSearchBean.quickSearch}"/>
</af:panelGroup>
```

```
<managed-bean>
  <managed-bean-name>EmpSearchBean</managed-bean-name>
  <managed-bean-class>view.SearchBean</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
  <managed-property>
    <property-name>handler</property-name>
    <value>#{EmployeeHandler}</value>
  </managed-property>
  <managed-property>
    <property-name>collectionBean</property-name>
    <value>#{EmpCollectionBean}</value>
  </managed-property>
</managed-bean>
```

```
<af:panelGroup layout="horizontal">
  <af:inputText label="Filter by Last Name"
    value="#{EmpSearchBean.searchText}"/>
  <af:commandButton text="Go"
    actionListener="#{EmpSearchBean.quickSearch}"/>
</af:panelGroup>
```

Filter by last name

**ORACLE®**

D E M O N S T R A T I O N

**JSF Web App**

**“the Smart Way”**

**ORACLE®**

# Calling the Business Service

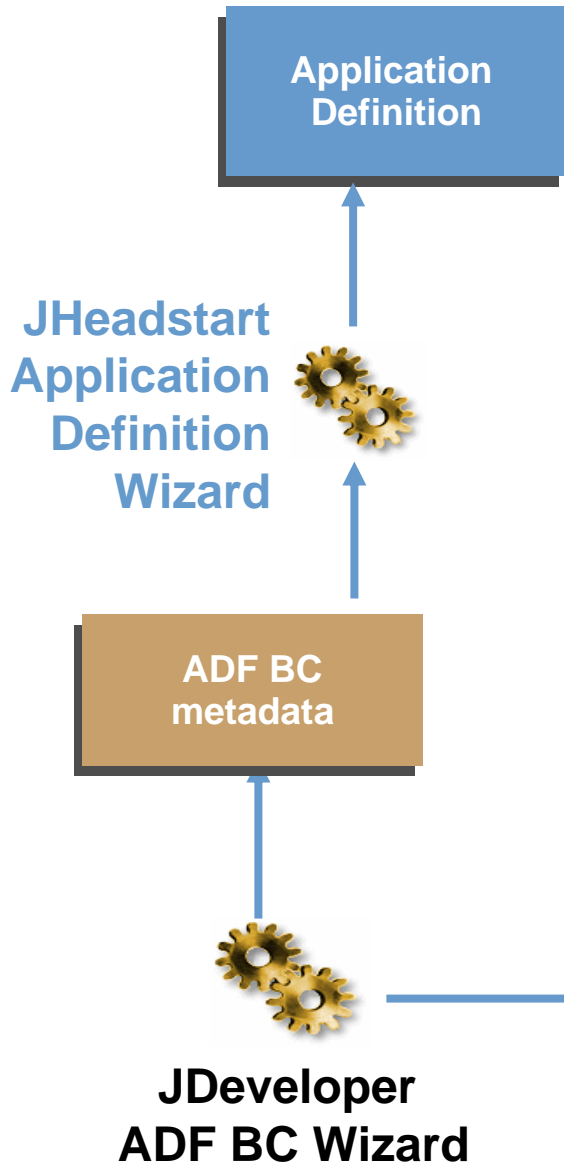
- Several Options to Call Specific Business Service Methods from Generic Beans:
  - Introspection
  - Generic Interfaces
  - Generic Service methods that delegate to specific DAO's, based on a method argument (e.g. Bean name to use with Spring Bean Factory Lookup)

# Getting Even More Productive

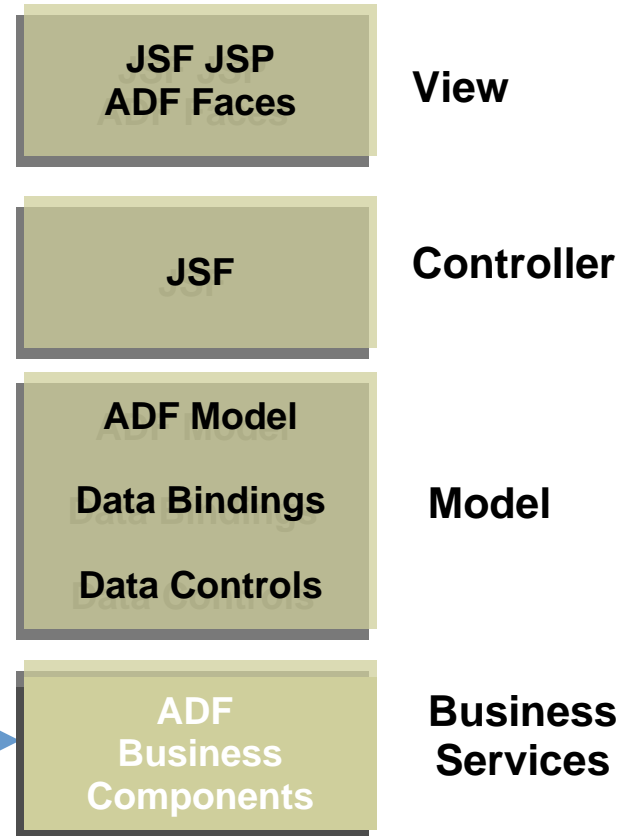
- Recurring code patterns have been eliminated
- What's left?
  - Recurring managed bean definitions in faces-config
  - Recurring JSF page snippets for each user interface pattern
- Generation of the pages and faces-config
  - Create XML metadata format that defines pages and user interface patterns on the pages
  - JHeadstart implements this approach

# What is JHeadstart?

- Proven Development toolkit for rapid J2EE application development
- Current JHeadstart release 10.1.3 Preview
  - Compatible with JDeveloper 10.1.3
  - Builds on ADF Runtime architecture
  - Fully integrates with ADF Design Time
- Adds an extra level of productivity
  - JHeadstart Application Generator generates complete ADF applications
  - Like the Designer Form Generator generates Forms apps, JHeadstart generates ADF apps
- Implements ADF Best Practices



# Generation Process



JHeadstart  
Application  
Definition  
Wizard

Application  
Definition

ADF BC  
metadata

JDeveloper  
ADF BC Wizard

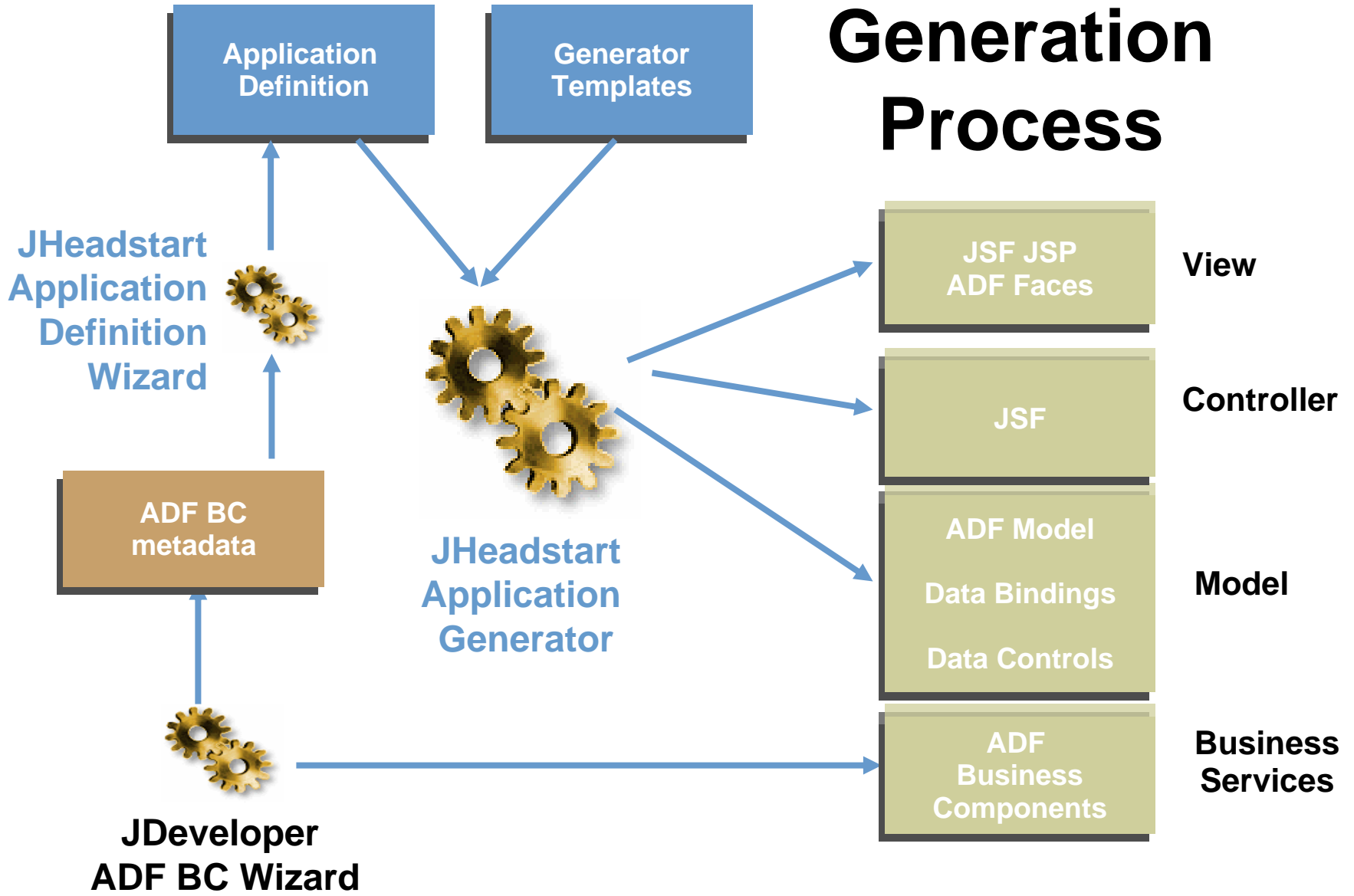
The screenshot shows the 'AppModuleApplicationDefinition.xml - Application Definition Editor' window. The left pane displays a tree view of the application structure, with 'Employees' selected under the 'AppModule' folder. The right pane shows the configuration for the 'Employees' entity, including identification, group layout, and query settings.

Identification	
Name	Employees
Short Name	
Description	Employees
Use as List of Values?	<input type="checkbox"/>
Group Image / Icon	
Group Layout	
Layout Style	table-form
Table Overflow Style	inline
Stack Detail Groups on Same...	<input type="checkbox"/>
Same Page?	<input type="checkbox"/>
Query Settings	
Data Collection	EmployeesView1
Data Collection Implementation	EmployeesView
Query Bind Parameters	

This attribute determines the number and layout of the pages that are generated to select, view and manipulate the group ViewObject.  
form: generates a single-record page.  
table: generates a single page in Table (multi-record) format.  
table-form: generates a Table Page, as well as a single-record Details page.  
select-form: generates a Select Page with a list box where the user can select a record, and a single-record page.

Properties Templates

Help Apply OK Cancel



# Generator Outputs

- JSF Pages using ADF Faces
  - Component hierarchies and binding EL
- Page Flow, Bean Definitions
  - faces-config.xml
- ADF Data Binding
  - DataBinding Context & Page Definitions
- Translatable Text
  - In Resource Bundles or Properties Files
- No Java Code is Generated, Just XML

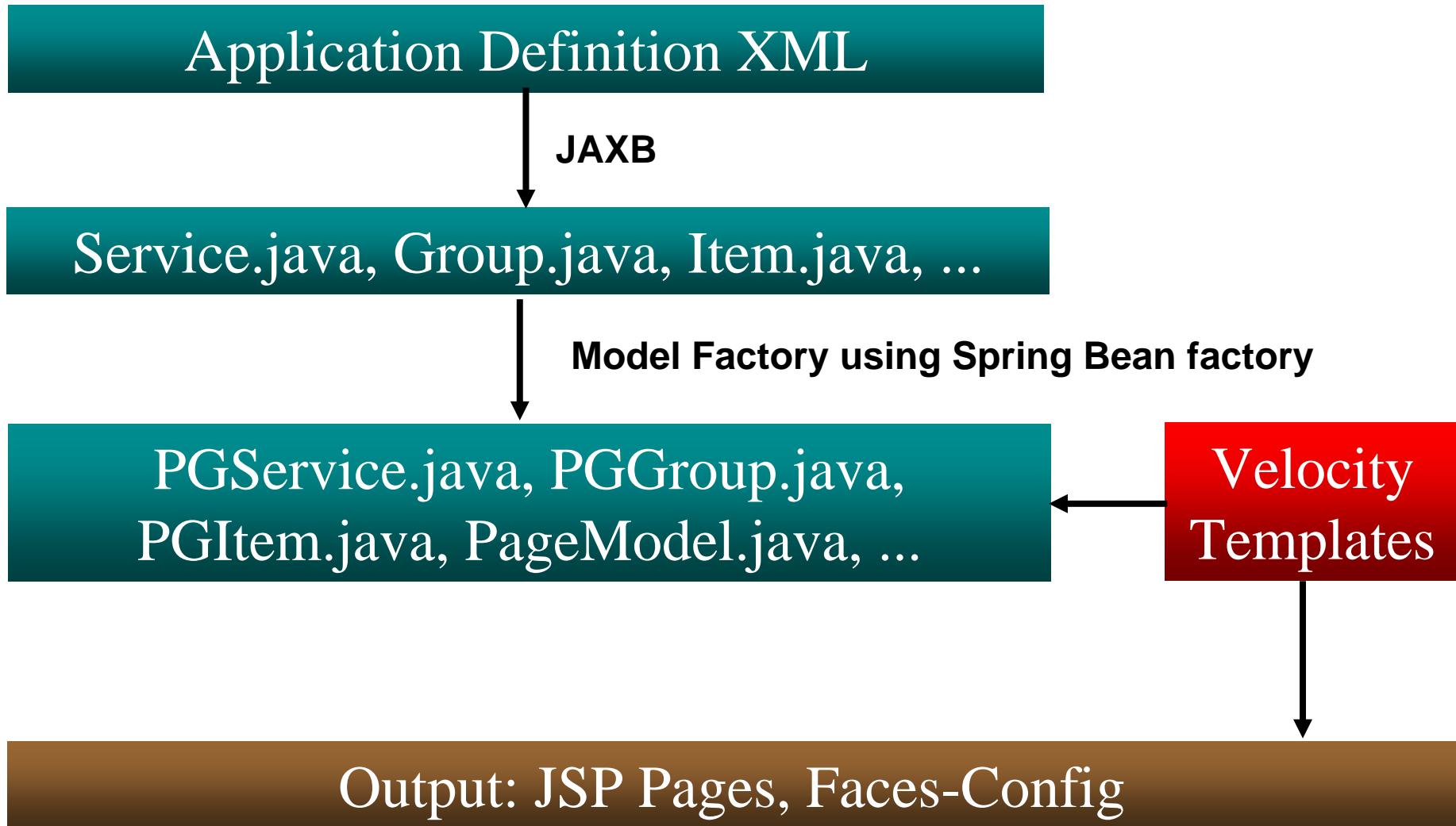
**ORACLE®**

D E M O N S T R A T I O N

# **JHeadstart Application Generator**

ORACLE®

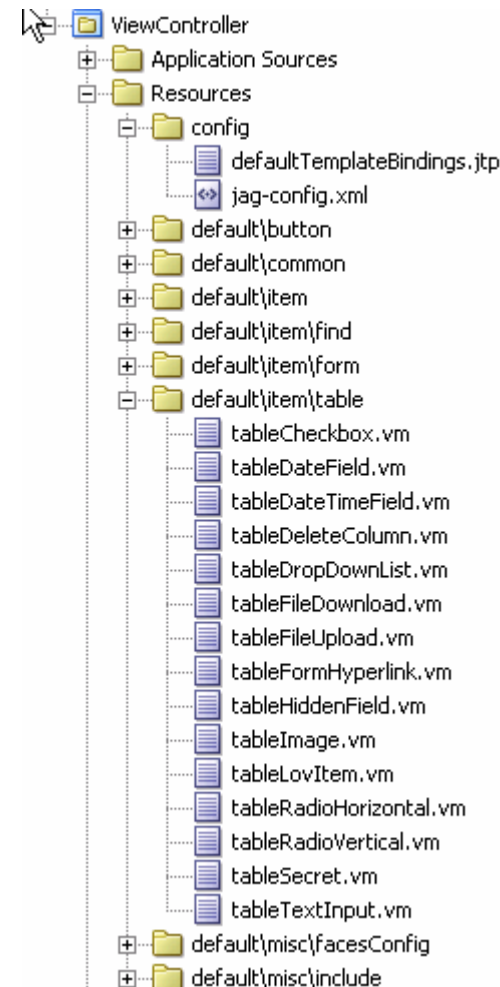
# JHeadstart Generator Architecture



# JHeadstart Generator Templates

- The content of generated pages is **completely** driven by generator templates
- The templates are categorized in groups
  - Page, pageComponent, button, item (table/form/search), search, misc, etc.
- Which template is used for which page “snippet” is handled by defaultTemplateBindings.jtp file.

```
# -----  
# TABLE ITEM TEMPLATES  
# -----  
TABLE_TEXT_INPUT=default/item/table/tableTextInput.vm  
TABLE_CHECK_BOX=default/item/table/tableCheckbox.vm  
TABLE_DROP_DOWN_LIST=default/item/table/tableDropDownList.vm  
TABLE_RADIO_VERTICAL=default/item/table/tableRadioVertical.vm  
TABLE_RADIO_HORIZONTAL=default/item/table/tableRadioHorizontal.vm  
TABLE_LIST=default/item/table/tableList.vm
```



# Velocity Template Language (VTL)

- The Generator Templates use VTL
- Velocity is Open Source Product from Apache Foundation
- Velocity Developers Guide
  - <http://jakarta.apache.org/velocity/docs/developer-guide.html>
- VTL Reference Guide
  - <http://jakarta.apache.org/velocity/docs/vtl-reference-guide.html>

# VTL Construct Examples

```
[ #if (!($subRegion.title=""))
  <af:panelHeader text="#REGION_TITLE($subRegion)">
  #JHS_PARSE($subRegion.templateIdentifier $subRegion)
</af:panelHeader>
#else
  <af:panelGroup #JHS_PROP("rendered" $subRegion.rendered)>
  #JHS_PARSE($subRegion.templateIdentifier $subRegion)
</af:panelGroup>
#end
```

If-then-else

```
#foreach($item in ${JHS.current.itemContainer.items})
#  #if ($item.displayInForm)
#    #JHS_PARSE($item.formTemplateIdentifier $item)
#  #end
#end
```

Loop

# Summary

- Start Building Reusable Managed Beans
  - Configure the Beans for usage in a page in Faces-Config
- Consider Building a View/Controller Generator
  - Easier than you might think when using tools like XSLT, JAXB, Spring and Velocity
- Or.... Start using JHeadstart
  - Extremely flexible and extendable

# More Info on JHeadstart

- Google for “JHeadstart Product Center”
- White Paper Available at Oracle Stand

# Latest News: England – T & T: 0-1



4. Yorke 0-1

ORA

**ORACLE®**