Introduction to Java Server Faces(JSF)

Deepak Goyal

Vikas Varma

Sun Microsystems

Objective

Understand the basic concepts of

Java Server™ Faces[JSF]

Technology.

Agenda

- What is and why JSF?
- Architecture Overview
- UI Component Model
- Development Steps

JavaServer™ Faces (JSF) Framework Is…

A server side user interface component framework for Java™ technology-based web applications

What is JSF?

- A specification and reference implementation for a web application development framework
  - Components
  - Events
  - Validators & converters
  - Navigation
  - Back-end-data integration

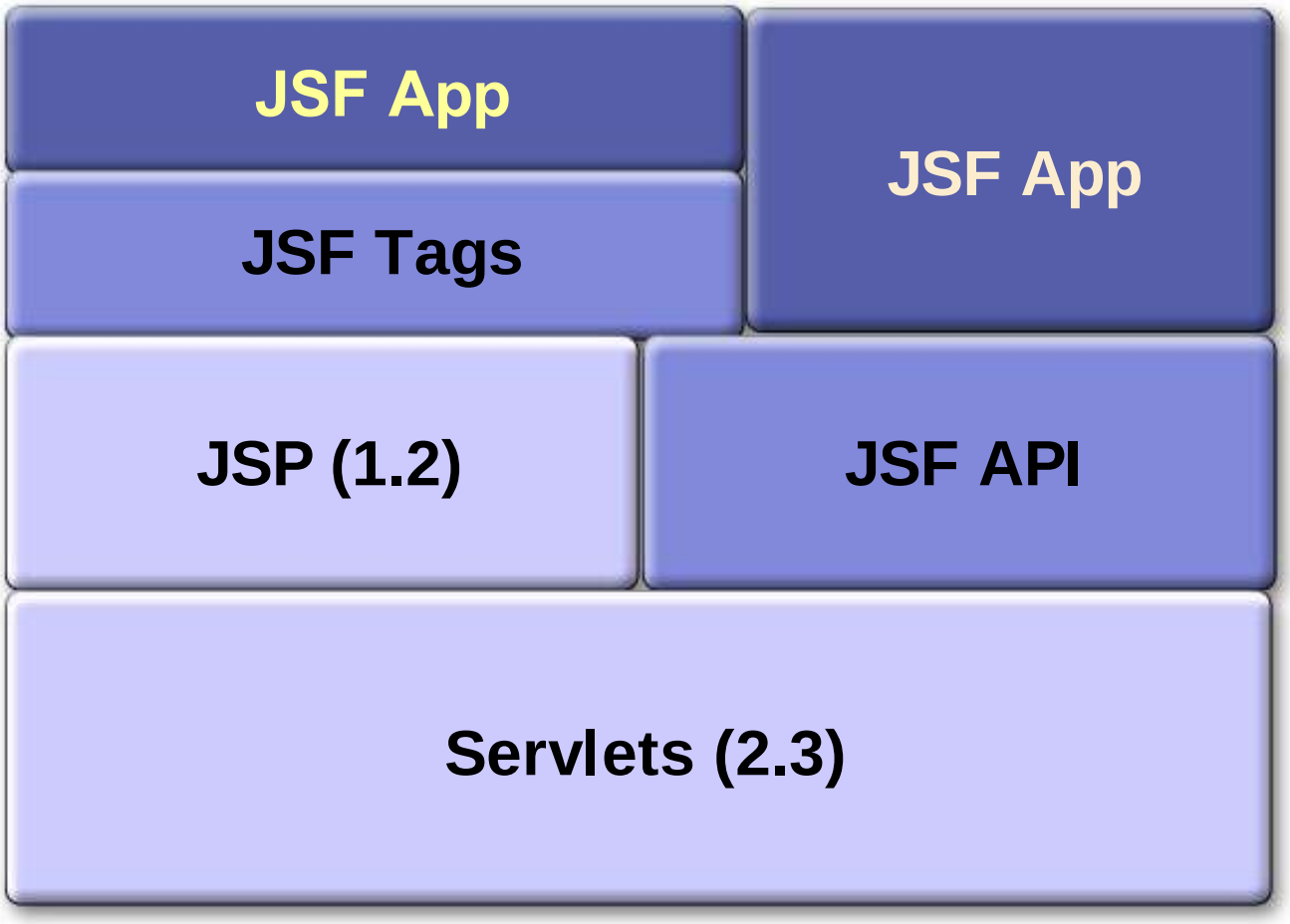
Why JSF? (page 1)

- MVC for web applications
- Clean separation of roles
- Easy to use
- Extendable Component and Rendering architecture
- Support for client device independence
- Standard
- Huge vendor and industry support

Why JSF? (page 2)

- JSP and Servlet
  - No built-in UI component model
- Struts (I am not saying you should not use Struts)
  - No built-in UI component model
  - No built-in event model for UI components
  - No built-in state management for UI components
  - No built-in support of multiple renderers
  - Not a standard (despite its popularity)
- Struts and JSF can be used together

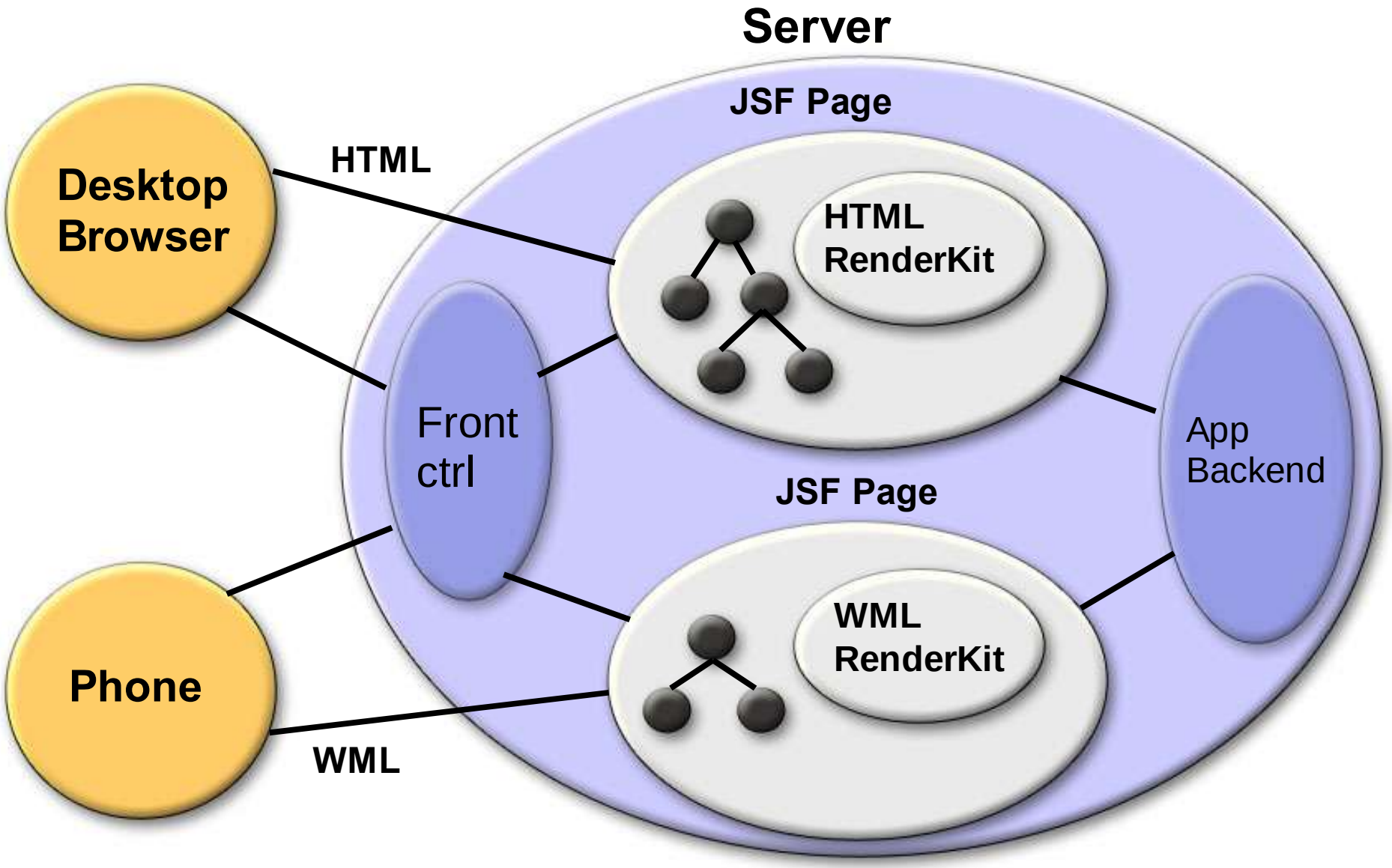
How the JSF Specification Fits In



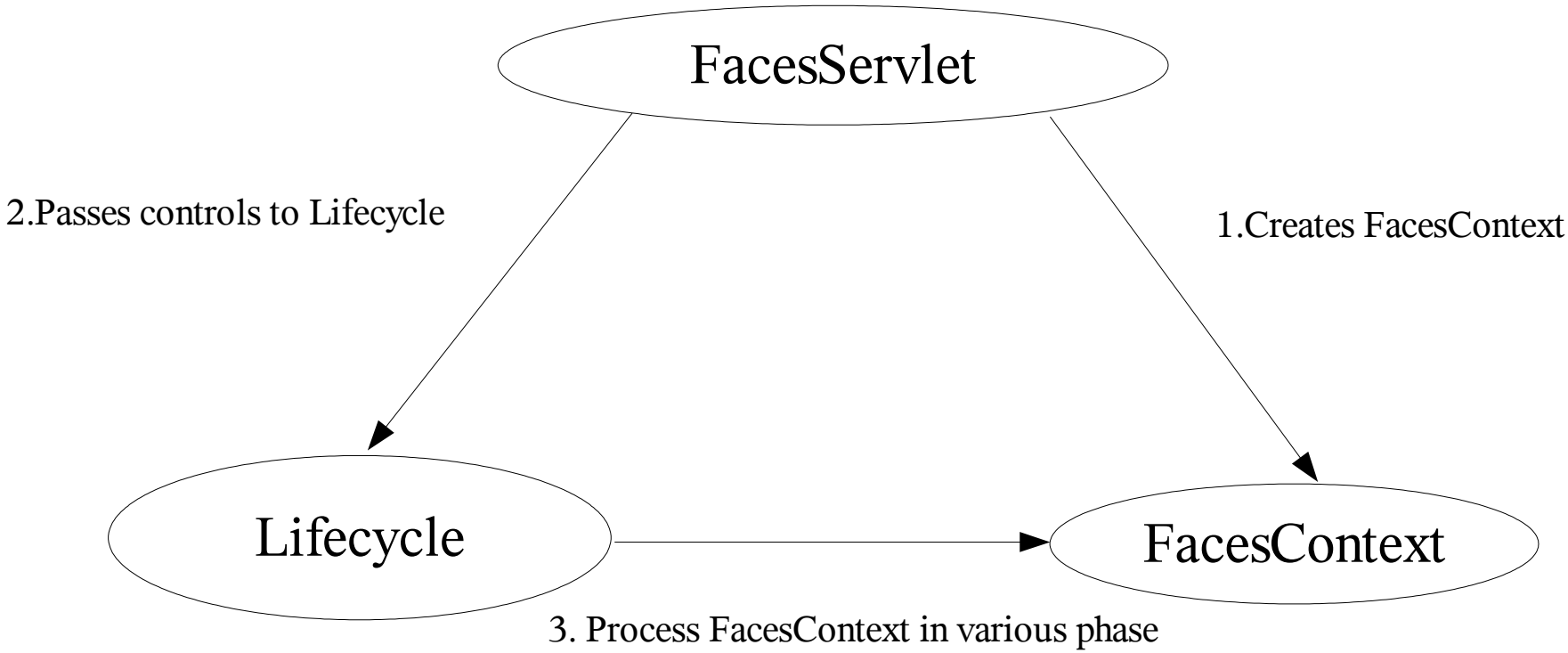
Agenda

- What is and why JSF?
- Architecture Overview
- UI Component Model
- Development Steps

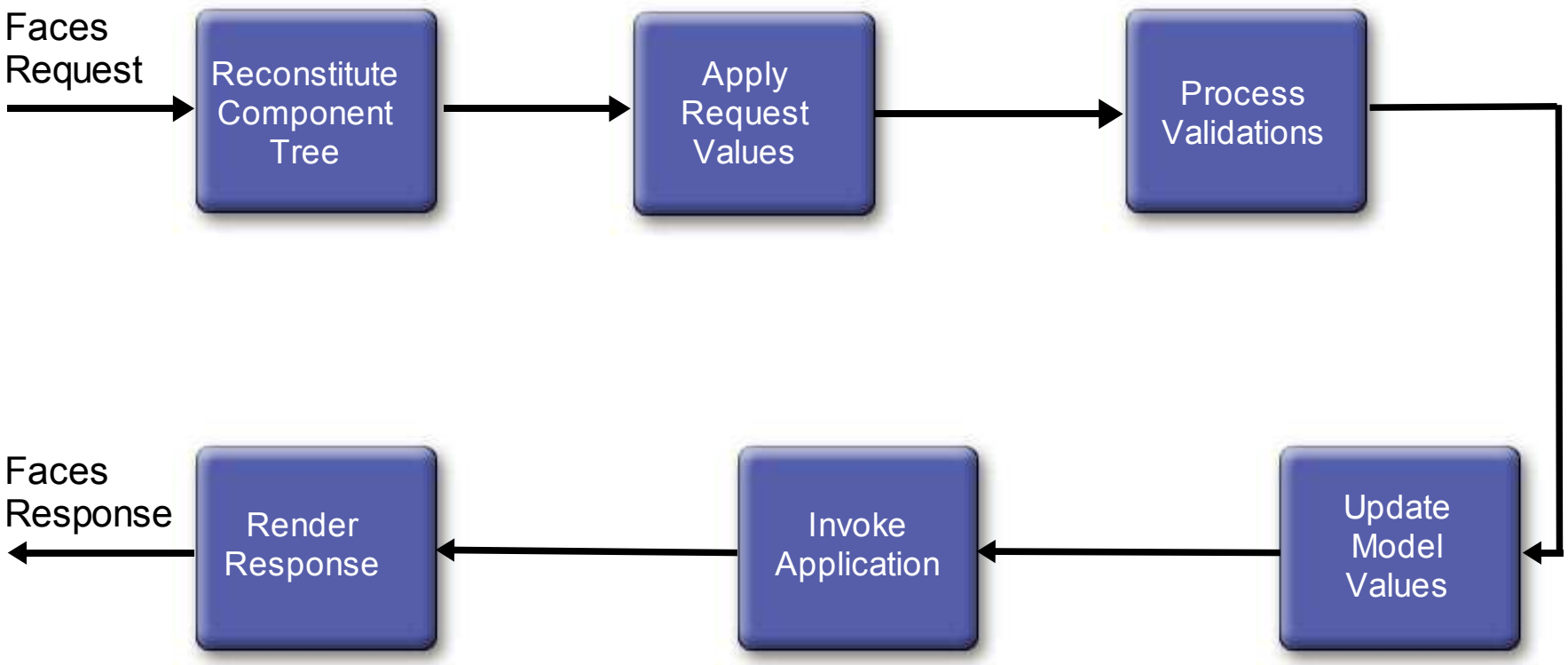
JSF Architecture [MVC]



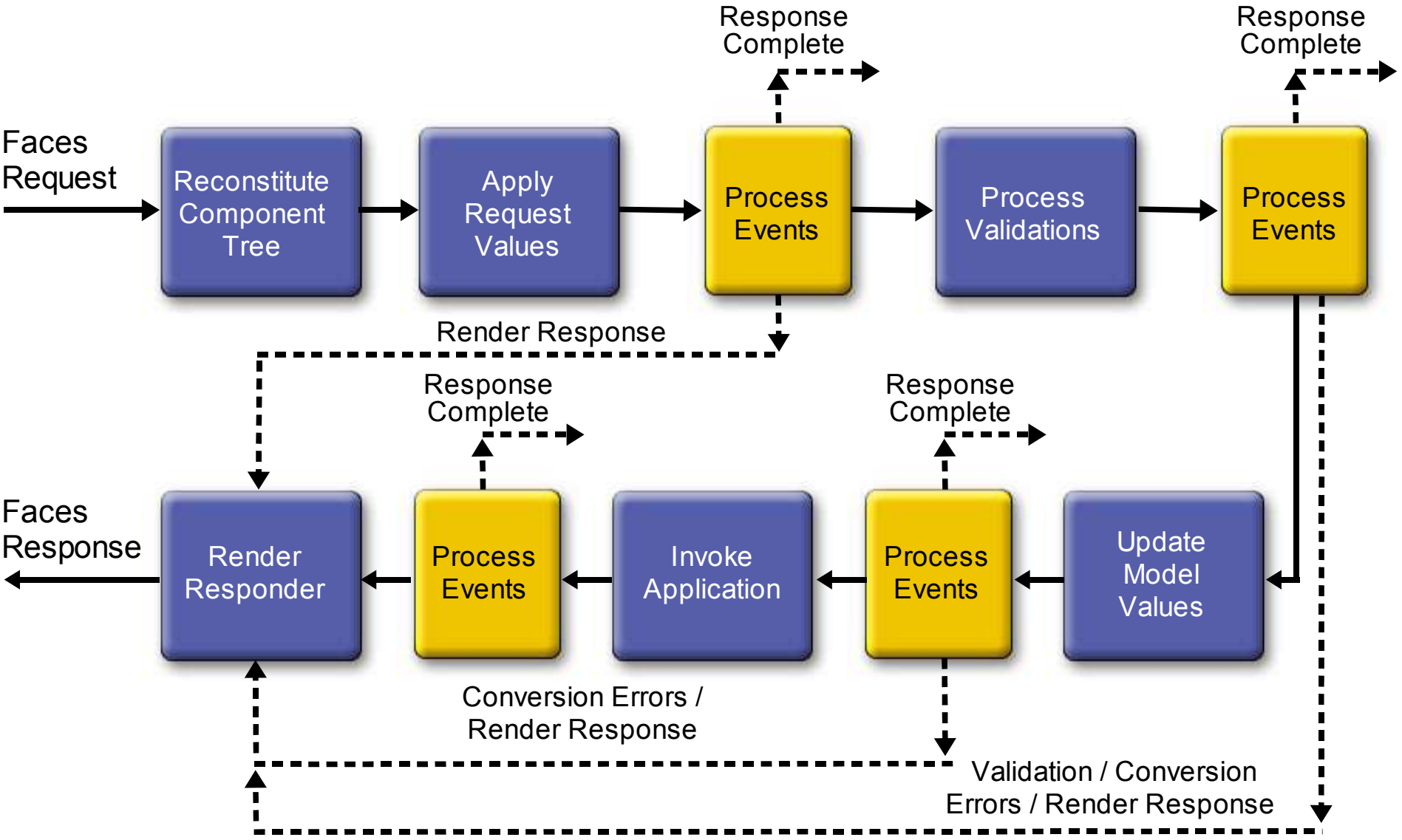
Request processing Lifecycle



Request Processing Lifecycle Phases



Request Processing Lifecycle



Request Processing Lifecycle Phases

1. Reconstitute component tree phase
2. Apply request values phase
3. Process validations phase
4. Update model values phase
5. Invoke application phase
6. Render response phase

Agenda

- What is and why JSF?
- Architecture Overview
- UI Component Model
- Development Steps

User Interface Component Model

- UI components
- Event handling model
- Conversion and Validation model
- Rendering model
- Page navigation support

UI Components

- UIComponent/UIComponentBase
  - Base class for all user interface components
- Standard UIComponent Subclasses
  - UICommand, UIForm, UIOutput
  - UIGraphic, UIInput, UIPanel, UIParameter
  - UISelectBoolean, UISelectMany, UISelectOne
- Example:

```
<h:inputText id="userNo"
             value="#{UserNumberBean.userNumber}"/>
```

Validators and Converters

- Validators—Perform correctness checks on UIInput values
  - Register one or more per component
  - Enqueue one or more messages on errors
  - Standard implementations for common cases
- Converters—Plug-in for conversions:
  - Output: Object to String
  - Input: String to Object
  - Standard implementations for common cases

Converters and Validators

Example:

Converters:

```
<h:input_text valueRef=”testingBean.today”
    convertor=”DateTime”/>
```

Validators:

```
<h:input_text valueRef=”testingBean.today”
<f:validator_length minimum=”6” maximum='10” />
```

Rendering Model

- Renderers-Adapt components to a specific markup language
  - Decoding
  - Encoding
- RenderKits—Library of Renderers
  - Map component classes to component tags
  - Is a custom tag library
  - Basic HTML RenderKit

Tag	Rendered as
command_button	Login
command_hyperlink	hyperlink

Events and Listeners

- Follows JavaBeans™ Specification design and naming patterns
- Standard events and listeners
  - ActionEvent—UICommand component activated by the user
  - ValueChangedEvent—UIInput component whose value was just changed

Navigation Model

- Application developer responsibility
  - Defined in Application configuration file (Facesconfig.xml)
- Navigation rules
  - Determine which page to go.
  - Navigation case

Navigation Model

```
<navigation-rule>
    <from-tree-id>/login.jsp</from-tree-id>

<navigation-case>
      <from-outcome>success</from-outcome>
     <to-tree-id>/menu.jsp</to-tree-id>
</navigation-case>

 <navigation-case>
      <from-outcome>failed</from-outcome>
      <to-tree-id>/error.jsp</to-tree-id>
   </navigation-case>

</navigation-rule>
```

Agenda

- What is and why JSF?
- Architecture Overview
- UI Component Model
- Development Steps

Steps in Development Process

1. Develop model objects which hold the data
2. Add model objects (managed bean) declarations to Application Configuration File faces-config.xml
3. Create Pages using UI component and core tags
4. Define Page Navigation in faces-config.xml
5. Configure web.xml

Step1: Develop model Objects (Managed Bean)

- The model (M) in MVC
- A regular JavaBeans with read/write properties
- May contain application methods and event handlers
- Use to hold data from a UI (page)
- Creation and lifetime is managed by JSF runtime
  - application, session, request
- JSF keeps the bean's data in sync with the UI

Step 2. Managed Bean Declaration

(Faces-config.xml)

```
01  <managed-bean>
02    <managed-bean-name>
03      LoginFormBean
04    </managed-bean-name>
05      <managed-bean-class>
06        myapp.LoginFormBean
07      </managed-bean-class>
08      <managed-bean-scope>
09        request
10      </managed-bean-scope>
11  </managed-bean>
```

Step 3: Create JSF Pages

- Must include JSF tag library
  - HTML and core tags
- All JSF tags must enclosed between a set of *view* tag
- Use JSF form and form component tags
  - <h:input_text> not <input type=”text”>
  - <h:command_button> not <input type=”submit”>
- May include validators and event listeners on any form components

Sample JSF™ Page (login.jsp)

```
01  <f:view>
02    <f:form formName="logonForm">
03      <h:panel_grid columns="2">
04        <h:output_text value="Username:"/>
05        <h:input_text id="username" length="16"
06                valueRef="logonBean.username"/>
07        <h:output_text value="Password:"/>
08        <h:input_secret id="password" length="16"
09                valueRef="logonBean.password"/>
10        <h:command_button type="submit"
11                    label="Log On"
12                actionRef="logonBean.logon"/>
13        <h:command_button type="reset"
14                    label="Reset"/>
15      </h:panel_grid>
16    </f:form>
17  </f:view>
```

Binding UI to Managed Bean

login.jsp

```
<h:input_text id="userName"
      valueRef="LoginFormBean.userName"/>
```

faces-config.xml

```
<managed-bean>
   <managed-bean-name>
      LoginFormBean
   </managed-bean-name>
   <managed-bean-class>
      myapp.LoginFormBean
   </managed-bean-class>
```

LoginFormBean.java

```
public class LoginFormBean
   ...
   public void
            setUserName(...) {
   public String
           getUserName(...) {
```

Step 4: Define Page Navigation Rules (Faces-config.xml)

```
01    <navigation-rule>
02      <from-tree-id>/login.jsp</from-tree-id>
03      <navigation-case>
04        <from-outcome>success</from-outcome>
05        <to-tree-id>/menu.jsp</to-tree-id>
06      </navigation-case>
07    </navigation-rule>
08
09    <navigation-rule>
010     <from-tree-id>/login.jsp</from-tree-id>
011     <navigation-case>
012       <from-outcome>failure</from-outcome>
013       <to-tree-id>/error.jsp</to-tree-id>
014     </navigation-case>
015   </navigation-rule>
```

Step 5: Configure (web.xml)

```
01    <context-param>
02        <param-name>
03         javax.faces.application.CONFIG_FILES
04        </param-name>
05        <param-value>/WEB-INF/faces-config.xml
06        </param-value>
07    </context-param>
08     <servlet>
09        <servlet-name>Faces Servlet</servlet-name>
10        <servlet-class>
11        javax.faces.webapp.FacesServlet</servlet-class>
12        <load-on-startup> 1 </load-on-startup>
13    </servlet>
14    <!-- Faces Servlet Mapping -->
15    <servlet-mapping>
16        <servlet-name>Faces Servlet</servlet-name>
17        <url-pattern>/faces/*</url-pattern>
18    </servlet-mapping>
```

JSF Application directory structure

WEB-INF/web.xml
WEB-INF/faces-config.xml
WEB-INF/classes/`LoginFormBean.class`
`login.jsp`

Required Jars:

WEB-INF/lib/jsf-api.jar
WEB-INF/lib/jsf-ri.jar
WEB-INF/lib/jstl.jar
WEB-INF/lib/jsf-el.jar
WEB-INF/lib/standard.jar
WEB-INF/lib/commons-beanutils.jar
WEB-INF/lib/commons-digester.jar
WEB-INF/lib/commons-collections.jar
WEB-INF/lib/commons-logging.jar

Summary

- JSF: Server side UI component framework
- MVC
- Developing application in JSF

Reference

- http://www.jsfcentral.com/reading/index.html
- http://java.sun.com/j2ee/javaserverfaces/
- http://www.jcp.org/en/jsr/detail?id=127

Q&A

deepak.goyal@sun.com

vikas.varma@sun.com

Sun microsystems