Java Server Faces (JSF)

Table of Contents

Chapter 1

JSF Introduction

Chapter 1 Outline

- JSF TimeLine.
- What is JSF?
- Why JSF?
- What JSF looks like?
- Why we need frameworks/ frameworks types?
- Evolution of technologies
- JSF Architecture
- JSF Life cycle
- JSF Application Model
- A simple JSF Application

JSF TimeLine

- **2002:** JavaOne.
- **2004:** JSF 1.0 specification by the JSF Expert Group.
- **2006:** JSF 1.2 incremental release
  - Open source Frameworks:
    - Facelets,
    - Ajax4jsf,
    - JSF Templates,
    - Pretty Faces,
    - ICEFaces,
    - …..
- **2009:** JSF 2.0

What is JSF?

- A standard server-side Java web framework.
- It simplifies development by providing a component-centric approach to developing Java Web user interfaces.
- A set of Web-based GUI controls and associated handlers
- JSF provides many prebuilt HTML-oriented GUI controls, along with code to handle their events.

What is JSF? (cont'd)

- A device-independent GUI control framework
- JSF can be used to generate graphics in formats other than HTML, using protocols other than HTTP.
- Extensible component model, and a large number of third-party components have become available.
- Companies involved : IBM, Oracle, BEA systems, Borland,…

Why JSF?

1. Standard.
2. Easy to use.
3. MVC for web applications.
4. Support for client device independence.

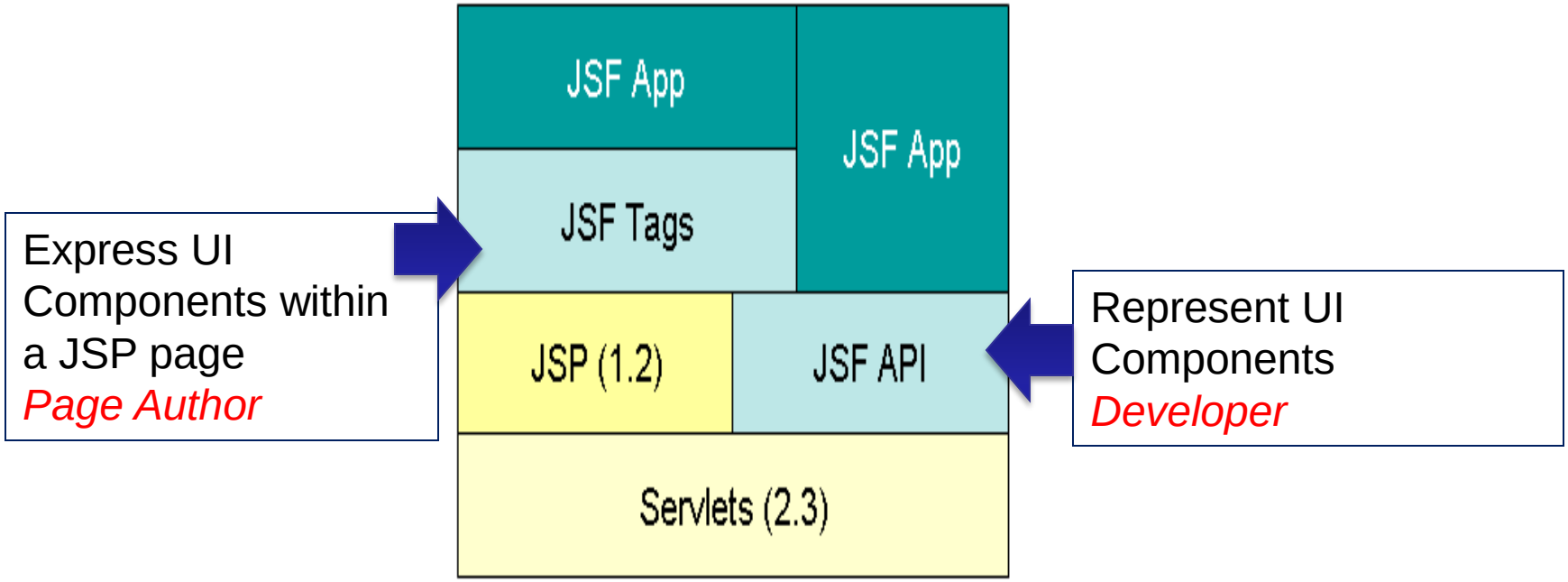
Why JSF? (cont'd)

5. Huge vendor and industry support.
6. Can work with any presentation technology including JSP.
7. Extendable Component and Rendering architecture.
8. UI elements is stateful objects on the server.

What JSF looks like?

- JSF is a **Rapid Application Development** (RAD)

- **The technology under the hood** :



Why do we need frameworks?

- **Why do we need frameworks?**
  - To help to carry some tedious tasks from the developer , and make it automatic which make it easy to scale

Types of Frameworks

- **Foundation framework:**
  - Form processing ,
  - Page management.
  - Type conversion,
  - Error handling,
  - Enforcing MVC model,
  - Not masking the fundamental request / response nature of HTTP.
  - Example ; Struts

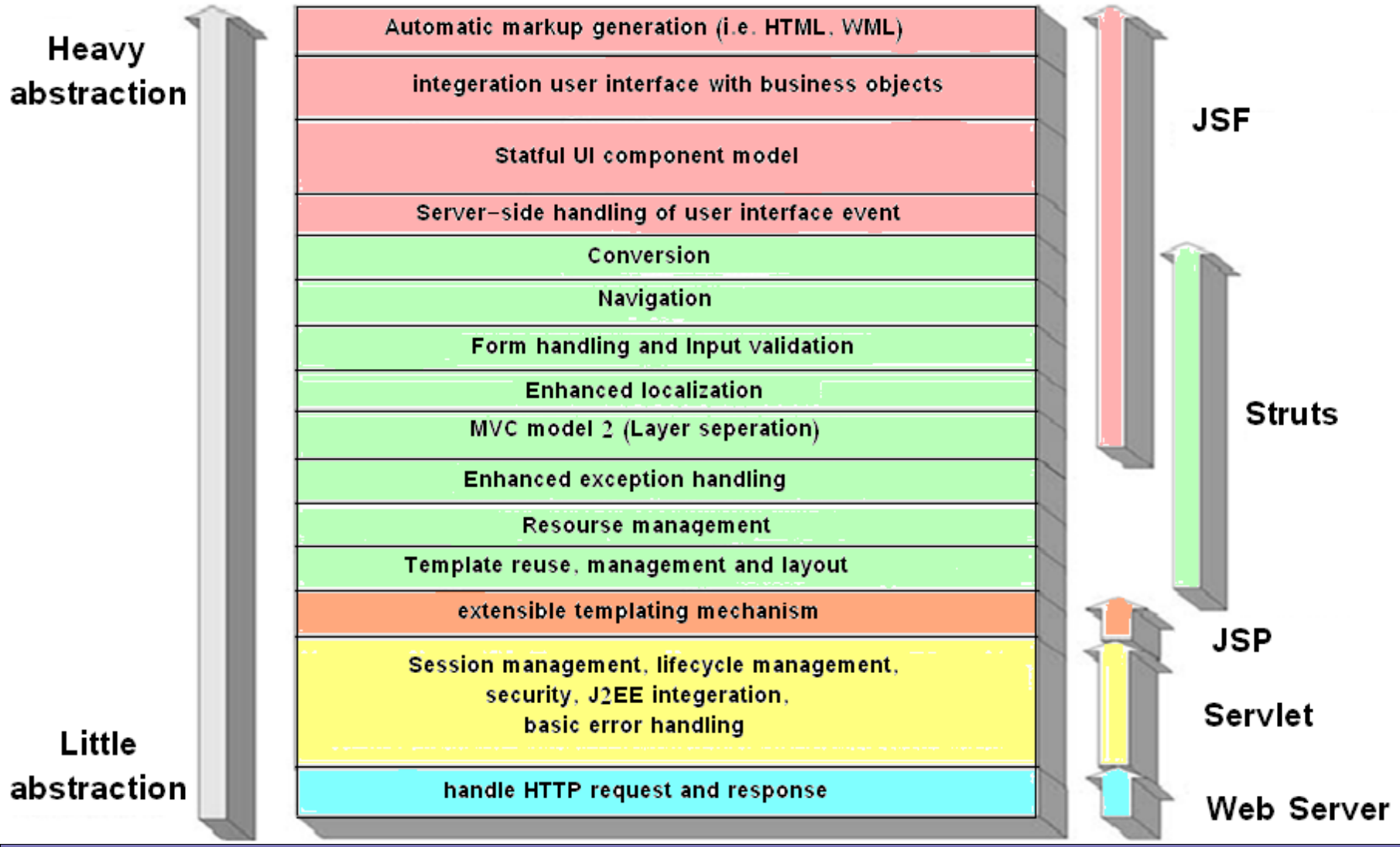
Types of Frameworks (Cont'd)

- **UI framework:**
  - Form processing ,
  - Page management.
  - Type conversion,
  - Error handling,
  - Enforcing MVC model,
  - Masking the fundamental request / response nature of HTTP.
  - Component based web application.
  - Example : ADF, JSF

You can have a role

- **Several types of users can benefit from JSF:**
  - Page authors:
    - Use markup languages, such as HTML
    - Use JSP tag library for expressing JSF UI Components.
  - Application developers:
    - Write the model objects and event handlers.
  - Component developers:
    - Create custom components based on the JSF components.
  - Tools vendors:
    - Provide tools that simplify the development of multi tier, web-based applications.
  - Application Server vendors:
    - Provide a runtime environment of Application Servers that can deploy multi tier, web-based applications.

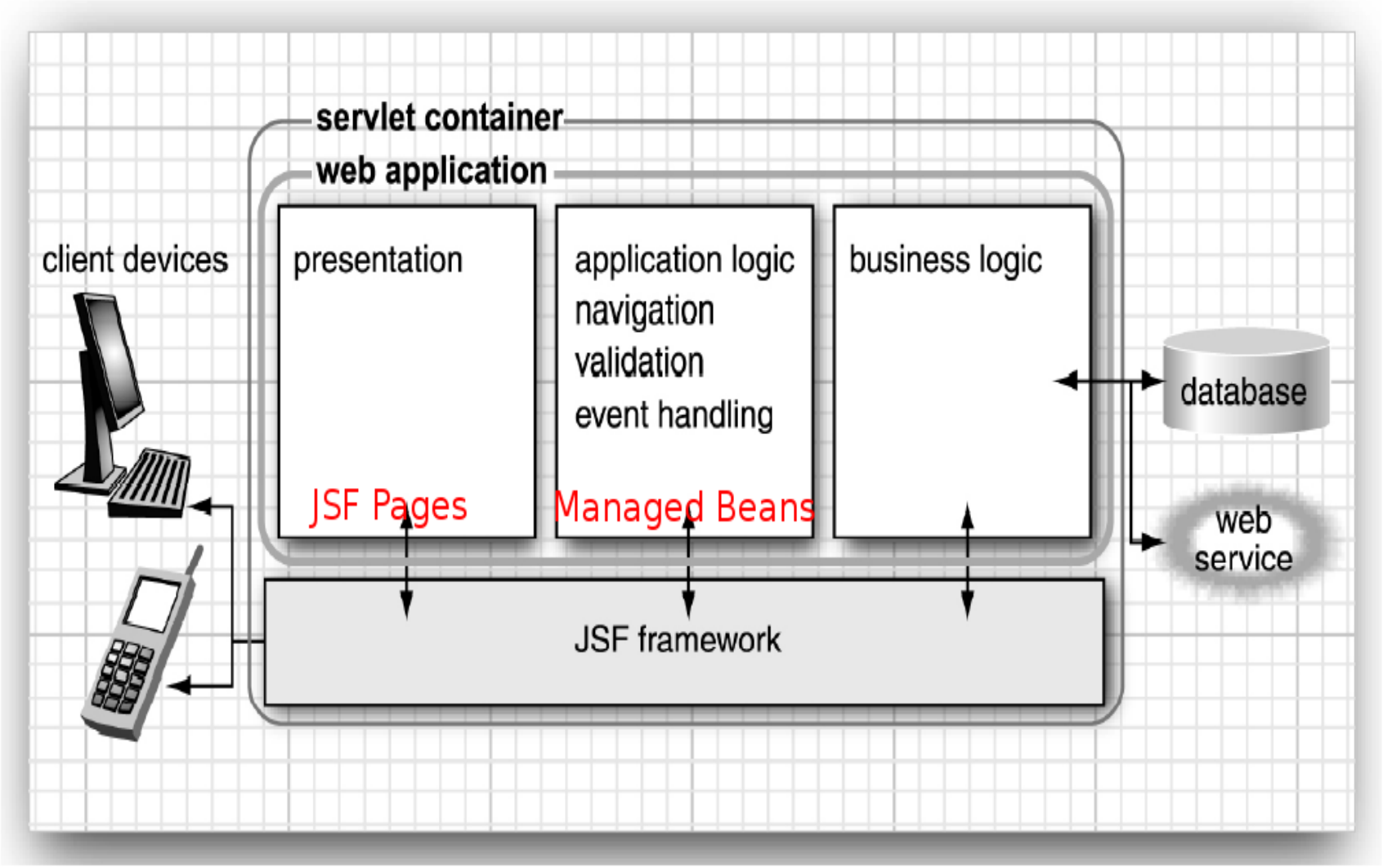
Evolution of Technologies



JSF and other Technologies

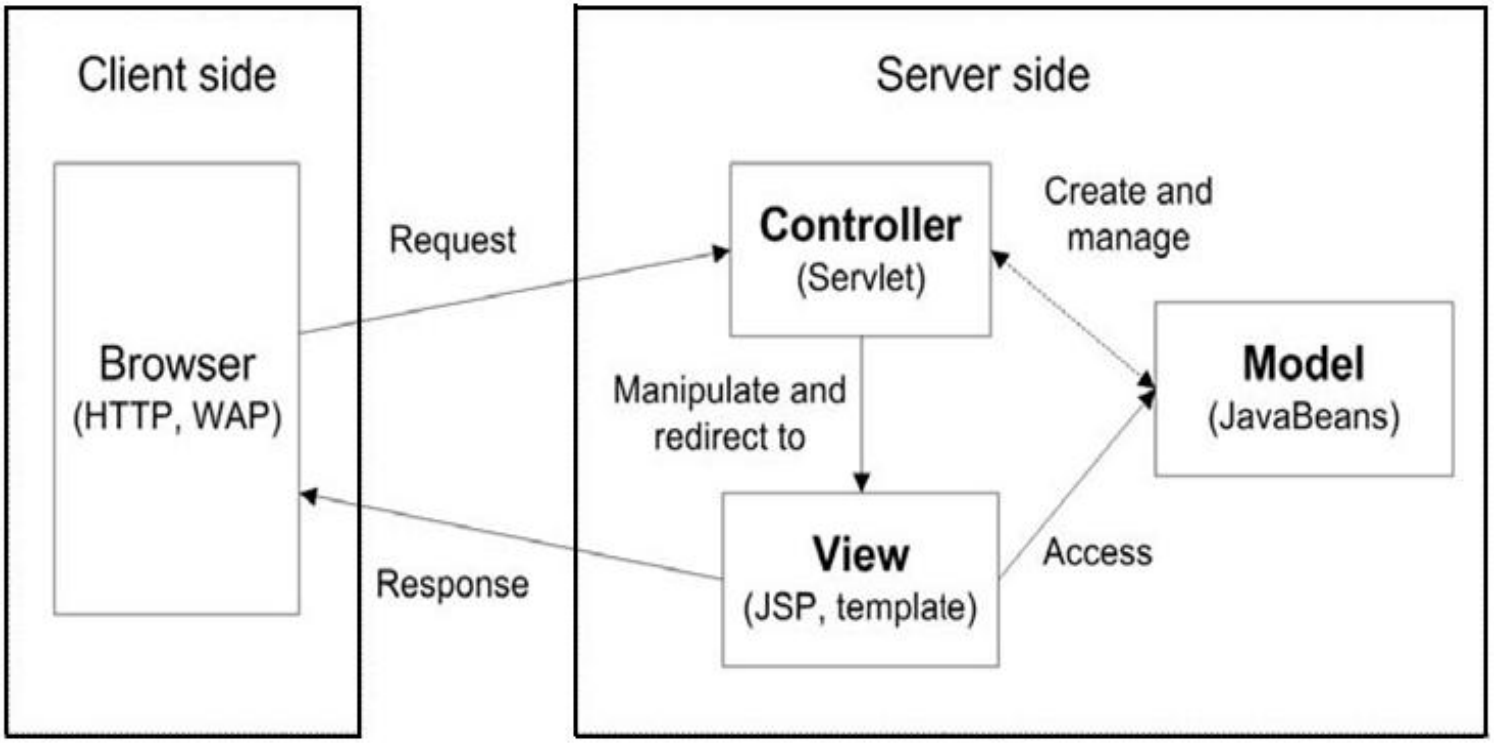
- **Try to define the relation between JSF and :**
  - Swing
  - Servlets & JSP& Java beans
  - AJAX
  - Portals
  - Struts

JSF Architecture

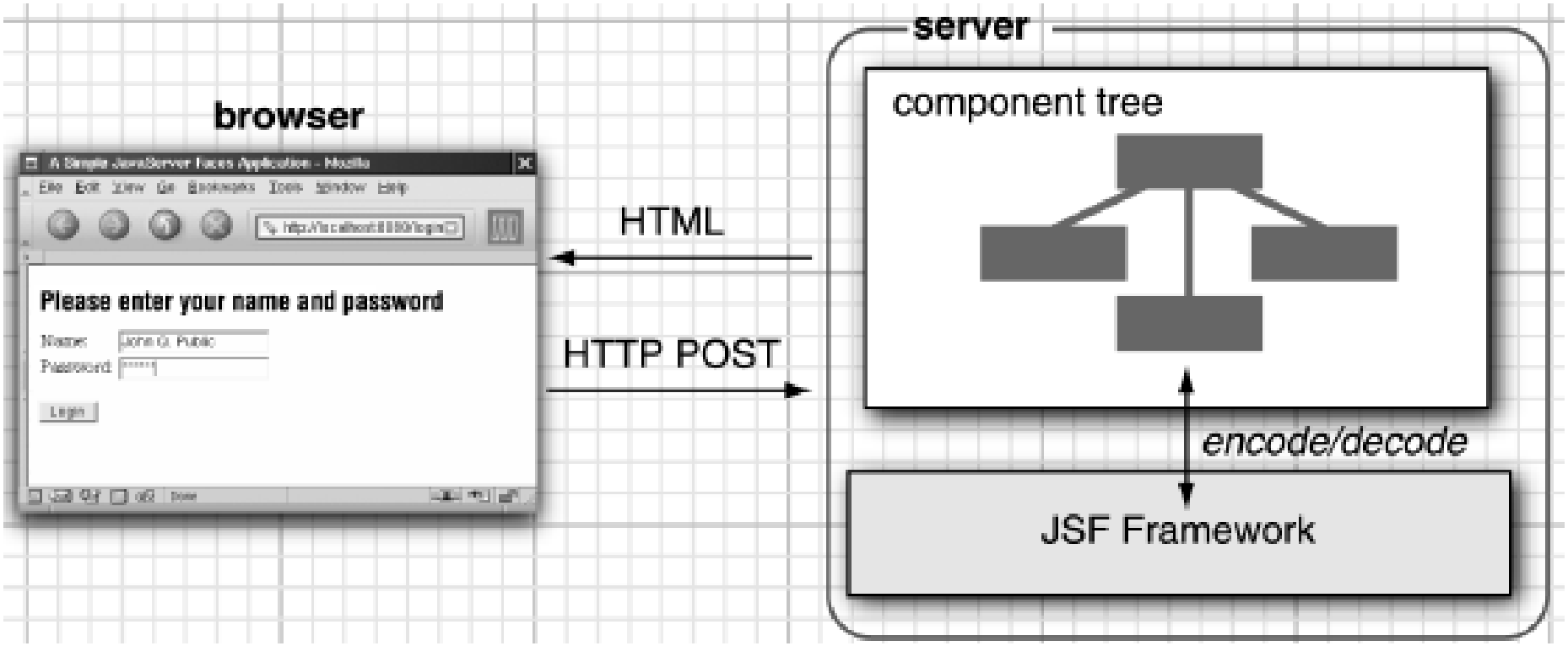


JSF Architecture (Cont'd)

- **JSF Framework Services :**
  - Model-view-controller architecture



Behind the scenes



Request Processing Lifecycle

- **Types of the request:**
  1. Initial Request:
     - A user requests the page for the first time.
     - Encoding Requests
     - Lifecycle only executes the restore view and render response phases.
  2. Post back:
     - A user submits the form on a page that was previously loaded into the browser.
     - Decoding Requests
     - Lifecycle executes all phases.

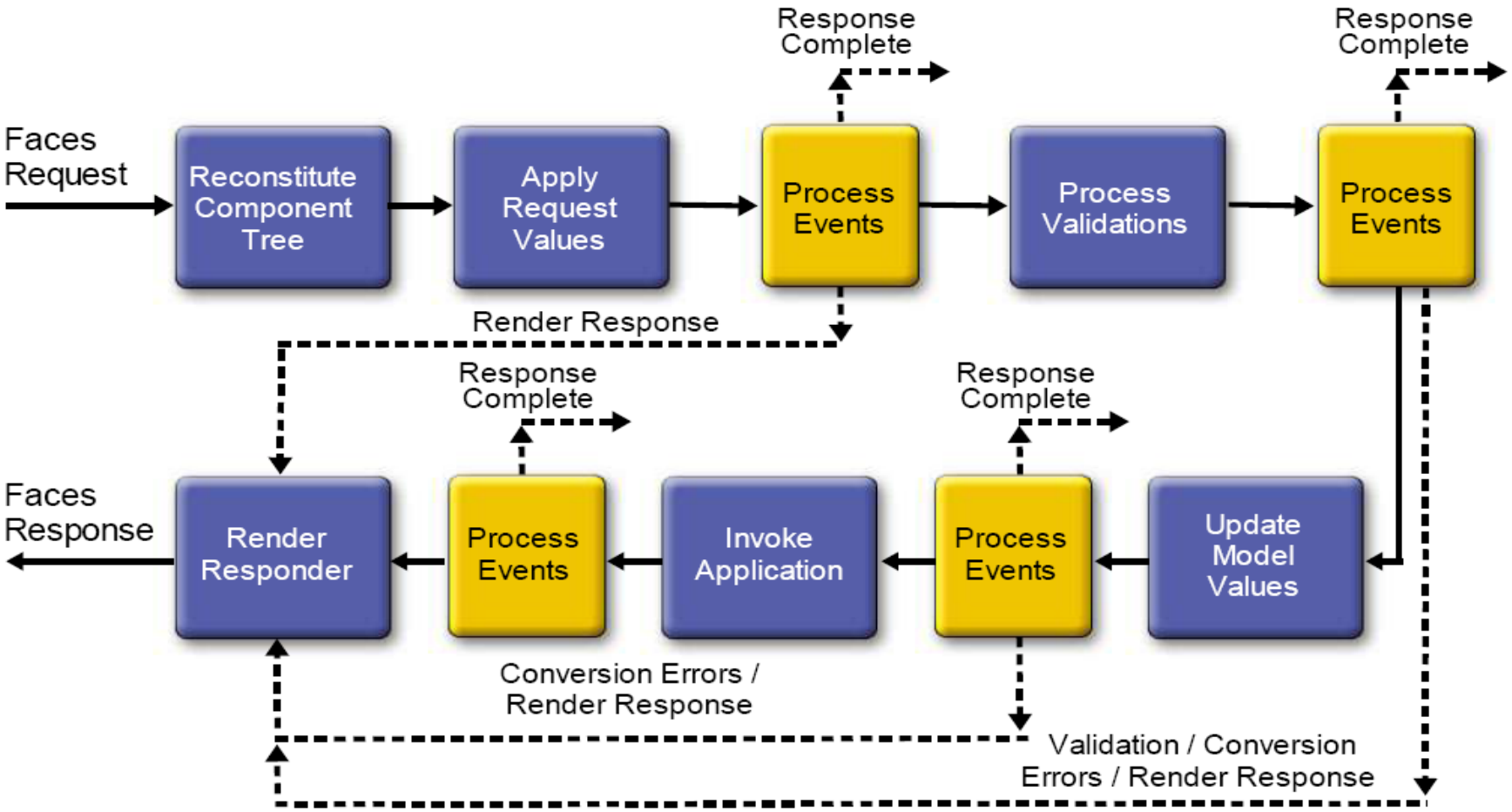
Lifecycle of JSF Page

- A JSF page is represented by a tree of UI components, called a view.

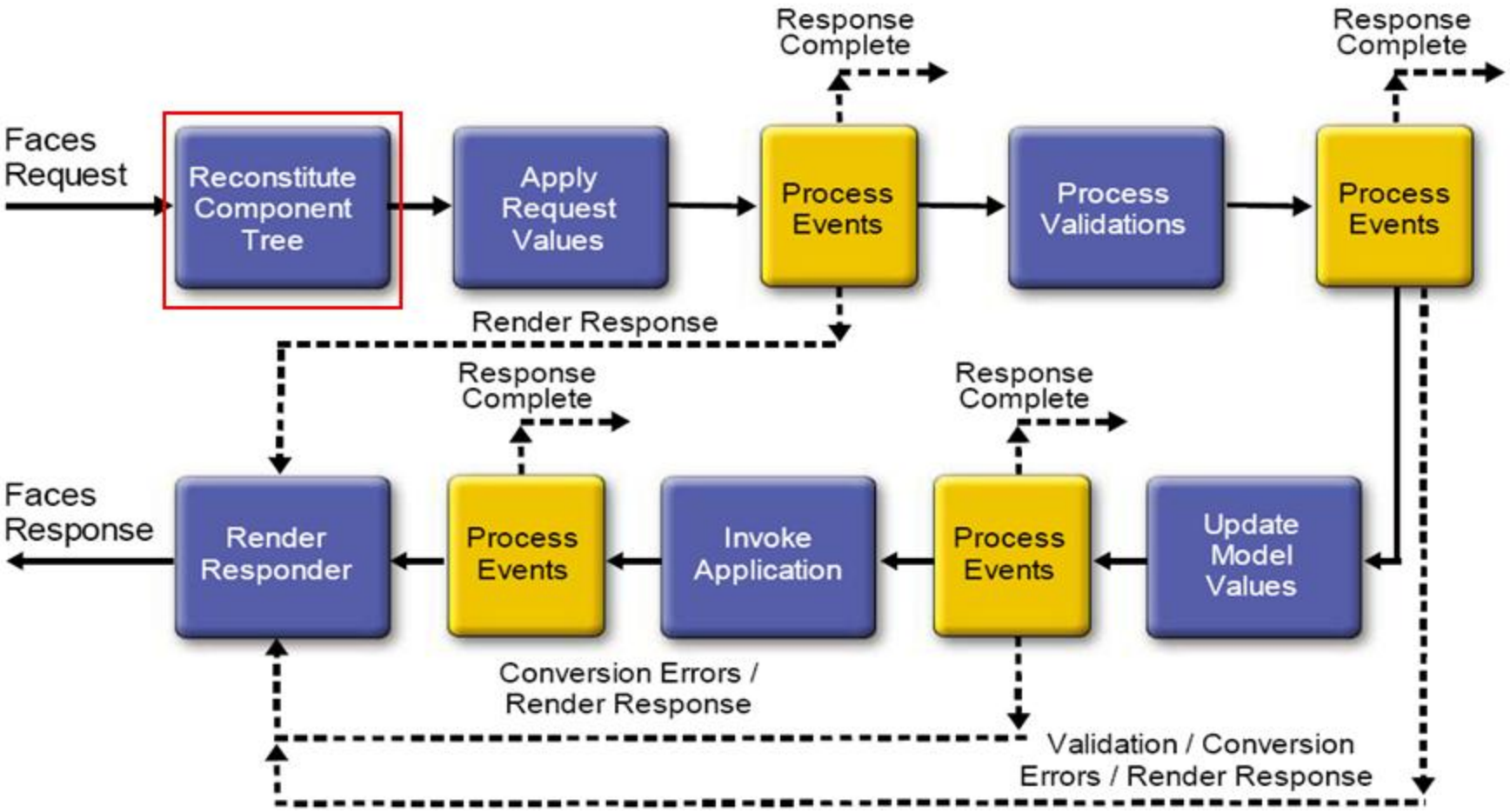
- When a client makes a request for the page, the lifecycle starts.
- During the lifecycle, JSF implementation must build the view while considering state saved from the previous post back.

- When the client performs a post back of the page, JSF implementation must perform lifecycle steps mainly :
  - conversion **- validation**

Lifecycle of JSF Page(cont'd)



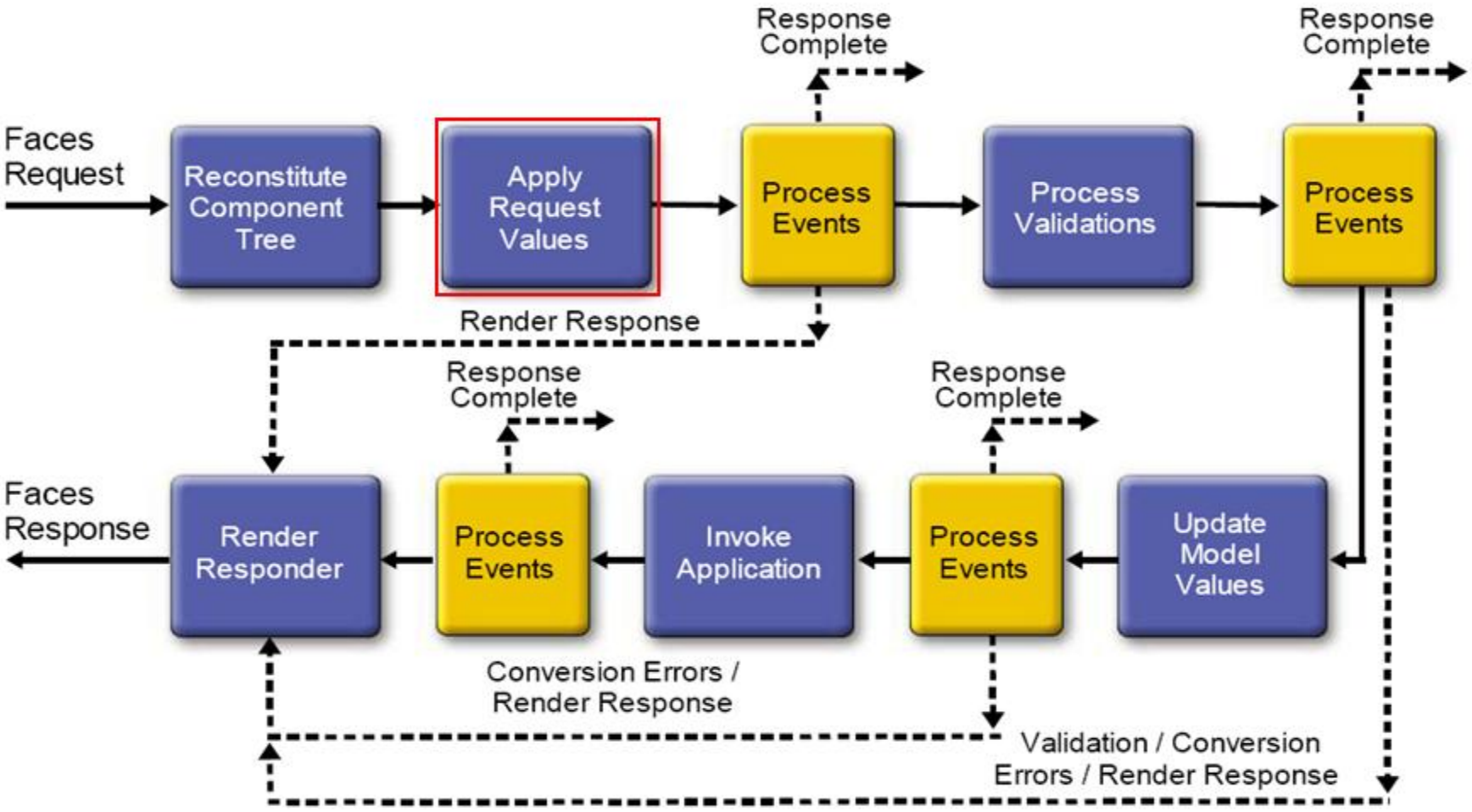
Lifecycle of JSF Page(cont'd)



Lifecycle of JSF Page(cont'd)

- **Reconstitute Component tree:**
  - On the first (non-postback) request it simply passes through two phases, RESTORE_VIEW and RENDER_RESPONSE.
  - This means it just creates the UI component tree and saves it in the FacesContext,
  - and then just renders it to the client.
  - You can then see the page in the browser.

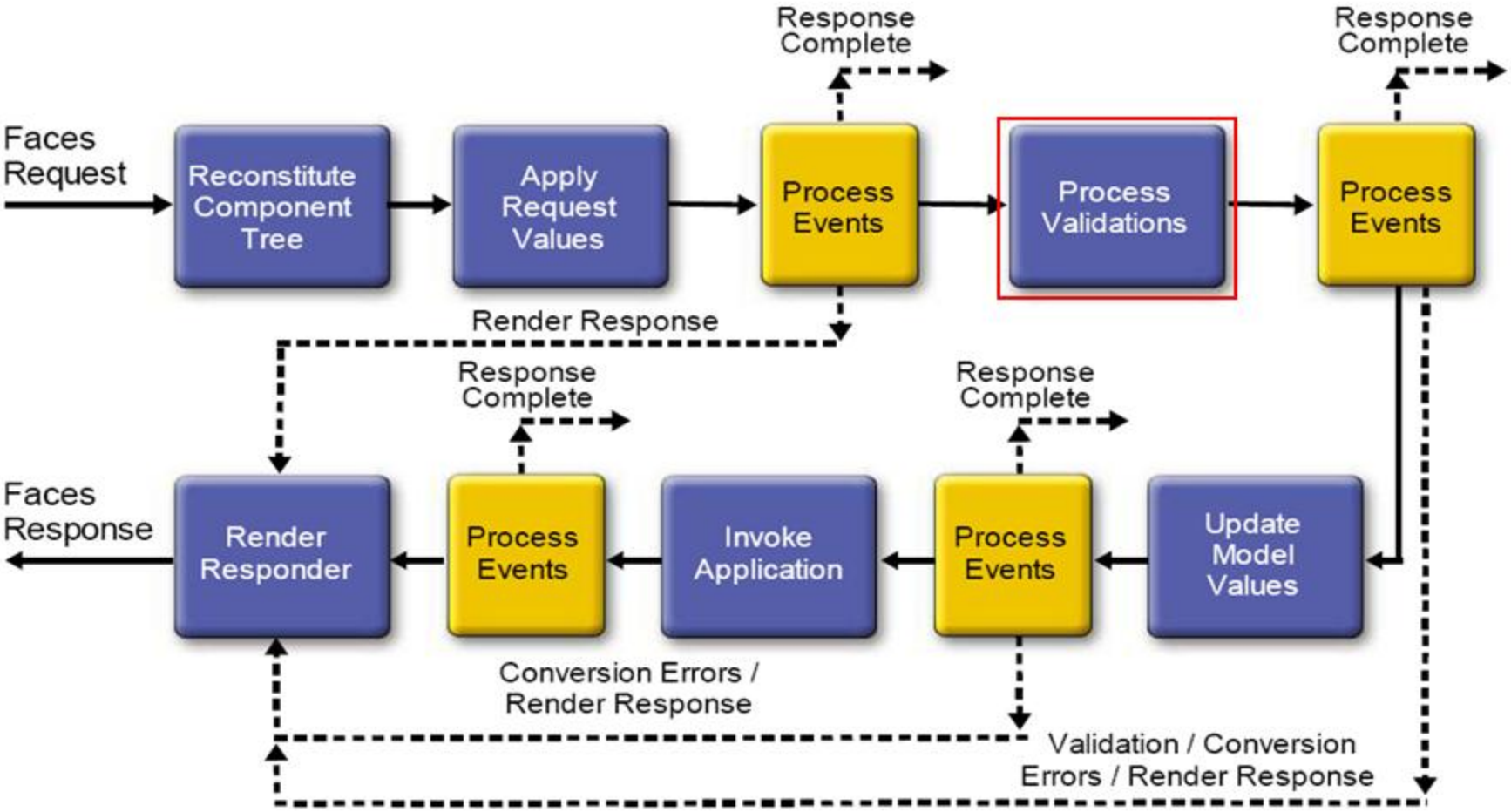
Lifecycle of JSF Page(cont'd)



Lifecycle of JSF Page(cont'd)

- **Apply Request Values:**
  - implementation iterates over the component objects in the component tree.
  - Each component object checks which request values belong to it and stores them.
  - The values stored in the component are called "local values".

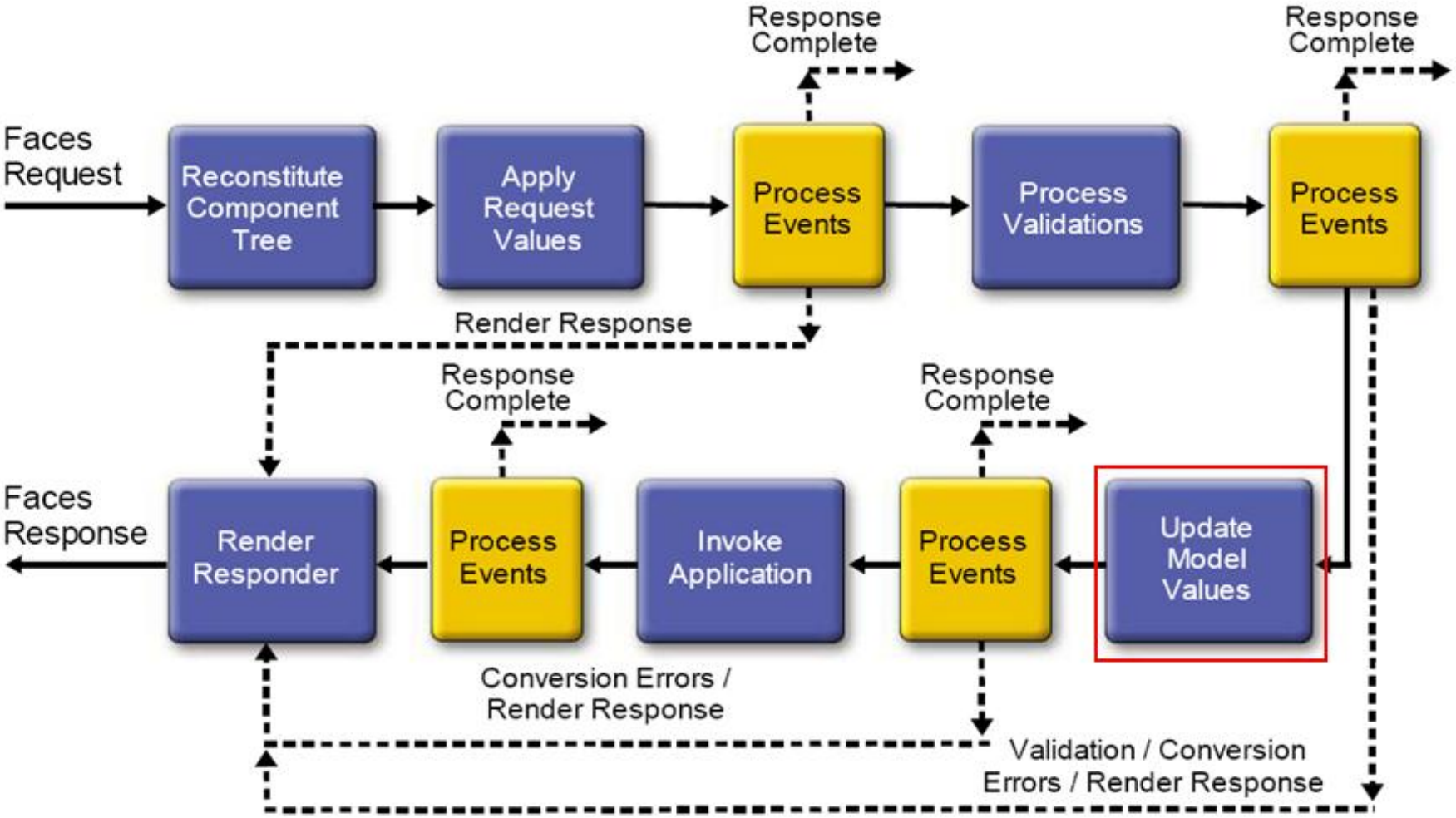
Lifecycle of JSF Page(cont'd)



Lifecycle of JSF Page(cont'd)

- **Process Validations:**
  - performs correctness checks on the local values
    - (asks each one to validate itself).
  - If the validation fails, the lifecycle advances directly to the Render Response phase to render the page with the error messages.

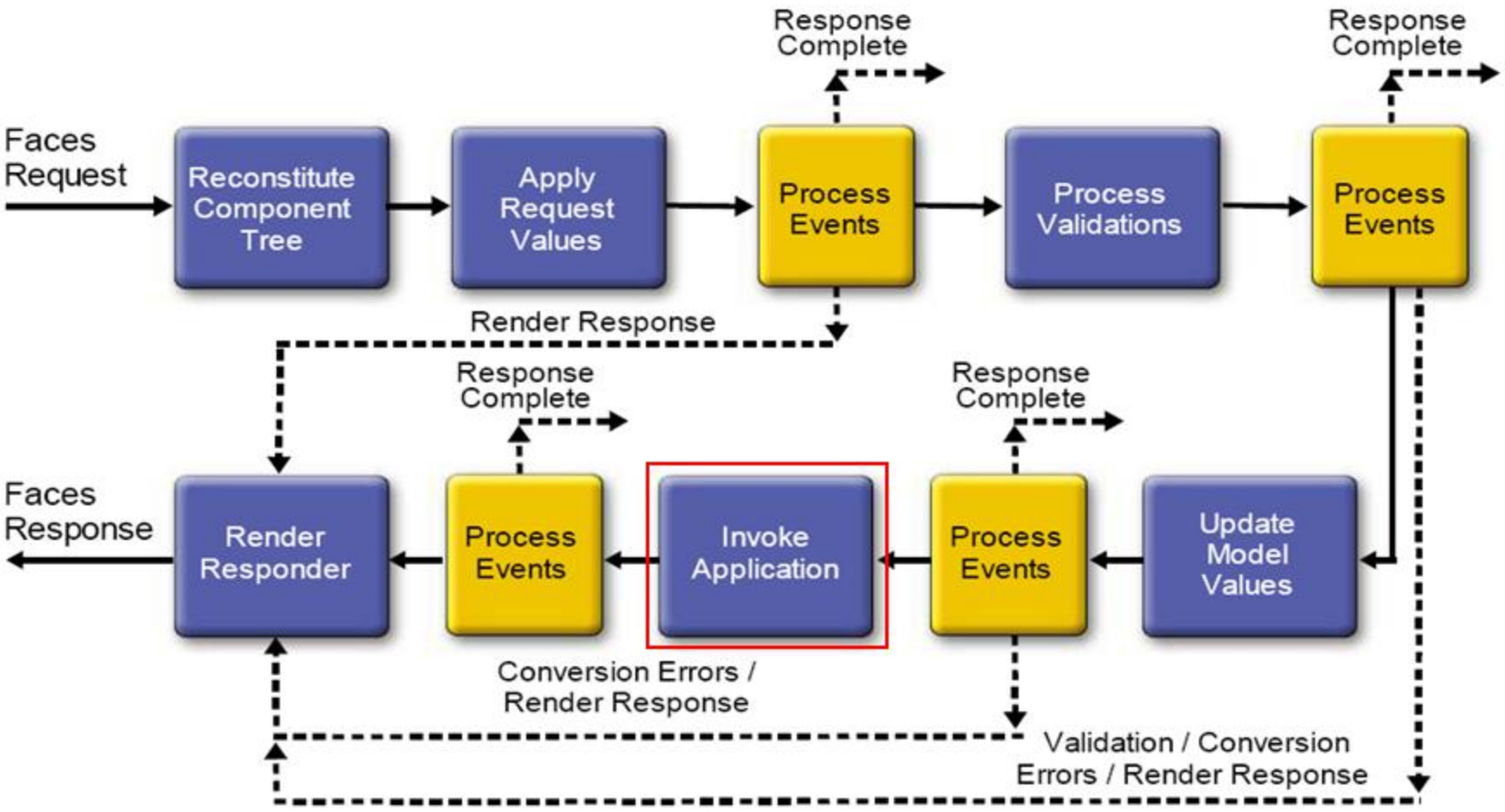
Lifecycle of JSF Page(cont'd)



Lifecycle of JSF Page(cont'd)

- **Update Model Values:**
  - Updates all the values of backing beans or model objects associated with Local Values.
  - Only input components that have valueRef expressions will be updated.
  - If the conversion fails, the lifecycle advances directly to render Response so that the page is re-rendered with errors displayed.

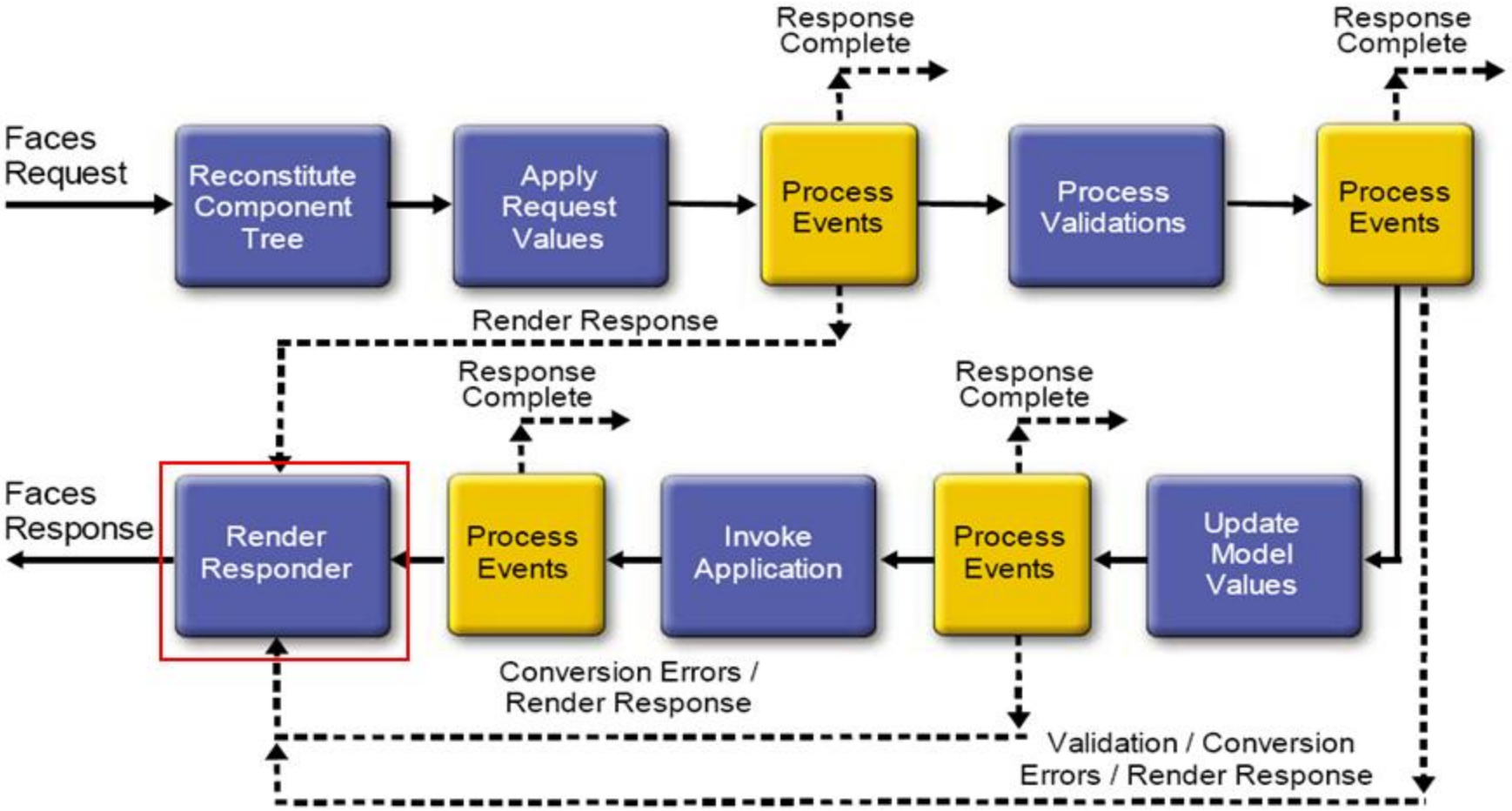
Lifecycle of JSF Page(cont'd)



Lifecycle of JSF Page(cont'd)

- **Invoke Application:**
  - Calls any registered action listener.
  - The listener passes the outcome to the default NavigationHandler.
  - The NavigationHandler matches the outcome to the proper navigation rule to determine what page needs to be displayed next.

Lifecycle of JSF Page(cont'd)



Lifecycle of JSF Page(cont'd)

- **Render Response:**
  - Displays the selected view.
  - After the content of the tree is rendered, the tree is saved so that subsequent requests can access it and it is available to the Reconstitute Component Tree phase.

Changing the lifecycle

- **immediate** property
- ***UICommand components:***
  - action is called immediately. No validation or model update.
- ***Only with UIInput components:***
  - No conversion and validation of other components!

JSF Application Model

JSF Application Model can be broken into

1. **User Interface model:**
   - UI Component class , Renderer model
   - Conversion model , Validation model
   - Event & listener model

2. **Navigation model**

3. **Backing Bean Model**

JSF Application Model (cont'd)

1. User Interface model:

i. UI Components:

- Are a stateful objects.
- Components are organized in a tree structure.
- There are two important classes in the UI model:
  - The UIViewRoot component class represents the root of the tree of all components for a particular page.
  - The FacesContext class serves as the access point for per-request information.

JSF Application Model (cont'd)

1. User Interface model:

ii. Renderer model:

- Responsible for displaying a UI component.
- Renderer can be work with one or more UI components.
- UI component can be associated with many different renderers.

JSF Application Model (cont'd)

1. User Interface model:

iii. Conversion model:

- Any Component's data has two views:
  - Model view: In the bean .
  - Presentation view: In the page.
- Converter used to convert the data between the model view and presentation view.
- UI component can be associate with a single converter.

JSF Application Model (cont'd)

1. User Interface model:

iv. Validation model :

- Validator is responsible for ensuring that the value entered by a user is acceptable.
- One or more Validators can be associated with a single UI component.
- Three ways for validation:
  - UIcomponent level
  - Validation methods in backing beans
  - Validation classes

JSF Application Model (cont'd)

1. **User Interface model:**

   v. Event & listener model:

   - JSF uses the JavaBeans event/listener model. (Like swing).

   - An Event object identifies the component that generated the event and store info about the event.

   - An application must provide a listener and register it on the component .

JSF Application Model (cont'd)

2. **Navigation model:**
   - **Navigation** is a set of rules for choosing the next page to be displayed after event fired.
   - The selection of the next page is determined by:
     - The page that is currently displayed.
     - The action method invoked of the component that generated the event.
     - An outcome string that was returned by the action method.

JSF Application Model (cont'd)

3. Backing Bean Model:

- Backing beans are objects that represent data and state of UI components.

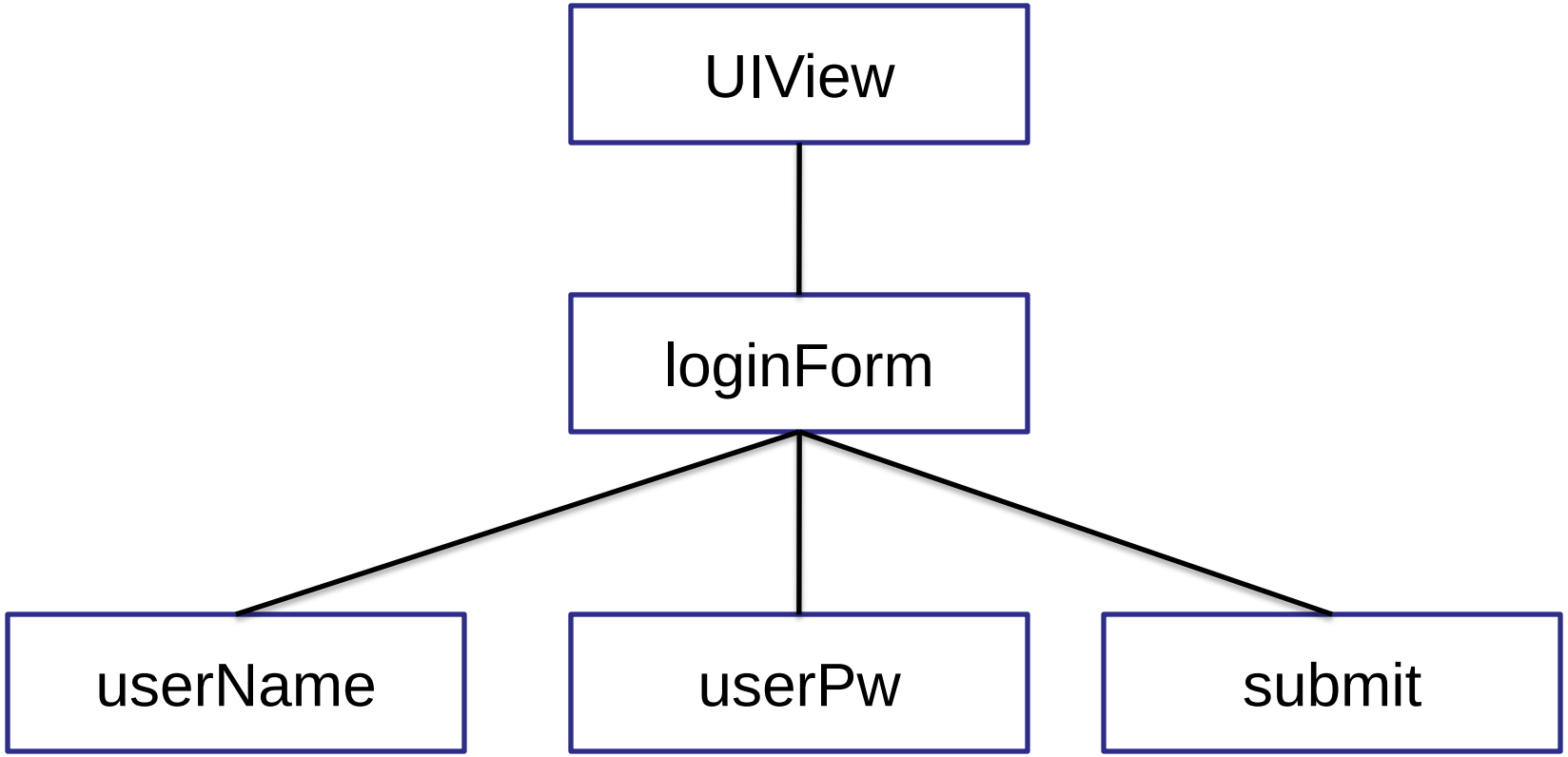
- It contains:
  - UI component properties.
  - Event handling methods.
  - Validation methods.
  - Converter code.

A simple JSF Application

- index.xhtml (Design View)

A simple JSF Application (cont'd)

- index.xhtml (JSF Pages View)



A simple JSF Application (cont'd)

- index.xhtml (JSF Pages View)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns=http://www.w3.org/1999/xhtml xmlns:h="http://java.sun.com/jsf/html">
        <h:head>
                <title>Welcome</title>
        </h:head>
        <h:body>
        <f:view>
                <h:form>
                        <h3>Please enter your name and password.</h3>
                        <table>
                                <tr>
                                        <td>Name:</td>
                                        <td> <h:inputText value="#{user.name}"/> </td>
                                </tr>
                                <tr>
                                        <td>Password:</td>
                                        <td> <h:inputSecret value="#{user.password}"/> </td>
                                </tr>
                        </table>
                        <p> <h:commandButton value="Login" action="welcome"/> </p>
                </h:form>
                </f:view>
        </h:body>
</html>
```

Standard HTML
JSF tags

A simple JSF Application (cont'd)

- welcome.xhtml (Design view & JSF Pages View)


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=http://www.w3.org/1999/xhtml
xmlns:h="http://java.sun.com/jsf/html">

    <h:head>
        <title> Welcome </title>
    </h:head>
    <h:body>
        <h3> Welcome to JavaServer Faces, #{user.name}!
    </h3>
    </h:body>

</html>
```

A simple JSF Application (cont'd)

- UserBean.java

```
import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean(name="user")
@SessionScoped

public class UserBean implements Serializable {
        private String name;
        private String password;

        public String getName() { return name; }
        public void setName(String newValue) { name = newValue; }

        public String getPassword() { return password; }
        public void setPassword(String newValue) { password = newValue; }
}
```

A simple JSF Application (cont'd)

- **Configuration Files:**
  - web.xml & beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app …….>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
  </servlet-mapping>

  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
</web-app>
```

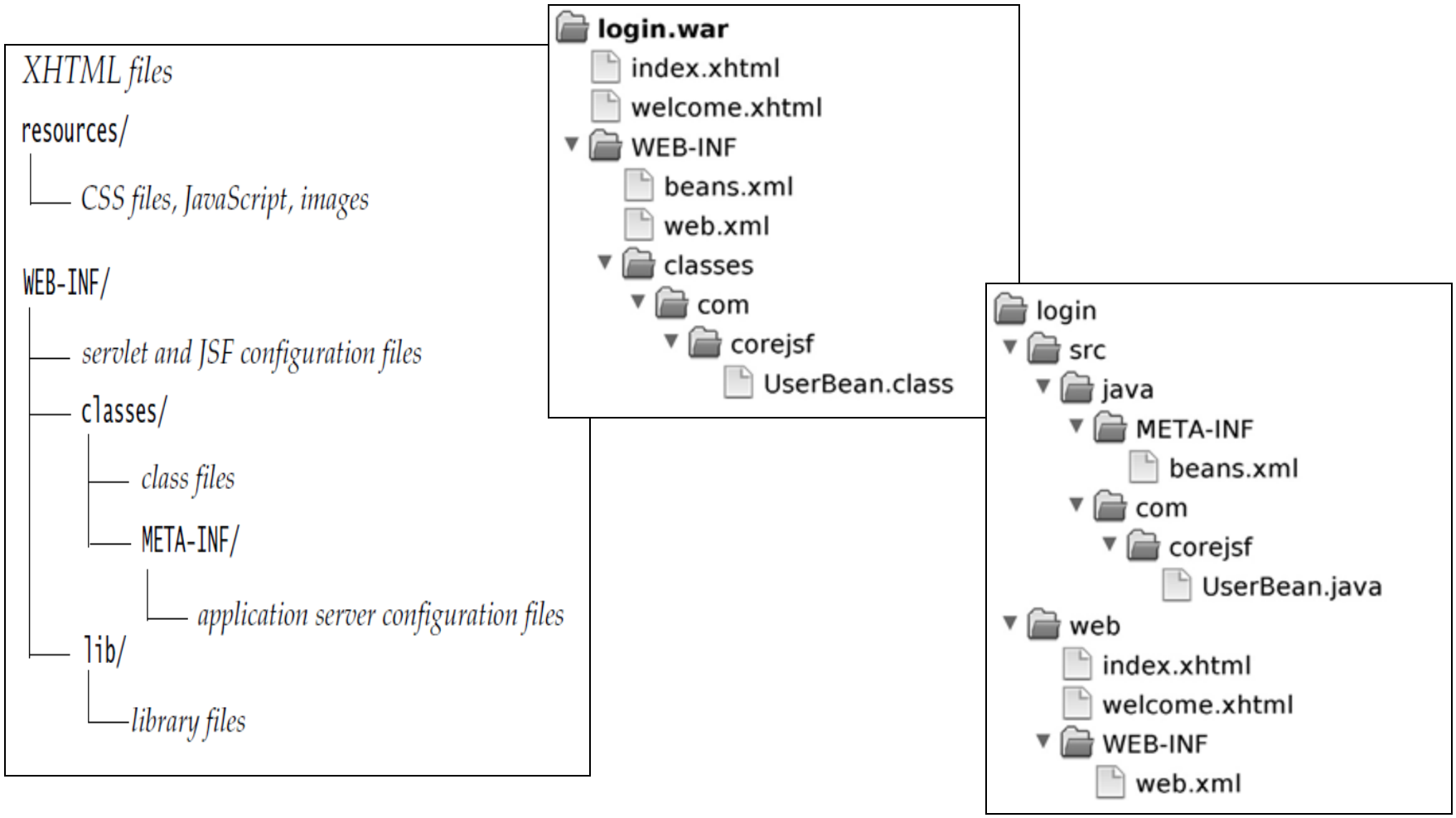
A simple JSF Application (cont'd)

- **Configuration Files** (Cont'd):
  - faces-config.xml:
    - For additional configuration parameters (e.g.navigation rules, converters, validators, render kits,…)

```
<?xml version="1.0"?>

<!--
 Copyright 2003 Sun Microsystems, Inc. All rights reserved.
 SUN PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
-->

<!DOCTYPE faces-config PUBLIC
  "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.0//EN"
  "http://java.sun.com/dtd/web-facesconfig_1_0.dtd">

<faces-config>

  <application>
    <locale-config>
      <default-locale>en</default-locale>
      <supported-locale>de</supported-locale>
      <supported-locale>fr</supported-locale>
      <supported-locale>es</supported-locale>
    </locale-config>
  </application>
```

A simple JSF Application (cont'd)

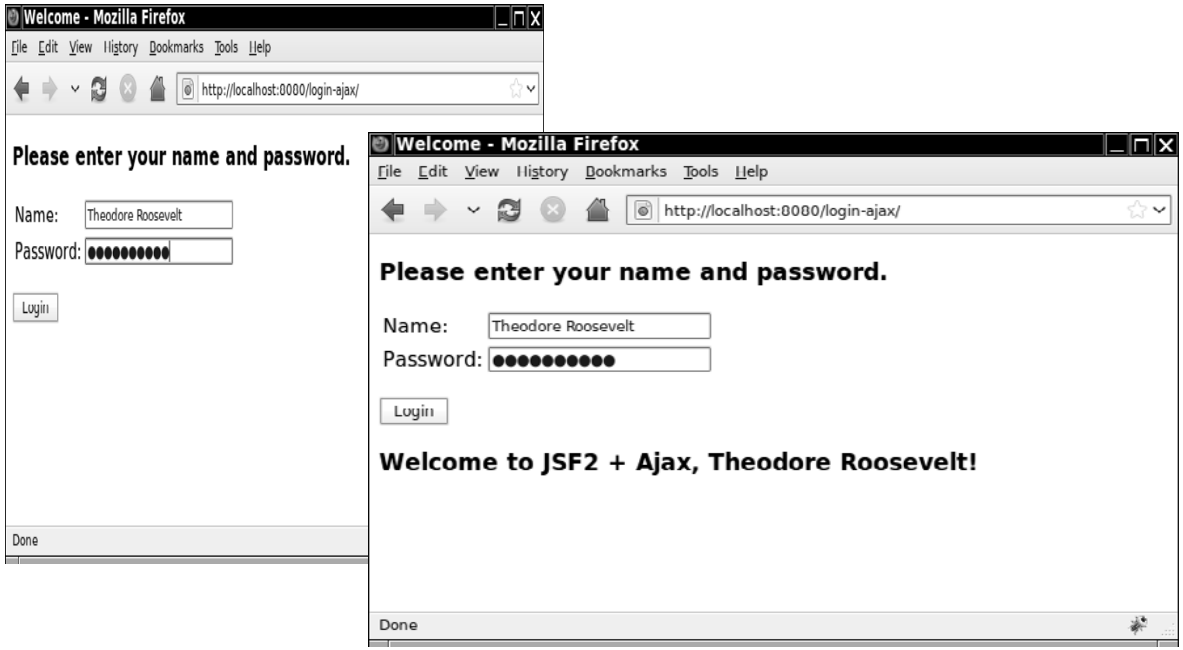
- **Directory Structure:**



A simple JSF Application (cont'd)

- **Welcome JSF 2.0 + AJAX**

  JSF 2.0 uses Ajax without having to understand the considerable complexities of the Ajax communication channel



A simple JSF Application (cont'd)

- **Welcome JSF 2.0 + AJAX** (Cont'd)
  - index.xhtml (JSF Pages View)

```
<h:form prependId="false" >
    <h3>Please enter your name and password.</h3>
    <table>
        <tr>
            <td>Name:</td>
            <td> <h:inputText value="#{user.name}“id="name" />
            </td>
        </tr>
        <tr>
            <td>Password:</td>
            <td> <h:inputSecret value="#{user.password}“
            id=“password”/> </td>
        </tr>
    </table>
    <p> <h:commandButton value="Login”>
            <f:ajax execute="name password" render="out"/>
        </h:commandButton>
    </p>
    <h3><h:outputText id="out" value="#{user.greeting}"/></h3>
</h:form>
```

A simple JSF Application (cont'd)

- **Welcome JSF 2.0 + AJAX** (Cont'd)
  - UserBean.java

```
public class UserBean implements Serializable {
                                    :
    private String greeting;

    public String getGreeting() {
    if (name.length() == 0) return "";
    else return "Welcome to JSF2 + Ajax, " + name + "!";
    }
    }
```

Lab Exercise

Assignments

- Getting familiar with Net Beans IDE.
- Make your first Website design :
  - Just Design website pages:
    - Home.
    - Login.
    - Register.
    - Search.
    - Vote.
  - (Today)Make Login page + add AJAX to it