Java Server Faces, JSF, Basics

JSF Slides #1

Component Based Approach

Why should I have to learn all this low level?
- HTTP, Url's, request, response, JavaScript, etc. … ?

I'm a (Java) programmer, I would like to program as usual!
- Give me **components** like JButton, JPanel, … and an event handling mechanism, like actionListeners, …

… **Java Server Faces** will try to make you happy…

Java Server Faces

A **server-side** user interface component technology for Java web applications (Think: Swing for the web)

Defines how to construct the server-side view (a **page description language, Facelets**)

Implement event/control parts as **managed beans** with **listener-methods**

Renders the server side GUI for the client (browser), using **a render kit**.
- Mandatory HTML-render kit, others possible...

A **navigation model** (default full page loads)

Specification: JSR 314, JSF 2.1

Server side GUI

In JSF the GUI will be represented as a **component tree** on the server, "the view"

Root of tree: javax.faces.component.UIViewRoot

All objects in tree extends javax.faces.component.UIComponent.
Examples (super :> sub)
- UIComponent :> UIOutput :> HtmlOuputLabel
- UIOutput :> UIInput :> HtmlInputText

Component categories
- **ActionSource**, anything producing events (button)
- **ValueHolder**, anything holding data (label)
- **EditableValueHolder**, anything holding editable data (text field)

View Id's

Server application have many clients
- Must handle many views (trees)
- One for each client

At request application must match client and view (lookup matching component tree), **a postback request...**
- Clients and views have matching id's
- Possible to access view id: viewRoot.getViewId()

...or create a new view tree, **the initial request**

JSF Event Model

Faces events only three (compare Swing)

- ActionEvent, triggered by buttons, some links
- ValueChangeEvent, drop downs etc
- SystemEvent, possible for components to publish and (un) subscribe to system events, … probably not used by us
- .. more

Attached objects

Components in tree can have attached objects, i.e. **non UIComponents** attached to UIComponents
- Converters, converting String ↔ Object, ... we normally don't
- Validators, validating data, ... we normally don't
- Beans, with listener methods and properties, ... **we do, a lot!**

JSF Basic Setup

A Facelets "page" with JSF-elements (tags)

- A few element libraries <f: ... />
- Elements specify the server side components (and more)
- Elements normally inside a JSF-form (else no submit)

A managed bean (Java class) with listener method(s), to be continued...

- A few different "types" of listener methods

Connect elements on page (later server side components) to listener method(s)

- Element attributes specifies which object, which method
- Let listener method(s) call model

The JSF Request Cycle

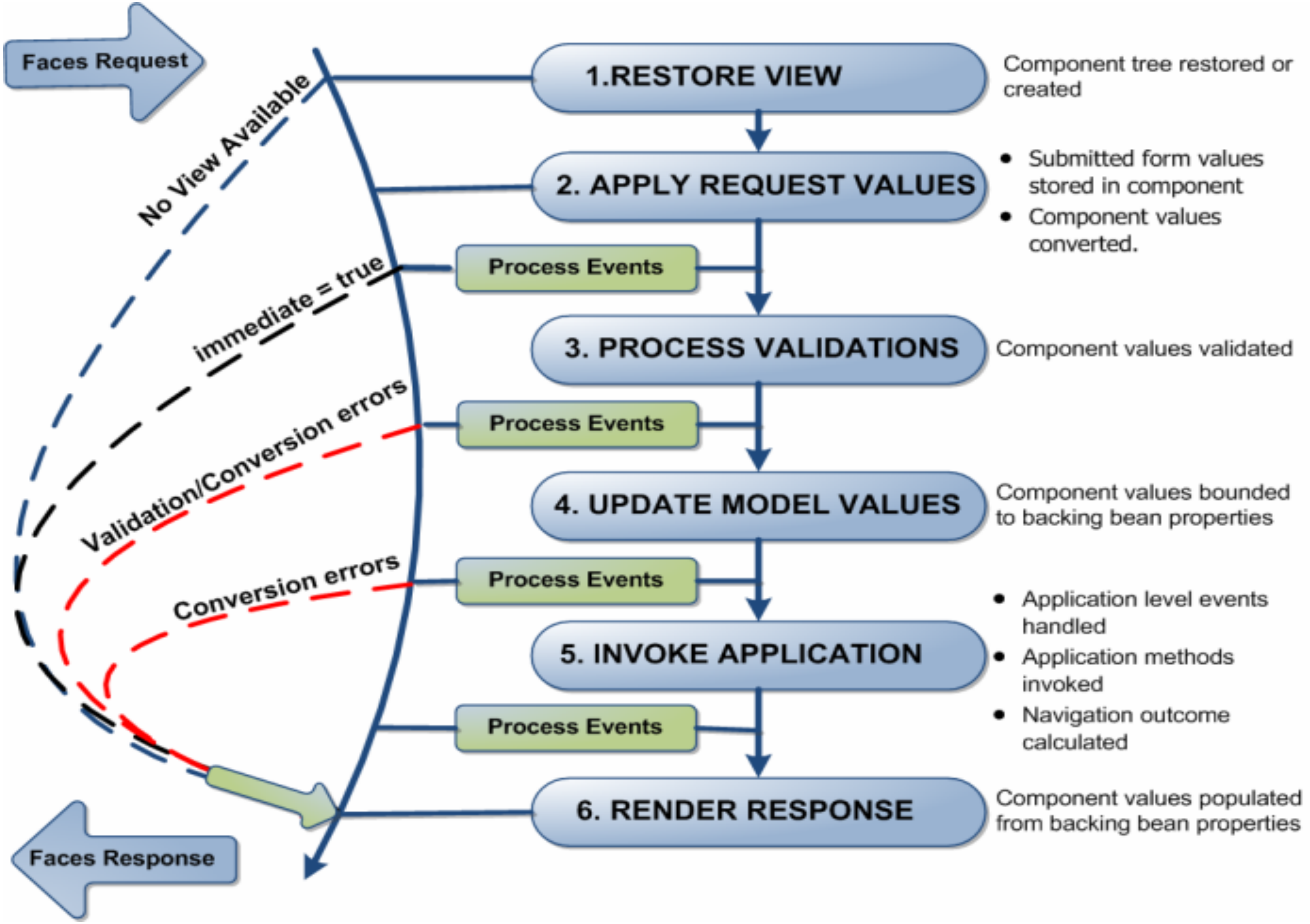
In Swing the event thread handles everything
- Just a method call inside the JVM

In JSF: GUI (image of server side view tree) and components (objects in tree) separated
- Have a network in between (a stateless protocol)
- Events will emanate from client GUI to objects on server side!

Must be handled in an ordered fashion. The JSF **Request Cycle** will handle this ...

The cycle is "ridiculously" customizable, ... we don't

Request Cycle Details



1.RESTORE VIEW

Component tree restored or created

2. APPLY REQUEST VALUES

- Submitted form values stored in component
- Component values converted.

3. PROCESS VALIDATIONS

Component values validated

4. UPDATE MODEL VALUES

Component values bounded to backing bean properties

5. INVOKE APPLICATION

- Application level events handled
- Application methods invoked
- Navigation outcome calculated

6. RENDER RESPONSE

Component values populated from backing bean properties

Conversion

Submitted values converted and stored in server side GUI components (in tree), conversion mostly automagically

If more specific needs, JSF Standard converters for
- DateTime, Number, Boolean, Byte, Character, Double, Float, Integer, Long, Short, BigDecimal, BigInteger

```
<!-- Using h- anf f- tags from JSF -->
<h:inputText id="date" value="#{testBean.date}" required="true">
    <f:convertDateTime type="date" dateStyle="long"/>
</h:inputText>
```

Custom Conversion

Customization: Converting phone numbers

```
@FacesConverter(value = "pnumbConverter")
public class PhoneNumberConverter implements Converter {
  public Object getAsObject(FacesContext context, UIComponent
                             component, String value) {...}

  public String getAsString(FacesContext context, UIComponent
                            component, Object value) {...}
}
```

Usage

```
 <h:inputText value="..." ....>
    <f:converter converterId="pNumbConverter"/>
</h:inputText>
```

Validation

Classes (objects) handling validation of incoming data

- Customization possible
- We don't, we use CDI upcoming ...

Request Cycle and HTTP Requests

Post
- Will trigger a full cycle
- Except if using attribute "immediate" in <h: ... /> (useful for "Cancel")

Get
- Default, short circuit cycle: Restore View -> Render Response
- Possible to invoke cycle using **view parameters,** but have to use system events to invoke application phase (more ...)

```
<!-- Using GET to set a value and call listener in a bean -->
<f:metadata>
  <f:viewParam name="foo" value="#{bean.foo}"/>
  <f:event type="preRenderView" listener="#{bean.doIt}"/>
</f:metadata>
```

The Faces Servlet

To invoke the JSF subsystem, request must go through the **FacesServlet** (supplied by JSF)
- Servlet "invisible" to developer
- Configuration of FacesServlet in web.xml
- URL pattern /faces/ to trigger Servlet (or *.xhtml or other, ...), specify in web.xml (possible multiple patterns)

If URL-pattern is /faces/

- Must use http:// ..... /faces/... (else no JSF request)

Faces Context

FacesContext (compare ServletContext) object contains all of the per-request state information related to the processing of a single JavaServer Faces request, and the rendering of the corresponding response. A entry to...

- ...the component tree, an event queue (storing events for later execution, due to the processing model), messages (possible failure messages to GUI)
- Also possible to access low level data (HTTPSession, ...)

Globally accessible

- FacesContext.getCurrentInstance();

JSF Session Handling

JSF built on HTTP
- There's an HTTP session associated with each user
- javax.servlet.http.HttpSession object (as with Servlets)

On logout or similar we should invalidate the session

```
 // Accessing low level data
FacesContext.getCurrentInstance().
                   getExternalContext().invalidateSession();
```

Client Side Behaviours

JSF has a generic solution for associating client-side scripts (JavaScript) with UI components
- AJAX, Client-side validation, DOM and style manipulation, animations and visual effects, Alerts and confirmation dialogs, Tooltips and hover content, Keyboard handling, Deferred (lazy) data fetching, Integration with 3rd party client libraries, Client-side logging

We'll avoid, ... better using a higher level component suite (ICEFaces, PrimeFaces upcoming...)

Servlet, JSP, Filters and JSF

Possible to combine JSF and Servlets, JSP's, Filters, REST-resources in same application (in real life avoid)

```
http://...myapp/faces/   (faces request)
http://...myapp/...      (normal request)
http://...myapp/rs/...   (JAX-RS, REST request)
```

Or (other mappings)

```
http:// ...myapp/...xhtml   (faces request)
http:// ...myapp/...jspx    (normal request)
```

NetBeans and JSF

Have to add JSF as framework (at project creation or later)

- Properties > Frameworks > Add ...