

# What's new in JavaServer Faces (JSF2)



PAVONE AG Kai Jemella  
- Consultant Business & IT-Services -

## What is new in JSF 2.0

- ◆ Configuration
- ◆ Facelets View Declaration Language / Templates
- ◆ Facelets Composite Components
- ◆ Validation
- ◆ State Saving / System Events / Behavior
- ◆ GET Support / Navigation / Scopes
- ◆ Resource Loading / Error Handling
- ◆ Ajax

# Configuration – Annotations

## ◆ JSF 1.2 : faces-config.xml

```
<managed-bean>
  <managed-bean-name>foo</managed-bean-name>
  <managed-bean-class>com.foo.Foo</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

## ◆ JSF 2.0 : Managed Bean Annotation

```
@ManagedBean
@SessionScoped
public class Foo {
}
```

# Configuration – Annotations

## ◆ JSF 2.0 : Component Annotations

@FacesComponent

@FacesRenderer

@FacesConverter

@FacesValidator

@FacesBehavior

## Configuration faces-config.xml Order

### ◆ JSF 1.2:

- ▶ Loaded based on an order derived from the containing jar file's name.

### ◆ JSF 2.0:

- ▶ Reference other faces-config.xml by `<name>` element e.g. for view-handler or phase-listener.

```
<faces-config>
  <name>foo</name>
  <ordering>
    <after>
      <name>bar</name>
    </after>
  </ordering>
</faces-config>
```

```
<faces-config>
  <name>bar</name>
</faces-config>
```

```
<faces-config>
  <name>foobar</name>
  <ordering>
    <before>
      <others/>
    </before>
  </ordering>
</faces-config>
```

## Configuration - Notice

Notice that even if we are using JSF 2.0, we still need the Facesconfig.xml file. Annotations and the implicit navigation allow us to write an application without needing a faces-config.xml file. but there are still cases where the configuration file is needed. Localization information, advanced features such as ELResolvers, PhaseListeners, or artifacts that rely on the decorator pattern still require a faces-config.xml file.

(JSF 2.0 Cookbook, Anghel Leonard, PACKT PUBLISHING)

## What is new in JSF 2.0

- ◆ Configuration
- ◆ Facelets View Declaration Language / Templates
- ◆ Facelets Composite Components
- ◆ Validation
- ◆ State Saving / System Events / Behavior
- ◆ GET Support / Navigation / Scopes
- ◆ Resource Loading / Error Handling
- ◆ Ajax

# Facelets View Declaration Language / Templates

- ◆ Improving JSF by Dumping JSP
- ◆ Several JSF Alternatives to JSP
  - ▶ Apache Shale's Clay
  - ▶ JSF Templating
  - ▶ Facelets
- ◆ JSF 2.0: addresses this by
  - ▶ ViewDeclarationLanguage allows frameworks to define their own view declaration language
  - ▶ Standard non JSP view declaration language Facelets 2.0 (should be very familiar to Facelets 1.x)

# Facelets View Declaration Language / Templates

## ◆ JSP

```
<@ taglib uri="http://java.sun.com/jsf/html" prefix="h">
<@ taglib uri="http://java.sun.com/jsf/core" prefix="f">
<html>
<head><title>Show Customer</title></head>
<body>
  <f:view>
    <h1><h:outputText value="Show Customer"/></h1>
```

## ◆ Facelets

```
<!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:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html">
<head><title>Show Customer</title></head>
<body>
  <h1><h:outputText value="Show Customer"/></h1>
```

# Facelets View Declaration Language / Templates

## ➤ Template html

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
<head>...</head>
<body>
  <div id="header">
    <ui:insert name="header"><h1>MyGourmet</h1></ui:insert>
    <div id="content"><ui:insert name="content"/></div>
  </div>
</body></html>
```

## ➤ Facelet

```
<body>
  <ui:composition template="template.xhtml">
    <ui:define name="content">
      <h2>Customer</h2>
      <h:panelGrid id="grid" columns="2">
        <h:outputText value="Given name"/>
        <h:outputText value="#{customer.givenName}"/>
        <h:outputText value="Lastname" />
        <h:outputText value="#{customer.lastName}"/>
      </h:panelGrid>
    </ui:define>
  </ui:composition>
</body>
```

# Facelets View Declaration Language / Templates

## ➤ Multiple templates

```
<ui:composition template="template.xhtml">
  ...
  <ui:define name="left_sidebar">
    <ui:decorate template="sideBox.xhtml">
      <ui:param name="title" value="Messages"/>
      <h:outputText value="#{bean.msg}"/>
    </ui:decorate>
  </ui:define>
  ...
</ui:composition>
```

## What is new in JSF 2.0

- ◆ Configuration
- ◆ Facelets View Declaration Language / Templates
- ◆ Facelets Composite Components
- ◆ Validation
- ◆ State Saving / System Events / Behavior
- ◆ GET Support / Navigation / Scopes
- ◆ Resource Loading / Error Handling
- ◆ Ajax

## Facelets Composite Components

- ◆ Simplifies custom component development
- ◆ JSF 2.0: Create Components with a single file without Java
- ◆ First create tag library xml file and declare it in web.xml

```
<facelet-taglib>
  <namespace>http://www.my.facelets.component.com/jsf</namespace>
  <tag>
    <tag-name>sayhello</tag-name>
    <source>sayhello.xhtml</source>
  </tag>
</facelet-taglib>
```

```
<context-param>
  <param-name>javax.faces.FACELETS_LIBRARIES</param-name>
  <param-value>/WEB-INF/facelets/tags/mycomponents.xml</param-value>
</context-param>
```

# Facelets Composite Components

## ➤ Composite Component

```
<!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"
      xmlns:composite="http://java.sun.com/jsf/composite">
<head>...</head>
<body>
<composite:interface>
  <composite:attribute name="who"/>
</composite:interface>

<composite:implementation>
  <h:outputText value="Hello, #{cc.attrs.who}!"/>
</composite:implementation>
</body>
</html>
```

## ➤ Use Component

```
<html ... xmlns:mycomp="http://java.sun.com/jsf/composite/greet">
  <mycomp:sayhello who="World"/>
</html>
```

## Facelets Composite Components

- ◆ Think LEGOs
- ◆ Passing sub-elements to composition components
- ◆ Passing actions to composition components
- ◆ Insert facets, adding children, attaching listeners, validators, ...
- ◆ Mixing JSF and Dojo widget for custom components

## What is new in JSF 2.0

- ◆ Configuration
- ◆ Facelets View Declaration Language / Templates
- ◆ Facelets Composite Components
- ◆ Validation
- ◆ State Saving / System Events / Behavior
- ◆ GET Support / Navigation / Scopes
- ◆ Resource Loading / Error Handling
- ◆ Ajax

## Validation

- ◆ JSF 2.0: Bean Validation built-in integration with JSR-303 constraints.

```
<!-- provides bean validation -->  
<f:validateBean>
```

- ◆ JSF 2.0: Additional Validation tags

```
<!-- provides required field validation -->  
<f:validateRequired>
```

```
<!-- provides regular expression-based validation -->  
<f:validateRegexp>
```

# Validation – Bean Validation

## ◆ JSF 2.0: Bean Validation Annotation

```
public class FooBar {  
  
    @NotNull @Min(value = 1000) @Max(value = 99999)  
    private Integer foo;  
  
    @NotNull  
    private String bar;  
  
}
```

## ◆ JSF 2.0: Bean Validation JSR-303 custom validator and constraints

```
@Constraint(validatedBy = FooBarValidator.class)  
...  
public @interface FooBar { ... }  
  
public class FooBarValidator  
    implements ConstraintValidator<FooBar, Date> { ... }
```

# Validation – Bean Validation

## ◆ JSF 2.0: Bean Validation Groups

```
public interface usersIdsGroup {}
public interface usersCredentialsGroup {}

public userBean{

    @NotEmpty(message = "The name cannot be empty!",
        groups = beans.usersIdsGroup.class)
    private String name;

    @Size(min = 5, max = 20, message = "You ...",
        groups = beans.usersCredentialsGroup.class)
    private String password;
```

## ◆ JSF 2.0: validationGroups

```
<f:validateBean validationGroups="beans.usersIdsGroup, otherGroup">
    <h:outputText value="Name:"/>
    <h:inputText value="#{userBean.name}"/>
</f:validateBean>
```

## Validation Annotations

```
@AssertFalse // check element is false
@AssertTrue // check element is true
@DecimalMax // check number lower or equal
@DecimalMin // check number higher or equal
@Digits // check number digits and fraction
@Future // check date is in the future
@Max // check less than or equal
@Min // check higher than or equal
@NotNull // check not null
@Null // check null
@Valid // validation by associated object
@Past // check date is in the past
@Pattern // check by pattern
@Size // check size ist between min and max

//not in bean validation specification -> hibernate validator
@Length, @NotEmpty, @Email, @Range
```

## Validation Notice

### ◆ JSF 2.0: @NotNull context parameter

```
<context-param>  
  <param-name>  
javax.faces.INTERPRET_EMPTY_STRING_SUBMITTED_VALUES_AS_NULL  
  </param-name>  
  <param-value>>true</param-value>  
</context-param>
```

## What is new in JSF 2.0

- ◆ Configuration
- ◆ Facelets View Declaration Language / Templates
- ◆ Facelets Composite Components
- ◆ Validation
- ◆ State Saving / System Events / Behavior
- ◆ GET Support / Navigation / Scopes
- ◆ Resource Loading / Error Handling
- ◆ Ajax

## State Saving

- ◆ JSF 2.0: new State Saving
- ◆ Problem size of the saved state can be large
- ◆ Component developers can now use `PartialStateHolder` and `StateHelper`
- ◆ Save only modified state since view initially created

# System Events

- ◆ JSF 2.0 new SystemEvents
- ◆ Global SystemEvents Application initialization / destruction
- ◆ Component System Events
  - ▶ Component added to view
  - ▶ Component rendered
  - ▶ Component validated

```
<h:inputText>  
  <f:event type="preValidate«  
    listener="#{bean.doSomePreValidation}"/>  
</h:inputText>
```

## Behavior

- ◆ JSF 2.0 new class ClientBehavior, ClientBehaviorHolder  
e.g. AjaxBehavior f:ajax
- ◆ Enhancing client-side of components
- ◆ Prevent hand-coding scripts and manually wiring these scripts

```
<h:commandButton>
```

```
<!--First ask for confirmation -->
```

```
<foo:confirm event="click"/>
```

```
<!-- If successful, send an Ajax request -->
```

```
<f:ajax event="click"/>
```

```
</h:commandButton>
```

# Behavior

## ◆ JSF 2.0 confirm behavior

```
@FacesBehavior("foo.behavior.Confirm")
public class ConfirmBehavior extends ClientBehaviorBase
{
    @Override
    public String getScript(
        ClientBehaviorContext behaviorContext) {
        return "return confirm('Are you sure?')";
    }
}
```

# Behavior

## ◆ JSF 2.0 Append Behavior in taglib

```
<?xml version='1.0' encoding='UTF-8'?>
<facelet-taglib xmlns="http://java.sun.com/xml/ns/javaee"
version="2.0">
  <namespace>http://example.org/foo</namespace>
  <tag>
    <tag-name>confirm</tag-name>
    <behavior>
      <behavior-id>foo.behavior.Confirm</behavior-id>
    </behavior>
  </tag>
</facelet-taglib>
```

## What is new in JSF 2.0

- ◆ Configuration
- ◆ Facelets View Declaration Language / Templates
- ◆ Facelets Composite Components
- ◆ Validation
- ◆ State Saving / System Events / Behavior
- ◆ GET Support / Navigation / Scopes
- ◆ Resource Loading / Error Handling
- ◆ Ajax

# Navigation

- ◆ Reduce navigation related grunch code
- ◆ Implicit navigation e.g. action outcome corresponds to a view id
- ◆ ConfigurableNavigationHandler (available navigation rules/cases)
- ◆ Dynamically control the application flow
- ◆ Conditional navigation for pre-condition

```
<navigation-rule>
  <from-view-id>/foo.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/bar.xhtml</to-view-id>^
    <if>#{foo.someCondition}</if>
  </navigation-case>
</navigation-rule>
```

## GET Support

- ◆ JSF 1.2: value processing of values via HTTP Post
- ◆ JSF 2.0: view parameters for HTTP e.g. page1.xhtml?foo=bar

```
<f:metadata>
  <f:viewParam name="foo" value="#{bean.foo}" required="true"
requiredMessage="This parameter is a must!"/>

  <!-- PreRenderViewEvent -->
  <f:event type="preRenderView" listener="#{bean.doSomething}"/>
</f:metadata>
```

- ◆ JSF 2.0: view parameter inverse mapping and manual parameter

```
<h:link outcome="success" includeViewParams="true">
  <f:param name="foo" value="bar"/>
</h:link>
```

## Scopes

- ◆ JSF 2.0: view scope
  - ◆ „longer than request, shorter than session“
  - ◆ Until the user finishes interaction with the current view
  - ◆ `UIViewRoot.getViewMap()` or EL `#{viewScope}`
- ◆ JSF 2.0: flash scope
  - ◆ Concept from Rails
  - ◆ Short-lived conversation
  - ◆ State survives a redirect e.g. `page2?faces-redirect=true`
  - ◆ `ExternalContext.getFlash()` or EL `#{flash}`
- ◆ JSF 2.0: Custom Scopes to place managed beans into custom scope

## What is new in JSF 2.0

- ◆ Configuration
- ◆ Facelets View Declaration Language / Templates
- ◆ Facelets Composite Components
- ◆ Validation
- ◆ State Saving / System Events / Behavior
- ◆ GET Support / Navigation / Scopes
- ◆ Resource Loading / Error Handling
- ◆ Ajax

## Resource Loading / Error Handling

- ◆ JSF 2.0: ResourceHandler e.g. images, JavaScript, stylesheets, ...

```
/resources/css/foobar.css  
/resources/images/foobar.jpg  
/resources/javascript/foobar.js
```

```
<h:graphicImage library="images" name="foobar.jpg"/>  
<h:outputStylesheet library="css" name="foobar.css"/>  
<h:outputScript library="javascript" name="foobar.js" target="head"/>
```

```
<!-- via EL -->  
<h:graphicImage value="#{resource['images:foobar.jpg']}" />
```

- ◆ JSF 2.0: @ResourceDependency

```
@ResourceDependency(name="my.css", library="libus")  
public class MyComponent extends UIComponentBase {  
...  
}
```

## Resource Loading / Error Handling

- ◆ JSF 2.0: ExceptionHandler API (central exception handling)

```
<error-page>  
  <exception-type>java.lang.NumberFormatException</exception-type>  
  <location>/faces/error.xhtml</location>  
</error-page>
```

## What is new in JSF 2.0

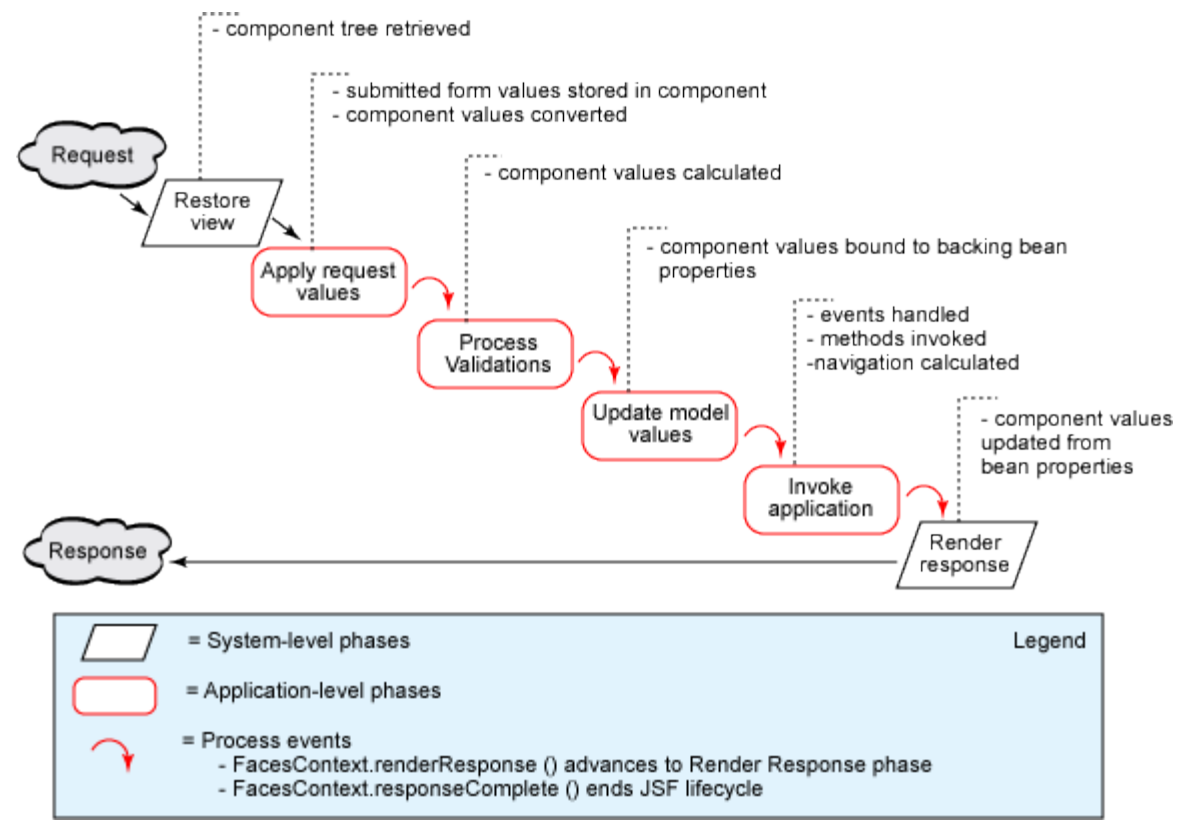
- ◆ Configuration
- ◆ Facelets View Declaration Language / Templates
- ◆ Facelets Composite Components
- ◆ Validation
- ◆ State Saving / System Events / Behavior
- ◆ GET Support / Navigation / Scopes
- ◆ Resource Loading / Error Handling
- ◆ Ajax

# Ajax

- ◆ JSF 2.0 (2009) four years since Ajax was coined.
- ◆ JSF Ajax problem has been solved many times!
- ◆ Many JSF/Ajax frameworks available.
  - ▶ DynamicFaces, RichFaces, ajax4jsf,
  - ▶ PrimeFaces, MyFaces Tomahawk ...
- ◆ Problem cross-framework compatibility
- ◆ Isn't it time to Ajax without 3rd party frameworks?

# Ajax

## Partial View Processing and Pratial View Rendering



(IBM: <http://www.ibm.com/developerworks/library/j-jsf2/>)

# Ajax

- ◆ JSF 2.0: Ajax Parameter
  - ▶ javax.faces.partial.ajax
  - ▶ javax.faces.source
  - ▶ javax.faces.partial.execute
  - ▶ javax.faces.partial.render

# Ajax

## ◆ JSF 2.0: JavaScript API

```
<h:form id="formID" prependId="false">
```

```
    <h:outputScript name="jsf.js" library="javax.faces"
        target="head" />
```

```
    <h:outputText id="randomID" value="#{random.random}" />
```

```
    <h:commandButton id="buttonID" value="Random"
        onclick="jsf.ajax.request(this, event, {execute:
        this.id, render: 'randomID'}); return false;" />
```

```
</h:form>
```

# Ajax

## ◆ JSF 2.0: f:ajax tag

```
<h:form id="formID" prependId="false">  
    <h:outputText id="randomID" value="#{random.random}" />  
    <h:commandButton id="buttonID" value="Random">  
        <f:ajax execute="formID:buttonID formID:randomID"  
            render="formID:randomID" />  
    </h:commandButton>  
</h:form>
```

# Ajax Keywords

Keyword	Execute	Render
@all	Every component on the page is submitted and processed. This is useful when you want to do a full-page submit.	Every component on the page is rendered. This is useful when you just want to re-render the whole page asynchronously. E.g. update client side state outside of JSF.
@none	Execute the lifecycle, including list phase listeners, but no components will be traversed.	Perform the Render Response phase, including firing any preRenderView events, but don't actually render anything.
@this	Submit and process only the component to which the <f:ajax> is applied.	Render only the component to which the <f:ajax> is applied
@form	Submit and process the entire <h:form> in which the component has the <f:ajax> is nested.	Render the entire <h:form> in which the component has the <f:ajax> is nested.

(JavaServer Faces 2.0: the complete reference By Ed Burns, Neil Griffin, Chris Schalk)

## Resources

◆ JSF Specification

<http://jcp.org/aboutJava/communityprocess/final/jsr314/index.html>

◆ Andy Schwartz

<http://andyschwartz.wordpress.com/2009/07/31/whats-new-in-jsf-2/>

◆ Irian [http://jsfatwork.irian.at/book\\_de/](http://jsfatwork.irian.at/book_de/)

◆ IBM Developerworks

<http://www.ibm.com/developerworks/java/library/j-jsf2fu2/index.html>

◆ JSF 2.0 Cookbook, Anghel Leonard, PACKT PUBLISHING

◆ JavaServer faces 2.0: the complete reference By Ed Burns, Neil Griffin, Chris Schalk