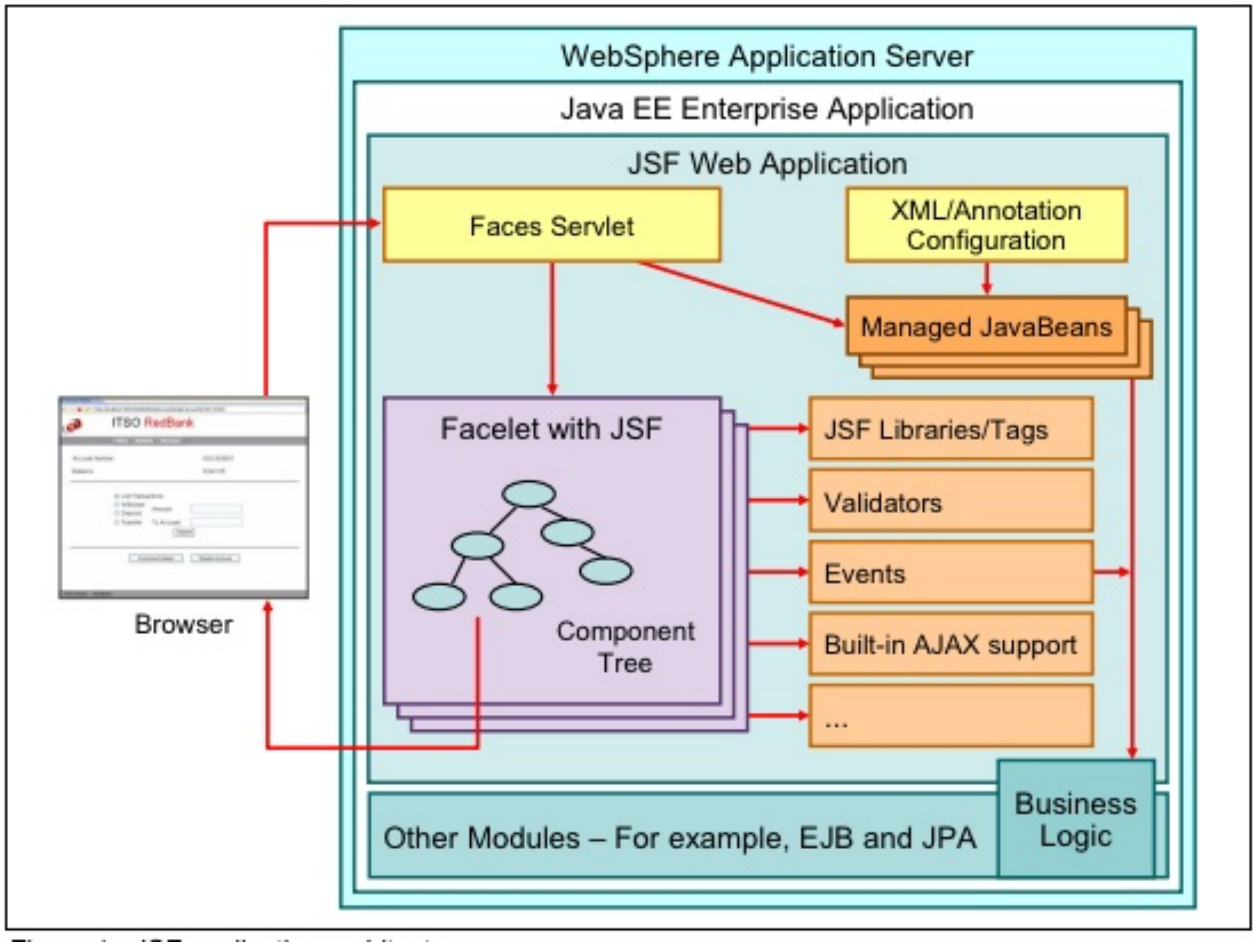
JSF: an introduction

Based on version 2.2

JSF architecture



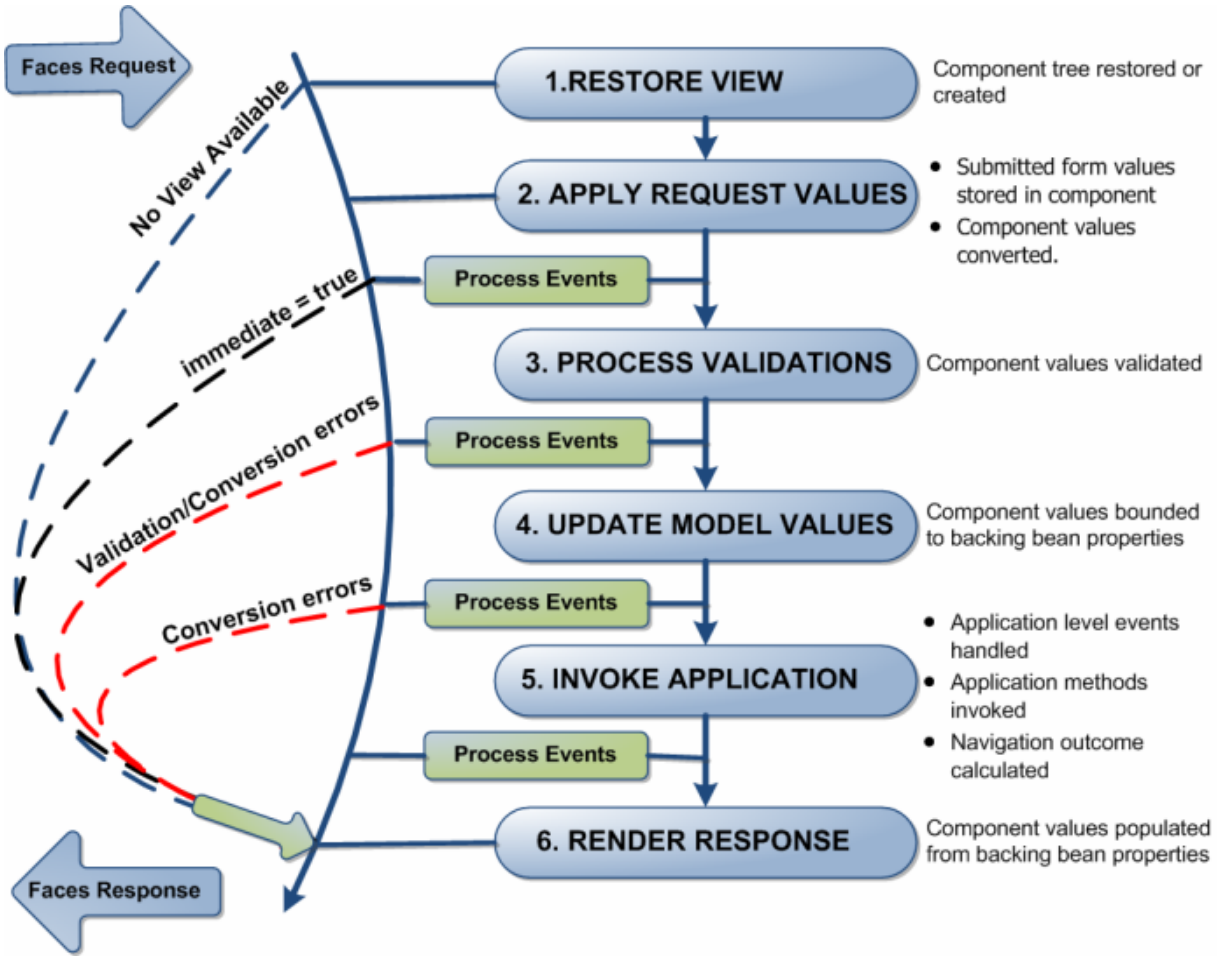
The container

To use JSF a bare Tomcat is not enough: you need a JSF application server (e.g. Glassfish).

Tomcat can be configured to run JSF:

See http://www.byteslounge.com/tutorials/how-to-configure-jsf-in-tomcat-example

The lifecycle



Interaction between the page and the beans

Variables in the bean can be accessed through the Expression Language

${myObj.myVar}

Accesses the myVar variable in the myObj bean

It was already available in JSP

JSTL

Java Standard Template Library

http://java.sun.com/products/jsp/tutorial/TagLibrariesTOC.html

JSTL

Core tags

<%@ taglib
uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>

```
<c:set var="foo" scope="session" value="..."/>

${foo}
```

Area	Function	Tags	Prefix
Core	Variable support	remove set	c
	Flow control	choose when otherwise forEach forTokens if	
	URL management	import param redirect param url param	
	Miscellaneous	catch out	

See http://download.oracle.com/javaee/5/tutorial/doc/bnakh.html

JSTL - xml

XML tags

<%@ taglib
uri="http://java.sun.com/jsp/jstl/xml"
prefix="x" %>

Area	Function	Tags	Prefix
XML	Core	out parse set	x
	Flow control	choose when otherwise forEach if	
	Transformation	transform param	

```
<c:if test="${applicationScope:booklist == null}" >
     <c:import url="${initParam.booksURL}" var="xml" />
    <x:parse doc="${xml}" var="booklist" scope="application" />
</c:if>
<x:set var="abook"
    select="$applicationScope.booklist/books/book[@id=$param:bookId]" />
<h2><x:out select="$abook/title"/></h2>
```

See http://download.oracle.com/javaee/5/tutorial/doc/bnakq.html

JSTL - sql

Area	Function	Tags	Prefix
Database	Setting the data source	setDataSource	sql
	SQL	query dateParam param transaction update dateParam param	

SQL tags

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>

```
<sql:setDataSource dataSource="jdbc/BookDB" />
<c:set var="bid" value="${param.Add}"/>
<sql:query var="books" >
    select * from PUBLIC.books where id = ?
    <sql:param value="${bid}" />
</sql:query>
```

See http://download.oracle.com/javaee/5/tutorial/doc/bnald.html

JSTL-fn

Area	Function	Tags
Functions	Collection length	length
	String manipulation	toUpperCase, toLowerCase
		substring, substringAfter, substringBefore
		trim
		replace
		indexOf, startsWith, endsWith, contains, containsIgnoreCase
		split, join
		escapeXml

function tags

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>

```
<c:if test="${fn:length(param.username) > 0}" >
  <%@include file="response.jsp" %>
</c:if>
```

See http://download.oracle.com/javaee/5/tutorial/doc/bnalg.html

JSTL-fmt

Area	Function	Tags	Prefix
I18N	Setting Locale	setLocale requestEncoding	fmt
	Messaging	bundle message param setBundle	
	Number and Date Formatting	formatNumber formatDate parseDate parseNumber setTimeZone timeZone	

i18n tags

<%@ taglib
uri="http://java.sun.com/jsp/jstl/fmt"
prefix="fmt" %>

```
<h3><fmt:message key="Choose"/></h3>
```

See http://download.oracle.com/javaee/5/tutorial/doc/bnakw.html

JSF Tags

JSF libraries/tags

HTML tags

http://docs.oracle.com/javaee/7/tutorial/doc/jsf-page002.htm

JSF Core Tag Library

http://docs.oracle.com/javaee/7/tutorial/doc/jsf-page003.htm

Composite Tag Library

http://docs.oracle.com/javaee/7/tutorial/doc/jsf-facelets005.htm

JSF HTML Tag Library

Contains JavaServer Faces component tags for all UIComponent + HTML RenderKit Renderer combinations defined in the JSF Specification.

h:head h:body

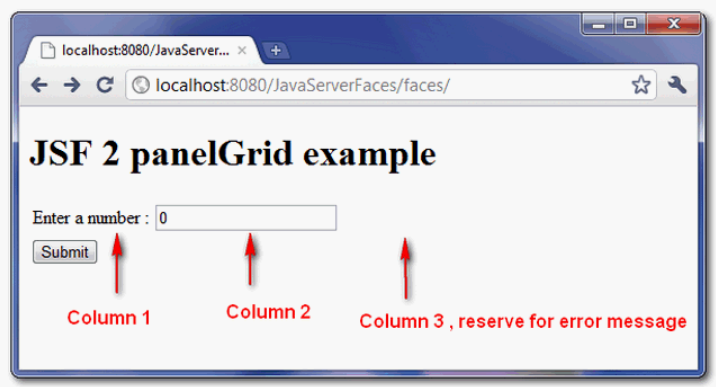
h:form h:inputText

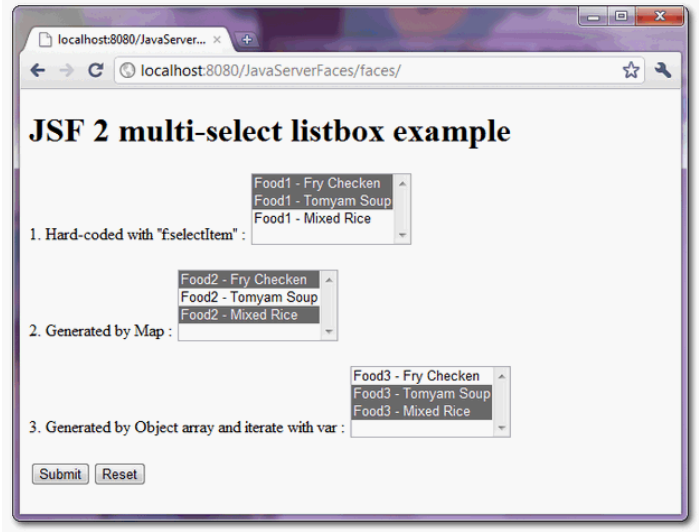
h:dataTable

http://www.jsftoolbox.com/documentation/help/12-TagReference/html/index.jsf

Widgets

<h:inputText>
<h:inputSecret>
<h:inputTextarea>
<h:inputHidden>
<h:selectBooleanCheckbox>
<h:selectManyCheckbox >
<h:selectOneRadio>
<h:selectOneListbox>
<h:selectManyListbox>
<h:selectOneMenu>
<h:selectManyMenu >
<h:outputText> tag.
<h:outputFormat>
<h:graphicImage>
<h:outputStylesheet>
<h:outputScript>
<h:button> and <h:commandButton>
<h:link>, <h:commandLink> <h:outputLink>
<h:panelGrid>
<h:messages> and <h:message>
<f:param>
<f:attribute>
<f:setpropertyactionlistener>





For examples see
http://www.mkyong.com/tutorials/jsf-2-0-tutorials/

JSF Core Tag Library

contains general purpose tags for conversion, validation, Ajax, and other use cases.

See http://www.jsftoolbox.com/documentation/help/12-TagReference/core/index.jsf

Composite Tag Library

The composite tag library supports the definition and implementation of composite UI components based on the Facelets view declaration language (VDL).

Composite tutorials

- https://weblogs.java.net/blog/driscoll/archive/2008/11/writing_a_simpl.html

- http://www.liferay.com/it/web/neil.griffin/blog/-/blogs/jsf-2-0-test-drive-part-1%3A-developing-a-facelet-composite-component

Extra libraries

RichFaces (http://www.jboss.org/richfaces)

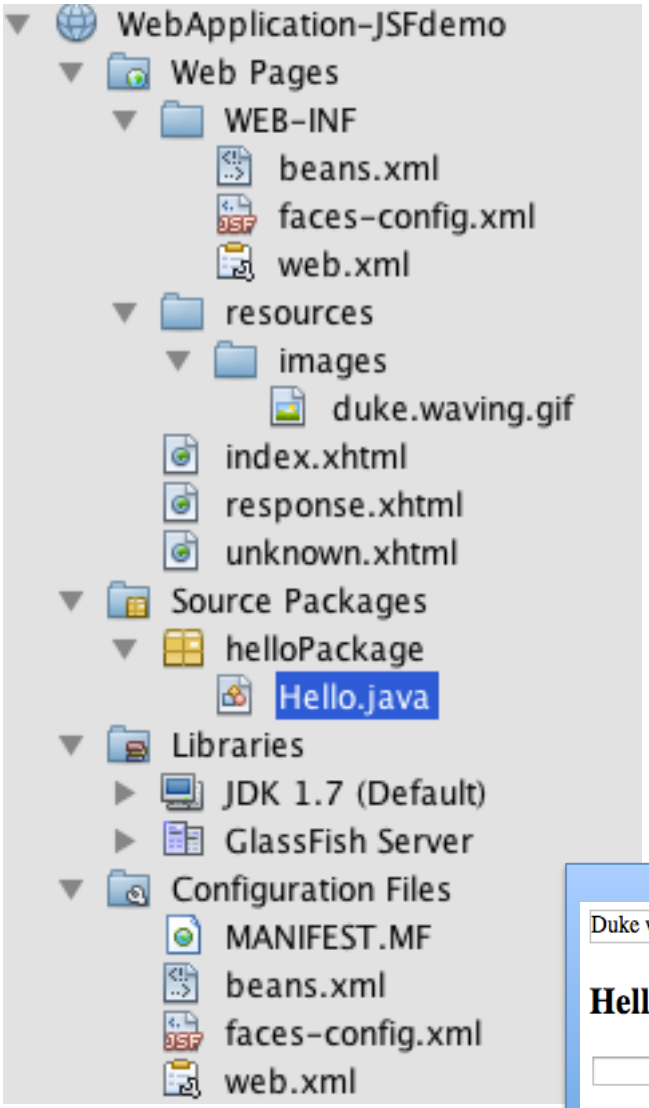
PrimeFaces (http://www.primefaces.org/)

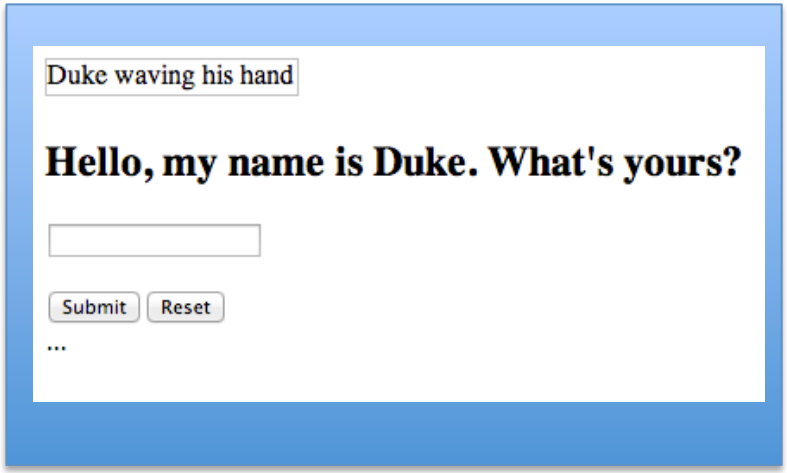
ICEFaces (http://www.icesoft.org/java/home.jsf)

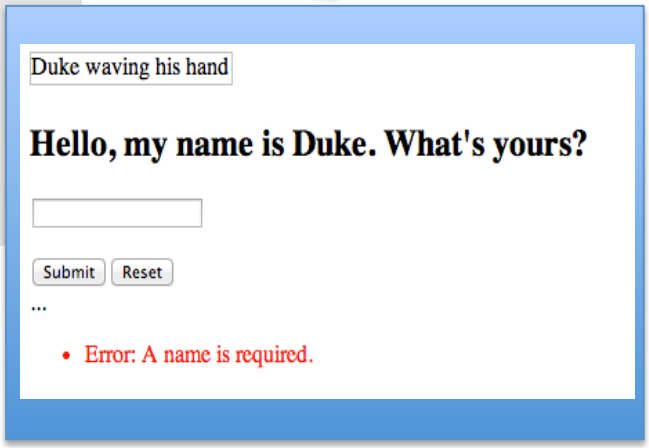
Trinidad(http://myfaces.apache.org/trinidad/)

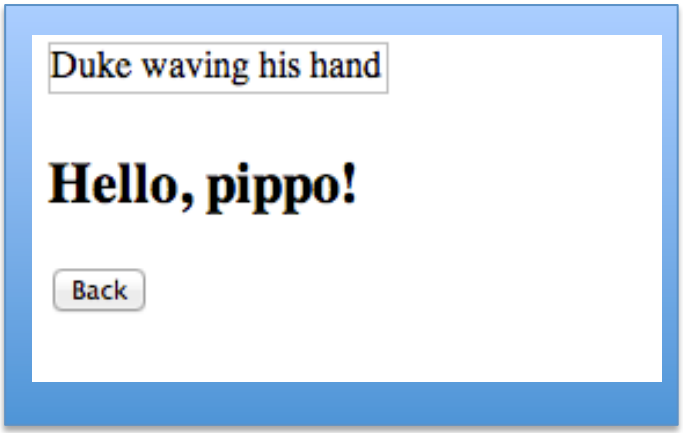
Tomahawk (http://myfaces.apache.org/tomahawk/)

An example


WebApplication-JSFdemo
Web Pages
WEB-INF
beans.xml
faces-config.xml
web.xml
resources
images
duke.waving.gif
index.xhtml
response.xhtml
unknown.xhtml
Source Packages
helloPackage
Hello.java
Libraries
JDK 1.7 (Default)
GlassFish Server
Configuration Files
MANIFEST.MF
beans.xml
faces-config.xml
web.xml


Duke waving his hand
Hello, my name is Duke. What's yours?
Submit
Reset
...

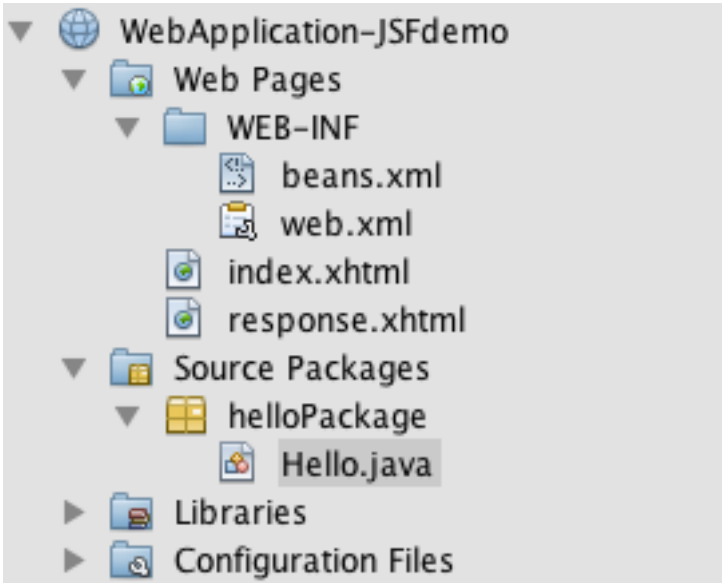

Duke waving his hand
Hello, my name is Duke. What's yours?
Submit
Reset
...
Error: A name is required.


Duke waving his hand
Hello, pippo!
Back

An example: hello bean

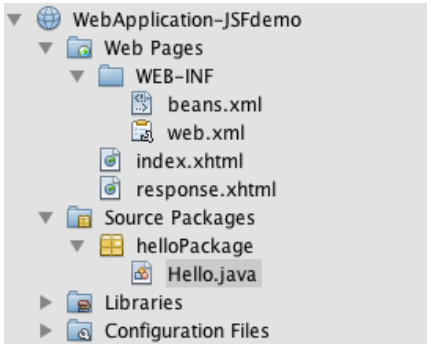
```
package helloPackage;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;
@Named
@RequestScoped
public class Hello {
   private String name;
   public Hello() { }
   public String getName() {
      return name;
   }
   public void setName(String user_name) {
      this.name = user_name;
   }
}
```

The bean will be called "hello"



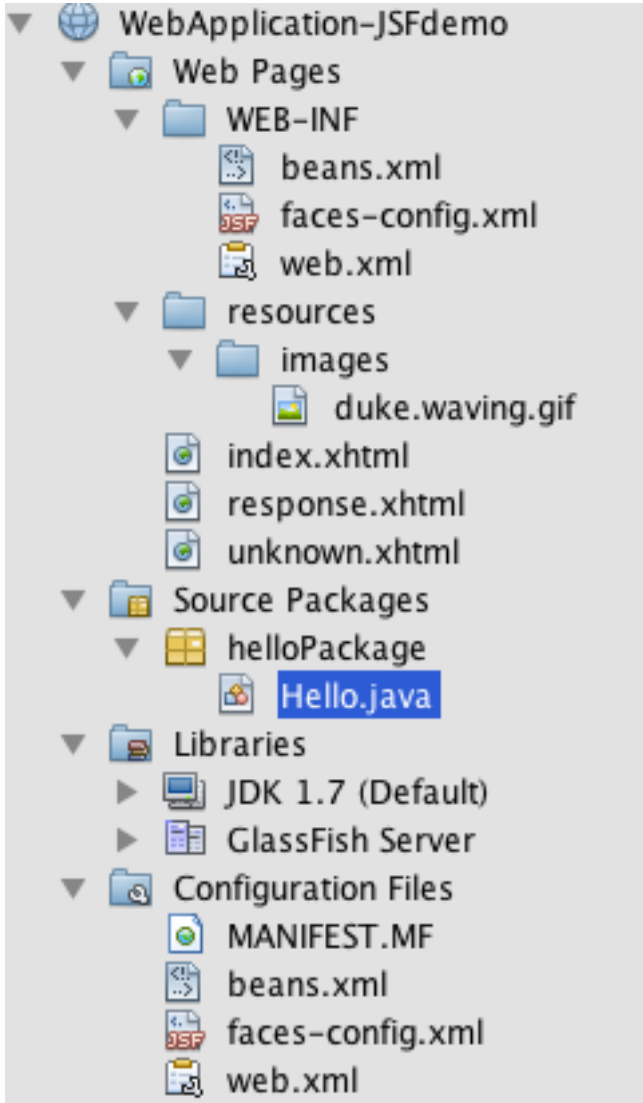
An example: index.xhtml

```
<?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 lang="en"  xmlns="http://www.w3.org/1999/xhtml"   xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
     <title>Facelets Hello Greeting</title>
   </h:head>
   <h:body>
      <h:form>
         <h:graphicImage url="#{resource['images:duke.waving.gif']}"  alt="Duke waving his hand"/>
         <h2>Hello, my name is Duke. What's yours?</h2>
         <h:inputText id="username"
                  title="My name is: "
                  value="#{hello.name}"
                  required="true"
                  requiredMessage="Error: A name is required."
                  maxlength="25" />
         <p></p>
         <h:commandButton id="submit" value="Submit" action="response">  </h:commandButton>
         <h:commandButton id="reset" value="Reset" type="reset"> </h:commandButton>
      </h:form>
      ...
   </h:body>
</html>
```



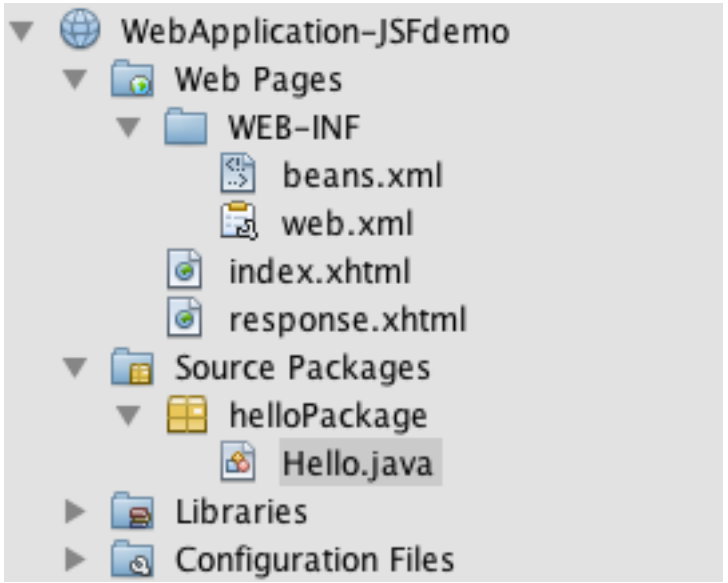
An example: response.xhtml

```
<?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 lang="en"
     xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html">
   <h:head>
      <title>Facelets Hello Response</title>
   </h:head>
   <h:body>
      <h:form>
      <h:graphicImage url="#{resource['images:duke.waving.gif']}"
                      alt="Duke waving his hand"/>
         <h2>Hello, #{hello.name}!</h2>
         <p></p>
         <h:commandButton id="back" value="Back" action="index" />
      </h:form>
   </h:body>
</html>
```

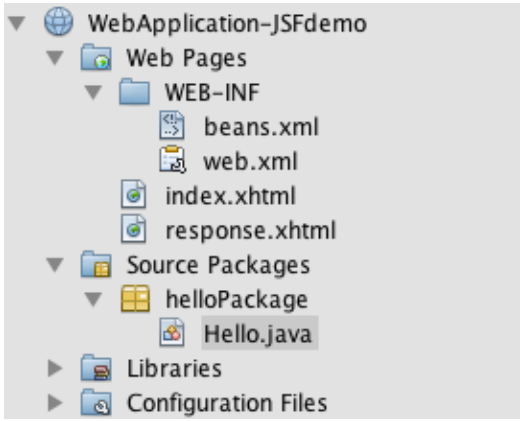


An example: web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>  OR <url-pattern>*.faces</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>  30   </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>faces/index.xhtml</welcome-file>
  </welcome-file-list>
</web-app>
```



An example: beans.xml




```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/
beans_1_1.xsd"
    bean-discovery-mode="annotated">
</beans>
```

The presence of a beans.xml file signals the servlet container that the application uses JSR-299 (Contexts and Dependency Injection) - without it, the application's classes will not be scanned for CDI annotations either and dependency injection will not be performed.
If you don't use CDI (e.g. you only use plain JSF managed beans), you don't need beans.xml.

Inner working

All JSF pages are processes by a special servlet (FacesServlet) that is a part of the JSF implementation code.

The JSF URLs have a special format to ensure that the correct servlet is activated when a JSF page is requested.

For example, they may have an extension .faces.

Ex: http://localhost:8080/hello/index.faces.

Or, you can also define a prefix mapping instead of the .faces extension mapping.

(managed or CDI) beans

Until JSF 2.0 the use of ManagedBeans was the way to go

```
@ManagedBean
public class SomeName { … }
```

From JSF 2.2 one uses CDI beans

Use of @ManagedBean from javax.faces.bean is now deprecated; instead one should use @Named from javax.inject package to signify a managed bean as a JSF controller class.

If you do, you can use the file (instead of annotations) to define CDI configurations – this allows you, for example, to provide a test configuration where some dependencies are replaced with mocks.

Bean scope

- ApplicationScoped - only one instance of the bean will exist in the entire application. After bean initialization, every time a client requests an instance of this bean the container will always provide the same bean instance.
- SessionScoped - used inside a web application context. An instance of this bean will exist per HTTP session.
- RequestScoped - used inside a web application context. An instance of this bean will exist per HTTP request.
- ConversationScoped - used inside a web application context. This kind of beans are used to represent a conversation between the client and the server. They may be used to keep state between multiple client AJAX requests or even between distinct page requests, if the client provides the conversation identifier to the server between page requests.

- **Dependent scope** is the default CDI bean scope, ie. if a bean does not declare a specific scope it will be injected as a Dependent scoped bean. This means that it will have the same scope as the bean where it's being injected. Dependent scoped bean instances are never shared between clients. Every client will always fetch a new instance of a Dependent scoped bean.

Conversational scope vs. session

Conversational scope is programmaticaly delimited.

```
private @Inject Conversation conversation;
conversation.begin();
conversation.end();
```

For examples, see:

- http://stackoverflow.com/questions/7788430/how-does-jsf-2-conversationscope-work
- http://www.byteslounge.com/tutorials/java-ee-cdi-conversationscoped-example

View scope?

- No more view scope in JSF 2.2. Replace it with the conversational scope.

- Managed CDI beans that are able to live for longer periods of time, such as SessionScoped and ConversationScoped, must implement the Serializable interface.

JSF: Expression Language

Based on version 2.2

Expression language

```
<some:tag>
   some text #{expr} some text
</some:tag>
<another:tag value="#{expr}"/>
<h:commandButton id="submit"
                 action="#{customer.submit}" />
<h:commandButton id="submit"
                 action="#{customer["submit”]}" />
<h:inputText value="#{userNumberBean.userNumber('5')}">
<h:commandButton action="#{trader.buy('SOMESTOCK')}"
value="buy"/>
```

Expression language

#{} are for deferred expressions (deferred expressions are resolved depending on the lifecycle of the page) and can be used to read or write from or to a bean or to make a method call.

${} are expressions for immediate resolution, as soon as they are encountered they are resolved. They are read-only.

Expression language

#{attributeName[entryName]}

Works for

- Fields. Equivalent to

  attributeName.getEntryName() (get or set)

– Array. Equivalent to

  theArray[index] (getting and setting)

– List. Equivalent to

  • theList.get(index) or theList.set(index, submitted-val)

– Map. Equivalent to

  • theMap.get(key) or theMap.put(key, submitted-val)

Calling Methods

```
<h:inputText value="#{userNumberBean.userNumber('5')}">
```

Expression Language – Implicit Objects

- facesContext. The FacesContext object.
  - E.g. #{facesContext.externalContext.session.id}
- param and paramValues. Request params.
  - E.g. #{param.custID}
- header and headerValues. Request headers.
  - E.g. #{header.Accept} or #{header["Accept"]}
  - #{header["Accept-Encoding"]}
- cookie. Cookie object (not cookie value).
  - E.g. #{cookie.userCookie.value} or
    #{cookie["userCookie"].value}
- initParam. Context initialization param.
- requestScope, sessionScope, applicationScope.

Expression Language - Operators

- Arithmetic + - * / div % mod
- Relational == eq != ne < lt > gt <= le >= ge
- Logical && and || or ! Not
- Empty Empty
- – True for null, empty string, empty array, empty list, empty map. False otherwise.
- ${ test ? expression1 : expression2 }

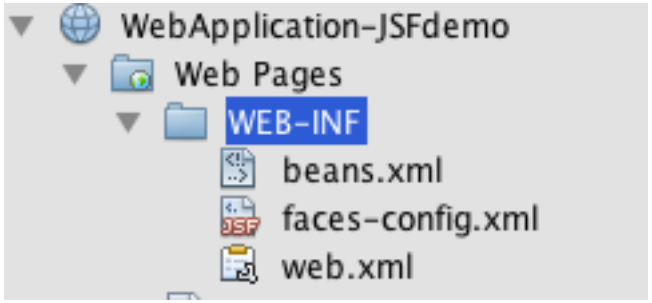
CAUTION

- Use extremely sparingly to preserve MVC model

JSF: Navigation

Based on version 2.2

Navigation



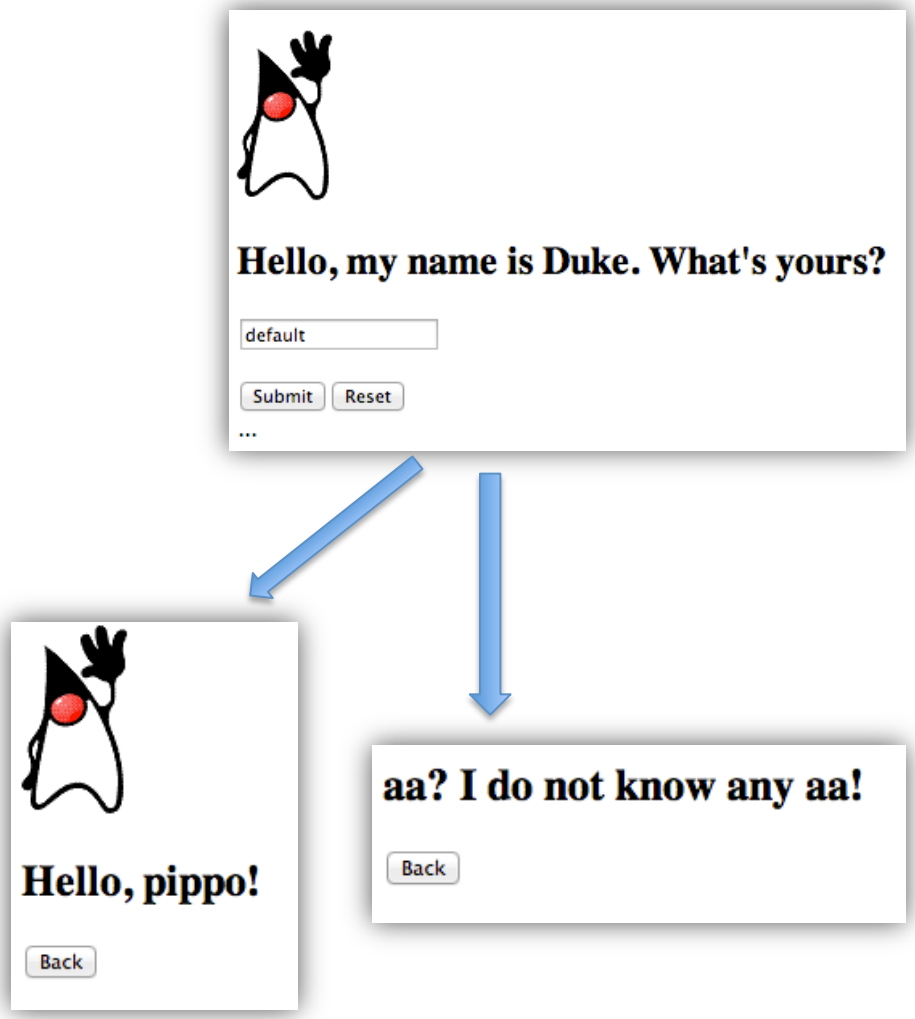
Netbeans: in Web-inf, New…Other, Java Server Faces, JSF configuration

1. The NavigationHandler selects the navigation rule that matches the page currently displayed.
2. It matches the outcome or the action method reference that it received from the default javax.faces.event.ActionListener with those defined by the navigation cases.
3. It tries to match both the method reference and the outcome against the same navigation case.
4. If the previous step fails, the navigation handler attempts to match the outcome.
5. Finally, the navigation handler attempts to match the action method reference if the previous two attempts failed.
6. If no navigation case is matched, it displays the same view again.

http://docs.oracle.com/javaee/7/tutorial/doc/jsf-intro005.htm#BNAQL
http://sqltech.cl/doc/oas10gR3/web.1013/b25386/web_PageNavigation002.htm

Hello.bean

```
package helloPackage;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;
@Named
@RequestScoped
public class Hello {
   private String name;
   public Hello() {}
   public String getName() {
      return name;
   }
   public void setName(String user_name) {
      this.name = user_name;
   }
   public String readName() {
      return name;
   }
}
```

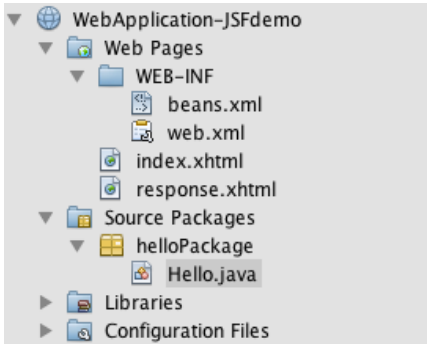


Faces.config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config version="2.2"
   xmlns="http://xmlns.jcp.org/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd">
<navigation-rule>
   <from-view-id>/index.xhtml</from-view-id>
   <navigation-case>
      <from-outcome>pippo</from-outcome>
      <to-view-id>/response.xhtml</to-view-id>
   </navigation-case>
  <navigation-case>
        <to-view-id>unknown.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
</faces-config>
```

An example: index.xhtml

```
<?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 lang="en"  xmlns="http://www.w3.org/1999/xhtml"   xmlns:h="http://xmlns.jcp.org/jsf/html">
  <h:head>
     <title>Facelets Hello Greeting</title>
   </h:head>
   <h:body>
      <h:form>
         <h:graphicImage url="#{resource['images:duke.waving.gif']}"  alt="Duke waving his hand"/>
         <h2>Hello, my name is Duke. What's yours?</h2>
         <h:inputText id="username"
                  title="My name is: "
                  value="#{hello.name}"
                  required="true"
                  requiredMessage="Error: A name is required."
                  maxlength="25" />
         <p></p>
         <h:commandButton id="submit" value="Submit" action="#{hello.readName}"> </h:commandButton>
         <h:commandButton id="reset" value="Reset" type="reset"> </h:commandButton>
      </h:form>
      ...
   </h:body>
</html>
```



Unknown.xhtml

```
<?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 lang="en"
     xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html">
   <h:head>
      <title>Facelets Hello Response</title>
   </h:head>
   <h:body>
      <h:form>
         <h2>#{hello["name"]}? I do not know any #{hello["name"]}!</h2>
         <p></p>
         <h:commandButton id="back" value="Back" action="index" />
      </h:form>
   </h:body>
</html>
```

JSF: Validation

Based on version 2.2

Declarative validator tags

- validateBean Registers a bean validator for the component.
- validateDoubleRange Checks whether the local value of a component is within a certain range. The value must be floating-point or convertible to floating-point.
- validateLength Checks whether the length of a component's local value is within a certain range. The value must be a java.lang.String.
- validateLongRange Checks whether the local value of a component is within a certain range. The value must be any numeric type or String that can be converted to a long.
- validateRegEx Checks whether the local value of a component is a match against a regular expression from the java.util.regex package.
- validateRequired Ensures that the local value is not empty on an javax.faces.component.EditableValueHolder component.

Declarative validator tags - examples

```
<h:outputLabel for="fName"
    value="*Enter First Name: " title="First Name"/>
<h:inputText id="fName"
    value="#{personView.per.fName}" title="First Name"
    required="true"
    requiredMessage="first name is required"/>
<h:message id="m1" for="fName" style="color:red"/>
```

Declarative validator tags - examples

```
<h:outputLabel for="rankingId" value="Enter Ranking: "
    title="Ranking"/>
<h:inputText id="rankingId" value="#{personView.ranking}"
    title="Ranking”size="3" validatorMessage="Registration
    Open Only for Level 3.0 to 4.5">
 <f:validateDoubleRange minimum="3.0" maximum="4.5"
    for="rankingId"/>
 <!--f:validateRegex pattern="[3-4]\.?[05]" for="rankingId"/-->
</h:inputText>
<h:message id="m6" for="rankingId" style="color:red"/>
```

Programmatic validation – the class

```
@FacesValidator("emailValidator")
public class EmailValidator implements Validator {
@Override
public void validate(FacesContext context, UIComponent component, Object value)
        throws ValidatorException {
   boolean matches = check(value.toString());
   if (!matches) {
     FacesMessage msg =
           new FacesMessage(" E-mail validation failed.",
           "Please provide E-mail address in this format: abcd@abc.com");
     msg.setSeverity(FacesMessage.SEVERITY_ERROR);
     throw new ValidatorException(msg);
   }
 }
 private boolean check(String s) {…}
}
```

Programmatic validation - invokation

```
<h:outputLabel for="email" value="*Enter Email: "
    title="Primary Email"/>
<h:inputText id="email" value="#{person.email}"
    required="true" requiredMessage="email is
    required"
    title="Primary Email" tabindex="3">
  <f:validator validatorId="emailValidator" />
</h:inputText>
<h:message id="m2a" for="email" style="color:red"/>
```

Programmatic validation – the class

```
@named
public class Person implements Validator {
…
public void anyName(FacesContext context, UIComponent
component, Object value) throws ValidatorException {
   boolean matches = check(value.toString());
   if (!matches) {
     …
     throw new ValidatorException(msg);
   }
  }
}
```

Programmatic validation - invokation

```
<h:outputLabel for="email" value="*Enter Email: "
    title="Primary Email"/>
<h:inputText id="email" value="#{person.email}"
    required="true" requiredMessage="email is
    required"
    title="Primary Email" tabindex="3”
    validator ="person.anyName" />
</h:inputText>
<h:message id="m2a" for="email" style="color:red"/>
```

Validation rules

1. If form validation fails, tell user that the form validation failed and list all the instructions/errors at the top of form.
2. Repeat the instructions in front/after the field.
3. If possible set focus on the "top" error message or on the field corresponding to the first error message.
4. Use correct font color and size for displaying error.
5. The error message should tell the user what needs to be done and NOT what is wrong.
6. Use correct tab index or make sure that form fields are in a logical tab order
7. When form controls are text input fields use the LABEL element.

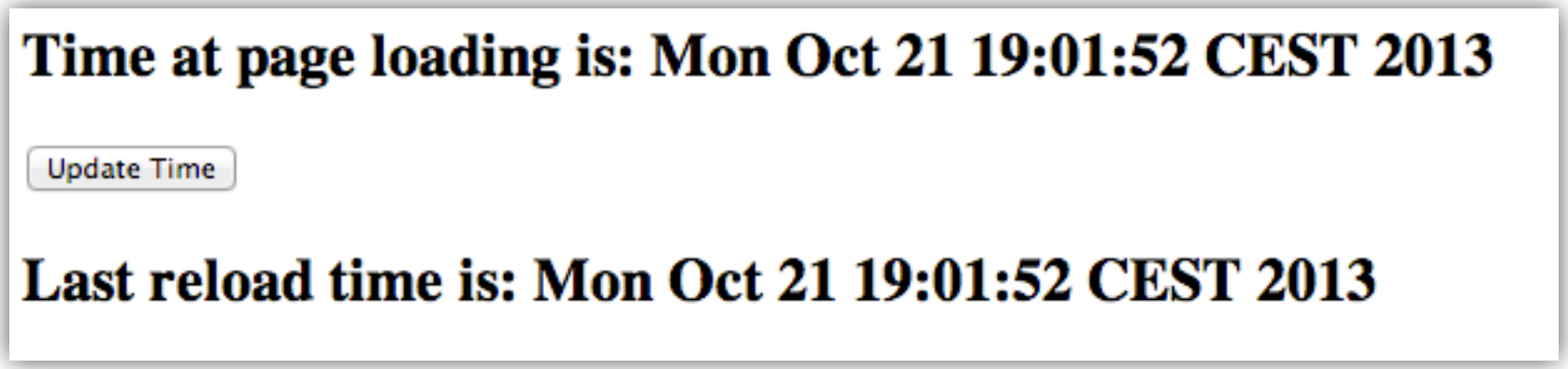
Validators

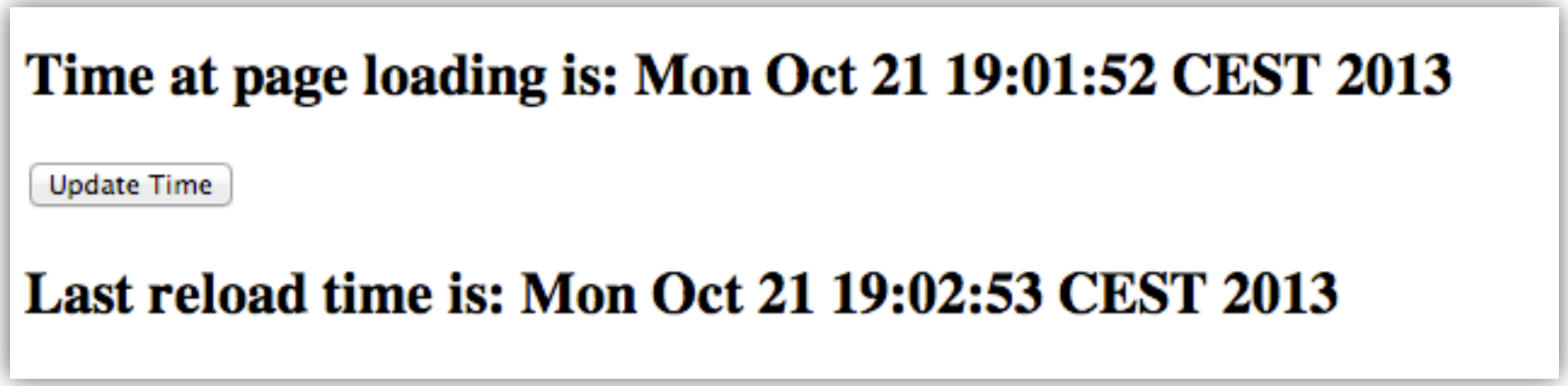
- http://incepttechnologies.blogspot.it/p/validation-in-jsf.html
- http://www.tutorialspoint.com/jsf/jsf_validation_tags.htm

JSF: Ajax support

Based on version 2.2

Ajax support





Ajax support

```
package pckg;
import java.util.*;
import javax.inject.Named;
import javax.enterprise.context.RequestScoped;

@Named(value = "timeBean")
@RequestScoped
public class AjaxTimeBean {
   Calendar x;
   public AjaxTimeBean() {}
   public String time() {
      x=new GregorianCalendar();
      Date d=x.getTime();
      return d.toString();
   }
}
```

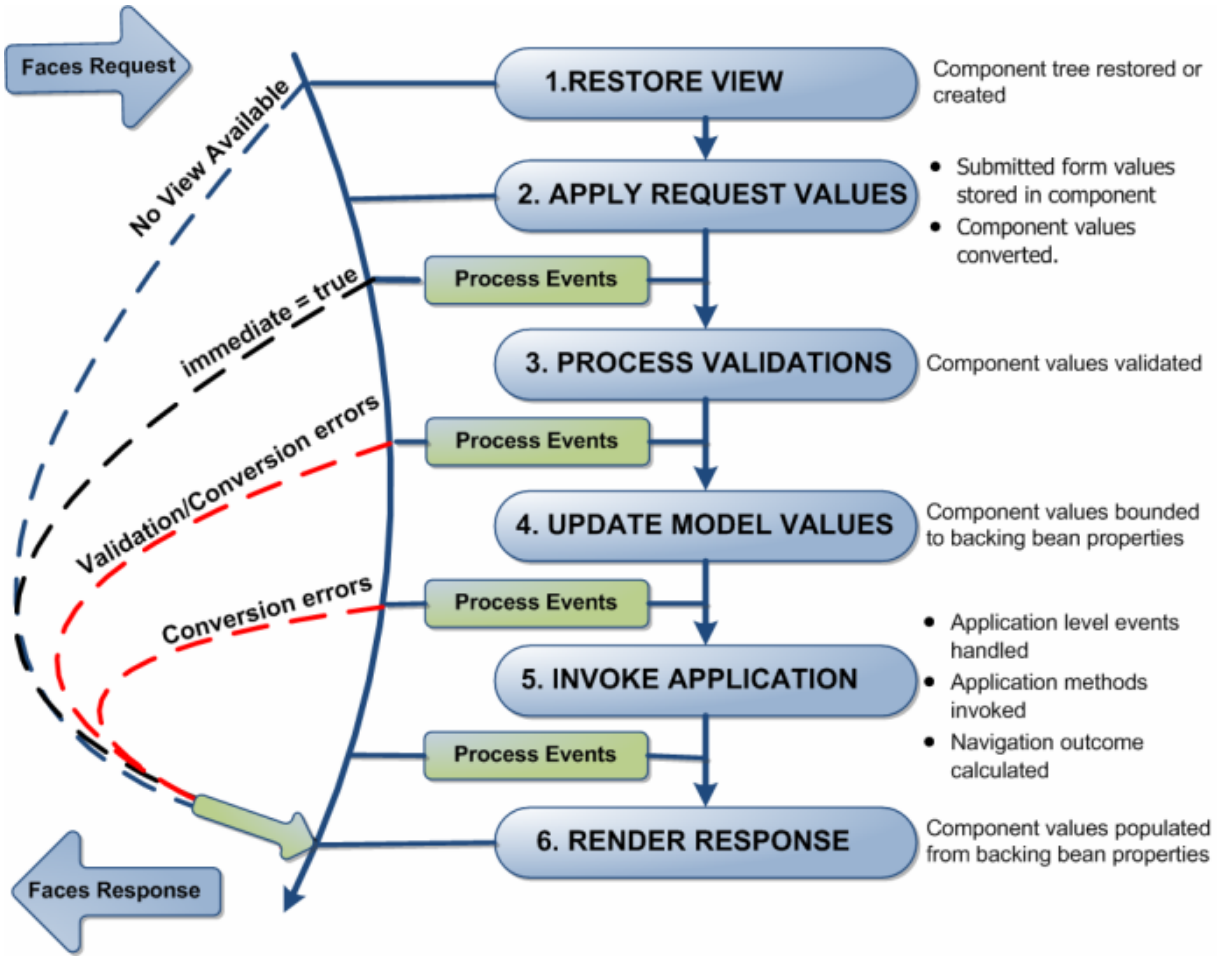
Index.xhtml

```
<?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 lang="en" xmlns="http://www.w3.org/1999/xhtml"
     xmlns:f="http://java.sun.com/jsf/core" xmlns:h="http://xmlns.jcp.org/jsf/html">
 <h:head>
      <h:outputScript name="jsf.js" library="javax.faces" target="head" />
      <title>AJAX Demo</title>
   </h:head>
   <h:body>
     <h2>Time at page loading is: #{timeBean.time()}</h2>
      <h:form>
         <h2>Last reload time is: <h:outputText value="#{timeBean.time()}" id="timeResult"/></h2>
         <h:commandButton id="submit" value="Update Time" action="nothing">
               <f:ajax render="timeResult"/>
         </h:commandButton>
      </h:form>
   </h:body>
</html>
```

JSF: Events

Based on version 2.2

The lifecycle



Events

Name of the event for which to install a listener. The following table lists the valid values for this attribute, and the corresponding event type for which the listener action is registered.

value for "type" tag attribute	**Type of event sent to listener method**
preRenderComponent	javax.faces.event.PreRenderComponentEvent
preRenderView	javax.faces.event.PreRenderViewEvent
postAddToView	javax.faces.event.PostAddToViewEvent
preValidate	javax.faces.event.PreValidateEvent
postValidate	javax.faces.event.PostValidateEvent

In addition to these values, the fully qualified class name of any java class that extends javax.faces.event.ComponentSystemEvent may be used as the value of the "type" attribute.

So , beside the values listed above , you can also use the fully qualified class name of direct known subclasses of javax.faces.event.ComponentSystemEvent

- https://javaserverfaces.java.net/nonav/docs/2.1/javadocs/javax/faces/event/ComponentSystemEvent.html
- http://www.tutorialspoint.com/jsf/jsf_event_handling.htm