

Desarrollo de componentes JSF

1 - Arquitectura JavaServer Faces



En este tema vamos a ver:

Arquitectura de JavaServer Faces

Java Specification Reference - JSR

Modelo de componentes en JSF

Ciclo de vida de una petición JSF

Introducción a EL, Tags y kits de renderizado

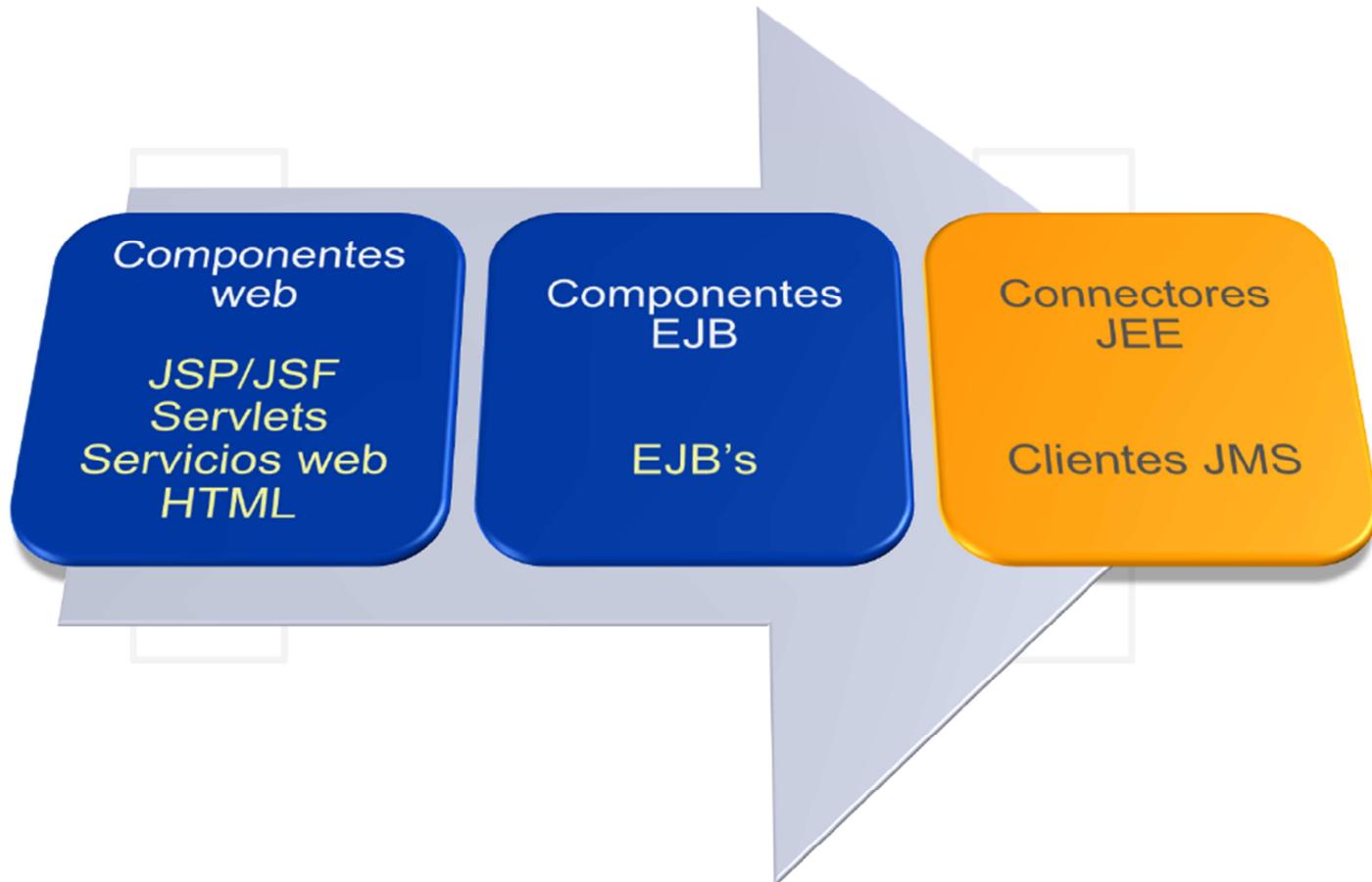
Introducción a Facelets

Introducción a los componentes JSF

Configuración inicial, Front Controller...

Elementos importantes de la configuración

Elementos de la arquitectura JEE

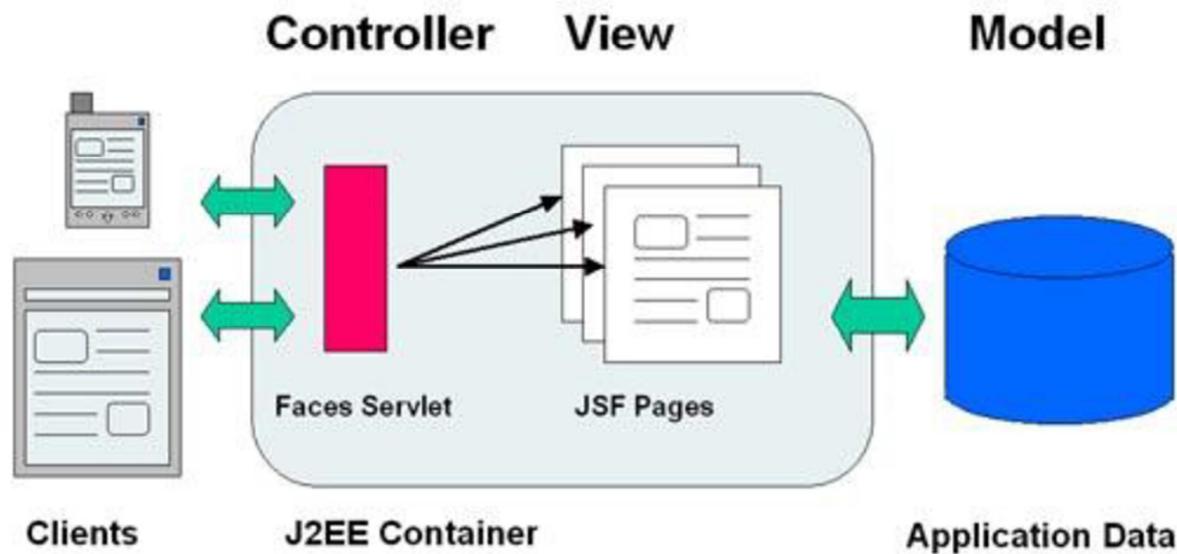


Java Specification Reference

- El grupo “jcp.org” es el encargado de dirigir/mantener las especificaciones de las API's de Java
- El **JSR-127** define y libera la especificación de JavaServer Faces 1.1 en 2004 en el que participó Craig McClanahan que lideró Apache Tomcat y Struts, entre otros.
- El **JSR-252** define y libera la especificación JSF 1.2 en 2006 que corresponde a los Servlet's 2.5/JSP 2.1.
- El **JSR-314** comienza su trabajo en Mayo de 2007 y planifica y libera la especificación JSF 2.0 en 2009 con soporte para Ajax.

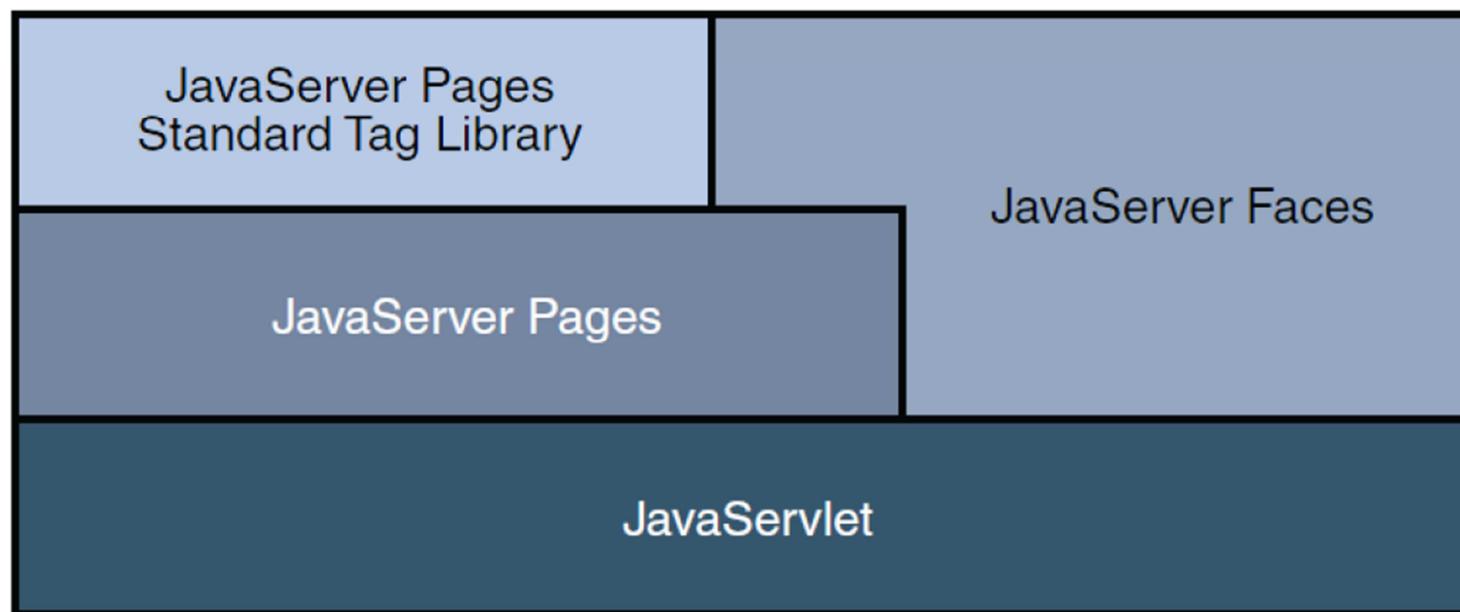
Introducción a JavaServer Faces

- Una aplicación JSF utiliza como base la estructura de una aplicación web Java EE, de esta manera, tiene los mismos requerimientos. El framework JSF añade un conjunto de elementos/componentes que permiten crear interfaces RIA y reducen el ciclo de desarrollo.

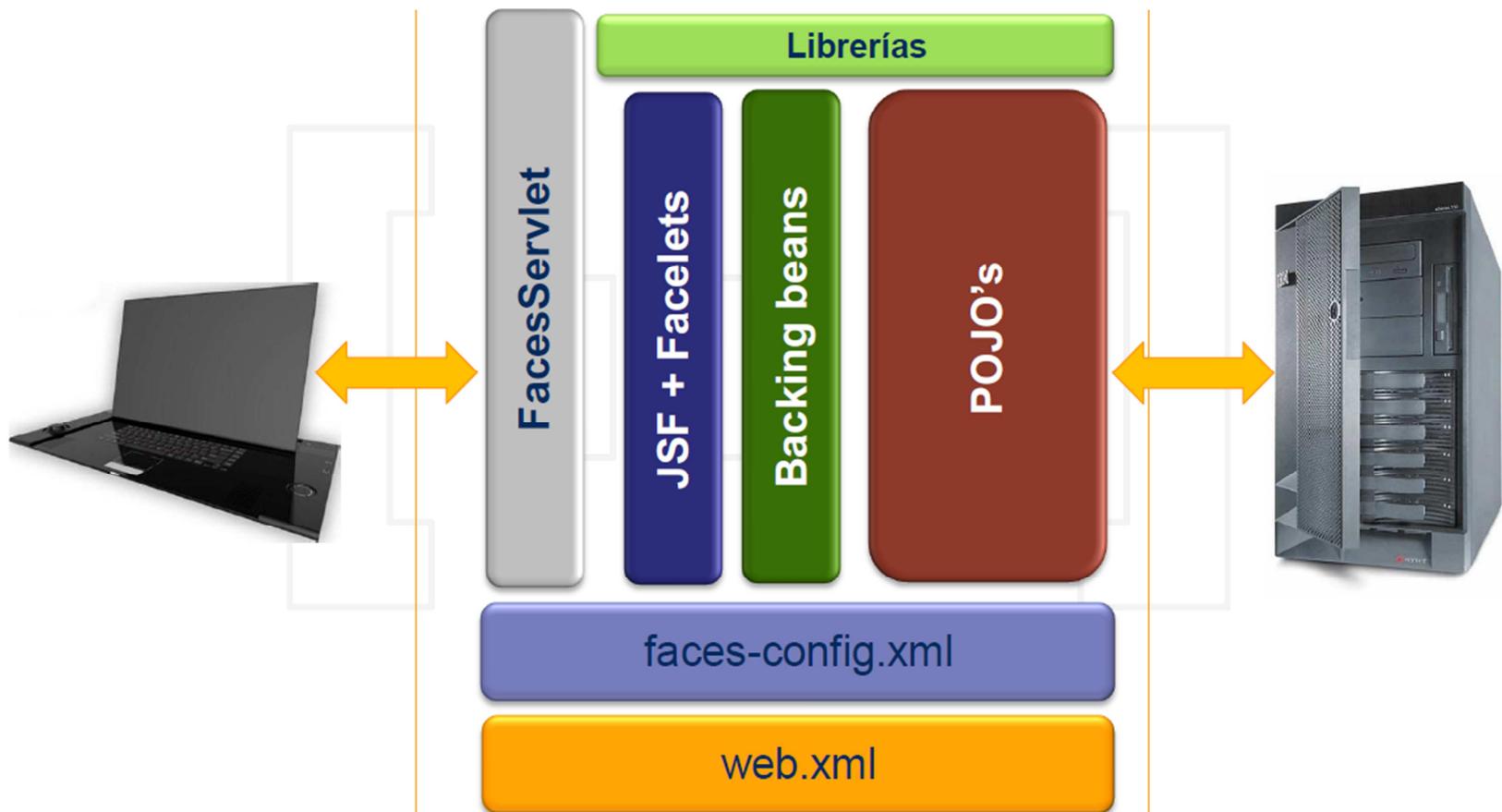


Introducción a JavaServer Faces

- Las JavaServer Faces amplían el concepto de desarrollo web basado en librerías de etiquetas, utilizando como base JSTL y EL. El desarrollo utiliza un catálogo de componentes estándar que pueden ser implementados y da la oportunidad al desarrollador de crear sus propios componentes.



Elementos de la arquitectura de JSF



Desarrollo con JavaServer Faces

- El ciclo de desarrollo habitual de una aplicación JSF, se puede dividir en las siguientes tareas:
 1. *Creación de los backing beans*
 2. *Añadir declaraciones de los backing beans creados*
 3. *Creación de las páginas web utilizando tags de componentes*
 4. *Mapeo de la instancia del FacesServlet (El Front Controller del framework JSF).*



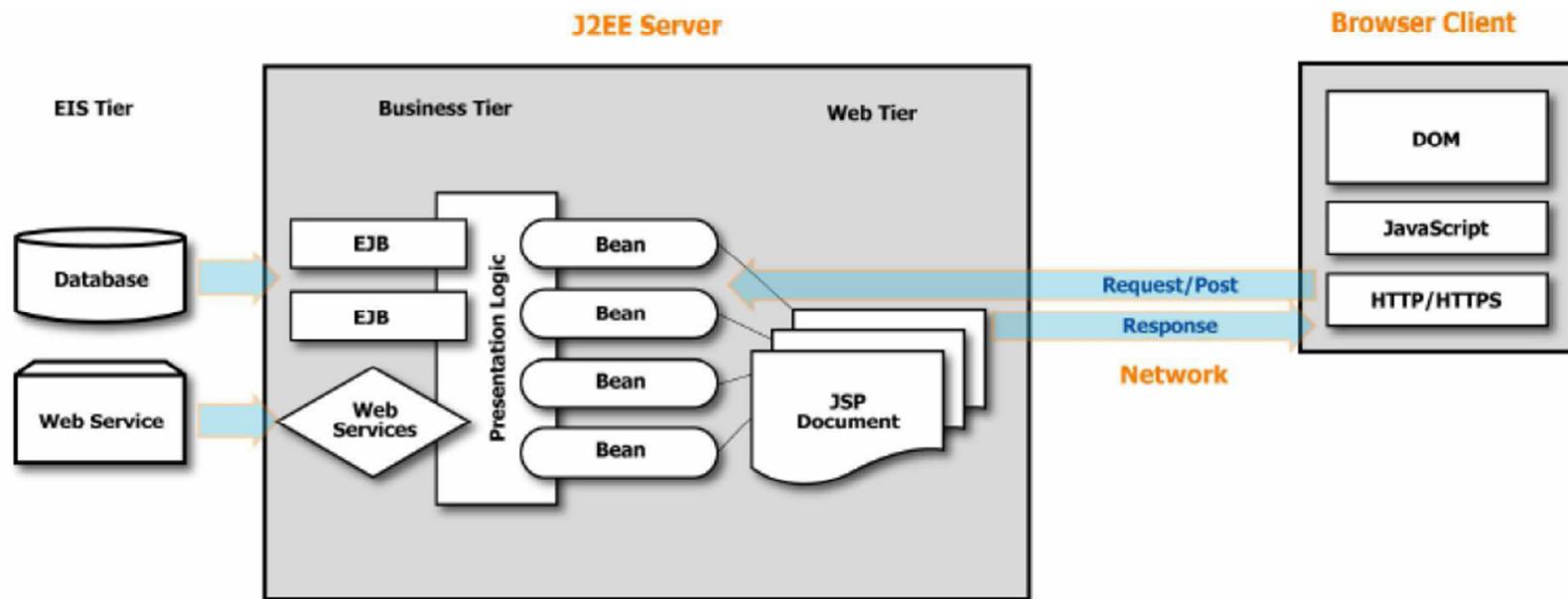
Modelo de componentes JSF

- La especificación JSF permite diferentes implementaciones de componentes como los que inicialmente utilizaba Sun en sus herramientas, como WoodStock que ha sustituido con una alianza con ICEfaces.
- Existen múltiples implementaciones de JSF en el mercado como Oracle ADF, Apache MyFaces o Jboss Seam (RICHFaces).

<p>Calendario por Defecto:</p> <p>Selecciona Fecha: 09/13/2010</p> <p>septiembre 2010</p> <table border="1"><tr><td>lun</td><td>mar</td><td>mié</td><td>jue</td><td>vie</td><td>sáb</td><td>dom</td></tr><tr><td></td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td></tr><tr><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td></tr><tr><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td></tr><tr><td>27</td><td>28</td><td>29</td><td>30</td><td></td><td></td><td></td></tr></table>	lun	mar	mié	jue	vie	sáb	dom			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30				<p>Calendario Emergente:</p> <p>Select Pattern : MM/dd/yyyy</p> <p>Selecciona Fecha: 09/13/2010</p> <p>09/13/2010</p>
lun	mar	mié	jue	vie	sáb	dom																																					
		1	2	3	4	5																																					
6	7	8	9	10	11	12																																					
13	14	15	16	17	18	19																																					
20	21	22	23	24	25	26																																					
27	28	29	30																																								

Arquitectura Java EE

- Dividida en tres capas: Presentación, Negocio e Integración



JavaServer Faces

- Es la especificación de un framework basado en componentes que trata de establecerse como estándar en el desarrollo de aplicaciones web.
- Existen varias implementaciones como Apache MyFaces, Oracle ADF, Jboss Seam, ICEfaces...
- Permite un tipo de desarrollo RAD.
- Posee un modelo de componentes gestionado por eventos.
- Utiliza componentes GUI reutilizables, sin acciones, más parecido a Swing y más lejano a Struts.
- Admite internacionalización de todas sus vistas (i18n).
- Es extensible, podemos personalizarlo y está siendo altamente utilizado en la industria.

JavaServer Faces, compuesto por:

1. Clases Java:

1. Componentes Faces: *del lado de servidor equivalentes a los componentes HTML (como input, button, check boxes...).*
2. FacesServlet (*Front Controller declarado en el web.xml*).
3. Helper clases: conversores, validadores...

2. Tag libraries:

1. *Librerías de etiquetas de Sun.*
2. *Usadas en JSP's y Facelets.*

3. Archivos de configuración:

1. *faces-config.xml: define los managed beans, navegación...*
2. *Estructura estándar web de Java.*

JavaServer Faces, etiquetas de componentes

- La API de JSF requiere un conjunto de etiquetas básicas de componentes UI que serán “renderizados” como componentes HTML.

Etiqueta	Etiqueta	Etiqueta
<h:form />	<h:commandButton />	<h:selectOneListbox />
<h:inputText />	<h:commandLink />	<h:selectOneMenu />
<h:inputTextarea />	<h:message />	<h:selectOneRadio />
<h:inputSecret />	<h:messages />	<h:selectBooleanCheckbox />
<h:inputHidden />	<h:panelGrid />	<h:selectManyCheckbox />
<h:outputLabel />	<h:panelGroup />	<h:selectManyListbox />
<h:outputLink />	<h:dataTable />	<h:selectManyMenu />
<h:outputFormat />	<h:column />	<h:outputText />

JavaServer Faces, etiquetas de componentes

- La API de JSF requiere un conjunto de etiquetas fundamentales de componentes UI.
- Un listado completo de las etiquetas JSF, podemos encontrarlo en:
<http://horstmann.com/corejsf/jsf-tags.html>.

Etiqueta	Etiqueta	Etiqueta
<f:view />	<f:converter />	<f:loadBundle />
<f:subview />	<f:convertDateTime />	<f:selectItems />
<f:facet />	<f:convertNumber />	<f:selectItem />
<f:attribute />	<f:validator />	<f:verbatim />
<f:param />	<f:validateDoubleRange />	<f:actionListener />
<f:validateLength />	<f:valueChangeListener />	<f:validateLongRange />

Introducción a Facelets

- Actualmente en la versión JSF 2.0 se ha integrado el framework de Facelets en la RI de JSF. Facelets se ha popularizado estos últimos años como un excelente framework de plantillas para crear rápidamente clientes RIA.
- Se ha creado un namespace <http://java.sun.com/jsf/faces> que permite acceder directamente a los tags para crear las plantillas.
- Se proporciona un completo soporte a EL, y puede validarse en tiempo de ejecución.
- Permite trabajar con cualquier renderkit, y no es necesario realizar una configuración adicional, porque está incluido en el framework de JSF.

JavaServer Faces uso de las etiquetas y EL

- Las páginas JSF se construyen con librerías de etiquetas y con el lenguaje de expresiones (EL: `#{xxxx.xxx}`) . Las etiquetas con sus atributos permiten personalizar el aspecto y el comportamiento de cada componente:

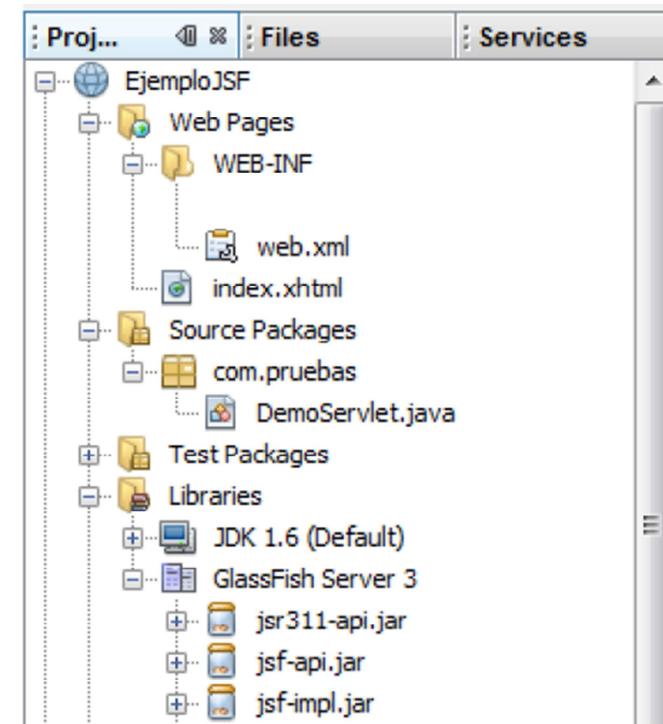
```
<h:outputText value="Referencia del cliente" rendered="true" />
```

- Las etiquetas se colocan anidadas en un formato jerárquico y conservadas así en el servidor:

```
<h: form>
    <h:panelGroup>
        <h:outputLabel for="nombre"/>
    </h:panelGroup>
</h:form>
```

Configuración inicial de JavaServer Faces

- Para trabajar con JSF 2.0, sólo es necesario añadir dos librerías, en el classpath de la aplicación. También será necesario modificar el archivo “/WEB-INF/web.xml” definiendo el FacesServlet y la máscara de mapping que será usada.
- En esta versión no es obligatorio el uso del archivo “faces-config.xml” aunque es posible utilizarlo de forma más avanzada.
- Por lo demás, JSF se apoya en la estructura estándar de un proyecto web con Java EE. Podemos usar un contenedor como Tomcat para una aplicación JSF.



Configuración inicial de JavaServer Faces

- Los primeros pasos, son declarar el FacesServlet en el archivo “WEB-INF/web.xml”, declaramos su orden de inicialización a 1 y definimos la máscara de mapping en el elemento “<url-pattern>/faces/*</url-pattern>”.

```
<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>
</servlet-mapping>
<welcome-file-list>
    <welcome-file>faces/index.xhtml</welcome-file>
</welcome-file-list>
```

Configuración del elemento ProjectStage

- Desde la versión JSF 2.0 se ha introducido un nuevo elemento de contexto: “PROJECT_STAGE” que sirve para indicarle al framework el modo de trabajo actual, alterando el comportamiento del ciclo de vida JSF, por defecto es “Development”. Podemos invocarla con `jsf.getProjectStage();`

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

- La documentación dice:
If the current stage is Development and no h:messages is present in the view, we'll add one automatically for the user. If the stage is Production we'd take no action (assuming the user would have this all corrected - no need to try to modify the tree).
- Hay 5 valores: **Production, Development, UnitTest, SystemTest y Extension**

Configuración de Managed Beans

- Un managed-bean representa una instancia de una clase creada en la aplicación. Durante la ejecución JSF procesa todos los managed-beans declarados en el archivo “/WEB-INF/faces-config.xml” o a través de annotations en el propio bean:

```
<managed-bean>
    <managed-bean-name>SessionBean1</managed-bean-name>
    <managed-bean-class>com.SessionBean1</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

```
@ManagedBean
@SessionScoped
public class CalculadoraBean {
    public CalculadoraBean() {
    }
}
```

Navegación en JSF

- Podemos definir la navegación entre páginas en el archivo XML “WEB-INF/faces-config.xml”, utilizando lo que se conoce como reglas de navegación.

```
<navigation-rule>
    <from-view-id>/greeting.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>response</from-outcome>
        <to-view-id>/success.xhtml</to-view-id>
    </navigation-case>
</navigation-rule>
```

2 - Desarrollo con componentes JSF



En este tema vamos a ver:

Patrones utilizados en JavaServer Faces

Ciclo de vida de una petición JSF

Modelo de componentes en JSF - DataTable

Introducción a Facelets

Configuración y dependencias

Concepto de plantilla

Concepto de plantilla cliente

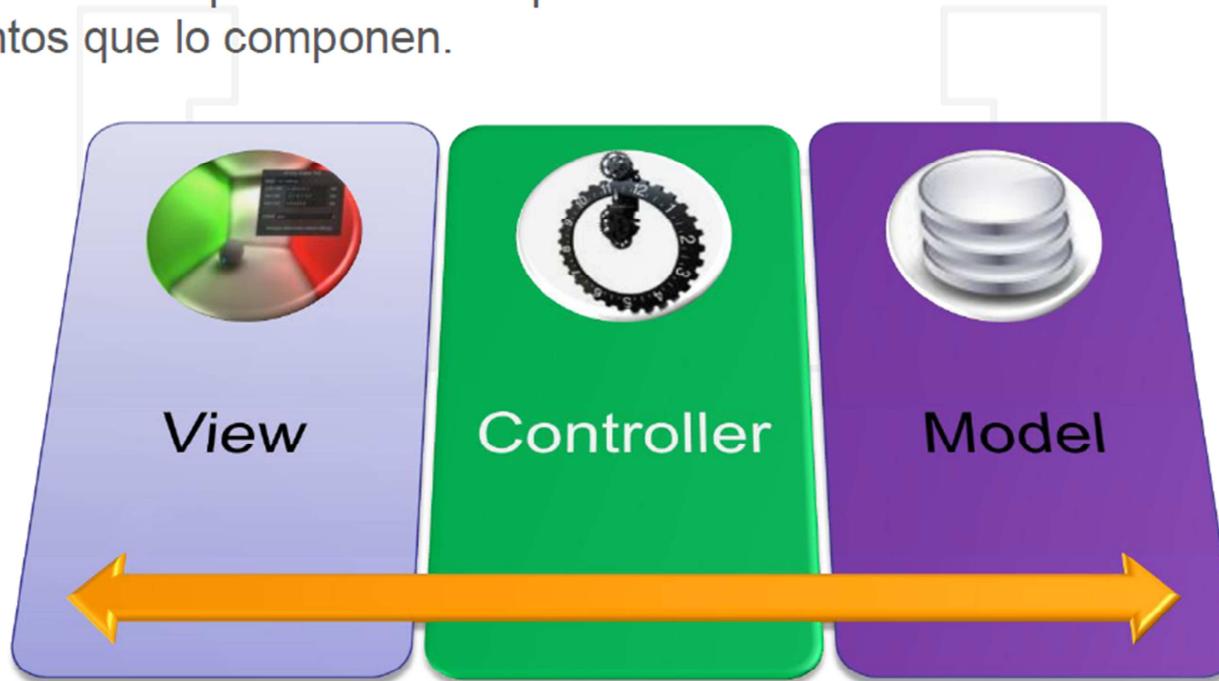
Etiquetas Facelets

Etiquetas de plantillas Facelets

Expression Language

Patrones usados en JSF

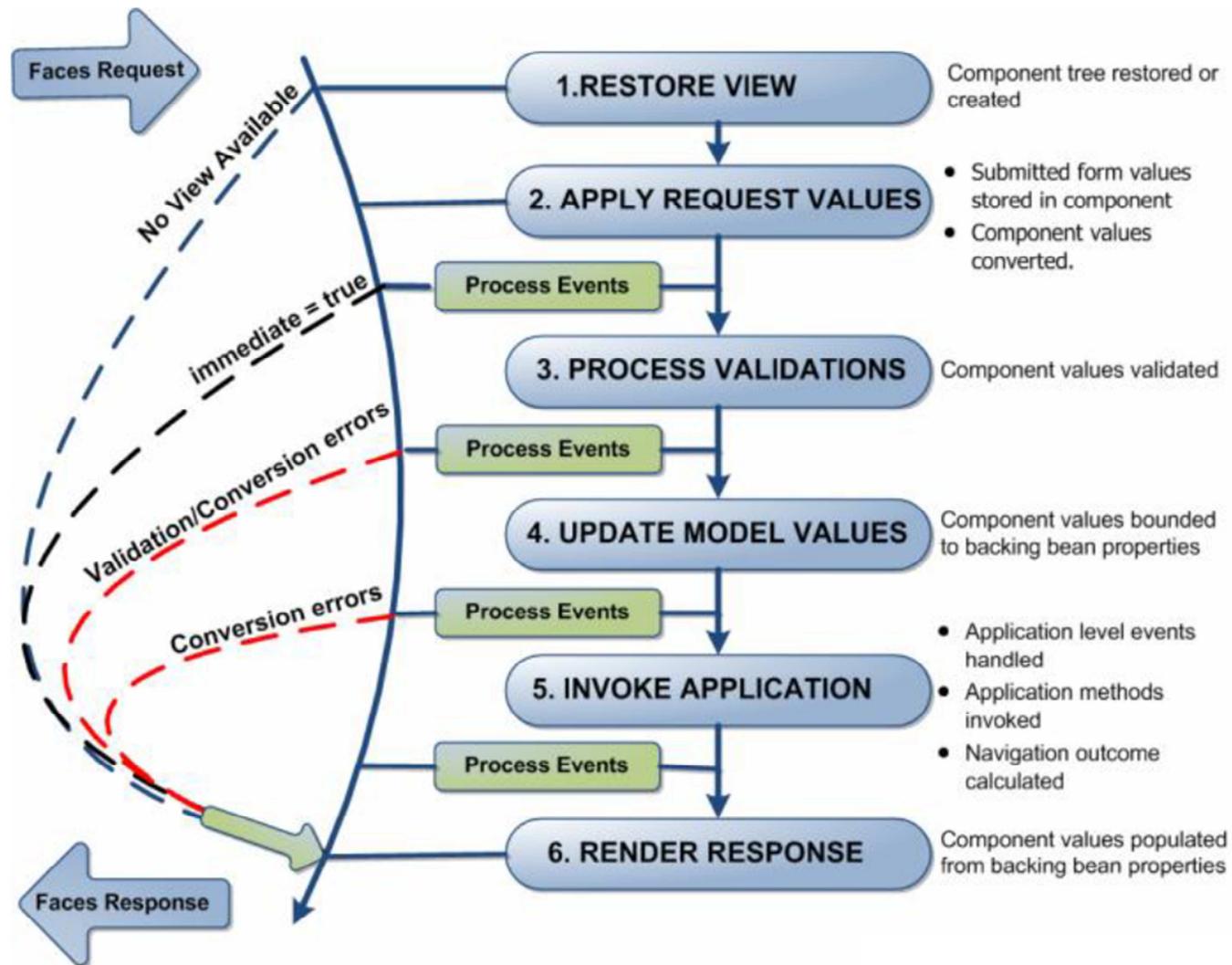
- El framework JSF implementa el patrón **Model-View-Controller** y establece una separación de responsabilidades entre cada uno de los elementos que lo componen.



Introducción

Arquitectura: La vista

- La capa de **Vista** o presentación está implementada por las páginas JSF y sus componentes que tienen la responsabilidad de realizar validaciones o detectar algunos tipos de ataques externos.
- El uso del lenguaje EL permite establecer el enlace entre componentes JSF y objetos Java.
- La API de facelets es una alternativa a JSP en el desarrollo de la interfaz gráfica de usuario.
- En la especificación JSF 2.0 es por defecto el gestor de vistas, y es hasta un 30% más rápido en el proceso de compilación de las páginas JSP.

Ciclo de vida JSF

Arquitectura: El Modelo

- La capa de **Modelo** es la responsable de la gestión de la lógica de negocio.
- En la arquitectura JSF está llevada a cabo por los managed beans.
- Estos son declarados en el archivo faces-config.xml y están implementados por POJO's con un constructor por defecto sin parámetros.
- Pueden ser usados conjuntamente con una capa de persistencia como Hibernate, JPA, Oracle Toplink...

Arquitectura: El Controlador

- Es el elemento del patrón que toma las decisiones de navegación entre las diferentes vistas de la aplicación web.
- JSF implementa el controlador a través de los navigation handler en combinación con los backing beans.
- El navigation handler por defecto del framework JSF proporciona las reglas de navegación automáticamente.
- Un controlador debe evitar incluir lógica de negocio o de presentación, ya que son responsabilidades de las otras dos capas.

Componente JSF: DataTable

```
...
<h:body>
    <h3 style="font-family: arial; font-variant: small-caps; color: #336699">DataTable Demo</h3>

    <h:dataTable var="row" value="#{JDBCEmpTable.empResultData}" border="1">
        <h:column>
            <f:facet name="header">#Empno</f:facet>#{row.empno}
        </h:column>
        <h:column>
            <f:facet name="header">Name</f:facet> #{row.ename}
        </h:column>
        <h:column>
            <f:facet name="header">Job</f:facet> #{row.job}
        </h:column>
    </h:dataTable>

    <p />
    <h:link outcome="index.jsp" value="Home" />
</h:body>
...
```

Componentes de entrada JSF: DataTable

```
...
<h:body>
    <h3 style="font-family: arial; font-variant: small-caps; color: #336699">DataTable Demo</h3>

    <h:dataTable var="row" value="#{JDBCEmpTable.empResultData}" border="1">
        <h:column>
            <f:facet name="header">#Empno</f:facet>#{row.empno}
        </h:column>
        <h:column>
            <f:facet name="header">Name</f:facet> #{row.ename}
        </h:column>
        <h:column>
            <f:facet name="header">Job</f:facet> #{row.job}
        </h:column>
    </h:dataTable>

    <p />
    <h:link outcome="index.jsp" value="Home" />
</h:body>
...
```

Introducción

¿Qué son las facelets?

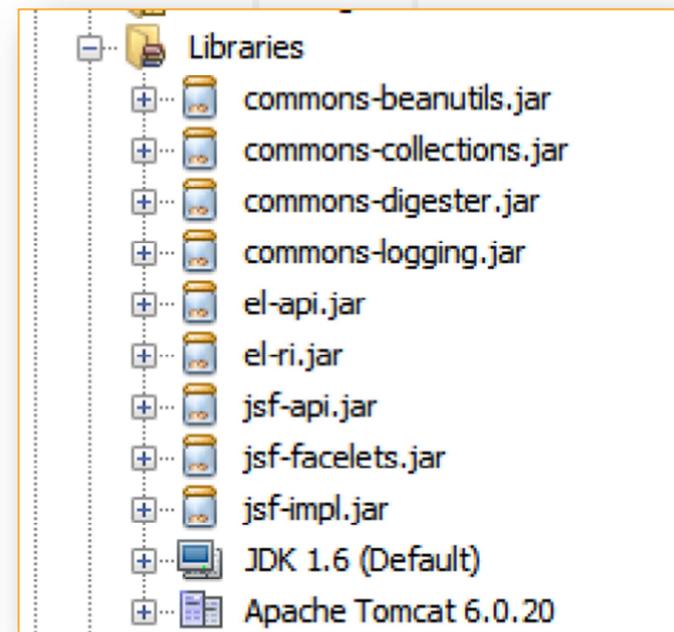
- Es un framework ligero basado en plantillas que se apoya en las JavaServer Faces fácilmente integrable y permite crear componentes personalizados.

Características

- Soporta JSF 1.1, JSF 1.2 y está integrado con JSF 2.0.
- Desarrollo rápido y fácil de plantillas para JSF.
- Etiquetas para la captura y presentación de errores.
- Pleno soporte para EL incluyendo funciones.
- No hay archivos de configuración extras.
- Reserva del atributo “jsfc” para decorar componentes HTML.
- Trabaja con cualquier kit de renderizado.
- No establece dependencias con ningún contenedor.

Dependencias

- Necesita los siguientes jar's para trabajar: jsf-facelets.jar, el-ri.jar y el-api.jar. Pueden ser descargados desde “<http://facelets.dev.java.net>”.
- Soporta JSF 1.1, 1.2 y 2.0.
- Utiliza la nueva API de EL y es una dependencia necesaria para poder utilizar facelets.
- Se declara en el deployment descriptor estándar de la aplicación web.



Configuración

- En el archivo web.xml se declara la utilización de facelets definiendo el FacesServlet:



web.xml

```
<!-- Faces Servlet -->
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Faces Servlet Mapping -->
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

Configuración

- Es posible añadir ciertos atributos para el ciclo de desarrollo que facilitan el uso de Facelets:

web.xml

```
<context-param>
    <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
    <param-value>.xhtml</param-value>
</context-param>
<context-param>
    <param-name>facelets.DEVELOPMENT</param-name>
    <param-value>true</param-value>
</context-param>
<context-param>
    <param-name>facelets.REFRESH_PERIOD</param-name>
    <param-value>2</param-value>
</context-param>
```

Configuración

- Facelets reemplaza el ViewHandler por defecto de JSF con FaceletViewHandler con su propia implementación.
- Facelets puede trabajar con múltiples tipos de frameworks, para añadir su funcionalidad a JSF es necesario incluir la siguiente declaración en el archivo faces-config.xml:

faces-config.xml

```
<faces-config>
    <application>
        <view-handler>com.sun.facelets.FaceletViewHandler</view-handler>
    </application>
</faces-config>
```

Utilización de Facelets

- Facelets promueve el uso de plantillas para crear aplicaciones web y poder reutilizarlas y mantenerlas fácilmente. Podemos distinguir dos roles:



Introducción

Plantilla

- Marcamos con “<ui:insert>” las zonas que va a cambiar. Ejemplo de creación de una plantilla facelets:

plantilla.xhtml

```
<body>
    <h1>
        <ui:insert name="title">Titulo por defecto</ui:insert>
    </h1>
    <p>
        <ui:insert name="body">Documento por defecto</ui:insert>
    </p>
</body>
```

Cliente plantilla

- Marcamos con “<ui:composite” el contenido que va a ser insertado en la plantilla. Ejemplo de creación de una cliente plantilla facelets:

plantilla.xhtml

```
<ui:composition template="/tplantilla.xhtml">
    <ui:define name="title">Title text</ui:define>
    <ui:define name="body">
        Ejemplo del contenido que será insertado en
        la sección del body de la plantilla
    </ui:define>
</ui:composition>
```

Introducción

Etiquetas Facelets

- Situaremos las plantillas en la carpeta /WEB-INF y las protegeremos de usuarios que desean acceder a ellas directamente.
- Es posible utilizar EL directamente al crear las plantillas o documentos con #{EL} o con \${EL}, es aconsejable establecer una convención para el grupo de desarrollo.
- Podemos utilizar diferentes editores de HTML para trabajar con Facelets, ya que las plantillas únicamente necesitan los tags “<ui:” y el resto se descarta.
- La etiqueta “<ui:composition” no muestra contenido fuera de la misma.

Introducción

Etiquetas Facelets

Tag Libraries Supported by Facelets

Tag Library	URI	Prefix	Example	Contents
JavaServer Faces Facelets Tag Library	http://java.sun.com/jsf/facelets	ui:	ui:component ui:insert	Tags for templating
JavaServer Faces HTML Tag Library	http://java.sun.com/jsf/html	h:	h:head h:body h:outputText h:inputText	JavaServer Faces component tags for all UIComponents.
JavaServer Faces Core Tag Library	http://java.sun.com/jsf/core	f:	f:actionListener f:attribute	Tags for JavaServer Faces custom actions that are independent of any particular RenderKit.

Introducción

Etiquetas Facelets

- El resto de etiquetas soportado por el framework de Facelets es el siguiente:



Tag Libraries Supported by Facelets

Tag Library	URI	Prefix	Example	Contents
JSTL Core Tag Library	http://java.sun.com/jsp/jstl/core	c:	c:forEach c:catch	JSTL 1.1 Core Tags
JSTL Functions Tag Library	http://java.sun.com/jsp/jstl/functions	fn:	fn:toUpperCase fn:toLowerCase	JSTL 1.1 Functions Tags

Introducción

Etiquetas de plantilla Facelets

- Relación de etiquetas que se utilizan para crear plantillas, indicando su funcionalidad:

Facelets Templating Tags

Tag	Function
ui:component	Defines a component that is created and added to the component tree.
ui:composition	Defines a page composition that optionally uses a template. Content outside of this tag is ignored.
ui:debug	Defines a debug component that is created and added to the component tree.
ui:define	Defines content that is inserted into a page by a template
ui:decorate	Similar to composition tag but does not disregard content outside this tag.

Introducción

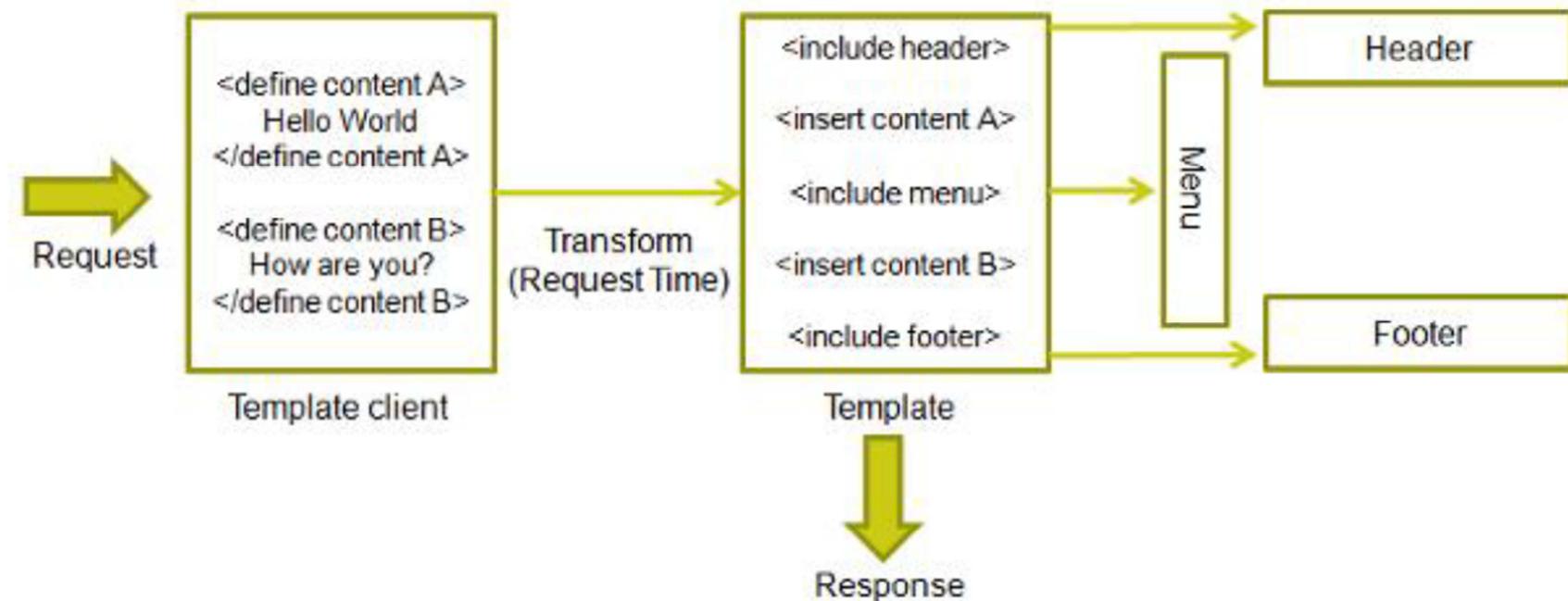
Etiquetas de plantilla Facelets

- Relación de etiquetas que se utilizan para crear plantillas, indicando su funcionalidad:

Facelets Templating Tags

Tag	Function
ui:fragment	Similar to component tag but does not disregard content outside this tag.
ui:include	Encapsulate and reuse content for multiple pages.
ui:insert	Inserts content into a template.
ui:param	Used to pass parameters to an included file.
ui:repeat	Used as an alternative for loop tags such as c:forEach or h:dataTable.
ui:remove	Removes content from a page.

Trabajando con Facelets



Expression Language (EL)

- Incluido en la API de JSP desde la versión 1.4. No es necesario importar nada, incorporado por defecto.
- Es usado para asociar los valores del modelo con la vista.
 - *ValueBinding*.
 - *MethodBinding*.
- Diferencias entre las versiones JSF 1.1 y JSF 1.2
 - *Sintáxis \${expresión}* Evaluación en Runtime.
 - *Sintáxis #\${expresión}* Evaluación en tiempo de compilación.
- Cuando una expresión se evalúa se localiza el bean y se resuelve la propiedad/método.

Objetos implícitos en expression Language (EL)

- Al utilizar el Expression language disponemos de un conjunto bien conocido de objetos implícitos:

Objeto	Tipo
cookie	Map
facesContext	FacesContext
header	Map
headerValues	Map
param	Map
paramValues	Map
requestScope	Map
sessionScope	Map
applicationScope	Map
initParam	Map
view	UIViewRoot

Operadores en expression Language (EL)

- Podemos construir expresiones con Expression language utilizando un conjunto de operadores:

Familia	Operadores
Aritmético	+, -, *, /, %
Relacional	==, !=, <>, <=, >=
Logicos	&&, , !
Condicionales	a?B:C (ternario)
Vacios	empty

```
<h:panelGroup  
    rendered="#{mybean.value not empty and mybean.rendered}">  
    ...  
</h:panelGroup>
```