

JORNADAS RIA OCTUBRE 2009

# Java Server Faces: RichFaces y IceFaces



0 Un poco de historia

1



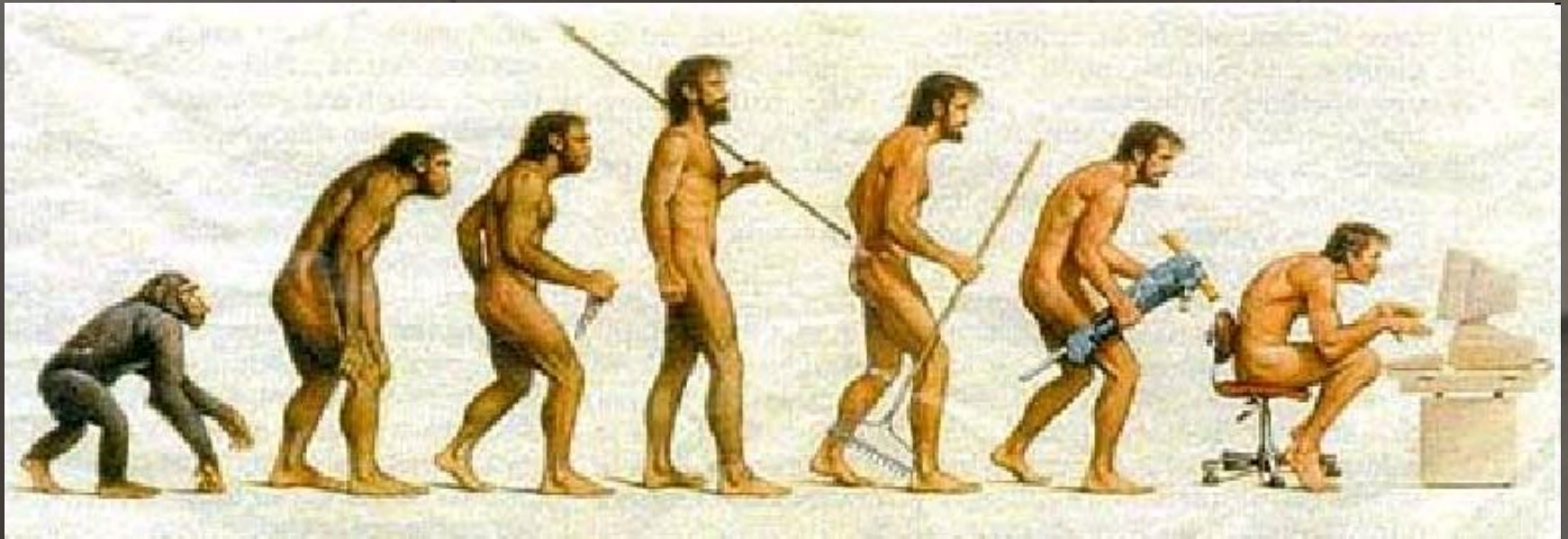


# Algunas preguntas transcendentales sobre JSF

**¿Quiénes somos?**

**¿De dónde venimos?**

**¿A dónde vamos?**



## **Evolución del desarrollo web:**

- **Desarrollos centrados en páginas (jsp a jsp)**

JSPs

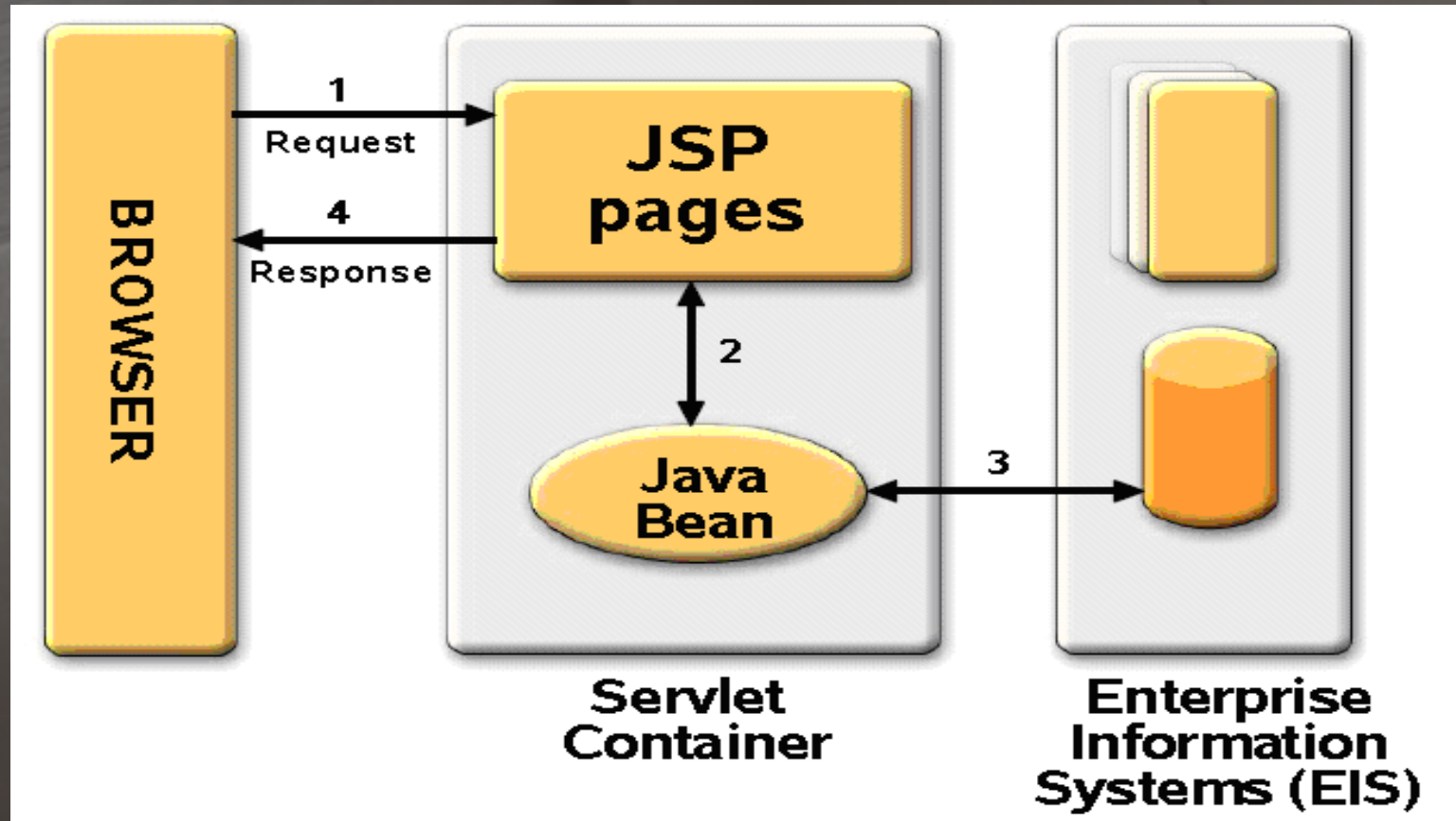
- **Desarrollos centrados en servlet**

MVC – Model 2

Tecnologías utilizadas Servlets, Jsp, Struts

# ¿De dónde venimos?

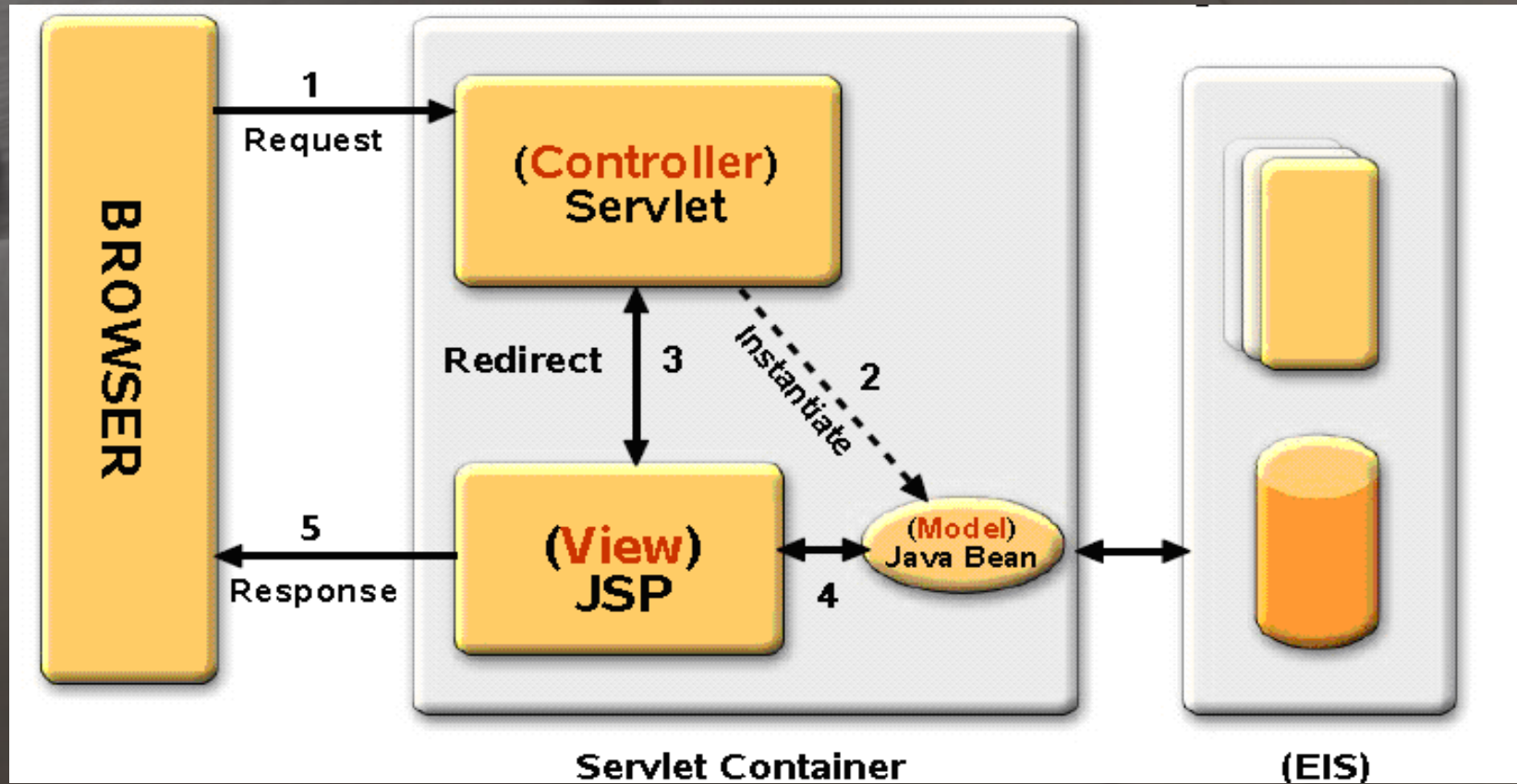
## Desarrollos centrados en páginas (jsp a jsp)





# ¿De dónde venimos?

## Desarrollos centrados en servlets (servlet a jsp)



# ¿A dónde vamos?

- **Aplicaciones ricas**

- Dinamismo
- Aplicaciones de escritorio en entorno web.

- **Desarrollo basado en componentes**

## **LLEGADA DE JSF**

- **Desarrollos ágiles**

- Agile Development
- Test Driven Development (TDD)
- Scrum

# JSF no es

- Librería (API) Java
- Librería AJAX
- Arquitectura de desarrollo
- Implementación de J2EE → Struts, Spring, Hibernate



- Una **especificación** de Sun que busca el desarrollo de aplicaciones J2EE orientadas a componentes con un modelo basado en eventos.
- Mediante herramientas drag & drop de componentes.

Existen diferentes implementaciones:

- SUN: JSF RI
- ORACLE: ADF
- APACHE: MyFaces
- JBOSS: RichFaces
- ICESOFT: IceFaces



# 0 Introducción a JSF

## 2



## Algunos datos de interés

La especificación fue desarrollada bajo la Java Community Process (JCP) :

### - **JSR 127** :

JSF 1.0 (2004): Soporta Servlet 2.3 y JSP 1.2

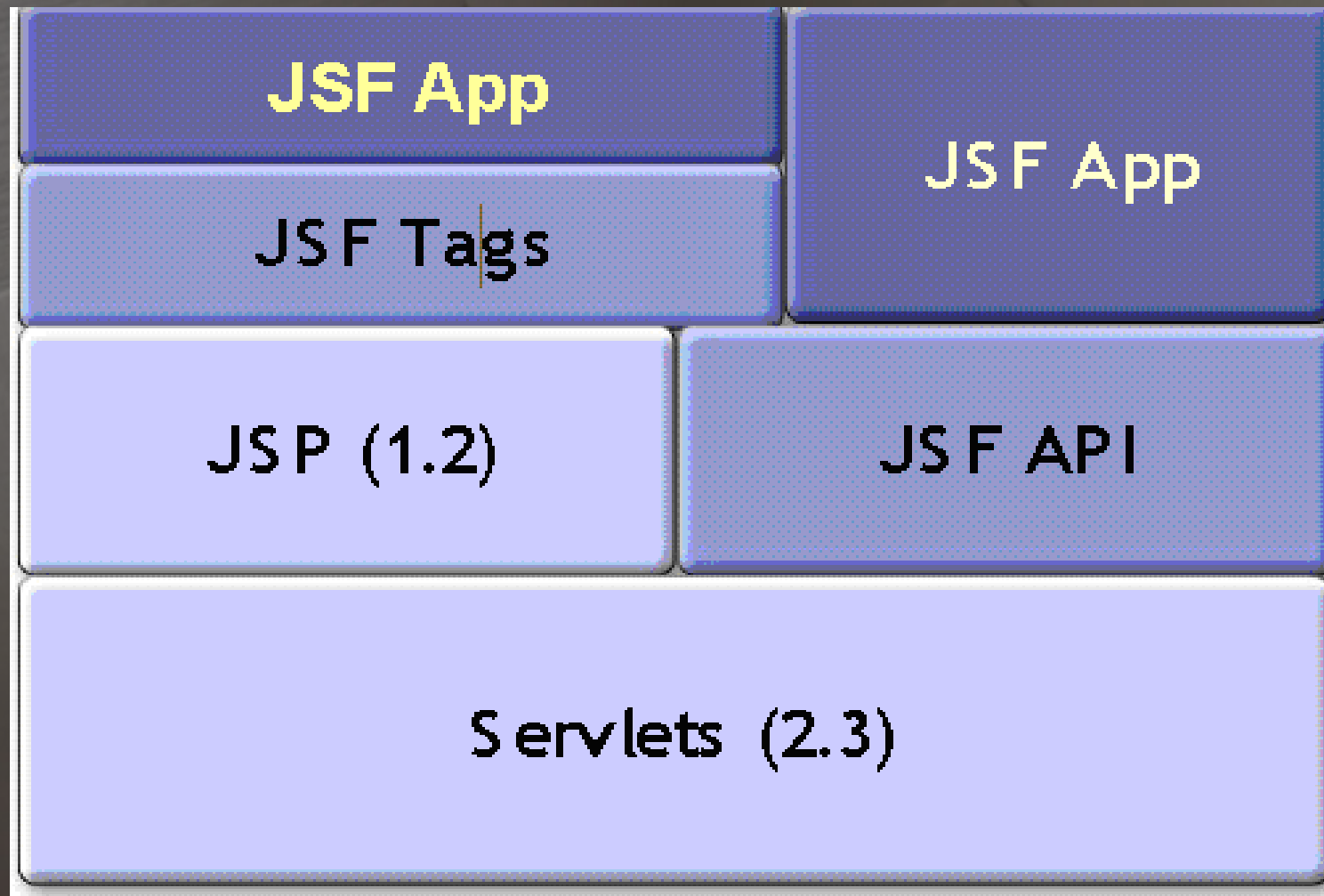
JSF 1.1 (2004): Resolución de problemas

### - **JSR 252** :

JSF 1.2 (2006). Soporta JEE 2.5 y JSP 2.1. Por tanto, necesario un tomcat 6.0

Ultimándose JSF 2.0 junto con JEE 2.6





# Objetivos (1)

La especificación persigue:

- Ser el **estándar para desarrollo** de aplicaciones web de Java.
- Creado sobre Servlet API.
- Marco de trabajo **orientado a componentes**.
- Componentes almacenados en el servidor.
- Fácil uso de componentes de terceros.
- **Modelo de programación orientada a Eventos.**



## Objetivos (2)

- Eventos generados por el usuario son manejados en el servidor.
- Manejo de navegación.
- Sincronizado automático de componentes.
- Soporte a múltiples dispositivos en cliente.
- Arquitectura extensible.
- Soporte a Internacionalización.
- Soporte a múltiples herramientas

Todo ello permita un desarrollo rápido de aplicaciones.

## Nuevas funcionalidades JSF 1.2 (1)

1. Unified Expression Language: Lenguaje de expresiones unificado que evita los problemas del uso de JSP EL y JSF EL.
2. Soporte a AJAX: Sencillez a la hora de utilizar AJAX
3. Nueva creación del árbol de componentes y mejor iteración con JSPs mejorando el FacesViewHandler de JSP y las clases que representan etiquetas personalizadas de JSP.
4. Integración con JSTL. Utilización correcta de `<c:forEach>`
5. Publica un único ResourceBundle para toda la aplicación mediante el EL. En anteriores versiones creaba un Resource Bundles por petición



## Nuevas funcionalidades JSF 1.2 (2)

5. Publica un único ResourceBundle para toda la aplicación mediante el EL. En anteriores versiones creaba un Resource Bundles por petición.
6. Uso de múltiples kits de renderizado (render kits)
7. Proporciona Schemas XML para la configuración de ficheros, en lugar de DTDs.
8. Mejoras de seguridad a la hora de guardar el estado en la parte cliente.
9. Resuelto problema de la duplicidad al pulsar un botón 2 veces.
10. Resueltos problemas relacionados con portlets.

## ¿ Por qué utilizar JSF ? (1)

- ✗ Es un estándar de Sun.
- ✗ Gran cantidad de implementaciones y componentes existentes.
- ✗ Nos permite desarrollar rápidamente aplicaciones de negocio dinámicas si la necesidad de utilizar JavaScript.
- ✗ Soporte para diferentes dispositivos (web, mobile, etc)

## ¿ Por qué utilizar JSF ? (2)

- × El código JSF con el que creamos las vistas (etiquetas jsp) es muy parecido al HTML estándar. Lo pueden utilizar fácilmente desarrolladores y diseñadores web.
- × JSF resuelve validaciones, conversiones, mensajes de error e internacionalización (i18n).
- × Permite desarrollo a medida de nuevos componentes.



# Riesgos de JSF

- ✗ La forma de desarrollo de aplicaciones cambia. Hay que adaptarse a esta nueva forma → mas parecido al desarrollo de aplicaciones de escritorio.
- ✗ Uso de etiquetas personalizadas.
- ✗ Ciclo de vida pesado.
- ✗ Tecnología pesada en servidor.
- ✗ Encontrar un buen editor visual.

# Riesgos de JSF

- ✗ La forma de desarrollo de aplicaciones cambia. Hay que adaptarse a esta nueva forma → mas parecido al desarrollo de aplicaciones de escritorio.
- ✗ Uso de etiquetas personalizadas.
- ✗ Ciclo de vida pesado.
- ✗ Tecnología pesada en servidor.
- ✗ Encontrar un buen editor visual.

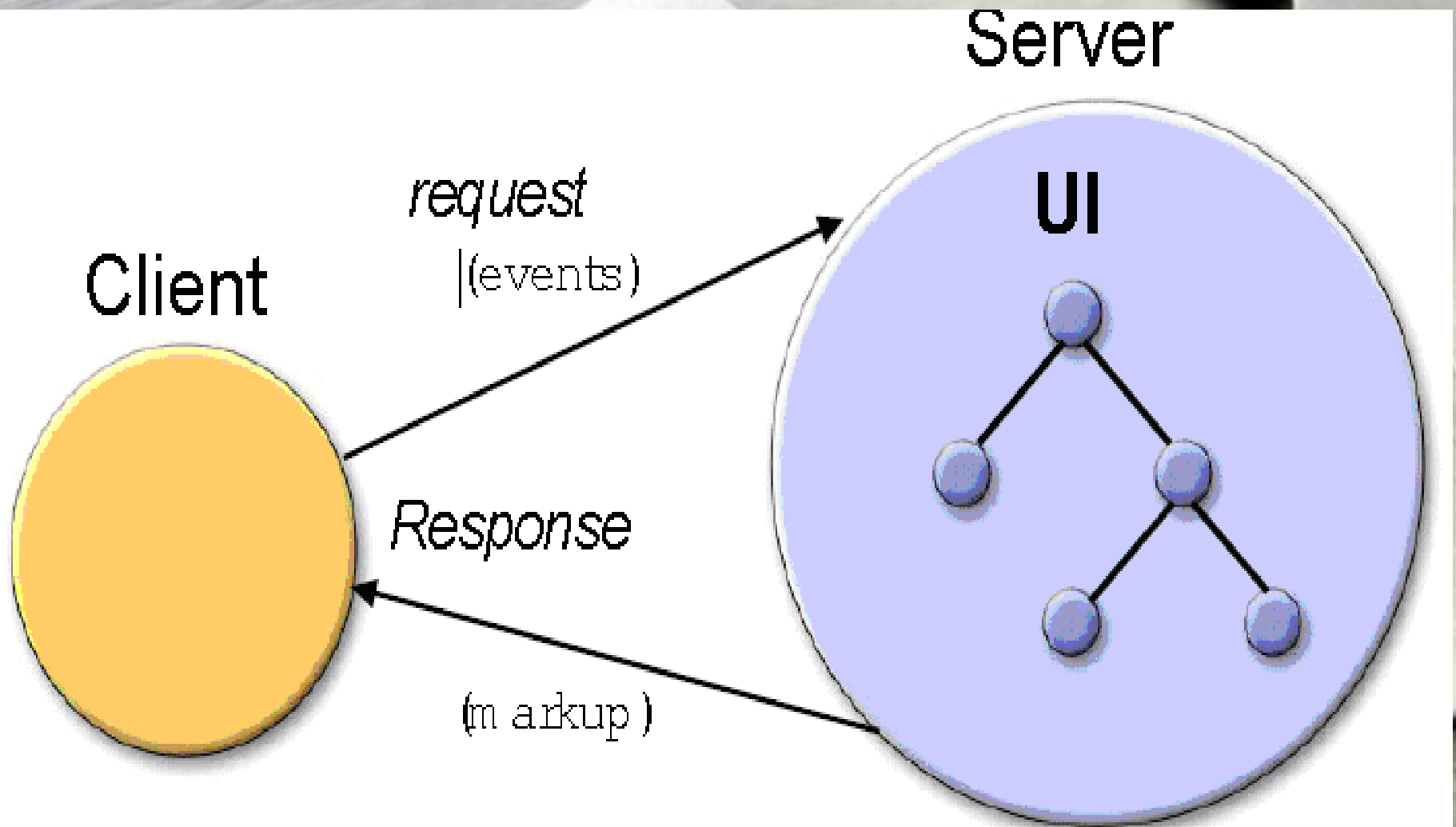
# 03 Arquitectura y Elementos



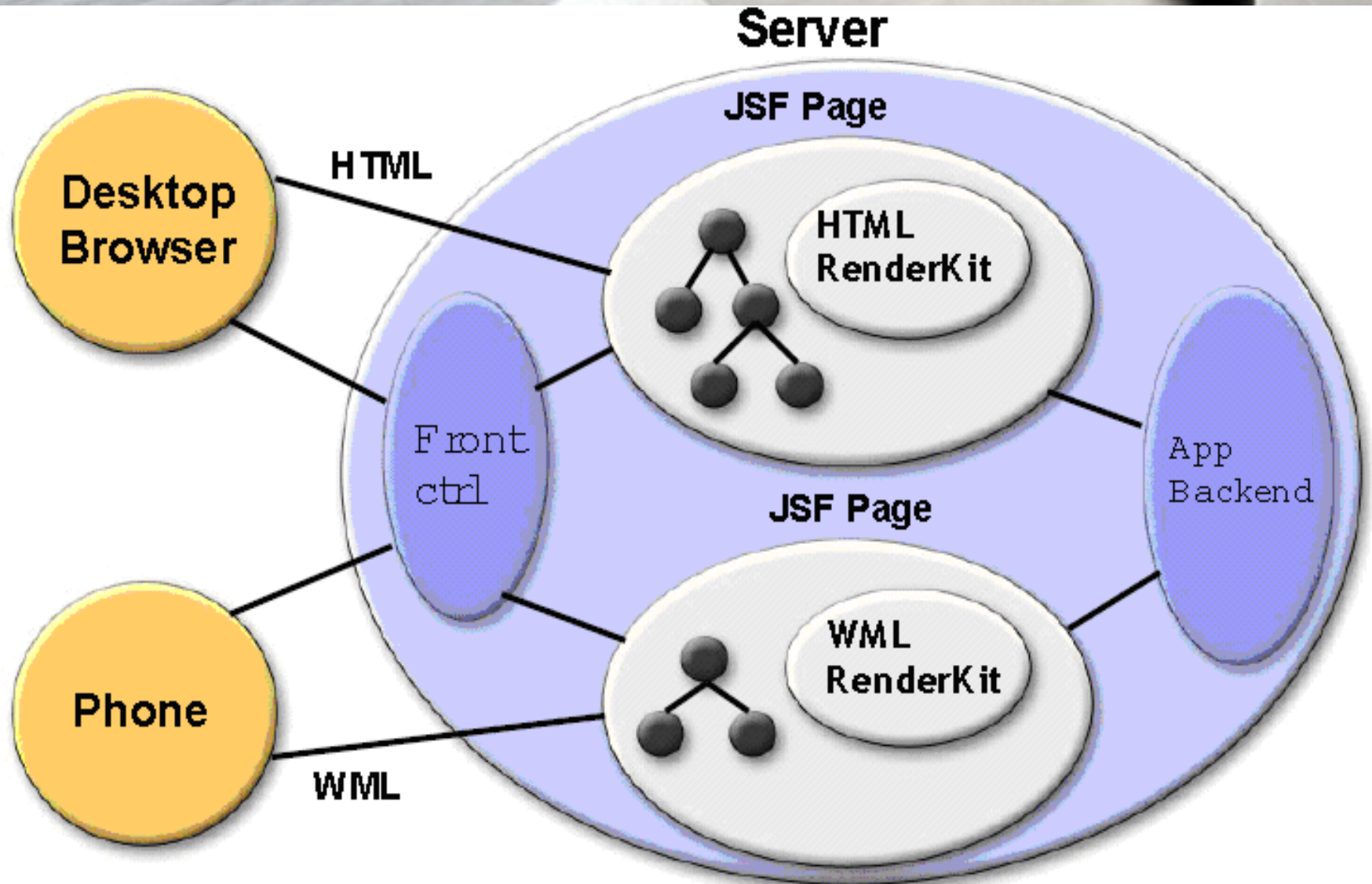
# Elementos principales (1)

- ✗ Vista: Mediante JSPs con etiquetas JSF.
- ✗ Servidor : Árbol de componentes (UIView)
  - Conversión y validación
  - Renderizado
- ✗ Clases Java (Backing Bean) que conectan el árbol de componentes con la parte de negocio.
- ✗ Fichero de configuración (faces-config.xml)
- ✗ Conjunto de objetos personalizados (Validadores y conversores)

## Elementos principales (2)





















# Arquitectura





# Componentes Estándar (1)

























JSF CORE

-  actionListener
-  attribute
-  convertDateTime
-  convertNumber
-  converter
-  facet
-  loadBundle
-  param
-  selectItem
-  selectItems
-  subview
-  validateDoubleRange
-  validateLength
-  validateLongRange
-  validator
-  valueChangeListener
-  verbatim
-  view

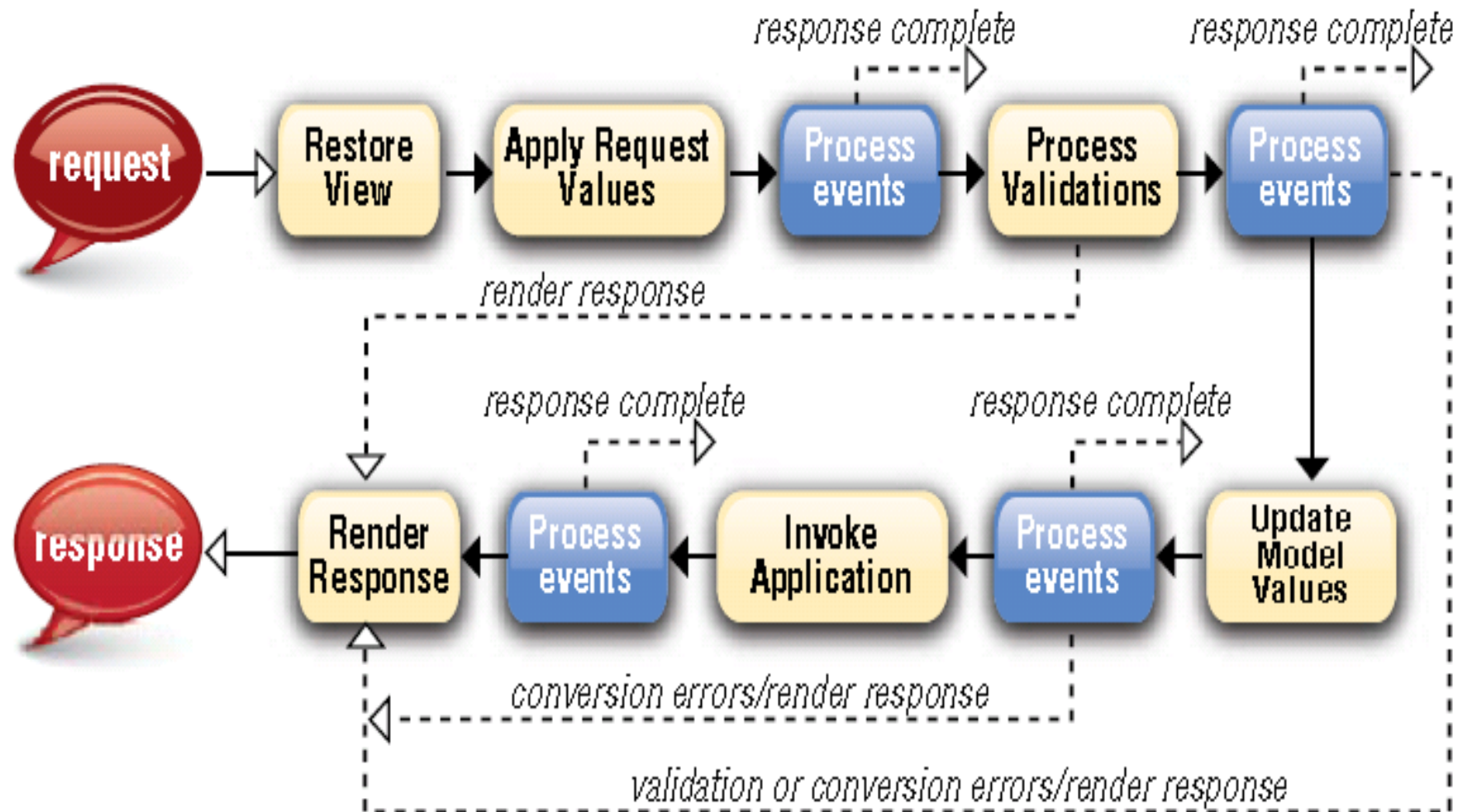
# Componentes Estándar (2)

JSF HTML

Componentes básicos  
HTML

Palette	
	Column
	Command Button
	Command Link
	Data Table
	Form
	Graphic Image
	Hidden Input
	Message
	Messages
	Output Format
	Output Label
	Output Text
	Panel Grid
	Panel Group
	Secret Input
	Select Boolean Checkbox
	Select Many Checkbox
	Select Many Listbox
	Select Many Menu
	Select One Listbox
	Select One Menu
	Select One Radio
	Text Input
	Textarea Input

# Ciclo de Vida (1)





## 1. Restore view

Puede ocurrir:

- Si es la primera vez que se accede a la página, se crea el árbol de componentes.
- Si ya existe árbol anterior se recupera

En ambos casos se muestran los componentes asociados al árbol.

## 2. Apply request values

Envío de campos del formulario que se almacenan en el componente. Se produce la conversión de valores.

## 3. Process Validations

Cálculo de los valores de los componentes. Se produce la validación de los mismos.

### **4. Update model values**

El valor de cada componente se asocia a las propiedades del backing bean.

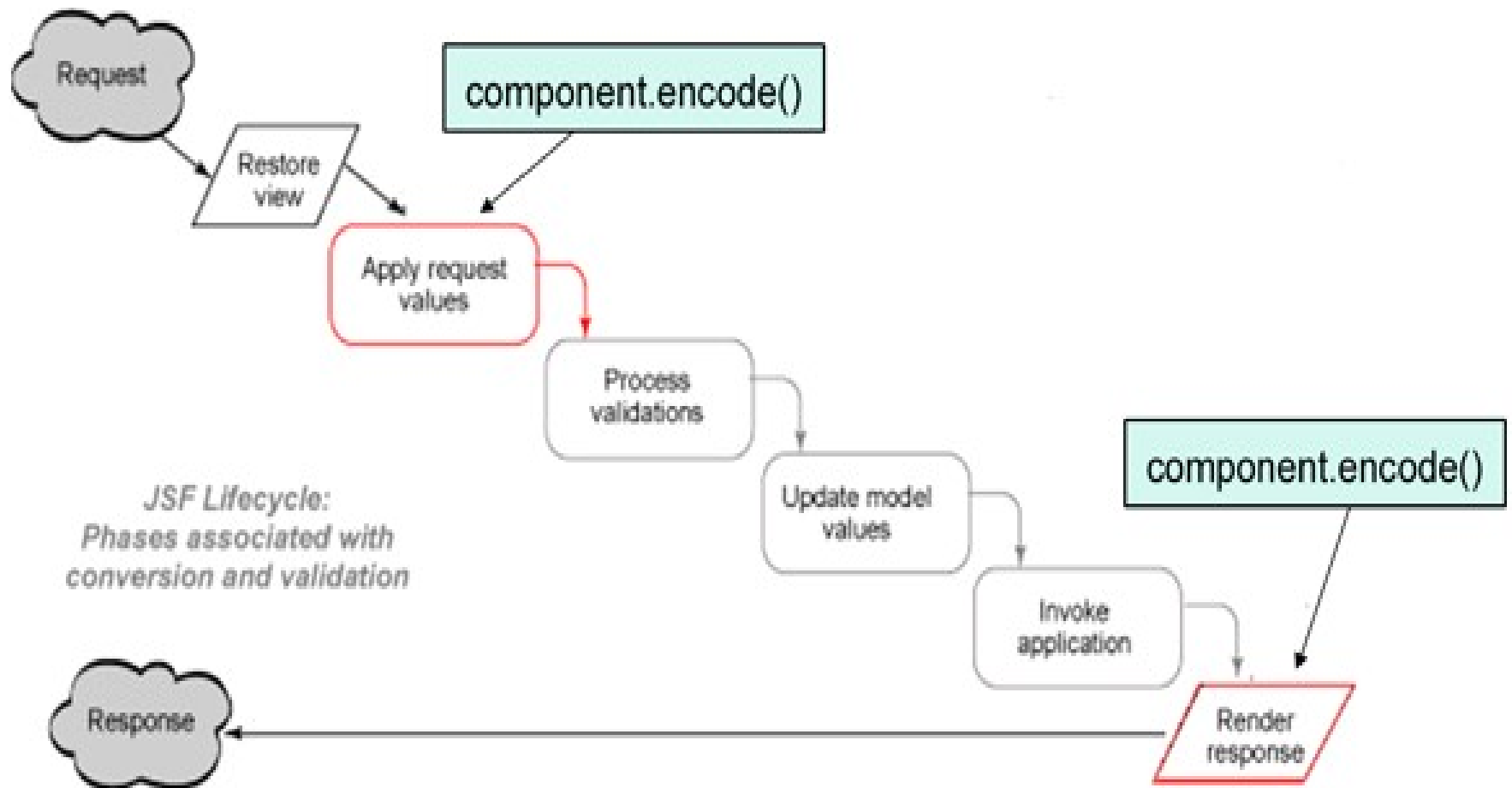
### **5. Invoke application**

Se manejan los eventos, se invocan los métodos del backing bean y se calculan las reglas de navegación.

### **6. Render response**

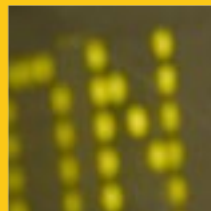
Se actualizan los valores de los componentes desde las propiedades del backing bean y se genera la respuesta.

## Ciclo de Vida (4)





0 RichFaces  
4

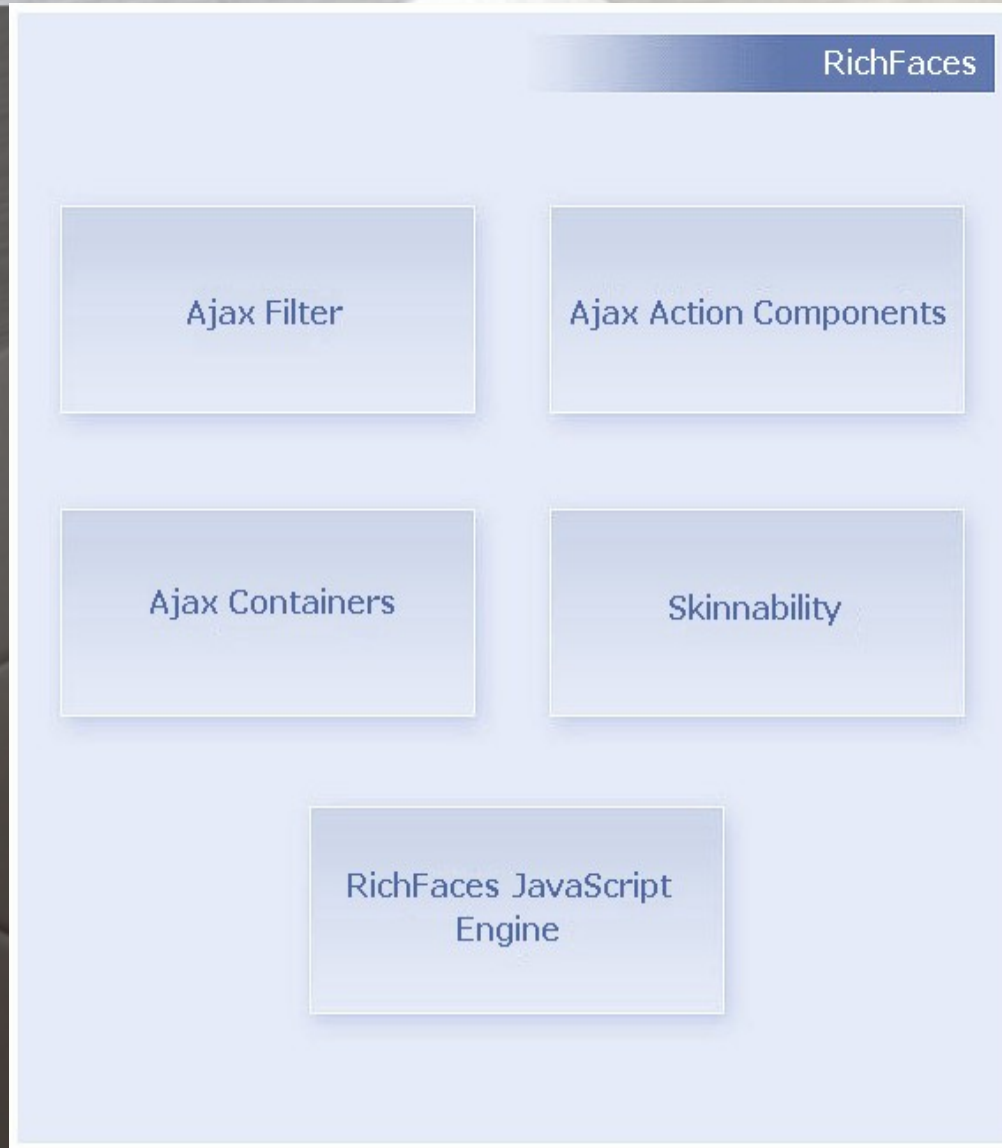


- ✗ Desarrollada por JBoss. Librería de componentes ricos para JSF.
- ✗ Además posee un framework avanzado para la integración sencilla de las funcionalidades AJAX dentro del desarrollo de aplicaciones de negocio.

- ✗ Los componentes están listos para usar por parte de desarrolladores.
- ✗ Soporte a skins (estilos)
- ✗ Beneficios de JSF
  - ✓ Ciclo de vida
  - ✓ Validaciones
  - ✓ Conversiones



# RichFaces: Componentes principales



## **Versiones de Java soportadas:**

- JDK 1.5 y superiores.

## **Implementaciones de JavaServer Faces soportadas:**

- Sun JSF 1.1 RI
- MyFaces 1.1.1 - 1.2
- Facelets JSF 1.1.1 - 1.2
- Seam 1.2. - 2.0

## **Navegadores admitidos**

- Internet Explorer 6.0 - 7.0
- Firefox 1.5 - 2.0
- Opera 8.5 - 9.0
- Netscape 7.0
- Safari 2.0



### **Sun JSF RI**

RichFaces trabaja con cualquier implementación de JSF (1.1 y 1.2) y con la mayoría de las bibliotecas de componentes JSF sin ninguna configuración adicional.

### Apache MyFaces

RichFaces trabaja con todos Apache MyFaces versiones (1.1.1 - 1.1.6), incluidas las bibliotecas específicas como Tomahawk, Sandbox y Trinidad (el anterior ADF Faces).

### Consideraciones

#### ➤ Problemas con filtros en web.xml

Para evitar estos problemas, el filtro de RichFaces debe ser el primero entre otros filtros dentro del archivo de configuración web.xml. Aún más, existen algunos problemas si se opta por utilizar MyFaces + Seam. Si se usa esta combinación, se debe usar el tag `a4j:page` dentro del `f:view`.

### **Facelets**

RichFaces ofrece un alto nivel de soporte para Facelets. Cuando se trabaja con RichFaces, no hay diferencia entre las diferentes releases de Facelets que se utilizan.

### **JBoss Seam**

Una de las novedades de RichFaces 3.1 es la integración con JBoss Seam. Esto mejora aún más la experiencia del usuario mediante la simplificación de la configuración y el "plumbing code", así como la prestación de estado y la concurrencia para la gestión de Ajax.



## 1. Definición en web.xml

```
<context-param>
  <param-name>org.richfaces.SKIN</param-name>
  <param-value>blueSky</param-value>
</context-param>

<filter>
  <display-name>RichFacesFilter</display-name>
  <filter-name>richfaces</filter-name>
  <filter-class>org.ajax4jsf.Filter</filter-class>
</filter>

<filter-mapping>
  <filter-name>richfaces</filter-name>
  <servlet-name>FacesServlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
```

### En las páginas JSP

```
<%@taglib uri="http://richfaces.org/a4j" prefix="a4j"%>  
<%@taglib uri="http://richfaces.org/rich" prefix="rich"%>
```

### En páginas xHTML

```
<xmlns:a4j="http://richfaces.org/a4j">  
<xmlns:rich="http://richfaces.org/rich">
```

## Como trabajar con RichFaces (3)

### Ya podemos utilizar las etiquetas

```
<a4j:commandButton value="update" reRender="infoBlock"/>  
<h:panelGrid id="infoBlock">  
  
....  
</h:panelGrid>
```



## Ajax4Jsf y RichFaces (1)

- Son una biblioteca open source que se integran totalmente en la arquitectura de JSF.
- Hereda las funcionalidades de sus etiquetas dotándolas con tecnología Ajax de forma limpia.
- **Sin añadir código Javascript.**

- Mediante este framework podemos variar el ciclo de vida de una petición JSF.
- Recargar determinados componentes de la página sin necesidad de recargarla por completo.
- Realizar peticiones al servidor automáticas, control de cualquier evento de usuario, etc.

## Etiquetas Ajax4Jsf (1)

**<aj4:support>** : Etiqueta que se puede añadir a cualquier otra etiqueta JSF para dotarla de funcionalidad Ajax. Permite al componente generar peticiones asíncronas mediante eventos (onclick, onblur, onchange,...) y actualizar campos de un formulario de forma independiente, sin recargar toda la página.

**<aj4:poll>** : Realiza cada cierto tiempo una petición al servidor.



## Etiquetas Ajax4Jsf (2)

**<aj4:commandButton>** : Botón de envío de formulario similar al de JSF. La principal diferencia es que se puede indicar que únicamente actualice ciertos componentes evitando la recarga de todo el formulario.

**<aj4:commandLink>** : Comportamiento similar a **<aj4:commandButton>** pero en un link.

**<aj4:htmlCommandLink>** : Muy parecida a la etiqueta anterior con pequeñas diferencias en la generación de links y cuando se utilizan etiquetas **< f:param >**.

**<aj4:region>** : Determina un área a decodificar en el servidor después de la petición Ajax.

**<aj4:status>** : Muestra el estado de la petición Ajax. Hay 2 estados posibles: procesando petición y petición terminada.

Por ejemplo mientras dure el proceso de la llamada al servidor y la evaluación de la petición se puede mostrar el texto "procesando..." y cuando termine la petición y se devuelva la respuesta a la página se cambia el texto por "petición finalizada".

## Etiquetas Ajax4Jsf (4)

**<aj4:form>** : Similar al `<h:form>` con la diferencia de que se puede enviar previamente el contenido al contenedor Ajax.

**<aj4:actionparam>** : Etiqueta que combina la funcionalidad de la etiqueta `<f:param>` y `<f:actionListener>`.

**<aj4:outputPanel>** : Se utiliza para agrupar componentes para aplicarles similares propiedades, por ejemplo a la hora de actualizar sus valores tras la petición Ajax.

**<aj4:ajaxListener>** : Similar a la propiedad `actionListener` o `valueChangeListener` pero con la diferencia de que la petición se hace al contenedor Ajax.



## Etiquetas Ajax4Jsf (5)

**<aj4:jsFunction>** : Se utiliza para pasarle un valor automáticamente a una función Javascript tras recibirlo del servidor.

**<aj4:loadScript>** : Inserta en la página las funciones Javascript contenidas en un archivo \*.js

**<aj4:loadStyle>** : Igual que la etiqueta anterior pero para una hoja de estilos \*.css

**<aj4:loadBundle>** : Similar al < f:loadBundle > de JSF.

**<aj4:log>** : Carga en la página una consola que muestra las trazas de los logs que devuelve el contenedor Ajax.

## Etiquetas Ajax4Jsf (6)

**<aj4:include>** : Se utiliza para incluir en la página el contenido de otra de acuerdo a la definición que se haga en las reglas de navegación del faces-config. Es decir la siguiente página a cargar de acuerdo a la navegación especificada se cargaría en la vista actual.

**<aj4:repeat>** : Etiqueta para iterar sobre una colección y mostrar todos sus campos.

**<aj4:keepAlive>** : Permite mantener un bean en un estado determinado durante peticiones.

**<aj4:mediaOutput>** : Componente que permite mostrar contenido multimedia como imágenes, vídeos, archivos sonoros, etc.

Más información: <http://www.jboss.org/jbossajax4jsf/docs/devguide/en/html/index.html>

# Etiquetas RichFaces (1)

**<rich:calendar>**: Este componente se utiliza para crear elementos de calendario.

<http://livedemo.exadel.com/richfaces-demo/richfaces/calendar.jsf?c=calendar>

**<rich:comboBox>** : Este es un componente, que proporciona combo Box editable.

<http://livedemo.exadel.com/richfaces-demo/richfaces/comboBox.jsf?c=comboBox>

**<rich:componentControl>** : Permite invocar a funciones API de JavaScript en los componentes definidos después de las acciones realizadas.

<http://livedemo.exadel.com/richfaces-demo/richfaces/componentControl.jsf?c=componentControl>



**<rich:datascroller>** : El componente diseñado para proporcionar la funcionalidad de los cuadros de desplazamiento utilizando peticiones Ajax.

<http://livedemo.exadel.com/richfaces-demo/richfaces/dataTableScroller.jsf?c=dataTableScroller>

**<rich:columns>** : Es un componente, que le permite crear una columnas dinámica.

<http://livedemo.exadel.com/richfaces-demo/richfaces/columns.jsf?c=columns>

## Eiemplo

```
<%@ taglib uri="http://richfaces.org/a4j" prefix="a4j"%>
<%@ taglib uri="http://richfaces.org/rich" prefix="rich"%>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<html>
  <head><title>repeater </title> </head>
  <body>
    <f:view>
      <h:form>
        <rich:panel header="Simple Echo">
          <h:inputText size="50" value="#{bean.text}" >
            <a4j:support event="onkeyup" reRender="rep"/>
          </h:inputText>
          <h:outputText value="#{bean.text}" id="rep"/>
        </rich:panel>
      </h:form>
    </f:view>
  </body>
</html>
```

- ◆ Al pertenecer RichFaces a un subproyecto de JBoss, su integracion con Seam es perfecta.
- ◆ Al ser RichFaces propiedad de Exadel, se ajusta perfectamente al IDE Red Hat Developer Studio (permite desarrollar aplicaciones visuales con RichFaces de forma fácil).



## Inconvenientes

- ◆ No se puede realizar una aplicación combinando RichFaces, IceFaces y Seam.
- ◆ Si no vas a utilizar RichFaces con Seam, y no es demasiado importante el tema de AJAX, IceFaces es interesante, porque trae un set de componentes mucho mas rico que el de RichFaces.
- ◆ IceFaces tiene actualmente una buena integración con seam, pero no es tan flexible como Ajax4jsf+RichFaces a la hora de trabajar con AJAX.



# Referencias RichFaces

- **Página oficial de Richfaces**

<http://labs.jboss.com/jbossrichfaces/>

- **Página con demos**

<http://livedemo.exadel.com/richfaces-demo/index.jsp>

- **Documentación y guía para el desarrollador**

<http://labs.jboss.com/jbossrichfaces/docs/index.html>

05

IceFaces

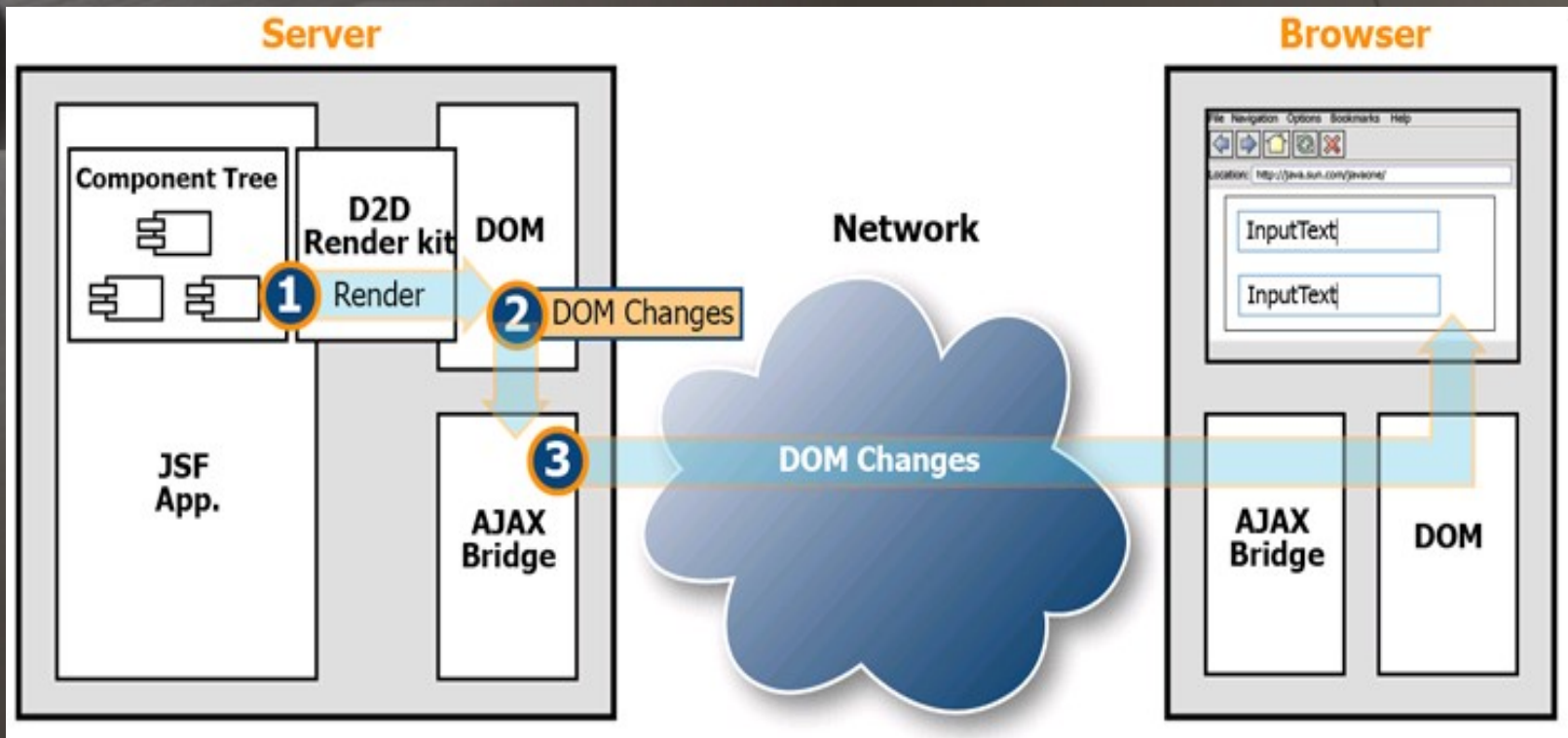




- Desarrollada por IceSoft. Librería de componentes ricos para JSF con un framework avanzado para la integración sencilla de las funcionalidades AJAX dentro del desarrollo de aplicaciones de negocio.
- Características similares a RichFaces.

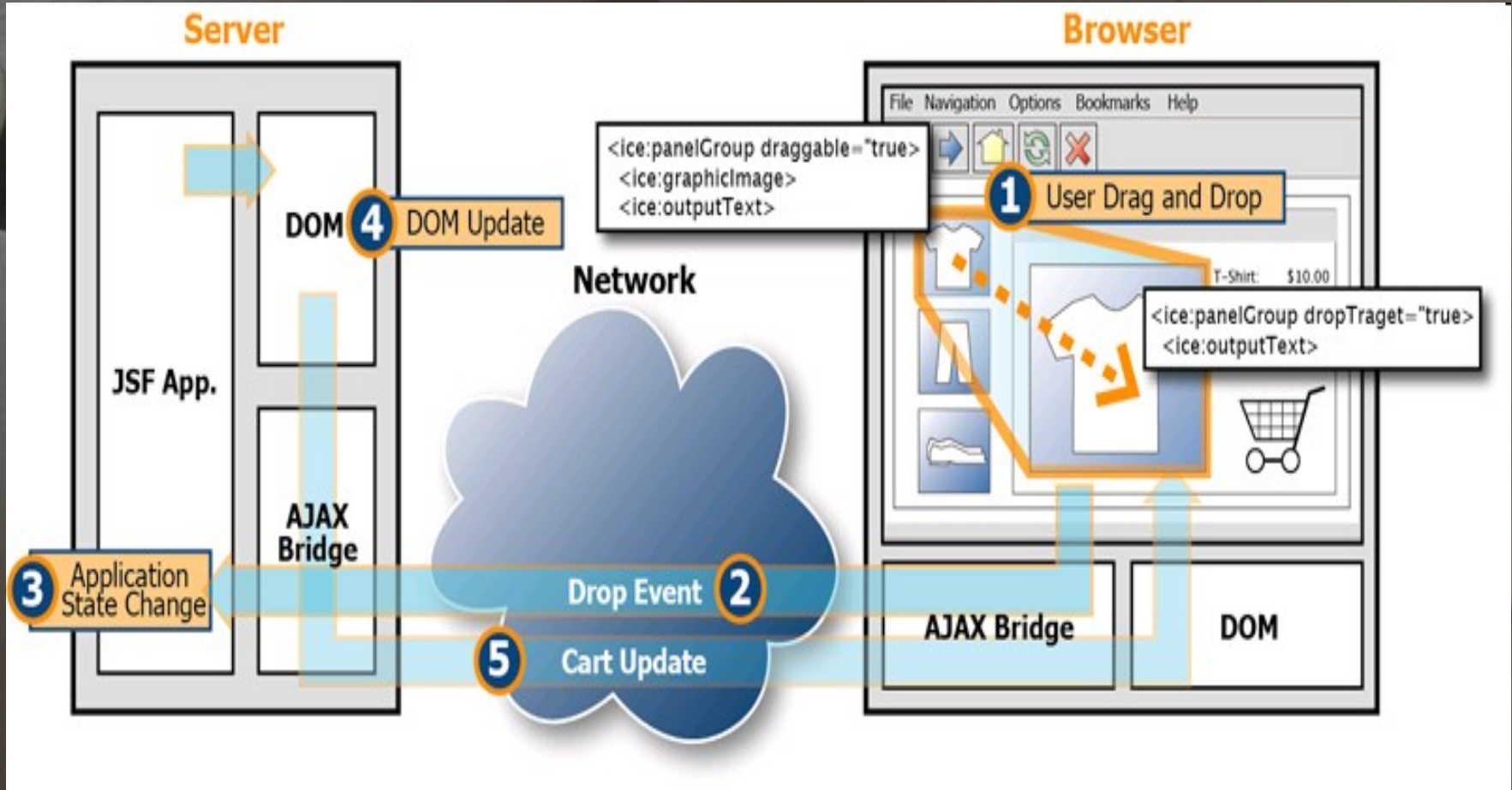
# Objetivos (1)

## Facilitar el uso de AJAX con JSF



# Objetivos (2)

## Desarrollo de aplicaciones ricas





## ■ Tutoriales sobre componentes

<http://facestutorials.icefaces.org/tutorial/index.html>

## ■ Demos de componentes

<http://component-showcase.icefaces.org/component-showcase/showcase.iface>

# FORO ABIERTO

Gracias por su atención

Israel Alcázar ([ialcazar@atsistemas.com](mailto:ialcazar@atsistemas.com))

PARA MÁS INFORMACIÓN:

[www.atsistemas.com](http://www.atsistemas.com) / e-mail: [at@atsistemas.com](mailto:at@atsistemas.com)