Universidad de los Andes

Model View Controller, Java Server Faces

Por: Luis Daniel Benavides Navarro Ph.D.

Departamento de Ingeniería de Sistemas y Computación

Especialización en Construcción de Software

Bogotá, COLOMBIA

Agenda

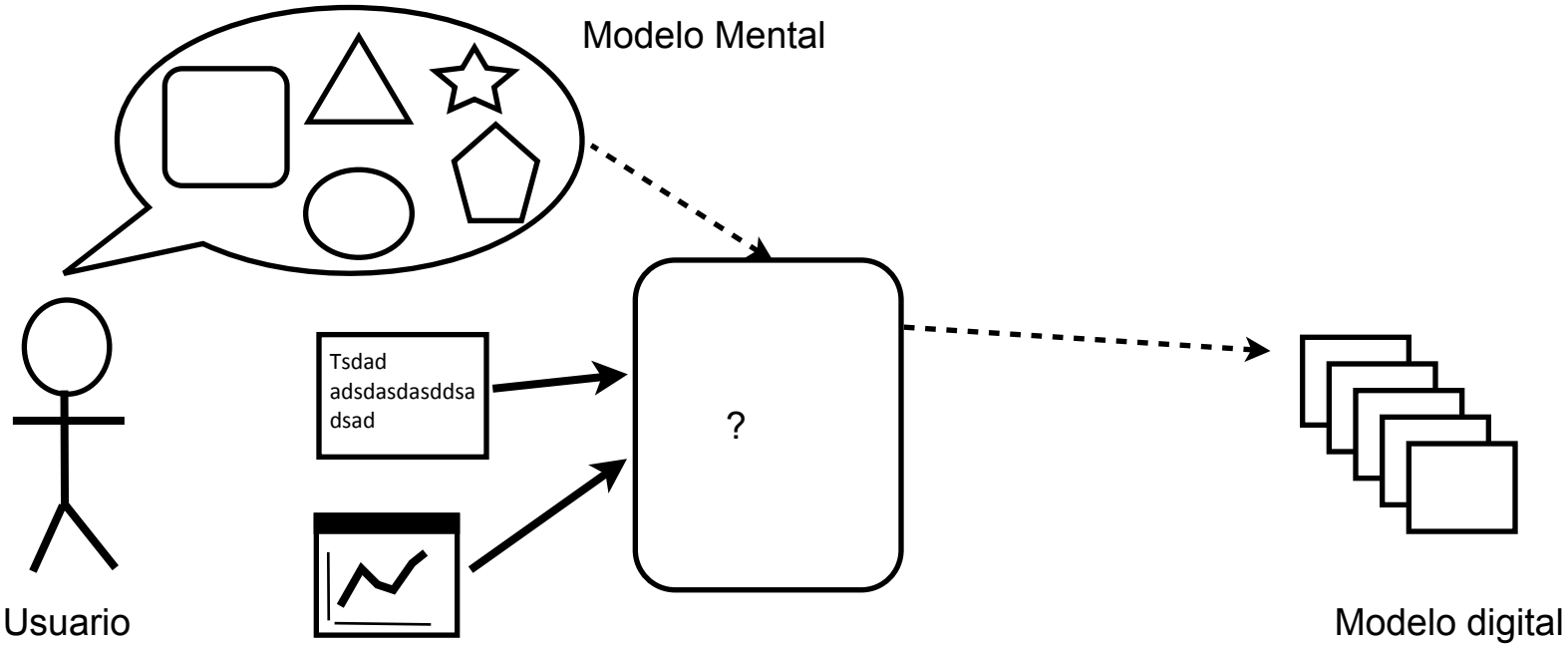
- Patrón Model View Controller
- Plataforma Java Server Faces

Model View Controller

Historia de MVC

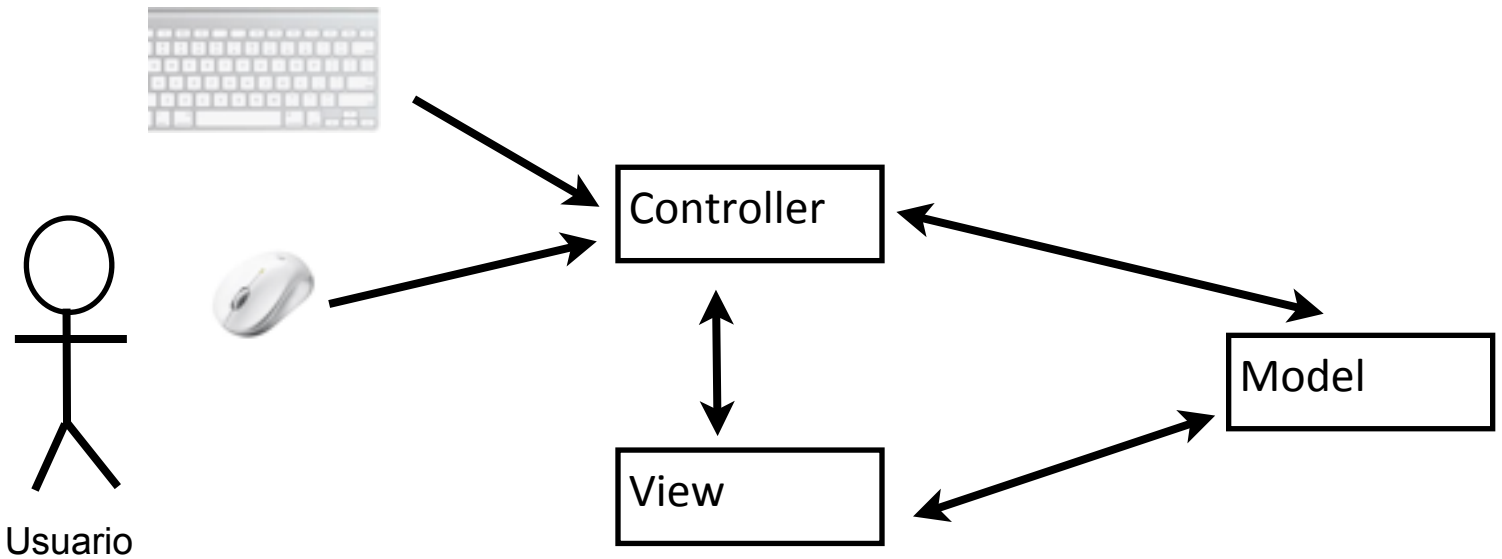
- Surge en el contexto de desarrollo de interfaces gráficas en Xerox Parc, en el grupo de Samalltalk
- Nota original de Trygve Reenskaug (Noruego) en 1979 (THING-MODEL-VIEW-EDITOR)
- Buen resumen de la implementación en el artículo de 1987: "Applications Programming in Smalltalk-80: How to use Model-View-Controller (MVC)" por S. Burbeck.

Conceptos básicos



- Separa la lógica de la aplicación de la interfaz del usuario
- Puente entre modelo mental y modelo digital

Solución



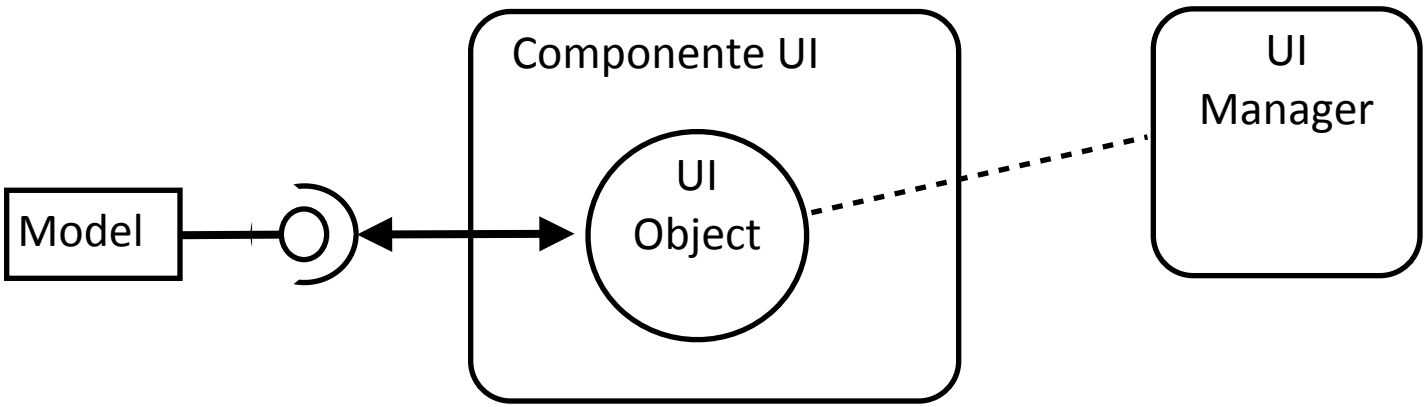
Responsabilidades

- **Modelo:** Representa el mundo del problema
- **Vista:** Representación visual de un modelo o parte de él
- **Controlador:**
  - Enlace entre el usuario y el sistema
  - Organiza las vistas y coordina como se presentan
  - Provee medios para que el usuario envíe comandos al sistema

Notas sobre la implementación

- Se convirtió en un estilo arquitectural
- Tiene muchas adaptaciones y versiones
- Es muy abusado en la literatura sin hacer énfasis en los detalles importantes
- Muchas implementaciones mezclan la vista y el controlador

Ejemplo: Swing



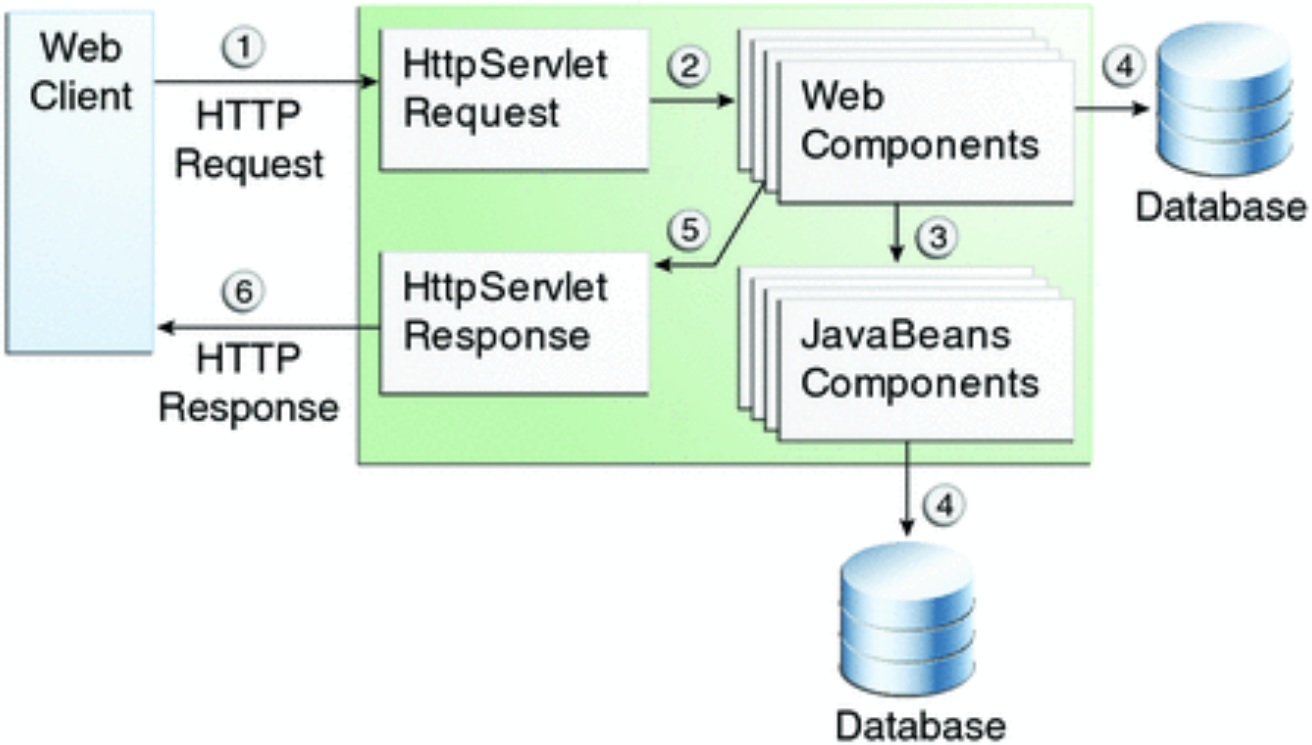
- Componentes:, e.g., JButton
- View/controller en los UI Objects, permiten interfaces portables
- Interfaces para los modelos, permite cambiar modelos
- Componentes son creados con un modelo por defecto
- Generalmente se comunica con modelos del usuario por medio de eventos
- Ver: http://java.sun.com/products/jfc/tsc/articles/architecture/

Java Server Faces

Tipos de aplicaciones WEB

- Orientadas a presentación, e.g. aplicaciones JSF
- Orientadas a servicios, e.g. web services

Manejo de Requests



Imágen extraida del Tutorial de Java EE 6

Motivación para JSF

- Crear un modelo de componentes gráficos para la web (a la Swing)
- Pensar en términos de componentes, eventos, beans administrados y sus interacciones
- Ocultar las abstracciones de request/response, y lenguaje de marcas

Aplicaciones Web usando JSF

Java Server Faces

- Un modelo de componentes de servidor
- Utilizado para construir aplicaciones web
- Esta compuesta por:
  - Un API de componentes para manejar estado, manejar eventos, validación del lado del servidor, definir navegación, internacionalización
  - Librerías de tags para adicionar componentes a las páginas y para conectar los componentes a objetos de servidor

Modelo de programación

- Crear página web
- Agregar componentes a la página usando tags
- Enlazar componentes a objetos del servidor
- Cablear eventos de componentes a código del lado del servidor
- Guardar y restaurar el estado de la aplicación
- Reusar y extender componentes por medio adaptación

Un primer ejemplo

- Hello World
- Con AJAX

Desarrollar un bean de apoyo

```
package hello;
import javax.faces.bean.ManagedBean;

@ManagedBean
public class Hello {

    final String world = "Hello World!";

    public String getworld() {
        return world;
    }
}
```

Creando una página

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Facelets Hello World</title>
    </h:head>
    <h:body>
        #{hello.world}
    </h:body>
</html>
```

Mapeando la instancia del FacesServlet

```
    <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>
```

- Se hace en el descriptor Web (web.xml)

Ajax

- En la página

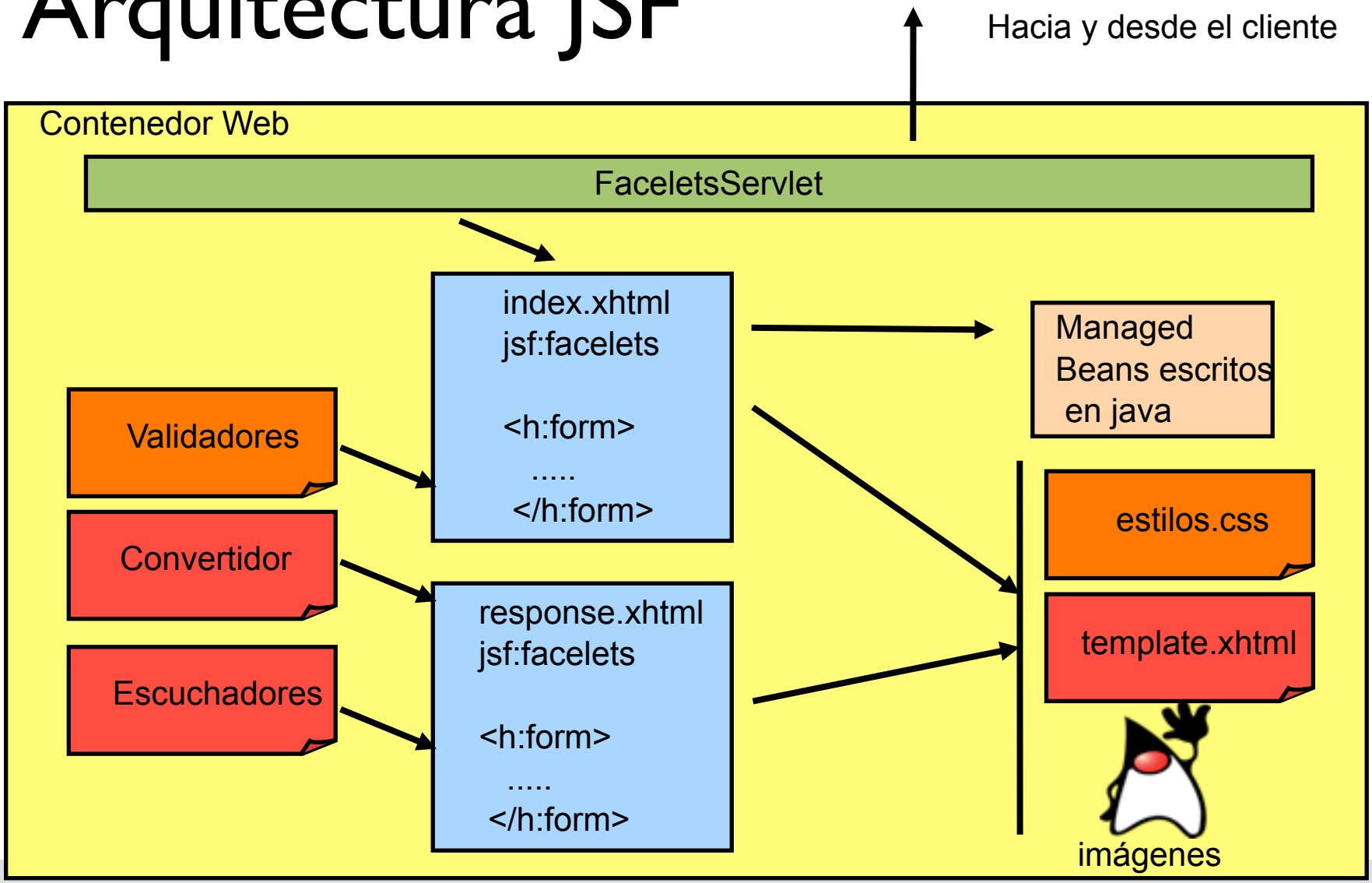
```
<h:form>
<h:outputLabel id="miMensaje" value="#{hello.message}">
            </h:outputLabel>
<h:commandButton value="Actualizar usando ajax">
                <f:ajax execute="" render="miMensaje"></f:ajax>
</h:commandButton>
</h:form>
```

- En el bean se agrega el campo *message* y los métodos get y set

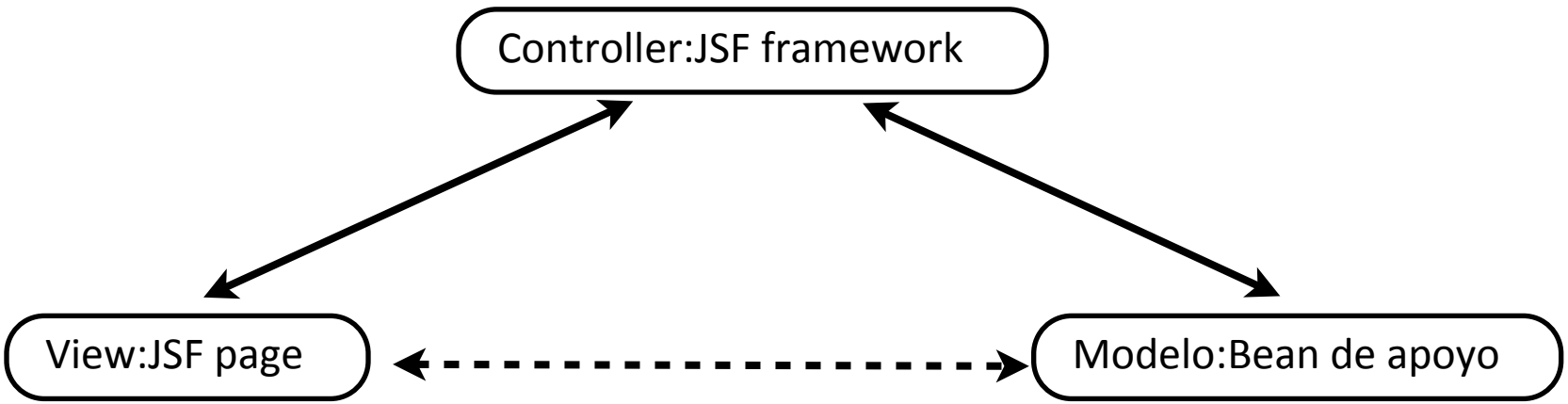
Demo

Arquitectura JSF

Arquitectura JSF



MVC a la JSF


Controller:JSF framework
View:JSF page
Modelo:Bean de apoyo

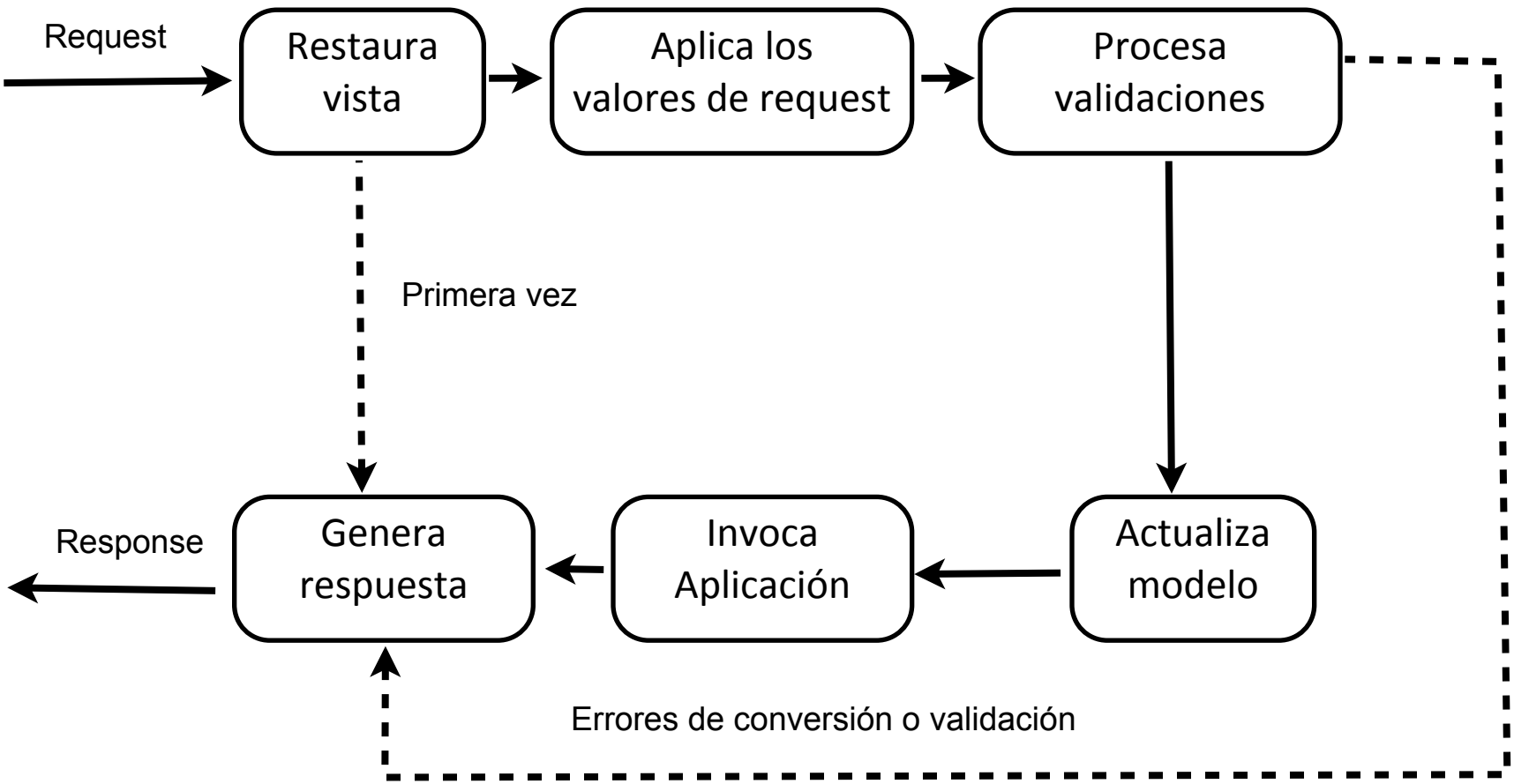
Servicios de JSF

- Arquitectura MVC
- Conversión de datos
- Validación y manejo de errores
- Internacionalización
- Componentes personalizados
- Soporte AJAX
- Soporte otras tecnologías de visualización

Lo que pasa detrás de bambalinas

- El cliente se conecta por primera vez con http://localhost:8080/jsfajax/faces/index.xhtml
- Se crea el árbol de componentes
- La página html generada
- le cliente envía una segunda solicitud, e.g, envía un formulario (POST)
- Se extraen parámetros
- Cada componente lee los param. que necesita (valores locales)
- Proceso de validación (si falla, reporta errores al usuario)
- Actualiza los beans necesarios
- Invoca métodos
- Se genera la página

El ciclo de vida



Beans Administrados

Definición de beans

- POJO
  - Campos
  - Métodos *getXX* y *setXX*
  - otros métodos
- Anotación ManagedBean
- Anotación SCOPE
- Los campos con solo método *get* son de solo lectura
- Administrados por el contenedor web

Alcances de Beans

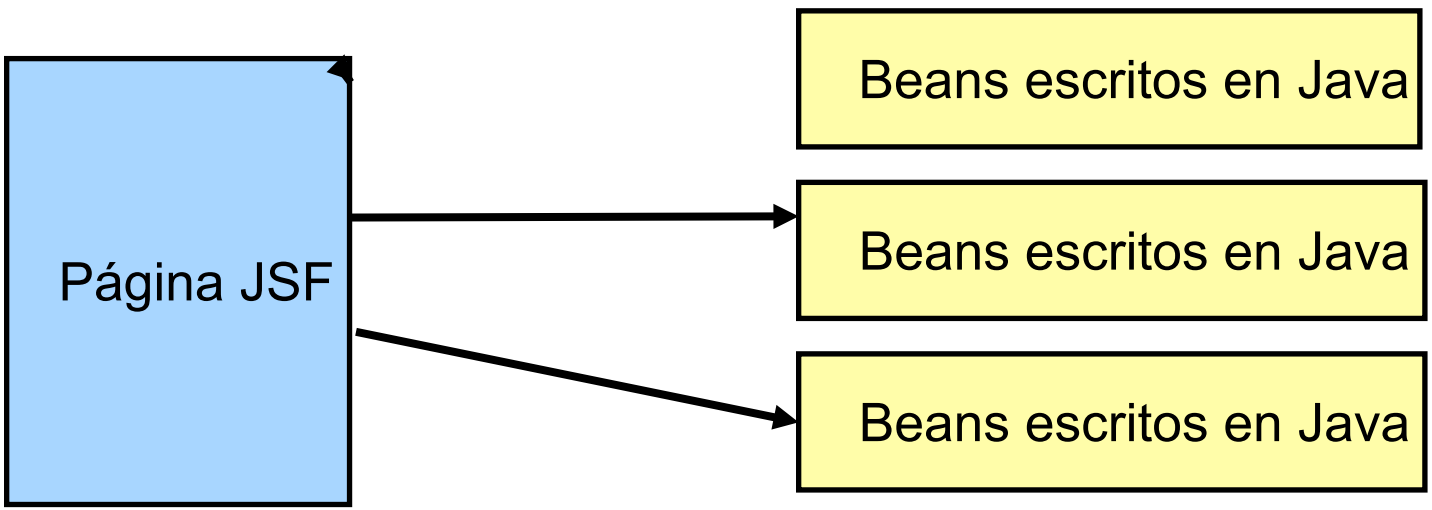
- Managed Beans
  - Sesión
  - Solicitud (Request)
  - Aplicación
  - View

- Revisar otro tipo de beans soportado: CDI
  - Conversation scope (investigar)

Unified Expression Language

¿Que es el Unified Expression Language?

- El Unified Expression Language es un mecanismo que permite que la capa de presentación (páginas web) pueda comunicarse con la lógica de la aplicación, es decir los beans de respaldo



Para que sirve el EL?

- La tecnología JavaServer Faces usa EL para lo siguiente:
  - Evaluación inmediata o diferida de expresiones
  - Para actualizar o leer datos de los beans
  - Para invocar expresiones de valor y métodos de los beans

Evaluación inmediata de expresiones

- Evalúa la expresión inmediatamente cuando la página es preparada para ser mostrada
- Las expresiones tienen la forma ${expr}
- Son de solo lectura
- Pueden estar en la parte estática de la página y como valores de atributos en un tag de JSF.
- Ejemplo:

```
<fmt:formatNumber value="${sessionScope.cart.total}"/>
```

Evaluación diferida de expresiones

- Evalúa la expresión cuando sea necesario dentro del ciclo de vida de la página web.
- Las expresiones tienen la forma #{expr}
- Pueden leer y actualizar campos de los beans (lectura/escritura)
- Solo pueden estar como valores de atributos de un tag de JSF.
- Ejemplo:

```
<h:inputText id="name" value="#{customer.name}" />
```

Expresiones de valor

- Pueden ser de lectura (${customer.name}) o de lectura/escritura (#{customer.name})
- Pueden usarse para referenciar objetos:

${customer}

  - El objeto se busca según el comportamiento de PageContext.findAttribute("customer")
  - El contenedor busca *customer* en los contextos de página, request,session y application

Expresiones de valor II

- Pueden usarse para referir propiedades de objetos
  - ${customer.name}
  - ${customer["name"]}

Expresiones de métodos

- Las expresiones pueden invocar métodos sin parámetros:

```
<h:form>
    <h:inputText
        id="name"
        value="#{customer.name}"
        validator="#{customer.validateName}"/>
    <h:commandButton
        id="submit"
        action="#{customer.submit}" />
</h:form>
```

Expresiones de métodos II

- Las expresiones pueden invocar métodos con parámetros.
- Ejemplos:
  - <h:inputText value="#{userNumberBean.userNumber('5')}">
  - <h:commandButton action="#{trader.buy('JAVA')}" value="buy"/>

Operadores

- +, - (binary), *, / and div, % and mod, - (unary)
- and, &&, or, ||, not, !
- ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le.
- empty , usada para ver si un valor es vacio o nulo
- Conditional: A ? B : C. Evaluate B or C, depending on the result of the evaluation of A.

Palabras reservadas

- and
- or
- not
- eq
- ne
- lt
- gt
- le
- ge
- true
- false
- null
- instanceof
- empty
- div
- mod

Ejemplos de expresiones

Expresión EL	Resultado
${1 > (4/2)}	false
${4.0 >= 3}	true
${100.0 == 100}	true
${(10*10) ne 100}	false
${'a' < 'b'}	true
${'hip' gt 'hit'}	false
${4 > 3}	true
${1.2E4 + 1.4}	12001.4
${3 div 4}	0.75
${10 mod 4}	2

Ejemplos de expresiones II

Expresión EL	Resultado
${!empty param.Add}	False si el paŕametros *Add* del request es nulo o vacio
${pageContext.request.contextPath}	El camino del contexto
${sessionScope.cart.numberOfItems}	El valor de la propiedad *numberOfItems* del atributo de session *cart*.
${param['mycom.productId']}	El valor del parámetro de request mycom.productId.
${header["host"]}	El host (servidor).
${departments[deptName]}	El valor de depName en el mapa departments
${requestScope['javax.servlet.forward.servlet_path']}	El valor del atributo de request javax.servlet.forward.servlet_path
#{customer.lName}	Obtiene el valor de la propiedad lName del bean customer durante el request inicial. Cambia el valor de lName cuando se necia el formulario de regreso.
#{customer.calcTotal}	El valor de retorno del método calcTotal del bean customer.

Tarea opcional

Tarea Opcional

Tutorial Facelets

http://netbeans.org/kb/docs/web/jsf20-intro.html

Ejercicios adicionales

- Escriba una página con facelets para probar todas las expresiones de ejemplo
- Escriba un proyecto utilizando templates que pida el id y el password.
  - El password no debe verse cuando se digita
  - Debe crear un bean de apoyo que pueda validar el usuario y el password, puede usar un solo usuario y password.
  - Agregue un menú estático en el template.

JSF:Librerías

Como utilizar la tecnología Java Server Faces en las páginas web

Lectura sugerida

- Capítulo 7 del tutorial de Java 6:
  - Using JavaServer Faces Technology in Web Pages

Resumen

- En esta sección veremos en detalle
  - Como configurar una página para soportar JSF
  - Como adicionar componentes usando tags de la librería básica
  - Como adicionar componentes usando los core-tags

Configurando una página para usar JSF

- Para usar los tags de la librería principal o de la librería de HTML debe declarar:
  - Las URI de las librerías
  - Los prefijos de las librerías
- Cuando se utilicen tags de una librería se debe anteponer el prefijo seguido de dos puntos
  - <h:form ....
  - Ejemplo:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
```

Usando tags de la librería de HTML en páginas

tag	Functions	Rendered As	Appearance
commandButton	Submits a form to the application	An HTML <input type=type> element, where the type value can be submit, reset, or image	A button
dataTable	Represents a data wrapper	An HTML <table> element	A table that can be updated dynamically
form	Represents an input form (inner tags of the form receive the data that will be submitted with the form)	An HTML <form> element	No appearance
graphicImage	Displays an image	An HTML <img> element	An image
inputHidden	Allows a page author to include a hidden variable in a page	An HTML <input type=hidden> element	No appearance
inputSecret	Allows a user to input a string without the actual string appearing in the field	An HTML <input type=password> element	A text field, which displays a row of characters instead of the actual string entered
inputText	Allows a user to input a string	An HTML <input type=text> element	A text field
inputTextarea	Allows a user to enter a multiline string	An HTML <textarea> element	A multi-row text field

Atributos comunes de los tags html

- Id: nombre identificador del componente
- style: especifica una hoja de estilos (CSS)
- styleClass: especifica una clase en la CSS
- value: Identifica una fuente externa que provee un valor
- binding: Identifica una propiedad de un bean y enlaza el componente a una instancia de el (investigar)

Demo/Ejercicio: página de validación de password

- Cree un proyecto JSF en netbeans
- Cree un a página para capturar el usuario y el password y otra para la respuesta
- Cree un bean para validar el usuario y el password. Configurelo para que sea administrado con alcance sesión
- Modifique sus páginas con tags de html y EL para lograr la funcionalidad

Usando los tags de la librería principal

- Ejecutan acciones que no ejecutan los tags de la librería jsf de html

Tag Categories	Tags	Functions
Event-handling tags	f:actionListener	Adds an action listener to a parent component
	f:phaseListener	Adds a PhaseListener to a page
Data conversion tags	f:converter	Adds an arbitrary converter to the parent component
	f:convertDateTime	Adds a DateTime converter instance to the parent component
Validator tags	f:validateDoubleRange	Adds a DoubleRangeValidator to a component
	f:validateLength	Adds a LengthValidator to a component
	f:validateLongRange	Adds a LongRangeValidator to a component

Ejercicios sugeridos

- Realice un programa en JSF para capturar los datos de un cliente potencial, incluyendo nombre de usuario y clave.
- Realice un programa calculadora que sume, divida, y multiplique. Debe capturar los datos y la operación y regresar el resultado.

Navegación

Navegación estática

- Lógica de navegación quemada en páginas JSF

```
<h:comandButton label="Login" action="welcome"/>
```

Navegación dinámica

- Lógica almacenada en métodos de acción
- El valor de retorno es convertido a String

```
<h:commandButton label="Login"
action="#{loginBean.verifyUser}"/>
```

Desacoplando navegación de la implementación

- Utilizar un esquema de nombres generados por los métodos de acción
- Mapear estos nombres a páginas concretas en el archivo *faces-config.xml*

```
<navigation-rule>
  <from-view-id>/index.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>sucess</from-outcome>
    <to-view-id>/welcome.xhtml </to-view-id>
  </navigation-case>
<navigation-rule>
```

Como utilizar convertidores y validadores

Lectura

- Capítulo 8 del tutorial de Java 6:
  - Using Converters, Listeners and Validators

Convertidores estándar

- La tecnología JSF provee convertidores estándar que permiten convertir los valores de las páginas a los valores deseados
- En los convertidores estándar se encuentran:
  - BigDecimalConverter, BigIntegerConverter, BooleanConverter, ByteConverter, CharacterConverter, DateTimeConverter, DoubleConverter, EnumConverter, FloatConverter, IntegerConverter, LongConverter, NumberConverter, ShortConverter

Para que sirven los convertidores

- Convierten una entrada hacia un tipo específico
- Automatizan el flujo de control de error en la entrada de un formulario

Como utilizar convertidores estándar

- Método 1: Inserte un tag de conversión standard (*convertDateTime* o *convertNumber)* en el tag de un componente:
- Conversión de fecha

```
<h:inputText id="birthDate" value="#{createUserAcoount.birthDate}" maxlength="8" size="8"
    required="true" label="Birth Date">
      <f:convertDateTime pattern  ="mm/dd/yy" />
</h:inputText>
```

- Conversión de número

```
<h:outputText value="#{cart.total}" >
 <f:convertNumber type="currency"/>
</h:outputText>
```

Como utilizar convertidores estándar

- Método 2: enlace el valor del componentes a la propiedad de un bean con un tipo específico:
- **En el bean:**

```
Integer age = 0;

public Integer getAge(){ return age;}

public void setAge(Integer age) {this.age = age;}
```

En la pagina:

```
<h:inputtext value="#{MyBean.age}" ...
```

Como utilizar convertidores estándar

- Método 3: si el valor no esta enlazado a una propiedad, puede usar directamente el atributo converter con una instancia directa del convertidor:

```
<h:inputText converter="javax.faces.convert.IntegerConverter" />
```

Como utilizar convertidores estándar

- Método 4: Insertar tags de conversión en los tags de los componentes. Esto puede ser con un tag *converter* con atributo *converterId* o con atributo *binding*:

```
<h:inputText value="#{LoginBean.Age}" />
   <f:converter converterId="Integer" />
</h:inputText>
```

Desplegando mensajes de error

- Utilizar el tag *h:message*

```
<h:inputText id="amount" label="#{msgs.amount}"
value="#{payment.amount}"/>
<h:message for="amount" showSummary="true" showDetail"true"/>
```

- Mensajes personalizados

```
<h:inputText id="amount" label="#{msgs.amount}"
value="#{payment.amount}" converterMessage="Número no
valido"/>
```

- Puede igualmente personalizar todos los mesnajes en un archivo separado.

Validadores estándar

- La tecnología JSF provee validadores estándar que permiten validar los valores de las páginas a los formatos deseados

Validator Class	Tag	Function
DoubleRangeValidator	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.
LengthValidator	validateLength	Checks whether the length of a component's local value is within a certain range. The value must be a java.lang.String.
LongRangeValidator	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.
RegexValidator	validateRegEx	Checks whether the local value of a component is a match against a regular expression from java.util.regex package.
BeanValidator	validateBean	Registers a bean validator for the component
RequiredValidator	validateRequired	Ensures that the local value is not empty on a EditableValueHolder component

Para que sirven los validadores

- Validan el formato de una entrada
- Automatizan el flujo de control de error en la entrada de un formulario

Como utilizar validadores estándar

- Método 1: Inserte un tag de uno de los validadores estándar en un tag de componente:
- Ejemplo numérico

```
<h:inputText id="quantity" size="4" value="#{item.quantity}" >
   <f:validateLongRange minimum="1"/>
</h:inputText>
```

- Ejemplo Expresión regular

```
<h:inputText id="email" value="#{createUserAcoount.email}" maxlength="15" size="16"
required="true" label="email">
    <f:validateRegex pattern="[a-zA-z]+@[a-zA-z]+.com"></f:validateRegex>
</h:inputText>
```

Como utilizar validadores estándar

- Método 2: enlace el validador hacia un método de bean específico:

- <h:inputtext value=”#{MyBean.age}” validator="#{checkoutFormBean.validateEmail}" ...

Ejercicios

- Realice un página que capture los siguientes datos del usuario:
  - Nombre, apellidos, cédula, dirección, teléfono, teléfono celular, email, fecha de nacimiento, password
  - El nombre y apellido no deben tener números
  - El password debe ser de 6 a 8 caracteres
  - El email debe estar bien formado
  - La fecha de nacimiento debe estar bien formada
- Almacene los valores y luego cree un página para el login

Eventos en el ciclo de vida de JSF

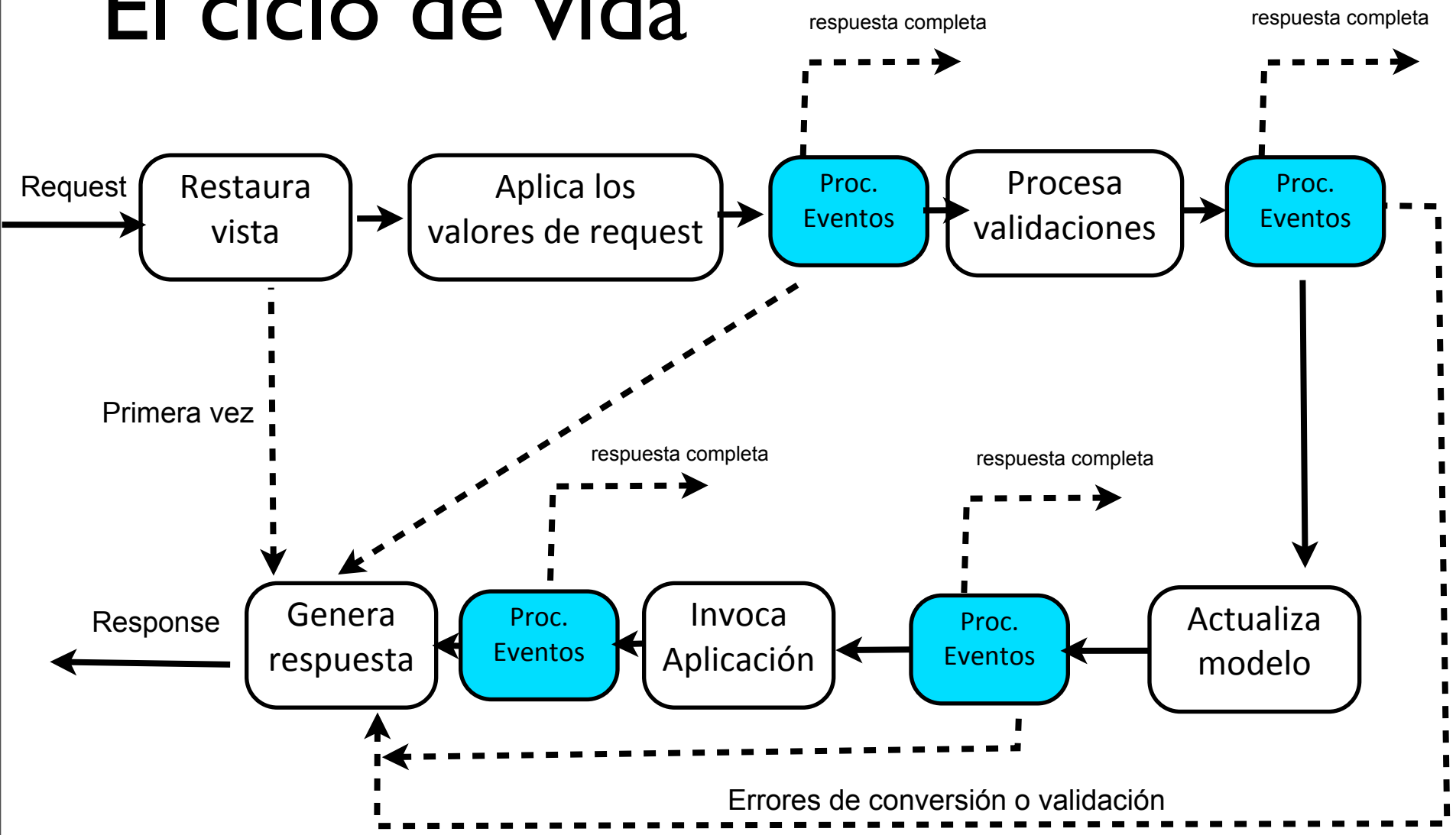
Eventos en el ciclo de vida de JSF

- Se registran manejadores de eventos con componentes

```
<h:inputText  id="name" size="50" value="#{cashier.name}"
required="true">
    <f:valueChangeListener type="listeners.NameChanged" />
</h:inputText>
```

- Los escuchadores pueden afectar el ciclo JSF
  - Dejandolo proceder normalmente
  - Saltar el ciclo hasta generar respuesta
  - Saltar todo el ciclo restante

El ciclo de vida



Eventos de cambio de valor

- Agregando un listener a un tag

```
<h:selectOneMenu value="#{form.country}" onchange="submit()"
      valueChangeListener="#{form.countryChanged}">
   <f:selectItems value="#{form.countries}" var="loc"
      itemLabel="#{loc.displayCountry}" itemValue="#{loc.country}"/>
</h:selectOneMenu>
```

- Método para atender evento

```
public void countryChanged(ValueChangeEvent event) {
   for (Locale loc : countries)
      if (loc.getCountry().equals(event.getNewValue()))
         FacesContext.getCurrentInstance().getViewRoot().setLocale(loc);
}
```

Eventos de acción

- Son disparados por botones y links
- Generalmente brinda información de navegación

```
<h:commandButton image="/resources/images/mountrushmore.jpg"
   actionListener="#{rushmore.handleMouseClick}"
   action="#{rushmore.act}"/>
```

- El action listener brinda información sobre la localización del mouse sobre la imagen
- El ejemplo responde de acuerdo a un mapa en la imagen

Bibliografía

- [Gea2010] David Geary, Cay Horstmann. Core JavaServer Faces. Third Edition.Prentice Hall. 2010.
- [Bie2009] Adam Bien. Real World Java EE Patterns – Rethinking Best Practices. Editorial: lulu.com, 2009.
- [JEE] The Java EE 6 Tutorial: Basic and Advanced Topics. Oracle – Sun Microsystems. Disponible en: http://download.oracle.com/javaee/6/tutorial/doc/
- ACM Digital Library (http://portal.acm.org/dl.cfm). Acceso a artículos científicos solo con subscripción.

Universidad de los Andes

Gracias !!!

Luis Daniel Benavides Navarro

dnielben@gmail.com