DESARROLLO DE APLICACIONES WEB

Tema 5.- Java Entreprise Edition

Java Server Pages. MVC: Vistas

Contenido

- Introducción a JSP
- Elementos JSP
- Objetos Implícitos en JSP y EL
- JSTL
- MVC: Vistas

Introducción

- ¿ Qué son los JSP ?

  - JSP es una tecnología de *scripting* del lado del servidor que permite incluir código Java embebido con HTML

    Realmente son Servlets generados por el contenedor a partir del código HTML

  - Tecnología Similar a PHP

  - Ventajas de JSP

    - Al ser un servlet, todas las ventajas de éstos (eficiencia, potencia, portabilidad, etc)

    - Pero, con respecto a los servlets, los JSP son más fáciles de usar para la creación de vistas

Introducción – Ejemplo JSP

JSP:

Código HTML con sentencias Java embebidas

```
<!--  holaPepe.jsp -->
<%
    String persona = new String("Pepe");
%>
<!DOCTYPE html>
<html>
<head>
    <title> Saludando a Pepe! </title>
    <meta charset="UTF-8" />
</head>
<body>
  <h1> Hola <%=persona%>! </h1>
</body>
</html>
```

Introducción

- Para el desarrollador, un JSP es un documento HTML con código Java embebido

- Formalmente, una página JSP es una implementación de la clase **javax.servlet.Servlet,** que describe como crear un objeto (response) respuesta **HttpServletResponse** a partir de un objeto (request) petición **HttpServletRequest.**

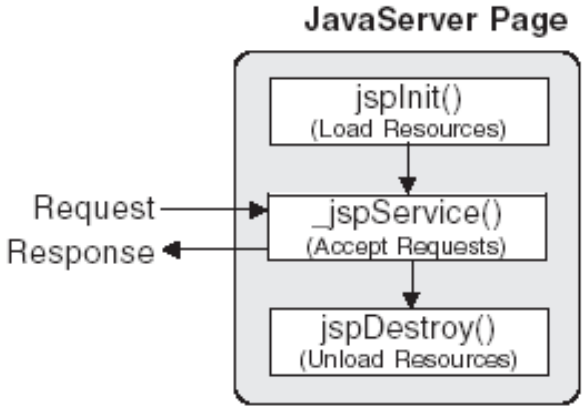
- Un JSP es transformado a un Servlet por el servidor de aplicaciones

- Ciclo de vida.

  Inicialización: jspInit()

  Servicio: jspService()

  Destrucción: jspDestroy()



Contenido

- Introducción a JSP
- **Elementos JSP**
- Objetos Implícitos en JSP y EL
- JSTL
- MVC

Elementos JSP

- Scripting. Inserción de código Java
  - Declaraciones
  - Expresiones
  - Scriptles
  - EL: Expression Language
- Directivas. Llamadas al Servidor de Aplicaciones
- Acciones. Inserción de código Java dinámico

JSP: Elementos de Scripting

- **Declaraciones**: Crea atributos y métodos del Servlet (externos a `jspservice()`).

  <%! `declaration; [ declaration; ]+ ...` **%>**

  Ejemplos:

```
<%!  int i=0; %>
<%!  Circle a = new Circle(2.0); %>

<%!
private BookDBAO bookDBAO;
public void jspInit() {
    bookDBAO = (BookDBAO) getServletContext().getAttribute("bookDB");
    if (bookDBAO == null)  System.out.println("Couldn't get database.");
}
%>
```

JSP: Elementos de Scripting

- **Expresiones**: Evalúa la expresión e incluye el resultado como parte de la respuesta.

<%= `expression` **%>**

Ejemplos

```
<%= 4+3+valor %>
<%= balance.haber() %>
<a href="fichero.<%= ext %>" >
```

JSP: Elementos de Scripting

- **Scriptlets**. Fragmento de código java que se incluirá en jspservice().

  <% codigo_java %>

  Ejemplo: (NOTA: los scriplets pueden aparecer con código HTML intercalado

```
<%  String name = null;
      if (request.getParameter("name") == null) {
%>
 <p>Error: Falta parámetro</p>

<%  } else {
      obj.procesa(request.getParameter("name"))

    }
%>
```

JSP: EL – Expression Language

- **Lenguaje de expresiones:** EL (Expression Language).

`${ expression }`

Ejemplo

${10*var+2}

${12 mod 2}

${a >= b}

${!empty param.nombClte}

${header[“user-Agent”]}

${sessionScope.numArticulos}

- **Comentarios.** Aclaraciones en el código

<!-- *comentarios* -->

<%-- *comentarios* --%>

JSP: Directivas (I)

- **Directivas:** Mensajes al contenedor de Servltes

```
<%@directiva …  %>
```

  - include: Insertar un fragmento JSP
  - page: propiedades de la página e importar clases
  - taglib: Incluir librerías de etiquetas
  - variable: Define variables
  - tag: Permite crear nuevas etiquetas de usuario
  - attribute: Permite añadir atributos para las etiquetas de usuario

JSP: Directivas (II)

- **include:** Inserta un fichero de texto o código Java

<%@include file="relativeURL" %>

Ejemplo:

```
<html>
<head><title>Ej Include</title></head>
<body>
Hoy es: <%@ include file="date.jspf" %>
</body>
</html>
```

```
<!-- Fichero date.jspf  -->
<%@ page import="java.util.*" %>
<%= (new java.util.Date() ).toLocaleString() %>
```

JSP: Directivas (III)

- **page:** Define los atributos de la página JSP.

<%@page *atrib*="*valor*" %>

Atributos

extends="package.class"

import="{package.class | package.*}, ..."

session="true|false"

buffer="none|8kb|sizekb"

isThreadSafe="true|false"

info="text"

errorPage="relativeURL"

contentType="mimeType [; charset=characterSet]" | "text/html ; charset=UTF-8"

isErrorPage="true|false"

pageEncoding="characterSet |UTF-8"

isELIgnored="true|false"

JSP: Directivas (y IV)

- **taglib**: Define una librería de etiquetas y un prefijo para usarlo en la página JSP

  <%@taglib uri="http:…" prefix="…" %>

  Ejemplo:

  ```
  <%@taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
  …
  <c:set var="saludo" value="Hola Mundo!" />
  ```

- **tag**: define las propiedades de una etiqueta de usuario

  <%@tag … %>

- **attribute**: Define atributos.

  <%@attribute name="attribute-name" … %>

- **variable**: Define variables estableciendo su ámbito para otros JSP

  <%@variable name-given="…" scope="…" … %>

JSP: Principales Acciones (actions)

- Acciones: Insertan código dinámico
  - <jsp:include> : Incluye un Servlet o JSP en otro.

    <jsp:include page="***rutaFichero***" />
  - <jsp:forward> : Redirige una petición a otro HTML, JSP o servlet.

    <jsp:forward page="***págDestino***"/>
  - <jsp:useBean> : Instancia o referencia un bean asignando nombre y ámbito.

    <jsp:useBean id="miBean" scope="ambito" class="ejemplos.ejtBean" />
  - <jsp:getProperty> : Obtiene el valor de una propiedad de un bean instanciado

    <jsp:getPropery name="miBean" property="nombre"/>
  - <jsp:setProperty> : Asigna el valor de una propiedad de un bean

    <jsp:setProperty name="miBean" property="nombre" value="Pepe" />

    <jsp:setProperty name="miBean" property="nombre" param="name" />

Contenido

- Introducción a JSP
- Elementos JSP
- **Objetos Implícitos en JSP y EL**
- JSTL
- MVC

JSP: Objetos implícitos

- Objetos creados por el contenedor de Servlets
  - **request** : Instancia de javax.servlet.ServletRequest. Encapsula la petición del cliente.
  - **response**: Instancia de javax.servlet.ServletResponse. Encapsula la respuesta generada por el JSP para enviar al cliente.
  - **out** : Instancia de javax.servlet.jsp.JspWriter, es un objeto PrintWriter usado para devolver la respuesta al cliente.
  - **session** : Instancia de javax.servlet.http.HttpSession. Representa la sesión creada para las peticiones de un cliente. Las sesiones se crean automáticamente. Mapeado a SessionScope para EL.
  - **application** : Instancia de javax.servlet.ServletContext. Representa el contexto dentro del cual el JSP se está ejecutando.
  - **pageContext** : Instancia de javax.servlet.jsp.PageContext. Encapsula el contexto de la página para un JSP específico.
  - **config** : Objeto que permite recuperar la información (atributos, etc) del fichero de configuración web.xml
  - **page** : Instancia de java.lang.Object. Representa a la instancia de la clase del JSP; es decir, el propio JSP.
  - **exception** : Instancia de java.lang.Throwable

JSP: Objetos implícitos con EL

pageScope // Atributos según su ámbito (equivalente getAttribute)

requestScope

sessionScope

applicationScope

param // Parámetros enviados en la petición (equivalente getParameter)

paramValues

header // Cabeceras (equivalenete a getHeaders)

headerValues

cookie

initParam // Párametros de inicialización

pageContext // Objeto pageContext

Contenido

- Introducción a JSP
- Elementos JSP
- Objetos Implícitos en JSP y EL
- **JSTL: JSP Standard Tag Library**
- MVC

JSP: Librerías de etiquetas estándar (JSTL)

- JSTL (JSP Standard Tag Library).

  Es una librería o colección de etiquetas estándar que encapsulan funcionalidad de uso muy frecuente en los JSP.

  Beneficios:

  - Reducir el tiempo de desarrollo de una página JSP.
  - Separación de conceptos: programación Java (lógica del negocio) del diseño web (presentación).

JSP: Librería de etiquetas estandar (JSTL)

- Funcionalidad encapsulada en JSTL. Áreas:

core	:	Entrada y Salida, condicionales y bucles
xml	:	Procesamiento de documentos XML
sql	:	Acceso y utilización de Bases de Datos
fmt	:	Capacidades de formateo de Internacionalización
fn	:	Funciones

- Configuración:

Incluir los ficheros standard.jar y jstl.jar en el directorio WEB-INF/lib.

- Utilización

```
<%@taglib  prefix="xxx"  uri="url_al_tld"  %>
…
<xxx:accion ….>  ….
```

JSP: Librería de etiquetas estandar (JSTL)

- Utilización

core: <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
xml: <%@taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
fmt: <%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
sql: <%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
fn: <%@taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

Jakarta

core: <%@taglib prefix="c" uri="jakarta.tags.core" %>
xml: <%@taglib prefix="c" uri="jakarta.tags.core" %>
fmt: <%@taglib prefix="c" uri="jakarta.tags.core" %>
sql: <%@taglib prefix="c" uri="jakarta.tags.core" %>
fn: <%@taglib prefix="c" uri="jakarta.tags.core" %>

JSP: Librería de etiquetas estandar (JSTL)

Ejemplo

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

<html>
<head> <title>Ejemplo con JSTL</title>  </head>

<body>
   <c:forEach var="i" begin="1" end="10" step="1">
     <c:out value="${i}" />   <br />
   </c:forEach>
</body>
</html>
```

JSP: Librería de etiquetas estandar (JSTL)

- Principales etiquetas de core.

Entrada/Salida

```
<c:set  var="nombre"   value="valor"  />
<c:out  value="valor"  />
```

Condicionales

```
<c:if test="exp_test"> …  </c:if>

<c:choose>
        <c:when test="exp_test1"> …  </c:when>
        <c:when test="exp_test2"> …  </c:when>
        …
        <c:otherwise> … </c:otherwise>
</c:choose>
```

Itetaritivas

```
<c:forEach var="nombre" begin="ini" end="fin" step="paso">
    …
</c:forEach>
```

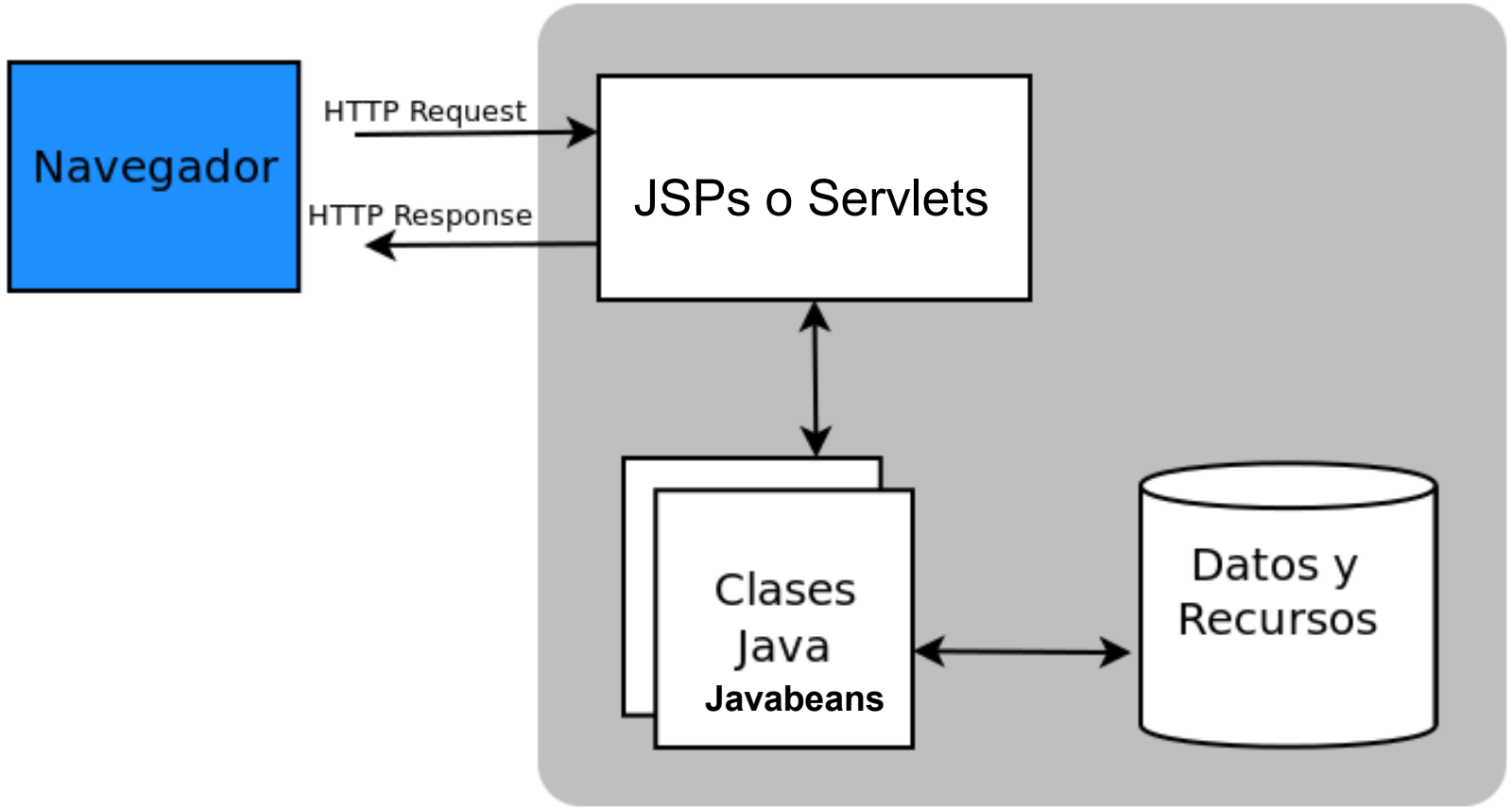
JSP: Librería de etiquetas estandar (JSTL)

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

<p>Asignado el valor del parámero numero a la variable num.
<c:set var="num" value="${param.numero}"/>  </p>

<p>Asignando la cadena "Hola, Mundo!" a la variable hola.
<c:set var="hola" value="Hola, Mundo!"/> </p>

<p>Mostrando una EL expresión: <c:out value="${1+5}" /></p>
<p>Mostrando variable hola: <c:out value="${hola}" /> </p>

<c:if test="${num>1 }">
  <p>num mayor que 10.
</c:if>
<c:if test="${num<1 }">
  <p>num menor que 10.
</c:if>

<p>
<c:forEach var="i" begin="1" end="10" step="1">
   <c:out value="${i}" />   <br />
</c:forEach>
</p>
```

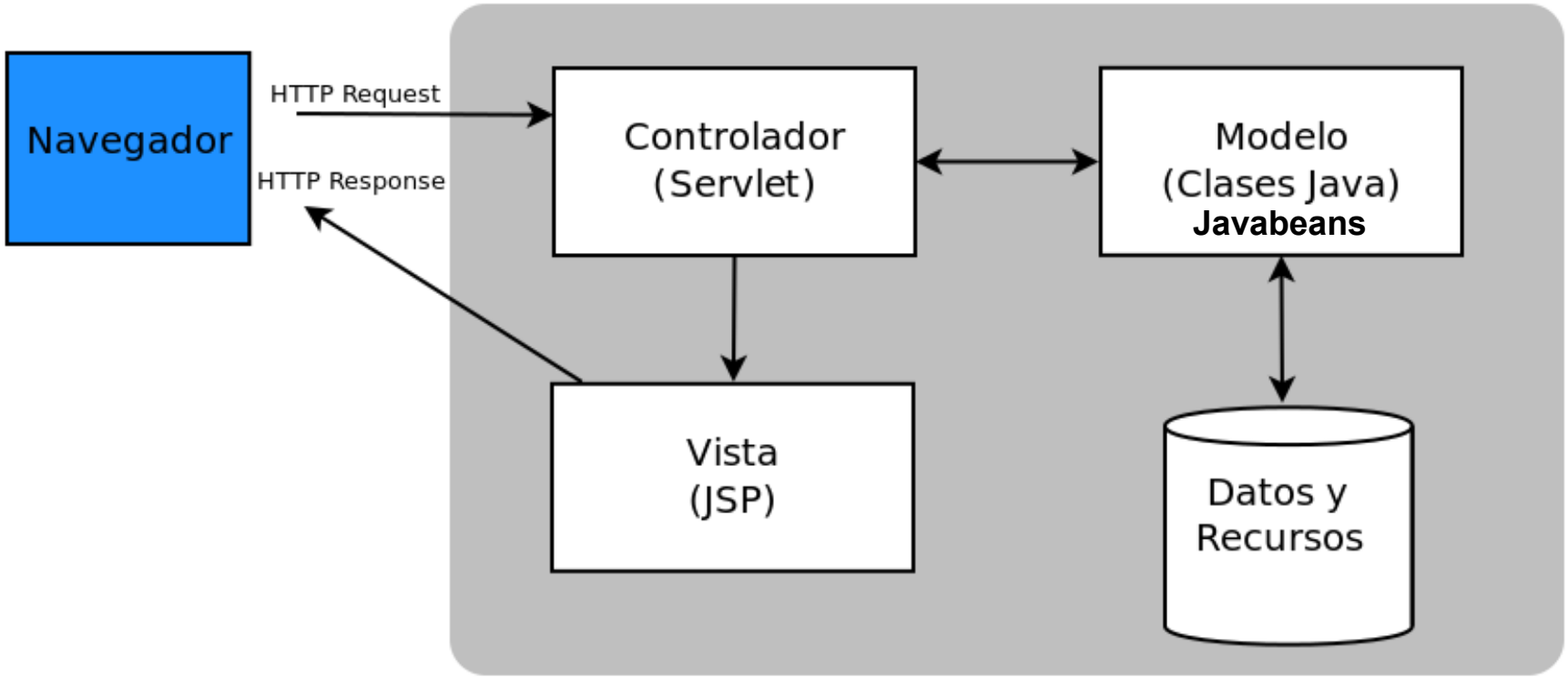
Contenido

- Introducción a JSP
- Elementos JSP
- Objetos Implícitos en JSP y EL
- JSTL
- MVC: Patrón Modelo-Vista-Controlador (Model 2)

MODEL 1


Navegador
HTTP Request
HTTP Response
JSPs o Servlets
Clases
Java
Javabeans
Datos y
Recursos

MODEL 2 – MVC: Model View Controller



MVC: Model View Controller (Model 2)

- Delegar peticiones de un Servlet
  - Obtener un objeto RequestDispacher desde el objeto petición:

    RequestDispacher rd = request.getRequestDispacher("vista");

  - Delegar la petición a partir de ese objeto

    rd.forward (request, response).

Los atributos creados en el ámbito de la petición son pasados a la vista.

Métodos: request.getAttribute() Objeto EL: requestScope
request.setAttribute()
request.removeAttribute()

MVC: Model View Controller (Model 2)

- Ejemplo 1: MVC

```
@WebServlet(name = "controller", urlPatterns = {"/do/*"})
public class controller extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String accion = request.getPathInfo();
        String vista;

        switch (accion) {
            case "/insertar":
                // Aquí vendría el código para insertar
                vista="/insert.jsp";
                break;
            case "/editar":
                // Aquí vandría el código para editar
                vista="/edit.jsp";
            // etc: todas las acciones
                break;
            default:
                vista="/error.jsp";
                break;
        }

        RequestDispatcher rd = request.getRequestDispatcher(vista);
        rd.forward(request, response);
```

MVC: Comunicación Model View Controller

- Atributos de la petición (ámbito request)

  request.setAttribute(“att”, objeto) : Crean un atributo de ámbito request
  request.getAtibute(“att”) : Obtiene el objeto “att” de la petición
  request.removeAttribute(“att”) : Elimina el atributo de la petición

- Atributos Globales (ámbito application)

```
ServletContext sc = getServletContext();
sc.setAttribute("att", objeto);
sc.getAttribute("att");
sc.removeAttribute("att", objeto);
```

- Atributos de sesión (ámbito session) (lo veremos más adelante)

MVC: Comunicación Model View Controller

- Ejemplo 2: MVC – Pasando datos a las vistas (Ámbitos)

```java
@WebServlet(name = "controller", urlPatterns = {"/do/*"})
public class controller extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        request.setAttribute("att1", "hola");
        ServletContext sc = getServletContext();
        sc.setAttribute("att1", new Integer(10));

        String accion = request.getPathInfo();
        String vista = "/index.jsp";

        /* ... */

        RequestDispatcher rd = request.getRequestDispatc
        rd.forward(request, response);
```

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>

        <p> Att1: <%= request.getAttribute("att1") %> </p>
        <p> Att1: ${requestScope.att1} </p>


        <p> Att1: <%= application.getAttribute("att1") %> </p>
        <p> Att1: ${applicationScope.att1} </p>

    </body>
</html>
```

MVC: Comunicación Model View Controller

- Ejemplo 3: MVC – Usar un Beans

```java
@WebServlet(name = "controller", urlPatterns = {"/do/*"})
public class controller extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        Usuarios u=new Usuarios(); // Completamos sus datos
        u.setNombre("Pepe");
        request.setAttribute("user", u);
        String vista = "/index.jsp";
        /* ... */
        RequestDispatcher rd = request.getRequestDispatcher(vista);
        rd.forward(request, response);
```

```java
package edu.daw.ejMVC;

public class Usuarios {

    private String nombre;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
}
```

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>

        <jsp:useBean id="user" scope="request" class="edu.daw.ejMVC.Usuarios" />

        <p>Hola <jsp:getProperty name="user" property="nombre"/></p>

        <jsp:setProperty name="user" property="nombre" value="Pedro"/>
        <p>Te he bautizado ahora como ${user.nombre}</p>

    </body>
</html>
```

MVC: Comunicación Model View Controller

- Ejemplo 4: MVC – Pasar una lista de objetos

Controlador

```
case "/accion3":
    Usuario u1 = new Usuario();
    u1.setNombre("Pepe");
    Usuario u2 = new Usuario();
    u2.setNombre("Juan");
    ArrayList<Usuario> lu = new ArrayList();
    lu.add(u1);
    lu.add(u2);
    request.setAttribute("users", lu);

    vista = "/vista3.jsp";
    break;
```

Vista

```
<h1>Usuarios</h1>
<%
    List<Usuario> lu = (List<Usuario>) request.getAttribute("users");
    if (lu != null) {
        out.println("<ul>");
        for (Usuario u : lu) {
            out.println("<li>" + u.getNombre() + "</li");
        }
        out.println("</ul>");
    } else {
        out.println("<p>No hay Usuarios</p>");
    }

%>
```

MVC: Comunicación Model View Controller

- Ejemplo 4: MVC – Pasar una lista de objetos (Alternativa JSTL)

Controlador

```
case "/accion3":
    Usuario u1 = new Usuario();
    u1.setNombre("Pepe");
    Usuario u2 = new Usuario();
    u2.setNombre("Juan");
    ArrayList<Usuario> lu = new ArrayList();
    lu.add(u1);
    lu.add(u2);
    request.setAttribute("users", lu);

    vista = "/vista3.jsp";
    break;
```

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

Vista

```
<h1>Usuarios</h1>
<c:choose>
    <c:when test="${!empty requestScope.users}">
        <ul>
            <c:forEach var="user" items="${requestScope.users}">
                <li>${user.nombre}</li>
            </c:forEach>
        </ul>
    </c:when>
    <c:otherwise>
        <p>No hay Usuarios</p>
    </c:otherwise>
</c:choose>
```