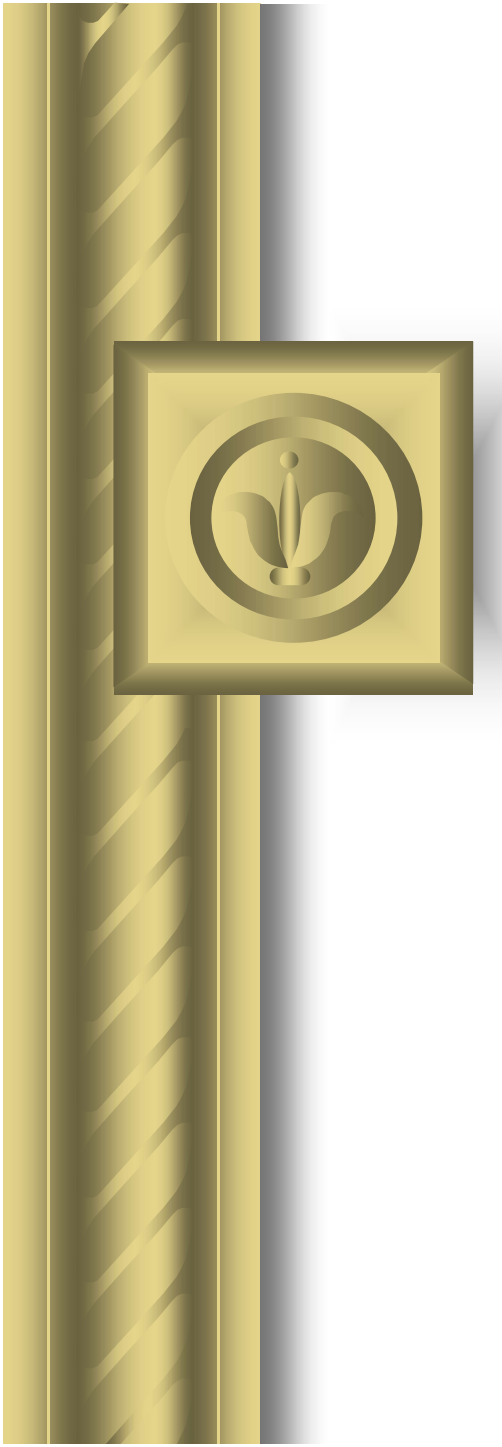
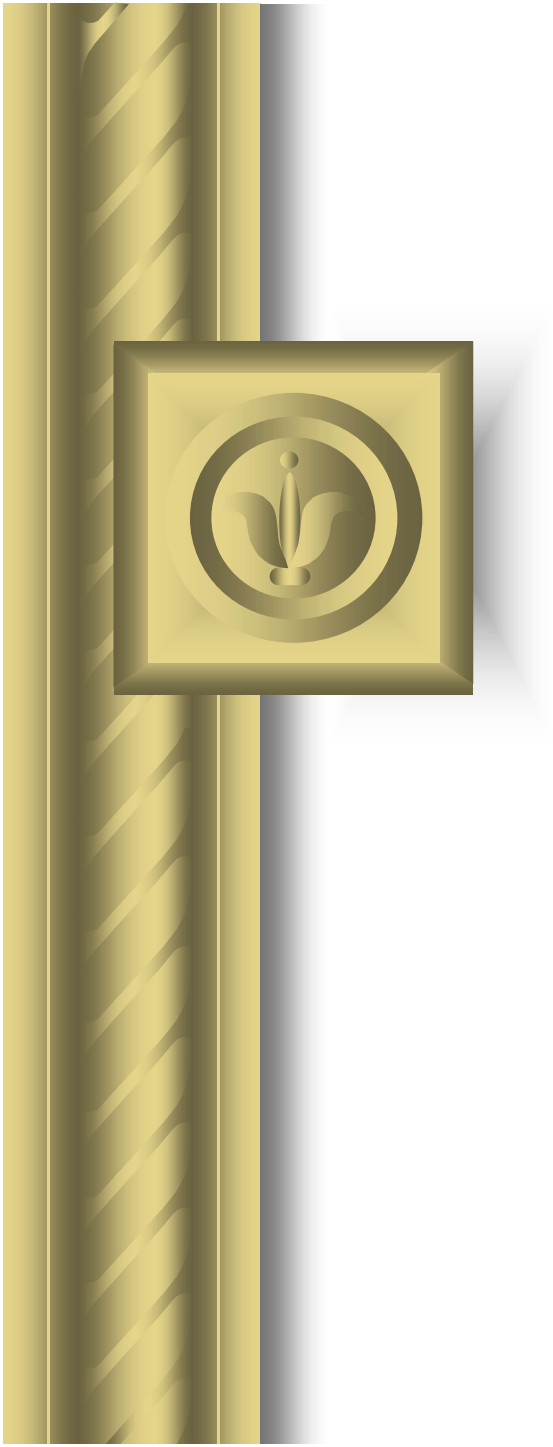


Java Server Pages (JSPs)

Desarrollo de Aplicaciones Web



Problema:

¿Cómo quitamos el código HTML de los Servlets?

Solución: Java Server Pages (JSPs)

– Documento de texto que puede ofrecer contenido estático y dinámico

– Contenido estático: HTML, XML, etc.

– Contenido dinámico: Código Java, etiquetas especiales, etc.

Ejemplo básico de JSP

A Simple JSP Page (Blue: static, Red: Dynamic contents)

```
<html>
<body>
  Hello World!
 <br>
 Current time is <%= new java.util.Date() %>
</body>
</html>
```

¿Por qué JSPs?

– Para **diseñadores Web**: Permiten crear páginas web dinámicas sin tener conocimiento de programación JAVA.

–Simples de crear: HTML con marcas para ejecutar código JAVA

– Simples de desplegar: No hace falta compilación previa.

– Para **programadores Web**: ¡¡Simplifican el Servlet!! Permiten quitar todo el código HTML de los Servlets.

¿JSPs VS Servlets?

- Servlets:
  1. Java con código HTML incrustado
  2. Fichero compilado previo a la petición

- JSP:
  1. HTML con marcas para ejecutar Java
  2. Código se compila en tiempo real en un Servlet

¿JSPs VS Servlets?: Ejemplo

GreetingServlet.java (1)

```
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * This is a simple example of an HTTP Servlet.  It responds to the GET
 * method of the HTTP protocol.
 */
public class GreetingServlet extends HttpServlet {

    public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException
 {

      response.setContentType("text/html");
      response.setBufferSize(8192);
      PrintWriter out = response.getWriter();

      // then write the data of the response
      out.println("<html>" +
```

¿JSPs VS Servlets?: Ejemplo

GreetingServlet.java (2)

```
        // then write the data of the response
        out.println("<body  bgcolor=\"#ffffff\">" +
            "<img src=\"duke.waving.gif\">" +
            "<h2>Hello, my name is Duke. What's yours?</h2>" +
            "<form method=\"get\">" +
            "<input type=\"text\" name=\"username\" size=\"25\">" +
            "<p></p>" +
            "<input type=\"submit\" value=\"Submit\">" +
            "<input type=\"reset\" value=\"Reset\">" +
            "</form>");

        String username = request.getParameter("username");

        // dispatch to another web resource
        if ( username != null && username.length() > 0 ) {
              RequestDispatcher dispatcher =
                 getServletContext().getRequestDispatcher("/response");

              if (dispatcher != null)
              dispatcher.include(request, response);

        }
        out.println("</body></html>");
        out.close();
    }
```

¿JSPs VS Servlets?: Ejemplo

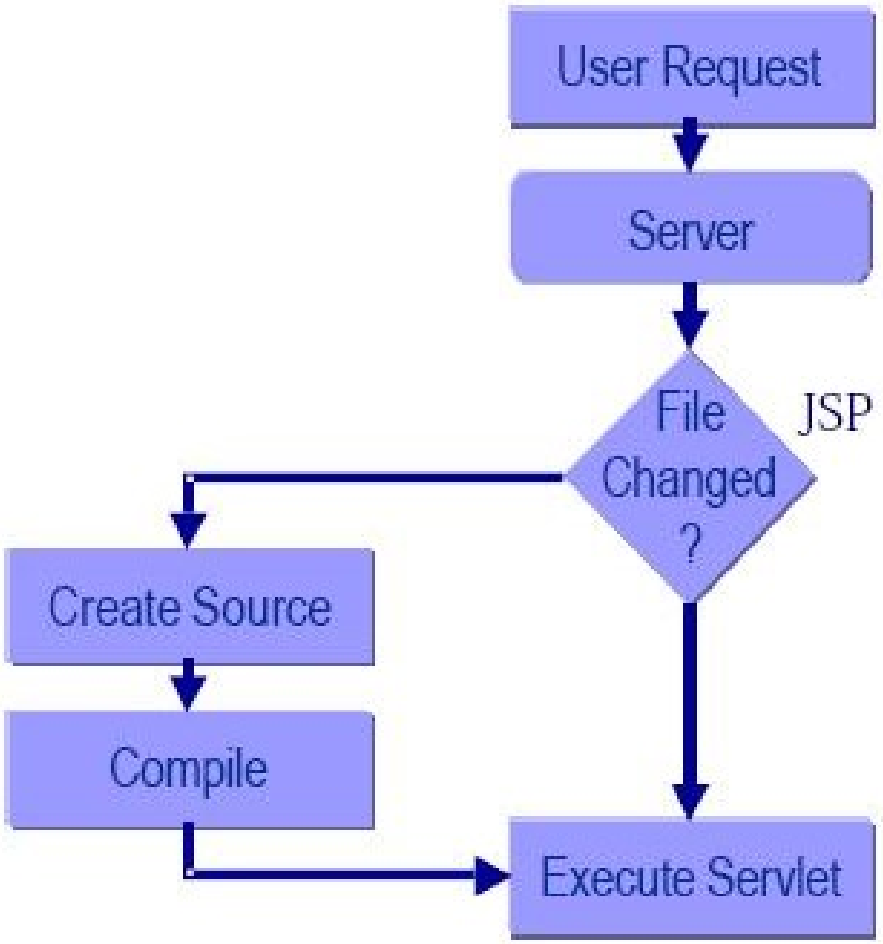
greeting.jsp

```
<html>
<head><title>Hello</title></head>
<body bgcolor="white">
<img src="duke.waving.gif">
<h2>My name is Duke. What is yours?</h2>

<form method="get">
<input type="text" name="username" size="25">
<p></p>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</form>

<%
    String username = request.getParameter("username");
    if ( username != null && username.length() > 0 ) {
%>
    <%@include file="response.jsp" %>
<%
    }
%>
</body>
</html>
```

Ciclo de vida



Elementos de script para insertar codigo en JSPs

– Scriptlets:

<% código Java %>

–Expresiones:

<%= Expresion %>

– Declaraciones:

<%! código Java %>

Ejemplos de elementos de script

– Scriptlets:

```
<%
String  DatosCliente = request.getParameter(“datosFormulario”);
out.println(“Los datos introducidos son: “ + DatosCliente);
 %>
```

–Expresiones:

```
Los datos introducidos son:
<%= request.getParameter(“datosFormulario”) %>
```

– Declaraciones:

```
<%! private int contador = 0 %>
```

¿Cómo llamamos a un JSP desde un Servlet?

Clase RequestDispatcher: Clase que reenvia la respuesta a la vista definida en el objeto “address”.

```
private void gotoPage(String address, HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // Creamos objeto RequestDispatcher
    RequestDispatcher dispatcher = getServletContext().getRequestDispatcher(address);
    dispatcher.forward(request, response);
}
```

Llamada al método gotoPage desde los métodos doGet o doPost:

```
// Presentamos Vista del Carrito
gotoPage("/VistaCarrito.jsp", request, response);
```

¿Cómo recuperamos objetos de la sesión desde un JSP?

1. Configuramos la página JSP para acceder a la sesión:

```
<%@page session="true" %>
```

2. Accedemos al objeto sesión desde un scriptlet

```
<%
// Extraemos un vector de la sesion
Vector vectorCD =
(Vector)session.getAttribute("vectorCD");
%>
```

Ejercicios:

- Sustitución de un Servlet por una página JSP. Vamos a sustituir el Servlet de la aplicación “despliegueTomcat2” por una página JSP que haga lo mismo. Es decir, va a recibir los parámetros del formulario de la página inicial y después los integra con el HTML para mostrarlos al usuario. La tarea, por tanto, consiste en completar la página login.jsp del ejercicio “despliegueTomcat2JSP”.

- Combinación de Servlets con JSPs. La tarea consiste en mejorar la implementación de la aplicación “despliegueTomcat2”. Para ello hay que modificar el Servlet “login.java” para que una vez leidos los parámetros redirija el flujo de la aplicación a una página JSP. Esta es la que se encargará de mostrar al usuario los parámetros introducidos.