

JSP

PARTE 1

Arquitectura MVC / Objetos implícitos / Vista

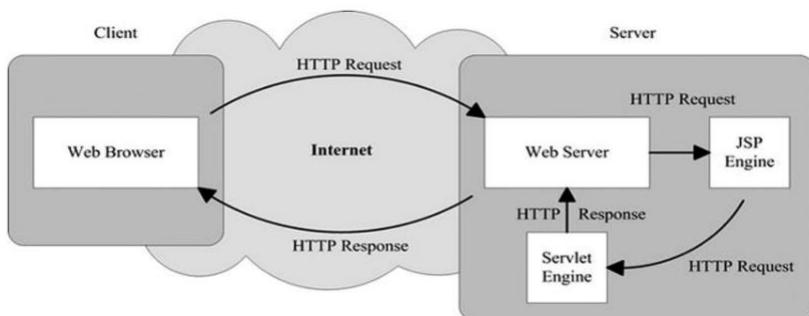
Con Apache Tomcat
(Servidor Web y JSP)



¿Qué es y para qué sirve?

Funcionalidad y características

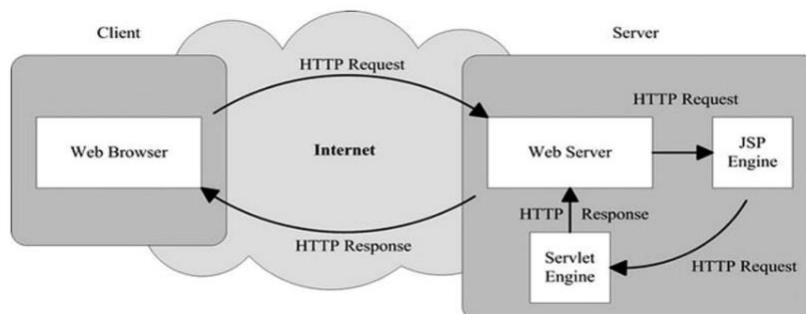
- Es una tecnología que permite añadir dinámicamente contenido Web a nuestras páginas/aplicaciones
- Ejecución de JSPs y Servlets en el lado del servidor (Servidor Web dinámico)
- La página Web no existe en el servidor, se genera dinámicamente



¿Qué es y para qué sirve?

Funcionalidad y características

- Cada página JSP es un fichero con código XHTML que incluye scripts con código Java
- Podemos hacer uso de código Java en la parte de los **scriptlets**
- Se puede hacer uso de etiquetas especializadas (*Custom Tags*) que nos facilitan el manejo de Html
- Se compila, se convierte en un servlet (solo la primera vez que se invoca) y se ejecuta como un Servlet





Pasos que sigue el servidor para crear una página Web a partir de una página JSP

Pasos (JSP -> HTML)

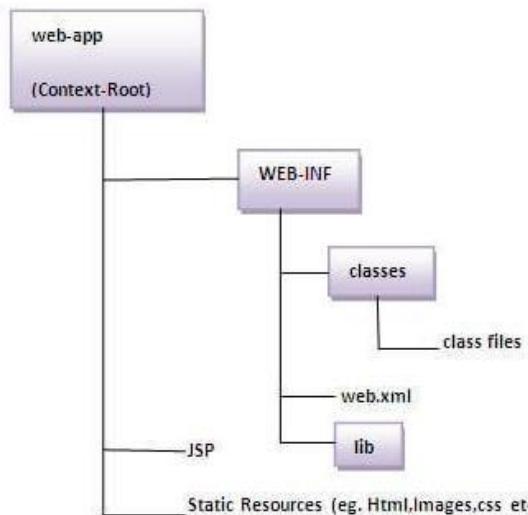
1. Se envía una solicitud HTTP desde un cliente (Navegador) a un servidor JSP, la solicitud es de una página .jsp
2. El Motor JSP (En el servidor), carga la página JSP y la convierte en un **servlet Java**
3. El motor JSP reenvia el servlet al motor de Servlet (Sólo la vuelve a convertir a servlet si la página JSP ha cambiado: eficiencia)
4. El motor servlet carga el servlet y lo ejecuta. El motor servlet produce una salida en formato HTML y esta es enviada al servidor Web
5. El servidor Web envía la respuesta al Navegador
6. El Navegador muestra la página devuelta como una página estática

(*) Todo proyecto Web tiene un fichero de configuración llamado **web.xml** donde podemos configurar parámetros a usar dentro de nuestra aplicación Web (**webapp\WEB-INF\web.xml**)



Proyecto JSP

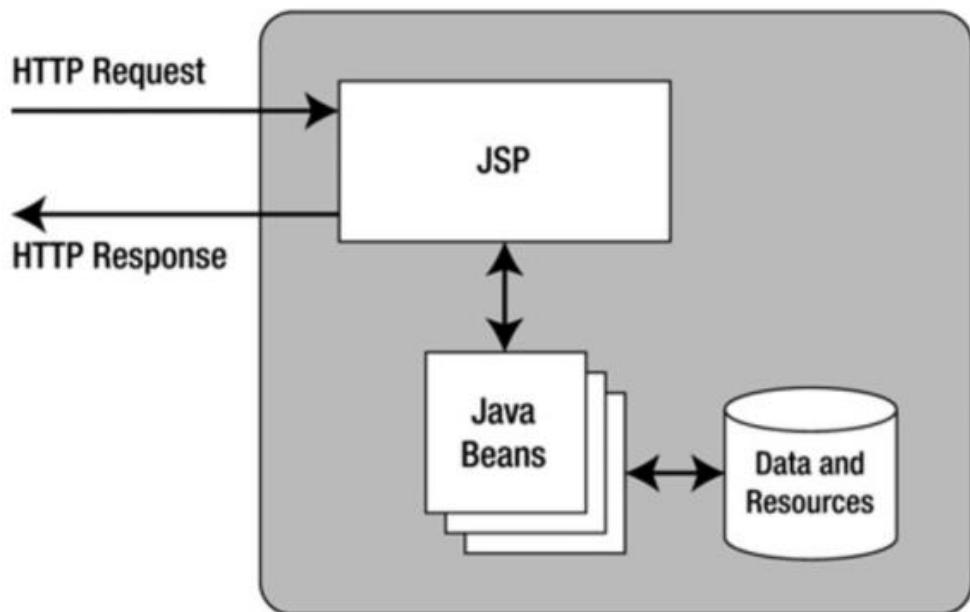
Estructura proyecto JSP



(*) Todo proyecto Web tiene un fichero de configuración llamado **web.xml** donde podemos configurar parámetros a usar dentro de nuestra aplicación Web (**webapp\WEB-INF\web.xml**)

JSP Modelo 1

Modelo 1





JSP

Arquitectura

JSP Modelo 1

Modelo 1

- El Modelo 1 es aceptable pero puede crear quebraderos de cabeza para aplicaciones complejas donde la capa de presentación es muy extensa, habría una mezcla de código Html, Css y Jsp poco mantenible.
- En este Modelo la página JSP se encarga tanto de la vista como de la lógica
- La página JSP se encarga de gestionar todas las peticiones Http , la lógica e instanciar los JavaBeans



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO

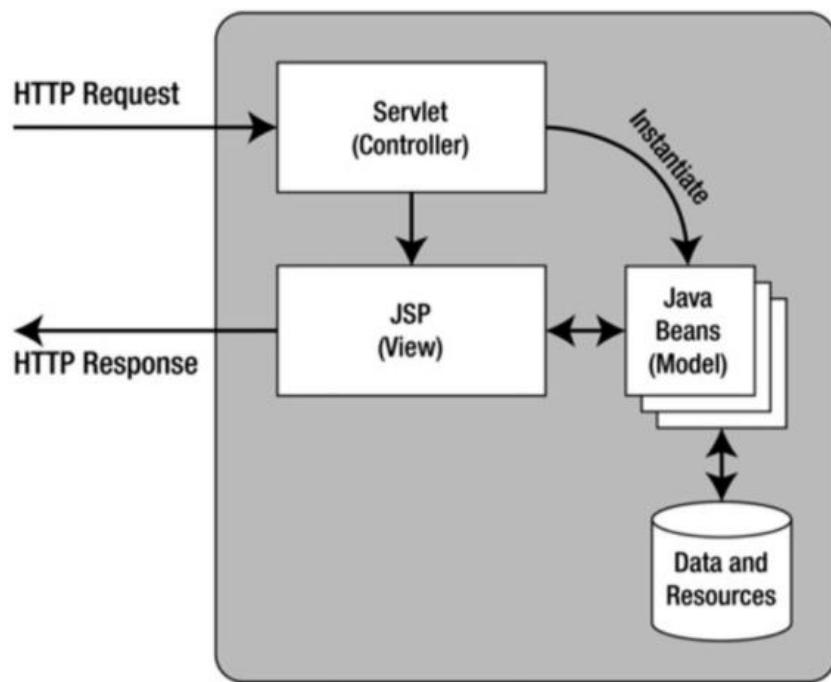


**Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil**
El FSE invierte en tu futuro



JSP Modelo 2 : MVC

MVC





JSP

Arquitectura

JSP Modelo 2 : MVC

MVC

- El Modelo Vista Controlador es el más acertado para desarrollo de aplicaciones más extensas y complejas
- Un Servlet procesa las peticiones, maneja la lógica de la aplicación e instancia los Java Beans
- Las páginas JSP se encargan de la presentación, obtienen los datos de los Beans



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



**Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil**
El FSE invierte en tu futuro





Ejercicio1

holaMundo.jsp

- Crear un fichero llamado **holaMundo.jsp** con el siguiente contenido y probarlo

```
| holaMundo.jsp ✘
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2     pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4<html>
5<head>
6 <meta charset="ISO-8859-1">
7 <title>Insert title here</title>
8 </head>
9<body>
.0     Hola Mundo !!!!|
.1 </body>
.2 </html>
```



JSP

Contenido dinámico

Mostrar contenido dinámico con **out**

Contenido dinámico

- Podemos introducir código Java en la página JSP dentro de <% ... Código Java ... %> esto es llamado **scriptlet** y es lo que es convertido a **servlet**
- Podemos mostrar contenido dinámico en nuestra página con la variable **out** la cual está definida en cada servlet generado
- Todo lo que mostremos con **out** tiene que ser **código Html**

```
<%  
    out.println ("<br/> <b>Esto es una prueba</b>");  
%>
```



GOBIERNO
DE ESPAÑA
MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Import de librerías como en Java

Uso de librerías y parámetros

- Como ya hemos dicho , los **scriptlets** contienen código Java. Para poder usar cualquier librería de Java debemos realizar un **import** de la misma , al igual que hacemos en una clase Java

```
<%@page import="java.util.* , java.io.*"%>
```

- Las páginas HTML a través de los formularios, envían parámetros a la páginas JSP para procesar las solicitudes (Parámetros para realizar consultas a servicios Java que a su vez consultan o realizan operaciones en las tablas de una Base de datos).
- Los **servlets** disponen también de un Objeto llamado **request** que se crea en todos los **servlets** y nos ayuda a recoger todos los parámetros enviados desde un formulario Html

```
<%  
Map map = request.getParameterMap();  
Object[] keys = map.keySet().toArray();  
%>
```

Mostrar contenido dinámico con *out*

Contenido dinámico

- Formas de definir en código Java en una página JSP :

- 1. <%= expresión %>** : se evalua esa expresión y se inserta en el código Html
- 2. <% código Java %>** : el código Java introducido se insertará en el método del servicio del Servlet
- 3. <%! declaraciones %>** : son declaraciones de variables
- 4. <%-- Comentario --%>** : son comentarios JSP



JSP

Contenido dinámico

Import de librerías como en Java

Uso de librerías y parámetros

```
<%  
  
Map map = request.getParameterMap();  
Object[] keys = map.keySet().toArray();  
  
for (int k = 0; k < keys.length; k++) {  
    String[] pars = request.getParameterValues((String)keys[k]);  
    out.print("<tr><td>" + k + "</td><td>" + keys[k] + "</td><td>");  
    for (int j = 0; j < pars.length; j++) {  
        if (j > 0) out.print(", ");  
        out.print(" " + pars[j] + " ");  
    }  
    out.println("</td></tr>");  
}  
%>
```



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



**Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil**
El FSE invierte en tu futuro





Mostrar contenido dinámico con *out*

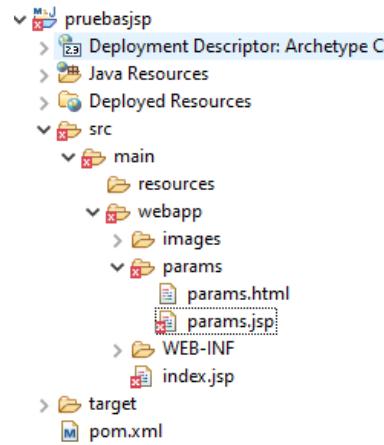
Contenido dinámico

- Crear una páginas Jsp llamada **formasCodigo.jsp** que haga lo siguiente :
 - a) Muestre una fecha desde Java usando `<%= expresión %>`
 - b) Muestre el resultado de ejecutar un bucle for de 1 a 10, mostrando todos los valores de las iteraciones , hacerlo con `<% código Java %>`
 - c) Declarar una variable String con valor “ESTE ES EL VALOR” usando `<%! declaraciones %>` y visualizarla después usando `<% código Java %>`

Procesamiento de parámetros

Uso de parámetros

- A partir de ahora vamos a crear un par **html/jsp** donde cada **html** enviará los parámetros de un formulario a su **jsp** emparejado
- Crear un fichero **params.html** y otro **params.jsp** dentro de una carpeta **webapp\params**



Procesamiento de parámetros

Uso de parámetros

```
params.html ✘
1 <!DOCTYPE html>
2<html>
3<head>
4 <meta charset="ISO-8859-1">
5 <title>Insert title here</title>
6 </head>
7<body>
8
9   <!-- ENVÍO POR GET -->
10<form method="GET" action=".//params.jsp">
11   <input type="text" name="texto">
12   <button type="submit">Enviar</button>
13</form>
14
15 <!-- ENVÍO POR POST -->
16<form method="POST" action=".//params.jsp">
17   <input type="text" name="texto">
18   <button type="submit">Enviar</button>
19</form>
20</body>
21</html>
```

```
params.jsp ✘
1 <%@page language="java" contentType="text/html"%>
2 <%@page import="java.util.*, java.io.*"%>
3<%
4   Map map = request.getParameterMap();
5 Object[] keys = map.keySet().toArray();
6 %>
7
8<html>
9<head>
10 <title>Request Parameters</title>
11 </head>
12<body>
13   Map size =
14   <%=map.size()%>
15   <table border="1">
16     <tr>
17       <td>Map element</td>
18       <td>Par name</td>
19       <td>Par value[s]</td>
20     </tr>
21   <%>
22     for (int k = 0; k < keys.length; k++) {
23       String[] pars = request.getParameterValues((String) keys[k]);
24       out.print("<tr><td>" + k + "</td><td>" + keys[k] + "</td><td>");
25       for (int j = 0; j < pars.length; j++) {
26         if (j > 0)
27           out.print(",");
28         out.print(" " + pars[j] + " ");
29       }
30       out.println("</td></tr>");
31     }
32   </table>
33 </body>
34 </html>
```



Procesamiento de parámetros

Uso de parámetros

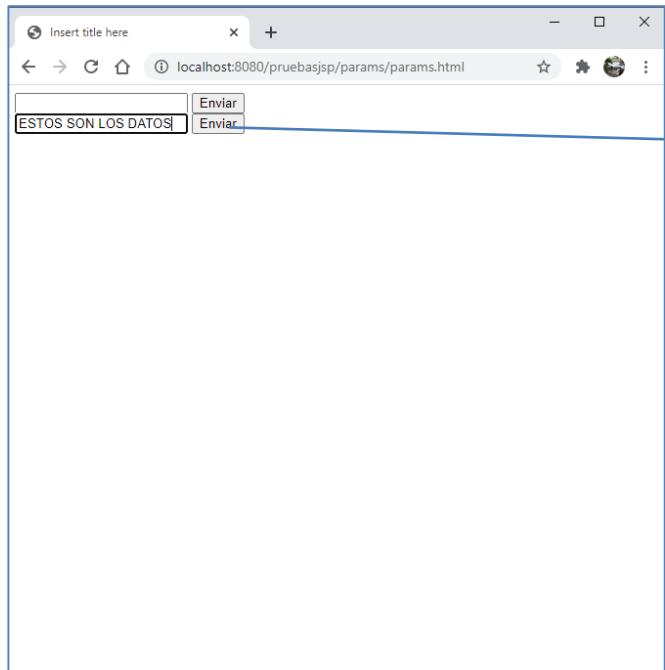
```
params.html ✘
1 <!DOCTYPE html>
2<html>
3<head>
4 <meta charset="ISO-8859-1">
5 <title>Insert title here</title>
6 </head>
7<body>
8
9   <!-- ENVÍO POR GET -->
10<form method="GET" action=".//params.jsp">
11  <input type="text" name="texto">
12  <button type="submit">Enviar</button>
13</form>
14
15 <!-- ENVÍO POR POST -->
16<form method="POST" action=".//params.jsp">
17  <input type="text" name="texto">
18  <button type="submit">Enviar</button>
19</form>
20</body>
21</html>
```

Cuando hagamos click en el botón de tipo **submit** en enviará los datos Del formulario a la página **jsp** que le hayamos indicado en el atributo **action** del elemento **form** (**params.jsp**) y el fichero **jsp** procesará los parámetros Que le hagan llegado y los mostrará por pantalla.

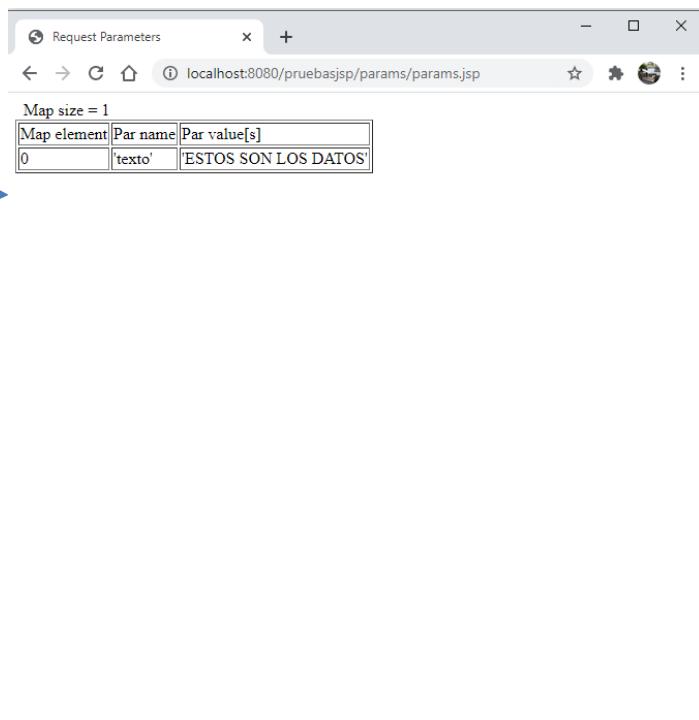
Enviará los parámetros por el método indicado en el atributo **method** del formulario

Procesamiento de parámetros

Uso de parámetros



A screenshot of a web browser window. The address bar shows "localhost:8080/pruebasjsp/params/params.html". The page contains a form with two input fields. The first field has the placeholder "ESTOS SON LOS DATOS". Below each field is a button labeled "Enviar". A blue arrow points from the right side of this window towards the "Request Parameters" window.



A screenshot of a browser window titled "Request Parameters". The address bar shows "localhost:8080/pruebasjsp/params/params.jsp". The content area displays a table with one row. The table has three columns: "Map element", "Par name", and "Par value[s]". The first row contains the values "0", "'texto'", and "ESTOS SON LOS DATOS".

Map element	Par name	Par value[s]
0	'texto'	ESTOS SON LOS DATOS

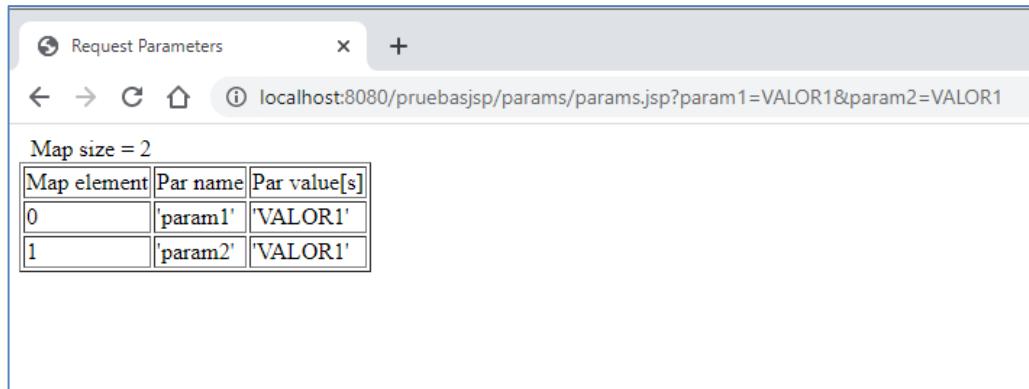
Procesamiento de parámetros

Uso de parámetros



A screenshot of a web browser window titled "Request Parameters". The address bar shows the URL "localhost:8080/pruebasjsp/params/params.jsp?param1=VALOR1¶m2=VALOR1".

- También puedo enviar los parámetros en la URL donde invoco al fichero JSP



A screenshot of a web browser window titled "Request Parameters". The address bar shows the URL "localhost:8080/pruebasjsp/params/params.jsp?param1=VALOR1¶m2=VALOR1". Below the address bar, a table displays the received parameters:

Map element	Par name	Par value[s]
0	'param1'	'VALOR1'
1	'param2'	'VALOR1'

Objetos implícitos JSP



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



**Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil**
El FSE invierte en tu futuro



Elementos de acción personalizados y lenguajes de expresiones

JSTL y EL

- El mecanismo de extensión de etiquetas nos permite crear acciones personalizadas. Hay un grupo de acciones que se han hecho de uso muy extendido y se han estandarizado (JSTL -> biblioteca de etiquetas estándar JSP)
- También es muy útil el lenguaje de expresiones el cual nos permite un fácil acceso a objetos (Beans de Java)
- Todo esto lo veremos más adelante

Objectos creados por el contenedor Web

Disponibles para su uso en todas las páginas JSP del proyecto

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

Objeto request

Uso en scriptlets

- El Objeto **request** nos permite gestionar la petición recibida por el cliente, nos permite acceder a la información de la solicitud, nos da acceso a todas los parámetros/variables enviado por una página cliente (Html, Jsp ,...) . Podemos recuperar toda la información enviada por una petición de cliente (parámetros, Información de la cabecera de la petición, parameter, info sobre la dirección remota, info sobre el nombre del servidor, info sobre el puerto del servidor, info sobre el tipo de contenido, info sobre el tipo de codificación usado ,).

```
String myPar = request.getParameter("par-name");
```

```
Enumeration headers = request.getHeaderNames();
```

Objeto request

Uso en scriptlets

- Métodos del Objeto **request** :

- **getParameter(String name)** : devuelve el valor del parámetro
- **getParameterNames()** : devuelve un enumerado con los nombres de los parámetros
- **getParameterValues(String name)** : devuelve un array con los valores de los parámetros
- **getAttribute(String name)** : devuelve el valor de un atributo
- **getAttributeNames()** : devuelve un enumerado con los nombres de los atributos de la sesión
- **setAttribute(String, Object)** : establece en valor de un atributo para la sesión
- **removeAttribute(String)** : elimina un atributo
- **getCookies()** : devuelve un array con las cookies recibidas desde el cliente
- **getHeader(String name)** : devuelve información de la cabecera de la petición
- **getHeaderNames()** : devuelve un enumerado con los nombres de las cabeceras
- **getRequestURI()** : devuelve la URL de la actual página JSP
- **getMethod()** : devuelve el método usado en la petición
- **getQueryString()** : devuelve la Query String asociada a la petición (<http://url?var1=valor1&var2=valor2>)

Objeto request

request.jsp

- Crear dos ficheros **request.html** y **request.jsp** y una hoja de estilos general en la ruta **css/estilos.css**
- **request.html** -> formulario con 4 parámetros (edad, nombre, apellidos, dirección) y que envíe esta información a la página **request.jsp** mediante el método GET

A diagram of a form with four input fields stacked vertically. Each field has a teal header and a white input box. The headers are "Edad", "Nombre", "Apellidos", and "Dirección". Below the input boxes is a single "Enviar" button.

- **request.jsp** -> cuando reciba la petición enviada por **request.html** que la muestre en pantalla de la siguiente manera
- Probar el resto de métodos descritos anteriormente

Edad :3
Nombre :noaso
Apellidos :asoi
Direcciónn :asoi

Objeto response

Uso en scriptlets

- El Objeto **response** nos permite gestionar/escribir la respuesta que queremos enviar al cliente. Incluye la definición de 41 código de estado para enviar el cliente el estado de la respuesta, son código numéricos : **100-199** son reservados para enviar información general, **200-299** son para enviar información sobre el éxito de la petición realizada, **300-399** son reservados para informar sobre warnings en la resolución de la petición, **400-499** son reservados para informar sobre errores en el cliente y **500-500** para informar sobre errores en el servidor.
- Los más usados son **200** para informar que todo ha ido correctamente y **404** para indicar que un recurso no existe (Por ejemplo una página JSP a la que se invoca)
- Podemos usarlo para reenviar la respuesta a otra página JSP o HTML con **sendRedirect(URL)**

```
response.sendRedirect(http://www.otrapagina.com);
```

Objeto response

Uso en scriptlets

Métodos del Objeto response :

- **setContentType(String type)** : indica al navegador el tipo de dato enviado en la respuesta (https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types)

```
response.setContentType("text/html");
response.setContentType("image/gif");
response.setContentType("image/png");
response.setContentType("application/pdf");
```
- **sendRedirect(String address)** : redirige la petición a una nueva página
- **addHeader(String name, String value)** : añade una cabecera a la respuesta
- **setHeader(String name, String value)** : modifica una cabecera existente por un nuevo valor
- **containsHeader(String name)** : devuelve true/false si la cabecera existe
- **addCookie(Cookie cookie)** : añade una cookie a la respuesta

Objeto response

Uso en scriptlets

Métodos del Objeto response :

- **sendError(int status_code, String message)** : devuelve un error con el código establecido (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>)

```
response.sendError(404, "Page not found error");
```

- **isCommitted()** : comprueba si la respuesta ha sido enviada al cliente (true=sí, false=no)

```
<% if(response.isCommitted())
{
    <%--do something --%>
} else
{
    <%--do something else --%>
} %>
```

- **setStatus(int statuscode)** : establece el estado de la respuesta

```
response.setStatus(404);
```

Objeto response

response.jsp

- Crear dos ficheros **response.html** y **response.jsp**
- **response.html**-> formulario con 4 parámetros (edad, nombre, apellidos, dirección) y que envíe esta información a la página **response.jsp** mediante el método POST

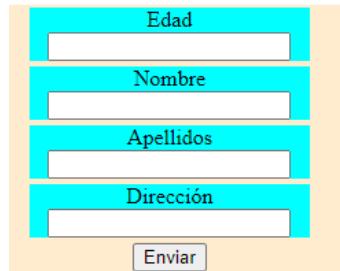


Diagrama de un formulario de respuesta. El formulario consta de cuatro campos de texto horizontalmente dispuestos: 'Edad', 'Nombre', 'Apellidos' y 'Dirección'. Abajo de estos campos hay un botón rectangular que dice 'Enviar'.

- **response.jsp** -> cuando reciba la petición enviada por **response.html** que la muestre en pantalla de la siguiente manera y redirija de nuevo a la página **response.html**
- Probar el resto de métodos descrito anteriormente

Edad :3
Nombre :noaso
Apellidos :asoi
Direcciónn :asoi

Objeto config

Uso en scriptlets

El Objeto **config** lo usa **Tomcat** para pasar información a los Servlets, por ejemplo lo podemos usar para recuperar el nombre del servlet u otros parámetros definidos en el fichero de configuración de la aplicación Web **web.xml**

index.html

```
<form action="welcome">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

web.xml file

```
<web-app>
<servlet>
<servlet-name>sonoojaishwal</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>

<init-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</init-param>

</servlet>

<servlet-mapping>
<servlet-name>sonoojaishwal</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

</web-app>
```

welcome.jsp

```
<%
out.print("Welcome "+request.getParameter("uname"));

String driver=config.getInitParameter("dname");
out.print("driver name is"+driver);
%>
```

Objeto config

Uso en scriptlets

- Métodos del Objeto config :

- **getInitParameter(String paramname)** : devuelve el valor de un parámetro de inicialización configurado en el fichero web.xml del proyecto

```
<web-app>
  ...
  <context-param>
    <param-name>parameter1</param-name>
    <param-value>ValueOfParameter1</param-value>
  </context-param>
</web-app>
```

- **getInitParameterNames()** : devuelve un enumerado con los parámetros declarados en el fichero web.xml
- **getServletContext()** : devuelve una referencia al contexto del Servlet
- **getServletName()** : devuelve el nombre del servlet definido en el fichero web.xml (<servlet-name>....</servelet-name>)



JSP

Ejercicio

Objeto config

config.jsp

- Crear dos ficheros **config.html** y **config.jsp**
- Definir unos parámetros de configuración en el fichero **web.xml** y leerlos desde **config.jsp**
- Probar el resto de métodos de config



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Objeto exception

Uso en scriptlets

- El Objeto **exception** estña sólo disponible en las páginas de error , las páginas de error las activamos con `isErrorPage="true"`. Con el Objeto **exception** podemos recuperar información sobre la expceión lanzada (`getStackTrace`, `getClassName`, `getFileName`, ...)

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

```
StackTraceElement[] trace = exception.getStackTrace();  
for (int k = 0; k < trace.length; k++) {  
    out.println(trace[k]);  
}
```

Objeto exception

Uso en scriptlets

- Tambien podemos tratar las excepciones dentro del código de los Scriptlets

```
<html>
<head>
<title>Exception handling using try catch blocks</title>
</head>
<body>
<%
try{
    //I have defined an array of length 5
    int arr[]={1,2,3,4,5};
    //I'm assinging 7th element to int num
    //which doesn't exist
    int num=arr[6];
    out.println("7th element of arr"+num);
}
catch (Exception exp){
    out.println("There is something wrong: " + exp);
}
%>
</body>
</html>
```



JSP

Objetos implícitos

Objeto exception

Uso en scriptlets

```
<%@ page errorPage="exception.jsp" %>
<%
String num1=request.getParameter("firstnum");
String num2=request.getParameter("secondnum");
int v1= Integer.parseInt(num1);
int v2= Integer.parseInt(num2);
int res= v1/v2;
out.print("Output is: "+ res);
%>
```

exception.jsp

```
<%@ page isErrorPage="true" %>
Got this Exception: <%= exception %>
Please correct the input data.
```



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro





JSP

Ejercicio

Objeto exception

exception.jsp

- Implementar las dos maneras explicadas de manejar excepciones



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



**Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil**
El FSE invierte en tu futuro



Objeto application

Uso en scriptlets

- **<%= expresión%>** : Inserta el resultado de evaluar una expresión en Java

```
<%@page import="java.util.Date"%>  
Fecha : <%=new Date()%>
```

- **application** : nos permite acceder a recursos compartidos dentro de la aplicación Web, por ejemplo `application.getInitParameter("parametro1")` devolverá el parámetro configurado en el fichero **web.xml**(Fichero de configuración del proyecto Web)
- Podemos establecer atributos compartidos en toda la aplicación con `application.setAttribute("nombre","valor")` recuperarlos después con `application.getAttribute("nombre")`

```
<%application.setAttribute("nombre","valor"); %>  
out.print(application.getAttribute("nombre"));|
```

Objeto application

Uso en scriptlets

- Métodos del Objeto **application** :

- **getAttribute(String attributeName)** : devuelve el valor de un atributo

```
String s = (String)application.getAttribute("MyAttr");
```

- **setAttribute(String attributeName, Object object)** : establece el valor de un atributo

- **removeAttribute(String objectName)** : elimina un atributo

- **getAttributeNames()** : devuelve un enumerado con todos los nombres de los atributos

- **getInitParameter(String paramname)** : devuelve el valor de un parámetro de inicialización declarado en el fichero **web.xml**

```
<web-app>
  ...
  <context-param>
    <param-name>parameter1</param-name>
    <param-value>ValueOfParameter1</param-value>
  </context-param>
</web-app>
```

Objeto application

Uso en scriptlets

- Métodos del Objeto **application** :
 - **getInitParameterNames()** : devuelve un enumerado con los los nombres de los parámetros de incialización
 - **getRealPath(String value)** : convierte el path pasado como parámetro en un path del FileSystem
 - **log(String message)** : escribe un mensaje de Log en el fichero de Log asociado a la aplicación
 - **getServerInfo()** : devuelve el nombre y la versión del contenedor Servlet



JSP

Ejercicio

Objeto application

[application.jsp](#)

- Probar todos los métodos descriptor anteriormente sobre el Objeto **application**



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Objeto out

Uso en scriptlets

- El Objeto **out** es similar a **System.out** de Java, lo podemos utilizar para escribir en la salida estándar. Podemos usarlo tanto con **print** como con **println**

```
<%out.print(expression);%>
```

Objeto out

Uso en scriptlets

- Métodos del Objeto **out** :

- **print()** : imprime un valor en el navegador del cliente
- **println()** : lo mismo que **print** pero añade una nueva línea al final
- **newLine()** : añade una nueva línea
- **clear()** : limpia el buffer de salida
- **flush()** : lo mismo que **clear** pero antes vuelca el contenido de **out** en la salida (Será escrito todo escrito en el navegador del cliente)
- **isAutoFlush()** : devuelve **true** si el buffer se hace de manera automática
- **getBufferSize()** : devuelve el tamaño del buffer (bytes)
- **getRemaining()** : número de bytes disponibles hasta llegar al máximo del buffer

Objeto pagecontext

Uso en scriptlets

- El Objeto **pageContext** se puede usar para acceder a todos los objetos y atributos de una página Jsp. El ámbito de los objetos puede ser **PAGE_SCOPE, REQUEST_SCOPE, SESSION_SCOPE, APPLICATION_SCOPE**
- JSP Page – Scope: PAGE_CONTEXT
- HTTP Request – Scope: REQUEST_CONTEXT
- HTTP Session – Scope: SESSION_CONTEXT
- Application Level – Scope: APPLICATION_CONTEXT
- Cuando se busca algún atributo lo busca en los cuatro niveles de contexto siguiendo el orden marcado (PAGE,REQUEST,SESSION,APPLICATION)

```
pageContext.removeAttribute("attrName", PAGE_SCOPE)
```

```
pageContext.removeAttribute("attrName")
```

Objeto pagecontext

Uso en scriptlets

- Métodos del Objeto **pageContext** :

- **findAttribute(String AttributeName)** : devuelve el valor de un atributo
- **getAttribute(String AttributeName, int Scope)** : devuelve el valor de un atributo en un contexto concreto

```
Object obj = pageContext.getAttribute("BeginnersBook", PageContext.SESSION_CONTEXT);
```

- **removeAttribute(String AttributeName, int Scope)** : elimina un atributo del contexto especificado
- **setAttribute(String AttributeName, ObjectAttributeValue, int Scope)** : crea un atributo en el contexto especificado



JSP

Ejercicio

Objeto pageContext

pageContext.jsp

- Probar todos los métodos descriptor anteriormente sobre el Objeto **pageContext**



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Objeto session

Uso en scriptlets

- El Objeto **sesión** es uno de los más usados, nos permite acceder/gestionar toda la información referente a los datos del usuario mientras su sesión está activa
- Podemos usarla para parar/recuperar información entre distintas páginas Jsp dentro de la misma sesión

```
session.setAttribute("sessname",uname);
```

```
String name=(String)session.getAttribute("sessname");
```

Objeto session

Uso en scriptlets

- Métodos del Objeto **session** :

- **setAttribute(String, object)** : crea y guarda un Objeto en la sesión del usuario
- **getAttribute(String name)** : recupera un Objeto guardado en la sesión del usuario
- **removeAttribute(String name)** : elimina un Objeto de la sesión del usuario
- **getAttributeNames()** : devuelve un enumerado con los nombres de los Objetos guardados en la sesión del usuario
- **getCreationTime()** : devuelve la fecha/hora desde que la sesión está activa
- **getId()** : devuelve el Id de la sesión
- **isNew()** : devuelve **true** si la sesión es nueva
- **invalidate()** : elimina una sesión
- **getMaxInactiveInterval() y getLastAccessedTime()** : devuelve información sobre el tiempo iactivo y de acceso de la sesión

Objeto session

Uso en scriptlets

```

<html>
<head>
<title>Welcome Page: Enter your name</title>
</head>
<body>
<form action="session.jsp">
<input type="text" name="inputname">
<input type="submit" value="click here!!"><br/>
</form>
</body>
</html>

```

output.jsp

```

<html>
<head>
<title>Output page: Fetching the value from session</title>
</head>
<body>
<%
String name=(String)session.getAttribute("sessname");
out.print("Hello User: You have entered the name: "+name);
%>
</body>
</html>

```

session.jsp

```

<html>
<head>
<title>Passing the input value to a session variable</title>
</head>
<body>
<%
String uname=request.getParameter("inputname");
out.print("Welcome " + uname);
session.setAttribute("sessname",uname);
%>
<a href="output.jsp">Check Output Page Here </a>
</body>
</html>

```



JSP

Ejercicio

Objeto session

session.jsp

- Probar todos los métodos descriptor anteriormente sobre el Objeto **session**



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Directivas

Uso de Directivas

- Las directivas controlan cómo se va a realizar el procesamiento de una página JSP.
- Dan instrucciones al servidor de cómo se debe procesar la página
- Hay 3 tipos de directivas
 1. De página
 2. De inclusión de otra página
 3. De TagLib

Directivas

De página

- **isThreadSafe** : indica si la página se ejecuta en varios Hilos (Por defecto vale **true**)

```
<%@ page isThreadSafe="false"%>
```

- **buffer** : para especificar el tamaño del buffer

```
<%@ page buffer="5kb"%>
```

- **extends** : para heredar de una clase concreta

```
<%@ page extends="mypackage.SampleClass"%>
```

Directivas

De página

- **info** : para dar una descripción de la página y poder recuperarla con `getServletInfo()`

```
<%@ page info="This code is given by Chaitanya Singh"%>
```

- **language** : para especificar el lenguaje usado en los scriplets

```
<%@ page language="java"%>
```

- **autoFlush** : indica que el buffer debería ser volcado/procesado cuando esté lleno

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

- **isELIgnored** : indica si las expresiones serán evaluadas o no

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



JSP

Contenido dinámico

Directivas

De inclusión de otra página

- **Include** : para incluir el contenido de una página en otra

```
<%@include file="myJSP.jsp"%>
```



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro





JSP

Contenido dinámico

Directivas

De TagLib (Etiquetas personalizadas)

- **taglib:** para incluir etiquetas personalizadas

```
<%@ taglib uri="http://www.sample.com/mycustomlib" prefix="demotag" %>
```



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro





JSP

Ejercicio

Directiva include

holaMundo.jsp

- Crear dos páginas más llamadas **holaMundo2.jsp** y **holaMundo3.jsp** e incluirlas en **holaMundo.jsp**



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Etiquetas de acción

Action Tags

- Son evaluadas cada vez que se carga la página Jsp
- **<jsp:include>** : inserta el código de una página Jsp en otra

```
<jsp:include page="sample.jsp" flush="false" />
```

- **<jsp:forward>** : redirige a otra página

```
<jsp:forward page="second.jsp" />
```

- **<jsp:param>** : para pasar parámetros de una página a otra

```
<jsp:forward page="second.jsp">
    <jsp:param name ="date" value="20-05-2012" />
    <jsp:param name ="time" value="10:15AM" />
    <jsp:param name ="data" value="ABC" />
</jsp:forward>
```

```
Date:<%= request.getParameter("date") %>
Time:<%= request.getParameter("time") %>
My Data:<%= request.getParameter("data") %>
```

Etiquetas de acción

Action Tags

- **<jsp:useBean>** : para usar Beans dentro de una página, una vez instanciado con **jsp:useBean** para acceder/setear las variables de la clase hay que usar **jsp:getProperty** y **jsp:setProperty**

```
<jsp:useBean id="student" class="javabeansample.StuBean"/>  
<jsp:setProperty name="student" property="*"/>  
  
name:<jsp:getProperty name="student" property="name"/><br>  
empno:<jsp:getProperty name="student" property="rollno"/><br>
```

EL : expression language

`${ expresión }`

- Podemos usarlo para acceder a cualquiera de los Objetos vistos de una manera más simple y poder mostrar la información más fácilmente. Para poder usarlo, añadir en la cabecera de la página JSP `<%@ page isELIgnored="false" %>`

pageScope

requestScope

sessionScope

applicationScope

param

paramValues

header

headerValues

cookie

initParam

pageContext

```
<form action="process.jsp">  
Enter Name:<input type="text" name="name" /><br/><br/>  
<input type="submit" value="go"/>  
</form>
```

`${ param.name }`

```
<%  
session.setAttribute("user","sonoo");  
%>
```

`${ sessionScope.user }`



JSP

Ejercicio

EL : expression language

`${ expresión }`

- Modificar el ejercicio de params y mostrar los parámetros usado EL



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



**Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil**
El FSE invierte en tu futuro

