Tutorial2ASIR_JSP

Índice de Contenidos:

Manuel Martínez Molina y Jose Gabriel Esparza Sánchez

1. ¿Qué es JSP?

JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

La principal ventaja de **JSP** frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolija. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

Otra ventaja es que JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios. Es común incluso que los desarrolladores trabajen en una plataforma y que la aplicación termine siendo ejecutada en otra.

Requisitos para programar JSP.

Para aprender JSP, aparte de conocer HTML, será necesario comprender y tener algo de experiencia en la programación en Java, que es un lenguaje de programación Orientado a Objetos por completo.

Ejemplo JSP.

```
HTML
BODY
   <%
      String user = request.getParameter("user");
      String pass = request.getParameter("password");
      if ("grupo5".equals(user) && "hola".equals(pass)) {
            out.println("login ok");
         } else {
            out.println("invalid login");
         }
    %>
/BODY
/HTML
```

2. Etiquetas JSP.

Son las etiquetas pertenecientes a la especificación JSP. Proporcionan una funcionalidad básica.

Un primer grupo de etiquetas proporciona funcionalidad a nivel de la página de una manera muy simple:

- **<jsp:forward>**, redirige la request a otra URL
- **<jsp:include>**, incluye el texto de un fichero dentro de la página
- **<jsp:plugin>**, descarga un plugin de Java (una applet o un Bean).

Un segundo grupo permite manipular componentes JavaBean sin conocimientos de Java.

- **<jsp:useBean>**, permite manipular un Bean (si no existe, se creará el Bean), especificando su ámbito (scope), la clase y el tipo.
- **<jsp:getProperty>**, obtiene la propiedad especificada de un bean previamente declarado y la escribe en el objeto response.
- **<jsp:setProperty>**, establece el valor de una propiedad de un bean previamente declarado.

3. Sesiones.

Una sesión es una serie de comunicaciones entre un cliente y un servidor en la que se realiza un intercambio de información. Por medio de una sesión se puede hacer un seguimiento de un usuario a través de la aplicación.

El tiempo de vida de una sesión comienza cuando un usuario se conecta por primera vez a un sitio web pero su finalización puede estar relacionada con tres circunstancias:

- Cuando se abandona el sitio web.
- Cuando se ha cerrado o reiniciado el servidor.
- Cuando se alcanza un tiempo de inactividad que es previamente establecido, en este caso la sesión es automáticamente eliminada. Si el usuario siguiera navegando se crearía una nueva sesión.

Manejo de sesiones.

Para obtener la sesión de un usuario se utiliza el método **getSession()** que devuelve una interfaz de tipo **HttpSession**.

```
<%
   HttpSession sesion=request.getSession();
%>
```

Una vez que tenemos los datos de sesión podemos acceder a los datos de esta, por ejemplio al ID de sesión

```
<%
   HttpSession sesion=request.getSession();
   out.println("Sesion ID:"+ sesion.getId());
%>
```

Guardar objetos en una sesión.

Para introducir objetos en un sesión utilizando el método **setAttribute()**

```
SetAttribute(java.lang.String name, java.lang.Object value);
```

- El primer parametro indicará el nombre que va a identificar al valor.
- El segundo parametro indicará el valor que vamos a introducir.

Ejemplo:

```
<%
   HttpSession sesion=request.getSession();
   String minombre = "grupo5";
   sesion.setAttribute("id", minombre);
%>
```

Hay que tener en cuenta que si es un valor primitivo no se puede transformar por lo que hay que machacar el dato.

Obtener objetos en una sesión.

Para recuperar un objeto en una sesion usamos el método **getAttribute()**

```
getAttribute(java.lang,String nombre)
```

Con el siguiente ejemplo veremos como extraer un objeto de la sesión.

```
<%
   HttpSession sesion=request.getSession();
   out.println(sesion.getAttribute("id"));
%>
```

Destruir una sesión.

Como hemos visto arriba, el usuario puede invalidar la sesión fácilmente pero nosotros podemos forzar que una sesión sea destruida con el método **invalidate()**

```
<%
   HttpSession sesion = request.getSession();
   sesion.invalidate();
%>
```

4. Cookies.

Las cookies constituyen una potente herramienta empleada por los servidores Web para almacenar y recuperar información acerca de sus visitantes. Mediante el uso de cookies se permite al servidor Web recordar algunos datos concernientes al usuario, como sus preferencias para la visualización de las páginas de ese servidor, nombre y contraseña, productos que más le interesan, etc

Una cookie no es más que un fichero de texto que algunos servidores piden a nuestro navegador que escriba en nuestro disco duro, con información acerca de lo que hemos estado haciendo por sus páginas**.**

¿Cómo crear una cookie?

Un cookie almacenado en el ordenador de un usuario está compuesto por un nombre y un valor asociado al mismo. Además, asociada a este cookie pueden existir una serie de atributos que definen datos como su tiempo de vida, alcance, dominio, etc.
Para crear un objeto de tipo Cookie se utiliza el constructor de la clase Cookie que requiere su nombre y el valor a guardar. El siguiente ejemplo crearía un objeto Cookie que contiene el nombre “nombre” y el valor “objetos” (String).

```
<%
Cookie miCookie=new Cookie(“nombre”,”objetos”);
%>
```

También se pueden guardar valores o datos que provengan de páginas anteriores y que hayan sido introducidas a través de un formulario:

```
<%
Cookie miCookie=null;
String ciudad= request.getParameter(“formCiudad”);
miCookie=new Cookie(“ciudadFavorita”,ciudad);
%>
```

Una vez que se ha creado un cookie, es necesario establecer una serie de atributos para poder ser utilizado. El primero de esos atributos es el que se conoce como tiempo de vida. Ejemplo si queremos recordar el login:

```
if(rec!=null)
{
        miCookie=new Cookie("login",login);
        miCookie.setMaxAge(60*60*24*31);
        miCookie.setPath("/");
        response.addCookie(miCookie);
}
```

¿Cómo recuperar las cookies?

```
<%
                Cookie [] todosLosCookies=request.getCookies();
                Cookie unCookie=null;
                /* El siguiente paso es crear un bucle que vaya leyendo todos los cookies.
*/
                for(int i=0;i<todosLosCookies.length;i++)
                {
                Cookie unCookie=todosLosCookies[i];
                if(unCookie.getName().equals("nombre"))
                break;
                }
                out.println("Nombre: "+unCookie.getName()+"<BR>");
                out.println("Valor: "+unCookie.getValue()+"<BR>");
                out.println("Path: "+unCookie.getPath()+"<BR>");
                out.println("Tiempo de vida:"+unCookie.getMaxAge()+"<BR>");
                out.println("Dominio: "+unCookie.getDomain()+"<BR>");
                %>
```

También podemos hacerlo así:

```
String cookie=new String();
int encontrado=-1;
String log=new String();
Cookie [] todosLosCookies=request.getCookies();
Cookie unCookie=null;
for(int i=0;(i<todosLosCookies.length) && (encontrado==-1);i++)
{
unCookie=todosLosCookies[i];
if(unCookie.getName().equals("login"))
{
encontrado=i;
}
}
if(encontrado!=-1)
{
cookie=unCookie.getValue();
}
```

5. Formularios.

Los formularios son una de las herramientas de que disponemos a la hora de hacer nuestras páginas Web interactivas, en el sentido de que nos permiten recopilar información de la persona que ve la página, procesarla y responder a ella, pudiendo de esta forma responder adecuadamente a sus acciones o peticiones.

Las etiquetas que definen un formulario son:

<form action="elemento_destino" method="metodo">…</form>

El contenido del formulario viene delimitado por las etiquetas <form> y </form>.

Los parámetros son:

- **action="elemento_destino": donde elemento_destino es la dirección del script o CGI que va a recibir los parámetros que se introduzcan en el formulario.**
- **method="método": mediante este parámetro especificamos el método mediante el cual se mandan los datos del formulario al script o CGI. Existen 2 métodos:**

GET: el conjunto de datos del formulario se agrega al URL especificado por el atributo action (con un signo de interrogación ("?") como separador) y este nuevo URL se envía al agente procesador.

POST: con el método HTTP post, el conjunto de datos del formulario se incluye en el cuerpo del formulario y se envía al agente procesador.

6. Request.

Request es un objeto de la clase HttpServletRequest. Su principal uso principal es el acceso a los parámetros de la petición.

Destacan los métodos:

getParameter()

Obtiene un parámetro enviado por una petición get o post.

String variable = request.getParameter("valor");

getParameterValues()

Igual que el anterior, pero recibe un array de valores.

String [] variables = request.getParameterValues("vector");

request.getAttribute()

Recoge atributos enviados con el metodo setAttribute.

String l = request.getAttribute("login")

request.getSession()

Tomar la sesion del usuario y la almacenarlen una variable de tipo HttpSession.

HttpSession repositorio = request.getSession();