Integración de Tecnologías
Grado en Ingeniería Informática en Sistemas de Información
EPD 2: Objetos implícitos y acciones en JSP

Objetivos

- Aprender a utilizar los objetos implícitos de JSP.
- Aprender a utilizar acciones de JSP.

Conceptos

1. Objetos implícitos

JSP nos ofrece una serie de objetos que se construyen de manera automática en el *servlet* generado a partir del JSP. Por tanto, no necesitamos instanciar ninguna clase para generarlos, ni declarar las referencias necesarias para ellos.

Los objetos predefinidos en JSP que se utilizarán en esta práctica se muestran en la Tabla 1.

Objeto	¿Clase o Interfaz?	Clase/Intefaz	Funcionalidad
out [1]	Implementa **clase**	javax.servlet.jsp.JspWriter	Objeto que escribe en el flujo de salida usado para enviar la respuesta al cliente. La salida de los JSP emplea un *buffer* que permite que se envíen cabeceras HTTP o códigos de estado, aunque ya se haya empezado a escribir en la salida (`out` no es un `PrintWriter` sino un objeto de la clase especial `JspWriter`).
request [2]	Instancia **interfaz**	javax.servlet.http.HttpServletRequest	Objeto asociado con la petición del cliente. Usado para capturar valores procedentes de formularios o de URLs.
response [3]	Instancia **interfaz**	javax.servlet.http.HttpServletResponse	Objeto asociado con la respuesta al cliente.
Session [4]	Instancia **interfaz**	javax.servlet.http.HttpSession	Objeto asociado con la sesión del cliente. En JSP, las sesiones se crean automáticamente, de modo que este objeto está instanciado, aunque no se cree explícitamente una sesión.
application [5]	Instancia **interfaz**	javax.servlet.ServletContext	Objeto que representa el contexto de la aplicación de la que forma parte. Común a todos los *servlets* de la aplicación web.
config [6]	Instancia **interfaz**	Javax.servlet.ServletConfig	Objeto empleado para leer parámetros de inicialización.

Tabla 1: Objetos implícitos de JSP utilizados en esta práctica.

Los objetos implícitos se pueden utilizar directamente dentro de un *scriptlet* de código Java, por ejemplo:

```
<%
        String strParam = request.getParameter("nombre_del_parametro");
        out.println( strParam );
%>
```

2. Acciones

Las **acciones** JSP usan construcciones de sintaxis XML que sirven para invocar determinados comportamientos en nuestra página. Por ejemplo, podemos insertar un fichero dinámicamente, reutilizar componentes *JavaBeans*, reenviar al usuario a otra página, o generar el código HTML necesario para incrustar un *plug-in* Java (un applet). Para más detalles sobre las acciones véase [7] cap. 8, pág. 241-274. Las acciones que se utilizarán en esta práctica son:

- **`jsp:include`** Esta acción nos permite insertar el código contenido en un fichero en una página que está **siendo generada**. Es menos eficiente que la directiva *include* ya que esta directiva inserta el código contenido en el fichero en el momento de la conversión de la página JSP a un *servlet* mientras que la acción lo inserta dinámicamente cada vez que la página es solicitada. No obstante, este mecanismo de inclusión permite incluir ficheros cuyo contenido cambie frecuentemente. Aunque esta acción imposibilita a la página incluida contener código JSP general (no puede seleccionar cabeceras HTTP, por ejemplo), proporciona una significativa flexibilidad. Un ejemplo de uso es el siguiente:

```
<jsp:include page="relative URL" flush="true" />
```

- **`jsp:forward`** Esta acción nos permite reenviar la petición a otra página. Tiene un sólo atributo, `page`, que debería consistir en una *URL* relativa. Este podría ser un valor estático, o podría ser calculado en el momento de la petición, como en estos dos ejemplos:

```
<jsp:forward page="/utils/errorReporter.jsp" />
<jsp:forward page="<%= someJavaExpression %>" />
```

En el primer ejemplo la URL indica que la página a mostrar será errorReporter.jsp, la cual estará contenida en la carpeta *utils* del proyecto.

Cuando reenviamos una petición, también podemos pasarle nuevos parámetros a dicha petición de la siguiente forma:

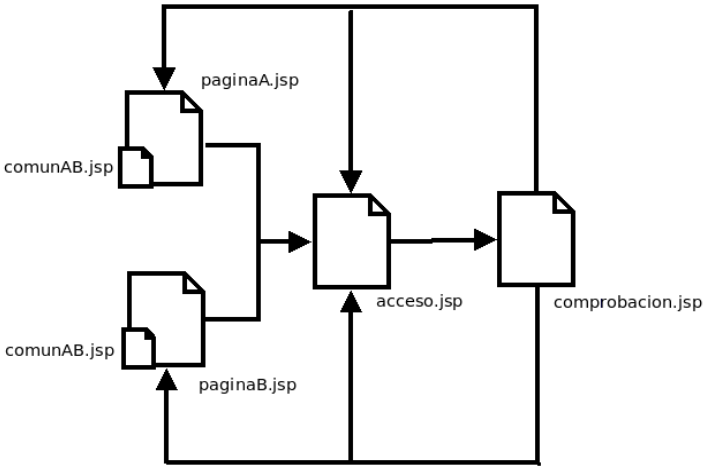
```
<jsp:forward page="pagina.jsp">
   <jsp:param name="usuario" value="Tom"/>
</jsp:forward>
```

Experimentos

Ex1 (30 mins). En este experimento haremos uso de directivas, objetos implícitos y acciones para implementar la restricción del acceso a una página, de tal forma que sólo se permita el acceso a usuarios registrados.

Las páginas a las que el usuario podrá acceder si se registra serán *paginaA.jsp* y *paginaB.jsp*. El código común de ambas está en el fichero *comunAB.jsp*. Este código común es el encargado de comprobar si el usuario ha iniciado o no sesión. En caso afirmativo, accede al contenido de la página, que en el ejemplo es un saludo dirigido al usuario identificando la página a la que se ha accedido. En caso negativo reenvía (realiza un forward) al usuario a la página *acceso.jsp* donde podrá introducir el nombre de usuario y la contraseña.

La labor de comprobar si los parámetros introducidos son o no correctos son delegados a *comprobacion.jsp*. En caso de que los datos sean correctos reenvía (realiza un forward) a la página inicial (*paginaA.jsp* o *paginaB.jsp*) y en caso contrario a la de acceso. Por lo tanto, las páginas a ejecutar primero serán *paginaA.jsp* o *paginaB.jsp.* A continuación, podemos ver el esquema:



El código de cada uno de los ficheros es el siguiente:

1. *paginaA.jsp*

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="comunAB.jsp" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
       <h1>Buenos dias desde la pagina <%=request.getRequestURL()%>,
            <%=sesUsuario%>
        </h1>
    </body>
</html>
```

2. *paginaB.jsp*

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@ include file="comunAB.jsp" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
       <h1>Buenas tardes desde la pagina <%=request.getRequestURL()%>,
            <%=sesUsuario%>
        </h1>
    </body>
</html>
```

3. *comunAB.jsp*

```
<%
    String sesUsuario = (String) session.getAttribute("usuario");
    if (sesUsuario == null) {
        String cp = request.getContextPath();
        String rp = request.getRequestURI();
        rp = rp.replace(cp, "");
       //definimos en la sesion el atributo fuente
        session.setAttribute("fuente", rp);
%>
       <jsp:forward page="acceso.jsp"/>
<%
    }
%>
```

4. acceso.jsp

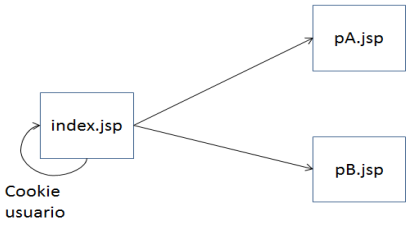
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            String fallo = (String) session.getAttribute("fallo");
            if (fallo != null) {
                session.removeAttribute("fallo");
        %>
                Nombre de usuario o clave incorrectos.
        <%
            }
        %>
        <h1>Introduzca su nombre de usuario y clave:</h1>
        <form action="comprobacion.jsp" method="post">
            <table border="1" cellspacing="2" cellpadding="2">
                <tr> <td>Usuario</td> <td><input type="text" name="usuario"></td> </tr>
                <tr> <td>Clave</td> <td><input type="password" name="clave"></td> </tr>
                <tr><td><input type="submit" value="Enviar" size="2"></td></tr>
            </table>
        </form>
    </body>
</html>
```

5. comprobacion.jsp

```
<%
    String usuario = request.getParameter("usuario");
    String clave = request.getParameter("clave");
    if (usuario.equals(clave)) {
        session.setAttribute("usuario", usuario);
        String fuente = (String) session.getAttribute("fuente");
        session.removeAttribute("fuente");
%>
        <jsp:forward page="<%=fuente%>"/>;
<%
    } else {
        session.setAttribute("fallo", "si");
%>
        <jsp:forward page="acceso.jsp"/>;
<%
    }
%>
```

a) El contenido del fichero *comunAB.jsp* en *paginaA.jsp* y *paginaB.jsp,* ¿se incluye en el momento de la petición?
b) Entre en las páginas *paginaA.jsp* y *paginaB.jsp*. ¿A qué pagina dirige? ¿Reenvía (forward) o redirige (redirect)?
c) En un navegador, explore dentro de la opción *inspeccionar elemento* el apartado *red*. ¿Por cuántas páginas distintas se pasa? ¿Los saltos entre páginas se realizan desde el servidor o desde el navegador del usuario?
d) Introduzca un nombre de usuario y clave diferentes entre si. ¿Qué pasa?
e) Introduzca ahora un nombre de usuario y una clave que sean iguales (en el experimento esta condición simula que los valores introducidos sean correctos). ¿A qué página nos dirige? ¿Reenvía (forward) o redirige (redirect)?
f) Modifique los saltos entre páginas de forma que, si antes se hacía un reenvío ahora se haga una redirección o viceversa. Acceda a las páginas *paginaA.jsp* y *paginaB.jsp* y analice los cambios que se producen.
g) En un navegador, explore dentro de la opción *inspeccionar elemento* el apartado *red*. ¿Por cuántas páginas distintas se pasa? ¿Los saltos entre páginas se realizan desde el servidor o desde el navegador del usuario?
h) ¿Por qué no funciona el salto que debería volver a *paginaA.jsp* o *paginaB.jsp*? Busque el posible fallo y soluciónelo.

Ex2 (10 mins). Con ayuda del material extra que se proporciona, cree una página JSP con un formulario donde el usuario pueda insertar su nombre y elegir a qué página quiere ser ir. (¿Que deberá usar, reenvió o redirección?) Se creará una cookie con el nombre del usuario que se mostrará en la página a la que se llegue. El esquema de la aplicación será como se muestra a continuación:



Bibliografía básica

1. http://java.sun.com/javaee/5/docs/api/javax/servlet/jsp/JspWriter.html
2. http://java.sun.com/javaee/5/docs/api/javax/servlet/http/HttpServletRequest.html
3. http://java.sun.com/javaee/5/docs/api/javax/servlet/http/HttpServletResponse.html
4. http://java.sun.com/javaee/5/docs/api/javax/servlet/http/HttpSession.html
5. http://java.sun.com/javaee/5/docs/api/javax/servlet/ServletContext.html
6. http://java.sun.com/javaee/5/docs/api/javax/servlet/ServletConfig.html
7. Beginning JavaServer Pages. Vivek Chopra, Sing Li, Rupert Jones, Jon Eaves y John T. Bell. Wiley, c2005. http://site.ebrary.com/lib/bupo/detail.action?docID=10114249

Ejercicios

EJ1 (20 mins.) Cree dos páginas denominadas index.jsp y resultado.jsp. En la página index.jsp implemente un formulario que solicite al usuario la siguiente información:

- Email
- Nombre de usuario
- Contraseña

Cuando el usuario haya rellenado el formulario pulsará un botón y se enviará la información a la página *resultado.jsp* empleando el método *post*. Utilizando los objetos `out` y `request`, imprimir, en una tabla en *resultado.jsp*, tanto los valores devueltos por el formulario, como cada uno de los métodos del objeto `request` vistos en clase. Cada uno de los métodos debe de ir acompañado de una breve explicación de lo que muestra. Ejemplo:

```
<td> getRequestedSessionId devuelve el identificador de la sesión del cliente: </td>
<td><%=request.getRequestedSessionId() %></td>
```

EJ2 (10 mins.) Implementar una página HTML que visualice una lista desplegable con las opciones que se muestran a continuación:

`Elija la página de referencia a objetos implícitos que desee visualizar:`

- `request`
- `out`
- `response`
- `config`
- `application`
- `session`

Implementar también un archivo JSP que dependiendo de la opción seleccionada por el usuario redireccione, utilizando los objetos implícitos, el navegador a las páginas de referencia oficial de cada objeto (del [1] al [6] en la bibliografía básica de este guión de EPD).

EJ3 (30 mins.) Implementar un archivo JSP que imprima, de la sesión actual, la siguiente información: ID de la sesión, fecha de creación (usar la clase *Date* para formatear la salida), y duración de la sesión. Además, esta página ofrecerá

un botón que permitirá invalidar la sesión al ser pulsado, tras lo que se volverá a la página original para mostrar los datos de la nueva sesión.

Problemas

P1 (25 mins) Implementar una página JSP de noticias. Cada noticia estará contenida en un archivo de texto. La página debe contar con al menos 3 noticias. En principio, deben mostrarse todas las noticias, de tal forma que haya que incluir dichos archivos en la página. Cada noticia, además, contará con un botón de tipo radio, con dos opciones: ocultar y mostrar la noticia. La página contará con un botón al final del todo, y dependiendo de lo elegido en los botones de radio, se volverá a mostrar la página con la o las respectivas noticias ocultas o mostrándose. Considerar como último requisito, que cuando se oculte el contenido de una noticia, se muestre un mensaje del tipo "información no mostrada"

P2 (15 mins) Modificar el ejercicio 2 utilizando una acción para reenviar a otra página JSP creada por usted. Una vez reenviado a dicha página, ésta deberá redireccionar automáticamente a la página seleccionada por el usuario.

P3 (40 mins.) Implemente una aplicación web que ayude a distinguir el sexo de sus usuarios. Para entrar en la aplicación se requerirá un registro (*index.jsp*), de forma que, si se viene de primeras a ella, mostrará un formulario que pedirá información del usuario (p.ej.: nombre, apellidos y sexo, siendo sexo un botón de radio a elegir entre hombre y mujer). Al recibir esa información se deberá de crear una *cookie* por cada uno de esos campos. Esas mismas *cookies* servirán para conocer si el usuario ya se registró (si las tiene todas y con valores válidos, se entiende que está registrado). En ese caso, al acceder a la página *index.jsp*, no se mostrará el formulario de registro, si no una página de bienvenida. Esta página de bienvenida, además, cambiará en función del sexo que haya elegido el usuario (una para las mujeres, *mujer.jsp*, y otra para los hombres, *hombre.jsp*) y mostrará la información contenida en la *cookie*. Añada a esas dos páginas, *hombre.jsp* y *mujer.jsp* la opción de deslogarse, para ello cada página tendrá un botón que, al pulsar el usuario, caducará la cookie y mostrará de nuevo la página del formlulario (*index.jsp*)

P4 (20 mins.) Realizar una página JSP de venta por internet, parecida a Amazon, que ofrezca diferentes productos (Libros, electrónica, videojuegos, etc) Por cuestiones organizativas, se desea que la información de cada producto se encuentre almacenada en páginas html independientes, de forma que se incluya la información de cada uno de estos en la página JSP de manera estática.

P5 (30 mins.) Implementar una página que examine todas las cabeceras que envía el navegador y que reenvíe al usuario a una página distinta en función del navegador que use. Para ello, haga uso de la información enviada en la cabecera en *user-agent*. Deberá implementar una página para cada navegador (p. ej. Chrome y Firefox) que contenga un mensaje en el que se indique el navegador que está usando. Finalmente, implemente otra página con un mensaje de error para el caso en que el usuario use un navegador no soportado. Aproveche la cabecera *Accept-Language* para comprobar si aceptará inglés, mostrando el mensaje en ese caso en dicho idioma.

P6 (20 mins.) Modificar la página JSP del problema 3 para que en lugar de emplear cookies, se haga uso de la sesión del usuario para almacenar los datos. (En este caso, para la opción de deslogarse, se invalidará la sesión y se mostrará la página principal *index.jsp*)

Ampliación de la bibliografía

1. JavaServer Pages. Bergsten, Hans. O'Reilly, 2001.
2. JavaServer Pages: Manual de usuario y tutorial. Froufe Quintas, Agustín. Ra-ma, 2002.
3. JavaServer Pages: pocket reference. Bergsten, Hans. O'Reilly, 2001.