Laboratorio de Informática Distribuida

Sesión III. JavaServer Pages

Introducción a JavaServer Pages

- Índice Sesión JSP:
  - Tecnologías JAVA para aplicaciones web.
  - Introducción a JSP.
  - Elementos JSP.
  - Ciclo de Vida.
  - Acceso a Datos.
  - Manejo de Sesiones.
  - JavaBeans.
  - Librería de Etiquetas.

Introducción a JavaServer Pages

- Tecnologías JAVA para aplicaciones web.
  - El lenguaje JAVA dispone de **dos** tecnologías para el desarrollo de aplicaciones en entorno web:
    - Tecnología JSP.
    - Tecnología Servlets.
  - Aunque distintas, están íntimamente relacionadas, ya las páginas JSP al final se transforman en Servlets que son las aplicaciones que realmente se ejecutan en el servidor web.

Introducción a JavaServer Pages

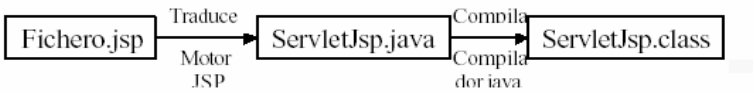
- Introducción a JSP.
  - Permite mezclar, en una página, código HTML para generar la parte estática, con contenido dinámico generado a partir de marcas especiales **<%** .... **%>**
  - El contenido **dinámico** se obtiene, en esencia, gracias a la posibilidad de **incrustar** dentro de la página **código Java** de diferentes formas.
  - Permite el acceso a Bases de Datos remotas.
  - Gestionar sesiones.

Introducción a JavaServer Pages

- Introducción a JSP.
  - El primer borrador de la especificación JSP, de Sun Microsystems, vio la luz en 1998, apareciendo la especificación v 1.0 al año siguiente.
  - La siguiente versión (v 1.1) se presento a finales de 1999.
  - Actualmente disponemos de la especificación v 1.2

Introducción a JavaServer Pages

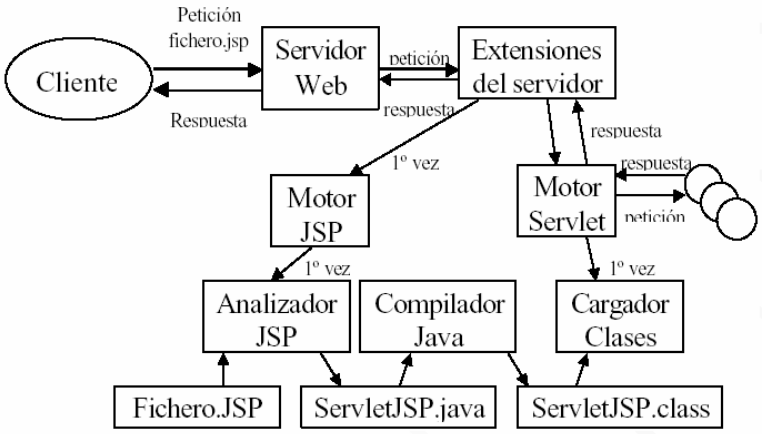
- Introducción a JSP.



  - La página JSP **se convierte en un Servlet**
  - La conversión la realiza la máquina servidora: el *motor o contenedor de JSP*, la primera vez que se solicita la página JSP al servidor web.
  - Este servlet generado procesa cualquier petición para esa página JSP.
  - Si se modifica el código de la página JSP, entonces se **regenera** y **recompila** automáticamente el Servlet y se recarga la próxima vez que sea solicitada.

Introducción a JavaServer Pages

- Introducción a JSP.



Introducción a JavaServer Pages

- Introducción a JSP.
  - Ejemplo de página JSP que dice Hola y escribe la fecha actual (**fichero hola.jsp**). Para editar las páginas se utilizará el EditPlus.

```
<%@ page info="Un ejemplo Hola Mundo" import="java.util.Date" %>
        <HTML>
        <head> <title> Hola, Mundo </title> </head>
        <body> <h1> ¡Hola, Mundo! </h1>
        La fecha de hoy es: <%= new Date().toString() %>
        </body>
        </HTML>
```

Introducción a JavaServer Pages

- Introducción a JSP.
  - **/ *.html *.jsp *.css**: Este directorio base contiene los elementos que comúnmente son utilizados en un sitio Web, Documentos en HTML , JSP's , CSS ,etc.
  - **/WEB-INF/web.xml**: Contiene elementos de seguridad de la aplicación así como detalles sobre los servlets que serán utilizados dentro de la aplicación.
  - **/WEB-INF/classes/**: Contiene a los servlets y a las clases Java adicionales a las del JDK que serán empleadas.
  - **/WEB-INF/lib/**: Contiene los JAR's que serán utilizados por la aplicación.

Directorio de Partida
C:\Tomcat5.0\webapps\MiAplicación

Introducción a JavaServer Pages

- Elementos JSP. Resumen.
  - **Comentarios.**
  - **Directivas <%@**
    - page **<%@ page ATRIBUTOS %>**
    - include **<%@ include file="Nombre del fichero" %>**
    - taglib **<%@ taglib uri ="taglibraryURI" prefix="tagPrefix" %>**
  - **Elementos para incrustar código JAVA**
    - **Scripting (guiones) <%**
    - **Declaraciones <% ! Declaración %>**
    - **Código Java arbitrario (scriptlets) <% código Java %>**
    - **Expresiones <%= Expresión Java a evaluar %>**
  - **Acciones** estándar *(formato XML) <jsp:*
    - <jsp:useBean> <jsp:setProperty> <jsp:getProperty>
    - <jsp:include> <jsp:param>
    - <jsp:forward> <jsp:param>

Introducción a JavaServer Pages

- Elementos JSP.
  - Comentarios.
    - Las paginas JSP admiten dos tipos de comentarios, los que están orientados o concebidos para presentarse en el código HTML que se generaran a partir de la pagina JSP y los comentarios ocultos, que solamente aparecen en la pagina JSP y están orientados al desarrollador.

```
<!-- Comentario HTML normal -->
<%-- Comentario oculto JSP --%>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Declaraciones:
    - Son elementos que se utilizan para declarar una variable, objeto o un método que se utilizará posteriormente en la pagina JSP. No generan ninguna salida, por lo que pueden utilizarse conjuntamente con otros elementos de las paginas JSP, como son expresiones o scriptlets.

```
<%! Declaracion, .... % >

<%! int i = 0, j = 10; % >
<%! java.util.Date fecha = new java.util.Date();%>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Expresiones:
    - Las expresiones son un mecanismo que evita tener que escribir el código completo de la sentencia **`out.println().`** Su sintaxis utiliza el símbolo de igual, tal y como vemos a continuación

```
<%= expresion >

<%=  sueldoBase + dietas %>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Scriptlets:
    - Un scriptlet es un bloque de código Java insertado en la página y ejecutado durante el procesamiento de la respuesta.
    - El código introducido se inserta directamente en el método _jspService() del servlet generado para la página.

```
<% código Java %>

    <% int i, j;
        for (i=0;i<3;i++) {
         j=j+1;
         }
    %>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Directivas:
    - Utilizadas para definir y manipular una serie de atributos dependientes de la página que afectan a todo el JSP.
    - Las directivas existentes son las siguientes:
      - Page
      - Include
      - Taglib

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *page*:
    - Esta directiva se utiliza para definir **atributos globales** que deben ser aplicados a la pagina JSP completa.
    - La directiva *page* se puede utilizar varias veces en una misma pagina JSP, aunque solamente se puede especificar el valor de un atributo una única vez. La excepción de esta regla es el atributo ***import*** al cual se le pueden asignar diferentes valores.

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *page*:
    - Sintaxis:

```
<%@ page ATRIBUTOS %>
```

    - Donde ATRIBUTOS aparecen en pares: nombre=”valor”
    - Ejemplo:

  - A continuación se presenta una lista de los atributos más utilizados.

```
<%@ page language=”Java” import=”Java.rmi.*, java.util.*” session=”true” %>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *page*:
    - language.
      - indica el lenguaje que se utiliza en la página (scriptles). Con la especificación 1.2, sólo está permitido el valor “java”.
    - import.
      - Lista de paquetes o clases, separados por coma, que serán **importados** para utilizarse dentro del código java.
    - session.
      - Especifica si la página participa en una sesión. Si se inicializa a true, está disponible el objeto implícito sesión.
    - buffer.
      - Especifica el tamaño de un “espacio” para manejar la salida de la página jsp al cliente.
    - autoflush, info, errorPage, isErrorPage,….

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *include*:
    - Indica al motor JSP que incluya el contenido de un fichero a una página JSP, insertándolo en el lugar de la directiva include JSP.
    - El contenido del fichero incluido, es analizado en el momento de la traducción del fichero JSP y se incluye una copia del mismo dentro del servlet generado.
    - Una vez incluido, si se modifica el fichero incluido **no** se verá reflejado en el servlet.

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *include*:
    - El tipo de fichero a incluir puede ser un
      - fichero HTML (estático)
      - fichero jsp (dinámico)
    - Sintaxis

```
<%@ include file="URL  fichero" %>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *include*: Ejemplo: directiva.jsp

```
<HTML>
<head>
<title> Página de prueba de directivas de compilación
</title>
</head>
<body>
<h1> Página de prueba de directivas de
compilación </h1>
<%@ include file=”/fich1.html” %>
<%@ include file=”/fich2.jsp” %>
</body>
</HTML>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *include*: Ejemplo: fich1.html

```
<HTML>
<head> <title> Hola, Mundo </title> </head>
<body> <h1> ¡Hola, Mundo! </h1>
</body>
</HTML>
```

fich2.jsp

```
<%@ page info="Un ejemplo Hola Mundo"
import="java.util.Date" %>
La fecha de hoy es: <%= new Date().toString() %>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *taglib*:
    - Permite extender los marcadores de JSP con etiquetas o marcas generadas por el propio usuario (etiquetas personalizadas).
    - Se hace referencia a una biblioteca de etiquetas que contiene código Java compilado definiendo las etiquetas que van a ser usadas, y que han sido definidas por el usuario.

Introducción a JavaServer Pages

- Elementos JSP.
  - Directiva *taglib*:
    - Sintaxis

```
<%@ taglib uri="WEF-INF/tlds/libreria.tld" prefix="lib" %>
```

    - Los atributos son:
      - uri: la dirección donde se sitúa el fichero descriptor de la librería de etiquetas (Tag Library Description - TLD) en formato XML.
      - prefix: Define el prefijo de una acción de la librería.

Introducción a JavaServer Pages

- Elementos JSP.
  - Acciones:
    - Son marcas-etiquetas **estándar**, en formato XML, que afectan al comportamiento en **tiempo de ejecución** de la página JSP.
    - En la traducción de la página JSP al servlet, la marca se reemplaza por cierto código Java que define a dicha marca. Una marca, por tanto, asocia cierto código Java.
    - Constan de un prefijo y un sufijo además de una serie de atributos. El prefijo es siempre **"jsp"** en las acciones **estándar**.

Introducción a JavaServer Pages

- Elementos JSP.
  - Acciones.
    - Sintaxis:

      <jsp:sufijo atributos/>

  - Lista de Acciones:
    - <jsp:include>
    - <jsp:forward>
    - <jsp:param>
    - <jsp:useBean>
    - <jsp:setProperty>
    - <jsp:getProperty>

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:include>
    - Permite incluir un recurso especificado por la URL, en la petición JSP en **tiempo de ejecución**.
    - Por tanto, cuando se **ejecuta el servlet**, se invoca al recurso que realiza la operación y devuelve el resultado al propio servlet.
    - Sintaxis

```
<jsp:include page="URL"/>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:include>
    - ¿Qué diferencia existe entre la directiva y la acción?
    - **Directiva:** <%@ include file="Nombre fichero" %> se añade el código al servlet que se genera para la página en **tiempo de compilación** y se incluye el contenido EXISTENTE EN EL MOMENTO INICIAL.
    - **Acción <jsp:include>** no se añade código al servlet, sino que se invoca al objeto en **tiempo de ejecución** y se ejecuta el contenido EXISTENTE EN EL MOMENTO DE LA PETICIÓN.

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:param>
    - Se usa como submarca dentro de cualquier otra acción.
    - Sirve para pasar parámetros.
    - Sintaxis

```
<jsp:include page="URL">
<jsp:param name="nombre clave" value="valor"/>
</jsp:include>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:forward>
    - Esta marca permite que la petición sea redirigida a otra página JSP, a otro servlet o a otro recurso estático.
    - Muy útil cuando se quiere separar la aplicación en diferentes vistas, dependiendo de la petición interceptada.
    - Sintaxis

```
<jsp:forward page="URL"/ >
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:forward>
    - Ejemplo. Formulario HTML que pide nombre y password y los envía a una página jsp que lo analiza. Página *comprobar.html.*

```
<HTML><head> <title> Ejemplo de uso del forward </title>
</head> <body>
<h1> Ejemplo de uso del forward </h1>
<form method="post" action="comprobar.jsp">
<input type="text" name="Nombre">
<br> y clave:
<input type="password" name="password"> </p>
<p><input type="submit" name="login">
</form></body></HTML>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:forward>
    - Ejemplo. EL fichero comprobar.jsp:

```
<% if
((request.getParameter("Nombre").equals("Roberto")) &&
(request.getParameter("password").equals("xyzzy"))) {
%>
<jsp:forward page="saludoforward.jsp"/>
<% } else { %>
<%@ include file="comprobar.html"%>
<% } %>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:forward>
    - Ejemplo. EL fichero saludoforward.jsp :

```
<HTML>
<head>
<title> Saludo al cliente </title>
</head>
<body>
<h1> Saludo al cliente</h1>
<%
out.println("Bienvenido a la nueva aplicación");
%>
</body> </HTML>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:useBean>
    - Esta acción se utiliza para instanciar un JavaBean si no existe, o localizar una instancia ya existente, para su uso desde la página JSP.
    - Los JavaBeans son objetos Java que cumplen ciertas características de diseño en la creación.
    - Se utilizan para reducir al máximo el código Java insertado en una página JSP.
    - Permite separar la lógica de ejecución (en el JavaBean) de la presentación (en el servlet generado)

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:useBean>
    - Los JavaBeans se caracterizan porque a sus atributos (llamados propiedades) se acceden (por convenio) a través de los métodos **set**NombreAtributo y **get**NombreAtributo
    - Si se usa un JavaBean en una página habrá que definir la **clase externa** correspondiente, creando los métodos **set** y **get**.
    - Dentro del servlet generado se puede llamar a métodos de un JavaBean que se encarguen de realizar ciertas operaciones.

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:useBean>
    - Sintaxis

```
<jsp:useBean id="nombre" scope="nombreámbito" detalles />
```

    - Características de los atributos de esta acción:
      - En id se define el nombre asignado al JavaBean (identificador asociado)
      - El ámbito se refiere a dónde puede referenciarse el Javabean. Permite compartir objetos en una sesión
        - "page","request","session" y "application"
      - Los detalles pueden ser:
        - class="Nombre de la clase del JavaBean"

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:setProperty>
    - Esta marca se utiliza junto con la marca useBean para asignar valor a las propiedades del Bean
    - En el método _jspService() del servlet generado se invoca al método set de la propiedad deseada.
    - Sintaxis

```
<jsp:setProperty name =“identificador del Bean”  detalles/>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:setProperty>
    - Sintaxis. Donde detalles puede ser:
      - **property=“*”** (se cogen como propiedades y valores todos los parámetros del objeto implícito request)
      - **property=“Nombre”** (se coge un parámetro con el mismo nombre del objeto implícito request)
      - **property=“Nombre” param=“NombreParámetro”** (si se desean nombres distintos)
      - **property=“Nombre” value=“valor parámetro”** (se asignan propiedades arbitrarias con valores concretos)

Introducción a JavaServer Pages

- Elementos JSP.
  - Acción <jsp:getProperty>
    - Se utiliza para obtener el valor de las propiedades de un Bean.
    - Dentro del método _jspService() del servlet generado se accede al valor de una propiedad, **lo convierte a string y lo imprime en la salida del cliente (objeto out)**.
    - Sintaxis

```
<jsp:getProperty> name="nombre del Bean"
property="Nombre de la propiedad" />
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Objetos Implícitos.
    - Los objetos implícitos, son aquellos que están accesibles al motor JSP, por lo que el desarrollador de páginas JSP puede utilizarlos cuando los necesite sin más que invocarlos adecuadamente.
    - Son objetos creador por el motor que no necesitan ser declarados para ser usados, sino que se pueden invocar directamente.
    - Modelan mucha de la funcionalidad básica de cualquier aplicación web: tratamiento de sesiones, procesamiento de un formulario, etc.

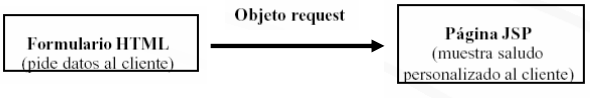
Introducción a JavaServer Pages

- Elementos JSP.
  - Objetos Implícitos.

out : Proporciona métodos para generar las salidas dentro de scriplets.
request: Para obtener los parámetros que vienen de un formulario.
response: Ligado con out, ayuda a generar la salida HTML.
pageContext: API para acceder a objetos relacionados con los servlets.
session: Se utiliza para mantener información sobre el cliente que ha establecido conexión con el servidor a la largo de un cierto tiempo.
application: Proporciona un medio de comunicación entre páginas JSP y el servidor sin intervención del objeto request.
page: Representa la instancia del servlet generado por la página JSP
exception: Solamente está disponible si la excepción no ha sido capturada y se ha utilizado el atributo ERRORPAGE de la directiva page de la página JSP
config: Acceso a los parámetros de inicialización del servlet o del motor JSP.

Introducción a JavaServer Pages

- Elementos JSP.
  - Ejemplo de uso de Objetos Implícitos.
    - Aplicación que pide el nombre al usuario y le devuelve un saludo. Se utiliza un fichero HTML con un formulario que pide los datos al cliente y se los pasa a una página JSP que muestra el saludo con éstos datos. El paso de los datos del formulario al JSP se realiza a través de un objeto especial: objeto **request**.



Introducción a JavaServer Pages

- Elementos JSP.
  - Ejemplo de uso de Objetos Implícitos.

```
<HTML><head>
<title> Formulario de petición de nombre
</title></head>
<body>
<h1> Formulario de petición de nombre </h1>
<!-- Se envía el formulario al JSP “saludo.jsp” -->
<form method="post" action="saludo.jsp">
<p> Por favor, introduce tu nombre:
<input type="text" name="nombre">
</p>
<p> <input type="submit" value="enviar
información"> </form> </body>
</HTML>
```

Introducción a JavaServer Pages

- Elementos JSP.
  - Ejemplo de uso de Objetos Implícitos.

```
<HTML> <head> <title> Saludo al cliente </title>
</head> <body>
<h1> Saludo al cliente</h1>
<%-- Los parámetros que le pasa el cliente en la
petición se obtienen del objeto implícito
request --%>
<%
String nombre = request.getParameter(”nombre");
out.println(”Es un placer conocerte, “ + nombre);
%>
<%-- Al evaluarse el código hay que escribir
explícitamente en la salida (objeto implícito
out) --%> </body> </HTML>
```

Introducción a JavaServer Pages

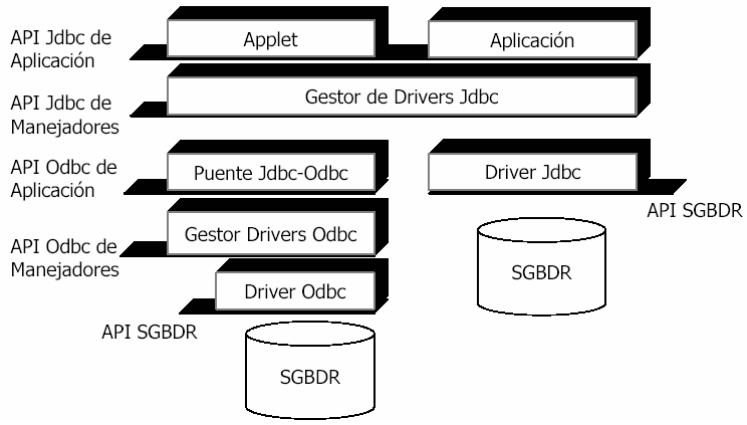
- Ciclo de Vida.
  - Cuando se llama por primera vez al fichero JSP, se genera un servlet con las siguientes operaciones
  - **jspInit()**
    - Inicializa el servlet generado
    - Sólo se llama en la primera petición
  - **jspService(petición,respuesta)**
    - Maneja las peticiones. Se invoca en cada petición, incluso en la primera
  - **jspDestroy()**
    - Invocada por el motor para eliminar el servlet

Introducción a JavaServer Pages

- Acceso a Datos.
  - **JDBC**, Java Database Connectivity, es una API de Java que se usa para acceder a Bases de Datos, tanto locales como remotas.
  - Existe una independencia del SGBDR: Si cambia el gestor se minimizan los cambios en la aplicación. Lenguaje de consulta SQL.
  - El API JDBC (java.sql) está formada por cinco grupos:
    - Gestión de Drivers.
    - API para manejadores JDBC.
    - Excepciones.
    - API (trabajar con los datos)
    - Utilidades.

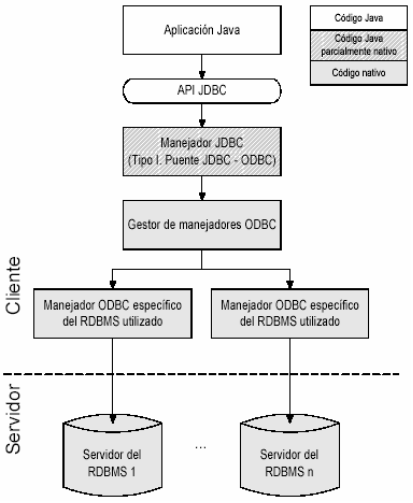
Introducción a JavaServer Pages

- Acceso a Datos. Arquitectura.



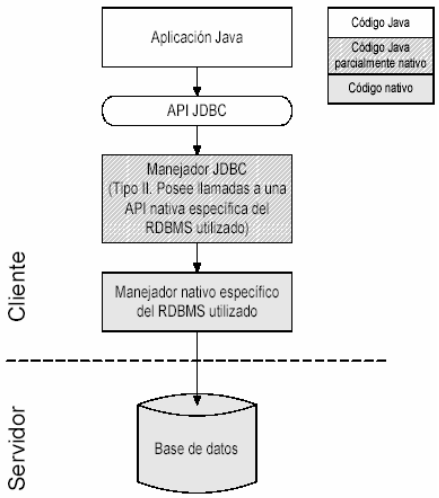
Introducción a JavaServer Pages

- Acceso a Datos. Arquitectura JDBC Tipo I
  - Puente Jdbc-Odbc
  - Permite conectar con cualquier SGBR que soporte ODBC.
  - Es muy lento



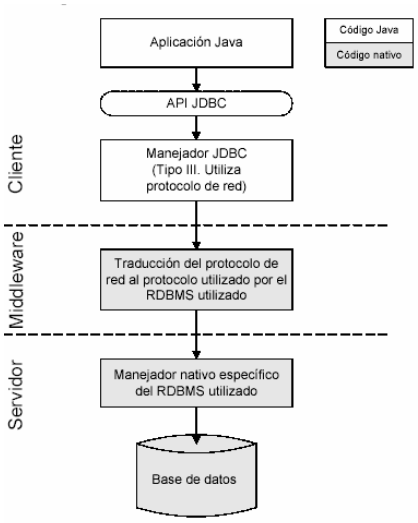
Introducción a JavaServer Pages

- Acceso a Datos. Arquitectura JDBC Tipo II
  - Se conecta directamente con un SGBD, siendo especifico de cada SGBD.
  - Es más rápido.
  - Un manejador diferente para cada sistema gestor.



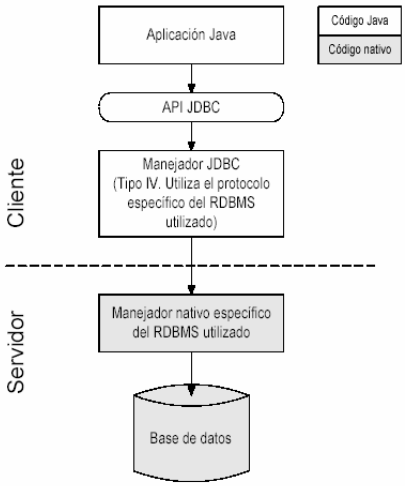
Introducción a JavaServer Pages

- Acceso a Datos. Arquitectura JDBC Tipo III
  - El manejador JDBC traduce las peticiones JDBC a un protocolo independiente del SGBDR.
  - Se requiere un middleware como traductor.
  - Mayor coste.
  - Menor Velocidad.



Introducción a JavaServer Pages

- Acceso a Datos.
  Arquitectura JDBC Tipo IV
  - Utiliza el protocolo propietario de red del SGBDR para hablar con él de forma remota.
  - No necesitamos middleware.
  - Mayor Velocidad.



Introducción a JavaServer Pages

- Acceso a Datos.
  - Cuando queremos trabajar con BBDD, debemos tener presente una serie de pasos **“básicos”:**
    - Abrir la conexión a la base de datos.
    - Ejecutar consultas contra la base de datos.
    - Procesar los resultados.
    - Cerrar la conexión a la base de datos.

Introducción a JavaServer Pages

- Acceso a Datos.
  - Abrir la conexión a la base de datos.

```
Connection conexion =
DriverManager.getConnection("jdbc:odbc:Nombre_ODBC",
"usuario","password");
```

  - Ejecutar consultas contra la base de datos

```
Statement Comando = conexion.createStatement();
ResultSet rs = Comando.executeQuery("select dni, nombre,
apellidos, edad from agenda");
```

Introducción a JavaServer Pages

- Acceso a Datos.
  - Procesar los resultados.

```
while (rs.next()) {
out.println("DNI ->"+ rs.getString("dni"));
out.println("NOMBRE ->"+ rs.getString("nombre"));
out.println("APELLIDOS ->"+ rs.getString("apellidos"));
out.println("EDAD ->"+ rs.getInt("edad"));
}
```

  - Cerrar conexión.

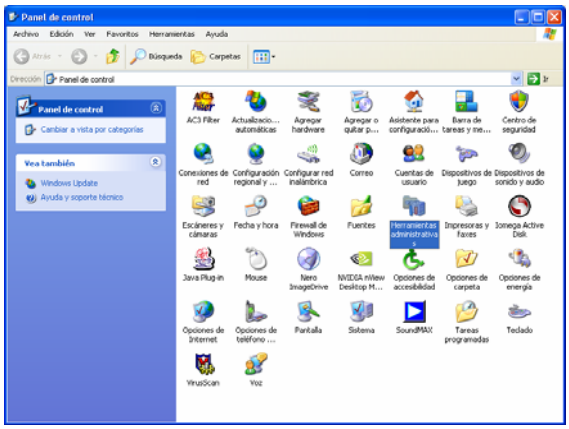
```
rs.close();
Comando.close();
conexion.close();
```

Introducción a JavaServer Pages

- Acceso a Datos.Conector JDBC-ODBC.
  - Cómo Crear un **Conector** vía JDBC-ODBC.
    - El puente JDBC-ODBC no necesita pasos específicos para su instalación, pero **ODBC** si.
    - Por ejemplo si asumimos que estamos utilizando un máquina con S.O. Windows necesitaremos configurar nuestra conexión mediante ODBC.
    - A continuación se presentan las pantallas necesarias para crear un conector.

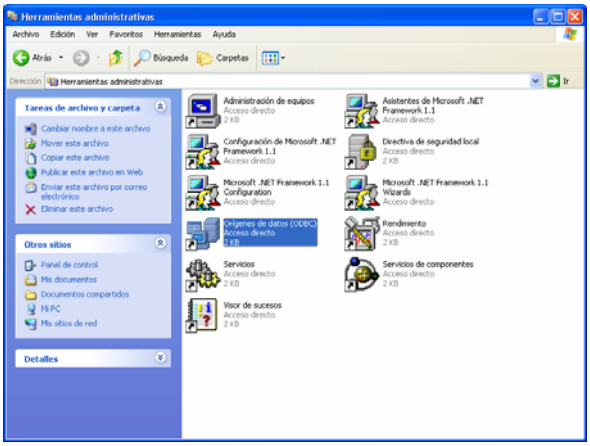
Introducción a JavaServer Pages

- Acceso a Datos. Conector JDBC-ODBC.



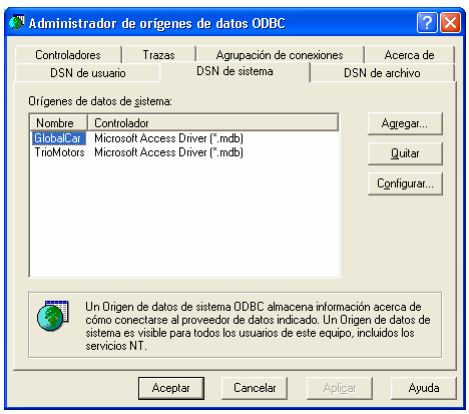
Introducción a JavaServer Pages

- Acceso a Datos. Conector JDBC-ODBC.



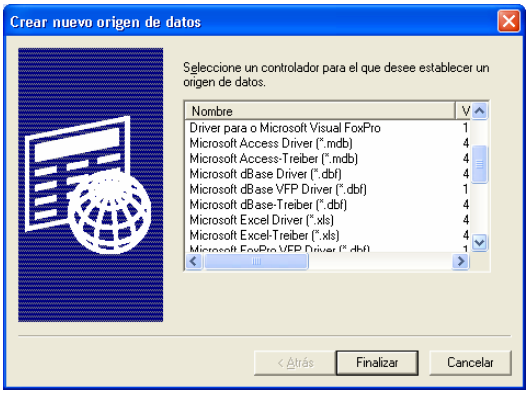
Introducción a JavaServer Pages

- Acceso a Datos. Conector JDBC-ODBC.



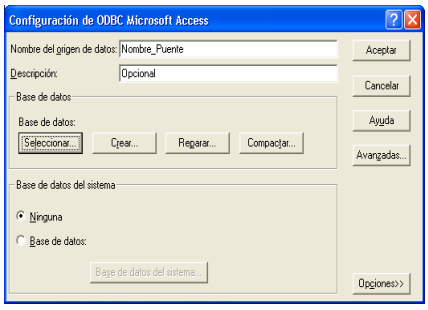
Introducción a JavaServer Pages

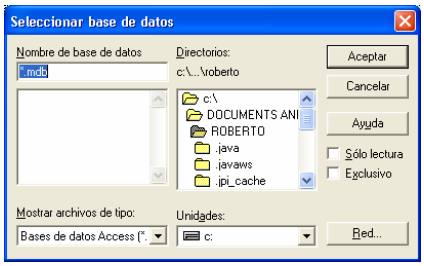
- Acceso a Datos. Conector JDBC-ODBC.



Introducción a JavaServer Pages

- Acceso a Datos. Conector JDBC-ODBC.





Introducción a JavaServer Pages

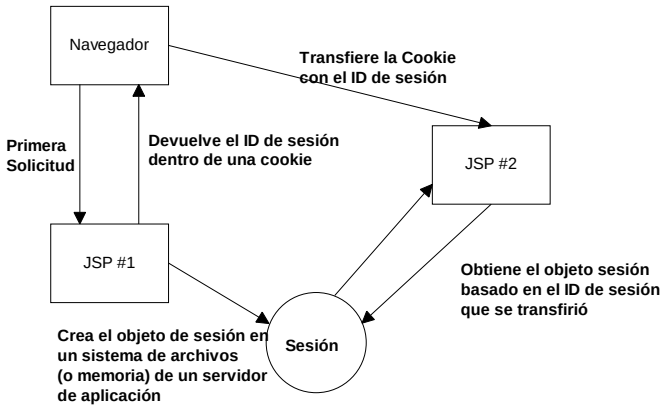
- Gestión de sesiones.
  - El protocolo HTTP es un protocolo sin estado. No posee funcionalidad interna para seguir datos de una solicitud a otra.
  - Aunque esto puede proporcionar algunos beneficios en cuanto al rendimiento, también supone un problema en alguna aplicaciones web. Por ejemplo, en aplicaciones tipo: tienda virtual, acceso a un banco, etc.
  - La tecnología JSP proporciona acceso a un objeto implícito ***session***, para almacenar y recuperar valores de una conexión lógica: sesión.

Introducción a JavaServer Pages

- Gestión de sesiones.
  - El objeto ***session*** es único para cada usuario concreto. La primera vez que un usuario solicita una página jsp, se crea la sesión, esta sesión está identificada por un **ID de sesión único** y asociado a ella.
  - Cada vez que el usuario solicita una página jsp, el ID de la sesión **se transfiere con la solicitud** y, por tanto, se asocia a la sesión adecuada con el usuario.
  - El ID es transferido entre el servlet (transformación página JSP) y el usuario a través de una **cookie** HTTP.

Introducción a JavaServer Pages

- Gestión de sesiones. Funcionamiento.



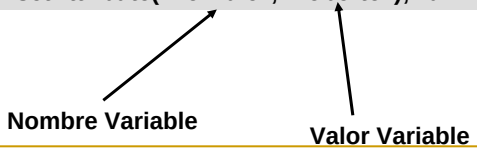
Introducción a JavaServer Pages

- Gestión de sesiones.
  - El objeto ***session*** es análogo a la clase **HttpSession** de la tecnología servlets.
  - Una vez obtenida la sesión, normalmente mediante el **objeto implícito**, es posible:
    - Escribir,
    - Obtener,
    - y Eliminar atributos del objeto **session**.

Introducción a JavaServer Pages

- Gestión de sesiones.
  - Por ejemplo, para escribir o dar de alta un atributo (variable de sesión) en la sesión, deberíamos utilizar el método **setAttribute** de la siguiente manera:

```
<% session.setAttribute("nombre","Roberto"); %>
```



Introducción a JavaServer Pages

- Gestión de sesiones.
  - Para recuperar este valor se utiliza el método **getAttribute**.

```
<%String SNombre=(String) session.getAttribute("nombre"); %>
```

  - Observar cómo se asigna el resultado de este método a String. Esto es necesario dado que la sesión, sólo almacena un objeto y devuelve un objeto, sin tener en cuenta el tipo de objeto que esté almacenado. Tenemos que realizar una conversión.

Introducción a JavaServer Pages

- Gestión de sesiones.
  - Para eliminar un atributo se debe utilizar el método **removeAttribute**.

```
<% session.removeAttribute("nombre"); %>
```

Introducción a JavaServer Pages

- Gestión de sesiones. Ejemplo.
  - Como ejemplo, se presentará un sencillo formulario que acepta el nombre de una persona.
  - Enviaremos este formulario a una página JSP que guardará el nombre en la sesión y presentará al usuario mediante dos sencillos hipervínculos a otras páginas JSP.
  - Una vez que el usuario vaya a una de dichas páginas, se recuperará el nombre de la sesión y aparecerá un mensaje personalizado.

Introducción a JavaServer Pages

- Gestión de sesiones. Formulario.

```
<HTML><head>
<title> Ejemplo de Sesión </title></head>
<body>
<h1> Ejemplo de sesión </h1>
<form method="post" action="sesionEje.jsp">
Por favor, introduce tu nombre:
<input type="text" name="nombre">
<input type="submit" value="enviar información"> </form>
</body>
</HTML>
```

Introducción a JavaServer Pages

- Gestión de sesiones. sesionEje.jsp

```
<HTML>
<head> <title> Ejemplo de Sesión </title> </head>
<body>
<%
String val = request.getParameter("nombre");
if (val != null) session.setAttribute("Nombre",val);
%>
<center> <h1>Ejemplo de Sesión</h1>
Donde quieres ir!!!
<a href="sesionEje1.jsp">Ir a Página 1</a>
<a href="sesionEje2.jsp">Ir a Página 2</a>
</body>
</HTML>
```

Introducción a JavaServer Pages

- Gestión de sesiones. sesionEje1.jsp

```
<HTML><head> <title> Ejemplo de Sesión </title> </head>
<body>
<center> <h1>Ejemplo de Sesión</h1>
Hola, <=%session.getAttribute("Nombre")%>
Bienvenido a la página 1
</body>
</HTML>
```

Introducción a JavaServer Pages

- Gestión de sesiones. sesionEje2.jsp

```
<HTML><head> <title> Ejemplo de Sesión </title> </head>
<body>
<center> <h1>Ejemplo de Sesión</h1>
Hola, <=%session.getAttribute("Nombre")%>
Bienvenido a la página 2
</body>
</HTML>
```

Introducción a JavaServer Pages

- Uso de JavaBean.
  - Un **JavaBean** es una clase Java que sigue un conjunto de **guías de estilo** para permitir la creación de componentes Java distribuidos.
  - La tecnología JSP soporta el uso de JavaBeans como mecanismo de acceder a objetos remotos.
  - El objetivo básico del uso de los JavaBean, es reducir el código Java que hay en las páginas JSP.

Introducción a JavaServer Pages

- Uso de JavaBean.
  - En el contexto de JSP, un JavaBean será una clase que obedece a las siguientes guías de estilo:
    - **Constructor público** sin argumentos. Si no se indica ningún constructor explícito, dispone del constructor por defecto.
    - Dispone de propiedades que se pueden consultar con métodos **getXXX** y/o modificar con métodos **setXXX**
    - Las propiedades pueden ser controladas por:
      - **public TipoPropiedad getNombrePropiedad()**
      - **public void setNombrePropiedad(TipoPropiedad propiedad)**
    - Una propiedad de tipo boolean tiene una convención de nombres diferente para el método “getXXX” -> “isXXX”

Introducción a JavaServer Pages

- Uso de JavaBean.
  - Las **acciones** son etiquetas especiales que afectan el comportamiento en **ejecución** de cualquier página JSP.
  - Las acciones estándares son las soportadas por todos los **contenedores web**. Para manejar JavaBeans se utilizan las siguientes:

```
<jsp: useBean>
<jsp: setProperty>
<jsp: getProperty>
```

Introducción a JavaServer Pages

- Uso de JavaBean. La acción useBean.

```
<jsp:useBean id="name"
scope="page"|"request"|"session"|"application"
class="Nombre de la Clase" />
```

Atributo	Nombre
id	Nombre del objeto. Es *case-sensitive*
scope	Alcance del objeto. Indica donde está disponible el objeto. El *default* es *page*.
class	Nombre completo de la clase.
beanName	El nombre de un Bean.
type	Especifica el tipo de la clase y respeta las reglas de *casting* de Java. El valor por *default* es el mismo que el *atributo class*.

<jsp:useBean id="test" scope="session" class="TestBean.class"/>

Introducción a JavaServer Pages

- Uso de JavaBean.
  - Acciones setProperty y getProperty.
    - Esta acciones estándar de JSP permiten obtener y modificar los valores de los atributos definidos en el JavaBean.
    - A continuación se presenta un ejemplo de utilización de todas las acciones relacionadas con el uso de un JavaBean.

Introducción a JavaServer Pages

- Uso de JavaBean. Ejemplo: MensajeBean

```
package ejemplos;
public class MensajeBean
{
      private String mensaje = "Valor no establecido";

      public String getMensaje()
      {
      return (mensaje);
      }
      public void setMensaje(String mensaje)
      {
      this.mensaje = mensaje;
      }
}
```

Introducción a JavaServer Pages

- Uso de JavaBean. Ejemplo: Bean.jsp

```
<html> <head> <title>Ejemplo de beans </title></head>
<body>
<center><h1>Ejemplo de JavaBeans</h1></center>

<jsp:useBean id="mensaje" class="ejemplos.MensajeBean"/>

<% mensaje.setMensaje("Hola mundo desde un Bean"); %>
<center> <h1>Mensaje: <i> <%=mensaje.getMensaje(); %>
<% mensaje.setMensaje("Hola otra vez desde el Bean"); %>
</i>Mensaje: <i> <%=mensaje.getMensaje(); %>
</i></h1></center>
</body>
 </html>
```


Introducción a JavaServer Pages

- Uso de JavaBean. Ejemplo: Bean.jsp

```
<html> <head> <title>Ejemplo de beans </title></head>
<body>
<center><h1>Ejemplo de JavaBeans</h1></center>

<jsp:useBean id="mensaje" class="ejemplos.MensajeBean"/>

<% mensaje.setMensaje("Hola mundo desde un Bean"); %>
<center> <h1>Mensaje: <i> <%=mensaje.getMensaje(); %>
<% mensaje.setMensaje("Hola otra vez desde el Bean"); %>
</i>Mensaje: <i> <%=mensaje.getMensaje(); %>
</i></h1></center>
</body>
 </html>
```

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - En JSP, las acciones eran elementos que pueden utilizarse para manejar objetos Java.
  - Existen acciones estándar, definidas por la especificación JavaServer Pages, o acciones personalizadas, creadas por el programador.
  - Una acción personalizada se invoca en una página JSP mediante una **etiqueta personalizada**.
  - Una **librería de etiquetas** es una colección de etiquetas personalizadas.

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - Una etiqueta JSP es una forma simple de abstracción del código Java fuera de la página JSP.
  - Posibilitando que no haya código Java en una página JSP.

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - Reducen la presencia de Scriptlets en las páginas JSP
  - Las etiquetas personalizadas tienen una sintaxis muy simple, mediante la notación XML.
  - Mejoran la calidad de los desarrollos.
  - Encapsular funciones simples o complejas que facilita la legibilidad de las páginas JSP.
  - Facilitan la reutilización de código.

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - Para que una página JSP pueda utilizar una librería de etiquetas, ésta debe estar compuesta por dos tipos de componentes:
    - El **fichero descriptor de la librería de etiquetas**.
    - Los **controladores** o manejadores de las etiquetas.

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - El fichero descriptor de la librería de etiquetas, o fichero ***Tag Library Descriptor*** **(TDL),** es un documento XML que describe la librería.
  - Contiene información acerca de la librería en su conjunto y acerca de cada una de las etiquetas que la componen en particular.
  - Este fichero es utilizado por el contenedor de JSP´s a la hora de validar la utilización de las etiquetas de la librería.

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - El fichero TDL contiene información de cabecera previa a los elementos utilizados para describir la librería. Entre esta información se encuentra:

<taglib> La librería de etiquetas en sí misma

<tlibversion> Versión de la librería de etiquetas

<jspversion> Versión de la especificación JSP soportada por la librería de etiquetas. Por defecto es 1.2

<shortname> Nombre corto que suele utilizarse como prefijo en la directiva *taglib*

<uri> Opcional que identifica la librería de etiquetas.

<info> Descripción opcional de la librería de etiquetas.

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - Tras la cabecera se encuentra la descripción de cada una de las etiquetas que la integran.
  - De los elementos que describen a estas etiquetas, solamente uno de ellos es obligatorio: **<tagclass>**
  - A continuación se muestra un ejemplo de fichero TLD, donde podemos ver lo comentado hasta ahora.

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas. Ejemplo de fichero TLD.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<taglib>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.2</jspversion>
  <shortname>ejemplo</shortname>
  <info> Ejemplo de Librería de Etiquetas
  </info>
<!-- Etiqueta simple de saludo -->
  <tag>
    <name>hola</name>
    <tagclass>ejemplos.HolaTag</tagclass>
    <bodycontent>empty</bodycontent>
    <info>Etiqueta simple de saludo</info>
    <!-- nombre personalizado -->
    <attribute>
      <name>nombre</name>
    </attribute>
  </tag>
  </taglib
```

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - Para utilizar la librería de etiquetas descrita en el archivo TDL en una página JSP, es necesario importar esa librería a través de la **directiva *taglib***, cuya sintaxis es la siguiente:

```
<%@ taglib uri="uri" prefix="prefijo"%>
```

  - Donde *URI* es la dirección que indica la localización del fichero TDL y *prefijo* es el identificador con el que se reconocen las etiquetas pertenecientes a esa librería en la página JSP.

Introducción a JavaServer Pages

- Uso de Librería de Etiquetas.
  - Cada etiqueta personalizada integrante de una librería de etiquetas debe estar definida en una clase controladora.
  - Cuando el motor JSP se encuentra con el elemento de comienzo de una etiqueta personalizada, inicializa el controlador de la etiqueta y luego invoca al método ***doStartTag()*** de esa etiqueta.
  - Cuando se encuentra la etiqueta de cierre, se invoca al método ***doEndTag()*** de la etiqueta personalizada.

Laboratorio de Informática Distribuida

Sesión III. JavaServer Pages