Introducción a los Sistemas de Información con Servlets y JSP

por

Gilberto Pacheco Gallegos

Copyright (c) 2009
Todos los derechos reservados

Índice de contenido

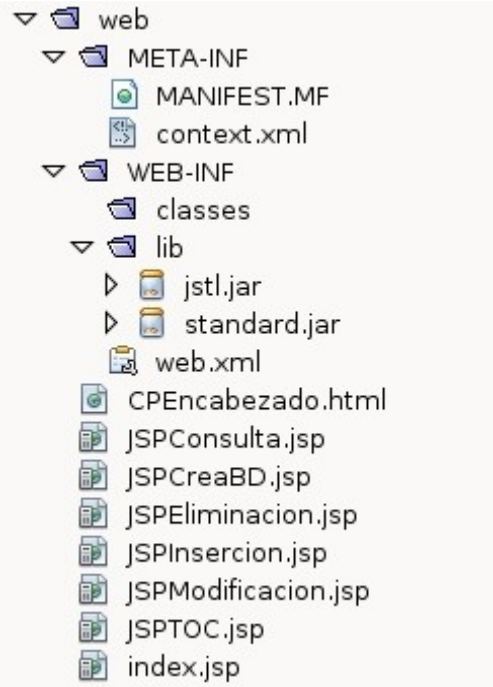
1. Estructura General de un Sitio Web JEE.

La tecnología para sitios web de Java puede montarse sobre distintos servidores, computadoras y sistemas operativos sin tener que realizar cambios en el código y en la configuración. Algunos productos que son servidores web de Java son Apache Tomcat y Jetty, que a su vez son usados en otros productos, como son: WEA Weblogic, Jboss, Glassfish y Geronimo, que son servidores de aplicaciones para empresas.

1.1. Estructura de Archivos.

Toda aplicación web de Java, consta de:

- Una carpeta del proyecto, donde se colocan las páginas web
- Una carpeta **WEB-INF** dentro de la carpeta del proyecto, que contiene archivos de configuración. El más usado de estos es el archivo **web.xml**.
- Clases compiladas pueden colocarse en la carpeta **WEB-INF/classes**.
- Archivos jar que pueden colocarse en la carpeta **WEB-INF/lib**.
- Algunos servidores, como Tomcat usan la carpeta **META-INF** para colocar el archivo context.xml.



2. Introducción a los Servlets.

Los servlets son la base de la creación de páginas dinámicas; es decir, páginas construidas de forma única para cada solicitud de un cliente web.

2.1. Un Servlet Sencillo.

Cuando este ejemplo se ejecuta, genera una página que muestra un saludo e indica el nombre de la aplicación.

2.1.1. SpServletSencillo.

```
 1 package servlets;
 2
 3 import java.io.IOException;
 4 import java.io.PrintWriter;
 5 import javax.servlet.ServletException;
 6 import javax.servlet.http.HttpServlet;
 7 import javax.servlet.http.HttpServletRequest;
 8 import javax.servlet.http.HttpServletResponse;
 9
10 /** Un servlet de ejemplo. */
11 public class SPServletSencillo extends HttpServlet {
12     /** Maneja el método <CODE>GET</CODE> de HTTP y genera la página a
13      * mostrar.
14      * @param solicitud la solicitud al servlet.
15      * @param respuesta la respuesta del servlet. */
16     @Override
17     protected void doGet(HttpServletRequest solicitud,
18             HttpServletResponse respuesta)
19             throws ServletException, IOException {
20         respuesta.setContentType("text/html;charset=UTF-8");
21         PrintWriter out = respuesta.getWriter();
22         try {
23             out.println("<!DOCTYPE HTML PUBLIC " +
24                     "\"-//W3C//DTD HTML 4.01//EN\" " +
25                     "\"http://www.w3.org/TR/html4/strict.dtd\">");
26             out.println("<HTML>");
27             out.println("<HEAD>");
28             out.println("<META http-equiv='Content-Type' " +
29                     "content='text/html; charset=UTF-8'>");
30             out.println("<TITLE>Server Page ServletSencillo</TITLE>");
31             out.println("</HEAD>");
32             out.println("<BODY>");
33             out.println("<H1>Saludos desde " + solicitud.getContextPath() +
34                     "</H1>");
35             out.println("</BODY>");
36             out.println("</HTML>");
37         } finally {
```

```
38              out.close();
39          }
40      }
41      /** Devuelve breve información del servlet.
42       * @return una breve información del servlet. */
43      @Override
44      public String getServletInfo() {
45          return "versión: 1.0. Copyright 2009 Gilberto Pacheco Gallegos.";
46      }
47  }
```

2.1.2. Archivo “web.xml”.

En este archivo define un servlet llamado `SPServletSencillo`, que se construye a partir de la clase `servlets.SPServletSencillo`.Se localiza en la ruta `/SPServletSencillo` dentro de la aplicación. Al invocarse el URL de la aplicación, se muestra el archivo `index.html`.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5           http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
6      <servlet>
7          <servlet-name>SPServletSencillo</servlet-name>
8          <servlet-class>servlets.SPServletSencillo</servlet-class>
9      </servlet>
10     <servlet-mapping>
11         <servlet-name>SPServletSencillo</servlet-name>
12         <url-pattern>/SPServletSencillo</url-pattern>
13     </servlet-mapping>
14     <welcome-file-list>
15         <welcome-file>index.html</welcome-file>
16     </welcome-file-list>
17 </web-app>
```

2.1.3. Archivo “index.html”.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2  "http://www.w3.org/TR/html4/strict.dtd">
3  <HTML>
4      <HEAD>
5          <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
6          <TITLE>Mi Primer Servlet</TITLE>
7      </HEAD>
8      <BODY>
9          <H1>Mi Primer Servlet</H1>
10         <P><A href="SPServletSencillo">Ejecutar</A></P>
11     </BODY>
12 </HTML>
```

2.2. Un Servlet Configurable.

Los servlets pueden configurarse con información almacenada en el archivo “web.xml”. Hay dos tipos de parámetros: Para toda la aplicación y para cada servlet.

2.2.1. SPServletParametrizado.

```
 1 package servlets;
 2
 3 import java.io.IOException;
 4 import java.io.PrintWriter;
 5 import javax.servlet.ServletContext;
 6 import javax.servlet.ServletException;
 7 import javax.servlet.http.HttpServlet;
 8 import javax.servlet.http.HttpServletRequest;
 9 import javax.servlet.http.HttpServletResponse;
10
11 /** Un servlet que lee los parámetros declarados en <CITE>web.xml</CITE>.
12  * los <STRONG>context-param</STRONG> se recuperan con el método
13  * <CODE>ServletContext.getInitParameter</CODE> y los
14  * <STRONG>init-param</STRONG> con
15  * <CODE>GenericServlet.getInitParameter</CODE>. */
16 public class SPServletParametrizado extends HttpServlet {
17     /** Maneja el método <CODE>GET</CODE> de HTTP.
18      * @param solicitud la solicitud al servlet.
19      * @param respuesta la respuesta del servlet. */
20     @Override
21     protected void doGet(HttpServletRequest solicitud,
22             HttpServletResponse respuesta)
23             throws ServletException, IOException {
24         respuesta.setContentType("text/html;charset=UTF-8");
25         PrintWriter out = respuesta.getWriter();
26         ServletContext contexto = getServletContext();
27         try {
28             out.println("<!DOCTYPE HTML PUBLIC " +
29                     "\"-//W3C//DTD HTML 4.01//EN\" " +
30                     "\"http://www.w3.org/TR/html4/strict.dtd\">" +
31                     "<HTML>" +
32                     "<HEAD>" +
33                     "<TITLE>Saludo</TITLE>" +
34                     "</HEAD>" +
35                     "<BODY>" +
36                     "<H1>" + getInitParameter("saludo") + "</H1>" +
37                     "<P>Se autoriza el uso de este software.</P> " +
38                     "<TABLE border='1'>" +
39                     "<CAPTION>Datos de la Licencia</CAPTION> " +
40                     "<THEAD>" +
41                     "<TR><TH>Empresa</TH><TH>Tipo de Licencia</TH></TR>" +
42                     "<TBODY>" +
43                     "<TR><TD>" + contexto.getInitParameter("empresa") +
44                     "</TD><TD>" + contexto.getInitParameter("tipo") +
```

```
45                      "</TD></TR>" +
46                      "</TABLE>" +
47                      "</BODY>" +
48                      "</HTML>");
49          } finally {
50              out.close();
51          }
52      }
53      /** Devuelve breve información del servlet.
54       * @return una breve información del servlet. */
55      @Override
56      public String getServletInfo() {
57          return "versión: 1.0. Copyright 2009 Gilberto Pacheco Gallegos.";
58      }
59  }
```

2.2.2. Archivo “web.xml”.

En este caso, con una misma clase se crean dos servlets con parámetros de inicio diferentes.

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
 3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 5           http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 6      <!-- Parámetros para toda la aplicación web. -->
 7      <context-param>
 8          <description>
 9              Nombre de la empresa que utiliza la aplicación
10          </description>
11          <param-name>empresa</param-name>
12          <param-value>El Patito Pirata</param-value>
13      </context-param>
14      <context-param>
15          <description>Tipo de cliente</description>
16          <param-name>tipo</param-name>
17          <param-value>Pirata</param-value>
18      </context-param>
19      <!-- Servlets. -->
20      <servlet>
21          <servlet-name>SPServlet1</servlet-name>
22          <servlet-class>servlets.SPServletParametrizado</servlet-class>
23          <init-param>
24              <param-name>saludo</param-name>
25              <param-value>Hola</param-value>
26          </init-param>
27      </servlet>
28      <servlet>
29          <servlet-name>SPServlet2</servlet-name>
30          <servlet-class>servlets.SPServletParametrizado</servlet-class>
31          <init-param>
32              <param-name>saludo</param-name>
```

```
33              <param-value>Hello</param-value>
34          </init-param>
35      </servlet>
36      <servlet-mapping>
37          <servlet-name>SPServlet1</servlet-name>
38          <url-pattern>/SPServlet1</url-pattern>
39      </servlet-mapping>
40      <servlet-mapping>
41          <servlet-name>SPServlet2</servlet-name>
42          <url-pattern>/SPServlet2</url-pattern>
43      </servlet-mapping>
44      <welcome-file-list>
45          <welcome-file>index.html</welcome-file>
46      </welcome-file-list>
47  </web-app>
```

2.2.3. Archivo “index.html”.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
2  "http://www.w3.org/TR/html4/frameset.dtd">
3  <HTML>
4      <HEAD>
5          <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
6          <TITLE>Servlets con Parámetros</TITLE>
7      </HEAD>
8      <FRAMESET cols="20%, 80%">
9          <FRAME src="CPToc.html">
10         <FRAME src="SPServlet1" name="dinamico">
11         <NOFRAMES>
12             <H1>Contenido</H1>
13             <UL>
14                 <LI><A href="SPServlet1">Servlet1</A></LI>
15                 <LI><A href="SPServlet2">Servlet2</A></LI>
16             </UL>
17         </NOFRAMES>
18     </FRAMESET>
19 </HTML>
```

2.2.4. Archivo “CPToc.html”.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
2  "http://www.w3.org/TR/html4/loose.dtd">
3  <HTML>
4      <HEAD>
5          <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
6          <TITLE>Contenido</TITLE>
7      </HEAD>
8      <BODY>
9          <H1>Contenido</H1>
10         <UL>
11             <LI><A href="SPServlet1" target="dinamico">Servlet1</A></LI>
```

```
12              <LI><A href="SPServlet2" target="dinamico">Servlet2</A></LI>
13          </UL>
14      </BODY>
15  </HTML>
```

2.3. La Interfaz ServletContext.

public interface **javax.servlet.ServletContext**

Define un conjunto de métodos que un servlet usa para comunicarse con su contenedor, por ejemplo, para obtener el tipo MIME de un archivo, para despachar solicitudes o para escribir a un archivo de **bitácora** (donde se registran lo eventos importantes y los fallos).

Solo existe un contexto para cada **aplicación web** y por máquina virtual de Java. (Una aplicación web es una colección de servlets y contenido, instalada bajo una ruta específica del espacio de nombres URL del servidor, por ejemplo /catalog y posiblemente instalado con un archivo de extensión ".war".)

En el caso de que una aplicación web sea marcada "distributed" en su descriptor de despliegue, habrá una instancia del contexto por máquina virtual. En este caso, el contexto no puede usarse para almacenar información global; se debe usar una instancia externa, por ejemplo una base de datos.

El objeto ServletContext se almacena entro del objeto ServletConfig, que el servidor web proporciona al servlet cuando lo inicializa.

2.3.1. Métodos de la Interfaz ServletContext.

- **String getContextPath().** Devuelve la **ruta del contexto** de la aplicación web, que es la la parte del URI de las solicitudes que se usa para seleccionar el contexto. Siempre aparece primero en un URI. Inicia con "/", pero no termina con este caracter. Para servlets en el contexto por defecto (root), este método devuelve "".

  Es posible que el contenedor de un servlet se ajuste a mas de una ruta de contexto. En tales casos, el método HttpServletRequest.getContextPath() devuelve el utilizado por la solicitud, que puede ser diferente de la ruta devuelta por este método, que debe considerarse como la ruta de contexto preferida para la aplicación.

- **ServletContext getContext(String rutaURI).** Devuelve un objeto **ServletContext** dentro del servidor que corresponde a la rutaURI especificada. Este método permite tener acceso al contexto de diversas partes del servidor y es necesaria para obtener reenviar solicitudes a otros contextos. La ruta siempre debe iniciar con "/" y se interpreta de forma relativa al inicio del documento raíz del servidor y se compara con las raíces de contexto de las otras aplicaciones

web hospedadas en el contenedor. En un ambiente consciente de la seguridad, el contenedor del servlet puede devolver null para algunas URL.

- **int getMajorVersion().** Devuelve el número mayor de versión de servlet que soporta este contenedor. Si la implementación cumple con la versión "2.5", este método debe devolver el entero "2".
- **int getMinorVersion().** Devuelve el número menor de versión de servlet que soporta este contenedor. Si la implementación cumple con la versión "2.5", este método debe devolver el entero "5".
- **String getMimeType(String archivo).** Devuelve el tipo MIME del archivo especificado, o null si es desconocido. Este valor es determinado por la configuración del contenedor de servlets, y se puede especificar en un descriptor de despliegue para la aplicación web. Un par de tipos MIME comunes son "text/html" e "image/gif".
- **Set getResourcePaths(String ruta).** Devuelve un listado de estructuras tipo directorio de todas las rutas a recursos dentro de la aplicación web, cuya subruta más larga coincide con la **ruta** proporcionada. Las rutas que indican subdirectorios terminan con "/". Todas las rutas devueltas son relativas a la raíz de la aplicación web y empiezan con "/". Por ejemplo, si una apicación web contiene
  - /welcome.html
  - /catalog/index.html
  - /catalog/products.html
  - /catalog/offers/books.html
  - /catalog/offers/music.html
  - /customer/login.jsp
  - /WEB-INF/web.xml
  - /WEB-INF/classes/com.acme.OrderServlet.class,

  getResourcePaths("/") devuelve {"/welcome.html", "/catalog/", "/customer/", "/WEB-INF/"}

  getResourcePaths("/catalog/") devuelve {"/catalog/index.html", "/catalog/products.html", "/catalog/offers/"}.
- **java.net.URL getResource(String ruta) throws java.net.MalformedURLException.** Devuelve un URL para el recurso correspondiente a la ruta especificada. La ruta debe iniciar con "/" y se interpreta de forma relativa la la raíz del contexto actual. Este método permite que los contenedores de servlets hagan que un recurso sea accesible a servlets desde cualquier origen, por ejemplo, sistemas de archivos remotos, bases de datos o archivos de extensión ".war". El contenedor de servlets debe implementar los manejadores de URL y objetos de conexión URL necesarios para acceder al recurso. Este método devuelve null si no hay ningún recurso asociado con la ruta.

- **InputStream getResourceAsStream(String ruta).** Devuelve el recurso localizado en la ruta como un objeto InputStream. Los datos en el InputStream pueden ser de cualquier longitud o tipo. La ruta se debe especificar de acuerdo a las reglas de **getResource**. Este método devuelve null si no hay ningún recurso asociado con la ruta. Metainformación tal como longitud del contenido o tipo de contenido que es accesible con **getResource**, se pierde al utilizar este método. El contenedor de servlets debe implementar los manejadores de URL y objetos de conexión URL necesarios para acceder al recurso.
- **RequestDispatcher getRequestDispatcher(String ruta).** Devuelve un objeto RequestDispatcher, que representa al recurso localizado en la ruta proporcionada. Dicho objeto puede usarse para reenviar una solicitud al recurso o para incluir este en la respuesta. El recurso puede ser dinámico o estático. La ruta debe iniciar con "/" y se interpreta de forma relativa la la raíz del contexto actual. Se puede usar **getContext** para obtener un RequestDispatcher para recursos en otros contextos. Este método devuelve null si el ServletContext no puede devolver un RequestDispatcher.
- **RequestDispatcher getNamedDispatcher(String nombre).** Devuelve un objeto RequestDispatcher, que representa al servlet correspondiente al nombre. Los servlets y páginas JSP pueden obtener nombres por medio del administración de servlet o del archivo descriptor de despliegue de la aplicación web. Una instancia de servlet puede determinar su nombre utilizando **ServletConfig.getServletName()**. Este método devuelve null si el ServletContext no puede devolver un RequestDispatcher.
- **void log(String mensaje).** Escribe el mensaje especificado a un archivo de bitácora para servlets, usualmente una bitácora de eventos. El nombre y tipo del archivo de bitácora es específico del contenedor de servlets.
- **void log(String message, Throwable throwable).** Escribe un mensaje de explicación y un trazado de pila en el archivo de bitácora del servidor. El nombre y tipo del archivo de bitácora es específico del contenedor de servlets, usualmente una bitácora de eventos.
- **String getRealPath(String ruta).** Devuelve una cadena de texto conteniendo la ruta real para una ruta virtual. Por ejemplo, para la ruta "/index.html" devuelve la ruta absoluta del archivo en el sistema de archivos del servidor que corresponde a la solicitud "http://host/rutaDelContexto/index.html", donde rutaDelContexto es la ruta de contexto de este ServletContext. La ruta real devuelta está en un formato apropiado para la computadora y sistema operativo en que el contenedor de servlets se ejecuta. Este método devuelve null si el contenedor de servlets no puede obtener la ruta real.
- **String getServerInfo().** Devuelve el nombre y versión del contenedor de servlets en que se ejecuta el servlet. La forma de la cadena devuelta es nombredelservidor/númerodeversión. Pro ejemplo, JavaServer Web Development Kit puede devolver la cadena "JavaServer Web Dev Kit/ 1.0". El contenedor de servlets puede devolver otra información adicional después de la cadena primaria en paréntesis, por ejemplo, "JavaServer Web Dev Kit/1.0 (JDK 1.1.6; Windows NT 4.0 x86)".

- **String getInitParameter(String nombre).** Devuelve una cadena que contiene el valor, dentro del contexto, que corresponde al parámetro de inicialización con el nombre indicado, o null si el parámetro no existe. Este método puede hacer disponible información de información útil en toda la aplicación web. Por ejemplo, puede proporcionar la dirección de correo electrónico del administrador del sitio.
- **java.util.Enumeration getInitParameterNames().** Devuelve los nombres de los parámetros de inicialización como un objeto Enumeration que contiene objetos String, o como una Enumeration sin elementos cuando el contexto no tiene parámetros de inicialización.
- **Object getAttribute(String nombre).** Devuelve un atributo del contenedor correspondiente al nombre indicado, o bien null si no hay un atributo con ese nombre. Un atributo permite proporcionar información adicional que no es proporcionada por esta interfaz. Los atributos son específicos de cada tipo de servidor. El atributo en devuelto como como un Object o alguna subclase. Los nombres de atributos deben seguir la misma convención que los nombres de paquetes. La especificación de servlets, reserva los nombres que inician con “java.”, “javax.” y “sun.”.
- **java.util.Enumeration getAttributeNames().** Devuelve un los nombres de los atributos disponibles para este contexto de servidores como un objeto Enumeration que contiene objetos String.
- **void setAttribute(String nombre, Object objeto).** Liga un objeto a un cierto nombre de atributo en este contexto de servidor. Si el nombre especificado ya está en uso por algún atributo, este método reemplaza el atributo con el nuevo objeto. Si hay observadores configurados en el ServletContext, el contenedor les notifica los cambios. Si se pasa un valor null, el efecto es el mismo que invocar a removeAttribute(). Los nombres de atributos deben seguir la misma convención que los nombres de paquetes. La especificación de servlets, reserva los nombres que inician con “java.”, “javax.” y “sun.”.
- **void removeAttribute(String nombre).** Saca el atributo con el nombre indicado del contexto del contexto de servlets. Después de sacarlo, las siguientes invocaciones de getAttribute(String) para recuperar el valor del atributo, devuelven null. Si hay observadores configurados en el ServletContext, el contenedor les notifica los cambios.
- **String getServletContextName().** Devuelve el nombre de esta aplicación web, correspondiente a este ServletContext tal y como se especifica en el descriptor de despliegue para esta aplicación web en el elemento **display-name**.

2.4. La Interfaz ServletConfig.

public interface **javax.servlet.ServletConfig**.

Un objeto usado por un contenedor de servlets para proporcionar información a un servlet durante su inicialización.

2.4.1. Métodos de la Interfaz ServletConfig.

- **String getServletName().** Devuelve el nombre de esta instancia de servlet. El nombre se puede proporcionar mediante administración del servidor, asignado en el descriptor de despliegue de la aplicación, o bien, es para una instancia de servlet no registrado (y por lo tanto sin nombre) se utiliza el nombre de la clase del servlet.
- **ServletContext getServletContext().** Devuelve una referencia al ServletContext en el que se ejecuta el servlet que invoca este método.
- **String getInitParameter(String nombre).** Devuelve una cadena que contiene el valor del parámetro de inicialización con el nombre indicado o null si el parámetro no existe.
- **java.util.Enumeration getInitParameterNames().** Devuelve los nombres de los parámetros de inicialización como un objeto Enumeration que contiene objetos String, o como una Enumeration sin elementos cuando el servlet no tiene parámetros de inicialización.

2.5. La Interfaz Servlet.

public interface **javax.servlet.Servlet**.

Un servlet es un pequeño programa de Java que se ejecuta dentro de un servidor web. Los servlets reciben y responden a solicitudes de clientes web, usualmente usando HTTP (HyperText Transfer Protocol, en español, Protocolo de Transferencia de HiperTexto).

Esta interfaz define los métodos que todo servlet debe implementar y le permiten ser inicializado, dar servicio a solicitudes y remover un servicio del servidor. Se conocen como métodos del ciclo de vida y se invocan en el siguiente orden:

1. Se construye el servlet y posteriormente se inicializa con el método **init**.
2. Se maneja cualquier llamada enviada desde los clientes al el método **service**.
3. El servlet se saca de servicio, luego se destruye con el método **destroy**, el recolector de basura lo recoge y lo finaliza.

Adicionalmente esta interfaz proporciona el método **getServletConfig**, con el cual el servlet puede obtener información de inicialización y el método **getServletInfo**, que permite al servlet devolver información básica sobre él, tal como autor, versión y copyright.

Para implementar esta interfaz de puede escribir un servlet genérico que extienda **javax.servlet.GenericServlet** o un servlet para HTTP que extienda **javax.servlet.http.HttpServlet**.

2.5.1. Métodos de la Interfaz Servlet.

- **void init(ServletConfig config) throws ServletException.** Invocado por el contenedor del servlet para indicarle que se ha puesto en servicio. El objeto config proporciona la configuración del servlet y sus parámetros de inicialización. Se invoca exactamente una vez después de instanciar el servlet. Este método debe completarse exitosamente antes de que el servlet pueda recibir cualquier solicitud. Un servlet no se puede poner en servicio si el método:

  1. Lanza una ServletException.
  2. No termina en el periodo de tiempo definido por el servidor web.

  Genera la siguiente excepción:
  - ServletException – si ocurre una excepción que interrumpa la operación normal del servlet.
- **ServletConfig getServletConfig().** Devuelve el objeto ServletConfig, que contiene los parámetros de inicialización para este servlet. El objeto ServletConfig devuelto es el que se pasa al método **init**. Las implementaciones de esta interfaz son responsables de almacenar el objeto ServletConfig para que se pueda devolver. La clase GenericServlet, que implementa esta interfaz, también hace esto.
- **void service(ServletRequest solicitud, ServletResponse respuesta) throws ServletException, java.io.IOException.** Invocado por el contenedor del servlet para permitir que el servlet procese solicitudes. Este método solo se invoca después de que el método **init** se ha completado exitosamente. El código de estado de la respuesta siempre se debe asignar cuando el servlet lance o envíe un error. Los servlets se ejecutan típicamente en servidores multihilo, por lo que deben manejar solicitudes concurrentes y sincronizar el acceso a recursos compartidos, que incluyen datos en memoria, tales como variables de instancia o de clase y objetos externos, como archivos, conexiones a bases de datos y conexiones de red. Genera las siguientes excepciones:
  - java.io.IOException – si ocurre una excepción de entrada o de salida.
  - ServletException – si ocurre una excepción que interrumpa la operación normal del servlet.
- **String getServletInfo().** Devuelve información sobre el servlet, tal como autor, versión y copyright. La cadena devuela por este método debe ser texto plano sin marcas de ningún tipo (como HTML, XML, etc.).
- **void destroy().** Invocado por el contenedor de servlets. Le indica al servlet que está fuera de servicio. Este método solo se invoca cuando todos los hilos dentro del método **service** han terminado o después de que el periodo de timeout ha expirado. Posteriormente el contenedor invoca este método, el cual **no** debe invocar al método de servicio otra vez. Este método proporciona al servlet una oportunidad para limpiar los recursos que posee (por ejemplo, memoria, archivos, hilo) y asegura que cualquier estado persistente se sincronice con el estado actual del servidor en memoria.

2.6. La Clase GenericServlet.

public abstract class **javax.servlet.GenericServlet**

implements **Servlet**, **ServletConfig**, **java.io.Serializable**.

Define un servlet genérico e independiente de protocolos. Para escribir un servlet de HTTP para usarse en la web, debe extenderse **HTTPServlet**. Para escribir un servlet genérico, es suficiente con implementar el método abstracto **service**. La clase **GenericServlet** proporciona versiones simples de los métodos del ciclo de vida **init** y **destroy**, así como de los métodos de la interfaz **ServletConfig**. También implementa el método **log**, declarado en la interfaz **ServletContext**.

2.6.1. Métodos de la clase GenericServlet.

- **public GenericServlet().** No hace nada. Toda la inicialización se realiza con los métodos **init**.
- **public void destroy().** Invocado por el contenedor de servlets. Le indica al servlet que está fuera de servicio. Especificado por el método **destroy** enl a interfaz **Servlet**.
- **public String getInitParameter(String nombre).** Devuelve una cadena que contiene el valor del parámetro de inicialización con el nombre indicado, o null si el parámetro no existe. Toma el valor del objeto **ServletConfig** del servlet. Especificado por el método **getInitParameter** en la interfaz **ServletConfig**.
- **public java.util.Enumeration getInitParameterNames().** Devuelve los nombres de los parámetros de inicialización como un objeto Enumeration que contiene objetos String, o como una Enumeration sin elementos cuando el servlet no tiene parámetros de inicialización. Toma los nombres del objeto ServletConfig del servlet. Especificado por el método **getInitParameterNames** en la interfaz **ServletConfig**.
- **public ServletConfig getServletConfig().** Devuelve el objeto **ServletConfig** que inicializa este servlet. Especificado por el método **getServletConfig** en la interfaz Servlet.
- **public ServletContext getServletContext().** Devuelve una referencia la objeto **ServletContext** en que se ejecuta este servlet. Toma el valor del objeto **ServletConfig** de este servlet. Especificado por el método **getServletConfig** en la interfaz **ServletConfig**.
- **public String getServletInfo().** Devuelve información acerca del servlet. Por defecto este método devuelve una cadena vacía. Se sobreescribe este método para devolver información significativa. Especificado por el método **getServletInfo** en la interfaz **Servlet**.
- **public void init(ServletConfig config) throws ServletException.** Invocado por el contenedor del servlet para indicarle que se ha puesto en servicio. Esta implementación almacena el objeto **ServletConfig** que recibe del contenedor para un uso posterior. Al sobreescribir este método

siempre debe invocarse **super.init(config)**. Especificado por el método **init** en la interfaz **Servlet**. Genera la siguiente excepción:

- ServletException – si ocurre una excepción que interrumpa la operación normal del servlet.

- **public void init() throws ServletException.** Un método que evita invocar a **super.init(config)**. Simplemente se sobreescribe este método y es invocado automáticamente por **GenericServlet.init(ServletConfig config)**. El objeto **ServletConfig** se puede obtener con el método **getServletConfig()**. Genera la siguiente excepción:
  - ServletException – si ocurre una excepción que interrumpa la operación normal del servlet.
- **public void log(String mensaje).** Escribe el mensaje especificado al archivo bitácora, indicando el nombre del servlet.
- **public void log(String mensaje, Throwable t).** Escribe un mensaje de explicación y un trazado de pila al archivo de bitácora del servidor, indicando el nombre del servlet.
- **public abstract void service(ServletRequest solicitud, ServletResponse respuesta) throws ServletException, java.io.IOException.** Invocado por el contenedor del servlet para permitir que el servlet procese solicitudes. Se declara abstracto para que sus subclases, por ejemplo **HttpServlet**, deban sobreescribirlo. Especificado por el método **service** en la interfaz **Servlet**. Genera las siguientes excepciones:
  - java.io.IOException – si ocurre una excepción de entrada o de salida.
  - ServletException – si ocurre una excepción que interrumpa la operación normal del servlet.
- **public String getServletName().** Devuelve el nombre de esta instancia de servlet. Especificado por el método **getServletName** en la interfaz **ServletConfig**.

2.7. La Clase HTTPServlet.

public abstract class **javax.servlet.http.HttpServlet** extends **GenericServlet**

implements **java.io.Serializable**

Clase base que permite modificarse para construir servlets que se pueden añadir a un sitio web. Una subclase, debe sobreescribir al menos un método, que normalmente es alguno de los siguientes:

- **doGet**, si soporta solicitudes GET de HTTP.
- **doPost**, para solicitudes POST de HTTP.
- **doPut**, para solicitudes PUT de HTTP.
- **doDelete**, para solicitudes DELETE de HTTP.
- **init** y **destroy**, para administrar recursos utilizados durante la vida del servlet.

- **getServletInfo**, que proporciona información sobre el servlet.

Hay otros métodos que casi nunca se sobreescriben, como son; **service**, **doOptions**, y **doTrace**.

Los servlets se ejecutan típicamente en servidores multihilo, por lo que deben manejar solicitudes concurrentes y sincronizar el acceso a recursos compartidos, que incluyen datos en memoria, tales como variables de instancia o de clase y objetos externos, como archivos, conexiones a bases de datos y conexiones de red.

2.7.1. Métodos de la Clase HTTPServlet.

- **public HttpServlet()**. No hace nada.
- **protected void doGet(HttpServletRequest solicitud, HttpServletResponse respuesta) throws ServletException, java.io.IOException.** Cuando se sobreescribe, permite que un servlet procese solicitudes GET y hay que realizar en orden los siguientes pasos:
    1. Leer los datos de la solicitud.
    2. Escribir los encabezados de la respuesta.
    3. Definir el tipo de contenido y la codificación.
    4. Obtener el objeto para escribir la respuesta, que puede ser un PrintWriter o un OutputStream.
    5. escribir la respuesta.

    Genera las siguientes excepciones:

    - java.io.IOException – si se detecta un error de entrada o de salida cuando el servidor procesa la solicitud.
    - ServletException – si la solicitud no se puede procesar.

    El método GET no debe generar cambios en el servidor o la base de datos utilizada y debe poder repetirse sin inducir cambios. Si la solicitud no está bien formada, devuelve un mensaje de HTTP “Bad Request”
- **protected long getLastModified(HttpServletRequest solicitud).** Devuelve la hora en la que los datos devueltos cambiaron por última vez, a partir de la media noche del primero de enero de 1970 GMT. Si este valor se desconoce, devuelve un número negativo (valor por defecto). Al sobreescribir este método, los navegadores y chaches de proxy trabajan más efectivamente, reduciendo la carga del servidor y recursos de red.
- **protected void doHead(HttpServletRequest solicitud, HttpServletResponse respuesta) throws ServletException, java.io.IOException.** Recibe una solicitud HEAD de HTTP proveniente de alguno de los métodos protegidos de servicio y maneja la respuesta. Normalmente el cliente envía la solicitud HEAD cuando solo desea ver los encabezados de la

respuesta, tales como Content-Type o Content-Length. Si la solicitud no está bien formada, devuelve un mensaje de HTTP "Bad Request".

- **protected void doPost(HttpServletRequest solicitud, HttpServletResponse respuesta) throws ServletException, java.io.IOException.** Cuando se sobreescribe, permite que un servlet procese solicitudes POST y hay que realizar en orden los siguientes pasos:

    1. Leer los datos de la solicitud.
    2. Escribir los encabezados de la respuesta.
    3. Definir el tipo de contenido y la codificación.
    4. Obtener el objeto para escribir la respuesta, que puede ser un PrintWriter o un OutputStream.
    5. escribir la respuesta.

    Genera las siguientes excepciones:

    - java.io.IOException – si se detecta un error de entrada o de salida cuando el servidor procesa la solicitud.
    - ServletException – si la solicitud no se puede procesar.

    El método POST permite al cliente enviar cantidades ilimitadas de datos y realizar cambios en el servidor. Si la solicitud no está bien formada, devuelve un mensaje de HTTP "Bad Request"

- **protected void doPut(HttpServletRequest solicitud, HttpServletResponse respuesta) throws ServletException, java.io.IOException.** Cuando se sobreescribe, permite que un servlet procese solicitudes PUT y hay que dejar intacto el contenido de los encabezados. Si el método no puede manejar el contenido, debe generar un mensaje de error (HTTP 501 - Not Implemented) y descartar la solicitud. Genera las siguientes excepciones:

    - java.io.IOException – si se detecta un error de entrada o de salida cuando el servidor procesa la solicitud.
    - ServletException – si la solicitud no se puede procesar.

    El método PUT permite al cliente colocar un archivo en el servidor y es similar a enviar un archivo por FTP. Si la solicitud no está bien formada, devuelve un mensaje de HTTP "Bad Request"

- **protected void doDelete(HttpServletRequest solicitud, HttpServletResponse respuesta) throws ServletException, java.io.IOException.** Cuando se sobreescribe, permite que un servlet procese solicitudes DELETE. Genera las siguientes excepciones:

    - java.io.IOException – si se detecta un error de entrada o de salida cuando el servidor procesa la solicitud.
    - ServletException – si la solicitud no se puede procesar.

El método DELETE permite al cliente eliminar un documento o página web del servidor. Si la solicitud no está bien formada, devuelve un mensaje de HTTP "Bad Request"

- **protected void doOptions(HttpServletRequest solicitud, HttpServletResponse respuesta) throws ServletException, java.io.IOException.** Invocado por el servidor (por medio del método service) para permitir que un servlet procese solicitudes OPTIONS. Genera las siguientes excepciones:
  - java.io.IOException – si se detecta un error de entrada o de salida cuando el servidor procesa la solicitud.
  - ServletException – si la solicitud no se puede procesar.

  El método OPTION permite al cliente determinar que métodos soporta el servidor.
- **protected void doTrace(HttpServletRequest solicitud, HttpServletResponse respuesta) throws ServletException, java.io.IOException.** Invocado por el servidor (por medio del método service) para permitir que un servlet procese solicitudes TRACE. Genera las siguientes excepciones:
  - java.io.IOException – si se detecta un error de entrada o de salida cuando el servidor procesa la solicitud.
  - ServletException – si la solicitud no se puede procesar.
- El método TRACE devuelve los encabezados enviados con la solicitud TRACE AL CLIENTE, para que puedan usarse en depuración.
- **protected void service(HttpServletRequest solicitud, HttpServletResponse respuesta) throws ServletException, java.io.IOException.** Recibe solicitudes HTTP estándar y las despacha hacia los métodos que empiezan con "do". Genera las siguientes excepciones:
  - java.io.IOException – si se detecta un error de entrada o de salida cuando el servidor procesa la solicitud.
  - ServletException – si la solicitud no se puede procesar.
- **public void service(ServletRequest solicitud, ServletResponse respuesta) throws ServletException, java.io.IOException.** Despacha las solicitudes del cliente al método service protegido. Genera las siguientes excepciones:
  - java.io.IOException – si se detecta un error de entrada o de salida cuando el servidor procesa la solicitud.
  - ServletException – si la solicitud no se puede procesar.

2.8. La Interfaz RequestDispatcher.

public interface **javax.servlet.RequestDispatcher**

Define un objeto que recibe peticiones del cliente y las envía a cualquier recurso (como un servlet, un archivo de HTML, o un archivo de JSP) en el servidor. El contenedor de servlets crea el objeto de RequestDispatcher, que es utilizado como una envoltura alrededor de un recurso de servidor situado en una ruta particular o identificado por un nombre particular.

2.8.1. Métodos de la Interfaz RequestDispatcher.

- **void forward(ServletRequest solicitud, ServletResponse respuesta) throws ServletException, java.io.IOException.** Transfiere una petición de un servlet a otro recurso (un servlet, un archivo de JSP, o un archivo de HTML) en el servidor. Este método permite que un servlet realice el procesamiento procesamiento preliminar de una petición y que otro recurso genere la respuesta. Para un RequestDispatcher obtenido vía getRequestDispatcher(), el objeto de ServletRequest tiene sus elementos de ruta y parámetros, ajustados para emparejar la ruta del recurso al que será reenviada la solicitud. La transferencia debe ser realizada antes de que la generación de la respuesta hacia el cliente haya completada (es decir, antes de que la salida de la respuesta haya recibido el método flush). Si la respuesta ya ha sido concluida, este método genera ua IllegalStateException. La salida disponible en el búfer de la respuesta es descartada automáticamente antes del reenvío. Los parámetros de petición y respuesta deben ser o los mismos objetos fueron pasados como parámetros al método del servicio invocado en el servlet o bien, subclases de ServletRequestWrapper o de ServletResponseWrapper que los envuelven. Genera las excepciones:
    - ServletException – Si el recurso hacia donde se realiza la transferencia genera esta excepción.
    - java.io.IOException – Si el recurso hacia donde se realiza la transferencia genera esta excepción.
    - java.lang.IllegalStateException – Si la respuesta ya ha sido completada.
- **void include(ServletRequest solicitud, ServletResponse respuesta) throws ServletException, java.io.IOException.** Incluye el contenido de un recurso (un servlet, un archivo de JSP, o un archivo de HTML) en la respuesta. El objeto de ServletResponse mantiene sin cambio sus elementos de ruta y parámetros. El servlet incluido no puede cambiar el código de estatus de respuesta ni encabezamientos fijos; cualquier intento para hacer un cambio es ignorado. Los parámetros de petición y respuesta deben ser o los mismos objetos que fueron pasados como al método de servicio invocado en el servlet o son subclases de ServletRequestWrapper o de ServletResponseWrapper que los envuelven.

2.9. La Interfaz ServletRequest.

public interface **javax.servlet.ServletRequest**

Define un objeto para proporcionar a cliente información de una petición a un servlet. El contenedor de servlets crea un objeto de ServletRequest y lo pasa como un argumento al método del servicio del servlet. Un objeto de ServletRequest proporciona datos incluyendo los nombre de los parámetros y sus valores, los atributos, y objeto InputStream. Las interfaces que extienden ServletRequest pueden proporcionar datos adicionales que sean específicos del protocolo (por ejemplo, los datos de HTTP proporcionados por HttpServletRequest).

2.9.1. Métodos de la Interfaz ServletRequest.

- **Object getAttribute(String nombre).** Devuelve el valor del atributo que corresponde al nombre como un Objeto, o nulo no existe ningún atributo con el nombre indicado. Los atributos pueden ser asignados dos maneras.
    - El contenedor de servlets puede asignar los atributos para hacer disponible información personalizada sobre una petición. Por ejemplo, para peticiones hechas utilizando HTTPS, el atributo javax.servlet.request.X509Certificate puede ser utilizado para recuperar información en el certificado del cliente.
    - Los atributos también pueden ser asignados programáticamente utilizando **setAttribute(String, Object)**. Esto permite información añadir información a una petición antes de invocar a RequestDispatcher.

  Los nombres de atributos deben seguir la misma convención que los nombres de paquetes. La especificación de servlets, reserva los nombres que inician con “java.”, “javax.” y “sun.”.
- **java.util.Enumeration getAttributeNames().** Devuelve un objeto de tipo Enumeration que contiene los nombres de los atributos disponibles para esta petición, o bien un Enumeration vacío si la solicitud no tiene atributos disponibles.
- **String getCharacterEncoding().** Devuelve el nombre de la codificación de caracteres utilizada en el cuerpo de esta petición. Este método devuelve null si la petición no especifica una codificación de caracteres.
- **void setCharacterEncoding(String codificación) throws java.io.UnsupportedEncodingException.** Sobreescribe el nombre de la codificación de caracteres utilizada en el cuerpo de esta petición. Este método debe ser llamado antes de leer los parámetros de la petición o de leer la entrada que utilizando getReader (); de otro modo, no tiene efecto. **Es necesario invocar a este método cuando la solicitud utiliza la codificación UTF-8**. Genera la excepción:
    - java.io.UnsupportedEncodingException – si este ServletRequest está todavía en un estado donde la codificación de caracteres se puede asignar, pero la codificación especificada es inválida.
- **int getContentLength().** Devuelve la longitud, en bytes, del cuerpo de la petición y que está disponible por medio del InputStream, o -1 si la longitud no se conoce. Para servlets de HTTP, es mismo valor de la variable CGI conocida como CONTENT_LENGTH.

- **String getContentType().** Vuelve el de tipo MIME del cuerpo de la petición, o null si el tipo no es conocido. Para servlets de HTTP, es mismo valor de la variable CGI conocida como CONTENT_TYPE.
- **ServletInputStream getInputStream() throws java.io.IOException.** Recupera el cuerpo de la petición como datos binarios que utilizan un ServletInputStream. Este método o getReader () puede ser llamado a leer el cuerpo, perno los dos. Genera las siguientes excepciones:
    - java.lang.IllegalStateException – si el método getReader() ya se ha invocado en esta solicitud.
    - java.io.IOException – si ocurre un error de entrada/salida.
- **String getParameter(String nombre).** Devuelve el valor del parámetro de la solicitud que corresponda al nombre, o null, si no hay ningún parámetro que corresponda al nombre. Los parámetros solicitados son información extra enviada con la petición. Para servlets de HTTP, los parámetros están contenidos en la cadena de consulta (query) o en los datos enviados por una forma. Este método solo debe usarse cuando es seguro que el parámetro tiene cuando mucho un solo valor. Si el parámetro tiene más de un valor, el uso de este parámetro devuelve el primer valor del arreglo devuelto por el método, **getParameterValues(String)**, que es más adecuado en este caso. Cuando el parámetro es enviado en el cuerpo de la solicitud, como en el caso de una solicitud POST de HTTP, el uso de getInputStream() o getReader() puede interferir con la ejecución de este método.
- **java.util.Enumeration getParameterNames().** Devuelve una referencia de tipo Enumeration de objetos tipo String que contienen los nombres de los parámetros contenidos en esta petición. Si la petición no tiene los parámetros, el método devuelve una Enumeration vacía.
- **String[] getParameterValues(String nombre).** Devuelve un arreglo de objetos tipo String con todos los valores que el parámetro del nombre indicado tiene en la petición, o nulo si el parámetro no existe.
- **java.util.Map getParameterMap().** Devuelve un java.util.Map de los parámetros para esta petición. Los parámetros de petición son datos extra enviados con la petición. Para servlets de HTTP, los parámetros están contenidos en la cadena de consulta (query) o en los datos enviados por una forma.
- **String getProtocol().** Devuelve el nombre y la versión del protocolo que usa la petición en la forma “protocolo/númeroMajorDeVersion.númeroMenorDeVersion”, por ejemplo, “HTTP/1.1”. Para servlets de HTTP, el valor vuelto es igual que el valor de la variable de CGI conocida como SERVER_PROTOCOL.
- **String getScheme().** Devuelve el nombre del esquema usado para realizar esta petición, por ejemplo, “http”, “http”, o “ftp”. Los esquemas tienen reglas diferentes para construir URLs, como se indica en RFC 1738.

- **String getServerName().** Devuelve el nombre de host del servidor al que la petición fue enviada. Es el valor de la parte antes de ":" en el valor de encabezado “Host”, si hay alguno, o bien, el nombre resuelto del servidor, o la dirección IP del servidor.
- **int getServerPort().** Devuelve el número del puerto al que la petición fue enviada. Es el valor de la parte después ":" en el valor de encabezado “Host”, si hay alguno, o bien, el puerto del servidor donde la conexión del cliente fue aceptada.
- **java.io.BufferedReader getReader() throws java.io.IOException.** Recupera el cuerpo de la petición como datos de tipo caracter que utilizan un BufferedReader. El lector traduce los datos según la codificación de caracteres utilizado en el cuerpo. Este método o getInputStream() puede ser llamados para leer el cuerpo de la solicitud, pero no los dos. Genera las siguientes excepciones:
    - UnsupportedEncodingException – si el conjunto de codificación de caracteres utilizado no es soportado y el texto no puede ser decodificado.
    - java.lang.IllegalStateException – si el método getInputStream() ya se ha invocado en esta petición.
    - java.io.IOException – si ocurre una excepción de entrada o salida.
- **String getRemoteAddr().** Devuelve la dirección del Internet Protocol (IP) del cliente o del último proxy que envió la petición. Para servlets de HTTP, es el mismo valor que la variable de CGI conocida como REMOTE_ADDR.
- **String getRemoteHost().** Devuelve el nombre completamente calificado del cliente o del últimos proxy que envió la petición. Si el motor no puede, o elige no resolver el nombre del host (para mejorar el desempeño), este método regresa una cadena de la forma con puntos de la dirección de IP. Para servlets de HTTP, es el mismo que la variable de CGI conocida como REMOTE_HOST.
- **void setAttribute(String nombre, Object valor).** Almacena un atributo en esta solicitud. Los atributos se eliminan entre peticiones. Este método se usa más comúnmente en combinación con RequestDispatcher. Los nombres de atributos deben seguir la misma convención que los nombres de paquetes. Los nombres que inician con “java.”, “javax.” y “com.sun.” están reservados. Si el valor recibido es null, el efecto es el mismo que invocar a removeAttribute(java.lang.String). Cuando la solicitud se despacha hacia servlets que residen en una aplicación web diferente por medio de RequestDispatcher, el objeto enviado por este método puede que no sea recuperada correctamente en el servlet que realiza la invocación.
- **void removeAttribute(String nombre).** Quita un atributo de la solicitud. Ente método no se necesita mucho porque los atributos solo se mantienen mientras la solicitud se está procesando. Los nombres de atributos deben seguir la misma convención que los nombres de paquetes. Los nombres que inician con “java.”, “javax.” y “com.sun.” están reservados.

- **java.util.Locale getLocale().** Devuelve la localizaión preferida que el cliente acapta para su contenido, basado en el encabezado " Accept-Language". Si la petición del cliente no proporciona este encabezado, se devuelve la localización predeterminada del servidor.
- **java.util.Enumeration getLocales().** Devuelve un objeto de tipo Enumeration de objetos tipo Locale indicando, en orden decreciente de preferencia, las localizaciones aceptadas por el cliente, basado en el encabezado " Accept-Language". Si la petición del cliente no proporciona este encabezado, se devuelve un objeto de tipo Enumeration que contiene un solo Locale, que es la localización predeterminada del servidor.
- **boolean isSecure().** Devuelve un booleano que indica si la solicitud utiliza un canal seguro, por ejemplo HTTPS.
- **RequestDispatcher getRequestDispatcher(String ruta).** Devuelve un objeto RequestDispatcher que actúa como envoltura para el recurso localizado en la ruta indicada. Los objetos RequestDispatcher se pueden utilizar para transferir una solicitud al recurso o incluir el recurso en una solicitud. El recurso puede ser tanto estático como dinámico. La ruta especificada puede ser relativa, aunque no puede extenderse fuera del contexto de servlets actual. Si la ruta comienza con "/" se interpreta como relativa a la raíz del contexto. Este método devuelve null si el contenedor de servlets no puede devolver un RequestDispatcher. La diferencia entre este método y ServletContext.getRequestDispatcher(java.lang.String) es que este método puede tomar una ruta relativa.
- **int getRemotePort().** Regresa el puerto de origen del Internet Protocolo (IP) del cliente o del último proyx que envió la solicitud.
- **String getLocalName().** Devuelve una cadena de texto que contiene del nombre de host en el Internet Protocol (IP) que recibió la solicitud.
- **String getLocalAddr().** Regresa la dirección del Internet Protocol (IP) que recibió la solicitud.
- **int getLocalPort().** Regresa el número de puerto del Internet Protocol (IP) que recibió la solicitud.

2.10. La Interfaz HttpServletRequest.

public interface **javax.servlet.http.HttpServletRequest** extends **ServletRequest**

Extiende la interfaz ServletRequest para proporcionar información sobre la solicitud a servlets de HTTP. El contenedor de servlets crea un objeto HttpServletRequest y lo pasa como argumento a los métodos de servicio de los servlet (doGet, doPost, etc.).

2.10.1. Campos de la Interfaz HttpServletRequest.

- **static final java.lang.String BASIC_AUTH.** Identificador para la autenticación básica. Valor: "BASIC".
- **static final java.lang.String FORM_AUTH.** Identificador para la autenticación por medio de forma. Valor: "FORM".
- **static final java.lang.String CLIENT_CERT_AUTH.** Identificador para la autenticación por medio de certificados del cliente. Valor: "CLIENT_CERT".
- **static final java.lang.String DIGEST_AUTH.** Identificador para la autenticación digerida (digest authentication). Valor: "BASIC".

2.10.2. Métodos de la Interfaz HttpServletRequest.

- **String getAuthType().** Devuelve el nombre del esquema de autenticación usado para proteger el servlet, que puede ser BASIC_AUTH, FORM_AUTH, CLIENT_CERT_AUTH, DIGEST_AUTH (estos valores se pueden comparar por medio del operador == con los campos de esta clase). Todos los contenedores de servlets soportan las autenticaciones básica, por medio de forma y por medio de certificado de cliente; adicionalmente pueden soportar la autenticación digerida (digest authentication). Si en servlet no está autenticado se regresa el valor null. Es el mismo que la variable de CGI conocida como AUTH_TYPE.
- **Cookie[] getCookies().** Regresa un arreglo que contiene todos los objetos de tipo Cookie enviados por el cliente con esta solicitud, o bien, devuelve null en caso de no enviarse ninguna cookie.
- **long getDateHeader(String nombre).** Devuelve la fecha del encabezado de petición con el nombre indicado como un dato de tipo long que representa el número de milisegundos transcurridos desde el primero de enero de 1970 GMT. El nombre del encabezado no es sensible al uso de mayúsculas o minúsculas. Este método se utiliza para encabezados que contienen fechas, por ejemplo If-Modified-Since. Si la solicitud no contiene un encabezado con el nombre especificado, devuelve -1. Genera la siguiente excepción:
  - java.lang.IllegalArgumentException – si el valor del encabezado no se puede convertir en fecha.
- **String getHeader(String nombre).** Devuelve el valor del encabezado con el nombre indicado, como una cadena de texto. Si la solicitud no incluye un encabezado con el nombre indicado, devuelve null. Cuando hay múltiples encabezados con el mismo nombre, este método devuelve el primer encabezado en la petición. El nombre del encabezado no es sensible al uso de mayúsculas o minúsculas. Se puede utilizar este método con cualquier encabezado de solicitud.
- **java.util.Enumeration getHeaders(String nombre).** Devuelve una referencia de tipo Enumeration de objetos tipo String que contiene todos los valores del encabezado con el nombre indicado. Si la petición no incluye encabezados para el nombre especificado, el método

devuelve una Enumeration vacía. Algunos encabezados, tales como Accept-Language se pueden enviar como varios encabezados, cada uno de ellos con un valor diferente, en vez de enviar el encabezado como una lista separada por comas.

- **java.util.Enumeration getHeaderNames().** Devuelve una enumeración de todos los nombres de encabezados que contiene esta solicitud. Si la petición no contiene encabezados, el método devuelve una enumeración vacía. Algunos contenedores de servlets no permiten que los servlets accedan a los encabezados utilizando este método; en este caso, el método devuelve null.
- **int getIntHeader(String nombre).** Devuelve el valor del encabezado con el nombre indicado, como un entero. Si el encabezado no contiene un encabezado con el nombre especificado, devuelve -1. El nombre del encabezado no es sensible al uso de mayúsculas o minúsculas. Genera la siguiente excepción:
  - java.lang.NumberFormatException – si el valor del encabezado no se puede convertir a entero.
- **String getMethod().** Devuelve el nombre del método de HTTP con que se generó la petición; por ejemplo, GET, POST, o PUT. Es el mismo valor que la variable CGI conocida como REQUEST_METHOD.
- **String getPathInfo().** Devuelve cualquier información extra asociada con el URL enviada por el cliente cuando se generó la solicitud. Esta información aparece después de la ruta del servlet, pero precede la cadena de consulta (query) y empieza con un caracter "/". El método devuelve null si no aparece esta información. Es el mismo valor que la variable CGI conocida como PATH_INFO.
- **String getPathTranslated().** Devuelve cualquier información extra asociada con el URL que después de la ruta del servlet, pero precede la cadena de consulta (query) y la traduce a una ruta real. Es el mismo valor que la variable CGI conocida como PATH_TRANSLATED. El método devuelve null si no aparece esta información o si el servlet no puede traducir la ruta virtual a una ruta real. El contenedor web no decodifica esta cadena.
- **String getContextPath().** Devuelve la parte del URI de la petición que indica el contexto. Esta parte siempre aparece primero en el URI de una solicitud; inicia con un caracter "/", pero no termina con un caracter "/". Para servlets en el contexto por defecto (raíz), este mpetodo devuelve "". El contenedor web no decodifica esta cadena. Es posible que un contenedor de servlets coincida con un contexto en mas de una ruta de contexto. En tales casos, el método devuelve la ruta de contexto actual utilizada por la solicitud y puede diferir del la ruta devuelta por ServletContext.getContextPath(), que debe considerarse como la ruta de contexto primaria o preferida por la aplicación.
- **String getQueryString().** Devuelve la cadena de consulta (query) contenida en la URL de la solicitud, después de la ruta. Este método devuelve null si la URL no contiene una cadena de consulta. Es el mismo valor que la variable CGI conocida como QUERY_STRING.

- **String getRemoteUser().** Si el usuario está autenticado, devuelve el login del usuario que realiza la solicitud; en caso contrario, regresa null. Depende del navegador y del tipo de autenticación si el nombre de usuario se envía o no en con cada petición subsecuente. Es el mismo valor que la variable CGI conocida como REMOTE_USER.
- **boolean isUserInRole(String rol).** Devuelve false si el usuario no se ha autenticado. Si el usuario está autenticado, devuelve true si el usuario pertenece al rol indicado, o false si no pertenece.
- **java.security.Principal getUserPrincipal().** Devuelve un objeto de tipo java.security.Principal que contiene en nombre del usuario actualmente autenticado. Si el usuario no se ha autenticado, este método devuelve null.
- **String getRequestedSessionId().** Devuelve el identificador de sesión especificado por el cliente. Puede no ser el mismo identificador de la sesión válida actual para esta solicitud. Si el cliente no especifica un identificador de sesión, este método devuelve null.
- **String getRequestURI().** Devuelve la parte de la URL de la solicitud, desde el nombre del protocolo, hasta la cadena de consulta (query) en la primera línea de la solicitud HTTP. El contenedor web no decodifica esta cadena. Por ejemplo:

Primera Línea de la URL de la Solicitud	Valor Devuelto
POST /some/path.html HTTP/1.1	/some/path.html
GET http://foo.bar/a.html HTTP/1.0	/a.html
HEAD /xyz?a=b HTTP/1.1	/xyz

Para reconstruir una URL con esquema y host, se debe usar **HttpUtils.getRequestURL(javax.servlet.http.HttpServletRequest)**.

- **java.lang.StringBuffer getRequestURL().** Reconstruye la URL usada por el cliente para realziar la solicitud. La URL devuelta contiene protocolo, nombre del servidor, número de puerto y ruta en el servidor, pero no incluye los parámetros de la cadena de consulta. Si esta solicitud se ha reenviado usando **RequestDispatcher.forward(javax.servlet.ServletRequest, javax.servlet.ServletResponse)**, la ruta de servidor de esta Url reconstruida, debe reflejar la ruta usada para obtener el RequestDispatcher y no la ruta de servidor especificada inicialmente por el cliente. Como este método devuelve un StringBuffer y no una cdena, se puede modificar la URL fácilmente, por ejemplo, para agregarle los parámetros de consulta. Este método es útil para crear mensajes de redireccionamiento y reporte de errores.
- **java.lang.String getServletPath().** Devuelve la parte de la URL de la petición que invoca al servlet. Esta ruta empieza con "/" e incluye tanto el nombre del servlet o una ruta que llega hasta el servlet, pero no incluye ninguna información de ruta extra ni la cadena de consulta. Es el mismo valor que la variable CGI conocida como SCRIPT_NAME. Este método devuelve una

cadena vacía (“”) si el servlet usado para procesar esta solicitud se encontró usando el patrón “/*”.

- **HttpSession getSession(boolean crea).** Devuelve el objeto HttpSession actual asociado con esta petición o, si no hay sesión actual y crea vale true, devuelve una nueva sesión. Cuando crea es false y no hay una HttpSession válida, devuelve null. Para asegurar que esta sesión se mantiene adecuadamente, se debe invocar este método antes de que la respuesta se concluya (commited). Si el contenedor usa cookies para mantener la integridad de la sesión y se le solicita crear una nueva sesión cuando la respuesta ya se ha completado, se lanza una IllegalStateException.
- **HttpSession getSession().** Devuelve la sesión actual asociada con esta solicitud; si no existe, la crea.
- **boolean isRequestedSessionIdValid().** Devuelve true si el identificador de sesión solicitado es válido; en caso contrario, devuelve false.
- **boolean isRequestedSessionIdFromCookie().** Devuelve true si el identificador de sesión solicitado proviene de una cookie; en caso contrario, devuelve false.
- **boolean isRequestedSessionIdFromURL().** Devuelve true si el identificador de sesión solicitado proviene de una parte de la URL de la petición; en caso contrario, devuelve false.

2.11. La Interfaz ServletResponse.

public interface **javax.servlet.ServletResponse**

Define un objeto que ayuda a los servlets para mandar una respuesta al cliente. El contenedor de servlets crea un objeto ServletResponse y lo pasa como parámetros al método **service** del servlet.

Para enviar datos binarios en un cuerpo de respuesta MIME, se utiliza el ServletOutputStream devuelto por getOutputStream(). Para enviar caracteres se utiliza el objeto PrintWriter devuelto por getWriter(). Para mezclar datos binarios y cracteres, por ejemplo para crear una respuesta multipart, se emplea un ServletOutputStream y se controlan manualmente las secciones de caracteres.

El conjunto de caracteres del cuerpo de la respuesta MIME se puede especificar explícitamente usando los métodos setCharacterEncoding(String) y setContentType(String), o implícitamente en el método setLocale(java.util.Locale). Las especificaciones explícitas toman precedencia sobre las impícitas. Si no se especifica ningún conjunto de caracteres, se utiliza ISO-8859-1. Los métodos setCharacterEncoding, setContentType o setLocale deben invocarse antes de getWriter y antes de finalizar (commit) la respuesta para la codificación de caracteres que se empleará.

2.11.1. Métodos de la Interfaz ServletResponse.

- **String getCharacterEncoding().** Regresa el nombre de la codificación de caracteres (MIME charset) utilizado para el cuerpo enviado por esta respuesta. La codificación de caracteres puede haber sido especificada utilizando explícitamente los métodos setCharacterEncoding(String) o setContentType(String), o utilizando implícitamente el método setLocale (Locale). Las especificaciones explícitas tienen prioridad sobre las especificaciones implícitas. Las llamadas hechas a estos métodos después de que getWriter haya sido llamado o después de que la respuesta haya sido finalizada no tienen efecto en la codificación de caracteres. Si ninguna codificación de caracteres ha sido especificada, se devuelve ISO-8859-1.
- **String getContentType().** Devuelve el tipo de contenido usado para el cuerpo MIME enviado en esta respuesta. Se debe especificar utilizando setContentType(String) antes de que la respuesta sea finalizada. Si ningún tipo de contenido se ha especificado, este método devuelve null. Si el tipo de contenido se ha especificado, y el la codificación de caracteres se ha definido explícitamente o implícitamente, como se describe en getCharacterEncoding() o en getWriter(), el parámetro "charset" se incluye en la cadena devuelta. Si no se especifica ninguna codificación, se omite el parámetro "charset".
- **ServletOutputStream getOutputStream() throws java.io.IOException.** Devuelve un ServletOutputStream adecuado para escribir datos binarios en la respuesta. El contenedor de servlets no codifica los datos binarios. La invocación del método flush() en el ServletOutputStream finaliza la respuesta. Se puede utilizar este método o getWriter() para escribir el cuerpo de la respuesta, pero no ambos. Genera las siguientes excepciones:
  - java.lang.IllegalStateException – si el método getWriter ya se ha invocado en esta respuesta.
  - java.io.IOException – si se genera una excepción de entrada o de salida.
- **java.io.PrintWriter getWriter() throws java.io.IOException.** Devuelve un objeto PrintWriter que envía caracteres de texto al cliente. El PrintWriter utiliza la codificación devuelta por getCharacterEncoding(). Si la codificación de la respuesta no se ha especificado como se describe en getCharacterEncoding(), getWriter la actualiza a ISO-8859-1. La invocación de flush() en el PrintWriter finaliza la respuesta. Se puede utilizar este método o getOutputStream() para escribir el cuerpo de la respuesta, pero no ambos. Genera las siguientes excepciones:
  - UnsupportedEncodingException – si la codificación devuelta por getCharacterEncoding no se puede usar.
  - java.lang.IllegalStateException – si el método getOutputStream ya se ha invocado en esta respuesta.
  - java.io.IOException – si se genera una excepción de entrada o de salida.
- **void setCharacterEncoding(String codificación).** Asigna la codificación (MIME charset) de la respuesta que se está enviando al cliente, por ejemplo UTF-8. Si esta codificación ya se ha asignado con setContentType(String) o setLocale(java.util.Locale), este método la modifica. La

invocación de setContentType(String) con la cadena “text/html” y la invocación de este método con la cadena “UTF-8”, es equivalente a invocar setContentType con la cadena “text/html; charset=UTF-8”. Este método se puede invocar repetidamente para modificar la codificación y no tiene efecto si se invoca después de que getWriter ha sido llamado o después de que la respuesta se ha finalizado. Los contenedores deben comunicar al cliente la codificación utilizada para la respuesta de servlets si el protocolo proporciona una forma para hacerlo. En el caso de HTTP, la codificación de caracteres se comunica como parte del encabezado Content-Type para textos. La codificación no se puede comunicar en encabezados HTTP si el servlet no especifica un tipo de contenido; aún así se usa para codificar el testo escrito por el servlet al writer de la respuesta.

- **void setContentLength(int longitud).** Asigna la longitud del cuerpo del contenido de la respuesta. En servlets HTTP este método asigna el encabezado Content-Length de HTTP.
- **void setContentType(String tipo).** Asigna el tipo del contenido de la respuesta devuelta al cliente, cuando la respuesta no ha sido finalizada aún. El tipo de contenido proporcionado puede incluir una especificación de codificación de caracteres, por ejemplo. “text/html; charset=UTF-8”. La codificación de caracteres de la respuesta solo e asigna a partir del tipo de contenido proporcionado, si este método se invoca antes de que se llame getWriter. Este método se puede invocar repetidamente para modificar el tipo de contenido y la codificación de caracteres. Este método no tiene efecto si se invoca después de que la respuesta se ha finalizado. Los contenedores deben comunicar al cliente el tipo de contenido y la codificación utilizada para la respuesta de servlets si el protocolo proporciona una forma para hacerlo. En el caso de HTTP, se comunica como parte del encabezado Content-Type.
- **void setBufferSize(int tamaño).** Asigna el tamaño de buffer preferido para generar el cuerpo de la respuesta. El contenedor de servlets usará un buffer al menos tan grande como el tamaño solicitado. El tamaño de buffer actual se puede obtener usando getBufferSize. Un tamaño de buffer mayor permite que se escriba mpas contenido antes de que cualquier dato se envíe, y por lo tanto proporciona al servlet más tiempo para asignar apropiadamente el códigos de estado y los encabezados. Un tamaño de buffer más pequeño disminuye la carga de memoria del servidor y permite que los clientes empiecen a recibir datos más rápidamente. Este método debe invocarse antes de de que se escriba cualquier contenido en el cuerpo de la respuesta. Genera la siguiente excepción:
    - java.lang.IllegalStateException – si este método se invoca después de que se ha escrito algún contenido o de que se ha finalizado la respuesta.
- **int getBufferSize().** Devuelve el tamaño de buffer actual. Si se utiliza buffer, este método devuelve 0.
- **void flushBuffer() throws java.io.IOException.** Obliga a que cualquier contenido en el buffer sea enviado al cliente. Una invocación a esta método automáticamente finaliza la respuesta, escribiendo el código de estado y los encabezados.

- **void resetBuffer().** Limpia el contenido del buffer en la respuesta sin limpiar los encabezados ni el código de estado. Si la respuesta se ha finalizado, este método lanza una IllegalStateException.
- **boolean isCommitted().** Devuelve true si la respuesta se ha finalizado; en caso contrario, devuelve false. Una respuesta finalizada ya ha escrito su código de estado y encbezados.
- **void reset().** Limpipa cualquier dato que exista en el buffer, incluyendo código de estado y encabezados. Genrea la siguietne excepción:
  - java.lang.IllegalStateException – si la respuesta ya ha sido finalizada.
- void setLocale(java.util.Locale localización). Asigna la configuración de localización de la respuesta siempre y cuando no haya sido finalizada aún. También asigna la codificación de caracteres apropiada para la localización, siempre y cuando la codificación no se haya asignado explícitamente usando setContentType(String) o setCharacterEncoding(String), getWriter no he haya invocado aún y la respuesta no se haya finalilzado. Si el descriptor de despliegue contiene un elemento locale-encoding-mapping-list donde proporciona la configuración para la localización proporcionada, se utiliza esta; de otra forma, la selección de localización a partir de la codificación es dependiente del contenedor. Este método se puede invocar repetidamente para modificar la localización y la codificación de caracteres. Este método no tiene efecto si se invoca después de que la respuesta se ha finalizado. Los contenedores deben comunicar al cliente la localización y la codificación utilizada para la respuesta de servlets si el protocolo proporciona una forma para hacerlo. En el caso de HTTP, la localización se comunica como parte del encabezado Content-Language y la codificación como parte del encabezado Content-Type para textos. La codificación no se puede comunicar en encabezados HTTP si el servlet no especifica un tipo de contenido; aún así se usa para codificar el testo escrito por el servlet al writer de la respuesta.
- java.util.Locale getLocale(). Devuelve la localización especificada para este respuesta utilizando el método setLocale(java.util.Locale). Las llamadas realizadas a setLocale después de que la respuesta se haya finalizado no tienen efecto. Si no hay localización alguna especificada, se devuelve la localización por defecto del servidor.

2.12. La Interfaz HttpServletResponse.

public interface **javax.servlet.http.HttpServletResponse** extends **ServletResponse**

Extiende la interfaz ServletResponse para proporcionar funcionalidad específica de HTTPal enviar una respuesta. Por ejemplo, tiene métodos para acceder a encabezados de HTTP y cookies. El contenedor de servlets crea un objeto HttpServletResponse y lo pasa como argumento a los métodos de servicio del servlet (doGet, doPost, etc).

2.12.1. Métodos de la Interfaz HttpServletResponse.

- **void addCookie(Cookie cookie).** Agrega la cookie especificada a la respuesta. Este método se puede invocar varias veces para asignar más de una cookie.
- **boolean containsHeader(String nombre).** Devuelve true si el encabezado de la respuesta ya está asignado; en otro caso, devuelve false.
- **java.lang.String encodeURL(String url).** Codifica la URL especificada, incluyendo el identificador de sesión en ella. En caso de no ser necesario modificarla, devuelve la URL intacta. Si el navegador soporta cookies o el manejo de sesiones está deshabilitado, no se codifica la URL. Para un manejo de sesión robusto, todas las URL generadas por un servlet, deben generarse con este método para poder trabajar con navegadores que no soportan cookies.
- **String encodeRedirectURL(String url).** Codifica la URL especificada para usarse en el método sendRedirect o, si no se necesita codificación, devuelve la URL sin cambio. Las reglas de codificación para redirect son distintas a la codificación de ligas normales.
- **void sendError(int códigoDeEstado, String mensaje) throws java.io.IOException.** Envía una respuesta de error al cliente usando el código de estado especificado. El servidor crea una respuesta de error predeterminada que luce como una página de error del servidor que contiene el mensaje de error especificado, asigna el tipo de contenido "text/html", dejando sin cambio las cookies y otros encabezados. Si se hace una declaración de página de error para la aplicación web correspondiente al código de error proporcionado, esta se enviará de regreso. Después de usar este método, la respuesta debe considerarse finalizada y no se debe escribir. Genera las siguientes excepciones:
  - java.io.IOException – si ocurre un error de entrada o salida.
  - java.lang.IllegalStateException – si la respuesta ya ha sido finalizada antes de invocar este método.
- **void sendError(int sc) throws java.io.IOException.** Envía una respuesta de error al cliente usando el código de estado especificado y limpiando el buffer. Después de usar este método, la respuesta debe considerarse finalizada y no se debe escribir. Genera las siguientes excepciones:
  - java.io.IOException – si ocurre un error de entrada o salida.
  - java.lang.IllegalStateException – si la respuesta ya ha sido finalizada antes de invocar este método.
- **void sendRedirect(String url) throws java.io.IOException.** Envía una respuesta de redireccionamiento temporal al cliente, usando la URL indicada. El método puede aceptar URLs relativas, que el contenedor de servlets convierte en una URL absoluta antes de enviar la respuesta al cliente. Si la URL es relativa y no inicia con el caracter “/”, se interpreta de forma relativa al URI de la petición actual. Si la URI es relativa y empieza con “/”, se interpreta de forma relativa a la raíz del contexto. Después de usar este método, la respuesta debe considerarse finalizada y no se debe escribir. Genera las siguientes excepciones:

  - java.io.IOException – si ocurre un error de entrada o salida.
  - java.lang.IllegalStateException – si la respuesta ya ha sido finalizada antes de invocar este método o si la URL parcial proporcionada no se puede convertir en una URL válida.

- **void setDateHeader(String nombre, long fecha).** Asigna un encabezado de la respuesta con el nombre indicado y un valor tipo fecha. La fecha se especifica como el número de milisegundos a partir del primero de enero de 1970 GMT. Si el encabezado ya está asignado, el nuevo valor sobreescribe el anterior. Se puede usar el método **containsHeader** para verificar la presencia de un encabezado antes de asignarlo.
- **void addDateHeader(String nombre, long fecha).** Agrega un encabezado de la respuesta con el nombre indicado y un valor tipo fecha. La fecha se especifica como el número de milisegundos a partir del primero de enero de 1970 GMT. Este método permite que los encabezados de respuesta tengan múltiples valores.
- **void setHeader(String nombre, String valor).** Asigna un encabezado de la respuesta con el nombre y valor indicados. Si el encabezado ya está asignado, el nuevo valor sobreescribe el anterior. Se puede usar el método **containsHeader** para verificar la presencia de un encabezado antes de asignarlo. Si el valor contiene texto en octetos, debe ser codificado de acuerdo a RFC 2047 (http://www.ietf.org/rfc/rfc2047.txt).
- **void addHeader(String nombre, String value).** Agrega un encabezado de la respuesta con el nombre y valor indicados. Este método permite que los encabezados de respuesta tengan múltiples valores. Si el valor contiene texto en octetos, debe ser codificado de acuerdo a RFC 2047 (http://www.ietf.org/rfc/rfc2047.txt).
- **void setIntHeader(String nombre, int valor).** Asigna un encabezado de la respuesta con el nombre indicado y un valor entero. Si el encabezado ya está asignado, el nuevo valor sobreescribe el anterior. Se puede usar el método **containsHeader** para verificar la presencia de un encabezado antes de asignarlo.
- **void addIntHeader(String nombre, int valor).** Agrega un encabezado de la respuesta con el nombre indicado y un valor tipo entero. Este método permite que los encabezados de respuesta tengan múltiples valores.
- **void setStatus(int códigoDeEstado).** Asigna el código de estado de esta respuesta. Este método se utiliza para asignar código de estado de la respuesta cuando no se presenta un error (por ejemplo, para los códigos de estado SC_OK o SC_MOVED_TEMPORARILY). Si hay un error y el que invoca decide llamar una página de error definida en la aplicación web, es mejor usar el método **sendError**. El contenedor limpia el buffer y asigna el encabezado Location, manteniendo cookies y otros encabezados.

3. Formularios.

Un uso de los servlets es procesar la información enviada por formularios de HTML.

3.1. Archivo "CPForma.html".

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
    <HEAD>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <TITLE>Mi Primera Forma</TITLE>
    </HEAD>
    <BODY>
        <H1>Mi Primera Forma</H1>
        <FORM action="SPForma">
            <TABLE border="0" cellspacing="12">
                <TR>
                    <TD>
                        <LABEL for="nom" accesskey="N">Nombre(s):</LABEL>
                    </TD>
                    <TD><INPUT type="text" id="nom" name="nombre"></TD>
                </TR>
                <TR>
                    <TD>
                        <LABEL for="aps" accesskey="A">Apellidos:</LABEL>
                    </TD>
                    <TD><INPUT type="text" id="aps" name="apellidos"></TD>
                </TR>
                <TR>
                    <TD colspan="2" align="right">
                        <INPUT type="submit" name="guarda" value="Guardar"
                               accesskey="G">
                        <INPUT type="reset" value="Limpiar">
                    </TD>
                </TR>
            </TABLE>
        </FORM>
    </BODY>
</HTML>
```

3.2. SpForma.

```
package servlets;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
 8 import javax.servlet.http.HttpServletResponse;
 9
10 /** Servlet que muestra el valor de los parámetros <CODE>nombre</CODE>,
11  * <CODE>apellidos</CODE> y <CODE>guarda</CODE>recibidos por el método
12  * <CODE>GET</CODE>. */
13 public class SPForma extends HttpServlet {
14     /** Maneja el método <CODE>GET</CODE> de HTTP.
15      * @param solicitud la solicitud al servlet.
16      * @param respuesta la respuesta del servlet. */
17     @Override
18     protected void doGet(HttpServletRequest solicitud,
19             HttpServletResponse respuesta)
20             throws ServletException, IOException {
21         solicitud.setCharacterEncoding("UTF-8");
22         respuesta.setContentType("text/html;charset=UTF-8");
23         PrintWriter out = respuesta.getWriter();
24         try {
25             out.println("<!DOCTYPE HTML PUBLIC " +
26                     "\"-//W3C//DTD HTML 4.01//EN\" " +
27                     "\"http://www.w3.org/TR/html4/strict.dtd\">" +
28                     "<HTML>" +
29                     "<HEAD>" +
30                     "<META http-equiv='Content-Type' " +
31                     "content='text/html; charset=UTF-8'>" +
32                     "<TITLE>Parámetros Recibidos</TITLE>" +
33                     "</HEAD>" +
34                     "<BODY>" +
35                     "<H1>Parámetros Recibidos</H1>" +
36                     "<TABLE border='1'>" +
37                     "<THEAD>" +
38                     "<TR><TH>Parámetro</TH><TH>Valor</TH></TR>" +
39                     "<TBODY>" +
40                     "<TR><TD> nombre </TD><TD>" +
41                     solicitud.getParameter("nombre") +
42                     "</TD></TR>" +
43                     "<TR><TD> apellidos </TD><TD>" +
44                     solicitud.getParameter("apellidos") +
45                     "</TD></TR>" +
46                     "<TR><TD> guarda </TD><TD>" +
47                     solicitud.getParameter("guarda") +
48                     "</TD></TR>" +
49                     "</TABLE>" +
50                     "</BODY>" +
51                     "</HTML>");
52         } finally {
53             out.close();
54         }
55     }
56     /** Devuelve breve información del servlet.
57      * @return una breve información del servlet. */
58     @Override
59     public String getServletInfo() {
```

```
60          return "versión: 1.0. Copyright 2009 Gilberto Pacheco Gallegos.";
61      }
62  }
```

3.3. Archivo “CPControles.html”.

```
 1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 2  "http://www.w3.org/TR/html4/loose.dtd">
 3  <HTML>
 4      <HEAD>
 5          <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
 6          <TITLE>Formas en HTML</TITLE>
 7      </HEAD>
 8      <BODY>
 9          <H1>Formas en HTML</H1>
10          <IFRAME name="resultado"
11                  style="height: 600px; width: 30%; float:right"></IFRAME>
12          <FORM action="SPControles" target="resultado">
13              <H2>Con el Elemento <DFN>INPUT</DFN></H2>
14              <FIELDSET>
15                  <LEGEND>Text</LEGEND>
16                  Nombre(s): <INPUT type="text" name="nombre"><BR>
17                  <!-- Ejemplo de desplegar texto en el componente -->
18                  Apellidos: <INPUT type="text" name="apellidos"
19                                    value="P&eacute;rez Sosa" size="12"
20                                    maxlength="20">
21              </FIELDSET>
22              <FIELDSET>
23                  <LEGEND>Password</LEGEND>
24                  Contrase&ntilde;a: <INPUT type="password" name="contrasena">
25              </FIELDSET>
26              <FIELDSET>
27                  <LEGEND>CheckBox</LEGEND>
28                  Tiene Automovil:
29                  <INPUT type="checkbox" name="automovil" value="si">
30                  <BR>
31                  Sabe Leer:
32                  <INPUT type="checkbox" checked name="lee" value="si"><BR>
33                  D&iacute;as que Trabaja:
34                  <INPUT type="checkbox" name="trabaja" value="lunes"> Lunes
35                  <INPUT type="checkbox" name="trabaja" value="martes"> Martes
36                  <INPUT type="checkbox" name="trabaja" value="miercoles">
37                  Mi&eacute;rcoles
38                  <INPUT type="checkbox" name="trabaja" value="jueves">
39                  Jueves
40                  <INPUT type="checkbox" name="trabaja" value="viernes">
41                  Viernes
42              </FIELDSET>
43              <FIELDSET>
44                  <LEGEND>Radio</LEGEND>
45                  Sexo:
46                  <LABEL accesskey="M">
```

```
47                      <INPUT type="radio" name="sexo" value="masculino">
48                      Masculino
49                  </LABEL>
50                  <INPUT type="radio" id="fem" name="sexo" value="femenino">
51                  <LABEL for="fem" accesskey="F">Femenino</LABEL>
52              </FIELDSET>
53              <FIELDSET>
54                  <LEGEND>Submit</LEGEND>
55                  <INPUT type="submit" name="guarda" value="Guardar">
56                  <INPUT type="submit" name="reporta" value="Reportar">
57                  <INPUT type="submit" name="elimina" value="Eliminar">
58              </FIELDSET>
59              <FIELDSET>
60                  <LEGEND>Reset</LEGEND>
61                  <INPUT type="reset" value="Reestablecer">
62              </FIELDSET>
63              <FIELDSET>
64                  <LEGEND>Hidden</LEGEND>
65                  <INPUT type="hidden" name="oculto" value="Aqu&iacute; toy">
66              </FIELDSET>
67              <FIELDSET>
68                  <LEGEND>Image</LEGEND>
69                  <INPUT type="image" src="recolectores16.gif" name="imagen"
70                         alt="Recolectores">
71              </FIELDSET>
72              <H2>Con el Elemento <DFN>BUTTON</DFN></H2>
73              <P>
74                  <BUTTON type="submit" name="imprime" value="pp">
75                      Imprimir
76                  </BUTTON>
77                  <BUTTON type="submit" name="envia" value="envia">
78                      Enviar <IMG src="recolectores16.gif" alt="Recolectores">
79                  </BUTTON>
70                  <BUTTON type="reset">Reiniciar</BUTTON>
81              </P>
82              <H2>Con el Elemento <DFN>SELECT</DFN></H2>
83              <P>
84                  Asignatura:
85                  <SELECT name="asignatura">
86                      <OPTION value="TICLOGP">
87                          L&oacute;gica de Programaci&oacute;n
88                      </OPTION>
89                      <OPTION value="TICESDA">Estructuras de Datos</OPTION>
90                      <OPTION value="TICPRVI" selected>
91                          Programaci&oacute;n Visual
92                      </OPTION>
93                      <OPTION value="TICPRAV">
94                          Programaci&oacute;n Avanzada
95                      </OPTION>
96                  </SELECT>
97              </P>
98              <P> Aficiones:
```

```
 99                 <SELECT multiple size="4" name="aficiones1">
100                     <OPTION selected value="tejer">Tejer</OPTION>
101                     <OPTION selected value="futbol">Jugar Futbol</OPTION>
102                     <OPTION>Leer Revistas</OPTION>
103 
104                     <OPTION value="chatear">Chatear con mis cuates</OPTION>
105                     <OPTION>Ver Futbol</OPTION>
106                     <OPTION>Bailar</OPTION>
107                 </SELECT>
108                 <SELECT multiple size="4" name="aficiones2">
109                     <OPTGROUP label="Tranquilas">
100                         <OPTION selected value="tejer">Tejer</OPTION>
111                         <OPTION>Leer Revistas</OPTION>
112                         <OPTION value="chatear">
113                             Chatear con mis cuates
114                         </OPTION>
115                         <OPTION>Ver Futbol</OPTION>
116                     </OPTGROUP>
117                     <OPTGROUP label="Movidas">
118                         <OPTION selected value="futbol">Jugar Futbol</OPTION>
119                         <OPTION>Bailar</OPTION>
120                     </OPTGROUP>
121                 </SELECT>
122             </P>
123             <H2>Con el Elemento <DFN>TEXTAREA</DFN></H2>
124             <P>
125                 Direcci&oacute;n:
126                 <TEXTAREA rows="3" cols="15">Calle del Sapo #15</TEXTAREA>
127             </P>
128             <H2>Controles Deshabilitados</H2>
129             <P>Solo aplica para:</P>
130             <UL>
131                 <LI>BUTTON</LI>
132                 <LI>INPUT</LI>
133                 <LI>OPTGROUP</LI>
134                 <LI>OPTION</LI>
135                 <LI>SELECT</LI>
136                 <LI>TEXTAREA</LI>
137             </UL>
138             <P>Los navegadores m&aacute;s viejos no lo soportan.</P>
139             <P> <INPUT type="submit" disabled name="destruye"
140                     value="Destrucci&oacute;n Total"></P>
141             <H2>Controles de Solo Lectura</H2>
142             <P>
143                 Solo para INPUT y TEXTAREA. Los navegadores m&aacute;s viejos
144                 no lo soportan.
145             </P>
146             <P>
147                 <INPUT type="text" readonly name="cambio"
148                     value="A que no me cambias">
149             </P>
150         </FORM>
```

```
151     </BODY>
152 </HTML>
```

3.4. SpControles.

```
  1 package servlets;
  2
  3 import java.io.IOException;
  4 import java.io.PrintWriter;
  5 import java.util.Enumeration;
  6 import javax.servlet.ServletException;
  7 import javax.servlet.http.HttpServlet;
  8 import javax.servlet.http.HttpServletRequest;
  9 import javax.servlet.http.HttpServletResponse;
 10
 11 /** Servlet que muestra todos los parámetros recibidos por el método
 12  * <CODE>GET</CODE>. */
 13 public class SPControles extends HttpServlet {
 14     /** Maneja el método <CODE>GET</CODE> de HTTP.
 15      * @param solicitud la solicitud al servlet.
 16      * @param respuesta la respuesta del servlet. */
 17     @Override
 18     protected void doGet(HttpServletRequest solicitud,
 19             HttpServletResponse respuesta)
 20             throws ServletException, IOException {
 21         solicitud.setCharacterEncoding("UTF-8");
 22         respuesta.setContentType("text/html;charset=UTF-8");
 23         PrintWriter out = respuesta.getWriter();
 24         try {
 25             out.println("<!DOCTYPE HTML PUBLIC " +
 26                     "\"-//W3C//DTD HTML 4.01//EN\" " +
 27                     "\"http://www.w3.org/TR/html4/strict.dtd\">" +
 28                     "<HTML>" +
 29                     "<HEAD>" +
 30                     "<META http-equiv='Content-Type' " +
 31                     "content='text/html; charset=UTF-8'>" +
 32                     "<TITLE>Par&aacute;metros Recibidos</TITLE>" +
 33                     "</HEAD>" +
 34                     "<BODY>" +
 35                     "<H2>Parámetros Recibidos</H2>" +
 36                     "<DL>");
 37             Enumeration<?> nombres = solicitud.getParameterNames();
 38             while (nombres.hasMoreElements()) {
 39                 String nombre = nombres.nextElement().toString();
 40                 out.println("<DT>" + nombre + "</DT>" +
 41                         "<DD>" +
 42                         "<TABLE border='1'>");
 43                 String[] valores = solicitud.getParameterValues(nombre);
 44                 for (String valor : valores) {
 45                     out.println("<TR><TD>" + valor + "</TD></TR>");
 46                 }
 47                 out.println("</TABLE>" +
```

```
48                          "</DD>");
49              }
50              out.println("</DL>" +
51                      "</BODY>" +
52                      "</HTML>");
53          } finally {
54              out.close();
55          }
56      }
57      /** Devuelve breve información del servlet.
58       * @return una breve información del servlet. */
59      @Override
60      public String getServletInfo() {
61          return "versión: 1.0. Copyright 2009 Gilberto Pacheco Gallegos.";
62      }
63  }
```

3.5. Archivo “CPSumas.html”.

```
 1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 2  "http://www.w3.org/TR/html4/loose.dtd">
 3  <HTML>
 4      <HEAD>
 5          <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
 6          <TITLE>Sumas</TITLE>
 7      </HEAD>
 8      <BODY>
 9          <H2>Sumas</H2>
10          <IFRAME name="resultado" width="100%" height="100"></IFRAME>
11          <FORM action="SPSumas" method="post" target="resultado">
12              <TABLE>
13                  <TR>
14                      <TD><LABEL for="a" accesskey="A">A:</LABEL></TD>
15                      <TD><INPUT type="text" id="a" name="a"></TD>
16                  </TR>
17                  <TR>
18                      <TD><LABEL for="b" id="b" accesskey="B">B:</LABEL></TD>
19                      <TD><INPUT type="text" name="b"></TD>
20                  </TR>
21                  <TR>
22                      <TD colspan="2" align="right">
23                          <INPUT type="submit" value="Sumar">
24                      </TD>
25                  </TR>
26              </TABLE>
27          </FORM>
28      </BODY>
29  </HTML>
```

3.6. SpSumas.

```
 1 package servlets;
 2
 3 import java.io.IOException;
 4 import java.io.PrintWriter;
 5 import javax.servlet.ServletException;
 6 import javax.servlet.http.HttpServlet;
 7 import javax.servlet.http.HttpServletRequest;
 8 import javax.servlet.http.HttpServletResponse;
 9
10 /** Servlet que realiza la suma de los parámetros enteros <CODE>a</CODE> y
11  * <CODE>b</CODE>, recibidos por el método <CODE>POST</CODE>. */
12 public class SPSumas extends HttpServlet {
13     /** Maneja el método <CODE>POST</CODE> de HTTP.
14      * @param solicitud la solicitud al servlet.
15      * @param respuesta la respuesta del servlet. */
16     @Override
17     protected void doPost(HttpServletRequest solicitud,
18             HttpServletResponse respuesta)
19             throws ServletException, IOException {
20         solicitud.setCharacterEncoding("UTF-8");
21         int a = Integer.parseInt(solicitud.getParameter("a"));
22         int b = Integer.parseInt(solicitud.getParameter("b"));
23         int c = a + b;
24         /* Asigna la codificación de la respuesta que se envía al
25          * navegador. */
26         respuesta.setContentType("text/html;charset=UTF-8");
27         PrintWriter out = respuesta.getWriter();
28         /* Construye una página que contiene la respuesta del servidor
29          * web. */
30         try {
31             out.println("<!DOCTYPE HTML PUBLIC " +
32                     "\"-//W3C//DTD HTML 4.01//EN\" " +
33                     "\"http://www.w3.org/TR/html4/strict.dtd\">" +
34                     "<HTML>" +
35                     "<META http-equiv='Content-Type' " +
36                     "content='text/html; charset=UTF-8'>" +
37                     "<HEAD>" +
38                     "<TITLE>Resultado</TITLE>" +
39                     "</HEAD>" +
40                     "<BODY>" +
41                     "<H3>Resultado</H3>" +
42                     "<P><STRONG>a + b = </STRONG> " + c + "</P>" +
43                     "</BODY>" +
44                     "</HTML>");
45         } finally {
46             out.close();
47         }
48     }
49     /** Devuelve breve información del servlet.
50      * @return una breve información del servlet. */
```

```
51      @Override
52      public String getServletInfo() {
53          return "versión: 1.0. Copyright 2009 Gilberto Pacheco Gallegos.";
54      }
55  }
```

3.7. web.xml.

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
 3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 5           http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 6      <servlet>
 7          <servlet-name>SPControles</servlet-name>
 8          <servlet-class>servlets.SPControles</servlet-class>
 9      </servlet>
10      <servlet>
11          <servlet-name>SPForma</servlet-name>
12          <servlet-class>servlets.SPForma</servlet-class>
13      </servlet>
14      <servlet>
15          <servlet-name>SPSumas</servlet-name>
16          <servlet-class>servlets.SPSumas</servlet-class>
17      </servlet>
18      <servlet-mapping>
19          <servlet-name>SPControles</servlet-name>
20          <url-pattern>/SPControles</url-pattern>
21      </servlet-mapping>
22      <servlet-mapping>
23          <servlet-name>SPForma</servlet-name>
24          <url-pattern>/SPForma</url-pattern>
25      </servlet-mapping>
26      <servlet-mapping>
27          <servlet-name>SPSumas</servlet-name>
28          <url-pattern>/SPSumas</url-pattern>
29      </servlet-mapping>
30      <welcome-file-list>
31          <welcome-file>index.html</welcome-file>
32      </welcome-file-list>
33  </web-app>
```

3.8. index.html.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
2  "http://www.w3.org/TR/html4/frameset.dtd">
3  <HTML>
4      <HEAD>
5          <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
6          <TITLE>Servlets con Par&aacute;metros</TITLE>
7      </HEAD>
```

```
 8     <FRAMESET rows="15%, 75%">
 9         <FRAME src="CPEncabezado.html">
10         <FRAMESET cols="20%, 80%">
11             <FRAME src="CPToc.html">
12             <FRAME src="CPForma.html" name="dinamico">
13         </FRAMESET>
14         <NOFRAMES>
15             <H1>Contenido</H1>
16             <UL>
17                 <LI><A href="CPForma.html">Forma Simple</A></LI>
18                 <LI><A href="CPControles.html">Controles Disponibles</A></LI>
19                 <LI><A href="CPSumas.html">Sumas</A></LI>
20             </UL>
21         </NOFRAMES>
22     </FRAMESET>
23 </HTML>
```

3.9. CPEncabezado.html.

```
 1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 2 "http://www.w3.org/TR/html4/strict.dtd">
 3 <HTML>
 4     <HEAD>
 5         <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
 6         <TITLE>Formas</TITLE>
 7     </HEAD>
 8     <BODY>
 9         <H1>Formas</H1>
10     </BODY>
11 </HTML>
```

3.10. CPToc.html.

```
 1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 2 "http://www.w3.org/TR/html4/loose.dtd">
 3 <HTML>
 4     <HEAD>
 5         <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
 6         <TITLE>Contenido</TITLE>
 7     </HEAD>
 8     <BODY>
 9         <H2>Contenido</H2>
10         <UL>
11             <LI>
12                 <A href="CPForma.html" target="dinamico">Forma Simple</A>
13             </LI>
14             <LI>
15                 <A href="CPControles.html" target="dinamico">
16                     Controles Disponibles
17                 </A>
18             </LI>
```

```
19              <LI><A href="CPSumas.html" target="dinamico">Sumas</A></LI>
20          </UL>
21      </BODY>
22  </HTML>
```

4. Introducción a JSP.

Normalmente una aplicación web consta de:

- Un ambiente de ejecución Java, donde se ejecuta la aplicación.
- Páginas JSP que manejan solicitudes y generan contenido dinámico.
- Servlets que manejan solicitudes y generan contenido dinámico.
- Páginas estáticas en HTML, DHTML, XHTML, XML, y páginas similares.
- En el lado cliente, applets de Java, componentes JavaBeans y clases arbitrarias.
- Clientes robustos que usar al ambiente de ejecución Java, que se pueden descargar con el plugin de Java y la tecnología Java™ Web Start technology.

Una página de JSP es un documento que contiene dos tipos de texto: datos estáticos, que pueden ser expresados en algún formato basado en texto (como HTML, SVG, WML, y XML), y los elementos de JSP, que construyen el contenido dinámico.

La extensión recomendada para archivos fuente de una página de JSP es ".jsp". La página puede estar compuesta por archivos de alto nivel que incluyen otros archivos que contiene o una página completa de JSP o un fragmento de una página de JSP. Estos últimos archivos se conocen como **segmentos de JSP** y su extensión recomendada es ".jspf". El elemento jsp-property-group de web.xml puede ser utilizado para indicar que algún grupo de archivos, que quizás no utiliza ninguna de las extensiones arriba, son páginas de JSP.

4.1. Ejemplo de un JSP.

Un archivo de JSP puede contener las siguientes construcciones, entre otras:

- **Directivas de página (<%@page ... %>).** Asignan el tipo de contenido devuelto por la página.
- **Directivas de biblioteca de etiquetas (<%@taglib ... %>).** Importan bibliotecas de etiquetas personalizadas.
- **jsp: useBean**. Elemento estándar que crea un objeto e inicializa un identificador que señala a ese objeto.
- **jsp: setProperty.** Elemento estándar que asigna el valor de una propiedad de objeto.
- **Expresiones del idioma de la expresiones de JSP ($ {}).** Recuperan el valor de propiedades de objeto. Los valores son utilizados para asignar valores para atributo de etiquetas de usuario y crear el contenido dinámico.

- **Etiquetas personalizadas.** Asignan una variable (c:set), iteran sobre una colección de nombres de lugar (c:forEach), condicionalmente insertan texto HTML en la respuesta (f:if, c:choose, c:when, c:otherwise) y muchas funciones más.
- **Scripts con código en Java (<%! %>, <% %> y <%= %>).** Permiten insertar código en Java. Debido a que uno de los principales objetivos de la tecnología JSP es separar los datos mostrados del proceso necesario para generarlos, su uso es poco recomendado.

4.1.1. JSPSencillo.jsp.

```
<%-- Este es un comentario de JSP. --%>
<%-- Al inicio de la página se colocan directivas, las cuales controlan la
     ejecución de una página. --%>
<%-- Las siguientes directivas controlan el tipo de respuesta a generar.
     contentType  - Tipo MIME. El valor "text/html; charset=UTF-8"
                    para páginas en html con codificación UTF-8. Debe
                    coincidir con el META http-equiv="Content-Type" indicado
                    en la página.
     pageEncodig  - El tipo de codificación también puede ponerse fuera del
                    content type, utilizando este atributo. --%>
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%-- La directiva "errorPage" asigna la página que procesa los errores
     generados durante la ejecución de esta página. --%>
<%@ page errorPage="error.jsp" %>
<%-- La directiva info se utiliza para generar el valor devuelto por el
     método getServletInfo del servlet generado al traducir este JSP. --%>
<%@ page info="versión 1.0 Copyright 2009 Gilberto Pacheco Gallegos." %>
<%-- La directiva import permite incluir recursos utilizados por scripts de
     código Java. --%>
<%@ page import="java.util.ArrayList,
         java.util.Date,
         java.util.HashMap"%>
<%-- Carga la librería de etiquetas básicas. Se usa con el prefijo c. --%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%-- La siguiente es una acción (porque empieza con "jsp:") y permite
     utilizar Java Beans (son clases de java que tienen un constructor en
     blanco, así como métodos get y set). Usa los siguientes atributos:
     id    - Nombre de la referencia al bean.
     scope - nivel de visibilidad de la variable. Los posibles valores son:
             page, request, session y application. --%>
<%! /* Esta sección define métodos y variables de instancia del JSP
     * compilado. Es un script de java. */
    /** Método de inicialización del jsp. */
    public void jspInit() {
        log("JSPSencillo iniciado");
    }
    /** Método de terminación del jsp. */
    public void jspDestroy() {
        log("JSPSencillo eliminado");
    }
%>
```

```
<jsp:useBean id="bean1" scope="request" class="beans.BeanSencillo"/>
<jsp:useBean id="texto1" scope="page" class="java.lang.String"/>
<%-- Esta acción asigna las propiedades de bean1 a partir de los parámetros
     recibidos en la solicitud. --%>
<jsp:setProperty name="bean1" property="*"/>
<%--La siguiente etiqueta personalizada define un url a esta página. La
    expresión que va entre ${} se utiliza para realizar evaluaciones y está
    escrita en un lenguaje conocido como lenguaje de expresiones. --%>
<c:url var="accion" value="${pageContext.request.servletPath}" />
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
  <HEAD>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <TITLE>Ejemplo de página JSP</TITLE>
  </HEAD>
  <BODY>
    <%@ include file="/WEB-INF/jspf/SegmentoSencillo.jspf" %>
    <H2>Tu eres: <jsp:getProperty name="bean1" property="nombre" />.</H2>
    <%-- Ejemplo de etiquetas personalizadas. Despliega una tabla de
         multiplicar. --%>
    <c:forEach var="i" begin="1" end="10">
      <P> ${i} * ${bean1.base} = ${i * bean1.base}
    </c:forEach>
    <% // Este script de java se incluye en la página.
        int a = 1;
        a = a + 1;
    %>
    <%-- De esta forma se puede mostrar el resultado de expresiones java.--%>
    <P> Resultado del script más 2: <%=a + 2%>
    <FORM action="${accion}" method="post">
      <TABLE>
        <CAPTION>Introduzca los siguientes valores:</CAPTION>
        <TR>
          <TD>Nombre:</TD>
          <TD><INPUT type="text" name="nombre"></TD>
        </TR>
        <TR>
          <TD>Base:</TD>
          <TD><INPUT type="text" name="base"></TD>
        </TR>
        <TR>
          <TD colspan="2" align="right">
            <INPUT type="submit" value="Enviar">
          </TD>
        </TR>
      </TABLE>
    </FORM>
  </BODY>
</HTML>
```

3.1.2. SegmentoSencillo.jspf

```
<%-- Ejemplo de un segmento de jsp. Está pensado para ser parte de una
     página. Este tipo de archivos normalmente se colocan en la carpeta
     WEB-INF/jspf para que no se puedan acceder desde fuera del servidor.
     Obsérvese que no se indica el contentType.
--%>
<%@ page pageEncoding="UTF-8" %>
<H1>¡Hola Lola!</H1>
```

3.1.3. error.jsp.

```
<%-- Ejemplo de una página de error. --%>
<%-- Indica el tipo de respuesta a generar.
     contentType  - Tipo MIME. El valor "text/html; charset=UTF-8"
                    para páginas en html con codificación UTF-8. Debe
                    coincidir con el META http-equiv="Content-Type" indicado
                    en la página.
     pageEncodig  - El tipo de codificación también puede ponerse fuera del
                    content type, utilizando este atributo.
--%>
<%@ page contentType="text/html; charset=UTF-8"%>
<%-- Indica que es una página de error --%>
<%@ page isErrorPage="true" %>
<%-- Carga la librería de etiquetas básicas. Se usa con el prefijo c. --%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
  <HEAD>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <TITLE>Error</TITLE>
  </HEAD>
  <BODY>
    <H1>Error</H1>
    <%-- Indica la causa del error --%>
    <P>Causa: ${pageContext.errorData.throwable.cause}</P>
    <P>Status Code:${pageContext.errorData.statusCode}</P>
  </BODY>
</HTML>
```

3.1.4. BeanSencillo.

```
package net.ramptors.beans;

/** Bean diseñado para apoyar la página JSPSencilla. */
public class BeanSencillo {
    private String nombre = "";
    private int base = -1;
    public String getNombre() {
        return nombre;
```

```
 9      }
10      public void setNombre(String nombre) {
11          this.nombre = nombre.trim();
12      }
13      public int getBase() {
14          return base;
15      }
16      public void setBase(int base) {
17          this.base = base;
18      }
19  }
```

3.2. El Ciclo de Vida de una Página de JSP.

Una página de JSP atiende a peticiones como un servlet. Así, el ciclo vital y muchas de las capacidades de páginas de JSP (en particular los aspectos dinámicos) son determinados por la tecnología de Servlets de Java.

Cuándo una petición es trazada a una página de JSP, el contenedor web verifica primero si el servlet de la página de JSP es más viejo que la página de JSP. Si el servlet es más viejo, el contenedor web traduce la página de JSP en una clase de servlet y compila la clase, que se conoce como **clase de implementación**. Durante el desarrollo, uno de las ventajas de páginas de JSP sobre servlets es que el proceso de construcción es realizado automáticamente. Las páginas JSP se pueden traducir antes de su uso, proporcionando a la aplicación web una clase servlet que sirve como la representación de la página JSP. La traducción también se puede realizar al momento de desplegar la aplicación o sobre demanda, cuando una solicitud llega a una página JSP no traducida.

3.3. Traducción y Compilación.

Durante la fase de la traducción cada tipo de datos en una página de JSP es tratado de forma distinta. Los datos estáticos son transformados en código que emitirá los datos en la respuesta. Los elementos de JSP son tratados de la siguiente manera:

- Las directivas son utilizadas para controlar cómo el contenedor web traduce y ejecuta la página de JSP.
- Los scripts de código son insertados en la clase del servlet de la página de JSP.
- Las expresiones del idioma de expresiones son pasadas como parámetros a llamadas al evaluador de la expresión de JSP.
- Los elementos jsp: [set|get]Property] son convertidos en llamadas de método a componentes JavaBeans.
- Los elementos jsp: [include|forward] son convertidos en invocaciones de la Java Servlet API.

- El el elemento jsp:plugin es convertido en etiquetas específicas del navegador para activar un applet.
- Las etiquetas personalizadas (custom tags) son convertidas en llamadas al manejador de etiquetas que correponde a la etiqueta personalizada.

Las fases de traducción y las de compilación pueden generar errores que son observados sólo cuando la página es solicitada por primera vez. Si un error es encontrado con durante cualquier fase, el servidor devolverá una excepción JasperException y un mensaje que incluye el nombre de la página de JSP y la línea donde el error ocurrió.

Después de que la página haya sido traducida y compilada, el servlet de la página de JSP (la mayor parte del tiempo) sigue el ciclo vital de un servlet :

1. Si una instancia del servlet de la página de JSP no existe, el contenedor:
   a) Carga la clase del servlet de la página de JSP.
   b) Crea una instancia de la clase de servlet.
   c) Inicializa la instancia del servlet llamando el método de jspInit.
2. El contenedor invoca el el método _jspService, pasando los objetos de petición y respuesta.

Si el contenedor necesita para quitar el servlet de la página de JSP, llama el método de jspDestroy.

Para los servlets se aplican los mismos conceptos de aplicación web, ServletContext, solicitudes y respuestas discutidos en el capítulo anterior.

Las páginas JSP y las clases servlets son referidos en conjunto como componentes web. Las páginas JSP normalmente son colocadas en un contenedor que le provee ciertos servicios específicos. La separación entre componentes y contenedores permite reutilizar los primeros.

Un contenedor JSP es una entidad a nivel de sistema que permite administrar el ciclo de vida y soporta la ejecución de componentes JSP y servlets. La solicitudes enviadas a una página JSP son transmitidas al correspondiente objeto de implementación de la página. El término contenedor web es sinónimo de contenedor JSP. El elemento servlet del archivo de despliegue “web.xml” se usa para describir los dos tipos de componentes web.

3.4. El Lenguaje de Expresiones.

El lenguaje de expresión se utiliza básicamente para calcular valores. Consta de expresiones escritas dentro de “${“ y “}”. Estas expresiones se pueden utilizar en el texto estático o en los atributos de las etiquetas.

Las expresiones más básicas son de los siguientes tipos:

- ${variable}. Devuelve el valor de una variable.
- ${'texto'}. Devuelve la cadena “texto” o bien un elemento de enum definido como enum X {texto, ...}.
- ${“texto”}. Devuelve la cadena “texto” o bien un elemento de enum definido como enum X {texto, ...}..
- ${variable.campo} o ${variable[“campo”]}. Puede devolver distintos tipos de valores.
  - Si “variable" es un java bean, devuelve variable.getCampo();
  - Si “variable” es un Map, devuelve variable.get(campo).
- $variable[3]. Devuelve el elemento en el índice 3 de un arreglo.
- ${f:función(p1, p2)}. Invoca una función y le pasa parámetros. Solo se vale para funciones definidas en librerías de etiquetas personalizadas.

Es posible usar los siguientes tipos de literales:

- **Booleanas: true** y **false**.
- **Enteras:** como en Java.
- **De Punto Flotante:** como en Java.
- **Cadenas de Texto:** delimitadas con apóstrofes o comillas. En el interior de ellas, “ se representa \”, ' se representa \' y \ se representa \\.

Se usan los siguientes operadores:

- **Arithméticos:** +, - (binario), *, /, div, %, mod y - (unario)
- **Lógicos:** and, &&, or, ||, not, !
- **Relacionales:** ==, eq, !=, ne, <, lt, >, gt, <=, ge, >=, le. Se pueden comparar contra otros valores, o contra literales.
- **Empty:** El operador empty es una operación prefija que puede usarse para determinar si un valor es nulo o una cadena vacía. Por ejemplo: ${empty texto}
- Condicional: A ? B : C. Si A es verdadero, devuelve B. Si A es falso, devuelve C.

La precedencia de operadores es la siguiente:

- # [] .
- # ()
- # - (unario) not ! empty
- # * / div % mod
- # + - (binario)
- # < > <= >= lt gt le ge
- # == != eq ne
- # && and
- # || or
- # ? :

Las siguientes son palabras reservadas y no se pueden utilizar como nombres de variables.

and eq gt true instanceof

or ne le false empty

not lt ge null div mod

3.5. Niveles de visibilidad de las variables JSP.

- **Alcance de página (page).** Los objetos con el alcance de página son accesibles sólo desde de la página donde son creados. Todas referencias a tal objeto serán eliminadas después de que la respuesta sea devuelta al cliente de la página de JSP o la petición es reenviada a otro componente web. Las referencias a objetos con el alcance de página son almacenadas en el objeto de **pageContext**.
- **Alcance de petición (request).** Los objetos con el alcance de petición son accesibles desde de páginas que procesan la petición donde fueron creados. Las referencias al objeto serán eliminadas después de que la petición sea procesada. En particular, si la petición es reenviada a un recurso en el mismo tiempo de ejecución, el objeto es todavía accesible. Las referencias a objetos con el alcance de petición son almacenadas en el objeto **request**.
- **Alcance de sesión (session).** Los objetos con el alcance de sesión son accesibles desde páginas que procesan peticiones que están en la misma sesión en que fueron creados. No es legal definir un objeto con el alcance de sesión de dentro de una página que no se coordina con sesiones. Todas referencias al objeto serán eliminadas después del fin de la sesión. Las referencias a objetos con el alcance de sesión son almacenadas en el objeto **session** asociado con la activación de la página.
- **Alcance de aplicación (application).** Los objetos con el alcance de aplicación son accesibles desde páginas que procesan peticiones que está en la misma aplicación como en que fueron

creados. Los objetos con el alcance de aplicación pueden ser definidos (y alcanzados) desde páginas que no se coordinan con sesiones. Las referencias a objetos con el alcance de aplicación son almacenadas en el objeto **application** asociado con la activación de la página. El objeto de aplicación es el contexto de servlet obtenido del objeto de configuración de servlet. Todas referencias al objeto serán elimnadas cuando el ambiente del tiempo de ejecución elimina el ServletContext.

Un nombre debe referirse a un objeto único en todas partes de la ejecución; eso es, todos los alcances diferentes deben comportarse realmente como un solo espacio de nombres. Una implementación del contenedor de JSP puede o no puede imponer esta regla explícitamente para razones de desempeño.

3.6. Objetos Predefinidos en páginas JSP.

- pageContext. El contexto para la página JSP. Proporciona a varios objetos, incluyendo:
  - servletContext. El contexto para el servlet de esta página.
  - session. El objeto de sesión para el cliente.
  - request. La petición que dispara la ejecución de la página JSP.
  - response. La respuesta devuelta por la página JSP.
- param. Permite obtener el valor asociado con un parámetro.
- paramValues. Permite obtener un arreglo con los valores asociados a un parámetro.
- header. Permite obtener el valor asociado con un encabezado de la petición.
- headerValues. Permite obtener un arreglo con los valores asociados a un encabezado de la petición.
- cookie: Permite obtener el valor asociado con un cookie.
- initParam: Permite obtener el valor de un parámetro de inicialización del contexto.
- pageScope: Permite obtener las variables con alcance de página.
- requestScope: Permite obtener las variables con alcance de petición.
- sessionScope: Permite obtener las variables con alcance de sesión.
- applicationScope: Permite obtener las variables con alcance de aplicación.

3.7. Objetos Predefinidos en Scripts de Java Dentro de Páginas JSP.

Nombre	Tipo	Semántica	Alcance
request	Depende del protocolo y es subtipo de javax.servlet.ServletRequest, por ejemplo javax.servlet.http.HttpServletRequest	Solicitud que dispara la invocación del servicio.	Petición (request).

Nombre	Tipo	Semántica	Alcance
response	Depende del protocolo y es subtipo de javax.servlet.ServletResponse , por ejemplo javax.servlet.http.HttpServletResponse	Respuesta a la solicitud.	Página (page).
pageContext	javax.servlet.jsp.PageContext	El contexto de página para esta página JSP.	Página (page).
session	javax.servlet.http.HttpSession	Objeto de sesión creado para el cliente (si es que existe). Solo es válido para protocolos HTTP.	Sesión (session).
application	javax.servlet.ServletContext	Contexto de servlet obtenido del servlet (como en la invocación getServletConfig(). getContext()).	Aplicación (application).
out	javax.servlet.jsp.JspWriter	Escribe la salida	Página (page).
config	javax.servlet.ServletConfig	ServletConfig para esta página JSP.	Página (page).
page	java.lang.Object	Instancia de la clase de implementación para esta página.	Página (page).

3.8. Etiquetas Personalizadas y JSTL.

Para simplificar el desarrollo de aplicaciones, la tecnología JSP permite desarrollar etiquetas personalizadas, con una significado bien establecido, que puede ser validado por las herramientas de desarrollo. Para usarlas, se necesita cargar una librería de etiquetas con la directiva <%@ taglib %>, con la cual se indica que librería se usa y se le asigna un prefijo para su uso y que sus etiquetas no se confundan con otras que se llamen igual, pero correspondan a otra librería.

Hay una librería básica, la JavaServer Pages Standard Tag Library (JSTL), que contiene algunas etiquetas y funciones de uso común. Está definida en partes.

3.8.1. La librería Core de JSTL.

- **URL:** http://java.sun.com/jsp/jstl/core

- **Prefijo utilizado comunmente:** c:.
- **Aspectos:**

Función	Etiquetas
Manejo de variables	remove set
Control de Flujo	choose when otherwise forEach forTokens if
Manejo de URLs	import param redirect param url param
Varios	catch out

3.8.2. La librería XML de JSTL.

- **URL:** http://java.sun.com/jsp/jstl/xml
- **Prefijo utilizado comunmente:** x:.
- **Aspectos:** Generación y Análisis de XML.

Función	Etiquetas
Básico	out parse set
Control de Flujo	choose when otherwise forEach

	if
Transformación	transform param

3.8.3. La librería de Internacionalización de JSTL.

- **URL:** http://java.sun.com/jsp/jstl/fmt
- **Prefijo utilizado comunmente:** fmt:.
- **Aspectos:** Generación y Análisis de XML.

Función	Etiquetas
Configuración de localización	setLocale requestEncoding
Manejo de mensajes.	bundle message param setBundle
Formato de números y fechas	formatNumber formatDate parseDate parseNumber setTimeZone timeZone

3.8.4. La librería de SQL de JSTL.

Esta librería se ofrece para desarrollo de prototipos. Para obtener la mayor eficiencia hay que usar Eterprise Java Beans (EJB).

- **URL:** http://java.sun.com/jsp/jstl/sql
- **Prefijo utilizado comunmente:** sql:.
- **Aspectos:** Generación y Análisis de XML.

Función	Etiquetas
Asignación de DataSource	setDataSource

Función	Etiquetas
SQL.	query dateParam param transaction update dateParam param

3.8.5. La librería de Funciones de JSTL.

- **URL:** http://java.sun.com/jsp/jstl/functions
- **Prefijo utilizado comunmente:** sql:.
- **Aspectos:** Generación y Análisis de XML.

Función	Etiquetas
Longitud de Colecciones	length
Manipulación de cadenas de texto.	toUpperCase, toLowerCase substring, substringAfter, substringBefore trim replace indexOf, startsWith, endsWith, contains, containsIgnoreCase split, join escapeXml

3.9. Especificación de URLs Relativas.

Los elementos pueden utilizar especificaciones de URLs relativas, llamadas rutas URI. Estas rutas son descritas en RFC 2396. Nos referimos a la parte de la ruta de esa especificación, no al esquema, ni las partes de autoridad.

• Una **ruta relativa al contexto** es una ruta que comienza con una diagonal (/). Se interpreta como relativa a la aplicación a la cual pertenece el archivo donde aparece. Esto es, su objeto ServletContext proporciona el contexto base de la URL.

• Una **ruta relativa a la página** es una ruta que no comienza con una diagonal (/). Se interpreta como relativa al archivo actual, dependiendo de donde se utilice la ruta. La página o el archivo actuales son denotados por alguna ruta que comienza con / que es entonces modificada por la nueva especificación para producir una ruta que comienza con /. La nueva ruta es interpretada por el objeto de ServletContext.

4. Cookies y Sesiones.

Los cookies son archivos que guarda un navegador de Internet y permanecen aún después de terminar su ejecución. Nos sirven para guardar información como el nombre de usuario o preferencias de uso de un sitio, por ejemplo, colores o colocación de páneles. Se guardan unas cuantas cookies por sitio (4 normalmente) y el total de estas generalmente es menor a 20.

Las sesiones almacenan información en uso por el usuario y se terminan cuando el usuario lo solicita o al cierra el navegador de Internet. Cuando se cierra la sesión, los datos asociados con esta se destruyen.

Los datos de sesión normalmente se guardan en una cookie o en un archivo temporal del servidor.

4.1. JSPCookies.jsp.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<c:url var="acción" value="/SPCookies"/>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
  <HEAD>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <TITLE>Cookies</TITLE>
  </HEAD>
  <BODY>
    <H2>Cookies</H2>
    <H3>Valor del cookie: <c:out value="${cookie['saludo'].value}"/></H3>
    <FORM action="${acción}" method="post">
      <TABLE>
        <TR>
          <TD><LABEL>Valor:</LABEL></TD>
          <TD>
            <INPUT type="text" name="valor" value="${cookie["saludo"].value}">
          </TD>
        </TR>
        <TR>
          <TD colspan="2" align="right">
            <INPUT type="submit" value="Guardar">
          </TD>
        </TR>
      </TABLE>
    </FORM>
  </BODY>
</HTML>
```

4.2. SPCookies.

```java
package servlets;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SPCookies extends HttpServlet {
    /** Maneja el método HTTP <code>POST</code>.
     * @param solicitud la solicitud al servlet.
     * @param respuesta la respuesta del servlet. */
    @Override
    protected void doPost(HttpServletRequest solicitud,
            HttpServletResponse respuesta)
            throws ServletException, IOException {
        solicitud.setCharacterEncoding("UTF-8");
        String valor = solicitud.getParameter("valor");
        Cookie nuevaCookie = new Cookie("saludo", valor);
        nuevaCookie.setMaxAge(7 * 24 * 60 * 60);
        respuesta.addCookie(nuevaCookie);
        /* Envía encabezados al cliente, incluyendo el cookie y le pide
         * al navegador que deseche esta página y solicite JSPCookies.jsp. */
        respuesta.sendRedirect("JSPCookies.jsp");
    }
    /** Devuelve breve información del servlet.
     * @return Una breve información del servlet. */
    @Override
    public String getServletInfo() {
        return "versión: 1.0. Copyright 2009 Gilberto Pacheco Gallegos.";
    }
}
```

4.3. JSPSesion.jsp.

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<c:url var="registra" value="/SPSesion" />
<HTML>
  <HEAD>
    <TITLE>Datos de la Sesión</TITLE>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </HEAD>
  <BODY>
    <H2>Datos de la Sesión</H2>
    <c:if test="${!empty mensaje}">
      <P><STRONG>${mensaje}</STRONG></P>
    </c:if>
```

```
15      <FORM method="post" action="${registra}">
16        <TABLE>
17          <TR>
18            <TD>Identificador:</TD>
19            <TD>
20              <INPUT type="text" name="identificador" size="16"
21                     value="${sessionScope['identificador']}">
22            </TD>
23          </TR>
24          <TR>
25            <TD>Nombre:</TD>
26            <TD>
27              <INPUT type="text" name="nombre" size="16"
28                     value="${sessionScope.nombre}">
29            </TD>
30          </TR>
31          <TR>
32            <TD colspan="2" align="right">
33              <INPUT type="submit" value="Aceptar">
34            </TD>
35          </TR>
36        </TABLE>
37      </FORM>
38    </BODY>
39  </HTML>
```

4.4. SPSesion.

```
1   package servlets;
2
3   import java.io.IOException;
4   import javax.servlet.RequestDispatcher;
5   import javax.servlet.ServletException;
6   import javax.servlet.http.HttpServlet;
7   import javax.servlet.http.HttpServletRequest;
8   import javax.servlet.http.HttpServletResponse;
9   import javax.servlet.http.HttpSession;
10
11  public class SPSesion extends HttpServlet {
12      /** Maneja el método HTTP <code>POST</code>.
13       * @param solicitud la solicitud al servlet.
14       * @param respuesta la respuesta del servlet. */
15      @Override
16      protected void doPost(HttpServletRequest solicitud,
17              HttpServletResponse respuesta)
18              throws ServletException, IOException {
19          solicitud.setCharacterEncoding("UTF-8");
20          String identificador = solicitud.getParameter("identificador");
21          String nombre = solicitud.getParameter("nombre");
22          HttpSession sesión = solicitud.getSession();
23          sesión.setAttribute("identificador", identificador);
24          sesión.setAttribute("nombre", nombre);
```

```
25          solicitud.setAttribute("mensaje", "Datos de la sesión guardados.");
26          // Obtiene una referencia a la página que desea mostrar.
27          RequestDispatcher despachador =
28                  solicitud.getRequestDispatcher("/JSPSesion.jsp");
29          // Muestra la página, pero no añade nada después de mostrarla.
30          despachador.forward(solicitud, respuesta);
31      }
32      /** Devuelve breve información del servlet.
33       * @return Una breve información del servlet. */
34      @Override
35      public String getServletInfo() {
36          return "versión: 1.0. Copyright 2009 Gilberto Pacheco Gallegos.";
37      }
38  }
```

4.5. index.jsp.

```
1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
3  "http://www.w3.org/TR/html4/frameset.dtd">
4  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
5  <c:url var="toc" value="/JSPTOC.jsp" />
6  <c:url var="encabezado" value="/JSPEncabezado.jsp" />
7  <c:url var="cookies" value="/JSPCookies.jsp" />
8  <c:url var="sesión" value="/JSPSesion.jsp" />
9  <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>JSP Page</title>
13     </head>
14     <frameset rows="110, *">
15         <frame src="${encabezado}">
16         <frameset cols="200, *">
17             <frame src="${toc}">
18             <frame src="${cookies}" name="seleccion">
19         </frameset>
20         <noframes>
21             <h1>Cookies y Sesiones</h1>
22             <H2>Contenido</H2>
23             <UL>
24                 <LI><A href="${cookies}">Cookie</A></LI>
25                 <LI><A href="${sesión}">Sesiones</A></LI>
26             </UL>
27         </noframes>
28     </frameset>
29 </html>
```

4.6. JSPTOC.jsp.

```
1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```
"http://www.w3.org/TR/html4/loose.dtd">
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<c:url var="cookies" value="/JSPCookies.jsp" />
<c:url var="sesión" value="/JSPSesion.jsp" />
<HTML>
  <HEAD>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <TITLE>Contenido</TITLE>
  </HEAD>
  <BODY>
    <H2>Contenido</H2>
    <UL>
      <LI><A href="${cookies}" target="seleccion">Cookie</A></LI>
      <LI><A href="${sesión}" target="seleccion">Sesiones</A></LI>
    </UL>
  </BODY>
</HTML>
```

4.7. JSPEncabezado.jsp.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<c:url var="salir" value="/SPSalir" />
<html>
    <head>
        <title>Encabezado</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <h1>Cookies y Sesiones</h1>
        <P><a href="${salir}" target="_top">Salir</a>
    </body>
</html>
```

4.8. SPSalir.

```
package servlets;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class SPSalir extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest solicitud,
            HttpServletResponse respuesta) throws ServletException,
            IOException {
```

```
15          solicitud.getSession().invalidate();
16          PrintWriter writer = respuesta.getWriter();
17          writer.println("Adios");
18      }
19  }
```

4.9. web.xml.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5           http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
6      <servlet>
7          <servlet-name>SPCookies</servlet-name>
8          <servlet-class>servlets.SPCookies</servlet-class>
9      </servlet>
10     <servlet>
11         <servlet-name>SPSesion</servlet-name>
12         <servlet-class>servlets.SPSesion</servlet-class>
13     </servlet>
14     <servlet>
15         <servlet-name>SPSalir</servlet-name>
16         <servlet-class>servlets.SPSalir</servlet-class>
17     </servlet>
18     <servlet-mapping>
19         <servlet-name>SPCookies</servlet-name>
20         <url-pattern>/SPCookies</url-pattern>
21     </servlet-mapping>
22     <servlet-mapping>
23         <servlet-name>SPSesion</servlet-name>
24         <url-pattern>/SPSesion</url-pattern>
25     </servlet-mapping>
26     <servlet-mapping>
27         <servlet-name>SPSalir</servlet-name>
28         <url-pattern>/SPSalir</url-pattern>
29     </servlet-mapping>
30     <session-config>
31         <session-timeout>
32             30
33         </session-timeout>
34     </session-config>
35     <welcome-file-list>
36         <welcome-file>index.jsp</welcome-file>
37     </welcome-file-list>
38 </web-app>
```

5. Acceso a Bases de Datos.

Para comunicarse con la base de datos, normalmente se usa un **pool**, que es un conjunto de conexiones que se mantienen abiertas el mayor tiempo posible. Cuando un servlet necesita conectarse a la base de datos solicita una conexión y cuando termina, en vez de cerrarse se devuelve al pool. En Apache Tomcat los pool se configuran en el archivo "context.xml", pero el mecanismo cambia para otros servidores.

5.1. context.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/JSPBD">
  <!--driverClassName: Nombre del controlador de JDBC.
      url:             URL para conectarse a la base de datos.
      username:        Usuario en la base de datos.
      password:        Contraseña en la base de datos.
      maxActive:       Número máximo de conexiones activas que se pueden
                       guardar en este pool al mismo tiempo. Si es negativo,
                       no hay límite.
      maxWait:         Número máximo de milisegundos que el pool espera al
                       momento de crear una conexión, cuando no hay ninguna
                       disponible. Si se sobrepasa este tiempo, se lanza una
                       excepción. Si vale -1 se espera indefinidamente.
      testOnBorrow:    Indica si se prueba que las conexiones están abiertas
                       antes de tomarlas del pool. Si la validación falla, la
                       conexión se saca del pool y se prueba obtener otra. Si
                       vale true, se debe asignar el parámetro
                       validationQuery.
      validationQuery: Instrucción SQL que se utiliza para validar las
                       conexiones. Si se especifica, DEBE ser un SELECT que
                       devuelva al menos un renglón. -->
    <Resource auth="Container" name="jdbc/eventos"
              type="javax.sql.DataSource"
              driverClassName="org.apache.derby.jdbc.EmbeddedDriver"
              url="jdbc:derby:eventos;create=true" username="" password=""
              maxActive="10" maxIdle="5" testOnBorrow="true"
              validationQuery="SELECT 1 FROM SYS.SYSTABLES"/>
</Context>
```

5.2. JSPCreaBD.jsp.

La biblioteca JSTL contiene algunas etiquetas que permiten realizar labores simples con la base de datos.

```
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<c:url var="accion" value="/JSPCreaBD.jsp"/>
<c:catch var="error">
```

```
    <c:if test="${!empty param.crear}">
        <sql:update var="resultado" dataSource="jdbc/eventos">
            CREATE TABLE Puesto (
            clave INTEGER PRIMARY KEY,
            nombre VARCHAR(20) NOT NULL,
            sueldoMensual NUMERIC(8,2) NOT NULL,
            entrada TIME NOT NULL,
            salida TIME NOT NULL,
            creacion DATE NOT NULL)
        </sql:update>
        <c:set var="mensaje" value="Base de datos creada." scope="request"/>
    </c:if>
</c:catch>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
    <HEAD>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <TITLE>Creaci&oacute;n de la Base de Datos</TITLE>
    </HEAD>
    <BODY>
        <H2>Creaci&oacute;n de la Base de Datos</H2>
        <c:if test="${!empty error}">
            <P>
                <STRONG><c:out value="${error.localizedMessage}"/></STRONG>
            </P>
        </c:if>
        <c:if test="${!empty mensaje}">
            <P><STRONG><c:out value="${mensaje}"/></STRONG></P>
        </c:if>
        <FORM action="${accion}" method="post">
            <INPUT type="submit" name="crear" value="Crear" accesskey="C">
        </FORM>
    </BODY>
</HTML>
```

5.3. JSPConsulta.jsp.

```
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
<c:catch var="error">
    <sql:query var="resultado" scope="request" dataSource="jdbc/eventos">
        SELECT * FROM Puesto
    </sql:query>
</c:catch>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
    <HEAD>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
15          <TITLE>Puestos Registrados</TITLE>
16      </HEAD>
17      <BODY>
18          <H2>Puestos Registrados</H2>
19          <c:if test="${error != null}">
20              <P>
21                  <STRONG><c:out value="${error.localizedMessage}"/></STRONG>
22              </P>
23          </c:if>
24          <TABLE border="1">
25              <CAPTION>Recolectores</CAPTION>
26              <!-- column headers -->
27              <THEAD>
28                  <TR>
29                      <TH>Clave</TH>
30                      <TH>Nombre</TH>
31                      <TH>Sueldo Mensual</TH>
32                      <TH>Entrada</TH>
33                      <TH>Salida</TH>
34                      <TH>Creaci&oacute;n</TH>
35                  </TR>
36              </THEAD>
37              <TBODY>
38                  <c:forEach var="renglón" items="${resultado.rows}">
39                      <TR>
40                          <TD align="right">
41                              <c:out value="${renglón.clave}"/>
42                          </TD>
43                          <TD><c:out value="${renglón['nombre']}"/></TD>
44                          <TD align="right">
45                              <c:out value="${renglón.sueldoMensual}"/>
46                          </TD>
47                          <TD>
48                              <fmt:formatDate pattern="HH:mm"
49                                              value="${renglón.entrada}"/>
50                          </TD>
51                          <TD>
52                              <fmt:formatDate pattern="HH:mm"
53                                              value="${renglón.salida}"/>
54                          </TD>
55                          <TD>
56                              <fmt:formatDate pattern="dd/MM/yyyy"
57                                              value="${renglón.creacion}"/>
58                          </TD>
59                      </TR>
60                  </c:forEach>
61              </TBODY>
62          </TABLE>
63      </BODY>
64  </HTML>
```

5.4. JSPInsercion1.jsp.

Al probar este JSP, hay que notar que el manejo de errores no es bueno. Por ello se prefiere utilizar el estilo indicado en 5.5. y 5.6.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<fmt:requestEncoding value="UTF-8" />
<c:url var="accion" value="/JSPInsercion1.jsp" />
<c:catch var="error">
    <c:if test="${!empty param.agregar}">
        <fmt:parseDate pattern="HH:mm" var="horaDeEntrada" scope="request"
                       value="${param.entrada}" />
        <fmt:parseDate pattern="HH:mm" var="horaDeSalida" scope="request"
                       value="${param.salida}" />
        <fmt:parseDate pattern="dd/MM/yyyy" var="fechaDeCreacion"
                       scope="request" value="${param.creacion}" />
        <sql:update var="resultado" dataSource="jdbc/eventos">
            INSERT INTO Puesto
            (clave, nombre, sueldoMensual, entrada, salida, creacion)
            VALUES (?, ?, ?, ?, ?, ?)
            <sql:param value="${param.clave}"/>
            <sql:param value="${param.nombre}"/>
            <sql:param value="${param.sueldoMensual}"/>
            <sql:dateParam type="time" value="${horaDeEntrada}"/>
            <sql:dateParam type="time" value="${horaDeSalida}"/>
            <sql:dateParam type="date" value="${fechaDeCreacion}"/>
        </sql:update>
        <c:set var="mensaje" value="Puesto agregado." scope="request"/>
    </c:if>
</c:catch>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
    <HEAD>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <TITLE>Alta de Puesto</TITLE>
    </HEAD>
    <BODY>
        <H2>Alta de Puesto</H2>
        <c:if test="${!empty error}">
            <P><STRONG>${error.localizedMessage}</STRONG></P>
        </c:if>
        <c:if test="${!empty mensaje}">
            <P><STRONG><c:out value="${mensaje}"/></STRONG></P>
        </c:if>
        <FORM action="${accion}" method="post">
            <TABLE cellspacing="12">
                <TR>
                    <TD>
```

```
48                          <LABEL for="clave" accesskey="C">Clave:</LABEL>
49                      </TD>
50                      <TD>
51                          <INPUT type="text" id="clave" name="clave"
52                                 value="${param.clave}" size="9" maxlength="9">
53                      </TD>
54                  </TR>
55                  <TR>
56                      <TD>
57                          <LABEL for="nombre" accesskey="N">Nombre:</LABEL>
58                      </TD>
59                      <TD>
60                          <INPUT type="text" id="nombre" name="nombre"
61                                 value="${param.nombre}" size="20"
62                                 maxlength="20">
63                      </TD>
64                  </TR>
65                  <TR>
66                      <TD>
67                          <LABEL for="sueldoMensual" accesskey="S">
68                              Sueldo Mensual:
69                          </LABEL>
70                      </TD>
71                      <TD>
72                          <INPUT type="text" id="sueldoMensual"
73                                 name="sueldoMensual"
74                                 value="${param.sueldoMensual}" size="11"
75                                 maxlength="11">
76                      </TD>
77                  </TR>
78                  <TR>
79                      <TD>
70                          <LABEL for="entrada" accesskey="E">
81                              Hora de Entrada:
82                          </LABEL>
83                      </TD>
84                      <TD>
85                          <INPUT type="text" id="entrada" name="entrada"
86                                 value="${param.entrada}" size="5"
87                                 maxlength="5">
88                      </TD>
89                  </TR>
90                  <TR>
91                      <TD>
92                          <LABEL for="salida" accesskey="S">
93                              Hora de Salida:
94                          </LABEL>
95                      </TD>
96                      <TD>
97                          <INPUT type="text" id="salida" name="salida"
98                                 value="${param.salida}" size="5"
99                                 maxlength="5">
```

```
100                     </TD>
101                 </TR>
102                 <TR>
103                     <TD>
104                         <LABEL for="creacion" accesskey="F">
105                             Fecha de Creaci&oacute;n:
106                         </LABEL>
107                     </TD>
108                     <TD>
109                         <INPUT type="text" id="creacion" name="creacion"
100                                value="${param.creacion}" size="10"
111                                maxlength="10">
112                     </TD>
113                 </TR>
114                 <TR>
115                     <TD colspan="2" align="right">
116                         <INPUT type="submit" name="agregar" value="Agregar"
117                                accesskey="A">
118                         <INPUT type="reset" value="Reestablecer"
119                                accesskey="R">
120                     </TD>
121                 </TR>
122             </TABLE>
123         </FORM>
124     </BODY>
125 </HTML>
```

5.5. JSPInsercion2.jsp.

Normalmente se utiliza un servlet para manejar la lógica y el acceso a la base de datos. Funciona como un controlador en una arquitectura MVC (Modelo, Vista Controlador). La vista es un servlet que muestra los resultados.

```
 1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
 2 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
 3 <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
 4 <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
 5 <fmt:requestEncoding value="UTF-8" />
 6 <c:url var="accion" value="/SPInsercion2" />
 7 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 8 "http://www.w3.org/TR/html4/strict.dtd">
 9 <HTML>
10     <HEAD>
11         <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <TITLE>Alta de Puesto</TITLE>
13     </HEAD>
14     <BODY>
15         <H2>Alta de Puesto</H2>
16         <c:if test="${!empty mensaje}">
17             <P><STRONG><c:out value="${mensaje}"/></STRONG></P>
18         </c:if>
19         <FORM action="${accion}" method="post">
```

```
20              <TABLE cellspacing="12">
21                  <TR>
22                      <TD>
23                          <LABEL for="clave" accesskey="C">Clave:</LABEL>
24                      </TD>
25                      <TD>
26                          <INPUT type="text" id="clave" name="clave"
27                                 value="${param.clave}" size="9" maxlength="9">
28                      </TD>
29                  </TR>
30                  <TR>
31                      <TD>
32                          <LABEL for="nombre" accesskey="N">Nombre:</LABEL>
33                      </TD>
34                      <TD>
35                          <INPUT type="text" id="nombre" name="nombre"
36                                 value="${param.nombre}" size="20"
37                                 maxlength="20">
38                      </TD>
39                  </TR>
40                  <TR>
41                      <TD>
42                          <LABEL for="sueldoMensual" accesskey="S">
43                              Sueldo Mensual:
44                          </LABEL>
45                      </TD>
46                      <TD>
47                          <INPUT type="text" id="sueldoMensual"
48                                 name="sueldoMensual"
49                                 value="${param.sueldoMensual}" size="11"
50                                 maxlength="11">
51                      </TD>
52                  </TR>
53                  <TR>
54                      <TD>
55                          <LABEL for="entrada" accesskey="E">
56                              Hora de Entrada:
57                          </LABEL>
58                      </TD>
59                      <TD>
60                          <INPUT type="text" id="entrada" name="entrada"
61                                 value="${param.entrada}" size="5"
62                                 maxlength="5">
63                      </TD>
64                  </TR>
65                  <TR>
66                      <TD>
67                          <LABEL for="salida" accesskey="S">
68                              Hora de Salida:
69                          </LABEL>
70                      </TD>
71                      <TD>
```

```
72                          <INPUT type="text" id="salida" name="salida"
73                                 value="${param.salida}" size="5"
74                                 maxlength="5">
75                      </TD>
76                  </TR>
77                  <TR>
78                      <TD>
79                          <LABEL for="creacion" accesskey="F">
70                              Fecha de Creaci&oacute;n:
81                          </LABEL>
82                      </TD>
83                      <TD>
84                          <INPUT type="text" id="creacion" name="creacion"
85                                 value="${param.creacion}" size="10"
86                                 maxlength="10">
87                      </TD>
88                  </TR>
89                  <TR>
90                      <TD colspan="2" align="right">
91                          <INPUT type="submit" name="agregar" value="Agregar"
92                                 accesskey="A">
93                          <INPUT type="reset" value="Reestablecer"
94                                 accesskey="R">
95                      </TD>
96                  </TR>
97              </TABLE>
98          </FORM>
99      </BODY>
100 </HTML>
```

5.6. SPInsercion2.

```
1  package servlets;
2
3  import java.io.IOException;
4  import java.sql.Connection;
5  import java.sql.PreparedStatement;
6  import java.sql.SQLException;
7  import java.sql.SQLIntegrityConstraintViolationException;
8  import java.sql.Time;
9  import java.text.ParseException;
10 import java.text.SimpleDateFormat;
11 import java.util.Date;
12 import javax.annotation.Resource;
13 import javax.servlet.RequestDispatcher;
14 import javax.servlet.ServletException;
15 import javax.servlet.http.HttpServlet;
16 import javax.servlet.http.HttpServletRequest;
17 import javax.servlet.http.HttpServletResponse;
18 import javax.sql.DataSource;
19
20 public class SPInsercion2 extends HttpServlet {
```

```
21      private final SimpleDateFormat formatoFecha =
22              new SimpleDateFormat("dd/MM/yyyy");
23      private final SimpleDateFormat formatoHora =
24              new SimpleDateFormat("HH:mm");
25      @Resource(name = "jdbc/eventos")
26      private DataSource ds;
27      @Override
28      protected void doPost(HttpServletRequest solicitud,
29              HttpServletResponse respuesta)
30              throws ServletException, IOException {
31          /* Codificación de la respuesta. El navegador hace el submit de los
32           * datos en el mismo formato en que recibió JSPInsercion.jsp (UTF-8).
33           * Si no se utiliza esta instrucción, el servidor web supone que el
34           * navegador envía los datos con ISO-8859-1, lo cual es incorrecto en
35           * este caso. */
36          solicitud.setCharacterEncoding("UTF-8");
37          Connection c = null;
38          PreparedStatement ps = null;
39          Date horaDeEntrada = null;
40          Date horaDeSalida = null;
41          Date fechaDeCreacion = null;
42          try {
43              try {
44                  horaDeEntrada =
45                          formatoHora.parse(solicitud.getParameter("entrada"));
46              } catch (ParseException e) {
47                  throw new Exception("Formato incorrecto para la entrada");
48              }
49              try {
50                  horaDeSalida =
51                          formatoHora.parse(solicitud.getParameter("salida"));
52              } catch (ParseException e) {
53                  throw new Exception("Formato incorrecto para la salida");
54              }
55              try {
56                  fechaDeCreacion = formatoFecha.parse(
57                          solicitud.getParameter("creacion"));
58              } catch (ParseException e) {
59                  throw new Exception(
60                          "Formato incorrecto para la fecha de creación");
61              }
62              c = ds.getConnection();
63              ps = c.prepareStatement("INSERT INTO Puesto " +
64                      "(clave, nombre, sueldoMensual, entrada, salida," +
65                      " creacion) " +
66                      "VALUES (?, ?, ?, ?, ?, ?)");
67              ps.setInt(1, Integer.parseInt(
68                      solicitud.getParameter("clave")));
69              ps.setString(2, solicitud.getParameter("nombre"));
70              ps.setString(3, solicitud.getParameter("sueldoMensual"));
71              ps.setTime(4, new Time(horaDeEntrada.getTime()));
72              ps.setTime(5, new Time(horaDeSalida.getTime()));
```

```
 73              ps.setDate(6, new java.sql.Date(fechaDeCreacion.getTime()));
 74              ps.executeUpdate();
 75              solicitud.setAttribute("mensaje", "Puesto ageregado.");
 76          } catch (NumberFormatException e) {
 77              solicitud.setAttribute("mensaje",
 78                      "Formato incorrecto para la clave.");
 79          } catch (SQLIntegrityConstraintViolationException e) {
 70              solicitud.setAttribute("mensaje", "Clave duplicada.");
 81          } catch (SQLException e) {
 82              throw new ServletException(e);
 83          } catch (Exception e) {
 84              solicitud.setAttribute("mensaje", e.getLocalizedMessage());
 85          } finally {
 86              if (ps != null) {
 87                  try {
 88                      ps.close();
 89                  } catch (SQLException e) {}
 90              }
 91              if (c != null) {
 92                  try {
 93                      c.close();
 94                  } catch (SQLException e) {}
 95              }
 96          }
 97          // Obtiene una referencia a la página que desea mostrar.
 98          RequestDispatcher despachador =
 99                  solicitud.getRequestDispatcher("/JSPInsercion2.jsp");
100          // Muestra la página, pero no añade nada después de mostrarla.
101          despachador.forward(solicitud, respuesta);
102      }
103  }
```

5.7. JSPModificacion.jsp.

```
 1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
 2  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
 3  <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
 4  <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
 5  <fmt:requestEncoding value="UTF-8" />
 6  <c:url var="accion" value="/SPModificacion" />
 7  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 8  "http://www.w3.org/TR/html4/strict.dtd">
 9  <HTML>
10      <HEAD>
11          <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
12          <TITLE>Modificaci&oacute;n de Datos de Puesto</TITLE>
13      </HEAD>
14      <BODY>
15          <H2>Modificaci&oacute;n de Datos de Puesto</H2>
16          <c:if test="${!empty error}">
17              <P>
18                  <STRONG><c:out value="${error.localizedMessage}"/></STRONG>
```

```
19              </P>
20          </c:if>
21          <c:if test="${!empty mensaje}">
22              <P><STRONG><c:out value="${mensaje}"/></STRONG></P>
23          </c:if>
24          <FORM action="${accion}" method="post">
25              <TABLE cellspacing="12">
26                  <TR>
27                      <TD>
28                          <LABEL for="clave" accesskey="C">Clave:</LABEL>
29                      </TD>
30                      <TD>
31                          <INPUT type="text" id="clave" name="clave"
32                                 value="${clave}" size="9" maxlength="9">
33                          <INPUT type="submit" name="buscar" value="Buscar"
34                                 accesskey="B">
35                      </TD>
36                  </TR>
37                  <TR>
38                      <TD>
39                          <LABEL for="nombre" accesskey="N">Nombre:</LABEL>
40                      </TD>
41                      <TD>
42                          <INPUT type="text" id="nombre" name="nombre"
43                                 value="${nombre}" size="20"
44                                 maxlength="20">
45                      </TD>
46                  </TR>
47                  <TR>
48                      <TD>
49                          <LABEL for="sueldoMensual" accesskey="S">
50                              Sueldo Mensual:
51                          </LABEL>
52                      </TD>
53                      <TD>
54                          <INPUT type="text" id="sueldoMensual"
55                                 name="sueldoMensual"
56                                 value="${sueldoMensual}" size="11"
57                                 maxlength="11">
58                      </TD>
59                  </TR>
60                  <TR>
61                      <TD>
62                          <LABEL for="entrada" accesskey="E">
63                              Hora de Entrada:
64                          </LABEL>
65                      </TD>
66                      <TD>
67                          <INPUT type="text" id="entrada" name="entrada"
68                                 value="${entrada}" size="5"
69                                 maxlength="5">
70                      </TD>
```

```
71                  </TR>
72                  <TR>
73                      <TD>
74                          <LABEL for="salida" accesskey="S">
75                              Hora de Salida:
76                          </LABEL>
77                      </TD>
78                      <TD>
79                          <INPUT type="text" id="salida" name="salida"
70                                 value="${salida}" size="5"
81                                 maxlength="5">
82                      </TD>
83                  </TR>
84                  <TR>
85                      <TD>
86                          <LABEL for="creacion" accesskey="F">
87                              Fecha de Creaci&oacute;n:
88                          </LABEL>
89                      </TD>
90                      <TD>
91                          <INPUT type="text" id="creacion" name="creacion"
92                                 value="${creacion}" size="10"
93                                 maxlength="10">
94                      </TD>
95                  </TR>
96                  <TR>
97                      <TD colspan="2" align="right">
98                          <INPUT type="submit" name="guardar" value="Guardar"
99                                 accesskey="G">
100                         <INPUT type="reset" value="Reestablecer"
101                                accesskey="R">
102                     </TD>
103                 </TR>
104             </TABLE>
105         </FORM>
106     </BODY>
107 </HTML>
```

5.8. SPModificacion.

```
1  package servlets;
2
3  import java.io.IOException;
4  import java.sql.Connection;
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.sql.Statement;
9  import java.sql.Time;
10 import java.text.ParseException;
11 import java.text.SimpleDateFormat;
12 import java.util.Date;
```

```
13 import javax.annotation.Resource;
14 import javax.servlet.RequestDispatcher;
15 import javax.servlet.ServletException;
16 import javax.servlet.http.HttpServlet;
17 import javax.servlet.http.HttpServletRequest;
18 import javax.servlet.http.HttpServletResponse;
19 import javax.sql.DataSource;
20
21 public class SPModificacion extends HttpServlet {
22     private final SimpleDateFormat formatoFecha =
23             new SimpleDateFormat("dd/MM/yyyy");
24     private final SimpleDateFormat formatoHora =
25             new SimpleDateFormat("HH:mm");
26     @Resource(name = "jdbc/eventos")
27     private DataSource ds;
28     @Override
29     protected void doPost(HttpServletRequest solicitud,
30             HttpServletResponse respuesta)
31             throws ServletException, IOException {
32         solicitud.setCharacterEncoding("UTF-8");
33         if (solicitud.getParameter("buscar") != null) {
34             busca(solicitud, respuesta);
35         } else if (solicitud.getParameter("guardar") != null) {
36             guarda(solicitud, respuesta);
37         }
38     }
39     private void busca(HttpServletRequest solicitud,
40             HttpServletResponse respuesta)
41             throws ServletException, IOException {
42         Connection c = null;
43         PreparedStatement ps = null;
44         ResultSet rs = null;
45         try {
46             int clave = Integer.parseInt(solicitud.getParameter("clave"));
47             c = ds.getConnection();
48             ps = c.prepareStatement("SELECT * FROM Puesto " +
49                     "WHERE clave = ?");
50             ps.setInt(1, clave);
51             rs = ps.executeQuery();
52             if (rs.next()) {
53                 solicitud.setAttribute("clave", rs.getObject("clave"));
54                 solicitud.setAttribute("nombre", rs.getObject("nombre"));
55                 solicitud.setAttribute("sueldoMensual",
56                         rs.getObject("sueldoMensual"));
57                 solicitud.setAttribute("entrada",
58                         formatoHora.format(rs.getTime("entrada")));
59                 solicitud.setAttribute("salida",
60                         formatoHora.format(rs.getTime("salida")));
61                 solicitud.setAttribute("creacion",
62                         formatoFecha.format(rs.getDate("creacion")));
63             } else {
64                 solicitud.setAttribute("mensaje",
```

```
65                          "Recolector no encontrado.");
66              }
67          } catch (NumberFormatException e) {
68              solicitud.setAttribute("mensaje",
69                      "Formato incorrecto para la clave.");
70          } catch (SQLException e) {
71              throw new ServletException(e);
72          } finally {
73              cierra(rs, ps, c);
74          }
75          muestraVista(solicitud, respuesta);
76      }
77      private void guarda(HttpServletRequest solicitud,
78              HttpServletResponse respuesta)
79              throws ServletException, IOException {
70          Connection c = null;
81          PreparedStatement ps = null;
82          Date horaDeEntrada = null;
83          Date horaDeSalida = null;
84          Date fechaDeCreacion = null;
85          solicitud.setAttribute("clave", solicitud.getParameter("clave"));
86          solicitud.setAttribute("nombre", solicitud.getParameter("nombre"));
87          solicitud.setAttribute("sueldoMensual",
88                  solicitud.getParameter("sueldoMensual"));
89          solicitud.setAttribute("entrada", solicitud.getParameter("entrada"));
90          solicitud.setAttribute("salida", solicitud.getParameter("salida"));
91          solicitud.setAttribute("creacion",
92                  solicitud.getParameter("creacion"));
93          try {
94              try {
95                  horaDeEntrada =
96                          formatoHora.parse(solicitud.getParameter("entrada"));
97              } catch (ParseException e) {
98                  throw new Exception("Formato incorrecto para la entrada");
99              }
100             try {
101                 horaDeSalida =
102                         formatoHora.parse(solicitud.getParameter("salida"));
103             } catch (ParseException e) {
104                 throw new Exception("Formato incorrecto para la salida");
105             }
106             try {
107                 fechaDeCreacion = formatoFecha.parse(
108                         solicitud.getParameter("creacion"));
109             } catch (ParseException e) {
110                 throw new Exception(
111                         "Formato incorrecto para la fecha de creación");
112             }
113             c = ds.getConnection();
114             int clave = Integer.parseInt(solicitud.getParameter("clave"));
115             ps = c.prepareStatement("UPDATE Puesto " +
116                     "SET nombre = ?, sueldoMensual = ?, entrada = ?, " +
```

```
117                     "salida = ?, creacion = ? " +
118                     "WHERE clave = ?");
119             ps.setString(1, solicitud.getParameter("nombre"));
120             ps.setString(2, solicitud.getParameter("sueldoMensual"));
121             ps.setTime(3, new Time(horaDeEntrada.getTime()));
122             ps.setTime(4, new Time(horaDeSalida.getTime()));
123             ps.setDate(5, new java.sql.Date(fechaDeCreacion.getTime()));
124             ps.setInt(6, clave);
125             ps.executeUpdate();
126             solicitud.setAttribute("mensaje", "Datos guardados.");
127         } catch (NumberFormatException e) {
128             solicitud.setAttribute("mensaje",
129                     "Formato incorrecto para la clave.");
130         } catch (SQLException e) {
131             throw new ServletException(e);
132         } catch (Exception e) {
133             solicitud.setAttribute("mensaje", e.getLocalizedMessage());
134         } finally {
135             cierra(null, ps, c);
136         }
137         muestraVista(solicitud, respuesta);
138     }
139     private void muestraVista(HttpServletRequest solicitud,
140             HttpServletResponse respuesta)
141             throws ServletException, IOException {
142         // Obtiene una referencia a la página que desea mostrar.
143         RequestDispatcher despachador =
144                 solicitud.getRequestDispatcher("/JSPModificacion.jsp");
145         // Muestra la página, pero no añade nada después de mostrarla.
146         despachador.forward(solicitud, respuesta);
147     }
148     private void cierra(ResultSet rs, Statement s, Connection c) {
149         if (rs != null) {
150             try {
151                 rs.close();
152             } catch (SQLException e) {}
153         }
154         if (s != null) {
155             try {
156                 s.close();
157             } catch (SQLException e) {}
158         }
159         if (c != null) {
160             try {
161                 c.close();
162             } catch (SQLException e) {}
163         }
164     }
165 }
```

5.9. JSPEliminacion.jsp.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<fmt:requestEncoding value="UTF-8" />
<c:url var="accion" value="/SPEliminacion"/>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
    <HEAD>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <TITLE>Eliminaci&oacute;n de Puesto</TITLE>
    </HEAD>
    <BODY>
        <H2>Eliminaci&oacute;n de Puesto</H2>
        <c:if test="${!empty error}">
            <P>
                <STRONG><c:out value="${error.localizedMessage}"/></STRONG>
            </P>
        </c:if>
        <c:if test="${!empty mensaje}">
            <P><STRONG><c:out value="${mensaje}"/></STRONG></P>
        </c:if>
        <FORM action="${accion}" method="post">
            <TABLE cellspacing="12">
                <TR>
                    <TD>
                        <LABEL for="clave" accesskey="C">Clave:</LABEL>
                    </TD>
                    <TD>
                        <INPUT type="text" id="clave" name="clave"
                               value="${clave}" size="9" maxlength="9">
                        <INPUT type="submit" name="buscar" value="Buscar"
                               accesskey="B">
                    </TD>
                </TR>
                <TR>
                    <TD><LABEL>Nombre:</LABEL></TD>
                    <TD><c:out value="${nombre}"/></TD>
                </TR>
                <TR>
                    <TD><LABEL>Sueldo Mensual:</LABEL></TD>
                    <TD><c:out value="${sueldoMensual}"/>
                    </TD>
                </TR>
                <TR>
                    <TD><LABEL>Hora de Entrada:</LABEL></TD>
                    <TD><c:out value="${entrada}"/></TD>
                </TR>
                <TR>
```

```
51                      <TD><LABEL>Hora de Salida:</LABEL></TD>
52                      <TD><c:out value="${salida}"/></TD>
53                  </TR>
54                  <TR>
55                      <TD><LABEL>Fecha de Creaci&oacute;n:</LABEL></TD>
56                      <TD><c:out value="${creacion}"/></TD>
57                  </TR>
58                  <TR>
59                      <TD colspan="2" align="right">
60                          <INPUT type="submit" name="eliminar" value="Eliminar"
61                                 accesskey="E">
62                      </TD>
63                  </TR>
64              </TABLE>
65          </FORM>
66      </BODY>
67  </HTML>
```

5.10. SPEliminacion.

```
 1  package servlets;
 2
 3  import java.io.IOException;
 4  import java.sql.Connection;
 5  import java.sql.PreparedStatement;
 6  import java.sql.ResultSet;
 7  import java.sql.SQLException;
 8  import java.sql.Statement;
 9  import java.text.SimpleDateFormat;
10  import javax.annotation.Resource;
11  import javax.servlet.RequestDispatcher;
12  import javax.servlet.ServletException;
13  import javax.servlet.http.HttpServlet;
14  import javax.servlet.http.HttpServletRequest;
15  import javax.servlet.http.HttpServletResponse;
16  import javax.sql.DataSource;
17
18  public class SPEliminacion extends HttpServlet {
19      private final SimpleDateFormat formatoFecha =
20              new SimpleDateFormat("dd/MM/yyyy");
21      private final SimpleDateFormat formatoHora =
22              new SimpleDateFormat("HH:mm");
23      @Resource(name = "jdbc/eventos")
24      private DataSource ds;
25      @Override
26      protected void doPost(HttpServletRequest solicitud,
27              HttpServletResponse respuesta)
28              throws ServletException, IOException {
29          solicitud.setCharacterEncoding("UTF-8");
30          if (solicitud.getParameter("buscar") != null) {
31              busca(solicitud, respuesta);
32          } else if (solicitud.getParameter("eliminar") != null) {
```

```
33              elimina(solicitud, respuesta);
34          }
35      }
36      private void busca(HttpServletRequest solicitud,
37              HttpServletResponse respuesta)
38              throws ServletException, IOException {
39          Connection c = null;
40          PreparedStatement ps = null;
41          ResultSet rs = null;
42          try {
43              int clave = Integer.parseInt(solicitud.getParameter("clave"));
44              c = ds.getConnection();
45              ps = c.prepareStatement("SELECT * FROM Puesto " +
46                      "WHERE clave = ?");
47              ps.setInt(1, clave);
48              rs = ps.executeQuery();
49              if (rs.next()) {
50                  solicitud.setAttribute("clave", rs.getObject("clave"));
51                  solicitud.setAttribute("nombre", rs.getObject("nombre"));
52                  solicitud.setAttribute("sueldoMensual",
53                          rs.getObject("sueldoMensual"));
54                  solicitud.setAttribute("entrada",
55                          formatoHora.format(rs.getTime("entrada")));
56                  solicitud.setAttribute("salida",
57                          formatoHora.format(rs.getTime("salida")));
58                  solicitud.setAttribute("creacion",
59                          formatoFecha.format(rs.getDate("creacion")));
60              } else {
61                  solicitud.setAttribute("mensaje",
62                          "Recolector no encontrado.");
63              }
64          } catch (NumberFormatException e) {
65              solicitud.setAttribute("mensaje",
66                      "Formato incorrecto para la clave.");
67          } catch (SQLException e) {
68              throw new ServletException(e);
69          } finally {
70              cierra(rs, ps, c);
71          }
72          muestraVista(solicitud, respuesta);
73      }
74      private void elimina(HttpServletRequest solicitud,
75              HttpServletResponse respuesta)
76              throws ServletException, IOException {
77          Connection c = null;
78          PreparedStatement ps = null;
79          try {
70              c = ds.getConnection();
81              int clave = Integer.parseInt(solicitud.getParameter("clave"));
82              ps = c.prepareStatement("DELETE FROM Puesto " +
83                      "WHERE clave = ?");
84              ps.setInt(1, clave);
```

```
 85              ps.executeUpdate();
 86              solicitud.setAttribute("mensaje", "Datos borrados.");
 87              muestraVista(solicitud, respuesta);
 88          } catch (NumberFormatException e) {
 89              throw new ServletException(
 90                      "Formato incorrecto para la clave.");
 91          } catch (SQLException e) {
 92              throw new ServletException(e);
 93          } finally {
 94              cierra(null, ps, c);
 95          }
 96      }
 97      private void muestraVista(HttpServletRequest solicitud,
 98              HttpServletResponse respuesta)
 99              throws ServletException, IOException {
100          // Obtiene una referencia a la página que desea mostrar.
101          RequestDispatcher despachador =
102                  solicitud.getRequestDispatcher("/JSPEliminacion.jsp");
103          // Muestra la página, pero no añade nada después de mostrarla.
104          despachador.forward(solicitud, respuesta);
105      }
106      private void cierra(ResultSet rs, Statement s, Connection c) {
107          if (rs != null) {
108              try {
109                  rs.close();
110              } catch (SQLException e) {}
111          }
112          if (s != null) {
113              try {
114                  s.close();
115              } catch (SQLException e) {}
116          }
117          if (c != null) {
118              try {
119                  c.close();
120              } catch (SQLException e) {}
121          }
122      }
123  }
```

5.11. web.xml.

El pool también debe declararse en este archivo.

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
 3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 5           http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
 6      <servlet>
 7          <servlet-name>SPInsercion2</servlet-name>
 8          <servlet-class>servlets.SPInsercion2</servlet-class>
```

```
    </servlet>
    <servlet>
        <servlet-name>SPModificacion</servlet-name>
        <servlet-class>servlets.SPModificacion</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>SPEliminacion</servlet-name>
        <servlet-class>servlets.SPEliminacion</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SPInsercion2</servlet-name>
        <url-pattern>/SPInsercion2</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>SPModificacion</servlet-name>
        <url-pattern>/SPModificacion</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>SPEliminacion</servlet-name>
        <url-pattern>/SPEliminacion</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <resource-ref>
        <description>Conexión a la base de datos</description>
        <res-ref-name>jdbc/reciclado</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
    </resource-ref>
</web-app>
```

5.12. index.jsp.

```
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<c:url var="icono" value="/img/favicon.ico" />
<c:url var="encabezado" value="/CPEncabezado.html"/>
<c:url var="contenido" value="/JSPTOC.jsp"/>
<c:url var="consulta" value="/JSPConsulta.jsp"/>
<c:url var="crear" value="/JSPCreaBD.jsp"/>
<c:url var="inserción" value="/JSPInsercion.jsp"/>
<c:url var="modificación" value="/JSPModificacion.jsp"/>
<c:url var="eliminación" value="/JSPEliminacion.jsp"/>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<HTML>
    <HEAD>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <LINK rel="shortcut icon" href="${icono}">
        <TITLE>Conexi&oacute;n a Base de Datos</TITLE>
    </HEAD>
```

```
    <FRAMESET rows="15%, 75%">
        <FRAME src="${encabezado}">
        <FRAMESET cols="20%, 80%">
            <FRAME src="${contenido}">
            <FRAME src="${consulta}" name="dinamico">
        </FRAMESET>
        <NOFRAMES>
            <H1>Contenido</H1>
            <UL>
                <LI><A href="${crear}">Crear BD</A></LI>
                <LI><A href="${consulta}">Consulta</A></LI>
                <LI><A href="${inserción}">Inserci&oacute;n</A></LI>
                <LI><A href="${modificación}">Modificaci&oacute;n</A></LI>
                <LI><A href="${eliminación}">Eliminaci&oacute;n</A></LI>
            </UL>
        </NOFRAMES>
    </FRAMESET>
</HTML>
```

5.13. JSPTOC.jsp.

```
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<c:url var="crear" value="/JSPCreaBD.jsp"/>
<c:url var="consulta" value="/JSPConsulta.jsp"/>
<c:url var="inserción1" value="/JSPInsercion1.jsp"/>
<c:url var="inserción2" value="/JSPInsercion2.jsp"/>
<c:url var="modificación" value="/JSPModificacion.jsp"/>
<c:url var="eliminación" value="/JSPEliminacion.jsp"/>
<!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>Contenido</TITLE>
    </HEAD>
    <BODY>
        <H2>Contenido</H2>
        <UL>
            <LI><A href="${crear}" target="dinamico">Crear BD</A></LI>
            <LI><A href="${consulta}" target="dinamico">Consulta</A></LI>
            <LI>
                <A href="${inserción1}" target="dinamico">
                    Inserci&oacute;n con JSP
                </A>
            </LI>
            <LI>
                <A href="${inserción2}" target="dinamico">
                    Inserci&oacute;n
                </A>
            </LI>
            <LI>
```

```
32                  <A href="${modificación}" target="dinamico">
33                      Modificaci&oacute;n
34                  </A>
35              </LI>
36              <LI>
37                  <A href="${eliminación}" target="dinamico">
38                      Eliminaci&oacute;n
39                  </A>
40              </LI>
41          </UL>
42      </BODY>
43  </HTML>
```

5.14. CPEncabezado.html.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
2  "http://www.w3.org/TR/html4/strict.dtd">
3  <HTML>
4    <HEAD>
5      <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
6      <TITLE>Conexi&oacute;n a Base de Datos</TITLE>
7    </HEAD>
8    <BODY>
9      <H1>Conexi&oacute;n a Base de Datos</H1>
10   </BODY>
11 </HTML>
```