Los servlets como puente de comunicación cliente-servidor empleando java

Servlets Like Communication Bridge Client-Server through Java

Carlos Alberto Vanegas
Ingeniero de sistemas de la Universidad Incca de Colombia, Especialista en Ingeniería de Software de la Universidad Distrital Francisco José de Caldas y Magíster en Ingeniería de sistemas, Universidad Nacional de Colombia, Docente tiempo completo de la Universidad Distrital Francisco José de Caldas adscrito a la Facultad Tecnológica.
cavanegas@udistrital.edu.co

Clasificación del artículo: reflexión

Fecha de recepción: 28 de noviembre de 2005

Fecha de aceptación: 2 de junio de 2006

***Palabras clave*:** enseñanza de la programación, protocolo, navegador, contenedor, servidor, interfase
***Key words*:** programming teaching, protocol, browse, container, server, interface.

Resumen

En este artículo se expone la creación de *servlets* como puente de comunicación entre clientes y servidores utilizando el protocolo HTTP para la creación de páginas *Web*. Aquí se realiza una comparación entre un *servlet* y un CGI, se explica el ciclo de vida de los *servlets* y los paquetes necesarios para su creación; también se hace una breve introducción al lenguaje HTML, se presenta una visión general del contenedor de *servlets* TOMCAT versión 1.4.12. y se muestran ejemplos prácticos que explican los pasos para la creación de *servlets.*

Abstract

This paper presents the creation process of *servlets* like a communication bridge between clients and servers using the HTTP protocol for web pages creation. It makes a comparison between *servlets* and CGI and shows the *servlets* life cycle and the necessary packages for their creation; a brief introduction about HTML language and a general vision of the TOMCAT *servlets* container, version 1.4.12, are included also. Finally, practical examples to explain the steps for *servlets* creation are developed.

1. Introducción

En la actualidad es común escuchar dos términos en el mundo de la informática: Internet y *World Wibe Web (Web)*. Internet es la red de computadores más extensa y compleja del mundo; la *Web* facilita su uso y el manejo de herramientas multimedia e hipertexto[1], empleando el protocolo http[2].

Los *servlets* son clases de Java que amplían la funcionalidad de un servidor *Web* mediante la generación dinámica de páginas *Web* [1]. Además, muestran la comunicación entre clientes y servidores a través del protocolo http y, por tanto, permiten crear páginas activas, esto es, aquellas cuyas respuestas varían en función de los datos que proporciona el cliente y la situación de contexto existente. Por ejemplo, se puede informar a un cliente acerca de los vuelos disponibles en un momento dado (situación de contexto), para un origen y un destino seleccionados (datos que proporciona el cliente) [2]. En forma gráfica, el funcionamiento de los *servlets* puede definirse en seis pasos enunciados a continuación:

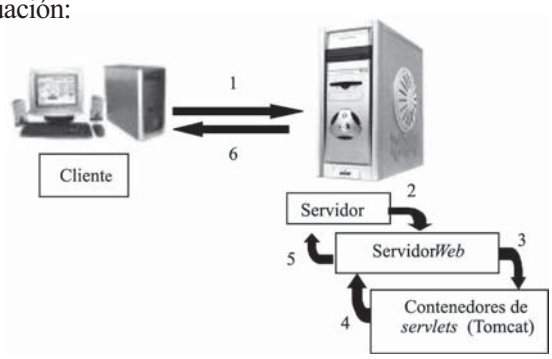

Figura 1. Funcionamiento de los *servlets*.

(1) El cliente solicita una petición http usando un navegador[3].

(2) La petición y los datos respectivos llegan al servidor a través de la red.

(3) El servidor *Web* ejecuta un *servlet* por medio del programa contenedor de *servlets.*

(4) Ejecutado el *servlet*, los resultados se envían en forma de página *Web* al servidor *Web.*

(5) El servidor *Web* envía la página *Web* de resultados a través del servidor.

(6) La página *Web* de respuesta llega al cliente, quien la visualiza usando el navegador.

2. Los *servlets* frente a los CGI

El CGI (Common Gateway Interface) es una norma para establecer comunicación entre un servidor *Web* y un programa, de tal modo que este último puede interactuar con Internet. Se trata de un programa que se ejecuta en tiempo real, en respuesta a una solicitud de un navegador; de esta forma, su propósito es proveer interactividad a un sitio *Web*. Los *servlets* realizan las mismas operaciones de un CGI pero son más eficientes, fáciles de usar, poderosos y portables.

En términos de eficiencia, los CGI ejecutan un nuevo proceso por cada solicitud http; empleando *servlets*, la máquina virtual de Java permanece ejecutada y cada nuevo proceso es manejado por un hilo[4] *(thread).* De igual forma, si se necesitan *N* peticiones de un CGI el código se carga N veces en memoria; con los *servlets* existen *N* hilos, pero solo una copia de la clase *Servlet.*

Una potencialidad de los *servlets* es que pueden interactuar directamente con el servidor *Web*, simplificando las operaciones que se realizan para la búsqueda de imágenes y otros datos. También tiene la facilidad de conexión con bases en datos, lo cual es difícil o incluso imposible con un CGI. Otra

[1] El hipertexto es un texto no secuencial compuesto por un bloque de palabras (o imágenes) vinculadas electrónicamente por múltiples caminos, descrita por palabras claves como enlace.
[2] http es un protocolo de transferencia de hipertexto.
[3] Un navegador es una aplicación que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores *Web* de todo el mundo a través de Internet.

[4] Los hilos son objetos o clases que permiten manipular el tiempo de un proceso que se encuentra activo; este proceso se puede iniciar, detener, pausar o volver a reiniciar.

característica relevante es la portabilidad: los *servlets* escritos en el servidor Tomcat pueden ser ejecutados sin ser modificados en Microsoft IIS o WebStar.

3. Ciclo de vida de un *servlet*

El ciclo de vida de un *servlet* empieza cuando el contenedor lo carga en memoria, en general en respuesta a la primera petición que recibe. Antes de que el *servlet* pueda encargarse de la petición, el contenedor invoca al método *INIT*; cuando *init* termina de ejecutarse el *servlet* puede responder a su primera petición. Todas las peticiones son manejadas por el método *service*, el cual recibe la petición, la procesa y envía al cliente. Cuando el contenedor termina el *servlet* se hace una llamada al método *destroy* para liberar recursos [3].

4. Paquetes empleados para crear *servlets* y otros requisitos del proceso

Los paquetes empleados en la creación de *servlets* son: *javax.servlet* y *javax.servlet.http*; en ellos se encuentran:

- La interfaz *Servlet*, del paquete *javax.servlet*, proporciona los métodos: *init*(), *destroy*() y *service*().
- La clase *HttpServlet*, del paquete *javax.servlet.http*, es la utilizada en casi todas las implementaciones de *servlets*. Además de conservar el método *service*() proporciona los siguientes métodos: *doGet*(), *doPost*(), doPut(), *doDelete*(), *doHead*(), *doOptions*() y *doTrace*().
- La interfaz *ServletRequest* se encarga de encapsular la comunicación desde el cliente al servidor. Algunos de sus métodos son: getParameter(), getProtocol(), getServerName(), getServerPort(), getRemoteAddr() y getRemoteHost().
- La interfaz *ServletResponse* se encarga de encapsular la comunicación que va desde el *servlet* hacia el cliente; sus métodos más comunes son: getWriter(), setContentLength() y setContentType().

Para la creación de *servlets* es necesario que el programador conozca los conceptos básicos del lenguaje de programación Java; además, debe tener conocimiento del lenguaje HTML y tener instalado el contenedor de *servlets* Tomcat. En los ejemplos de aplicación se utiliza el manejador de base de datos Access; en este tipo de casos además es necesario saber crear tablas en aquella herramienta.

5. Lenguaje de formato de documentos de hipertexto (HTML)

HTML es el lenguaje con que se definen las páginas *Web*; se trata de un conjunto de etiquetas que sirven para definir cómo se presenta el texto y otros elementos de la página. Las páginas *Web* pueden ser vistas por el usuario mediante una aplicación llamada navegador; por tanto, puede decirse que HTML es el lenguaje usado por los navegadores para mostrar las páginas *Web* al usuario, y hoy día es la interfaz más extendida en Internet.

6. El servidor Apache Tomcat

Jakarta Tomcat es un software que se instala en un servidor, con el fin de permitir la ejecución de los *servlets* de Java; fue desarrollado en el marco del proyecto *Jakarta* en la Apache Software Foundation.[5] Para el desarrollo de los ejercicios contenidos en este artículo puede descargarse la versión *jakarta-tomcat-4.1.12-LE-jdk14.exe*, o una más reciente; este archivo le permitirá realizar la instalación completa del Tomcat 4.1.12, que servirá para ejecutar los *servlets* propuestos.

Después de instalado, *Tomcat* inicia el servidor con el archivo *startup*, de la carpeta *bin*; para detenerlo se ejecuta *shutdown*, de la carpeta *bin*. Si es instalado en Windows mediante la opción *Inicio-Programas*

[5] Para descargar una versión gratuita se recomienda la siguiente dirección electrónica: http://jakarta.apache.org/site/downloads/index.html

se encuentra una opción llamada *apache tomcat*, que contiene las alternativas de inicio (*start tomcat*) y finalización (*stop tomcat*). Para verificar que *Tomcat* se esté ejecutando y pueda responder a peticiones

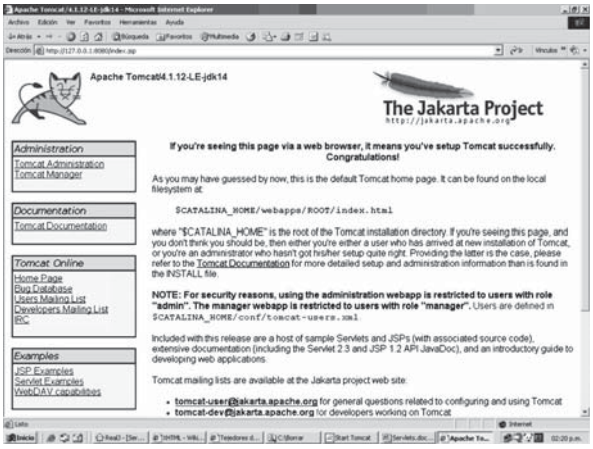

Figura 2. Página inicial de documentación *Tomcat*.

debe abrirse el navegador y escribir la siguiente dirección: http://locathost:8080 ó http://127.0.0.1:8080; luego se visualizará la pantalla de la figura 2.

- *Entorno.* Con la instalación de *Tomcat* se definen las variables de entorno *java_home* y *catalina_home*. La primera se direcciona a la carpeta que contiene el *kit* de desarrollo de Java *(jsdk*); la segunda a la carpeta en que el software fue instalado. Dado que *Tomcat* fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual de Java[6].
- *Descriptor de despliegue*. Para configurar la aplicación *Web* para el manejo de las peticiones de los *servlets*, estas últimas se realizan en un archivo descriptor de despliegue llamado *web.xml*, el cual especifica parámetros de configuración como el nombre usado (alias) para invocar el *servlet*, su descripción, el nombre de la clase y una asociación de *servlet*; está última es la ruta que hace que el contenedor respectivo invoque el *servlet.*

7. Ejemplos de *servlets*

Los siguientes ejemplos se presentan con propósitos didácticos y demostrativos. En cada caso se explicarán las líneas que se emplean por primera vez. Antes de iniciar se recomienda ejecutar el siguiente procedimiento:

(1) Instalado *Tomcat*, crear la siguiente estructura de carpetas en la subcarpeta *webapps*:

examples
servlets
WEB-INF
Classes

- *Servlets*: para todos los documentos *.hmtl*
- *WEB-INF*: para alojar el descriptor de despliegue *web.xml*
- *Classes*: se guarda el servlet de Java (los archivos con extensión *.java* y *.class*. Se aclara que no es indispensable tener el programa fuente *.java* es esta carpeta.

(2) Crear el *servlet* de Java y adicionar el archivo *servlet.jar*, que se encuentra en la carpeta *common/lib* de *Tomcat* y permite la compilación del *servlets*; como editor del programa fuente puede utilizarse el *freejava*, *realj* ó *jcreator*; estos facilitan la creación de los *servlets*, son fáciles de manejar y de libre distribución.[7]

[6] La máquina virtual de Java es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario, el cual es generado por el compilador del lenguaje Java.

[7] En las siguientes direcciones se puede encontrar el programa y la documentación correspondiente:
Realj: http://www.udesarrollo.cl/cursos/freeware/freeware.htm
Jcreator: http://www.entrebits.com/descargas/programas/jcreator/
FreeJava: http://freejava.softonic.com/ie/7115

7.1. Ejercicio 1. Realizar un servlet que permite enviar al navegador del cliente un texto

- Crear programa fuente (*servlet*).

```
1       import javax.servlet.*;
2       import javax.servlet.http.*;
3       import java.io.*;
4       public class ServletBienvenida extends HttpServlet      {
5        protected void doGet(HttpServletRequest peticion,
HttpServletResponse respuesta) throws ServletException, IOException      {
6  respuesta.setContentType(«text/html»);
7  PrintWriter salida=respuesta.getWriter();
8  salida.println(«<head>»);
9  salida.println(«<title> Llamado al Servlet de Java</title>»);
10  salida.println(«</head>»);
11  salida.println(«<body>»);
12  salida.println(«<h1>! Bienvenido a los Servlets - Java !</h1>»);
13  salida.println(«</body>»);
14  salida.println(«</html>»);
15  salida.close(); }}
```

- *Líneas 1 y 2*: importan todas las clases de los paquetes *servlet* y *servlet.http*
- *Línea 3*: importa el paquete *java.io*, necesario para las operaciones de entrada-salida de datos y resultados a través de los objetos *HttpServletRequest* y *HttpServletResponse*.
- *Línea 4*: crea la clase *ServletBienvenida* y heredar de la clase *HttpServlet*.
- *Línea 5*: implementa el método *doGet*, que recibe dos argumentos: un objeto *HttpServletRequest* y uno *HttpServletResponse*; el primero representa la petición del cliente y el segundo la respuesta del servidor al cliente. Si un método *doGet* no puede manejar la petición del cliente lanza una excepción tipo *javax. servlet. ServletException*; asimismo, si encuentra un error durante el procesamiento de los flujos de información (leer del cliente o escribir al cliente), lanza una excepción tipo *javax.io.IOException*.
- *Línea 6*: empleando el método *setContentType*, de la interfaz *HttpServletResponse*, indica el tipo de dato utilizado como respuesta; en este caso es tipo *text/html*, para indicar al navegador que la respuesta es un documento HTML.
- *Línea 7*: el método *getWriter* devuelve un flujo de tipo *PrintWriter*, que permite al servidor enviar el contenido al cliente.
- *Líneas 8-14:* crean el documento HTML escribiendo sentencias html mediante el método *println*.
- *Línea 15*: cerrar el flujo de salida y enviar la información al cliente.[8]

- Crear el archivo *ServletdeBienvenida.html*. Este debe ser guardado con el nombre *ServletdeBienvenida.html* en la subcarpeta *servlet* de la carpeta *webapps* de *Tomcat*.

[8] Este archivo debe ser compilado y guardado en la subcarpeta *classes*.

```
1    <html>
2      <head>
3          <title> Un ejemplo de un Servlet</title>
4      </head>
5      <body>
6          <form action =»/examples/bienvenido1" method =»get»>
7            <p><label>Haga clic en el botón para invocar el servlet
8            <input type =»submit» value=»visualizar documento de HTML»/>
9             </label></p>
10         </form>
11     </body>
12   </html>
```

- *Líneas 1 y 12*: delimitan el documento html.
- *Líneas 2 y 4*: crean el encabezado del documento.
- *Línea 3*: definen el título del encabezado.
- *Líneas 5 y 11*: crean el cuerpo del documento
- *Línea 6*: El parámetro *action de form* especifica la ruta URL, la cual invoca al *servlet;* por su parte, *method de form* indica que el navegador envía una petición *get* al servidor, el cual a su vez realiza una llamada al método *doGet* del *servlet*.
- *Líneas 7-9*: la etiqueta <*p*> define un párrafo, y <*label*> define un texto. La sentencia <*input type=»submit»* crea un botón y *value* permite asignar un texto al botón creado.

• Crear el descriptor de despliegue *(web.xml)*. Este archivo debe ser guardado con el nombre *web.xml* en la carpeta WEB-INF de la subcarpeta *webapps* de *Tomcat*. Para el resto de ejemplos solo se adiciona la información necesaria para ejecutar un nuevo *servlet*.

```
1    <web-app>
2      <display-name>Ejemplo de Servlets en Java</display-name>
3      <description>
4            El primer Servlet
5       </description>
6       <servlet>
7          <servlet-name>bienvenido1</servlet-name>
8          <servlet-class>
9          ServletBienvenida
10         </servlet-class>
11    </servlet>
12    <servlet-mapping>
13        <servlet-name>bienvenido1</servlet-name>
14        <url-pattern>/bienvenido1</url-pattern>
15     </servlet-mapping>
16   </web-app>
```

- *Líneas 1 - 16*: definen la configuración de cada *servlet* en la aplicación *Web*.
- *Línea 2*: especifica un nombre que puede mostrarse al administrador del servidor, para indicar en dónde está instalada la aplicación *Web*.
- *Líneas 3 - 5*: hacen una descripción de la aplicación.
- *Líneas 6 y 11*: el elemento <*servlet-name*> es el nombre (alias) que se elige para el *servlet*; <*servlet-class*> especifica el nombre del *.class* de Java.
- *Líneas 12 - 15*: El elemento <*url-pattern*> ayuda al servidor a determinar las peticiones que se envían al *servlet*.

• Ejecutar el *servlet.* Previa inicialización de *Tomcat* se debe ingresar al navegador y escribir la dirección URL: http://127.0.0.1:8080/examples/servlets/ServletdeBienvenida.html. Allí se visualizará la pantalla que se muestra en la figura 3.

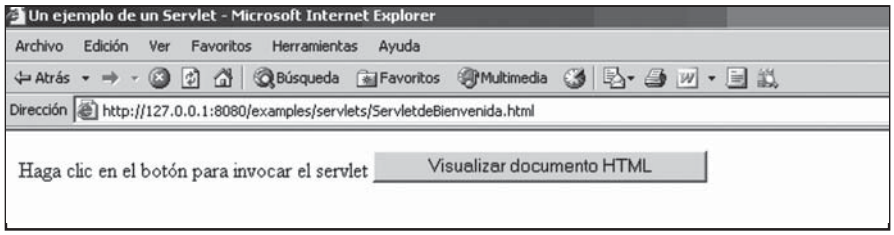

Figura 3. Ejecución inicial del *servlet*.

Al pulsar el botón se visualizará el contenido de la figura 4.

Figura 4. Finalización de la ejecución del *servlet***.**

7.2. Ejercicio 2. *Realizar un* servlet *que permita la utilización de parámetros para enviar a un servidor un nombre de una persona y la marca de carro favorita utilizando el método* get.

• Crear programa fuente *(servlet*).

```
1 import javax.servlet.*;
2 import javax.servlet.http.*;
3 import java.io.*;
4 public class ServletConParametros extends HttpServlet {
5 protected void doGet(HttpServletRequest peticion, HttpServletResponse respuesta)
throws ServletException, IOException {
6 String nombre=peticion.getParameter(«nombre»);
7 String carro=peticion.getParameter(«carro»);
8  respuesta.setContentType(«text/html»);
9 PrintWriter salida=respuesta.getWriter();
```

```
10  salida.println(«<head>»);
11  salida.println(«<title> Procesamiento de peticiones get con datos</title>»);
12  salida.println(«</head>»);
13  salida.println(«<body>»);
14  salida.println(«<h1>! hola :»+nombre+»,<br/>»);
15  salida.println(«!Bienvenido a los servlets!<br/>»);
16  salida.println(«Su marca de carro favorita es :»+carro+»</h1>»);
17  salida.println(«</body>»);
18  salida.println(«</html>»);
19  salida.close(); }}
```

- *Líneas 6 y 7*: crean dos objetos *String* (nombre, carro); utilizando el método *getParameter* se asignan a los objetos los valores enviados por el cliente.
- *Líneas 13-17*: crean el documento HTML; escribiendo sentencias HTML, mediante el método *println* se realiza la impresión de los parámetros enviados por el cliente.

• Crear el archivo *ServletConParametros.html*

```
1 </head>
2 <body>
3       <form action =»/examples/parametros» method =»get»>
4       <p><label>Escriba su nombre, marca de carro favorita y pulse el botón enviar
5       <br/> <input type =»text» name =»nombre»/>
6       <br/> <input type =»text» name =»carro»/>
7       <input type = «submit» value=»Enviar»/>
8       </p></label>
9       </form>
10      </body>
11      </html>
```

- *Líneas 6 y 7*: crean los dos campos de texto para enviar el nombre y la marca de carro favorita al servidor.

• Adicionar al archivo descriptor de despliegue *web.xml*

```
<servlet>
<servlet-name>parámetros </servlet-name>
<servlet-class>
ServletConParámetros
</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>parámetros </servlet-name>
<url-pattern>/parámetros </url-pattern>
</servlet-mapping>
```

- Ejecutar el *servlet.*Ingresar al navegador y escribir la siguiente dirección URL: http://127.0.0.1:8080/examples/servlets/ServletConParametros.html. El resultado será visualizar la pantalla presentada en la figura 5.

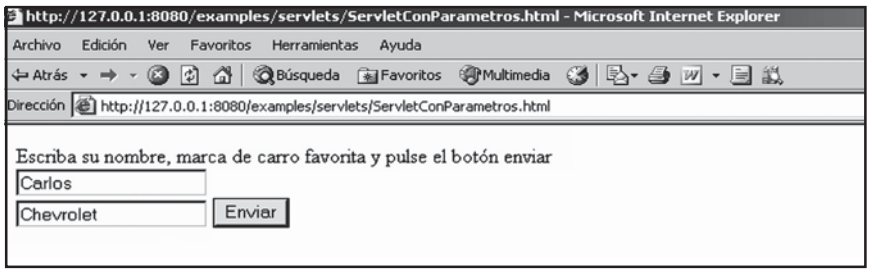

Figura 5. Captura de los datos del cliente.

Al digitar la información solicitada y pulsar el botón «*enviar*» se visualizará la pantalla presentada en la figura 6.

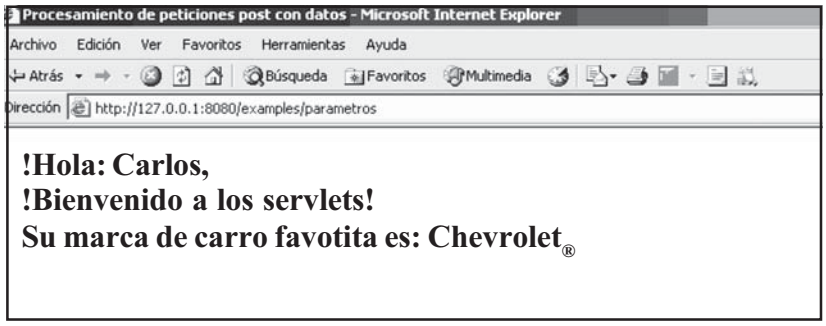

Figura 6. Página con la información recibida por el servidor.

7.3. Ejercicio 3. *Hacer un* servlet *que permita realizar la conexión a una base de datos realizada en* Access *llamada* opiniones*. El servlet deberá visualizar el contenido de la tabla* alumnos.

- Crear una base de datos en Access llamada *opiniones.* En ella**,** crear una tabla llamada *alumnos***,** con la siguiente estructura:

- carnet	Número	
- nombres	Texto	25
- apellidos	Texto	25
- curso	Número	

```
1   <html>
2     <head>
3       <title> Un ejemplo de un Servlet</title>
4     </head>
5     <body>
6       <form action =»/examples/bienvenido1" method =»get»>
7         <p><label>Haga clic en el botón para invocar el servlet
8         <input type =»submit» value=»visualizar documento de HTML»/>
9         </label></p>
10      </form>
11    </body>
12  </html>
```

La tabla con seis registros quedará como se muestra en la figura 9.

alumnos : Tabla

carnet	nombres	apellidos	curso
2000	José	Ramirez	2
4000	Cristian	Vanegas	1
6000	Rosa	Cetina	2
8000	Nancy	Valbuena	1
10000	María	Rodríguez	3
12000	Carlos	Moreno	3
0			0

Figura 9. Registros de la tabla alumnos.

- *Crear el puente ODBC:JDBC.* Se debe abrir el panel de control y escoger el icono *Fuente de Datos ODBC*, que mostrará la ventana *Administrador de orígenes de datos ODBC*.

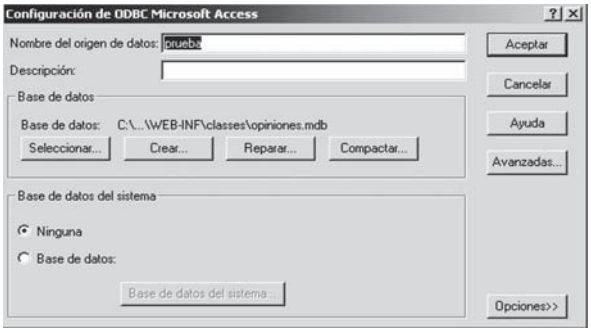

Figura 10. Administrador de orígenes de datos ODBC.

En la página DSN de usuario se pulsa el botón *Agregar* y se selecciona el controlador que se necesita, en este caso *Controlador para Microsoft Access (*.mdb)*. A continuación se observará la ventana de configuración de ODBC Microsoft Access de la figura 11.

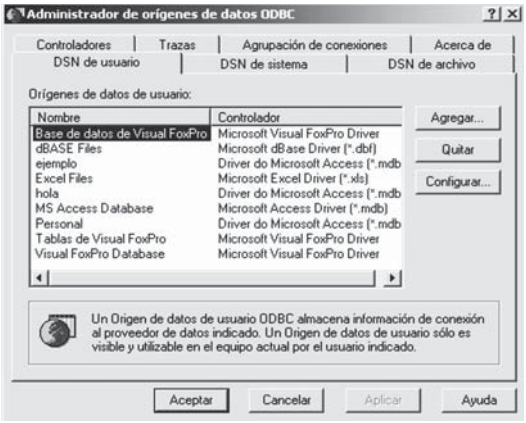

Figura 11. Configuración de ODBC Microsoft Access.

En este caso, en la opción origen de datos se escribe *prueba*. En el botón *Seleccionar* se escoge la base de datos *opiniones* y se pulsa el botón *Aceptar*.

- Crear el programa fuente (*servlet*)

```
1  import javax.servlet.*;
2  import javax.servlet.http.*;
3  import java.io.*;
4  import java.sql.*;
5  import java.util.*;
6  public class PruebaAlumnos extends HttpServlet {
7   Connection conexion=null;
8   PrintWriter salida=null;
9   public void init(ServletConfig configuracion)throws ServletException {
10      super.init(configuracion);
11      String base=new String(«jdbc:odbc:prueba»);
12       try{
13          Class.forName(«sun.jdbc.odbc.JdbcOdbcDriver»);
14      }catch(ClassNotFoundException e){}
15      try{
16          conexion=DriverManager.getConnection(base,»»,»»);
17      }catch(SQLException sqle){}}
```

```
18  public void destroy() {
19  super.destroy();
20        try{
21        conexión.close();
22   }catch(SQLException sqlex){} }
23  public void doGet(HttpServletRequest req, HttpServletResponse resp)
                          throws ServletException, IOException     {
24        PrintWriter pagina;
25        resp.setContentType(«text/html»);
26        página = resp.getWriter();
27  Statement sentencia=null;
28  ResultSet tabla;
29  try {
30  sentencia=conexion.createStatement();
31  tabla=sentencia.executeQuery(«select * from alumnos»);
32              página.println(«<TABLE Border=10 CellPadding=5><TR>»);
33              página.println(«<th bgcolor=Green>CARNET</th><th bgcolor=White>NOMBRE</
th><th bgcolor=Red>APELLIDOS</th><th bgcolor=blue>CURSO</th></TR>»);
34        while(tabla.next()) {
35              página.println(«<TR>»);
36              página.println(«<TD>»+tabla.getInt(1)+»</TD>»);
37              página.println(«<TD>»+tabla.getString(2)+»</TD>»);
38              página.println(«<TD>»+tabla.getString(3)+»</TD>»);
39              página.println(«<TD>»+tabla.getInt(4)+»</TD>»);
40              página.println(«</TR>»); }; // fin while
41        página.println(«</TABLE></CENTER></DIV></HTML>»);
42        tabla.close();
43        página.close();
44   }catch(SQLException sqlexe){}      }}
```

- *Línea 4*: importa el paquete *sql*, necesario para la manipulación de la base de datos.
- *Línea 7*: crea un objeto conexión tipo *Connection*, que servirá para realizar la conexión a la base de datos.
- *Línea 10:* mediante el método *INIT*, envía la configuración inicial a la clase en que se realizó la herencia.
- *Línea 11*: crea el objeto *base,* tipo *String*, que es el nombre de los datos de origen del administrador ODBC.
- *Líneas 12-14 y 15-17*: realizan el *try-catch*, que permitirá el manejo de excepciones en las operaciones de entrada/salida de los datos.
- *Líneas 18-22*: crean el método *destroy*, que permitirá la liberación de los recursos al finalizar el *servlet*.
- *Línea 27*: crea un objeto *sentencia* tipo *Statement,* que sirve para consultar la base de datos.
- *Línea 31*: crea un objeto *tabla* tipo *ResulSet*, en donde se ejecuta el método *executeQuery* de la interfaz *Statement*, para crear una consulta de todos los datos que contenga la tabla *alumnos*.
- *Líneas 32-33*: crean los títulos de los campos por consultar.

- *Líneas 34-40*: definen un ciclo para recorrer la tabla *alumnos*, con el fin de mostrar todos sus registros.
- *Líneas 42-43*: utilizan el método close() para cerrar todos los objetos inicializados previamente.

• Crear el archivo *Prueba.html*

```
<head>
    <title> Un ejemplo de una consulta de una base de datos</title>
</head>
<body>
    <form action =»/examples/pruebita» method =»get»>
    <p><label>Haga clic en el botón para invocar el servlet
 <input type =»submit» value=»Consultar Base de Datos»/>
    </label></p>
    </form>
</body>
</html>
```

• Adicionar al archivo descriptor de despliegue (*web.xml*)

```
<servlet>
      <servlet-name>pruebita</servlet-name>
   <servlet-class>
       Pruebaalumnos
    </servlet-class>
</servlet>
<servlet-mapping>
     <servlet-name>pruebita </servlet-name>
     <url-pattern>/pruebita</url-pattern>
</servlet-mapping>
```

• Ejecutar el *servlet***.** Ingresar al navegador y escribir la dirección URL:

http://127.0.0.1:8080/examples/servlets/Prueba.html. El resultado se muestra en la figura 12:

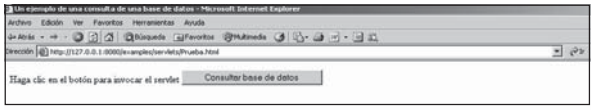

Figura 12. Página inicial para la consulta de los registros de la tabla alumnos**.**

Al pulsar el botón *Consultar base de datos* se visualizará la figura 13:

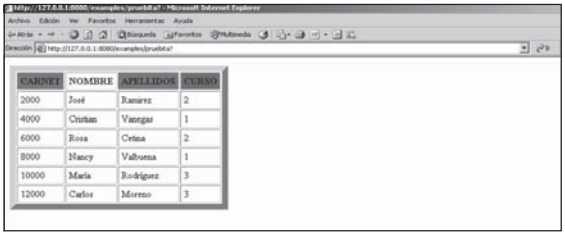

Figura 13. Página con registros de la tabla alumnos

8. Conclusiones

• *Los servlets* sirven para desarrollar soluciones basadas en *Web* proporcionando el acceso seguro al sitio respectivo. Ellos son independientes del servidor utilizado y de su sistema operativo.

- Los *servlets* de Java permiten la creación de páginas *Web* dinámicas. Pueden actuar como enlace entre el cliente y una o varias bases de datos en arquitecturas cliente-servidor.
- Los *servlets* son módulos que extienden los servidores orientados a petición-respuesta, como los servidores *Web* compatibles con Java. A manera de ejemplo, uno podría ser responsable de tomar los datos de un formulario de entrada de pedidos en HTML y aplicarle la lógica de negocios utilizada para actualizar la base de datos de pedidos de una compañía.
- Los servlets son más eficientes, fáciles de usar y más portables que un CGI.

Referencias bibliográficas

[1] Hanna P. (2002) *Manual de Referencia JSP,* 1ª ed., McGraw Hill.

[2] Bobadilla J. y Sancho A. (2003) *Comunicaciones y bases de datos con Java a través de ejemplos*. 1.ª ed. Ra-Ma.

[3] Deitel H. y Deitel P. (2004) *Cómo programa Java*. 5.ª ed., Prentice Hall.

[4] Deitel H.M. (1998). *Cómo programar en Java*, 2.ª ed. Ed. Prentice Hall.

[5] Wang P. (2000). *Java con programación orientada a objetos y aplicaciones en la www*. 1.ª ed. International Thomson Ed. S.A.

Infografía

[6] The on-line Java 2 SDK Documentation.2001. En: http://java.sun.com/j2se/1.3/docs/index.html

[7] http://www.programacion.com/tutorial/servlets_basico/3/

[8] http://www.programacion.com/java/tutorial/servic_web/1/

[9] http://www.desarrolloweb.com/articulos/534.php?manual=21

[10] http://www.desarrolloweb.com/articulos/535.php?manual=21

[11] http://www.desarrolloweb.com/articulos/537.php?manual=21

[12] http://www.tejedoresdelweb.com/307/article-10152.html

[13] http://es.wikipedia.org/wiki/Tomcat

[14] http://www.programacion.com/java/tutorial/tomcatintro/1/