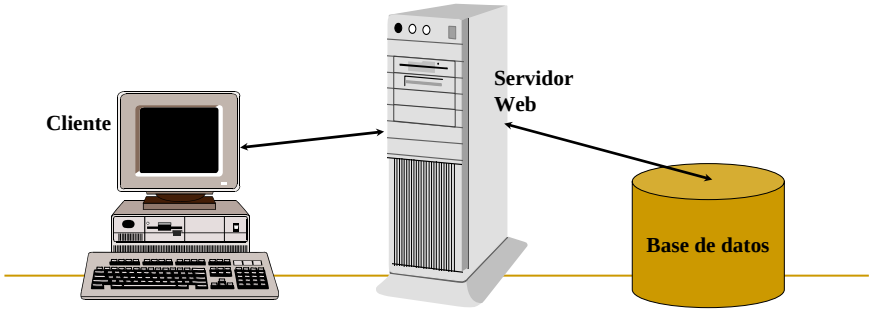
Laboratorio de Informática Distribuida

Sesión II. Servlets

Servlets

- Tecnología Servlets
  - Se puede definir un **Servlet** como un **programa JAVA** que se ejecuta en un entorno distribuido en red, como un **servidor web**, y que recibe y responde a las peticiones de un cliente a través del protocolo **HTTP**.



Servlets

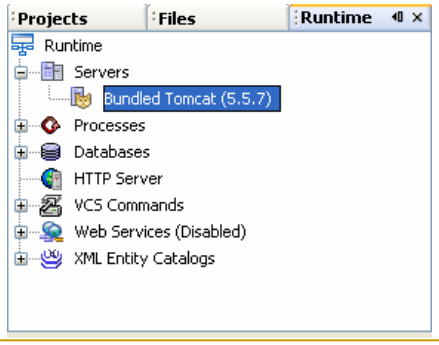
- Entorno de Trabajo:
  - Sistema operativo.
  - Servidor Web: Apache Tomcat.
  - Configuración de las variables de entorno.
  - Cómo crear una aplicación Web y desplegarla.
  - Formas de ejecutar un servlet.

Servlets

- **Sistema operativo.**
  - A lo largo de la sesión se asumirá que se trabaja sobre una máquina con Windows 95/98/2000/NT/XP.
  - En la elección del sistema operativo se ha tenido en cuenta que la mayoría de las personas que leen este tema dispondrán y conocerán este popular sistema operativo.
  - Pero también es factible construir un entorno de programación de la tecnología de los Servlets en un sistema operativo con Unix.

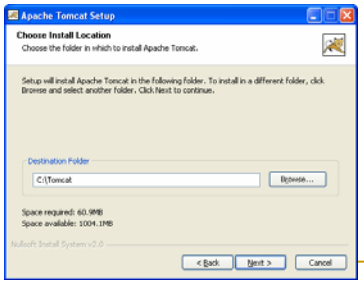
Servlets

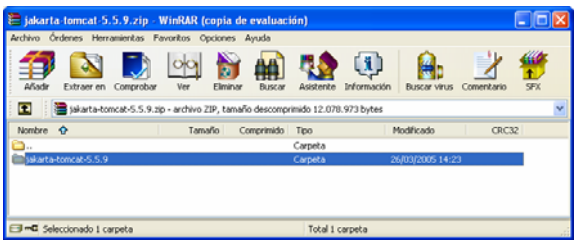
- **Servidor Web (I).**
  - El Servidor Web elegido es **Tomcat**. Para su utilización dispones de **dos posibilidades**:
    - 1. Dentro del **NetBeans** tenemos pre-instalado el Tomcat 5.5.7.



Servlets

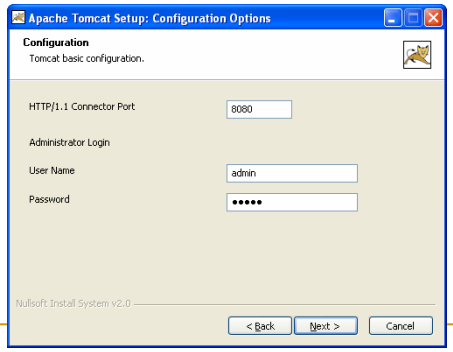
- **Servidor Web (II).**
  - El Servidor Web Tomcat.
    - 2. Instalar el Tomcat de forma **independiente**, simplemente será necesario descomprimir o ejecutar el fichero de jakarta-tomcat-5.5.9 en C:





Servlets

- **Servidor Web (III).**
  - Cuando pregunte por el nombre y la clave del administrador se recomienda utilizar **admin – admin**. El puerto por defecto es el 8080.



Servlets

- **Servidor Web (IV)**
  - Para arrancar el servidor vamos al icono del Tomcat y pulsamos (botón derecho del ratón) en Start service.
  - Una vez arrancado el servidor podemos comprobar su funcionamiento ejecutando el navegador de Internet y poniendo la dirección: http://localhost:8080 o http://127.0.0.1:8080

Servlets

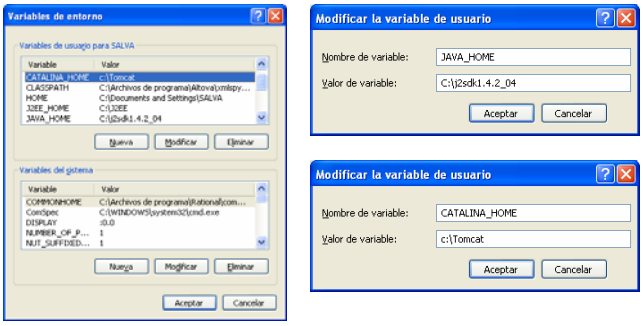
- **Servidor Web (V).**
  - La página por defecto del Servidor Web.

Servlets

- **Configuración de las variables de entorno.**
  - Se tendrán que añadir dos nuevas variables de entorno al sistema operativo. Estas son **JAVA_HOME** que especifica donde se encuentra instalado el JDK y otra es **CATALINA_HOME** que indica donde está instalado el Tomcat.
  - A continuación mostramos las pantallas en la que se especifica esta información (Panel de Control – Sistema – Opciones Avanzadas – Variables de Entorno):

Servlets

- **Configuración de las variables de entorno.**



Servlets

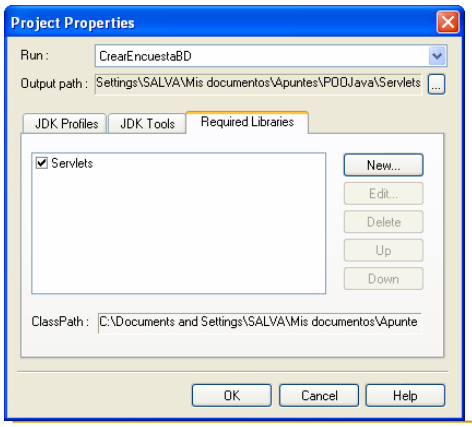
- **Configuración de las variables de entorno.**
  - Si el entorno de desarrollo **NO** es NetBeans, para poder compilar las clases que representan los **servlets** debemos decirle al compilador donde se encuentra el **API de los servlets**.
  - Esto es así porque el API de los servlets no viene con el API estándar del JDK.
  - La librería que tenemos que añadir se llama ***servlet-api.jar*** y se encuentra dentro del Tomcat en el directorio ***/common/lib*.**
  - Si se utiliza NetBeans esto no es necesario!!!

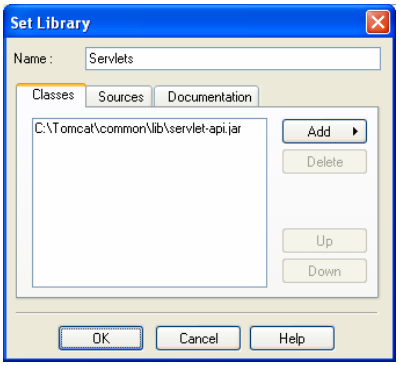
Servlets

- **Configuración de las variables de entorno.**
  - Si utilizamos el entorno de Java JCreator, podremos especificar donde buscar las librerías externas al JDK para incluirlas en nuestro proyecto y poder buscar las clases necesarias para compilar los **servlets.**
  - Para hacerlo debemos ir a las propiedades del proyecto y en la pestaña de ***Required Libraries*** introducir una nueva librería llamada **servlets** que apunte al fichero ***servlet-api.jar*.**

Servlets

- **Configuración de las variables de entorno.**



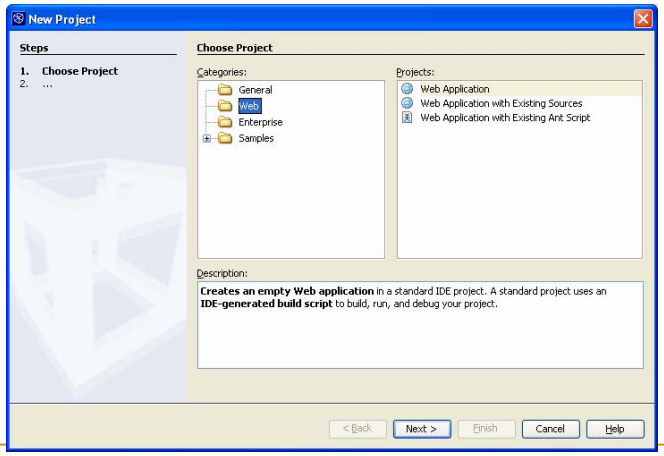


Servlets

- Cómo crear una aplicación Web en NetBeans.
  - Pasos:
    1. Elegir un nuevo proyecto Web.
    2. Configurarlo: Nombre del proyecto, su ubicación, el servidor web, etc.
    3. Cerrar configuración configuración

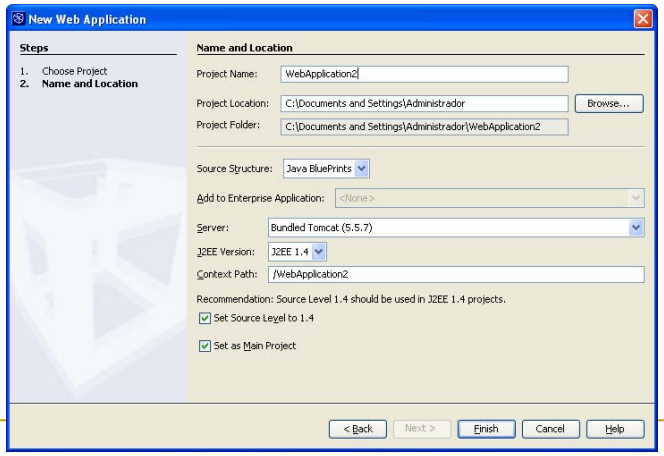
Servlets

- Elegir un nuevo proyecto en NetBeans



Servlets

- Configurarlo en NetBeans.

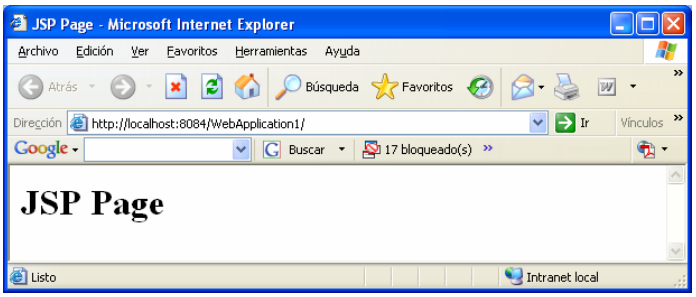


Servlets

- Una vez terminada la configuración, el proyecto web ya está creado y ya podemos ejecutarlo en NetBeans.
  - Para ejecutarlo debemos pulsar el botón de Run Main Proyect - > F6.

Servlets

- Y podremos visualizar, a través de un navegador web, la página por defecto de nuestro proyecto no es otra cosa que una página jsp



Servlets

- **Cómo crear una aplicación y desplegarla, sin NetBeans.**
  - Para crear una aplicación Web y desplegarla en el servidor es necesario **crear una estructura de directorios** en la carpeta **webapps** del Tomcat.
  - 1.- Creamos una carpeta con el nombre del proyecto Web que queremos realizar, por ejemplo:

    C:\Tomcat\webapps\Miproyecto
  - 2.- En el directorio MiProyecto situamos los siguientes ficheros y directorios:

Servlets

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

Servlets

- **Cómo crear una aplicación y desplegarla.**
  - Una vez situados todos los ficheros en los directorios adecuados para crear nuestra aplicación Web, debemos crear un fichero llamado **web.xml**.
  - Este fichero se conoce como el **descriptor de despliegue** y sirve para registrar nuestros servlets en el **contenedor**.
  - El entorno de NetBeans nos ayuda enormemente en la creación y posterior manipulación de este fichero.
  - Para crear fácilmente un descriptor de despliegue, se suele editar uno existente. A continuación tenemos un esqueleto de un fichero **web.xml**:

Servlets

- **Fichero Web.xml.**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
             http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
            version="2.4">
   <session-config>
      <session-timeout>
         30
      </session-timeout>
   </session-config>
   <welcome-file-list>
         <welcome-file>
        index.jsp
      </welcome-file>
   </welcome-file-list>
</web-app>
```

Servlets

- **Cómo crear una aplicación y desplegarla.**
  - Además de la información anterior debemos describir cuándo el **contenedor de servlets** debe invocar al servlet, lo que se suele llamar el **mapeo** del servlet.
  - En otras palabras, debemos describir cómo se enlaza una **URL al servlet**. En el fichero web.xml, las URLs se mapean de esta forma:

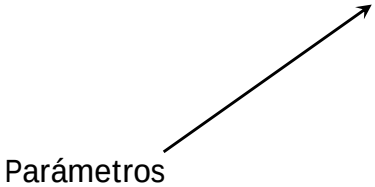
```
<servlet-mapping>
<servlet-name>nombre</servlet-name>
<url-pattern>pattern</url-pattern>
</servlet-mapping>
```

Servlets

- **Formas de ejecutar un Servlet.**
  - Introducir la **dirección URL del servlet** en un navegador web.
  - **Llamar** al servlet desde una **página web o formulario**.
  - Ejecutar un servlet **llamándolo desde otro servlet.**

Servlets

- Desde un navegador web
  - Ejemplo:
    http://nombre_maquina:puerto/ruta_servlet/nombre_servlet
  - Las llamadas a servlets pueden contener parámetros:
    http://www.cc.alcala.es/~barchino/servlets/ejem1?dato1=3&dato2=4



Servlets

- Desde una página Web
  - Un *servlet* también puede ser llamado desde el código de una página web, al igual que se llamaría a cualquier CGI.
  - Ejemplo:

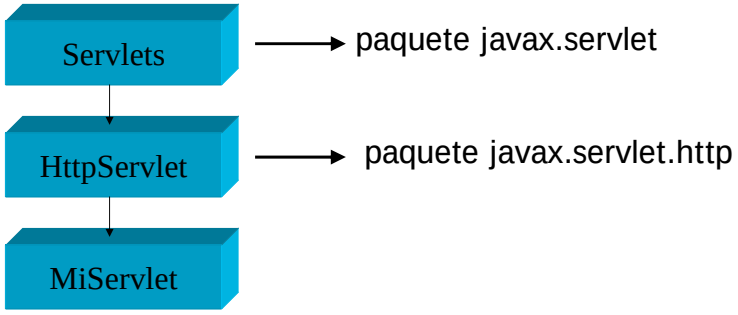
```
<form action =
   "http://www.cc.alcala.es/~barchino/servlets/serv2"
   method="post">
   ...distintas entradas en el formulario…
</form>
```

Servlets

- Desde otro servlet
  - Se puede lanzar la ejecución de un servlet desde otro.
  - Pasos:
    - Conocer el nombre del servlet que queremos llamar.
    - Proporcionar acceso al objeto "Servlet" del servlet llamado.
    - Llamar al método público del servlet.

Servlets

- El API de los Servlets
  - Biblioteca 'javax.servlet'



Servlets

- El API de los Servlets
  - Dos paquetes:
    - javax.servlet
      - Interfaz Servlet
      - Interfaz ServlerRequest
      - Interfaz ServlerResponse
      - Interfaz ServletConfig
      - Interfaz ServletContext
      - Interfaz SingleThreadModel
      - Clase GenericServlet
    - javax.servlet.http
      - Interfaz HttpServlet
      - Interfaz HttpServletRequest
      - Interfaz HttpServletResponse
      - Interfaz HttpServletConfig
      - ……

Servlets

- **La Clase HttpServlet**
  - Extiende de la clase GenericServlet y proporciona una implementación de la interfaz Servlet mucho más específica para el **protocolo HTTP**. Esta es la clase que extienden la mayoría de los servlets que hay en la actualidad.
  - **Métodos** (throws ServletException, IOException)
    - Gestionan el servicio
      - **public** void service (HttpServletRequest req, HttpServletResponse res)
    - Implementan operaciones propias de HTTP
      - GET doGet (HttpServletRequest req, HttpServletResponse res)
      - POST doPost (HttpServletRequest req, HttpServletResponse res)

Servlets

- **Javax.servlet.http.HttpServletRequest**
  - Una interfaz que encapsula la funcionalidad de las peticiones que hace el cliente.
  - **IMPORTANTE:** Uno de los métodos de esta interfaz es el método **getParameter (String Name)** utilizado en la mayoría de los servlets que recuperan valores de los de los formularios HTML rellenados por los clientes de las aplicaciones.
  - Un objeto ***HttpServletRequest*** se pasa como parámetro al método ***service*** de la clase ***HttpServlet***.

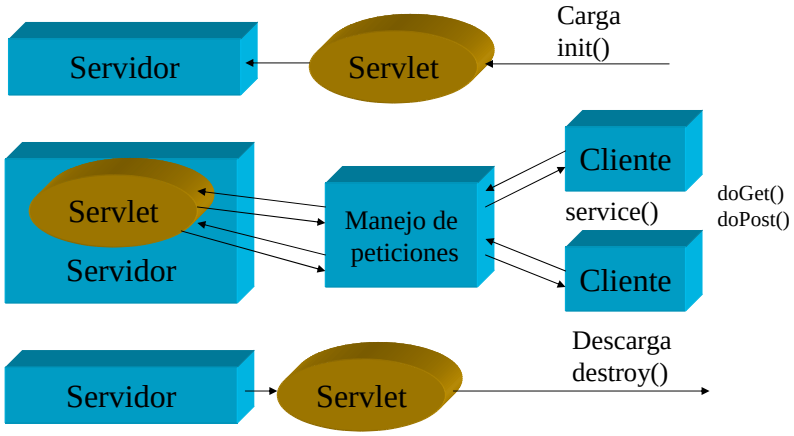
Servlets

- **Javax.servlet.http.HttpServletResponse**
  - Esta interfaz encapsula la funcionalidad de una respuesta HTTP, incluyendo el manejo de las cabeceras del propio protocolo.
  - Interfaz que encapsula la funcionalidad de la respuesta **MIME** que será enviada al cliente.
  - Un objeto ***ServletResponse*** se pasa como parámetro al método ***service*** de la clase ***Servlet*.**

Servlets

- Ciclo de Vida:
  - La interfaz que se utiliza para implementarlos define una serie de métodos para cada una de las etapas.
  - El orden es el siguiente:
    - Cuando el servidor carga el Servlet invoca al método **init**. (Todas las peticiones que lleguen ante de que el método termine deberán esperar).
    - Una vez ejecutado el método **init**, todas las peticiones serán atendidas por el método **service** . IMPORTANTE: El servidor manejará varias peticiones simultáneamente, asignando un hilo propio a cada petición.
    - Por último cuando el servidor web descarga al Servlet se invocará al método **destroy.**

Servlets

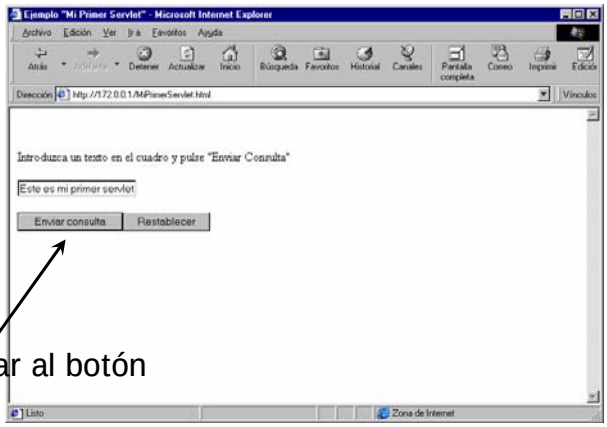


Servlets

- Creación del primer Servlet:
  - Se van a seguir todos los pasos necesarios para hacer funcionar sin problemas un **Servlet**, y se va a realizar un primer análisis de su estructura básica.
  - El ejemplo va a consistir en una página HTML en la que existirá un campo de entrada en el formulario, con los correspondientes botones **Submit** (Enviar consulta) y **Reset** (Restablecer) típicos de cualquier formulario de entrada de datos.

Servlets

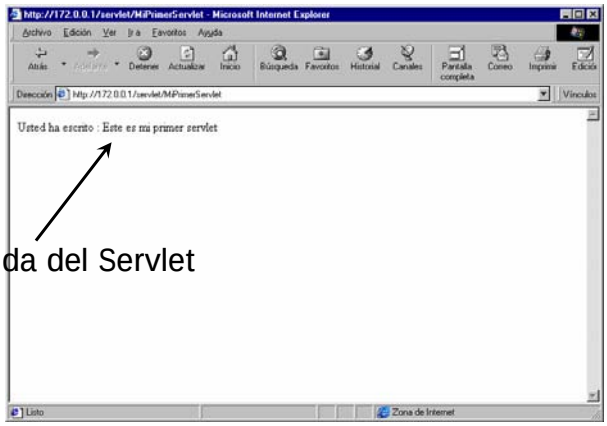
- Creación del primer Servlet. Pagina Web.



Al pulsar al botón

Servlets

- Creación del primer Servlet. Resultado.



Salida del Servlet

Servlets

- Código HTML:

```
<html>
 <head>
  <title>Ejemplo "Mi Primer Servlet"</title>
</head>
<body>
 <form action=/MiPrimerServlet/MiPrimerServlet method=POST>
<BR> <BR>Introduzca un texto en el cuadro y pulse "Enviar
Consulta"<BR>     <BR>
 <input type=text name=TEXTO>
  <BR>   <BR><input type=submit><input type=reset></form>
      </body>
</html>
```

Servlets

- Código del Servlet:

```
//    MiPrimerServlet.java
//
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class MiPrimerServlet extends HttpServlet
{
   string nombre;
    public void service(HttpServletRequest req, HttpServletResponse res)
              throws ServletException, IOException
                {
                 nombre = req.getParameter("TEXTO");
                 PrintStream out = new PrintStream(res.getOutputStream());
                  res.setContentType("text/html");
                  out.println("<p>Usted ha escrito : "+nombre+"</p>");
                 out.close();
                }
}
```

Servlets

- Código del web.xml

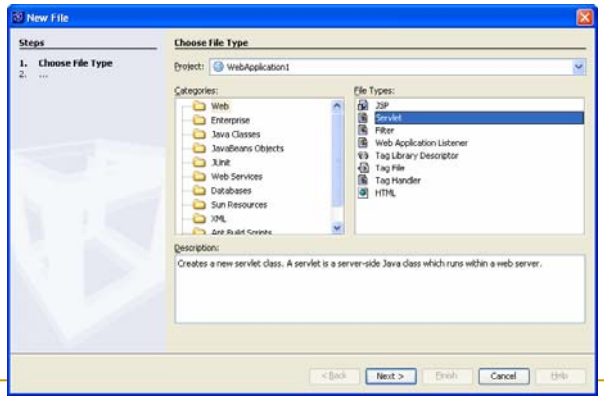
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <display-name>Servidor Hola</display-name>
    <description>Captura un texto como parametro</description>
  <servlet>
     <servlet-name> MiPrimerServlet </servlet-name>
     <servlet-class> MiPrimerServlet </servlet-class>
    <servlet-class>package.nombre.MiClass</servlet-class>
  </servlet>
<servlet-mapping>
    <servlet-name> MiPrimerServlet </servlet-name>
    <url-pattern>/MiPrimerServlet </url-pattern>
  </servlet-mapping>
</web-app>
```

Servlets

- Crear un Servlet, por defecto, en el entorno de NetBeans.
  - Pasos:
    - Crear un nuevo fichero de tipo servlet sobre el proyecto web previamente creado.
    - Configurar servlet con el nombre y demás parámetros de configuración como veremos a continuación.
    - Ejecutar la aplicación y probar el ejecución del servlet.

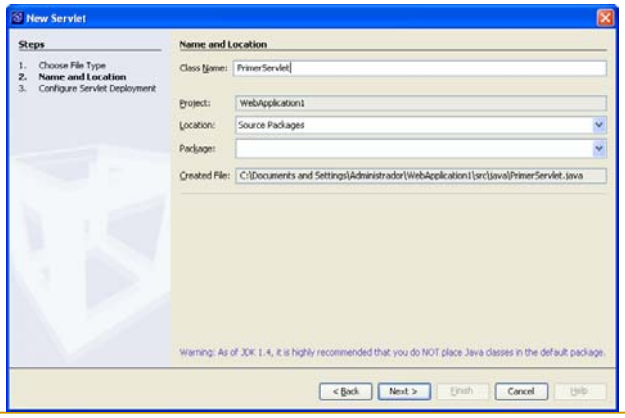
Servlets

- Creación de una servlet en el entorno del NetBeans, una vez creado el proyecto:



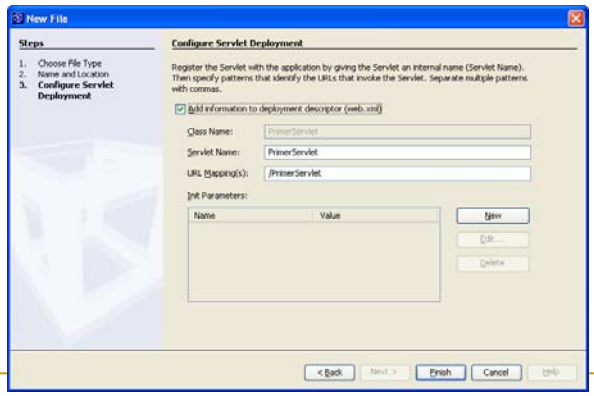
Servlets

- Configurar (I) el servlet con el nombre :



Servlets

- Configurar (II) el servlet con la información necesaria para el mapeo :



Servlets

- Además podemos visualizar el código que se genera por defecto.

```
import javax.servlet.*;
import javax.servlet.http.*;

/**...*/
public class PrimerServlet extends HttpServlet {

    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        // TODO output your page here
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet PrimerServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet PrimerServlet at " + request.getContextPath () + "</h1>");
        out.println("</body>");
        out.println("</html>");
        // */
        out.close();
    }
```

Servlets

- Y el fichero de configuración web.xml.

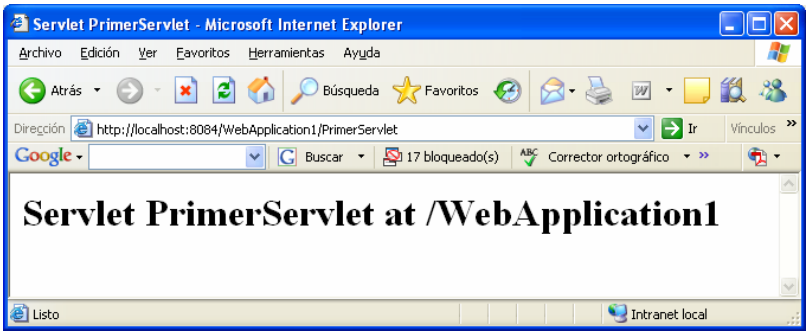
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>PrimerServlet</servlet-name>
    <servlet-class>PrimerServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>PrimerServlet</servlet-name>
    <url-pattern>/PrimerServlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
            30
        </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>
            index.jsp
        </welcome-file>
  </welcome-file-list>
</web-app>
```

Servlets

- Una vez terminada la configuración, del servlet, podemos ejecutarlo.
  - Para ejecutarlo debemos pulsar el botón de Run Main Proyect - > F6.

Servlets

- Resultado de la ejecución.



Servlets

- Servlet y JDBC.
  - La clave del éxito de las aplicaciones Web, reside en la consulta de bases de datos por parte del aplicativo software, que hace de intermediario entre el cliente y el sistema que almacena la información.
  - A las aplicaciones cliente-servidor que utilizan este tipo de arquitectura, se las denomina aplicaciones de **tres** capas.

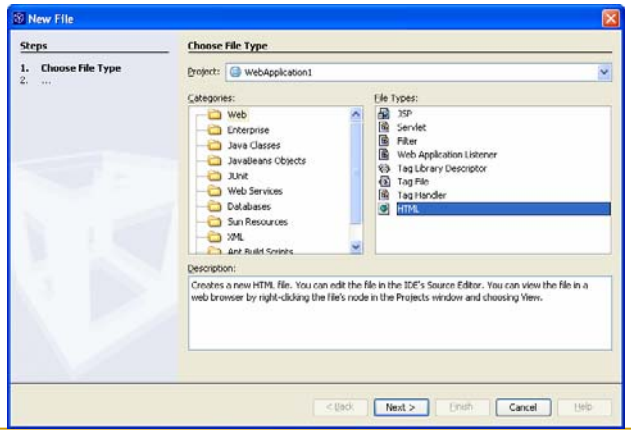
Servlets

- Creación del primer Servlet:
  - Se van a seguir todos los pasos necesarios para hacer funcionar sin problemas un **Servlet**, y se va a realizar un primer análisis de su estructura básica.
  - El ejemplo va a consistir en una página HTML en la que existirá un campo de entrada en el formulario, con los correspondientes botones **Submit** (Enviar consulta) y **Reset** (Restablecer) típicos de cualquier formulario de entrada de datos.

Servlets

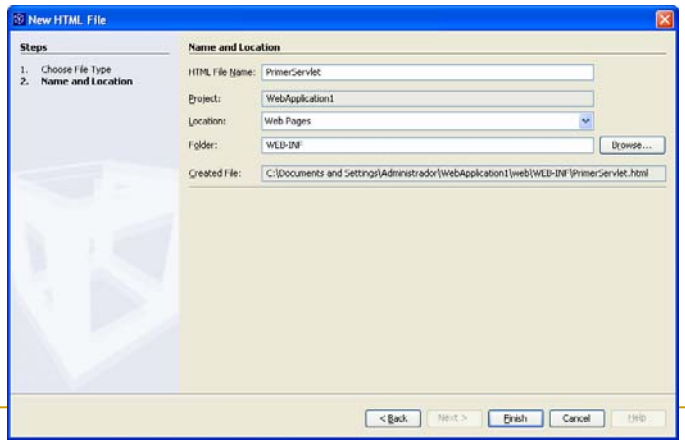
- Creación de una página en el entorno del NetBeans (I):

File –> New File
Ctrl + N



Servlets

- Creación de una página en el entorno del NetBeans (II):



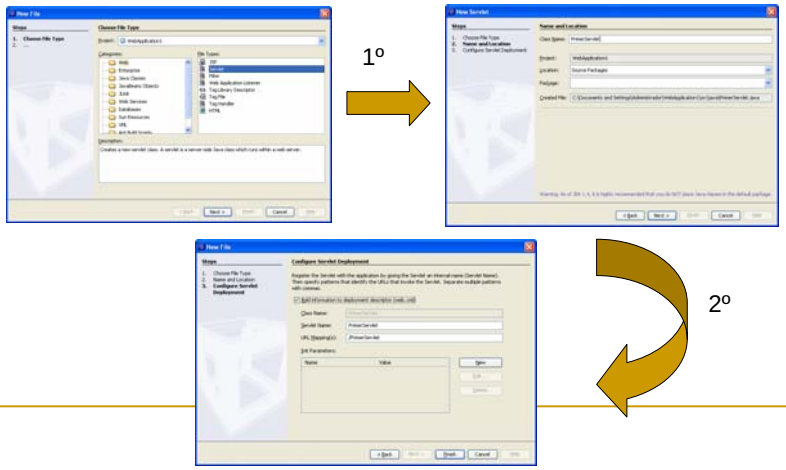
Servlets

- Código HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Ejemplo "Mi Primer Servlet"</title>
</head>
<body>
 <form action=/WebApplication1/PrimerServlet method=POST>
<BR> <BR>Introduzca un texto en el cuadro y pulse "Enviar Consulta"<BR>     <BR>
 <input type=text name=TEXTO>
  <BR>   <BR><input type=submit><input type=reset></form>
 </body>
</html>
```

Servlets

- Crear el servlet:



Servlets

- Código Servlet:

```
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

/**...*/
public class PrimerServlet extends HttpServlet {
    String nombre;


    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        // TODO output your page here
       nombre = request.getParameter("TEXTO");
       response.setContentType("text/html");
       out.println("<p>Usted ha escrito : "+nombre+"</p>");
        // */
        out.close();
    }
```

Servlets

- Código del web.xml

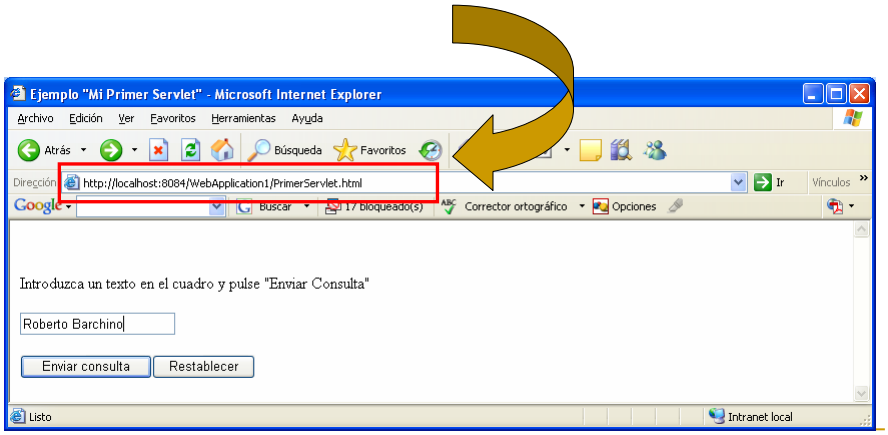
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>PrimerServlet</servlet-name>
    <servlet-class>PrimerServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>PrimerServlet</servlet-name>
    <url-pattern>/PrimerServlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
            30
        </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>
            index.jsp
        </welcome-file>
  </welcome-file-list>
</web-app>
```

Servlets.

- Una vez terminada la creación y configuración, del servlet y la página web, podemos ejecutar esta primera aplicación.
  - Para ejecutarlo debemos pulsar el botón de Run Main Proyect - > F6.

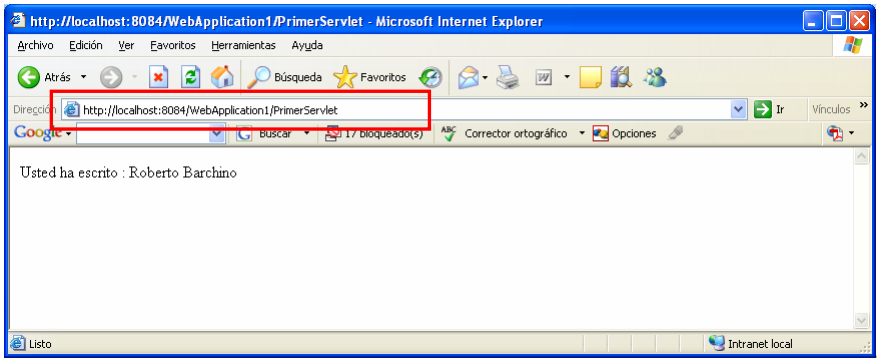
Servlets

- Acceso a la aplicación



Servlets

- Resultado de la Aplicación.

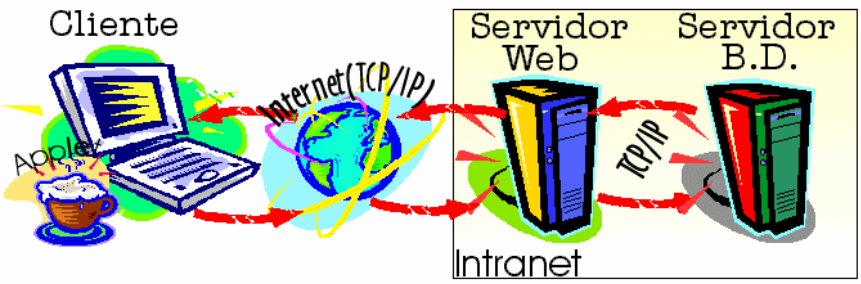


Servlets

- Servlet y JDBC.
  - La clave del éxito de las aplicaciones Web, reside en la consulta de bases de datos por parte del aplicativo software, que hace de intermediario entre el cliente y el sistema que almacena la información.
  - A las aplicaciones cliente-servidor que utilizan este tipo de arquitectura, se las denomina aplicaciones de **tres** capas.

Servlets

- Servlet y JDBC: Arquitectura en tres capas.

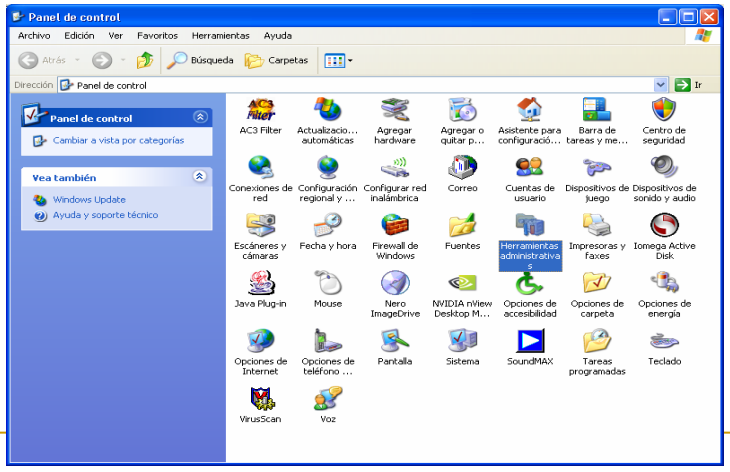


Servlets

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

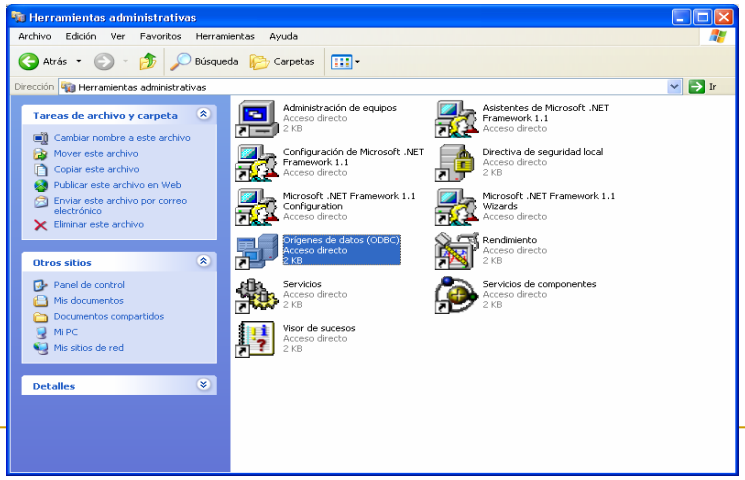
Servlets

- Acceso a Datos. Conector JDBC-ODBC.



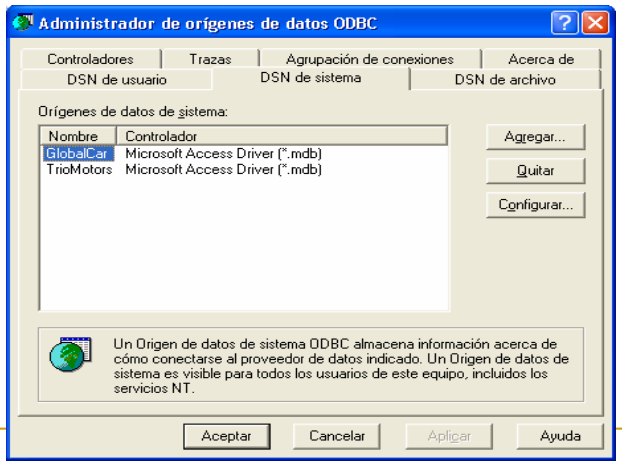
Servlets

- Acceso a Datos. Conector JDBC-ODBC.



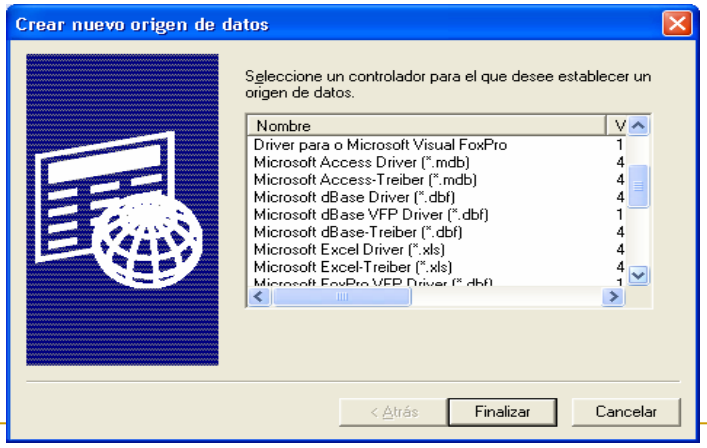
Servlets

- Acceso a Datos. Conector JDBC-ODBC.



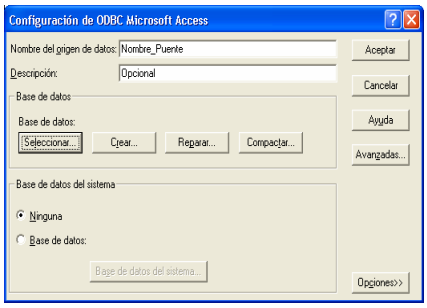
Servlets

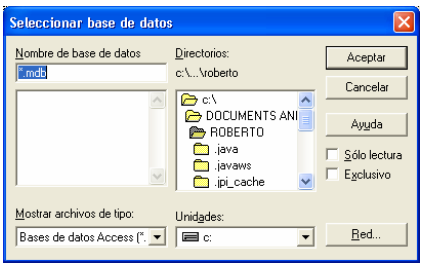
- Acceso a Datos. Conector JDBC-ODBC.



Servlets

- Acceso a Datos. Conector JDBC-ODBC.





Servlets

- Servlet y JDBC. Aplicación Ejemplo. **Encuesta**.
  - En nuestra página principal insertaremos un pequeño formulario, donde se deberá responder a una sencilla pregunta, e introducir los datos personales necesarios para que podamos contactar con el posible ganador del premio.
  - Una vez el participante haya introducido los datos, y pulse el botón para mandar los datos al servidor, recibirá un resumen con la estadística resultante hasta ese momento junto con un mensaje de agradecimiento.

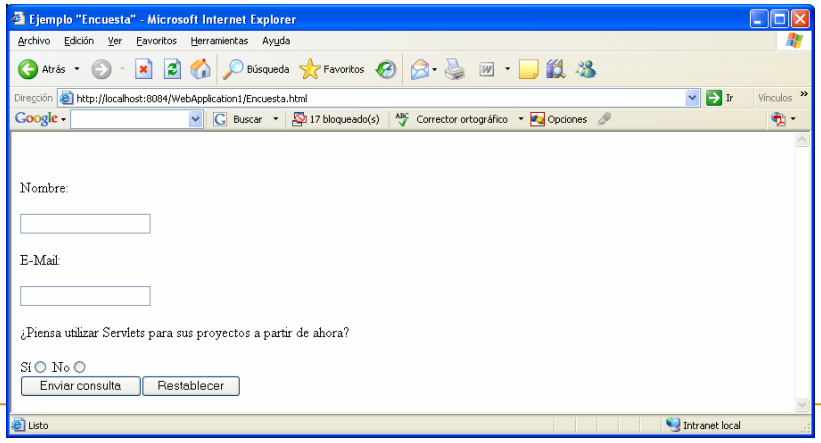
Servlets

- Aplicación Ejemplo. Encuesta. Página HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Ejemplo "Encuesta"</title>
  </head>
  <body>
    <form action=/EncuestaServlet/EncuestaServlet method=POST>
    <BR>
    <BR>Nombre:<BR>
    <BR>
    <input type=text name=NOMBRE>
    <BR>
    <BR>E-Mail:<BR>
    <BR>
    <input type=text name=EMAIL>
    <BR>
    <BR>¿Piensa utilizar Servlets para sus proyectos a partir de ahora?<BR>
    <BR>
        Sí<input type=radio name=RESPUESTA value=SI>
        No<input type=radio name=RESPUESTA value=NO>
<BR><input type=submit><input type=reset></form>
  </body>
</html>
```

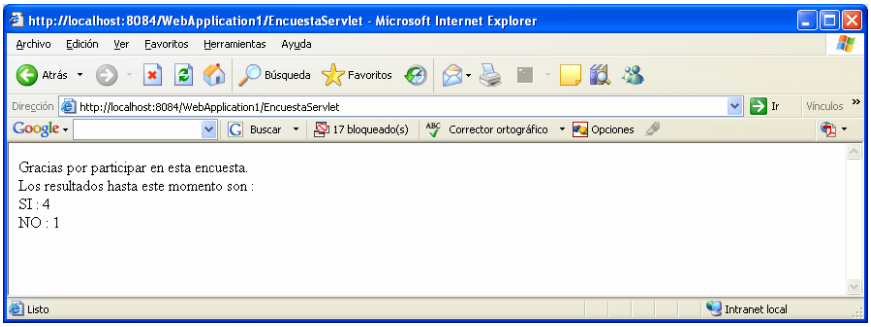
Servlets

- Aplicación Ejemplo. Encuesta. Página HTML



Servlets

- Aplicación Ejemplo. Encuesta. Resultado



Servlets

- Aplicación Ejemplo. Encuesta. Servlet (I)

```
/*
 * EncuestaServlet.java
 *
 * Created on 12 de noviembre de 2005, 11:35
 */

import java.io.*;
import java.net.*;
import java.sql.*;

import javax.servlet.*;
import javax.servlet.http.*;

/**...*/
public class EncuestaServlet extends HttpServlet {

    /**...*/

  Statement  mandato  = null;
  Connection conexion = null;
```

Servlets

- Aplicación Ejemplo. Encuesta. Servlet (II)

```
public void init(ServletConfig config) throws ServletException
{
  super.init(config);
  String URL        = "jdbc:odbc:Access0DBC";
  String usuario    = "";
  String contraseña = "";
  try
    {
      Class.forName("sun.jdbc.odbc.Jdbc0dbcDriver");
    }  catch (Exception e)
       {
           System.out.println("Error al cargar el driver JDBC/ODBC.");
           return;
       }
   try
     {
      conexion = DriverManager.getConnection ( URL, usuario, contraseña);
      mandato  = conexion.createStatement();
     }   catch (Exception e)
         {
            System.out.println("Problemas al conectar con "+URL);
            return;
         }
}
```

Servlets

- Aplicación Ejemplo. Encuesta. Servlet (III)

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
 /* creación del flujo de salida hacia el cliente */

/* recuperamos los valores que nos manda el cliente */
String strNombre    = request.getParameter("NOMBRE");
String strEmail     = request.getParameter("EMAIL");
String strRespuesta = request.getParameter("RESPUESTA");

/* insertamos los datos en la base de datos */
try
{
  mandato.executeUpdate("INSERT INTO ENCUESTA VALUES( '" + strNombre+ "', '" + strEmail+ "', '"
          + strRespuesta+ "');");
}
catch(java.sql.SQLException e)
{
  System.out.println(e);
  return;
}
```

Servlets

- Aplicación Ejemplo. Encuesta. Servlet (IV)

```
try
 {
   int intSI = 0;
   int intNO = 0;
   ResultSet resultado = mandato.executeQuery("SELECT RESPUESTA FROM ENCUESTA;");

     while(resultado.next()){
     String resp = resultado.getString("RESPUESTA");
     if(resp.compareTo("SI")==0) intSI++;
     else intNO++;
    }
    out.println("Gracias por participar en esta encuesta.");
    out.println("<BR>Los resultados hasta este momento son :");
    out.println("<BR>              SI : "+intSI);
    out.println("<BR>              NO : "+intNO);
 }
 catch(java.lang.Exception e)
 {
   System.out.println(e);
   return;
 }


    out.close();
 }
```

Servlets

- Aplicación Ejemplo. Encuesta. Servlet (V)

```
    public void destroy()
  {
    try
    {
      conexion.close();
    }
    catch(java.sql.SQLException e)
    {
      System.out.println(e);
    return;
    }
  }

    HttpServlet methods. Click on the + sign on the left to edit the code.
}
```

Servlets

- Servlet y el Manejo de Sesiones.
  - Los protocolos de Comunicaciones en Internet se pueden dividir en 2 tipos:
    - **Con Estado** (FTP, Telnet, etc.)
    - **Sin Estado** (WWW - HTTP)
  - **ESTADO**: Se sabe quien es el usuario y qué es lo que quiere realizar en cada momento, mientras esté conectado al servidor.

Servlets

- Servlet y el Manejo de Sesiones.
  - El protocolo **HTTP** es un protocolo **SIN estado**. Por tanto, el servidor no necesita saber si un determinado conjunto de peticiones vienen del mismo cliente o de clientes diferentes.
  - Pero, en algunas situaciones, puede ser interesante cuando se desarrollan aplicaciones WEB que este aspecto se tenga en cuenta (estado), por ejemplo en una tienda virtual.
  - El **objetivo de una sesión** es mantener el estado del usuario en el servidor.

Servlets

- Servlet y el Manejo de Sesiones.
  - Una **sesión** se puede definir como un conjunto de interacciones relacionadas, entre un cliente y un Servidor WEB, que tienen lugar en un periodo de tiempo determinado.
  - Debido a que el protocolo HTTP es un protocolo sin estado, el servidor web no conoce a qué sesión pertenece una petición determinada, existen en general dos formas para conocer el estado del usuario:
    - Que el cliente se identifique cada vez que realiza una petición determinada. (REPETITIVA)
    - Usando distintas técnicas que se presentan a continuación

Servlets

- Servlet y el Manejo de Sesiones.
  - Si en el desarrollo de la aplicación WEB utilizamos Servlets para gestionar las peticiones, el desarrollador tiene a su disposición una serie de posibilidades que le ayudan a mantener una sesión:
    - Introduciendo **campos ocultos** en el formulario HTML.
    - **Reescribiendo la URL** incluyendo estos datos como parámetros de la misma.
    - Usando **cookies.**
    - Usando las **herramientas** de seguimiento de sesiones que se incluyen dentro de la **API de Servlets**.

Servlets

- Servlet y el Manejo de Sesiones.
  - La API de JAVA para desarrollar Servlets proporciona un método para el seguimiento de los datos de una sesión, el cual utiliza dos técnicas diferentes:
    - Las **cookies**
    - El objeto **HttpSession**
  - El objeto **HttpSession** se usa para almacenar los datos de una sesión en el contexto del servlet actual.
  - Las **cookies** se utilizan para relacionar a un usuario particular con su objeto de sesión asociado, gracias a un identificador de sesión

Servlets

- Servlet y el Manejo de Sesiones.
  - El objeto **HttpSession** viene definido por la interfaz HttpSession y se obtiene usando el método **getSession()** del objeto **HttpServletResponse**.
  - El objeto HttpSession posee los siguientes métodos:
    - **getCreationTime()** Devuelve la fecha en la fue creada la sesión.
    - **getID()** Devuelve el identificador de la sesión.
    - **getLastAccesedTime()** Devuelve el lapso de tiempo transcurrido desde que un usuario con el mismo identificador de sesión, realizó una petición al servidor.

Servlets

- Servlet y el Manejo de Sesiones.
  - El objeto HttpSession posee los siguientes métodos:
    - **invalidate()** Elimina la sesión actual.
    - **isNew()** Devuelve true si el objeto sesión ha sido creado por el servidor, pero todavía no se ha enviado de vuelta al navegador del usuario correspondiente.
    - **getValueNames()** Devuelve un array de textos que representan todos los nombres de los valores que son almacenados para esta sesión.

Servlets

- Servlet y el Manejo de Sesiones.
  - El objeto HttpSession posee los siguientes métodos:
    - **getValue(String name)** Devuelve el valor almacenado para el nombre pasado como parámetro.
    - **putValue(String name, Object value)** Añade un item “clave=valor” a la sesión.
    - **removeValue(String name)** Elimina un item de la sesión.

Servlets

- Servlet y el Manejo de Sesiones.
  - El objeto HttpSession posee los siguientes métodos:
    - **setMaxInactiveInternal(int interval)** Especifica el máximo intervalo de tiempo, en segundos, que el servidor mantiene la sesión si que exista ninguna otra petición dirigida a la sesión.
    - **getMaxInactiveInternal()** Devuelve el máximo intervalo de tiempo que la sesión ha estado inactiva.

Servlets

- Servlet y el Manejo de Sesiones.
  - Pasos a seguir para la realización de sesiones usando servlets:
    - Lo primero es crear un **objeto sesión**. Utilizando para ello el método **getSession(boolean)** de la clase **HttpServletRequest**.
    - Después una vez creado el objeto sesión, **sólo** queda **leer** de él y **escribir** en él. Cuando **finaliza el servlet**, se invalida la sesión.

Servlets

- Servlet y el Manejo de Sesiones.

Ejemplo (I): Como escribir datos en una sesión.

```
//Se obtiene un objeto sesion
HttpSession sesion=peticion.getSession(true);

//Se añade un nuevo item a la sesión
sesion.putValue("Cliente",strCodigo[0]);
```

Nota: **peticion** es un objeto de la clase HttpServletsRequest

Servlets

- Servlet y el Manejo de Sesiones.

Ejemplo (II): Como leer datos de una sesión

```
//Se obtiene un objeto sesion
HttpSession sesion=peticion.getSession(true);
//Se leen los datos realizando los casting necesarios
String strCodigoC=(String) sesion.getValue("Cliente");
```

Laboratorio de Informática Distribuida

Sesión II. Servlets