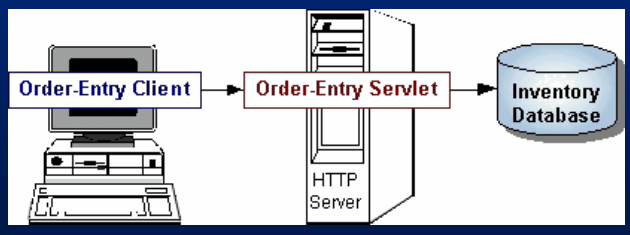
# JAVA SERVLETS.....jsdk

Una forma eficiente para el desarrollo de aplicaciones cliente/servidor en Internet

# SERVLETS

- Los "Servlets" son módulos escritos en JAVA que extienden los servidores orientados al modelo petición-respuesta, tales como los servidores web:
  - Un cliente (navegador) realiza una petición a un servidor a través del protocolo HTTP.
  - Un servidor responde devolviendo un mensaje al navegador en código HTML




2

# Peticiones HTTP

- La sintaxis de la **petición HTTP** está dividida en tres unidades:
  - 1.- La línea de petición consiste en tres campos de texto separados por espacios:
    - El primer campo, conocido como método o comando, especifica el mecanismo que el servidor aplicará al recurso. El método más común es GET, que pregunta al sevidor para que envíe una copia del recurso al cliente.
    - El segundo campo indica el nombre del recurso destinatario de la solicitud, que no es más que el URL y el nombre de dominio del servidor.
    - El tercer campo detalla la versión del protocolo HTTP empleada por el cliente.
  - 2.- Los campos cabecera de la petición: permiten el intercambio de información adicional sobre la solicitud y el cliente. Estos campos actúan como parámetros RPC. Cada cabecera consiste en un nombre seguido por ":" y el valor del campo. El orden en que aparecen no es significativo.
  - 3.- El cuerpo: se usa opcionalmente por los clientes para el envío de información masiva a los servidores.
  - Ejemplo:

```
Línea de petición  { GET /path/fichero.html  HTTP/1.0
Campos cabecera    { Accept:   text/html
                   { Accept:   audio/x
                   { User_agent: MacWeb
```

3

# Respuestas HTTP

- La sintaxis de la **respuesta HTTP** está dividida en tres unidades:
  - 1.- La línea de cabecera de respuesta: consiste en tres campos de texto separados por espacios en blanco.
    - El primer campo es la versión del protocolo HTTP empleado.
    - El segundo campo indica el estado de la respuesta.
    - El último campo es opcional y consiste en una explicación del estado devuelto. Puede ser, por ejemplo, OK.
  - 2.- Los campos cabecera de la respuesta: permiten el intercambio de información adicional sobre el servidor y el documento HTML devuelto al cliente. Cada cabecera consiste en un nombre seguido por ":" y el valor del campo. El orden en que aparecen no es significativo.
  - 3.- El cuerpo: generalmente contiene el documento HTML que el cliente solicitó.
  - Ejemplo:

```
Línea de petición  { HTTP/1.0  200   OK
Campos cabecera    { Server:   NCSA/1.3
                   { Mime_version:  1.0
                   { Content_type:  text/html
                   { Content_length: 2000

Cuerpo             { <HTML> ..... </HTML>
```

4

## Características de los Servlets

- Permite independencia de la plataforma: Java
- API bien definida: Java Servlet Development Kit (JSDK).
- La API está basada en interfaces Java incluidas en las extensiones de "*javax*".
- La mayoría de las peticiones de los clientes se realizan a través de formularios HTML. Las respuestas de los servidores son páginas Web a los clientes.
- **El servidor sólo carga una instancia de cada servlet.**
- **Una vez cargado, permanece en memoria disponible a cualquier petición de los clientes.**
- **Diferentes *threads* pueden ejecutar concurrentemente el método *service()*. El sincronismo se puede realizar dentro del método *service()*.**

5

## Sustitución de los CGI

- Los CGI (Common Gateway Interface) se pueden desarrollar en muchos lenguajes: C, C++, perl, java, etc.,
- La mayoría de los servidores Web soportan CGI.
- Básicamente hacen uso de los formularios HTML:
  - FORM ACTION="http://www.website.com/cgi-bin/cgi-exec-file"
- Principal problema:
  - No son eficientes ya que el servidor ejecuta una copia del CGI por cada petición del cliente y por tanto la sobrecarga que se produce es significativa.
- Servlets sólo ejecutan una copia del mismo.
- Los "Servlets" son una forma eficiente de reemplazar los scripts CGI

6

# Servlets vs. CGI

- Servlets se escriben en Java y por tanto pueden hacer uso de los paquetes Java.
- Los Servlets son "portables" entre diferentes servidores, cumpliendo la premisa java: escribir una vez, ejecutar en cualquier lugar.
- Los Servlets son seguros cumpliendo las características de seguridad de java, como el manejo de la memoria y la seguridad de la máquina virtual.
- Los Servlets son más rápidos que los CGI al estar cargados en memoria, y no degradan el sistema cuando existen muchas peticiones de clientes.

7

# Arquitectura de los Servlets

- El paquete **javax.servlet** está formado por:

**3 Clases**

GenericServlet, ServletInputStream, ServletOutputStream

**5 Interfaces:**

Servlet, ServletConfig, ServletRequest, ServletResponse, ServletContext

8

# Clase GenericServlet

- Implementa las interfaces *Servlet* y *ServletConfig*
- Esta clase ofrece una implementación simple de los métodos *init()* y *destroy()* del ciclo de vida de los servlets.
- También la implementación del método de gestión de "logs" definida en la interfaz *ServeltContext*.
- Los programadores de servlets deben sobreescribir e implementar el método abstracto "*service()*".

9

# Interfaz Servlet

- Define los métodos estándar que todo servlet debe implementar.
- Para crear un servlet, se suele extender la clase GenericServlet o la más especializada HttpServlet.
- Un servidor inicializa un servlet llamando a su método *init()*. Este método pasa al servlet un objeto tipo *ServletConfig*, que contiene la configuración de arranque y los parámetros de inicialización.
- Un servidor invoca al método *service()* para pasar las peticiones al servlet para su procesamiento. Este método toma un objeto *ServletRequest* de entrada y devuelve un objeto *ServletResponse*.
- Para finalizar, un servidor puede eliminar un servlet llamando al método *destroy()*.
- El método *getServletConfig()* devuelve el objeto ServletConfig.
- El método *getServletInfo()* devuelve una string con información acerca del servlet (por ejemplo, autor del servlet, fecha, copyright).

10

# Interfaz ServletConfig

- Da acceso al servlet a sus datos de configuración.
- Un servidor pasa un objeto de este tipo cuando se carga por primera vez.
- El método *getInitParameter()* devuelve una string con el valor de los parámetros de inicialización del servlet.
- El método *getInitParameterNames()* devuelve una "enumeration of strings" con los nombres de los parámetros de inicialización.
- El método *getServletContext()* devuelve un objeto tipo ServletContext que contiene el contexto del servlet que lo ejecuta.

11

# Métodos de GenericServlet

- destroy() Elimina el servlet, limpiando cualquier recurso que se está utilizando, además de escribir en el "log", la destrucción del servlet.
- getServletInfo() Devuelve una cadena con información del servlet, como nombre del autor, versión, Copyright.
- init(ServletConfig) Inicia el servlet y hace "log" de la inicialización.
- log(String) Escribe el nombre de la clase y un mensaje en el fichero "log".
- service(ServletRequest, ServletResponse) Realiza el procesamiento de las peticiones del cliente.
- getServletConfig() Devuelve un objeto *servletConfig* conteniendo información de inicio del servlet.
- getInitParameter(String) Devuelve una cadena con el valor del parámetro de inicialización solicitado, o null si no existe.
- getInitParameterNames() Devuelve el nombre de los parámetros de inicialización del servlet como un tipo enumerado de string, o vacío si no existen parámetros.
- getServletContext() Devuelve un objeto *ServletContext* que contiene información acerca del servicio de red en el que se ejecuta el servlet.

12

# Interfaz ServletContext

- Proporciona al servlet información acerca de su entorno.
- También permite hacer logs de operaciones relevantes. El programador es responsable de decidir que quiere tener en los logs.
- El método *getAttribute()* devuelve el valor de un atributo del servicio de red especificado.
- El método *getMimeType()* devuelve el tipo mime del fichero especificado.
- El método *getRealPath()* devuelve el valor del path real.
- El método *getServerInfo()* devuelve el nombre y versión del servidor sobre el que se ejecuta el servlet.
- El método *getServlet()* devuelve el servlet de un nombre especificado. El método *getServlets()* devuelve una enumeración de los objetos servlets del servidor.
- El método *log()* permite escribir un mensaje en el fichero log del servlet.

13

# Interfaz ServletRequest

- Se utiliza para acceder a los datos pasados desde el cliente al servlet.
- Los datos contenidos en un objeto *ServletRequest* incluye nombres y valores de parámetros, atributos y el *ServletInputStream*.
- Un objeto *ServletRequest* es pasado al método *service()*.
- Métodos:
  - *getAttribute(string name)*.... Atributos especiales ( SSL, ....)
  - *getContentLength()*.... Tamaño de los datos de petición.
  - *getContentType()*
  - *getInputStream()*.... Devuelve un objeto tipo ServletInputStream
  - *getParameter(string name)*.... Devuelve el valor del parámetro *name*
  - *getParameterNames()*..Devuelve el nombre de parámetros (enumeration)
  - *getParameterValues()*..Devuelve el valor de parámetros (array de string)
  - *getProtocol()*, *getRemoteAddress()*, *getRemoteHost()*, *getServerName()*, *getServerPort()*
  - *getScheme()*.... Devuelve el tipo de URL utilizado: ftp, http. https

14

# Interfaz ServletResponse

- Se utiliza para pasar los datos desde el servlet al cliente.
- Un objeto *ServletResponse* es pasado al método *service()*.
- Métodos:
  - *getOutputStream()*....Devuelve un flujo de salida que el servlet utilizará para enviar datos al cliente.
  - *setContentLength()*....Define el tamaño de los datos a enviar al cliente.
  - *setContentType()*....Define el tipo de contenido a enviar al cliente.

15

# Clase ServletInputStream

- Ofrece un flujo de entrada (input stream) que el servlet utiliza para leer las peticiones y los datos que vienen de los clientes.
- Define un método abstracto *readLine()* que lee un array de bytes con la posibilidad de indicar un offset. Este método lee un número de bytes determinado o hasta el siguiente fin de línea.

16

## Clase ServletOutputStream

- Ofrece un flujo de salida (output stream) que el servlet utiliza para enviar las respuestas a los clientes.
- El método sobrecargado *print()* permite escribir en la respuesta al cliente tipos básicos de java como booleanos, caracteres, floats, ints.
- El método sobrecargado *println()* permite lo mismo que el anterior pero añade al final un CRLF.

17

## javax.servlet.http

- Es la versión Servlet de los CGI.
- Consiste de 2 clases y dos interfaces que extienden a los servlets para su uso con el protocolo HTTP:
  - La clase abstracta *HttpServlet*.
    - Es la base de este paquete.
    - Extiende la clase GenericServlet ofreciendo soporte de peticiones http.
    - Las interfaces *HttpServletRequest* y *HttpServletResponse* son extensiones de *ServletRequest* y *ServletResponse*, que permiten la gestión de parámetros en el protocolo HTTP.
  - La clase *HttpUtils* proporciona funciones de apoyo de HTTP.

18

# Clase HttpServlet

- Extiende la clase GenericServlet con 6 nuevos métodos para su uso con el protocolo HTTP.
- Debemos sobreescribir al menos un método, normalmente el método *doPost()* si hacemos peticiones con el POST de HTTP o con el *doGet()* si las realizamos con el GET de HTTP.
- Si hacemos gestión de recursos propia, se deben sobreescribir los métodos *init()* y *destroy()*.
- Para gestionar comandos HTTP como OPTIONS, PUT, DELETE, TRACE, debemos sobreescribir el método *service()*.
- Sólo el método *service()* es público, el cual hace llamadas a otros métodos definidos dentro de esta clase.
- *getLastModified()* devuelve el tiempo en que la página fue modificada por última vez.

19

# Métodos de la Clase HttpServlet

- doPost(HttpServletRequest, HttpServletResponse)
- doGet(HttpServletRequest, HttpServletResponse)
- service(HttpServletRequest, HttpServletResponse)
- getLastModified(HttpServletRequest)

20

## Interfaz HttpServletRequest

- Representa una petición HTTP al servlet.
- La implementación de HttpServletRequest la hace el programador de servicios de red para uso de servlets.
- Ofrece métodos para obtener información de la cabecera HTTP.
- Métodos:
  - *getAuthType()*, *getDateHeader()*, *getHeader()*, *getHeaderNames()*, *getIntHeader()*
  - *getMethod()*
  - *getPathInfo()*, *getPathTranslated()*
  - *getQueryString()*
  - *getRemoteUser()*, *getRequestURI()*, *getServletPath()*

21

## Interfaz HttpServletRequest (+)

- Métodos gestión de cookies:
  - *getCookies()*
- Métodos gestión de sesiones de usuario:
  - *getRequestedSessionId()*
  - *getSession(boolean)*
  - *isRequestedSessionIdFromCookie()*
  - *isRequestedSessionIdFromUrl()*
  - *isRequestedSessionIdValid()*

22

## Interfaz HttpServletResponse

- Representa la respuesta HTTP del servlet al cliente.
- La implementación de HttpServletResponse la hace el programador de servicios de red para uso de servlets.
- Ofrece métodos para generar dinámicamente información para el cliente.
- Métodos:
  - *sendError()*: enviar un error al cliente con mensaje y código de error
  - *sendRedirect()*: enviar al cliente como respuesta una nueva URL.
  - *setDateHeader()*: insertar campo fecha a la cabecera respuesta.
  - *setHeader()*: insertar una campo nuevo en la cabecera respuesta.
  - *setIntHeader()*: insertar una campo y un valor entero en la cabecera.
  - *setStatus()*: especificar un código de estado y un mensaje a ese código.
  - *containsHeader()*: "true" si existe un campo en la cabecera con un nombre determinado.
  - *addCookies()*: permite añadir una Cookie a la respuesta.

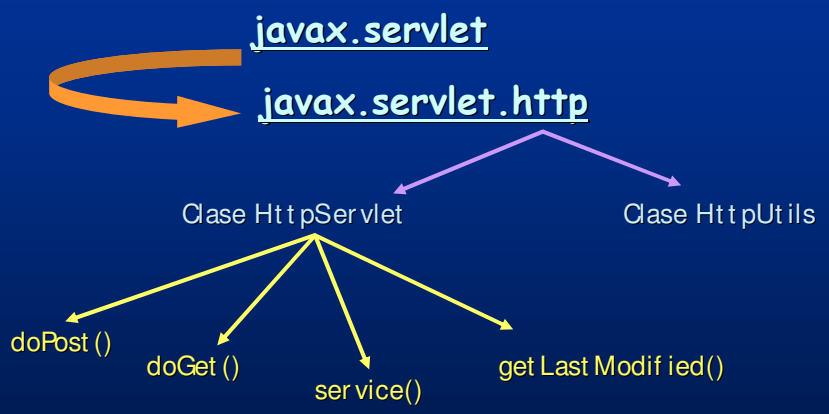
23

## Clase HttpUtils

- Ofrece un conjunto de métodos útiles para HTTP.
- Métodos:
  - *getRequestURL()*: toma como entrada HttpServletRequest y devuelve el URL desde donde el cliente se conectó.
  - *parsePostData()*: toma como entrada un ServletInputStream, lo procesa y devuelve como tabla hash los datos del formulario POST como pares (nombre/valor).
  - *parseQueryString()*: analiza una query string y devuelve una tabla hash con los pares (nombre/valor).

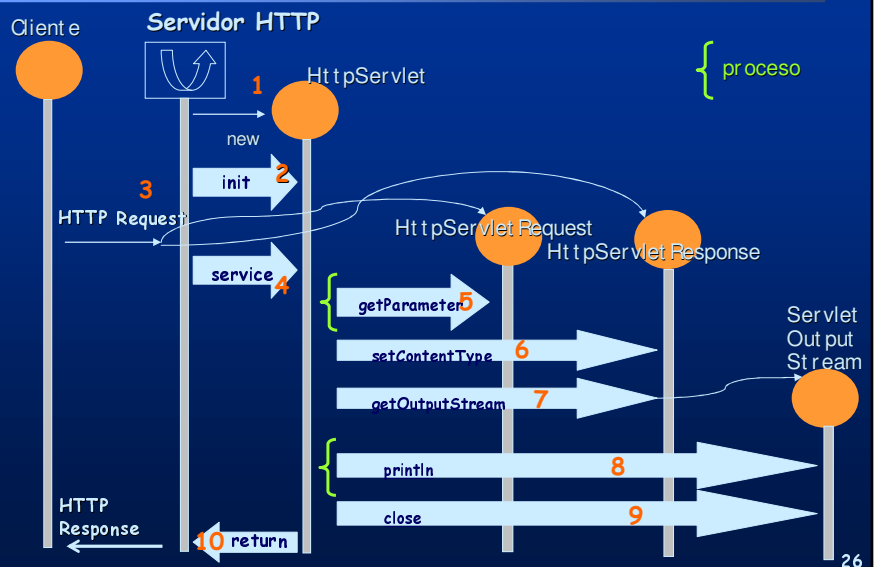
24

# Resumen Arquitectura Servlets




25

# Ciclo de Vida




26

## Método *init()*

- Permite inicializar un servlet.
- Acepta objetos tipo *ServletConfig* como parámetro.
- El objeto *ServletConfig* contiene la información de configuración del servidor.
- En este método se suelen ejecutar todas aquellas funciones comunes y que sólo deben realizarse una sóla vez:
  - Cargar drivers de JDBC.
  - Abrir bases de datos.
  - Obtención de parámetros de inicialización del servlet.

27

## Método *getServletConfig()*

- Devuelve el objeto *ServletConfig* a pasar al método *init()*.

## Método *destroy()*

- Finaliza la ejecución de un servlet.
- Sobre-escribiéndolo permite realizar operaciones de cerrar bases de datos, etc.

## Método *getServletInfo()*

- Devuelve una string con información del autor, versiones y copyright, por ejemplo.

28

# Método *service()*

- Es el núcleo de los servlets.
- El servidor llama al método service() para ejecutar las peticiones.
- Acepta objetos tipo *ServletRequest* y *ServletResponse* como parámetros.
- El objeto *ServletRequest* encapsula el flujo de datos del cliente al servidor
- El objeto *ServletResponse* encapsula el flujo del servidor al cliente.

29

# Ejemplo

- Objetivo: Un simple HolaMundo !!
- Formas de ejecutarlo:
  - 1) Llamar directamente al servlet:
    - http://www.csi.ull.es:8080/servlet/HolaMundo
  - 2) Página WEB que hace las funciones de cliente:
    - http://www.csi.ull.es/Holamundo.html

30

## Servidor con soporte de Servlets

- Muchos servidores Web tienen soporte de servlets: IIS, Apache, .....

  *www.csi.ull.es/servlet/*

  - Sólo el administrador del sistema tiene acceso a ese directorio.

- Para pruebas de desarrollo de servlets se dispone en el JSDK del "*servletrunner*"

  - Permite ejecución en un ordenador necesidad de un servidor Web.

31

## Ejecutar con servletrunner

```
E:> servletrunner -d c:\directorio_de_los_servlets\
        servletrunner starting with settings:
          port = 8080
          backlog = 50
          max handlers = 100
          timeout = 5000
          servlet dir = c:\directorio_de_los_servlets\
          document dir = c:\directorio_de_los_html\
          servlet propfile = c:\directorio_de_los_servlets\servlet.properties


            Usage: servletrunner [options]
            Options:
              -p port     the port number to listen on
              -b backlog  the listen backlog
              -m max      maximum number of connection handlers
              -t timeout  connection timeout in milliseconds
              -d dir      servlet directory
              -s filename servlet property file name
```

32

# Java Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HolaMundo extends HttpServlet {
   public void doGet ( HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
            PrintWriter out;
            String  title = "Desde el Servlet";
            // set content type and other response header fields first
            response.setContentType("text/html");
            // then write the data of the response
            out = response.getWriter();
            out.println("<HTML><HEAD><TITLE>");
            out.println(title);
            out.println("</TITLE></HEAD><BODY>");
            out.println("<H1>" + title + "</H1>");
            out.println("<P>1,2,3 ... Hola Mundo !!!");
            out.println("</BODY></HTML>");
            out.close();
   }
}
```

33

# Compilar el servlet

- **Compilación del servlet consiste en:**

**javac -c classpath \directorio_jsdk\lib\jsdk.jar  mi_servlet.java**

- **Otra alternativa es colocar el fichero jsdk.jar que viene con la distribución de JSDK en el directorio \jdk1.2\jre\lib\ext y compilar:**

**javac  mi_servlet.java**

34

# Fichero "*servlet.properties*"

- **Es necesario un fichero donde se describen las propiedades de los diferentes servlets:**

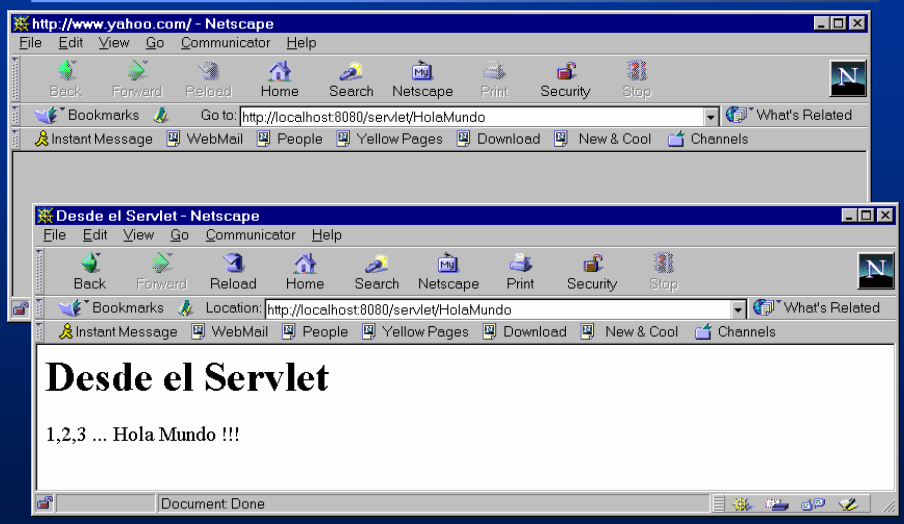
```
# @(#)servlets.properties 1.86 97/11/14
# Servlets Properties
#
# servlet.<name>.code=class name (foo or foo.class)
# servlet.<name>.initArgs=comma-delimited list of {name, value} pairs
#                         that can be accessed by the servlet using the
#                         servlet API calls
# HolaMundo servlet
servlet.HolaMundo.code=HolaMundo
# form_get servlet
servlet.form_get.code=form_get
```

Nombre del servlet

Nombre del .class

35

# Salida *servlet HolaMundo*



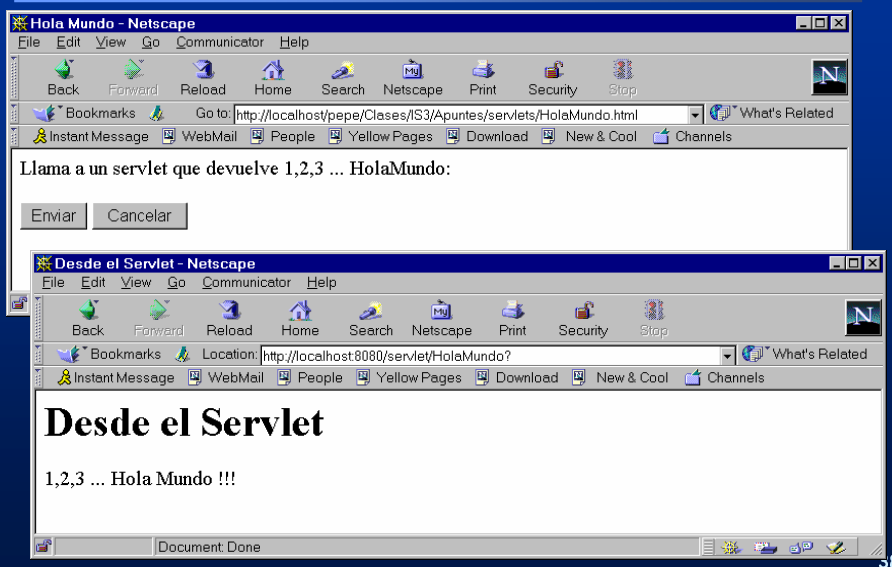

36

# Cliente HTML (Holamundo.html)

```
<html>
<head> <title>Hola Mundo</title> </head>
<body>
Llama a un servlet que devuelve 1,2,3 ... HolaMundo:
<p>
<form method="GET"
    action="http://localhost:8080/servlet/HolaMundo">
    <input type=submit value=Enviar>
    <input type=reset value=Cancelar>
</form>
</body>
</html>
```

37

# *Desde un navegador: HolaMundo.html*




38

# Cliente HTML (form_get.html)

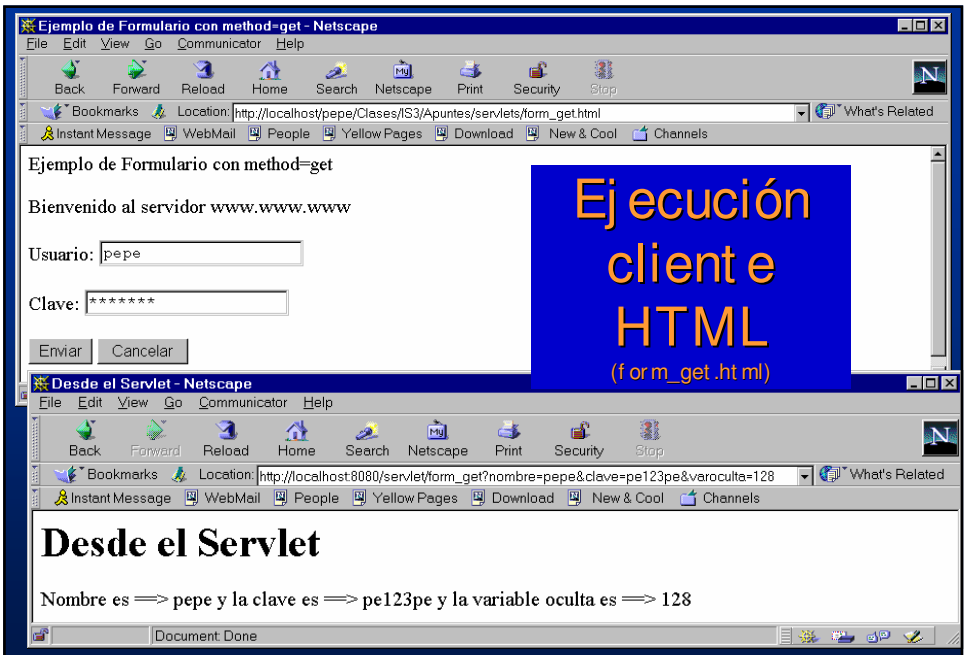
```
<html>
<head> <title>Ejemplo de Formulario con method=POST</title> </head>
<body>
<p>Llama a un servlet que devuelve 1,2,3 ... HolaMundo: </p>
<form method="GET" action="http://localhost:8080/servlet/form_get">
   <p>Usuario: <input type=text name=nombre> </p>
   <p>Clave: <input type=password name=clave> </p>
   <input type=hidden name=varoculta value=128>
   <input type=submit value=Enviar>
   <input type=reset value=Cancelar>
</form>
</body>
</html>
```

39

# servlet (form_get.java)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class form_get extends HttpServlet {
    public void doGet ( HttpServletRequest request, HttpServletResponse
                        response) throws ServletException, IOException {
        PrintWriter out; String title = "Desde el Servlet";
        String nombre = request.getParameter("nombre");
        String clave = request.getParameter("clave");
        String varoculta = request.getParameter("varoculta");
        // set content type and other response header fields first
        response.setContentType("text/html");
        // then write the data of the response
        out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>");
        out.println(title); out.println("</TITLE></HEAD><BODY>");
        out.println("<H1>" + title + "</H1>");
        out.println("<P>Nombre es ==> "+nombre+" y la clave es ==> "+clave+" y la
                    variable oculta es ==>  "+varoculta);
        out.println("</BODY></HTML>");
        out.close();
    }
}
```

40




41

# Cliente HTML (form_post.html)

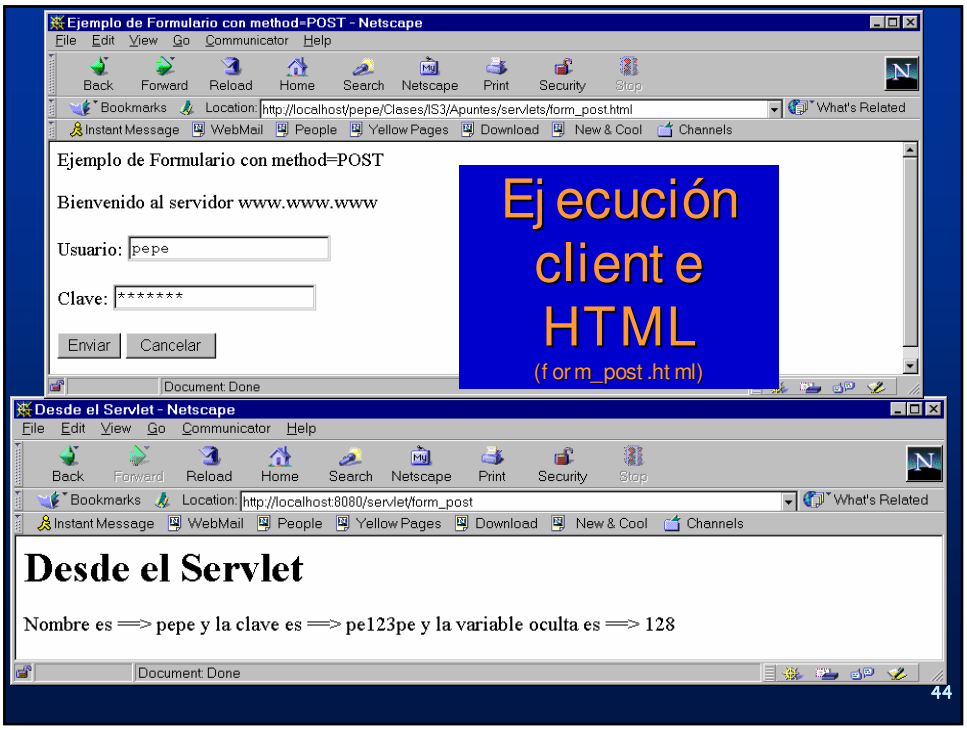
```
<html>
<head> <title> Ejemplo de Formulario con method=POST </title> </head>
<body>
<p>Llama a un servlet que devuelve 1,2,3 ... HolaMundo: </p>
<form method="POST" action="http://localhost:8080/servlet/form_post">
    <p>Usuario: <input type=text name=nombre> </p>
    <p>Clave: <input type=password name=clave> </p>
    <input type=hidden name=varoculta value=128>
    <input type=submit value=Enviar>
    <input type=reset value=Cancelar>
</form>
</body>
</html>
```

42

# servlet (form_post.java)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class form_post extends HttpServlet {
    public void doPost ( HttpServletRequest request, HttpServletResponse
                         response) throws ServletException, IOException {
        PrintWriter out; String title = "Desde el Servlet";
        String nombre = request.getParameter("nombre");
        String clave = request.getParameter("clave");
        String varoculta = request.getParameter("varoculta");
        // set content type and other response header fields first
        response.setContentType("text/html");
        // then write the data of the response
        out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>");
        out.println(title); out.println("</TITLE></HEAD><BODY>");
        out.println("<H1>" + title + "</H1>");
        out.println("<P>Nombre es ==> "+nombre+" y la clave es ==> "+clave+" y la
                     variable oculta es ==>  "+varoculta);
        out.println("</BODY></HTML>");
        out.close();
    }
}
```

43




44

## Parámetros inicialización en "*servlet.properties*"

```
# @(#)servlets.properties 1.86 97/11/14
# Servlets Properties
#
# servlet.<name>.code=class name (foo or foo.class)
# servlet.<name>.initArgs=comma-delimited list of {name, value} pairs
#                         that can be accessed by the servlet using the
#                         servlet API calls
# HolaMundo servlet
servlet.HolaMundo.code=HolaMundo
# form_get servlet
servlet.form_get.code=form_get
# encuesta servlet
servlet.encuesta.code=encuesta
servlet.encuesta.initArgs=\
                    nombre_fichero = encuesta.txt,\
                    directorio = dir_de_trabajo
```

45

## Método init ()

- En el método *init()* se pueden leer los parámetros de inicialización y realizar otros procesos como iniciar las bases de datos.

```
public void init(ServletConfig config) throws ServletException {
      super.init(config);
      fichero = getInitParameter("nombre_fichero");
      if (fichero == null) {
         throw new UnavailableException(this, "falta fichero entrada");
      }
}
```

46

## Ejemplo con acceso a Dases de Datos

- Ejemplo de cliente que desde un formulario HTML hace una consulta a una base de datos a través de un servlet.

- *init()*
  - conecta drivers
  - cre conexión
- *destroy()*
  - cierra la conexión a base de datos
- *doPost()*
  - recoje parámetros desde HTML form y hace consulta en BD. Deeuelve código HTML al cliente.

47

## Ejemplo con Bases de datos

```
<HTML><HEAD> <TITLE>Servlets</TITLE></HEAD>
<BODY><H1>Ejemplo de servlets</H1><BR>
 <FORM NAME="formulario" METHOD=POST
    ACTION="http://localhost:8080/servlet/Usuarios">
  <INPUT TYPE="HIDDEN" NAME="funcion">
  Nombre: <INPUT TYPE="TEXT" NAME="nombre" VALUE=""><BR>
  Apellidos: <INPUT TYPE="TEXT" NAME="apellidos" VALUE=""><BR>
  <INPUT TYPE="BUTTON" VALUE="Insertar" NAME="insertar"
    onClick="document.formulario.funcion.value = 'insertar';
    document.formulario.submit();">
  <INPUT TYPE="BUTTON" DISABLED VALUE="Borrar" NAME="borrar"
    onClick="document.formulario.funcion.value = 'borrar'; document.formulario.submit();">
  <INPUT TYPE="BUTTON" DISABLED VALUE="Modificar" NAME="modificar"
    onClick="document.formulario.funcion.value = 'modificar';
    document.formulario.submit();">
  <INPUT TYPE="BUTTON" VALUE="Buscar" NAME="buscar"
    onClick="document.formulario.funcion.value = 'buscar'; document.formulario.submit();">
 </FORM>
</BODY>
```

48

# Ejemplo con Bases de Datos

```
import java.sql.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
/**********************************************************
    Servlet de ejemplo, que permite insertar, consultar, borrar y modificar en la
    base de datos Usuarios.
**********************************************************/
public class Usuarios extends HttpServlet {
  /** Conexión a la base de datos. */
  private Connection conn;
  /** Sentencia a ejecutar en la BD. */
  private Statement stmt;
  /** Nombre buscado anteriormente. */
  private String antiguoNombre = null;
```



# Ejemplo con Bases de Datos

```
  /*********************************************************
   * Conecta con la base de datos a través del ODBC.
   *********************************************************/
  public void init(ServletConfig config) throws ServletException {
    super.init(config);
    try {
      // Se obtiene la referencia al driver JdbcOdbc
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      // Se abre la base de datos.
      conn = DriverManager.getConnection("jdbc:odbc:USUARIOS");
      // Se crea la sentencia para actuar sobre la BD.
      stmt = conn.createStatement();
    }
    catch (SQLException e) {
      System.out.println ("Error al abrir la base de datos.");
    }
    catch (ClassNotFoundException e) {
      System.out.println ("No se ha encontrado la clase JdbcOdbcDriver.");
    }
  }
```

# Ejemplo con Bases de Datos

```
/*************************************************************
 Determina que opción ha mandado el usuario y emite la página Web de respuesta.
*************************************************************
public void doPost (HttpServletRequest request, HttpServletResponse response
    ) throws ServletException, IOException { ...}
    // set content type and other response header fields first
          response.setContentType("text/html");
    // then write the data of the response
          PrintWriter out = response.getWriter();
    // Crea la página web de respuesta.
          out.println("<HTML><HEAD><TITLE>");
          out.println("Servlets");
          out.println("</TITLE></HEAD><BODY>");
          out.println("<H1>Ejemplo de servlets</H1>");
          out.println("<BR><BR>");
          out.println("<FORM NAME=\"formulario\" METHOD=POST
                    ACTION=\"http://localhost:8080/servlet/Usuarios\">");
          out.println("<INPUT TYPE=\"HIDDEN\" NAME=\"funcion\">");
```

51

```
/*************************************************************
 Determina que opción ha mandado el usuario y emite la página web de respuesta.
*************************************************************/
String funcion = request.getParameter("funcion");
if (funcion.compareTo("insertar") == 0) {
  try{
    stmt.executeUpdate ("Insert Into USUARIOS (NOMBRE, APELLIDOS) Values ('" +
      request.getParameter("nombre") +  "', '" + request.getParameter("apellidos") + "');");
    out.println("Nombre: <INPUT TYPE=\"TEXT\" NAME=\"nombre\" VALUE=\"\"><BR>");
    out.println("Apellidos: <INPUT TYPE=\"TEXT\" NAME=\"apellidos\"VALUE=\"\"><BR>");
    out.println("<BR>Registro insertado<BR><BR>");
    out.println("<INPUT TYPE=\"BUTTON\" VALUE=\"Insertar\" NAME=\"insertar\"
                onClick=\"document.formulario.funcion.value = 'insertar';
   document.formulario.submit();\">");
        out.println("<INPUT TYPE=\"BUTTON\" DISABLED VALUE=\"Borrar\" NAME=\"borrar\"
              onClick=\"document.formulario.funcion.value = 'borrar';
   document.formulario.submit();\">");
        out.println("<INPUT TYPE=\"BUTTON\" DISABLED VALUE=\"Modificar\"
   NAME=\"modificar\"
            onClick=\"document.formulario.funcion.value = 'modificar';
   document.formulario.submit();\">");
    out.println("<INPUT TYPE=\"BUTTON\" VALUE=\"Buscar\" NAME=\"buscar\"
            onClick=\"document.formulario.funcion.value = 'buscar';
   document.formulario.submit();\">");
      }
   catch (SQLException e) {  out.println ("No se pudo insertar el registro<BR><BR>");
   }
```

52

# Ejemplo con Bases de Datos

```
/*****************************************
   ***
 después de realizar todas las operaciones, salir bien del
    código HTML
*****************************************
   ***/
   out.println("</FORM>");
   out.println("</BODY></HTML>");
   out.close();

   } /* fin del método doPost() */
} /* fin del servlet */
```

53