



Taller práctico

Mi Primer Servlet



Taller práctico.

Mi Primer Servlet

En primer lugar, vamos a desarrollar el servlet HolaMundo que ante una petición HTTP, retorne una página HTML con el saludo de rigor. Para desarrollar el servlet:

- Arrancamos eclipse e importamos el proyecto **trabajo 0.0**.
- Antes de nada especificamos en las propiedades del proyecto que el directorio src/java es un directorio fuente en el **Java Build Path**.
- Configuramos el ant como herramienta externa de compilación.
- Ahora que ya tenemos creado el proyecto, añadimos la clase **com.dflanvin.servlet.HolaMundoServlet** por medio del wizard de creación de clase. Debemos especificarle que extiende la clase **javax.servlet.http.HttpServlet**



Taller práctico. Mi Primer Servlet

- Tenemos ya el esqueleto del servlet que vamos a desarrollar. La lógica del servlet debe ser implementada en los métodos **doGet** y/o **doPost**, dependiendo si queremos que nuestro servlet responda a peticiones de uno, otro o ambos tipos. En primer lugar implementamos el doGet(...).

```
public void doGet( HttpServletRequest req, HttpServletResponse res ) throws
    IOException, ServletException
{
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hola Mundo!</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("Bienvenido a mi primera página Güev!");
    out.println("</BODY></HTML>");
}
```



Taller práctico. Mi Primer Servlet

- El método doGet será invocado por el service cuando la petición que llegue sea de tipo GET. Como queremos hacer un servlet muy cordial, hacemos que el doPost delegue también en el doGet (...).

```
public void doPost( HttpServletRequest req, HttpServletResponse
    res) throws IOException, ServletException
{
    doGet( req,res );
}
```



Taller práctico. Mi Primer Servlet

- Ya tenemos el servlet implementado. Eclipse señalará y propondrá los imports que son necesarios para poder compilar el código del servlet. Una vez añadidos, el servlet está terminado, aunque aún no podemos acceder a él desde el navegador.

¿Porqué?

Es necesario darlo de alta en el descriptor de despliegue de la aplicación: el web.xml.



Taller práctico. Mi Primer Servlet

- Insertamos en el **web.xml** la declaración del servlet y del servlet-mapping

```
<servlet>
  <servlet-name>HolaMundo</servlet-name>
  <servlet-
    class>com.dflanvin.servlet.HolaMundoServlet</servlet-
    class>
</servlet>
<!-- Standard Action Servlet Mapping -->
<servlet-mapping>
  <servlet-name>HolaMundo</servlet-name>
  <url-pattern>/HolaMundoCordial</url-pattern>
</servlet-mapping>
```

Taller práctico. Mi Primer Servlet. FIN.

- Desplegamos la aplicación y probamos el servlet accediendo desde el navegador mediante <http://localhost:8888/appblank/HolaMundoCordial>
- Debemos obtener la página web generada por el servlet como respuesta.
Examinar el código fuente de la misma desde el Internet Explorer mediante Ver/Código Fuente.



Más Servlets: Recogiendo la información de usuario.

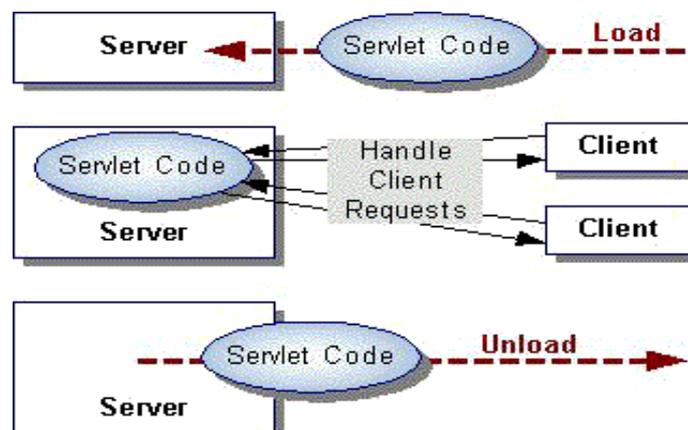
- En CGI, recoger parámetros de un usuario era muy tedioso. Con servlets, trabajamos SIEMPRE con **objetos** java.
- Los parámetros nos llegan en la **request**, que representa el objeto de tipo `HttpServletRequest` que recibimos en la invocación del `doXXX(...)`.
- **Object** `HttpServletRequest.getParameter(nombre)` devuelve:
 - "" (si no hay valor)
 - `null` (si no existe).
 - El valor en caso de haber sido establecido.

Servlets: Políticas de acceso concurrente (threading)

- Los servlets están diseñados para soportar múltiples accesos simultáneos por defecto.
- **Ojo!** El problema puede surgir cuando se hace uso de un **recurso compartido**. Ej, abrimos un fichero desde un servlet.
- Solución,
 - Hacer que el recurso sea el que posea la política de acceso concurrente. Ej: Las bases de datos están preparadas para ello.

Servlets: Ciclo de vida

- Ciclo de vida de un servlet:





Servlets: Ciclo de vida

INICIALIZACIÓN:

- Una única llamada al metodo "init" por parte del servidor.

```
public void init(ServletConfig config) throws ServletException
```

- Se pueden recoger unos parametros concretos con "getInitParameter" de "ServletConfig". Estos parámetros se especifican en el descriptor de despliegue de la aplicación: **web.xml**

DESTRUCCIÓN:

- Cuando todas las llamadas desde el cliente cesen o un temporizador del servidor así lo indique. Se usa el método "destroy"

```
public void destroy()
```



Taller práctico

Captura de la información del usuario



Ejemplo Servlet con parámetros: HolaMundo Personalizado

- Creamos index.html, página con un formulario que nos pasa el parámetro "**Nombre**":

```
<html>
<head>
  <title>Ejemplo "Mi Primer Servlet" </title>
</head>
<body>
  <form
action="http://localhost:8888/appblank/HolaMundoCordial"
method=POST>
  <BR>
  <BR>Introduzca un texto en el cuadro y pulse
"Submit" <BR>
  <BR>
  <input type="text" name="NOMBRE">
  <BR>
  <BR><input type=submit> <input type=reset></form>
</body>
</html>
```



Ejemplo Servlet con parámetros: HolaMundo Personalizado

```
// MiPrimerServlet.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class HolaMundoServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
  {
    ...
    out.println("Bienvenido "+(String)req.getParameter("NOMBRE")+" a
mi primera página Güev!");
    ...
  }
}
```



Gestión de la Sesión.

Mantenimiento del estado de la sesión.

- El protocolo HTTP no posee la capacidad de almacenar estados.
- Se complican mucho las tareas de guardar las acciones (Ej, las Compras) de un usuario.
- Posibles soluciones:
 - Cookies.
 - Añadir información en la URL
 - Usar campos ocultos de formularios (HIDDEN)
 - Empleo del objeto `HttpSession` del servlet.



Servlets: Seguimiento de sesión

- Los servlets proporcionan una solución técnica: La API **HttpSession**.
- Una interfaz de alto nivel construida sobre los cookies y la reescritura de las urls (pero transparente para el desarrollador).
- Permite almacenar objetos.



Servlets: Seguimiento de sesión

- Trabajar con sesiones:
 - **BUSCAR EL OBJETO HttpSession ASOCIADO A UNA PETICIÓN:** Se usa el método "getSession" de "HttpServletRequest" que devuelve `null` si no hay una sesión asociada. Entonces podríamos crear una pero al ser una tarea sumamente común, se pasa `true` y él mismo se encarga de crear una.



Servlets: Seguimiento de sesión

- AÑADIR y RECUPERAR INFORMACION DE UNA SESION
 - Método **getAttribute("nombre_variable")**. Devuelve:
 - Una instancia de `Object` en caso de que la sesión ya tenga *algo* asociado a la etiqueta **nombre_variable**
 - `null` en caso de que no se haya asociado nada aún.
 - Método **setAttribute("nombre_variable", referencia)**. Coloca el objeto referenciado por **referencia** en la sesión del usuario bajo el nombre **nombre_variable**. A partir de este momento, el objeto puede ser recuperado por este mismo usuario en sucesivas peticiones. Si el objeto ya existiera, **lo sobrescribe**.
 - Método **getAttributes()** retorna una *Enumeration* con los nombres de todos los atributos establecidos en la sesión del usuario.



Servlets: Seguimiento de sesión

getId. Este método devuelve un identificador único generado para cada sesión. Algunas veces es usado como el nombre clave cuando hay un sólo valor asociado con una sesión, o cuando se uso la información de logging en sesiones anteriores.

isNew. Esto devuelve true si el cliente (navegador) nunca ha visto la sesión, normalmente porque acaba de ser creada en vez de empezar una referencia a una petición de cliente entrante. Devuelve false para sesión preexistentes.

getCreationTime. Devuelve la hora, en milisegundos desde 1970, en la que se creo la sesión. Para obtener un valor útil para impresión, pasamos el valor al constructor de Date o al método setTimeInMillis de GregorianCalendar.

getLastAccessedTime. Esto devuelve la hora, en milisegundos desde 1970, en que la sesión fue enviada por última vez al cliente.



Servlets: Seguimiento de sesión

○ CADUCIDAD DE LA SESION:

- Peculiaridad de las Aplicaciones WEB: **No sabemos cuando el usuario se desconecta del servidor**
- Automáticamente el servidor web invalida tras un periodo de tiempo (30') sin peticiones o manualmente usando el método "invalidate".

OJO!

¡SOBRECARGAR LA SESIÓN ES PELIGROSO!

Los elementos almacenados no se liberan hasta que no salta el timeout



Servlets: Contexto de la aplicación

- Se trata de un saco “**común**” a todas las sesiones de usuario activas en el servidor.
- Nos permite compartir información y objetos entre los distintos usuarios.
- Se accede por medio del objeto “`ServletContext`”.

```
public ServletContext getServletContext()
```



Servlets: Contexto de la aplicación

- Para colocar o recuperar objetos del contexto...
 - **Añadir un atributo:** Se usa el método “`setAttribute`” de “`ServletContext`”. El control de que varios servlets manejen un mismo atributo es responsabilidad del desarrollador.
 - **Recoger un atributo:** Se usa el método “`getAttribute`” de “`ServletContext`”. Hay que convertir el objeto que devuelve al tipo requerido (Retorna un tipo `Object`!)

Ejemplo Contador de Visitas.