PRINCIPIO BÁSICO DE FUNCIONAMIENTO DE LA WORLD WIDE WEB

Usuario que mediante un browser (cliente) solicita un servicio (páginas HTML, etc.) a una computadora que hace las veces de servidor

Servidores HTTP: En primera instancia se pensaron como servidor de ficheros HTML, pero evolucionaron en dos direcciones complementarias:

1. Añadir más inteligencia en el servidor, y
2. Añadir más inteligencia en el cliente: existen diversas formas de añadir inteligencia a los clientes (páginas HTML):

   - Javascript (<SCRIPT> … </SCRIPT>)
   - Applets de Java (clases de Java que se cargan y se ejecutan en el cliente).
   - Programas CGI (Common Gateway Interface), unidos a los formularios HTML permiten de alguna manera invertir el sentido del flujo de la información.

Los CGI surgieron de la necesidad de generar respuestas dinámicas HTML. Las aplicaciones CGI se ejecutan en el servidor, y corresponden a una interfaz para comunicar al servidor, con programas externos que implementen alguna funcionalidad (cliente), pueden estar escritos en Perl y C/C++. Existen dos formas principales de pasar los datos del formulario al programa CGI: métodos GET y POST. La forma de enviar la respuesta al cliente desde el servidor es a través de una página HTML.

LOS SERVLETS

Los Servlets son las respuesta de la tecnología Java a la programación CGI. Son programas que se ejecutan en un servidor Web y construyen páginas Web dinámicas, las cuales se construyen por un número de razones:

- La página Web está basada en datos enviados por el usuario, por ejemplo, las páginas de resultados de los motores de búsqueda
- Los datos cambian frecuentemente, por ejemplo, un informe sobre el tiempo o páginas de noticias
- Las páginas Web que usan información desde bases de datos corporativas u otras fuentes.

CARACTERÍSTICAS DE LOS SERVLETS

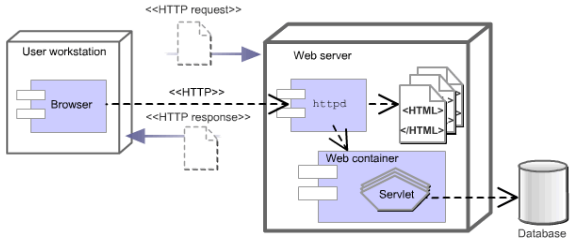
- Los servlets se utilizan para extender o implementar funcionalidades en un servidor.
- Son módulos de software que se ejecutan dentro del entorno de un servidor y proveen servicios de tipo petición/respuesta.
- Son componentes escritos en Java, situados en los servidores e independientes de cualquier protocolo y de cualquier plataforma.
- Son los equivalentes en los servidores a los applets en el cliente. Sin embargo, a diferencia de los applets, los servlets no tienen representación gráfica.
- Los servlets pueden estar en muchos tipos de servidores. Pero su uso más común es en servidores web. Existen muchos servidores web que soportan el uso de servlets (JWS, Tomcat, JRun, Websphere, etc.).
- Un servlet se instancia la primera vez y se mantiene en memoria esperando nuevas invocaciones (el servidor web tiene una máquina virtual java que es la que ejecuta el servlet).
- La comunicación entre servlets es fácil porque ya están en memoria. El servidor web da el mecanismo para que se comuniquen entre ellos.

QUE PUEDE HACER UN SERVLET

- Leer los datos enviados por un usuario: Usualmente de formularios en páginas Web. Pueden venir de applets de Java o programas cliente HTTP.
- Buscar cualquier otra información sobre la petición que venga incluida en esta: Detalles de las capacidades del navegador, cookies, nombre del host del cliente, etc.
- Generar los resultados: Puede requerir consultas a Base de Datos, invocar a otras aplicaciones, computar directamente la respuesta, etc.
- Dar formato a los resultados en un documento: Incluir la información en una página HTML,
- Establecer los parámetros de la respuesta HTTP: Decirle al navegador el tipo de documento que se va a devolver, establecer las cookies, etc.
- Enviar el documento al cliente

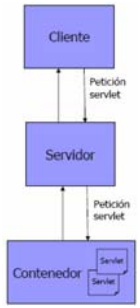
CUANDO Y POR QUÉ USAR UN SERVLET

1. Muchas peticiones desde navegador se satisfacen retornando ***documentos HTML estáticos***, es decir, que están en ficheros
2. En ciertos casos, es necesario generar las páginas HTML para cada petición:
   - **Página Web basada en datos enviados por el cliente:** Motores de búsqueda, confirmación de pedidos
   - **Página Web derivada de datos que cambian con frecuencia:** Informe del tiempo o noticias de última hora
   - **Página Web que usa información de bases de datos corporativas u otras fuentes del la parte del servidor:** Comercio electrónico: precios y disponibilidades



CONTENEDORES DE SERVLETS

Un servlet es un objeto que se ejecuta en un servidor o contenedor, los cuales son los responsables de manejar las peticiones de los clientes, proporcionárselas al servlet correspondiente y devolver la respuesta al cliente.



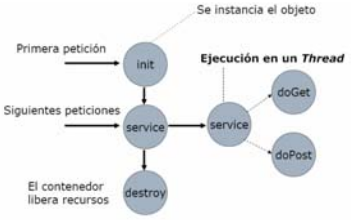
Se encargan de determinar los detalles de la comunicación, cómo se entrega una petición al servlet, cómo se envía una respuesta al cliente.

Se ajustan a la API de servlets, estableciendo el mecanismo de comunicación.

CICLO DE VIDA DE UN SERVLET

1. El cliente solicita una petición a un servidor vía URL.
2. El servidor recibe la petición:

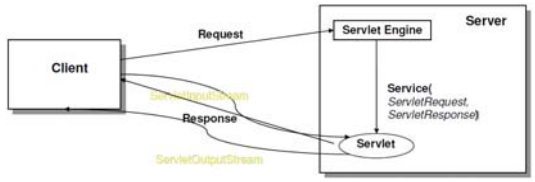
- Si es la 1ra. vez, se utiliza el motor de Servlets para cargarlo y se llama al método init().
- Si ya esta iniciado, la petición se convierte en un nuevo hilo. Un Servlet puede manejar múltiples peticiones de clientes.



3. Se llama al método service() para procesar la petición, devolviendo el resultado al cliente.
4. Cuando termina la ejecución del Servlet y no hay mas requerimientos, se llama al método destroy(), que lo destruye y libera los recursos abiertos. Todo esto lo maneja el contenedor de servlets.

PROCESO DE UNA SOLICITUD (REQUEST)

1. Un cliente hace una solicitud al servidor
2. La solicitud es asignado a un servlet por el servidor.
3. El servlet llama al método service() con el objeto Request y el objeto Response (respuesta)
4. El Server devuelve una respueste a través del servlet.



TIPOS DE PETICIONES POR FORMULARIO

Existen dos métodos de envío de formulario en html:

- POST sirve para enviar datos ocultos entre páginas, o sea los datos no se ven en la barra de direcciones (URL).
- GET envía los datos a través de la barra de direcciones, siendo más riesgoso en cuanto a seguridad se refiere.

Los servlets cuentan con dos métodos: doGet y doPost, los cuales capturan información enviada a través de GET y POST, respectivamente. Por defecto el ingreso a cualquier página (index.html) es por medio de GET.

SERVLETS: MÉTODOS DOGET Y DOPOST

Estos métodos reciben interfaces instanciadas: "HttpServletRequest" para manejo de la información enviada por el usuario, y "HttpServletResponse" para poder enviar una respuesta en forma de página web.

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, java.io.IOException

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, java.io.IOException

Normalmente se implementa uno de los dos y desde el otro, delegamos en el implementado, de forma que pueda responder ambos tipos de peticiones.

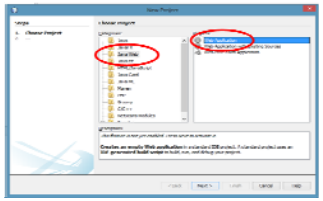
RESPONDIENDO EN HTML

La salida del servlet será normalmente un documento HTML, en el cual se debe:

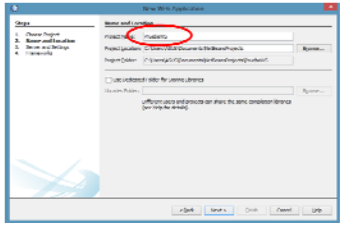
- Indicar la cabecera de la respuesta el tipo de contenido que se va a retornar. El caso más habitual será devolver HTML o XML. Al ser un proceso tan común existe un método que nos lo soluciona directamente: "setContentType" de "HttpServletResponse"
- Crear y enviar código HTML válido.

COMO CREAR UN SERVLET

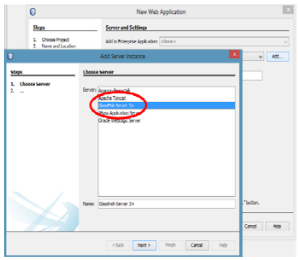
1. Configure su servidor web, ya sea Tomcat o Glassfish. Compruebe que este carga accediendo a la dirección http://localhost: 8080
2. Crear un proyecto nuevo: java Web y dentro de él clic en aplicación web

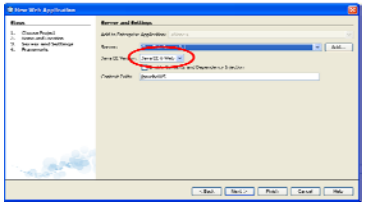


3. Nombre al proyecto, y los demás valores se dejan por defecto

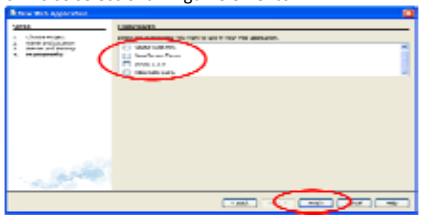


4. Escoger el servidor, por ejemplo: GlassFish Server 3 y la versión de java EE la dejamos como JAVA EE 6 Web

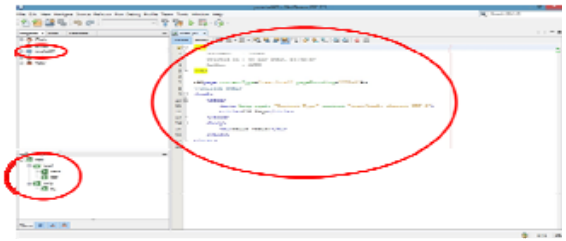




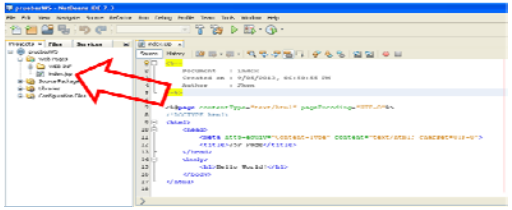
5. La ventana a continuación no se selecciona ningún elemento



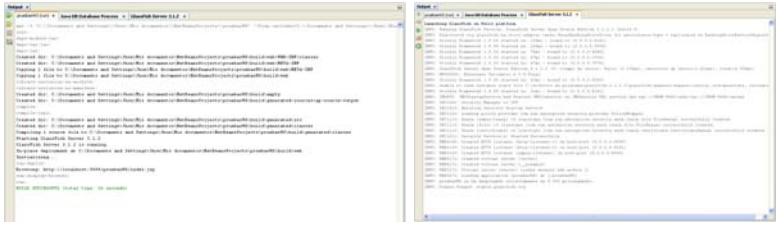
6. Terminada la configuración, se visualiza código HTML, que muestra la página de inicio (“Hello World!“)



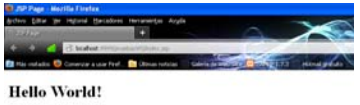
7. Clic en ejecutar, para levantar los servicios, al arrancar se verá un explorador con el mensaje “Hello World!



8. Se iniciará la carga del Servlet



9. Cuando se termine de compilar, se abrirá el navegador por defecto, y mostrará el mensaje



10. Tenga en cuenta que Si tiene instalado por ejemplo ORACLE, este puede estar bloqueando el puerto 8080, por lo que GlassFish producirá un error y no cargará. Para ello edite el archivo → domain.xml, que se encuentra en el directorio de configuración C:\Archivos de programa\glassfish-3.1.2.2\glassfish\domains\domain1\config \domain.xml
11. En la index.jsp modifique el código que se encuentre entre las etiquetas HTML, introduciendo el código que sigue; y ejecute el jsp nuevamente

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Mi Primer JSP</title>
  </head>
  <body bgcolor="red" text="yellow">
    <h1>Esta es mi Pagina</h1>
    <p>Esta es una prueba de mi primera pagina en html, que es ejecutada por un Web
Services</p>
    <br><br><br>
    <marquee bgcolor="black" scrolldelay="100" scrollamount="5" direction="left"
loop="infinite">Escriba aca su nombre</marquee>
  </body>
</html>
```

12. Prueba el siguiente código en un nuevo proyecto de servlets

En el JSP	En el servlet

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP - Pagina de Ejemplo</title>
</head>
 Ejemplo de ServLets
<FORM ACTION="Servlet2" METHOD="POST">
<table>
<tr><td>Digite el Primer Valor:</td><td><INPUT NAME="caja1"></td></tr>
<tr><td>Digite el Segundo Valor: </td><td>
<INPUT NAME="caja2"></td></tr>
<tr><td><INPUT TYPE="SUBMIT" VALUE="Enviar"> </td></tr>
</table>
</FORM>
</html>
	public class Servlet2 extends HttpServlet {
	
	@Override
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
	throws ServletException, IOException {
	doPost(request, response);
	}
	
	@Override
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
	
	String caja1 = request.getParameter("caja1");
	String caja2 = request.getParameter("caja2");
	if (caja1.equals("")) { caja1 = "Nulo";
	} else { caja1 = caja1.toUpperCase(); }
	if (caja2.equals("")) { caja2 = "Nulo";
	} else { caja2 = caja2.toLowerCase(); }
	String Buffer buffer = new StringBuffer();
	buffer.append("<HTML>\n");
	buffer.append("<HEAD>\n");
	buffer.append("<TITLE>Resultado con POST</TITLE>\n");
	buffer.append("</HEAD>\n");
	buffer.append("<BODY BGCOLOR=\"RED\" TEXT=\"white\">\n");
	buffer.append("<H1>Servlet 2 atendio POST</H1>\n");
	buffer.append("<P>Me mandaron:\n");
	buffer.append("<P>Primer Dato Enviado: " + caja1);
	buffer.append("<P>Segundo Dato Enviado: " + caja2);
	buffer.append("<P>La Hora actual es: " + new Date().toString());
	buffer.append("</BODY>\n");
	buffer.append("</HTML>");
	response.setContentType("text/html");
	response.setContentLength(buffer.length());
	response.getOutputStream().print(buffer.toString());
	}
	}

Fuentes:
Web Services Architecture: http://www.w3.org/TR/ws-arch/
Web Services Tutorial: http://www.w3schools.com/webservices/default.asp
Servicios web del API de Google Maps
https://developers.google.com/maps/documentation/webservices/?hl=es
http://www.gxtechnical.com/gxdlsp/pub/genexus/internet/technicalpapers/web_services.htm