Universidad Andrés Bello®

Conectar · Innovar · Liderar

DESARROLLO DE APLICACIONES WEB DINÁMICAS JAVA

Servlets

Introducción a los Servlets y Contenedores Web

Los Servlets son módulos escritos en Java que se utilizan en un servidor, que puede ser ono ser servidor web, para extender sus capacidades de respuesta a los clientes al utilizar las potencialidades de Java. Los Servlets son para los servidores lo que los applets para los navegadores, aunque los servlets no tienen una interfaz gráfica.

Los servelts pueden ser incluidos en servidores que soporten la API de Servlet. La API no realiza suposiciones sobre el entorno que se utiliza, como tipo de servidor o plataforma, ni del protocolo a utilizar, aunque existe una API especial para HTTP.

Los Servlets son un reemplazo efectivo para los CGI en los servidores que los soporten ya que proporcionan una forma de generar documentos dinámicos utilizando las ventajas dela programación en Java como conexión a alguna base de datos, manejo de peticiones concurrentes, programación distribuida, etc. Por ejemplo, un Servlet podría ser responsable de procesar los datos desde un formulario en HTML como registrar latransacción, actualizar una base de datos, contactar algún sistema remoto y retornar un documento dinámico o redirigir a otro Servlet u alguna otra cosa.

¿Cómo es un Servlet?

Un pequeño Servlet de ejemplo es el siguiente:

```
public class SimpleServlet extends HttpServlet {

    // Maneja el método GET de HTPP para
    // construir una sencilla página Web.

    public void doGet (HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    PrintWriter out;
    String title = "Simple Servlet Output";

    // primero selecciona el tipo de contenidos y otros campos de cabecera de la respuesta
    response.setContentType("text/html");
    // Luego escribe los datos de la respuesta
    out = response.getWriter();
    out.println("<HTML><HEAD><TITLE>");
    out.println(title);
    out.println("</TITLE></HEAD><BODY>");
    out.println("<H1>" + title + "</H1>");
    out.println("<P>This is output from SimpleServlet.");
    out.println("</BODY></HTML>");
    out.close();
    }
}
```

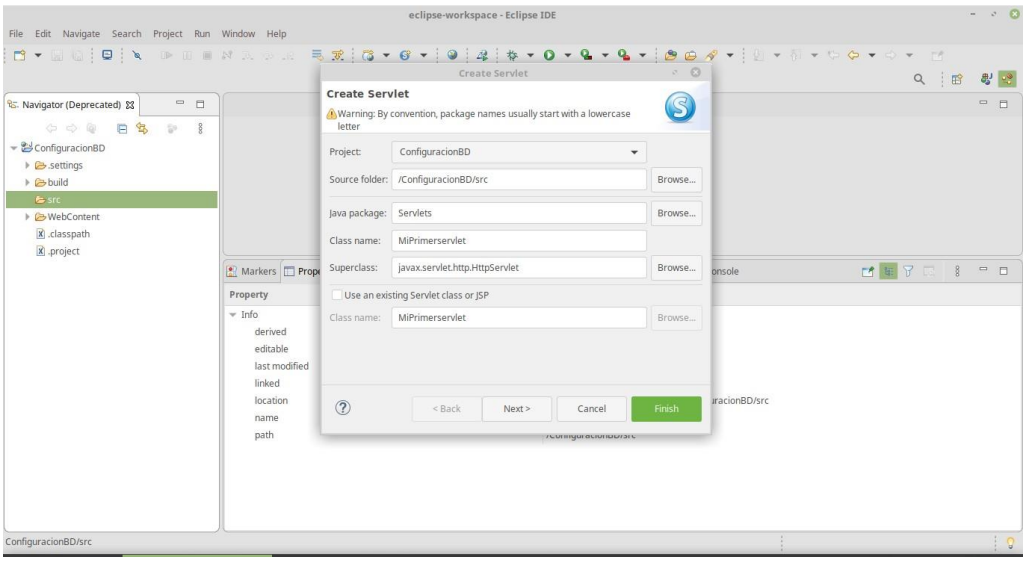
Este Servlet puede ser puesto en un servidor web ya que utiliza el protocolo HTTP para comunicarse. Primero es necesario señalar que el Servlet será del tipo HTTP por lo que seextiende de la clase **HttpServlet**. Al extender de esta clase es necesario definir el método**doGet** para responder la petición. Este método recibe los parámetros dados por el clientea través de la clase **HttpServletRequest** y encapsula la respuesta que se le dará al clientea través de la clase **HttpServletResponse**. El Servlet puede retornar al cliente cualquier tipo de información, desde texto plano hasta un ejecutable, por lo que es necesario señalar inicialmente qué tipo de respuesta se dará a través del método **setContentType**.Luego se obtiene el objeto para poder escribir texto al cliente a través del método **getWriter** con el cual se puede retornar una página web llamado sucesivamente el método **println** hasta terminar con **close**.

Propiedades

- **Manejo de Sesiones:** Se puede hacer seguimiento de usuarios a través de distintosservlets a través de la creación de sesiones.
- **Utilización de Cookies:** Las cookies son pequeños datos en texto plano quepueden ser guardados en el cliente. La API de servlets permite un manejo fácil y limpio de ellas.
- **Multi-thread:** Los servlets soportan el acceso concurrente de los clientes, aunquehay que tener especial cuidado con las variables compartidas a menos que se utilize la interfaz SingleThreadModel.
- **Programación en Java:** Se obtienen las características de multiplataforma o acceso a APIs como JDBC, RMI, etc.

Como crear un Servlet

En nuestro proyecto se buscar la carpeta donde alojaremos nuestro nuevo Servlet (esta carpeta puede variar según la infraestructura que se esté utilizando para trabajar); hacerclick secundario, new > Servlet.



Sesiones y cookies

¿Qué es una Cookie?

En pocas palabras, una cookie es un pequeño dato almacenado en el lado del cliente quelos servidores usan cuando se comunican con los clientes.

Se utilizan para identificar a un cliente al enviar una solicitud posterior. También se pueden usar para pasar algunos datos de un Servlet a otro.

Creando una cookie

La clase Cookie se define en el paquete ***javax.servlet.http***.

Para enviarlo al cliente, necesitamos crear uno y agregarlo a la respuesta:

```
1  Cookie uiColorCookie = new Cookie("color", "red");
2  response.addCookie(uiColorCookie);
```

Establecer la fecha de vencimiento de la cookie

Podemos establecer la edad máxima (con un método maxAge (int)) que define cuántos segundos debe ser válida una cookie determinada para:

```
1  uiColorCookie.setMaxAge(60*60);
```

Establecemos una edad máxima de una hora. Después de este tiempo, la cookie no puedeser utilizada por un cliente (navegador) al enviar una solicitud y también debe eliminarsede la memoria caché del navegador.

Establecer el dominio de la cookie

Un método muy útil en la API de cookies es setDomain (String).

Esto nos permite especificar nombres de dominio a los que el cliente debe entregarlos. También depende de si especificamos el nombre de dominio explícitamente o no.

Vamos a configurar el dominio para una cookie:

```
uiColorCookie.setDomain("example.com");
```

La cookie se entregará a cada solicitud realizada por example.com y sus subdominios. Si no especificamos un dominio explícitamente, se establecerá en el nombre de dominioque creó una cookie .

Por ejemplo, si creamos una cookie de example.com y dejamos el nombre de dominio vacío, se enviará a www.example.com (sin subdominios).

Establecer la ruta de la cookie
Es la ruta especifica en dónde se entregará una cookie.

Si especificamos una ruta explícitamente, se enviará una cookie a la URL dada y a todos sus subdirectorios:

```
uiColorCookie.setPath("/welcomeUser");
```

Implícitamente, se establecerá en la URL que creó una cookie y todos sus subdirectores.

Parámetros

- **Dominio:** Establece el nombre de domino (o dominios) para el cual una cookie esvalida. Se establece a través del método setDomain(String dominio) de la clase Cookie
- **Path:** Establece la uri especifica en la que la cookie será valida (e.g: /admin) es útilsi no queremos exponer la cookie en toda la aplicación. Se establece como es de esperarse con el metodo setPath(String path) de la clase Cookie
- **Expiración:** Indica cuanto debe durar una cookie en el navegador. Este valor se establece en segundos a través del método setMaxAge(int segundos) de la clase Cookie
- **HostOnly:** Esta propiedad es muy importante en términos de seguridad, ya que establece si una cookie podrá ser leída por el cliente o solo por el servidor. Piensenque si alguien logra un ataque XSS bien podrían robar las cookies de sus usuarios sin mayores pormenores. Por eso HostOnly debería ser true siempre que el dominio desde el cual se está intentando leer no sea igual al dominio de origen establecido. Por lo tanto, la clase no expone un método para manipular esta bandera, si no que más bien se establece automáticamente siguiendo el mecanismo establecido en la especificación RFC-6265 Sección 5.3 numeral 6.
- **Secure:** Otro parámetro muy importante que indica que la cookie en cuestión puede ser leída únicamente a través de un protocolo seguro HTTPS o SSL. Y se establece con el metodo setSecure(boolean isSecure) de la clase Cookie.

- **HttpOnly:** Es similar a HostOnly excepto que este si se puede manipular. Funcionacomo una medida de seguridad extra a la especificación establecida y fue desarrollada por microsoft en el 2002 gracias a la incapacidad del IE6 para apegarse a los estándares mundiales. Java al ser un lenguaje de clase mundial ofrece el método setHttpOnly(boolean isHttpOnly) para estos efectos.

¿Qué es una sesión?

Una buena alternativa para no manejar información sensible de nuestros usuarios del lado del cliente son las sesiones, que no es otra cosa que una estructura de datos que esaccedida exclusivamente del lado del servidor. Si sus sistemas son muy concurridos o están detrás de un balanceador de carga manejar sesiones podría representar un reto interesante de resolver, sin embargo, la mayoría de las veces funciona muy bien con su configuración por defecto.

¿Qué es una HttpSession?

La HttpSession es otra opción para almacenar datos relacionados con el usuario en diferentes solicitudes. Una sesión es un almacenamiento del lado del servidor que contiene datos contextuales.

Los datos no se comparten entre diferentes objetos de sesión (el cliente solo puede acceder a los datos de su sesión). También contiene pares clave-valor, pero en comparación con una cookie, una sesión puede contener un objeto como valor. Elmecanismo de implementación de almacenamiento depende del servidor.
Conseguir una sesión

Podemos obtener una HttpSession directamente de una solicitud:

```
1 HttpSession session = request.getSession();
```

El código anterior creará una nueva sesión en caso de que no exista. Podemos lograr lo mismo llamando a:

```
1 request.getSession(true)
```

Compartiendo información entre Servlets

```
private static final void home(HttpServletRequest request,
HttpServletResponse response)
{
  HttpSession sesion = request.getSession();
  session.setAttribute("usuario", "awakelab");
}
```

y así como si nada hemos creado una sesión y le hemos insertado un registro que describeun nombre de usuario. Luego podemos acceder a esta sesión en un request completamente diferente.

```
private static final void panel(HttpServletRequest request,
HttpServletResponse response)
{
  HttpSession sesion = request.getSession();
  Object usuario = (String) session.getAttribute("usuario");
}
```

Aquí obtenemos el nombre de usuario. Vemos que como el getAttribute devuelve unobjeto lo casteamos al tipo correcto para su posterior utilización.

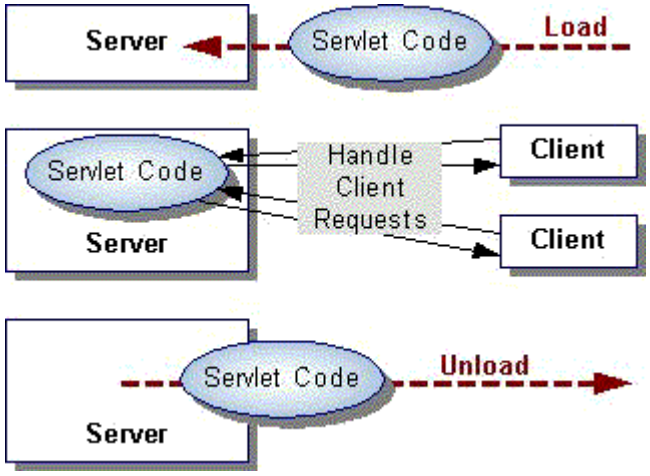
Finalmente, si quisiéramos borrar (invalidar) la sesión, podemos hacerlo de la siguientemanera:

```
private static final void salir(HttpServletRequest
request, HttpServletResponse response)
{
  HttpSession sesion = request.getSession();
  sesion.invalidate();
}
```

Concurrencia con los Servlets

Cada Servlet tiene el mismo ciclo de vida:

- Un servidor carga e inicializa el Servlet.
- El Servlet maneja cero o más peticiones de cliente.
- El servidor elimina el Servlet.



Inicializar un Servlet

Cuando un servidor carga un Servlet, ejecuta el método init del Servlet. La inicialización secompleta antes de manejar peticiones de clientes y antes de que el Servlet sea destruido.

Aunque muchos servlets se ejecutan en servidores multi-thread, los servlets no tienen problemas de concurrencia durante su inicialización. El servidor llama sólo una vez al método init al crear la instancia del Servlet, y no lo llamará de nuevo a menos que vuelvaa recargar el Servlet. El servidor no puede recargar un Servlet sin primero haber destruidoel Servlet llamando al método destroy.

Interactuar con Clientes

Después de la inicialización, el Servlet puede manejar peticiones de clientes. Estas respuestas son manejadas por la misma instancia del Servlet por lo que hay que tener cuidado con acceso a variables compartidas por posibles problemas de sincronización entre requerimientos concurrentes.

Destruir un Servlet

Los servlets se ejecutan hasta que el servidor los destruye, por cierre el servidor o bien apetición del administrador del sistema. Cuando un servidor destruye un Servlet, ejecuta el método destroy del propio Servlet. Este método sólo se ejecuta una vez y puede ser llamado cuando aún queden respuestas en proceso por lo que hay que tener la atenciónde esperarlas. El servidor no ejecutará de nuevo el Servlet, hasta haberlo cargado e inicializado de nuevo.

Creando formularios para capturar información

Código Vista JSP - Ejemplo login.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html;charset=ISO-8859-1">
<title>Login</title>
</head>
<body>
     <h1>Bienvenido a Login</h1>
     <table>
          <form method="get">

               <tr>
                    <td>Usuario</td>
                    <td><input type="text"
name="usuario"></td>
               </tr>
               <tr>
                    <td>Contraseña</td>
                    <td><input
type="password"name="contrasena"></td>
               </tr>
               <tr>
               <td><input type="hidden" name="parametro"
value="sesion"></td>
                    <td><input type="submit" value="Iniciar
Sesión"></td>
               </tr>
          </form>
     </table>
</body>
</html>
```

Esta vista simula un **formulario login**, contiene los datos de usuario y contraseña que vana ser enviados al Servlet.

Antes que nada recordar que para que puedas enviar una petición al servidor y esta puedaser procesada por un Servlet, los datos que vayas enviar deben estar dentro de un **formulario**, en este caso con la propiedad **method=”get”**.

Entonces para que pases un parámetro o varios al Servlet estos siempre deben estar dentro de un formulario y como propiedad el método con el que quieres enviar **POST** o **GET**.

También recordarte que para obtener cualquier valor dentro del Servlet lo haces a travésdel nombre o **name** del elemento HTML de tu vista.

El valor del elemento HTML lo puedes dar dinámicamente como es el caso del elemento **<input type=”text name=”usuario”>** que es una caja de texto para guardar el usuario.

O también lo puedes hacer estáticamente a través de la propiedad **value** del elementoHTML, como es el caso del elemento

```
<input type=”hidden” name=”parametro” value=”sesion”>
```

Controlando parámetros de un GET request

El siguiente código muestra como recibir un parámetro proveniente desde un formulario,y generar una acción a través del mismo.

Código Servlet get

```
String parametro = request.getParameter("parametro");
if (parametro != null) {

    if (parametro.equals("sesion")) {
        getServletContext()

        .getRequestDispatcher("/vista/verLogin.jsp")

        .forward(request, response);

    } else if (parametro.equals("login")) {
        getServletContext()

        .getRequestDispatcher("/vista/login.jsp")

        .forward(request, response);

    }

} else {
```

Controlando parámetros de un POST request

Esta vista simula un formulario login, contiene los datos de usuario y contraseña que vana ser enviados al Servlet.

Antes que nada recordar que para que puedas enviar una petición al servidor y esta puedaser procesada por un Servlet, los datos que vayas enviar deben estar dentro de un formulario, en este caso con la propiedad method=”get”.

Entonces para que pases un parámetro o varios al Servlet estos siempre deben estar dentro de un formulario y como propiedad el método con el que quieres enviar POST o GET.

También recordarte que para obtener cualquier valor dentro del Servlet lo haces a travésdel nombre o name del elemento HTML de tu vista.

El valor del elemento HTML lo puedes dar dinámicamente como es el caso del elemento <input type=”text name=”usuario”> que es una caja de texto para guardar el usuario, o también lo puedes hacer estáticamente a través de la propiedad value del elemento HTML, como es el caso del elemento <input type=”hidden” name=”parametro” value=”sesion”>.

El archivo postlogin.jsp deberá quedar de la siguiente manera:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Login</title>
</head>
<body>
      <h1>Bienvenido a Login</h1>
      <table>
            <form method="post" action="?parametro=sesion">
                  <tr>
                        <td>Usuario</td>
                        <td><input type="text" name="usuario"></td>
                  </tr>
                  <tr>
                        <td>Contraseña</td>
                        <td><input type="password"
name="contrasenia"></td>
                  </tr>
                  <tr>
                        <td><input type="submit" value="Iniciar
Sesión"></td>
                  </tr>
            </form>

      </table>
</body>
</html>
```

Las dos cosas principales que cambien con respecto al método GET es el método en el formulario que es POST y la propiedad action que en este caso es action=”?parametro=sesion”.

```
String parametro = request.getParameter("parametro");
if (parametro == null) {
    getServletContext()
    .getRequestDispatcher("/vista/login.jsp")
    .forward(request, response);
}else{
    getServletContext()
    .getRequestDispatcher("/vista/login.jsp")
    .forward(request, response);
}
```

La propiedad action permite definir a qué Servlet se va hacer la petición.

Ahora definimos el código que va llevar el método doPost(), puesto que vamos a enviarparámetros por POST:

```
String parametro = request.getParameter("parametro");
System.out.println("Parámetros a través de POST");
if (parametro != null) {
    if (parametro.equals("sesion")) {
        getServletContext()
        .getRequestDispatcher("/vista/verLogin.jsp")
        .forward(request, response);
    } else if (parametro.equals("login")) {
        getServletContext()
        .getRequestDispatcher("/vista/login.jsp")
        .forward(request, response);
    }
} else {
    getServletContext()
    .getRequestDispatcher("/vista/login.jsp")
    .forward(request, response);
}
```

Anexo: Referencias

[1] Qué es Java Enterprise (J2ee, JEE)
Referencia: https://www.fundesem.es/bt/publicacion-java-ee-y-el-desarrollo-web-un-enfoque-de-aprendizaje

[2] Qué son los Servlets
Referencia: https://users.dcc.uchile.cl/~jbarrios/servlets/general.html

[3] Utilizando métodos post y get
Referencia: https://www.ecodeup.com/enviar-parametros-servlet-desde-una-vist a-utilizando-post-get/

[4] Sesiones y cookies
Referencia: https://ricardogeek.com/manejo-de-sesiones-y-cookies-en-servlets-java/

[5] Etiquetas JSTL
Referencia: http://www.jtech.ua.es/ayto/ayto2008/jsp/sesion07-apuntes.html