Manejo de información entre servlets

¡Comencemos!

¿Qué aprenderás?

- Entender los conceptos de request y response.
- Entender el funcionamiento de métodos getParams.
- Enviar parámetros desde el cliente.
- Aplicar el envío de información entre servlets.

Introducción

Los servlet no solamente se encargan de devolver recursos a los clientes que lo solicitan, también pueden comunicarse con otras clases servlets para compartir información y articular el flujo correcto de un sistema web.

Esta capacidad es una gran razón del porqué esta tecnología es tan demandada y veremos la forma en la cual estas clases pueden comunicarse entre sí compartiendo sus variables y dando funcionalidad al sistema.

Creación de servlet de generación respuesta

Los servlets tienen la capacidad de recibir parámetros desde los clientes, para luego procesarlos y generar salidas. El objeto request es el encargado de manejar tales parámetros y gracias a él podemos acceder a los datos enviados. Para entender el funcionamiento de esta característica vamos a implementar una nueva funcionalidad en nuestro primer ejemplo.

El objetivo del ejercicio es permitir que el servlet obtenga dos parámetros enviados por nosotros y que luego nos devuelva un saludo con nuestro nombre, apellido y la fecha actual.

```
    @WebServlet("/generadorRespuesta")
    public class GeneradorRespuesta extends HttpServlet {
      private static final long serialVersionUID = 1L;

      @Override
      protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException {
            String nombre;
            String apellido;
            String fechaActual;
            PrintWriter printWriter = response.getWriter();
            SimpleDateFormat formato = new
SimpleDateFormat("dd-MM-yyyy");
            fechaActual = formato.format(new Date());
            nombre = request.getParameter("nombre");
            apellido = request.getParameter("apellido");

            printWriter.println("<html>");
            printWriter.println("<body>");
            printWriter.println("Bienvenido/a "+ nombre + " " +
apellido);
            printWriter.println("La fecha es: " + fechaActual);
            printWriter.println("</body>");
            printWriter.println("</html>");
      }
    }
```

El código mostrado corresponde a una nueva clase de nombre *GeneradorRespuesta*, la cual debes crear en el mismo paquete que aloja nuestro ejemplo anterior:

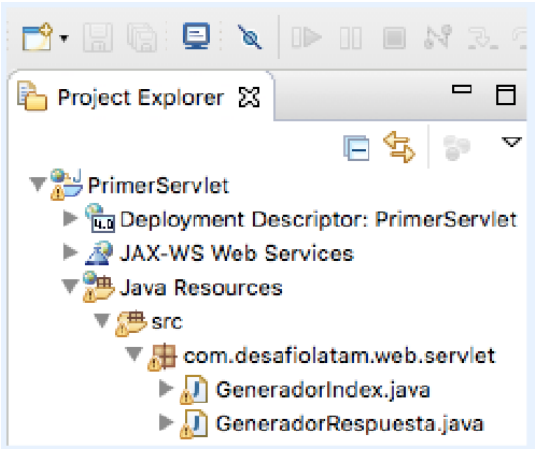

Imagen 1. Estructura del proyecto.
Fuente: Desafío Latam

No olvides el procedimiento para crear un *servlet*:

- La clase debe extender de *HttpServlet*.
- La clase debe implementar algún método del padre, en nuestro caso *doGet*.
- La clase debe ir acompañada de la anotación *@WebServlet("path")*.

Si analizamos el código, la mayoría sigue igual, a excepción de las dos variables de tipo String que declaramos y su asignación.

En cada variable *String (nombre, apellido)* estamos guardando el valor del request mediante:

```
nombre = request.getParameter("nombre");
apellido = request.getParameter("apellido");
```

La variable request, es el parámetro de entrada de nuestro método.

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response)
```

En estas variables van los parámetros que el cliente envía al *servlet*. Como te podrás dar cuenta la variable request posee el método *getParameter()*, y además de él tiene otros métodos más, los cuales no utilizaremos aún.

Con esas sentencias ya tenemos en nuestro poder el nombre y el apellido que vienen desde el cliente, y podemos hacer lo que queramos con ellas. En este caso solo vamos a mostrar un saludo con la fecha.

Imagen 2. Respuesta en nuestro servlet.
Fuente: Desafío Latam

Perfecto, nuestro servlet ya puede recibir parámetros, procesarlos y generar una salida.

En los inicios de la unidad, mencionamos que los servlet pueden devolver contenido dinámico y, si analizamos nuestro ejemplo, podemos ver que la generación es totalmente dinámica, ya que por un lado la fecha se actualizará según el día en que se llame al servicio, y el nombre y apellido cambiará de acuerdo al ingreso que haga el usuario.

Si bien es un ejemplo básico, tienes que pensar en las múltiples aplicaciones de esta mecánica; por ejemplo, puedes pensar en enviar un formulario completo con información de las cuentas bancarias de los usuarios para que el *servlets* las valide, las procese y las guarde a una base de datos, o también puedes pensar en un generador de memes, en donde se le envíe al *servlet* los textos para que éste pueda armar una foto con lo que le enviaste y así generar la imagen, o quizás alguna otra aplicación que se te pueda ocurrir.

Envío de parámetros desde el cliente

Ya tenemos funcionando nuestro servlet y puede recibir peticiones http con parámetros, pero ¿cómo se envían los parámetros desde el cliente? Vamos a analizar la forma básica:

http://localhost:8080/PrimerServlet/generadorRespuesta?nombre=JUANITO&apellido=PEREZ

Por el momento no estamos utilizando una página web como tal, pero la idea es la misma y más que la forma, analizaremos el fondo.

Vemos que la url se compone de:

- **http://localhost:8080**

Corresponde a la dirección del servidor que aloja al servlet. En este caso como estamos trabajando sobre nuestras máquinas, el servidor es local y nuestra dirección es localhost (ip 127.0.0.1). La dirección ip va acompañada del puerto de conexión, el cual por defecto es 8080. Es importante entender que en un entorno real, el *servlet* estará en algún servidor externo a nosotros, por lo cual en vez de tener la dirección localhost:8080, tendremos algo así: **http://192.186.8.123:8080.**

Esa sería la dirección del servidor remoto que contiene el servlet que queremos consultar.

- **/PrimerServlet**: Es el nombre de nuestro proyecto.
- **/generadorRespuesta**: Es el nombre de nuestro servicio, el cual nosotros configuramos con *@WebParam()*.

Aquí es importante entender que este nombre es por el cual, desde internet, podemos acceder a nuestra clase. Y aquí viene lo interesante... luego del nombre de nuestro servicio vemos un símbolo de interrogación. Este símbolo nos indica que desde aquí irán los parámetros que vamos a enviar al servlet.

- **nombre=JUANITO&apellido=PEREZ**

Siempre que se envían parámetros por url (en este caso por get, más adelante veremos los verbos http) tenemos que indicar el nombre de la variable que va a contener el valor y su contenido: en este caso la variable es nombre y su contenido es JUANITO. En nuestro servlet nosotros manipulamos el objeto request con:

```
request.getParameter("nombre");
```

El valor que pasamos por parámetro al método getParameter es el nombre de la variable que estamos enviando por la url. Siempre que se envíe más de 1 parámetro, se separan los valores con el símbolo &.

Aplicar el envío de información entre Servlets

Si bien el flujo normal entre un cliente servidor es desde navegador a servidor, hay ocasiones en que se requiere que en la capa de negocio un servlet pueda comunicarse con otro servlet.

Para lograr este objetivo, java nos provee de los métodos:

```
    getRequestDispatcher();
    getServletContext();
```

- **getServletContext**: Método otorgado por HttpRequest, que lo que hace es obtener el contexto del servlet que queremos utilizar.

- **getServletDispatcher**: Luego de obtener el contexto, le enviamos la dirección del servlet.

Para entender mejor este mecanismo, vamos a modificar nuestro recién creado método *Saludo.java*

Las líneas sombreadas tienen la implementación. Si te fijas puedes ver que obtenemos el contexto del *servlet* y luego mediante *requestDispatcher* llamamos al *@WebServlet* de nombre *'inicio'* (es nuestro primer servlet de ejemplo).

```
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException, ServletException {
      PrintWriter printWriter = response.getWriter();
      SimpleDateFormat formato = new SimpleDateFormat("dd-MM-yyyy");
      printWriter.println("<html><body>");
      printWriter.println("Tu primer servlet");
      printWriter.println("La fecha actual es: " + fechaActual);
printWriter.println("</body></html>");

request.getServletContext().getRequestDispatcher("/inicio").forward(requ
est, response);
      }
    }
```

Si ejecutas la llamada:

```
http://localhost:8080/PrimerServlet/generadorRespuesta?nombre=JUANITO&apellido=PEREZ
```

Verás que se mostrará en el navegador el resultado del primer ejemplo de servlet que ejecutamos.

En la clase *GeneradorIndex.java,* agrega un mensaje representativo para que sepamos que estamos llamando al servlet. Como ejemplo:

```
    writer.println("Tu primer servlet.. me estan llamando desde un servlet?? ");
```

El resultado de la llamada es:

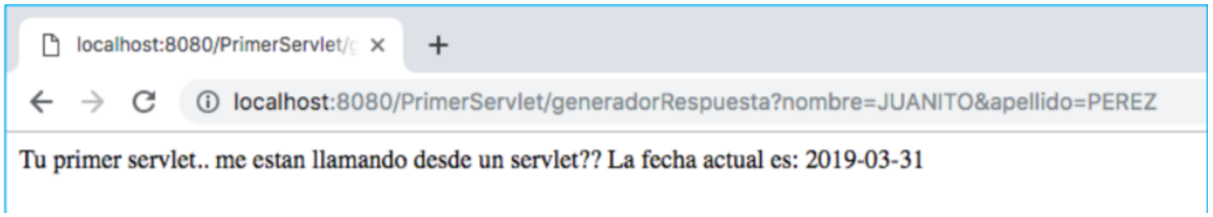

Imagen 3. Probando el Servlet.
Fuente: Desafío Latam