Práctica 8

Servlets

Programación Orientada a Objetos

Humberto Alejandro Ortega Alcocer

Opción: 2 “Triángulo”

19 de enero de 2021

Grupo: 2CM1

Introducción

En nuestro mundo globalizado el desarrollo de sistemas distribuidos corresponde a uno de los usos más generales en donde la mayoría de los programadores trabajan o estarán trabajando en un futuro próximo. Para realizar el desarrollo de estos sistemas se utilizan distintos *"stacks"*, o conjuntos de tecnologías, para programar los servidores, clientes y operaciones intermedias para satisfacer la demanda de servicios web. En el caso de Java, los servlets nos permiten una forma fácil y eficiente para desarrollar aplicaciones que empleen la web y ofrecer servicios que respondan a las necesidades de los usuarios usando una API simple y eficiente. En esta práctica, se pretende crear un sitio simple que dibuje un triángulo con una declaración estándar de HTML pero con un parámetro variable de renglones para el triángulo a dibujar en pantalla.

El planteamiento de la práctica, en palabras del profesor Roberto Tecla es:

Opción 2

Codificar un servlet que genere HTML para mostrar un **triangulo** de **asteriscos** de n renglones. Abajo se muestra el triangulo cuando n=5. El numero de renglones se introduce en un formulario (codifíquelo también).

<h1> Triangulo de 5 renglones de asteriscos</h1>

*
**
***
****
*****

Desarrollo

Lo primero que se debe entender para el desarrollo de la práctica es el cómo vamos a ejecutar nuestro código. Para la ejecución debemos usar un servidor, la documentación nos ofrece varias opciones como Tomcat o usando CGI con opciones como Nginx y Apache pero, y para simplificar la ejecución de la práctica (además de favorecer la portabilidad del código para su revisión) el profesor nos proporcionó un servidor de Winstone, que permite ejecutar el código del servlet de forma rápida y eficiente sin requerir la instalación de librerías o paqueterías externas o ajenas a nuestro sistema.

Antes de adentrarnos entonces a la descripción del servlet en si, me gustaría mostrar el proceso que se sigue para poder llevar a cabo la compilación y ejecución del código.

Si bien, los comandos para realizar las operaciones son muy simples, para realizar el desarrollo de la práctica y realizar cambios a los archivos, puede volverse muy tedioso dada la estructura de archivos que requiere Winstone. Es por eso que lo primero que hice fue escribir un pequeño script de bash para estas simples tareas.

El script ejecuta las siguientes tareas:

1. Compila el código del Triángulo en el directorio actual, utilizando el .jar con la definición de la librería para uso de servlets.
2. Copia tanto Triangulo.java como Triangulo.class a la carpeta destino dónde deberán colocarse estos archivos para que Winstone pueda encontrarlos y posteriormente servirlos.
3. Inicializa el servidor de Winstone en el puerto 5050.

Aunque este script es completamente opcional, me parece que fue uno de los aspectos más relevantes para realizar la práctica ya que sin éste uno puede perderse rápidamente sin saber si un error corresponde a Winstone en su ejecución, la ubicación de los archivos para su ejecución o simplemente algún error durante la compilación de nuestro servlet principal.
El script se llama “compilar-y-ejecutar.sh” y sus contenidos son:

```
#!/bin/bash

# En este archivo se incluyen los comandos para:
#   1) Compilar el archivo Triangulo.java
#   2) Mover el archivo resultante + su .class a la carpeta correspondiente.
#   3) Ejecutar el servidor de winstone

# 1) Compilar el archivo Triangulo.java
javac -cp .:./webapp/WEB-INF/lib/servlet-api.jar Triangulo.java

# 2) Copiar y mover los resultados a la carpeta destino para ejecución con winstone.
cp Triangulo.* ./webapp/WEB-INF/classes/

# 3) Ejecutar el servidor winstone en el puerto 5050.
java -jar winstone-0.9.10.jar --webroot=webapp --httpPort=5050

# Humberto Alejandro Ortega Alcocer
```

Como se puede observar, es muy simple, pero nos permite como programadores despreocuparnos de este procedimiento y enfocarnos en lo que es realmente importante: el servlet para mostrar el triángulo.
En cuanto al servlet principal, su contenido es muy simple ya que su operación, como fue definida por el planteamiento de la práctica, es también muy simple. Lo más importante es entender la clase Triángulo, cuya definición en UML se muestra a continuación:

Como se puede observar, la función más relevante es “doGet”, la cual será ejecutada cuando el servidor recibe un verbo GET desde el cliente que se conecte a la ruta correspondiente al archivo especificado. La definición de esta función, dónde se describe realmente el funcionamiento general del servlet es:

```
// GET /
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
  // Imprimimos el .html
  res.setContentType("text/html");
  PrintWriter out = res.getWriter();

  // Obtenemos el query parameter.
  try {
    String parametroStr = req.getParameter("renglones");
    noRenglones = Integer.parseInt(parametroStr);
  } catch (Exception e) {
    // Ignoramos la excepción, el default de renglones es 5.
    noRenglones = 5;
  }

  // Añadimos código HTML para la respuesta.
  out.println("<HTML>");
  out.println("<HEAD><TITLE>Triángulo</TITLE></HEAD>");
  out.println("<BODY>");
  out.println("<H1>Triángulo</H1>");

  // Realizamos la impresión del triángulo.
  for (int i = 0; i < noRenglones; i++) {
    for (int j = 0; j < i + 1; j++) {
      out.println("*");
    }
    out.println("<BR>");
  }

  // Cerramos HTML.
  out.println("<BR><BR><p>Humberto Alejandro Ortega Alcocer (2016630495)</p></BODY></HTML>");
}
```

La función es muy simple, sin embargo la parte de la obtención del parámetro para determinar el número de renglones a mostrar del triángulo tiene una consideración especial y es que, si bien el parámetro puede ser marcado como “requerido”, en la definición de HTTP los parámetros web se especifican como opcionales, por lo que puede, o no, existir, y en caso de que no exista, al tratar de obtenerlo y más adelante convertirlo en un entero, arrojará una excepción por lo que no se podrá visualizar el triángulo.

Para solventar este problema es que se coloca esa operación dentro de un bloque try/catch y se coloca, en caso de arrojarse la excepción, el número de renglones en cinco por defecto. De esta forma, en caso de que no se proporcione el parámetro, de igual forma se mostrará un triángulo de cinco renglones.

Finalmente, al ejecutar el script de compilación y ejecución, tendremos la siguiente salida en la terminal:

```
humbertowoody@mbahumbertowoody 10049 1.71G 2.65 ~/Proyectos/IPN/Materias/POO/Prácticas/practica-8 main
└ ./compilar-y-ejecutar.sh                                    17:05 19.01.21 100% 192.168.1.65
[Winstone 2021/01/19 17:05:52] - XML Error (Line 120): The content of element type "web-app" must match "(icon?,display-name?,description?,distributable?,context-param*,servlet
*,servlet-mapping*,session-config?,mime-mapping*,welcome-file-list?,error-page*,taglib*,resource-ref*,security-constraint*,login-config?,security-role*,env-entry*,ejb-ref*)".
[Winstone 2021/01/19 17:05:52] - Winstone Servlet Engine v0.9.10 running: controlPort=disabled
[Winstone 2021/01/19 17:05:52] - HTTP Listener started: port=5050
[Winstone 2021/01/19 17:05:52] - AJP13 Listener started: port=8009
```

Y la visualización en la página web, accediendo a la liga http://localhost:5050/servlets/servlet/Triangulo?renglones=12 el sitio mostrará:

Conclusión

Es importante que, como futuros desarrolladores, tengamos siempre presente las utilidades y servicios que podemos emplear para llevar a cabo los proyectos con los que nos encontraremos en el mundo laboral. Si bien un servlet no representa, bajo ninguna métrica, un marco de trabajo pensado para cargas enormes o para un uso práctico en producción, en esta práctica se lograron visualizar los distintos requisitos y complejidades que pueden surgir al trabajar en un ambiente distribuido en web.

Para poder llevar a cabo la práctica se tuvo que realizar una investigación exhaustiva referente a los distintos esquemas de compilación y ejecución para servlets, lo cual me permitió visualizar una parte de Java que no conocía en cuanto al ligado de dependencias y modelos de ejecución dinámicos que existen para este tipo de proyectos.