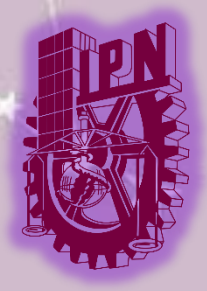


Instituto Politecnico Nacional

Escuela Superior de Cómputo

Bocanegra Heziquio Yestlanezi

Practica 8

Servlets

Opción 2

Fecha: 15 de junio de 2021

2CM13

Programación Orientada a Objetos

Contenido

INTRODUCCION

Servlet

Concepto de servlet

Un **servlet** es un programa Java que se ejecuta en un servidor Web y construye o sirve páginas web. De esta forma se pueden construir páginas dinámicas, basadas en diferentes fuentes variables: datos proporcionados por el usuario, fuentes de información variable (páginas de noticias, por ejemplo), o programas que extraigan información de bases de datos.

Comparado con un CGI, un servlet es más sencillo de utilizar, más eficiente (se arranca un hilo por cada petición y no un proceso entero), más potente y portable. Con los servlets podremos, entre otras cosas, procesar, sincronizar y coordinar múltiples peticiones de clientes, reenviar peticiones a otros servlets o a otros servidores, etc.

Recursos de servlet

Normalmente al hablar de servlets se habla de JSP y viceversa, puesto que ambos conceptos están muy interrelacionados. Para trabajar con ellos se necesitan tener presentes algunos recursos:

- Un **servidor web** que dé soporte a servlets / JSP (contenedor de servlets y páginas JSP). Ejemplos de estos servidores son Apache Tomcat, Resin, JRun, Java Web Server, BEA WebLogic, etc.
- Las **librerías** (clases) necesarias para trabajar con servlets / JSP. Normalmente vienen en ficheros JAR en un directorio *lib* (*common/lib* en Tomcat): *servlets.jar* (con la API de servlets), y *jsp.jar*, *jspengine.jar* o *jasper.jar* para JSP. Al desarrollar nuestra aplicación, deberemos incluir las librerías necesarias en el classpath para que compilen los ficheros. También se puede utilizar el fichero JAR *j2ee.jar* que viene con Java Enterprise Edition, pero no es recomendable si se puede disponer de las librerías específicas del servidor.
- La **documentación** sobre la API de servlets / JSP (no necesaria, pero sí recomendable)

Para encontrar información sobre servlets y JSP, son de utilidad las siguientes direcciones:

- http://java.sun.com/j2ee/: referencia de todos los elementos que componen J2EE
- http://java.sun.com/products/jsp: referencia para las últimas actualizaciones en JSP
- http://java.sun.com/products/servlets: referencia para las últimas actualizaciones en servlets

Arquitectura del paquete servlet

Dentro del paquete **javax.servlet** tenemos toda la infraestructura para poder trabajar con servlets. El elemento central es la interfaz **Servlet**, que define los métodos para cualquier servlet. La clase **GenericServlet** es una clase abstracta que implementa dicha interfaz para un servlet genérico, independiente del protocolo. Para definir un servlet que se utilice vía web, se tiene la clase **HttpServlet** dentro del subpaquete **javax.servlet.http**. Esta clase hereda de *GenericServlet*, y también es una clase abstracta, de la que heredaremos para construir los servlets para nuestras aplicaciones web.

Cuando un servlet acepta una petición de un cliente, se reciben dos objetos:

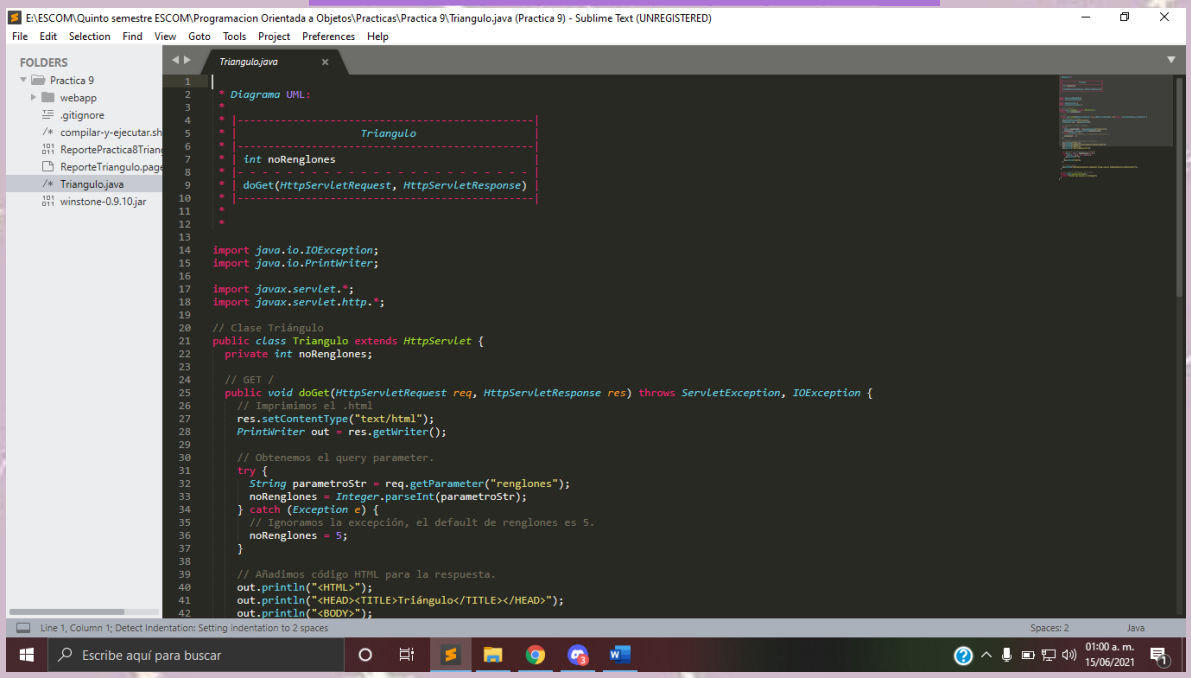
- Un objeto de tipo **ServletRequest** que contiene los datos de la petición del usuario (toda la información entrante). Con esto se accede a los parámetros pasados por el cliente, el protocolo empleado, etc. Se puede obtener también un objeto **ServletInputStream** para obtener datos del cliente que realiza la petición. La subclase **HttpServletRequest** procesa peticiones de tipo HTTP.
- Un objeto de tipo **ServletResponse** que contiene (o contendrá) la respuesta del servlet ante la petición (toda la información saliente). Se puede obtener un objeto **ServletOutputStream**, y un *Writer*, para poder escribir la respuesta. La clase **HttpServletResponse** se emplea para respuestas a peticiones HTTP.

Ciclo de vida de un servlet

Todos los servlets tienen el mismo ciclo de vida:

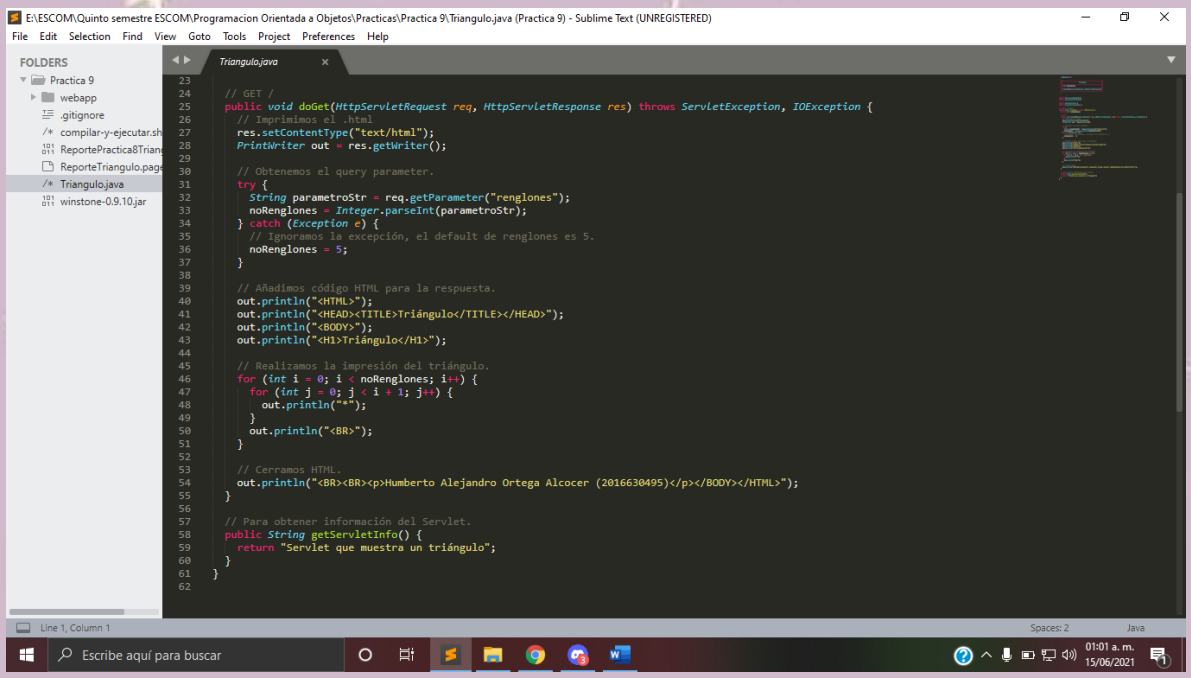
- Un servidor carga e inicializa el servlet
- El servlet procesa cero o más peticiones de clientes (por cada petición se lanza un hilo)
- El servidor destruye el servlet (en un momento dado o cuando se apaga)

DESARROLLO



```
/*
 * Diagrama UML:
 *
 * |-----------------------------------------------|
 * |                   Triangulo                   |
 * |-----------------------------------------------|
 * | int noRenglones                               |
 * |- - - - - - - - - - - - - - - - - - - - - - - -|
 * | doGet(HttpServletRequest, HttpServletResponse)|
 * |-----------------------------------------------|
 *
 *
 */

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.*;
import javax.servlet.http.*;

// Clase Triángulo
public class Triangulo extends HttpServlet {
  private int noRenglones;

  // 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>");
```

Código 1 Parte 1



```
    // 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>");
    }

    // Para obtener información del Servlet.
    public String getServletInfo() {
      return "Servlet que muestra un triángulo";
    }
}
```

Código 2 Parte 2

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.