Servlet & JSP

Bryan Ivan Montiel Ortega

Noviembre 2023

Índice

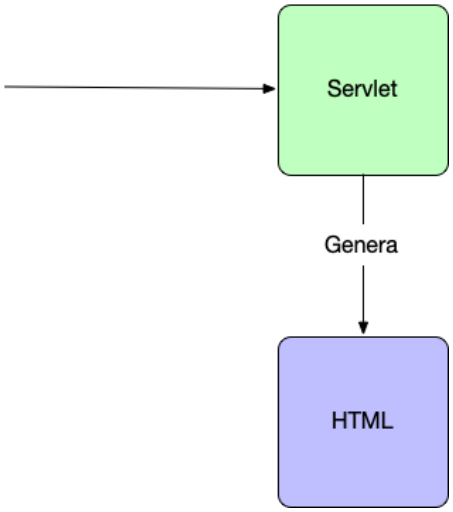
¿Qué es un Servlet?

Es una tecnología que nos permite crear aplicaciones web interactivas (dinámicas), es decir, le permite al usuario interactuar con la aplicación (hacer consultas, insertar y eliminar datos, …)

Un Servlet es un objeto java que pertenece a una clase que extiende de javax.servlet.http.HttpServlet

Son pequeños programas escritos en Java que admiten peticiones a través del protocolo HTTP. Los servlets reciben peticiones desde un navegador web, las procesan y devuelven una respuesta al navegador, normalmente en HTML. Para realizar estas tareas podrán utilizar las clases incluidas en el lenguaje Java. Estos programas son los intermediarios entre el cliente (casi siempre navegador web) y los datos (BBDD)

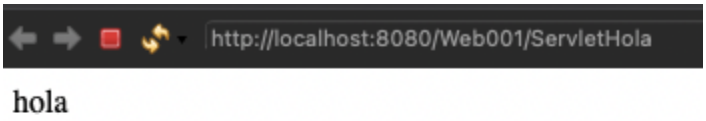
Para empezar hay que entender que un Servlets es el concepto más básico de componente web a nivel de Java EE. Se trata de una clase que genera una página HTML.




```java
package com.arquitecturajava;

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

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class ServletHola
 */
@WebServlet("/ServletHola")
public class ServletHola extends HttpServlet {
  private static final long serialVersionUID = 1L;
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

      PrintWriter writer = response.getWriter();
      writer.println("<html>");
      writer.println("<body>");
      writer.println("hola");
      writer.println("</body>");
      writer.println("</html>");
    }
}
```

En este caso estamos ante un Servlet que mapea la url de /ServletHola y que genera una pagina HTML de bienvenida sin mas .Para ello usa el método doGet que se encarga de procesar la petición GET.



No hay más que verlo para darnos cuenta que la tecnología es bastante arcaica. Bueno las cosas no terminan así primero porque hay que darse cuenta de todas las aplicaciones legacy que incluyen Servlets y por lo tanto no está mal saber cómo estos funcionan cuando nos tenemos que encontrar con código antiguo. Sin embargo el tema es más profundo.

¿Cómo funciona un contenedor de Servlets?

- El navegador (cliente) pide una página al servidor HTTP que es un contenedor de Servlets.
- El servlet procesa los argumentos de la petición, es decir, el contenedor de Servlets delega la petición a un Servlet en particular elegido de entre los Servlets que contiene.
- El Servlet, que es una objeto java, se encarga de generar el texto de la página web que se entrega al contenedor.
- El contenedor devuelve la página web al navegador (cliente) que la solicitó, normalmente en HTML.

Por lo tanto nos encontramos en una arquitectura Cliente-Servidor. Lo normal para esto es utilizar Apache Tomcat como contenedor de servlets. Recordar que apache es un servidor HTTP.

Ciclo de vida de un Servlet

4.1. init()

El método init está diseñado para ser llamado solo una vez. Si no existe una instancia del servlet, el contenedor web:

1. Carga la clase de servlet
2. Crea una instancia de la clase de servlet
3. Lo inicializa llamando al método init

El método init debe completarse correctamente antes de que el servlet pueda recibir solicitudes. El contenedor de servlets no puede poner el servlet en servicio si el método init produce una excepción ServletException o no devuelve dentro de un período de tiempo definido por el servidor web.

```
public void init() throws ServletException {
    // Initialization code like set up database etc....
}
```

4.2. servicio()

Este método solo se llama después de que el método init() del servlet se haya completado correctamente.

El contenedor llama al método service() para controlar las solicitudes procedentes del cliente, interpreta el tipo de solicitud HTTP (GET, POST, PUT, DELETE, etc.) y llama a los métodos doGet, doPost, doPut, doDelete, etc. según corresponda.

```
public void service(ServletRequest request, ServletResponse response)
  throws ServletException, IOException {
    // ...
}
```

4.3. destruir()

Llamado por el contenedor de servlets para sacar el servlet de servicio.

Este método sólo se llama una vez que se han cerrado todas las hebras del método de servicio del servlet o después de que haya transcurrido un período de tiempo de espera. Después de que el contenedor llame a este método, no volverá a llamar al método de servicio en el servlet.

```
public void destroy() {
    // }
```

JSP

La tecnología Java para la creación de páginas web con programación en el servidor.

JSP es un acrónimo de Java Server Pages, que en castellano vendría a decir algo como Páginas de Servidor Java. Es, pues, una tecnología orientada a crear páginas web con programación en Java.

Con JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Por tanto, las JSP podremos escribirlas con nuestro editor HTML/XML habitual.

Características

- El Servidor Java de Páginas (Java Server Pages, JSP) ofrece no sólo la independencia de operar en diferentes plataformas y servidores de páginas Web, sino que además combina el poder de la tecnología Java en el servidor con la facilidad de visualizar el contenido de las páginas HTML.
- JSP es una tecnología que generalmente incluye:
    - Componentes estáticos HTML/XML
    - Elementos JSP especiales
    - Fragmentos especiales de código escritos en lenguaje Java llamados scriptlets
- La especificación JSP extiende la tecnología de servlets para reducir la programación requerida en el desarrollo de páginas Web dinámicas.
- JSP permite apreciar mejor la distinción entre el contenido de la información y su presentación

Ventajas

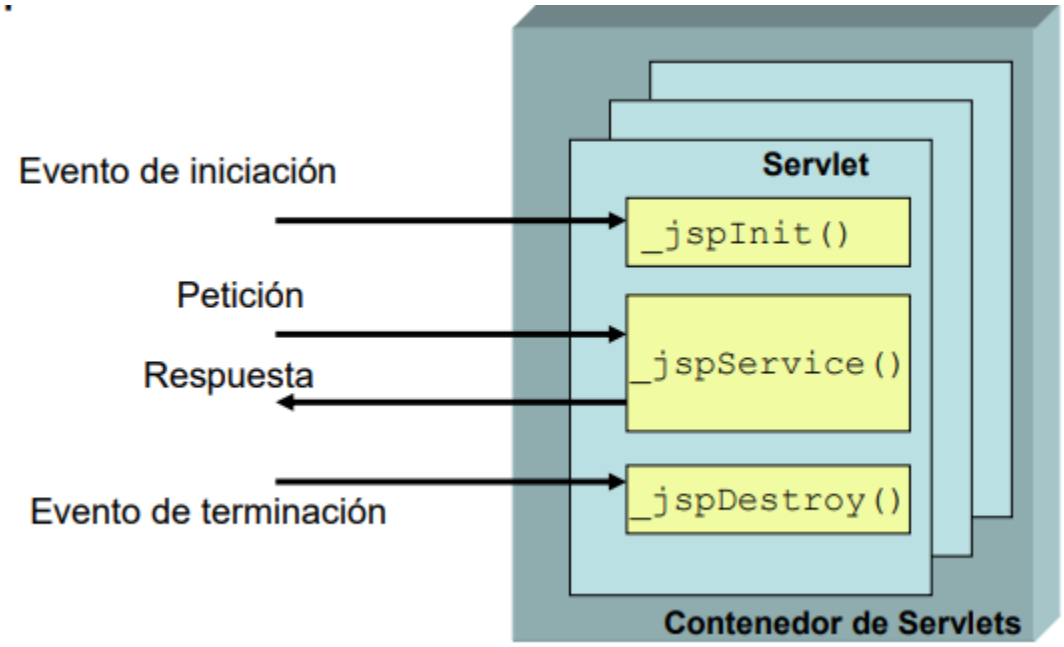
- Independencia del servidor Web.
- Substitución del API de los servlets por conjuntos de elementos de marcado (tags) y fragmentos de programación (scriptlets).
- Altamente recomendable para conseguir el acceso al servidor Web en arquitecturas de partes múltiples (multi-tier).
- Separación del contenido estático del dinámico.
- Separación del contenido dinámico del formato de la presentación

Arquitectura de JSP

- El propósito de JSP es ofrecer una forma más declarativa y más guiada por la presentación que aquella conseguida con los servlets.
- Las páginas JSP se utilizan en un proceso que involucra dos fases:
    a. 1. Una fase de traducción dinámica a servlets que se realiza únicamente cuando la página JSP se crea o modifica.

b. 2. Una fase de contestación a peticiones que se consigue cuando las clases de los servlets generados se cargan en el contenedor de servlets.

- En la segunda fase, para atender concurrentemente al procesamiento de peticiones, el servidor lanza un proceso ligero (`thread`) por cada uno de los clientes el cual corre el método `_jspService()`.
- El método `_jspService()` no puede reemplazarse por otros métodos definidos por el programador, pero en cambio puede adecuarse su inicialización y terminación con los métodos `jspInit()` y `jspDestroy()`



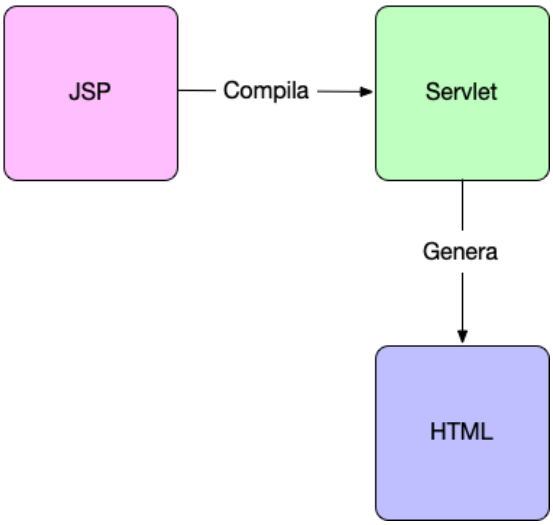
Motor JSP

El motor de las páginas JSP está basado en los servlets de Java -programas en Java destinados a ejecutarse en el servidor-, aunque el número de desarrolladores que pueden afrontar la programación de JSP es mucho mayor, dado que resulta mucho más sencillo aprender que los servlets.

En JSP creamos páginas de manera parecida a como se crean en ASP o PHP -otras dos tecnologías de servidor-. Generamos archivos con extensión .jsp que incluyen, dentro de la estructura de etiquetas HTML, las sentencias Java a ejecutar en el servidor. Antes de que sean funcionales los archivos, el motor JSP lleva a cabo una fase de traducción de esa página en un servlet, implementado en un archivo class (Byte codes de Java). Esta fase de traducción se lleva a cabo habitualmente cuando se recibe la primera solicitud de la página .jsp, aunque existe la opción de precompilar en código para evitar ese tiempo de espera la primera vez que un cliente solicita la página.

JSP y Servlet

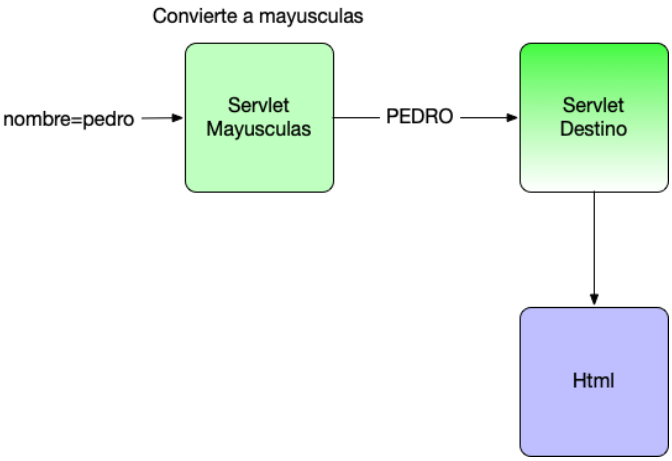
Cuando cualquier persona construye una página JSP esta página realmente no existe a nivel de Java ya que Java lo único que entiende son clases. Por lo tanto el fichero JSP ha de convertirse por medio de un compilador en un Servlets.



Así pues cualquier fichero JSP que hayas creado en algún momento es un Servlet. Esto hace que la tecnología hoy por hoy siga estando muy viva. Aún así muchas personas piensan que los Servlets son únicamente utilizados en aplicaciones antiguas con JSP . Las cosas no son como parecen ya que los Servlets tienen un diseño muy flexible a nivel de Arquitectura y permiten pasar información entre ellos de una forma transparente con un modelo de delegación.

Servlet y Despachadores

Por ejemplo nosotros podríamos tener un Servlet que reciba una variable nombre , convierta esta variable a mayúsculas y delegue en otro Servlets que sea el encargado de mostrar la información.



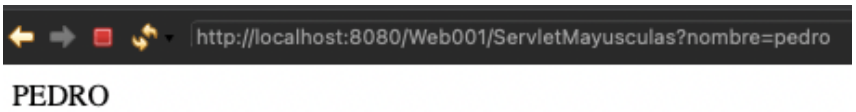
```java
package com.arquitecturajava;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class ServletMayusculas
 */
@WebServlet("/ServletMayusculas")
public class ServletMayusculas extends HttpServlet {
  private static final long serialVersionUID = 1L;

  protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

  String nombre=request.getParameter("nombre");
  request.setAttribute("nombre", nombre.toUpperCase());
  RequestDispatcher despachador= request.getRequestDispatcher("/ServletDestino");
  despachador.forward(request, response);
  }
}
```

En este caso hemos diseñado un Servlet que recibe un nombre y lo pasa a mayúsculas. Este Servlet no genera contenido sino que usa el objeto `'Request'` para pasar datos a otro `'Servlet'` que será el encargado de procesarlo e imprimir la información . Este Servlet se denomina `'ServletDestino'`.

```java
package com.arquitecturajava;

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

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
```

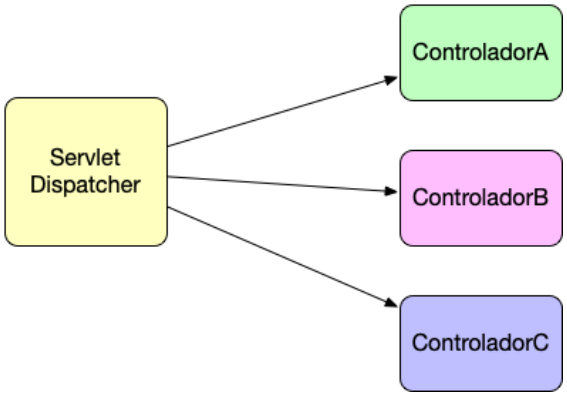
```
 * Servlet implementation class ServletDestino
 */
@WebServlet("/ServletDestino")
public class ServletDestino extends HttpServlet {
  private static final long serialVersionUID = 1L;


  protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    // TODO Auto-generated method stub
    PrintWriter writer= response.getWriter();
    writer.println("<html>");
    writer.println("<body>");
    writer.println(request.getAttribute("nombre"));
    writer.println("</body>");
    writer.println("</html>");
  }
}
```

Este último Servlet es el encargado de mostrar la información . Si nosotros invocamos la url de /ServletMayusculas , el primer Servlet pasara los datos a mayusculas y el segundo imprimirá los datos de forma que se reparten las responsabilidades de forma transparente.



FrontController y Servlets

Puede parecer poco importante este sistema de delegación pero es la base de todos los frameworks modernos web de Java ya que por ejemplo Spring MVC usa un Servlet Principal o FrontController para recibir todas las peticiones web y redirigirlas a cada una de los controladores que tiene .Este Servlet se denomina DispacherServlet a nivel de Spring y puede que en algún momento nos veamos en la obligación de tocar su configuración de alguna forma.



De igual forma funciona JSF o JAX-RS suelen disponer de Servlets que disponen de un mapeo /* o similar y capturan todas las peticiones web para luego redirigirlas a los componentes adecuados. Por lo tanto aunque nos parezca que los Servlets son tecnologías muy antiguas hoy por hoy siguen estando muy vivas. De hecho en los frameworks más modernos son el punto de entrada de todas las peticiones , eso sí su rol esta más ligado a ser el corazón de la arquitectura a ser un componente que genere HTML.

Bibliografía

1. baeldung. (2022, agosto 22). Introduction to Java Servlets. Baeldung.com. https://www.baeldung.com/intro-to-servlets
2. Caules, C. Á. (2021, diciembre 17). ¿Qué es un Servlet? Arquitectura Java. https://www.arquitecturajava.com/que-es-un-servlet/
3. desarroladorweb. (2002, julio 8). Qué es JSP. Desarrolloweb.com. https://desarrolloweb.com/articulos/831.php
4. Que es un servlet y su función al momento de crear un pagina. (s/f). Codea.app. Recuperado el 15 de noviembre de 2023, de https://codea.app/blog/que-es-un-servlet-en-java
5. Sosa, V. J. (s/f). Java Server Pages (JSP). Cinvestav.mx. Recuperado el 15 de noviembre de 2023, de https://www.tamps.cinvestav.mx/~vjsosa/clases/sd/DAAI_JSP.pdf