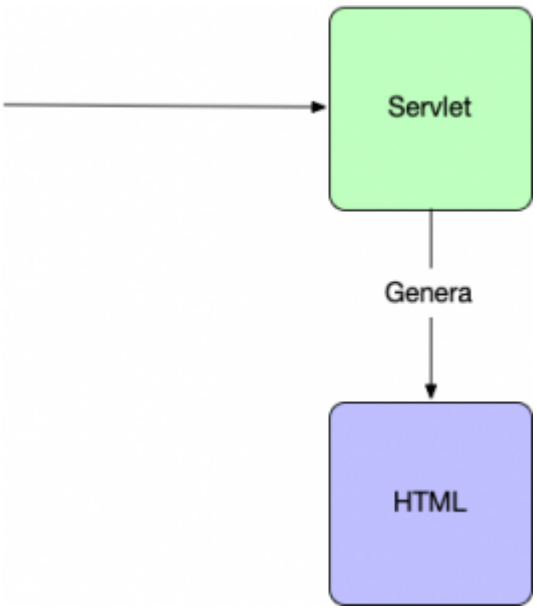
Tabla de Contenidos

- JSP y Servlet
- Servlet y Despachadores
- FrontController y Servlets
- Otros artículos relacionados

Acabamos de ver como usar Java equals y hashcode

¿Qué es un Servlet? . Esta pregunta me la siguen haciendo bastantes personas con los desarrollos modernos . ¿Ya no se usan los Servlets , han pasado a la historia? . ¿Son tecnologías antiguas? . Vamos a verlo todo un poco a detalle . 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.



Algo en principio muy muy sencillo :

```
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 {

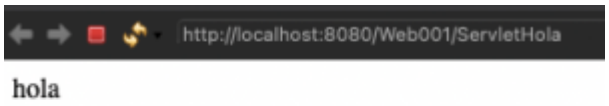
    PrintWriter writer = response.getWriter();
    writer.println("<html>");
    writer.println("<body>");
    writer.println("hola");
    writer.println("</body>");
    writer.println("</html>");

  }

}
```

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

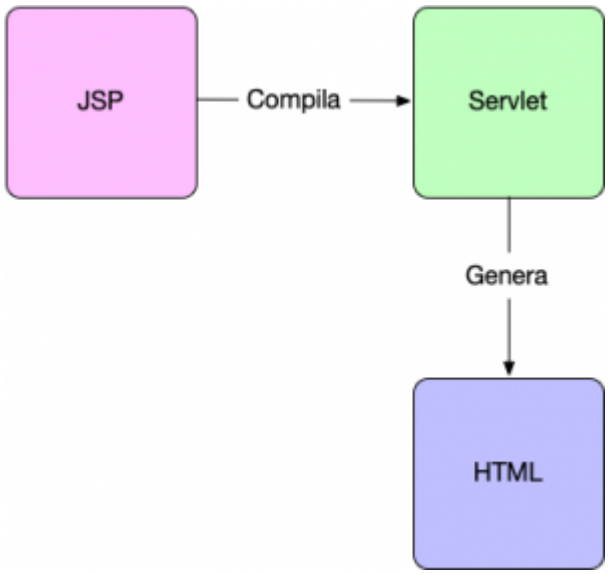
procesar la petición GET.



No hay mas que verlo para darnos cuenta que la tecnología es bastante arcaica . ¿Entonces los Servlets han pasado a mejor vida y ya no se usan? . 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 esta mal saber cómo estos funcionan cuando nos tenemos que encontrar con código antiguo. Sin embargo el tema es más profundo.

JSP y Servlet

Cuando cualquier persona construye una página JSP esta pagina 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.

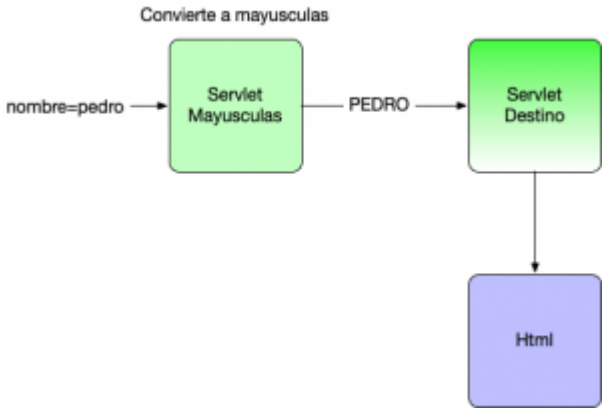


Así pues cualquier fichero JSP que hayas creado en algún momento es un Servlet. Esto hace que la tecnología hoy por hoy siga estando muy viva. Aún así muchas personas piensan que

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 mayusculas y delegue en otro Servlets que sea el encargado de mostrar la información.



Veamoslo en código:

```
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 mayusculas. 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.

```
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>");
  }


}
```

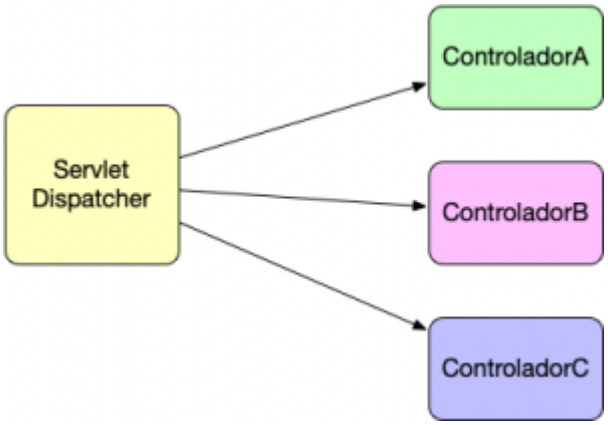
Este último Servlet es el encargado de mostrar la información . Si nosotros invocamos la url de /ServletMayusculas , el primer Servlet pasara los datos a mayusculas y el segundo imprimirá los datos de forma que se reparten las responsabilidades de forma transparente.



Acabamos de dividir las responsabilidades a nivel de Servlets

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.

Otros artículos relacionados

- Servlets 3.0 (I)
- Servlet 3.0 (II) Servlets Asincronos
- Spring Boot JSP y su configuración
- Entendiendo el Servlet Lifecycle en Java EE