Práctica 2
Filtros

DOMÍNGUEZ ESPINOZA JUANA DEL ROSARIO
Escuela Superior de Cómputo
2 de octubre de 2018

Introducción

Un filtro intercepta dinámicamente solicitudes y respuestas para transformar o usar la información contenida en las solicitudes o respuestas. Por lo general, los filtros no crean respuestas, sino que proporcionan funciones universales que pueden .ªdjuntarse.ª cualquier tipo de servlet o página JSP. Los filtros son importantes por una serie de razones. Primero, brindan la capacidad de encapsular tareas recurrentes en unidades reutilizables.

En segundo lugar, los filtros se pueden usar para transformar la respuesta de un servlet o una página JSP. Una tarea común para la aplicación web es formatear los datos enviados al cliente. Cada vez más los clientes requieren formatos (por ejemplo, WML) que no sean solo HTML. Para acomodar a estos clientes, generalmente hay un fuerte componente de transformación o filtrado en una aplicación web con todas las características. Muchos contenedores servlet y JSP han introducido mecanismos de filtro patentados, lo que resulta en una ganancia para el desarrollador que se implementa en ese contenedor, pero que reduce la reutilización de dicho código.

Los filtros pueden realizar diferentes funciones, por ejemplo:

- Autenticación: solicitudes de bloqueo basadas en la identidad del usuario.
- Registro y auditoría: seguimiento de usuarios de una aplicación web.
- Conversión de imagen: escala de mapas, etc.
- Compresión de datos: reducir las descargas.
- Localización: segmentación de la solicitud y respuesta a una configuración en particular.
- Transformaciones XSL/T de contenido XML: segmentación de respuestas de aplicaciones web a más de un tipo de cliente.

El método más importante en la interfaz de filter es el método doFilter, que es el corazón del filtro. Además de doFilter, existen los métodos init y destroy. El contenedor llama al método init cuando se crea una instancia del filtro. Si desea pasar los parámetros de inicialización al filtro, los recupera del objeto FilterConfig pasado a init. El metoo destroy es para destruir el filtro.

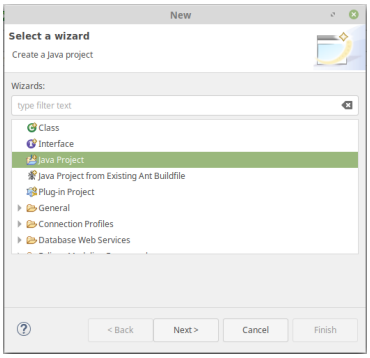
Desarrollo

Implementar un filtro que intercepte cualquier petición que se haga a la aplicación web e imprima en log, con el siguiente formato.

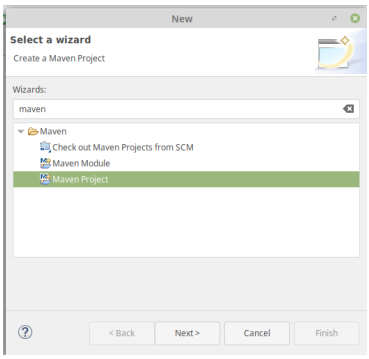
```
Filter: IP fecha y hora método recurso
```

La práctica se va a realizar en el IDE eclipse.

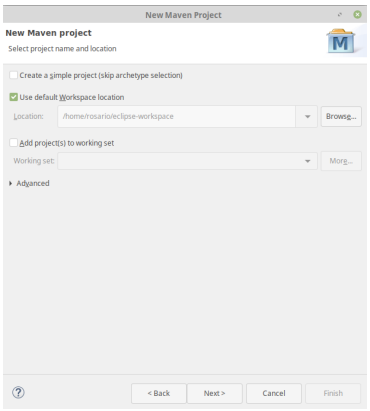
1. Para crear un nuevo proyecto seleccionamos el menu File->New->Other, se desplegara la siguiente ventana.



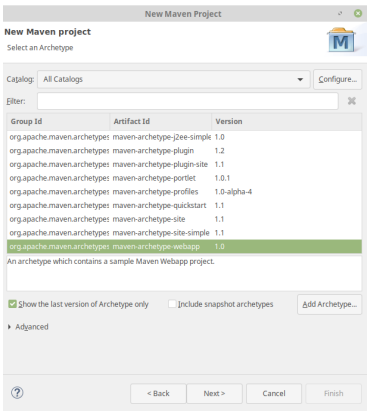
En el cuadro de texto escribimos maven y seleccionamos la opción "Maven Project".



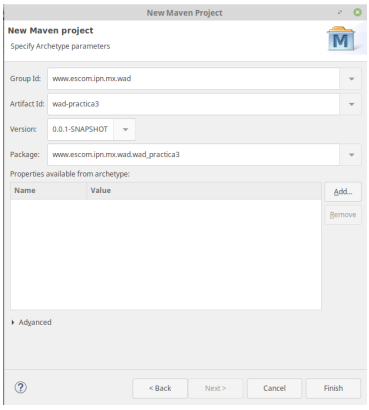
Seleccionamos el botón Next y se mostrara la siguiente ventana.



Seleccionamos botón Next, en la ventana que se mostrara seleccionamo la ultima opcion (opcion seleccionada en la imagen).



La ventana que se mostrara debera tener los valores siguientes. Para terminar de crear nuestro proyecto seleccionamos el botón Finish.



2. Para agregar la libreria javax servlet, realizamos la siguiente busqueda en el navegador.

Selecionamos la primera opción y copiamos lo siguiente.

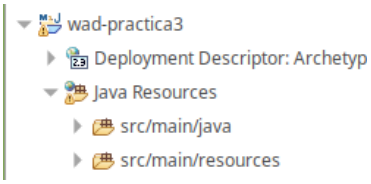
Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

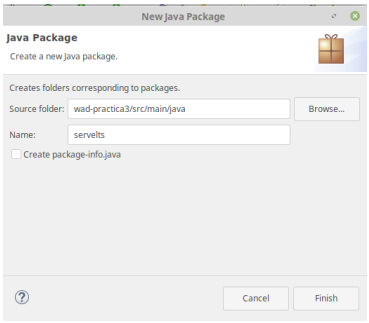
```
<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
```

Buscamos en el proyecto el archivo pom.xml y pegamos el contenido que se copio.

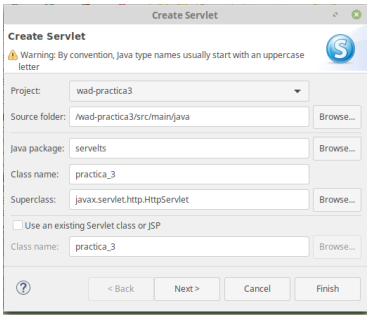
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>www.escom.ipn.mx.wad</groupId>
  <artifactId>wad-practica3</artifactId>
  <packaging>war</packaging>
  <version>0.0.1-SNAPSHOT</version>
  <name>wad-practica3 Maven Webapp</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
      <scope>provided</scope>
    </dependency>

  </dependencies>
  <build>
    <finalName>wad-practica3</finalName>
  </build>
</project>
```

3. Para crear el servlet a utilizar, con click derecho en la carpeta src/main/java. New->Package y se mostrara la siguiente ventana.

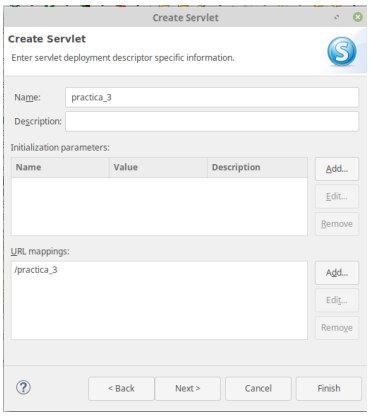




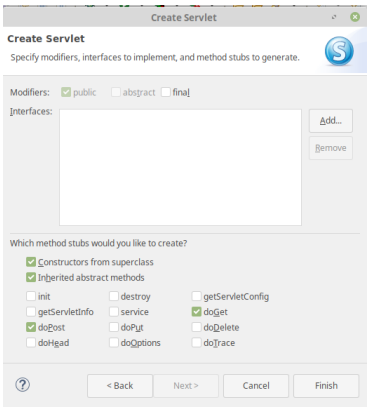
Seleccionamos con click derecho el paquete creado, New->Servlet, se mostrara la siguiente ventana.



Escribimos el nombre del servlet, en este caso practica_3 y seleccionamos el botón Next. Se mostrara la siguiente ventana.



Seleccionamos el botón Next. Se mostrara la siguiente ventana.



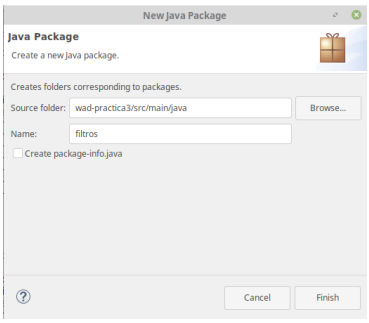
Seleccionamos el botón Finish.

Nuestro servlet practica_3.java deberá tener el siguiente codigo en sus metodos doGet y doPost.

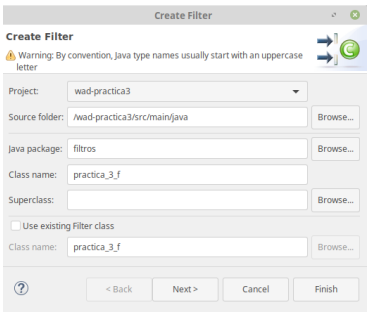
```
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
     // TODO Auto-generated method stub
     PrintWriter out = response.getWriter();
     response.setContentType("text/html");
     out.println("<html>");
     out.println("<head>");
     out.println("</head>");
     out.println("<body>");
     out.println("<h1>Holi </h1>");
     out.println("</body>");
     out.println("</html>");

 }
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    response.setContentType("text/html");
    out.println("<html>");
    out.println("<head>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Ciclo de vida del servlet </h1>");
    out.println("</body>");
    out.println("</html>");
    doGet(request, response);
}
```

Es importate señalar que el contenido de nuestro servlet no debe de ser tan complejo, al menos en esta práctica.

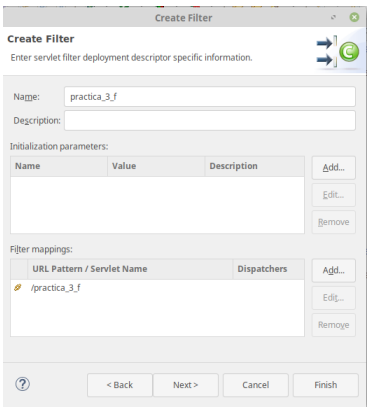
4. Para crear nuestro filtro. Primero debemo crear el paquete filtros en /src/main/java, como se muestra en lo siguiente.



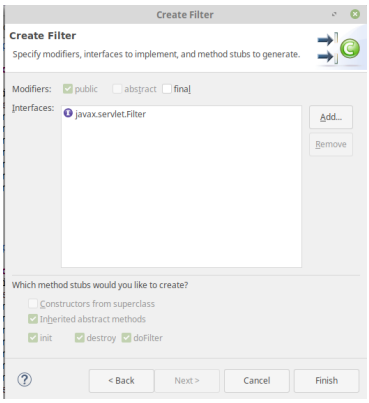
Depués seleccionamos el paquete filtros, New->Filter, y se mostrara la siguiente ventana. El nombre de nuestro filtro será practica_3_f.



Seleccionamos el botón Next y aparecera la siguiente ventana.



Selecionamos el botón Next y apareccera la siguiente ventana.



Seleccionamos el botón Finish para finalizar la creación del filtro.

El código de nuestro filtro seŕa el siguiente

```
public void destroy() {
    System.out.println("Fin del Filtro");
}
 public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException,
     ServletException
 {
     HttpServletRequest req = (HttpServletRequest) request;
     Calendar Calendario = Calendar.getInstance();
     int hora,min, seg;
     int dia,mes,año;
     dia=Calendario.get(Calendar.DATE);
     mes=Calendario.get(Calendar.MONTH);
     año=Calendario.get(Calendar.YEAR);
     hora= Calendario.get(Calendar.HOUR_OF_DAY);
     min= Calendario.get(Calendar.MINUTE);
     seg=Calendario.get(Calendar.SECOND);

     System.out.println("Filter "  + req.getRemoteAddr()+ "  "+dia+"-"+mes
                                +"-"+año +"  "+hora+":"+min+":"+seg +"  "
                                + req.getMethod()+"  "+ req.getRequestURI());
     chain.doFilter(request, response);
 }
public void init(FilterConfig fConfig) throws ServletException {

    System.out.println("Inicio del Filtro");
}
```

5. Cuando se crean los filtros y los servlets, el archivo web.xml agrega de automaticamente los archivos creados. Sin embargo en la etiqueta <url-pattern>del filtro borramos todo y agregamos un /*, esto para que filtre todos los archivos que se utilizaran en la aplicación, el archivo web.xml debe de quedar de la siguiente forma.

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <filter>
    <filter-name>practica_3_f</filter-name>
    <display-name>practica_3_f</display-name>
    <description></description>
    <filter-class>filtros.practica_3_f</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>practica_3_f</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <servlet>
    <servlet-name>practica_3</servlet-name>
    <display-name>practica_3</display-name>
    <description></description>
    <servlet-class>servelts.practica_3</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>practica_3</servlet-name>
    <url-pattern>/practica_3</url-pattern>
  </servlet-mapping>
</web-app>
```

6. El archivo index.jsp debe de quedar de la siguiente forma.

```
<html>
<head>
 <title>Web Application Development: Practica 3</title>
</head>
<body>
    <h1>Filtros </h1>
    <ol>
        <li><a href="practica_3">Filtro:</a>filtro ip</li>

    </ol>
</body>
</html>
```

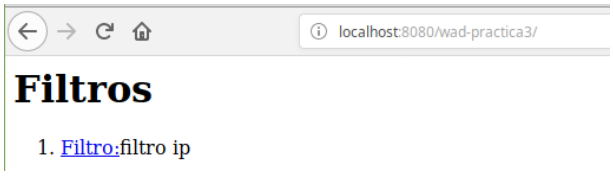
7. Para poder visualizar nuestra aplicación, seleccionamos el proyecto con click derecho, Run As->Run Jetty.

   En la consola aparecerá lo siguiente

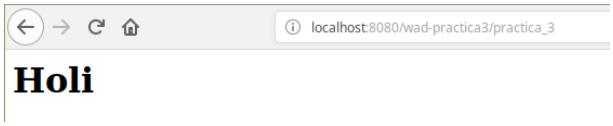
```
wad-practica3 [Jetty Webapp] /usr/local/java/jdk1.8.0_181/bin/java (13/09/2018 03:25:32)
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0
Running Jetty 6.1.26
2018-09-13 03:25:33.233:INFO::Logging to STDERR via org.mortbay.log.StdErrLog
ParentLoaderPriority enabled
Context path:/wad-practica3
ProjectClassLoader: entry=/home/rosario/eclipse-workspace/wad-practica3/target/classes
ProjectClassLoader: entry=/home/rosario/.m2/repository/junit/junit/3.8.1/junit-3.8.1.jar
ProjectClassLoader: entry=/home/rosario/.m2/repository/javax/servlet/javax.servlet-api/3.1.0/javax.servlet-api-3.1.0.jar
Excluded entry=/home/rosario/eclipse-workspace/wad-practica3/target/test-classes
2018-09-13 03:25:33.328:INFO::jetty-6.1.26
Inicio del Filtro
2018-09-13 03:25:34.058:INFO::Started SelectChannelConnector@0.0.0.0:8080
```

Pruebas

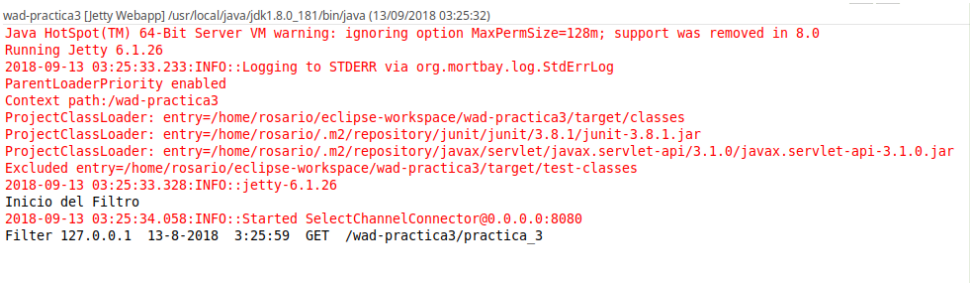
1. Abrimos una nueva pestaña en nuestro navegador con la siguiente URL. Contemplando que nuestro IDE esta corriendo la aplicación.



2. Seleccionamos el link y nos aparecera lo siguiente.



3. Regresamos al IDE y en la consola se observará lo siguiente.



Conclusiones

Los filtros son importantes ya que permiten las restricciones del proyecto, de acuerdo a las tareas a realizar o a los usuarios que vayan a acceder al sistema.

Referencias

- https://www.oracle.com/technetwork/java/filters-137243.html
- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api/3.1.0