# Filter(过滤器)

## Filter简介

* Filter中文意思为过滤器。顾名思义，过滤器可在浏览器以及目标资源之间起到一个过滤的作用。例如：水净化器，可以看成是生活中的一个过滤器，他可以将污水中的杂质过滤，从而使进入的污水变成净水。
* 对于WEB应用来说，过滤器是一个驻留在服务器中的WEB组件，他可以截取客户端和WEB资源之间的请求和响应信息。
* WEB资源可能包括Servlet、JSP、HTML页面等

服务器

请求

WEB资源

过滤器

浏览器

响应

* 当服务器收到特定的请求后，会先将请求交给过滤器，程序员可以在过滤器中对请求信息进行读取修改等操作，然后将请求信息再发送给目标资源。目标资源作出响应后，服务器会再次将响应转交给过滤器，在过滤器中同样可以对响应信息做一些操作，然后再将响应发送给浏览器。
* 也就是说过滤器可以在WEB资源收到请求之前，浏览器收到响应之前，对请求和响应信息做一些相应的操作。
* 在一个WEB应用中可以部署多个过滤器，多个过滤器就组成了一个过滤器链，请求和响应必须在经过多个过滤器后才能到达目标；

服务器

过滤器

过滤器

请求

WEB资源

过滤器

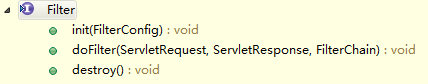
浏览器

响应

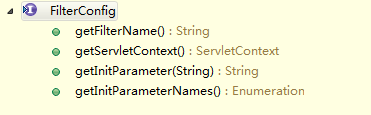
* 过滤器不是必须将请求传送到下一个过滤器（或WEB资源），也可以自己来处理请求，发送响应。

# Filter相关接口

* 编写Filter和编写Servlet类似，都需要实现接口。
* 编写Filter需要实现Filter接口，我们来看一下Filter接口的主要方法：



* Filter接口，实现Filter需要实现该接口
  1. init()方法用于初始化Filter
  2. doFilter()作用和service()方法类似，是过滤请求和响应的主要方法。
  3. destroy()用于在Filter对象被销毁前做一些收尾工作。如：释放资源等。



* FilterConfig对象在服务器调用init()方法时传递进来。

1. getFilterName() 获取Filter的名字
2. getServletContext() 获取ServletContext对象（即application）
3. getInitParameter() 获取Filter的初始化参数
4. getInitParameterNames() 获取所有初始化参数的名字



* FilterChain对象是在doFilter()方法被调用时作为参数传递进来的。

1. doFilter()方法用于调用Filter链上的下一个过滤器，如果当前过滤器为最后一个过滤器则将请求发送到目标资源。

# 创建Filter

### 实现Filter接口

|  |
| --- |
| **public** **class** HelloFilter **implements** Filter {  @Override  **public** **void** init(FilterConfig filterConfig) **throws** ServletException {  }  @Override  **public** **void** doFilter(ServletRequest request, ServletResponse response,  FilterChain chain) **throws** IOException, ServletException {  System.*out*.println("Hello Filter");  }  @Override  **public** **void** destroy() {  }  } |

### 在web.xml注册Filter

|  |
| --- |
| <filter>  <filter-name>HelloFilter</filter-name>  <filter-class>com.atguigu.web.filter.HelloFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>HelloFilter</filter-name>  <url-pattern>/index.html</url-pattern>  </filter-mapping> |

* 我们编写的这个Filter用来过滤发送到index.html的请求。
* 在doFilter()方法中我们只写了一个打印语句，并没有将请求转交给目标资源处理。
* 所以这个过滤器的效果是我们将不能正常查看到index.html页面，在访问index.html时页面上将会是一片空白。同时，控制台将输出Hello Filter。
* 如果想要index.html正确显示，需要在输出语句后，调用一个chain.doFilter(request, response)将请求放行。
* Filter的配置文件和Servlet的配置文件类似
  + filter-name 指定的是过滤器的名字
  + filter-class 指定的是过滤器的全类名
  + url-pattern 指定的是需要过滤器处理的请求地址
* 除此之外在filter-mapping还有一个子标签dispatcher，该标签用来指定需要Filter处理的请求类型，该标签可以配置四个值：
  + <dispatcher>REQUEST</dispatcher>
    - 用户直接访问资源时，会调用Filter
  + <dispatcher>FORWARD</dispatcher>
    - 通过转发访问时，会调用Filter
  + <dispatcher>INCLUDE</dispatcher>
    - 通过动态包含获取时，会调用Filter
  + <dispatcher>ERROR</dispatcher>
    - 当通过异常处理访问页面时，会调用Filter
* 这四种情况可以设置一个，也可以同时设置多个，如果不设置那么默认为REQUEST。

# Filter的生命周期

* Filter的生命周期和Servlet类似
  + 构造器：创建Filter实例是调用，Filter实例服务器一旦启动就会被创建
  + init()：实例创建后马上被调用，用来对Filter做一些初始化的操作
  + doFilter()：Filter的主要方法，用来完成过滤器主要功能的方法，每次访问目标资源时都会调用。
  + destroy()：服务器停止时调用，用来释放资源。

# Filter的目标资源

* Filter的目标资源指的是需要调用Filter来进行过滤处理的资源，例如上文我们配置的/index.html就是我们的目标资源，当我们访问项目根目录下的index.html时就会调用HelloFilter来进行过滤。
* 目标资源的配置方式主要有两种：
  + 第一种：
    - 通过filter-mapping的url-pattern来配置
    - 这里的配置和Servlet的url-pattern的规则相同
      * 精确匹配：/路径/资源名。
        + 如：/index.html 、/hello/index.jsp 、 /client/LoginServlet
        + 精确匹配只要在请求地址完全一样时才会调用Filter
      * 路径匹配：/路径名/\*
        + 如：/hello/\* 、/\*
        + 路径匹配只要是所请求的资源是在设置的路径下就会调用Filter

如：/hello/\* 只要访问 项目根目录/hello/ 下的任意资源就会调用Filter

如: /\* 只要访问 项目根目录下的资源就会调用Filter

* + - * 后缀匹配：\*.后缀名
        + 如：\*.jsp 、 \*.html
        + 后缀匹配只要访问的资源路径是以指定后缀结尾就会调用Filter
  + 第二种
    - 通过filter-mapping中的servlet-name来指定要过滤的Servlet
    - 如：以下是一个项目中的web.xml配置文件，在项目中有一个Filter加做HelloFilter，一个Servlet叫做HelloServlet。在Filter的filter-mapping中增加了一个servlet-name标签，将该标签的值设置成Servlet的名字，在访问Servlet时就会调用该过滤器过滤请求。

|  |
| --- |
| <filter>  <filter-name>HelloFilter</filter-name>  <filter-class>com.atguigu.web.filter.HelloFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>HelloFilter</filter-name>  <servlet-name>HelloServlet</servlet-name>  </filter-mapping>  <servlet>  <servlet-name>HelloServlet</servlet-name>  <servlet-class>com.atguigu.web.servlet.HelloServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>HelloServlet</servlet-name>  <url-pattern>/HelloServlet</url-pattern>  </servlet-mapping> |

# Filter的执行顺序

* 我们可以为一个资源配置多个Filter，当配置多个Filter以后就有一个执行顺序的问题，实际执行顺序是按照在web.xml文件中servlet-mapping的顺序决定的，如果顺序越靠前越先被调用。