# 过滤器

## 过滤器概述

### 什么是过滤器

Servlet/Filter/Listener统称为JavaWeb的三大组件

过滤器也叫做Filter, 所谓的过滤器其实就是拦截用户对资源的访问.

一个过滤器可以配置拦截多个资源, 一个资源也可以配置多个过滤器进行拦截.

### 开发过滤器的步骤

1.写一个类, 并实现一个Filter接口(并实现其中的方法)

2.在web.xml文件中配置过滤器所拦截的资源路径

### 过滤器的示例

1.FilterDemo1类

public class FilterDemo1 implements Filter {

/\* 在Filter实例创建之后立即执行,进行初始

\* 化的操作 \*/

public void init(FilterConfig arg0) throws ServletException {

System.out.println("FilterDemo1.init()");

}

/\*\* 当过滤器拦截到请求后, 将会调用doFilter

\* 方法, 对请求进行相关处理 \*/

public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)

throws IOException, ServletException {

System.out.println("FilterDemo1.doFilter()");

//模拟判断当前用户是否登录

String username =

req.getParameter("username");

if("张飞".equals(username)) {

System.out.println("用户已登录,放行...");

//放行过滤器(才能访问到后面的资源)

chain.doFilter(req, res);

}else {

//模拟跳转到登录页面, 提示用户登录

res.setContentType(

"text/html;charset=utf-8");

res.getWriter().write(

"您还没有登录, 请先登录...");

}

}

/\* 在Filter实例销毁之前执行destroy方法,

\* 进行善后的处理 \*/

public void destroy() {

System.out.println("FilterDemo1.destroy()");

}

}

2.FilterDemo1在web.xml中的配置

<filter>

<filter-name>FilterDemo1</filter-name>

<filter-class>com.tedu.filter.FilterDemo1</filter-class>

</filter>

<filter-mapping>

<filter-name>FilterDemo1</filter-name>

<url-pattern>/HelloServlet</url-pattern>

<url-pattern>/CartServlet</url-pattern>

</filter-mapping>

## 过滤器中的方法

### init方法

public void init(FilterConfig arg0) throws ServletException {

System.out.println("FilterDemo1.init()");

}

-- 在Filter实例创建之后立即执行, 进行初始化的操作

-- Filter实例是在, 服务器启动时, web应用加载之后, 服务器会创建出当前Web应用的所有过滤器实例, 创建之后, 会立即执行init方法

### doFilter方法

public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException {

}

-- 当过滤器拦截到用户对资源的访问时, 服务器会调用当前Filter的doFilter方法对请求进行相关的处理.

-- 因此doFilter方法是对所拦截到的请求进行相关处理的核心方法

参数1: req 就是request对象, 表示请求信息

参数2: res 就是response对象, 表示服务器将要响应给浏览器的响应信息

参数3: FilterChain: 表示过滤器链的对象

一个资源的访问可以配置多个过滤器进行拦截, 多个过滤器所组成的就是一个过滤器链, 这个过滤器链就可以用一个FilterChain对象进行表示

只有当前过滤器链中的所有过滤器都放行, 最终才能能访问到服务器中的那个资源, 否则就访问不到.

其中Filterchain的作用就是用来放行过滤器. 放行过滤器的方法是:

//放行过滤器, 接着往后执行...

chain.doFilter(request, response);

### destroy方法

public void destroy() {

System.out.println("FilterDemo1.destroy()");

}

-- 在filter实例销毁之前, 服务器会调用destroy方法进行善后的处理

-- Filter实例在服务器关闭, 或者Web应用被移出容器, 随着Web应用的销毁, Filter实例也会跟着销毁, 在销毁之前会调用destroy方法进行善后处理.

### 过滤器的配置

<filter>

<filter-name>FilterDemo1</filter-name>

<filter-class>com.tedu.filter.FilterDemo1</filter-class>

</filter>

<filter-mapping>

<filter-name>FilterDemo1</filter-name>

<url-pattern>/HelloServlet</url-pattern>

<url-pattern>/CartServlet</url-pattern>

</filter-mapping>

# 监听器

## 监听器的介绍

### 什么是监听器

监听器其实就是一个实现特定接口的Java程序, 这个Java程序通常用于监听另外一个Java对象的属性状态(比如, 对象的创建或者对象的销毁, 对象中属性的更新)变化, 当上面对象属性状态发生变化时, 就会通知监听器, 监听器就会调用特定的方法来进行处理!!

### 监听器的特点

(1)通常情况下, 监听器是一个接口, 其中的方法需要开发人员自己来实现

(2)通常监听器(listener)需要注册到事件源(btn)上

(3)监听器中的方法会在特定事件发生时执行

## JavaWeb中的监听器

### 监听器的开发步骤

1.写一个类, 需要实现一个监听器接口

2.在web.xml文件中注册监听器完成监听

### ServletContextListener监听器示例

需求: 监听ServletContext对象的创建与销毁, 当对象创建时, 输出”ServletContext轻轻地来了..”, 当对象销毁时, 输出”ServletContext轻轻的走了..”

## 总结

Servlet:Servlet本质上是一个java程序, 最擅长的是(主要作用)处理请求, 当服务器接收到请求后, 会调用Servlet来处理请求, 最后处理的结果由JSP进行显示.

Filter: 过滤器的主要功能不是对请求进行处理, 而是将用户对资源的访问拦截下来, 做一些相关的处理(比如, 只要在用户登录后才可以查看购物车或者订单, 再比如, 通过过滤器实现全站乱码处理)

Listener: 监听器就是一个实现特定接口的Java程序, 是对特定的对象进行监听, 监听对象的属性状态变化, 当对象发生这些变化时, 会通知监听器, 监听器再调用方法进行处理.