# 过滤器

## 过滤器概述

### 什么是过滤器?

Servlet技术规范中，定义了Servlet、Filter、Listener三门技术， 其中Filter也叫做过滤器，通过过滤器技术，开发人员可以实现用户在访问某个资源之前或之后，对访问的请求和响应进行拦截，从而做一些相关的处理。

总结：

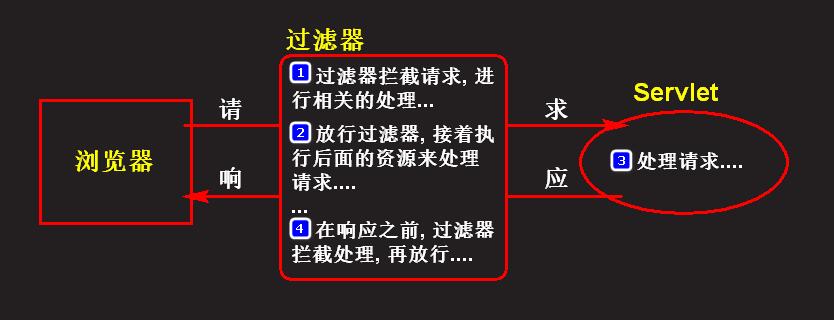
1.所谓的过滤器，就是拦截用户对资源的访问

2.一个过滤器可以拦截多个资源，一个资源也可以配置多个过滤器进行拦截

3.其实所谓的拦截，就是将代表请求的request对象和代表响应的response对象拦截下来,拦截下来后:

（1）控制是否允许访问 – 比如:用户登陆之后才能访问自己的订单或购物车

（2）在访问资源之前或之后做一些处理 比如: 全站乱码解决



## 开发过滤器

### 开发过滤器的步骤

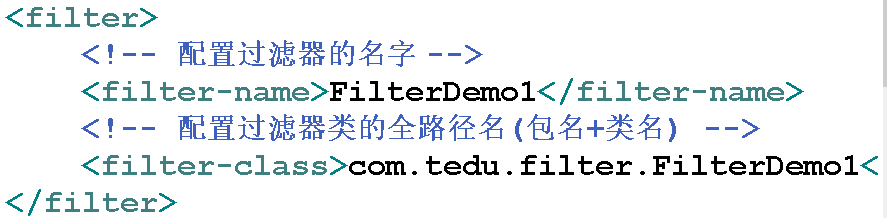
Servlet API中提供了一个Filter接口，开发web应用时，如果编写一个类实现了这个接口，则这个类就是一个过滤器类。

（1）写一个类实现Filter接口，并实现其中的方法

（2）在web应用的web.xml中配置过滤器

### 过滤器的配置

在创建完Filter类之后，需要在web应用的web.xml文件中配置过滤器所要拦截的资源路径，具体配置如下:



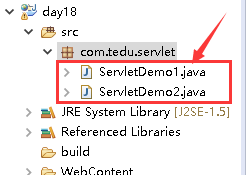


### 过滤器案例1

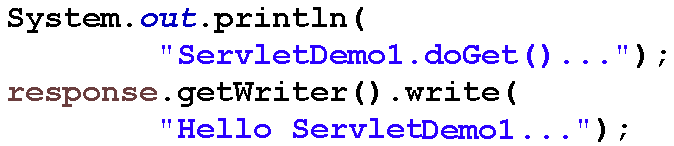
**1、需求：**创建两个Servlet，分别为ServletDemo1和ServletDemo2.分别来处理不同的请求。创建一个过滤器并配置过滤器拦截浏览器对ServletDemo1和ServletDemo2的访问.

**2、实现步骤：**

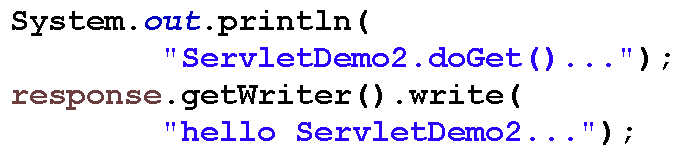
（1）创建两个Servlet，分别是ServletDemo1和ServletDemo2。



（2）ServletDemo1代码实现如下



（2）ServletDemo2代码实现如下



（3）测试ServletDemo1:



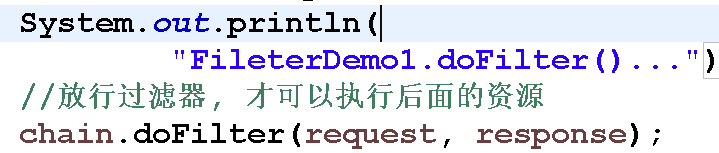
（4）测试ServletDemo2:



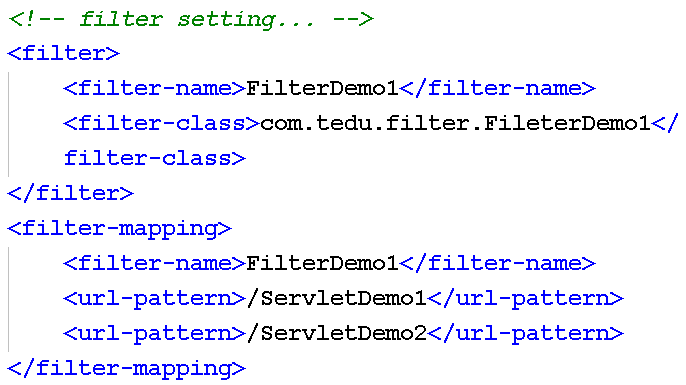
（5）创建一个过滤器，实现Filter接口



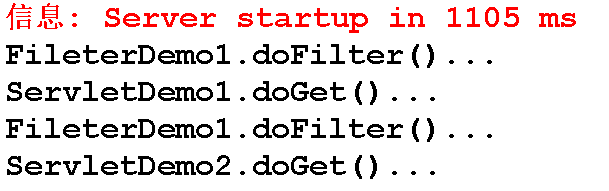
（6）FilterDemo1中doFilter方法的实现:



（7）配置过滤器对两个Servlet进行拦截.



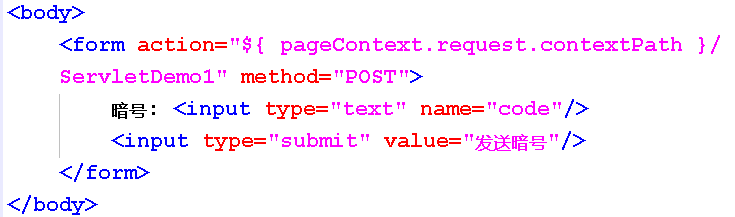
（8）分别访问ServletDemo1和ServletDemo2，查看FilterDemo是否会对这两资源的请求进行拦截处理.



### 过滤器案例2

改造上面的过滤器案例，允许用户在访问之前，在请求中携带两个“暗号“，接着配置两个过滤器对访问ServletDemo1和ServletDemo2的请求进行拦截，拦截之后判断输入的暗号。例如第一个过滤器判断第一个暗号，第二个过滤器判断第二个暗号。只有当两个暗号都正确时，才允许访问ServletDemo1和ServletDemo2。只要有一个过滤器判断暗号错误，将会跳转到提示页面提示暗号错误。

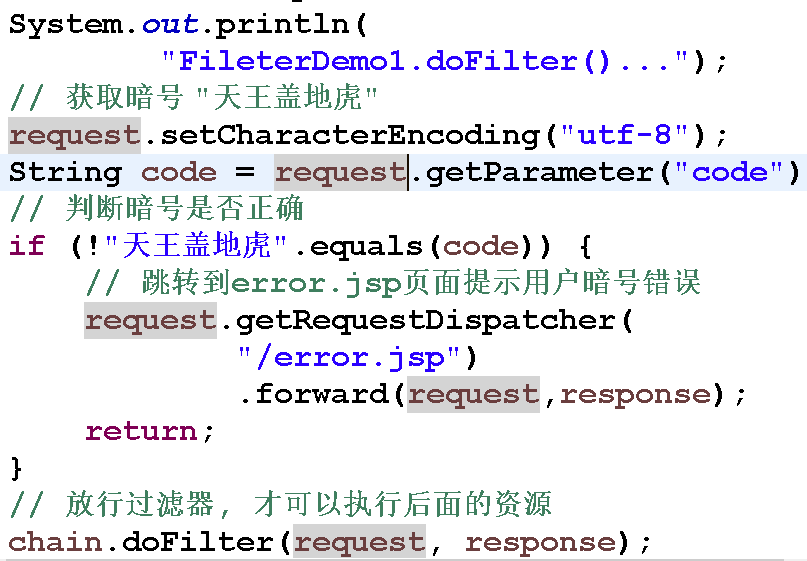
（1）编写code.jsp用来让用户输入暗号



(2)编写error.jsp用来提示(当用户输入的暗号不正确时，跳转到error页面进行提示)



（2）修改FilterDemo1中doFilter方法，对拦截的请求进行处理：获取请求中携带的暗号并判断暗号是否正确，如果不正确就跳转到error.jsp提示用户暗号错误，如果正确就放行访问ServletDemo1。



（3）测试访问

（4）创建FilterDemo2判断第二个暗号是否正确（自己实现）

## 过滤器中的方法

### init方法

public void init(FilterConfig arg0)

-- 当Filter实例创建之后，服务器立即调用init方法进行初始化的操作.

### doFilter方法

public void doFilter(ServletRequest request，ServletResponse response，FilterChain chain) throws IOException，ServletException

-- 当过滤器拦截到对资源的访问时，服务器会调用doFilter方法进行处理.

**参数FilterChain代表多个过滤器组成的过滤器链对象**.

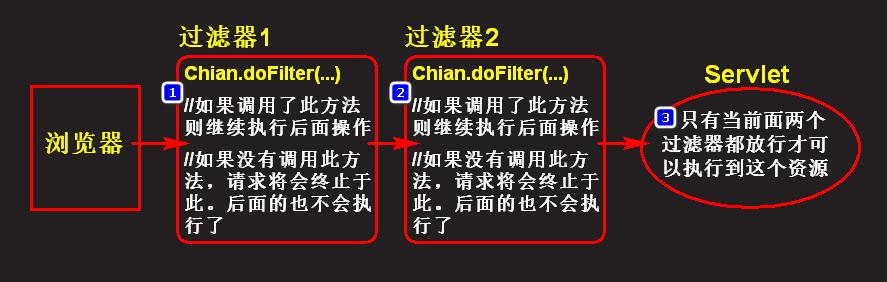
（1）一个资源可以配置多个过滤器进行拦截，多个过滤器执行的顺序是按照Filter在web.xml中对应的filter-mapping标签的先后配置顺序执行的.多个过滤器就组成了一条过滤器链.

（2）当过滤器拦截到对资源的访问时，如果处理之后放行过滤器，即调用FilterChain中的doFilter方法来放行过滤器. 接着才可以执行后面的资源



（3）如果后面仍然是过滤器，则也需要在过滤器的doFilter方法中调用FilterChain.doFilter方法才可以放行过滤器，执行后面的资源.

（4）如果后面没有过滤器，则访问对应的资源. 也就是说当所有的过滤器都调用了FilterChain的doFilter方法时，才可以放行所有的过滤器，才可以访问到对应的资源.



### destroy方法

public void destroy()

-- 在Filter实例销毁之前，执行destroy方法进行善后的处理

### 过滤器的生命周期（了解）

当服务器启动时，web应用加载后会立即创建出当前web应用中的所有的Filter对象，创建出来后，立即调用init方法进行初始化出操作. 从此以后这个Filter对象一直驻留在服务器的内存中, 为后续所拦截的请求服务，每次过滤到对资源的访问时，都会执行doFilter这个方法进行拦截处理，直到服务器关闭或者web应用移出容器为止，随着web应用的销毁，过滤器实例也跟着销毁，在销毁之前会调用destroy方法执行善后的处理.

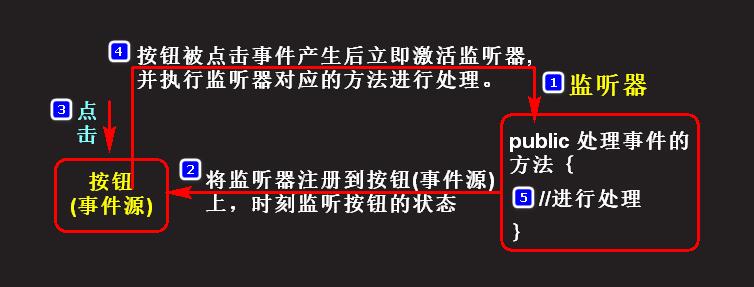
# 监听器

## 监听器概述

### 监听器概述

Listener也叫做监听器，是JavaWeb的三大组件之一。

所谓的监听器就是一个实现特定接口的java程序。这个程序专门用于监听另一个 java 对象状态变化（比如：对象的创建、销毁或属性改变等），当被监听对象发生上述事件后，就会通知监听器，监听器中的某个方法就会立即执行，来处理该事件。



**1.监听器:**

（1）事件源：按钮、车

（2）事件：按钮被点击、车被偷

（3）监听器：listener、警察

**2.监听器的特点：**

（1）通常是一个接口，其中的内容由我们开发人员来实现

（2）通常需要将监听器注册到事件源上

（3）监听器中的方法会在特定事件触发时执行.

### 监听器案例：按钮被点击

public static void main(String[] args) {

//1.创建一个窗口

JFrame frame = new JFrame();

//2.设置窗口的大小

frame.setSize(250, 200);

//3.设置窗口的位置

frame.setLocation(350, 250);

//4.创建一个按钮, 并添加到窗口中

JButton btn = new JButton("按钮");

/\*

\* 需求: 当按钮被点击时, 在控制台打印"hello btn~~!!"

\* 事件源: 按钮

\* 事件: 按钮被点击

\* 监听器: 当把监听器注册到事件源开始, 监听器会一直

\* 监听着事件源(按钮), 直到按钮被点击事件发生, 就会

\* 激活监听器, 执行监听器中对应的方法, 从事实现特定功能

\*/

//>>创建一个监听器(行为监听器)

ActionListener listener = new ActionListener() {

//处理事件的方法(事件处理器)

public void actionPerformed(ActionEvent e) {

String dateStr = new Date(e.getWhen()).toLocaleString();

System.out.println(dateStr);

System.out.println("hello btn~!!");

}

};

//>>将监听器注册到按钮上(此后监听器会一直监听按钮的状态)

btn.addActionListener(listener);

//5.将按钮添加到窗口中

frame.add(btn);

//6.设置窗口为显示状态

frame.setVisible(true);

System.out.println("执行完成...");

}

## 开发监听器

### 开发监听器的步骤

（1）写一个类(比如: MyServletContextListener)，必须要实现对应的监听器接口(如: ServletcontextListener)

（2）在web应用的web.xml文件配置监听器完成注册

### JavaWeb中的监听器

**ServletContextListener：**ServletContext对象的生命周期监听器，其中提供两个方法，一个是在ServletContext对象创建时调用，另一个是在ServletContext对象销毁时调用！

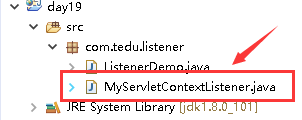
void contextInitialized(ServletContextEvent sce) 在创建ServletContext对象时调用

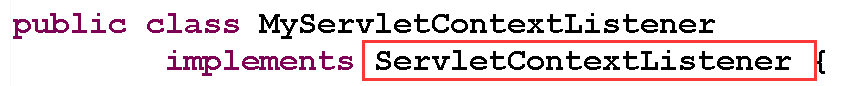
void contextDestroyed(ServletContextEvent sce) 在销毁ServletContext对象时地调用

### 监听器案例

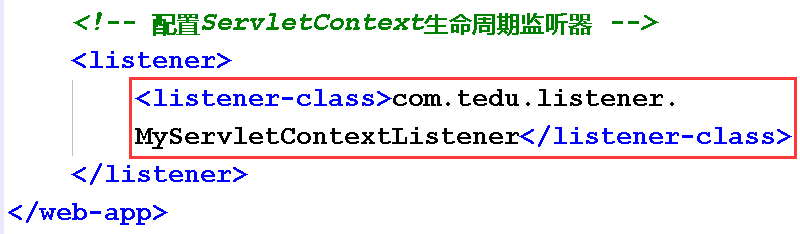
写一个类实现ServletContextListener(生命周期监听器)接口，实现contextInitialized和contextDestroyed方法，查看ServletContext对象何时创建以及何时销毁。

（1）写一个类MyServletContextListener，实现ServletContextListener接口





（2）在web.xml中配置监听器完成注册.



（3）实现contextInitialized和contextDestroyed方法并测试访问。