Filter

# Filter

### 概念

Filter称之为过滤器，web开发时，可以通过filter实现对访问的控制

对请求拦截，做一些操作再放行或不放行

### 特点

#### 可以拦截对资源的访问

#### 一个过滤器可以拦截多个资源，一个资源可以被多个过滤器拦截

#### 可以根据访问的url地址判断是否拦截

#### 所谓的拦截就是拦下来代表请求的request和代表响应的response

#### 拦截后可以控制是否放行，或者在放行之前做一些额外操作

# 开发过滤器

### 写一个类实现Filter接口



#### init()

初始化方法，当Filter对象创建时调用此方法

#### doFilter()

拦截器的核心方法，拦截后都会执行该方法，若不做任何操作，默认不会放行

##### 放行

可通过 传递的参数FilterChain放行

filterChain.doFilter(servletRequest,servletResponse);

##### 拦截规则

匹配某个路径 /path

通配符多有资源：/\*

##### 执行流程

###### 单个拦截器

访问某个资源->拦截器匹配拦截->放行->资源执行完成->返回拦截器继续执行

###### 多个拦截器

访问某个资源->拦截器1匹配拦截->放行->拦截器2继续拦截->放行->资源执行完成->返回拦截器2继续执行->返回拦截器1继续执行

注意：当多个拦截器匹配相同资源时，根据web.xml中的mapping声明的先后顺序来判断哪个先执行，先声明先拦截

#### destory()

### 在web.xml中配置过滤器



# Filter相关对象

### init()中的FilterConfig参数

可以用来获取ServletContext

### doFilter()中的FilterChain

代表过滤器链，提供了doFilter方法，用来放行当过滤器

# 生命周期

Web应用启动时，filter对象随之创建，创建后执行init方法，一旦创建一直存活，直到道web应用被销毁时filter随之销毁，销毁之前调用destory方法

# 应用场景

### 统一设置编码

所有页面需要设置编码，可用filter过滤，统一设置编码后放行

#### 问题点

Response编码和request的post请求编码都可用一个方法解决，但request的get请求需手动编解码，我应该怎么手动编解码？编码后如何传到页面中？

##### web.xml中配置初始化参数

#### 编码(用装饰者设计模式)

##### Response编码

response编码可以通过setContentType("text/html;charset="+enCode);

解决

##### Request编码

Request编码分为get和post编码

###### post

setCharacterEncoding("utf-8");

###### get