# 过滤器（Filter）

## 作用

* 会在一组资源（jsp,servlet,.css,.html…）的前面执行;
* 它可以让请求得到目标资源，也可以不让请求得到目标资源;
* 即 过滤器有拦截请求的能力.

### 如何编写

* 写一个类实现Filter接口；
* 在 web.xml 中配置

### Filter 接口

* Void init(FilterConfig)

创建之后，马上执行：Filter 会在服务器启动时就创建；

* Void destroy()

销毁之前执行，在服务器关闭时销毁；

* Void doFilter(ServletResponse,ServletResquest,FilterChain)

每次过滤时都会执行

Filter 是单例的！

### 相关参数

* FilterConfig 🡪 与 ServletConfig 相似
* 获取初始化参数： getInitParameter();
* 获取过滤器名称；
* 获取application: getServletContext();
* FilterChain
* doFilter(ServletRequest,ServletResponse):放行！

放行，就相当于执行了目标 servlet 的 service() 方法！

## 多过滤器

FilterChain.doFilter()方法：

用来执行目标资源，或是执行下一个过滤器；

如果没有下一个过滤器就执行目标资源，如果有就执行下一个过滤器。

## 四种拦截方式

* 请求 dispatcher
* 转发 forward
* 包含 include
* 错误 error

### 配置

在 <filter-mapping> 中进行配置！

<filter>

<filter-name>FilterA</filter-name>

<filter-class>cn.itcast.web.Filter.FilterA</filter-class>

</filter>

<filter-mapping>

<filter-name>FilterA</filter-name>

<url-pattern>/\*</url-pattern>

<dispatcher>REQUEST</dispatcher> 默认的额

<dispatcher>FORWARD</dispatcher>

<dispatcher>INCLUDE</dispatcher>

<dispatcher>ERROR</dispatcher>

</filter-mapping>

## 多个 Filter 的执行顺序

按照 web.xml 中的 <filter-mapping> 中的配置顺序执行。

## 使用场景

### 预处理

执行目标资源之前做预处理工作，例如设置编码，通常都会放行，只是在目标资源执行之前做一些准备工作；

比如，几乎所有的 servlet 执行之前都会写 request.setCharacterEncoding()，那么就可以把该语句放入 Filter 中。

### 前处理

通过条件判断是否放行，例如校验当前用户是否已经登录，或者用户IP 是否已被禁用。

### 后处理

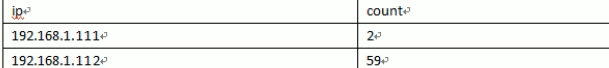
在目标资源执行之后，做一些后序的特殊处理工作，例如把目标资源输出的数据进行处理。

## 案例

### 分 IP 统计网站的访问次数

* 目标

在页面中使用列表显示各个 IP 的访问次数



* 如何实现

使用 Filter + map<string,integer>

* Map 何时创建，保存到哪里
* Map 需要在 Filter 中用来保存数据；
* Map 需要在页面中使用，打印 map 中的数据；

所以，需要将 map 保存到 servletContext 中。

由于一个项目仅需要一个 map ,所以使用 servletContextListener 在服务器启动时创建 map 即可。

### 基于角色的粗粒度权限控制

* 角色

User admin vistor

对应三个 jsp 文件

* 如何实现

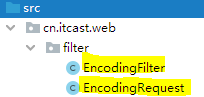
登录之后，往 Session 中保存用户的身份

设置过滤器，在用户访问相关 jsp 的时候，过滤器会先获取 Session 中是否有身份相关的属性，若有对应的权限，则放行（显示页面），否则转发到登录页面，并给出相关提示。

### 解决全站乱码问题（重要）

项目：encoding\_filter

其中：



EncodingFilter和EncodingRequest可以重复使用。

## 页面静态化

### 是什么？

首次访问去数据库取数据，然后把数据保存到一个静态 html 页面中，

二次访问，就不去数据库获取数据了，而是直接显示静态 html 页面。

从而加快页面响应速度，减少 I/O 资源消耗。

### 演示步骤

* 创建



* 静态化

项目：PageStatic\_filter

给出一个过滤器，把servlet 请求的资源做输出保存到HTML 中，重定向到 html 页面中。

第二次访问时，这个 html 页面已经存在，直接重定向，不用再去访问 servlet .