Java EE 企业应用系统设计

Java EE 过滤器编程

王晓东

wangxiaodong@ouc.edu.cn

中国海洋大学

December 21, 2017

参考书目

1. 吕海东，张坤编著，Java EE 企业级应用开发实例教程，清华大学出版社，2010 年 8 月

本章学习目标

1. 什么是过滤器？
2. 掌握过滤器的主要功能。
3. 掌握过滤器编程。
4. 掌握过滤器配置。

大纲

接下来…

过滤器概述

Java EE 过滤器 API

Java EE 过滤器编程和配置

过滤器的主要任务

过滤器应用实例：用户登录验证和权限验证

监听器概述

❖ Web 开发所遇到的常见问题

- 用户登录验证
- 开发中文 Web 遇到的汉字乱码

常规开发会带来大量的代码冗余，需要将处理上述问题的代码从每个 Web 组件中抽取出来，放在一个公共的地方，供所有需要这些公共功能代码的 Web 组件调用。
在 Servlet 2.3 规范中引入了新的 Web 组件技术——**过滤器**（Filter），使上述难题迎刃而解。

过滤器的基本概念

过滤器，对某种数据流动进行过滤处理的对象。在 Java EE Web 应用中，这种数据流动就是 HTTP 请求数据流和响应数据流。

- Filter 是对 HTTP 请求和响应的头（Header）和体（Body）进行特殊操作的 Web 组件。
- Filter 本身不生成 Web 响应，只对 Web 的请求和响应做过滤处理。这些操作都是在 Web 组件和浏览器毫不知情的情况下进行的。

过滤器的基本功能

过滤器采用 AOP（Aspect Oriented Programming）编程思想，使用拦截技术，在 HTTP 请求和响应达到目标之前，对请求和响应的数据进行预处理。主要包括：

- 对 HTTP 请求作分析，对输入流进行预处理。
- 阻止请求或响应的进行。
- 根据需求改动请求头的信息和数据体。
- 根据需求改动响应的头（Header）和体（Body）数据。
- 与其他 Web 资源进行协作。

过滤器的主要应用领域

- 登录检验
- 权限审核
- 数据验证
- 日志登记
- 数据压缩/解压缩
- 数据加密/解密

接下来…

过滤器概述

Java EE 过滤器 API

Java EE 过滤器编程和配置

过滤器的主要任务

过滤器应用实例：用户登录验证和权限验证

Javax.servlet.Filter 接口

所有过滤器必须实现 javax.servlet.Filter 接口。

```
public void init(FilterConfig filterConfig) throws ServletException
```

初始化方法，在 Web 容器创建过滤器对象后被调用，用于完成过滤器初始化操作，如取得过滤器配置的参数，连接外部资源。

```
public void doFilter(ServletRequest request, ServletResponse response,FilterChain chain)
```

过滤器的核心方法，在满足过滤器过滤目标 URL 的请求和响应时调用，开发人员在此方法中编写过滤功能代码。

```
public void destroy()
```

在过滤器销毁之前此方法被调用，此方法主要编写清理和关闭打开的资源操作，如关闭数据库连接、将过滤信息保存到外部资源操作。

Filter 的 doFilter() 方法

参数 1 请求对象 javax.servlet.ServletRequest

参数 2 响应对象 javax.servlet.ServletResponse

参数 3 过滤链对象 javax.servlet.Filter

- 此方法在每次过滤被激活时被调用。
- 此方法代码完成过滤器的操作功能。
- 如果是 HTTP 请求，需要强转为 HttpServletRequest 和 HttpServletResponse。
- 过滤器的请求和响应对象会被传递到被过滤的 JSP 或 Servlet。
- 可以通过对 request 对象操作，在 Servlet 之前修改请求对象的信息。
- 通过 response 对象操作，在 Servlet 响应之前修改响应信息。

javax.servlet.FilterChain 接口

此接口的对象表达过滤器链，在 Java EE 规范中对每个 URL 的请求和响应都可以定义多个过滤器，这些过滤器构成过滤器链。

- 过滤器使用 FilterChain 接口的 doFilter() 方法来调用过滤器链中的下一个过滤器，如果没有下级过滤器，则将用 doFilter 方法调用末端的 JSP 和 Servlet。
- 如果在过滤器的过滤方法中不调用 FilterChain 的传递方法 doFilter()，则将截断对下级过滤器或 JSP/Servlet 的请求和响应，使得 Web 容器没有机会运行 JSP 或 Servlet，达到阻断请求和响应的目的。

FilterChain 的 doFilter() 方法

```
public void doFilter(ServletRequest request, ServletResponse response) throws IOExcepti
```

- 此方法完成调用下级过滤器或最终请求资源，如 JSP 和 Servlet。
- 该方法传递请求对象和响应对象两个参数，并将请求对象和响应对象传递到下级过滤器或 Web 组件，这个过滤器链就可以共用一个请求对象和响应对象。

javax.servlet.FilterConfig 接口

FilterConfig 接口定义了取得过滤器配置的初始参数方法，通过实现了该接口的对象，可以取得在配置过滤器时初始参数和 ServletContext 上下文对象，进而取得 Web 应用信息。

```
public String getInitParameter(String name)
```

取得过滤器配置的初始参数。

```
public Enumeration getInitParameterNames()
```

取得过滤器配置的所有初始参数，以枚举器类型返回。

```
public String getFilterName()
```

取得配置的过滤器名称。

```
public ServletContext getServletContext()
```

取得过滤器运行的 Web 应用环境对象，通过 ServletContext 对象，过滤器可以取得所有 Web 应用环境数据供过滤器使用。

接下来…

过滤器概述

Java EE 过滤器 API

Java EE 过滤器编程和配置

过滤器的主要任务

过滤器应用实例：用户登录验证和权限验证

Java EE 过滤器编程和配置

1. 实现 javax.servlet.Filter 接口，编写过滤器类。
2. 在 Web 配置文件/WEB-INF/web.xml 中完成对过滤器的配置。

过滤器编程示例

编写过滤器类 CharEncodingFilter.java，实现 Filter 接口的所有方法。

```
1  public class CharEncodingFilter implements Filter {
2    private FilterConfig config = null;
3    private String contentType = null;
4    private String code = null;
5
6    public void init(FilterConfig config) throws ServletException {
7      this.config = config;
8      contentType = config.getInitParameter("contentType");
9      code = config.getInitParameter("encoding");
10   }
11
12   public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
13   throws IOException, ServletException {
14     //转换为 HTTP 请求对象
15     HttpServletRequest request = (HttpServletRequest)req;
16     if (request.getContentType().equals(contentType)) {
17       request.setCharacterEncoding(code); // 设置字符编码集
18     }
19     //继续下个过滤器
20     chain.doFilter(req, res);
21   }
22
23   public void destroy() {
24     //放置过滤器销毁的处理代码
25   }
26 }
```

过滤器编程示例

配置过滤器，在 Web 应用的配置文件/WEB-INF/web.xml 中配置声明和过滤 URL 地址映射。

❖ 过滤器声明

```
<filter>
  <description>此过滤器完成对请求数据编写及进行修改</description>
  <display-name>字符集编码过滤器</display-name>
  <filter-name>EncodingFilter</filter-name><!-- * -->
  <filter-class>javaee.ch08.CharEncodingFilter</filter-class><!-- * -->
  <init-param>
    <description>Content Type</description>
    <param-name>contentType</param-name>
    <param-value>text/html</param-value>
  </init-param>
  <init-param>
    <description>Encoding</description>
    <param-name>encoding</param-name>
    <param-value>utf-8</param-value>
  </init-param>
</filter>
```

过滤器编程示例

配置过滤器，在 Web 应用的配置文件/WEB-INF/web.xml 中配置声明和过滤 URL 地址映射。

❖ 过滤器 URL 映射

- 过滤器需要对所过滤的 URL 进行映射。当浏览器访问的 Web 文档 URL 地址符合过滤器的映射地址时，此过滤器自动开始工作。
- 如果有多个过滤器对某个 URL 地址都符合时，这些过滤器构成过滤器链，先声明的过滤器先运行，运行的顺序与声明的次序一致。

过滤器编程示例

过滤器映射语法：

```
<filter-mapping>
  <filter-name>EncodingFilter</filter-name>
  <servlet-name>SaveCookie</servlet-name>
  <servlet-name>GetCookie</servlet-name>
  <url-pattern>/employee/add.do</url-pattern>
  <url-pattern>/admin/*</url-pattern>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

过滤器编程示例

对过滤器映射标记的说明：

<filter-mapping> 与 <filter> 标记平级，且在 <filter> 之后，即先声明后映射的原则。

<filter-name> 应该与过滤器声明中的 <filter-name> 一致。

<url-pattern> 过滤器映射地址声明，每个过滤器映射可以定义多个。

<servlet-name> 指示过滤器对指定的 Servlet 进行过滤，每个过滤器映射可以定义多个。

过滤器编程示例

<dispatcher> 从 Servlet API 2.4 开始，过滤器映射增加了根据请求类型有选择的对映射地址进行过滤，提供标记 <dispatcher> 实现请求类型的选择。

- REQUEST 当请求直接来自客户时，过滤器才工作。
- FORWARD 当请求是来自 Web 组件转发到另一个组件时，过滤器工作。
- INCLUDE 当请求来自 include 操作时，过滤器生效。
- ERROR 当转发到错误页面时，过滤器起作用。

过滤器生命周期

1. **创建阶段** 根据 <filter-class> 标记定义的过滤器类，将累定义加载到服务器内存，并调用此类的默认构造方法，创建过滤器对象。
2. **初始化阶段** 创建 FilterConfig 对象，调用过滤器的 init() 方法，传入 FilterConfig 象，完成初始化工作。
3. **过滤服务阶段** 每次请求符合过滤器配置的 URL 时，过滤方法都将执行一次。
4. **销毁阶段** 当 Web 应用卸载或 Web 容器停止之前，destroy 方法被 Web 容器调用，完成卸载操作，Web 容器销毁过滤器对象。

接下来…

处理 HTTP 请求

❖ 修改请求头

调用 ServletRequest 或 HttpServletRequest 的各种 set 请求头方法对请求头进行修改。

```
1 request.setCharacterEncoding{"code"};
```

对文字乱码问题，可以在过滤器中进行集中处理，避免在每个 Servlet 或 JSP 中进行请求字符编码的转换。

❖ 修改请求对象的属性

调用请求对象的 setAttribute 方法对请求对象的属性进行增加、修改和删除。

```
1 request.setAttribute("infoType", "image/jpeg"); //设定请求对象的一个属性
2 request.removeAttribute("userId");
```

处理 HTTP 响应

过滤器可以在 HTTP 响应到达客户端浏览器之前，对响应头和响应体进行转换、修改等操作。
过滤器实现对响应处理的代码要在 FilterChain 的 doFilter() 方法之后完成，而对请求处理的代码要在 doFilter() 方法之前进行，流程如下：

```
1 public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
2    throws IOException, ServletException {
3   ... // 处理请求的代码放在 doFilter 之后
4   chain.doFilter(req, res); // 传递到链中的下个过滤器
5   ... // 处理响应的代码放在过滤器链传递之后
6 }
```

处理 HTTP 响应

❖ 修改响应头

```
1 response.setContentType("application/pdf");
2 response.setContentType("GBK");
```

❖ 修改响应体内容

过滤器使用 HTTP 响应对象的包装类 javax.servlet.http.HttpServletResponseWrapper 可以将响应体内容进行重新包装和处理，使浏览器接收到的是经过过滤器处理修改过的响应数据。
见本章后续示例。

阻断 HTTP 请求

常用于用户登录验证等操作。
在某种条件下，实现对请求的阻断，不让请求传递到链中的下一个对象，只要在过滤器的过滤方法中，不执行 FilterChain 的传递方法 doFilter() 即可。

```
1 if ( 某条件成立 ) {
2   chain.doFilter(request, response); // 继续传递请求
3 } else {
4   response.sendRedirect("url"); // 阻断请求，直接重定向到指定的 URL
5 }
```

接下来…

过滤器概述

Java EE 过滤器 API

Java EE 过滤器编程和配置

过滤器的主要任务

过滤器应用实例：用户登录验证和权限验证

THE END

wangxiaodong@ouc.edu.cn