Java Filter过滤机制详解----(转载)

Java Filter过滤机制详解

以前以为Filter只是一个特殊点的Servlet，用进只需在web.xml中配置一下拦截路径就可以了，可经过这两天的深入学习，才知道以前对Filter类的了解和使用都太过浅薄了。

在网上看了很多篇相关的技术文章，现提取它们中的精髓（我认为能说明问题的东西）及我个人的一些学习经验作如下总结，希望能帮助大家更好地理解Filter的过滤机制。

要学习Filter,就必须先理解和掌握servlet的调用机制和流程。关于servlet，网上也有很多相关文章，大家若有不明白的可以到百度中搜一下，我这里为了突出重点就不多说了。好，下面进入正题。

一、什么是Filter

Filter 技术是servlet 2.3 新增加的功能.servlet2.3是sun公司与2000年10月发布的,它的开发者包括许多个人和公司团体,充分体现了sun公司所倡导的代码开放性原则.由于众多的参与者的共同努力,servlet2.3比以往功能都强大了许多,而且性能也有了大幅提高.

　　它新增加的功能包括:

　　1. 应用程序生命周期事件控制;

　　2. 新的国际化;

　　3. 澄清了类的装载规则;

　　4. 新的错误及安全属性;

　　5. 不赞成使用HttpUtils 类;

　　6. 各种有用的方法;

　　7. 阐明并扩展了几个servlet DTD;

　　8. filter功能.

　　其中最重要的就是filter功能.它使用户可以改变一个request和修改一个response. Filter 不是一个servlet,它不能产生一个response,它能够在一个request到达servlet之前预处理request,也可以在离开servlet时处理response.换种说法,filter其实是一个”servlet chaining”(servlet 链).一个filter 包括:

　　1. 在servlet被调用之前截获;

　　2. 在servlet被调用之前检查servlet request;

　　3. 根据需要修改request头和request数据;

　　4. 根据需要修改response头和response数据;

　　5. 在servlet被调用之后截获.

你能够配置一个filter 到一个或多个servlet;单个servlet或servlet组能够被多个filter 使用.几个实用的filter 包括:用户辨认filter,日志filter,审核filter,加密filter,符号filter,能改变xml内容的XSLT filter等.

一个filter必须实现javax.servlet.Filter接口定义的三个方法: doFilter、init和destroy。(在三个方法在后面后有详细的介绍).

二、Filter体系结构

2.1、Filter工作原理（执行流程）

当客户端发出Web资源的请求时，Web服务器根据应用程序配置文件设置的过滤规则进行检查，若客户请求满足过滤规则，则对客户请求／响应进行拦截，对请求头和请求数据进行检查或改动，并依次通过过滤器链，最后把请求／响应交给请求的Web资源处理。请求信息在过滤器链中可以被修改，也可以根据条件让请求不发往资源处理器，并直接向客户机发回一个响应。当资源处理器完成了对资源的处理后，响应信息将逐级逆向返回。同样在这个过程中，用户可以修改响应信息，从而完成一定的任务。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

在这里，我要插几句——关于过滤链的问题：上面说了，当一个请求符合某个过滤器的过滤条件时该请求就会交给这个过滤器去处理。那么当两个过滤器同时过滤一个请求时谁先谁后呢？这就涉及到了过滤链FilterChain。

所有的奥秘都在Filter的FilterChain中。服务器会按照web.xml中过滤器定义的先后循序组装成一条链，然后一次执行其中的doFilter()方法。（注：这一点Filter和Servlet是不一样的，具体请参看我的另一篇文章：Servlet和Filter映射匹配原则之异同）执行的顺序就如下图所示，执行第一个过滤器的chain.doFilter()之前的代码，第二个过滤器的chain.doFilter()之前的代码，请求的资源，第二个过滤器的chain.doFilter()之后的代码，第一个过滤器的chain.doFilter()之后的代码，最后返回响应。

这里还有一点想补充：大家有没有想过，上面说的“执行请求的资源”究竟是怎么执行的？对于“执行第一个过滤器的chain.doFilter()之前的代码，第二个过滤器的chain.doFilter()之前的代码”这些我可以理解，无非就是按顺序执行一句句的代码，但对于这个“执行请求的资源”我刚开始却是怎么也想不明白。直到我见到上面这张图片才恍然大悟（我说过了，这篇文章中的资料都是我从网上收集来的，当我看到上面的文字时是没有图片看的）。其实是这样的：

通常我们所访问的资源是一个servlet或jsp页面，而jsp其实是一个被封装了的servlet（每个jsp执行前都会被转化为一个标准的servlet，这点若还有不明白的请自己到网上查一下吧），于是我们就可以统一地认为我们每次访问的都是一个Servlet,而每当我们访问一个servlet时，web容器都会调用该Servlet的service方法去处理请求。而在service方法又会根据请求方式的不同（Get/Post）去调用相应的doGet()或doPost()方法，实际处理请求的就是这个doGet或doPost方法。写过servlet的朋友都应该知道，我们在doGet（或doPost）方法中是通过response.getWriter（）得到客户端的输出流对象，然后用此对象对客户进行响应。

到这里我们就应该理解了过滤器的执行流程了：执行第一个过滤器的chain.doFilter()之前的代码——>第二个过滤器的chain.doFilter()之前的代码——>……——>第n个过滤器的chain.doFilter()之前的代码——>所请求servlet的service()方法中的代码——>所请求servlet的doGet()或doPost()方法中的代码——>第n个过滤器的chain.doFilter()之后的代码——>……——>第二个过滤器的chain.doFilter()之后的代码——>第一个过滤器的chain.doFilter()之后的代码。

以上是我的理解，若有不对之处，还有大家不吝指教啊，大家一起学习！！

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2.2、 Servlet过滤器API

Servlet过滤器API包含了3个接口，它们都在javax.servlet包中，分别是Filter接口、FilterChain接口和FilterConfig接口。

2.2.1 public Interface Filter

所有的过滤器都必须实现Filter接口。该接口定义了init,doFilter0，destory()三个方法：

(1) public void init (FilterConfig filterConfig) throws ServletException．

当开始使用servlet过滤器服务时，Web容器调用此方法一次，为服务准备过滤器；然后在需要使用过滤器的时候调用doFilter()，传送给此方法的FilterConfig对象，包含servlet过滤器的初始化参数。

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

throws java.io.IOException，ServletException．

每个过滤器都接受当前的请求和响应，且FilterChain过滤器链中的过滤器（应该都是符合条件的）都会被执行。doFilter方 法中，过滤器可以对请求和响应做它想做的一切，通过调用他们的方法收集数据，或者给对象添加新的行为。过滤器通过传送至 此方法的FilterChain参数，调用chain．doFilterO将控制权传送给下一个过滤器。当这个调用返回后，过滤器可以在它的 Filter方法的最后对响应做些其他的工作。如果过滤器想要终止请求的处理或得到对响应的完全控制，则可以不调用下一个过滤 器，而将其重定向至其它一些页面。当链中的最后一个过滤器调用chain．doFilterO方法时，将运行最初请求的Servlet。

(3)public void destroy()

一旦doFilterO方法里的所有线程退出或已超时，容器调用

此方法。服务器调用destoryO以指出过滤器已结束服务，用于释

放过滤器占用的资源。

2.2.2 public interface FilterChain

public void doFilter(ServletRequest request，ServletResponse response)

thlows java．io．IOException，ServletException

此方法是由Servlet容器提供给开发者的，用于对资源请求过滤链的依次调用，通过FilterChain调用过滤链中的下一个过滤 器，如果是最后一个过滤器，则下一个就调用目标资源。

2.2.3 public interface FilterConfig

FilterConfig接口检索过滤器名、初始化参数以及活动的Servlet上下文。该接口提供了以下4个方法：

(1)public java．1ang．String getFilterName0

返回web．xml部署文件中定义的该过滤器的名称。

(2)public ServletContext getServletContextO

返回调用者所处的servlet上下文。

(3)public java.1ang.String getlnitParameter(java.1ang.String name)

返回过滤器初始化参数值的字符串形式，当参数不存在时，返回nul1．name是初始化参数名。

(4)public java.util.Enumeration getlnitParameterNames()

以Enumeration形式返回过滤器所有初始化参数值，如果没有初始化参数，返回为空。

2.3、过滤器相关接口工作流程

从编程的角度看，过滤器类将实现Filter接口，然后使用这个过滤器类中的FilterChain和FilterConfig接口。该过滤器类的

— 个引用将传递给FilterChain对象，以允许过滤器把控制权传递给链中的下一个资源。FilterConfig对象将由容器提供给过滤

器，以允许访问该过滤器的初始化数据。详细流程如下图所示:

2.4、过滤器配置

过滤器通过Web应用程序中的配置描述符web.xml文件中的明，包括部分：过滤器定义，由<filter>

元素表示，主要包括<filter-name>和<f'flter-class>两个必须的子元素和<icon>、<init-param>,<display-name>,<description>这4个可选的子元素。<filter-name>子元素定义了—个过滤器的名字，<filter-class>指定了由容器载入的实际类，<init-param>子元素为过滤器提供初始化参数。

<filter-mapping> 主要由<filter-name>,<servlet-name>和<url-pattem>子元素组成。<servlet-name>将过滤器映射到一个或多个Servlet上，<url-pattem>将过滤器映射到—个或多个任意特征的URL的JSP页面。

三、应用实例

从上面分析可知，实现Servlet过滤器，需要两步：第一步开发过滤器，设计—个实现Fiker接口的类；第二步通过web.xml配置过滤器，实现过滤器和Servlet、JSP页面之间的映射。以下设计一个简单的IP地址过滤器，根据用户的IP地址进行对网站的访问控制。

(1)过滤器的设计ipfilter.java

package ipf；

imp0rt java．io．IOException；

imp0rt javax．servlet．\*；

public class ipfilter implements Filter／／实现Filter接口

{protected FilterConfig config；

protected String rejectedlP；

public void init(FilterConfig filterConfig)throws

ServletException

{this．config=filterConfig；／／从Web)lE务器获取过滤器配置对象

rejectedlP=config．getlnitParameter( RejectedlP”)：

，，从配置中取得过滤lP

if(rejectedlP=：nul1)‘rejectedlP= )

)

public void doFilter(ServletRequest request，

ServletResponse response．FilterChain chain)throws

IOException，ServletException

{RequestDispatcher dispatcher=request.getRequestDispatcher("");

String remotelP=request．getRemoteAddrO；//获取客户请求lP

int i=remotelP．1astlndexOf("．")；

int r=rejectedlP．1astlndexOf(”．”)：

String relPscope=rejectedlP．substring(0，r);／／过滤lP段

if(relPscope．equals(remotelP．substring(O．i)))

{ dispatcher．forward(request，response)；//重定向到rejectedError．jsp页面

retum；／／阻塞，直接返Web回客户端

}

else{chain．doFilter(request，response)；//调用过滤链上的下一个过滤器

}

}

public void destroy()

//过滤器功能完成后，由Web服务器调用执行，回收过滤器资源

注意：chain．doFilterO语句以前的代码用于对客户请求的处理；以后的代码用于对响应进行处理。

(2)配置过滤器

在应用程序Web—INF目录下的web．xml描述符文件中添加以下代码：

<filter>

<filter-name>ipfIter<／filter-name>／／过滤器名称

<filter-class>ipf．ipfiIter<／filter-class>／／实现过滤器的类

<init—param>

<param—name>RejectedlP<／param-name>／／过滤器初始化参数名RejectedlP

<param-value>192.168.12.\*／param-value>

<／init—pamm>

<／filter>

<filter-mapping>／／过滤器映射(规律规则)

<filter-name>ipfiIter<／filter-name>

<url—pattem>/\*<／ud-pattem>

／／映射到Web应用根目录下的所有JSP文件

<／filter-mapping>

通过以上设计与配置，就禁止了IP地址处在192.168.12网段的用户对网站的访问。

四、 结束语

Servlet过滤器功能强大，应用广泛，除支持Servlet和JSP页面的基本功能，比如13志记录、性能、安全、会话处理、XSLT转换等外，在J2EE应用程序中使用Java Servlet过滤器转换其输出，以便兼容任何类型客户端也表现出了很好的前景。Servlet过滤器能够侦测到来自使用WAP协议(无线应用协议)的移动客户端的呼叫，并且将答复内容转换成WML(无线标记语言)格式。Servlet过滤器也能检测到来自iMode无线客户的呼叫，并将其转变成cHTML(紧凑HTML)格式等等。因而，深刻理解Servlet过滤器的工作机制，熟练掌握编程技术，在实际的开发过程中，可以不断地发现新的用途，增强组件的可重用性，提高Web应用程序的可维护性。

2009-06-09

Servlet与Filter

1、Servlet是使用Java Servlet应用程序设计接口及相关类和方法的Java程序。它通过创建一个框架来扩展服务器的能力，以提供在Web上进行请求和响应服务，充当一个桥梁的角色来联系客户机与应用程序，负责把客户机发出的请求封装成程序人员编写的应用程序中需要的对象。当客户机发送请求至服务器时，服务器可以将请求信息发送给 Servlet，并让 Servlet 建立起服务器返回给客户机的响应。 当启动 Web 服务器或客户机第一次请求服务时，可以自动装入 Servlet。装入后， Servlet 继续运行直到其它客户机发出请求。

2、Servlet的生命周期始于将它装入Web服务器的内存时，并在终止或重新装入Servlet时结束。

　　(1) 初始化

　　在下列时刻装入 Servlet：

　如果已配置自动装入选项，则在启动服务器时自动装入

　在服务器启动后，客户机首次向 Servlet 发出请求时

　重新装入 Servlet 时装入 Servlet 后，服务器创建一个 Servlet 实例并且调用 Servlet 的 init() 方法。在初始化阶段，Servlet 初始化参数被传递给 Servlet 配置对象。

　　(2) 请求处理

　　对于到达服务器的客户机请求，服务器创建特定于请求的一个“请求”对象和一个“响应”对象。服务器调用 Servlet 的 service() 方法，该方法用于传递“请求”和“响应”对象。service() 方法从“请求”对象获得请求信息、处理该请求并用“响应”对象的方法以将响应传回客户机。service() 方法可以调用其它方法来处理请求，例如 doGet()、doPost() 或其它的方法。

　　(3) 终止

　　当服务器不再需要 Servlet, 或重新装入 Servlet 的新实例时，服务器会调用 Servlet 的 destroy() 方法。

3、servlet的配置

Java代码

<servlet>

<servlet-name>advExtract</servlet-name>

<servlet-class>

com.anne.servlet.AdvExtractServlet

</servlet-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>advExtract</servlet-name>

<url-pattern>/ad/</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>advExtract</servlet-name>

<servlet-class>

com.anne.servlet.AdvExtractServlet

</servlet-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>advExtract</servlet-name>

<url-pattern>/ad/</url-pattern>

</servlet-mapping>

Servlet的配置一定要有<servlet></servlet>与<servlet-mapping></servlet-mapping>。

<init-param>的作用：

配置了初始化servlet需要用到的参数。

Java代码

public class AdvExtractServlet extends HttpServlet{

private String encoding;

@Override

public void init(ServletConfig config) throws ServletException {

super.init(config);

encoding = config.getInitParameter("encoding");

……

}

}

public class AdvExtractServlet extends HttpServlet{

private String encoding;

@Override

public void init(ServletConfig config) throws ServletException {

super.init(config);

encoding = config.getInitParameter("encoding");

……

}

}

如上面<init-param>中配置了encoding为UTF-8，那么在override HttpServlet类的init()方法的时候，就可以从config中取出”encoding”的值”UTF-8”。

4、Filter。

Filter与servlet是密不可分的。Tomcat在接收到一个来自客户端（如浏览器）的请求之后，在这个请求被传递给servlet之前，需要经过层层Filter的过滤。它能够在一个request到达servlet之前预处理request,也可以在离开servlet时处理response。

一个filter必须实现javax.servlet.Filter接口并定义三个方法:

1.void setFilterConfig(FilterConfig config) //设置filter 的配置对象

2. FilterConfig getFilterConfig() //返回filter的配置对象;

3. void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) //执行filter 的工作

Filter与servlet是一个层层包裹的关系，如下图：

对于每个request，将会先进入Filter1的doFilter()方法，在这个方法中会调用chain.doFilter()方法把控制权交给Filter2，然后在Filter2的doFilter()方法中会调用servlet的servici()方法，进入servlet，在离开servlet一层之后，依次退回到Filter2与Filter1，进行doFilter()中的后续处理，这里可以对从servlet返回来的response进行处理。

因此filter的实质就是拦截器(interceptor)，它与Spring AOP中的包围通知（MethodInterceptor）类似，层层的包围住servlet。

常见的filter的应用有：权限控制（如Acegi的实现），记录日志，转换编码格式，加密等。