



Рафеенко Е.Д.

Web- программирование

Фильтры

Содержание

- ▶ Понятие фильтра
- ▶ Доступ к контексту приложения
- ▶ Использование параметров инициализации
- ▶ Блокирование запросов



Фильтры: Обзор

- Взаимодействует с любым количеством сервлетов и динамических html страниц
- Проверяет request пришедший к сервлету или странице, затем:
 - **Вызывает запрашиваемый ресурс (сервлет/ html) в обычной форме.**
 - **Вызывает запрашиваемый ресурс с измененными параметрами запроса.**
 - **Вызывает запрашиваемый ресурс, но изменяет response перед отсылкой клиенту.**
 - **Предупреждает запрашиваемый ресурс перед вызовом и вместо редиректа к другому ресурсу, возвращает частичный статус, или генерирует заменяющий контент.**



Преимущества фильтров

- **Инкапсулирует общее поведение.**
 - Задача: есть 30 разных сервлетов или страниц, контент которых надо сжать для сокращения времени загрузки.
 - Решение: сделать 1 *compression filter* и применить его ко всем 30 ресурсам.
- **Отделяет high-level access decisions от кода представления.**
 - Задача: запретить доступ определенных сайтов без модификации индивидуальных страниц, к которым эти ограничения доступа применены.
 - Решение: создать *access restriction filter* и применить его к любому числу страниц.
- **Применяет многочисленные изменения к большому количеству разнообразных ресурсов.**
 - Задача: Есть много ресурсов, в которых надо изменить название компании.
 - Решение: Создать *string replacement filter* и применить его.



Шаги для создания фильтра

- **1. Создать класс, который имплементирует интерфейс Filter.**
 - Методы: `doFilter`, `init`, `destroy`
- **2. Описать поведение фильтра в методе `doFilter`.**
 - Аргументы: `ServletRequest`, `ServletResponse`, `FilterChain`
- **3. Вызвать метод `doFilter` из `FilterChain`.**
 - Это действие вызовет следующий *filter* (или несколько) или текущий ресурс.
- **4. Зарегистрировать связь фильтра с соответствующим URL.**
 - Использовать аннотацию `@WebFilter` или элементы `filter` и `filter-mapping` в `web.xml`.



Метод doFilter

- Форма записи

```
public void doFilter(ServletRequest request,  
    ServletResponse response, FilterChain chain)  
    throws ServletException, IOException {  
    ...  
    chain.doFilter(request, response);  
}
```

- Заметка!

– Первые два аргумента имеют тип `ServletRequest` и `ServletResponse`, а не `HttpServletRequest` и `HttpServletResponse`.

– Используйте приведение типов, если хотите использовать *HTTP* свойства.

- Заметка!

– Третий аргумент `FilterChain` – не `Filter`. Его метод *doFilter* отличается тем, что имеет только два аргумента.



Простой Reporting фильтр

```
public class ReportFilter implements Filter {  
    public void doFilter(ServletRequest request,  
        ServletResponse response,  
        FilterChain chain)  
        throws ServletException, IOException {  
        HttpServletRequest req =  
            (HttpServletRequest) request;  
        System.out.println(req.getRemoteHost() +  
            " tried to access " +  
            req.getRequestURL() + " on "  
            + new Date() + ".");  
        chain.doFilter(request, response);  
    }  
  
    public void init(FilterConfig config) throws  
        ServletException { }  
  
    public void destroy() {}  
}
```



Определение Reporting фильтра

...

```
<web-app...>
  <!-- Register the name "Reporter" for
        ReportFilter. -->
  <filter>
    <filter-name>Reporter</filter-name>
    <filter-class>
      by.bsu.filters.ReportFilter
    </filter-class>
  </filter>
```

- Заметка!
– Сервер загружает фильтры в память, когда Web приложение стартует. Поэтому, если фильтр не найден, все Web приложение будет заблокировано.



Связь Reporting фильтра с данными URLами

```
<!-- Apply Reporter filter to home page. -->
<filter-mapping>
    <filter-name>Reporter</filter-name>
    <url-pattern>/index.html</url-pattern>
</filter-mapping>
<!-- Also apply Reporter filter to
    servlet named "TodaysSpecial". -->
<filter-mapping>
    <filter-name>Reporter</filter-name>
    <servlet-name>TodaysSpecial</servlet-name>
</filter-mapping>
...
</web-app>
```




Reporting фильтр (Продолжение)

- Вывод стандартного потока вывода двух страниц, показанных выше:
 - **bsu.by** **tried to access**
<http://www.filtersrus.com/filters/index.html>
on Fri Feb 11 13:19:14 EDT 2023.
 - **bsu.by** **tried to access**
<http://www.filtersrus.com/filters/TodaysSpecial>
on Fri Feb 11 13:21:56 EDT 2023.
- Заметка: единственный фильтр может применяться к большому количеству разнообразных ресурсов
 - **Каждый** ресурс не нуждается в изменении кода



Доступ к контексту сервлета

- фильтру нужно прочитать или записать параметры Web-приложения
 - Для этого следует использовать методы класса `ServletContext`
- Чтобы получить `ServletContext` во время инициализации фильтра можно:
 - вызвать метод `getServletContext` **класса** `FilterConfig` переданного, как аргумент в `init()`
 - сохранить результат в поле фильтра
 - использовать поле в методе `doFilter`



LogFilter

```
public class LogFilter implements Filter{  
    protected FilterConfig config;  
    private ServletContext context;  
    private String filterName;  
  
    public void init(FilterConfig config)  
        throws ServletException {  
        // In case it is needed by subclass.  
        this.config = config;  
        context = config.getServletContext();  
        filterName = config.getFilterName();  
    }  
}
```



LogFilter (Продолжение)

```
public void doFilter(ServletRequest request,
    ServletResponse response,
    FilterChain chain)
    throws ServletException, IOException {
    HttpServletRequest req =
        (HttpServletRequest) request;
    context.log(req.getRemoteHost() +
        " tried to access " +
        req.getRequestURL() + " on " +
        new Date() + ". " +
        "(Reported by " + filterName + ".)");
    chain.doFilter(request, response);
}
```



Применение лог фильтра к Web приложению

```
<web-app>
...
<filter>
  <filter-name>Logger</filter-name>
  <filter-class>
    by.bsu.filters.LogFilter
  </filter-class>
</filter>
<filter-mapping>
  <filter-name>Logger</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
...
</web-app>
```



LogFilter (Окончание)

- Log файл:
 - by.bsu tried to access **<http://www.filtersrus.com/filters/business-plan.html>**
on Fri Feb 11 15:16:15 EDT 2023.
(Reported by Logger.)
 - by.bsu tried to access **<http://www.filtersrus.com/filters/tax-shelter/>**
on Fri Feb 11 10:24:11 EDT 2023.
(Reported by Logger.)



Определение параметров инициализации фильтра в web.xml

```
<filter>
  <filter-name>LateAccessFilter</filter-name>
  <filter-class>
    moreservlets.filters.LateAccessFilter
  </filter-class>
  <init-param>
    <param-name>startTime</param-name>
    <param-value>2</param-value>
  </init-param>
  <init-param>
    <param-name>endTime</param-name>
    <param-value>10</param-value>
  </init-param>
</filter>
```



Чтение параметров инициализации: Access Time Filter

```
public void init(FilterConfig config)
    throws ServletException {
    this.config = config;
    context = config.getServletContext();
    formatter =
        DateFormat.getDateTimeInstance(DateFormat.MEDIUM,
                                        DateFormat.MEDIUM);

    try {
        startTime = formatter.parse
            (config.getInitParameter("startTime"));
        endTime = formatter.parse
            (config.getInitParameter("endTime"));
    } catch (ParseException pe) { // Malformed/null
        // Default: access at or after 10 p.m. but before 6
        // a.m. is considered unusual.
        startTime = 22;          // 22:00
        endTime = 6;             // 6:00
    }
}
```




Access Time Filter (Продолжение)

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain chain)
    throws ServletException, IOException {
    HttpServletRequest req =
        (HttpServletRequest) request;
    GregorianCalendar calendar =
        new GregorianCalendar();
    int currentTime =
        calendar.get(calendar.HOUR_OF_DAY);
    if (isUnusualTime(currentTime, startTime, endTime)) {
        context.log("WARNING: " +
            req.getRemoteHost() + " accessed " +
            req.getRequestURL() + " on " +
            formatter.format(calendar.getTime()));
    }
    chain.doFilter(request, response);
}
```



Блокирование response'а

- **Обычная ситуация: вызов** doFilter объекта FilterChain
- **Необычная ситуация: редирект или генерация частного вывода**
- Пример

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain chain)
    throws ServletException, IOException {
    HttpServletRequest req =
        (HttpServletRequest) request;
    HttpServletResponse res =
        (HttpServletResponse) response;
    if (isUnusualCondition(req)) {
        res.sendRedirect("http://www.somesite.com");
    } else {
        chain.doFilter(req, res);
    }
}
```



BannedAccessFilter

```
public class BannedAccessFilter implements Filter {
    private HashSet bannedSiteTable;
    public void init(FilterConfig config)
        throws ServletException {
        bannedSiteTable = new HashSet();
        String bannedSites =
            config.getInitParameter("bannedSites");
        StringTokenizer tok =
            new StringTokenizer(bannedSites);
        while(tok.hasMoreTokens()) {
            String bannedSite = tok.nextToken();
            bannedSiteTable.add(bannedSite);
            System.out.println("Banned " +
                bannedSite);
        }
    }

    public void destroy() {}
}
```



BannedAccessFilter (Продолжение)

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain chain)
    throws ServletException, IOException {
    HttpServletRequest req = (HttpServletRequest)request;
    String requestingHost = req.getRemoteHost();

    String bannedSite = null;
    boolean isBanned = false;
    if (bannedSiteTable.contains(requestingHost)) {
        bannedSite = requestingHost; isBanned = true;
    }

    if (isBanned) {
        showWarning(response, bannedSite); // Custom response
    } else {
        chain.doFilter(request, response);
    }
} ...
```



BannedAccessFilter (Продолжение)

```
private void showWarning(ServletResponse response,
    String bannedSite) throws ServletException, IOException
{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 \" +
        \"Transitional//EN\">\n";
    out.println
    (docType +
    "<HTML>\n" +
    "<HEAD><TITLE>Access Prohibited</TITLE></HEAD>\n"+
    "<BODY BGCOLOR=\"WHITE\">\n" +
    "<H1>Access Prohibited</H1>\n" +
    "Sorry, access from or via " + bannedSite + "\n"+
    "is not allowed.\n" +
    "</BODY></HTML>");
}
```



Регистрация BannedAccessFilter в web.xml

...

```
<web-app>
```

```
  <filter>
```

```
    <filter-name>BannedAccessFilter</filter-name>
```

```
    <filter-class>
```

```
      moreservlets.filters.BannedAccessFilter
```

```
    </filter-class>
```

```
    <init-param>
```

```
      <param-name>bannedSites</param-name>
```

```
      <param-value>
```

```
        www.competingsite.com
```

```
        www.bettersite.com
```

```
        www.moreservlets.com
```

```
      </param-value>
```

```
    </init-param>
```

```
  </filter>
```

...



Регистрация BannedAccessFilter (Продолжение)

...

```
<filter-mapping>
  <filter-name>BannedAccessFilter</filter-name>
  <servlet-name>TodaysSpecial</servlet-name>
</filter-mapping>
<servlet>
  <servlet-name>TodaysSpecial</servlet-name>
  <servlet-class>
    moreservlets.TodaysSpecialServlet
  </servlet-class>
</servlet>
  <servlet-mapping>
    <servlet-name>TodaysSpecial</servlet-name>
    <url-pattern>/TodaysSpecial</url-pattern>
  </servlet-mapping>
```

...

```
</web-app>
```



Резюме

- Реализовать интерфейс Filter
- Переопределить методы `doFilter`, `init`, и `destroy`
 - `init()` и `destroy()` часто бывают пустыми
- Зарегистрировать фильтр в `web.xml`
 - Дать ему имя и назначить URL, к которому он применяется
- Доступ к контексту сервлета
 - Получить его в `init()` и сохранить его в поле класса
- Фильтры имеют параметры инициализации
- Блокирование ресурсов
 - Просто отмените вызов `FilterChain.doFilter`
- Модифицированный response
 - Перенаправить `wrapper` к ресурсу, вызвать ресурс, получить `output` от `wrapper`'а, модифицировать его, перенаправить его клиенту.



Пример

// фильтр, устанавливающий кодировку запроса :

```
package com.epam.filters;
import java.io.IOException;
import javax.servlet.*;

public class SetCharacterEncodingFilter implements Filter {
    private FilterConfig filterConfig = null; private String encoding;
    public void init(FilterConfig config) throws ServletException {
        this.filterConfig = config; encoding = config.getInitParameter("encoding");
    }
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        // чтение кодировки из запроса
        String encodingFromReq = request.getCharacterEncoding();
        System.out.println(encodingFromReq);
        // установка заданной кодировки, если не установлена
        if (! encodingFromReq.equalsIgnoreCase(encoding))
            response.setCharacterEncoding(encoding);
        chain.doFilter(request, response);
    }
    public void destroy() {    }}
```





Web.xml

<filter>

<filter-name>Set Character Encoding</filter-name>

<filter-class>com.epam.filters.SetCharacterEncodingFilter</filter-class>

<init-param>

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

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

</init-param>

</filter>

<!-- Define filter mappings for the defined filters -->

<filter-mapping>

<filter-name>Set Character Encoding</filter-name>

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

</filter-mapping>



Информационные ресурсы Организационные вопросы

