过滤器

Filter

功能：

1、用来拦截传入的请求和传出的响应。

2、修改或以某种方式处理正在客户端和服务端之间交换的数据流。

如何使用?

与使用 Servlet 类似，Filter 是 Java WEB 提供的一个接口，开发者只需要自定义一个类并且实现该接口即可。

```
package com.southwind.filter;

import javax.servlet.*;
import java.io.IOException;

public class CharacterFilter implements Filter {

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException
{
        servletRequest.setCharacterEncoding("UTF-8");
        filterChain.doFilter(servletRequest,servletResponse);
    }

}
```

web.xml 中配置 Filter

```
<filter>
  <filter-name>charcater</filter-name>
  <filter-class>com.southwind.filter.CharacterFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>charcater</filter-name>
  <url-pattern>/login</url-pattern>
  <url-pattern>/test</url-pattern>
</filter-mapping>
```

注意：doFilter 方法中处理完业务逻辑之后，必须添加 filterChain.doFilter(servletRequest,servletResponse);

否则请求/响应无法向后传递，一直停留在过滤器中。

Filter 的生命周期

当 Tomcat 启动时，通过反射机制调用 Filter 的无参构造函数创建实例化对象，同时调用 init 方法实现初始化，doFilter 方法调用多次，当 Tomcat 服务关闭的时候，调用 destory 来销毁 Filter 对象。

无参构造函数：只调用一次，当 Tomcat 启动时调用（Filter 一定要进行配置）

init 方法：只调用一次，当 Filter 的实例化对象创建完成之后调用

doFilter：调用多次，访问 Filter 的业务逻辑都写在 Filter 中

destory：只调用一次，Tomcat 关闭时调用。

同时配置多个 Filter，Filter 的调用顺序是由 web.xml 中的配置顺序来决定的，写在上面的配置先调用，因为 web.xml 是从上到下顺序读取的。

```
<filter>
  <filter-name>my</filter-name>
  <filter-class>com.southwind.filter.MyFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>my</filter-name>
  <url-pattern>/login</url-pattern>
</filter-mapping>

<filter>
  <filter-name>charcater</filter-name>
  <filter-class>com.southwind.filter.CharacterFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>charcater</filter-name>
  <url-pattern>/login</url-pattern>
  <url-pattern>/test</url-pattern>
</filter-mapping>
```

1、MyFilter

2、CharacterFilter

也可以通过注解的方式来简化 web.xml 中的配置

```
<filter>
  <filter-name>my</filter-name>
  <filter-class>com.southwind.filter.MyFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>my</filter-name>
  <url-pattern>/login</url-pattern>
</filter-mapping>
```

等于

```
@WebFilter("/login")
public class MyFilter implements Filter {

}
```

实际开发中 Filter 的使用场景：

1、统一处理中文乱码。

2、屏蔽敏感词。

```
package com.southwind.filter;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import java.io.IOException;

@WebFilter("/test")
public class WordFilter implements Filter {
    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException
{
        servletRequest.setCharacterEncoding("UTF-8");
        //将"敏感词"替换成"***"
        String name = servletRequest.getParameter("name");
        name = name.replaceAll("敏感词","***");
        servletRequest.setAttribute("name",name);
        filterChain.doFilter(servletRequest,servletResponse);
    }
}
```

```
package com.southwind.servlet;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet("/test")
public class TestServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        String name = (String) req.getAttribute("name");
        System.out.println("servlet:"+name);
    }
}
```

3、控制资源的访问权限。

```
package com.southwind.filter;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

@WebFilter("/download.jsp")
public class DownloadFilter implements Filter {

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain) throws IOException, ServletException
{
        HttpServletRequest request = (HttpServletRequest) servletRequest;
        HttpServletResponse response = (HttpServletResponse) servletResponse;
        HttpSession session = request.getSession();
        String name = (String) session.getAttribute("name");
        if(name == null){
            //不是登录状态
            response.sendRedirect("/login.jsp");
        }else{
            filterChain.doFilter(servletRequest,servletResponse);
        }
    }
}
```

文件上传下载

- JSP

1、input 的 type 设置为 file

2、form 表单的 method 设置 post，get 请求会将文件名传给服务端，而不是文件本身

3、form 表单的 enctype 设置 multipart/form-data，以二进制的形式传输数据

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form enctype="multipart/form-data" action="/upload" method="post">
        <input name="desc" type="text"/><br/>
        <input name="text" type="file"/><br/>
        <input type="submit" value="上传"/>
    </form>
</body>
</html>
```

- Servlet

fileupload 组件可以将所有的请求信息都解析成 FileIteam 对象，可以通过对 FileItem 对象的操作完成上传，面向对象的思想。

```
package com.southwind.servlet;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;
import java.util.List;

@WebServlet("/upload")
public class UploadServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
```

```
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
//        //通过输入流获取客户端传来的数据流
//        InputStream inputStream = req.getInputStream();
//        Reader reader = new InputStreamReader(inputStream);
//        BufferedReader bufferedReader = new BufferedReader(reader);
//        //通过输出流将数据流输出到本地硬盘
//        //获取文件夹的绝对路径
//        String path = req.getServletContext().getRealPath("file/copy.txt");
//        OutputStream outputStream = new FileOutputStream(path);
//        Writer writer = new OutputStreamWriter(outputStream);
//        BufferedWriter bufferedWriter = new BufferedWriter(writer);
//        String str = "";
//        while((str = bufferedReader.readLine())!=null){
//            System.out.println(str);
//            bufferedWriter.write(str);
//        }
//        bufferedWriter.close();
//        writer.close();
//        outputStream.close();
//        bufferedReader.close();
//        reader.close();
//        inputStream.close();
        try {
            DiskFileItemFactory fileItemFactory = new DiskFileItemFactory();
            ServletFileUpload servletFileUpload = new
ServletFileUpload(fileItemFactory);
            List<FileItem> list = servletFileUpload.parseRequest(req);
            for(FileItem fileItem : list){
                if(fileItem.isFormField()){
                    String name = fileItem.getFieldName();
                    String value = fileItem.getString("UTF-8");
                    System.out.println(name+":"+value);
                }else{
                    String fileName = fileItem.getName();
                    long size = fileItem.getSize();
                    System.out.println(fileName+":"+size+"Byte");
                    InputStream inputStream = fileItem.getInputStream();
//                    Reader reader = new InputStreamReader(inputStream);
//                    BufferedReader bufferedReader = new
BufferedReader(reader);
                    String path =
req.getServletContext().getRealPath("file/"+fileName);
                    OutputStream outputStream = new FileOutputStream(path);
//                    Writer writer = new OutputStreamWriter(outputStream);
```

```
//                    BufferedWriter bufferedWriter = new
BufferedWriter(writer);
                    int temp = 0;
                    while((temp = inputStream.read())!=-1){
                        outputStream.write(temp);
                    }
//                    bufferedWriter.close();
//                    writer.close();
                    outputStream.close();
//                    bufferedReader.close();
//                    reader.close();
                    inputStream.close();
                    System.out.println("上传成功");
                }
            }
        } catch (FileUploadException e) {
            e.printStackTrace();
        }
    }
}
```