# 过滤器

过滤器,设计执行流程:

1. 用户访问服务器
2. 过滤器:对Servlet进行拦截
3. 先进入过滤器,过滤器处理
4. 过滤器处理完后,在放行,此时请求到达Servlet/Jsp
5. Servlet处理
6. Servlet处理完,再回到过滤器,最后由tomcat服务器响应用户

import javax.servlet.\*;

接口Fileter (过滤器也是一个Servlet)

开发步骤:

1. 写一个普通java类,实现Filter接口
2. 配置过滤器

范例:

|  |
| --- |
| package filter;  import javax.servlet.\*;  import java.io.IOException;  public class HelloFilter implements Filter {  public HelloFilter(){  System.out.println("0.创建过滤器实例");  }  @Override  public void init(FilterConfig filterConfig) throws ServletException {  System.out.println("1.执行过滤器初始化方法");  }  //过滤器业务处理方法,在请求到达servlet之前,先进入此方法处理公用的业务逻辑操作  @Override  public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {  System.out.println("2.执行过滤器处理方法");  //如果有下一个过滤器,进入下一个过滤器,否则执行访问的servlet  filterChain.doFilter(servletRequest,servletResponse);  System.out.println("4.Servlet处理完成,又回到过滤器");  }  @Override  public void destroy() {  System.out.println("5.销毁过滤器实例");  }  } |

web.xml中配置:

|  |
| --- |
| <!--过滤器配置-->  <filter>  <!--过滤器名称-->  <filter-name>HelloFilter</filter-name>  <!--过滤器类-->  <filter-class>filter.HelloFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>HelloFilter</filter-name>  <!--拦截的url-->  <url-pattern>/\*</url-pattern>  </filter-mapping> |

# 2.相关的api

interface Filter 过滤器核心接口

void init(FilterConfig);初始化方法,在服务器启动时执行

void doFilter(request,response,filterChain);过滤器拦截的业务处理方法

void destory();销毁过滤器实例时候调用

interface FilterConfig 获取初始化参数信息

Stirng getInitParameter(String name) //获取参数名对应的值

Enumeration getInitParameterNames() //获取一个参数名集合的迭代器

interface FilterChain 过滤器链参数,一个个过滤器形成一个执行链.

void doFilter(ServletRequest request,ServletResponse response)//执行下一个过滤器或者Servlet/Jsp

# 拦截

注意:拦截是有顺序的,根据web.xml中的从上到下的顺序

<filter-mapping>

...

url-pattern可以写多个

1.拦截所有

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

2.拦截指定jsp

<url-pattern>/index.jsp</url-pattern>

<url-pattern>/list.jsp</url-pattern>

3.拦截所有的jsp

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

4根据servlet的内部名称拦截

<url-pattern>HelloServlet</url-pattern>

5.拦截指定的servlet

<url-pattern>/hello</url-pattern>

6.指定拦截指定的类型,可以写多个

//默认拦截的类型

<dispatcher>REQUEST</dispatcher>

//拦截转发

<dispatcher>FORWARD</dispatcher

//拦截包含的页面(RequestDispatcher.include(/page.jsp);)对page.jsp也进行拦截

<dispatcher>INCLUDE</dispatcher>

//拦截声明式异常信息:

<dispatcher>ERROR</dispatcher>

</filter-mapping>

# 4.案例

## 4.1编码统一处理

几乎每一个Servlet都要涉及编码处理,把公用代码抽取出来,用过滤器处理

代码实现输入:

1. Login.jsp 登陆，输入”中文”
2. LoginServlet.java 直接处理登陆请求
3. EncodingFilter.java 过滤器处理请求数据编码:GET/POST

## 4.2过滤无效数据

模拟:论坛敏感词汇

实现思路:

1. Dis.jsp 讨论区界面
2. DisServlet.java 处理提交

-->获取请求参数

-->保存到request域

-->:跳转到dis.jsp(从request取数据)

3 DataFilter.java 过滤器

-->编码

-->无效数据处理