一.Filter:过滤器,在浏览器发起请求到Servlet之前,可以对请求进行过滤处理

创建Filter

1.实现Filter

2.重写doFilter方法

3.配置过滤器的规则

过滤规则:

1.通配符:/*,匹配所有请求

2./user/*:路径匹配,匹配/user/开头的请求

3.*.jsp:后缀匹配,匹配以.jsp结尾的请求

4./login:精确匹配,匹配/login这个请求

注:路径匹配和后缀匹配不能同时出现

优先级:精确匹配>路径匹配>通配符>后缀匹配

Filter的生命周期(从创建到销毁的过程):

1.服务器启动后,创建Filter的对象,调用init方法

2.当请求满足过滤器的规则时,进入都Filter

3.当服务器停止前,销毁Filter对象,执行destroy方法

```
@WebFilter(filterName = "LoginFilter",urlPatterns = {"/student/*","/user/modify"})
public class LoginFilter implements Filter {
 public void destroy() {
 System.out.println("LoginFilter销毁");
 }

public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws ServletException, IOException {

//转成http
 HttpServletRequest request= (HttpServletRequest) req;
 HttpServletResponse response= (HttpServletResponse) resp;
 String path = request.getRequestURI();
 System.out.println(path+"过滤处理中");
 HttpSession session = request.getSession();
 Object user = session.getAttribute("user");
 if (user!=null){
 //放行
 chain.doFilter(req, resp);
 }else {
 //过滤掉
request.getRequestDispatcher("/user/toLogin").forward(request,response);
 return;
 }


 }

 public void init(FilterConfig config) throws ServletException {

 System.out.println("LoginFilter创建成功!");
 }

}
```

```
/*Servlet生命周期:
1.服务器启动后,创建servlet对象,调用init方法,
2.当收到请求时,调用service方法
3.再根据请求方式,调用对应的doGet或dopost
4,当服务器停止时,销毁servlet对象,调用destroy方法
```

二.监听器

1.创建实现接口(ServletContextListener,

HttpSessionListener, HttpSessionAttributeListener, 选其一个)的类

2.重写接口的方法

```java
@WebListener()
public class CountListener implements ServletContextListener,
 HttpSessionListener, HttpSessionAttributeListener {

 //记录在线人数
 private int count;

 // Public constructor is required by servlet spec
 public CountListener() {
  }

  // -------------------------------------------------------
  // ServletContextListener implementation
  // -------------------------------------------------------
  public void contextInitialized(ServletContextEvent sce) {
  /* This method is called when the servlet context is
  initialized(when the Web application is deployed).
  You can initialize servlet context related data here.
  */
  }

  public void contextDestroyed(ServletContextEvent sce) {
  /* This method is invoked when the Servlet Context
  (the Web application) is undeployed or
  Application Server shuts down.
  */
  }

  // -------------------------------------------------------
  // HttpSessionListener implementation
  // -------------------------------------------------------
  public void sessionCreated(HttpSessionEvent se) {
  /* Session is created. */
  count++;
  ServletContext application = se.getSession().getServletContext();
  application.setAttribute("count",count);
  }

  public void sessionDestroyed(HttpSessionEvent se) {
  /* Session is destroyed. */
  count--;
  ServletContext application = se.getSession().getServletContext();
  application.setAttribute("count",count);
  }

  // -------------------------------------------------------
  // HttpSessionAttributeListener implementation
  // -------------------------------------------------------

  public void attributeAdded(HttpSessionBindingEvent sbe) {
  /* This method is called when an attribute
  is added to a session.
  */
  }

  public void attributeRemoved(HttpSessionBindingEvent sbe) {
  /* This method is called when an attribute
  is removed from a session.
  */
  }

  public void attributeReplaced(HttpSessionBindingEvent sbe) {
  /* This method is invoked when an attibute
  is replaced in a session.
  */
  }
 }
```

```html
<h2>当前在线人数${applicationScope.count}</h2>

<a href="${pageContext.request.contextPath}/user/logout">注销</a>
```