拦截器

定义

Spring Web MVC 的处理器拦截器类似于Servlet 开发中的过滤器Filter，用于对处理器进行预处理和后处理。

1. 拦截器定义 实现HandlerInterceptor接口，如下：

```
Public class HandlerInterceptor1 implements HandlerInterceptor{

 /**
  * controller执行前调用此方法
  * 返回true表示继续执行，返回false中止执行
  * 这里可以加入登录校验、权限拦截等
  */
 @Override
 Public boolean preHandle(HttpServletRequest request,
   HttpServletResponse response, Object handler) throws Exception {
  // TODO Auto-generated method stub
  Return false;
 }
 /**
  * controller执行后但未返回视图前调用此方法
  * 这里可在返回用户前对模型数据进行加工处理，比如这里加入公用信息以便页面显示
  */
 @Override
 Public void postHandle(HttpServletRequest request,
   HttpServletResponse response, Object handler,
   ModelAndView modelAndView) throws Exception {
  // TODO Auto-generated method stub

 }
 /**
  * controller执行后且视图返回后调用此方法
  * 这里可得到执行controller时的异常信息
  * 这里可记录操作日志，资源清理等
  */
 @Override
 Public void afterCompletion(HttpServletRequest request,
   HttpServletResponse response, Object handler, Exception ex)
   throws Exception {
  // TODO Auto-generated method stub

 }

}
```

2. 拦截器配置 针对某种handlermapping配置拦截器 注册的handler就会被拦截。（前提 各个handler配置，并非扫描添加）

```
<bean
 class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping">
 <property name="interceptors">
  <list>
   <ref bean="handlerInterceptor1"/>
   <ref bean="handlerInterceptor2"/>
  </list>
 </property>
</bean>
<bean id="handlerInterceptor1" class="springmvc.intercapter.HandlerInterceptor1"/>
<bean id="handlerInterceptor2" class="springmvc.intercapter.HandlerInterceptor2"/>
```

3. 针对所有mapping配置全局拦截器

```
<!--拦截器 -->
<mvc:interceptors>
 <!--多个拦截器,顺序执行 -->
 <mvc:interceptor>
  <!--/**表示所有的url，/*表示根url-->
  <mvc:mapping path="/**"/>
  <bean class="cn.yxd.filter.HandlerInterceptor1"></bean>
 </mvc:interceptor>
 <mvc:interceptor>
  <mvc:mapping path="/**"/>
  <bean class="cn.yxd.filter.HandlerInterceptor2"></bean>
 </mvc:interceptor>
</mvc:interceptors>
```

4. 测试
5. 总结：

preHandle按拦截器定义顺序调用

postHandler按拦截器定义逆序调用

afterCompletion按拦截器定义逆序调用

postHandler在拦截器链内所有拦截器返成功调用

afterCompletion只有preHandle返回true才调用

6. 拦截器应用

    1. 用户身份认证

```
Public class LoginInterceptor implements HandlerInterceptor{

 @Override
 Public boolean preHandle(HttpServletRequest request,
   HttpServletResponse response, Object handler) throws Exception {

  //如果是登录页面则放行
  if(request.getRequestURI().indexOf("login.action")>=0){
   return true;
  }
  HttpSession session = request.getSession();
  //如果用户已登录也放行
  if(session.getAttribute("user")!=null){
   return true;
  }
  //用户没有登录挑战到登录页面
  request.getRequestDispatcher("/WEB-INF/jsp/login.jsp").forward(request, response);

  return false;
 }
}
```

2.用户登陆controller

```
//登陆提交
 //userid：用户账号，pwd：密码
 @RequestMapping("/login")
 public String loginsubmit(HttpSession session,String userid,String pwd)throws Exception{

  //向session记录用户身份信息
  session.setAttribute("activeUser", userid);

  return "redirect:item/queryItem.action";
 }
```

```
//退出
@RequestMapping("/logout")
public String logout(HttpSession session)throws Exception{

 //session过期
 session.invalidate();

 return "redirect:item/queryItem.action";
}
```

完