登录验证功能如何实现?

要在目标资源响应给浏览器之前对请求进行判断，判断用户是否已经登录。

由于在每个handler方法中都进行验证用户是否已经登录，代码的冗余太多，因为验证用户是否已经登录的这个代码是业务管理功能对应的各个Handler方法都需要的。为了能不重复写验证用户是否登录的代码，可以利用Filter或Interceptor实现。

下面示范如何用Interceptor实现登录验证功能：

首先要思考：验证用户是否登录的代码应该写在HandlerInterceptor接口的preHandle方法中还是postHandle方法中?

答：preHandle方法中，因为应该在执行handler方法之前就去验证用户是否已经登录。postHandle方法是在handler方法执行之后执行。

如何验证用户是否已经登录?

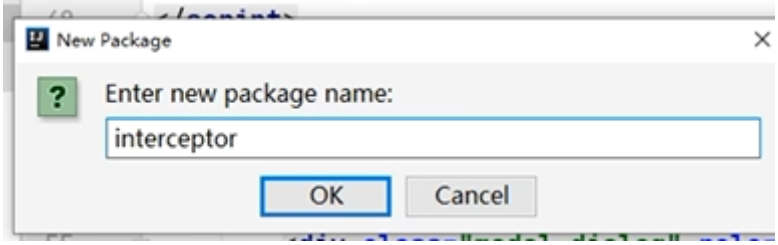
之前在做登录管理功能的时候，在用户登录成功之后，将用户的信息存入Session中。

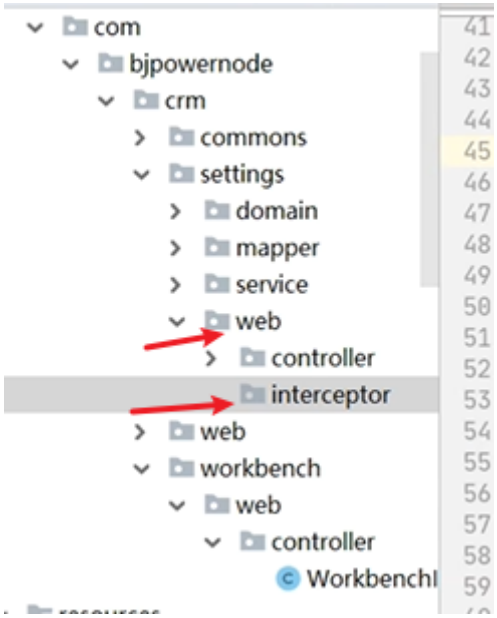
之前登录管理功能的代码：

```
        returnObject.setMessage("状态被锁定");
    }else if(!user.getAllowIps().contains(request.getRemoteAddr())){
        //登录失败，ip受限
        returnObject.setCode(Contants.RETURN_OBJECT_CODE_FAIL);
        returnObject.setMessage("ip受限");
    }else{
        //登录成功
        returnObject.setCode(Contants.RETURN_OBJECT_CODE_SUCCESS);

        //把user保存到session中
        session.setAttribute(Contants.SESSION_USER,user);

        //如果需要记住密码，则往外写cookie
        if("true".equals(isRemPwd)){
            Cookie c1=new Cookie( name: "loginAct",user.getLoginAct());
            c1.setMaxAge(10*24*60*60);
            response.addCookie(c1);
```

也就是说，只要用户登录成功了，Session中就一定有这个用户的信息。所以想要验证用户是否已经登录，只要去看看session中有没有用户信息即可。

登录验证功能的实现步骤：





1.创建一个类并实现HandlerInterceptor接口，在preHandle方法中写验证用户是否登录的代码。

```
public class LoginInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, C
        //如果用户没有登录成功，则跳转到登录页面
        HttpSession session=httpServletRequest.getSession();
        User user=(User) session.getAttribute(Contants.SESSION_USER);
        if(user==null){
            httpServletResponse.sendRedirect(httpServletRequest.getContextPath());//重定向时，url必须加项目的名称
            return false;
        }
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Obj

    }

    @Override
    public void afterCompletion(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse
```

2.配置这个自定义的拦截器类。

本来是应该在springmvc.xml中配置，但是由于ssm整合了，SpringMVC的配置是写到了applicationContext-mvc.xml中。所以应该在applicationContext-mvc.xml中配置。

思考：如果用户未登录，则根据什么东西判断是否要对请求进行拦截?

是根据请求的路径判断是否要进行拦截，因为地址栏输入的一定是RequestMapping，所以是根据被请求的RequestMapping判断是否要进行拦截。

请求的如果是没有登录就不能访问的RequestMapping，则这个请求就要被拦截。

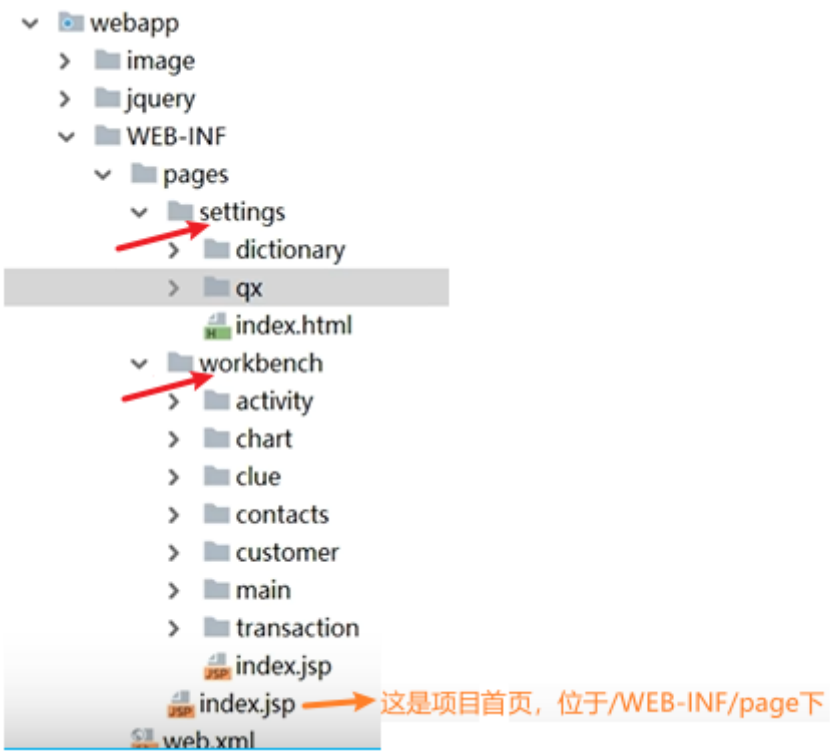
没有登录就能访问的RequestMapping：

1. 浏览器最终拿到的响应是首页、登录页面的`RequestMapping`。因为即使用户没登录，这两个页面都是允许浏览器显示给用户看的。
2. 实现登录功能的那个`RequestMapping`(也就是点击登录按钮后请求的那个`RequestMapping`)。因为正因为没有登录所以要访问这个`RequestMapping`。

除此之外，其它的RequestMapping都是没有登录就不能访问的。

思考：应该如何配置?

答：在写Handler方法的RequestMapping的value的时候，写的是" 浏览器最终得到的响应页面的文件夹路径 + handler方法名 "。
由于所有页面都是在WEB-INF/pages/下，所以WEB-INF/pages这部分就不要了。除了首页，其它的页面都是在settings或workbench下。所以没有登录就不能访问的RequestMapping的value值要么是以/settings开头，要么是以/workbench开头。



配置如下：

```
<beans ...

        .
        .
        .

    <bean> ... </bean>

    <mvc:interceptors>
        <mvc:interceptor>
            <!--配置拦截的请求  /**：匹配所有层级  不能用/*，因为/*是匹配一个层级 -->
            <mvc:mapping path="/settings/**"/>
            <mvc:mapping path="/workbench/**"/>
            <!--配置拦截的请求中要排除拦截的请求-->
            <mvc:exclude-mapping path="/settings/qx/user/toLogin.do"/>
            <mvc:exclude-mapping path="/settings/qx/user/login.do"/>
            <!--拦截器类-->
            <bean
class="com.bjpowernode.crm.settings.web.interceptor.LoginInterceptor"/>
        </mvc:interceptor>
    </mvc:interceptors>

        .
        .
        .

</beans>
```