1.实现接口

**public class** LoginInterceptor **implements** HandlerInterceptor {  
  
 */\*\*  
 \* 在控制器执行之前完成业务逻辑操作  
 \** ***@return*** *方法的返回值决定逻辑是否继续执行  
 \* 若为true：表示可以继续执行  
 \* 若为false：表示不再继续执行  
 \*/* **public boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** IOException {  
 *//判断当前用户是否已经登录* HttpSession session=request.getSession();  
 User loginUser=(User) session.getAttribute(**"userInf"**);  
 **if**(loginUser==**null**){  
 String APP\_PATH=session.getServletContext().getContextPath();  
 response.sendRedirect(APP\_PATH+**"/login"**);  
 **return false**;  
 }**else**{  
 **return true**;  
 }  
 }  
  
  
 */\*\*  
 \* 在完成视图渲染之后，执行此方法  
 \*/* **public void** afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) {  
  
 }  
  
 */\*\*  
 \* 表示在控制器执行完毕之后，执行的逻辑操作  
 \*/* **public void** postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) {  
  
 }  
}

2.继承接口的适配器

**public class** AuthorityInteceptor **extends** HandlerInterceptorAdapter {  
  
 @Autowired  
 **private** PermissionService **permissionService**;  
 *//快速重写父类方法的快捷键ctrl+o* @Override  
 **public boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) **throws** Exception {  
 *//获取用户请求地址* String uri=request.getRequestURI();  
 *//判断当前路径是否需要进行验证  
 //查询所有需要验证的集合* List<Permission> permissions=**permissionService**.queryAllPermission();  
 *//此处用set的原因是，希望不要重复* Set<String> uriSet=**new** HashSet<String>();  
 **for**(Permission permission:permissions){  
 **if**(StringUtil.*isNotEmpty*(uri)){  
 uriSet.add(request.getContextPath()+ permission.getUrl());  
 }  
 }  
 *//判断是当前uri是否跟权限有关，如果有关则拦下来判断，若无关则放行* **if**(uriSet.contains(uri)){  
 *//权限验证  
 //判断当前用户是否拥有对应的权限* Set<String> authUriSet=(Set<String>)request.getSession().getAttribute(**"authUriSet"**);  
 **if**(authUriSet.contains(uri)){  
 **return true**;  
 }**else**{  
 response.sendRedirect(request.getContextPath()+**"/error"**);  
 **return false**;  
 }  
 }**else**{  
 **return true**;  
 }  
  
 }  
}

3.弄完拦截器类之后再去springmvc配置文件中配置

*<!--配置拦截器，使得没登录的额用户无法通过链接访问到页面-->*<**mvc:interceptors**>  
 <**mvc:interceptor**>  
 <**mvc:mapping path="/\*\*"**/>  
 <**mvc:exclude-mapping path="/login"**/>  
 <**mvc:exclude-mapping path="/ajaxLogin"**/>  
 <**mvc:exclude-mapping path="/tools/\*\*"**/>  
 <**bean class="com.hua.web.LoginInterceptor"**></**bean**>  
 </**mvc:interceptor**>  
  
 <**mvc:interceptor**>  
 <**mvc:mapping path="/\*\*"**/>  
 <**mvc:exclude-mapping path="/login"**/>  
 <**mvc:exclude-mapping path="/ajaxLogin"**/>  
 <**mvc:exclude-mapping path="/tools/\*\*"**/>  
 <**bean class="com.hua.web.AuthorityInteceptor"**></**bean**>  
 </**mvc:interceptor**>  
</**mvc:interceptors**>