## 权限管理

权限管理，一般指根据系统设置的安全规则或者[安全策略](http://baike.baidu.com/view/160028.htm)，用户可以访问而且只能访问自己被授权的资源。

##### 权限管理设计模型

用户权限管理模型(用到5张表：权限表、角色表、用户表、角色和权限关系表、用户和角色关系表)

本系统设计：

权限表：

使用第三方用户授权系统，权限表包括：系统表、模块表、操作表

用户表：

使用groupid区分不同的用户类型

使用sysid存储用户所属单位id

角色表：

用户角色对用户类型一对一。

用户角色采用系统初始化方式在系统中初始化。

用户和角色关系表：

使用第三方用户授权系统，用户和角色关系表包括：角色和系统对应关系表，角色和部署结点对应关系表，角色和模块对应关系表，角色和操作对应关系表

##### 用户授权第三方系统集成

使用第三方系统管理完成用户授权，通过用户session接入接口，顺利进入第三方系统页面，完成用户授权操作。

###### Session接入接口定义

**接口名称**：模块操作Session接入

**接口协议**：http

**接口地址**： http://系统管理地址? loginkeyString=

**接口功能**：通过接口方式进行在系统管理中进行用户认证

接口方向：药品采购系统🡪第三方系统管理

**发送数据：**

|  |  |
| --- | --- |
| 参数名称： | 说明 |
| loginkeyString | 加密参数，加密格式如下：  使用3des加密算法，将用户账号、用户密码(md5)、当前时间戳（毫秒）进行加密，三个中间用#分割，如下：  用户账号#用户密码#当前时间戳  第三方系统解密后拿到用户名、密码、时间戳  判断时间戳具当前差是多长时间，如果大于10秒则第三方拒绝处理  说明：第三方系统如果操作接口调用端口不用关心，调用接口人员将接口调用端口代码按照接口规范编写即可。 |

**返回数据**：

无

##### 获取用户权限

###### 菜单权限

用户登录成功根据用户角色从数据库查询用户的菜单权限，并将菜单权限存储至session中，实现不同用户登录显示不同的菜单

###### 操作权限

用户登录成功根据用户角色从数据库查询用户的操作权限，并将操作权限存储至session中，在拦截器中获取用户的操作权限，用户请求的url不在自己的操作权限范围内则为越权操作，提示用户：无此操作权限。

##### 权限拦截

常用的有过虑器方式、框架拦截器(springmvc拦截，struts拦截器等)

###### 配置公共访问地址

公开访问地址在resources/commonActions.properties文件中进行配置。

###### 权限拦截流程：



###### 权限拦截器

/\*\*

\* 权限拦截器

\* @author Thinkpad

\*

\*/

public class PermissionInterceptor implements HandlerInterceptor {

//进入action方法前要执行

@Override

public boolean preHandle(HttpServletRequest request,

HttpServletResponse response, Object handler) throws Exception {

// TODO Auto-generated method stub

//用户访问地址：

String url = request.getRequestURI();

//判断是否公开权限，如果是放行

//获取公开地址

List<String> url\_list\_open= ResourcesUtil.gekeyList(Config.ANONYMOUS\_ACTIONS);

//便利公开的地址，如果用户访问的地址url包括了公开的地址，则说明用户访问的地址就是公开的地址

for(String url\_v:url\_list\_open){

if(url.contains(url\_v)){

return true;//放行，用户继续访问

}

}

//校验用户访问的是否是公共权限

//获取公共权限地址

List<String> url\_list= ResourcesUtil.gekeyList(Config.COMMON\_ACTIONS);

//便利公开的地址，如果用户访问的地址url包括了公开的地址，则说明用户访问的地址就是公开的地址

for(String url\_v:url\_list){

if(url.contains(url\_v)){

return true;//放行，用户继续访问

}

}

//从session获取用户权限信息

HttpSession session =request.getSession();

ActiveUser activeUser = (ActiveUser)session.getAttribute(Config.ACTIVEUSER\_KEY);

//获取用户操作权限

List<Operation> operation\_list = activeUser.getOperationList();

//校验用户访问地址是否在用户权限范围内

for(Operation operation:operation\_list){

String url\_operation = operation.getActionUrl();

if(url.contains(url\_operation)){

return true;//放行，用户继续访问

}

}

//跳转到页面

request.getRequestDispatcher("/WEB-INF/jsp/base/refuse.jsp").forward(request, response);

return false;

}

//action方法执行完但返回视图前执行此方法

@Override

public void postHandle(HttpServletRequest request,

HttpServletResponse response, Object handler,

ModelAndView modelAndView) throws Exception {

// TODO Auto-generated method stub

}

//action方法执行完且视图返回（渲染）完成执行此方法

@Override

public void afterCompletion(HttpServletRequest request,

HttpServletResponse response, Object handler, Exception ex)

throws Exception {

// TODO Auto-generated method stub

}

}

###### 拦截器抛出异常

由于本系统提交结果为json格式数据，页面上由js函数进行解析，如果这里拦截器重定向到一个页面会导致js函数无法正确解析，这里需要将：

request.getRequestDispatcher("/WEB-INF/jsp/base/refuse.jsp").forward(request, response);

改为：

//105表示“没有操作权限”

ResultUtil.throwExcepion(ResultUtil.createFail(Config.MESSAGE, 105, null));

异常处理器需要在处理返回页面方法中添加如下处理代码：

