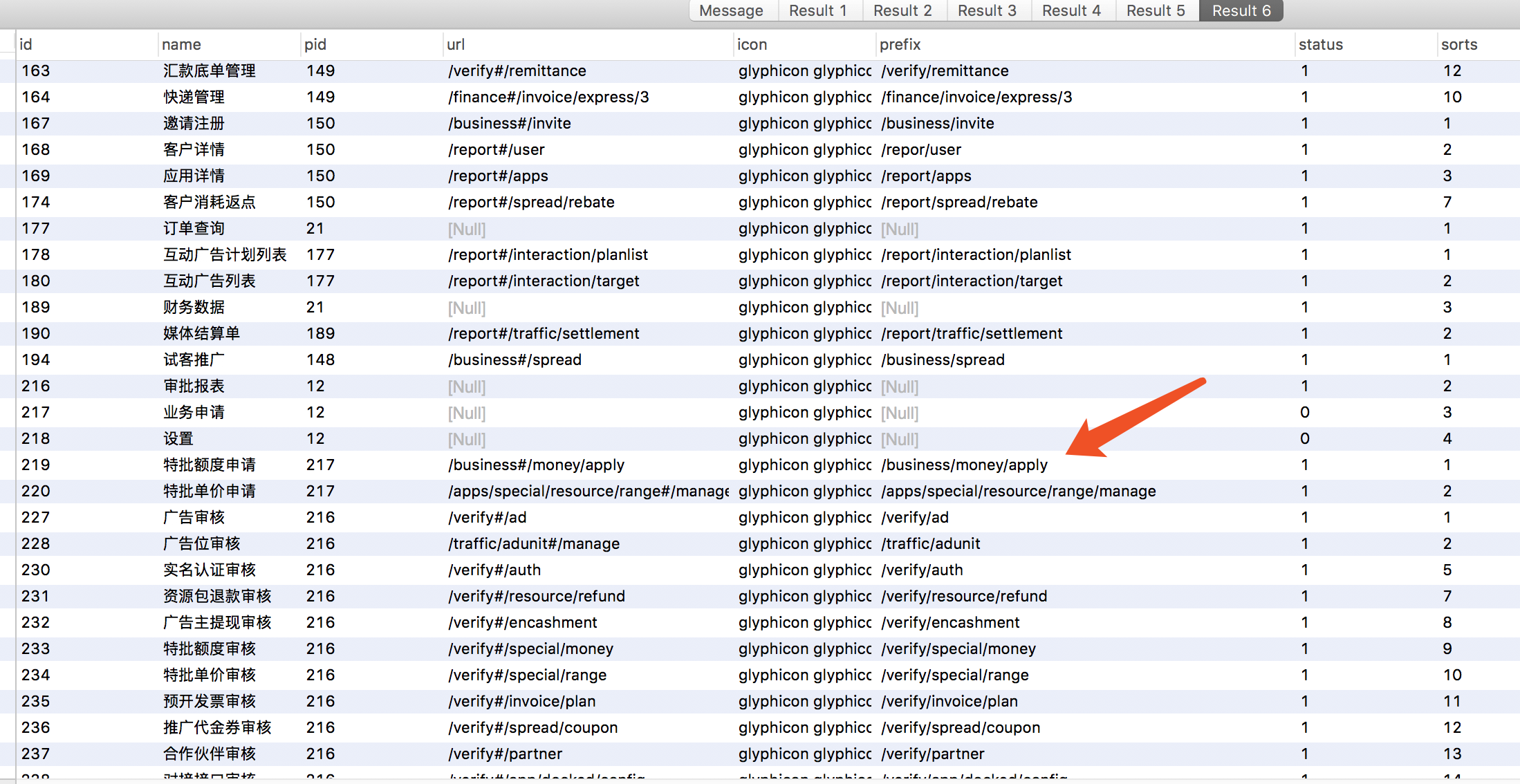
不同用户角色 权限访问rul的控制-拦截器、

# 看了代码才知道，这里的权限控制其实和当时大连理工大学基本上一致，大连理工大学是shiro控制了url（student,teache）,而我们这里的用户的在角色确定的时候，菜单所对应的（前缀）路径已经全部确定了。有就能访问，没有就当然不行喽

## 菜单数据表



# 1、拦截器

@Component  
public class PermissionInterceptor implements HandlerInterceptor {  
  
 private Logger logger = LoggerFactory.getLogger(PermissionInterceptor.class);  
  
  
 @Resource  
 private LoginService loginService;  
  
 @Override  
 public boolean preHandle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o) throws Exception {  
 String uri = httpServletRequest.getRequestURI();  
  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
 if(adminUser.getIsSuper() != null && adminUser.getIsSuper().intValue() == 1){  
 return true;  
 }  
#回去菜单全部的url点缀  
 Set<String> sysProtects = loginService.getSystemProtectUrl();  
 Set<String> admProtects = loginService.getAdmUserProtectUrl(adminUser.getId());  
  
 for (String c : sysProtects){  
 if(StringUtils.isNotBlank(c)){  
 if (uri.startsWith(c)){  
 if (admProtects.contains(c)){  
 return true;  
 } else {  
 logger.info("prefix:" + uri + ":" + c);  
 nopermission(httpServletRequest,httpServletResponse);  
 return false;  
 }  
 }  
 }  
 }  
 return true;  
 }  
  
 @Override  
 public void postHandle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o, ModelAndView modelAndView) throws Exception {  
  
 }  
  
 @Override  
 public void afterCompletion(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o, Exception e) throws Exception {  
  
 }  
  
  
 private void invalidate(HttpServletRequest request,HttpServletResponse response) throws IOException {  
 request.getSession().invalidate();  
 String ajax = request.getHeader("X-Requested-With");  
 if (StringUtils.equals(ajax,"XMLHttpRequest")) {  
 //设置登陆超时header  
 response.addHeader("sessionout","true");  
 } else {  
 response.sendRedirect("/public/sessionout");  
 }  
 }  
  
 private void nopermission(HttpServletRequest request,HttpServletResponse response) throws IOException{  
 String ajax = request.getHeader("X-Requested-With");  
 if (StringUtils.equals(ajax,"XMLHttpRequest")){  
 response.addHeader("nopermission","true");  
 } else {  
 response.sendRedirect("/public/nopermission");  
 }  
 }  
}

## 2、service

### 1、获取系统菜单全部前缀

@Override  
@Cacheable(value = CacheConstants.***CACHE\_PUBLIC***, key = "'getSystemProtectUrl'")  
public Set<String> getSystemProtectUrl() {  
 return sysMenuDAO.findSystemProtectUrls();  
}

@Query("select prefix from SysMenu where status = 1 and prefix is not null")  
public Set<String> findSystemProtectUrls();

### 2、获取当前用户所具备菜单的前缀

@Override  
@Cacheable(value = CacheConstants.***CACHE\_PUBLIC***, key = "'getAdmUserProtectUrl' + #admId")  
public Set<String> getAdmUserProtectUrl(Long admId) {  
 Set<Long> menuIds = new HashSet<>(sysAuthRoleMenuDAO.findAdmUserMenuIds(admId));  
 Set<String> result = new HashSet<>();  
  
 for (Long menuId : menuIds){  
 SysMenu sysMenu = sysMenuDAO.findOne(menuId);  
 if (sysMenu != null && sysMenu.getStatus() != null  
 && sysMenu.getStatus().compareTo(SysMenu.***STATUS\_ENABLE***) == 0  
 && StringUtils.isNotBlank(sysMenu.getPrefix())){  
 result.add(sysMenu.getPrefix());  
 }  
 }  
 return result;  
}

@Query("select menuId from SysAuthRoleMenu where roleId in (select roleId from SysAuthRoleAdminUser where admId = :admId)")  
public List<Long> findAdmUserMenuIds(@Param("admId") Long admId);

# 2、拦截成功controller，public 不拦截

@Controller  
@RequestMapping("public")  
public class PublicController {  
  
 private Logger logger = LoggerFactory.getLogger(PublicController.class);  
  
 @RequestMapping("datepicker/template")  
 public String page(){  
 return "tpl/datetimepicker.ftl";  
 }  
  
 @RequestMapping("sessionout")  
 public String sessionout(){  
 return "sessionout.ftl";  
 }  
  
 @RequestMapping("nopermission")  
 public String nopermission(){  
 return "nopermission.ftl";  
 }

# 3、html

<#assign ctx=Request.request.getContextPath()>  
<#import "/spring.ftl" as spring />  
  
<!DOCTYPE html>  
<html>  
<head>  
<body>  
<p>您没有合适的权限访问此资源</p>  
</body>  
</html>

# 4、springboot启动加载拦截器，和控制拦截的路径

@Configuration  
public class InterceptorConfig extends WebMvcConfigurerAdapter{  
  
 @Resource  
 private PermissionInterceptor permissionInterceptor;  
 @Resource  
 private APISessionInterceptor apiSessionInterceptor;  
 @Resource  
 private SessionNewInterceptor sessionNewInterceptor;  
  
 @Override  
 public void addInterceptors(InterceptorRegistry registry) {  
 registry.addInterceptor(sessionNewInterceptor).addPathPatterns("/\*\*")  
 .excludePathPatterns("/api/\*\*")  
 .excludePathPatterns("/register/\*\*")  
 .excludePathPatterns("/error")  
 .excludePathPatterns("/public/\*\*")  
 .excludePathPatterns("/fan/\*\*")  
 .excludePathPatterns("/custom/\*\*")  
 .excludePathPatterns("/templet/\*\*")  
 .excludePathPatterns("/workflow/ding/\*\*")  
 ;  
  
 registry.addInterceptor(permissionInterceptor).addPathPatterns("/\*\*")  
 .excludePathPatterns("/api/\*\*")  
 .excludePathPatterns("/register/\*\*")  
 .excludePathPatterns("/error")  
 .excludePathPatterns("/public/\*\*")  
 .excludePathPatterns("/fan/\*\*")  
 .excludePathPatterns("/custom/\*\*")  
 .excludePathPatterns("/templet/\*\*")  
 .excludePathPatterns("/workflow/ding/\*\*")  
 ;  
  
 registry.addInterceptor(apiSessionInterceptor).addPathPatterns("/api/\*\*");  
 super.addInterceptors(registry);  
 }  
  
  
 @Override  
 public void configureMessageConverters(List<HttpMessageConverter<?>> converters) {  
 MappingJackson2HttpMessageConverter jackson2HttpMessageConverter = new MappingJackson2HttpMessageConverter();  
 List<MediaType> mediaTypes = new ArrayList<>();  
 mediaTypes.add(MediaType.parseMediaType("text/html"));  
 mediaTypes.add(MediaType.parseMediaType("application/json"));  
 mediaTypes.add(MediaType.parseMediaType("application/xhtml+xml"));  
 mediaTypes.add(MediaType.parseMediaType("application/xml"));  
 mediaTypes.add(MediaType.parseMediaType("application/xhtml"));  
  
 jackson2HttpMessageConverter.setSupportedMediaTypes(mediaTypes);  
 converters.add(jackson2HttpMessageConverter);  
 }  
}