现勘3.0框架说明

### 一、框架结构



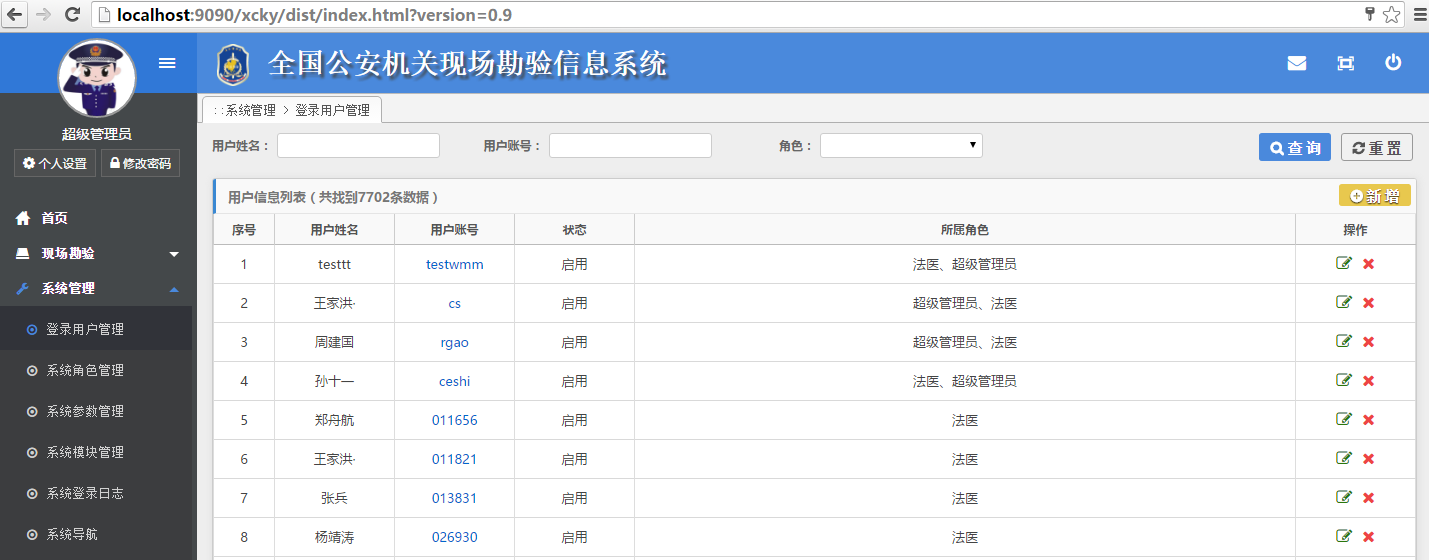
xcky-service-common是service层common的依赖

xcky-web-common是web层common的依赖

xcky-service-sceneCollecting、xcky-service-system这两个service是独立的，打包部署启动也是分别的，启动没有先后顺序。

xcky-web是前端控制层和表现层，dubbo.reference.check=false的加入，也使得启动的时候不会去检查相关的service是否已经注册成功。简单来说，service和web谁先启动都可以。

### 二、前后端交互



用户登录成功，前端用HTML5的localStorage来存储用户相关的属性（页面导航、用户信息、token之类的）。

整个页面都是用的index.html，通过更改右侧iframe的src来进行页面的跳转。

前后端整体采用ajax技术来进行交互，框架对之也进行了封装。

例如：

1、get方式ajax

$get(<http://localhost:8080/xcky/api/0/system/user/view>,{id:id},function(msg){

$template('#user-view-value', msg.data,function(item,i){

var t\_openFlag = item.openFlag;

item.openFlag = t\_openFlag=='1'?'启用':'禁用';

});

});

@RequestMapping(value="/view", method= RequestMethod.GET)

@ResponseBody

public JsonResult findSysUser(@RequestParam(value="id", required=true) String id){

JsonResult jsonResult = new JsonResult();

SysUser sysUser = new SysUser();

sysUser.setId(id);

try {

logger.info("查看id为[{}]的登录用户", id);

List<SysUser> list = sysUserService.findSysUserList(sysUser);

jsonResult.setSuccessData(list.get(0), 0);

return jsonResult;

} catch (Exception e) {

logger.error("查看id为[{}]的登录用户失败", id, e);

jsonResult = new JsonResult(0, "查看登录用户失败！");

return jsonResult;

}

}

2、post方式ajax

var saveAction= <http://localhost:8080/xcky/api/0/system/user/save>,

$post(saveAction,$("#user-add-form").serializeObject(), function(res){

toast('新增成功！',600,function(){

intoUserEdit(res.id);

}).ok();

},true);

3、查询分页

var searchAction=<http://localhost:8080/xcky/api/0/system/user/list>,

$('#user-query-result').pagingList({

action: searchAction,

currentPage:currentPage,

jsonObj:{

userName:$('#userName').val().trim(),

trueName:$('#trueName').val().trim(),

roleName:$('#query-role').val()

},

callback:function(data){

$template('#user-table tbody',data,function(item,i){

});

}

});

关于页面list template的遍历：

ajax请求回来的的数据需要进行遍历

<script type="text/template" id="user-list">

<tr>

<td>{rownum}</td>

<td>{trueName}</td>

<td>{openFlagTxt}</td>

</tr>

</script>

$template('#user-table tbody',data,function(item,i){

});

### 三、注意事项

1、前后端的每一个交互都要传递token参数，框架本身可以通过top.token来获得， 如果是自定义的ajax请求或者跳转，需要传到后台，否者会被拦截。

2、字段验证，具体参考sys.js文件



3、自定义注解TranslateObject、CurrentUser

@ResponseBody

@RequestMapping(value="/list",method=RequestMethod.POST)

public JsonResult list(@TranslateObject Assign assign,@CurrentUser SysUser sysUser) {

assign.setEvidenceAmount("1");

JsonResult result = new JsonResult();

if(StringUtils.isEmpty(assign.getAjssjq())){

assign.setAjssjq(sysUser.getUserUnit());

}

assign.setUser(sysUser);

List<Assign> list = caseInfoService.queryByXclr(assign);

Integer totalCount=caseInfoService.queryCount(assign);

result.setTotalCount(totalCount);

result.setFlag(1);

result.setData(list);

return result;

}

TranslateObject把前台传过来的json串转化为参数对象

CurrentUser则是获取当前用户

4、excel文件导出统一采用模板的是方式

参考代码：

List<SceneInvestigation> list=sceneInfoService.queryAllBy(sceneInvestigation,user);

String fileName = "excelTemplates/sceneInfo.vm";

Map<String, Object> map = new HashMap<String, Object>();

map.put("list", list);

String htmlData = TemplateUtil.parseTemplate(map, fileName);

try {

ExcelUtil.exportFile(request, response, htmlData, "sheet1", "test");

} catch (Exception e) {

logger.error("excel export error!", e);

}

5、cache的使用

(1)、对象的缓存

@RequestMapping(value="/api/dict/multi/{root}", method= RequestMethod.GET)

@ResponseBody

public JsonResult multiDict(@PathVariable("root") String root) {

String cacheKey = "dict\_multi\_"+root;

List<SysDict> list = null;

list = cacheService.get(cacheKey, List.class);

if(list == null||list.size()==0) {

list = sysDictService.queryMultiDictByRoot(root);

cacheService.put(cacheKey, list);

}

JsonResult result = new JsonResult();

result.setFlag(1);

result.setData(list);

return result;

}

(2)、sql的缓存，更新数据，增加查询条件都可以重新设置缓存

<cache type="org.mybatis.caches.ehcache.EhcacheCache" />

(3)、redis，系统启动的时候，把模板，用户信息，token，字典之类的存入redis

参考RedisServiceImpl类，set，get

6、分页插件PageHelper的使用

PageHelper.startPage(assign.getPageNum(), assign.getPageSize());

必须在service层实现类使用才可以，action试过不太好使

插件使用方法，只需要写一个查询全部数据的方法，该插件底层会分两步：

第一步、查询count

第二步、查询分页list

@Override

public JsonResult queryByXclr(Assign assign) {

if(assign.getPageSize()!=0){

PageHelper.startPage(assign.getPageNum(), assign.getPageSize());

}

Map<String, Object> map = new HashMap<String, Object>();

map.put("para", assign);

List<Assign> resList=new ArrayList<Assign>();

if(assign.getIsinvestigation().equals("2")){

resList = assignMapper.queryAllByXclr(map);

}else{

resList = assignMapper.queryByXclr(map);

}

PageInfo<Assign> pageInfo=new PageInfo<Assign>(resList);

return JsonResultUtil.success(pageInfo.getTotal(), resList);

}

7、AppContextListener类的加入，用户项目启动的时候初始化各种参数（字典缓存之类的）