# **环境搭建**

## 数据库与表结构

以建表循序列出

1用户权限表(GE\_RMS\_USERPOWER)； 主

2用户组表(GE\_RMS\_GROUP)；主

3角色表(GE\_RMS\_ROLE)；主

4角色指派表(GE\_RMS\_ROLE\_DESIGNATE)；

5功能表(GE\_RMS\_TASK)；主

6功能授权表(GE\_RMS\_TASK\_AUTH)；

7数据规则表(GE\_RMS\_DATARULE)；

8用户与组关系表(GE\_RMS\_USERGROUP)；

9用户组角色表(GE\_RMS\_GROUPROLE)；

10角色功能表(GE\_RMS\_ROLETASK)；

11用户权限除外表(GE\_RMS\_EXCPOWER)；

12人员数据权限表(GE\_RMS\_BUSPOWER)；

13规则业务数据信息表(GE\_RMS\_RULE\_BUSDATAINFO)；

建表语句见:权限设计-pdm

基本数据库逻辑：

1根据用户代码和机构代码查出对应的用户权限ID，然后根据用户权限ID查出对应的用户组，根据用户组ID查出所有的角色；

2根据归属机构代码查出角色指派表(GE\_RMS\_GROUP)中的所有的角色，根据1和2查询出结果取角色ID交集

3根据角色ID机构代码联合查出功能授权ID，进而查出角色所能看到的功能

4.根据用户权限ID查出用户权限除外表(GE\_RMS\_EXCPOWER)所对应的功能

5.将3中结果减去4中结果，即为用户所具有有效功能

## 基本jar包

所需环境及相关jar包JKD1.5+,

activation-1.1.1.jar

antlr-2.7.6.jar

aopalliance-1.0.jar

arch4-4.2.7.jar

asm-3.3.1.jar

asm-attrs-1.5.3.jar

aspectjrt-1.6.9.jar

aspectjweaver-1.6.9.jar

castor-core-1.3.1.jar

castor-xml-1.3.1.jar

cglib-2.2.2.jar

commons-beanutils-1.8.3.jar

commons-codec-1.4.jar

commons-collections-3.2.1.jar

commons-dbcp-1.4.jar

commons-fileupload-1.2.1.jar

commons-io-1.4.jar

commons-lang-2.5.jar

commons-logging-1.1.1.jar

commons-pool-1.5.4.jar

cxf-api-2.6.2.jar

dom4j-1.6.1.jar

ehcache-1.6.2.jar

ehcache-core-2.5.0.jar

ejb3-persistence-1.0.1.GA.jar

ezmorph-1.0.6.jar

freemarker-2.3.16.jar

geronimo-javamail\_1.4\_spec-1.7.1.jar

geronimo-ws-metadata\_2.0\_spec-1.1.3.jar

groovy-all-2.0.0.jar

guava-11.0.2.jar

hibernate-3.2.6.ga.jar

hibernate-annotations-3.4.0.GA.jar

hibernate-commons-annotations-3.0.0.ga.jar

hibernate-core-3.3.2.GA.jar

hibernate-ehcache-3.3.2.GA.jar

hibernate-entitymanager-3.3.2.GA.jar

jackson-annotations-2.0.4.jar

jackson-core-2.0.4.jar

jackson-core-asl-1.9.2.jar

jackson-databind-2.0.4.jar

jackson-mapper-asl-1.9.2.jar

jackson-module-hibernate-0.7.0.jar

jackson-module-jaxb-annotations-2.0.4.jar

javassist-3.11.0.GA.jar

javax.annotation-3.1.1.jar

jaxb-api-2.2.7.jar

jaxws-api-2.2.8.jar

joda-time-2.0.jar

json-lib-2.4-jdk15.jar

jsr181-api-1.0-MR1.jar

jsr250-api-1.0.jar

jsr305-1.3.9.jar

jta-1.1.jar

kaptcha-2.3.2.jar

log4j-1.2.16.jar

ognl-2.7.3.jar

ojdbc14-10.2.0.2.0.jar

saaj-api-1.3.4.jar

shiro-core-1.2.0.jar

shiro-ehcache-1.2.0.jar

shiro-spring-1.2.0.jar

shiro-web-1.2.0.jar

slf4j-api-1.6.1.jar

slf4j-log4j12-1.6.1.jar

spring-aop-3.0.5.RELEASE.jar

spring-asm-3.0.5.RELEASE.jar

spring-beans-3.0.5.RELEASE.jar

spring-context-3.0.5.RELEASE.jar

spring-context-support-3.0.5.RELEASE.jar

spring-core-3.0.5.RELEASE.jar

spring-expression-3.0.5.RELEASE.jar

spring-integration-core-2.1.3.RELEASE.jar

spring-integration-groovy-2.1.3.RELEASE.jar

spring-integration-scripting-2.1.3.RELEASE.jar

spring-jdbc-3.0.5.RELEASE.jar

spring-orm-3.0.5.RELEASE.jar

spring-tx-3.0.5.RELEASE.jar

spring-web-3.0.5.RELEASE.jar

spring-webmvc-3.0.5.RELEASE.jar

stax2-api-3.1.1.jar

stax-api-1.0.1.jar

struts2-core-2.2.1.jar

struts2-spring-plugin-2.2.1.jar

woodstox-core-asl-4.1.4.jar

wsdl4j-1.6.2.jar

xmlschema-core-2.0.3.jar

xwork-core-2.2.1.jar

以及项目主要包：

rms-0.1.2-SNAPSHOT.jar,rmsClient-0.0.1-SNAPSHOT.jar,arch4.jar,util-1.0.0-SNAPSHOT.jar

如果为maven工程则添加以下依赖即可

|  |
| --- |
| <!-- struts2 begin -->  <dependency>  <groupId>org.apache.struts</groupId>  <artifactId>struts2-core</artifactId>  <version>2.2.1</version>  </dependency>  <dependency>  <groupId>org.apache.struts</groupId>  <artifactId>struts2-spring-plugin</artifactId>  <version>2.2.1</version>  </dependency>  <!-- struts2 end -->  <dependency>  <groupId>javax.servlet.jsp</groupId>  <artifactId>jsp-api</artifactId>  <version>2.1</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>net.sf.json-lib</groupId>  <artifactId>json-lib</artifactId>  <classifier>jdk15</classifier>  </dependency>  <dependency>  <groupId>org.mortbay.jetty</groupId>  <artifactId>jetty</artifactId>  <version>6.1.26</version>  </dependency>  <dependency>  <groupId>org.mortbay.jetty</groupId>  <artifactId>jsp-2.1-jetty</artifactId>  <version>6.1.26</version>  </dependency>  <dependency>  <groupId>cglib</groupId>  <artifactId>cglib</artifactId>  <version>2.2.2</version>  </dependency>  <dependency>  <groupId>asm</groupId>  <artifactId>asm</artifactId>  <version>3.3.1</version>  </dependency>  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-api</artifactId>  <version>2.6.2</version>  </dependency>  <dependency>  <groupId>org.oracle</groupId>  <artifactId>ojdbc14</artifactId>  <version>10.2.0.2.0</version>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <exclusions>  <exclusion>  <groupId>xml-apis</groupId>  <artifactId>xml-apis</artifactId>  </exclusion>  </exclusions>  <version>3.3.2.GA</version>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-entitymanager</artifactId>  <version>3.3.2.GA</version>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-annotations</artifactId>  <version>3.4.0.GA</version>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-ehcache</artifactId>  <version>3.3.2.GA</version>  </dependency> |

|  |
| --- |
| <dependency>  <groupId>com.sinosoft.one</groupId>  <artifactId>rms</artifactId>  <version>0.1.2-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.sinosoft.one</groupId>  <artifactId>util</artifactId>  <version>1.0.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.sinosoft.one</groupId>  <artifactId>rmsClient</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency> |

## 基本资源及代码

完成以上工作后，向项目中添加以下java类（struts2-spring-hibernate）

**Com.sinosoft.ebusiness.rms.web包下**

EmployeesConfigAction.java

GroupAction.java

LoginAction.java

RoleAction.java

TaskAction.java

TaskAuthAction.java

**Com.sinosoft.ebusiness.rms.util包下**

Item.java

Tree.java

以及资源文件，以下文件仅列出必要的配置

**applicationContext-shiroURL.xml**

1. **<bean** id="loginFilter" class=" com.sinosoft.one.rms.client.shiro.LoginFilter "**>**
2. **<property** name="securityManager" ref="securityManager"**/>**
3. **<property** name="loginUrl" value="/login.jsp"**/>**
4. **<property** name="successUrl" value="/home.jsp"**/>**
5. **<property** name="unauthorizedUrl" value="/unauthorized.jsp"**/>** --**>**
6. **<property** name="filters"**>**
7. **<map>**
8. **<enty** key="loginfilter" value-ref="loginfilter"**/>**
9. **</map>**
10. **</property>**
11. **<property** name="filterChainDefinitions"**>**
12. **<value>**
13. /=anon
14. /services/RMSWebService\*\*=anon
15. /rms/login/checkUser\*\* =anon
16. /rms/login/\*\* = loginfilter
17. /js/\*\* = anon
18. /css/\*\* = anon
19. /common/\*\* = anon
20. /images/\*\* = anon
21. <!--页面控制配置开始-->
23. <!--页面控制配置结束-->
24. /\*\* = loginfilter
25. **</value>**
26. **</property>**
27. **</bean>**

**applicationContext.xml**

1. <!-- 使用annotation 自动注册bean,并检查@Required,@Autowired的属性已被注入 -->
2. **<context:**component-scan base-package=**"**com.sinosoft**" />**
3. <!—以下接口为自实现-->
4. **<bean** id=**"**employeServiceInterface**"** class="com.sinosoft.one.rms.service.spring.EmployeServiceInterfaceImpl"**></bean>**
5. **<bean** id=**"**companyServiceInterface**"** class="com.sinosoft.one.rms.service.spring.CompanyServiceInterfaceImpl"**></bean>**

**applicationContext-action.xml>**

1. <!—如果class路径改变请确保路径正确-->
2. <bea**n** id=**"**loginAction**"** class="com.sinosoft.ebusiness.rms.web.LoginAction" **scope**=**"**prototype**"></bean>**
3. **<bean** id=**"**groupAction**"** class="com.sinosoft.ebusiness.rms.web.GroupAction" **scope**=**"**prototype**"></bean>**
4. **<bean** id=**"**roleAction**"** class="com.sinosoft.ebusiness.rms.web.RoleAction" **scope**=**"**prototype**"></bean>**
5. **<bean** id=**"**taskAction**"** class="com.sinosoft.ebusiness.rms.web.TaskAction" **scope**=**"**prototype**"></bean>**
6. **<bean** id=**"**taskAuthAction**"** class=" com.sinosoft.ebusiness.rms.web.TaskAuthAction" **scope**=**"**prototype**"></bean>**
7. **<bean** id=**"**employeesConfigAction**"** class="com.sinosoft.ebusiness.rms.web.EmployeesConfigAction" **scope**=**"**prototype**"></bean>**

**applicationContext-hibernate.xml**

1. **<bean** id=**"**sessionFactory**"** class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"**>**
2. **<property** name="packagesToScan"**>**
3. **<list>**
4. **<value>**com.sinosoft.one.\*.model**</value>**
5. **</list>**
6. **</property>**
7. **</bean>**

# 开发人员帮助

## Shiro权限控制

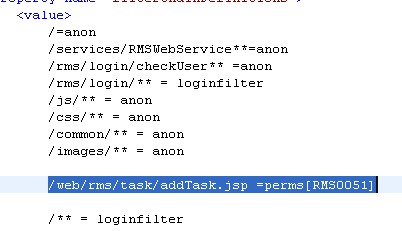
我们将权限控制分为3个部分，普通的jsp页面访问控制，页面元素访问控制，java代码访问控制

### jsp页面访问控制

对单独的JSP页面的访问控制进行配置。

当项目中需要控制某个页面不让其被直接于地址栏中进行访问时可以进行如下配置

在applicationContext-shiroURL.xml文件中id为shiroFilter的bean下filterChainDefinitions属性添加如下图配置



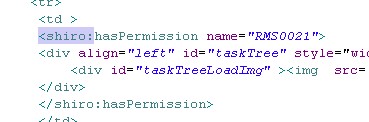
如图中蓝色选中部分，[]中RMS0051为具体的权限ID。

意思为：当用户具有ID编号为RMS0051的权限时可以直接访问/web/rms/task/addTask.jsp页面

### 页面元素访问控制

对页面Dom元素，如div、table、input等元素甚至JAVASPRIPT中的语句的访问控制进行配置。

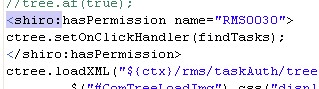
如图



表示：当具有ID编号为RMS0021的权限时可以访问标签内的DIV块



表示：当具有ID编号为RMS0022的权限时可以访问标签内的input标签



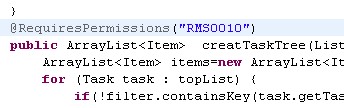
表示：当具有ID编号为RMS0030的权限时可以访问标签内的脚本语句

### Java代码访问控制

对java代码中的方法级别的代码块进行访问控制

当需要控制java的方法访问时可以在方法体上增加注解@RequiresPermissions("RMS0010")

如图



意思为：当具有ID编号为RMS0010的权限时可以访问被注解方法。

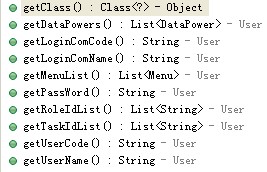
**注：其他更多详细配置可以参阅：详细控制配置.doc**

## 获取相关信息

当需要获取用户信息，如用户权限、角色、用户代码等信息时可以通过以下方式获得

User user=ShiroLoginUser.*getLoginUser*();

User对象封装了登陆对象的相关信息



## 相关接口

人员model接口：

EmployeModelInterface,项目中的人员实体需实现该接口

人员部分接口：

EmployeServiceInterface，提供相关的人员部分业务方法，根据需要自行实现。

机构model接口：

CompanyModelInterface,项目中的机构实体需实现该接口

机构部分接口：

CompanyServiceInterface，提供相关的机构部分业务方法，根据需要自行实现。

功能菜单部分接口：

TaskServcie,提供功能菜单管理相关的方法,spring bean id为rmsTaskServcie;

功能授权部分接口：

TaskAuthServcie,提供功能授权想干方法，Spring bean id为taskAuthServcie;

角色部分接口：RoleService,提供角色管理部分相关方法,Spring bean id 为roleServcie;

用户组部分接口：

GroupServcie,提供用户组管理部分相关方法，Spring bean id 为groupServcie;

人员配置相关方法：

RmsServcie，提供人员配置管理部分相关方法，spring Bean id为rmsServcie;

**注**：各接口详细方法说明见**权限API**。

# 主要业务操作说明

## 添加功能菜单

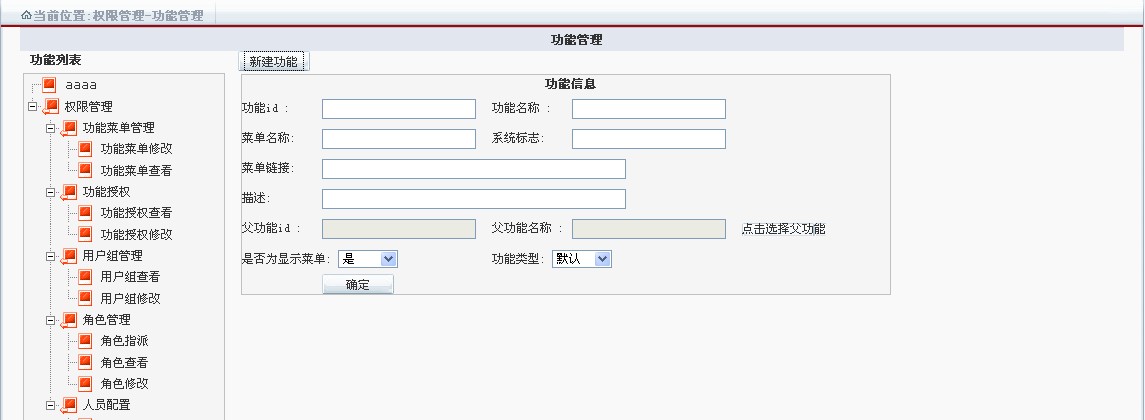
为系统添加新的功能，选择权限管理—功能菜单管理。选择后显示当前所有功能的信息。

如图：



点击新建功能按钮，显示功能信息输入框

如图：



各输入框说明：

功能ID：功能的编号，必须输入，输入后不可修改。

功能名称：功能的名称，非必须输入，可修改。

菜单链接：作为菜单时的连接，非必须输入，可修改。

菜单名称：作为菜单时显示的名称，非必须输入，可修改。

系统标志：为系统里的功能提供一个统一的标识，必须输入，可修改。（当控制多个系统时，由该标志标识系统功能，进行区别）

若需要将新建的功能放入已有的功能节点下，则点击选择父节点按钮，弹出显示新的功能树框。

如图：



点击选择需要加入的节点，将父功能带入输入框中，完成父功能的选择。

如图：



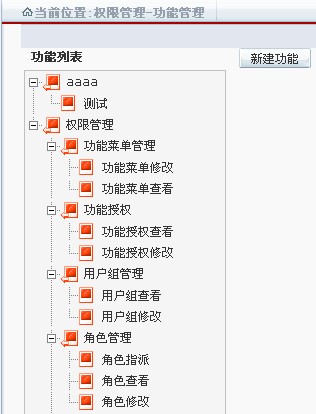
状态选择框：功能的使用状态是，使用中/不使用。

是否显示为菜单：是否将功能作为菜单显示，是/否。

功能类型：是否是所有级别的机构可以使用，所有可见:所有的机构可以使用不需要授权，默认：创建时的机构可见，下级机构需要授权可以使用。不可修改。

输入完信息后保存，左侧功能树刷新显示新建功能。

如图：



## 功能授权

为机构授权可使用的功能。

选择**权限管理—功能授权**。

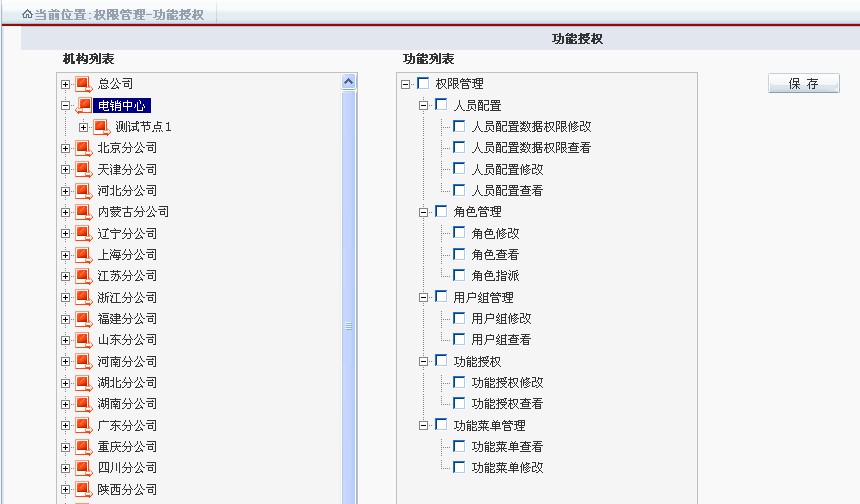
显示当前登陆用户所机构以下的子机构。

如图：



选择需要授权功能的机构节点，右边弹出该节点其父节点具有的功能，即任一节点的功能都从父节点而来（授权操作中，所授权的功能为3.1中配置的功能类型为‘默认’的功能，‘所有可见’类型的功能不需要授权。）

如图：

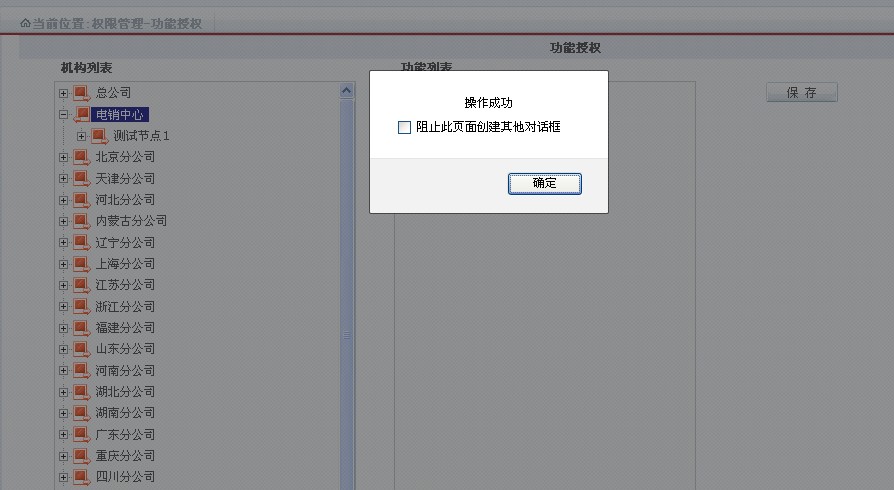


选择授权的功能后，点击保存，弹出成功提示后，完成授权操作。

如图：



如图：



## 角色管理

角色：角色定义为一个或多个功能的集合。

### 查询角色

查看系统已经创建的角色。

点击权限管理—角色管理菜单。

显示角色查询页面。直接显示系统已经创建的角色。

如图：

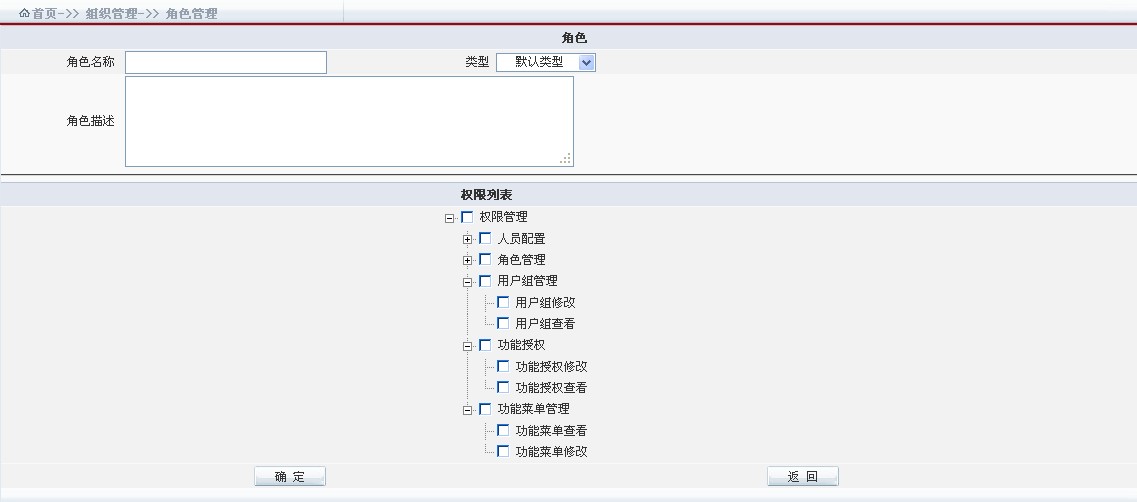


### 添加角色

为系统创建一个新的角色

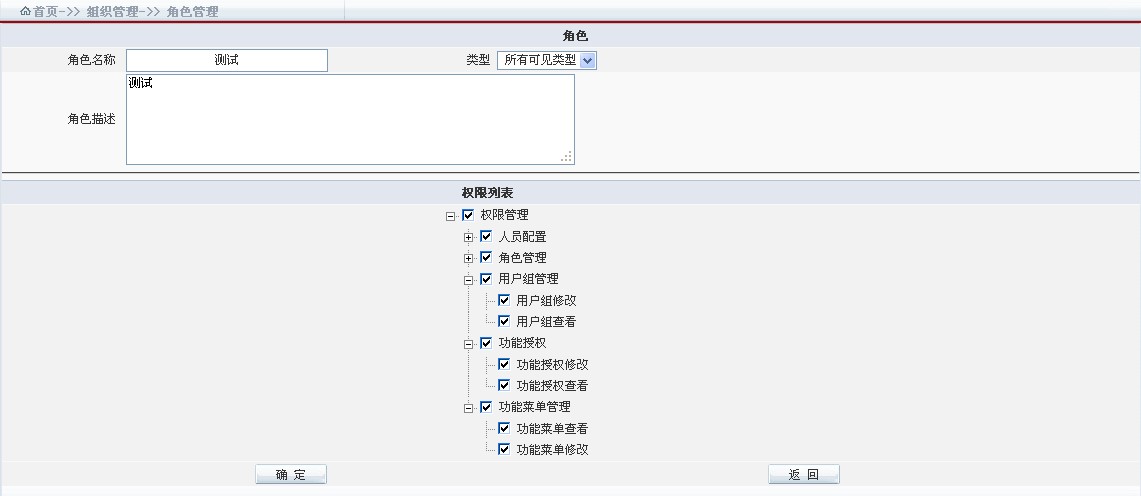
在3.3.1查询页面中选择增加按钮。跳转角色创建页面。

如图：



为角色添加基本信息：角色名称、角色描述、角色含有的功能（功能集合根据登陆用户所在的机构所具有的功能获得）。角色类型，角色类型分为默认及所有可见，默认类型为：所创建的角色只能在本机构使用，需要让其他机构使用需要进行角色指派操作。所有可见类型为：所创建的角色直接在任何机构中使用。

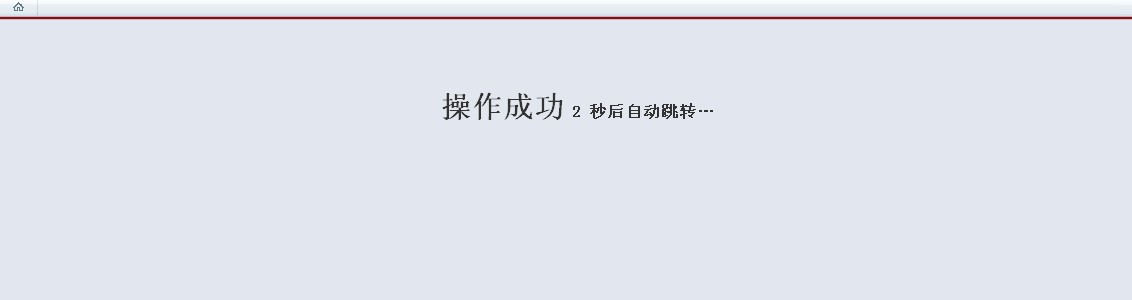
如图：



完成角色基本信息，点击保存后跳转成功页面，完成新建角色。之后返回角色查询页面。

如图：





### 修改与查看角色

为已经创建好的角色修改角色信息。

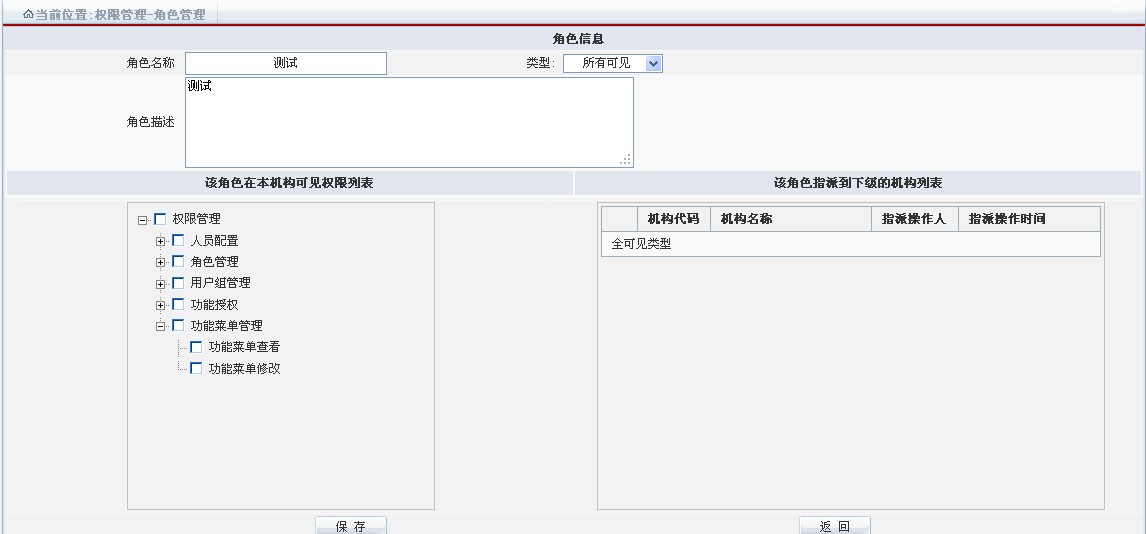
点击权限管理—角色管理菜单，进入查询页面，选择需要修改的角色点击修改按钮。

如图：

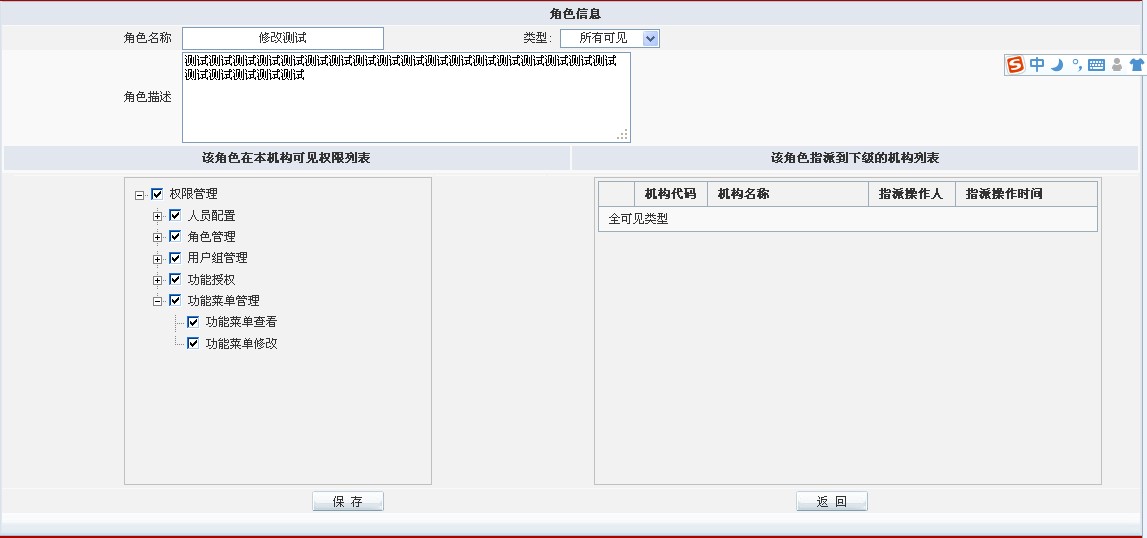


进入角色详细信息页面

如图：



用户修改角色信息时必须为与创建该角色时相同的机构，否则不能修改。即指派而来的角色无法修改。右下方将显示该角色已经指派到何机构列表，如果为所有可见则不显示。



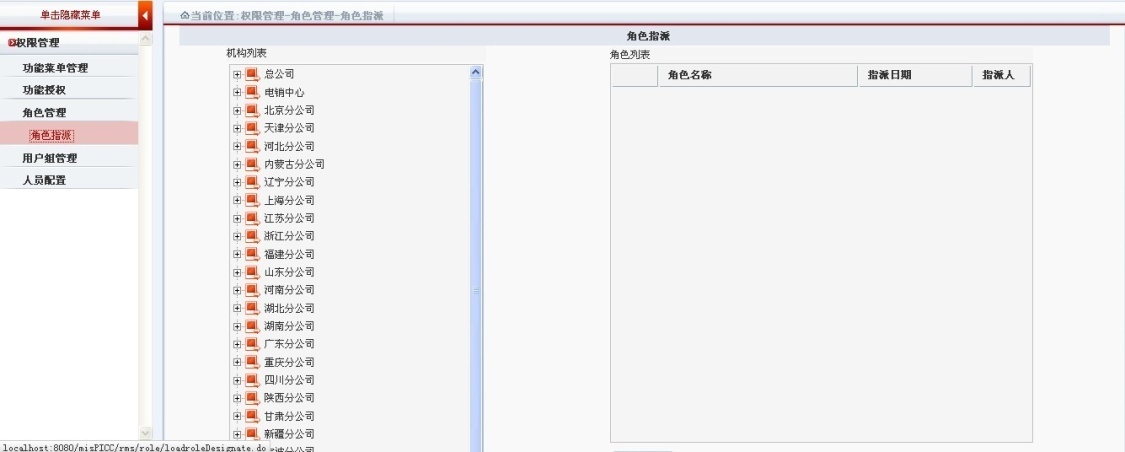
修改完成后点击保存，在弹出保存提示后，跳转操作成功页面完成操作。最后回到查询页面。

### 角色指派

为机构指派角色，使机构获得角色的使用权。

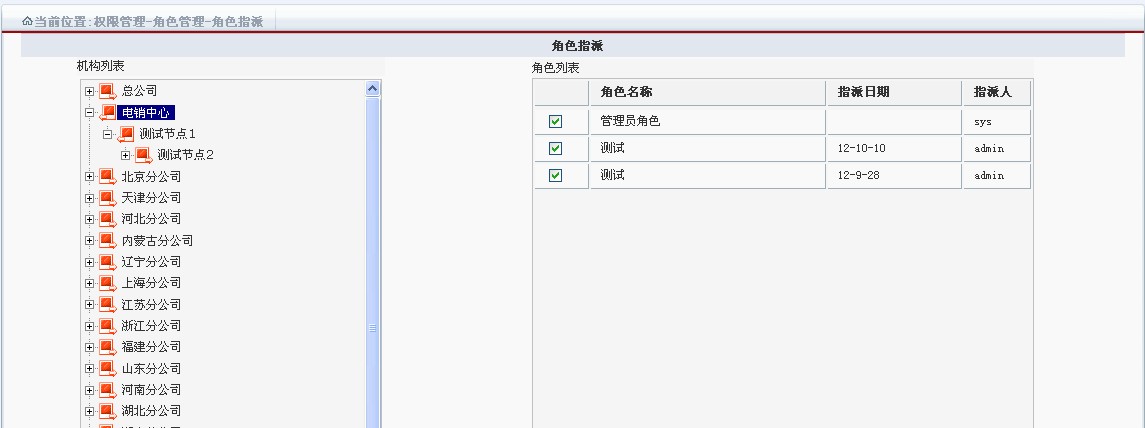
点击权限管理—角色管理—角色指派菜单，进入角色指派页面。显示当前登陆用户所在机构其下的子机构。

如图：



选择点击机构节点，右边显示该机构可以指派的角色列表（该列表为登陆用户所在机构中已有的角色），已选的为已经指派的角色。

如图：



**注**:其中角色类型为**所有可见**的则直接显示为已指派的选中状态。

当为某机构第一次进行角色指派后会为该机构创建一个**默认用户组**，并将该角色关联到改用户组中。如果已经创建默认用户组则不会再创建。



## 用户组管理

用户组：用户组为一个或者多个角色的集合，使组成员获得角色权限的单位。

### 查询角色

查看系统已经创建的用户组。

点击**权限管理—用户组管理**菜单。

显示角色查询页面。直接显示系统已经创建的角色。

如图：



### 添加用户组

为系统创建一个新的用户组。

在3.3.1查询页面中选择**增加按钮。**跳转用户组创建页面。

如图：



为角色添加基本信息：**用户组名称**、**用户组描述**、**用户组关联的的角色**（多选）。**用户组类型**，角色类型分为**默认**及**所有可见**，默认类型为：所创建的用户组只能在本机构使用。所有可见类型为：所创建的用户组直接在任何机构中使用。

如图：



### 修改与查看用户组

为已经创建好的角色修改角色信息。

点击**权限管理—角色管理**菜单，进入查询页面，选择需要修改的角色点击**修改**按钮。

如图：



进入用户组详细信息页面

如图：



点击管理用户组成员，弹出查询框，左边为已在用户组中的成员，可进行删除引入操作。右边框为未在用户组中的成员，可进行引入操作，执行引入用户组操作后人员直接在所在机构下获得权限。

如图：



修改完成后点击保存，在弹出保存提示后，跳转操作成功页面完成操作。最后回到查询页面。

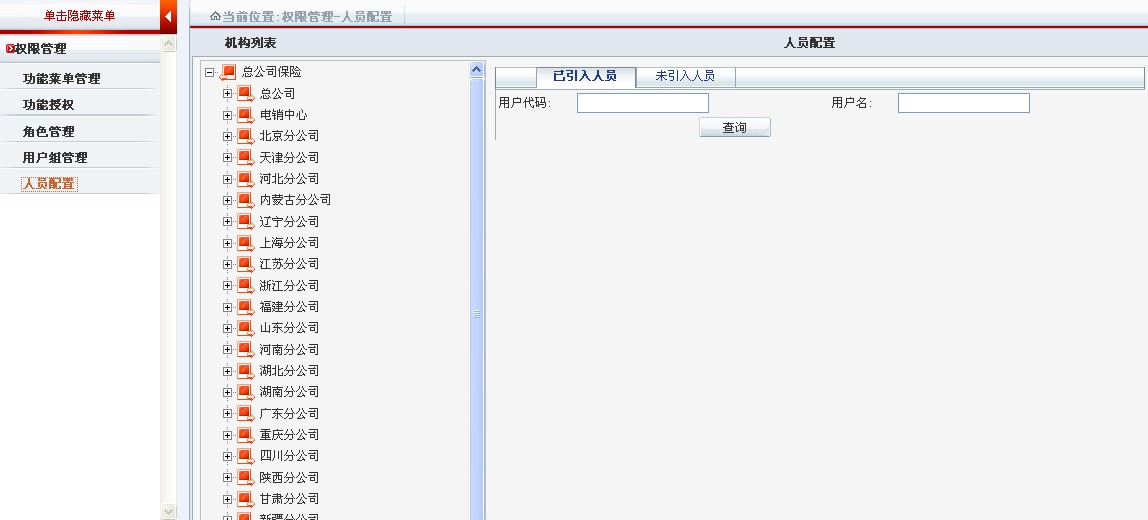
## 人员配置

未人员配置具体的机构权限。

点击权限管理—人员配置菜单。

显示人员配置页面。

如图：



直接选择机构节点，查询已引入、未引入该机构的人员。

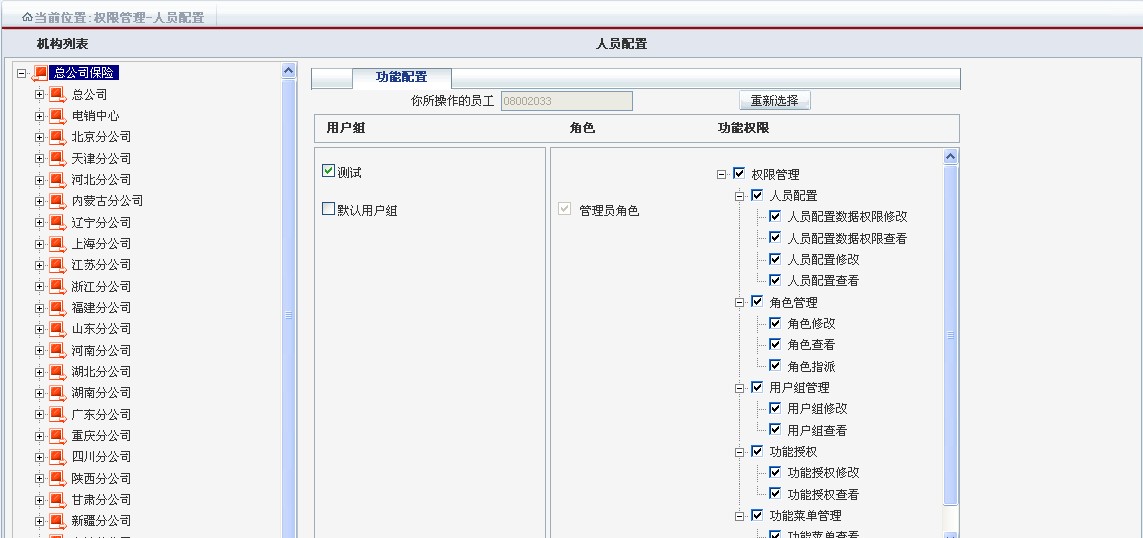
如图：





在未引入人员中，选择一位人员点击引入按钮，进入功能配置页面。

如图：



为用户选择用户组后，将显示用户所关联的角色与角色所关联的功能。

（此处的角色为用户组关联的角色，并且为该机构可见的角色，范围不超过全可见角色与指派于改机构的角色的总和。所显示的功能范围不超过授权于该机构的功能与所有可见的功能总和）

配置好人员功能与用户组信息后点击保存，完成配置。

