**权限管理系统开发说明**

2016.10.17

**一、概述**

新能源可视化基础平台的系统管理模块是整个系统的基础功能模块。它实现了系统对用户、角色、菜单以及权限的配置和管理。此篇归档说明首先说明系统管理模块各个子模块的功能实现脉络，然后简单介绍页面配置的过程，最后通过一个页面开发示例梳理新能源可视化平台页面开发的流程。

**二、系统管理模块**

新能源可视化平台的系统管理模块基于SSH技术框架实现。该模块包含四个子模块：用户管理、角色管理、菜单管理和权限管理。大致的实现思路是：

1.利用Hibernate技术在数据库中建立用户、角色、菜单以及权限的对象模 型；

2.使用Spring框架分别实现用户、角色、菜单和权限子模块的后台服务管理；

3.使用Struts技术实现系统前后台的交互。

**2.1用户管理**

(一)实现说明

1、创建对象模型。   
 在工程项目com.rbac.entity包中创建了SysAccount类，定义用户模型。在同级目录 下创建的SysAccount.hbm.xml文件，是SysAccount类映射到数据库的Hibernate映射文件，它实现了Java类和数据库表的映射关系。通过这层映射关系，可以直接使用Java对象实现对用户的各种操作，避免了烦杂的数据库层面处理。

2、实现用户管理服务。   
 这是整个用户管理子模块的核心功能。首先，在工程项目com.rbac.dao包中定义AccountDao类(Data Access Object，数据访问对象之意，Java EE模式下对数据源访问操作封装的接口)，实现对用户数据源的访问及相应的操作之功能；然后，在工程项目的com.rbac.service包中实现AccountService类，AccountService类中代码段：

1. @Autowired
2. private AccountDao accountDao;

实现了AccountDao对象的自动装载，至此，我们就可以通过AccountDao对象，实现对用户对象的查询、创建、修改以及删除操作；最后，在配置文件中加入如下配置，将AccountDao类和AccountService类封装成JavaBean。

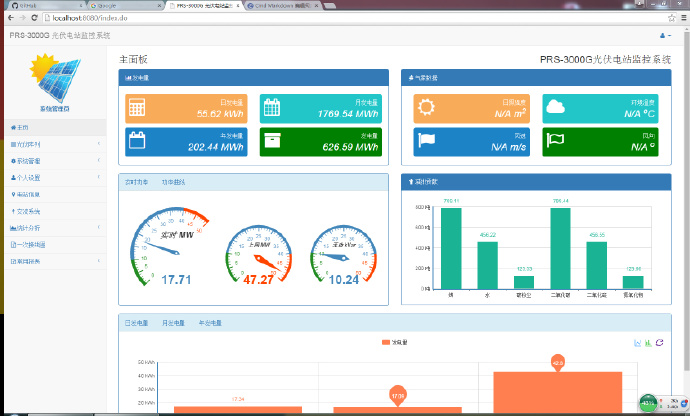
1. <!-- 自动将这些包下的类封装为spring bean -->
2. <context:component-scan base-package="com.rbac.dao">
3. </context:component-scan>
4. <context:component-scan base-package="com.rbac.service">
5. </context:component-scan>

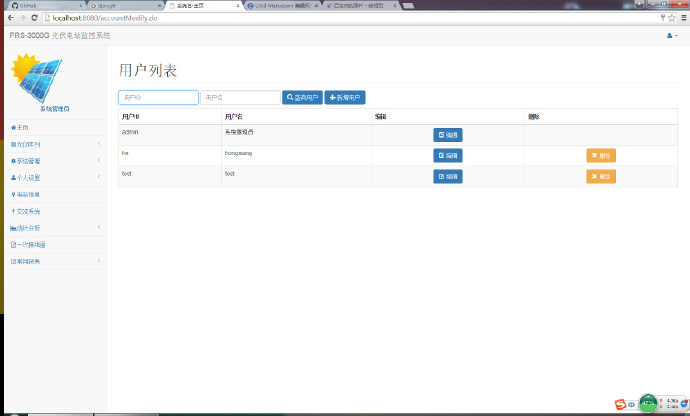
3、页面交互。   
 完成创建用户对象模型和用户管理服务，接下来需要实现在前台页面上对用户列表的显示和相应的维护功能，步骤如下：   
1）在com.rbac.form.system包中创建AccountListForm类和AccountModifyForm类，分别对应创建用户和修改用户时，前后台交互所要用到的用户form表单数据；   
2）在com.rbac.action.system包中创建AccountListAction类和AccountModifyAction类，实现前台和后台的交互。其中：AccountListAction类实现用户列表的显示和删除用户功能，AccountModifyAction类实现新增用户和修改用户的功能；   
3）在WebRoot目录下创建accountList.jsp和accountModify.jsp，作为用户管理子模块的用户列表页面和用户维护页面文件；   
4）在工程的struts-config.xml配置文件中，增加如下用于前后台交互的配置项：

1. <!-- 将Form类封装成form-bean -->
2. <form-bean name="accountListForm" type="com.rbac.form.system.AccountListForm"/>
3. <form-bean name="accountModifyForm" type="com.rbac.form.system.AccountModifyForm"/>
4. <!-- action各个属性的含义：
5. input: 输入action的jsp页面文件
6. name: action的名称
7. path: action的路径
8. scope: action在页面交互时所用的方法（request/response）
9. type: action的实现主体
10. -->
11. <action
12. input="/accountList.jsp"
13. name="accountListForm"
14. path="/accountList"
15. scope="request"
16. type="com.rbac.action.system.AccountListAction"/>
17. <action
18. attribute="accountModifyForm"
19. input="/accountModify.jsp"
20. name="accountModifyForm"
21. path="/accountModify"
22. scope="request"
23. type="com.rbac.action.system.AccountModifyAction">
24. <forward name="success" path="/accountList.do"/>
25. </action>

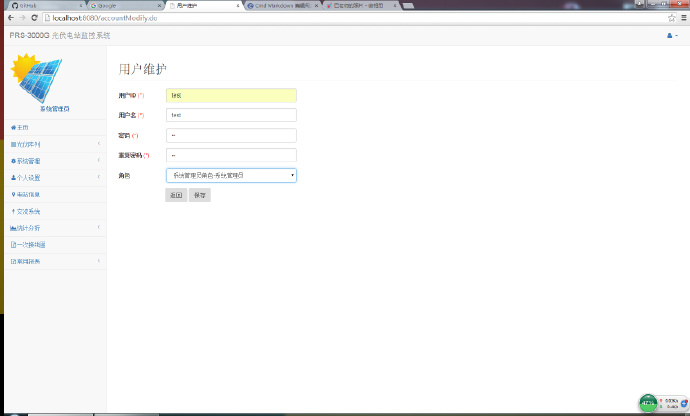
(2)配置说明

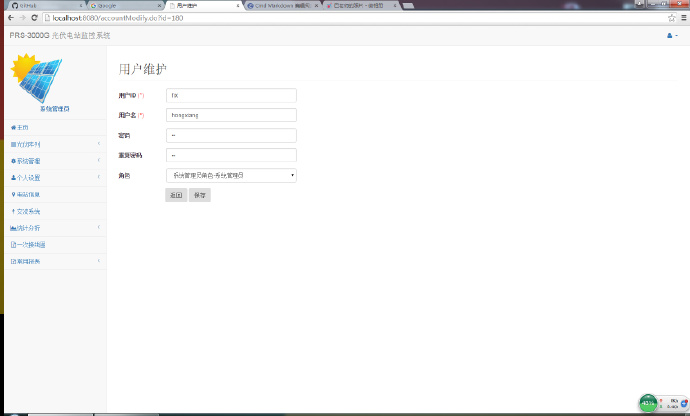
1、显示用户列表

启动，登录系统，主页面如下图所示：

点击系统管理菜单，选择用户管理，页面显示出当前的用户列表：

2、用户维护操作

点击新增用户按钮，输入用户信息，点击保存创建新用户，页面如下：

在用户列表中选择一个用户，点击编辑按钮，修改用户信息，点击保存修改用户，页面如下：

在用户列表中选择一个用户，点击删除按钮，点击提示框的确定按钮，直接删除用户。

**2.2角色管理**

(一)实现说明   
 角色管理的实现流程和用户管理类似，主要概括为以下三个步骤：

1、创建角色对象模型。   
com.rbac.entity包中的SysRole类定义了角色对象模型，通过Hibernate映射文件SysRoleMenu.hbm.xml，将角色对象模型映射到数据库。跟角色对象模型相关的还有com.rbac.entity包中的SysMenu类和它所对应的SysMenu.hbm.xml数据库映射文件，代表一个角色所关联的菜单对象列表。

2、实现角色管理服务。   
在com.rbac.dao包中定义了RoleDao类和MenuDao类，用于对角色对象关联的数据进行查询和相关操作。com.rbac.service包中的RoleService类，利用自动装配好的RoleDao和MenuDao对象（代码如下），实现对角色对象和角色所关联的菜单列表进行操作。

1. @Service("roleService")
2. public class RoleService {
3. @Autowired
4. private RoleDao roleDao;
5. @Autowired
6. private MenuDao menuDao;

3、页面交互。   
完成上述开发，就可以进行前台页面和后台的交互开发：   
1) 在com.rbac.form.system包中创建RoleListForm类和RoleModifyForm类，用于显示角色列表和添加、修改角色时角色的form表单数据；   
2）在com.rbac.action.system包中创建RoleListAction类和RoleModifyAction类，RoleListAction类用于从后台获取角色列表，发送到前台，并实现对角色的删除操作；RoleModifyAction类用于保存新添加以及修改的角色对象；   
3）在WebRoot目录下创建roleList.jsp和roleModify.jsp文件，在页面上实现角色列表的显示和角色的添加、删除、修改功能；   
4）在struts-config.xml配置文件中增加配置项参照用户管理子模块。

(二)配置说明

1、显示角色列表   
 启动，登录系统，展开系统管理菜单，选择角色管理，显示角色列表页面：

2、用户维护操作   
 点击新增角色按钮，出现以下页面。输入角色信息，选择角色关联的菜单树状列表，点击保存按钮保存新增的角色。   


在角色列表页面选择一个角色，点击编辑按钮，修改所选角色的信息，点击保存按钮保存修改。

在角色列表页面选择一个角色，点击删除按钮，弹出提示框，点击确定，直接删除所选的角色.

**2.3菜单管理**

实现流程以及页面配置方式和用户管理、角色管理类似，不再详细介绍。

**2.4权限管理**

实现流程以及页面配置方式和用户管理、角色管理类似，不再详细介绍。

**3、基于权限系统的页面开发实例**

下面通过一个实例来讲解基于权限系统的页面开发流程。

**3.1 功能需求**

系统菜单栏新增一个菜单，关联一个页面。打开此页面，显示当前时间。

**3.2 开发流程**

1、在工程项目中新建com.demo.form包和com.demo.action包，在两个包里面分别创建DemoForm类和DemoAction类，代码如下：

public class DemoForm extends BaseForm {

String time;

public String getTime() {

return time;

}

public void setTime(String time) {

this.time = time;

}

}

public class DemoAction extends BaseAction {

public ActionForward execute(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response) {

DemoForm demoForm = (DemoForm)form;

String time = demoForm.getTime();

if(time == null){

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");//设置日期格式

time = df.format(new Date());

}

request.setAttribute("time", time);

return mapping.getInputForward();

}

}

2、struts-config.xml文件中增加如下配置：

<!-- 将DemoForm类封装成名为demoForm的form-bean -->

<form-bean name="demoForm" type="com.demo.form.DemoForm"/>

<!--

attribute：指定action的From属性，即上述配置中的DemoForm

input: 指定action的输入页面文件为demo.jsp

name: 指定action的名称

path: 指定action的路径

scope: 指定action在页面交互时所用的方法为request

type: 指定action的实现主体为com.demo.action包中的DemoAction类

-->

<action

attribute="demoForm"

input="/demo.jsp"

name="demoForm"

path="/demo"

scope="request"

type="com.demo.action.DemoAction"/>

3、WebRoot中新增demo.jsp文件，代码如下：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!--页面字符集声明-->

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

<!--引入页面依赖的lib-->

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>

<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>

<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html lang="zh\_CN">

<head>

<title>开发示例</title>

<!--引入css层叠样式表-->

<link href="bower\_components/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">

<link href="bower\_components/metisMenu/dist/metisMenu.min.css" rel="stylesheet">

<link href="dist/css/sb-admin-2.css" rel="stylesheet">

<link href="bower\_components/font-awesome/css/font-awesome.min.css" rel="stylesheet" type="text/css">

<!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->

<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->

<!--[if lt IE 9]>

<script src="dist/js/html5shiv.min.js"></script>

<script src="dist/js/respond.min.js"></script>

<![endif]-->

</head>

<body>

<!--引入页面执行所需的js文件-->

<script src="bower\_components/jquery/dist/jquery.min.js"></script>

<script src="bower\_components/bootstrap/dist/js/bootstrap.min.js"></script>

<script src="bower\_components/metisMenu/dist/metisMenu.min.js"></script>

<script src="dist/js/sb-admin-2.js"></script>

<script src="dist/js/bootbox.min.js"></script>

<div id="wrapper">

<!--包含menu.jsp文件-->

<jsp:include page="meun.jsp"/>

<div id="page-wrapper">

<div class="row">

<div class="col-lg-12">

<!--将从后台取到的时间以h1标签显示在前台页面上-->

<h1 class="page-header">${time}</h1>

</div>

</div>

</div>

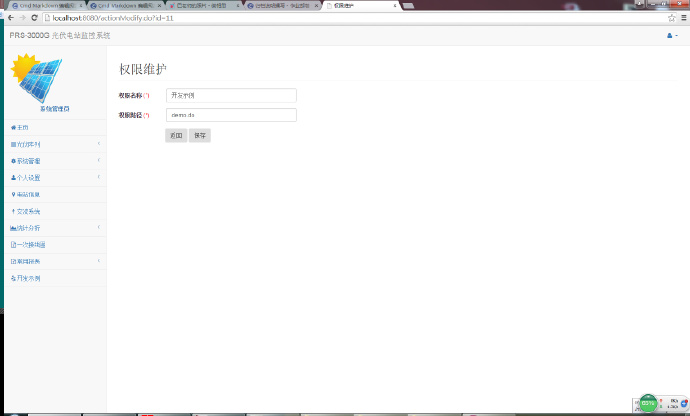
</div>

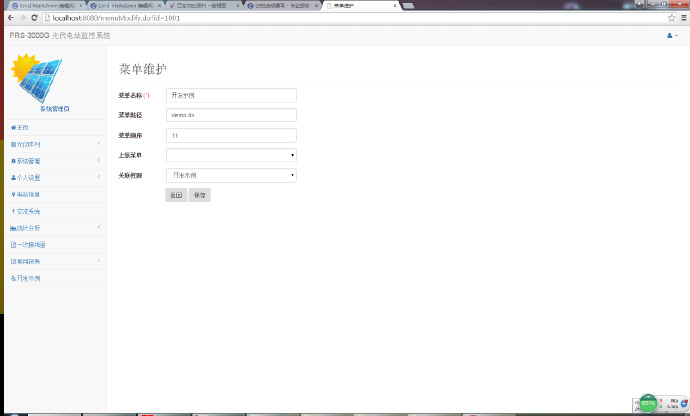
</body>

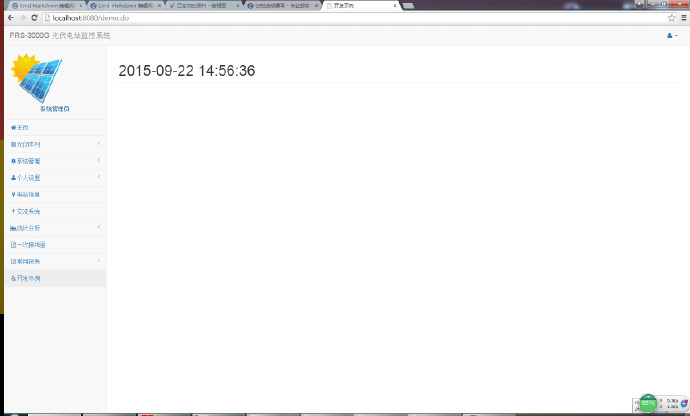
</html>

**4.新增权限和菜单**

(一)新增权限

在系统管理-权限管理页面中，点击新增权限按钮，新增开发示例权限，权限路径设为：demo.do，如下图所示：   


(二)新增菜单   
 在系统管理-菜单管理页面中，点击新增菜单按钮，新增开发示例菜单，菜单路径设为：demo.do，关联权限选择：开发示例权限，如下图所示：   


5、重启系统   
 重启系统，登录主页，左侧菜单栏出现：开发示例菜单，点击菜单，出现如下页面：   


页面中成功显示当前时间，这样就走通了整个开发流程。