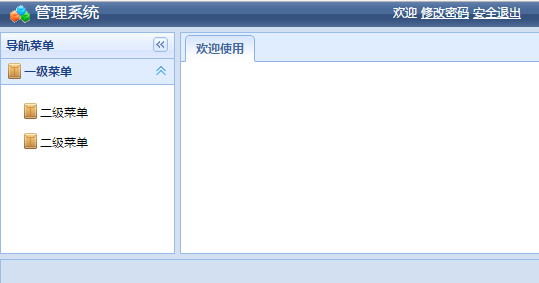
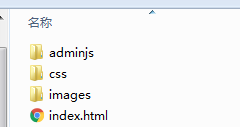
ERP项目笔记 Day-04

1. 登陆与密码管理
2. 主界面与登陆
   1. 主界面的构建

在网上找一款漂亮的后台模板

* + 1. 

将后台模板下所有的文件都复制到erp\_web的webapp目录下



* 1. 实现登陆功能
     1. 登陆页面

网上下载个漂亮的登陆页面，登陆页面的风格尽量与1.1中找到的后台模板的风格相同



把登陆页面里的3个文件/文件夹都复制到erp\_web的webapp下

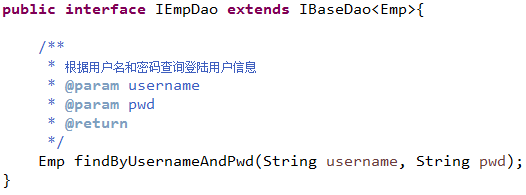


修改标题信息，修改login.html的title跟登陆框上的文字信息为“蓝云ERP旗舰版3.1”。

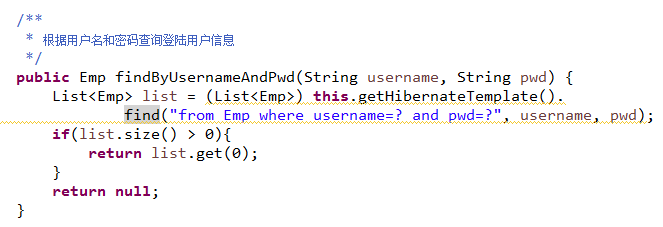
* + 1. 扩展IEmpDao和EmpDao方法

实现按用户名和密码查询实体类功能

Dao接口:

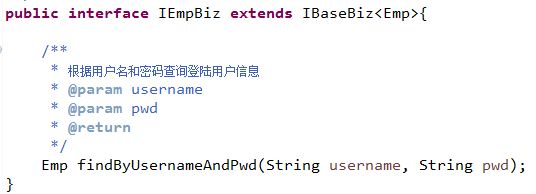


Dao实现:

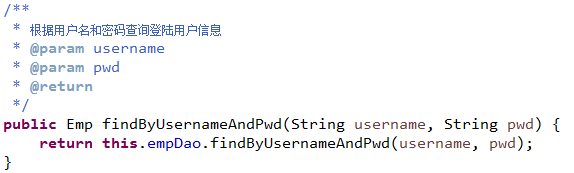


* + 1. 扩展IEmpBiz接口跟EmpBiz实现

IEmpBiz:



EmpBiz:



* + 1. 添加LoginAction

添加checkUser方法，用于登陆验证

|  |
| --- |
| **package** cn.itcast.erp.action;  **import** java.io.IOException;  **import** java.util.HashMap;  **import** java.util.Map;  **import** javax.servlet.http.HttpServletResponse;  **import** org.apache.struts2.ServletActionContext;  **import** com.alibaba.fastjson.JSON;  **import** com.opensymphony.xwork2.ActionContext;  **import** cn.itcast.erp.biz.IEmpBiz;  **import** cn.itcast.erp.entity.Emp;  /\*\*  \* 实现用户登陆与登出功能  \* **@author** Administrator  \*  \*/  **public** **class** LoginAction {  /\*\* 登陆用户名 \*/  **private** String username;  /\*\* 登陆密码 \*/  **private** String pwd;    **private** IEmpBiz empBiz;    /\*\*  \* 登陆验证请求  \*/  **public** **void** checkUser(){  **try**{  //验证登陆  Emp loginUser = empBiz.findByUsernameAndPwd(username, pwd);  **if**(**null** == loginUser){  ajaxReturn(**false**, "用户名或密码不正确");  **return**;  }  //保存到session中，表示用户已经登陆了  ActionContext.*getContext*().getSession().put("loginUser", loginUser);  ajaxReturn(**true**,"");  }**catch**(Exception ex){  ex.printStackTrace();  ajaxReturn(**false**,"登陆失败");  }  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getPwd() {  **return** pwd;  }  **public** **void** setPwd(String pwd) {  **this**.pwd = pwd;  }  **public** **void** setEmpBiz(IEmpBiz empBiz) {  **this**.empBiz = empBiz;  }    /\*\*  \* 返回前端操作结果  \* **@param** success  \* **@param** message  \*/  **public** **void** ajaxReturn(**boolean** success, String message){  //返回前端的JSON数据  Map<String, Object> rtn = **new** HashMap<String, Object>();  rtn.put("success",success);  rtn.put("message",message);  write(JSON.*toJSONString*(rtn));  }    /\*\*  \* 输出字符串到前端  \* **@param** jsonString  \*/  **public** **void** write(String jsonString){  **try** {  //响应对象  HttpServletResponse response = ServletActionContext.*getResponse*();  //设置编码  response.setContentType("text/html;charset=utf-8");  //输出给页面  response.getWriter().write(jsonString);  } **catch** (IOException e) {  e.printStackTrace();  }  }  } |

* + 1. 配置action

1. applicationContext-action.xml

|  |
| --- |
| <!-- 登陆/退出 -->  <bean id=*"loginAction"* class=*"cn.itcast.erp.action.LoginAction"* scope=*"prototype"*>  <property name=*"empBiz"* ref=*"empBiz"*></property>  </bean> |

1. Struts.xml

|  |
| --- |
| <!-- 登陆/退出 -->  <action name=*"login\_\*"* class=*"loginAction"* method=*"{1}"*></action> |

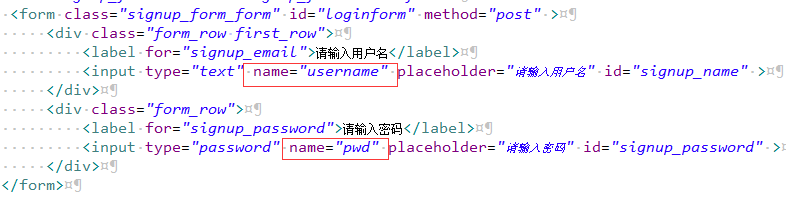
* + 1. 修改login.html

1. 修改文件名并添加easyui引用

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>蓝云ERP旗舰版3.1</title>  <link rel=*"stylesheet"* type=*"text/css"* href=*"css/login.css"*/>  <link rel=*"stylesheet"* type=*"text/css"* href=*"ui/themes/default/easyui.css"*>  <link rel=*"stylesheet"* type=*"text/css"* href=*"ui/themes/icon.css"*>  <script type=*"text/javascript"* src=*"ui/jquery.min.js"*></script>  <script type=*"text/javascript"* src=*"ui/jquery.easyui.min.js"*></script>  <script type=*"text/javascript"* src=*"ui/locale/easyui-lang-zh\_CN.js"*></script>  <script type=*"text/javascript"* src=*"ui/jquery.serializejson.min.js"*></script> |

1. 登陆表单form修改

给用户名输入框加入name=”username”，给密码输入框加入name=”pwd”



给“登陆”按钮加上id属性，用于登陆方法的绑定

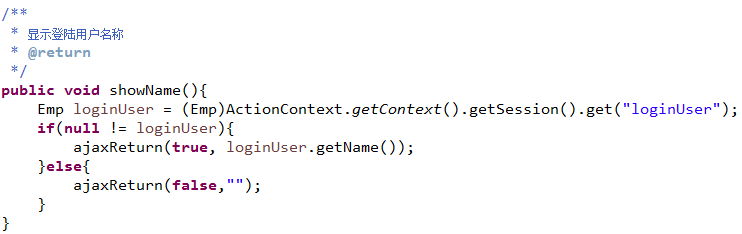


1. 编写ajax方法，实现用户登陆



* 1. 显示登陆用户名
     1. action修改

增加showName方法



* + 1. Index.html修改

添加<span>用于显示登陆用户名

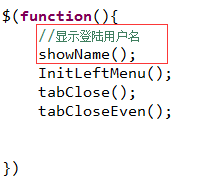


* + 1. index.js修改

加入showName方法

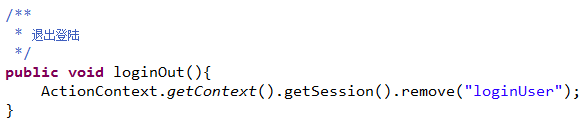
|  |
| --- |
| /\*\*  \* 显示登陆用户名  \*/  **function** showName(){  $.ajax({  url:'login\_showName',  method: 'post',  dataType: 'json',  success: **function**(rtn){  **if**(rtn.success){  //登陆成功的，则显示登陆用户名称  $('#username').html(rtn.message);  }**else**{  //没有登陆的，则跑转到登陆页面  location.href="login.html";  }  }  });  } |

在index.js的初始化方法中加入showName()方法调用



* 1. 退出登陆
     1. 修改action

LoginAction中加入loginOut方法



* + 1. 修改index.js

初始化方法中加入loginOut的绑定单击事件，该事件请求退出登陆

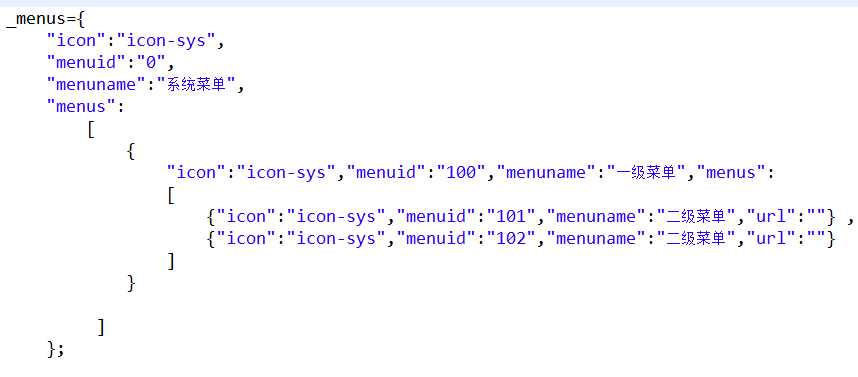
|  |
| --- |
| //退出登出  $('#loginOut').bind('click',**function**(){  $.ajax({  url: 'login\_loginOut',  success:**function**(rtn){  location.href="login.html";  }  });  }); |

1. 菜单动态读取（重点）

我们之前每做完一个页面，都需要在index.js中增加相应的菜单链接，才能让其在主页菜单中显示出来。有没有更好的实现？让它动态出现呢

* 1. 菜单表设计
     1. 菜单结构分析

分析index.js中的\_menus的json数据。可看出菜单里包含icon, menuid, menuname, url, menus数组



* + 1. 设计数据库菜单表

第个菜单都有一个上级菜单，我们可以通过层级结构进行设计。给每个菜单设置一个上级菜单(pid)，就可以实现无限级别的树形（层级）结构的菜单。

查看数据库模型图，



如果数据库还没有建表，则执行下面的建表语句：

|  |
| --- |
| **CREATE** **TABLE** MYERPUSER.MENU  (  MENUID **VARCHAR**(30) **NOT** **NULL PRIMARY KEY**,  MENUNAME **VARCHAR**(30) **NOT** **NULL**,  URL **VARCHAR**(200),  ICON **VARCHAR**(100),  PID **VARCHAR**(30) **NOT** **NULL**  ) TABLESPACE MYERP;  COMMENT **ON** **TABLE** MYERPUSER.MENU **IS** '系统菜单';  COMMENT **ON** **COLUMN** MYERPUSER.MENU.MENUID **IS** '编号';  COMMENT **ON** **COLUMN** MYERPUSER.MENU.MENUNAME **IS** '名称';  COMMENT **ON** **COLUMN** MYERPUSER.MENU.URL **IS** '对应URL';  COMMENT **ON** **COLUMN** MYERPUSER.MENU.ICON **IS** '图标样式';  COMMENT **ON** **COLUMN** MYERPUSER.MENU.PID **IS** '上一级菜单编号'; |

执行添加菜单脚本：

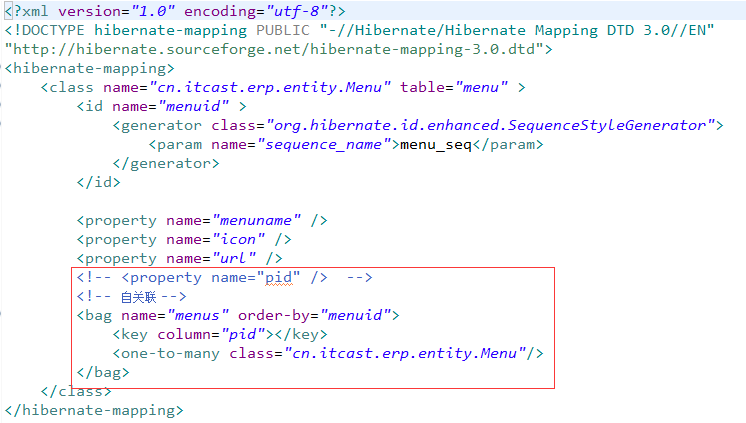
|  |
| --- |
| INSERT INTO MENU (MENUID, MENUNAME, URL, ICON, PID) VALUES ('0', '系统菜单', '-', 'icon-sys', '-1');  INSERT INTO MENU (MENUID, MENUNAME, URL, ICON, PID) VALUES ('100', '基础数据', '-', 'icon-sys', '0');  INSERT INTO MENU (MENUID, MENUNAME, URL, ICON, PID) VALUES ('101', '商品类型', 'goodstype.html', 'icon-sys', '100');  INSERT INTO MENU (MENUID, MENUNAME, URL, ICON, PID) VALUES ('102', '商品', 'goods.html', 'icon-sys', '100');  INSERT INTO MENU (MENUID, MENUNAME, URL, ICON, PID) VALUES ('105', '仓库', 'store.html', 'icon-sys', '100');  INSERT INTO MENU (MENUID, MENUNAME, URL, ICON, PID) VALUES ('200', '人事管理', '-', 'icon-sys', '0');  INSERT INTO MENU (MENUID, MENUNAME, URL, ICON, PID) VALUES ('201', '部门', 'dep.html', 'icon-sys', '200');  INSERT INTO MENU (MENUID, MENUNAME, URL, ICON, PID) VALUES ('202', '员工', 'emp.html', 'icon-sys', '200');  COMMIT; |

* 1. 代码实现
     1. 实现类

1. 实体类代码如下：

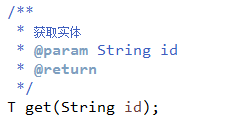
|  |
| --- |
| **package** cn.itcast.erp.entity;  **import** java.util.List;  /\*\*  \* 菜单实体类  \* **@author** Administrator \*  \*/  **public** **class** Menu {  **private** String menuid;//菜单ID  **private** String menuname;//菜单名称  **private** String icon;//图标  **private** String url;//URL  //private String pid;//上级菜单ID  **private** List<Menu> menus;// 子菜单列表  **public** String getMenuid() {  **return** menuid;  }  **public** **void** setMenuid(String menuid) {  **this**.menuid = menuid;  }  **public** String getMenuname() {  **return** menuname;  }  **public** **void** setMenuname(String menuname) {  **this**.menuname = menuname;  }  **public** String getIcon() {  **return** icon;  }  **public** **void** setIcon(String icon) {  **this**.icon = icon;  }  **public** String getUrl() {  **return** url;  }  **public** **void** setUrl(String url) {  **this**.url = url;  }  **public** List<Menu> getMenus() {  **return** menus;  }  **public** **void** setMenus(List<Menu> menus) {  **this**.menus = menus;  }  } |
|  |

1. 配置映射文件，添加自关联，使用order-by指定排序

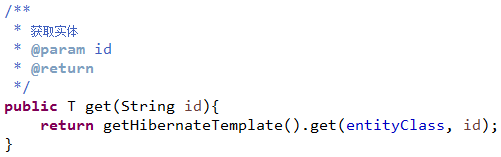


* + 1. Dao层
    2. 因为menu的主键为字符串，原来的get方法类型不能匹配，我们需要重载get方法，可以通过字符串主键获取实体

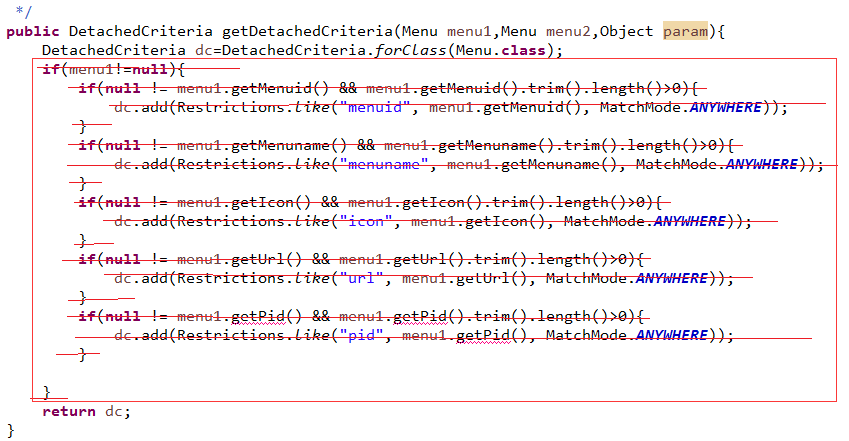
1. 修改IBaseDao，加入T get(String id)方法



1. 修改BaseDao，加入实现

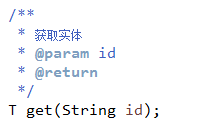


1. 删除MenuDao红框中的代码

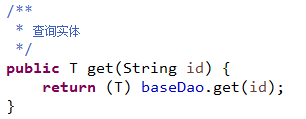


* + 1. Biz层

1. 修改IBaseBiz，添加T get(String id)方法

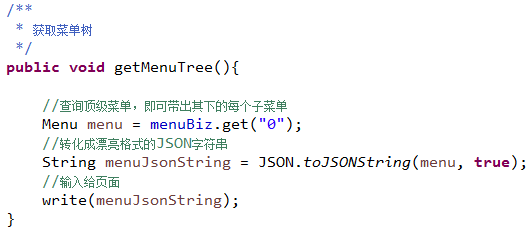


1. 修改BaseBiz，加入实现



* + 1. Action层

修改MenuAction，添加getMenuTree方法



0是顶级菜单的ID，通过自关联就可以把所有的菜单都加载进来。

浏览器访问：<http://localhost:8080/erp/menu_getMenuTree>

运行结果：



* + 1. 前端JS修改

修改index.js，使用ajax请求菜单地址来获取菜单数据



1. 密码加密
   1. 概述

我们在前期开发阶段，为了便于测试，密码采用明码存储。一旦程序部署到生产环境上，明友存储密码是非常不安全的，必须要对密码进行加密运算。

加密主要分为两种：可逆运算的和不可逆

1. 可逆运算的加密是通过一个秘钥，对一串字符进行加密运算，同样可以通过这个秘钥进行解密运算。
2. 不可逆运算的加密对一串字符进行加密，但是不能还原成原来的字符串。（散列）

那我们接下来做的加密就是使用不可逆加密的MD5加密算法

* 1. 添加shiro依赖

修改erp\_parent下的POM.xml文件，

定义shiro的版本号

|  |
| --- |
| <shiro.ver>1.2.3</shiro.ver> |

加入shiro的依赖

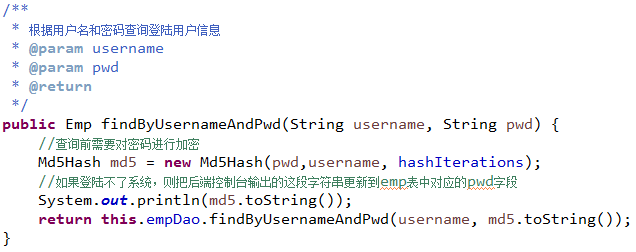
|  |
| --- |
| <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-core</artifactId>  <version>${shiro.ver}</version>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-web</artifactId>  <version>${shiro.ver}</version>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-spring</artifactId>  <version>${shiro.ver}</version>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-aspectj</artifactId>  <version>${shiro.ver}</version>  </dependency>  </dependencies> |

* 1. 代码实现

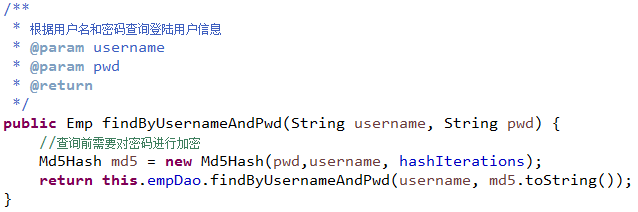
1. 修改EmpBiz类，添加属性



1. 修改EmpBiz类，重写add方法



1. 修改EmpBiz类的findByUsernameAndPwd方法，查询能对密码进行加密



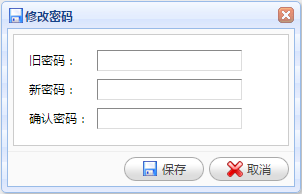
1. 修改密码
   1. 需求分析

每个员工都可以修改自己的密码，输入旧密码、新密码、确认密码后即可修改密码。

点击主页右上方 “修改密码”链接



弹出窗口



旧密码必须正确，新密码与确认密码必须相同，点击保存后更新密码

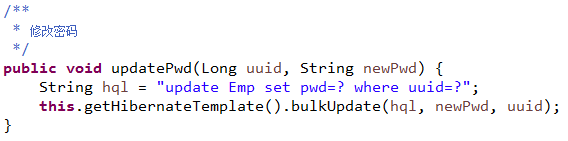
* 1. 后端实现

1. Dao修改，添加接口方法与实现

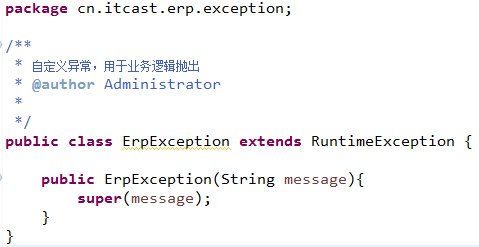
IEmpDao修改：



EmpDao实现：



1. 在erp\_biz中创建自定义异常cn.itcast.erp.ERPException，用于处理业务逻辑出错时抛出

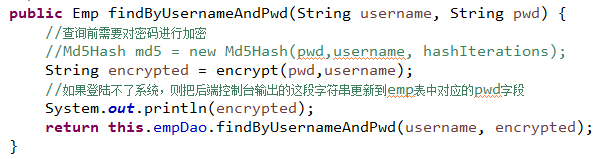


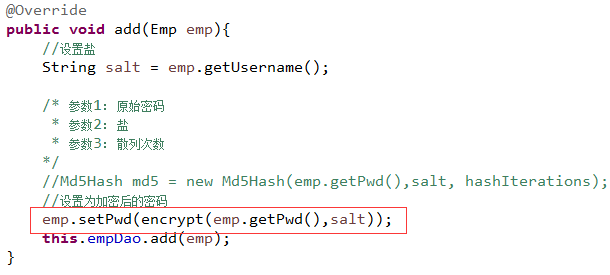
1. 修改EmpBiz

抽出加密方法，修改使用到密码加密的地方都调用该方法

|  |
| --- |
| /\*\*  \* MD5加密  \* **@param** src 原密码  \* **@param** salt 盐  \* **@return**  \*/  **private** String encrypt(String src, String salt){  Md5Hash md5 = **new** Md5Hash(src,salt, hashIterations);  **return** md5.toString();  } |

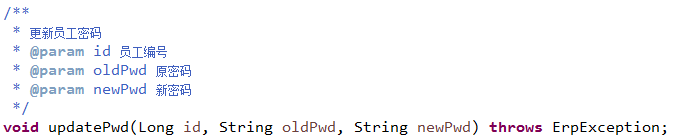
修改加密码加密的地方都改成调用加密方法：



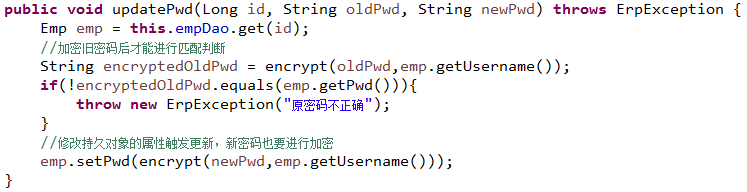


给IEmpBiz和EmpBiz增加修改密码的方法updatePwd

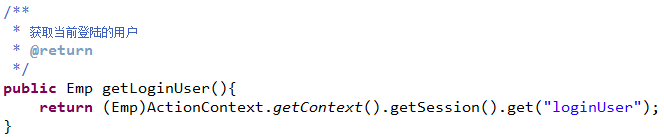
IEmpBiz



EmpBiz实现：

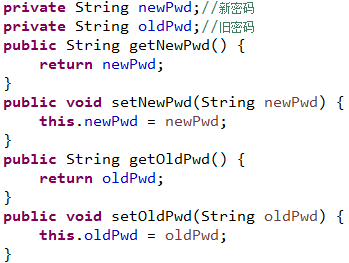


1. 在BaseAction中添加获取当前登陆用户的方法



1. 在EmpAction中添加调用修改密码的方法

添加接收新旧密码的属性：



添加updatePwd方法：



* 1. 前端实现
     1. 修改index.html

修改 修改密码窗口，改成如下：

|  |
| --- |
| <!--修改密码窗口-->  <div id=*"w"* class=*"easyui-dialog"*>  <div class=*"easyui-layout"* fit=*"true"*>  <div region=*"center"* border=*"false"* style="padding: *10px*; background: *#fff*;">  <table cellpadding=*3*>  <tr>  <td>旧密码：</td>  <td><input id=*"txtOldPass"* type=*"password"* class=*"txt01"* /></td>  </tr>  <tr>  <td>新密码：</td>  <td><input id=*"txtNewPass"* type=*"password"* class=*"txt01"* /></td>  </tr>  <tr>  <td>确认密码：</td>  <td><input id=*"txtRePass"* type=*"password"* class=*"txt01"* /></td>  </tr>  </table>  </div>  </div>  </div> |

* + 1. 修改index.js

修改openPwd()方法，把serverLogin的方法都复制到“保存”按钮的handler中，修改后的openPwd代码如下：

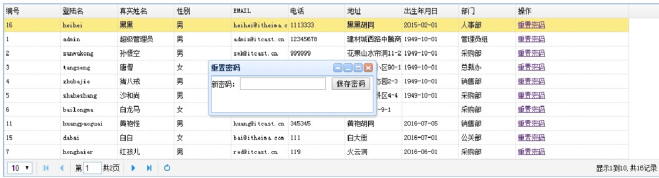
|  |
| --- |
| //修改密码  **function** openPwd() {  $('#w').dialog({  title: '修改密码',  width: 300,  modal: **true**,  closed: **true**,  height: 192,  buttons:[  {text:'保存',iconCls:'icon-save',handler:**function**(){  **var** $oldpass = $('#txtOldPass');  **var** $newpass = $('#txtNewPass');  **var** $rePass = $('#txtRePass');  **if** (!$oldpass.val()) {  $.messager.alert('提示信息','请输入原密码！','info',**function**(){  $oldpass.select();  })  **return** **false**;  }  **if** (!$newpass.val()) {  $.messager.alert('提示信息','请输入新密码！','info',**function**(){  $newpass.select();  })  **return** **false**;  }  **if** (!$rePass.val()) {  $.messager.alert('提示信息','请再一次输入密码！','info',**function**(){  $rePass.select();  })  **return** **false**;  }  **if** ($newpass.val() != $rePass.val()) {  $.messager.alert('提示信息','两次密码不一至！请重新输入','info',**function**(){  $rePass.select();  })  **return** **false**;  }  $.ajax({  url:'emp\_updatePwd',  dataType:'json',  type:'post',  data: {oldPwd:$oldpass.val(), newPwd:$newpass.val()},  success:**function**(rtn){  $.messager.alert('提示',rtn.message,'info',**function**(){  **if**(rtn.success){  $oldpass.val('');  $newpass.val('');  $rePass.val('');  $('#w').dialog('close');  }  });  }  });  }},  {text:'取消',iconCls:'icon-cancel',handler:**function**(){  $('#w').dialog('close');  }}  ]  });  } |

1. 管理员重置密码
   1. 需求分析

当员工忘记密码的时候，管理员可以重置员工的密码。

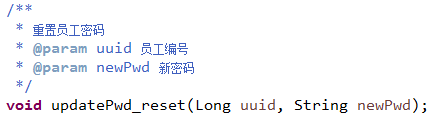
我们将员工的密码管理功能，从员工管理中独立出来

点击重置密码超链接，弹出窗口，输入新密码，保存更新密码后，关闭窗口

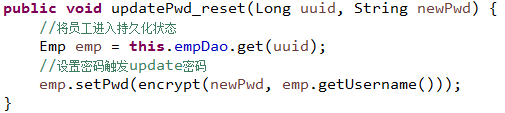


* 1. 后端实现
     1. Biz层修改

1. 修改IEmpBiz，增加updatePwd\_reset

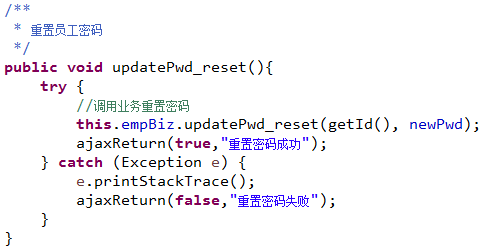


1. 修改EmpBiz，增加实现



* + 1. Action层修改

修改EmpAction，增加updatePwd\_reset方法，供页面重置密码时调用



* 1. 前端实现
     1. 新增pwd.html

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>  <title>重置员工密码</title>  <link rel=*"stylesheet"* type=*"text/css"* href=*"ui/themes/default/easyui.css"*>  <link rel=*"stylesheet"* type=*"text/css"* href=*"ui/themes/icon.css"*>  <script type=*"text/javascript"* src=*"ui/jquery.min.js"*></script>  <script type=*"text/javascript"* src=*"ui/jquery.easyui.min.js"*></script>  <script type=*"text/javascript"* src=*"ui/locale/easyui-lang-zh\_CN.js"*></script>  <script type=*"text/javascript"* src=*"ui/jquery.serializejson.min.js"*></script>  <script type=*"text/javascript"* src=*"ui/date.js"*></script>  <script type=*"text/javascript"* src=*"js/pwd.js"*></script>  </head>  <body>  <table id=*"grid"*></table>  <div id=*"editDlg"* class=*"easyui-dialog"* style="padding:*8px*;">  <form id=*"editForm"*>  <input name=*"id"* type=*"hidden"*>  <table>  <tr>  <td>新密码:</td>  <td>  <input id=*"txtpwd"* name=*"newPwd"* type=*"text"*>  </td>  </tr>  </table>  </form>  </div>  </body>  </html> |

* + 1. 增加pwd.js

在js目录下创建pwd.js

|  |
| --- |
| $(**function**(){  $('#grid').datagrid({  title: '员工信息列表',  url:'emp\_listByPage',  columns:[[  {checkbox:**true**},  {field:'uuid',title:'编号',width:100},  {field:'username',title:'系统登陆名称',width:100},  {field:'name',title:'真实姓名',width:100},  {field:'gender',title:'性别',width:100,formatter:**function**(gender){  **switch**(gender){  **case** 0: **return** '女';  **case** 1: **return** '男';  **default**: **return** '未知';  }  }},  {field:'email',title:'电子邮箱地址',width:100},  {field:'tele',title:'联系电话',width:100},  {field:'address',title:'联系地址',width:100},  {field:'birthday',title:'出生年月日',width:100,formatter:**function**(birthday){  **if**(birthday){  **return** **new** Date(birthday).Format("yyyy-MM-dd");  }  **return** "";  }},  {field:'dep',title:'部门',width:100,formatter:**function**(dep){  **if**(dep){  **return** dep.name;  }  **return** "";  }},  {field:'-',title:'操作',width:200,formatter:**function**(value,row,index){  **var** operation = '<a href="javascript:void(0)" onclick="updatePwd\_reset(' + row.uuid + ')">重置密码</a> ';  **return** operation;  }}  ]],  singleSelect: **true**,  pagination: **true**,  rownumbers: **true**  });  $('#btnSave').bind('click',**function**(){  **if**($('#formPwdReset').form('validate') == **false**){  **return**;  }  **var** datas = $('#formPwdReset').serializeJSON();  $.ajax({  url: 'emp\_updatePwd\_reset',  dataType:'json',  data: datas,  type: 'post',  success: **function**(rtn){  **if**(rtn.success){  $.messager.alert("提示信息", "重置密码成功!", 'info', **function**(){  //关闭窗口  $('#dlgPwdReset').window('close');  //刷新列表  $('#grid').datagrid('reload');  });  }**else**{  $.messager.alert("提示信息", "重置密码失败!<br>" + rtn.message, 'info');  }  }  });  });  $('#editDlg').dialog({  title:"重置员工密码",  width:260,  height:120,  iconCls:'icon-save',  modal:**true**,  closed:**true**,  buttons:[  {  text:'保存',  iconCls: 'icon-save',  handler:**function**(){  **var** datas = $('#editForm').serializeJSON();  $.ajax({  url: 'emp\_updatePwd\_reset',  dataType:'json',  data: datas,  type: 'post',  success: **function**(rtn){  $.messager.alert("提示", rtn.message, 'info', **function**(){  **if**(rtn.success){  //关闭窗口  $('#editDlg').dialog('close');  //刷新列表  $('#grid').datagrid('reload');  }**else**{  $.messager.alert("提示", "重置密码失败!<br>" + rtn.message, 'info');  }  });  }  });  }  }  ]  });  });  **function** updatePwd\_reset(uuid){  //清除表单内容，防止数据混乱  $('#editForm').form('load',{id:uuid,newPwd:""});  //弹出窗口  $('#editDlg').dialog('open');  } |

1. 员工管理去掉密码修改
   1. 需求分析

我们已经完成了专门的密码管理，那员工管理中的密码部分应该去掉，保证一个统一的入口。

* 1. 代码实现
     1. 修改emp.html

删除编辑员工中的密码编辑框



问题：当我们修改某个员工后，我们会发现，他的密码也变成空的，这是为什么？怎么解决？有以下3种解决方式

1. 使用触发器，更新的时候，如果密码为空就把旧的密码赋给它
2. 先让对象进入持久化状态，再设置它的属性值
3. Hbm.xml里加入update=false属性，不更新pwd字段（推荐）
   * 1. 后端修改
     2. 创建触发器

|  |
| --- |
| CREATE OR REPLACE TRIGGER TRI\_EMP\_UPDATE\_PWD  BEFORE  UPDATE OF PWD ON EMP FOR EACH ROW  DECLARE  *--声明变量*  BEGIN  *--如果修改的密码为NULL ，让修改的密码为原密码*  *--伪记录 :OLD 修改之前的记录 :NEW 修改之后的记录*  IF :NEW.PWD IS NULL THEN  :NEW.PWD := :OLD.PWD;  END IF;  END; |

* + 1. 设置持久化状态对象的属性

修改EmpBiz，重写update方法

|  |
| --- |
| @Override  **public** **void** update(Emp emp){  //查出要更新的对象，让其进入持久化状态  Emp dbEmp = **this**.empDao.get(emp.getUuid());  **if**(**null** == dbEmp){  **throw** **new** ERPException("没有找到相应的员工!");  }  //修改真实姓名  dbEmp.setName(emp.getName());  //修改联系电话  dbEmp.setTele(emp.getTele());  //修改联系地址  dbEmp.setAddress(emp.getAddress());  //修改出生年月日  dbEmp.setBirthday(emp.getBirthday());  //修改邮箱地址  dbEmp.setEmail(emp.getEmail());  //修改所属部门  dbEmp.setDep(emp.getDep());  } |

* + 1. Hbm.xml里加入update=false属性，不更新pwd字段

修改emp.hbm.xml文件:



1. 新增员工设置初始密码
   1. 需求分析

我们在新增员工的时候给他一个默认的密码，密码为登陆名

* 1. 实现

修改EmpBiz的新增员工方法



1. 静态页面传参
   1. 需求分析
2. 对于采购的订单，我们需要记录它是从哪一家供应商采购的；对于销售订单，我们也需要记录是卖给哪个客户的。因此我们需要管理供应商及客户的信息
3. 供应商信息管理都包括些什么？主键、名称、地址、联系人、电话等。客户信息管理又包括了哪些呢，也是主键、名称、地址、联系人、电话等。大家可以看到它们两个的字段其实是一样的。遇到这种两对不同类型的对象但其字段又相同的情况下，我们通常是把它们记录在同一个表里，用一个字段来区分。我们这里可以用type来进行区分，1：供应商，2：客户
   1. 实现思路

把相似的信息放入同一个表，可以极大的节省开发时间，降低维护成本，这两个信息对应的实体层、数据访问层、业务层、页面都用同一套代码，只是通过参数进行区分。

我们可以在管理供应商的页面进行修改，通过参数，将此功能一分为二。既可实现供应商管理，又能实现客户管理。

静态页面可传参？Of cause, 我们试试看吧

* 1. 代码实现
     1. 静态页面传参实现

先做个测试，给supplier.html页面传参数type

If(type==1){

页面标题=供应商管理

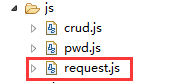
}

If(type==2){

页面标题=客户管理

}

将request.js复制到js目录下，并在supplier.html中加入引用



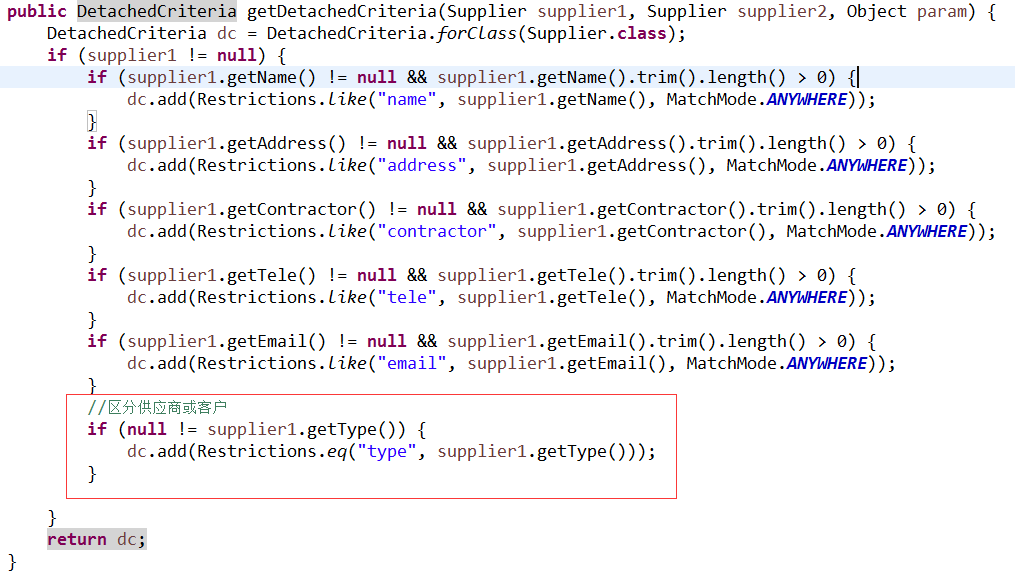
在supplier.html中添加script

|  |
| --- |
| **if**(Request['type']==1){  document.title="供应商管理";  }  **if**(Request['type']==2){  document.title="客户管理";  } |

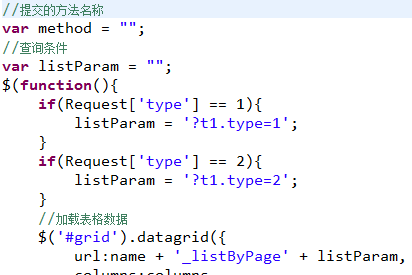
测试运行后发现，我们通过传参?type=value可以改变页面的标题

* + 1. 修改查询

1. 修改SupplierDao类，加上type条件，以区分查询的是供应商还是客户



1. 修改crud.js文件，加入全局变量listParam，根据type的值进入赋值



1. 删除原有type列或行

删除columns中的type列:



删除查询表单中type的条件

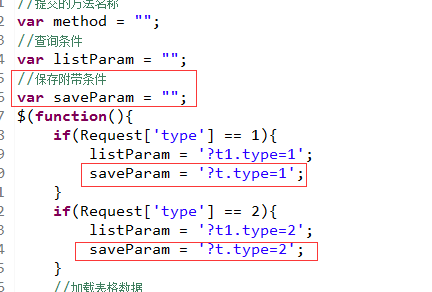


删除新增/修改窗口中type的显示

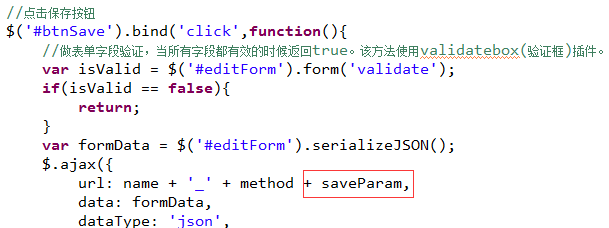


* + 1. 修改保存功能

增加saveParam变量，记录保存url提交的type参数



修改btnSave方法，提交的url中加入saveParam



* + 1. 修改菜单url

更新menu表中对应菜单的url，把供应商的url后面加上?type=1；把客户的url后面加上?type=2



打开chrome浏览器的开发工具，验证修改成功

1. 总结:
   1. 树状数据结构的表设计

在表中添加一个字段表示上一级ID，使用hibernate建立自关联

* 1. 使用shiro MD5对密码进行加密
  2. 静态页面传参

会使用request.js插件，实现静态页面的传参

* 1. 项目经验

1. 对两种或多种以上不同的对象，但又具有相同的属性字段，可以采用一个字段来区分，然后用相同的代码来实现功能。
2. 修改公用类或公用js文件时，一定要考虑当前的修改是否影响到其它功能。如果有影响应该怎么处理？项目开发中，往往是多人合作开发，会存在多人引用/修改同一个文件，因此在修改时应该要多考虑一下，尽量做到修改后不影响其它功能的正常运行。修改的同时记得加上注释。
3. 功能尽量单一化。即同一个功能，就只有一个入口。如：修改密码功能的修改，去掉了编辑员工时的密码修改功能，统一由重置功能来实现；新增时使用初始化密码来实现
4. 使用自定义异常的场景：用于业务逻辑校验不符合时抛出，action端要专门捕获这种已知的异常。
5. 对多次使用相同的代码块，应该抽出来独立成方法，用到它的地方都调用该方法。如：EmpBiz中的密码加密码方法encrypt，把散列的次数封装到encrypt方法里，保证每次调用后得到的结果都是相符合的。
6. 项目中用到的常量（数字或字符串），特别是用于判断或session.get/map.get，一定要封装到常量类中，并加上注释。常量可以在接口中定义