

**BOS物流项目**

# 教学计划

1、jQuery easyUI messager使用方式

* + alert方法
  + confirm方法
  + show方法

2、jQuery easyUI menubutton使用方式

3、自定义拦截器实现用户未登录自动跳转到登录页面

* + 创建拦截器
  + 注册拦截器

4、基于ajax实现用户修改密码功能

* + easyUI中validatebox校验功能
  + 发送ajax请求
  + 服务端修改数据库

5、基础设置部分需求分析

6、创建基础设置部分数据表

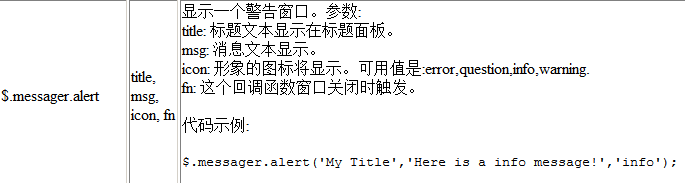
* + 取派员表
  + 区域表
  + 分区表
  + 定区表

7、取派员添加功能

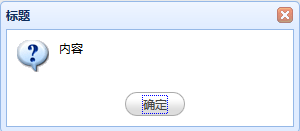
* + 扩展validatebox校验规则
  + 页面调整、提交表单
  + 服务端操作数据库

# jQuery easyUI messager使用方式

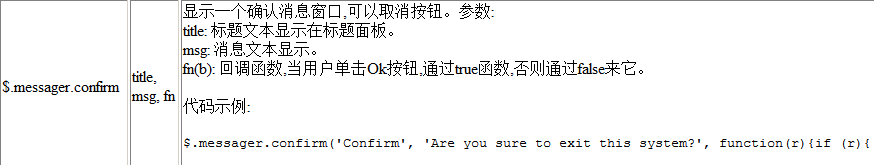
## alert

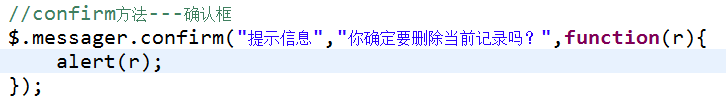


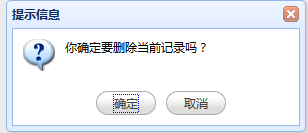




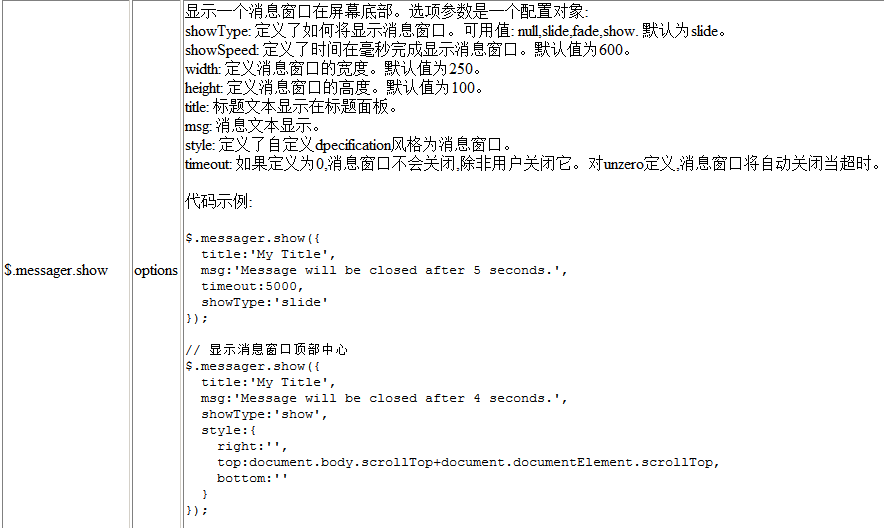
## confirm

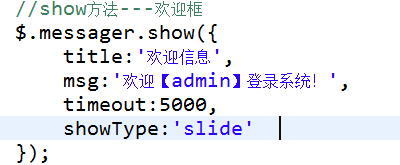


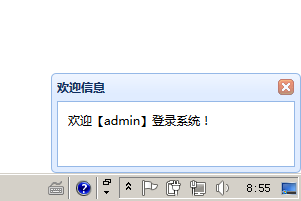




## show







# jQuery easyUI menubutton菜单使用

<!-- 制作菜单 -->

<a data-options=*"iconCls:'icon-help',menu:'#mm'"* class=*"easyui-menubutton"*>控制面板</a>

<!-- 使用div元素制作下拉菜单 -->

<div id=*"mm"*>

<div onclick="alert(1111)" data-options=*"iconCls:'icon-edit'"*>修改密码</div>

<div>联系管理员</div>

<div class=*"menu-sep"*></div>

<div>退出系统</div>

</div>



# 自定义拦截器实现用户未登录自动跳转到登录页面

## 编写拦截器类

需要继承struts2框架的MethodFilterInterceptor

提供工具类：

/\*\*

\* BOS项目的工具类

\* **@author** zhaoqx

\*

\*/

**public** **class** BOSUtils {

//获取session对象

**public** **static** HttpSession getSession(){

**return** ServletActionContext.*getRequest*().getSession();

}

//获取登录用户对象

**public** **static** User getLoginUser(){

**return** (User) *getSession*().getAttribute("loginUser");

}

}

/\*\*

\* 自定义的拦截器，实现用户未登录自动跳转到登录页面

\* **@author** zhaoqx

\*

\*/

**public** **class** BOSLoginInterceptor **extends** MethodFilterInterceptor{

//拦截方法

**protected** String doIntercept(ActionInvocation invocation) **throws** Exception {

//从session中获取用户对象

User user = BOSUtils.*getLoginUser*();

**if**(user == **null**){

//没有登录，跳转到登录页面

**return** "login";

}

//放行

**return** invocation.invoke();

}

}

## 配置拦截器

在struts.xml中配置

<interceptors>

<!-- 注册自定义拦截器 -->

<interceptor name=*"bosLoginInterceptor"* class=*"com.itheima.bos.web.interceptor.BOSLoginInterceptor"*>

<!-- 指定哪些方法不需要拦截 -->

<param name=*"excludeMethods"*>login</param>

</interceptor>

<!-- 定义拦截器栈 -->

<interceptor-stack name=*"myStack"*>

<interceptor-ref name=*"bosLoginInterceptor"*></interceptor-ref>

<interceptor-ref name=*"defaultStack"*></interceptor-ref>

</interceptor-stack>

</interceptors>

<default-interceptor-ref name=*"myStack"*/>

<!-- 全局结果集定义 -->

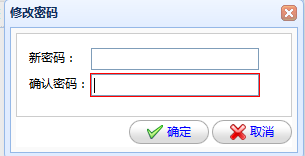
<global-results>

<result name=*"login"*>/login.jsp</result>

</global-results>

# 基于ajax实现用户修改密码功能

页面：WEB-INF/pages/common/index.jsp



## easyUI中validatebox使用

提供的校验规则：

1、非空校验required="required"

2、使用validType指定

email: 正则表达式匹配电子邮件规则。

url: 正则表达式匹配的URL规则。

length[0,100]: 在x和x字符允许。

remote['http://.../action.do','paramName']: 发送ajax请求做验证值,返回“true”当成功。

<table cellpadding=*3*>

<tr>

<td>新密码：</td>

<td><input required=*"true"* data-options=*"validType:'length[4,6]'"* id=*"txtNewPass"* type=*"Password"* class=*"txt01 easyui-validatebox"* /></td>

</tr>

<tr>

<td>确认密码：</td>

<td><input required=*"true"* data-options=*"validType:'length[4,6]'"* id=*"txtRePass"* type=*"Password"* class=*"txt01 easyui-validatebox"* /></td>

</tr>

</table>

## 为确定按钮绑定事件

//为确定按钮绑定事件

$("#btnEp").click(**function**(){

//进行表单校验

**var** v = $("#editPasswordForm").form("validate");

**if**(v){

//表单校验通过，手动校验两次输入是否一致

**var** v1 = $("#txtNewPass").val();

**var** v2 = $("#txtRePass").val();

**if**(v1 == v2){

//两次输入一致，发送ajax请求

$.post("userAction\_editPassword.action",{"password":v1},**function**(data){

**if**(data == '1'){

//修改成功，关闭修改密码窗口

$("#editPwdWindow").window("close");

}**else**{

//修改密码失败，弹出提示

$.messager.alert("提示信息","密码修改失败！","error");

}

});

}**else**{

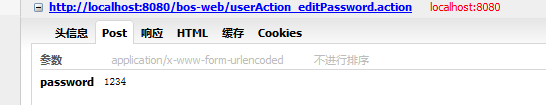
//两次输入不一致，弹出错误提示

$.messager.alert("提示信息","两次密码输入不一致！","warning");

}

}

});



## 服务端处理

第一步：在UserAction中创建修改密码方法editPassword

/\*\*

\* 修改当前用户密码

\* **@throws** IOException

\*/

**public** String editPassword() **throws** IOException{

String f = "1";

//获取当前登录用户

User user = BOSUtils.*getLoginUser*();

**try**{

userService.editPassword(user.getId(),model.getPassword());

}**catch**(Exception e){

f = "0";

e.printStackTrace();

}

ServletActionContext.*getResponse*().setContentType("text/html;charset=utf-8");

ServletActionContext.*getResponse*().getWriter().print(f);

**return** ***NONE***;

}

第二步：在UserService中提供修改密码方法

/\*\*

\* 根据用户id修改密码

\*/

**public** **void** editPassword(String id, String password) {

//使用MD5加密密码

password = MD5Utils.*md5*(password);

userDao.executeUpdate("user.editpassword", password,id);

}

第三步：在BaseDao中提供通用更新方法

//执行更新

**public** **void** executeUpdate(String queryName, Object... objects) {

Session session = **this**.getSessionFactory().getCurrentSession();

Query query = session.getNamedQuery(queryName);

**int** i = 0;

**for** (Object object : objects) {

//为HQL语句中的？赋值

query.setParameter(i++, object);

}

//执行更新

query.executeUpdate();

}

第四步：在User.hbm.xml中定义更新语句

<query name="user.editpassword">

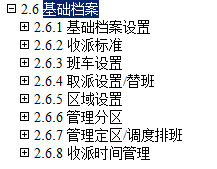
UPDATE User SET password = ? WHERE id = ?

</query>

# 基础设置部分需求分析

整个BOS项目分为基础设置、取派、中转、路由、报表、财务等部分。

基础设置部分对应需要规格说明书的2.6章节



## 基础档案设置

主要是针对本系统中的一些自定义项，需要参照录入，并作为统计分析和计算的维度，用户根据自己的需要动态设置的基础档案；对于自定义的档案支持多级定义；

本模块在其他的系统中也可以称为“数据字典”。



## 收派标准

将承接的货物，按照重量和体积两个标准，双纬度定义货物的标准。以便将不同标准的货物分给不同收取和派送能力的人员。由运营部门制定。

## 班车设置

将设置的线路和车辆建立对应关系。

## 取派员设置

小件员设置主要是设置小件员和快递员的资源信息，主要是为了自动下单和取派任务件使用；包括增加小件员的交通工具和通讯设备，以及取派的重量和体积标准。

使用角色为各级组织机构的系统管理人员在添加。

取派设置中包括小件员的替班信息设置。

以及被替班人信息的查询功能。

## 区域设置

此功能区域为行政区域列表，为价格、商务网站、工作单始发站、目的站等来源的基础数据。

## 管理分区

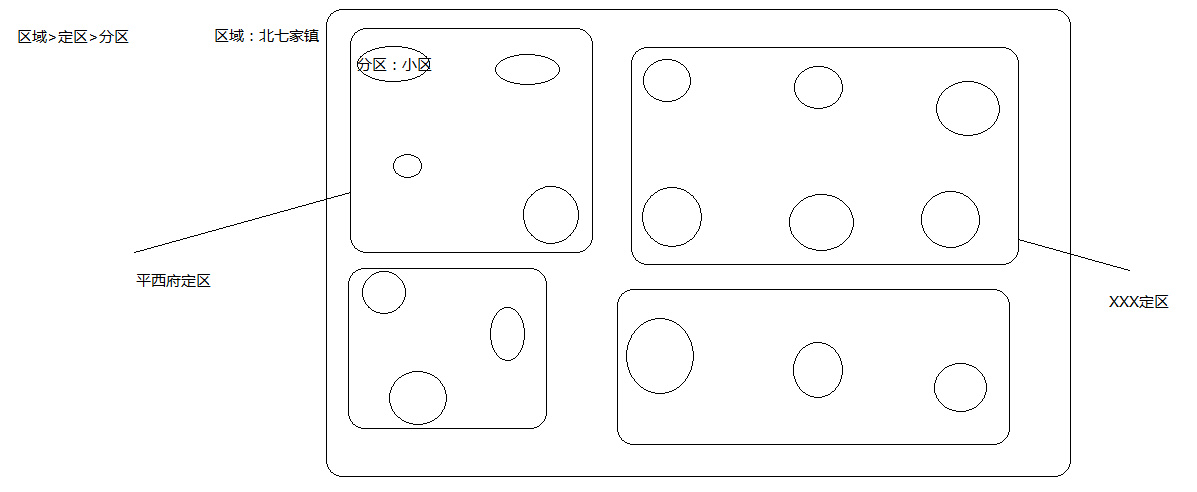
区域是由国家划分的行政区域，区域往往很大，不规则，不便于进行物人员分配。对于区域要进行细分，产生分区。

对管理分区进行维护,为定区管理,自动下单等后续操作提供基础数据支持。

打算引入一个复制功能按钮，用来增加信息基本相同的数据。

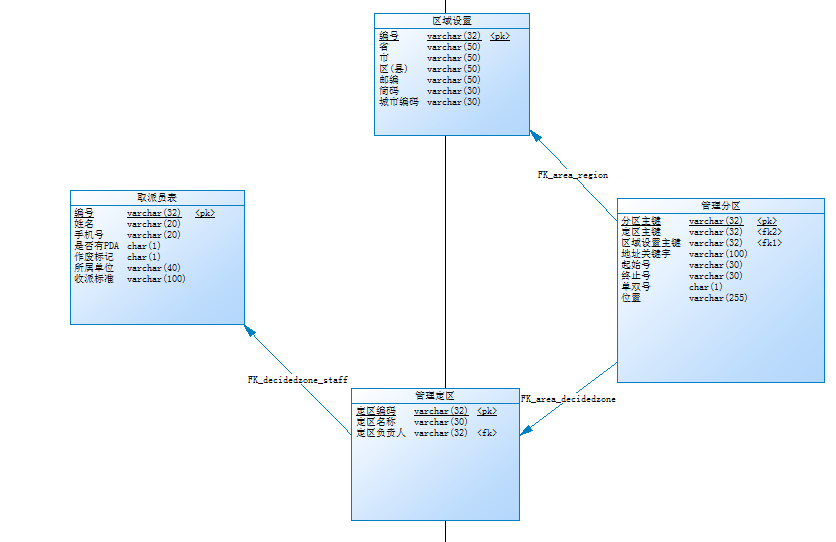
## 管理定区

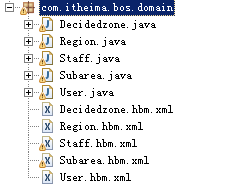
定区是物流分配的基本单位，可以将客户信息、取派员信息、分区信息进行关联，为自动分单提供基础数据。



## 时间管理

上班时间的管理,根据取派人员的可上班时间,将区域取派任务在不同时限分配给不同的取派人员。建议时间管理由总公司部门管理,固化出各种上班时间点。下级厅点只有调用权限,没有修改权限。





# 取派员添加功能

页面：WEB-INF/pages/base/staff.jsp

## 页面调整

第一步：扩展一个手机号的校验规则

<script type="text/javascript">

$(**function**(){

**var** reg = /^1[3|4|5|7|8][0-9]{9}$/;

//扩展手机号校验规则

$.extend($.fn.validatebox.defaults.rules, {

telephone: {

validator: **function**(value,param){

**return** reg.test(value);

},

message: '手机号输入有误！'

}

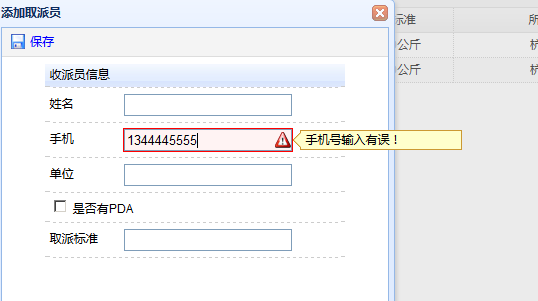
});

});

</script>

第二步：为手机号输入框应用规则





第三步：为保存按钮绑定事件，提交表单

//为保存按钮绑定事件

$("#save").click(**function**(){

//表单校验，如果通过，提交表单

**var** v = $("#addStaffForm").form("validate");

**if**(v){

//$("#addStaffForm").form("submit");

$("#addStaffForm").submit();

}

});



## 服务端实现

创建取派员对应的Action、Service、Dao