

**BOS物流项目**

# 教学计划

1、datagrid使用方法（重要）

* + 将静态HTML渲染为datagrid样式
  + 发送ajax请求获取json数据创建datagrid
  + 使用easyUI提供的API创建datagrid（掌握）

2、实现取派员分页查询

* + 调整页面基于datagrid发送ajax请求
  + 创建PageBean封装分页参数
  + 定义通用分页查询方法
  + 将分页查询结果转为json返回

3、取派员批量删除

* + 页面调整
  + 服务端实现

4、取派员修改

* + 页面调整
  + 服务端实现

# datagrid使用方法（重要）

## 将静态HTML渲染为datagrid样式

<!-- 方式一：将静态HTML渲染为datagrid样式 -->

<table class=*"easyui-datagrid"*>

<thead>

<tr>

<th data-options=*"field:'id'"*>编号</th>

<th data-options=*"field:'name'"*>姓名</th>

<th data-options=*"field:'age'"*>年龄</th>

</tr>

</thead>

<tbody>

<tr>

<td>001</td>

<td>小明</td>

<td>90</td>

</tr>

<tr>

<td>002</td>

<td>老王</td>

<td>3</td>

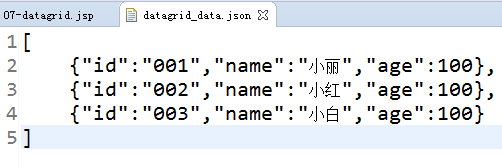
</tr>

</tbody>

</table>

## 发送ajax请求获取json数据创建datagrid

提供json文件：



<!-- 方式二：发送ajax请求获取json数据创建datagrid -->

<table data-options=*"url:'*${pageContext.request.contextPath }*/json/datagrid\_data.json'"*

class=*"easyui-datagrid"*>

<thead>

<tr>

<th data-options=*"field:'id'"*>编号</th>

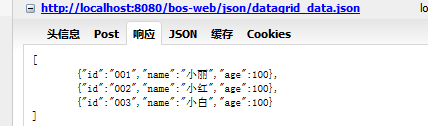
<th data-options=*"field:'name'"*>姓名</th>

<th data-options=*"field:'age'"*>年龄</th>

</tr>

</thead>

</table>



## 使用easyUI提供的API创建datagrid（掌握）

<!-- 方式三：使用easyUI提供的API创建datagrid -->

<script type=*"text/javascript"*>

$(**function**(){

//页面加载完成后，创建数据表格datagrid

$("#mytable").datagrid({

//定义标题行所有的列

columns:[[

{title:'编号',field:'id',checkbox:**true**},

{title:'姓名',field:'name'},

{title:'年龄',field:'age'},

{title:'地址',field:'address'}

]],

//指定数据表格发送ajax请求的地址

url:'${pageContext.request.contextPath }/json/datagrid\_data.json',

rownumbers:**true**,

singleSelect:**true**,

//定义工具栏

toolbar:[

{text:'添加',iconCls:'icon-add',

//为按钮绑定单击事件

handler:**function**(){

alert('add...');

}

},

{text:'删除',iconCls:'icon-remove'},

{text:'修改',iconCls:'icon-edit'},

{text:'查询',iconCls:'icon-search'}

],

//显示分页条

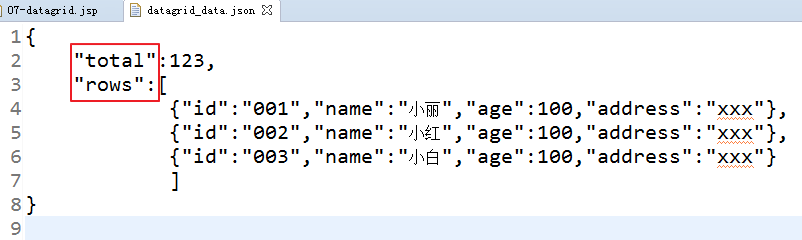
pagination:**true**

});

});

</script>

如果数据表格中使用了分页条，要求服务端响应的json变为：



请求：



响应：

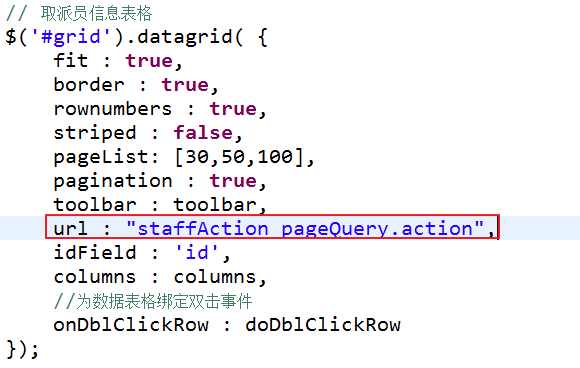


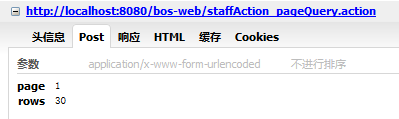
# 取派员分页查询

页面：WEB-INF/pages/base/staff.jsp

## 页面调整

* 修改页面中datagrid的URL地址

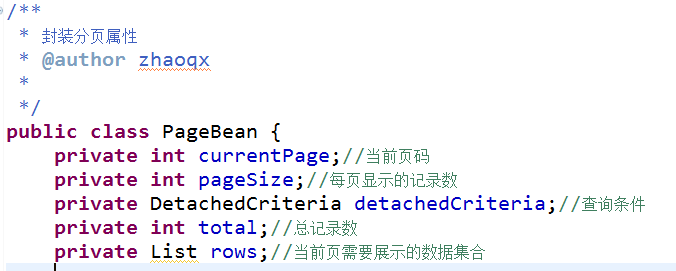




## 服务端实现

### 包装PageBean工具类

封装分页相关的属性



### 在BaseDao中扩展通用分页查询方法

/\*\*

\* 通用分页查询方法

\*/

**public** **void** pageQuery(PageBean pageBean) {

**int** currentPage = pageBean.getCurrentPage();

**int** pageSize = pageBean.getPageSize();

DetachedCriteria detachedCriteria = pageBean.getDetachedCriteria();

//查询total---总数据量

detachedCriteria.setProjection(Projections.*rowCount*());//指定hibernate框架发出sql的形式----》select count(\*) from bc\_staff;

List<Long> countList = (List<Long>) **this**.getHibernateTemplate().findByCriteria(detachedCriteria);

Long count = countList.get(0);

pageBean.setTotal(count.intValue());

//查询rows---当前页需要展示的数据集合

detachedCriteria.setProjection(**null**);//指定hibernate框架发出sql的形式----》select \* from bc\_staff;

**int** firstResult = (currentPage - 1) \* pageSize;

**int** maxResults = pageSize;

List rows = **this**.getHibernateTemplate().findByCriteria(detachedCriteria, firstResult, maxResults);

pageBean.setRows(rows);

}

### 在StaffAction中提供分页查询方法

//属性驱动，接收页面提交的分页参数

**private** **int** page;

**private** **int** rows;

/\*\*

\* 分页查询方法

\* **@throws** IOException

\*/

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

PageBean pageBean = **new** PageBean();

pageBean.setCurrentPage(page);

pageBean.setPageSize(rows);

//创建离线提交查询对象

DetachedCriteria detachedCriteria = DetachedCriteria.*forClass*(Staff.**class**);

pageBean.setDetachedCriteria(detachedCriteria);

staffService.pageQuery(pageBean);

//使用json-lib将PageBean对象转为json，通过输出流写回页面中

//JSONObject---将单一对象转为json

//JSONArray----将数组或者集合对象转为json

JsonConfig jsonConfig = **new** JsonConfig();

//指定哪些属性不需要转json

jsonConfig.setExcludes(**new** String[]{"currentPage","detachedCriteria","pageSize"});

String json = JSONObject.*fromObject*(pageBean,jsonConfig).toString();

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

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

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

}

# 取派员批量删除

在取派员表中存在一个删除标识位deltag，1表示已删除 ，0表示未删除

## 页面调整

* 数据表格datagrid提供的方法，用于获取所有选中的行：



修改删除按钮绑定的事件：

**function** doDelete(){

//获取数据表格中所有选中的行，返回数组对象

**var** rows = $("#grid").datagrid("getSelections");

**if**(rows.length == 0){

//没有选中记录，弹出提示

$.messager.alert("提示信息","请选择需要删除的取派员！","warning");

}**else**{

//选中了取派员,弹出确认框

$.messager.confirm("删除确认","你确定要删除选中的取派员吗？",**function**(r){

**if**(r){

**var** array = **new** Array();

//确定,发送请求

//获取所有选中的取派员的id

**for**(**var** i=0;i<rows.length;i++){

**var** staff = rows[i];//json对象

**var** id = staff.id;

array.push(id);

}

**var** ids = array.join(",");//1,2,3,4,5

location.href = "staffAction\_deleteBatch.action?ids="+ids;

}

});

}

}

## 服务端实现

第一步：在StaffAction中创建deleteBatch批量删除方法

//属性驱动，接收页面提交的ids参数

**private** String ids;

/\*\*

\* 取派员批量删除

\*/

**public** String deleteBatch(){

staffService.deleteBatch(ids);

**return** ***LIST***;

}

第二步：在Ｓｅｒｖｉｃｅ中提供批量删除方法

/\*\*

\* 取派员批量删除

\* 逻辑删除，将deltag改为1

\*/

**public** **void** deleteBatch(String ids) {//1,2,3,4

**if**(StringUtils.*isNotBlank*(ids)){

String[] staffIds = ids.split(",");

**for** (String id : staffIds) {

staffDao.executeUpdate("staff.delete", id);

}

}

}

第三步：在Staff.hbm.xml中提供HQL语句，用于逻辑删除取派员

<!-- 取派员逻辑删除 -->

<query name=*"staff.delete"*>

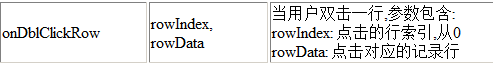
UPDATE Staff SET deltag = '1' WHERE id = ?

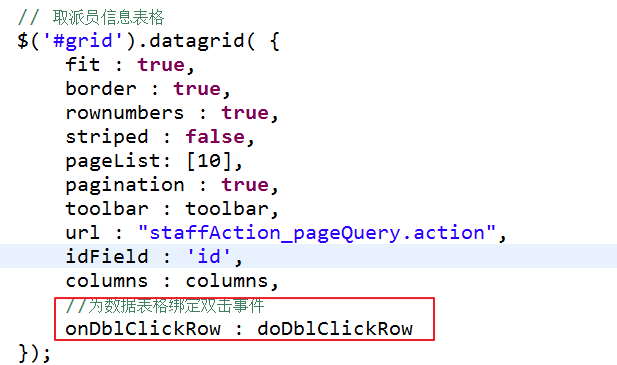
</query>

# 取派员修改功能

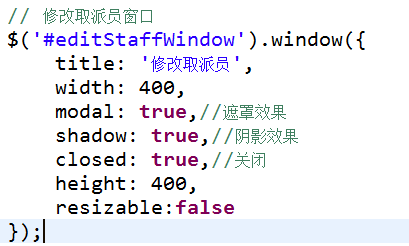
## 页面调整

第一步：为数据表格绑定双击事件





第二步：复制页面中添加取派员窗口，获得修改取派员窗口



第三步：定义function

//数据表格绑定的双击行事件对应的函数

**function** doDblClickRow(rowIndex, rowData){

//打开修改取派员窗口

$('#editStaffWindow').window("open");

//使用form表单对象的load方法回显数据

$("#editStaffForm").form("load",rowData);

}

## 服务端实现

在StaffAction中创建edit方法，修改取派员信息

/\*\*

\* 修改取派员信息

\*/

**public** String edit(){

//显查询数据库，根据id查询原始数据

Staff staff = staffService.findById(model.getId());

//使用页面提交的数据进行覆盖

staff.setName(model.getName());

staff.setTelephone(model.getTelephone());

staff.setHaspda(model.getHaspda());

staff.setStandard(model.getStandard());

staff.setStation(model.getStation());

staffService.update(staff);

**return** ***LIST***;

}