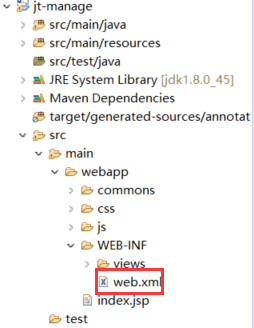
# 京淘后台展现

## 页面展现

### 导入静态资源文件

说明.将课前资料中的webapp文件导入,添加静态资源文件.但是注意web.xml使用自己编辑的.



### 实现首页跳转

@Controller

**public** **class** IndexController {

//跳转页面 返回值必须是string类型

//如果返回json 返回值类型一般是对象 @ResponseBody

@RequestMapping("/index")

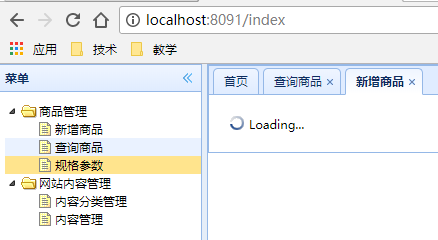
**public** String index() {

**return** "index"; //返回页面逻辑名称

}

}

### 页面跳转



## EasyUI介绍

### 下载地址

网址:http://www.jeasyui.net/download/jquery.html



## EasyUI入门案例

详情参数入门案例和API

1. 引入js/css样式

<!--引入jquery的js，EasyUI的执行需要依赖于jQuery -->

<script type=*"text/javascript"*

src=*"/js/jquery-easyui-1.4.1/jquery.min.js"*></script>

<!--easyUI的js主文件 -->

<script type=*"text/javascript"*

src=*"/js/jquery-easyui-1.4.1/jquery.easyui.min.js"*></script>

<!--国际化的js文件 -->

<script type=*"text/javascript"*

src=*"/js/jquery-easyui-1.4.1/locale/easyui-lang-zh\_CN.js"*></script>

<!--引入easyUI的样式 -->

<link rel=*"stylesheet"* type=*"text/css"*

href=*"/js/jquery-easyui-1.4.1/themes/icon.css"* />

<link rel=*"stylesheet"* type=*"text/css"*

href=*"/js/jquery-easyui-1.4.1/themes/default/easyui.css"* />

### 拖拽

修改样式:

<body>

<div class=*"easyui-draggable"*>拖动DIV</div>

<div class=*"easyui-draggable"*>测试div</div>

</body>

### 页面中url跳转

<ul>

<li data-options=*"attributes:{'url':'/page/item-add'}"*>新增商品</li>

<li data-options=*"attributes:{'url':'/page/item-list'}"*>查询商品</li>

<li data-options=*"attributes:{'url':'/page/item-param-list'}"*>规格参数</li>

</ul>

## 页面通用跳转功能

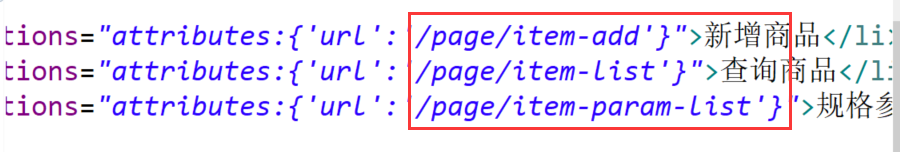
### 跳转问题

问题说明:

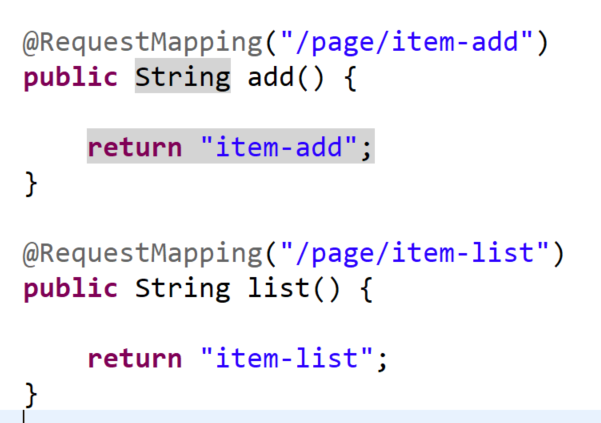
如果用户跳转页面时,如果跳转的路径**类似**,则可以使用通用的页面跳转技术.如果页面跳转使用传统方式开发,每有一个url,在Controller中必须写一个与之对应的@RequestMapping注解标识的方法!!!

问题案例:

1. 页面url



1. 传统controller写法



### REST实现

/\*\*

\* 思考:能够动态的从url中获取路径,则获取跳转页面的名称

\* 实现:REST结构 REST-FUL

\* 要求:

\* 1.发送方 参数必须拼接在url中,并且以/分割

\* 2.接收方 url中的参数必须使用{}包裹

\* 3.为了转化参数

\* 在方法中添加一个名称一致的参数,

\* 同时添加转化的注解@PathVariable

\* **@return**

\*/

///page/item-list

@RequestMapping("/page/{moduleName}")

**public** String add(@PathVariable String moduleName) {

**return** moduleName;

}

# 实现商品列表展现

## JSON格式

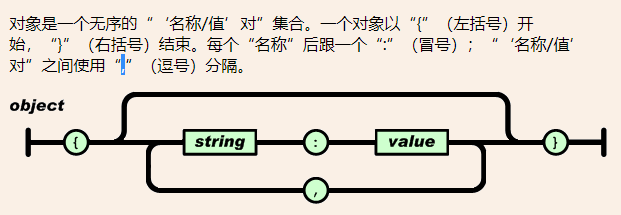
### 什么是JSON

JSON是一种轻量级的数据交互格式(**字符串**写法)

**JSON**(JavaScript Object Notation) 是一种轻量级的数据交换格式。 易于人阅读和编写。同时也易于机器解析和生成。

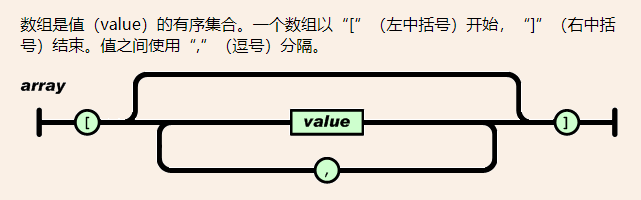
### Object格式(简单)

语法:{key1:value,key2:value}



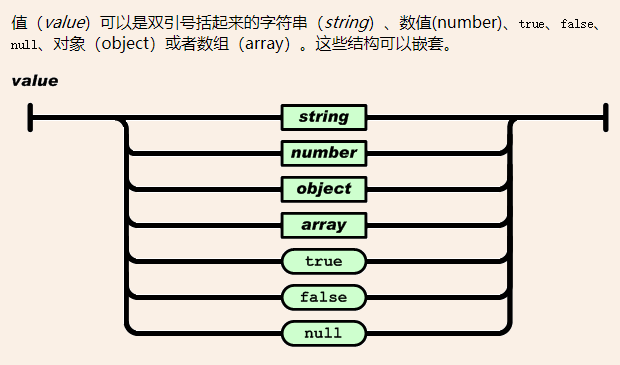
### Array(简单)

语法:[value1,value2…..]



### 复杂格式

说明:JSON复杂格式可以进行无限层级的**嵌套**. **Value**可以是任何的值



例子:

{id:1,name:”tomcat猫”,hobby:[“吃”,”玩”],eat:{id:100,type:”老干妈型”}}

## PowderDesigner

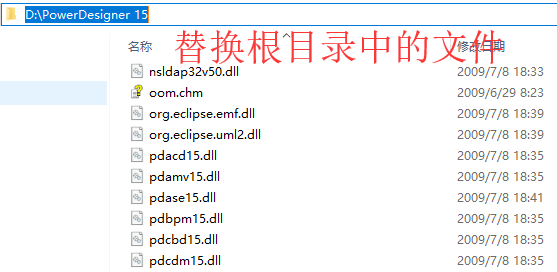
### PowderDesigner介绍

PowerDesigner最初由Xiao-Yun Wang（[王晓昀](https://baike.baidu.com/item/%E7%8E%8B%E6%99%93%E6%98%80/9236799)）在SDP Technologies公司开发完成。PowerDesigner是[Sybase](https://baike.baidu.com/item/Sybase" \t "_blank)的企业建模和设计解决方案，采用模型驱动方法，将业务与IT结合起来，可帮助部署有效的企业体系架构，并为研发生命周期管理提供强大的分析与设计技术。PowerDesigner独具匠心地将多种标准**数据建模**技术（UML、业务流程建模以及市场领先的[数据建模](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BB%BA%E6%A8%A1/720111" \t "_blank)）集成一体，并与 .NET、WorkSpace、PowerBuilder、Java™、Eclipse 等主流开发平台集成起来，从而为传统的软件开发周期管理提供业务分析和规范的数据库设计解决方案。此外，它支持60多种关系数据库管理系统（RDBMS）/版本。PowerDesigner运行在Microsoft Windows平台上，并提供了Eclipse插件。[1]

PD:可以以图形化界面的形式展现表与表之间的关联关系.可以根据不同的数据库自动的生成建库建表语句!!!

### 汉化破解

将课前资料中的汉化补丁复制后替换根目录中的文件



## EasyUI-表格样式

### 页面结构

<div>

定义表格，并且通过url访问json数据， fitColumns:true表示自动适应，singleSelect:true 表示选中单个

<table class=*"easyui-datagrid"* style="width:*500px*;height:*300px*" data-options=*"url:'datagrid\_data.json',fitColumns:true,singleSelect:true,pagination:true"*>

<thead>

<tr>

<th data-options=*"field:'code',width:100"*>Code</th>

<th data-options=*"field:'name',width:100"*>Name</th>

<th data-options=*"field:'price',width:100,align:'right'"*>Price</th>

</tr>

</thead>

</table>

</div>

### 数据回显样式

{

"total":200,

"rows":[

{"code":*"A"*,"name":*"果汁"*,"price":*"20"*},

{"code":*"B"*,"name":*"汉堡"*,"price":*"30"*},

{"code":*"C"*,"name":*"鸡柳"*,"price":*"40"*},

{"code":*"D"*,"name":*"可乐"*,"price":*"50"*},

{"code":*"E"*,"name":*"薯条"*,"price":*"10"*},

{"code":*"F"*,"name":*"麦旋风"*,"price":*"20"*},

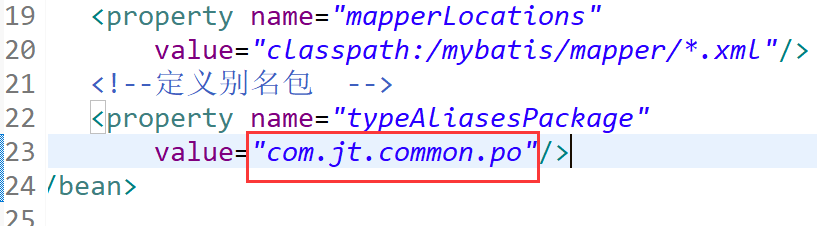
{"code":*"G"*,"name":*"套餐"*,"price":*"100"*}

]

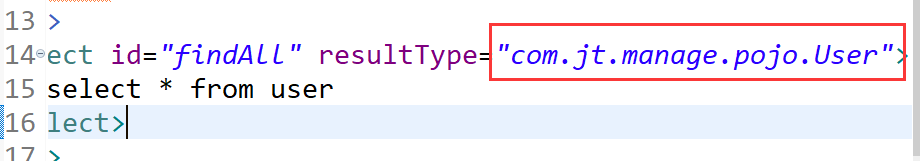
}

### 项目准备

为了防止,以后编码是pojo出现多份,则将公用的POJO对象保存到jt-common中.



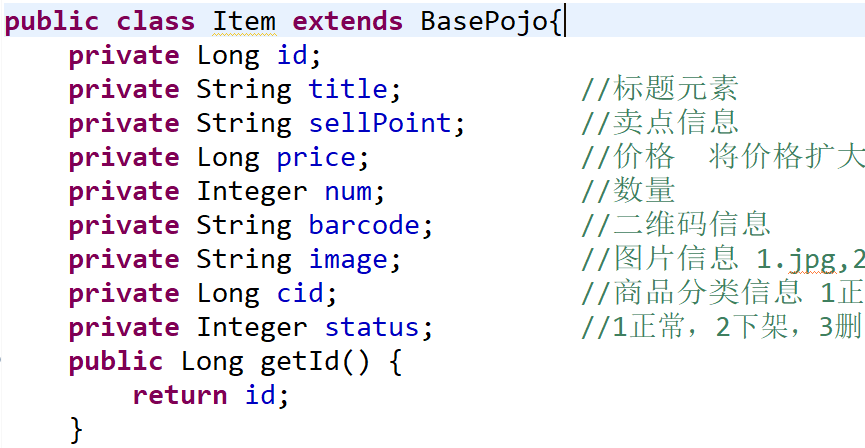
2.mapper文件修改,将resultType中的数据改为全路径型



## 实现商品列表展现

### 定义pojo对象

编辑完成后,将common打包



### 根据返回数据定义VO对象

**public** **class** EasyUI\_Data **implements** Serializable{

**private** Integer total; //商品记录总数

**private** List<Item> rows;

**public** Integer getTotal() {

**return** total;

}

**public** **void** setTotal(Integer total) {

**this**.total = total;

}

**public** List<Item> getRows() {

**return** rows;

}

**public** **void** setRows(List<Item> rows) {

**this**.rows = rows;

}

**public** EasyUI\_Data() {

//准备无参构造

}

**public** EasyUI\_Data(Integer total, List<Item> rows) {

**super**();

**this**.total = total;

**this**.rows = rows;

}

}

### 页面url分析

1. 页面url



1. 页面js

data-options="singleSelect:false,collapsible:true,pagination:true,

**url:'/item/query'**,method:'get',pageSize:10,toolbar:toolbar"

### 编辑Controller

@Controller

@RequestMapping("/item")

**public** **class** ItemController {

@Autowired

**private** ItemService itemService;

//实现商品列表的分页查询

@RequestMapping("/query")

@ResponseBody //将数据转化为JSON

**public** EasyUI\_Data findItemByPage(Integer page,Integer rows) {

**return** itemService.findItemByPage(page,rows);

}

}

### 编辑Service

@Service

**public** **class** ItemServiceImpl **implements** ItemService {

@Autowired

**private** ItemMapper itemMapper;

@Override

**public** EasyUI\_Data findItemByPage(Integer page, Integer rows) {

**int** total = itemMapper.findCount(); //商品记录总数

/\*\*

\* SELECT \* FROM tb\_item LIMIT 起始位置,查询记录数

查询第一页

SELECT \* FROM tb\_item LIMIT 0,20 (0-19)

查询第二页

SELECT \* FROM tb\_item LIMIT 20,20 (20-39)

查询第N页

SELECT \* FROM tb\_item LIMIT (n-1)\*20,20\*/

**int** start = (page - 1) \* rows;

List<Item> itemList =

itemMapper.findItemByPage(start,rows);

//分页后 查询的结果

**return** **new** EasyUI\_Data(total,itemList);

}

}

### 编辑Mapper接口和实现类

public interface ItemMapper {

//@Insert

//@Update

//@Delete

@Select("select count(\*) from tb\_item")

int findCount();

/\*\*

\* Mybatis中如何有多值传参,则需要将多值封装为单值

\* 例子:封装为对象/Map集合/List/array

\* @param start

\* @param rows

\* @return

\*/

List<Item> findItemByPage(@Param("start")Integer start,

@Param("rows")Integer rows);

}

编辑映射文件:

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"com.jt.manage.mapper.ItemMapper"*>

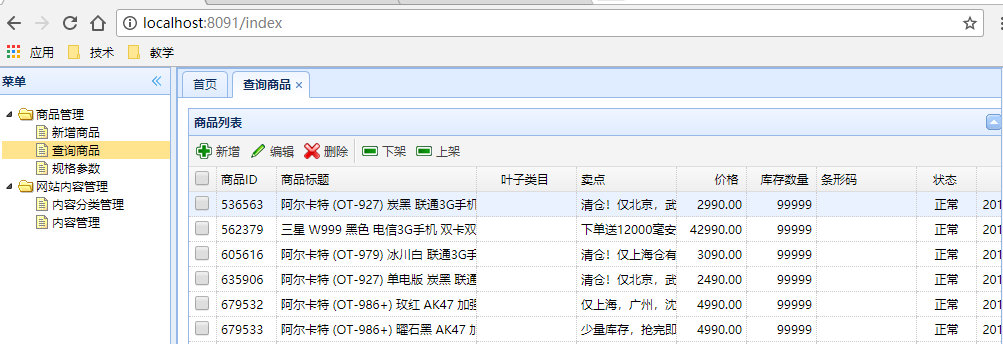
<select id="findItemByPage" resultType="Item">

select \* from tb\_item order by updated desc limit #{start},#{rows}

</select>

</mapper>

### 页面效果



## 实现商品分类目录回显

### 格式化价格

// 格式化价格

formatPrice : **function**(val,row){

**return** (val/100).toFixed(2);

},

### 格式化时间

// 格式化时间

formatDateTime : **function**(val,row){

**var** now = **new** Date(val);

**return** now.format("yyyy-MM-dd hh:mm:ss");

},

### 格式化商品分类JS

//格式化名称

findItemName : **function**(val,row){

**var** name;

$.ajax({

type:"post",

url:"/item/cat/queryItemName",

data:{itemId:val},

cache:**true**, //缓存

async:**false**, //表示同步 默认的是异步的true

dataType:"text",

success:**function**(data){

name = data;

}

//一个ajax请求或有多个ajax嵌套,则同步刷新

});

**return** name;

},

### 编辑Controller

/\*\*

\* spring4及以下 如果返回值是string类型,则采用ISO-8859-1格式转码

\* 如果返回值是对象,则采用UTF-8格式编码

\*

\* StringHttpMessageConverter

\* public static final Charset DEFAULT\_CHARSET = Charset.forName("ISO-8859-1");

\*

\* AbstractJackson2HttpMessageConverter

\* public static final Charset DEFAULT\_CHARSET = Charset.forName("UTF-8");

\* **@param** itemId

\* **@return**

\*/

//根据商品分类ID查询分类名称

@RequestMapping(value="/cat/queryItemName",

produces="text/html;charset=utf-8")

@ResponseBody

**public** String findItemCatNameById(Long itemId,

HttpServletResponse response) {

**return** itemService.findItemCatNameById(itemId);

}