# 京淘商品管理实现

## 跳转后台首页

### 编辑Controller

@Controller

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

//跳转到首页

@RequestMapping("/index")

**public** String index(){

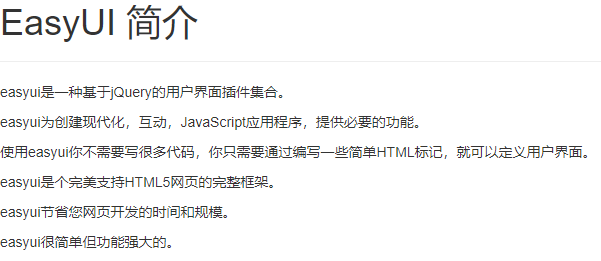
**return** "index";

}

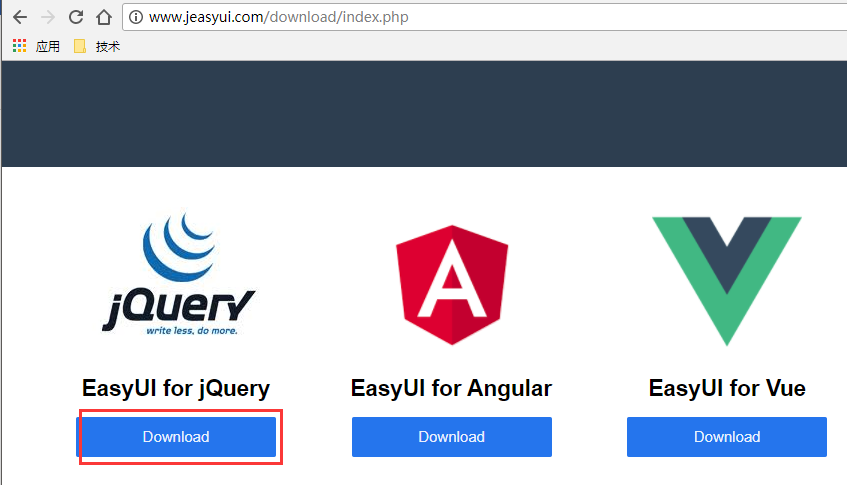
}

## EasyUI介绍

### EasyUI



### EasyUI下载



### EasyUI入门案例

1.引入js

<!--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"* />

1. 案例1 拖动

<body>

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

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

</body>

1. 进度条

$(function(){

$("#b").click(function(){

onload();

})

/\*采用递归的方法实现进度条刷新 \*/

var i = 0;

function onload(){

$('#p').progressbar({ value:i++});

if(i<=100){

/\*定时超时,到了指定的时间,则执行函数,时间

单位是毫秒\*/

setTimeout(function(){

onload();

}, 1)

}

}

})

剩余的案例,自行学习.

### 页面布局

**<div** id="cc" class="easyui-layout" style="width:600px;height:400px;"**>**

**<div** data-options="region:'north',title:'North Title',split:true" style="height:100px;"**></div>**

**<div** data-options="region:'south',title:'South Title',split:true" style="height:100px;"**></div>**

**<div** data-options="region:'east',iconCls:'icon-reload',title:'East',split:true" style="width:100px;"**></div>**

**<div** data-options="region:'west',title:'West',split:true" style="width:100px;"**></div>**

**<div** data-options="region:'center',title:'center title'" style="padding:5px;background:#eee;"**></div>**

**</div>**

### 树形结构

**<ul** id="tt" class="easyui-tree"**>**

**<li>**

**<span>**Folder**</span>**

**<ul>**

**<li>**

**<span>**Sub Folder 1**</span>**

**<ul>**

**<li>**

**<span><a** href="#"**>**File 11**</a></span>**

**</li>**

**<li>**

**<span>**File 12**</span>**

**</li>**

**<li>**

**<span>**File 13**</span>**

**</li>**

**</ul>**

**</li>**

**<li>**

**<span>**File 2**</span>**

**</li>**

**<li>**

**<span>**File 3**</span>**

**</li>**

**</ul>**

**</li>**

**<li>**

**<span>**File21**</span>**

**</li>**

**</ul>**

### 选项卡技术

<body class="easyui-layout">

<div data-options="region:'west',title:'菜单',split:true" style="width:10%;">

<ul class="easyui-tree">

<li>

<span>商品管理</span>

<ul>

<li><a onclick="addTab('商品新增','/item-add')">商品新增</a></li>

<li><a onclick="addTab('商品查询','/item-list')">商品查询</a></li>

<li><a onclick="addTab('商品更新','/item-update')">商品更新</a></li>

</ul>

</li>

<li>

<span>网站内容管理</span>

<ul>

<li>内容新增</li>

<li>内容修改</li>

</ul>

</li>

</ul>

</div>

<div id="tt" class="easyui-tabs" data-options="region:'center',title:'首页'"></div>

</body>

<script type="text/javascript">

**function** addTab(title, url){

**if** ($('#tt').tabs('exists', title)){

$('#tt').tabs('select', title);

} **else** {

**var** content = '<iframe scrolling="auto" frameborder="0" src="'+url+'" style="width:100%;height:100%;"></iframe>';

$('#tt').tabs('add',{

title:title,

content:content,

closable:**true**

});

}

}

</script>

## RESTFul结构

### 需求

需求:

用户的请求如果很多,那么Controller中必须写多个的方法与之对应,实现页面跳转.这样做,会很繁琐.如何优化???

代码:

<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>

### 采用RESTFul实现通用页面跳转

/\*\*

\* /page/item-add

\* /page/item-list

\* /page/item-param-list

\* **@return**

\*

\* RESTFul结构

\* 格式特点:

\* 1.在url中将需要提交的参数使用"/"进行分割

\* localhost:8091/addUser/1/tom

\* 2.在接收端,将参数使用{}包裹并且参数的位置固定.

\* 3.变量的名称必须和{}中的名称一致,之后通过@PathVariable

\* 实现数据的传递.

\*

\* 功能:

\* 1.实现了通用页面的跳转问题.

\* 2.减少了代码传递的字节数.

\*/

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

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

**return** moduleName;

}

## 实现商品数据展现

### EasyUI中表格数据

1. 页面请求样式

<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>

1. 返回JSON数据

{

**"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"}

]

}

### 商品展现页面

<table class="easyui-datagrid" id="itemList" title="商品列表"

data-options="singleSelect:false,collapsible:true,pagination:true,**url:'/item/query'**,method:'get',pageSize:20,toolbar:toolbar">

<thead>

<tr>

<th data-options="field:'ck',checkbox:true"></th>

<th data-options="field:'id',width:60">商品ID</th>

<th data-options="field:'title',width:200">商品标题</th>

<th data-options="field:'cid',width:100,align:'center',formatter:KindEditorUtil.findItemName">叶子类目</th>

<th data-options="field:'sellPoint',width:100">卖点</th>

<th data-options="field:'price',width:70,align:'right',formatter:KindEditorUtil.formatPrice">价格</th>

<th data-options="field:'num',width:70,align:'right'">库存数量</th>

<th data-options="field:'barcode',width:100">条形码</th>

<th data-options="field:'status',width:60,align:'center',formatter:KindEditorUtil.formatItemStatus">状态</th>

<th data-options="field:'created',width:130,align:'center',formatter:KindEditorUtil.formatDateTime">创建日期</th>

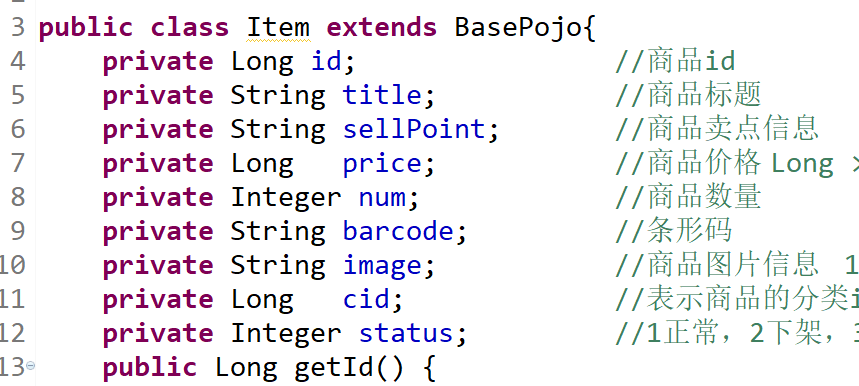
<th data-options="field:'updated',width:130,align:'center',formatter:KindEditorUtil.formatDateTime">更新日期</th>

</tr>

</thead>

</table>

### 创建ItemPOJO对象



### 修改Mybaits别名包路径

<bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"></property>

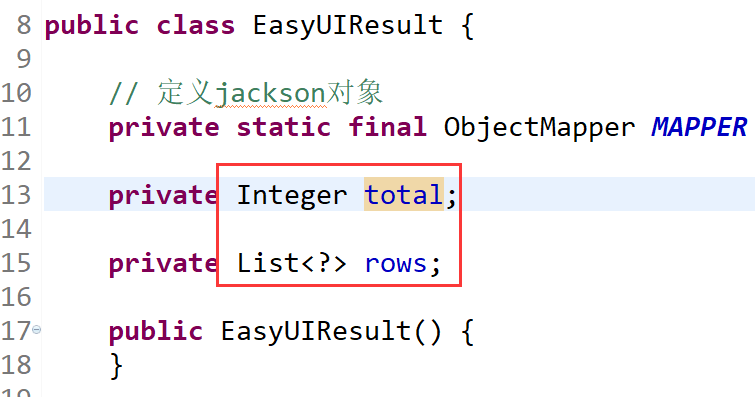
<property name="configLocation" value="classpath:/mybatis/mybatis-config.xml"/>

<property name="mapperLocations" value="classpath:/mybatis/mapper/\*.xml"/>

<property name="typeAliasesPackage" value="**com.jt.common.po**"/>

</bean>

### 编辑VO对象



### 编辑前台Controller

用户请求: <http://localhost:8091/item/query?page=1&rows=20>

@Controller

@RequestMapping("/item")

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

@Autowired

**private** ItemService itemService;

//现在的Item对象真的可以使用吗????

/\*\*

\* http://localhost:8091/item/query?page=1&rows=20

\* 返回JSON格式要求:

\* total:返回记录总数

\* rows: 返回的列表信息

\*/

@RequestMapping("/query")

@ResponseBody

**public** EasyUIResult 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** EasyUIResult findItemByPage(Integer page, Integer rows) {

//1.查询商品的记录总数

**int** total = itemMapper.findItemCount();

/\*\*

\* 第一页 SELECT \* FROM tb\_item ORDER BY updated DESC LIMIT 0,10

\* 第二页 SELECT \* FROM tb\_item ORDER BY updated DESC LIMIT 10,10

\* 第三页 SELECT \* FROM tb\_item ORDER BY updated DESC LIMIT 20,10

\* 第n页 SELECT \* FROM tb\_item ORDER BY updated DESC LIMIT (n-1)\*rows,rows

\*/

**int** start = (page - 1) \* rows; //起始位置

List<Item> itemList =

itemMapper.findItemByPage(start,rows);

**return** **new** EasyUIResult(total, itemList);

}

}

### 编辑Mapper

public interface ItemMapper {

/\*@Insert

@Update

@Delete\*/

/\*\*

\* 说明:因为不同的数据库对于大小写,有不同的要求.

\* 操作系统中.window不区分大小写,但是在Linux系统中

\* 严格区分大小写.所以为了保持一致统统都小写.

\* ctrl + shift + y 小写

\* ctrl + shift + x 大写

\* @return

\*/

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

int findItemCount();

/\*\*

\* Mybatis的Mapper接口不允许多值传输

\* 思路:

\* 将多值封装为单值

\* 1.将值封装到对象中pojo

\* 2.将值封装为集合 数组array/List集合/Map集合

\* @param start

\* @param rows

\* @return

\*/

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

}

### 编辑Mapper映射文件

<!--

关于#号和$的区别

select \* from user where username = "?" and password = ?"

#号:

1.使用#号传参有预编译的效果. 防止sql注入攻击

2.为参数添加一对""号

约定:能用#号不用$

$符:

只有以字段的名称为参数时,才使用$符

例子:

select

-->

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

select \* from tb\_item order by updated desc

limit #{start},#{rows}

</select>

## 实现商品分类信息回显

### 页面分析

<th data-options="field:'price',width:70,align:'right',formatter:KindEditorUtil.formatPrice">价格</th>

### 格式化价格

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

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

},

### 实现商品分类回显JS

1.页面数据回显583.需要将583转化为具体分类名称



//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

//设置同步的原因,表格数据是通过ajax加载的,在加载表格数据的同时,再次发起ajax请求.需要页面动态刷新2次.才能展现数据.但是页面中无法实现2次刷新.所以设置为同步.

dataType:"text",//表示返回值参数类型

success:**function**(data){

name = data;

}

});

**return** name;

}

### 编辑Controller

/\*\*

\* 根据商品分类的Id查询名称

\* **@ResponseBody进行数据解析时**,如果解析的是对象(pojo/vo/List/Map..)

\* 默认格式都采用utf-8格式解析.

\* 如果解析的数据是String类型,则按照iso-8859-1格式进行解析.

\* public class StringHttpMessageConverter extends AbstractHttpMessageConverter<String> {

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

public abstract class AbstractJackson2HttpMessageConverter extends AbstractHttpMessageConverter<Object>

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

\* **@param** itemId

\* **@return**

\*/

@RequestMapping(value="/cat/queryItemName",produces="text/html;charset=utf-8")

@ResponseBody

**public** String findItemCatNameById(String itemId,HttpServletResponse response){

//response.setContentType("text/html;charset=utf-8");

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

}

### 编辑Service

@Override

**public** String findItemCatNameById(String itemId) {

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

}

### 编辑mapper

@Select("select name from tb\_item\_cat where id = #{itemId}")

String findItemCatNameById(String itemId);

### 页面效果

