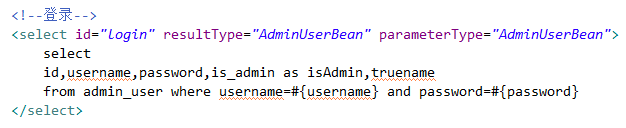
Spring+springMVC+MyBatis整合版本功能，请求都是通过ajax

顺便也把这个项目的内容都总结下

# 一、后台内容

## Login

1.MyBatis的Mapper



2.对应的dao层接口, 无需实现类,注意方法名要和mapper文件的名字对应

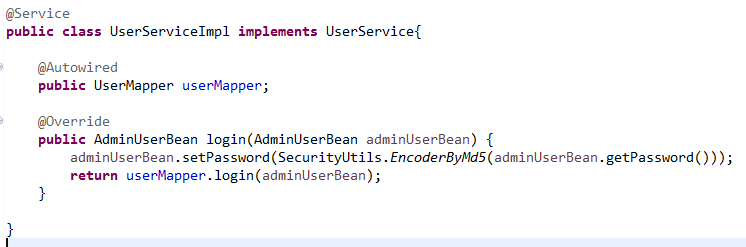


3.Servive层的接口



4.Service实现类

注：调用dao层方法，需要装配dao层对象



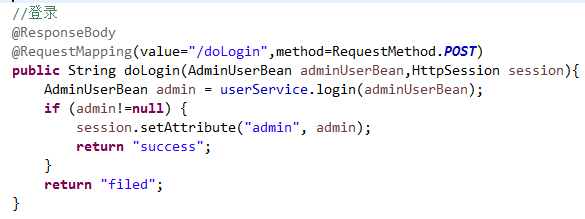
注意：1.类上打Service注解

2.调用dao层方法,用Autowired注解自动装配,不用new

3.传入的密码通过MD5工具类加密

5.controller

注:调用service层方法，需要装配service对象



6.form表单(并不是通过表单提交了，form表单只是收集数据，通过ajax提交)



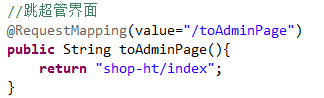
流程：1.给表单里的a标签添加点击事件,点击后，调到controller里的方法

2.controller里调用service方法，service调用dao层的Mapper接口

3.将返回的bean存入session,返回给页面success信息

4.页面根据返回的信息再进行跳转

5.注意dao层,service层,controller层的注解



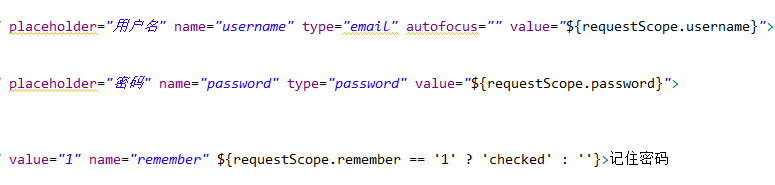
Ps:登录时cookie的实现(记住密码)

1.controller内容



解释：调到登录界面时，就从浏览器查询所有cookie,如果不为空的话，分别设置到request

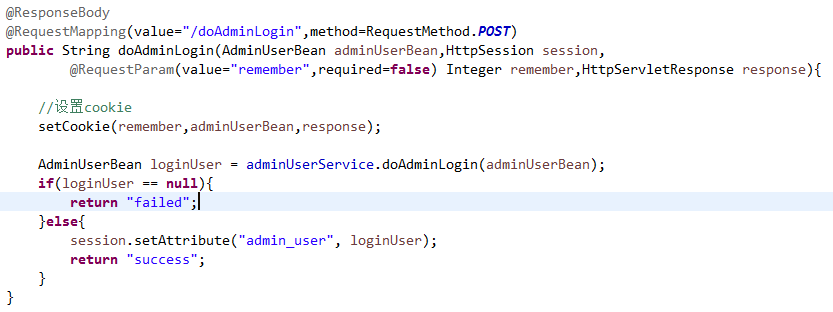
页面取值如下:



2.当然登录的时候要把cookie设置到浏览器



上边是设置cookie的方法,很简单,登录的方法如下



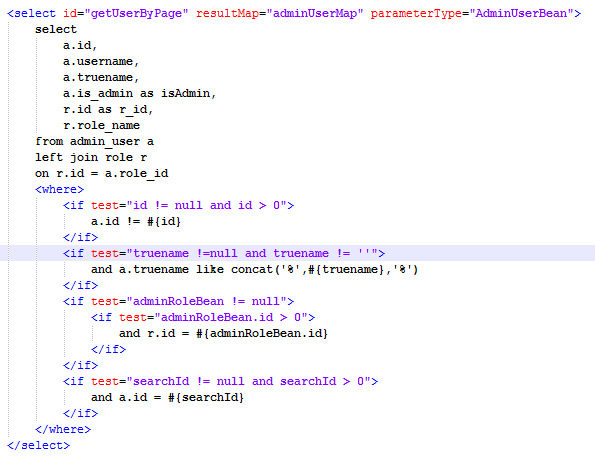
## 获取所有用户(分页,条件查询)

查询所有用户信息，有多种查询方式

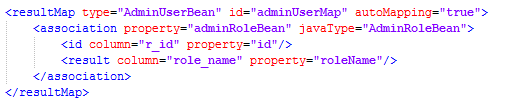
1. 获取所有2.分页3.条件查询

所以即使是查所有，也要传入一个bean,便于条件查询使用

Mapper



对应的resultMap



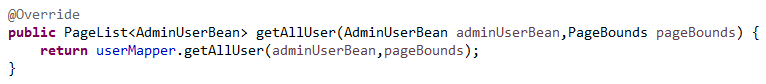
Mapper接口



Service接口



Service实现类

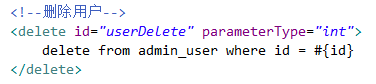


Controller

## 删除用户

删除没啥需要注意的，注意传入id即可

Mapper.xml



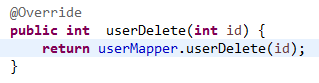
Mapper接口



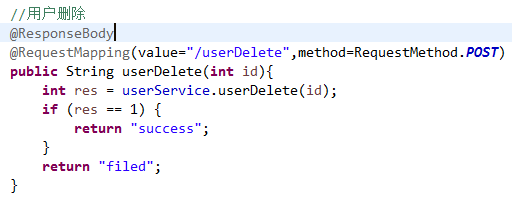
Service接口



Service实现类



Controller



Jsp界面

1.给a标签绑定点击事件onclick,写函数，并传入参数



2.写点击事件函数

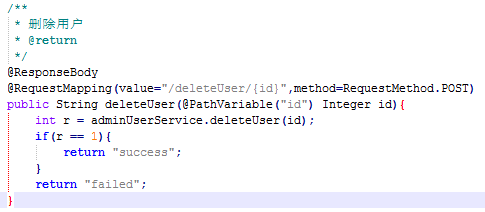


或者通过

传参



通过连接传参



Controller里这样接参数 @PathVariable(“id”) integer id

以上是两种传参方法

## 修改用户信息

点击a标签跳转到修改用户信息界面，并传递用户id参数



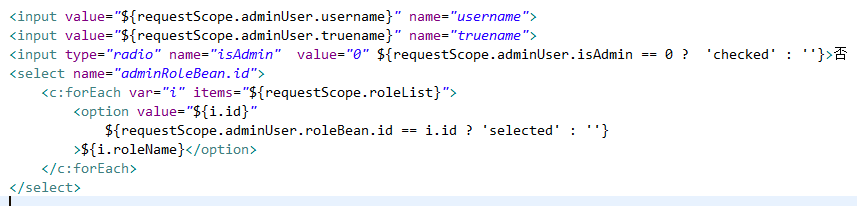
跳转的时候，要先查询用户信息，并显示在对应的位置,获取用户信息的方法已经写过了。

注意:获取所有用户，有三种获取条件1.所有2.条件查询3.分页信息

这里传入了一个通过bean传入了一个id,所以，只查询当前用户的信息，并返回,这里看一下conreoller的内容



这样在页面可以通过requestScope获取参数,像这样



**注意在标签上绑定信息的方式！1.普通input框，2.单选按钮3.下拉框**

**这样在页面就可以愉快的显示用户信息了！**

Mapper.xml

\

注意判断条件

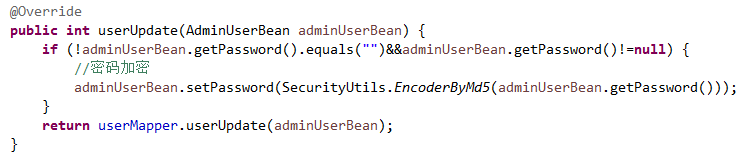
Mapper接口



Service接口

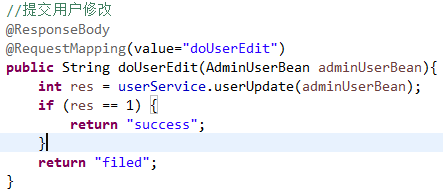


service实现类，将修改的用户密码加密



Controller

**注意：通过json传递信息，要写上responseBody注解！**



页面信息都是在一个form表单里



给表单一个id，便于选中



对应标签写上对应的name属性，对应bean里的属性名



级联属性这么写！

然后通过a标签提交，注意a标签的id



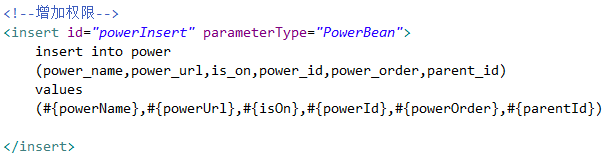
Ajax方法



## 增加

新增比较简单，点击a标签，直接跳转，不需要传参

Mapper.xml



传入一个bean

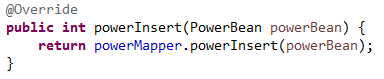
Mapper接口



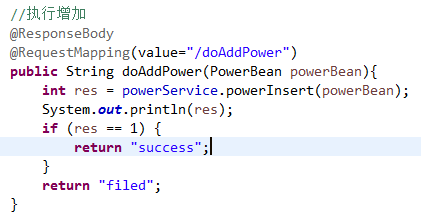
Service接口



Service实现类



Controller



Jsp



注意：1.表单提交2.a标签ajax方法3.标签name属性对应bean里的属性名

Ajax方法：



这样增删改查就都有啦！睡觉去~~~~~~~~~~

## 6.不同用户登录显示不同的权限信息

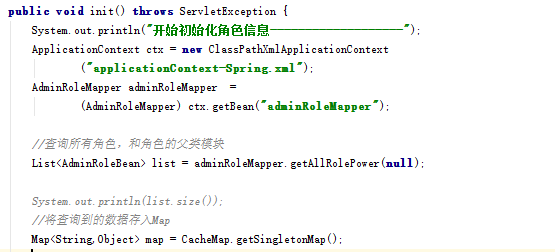
在启动服务器的时候，就查询所有的用户权限，并把它保存到一个map里

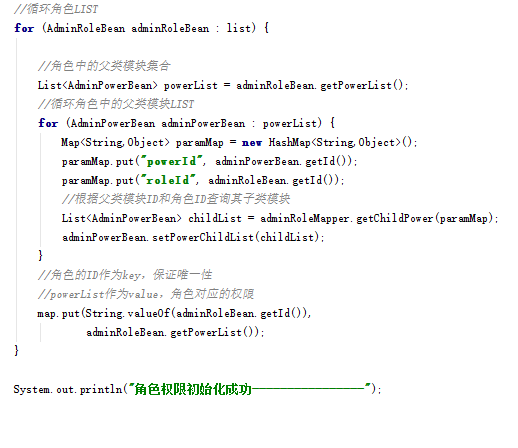
(为什么?因为这样每个用户登录的时候,就不用每次都从数据库查,而是直接从内存读取,效率高很多)

Web.xml配置

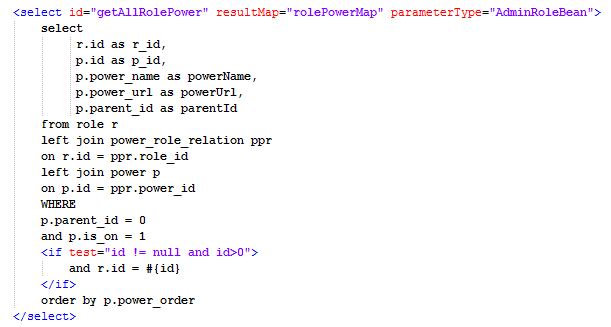


走的servlet,重写init方法

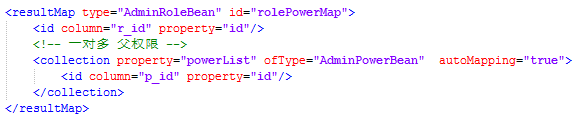




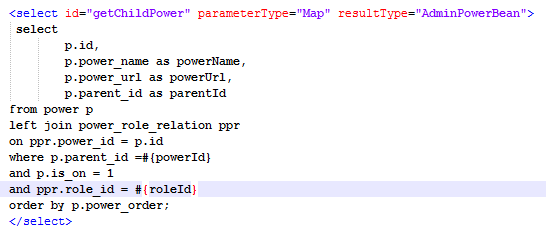
1. 查询所有的父权限



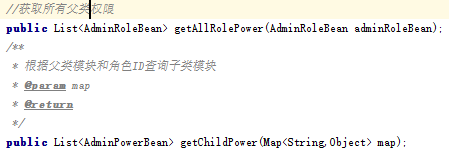
对应的resultMap



1. 获取所有的子权限



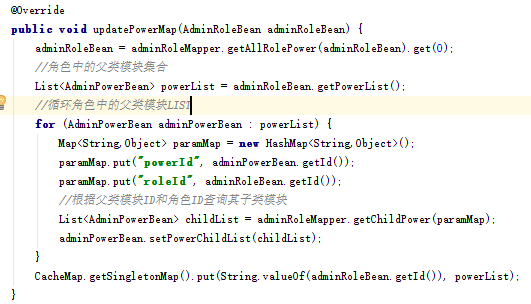
Mapper接口



Service接口



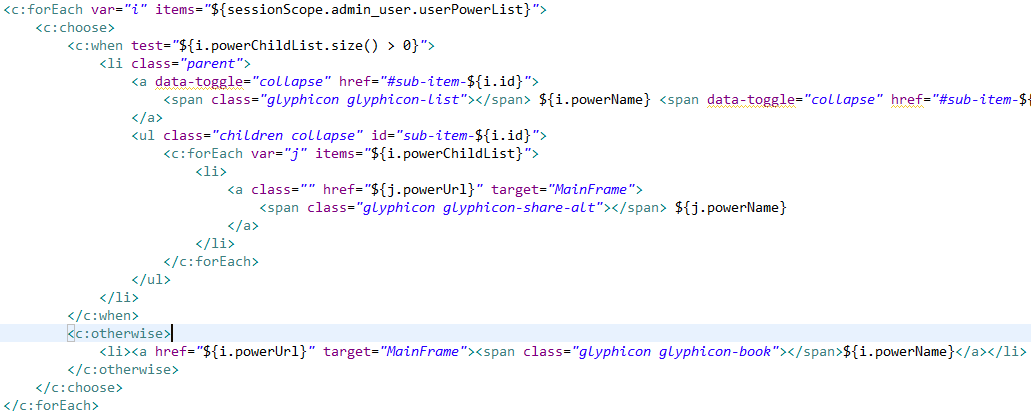
Service实现类



拦截器



页面



实现了不同用户登录，页面左侧显示不同权限的功能

流程是这样的:

1. 启动服务器的时候，查询所有的用户权限，放到缓存里。是一个静态的map
2. 登录的时候，走一个登录拦截器，如果登录了，有用户信息，根据用户的角色信息，

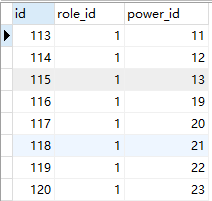
查询该用户对应的权限信息，然后在页面判断，显示就可以啦~~

## 7.角色与权限操作

解释：后台管理系统包含角色和权限模块,角色和权限是一对多的关系,这样，给无需给用户分配权限，直接给用户分配角色，角色和权限一对多，这样修改的时候，直接修改角色信息即可。

1. 功能包括：正常的增删改查，模糊查询，分页等。

注意权限和角色是有关系表的。



这是我们人为规定的一个表，用来显示角色和权限的关系。

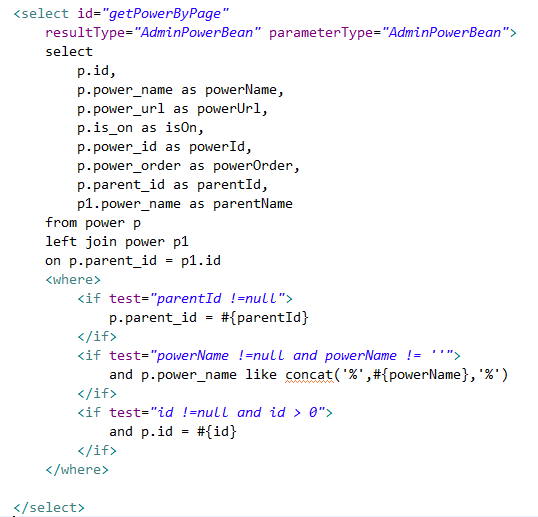
插入角色和权限的时候，也要向这个表插入信息。这样，查询的时候，就方便很多了。

权限分为父权限和子权限,例如：



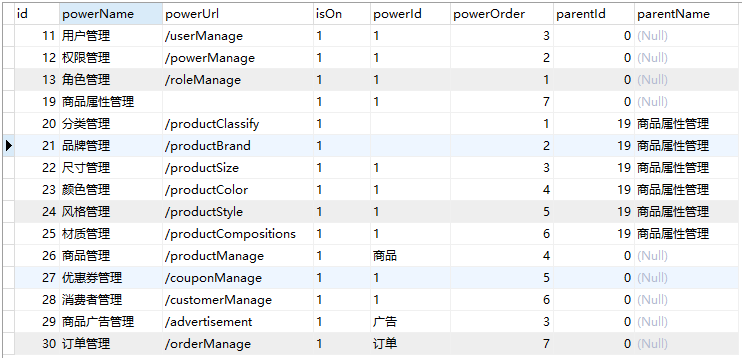
商品属性管理包括：分类、品牌等等管理权限.

我们呢查询权限信息的时候，用到了一个(表自己左连接自己的)的技巧。



为什么？因为这样，在查询子权限信息的时候，才能把父权限的信息显示出来。

查询结果是这样的：



我们可以看到，拥有父权限的子权限，父权限也显示出来了。

这里可以自行结合页面来理解.

# 二、前台内容

## 1.拦截器的实现(spring来实现的)

页面的大部分操作都需要用户登录后才能实现，用户进行某些关键操作的时候，我们要实时的检查用户是否登录了，否则一概跳转登录界面

来看下sprinfMVC的配置文件:



< mvc:mapping path=*"/cart/\*\*"*/>

表示拦截cart路径后所有的方法

<mvc:exclude-mapping path=*"/cart/getShopByCus"*/>

表示不拦截cart路径后的getShopByCus方法

<bean class=*"com.shop.interceptors.UserLoginInterceptor"* />

表示当前拦截器的路径,是interceptors包下的*UserLoginInterceptor方法*

这个方法是我们自己规定的



为什么要判断request请求来自ajax还是普通请求?

因为页面放久了，session里的内容消失了，走拦截器的时候，普通请求是可以走

response.sendRedirect("/user/userLogin");这个方法。

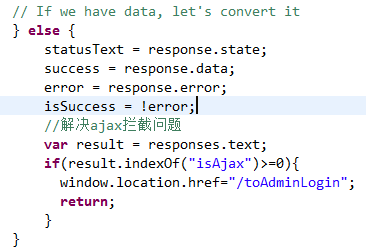
但是ajax请求是不能走这个方法的，页面就会卡住(虽然走了拦截器，但是ajax请求不能走response的方法,就卡住了)

所以我们要判断下，如果是ajax请求，我们需要走这个方法

response.getWriter().write("isAjax");

向页面写字符串isAjax,

然后我们需要改一下jq的源码，



这几行代码大概位于1.11版本jq的9300行左右。

就是处理方法的返回值,如果返回值为isAjax的话,跳转登录界面.

这样解决了因为session里的内容消失，ajax请求会卡住页面的问题。

反正拦截器无论如何都会生效的

## 2.商品图上传(多图上传)

1.前台头像上传

详见ssm\_shop项目的userCenter.jsp和UserCenterController

2.后台多图上传

详见ssm\_shop项目的productPicture.jsp和ProductController

## 3.

# 三、页面数据的绑定和获取

获取:(所有的表单获取name属性都需要同bean里的属性相对应)

1. 文本框

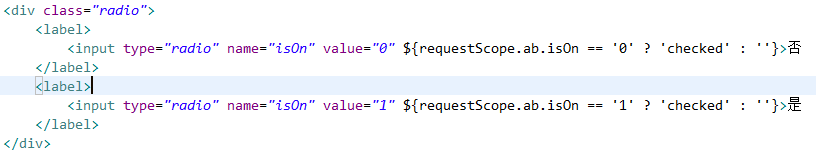


数据绑定：

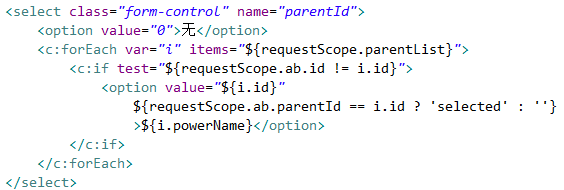
1. 文本框



1. 单选框



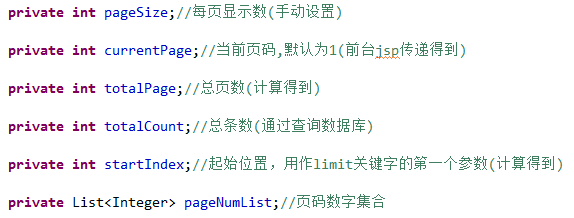
1. 下拉框



# 四、分页总结

1. 普通分页

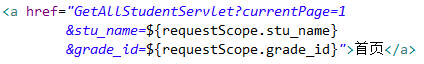
我们先写个分页的工具类，看看我们都需要哪些数据



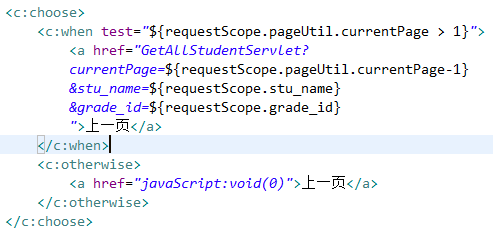
这些是我们分页功能需要的参数,获得的方法不在此赘述.(工具类的文件夹里有)

然后页面:

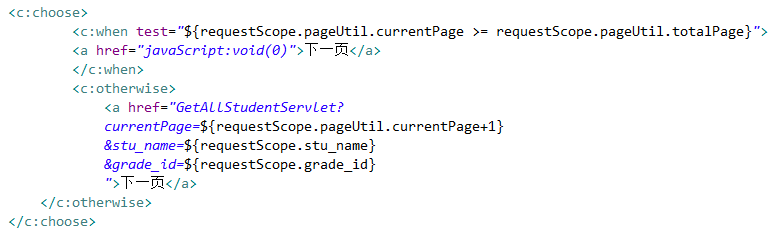
首页:



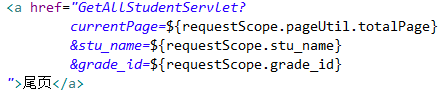
上一页：



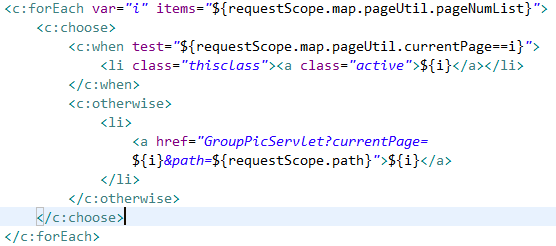
下一页：



尾页：



具体的页码(工具类里有具体的计算方法)：



1. Mybatis分页

mybatis的分页用到了一个jar包。

mybatis-paginator-1.2.5.jar

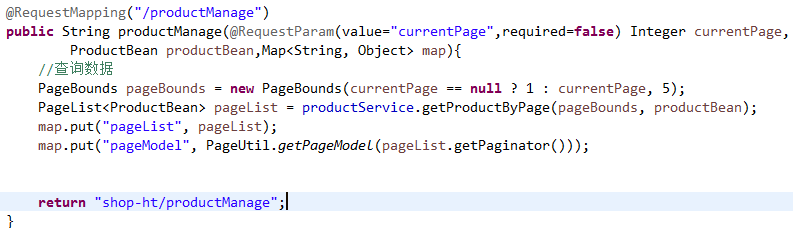
mapper接口：



返回值要为PageList,参数要传递一个pageBounds

service层同理

controller如下：



1.要传一个currentPage的参数

2. 将返回的pageList放到map里

3.同时还要返回一个pageModel到页面

页面如下：

