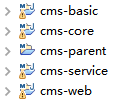
**CMS开发流程白皮书**

1. 创建Maven项目

创建好以下的几层Maven项目



cms-parent提供基础的jar包支持，并且将所有的子层统一进行管理

cms-basic提供基础公用的basedao的方法，以及提供工具util

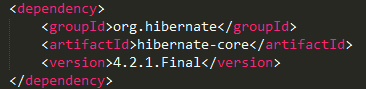
cms-core提供本项目特定的各种dao的方法

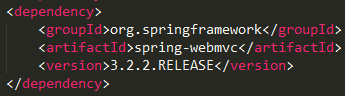
cms-service提供本项目特定的各种service的方法

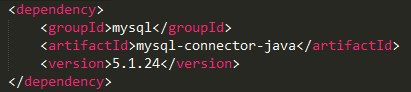
cms-web提供本项目控制、页面以及web的启动

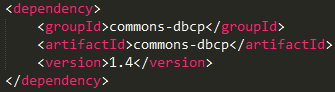
1. 导入需要的jar包

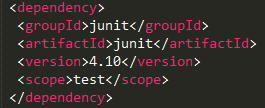
CMS-PARENT,考虑到这里使用的是spring,springmvc,hibernate，那么这些基本的包都是需要导入的。

Hibernate:

SpringMVC: 

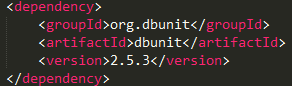
Mysql: 

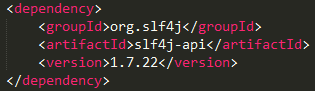
DBCP连接池：

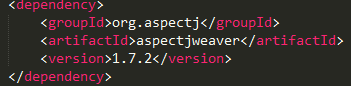
Junit: 

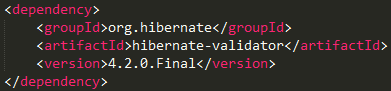
Cms-basic: 

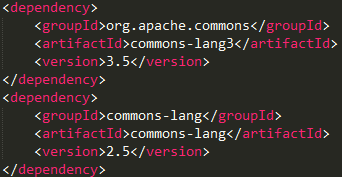
Spring:

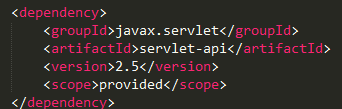
junit扩展的数据库测试框架：

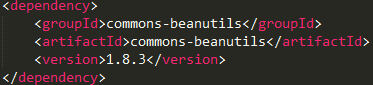
日志：

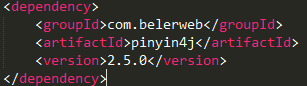
面向切面的框架：

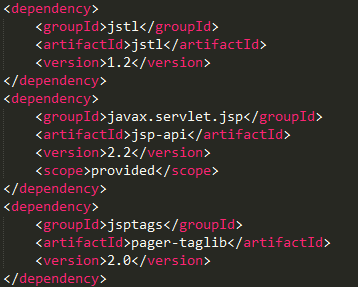
Hibernate的验证器：

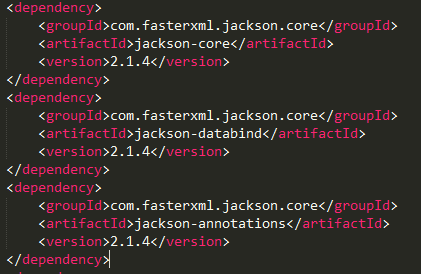
字符串操作：

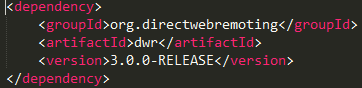
支持servlet: 

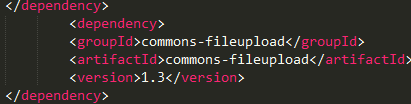
操作bean: 

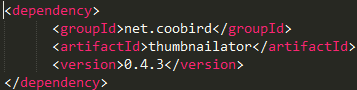
拼音获取名字等：

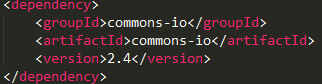
Jsp: 

Json: 

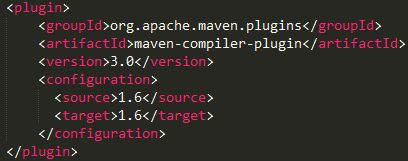
Easy for ajax: 

文件上传：

缩略图: 

Io流处理：

插件



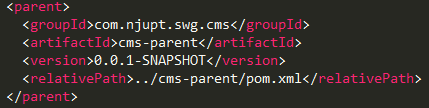


到此为止，几乎所能用到的所有的jar都已经包含进来了，还剩几个关于子项目的没有关联 进来。新增一个加一个。

1. CMS-BASIC

下面就是先实现basic中最公用的底层数据库方法以及必用的一些类，比如分页。还有就是基本的util也放在这里，后面用到的时候再在这里添加。

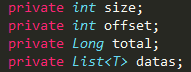
第一步：cms-basic.pom

首先是引入父亲pom：

其次是引入相应的需要的包：

Junit、hibernate、springframework、mysql、inject、dbcp、dbunit、slf4j、aspectj、springtestdbunit

第二步：分页的基本Bean---Pager.java:用于设置分页的基本参数



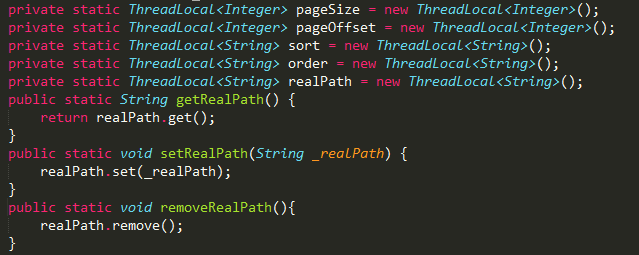
Size:分页大小

Offset:分页的起始页

Total:总记录数

Datas:分页数据

SystemContext.java：用于设置前后台交互的参数



pageSize:分页的大小

pageOffset:分页的起始页

sort:排序的依据，即order by wich sort

order:asc or desc

realPath:获取项目路径

第三步：写BaseDao,见附件IBaseDao和BaseDao，baseDao的编写思路及测试.doc

1. CMS-CORE

这里是实现专用的对象的dao层方法。

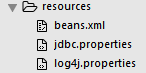
第一步：cms-basic.pom

首先是引入父亲pom，同上。

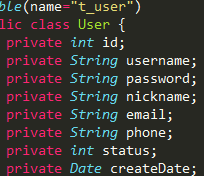
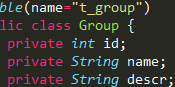
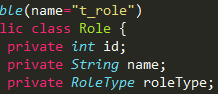
其次就是这里需要一些验证的包，以及需要继承于CMS-BASIC，故pom应为：



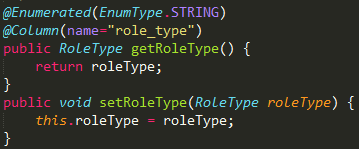
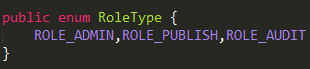
第二步：引入配置文件



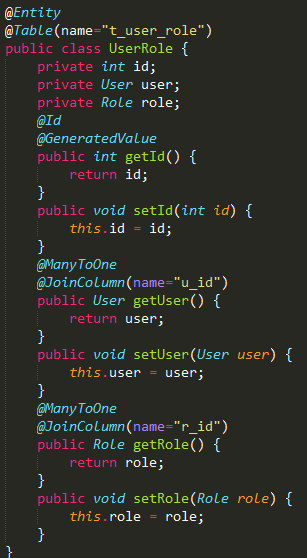
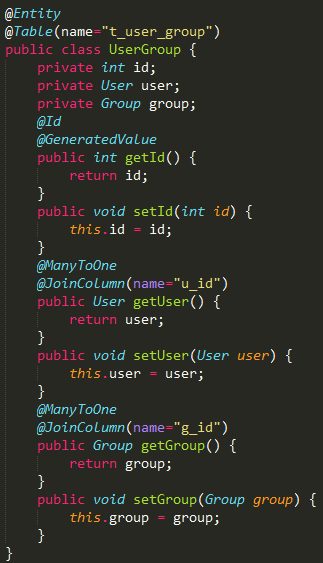
第三步：先创建User、Group、Role这三个实体类

RoleType

UserGroup和UserRole



第四步：完成对应的userdao方法

获取用户的所有角色信息

获取用户的所有角色的id

获取用户的所有分组信息

获取用户的所有分组的id

根据用户和角色获取用户角色的关联对象

根据用户和组获取用户组关联对象

根据用户名获取用户对象

根据角色id获取用户列表

根据角色类型获取用户对象

根据分组id获取用户列表

添加用户角色对象

添加用户组对象

根据用户id删除角色信息

根据用户id删除组信息

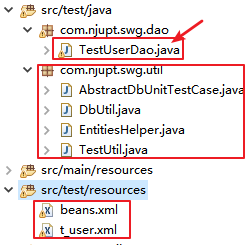
用户列表

删除用户角色对象

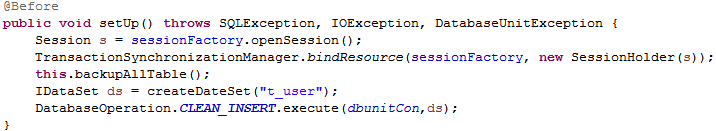
删除用户组对象

第五步：对这些类进行相应的测试。

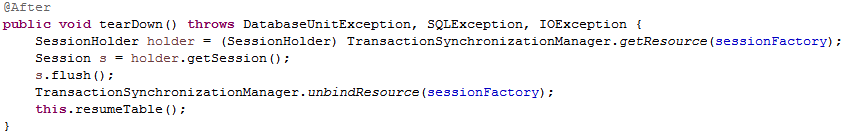
将配置文件以及初始化的数据全部放在src/test/resources文件夹下，然后再src/test/java中写入一些帮助类，这些就是用于测试的连接数据库、初始化数据、进行一些数据比对等方法的实现。最后写TestUserDao这个测试类，同baseDao的编写思路及测试.doc中的测试类。



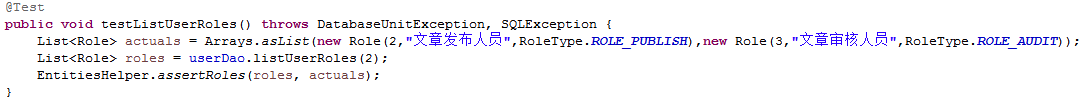
初始化数据库和session



数据库的回滚



测试第一个方法ListUserRoles



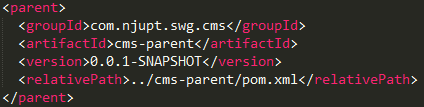
如果都没有报错，说明从parent到basic再到core串成一起都没有什么错误，这里的测试就不一一演示了。

这里暂时先写一个UserDao，接下来写UserService，最后写页面层，如果一切正常再回头考虑其他的逻辑。

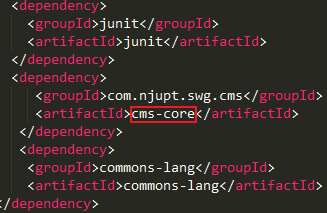
1. CMS-SERVICE

第一步：cms-service.pom

首先是引入cms-parent.pom



引入上面含有UserDao的cms-core

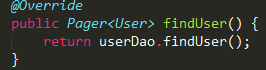


第二步：创建IUserService接口及其实现类

这里的接口先只写一个方法，就是用户列表查询，分页的。



相应的实现类是：

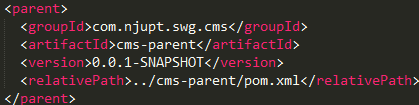


那么service层就先写到这里，下面先完成界面层的显示。

1. CMS-WEB

第一步：cms-service.pom

首先是引入cms-parent.pom



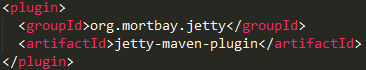
引入上面含有UserDao的cms-service

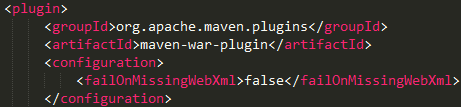
当然，下面引入的很多暂时还不需要，但是这里图省事就先可能要用的先全部粘贴在这里了。

这里主要是要用到一个spring-mvc和servlet-api以及jsp的两个，即dwr和fileupload暂时还不需要。



引入jetty用于项目启动的容器。



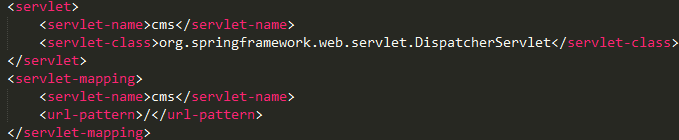


第二步：配置cms-servlet.xml文件

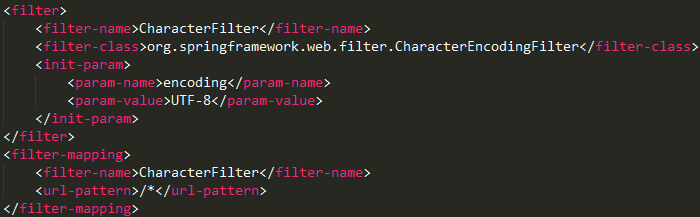
这个文件中首先是配置好一些资源文件所在的路径，再配置好访问的前缀和后缀，进行限制。

第三步：配置web.xml文件

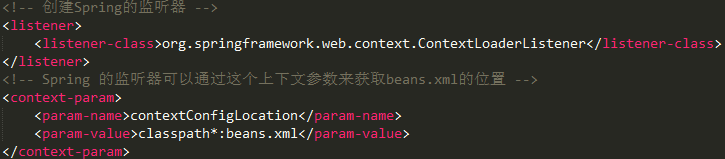
这个文件中首先是配置好DispatcherServlet,提供Spring Web MVC的集中访问点



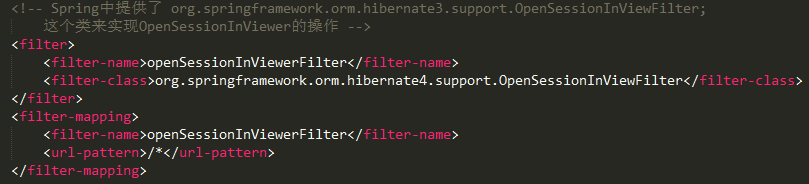
然后是配置编码过滤器，即编码转为utf-8才可以



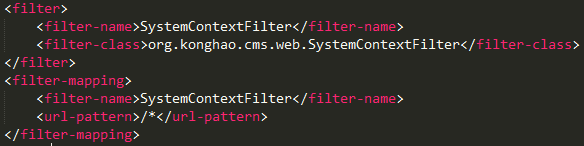
创建spring监听器



解决hibernate的session的关闭与开启问题，由spring来管理



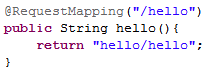
全局变量的赋值，比如分页的参数



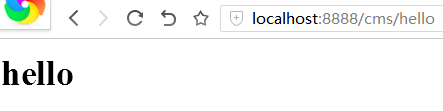
以上的具体配置信息都可以在onenote中的软件管理中的cms项目开发(3)中看到。

第四步：Controllser这一层的编写。注意annotation不要忘记写，并且将service层注入进来。

首先写一个最简单的hello的程序，看是否能够正常跳转和访问。



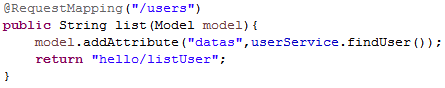
相应地，在webapp下面的WEB-INF下信件一个文件叫做Jsp，并且在该文件夹下面再建一个名字叫做hello的子文件夹，在这个文件夹中新建一个叫做hello.jsp的文件，这里没有参数的传递，就直接页面显示hello即可。



下面再进行相应的参数的传递，这里在数据库中有三个用户的信息，这里将这三个用户的信息依次以表格的形式展现在页面上，数据库信息如下：



首先在控制层中，用model盛放userService获取到的带分页的所有用户的信息，将其存入到名字叫datas中，然后指定跳转的路径。那么这里依然在hello文件夹下新建一个名字叫做listUser.jsp的页面。



那么在页面层，用jstl标签获取集合中的内容。首先是引入jstl标签：



下面就是获取内容:



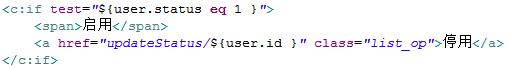
然后是获取每一个参数的值：



这样，最后呈现的效果如下：



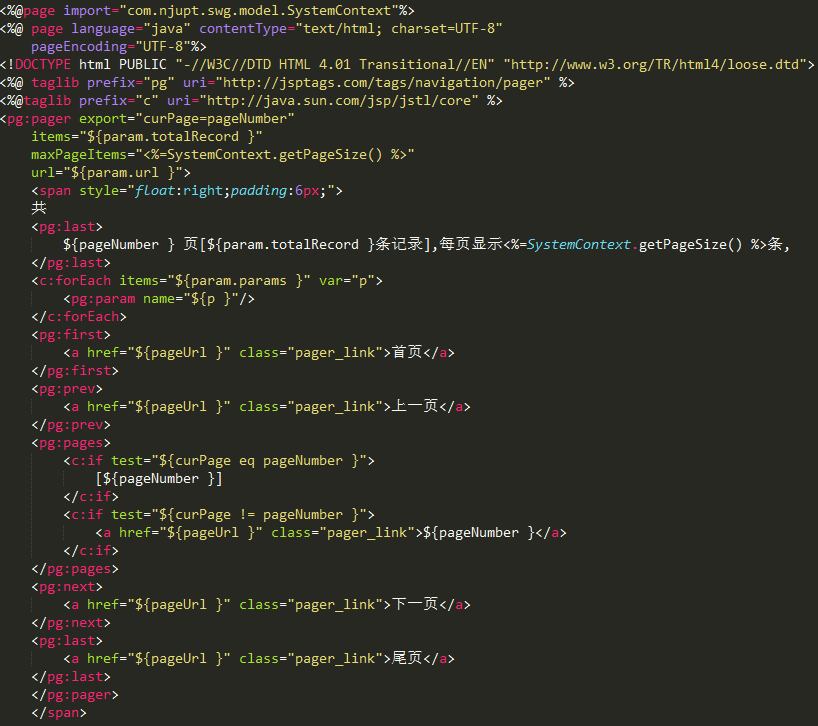
其中，用户状态中，1表示启用，0表示停用。



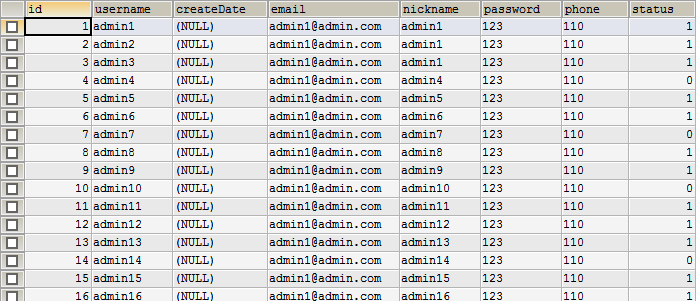
接下来再接再厉，完成分页功能。在这里调试的时候，发现了yige 错误，就是在baseDao这个类中对分页参数进行初始化的时候，误将获取记录数的对象传进去了，而是应该将获取到集合对象传进去：



另外，在刚才的listUser.jsp的页脚处增加分页的显示，主要是将获取到的记录放入进来，作为分页的参数。这里调用pager.jsp来显示。将这个文件放进分页文件中以供查看。



效果：这里有16条数据





至此，基础框架搭建已经全部打通，下面就是完善这个cms的功能。

1. User

昨天，我们将用户的所有信息以列表的形式呈现在页面上，那么我们现在就要对其进行增删改查的处理。

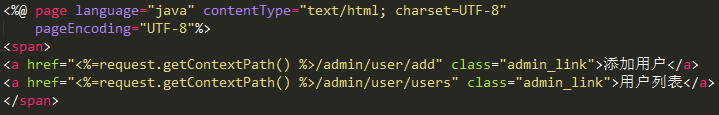
第一步：在页面顶部添加一个公共的连接，效果如下：



这里用一个单独的页面写，然后在其他所有需要的页面上包含进来即可。



这里的inc.jsp页面如下：



第二步：show方法

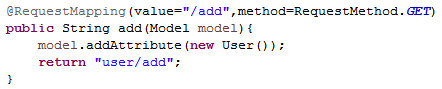
页面部分的链接为：localhost:8888/cms/user/id，所以controller就直接可以获取到id,思路就是首先获取id，然后加载出来，因为是User对象，所以页面可以自动转为user来获取相应的值。



再添加一个show.jsp的页面即可。这里暂时先不涉及到组和角色的关系。后面添加。

第三步：添加用户add

首先是在controller中，增加第一个get的增加方法：

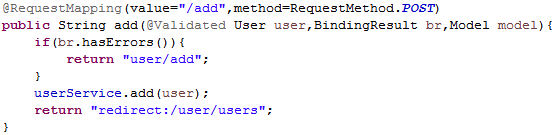


然后增加一个add.jsp的页面，因为在user这个bean中已经添加了相应的验证器，那么页面上利用即可进行简单的验证，比如用户名不能为空，否则不能提交，并且在后面显示错误信息。

状态的值这样传递：



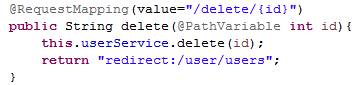
下面，这个页面暂时以user的形式提交，再在后台进行添加：



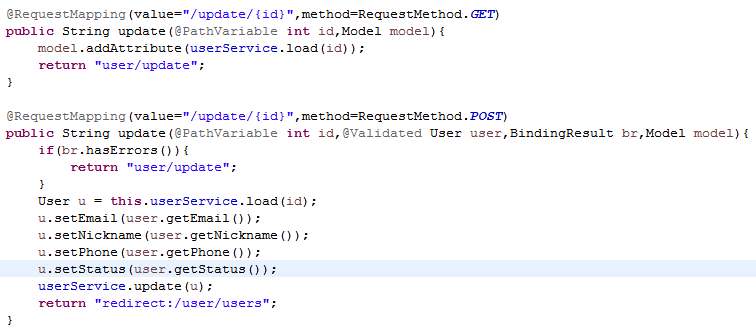
这样就完成了整个用户自身的添加过程。

第三步：delete

现在的删除很简单，只是删除用户本身而已。



第四步：update，与add方法很类似，另外有一个Update.jsp页面。

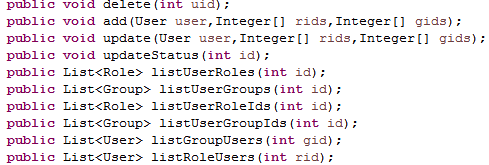


因为这里没有对Username和password进行修改，所以提交的时候，需要将其以隐藏域的形式发送过去，否则就是控制，会有错而无法提交。



至此，最简单的用户单表的增删改查已经全部完成，下面就是将分组和角色添加进来。

1. 在UserService中扩充带有Group和Role的一些方法：

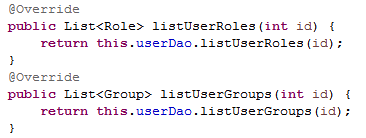


所以我们首先准备好GroupDao和RoleDao中的一些方法，用来调用。暂时里面为空，先把接口暴露出来，需要什么写什么。再在UserService中将这两个接口全部注入进来用来调用。

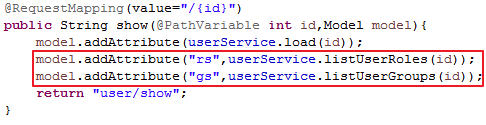
注入完毕之后，实现上面接口中的所有的方法。

第一步，是改动刚才的show方法

首先实现以下两个方法：



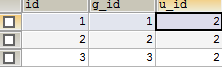
然后再controller层中的show方法中，将其传递过去：



在show.jsp页面中，增加显示分组和角色的信息：



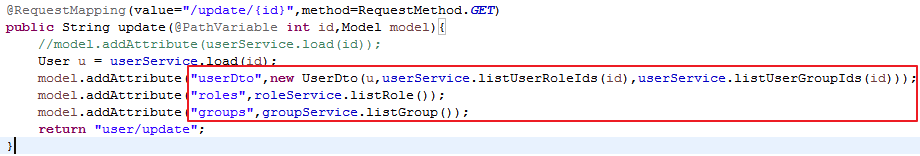
效果：





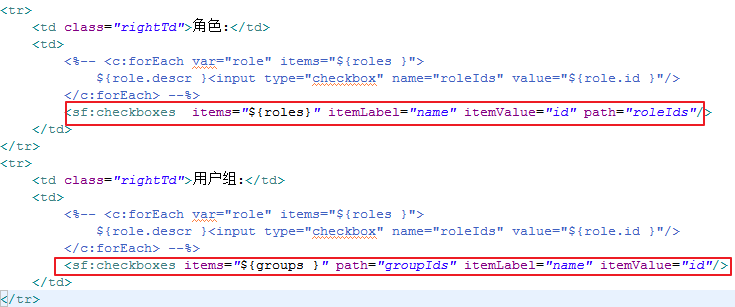
第二步是update方法

将第一个get改为：

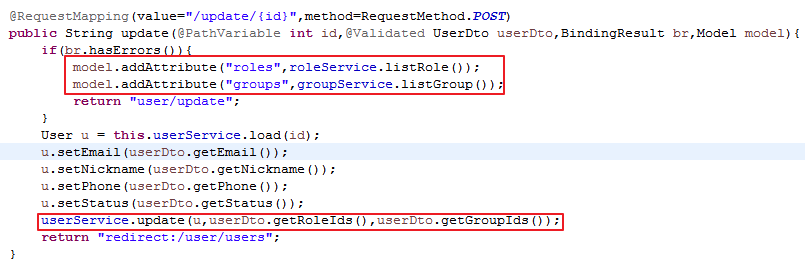


其中，需要借助一个UserDto的类。这个类中，不仅含有User的所有属性，并且还含有rids和gids这些数组。用于存储页面传递过来的值。详见Dto文件下的UserDto中的内容。

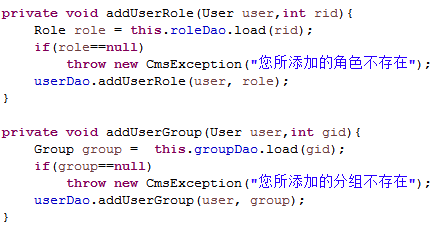
将所有的role和group全部列举出来，用多选框来选。

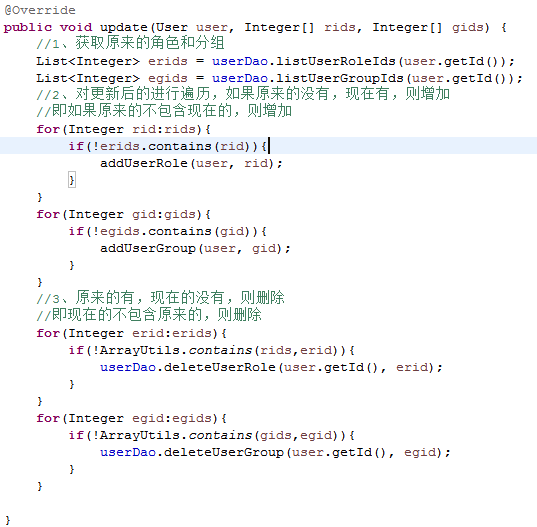


第二个post方法：



这里涉及到比较重要的方法:如何将复选框中原来选中现在不选中的删除，或者是原来没选现在选中的添加进来。

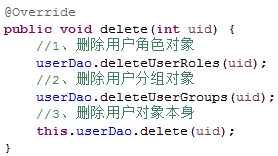




核心的思路就是：获取到数据库中所有的角色和分组的id，然后拿过来和页面新选中的进行对比。

第三步：delete

userController这一层不需要变，只需要将UserService这一层的delete方法变成如下：

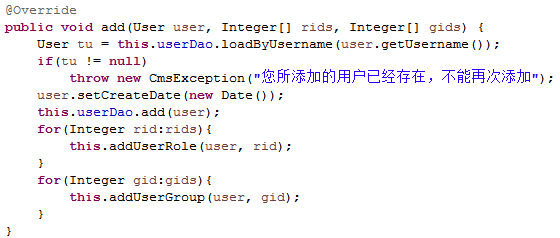


这样，就可以顺利删除对象了。

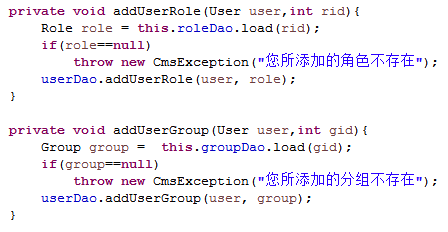
第四步：add

Jsp中增加两行，与update一样。

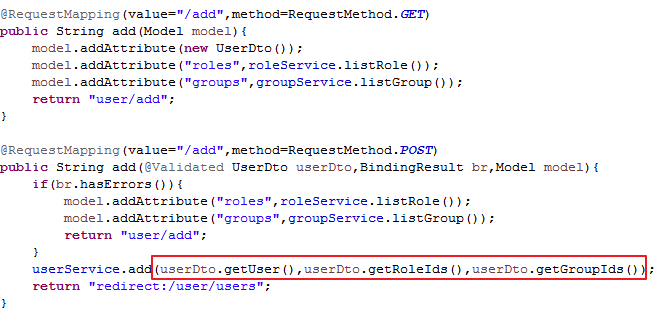
userService中add方法



其中，添加角色和分组的两个方法为：



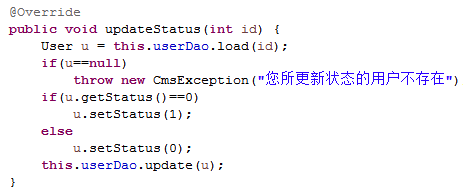
最后在controller层中，用UserDto



这样就完成了添加功能。

第五步：updateStatus

在UserService中的updateStatus方法：



然后再userController这一层中：

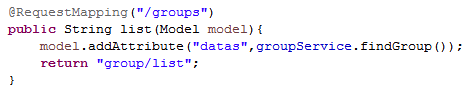


这样，就完成了用户状态更新。基本完成用户这一块的内容。

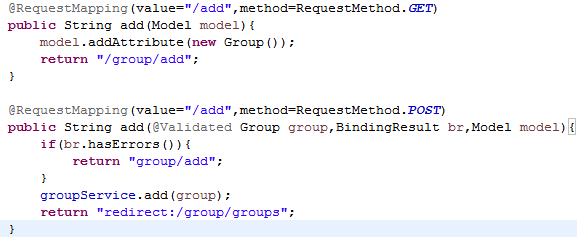
1. 完善分组功能

第一步：list分组列表

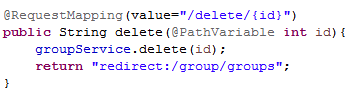
同User，将分页的集合传递过去，不重复赘述



第二步：add增加分组

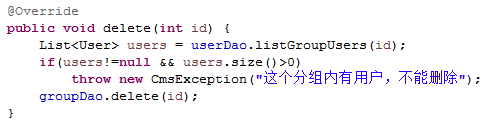


第三步：删除分组

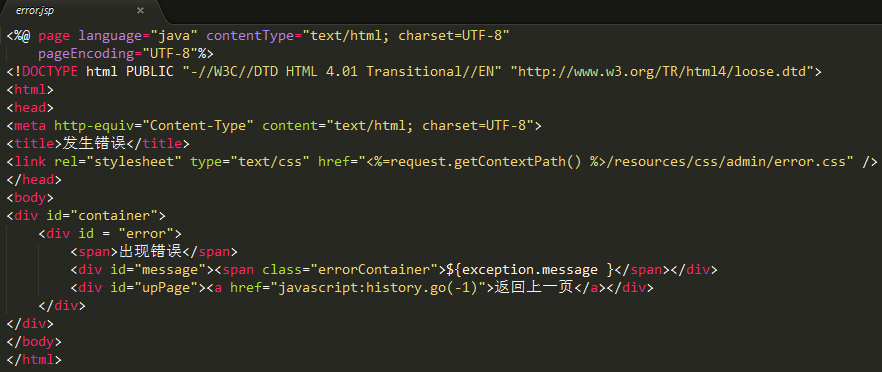


这里涉及到一个问题：假如该分组内还有用户，那么就不能删除，需要在清空用户之后才能删除

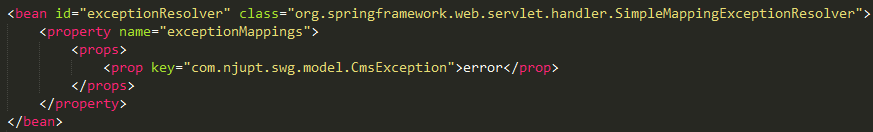
所以在service层的逻辑是这样的：



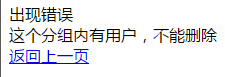
那么现在，假如删除了带有用户的分组，那么就会抛出异常，正常情况下应该跳转到一个错误页面，提示报错信息，然后给一个返回上一层的链接，重新进行操作。所以这里定义了一个error.jsp的错误页面用来跳转:



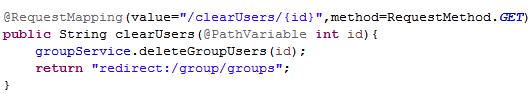
还需要全局异常的一个配置，在cms-servlet的配置文件中增加一下信息：



这样，在发生异常之后，直接跳转到错误页面，效果：

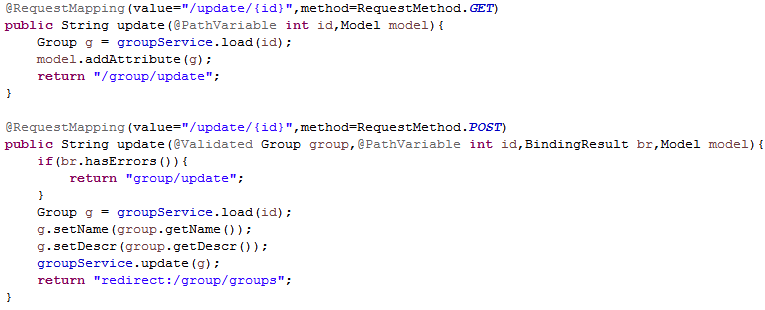


第四步：clearUsers清空分组用户

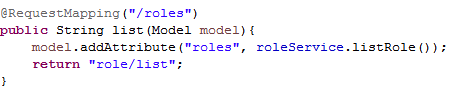


第五步：update更新分组

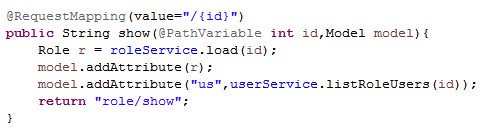
同User的更新。不赘述。栏目管理先放一下，下面先完成角色的基本功能。

10、完善角色功能

第一步：list角色列表功能



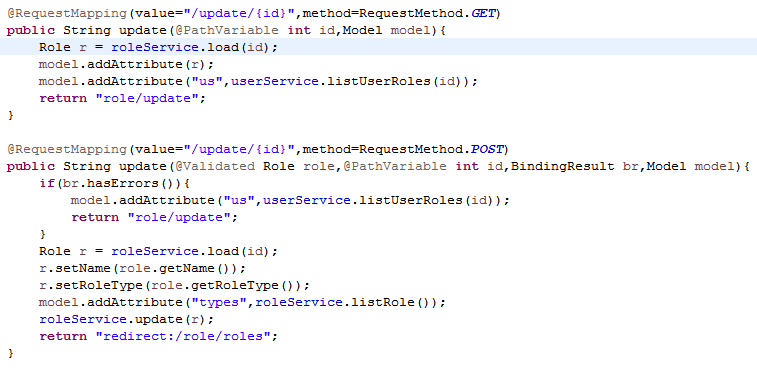
第二步：show角色展现



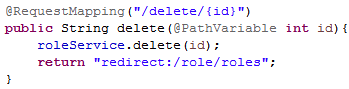
第三步：add添加角色

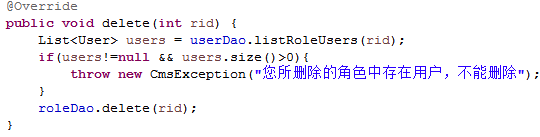


第四步：update更新功能

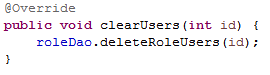


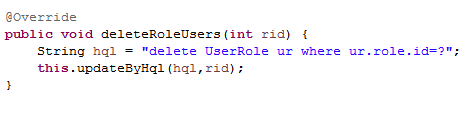
第五步：delete删除功能





第六步：清空角色中的所有用户

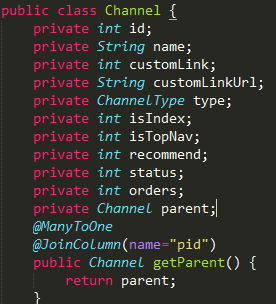




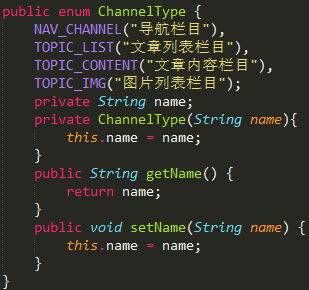
至此，完成了角色的基本功能。下面实现栏目的展现及管理。

10、栏目

第一步：创建Channel类



相应地创建枚举类型的ChannelType

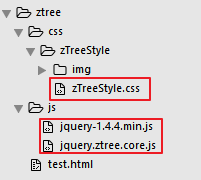


因为涉及到树的操作，所以这里添加一个channelTree的类。

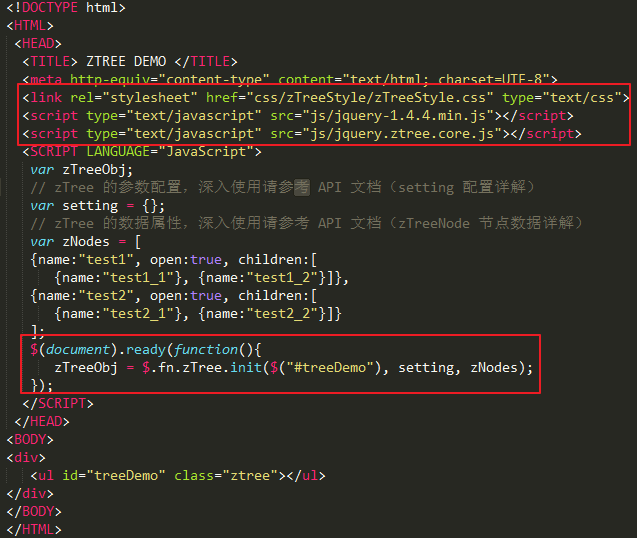


Ztree入门：

到官网下载依赖文件---》拿出里面的js和css文件夹----》创建新的html文件：

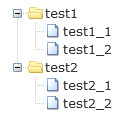


引入css和相应的js的包，并且进行最简单的树的演示。



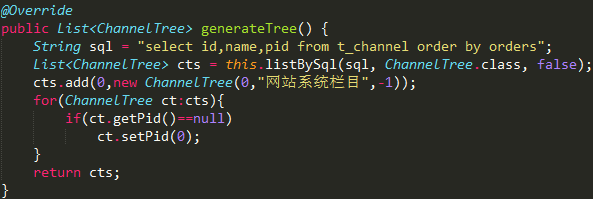
注意，这里的数据源是json格式。

效果：



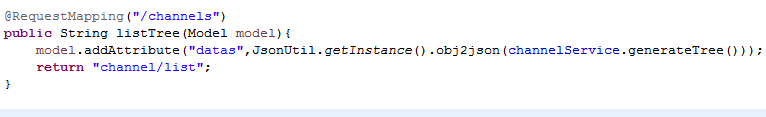
应用：同样的，将以上的css和两个js文件引到相应的位置。

第一步：生成树，根据channelTree里面的三个参数:id,name.pid,在channelDao层:



第二步：因为这样获取到的是list的集合，因为需要时json数据格式，所以这里需要一个工具类，主要是用了jackson来实现，详见本文件夹下的资料。

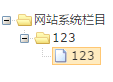
第三步：在channelController这一层中，获取生成的



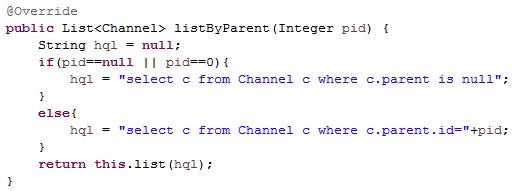
第四步：前台接收这个json数据，显示成树



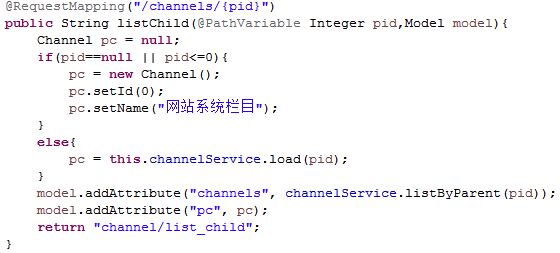
效果：



第五步：显示所有子栏目

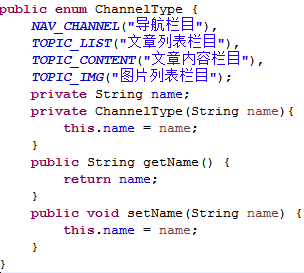


接收从页面传递过来的点击的栏目的id，即父Id，获取到之后先判断是否为空或者为0 ，如果是那就说明是根结点，否则就用Load方法进行加载出来这个父亲节点，用来给页面显示父亲节点的信息。再根据父亲节点获取所有的子节点，传给页面。



第六步：添加子栏目

同样地，还是接收pid，判断是否为空，将pid对应的值传到页面上去显示。这里有一个重要的点：如何将枚举类型以合适的形式显示在页面上。ChannelType如下：



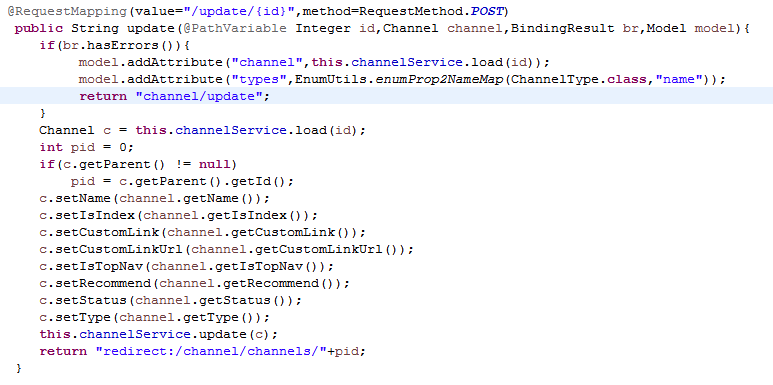
这个方法可以给枚举类型进行相应的标识，括号中的汉字就是其name(key),枚举值就是value.



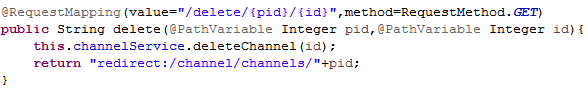
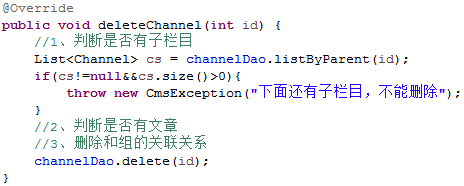
这样，可以将Name中的值传给页面。

第七步：更新栏目





第八步：删除栏目



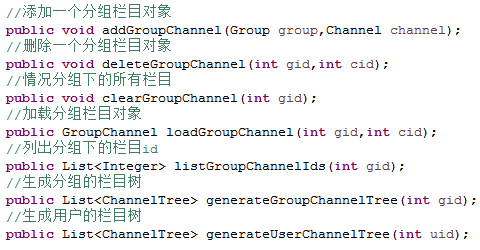
至此，完成了树的展示和栏目的基本操作。

11、组栏目管理

第一步：添加GroupChannel这个关系类



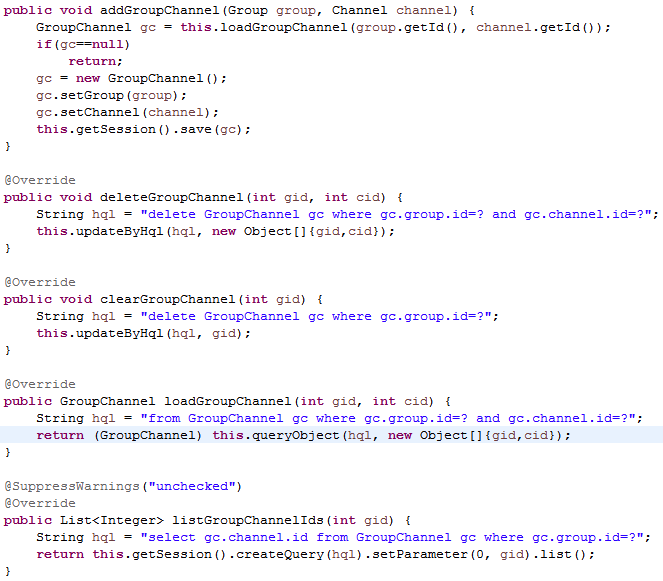
第二步：建立表之后，然后再groupDao中添加关于group和channel的一些方法

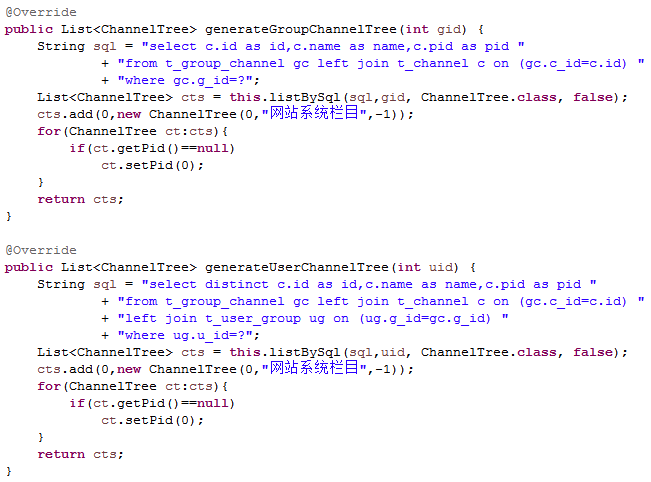


这些方法的实现：

前面几个都比较简单：

后面两个生成树的方法稍微麻烦一点，思路：首先考虑生成分组的栏目树，要生成树，由前面的栏目树可知，需要知道三个参数：id,name,pid,所以这里用sql语句提取出这三个参数，注意，首先从关联表t\_group\_channel中提取出来，然后左连接t\_channe表，连接条件是第一张表的channel的id与第二张表的Id相同，这样，group和channel之间的树就可以提取出来了。下面是生成用户的栏目树，由t\_group\_channel到t\_channel再到t\_user\_group.

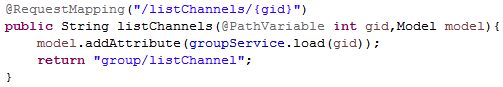




第三步：GroupServcice中创立差不多的方法，调用一下，没有什么逻辑。

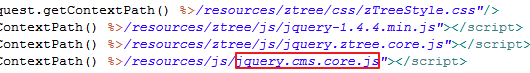
第四步：GroupController中进行处理，首先是listGroupChannels

这里只是先将当前的Group先传递过去。

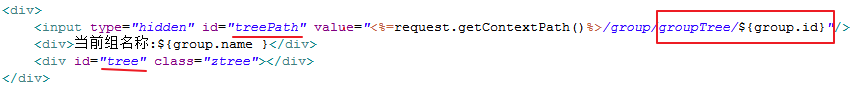


进入到listChannel页面中：

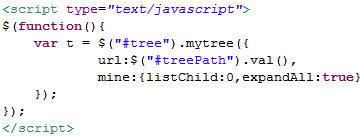
这里呢，已经将ztree树展示方法封装成插件，放在jquery-cms-core.js中，这里引用过来。当然，ztree相关也要引用过来。



进入到这个页面之后，起一个url，将后台关于栏目的信息提取过来。



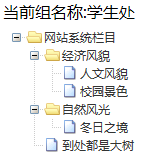
这样，js部分将其展现为一棵树，并且初始全部展开。



整个页面大概为：

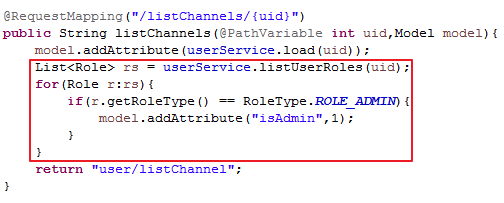


展现的效果为:

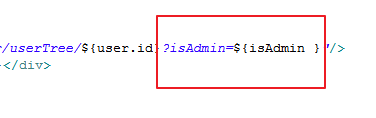


12、用户栏目管理

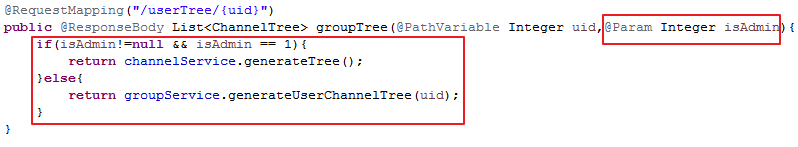
第一步：因为在groupDao和GroupService中已经将生成用户的栏目树已经完成，下面直接在userController这一层中调用即可。但是这里需要考虑一个问题：加入用户的角色是管理员，那么就应该可以看到所有的栏目树，这里就多一个判断。



那么在页面上，基本同上面的group的listChannel的页面，不过在传递url的时候，需要在最后加上isAdmin这个参数，再传递到后台，后台根据这个admin决定是否是展现所有的树。



这样子，后台的展现userTree的方法则变为：

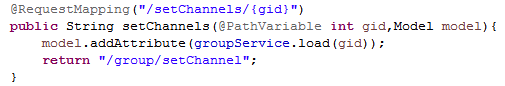


这样，就完成了用户管理栏目的功能。**尤其注意，这里用到了jquery的插件的封装和调用，这一块暂时还不会，但是很重要后面需要好好练习。下面还会大量使用，也是暂时先使用其他人写好的，先完成功能再说。**

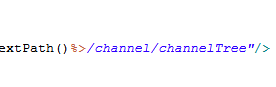
13、ztree树的节点获取

这里完成的功能是对栏目树的编辑。

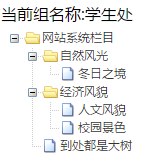
第一步：首先是将所有的栏目显示出来



写一个setChannel.jsp，暂时与之前的listChannel.jsp一样，不同的地方在于调用channel中显示所有栏目树的那个方法：



这样就可以显示所有的栏目了。

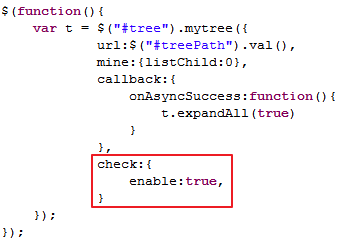


第二步：在每个栏目之前增加可以选择的框。

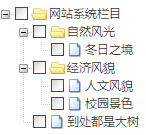
首先是引入一个js文件：



然后，js部分：

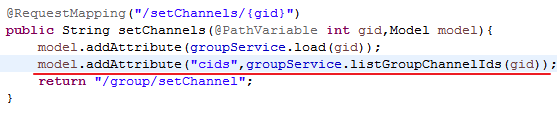


这样，就可以显示出选择框了：

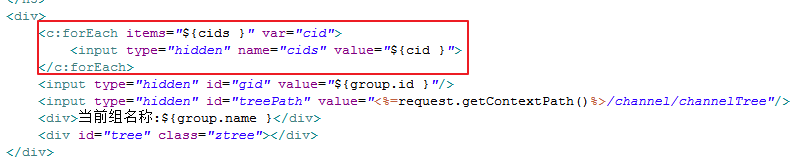


第三步：选中已经该分组中已经拥有的栏目

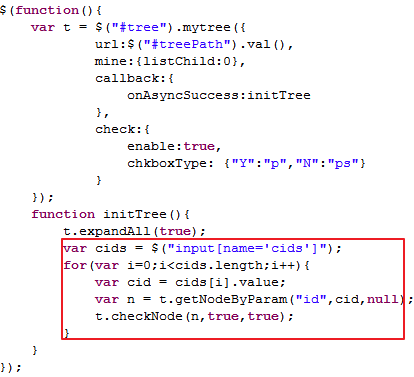
这里首先需要将已经拥有的栏目传到页面上：



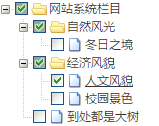
在页面上:



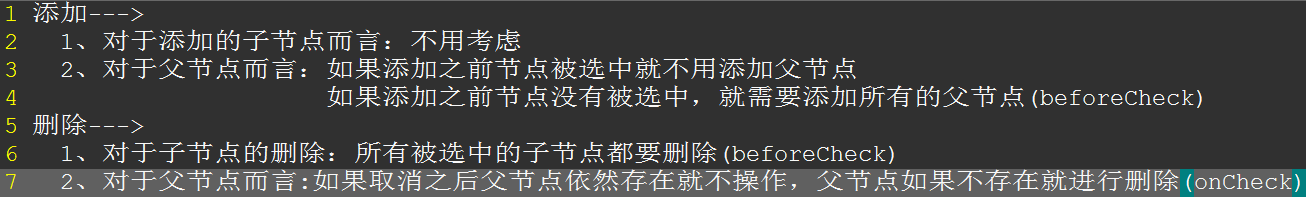
然后，利用ztree中获取节点参数的方法来选中:



这是根据id来选中的，将cid传递进来，然后选中，效果：



第三步：考虑节点点击之后的操作，即删除和增加操作逻辑如下：



对于节点的分析以及增加、删除操作到下面的dwr搞定之后再回头说。

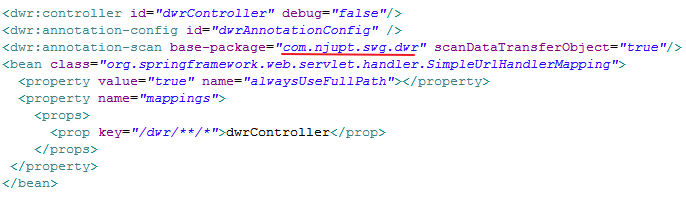
14、DWR

这里直接与spring整合，比较详细的介绍见我的onenote笔记。

* Web-xml

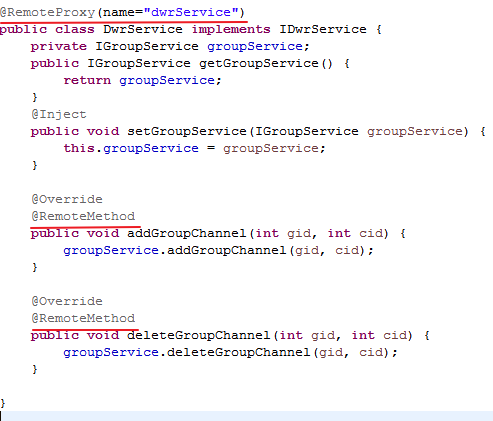


* Cms-servlet.xml,注意还需要两句dtd



* 创建类



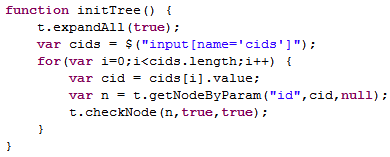
相当于把方法声明出来。

* 页面：先引入两个文件，自带的，写上即可，不许手动添加文件

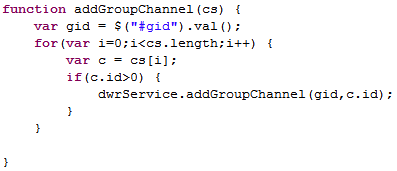


* 下面着重处理一下节点的增加和删除的逻辑，接上。

第一步：初始化栏目树

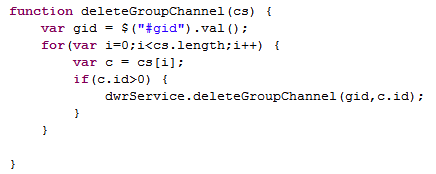


第二步：增加节点



注：这里传递过来的cs既包含了自身，也包含了未被选中的父节点。

第三步：删除节点



注1：删除分为两种情况，第一种情况：取消了一个本身节点，他的父节点仍然处于被选中的状态，则只删除本身；第二种情况：取消了一个本身节点，他的父节点也处于被取消的状态，那么既要删除本身节点，也要删除父节点。

注2:在beforeCheck如果本身节点被取消，一旦父节点也取消，这个时候就会触发onCheck中的方法，将父节点也删除掉。

注3：总之，这里是分为两个状态的，即在点击取消之前看父节点究竟会不会取消，一旦取消，则在点击的时候就删除。看下面的程序，在回调中执行。

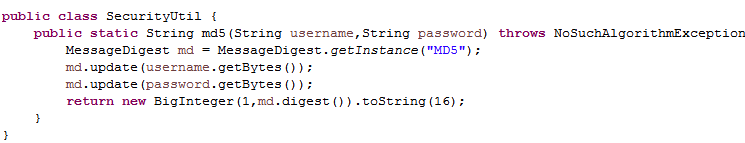
注4:这些js被封装成了一个叫做setChannel.js的插件。



效果：对树进行点击的时候，同时进行数据库的处理。

15、MD5加密

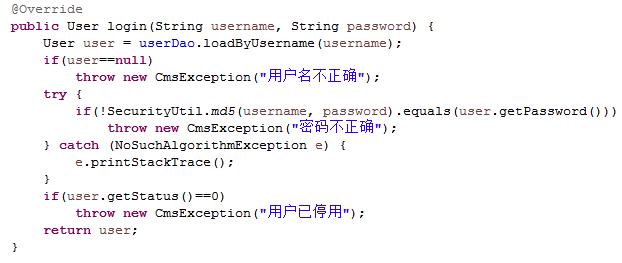
第一步：创建sercurityUtil.java，整合用户名和密码进行加密



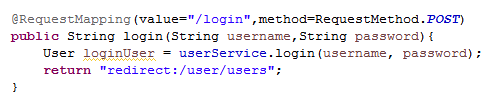
第二步：在userService中修改添加用户，增加一条加密程序



第三步：登陆的时候，进行验证，所以在userService中添加一个login的方法5：

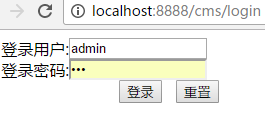


第四步:新建LoginController，用于处理登陆，退出等操作



这里暂时返回用户列表页面。

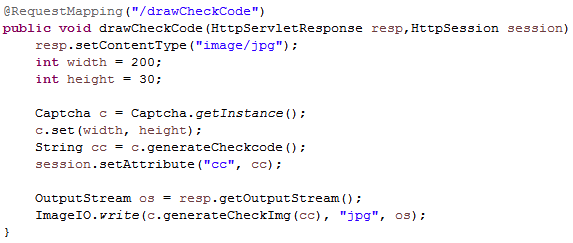
第五步：登陆页面login.jsp,效果为：极其精简的登陆，下面会添加一些验证以及验证码。



16、图形验证码

第一步：创建一个工具类Captcha,这里主要是定义了图形验证码 的一些基本的属性，比如宽度，高度，验证的内容范围，以及显示的验证的个数等。最后就是生成这个验证码图片，主要是获取到一个图片对象，然后用画笔进行写字和增加一些干扰项。

第二步:在LoginController中，进行基本参数的注入，以及获取到相应的验证码图片对象，将其放到一个session中，便于后面登陆的时候调用。最后是一个输出流。

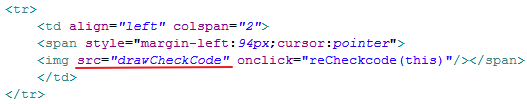


第三步:效果



第四步：将其与登陆进行整合

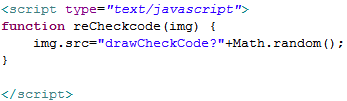
首先是在页面中将这个图片显示在登陆页面上：



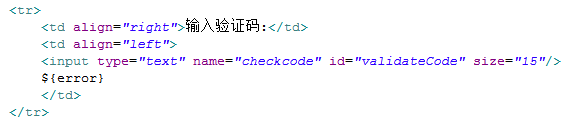
效果：



点击图片一次，就刷新一次验证码：



第五步：增加输入验证码的框

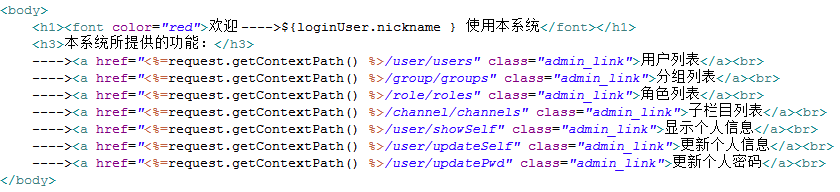


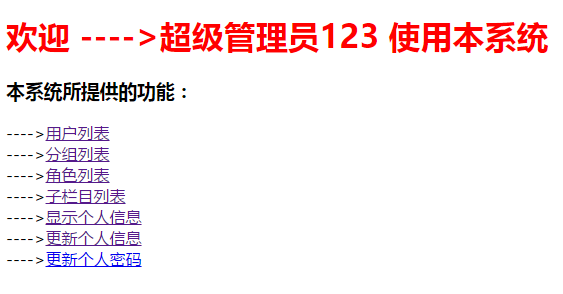
第六步：控制层，验证图形验证码是否一致



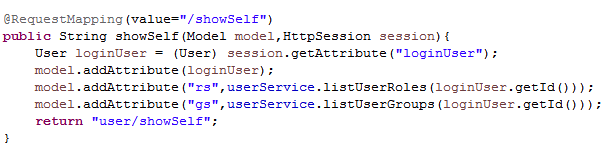
17、显示、更新个人信息、修改个人密码以及退出系统的小功能的完善

第一步：整理了一个Index的页面，提供页面跳转的管理

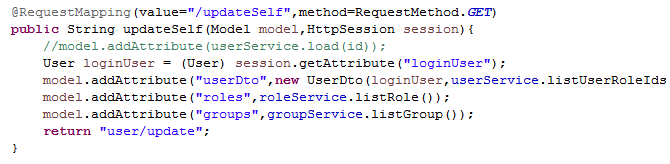




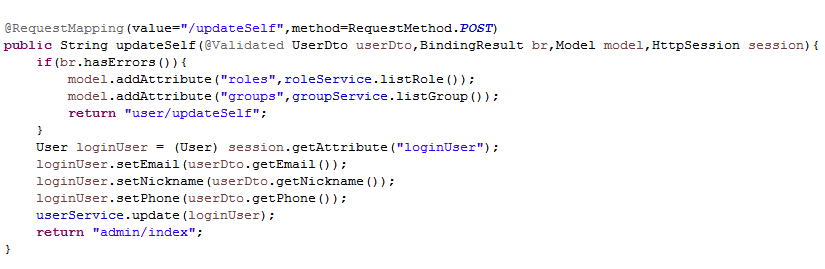
第二步：显示自身信息，页面同之前的show



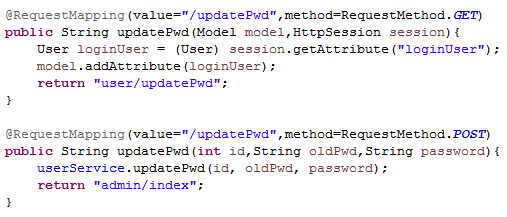
第三步：更新自身信息，基本同update



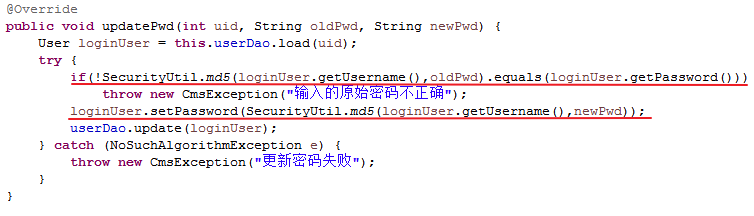
下面就是将更新信息获取到并且存进数据库中：



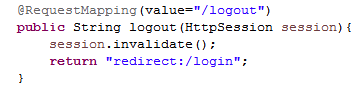
第四步：更新自身密码



其中，service层中更新密码：



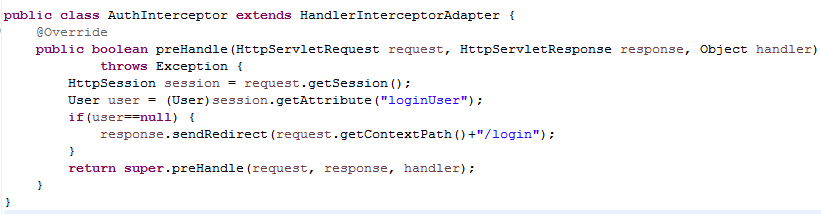
第五步：退出系统



18、检测登陆

第一步：

在springMvc中提供了HandlerInterceptorAdapter的父类，用来进行一些预处理或者其他的处理。这里主要用到它的预处理的方法:preHandle。创建一个AuthInterceptor.java



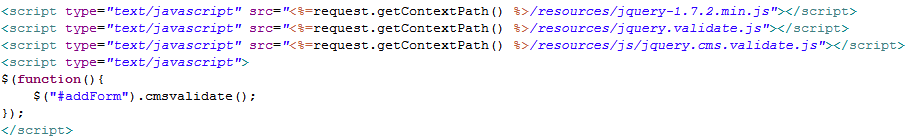
第二步：cms-servlet.xml中配置



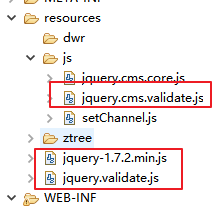
表示对index之后所有的方法都要进行是否已经登陆的验证，那么相应地，在各个controller中的requesMapping后面都要增加/index，让所有的方法都被管理，在没有登陆的情况下是不能通过url直接进入的。

19、完成页面中的所有的前端验证功能

第一步：在相应的jsp页面添加：



引入的三个文件为：



其中js调用的方法为：



第二步：添加错误信息提示：比如我设定输入的Username不能为空

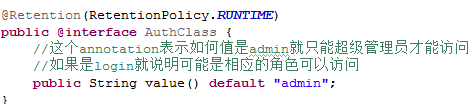


这样，就可以完成最基本的页面的验证了。

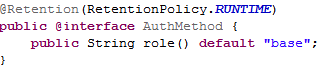
20、权限控制

这里完成一个简单的权限控制，即权限是写死的，初始化的时候就设定好该角色可以访问的方法。

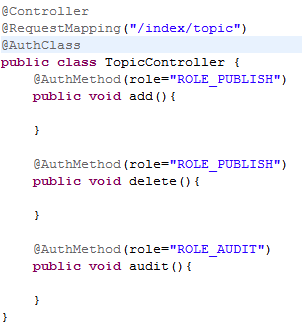
第一步：写一个AuthClass的annotation



这里设定,假如value的值默认是admin，就是超级管理员，那么他可以访问所有的方法。若不是admin，则为普通的login，根据下面的authMethod来确定登陆用户的角色所对应的可以访问的方法。



只要方法加上这个annotation，则默认是可以被所有登陆用户访问，如果值为ROLE\_PUBLISH的话，说明只能被文章发布人员访问，这里为了更好地演示AuthMethod，首先创建TopicController在里面写几个空的方法：



那么，在初始化的时候，就会存在一个大数组，类似于这种：



第二步：AuthUtil类，获取包下面的类以及类中的方法

第三步：创建InitServlet，初始化spring的工厂，在一开始加载权限信息



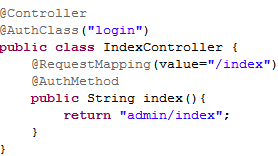
然后在web.xml中，启动这个servlet



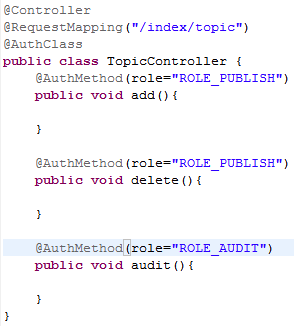
第三步：在每个controller的方法中配置annotation

比如indexController，他应该是所有登陆用户都可以访问到的，所以是：@AuthClass(“login”)

并且index这个方法也应该被包含进base中，用@AuthMethod.



至于栏目、分组、用户列表、角色列表等都只能是管理员访问，在这些头部直接加@AuthClass表明是管理员身份访问的。但是对于UserController中，有一些方法仍然是登陆者都能看到的，比如修改自己的信息、展现自己的信息等。那么对于这些方法，需要在每个方法的头部增加@AuthMethod将其放进base中。对于文章来说，需要更加细分，比如有的是文章发布人员才能看到的，有的是文章管理管理人员才能看到的，这里根据角色进行细分：



这样，在初始化的过程中，每个方法属于哪种role:base还是role\_publish还是role\_audit?

将这三种角色的信息在初始化的阶段全部存进session之中。在登陆时，首先是获取登录用户，根据登陆用户获取是否是管理员，如果不是管理员，则将这个用户角色对应的方法提取出来，并且还要加一个Base，送给拦截器。所以在LoginController的login之中：



第四步：之前设置所有未登陆，所有页面都跳转到login的springMVC的拦截器配置中，在获取到登陆人员之后，判断是否为空，若为空，则跳转到登陆页面；若不为空，则先判断是否为超级管理员，若不是超级管理员，就获取登陆用户的角色对应的方法，并且获取访问链接的后缀即方法，一旦这个方法不是登陆用户的角色对应的方法中的，就说明没有权限访问。



并且在cms-servlet.xml中配置这个拦截器。

第五步：在页面上判断是否是管理员，然后相应地显示所能访问的链接。

用c标签：



判断：



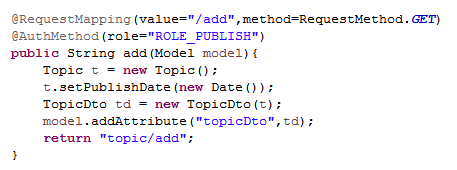
至此，简单的权限控制功能完成。

21、文章添加

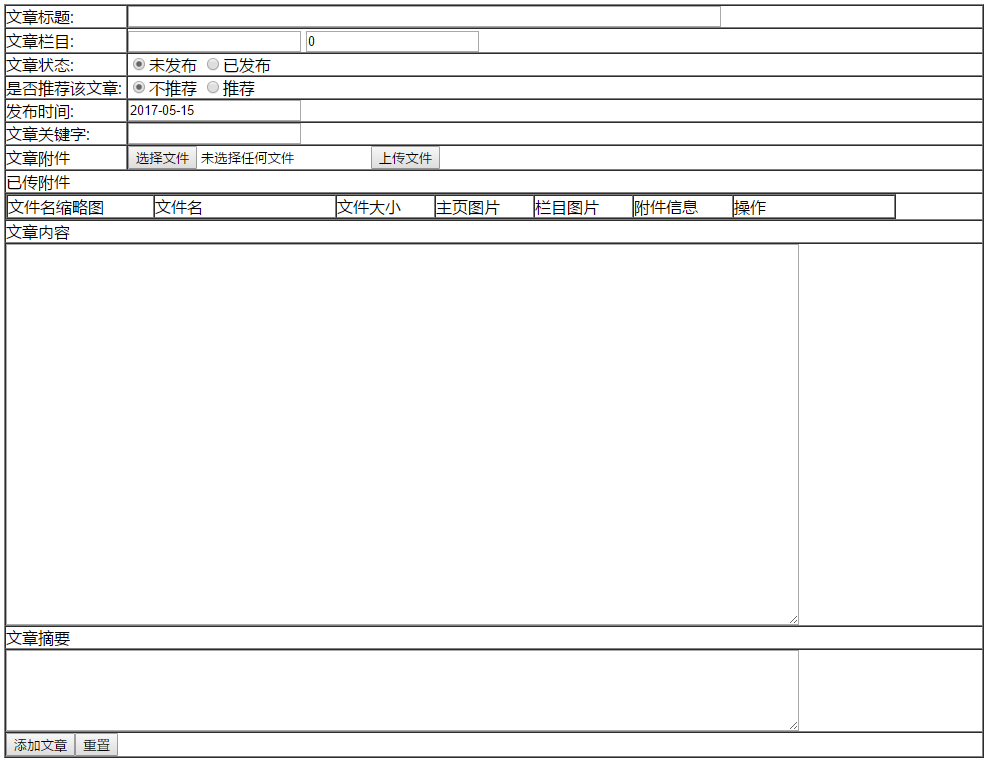
此功能模块比较庞大，做好心理准备。

第一步：topic文章的添加页面

因为在文章的添加页面中，涉及到关于文章的一些属性，也涉及到栏目、关键字等属性。光一个Topic是不够方便的，这里使用TopicDto，将这些属性全部放在一个类中。并且用构造函数的方式，将属性值注入进来。添加的controller中就是构造出TopicDto对象，用来接收来自页面的值。



让我们回到jsp页面。初始的页面如下：包括文章的标题、栏目树的选择、文章状态、是否推荐文章、发布时间、关键字、附件、附件的详细信息、文章内容和文章摘要。



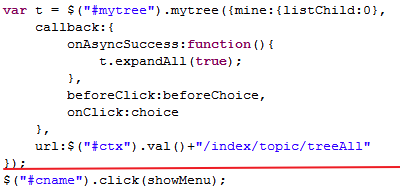
其中，文章所属栏目的选择，是通过点击第一个框，然后跳出一个栏目树，叶子节点的栏目可以被选中并且在后面的一个只读框中显示出选中的栏目的标识。实现过程：

首先是引入ztree的样式和js



写一个div用来显示栏目树，id是menuContent,并且一开始是display:none。Div中是id为mytree的ul。文章栏目后面的框的id是cid,所以下面就是写js，在topicAdd.js中：

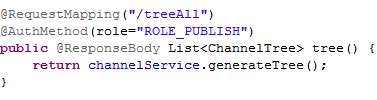
首先是将整个ztree传给这个div:



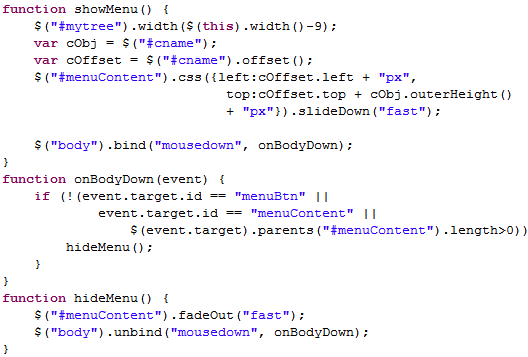
页面中奖当前的路径以隐藏域的形式传给js方便到找到路径。



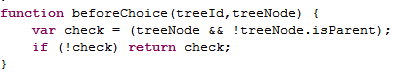
这里的一个路径就是从后台controller中获取到整个树:



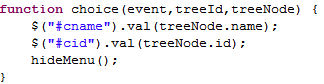
这样，就成功地将整个树传给了div。一开始是不显示的，所以需要一个点击事件来触发显示整个树。如上的click方法，里面有一个showmenu的方法。以及如何隐藏栏目树的实现。



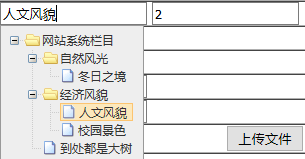
在实现点击展开整个树之后，我们需要进行两个状态的判断，一个事beforeChoice，还有一个choice。前者是指在栏目树展开后选择某个具体栏目操作之前的工作：加入有已经被选择的栏目，即在同一个页面第二次打开这个栏目树，那么之前选中的那个栏目显示被选中的状态，这里还有一点需要注意的是不能为父栏目，通过isParent这个属性判断自己本身下面是否还有子栏目，如果有，是不能被选中的。



在选中之后，我们需要获取被选中栏目的名称以及标识，一个是友好，一个是用来方便存储。因为构造的树只有三个属性，就是id,name,pid，所以将节点的name和id放进前台的两个text中就可以显示了。



效果：没点击之前，什么都没有，点击之后：

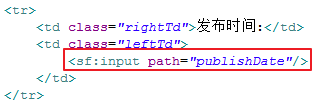


第二个完成日历的显示，用来设定发布的时间：

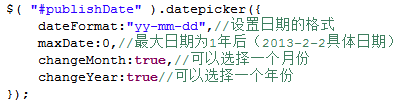
首先是引入css和js:

然后页面上写一个td



对这个日历进行一些小的设置：



这里一开始，如果没有设置好日期的格式，进来的时候是这样的：

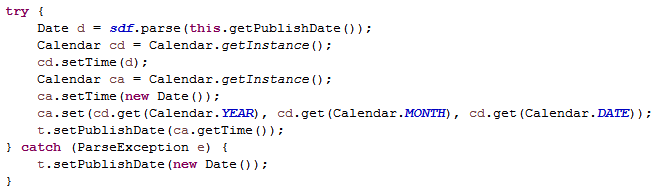


显然是不好的，这里的处理办法是在TopicDto之中对publishDate这个属性进行格式的变换：







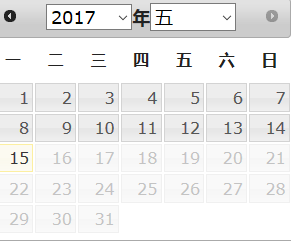


总而言之，就是用simpleDateFormat对日期进行一个格式的变化，所以定义的publishDate是一个字符串类型的。所以最后选中的日期是以字符串的形式设置进去的。

这样进来的效果是：



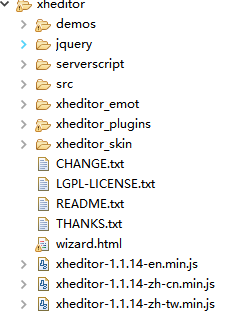
点击展开日历：



第三个就是xheditor的文本编辑器：

将相关的文件包导进来：





页面上用一个path为comtent的textarea来作为文本框内容输入区域。Js中就是设置其完全显示即可：

效果：



这样就完成了xheditor文本输入框的设置。

第四个：文件上传操作

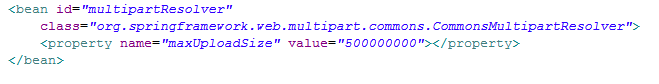
首先在页面中显示文件上传的file



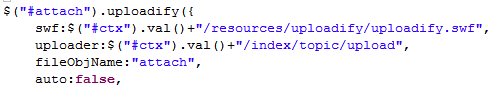
导入uploadify的文件夹、css、js等，注意在之前要确保commous-io、upload的jar包要被引入进来：



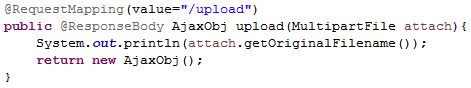
然后要在cms-servlet之中配置：



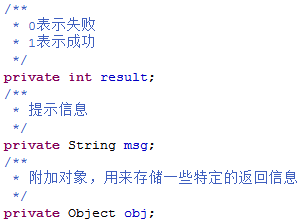
Js中，配置相应的事件：



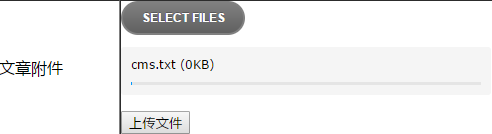
这里的auto是指不自动完成上传，等待按钮确认之后再上传。相应地在topicController中：



因为一个字符串肯定是不足够表示一个附件的，并且附件上传之后是一个json数据，里面的属性应该包括对象和一些标识信息。这里的ajaxDto:



效果：

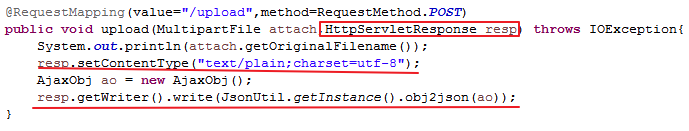


这个时候会报错，说是无法找到cancel.jpg这个图片。那么我们就新建文件夹img，将这个图片放进去即可。



这样，就有了文件上传的基本样式。下面就是点击按钮，触发上传操作。获取到上传的文件的状态信息。

首先改写upload方法：这里只是测试如何将后台的json数据传到前台显示，这对于后面的意义重大，因为后面真正上传的数据中有一个obj，这个就是存储的附件本身对象，我们不仅要将其存到数据库和服务器中，也要在前台显示信息（如果附件是图片的话，就需要显示出来）。

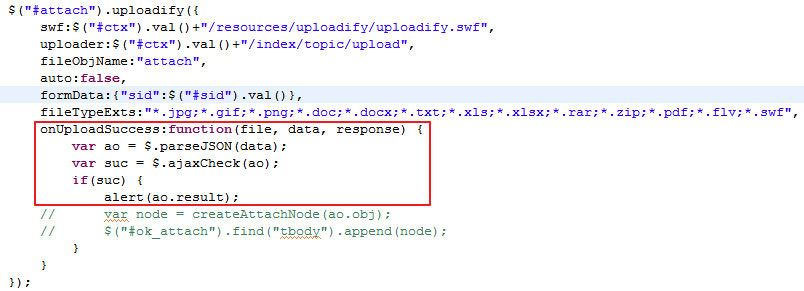


因为uploadify不能接受json，只能接受字符串。所以这里将对象转换为json数据格式之后，用resp以字符串的形式传给前台。前台的接收返回的数据之后，再将其转为json格式。这个就很方便，js本身就有转换工具。

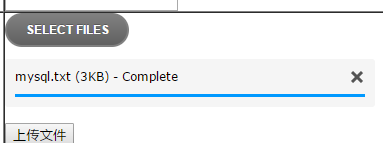
点击按钮确认上传：



一旦上传成功，触发onUploadSuccess函数，将返回值调出来。先转为json,再显示这里的result，默认为1说明上传没有问题了。



效果：

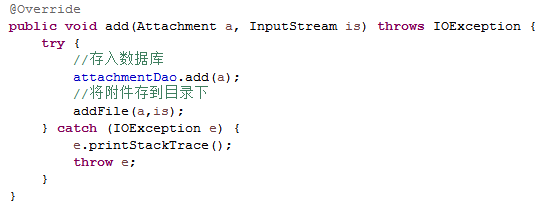


下面是完成附件的整个添加：

上面以及可以在前台拿到上传的文件对象，下面是对这个文件对象进行处理，并且分为普通的文本附件和图片附件两种。完整的controller为：



其中的service层：



其中的addFile为：



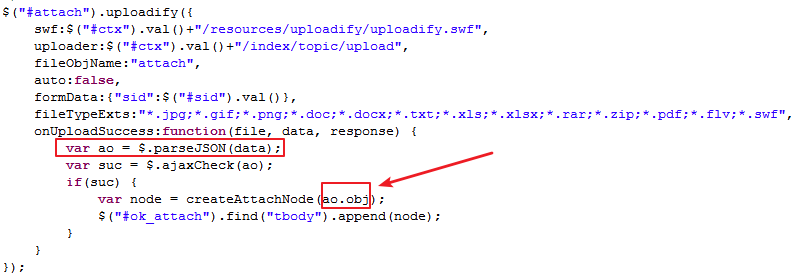
以上包含缩略图的处理：thumbnaliator

这里完成一个效果是加入附件是一张图片的时候，那么我们会再下面的附件具体信息栏显示一张缩略图，否则 就是普通附件。

首先是在service这一层的pom中引入thumbnaliator和common-io。

然后根据图片的流将其进行等比例压缩和切割。

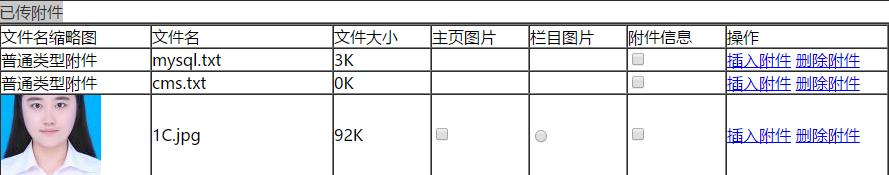
这样，整个附件流的信息包装再json转成的字符串中，在前台再将其转为json拿出其中的附件对象即可。这部分是js完成的：



拿到附件对象之后，将其属性再逐一显示出来，然后拼接在一起：

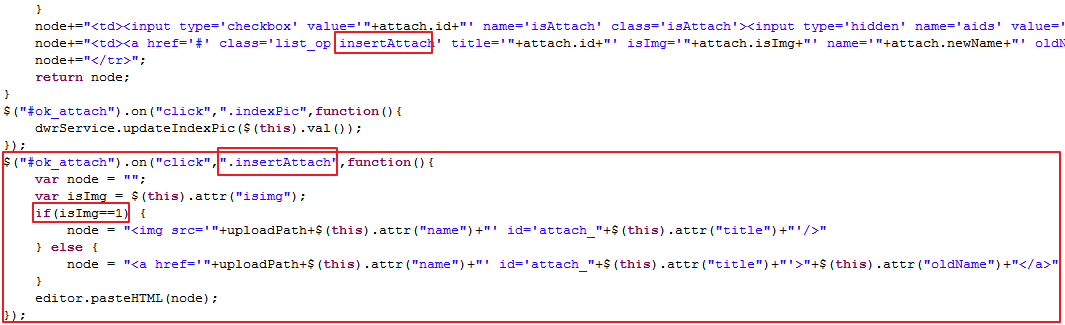


效果：

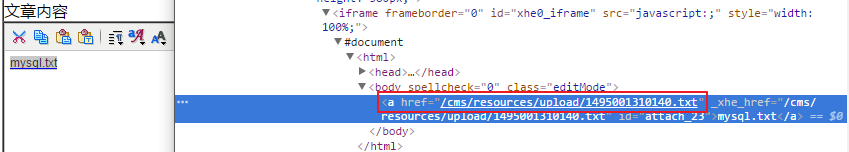


插入附件：

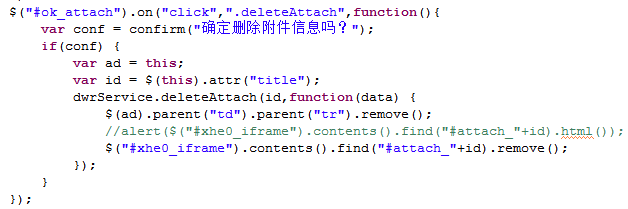
这个功能实现的是将附件插入到编辑器里面：



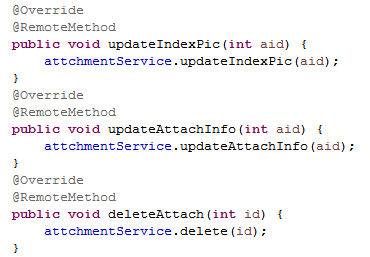
核心思想就是找到这个附件的上传的物理地址，即这到附件对象本身，然后通过parseHTML方法将这个附件或者附件的链接显示在页面上。



删除附件：不仅要删除附件本身，还要删除节点，使之不处于已传附件的列表中：



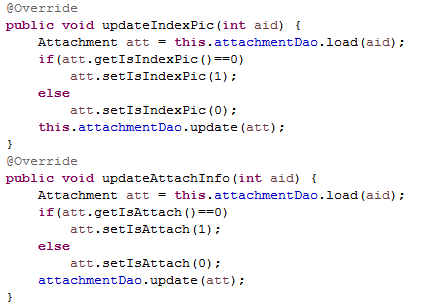
要再页面完成这种无刷新的操作，需要借助dwr:



attachmentService中：

删除操作：先删除数据库再删除目录下文件





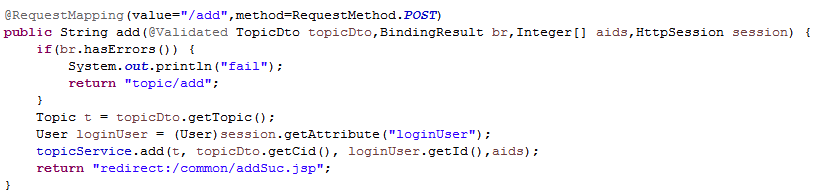
这样，就可以完成附件的无刷新的删除了。

同样地，可以再页面进行无刷新的更新是否是主页图片、是否是附件信息。





第五个：文章的提交



即将登陆用户信息、文章对象、栏目信息、附件信息全部添加进去：

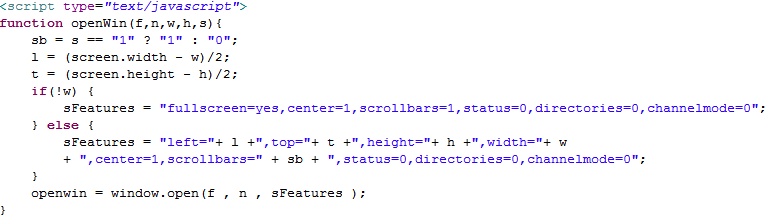


这样就完成了整个文章的添加过程。

第六个：单独页面的跳出和关闭、前台的简单验证

对于跳出一个新页面来进行添加文章的操作：





关闭页面：

这里可以用一个定时器来确定多长时间后关闭这个页面、也可以直接手动关闭：

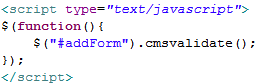


对于文章前台的简单验证：

同上：引入validate:



然后js:

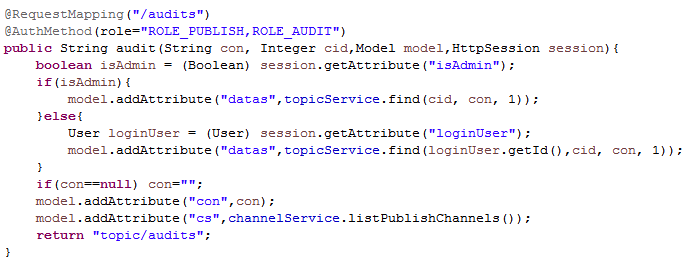


这样就完整地实现了文章的添加功能！

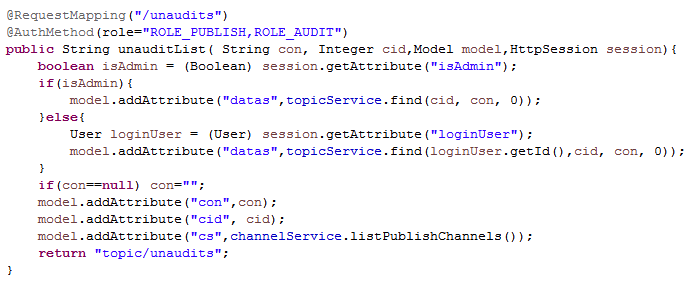
22、文章列表

分为已发布和未发布两种

已发布：



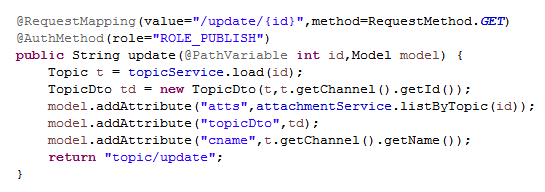
未发布：

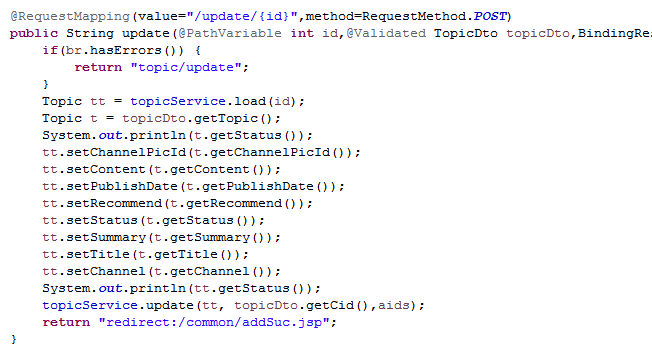


思想同上，不赘述。

23、文章的更新

也是同理

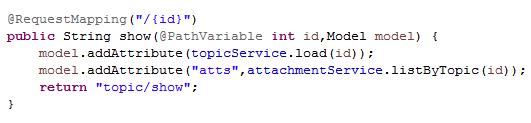




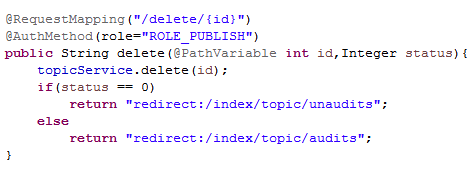
最后返回成功页面。

24、文章的详细信息展示和删除

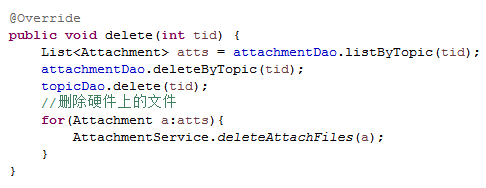
Show:



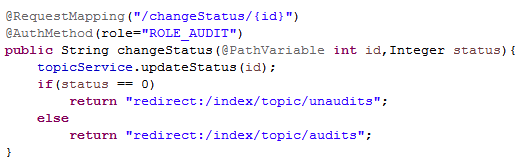
Delete:

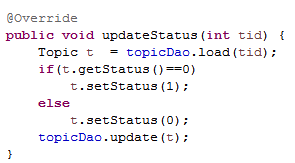


思路是先删除数据库中附件、再删除数据库中文章，最后删除目录下附件。

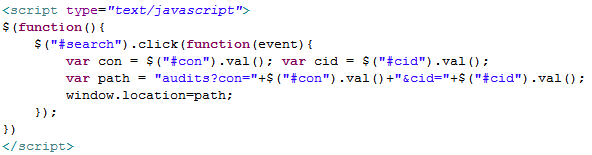


更改文章状态，即未发布改为已发布，已发布改为未发布。





最后就是搜索功能：



将这个值传递给audits和unaudits，让他们显示出对应的列表。

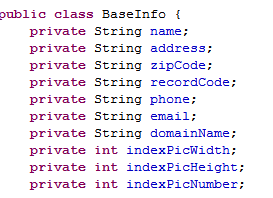
至此，文章的主要功能全部完成。

25、展示文章基本信息和修改

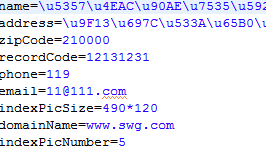
形如：



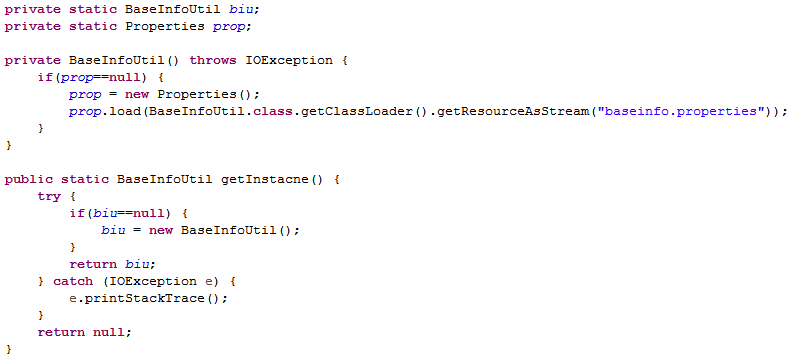
第一步：定义一个属性类：



第二步：写一个属性文件，确定默认信息



第三步：写一个读配置文件和写配置文件的类



首先是获取到配置文件以及创建一个实例。

读配置文件：



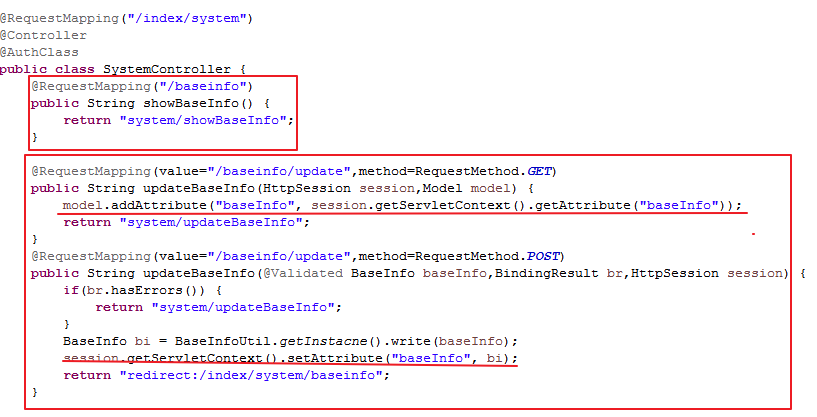
写配置文件：



第四步：在初始化的时候将这个文件塞进session



第五步：控制类



第六步：写相应的Jsp页面。

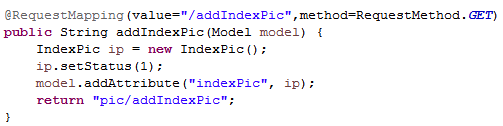
略。

26、首页图片管理

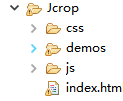
第一步：添加首页宣传图片

首先是创建IndexPic的类，涉及：标题、子标题、首页的链接类型、链接地址、图片的新名称、图片的原始名称、图片的状态、位置。

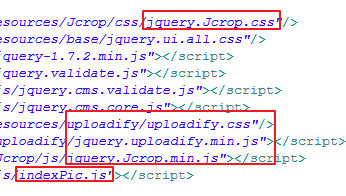
然后再控制类中：将空对象传到页面上：



然后完成一个页面：addIndexPic,这里的重点是对一张有效图片的截取和保存，作为缩略图使用。用到Jcrop,所以先将对应的文件复制到对应的目录下：



所需要导入的js和css如下：



其中，核心是indexPic.js这个文件。

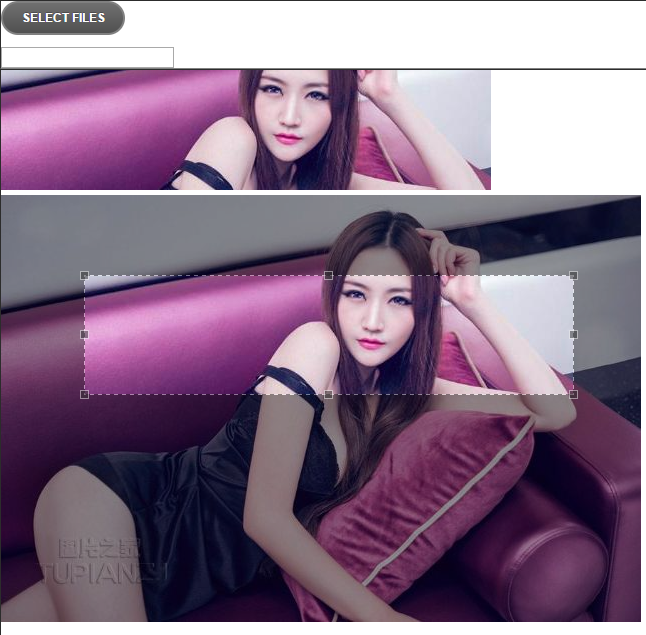
尤其注意下面这一句需要：



这样，就可以进行下一步的上传了：



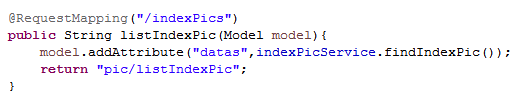
这里是对上传的图片进行了一些验证，并且根据位置的不同，获取到不同的缩略图，将其存到ipd中待传到前台。结合js形成一下效果：



然后就是确认选中的区域：将原图和缩略图分别写到不同的位置。



这样，基本上添加的主要功能就完成了。下面是将添加的首页图片以列表的形式展现出来。

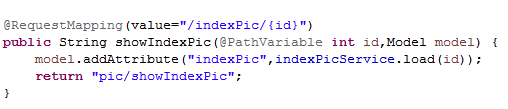


这样，就可以显示出来：



第二步：首页图片的show、update、delete等简单方法：

Show:



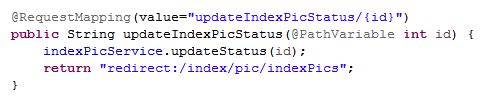
Delete:



Update:

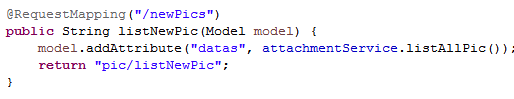


更改状态：



第三步：显示首页新闻图片：

就是将发表文章中附件为图片的显示出来而已：



效果：



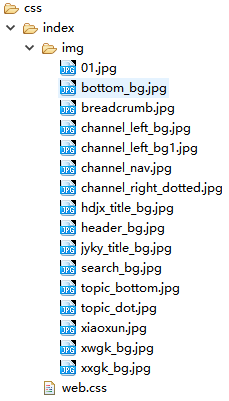
28、超链接管理

29、网站清理

30、文章数据备份

至此，后台管理功能全部实现。‘

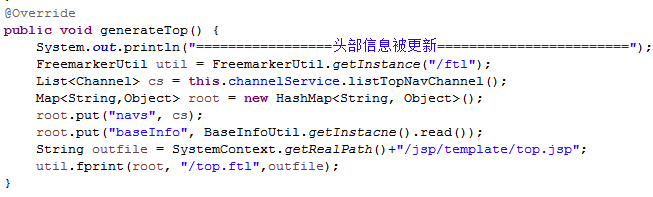
前台，需要写CSS，略过，直接复制。



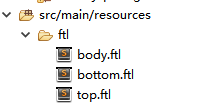
因为网站往往被分为上中下三个部分，所以要写三个ftl文件生成对应的jsp文件：

第一步：头部

写了一个IndeService这个类，用来实现freemarker的控制：



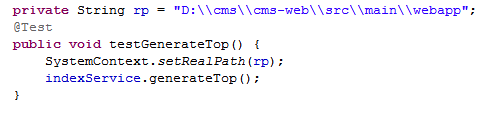
这里找到ftl下的ftl文件：



其中，top.ftl是：



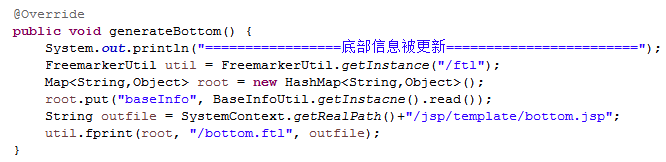
接下来写一个测试类：根据这个ftl文件生成对应的jsp:



对应地，就可以取出相应的东西显示在页面上了：



这样，一个ftl静态化页面就被搞定了，下面依次是body和bottom:





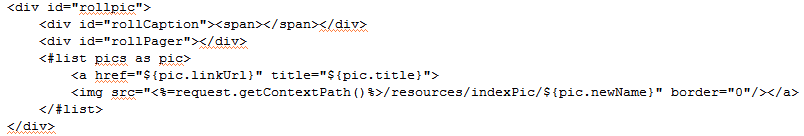
这里的方法触发就不用aop了，直接再controller层对应操作的地方增加这个方法，就可以实现页面的自动更新了。

在这里要注意的一点是首页滚动图片的实现：

首先是引入下面两个js文件：



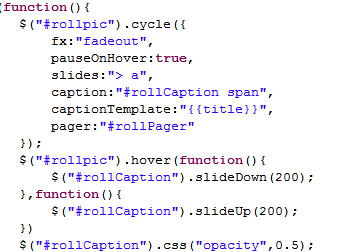
由ftl文件：



到生成的Jsp文件：



再到对应的Js: 即main.js：



这样，就实现了首页的滚动图片的效果。

至于程序中generateBody方法，因为要显示很多的栏目，而每个栏目显示的内容不一样多，所以用了一个配置文件，读进来：

**这里涉及到一个读和写配置文件的类。前台内容不作过多藐视，至此结束。**