简版CMS笔记

1. 登录，链接数据库验证用户名，密码等
2. 更改后台显示的用户名，很多请求共用一个变量，用session
3. 退出系统清除session
4. 频道管理，反射机制，baseservlet中央控制器
5. ChannelDao用来链接数据库，执行sql语句， ChannelServlet对外提供增删改查接口
6. 登录数据库表

create table t\_admin(

id int primary key auto\_increment,

username varchar(20),

password varchar(20)

);

1. 频道管理数据库设计

create table t\_channel(

id varchar(20) primary key,

name varchar(50),

description longtext

);

1. 文章管理数据库设计：

create table t\_article(

id int primary key auto\_increment,

title varchar(200),

source varchar(200),

author varchar(200),

content longtext,

create\_time datetime,

update\_time datetime,

publish\_time datetime,

channel\_id varchar(20),

is\_header\_line boolean,

is\_recommend boolean,

click\_score int,

msg\_count int

);

//is\_header\_line char(1), 是否头条

//is\_recommend char(1), 是否推荐

//msg\_count int 留言数

//设置外键

alter table t\_article add constraint foreign key (channel\_id) references t\_channel(id);

判断复选框和下拉列表是否选中，用到三目运算符

select \* from t\_article where title like ?

pstmt.setString(1, "%"+titleKey+"%");

1. 文章管理编码顺序：

添加，查询，删除；编辑，发布；搜索；全选；多删；分页（数据库分页）

1. 链接也可以是Javascript代码
2. 分页：select \* from t\_article limit ?,? 第一个代表起始索引（默认编号从零开始） 第二个代表页面大小 分页的关键在于记录，页号等之间的换算
3. 抽象出一个PageModel对象，统一管理分页的一些属性
4. 单个删除时请求提交时只传了id，删除后页面Article\_list重新读取，要想显示与删除前Article\_list显示一致的话，需要把整个表单提交上去，ArticleServlet 里的del() 是forword 方式跳转，提交的titlekey pageNo 不会丢失
5. 编辑采用打开新窗口的方式，使与提交之前的列表保持一致（有问题：之后显示的页面都在新页面）
6. 检索提交之前把显示页号置为1，批量删除后也回到第一页
7. 选择分页大小，设置pageSize，以及选中
8. 特殊情况下首页，上页，尾页，下页不能使用，添加限制条件；显示的页数控制
9. 解决缓存问题两种方法，一是在访问地址后面加一个随机数，二是在jsp中加<meta>标签
10. 编辑页面提交后关闭编辑窗口（刷新父窗口问题没有解决）
11. 编辑文章和添加文章获取频道信息的时机不一样，添加文章需要一个预先读取频道信息，left.jsp点击“添加文章”到articleservlet addInput()到add\_article.jsp 到articleservlet add()，add调用ArticleDao save()，而编辑是到articleservlet modifyInput() （此时读取频道信息） 到update\_input.jsp到

articleservlet modify() ,modify() 调用ArticleDao update()

1. 一个A.jsp页面包含B.jsp页面，静态包含时

B 中值需要写编码格式

B中链接的路径都是在A基础上的，只要B不设置basepath

<%@include file=”” %> 需要服务器端解析，所以其路径与url-patten，forword一样，绝对路径的第一个“/”是代表到项目名

1. 动态sql，重载ArticleDao里的query()
2. Math.random 和 new Random().nextInt()区别
3. 什么时候用整型包装，什么时候用向下转型
4. 前台频道内容显示也要分页，所以要用到PageModel，把查询到的第一页显示出来，query()与getRowCount()不冲突，查询方法有参数pm，在其中有获得总记录数的代码，获得总记录数时按照查询条件获得记录数，不涉及分页
5. 注意：文章发表了才可以在前台显示
6. 把getPageInfo封装起来
7. 控制头条显示的信息的多少，链接有悬停提示 title
8. topic.jsp 类似index.jsp，只不过是动态加载的servlet少了，topic.jsp主要动态加载 IndexServlet 里的topic方法，在include 到topic\_list.jsp
9. 大部分网站上的分页是可传阅的分页，即每一页都对应地址栏一个地址，换句话说就是每次都是一个新的请求，而不是跳转，我们用的服务器端跳转没有地址栏的变化，所以造成了不可传阅
10. 动态包含写的page里写的地址不能传动态的参数

page=”topic.jsp?m=topic&channelId=<%=request.getparameter(“channelId”)%>”这种写法中传动态的channelId是不可以的

1. 跨表连接，现在两个表关联上了，可以依据Article表里的channel\_Id到Channel表里查的对应的Channel的名称等信息
2. 更新点击量，只是单纯的视为请求一次查看文章内容点击量加1，刷新页面也加1
3. 后台编辑页面美化，加入ckeditor
4. 留言，新建一张数据库表

create table t\_message(

id int primary key auto\_increment,

article\_id int,

content longtext,

publisher varchar(40),

publish\_time datetime

);

添加外键alter table t\_message add constraint foreign key (article\_id) references t\_article(id);

1. 查不到记录是将出现sql语句不正确的错误，可以先判断记录数是否为空，不为空在进行分页
2. 会员管理

会员注册 会员登录 修改密码 找回密码 后台会员 还可能考虑到的需求（阅读文章扣费）

1. 会员相关：

//会员信息

create table t\_consumer(

id int primary key auto\_increment,

username varchar(20),

password varchar(20),

nickname varchar(20),

reg\_time datetime

);

//会员账户

create table t\_account(

id int primary key auto\_increment,

consumer\_id int,

score int

);

//会员充值

create table t\_charge(

id int primary key auto\_increment,

consumer\_id int,

money int,

charge\_time Datetime

);

create table t\_consume(

id int primary key auto\_increment,

consumer\_id int,

article\_id int,

consume\_date datetime

);

问题：Articleservlet里边调用ArticleDao对象为什么使用this

Right.jsp不能加入Java代码，不能当作Article\_list.jsp用

加外键不能往表里填数据

全选失灵

Hibernate 里的HQL是面向对象的查询方式，sql语句中不能出现表名，表字段，而应该是实体的属性

<many-to-one>默认是懒加载的

删除留言时，点击一次删除却提交两次删除请求，加javascript事件后就没问题了，为什么？？

Struts2版本的没有验证码后台： 前进、后退、刷新、显示登录用户

LoginOutServlet login.jsp

初始化参数在web.xml里

CheckcodeServlet top.jsp

UserDao.login()

left.jsp

login.jsp LoginServlet main.jsp center.jsp

right.jsp

down.jsp

添加

articleDao.query() ArticleDao.publish()（多次调用）

查询 AS.list() article\_list.jsp 多发布 AS.pubBatch() AS.list()

ArticleDao.delete()（多次调用）

多删 AS.delBatch() AS.list()

单发 AS.pub() AS.list()

单删 AS.del() AS.list()

文章 ChannelDao.query()

ArticleDao.query()

编辑 AS.modifyInput() update\_input.jsp

ArticleDao.update()

AS.modify()（关闭）

检索 AS.list()

ArticleDao.save()

添加 AS.addinput() add\_article.jsp AS.add() add\_article\_success.jsp

left.jsp ArticleDao.query() MessageDao.query()

留言 MS.articlelist() article\_title\_list.jsp MS.backlist() message\_list.jsp

MessageDao.delete()

ChannelDao.query() MS.del() 删除

CS.get() update\_input.jsp CS.modify() CS.list()

查询 CS.list() channel\_list()

CS.del() CS.list()

频道 添加 add\_input.jsp CS.add() add\_success.jsp

前台 主要是页面的包含

Logo: header.jsp

搜索:search.jsp

技术专题:technology.jsp

版权:footer.jsp

ChannelDao.query() ArticleDao.query()

（包含的页面与主页面大致一样） （也是包含页面）

导航条:IS.nav() nav.jsp topic.jsp IS.topic() topic\_list.jsp article.jsp

MessageDao.save() MeaageDao.query() ArticleDao.get()

article.jsp MS.add() message\_list.jsp MS.list() article\_detail.jsp IS.article()

ArticleDao.query()

index.jsp

头条: IS.headline() headline.jsp article.jsp

ArticleDao.query()

频道专题: IS.channelIndex() channelIndex.jsp

ArticleDao.query()

推荐阅读:IS.recommend() recommend.jsp

ArticleDao.query()

最新发表:IS.publish() publish.jsp

1. **设置验证码**

在web.xml 里设置初始化参数

<init-param>

<param-name>codes</param-name>

<param-value>ABCDEFGHIGKLMNOPQRSTUVWXYZ</param-value>

</init-param>

重写带参数的init()方法

在servlet里取得参数

int width = Integer.parseInt(config.getInitparameter(width));

response.setContentType("image/jpeg");

//创建图片

BufferedImage image = new BufferedImage(width,height,BufferedImage.TYPE\_INT\_RGB);

Graphics2D g = image.createGraphics();

//创建白色背景

g.setColor(Color.WHITE);

g.fillRect(0, 0, width, height);

//画边框

g.setColor(Color.BLACK);

g.drawRect(0, 0, width-1, height-1);

Random random = new Random();

//设置字符间距

int x = (width-1)/number;

int y = height-4;

//随机生成字符

for(int i=0;i<number;i++){

String code = String.valueOf(codes.charAt(random.nextInt(codes.length())));

int red = random.nextInt(255);

int green = random.nextInt(255);

int blue = random.nextInt(255);

g.setColor(new Color(red,green,blue));

Font font = new Font("Arial",Font.PLAIN,random(height,height/2));

g.setFont(font);

g.drawString(code, x\*i+1, y); //参数是字的位置

}

//随机产生点

for(int i=0;i<40;i++){

int red = random.nextInt(255);

int green = random.nextInt(255);

int blue = random.nextInt(255);

g.setColor(new Color(red,green,blue));

g.drawOval(random.nextInt(width), random.nextInt(height), 1, 1);

} //前面参数是位置，后面参数是大小

OutputStream out = response.getOutputStream();

//将图片转换成JPEG格式

JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);

encoder.encode(image);

out.flush();

out.close();

}

//取得自定义范围内的随机数

private int random(int max,int min){

int m = new Random().nextInt(99999)%(max-min);

return m+min;

}

1. **登录**
2. 第一步是检测验证码

CheckcodeServlet里创建一个StringBuffer用来装验证码

StringBuffer str = new StringBuffer();

Str.append(code);

//把验证码放在session里

Request.getsession().SetAttribute(“code”,str.toString());

在loginServlet里取得登录页面传过来的参数，先比较验证码是否与session里的一致，若不一致，forword回到login.jsp，并返回相应的提示信息。把之前输入过的用户名和密码也返回去。

在login.jsp取得Servlet里的参数：

第一种：在Servlet里setAttribute，在jsp里用EL表达式接受，${name};

第二种：直接取参数，${param.name}，这里是直接取得form表单传过来的参数

1. 第二步是检验用户名和密码

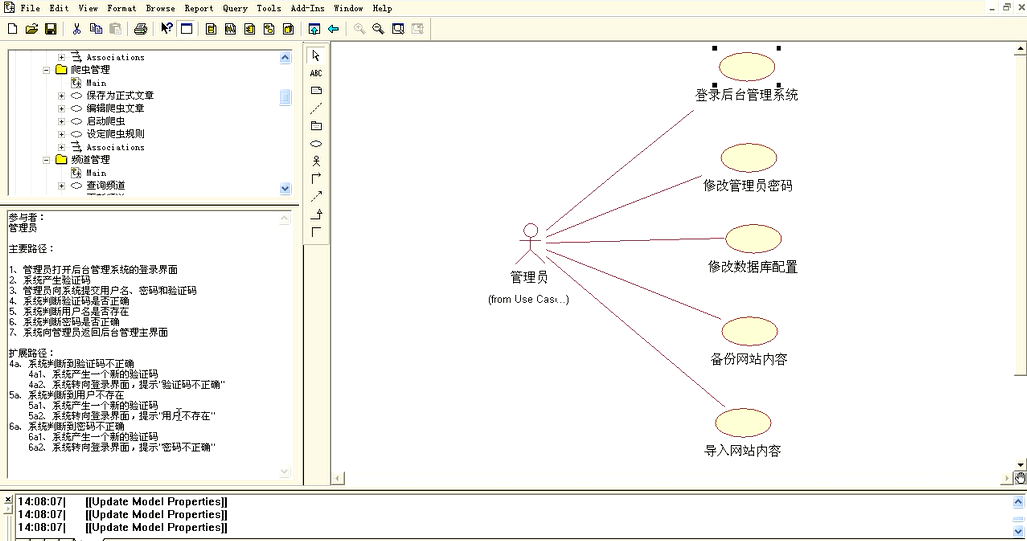
主要是连数据库，这里是调用BD.java，用表单传过来的name 与数据库的信息比对，如果存在，验证密码，若密码不正确则给出提示，如果不存在则提示用户名不存在。

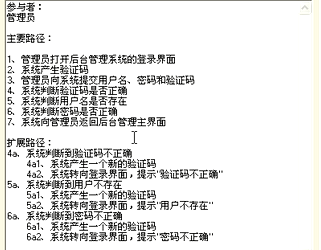
1. 如果前两步都验证成功了，则跳到后台页面

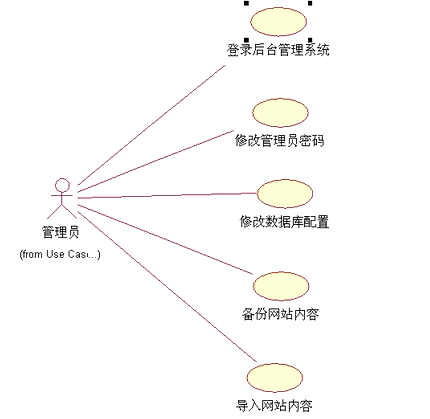
刷地址栏验证码变，而点图片不变，跟src有关系，如果判断两次的src没变的话，不重新发起请求，需在后面加

Class.forName(“com.mysql.jdbc.Driver”)利用反射把字节码加载到方法区中，可以使用其中的静态代码块，静态代码块在项目启动时自动执行，而静态方法在项目启动时只初始化，在被调用的时候才执行

1. **用例文档**







**时序图**描述[对象](http://baike.baidu.com/view/2387.htm)是如何交互的，并且将重点放在[消息](http://baike.baidu.com/view/89742.htm)序列上。也就是说，描述[消息](http://baike.baidu.com/view/89742.htm)是如何在[对象](http://baike.baidu.com/view/2387.htm)间发送和接收的。它通过描述[对象](http://baike.baidu.com/view/2387.htm)之间发送[消息](http://baike.baidu.com/view/89742.htm)的时间顺序显示多个对象之间的动态协作。它可以表示[用例](http://baike.baidu.com/view/706238.htm)的行为顺序，当执行一个用例行为时，时序图中的每条[消息](http://baike.baidu.com/view/89742.htm)对应了一个类操作或[状态机](http://baike.baidu.com/view/1906565.htm)中引起转换的触发事件。

**状态图**（流程图）用来表示条件判断，转移等交互，用来描述程序的工作流

1. **原始的登录验证**

在每个页面加是否登录的验证



1. **常用的登录验证：**

一个Filter可以有多个filter-mapping

在配置filter时，url-pattern要么写 \*.jsp 要么写 /backend/\* ，不能写/backend/\*.jsp

url-pattern三种定义方式：

精确匹配：/backend/login.jsp

目录匹配：/backend/\*

扩展名匹配：\*.jsp

1. **查询文章**

实体类里边不要出现数据库字段类型，用java.util.\*下的

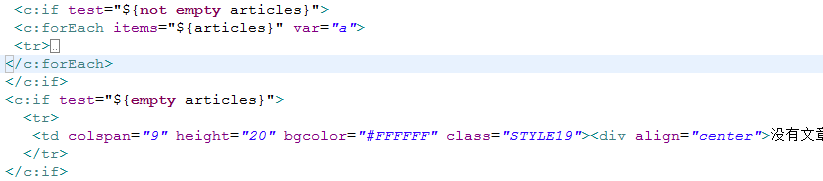
写一个单独的JSP用来加taglib

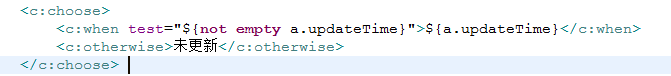


在其他的jsp中加载此jsp



显示文章列表

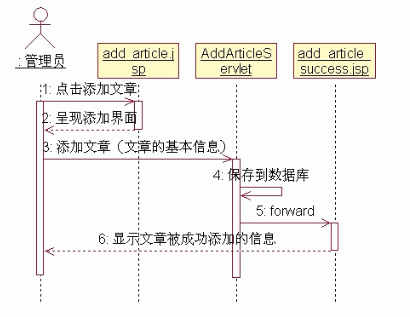




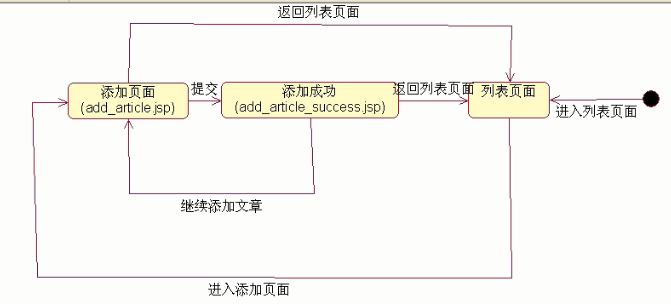
解决登录页面嵌套在其他页面中的问题：window指当前窗口，如果当前窗口的父窗口等于它本身，则正常，否则就在父窗口中打开当前窗口，直到没有父窗口为止。



添加文章时序图：



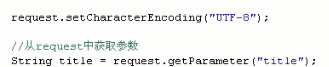
添加文章状态图：





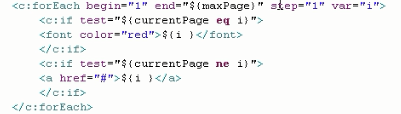
For的作用是与输入域绑定，点击“文章内容”也可以选定输入域

Post方式提交时，在jsp页面取参数之前设置编码格式：



1. **分页**

显示页号：forEach中的items表示要循环的集合

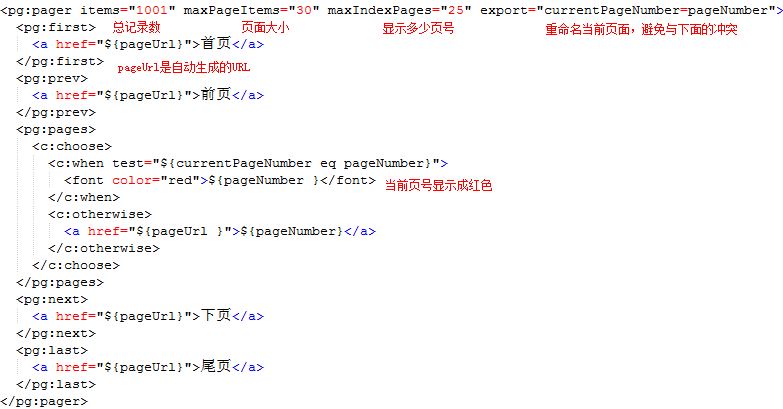


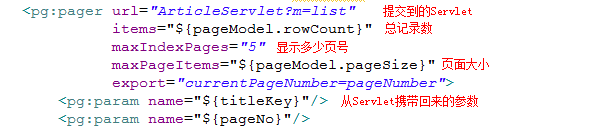
用pager-taglib进行分页

拷贝pager-taglib包

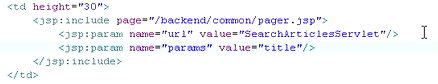
引入标签：







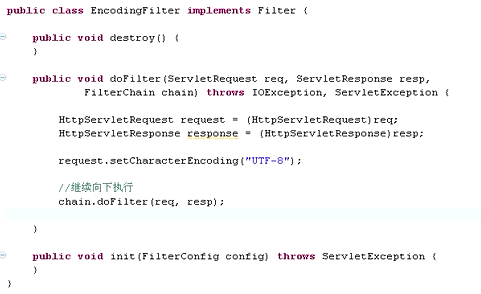
把分页封装到一个jsp页面，动态包含该页面，往被包含页面传递参数用jsp:param



Tiglib默认使用操作系统的编码

改taglib源代码，更改编码方式

用过滤器控制post请求编码



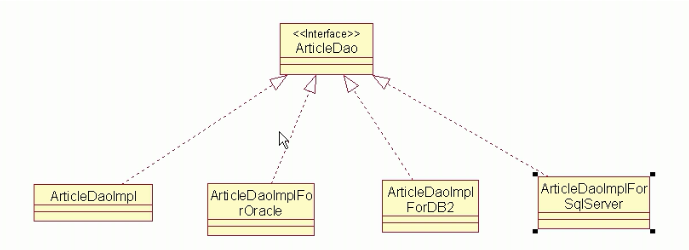
把编码格式写到配置文件里，更灵活

Myeclipse 断点调试

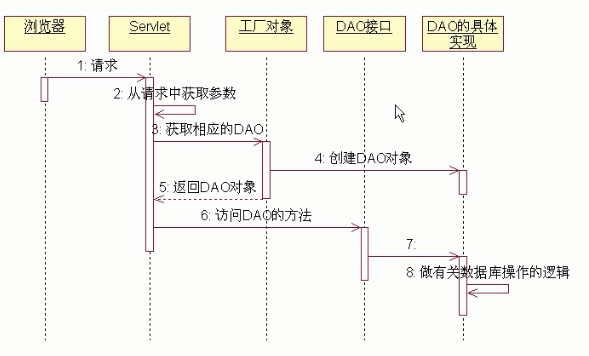
Vo value-object 值对象

1. **DAO层**

定义接口ArticleDao，不同的数据库操作的类实现该接口



利用工厂返回Dao对象



简单工厂模式，违反OCP原则（对修改关闭，对扩展开放），把创建Dao的职责转移给工厂，减少客户端对具体实现类的依赖，BeanFactory返回一个ArticleDaoImpl对象



利用工厂方法，满足OCP原则，通过扩展增强BeanFactory的能力，但是客户端依赖于具体实现类，BeanFactory是接口，下面两个是实现了该接口，返回特定的Dao对象



抽象工厂方法，一个工厂产生一类产品，即接口里定义多个抽象方法，下面的具体工厂返回一种类型的多个Dao对象，比如，一个工厂产生针对mysql的articleDao和channelDao，另

一个工厂产生针对Oracle的articleDao和channelDao

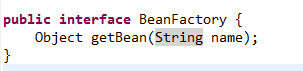
用配置文件屏蔽客户端对具体实现类的依赖

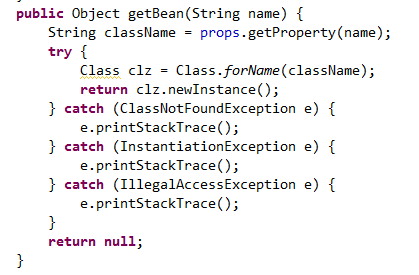






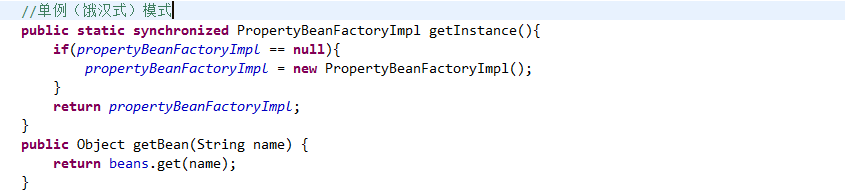
把Dao的类型隐藏起来





把Dao对象设置成单例的，产生PropertyBeanFactory时预先初始化dao对象，把PropertyBeanFactory也设置成单例的

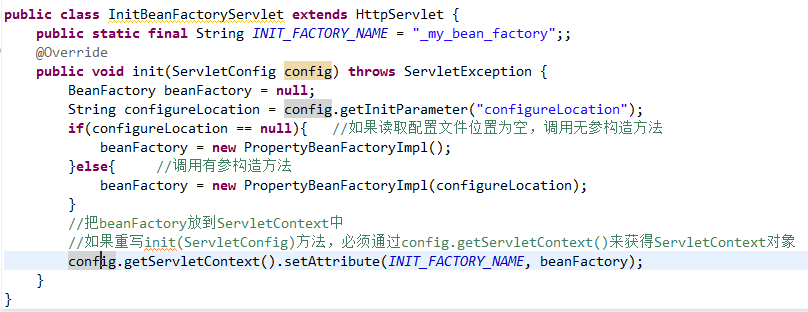


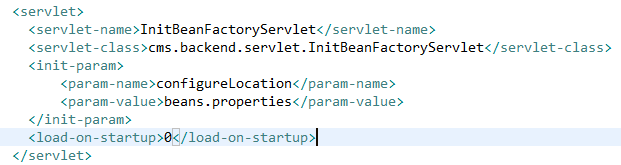




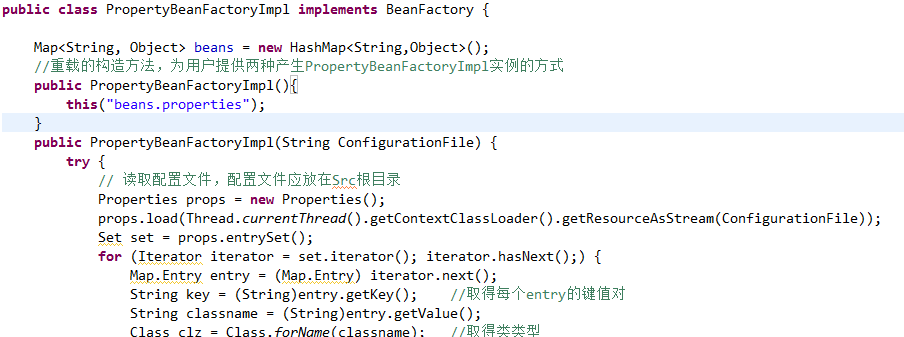
但上述方式使客户端仍依赖于一个具体的BeanFactory的实现，用一个servlet初始化BeanFactory，然后放到ServletContext中，beans.properties的路径写到web.xml中

定义InitBeanFactoryServlet初始化BeanFactory



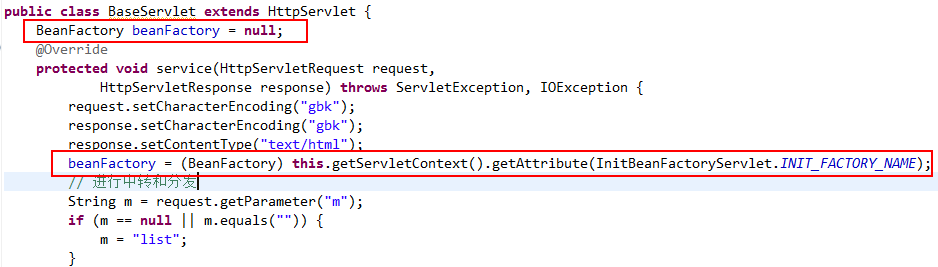


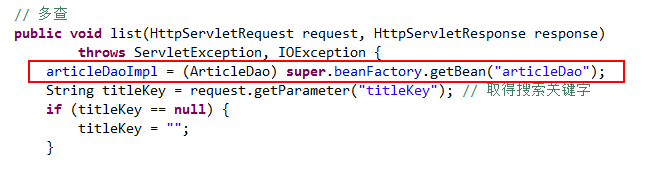
没有Servlet-mapping



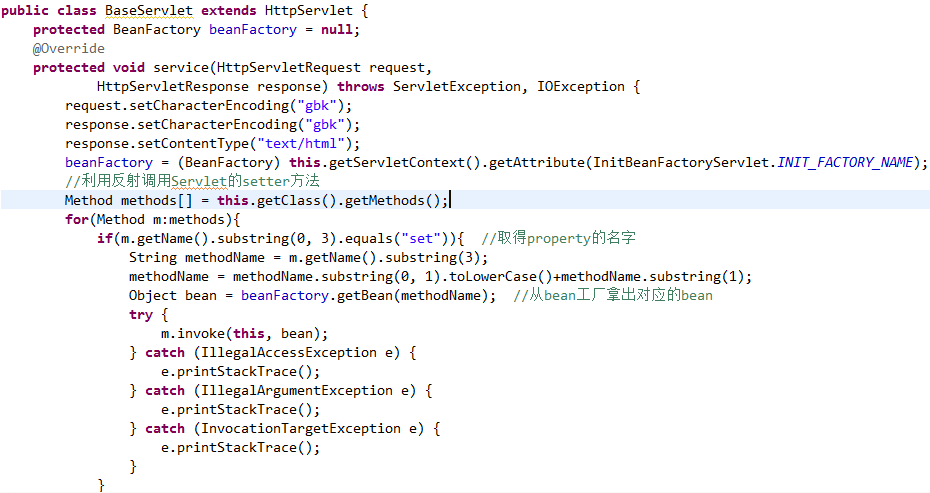
1. **BaseServlet和其他Servlet的关系**

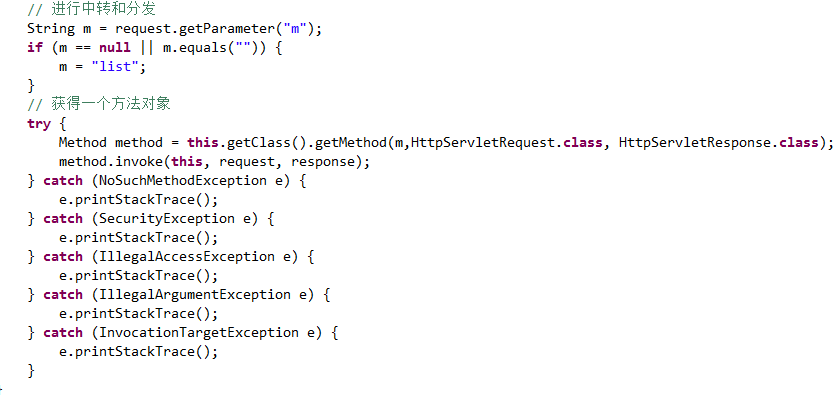
当访问ArticleServlet?m=list时，才会执行ArticleServlet继承来的BaseServlet的service()方法，this是实际调用的子类对象

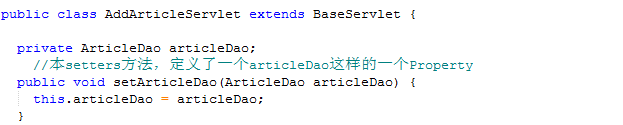




利用依赖注入DI，去掉客户端对工厂的依赖，客户端只提供一个setter方法，依赖前端调用该setter方法，注入参数



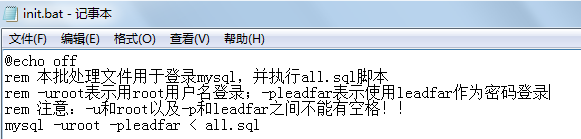




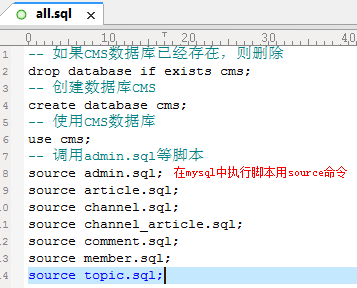
1. **数据库脚本，批处理文件**

要在cmd执行init.bat，需在path中加入mysql的bin，mysql客户端编码要和cmd编码一致

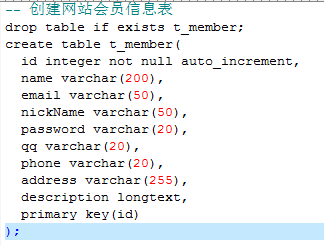
批处理文件 \*.bat



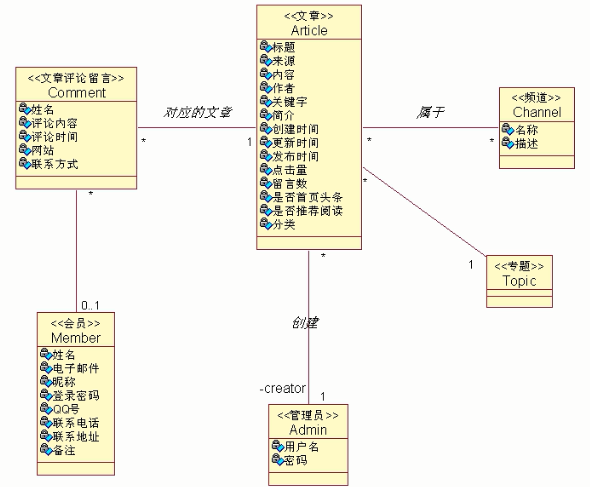
All.sql



Member.sql

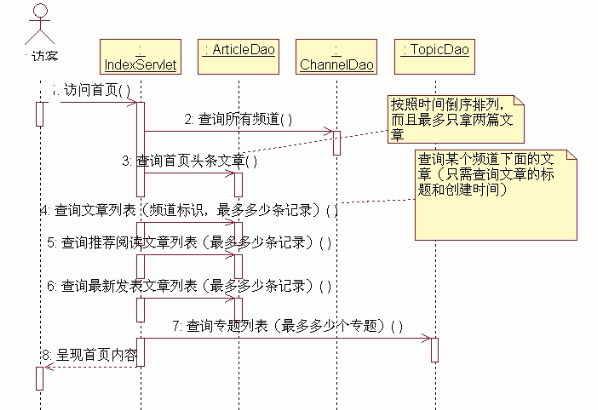


1. **分析整个交互过程**

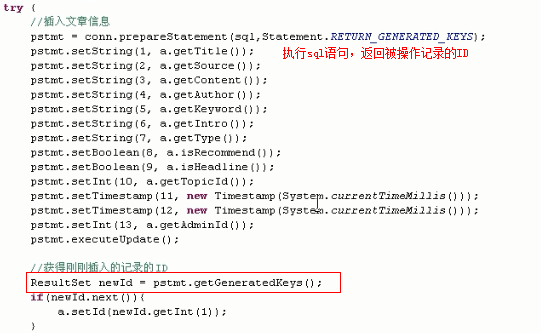


支持多选的select标签





1. **完善Dao层**



1. **IBatis**
2. 配置文件（SqlMapConfig.xml）：配置JDBC连接的有关信息，比如URL、用户名、密码，拷贝jar包：mybatis-3.0.1.jar



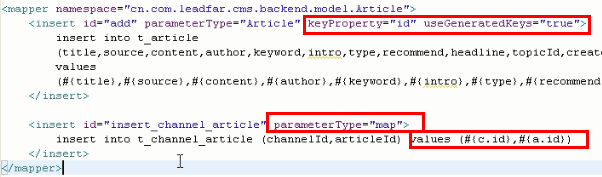
1. 映射文件：编写SQL语句，同时给每个SQL语句定义唯一的标识（ID），在程序中通过ID调用SQL语句



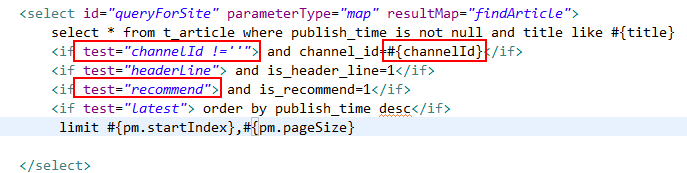
1. 利用IBatis实现添加



1. 利用IBatis实现数据库操作，并返回操作对象的Id，id是Article的一个字段

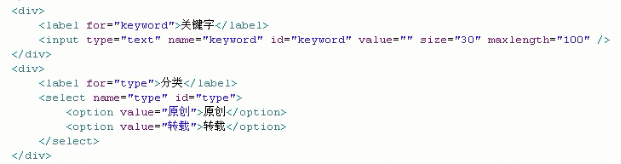


利用IBatis实现动态sql语句,取map里的参数不用加# 号,判断字符串是否为空串，用channelId != ‘’,Boolean类型直接写值

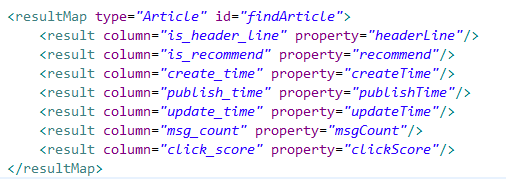


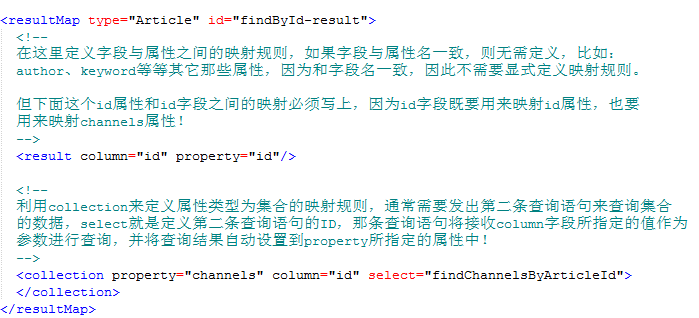
实体类中boolean类型的值IBatis会自动转化成1或0存到数据库中

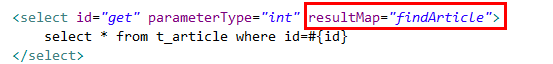
文章实体新增内容：



Column和property命名不一致时，要指明对应关系







1. **Apache-commons-beanutils对请求参数封装**
2. 拷贝jar包

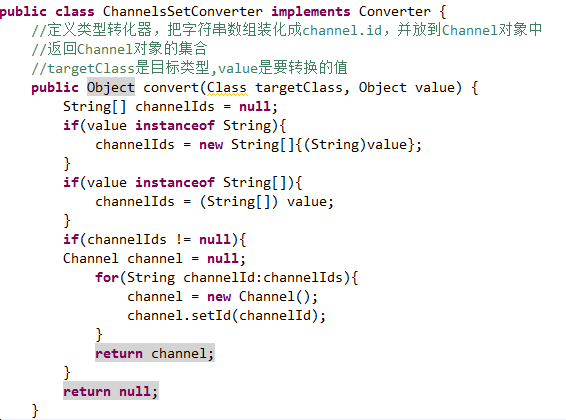
commons-beanutils.jar

commons-logging-1.1.1.jar

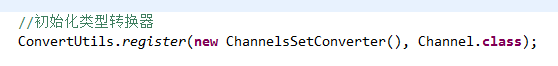
1. 把request中的参数封装到实体中



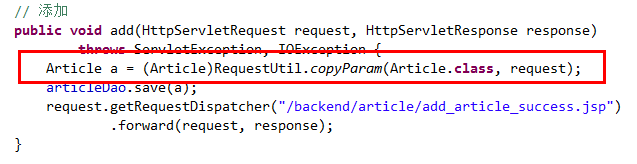
定义类型转换器



在InitBeanFactoryServlet初始化类型转换器，当遇到Channel类型时调用该转换器，只要在调用之前注册就行



1. 封装参数



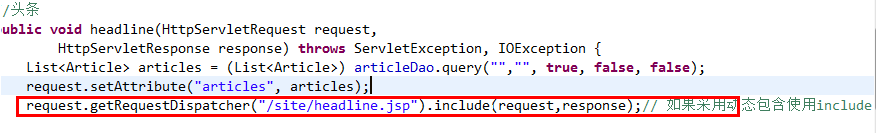
1. **用BaseDao对dao层通用方法进行封装，dao继承BaseDao**
2. **ThreadLocal**
3. 当多个线程共享一个静态变量时，用ThreadLocal使各个线程之间相互独立，访问的都是自己线程的变量值
4. 定义ThreadLocal

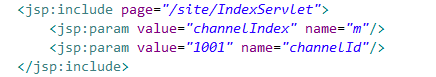


1. BaseServlet中设置ThreadLocal中的值

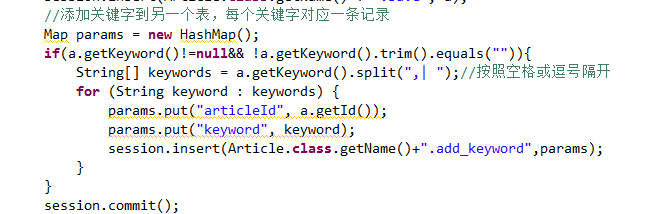


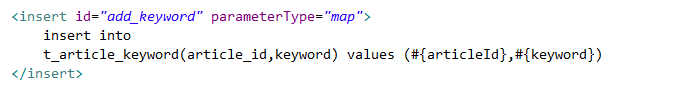
1. **用jsp动态包含公共页面**
2. 用jsp页面include，后台用include把各个jsp页面的请求包含到一个请求中



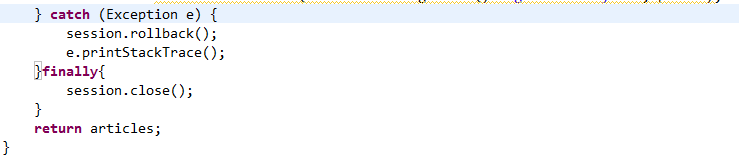


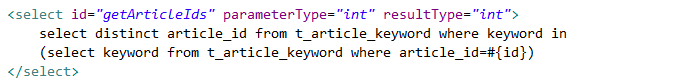
1. **相关文章的查询**

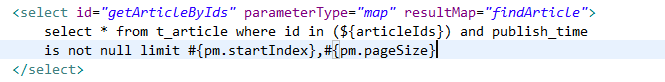












**CMS\_43，自定义JSTl函数**

1. **会员相关**

会员登录，登录完成后仍在登录前访问的页面



1. **用+**
2. **cript进行表单验证**

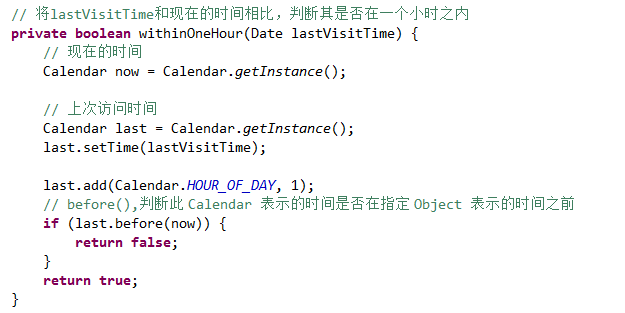
防止表单自动提交，用return



1. **更新文章点击量，避免重复点击增加点击量**

统一IP在一小时内对同一文章不重复计算点击量





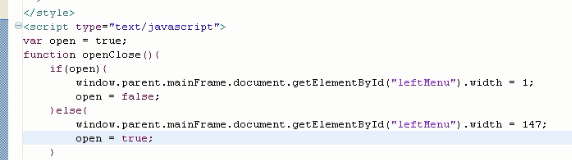
1. **CKediter**
2. 引入js文件



1. Onload方法，在整个页面加载完后才调用编辑器

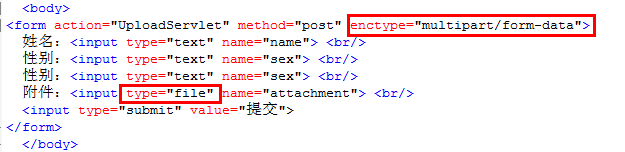


1. **缩放左边导航菜单**



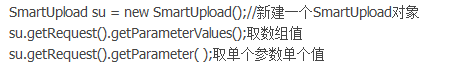
添加频道号相同，后台报错

1. **文件上传**
2. 表单



表单中enctype="multipart/form-data"的意思，是设置表单的MIME编码。默认情况，这个编码格式是application /x-www-form-urlencoded，不能用于文件上传；只有使用了multipart/form-data，才能完整的传递文件数据，

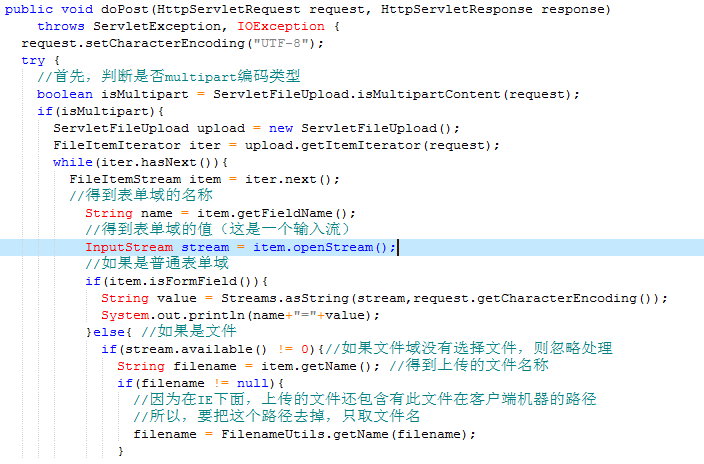
enctype="multipart/form-data"是上传二进制数据; form里面的input的值以2进制的方式传过去。form里面的input的值以2进制的方式传过去，所以request就得不到值了。 也就是说加了这段代码,用request就会传递不成功,取表单值加入数据库时，用到下面的：

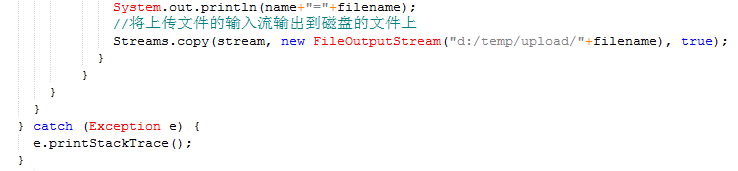


1. 拷贝jar包

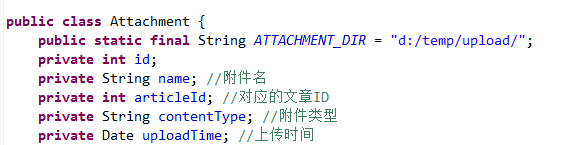


1. UploadServlet，从表单获得普通表单域和文件

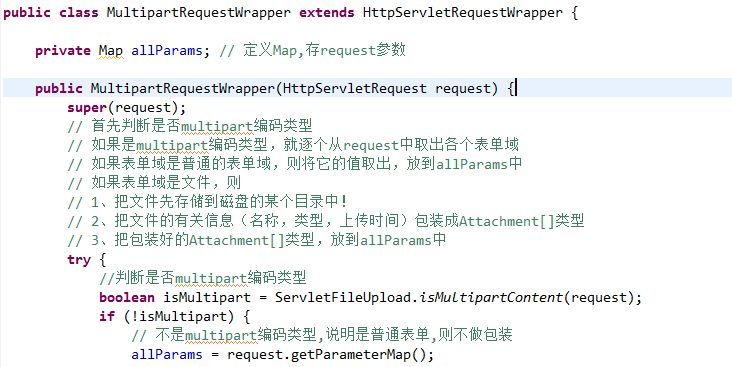




1. 创建附件类Attachment.java

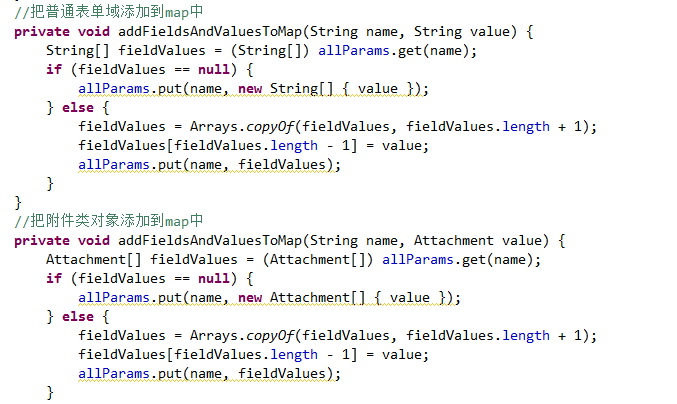


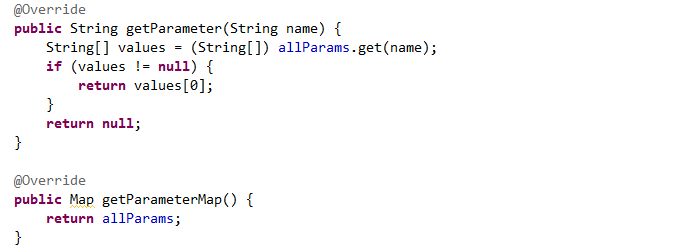
1. 创建请求包装类，包装request请求











1. BaseServlet





表单提交的参数与类属性不对应，就不会调用RequestUtil的copyParam()方法