摘 要

在编程语言排行榜网站Tiobe上我们可以看到Java语言一直是高居榜首，证明了这个语言的使用群体是多么的庞大，能够有如此庞大的用户群体，自然Java语言的生态环境肯定是异常丰富的，在我们本次项目中使用到的Spring,SpringMVC，Mybatis都是属于Java生态圈中名号响当当的成员，对于现在的web开发来说，越来越追求性能，追求程序的开发效率，自然轻量级的框架是深受广大的企业或者是开发者的喜爱，现在的这些Web开发的轻量级的框架相较于以前的EJB这一套的组建模型的复杂度要降低很多，并且从开发人员的角度来说的话，使用的复杂程度有了很大的改进，更加利于上手了，对于企业来说可以节约各方面的资源，从而有效节约成本，在Web开发和Java相对比的还有一个微软的.Net,虽然是开源的时间很晚，但是在这几年开源之后也是势头很猛，但是论生态环境来比对的话，还是Java略胜一筹的，Java社区更加的活跃，开源项目贡献的开发者也很多，并且.Net早期是只能在Windows的服务器上面部署的，也是最近几年才支持的跨平台，但是其更新迭代的速度很慢，跟不上市场的脚步，加上正版的VS和Windows Server价格比较昂贵，相较于Java众多免费的资源来说，提高了开发的成本，所以在Web端开发，Java还是占有了更多的优势。

**关键词：**Java Spring Mybatis Windows Web框架

ABSTRACT

On the programming language list website Tiobe, we can see that the Java language has always topped the list, demonstrating how large the community of users of this language is and can have such a large user base. The natural Java language ecological environment is definitely exceptionally rich. The Spring, SpringMVC, and Mybatis used in this project are members of the famous name in the Java ecosystem. For today's web development, more and more pursuit of performance, the pursuit of the development efficiency of the program, natural light The framework of magnitude is deeply loved by the majority of companies or developers. Nowadays, these lightweight frameworks for web development are much less complex than previous EJBs. From the perspective of personnel, the complexity of use has been greatly improved, and it is more conducive to getting started. For enterprises, resources can be saved in all aspects, thus effectively saving costs. There is also a comparison between web development and Java. Microsoft's .Net, although it is open source late, but in the past few years after the open source is also a strong momentum, but on the ecological environment In comparison, Java is a bit better, the Java community is more active, and there are many developers who contributed to the open source project. Early on, .net was deployed only on the Windows server and was also supported in recent years. Platform, but the speed of its update iteration is slow, not keeping up with the pace of the market, plus the price of genuine VS and Windows Server is relatively expensive, compared to many free resources of java, to increase the cost of development, so in the web Side development, java still has more advantages.

**Key words ：**Java Spring Mybatis Windows Web FrameWor

目录

[第1章 框架的概念和原理 1](#_Toc512787076)

[1.1 SpringMVC框架 1](#_Toc512787077)

[1.1.2 SpringMVC的工作流程 2](#_Toc512787078)

[1.1.3 图示SpringMVC异常处理机制 2](#_Toc512787079)

[1.1.4 SpringMVC解决POST时中文乱码问题 3](#_Toc512787080)

[1.2 Spring框架 4](#_Toc512787081)

[1.2.1 Spring中的核心部分： 4](#_Toc512787082)

[1.2.2 Spring的IOC操作说明： 4](#_Toc512787083)

[1.2.3 ioc底层原理 4](#_Toc512787084)

[1.2.4 Spring的Bean管理（xml方式） 5](#_Toc512787085)

[1.2.5 Bean属性注入介绍 6](#_Toc512787086)

[1.2.6 IOC和DI的区别 8](#_Toc512787087)

[1.2.7 AOP的概念及原理 8](#_Toc512787088)

[1.3 Mybatis框架 10](#_Toc512787089)

[1.3.2 Mybatis和Hibernate的对比 10](#_Toc512787090)

[1.3.3 图解Mybatis架构 11](#_Toc512787091)

[1.3.4 图示Mybatis操作步骤 11](#_Toc512787092)

[1.3.5 parameterType和resultType 12](#_Toc512787093)

[1.3.6 selectOne和selectList 12](#_Toc512787094)

[1.3.7 Mybatis的Mapper动态代理 12](#_Toc512787095)

[1.3.8 Mybatis的typeAliases（类型别名） 13](#_Toc512787096)

[1.3.9 Mybatis缓存介绍 14](#_Toc512787097)

[第2章 项目中涉及的工具 17](#_Toc512787098)

[2.1 Maven工具 17](#_Toc512787099)

[2.1.1 Maven的好处以及实现的原理 17](#_Toc512787100)

[2.1.2 图解Maven查找Jar包的过程 17](#_Toc512787101)

[2.1.3 使用IDE自动创建Maven项目目录结构 18](#_Toc512787102)

[2.1.4 Maven常用命令 18](#_Toc512787103)

[2.1.5 Maven项目生命周期 19](#_Toc512787104)

[2.2 Log4j 19](#_Toc512787105)

[2.2.1 Maven项目中使用Log4j 19](#_Toc512787106)

[2.3 Druid数据库连接池 20](#_Toc512787107)

[2.3.1 Maven中使用Druid 20](#_Toc512787108)

[第3章 SSM框架项目整合的搭建 21](#_Toc512787109)

[3.1 工具的介绍 21](#_Toc512787110)

[3.1.2 工具的版本 21](#_Toc512787111)

[3.2 项目具体实施 21](#_Toc512787112)

[3.2.1 创建Maven项目 21](#_Toc512787113)

[3.2.2 添加依赖 21](#_Toc512787114)

[3.2.3 web.xml中的部分配置 24](#_Toc512787115)

[3.2.4 配置Druid连接池以及Mapper扫描器 25](#_Toc512787116)

[3.2.5 配置各类ServiceBean(仅截取部分代码) 25](#_Toc512787117)

[3.2.6 配置事务及相关 26](#_Toc512787118)

[3.2.7 SpringMVC的配置文件（仅截取部分代码） 26](#_Toc512787119)

[3.2.8 Mapper的开发（部分代码） 27](#_Toc512787120)

[3.2.9 表现层的开发（部分代码） 28](#_Toc512787121)

[3.2.10 业务逻辑层的开发（部分代码） 28](#_Toc512787122)

[第4章 主要功能实现 30](#_Toc512787123)

[4.1 登陆功能的实现及结果展示 30](#_Toc512787124)

[4.1.1 登陆结果展示 31](#_Toc512787125)

[4.2 照片发布的实现及结果展示 32](#_Toc512787126)

[4.2.1 照片发布结果展示 35](#_Toc512787127)

[4.3 用户点赞的实现及结果展示 36](#_Toc512787128)

[4.3.1 点赞结果展示 37](#_Toc512787129)

[4.4 上一篇或下一篇文章功能实现及展示 38](#_Toc512787130)

[4.4.1 上一篇下一篇文章功能展示 39](#_Toc512787131)

[4.5 搜索文章的功能及结果展示 40](#_Toc512787132)

[4.5.1 搜索功能结果展示 41](#_Toc512787133)

[4.6 图片文章删除功能及结果展示 42](#_Toc512787134)

[4.6.1 图片删除功能展示 43](#_Toc512787135)

[4.7 图片分类功能实现及结果展示 43](#_Toc512787136)

[4.7.1 分类结果展示 46](#_Toc512787137)

[第5章 总结 48](#_Toc512787138)

[5.1 总结 48](#_Toc512787139)

[参考文献 49](#_Toc512787140)

[致 谢 50](#_Toc512787141)

# 第1章 框架的概念和原理

Spring MVC属于Spring中的一部分，Spring 众多开发Web应用程序的模块。我们在编写web程序的表现层时可以使用SpringMVC，不在通过中间整合层，并且可以和Spring无缝整合， SpringMVC搭配注解开发的时候很方便，可以使我们少写很多代码，代码看起来也是更加的清楚明了。

## 1.1 SpringMVC框架

1.SpringMVC和Struts2接受请求的方式不同，SpringMVC前端控制器来接受请求，而struts2过滤器接受请求。

2. Struts2通过类开发， struts2局限性比较大，只能设计成为多例模式，因为传参是通过类的属性，这个类型的属性也会被很多的方法所使用，没有办法搭配Spring的时候使用注解，并且最近几年在安全性上面Struts是饱受争议，SpringMVC是通过方法开发的，但是SpringMVC可以设置为多例和单例两种模式，请求的数据通过参数来传递，正是因为这样子的设计，让我们的注解有了更好的使用环境。



图1-1-1 SpringMVC架构

### 1.1.2 SpringMVC的工作流程

1.页面发送请求到前端控制器DispatcherServlet

2.前端控制器收到发出请求后调用处理器映射器HandlerMapping。

3. HandlerMapping根据请求的url找到对应的Controller，同时生成处理器对象及处理器拦截器返回给前端控制器。

4.前端控制器通过处理器适配器HandlerAdapter调用Controller，与此同时执行后端控制器。

5.Controller执行完成返回ModelAndView

6.HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet

7.DispatcherServlet将ModelAndView传给ViewReslover视图解析器

8.ViewReslover解析后返回具体View

9.前端控制器对View进行渲染。

10.DispatcherServlet响应发送给用户

### 1.1.3 图示SpringMVC异常处理机制

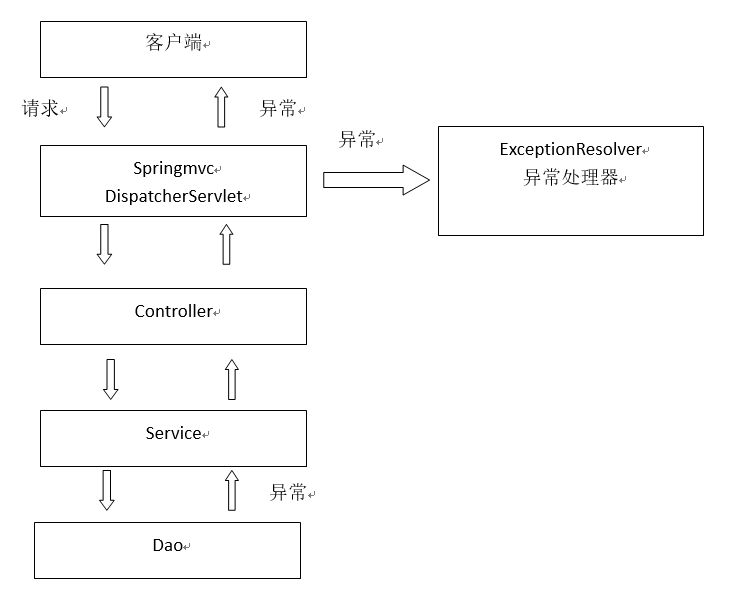


图1-1-2 SpringMVC异常处理流程

### 1.1.4 SpringMVC解决POST时中文乱码问题

在web.xml中加入：

<filter>

<filter-name>CharacterEncodingFilter</filter-name>

<fi2lter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>CharacterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

以上可以解决post请求乱码问题。

对于get请求中文参数出现乱码解决方法有两个：

修改tomcat配置文件添加编码与工程编码一致，如下：

<Connector URIEncoding="utf-8" connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>

另外一种方法对参数进行重新编码：

String userName new

String(request.getParamter("userName").getBytes("ISO8859-1"),"utf-8")

ISO8859-1是tomcat默认编码，需要将tomcat编码后的内容按utf-8编码

## 1.2 Spring框架

Spring是一个轻量级的开源框架，其中囊括了web三层开发中的解决技术，被很多的人称之为一站式框架，也是因为这个名字的中文翻译，被成为程序员的春天，Spring的使用，极大的降低了代码的耦合度，更好的维护代码，封装了大量的API在其中，对象的创建不在需要开发人员手动来new,直接统一交给Spring来管理。

### 1.2.1 Spring中的核心部分：

1.aop：面向切面编程，使代码的扩展性大大提高，不需要修改源代码就可以扩展。

2.ioc：控制反转，最大好处就是不需要程序员在创建对象的时候通过new方式实现，而是交给spring配置创建类对象

### 1.2.2 Spring的IOC操作说明：

1.把对象的创建交给spring进行管理

2.ioc操作两部分：

（1）ioc的配置文件方式

（2）ioc的注解方式

### 1.2.3 ioc底层原理

1.ioc底层原理使用技术

（1）xml配置文件

（2）dom4j解决xml

（3）工厂设计模式

（4）反射技术

2.图示ioc底层原理

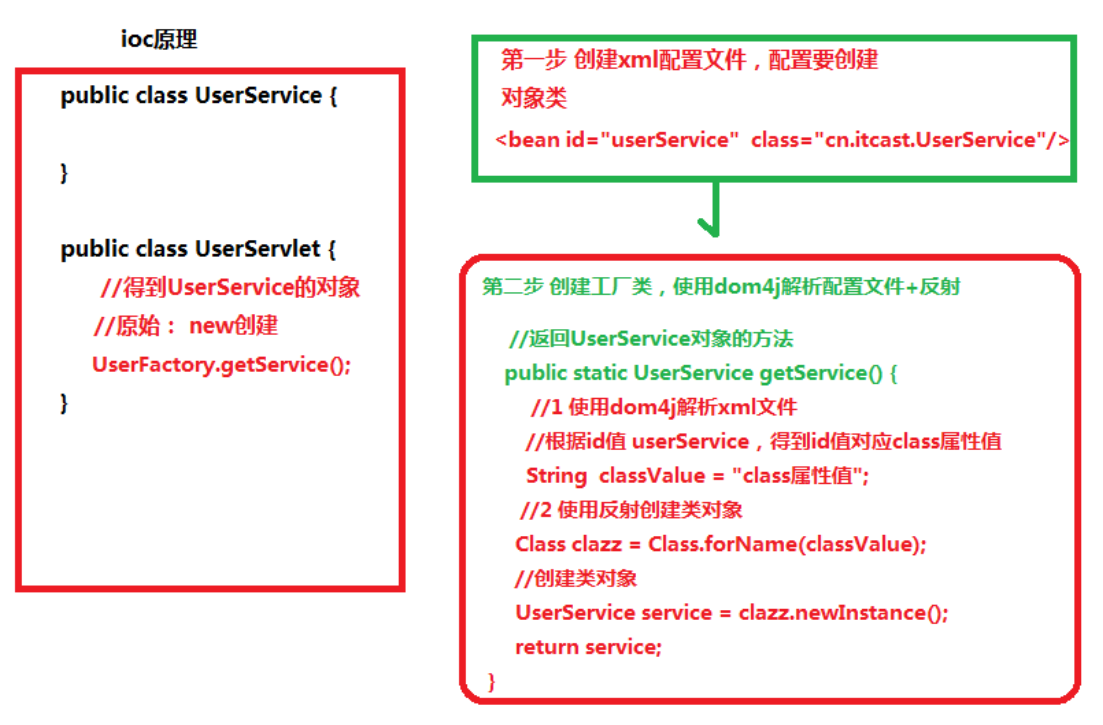


图1-2-1 ioc底层原理

### 1.2.4 Spring的Bean管理（xml方式）

#### 1.2.4.1 Bean的三种实例化方式

1.在spring里面通过配置文件创建对象

2.Bean实例化三种方式实现

第一种 使用类的无参数构造创建（重点）

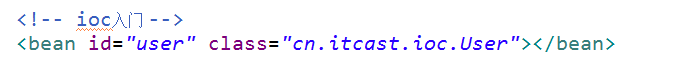
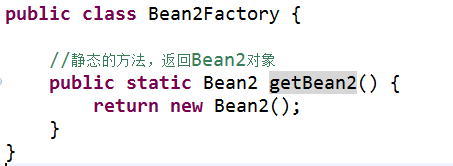
****

图1-2-2 无参构造创建bean方式

第二种 使用静态工厂创建

1.创建静态的方法，返回类对象

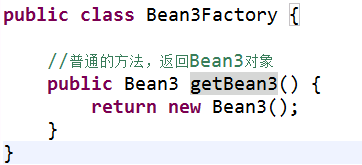
****

****

图1-2-3 静态工厂创建bean方式图

第三种 使用实例工厂创建

1.创建不是静态的方法，返回类对象

****

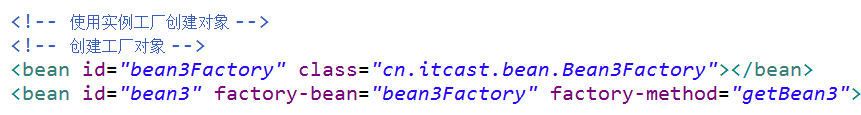
****

图1-2-4 实例工厂创建bean方式图

### 1.2.5 Bean属性注入介绍

1.创建对象时候，向类里面属性里面设置值

2.属性注入的方式介绍（三种方式）

（1）使用set方法注入

（2）使用有参数构造注入

（3）使用接口注入

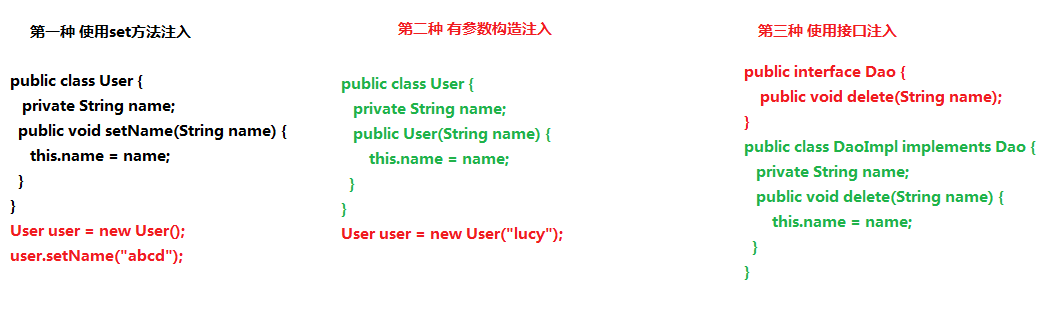
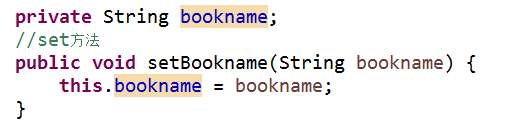
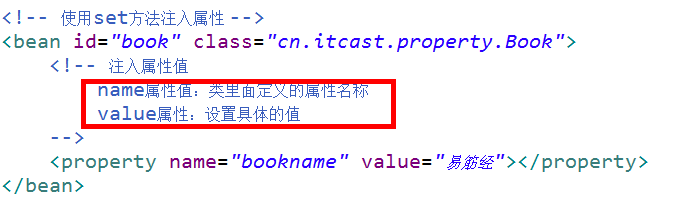
****

图1-2-5 属性注入三种方式图

3.在spring框架里面，支持前两种方式

（1）set方法注入（重点）

****

****

（2）有参数构造注入

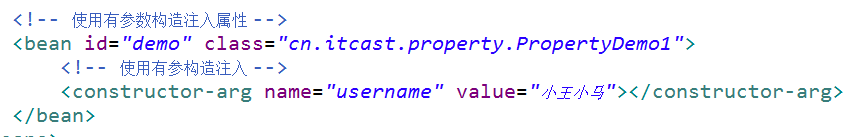
****

图1-2-6 set方法注入

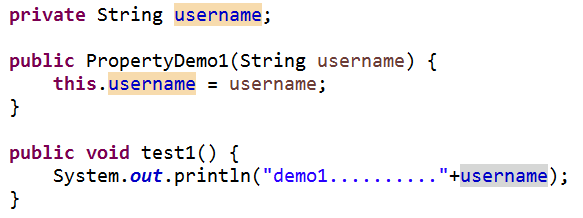
****

图1-2-7 有参数构造注入

### 1.2.6 IOC和DI的区别

（1）IOC: 控制反转，把对象创建交给spring进行配置

（2）DI: 依赖注入，向类里面的属性中设置值

（3）关系：依赖注入不能单独存在，需要在ioc基础之上完成操作

### 1.2.7 AOP的概念及原理

#### 1.2.7.1 AOP的概念

1.AOP：面向切面（方面）编程，扩展功能不修改源代码实现

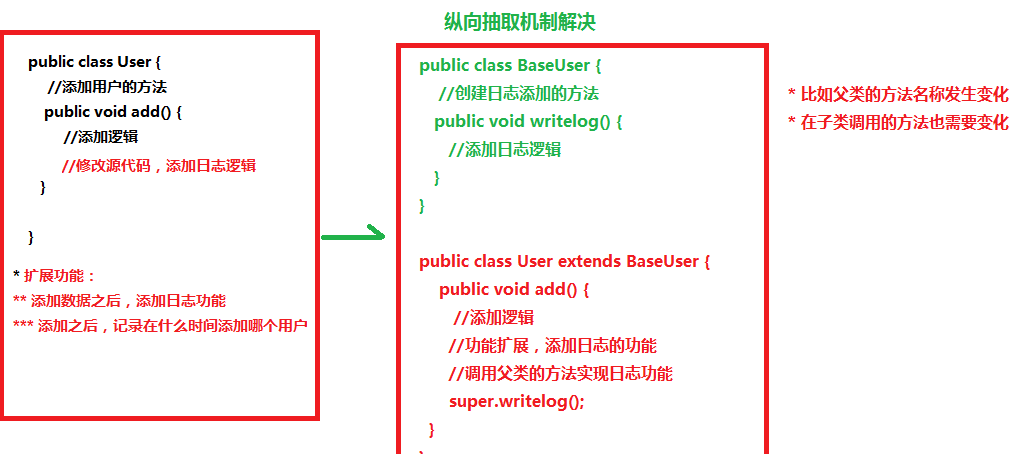
2.AOP采取横向抽取机制，取代了传统纵向继承体系重复性代码

3.AOP底层使用动态代理实现

（1）第一种情况，有接口情况，使用动态代理创建接口实现类代理对象

（2）第二种情况，没有接口情况，使用动态代理创建类的子类代理对象

#### 1.2.7.2 图示AOP的原理

****

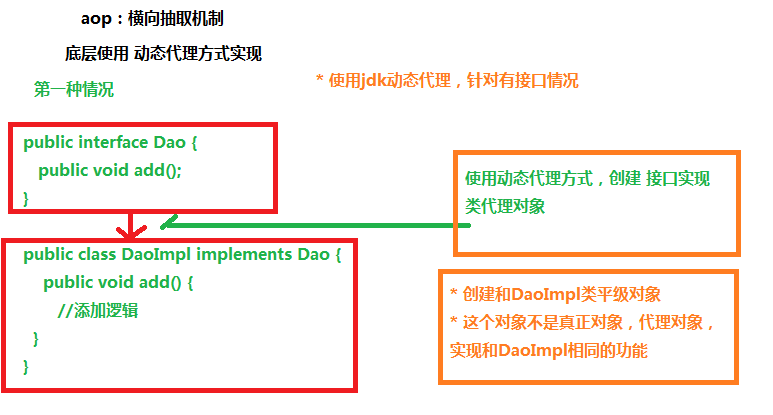
****

图1-2-8 AOP原理图

#### 1.2.7.3 AOP常用术语

Joinpoint(连接点): 类里面可以被增强的方法，这些方法称为连接点

Pointcut(切入点):所谓切入点是指我们要对哪些Joinpoint进行拦截的定义.

Advice(通知/增强):所谓通知是指拦截到Joinpoint之后所要做的事情就是通知.通知分为前置通知,后置通知,异常通知,最终通知,环绕通知(切面要完成的功能)

Aspect(切面): 是切入点和通知（引介）的结合

Introduction(引介):引介是一种特殊的通知在不修改类代码的前提下, Introduction可以在运行期为类动态地添加一些方法或Field.

#### 1.2.7.4 Spring的AOP操作

1.在spring里面进行aop操作，使用aspectj实现

（1）aspectj不是spring一部分，和spring一起使用进行aop操作

（2）Spring2.0以后新增了对AspectJ支持

2.使用aspectj实现aop有两种方式

（1）基于aspectj的xml配置

（2）基于aspectj的注解方式

## 1.3 Mybatis框架

Mybatis 也是属于一个开源项目,它是属于Web开发中持久层框架，底层对于JDBC进行了封装，开发人员直接编写Sql语句，不需要去操作JDBC复杂的代码。

### 1.3.2 Mybatis和Hibernate的对比

Mybatis和hibernate不同，它不属于一个纯粹的ORM框架，因为Mybatis需要程序员自己编写Sql语句， mybatis可以通过XML或注解方式灵活配置要执行的sql语句，并将POJO和sql语句映射生成最终执行的sql，最后将sql执行的结果再映射生成POJO。

Mybatis对于新手程序员来说是非常的友好的，程序员可以直接编写sql语句，灵活度非常的高，适合需求变化频繁的软件开发。但是数据库无关性做的不好，如果需要实现支持多种数据库的软件则需要自定义多套sql不同的映射文件，工作复杂程度也是很高的。

因为Hibernate是属于标准的ORM框架，所以它的对象关系映射能力强，与数据库无关性做的非常的好，但是不如Mybatis灵活，只适合需求固定的软件，在一些特定的场景下用hibernate开发可以少写很多的代码，提高开发效率。但是Hibernate体系非常的庞大，对比很多新手程序员来说比较困难，对于设计O/R映射，在性能和对象模型的平衡性要通过开发经验来判定。

### 1.3.3 图解Mybatis架构



图1-3-1 Mybatis架构图

### 1.3.4 图示Mybatis操作步骤

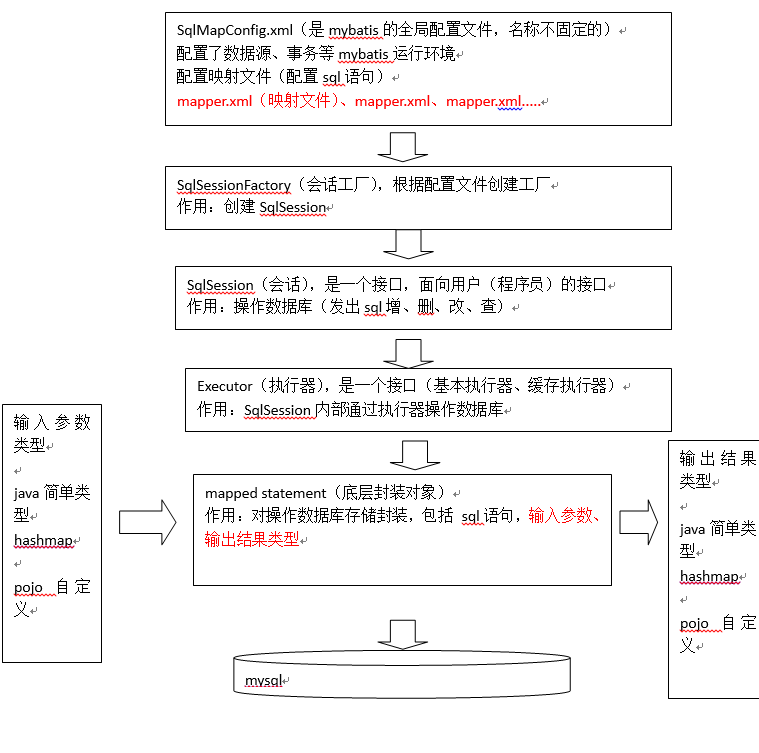


图1-3-2 Mybatis操作步骤

### 1.3.5 parameterType和resultType

parameterType：指定输入参数类型，mybatis从输入对象中获取参数值拼接在sql中。

resultType：指定输出结果类型，将sql查询结果的记录数据映射为resultType指定类型的对象。

### 1.3.6 selectOne和selectList

1.selectOne只能查询数据库中的一条记录，使用selectOne查询多条数据库记录会则抛出异常：

org.apache.ibatis.exceptions.TooManyResultsException: Expected one result (or null) to be returned by selectOne(), but found: 3 at org.apache.ibatis.session.defaults.DefaultSqlSession.selectOne(DefaultSqlSession.java:70)

2.selectList则可以查询数据库中的多条记录或一条记录。

### 1.3.7 Mybatis的Mapper动态代理

#### 1.3.7.1原始Dao开发的问题：

1.Dao方法中存在许多重复的代码：通过会话工厂创建SqlSession，通过调用SqlSession的数据库操作方法，进行数据库的操作。

2.在调用sqlSession的数据库操作方法需要指定statement的id，存在硬编码，不得于开发维护。

#### 1.3.7.2 Mapper实现原理

Mapper接口开发方法只需要程序员编写Mapper接口，由Mybatis框架根据接口定义创建接口的代理对象。

#### 1.3.7.3 Mapper接口开发的规范：

1.Mapper.xml中的namespace与mapper接口的类路径保持一致。

2.Mapper接口中方法名和Mapper.xml中statement的id一致

3.Mapper接口中方法的输入参数类型和mapper.xml中对应的parameterType的类型相同。

4.Mapper接口方法的输出参数类型和mapper.xml中对应resultType的类型相同。

### 1.3.8 Mybatis的typeAliases（类型别名）

#### 1.3.8.1 Mybatis支持的部分别名

|  |  |
| --- | --- |
| 别名 | 映射的类型 |
| \_byte | byte |
| \_long | long |
| \_short | short |
| \_int | int |
| \_integer | int |
| \_double | double |
| \_float | float |
| \_boolean | boolean |
| string | String |
| byte | Byte |
| long | Long |
| short | Short |
| int | Integer |

#### 1.3.8.2 自定义别名

在SqlMapConfig.xml中配置：

<typeAliases>

<!-- 单个别名定义 -->

<typeAlias alias="user" type="cn.wbz.mybatis.po.User"/>

<!-- 批量别名定义，扫描整个包下的类-->

<package name="cn.wbz.mybatis.po"/>

<package name="其它包"/>

</typeAliases>

### 1.3.9 Mybatis缓存介绍

#### 1.3.9.1 mybatis一级缓存和二级缓存的区别图解

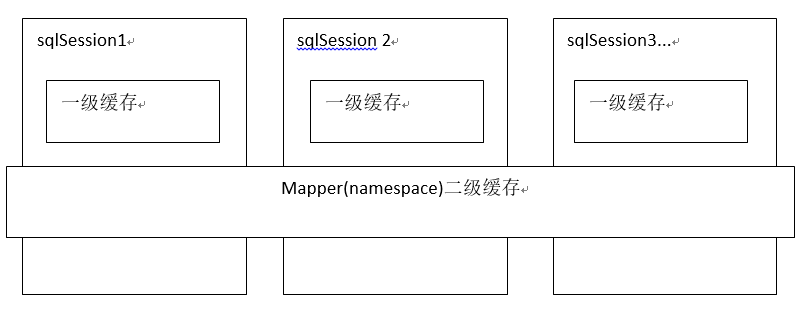


图1-3-3 mybatis缓存原理

Mybatis一级缓存是默认开启的，它的作用域是同一个SqlSession，如果是在一个sqlSession中两次执行相同的sql语句，第一次执行完毕会将数据库中查询的数据写到缓存中，第二次不会再进行数据库的操作，直接从缓存里面取得数据，提高了查询的速度，不再重新查询，对于资源的利用率大大提高，降低服务器的负担。当sqlSession结束后该sqlSession中的一级缓存也就清空了。

Mybatis二级缓存是多个SqlSession共享的，其作用域是mapper的同一个namespace，如果不同的sqlSession两次执行相同namespace下的sql语句且向sql中传递参数也相同即最终执行相同的sql语句，第一次执行的数据库操作的数据会存入到缓存中，第二次直接从缓存中读取数据，不再进兴数据库的操作，Mybatis需要开发人员手动开启二级缓存。

#### 1.3.9.2 一级缓存原理

一级缓存区域是根据SqlSession划分的。Mybatis缓存使用一个HashMap，key是hashCode+sqlId+Sql语句。Value是查询出来映射生成的java对象，sqlSession执行增删改查等操作提交后，会清空缓存区域。

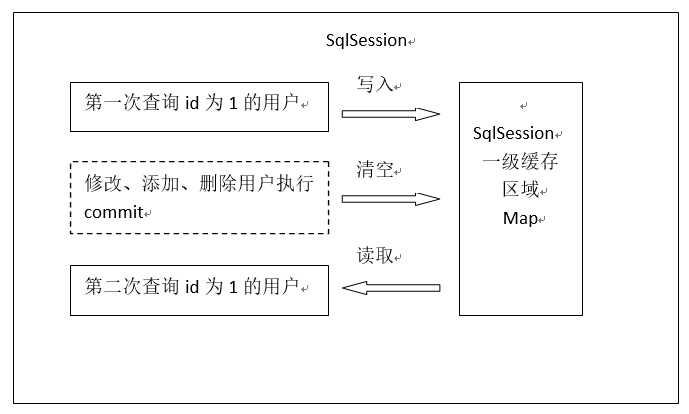


图1-3-4 根据id查询用户一级缓存

#### 1.3.9.3 二级缓存原理

二级缓存区域是根据mapper的namespace划分的，相同namespace的mapper查询数据放在同一个区域，如果使用mapper代理方法每个mapper的namespace都不同，此时可以理解为二级缓存区域是根据mapper划分。

每次查询会先从缓存区域找，如果找不到从数据库查询，查询到数据将数据写入缓存。

Mybatis二级缓存依旧和一级缓存相同使用一个HashMap。

sqlSession执行增删改查等操作之后再提交，会清空缓存。

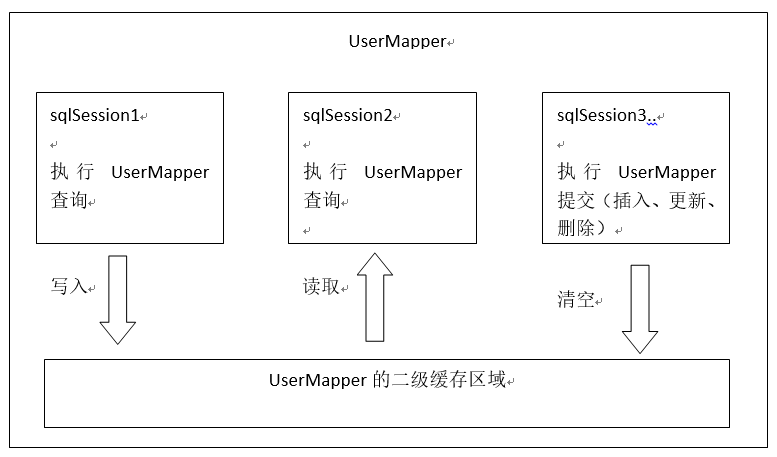


图1-3-5 多个SqlSession请求UserMapper的二级缓存

# 第2章 项目中涉及的工具

## 2.1 Maven工具

Maven是Apache公司的开源项目，用于构建项目的工具，用来依赖管理，这里的依赖就是我们平时在Java项目中俗称的Jar包，使用了Maven之后我们就不需要在项目中手动导入很多的复杂的Jar包。

### 2.1.1 Maven的好处以及实现的原理

使用了Maven之后的项目，相比较传统的Java项目的文件大小要小很多。因为maven项目中没有jar包。

maven的两大核心：

（1）依赖管理:对jar包管理过程

（2）项目构建：项目在编码完成后，对项目进行编译、测试、打包、部署等一系列的操作都通过命令来实现

### 2.1.2 图解Maven查找Jar包的过程

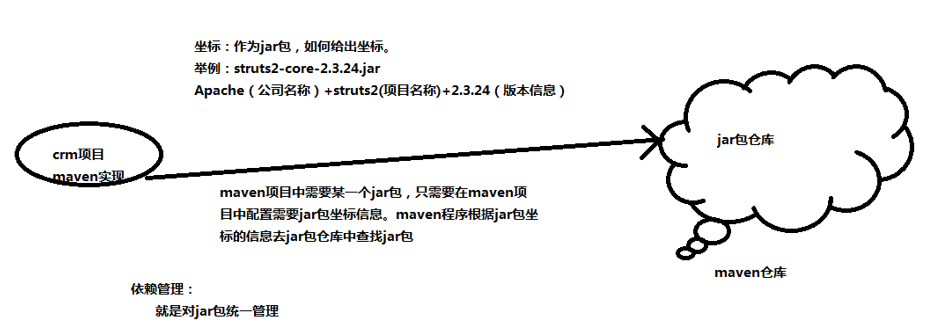


图2-1-1 Maven查找Jar包的过程

### 2.1.3 使用IDE自动创建Maven项目目录结构

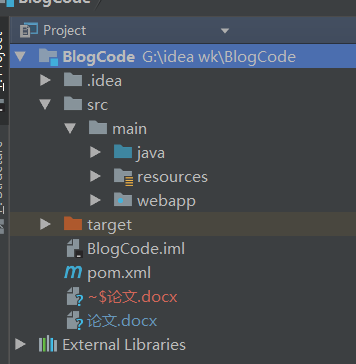


图2-1-2 Maven项目结构图

### 2.1.4 Maven常用命令

clean:清理

将项目根目录下target目录清理掉。

compile:编译

将项目中.java文件编译为.class文件

test:单元测试

单元测试类名有要求：XxxxTest.java

将项目根目录下src/test/java目录下的单元测试类都会执行。

package:打包

web project ---- war包

java project -----jar包

将项目打包，打包项目根目录下taget目录

install:安装

解决本地多个项目公用一个jar包。

打包到本地仓库

### 2.1.5 Maven项目生命周期

在maven中存在三类不同的生命周期，每一类生命周期相互独立，互不影响。在一类生命周期内，执行后面的命令前面操作会自动执行

CleanLifeCycle：清理生命周期

clean

DefaultLifeCycle：默认生命周期

compile,test,package,install,deploy

SiteLifeCycle：站点生命周期

site

## 2.2 Log4j

Log4j也属于Apache下的一个开源项目，在我们的项目中使用Log4j我们可以控制项目中日志的输出，可以看到项目运行过程中更加详细的信息,而且使用非常的简单，只需要在项目中添加jar包以及Log4j配置文件即可。

### 2.2.1 Maven项目中使用Log4j

在Maven项目的Pom.xml文件中，编写如下语句即可引入Log4j依赖

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

然后创建Log4j配置文件文件内容为下图所示

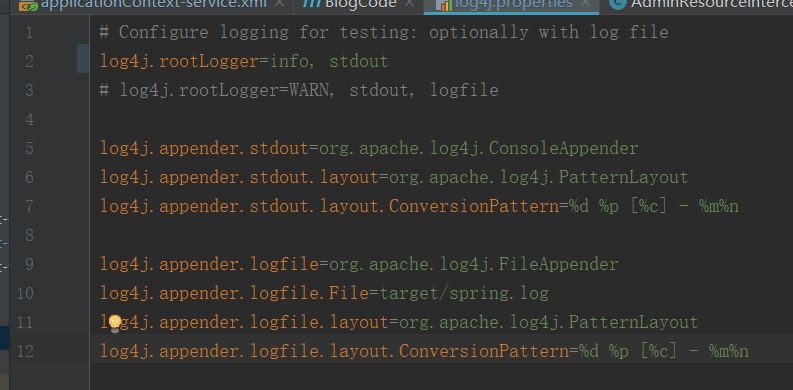


图2-2-1 Log4j配置文件

在第二行代码是设置日志的级别，如果是设置为info，可以看到基本信息，如果设置为Debug可以看到更加详细的日志信息。

## 2.3 Druid数据库连接池

Druid是属于阿里巴巴旗下的一款开源项目，这款数据库连接池被广大的开发者们称之为最强大的Java数据库连接池，它能够提供强大的监控和扩展功能。

### 2.3.1 Maven中使用Druid

在Maven项目的Pom.xml文件中，编写如下语句就可以添加Druid的依赖

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.0.16</version>

</dependency>

# 第3章 SSM框架项目整合的搭建

## 3.1 工具的介绍

### 3.1.2 工具的版本

本次项目的集成开发环境是Intellij idea 2017,使用Jdk1.8版本，Maven使用3.3.9版本，Tomcat7.0版本，Maven中Spring版本锁定为14.2.0，Mybatis版本3.4.0，Junit版本3.8.1，Log4j版本1.2.17，Druid数据库连接池版本1.0.16。

## 3.2 项目具体实施

### 3.2.1 创建Maven项目

使用Intellij idea创建maven项目，项目结构如图所示

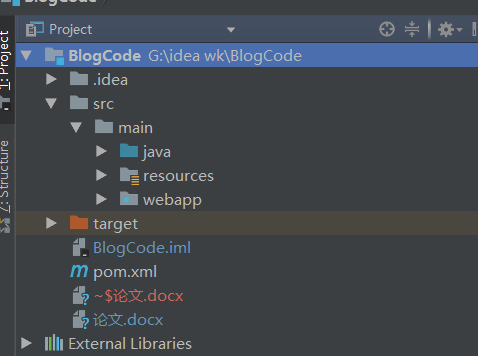


图 3-2-1 Maven项目结构

### 3.2.2 添加依赖

#### 3.2.2.1 添加Spring和Mybatis的依赖（仅部分代码）

点击项目中的pom.xml文件，添加如下代码

<!-- 添加spring支持 -->  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>${spring.version}</version>  
</dependency>  
  
<!-- 添加mybatis支持 -->  
<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>${mybatis.version}</version>  
</dependency>

<dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>1.3.0</version>  
</dependency>

#### 3.2.2.2 添加Jdk,Druid以及 Junit,Log4j,Jdbc等依赖

<!-- jdbc驱动包 -->  
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.30</version>  
</dependency>

<!-- 添加Druid支持 -->  
<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>1.0.16</version>  
</dependency>

<!-- 添加log4j日志 -->  
<dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>1.2.17</version>  
</dependency>

<!-- 添加junit支持 -->  
<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.12</version>  
 <scope>test</scope>  
</dependency>

<!-- 编码和编译和JDK版本 -->  
<plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>2.3.2</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 <encoding>utf8</encoding>  
 </configuration>  
</plugin>

### 3.2.3 web.xml中的部分配置

#### 3.2.3.1 配置post提交乱码的过滤器

<!--post乱码过滤器-->  
<!-- 配置springMVC编码过滤器 -->  
<filter>  
 <filter-name>CharacterEncodingFilter</filter-name>  
 <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  
 <!-- 设置过滤器中的属性值 -->  
 <init-param>  
 <param-name>encoding</param-name>  
 <param-value>UTF-8</param-value>  
 </init-param>  
 <!-- 启动过滤器 -->  
 <init-param>  
 <param-name>forceEncoding</param-name>  
 <param-value>true</param-value>  
 </init-param>  
</filter>  
<!-- 过滤所有请求 -->  
<filter-mapping>  
 <filter-name>CharacterEncodingFilter</filter-name>  
 <url-pattern>/\*</url-pattern>  
</filter-mapping>

#### 3.2.3.2 加载spring容器

<!--加载spring容器-->  
<context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>  
 classpath:spring/applicationContext-\*.xml  
 </param-value>  
</context-param>

<!--配置监听器，来加载spring容器-->  
<listener>  
 <listener-class>  
 org.springframework.web.context.ContextLoaderListener  
 </listener-class>

### 3.2.4 配置Druid连接池以及Mapper扫描器

创建ApplicationContext-dao.xml文件添加Druid配置信息：

<!--加载db.properties-->  
<context:property-placeholder location="classpath:db.properties"/>

<!--配置druid连接池-->  
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">  
 <!-- 基本属性 url、user、password -->  
 <property name="url" value="${jdbc\_url}" />  
 <property name="username" value="${jdbc\_user}" />  
 <property name="password" value="${jdbc\_password}" />

<!--mapper扫描器-->  
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <property name="basePackage" value="com.wbz.demo.mapper"/>  
 <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>  
</bean>

### 3.2.5 配置各类ServiceBean(仅截取部分代码)

创建AppilicationContext-service.xml文件添加如下代码：

<!-- 用户管理的service -->  
<bean id="userService" class="com.wbz.demo.service.impl.UserServiceImpl"/>  
  
<!-- 文章管理的service -->  
<bean id="articleService" class="com.wbz.demo.service.impl.ArticleServiceImpl"/>

### 3.2.6 配置事务及相关

创建ApplicationContext-transaction.xml添加如下代码：

<!-- 事务管理器 -->  
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
 <property name="dataSource" ref="dataSource"/>  
</bean>  
  
<!-- 通知 -->  
<tx:advice id="txAdvice" transaction-manager="transactionManager">  
 <tx:attributes>  
 <tx:method name="save\*" propagation="REQUIRED"/>  
 ….  
 </tx:attributes>  
</tx:advice>  
<aop:config>  
 <aop:advisor advice-ref="txAdvice" pointcut="execution(\* com.wbz.demo.service.impl.\*.\*(..))"/>  
</aop:config>

### 3.2.7 SpringMVC的配置文件（仅截取部分代码）

<!--解决映射器和适配器的配置注解配置 -->  
<mvc:annotation-driven/>  
<!-- 配置视图解析器-->  
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver" >  
 <!--配置前缀和后缀-->  
 <property name="prefix" value="/WEB-INF/view/"/>  
 <property name="suffix" value=".jsp"/>  
</bean>  
  
<!--文件上传配置-->  
<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  
 <!--设置上传最大尺寸为50MB-->  
 <property name="maxUploadSizePerFile" value="52428800"/>  
 <property name="defaultEncoding" value="UTF-8"/>  
 <property name="resolveLazily" value="true"/>  
</bean>

### 3.2.8 Mapper的开发（部分代码）

创建接口UserMapperCustom,编写如下代码：

public interface UserMapperCustom {  
 //获得用户列表  
 public List<UserCustom> listUser() throws Exception;  
 //根据用户名查用户  
 public User getUserByName(String name) throws Exception;

创建UserMapperCustom.xml,其中的namespace是mapper接口的地址，编写如下代码：

<mapper namespace="com.wbz.demo.mapper.custom.UserMapperCustom" >  
 <sql id="user\_table\_all\_columns">  
 user\_id,  
 user\_name,  
 user\_pass,  
 </sql>  
  
 <!--用户列表查询-->  
 <select id="listUser" parameterType="com.wbz.demo.entity.custom.UserCustom"  
 resultType="com.wbz.demo.entity.custom.UserCustom">  
 SELECT  
 <include refid="user\_table\_all\_columns"/>  
 FROM `user`  
 ORDER BY `user\_status` ASC  
 </select>

### 3.2.9 表现层的开发（部分代码）

创建AdminController.java文件，编写如下代码：  
  
 //后台用户列表显示  
 @RequestMapping(value = "")  
 public ModelAndView userList() throws Exception {  
 ModelAndView modelandview = new ModelAndView();  
  
 List<UserCustom> userCustomList = userService.listUser();  
 modelandview.addObject("userCustomList",userCustomList);  
  
 modelandview.setViewName("Admin/User/index");  
 return modelandview;  
  
 //后台添加用户页面显示  
 public ModelAndView insertUserView() throws Exception {  
 ModelAndView modelAndView = new ModelAndView();  
  
 modelAndView.setViewName("Admin/User/insert");  
 return modelAndView;  
 }

### 3.2.10 业务逻辑层的开发（部分代码）

1.创建接口UserService,编写如下代码：

public interface UserService {  
 public List<UserCustom> listUser() throws Exception;  
   
 //根据id查询用户信息  
 public UserCustom getUserById(Integer id) throws Exception;  
   
 //修改用户信息  
 public void updateUser(User user) throws Exception;

2.创建UserServiceImpl实现类，实现UserService,编写如下代码：  
public class UserServiceImpl implements UserService {  
 public List<UserCustom> listUser() throws Exception {  
 List<UserCustom> userCustomList = userMapperCustom.listUser();  
 for(int i=0;i<userCustomList.size();i++) {  
 Integer articleCount = userMapperCustom.countArticleByUser(userCustomList.get(i).getUserId());  
 userCustomList.get(i).setArticleCount(articleCount);  
 }  
 return userCustomList;  
 }  
   
 public UserCustom getUserById(Integer id) throws Exception {  
 User user = userMapper.selectByPrimaryKey(id);  
 UserCustom userCustom = new UserCustom();  
 BeanUtils.copyProperties(user,userCustom);  
 return userCustom;  
 }

# 第4章 主要功能实现

## 4.1 登陆功能的实现及结果展示

本模块主要实现的是用户登陆的模块，在用户通过正确的用户名以及密码才可以登陆成功，否则系统提示登陆失败。部分代码如下所示：

1.后台代码  
//登录验证  
public String loginVerify(HttpServletRequest request, HttpServletResponse response) throws Exception {  
 Map<String, Object> map = new HashMap<String, Object>();  
 String username = request.getParameter("username");  
 String password = request.getParameter("password");  
 String rememberme = request.getParameter("rememberme");  
 User user = userService.getUserByNameOrEmail(username);  
 if(user==null) {  
 map.put("code",0);  
 map.put("msg","用户名无效！");  
 } else if(!user.getUserPass().equals(password)) {  
 map.put("code",0);  
 map.put("msg","密码错误！");  
 } else {  
 //登录成功  
 map.put("code",1);  
 map.put("msg","");

2.前端代码

<form name="loginForm" id="loginForm" method="post">  
 <p>  
 <label for="user\_login">用户名或电子邮件地址<br />  
 <input type="text" name="username" id="user\_login" class="input" value="<%=username%>" size="20" required/></label>  
 <label for="user\_pass">密码<br />  
 <input type="password" name="password" id="user\_pass" class="input" value="<%=password%>" size="20" required/>  
 </label>  
 </p>

<%--登录验证--%>  
$("#submit-btn").click(function () {  
 var user = $("#user\_login").val();  
 var password = $("#user\_pass").val();  
 if(user=="") {  
 alert("用户名不可为空!");  
 } else if(password==""){  
 alert("密码不可为空!");

### 4.1.1 登陆结果展示

1.数据库中的数据：

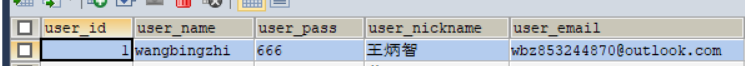


图4-1-1 数据库用户数据

2.前端登陆页面：



图4-1-2 前端登陆页面显示

3.登陆成功的显示：

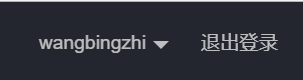


图4-1-3 登陆成功显示

## 4.2 照片发布的实现及结果展示

本模块实现的是用户发布照片的操作，其中需要用到tomcat的虚拟路径来存储用户上传到服务器中的照片，在本次的项目中，我们直接在intellij idea中创建tomcat的虚拟路径，再通过照片路径来存储到数据库中，并且在前端页面中显示用户发布的照片，部分代码实现如下所示：

后台代码：

通过mybatis的逆向工程，创建文章mapper:

public interface ArticleMapper {  
 int deleteByPrimaryKey(Integer articleId);  
 int insert(Article record);  
 int insertSelective(Article record);  
 Article selectByPrimaryKey(Integer articleId);  
}

创建UploadFileController.java,用于管理上传的照片按照时间分类存入本地服务器中，

//本地使用,上传位置  
String rootPath ="F:\\uploads\\";  
//文件的完整名称,如spring.jpeg  
String filename = file.getOriginalFilename();  
//文件名,如spring  
String name = filename.substring(0,filename.indexOf("."));  
//文件后缀,如.jpeg  
String suffix = filename.substring(filename.lastIndexOf("."));  
//创建年月文件夹  
Calendar date = Calendar.getInstance();  
File dateDirs = new File(date.get(Calendar.YEAR)  
 + File.separator + (date.get(Calendar.MONTH)+1));  
//目标文件  
File descFile = new File(rootPath+File.separator+dateDirs+File.separator+filename);  
int i = 1;  
//若文件存在重命名  
String newFilename = filename;  
while(descFile.exists()) {  
 newFilename = name+"("+i+")"+suffix;  
 String parentPath = descFile.getParent();  
 descFile = new File(parentPath+File.separator+newFilename);  
 i++;  
//判断目标文件所在的目录是否存在  
if(!descFile.getParentFile().exists()) {  
 //如果目标文件所在的目录不存在，则创建父目录  
 descFile.getParentFile().mkdirs();  
//将内存中的数据写入磁盘  
file.transferTo(descFile);  
//完整的url  
String fileUrl = "/pics/"+dateDirs+ "/"+newFilename;  
ResultVO resultVO = new ResultVO();  
resultVO.setCode(0);  
resultVO.setMsg("成功");  
  
UploadFileVO uploadFileVO = new UploadFileVO();  
uploadFileVO.setTitle(filename);  
uploadFileVO.setSrc(fileUrl);  
resultVO.setData(uploadFileVO);  
return resultVO;

创建BackPageController.java, 执行发布图片之后的提交操作，编写如下代码：

//后台添加页面提交操作  
@RequestMapping(value = "/insertSubmit",method = RequestMethod.POST)  
public String insertPageSubmit(Page page) throws Exception {  
 //判断别名是否存在  
 PageCustom checkPage = pageService.getPageByKey(null,page.getPageKey());  
 if(checkPage==null) {  
 page.setPageCreateTime(new Date());  
 page.setPageUpdateTime(new Date());  
 page.setPageStatus(1);  
 pageService.insertPage(page);  
 }  
 return "redirect:/admin/page";  
}

2.前端页面部分代码展示，基于Layui上传照片的编辑页面以及前端显示文章的标题和内容主体：

先创建一个基于Layui的内容编辑器  
//创建一个编辑器  
var editIndex = layedit.build('content',{  
 height:350,  
 }

编写编辑器中的功能图标：

layedit.build('content', {  
 tool: [  
 'strong' //加粗  
 …  
 ,'image' //插入图片

发布的照片的内容详细页面：

<main id="main" class="site-main" role="main">  
 <article class="post">  
 <header class="entry-header">  
 <h1 class="entry-title">  
 ${articleDetailVo.articleCustom.articleTitle}  
 </h1>  
 </header><!-- .entry-header -->  
 <div class="entry-content">  
 <div class="single-content">  
 ${articleDetailVo.articleCustom.articleContent}  
 </div>

3.Idea中创建tomcat的虚拟路径为/pics用于专门存放图片，对应本机中的uploads文件夹，如图所示：

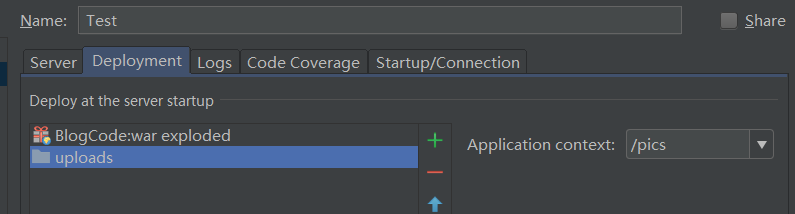


图4-2-1 Idea中tomcat虚拟路径

### 4.2.1 照片发布结果展示

添加标题以及图片：

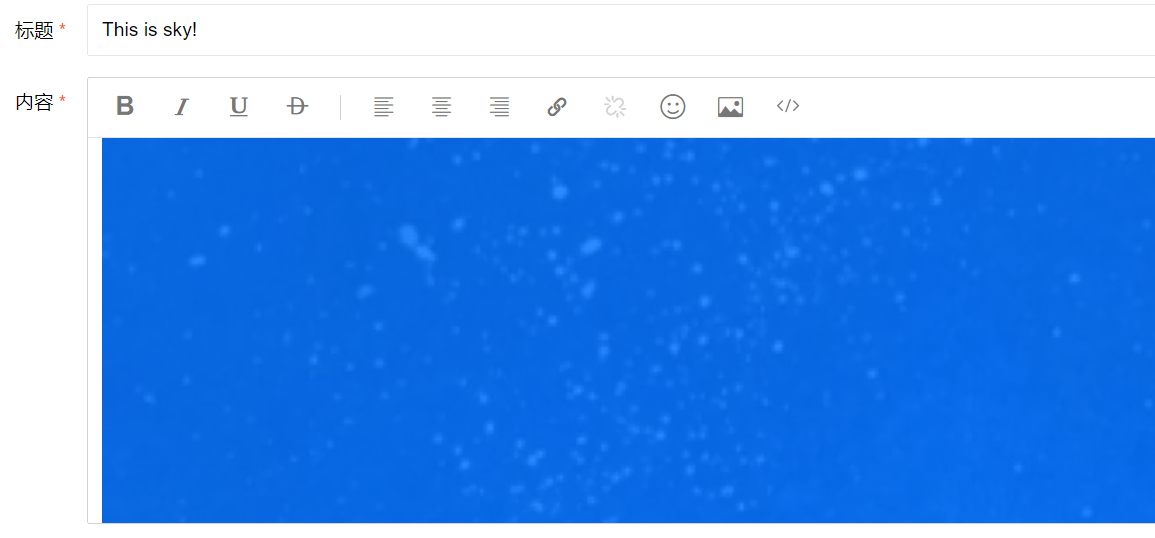


图4-2-2 照片发布演示

前端页面显示：

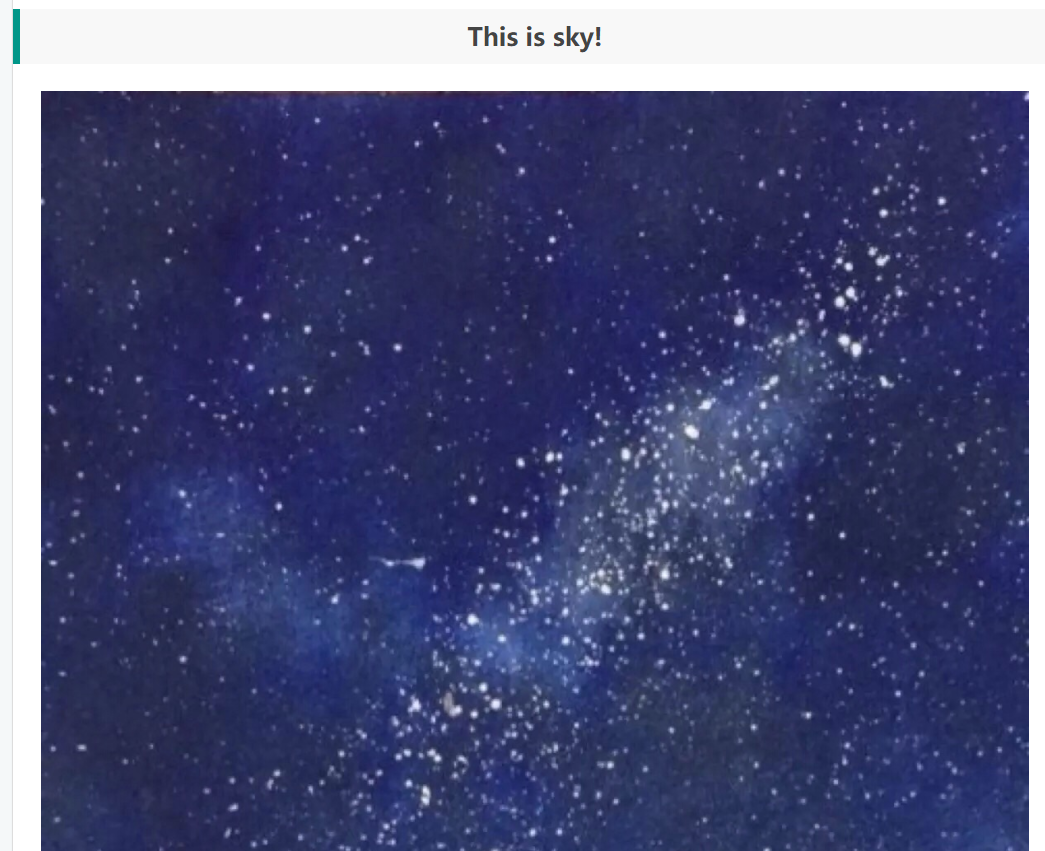


图4-2-3 前端页面展示

数据库中显示该文章：

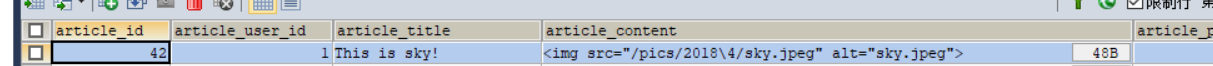


图4-2-4 数据库展示文章

## 4.3 用户点赞的实现及结果展示

该功能用户通过点击前端页面的点赞图标，并且后台数据库记录点赞数量，从而显示到前端页面的点赞图标中，并且限制同一个用户不能重复点赞

1.后端代码实现

Article.java中编写文章的点赞属性，并且创建geter/seter方法

private Integer articleLikeCount;

public Integer getArticleLikeCount() {  
 return articleLikeCount;  
}  
  
 public void setArticleLikeCount(Integer articleLikeCount) {  
 this.articleLikeCount = articleLikeCount;  
}

在ArticleController.java中添加文章点赞数的逻辑：

//文章点赞数增加  
public Integer increaseLikeCount(@PathVariable("id") Integer id)  
 throws Exception {  
 ArticleCustom articleCustom = articleService.getArticleById(1,id);  
 int articleCount = articleCustom.getArticleLikeCount();  
 articleCustom.setArticleLikeCount(articleCount + 1);  
 articleService.updateArticle(id, articleCustom);  
 return articleCount+1;  
}

2.前端页面文章点赞界面逻辑：

<span class="like">  
 <a href="javascript:;" data-action="ding" data-id="1" title="点赞"  
 class="favorite" onclick="increaseLikeCount()">  
 <i class="fa fa-thumbs-up"></i>赞  
 <i class="count"  
 id="count-${articleDetailVo.articleCustom.articleId}">${articleDetailVo.articleCustom.articleLikeCount}</i>

### 4.3.1 点赞结果展示

1.点赞前，前端页面显示点赞图片及点赞数量

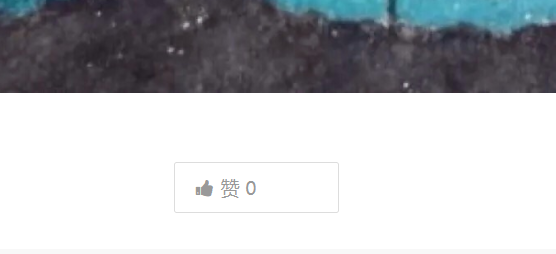


图4-3-1 点赞前图标

2.点赞后，前端页面显示点赞图片及点赞数

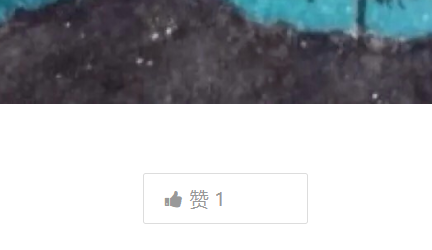


图4-3-2 点赞后图标

3.数据库显示点赞数

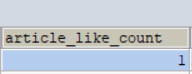


图4-3-3 数据库显示图片被点赞数量

## 4.4 上一篇或下一篇文章功能实现及展示

该功能主要是在前端页面中用基本的JSTL表达式，显示上一篇或者下一篇文章的名字，并且点击之后跳转到该文章，实现的部分代码如下所示：

在ArticleDetail.jsp文件的开头引入jstl,代码如下：

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<c:choose>  
 <c:when test="${preArticle!=null}">  
 <a href="/article/${preArticle.articleId}" rel="next">  
 <span class="meta-nav">  
 <span class="post-nav">上一篇  
 <i class="fa fa-angle-left"></i>  
 </span>  
 <br>${preArticle.articleTitle}  
 </span>  
 </a>  
 </c:when>  
 <c:otherwise>  
 <span class="meta-nav">  
 <span class="post-nav">  
 没有了<br>  
 </span>已是第一篇文章  
 </span>  
 </c:otherwise>  
 </c:choose>  
 <c:choose>  
 <c:when test="${afterArticle!=null}">  
 <a href="/article/${afterArticle.articleId}" rel="next">  
 <span class="meta-nav">  
 <span class="post-nav">下一篇  
 <i class="fa fa-angle-right"></i>  
 </span>  
 <br>${afterArticle.articleTitle}  
 </span>  
 </c:when>  
 <c:otherwise>  
 <span class="meta-nav">  
 <span class="post-nav">  
 没有了<br>  
 </span>已是最后文章

### 4.4.1 上一篇下一篇文章功能展示



图4-4-1 上一篇下一篇文章功能展示

## 4.5 搜索文章的功能及结果展示

该功能需要在搜索框中输入需要搜索的内容关键字，通过在数据库中的查找与关键字相关的匹配之后，给用户反馈出来相关的结果，若存在的多条数据对搜索出来的结果进行分页显示，前端页面的美化部分用到了layui的美化效果，功能部分代码如下:

1.后端代码

//文章搜索实现  
@RequestMapping("/search")  
@ResponseBody  
public ModelAndView SearchPageView(HttpServletRequest request,Model model) throws Exception {  
 ModelAndView modelAndView = new ModelAndView();  
 //设置每页显示的数量  
 int pageSize = 10;  
 String query = request.getParameter("query");  
 List<ArticleSearchVo> articleSearchVoList = articleService.listSearchResultByPage(1,request,model,null,pageSize,query);  
 if(articleSearchVoList!=null) {  
 modelAndView.addObject("articleSearchVoList", articleSearchVoList);  
 } else {  
 modelAndView.addObject("articleSearchVoList", null);  
 }  
 modelAndView.setViewName("Home/Page/search");  
 return modelAndView;  
}

2.前端代码

<%--搜索框 start--%>  
 <div class="searchbar">  
 <form method="get" id="searchform" action="/search">  
 <span>  
 <input type="text" value="" name="query" id="s" placeholder="输入搜索内容"required="">  
 <button type="submit" id="searchsubmit">搜索</button>

通过JSTL对搜索出来的文章作分页显示

<c:choose>  
 <c:when test="${articleSearchVoList[0].page.totalPageCount <= 3 }">  
 <c:set var="begin" value="1"/>  
 <c:set var="end" value="${articleSearchVoList[0].page.totalPageCount }"/>  
 </c:when>  
 <c:otherwise>  
 <c:set var="begin" value="${articleSearchVoList[0].page.pageNow-1 }"/>  
 <c:set var="end" value="${articleSearchVoList[0].page.pageNow + 2}"/>  
 <c:if test="${begin < 2 }">  
 <c:set var="begin" value="1"/>  
 <c:set var="end" value="3"/>  
 </c:if>  
 <c:if test="${end > articleSearchVoList[0].page.totalPageCount }">  
 <c:set var="begin" value="${articleSearchVoList[0].page.totalPageCount-2 }"/>  
 <c:set var="end" value="${articleSearchVoList[0].page.totalPageCount }"/>  
 </c:if>

### 4.5.1 搜索功能结果展示

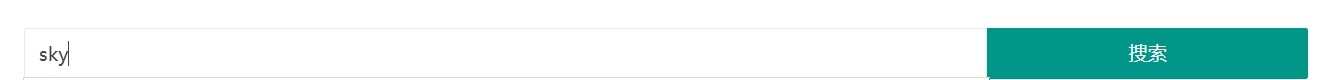


图4-5-1 搜索框图

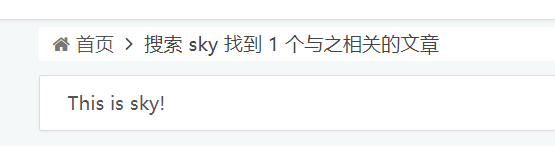


图4-5-2 搜索结果

## 4.6 图片文章删除功能及结果展示

该部分功能的实现主要是对于数据库的操作，只有该文章的创建人或者是后台管理用户才可以在后台中删除文章，在网站的后台点击删除文章按钮，后端响应，从数据库中删除该篇文章，该文章从前端代码消失，删除部分代码如下所示：

1.后端代码：

在ArticleController.java中执行的删除文章操作

//删除文章  
@RequestMapping(value = "/delete/{id}")  
public void deleteArticle(@PathVariable("id") Integer id) throws Exception {  
 //调用service批量删除  
 articleService.deleteArticle(id);

在ArticleService.java中执行操作

//删除文章  
public void deleteArticle(Integer id) throws Exception;

在ArticleMapper.xml中执行操作

<delete id="deleteByPrimaryKey" parameterType="java.lang.Integer" >  
 delete from article where article\_id = #{articleId,jdbcType=INTEGER}  
</delete>

2.前台代码

<a href="/admin/article/edit/${a.articleCustom.articleId}"  
 class="layui-btn layui-btn-mini">编辑</a>  
<a href="javascript:void(0)"  
 onclick="deleteArticle(${a.articleCustom.articleId})"  
 class="layui-btn layui-btn-danger layui-btn-mini">删除</a>

### 4.6.1 图片删除功能展示

1.后台删除文章展示



图4-6-1 删除图片列表

2.数据库中显示该图片文章存在与否



图4-6-2 删除前数据库中存在该图片文章



图4-6-3 删除后数据库中不存在该图片文章

## 4.7 图片分类功能实现及结果展示

该功能的可以在发布图片的时候选择图片分类，后台中实现添加文章分类，删除文章分类，以及编辑文章分类名称，该功能分成两个节点，有一个父节点，父节点下可以存在多个子节点，在数据库中两者也是用Pid(父节点)和id(子节点)作区分，在添加分类的时候，直接编写节点名称之后，父节点选择无，就默认把该节点设置为一个父节点，如果选择了父节点，则该节点为子节点，并且有文章的分类是无法删除的，部分代码实现如下：

1. 后台代码

创建Category.java，设置其属性

public class Category {  
 private Integer categoryId;  
  
 private Integer categoryPid;  
  
 private String categoryName;

创建CategoryMapperCustom.java接口，编写如下方法：  
//获得分类列表  
public List<CategoryCustom> listCategory(@Param(value = "status") Integer status) throws Exception;  
//根据分类id获得分类信息  
public CategoryCustom getCategoryById(@Param(value = "status") Integer status, @Param(value = "id") Integer id) throws Exception;

//根据父分类找子分类  
public List<CategoryCustom> findChildCategory(@Param(value = "status") Integer status, @Param(value = "id") Integer id) throws Exception;

创建CategoryMapperCustom.xml中编写sql语句执行分类数据库操作：

<!--获得分类总数-->  
<select id="countCategory" resultType="Integer">  
 SELECT COUNT(\*) FROM `category`  
 <where>  
 <if test="status!=null">  
 category\_status=#{status}  
 </if>  
  
<!--获得含有该分类的文章列表-->  
<selectid="listArticleWithCategoryByPage" resultType="com.wbz.demo.entity.custom.ArticleCustom">  
 SELECT \* FROM `article`  
 <where>  
 <if test="status!=null">  
 article\_status=#{status} AND  
 </if>  
 article\_parent\_category\_id=#{cateId} OR  
 article\_child\_category\_id=#{cateId}  
 </where>  
 ORDER BY article\_id DESC  
 limit #{startPos},#{pageSize}  
</select>

Service中的部分分类相关方法  
//获得分类总数  
public Integer countCategory(Integer status) throws Exception;  
//获得带有该分类的文章列表  
public List<ArticleListVo> listArticleWithCategoryByPage(Integer status, Integer pageNow, Integer pageSize, Integer cateId) throws Exception;

//获得某个分类信息  
public CategoryCustom getCategory(Integer status, Integer id) throws Exception;  
//删除分类  
public void deleteCategory(Integer id) throws Exception;  
//根据id查询分类信息  
public CategoryCustom getCategoryById(Integer status, Integer id) throws Exception;

2.前端部分分类相关代码

<div class="layui-input-block">  
 名称 <span style="color: #FF5722; ">\*</span>  
 <input type="text" name="categoryName" placeholder="请输入分类名称" autocomplete="off" class="layui-input" required>

<div class="layui-input-block">  
 父节点 <span style="color: #FF5722; ">\*</span>  
 <select name="categoryPid" class="layui-input" required>  
 <option value="0">无</option>  
 <c:forEach items="${categoryCustomList}" var="c">  
 <c:if test="${c.categoryPid==0}">  
 <option value="${c.categoryId}">${c.categoryName}</option>  
 </c:if>

3.分类部分数据库代码展示

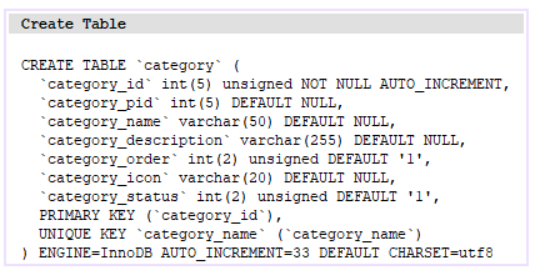


图4-7-1 分类数据库代码

### 4.7.1 分类结果展示

1.前端页面效果展示

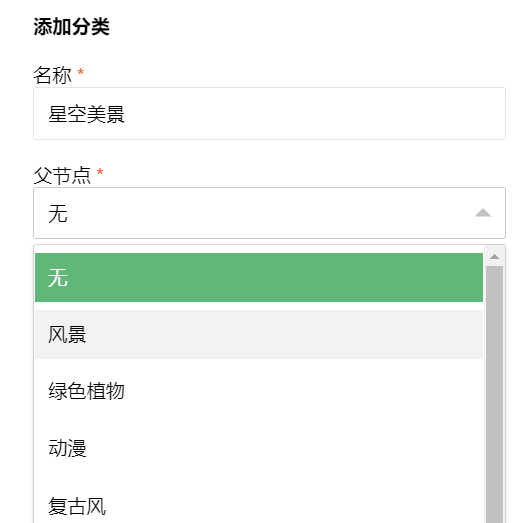


图4-7-2 分类前端页面

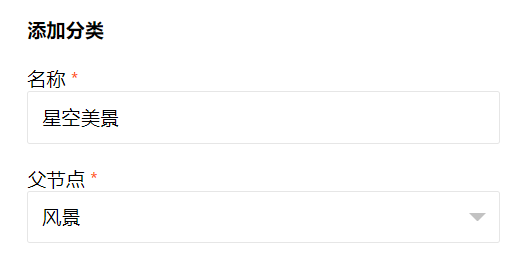


图4-7-3 分类父节点子节点

2.数据库中显示添加的分类细腻





图4-7-4 数据库中的分类信息

# 第5章 总结

## 5.1 总结

本次项目的实践中基于SSM(SpringMVC+Spring+Mybatis)框架实现的照片分享网站后台开发,再配上Mysql,Tomcat以及Maven项目管理工具，使用了阿里巴巴数据库连接池Druid。

在项目实践全部由一人完成，由于对于前端不是很懂，所以网站的前端页面由爬取别人的页面在填写上自己后台的接口所成，期间遇到了很多的困难和挫折，但是在克服了这些困难和挫折之后感觉风雨过后就是彩虹，Maven的加入使项目构建更加的简单化，项目中并没有将Tomcat融合进入Maven中，而是单独在集成开发环境中使用的Tomcat,Log4j的使用在项目运行过程中，对于项目的突发情况掌控更加的细节，让我对于SSM框架有了更加系统的认识，以前的时候对于SSM框架仅仅是停留在会用的阶段上面，现在对于SSM框架的底层有了更进一步的认识，SpringMVC的注解使用让人更加的喜欢，明白了Spring框架底层Ioc的实现原理，对于Java的反射原理也有了更加深刻的印象，对于HashMap的使用更加的熟练。前端页面中使用了大量的JSTL标签，对于JSTL标签的使用也是更进一步，因为本项目并没有按照Dao层开发，而是使用的和Mybatis融合很好的Mapper代理开发，相比较Hibernate来说，Mybatis给了我更加好的印象，Hibernate体系感觉非常的庞大复杂，虽然Hibernate不需要自己编写Sql语句，但是使用过程特别的繁杂，特别是多表联合的操作上会让我这个新手开发觉得迷糊，Mybatis可以程序员自己编写Sql语句，在很多的使用上相较于Hibernate更加的简便，不至于会让新手感觉迷糊，所以本次的持久层框架采用了Mybatis并且Hibernate。

# 参考文献

1. 张桂珠 [刘丽](https://www.baidu.com/s?wd=%E5%88%98%E4%B8%BD&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3) 陈爱国 .Java面向对象程序设计（第2版）[北京邮电大学出版社](https://www.baidu.com/s?wd=%E5%8C%97%E4%BA%AC%E9%82%AE%E7%94%B5%E5%A4%A7%E5%AD%A6%E5%87%BA%E7%89%88%E7%A4%BE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3)
2. 毕广吉.Java程序设计实例教程[M]. 北京：[冶金工业出版社](https://www.baidu.com/s?wd=%E5%86%B6%E9%87%91%E5%B7%A5%E4%B8%9A%E5%87%BA%E7%89%88%E7%A4%BE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3)，2007年
3. 王[保罗](https://www.baidu.com/s?wd=%E4%BF%9D%E7%BD%97&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3).Java面向对象程序设计[M]. 北京：清华大学出版社，2003年
4. 高林，[周海燕](https://www.baidu.com/s?wd=%E5%91%A8%E6%B5%B7%E7%87%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3).Jsp网上书店系统与案例分析[M]. 北京：[人民邮电出版社](https://www.baidu.com/s?wd=%E4%BA%BA%E6%B0%91%E9%82%AE%E7%94%B5%E5%87%BA%E7%89%88%E7%A4%BE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3)，2004年
5. [刘腾红](https://www.baidu.com/s?wd=%E5%88%98%E8%85%BE%E7%BA%A2&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3)，孙细明.信息系统分析与设计[M]. 北京：[科学出版社](https://www.baidu.com/s?wd=%E7%A7%91%E5%AD%A6%E5%87%BA%E7%89%88%E7%A4%BE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3)，2003年[6].林邦杰，彻底研究java.北京：[电子工业出版社](https://www.baidu.com/s?wd=%E7%94%B5%E5%AD%90%E5%B7%A5%E4%B8%9A%E5%87%BA%E7%89%88%E7%A4%BE&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPHcknhfsnH6sn1PBPjwb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTknW63njf3)，2002年
6. 邓成功，张敏. 基于用户体验的交互式信息效劳模型构建[J]. 中国图书馆学报. 2009(01)
7. 江泽民. 新时期我国信息技术产业的开展[J]. 上海交通大学学报. 2008(10)
8. 吴宏森，宋顺林. MVC架构在工程项目管理零碎中的使用[J]. 微计算机信息. 2006(22)
9. 王华, 马亮, 顾明. 线程池[技术](http://lunwen.qcrx.cn/410/jishulunwen.html)研究与应用. 计算机应用研究. 2005. 22( 11) :141-142
10. 杨开杰, 刘秋菊, 徐汀荣. 线程池的多线程并发控制技术研究. 计算机应用与软件. 2010, 27(1): 168-170, 179
11. 李刚, 金蓓弘. 基于线程的并发控制技术研究与应用. 计算机[工程](http://lunwen.qcrx.cn/400/gongchenglunwen.html). 2007,30(14): 43-45
12. John Calcote. Thread Pools and Server Performance. Dr.Dobbs Journal. 1997,20(7): 60-64

# 致 谢

本次毕业设计的完成跟胡老师的帮助是密不可分的，在此对胡老师表示衷心的感谢，不知不觉三年时间已过，我们面临毕业，三年来胡老师兢兢业业，对学生们的照顾有加，使我们在Java的学习上更加的轻松，对于越来越多的人涌向Java学习，虽然是一个非常好的现象，壮大了Java家族，但是竞争也是更加的激烈，所以胡老师一直告诫我们要多了解技术的底层原理，只有这样子才可以做到一通百通，举一反三，让我们在今后对于其他的技术学习上学的更好学得更快。再次感谢胡老师对于我们学习过程中的付出。