# 实验六 Spring Boot整合MyBatis

一、使用Spring Initializr方式新建Spring Boot项目

1.基础环境搭建

在MySQL中，创建springbootdata数据库，并在其中创建表t\_article和t\_comment

数据库脚本如下：

--创建数据库

create database springbootdata;

--打开数据库

use springbootdata;

-- Table structure for `t\_article`

DROP TABLE IF EXISTS `t\_article`;

CREATE TABLE `t\_article` (

`title` varchar(200) DEFAULT NULL COMMENT '文章标题',

`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '文章id',

`content` longtext COMMENT '文章内容',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=3 DEFAULT CHARSET=utf8;

-- Records of t\_article

INSERT INTO `t\_article` VALUES ('SpringBoot基础入门', '1', '从入门到精通讲解...');

INSERT INTO `t\_article` VALUES ('SpringCloud基础入门', '2', '从入门到精通讲解...');

-- Table structure for `t\_comment`

DROP TABLE IF EXISTS `t\_comment`;

CREATE TABLE `t\_comment` (

`id` int(11) NOT NULL COMMENT '评论id',

`content` longtext COMMENT '评论内容',

`author` varchar(200) DEFAULT NULL COMMENT '评论作者',

`a\_id` int(20) DEFAULT NULL COMMENT '关联的文章id',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Records of t\_comment

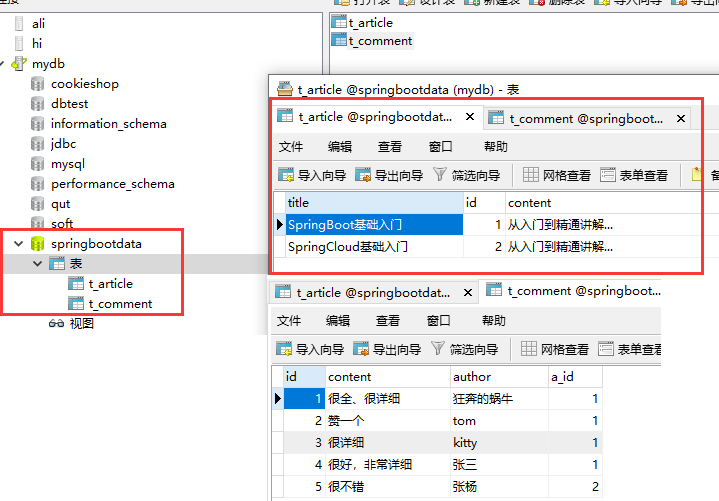
INSERT INTO `t\_comment` VALUES ('1', '很全、很详细', '狂奔的蜗牛', '1');

INSERT INTO `t\_comment` VALUES ('2', '赞一个', 'tom', '1');

INSERT INTO `t\_comment` VALUES ('3', '很详细', 'kitty', '1');

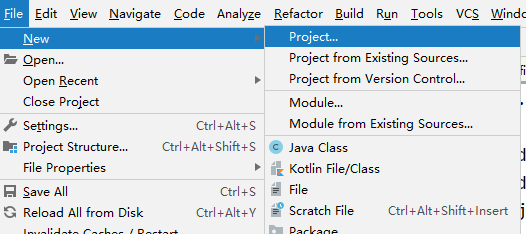
INSERT INTO `t\_comment` VALUES ('4', '很好，非常详细', '张三', '1');

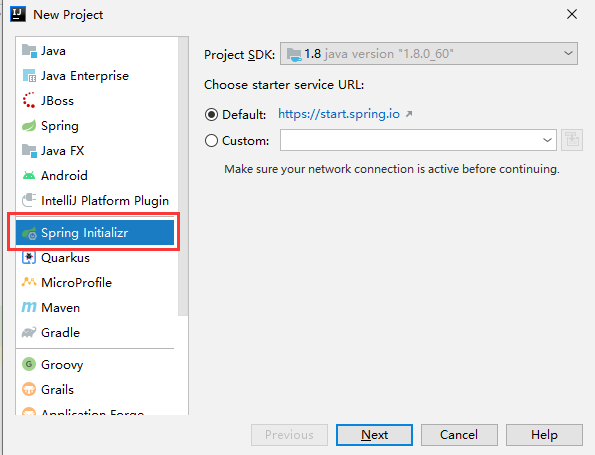
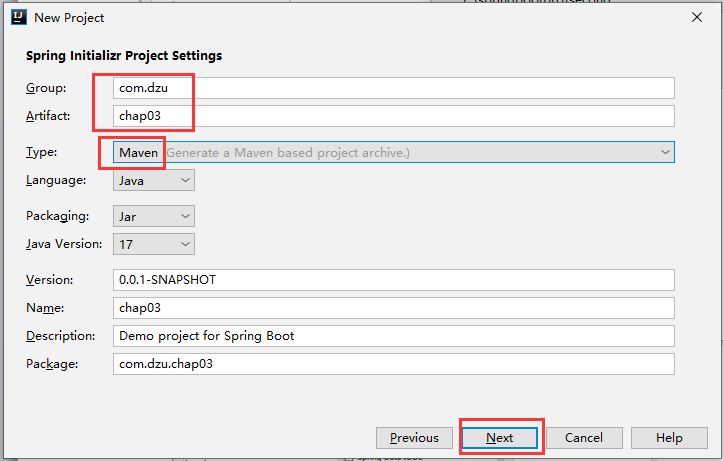
INSERT INTO `t\_comment` VALUES ('5', '很不错', '张杨', '2');



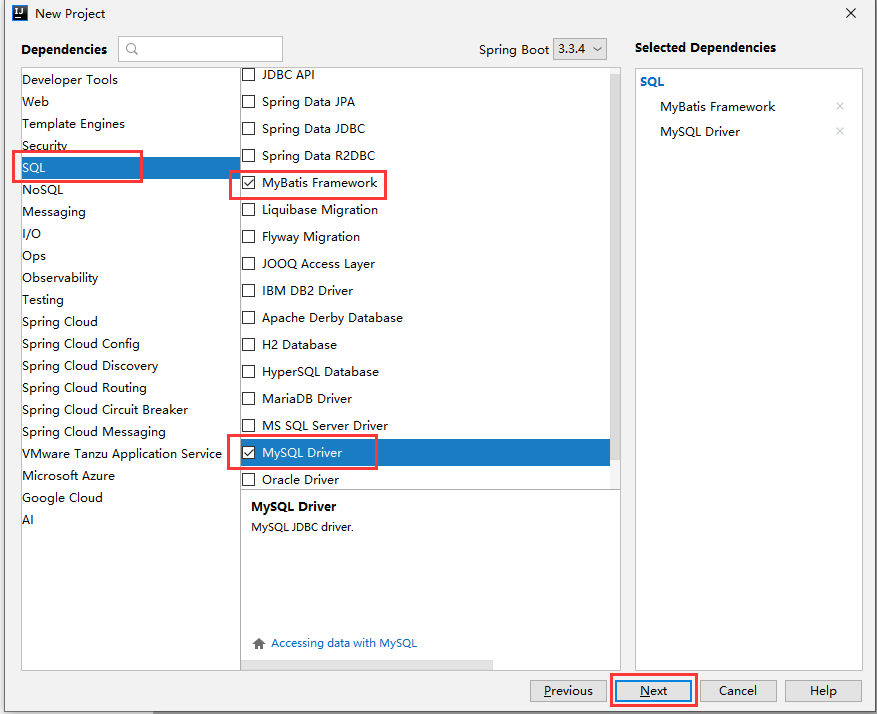
2.创建项目并引入相应的启动器

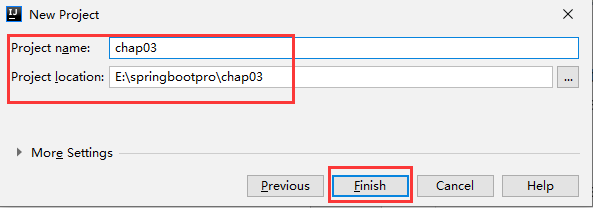
（1）使用Spring Initializr方式新建项目



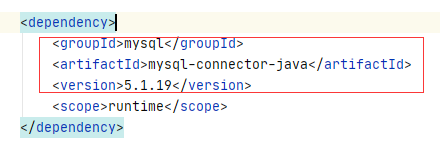
（2）选择SQL和MyBatis Framework依赖



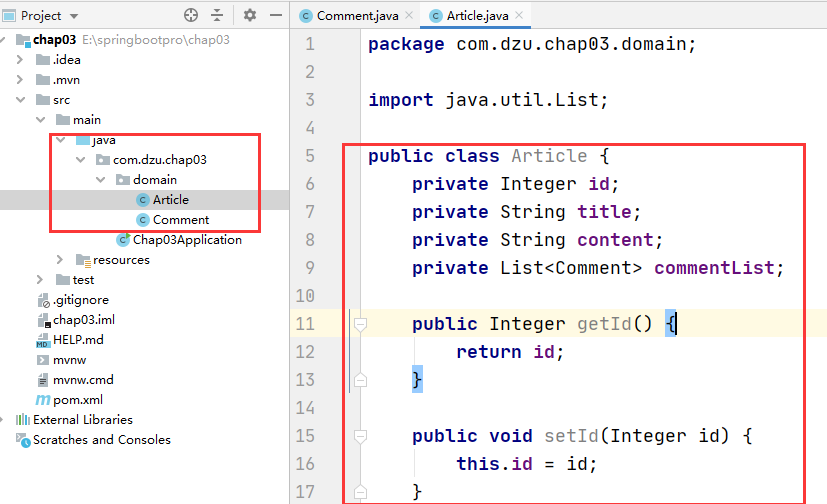


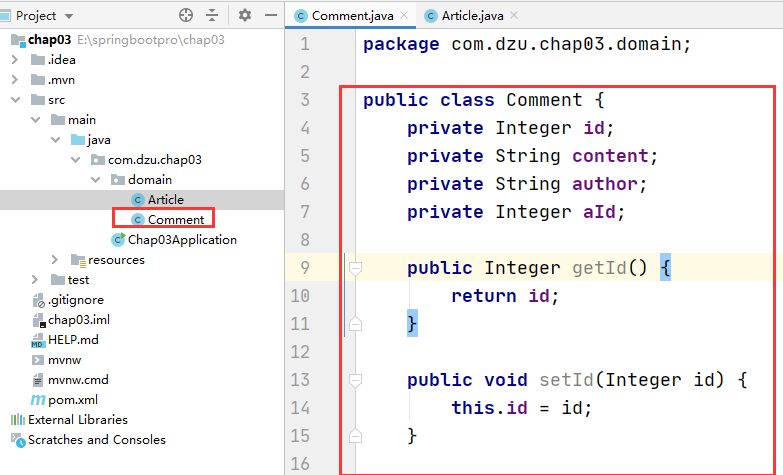
（3）在pom.xml中修改Java版本、父工程版本、MyBatista版本，设置Maven配置文件和本地仓库文件



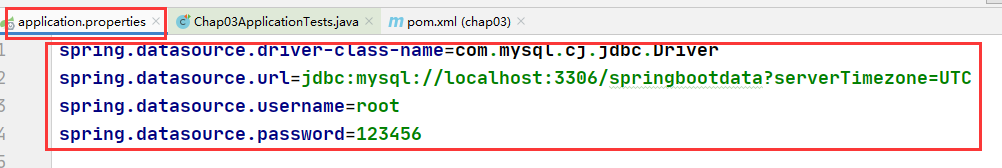


（4）编写对应实体类article和comment，自动生成getter、setter和toString方法



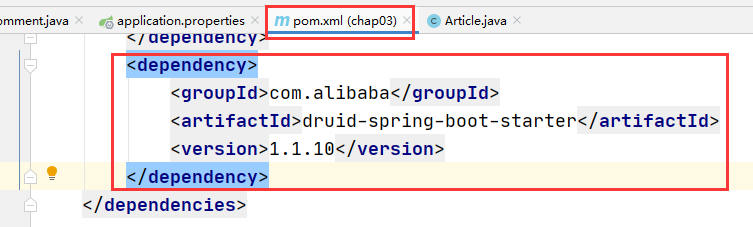


（5）进行全局配置



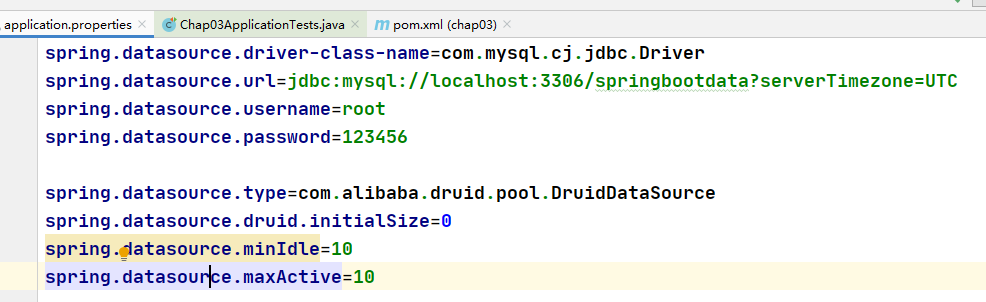
**spring.datasource.driver-class-name**=**com.mysql.cj.jdbc.Driver  
spring.datasource.url**=**jdbc:mysql://localhost:3306/springbootdata?serverTimezone=UTC  
spring.datasource.username**=**root  
spring.datasource.password**=**123456**

（6）数据源类型选择配置



<**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>druid-spring-boot-starter</**artifactId**>  
 <**version**>1.1.10</**version**>  
</**dependency**>

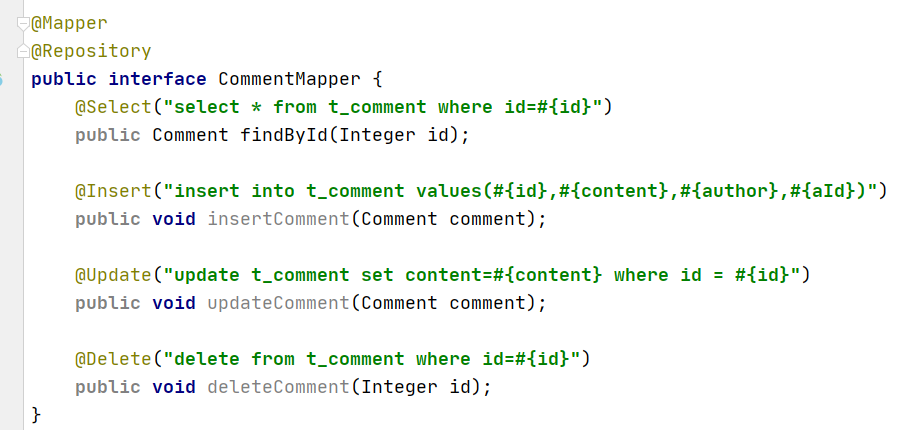
（7）修改数据源对应参数



**spring.datasource.type**=**com.alibaba.druid.pool.DruidDataSource  
spring.datasource.druid.initialSize**=**0  
spring.datasource.minIdle**=**10  
spring.datasource.maxActive**=**10**

二、使用注解方式整合MyBatis

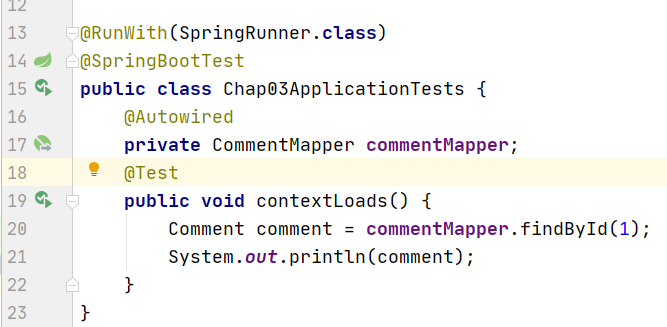
1.创建Mapper接口文件

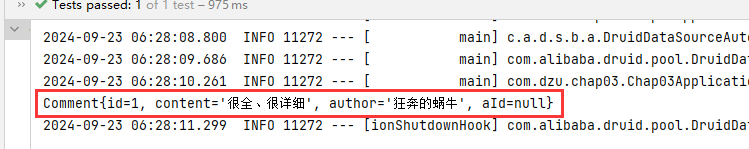


@Mapper  
@Repository  
**public interface** CommentMapper {  
 @Select(**"select \* from t\_comment where id=#{id}"**)  
 **public** Comment findById(Integer id);  
  
 @Insert(**"insert into t\_comment values(#{id},#{content},#{author},#{aId})"**)  
 **public void** insertComment(Comment comment);  
  
 @Update(**"update t\_comment set content=#{content} where id = #{id}"**)  
 **public void** updateComment(Comment comment);  
  
 @Delete(**"delete from t\_comment where id=#{id}"**)  
 **public void** deleteComment(Integer id);  
}

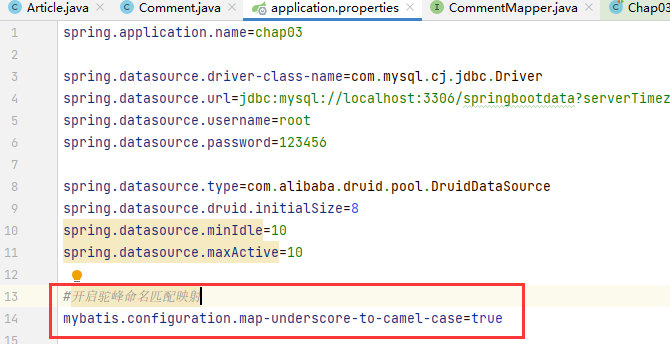
三、测试

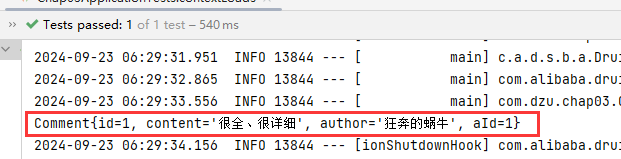
1．查询方法测试



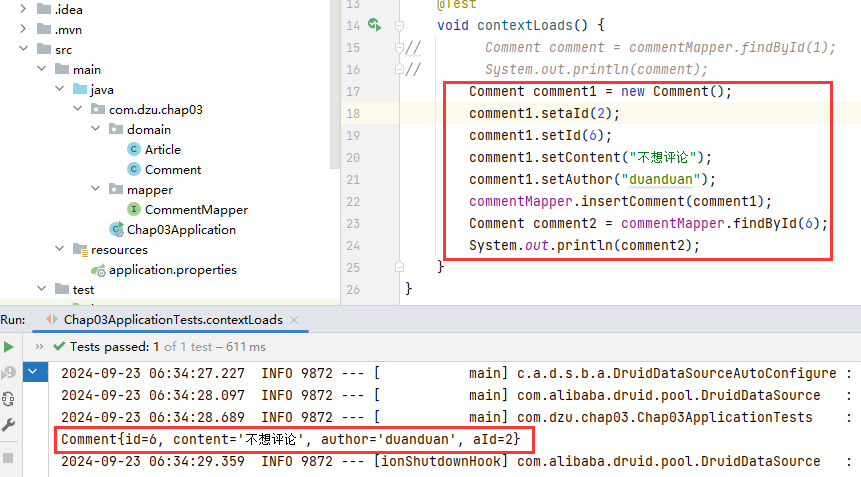


开启驼峰命名匹配映射





2.插入方法测试



3.修改方法测试

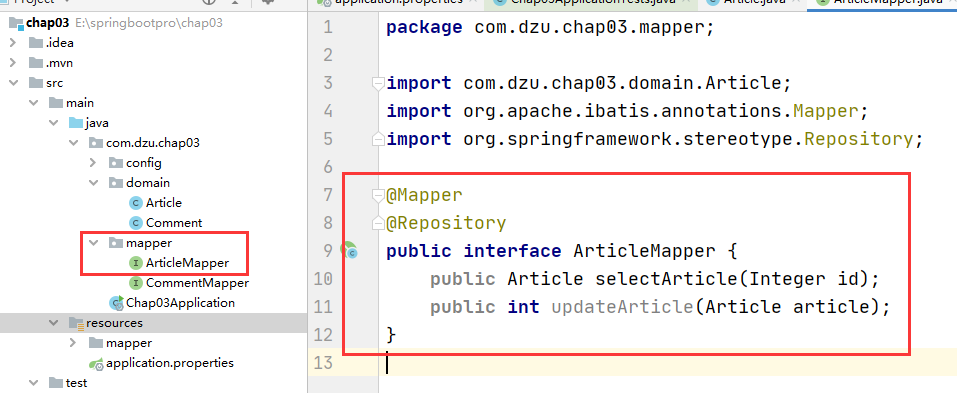


4.删除方法测试



四、使用配置文件方式整合MyBatis

1.创建ArticleMapper接口文件

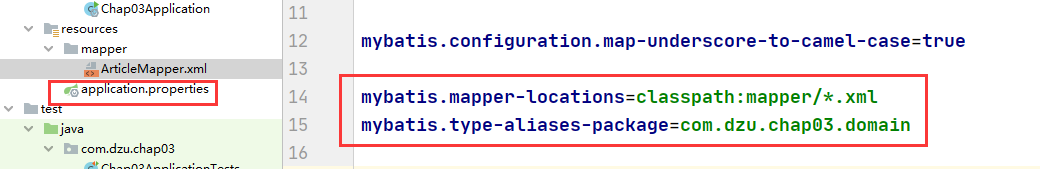


2.在resources中创建mapper文件夹，在mapper中创建ArticleMapper.xml

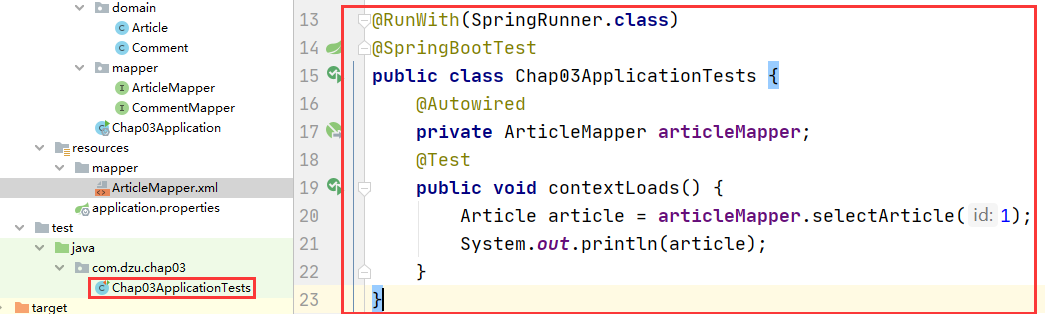


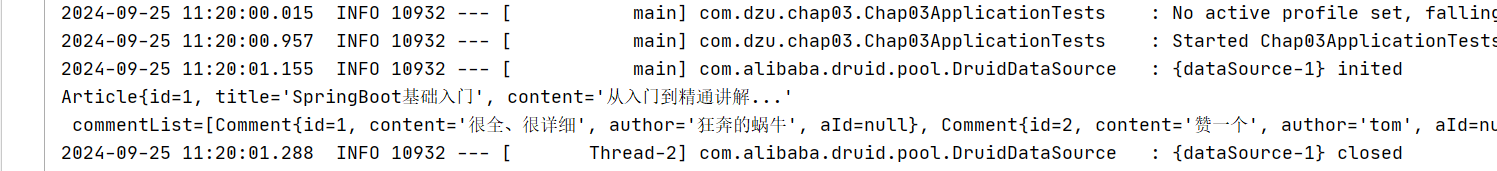
**<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>***<**mapper namespace="com.dzu.chap03.mapper.ArticleMapper"**>  
 *<!-- 1、查询文章详细（包括评论信息） -->  
 <?*xml version="1.0" encoding="UTF-8" *?>*<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>*<mapper namespace="com.example.dbtest.mapper.ArticleMapper">  
*<!-- 多表查询-->* <select id="selectArticle" resultMap="articleWithComment" parameterType="int">  
 select a.\*,c.id c\_id,c.content c\_content,c.author,c.a\_id  
 from t\_article a, t\_comment c  
 where a.id = c.a\_id and a.id=#{id};  
 </select>  
 <resultMap id="articleWithComment" type="Article">  
 <id property="id" column="id"></id>  
 <result property="title" column="title"></result>  
 <result property="content" column="content"></result>  
 <collection property="commentList" ofType="Comment">  
 <id property="id" column="c\_id"></id>  
 <result property="content" column="c\_content"></result>  
 <result property="author" column="author"></result>  
 </collection>  
 </resultMap>  
  
*<!--插入数据-->* <insert id="insertArticle" parameterType="Article">  
 insert into t\_article values (#{id},#{title},#{content})  
 </insert>  
  
*<!-- 单表查询-->* <select id="findArticle" resultType="Article" parameterType="int">  
 select *\** from t\_article where id = #{id}  
 </select>  
  
*<!-- 根据文章id更新文章信息->* <update id="updateArticle" parameterType="Article">  
 update t\_article  
 <set>  
 <if test="title !=null and title !=''">  
 title=#{title},  
 </if>  
 <if test="content != null and content!=''">  
 content = #{content}  
 </if>  
 </set>  
 where id=#{id}  
 </update>  
  
*<!-- 根据id删除文章数据-->* <delete id="deleteArticle" parameterType="int">  
 delete from t\_article where id= #{id}  
 </delete>  
</mapper>

3.在application.properties中配置XML映射文件路径



4.测试





@RunWith(SpringRunner.class)  
@SpringBootTest  
public class DbtestApplicationTests {  
 @Autowired  
 private CommentMapper commentMapper;  
  
 @Autowired  
 private ArticleMapper articleMapper;  
 @Test  
 public void contextLoads2(){  
 *//插入数据  
// Article article = new Article();  
// article.setId(3);  
// article.setTitle("hi");  
// article.setContent("hello");  
// int i = articleMapper.insertArticle(article);  
// System.out.println(i);  
 //单表查询  
// Article article1 = articleMapper.findArticle(3);  
// System.out.println(article1);  
  
 //修改数据  
// Article article = articleMapper.findArticle(3);  
// article.setTitle("早上好");  
// article.setContent("星期一的早上好呀");  
// int i = articleMapper.updateArticle(article);  
// System.out.println(i);  
// Article article1 = articleMapper.findArticle(3);  
// System.out.println(article1);  
  
 //删除数据* int i = articleMapper.deleteArticle(3);  
 System.*out*.println(i);  
 Article article = articleMapper.findArticle(3);  
 System.*out*.println(article);  
 }

}