# 实验七 Spring Boot整合JPA

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

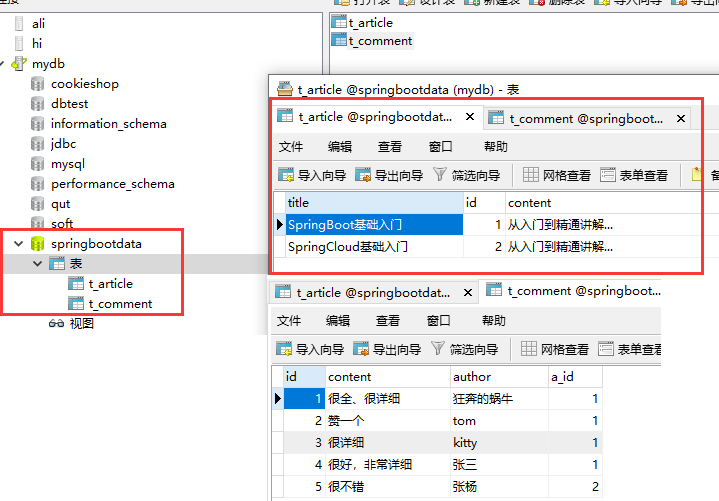
INSERT INTO `t\_comment` VALUES ('1', '很全、很详细', '狂奔的蜗牛', '1');

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

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

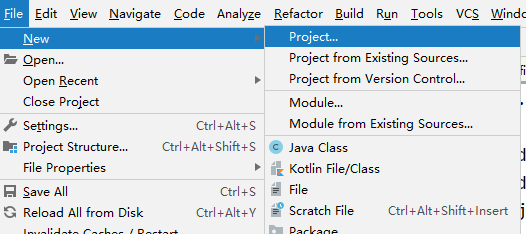
INSERT INTO `t\_comment` VALUES ('4', '很好，非常详细', '张三', '1');

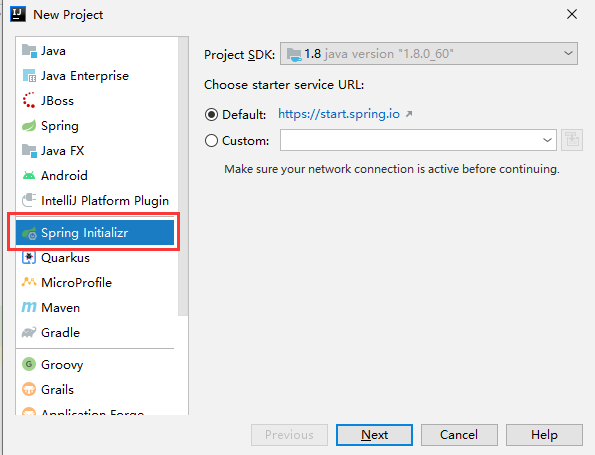
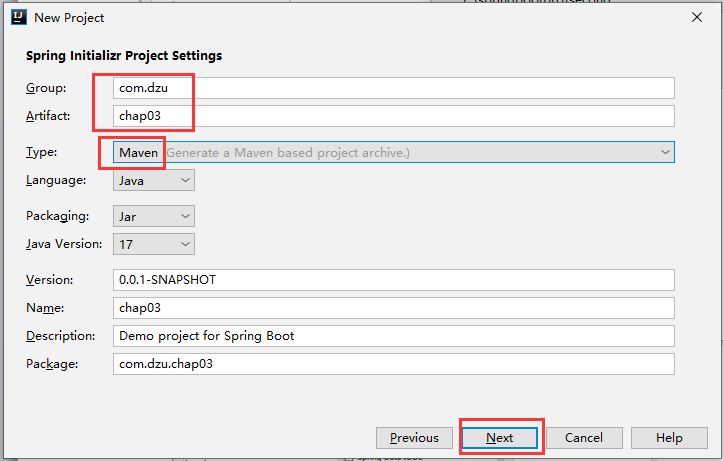
INSERT INTO `t\_comment` VALUES ('5', '很不错', '张杨', '2');



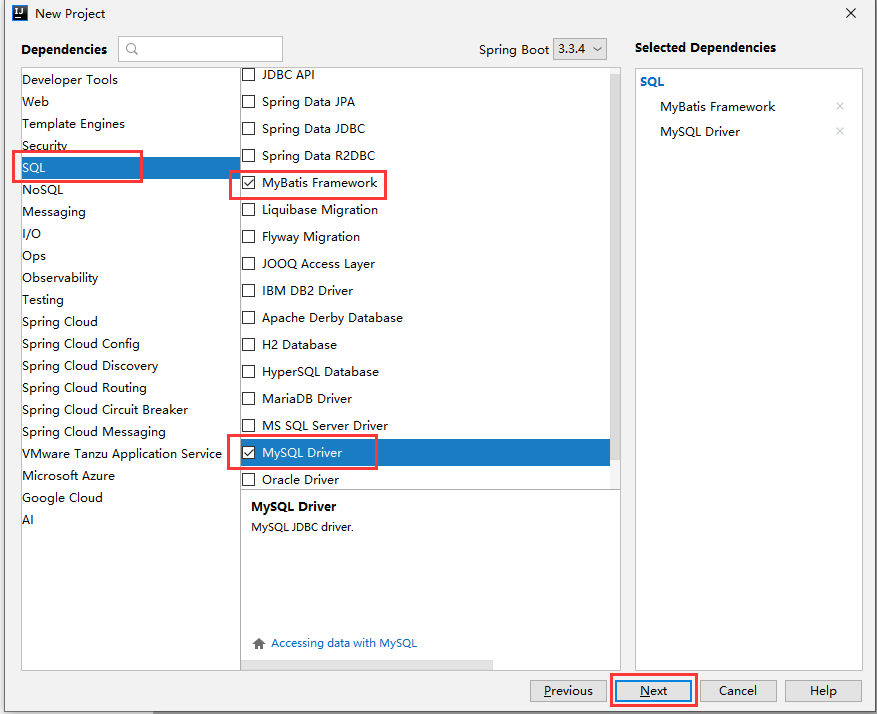
2.创建项目并引入相应的启动器

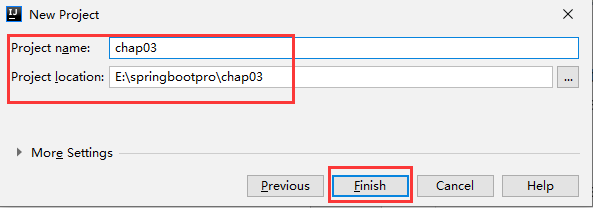
（1）使用Spring Initializr方式新建项目



（2）选择SQL依赖(下图中可以不选mybatis)





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



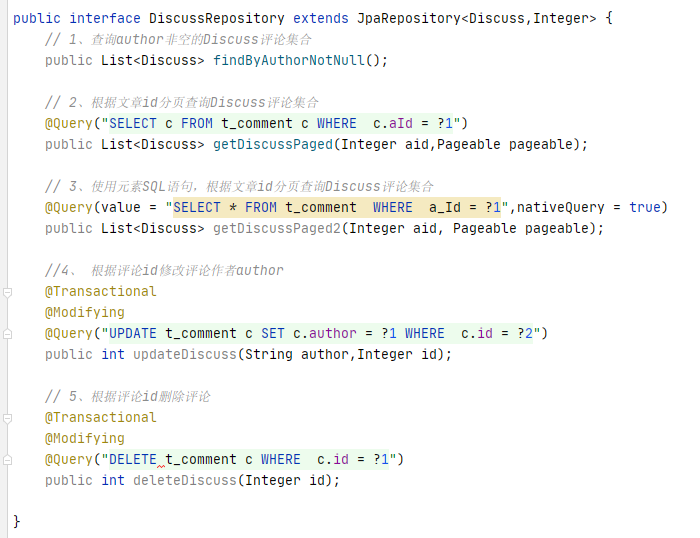
二、编写ORM实体类

声明实体类，添加注解，指明和哪个表进行映射，每一个属性和哪一个字段进行对应



package com.example.dbtest.domain;  
  
import javax.persistence.\*;  
  
@Entity(name = "t\_comment") *//ORM实体类*public class Discuss {  
 @Id *//属性与表中的主键进行对应* @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id;  
 @Column(name="content")  
 private String content;  
 private String author;  
 @Column(name = "a\_id")  
 private Integer aId;  
  
 public Integer getId() {  
 return id;  
 }  
  
 public void setId(Integer id) {  
 this.id = id;  
 }  
  
 public String getContent() {  
 return content;  
 }  
  
 public void setContent(String content) {  
 this.content = content;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(String author) {  
 this.author = author;  
 }  
  
 public Integer getaId() {  
 return aId;  
 }  
  
 public void setaId(Integer aId) {  
 this.aId = aId;  
 }  
  
 @Override  
 public String toString() {  
 return "Discuss{" +  
 "id=" + id +  
 ", content='" + content + '\'' +  
 ", author='" + author + '\'' +  
 ", aId=" + aId +  
 '}';  
 }  
}

三、编写符合JPA规范的Repository接口



四、测试

import com.example.dbtest.domain.Discuss;  
import com.example.dbtest.repository.DiscussRepository;  
import org.junit.Test;  
import org.junit.runner.RunWith;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.data.domain.Example;  
import org.springframework.data.domain.ExampleMatcher;  
import org.springframework.data.domain.PageRequest;  
import org.springframework.data.domain.Pageable;  
import org.springframework.test.context.junit4.SpringRunner;  
  
import java.util.List;  
import java.util.Optional;  
  
import static org.springframework.data.domain.ExampleMatcher.GenericPropertyMatchers.*startsWith*;  
  
@RunWith(SpringRunner.class)  
@SpringBootTest  
public class JpaTests {  
 @Autowired  
 private DiscussRepository repository;  
 *// 1、使用JpaRepository内部方法进行数据操作* @Test  
 public void selectComment() {  
 Optional<Discuss> optional = repository.findById(1);  
 if(optional.isPresent()){  
 System.*out*.println(optional.get());  
 }  
 System.*out*.println();  
 }  
 *// 2、使用方法名关键字进行数据操作* @Test  
 public void selectCommentByKeys() {  
 List<Discuss> list = repository.findByAuthorNotNull();  
 System.*out*.println(list);  
 }  
 *// 3、使用@Query注解进行数据操作* @Test  
 public void selectCommentPaged() {  
 Pageable pageable = PageRequest.*of*(0,3);  
 List<Discuss> allPaged = repository.getDiscussPaged(1, pageable);  
 System.*out*.println(allPaged);  
 }  
 *// 4、使用Example封装参数进行数据查询操作* @Test  
 public void selectCommentByExample() {  
 Discuss discuss =new Discuss();  
 discuss.setAuthor("张三");  
 Example<Discuss> example = Example.*of*(discuss);  
 List<Discuss> list = repository.findAll(example);  
 System.*out*.println(list);  
 }  
 @Test  
 public void selectCommentByExampleMatcher() {  
 Discuss discuss =new Discuss();  
 discuss.setAuthor("张");  
 ExampleMatcher matcher = ExampleMatcher.*matching*()  
 .withMatcher("author",*startsWith*());  
 Example<Discuss> example = Example.*of*(discuss, matcher);  
 List<Discuss> list = repository.findAll(example);  
 System.*out*.println(list);  
 }  
}