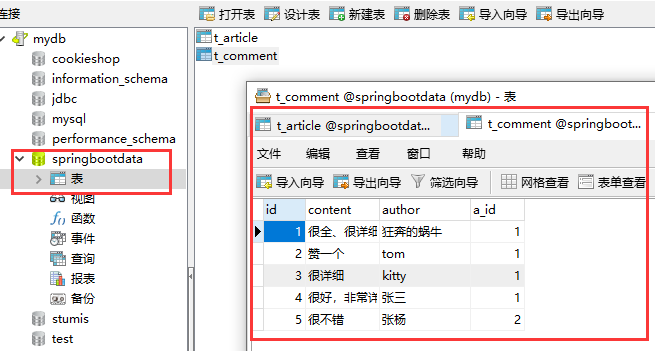
# 实验十三 SpringBoot默认缓存管理

一、创建数据库springbootdata



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');

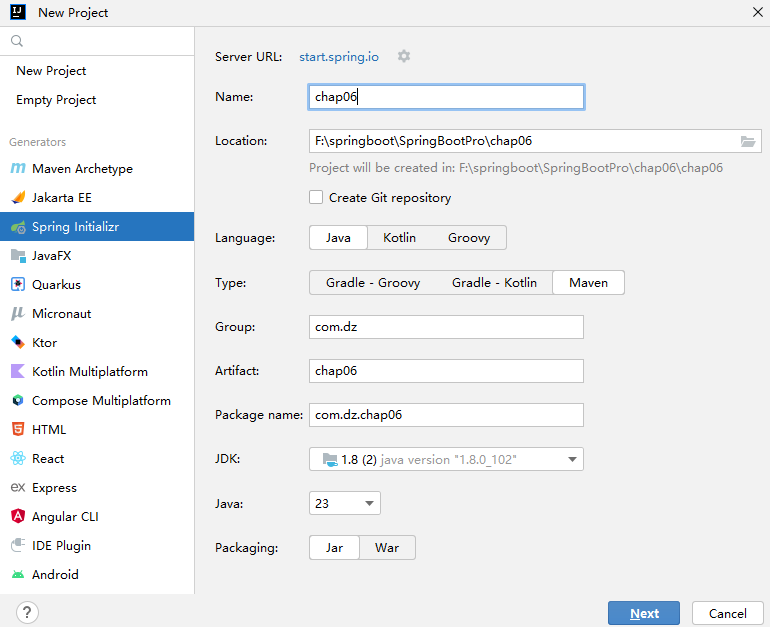
INSERT INTO `t\_comment` VALUES ('3', '很详细', 'kitty', '1');

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

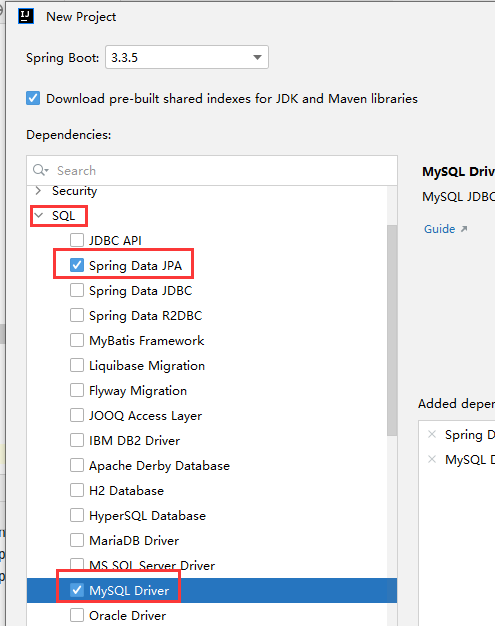
INSERT INTO `t\_comment` VALUES ('5', '很不错', '张杨', '2');

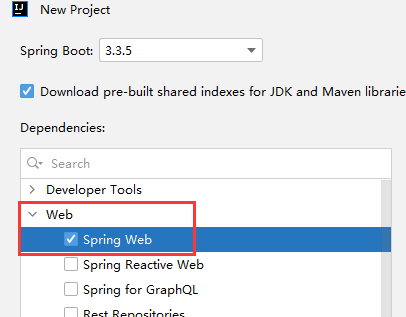
二、创建项目

1.创建项目



2.选择依赖

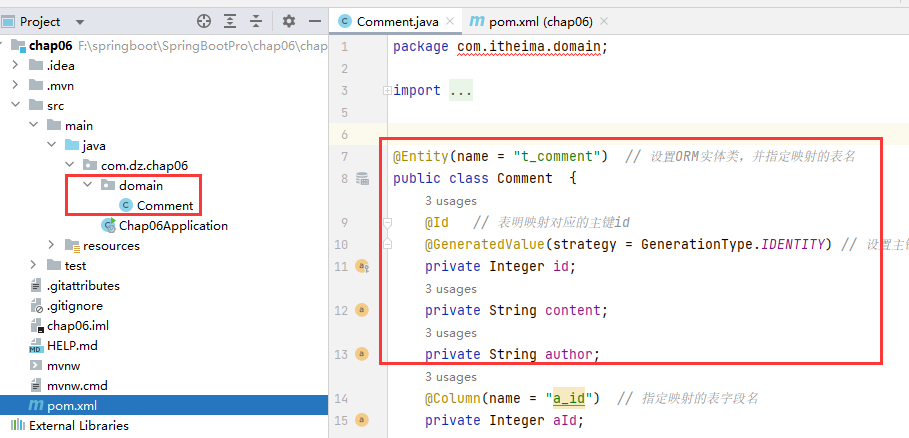




3.修改pom.xml中的父工程版本、Java版本、本地仓库位置

*<?*xml version="1.0" encoding="UTF-8"*?>*<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version> 2.2.2.RELEASE</version>  
 <relativePath/> *<!-- lookup parent from repository -->* </parent>  
 <groupId>com.itheima</groupId>  
 <artifactId>chapter01</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>chapter01</name>  
 <description>Demo project for Spring Boot</description>  
  
 <properties>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 *<!-- 选择的Web模块依赖启动器 -->* <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 *<!-- 测试类依赖 -->* <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 *<!-- 引入热部署依赖 -->* <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 *<!-- Maven打包工具插件 -->* <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

4．编写数据库表对应的实体类



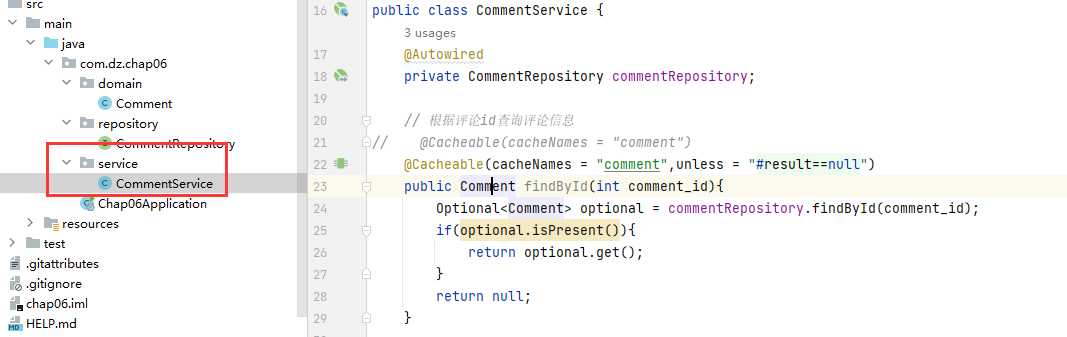
@Entity(name = "t\_comment") *// 设置ORM实体类，并指定映射的表名*public class Comment {  
 @Id *// 表明映射对应的主键id* @GeneratedValue(strategy = GenerationType.*IDENTITY*) *// 设置主键自增策略* private Integer id;  
 private String content;  
 private String author;  
 @Column(name = "a\_id") *// 指定映射的表字段名* private Integer aId;  
  
 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 "Comment{" +  
 "id=" + id +  
 ", content='" + content + '\'' +  
 ", author='" + author + '\'' +  
 ", aId=" + aId +  
 '}';  
 }  
}

5.编写数据库操作对应的Repository接口文件CommentRepository



public interface CommentRepository extends JpaRepository<Comment,Integer> {  
 *// 根据评论id修改评论作者评论作者author* @Transactional  
 @Modifying  
 @Query("UPDATE t\_comment c SET c.author= ?1 WHERE c.id = ?2")  
 public int updateComment(String author,Integer id);  
}

6.编写业务操作类CommenService



@Service  
public class CommentService {  
 @Autowired  
 private CommentRepository commentRepository;  
  
 *// 根据评论id查询评论信息* public Comment findById(int comment\_id){  
 Optional<Comment> optional = commentRepository.findById(comment\_id);  
 if(optional.isPresent()){  
 return optional.get();  
 }  
 return null;  
 }  
  
 *// 更新评论信息* public Comment updateComment(Comment comment){  
 commentRepository.updateComment(comment.getAuthor(), comment.getaId());  
 return comment;  
 }  
 *// 删除评论信息*  public void deleteComment(int comment\_id){  
 commentRepository.deleteById(comment\_id);  
 }  
}

7.编写CommentController

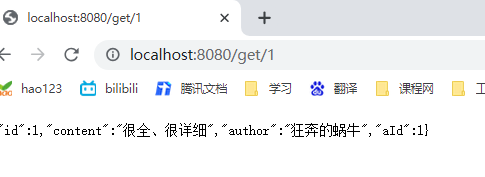


@RestController  
public class CommentController {  
 @Autowired  
 private CommentService commentService;  
  
 @GetMapping("/get/{id}")  
 public Comment findById(@PathVariable("id") int comment\_id){  
 Comment comment = commentService.findById(comment\_id);  
 return comment;  
 }  
  
 @GetMapping("/update/{id}/{author}")  
 public Comment updateComment(@PathVariable("id") int comment\_id,  
 @PathVariable("author") String author){  
 Comment comment = commentService.findById(comment\_id);  
 comment.setAuthor(author);  
 Comment updateComment = commentService.updateComment(comment);  
 return updateComment;  
 }  
  
 @GetMapping("/delete/{id}")  
 public void deleteComment(@PathVariable("id") int comment\_id){  
 commentService.deleteComment(comment\_id);  
 }  
}

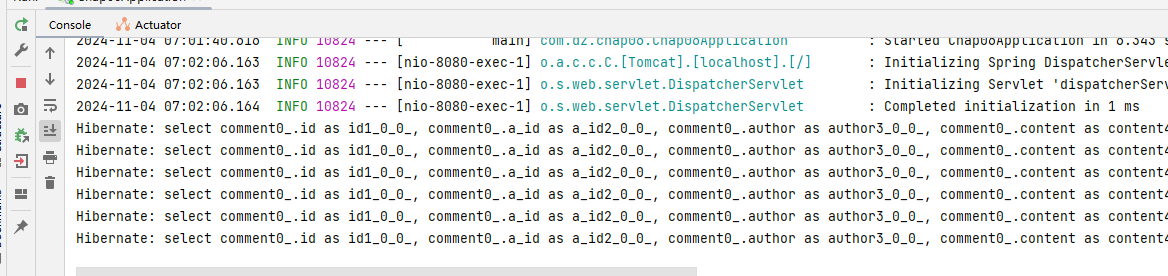
8.修改application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/springbootdata?serverTimezone=UTC&useSSL=false  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver  
spring.datasource.username=root  
spring.datasource.password=123456  
spring.jpa.show-sql=true

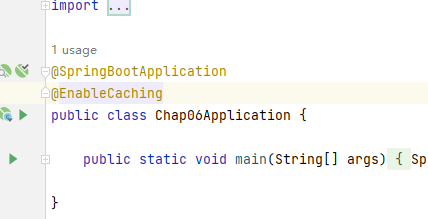
9.测试



刷新多次，查看控制台输出



10.添加默认缓存注解





重启项目，再次测试

