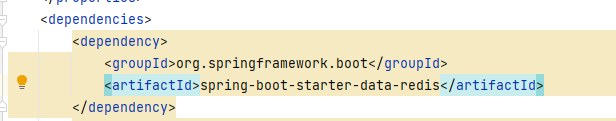
# 实验十四 SpringBoot整合Redis实现缓存管理

一、基于注解的Redis缓存实现

在实验十三项目基础上进行下列步骤

1.添加Spring Data Redis依赖启动器



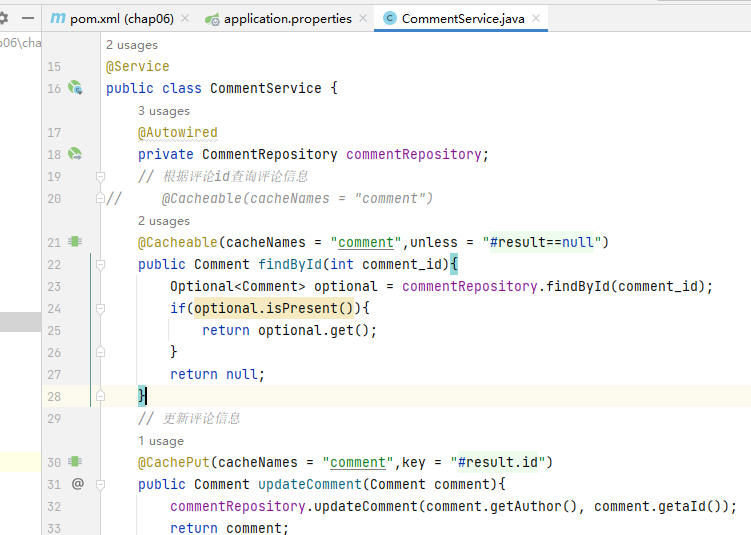
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-redis</artifactId>  
</dependency>

2.在application.properties中增加Redis服务连接配置



*# Redis服务地址*spring.redis.host=127.0.0.1  
*# Redis服务器连接端口*spring.redis.port=6379  
*# Redis服务器连接密码*spring.redis.password=

3.对CommentService类中的查询、修改、删除方法添加注解

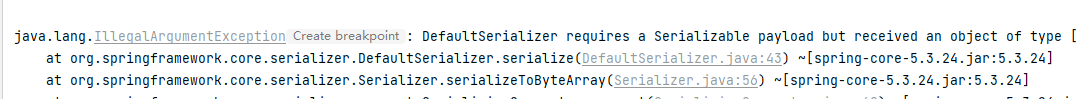


@Service  
public class CommentService {  
 @Autowired  
 private CommentRepository commentRepository;  
  
 *// 根据评论id查询评论信息* @Cacheable(cacheNames = "comment",unless = "#result==null")  
 public Comment findById(int comment\_id){  
 Optional<Comment> optional = commentRepository.findById(comment\_id);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

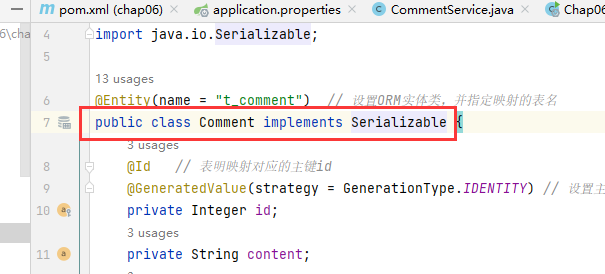
4.开启redis服务和redis管理器

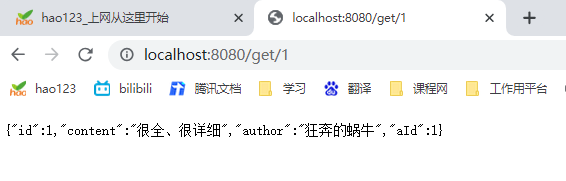
5.运行项目启动类，进行测试

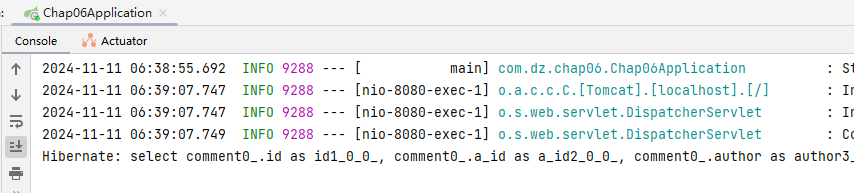
出现序列号错误



6.Comment类实现序列号接口Serializable，重新测试

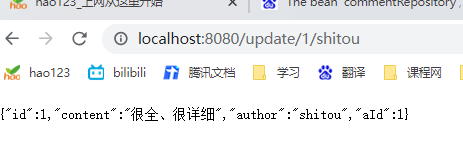


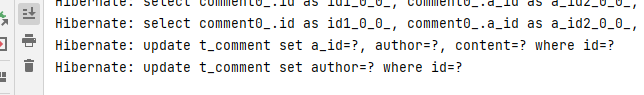




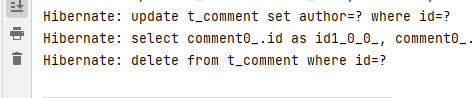
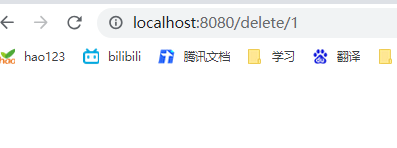
重复刷新网址，看是否实现了缓存管理

7.基于注解的Redis缓存更新测试



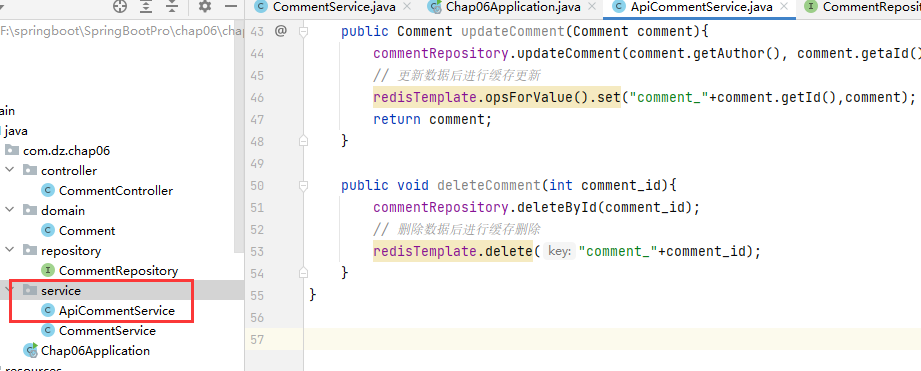


8.基于注解的Redis缓存删除测试



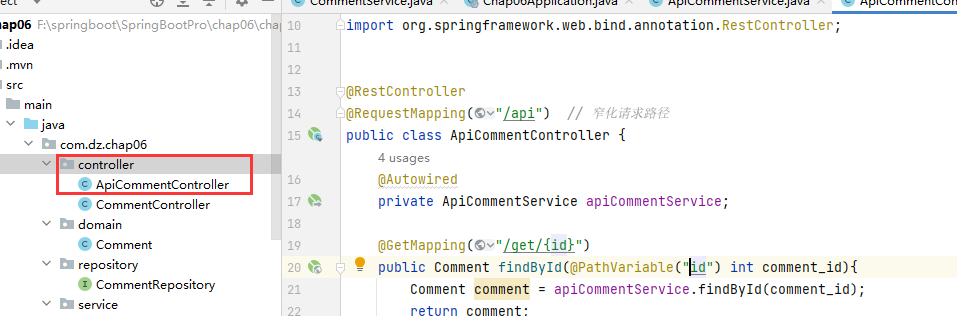
二、基于API的Redis缓存实现

1.创建ApiCommentService



@Service  
public class ApiCommentService {  
 @Autowired  
 private CommentRepository commentRepository;  
 @Autowired  
 private RedisTemplate redisTemplate;  
  
 public Comment findById(int comment\_id){  
 *// 先从Redis缓存中查询数据* Object object = redisTemplate.opsForValue().get("comment\_"+comment\_id);  
 if (object!=null){  
 return (Comment)object;  
 }else {  
 *// 缓存中没有，就进入数据库查询* Optional<Comment> optional = commentRepository.findById(comment\_id);  
 if(optional.isPresent()){  
 Comment comment= optional.get();  
 *// 将查询结果进行缓存，并设置有效期为1天* redisTemplate.opsForValue().set("comment\_"+comment\_id, comment,1, TimeUnit.*DAYS*);  
  
*// redisTemplate.opsForValue().set("comment\_"+comment\_id,comment);  
// redisTemplate.expire("comment\_"+comment\_id,90,TimeUnit.SECONDS);* return comment;  
 }else {  
 return null;  
 }  
 }  
 }  
  
 public Comment updateComment(Comment comment){  
 commentRepository.updateComment(comment.getAuthor(), comment.getaId());  
 *// 更新数据后进行缓存更新* redisTemplate.opsForValue().set("comment\_"+comment.getId(),comment);  
 return comment;  
 }  
  
 public void deleteComment(int comment\_id){  
 commentRepository.deleteById(comment\_id);  
 *// 删除数据后进行缓存删除* redisTemplate.delete("comment\_"+comment\_id);  
 }  
}

2.编写ApiCommentController



@RestController  
@RequestMapping("/api") *// 窄化请求路径*public class ApiCommentController {  
 @Autowired  
 private ApiCommentService apiCommentService;  
  
 @GetMapping("/get/{id}")  
 public Comment findById(@PathVariable("id") int comment\_id){  
 Comment comment = apiCommentService.findById(comment\_id);  
 return comment;  
 }  
  
 @GetMapping("/update/{id}/{author}")  
 public Comment updateComment(@PathVariable("id") int comment\_id,  
 @PathVariable("author") String author){  
 Comment comment = apiCommentService.findById(comment\_id);  
 comment.setAuthor(author);  
 Comment updateComment = apiCommentService.updateComment(comment);  
 return updateComment;  
 }  
  
 @GetMapping("/delete/{id}")  
 public void deleteComment(@PathVariable("id") int comment\_id){  
 apiCommentService.deleteComment(comment\_id);  
 }  
}

3.



4.测试

