HW5

在项目中增加 Redis 缓存

由于写操作作为单独的完整的事务，而且写操作写入缓存中的数据涉及到和数据库中持久化的数据的同步、合并、回滚等问题，因此对于 ORDER 等写入操作频繁，但是读取较少的数据不适合放入缓存。但用户对书本信息的访问频次很高，因此适合将图书信息放入缓存中。将书籍的持久化操作改写为通过缓存来执
行，包括对图书的增删改查。

以下代码已删除日志信息。

书本查询

用户进行书本信息的查询时，首先去查找 `Redis` 判断书本信息是否被缓存在 `Redis` 数据库中，若找到则直接返回书本信息。若未找到，则去持久化数据库（本项目使用的是 `MySQL`）中查找相关信息,并将书本信息放入缓存。

Java

```
@Override
public Book getBook(Integer id) {
    Object value = redisUtil.get("{bookId:}"+id);
    if (value != null){
        return JSONArray.parseObject(value.toString(), Book.class);
    } else {
        Book book = bookDao.findById(id);
        redisUtil.set("{bookId:}" + id, JSONArray.toJSON(book));
        return book;
    }
}
```

书本增加/更新

管理员进行书本的增加和更新时，考虑到用户会在进行读取时会先读到缓存中的信息，因此需要先修改缓存内容：首先判断目标书本是否在 `Redis` 缓存中，若存在则修改对应数据，之后在持久化数据库中进行修改，若不存在则直接在持久化数据库中增加/更新，并将书本信息缓存。

Java

```
@Override
@Transactional
public Boolean saveBook(Book book) {
    Object value = redisUtil.get("{bookId:}"+book.getId());
    if(value != null)  {
        log.info("the book of id: "+ book.getId() + "is in redis cache");
        redisUtil.set("{bookId:}" + book.getId(), JSONArray.toJSON(book));
    }
    // redis data should be updated if the book is a new one
    return bookDao.saveOne(book) > 0;
}
```

书本删除

和书本的增加或更新类似，首先判断目标书本是否在 `Redis` 缓存中，若存在则删除对应数据，之后在持久化数据库中删除，若不存在则直接在持久化数据库中删除。

Java

```
@Override
@Transactional
public Boolean deleteBookById(Integer id) {
    redisUtil.del("{bookId:}" + id);
    Boolean result = bookDao.deleteById(id);
```

```
        return result;
    }
```

人为关掉 Redis 的结果

关闭 Redis 服务后发送查询请求，得到相应如下：

```
: Reconnecting, last destination was localhost/127.0.0.1:6379
: Reconnecting, last destination was localhost/127.0.0.1:6379
: Cannot reconnect to [localhost:6379]: Connection refused: no further information: localhost/127.0.0.1:6379
: Cannot reconnect to [localhost:6379]: Connection refused: no further information: localhost/127.0.0.1:6379
: Reconnecting, last destination was localhost:6379
: Reconnecting, last destination was localhost:6379
: Cannot reconnect to [localhost:6379]: Connection refused: no further information: localhost/127.0.0.1:6379
: Cannot reconnect to [localhost:6379]: Connection refused: no further information: localhost/127.0.0.1:6379
: Reconnecting, last destination was localhost:6379
: Reconnecting, last destination was localhost:6379
: Cannot reconnect to [localhost:6379]: Connection refused: no further information: localhost/127.0.0.1:6379
: Cannot reconnect to [localhost:6379]: Connection refused: no further information: localhost/127.0.0.1:6379
: Reconnecting, last destination was localhost:6379
: Reconnecting, last destination was localhost:6379
```

可以看到 `Spring Boot` 一直在尝试重新建立和 `Redis` 的连接但是失败，发送的请求此时一直无法得到相应。

一段时间后抛出超时错误，请求失败。

```
d; nested exception is org.springframework.dao.QueryTimeoutException: Redis command timed out; n
```

```
{
    "timestamp": "2022-11-02T14:24:30.547+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "trace": "org.springframework.dao.QueryTimeoutException: Redis command timed out; nested except:
        \r\n\tat org.springframework.data.redis.connection.lettuce.LettuceExceptionConverter.convert
        LettuceExceptionConverter.convert(LettuceExceptionConverter.java:41)\r\n\tat org.springframe
        (PassThroughExceptionTranslationStrategy.java:44)\r\n\tat org.springframework.data.redis.Fa
        \r\n\tat org.springframework.data.redis.connection.lettuce.LettuceConnection.convertLettuce
        lettuce.LettuceConnection.await(LettuceConnection.java:1063)\r\n\tat org.springframework.dat
        \r\n\tat org.springframework.data.redis.connection.lettuce.LettuceInvoker$Synchronizer.invok
        LettuceInvoker.just(LettuceInvoker.java:94)\r\n\tat org.springframework.data.redis.connectic
        springframework.data.redis.connection.DefaultedRedisConnection.get(DefaultedRedisConnection.
        (DefaultValueOperations.java:58)\r\n\tat org.springframework.data.redis.core.AbstractOperat:
        springframework.data.redis.core.RedisTemplate.execute(RedisTemplate.java:223)\r\n\tat org.sp
        springframework.data.redis.core.AbstractOperations.execute(AbstractOperations.java:97)\r\n\t
        java:54)\r\n\tat com.wave.backend.utils.RedisUtil.get(RedisUtil.java:24)\r\n\tat com.wave.ba
```

对于运行过程中的日志的分析

- 第一次查找书籍信息

```
Book: 1 is not in Redis
Searching Book: 1 in DB
```

- 之后再查找相关书籍

```
the book of id: 1is in redis cache
```

这是由于第一次查找该书本时缓存中没有需要的数据（冷启动），需要去 `MySQL` 中取，并放入 `Redis` 中，之后再查找，直接通过 `Redis` 缓存即可取到书本信息。

- 关闭 `Redis` 后，由于 RedisConnection 一直无法获取, 因此报错超时。