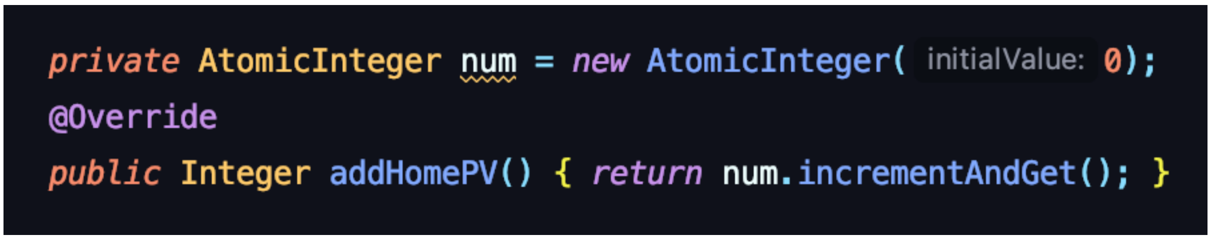
hw3: Multithread & Caching

Multithread

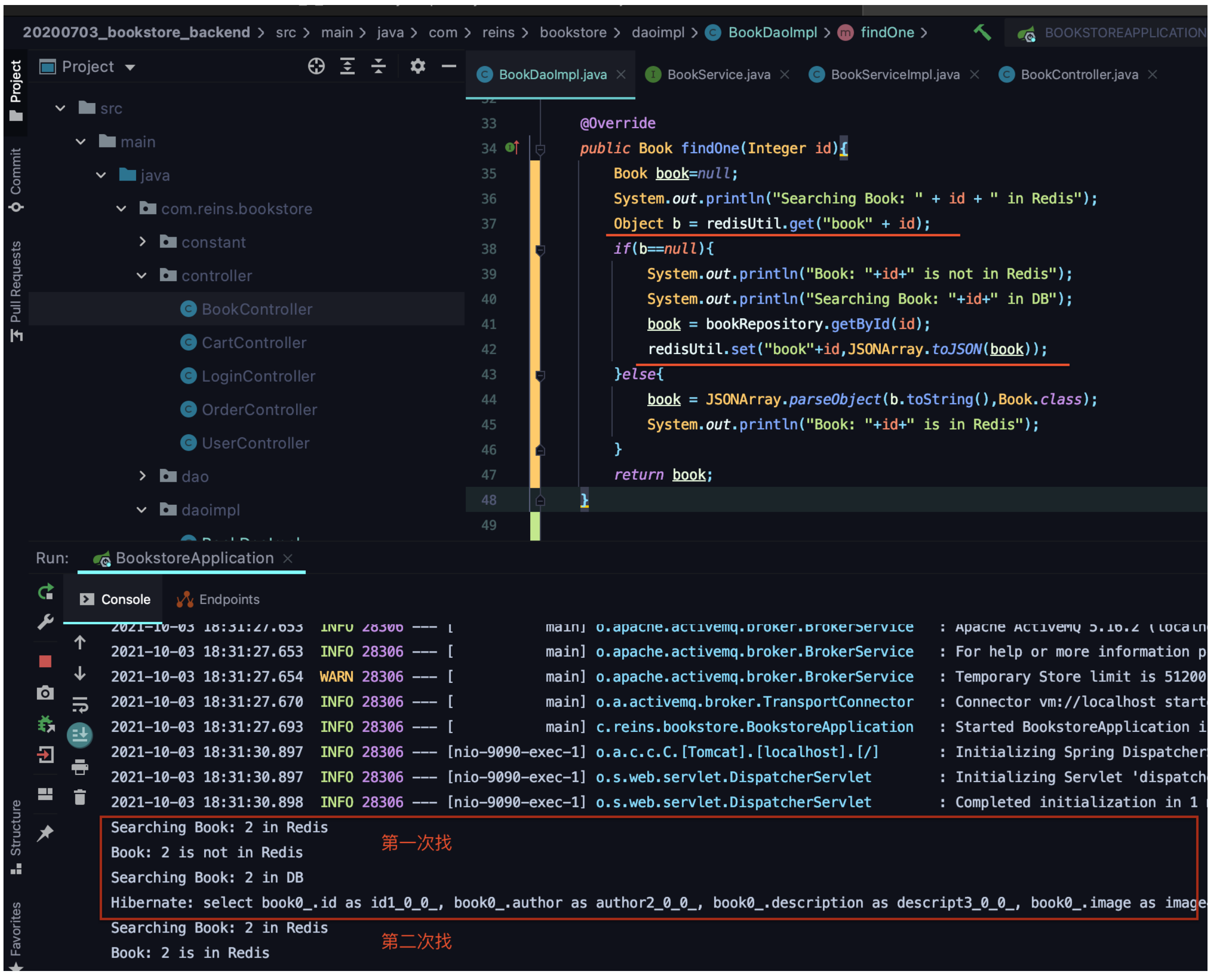


```java
private AtomicInteger num = new AtomicInteger( initialValue: 0);
@Override
public Integer addHomePV() { return num.incrementAndGet(); }
```

使用AtomicInteger保证原子性

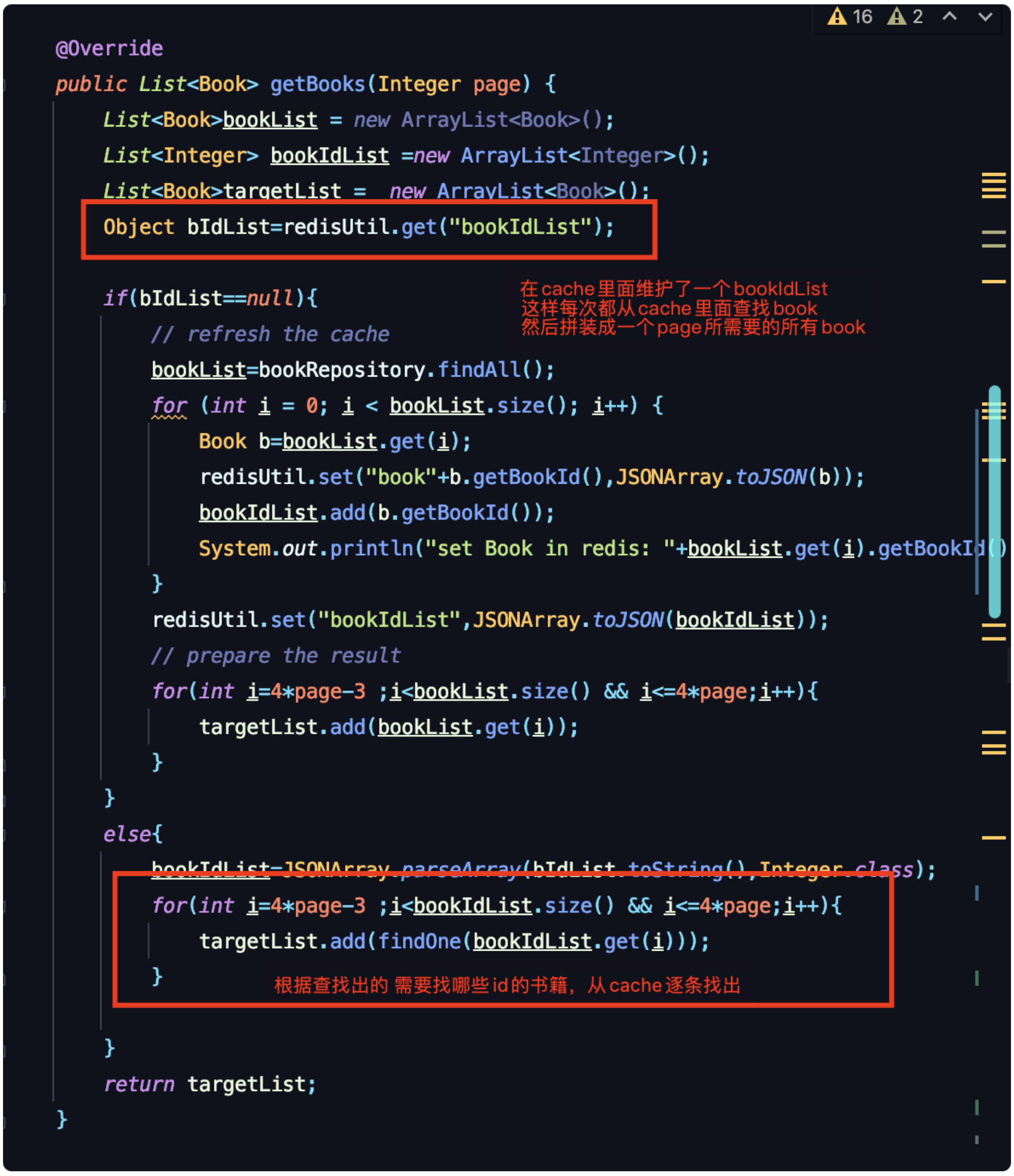
Caching

查找书籍



```java
    @Override
    public Book findOne(Integer id){
        Book book=null;
        System.out.println("Searching Book: " + id + " in Redis");
        Object b = redisUtil.get("book" + id);
        if(b==null){
            System.out.println("Book: "+id+" is not in Redis");
            System.out.println("Searching Book: "+id+" in DB");
            book = bookRepository.getById(id);
            redisUtil.set("book"+id,JSONArray.toJSON(book));
        }else{
            book = JSONArray.parseObject(b.toString(),Book.class);
            System.out.println("Book: "+id+" is in Redis");
        }
        return book;
    }
```

```
2021-10-03 18:31:27.653  INFO 28306 --- [           main] o.apache.activemq.broker.BrokerService   : Apache ActiveMQ 5.16.2 (localh
2021-10-03 18:31:27.653  INFO 28306 --- [           main] o.apache.activemq.broker.BrokerService   : For help or more information p
2021-10-03 18:31:27.654  WARN 28306 --- [           main] o.apache.activemq.broker.BrokerService   : Temporary Store limit is 51200
2021-10-03 18:31:27.670  INFO 28306 --- [           main] o.a.activemq.broker.TransportConnector   : Connector vm://localhost start
2021-10-03 18:31:27.693  INFO 28306 --- [           main] c.reins.bookstore.BookstoreApplication   : Started BookstoreApplication i
2021-10-03 18:31:30.897  INFO 28306 --- [nio-9090-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring Dispatcher
2021-10-03 18:31:30.897  INFO 28306 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatch
2021-10-03 18:31:30.898  INFO 28306 --- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1
Searching Book: 2 in Redis
Book: 2 is not in Redis
Searching Book: 2 in DB
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.description as descript3_0_0_, book0_.image as image
Searching Book: 2 in Redis
Book: 2 is in Redis
```



```java
@Override
public List<Book> getBooks(Integer page) {
    List<Book>bookList = new ArrayList<Book>();
    List<Integer> bookIdList =new ArrayList<Integer>();
    List<Book>targetList =  new ArrayList<Book>();
    Object bIdList=redisUtil.get("bookIdList");

    if(bIdList==null){
        // refresh the cache
        bookList=bookRepository.findAll();
        for (int i = 0; i < bookList.size(); i++) {
            Book b=bookList.get(i);
            redisUtil.set("book"+b.getBookId(),JSONArray.toJSON(b));
            bookIdList.add(b.getBookId());
            System.out.println("set Book in redis: "+bookList.get(i).getBookId()
        }
        redisUtil.set("bookIdList",JSONArray.toJSON(bookIdList));
        // prepare the result
        for(int i=4*page-3 ;i<bookList.size() && i<=4*page;i++){
            targetList.add(bookList.get(i));
        }
    }
    else{
        bookIdList=JSONArray.parseArray(bIdList.toString(),Integer.class);
        for(int i=4*page-3 ;i<bookIdList.size() && i<=4*page;i++){
            targetList.add(findOne(bookIdList.get(i)));
        }

    }
    return targetList;
}
```

在cache里面维护了一个bookIdList
这样每次都从cache里面查找book
然后拼装成一个page所需要的所有book

根据查找出的 需要找哪些id的书籍，从cache逐条找出

修改书籍信息

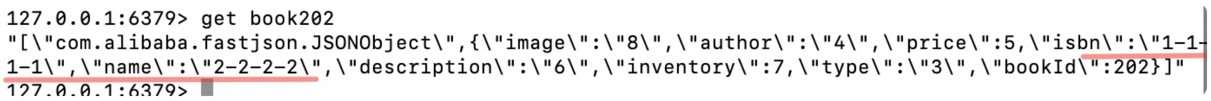
先写cache,再写数据库，set方法可以覆盖原来redis的键值对

原来的cache

```
127.0.0.1:6379> get book202
"[\"com.alibaba.fastjson.JSONObject\",{\"image\":\"8\",\"author\":\"4\",\"price\":5,\"isbn\":\"1\",
\"name\":\"2\",\"description\":\"6\",\"inventory\":7,\"type\":\"3\",\"bookId\":202}]"
```

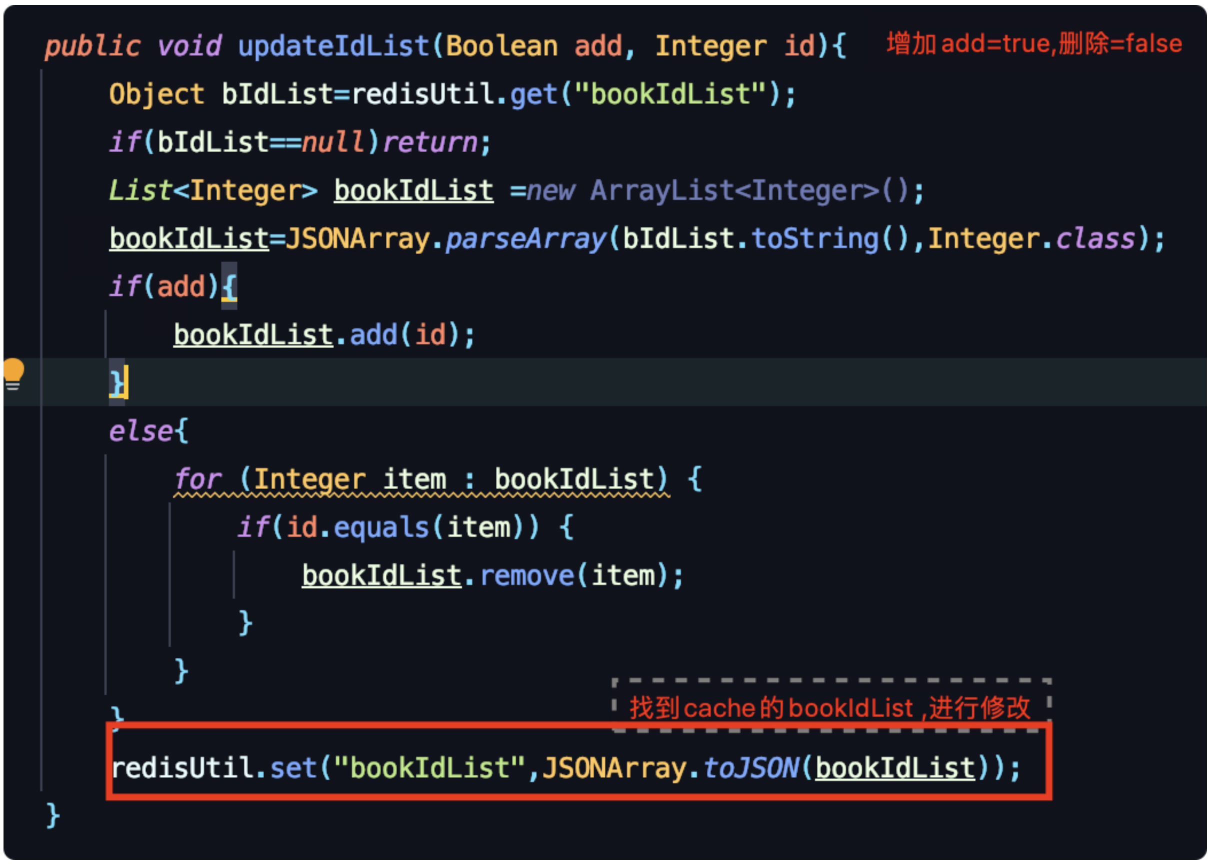
```
http://localhost:9090/changeBook?isbn=1-1-1-1&name=2-2-2-2&type=3&author=4&price=5&description=6&inventory=7&image=8&id=202
```

修改后的cache



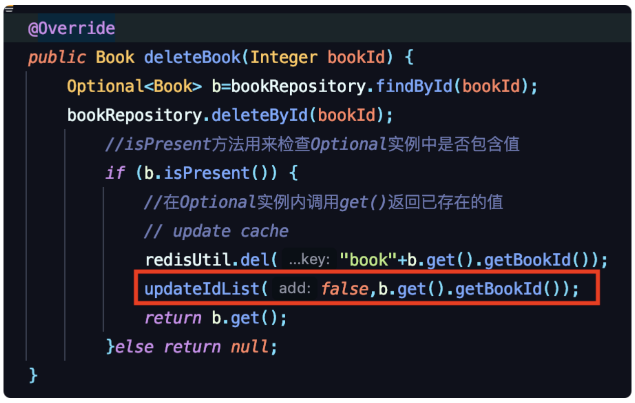
```
127.0.0.1:6379> get book202
"[\"com.alibaba.fastjson.JSONObject\",{\"image\":\"8\",\"author\":\"4\",\"price\":5,\"isbn\":\"1-1-
1-1\",\"name\":\"2-2-2-2\",\"description\":\"6\",\"inventory\":7,\"type\":\"3\",\"bookId\":202}]"
127.0.0.1:6379>
```

对于查找删除操作更新cache中的bookidList,然后再删除增加对应的cache记录

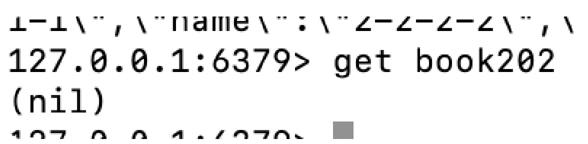


```java
public void updateIdList(Boolean add, Integer id){
    Object bIdList=redisUtil.get("bookIdList");
    if(bIdList==null)return;
    List<Integer> bookIdList =new ArrayList<Integer>();
    bookIdList=JSONArray.parseArray(bIdList.toString(),Integer.class);
    if(add){
        bookIdList.add(id);
    }
    else{
        for (Integer item : bookIdList) {
            if(id.equals(item)) {
                bookIdList.remove(item);
            }
        }
    }
    redisUtil.set("bookIdList",JSONArray.toJSON(bookIdList));
}
```

删除

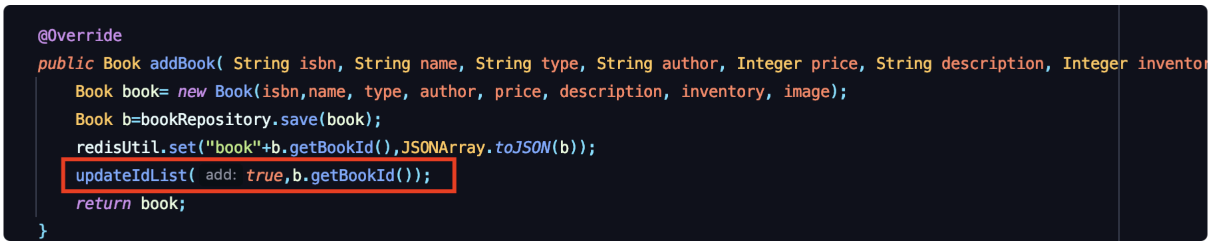


```java
@Override
public Book deleteBook(Integer bookId) {
    Optional<Book> b=bookRepository.findById(bookId);
    bookRepository.deleteById(bookId);
        //isPresent方法用来检查Optional实例中是否包含值
        if (b.isPresent()) {
            //在Optional实例内调用get()返回已存在的值
            // update cache
            redisUtil.del( ...key: "book"+b.get().getBookId());
            updateIdList( add: false,b.get().getBookId());
            return b.get();
        }else return null;
}
```



```
1-1\",\"name\":\"2-2-2-2\",\
127.0.0.1:6379> get book202
(nil)
127.0.0.1:6379>
```

增加



```
@Override
public Book addBook( String isbn, String name, String type, String author, Integer price, String description, Integer inventor
    Book book= new Book(isbn,name, type, author, price, description, inventory, image);
    Book b=bookRepository.save(book);
    redisUtil.set("book"+b.getBookId(),JSONArray.toJSON(b));
    updateIdList( add: true,b.getBookId());
    return book;
}
```