

第6章 Spring Boot整合缓存

《Spring Boot企业级开发教程（第2版）》



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

学习目标/Target



了解Spring Boot默认缓存方案，能够说出默认缓存方案的执行流程

掌握声明式缓存注解，能够说出@EnableCaching、@Cacheable、@CachePut、@CacheEvict、@Caching、@CacheConfig注解及常用属性的作用

掌握声明式缓存注解的应用，能够在Spring Boot项目中正确应用声明式缓存注解

学习目标/Target



了解Ehcache概述，能够说出Ehcache的特点

掌握整合Ehcache，能够在Spring Boot项目中整合Ehcache，并正确应用声明式缓存注解

掌握SpringBoot整合Redis缓存，能够在Spring Boot项目中整合Redis缓存，并正确应用声明式缓存注解

章节概述/ Summary



企业级应用为了避免读取数据时受限于数据库的访问效率而导致整体系统性能偏低，通常会在应用程序与数据库之间建立一种临时的数据存储机制，该临时存储数据的区域称为缓存。缓存是一种介于数据永久存储介质与应用程序之间的数据临时存储介质，可以提供临时的数据存储空间，合理使用缓存可以有效减少低速数据读取（例如磁盘IO）过程的次数，以提高系统性能。Spring Boot提供了几乎所有主流缓存技术的整合方案。下面将对Spring Boot整合常见的缓存技术进行讲解。



6.1

Spring Boot默认缓存管理

6.2

Spring Boot整合Ehcache缓存

6.3

SpringBoot整合Redis缓存



6.1

Spring Boot默认缓存管理



Spring框架支持透明地向应用程序添加缓存，以及对缓存进行管理，其管理缓存的核心是将缓存应用于操作数据的方法，从而减少操作数据的执行次数，同时不会对程序本身造成任何干扰。Spring Boot继承了Spring框架的缓存管理功能，下面将对Spring Boot内置的缓存方案进行讲解。

>>> 6.1.1 Spring Boot默认缓存方案



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌



了解Spring Boot默认缓存方案，能够
说出默认缓存方案的执行流程

6.1.1 Spring Boot默认缓存方案



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

Spring的缓存机制将提供的缓存作用于Java 方法上，基于缓存中的可用信息，可以减少方法的执行次数。每次目标方法调用时，抽象使用缓存行为来检查执行方法，即检查执行方法是否给定了缓存的执行参数，如果是，则返回缓存结果，不执行具体方法；如果否，则执行方法，并将结果缓存后，返回给用户。

Spring的默认的缓存方案通过org.springframework.cache.Cache和org.springframework.cache.CacheManager接口来统一不同的缓存技术。

Cache接口：缓存的组件定义规范，包含缓存的各种操作集合。Spring中为Cache接口提供了各种缓存的实现：RedisCache, EhCache, ConcurrentMapCache等

CacheManager接口：缓存管理器，基于缓存名称对缓存进行管理，并制定了管理Cache的规则。

6.1.1 Spring Boot默认缓存方案



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

在项目中添加某个缓存管理组件（如Redis）后，Spring Boot项目会选择并启用对应的缓存管理器。如果项目中同时添加了多个缓存组件，且没有定义类型为CacheManager的Bean组件或者名为cacheResolver的缓存解析器，Spring Boot将尝试按以下列表的顺序查找有效的缓存组件进行缓存管理。

- (1) Generic
- (2) JCache (EhCache 3、Hazelcast、Infinispan等)
- (3) EhCache 2.x
- (4) Hazelcast
- (5) Infinispan
- (6) Couchbase
- (7) Redis
- (8) Caffeine
- (9) Simple



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌



掌握声明式缓存注解，能够说出
@EnableCaching、@Cacheable、
@CachePut、@CacheEvict、
@Caching、@CacheConfig注解及常
用属性的作用



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

要想使用Spring提供的默认缓存，需要对缓存进行声明，也就是标志缓存的方法及缓存策略。对于缓存声明，Spring提供了一系列的注解，使用这些注解可以实现Spring 默认的基于注解的缓存管理。

1. @EnableCaching注解

@EnableCaching是Spring框架提供的用于开启基于注解的缓存支持的注解，当配置类上使用@EnableCaching注解，会默认提供CacheManager的实现，并通过AOP将缓存行为添加到应用程序。执行操作时，会检查是否已经存在注解对应的缓存。如果找到了，就会自动创建一个代理拦截方法调用，使用缓存的Bean执行处理。



2. @Cacheable注解

@Cacheable注解用于标注可缓存的方法，通常标注的方法为数据查询方法。标注@Cacheable注解的方法在执行时，会先查询缓存，如果查询到的缓存为空，则执行该方法，并将方法的执行结果添加到缓存；如果查询到缓存数据，则不执行该方法，而是直接使用缓存数据。

2. @Cacheable注解

@Cacheable注解提供了多个属性，用于对缓存进行相关配置。

@Cacheable注解属性及说明

属性名	说明
value/cacheNames	指定缓存的名称，必备属性，这两个属性二选一使用
key	指定缓存数据的key，默认使用方法参数值，可以使用SpEL表达式
keyGenerator	指定缓存数据的key的生成器，与key属性二选一使用
cacheManager	指定缓存管理器
cacheResolver	指定缓存解析器，与cacheManager属性二选一使用
condition	指定在符合某条件下进行数据缓存
unless	指定在符合某条件下不进行数据缓存
sync	指定是否使用异步缓存，默认为false



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

2. @Cacheable注解

(1) value/cacheNames属性

value和cacheNames属性作用相同，用于指定缓存的名称，方法的返回结果会存放在指定名称的缓存中。这两个属于必备选项，且要二选一使用。如果@Cacheable注解只配置value或者cacheNames属性，那么属性名可以省略。

```
@Cacheable("book")
public Book findById(Integer id){
    return bookDao.findById(id).get();
}
```



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

2. @Cacheable注解

(1) value/cacheNames属性

@Cacheable注解中可以指定多个缓存的名称，以便使用多个缓存。

```
@Cacheable({"book","hotBook"})
public Book findById(Integer id){
    return bookDao.findById(id).get();
}
```




6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

2. @Cacheable注解

(2) key属性

缓存的本质是键值对存储，key用于指定唯一的标识，value用于指定缓存的数据，所以每次调用缓存方法都会转换为访问缓存的键。缓存的键通过key属性进行指定，进行数据缓存时，如果没有指定key属性，Spring Boot默认配置类SimpleKeyGenerator中的generateKey(Object... params)方法会根据方法参数生成key值。对于没有参数的方法，其key是默认创建的空参SimpleKey[]对象；对于一个参数的方法，其key是默认参数值；对于有多个参数的方法，其key是包含所有参数的SimpleKey对象。



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

2. @Cacheable注解

(2) key属性

如果方法有多个参数，但是部分参数对缓存没有任何用处，通常会选择手动指定key属性的值，key属性的值可以通过SpEL表达式选择所需要的参数。

```
@Cacheable(cacheNames="book", key="#id")
public Book findBookById(Integer id, boolean includeUsed){
    return bookDao.findById(id).get();
}
```

2. @Cacheable注解

Cache缓存支持的SpEL表达式及说明

参数名	位置	描述	示例
methodName	root对象	当前被调用的方法名	#root.methodName
method	root对象	当前被调用的方法	#root.method.name
target	root对象	当前被调用的目标对象实例	#root.target
targetClass	root对象	当前被调用的目标对象的类	#root.targetClass
args	root对象	当前被调用的方法的参数列表	#root.args[0]
caches	root对象	当前被调用的方法的缓存列表	#root.caches[0].name
argumentName	执行上下文	当前被调用的方法参数，可以用#参数名或者#a0、#p0的形式（0代表参数索引，从0开始）	#comment_id、#a0、#p0
result	执行上下文	当前方法执行后的返回结果	#result



2. @Cacheable注解

(3) keyGenerator属性

keyGenerator属性与key属性本质作用相同，都是用于指定缓存数据的key，只不过keyGenerator属性指定的不是具体的key值，而是key值的生成器规则，由其中指定的生成器生成具体的key。使用时，keyGenerator属性与key属性要二者选一。关于自定义key值生成器的定义，可以参考Spring Boot默认配置类SimpleKeyGenerator的定义方式，这里不再做具体说明。



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

2. @Cacheable注解

(4) cacheManager/cacheResolver属性

cacheManager和cacheResolver属性分别用于指定缓存管理器和缓存解析器，这两个属性也是二选一使用，默认情况不需要配置，对于需要使用多个缓存管理器（如Redis、Ehcache等）的应用，可以为每个操作设置一个缓存管理器或缓存解析器。



2. @Cacheable注解

(5) condition属性

condition属性用于对数据进行有条件的选择性存储，只有当指定条件为true时才会对查询结果进行缓存，可以使用SpEL表达式指定属性值。

```
@Cacheable(cacheNames="book", condition="#id > 1")  
public Book findBook(Integer id){  
    return bookDao.findById(id).get();  
}
```



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

2. @Cacheable注解

(6) unless属性

unless属性的作用与condition属性相反，当指定的条件为true时，方法的返回值不会被缓存，也可以使用SpEL表达式指定。

```
@Cacheable(cacheNames="book", unless = "#result==null")
public Book findBook(Integer id){
    return bookDao.findById(id).get();
}
```



2. @Cacheable注解

(7) sync属性

在多线程程序中，某些操作可能会同时引用相同的参数，导致相同的对象被计算好几次，从而达不到缓存的目的。对于这种情况，可以使用sync属性，sync属性表示数据缓存过程中是否使用同步模式，默认值为false，通常不会使用该属性。



3. @CachePut注解

@CachePut注解的作用是更新缓存数据，当需要更新缓存且不影响方法执行时，可以使用@CachePut注解，通常用在数据更新方法上。@CachePut注解的执行顺序是，先进行方法调用，然后将方法结果更新到缓存中。

@CachePut注解也提供了多个属性，这些属性与@Cacheable注解的属性完全相同。通常不建议在同一个方法同时使用@CachePut和@Cacheable注解，这两个注解关注不同的行为，@CachePut注解会强制执行方法并进行缓存更新，使用@Cacheable注解时，如果请求能够在缓存中获取到对应的数据，就不会执行当前被@Cacheable注解标注的方法。



4. @CacheEvict注解

@CacheEvict注解的作用删除缓存中的数据，通常标注在数据删除方法上。@CacheEvict注解的默认执行顺序是先进行方法调用，然后将缓存清除。

@CacheEvict注解也提供了多个属性，这些属性与@Cacheable注解的属性基本相同，除此之外，还额外提供了两个特殊属性allEntries和beforeInvocation，下面对这两个属性分别进行讲解。



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

4. @CacheEvict注解

(1) allEntries属性

allEntries属性表示是否清除指定缓存空间中的所有缓存数据，默认值为false，即默认只删除指定key对应的缓存数据。

```
@CacheEvict(cacheNames = "book",allEntries = true)
public void delById(Integer id){
    bookDao.deleteById(id);
}
```



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

4. @CacheEvict注解

(2) beforeInvocation属性

beforeInvocation属性表示是否在方法执行之前进行缓存清除，默认值为false，即默认在执行方法后再进行缓存清除。

```
@CacheEvict(cacheNames = "book",beforeInvocation = true)
public void delById(Integer id){
    bookDao.deleteById(id);
}
```



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

5. @Caching注解

如果不同缓存之间的条件或者键表达式不同，就需要指定相同类型的多个注解，例如需要同时指定多个@CacheEvict或@CachePut，这个时候可以使用@Caching注解。@Caching注解用于针对复杂规则的数据缓存管理，@Caching注解中允许使用多个嵌套的@Cacheable、@CachePut或@CacheEvict。在@Caching注解内部包含有Cacheable、put和evict三个属性，分别对应于@Cacheable、@CachePut和@CacheEvict三个注解。

```
@Caching(evict = { @CacheEvict("primary"),
@CacheEvict(cacheNames="secondary", key="#date")})
public void delById(Integer id, Date date){
    bookDao.deleteById(id);
}
```



6.1.2 声明式缓存注解



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

6. @CacheConfig注解

@CacheConfig注解使用在类上，主要用于统筹管理类中所有使用@Cacheable、@CachePut和@CacheEvict标注方法中的公共属性。

```
@CacheConfig(cacheNames = "book")
@Service
public class BookService {
    @Autowired
    private BookRepository bookRepository;
    @Cacheable
    public Book findById(Integer id){
        return bookRepository.findById(id).get();
    }
}
```



6.1.3 声明式缓存注解的应用



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌



掌握声明式缓存注解的应用，能够在Spring Boot项目中正确应用声明式缓存注解



下面在Spring Boot项目中进一步对[声明式缓存注解的应用](#)进行讲解。

1. 创建项目

在IDEA中[创建Spring Boot项目](#)chapter06，读者可以根据自己当前情况选择使用Spring Initializr方式或者Maven方式进行创建，在此选择使用Maven方式创建项目。



2.配置依赖

为了更好地演示应用缓存后的效果，创建的Spring Boot项目中整合了Spring MVC、Spring Data JPA、MySQL。在项目chapter06的pom.xml文件中配置Spring MVC、Spring Data JPA、MySQL，以及Spring Boot提供的缓存启动器的依赖，具体如文件6-1所示。



源代码

文件6-1
[pom.xml](#)





3. 设置配置信息

在项目的resources目录下创建application.yml文件，在该文件中指定数据库连接信息和JPA的配置信息，具体如文件6-2所示。



源代码

文件6-2
[application.xml](#)





4. 创建实体类

在项目的java目录下创建包com.itheima.chapter06.entity，并在该包下创建**实体类Book**，具体如文件6-3所示。



源代码

文件6-3
[Book.java](#)





5. 创建Repository接口

在java目录下创建包com.itheima.chapter06.dao，在该包下自定义接口BookRepository继承JpaRepository接口，具体如文件6-4所示。



源代码

文件6-4

BookRepository.java





6.创建Service接口和实现类

在项目的java目录下创建包com.itheima.chapter06.service，并在该包下创建Service接口和实现类，具体如文件6-5和文件6-6所示。



源代码

文件6-5 [BookService.java](#)

文件6-6 [BookServiceImpl.java](#)





7.创建控制器类

在项目的java目录下创建包com.itheima.chapter06.controller，并在该包下创建控制器类BookController，在该类中定义查询、更新、删除图书信息的方法，具体如文件6-7所示。



源代码

文件6-7

BookController.java





8.创建项目启动类

在项目的java目录下的com.itheima.chapter06包下创建Spring Boot项目的启动类，在启动类上开启缓存，具体如文件6-8所示。



源代码

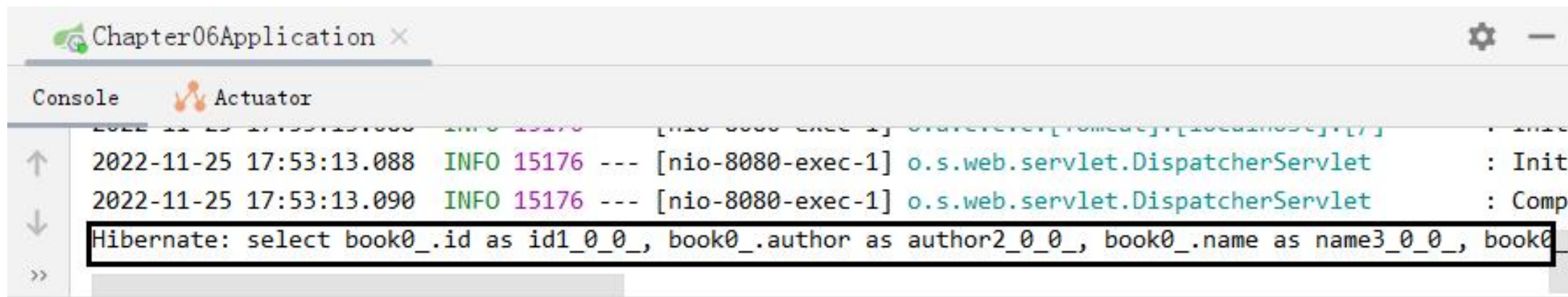
文件6-8

[Chapter06Application.java](#)



9.测试缓存效果

运行文件6-8后，在浏览器中访问<http://localhost:8080/book/findById/3>，查询图书信息，控制台输出信息。



The screenshot shows an IDE window titled 'Chapter06Application'. The 'Console' tab is active, displaying the following log messages:

```
2022-11-25 17:53:13.088 INFO 15176 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Init
2022-11-25 17:53:13.090 INFO 15176 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Comp
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_.
```

The last line of the log, which is the Hibernate SQL query, is highlighted with a black rectangular box.



9.测试缓存效果

查询图书信息后，浏览器中查询到图书信息。





9.测试缓存效果

再次在浏览器中访问<http://localhost:8080/book/findById/3>，查询id为3的图书信息，控制台输出信息。

```
Chapter06Application x
Console Actuator
2022-11-25 17:53:13.088 INFO 15176 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Init
2022-11-25 17:53:13.090 INFO 15176 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Comp
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_.
```



9.测试缓存效果

在浏览器中访问<http://localhost:8080/book/editById/3/西游释厄传>，将id为3的图书名称更新为“西游释厄传”，此时控制台输出信息。

```
Run: Chapter06Application x
Console Actuator
2022-11-25 17:53:13.088 INFO 15176 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Init
2022-11-25 17:53:13.090 INFO 15176 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Comp
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_
Hibernate: update book set author=?, name=?, press=?, status=? where id=?
```



9.测试缓存效果

更新图书信息后，浏览器中查询图书信息。





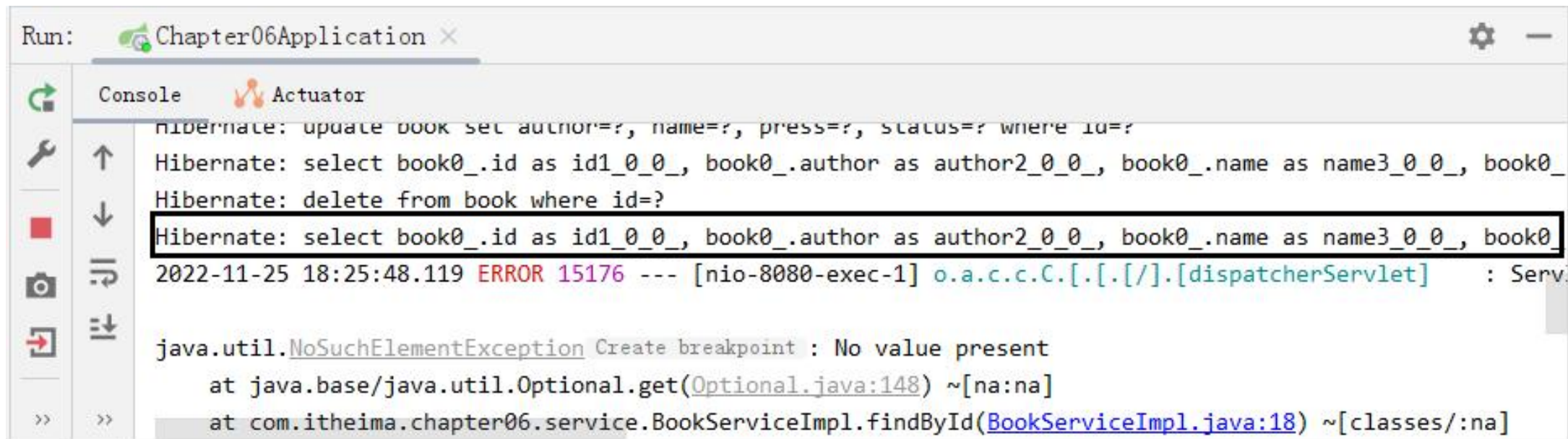
9.测试缓存效果

浏览器中访问<http://localhost:8080/book/delById/3>，删除id为3的图书信息，控制台输出信息。

```
Run: Chapter06Application x
Console Actuator
2022-11-25 17:53:13.088 INFO 15176 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Init
2022-11-25 17:53:13.090 INFO 15176 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Comp
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_
Hibernate: update book set author=?, name=?, press=?, status=? where id=?
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_
Hibernate: delete from book where id=?
```


9.测试缓存效果

在浏览器中再次访问<http://localhost:8080/book/findById/3>，查询id为3的图书信息，控制台输出信息。



```
Run: Chapter06Application x
Console
Hibernate: update book set author=?, name=?, press=?, status=? where id=?
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_
Hibernate: delete from book where id=?
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as name3_0_0_, book0_
2022-11-25 18:25:48.119 ERROR 15176 --- [nio-8080-exec-1] o.a.c.c.C.[.][.][dispatcherServlet] : Serv
java.util.NoSuchElementException Create breakpoint : No value present
    at java.base/java.util.Optional.get(Optional.java:148) ~[na:na]
    at com.itheima.chapter06.service.BookServiceImpl.findById(BookServiceImpl.java:18) ~[classes/:na]
```



6.2

Spring Boot整合Ehcache缓存



Spring 提供的缓存是一种抽象的服务，开发人员只需引入缓存接口的具体实现，而不必编写缓存的具体逻辑，便于进行缓存技术的开发与管理。在Spring Boot项目中，只需要引入对应缓存实现的依赖，即可使用该缓存，其中Ehcache是当前最快的Java缓存之一，下面对Spring Boot整合Ehcache缓存的相关内容讲解。



了解Ehcache概述，能够说出Ehcache的特点



Ehcache是一种开源的缓存框架，它配置简单、结构清晰、功能强大，是当前使用最广泛的基于Java语言的缓存之一。Ehcache可以很便捷地与其他流行的库和框架进行集成，其中，Hibernate默认的缓存提供者就是EhCache。

Ehcache能够成为目前使用最广泛的缓存框架之一，主要得益于它的以下几个特点。

- 1.快速轻量
- 2.伸缩性
- 3.灵活性
- 4.标准支持
- 5.可扩展性
- 6.监听器
- 7.Java企业级应用缓存
- 8.开源协议Apache 2.0 license



Ehcache支持分层缓存，所有分层缓存都可以单独使用，如下为Ehcache支持的分层选项。

- 堆：堆内存速度快，不需要序列化，但是容量有限。
- OffHeap（堆外）：OffHeap存储只在企业版本的Ehcache中提供，原理是利用NIO的DirectByteBuffer实现，比存储到磁盘上快，而且完全不受GC（Garbage Collection，垃圾收集）的影响，可以保证响应时间的稳定性；OffHeap存储的对象必须在存储过程中进行序列化，读取则进行反序列化操作，它的速度大约比堆内存存储慢一个数量级。
- 磁盘：“磁盘”层数据存储在硬盘上，磁盘越快、越专用，访问数据的速度就越快。在磁盘写入/读取的数据必须序列化/反序列化，因此磁盘存储比堆存储和OffHeap存储要慢。
- 集群：使用集群层存储意味着客户端连接到Terracotta服务器阵列，缓存的数据存储在Terracotta服务器，这也是JVM之间共享缓存的一种方式。



如果需要使用多个层，必须遵守如下要求。

- (1) 在多层设置中必须始终存在堆层。
- (2) 不能将磁盘层和集群层组合在一起。
- (3) 层的大小应该按照金字塔的方式调整，也就是说，金字塔上更高的层比更低的层使用更少的内存。





6.2.2 整合Ehcache



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌



掌握整合Ehcache，能够在Spring Boot项目中整合Ehcache，并正确应用声明式缓存注解



6.2.2 整合Ehcache



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

1.配置依赖

在项目chapter06的pom.xml文件中配置添加Ehcache的依赖。





6.2.2 整合Ehcache



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

2. 设置配置信息

在项目chapter06的application.yml文件中添加Ehcache的相关配置。





6.2.2 整合Ehcache



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

2. 设置配置信息

在项目chapter06的resources目录下创建文件`ehcache.xml`，在该文件中添加Ehcache的相关配置，具体如文件6-9所示。



源代码

文件6-9
`ehcache.xml`





6.2.2 整合Ehcache



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

3.设置实体类

由于本案例要将缓存序列化到本地磁盘，所以需要业务中操作的**实体类**需要**实现序列化接口**。将文件文件6-3中的Book类实现序列化接口，修改后如文件6-10所示。



源代码

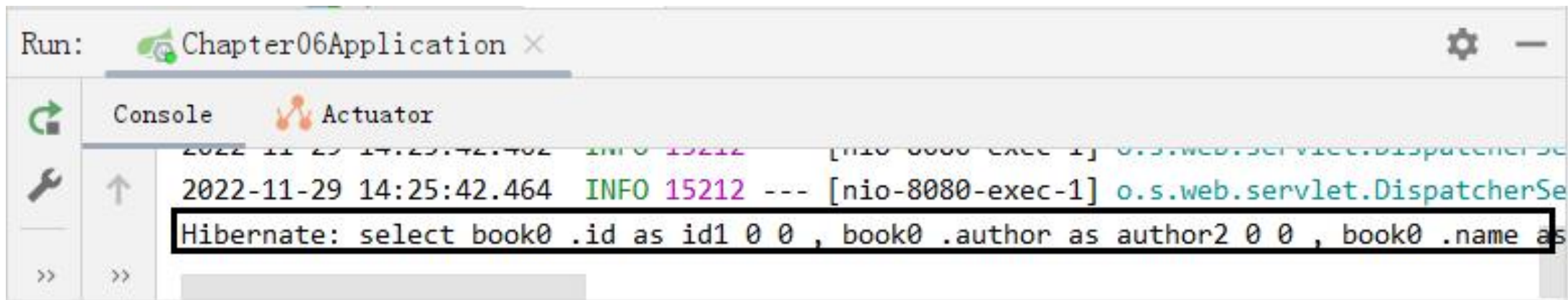
文件6-10

[Book.java](#)



4.测试缓存效果

运行文件6-8后，在浏览器中访问<http://localhost:8080/book/findById/5>，查询图书信息，控制台输出信息。



The screenshot shows a console window for 'Chapter06Application'. The 'Console' tab is active, displaying logs. A specific log line is highlighted with a black box:

```
2022-11-29 14:25:42.464 INFO 15212 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherSe  
Hibernate: select book0 .id as id1 0 0 , book0 .author as author2 0 0 , book0 .name as
```



6.2.2 整合Ehcache



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

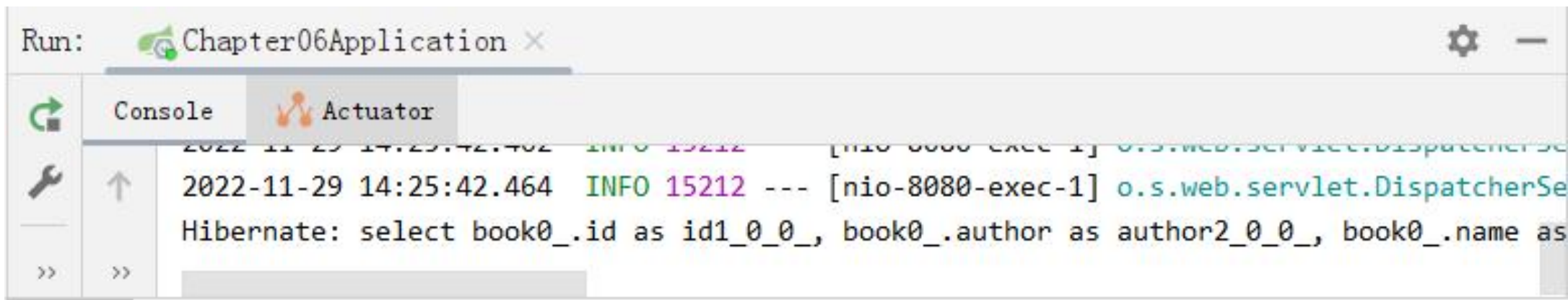
4.测试缓存效果

查询图书信息后，浏览器中查询到图书信息。



4.测试缓存效果

再次在浏览器中访问<http://localhost:8080/book/findById/5>，查询id为5的图书信息，控制台输出信息。

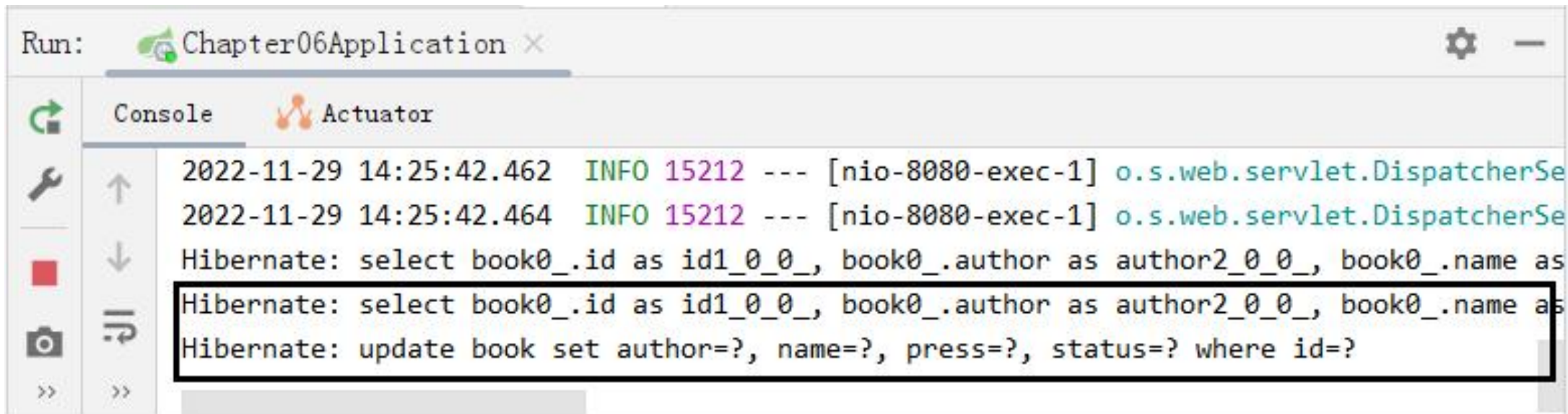


The screenshot shows a console window for 'Chapter06Application'. The 'Console' tab is active, displaying the following log entries:

```
2022-11-29 14:25:42.402 INFO 15212 [nio-8080-exec-1] o.s.web.servlet.DispatcherSe
2022-11-29 14:25:42.464 INFO 15212 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherSe
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as
```

4.测试缓存效果

在浏览器中访问<http://localhost:8080/book/editById/5/观堂别集>，将id为5的图书名称更新为“观堂别集”，此时控制台输出信息。



```
Run: Chapter06Application x
Console Actuator
2022-11-29 14:25:42.462 INFO 15212 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherSe
2022-11-29 14:25:42.464 INFO 15212 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherSe
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_, book0_.name as
Hibernate: update book set author=?, name=?, press=?, status=? where id=?
```



6.2.2 整合Ehcache



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

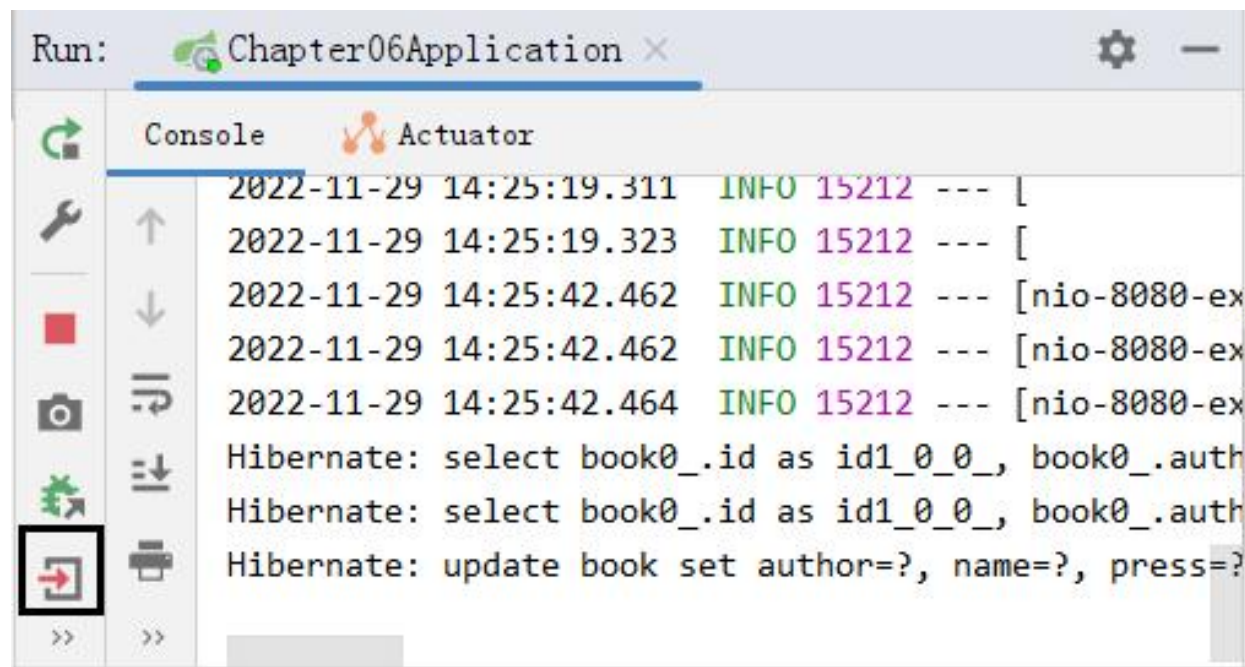
4.测试缓存效果

更新图书信息后，浏览器中查询到图书信息。



4.测试缓存效果

单击IDEA控制台左侧的按钮，正常关闭程序。





6.2.2 整合Ehcache

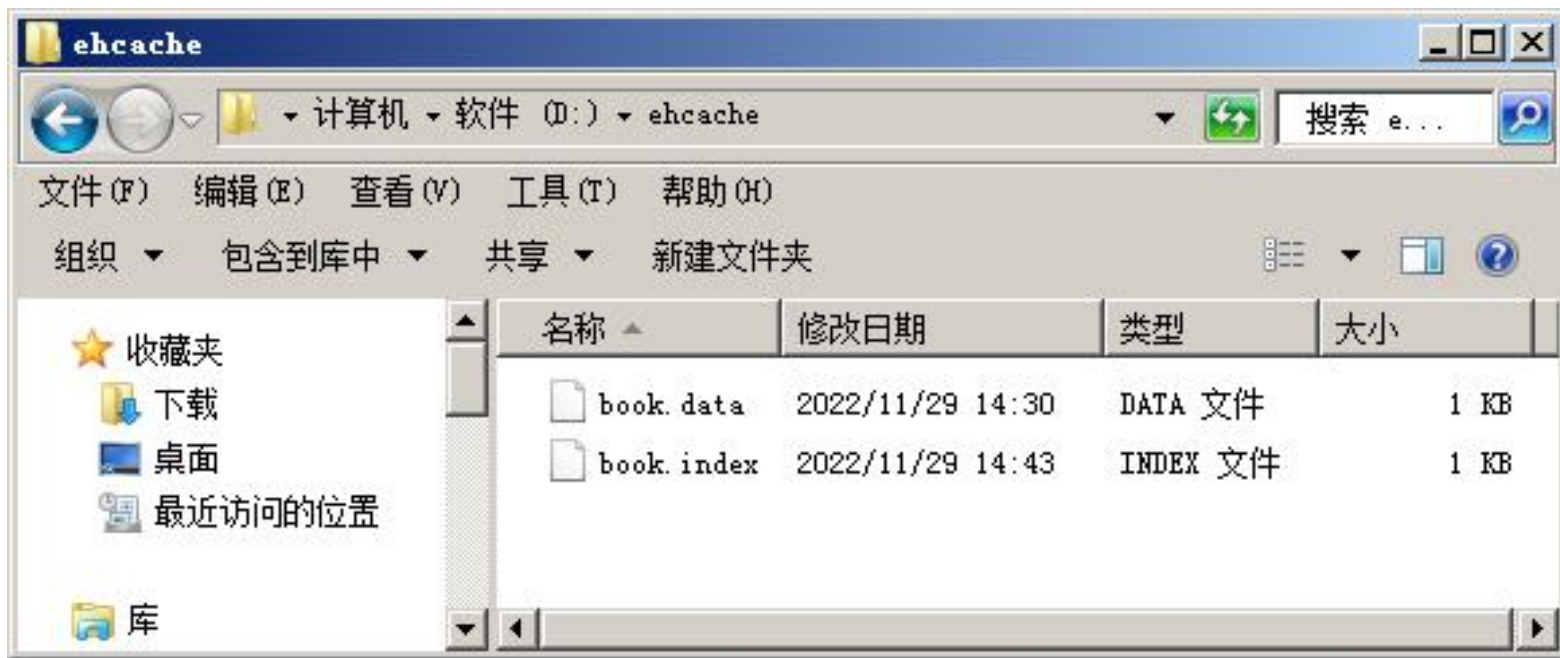


黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

4.测试缓存效果

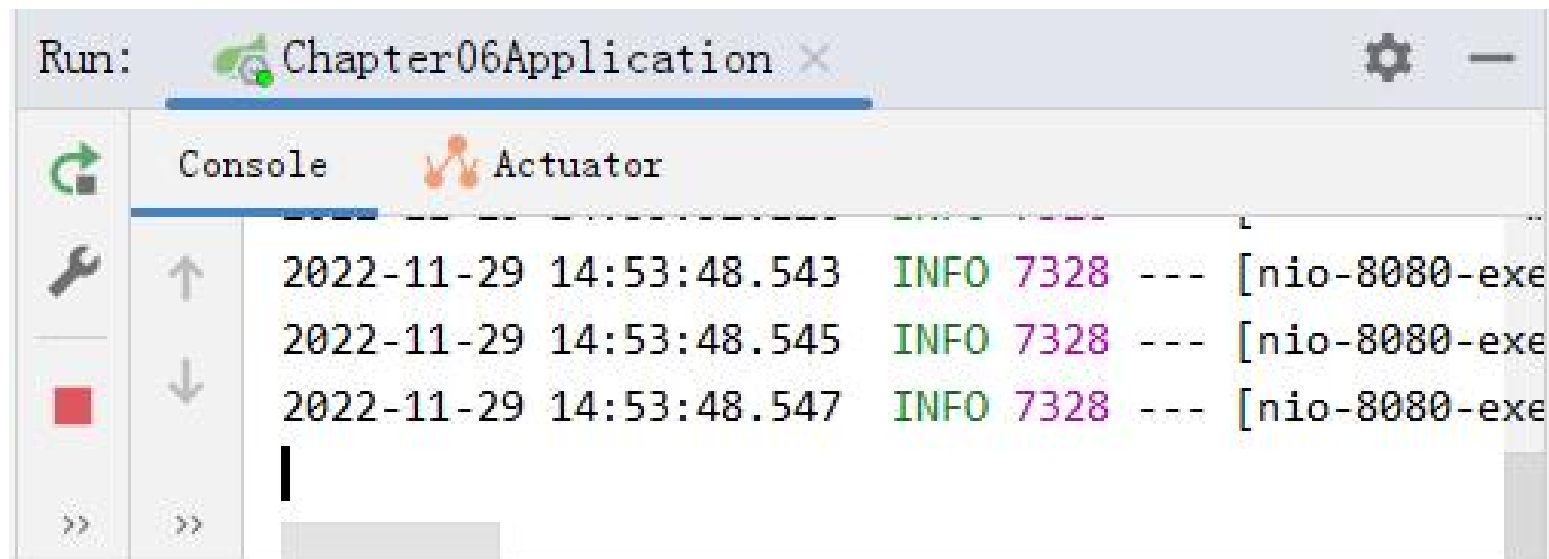
程序关闭后，打开之前设置的缓存数据保存到本地磁盘的文件路径。





4.测试缓存效果

此时，再次运行文件6-8后，在浏览器中访问<http://localhost:8080/book/findById/5>，查询图书信息，控制台输出信息。





6.2.2 整合Ehcache



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

4.测试缓存效果

查询图书信息后，浏览器中查询到图书信息。





6.3

SpringBoot整合Redis缓存



掌握SpringBoot整合Redis缓存，能够在Spring Boot项目中整合Redis缓存，并正确应用声明式缓存注解



Redis优秀的数据处理能力和丰富的数据结构，使其可以使用的业务场景非常广泛，既可以作为做数据库使用，又可以作为缓存使用。关于Redis 作为数据的相关演示在第5章已经完成，下面通过案例演示Redis 作为缓存在Spring Boot项目中的使用。



1.配置依赖

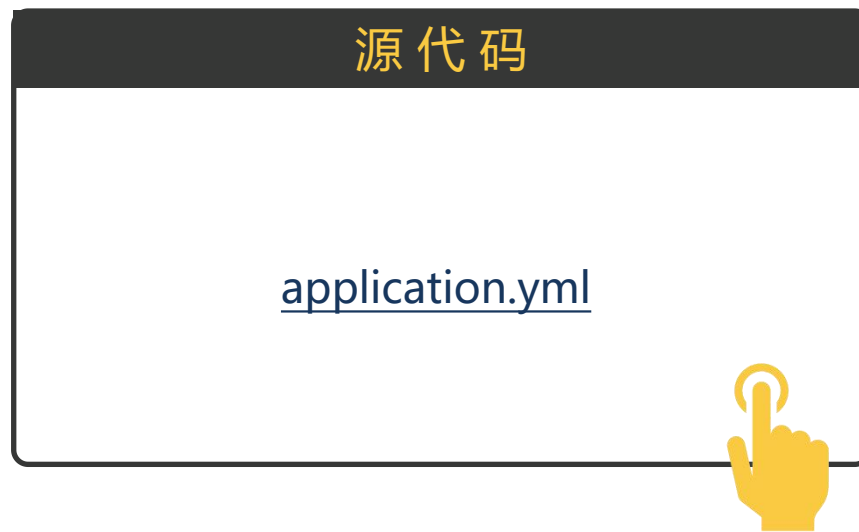
在项目chapter06的pom.xml文件中将Ehcache的依赖进行注释，配置添加Redis的依赖。





2. 设置配置信息

在项目chapter06的application.yml文件中将原有配置的Ehcache相关的信息进行注释，添加Redis缓存相关的配置。





3.测试缓存效果

启动Redis服务，运行文件6-8后，在浏览器中访问<http://localhost:8080/book/findById/5>，查询图书信息，控制台输出信息。

```
Run: Chapter06Application x
Console Actuator
2022-11-29 15:50:49.506 INFO 15388 --- [nio-8080-exec-1] o.a.c.c.C.[To
2022-11-29 15:50:49.506 INFO 15388 --- [nio-8080-exec-1] o.s.web.servl
2022-11-29 15:50:49.507 INFO 15388 --- [nio-8080-exec-1] o.s.web.servl
Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_,
```




3.测试缓存效果

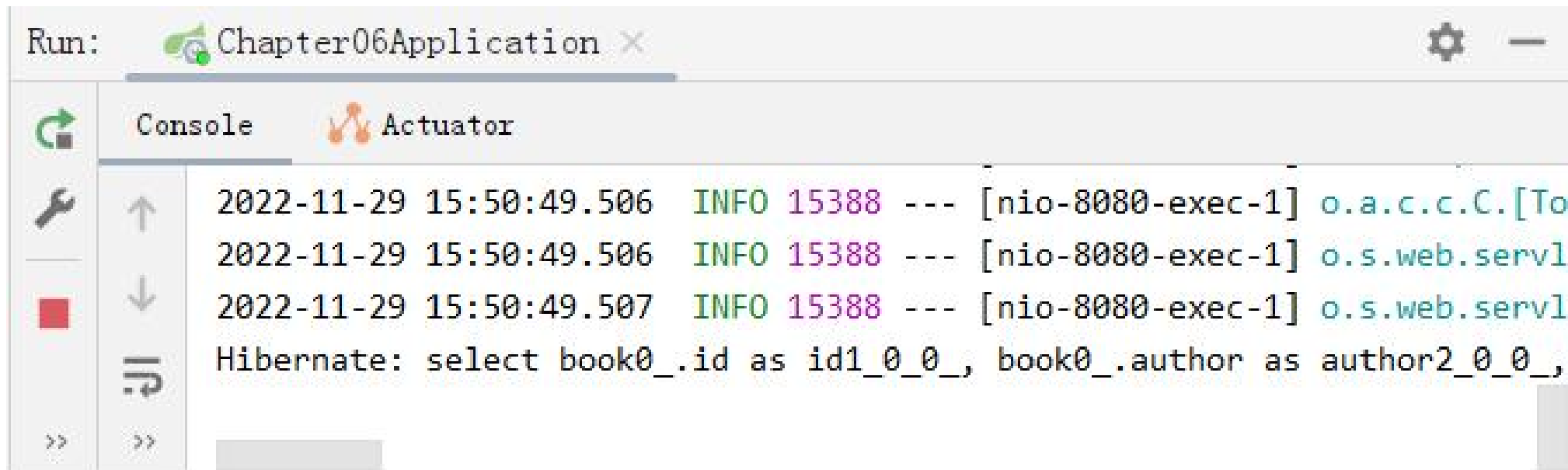
查询图书信息后，浏览器中查询到图书信息。





3.测试缓存效果

再次查询id为5的图书信息，控制台输出信息。



The screenshot shows an IDE console window titled "Run: Chapter06Application". It has tabs for "Console" and "Actuator". The console output displays three log entries:

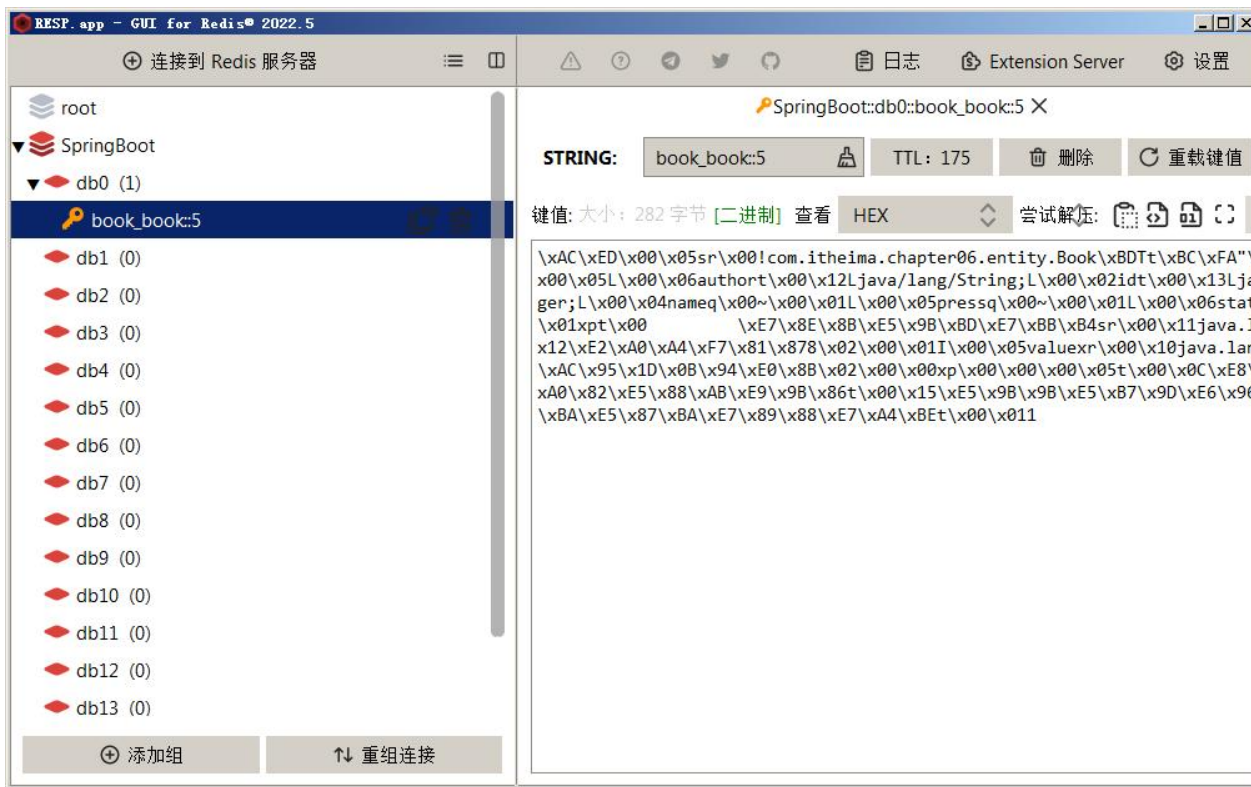
```
2022-11-29 15:50:49.506 INFO 15388 --- [nio-8080-exec-1] o.a.c.c.C.[To
2022-11-29 15:50:49.506 INFO 15388 --- [nio-8080-exec-1] o.s.web.servl
2022-11-29 15:50:49.507 INFO 15388 --- [nio-8080-exec-1] o.s.web.servl
```

Below these logs, the text "Hibernate: select book0_.id as id1_0_0_, book0_.author as author2_0_0_," is visible, indicating a database query.



3.测试缓存效果

打开RESP.app，查看Redis中的数据。





本章小结

本章主要对Spring Boot整合缓存进行了讲解。首先讲解了Spring Boot默认缓存管理；然后讲解了Spring Boot整合Ehcache缓存；最后讲解了Spring Boot整合Redis缓存。通过本章的学习，希望大家可以在Spring Boot项目中正确应用缓存技术，为后续更深入地学习Spring Boot做好铺垫。

為千萬學生少走彎路而著書



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌