# Spring Boot缓存简介

Spring 3.1版本开始对缓存提供支持，核心思路是对方法的缓存，当调用一个方法时，将该方法的参数和返回值作为key/value缓存起来，当再次调用该方法时，如果缓存中有数据，就直接从缓存中获取，否则再去执行该方法。

在Spring Boot中支持的缓存有如下几种：Jcache、EhCache 2.x、Hazelcast、Infinispan、Couchbase、Redis 、Caffeine、Simple。

# Ehcache2.x缓存

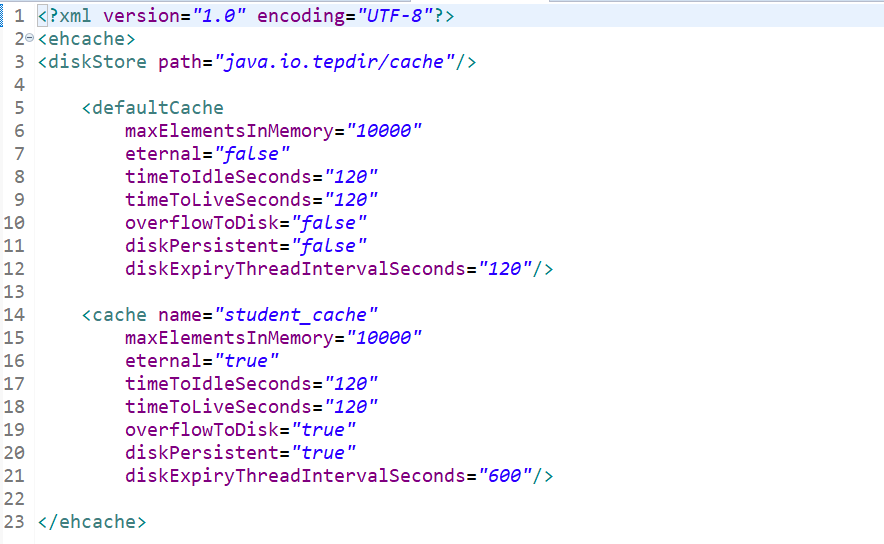
Ehcache缓存在Java开发领域已是久负盛名。

## 配置

1. 添加spring-boot-starter-cache依赖，如下：



1. 在classpath路径下添加缓存配置文件ehcache.xml，如下：



在配置文件中，提供了两个缓存策略，一个是默认的，另一名为student\_cache；name表示缓存名称；

maxElementsInMemory表示缓存最大个数；

eternal表示缓存对象是否永久有效，一旦设置了永久有效，timeout将不起作用；

timeToIdleSeconds表示缓存对象在失效前的允许闲置时间，单位秒，只有当eternal=false时（即缓存对象不是永久有效时），该属性才会生效；

timeToLiveSeconds表示缓存对象在失效前允许存活的时间，单位秒，只有当eternal=false时（即缓对象不是永久有效时），该属性才会生效；

overflowToDisk表示当内存中的对象数量达到了maxElementsInMemory（缓存最大个数）时，Ehcache是否将缓存对象写入磁盘中；

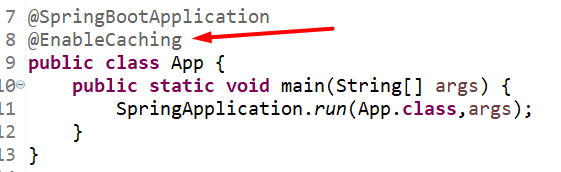
diskExpiryThreadIntervalSeconds表示磁盘失效线程运行时间间隔；

除了这些，还有其他更为详细的Ehcache配置。

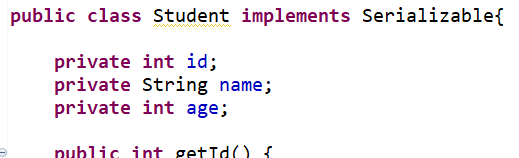
1. 如果开发者想自定义Ehcache配置文件的名称和位置，可以在application.properties中配置，如下：

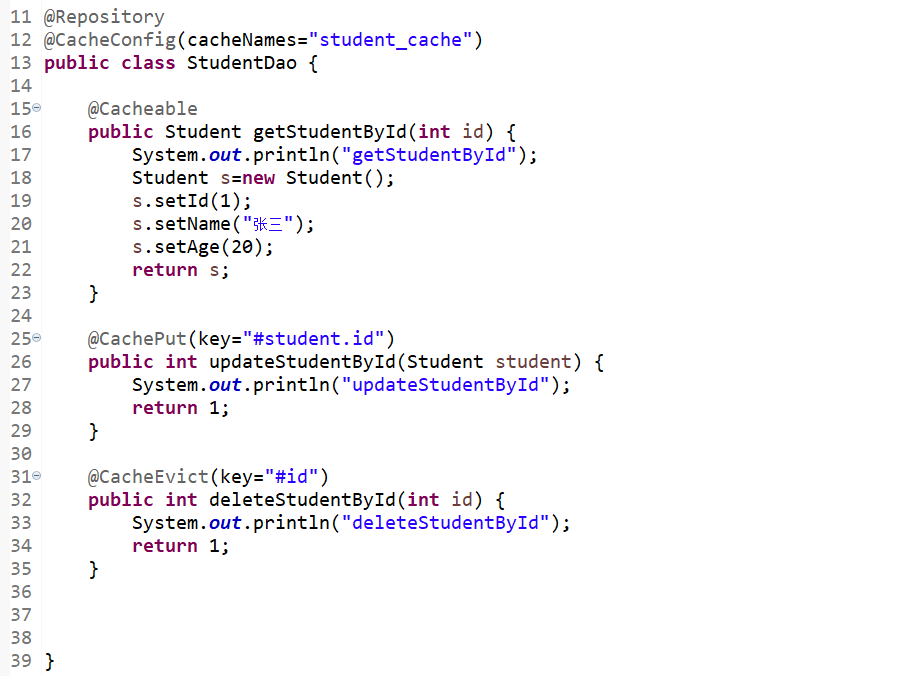


1. 在项目启动类上添加@EnableCaching注解开启缓存，如下：



## 使用

1. 创建体类Student，需要缓存的对象必须实现序列化接口，如下：
2. 创建StudentDao，如下



在StudentDao上添加@CacheConfig指明使用的缓存策略名称，这个配置是可选的，若不使用@CacheConfig注解，则直接在@Cacheable注解中指明缓存策略名称；

在getStudentById方法上添加@Cacheable注解表示对该方法进行缓存，默认情况下，缓存的key是方法的参数，缓存的value是方法的返回值；当开发者在其他类中调用该方法时，首页会根据调用参数查看缓存中是否有相关数据，若有，则直接使用缓存数据，该方法不会执行，否则执行该方法；如果是在当前类中调用该方法，则缓存不会生效；

第25行的@CachePut注解一般标注在数据更新的方法上，与@Cacheable注解不同，添加了@CachePut注解的方法每次在执行时都不会去检查缓存中是否有数据，而是直接执行方法，然后将方法的执行结构缓存起来，如果该key对于的数据已经被缓存起来了，就会覆盖之前的数据，这样就可以避免再次加载数据时获取到脏数据。同时，@CachePut具有和@Cacheable相似的属性（请自行查询）。

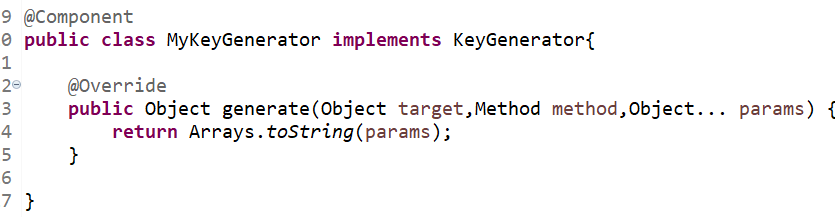
第31行的@CacheEvict注解一般用于删除方法上，表示移除一个key对应的缓存。@CacheEvict注解有两个特殊的属性：allEntries和beforeInvocation；allEntries表示是否将所有的缓存数据都移除，默认为false，beforeInvocation表示是否在方法执行之前移除缓存中的数据，默认为false，即在方法执行之后移除缓存中的数据；

@Cacheable注解还有一个可选属性condition，用来描述缓存的执行时机，例如@Cacheable(condition=”#id%2==0”)表示当形参id的值取模为0时才进行缓存，否则不缓存；

如果开发者不想使用默认的key，也可以像第25行和31行那样自定义key，在第25行，通过@CachePut，将形参student对象的id属性作为key；在第31行，通过@CacheEvict，将形参id作为key；如果还不能满足开发者定义自定义key的需求，Spring中提供了一个root对象用来生成key，如下：

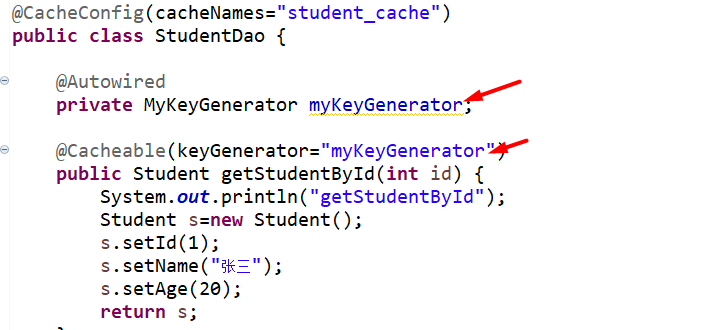
|  |  |  |
| --- | --- | --- |
| 属性名称 | 属性描述 | 用法示例 |
| mehtodName | 当前方法名 | #root.methodName |
| method | 当前方法对象 | #root.method.name |
| caches | 当前方法使用的缓存 | #root.caches[0].name |
| target | 当前被调用的对象 | #root.target |
| targetClass | 当前被调用的对象的class | #root.targetClass |
| args | 当前方法参数数组 | #root.args[0[ |

如果这些key还不能满足开发者需求，开发者也可以自定义缓存key的生存器KeyGenerator，如下：

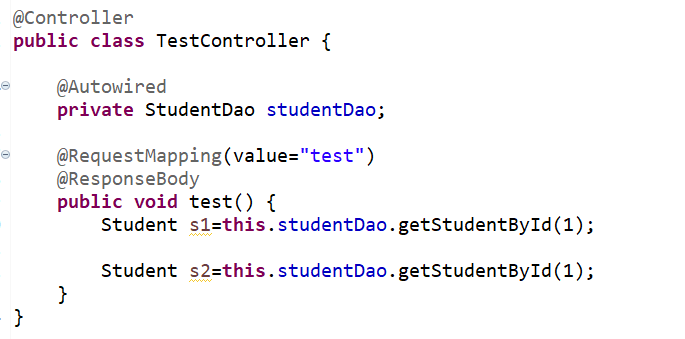


自定义MyKeyGenerator实现KeyGenerator接口，然后实现该接口中的generator方法，方法的三个参数分别是当前对象，当前请求的方法以及方法的参数，开发者可根据这些信息生成一个新的key返回，返回值就是缓存的key。

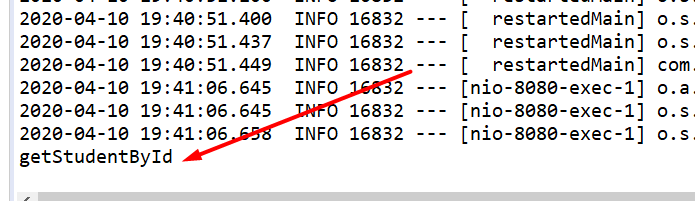
然后通过以下方式引用：



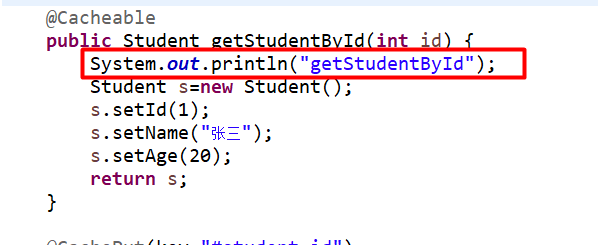
1. 为了方便起见，直接在控制层中注入持久层实例进行测试，如下：



1. 启动项目，打开浏览器访问路径，观察打印台输出。



在控制层中调用了两次getStudentById方法，按照StudnetDao中getStudentById方法的代码，如下：

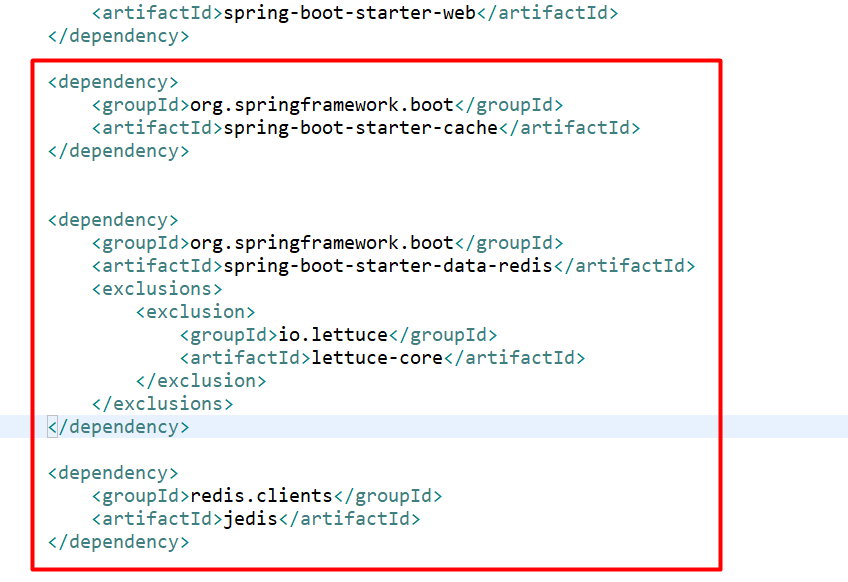


它应该打印两次，而事实只打印了一次，因为第二次使用了缓冲。

# Redis单机缓存

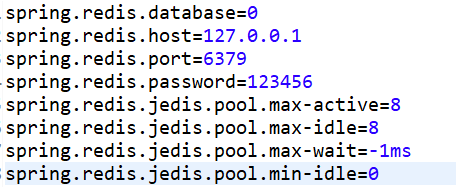
## 配置

1. 添加spring-booot-starter-cache和Redis依赖，如下：



在这里排除了lettuce，使用Jedis客户端；

1. 配置Redis连接信息，如下：



可参考Spring Boot整合NoSQL章节；

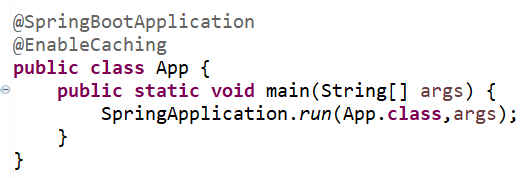
1. 配置Redis缓存，如下：



cache-names表示缓存名称，Redis中的key都有一个名称，默认前缀就是“缓存名称::”；

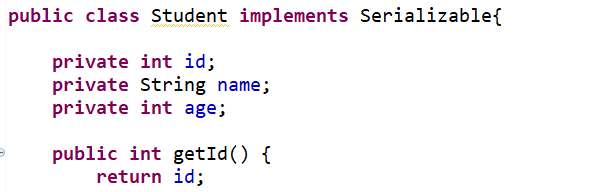
time-to-live配置key的有效期；

1. 在项目启动类上使用@EnableCaching开启缓存，如下：



## 使用

1. 创建Student实体类，如下：



注意：该实体类实现了序列化接口；

1. 为了方便，创建StudentDao，如下：

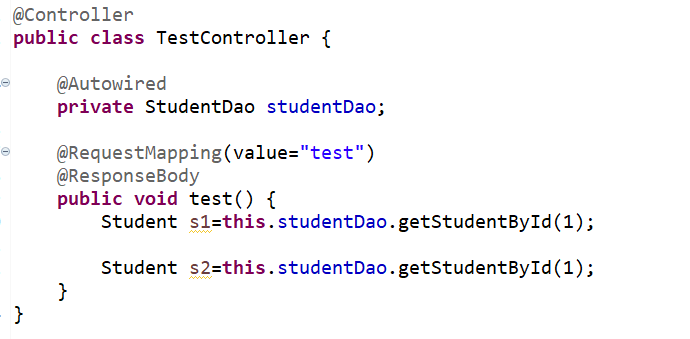
@CacheConfig指定缓存名称；

@Cacheable表示缓存该方法；

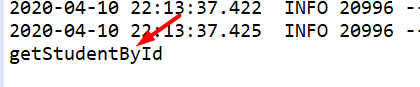
@CachePut表示更新缓存；

@CacheEvict表示清除指定的缓存；

1. 为了方便，直接使用在控制层注入持久层实例，如下：



1. 启动项目，进入浏览器访问，观察打印台输出，如下：



很明显，这里只打印了一次，而我在请求处理方法中调用了两次，可以分析得出，第二次使用了缓存。