



## 《Spring Boot 基础教程》

## 第 19 节

## 使用 Caching-EhCache

Spring boot 支持的缓存：

- Generic
- JCache (JSR-107)
- EhCache 2.x
- Hazelcast
- Infinispan
- Couchbase
- Redis
- Caffeine
- Guava
- Simple

最常用的是 EhCache，文档多，资料全

### 一、添加依赖

```
<!-- caching -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-cache</artifactId>
</dependency>
<dependency>
    <groupId>net.sf.ehcache</groupId>
    <artifactId>ehcache</artifactId>
</dependency>
```

### 二、配置文件：

```
spring.cache.type=ehcache
```

```
spring.cache.ehcache.config=classpath:config/ehcache.xml
```

ehcache.xml

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="ehcache.xsd">
```

```
    <cache name="roncooCache"
        eternal="false"
        maxEntriesLocalHeap="0"
        timeToIdleSeconds="50"></cache>
```

```
<!-- eternal: true表示对象永不过期，此时会忽略timeToIdleSeconds和
timeToLiveSeconds属性，默认为false -->
```

```
<!-- maxEntriesLocalHeap: 堆内存中最大缓存对象数，0没有限制 -->
```



```
<!-- timeToIdleSeconds: 设定允许对象处于空闲状态的最长时间，以秒为
单位。当对象自从最近一次被访问后，如果处于空闲状态的时间超过了
timeToIdleSeconds属性值，这个对象就会过期，EHCache将把它从缓存中清空。
只有当eternal属性为false，该属性才有效。如果该属性值为0，则表示对象可以
无限期地处于空闲状态 -->
</ehcache>
```

### 三、启用注解支持：

@EnableCaching：启用缓存注解

代码实现：

```
/**
 * @author wujing
 */
public interface RoncooUserLogCache {

    /**
     * 查询
     *
     * @param id
     * @return
     */
    RoncooUserLog selectById(Integer id);

    /**
     * 更新
     *
     * @param roncooUserLog
     * @return
     */
    RoncooUserLog updateById(RoncooUserLog roncooUserLog);

    /**
     * 删除
     *
     * @param id
     * @return
     */
    String deleteById(Integer id);
}
```



实现类:

```
/**
 * @author wujing
 */
@CacheConfig(cacheNames = "roncooCache")
@Repository
public class RoncooUserLogCacheImpl implements
RoncooUserLogCache {

    @Autowired
    private RoncooUserLogDao roncooUserLogDao;

    @Cacheable(key = "#p0")
    @Override
    public RoncooUserLog selectById(Integer id) {
        System.out.println("查询功能，缓存找不到，直接读库， id=" +
id);
        return roncooUserLogDao.findOne(id);
    }

    @CachePut(key = "#p0")
    @Override
    public RoncooUserLog updateById(RoncooUserLog
roncooUserLog) {
        System.out.println("更新功能，更新缓存，直接写库， id=" +
roncooUserLog);
        return roncooUserLogDao.save(roncooUserLog);
    }

    @CacheEvict(key = "#p0")
    @Override
    public String deleteById(Integer id) {
        System.out.println("删除功能，删除缓存，直接写库， id=" + id);
        return "清空缓存成功";
    }
}
```

注解说明:

@CacheConfig: 缓存配置

@Cacheable: 应用到读取数据的方法上，即可缓存的方法，如查找方法：先从缓存中读取，如果没有再调用方法获取数据，然后把数据添加到缓存中。**适用于查找**

@CachePut: 主要针对方法配置，能够根据方法的请求参数对其结果进行缓存，和 @Cacheable 不同的



是，它每次都会触发真实方法的调用。**适用于更新和插入**

@CacheEvict：主要针对方法配置，能够根据一定的条件对缓存进行清空。**适用于删除**

#### 测试：

```
@RequestMapping(value = "/select", method = RequestMethod.GET)
public RoncooUserLog get(@RequestParam(defaultValue = "1") Integer id) {
    return RoncooUserLogCache.selectById(id);
}

@RequestMapping(value = "/update", method = RequestMethod.GET)
public RoncooUserLog update(@RequestParam(defaultValue = "1") Integer id) {
    RoncooUserLog bean = RoncooUserLogCache.selectById(id);
    bean.setUserName("测试");
    bean.setCreateTime(new Date());
    RoncooUserLogCache.updateById(bean);
    return bean;
}

@RequestMapping(value = "/del", method = RequestMethod.GET)
public String del(@RequestParam(defaultValue = "1") Integer id) {
    return RoncooUserLogCache.deleteById(id);
}
```

更多课程信息，请关注 **龙果学院** 官方网站 <http://www.roncoo.com/>

或关注 **龙果** 微信公众号 **RonCoo\_com**

