# 为什么要使用缓存

## 缓存的分类

## 客户端缓存

页面缓存

浏览器缓存

App客户端缓存

## 网络缓存

代理缓存

CDN缓存

## 服务器缓存

数据库缓存

平台缓存级缓存

缓存最终的目的是为减轻服务端压力，减少网络传输请求

# 基于Map集合实现本地缓存

## 定义Map缓存工具类

|  |
| --- |
| @Component  **public** **class** MapEhcaChe<K, V> {  **private** Map<K, V> ehcaCheMap = **new** ConcurrentHashMap<K, V>();  **public** **void** put(K k, V value) {  ehcaCheMap.put(k, value);  }  **public** V get(K k) {  **return** ehcaCheMap.get(k);  }  **public** **void** remove(K k) {  ehcaCheMap.remove(k);  }  } |

## 使用案例

|  |
| --- |
| @RequestMapping("/ehcaChePut")  **public** String put(String key, String value) {  mapEhcaChe.put(key, value);  **return** "success";  }  @RequestMapping("/get")  **public** String get(String key) {  String value = mapEhcaChe.get(key);  **return** value;  } |

# 本地缓存Ehcache

## 什么是Ehcache

 Ehcache是纯java的开源缓存框架，具有快速、精干等特点，是Hibernate中默认的CacheProvider。它主要面向通用缓存、Java EE和轻量级容器，具有内存和磁盘存储、缓存加载器、缓存扩展、缓存异常处理程序。

 Ehcache最初由Greg Luck于2003年开始开发。2009年,该项目被Terracotta购买。软件仍然开源,但一些新的主要功能(例如，快速可重启性之间的一致性的)只能在商业产品中使用。

**Ehcache 被广泛用于在Hibernate、Spring、Cocoon等其他开源系统。**

## Ehcache的主要特性

1.快速；

2.简单；

3.多种缓存策略；

4.缓存数据有两级：内存和磁盘，因此无需担心容量问题；

5.缓存数据会在虚拟机重启的过程中写入磁盘；

6.可以通过 RMI、可插入 API 等方式进行分布式缓存；

7.具有缓存和缓存管理器的侦听接口；

8.支持多缓存管理器实例，以及一个实例的多个缓存区域；

9.提供 Hibernate 的缓存实现；

## Ehcache使用介绍

Ehcache是用来管理缓存的一个工具，**其缓存的数据可以是存放在内存里面的，也可以是存放在硬盘上的**。其核心是CacheManager，一切Ehcache的应用都是从CacheManager开始的。它是用来管理Cache（缓存）的，一个应用可以有多个CacheManager，而一个CacheManager下又可以有多个Cache。Cache内部保存的是一个个的Element，而一个Element中保存的是一个key和value的配对，相当于Map里面的一个Entry。

## Ehcache缓存过期策略

当缓存需要被清理时（比如空间占用已经接近临界值了），需要使用某种淘汰算法来决定清理掉哪些数据。常用的淘汰算法有下面几种：

FIFO：First In First Out，先进先出。判断被存储的时间，离目前最远的数据优先被淘汰。

LRU：Least Recently Used，最近最少使用。判断最近被使用的时间，目前最远的数据优先被淘汰。

LFU：Least Frequently Used，最不经常使用。在一段时间内，数据被使用次数最少的，优先被淘汰。

## Ehcache基本用法

SpringBoot2.0整合Ehcache框架

### Maven环境依赖

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.0.RELEASE</version>  </parent>  <dependencies>  <!-- SpringBoot 对lombok 支持 -->  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  </dependency>  <!-- SpringBoot web 核心组件 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-tomcat</artifactId>  </dependency>  <!-- SpringBoot 外部tomcat支持 -->  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  </dependency>  <!-- springboot-log4j -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-log4j</artifactId>  <version>1.3.8.RELEASE</version>  </dependency>  <!-- springboot-aop 技术 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-aop</artifactId>  </dependency>  <!-- https://mvnrepository.com/artifact/commons-lang/commons-lang -->  <dependency>  <groupId>commons-lang</groupId>  <artifactId>commons-lang</artifactId>  <version>2.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient -->  <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  </dependency>  <!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>fastjson</artifactId>  <version>1.2.47</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency>  <dependency>  <groupId>taglibs</groupId>  <artifactId>standard</artifactId>  <version>1.1.2</version>  </dependency>  <!--开启 cache 缓存 -->  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-cache</artifactId>  </dependency>  <!-- ehcache缓存 -->  <dependency>  <groupId>net.sf.ehcache</groupId>  <artifactId>ehcache</artifactId>  <version>2.9.1</version><!--$NO-MVN-MAN-VER$ -->  </dependency>  <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  <version>1.1.1</version>  </dependency>  <!-- mysql 依赖 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  </dependency>  </dependencies> |

### YML配置文件信息

|  |
| --- |
| ###端口号配置  server:  port: 8081  ###数据库配置  spring:  datasource:  url: jdbc:mysql://localhost:3306/test  username: root  password: root  driver-class-name: com.mysql.jdbc.Driver  test-while-idle: **true**  test-on-borrow: **true**  validation-query: SELECT 1 FROM DUAL  time-between-eviction-runs-millis: 300000  min-evictable-idle-time-millis: 1800000  # 缓存配置读取  cache:  type: ehcache  ehcache:  config: classpath:app2\_ehcache.xml |

### App启动方式

|  |
| --- |
| @MapperScan(basePackages = { "com.itmayiedu.mapper" })  @EnableCaching  @SpringBootApplication  **public** **class** App {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(App.**class**, args);  }  } |

**@EnableCaching 开启ehcache缓存模式**

### 项目使用

|  |
| --- |
| @CacheConfig(cacheNames = "userCache")  **public** **interface** UserMapper {  @Select("SELECT ID ,NAME,AGE FROM users where id=#{id}")  @Cacheable  List<Users> getUser(@Param("id") Long id);  } |

@Cacheable 加了该注解的方法表示可以缓存

@CacheConfig 表示创建缓存配置，Key为userCache

### EhCache配置

#### app1\_ehcache

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <ehcache xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:noNamespaceSchemaLocation=*"http://ehcache.org/ehcache.xsd"*>  <diskStore path=*"java.io.tmpdir/ehcache-rmi-4000"* />  <!-- 多台机器配置 rmiUrls=//192.168.8.32:400002/demoCache|//192.168.5.231:400003/demoCache -->  <cacheManagerPeerProviderFactory  class=*"net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"*  properties=*"peerDiscovery=manual,rmiUrls=//127.0.0.1:5000/userCache"*>  </cacheManagerPeerProviderFactory>  <!-- 配置 rmi 集群模式 -->  <cacheManagerPeerListenerFactory  class=*"net.sf.ehcache.distribution.RMICacheManagerPeerListenerFactory"*  properties=*"hostName=127.0.0.1,port=4000,socketTimeoutMillis=120000"* />  <!-- 多播方式配置 搜索某个网段上的缓存 timeToLive 0是限制在同一个服务器 1是限制在同一个子网 32是限制在同一个网站 64是限制在同一个region  128是限制在同一个大洲 255是不限制 <cacheManagerPeerProviderFactory class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"  properties="peerDiscovery=automatic, multicastGroupAddress=224.1.1.1, multicastGroupPort=40000,  timeToLive=32" /> -->  <!-- 默认缓存 -->  <defaultCache maxElementsInMemory=*"1000"* eternal=*"true"*  timeToIdleSeconds=*"120"* timeToLiveSeconds=*"120"* overflowToDisk=*"true"*  diskSpoolBufferSizeMB=*"30"* maxElementsOnDisk=*"10000000"*  diskPersistent=*"true"* diskExpiryThreadIntervalSeconds=*"120"*  memoryStoreEvictionPolicy=*"LRU"*>  </defaultCache>  <!-- demo缓存 -->  <cache name=*"userCache"* maxElementsInMemory=*"1000"* eternal=*"false"*  timeToIdleSeconds=*"120"* timeToLiveSeconds=*"120"* overflowToDisk=*"true"*  diskSpoolBufferSizeMB=*"30"* maxElementsOnDisk=*"10000000"*  diskPersistent=*"false"* diskExpiryThreadIntervalSeconds=*"120"*  memoryStoreEvictionPolicy=*"LRU"*>  <cacheEventListenerFactory  class=*"net.sf.ehcache.distribution.RMICacheReplicatorFactory"* />  <!-- 用于在初始化缓存，以及自动设置 -->  <bootstrapCacheLoaderFactory  class=*"net.sf.ehcache.distribution.RMIBootstrapCacheLoaderFactory"* />  </cache>  </ehcache> |

#### app2\_ehcache

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <ehcache xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:noNamespaceSchemaLocation=*"http://ehcache.org/ehcache.xsd"*>  <diskStore path=*"java.io.tmpdir/ehcache-rmi-5000"* />  <!-- 多台机器配置 -->  <cacheManagerPeerProviderFactory  class=*"net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"*  properties=*"peerDiscovery=manual,rmiUrls=//127.0.0.1:4000/userCache"*>  </cacheManagerPeerProviderFactory>  <cacheManagerPeerListenerFactory  class=*"net.sf.ehcache.distribution.RMICacheManagerPeerListenerFactory"*  properties=*"hostName=127.0.0.1,port=5000,socketTimeoutMillis=120000"* />  <!-- 多播方式配置 搜索某个网段上的缓存 timeToLive 0是限制在同一个服务器 1是限制在同一个子网 32是限制在同一个网站 64是限制在同一个region  128是限制在同一个大洲 255是不限制 <cacheManagerPeerProviderFactory class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"  properties="peerDiscovery=automatic, multicastGroupAddress=224.1.1.1, multicastGroupPort=40000,  timeToLive=32" /> -->  <!-- 默认缓存 -->  <defaultCache maxElementsInMemory=*"1000"* eternal=*"true"*  timeToIdleSeconds=*"120"* timeToLiveSeconds=*"120"* overflowToDisk=*"true"*  diskSpoolBufferSizeMB=*"30"* maxElementsOnDisk=*"10000000"*  diskPersistent=*"true"* diskExpiryThreadIntervalSeconds=*"120"*  memoryStoreEvictionPolicy=*"LRU"*>  </defaultCache>  <!-- demo缓存 -->  <cache name=*"userCache"* maxElementsInMemory=*"1000"* eternal=*"false"*  timeToIdleSeconds=*"120"* timeToLiveSeconds=*"120"* overflowToDisk=*"true"*  diskSpoolBufferSizeMB=*"30"* maxElementsOnDisk=*"10000000"*  diskPersistent=*"false"* diskExpiryThreadIntervalSeconds=*"120"*  memoryStoreEvictionPolicy=*"LRU"*>  <cacheEventListenerFactory  class=*"net.sf.ehcache.distribution.RMICacheReplicatorFactory"* />  <!-- 用于在初始化缓存，以及自动设置 -->  <bootstrapCacheLoaderFactory  class=*"net.sf.ehcache.distribution.RMIBootstrapCacheLoaderFactory"* />  </cache>  </ehcache> |

### 清除缓存

|  |
| --- |
| @Autowired  **private** CacheManager cacheManager;  @RequestMapping("/remoKey")  **public** **void** remoKey() {  cacheManager.getCache("userCache").clear();  } |

### 参数相关配置

1、diskStore ：指定数据(.data and .index)存储位置，可指定磁盘中的文件夹位置期 The diskStore element is optional. It must be configured if you have overflowToDisk or diskPersistent enabled for any cache. If it is not configured, a warning will be issues and java.io.tmpdir will be used.

２、defaultCache ： 默认的管理策略

Ehcache 使用Map集合实现的 element 其实就是 key 和value

一、以下属性是必须的：

　　1、name： Cache的名称，必须是唯一的(ehcache会把这个cache放到HashMap里)。

　　2、maxElementsInMemory：在内存中缓存的element的最大数目。

　　3、maxElementsOnDisk：在磁盘上缓存的element的最大数目，默认值为0，表示不限制。

　　４、eternal：设定缓存的elements是否永远不过期。如果为true，则缓存的数据始终有效，如果为false那么还要根据timeToIdleSeconds，timeToLiveSeconds判断。

　　５、overflowToDisk： 如果内存中数据超过内存限制，是否要缓存到磁盘上。

二、以下属性是可选的：

　　１、timeToIdleSeconds： 对象空闲时间，指对象在多长时间没有被访问就会失效。只对eternal为false的有效。默认值0，表示一直可以访问。

　　２、timeToLiveSeconds： 对象存活时间，指对象从创建到失效所需要的时间。只对eternal为false的有效。默认值0，表示一直可以访问。

　　３、diskPersistent： 是否在磁盘上持久化。指重启jvm后，数据是否有效。默认为false。

　　４、diskExpiryThreadIntervalSeconds： 对象检测线程运行时间间隔。标识对象状态的线程多长时间运行一次。

　　５、diskSpoolBufferSizeMB： DiskStore使用的磁盘大小，默认值30MB。每个cache使用各自的DiskStore。

　　６、memoryStoreEvictionPolicy： 如果内存中数据超过内存限制，向磁盘缓存时的策略。默认值LRU，可选FIFO、LFU。

## Ehcache集群模式

由于 EhCache 是进程中的缓存系统，一旦将应用部署在集群环境中，每一个节点维护各自的缓存数据，当某个节点对缓存数据进行更新，这些更新的数据无法在其它节点中共享，这不仅会降低节点运行的效率，而且会导致数据不同步的情况发生。例如某个网站采用 A、B 两个节点作为集群部署，当 A 节点的缓存更新后，而 B 节点缓存尚未更新就可能出现用户在浏览页面的时候，一会是更新后的数据，一会是尚未更新的数据，尽管我们也可以通过 Session Sticky 技术来将用户锁定在某个节点上，但对于一些交互性比较强或者是非 Web 方式的系统来说，Session Sticky 显然不太适合。

## 常用集群模式

EhCache从1.7版本开始，支持五种集群方案，分别是：

Terracotta、RMI、JMS、JGroups、EhCache Server

### RMi集群模式

你如何知道集群环境中的其他缓存？

• 分布式传送的消息是什么形式？

• 什么情况需要进行复制？增加（Puts），更新（Updates）或是失效（Expiries）？

• 采用什么方式进行复制？同步还是异步方式？

1、正确的元素类型：只有可序列化的元素可以进行复制。一些操作，比如移除，只需要元素的键值而不用整个元素；在这样的操作中即使元素不是可序列化的但键值是可序列化的也可以被复制。

2、成员发现（Peer Discovery）：Ehcache进行集群的时候有一个cache组的概念。每个cache都是其他cache的一个peer，没有主cache的存在。成员发现（Peer Discovery）正是用来解决 “你如何知道集群环境中的其他缓存?” 这个问题的。Ehcache提供了两种机制用来进行成员发现，即：自动成员发现和手动成员发现。要使用一个内置的成员发现机制要在ehcache的配置文件中指定cacheManagerPeerProviderFactory元素的class属性为

net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory。

## Ehcache的使用场景

使用纯java的ehcache作为本地缓存

Reids 作为远程分布式缓存

解决redis缓存压力过大，提高缓存速度，以及缓存性能。

## Redis和Ehcache缓存的区别

如果是单个应用或者对缓存访问要求很高的应用，用ehcache。  
如果是大型系统，存在缓存共享、分布式部署、缓存内容很大的，建议用redis。

## 实际工作中使用Ehcache

我们在项目中使用集中式缓存(Redis或者式Memcached等)，通常都是检查缓存中是否存在

期望值的数据，如果存在直接返回，如果不存在就查询数据库让后在将数据库缓存，

这个时候如果缓存系统因为某写原因宕机，造成服务无法访问，那么大的量请求直接穿透到数据库，最数据库压力非常大。

这时候我们让ehcache作为二级缓存，当redis服务器宕机后，可以查询ehcache缓存。

这样能够有效的扛住服务器请求压力。

# 分布式Redis缓存