



## 第1章 Redis从入门到高级实现课程介绍

### 第1集 Redis从入门到高级实现课程介绍

- 课件介绍
- 适学人员
- 学后水平



## 第2章 带你进入分布式缓存Redis的世界

### 第1集 Redis是什么，为什么大小公司都在用

简介：Redis 微信（微信红包）、微博、淘宝、天猫、京东、唯品会，温故而知新，从我们熟知的Mysql入手讲解Redis，为什么要用redis

- 讲到redis不得不讲nosql
  - NoSQL是不同于传统的关系数据库的数据库管理系统的统称。其两者最重要的区别是NoSQL不使用SQL作为查询语言。NoSQL数据存储可以不需要固定的表格模式。NoSQL是基于键值对的，可以想象成表中的主键和值的对应关系。
  - NoSQL：redis、memcached、mongodb、guava（loadingCache）
- redis的定义：
  - Redis 是一个开源（BSD许可）的，内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件。它支持多种类型的数据结构，如字符串（strings）、散列（hashes）、列表（lists）、集合（sets）、有序集合（sorted sets）等。
- 从大家熟知的mysql出发来认识redis
  - 概念

关系型数据库的一个常见用法是存储长期的报告数据，并将这些报告数据用作固定时间范围内的聚合数据。收集聚合数据的常见做法是：先将各个行插入一个报告表里面，之后再通过扫描这些行来收集聚合数据，并更新聚合表中已有的那些行。
  - 图解剖析mysql的执行过程

课程回顾：nosql的定义==》redis的定义==》mysql的定义==》redis的出现

## 第2集 剖析Redis和memcached和mysql之间的区别

简介：分析三者的区别和为什么越来越多的人抛弃memcached选择redis

- 从数据库类型、数据存储方式、特殊功能讲解Redis和memcached和mysql的区别
- 作为同款功能的内存缓存产品，redis和memcached各有什么优势
  - 内存管理机制
    - Memcached默认使用Slab Allocation机制管理内存，其主要思想是按照预先规定的大小，将分配的内存分割成特定长度的块以存储相应长度的key-value数据记录，以完全解决内存碎片问题。空闲列表进行判断存储状态，【类似于Java虚拟机对象的分配，空闲列表】
    - Redis使用现场申请内存的方式来存储数据，并且很少使用free-list等方式来优化内存分配，会在一定程度上存在内存碎片，【CPU内存是连续，类似于Java虚拟机对象的分配，直接内存分配（指针碰撞）】
  - 数据持久化方案
    - memcached不支持内存数据的持久化操作，所有的数据都以in-memory的形式存储。
    - redis支持持久化操作。redis提供了两种不同的持久化方法来讲数据存储到硬盘里面，第一种是rdb形式，一种是aof形式
      - rdb：属于全量数据备份，备份的是数据
      - aof：append only if,增量持久化备份，备份的是指令
  - 缓存数据过期机制
    - 概念：key，设计一个小时之后过期，超过一个小时查数据就会查不到
    - Memcached 在删除失效主键时也是采用的消极方法，即 Memcached 内部也不会监视主键是否失效，而是在通过 Get 访问主键时才会检查其是否已经失效
    - Redis 定时、定期等多种缓存失效机制，减少内存泄漏
  - 支持的数据类型
    - Memcached支持单一数据类型,[k,v]
    - redis支持五种数据类型

课程回顾：宏观讲解三种数据库的对比==》剖析memcached和redis区别==》还是用redis靠谱

## 第3集 redis作为数据库和作为缓存的选择，线上怎么优雅的使用redis

简介：redis作为数据库和作为内存缓存的两种使用方法

- redis作为数据库的使用有什么优缺点
  - 优点
    - 没有Scheme约束，数据结构的变更相对容易，一开始确定数据类型，抗压能力强，性能极高，10万/qps

- 缺点
  - 没有索引，没有外键，缺少int/date等基本数据类型，多条件查询需要通过集合内联(sinter,zinterstore) 和连接间接实现开发效率低，可维护性不佳
- redis作为缓存的使用，搭配数据库使用的两种方案
  - jedis整合使用方案 set key,value ["11","22"] 第一层在缓存进行查询，如果得到数据则直接返回，第二层在数据库进行查询，并且刷新缓存，方便下次查询 ["33","44"]
  - 作为mybatis/hibernate二级缓存使用方案，一级缓存：sqlSession，进程缓存，单次链接有效
- 图解分析加redis前后的架构区别

课程回顾：redis作为数据库？==》redis作为缓存==》使用redis前后架构上的区别



小·吊·课堂 愿景：“让编程不在难学，让技术与生活更加有趣” 更多教程请访问 [xdclass.net](http://xdclass.net)

## 第3章 手把手Redis安装启动

### 第1集 手把手mac环境下centos7安装过程（windows同理）

简介：redis环境准备之搭建centos7虚拟机

- 选择清华镜像：[http://mirrors.tuna.tsinghua.edu.cn/centos/7.5.1804/isos/x86\\_64/CentOS-7-x86\\_64-DVD-1804.iso](http://mirrors.tuna.tsinghua.edu.cn/centos/7.5.1804/isos/x86_64/CentOS-7-x86_64-DVD-1804.iso)

### 第2集 手把手centos7环境下redis4.0安装

- 解决pwd /could not retrieve mirrorlist
  - 1.sudo vim /etc/sysconfig/network-scripts/ifcfg-ens33
  - 2.将ONBOOT改为yes，wq!保存退出
  - 3.重新启动网络 \$ service network restart

- 安装wget yum install wget
- 下载redis安装包 wget <http://download.redis.io/releases/redis-4.0.6.tar.gz>
- 解压压缩包 tar -zxvf redis-4.0.6.tar.gz
- yum install gcc
- 跳转到redis解压目录下 cd redis-4.0.6
- 编译安装 make MALLOC=libc
- cd src ./redis-server

## 第3集 redis三种启动方式以及其中的使用区别

简介：Redis 微信（微信红包）、微博、淘宝、天猫、京东、唯品会，温故而知新，从我们熟知的Mysql入手讲解Redis，为什么要用redis

- 直接启动
- 通过指定配置文件启动
- 使用redis启动脚本设置开机自启动，linux配置开启自启动 /etc/init.d
- 配置步骤
  - 启动脚本 redis\_init\_script 位于Redis的 /utils/ 目录下
  - mkdir /etc/redis
  - cp redis.conf /etc/redis/6379.conf
  - 将启动脚本复制到/etc/init.d目录下，本例将启动脚本命名为redisd（通常都以d结尾表示是后台自启动服务）。

```
cp redis_init_script /etc/init.d/redisd
```

- 设置为开机自启动，直接配置开启自启动 chkconfig redisd on 发现错误：service redisd does not support chkconfig  
解决办法，在启动脚本开头添加如下注释来修改运行级别：

```
#!/bin/sh
# chkconfig: 2345 90 10
```
- 设置为开机自启动服务器

```
chkconfig redisd on
service redisd start 打开服务
service redisd stop 关闭服务
```

## 第4集 ssh的安装过程

- 首先，检查CentOS7是否安装了openssh-server，在终端中输入 yum list installed | grep openssh-server 此处显示已经安装了 openssh-server，如果又没有任何输出显示表示没有安装，通过输入yum install openssh-server 安装

- 找到了 /etc/ssh/ 目录下的sshd服务配置文件 sshd\_config，用Vim编辑器打开将文件中，关于监听端口、监听地址前的 # 号去除

```
Port 22
ListenAddress 0.0.0.0
ListenAddress ::
PermitRootLogin yes
PasswordAuthentication yes
```

- 开启sshd服务，输入 `sudo service sshd start`
- 检查 sshd 服务是否已经开启，输入 `ps -ef | grep sshd`
- 使用ip addr查看地址
- 为了免去每次开启 CentOS 时，都要手动开启 sshd 服务，将 sshd 服务添加至自启动列表中，输入 `systemctl enable sshd.service`
- 通过本机工具进行连接



小D课堂 愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 [xdclass.net](http://xdclass.net)

## 第4章 你不得不懂的Redis五种数据类型和消息订阅

### 第1集 Redis Key/Value数据类型

简介：String是最常用的一种数据类型，普通的key/value存储都可以归为此类。

- set/get
  - 设置key对应的值为String类型的value
  - 获取key对应的值
- mget
  - 批量获取多个key的值，如果不存在则返回nil
- incr && incrby
  - incr对key对应的值进行加1操作，并返回新的值;incrby加指定值
- incr && incrby
  - incr对key对应的值进行加1操作，并返回新的值;incrby加指定值
- setnx
  - 设置key对应的值为String类型的value，如果key已经存在则返回0
- setex

- 设置key对应的值为String类型的value，并设定有效期
- 其他命令
  - getrange 获取key对应value的子字符串
  - mset 批量设置多个key的值，如果成功表示所有值都被设置，否则返回0表示没有任何值被设置
  - msetnx，同mset，不存在就设置，不会覆盖已有的key
  - getset 设置key的值，并返回key旧的值
  - append：给指定key的value追加字符串，并返回新字符串的长度
- 课程回顾：redis String命令的使用实战，记住1、3、5、6这几个重要命令

## 第2集 手把手进行Hash类型讲解

- Hash是一个String类型的field和value之间的映射表
- redis的Hash数据类型的key（hash表名称）对应的value实际的内部存储结构为一个HashMap
- Hash特别适合存储对象
- 相对于把一个对象的每个属性存储为String类型，将整个对象存储在Hash类型中会占用更少内存。
- 所存储的成员较少时数据存储为zipmap，当成员数量增大时会自动转成真正的HashMap,此时encoding为ht。
- 运用场景：如用一个对象来存储用户信息，商品信息，订单信息等等。
- 
- Hash命令讲解
  - hset——设置key对应的HashMap中的field的value
  - hget——获取key对应的HashMap中的field的value
  - hgetall——获取key对应的HashMap中的所有field的value
  - hlen--返回key对应的HashMap中的field的数量

## 第3集 手把手进行List类型讲解

简介：List的命令讲解以及使用场景剖析

- lpush——在key对应的list的头部添加一个元素
- lrange——获取key对应的list的指定下标范围的元素，-1表示获取所有元素
- lpop——从key对应的list的尾部删除一个元素，并返回该元素
- rpush——在key对应的list的尾部添加一个元素
- rpop——从key对应的list的尾部删除一个元素，并返回该元素

## 第4集 手把手进行Set类型讲解

简介：Set的命令讲解以及使用场景剖析

- sadd——在key对应的set中添加一个元素
- smembers——获取key对应的set的所有元素
- spop——随机返回并删除key对应的set中的一个元素
- suion——求给定key对应的set并集

- sinter——求给定key对应的set交集

## 第5集 手把手进行SortSet类型讲解

简介：set的基础增加顺序score，再根据score进行排序 实战：通过sortset实现排行榜

- zadd ——在key对应的zset中添加一个元素
- zrange——获取key对应的zset中指定范围的元素，-1表示获取所有元素
- zrem——删除key对应的zset中的一个元素
- zrangebyscore——返回有序集key中，指定分数范围的元素列表,排行榜中运用
- zrank——返回key对应的zset中指定member的排名。其中member按score值递增(从小到大)；排名以0为底，也就是说，score值最小的成员排名为0,排行榜中运用
- set是通过hashmap存储，key对应set的元素，value是空对象 sortset是怎么存储并实现排序的呢，hashmap存储，还加了一层跳跃表 跳跃表：相当于双向链表，在其基础上添加前往比当前元素大的跳转链接
  - Redis 是一个开源（BSD许可）的，内存中的数据结构存储系统，它可以用作数据库、缓存和消息中间件。它支持多种类型的数据结构，如字符串（strings）、散列（hashes）、列表（lists）、集合（sets）、有序集合（sorted sets）等。

## 第6集 redis消息订阅发布

简介：redis消息订阅发布讲解，基础使用

- 作用：发布订阅类似于信息管道，用来进行系统之间消息解耦，类似于mq，rabbitmq、rocketmq、kafka、activemq主要有消息发布者和消息订阅者。比如运用于：订单支付成功，会员系统加积分、钱包进行扣钱操作、发货系统（下发商品）
- PUBLISH 将信息message发送到指定的频道channel。返回收到消息的客户端数量
- SUBSCRIBE 订阅给指定频道的信息
- UNSUBSCRIBE 取消订阅指定的频道，如果不指定，则取消订阅所有的频道。
- redis的消息订阅发布和mq对比？答：redis发布订阅功能比较薄弱但比较轻量级，mq消息持久化，数据可靠性比较差，无后台功能可msgId、msgKey进行查询消息



小D课堂 愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 [xdclass.net](http://xdclass.net)

---

## 第5章 传统关系型数据库事务与Redis事务

# 第1集 深入浅出剖析传统关系型数据库事务

简介：通过类比法进行学习可以增强我们的知识掌握程度，讲解事务概要和事务隔离级别

- 一个数据库事务通常包含了一个序列的对数据库的读/写操作。它的存在包含有以下两个目的：
  - 为数据库操作序列提供了一个从失败中恢复到正常状态的方法，同时提供了数据库即使在异常状态下仍能保持一致性的方法。
  - 当多个应用程序在并发访问数据库时，可以在这些应用程序之间提供一个隔离方法，以防止彼此的操作互相干扰。
- 事务的ACID四大特性
  - 原子性（Atomicity）：事务作为一个整体被执行，包含在其中的对数据库的操作要么全部被执行，要么都不执行
  - 一致性（Consistency）：事务应确保数据库的状态从一个一致状态转变为另一个一致状态。一致状态的含义是数据库中的数据应满足完整性约束
  - 隔离性（Isolation）：多个事务并发执行时，一个事务的执行不应影响其他事务的执行
  - 持久性（Durability）：已被提交的事务对数据库的修改应该永久保存在数据库中
- 事务隔离机制
  - 语法：set global transaction isolation level read uncommitted;
  - 种类：read uncommitted、read committed、repeatable read、serializable

## 第2集 浅谈mysql事务隔离机制和MVCC

###

- redis事务隔离机制可重复读讲解（repeatable read）
- InnoDB MVCC多版本并发控制功能讲解
  - 在每一行数据中额外保存两个隐藏的列：当前行创建时的版本号和删除时的版本号（可能为空，其实还有一列称为回滚指针，用于事务回滚，不在本文范畴）。这里的版本号并不是实际的时间值，而是系统版本号。每开始新的事务，系统版本号都会自动递增。事务开始时刻的系统版本号会作为事务的版本号，用来和查询每行记录的版本号进行比较
- 图解InnoDB MVCC的组成和原理

## 第3集 redis事务机制

简介：讲解redis事务基本命令，分析redis事务的基本原理

- MULTI 与 EXEC命令
  - 以 MULTI 开始一个事务，然后将多个命令入队到事务中，最后由 EXEC 命令触发事务，一并执行事务中的所有命令



- DISCARD命令
  - DISCARD 命令用于取消一个事务，它清空客户端的整个事务队列，然后将客户端从事务状态调整回非事务状态，最后返回字符串 OK 给客户端，说明事务已被取消。
- WATCH命令
  - WATCH 命令用于在事务开始之前监视任意数量的键：当调用 EXEC 命令执行事务时，如果任意一个被监视的键已经被其他客户端修改了，那么整个事务不再执行，直接返回失败。
- 图解redis执行事务过程原理

## 第4集 redis事务与传统关系型事务的比较

简介：讲解redis事务ACID

- 原子性 ( Atomicity )
  - 单个 Redis 命令的执行是原子性的，但 Redis 没有在事务上增加任何维持原子性的机制，所以 Redis 事务的执行并不是原子性的。如果一个事务队列中的所有命令都被成功地执行，那么称这个事务执行成功
- 一致性 ( Consistency )
  - 入队错误
    - 在命令入队的过程中，如果客户端向服务器发送了错误的命令，比如命令的参数数量不对，等等，那么服务器将向客户端返回一个出错信息，并且将客户端的事务状态设为 REDIS\_DIRTY\_EXEC 。
  - 执行错误
    - 如果命令在事务执行的过程中发生错误，比如说，对一个不同类型的 key 执行了错误的操作，那么 Redis 只会将错误包含在事务的结果中，这不会引起事务中断或整个失败，不会影响已执行事务命令的结果，也不会影响后面要执行的事务命令，所以它对事务的一致性也没有影响
  - 隔离性 ( Isolation )
    - WATCH 命令用于在事务开始之前监视任意数量的键：当调用 EXEC 命令执行事务时，如果任意一个被监视的键已经被其他客户端修改了，那么整个事务不再执行，直接返回失败
  - 持久性 ( Durability )
    - 因为事务不过是用队列包裹起了一组 Redis 命令，并没有提供任何额外的持久性功能，所以事务的持久性由 Redis 所使用的持久化模式决定



小D课堂 愿景：“让编程不在难学，让技术与生活更加有趣” 更多教程请访问 [xdclass.net](http://xdclass.net)

## 第6章 Redis从入门到高级实现课程介绍

### 第1集 springboot项目搭建并编写HelloWorld

简介：代码实战springboot项目搭建过程

- 下载springboot项目
- 修改配置文件，springboot需要引入：spring-boot-starter-web
- 启动WEB项目
- 测试HelloWorld
- springboot两种启动方式
  - 以web启动的项目（war）：比如：用tomcat容器启动项目，提供http接口
  - 以jar启动的项目：比如：用java -jar启动 main方法启动，只比如用dubbo提供对外

## 第2集 springboot结合redis实战

### 简介：讲解redis事务ACID

- 引入bean redisTemplate的使用，类型于：monogoTemplate、jdbcTemplate数据库连接工具
- 配置步骤
  - 1)引入pom依赖  
org.springframework.boot spring-boot-starter-data-redis
  - 2) 编写redisTemplate类，设置redisConnectionFactory
  - 3) 引入配置文件

## 第3集 redisTemplate api讲解

### 简介：redisTemplate源码讲解

- opsForValue 操作String，Key，Value，包含过期key，setBit位操作等
- opsForSet 操作set
- opsForHash 操作hash
- opsForZset 操作SortSet
- opsForList 操作list队列
- opsForHash 操作hash opsForZset 操作SortSet opsForList 操作list队列

## 第4集 springboot和mybatis步骤整合讲解

### 简介：redis比mysql性能好，那么到底好在哪里呢？

- ◦ 1、导入maven依赖pom.xml

```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>1.3.0</version>
</dependency>
```

```

<dependency>
  <groupId>org.mybatis.generator</groupId>
  <artifactId>mybatis-generator-core</artifactId>
  <scope>test</scope>
  <version>1.3.2</version>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid</artifactId>
  <version>1.1.10</version>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.7</version>
</dependency>

```

- 引入配置文件
  - Druid是阿里旗下的数据库连接池，同款产品有C3P0
- 引入mybatis-config.xml
- 引入DataSourceConfig配置

## 第5集 redis作为mybatis缓存整合讲解（上）

简介：redisTemplate整合mybatis

- 用户第一次访问的时候获取数据库的值，再次访问时直接从缓存中获取数据
- 设置缓存过期时间
- 项目8080端口是对外端口（向外部暴露的端口），区别于内部进程号,查内部端口用ps -ef|grep port，查外部端口用ls -l | grep -i:port

## 第6集 redis作为mybatis二级缓存整合讲解（中）

简介：整合讲解

- springboot cache的使用：可以结合redis、ehcache等缓存 一级缓存是：sqlSession，sql建立连接到关闭连接的数据缓存 二级缓存是：全局
  - @CacheConfig(cacheNames="userInfoCache") 在同个redis里面必须唯一
  - @Cacheable(查)：来划分可缓存的方法 - 即，结果存储在缓存中的方法，以便在后续调用（具有相同的参数）时，返回缓存中的值而不必实际执行该方法
  - @CachePut（修改、增加）：当需要更新缓存而不干扰方法执行时，可以使用@CachePut注释。也就是说，始终执行该方法并将其结果放入缓存中（根据@CachePut选项）
  - @CacheEvict（删除）：对于从缓存中删除陈旧或未使用的数据非常有用，指示缓存范围内的驱逐是否需要执行而不仅仅是一个条目驱逐
- springboot cache的整合步骤

- 引入pom.xml依赖  
org.springframework.boot spring-boot-starter-cache
- 开启缓存注解：@EnableCaching
- 在方法上面加入SpEL
- springboot cache 存在什么问题
  - 第一，生成key过于简单，容易冲突userCache::3
  - 第二，无法设置过期时间，默认过期时间为永久不过期
  - 第三，配置序列化方式，默认的是序列化JDKSerialazable
- springboot cache自定义项
  - 自定义KeyGenerator
  - 自定义cacheManager，设置缓存过期时间
  - 自定义序列化方式，Jackson

## 第7集 redis作为mybatis缓存整合讲解（下）

简介：通过类比法进行学习可以增强我们的知识掌握程度，讲解事务概要和事务隔离级别

- 一个数据库事务通常包含了一个序列的对数据库的读/写操作。它的存在包含有以下两个目的：
  - 为数据库操作序列提供了一个从失败中恢复到正常状态的方法，同时提供了数据库即使在异常状态下仍能保持一致性的方法。
  - 当多个应用程序在并发访问数据库时，可以在这些应用程序之间提供一个隔离方法，以防止彼此的操作互相干扰。
- 事务的ACID四大特性
  - 原子性（Atomicity）：事务作为一个整体被执行，包含在其中的对数据库的操作要么全部被执行，要么都不执行
  - 一致性（Consistency）：事务应确保数据库的状态从一个一致状态转变为另一个一致状态。一致状态的含义是数据库中的数据应满足完整性约束
  - 隔离性（Isolation）：多个事务并发执行时，一个事务的执行不应影响其他事务的执行
  - 持久性（Durability）：已被提交的事务对数据库的修改应该永久保存在数据库中
  -
- 事务隔离机制
  - 语法：set global transaction isolation level read uncommitted;
  - 种类：read uncommitted、read committed、repeatable read、serializable

## 第8集 redis作为mybatis缓存整合讲解（下）

简介：压力测试redis缓存和数据库的对比级别

- apache abtest
  - ab是Apache HTTP server benchmarking tool，可以用以测试HTTP请求的服务器性能

- 使用：ab -n1000 -c10 <http://localhost:8080/getByCache?id=2> ab -n1000 -c10 <http://localhost:8080/getUser?id=2>
  - -n:进行http请求的总个数
  - -c:请求的client个数，也就是请求并发数
  - 统计qps：qps即每秒并发数，request per second
    - 统计

10个并发的情况下

redis qps: 963.85 [#/sec] (mean)

DB qps: 766.75 [#/sec] (mean)

100个并发的情况下 1000个

redis qps: 1130.60 [#/sec] (mean)

DB qps: 956.15 [#/sec] (mean)

100个并发的情况下，进行10000个请求

redis qps: 2102.39 [#/sec] (mean)

DB qps: 679.07 [#/sec] (mean)

500个并发的情况下，进行10000个请求

redis qps: 374.91 [#/sec] (mean)

DB qps: 扛不住

## 第9集 redis实现分布式集群环境session共享

**\*\* 简介：多机器部署同一套服务（代码），性能更好，更承受更高的用户并发**

- cookie与session
  - Cookie是什么？Cookie 是一小段文本信息，伴随着用户请求和页面在 Web 服务器和浏览器之间传递。Cookie 包含每次用户访问站点时 Web 应用程序都可以读取的信息，我们可以看到在服务器写的cookie，会通过响应头Set-Cookie的方式写入到浏览器
  - HTTP协议是无状态的，并非TCP一样进行三次握手，对于一个浏览器发出的多次请求，WEB服务器无法区分是不是来源于同一个浏览器。所以服务器为了区分这个过程会通过一个 sessionid来区分请求，而这个sessionid是怎么发送给服务端的呢。cookie相对用户是不可见的，用来保存这个sessionid是最好不过了
  - Cookie是什么？Cookie 是一小段文本信息，伴随着用户请求和页面在 Web 服务器和浏览器之间传递。Cookie 包含每次用户访问站点时
  - Web 应用程序都可以读取的信息，我们可以看到在服务器写的cookie，会通过响应头Set-Cookie的方式写入到浏览器
- redis实现分布式集群配置过程
  - org.springframework.session spring-session-data-redis
  - @EnableRedisHttpSession 开启redis session缓存
  - maxInactiveIntervalInSeconds指定缓存的时间 spring:session:sessions:expires:+‘sessionId’的过期时间
- 验证过程
  - 打开隐身模式清空cookie来验证缓存的时间



## 第7章 redis项目实战之排行榜实现

### 第1集 ZSetOperations重要api讲解 ( sortedSet )

简介：通过类比法进行学习可以增强我们的知识掌握程度，讲解事务概要和事务隔离级别

- 排行榜：
  - 排行榜功能是一个很普遍的需求。使用 Redis 中有序集合的特性来实现排行榜是又好又快的选择。一般排行榜都是有实效性的，比如“用户积分榜”，游戏中活跃度排行榜，游戏装备排行榜等。
  - 面临问题：数据库设计复杂，并发数较高，数据要求实时性高
- redis实现排行榜api讲解

### 第2集 浅谈mysql数据库表设计过程中几个关键点

\* 简介：数据库表score\_flow ( 积分流水表 )、user\_score ( 用户积分表总表 ) 设计，用于：1 ) 查top100 2 ) 查用户的排名

- 表设计过程中应该注意的点即数据类型
    - 1 ) 更小的通常更好 控制字节长度
    - 2 ) 使用合适的数据类型：如tinyint只占8个位，char(1024)与varchar(1024)的对比,char用于类似定长数据存储比varchar节省空间，如：uuid ( 32 )，可以用char(32).
    - 3 ) 尽量避免NULL建议使用NOT NULL DEFAULT "
    - 4 ) NULL的列会让索引统计和值比较都更复杂。可为NULL的列会占据更多的磁盘空间，在Mysql中也需要更多复杂的处理程序
  - 索引设计过程中应该注意的点
    - 选择唯一性索引 唯一性索引的值是唯一的，可以更快速的通过该索引来确定某条记录,保证物理上面唯一
    - 为经常需要排序、分组和联合操作的字段建立索引，经常需要ORDER BY、GROUP BY、DISTINCT 和UNION等操作的字段，排序操作会浪费很多时间
    - 常作为查询条件的字段建立索引 如果某个字段经常用来做查询条件，那么该字段的查询速度会影响整个表的查询速度
    - 数据少的地方不必建立索引
- org.mybatis.generator配置讲解 引入：

<dependency>

```
<groupId>org.mybatis.generator</groupId>
<artifactId>mybatis-generator-core</artifactId>
<scope>test</scope>
<version>1.3.2</version>
<optional>true</optional>
</dependency>

<dependency>
<groupId>commons-io</groupId>
<artifactId>commons-io</artifactId>
<version>2.5</version>
</dependency>
```

## 第3集 排行榜三大接口讲解

**\*\* 简介：**多机器部署同一套服务（代码），性能更好，更承受更高的用户并发

- 添加用户积分
- 获取top N 排行
  - redisService新增方法reverseRangeWithScores()
- 根据用户ID获取排行
  - zset.rank(key,value)，key为set的名称，value为用户id

## 第4集 springboot项目初始化加载讲解

**\* 场景：**将一千万用户白名单load缓存，用户请求的时候判断该用户是否是缓存里面的用户

- springboot实现初始化加载配置（实现缓存预热）
  - 采用实现springboot ApplicationRunner 该方法仅在SpringApplication.run(...)完成之前调用
  - 采用实现InitializingBean  
InitializingBean接口为bean提供了初始化方法的方式，它只包括afterPropertiesSet()方法。在spring初始化bean的时候，如果bean实现了InitializingBean接口，在对象的所有属性被初始化后之后才会调用afterPropertiesSet()方法
- 初始化同步redis数据
- 初始化完成再放入请求



# 第8章 2018支付宝蚂蚁金服Redis面试题分析

## 第1集 缓存的收益和成本

### 支付宝面试题讲解

- 缓存带来的回报
  - 高速读写
    - 缓存加速读写速度：CPU L1/L2/L3 Cache、Linux page Cache加速硬盘读写、浏览器缓存、Ehcache缓存数据库结果
  - 降低后端负载
    - 后端服务器通过前端缓存降低负载：业务端使用Redis降低后端MySQL负载等
- 缓存带来的代价
  - 数据不一致
    - 缓存层和数据层有时间窗口不一致，和更新策略有关
  - 代码维护成本
    - 原本只需要读写MySQL就能实现功能，但加入了缓存之后就要去维护缓存的数据，增加了代码复杂度。
  - 堆内缓存可能带来内存溢出的风险影响用户进程，如ehCache、loadingCache
  - 堆内缓存和远程服务器缓存redis的选择
    - 堆内缓存一般性能更好，远程缓存需要套接字传输
    - 用户级别缓存尽量采用远程缓存
    - 大数据量尽量采用远程缓存，服务节点化原则

## 第2集 2018支付宝面试题之缓存雪崩

### \*\* 简介：支付宝面试题讲解

- 什么是缓存雪崩？你有什么解决方案来防止缓存雪崩？
  - 如果缓存集中在一段时间内失效，发生大量的缓存穿透，所有的查询都落在数据库上，造成了缓存雪崩。由于原有缓存失效，新缓存未到期间所有原本应该访问缓存的请求都去查询数据库了，而对数据库CPU 和内存造成巨大压力，严重的会造成数据库宕机
  - 你有什么解决方案来防止缓存雪崩？
    - 加锁排队
      - key：whiltList value：1000w个uid 指定setNx whiltList value nullValue mutex互斥锁解决，Redis的SETNX去set一个mutex key，当操作返回成功时，再进行load db的操作并回设缓存；否则，就重试整个get缓存的方法
    - 数据预热
      - 缓存预热就是系统上线后，将相关的缓存数据直接加载到缓存系统。这样就可以避免在用户请求的时候，先查询数据库，然后再将数据缓存的问题!用户直接查询事先被预热的缓存数据!可以通过缓存reload机制，预先去更新缓存，再即将发生大并发访问前手动触发加载缓存不同的key
    - 双层缓存策略



- C1为原始缓存，C2为拷贝缓存，C1失效时，可以访问C2，C1缓存失效时间设置为短期，C2设置为长期。
- 定时更新缓存策略
  - 失效性要求不高的缓存，容器启动初始化加载，采用定时任务更新或移除缓存
- 设置不同的过期时间，让缓存失效的时间点尽量均匀

## 第3集 2018支付宝面试题之缓存穿透

### 简介：支付宝面试题讲解

- 什么是缓存穿透？你有什么解决方案来防止缓存穿透？
  - 缓存穿透是指用户查询数据，在数据库没有，自然在缓存中也不会有。这样就导致用户查询的时候，在缓存中找不到对应key的value，每次都要去数据库再查询一遍，然后返回空(相当于进行了两次 无用的查询)。这样请求就绕过缓存直接查数据库
- 你有什么解决方案来防止缓存穿透？
  - 采用布隆过滤器BloomFilter
    - 将有可能存在的数据哈希到一个足够大的 bitmap 中，一个一定不存在的数据会被这个 bitmap 拦截掉，从而避免了对底层存储系统的查询压力
  - 缓存空值
    - 如果一个查询返回的数据为空(不管是数据不存在，还是系统故障)我们仍然把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。通过这个直接设置的默认值存放到缓存，这样第二次到缓存中获取就有值了，而不会继续访问数据库

**小D课堂，愿景：让编程不在难学，让技术与生活更加有趣**

**相信我们，这个是可以让你学习更加轻松的平台，里面的课程绝对会让你技术不断提升**

**欢迎加小D讲师的微信：jack794666918**

我们官方网站：<https://xdclass.net>

**千人IT技术交流QQ群：718617859**

**重点来啦：免费赠送你干货文档大集合，包含前端，后端，测试，大数据，运维主流技术文档（持续更新）**

<https://mp.weixin.qq.com/s/qYnjcDYGFQorWmSfE7lpQ>