



Artigo

Invista em você! Saiba como a DevMedia pode ajudar sua carreira.



# Java Redis: Utilizando o Redis com coleções Java

O Redis é um banco NOSQL de chave valor de arquitetura distribuída feito para memória. O objetivo desse artigo é falar sobre esse banco e mostrar como implementar Coleções no Java.

Marcar como lido



Anotar



Artigos



Java



Java Redis: Utilizando o Redis com coleções Java

O **Redis** é um banco de dados NoSQL escrito em C. O Remote Dictionary Server é

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar

Mas qual a diferença entre ele o cache? O que acontece quando o banco cai? Haverá perda total dos dados? O objetivo desse artigo é falar um pouco desse



open source, o redis-collections.





Considerando que as ferramentas de caches, como memcache e infinitspam, também possui esse comportamento de chave-valor, a primeira dúvida normal de um desenvolvedor é saber qual a diferença entre os dois. A primeira diferença está na serialização dos objetos valores: enquanto os caches escritos em **Java** são serializados de forma binária, com o Kryo ou o `java.io.Serializable`, por exemplo, para o redis tudo é texto. Um problema da serialização binária é a impossibilidade de realizar alterações de forma manual como em um texto. Ao usar o Kryo ou o `java.io.Serializable`, caso haja uma alteração significativa dentro da estrutura do objeto, não será mais possível deserializar tal estrutura no novo objeto, ou seja, ao retornar para a estrutura anterior ou descartar as informações. A razão disso é que os binários tendem a ser mais rápido para leitura e escrita e mais compactos, e os caches não são feitos para perdurar por longo período de tempo, A grande maioria dura enquanto o programa está rodando.

**Nota:** Entre de uma vez por todas no mercado de desenvolvimento em Java com os Cursos DevMedia.

Outra diferença está no tipo de valores: enquanto o cache trata apenas chave e valor, sendo que o valor é um objeto, no redis existem diferentes estruturas de

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar

- **List:** uma coleção de String ordenado de acordo com a ordem de inserção, semelhante a coleção `java.util.LinkedList`;
- 
- **Sorted Set:** similar ao set, por não permitir valores duplicados, mas cada elemento está associado para um valor flutuante, chamado de score. Esses elementos são sempre ordenados por esse campo score;
  - **Hashset:** Composto de campos e seus respectivos valores (ambos são String), semelhante ao Map.

Além desses que são os mais conhecidos, existem também o Bit arrays e HyperLog que não serão abordados nesse artigo.

Uma vez explicando as diferenças básicas entre o Redis e um cache, o próximo passo será a instalação do banco que é bastante simples:

1. Realizar o Download aqui;
2. Descompactar , abrir no terminal do redis e em seguida compilar o redis, conforme o código abaixo:

```
1 | tar zxvf redis-versao.x.tar.gz
2 | cd redis-versao.x
3 | make
```

3. Uma vez compilado, entrar na pasta `src` e executar o servidor, conforme o código abaixo:

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar

Pronto, o Redis está rodando. Para realizar alguns testes, basta executar o cliente que já vem com o ele. Para isso, basta ir em `REDIS_HOME/src` e em seguida



Para saber mais sobre os comandos, acesse o [Site da redis](#)

Uma vez instalado e testado, o nosso objetivo agora será a utilização de algumas estruturas de dados em cima do redis. Nosso objetivo será a utilização e a implementação de `java.util.List`, `java.util.Set`, `java.util.Queue` e `java.util.Map`, além de mais três estruturas: o conceito de chave e valor (semelhante ao cache), uma para contador e outra para fazer Ranking, `sorted Set`.

Ao idealizar a API, vamos ter o mesmo código da **Listagem 1**.

```
1 public interface keyValueRedisStructure<T> {
2
3     T get(String key);
4
5     void set(String key, T bean);
6
7     List<T> multiplesGet(Iterable<String> keys);
8
9     void delete(String key);
10 }
11
12
13 public interface CountStructure<T extends Number> {
14
15     T get();
16
17     T increment();
18
19     T increment(T count);
20
21     T decrement();
```

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar

```

28     void persist();
29 }
30
31 public interface ListStructure <T> extends Expirable {
32     List<T> get(String key);
33
34     void delete(String key);
35 }
36
37 public interface MapStructure <T> extends Expirable{
38     Map<String, T> get(String key);
39
40     void delete(String key);
41 }
42
43 public interface QueueStructure <T> extends Expirable {
44     Queue<T> get(String key);
45
46     void delete(String key);
47 }
48
49 public interface RankingStructure<T extends Number> extends Expirable {
50     ScoresPoint<T> create(String key);
51
52     void delete(String key);
53 }
54
55
56
57
58
59
60
61
62
63
64

```

### Listagem 1. Definição das estruturas

Uma vez vendo a estrutura, uma das dúvidas é o Expirable, devido a ideia em

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar

mantidas. Como dito anteriormente, o redis trata como a chave e os valores como String. Dessa forma, para serializar e deserializar o objetivo precisar ser em texto.



leitura e escrita de JSON do Google - o GSON, mas poderia ser qualquer outra forma, por exemplo, o XML tem campos separados por pipe “|” etc. E para se comunicar com o redis será utilizado o Jedis.

## Porque java.util?

As coleções dentro do `java.util` certamente são conhecidas pela grande maioria dos desenvolvedores Java. Dessa forma, não existiria dificuldade destes para a utilizarem.

A grande diferença entre as implementações dentro do JDK e a dessas coleções é que as últimas serão implementadas no modo lazy. Enquanto as implementações de List, como `ArrayList` e `LinkedList`, já possuem as informações, por exemplo, o `RedisList` usará o Jedis, a API de comunicação com o redis para realizar uma operação (seja inserir, verificar o tamanho ou retornar os valores dentro do redis). Desse é dito que um `RedisList` é igual ao outro se ele tiver a mesma chave.

## Convenção do namespace

Por convenção, o redis utiliza o conceito de `namespace` que funciona como um prefixo da chave. O seu formato é: `namespace:chave`.

Por exemplo, para registrar informações do usuário na seção seria `users:nickname`.

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar

O Jedis é o client para o redis feito para o Java e com ele é possível realizar todas as operações em cima das chaves. O seu uso é bastante simples, conforme mostra



```
1 Jedis jedis = new Jedis("localhost");  
2 jedis.set("foo", "bar");  
3 String value = jedis.get("foo");
```

No exemplo acima foi criada uma conexão para o redis na nossa máquina localhost e, em seguida, foram feitas duas operações simples: Setar o bar dentro da chave foo e retornar o valor dentro da chave “foo”.

O Redis possui uma arquitetura distribuída, ou seja, pode-se trabalhar com clusters. Para fazer o client com nós dentro do Jedis, é necessário utilizar o comando da **Listagem 3**.

```
1 Set<HostAndPort> jedisClusterNodes = new HashSet<HostAndPort>();  
2 //O jedis cluster irá descobrir os outros nós automaticamente.  
3 jedisClusterNodes.add(new HostAndPort("127.0.0.1", 7379));  
4 JedisCluster jc = new JedisCluster(jedisClusterNodes);  
5 jc.set("foo", "bar");  
6 String value = jc.get("foo");
```

### Listagem 3. Client do Jedis

## Criando as estruturas

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar

a nossa comunicação entre a nossa aplicação e o Redis. Acompanhe o código da

#### Listagem 4.



```
1 ListStructure<ProductCart> shippingCart = RedisStrutureBuilder.ofList(jedis,
2 ProductCart.class).withNameSpace("list_producs").build();
3 List<ProductCart> fruitsCarts = shippingCart.get(FRUITES);
4 fruitsCarts.add(banana);
5 ProductCart banana = fruitsCarts.get(0);
6
7
8 User otaviojava = new User("otaviojava");
9 User felipe = new User("ffrancesquini");
10 SetStructure<User> socialMediaUsers = RedisStrutureBuilder.ofSet(RedisConnection
11 User.class).withNameSpace("socialMedia").build();
12 Set<User> users = socialMediaUsers.createSet("twitter");
13 users.add(otaviojava);
14 users.add(otaviojava);
15 users.add(felipe);
16 users.add(otaviojava);
17 users.add(felipe);
18 //haverá apenas um objeto otaviojava e um felipe, já que ele impede a
19 duplicação da lista
20 //lembrando que os métodos de equals e hascode dos objetos não serão
21 utilizados, o que será considerado é a String do objeto gerado.
22
23
24 Species mammals = new Species("lion", "cow", "dog");
25 Species fishes = new Species("redfish", "glassfish");
26 Species amphibians = new Species("crocodile", "frog");
27
28 MapStructure<Species> zoo = =
29 RedisStrutureBuilder.ofMap(RedisConnection.JEDIS, Species.class)
30 .withNameSpace("animalZoo").build();
31
32 Map<String, Species> vertebrates = zoo.get("vertebrates");
33 vertebrates.put("mammals", mammals);
34 vertebrates.put("mammals", mammals);
35 vertebrates.put("fishes", fishes);
36 vertebrates.put("amphibians", amphibians);
37
38 QueueStructure<LineBank> serviceBank =
```

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar



```
45 Queue<LineBank> lineBank = serviceBank.get("createAccount");  
46 lineBank.add(new LineBank("Otavio", 25));  
47 LineBank otavio = lineBank.poll();
```



#### Listagem 4. RedisStrutureBuilder

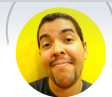
Com isso, foi discutido nesse artigo um pouco mais sobre o redis e um exemplo prático utilizando implementações das coleções Java para o redis. Salientamos a diferença entre o redis o cache. Os caches apresentados acima, por usarem formato binário e usarem o Kryo ou o `java.io.Serializable` acabam sendo mais compactos e rápidos na leitura e escrita. No entanto, uma grande mudança no objeto acaba impossibilitando a recuperação das informações antes da mudança, sendo imprescindível a eliminação das informações anteriores, afinal. É esse o objetivo do cache: um bloco de memória de acesso rápido de armazenamento temporário. Já o redis é um banco de dados chave valor cujo o armazenamento fica na memória. No entanto, de tempos em tempos é feito um `autobackup` das informações no disco com o intuito de, caso aconteça uma queda, não haja perdas das informações. E o outro fator é que o redis está no seu armazenamento utilizando String tanto na chave como no valor e possui algumas estruturas na parte do valor.

#### Tecnologias:

[Java](#)[NoSQL](#)[Redis](#)

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar



Por Otávio

Em 2014



## RECEBA NOSSAS NOVIDADES

Informe o seu e-mail

Receber Newsletter

Suporte ao aluno - Deixe a sua dúvida.

## FAÇA PARTE DESSE TIME

Faça parte dessa comunidade 100% focada em programação e tenha acesso ilimitado. Nosso compromisso é tornar a sua experiência de estudo cada vez mais dinâmica e eficiente. Portanto, se você quer programar de verdade seu lugar é aqui. Junte-se a mais de...

Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar

# GRÁTIS



Séries

Projetos completos

Cursos

Guias de carreiras

DevCasts

Desafios

Artigos

App

Suporte em tempo real

Cadastre-se



**Plataforma para Programadores**

Revistas

Fale conosco

Trabalhe conosco

Assinatura para empresas



Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar



Utilizamos cookies para fornecer uma melhor experiência para nossos usuários. Para saber mais sobre o uso de cookies, consulte nossa política de privacidade. Ao continuar navegando em nosso site, você concorda com a nossa política.

Aceitar