Hướng dẫn kết nối Redis sử dụng Redisson trong Java

1. Giới thiệu

Redisson là một Redis client trong Java. Trong bài viết này, chúng ta sẽ tìm hiểu một số tính năng và demo cách xây dựng một ứng dụng sử dụng Redisson.

2. Dependency

Khai báo dependency trong maven pom.xml

```
<dependency>
    <groupId>org.redisson</groupId>
    <artifactId>redisson</artifactId>
    <version>3.3.0</version>
</dependency>
```

3. Cấu hình

Chúng ta cần cấu hình Redisson để kết nối đến Redis. Redisson hỗ trợ các kết nối đến Redis như sau:

- Single node (Kết nối đến Redis dạng singleton) - Get single node settings here
- Master with slave nodes (Kết nối đến Redis dạng master-slave) - Get master-slave node settings here

- Sentinel nodes (Kết nối đến Redis kiểu sentinel) - Get sentinel node settings here
- Clustered nodes (Kết nối đến Redis cluster) - Get clustered node settings here
- Replicated nodes (Kết nối đến Redis replicated) - Get replicated node settings here

3.1. Java Configuration - Khai báo cấu hình trực tiếp trong Java

Chúng ta có thể cấu hình để tạo kết nối đến Redis trực tiếp trong Java như sau:

```
Config config = new Config();
config.useSingleServer()
  .setAddress("127.0.0.1:6379");

RedissonClient client = Redisson.create(config);
```

3.2. File Configuration - Khai báo cấu hình trên file

Redisson hỗ trợ load cấu hình từ file định dạng JSON hoặc YAML.

```
Config config = Config.fromJSON(new File("singleNodeConfig.json"));
RedissonClient client = Redisson.create(config);
```

Đây là mẫu config cho `singleNodeConfig.json`

```
{
    "singleServerConfig": {
        "idleConnectionTimeout": 10000,
        "pingTimeout": 1000,
        "connectTimeout": 10000,
        "timeout": 3000,
        "retryAttempts": 3,
        "retryInterval": 1500,
        "reconnectionTimeout": 3000,
        "failedAttempts": 3,
        "password": null,
        "subscriptionsPerConnection": 5,
        "clientName": null,
        "address": "redis://127.0.0.1:6379",
        "subscriptionConnectionMinimumIdleSize": 1,
        "subscriptionConnectionPoolSize": 50,
        "connectionMinimumIdleSize": 10,
        "connectionPoolSize": 64,
```

```
        "database": 0,
        "dnsMonitoring": false,
        "dnsMonitoringInterval": 5000
    },
    "threads": 0,
    "nettyThreads": 0,
    "codec": null,
    "useLinuxNativeEpoll": false
}
```

Đây là mẫu config cho `singleNodeConfig.yaml`

```
singleServerConfig:
    idleConnectionTimeout: 10000
    pingTimeout: 1000
    connectTimeout: 10000
    timeout: 3000
    retryAttempts: 3
    retryInterval: 1500
    reconnectionTimeout: 3000
    failedAttempts: 3
    password: null
    subscriptionsPerConnection: 5
    clientName: null
    address: "redis://127.0.0.1:6379"
    subscriptionConnectionMinimumIdleSize: 1
    subscriptionConnectionPoolSize: 50
    connectionMinimumIdleSize: 10
    connectionPoolSize: 64
    database: 0
    dnsMonitoring: false
    dnsMonitoringInterval: 5000
threads: 0
nettyThreads: 0
codec: !<org.redisson.codec.JsonJacksonCodec> {}
useLinuxNativeEpoll: false
```

Để lưu từ cấu hình Java sang file JSON hoặc YAML config, chúng ta thực hiện như sau:

```
Config config = new Config();
// ... we configure multiple settings here in Java
String jsonFormat = config.toJSON();
String yamlFormat = config.toYAML();
```

4. Redisson operation

Redisson hỗ trợ các interface để thực thi synchronous, asynchronous and reactive interfaces.

VD: Synchronous

```
RedissonClient client = Redisson.create();
RAtomicLong myLong = client.getAtomicLong('myLong');
```

VD: Asynchronous

```
RFuture<Boolean> isSet = myLong.compareAndSetAsync(6, 27);
isSet.handle((result, exception) -> {
    // handle the result or exception here.
});
```

VD: Reactive

```
RedissonReactiveClient client = Redisson.createReactive();
RAtomicLongReactive myLong = client.getAtomicLong("myLong");

Publisher<Boolean> isSetPublisher = myLong.compareAndSet(5, 28);
```

5. Objects

Redisson định nghĩa các Objec như sau:

- ObjectHolder
- BinaryStreamHolder
- GeospatialHolder
- BitSet
- AtomicLong
- AtomicDouble
- Topic
- BloomFilter
- HyperLogLog

5.1. ObjectHolder

ObjectHolder có thể lưu trữ tất cả các kiểu dữ liệu.

```
RBucket<Ledger> bucket = client.getBucket("ledger");
bucket.set(new Ledger());
Ledger ledger = bucket.get();
```

5.2. BinaryStreamHolder

5.3. GeospatialHolder

5.4. BitSet

5.5. AtomicLong

AtomicLong lưu trữ giá trị kiểu long

```
RAtomicLong atomicLong = client.getAtomicLong("myAtomicLong");
atomicLong.set(5);
atomicLong.incrementAndGet();
```

5.6. AtomicDouble

5.7. Topic

Topic để lưu trữ các bản tin trong cơ chế pub-sub của Redis.

```
RTopic<CustomMessage> subscribeTopic = client.getTopic("baeldung");
subscribeTopic.addListener(
  (channel, customMessage)
  -> future.complete(customMessage.getMessage()));
```

5.8. BloomFilter

5.9. HyperLogLog

6. Collections

Redisson định nghĩa và hỗ trợ các kiểu Collection như sau:

- Map
- Multimap
- Set
- SortedSet

- ScoredSortedSet
- LexSortedSet
- List
- Queue
- Deque
- BlockingQueue
- BoundedBlockingQueue
- BlockingDeque
- BlockingFairQueue
- DelayedQueue
- PriorityQueue
- PriorityDeque

6.1. Map

Redisson Map được implement từ *java.util.concurrent.ConcurrentMap* và *java.util.Map* interfaces.

Bao gồm: *RMap*, *RMapCache*, *RLocalCachedMap* và *RClusteredMap*

```
RMap<String, Ledger> map = client.getMap("ledger");
Ledger newLedger = map.put("123", new Ledger());
```

6.2. Multimap

6.3. Set

Redisson *Set* được implement từ *java.util.Set* interface.

Bao gồm: *RSet*, *RSetCache*, và *RClusteredSet*.

```
RSet<Ledger> ledgerSet = client.getSet("ledgerSet");
ledgerSet.add(new Ledger());
```

6.4. SortedSet

6.5. ScoredSortedSet

6.6. LexSortedSet

6.7. List

Redisson *Lists* được implement từ *java.util.List* interface.

Let's create a *List* with Redisson:

```
RList<Ledger> ledgerList = client.getList("ledgerList");
ledgerList.add(new Ledger());
```

6.8. Queue

6.9. Deque

6.10. BlockingQueue

6.11. BoundedBlockingQueue

6.12. BlockingDeque

6.13. BlockingFairQueue

6.14. DelayedQueue

6.15. PriorityQueue

6.16. PriorityDeque

7. Locks and Synchronizers

Redisson cho phép các thread thực hiện khóa (lock), đồng bộ (synchronization) qua applications/servers. Bao gồm:

- Lock
- FairLock
- MultiLock
- ReadWriteLock
- Semaphore
- PermitExpirableSemaphore
- CountDownLatch

7.1. Lock

Redisson's *Lock* được implement từ *java.util.concurrent.locks.Lock* interface.

```
RLock lock = client.getLock("lock");
lock.lock();
// perform some long operations...
lock.unlock();
```

7.2. FairLock

7.3. MultiLock

Redisson's *RedissonMultiLock* nhòm nhiều đối tượng kiểu *RLock* để trở thành 1 lock block:

```
RLock lock1 = clientInstance1.getLock("lock1");
RLock lock2 = clientInstance2.getLock("lock2");
RLock lock3 = clientInstance3.getLock("lock3");

RedissonMultiLock lock = new RedissonMultiLock(lock1, lock2, lock3);
lock.lock();
// perform long running operation...
lock.unlock();
```

7.4. ReadWriteLock

7.5. Semaphore

7.6. PermitExpirableSemaphore

7.7. CountDownLatch

8. Service

8.1. Remote Service

Cơ chế này hỗ trợ kiểu thực thi trên server-side từ remote method được gọi từ client-side:

Server-side register:

```
RRemoteService remoteService = client.getRemoteService();
LedgerServiceImpl ledgerServiceImpl = new LedgerServiceImpl();

remoteService.register(LedgerServiceInterface.class, ledgerServiceImpl);
```

Client-side call remote method:

```
RRemoteService remoteService = client.getRemoteService();
LedgerServiceInterface ledgerService
  = remoteService.get(LedgerServiceInterface.class);
```

```
List<String> entries = ledgerService.getEntries(10);
```

8.2. Live Object Service

9. Pipelining

Redisson hỗ trợ pipelining để thực thi một batch các lệnh:

```
RBatch batch = client.createBatch();
batch.getMap("ledgerMap").fastPutAsync("1", "2");
batch.getMap("ledgerMap").putAsync("2", "5");

List<?> result = batch.execute();
```

10. Scripting

Redisson hỗ trợ LUA scripting. Chúng ta có thể thực thi LUA scripts đến Redis:

```
client.getBucket("foo").set("bar");
String result = client.getScript().eval(Mode.READ_ONLY,
"return redis.call('get', 'foo')", RScript.ReturnType.VALUE);
```

11. Low-Level Client

Redisson hỗ trợ Low-level Client để thực thi các native Redis command:

```
RedisClient client = new RedisClient("localhost", 6379);
RedisConnection conn = client.connect();
conn.sync(StringCodec.INSTANCE, RedisCommands.SET, "test", 0);

conn.closeAsync();
client.shutdown();
```

12. Ví dụ mẫu

Xem thêm tại đây

Revision #2
Created 25 April 2020 06:38:46 by Laptrinh.vn
Updated 25 April 2020 07:26:08 by Laptrinh.vn