

Java data structures powered by **Redis**. Introduction to **Redisson**

Nikita Koksharov

Founder of  **redisson**

WHY **REDISSON**? WHY DO WE
NEED ANOTHER **REDIS** CLIENT?

COMMUNITY ORIENTED
OPEN SOURCE (APACHE LICENCE)

DISTRIBUTED COLLECTIONS

- ▶ Map *
- ▶ MultiMap *
- ▶ LocalCachedMap
- ▶ Set *
- ▶ SortedSet
- ▶ ScoredSortedSet
- ▶ LexSortedSet
- ▶ List
- ▶ Queue
- ▶ Deque
- ▶ BlockingQueue
- ▶ BlockingDeque
- ▶ BoundedBlockingQueue
- ▶ BlockingFairQueue
- ▶ DelayedQueue

* Supports **individual** element eviction

MAP

```
ConcurrentMap<Integer, MyObject> map = new ConcurrentHashMap<>();  
map.put(20, new MyObject("oldobj"));  
map.putIfAbsent(20, new MyObject("newobj"));  
map.containsKey(1);
```

REDISSON MAP

```
ConcurrentMap<Integer, MyObject> map = redisson.getMap("someMap");  
map.put(20, new MyObject("oldobj"));  
map.putIfAbsent(20, new MyObject("newobj"));  
map.containsKey(1);
```

REDISSON MAP EVICTION

```
RMapCache<Integer, String> map = redisson.getMapCache("someMap");  
map.put(20, "oldobj", 20, TimeUnit.MINUTES);  
map.containsKey(4);  
map.putIfAbsent(2, "oldobj", 5, TimeUnit.SECONDS);
```

REDISSON SET

```
Set<String> set = redisson.getSet("someSet");  
set.add("value");  
set.contains("value");  
set.remove("value");
```


REDISSON BLOCKINGQUEUE

```
BlockingQueue<MyObj> queue = redisson.getSet("someSet");  
set.add(new MyObj("value"));  
MyObj obj = queue.peek();  
MyObj obj = queue.poll(10, TimeUnit.MINUTES);
```

DISTRIBUTED LOCKS AND SYNCHRONIZERS

- ▶ Lock
- ▶ FairLock
- ▶ RedLock
- ▶ MultiLock
- ▶ ReadWriteLock
- ▶ Semaphore
- ▶ PermitExpirableSemaphore
- ▶ CountdownLatch
- ▶ Phaser (Planned)

REDISSON LOCK

```
RLock lock = redisson.getLock("lock");  
lock.lock();  
// or  
lock.lock(10, TimeUnit.MINUTES);  
//...  
lock.unlock();
```

DISTRIBUTED OBJECTS

- ▶ Bucket (Object Holder)
- ▶ BinaryStream (Input & Output Stream)
- ▶ Geo (Geospatial Object Holder)
- ▶ BitSet
- ▶ AtomicLong
- ▶ AtomicDouble
- ▶ Topic (Pub/Sub)
- ▶ BloomFilter
- ▶ HyperLogLog

REDISSON PUB/SUB

```
RTopic<SomeMessage> topic = redisson.getTopic("someTopic");
topic.addListener(new MessageListener<SomeMessage>() {
    @Override
    public void onMessage(String channel, SomeMessage message) {
        System.out.println(message);
    }
});
```

// in other thread or other JVM

```
RTopic<SomeMessage> topic = redisson.getTopic(" someTopic");
topic.publish(new SomeMessage("new message"));
```

INTERGRATION WITH FRAMEWORKS

- ▶ Spring Cache
- ▶ Hibernate Cache
- ▶ JCache API (JSR-107) implementation
- ▶ Tomcat Session Manager
- ▶ Spring Session

CONNECTION MODES

- ▶ Replicated nodes *
- ▶ Cluster nodes *
- ▶ Sentinel nodes
- ▶ Master with Slave nodes
- ▶ Single node

* Also supports **AWS ElastiCache** and **Azure Redis Cache**

DATA SERIALIZATION

- ▶ Jackson JSON
- ▶ Avro
- ▶ Smile
- ▶ CBOR
- ▶ MsgPack
- ▶ Snappy
- ▶ Kryo
- ▶ FST
- ▶ LZ4
- ▶ JDK Serialization

HOW TO START

```
// 1. Create config object
```

```
Config = new Config();
```

```
config.useClusterServers()
```

```
    .addNodeAddress("myserver.com:7000", "myserver.com:7001");
```

```
// 2. Create Redisson instance
```

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

```
// 3. Get object you need
```

```
Map<String, String> map = redisson.getMap("myMap");
```

ASYNCHRONOUS COMMAND EXECUTION

```
RMapAsync<Integer, String> map = redisson.getMap("someMap");  
Future<String> putIfFuture = map.putIfAbsentAsync(20, "object");  
Future<String> getFuture = map.getAsync(20);
```

```
getFuture.addListener(new FutureListener<Boolean>() {  
    @Override  
    public void operationComplete(Future<Boolean> future)  
        throws Exception {  
        //...  
    }  
});
```

REACTIVE COMMAND EXECUTION

```
RedissonReactive redisson = Redisson.createReactive(config);  
RMapReactive<Integer, String> map = redisson.getMap("someMap");  
Publisher<String> putRes = map.put(20, "object");  
Publisher<String> value = map.getAsync(20);
```


USED BY

- ▶ Electronic Arts
- ▶ Baidu
- ▶ Infor
- ▶ New Relic Synthetics
- ▶ Singtel
- ▶ Crimson Hexagon
- ▶ Brookhaven National Laboratory
- ▶ Netflix Dyno client
- ▶ 武林Q传
- ▶ Monits
- ▶ Ocous
- ▶ Invaluable
- ▶ Clover
- ▶ Apache Karaf Decanter
- ▶ Atmosphere Framework
- ▶ BrandsEye
- ▶ Datorama
- ▶ BrightCloud
- ▶ Azar
- ▶ Snapfish
- ▶ ...

THANK YOU!

<http://redisson.org>