Leap Ahead with Redis 6.2

Brian Sam-Bodden

- Developer Advocate at Redis
- @bsbodden on (Twitter|GitHub|GitLab|LinkedIn)
- Technologist
- Entrepreneur
- Author
- Teacher, and Student
- Aims to push the envelope of technology and fill the world with better, more passionate programmers.
- Brings vision and energy, and willingness to start at the beginning.

DaShaun Carter

- 319 days as Partner Solution Architect, Redis
- @dashaun on (Twitter|GitHub|GitLab|LinkedIn)
- A husband, father of four, volunteer and struggling athlete.
- Deliberately practicing to build better software, faster.
- Generalist.
- Computer Science. MBA.
- Continuously learning.
- Believes that every rep counts.
- Doesn't have anything figured out.
- Trying to get a little better every day.
- Understanding in order to be understood.

Redis

Redis

- The "Most Loved" database in StackOverflow's Developer Survey
- For the 5th year in a row
- Data stored in memory, not on disk
- < 1m latency
- 162 clients in 50 different languages

Redis OSS 6.2

The Community Edition

Redis OSS 6.2

The Community Edition

- 6.2 was driven by the community
- Loads of community contributed features
- 15 new commands
- 5 commands changed
- 8 commands deprecated

Spring Data

- Data comes in many shapes and flavors
- The Spring Data family of project gives you:
    - An Java/Spring idiomatic way to access that data
    - From low-level constructs...
    - To high-level OO abstractions... Reactive/Non-Reactive/Functional-ish
    - Object-Mapping and Data Repositories
- Learn more at https://spring.io/projects/spring-data

Spring Data Redis (SDR)

- Part of the Spring Data Family
- Provides easy configuration and access to Redis
- Low-level connectivity via Lettuce & Jedis libraries
- RedisTemplate(s): High-level abstractions for Redis Ops
- Implements Key-Value Mapping/Repositories
- Learn more at https://spring.io/projects/spring-data-redis

String : ValueOperations

SET -> set

In [9]:

```
redis-cli set leapaheadkey leapaheadvalue
```

```
OK
```

```java
@Component
public class Set implements Function<Map<String,String>, String> {
    private final StringRedisTemplate redisTemplate;

    public Set(StringRedisTemplate redisTemplate) {
        this.redisTemplate = redisTemplate;
    }

    @Override
    public String apply(Map<String,String> input) {
        redisTemplate
                .opsForValue()
                .set(input.get("key"), input.get("value"));
        return "OK";
    }
}
```

```
In [10]: curl -H "Content-Type: application/json" localhost:8080/set -d '{"key":"leapahead-string"
```

```
OK
```

String

GET -> get

In [11]:

```
redis-cli get leapaheadkey
```

```
"leapaheadvalue"
```

```java
@Component
class Get implements Function<String, String> {
    private final StringRedisTemplate redisTemplate;
    public ValueGet(StringRedisTemplate redisTemplate){
        this.redisTemplate = redisTemplate;
    }
    @Override
    public String apply(String input) {
        return redisTemplate.opsForValue().get(input);
    }
}
```

In [12]:

```
curl -H "Content-Type: text/plain" localhost:8080/get -d 'leapaheadkey'
```

```
leapaheadvalue
```

String : ValueOperations

GETSET -> getAndSet

In [13]:

```
redis-cli set leapaheadkey leapaheadvalue
redis-cli get leapaheadkey
redis-cli getset leapaheadkey updatedvalue
redis-cli getset leapaheadkey anotherupdatedvalue
redis-cli get leapaheadkey
```

```
OK
"leapaheadvalue"
"leapaheadvalue"
"updatedvalue"
"anotherupdatedvalue"
```

```java
@Component
public class GetAndSet implements Function<Map<String,String>, String> {
    private final StringRedisTemplate redisTemplate;

    public GetAndSet(StringRedisTemplate redisTemplate) {
        this.redisTemplate = redisTemplate;
    }

    @Override
    public String apply(Map<String,String> input) {
        return redisTemplate
                .opsForValue()
                .getAndSet(input.get("key"), input.get("value"));
    }
}
```

In [14]:

```
curl -H "Content-Type: application/json" localhost:8080/set -d '{"key":"leapahead-string"
echo ''
curl -H "Content-Type: application/json" localhost:8080/getAndSet -d '{"key":"leapahead-s
echo ''
curl -H "Content-Type: text/plain" localhost:8080/get -d 'leapahead-string'
```

```
OK
Spring Cloud Function
This is so cool!  Shout out to Mark Paluch!!!!
```

String : ValueOperations

GETEX -> getAndExpire

In [15]:

```
redis-cli set leapaheadkey 'This message should self destruct in 3 seconds'
redis-cli getex leapaheadkey ex 3
sleep 1
redis-cli ttl leapaheadkey
sleep 2
redis-cli get leapaheadkey
```

```
OK
"This message should self destruct in 3 seconds"
(integer) 2
(nil)
```

```java
@Component
public class GetEx implements Function<Map<String,String>, String> {
    private final StringRedisTemplate redisTemplate;

    public GetEx(StringRedisTemplate redisTemplate) {
        this.redisTemplate = redisTemplate;
    }

    @Override
    public String apply(Map<String,String> input) {
        Duration t = Duration.ofSeconds(Long.parseLong(input.get("value")));
        return redisTemplate
                .opsForValue()
                .getAndExpire(input.get("key"), t);
    }
}
```

In [16]:

```
curl -H "Content-Type: application/json" localhost:8080/set -d '{"key":"leapahead-string"
echo ''
curl -H "Content-Type: application/json" localhost:8080/getEx -d '{"key":"leapahead-strin
echo ''
echo '(Sleeping for 3 seconds)'
sleep 3
curl -H "Content-Type: text/plain" localhost:8080/get -d 'leapahead-string'
```

```
OK
Self-Destructing in 3 seconds
(Sleeping for 3 seconds)
{"timestamp":"2021-09-02T20:15:09.300+00:00","path":"/get","status":500,"e
rror":"Internal Server Error","requestId":"e36c6a60-1"}
```

String : ValueOperations

GETDEL -> getAndDelete

In [17]:

```
redis-cli set leapaheadkey 'How about a once per customer use-case?'
redis-cli getdel leapaheadkey
redis-cli get leapaheadkey
```

```
OK
"How about a once per customer use-case?"
(nil)
```

```java
@Component
class GetDel implements Function<String, String> {
    private final StringRedisTemplate redisTemplate;
    public GetDel(StringRedisTemplate redisTemplate){
        this.redisTemplate = redisTemplate;
    }
    @Override
    public String apply(String input) {
        return redisTemplate.opsForValue().getAndDelete(input);
    }
}
```

In [18]:

```
curl -H "Content-Type: application/json" localhost:8080/set -d '{"key":"leapahead-string"
echo ''
curl -H "Content-Type: application/json" localhost:8080/getDel -d 'leapahead-string'
echo ''
echo ''
curl -H "Content-Type: text/plain" localhost:8080/get -d 'leapahead-string'
```

```
OK
One-time-use-token AKA OTUT

{"timestamp":"2021-09-02T20:15:40.884+00:00","path":"/get","status":500,"e
rror":"Internal Server Error","requestId":"42d61014-1"}
```

String : ValueOperations

SET PXAT & EXAT

Add PXAT/EXAT arguments to SET command (#8327)

In [19]:
```
redis-cli set leapaheadkey 'This offer will not last' PXAT 1662163200000
echo '1662163200000 is September 3, 2022'
echo 'There are 31536000 seconds in a year'
redis-cli ttl leapaheadkey
echo ''
redis-cli set leapaheadkey2 'I will glady pay you later for a hamburger today' EXAT 16621
echo ''
redis-cli ttl leapaheadkey2
```

```
OK
1662163200000 is September 3, 2022
There are 31536000 seconds in a year
(integer) 31549407

OK

(integer) 31549407
```

```java
@Component
class SetPX implements Function<PX, String> {
    private final StringRedisTemplate redisTemplate;
    public SetPX(StringRedisTemplate redisTemplate){
        this.redisTemplate = redisTemplate;
    }
    @Override
    public String apply(PX input) {
        Objects.requireNonNull(redisTemplate.getConnectionFactory())
                .getConnection()
                .set(input.getKey().getBytes(StandardCharsets.UTF_8),
                        input.getValue().getBytes(StandardCharsets.UTF_8),
                        Expiration.unixTimestamp(Long.parseLong(input.getP()),
TimeUnit.MILLISECONDS),
                        RedisStringCommands.SetOption.UPSERT);
        return "OK";
    }
}
```

In [20]:

```
curl -H "Content-Type: application/json" localhost:8080/setPxAt -d '{"key":"leapahead-str
echo ''
curl -H "Content-Type: text/plain" localhost:8080/get -d 'leapahead-string'
sleep 2
echo ''
curl -H "Content-Type: text/plain" localhost:8080/get -d 'leapahead-string'
```

```
OK
Self-Destructing
{"timestamp":"2021-09-02T20:17:35.845+00:00","path":"/get","status":500,"e
rror":"Internal Server Error","requestId":"6435f396-1"}
```

```java
@Component
public class SetExAt implements Function<Map<String,String>, String> {
    private final RedisTemplate redisTemplate;

    public SetExAt(RedisTemplate redisTemplate) {
        this.redisTemplate = redisTemplate;
    }

    @Override
    public String apply(Map<String,String> input) {
        Objects.requireNonNull(redisTemplate.getConnectionFactory())
                .getConnection()
                .set(input.get("key").getBytes(StandardCharsets.UTF_8),
                        input.get("value").getBytes(StandardCharsets.UTF_8),
                        Expiration.unixTimestamp(Long.parseLong(input.get("e")),
TimeUnit.SECONDS),
                        RedisStringCommands.SetOption.UPSERT);
        return "OK";
    }
}
```

In [21]:

```
curl -H "Content-Type: application/json" localhost:8080/setExAt -d '{"key":"leapahead-str
echo ''
curl -H "Content-Type: text/plain" localhost:8080/get -d 'leapahead-string'
sleep 2
echo ''
curl -H "Content-Type: text/plain" localhost:8080/get -d 'leapahead-string'
```

```
OK
Self-Destructing
{"timestamp":"2021-09-02T20:18:03.126+00:00","path":"/get","status":500,"e
rror":"Internal Server Error","requestId":"ebf9ec44-1"}
```

Quiz Time

What is the maximum size of a Redis Key?

What is the maximum size of a Redis Key?

512MB

What is the maximum size of a Redis Value?

What is the maximum size of a Redis Value?

512MB

Hash

- Maps between string fields and string values
- A single hash can store over 4 billion field-value pairs
- Closely resembles Java Map

Hash : HashOperations

HSET / HMSET / HGETALL

In [22]:

```
redis-cli hmset leapahead:session status "So far, so good" track "Beginner, trending in t
redis-cli hset leapahead:session currentTime "$(date)"
redis-cli hgetall leapahead:session
```

```
OK
(integer) 1
 1) "status"
 2) "So far, so good"
 3) "track"
 4) "Beginner, trending in the right direction"
 5) "conf"
 6) "SpringOne 2021"
 7) "presentedBy"
 8) "@dashaun, @bsbodden"
 9) "currentTime"
10) "Thu Sep  2 15:19:51 CDT 2021"
```

```
@Component
public class Hset implements Function<Map<String,String>, String> {
        private final StringRedisTemplate redisTemplate;
        public Hset(StringRedisTemplate redisTemplate){
            this.redisTemplate = redisTemplate;
        }
        @Override
        public String apply(Map<String,String> input) {

redisTemplate.opsForHash().put(input.get("k"),input.get("f"),input.get("v"));
            return "OK";
        }
}
```

```
@Component
public class Hmset implements Function<Map<?,?>, String> {
        private final StringRedisTemplate redisTemplate;
        public Hmset(StringRedisTemplate redisTemplate){
            this.redisTemplate = redisTemplate;
        }
        @Override
        public String apply(Map<?,?> input) {
            redisTemplate.opsForHash().putAll((String)input.get("key")
                    ,(Map<?,?>)input.get("f"));
            return "OK";
        }
}
```

```java
@Component
public class HgetAll implements Function<String, Map<?,?>> {
        private final RedisTemplate redisTemplate;
        public HgetAll(RedisTemplate redisTemplate){
            this.redisTemplate = redisTemplate;
        }
        @Override
        public Map<?,?> apply(String input) {
            return redisTemplate.opsForHash().entries(input);
        }
}
```

In [23]:

```
curl -H "Content-Type: application/json" localhost:8080/hset -d '{"k":"SpringOne2021","f"
echo ''
curl -H "Content-Type: application/json" localhost:8080/hmset -d '{"key":"SpringOne2021",
echo ''
curl -H "Content-Type: text/plain" localhost:8080/hgetAll -d 'SpringOne2021'
echo ''
redis-cli hgetall SpringOne2021
```

```
OK
OK
{}
1) "start"
2) "dotspringdotio"
3) "Leap"
4) "Ahead"
5) "Redis"
6) "6.2"
```

Hash : HashOperations

HRANDFIELD

In [24]:

```
redis-cli hmset team:frontend 1 "Johnny" 2 "Pat" 3 "Nat" 4 "Whit" 5 "Sandy"
echo ''
redis-cli hrandfield team:frontend 4 WITHVALUES
```

```
OK

1) "5"
2) "Sandy"
3) "4"
4) "Whit"
5) "2"
6) "Pat"
7) "1"
8) "Johnny"
```

```java
@Component
public class HrandField implements Function<Map<String,String>, List<String>> {
        private final StringRedisTemplate redisTemplate;
        public HrandField(StringRedisTemplate redisTemplate){
            this.redisTemplate = redisTemplate;
        }
        @Override
        public List<String> apply(Map<String,String> input) {
            return
((StringRedisConnection)redisTemplate.getConnectionFactory().getConnection())
                    .hRandField(input.get("key"),
                            Long.parseLong(input.get("count")));
        }
}
```

```java
@Component
public class HrandFieldWithValues implements Function<Map<String,String>,
List<Map.Entry<String, String>>> {
        private final StringRedisConnection stringRedisConnection;
        public HrandFieldWithValues(StringRedisConnection stringRedisConnection){
            this.stringRedisConnection = stringRedisConnection;
        }
        @Override
        public List<Map.Entry<String, String>> apply(Map<String,String> input) {
            return stringRedisConnection.hRandFieldWithValues(input.get("key"),
                                Long.parseLong(input.get("count")));
        }
}
```

In [25]:

```
curl -H "Content-Type: application/json" localhost:8080/hmset -d '{"key":"Leaping","f":{"
echo ''
curl -H "Content-Type: application/json" localhost:8080/hrandField -d '{"key":"Leaping","
echo ''
curl -H "Content-Type: application/json" localhost:8080/hrandField -d '{"key":"Leaping","
echo ''
curl -H "Content-Type: application/json" localhost:8080/hrandFieldWithValues -d '{"key":"
```

```
OK
["RedisBloom","Azure"]
["RedisBloom","Harvester","RedisBloom","OpenShift","OpenShift"]
[{"Harvester":"Suse"},{"OpenShift":"Redis Enterprise Operator"}]
```

Cheat Code

Cheat Code

Don’t try to remember the Redis commands.

Cheat Code

Don’t try to remember the Redis commands.

Remember Spring Data Redis!

List

- An ordered sequence of strings
- Comparable to a Java ArrayList
- Multi-purpose:
    - Stacks
    - Queues
- A single List can hold over 4 billion entries.

Adding to a List

- You can **add** things by:
  - **Pushing** in: `LPUSH` / `LPUSHX` , `RPUSH` / `RPUSHX`
  - **Inserting** before/after: `LINSERT`
  - **Setting** the value at an **index**: `LSET`

Remove from a List

- You can **remove things** things by:
    - **Popping** off: `LPOP` / `RPOP` / `BLPOP` / `BRPOP`
    - **By value** `LREM`
    - **By index** range: `LTRIM`

Accessing Elements

- **By index**: `LINDEX`
- **By range**: `LRANGE`

Between Lists

- **Last** from one list, **to first** in another: `RPOPLPUSH` / `BRPOPLPUSH`
- **Pop** and then **Push**: `LMOVE` / `BLMOVE`

In [26]:

```
redis-cli DEL funny_words
redis-cli RPUSH funny_words "Shenanigans" "Bamboozle" "Bodacious"
echo ''
redis-cli LRANGE funny_words 0 -1
```

```
(integer) 0
(integer) 3

1) "Shenanigans"
2) "Bamboozle"
3) "Bodacious"
```

In [27]:

```
redis-cli LRANGE funny_words 0 -1
echo ''
redis-cli LPUSH funny_words "Bumfuzzle"
echo ''
redis-cli LRANGE funny_words 0 -1
```

```
1) "Shenanigans"
2) "Bamboozle"
3) "Bodacious"

(integer) 4

1) "Bumfuzzle"
2) "Shenanigans"
3) "Bamboozle"
4) "Bodacious"
```

In [28]:

```
redis-cli LRANGE funny_words 1 3
echo ''
redis-cli LINSERT funny_words BEFORE "Bamboozle" "Brouhaha"
echo ''
redis-cli LSET funny_words -2 "Flibbertigibbet"
echo ''
redis-cli LRANGE funny_words 0 -1
```

```
1) "Shenanigans"
2) "Bamboozle"
3) "Bodacious"

(integer) 5

OK

1) "Bumfuzzle"
2) "Shenanigans"
3) "Brouhaha"
4) "Flibbertigibbet"
5) "Bodacious"
```

In [29]:

```
redis-cli LRANGE funny_words 0 -1
echo ''
redis-cli LPOP funny_words
echo ''
redis-cli LRANGE funny_words 0 -1
```

```
1) "Bumfuzzle"
2) "Shenanigans"
3) "Brouhaha"
4) "Flibbertigibbet"
5) "Bodacious"

"Bumfuzzle"

1) "Shenanigans"
2) "Brouhaha"
3) "Flibbertigibbet"
4) "Bodacious"
```

In Spring Data Redis

```java
public class ListExample {

  @Autowired
  private StringRedisTemplate redisTemplate;

  @Resource(name="stringRedisTemplate")
  private ListOperations<String, String> listOps;

  public void playWithLists() {
    //...
  }
}
```

In Spring Data Redis

```java
public void playWithLists() {
  listOps.rightPushAll("funny_words", "Shenanigans", "Bamboozle", "Bodacious");
  List<String> range = listOps.range("funny_words", 0, -1);
  System.out.println(range.toArray());

  listOps.leftPush("funny_words", "Bumfuzzle");
  range = listOps.range("funny_words", 1, 3);

  listOps.leftPush("funny_words", "Bamboozle", "Brouhaha");
  // ...

  listOps.set("funny_words", -2, "Flibbertigibbet");
  // ...
  System.out.println(listOps.size("funny_words"));
}
```

What’s new with Lists in 6.2?

- Add `LMOVE` and `BLMOVE` commands that pop and push arbitrarily (#6929)
- Add the `COUNT` argument to `LPOP` and `RPOP` (#8179)

`LMOVE` on the CLI

- Right-most from `list_one` to the left of `list_two`
- Left-most from `list_one` to the left of `list_two`

In [30]:
```
redis-cli DEL list_one
redis-cli DEL list_two
redis-cli RPUSH list_one "one" "two" "three"
echo ''
redis-cli LMOVE list_one list_two RIGHT LEFT
echo ''
redis-cli LMOVE list_one list_two LEFT RIGHT
echo ''
redis-cli LRANGE list_one 0 -1
echo ''
redis-cli LRANGE list_two 0 -1
```

```
(integer) 0
(integer) 0
(integer) 3

"three"

"one"

1) "two"

1) "three"
2) "one"
```

`LMOVE` on SDR as a JUnit Test

```
@Test
void testLMOVE() {
  listOps.rightPushAll("list_one", "one", "two", "three");
  listOps.move("list_one", RIGHT, "list_two", LEFT);
  listOps.move("list_one", LEFT, "list_two", RIGHT);

  List<String> listOne = listOps.range("list_one", 0, -1);
  List<String> listTwo = listOps.range("list_two", 0, -1);

  assertTrue(listOne.containsAll(List.of("two")));
  assertTrue(listTwo.containsAll(List.of("three", "one")));
}
```

`COUNT` on `LPOP`/`RPOP`

Pop "n" things from the left or the right

In [31]:

```
redis-cli RPUSH mylist "one"
redis-cli RPUSH mylist "two"
redis-cli RPUSH mylist "three"
redis-cli LPOP mylist 2
redis-cli LRANGE mylist 0 -1
```

```
(integer) 1
(integer) 2
(integer) 3
1) "one"
2) "two"
1) "three"
```

`COUNT` on `LPOP` / `RPOP` on SDR as a JUnit Test

```java
@Test
void testLPOP() {
  listOps.rightPush("mylist", "one");
  listOps.rightPush("mylist", "two");
  listOps.rightPush("mylist", "three");
  listOps.leftPop("mylist", 2);
  List<String> myList = listOps.range("mylist", 0, -1);
  assertTrue(myList.containsAll(List.of("three")));
}
```

Set

- Collections of unique, unsorted string elements.
- Set Operations (Union/Intersection/Subtraction)
- Most operations in constant time (`O(n)`)

Set Use Cases

- Unique item management (tags/folksonomies)
- Tracking IPs, content filtering
- As a support data structure to manage membership
    - SDR maintains Primary Keys for mapped classes in a Redis Set

Working with Sets

- Add/Remove: `SADD` / `SPOP` / `SREM`
- Access/Retrieve: `SMEMBERS` / `SRANDMEMBERS` / `SSCAN`
- Set Info: `SCARD` / `SISMEMBER` / `SMISMEMBER`
- Set Ops: `SDIFF*`/ `SINTER*` / `SUNION*` / `SMOVE`

In [32]:

```
redis-cli DEL colors
redis-cli SADD colors "red" "yellow" "green" "fushia"
redis-cli SADD colors "yellow"
redis-cli SISMEMBER colors "green"
redis-cli SISMEMBER colors "magenta"
redis-cli SREM colors "green"
redis-cli SREM colors "green"
redis-cli SMEMBERS colors
```

```
(integer) 0
(integer) 4
(integer) 0
(integer) 1
(integer) 0
(integer) 1
(integer) 0
1) "yellow"
2) "fushia"
3) "red"
```

Sets in Spring Data Redis

```
@Test
void testSimpleExample() {
  setOps.add("colors", "red", "yellow", "green", "fushia");
  setOps.add("colors", "yellow");
  Set<String> members = setOps.members("colors");
  assertTrue(members.containsAll(List.of("red", "yellow", "green", "fushia")));
  assertTrue(setOps.isMember("colors", "green"));
  assertFalse(setOps.isMember("colors", "magenta"));
  assertEquals(1, setOps.remove("colors", "green"));
  members = setOps.members("colors");
  assertTrue(members.containsAll(List.of("red", "yellow", "fushia")));
}
```

What’s new with Sets in 6.2?

- Add `SMISMEMBER` command that checks multiple members (#7615)

In [33]:

```
redis-cli DEL colors
redis-cli SADD colors "red" "yellow" "green" "fushia"
redis-cli SMISMEMBER colors "red" "black" "green"
```

```
(integer) 1
(integer) 4
1) (integer) 1
2) (integer) 0
3) (integer) 1
```

`SMISMEMBER` on SDR as a JUnit Test

```java
@Test
void testSMISMEMBER() {
  setOps.add("colors", "red", "yellow", "green", "fushia");
  Map<Object, Boolean> memberCheck = setOps.isMember("colors", "red", "black",
"green");
  assertTrue(memberCheck.get("red"));
  assertFalse(memberCheck.get("black"));
  assertTrue(memberCheck.get("green"));
}
```

In [34]:

```
redis-cli flushdb
redis-cli SADD colors "red" "yellow" "green" "fushia"
redis-cli SMISMEMBER colors "red" "black" "green"
```

```
OK
(integer) 4
1) (integer) 1
2) (integer) 0
3) (integer) 1
```

Sorted Set

- A weighted Sets: A mix between a `Set` and a `Hash`
- Elements
    - are tuples with a **value** and a **score**
    - are always taken sorted by their score
    - can be retrieved in ranges

Sorted Set Use Cases

- Priority queues
- Low-latency leaderboards
- Secondary indexing in general

In [35]:
```
redis-cli ZADD game1 100 "Frank" 740 "Jennifer" 200 "Pieter" 512 "Dave" 690 "Ana"
redis-cli ZADD game2 212 "Joe" 230 "Jennifer" 450 "Mary" 730 "Tom" 512 "Dave" 200 "Frank"
echo ''
redis-cli ZRANGE game2 0 -1 WITHSCORES
```

```
(integer) 5
(integer) 6

 1) "Frank"
 2) "200"
 3) "Joe"
 4) "212"
 5) "Jennifer"
 6) "230"
 7) "Mary"
 8) "450"
 9) "Dave"
10) "512"
11) "Tom"
12) "730"
```

In [36]:
```
redis-cli ZINTER 2 game1 game2 WITHSCORES
echo ''
redis-cli ZINTER 2 game1 game2 WITHSCORES AGGREGATE max
echo ''
redis-cli ZDIFF 2 game1 game2 WITHSCORES
```

```
1) "Frank"
2) "300"
3) "Jennifer"
4) "970"
5) "Dave"
6) "1024"

1) "Frank"
2) "200"
3) "Dave"
4) "512"
5) "Jennifer"
6) "740"

1) "Pieter"
2) "200"
3) "Ana"
4) "690"
```

`ZADD` in SDR

```java
Set<TypedTuple<String>> game1 = Set.of( //
    TypedTuple.of("Frank", 100.0), TypedTuple.of("Jennifer", 740.0),
    TypedTuple.of("Pieter", 200.0), TypedTuple.of("Dave", 512.0),
    TypedTuple.of("Ana", 690.0));

Set<TypedTuple<String>> game2 = Set.of( //
    TypedTuple.of("Joe", 212.0), TypedTuple.of("Jennifer", 230.0),
    TypedTuple.of("Mary", 450.0), TypedTuple.of("Tom", 730.0),
    TypedTuple.of("Dave", 512.0), TypedTuple.of("Frank", 200.0));

zSetOps.add("game1", game1);
zSetOps.add("game2", game2);
```

`ZRANGE` in SDR

```java
Set<String> game1Players = zSetOps.range("game1", 0, -1);
assertArrayEquals(new String[] { "Frank", "Pieter", "Dave", "Ana", "Jennifer"},
game1Players.toArray());

Set<TypedTuple<String>> game2PlayersWithScores = zSetOps.rangeWithScores("game2",
0, -1);
TypedTuple<String> frankInGame2 = game2PlayersWithScores.iterator().next();
assertEquals("Frank", frankInGame2.getValue());
assertEquals(200.0, frankInGame2.getScore());
```

ZINTER in SDR

```
Set<TypedTuple<String>> inBothGames = zSetOps.intersectWithScores("game1",
"game2");
TypedTuple<String> frankInBothGamesTotal = inBothGames.iterator().next();
assertEquals("Frank", frankInBothGamesTotal.getValue());
assertEquals(300.0, frankInBothGamesTotal.getScore());

Set<TypedTuple<String>> inBothGamesWithMax = zSetOps.intersectWithScores("game1",
Set.of("game2"), Aggregate.MAX);
TypedTuple<String> frankInBothGamesMax = inBothGamesWithMax.iterator().next();
assertEquals("Frank", frankInBothGamesMax.getValue());
assertEquals(200.0, frankInBothGamesMax.getScore());
```

`ZDIFF` in SDR

```
Set<TypedTuple<String>> onlyInGame1 = zSetOps.differenceWithScores("game1",
"game2");
List<String> players = onlyInGame1.stream().map(t ->
t.getValue()).collect(Collectors.toList());
assertTrue(players.containsAll(Set.of("Pieter", "Ana")));
```

Sorted Set

- Add ZMSCORE command that returns an array of scores (#7593)
- Add ZDIFF and ZDIFFSTORE commands (#7961)
- Add ZINTER and ZUNION commands (#7794)
- Add ZRANDMEMBER command (#8297)
- Add the REV, BYLEX and BYSCORE arguments to ZRANGE, and the ZRANGESTORE command (#7844)

In [37]:

```
echo 'ZMSCORE w/ an array of scores'
echo ''
redis-cli ZADD myzset 1 "one"
redis-cli ZADD myzset 2 "two"
redis-cli ZMSCORE myzset "one" "two" "nofield"
```

```
ZMSCORE w/ an array of scores

(integer) 1
(integer) 1
1) "1"
2) "2"
3) (nil)
```

ZMSCORE w/ an array of scores

```
@Test
void testZMSCORE() {
  zSetOps.add("myzset", "one", 1);
  zSetOps.add("myzset", "two", 2);
  List<Double> scores = zSetOps.score("myzset", "one", "two", "nofield");

  assertArrayEquals(new Double[] { 1.0, 2.0, null }, scores.toArray());
}
```

```
In [38]:
echo 'ZDIFF Commands'
echo ''
redis-cli ZADD zset1 1 "one"
redis-cli ZADD zset1 2 "two"
redis-cli ZADD zset1 3 "three"
redis-cli ZADD zset2 1 "one"
redis-cli ZADD zset2 2 "two"
redis-cli ZDIFF 2 zset1 zset2
echo ''
redis-cli ZDIFF 2 zset1 zset2 WITHSCORES
```

```
ZDIFF Commands

(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 1
1) "three"

1) "three"
2) "3"
```

ZDIFF commands

```
@Test
void testZDIFF() {
  zSetOps.add("zset1", "one", 1);
  zSetOps.add("zset1", "two", 2);
  zSetOps.add("zset1", "three", 3);
  zSetOps.add("zset2", "one", 1);
  zSetOps.add("zset2", "two", 2);

  Set<String> diffs = zSetOps.difference("zset1", "zset2");
  assertArrayEquals(new String[] { "three" }, diffs.toArray());

  Set<TypedTuple<String>> diffsWScores = zSetOps.differenceWithScores("zset1",
"zset2");
  assertEquals(1, diffsWScores.size());
  TypedTuple<String> dtt = diffsWScores.iterator().next();
  assertEquals("three", dtt.getValue());
  assertEquals(3.0, dtt.getScore());
}
```

In [39]:

```
echo 'ZDIFFSTORE commands'
echo ''
redis-cli ZADD zset1 1 "one"
redis-cli ZADD zset1 2 "two"
redis-cli ZADD zset1 3 "three"
redis-cli ZADD zset2 1 "one"
redis-cli ZADD zset2 2 "two"
echo ''
redis-cli ZDIFFSTORE out 2 zset1 zset2
echo ''
redis-cli ZRANGE out 0 -1 WITHSCORES
```

```
ZDIFFSTORE commands

(integer) 0
(integer) 0
(integer) 0
(integer) 0
(integer) 0

(integer) 1

1) "three"
2) "3"
```

ZDIFFSTORE commands

```
@Test
void testZDIFFSTORE() {
  zSetOps.add("zset1", "one", 1);
  zSetOps.add("zset1", "two", 2);
  zSetOps.add("zset1", "three", 3);
  zSetOps.add("zset2", "one", 1);
  zSetOps.add("zset2", "two", 2);

  zSetOps.differenceAndStore("zset1", List.of("zset2"), "out");
  Set<TypedTuple<String>> withScores = zSetOps.rangeWithScores("out", 0, -1);
  assertEquals(1, withScores.size());
  TypedTuple<String> dtt = withScores.iterator().next();
  assertEquals("three", dtt.getValue());
  assertEquals(3.0, dtt.getScore());
}
```

Geo

- Sorted Set
- Latitude and longitude encoded into the score of the sorted set using the geohash algorithm
- "lat long" isn't the case here, it's "long lat"

In [40]:

```
redis-cli GEOADD running-poi -94.188224 39.013319 "Football"
redis-cli GEOADD running-poi -94.194606 39.018836 "Track"
redis-cli GEOADD running-poi -94.207570 39.019566 "Trail"
redis-cli GEOADD running-poi -94.238336 39.029377 "Lake"
redis-cli GEOADD running-poi -94.231930 39.066619 "Park"
redis-cli GEOADD running-poi -94.213729 39.038229 "Basketball"
redis-cli GEODIST running-poi "Park" "Basketball"
redis-cli DEL running-poi
```

```
(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 1
(integer) 1
"3527.3727"
(integer) 1
```

```
@Component
public class GeoAdd implements Function<Map<String,String>, Long> {
        private final StringRedisTemplate redisTemplate;
        public GeoAdd(StringRedisTemplate redisTemplate){
            this.redisTemplate = redisTemplate;
        }
        @Override
        public Long apply(Map<String,String> input) {
            Point p = new Point(Double.parseDouble(input.get("long")),
                    Double.parseDouble(input.get("lat")));
            return
redisTemplate.opsForGeo().add(input.get("k"),p,input.get("m"));
        }
}
```

In [41]:

```
curl -H "Content-Type: application/json" localhost:8080/geoAdd -d '{"k":"running-poi","la
echo ''
curl -H "Content-Type: application/json" localhost:8080/geoAdd -d '{"k":"running-poi","la
echo ''
```

```
1
1
```

In [42]:

```
curl -H "Content-Type: application/json" localhost:8080/geoSearchFromMemberByBoxKm -d '{"
echo ''
redis-cli ZRANGE inTheBox 0 -1 WITHSCORES
```

```
2
1) "Trail"
2) "1405606775774309"
3) "Basketball"
4) "1405609652767867"
```

WERE YOU PAYING ATTENTION?

WERE YOU PAYING ATTENTION?

I TOTALLY CHEATED

WERE YOU PAYING ATTENTION?

I TOTALLY CHEATED

I USED ZRANGE TO GET THE MEMBERS OF THE GEO

WERE YOU PAYING ATTENTION?

I TOTALLY CHEATED

I USED ZRANGE TO GET THE MEMBERS OF THE GEO

```
In [43]: redis-cli ZRANGE inTheBox 0 -1 WITHSCORES
```

```
1) "Trail"
2) "1405606775774309"
3) "Basketball"
4) "1405609652767867"
```

STREAMS

- Before Redis 6.2, streams could only be trimmed to an exact or approximate number of entries.
- This is a little at odds with the way that we do steam processing
- Each entry in a stream must have a unique ID greater than any previously seen in the stream.
- Redis by default uses millisecond timestamps for this.
- We now allow you to trim based on ID!

Brian!

What we’re up to at Redis

- Extending/complementing Spring Data Redis with:
    - Access to module commands via Spring's Templates
    - Multi-model Object-Mapping support
    - JSON object-mapping + RediSearch integration
    - Graph object-mapping
    - RediSearch integration for existing Redis Hash mapped entities

What we’re up to at Redis

- Plus...
    - Encapsulated "Use Cases" that can be applies declaratively
    - Graph object-mapping
    - RediSearch integration for existing Redis Hash mapped entities
    - Encapsulated "Use Cases" that can be applies declaratively

Redis Modules Templates

- Follow's Spring Data Redis "`opsForXXX()`" pattern
- Provide's a Spring Native way to interact at the command-level with:
- RedisJSON, RedisGraph, RediSearch, RedisAI, RedisBloom, and RedisTimeSeries

Redis Modules Templates

Inject a `RedisModulesOperations` bean:

```java
@Autowired
RedisModulesOperations<String, String> modulesOperations;
```

Retrieve a module template to use its commands:

```java
GraphOperations<String> graph = modulesOperations.opsForGraph();
// Create both source and destination nodes
graph("social", "CREATE (:person{name:'roi',age:32})");
graph("social", "CREATE (:person{name:'amit',age:30})");
```

RedisJSON Document-Object Mapping

Annotate a Java class with `@Document` annotation:

```java
@Document("company")
public class Company {
  @Id
  private String id;
  @NonNull
  private String name;
  private boolean publiclyListed;
}
```

RedisJSON Document-Object Mapping

Repository support via `RedisDocumentRepository` interface:

```
interface CompanyRepository extends RedisDocumentRepository<Company, String> {}
```

RedisJSON Document-Object Mapping

In action:

```
@Autowired CompanyRepository repository;

Company redislabs = repository.save(Company.of("RedisLabs"));
Company microsoft = repository.save(Company.of("Microsoft"));

System.out.println(">>>> Items in repo: " + repository.count());

Optional<Company> maybeRedisLabs = repository.findById(redislabs.getId());
Optional<Company> maybeMicrosoft = repository.findById(microsoft.getId());

System.out.println(">>>> Company: " + maybeRedisLabs.get());
```

RedisJSON/RediSearch Integration

- `@XXXIndexed` Annotations for RediSearch index creation and maintenance

```
@Document("my-doc")
public class MyDoc {
  @Id
  private String id;
  @NonNull
  @TextIndexed(alias = "title")
  private String title;
  @TagIndexed(alias = "tag")
  private Set<String> tag = new HashSet<String>();
}
```

Search anything...

- `@Query` and `@Aggregation` for powerful native RediSearch Queries and Aggregations:

```java
public interface MyDocRepository
  extends RedisDocumentRepository<MyDoc, String>, MyDocQueries {

  @Query(returnFields = {"$.tag[0]", "AS", "first_tag"})
  SearchResult getFirstTag();

  @Query("@title:$title @tag:{$tags}")
  Iterable<MyDoc> findByTitleAndTags(@Param("title") String title, @Param("tags")
Set<String> tags);

  @Aggregation(load = {"$.tag[1]", "AS", "tag2"})
  AggregationResult getSecondTagWithAggregation();
}
```

Looking Forward

- Today: 2.6 Milestone 2
- November 2021: Spring Data Redis 2.6 GA

Looking Forward

- Today: 2.6 Milestone 2
- November 2021: Spring Data Redis 2.6 GA

- One of the best part of this community is contributing
- Take the milestone for a spin
- Let us know how it works for you

THANKS

- @bsbodden
- @dashaun

https://github.com/LeapAheadWithRedis6-2