Урок 3. Введение Redis

docker run --name skillbox-redis -d redis:6.2.5 – скачивание и установка контейнера

docker exec -it skillbox-redis redis-cli – соединяемся с контейнером

ping – команда наша

PONG – ответ сервера

echo hello – передача сообщение серверу

"hello" – подтверждение получения сообщения сервером

set hello world – запишем на сервер сообщение

OK – подтверждение от сервера о записи сообщения

get hello – получение от сервера значения по ключу

"world" – ответ от сервера

set user:10:name Egor – запишем на сервер значени

OK – ответ от сервера

get user:10:name – получение от сервера значения по ключу

"Egor" – ответ от сервера

exists user:10 – проверка существования ключа

(integer) 0 – ответ от сервера, что ключ не существует

exists user:10 get user:10:name – проверка существования ключа

(integer) 1 – ответ от сервера, что ключ существует

exists user:10:name – проверка существования ключа

(integer) 1 – ответ от сервера, что ключ существует

del hello – удаление значения по ключу

(integer) 1 – ответ от сервера, что ключ удален

exists hello – проверка, что ключ действительно удален

(integer) 0 – ответ от сервера, что ключ действительно был удален

set hello world ex 60 – создание ключа с временем жизни 60 секунд

OK – ответ от сервера, что ключ создан

ttl hello – запрос от сервера сколько времени жизни значения

(integer) 55 – ответ от сервера

persist hello – остановка времени жизни. Ключ не удалится

(integer) 0 – ответ от сервера, что время остановилось

append hello "!!!" – дописываем значение по ключу

(integer) 8 – ответ от сервера (длинна значения)

set visitor:index 0 – создание счетчика

OK – подтверждение от сервера, что счетчик создан

incr visitor:index – увеличение счетчика

(integer) 1 – ответ от сервера значение индекса

decr visitor:index – уменьшение значение счетчика

(integer) 3 – ответ от сервера значение индекса

get visitor:index – запрос значения счетчика от сервера

"3" – ответ от значение счетчика

rename visitor:index visitor:main – переименование счетчика

OK – ответ от сервера

hset user:10 name Egor – записываем значение в хэшмеп

(integer) 1 – ответ от сервера

hget user:10 name – получение значение по ключу

"Egor"

hgetall user:10 – получить все значения по ключу

1) "name"

2) "Egor"

3) "occupation"

4) "programmer"

hincrby user:10 karma 10 – увеличение значения

(integer) 20

hgetall user:10

1) "name"

2) "Egor"

3) "occupation"

4) "programmer"

5) "karma"

6) "20"

sadd users [one@exemple.com](mailto:one@exemple.com) – добавим список электронных почт

(integer) 1

sadd users [one@exemple.com](mailto:one@exemple.com) – если попытаться добавить такую же почту, сервер ответит, что не добавил.

(integer) 0 – почта не добавлена

smembers users – просмотреть список добавленных ранее почт

1) "one@exemple.com"

2) [two@exemple.com](mailto:two@exemple.com)

sdiff users ban – покажет разницу в хранящихся данных в списках user и ban. Т.е. показывает оставшиеся после вычитания в user

1. "two@exemple.com"

sdiff ban users – покажет оставшиеся в ban

1. [three@exemple.com](mailto:three@exemple.com)

sunion ban users – покажет список уникальных объектов в объединенном множестве

1) "one@exemple.com"

2) "three@exemple.com"

3) [two@exemple.com](mailto:two@exemple.com)

spop ban – убрали из списка ban почту

[three@exemple.com](mailto:three@exemple.com)

Создаем отсортированный список

zadd events 20 event2

(integer) 1

127.0.0.1:6379> zadd events 5 event1

(integer) 1

127.0.0.1:6379> zadd events 10 event3

(integer) 1

zrange events 0 1 – выбираем из списка значения 0 и 1

1) "event1"

2) "event3"

zpopmin events – получения значение с минимальным индексом

1) "event1"

2) "5"

zpopmax events – получения значение с максимальным индексом

1) "event2"

2) "20"

Добавим в список значения слева

lpush user:10:scills redis

(integer) 1

127.0.0.1:6379> lpush user:10:scills mongodb

(integer) 2

lrange user:10:scills 0 5 – выведем добавленные значения

1) "mongodb"

2) "redis"

rpush user:10:scills mysql – добавляем элемент справа

(integer) 3

lrange user:10:scills 0 5 – выводим значения

1) "mongodb"

2) "redis"

3) "mysql"

lpop user:10:scills – вывод элемента слева (с удалением)

"mongodb"

rpop user:10:scills 2 – вывод 2 элементов справа (с удалением)

1) "mysql"

2) "redis"

llen user:10:scills – проверяем оставшиеся элементы в списке

(integer) 0

subscribe chat1 – подписались на канал chat1

Reading messages... (press Ctrl-C to quit)

1) "subscribe"

2) "chat1"

3) (integer) 1

psubscribe chat\* - подписываемся на все каналы chat

Reading messages... (press Ctrl-C to quit)

1) "psubscribe"

2) "chat\*"

3) (integer) 1

pubsub channels – узнать на какие каналы подписаны пользователи

1. "chat1"

pubsub numsub chat1 – узнать сколько подписчиков на канал chat1. Сервер ответил, что подписчик 1

1) "chat1"

2) (integer) 1

eval "redis.call('set', 'user:'..KEYS[1]..':name', ARGV[1])" 1 10 Ivan – вызываем хранимую процедуру и передаем ей значения

get user:10:name – запрашиваем у сервера значение, которое передали через хранимую процедуру

"Ivan"

script load "redis.call('set', 'user:'..KEYS[1]..':name', ARGV[1])" – сохраняем в скрипте вызов хранимой процедуры

"b37ab27103ebfb69de589d9f4e17d0721735173c" – сервер возвращает хэш, с помощью которого мы можем вызывать процедуру

evalsha b37ab27103ebfb69de589d9f4e17d0721735173c 2 101 102 Oleg Igor – вызываем процедуру и передаем ей значения

(nil)

get user:101:name – запрашиваем у сервера переданные значения

"Oleg"

script load "for i=1,#KEYS do redis.call('set', 'user:'..KEYS[i]..':name', ARGV[i]) end" – в скрипт записываем процедуру для сохранения сразу нескольких значений

"76c6474040cfb55b545569718f418350cec3cd26"

Передаем сразу несколько значений и потом их запрашиваем

evalsha 76c6474040cfb55b545569718f418350cec3cd26 2 101 102 Oleg Igor

(nil)

127.0.0.1:6379> get user:101:name

"Oleg"

127.0.0.1:6379> get user:102:name

"Igor"

multi – для начала транзакции

OK

127.0.0.1:6379(TX)>

set hello world – передаем значение

QUEUED – сервер отвечает, что значение поставлено в очередь на запись

127.0.0.1:6379(TX)> exec – комитим переданное значение (подтверждаем)

1) OK – сервер отвечает, что значение записано.

get hello – запрашиваем с сервера записанное значение

"world"

127.0.0.1:6379> multi – для начала транзакции

OK

127.0.0.1:6379(TX)> set hello world – передаем значение

QUEUED

127.0.0.1:6379(TX)> set hello world2 – передаем значение

QUEUED

127.0.0.1:6379(TX)> discard – команда для отмены транзакции без сохранения значений

OK