Redis – Wprowadzenie

Mirosław Boruta

Silesian Ruby User Group

19 lutego 2011

Redis jest. . .

RE mote DI ctionary S ervice

Redis jest. . .

Redis jest bazą przechowującą pary klucz-watrość, przy czym na wartości można dodatkowo wykonywać operacje.

Mocne strony Redis-a

Mocne strony Redis-a

- Wydajność

Mocne strony Redis-a

- Wydajność
- Atomiczność podstawowych operacji

Mocne strony Redis-a

- Wydajność
- Atomiczność podstawowych operacji
- Wbudowane typy danych

Słabe strony Redis-a

Słabe strony Redis-a

- Trwałość danych

Słabe strony Redis-a

- Trwałość danych
- RAM

Słabe strony Redis-a

- Trwałość danych
- RAM
- Optymalizacja pod względem szybkości kosztem zajmowanej przestrzeni pamięci

Klucze

Klucze są ciągami znaków:

Klucze

Klucze są ciągami znaków:

```
redis> SET foo bar
OK
```

Klucze

Klucze są ciągami znaków:

```
redis> SET foo bar
OK
redis> GET foo
"bar"
```

Klucze

Klucze są ciągami znaków:

```
redis> SET foo bar
OK
redis> GET foo
"bar"
redis> GET baz
(nil)
```

Klucze

Klucze mogą zawierać nie tylko litery:

Klucze

Klucze mogą zawierać nie tylko litery:

```
redis> SET user:1:name Bob
OK
```

Klucze

Klucze mogą zawierać nie tylko litery:

```
redis> SET user:1:name Bob
OK
redis> SET user:1:e-mail bob@ex
OK
```

Klucze

Pobieranie listy kluczy na podstawie wzorca:

Klucze

Pobieranie listy kluczy na podstawie wzorca:

```
redis> KEYS user*
1. "user:1:name"
2. "user:1:e-mail"
```

Klucze

Pobieranie listy kluczy na podstawie wzorca:

```
redis> KEYS user*
1. "user:1:name"
2. "user:1:e-mail"
redis> KEYS user:*:name
1. "user:1:name"
```

Wygasanie kluczy

Wygasanie kluczy

```
redis> EXPIRE foo 2
(integer) 1
```

Wygasanie kluczy

```
redis> EXPIRE foo 2
(integer) 1
```

2 sekundy później...

Wygasanie kluczy

```
redis> EXPIRE foo 2
(integer) 1
```

2 sekundy później...

```
redis> GET foo
(nil)
```

Obsługiwane typy danych

Obsługiwane typy danych

- Ciąg znaków (*String*)

Obsługiwane typy danych

- Ciąg znaków (*String*)
- Lista (*List*)

Obsługiwane typy danych

- Ciąg znaków (*String*)
- Lista (*List*)
- Zbiór (*Set*)

Obsługiwane typy danych

- Ciąg znaków (*String*)
- Lista (*List*)
- Zbiór (*Set*)
- Zbiór sortowany (*Sorted set*)

Obsługiwane typy danych

- Ciąg znaków (*String*)
- Lista (*List*)
- Zbiór (*Set*)
- Zbiór sortowany (*Sorted set*)
- Hash (*Hash*)

Ciąg znaków

Przydatne operacje na ciągach znaków:

Ciąg znaków

Przydatne operacje na ciągach znaków:

- SET, GET

Ciąg znaków

Przydatne operacje na ciągach znaków:

- SET, GET, MSET, MGET

Ciąg znaków

Przydatne operacje na ciągach znaków:

- SET, GET, MSET, MGET
- INCR, DECR

Ciąg znaków

Przydatne operacje na ciągach znaków:

- SET, GET, MSET, MGET
- INCR, DECR, INCRBY, DECRBY

Ciąg znaków

Przydatne operacje na ciągach znaków:

- SET, GET, MSET, MGET
- INCR, DECR, INCRBY, DECRBY
- SETEX

Ciąg znaków

Przydatne operacje na ciągach znaków:

- SET, GET, MSET, MGET
- INCR, DECR, INCRBY, DECRBY
- SETEX
- SETNX

Ciąg znaków – przykład licznika

Ciąg znaków – przykład licznika

```
redis> SET foo 0
OK
```

Ciąg znaków – przykład licznika

```
redis> SET foo 0
OK
redis> INCR foo
(integer) 1
```

Ciąg znaków – przykład licznika

```
redis> SET foo 0
OK
redis> INCR foo
(integer) 1
redis> INCRBY foo 2
(integer) 3
```

Ciąg znaków – przykład licznika

```
redis> SET foo 0
OK
redis> INCR foo
(integer) 1
redis> INCRBY foo 2
(integer) 3
redis> GET foo
"3"
```

Ciąg znaków – przykład

Ciąg znaków – przykład

```
redis> SETEX 60 token h4x0rz
OK
```

Lista

Przydatne operacje na listach:

Lista

Przydatne operacje na listach:

- LPUSH, RPUSH

Lista

Przydatne operacje na listach:

- LPUSH, RPUSH
- LPOP, RPOP

Lista

Przydatne operacje na listach:

- LPUSH, RPUSH
- LPOP, RPOP
- LRANGE

Lista

Przydatne operacje na listach:

- LPUSH, RPUSH
- LPOP, RPOP
- LRANGE
- LTRIM

Lista

Przydatne operacje na listach:

- LPUSH, RPUSH
- LPOP, RPOP
- LRANGE
- LTRIM
- RPOPLPUSH

Lista – przykład

Lista – przykład

```
redis> LPUSH newset a
```

Lista – przykład

```
redis> LPUSH newset a
redis> LRANGE newset 0 -1
1. "a"
```

Lista – przykład

```
redis> LPUSH newset a
redis> LRANGE newset 0 -1
1. "a"
redis> LPUSH newset b
```

Lista – przykład

```
redis> LPUSH newset a
redis> LRANGE newset 0 -1
1. "a"
redis> LPUSH newset b
redis> RPUSH newset c
```

Lista – przykład

```
redis> LPUSH newset a
redis> LRANGE newset 0 -1
1. "a"
redis> LPUSH newset b
redis> RPUSH newset c
redis> LRANGE newset 0 -1
1. "b"
2. "a"
3. "c"
```

Lists – przykład kolejki

Lists – przykład kolejki

```
redis> RPOPLPUSH newset newset
"c"
```

Lists – przykład kolejki

```
redis> RPOPLPUSH newset newset
"c"
redis> LRAGNE newset 0 -1
1. "c"
2. "b"
3. "a"
```

Lists – przykład kolejki

```
redis> RPOPLPUSH newset newset
"c"
redis> LRAGNE newset 0 -1
1. "c"
2. "b"
3. "a"
redis> RPUSH newset d
```

Lists – przykład kolejki

```
redis> RPOPLPUSH newset newset
"c"
redis> LRAGNE newset 0 -1
1. "c"
2. "b"
3. "a"
redis> RPUSH newset d
redis> RPOPLPUSH newset newset
"d"
```

List – przykład listy ograniczonej

List – przykład listy ograniczonej

```
redis> LLEN newset
(integer) 4
```

List – przykład listy ograniczonej

```
redis> LLEN newset
(integer) 4
redis> LPUSH newset e
```

List – przykład listy ograniczonej

```
redis> LLEN newset
(integer) 4
redis> LPUSH newset e
redis> LTRIM newset 0 3
```

List – przykład listy ograniczonej

```
redis> LLEN newset
(integer) 4
redis> LPUSH newset e
redis> LTRIM newset 0 3
redis> LRANGE newset 0 -1
1. "e"
2. "d"
3. "c"
4. "b"
```

Zbiór

Przydatne operacje na zbiorach:

Zbiór

Przydatne operacje na zbiorach:

- SADD, SREM

Zbiór

Przydatne operacje na zbiorach:

- SADD, SREM
- SMEMBERS, SRANDMEMBER, SPOP

Zbiór

Przydatne operacje na zbiorach:

- SADD, SREM
- SMEMBERS, SRANDMEMBER, SPOP
- SISMEMBER

Zbiór

Przydatne operacje na zbiorach:

- SADD, SREM
- SMEMBERS, SRANDMEMBER, SPOP
- SISMEMBER
- SDIFF, SDIFFSTORE

Zbiór

Przydatne operacje na zbiorach:

- SADD, SREM
- SMEMBERS, SRANDMEMBER, SPOP
- SISMEMBER
- SDIFF, SDIFFSTORE
- SINTER, SINTERSTORE

Zbiór

Przydatne operacje na zbiorach:

- SADD, SREM
- SMEMBERS, SRANDMEMBER, SPOP
- SISMEMBER
- SDIFF, SDIFFSTORE
- SINTER, SINTERSTORE
- SUNION, SUNIONSTORE

Zbiór – przykład

Zbiór – przykład

$A = \{ 1, 2 \}, B = \{ 2, 3 \}$

Zbiór – przykład

```
A = { 1, 2 }, B = { 2, 3 }
redis> SDIFF A B
1. "1"
```

Zbiór – przykład

```
A = { 1, 2 }, B = { 2, 3 }
redis> SDIFF A B
1. "1"
redis> SINTER A B
1. "2"
```

Zbiór – przykład

```
A = { 1, 2 }, B = { 2, 3 }
redis> SDIFF A B
1. "1"
redis> SINTER A B
1. "2"
redis> SUNION A B
1. "3"
2. "1"
3. "2"
```

Zbiór sortowany

Przydatne operacje:

Zbiór sortowany

Przydatne operacje:

- ZADD, ZREM

Zbiór sortowany

Przydatne operacje:

- ZADD, ZREM
- ZINCRBY

Zbiór sortowany

Przydatne operacje:

- ZADD, ZREM
- ZINCRBY
- ZSCORE, ZRANK, ZREVRANK

Zbiór sortowany

Przydatne operacje:

- ZADD, ZREM
- ZINCRBY
- ZSCORE, ZRANK, ZREVRANK
- ZRANGE, ZREVRANGE

Zbiór sortowany

Przydatne operacje:

- ZADD, ZREM
- ZINCRBY
- ZSCORE, ZRANK, ZREVRANK
- ZRANGE, ZREVRANGE
- ZRANGEBYSCORE, ZREVRANGEBYSCORE

Zbiór sortowany – przykład

Zbiór sortowany – przykład

```
redis> ZADD days 1 Monday
```

Zbiór sortowany – przykład

```
redis> ZADD days 1 Monday
redis> ZADD days 2 Tuesday
```

Zbiór sortowany – przykład

```
redis> ZADD days 1 Monday
redis> ZADD days 2 Tuesday
[...]
```

Zbiór sortowany – przykład

```
redis> ZADD days 1 Monday
redis> ZADD days 2 Tuesday
[...]
redis> ZRANGE days 0 -1
1. "Monday"
2. "Tuesday"
[...]
7. "Sunday"
```

Hash

Przydatne operacje na hashach:

Hash

Przydatne operacje na hashach:

- HSET, HMSET

Hash

Przydatne operacje na hashach:

- HSET, HMSET
- HGET, HMGET

Hash

Przydatne operacje na hashach:

- HSET, HMSET
- HGET, HMGET
- HDEL, HEXISTS

Hash

Przydatne operacje na hashach:

- HSET, HMSET
- HGET, HMGET
- HDEL, HEXISTS
- HGETALL, HKEYS, HVALS

Hash – przykład

Hash – przykład

```
redis> HSET user:1 name Bob
```

Hash – przykład

```
redis> HSET user:1 name Bob
redis> HSET user:1 email bob@ex
```

Hash – przykład

```
redis> HSET user:1 name Bob
redis> HSET user:1 email bob@ex
redis> HGET user:1 name
"Bob"
```

Hash – przykład

```
redis> HSET user:1 name Bob
redis> HSET user:1 email bob@ex
redis> HGET user:1 name
"Bob"
redis> HGETALL user:1
1. "name"
2. "Bob"
3. "email"
4. "bob@example.com"
```

Transakcje

Transakcje w Redis pozwalają zgrupować wiele operacji i wykonać je jako jedną atomiczną operację

Transakcje

Polecenia operujące na transakcjach:

Transakcje

Polecenia operujące na transakcjach:

- MULTI, EXEC, DISCARD

Transakcje

Polecenia operujące na transakcjach:

- MULTI, EXEC, DISCARD
- WATCH, UNWATCH

Transakcje – przykład

Transakcje – przykład

```
redis> MULTI
```

Transakcje – przykład

```
redis> MULTI
redis> SET foo 0
QUEUED
```

Transakcje – przykład

```
redis> MULTI
redis> SET foo 0
QUEUED
redis> INCR foo
```

Transakcje – przykład

```
redis> MULTI
redis> SET foo 0
QUEUED
redis> INCR foo
QUEUED
```

Transakcje – przykład

```
redis> MULTI
redis> SET foo 0
QUEUED
redis> INCR foo
QUEUED
redis> EXEC
1. OK
(integer) 1
```

Transakcje – przykład

Problem

```
redis> GET foo
QUEUED
```

Transakcje – przykład

Rozwiązanie

Dostępne dopier od wersji 2.1.0

```
redis> WATCH foo
redis> GET foo
redis> MULTI
[...]
redis> EXEC
```

Publish/Subscribe

Przydatne polecenia:

Publish/Subscribe

Przydatne polecenia:

- PUBLISH

Publish/Subscribe

Przydatne polecenia:

- PUBLISH
- SUBSCRIBE, UNSUBSCRIBE

Publish/Subscribe

Przydatne polecenia:

- PUBLISH
- SUBSCRIBE, UNSUBSCRIBE
- PSUBSCRIBE, PUNSUBSCRIBE

Pub/Sub – przykład producenta

Pub/Sub – przykład producenta

```
redis> PUBLISH msgs Hi!
(integer) 1
```

Pub/Sub – przykład producenta

```
redis> PUBLISH msgs Hi!
(integer) 1
redis> PUBLISH msgs Hi
(integer) 0
```

Pub/Sub – przykład konsumenta

Pub/Sub – przykład konsumenta

```
redis> SUBSCRIBE
1. "subscribe"
2. "msgs"
3. (integer) 1
```

Pub/Sub – przykład konsumenta

```
redis> SUBSCRIBE
1. "subscribe"
2. "msgs"
3. (integer) 1

1. "message"
2. "msgs"
3. "Hi!"
```

Pub/Sub – przykład konsumenta

```
redis> SUBSCRIBE
1. "subscribe"
2. "msgs"
3. (integer) 1

1. "message"
2. "msgs"
3. "Hi!"
redis> UNSUBSCRIBE
```

Podsumowanie

Podsumowanie możliwości Redis-a:

Podsumowanie

Podsumowanie możliwości Redis-a:

- Szybka baza danych

Podsumowanie

Podsumowanie możliwości Redis-a:

- Szybka baza danych
- Przechowuje ciągi znaków, listy, zbiory, zbiory sortowane i hashe

Podsumowanie

Podsumowanie możliwości Redis-a:

- Szybka baza danych
- Przechowuje ciągi znaków, listy, zbiory, zbiory sortowane i hashe
- ...i potrafi na nich wykonywać operacje

Podsumowanie

Podsumowanie możliwości Redis-a:

- Szybka baza danych
- Przechowuje ciągi znaków, listy, zbiory, zbiory sortowane i hashe
- ...i potrafi na nich wykonywać operacje
- Zawiera obsługę transakcji

Podsumowanie

Podsumowanie możliwości Redis-a:

- Szybka baza danych
- Przechowuje ciągi znaków, listy, zbiory, zbiory sortowane i hashe
- ...i potrafi na nich wykonywać operacje
- Zawiera obsługę transakcji
- Implementuje Pub/Sub

Koniec

To be continued. . .

Koniec

Pytania?

Odnośniki

- http://redis.io
- http://blog.mjrusso.com/2010/10/17/redis-from-the-ground-up.html
- Ruby gem: redis
- Node.js lib: gh:mranney/node_redis