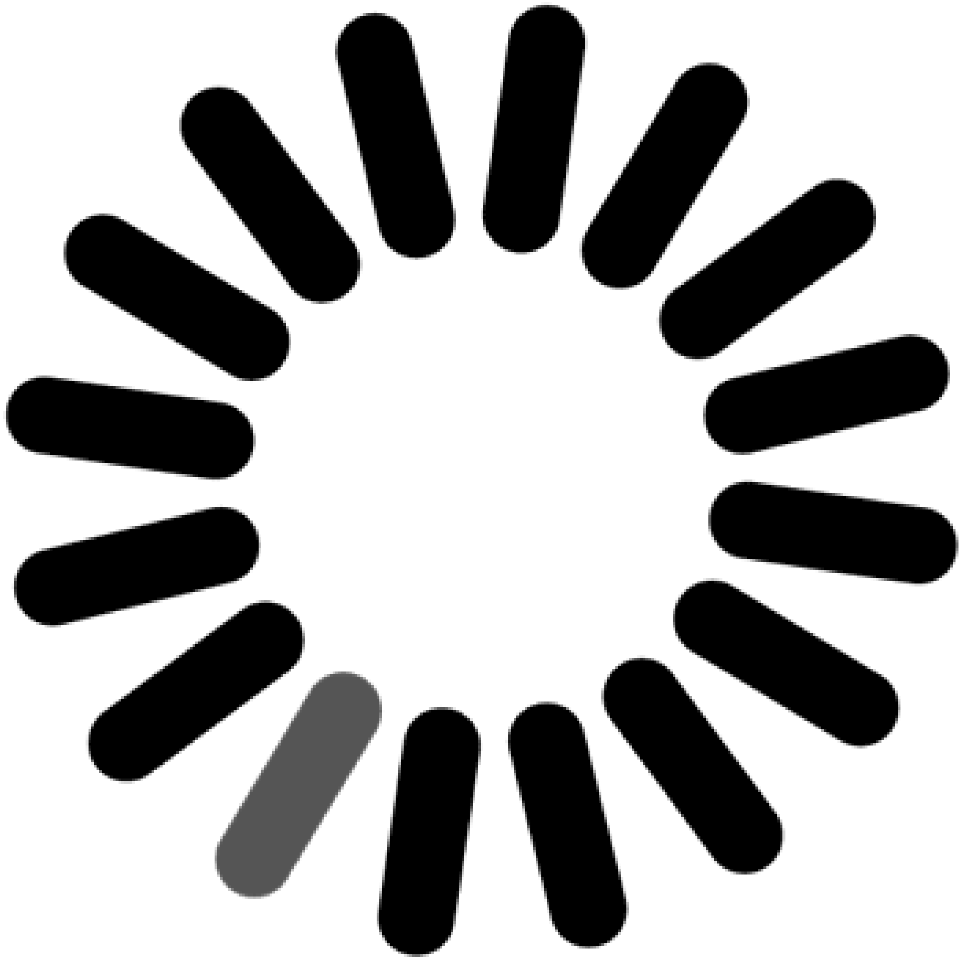
winch



In-app latencies translate into...

frustrated users :(

Common app pattern

Every user action requires downloading data

- **pros**: simple to implement,
- **cons**: poor user experience.

"Speed is a feature"

Minimize in-app latencies…

to make your users happy :)

winch

Build Faster Mobile Apps

Additional benefits

Besides reacting faster, your app:

1. works off-line,
2. stays up-to-date seamlessly.

How does it work?

As simple as 1, 2, 3

1. import your data,
2. drag and drop the Winch iOS framework,
3. call `sync()`.

Concepts

In short

“ *Winch is a **key-value** data store.*

Think of it as...

datastore	~ database
namespace	~ table
key	~ primary key
value	~ data

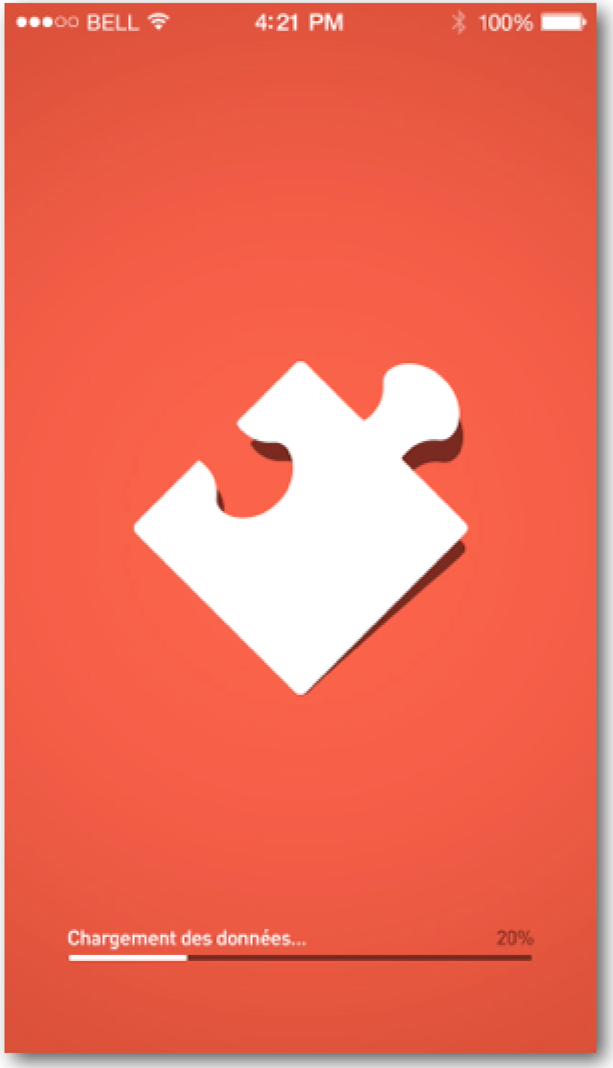
Example:
Snippets for iOS

What is Snippets?

- an iOS app to learn and experience your favorite tech on-the-go,
- it works with Redis so far,
- it is open source,
- it is powered by Winch.

Initiated at AngelHack Paris, October 2013.

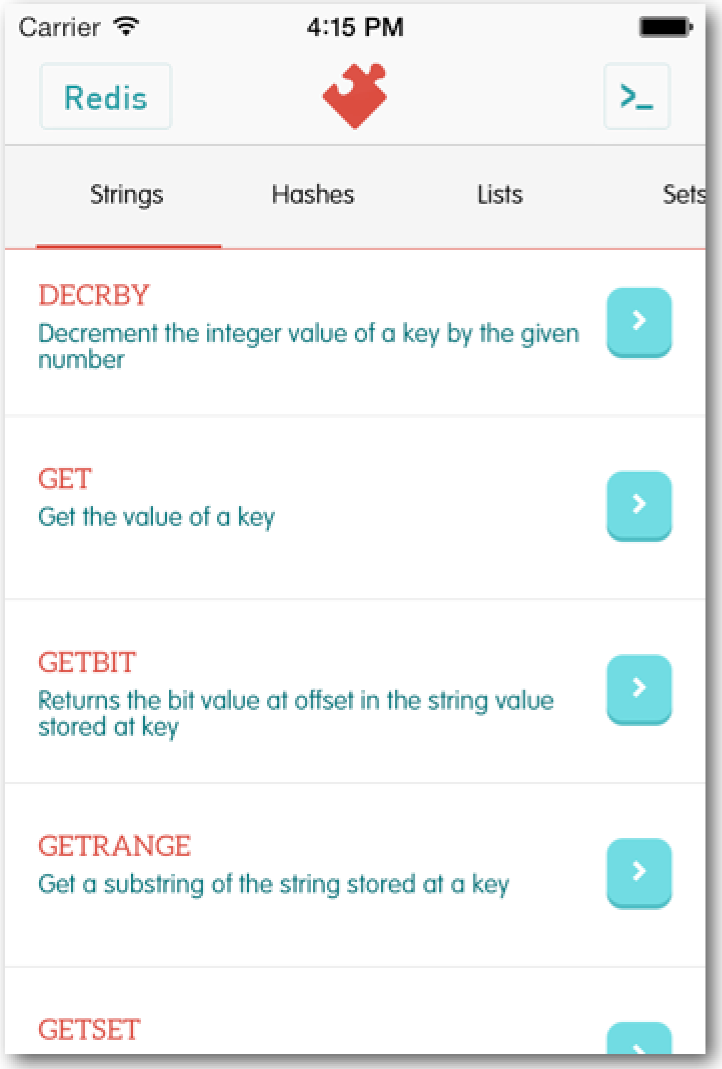
Overview



●●●○○ BELL
4:21 PM
100%
Chargement des données...
20%

Groups

Redis commands



DECRBY
Decrement the integer value of a key by the given number

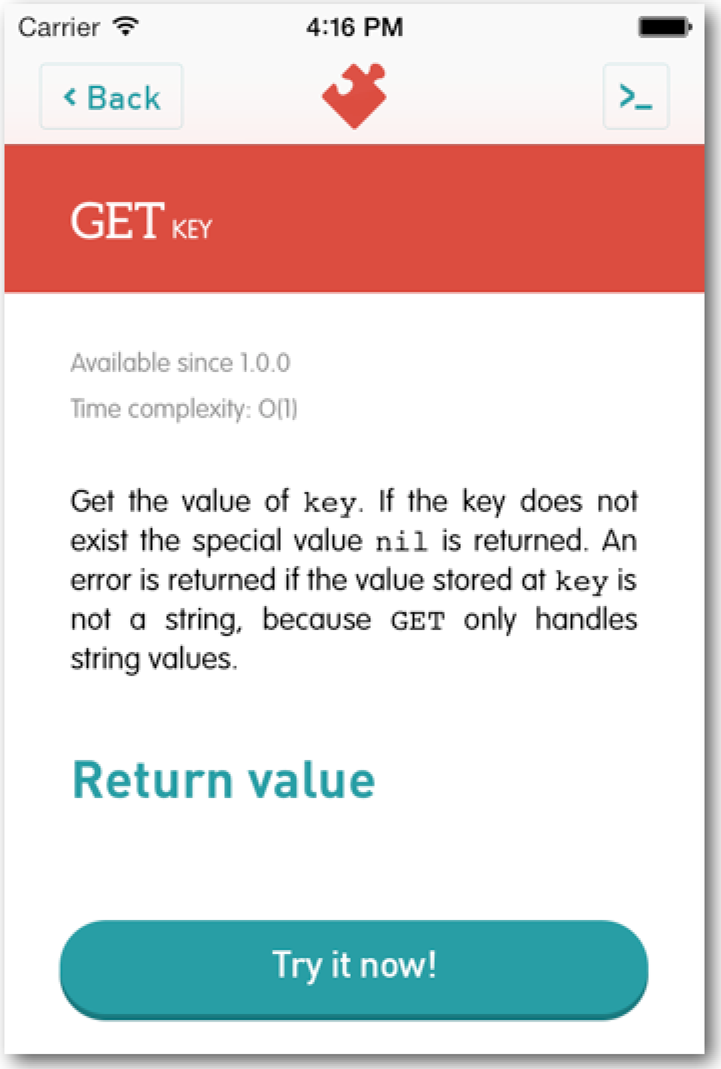
GET
Get the value of a key

GETBIT
Returns the bit value at offset in the string value stored at key

GETRANGE
Get a substring of the string stored at a key

GETSET

Command doc



GET KEY

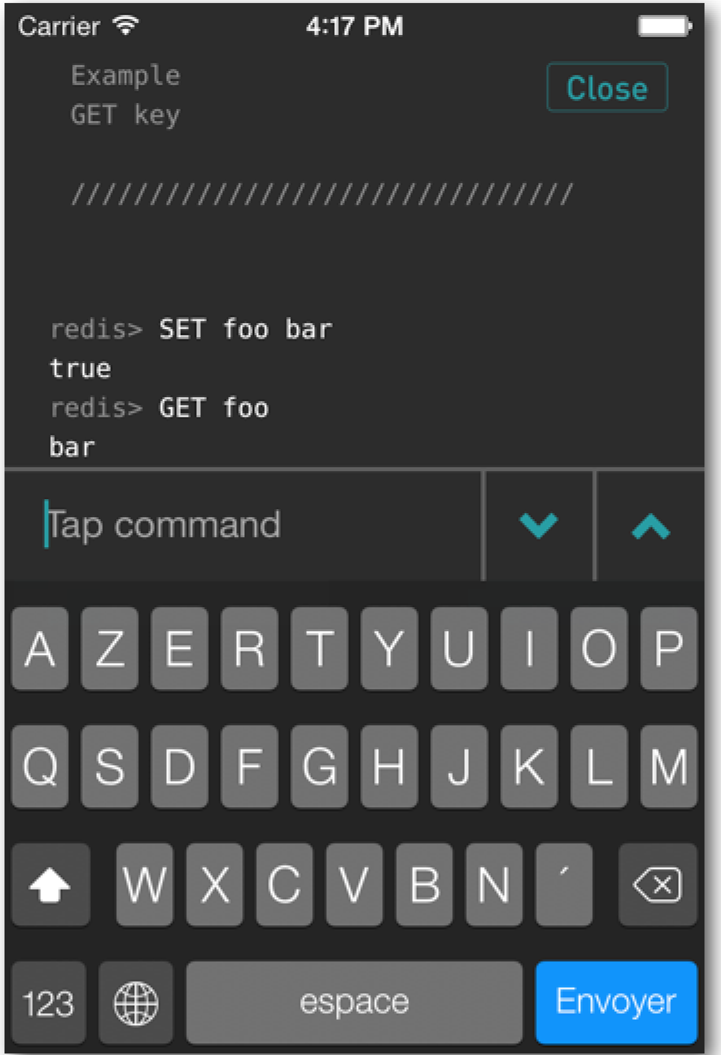
Available since 1.0.0

Time complexity: O(1)

Get the value of `key`. If the key does not exist the special value `nil` is returned. An error is returned if the value stored at `key` is not a string, because `GET` only handles string values.

Return value

In-app Redis console (!)



```
Example
GET key

////////////////////////////////

redis> SET foo bar
true
redis> GET foo
bar
```

Data Model

Commands: `rds:cmds`

- key: command unique ID, e.g `"get"`
- value: JSON doc with command name, summary, CLI sample, e.g:

```
{
"name": "GET",
"summary": "Get the value of a key"
"cli":["GET nonexisting",...]
}
```

Groups: `rds:groups`

- key: integer key, e.g `"0003"`
- value: JSON doc with group name and related commands, e.g:

```
{
"name": "Hashes",
"cmds": ["hdel", "hget", ...]
}
```

HTML docs: `rds:docs`

- key: command unique ID, e.g `"get"`
- value: HTML fragment, e.g:

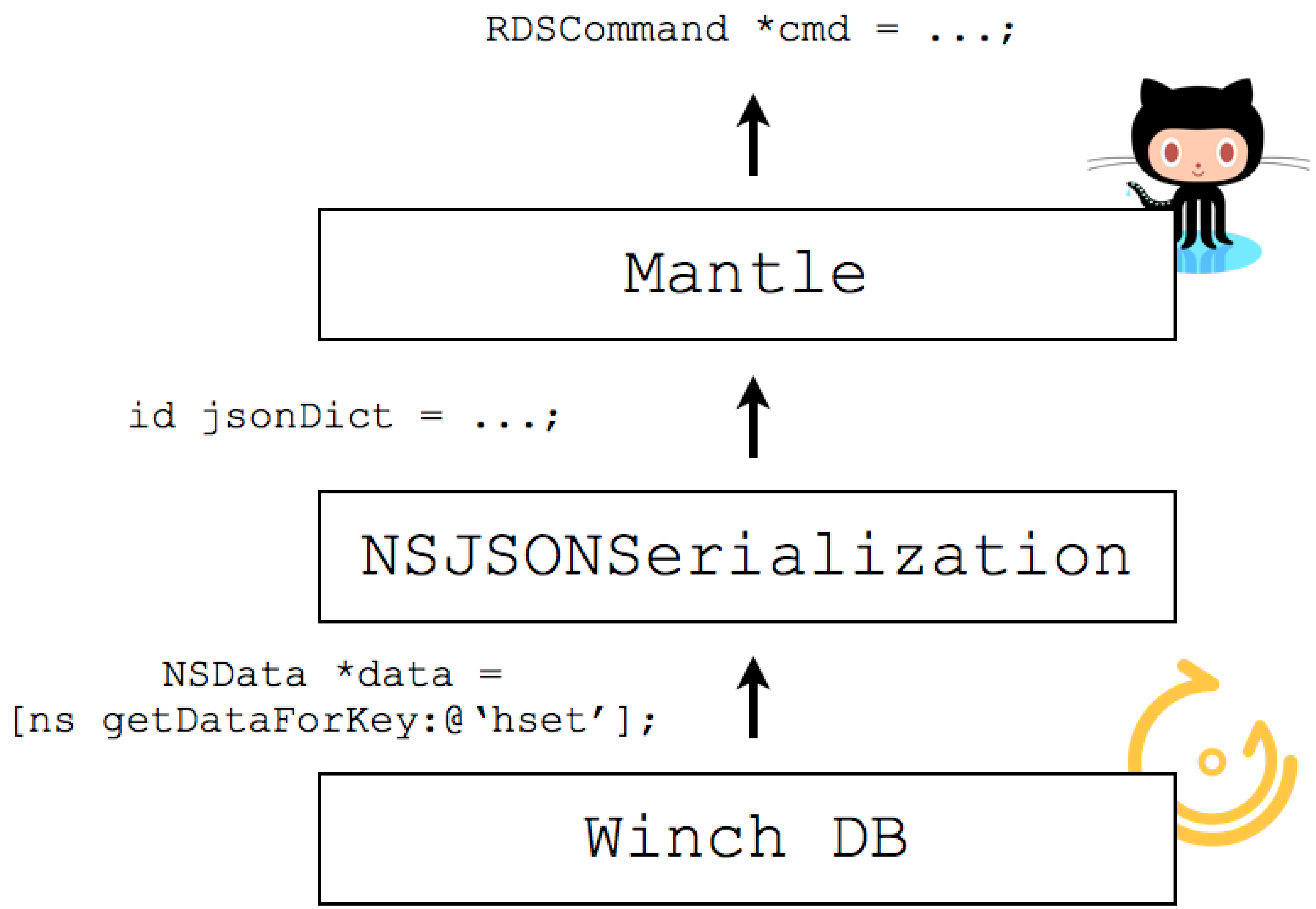
```
<h1>GET key</h1>
<h3>Available since 1.0.0</h3>
...
```

It is used with: `[webView loadHTMLString:html baseURL:nil];`

Model Layer

Use Mantle™, by GitHub

"*Mantle makes it easy to write a simple model layer for your Cocoa or Cocoa Touch application.*



```
RDSCommand *cmd = ...;
```

```
id jsonDict = ...;
```

```
    NSData *data =
[ns getDataForKey:@'hset'];
```

RDSCommand.h

```
#import <Mantle.h>

@interface RDSCommand : MTLModel <MTLJSONSerializing>

@property (nonatomic, copy) NSString *ID;
@property (nonatomic, copy, readonly) NSString *name;
@property (nonatomic, copy, readonly) NSString *summary;
@property (nonatomic, copy, readonly) NSArray *cli;

@end
```

RDSCommand.m

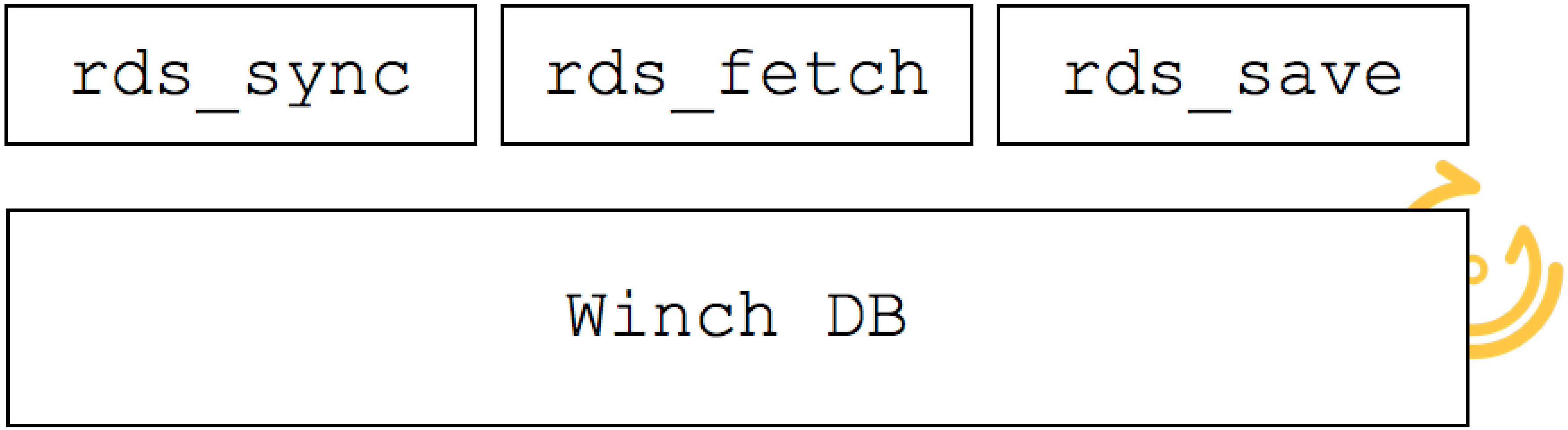
```
@implementation RDSCommand

+ (NSDictionary *)JSONKeyPathsByPropertyKey
{
    return @{
        @"ID": NSNull.null
    };
}

@end
```

Winch Category

Write app specific helpers

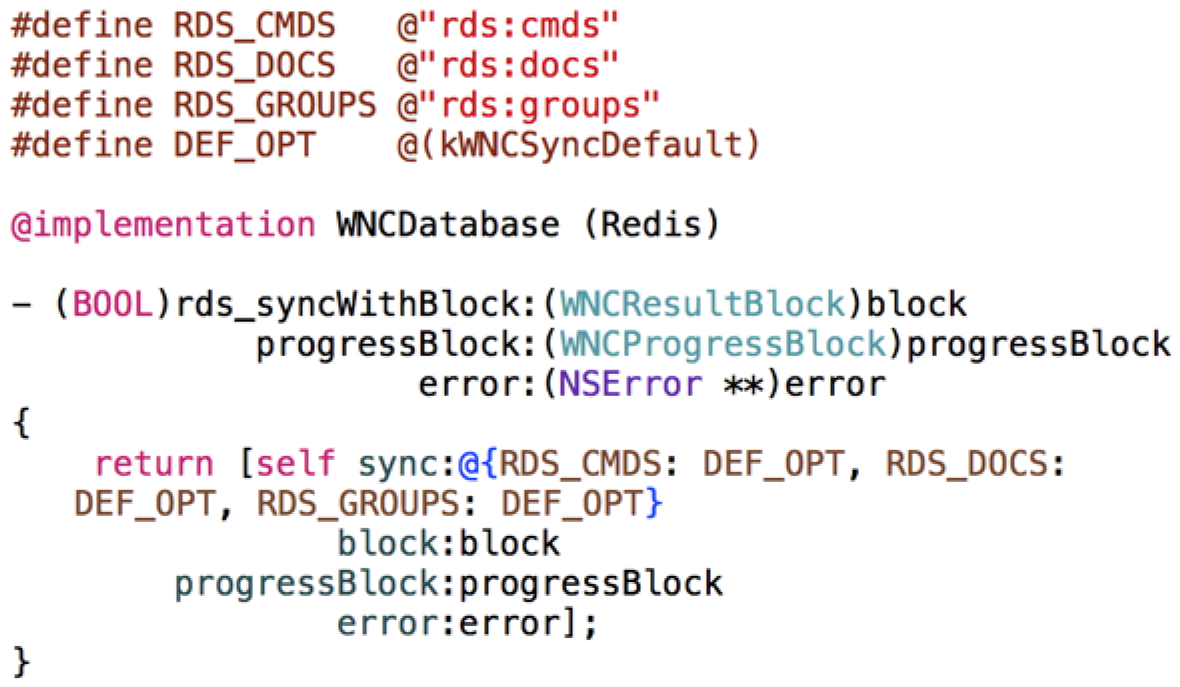
> *Create a `WNCDatabase+MyApp` category and implement custom extensions for your app data model.*

Overview



Sync

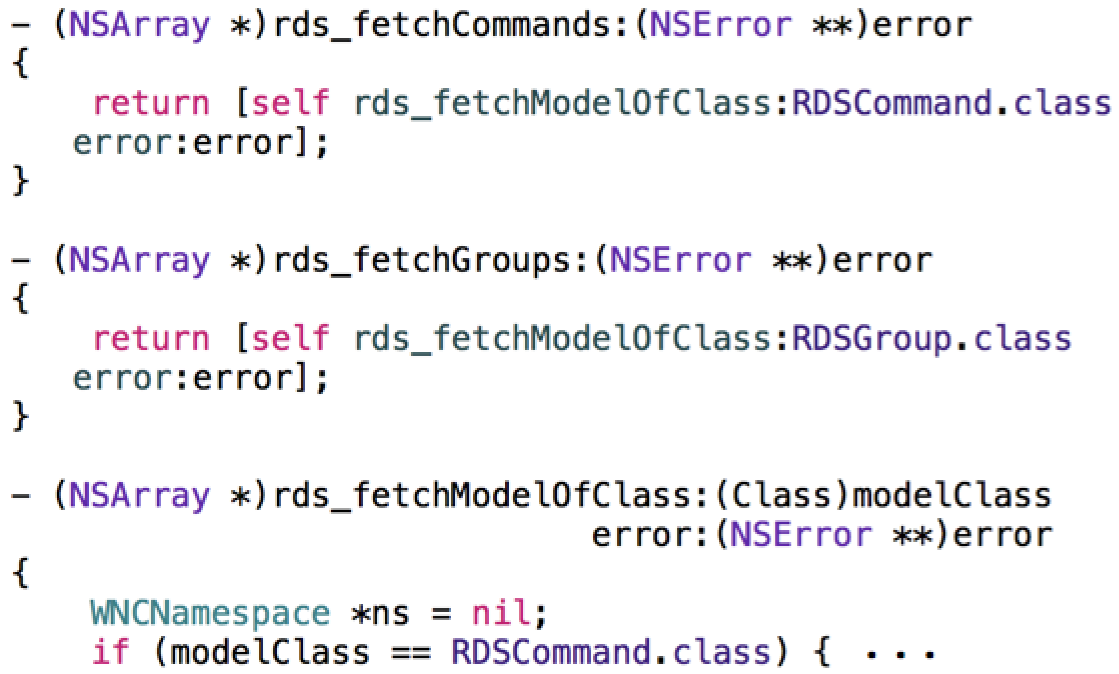
Abstract your namespaces under a custom sync method:



```
#define RDS_CMDS   @"rds:cmds"
#define RDS_DOCS   @"rds:docs"
#define RDS_GROUPS @"rds:groups"
#define DEF_OPT    @(kWNCSyncDefault)

@implementation WNCDatabase (Redis)

- (BOOL)rds_syncWithBlock:(WNCResultBlock)block
            progressBlock:(WNCProgressBlock)progressBlock
                    error:(NSError **)error
{
    return [self sync:@{RDS_CMDS: DEF_OPT, RDS_DOCS:
   DEF_OPT, RDS_GROUPS: DEF_OPT}
                block:block
        progressBlock:progressBlock
                error:error];
}
```

Fetch

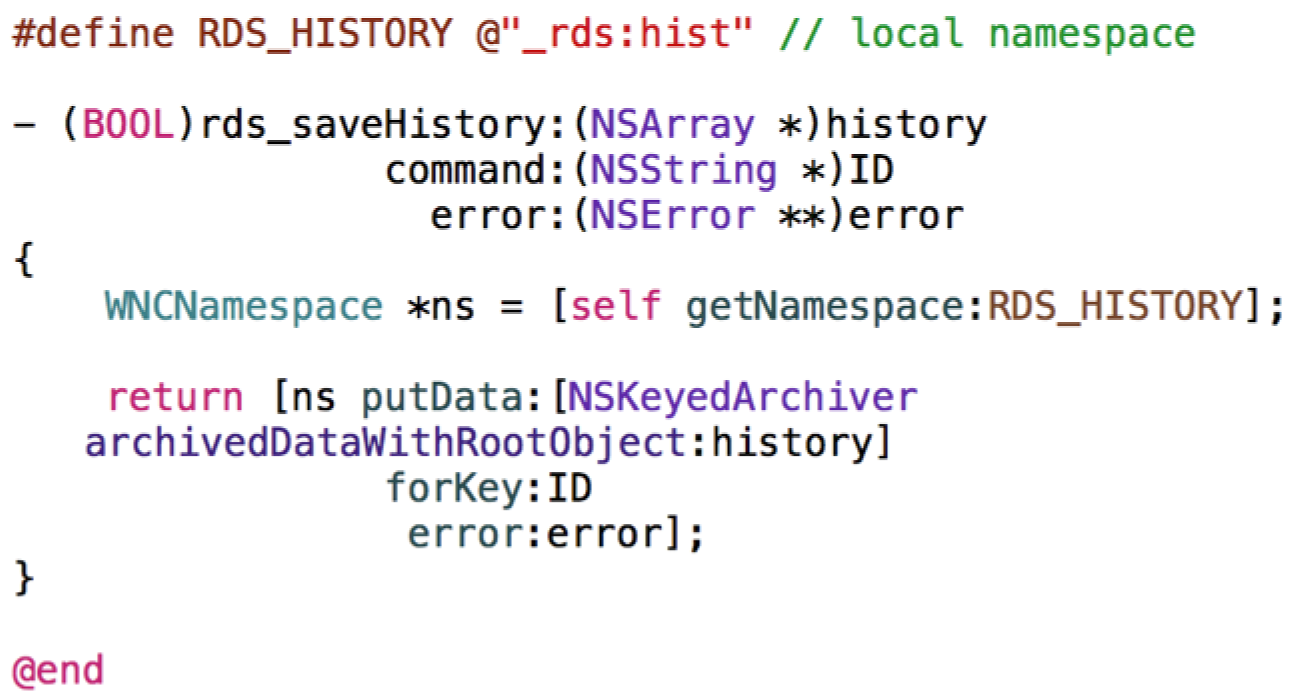
Get a collection of model objects via a single call:



```
- (NSArray *)rds_fetchCommands:(NSError **)error
{
    return [self rds_fetchModelOfClass:RDSCommand.class
   error:error];
}

- (NSArray *)rds_fetchGroups:(NSError **)error
{
    return [self rds_fetchModelOfClass:RDSGroup.class
   error:error];
}

- (NSArray *)rds_fetchModelOfClass:(Class)modelClass
                             error:(NSError **)error
{
    WNCNamespace *ns = nil;
    if (modelClass == RDSCommand.class) { ...
```

Save

Persist local data with a local namespace:



```
#define RDS_HISTORY @"_rds:hist" // local namespace

- (BOOL)rds_saveHistory:(NSArray *)history
                command:(NSString *)ID
                  error:(NSError **)error
{
    WNCNamespace *ns = [self getNamespace:RDS_HISTORY];

    return [ns putData:[NSKeyedArchiver
   archivedDataWithRootObject:history]
                forKey:ID
                 error:error];
}

@end
```

Questions?

Thanks :)