Implementation of Redis-like storage layer in Caché

Software requirements specification

(short project description)

3/9/2017 Marek Bernád

Description of project future features:

1. Caché ObjectScript API:

Redis is an open source(BSD licensed) in-memory key-value store used as a database, cache and message broker. It supports data structures such as [strings](https://redis.io/topics/data-types-intro#strings), [hashes](https://redis.io/topics/data-types-intro#hashes), [lists](https://redis.io/topics/data-types-intro#lists), [sets](https://redis.io/topics/data-types-intro#sets), [sorted sets](https://redis.io/topics/data-types-intro#sorted-sets) with range queries and lot more.

I will implement Redis commands in Caché ObjectScript using globals for the actual storage of data. The commands will be exposed to the developers via methods in the “Redis.Commands” class. This API will be easily extensible.

These commands will be available:

GET – will return value of the variable

DECR – will decrement value of the specific variable by one

DEL – will permanently remove chosen variable

HELP – will return help information for particular command

INCR – will increase value of the variable by one

SET – will set up new variable into Caché global

DECRBY – will decrease value of the variable by specific value

INCRBY – will increase value of the variable saved by specific value

EXISTS – will return information if tested variable still exist

EXPIRE – will set variable for specific amount of time in seconds

PEXPIRE – will set variable for specific amount of time in miliseconds

TTL – will return the remaining time to live of the variable in seconds

PTTL – will return the remaining time to live of the variable in miliseconds

TYPE – will determine a type of the variable

APPEND – will append new data to the variable and will not delete existing

STRLEN – will return length of the value stored under the key

TIME – will return current server time

PING – will ping a server

Description of all the commands above and options that they use can be found here: <https://redis.io/commands>

1. RESP TCP-based access:

Most of the existing applications using Redis connect to Redis with help of the TCP-based protocol called RESP (described at <https://redis.io/topics/protocol)> or some of the libraries (listed at <https://redis.io/clients)> which implement the client-side of this protocol internally.

In order to be able to demonstrate replaceability of Redis by my Caché implementation, I will implement the server-side of the protocol in Caché.

1. Redis Shell:

Creation of the shell environment for executing the Redis commands. I will implement a class „Redis.Shell“ with multiple methods. The first method - „Execute()“ - will accept Redis textual command as a parameter and as a result, it will return the string result value of the command execution exactly as Redis would. Another method will be implemented for running the shell itself that will evaluate the Redis commands interactively.

1. REST API:

The Redis commands implemented in Caché will also be available via a REST API. For every command, there will be a URL path for executing it directly. The API will also offer a URL for executing arbitrary Redis command text.

1. Special Feature – timing variable removal:

Redis provides the possibility of specifying the expire time for individual data values. Users are able to create any variables or other data that will exist only for a specific amount of time. After the time expires, the data is automatically removed.

In order to support this feature, I will design and implement an algorithm that will ensure that only non-expired variables are returned and no data which expired more than 10 second ago is actually stored in the database.

1. Source code:

The source code will be provided in the form of UDL files in a directory structure following the package structure.

1. Installation:

I will implement a one-step installer solution for installing the project in Caché or create a step-by-step installation instructions manual.

1. Performance comparison:

I will create a benchmark or use an existing one (such as the redis-benchmark utility described at <https://redis.io/topics/benchmarks)> in order to compare the performance of Redis and my implementation.

1. Article on the Developer Community:

I will present the project deliverables as well as the benchmark results (including some charts) in an article posted on <https://community.intersystems.com>.