# Redis: String Values

Gradually with time, the web project I am working on became a heavy-data oriented project where each day almost thousands of data-flow, transactions, and database operations were occurring. This led to very serious performance issues. Despite multi layered and multi leveled review and optimizations of code, performance was a bottleneck.

In an effort to improve performance, thorough research started to find a tool, package or solution that not only can easily integrate to the currently heavy and complicated business-logic implementation but also can provide scalability, extensibility and easy manageability to overall data related operations. While searching, we bumped into ‘Redis’.

Prerequisites:

* Basic and primitive knowledge of command line
* Basic and primitive understanding of caches
* Basic and primitive understanding of programming terminologies like data types, variables etc.
* Should know how to install and activate Microsoft service.

**What is Redis?**

Redis is simply, an in-memory **key-value** data structure **store** that can be used as another layer between **data access layer** and **database layer** in any application. It supports **caching** of data and provides on-disk **persistence** options. Despite it offers a large set of diverse optimization, replication and caching features, it is extremely **light-weight**. It can be easily integrated with any existing applications.

What makes redis different from other conventional key-value stores is that Redis keys can hold values from simple to complex data structure types like:

1. String -> Covered in this article
2. Lists
3. Sets
4. Sorted Sets
5. Hashes
6. Bit Arrays or Bitmaps
7. Hyperloglogs

The Redis-key, however, is of String type and is binary safe that can safely map any value from simple strings to images to complex of hyper log logs

In this article, we will dive into basics of Redis and use **String values with the command line.** There is even an online interactive tutorial by redis here: <http://try.redis.io/>. (We’ll also dive on using Redis (Stackexchange.Redis) in our C# application for optimization and caching in other articles to come.)

**Installing Redis**

Well, if you are a windows user like me – it will be hard to find an installer of Redis for windows because there doesn’t exist one (according to their official website); however, the same official website also informs that Microsoft OpenTech has developed an executable that can run Redis on Windows. You can get the setup file from [here](https://github.com/MicrosoftArchive/redis/releases). Run it, install it, keep Redis service on – and open. After proper installation, you should see Redis service running in Task Manager.

Before proceeding, it is important to note that Redis is a key-value store, this means all the sections below onwards and in upcoming articles will focus on ‘Key’ and ‘Value’. It is as simple as that – make a key, and associate a value to it. That’s it. All commands revolve around creating, fetching, updating and deleting key-values.

**Getting Started – Redis CLI – String commands**

As mentioned in the previous section, Redis keys are of String Data Type and when storing a string value it becomes a simple “string to string” mapping.

So let’s get started using Redis CLI

1. **SET** - Store any String Value

**Syntax**: SET Key-Name String-Value [EX-Seconds] [PX-milliseconds] [NX|XX]

* + You can set any key name and store any string value, even a jpeg image (max 512 MB).
  + If there is a need to expire key’s value after some time it can be done by using EX command. Alternatively, key’s value can persist, using PX command
  + If a value is already set to a key that already contains some other value then it will be overridden.
  + If NX option is used, the new value won’t be assigned to a key that already has a value assigned. And XX option does its complete reverse. It allows assigning a value to only that key that has any value already assigned.

1. **GET** - Retrieve any String Value

**Syntax**: GET Key-Name

* + Returns value for input key-name

1. **GETSET** – Sets new value for a key and returns previously assigned value in response

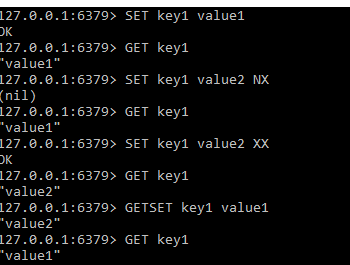
**Syntax**: GETSET key-name value

1. **MSET** – Sets multiple key values in one command.

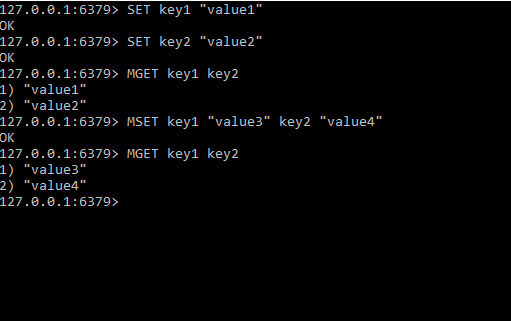
**Syntax**: MSET key-name1 value1 [key-name2 value2 key-name3 value3 …]

1. **MGET** – Gets values for multiple keys in one command.

**Syntax**: MGET key-name1 [key-name2 key-name3…]



1 Usage of SET, GET, NX, XX, GETSET commands



2 Usage of MSET and MGET

[This](https://redis.io/commands#string) is the complete list of all CLI commands for String data type in redis.

That’s it for this post, in next article we will look for commands of List and set values.