Overview

The framework allows to work with six types of cache differing by data storage: file caching, caching in memory on the basis of APC, Memcache or Redis extensions and caching in PHP session.

Cache settings are specified in the configuration variable **cache** of the configuration file. The value of this variable is an associative array. Elements of this array define cache type and additional parameters of the cache. All available cache configurations are shown below (the example of the configuration INI-file):

```
; The cache based on the file system.

[cache]
; The cache type.
type          = "file"
; The cache directory or its alias.
; If the cache directory is not defined the alias "cache" is used.
directory     = "cache"
; The real number indicating the probability (in percent) of calling of
; the garbage collector of the cache.
gcProbability = 33.333

...

; The cache based on the APC extension.

[cache]
; The cache type.
type          = "apc"

...

; The cache based on the Memcached extension.

[cache]
; The cache type.
type          = "memory"
; The boolean parameter which responsible for the compressing of
; data in memory. The default value is FALSE.
compress      = 1
; The array of servers for connection to the memcached extension.
; More information about the structure of this array you can find here:
; http://php.net/manual/ru/memcache.addserver.php
servers       = "..."

...

; The cache based on the Redis extension or
; the direct access to the Redis server.

[cache]
; The cache type. Can be "phpredis" (if we want to use the Redis PHP-extension
; or "redis" (if we want to use the direct connection to the Redis server).
type          = "phpredis" ; or "redis"
; The host or path to a unix domain socket for a redis connection, optional.
; The default value is "127.0.0.1"
host          = "112.98.52.2"
; The port for a connection, optional. The default value is 6379.
port          = 6379
; The connection timeout, in seconds. The default value is 0.
timeout       = 1
; The password for the server authentication, optional.
password      = ""
; The number of the redis database to use. The default value is 0.
database      = 0
```

```
...

; The cache based on the PHP session.

[cache]
; The cache type.
type          = "session"
```

These parameters determine the behavior of the internal cache of the framework that used by different modules of the framework for caching required data. You can get an object of the internal cache of the framework as follows:

```
$cache = CB::getInstance()->getCache();
```

Also you can create own cache object using static method **getInstance()** of classs **ClickBlocks\Cache\Cache**. The method takes the cache type (**memory**, **apc**, **redis**, **phpredis** or **file**) as the first parameter and an array of additional parameters of the cache as the second one.

```
$cache = Cache::getInstance('memory', ['compress' => true]);
```

If method **getInstance()** is invoked without parameters then the cache settings defined in the appropriate configuration variable is used.

In addition there is the possibility to replace the internal cache of the framework:

```
// Creates new cache object
$cache = Cache\Cache::getInstance('apc');
// Replaces the internal framework cache by new one.
CB::getInstance()->setCache($cache);
```

Features of different types of caching

There are different nuances of usage of different cache types. In particular, different types of cache have their maximum possible cache lifetime. For types **apc** and **memory** maximum cache lifetime is one month. After this time all data will be destroyed. The file cache has unlimited cache lifetime. However, for practical reasons this time is set to one year.

You can set and read cache lifetime using the appropriate method:

```
// Gets the cache object.
$cache = CB::getInstance()->getCache();
// Finds out the maximum cache lifetime.
echo $cache->getVaultLifeTime();
// Sets new maximum cache lifetime (in seconds).
$cache->setVaultLifeTime(108000);
```

The second feature of the use of different types of caches is a garbage collector. By saying the garbage collector we means a mechanism of removing expired data from the cache. For caching systems store data in memory (types **memory**, **apc** or **redis**) garbage collector is implemented at the level of the appropriate extension php. Garbage collector of the file cache is implemented at the level of the framework. You can call it in your code, if necessary:

```
// Gets the cache object.
$cache = CB::getInstance()->getCache();
// Call garbage collector (100% probability).
$cache->gc();
// Call garbage collector with 10% probability.
$cache->gc(10);
```

> Besides of deleting expired cache data **gc()** also normalizes the vault of group keys by removing keys of the expired data.

File cache stores all its data in a directory, which can be read or set using the appropriate method:

```
// Gets the cache object.
$cache = CB::getInstance()->getCache();
// Reads the current cache directory.
```

```
echo $cache->getDirectory();
// Sets new cache directory.
$cache->setDirectory('/path/to/new/directory');
```