ClickBlocks\Cache\Cache

General information

Inheritance	no
Child classes	ClickBlocks\Cache\File, ClickBlocks\Cache\Memory, ClickBlocks\Cache\APC, ClickBlocks\Cache\PHPRedis, ClickBlocks\Cache\Redis, ClickBlocks\Cache\Session
Interfaces	Countable
Source	Framework/cache/cache.php

An abstract class is the base class of all framework's classes, which are used for caching data. Provides methods for working with groups of cached data, as well as factory method to get an instance of the cache of a given type.

```
// Gets an instance of the file cache.
$cache = Cache::getInstance('file');
// Stores 'some data' in cache during 10 seconds with identifier 'key' and group name 'foo'
$cache->set('key', 'some data', 10);
// Reads data.
$data = $cache->get('key');
// Removes cache by its identifier.
$cache->remove('key');
// Checks that the data has been deleted.
var_dump($cache->isExpired('key'));
// Creates three cache blocks that combined to one group 'foo'
$cache->set('k1', 'v1', 5, 'foo');
$cache->set('k2', 'v2', 5, 'foo');
$cache->set('k3', 'v3', 5, 'foo');
// Gets all data of group 'foo'.
$data = $cache->getByGroup('foo');
// Removes all group data.
$cache->cleanByGroup('foo');
```

Public static methods

getInstance()

```
public static ClickBlocks\Cache\Cache getInstance(string $type = null, array $params = null)
```

$type	string	type of class for caching.
$params	array	cache parameters.

Returns the object of a certain class of caching. For each type of cache can transfer its parameters as the second argument to the method. Below is a list of parameters depending on the cache type:

1. Type **File**:
   - **directory** - cache directory. If such directory does not exist the framework will try to create it.
2. Type **Memory**:
   - **servers** - an array of configuration parameters for the memcache servers. To get more information see http://php.net/manual/ru/memcache.addserver.php
   - **compress** - boolean parameter that determines whether the data is compressed before being placed in the cache. The default value is TRUE.
3. Type **APC**. It has no config parameters.
4. Type **PHPRedis**:

    - **host** - host or path to a unix domain socket for a redis connection.
    - **port** - for a connection, optional.
    - **timeout** - the connection timeout, in seconds.
    - **password** - password for server authentication, optional.
    - **database** - number of the redis database to use.
5. Type **Redis**. It has the same parameters as **PHPRedis**.
6. Type **Session**. It has no config parameters.

If the cache type is not specified, the method will attempt to read the cache type and its parameters from a configuration file from a section of the cache. If the section is not definite cache or cache type is not specified in the configuration file, the type of cache will be taken by default - file cache.

isAvailable()

```
public static boolean isAvailable()
```

Returns TRUE, if the cache of the appropriate type is available for use, and FALSE otherwise.

Public non-static methods

set()

```
abstract public void set(string $key, mixed $content, integer $expire, string $group = null)
```

$key	string	unique identifier of caching data.
$content	mixed	data to cache.
$expire	integer	cache lifetime in seconds.
$group	string	cache group name.

Stores data in cache by their unique identifier.

get()

```
abstract public mixed get(string $key)
```

$key	string	unique identifier of caching data.

Returns previously stored data by their identifier.

remove()

```
abstract public void remove(string $key)
```

$key	string	unique identifier of caching data.

Removes data from cache by their identifier.

isExpired()

```
abstract public boolean isExpired(string $key)
```

$key	string	unique identifier of caching data.

Returns TRUE if cache is expired and FALSE otherwise.

clean()

```
abstract public void clean()
```

Removes all data from cache.

gc()

```
public void gc(float $probability = 100)
```

$probability	float	probability of method call in percent.

Garbage collector. Removes all expired cache data and normalizes the vault of group keys.

count()

```
public integer count()
```

Returns number of group keys. This method is part of interface **Countable**. This means that you can apply function **count()** to the cache object.

getVault()

```
public array getVault()
```

Returns key array of all data that combined to groups. Format of this array:

```
[
 'group name' => ['key' => lifetime, 'key' => lifetime, ... ],
 'group name' => ['key' => lifetime, 'key' => lifetime, ... ],
  ...
]
```

getVaultLifeTime()

```
public integer getVaultLifeTime()
```

Returns lifetime (in seconds) of cache vault.

setVaultLifeTime()

```
public integer setVaultLifeTime(integer $vaultLifeTime)
```

$vaultLifeTime	integer	cache vault lifetime in seconds.

Sets cache vault lifetime in seconds.

normalizeVault()

```
public void normalizeVault()
```

Removes keys of the expired data from the key vault.

getByGroup()

```
public array getByGroup(string $group)
```

$group	string	cache group name.

Returns an array of data stored in the cache under the same specified group. Example:

```
// Write the data in the cache, the group of 'my group'
foreach ($i = 1; $i <= 5; $i++) $cache->set('key' . $i, $i, 100, 'my group');
// Gets array of cached data of group 'my group'
print_r($cache->getByGroup('my group'));
// Displays an array like
// ['key1' => 1, 'key2' => 2, 'key3' => 3, 'key4' => 4, 'key5' => 5];
```

cleanByGroup()

```
public void cleanByGroup(string $group)
```

$group	string	cache group name.

Removes group of cached data by its name.

Protected non-static properties

vaultLifeTime

```
protected integer $vaultLifeTime = 31536000
```

The lifetime of the cache IDs of cached data. Lifetime is defined in seconds.

Protected non-static methods

saveKeyToVault()

```
protected void saveKeyToVault(string $key, integer $expire, string $group)
```

$key	string	unique identifier of caching data.
$expire	integer	cache lifetime.
$group	string	cache group name.

Stores the identifier of the cache in the vault of identifiers. This method should be called at the appropriate implementation of the abstract method **set()**.

> If **$group** is equal NULL, **$key** won't be stored in the vault.