FizeCache 参考手册

2020 年 01 月 03 日

Contents

CHAPTER 1

欢迎使用

1.1 欢迎使用

FizeCache 是一个易于扩展的缓存类库。

FizeCache 允许您将缓存存放在任意的位置，无论是数据库、文件、Memcached 还是 Redis。

FizeCache 可以进行处理器选择，因此您可以根据系统环境自行选择 Cache 处理器。

FizeCache 有非常完善的参考文档，且其功能追求简洁明了，相信您会喜欢上这样的缓存类库。

1.1.1 处理器支持

目前 FizeCache 已支持的处理器如下：

- *Database*：数据库，具体可以参考 FizeDb 。
- *File*：文件形式。
- *Memcache*：Memcache 形式。
- *Memcached*：Memcached 形式。
- *Redis*：Redis 形式。

1.1.2 入门三部曲

1. 配置参数

根据参数配置 进行 FizeCache 配置。

2. 设置默认连接或者设置新连接

使用 *new Cache($handler, $config);*进行默认缓存设置，或者 *'Cache::getInstance($handler, $config)* 方法获取新缓存实例

3. 进行缓存操作

FizeCache 简化了缓存的操作，日常您使用的方法如下。

- *Cache::get()*：获取缓存。
- *Cache::set()*：设置缓存。
- *Cache::has()*：判断指定缓存是否存在。
- *Cache::remove()*：删除指定缓存。
- *Cache::clear()*：清空缓存。

1.1.3 入门示例

```
use fize\cache\Cache;

$config = [
        'host'    => '192.168.56.101',
        'port'    => 6379,
        'timeout' => 10,
        'expire'  => 0,
        'dbindex' => 15
];
new Cache('Redis', $config);

Cache::set('cfz', 'hello world!');
$cache1 = Cache::get('cfz');
var_dump($cache1);  //hello world!

Cache::remove('cfz2');
$cache2 = Cache::get('cfz2');
var_dump($cache2);  //null
```

CHAPTER 2

安装说明

2.1 安装说明

FizeCache 的环境要求如下：

- “php” ： “>=5.4.0”
- 如果使用 Database 处理器，请安装 FizeDb。
- 如果使用 File 处理器，请安装 FizeCrypt 和 FizeIo。
- 如果使用 Memcache 处理器，请开启 memcache 扩展。
- 如果使用 Memcached 处理器，请开启 memcached 扩展。
- 如果使用 Redis 处理器，请开启 redis 扩展。

2.1.1 使用 Composer 安装

FizeCache 支持使用 Composer 安装，也是唯一官方推荐的安装方法。

注解： 如果您尚未安装 composer ，请参考 安装 composer 。

使用 阿里云镜像 以提高下载速度及稳定性。

在命令行下面，切换到您的项目根目录下面并执行下面的命令：

```
composer require fize/cache
```

根据需要，选择 FizeCache 处理器。使用 composer 下载依赖或者开启相应扩展。

好了！您现在可以开始使用 FizeCache 了，就是这么简单！~

注解： Fize 项目 (包括所有子项目) 严格遵守 语义化版本 ，您可以放心大胆的使用。

CHAPTER 3

更新日志

3.1 更新日志

- *v2.3.0 (2019-11-18)*：添加常规调用的静态便捷方法。
- *v2.2.1 (2019-09-30)*：修复 File 驱动 remove 方法在缓存不存在时出现文件不存在的错误。
- *v2.2.0 (2019-09-29)*：代码风格优化，引入单元测试，composer 版本依赖确认。
- *v2.1.1 (2019-09-03)*：驱动类的依赖项不再写入 composer.json 文件，防止 composer 报不必要的错误。
- *v2.1.0 (2019-09-03)*：添加方法 Cache::getNew($driver, array $options = []) 用于新建实例以便于单独调用。
- *v2.0.0 (2019-09-02)*：BUG 修复，代码优化。
- *v1.0.0 (2019-08-28)*：发布首个版本。

CHAPTER 4

许可协议

4.1 许可协议

4.1.1 The MIT License (MIT)

注解: 以下为中文译文

4.1.2 MIT 开源许可协议

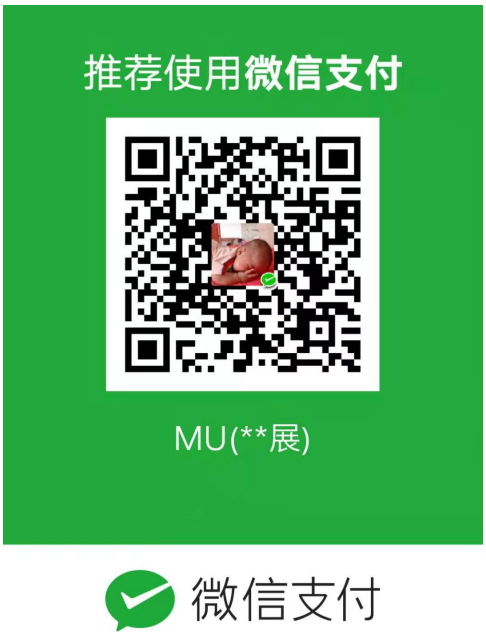
CHAPTER 5

捐赠我们

5.1 捐赠我们

Fize 项目及其下所有子项目目前都为个人维护，坚持开源和免费提供使用。如果您对我们的成果表示认同并且觉得对你有所帮助我们愿意接受来自各方面的捐赠。

使用手机支付宝扫描进行捐赠



使用手机微信扫描进行捐赠

以下是捐赠明细 (截止 2019-11-19)：

- 梁 * 萍 50.00 元
- 董 * 辉 100.00 元
- 曾 * 庆 20.00 元
- 许 * 钦 10.00 元
- 陈 * 88.88 元

CHAPTER 6

处理器配置

6.1 处理器配置

6.1.1 DataBase

数据库处理器配置

参数名	说明	是否可选	默认值
db	数据库配置，含 [‘type’ ， ‘mode’ ， ‘config’] 3 个部分。	否	
table	表名	是	cache
expire	有效时间，以秒为单位,0 表示永久有效。	是	0

注解： 参数 *db* 请参考 FizeDb 参考手册

6.1.2 File

文件处理器配置

参数名	说明	是否可选	默认值
path	缓存文件保存路径	是	‘./data/cache’
expire	有效时间，以秒为单位,0 表示永久有效。	是	0

6.1.3 Memcache

Memcache 处理器配置

参数名	说明	是否可选	默认值
host	Memcache 服务器	是	‘localhost’
port	Memcache 端口	是	11211
timeout	Memcache 超时时间	是	10
pconnect	是否长连接	是	false
debug	是否调试模式	是	false
expire	有效时间，以秒为单位,0 表示永久有效。	是	0

警告： Memcache 官方已停止维护，不建议使用。Memcache 处理器暂未进行单元测试，请根据实际情况酌情使用。

6.1.4 Memcached

Memcached 处理器配置

参数名	说明	是否可选	默认值
servers	Memcached 初始化参数	是	[[‘localhost’ , 11211, 100]]
timeout	Memcached 超时时间	是	10
expire	有效时间，以秒为单位,0 表示永久有效。	是	0

警告： Memcached 处理器暂未进行单元测试，请根据实际情况酌情使用。

6.1.5 Redis

Redis 处理器配置

参数名	说明	是否可选	默认值
host	Redis 服务器	是	'127.0.0.1'
port	Redis 端口	是	6379
timeout	Redis 超时时间	是	0
expire	有效时间，以秒为单位,0 表示永久有效。	是	0

CHAPTER 7

示例参考

7.1 示例参考

7.1.1 初始化

```php
use fize\cache\Cache;

//使用 Cache 静态方法前必须先 Cache 初始化

$config = [
        'host'    => '192.168.56.101',
        'port'    => 6379,
        'timeout' => 10,
        'expire'  => 0,
        'dbindex' => 15
];
new Cache('Redis', $config);

//可以开始使用 Cache 静态方法
```

7.1.2 设置缓存

```
use fize\cache\Cache;

$config = [
        'host'    => '192.168.56.101',
        'port'    => 6379,
        'timeout' => 10,
        'expire'  => 0,
        'dbindex' => 15
];
new Cache('Redis', $config);

Cache::set('cfz', ' 我想在里面填什么都可以', 100);
$cache1 = Cache::get('cfz');
var_dump($cache1);

Cache::set('cfz2', ' 我想在里面填什么都可以 2');
$cache2 = Cache::get('cfz2');
var_dump($cache2);
```

7.1.3 获取缓存

```
use fize\cache\Cache;

$config = [
        'host'    => '192.168.56.101',
        'port'    => 6379,
        'timeout' => 10,
        'expire'  => 0,
        'dbindex' => 15
];
new Cache('Redis', $config);

Cache::set('cfz', 'hello world!');
$cache1 = Cache::get('cfz');
var_dump($cache1);  //hello world!

Cache::remove('cfz2');
$cache2 = Cache::get('cfz2');
```

(下页继续)

(续上页)

```
var_dump($cache2);  //null
```

7.1.4 判断缓存

```
use fize\cache\Cache;

$config = [
        'host'    => '192.168.56.101',
        'port'    => 6379,
        'timeout' => 10,
        'expire'  => 0,
        'dbindex' => 15
];
new Cache('Redis', $config);

Cache::remove('cfz1');
$has1 = Cache::has('cfz1');
var_dump($has1);  //false

Cache::set('cfz1', 'hello world2!');
$has2 = Cache::has('cfz1');
var_dump($has2);  //true
```

7.1.5 删除缓存

```
use fize\cache\Cache;

$config = [
        'host'    => '192.168.56.101',
        'port'    => 6379,
        'timeout' => 10,
        'expire'  => 0,
        'dbindex' => 15
];
new Cache('Redis', $config);

Cache::set('cfz', ' 我想在里面填什么都可以');
Cache::remove('cfz');
```

(下页继续)

(续上页)

```
$cache1 = Cache::get('cfz');
var_dump($cache1);  //null
```

7.1.6 清空缓存

```
use fize\cache\Cache;

$config = [
        'host'    => '192.168.56.101',
        'port'    => 6379,
        'timeout' => 10,
        'expire'  => 0,
        'dbindex' => 15
];
new Cache('Redis', $config);

Cache::clear();  //cache 被清空
```

7.1.7 创建新实例

```
use fize\cache\Cache;

$cache = Cache::getInstance('File');

// 使用 cache 的实例方法进行操作

$cache->set('key', 'value');
$val = $cache->get('key');
var_dump($val);
```

CHAPTER 8

类库参考

8.1 类库参考

8.1.1 缓存

属性	值
命名空间	fize\cache
类名	AbstractCache
修饰符	abstract
实现接口	fize\cache\CacheInterface, Psr\SimpleCache\CacheInterface

方法

方法名	说明
___construct()	构造函数
get()	获取一个缓存
set()	设置一个缓存
delete()	删除一个缓存
clear()	清除所有缓存
getMultiple()	获取多个缓存
setMultiple()	设置多个缓存
deleteMultiple()	删除多个缓存
has()	判断缓存是否存在

方法

__construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

get()

获取一个缓存

```
public function get (
    string $key,
    mixed $default = null
) : mixed
```

参数

名称	说明
key	键名
default	默认值

set()

设置一个缓存

```
public function set (
    string $key,
    mixed $value,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
key	键名
value	值
ttl	以秒为单位的过期时长

delete()

删除一个缓存

```
public function delete (
    string $key
) : bool
```

参数

名称	说明
key	键名

clear()

清除所有缓存

```
public function clear () : bool
```

getMultiple()

获取多个缓存

```
public function getMultiple (
    iterable $keys,
    mixed $default = null
) : iterable
```

参数

名称	说明
keys	键名数组
default	默认值

setMultiple()

设置多个缓存

```
public function setMultiple (
    iterable $values,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
values	[键名 => 值] 数组
ttl	以秒为单位的过期时长

deleteMultiple()

删除多个缓存

```
public function deleteMultiple (
    iterable $keys
) : bool
```

参数

名称	说明
keys	键名数组

has()

判断缓存是否存在

```
public function has (
    string $key
) : bool
```

参数

名称	说明
key	键名

8.1.2 缓存池

属性	值
命名空间	fize\cache
类名	AbstractPool
修饰符	abstract
实现接口	fize\cache\PoolInterface, Psr\Cache\CacheItemPoolInterface

方法

方法名	说明
__construct()	构造
hasItem()	检查是否有对应的缓存项
getItems()	返回一个可供遍历的缓存项集合
deleteItems()	移除多个缓存项
saveItems()	设置多个缓存项
saveDeferred()	稍后为缓存项做数据持久化
commit()	提交所有的正在队列里等待的请求到数据持久层
getItem()	Returns a Cache Item representing the specified key.
clear()	Deletes all items in the pool.
deleteItem()	Removes the item from the pool.
save()	Persists a cache item immediately.

方法

__construct()

构造

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

hasItem()

检查是否有对应的缓存项

```
public function hasItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

getItems()

返回一个可供遍历的缓存项集合

```
public function getItems (
    array $keys = []
) : \CacheItemInterface[]
```

参数

名称	说明
keys	键名组成的数组

deleteItems()

移除多个缓存项

```
public function deleteItems (
    array $keys
) : bool
```

参数

名称	说明
keys	键名组成的数组

saveItems()

设置多个缓存项

```
public function saveItems (
    \CacheItemInterface[] $items
) : bool
```

参数

名称	说明
items	

saveDeferred()

稍后为缓存项做数据持久化

```
public function saveDeferred (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	

commit()

提交所有的正在队列里等待的请求到数据持久层

```
public function commit () : bool
```

getItem()

Returns a Cache Item representing the specified key.

```
abstract public function getItem (
    string $key
) : \Psr\Cache\CacheItemInterface
```

参数

名称	说明
key	The key for which to return the corresponding Cache Item.

返回值 The corresponding Cache Item.

```
This method must always return a CacheItemInterface object, even in case of
a cache miss. It MUST NOT return null.
```

clear()

Deletes all items in the pool.

```
abstract public function clear () : bool
```

返回值 True if the pool was successfully cleared. False if there was an error.

deleteItem()

Removes the item from the pool.

```
abstract public function deleteItem (
    string $key
) : bool
```

参数

名称	说明
key	The key to delete.

返回值 True if the item was successfully removed. False if there was an error.

save()

Persists a cache item immediately.

```
abstract public function save (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	The cache item to save.

返回值 True if the item was successfully persisted. False if there was an error.

8.1.3 简易缓存

遵循 PSR16 规范，使用静态方法调用

属性	值
命名空间	fize\cache
类名	Cache

方法

方法名	说明
___construct()	常规调用请先初始化
getInstance()	取得实例
get()	获取一个缓存
set()	设置一个缓存
delete()	删除一个缓存
clear()	清空所有缓存
getMultiple()	获取多个缓存
setMultiple()	设置多个缓存
deleteMultiple()	删除多个缓存
has()	判断缓存是否存在

方法

___construct()

常规调用请先初始化

```
public function __construct (
    string $handler,
    array $config = []
)
```

参数

名称	说明
handler	使用的实际接口名称
config	配置项

getInstance()

取得实例

```
public static function getInstance (
    string $handler,
    array $config = []
) : \fize\cache\CacheInterface
```

参数

名称	说明
handler	使用的实际接口名称
config	配置

get()

获取一个缓存

```
public static function get (
    string $key,
    mixed $default = null
) : mixed
```

参数

名称	说明
key	键名
default	默认值

set()

设置一个缓存

```
public static function set (
    string $key,
    mixed $value,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
key	键名
value	值
ttl	以秒为单位的过期时长

delete()

删除一个缓存

```
public static function delete (
    string $key
) : bool
```

参数

名称	说明
key	键名

clear()

清空所有缓存

```
public static function clear () : bool
```

getMultiple()

获取多个缓存

```
public static function getMultiple (
    iterable $keys,
```

(下页继续)

(续上页)

```
    mixed $default = null
) : iterable
```

参数

名称	说明
keys	键名数组
default	默认值

setMultiple()

设置多个缓存

```
public static function setMultiple (
    iterable $values,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
values	[键名 => 值] 数组
ttl	以秒为单位的过期时长

deleteMultiple()

删除多个缓存

```
public static function deleteMultiple (
    iterable $keys
) : bool
```

参数

名称	说明
keys	键名数组

has()

判断缓存是否存在

```
public static function has (
    string $key
) : bool
```

参数

名称	说明
key	键名

8.1.4 缓存异常

属性	值
命名空间	fize\cache
类名	CacheException
父类	RuntimeException
实现接口	Throwable, Psr\Cache\CacheException, Psr\SimpleCache\CacheException

8.1.5 简易缓存接口

属性	值
命名空间	fize\cache
类名	CacheInterface
实现接口	Psr\SimpleCache\CacheInterface

方法

方法名	说明
___construct()	构造函数
get()	Fetches a value from the cache.
set()	Persists data in the cache, uniquely referenced by a key with an optional expiration TTL time.
delete()	Delete an item from the cache by its unique key.
clear()	Wipes clean the entire cache’ s keys.
getMultiple()	Obtains multiple cache items by their unique keys.
setMultiple()	Persists a set of key => value pairs in the cache, with an optional TTL.
deleteMultiple()	Deletes multiple cache items in a single operation.
has()	Determines whether an item is present in the cache.

方法

___construct()

构造函数

```
abstract public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

get()

Fetches a value from the cache.

```
abstract public function get (
    string $key,
    mixed $default = null
) : mixed
```

参数

名称	说明
key	The unique key of this item in the cache.
default	Default value to return if the key does not exist.

返回值 The value of the item from the cache, or $default in case of cache miss.

set()

Persists data in the cache, uniquely referenced by a key with an optional expiration TTL time.

```
abstract public function set (
    string $key,
    mixed $value,
    null|int|\DateInterval $ttl = null
) : bool
```

参数

名称	说明
key	The key of the item to store.
value	The value of the item to store, must be serializable.

|value |The value of the item to store, must be serializable. | +——-+

+ |ttl |Optional. The TTL value of this item. If no value is sent and the driver supports TTL then the library may set a default value for it or let the driver take care of that. | +——-+

+

返回值 True on success and false on failure.

delete()

Delete an item from the cache by its unique key.

```
abstract public function delete (
    string $key
) : bool
```

参数

名称	说明
key	The unique cache key of the item to delete.

返回值 True if the item was successfully removed. False if there was an error.

clear()

Wipes clean the entire cache’ s keys.

```
abstract public function clear () : bool
```

返回值 True on success and false on failure.

getMultiple()

Obtains multiple cache items by their unique keys.

```
abstract public function getMultiple (
    iterable $keys,
    mixed $default = null
) : iterable
```

参数

名称	说明
keys	A list of keys that can obtained in a single operation.
default	Default value to return for keys that do not exist.

返回值 A list of key => value pairs. Cache keys that do not exist or are stale will have $default as value.

setMultiple()

Persists a set of key => value pairs in the cache, with an optional TTL.

```
abstract public function setMultiple (
    iterable $values,
    null|int|\DateInterval $ttl = null
) : bool
```

参数

名称	说明
values	A list of key => value pairs for a multiple-set operation.

|values |A list of key => value pairs for a multiple-set operation. | +——-+

+ |ttl |Optional. The TTL value of this item. If no value is sent and the driver supports TTL then the library may set a default value for it or let the driver take care of that. | +——-+

+

返回值 True on success and false on failure.

deleteMultiple()

Deletes multiple cache items in a single operation.

```
abstract public function deleteMultiple (
    iterable $keys
) : bool
```

参数

名称	说明
keys	A list of string-based keys to be deleted.

返回值 True if the items were successfully removed. False if there was an error.

has()

Determines whether an item is present in the cache.

```
abstract public function has (
    string $key
) : bool
```

参数

名称	说明
key	The cache item key.

```
NOTE: It is recommended that has() is only to be used for cache warming type purposes
and not to be used within your live applications operations for get/set, as this method
is subject to a race condition where your has() will return true and immediately after,
another script can remove it making the state of your app out of date.
```

8.1.6 参数异常

属性	值
命名空间	fize\cache
类名	InvalidArgumentException
父类	InvalidArgumentException
实现接口	Throwable, Psr\Cache\InvalidArgumentException, Psr\Cache\CacheException, Psr\SimpleCache\InvalidArgumentException, Psr\SimpleCache\CacheException

8.1.7 缓存项

```
禁止擅自初始化「Item」对象
该类实例只能使用「CacheItemPoolInterface」对象的 getItem() 方法来获取
```

属性	值
命名空间	fize\cache
类名	Item
实现接口	fize\cache\ItemInterface, Psr\Cache\CacheItemInterface

方法

方法名	说明
___construct()	构造
getKey()	获取键名
isHit()	是否命中
get()	获取值
set()	设置值
expiresAt()	设置缓存项的准确过期时间点
expiresAfter()	设置缓存项的过期时间
setHit()	设置是否命中
getExpires()	获取缓存项的过期时间戳
checkHit()	根据设置判断缓存是否有效

方法

___construct()

构造

```
public function __construct (
    string $key
)
```

参数

名称	说明
key	键名

getKey()

获取键名

```
public function getKey () : string
```

isHit()

是否命中

```
public function isHit () : bool
```

get()

获取值

```
public function get () : mixed
```

set()

设置值

```
public function set (
    mixed $value
) : $this
```

参数

名称	说明
value	值

expiresAt()

设置缓存项的准确过期时间点

```
public function expiresAt (
    \DateTimeInterface|null $expiration
) : $this
```

参数

名称	说明
expiration	过期时间点

```
参数 `$expiration`:
为 null 表示使用默认设置
```

expiresAfter()

设置缓存项的过期时间

```
public function expiresAfter (
    \DateInterval|int|null $time
) : $this
```

参数

名称	说明
time	以秒为单位的过期时长

setHit()

设置是否命中

```
public function setHit (
    bool $is_hit
) : $this
```

参数

名称	说明
is_hit	是否命中

外部不应直接调用该方法

getExpires()

获取缓存项的过期时间戳

```
public function getExpires () : int|null
```

返回值 返回 null 表示永不过期

checkHit()

根据设置判断缓存是否有效

```
public function checkHit () : bool
```

8.1.8 缓存项接口

属性	值
命名空间	fize\cache
类名	ItemInterface
实现接口	Psr\Cache\CacheItemInterface

方法

方法名	说明
___construct()	构造
setHit()	设置是否命中
getExpires()	获取缓存项的过期时间戳
checkHit()	根据设置判断缓存是否有效
getKey()	Returns the key for the current cache item.
get()	Retrieves the value of the item from the cache associated with this object’ s key.
isHit()	Confirms if the cache item lookup resulted in a cache hit.
set()	Sets the value represented by this cache item.
expiresAt()	Sets the expiration time for this cache item.
expiresAfter()	Sets the expiration time for this cache item.

方法

___construct()

构造

```
abstract public function __construct (
    string $key
)
```

参数

名称	说明
key	键名

```
禁止擅自初始化「CacheItemInterface」对象
该类实例只能使用「CacheItemPoolInterface」对象的 getItem() 方法来获取
```

setHit()

设置是否命中

```
abstract public function setHit (
    bool $is_hit
) : $this
```

参数

名称	说明
is_hit	是否命中

外部不应直接调用该方法

getExpires()

获取缓存项的过期时间戳

```
abstract public function getExpires () : int|null
```

返回值 返回 null 表示永不过期

checkHit()

根据设置判断缓存是否有效

```
abstract public function checkHit () : bool
```

getKey()

Returns the key for the current cache item.

```
abstract public function getKey () : string
```

返回值 The key string for this cache item.

```
The key is loaded by the Implementing Library, but should be available to
the higher level callers when needed.
```

get()

Retrieves the value of the item from the cache associated with this object's key.

```
abstract public function get () : mixed
```

返回值 The value corresponding to this cache item's key, or null if not found.

```
The value returned must be identical to the value originally stored by set().

If isHit() returns false, this method MUST return null. Note that null
is a legitimate cached value, so the isHit() method SHOULD be used to
differentiate between "null value was found" and "no value was found."
```

isHit()

Confirms if the cache item lookup resulted in a cache hit.

```
abstract public function isHit () : bool
```

返回值 True if the request resulted in a cache hit. False otherwise.

```
Note: This method MUST NOT have a race condition between calling isHit()
and calling get().
```

set()

Sets the value represented by this cache item.

```
abstract public function set (
    mixed $value
) : static
```

参数

名称	说明
value	The serializable value to be stored.

返回值 The invoked object.

```
The $value argument may be any item that can be serialized by PHP,
although the method of serialization is left up to the Implementing
Library.
```

expiresAt()

Sets the expiration time for this cache item.

```
abstract public function expiresAt (
    \DateTimeInterface|null $expiration
) : static
```

参数 If null is passed explicitly, a default value MAY be used. If none is set, the value should be stored permanently or for as long as the implementation allows. | +———–+ -+

返回值 The called object.

expiresAfter()

Sets the expiration time for this cache item.

```
abstract public function expiresAfter (
    int|\DateInterval|null $time
) : static
```

参数 expired. An integer parameter is understood to be the time in seconds until expiration. If null is passed explicitly, a default value MAY be used. If none is set, the value should be stored permanently or for as long as the implementation allows. | +——-+ –+

返回值 The called object.

8.1.9 缓存池

```
遵循 PSR6 规范，使用静态方法调用
```

属性	值
命名空间	fize\cache
类名	Pool

方法

方法名	说明
___construct()	常规调用请先初始化
getInstance()	取得实例
getItem()	获取缓存项
getItems()	返回一个可供遍历的缓存项集合
hasItem()	检查是否有对应的缓存项
clear()	清空缓存池
deleteItem()	从缓存池里移除缓存项
deleteItems()	移除多个缓存项
save()	立刻为对象做数据持久化
saveDeferred()	稍后为缓存项做数据持久化
commit()	提交所有的正在队列里等待的请求到数据持久层
saveItems()	设置多个缓存项

方法

___construct()

常规调用请先初始化

```
public function __construct (
    string $handler,
    array $config = []
)
```

参数

名称	说明
handler	使用的实际接口名称
config	配置项

getInstance()

取得实例

```
public static function getInstance (
    string $handler,
    array $config = []
) : \fize\cache\PoolInterface
```

参数

名称	说明
handler	使用的实际接口名称
config	配置

getItem()

获取缓存项

```
public static function getItem (
    string $key
) : \Psr\Cache\CacheItemInterface
```

参数

名称	说明
key	键名

getItems()

返回一个可供遍历的缓存项集合

```
public static function getItems (
    array $keys = []
) : \CacheItemInterface[]
```

参数

名称	说明
keys	键名组成的数组

hasItem()

检查是否有对应的缓存项

```
public static function hasItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

clear()

清空缓存池

```
public static function clear () : bool
```

deleteItem()

从缓存池里移除缓存项

```
public static function deleteItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

deleteItems()

移除多个缓存项

```
public static function deleteItems (
    array $keys
) : bool
```

参数

名称	说明
keys	键名组成的数组

save()

立刻为对象做数据持久化

```
public static function save (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	缓存对象

saveDeferred()

稍后为缓存项做数据持久化

```
public static function saveDeferred (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	

commit()

提交所有的正在队列里等待的请求到数据持久层

```
public static function commit () : bool
```

saveItems()

设置多个缓存项

```
public static function saveItems (
    \CacheItemInterface[] $items
) : bool
```

参数

名称	说明
items	

8.1.10 缓存池接口

属性	值
命名空间	fize\cache
类名	PoolInterface
实现接口	Psr\Cache\CacheItemPoolInterface

方法

方法名	说明
___construct()	构造
saveItems()	设置多个缓存项
getItem()	Returns a Cache Item representing the specified key.
getItems()	Returns a traversable set of cache items.
hasItem()	Confirms if the cache contains specified cache item.
clear()	Deletes all items in the pool.
deleteItem()	Removes the item from the pool.
deleteItems()	Removes multiple items from the pool.
save()	Persists a cache item immediately.
saveDeferred()	Sets a cache item to be persisted later.
commit()	Persists any deferred cache items.

方法

___construct()

构造

```
abstract public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

saveItems()

设置多个缓存项

```
abstract public function saveItems (
    \CacheItemInterface[] $items
) : bool
```

参数

名称	说明
items	

getItem()

Returns a Cache Item representing the specified key.

```
abstract public function getItem (
    string $key
) : \Psr\Cache\CacheItemInterface
```

参数

名称	说明
key	The key for which to return the corresponding Cache Item.

返回值 The corresponding Cache Item.

```
This method must always return a CacheItemInterface object, even in case of
a cache miss. It MUST NOT return null.
```

getItems()

Returns a traversable set of cache items.

```
abstract public function getItems (
    string[] $keys = []
) : array|\Traversable
```

参数

名称	说明
keys	An indexed array of keys of items to retrieve.

返回值 A traversable collection of Cache Items keyed by the cache keys of each item. A Cache item will be returned for each key, even if that key is not found. However, if no keys are specified then an empty traversable MUST be returned instead.

hasItem()

Confirms if the cache contains specified cache item.

```
abstract public function hasItem (
    string $key
) : bool
```

参数

名称	说明
key	The key for which to check existence.

返回值 True if item exists in the cache, false otherwise.

```
Note: This method MAY avoid retrieving the cached value for performance reasons.
This could result in a race condition with CacheItemInterface::get(). To avoid
such situation use CacheItemInterface::isHit() instead.
```

clear()

Deletes all items in the pool.

```
abstract public function clear () : bool
```

返回值 True if the pool was successfully cleared. False if there was an error.

deleteItem()

Removes the item from the pool.

```
abstract public function deleteItem (
    string $key
) : bool
```

参数

名称	说明
key	The key to delete.

返回值 True if the item was successfully removed. False if there was an error.

deleteItems()

Removes multiple items from the pool.

```
abstract public function deleteItems (
    string[] $keys
) : bool
```

参数

名称	说明
keys	An array of keys that should be removed from the pool.

返回值 True if the items were successfully removed. False if there was an error.

save()

Persists a cache item immediately.

```
abstract public function save (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	The cache item to save.

返回值 True if the item was successfully persisted. False if there was an error.

saveDeferred()

Sets a cache item to be persisted later.

```
abstract public function saveDeferred (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	The cache item to save.

返回值 False if the item could not be queued or if a commit was attempted and failed. True otherwise.

commit()

Persists any deferred cache items.

```
abstract public function commit () : bool
```

返回值 True if all not-yet-saved items were successfully saved or there were none. False otherwise.

8.1.11 处理器

database

简易缓存

属性	值
命名空间	fize\cache\handler\database
类名	Cache
父类	fize\cache\AbstractCache
实现接口	Psr\SimpleCache\CacheInterface, fize\cache\CacheInterface

方法

方法名	说明
___construct()	构造函数
get()	获取一个缓存
set()	设置一个缓存
delete()	删除一个缓存
clear()	清除所有缓存
getMultiple()	获取多个缓存
setMultiple()	设置多个缓存
deleteMultiple()	删除多个缓存
has()	判断缓存是否存在

方法

___construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

get()

获取一个缓存

```
public function get (
    string $key,
    mixed $default = null
) : mixed
```

参数

名称	说明
key	键名
default	默认值

set()

设置一个缓存

```
public function set (
    string $key,
    mixed $value,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
key	键名
value	值
ttl	以秒为单位的过期时长

delete()

删除一个缓存

```
public function delete (
    string $key
) : bool
```

参数

名称	说明
key	键名

clear()

清除所有缓存

```
public function clear () : bool
```

getMultiple()

获取多个缓存

```
public function getMultiple (
    iterable $keys,
    mixed $default = null
) : iterable
```

参数

名称	说明
keys	键名数组
default	默认值

setMultiple()

设置多个缓存

```
public function setMultiple (
    iterable $values,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
values	[键名 => 值] 数组
ttl	以秒为单位的过期时长

deleteMultiple()

删除多个缓存

```
public function deleteMultiple (
    iterable $keys
) : bool
```

参数

名称	说明
keys	键名数组

has()

判断缓存是否存在

```
public function has (
    string $key
) : bool
```

参数

名称	说明
key	键名

缓存池

属性	值
命名空间	fize\cache\handler\database
类名	Pool
父类	fize\cache\AbstractPool
实现接口	Psr\Cache\CacheItemPoolInterface, fize\cache\PoolInterface

方法

方法名	说明
___construct()	构造函数
getItem()	获取缓存项
clear()	清空缓存池
deleteItem()	从缓存池里移除缓存项
save()	立刻为对象做数据持久化
initMysql()	初始化，如果尚未建立 cache 表，可以运行该方法来建立表
hasItem()	检查是否有对应的缓存项
getItems()	返回一个可供遍历的缓存项集合
deleteItems()	移除多个缓存项
saveItems()	设置多个缓存项
saveDeferred()	稍后为缓存项做数据持久化
commit()	提交所有的正在队列里等待的请求到数据持久层

方法

___construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

getItem()

获取缓存项

```
public function getItem (
    string $key
) : \Psr\Cache\CacheItemInterface
```

参数

名称	说明
key	键名

clear()

清空缓存池

```
public function clear () : bool
```

deleteItem()

从缓存池里移除缓存项

```
public function deleteItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

save()

立刻为对象做数据持久化

```
public function save (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	缓存对象

initMysql()

初始化，如果尚未建立 cache 表，可以运行该方法来建立表

```
public static function initMysql (
    array $config
)
```

参数

名称	说明
config	

```
适用于 mysql
```

hasItem()

检查是否有对应的缓存项

```
public function hasItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

getItems()

返回一个可供遍历的缓存项集合

```
public function getItems (
    array $keys = []
) : \CacheItemInterface[]
```

参数

名称	说明
keys	键名组成的数组

deleteItems()

移除多个缓存项

```
public function deleteItems (
    array $keys
) : bool
```

参数

名称	说明
keys	键名组成的数组

saveItems()

设置多个缓存项

```
public function saveItems (
    \CacheItemInterface[] $items
) : bool
```

参数

名称	说明
items	

saveDeferred()

稍后为缓存项做数据持久化

```
public function saveDeferred (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	

commit()

提交所有的正在队列里等待的请求到数据持久层

```
public function commit () : bool
```

file

简易缓存

属性	值
命名空间	fize\cache\handler\file
类名	Cache
父类	fize\cache\AbstractCache
实现接口	Psr\SimpleCache\CacheInterface, fize\cache\CacheInterface

方法

方法名	说明
___construct()	构造函数
get()	获取一个缓存
set()	设置一个缓存
delete()	删除一个缓存
clear()	清除所有缓存
getMultiple()	获取多个缓存
setMultiple()	设置多个缓存
deleteMultiple()	删除多个缓存
has()	判断缓存是否存在

方法

___construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

get()

获取一个缓存

```
public function get (
    string $key,
    mixed $default = null
) : mixed
```

参数

名称	说明
key	键名
default	默认值

set()

设置一个缓存

```
public function set (
    string $key,
    mixed $value,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
key	键名
value	值
ttl	以秒为单位的过期时长

delete()

删除一个缓存

```
public function delete (
    string $key
) : bool
```

参数

名称	说明
key	键名

clear()

清除所有缓存

```
public function clear () : bool
```

getMultiple()

获取多个缓存

```php
public function getMultiple (
    iterable $keys,
    mixed $default = null
) : iterable
```

参数

名称	说明
keys	键名数组
default	默认值

setMultiple()

设置多个缓存

```php
public function setMultiple (
    iterable $values,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
values	[键名 => 值] 数组
ttl	以秒为单位的过期时长

deleteMultiple()

删除多个缓存

```php
public function deleteMultiple (
    iterable $keys
) : bool
```

参数

名称	说明
keys	键名数组

has()

判断缓存是否存在

```
public function has (
    string $key
) : bool
```

参数

名称	说明
key	键名

缓存池

属性	值
命名空间	fize\cache\handler\file
类名	Pool
父类	fize\cache\AbstractPool
实现接口	Psr\Cache\CacheItemPoolInterface, fize\cache\PoolInterface

方法

方法名	说明
__construct()	构造函数
getItem()	获取缓存项
clear()	清空缓存池
deleteItem()	从缓存池里移除缓存项
save()	立刻为对象做数据持久化
hasItem()	检查是否有对应的缓存项
getItems()	返回一个可供遍历的缓存项集合
deleteItems()	移除多个缓存项
saveItems()	设置多个缓存项
saveDeferred()	稍后为缓存项做数据持久化
commit()	提交所有的正在队列里等待的请求到数据持久层

方法

___construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

getItem()

获取缓存项

```
public function getItem (
    string $key
) : \Psr\Cache\CacheItemInterface
```

参数

名称	说明
key	键名

clear()

清空缓存池

```
public function clear () : bool
```

deleteItem()

从缓存池里移除缓存项

```
public function deleteItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

save()

立刻为对象做数据持久化

```
public function save (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	缓存对象

hasItem()

检查是否有对应的缓存项

```
public function hasItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

getItems()

返回一个可供遍历的缓存项集合

```
public function getItems (
    array $keys = []
) : \CacheItemInterface[]
```

参数

名称	说明
keys	键名组成的数组

deleteItems()

移除多个缓存项

```
public function deleteItems (
    array $keys
) : bool
```

参数

名称	说明
keys	键名组成的数组

saveItems()

设置多个缓存项

```
public function saveItems (
    \CacheItemInterface[] $items
) : bool
```

参数

名称	说明
items	

saveDeferred()

稍后为缓存项做数据持久化

```
public function saveDeferred (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	

commit()

提交所有的正在队列里等待的请求到数据持久层

```
public function commit () : bool
```

memcached

简易缓存

属性	值
命名空间	fize\cache\handler\memcached
类名	Cache
父类	fize\cache\AbstractCache
实现接口	Psr\SimpleCache\CacheInterface, fize\cache\CacheInterface

方法

方法名	说明
___construct()	构造函数
get()	获取一个缓存
set()	设置一个缓存
delete()	删除一个缓存
clear()	清除所有缓存
getMultiple()	获取多个缓存
setMultiple()	设置多个缓存
deleteMultiple()	删除多个缓存
has()	判断缓存是否存在

方法

___construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

get()

获取一个缓存

```
public function get (
    string $key,
    mixed $default = null
) : mixed
```

参数

名称	说明
key	键名
default	默认值

set()

设置一个缓存

```
public function set (
    string $key,
    mixed $value,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
key	键名
value	值
ttl	以秒为单位的过期时长

delete()

删除一个缓存

```
public function delete (
    string $key
) : bool
```

参数

名称	说明
key	键名

clear()

清除所有缓存

```
public function clear () : bool
```

getMultiple()

获取多个缓存

```
public function getMultiple (
    iterable $keys,
    mixed $default = null
) : iterable
```

参数

名称	说明
keys	键名数组
default	默认值

setMultiple()

设置多个缓存

```
public function setMultiple (
    iterable $values,
```

(下页继续)

(续上页)

```
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
values	[键名 => 值] 数组
ttl	以秒为单位的过期时长

deleteMultiple()

删除多个缓存

```
public function deleteMultiple (
    iterable $keys
) : bool
```

参数

名称	说明
keys	键名数组

has()

判断缓存是否存在

```
public function has (
    string $key
) : bool
```

参数

名称	说明
key	键名

缓存池

属性	值
命名空间	fize\cache\handler\memcached
类名	Pool
父类	fize\cache\AbstractPool
实现接口	Psr\Cache\CacheItemPoolInterface, fize\cache\PoolInterface

方法

方法名	说明
___construct()	构造函数
___destruct()	析构时关闭 Memcached 连接
getItem()	获取缓存项
clear()	清空缓存池
deleteItem()	从缓存池里移除缓存项
save()	立刻为对象做数据持久化
hasItem()	检查是否有对应的缓存项
getItems()	返回一个可供遍历的缓存项集合
deleteItems()	移除多个缓存项
saveItems()	设置多个缓存项
saveDeferred()	稍后为缓存项做数据持久化
commit()	提交所有的正在队列里等待的请求到数据持久层

方法

___construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

___destruct()

析构时关闭 Memcached 连接

```
public function __destruct ()
```

getItem()

获取缓存项

```
public function getItem (
    string $key
) : \Psr\Cache\CacheItemInterface
```

参数

名称	说明
key	键名

clear()

清空缓存池

```
public function clear () : bool
```

deleteItem()

从缓存池里移除缓存项

```
public function deleteItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

save()

立刻为对象做数据持久化

```
public function save (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	缓存对象

hasItem()

检查是否有对应的缓存项

```
public function hasItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

getItems()

返回一个可供遍历的缓存项集合

```
public function getItems (
    array $keys = []
) : \CacheItemInterface[]
```

参数

名称	说明
keys	键名组成的数组

deleteItems()

移除多个缓存项

```
public function deleteItems (
    array $keys
) : bool
```

参数

名称	说明
keys	键名组成的数组

saveItems()

设置多个缓存项

```
public function saveItems (
    \CacheItemInterface[] $items
) : bool
```

参数

名称	说明
items	

saveDeferred()

稍后为缓存项做数据持久化

```
public function saveDeferred (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	

commit()

提交所有的正在队列里等待的请求到数据持久层

```
public function commit () : bool
```

redis

简易缓存

属性	值
命名空间	fize\cache\handler\redis
类名	Cache
父类	fize\cache\AbstractCache
实现接口	Psr\SimpleCache\CacheInterface, fize\cache\CacheInterface

方法

方法名	说明
__construct()	构造函数
get()	获取一个缓存
set()	设置一个缓存
delete()	删除一个缓存
clear()	清除所有缓存
getMultiple()	获取多个缓存
setMultiple()	设置多个缓存
deleteMultiple()	删除多个缓存
has()	判断缓存是否存在

方法

__construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	配置

get()

获取一个缓存

```
public function get (
    string $key,
    mixed $default = null
) : mixed
```

参数

名称	说明
key	键名
default	默认值

set()

设置一个缓存

```
public function set (
    string $key,
    mixed $value,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
key	键名
value	值
ttl	以秒为单位的过期时长

delete()

删除一个缓存

```
public function delete (
    string $key
) : bool
```

参数

名称	说明
key	键名

clear()

清除所有缓存

```
public function clear () : bool
```

getMultiple()

获取多个缓存

```
public function getMultiple (
    iterable $keys,
    mixed $default = null
) : iterable
```

参数

名称	说明
keys	键名数组
default	默认值

setMultiple()

设置多个缓存

```
public function setMultiple (
    iterable $values,
    \DateInterval|int|null $ttl = null
) : bool
```

参数

名称	说明
values	[键名 => 值] 数组
ttl	以秒为单位的过期时长

deleteMultiple()

删除多个缓存

```
public function deleteMultiple (
    iterable $keys
) : bool
```

参数

名称	说明
keys	键名数组

has()

判断缓存是否存在

```
public function has (
    string $key
) : bool
```

参数

名称	说明
key	键名

缓存池

属性	值
命名空间	fize\cache\handler\redis
类名	Pool
父类	fize\cache\AbstractPool
实现接口	Psr\Cache\CacheItemPoolInterface, fize\cache\PoolInterface

方法

方法名	说明
___construct()	构造函数
getItem()	获取缓存项
clear()	清空缓存池
deleteItem()	从缓存池里移除缓存项
save()	立刻为对象做数据持久化
hasItem()	检查是否有对应的缓存项
getItems()	返回一个可供遍历的缓存项集合
deleteItems()	移除多个缓存项
saveItems()	设置多个缓存项
saveDeferred()	稍后为缓存项做数据持久化
commit()	提交所有的正在队列里等待的请求到数据持久层

方法

___construct()

构造函数

```
public function __construct (
    array $config = []
)
```

参数

名称	说明
config	初始化默认选项

getItem()

获取缓存项

```
public function getItem (
    string $key
) : \Psr\Cache\CacheItemInterface
```

参数

名称	说明
key	键名

clear()

清空缓存池

```
public function clear () : bool
```

deleteItem()

从缓存池里移除缓存项

```
public function deleteItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

save()

立刻为对象做数据持久化

```
public function save (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	缓存对象

hasItem()

检查是否有对应的缓存项

```
public function hasItem (
    string $key
) : bool
```

参数

名称	说明
key	键名

getItems()

返回一个可供遍历的缓存项集合

```
public function getItems (
    array $keys = []
) : \CacheItemInterface[]
```

参数

名称	说明
keys	键名组成的数组

deleteItems()

移除多个缓存项

```
public function deleteItems (
    array $keys
) : bool
```

参数

名称	说明
keys	键名组成的数组

saveItems()

设置多个缓存项

```
public function saveItems (
    \CacheItemInterface[] $items
) : bool
```

参数

名称	说明
items	

saveDeferred()

稍后为缓存项做数据持久化

```
public function saveDeferred (
    \Psr\Cache\CacheItemInterface $item
) : bool
```

参数

名称	说明
item	

commit()

提交所有的正在队列里等待的请求到数据持久层

```
public function commit () : bool
```