Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

cosenary / Simple-PHP-Cache Watch 24 Star 182 Fork 58

Code Issues 6 Pull requests 8 Pulse Graphs

Branch: master ▾ Simple-PHP-Cache / cache.class.php Find file Copy path

cosenary Version 1.6 ce840c6 Jan 4, 2014

3 contributors

314 lines (288 sloc) 6.81 KB Raw Blame History

```php
<?php

/**
 * Simple Cache class
 * API Documentation: https://github.com/cosenary/Simple-PHP-Cache
 *
 * @author Christian Metz
 * @since 22.12.2011
 * @copyright Christian Metz - MetzWeb Networks
 * @version 1.6
 * @license BSD http://www.opensource.org/licenses/bsd-license.php
 */

class Cache {

  /**
   * The path to the cache file folder
   *
   * @var string
   */
  private $_cachepath = 'cache/';

  /**
   * The name of the default cache file
   *
   * @var string
   */
  private $_cachename = 'default';

  /**
   * The cache file extension
   *
   * @var string
   */
  private $_extension = '.cache';

  /**
   * Default constructor
   *
   * @param string|array [optional] $config
   * @return void
   */
  public function __construct($config = null) {
    if (true === isset($config)) {
      if (is_string($config)) {
        $this->setCache($config);
      } else if (is_array($config)) {
        $this->setCache($config['name']);
        $this->setCachePath($config['path']);
        $this->setExtension($config['extension']);
      }
    }
  }

```

```php
  /**
   * Check whether data accociated with a key
   *
   * @param string $key
   * @return boolean
   */
  public function isCached($key) {
    if (false != $this->_loadCache()) {
      $cachedData = $this->_loadCache();
      return isset($cachedData[$key]['data']);
    }
  }

  /**
   * Store data in the cache
   *
   * @param string $key
   * @param mixed $data
   * @param integer [optional] $expiration
   * @return object
   */
  public function store($key, $data, $expiration = 0) {
    $storeData = array(
      'time'   => time(),
      'expire' => $expiration,
      'data'   => serialize($data)
    );
    $dataArray = $this->_loadCache();
    if (true === is_array($dataArray)) {
      $dataArray[$key] = $storeData;
    } else {
      $dataArray = array($key => $storeData);
    }
    $cacheData = json_encode($dataArray);
    file_put_contents($this->getCacheDir(), $cacheData);
    return $this;
  }

  /**
   * Retrieve cached data by its key
   *
   * @param string $key
   * @param boolean [optional] $timestamp
   * @return string
   */
  public function retrieve($key, $timestamp = false) {
    $cachedData = $this->_loadCache();
    (false === $timestamp) ? $type = 'data' : $type = 'time';
    if (!isset($cachedData[$key][$type])) return null;
    return unserialize($cachedData[$key][$type]);
  }

  /**
   * Retrieve all cached data
   *
   * @param boolean [optional] $meta
   * @return array
   */
  public function retrieveAll($meta = false) {
    if ($meta === false) {
      $results = array();
      $cachedData = $this->_loadCache();
      if ($cachedData) {
        foreach ($cachedData as $k => $v) {
          $results[$k] = unserialize($v['data']);
        }
      }
      return $results;
    } else {
      return $this->_loadCache();
    }
  }

  /**
```

```
129     * Erase cached entry by its key
130     *
131     * @param string $key
132     * @return object
133     */
134    public function erase($key) {
135      $cacheData = $this->_loadCache();
136      if (true === is_array($cacheData)) {
137        if (true === isset($cacheData[$key])) {
138          unset($cacheData[$key]);
139          $cacheData = json_encode($cacheData);
140          file_put_contents($this->getCacheDir(), $cacheData);
141        } else {
142          throw new Exception("Error: erase() - Key '{$key}' not found.");
143        }
144      }
145      return $this;
146    }
147
148    /**
149     * Erase all expired entries
150     *
151     * @return integer
152     */
153    public function eraseExpired() {
154      $cacheData = $this->_loadCache();
155      if (true === is_array($cacheData)) {
156        $counter = 0;
157        foreach ($cacheData as $key => $entry) {
158          if (true === $this->_checkExpired($entry['time'], $entry['expire'])) {
159            unset($cacheData[$key]);
160            $counter++;
161          }
162        }
163        if ($counter > 0) {
164          $cacheData = json_encode($cacheData);
165          file_put_contents($this->getCacheDir(), $cacheData);
166        }
167        return $counter;
168      }
169    }
170
171    /**
172     * Erase all cached entries
173     *
174     * @return object
175     */
176    public function eraseAll() {
177      $cacheDir = $this->getCacheDir();
178      if (true === file_exists($cacheDir)) {
179        $cacheFile = fopen($cacheDir, 'w');
180        fclose($cacheFile);
181      }
182      return $this;
183    }
184
185    /**
186     * Load appointed cache
187     *
188     * @return mixed
189     */
190    private function _loadCache() {
191      if (true === file_exists($this->getCacheDir())) {
192        $file = file_get_contents($this->getCacheDir());
193        return json_decode($file, true);
194      } else {
195        return false;
196      }
197    }
198
199    /**
200     * Get the cache directory path
201     *
202     * @return string
```

```
203     */
204    public function getCacheDir() {
205      if (true === $this->_checkCacheDir()) {
206        $filename = $this->getCache();
207        $filename = preg_replace('/[^0-9a-z\.\_\-]/i', '', strtolower($filename));
208        return $this->getCachePath() . $this->_getHash($filename) . $this->getExtension();
209      }
210    }
211
212    /**
213     * Get the filename hash
214     *
215     * @return string
216     */
217    private function _getHash($filename) {
218      return sha1($filename);
219    }
220
221    /**
222     * Check whether a timestamp is still in the duration
223     *
224     * @param integer $timestamp
225     * @param integer $expiration
226     * @return boolean
227     */
228    private function _checkExpired($timestamp, $expiration) {
229      $result = false;
230      if ($expiration !== 0) {
231        $timeDiff = time() - $timestamp;
232        ($timeDiff > $expiration) ? $result = true : $result = false;
233      }
234      return $result;
235    }
236
237    /**
238     * Check if a writable cache directory exists and if not create a new one
239     *
240     * @return boolean
241     */
242    private function _checkCacheDir() {
243      if (!is_dir($this->getCachePath()) && !mkdir($this->getCachePath(), 0775, true)) {
244        throw new Exception('Unable to create cache directory ' . $this->getCachePath());
245      } elseif (!is_readable($this->getCachePath()) || !is_writable($this->getCachePath())) {
246        if (!chmod($this->getCachePath(), 0775)) {
247          throw new Exception($this->getCachePath() . ' must be readable and writeable');
248        }
249      }
250      return true;
251    }
252
253    /**
254     * Cache path Setter
255     *
256     * @param string $path
257     * @return object
258     */
259    public function setCachePath($path) {
260      $this->_cachepath = $path;
261      return $this;
262    }
263
264    /**
265     * Cache path Getter
266     *
267     * @return string
268     */
269    public function getCachePath() {
270      return $this->_cachepath;
271    }
272
273    /**
274     * Cache name Setter
275     *
276     * @param string $name
```

```
277     * @return object
278     */
279    public function setCache($name) {
280      $this->_cachename = $name;
281      return $this;
282    }
283
284    /**
285     * Cache name Getter
286     *
287     * @return void
288     */
289    public function getCache() {
290      return $this->_cachename;
291    }
292
293    /**
294     * Cache file extension Setter
295     *
296     * @param string $ext
297     * @return object
298     */
299    public function setExtension($ext) {
300      $this->_extension = $ext;
301      return $this;
302    }
303
304    /**
305     * Cache file extension Getter
306     *
307     * @return string
308     */
309    public function getExtension() {
310      return $this->_extension;
311    }
312
313  }
```

© 2016 GitHub, Inc. Terms Privacy Security Contact Help Status API Training Shop Blog About