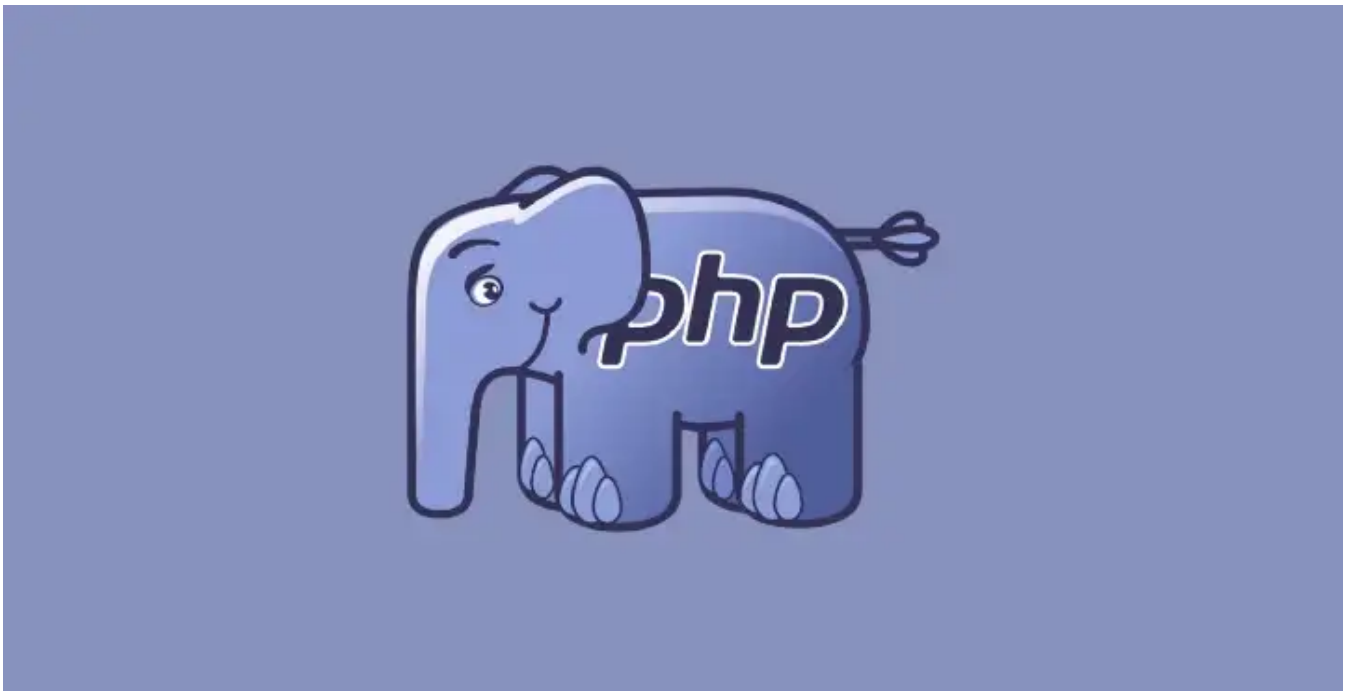
Atakan Demircioğlu Follow
Nov 15, 2022 · 2 min read · List

Save



php-magic-methods

PHP Magic Methods Explained

Here are my notes about PHP Magic methods.

What are PHP Magic Methods?

- special methods that are called automatically when certain conditions are met
- start with a double underscore (__)
- predefined
- neither can be created nor removed
- automatically called

- All magic methods, with the exception of __construct(), __destruct(), and __clone(), *must* be declared as `public`, otherwise an **`E_WARNING`** is emitted.

__construct Magic Method

- automatically every time the object of a particular class is created
- useful to initialization what the object may need before going further

__destruct Magic Method

- called when the object is destroyed

__call Magic Method

- triggered when we call a method that doesn't define yet

__callStatic Magic Method

- similar to the __call() method but this is for static context.

__get Magic Method

- called when an inaccessible variable or non-existing variable is used

__set Magic Method

- called when an inaccessible variable or non-existing variable is written

__isset Magic Method

- triggered by calling isset() or empty() on inaccessible (protected or private) or non-existing properties.

__unset Magic Method

- invoked when unset() is used on inaccessible (protected or private) or non-existing properties.

__toString Magic Method

- called when we need to convert the object into a string

__invoke Magic Method

- called when the object is being called as a function

__clone Magic Method

- triggered where cloning objects is done by using the clone keyword

__debugInfo Magic Method

- triggered when the var_dump function is called

__sleep Magic Method

- serialize() function triggers
- should return an array holding all the object's properties that should be included in the serialized version of that object.
- For **resource types,** PHP fails at serializing it

__wakeup Magic Method

- need to initialize the value whenever we get a new fresh instance from the unserialize function, and that's what we're doing inside the __wakeup method.

__serialize() and __unserialize()

- serialize() checks if the class has a function with the __serialize() magic method. If yes, the function is executed first before any serialization operation.
- If there are both __serialize and _sleep in the same object, just serialize will be called.

Note: For a better understanding of what is happening in __sleep and __wakeup magic methods, you can read this article.

Real Life Example

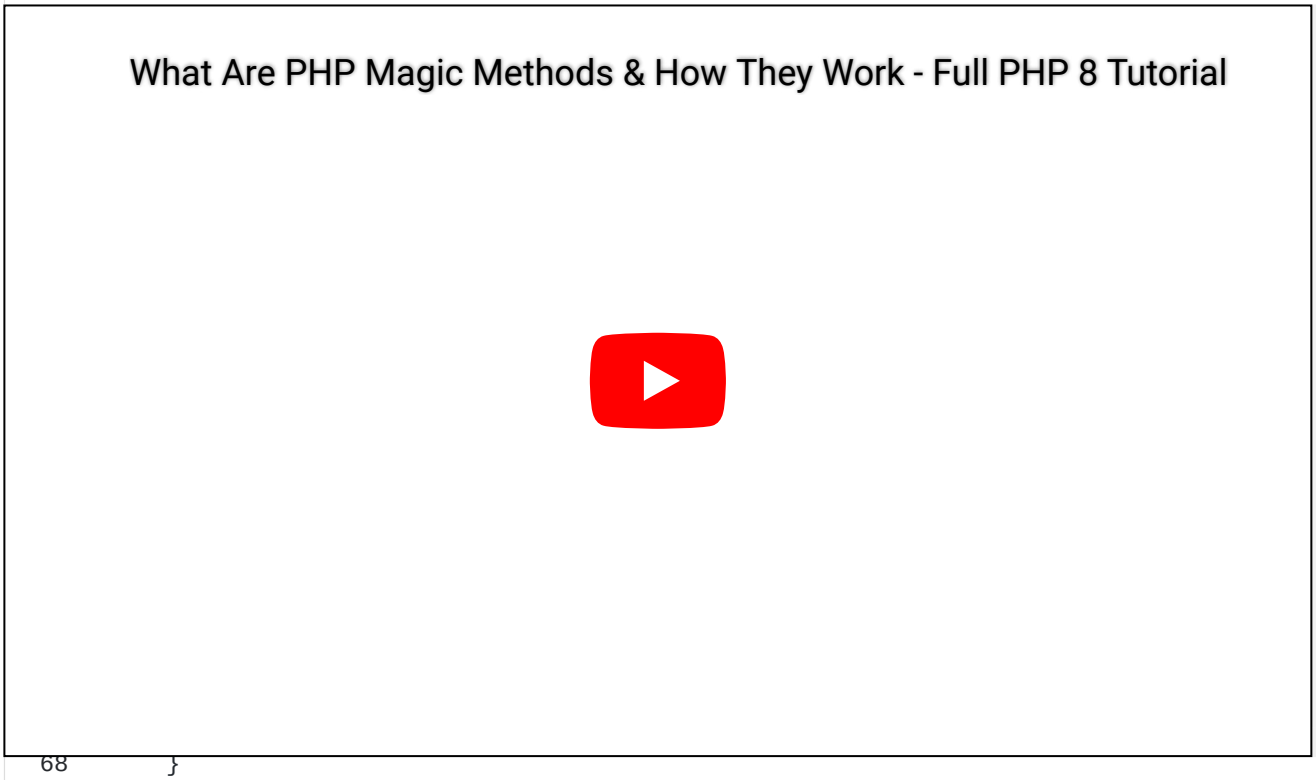
(__get, __set, __isset, __unset)

```php
<?php

class PSession
{
    /**
     * @var array
     */
    private array $options = [];
    /**
     * @var array|string[]
     */
    private array $phpSessionValidOptions = [
        'save_path',
        'name',
        'save_handler',
        'auto_start',
        'gc_probability',
        'gc_divisor',
        'gc_maxlifetime',
        'serialize_handler',
        'cookie_lifetime',
        'cookie_path',
        'cookie_domain',
        'cookie_secure',
        'cookie_httponly',
        'cookie_samesite',
        'use_strict_mode',
        'use_cookies',
        'use_only_cookies',
        'referer_check',
        'cache_limiter',
        'cache_expire',
        'use_trans_sid',
        'trans_sid_tags',
        'trans_sid_hosts',
        'sid_length',
        'sid_bits_per_character',
        'upload_progress.enabled',
        'upload_progress.cleanup',
        'upload_progress.prefix',
        'upload_progress.name',
        'upload_progress.freq',
        'upload_progress.min-freq',
        'lazy_write'
    ];

    /**
     * @param array $options
```

```
49          * @throws InvalidOption
50          */
```




```
68      }
69
70      /**
71       * @return bool
72       */
73      public function destroy(): bool
74      {
75          return session_destroy();
76      }
77
78      /**
79       *
80       */
81      public function clear(): void
82      {
83          session_unset();
84      }
85
86      /**
87       * @return bool
88       */
89      public function regenerateId(): bool
90      {
91          return session_regenerate_id();
92      }
93
94      /**
95       * @return string
```

```
96          */
97         public function getId(): string
98         {
99             return session_id();
100        }
101
102        /**
103         * @param string $id
104         */
105        public function setId(string $id): void
106        {
107            session_id($id);
108        }
109
110        /**
111         * @return string
112         */
113        public function getName(): string
114        {
115            return session_name();
116        }
117
118        /**
119         * @param string $name
120         */
121        public function setName(string $name): void
122        {
123            session_name($name);
124        }
125
126        /**
127         * @return array
128         */
129        public function all(): array
130        {
131            return $_SESSION ?? [];
132        }
133
134        /**
135         * @return bool
136         */
137        public function isStarted(): bool
138        {
139            return $this->getStatus() === PHP_SESSION_ACTIVE;
140        }
141
142        /**
143         * @return int
```

```
144         */
145        public function getStatus(): int
146        {
147            return session_status();
148        }
149
150        /**
151         * @return bool
152         */
153        public function isNotStarted(): bool
154        {
155            return $this->getStatus() === PHP_SESSION_NONE;
156        }
157
158        /**
159         * @return bool
160         */
161        public function isDestroyed(): bool
162        {
163            return $this->getStatus() === PHP_SESSION_DISABLED;
164        }
165
166        /**
167         * @param string $key
168         * @return mixed|null
169         */
170        public function __get(string $key)
171        {
172            return $this->get($key, null);
173        }
174
175        /**
176         * @param string $key
177         * @param $value
178         */
179        public function __set(string $key, $value): void
180        {
181            $this->set($key, $value);
182        }
183
184        /**
185         * @param string $key
186         * @param null $defaultValue
187         * @return mixed|null
188         */
189        public function get(string $key, $defaultValue = null)
190        {
```

```php
        $keys = explode('.', $key);
        $session = $_SESSION;
        foreach ($keys as $key) {
            // allow both object and array access
            if (is_array($session) && isset($session[$key])) {
                $session = $session[$key];
            } elseif (is_object($session) && isset($session->$key)) {
                $session = $session->$key;
            } else {
                return $defaultValue;
            }
        }

        return $session ?? $defaultValue;
    }

    /**
     * @param string $key
     * @param null $value
     */
    public function set(string $key, $value = null): void
    {
        // both object and array access
        $keys = explode('.', $key);
        $session = &$_SESSION;
        foreach ($keys as $key) {
            if (is_array($session) && !isset($session[$key])) {
                $session[$key] = [];
            } elseif (is_object($session) && !isset($session->$key)) {
                $session->$key = [];
            }

            if (is_array($session)) {
                $session = &$session[$key];
            } elseif (is_object($session)) {
                $session = &$session->$key;
            }
        }

        $session = $value;
    }

    /**
     * @param string $key
     * @return bool
     */
    public function __isset(string $key): bool
    {
```

```
239             return $this->has($key);
240         }
241 
242         /**
243          * @param string $key
244          * @return bool
245          */
246         public function has(string $key): bool
247         {
248             return isset($_SESSION[$key]);
249         }
250 
251         /**
252          * @param string $key
253          */
254         public function __unset(string $key): void
255         {
256             $this->remove($key);
257         }
258 
259         /**
260          * @param string $key
261          */
262         public function remove(string $key): void
263         {
```

PHP Php Developers Php Development Php 7 Php 8