PHP magic methods

Andrii Sukhoi

16.04.2023

Contents

- Special methods automatically called by php in response to some events
- Started with double underscores

https://www.php.net/manual/en/language.oop5.magic.php

__construct()

- automatically called when an object is created from a class

```php
class User
{
    private string $name;

    public function __construct(string $name)
    {
        $this->name = $name;
    }
}
```

```php
require __DIR__ . '/vendor/autoload.php';

$user = new User( name: 'Andrii');
dump($user);
```

```
^ App\User {#2 ▼
  -name: "Andrii"
}
```

```php
class User
{
    private string $name;
    private string $surname;
    private string $signature;

    public function __construct(string $name, string $surname)
    {
        $this->name = $name;
        $this->surname = $surname;
        $this->signature = "{$this->name} {$this->surname}";
    }
}
```

```php
use App\User;

require __DIR__ . '/vendor/autoload.php';

$user = new User( name: 'Andrii',  surname: 'Sukhoi');
dump($user);
```

```
^ App\User {#2 ▼
  -name: "Andrii"
  -surname: "Sukhoi"
  -signature: "Andrii Sukhoi"
}
```

__destruct()

- called when an object is destroyed

```php
<?php

declare(strict_types=1);


namespace App\Destruct;


class User
{
    public function __construct(private string $name)
    {
    }

    public function __destruct()
    {
        dump( var: 'destructing');
    }
}
```

```php
require __DIR__ . '/vendor/autoload.php';

$user = new \App\Destruct\User( name: "Andrew");
dump($user);

unset($user);
```

```
^ App\Destruct\User {#2 ▼
  -name: "Andrew"
}
```

```
^ "destructing"
```

__toString()

```php
<?php

declare(strict_types=1);


require __DIR__ . '/vendor/autoload.php';

$user = new \App\ToString\User( name: 'Andrew');

echo $user;
```

Fatal error: Uncaught Error: Object of class App\ToString\User could not be converted to string in /var/www/html/index.php:10 Stack trace: #0 {main} thrown in **/var/www/html/index.php** on line **10**

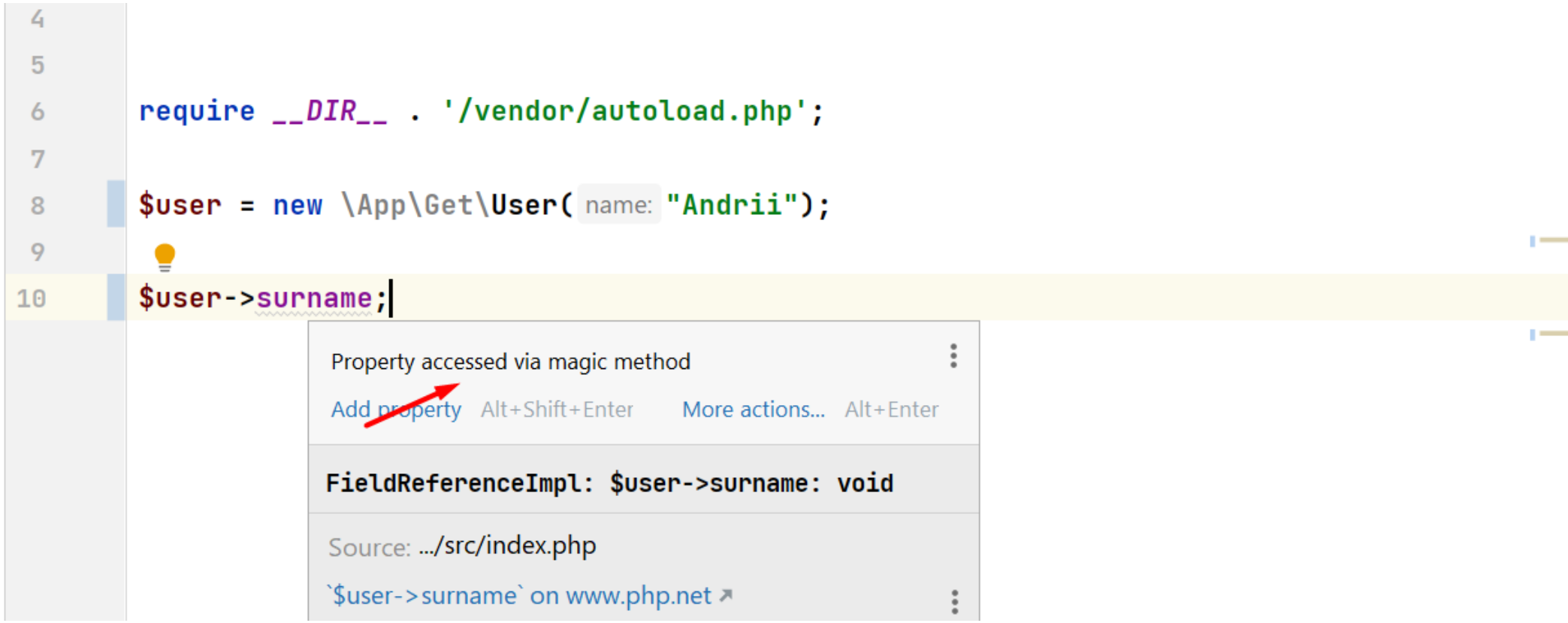
```php
class User
{
    public function __construct(private string $name)
    {}

    public function __toString(): string
    {
        return "String representation of object:
         User name is {$this->name}";
    }
}
```

String representation of object: User name is Andrew

__get and __set()

- __get – called when trying to get properties which are NOT EXIST or are NOT ACCESSIBLE
- __set – called when you are trying to set the value for properties which are NOT EXIS or are NOT ACCESSIBLE

```php
class User
{
    public function __construct(private string $name)
    {}

    // will be called when you try to get unavailable property
    public function __get(string $name) {
        dump($name);
    }

}
```


4
5
6 require __DIR__ . '/vendor/autoload.php';
7
8 $user = new \App\Get\User(name: "Andrii");
9
10 $user->surname;
Property accessed via magic method
Add property Alt+Shift+Enter More actions... Alt+Enter
FieldReferenceImpl: $user->surname: void
Source: .../src/index.php
`$user->surname` on www.php.net

```
^ "surname"
```

```php

class User
{
    public function __construct(private string $name)
    {}

    public function __set(string $name, $value)
    {
        dump( var: "Magic method __set called for $name property. Value = $value");
    }
}
```

```php
declare(strict_types=1);


require __DIR__ . '/vendor/autoload.php';

$user = new \App\Set\User( name: "Andrew");
$user->surname = 'Sukhoi';
```

```
^ "Magic method __set called for surname property. Value = Sukhoi"
```

__isset()

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

__unset()

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

_call()

- __call() is triggered when invoking inaccessible methods in an object context.

__callStatic()

- __callStatic() is triggered when invoking inaccessible methods in a static context.

```php
namespace App\Call;

class User
{
    public function __construct(private string $name)
    {
    }

    public function __call($name, $arguments)
    {
        dump( var: "Calling object method '$name', arguments:"
            . implode( separator: ', ', $arguments). "\n");
    }

    public static function __callStatic($name, $arguments)
    {
        dump ( var: "Calling static method '$name', arguments:"
            . implode( separator: ', ', $arguments). "\n");
    }
}
```

```php

require __DIR__ . '/vendor/autoload.php';

$user = new \App\Call\User( name: "Andrew");
$user->name('Andrii');
$user::name('Andrii');

```

```
^ "Calling object method 'name', arguments:Andrii\n"
```

```
^ "Calling static method 'name', arguments:Andrii\n"
```

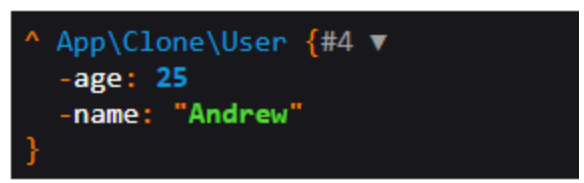
__clone()

- Once the cloning is complete, if a __clone() method is defined, then the newly created object's __clone() method will be called, to allow any necessary properties that need to be changed.

```php
<?php

declare(strict_types=1);


namespace App\Clone;


class User
{
    private int $age = 15;

    public function __construct(private string $name)
    {
    }

    public function __clone()
    {
        $this->age = 25;
    }
}
```

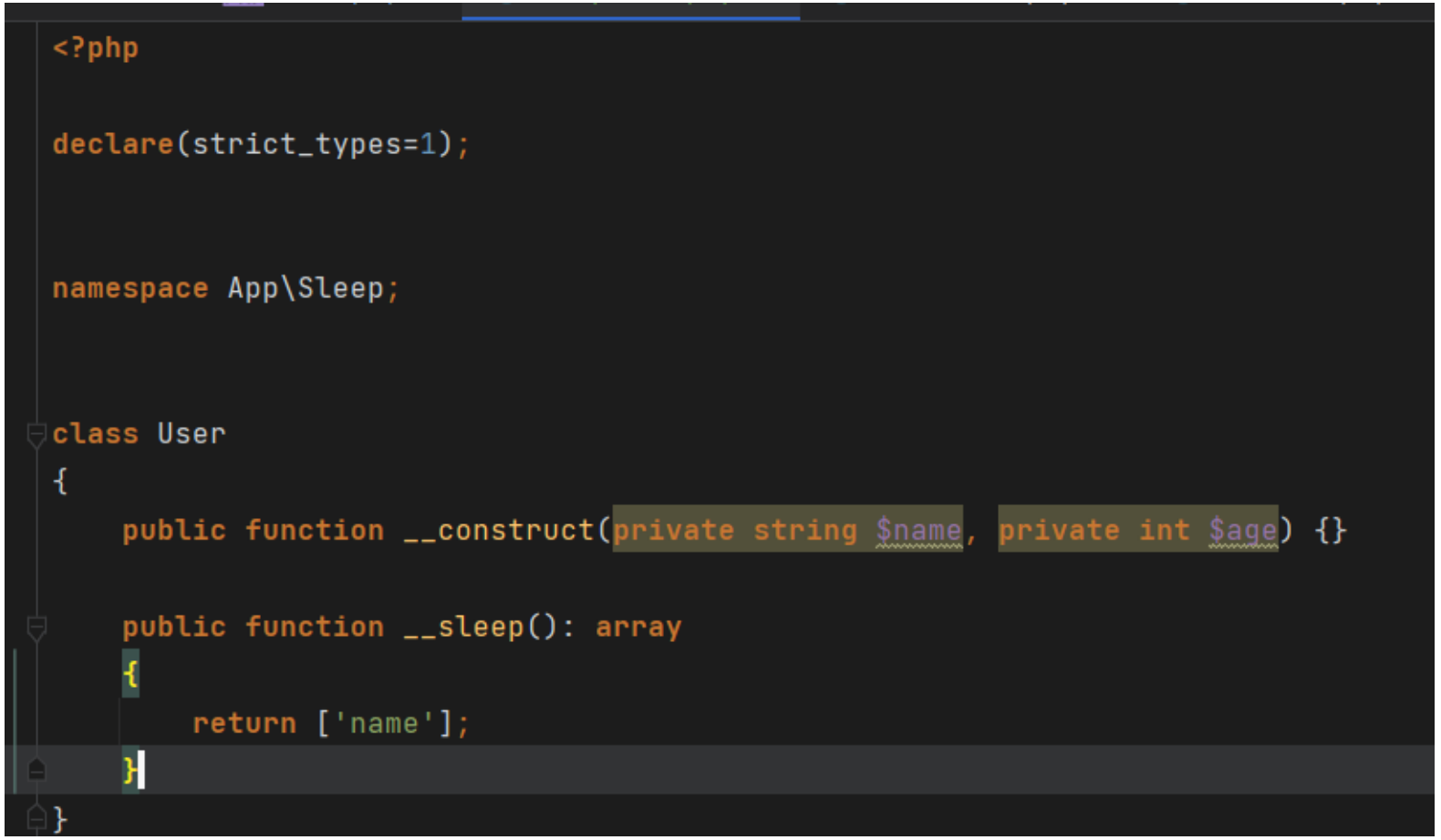
```php
<?php

declare(strict_types=1);



require __DIR__ . '/vendor/autoload.php';

$user = new \App\Clone\User( name: "Andrew");
$andrew2 = clone $user;

dump($user, $andrew2);

```


^ App\Clone\User {#2 ▼
-age: 15
-name: "Andrew"
}


^ App\Clone\User {#4 ▼
-age: 25
-name: "Andrew"
}

__sleep()

- serialize() checks if the class has a function with the magic name __sleep(). If so, that function is executed prior to any serialization. It can clean up the object and is supposed to return an array with the names of all variables of that object that should be serialized. If the method doesn't return anything then null is serialized and E_NOTICE is issued.



```php
<?php

declare(strict_types=1);


namespace App\Sleep;


class User
{
    public function __construct(private string $name, private int $age) {}

    public function __sleep(): array
    {
        return ['name'];
    }
}
```

```php
declare(strict_types=1);


require __DIR__ . '/vendor/autoload.php';

$user = new \App\Sleep\User( name: "Andrew",  age: 28);

echo serialize($user);
```

O:14:"App\Sleep\User":1:{s:20:"App\Sleep\Username";s:6:"Andrew";}

__wakeup()

- unserialize() checks for the presence of a function with the magic name __wakeup(). If present, this function can reconstruct any resources that the object may have. The intended use of __wakeup() is to reestablish any database connections that may have been lost during serialization and perform other reinitialization tasks.

```php
class User
{
    private $id;
    private $name;
    private $age;

    public function __construct($id, $name, $age)
    {
        $this->id = $id;
        $this->name = $name;
        $this->age = $age;
    }

    // ... other methods ...

    public function __sleep()
    {
        // Serialize only the id and name properties
        return array('id', 'name');
    }

    public function __wakeup()
    {
        // Restore the age property from the database during deserialization
        $this->age = $this->fetchAgeFromDatabase($this->id);
    }

    private function fetchAgeFromDatabase($id)
    {
        // Fetch the age from the database based on the id
        // ... implementation ...
    }
}
```

__invoke()

- The __invoke() method is called when a script tries to call an object as a function.

```php
declare(strict_types=1);


namespace App\Invoke;


class User
{
    public function __construct(private string $name)
    {

    }

    public function __invoke()
    {
        dump( var: '__invoke');
    }
}
```


2
3 declare(strict_types=1);
4
5
6 require __DIR__ . '/vendor/autoload.php';
7
8 $user = new \App\Invoke\User(name: "Andrew");
9 $user();

__serialize() and __unserialize()

- serialize() checks if the class has a function with the magic name __serialize(). If so, that function is executed prior to any serialization. It must construct and return an associative array of key/value pairs that represent the serialized form of the object

- Conversely, unserialize() checks for the presence of a function with the magic name __unserialize(). If present, this function will be passed the restored array that was returned from __serialize(). It may then restore the properties of the object from that array as appropriate.

```php
class User
{
    public function __construct(private string $name, private int $age)
    {
    }

    public function __serialize(): array
    {
        return [
            'name' => $this->name . '_serialized',
        ];
    }
}
```

```php

require __DIR__ . '/vendor/autoload.php';

$user = new \App\Serialize\User( name: "andrew",  age: 28);


echo serialize($user);
```

O:18:"App\Serialize\User":1:{s:4:"name";s:17:"andrew_serialized";}

```php
public function __unserialize(array $data): void
{
    $this->dsn = $data['dsn'];
    $this->username = $data['user'];
    $this->password = $data['pass'];

    $this->connect();
}
```

Thanks