PHP : Sérialisation

L3 Informatique - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2020

Sérialisation/Désérialisation

Sérialisation (alias linéarisation)

- Transmettre des valeurs (scalaires, tableaux, objets) entre pages web en les représentant sous forme de chaînes de caractères (sérialisation) pour pouvoir ensuite les reconstruire (désérialisation).
- Choisir un format de sérialisation standard (eg. JSON).
- Les sérialisations sont des chaînes binaires qui peuvent être stockées en fichier ou base de données (champ de type `BLOB`).
- Mécanisme utilisé par PHP notamment pour la sauvegarde et restitution des données de session.

Fonction `serialize`

`string serialize(mixed $value)`

- S'applique à tout type de valeur sauf `resource`.
- Les références non-circulaires sont perdues.
- Sur les objets :
  - Linéarise les propriétés mais pas les méthodes.
  - Invoque la méthode magique `__sleep` au préalable si elle est définie ou la méthode `serialize` si la classe implémente l'interface `Serializable`.

Fonction `unserialize`

`mixed unserialize(string $str[, array $options])`

- Reconstruit une valeur à partir de sa linéarisation.
- Ne peut reconstruire intégralement un objet que si la définition de sa classe est dans la portée.
- Sur les objets, invoque la méthode magique `__wakeup` au préalable si elle est définie ou la méthode `unserialize` si la classe implémente l'interface `Serializable`.

Exemple de (dé)sérialisation d'objet

serialize-class.php

```php
class A {
    private $entier = 1234;
    public $nom     = "nom";
    public function __construct(int $i,string $n) {
        $this->entier = $i;
        $this->nom = $n;
    }
    public function __toString() : string {
        return $this->entier." -- ".$this->nom;
    }
}
```

Exemple de sérialisation d'objet

serialize.php

```php
include("serialize-class.php");
$a1 = new A(111,"name_a1");
$s = serialize($a1);
echo $s;//O:1:"A":2:{s:9:"Aentier";i:111;s:3:"nom";s:7:"name_a1";}
// enregistrer $s où unserialize.php peut le retrouver
file_put_contents(__DIR__.'/serialize-store.txt', $s);
```

unserialize.php

```php
// nous avons besoin de la définition de la classe
// pour que unserialize() fonctionne
require "serialize-class.php";
$s = file_get_contents(__DIR__.'/serialize-store.txt');
$a = unserialize($s);
echo $a;//111 -- name_a1
```

Interface `Serializable`

interface-serializable.php

```php
interface Serializable {
    public function serialize();
    public function unserialize($serialized);
}
```

Utilisation

- La fonction `serialize` appliquée à un objet appelle la méthode de même nom si la classe de l'objet implémente l'interface `Serializable`.
- La fonction `unserialize` devant retourner un objet appelle la méthode de même nom si la classe implémente l'interface `Serializable`.

Sérialisation et héritage

Classe parente (serialize1.php)

```
1  class Base implements Serializable {
2     private $base_var;
3     public function __construct() {
4        $this->base_var='hello';
5     }
6     public function serialize() {
7        return serialize($this->base_var);
8     }
9     public function unserialize($serialized) {
10       $this->base_var=unserialize($serialized);
11    }
12 }
```

Sérialisation et héritage

Sous-classe (serialize2.php)

```
1  class SubClass extends Base {
2     private $sub_var;
3     public function __construct() {
4        parent::__construct();
5        $this->sub_var='world';
6     }
7     public function serialize() {
8        $base=parent::serialize();
9        return serialize(array($this->sub_var,$base));
10    }
11    public function unserialize($serialized) {
12       $data=unserialize($serialized);
13       $this->sub_var=$data[0];
14       parent::unserialize($data[1]);
15    }
16 }
```