Search

Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

g h
: Goto homepage

g s
: Goto search
(current page)

/
: Focus search box

Ковариантность и контравариантность »
« Объекты и ссылки

- Руководство по PHP
- Справочник языка
- Классы и объекты

Change language: Russian

Сериализация объектов

Сериализация объектов - сохранение объектов между сессиями

Функция serialize() возвращает строковое представление любого значения, которое может быть сохранено в PHP. Функция unserialize() может использовать эту строку для восстановления исходного значения переменной. Использование сериализации для сохранения объекта сохранит все его переменные. Методы в объекте не будут сохранены, только имя класса.

Чтобы десериализовать объект функцией unserialize(), необходимо заранее определить класс этого объекта. То есть, если есть экземпляр класса A, и он будет сериализован, будет получено его строковое представление, которое содержит значение всех переменных, содержащихся в нем. Чтобы восстановить объект из строки в другом PHP-файле, класс A должен быть определён в этом файле заранее. Это можно сделать, например, путём сохранения определения класса A в отдельном файле и подключения этого файла или вызовом функции spl_autoload_register() для автоматического подключения.

```
<?php
// A.php:

class A {
public $one = 1;

public function show_one() {
echo $this->one;
}
}

// page1.php:

include "A.php";

$a = new A;
$s = serialize($a);
// сохраняем $s где-нибудь, откуда page2.php сможет его получить.
file_put_contents('store', $s);

// page2.php:

// это нужно для того, чтобы функция unserialize работала правильно.
include "A.php";

$s = file_get_contents('store');
$a = unserialize($s);

// теперь можно использовать метод show_one() объекта $a.
$a->show_one();
?>
```

Если в приложении сериализуются объекты, которые будут использованы в приложении позже, следуют строгой рекомендации — подключать определение класса для этого объекта во всём приложении. При невыполнении этого требования десериализация объекта пройдёт и без определения класса, но PHP назначит этому объекту класс **__PHP_Incomplete_Class_Name**, у которого нет методов, и сделает объект бесполезным.

Поэтому, как в примере выше, если переменная *$a* стала частью сессии путём добавления нового ключа в суперглобальный массив *$_SESSION*, нужно подключать файл `A.php` на всех страницах, а не только на страницах *page1.php* и *page2.php*.

Обратите внимание, что, кроме уже приведённого совета, можно подключиться к событиям сериализации и десериализации объекта через методы __sleep() и __wakeup(). В методе __sleep() можно управлять тем, какие свойства объекта будут сериализованы.

User Contributed Notes 4 notes

up
down
227
php at lanar dot com dot au ¶
14 years ago
Note that static members of an object are not serialized.
up
down
25
michael at smith-li dot com ¶
8 years ago
Reading this page you'd be left with the impression that a class's `serialize` and `unserialize` methods are unrelated to the `serialize` and `unserialize` core functions; that only `__sleep` and `__unsleep` allow you to customize an object's serialization interface. But look at http://php.net/manual/en/class.serializable.php and you'll see that there is a more straightforward way to control how a user-defined object is serialized and unserialized.
up
down
2
Anonymous ¶
3 years ago
Until such time as these documents are updated, note that `session_register()` is not needed to automatically serialize & unserialize objects in sessions. Any objects in `$_SESSION` are serialized when the session is closed (via `session_write_close()` or upon script exit) & unserialized when opened (via `session_start()`). The note about including classes throughout an app (either by including the definition globally or autoloading it) still holds.
up
down
-8
Yahia Fouda ¶
1 year ago
This trick can be useful when you need to pass object data as strings of text between scripts and applications. Common situations include:

* Passing objects via fields in web forms
* Passing objects in URL query strings
* Storing object data in a text file, or in a single database field

and Sometimes it’s useful to do some cleaning up before serializing an object. For example, you might want to write unsaved object data to a database and close the database connection. Similarly, after you’ve unserialized an object, you might want to restore its database connection and perform other setup tasks so that the new object can be used properly.