Serializing objects - objects in sessions

`serialize` returns a string containing a byte-stream representation of any value that can be stored in PHP. `unserialize` can use this string to recreate the original variable values. Using serialize to save an object will save all variables in an object. The methods in an object will not be saved, only the name of the class.

In order to be able to `unserialize` an object, the class of that object needs to be defined. That is, if you have an object of class A and serialize this, you'll get a string that refers to class A and contains all values of variables contained in it. If you want to be able to unserialize an object of class A in another file, the definition of class A must be present in that file first. This can be done for example by storing the class definition of class A in an include file and including this file or making use of the `spl_autoload_register` function.

```
<?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);
  // store $s somewhere where page2.php can find it.
  file_put_contents('store', $s);

// page2.php:
```

```
  // this is needed for the unserialize to work properly.
  include "A.php";

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

  // now use the function show_one() of the $a object.
  $a->show_one();
?>
```

It is strongly recommended that if an application serializes objects, for use later in the application, the application includes the class definition for that object throughout the application. Not doing so might result in an object being unserialized without a class definition, which will result in PHP giving the object a class of `__PHP_Incomplete_Class_Name`, which has no methods and would render the object useless.

So if in the example above `$a` became part of a session by adding a new key to the `$_SESSION` superglobal array, you should include the file `A.php` on all of your pages, not only `page1.php` and `page2.php`.

Beyond the above advice, note that you can also hook into the serialization and unserialization events on an object using the __sleep() and __wakeup() methods. Using __sleep() also allows you to only serialize a subset of the object's properties.

□□□ #2
□ □□□ □□□ 2026-10-31 17:40:46 UTC
□ □□□ □□□ 2026-10-31 17:40:47 UTC