PHP - Cloning Objects

A PHP statement such as "$obj1 = $obj2" merely creates another reference to the same object in memory. Hence, changes in attribute reflect both in original and duplicate object. The **clone** keyword in PHP creates a shallow copy of an object.

```
$obj2 = $obj1
```

Changes in the original object do not reflect in the shallow copy.

Example

Take a look at the following example –

</> Open Compiler

```
<?php
   class foo {
      var $var1 = 'Hello';
   }
   $x = new foo();
   $y = $x;      # reference copy
   echo $x->var1 . " " . $y->var1 . PHP_EOL;

   $x->var1 = "Hello World";
   echo $x->var1 . " " . $y->var1 . PHP_EOL;
?>
```

It will produce the following **output** –

```
Hello Hello
Hello World Hello World
```

In the first case, **$y** is just a reference copy of **$x**. Hence, any change in **var1** property reflects in both.

However, if we declare **$y** as a clone of **$x**, then any change in the original object is not reflected in its shallow copy.

Example

Take a look at the following example –

```php
<?php
   class foo {
      var $var1 = 'Hello World';
   }

   $x = new foo();

   # shallow copy
   $y = clone $x;
   echo $x->var1 . " " . $y->var1 . PHP_EOL;

   $x->var1 = "Hello PHP";
   echo $x->var1 . " " . $y->var1 . PHP_EOL;
?>
```

It will produce the following **output** –

```
Hello World Hello World
Hello PHP Hello World
```


Example

In the following code, **myclass** has one of attributes as object of address class. An object of myclass is duplicated by assignment. Any change in the value of its embedded address object is reflected in both the objects, but change in the name property is not effected in the cloned object.

```php
<?php
   class address {
      var $city="Nanded";
      var $pin="431601";
      function setaddr($arg1, $arg2) {
         $this->city=$arg1;
         $this->pin=$arg2;
      }
   }
   class myclass {
      var $name="Raja";
      var $obj;
      function setname($arg) {
         $this->name=$arg;
      }
   }

   $obj1=new myclass();
   $obj1->obj=new address();
   echo "original object\n";
   print_r($obj1);
   echo "\n";

   $obj2=$obj1;     # reference copy
   $obj1->setname("Ravi");
   $obj1->obj->setaddr("Mumbai", "400001");
   echo "after change: Original object\n";
   print_r($obj1);
   echo "\nCopied object\n";
   print_r($obj2);
?>
```

It will produce the following **output** –

```
original object
myclass Object
(
  [name] => Raja
  [obj] => address Object
  (
    [city] => Nanded
    [pin] => 431601
```

```
   )
)

after change: Original object
myclass Object
(
   [name] => Ravi
   [obj] => address Object
   (
      [city] => Mumbai
      [pin] => 400001
   )
)

Copied object
myclass Object
(
   [name] => Ravi
   [obj] => address Object
   (
      [city] => Mumbai
      [pin] => 400001
   )
)
```

Using the "clone" Keyword

In a shallow copy, any properties of the original object that are references to other variables will remain references. The clone keyword does not copy the contained objects of the copied objects.

We now create a clone of myclass object, so that **$obj2** is the clone of **$obj1**. We change the name property of **$obj1** from **Raja** to **Ravi**, and also modify the embedded address object. The property change will not reflect in its clone, but the referred address object will be changed.

Example

Take a look at the following example –

```php
<?php
   class address {
      var $city="Nanded";
      var $pin="431601";
      function setaddr($arg1, $arg2) {
         $this->city=$arg1;
         $this->pin=$arg2;
      }
   }
   class myclass {
      var $name="Raja";
      var $obj;
      function setname($arg) {
         $this->name=$arg;
      }
   }
   $obj1=new myclass();
   $obj1->obj=new address();
   echo "original object\n";
   print_r($obj1);
   echo "\n";

   $obj2=clone $obj1;        # clone copy
   $obj1->setname("Ravi");
   $obj1->obj->setaddr("Mumbai", "400001");
   echo "after change: Original object\n";
   print_r($obj1);
   echo "\nCopied object\n";
   print_r($obj2);
?>
```

It will produce the following **output** –

```
original object
myclass Object
(
  [name] => Raja
  [obj] => address Object
  (
    [city] => Nanded
    [pin] => 431601
  )
```

```
)

after change: Original object
myclass Object
(
   [name] => Ravi
   [obj] => address Object
   (
      [city] => Mumbai
      [pin] => 400001
   )
)

Copied object
myclass Object
(
   [name] => Raja
   [obj] => address Object
   (
      [city] => Mumbai
      [pin] => 400001
   )
)
```

Using __clone() Method

The clone keyword creates a shallow copy of the object. When an object is cloned, PHP will perform a shallow copy of all of the object's properties. Any properties that are references to other variables will remain references. Hence, any changes done to the original object will also appear in the cloned object.

If you wish to prevent the copied object to update automatically, we need to create a deep copy of the object with the __clone() method. It is one of the magic methods in PHP.

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.

Example

In the above example, we have an object of myclass, one of its attributes $obj holding the reference to an object of address class. To achieve a deep copy, we override the __clone() magic method in myclass.

</> Open Compiler

```php
<?php
   class address {
      var $city="Nanded";
      var $pin="431601";
      function setaddr($arg1, $arg2) {
         $this->city=$arg1;
         $this->pin=$arg2;
      }
   }
   class myclass {
      var $name="Raja";
      var $obj;
      function setname($arg) {
         $this->name=$arg;
      }
      public function __clone() {
         $this->obj = clone $this->obj ;
      }
   }
   $obj1=new myclass();
   $obj1->obj=new address();
   echo "original object\n";
   print_r($obj1);
   echo "\n";

   $obj2=clone $obj1;        # cloned deep copy
   $obj1->setname("Ravi");
   $obj1->obj->setaddr("Mumbai", "400001");
   echo "after change: Original object\n";
   print_r($obj1);
   echo "\nCloned object\n";
   print_r($obj2);
?>
```

You will now see that the changes in the original object (we change the address attributes) won't reflect in the cloned object, as the following **output** shows –

```
original object
myclass Object
```

```
(
  [name] => Raja
  [obj] => address Object
  (
    [city] => Nanded
    [pin] => 431601
  )
)

after change: Original object
myclass Object
(
  [name] => Ravi
  [obj] => address Object
  (
    [city] => Mumbai
    [pin] => 400001
  )
)

Cloned object
myclass Object
(
  [name] => Raja
  [obj] => address Object
  (
    [city] => Nanded
    [pin] => 431601
  )
)
```