PHP - Object Iteration

A **foreach** loop may be employed to iterate through all the publicly visible members of an object of a PHP class. This feature has been available in versions of PHP 5 onwards. You can of course access the list of private properties inside an instance method. PHP also defines Iterator interface which can be used for the purpose.

Using foreach Loop

In the example below, the public properties of the class are listed with the use of **foreach** loop.

Example

</> Open Compiler

```
<?php
   class myclass {
      private $var;
      protected $var1;
      public $x, $y, $z;
      public function __construct() {
         $this->var="Hello World";
         $this->var1=array(1,2,3);
         $this->x=100;
         $this->y=200;
         $this->z=300;
      }
   }
   $obj = new myclass();
   foreach($obj as $key => $value) {
      print "$key => $value\n";
   }
?>
```

It will produce the following **output** –

```
x => 100
y => 200
```

```
z => 300
```

Note that only the public members are accessible outside the class. If the class includes a method, all the members (public, private or protected) can be traversed with a **foreach** loop from inside it.

Let us add an iterate method in the above **myclass**.

```
public function iterate() {
   foreach ($this as $k=>$v) {
      if (is_array($v)) {
         var_dump($v);
         echo PHP_EOL;
      } else {
         echo "$k : $v". PHP_EOL;
      }
   }
}
```

Call this instance method to get the list of all the members.

It will produce the following **output** –

```
var : Hello World
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  int(3)
}
x : 100
y : 200
z : 300
```

Using Iterator Interface

PHP provides Iterator interface for external iterators or objects that can be iterated themselves internally. It defines following abstract methods which need to be implemented in the user defined class.

```
interface Iterator extends Traversable {
   /* Methods */
   public current(): mixed
   public key(): mixed
   public next(): void
   public rewind(): void
   public valid(): bool
}
```

- The rewind() method rewinds the Iterator to the first element. This is the first method called when starting a foreach loop. It will not be executed after foreach loops.
- The current() method returns the current element.
- The key() method returns the key of the current element on each iteration of foreach loop.
- The next() method is called after each iteration of foreach loop and moves forward to next element.
- The valid() method checks if current position is valid.

Example

The following example demonstrates object iteration by implementing Iterator interface

</> Open Compiler

```
<?php
   class myclass implements Iterator {
      private $arr = array('a','b','c');

      public function rewind():void {
         echo "rewinding\n";
         reset($this->arr);
      }

      public function current() {
         $var = current($this->arr);
         echo "current: $var\n";
         return $var;
      }
```

```
      public function key() {
         $var = key($this->arr);
         echo "key: $var\n";
         return $var;
      }

      public function next() : void {
         $var = next($this->arr);
         echo "next: $var\n";
         # return $var;
      }

      public function valid() : bool {
         $key = key($this->arr);
         $var = ($key !== NULL && $key !== FALSE);
         echo "valid: $var\n";
         return $var;
      }
   }

   $obj = new myclass();

   foreach ($obj as $k => $v) {
      print "$k: $v\n";
   }
?>
```

It will produce the following **output** –

```
rewinding
valid: 1
current: a
key: 0
0: a
next: b
valid: 1
current: b
key: 1
1: b
next: c
valid: 1
current: c
```

```
key: 2
2: c
next:
```