Experiment No: 09

Write a simple PHP program on Introspection and Serialization.

Resources required:

Hardware	**Software**
Computer System	Any database tools such as XAMPP

Practical Significance:

Introspection

– Introspection in PHP offers the useful ability to examine an object's characteristics, such as its name, parent class
(if any) properties, classes, interfaces and methods.

– PHP offers a large number functions that you can use to accomplish the task.

– In-built functions in PHP Introspection :

Function	**Description**
class_exists()	Checks whether a class has been defined.
get_class()	Returns the class name of an object.
get_parent_class()	Returns the class name of an object's parent class.
is_subclass_of()	Checks whether an object has a given parent class.
get_declared_classes()	Returns a list of all declared classes.
get_class_methods()	Returns the names of the class' methods.
get_class_vars()	Returns the default properties of a class.
interface_exists()	Checks whether the interface is defined.
method_exists()	Checks whether an object defines a method.

Program Code:

```
<?php
class Derived
{
   public function details()
   {
      echo "I am a Derived(super) class for the Child(sub) class. <BR>";
   }
}
class sub extends Derived
```

```
{
   public function details()
 {
       echo "I'm " .get_class($this) , " class.<BR>";
       echo "I'm " .get_parent_class($this) , "'s child.<BR>";
   }
}
//details of parent class
if (class_exists("Derived"))
{
   $der = new Derived();
   echo "The class name is: " .get_class($der) . "<BR>";
   $der->details();
}
//details of child class
if (class_exists("sub"))
{
   $s = new sub();
   $s->details();

   if (is_subclass_of($s, "Derived"))
   {
      echo "Yes, " .get_class($s) . " is a subclass of Derived.<BR>";
   }
   else
   {
      echo "No, " .get_class($s) . " is not a subclass of Derived.<BR>";
   }
}
```

Output :

```
The class name is: Derived
I am a Derived (super) class for the Child(sub) class.
I'm sub class.
I'm Derived's child.
Yes, sub is a subclass of Derived.
```

Serialization

– Serialization is a technique used by programmers to preserve their working data in a format that can later be restored to its previous form.

– Serializing an object means converting it to a byte stream representation that can be stored in a file. Serialization in PHP is mostly automatic, it requires little extra work from you, beyond calling the serialize () and unserialize( ) functions.

Serialize() :

– The serialize() converts a storable representation of a value.

– The serialize() function accepts a single parameter which is the data we want to serialize and returns a serialized string.

– A serialize data means a sequence of bits so that it can be stored in a file, a memory buffer or transmitted across a network connection link. It is useful for storing or passing PHP values around without losing their type and structure.

Syntax :

```
serialize(value1);
```

unserialize() : unserialize() can use string to recreate the original variable values i.e. converts actual data from serialized data.

Syntax :

```
unserialize(string1);
```

Program Code:

```
<?php
$s_data= serialize(array('Welcome', 'to', 'PHP'));
print_r($s_data . "<br>");
$us_data=unserialize($s_data);
print_r($us_data);
?>
```

Output :

```
a:3:{i:0;s:7:"Welcome";i:1;s:2:"to";i:2;s:3:"PHP";}
Array ( [0] => Welcome [1] => to [2] => PHP )
```

Practical related questions:

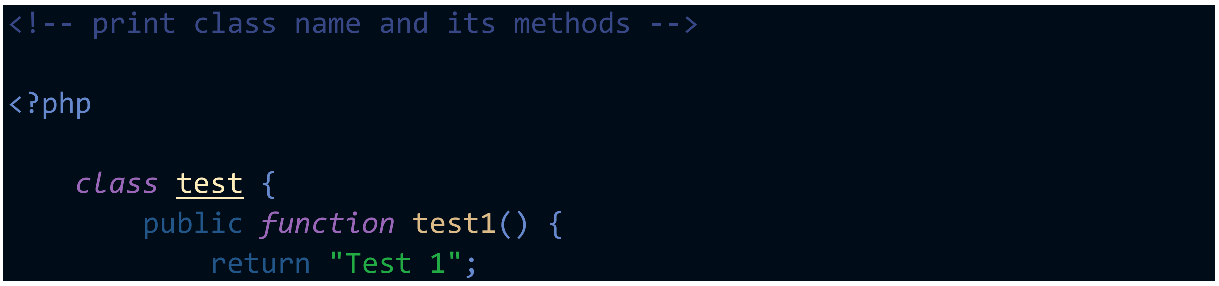
1. State the use of unserialize().
   unserialize() is converts serialized data to normal form

2. List the Magic Methods with PHP Object Serialization.

   serialize() and unserialize()

Exercise:

1. Declare a class as “Test” with three user defined functions. List name of the class and functions declared in the class “Test”.



```
<!-- print class name and its methods -->

<?php

    class test {
        public function test1() {
            return "Test 1";
        }
        public function test2() {
            return "Test 2";
        }
        public function test3() {
            return "Test 3";
        }
    }

    $t = new test();
    echo get_class($t);
    echo "<br />";
    print_r(get_class_methods($t));

?>

<!-- output -->
<!-- test
Array ( [0] => test1 [1] => test2 [2] => test3 )  -->
```

2. Declare an interface “MyInterface’. Write a script for whether interface exits or not.

```
<!-- Interface Exists -->

<?php

    interface testI {
        public function test1();
        public function test2();
    }

    if(interface_exists('testI')) {
        echo "Interface testI exists";
    } else {
        echo "Interface testI does not exists";
    }

?>

<!-- Output -->
<!-- Interface testI exists -->
```

3. Write a script to based on unserialize().

```
<!-- Serialize and Unserialize -->

<?php

$s_data= serialize(array('I', 'am', 'Jayesh'));

print_r($s_data . "<br>");

$us_data=unserialize($s_data);

print_r($us_data);

?>

<!-- Output -->
<!-- a:3:{i:0;s:1:"I";i:1;s:2:"am";i:2;s:6:"Jayesh";}
Array ( [0] => I [1] => am [2] => Jayesh )  -->
```