SEHS4517 Web Application Development and Management

Lecture 7 - Object Orient Programming in PHP, PHP Data Objects

Learning Objectives

- Relationship between Class, Objects, Attributes & Methods
- Instantiate objects from a class
- Assign values to attributes
- Invoke methods
- Use PDO to connect and read records from MySQL

Object Oriented Programming (OOP)

- The most used programming paradigms
- The preferred approach since PHP 5
- Libraries are evolving toward OOP

Object Oriented Programming (OOP)
Class, Attributes, Methods

Terminology: Class

- A class provides the basic characteristics of an object in real-life. In a bookstore, we can have a class called Book.
- A class can have one or more of the followings:
  - Attributes (also called properties)
    - These can be used to describe a thing in that class.
    - E.g. title, author, quantity
  - Methods (also called behaviors, operations, events)
    - These are what the class can do or what can happen to the class.
    - E.g. borrow, return, reserve
- Think of the class is a blueprint or a definition that describes the nature of something.

Creating a Class with Attributes

- To create a new class, use the keyword **class**.
- Here, we create the **Book.** It is to be used an application for a bookstore.
- The public key word means the attribute can be seen by all other codes in the application.
- Think of properties as variables inside the class.

```
<?php
  class Book {
        public $title;
        public $author;
        public $qty;
  }
?>
```

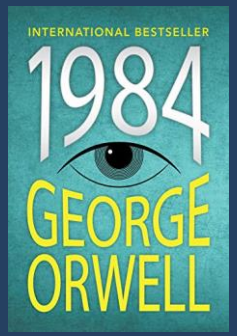
Poll

Terminology: Instance

- An instance is the actual object created from a class definition.
- To create a new instance, use the keyword new. This is known as instantiating an object or instantiation.
- Object and Instance are often used interchangeably.
- Here, we create two instances - $book1 and $book2.
- Note the use of the object operator ' -> ' .

```
<?php
  $book1 = new Book();
  $book1->title  = "1984";
  $book1->author = "George Orwell";

  $book2 = new Book();
  $book1->title  = "Spider-man";
  $book1->author = "Stan Lee";
  ?>
```



The object operator ->

- The object operator, ' -> ', is simply PHPs way of accessing, running, or assigning "stuff" within an object.
- It is used to access methods and properties of an object.
- It is also known as the dash.
- Exercise:
  - Write a line that creates an instance called $c from the class Customer.
  - Write the line that assigns the value of 10 to $book1's quantity.
  - Write the line that prints the value $book2's quantity.

Poll

Data types of Attributes

- You can use print_r() or var_dump() to show the values of the attributes of an object.

```
print_r($book1);
```

```
Book Object
(
    [isbn] =>
    [title] => Spider-man
    [author] => Stan Lee
    [quantity] =>
)
```

- You can see that properties have a type at the moment of printing, but we did not define this type explicitly.
- The variable took the type of the value assigned. This works in the same way that normal PHP variables do.

Terminology: Method

- Methods are functions defined inside a class.
- Methods can have optional arguments and returns.
- Here is an example of a method in the Book class.

```
class Book {
\\ codes for attributes skipped
public function getCopy(): bool {
  if ($this-> qty < 1) {
     return false;
  } else {
      $this-> qty--;
      return true;
  }
}
```

- We use $this to refer to the object itself. It allows you to access the properties and methods of that same object.

Poll

Methods can update attributes

- In a method, you can also update the values of the current object from one of its functions.
- For example, `getCopy()` reduces $qty by one when a customer gets one copy of the book.
- How the `getCopy()` method works?
  - First, it checks if we have at least one available unit. If we do not, we return false to let them know that the operation was not successful.
  - If we do have a unit for the customer, we decrease the number of available units, and then return true, letting them know that the operation was successful.

Complete Example of using a Method

- Here is how the method of `getCopy()` will work.

```
<?php
  $b1 = new Book();
  $b1->title = "1984";
  $b1->author = "George Orwell";
  $b1->qty = 12;
  if ($b1->getCopy()) {
    echo 'Here, your copy.';
  } else {
    echo 'I am afraid that book is not available.';
}
```

- We use $this to refer to the object itself. It allows you to access the properties and methods of that same object.

Method that produces HTML

- Note: It is a not a good idea to use a function to print output directly, or to produce HTML code.
- The following example is just what a function may do.

```
class Book {  // in a file called Book.class-simple.php
    public $title;
    public $cover;

    public function showCover(){
        echo "<div style='float: left; margin-right:40px;'>";
        echo "<font size='3'>{$this->title}</font><br/>";
        echo "<img src='img/{$this->cover}'/><br/>";
        echo "</div>";
    } // end showCover ()

} // end class Book
```

Method that produces HTML

- Here is the PHP script that actually uses the method to produce a web page.

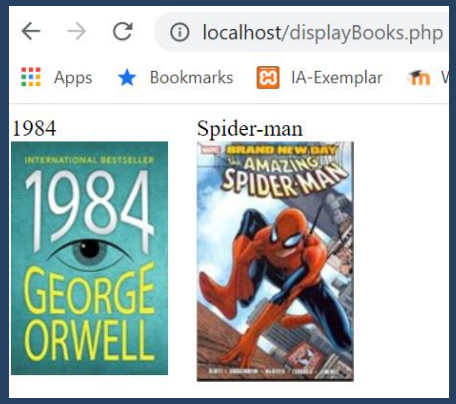
```
<?php

include 'Book.class-simple.php';

$b1 = new Book;
$b1->title = '1984';
$b1->cover = '1984-small.jpg';
$b1->showCover();

$b2 = new Book;
$b2->title = 'Spider-man';
$b2->cover = 'spider-man-small.jpg';
$b2->showCover();

?>
```



Object Oriented Programming (OOP)
Constants, Constructors

Constants in a class

- A class constant is declared inside a class with the const keyword.
- Class constants are case-sensitive.
- Suggestion: Name the constants in all uppercase letters.
- You can access a constant from inside the class by using self followed by the scope resolution operator (::) followed by the constant name.

```
class Book {
    const SFONT = "<font size='3'>";
    public $cover;
    public function showTitle(){
        echo self::SFONT.$this->title."</font><br/>";
} // end showCover ()
```

Constructors, why do we need them?

- The following code has two problems:

```
<?php
  $b1 = new Book();
  $b1->title = "1984";
  $b1->author = "George Orwell";
  $b1->qty = 12;
  // rest of application code skipped
}
```

- Problems:
  1. It is clumsy to use multiple lines.
  2. The properties may be assigned wrong values.
     - E.g. `$b1->qty = 'a lot';`

Poll

Class constructors

- The constructor of a class is a function that is used to create an instance of that class.
- The name is always `__construct()`.
- Note that there are two underscores (i.e. _ _).
- To instantiate the Book class, we use the following:

```
public function __construct(string $title, string $author,
int $qty) {
        $this->title  = $title;
        $this->author = $author;
        $this->qty    = $qty;
}
$book = new Book("1984", "George Orwell", 12);
```

Constructors
Optional arguments

- As a constructor is still a function, it can use default arguments.

```
public function __construct(
        string $title,
        string $author,
        int $qty = 0
) {
        $this->title = $title;
        $this->author = $author;
        $this->qty = $qty;
}
```

Private Attributes

- **Problem:** Even when constructors are available, we need to prevent programmers from assigning values to attributes directly such as follows:

```
$bx = new Book();
$bx->qty = -1;
```

- **Solution: M**ake the attributes private, so that no code outside the class can use them directly.

```
class Book {
        private $title;
        private $author;
        private $qty;
}
```

PHP Data Objects (PDO)

What is PDO

- The PHP Data Objects (PDO) extension defines a lightweight, consistent interface for accessing databases in PHP.
- Note that you cannot perform any database functions using the PDO extension by itself; you must use a database-specific PDO driver to access a database server.
- PDO provides a data-access abstraction layer. This means that, regardless of which database you're using, you use the same functions to issue queries and fetch data.
- PDO requires the new OO features in the core of PHP 5, and so will not run with earlier versions of PHP.

What Databases are Supported?

- At this time PDO offers the following drivers:
  - MySQL 3,4,5 (depends on client libs)
  - PostgreSQL
  - SQLite 2 & 3
  - ODBC
  - DB2
  - Oracle
  - Firebird
  - FreeTDS/Sybase/MSSQL

Using PDO to connect to different databases

- MySQL connection
  - new PDO('mysql:host=localhost;dbname=testdb', $login, $passwd);
- PostgreSQL
  - new PDO('pgsql:host=localhost port=5432 dbname=testdb user=john password=mypass');
- SQLite
  - new PDO('sqlite:/path/to/database_file');

PDO - Connect to the database

- In order to connect to the database, we need to instantiate an object from the PDO class.
- The constructor of this class expects three arguments:
  - **Data Source Name** (**DSN**), which is a string that represents the type of database to use;
  - the name of the user;
  - the password

```
$pdo = new PDO(
'mysql:host=127.0.0.1;dbname=bookstore',
        'root',
        'foo1234',
);
```

PDO - Selecting records

- In order to connect to the database, we need to instantiate an object from the PDO class.

```
$pdo=new PDO('mysql:host=localhost… // code skipped
$s = $pdo->query("SELECT * FROM users");
while ( $row = $s->fetch(PDO::FETCH_ASSOC) ) {
    print_r($row);
}
```

```
mysql> select * from users;
+----+-------+---------------+----------+
| id | name  | email         | password |
+----+-------+---------------+----------+
|  1 | Chuck | csev@umich.edu | 123      |
|  2 | Glenn | gg@umich.edu   | 456      |
+----+-------+---------------+----------+
```

```
Array(
    [id] => 1
    [name] => Chuck
    [email] => csev@umich.edu
    [password] => 123
)

Array(
    [id] => 2
    [name] => Glenn
    [email] => gg@umich.edu
    [password] => 456
)
```

Summary

- In this lecture, we have learnt the following:
  - Relationship between Class, Objects, Attributes & Methods
  - Instantiate objects from a class
  - Assign values to attributes
  - Invoke methods
  - Use PDO to connect and read records from MySQL