**Object-Oriented Programming Basics**

Before you can start working on OOP coding, you need to know how OOP works. OOP uses a completely different paradigm than how you've been coding so far. This requires that you think differently about how your programs work and how you code them.

So far, you've been writing our PHP code using the *procedural style* of programming. In the procedural style of programming, you create variables and functions within your PHP code to perform certain procedures, such as retrieve data records from the MySQL database and display them on a Web page. The data that you use and the functions you create are completely separate entities with no specific relation to each other.

In OOP, everything is related to a *class*. A class defines the characteristics of an object that you're using in your application. Every application revolves around handling objects. Our Food Store application revolves around handling three objects: products, customers, and orders.

OOP classes define the objects that your application uses. Each class contains both the data and functions required to interact with the object. Once you define the objects, all your application code needs to do is use the data and functions defined for the objects to process and manipulate them.

The benefit to OOP is that once you create a class for an object, you can use that same object anytime in any other application, because you already have the code handy for using that object. You just plug in the class definition code, and you can use it in your application.

An OOP class is comprised of *members*. There are two types of members. Class *properties* define attributes for the object (such as the description, price, and quantity in stock of a product). A class can contain many different property members with each property describing a different feature of the object.

The other type of class member is *methods*. A class method is similar to the standard PHP functions that you've already been using. Just like a function, the method performs an operation using the properties you define in the class. You create class methods to perform specific functions on the class data, such as a method to buy a product (where you subtract a value from the quantity property) or change the price of the product (where you add or subtract a value to the price property). Each class should be self-contained. The methods in a class should only operate on properties within the class and shouldn't deal with properties in other classes.

**Creating a Class in PHP**

If your head is spinning right now, don't worry. Hopefully we can clear this up by looking at some code. The first thing you need to do in OOP is create a class. You do that by using a *class definition*. The class definition declares all the members that comprise the class, both properties and methods. Here's an example of a simple class definition in PHP:

class Product {  
public $description;  
public $price;  
public $quantity;  
public $onsale;

public function buyProduct($amount) {  
$this->quantity -= $amount;  
}  
}

This class defines four property members and one method member. Each member is defined using one of three *visibility* classifications. The visibility of a member determines where you can use or reference that member. There are three visibility keywords used in PHP:

* **Public:** The member can be accessed from outside of the class code.
* **Private:** The member can only be accessed from inside the class code.
* **Protected:** The member can only be accessed from a child class (we'll talk about that a little later).

The Product class example declares all of the members as public, so you can reference them anywhere in your PHP code.

The buyProduct() method uses an odd variable name in the function, *$this->quantity*. The *$this* variable is a special identifier that references the current class object. In this example, it points to the $quantity property of the class. Notice the removal of the dollar sign from the quantity variable when referencing it this way. This helps PHP know that you're referencing the $quantity variable from within the class.

Now, let's take a look at how to create an actual object using the Product class.

|  |
| --- |
| **Warning:** Between PHP versions 4.x and 5.x PHP vastly changed the way to define and use classes. The use of the code in this lesson follows the PHP version 5 standards. If you're using a PHP version 4.x server, please consult the PHP online manual for how to create and use classes. You can find the URL for that section of the manual in the Supplemental Material section of this lesson. |

**Creating Objects**

Your PHP code uses the class definition to define objects. *Instantiating* is the process of creating an object using a class definition. Once you instantiate a class into an object, you can use the object within your application code to reference properties and methods. To instantiate an object in PHP code, you use the following format:

$prod1 = new Product();

This creates the object called $prod1 using the Product class. Once you instantiate an object, you can access the public members of that class from anywhere in your application:

$prod1->description = "carrot";  
$prod1->price = 1.50;  
$prod1->quantity = 10;

This code sets values for the properties of this object. Notice that you must use the -> symbol to reference the property of the object. The $prod1 variable now contains these values for the properties, and you can use them anywhere within your application code to reference these values. The same applies when you need to use a public method of an object:

$prod1->buyProduct(4);

This calls the buyProduct() method, passing the value 4. Since the buyProduct() method alters the $quantity value in the object, the next time you reference $prod1->quantity, it'll have the value 6.

**Writing OOP Code in PHP**

Let's write an example program using the Products class so we can see OOP in action. Just follow these steps:

1. Create a folder called *oop* in the WampServer www folder.
2. Create a file called *example1.php* in the oop folder.
3. Open the file in a text editor, and add the following code:

<?php

class Product {

public $description;

public $price;

public $quantity;

public function printProduct() {

echo "Product: $this->description<br>\n";

printf("Price: $%.2f<br>\n", $this->price);

echo "Quantity: $this->quantity<br>\n";

}

public function buyProduct($amount) {

$this->quantity -= $amount;

}

}

$prod1 = new Product();

$prod1->description = "Carrots";

$prod1->price = 1.50;

$prod1->quantity = 10;

echo "Just added product:<br>\n";

$prod1->printProduct();

echo "<br>Buying 4 carrots.\n";

$prod1->buyProduct(4);

echo "Quantity is now: $prod1->quantity<br>\n";

?>

1. Save the file, and exit the text editor.
2. Open your browser, and go to the URL *http://localhost/oop/example1.php*. When you view the example1.php Web page, you should see the following output:

The output of the example1.php program

This example defines the Product class, which contains the four properties we discussed, plus two class methods: the buyProduct() method you've already seen and the printProduct() method. The printProduct() method is a quick way to print the property values of the Product object. You can use this method anytime you need to display the object values in your application.

That covers the bare basics of OOP. Now you can say you've written and used an OOP program! Let's move on to Chapter 3 and look at some more features of OOP.