From Technologies to Solutions

Object-Oriented Programming with PHP5

Learn to leverage PHP5's OOP features to write manageable applications with ease

Hasin Hayder

PACKT PUBLISHING

Object-Oriented Programming with PHP5

Learn to leverage PHP5's OOP features to write manageable applications with ease

Hasin Hayder

PACKT PUBLISHING

BIRMINGHAM - MUMBAI

Object-Oriented Programming with PHP5

First published: December 2007

Production Reference: 1031207

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-847192-56-1

www.packtpub.com

Cover Image by Karl Moore (karl.moore@ukonline.co.uk)

Credits

Author

Hasin Hayder

Reviewers

Kalpesh Barot

Murshed Ahmed Khan

Development Editor

Nanda Padmanabhan

Assistant Development Editor

Rashmi Phadnis

Technical Editor

Divya Menon

Editorial Team leader

Mithil Kulkarni

Project Manager

Abhijeet Deobhakta

Indexer

Monica Ajmera

Proofreader

Damian Carvill

Production Coordinator

Shantanu Zagade

Cover Designer

Shantanu Zagade

About the Author

Hasin Hayder is a Zend Certified Engineer and open-source enthusiast from Bangladesh. Besides his regular job as Technical Director at Trippert Labs (`www.trippert.com`), he is often found developing localized Bangla applications and blogging at `http://hasin.wordpress.com`. He lives in Bangladesh with his wife Ayesha, son Afif and plenty of toys around!

About the Reviewers

Kalpesh Barot has about 4 years of experience in the world of PHP. He has extensively worked on small and large scale social networking websites developed in PHP. He has been involved in varied projects, from planning and developing web sites to creating custom modules on big social networking websites.

Kalpesh received a Masters degree in Enterprise software Engineering from the University of Greenwich, UK in 2004. There he learned the theory behind his computer experience and became a much more efficient computer programmer.

Kalpesh has worked actively in the IT sector since his freshman year at university. He has been a PHP developer since then and has developed his skills in this field.

Through his increasing responsibilities, he has learned to prioritize needs and wants, and applies this ability to his projects.

I would like to thank my wife Bansari for her consistent support.

Murshed Ahmmad Khan is a young web developer who believes that nothing is impossible in the arena of programming. With his extensive 5 years work experience in web & system level programming he wants to create cool, applicable and useful systems for many people throughout the web.

He graduated (B.Sc. in CSE) from Rajshahi University of Engineering & Technology (RUET) Rajshahi, Bangladesh, in Computer Science & Engineering (CSE).

Murshed Ahmmad Khan worked on BangladeshInfo.com (`http://www.bangladeshinfo.com`), and Global Online Services Limited (`http://www.global.com.bd`) gaining an immense reputation. BangladeshInfo.com & Global Online Services Limited are both a concern of Texas Group Bangladesh and a renowned IT firm in the local market for corporate and multinational companies.

He also worked in THPB (The Hunger Project, Bangladesh - `http://www.thp.org`) and SHUJAN (SHUJAN is a citizen movements to achieve good governance) as a lead developer for developing various e-governance sites for increasing the accountability of the candidates of national elections. From SHUJAN (`http://www.shujan.org`) he also developed the country's first ever online.

Table of Contents

Introduction

Object-oriented programming is largely about the ability to hide what's not important to the user and to highlight what is. PHP 5 offers standardized means for specifying the variety of property scopes typically offered by full-featured OO languages.

What This Book Covers

Chapter 1 introduces object-oriented programming and how it fits for PHP. Some benefits of functional programming over procedural programming are highlighted.

In *Chapter 2* you learn to create objects and define their properties and methods. Details of classes, properties, and methods follow, along with the scope of methods. This chapter shows you the benefits of using interfaces and a few other basic OOP features in PHP to kick start your journey through OOPing in PHP.

Now that you have got your basics done for OOP in PHP, *Chapter 3* helps you to strengthen your base. It helps you to deal with more details and some advanced features. For example, you learn about class information functions, which allows you to investigate details of any class. This chapter takes you through some handy object-oriented information functions, exception handling, iterators, and storing objects using serialization.

In *Chapter 4* you learn some of the Design Patterns and how to implement them in PHP. These are an essential part of OOP and make your code more effective, more efficient, and easier to maintain. Sometimes we implement these design patterns in our code without knowing that these solutions are defined by design patterns. Proper usage of the correct pattern can make your code perform better; similarly using them improperly could make your code slower and less efficient.

Chapter 5 focuses on two very important features of object-oriented programming in PHP, reflection and unit testing. PHP5 replaces many old APIs with smarter new ones. One of these is the Reflection API, with which you can reverse or engineer any class or object to figure out its properties and methods. You can invoke those methods dynamically and more. Unit testing is an essential part of good, stable, and manageable application design. We focus on one very popular package, PHPUnit, which is a port of JUnit to PHP. If you follow the guidelines provided in this chapter you will be able to design your own unit tests successfully.

Some built-in objects and interfaces in PHP make life much easier for PHP developers. In *Chapter 6* you will learn about the huge object repository named the Standard PHP Library or SPL.

Chapter 7: In this chapter we discuss the improved MySQL API known as MySQLi and take a basic look at PHP Data Objects (PDO), adoDB, and PEAR::MDB2. We take a look at the Active Record pattern in PHP using adoDB's active record library and the Object-Relational Mapping (ORM) pattern using Propel. We focus on some specific topics that are interesting for PHP developers doing database access the OO way.

In *Chapter 8*, you learn to process XML with PHP. You get to know about different APIs like the SimpleXML API to read XML and the DOMDocument object to parse and create XML documents.

Chapter 9: In Chapter 4 you learned how design patterns can simplify your daily life in programming by providing you with a common approach for solving problems. One of the most used design patterns for application architecture is Model-View-Controller (MVC). In this chapter we discuss the basic structure of MVC frameworks and then introduce you to some of these popular frameworks. Frameworks play a very important role in Rapid Development of PHP applications. You will learn how to build a framework in this chapter, which will also help you to understand object loading, data abstraction layers, and the importance of separation and finally you get a closer look at how applications are done.

Who is This Book for

From beginners to intermediate users of PHP5

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

There are three styles for code. Code words in text are shown as follows: "In some cases you may need to investigate which classes are in the current scope. You can do it easily with `get_declared_classes()` function."

A block of code will be set as follows:

```
<?
class ParentClass
{
}
class ChildClass extends ParentClass
{
}
$cc = new ChildClass();
if  (is_a($cc,"ChildClass")) echo "It's a ChildClass Type Object";
echo "\n";
if  (is_a($cc,"ParentClass")) echo "It's also a ParentClass Type
Object";
?>
```

New terms and **important words** are introduced in a bold-type font. Words that you see on the screen, in menus or dialog boxes for example, appear in our text like this: " If you place the server in your web server (here `localhost`) document, root in a folder named `proxy` and then access the client, you will get the following output:

March, 28 2007 16:13:20".

Important notes appear in a box like this.

Tips and tricks appear like this.

Reader Feedback

Feedback from our readers is always welcome. Let us know what you think about this book, what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply drop an email to `feedback@packtpub.com`, making sure to mention the book title in the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on `www.packtpub.com` or email `suggest@packtpub.com`.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer Support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the Example Code for the Book

Visit `http://www.packtpub.com/files/code/2561_Code.zip`, and select this book from the list of titles to download any example code or extra resources for this book. The files available for download will then be displayed.

The downloadable files contain instructions on how to use them.

Errata

Although we have taken every care to ensure the accuracy of our contents, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in text or code—we would be grateful if you would report this to us. By doing this you can save other readers from frustration, and help to improve subsequent versions of this book. If you find any errata, report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **Submit Errata** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata are added to the list of existing errata. The existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with some aspect of the book, and we will do our best to address it.

This book is dedicated to
my Son
Afif – The Little Einstein

1
OOP vs. Procedural Programming

PHP is one of the most popular scripting languages of the last couple of years. Almost 60% of web servers are running on Apache with PHP. It is so popular that millions of websites and web applications are developed every month using PHP. PHP started its journey as a simple replacement for Perl, and in a few years it became tremendously popular and powerful. The language itself is closely similar to ANSI C.

One of the reasons why PHP became so popular is its short learning curve. Learning PHP is not a big job, especially if you are familiar with the syntax of Java or C. As writing PHP scripts is easy, anyone can write PHP code without following conventions and mixing presentation layers with business logics (which is one of the main reasons why there are large amounts of unmanageable projects floating around). Because there are no strict coding conventions followed in PHP, over the years as a project gets bigger, it can turn into an unmanageable demon.

OOP or **Object Oriented Programming** is a good programming practise to create manageable projects more easily. Procedural programming means writing code without objects. Procedural programming consists of codes with or without routines. OOP enlightens any language for better coding, for best performance and for writing very big projects without worrying a lot about managing them. OOP gives you facilities to create reusable objects that you or other developers can use in their projects without reinventing them again and again. OOP removes the hassles and difficulties of writing and managing big applications.

In this book we are going to discuss how you can achieve maximum benefits using OOP with PHP, using step-by-step instructions, real life examples how OOP helps you to write effective code, how to improve your coding style, and how to reuse them over time. This book won't work as a reference for PHP language; we will just cover OOP features of PHP and not the basics of general PHP. If you are looking for a good reference book, consult the PHP manual at first and then you can study *Core PHP Programming*, a very good book written by Leon Atkinson.

Introduction to PHP

This section is not for you if you are already a PHP developer, but for those who are new to PHP and starting with this book. Though I said at the very beginning that I assume you will have some pre development experience in PHP while reading this book, but if you are a total fresher and want to learn OOP with this book, this section may be worth recalling the basic PHP language features. If you are already familiar enough, don't skip this section as we have other topics to discuss here.

So you may ask where is the introduction to PHP, I am not seeing any code here! Well, you don't need to. The best resource on the internet is for free. Please go to `http://www.php.net` and download the manual and read the basic chapters. For a detailed learning of PHP, you can study the book *Learning PHP5* written by David Sklar.

Ready, Set, Go

In this book, we are using PHP5.1.2 for our examples but for almost 99% of cases it will run with PHP version 5x. We have MySQL 5 in our machine and Apache 2 as our web server. If you aren't familiar with configuring all these in your machine, you can download pre configured WAMP or LAMP distributions like XAMPP (`http://apachefriends.org`) or Apache2Triad (`http://www.apache2triad.net`). You will find corresponding documentation for installation and customization on each of these product's website.

A Little History of OOP in PHP

When PHP was developed, it did not implement OO features in itself. After PHP/FI, when Zeev, Rasmus, and Andy rewrote the core and released PHP3, very basic OO features were introduced. When PHP4 was released, OO features got matured with huge performance improvement. But the PHP team rewrote the core engine again to introduce completely new object models and released PHP5. Now there are two versions of PHP being developed. Don't get confused by comparing PHP versions with other languages. PHP5 doesn't mean it is the latest PHP version. As I said a while ago, PHP4 and PHP5 are being released actively (though there will be no more releases of PHP4 after December 2007). Between these two, PHP5 implements almost complete OO features while PHP4 doesn't. At the time of writing this book the latest version of these two streams are PHP5.2 and PHP4.4.

Procedural vs. OO Coding Style

PHP allows you to write code in two flavours, one is procedural and the other is object oriented. You can even write procedural code in PHP5 and it will run without any problems. If you are not clear about procedural and object oriented programming, then we will have a look at these two different coding styles. The following two examples are not fully running examples rather a pseudo code:

```
<?
$user_input = $_POST[‘field‘];
$filtered_content = filter($user_input); //user input filtering
mysql_connect("dbhost","dbuser","dbpassword"); //database
mysql_select_db("dbname");
$sql = "some query";
$result = mysql_query($sql);
while ($data = mysql_fetch_assoc())
{
  process ($data);
}
process_user_input($filtered_content);
?>
```

You will notice using a lot of inline processing either directly or via using functions. It may stand as an example of typical procedural operation. Let's see how it looks after converting it to OOP:

```
<?
$input_filter = new filter();
$input_filter->filter_user_input(); //filter the user inputs
$db = new dal("mysql"); //data access layer
$db->connect($dbconfig);//we wre using mysql
$result = $db->execute($sql);
ReportGenerator::makereport($result); //process data
$model = new Postmodel($filter->get_filtered_content());
$model->insert();
?>
```

Now if you take a look into these two code snippets, you will find that the latter one is much more readable. Well, you can make the first one more readable by introducing some more functions into it, but how many functions are you ready to search into when you use them? The latter snippet is better organized because you know which object is handling which process. If you write big applications in procedural style, it will be almost impossible to manage after a few versions. Of course you can implement strict coding conventions, but it is agreed by millions of developers that it won't give you the ultimate manageability and usability if it's procedural unless you do it in OO style. Almost all big applications are written using the object oriented approach.

Benefits of OOP

OOP is invented to make the developer's life easier. Using OOP you can split your problems into smaller problems that are comparatively easy to comprehend. The main goal of OOP is: everything you want to do, do it via objects. Objects are basically small discrete pieces of code which, can incorporate data and behaviors together. In an application all these objects are connected to each other, they share data among them and solve problems.

OOP can be considered better from many aspects, especially when you consider the development time and maintenance overhead. The main benefits of OOP can be considered as follows:

- **Reusability**: An object is an entity which has bundles of properties and methods and can interact with other objects. An object can be sufficient or it may have dependencies over other objects. But an object is usually developed to solve a specific set of problems. So when other developers suffer from the same set of problems, they can just incorporate your class to their project and use it without affecting their existing workflow. It prevents from DRY, which means *Don't Repeat Yourself*. In functional or modular programming, reusing is possible but complex.
- **Refactoring**: When you need to refactor your projects, OOP gives you the maximum benefit because all objects are small entities and contain its properties and methods as a part of itself. So refactoring is comparatively easier.
- **Extensible**: If you need to add features to your project, you can achieve best results from OOP. One of the core OOP features is extensibility. You can refactor your object to add the feature. While doing it, you can still maintain backward compatibility of this object so that it works fine with an old code base. Or you can extend the object and create a totally new object that retains all the necessary properties and methods of the parent object from which it has been derived, and then expose new features. This is termed "inheritance" and is a very important feature of OOP.
- **Maintenance**: Object oriented code is easier to maintain because it follows somewhat strict coding conventions and is written in a self explanatory format. For example, when a developer extends it, refactors it, or debugs it, they can easily find out the inner coding structure and maintain the code time after time. Moreover, whenever there is a team development environment in your project, OOP could be the best solution because you can distribute your code after splitting it into small parts. These small parts could be developed as a separate object, so developers can develop them almost independently. Finally, it will be very easy to merge the code.

- **Efficiency**: The concept of object oriented programming is actually developed for better efficiency and ease of development process. Several design patterns are developed to create better and efficient code. Moreover in OOP, you can think of your solution in a much better approach than procedural programming. Because you first split your problem into a small set of problems and then find solutions for each of them, the big problem is solved automatically.

Dissection of an Object

So what is an object? Well, it's nothing but a piece of code with a bunch of properties and methods. So is it similar to an array, as arrays can store data identified by properties (well, they are called keys)? Objects are much more than arrays because they contain some methods inside them. They can either hide them or expose them, which are not possible in arrays. The object is somewhat comparable with a data structure, data structure, and can incorporate a lot of other objects in itself and either creates a tight coupling among them or a loose one. And object can incorporate a lot of other object in itself and either creates a tight coupling among them or a loose one. We will learn more about loose coupling and tight coupling later in this book and understand how they will be useful for us.

Let's see the code of an object in PHP. The following object is a very simple object which can send email to a bunch of users. In PHP5, objects are a lot more different than an object in PHP4. We will not discuss the details of it, this is just an introductory object to see how the objects are written in PHP.

```
<?
//class.emailer.php
class emailer
{
  private $sender;
  private $recipients;
  private $subject;
  private $body;
  function __construct($sender)
  {
    $this->sender = $sender;
    $this->recipients = array();
  }

  public function addRecipients($recipient)
  {
    array_push($this->recipients, $recipient);
  }
```

```
    public function setSubject($subject)
    {
      $this->subject = $subject;
    }
    public function setBody($body)
    {
      $this->body = $body;
    }
    public function sendEmail()
    {
      foreach ($this->recipients as $recipient)
      {
        $result = mail($recipient, $this->subject, $this->body,
                                  "From: {$this->sender}\r\n");
        if ($result) echo "Mail successfully sent to
                                            {$recipient}<br/>";
      }
    }
}
?>
```

The above object contains four private properties and three accessor methods and finally one more method to dispose the email to recipients. So how we are going to use it in our PHP code? Let's see below:

```
<?
$emailer = new emailer("hasin@pageflakes.com"); //construcion
$emailer->addRecipients("hasin@somewherein.net"); //accessing methods
// and passing some data
$emailer->setSubject("Just a Test");
$emailer->setBody("Hi Hasin, How are you?");
$emailer->sendEmail();
?>
```

I am sure that the above code snippet is much more self explanatory and readable. If you follow proper conventions, you can make your code easy to manage and maintain. Wordpress developers use a motto on their site `www.wordpress.org` which is "Coding is poetry". Coding is exactly a poem; if you just know how to write it.

Difference of OOP in PHP4 and PHP5

Objects in PHP5 differ a lot from objects in PHP4. OOP became matured enough in true sense from PHP5. OOP was introduced since PHP3 but that was just an illusion for real object oriented programming. In PHP4 you can create objects but you can't feel the real flavour of an object there. In PHP4 it was almost a poor object model.

One of the main differences of OOP in PHP4 is that everything is open; no restrictions about the usage of methods or properties. You can't use public, private, and protected modifiers for your methods. In PHP4 developers usually declare private methods with a double underscore. But it doesn't mean that declaring a method in that format actually prevents you from accessing that method outside the class. It's just a discipline followed.

In PHP4 you can find interfaces but no abstract or final keyword. An interface is a piece of code that any object can implement and that means the object must have all the methods declared in the interface. It strictly checks that you must implement all the functions in it. In the interface you can only declare the name and the access type of any method. An abstract class is where some methods may have some body too. Then any object can extend that abstract class and extend all these methods defined in that abstract class. A final class is an object which you are not allowed to extend. In PHP5 you can use all of these.

In PHP4 there are no multiple inheritances for interfaces. That means an interface can extend only one interface. But in PHP5 multiple inheritance is supported via implementing multiple interfaces together.

In PHP4, almost everything is static. That means if you declare any method in the class, you can call it directly without creating an instance of it. For example the following piece of code is valid in PHP4:

```
<?
class Abc
{
  var $ab;
  function abc()
  {
    $this->ab = 7;
  }
  function echosomething()
  {
    echo $this->ab;
  }
}
echo abc::echosomething();
?>
```