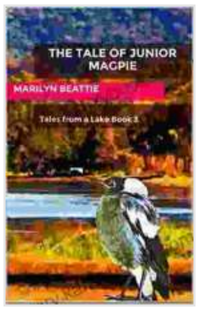
PHP Object Oriented Programming With Oracle: The Ultimate Guide

Are you ready to take your PHP skills to the next level? If so, then you need to learn object oriented programming (OOP). OOP is a powerful programming paradigm that can help you write more efficient, maintainable, and reusable code. And when it comes to OOP, there's no better language to use than PHP.



Web Programming for Business: PHP Object-Oriented Programming with Oracle by Marilyn Beattie

★★★★☆ 4.8 out of 5

Language	: English
File size	: 10696 KB
Screen Reader	: Supported
Print length	: 45 pages

FREE DOWNLOAD E-BOOK

In this comprehensive guide, we'll teach you everything you need to know about PHP OOP. We'll start with the basics, such as classes, objects, and methods. Then, we'll move on to more advanced topics, such as inheritance, polymorphism, and encapsulation.

By the end of this guide, you'll be able to write PHP OOP code like a pro. You'll be able to create complex applications that are both efficient and easy to maintain. So what are you waiting for? Let's get started!

What is OOP?

OOP is a programming paradigm that emphasizes the use of objects and classes. Objects are data structures that contain data and methods. Classes are blueprints that define the structure and behavior of objects.

OOP has many advantages over other programming paradigms. OOP code is more efficient, maintainable, and reusable. OOP also makes it easier to create complex applications.

Why use PHP for OOP?

PHP is a great language for OOP. It has a rich set of features that make it easy to write OOP code. PHP also has a large community of developers who can help you learn OOP and solve problems.

Getting started with OOP in PHP

To get started with OOP in PHP, you need to create a class. A class is a blueprint that defines the structure and behavior of objects.

Here is an example of a simple PHP class:

php class Person { public $name; public $age;

public function __construct($name, $age){$this->name = $name; $this->age = $age; }

public function greet(){echo "Hello, my name is {$this->name}and I am {$this->age}years old."; }}

This class defines two properties, `name` and `age`. It also defines a constructor method, which is called when a new object is created. The constructor method initializes the `name` and `age` properties.

It also defines a method called `greet()`, which prints a greeting message.

To create an object, you use the `new` keyword. For example, the following code creates a new `Person` object:

php $person = new Person('John Doe', 30);

Once you have created an object, you can access its properties and methods using the dot operator. For example, the following code accesses the `name` property of the `$person` object:

php echo $person->name;

The following code calls the `greet()` method of the `$person` object:

php $person->greet();

Inheritance

Inheritance is a powerful feature of OOP that allows you to create new classes from existing classes. The new class, called the child class, inherits the properties and methods of the existing class, called the parent class.

Here is an example of how to create a child class in PHP:

php class Student extends Person { public $gpa;

public function __construct($name, $age, $gpa) {parent::__construct($name, $age); $this->gpa = $gpa; }

public function getGpa(){return $this->gpa; }}

The `Student` class inherits the `name` and `age` properties from the `Person` class. It also defines a new property, `gpa`.

The `Student` class also defines a new constructor method, which initializes the `gpa` property.

It also defines a new method, `getGpa()`, which returns the `gpa` property.

To create a `Student` object, you use the `new` keyword. For example, the following code creates a new `Student` object:

php $student = new Student('John Doe', 30, 3.5);

Once you have created a `Student` object, you can access its properties and methods using the dot operator. For example, the following code accesses the `name` property of the `$student` object:

php echo $student->name;

The following code calls the `getGpa()` method of the `$student` object:

php echo $student->getGpa();

Polymorphism

Polymorphism is a powerful feature of OOP that allows you to write code that can work with different types of objects. Polymorphism is achieved through method overriding.

Method overriding allows you to define a method in a child class that has the same name and signature as a method in the parent class. When you call the method on a child class object, the child class method will be executed.

Here is an example of how to override a method in PHP:

```
php class Person { public function greet(){echo "Hello, my name is {$this->name}."; }}
```

```
class Student extends Person { public function greet(){parent::greet(); echo " I am a student."; }}
```

The `Student` class overrides the `greet()` method of the `Person` class. When you call the `greet()` method on a `Student` object, the `Student` class method will be executed.

Encapsulation

Encapsulation is a powerful feature of OOP that allows you to hide data and methods from other parts of your program. Encapsulation is achieved through access modifiers.

Access modifiers control who can access data and methods. There are three access modifiers in PHP: public, protected, and private.

- Public properties and methods can be accessed by anyone.
- Protected properties and methods can be accessed by the class itself and its subclasses.

- Private properties and methods can only be accessed by the class itself.

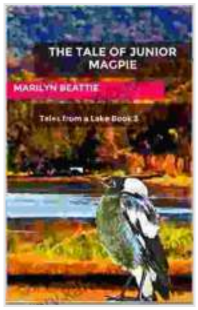
Here is an example of how to use access modifiers in PHP:

```
php class Person { private $name; protected $age; public $gender;

public function __construct($name, $age, $gender){$this->name = $name;
$this->age = $age; $this->gender = $gender; }

public function getName(){return $this->name; }

protected function getAge(){return $this->age; }

private function getGender(){return $this->gender; }}
```

The `name` property is public, which means that it can be accessed by anyone. The `age` property is protected, which means that it can be accessed by the `Person` class itself and its subclasses. The `gender` property is private, which means that it can only be accessed by the `Person` class itself.

OOP is a powerful programming paradigm that can help you write more efficient, maintainable, and reusable code. PHP is a great language for OOP, and this guide has taught you everything you need to know to get started with OOP in PHP.

So what are you waiting for? Start writing OOP code today!

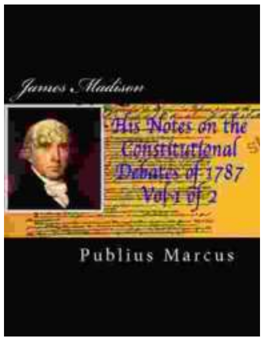


Web Programming for Business: PHP Object-Oriented Programming with Oracle by Marilyn Beattie

★★★★☆ 4.8 out of 5

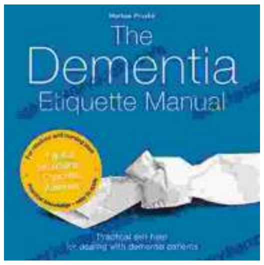
Language	: English
File size	: 10696 KB
Screen Reader	: Supported
Print length	: 45 pages

FREE DOWNLOAD E-BOOK



James Madison: His Notes on the Constitutional Debates of 1787, Vol. I

James Madison's Notes on the Constitutional Debates of 1787 are a vital source for understanding the creation of the United States Constitution. This...



The Dementia Etiquette Manual: A Comprehensive Guide to Understanding and Caring for Persons with Dementia

If you're like most people, you probably don't know much about dementia. That's understandable. Dementia is a complex and challenging condition that affects...