Unit 4 Module 3: Object Oriented PHP

Dylan Lane McDonald

CNM STEMulus Center
Web Development with PHP

September 11, 2014

Outline

Introduction to Object Oriented Programming

Definition

Object Oriented Programming *is a paradigm where all the actors in a program are modeled first and relations and behaviors are then implemented.*

Introduction to Object Oriented Programming

Definition

Object Oriented Programming *is a paradigm where all the actors in a program are modeled first and relations and behaviors are then implemented.*

The fundamental shift from the imperative paradigm to the object oriented paradigm is in what you conceptualize first: namely, the actors. By contrast, the imperative paradigm conceptualizes the procedures and steps before conceptualizing the actors.

What is an Object?

Definition

An **object** *is the nucleus of object oriented programming. Every object has a state and behavior. The state of an object is the data contained in the object. The behavior of the object are the methods that contain code that manipulate and use the object.*

What is an Object?

Definition

An **object** *is the nucleus of object oriented programming. Every object has a state and behavior. The state of an object is the data contained in the object. The behavior of the object are the methods that contain code that manipulate and use the object.*

In creating an object, one starts by first determining the state of the object; thus answering the question "what is this object for?" and then the behavior, which addresses the question "what does this object do?" This is the typical thought process when writing object oriented programs.

Encapsulation

In object oriented programming, direct access to the object's state is problematic because it subverts the behavior. Therefore, the object's state is altered as a by product of its behavior.

Encapsulation

In object oriented programming, direct access to the object's state is problematic because it subverts the behavior. Therefore, the object's state is altered as a by product of its behavior.

Definition

Encapsulation *is the prohibition of directly changing the object's state, and instead restricting state changes to methods defined by its behavior.*

Encapsulation

In object oriented programming, direct access to the object's state is problematic because it subverts the behavior. Therefore, the object's state is altered as a by product of its behavior.

Definition

Encapsulation *is the prohibition of directly changing the object's state, and instead restricting state changes to methods defined by its behavior.*

The key way encapsulation works is by setting access levels to each member and method:

- **public**: everyone can access (default)
- **protected**: only this object and child classes can access
- **private**: only this object can access

The general strategy is to make the object's state private and methods public.

Classes vs Objects

As a developer, one writes a **class**. The end users see the results of the class, the **object**.

Definition

A **class** *is a large code block that defines an object's state and behavior.*

Classes vs Objects

As a developer, one writes a **class**. The end users see the results of the class, the **object**.

Definition

A **class** *is a large code block that defines an object's state and behavior.*

A useful metaphor would be to think as the class as a blueprint for how to make an object. The object is the tangible item **constructed** (or **instantiated**) using the class's schematic. This is a subtle distinction even experienced developers sometimes stumble on.

Writing Our First Class

First, one defines the class's state and behavior. In this example, take a simple car class:

1. **Model**: Any alpha numeric string
2. **Cylinders**: Any positive, even integer

Here, we have defined two items for the class's state and how we expect the state to behave.

```
class Car {
    private $model;
    private $cylinders;
}
```

Listing 1: The Car's State

Accessor Methods

Encapsulation does not directly allow one to read from the object's state. Therefore, we write **accessor methods**.

```
class Car {
    public function getCylinders() {
        return($this->cylinders);
    }
    public function getModel() {
        return($this->model);
    }
}
```

Listing 2: The Car's Accessor Methods

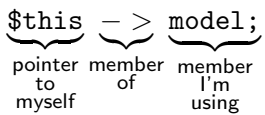
Have Some of `$this`!

In the previous slide, we saw a special keyword: `this`. The `this` keyword is a pointer to itself. The `this` keyword allows PHP to distinguish between a local variable in the function or a variable that is part of the object's state.

Have Some of `$this`!

In the previous slide, we saw a special keyword: `this`. The `this` keyword is a pointer to itself. The `this` keyword allows PHP to distinguish between a local variable in the function or a variable that is part of the object's state.

The other operator we just saw is the -> operator. This is actually pronounced the "arrow operator". The arrow operator signifies that the right hand side is a member of the class defined on the left hand side. This is the same as the . operator in JavaScript.



Mutator Methods

```
class Car {
    public function setCylinders($newCylinders) {
        if($newCylinders <= 0 || $newCylinders % 2 != 0) {
            throw(new Exception("Cylinders must be even"));
        }
        $this->cylinders = $newCylinders;
    }
}
```

Listing 3: The Car's Mutator Methods

Mutator Functions

```
class Car {
    public function setModel($newModel) {
        $this->model = $newModel;
    }
}
```

Listing 4: The Car's Mutator Methods (cont)

Notice the syntax for throwing an exception. The `Exception` class is a system defined class in PHP and is ready-made for throwing a generic exception. The exception class in PHP is much more powerful to the throwing of Strings we did in JavaScript, and is much more akin to Java's `Exception` class.

Constructors & Destructors

When an object is instantiated, the **constructor**, a method which sets the object's state, is executed. Conversely, when the object goes out of scope, the **destructor** is executed.

```
class Car {
    public function __construct($newModel, $newCylinders) {
        $this->setModel($newModel);
        $this->setCylinders($newCylinders);
    }
}
```

Listing 5: The Car's Constructor

Creating An Object

After the class is defined, it is ready to be created and used in your program. Listing 6 shows a simple example of how to use the class.

```
// the constructor is executed here
$honda = new Car("Honda", 6);
// change the object's state
$honda->setCylinders(8);
// access the object's state
echo $honda->getCylinders() . "<br />";
```

Listing 6: Creating a Car Object

Note the use of the variable name `$honda` at the beginning every time the object is used. This is required to properly reference the object.

Model-View-Controller

Definition

A **design pattern** *is a reusable and accepted solution to a particular software engineering problem. Design patterns are not directly translatable into code. Instead, they provide a template that can be followed while implementing a particular solution.*

Model-View-Controller

Definition

A **design pattern** *is a reusable and accepted solution to a particular software engineering problem. Design patterns are not directly translatable into code. Instead, they provide a template that can be followed while implementing a particular solution.*

Definition

Model-View-Controller *is a design pattern for user interfaces. It clearly delineates the front and back end of a software solution by putting the front end in the view and the back end in the model & controller.*

Model-View-Controller

The three components of Model-View-Controller:

1. **Model**: the data being represented. This is typically implemented as an object representing a database row.
2. **View**: the screen the user sees. This usually is HTML output of the model.
3. **Contoller**: the mechanism that allows for the manipulation of the model. This is usually input form that allows the user to modify or add to the model.

Model-View-Controller is a commonly deployed design pattern in web programming. In fact, is is the entire basis for the Ruby on Rails[1] framework.

[1] As well as Groovy on Grails

Model-View-Controller

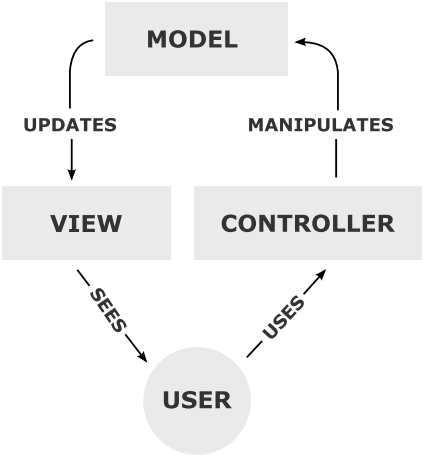


Figure 1: Model-View-Controller