OBJECT-ORIENTED PROGRAMMING IN PRACTICE

Oscar Merida, @omerida

November 3, 2018

GOOD MORNING!

SLIDES

https://phpa.me/oop-in-practice-mad18

PROCEDURAL CODE

EVENTS

We have a simple application to list upcoming events and track RSVPs from users.

LIST.PHP

```
$dbh = new PDO('sqlite:../invites.db');

$stmt = $dbh->prepare("SELECT * FROM events");
$stmt->execute();
?>
<html>
<body>
<h1>Events</h1>
```

LIST.PHP (CONTINUED)

```
<?php
while ($event = $stmt->fetch(PDO::FETCH_ASSOC)):
?>
   <h2><?= htmlentities($event['title']) ?></h2>
<?php
   $event_date = new \DateTime($event['event_date']);
   switch ($event['event_type']) :
      case 'all_day':
?>
         <p>This is an all day event on <?= $event_date->format("F j, Y") ?>
<?php
         break;
      default:
?>
          <p>This event starts at <?= $event_date->format('F j, Y H:i a') ?></p>
<?php
         break;
   endswitch;
?>
   <p><a href="./rsvp.php?event=<?= $event['uid'] ?>">RSVP</a></p>
   <hr />
<?php
endwhile;
?>
</body>
</html>
```

RSVP.PHP

```php
$dbh = new PDO('sqlite:../invites.db');

$event_id = $_GET['event'] ?? null;

if (empty($event_id)) {
    trigger_error("Invite ID missing", E_USER_ERROR);
}

// fetch the invite as an arrow from the database
$stmt = $dbh->prepare("SELECT * FROM events WHERE uid=?");
if ($result = $stmt->execute([$event_id])) {
    $invite = $stmt->fetch(PDO::FETCH_ASSOC);
}
```

RSVP.PHP (CONTINUED)

```php
if (!empty($_POST)) {
    $stmt = $dbh->prepare(
        "REPLACE INTO replies (event_id, email, reply, updated)
         VALUES (?, ?, ?, ?)");
    $now = new \DateTime();
    $result = $stmt->execute([
        $event_id,
        filter_var($_POST['email'], FILTER_SANITIZE_EMAIL),
        $_POST['response'],
        $now->format('Y-m-d H:i:s'),
    ]);

    $msg = "Your response was saved";
}
```

RSVP.PHP (CONTINUED)

```
// display form
?>
<html>
<body>
<h1><?= htmlentities($invite['title']) ?></h1>
<?php
$event_date = new \DateTime($invite['event_date']);

switch ($invite['event_type']) :
   case 'all day':
?>
      <p>This is an all day event on <?= $event_date->format("F j, Y") ?>
<?php
      break;
   default:
?>
      <p>This event starts at <?= $event_date->format('F j, Y H:i a') ?></p>
<?php
         break;
endswitch;
?>
```

RSVP.PHP (CONTINUED)

```
<?php if (isset($msg)): ?>
   <p><?php echo $msg ?></p>
<?php endif; ?>

<form method="POST" action="">
   <label>Your Email <input type="email" name="email" value="" required="required"/>
  </label>
   <p><b>Will You Attend?</b></p>
   <div>
   <label><input type="radio" name="response" value="yes">Yes</label>
   <label><input type="radio" name="response" value="no">No</label>
   <label><input type="radio" name="response" value="maybe">Maybe</label>
   </div>
   <input type="submit" name="submit"value="Save Your Reply" />
</form>
</body>

</html>
```

YES IT WORKS...

1. Many actions in one place
2. Depending on shared global state

AUTOLOADING

SPECIFY PATHS

```
    "autoload": {
        "psr-4": {
            "Phparch\\": "src/Phparch",
        }
    }
```

CREATE THE AUTOLOADER

Then

```
$ composer dumpautoload
```

And in PHP files

```
require ('vendor/autoload.php');
```

REFACTOR 1

EVENTS

- Let’s make an Events class
- … or do we need more than one?

WHAT DEFINES EVENTS?

WHAT DEFINES EVENTS?

- All events have a title, unique id for RSVPs, an `id` field in the database

WHAT DEFINES EVENTS?

- All events have a title, unique id for RSVPs, an `id` field in the database
- We have different types of events

WHAT DEFINES EVENTS?

- All events have a title, unique id for RSVPs, an `id` field in the database
- We have different types of events
  - All day events have no start time

WHAT DEFINES EVENTS?

- All events have a title, unique id for RSVPs, an `id` field in the database
- We have different types of events
  - All day events have no start time
  - A “Timed” event with a specific start time

WHAT DEFINES EVENTS?

- All events have a title, unique id for RSVPs, an `id` field in the database
- We have different types of events
  - All day events have no start time
  - A “Timed” event with a specific start time
  - Could add new types in the future

WHAT DEFINES EVENTS?

- All events have a title, unique id for RSVPs, an `id` field in the database
- We have different types of events
  - All day events have no start time
  - A “Timed” event with a specific start time
  - Could add new types in the future
- Given any event, we want to generate

WHAT DEFINES EVENTS?

- All events have a title, unique id for RSVPs, an `id` field in the database
- We have different types of events
  - All day events have no start time
  - A “Timed” event with a specific start time
  - Could add new types in the future
- Given any event, we want to generate
  - An event date

WHAT DEFINES EVENTS?

- All events have a title, unique id for RSVPs, an `id` field in the database
- We have different types of events
  - All day events have no start time
  - A “Timed” event with a specific start time
  - Could add new types in the future
- Given any event, we want to generate
  - An event date
  - A description of the event

AN EVENT CLASS

```php
namespace Phparch;

class Event {
   protected $title;
   protected $date;
   protected $type;
   protected $uid;

   public function __construct(?int $id, string $title, string $date,
                               string $type, string $uid) {
      $this->id = $id;
      $this->title = $title;
      $this->date = $date;
      $this->type = $type;
      $this->uid = $uid;
   }

   public function getDescription() : string {
      $date = new \DateTime($this->date);
      switch ($this->type) {
         case 'all_day':
            return 'This is an all day event on ' . $date->format("F j, Y");
         case 'timed':
            return 'This event starts at ' . $date->format('F j, Y H:i a');
      }
   }
```

```
    // getters follow
}
```

LIST.PHP

```
<?php
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)):
    $event = new \Phparch\Event($row['id'], $row['title'],
        $row['event_date'], $row['event_type'], $row['uid'])
?>
    <h2><?= htmlentities($event->getTitle()) ?></h2>
    <p><?= $event->getDescription() ?></p>
    <p><a href="./rsvp.php?event=<?= $event->getUid() ?>">RSVP</a></p>
    <hr />
<?php
endwhile;
```

PROGRESS

- Typehints and completion in IDE
- Started to specify what kind of inputs we expect

BUT

- Our class has to keep track of event types

R2: EVENT TYPES

ABSTRACT CLASSES

- We have different event types
- We have shared functionality

A BASE CLASS

```php
<?php
namespace Phparch;

abstract class Event
{
    // protected properties here

    public function __construct(?int $id, string $title, string $date,
                                string $type, string $uid) {
        $this->id = $id;
        $this->title = $title;
        $this->date = $date;
        $this->type = $type;
        $this->uid = $uid;
    }

    abstract function getDescription() : string;

    public function formatDate(string $string) {
        $date = new \DateTime($this->date);
        return $date->format($string);
    }

    // getter methods follow
```

TYPE SPECIFIC CLASSES

```php
<?php
namespace Phparch\Event;

class AllDay extends \Phparch\Event
{
    public function getDescription() : string {
        return 'This is an all day event on '
               . $this->formatDate("F j, Y");
    }
}
```

TYPE SPECIFIC CLASSES (CONTINUED)

```php
<?php
namespace Phparch\Events;

class Timed extends \Phparch\Event
{
    public function getDescription() : string {
        return 'This event starts at '
              . $this->formatDate('F j, Y H:i');
    }
}
```

OOOPS

```
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)):
    switch ($row['event_type']) {
        case 'all_day':
            $event = new \Phparch\Event\AllDay(
                $row['id'], $row['title'],
                $row['event_date'], $row['event_type'], $row['uid']
            );
        case 'timed':
            $event = new \Phparch\Event\Timed(
                $row['id'], $row['title'],
                $row['event_date'], $row['event_type'], $row['uid']
            );
    }
?>
    <h2><?= htmlentities($event->getTitle()) ?></h2>
    <p><?= $event->getDescription() ?></p>
    <p><a href="./rsvp.php?event=<?= $event->getUid() ?>">RSVP</a></p>
    <hr />
<?php
endwhile;
```

R3: FACTORIES

WHY FACTORIES?

- Allow us to change, rename, replace the class later in one place
- If construction is complicated, do all the work in one place (DRY)

A FACTORY CLASS

```php
<?php
namespace Phparch\Events;

class Factory
{
    private function __construct()
    {
    }

    public static function fromArray(array $row) : \Phparch\Event
    {
        switch ($row['event_type']) {
            case 'all_day':
                return new AllDay(
                    $row['id'], $row['title'],
                    $row['event_date'], $row['event_type'], $row['uid']
                );
            case 'timed':
                return new Timed(
                    $row['id'], $row['title'],
                    $row['event_date'], $row['event_type'], $row['uid']
                );
        }
    }
}
```

CLEANS UP LIST.PHP

```
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)):
    $event = \Phparch\Events\Factory::fromArray($row);
?>
    <h2><?= htmlentities($event->getTitle()) ?></h2>
    <p><?= $event->getDescription() ?></p>
    <p><a href="./rsvp.php?event=<?= $event->getUid() ?>">RSVP</a></p>
    <hr />
<?php
endwhile;
```

NEW REQUIREMENTS

What if you're tasked with adding a new kind of event?

1. Online Events must have a URL.
2. The URL must display as part of the description.

WEBINAR CLASS

```php
namespace Phparch\Events;

class Webinar extends \Phparch\Event
{
   protected $link;

   public function __construct(?int $id, string $title, string $date,
                               string $type, string $uid, string $link) {

       parent::__construct($id, $title, $date, $type, $uid);

       $link = filter_var($link, FILTER_VALIDATE_URL);
       if (empty($link)) {
          throw new \UnexpectedValueException("Invalid URL");
       }
       $this->link = $link;
   }

   public function getDescription() : string {
      return 'This webinar will begin ' . $this->formatDate("F j, Y ")
          . ' at ' . $this->formatDate("h:i a") . '.'
          . ' <a href="' . $this->link .'">Join the Webinar</a>';
   }
}
```

UPDATED FACTORY

```
public static function fromArray(array $row) : \Phparch\Event
{
    switch ($row['event_type']) {
        case 'all_day':
            return new AllDay(
                $row['id'], $row['title'], $row['event_date'],
                $row['event_type'], $row['uid']
            );
        case 'timed':
            return new Timed(
                $row['id'], $row['title'], $row['event_date'],
                $row['event_type'], $row['uid']
            );
        case 'webinar':
            return new Webinar(
                $row['id'], $row['title'], $row['event_date'],
                $row['event_type'], $row['uid'], $row['link']
            );
    }
}
```

R4: DEPENDENCIES

DEPENDENCY INJECTION

Dependencies are other objects which perform a service for you.

- Retrieve rows from a database
- Write to a log file
- Send emails
- Render HTML, XML, Json, …

NOT DIC!

Dependency Injection Container !== Dependency Injection

WHAT'S OUR DEPENDENCY?

The Database.

```
$dbh = new PDO('sqlite:../invites.db');
$stmt = $dbh->prepare("SELECT * FROM events LIMIT 2");
$stmt->execute();
```

REPOSITORY PATTERN

AN EVENT REPOSITORY

```php
namespace Phparch\Events;

class Repository
{
    protected $dbh;

    public function __construct(\PDO $dbh) {
        $this->dbh = $dbh;
    }

    public function getAll() {
        $stmt = $this->dbh->prepare("SELECT * FROM events");
        $stmt->execute();

        while ($row = $stmt->fetch(\PDO::FETCH_ASSOC)) {
            $event = Factory::fromArray($row);
            yield $event; // generators!
        }
    }
```

SIMPLIFIES LIST.PHP

```php
$EventRepository = new Phparch\Events\Repository($dbh);
foreach ($EventRepository->getAll() as $event):
?>
    <h2><?= htmlentities($event->getTitle()) ?></h2>
    <p><?= $event->getDescription() ?></p>
    <p><a href="./rsvp.php?event=<?= $event->getUid() ?>">RSVP</a></p>
    <hr />
<?php
endforeach;
```

REPLY CLASS HANDLES VALIDATION

```php
namespace Phparch\Replies;
use \DateTime;

class Reply
{
    protected $event_id;
    protected $email;
    protected $reply;
    protected $updated;

    public function __construct(string $event_id, string $email,
                                string $reply, ?DateTime $updated) {
        $this->event_id = $event_id;
        if (false == filter_var($email,FILTER_VALIDATE_EMAIL)) {
            throw new \UnexpectedValueException("Invalid email argument");
        }
        $this->email = $email;

        if (!in_array(strtolower($reply), ['yes', 'no', 'maybe'])) {
            throw new \UnexpectedValueException("Invalid reply state.");
        }
        $this->reply = strtolower($reply);
        $this->updated = $updated ?? new DateTime("now");
    }
```

REPOSITORY CLASS HANDLES SAVES

```php
namespace Phparch\Replies;

class Repository
{
    protected $dbh;

    public function __construct(\PDO $dbh) {
        $this->dbh = $dbh;
    }

    public function save(Reply $Reply) {
        $stmt = $this->dbh->prepare(
            "REPLACE INTO replies (event_id, email, reply, updated)
            VALUES (?, ?, ?, ?)");
        $stmt->execute([
            $Reply->getEventId(),
            $Reply->getEmail(),
            $Reply->getReply(),
            $Reply->getUpdated()->format('Y-m-d H:i:s')
        ]);
    }
}
```

SIMPLIFIES RSVP.PHP

```php
$EventRepository = new Phparch\Events\Repository($dbh);
$event = $EventRepository->findByUid($event_uid);

if (!empty($_POST)) {
    try {
        $ReplyRepository = new Phparch\Replies\Repository($dbh);
        $Reply = new Phparch\Replies\Reply(
            $event->getId(), $_POST['email'], $_POST['response']
        );

        $ReplyRepository->update($Reply);
        $msg = "Your response was saved";
    } catch (\UnexpectedValueException $ex) {
        $msg = $ex->getMessage();
    }
}
```

WHAT HAVE WE DONE?

SO FAR...

SO FAR...

1. Used Abstract and Class Inheritance to encapsulate behaviors which varied.

SO FAR...

1. Used Abstract and Class Inheritance to encapsulate behaviors which varied.
2. Moved business logic to class methods

SO FAR...

1. Used Abstract and Class Inheritance to encapsulate behaviors which varied.
2. Moved business logic to class methods
3. Isolated actions which talk to external dependencies.

R5: INTERFACES

BUT FIRST ... LET'S GET CONTROL

```
composer require nikic/fast-route
```

INDEX.PHP

```php
$r->addRoute('GET', '/events', function() {
    $dbfile = realpath(__DIR__ . '/../invites.db');
    $dbh = new PDO('sqlite:' . $dbfile);
    $controller = new \Phparch\Controllers\Events($dbh);
    return $controller->index();
});
```

A FIRST TRY MIGHT BE

```
namespace Phparch\Controllers;

class Events {

    protected $dbh;

    public function __construct(\PDO $dbh) {
        $this->dbh = $dbh;
    }

    public function index() {
        $EventRepository = new \Phparch\Events\Repository($this->dbh);
?>
<html>
<body>
<h1>Events</h1>

<?php
foreach ($EventRepository->getAll() as $event):
?>
    <h2><?= htmlentities($event->getTitle()) ?></h2>
    <p><?= $event->getDescription() ?></p>
    <p><a href="/rsvp/<?= $event->getUid() ?>">RSVP</a></p>
    <hr />
<?php
```

```
endforeach;
?>
</body>
</html>
<?php
    }
}
```

INTERFACES

Why **couple** our controller to a Database Repository?

Interfaces let us specify how unrelated classes have to work *in the same way*.

Only specify method signatures.

COMMON LANGUAGE FOR EVENT REPOSITORIES

```php
namespace Phparch\Events;

interface RepositoryInterface
{
    /**
     * @return \Traversable
     */
    public function getAll();

    public function findByUid(string $uid): \Phparch\Event;
}
```

GREAT FOR TYPEHINTS!

```
namespace Phparch\Controllers;

use Phparch\Events\RepositoryInterface;

class Events
{
    protected $EventRepository;

    public function __construct(RepositoryInterface $EventRepository) {
        $this->EventRepository = $EventRepository;
    }

    public function index() {
?>
<html>
<body>
<h1>Events</h1>

<?php
foreach ($this->EventRepository->getAll() as $event):
```

MARSHALL DEPENDENCIES AT BOOTSTRAP

```
$r->addRoute('GET', '/events', function() {
    $dbfile = realpath(__DIR__ . '/../invites.db');
    $dbh = new PDO('sqlite:' . $dbfile);
    $EventRepository = new \Phparch\Events\Repository($dbh);
    $controller = new \Phparch\Controllers\Events($EventRepository);
    return $controller->index();
});
```

DECOUPLED CONTROLLERS

Our Controller doesn't have to know WHERE events are persisted.

```
$r->addRoute('GET', '/events', function() {
  $EventRepository = new
  \ExternalApi\Events\Repository('https://example.com/events.json');
  $controller = new \Phparch\Controllers\Events($EventRepository);
  return $controller->index();
});
```

TESTS

WHAT DID WE GAIN?

WHAT DID WE GAIN?

- Broke up a monolithic, procedural script

WHAT DID WE GAIN?

- Broke up a monolithic, procedural script
- Encapsulated single-responsibility code

WHAT DID WE GAIN?

- Broke up a monolithic, procedural script
- Encapsulated single-responsibility code
- Added flexibility in the sytem

TESTING EVENTFACTORY

```php
<?php
use PHPUnit\Framework\TestCase;

class EventFactoryTest extends TestCase
{
    public function testMakesAllDayEvent() {
        $Event = Phparch\Events\Factory::fromArray([
            'id' => 1,
            'title' => "All Day Event",
            'event_date' => '2018-12-31',
            'event_type' => 'all_day',
            'uid' => '89duT7a',
        ]);

        $this->assertInstanceOf('Phparch\Events\AllDay', $Event);
        $this->assertEquals(
            'This is an all day event on December 31, 2018',
            $Event->getDescription()
        );
    }
}
```

TESTING WEBINAR VALIDATION

```php
<?php
use PHPUnit\Framework\TestCase;

class WebinarValidationTest extends TestCase
{
    /**
     * @expectedException UnexpectedValueException
     */
    public function testInvalidUrl () {
        $Webinar = new \Phparch\Events\Webinar(
            1, "Sample Webinar", "2019-04-01 13:00",
            "webinar", 'W7cnj5e',
            "tel:+1-816-555-1212" // will fail
        );
    }
}
```

JUST SCRATCHING THE SURFACE

Unlike our initial code, we now have components we can test independently.

A robust, updated test suite is also invaluable for documenting requirements.

THANK YOU

Feedback: https://joind.in/talk/041b6

@omerida

I publish php[architect], a monthly magazine for PHP developers.

www.phparch.com