

# PHP Frameworks

## So. Many. Frameworks.

### Full stack:

- ▶ Symfony
- ▶ Zend Framework
- ▶ Laravel
- ▶ CakePHP
- ▶ CodeIgniter
- ▶ Kohana
- ▶ Yii
- ▶ Lithium

### Micro:

- ▶ Silex
- ▶ Slim
- ▶ Fat-Free Framework
- ▶ Aura
- ▶ Phalcon

And more...

# Organization Patterns

Model, View, Controller

## 4 What is MVC Design Pattern

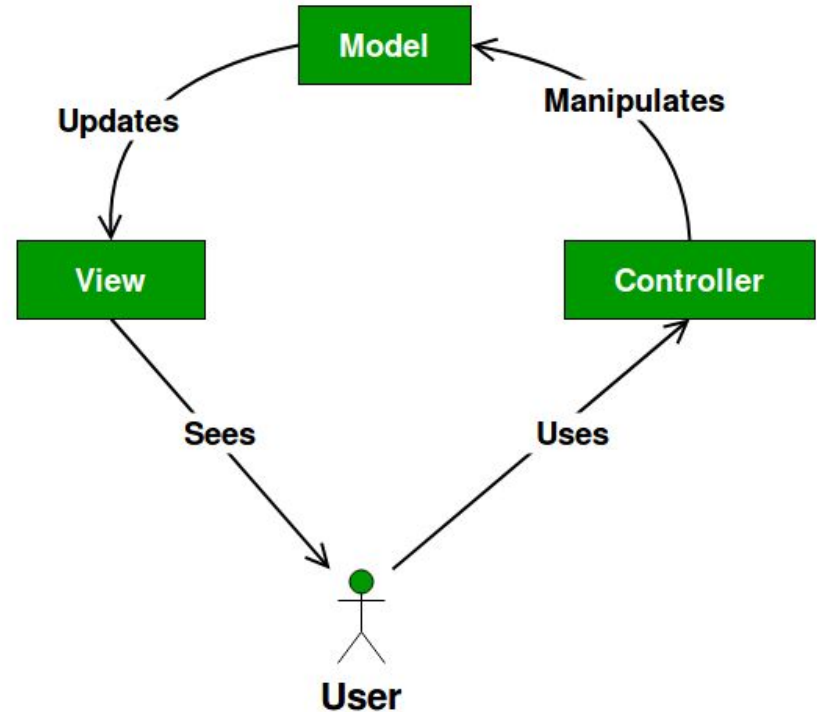
- ▶ "Design patterns" describe solution to common programming problems in general terms which can be used in any language.
- ▶ The Model-View-Controller (MVC) pattern is one of the most common design patterns.
- ▶ This pattern can be found in many popular programming language.
  - ▶ Java, C#, Ruby, and PHP all have frameworks that are used in web application development with MVC straight out of the box.

## Model-View-Controller

**Model** represents the data

**View** is the visual component that presents the interface for users to interact with that data

**Controller** is the coordinator. Responsible for coordinating what specific actions need to be performed to return the response



# Project Setup

Generic configuration

## 7 Project Structure: Skeleton

- ▷ data
  - ▶ exampleapp.db *\*sqlite database\**
- ▷ public *\*this is where you want to point the root of your webserver\**
  - ▶ index.php
- ▷ src
  - ▶ Controller
  - ▶ Model
  - ▶ View
  - ▶ app setting (config, dependencies, routes)
- ▷ tests
- ▷ project settings (composer, phpunit)

## Front Controller

```
<?php  
require __DIR__ . '/../vendor/autoload.php';  
require __DIR__ . '/../src/config.php';  
$app = new \Slim\App(["settings" => $config]);  
require __DIR__ . '/../src/dependencies.php';  
require __DIR__ . '/../src/routes.php';  
$app->run();
```



## Config

```
<?php
```

```
$config['displayErrorDetails'] = true;
```

```
$config['db']['dsn'] = 'sqlite';
```

```
$config['db']['dbname'] = "../data/exampleapp.db";
```

## Dependencies

```
$container = $app->getContainer();  
$container['view'] = new \Slim\Views\PhpRenderer("../src/View/");  
$container['db'] = function ($c) {  
    $db = $c['settings']['db'];  
    $pdo = new PDO('sqlite:' . $db['dbname']);  
    $pdo->setAttribute(  
        PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION,  
        PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC  
    );  
    return $pdo;  
};
```

## Basic Route with Slim\Views\PhpRenderer

```
use Slim\Http\Request;  
use Slim\Http\Response;  
  
$app->get('/', function (Request $request, Response $response, array  
$args) {  
    $response = $this->view->render($response, "home.phtml");  
    return $response;  
});
```

## Routes given to Controllers

```
use App\Controller\  
    TicketsController,  
    NewTicketController,  
    AddTicketController,  
    FindTicketController  
;  
$app->get('/tickets', TicketsController::class);  
$app->get('/ticket/new', NewTicketController::class);  
$app->post('/ticket/new', AddTicketController::class);  
$app->get('/ticket/{id}', FindTicketController::class)  
    ->setName('ticket-detail');
```

## Grouped Routes

```
$app->group('/ticket', function () {  
    $app->get('/new', NewTicketController::class);  
    $app->post('/new', AddTicketController::class);  
    $app->get('/{id}', FindTicketController::class)  
        ->setName('ticket-detail');  
});
```

# Controllers

The coordinator

## Example: FindTicketController

```
namespace App\Controller;
use Slim\Http\Request;
use Slim\Http\Response;
use Psr\Container\ContainerInterface;
use App\Model\TicketMapper;

class FindTicketController
{
    protected $container;
    public function __construct(ContainerInterface $container) {
        $this->container = $container;
    }...
```

```
public function __invoke(Request $request, Response $response, array $args) {  
    $ticket_id = (int)$args['id'];  
    $ticket = $this->getTicketById($ticket_id);  
    $response = $this->container->view->render(  
        $response, "ticketdetail.phtml", ["ticket" => $ticket]  
    );  
    return $response;  
}  
  
private function getTicketById($ticket_id) {  
    $mapper = new TicketMapper($this->container->db);  
    return $mapper->getTicketById($ticket_id);  
}
```



# Model

The Data

```
namespace App\Model;
```

```
class TicketEntity
```

```
{
```

```
    protected $id;
```

```
    protected $title;
```

```
    public function __construct(array $data) {
```

```
        if(isset($data['id'])) { // no id if we're creating
```

```
            $this->id = $data['id'];
```

```
        }
```

```
        $this->title = $data['title'];
```

```
    }
```

```
    public function getId() {
```

```
        return $this->id;
```

```
    }
```

# View

Visual Presentation

```
<html>
<head>
  <title>Example Application</title>
  <link rel="stylesheet"
href="http://yui.yahooapis.com/pure/0.6.0/pure-min.css">
</head>
<body>
<h1><?=$ticket->getTitle()?></h1>

<p><strong><?=$ticket->getDescription()?></strong></p>

<p>Component: <?=$ticket->getComponent()?></p>
</body>
</html>
```

# Framing the Future

How frameworks help you

## So. Many. Frameworks.

### Full stack:

- ▶ Symfony
- ▶ Zend Framework
- ▶ Laravel
- ▶ CakePHP
- ▶ CodeIgniter
- ▶ Kohana
- ▶ Yii
- ▶ Lithium

### Micro:

- ▶ Silex
- ▶ Slim
- ▶ Fat-Free Framework
- ▶ Aura
- ▶ Phalcon

And more...

# Questions and Answers

Thank you for your participation