Foundations of Zend Framework

By:
Adam Culp
Twitter: @adamculp

[https://joind.in/**14923**](https://joind.in/14923)

Foundations of Zend Framework

- **About me**
  - PHP 5.3 Certified
  - Consultant at Zend Technologies
  - Organizer SoFloPHP (South Florida)
  - Organizer SunshinePHP (Miami)
  - Long Distance (ultra) Runner
  - Judo Black Belt Instructor

Foundations of Zend Framework

- **What is...**
  - Uses PHP >= 5.5
  - Open Source
    - On GitHub
  - Diverse Install
    - Pyrus, Composer, Git Submodules
  - Built on MVC design pattern
  - Can be used as components or entire framework

Foundations of Zend Framework

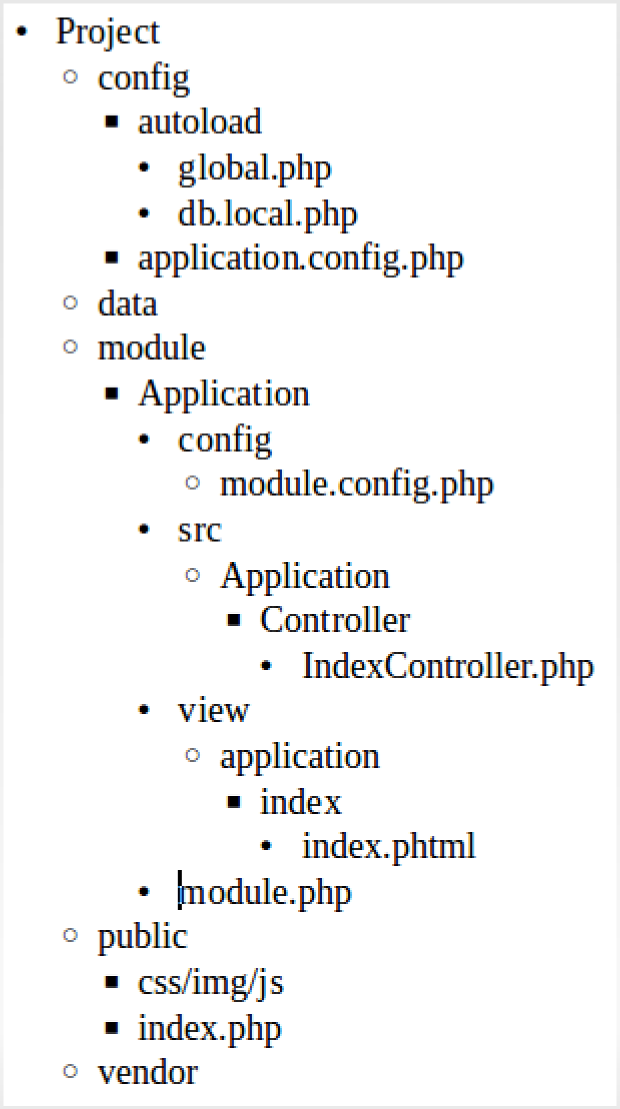
- **Skeleton Application**
  - Git clone Zendframework Skeleton Application
    - Github /zendframework/ZendSkeletonApplication

Foundations of Zend Framework

- **Composer**
  - Install Zend Framework 2
    - php composer.phar install
      - Creates and/or populates '`/vendor`' directory
      - Clones Zend Framework 2
      - Sets up Composer autoloader (PSR-0)
    - composer create-project zendframework/skeleton-application

Foundations of Zend Framework

- **Structure**



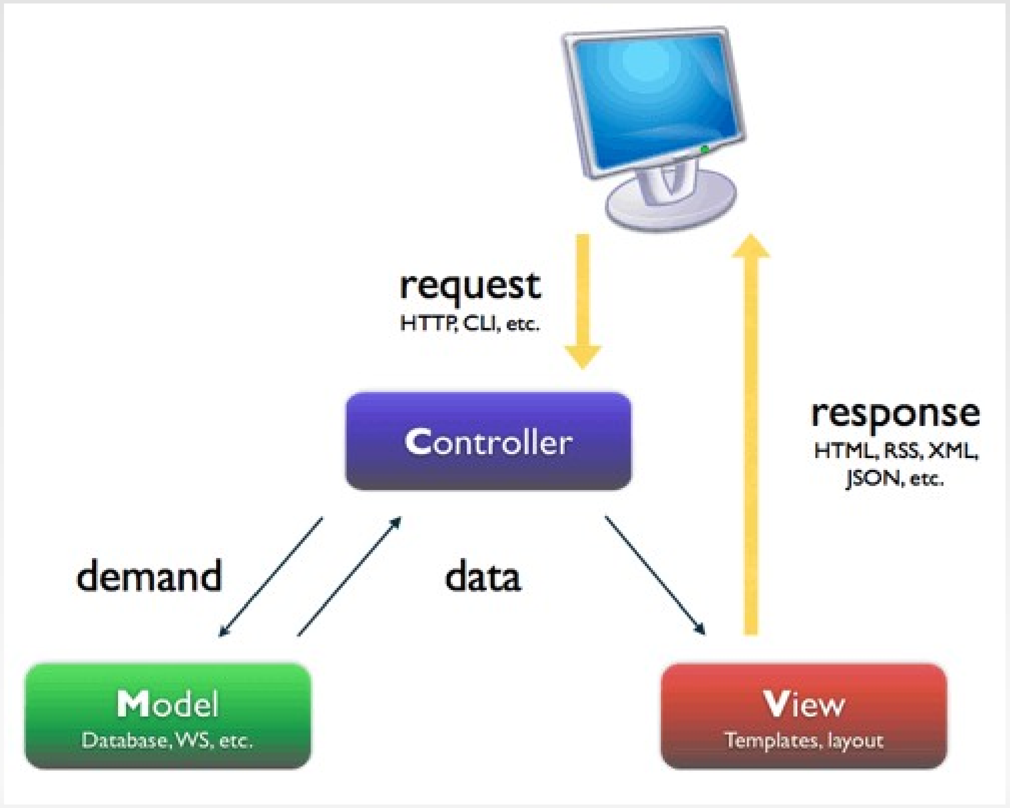
- Project
  - config
    - autoload
      - global.php
      - db.local.php
    - application.config.php
  - data
  - module
    - Application
      - config
        - module.config.php
      - src
        - Application
          - Controller
            - IndexController.php
      - view
        - application
          - index
            - index.phtml
      - module.php
  - public
    - css/img/js
    - index.php
  - vendor

Foundations of Zend Framework

- **Zend Framework 2 Usage**
  - NO MAGIC!!!
    - Configuration driven
    - No forced structure
    - Uses namespaces

Foundations of Zend Framework

- **MVC**
  - Very briefly



Foundations of Zend Framework

- **Typical Application Flow - Load**
  - index.php
    - Loads autoloader (PSR-0 = default)
    - init Application using application.config.php

Foundations of Zend Framework

- **Typical Application Flow – App Config**
  - application.config.php
    - Loads modules one at a time
      - (Module.php = convention)
    - Specifies where to find modules
    - Loads configs in autoload directory (DB settings, etc.)

Foundations of Zend Framework

- **Typical Application Flow – Modules**
  - Module.php (convention)
    - Makes MvcEvent accessible via onBootstrap()
      - Giving further access to Application, Event Manager, and Service Manager.
    - Loads module.config.php
    - Specifies autoloader and location of files.
    - May define services and wire event listeners as needed.

Foundations of Zend Framework

- **Typical Application Flow – Module Config**
  - module.config.php
    - Containers are component specific
      - Routes
      - Navigation
      - Service Manager
      - Translator
      - Controllers
      - View Manager
    - Steer clear of Closures (Anonymous Functions)
      - Do not cache well within array.
      - Less performant (parsed and compiled on every req) as a factory only parsed when service is used.

Foundations of Zend Framework

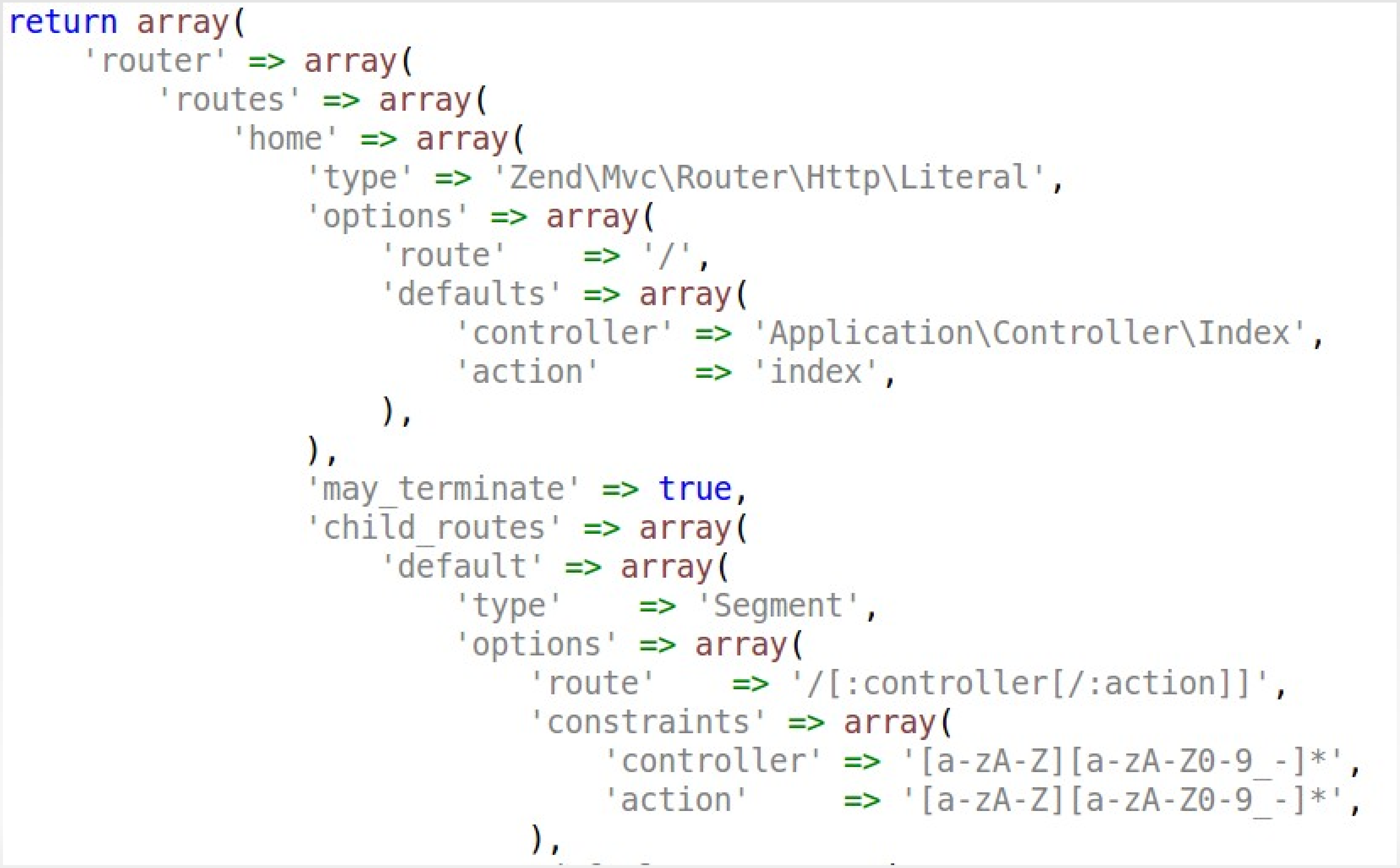
- Routes

Foundations of Zend Framework

- **Routes**
  - Carries how controller maps to request
  - Types:
    - **Hostname** – 'me.adamculp.com'
    - **Literal** - '/home'
    - **Method** – 'post,put'
    - **Part** – creates a tree of possible routes
    - **Regex** – use regex to match url '/blog/?<id>[0-9]?'
    - **Scheme** – 'https'
    - **Segment** - '/:controller[/:action][/]'
    - **Query** – specify and capture query string params

Foundations of Zend Framework

- **Route Example**



```
return array(
    'router' => array(
        'routes' => array(
            'home' => array(
                'type' => 'Zend\Mvc\Router\Http\Literal',
                'options' => array(
                    'route'    => '/',
                    'defaults' => array(
                        'controller' => 'Application\Controller\Index',
                        'action'     => 'index',
                    ),
                ),
                'may_terminate' => true,
                'child_routes' => array(
                    'default' => array(
                        'type'    => 'Segment',
                        'options' => array(
                            'route'    => '/[:controller[/:action]]',
                            'constraints' => array(
                                'controller' => '[a-zA-Z][a-zA-Z0-9_-]*',
                                'action'     => '[a-zA-Z][a-zA-Z0-9_-]*',
                            ),
```

/module/Application/config/module.config.php

Foundations of Zend Framework

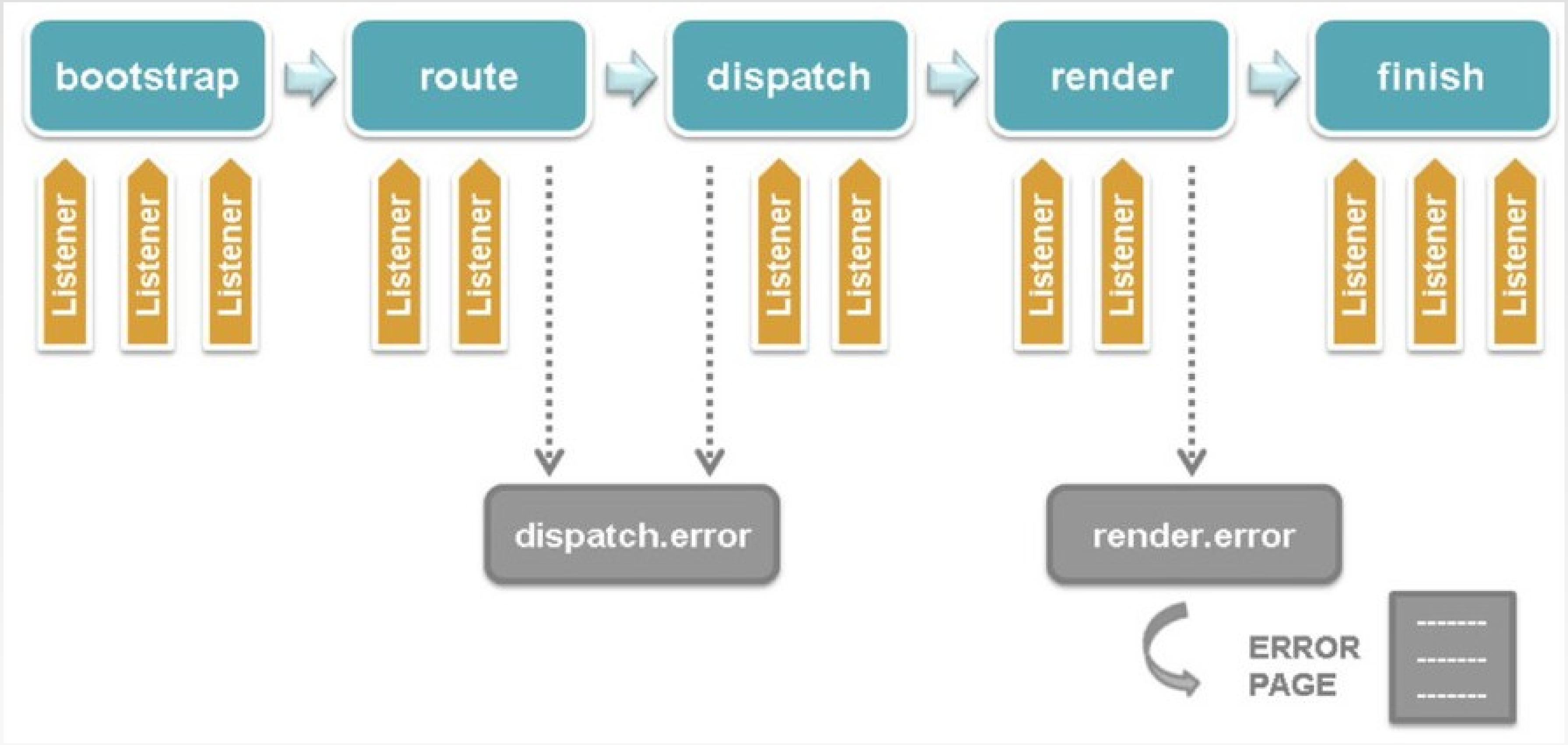
- **Event Manager**

Foundations of Zend Framework

- **Event Manager**
  - Many Event Managers
  - Each is isolated
  - Events are actions
  - Many custom we create
  - Defaults (following slide)

Foundations of Zend Framework

- **Diagram of MVC Events**



Foundations of Zend Framework

- **Event Manager Example**

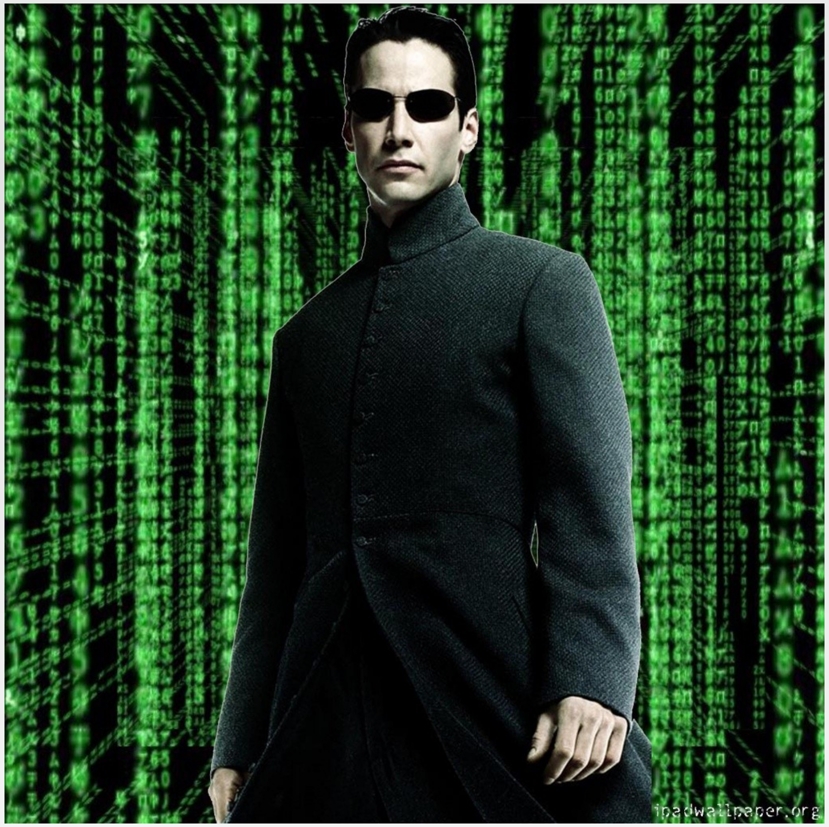
```
class Module
{
    public function onBootstrap(MvcEvent $e)
    {
        $eventManager        = $e->getApplication()->getEventManager();
        $eventManager->attach(MvcEvent::EVENT_DISPATCH, [$this, 'onDispatch']);
    }

    public function onDispatch(MvcEvent $e) {
        $viewModel = $e->getViewModel();
        $viewModel->setVariable('title', 'My Great Title!');
    }
```

/module/Application/Module.php

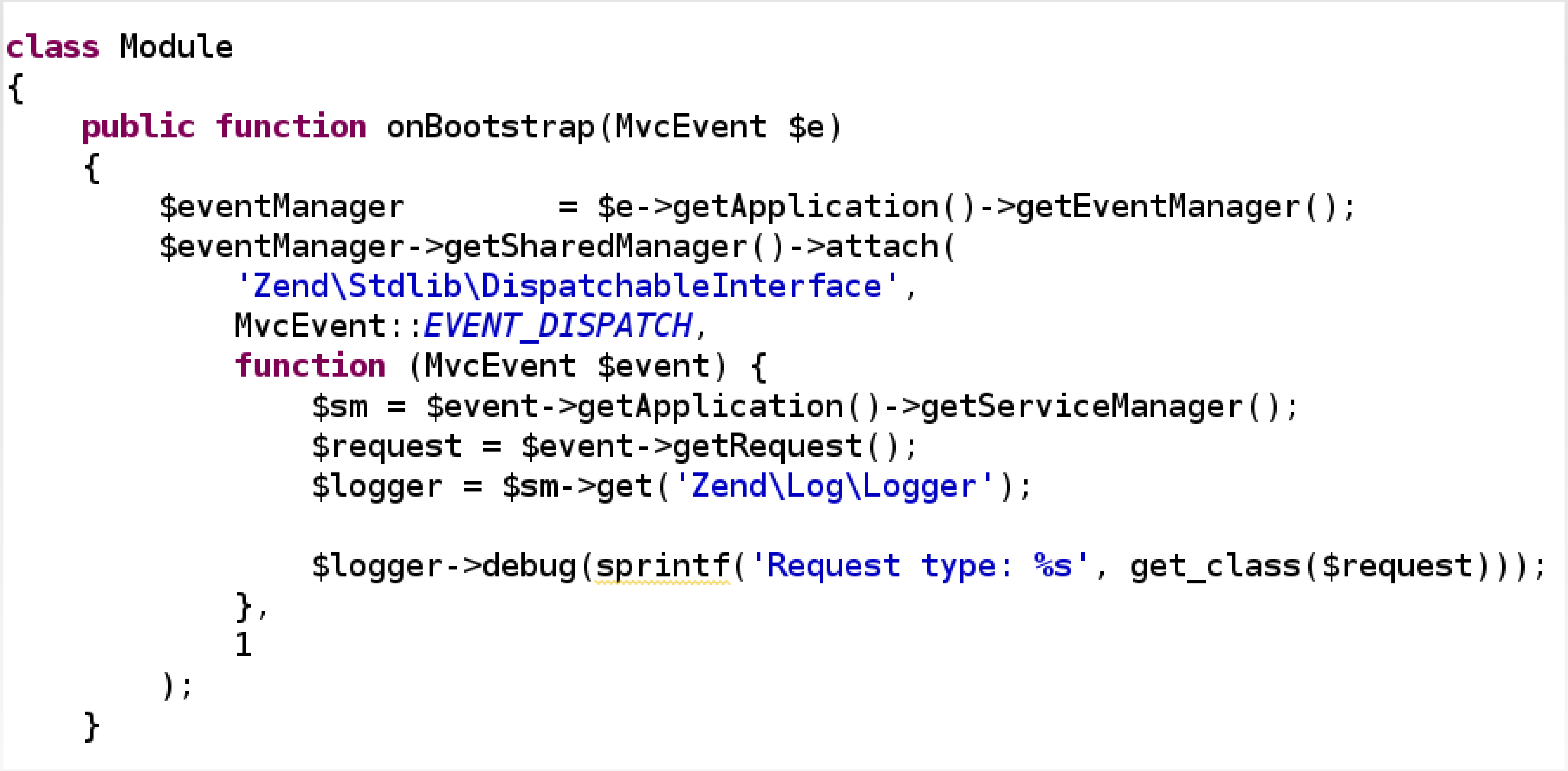
Foundations of Zend Framework

- **Shared Event Manager**
  - There is only one!
  - Similar to Event Manager
  - Obtain from Event Manager
  - Allows other lower level Event Managers to communicate with each other.
  - Globally available
  - Why?
    - May want to attach to objects not yet created, such as attaching to all controllers.



Foundations of Zend Framework

- **Shared Event Manager Example**



```
class Module
{
    public function onBootstrap(MvcEvent $e)
    {
        $eventManager        = $e->getApplication()->getEventManager();
        $eventManager->getSharedManager()->attach(
            'Zend\Stdlib\DispatchableInterface',
            MvcEvent::EVENT_DISPATCH,
            function (MvcEvent $event) {
                $sm = $event->getApplication()->getServiceManager();
                $request = $event->getRequest();
                $logger = $sm->get('Zend\Log\Logger');

                $logger->debug(sprintf('Request type: %s', get_class($request)));
            },
            1
        );
    }
```

/module/Application/Module.php

Foundations of Zend Framework

- **Event Manager Characteristics**
  - An Object
  - Attach Triggers to Events
  - Listeners are callbacks when Trigger satisfied
    - Function or Anon Function (action)
  - Queues by priority (last parameter in call)
  - Patterns
    - Pub/Sub
      - Class that triggers the event is publisher
      - Listener is subscribing to the event
    - Observer - (Subject/Observer)
      - Listener is the observer
      - Class Triggering the event is the subject

Foundations of Zend Framework

- **Services**
  - ALL THE THINGS!

Foundations of Zend Framework

- **Service Manager**
  - Recommended alternative to Zend\Di
    - Di pure DIC, SM is factory-based container
  - Everything is a service, even Controllers
  - Can be created from:
    - Application configuration
    - Module classes
      - Useful if anon functions desired
    - Module configuration (most common)
      - No anon functions due to caching issues
    - Local override configuration
      - For times when vendor keys need over-written
      - Specified in application config

Foundations of Zend Framework

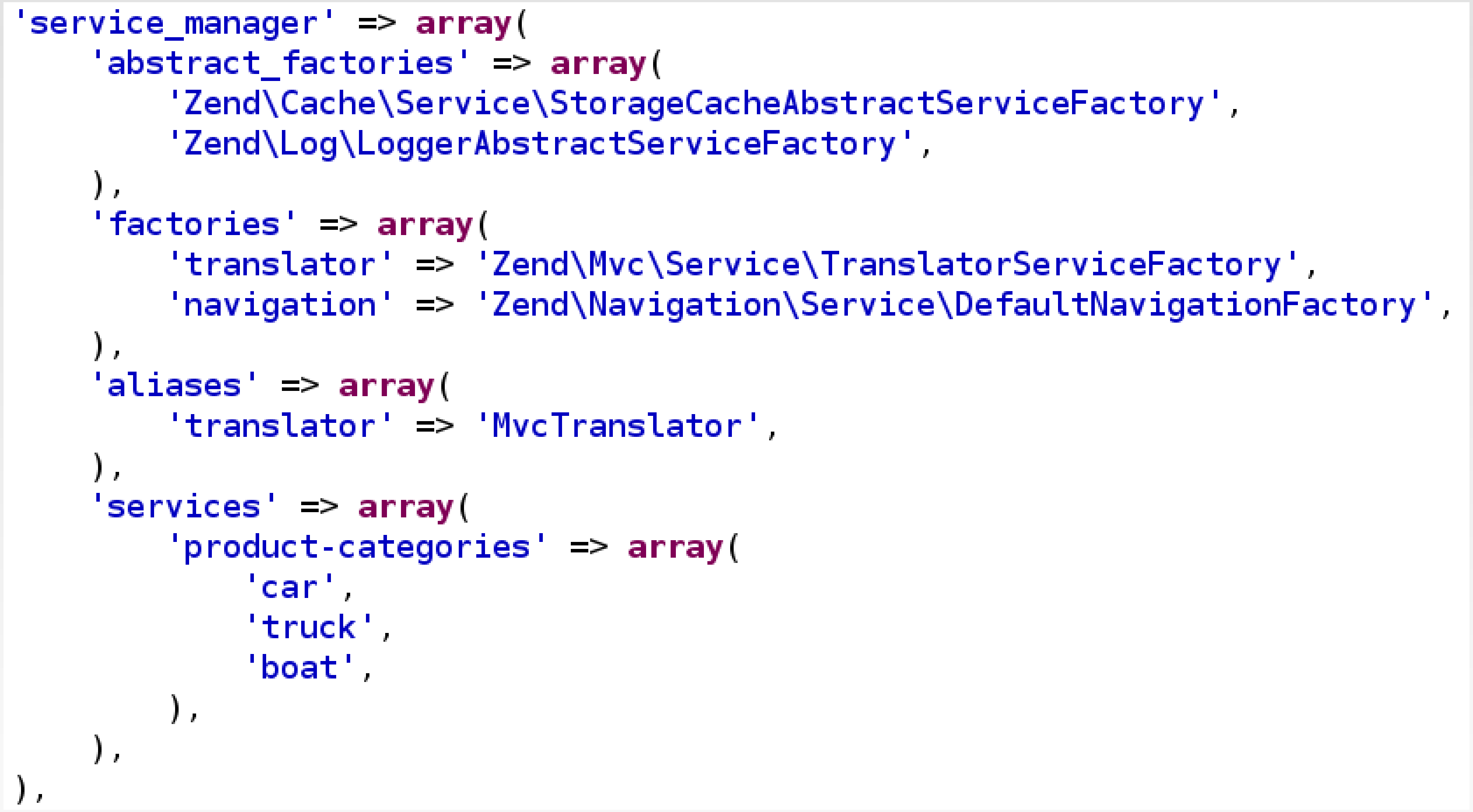
- **Defining Services**
  - Beware key name overwritting
    - Use fully qualified class name when applicable
      - \MyApp\Controller\Product\Index
    - Or be descriptive
      - {module}-{name}
      - 'product-category' instead of 'category'
    - All keys get normalized
      - \App\Controller\Product\Index == app-controller-product-index
  - Hierarchy of definitions
    - Module.php – initial
    - module.config.php over-rides Module.php
    - Local over-rides module.config.php

Foundations of Zend Framework

- **Service Types**
  - Types:
    - Services – Explicit
      - key => value pairs (string, boolean, float, object)
    - Invokables
      - key => class (class/object with no needed dependencies)
    - Factories
      - key => object (class/object with needed dependencies)
    - Aliases (name => some other name)
    - Abstract Factories (unknown services)
    - Scoped Containers (limit what can be created)
    - Shared (or not; you decide)

Foundations of Zend Framework

- **Service Config Example**

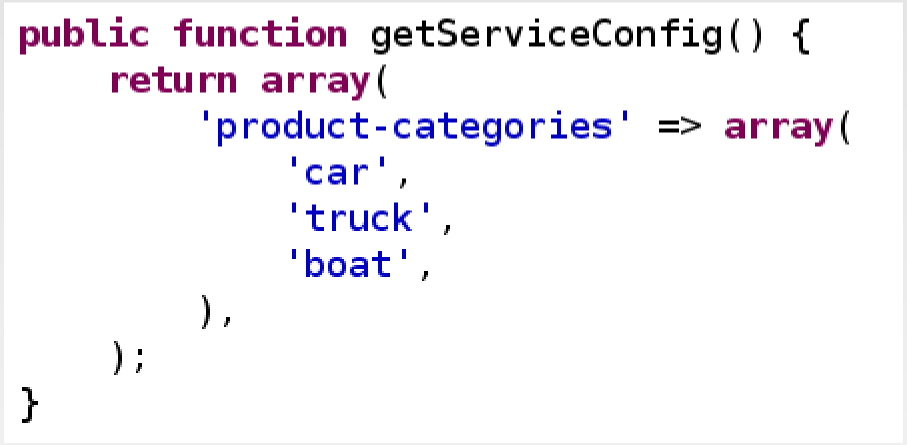


```php
'service_manager' => array(
    'abstract_factories' => array(
        'Zend\Cache\Service\StorageCacheAbstractServiceFactory',
        'Zend\Log\LoggerAbstractServiceFactory',
    ),
    'factories' => array(
        'translator' => 'Zend\Mvc\Service\TranslatorServiceFactory',
        'navigation' => 'Zend\Navigation\Service\DefaultNavigationFactory',
    ),
    'aliases' => array(
        'translator' => 'MvcTranslator',
    ),
    'services' => array(
        'product-categories' => array(
            'car',
            'truck',
            'boat',
        ),
    ),
),
```

/module/Application/config/module.config.php

Foundations of Zend Framework

- **Service Module Example**



```php
public function getServiceConfig() {
    return array(
        'product-categories' => array(
            'car',
            'truck',
            'boat',
        ),
    );
}
```

/module/Application/config/module.php

Foundations of Zend Framework

- **Using Service Example**

```
public function onDispatch(MvcEvent $e) {
    $viewModel = $e->getViewModel();
    $serviceManager = $e->getApplication()->getServiceManager();
    $viewModel->setVariable('categories', $serviceManager->get('product-categories'));
}
```

/module/Application/Module.php

```
<div class="col-lg-2">
    <?php echo $this->htmllist($this->categories); ?>
</div>
```

/module/Application/view/layout/layout.phtml

Foundations of Zend Framework

- **Module Manager**

Foundations of Zend Framework

- **Module Manager**
  - Gets directives from application.config.php
    - Modules to load
      - Order is important if module depends on another
    - Where to find modules (convention found in)
      - Modules directory
      - Vendor directory
  - Loads each module
    - Module.php
      - For dynamic content/settings
    - module.config.php (if getConfig() in Module.php)
      - Over-rides Module.php
  - Then hand off to MvcEvent process to Bootstrap.

Foundations of Zend Framework

- **Module Basics**
  - Related for a *specific* “problem”.
  - Logical separation of application functionality
    - Reusable
    - Removing a module doesn't kill the application
    - Contains everything specific to given module
  - Keep init() and onBootstrap() in modules light.
  - Do not add data to module structure.

Foundations of Zend Framework

- **Module Contents**
  - Contents
    - PHP Code
    - MVC Functionality
    - Library Code
      - Though better in Application or via Composer
      - May not be related to MVC
    - View scripts
    - Public assets (images, css, javascript)
    - More?

Foundations of Zend Framework

- **Module Skeleton**
  - Easy creation using Zend Skeleton Module
    - GitHub /zendframework/ZendSkeletonModule

Foundations of Zend Framework

- **Leveraging Middleware and PSR-7**
  - Vi

Foundations of Zend Framework

- **Other Things Worth Investigating**
  - Views
  - Forms
  - Databases
  - Navigation
  - View Strategies (Action or Restful)

Sorry, just not enough time in a regular talk.

Foundations of Zend Framework

- **Resources**
  - http://framework.zend.com
  - http://www.zend.com/en/services/training/course-catalog/zend-framework-2
  - http://www.zend.com/en/services/training/course-catalog/zend-framework-2-advanced
  - http://zendframework2.de/cheat-sheet.html
  - http://apigility.org

Foundations of Zend Framework

- **Thank You!**
  - Rate this talk: https://joind.in/**14923**
  - Code: https://github.com/adamculp/foundations-zf2-talk

Adam Culp

http://www.geekyboy.com

http://RunGeekRadio.com

Twitter @adamculp