

# Как использовать Yii2 в Enterprise?

Взять лучшие практики из Symfony!

# Ссылка на презентацию

<https://github.com/wowworks-team/phpconf>



# Обо мне

CIO Wowworks.

Более 10 лет в  
веб разработке.

**wowworks**

**ELEMEN  
TAREE**



**Вебзавод**  
системный интегратор

Главный критерий

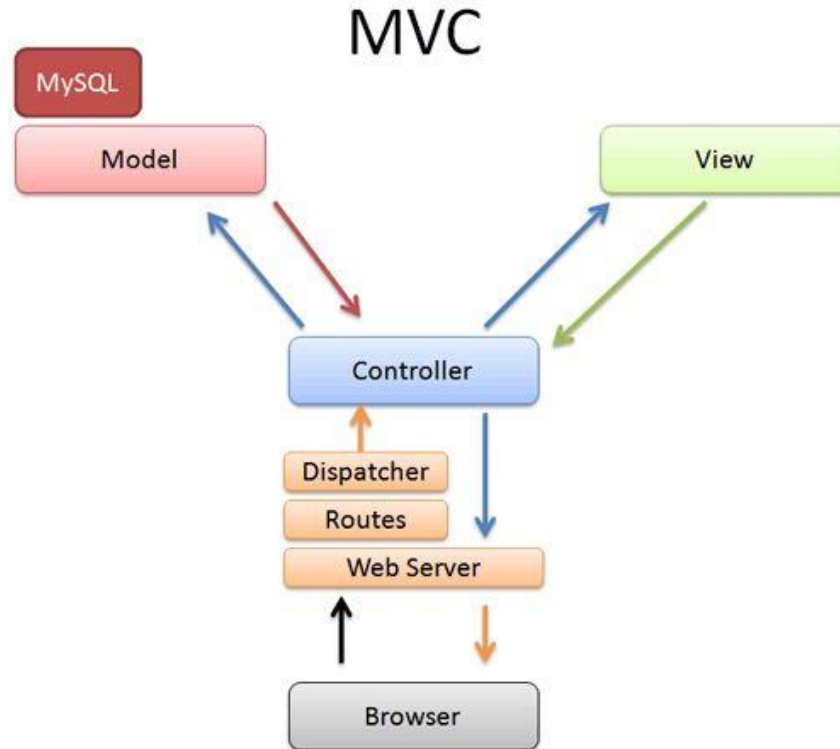
Unit test

# 8 СОВЕТОВ

для хорошего кода

# 1. Тонкий контроллер

# MVC



# Пример из документации

```
24 // https://www.yiiframework.com/doc/guide/2.0/en/structure-controllers
25
26 class PostController extends Controller
27 {
28     public function actionView($id)
29     {
30         $model = Post::findOne($id);
31         if ($model === null) {
32             throw new NotFoundException;
33         }
34
35         return $this->render( view: 'view', [
36             'model' => $model,
37         ]);
38     }
39 }
```



# Толстый/тонкий контроллер



# Тонкий контроллер

```
<?php
```

```
namespace app\modules\admin\controllers;
```

```
use yii\web\Controller;
```

```
/**  
 * Default controller for the `admin` module  
 */
```

```
class DefaultController extends Controller  
{
```

```
    /**  
     * Renders the index view for the module  
     * @return string  
     */
```

```
    public function actionIndex()  
    {  
        return $this->render('index');  
    }  
}
```

# Толстый контроллер

<https://github.com/magento/magento2/blob/2.3-develop/app/code/Magento/Checkout/Controller/Cart/Add.php>

```
104     $product = $this->_initProduct();
105     $related = $this->getRequest()->getParam('related_product');
106
107     /**
108      * Check product availability
109      */
110     if (!$product) {
111         return $this->goBack();
112     }
113
114     $this->cart->addProduct($product, $params);
115     if (empty($related)) {
116         $this->cart->addProductsByIds(explode(',', $related));
117     }
118
119     $this->cart->save();
120
121     /**
122      * @todo remove wishList observer \Magento\Wishlist\Observer\AddToCart
123      */
124     $this->eventManager->dispatch(
125         'checkout_cart_add_product_complete',
126         ['product' => $product, 'request' => $this->getRequest(), 'response' => $this->getResponse()]
127     );
128
129     if (!$this->checkoutSession->getNoCartRedirect(true)) {
130         if (!$this->cart->getQuote()->getHasError()) {
131             if ($this->shouldRedirectToCart()) {
132                 $message = __(
133                     'You added %1 to your shopping cart.',
134                     $product->getName()
135                 );
136                 $this->messageManager->addSuccessMessage($message);
137             } else {
138                 $this->messageManager->addComplexSuccessMessage(
139                     'addCartSuccessMessage',
140                     [
141                         'product_name' => $product->getName(),
142                         'cart_url' => $this->getCartUrl(),
143                     ]
144                 );
145             }
146         }
147         return $this->goBack(null, $product);
148     }
```

# Пример из документации

```
public function actionCreate()
{
    $model = new Post;

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render( view: 'create', [
            'model' => $model,
        ]);
    }
}
```

## Добавим логику

```
public function actionCreate()
{
    $model = new Post;

    $model->is_published = false;
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render( view: 'create', [
            'model' => $model,
        ]);
    }
}
```

# Новая задача от бизнеса

```
public function actionCreateConsole($title, $content)
{
    $model = new Post;

    $model->title = $title;
    $model->content = $content;
    $model->is_published = false;
    if ($model->save()) {
        Console::output(sprintf( format: 'Post %d created', $model->id));
    } else {
        Console::error(sprintf( format: 'Post not created'));
    }
}
```

# Убираем копипасту

```
public function actionCreateService($title, $content)
{
    $service = $this->getPostService();
    $model = new Post;

    $model->title = $title;
    $model->content = $content;
    if ($service->createPost($model)) {
        Console::output(sprintf( format: 'Post %d created', $model->id));
    } else {
        Console::error(sprintf( format: 'Post not created'));
    }
}
```


# Сервис

```
class PostService
{
    public function createPost(Post $model)
    {
        $model->is_published = false;

        return $model->save();
    }
}
```



# А еще оповещения нужны ...

```
class NotificationService  
{  
    public function send(string $text): bool  
    {  
        // send some notification  
        return true;  
    }  
}
```

```
class PostService2
```

```
{
```

```
    /** @var NotificationService */
```

```
    private $notificationService;
```

```
public function __construct(NotificationService $notificationService)
```

```
{
```

```
    $this->notificationService = $notificationService;
```

```
}
```

```
public function createPost(Post $model): bool
```

```
{
```

```
    $model->is_published = false;
```

```
    $result = $model->save();
```

```
    if ($result) {
```

```
        $this->notificationService->send( text: 'New post created');
```

```
    }
```

```
    return $result;
```

```
}
```

```
}
```

## 2. Разные типы моделей

# Модели

- Entity
- Repository
- Service
- Others
  - Helper
  - Query
  - Event
  - Handler
  - Exception
  - Form
  - ...

# Модели

Entity - машина



Repository - поиск нужной машины

Service - езда



# Entity



```
/**
 * Class User
 *
 * @property int $id
 */
class User extends ActiveRecord
{
}
}
```

# Repository



```
class PostRepository
{
    public function getPostsQuery(User $user): ActiveQuery
    {
        return Post::find()->where(['user_id' => $user->id]);
    }
}
```

# Service



```
class PostService
{
    public function createPost(Post $model)
    {
        $model->is_published = false;

        return $model->save();
    }
}
```



# Модели

ActiveRecord =

Entity + Repository + Entity Manager

Doctrine 2 ORM



## Active Record

- Сама сущность
- Методы поиска (find, findOne, findAll)
- Методы работы с базой данной (insert, update, delete, refresh)

### 3. Используйте Dependency injection

# DI

```
public function __construct(  
    NotificationService $notificationService,  
    VkService $vkService,  
    InstagramService $instagramService,  
    StatisticsService $statisticsService  
) {  
    $this->notificationService = $notificationService;  
    $this->vkService = $vkService;  
    $this->instagramService = $instagramService;  
    $this->statisticsService = $statisticsService;  
}
```

# Много зависимостей

```
$notificationService = new NotificationService(/* with some settings */);  
$vkService = new VkService(/* with API key*/);  
$instagramService = new InstagramService(/* with API key*/);  
$statisticsService = new StatisticsService(/* with settings */);  
  
new PostService3($notificationService, $vkService, $instagramService, $statisticsService);
```

# DI

<https://www.yiiframework.com/doc/guide/2.0/en/concept-di-container>

Официальная документация

<https://slides.silverfire.me/2016/yii-di/#/>

Дмитрий Науменко



# DI Использование

```
public function actionCreateService($title, $content)
{
    $service = Yii::createObject( type: PostService::class);
    $model = new Post;

    $model->title = $title;
    $model->content = $content;
    if ($service->createPost($model)) {
        Console::output(sprintf( format: 'Post %d created', $model->id));
    } else {
        Console::error(sprintf( format: 'Post not created'));
    }
}
```

DI Использование

Yii::createObject

Controllers only



# DI Подключение

common/config/main.php

```
<?php
return [
    // ...
    // https://www.yiiframework.com/doc/guide/2.0/en/concept-configurations#application-configurations
    // https://www.yiiframework.com/doc/guide/2.0/en/concept-di-container#advanced-practical-usage
    'container' => [
        'definitions' => [
            'yii\widgets\LinkPager' => ['maxButtonCount' => 5]
        ],
        'singletons' => [
            // Dependency Injection Container singletons configuration
        ]
    ],
    'components' => [
        // ...
    ],
];
```

# DI Подключение

common/config/main.php

```
1  <?php
2  return [
3      // ...
4      'bootstrap' => [
5          'common\config\BootstrapContainer',
6      ],
7      'components' => [
8          // ...
9      ],
10 ];
11
```

# DI common/config/BootstrapContainer.php

```
use common\services\Jira\JiraService;
use Yii;
use yii\base\BootstrapInterface;

class BootstrapContainer implements BootstrapInterface
{
    /**
     * @inheritdoc
     */
    public function bootstrap($app)
    {
        $container = Yii::$container;

        $container->setSingleton(
            class: JiraService::class,
            [
                'class' => JiraService::class,
            ],
            [
                $app->params['jira']['host'],
                $app->params['jira']['user'],
                $app->params['jira']['token'],
            ]
        );
    }
}
```

## 4. Правильные Events

# Events

- Отдельный файл конфига
- Тонкий handler
- Логика в сервисах

# Events config/BootstrapEvents.php

```
1  <?php
2
3  namespace common\config;
4
5  use common\eventHandlers\InsertEventHandler;
6  use Yii;
7  use yii\base\BootstrapInterface;
8  use yii\base\Event;
9  use yii\db\ActiveRecord;
10
11 class BootstrapEvents implements BootstrapInterface
12 {
13     /**
14      * @inheritdoc
15      */
16     public function bootstrap($app)
17     {
18         $this->attachActiveRecordHandlers();
19     }
20
21     private function attachActiveRecordHandlers()
22     {
23         Event::on( class: ActiveRecord::class, name: ActiveRecord::EVENT_AFTER_INSERT, [
24             Yii::createObject( type: InsertEventHandler::class),
25             'handle'
26         ]);
27     }
28 }
29
```

# Events config/main.php

```
<?php
return [
    // ...
    'bootstrap' => [
        'common\config\BootstrapContainer',
        'common\config\BootstrapEvents',
    ],
    'components' => [
        // ...
    ],
];
```

# Events \common\eventHandlers\InsertEventHandler

```
<?php

namespace common\eventHandlers;

use common\services\LoggerService;
use yii\base\Event;
use yii\db\ActiveRecord;

class InsertEventHandler
{
    /** @var LoggerService */
    private $loggerService;

    public function __construct(LoggerService $loggerService)
    {
        $this->loggerService = $loggerService;
    }

    public function handle(Event $event)
    {
        if ($event->sender instanceof ActiveRecord) {
            // call service and do some logic
            $this->loggerService->logActiveRecord($event->sender);
        }
    }
}
```



# Тесты

# Codeception

- acceptance - selenium / phpbrowser
- functional - routing in framework
- api - удобное тестирование REST API
- unit - phpunit

# Simple Unit Test

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

class StackTest extends TestCase
{
    public function testPushAndPop()
    {
        $stack = [];
        $this->assertSame(0, count($stack));

        array_push($stack, 'foo');
        $this->assertSame('foo', $stack[count($stack)-1]);
        $this->assertSame(1, count($stack));

        $this->assertSame('foo', array_pop($stack));
        $this->assertSame(0, count($stack));
    }
}
```

## 5. DateHelper & 6. Mock

```
/**
 * Class Profile
 *
 * @property string $updated_at
 */
class Profile extends \yii\db\ActiveRecord
{
    public function getUpdatedAt(): ?DateTime
    {
        return $this->updated_at ? new DateTime($this->updated_at) : null;
    }

    // some logic with date
    public function isInactive(): bool
    {
        $date = new DateTime( time: '-3 month' );
        return $this->getUpdatedAt() < $date;
    }
}
```

```
class ProfileTest extends \PHPUnit\Framework\TestCase
{
    /** @var Profile */
    private $object;

    protected function setUp()
    {
        parent::setUp();
        $this->object = new Profile();
    }

    public function testGetUpdatedAt()
    {
        $this->object->updated_at = null;
        $this->assertNull($this->object->getUpdatedAt());

        $this->object->updated_at = '2019-01-01';
        $this->assertInstanceOf( expected: DateTime::class, $this->object->getUpdatedAt());
    }
}
```



```
// some logic with date  
public function isInactive(): bool  
{  
    $date = new DateTime( time: '-3 month');  
    return $this->getUpdatedAt() < $date;  
}
```

```
public function testIsInactive()  
{  
    // ... ???  
}
```

```
- class ProfileService
{
    /** @var DateHelper */
    private $dateHelper;

    /**
     * ProfileService constructor.
     * @param DateHelper $dateHelper
     */
    public function __construct(DateHelper $dateHelper)
    {
        $this->dateHelper = $dateHelper;
    }

    public function isInactive(Profile $profile): bool
    {
        $date = $this->dateHelper->getDateTime( time: '-3 month');
        return $profile->getUpdatedAtDateTime() < $date;
    }
}
```



```
<?php
```

```
namespace App\Helper;
```

```
use DateTime;
```

```
class DateHelper
```

```
{
```

```
    public function getDateTime(?string $time = null): DateTime
```

```
    {
```

```
        return new DateTime($time);
```

```
    }
```

```
}
```

```
// not working example, depends on current time
public function testIsInactiveBad()
{
    $helper = new DateHelper();
    $service = new ProfileService($helper);
    $profile = new Profile();

    $profile->updated_at = '2018-01-01';
    $this->assertTrue($service->isInactive($profile));

    $profile->updated_at = '2019-01-01';
    $this->assertFalse($service->isInactive($profile));
}
```

```
public function testIsInactive()
{
    $helper = new DateHelperMock();
    $service = new ProfileService($helper);
    $profile = new Profile();

    $helper->setCurrentDate('2019-01-01');
    $profile->updated_at = '2018-01-01';
    $this->assertTrue($service->isInactive($profile));

    $helper->setCurrentDate('2019-01-01');
    $profile->updated_at = '2019-01-01';
    $this->assertFalse($service->isInactive($profile));
}
```

```
class DateHelperMock extends DateHelper
{
    /** @var DateInterval */
    private $diff;

    public function setCurrentDate(string $date)
    {
        $today = $this->getDateTime();
        $newDate = $this->getDateTime($date);
        $this->diff = $today->diff($newDate);
    }

    public function getDateTime(?string $time = null): DateTime
    {
        $dateTime = parent::getDateTime($time);
        return $this->diff ? $dateTime->add($this->diff) : $dateTime;
    }
}
```

# Mockery

<https://github.com/mockery/mockery>

<http://docs.mockery.io>



# Mock active record

```
public function testGetUpdatedAtDateTimeFailed()  
{  
    $object = new Profile();  
    $object->getUpdatedAtDateTime();  
}
```

/usr/bin/php /Users/vdemin/phpconf/vendor/phpunit/phpunit/phpunit --no-configuration A

Testing started at 14:48 ...

PHPUnit 7.5.7 by Sebastian Bergmann and contributors.

Error : Call to a member function getDb() on null

/Users/vdemin/phpconf/vendor/yiisoft/yii2/db/ActiveRecord.php:135

/Users/vdemin/phpconf/vendor/yiisoft/yii2/db/ActiveRecord.php:433

/Users/vdemin/phpconf/vendor/yiisoft/yii2/db/ActiveRecord.php:469

/Users/vdemin/phpconf/vendor/yiisoft/yii2/db/BaseActiveRecord.php:500

/Users/vdemin/phpconf/vendor/yiisoft/yii2/db/BaseActiveRecord.php:291

/Users/vdemin/phpconf/src/ActiveRecord/Profile.php:17

/Users/vdemin/phpconf/tests/ActiveRecord/ProfileTest.php:21

# Mock active record

```
/**
 * @return Profile|Mockery\Mock
 */
private function getMock()
{
    /** @var Profile|Mockery\Mock $object */
    $object = Mockery::mock( ...args: Profile::class);
    $object->shouldReceive( ...methodNames: 'attributes' )->andReturn( ['updated_at'] );
    $object->makePartial();

    return $object;
}
```



# Mock active record

```
/**
 * @dataProvider emptyDataProvider
 * @param $value
 */
public function testGetUpdatedAtDateTimeWithEmpty($value)
{
    $object = $this->getMock();
    $object->updated_at = $value;
    $this->assertNull($object->getUpdatedAtDateTime());
}

public function emptyDataProvider(): array
{
    return [
        [null],
        [''],
        [0],
        [false],
    ];
}
```



```

/**
 * @dataProvider dateTimeDataProvider
 * @param $value
 */
public function testGetUpdatedAtDateTime($value)
{
    $object = $this->getMock();
    $object->updated_at = $value;
    $this->assertInstanceOf( expected: DateTime::class, $object->getUpdatedAtDateTime());
}

public function dateTimeDataProvider(): array
{
    return [
        ['now'],
        ['2019-01-01'],
        ['tomorrow'],
        ['+1 day'],
    ];
}

```



7. Static/global are evil

8. Оберните в геттеры вызовы к `Yii::`,  
если не можете их избежать

# Static/global are evil

`Yii::$app->request`

`Yii::$app->cache`

`Yii::debug()`



```

class PostFromGuideController extends Controller
{
    public function actionView($id)
    {
        $model = Post::findOne($id);
        if ($model === null) {
            throw new NotFoundException;
        }

        return $this->render( view: 'view', [
            'model' => $model,
        ]);
    }

    public function actionCreate()
    {
        $model = new Post;

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        } else {
            return $this->render( view: 'create', [
                'model' => $model,
            ]);
        }
    }
}

```

```
public function actionView($id)
{
    $model = $this->getRepository()->findOne((int) $id);
    if ($model === null) {
        throw new NotFoundException;
    }

    return $this->render([view: 'view', [
        'model' => $model,
    ]);
}
```

```
protected function getRepository(): PostRepository
{
    return $this->postRepository;
}
```

```
/** @var PostRepository */  
protected $postRepository;  
  
// https://www.yiiframework.com/doc/guide/2.0/en/concept-di-container  
public function __construct(  
    $id,  
    $module,  
    PostRepository $postRepository,  
    $config = []  
) {  
    $this->postRepository = $postRepository;  
  
    parent::__construct($id, $module, $config);  
}
```

```
/**
 * @covers \App\Controller\PostController::actionView
 */
public function testActionViewNotFound()
{
    /** @var Mockery\Mock|PostRepository $repository */
    $repository = Mockery::mock( ...args: PostRepository::class);
    $repository->shouldReceive( ...methodNames: 'findOne');

    $object = new PostController(
        id: null,
        module: null,
        $repository
    );
    $this->expectException(NotFoundHttpException::class);
    $object->actionView( id: 0);
}
```



```
protected function tearDown()
{
    Mockery::close(); // https://github.com/mockery/mockery#method-call-expectations-
}

/**
 * @covers \App\Controller\PostController::actionView
 */
public function testActionView()
{
    $object = Mockery::mock( ...args: PostController::class);

    /** @var Mockery\Mock|PostRepository $repository */
    $repository = Mockery::mock( ...args: PostRepository::class);
    $repository->shouldReceive( ...methodNames: 'findOne')->andReturn(new Post());
    $object->shouldAllowMockingProtectedMethods();
    $object->shouldReceive( ...methodNames: 'getRepository')->andReturn($repository);
    $object->makePartial();

    $object->shouldReceive( ...methodNames: 'render')->once();
    $object->actionView( id: 1);
    $this->expectNotToPerformAssertions();
}
```



// hard example, use SPA and REST API instead ;) Logic move to service

```
public function createAction()
{
    $model = $this->getModel();

    if ($model->load($this->getRequest()->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render( view: 'create', [
            'model' => $model,
        ]);
    }
}
```

```
protected function getModel(): Post
{
    return $this->post;
}
```

```
/**
 * @return \yii\console\Request|\yii\web\Request
 */
protected function getRequest()
{
    return Yii::$app->request;
}
```

```
class PostControllerTest extends TestCase
{
  /** @var Mockery\Mock\PostController */
  private $object;

  protected function setUp()
  {
    parent::setUp();

    $this->object = Mockery::mock( ...args: PostController::class);
  }

  protected function tearDown()
  {
    Mockery::close(); // https://github.com/mockery/mockery#method-call-expectations-
  }
}
```

```

/**
 * @covers \App\Controller\PostController::actionCreate
 */
public function testActionCreateEmpty()
{
    /** @var Mockery\Mock\Request $request */
    $request = Mockery::mock( ...args: Request::class )->makePartial();
    /** @var Mockery\Mock\Post $model */
    $model = Mockery::mock( ...args: Post::class )->makePartial();
    $this->object->shouldAllowMockingProtectedMethods();
    $this->object->shouldReceive( ...methodNames: 'getRequest' )->andReturn($request);
    $this->object->shouldReceive( ...methodNames: 'getModel' )->andReturn($model);
    $this->object->makePartial();

    $this->object->shouldReceive( ...methodNames: 'render' )->once();
    $this->object->actionCreate();
    $this->expectNotToPerformAssertions();
}

```

```

/**
 * @covers \App\Controller\PostController::actionCreate
 */
public function testActionCreateRedirect()
{
    /** @var Mockery\Mock|Request $request */
    $request = Mockery::mock( ...args: Request::class);
    $request->shouldReceive( ...methodNames: 'post')->andReturn([]);
    /** @var Mockery\Mock|Post $model */
    $model = Mockery::mock( ...args: Post::class);
    $model->shouldReceive( ...methodNames: 'attributes')->andReturn(['id', 'text']);
    $model->shouldReceive( ...methodNames: 'load')->andReturn(true);
    $model->shouldReceive( ...methodNames: 'save')->andReturn(true);
    $model->makePartial();
    $this->object->shouldAllowMockingProtectedMethods();
    $this->object->shouldReceive( ...methodNames: 'getRequest')->andReturn($request);
    $this->object->shouldReceive( ...methodNames: 'getModel')->andReturn($model);
    $this->object->makePartial();

    $this->object->shouldReceive( ...methodNames: 'render')->never();
    $this->object->shouldReceive( ...methodNames: 'redirect')->once();
    $this->object->actionCreate();
    $this->expectNotToPerformAssertions();
}

```

# Итого

1. Тонкий контроллер
2. Модели разные, каждая отвечает за свое (SRP)
3. Используйте Dependency Injection
4. Правильно используйте Events
5. Создайте DateHelper для работы с датами
6. Пишите моки (Mockery)
7. Избегайте глобальных переменных во всех проявлениях и static
8. Оберните в геттеры вызовы к Yii::, если не можете их избежать

Пишите Unit тесты!

Тогда автоматически код станет  
лучше!

# Контакты

<https://www.facebook.com/2spacemen>

<https://www.linkedin.com/in/victor-demin/>

Презентация и примеры: <https://github.com/wowworks-team/phpconf>

