

Четвертая конференция разработчиков ПО «DevParty»

2 апреля 2016 года, Вологда



DEV PARTY
конференция разработчиков

Архитектура MVC в контексте web-разработки – проблемы и решения

Валерий Чугреев
vk.com/chugreev

План выступления

1. Вводная часть: фреймворки, MVC, ActiveRecord.
2. Имеющиеся проблемы и сложности, порождаемые MVC-фреймворками.
3. Пути решения проблем. Демонстрационный проект.
4. Плюсы и минусы предлагаемого подхода.

Примеры на PHP, Yii2

1. Вводная часть: фреймворки, MVC, ActiveRecord

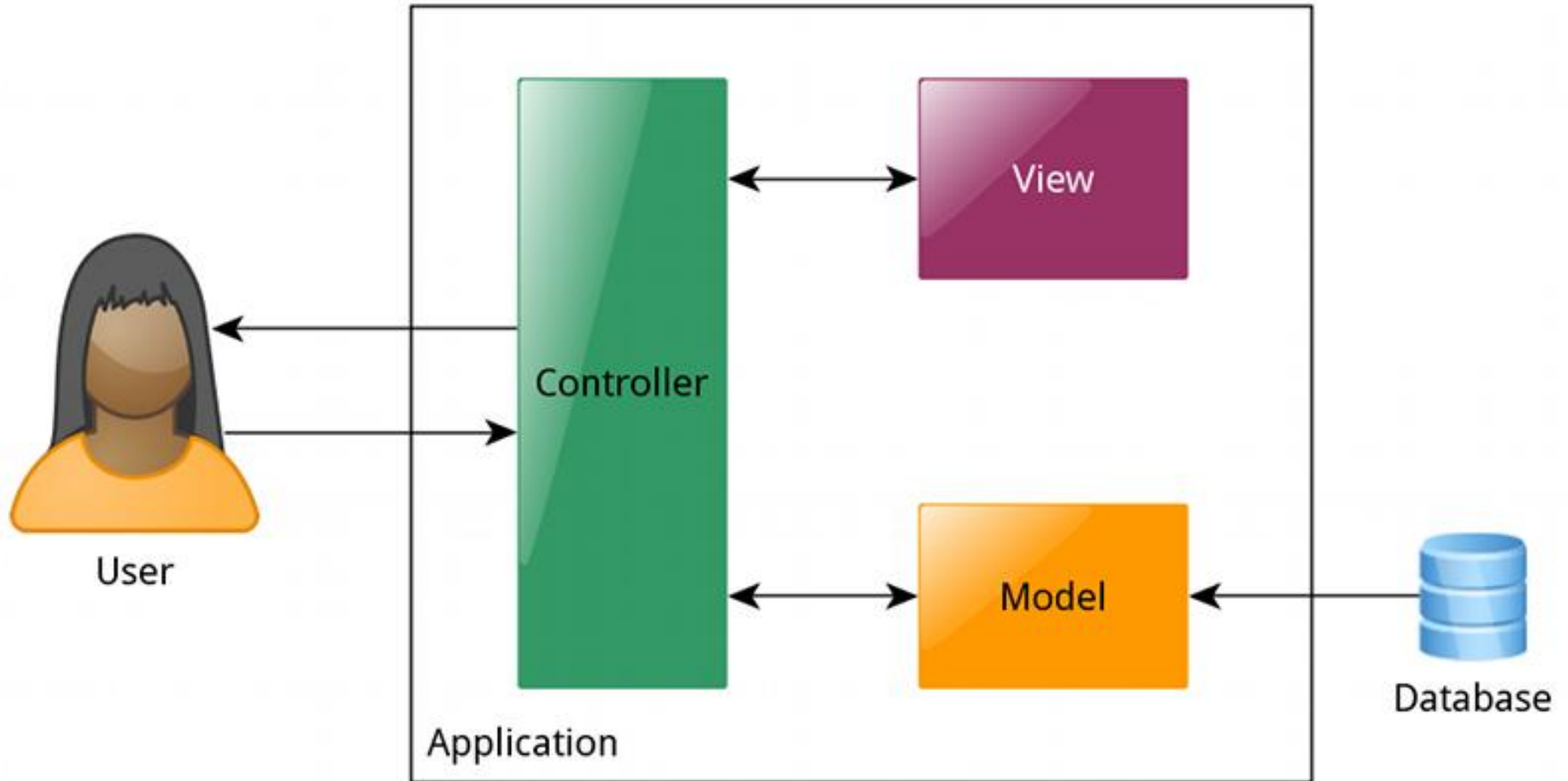
Фреймворк – это «каркас»,
программная инфраструктура,
позволяющая достаточно быстро
разработать программное решение.

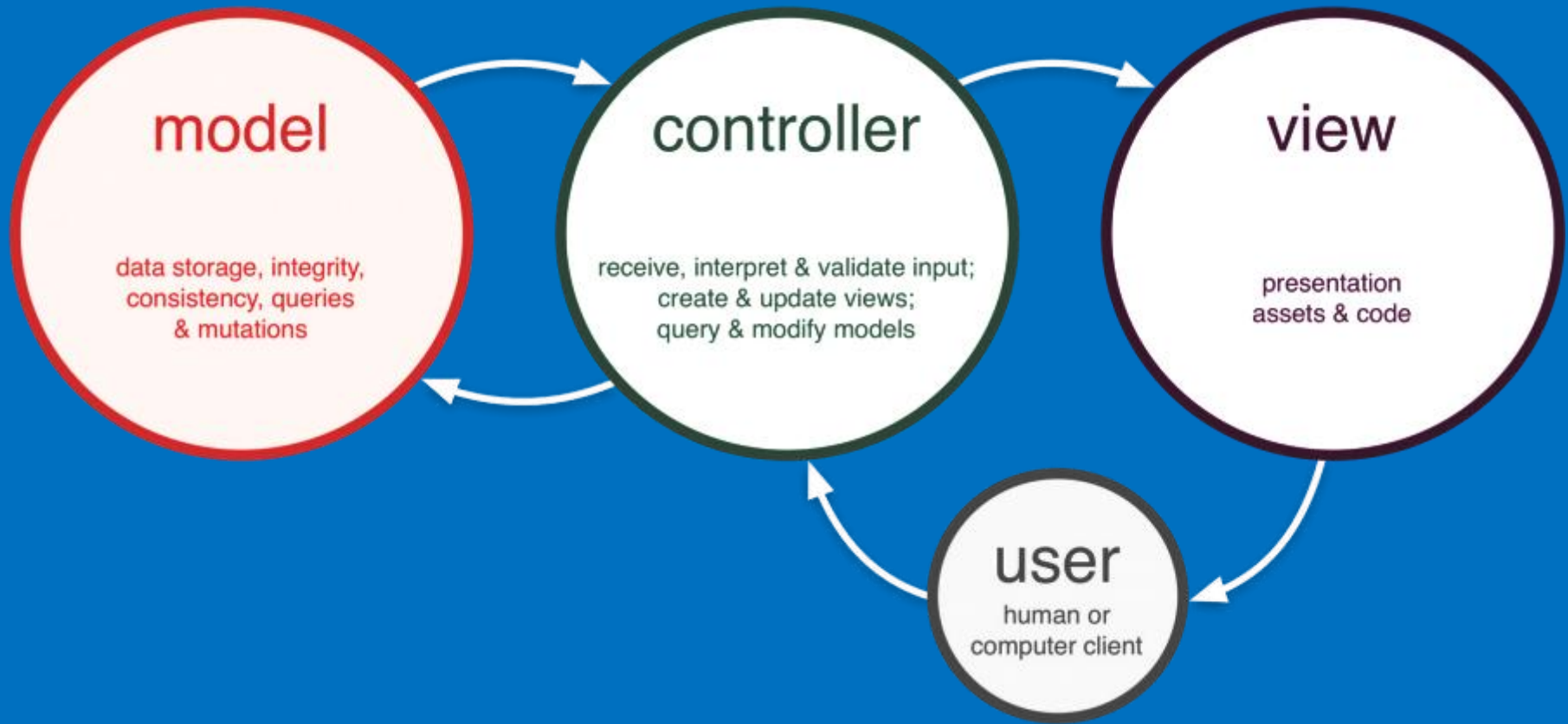
Аналогия со строительством дома...

«Фреймворк – набор программных сущностей (таких как классы, объекты и компоненты), которые помогают строить reusable архитектуру для схожих приложений. Ожидается, что для создания приложения разработчик расширит и настроит фреймворк путем добавления своей логики».

Источник: «Принципы, которые формируют успешные фреймворки» (перевод статьи Qiang Xue – создателя фреймворка Yii) <http://haru-atari.com/blog/4/philosophies-that-shaped-successful-frameworks>

MVC





Active Record

«Active Record обеспечивает объектно-ориентированный интерфейс для доступа и манипулирования данными, хранящимися в базах данных.

Класс Active Record соответствует таблице в базе данных, объект Active Record соответствует строке этой таблицы, а атрибут объекта Active Record представляет собой значение отдельного столбца строки».

Из документации Yii.

Project
\$id \$title \$content
find() insert() save() update() save()



id	title	content
1	Проект №1	Содержимое 1
2	Проект №2	Содержимое 2

2. Имеющиеся проблемы и сложности, порождаемые MVC-фреймворками

- Нарушение принципа единичной ответственности.
- Высокая связность.

Степень связанности — это мера, определяющая, насколько жестко один элемент связан с другими элементами, либо каким количеством данных о других элементах он обладает. Объект с низкой степенью связанности зависит от не очень большого числа других объектов и имеет следующие свойства:

- Малое число зависимостей между классами (подсистемами).
- Слабая зависимость одного класса (подсистемы) от изменений в другом классе (подсистеме).
- Высокая степень повторного использования подсистем.

Крэг Ларман, «Применение UML 2.0 и шаблонов проектирования».

Controller

```
public function actionAll()  
{  
    return $this->render('all', ['model' => new Project()]);  
}
```

View

```
<?php  
    foreach ($model->find()->all() as $project) {  
        echo $project->title . '<br>';  
    }  
?>
```

View

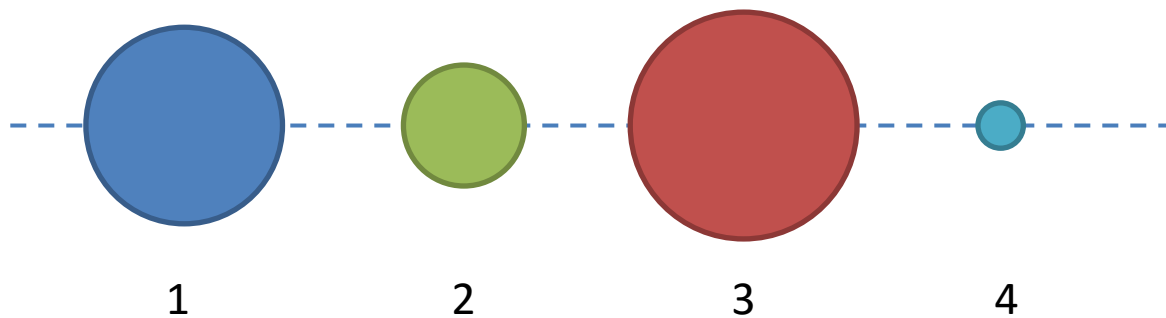
```
<?php
    foreach (Project::find()->all() as $project) {
        echo $project->title . '<br>';
    }
?>
```

```
public function actionHistory($id)
{
    $task = $this->findTask($id);
    $project = $task->project;
    $this->currentUserCanManage($project);

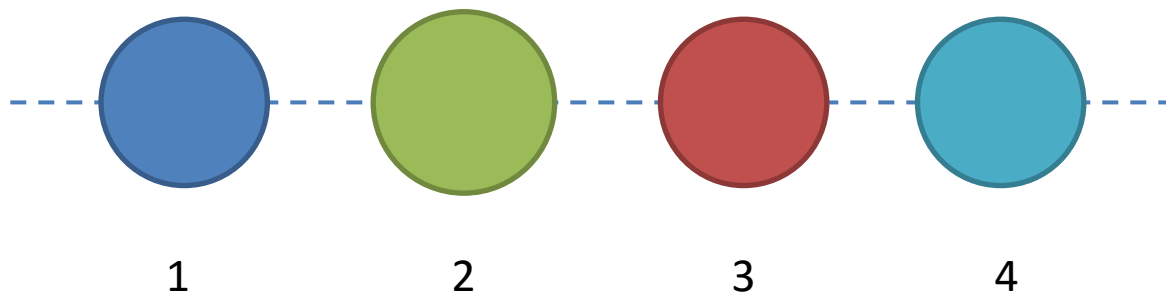
    $query = Modification::find()->confirmed()
        ->andWhere(['latest_id' => $task->id])
        ->orderBy(['created_at' => SORT_DESC]);

    $params = ArrayHelper::merge(
        ['query' => $query], Yii::$app->request->queryParams
    );
    ...
return $this->render('history', ...
}
```

На разных уровнях абстракции



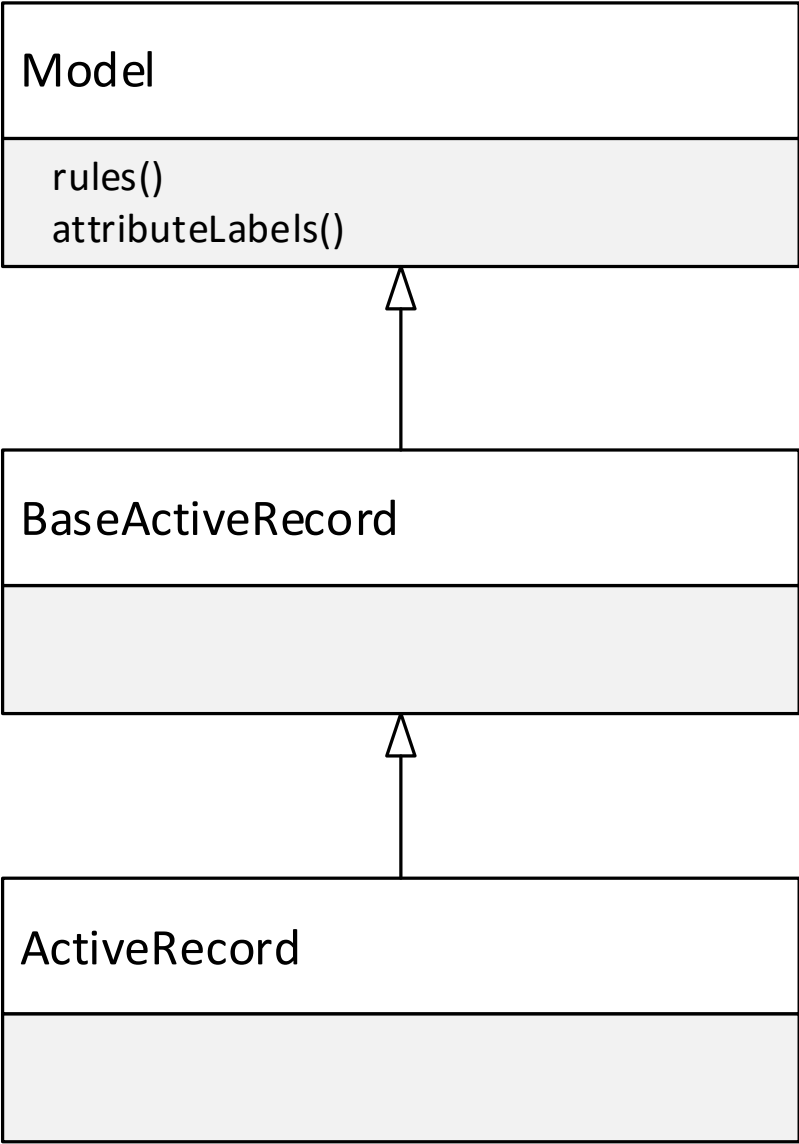
На одном уровне абстракции



```
/**  
 * @property integer $id  
 * @property string $title  
 * @property string $content  
 */
```

Model

```
class Project extends \yii\db\ActiveRecord  
{  
    public static function tableName()  
    {  
        return 'project';  
    }  
    public function rules()  
    {  
        return [  
            [['title', 'content'], 'required'],  
            [['content'], 'string'],  
            [['title'], 'string', 'max' => 255]  
        ];  
    }  
}
```

```
use app\models\Project;
```

Controller

```
public function actionView($id)
```

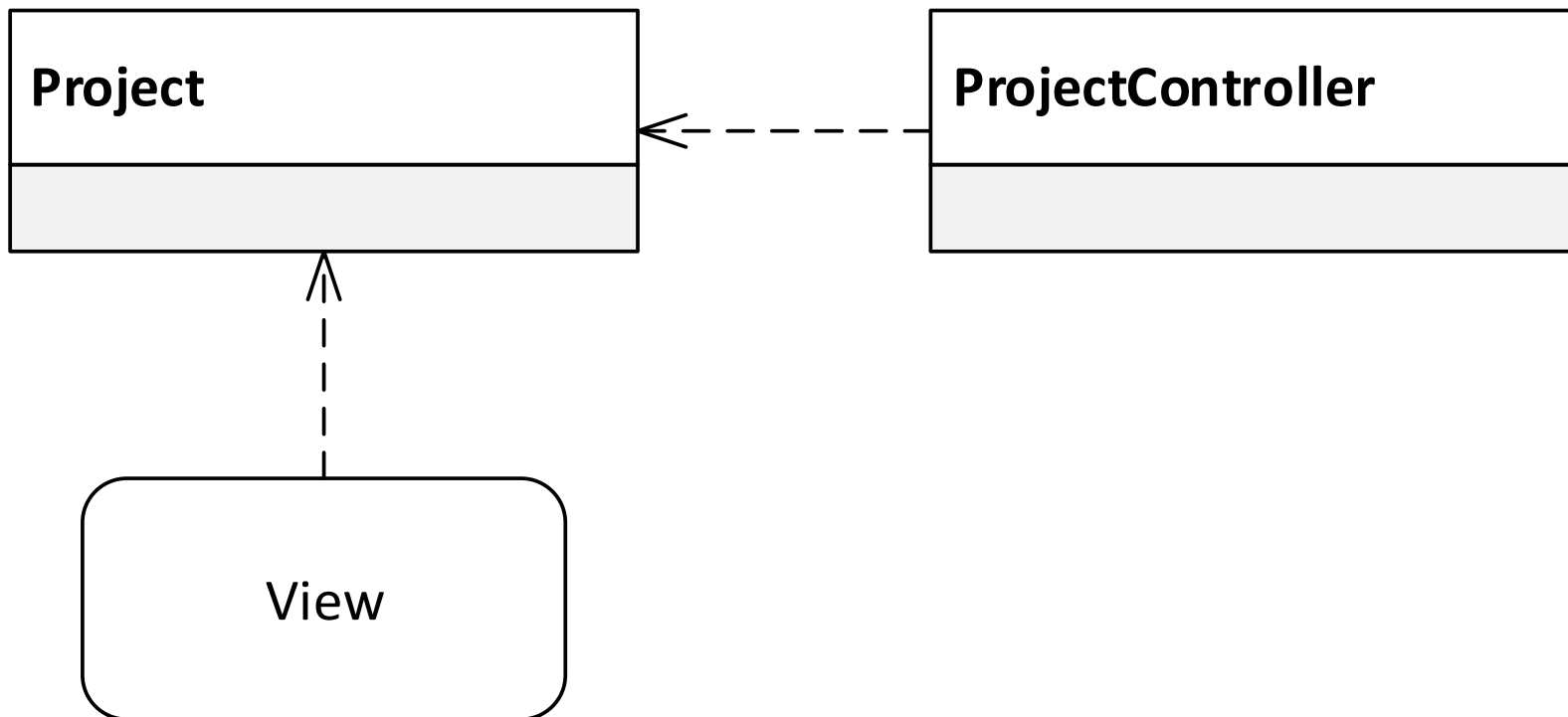
```
{  
    return $this->render('view', [  
        'model' => $this->findModel($id),  
    ]);  
}
```

```
protected function findModel($id)
```

```
{  
    if (($model = Project::findOne($id)) !== null) {  
        return $model;  
    } else {  
        throw new NotFoundException('The requested page  
does not exist.');
```

View

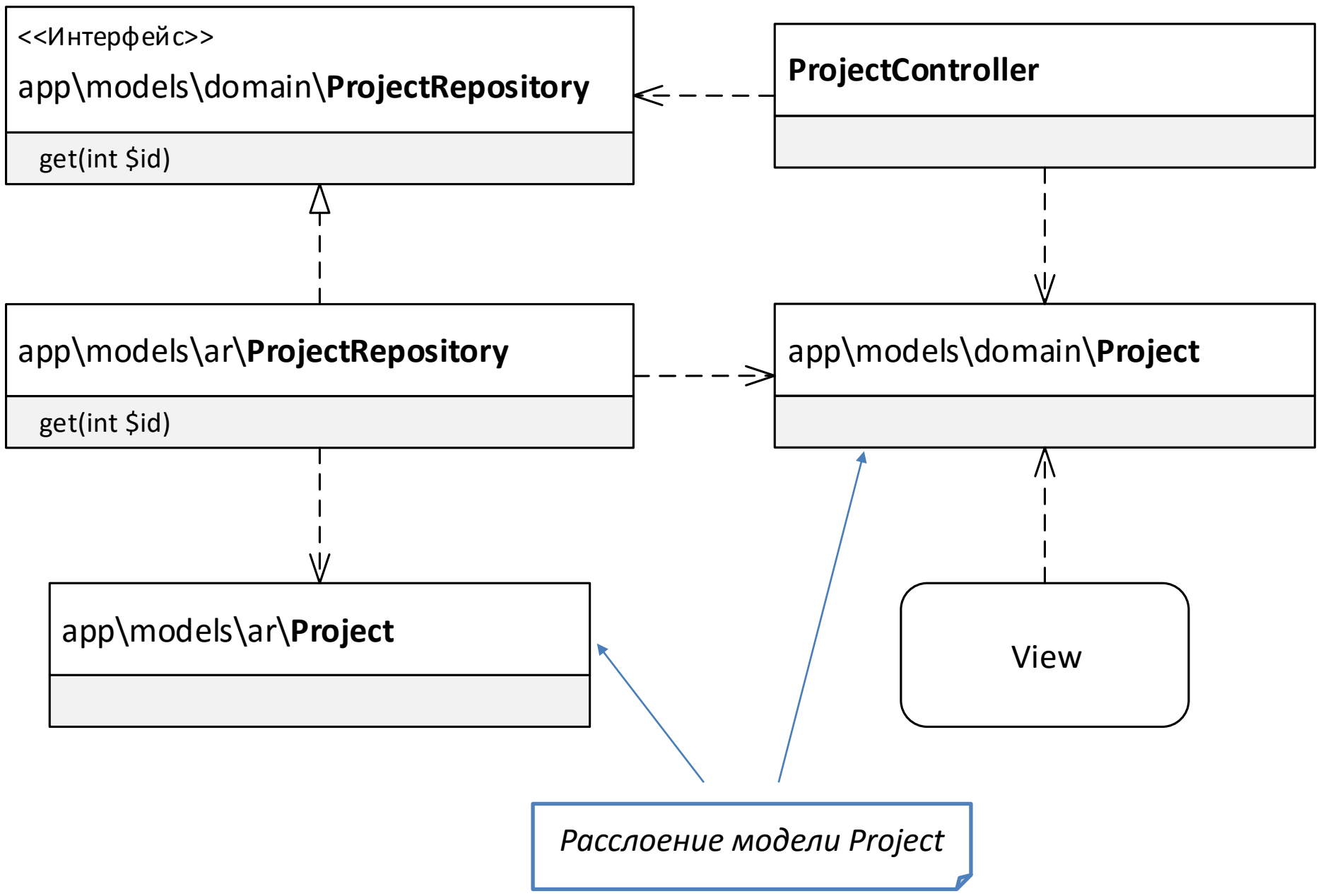
```
<h1><?=$model->title ?></h1>  
<div class="content">  
    <?=$model->content ?>  
</div>
```











3. Пути решения проблем

- Выделение слоя логики (Domain Layer).
- Разграничение ответственности.
- Внедрение зависимостей (Dependency Injection).

Демонстрационный проект...



- ▼  models
 - ▼  domain
 -  _Project.php
 -  Project.php
 -  ProjectRepository.php
 - ▼  storage
 -  Project.php
 -  ProjectRepository.php

```
/**  
 * @property integer $id  
 * @property string $title  
 * @property string $content  
 */  
class Project extends \yii\db\ActiveRecord  
{  
    public static function tableName()  
    {  
        return 'project';  
    }  
}
```



```
namespace app\models\domain;
```

```
trait _Project {  
    public $id;  
    public $title;  
    public $content;  
}
```

```
namespace app\models\domain;  
use yii\base\Model;
```

```
class Project extends Model
```

```
{
```

```
    use _Project;
```

```
    use \app\utility\Replicate;
```

```
    public function rules()
```

```
{
```

```
        return [
```

```
            [['title', 'content'], 'required'],
```

```
            [['content'], 'string'],
```

```
            [['id'], 'integer'],
```

```
            [['title'], 'string', 'max' => 255]
```

```
        ];
```

```
}
```

```
use app\models\domain\ProjectRepository;
```

```
class ProjectController extends Controller
```

```
{
```

```
    private $repository;
```

```
    public function __construct($id, $module,  
                               ProjectRepository $repository, $config = [])
```

```
{
```

```
    parent::__construct($id, $module, $config = []);
```

```
    $this->repository = $repository;
```

```
}
```

```
...
```

```
}
```

...

```
Yii::$container->set('app\models\domain\ProjectRepository',  
                    'app\models\storage\ProjectRepository');
```

```
(new yii\web\Application($config))->run();
```

```
use app\models\domain\Project;
```

```
ProjectController.php
```

```
public function actionView($id)
```

```
{
```

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

```
}
```

```
protected function findModel($id) : Project
```

```
{
```

```
    $model = $this->repository->get($id);
```

```
    if ($model !== null) {
```

```
        return Project::createObject($model);
```

```
    } else {
```

```
        throw new NotFoundException(
```

```
            'The requested page does not exist.');
```

```
    }
```

```
}
```

```
namespace app\models\domain;
```

```
interface ProjectRepository
```

```
{
```

```
    function get(int $id);
```

```
    function create(Project $project) : int;
```

```
    function save(Project $project) : bool;
```

```
    function delete(int $id) : bool;
```

```
}
```

```
namespace app\models\storage;
```

```
use app\models\domain as domain;
```

```
class ProjectRepository implements domain\ProjectRepository
```

```
{
```

```
    public function get(int $id)
```

```
    {
```

```
        $project = Project::findOne($id);
```

```
        return !is_null($project)
```

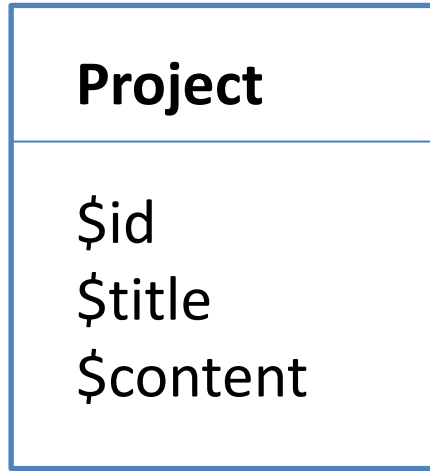
```
            ? domain\Project::createObject($project) : null;
```

```
    }
```

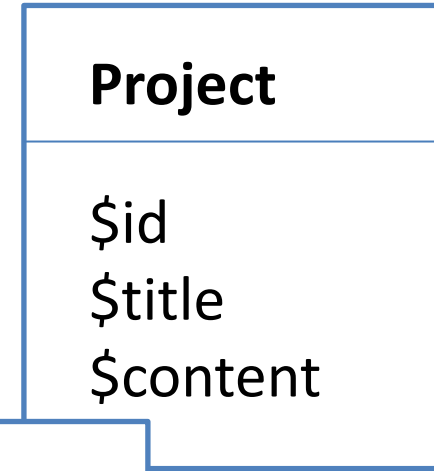
```
    ...
```

```
}
```

models\storage\Project.php



models\domain\Project.php



trait Replicate

utility\Replicate.php

```
{  
    static private $domainTrait = "";  
    static private $domainNamespace = 'app\models\domain';  
  
    static function createObject($copiedObject)  
    {  
        $currentClass = get_called_class();  
        $newObject = new $currentClass;  
        self::copyObject($newObject, $copiedObject);  
        return $newObject;  
    }  
    static function copyObject($toObject, $fromObject)  
    {  
        $properties = get_class_vars(self::pattern());  
        foreach ($properties as $name => $value) {  
            $toObject->$name = $fromObject->$name;  
        }  
    }  
}
```

```
static private function pattern()  
{  
    if (!empty(static::$domainTrait)) {  
        return static::$domainTrait;  
    }  
    self::setDomainTraitDefault();  
    return static::$domainTrait;  
}
```

```
static private function setDomainTraitDefault()  
{  
    $reflection = new \ReflectionClass(get_called_class());  
    $name = $reflection->getShortName();  
    static::$domainTrait = '\\'. self::$domainNamespace  
        . '\\_' . $name;  
}  
}
```

```
public function get(int $id)
```

```
{
```

```
    $project = Project::findOne($id);
```

```
    return !is_null($project)
```

```
        ? domain\Project::createObject($project) : null;
```

```
}
```

```
public function save(domain\Project $project) : bool
```

```
{
```

```
    $projectStorage = Project::findOne($project->id);
```

```
    domain\Project::copyObject($projectStorage, $project);
```

```
    return $projectStorage->save();
```

```
}
```

4. Плюсы и минусы подхода

Спасибо за внимание!

Валерий Чугреев

vk.com/chugreev