Разработка на Yii

Системный архитектор
Климов П.В.

QuartSoft Corp.

Yii – PHP Framework

Основные характеристики:

- ООП
- Модульность
- Простота
- Высокое быстродействие

Истоки Yii:

- Prado
- Ruby on Rails
- jQuery
- Symfony
- Joomla

Магия в PHP

```
class Component {
    public $publicProperty;
    protected $_protectedProperty;

    public function setProtectedProperty($value) {
        $this->_protectedProperty = $value;
        return true;
    }

    public function getProtectedProperty() {
        return $this->_protectedProperty;
    }
}
```

```
class Component {

    public function __get($propertyName) {
        $methodName = 'get'.$propertyName;
        if (method_exists($this, $methodName)) {
            return call_user_func( array($this, $methodName) );
        } else {
            throw new Exception("Missing property {$propertyName}'!");
        }
    }

    public function __set($propertyName, $value) {
        $methodName = 'set'.$propertyName;
        if (method_exists($this, $methodName)) {
            return call_user_func( array($this, $methodName), $value );
        } else {
            throw new Exception("Missing property {$propertyName}'!");
        }
    }
```

```php
$component = new Component();

$component->publicProperty = 'Public value';
echo($component->publicProperty);

$component->protectedProperty = 'Protected value';
echo($component->protectedProperty);
```

Автозагрузка классов

Подключение файлов по принципу DLL:

```
require_once('components/SomeClass.php');
$someObj = new SomeClass();
…
require_once('components/OtherClass.php');
$otherObj = new OtherClass();
…
require_once('components/SomeClass.php');
$anotherSomeObj = new SomeClass();
```

```
class Autoloader {
    public function autoload($className) {
        $classFileName = ‘components/'.$className.'.php';
        if (file_exists($classFileName)) {
            require_once($classFileName);
            return true;
        }
        return false;
    }

    public function register() {
        return spl_autoload_register( array($this, 'autoload') );
    }

    public function __construct() {
        $this->register();
    }
}
```

Автозагрузка классов в контексте Yii:

```
Yii::import(‘application.components.SomeClass');
Yii::import(‘application.components.OtherClass');
…
$someObj = new SomeClass();
```

«Карта» автозагрузки классов:

‘SomeComponent’ => ‘/home/www/…/components/SomeClass.php’,
‘OtherComponent’ => ‘/home/www/…/components/OtherClass.php’,

Порождение компонентов

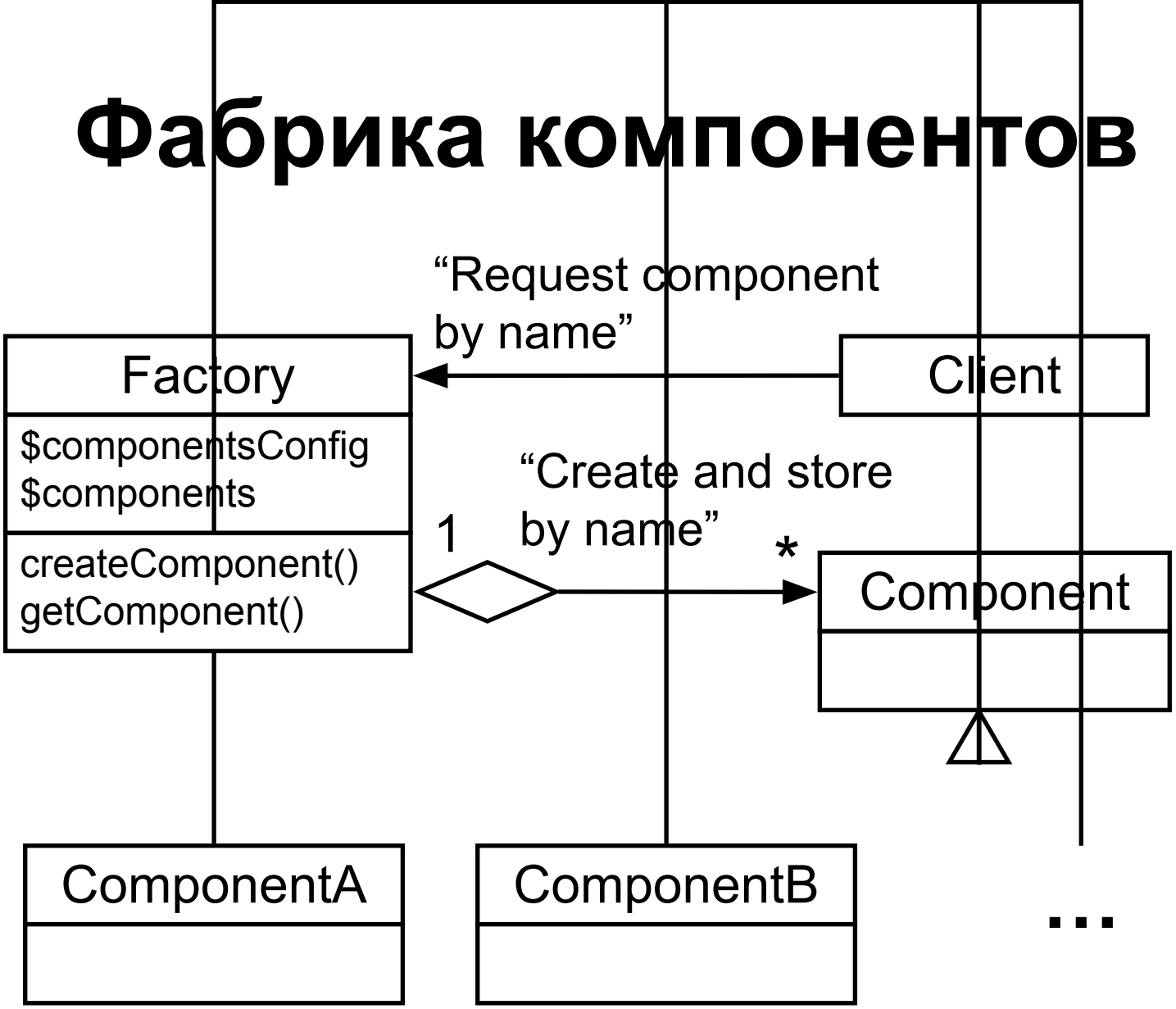
```
function createComponent(array $componentConfig) {
    $className = $componentConfig['class'];
    if (empty($className)) {
        throw new Exception('Missing parameter "class"!');
    }
    unset($componentConfig['class']);
    if (!class_exists($className)) {
        Yii::import($className); // Автозагрузка
    }
    $component = new $className();
    foreach($componentConfig as $name=>$value) {
        $component->$name = $value; // Конфигурация
    }
    return $component;
}
```

Задание любого объекта через массив:

```
$componentConfig = array(
    'class'=>'CUrlManager',
    'urlFormat'=>'path',
    'showScriptName'=>false,
    'rules'=>array(
        '/'=>'site/index',
        '<controller:\w+>/<id:\d+>*'=>'<controller>/view',
    ),
);

$component = createComponent($componentConfig);
```

Фабрика компонентов



Одиночка (Singleton)

```
class Singleton {
    private static $_selfInstance = null;

    public static function getInstance() {
        if (!is_object(self::$_selfInstance)) {
            self::$_selfInstance = new Singleton();
        }
        return self::$_selfInstance;
    }

    private function __construct() {
        // закрытый конструктор
    }
}

$singleton = Singleton::getInstance();
```

Фабрика компонентов(Component Factory)

+

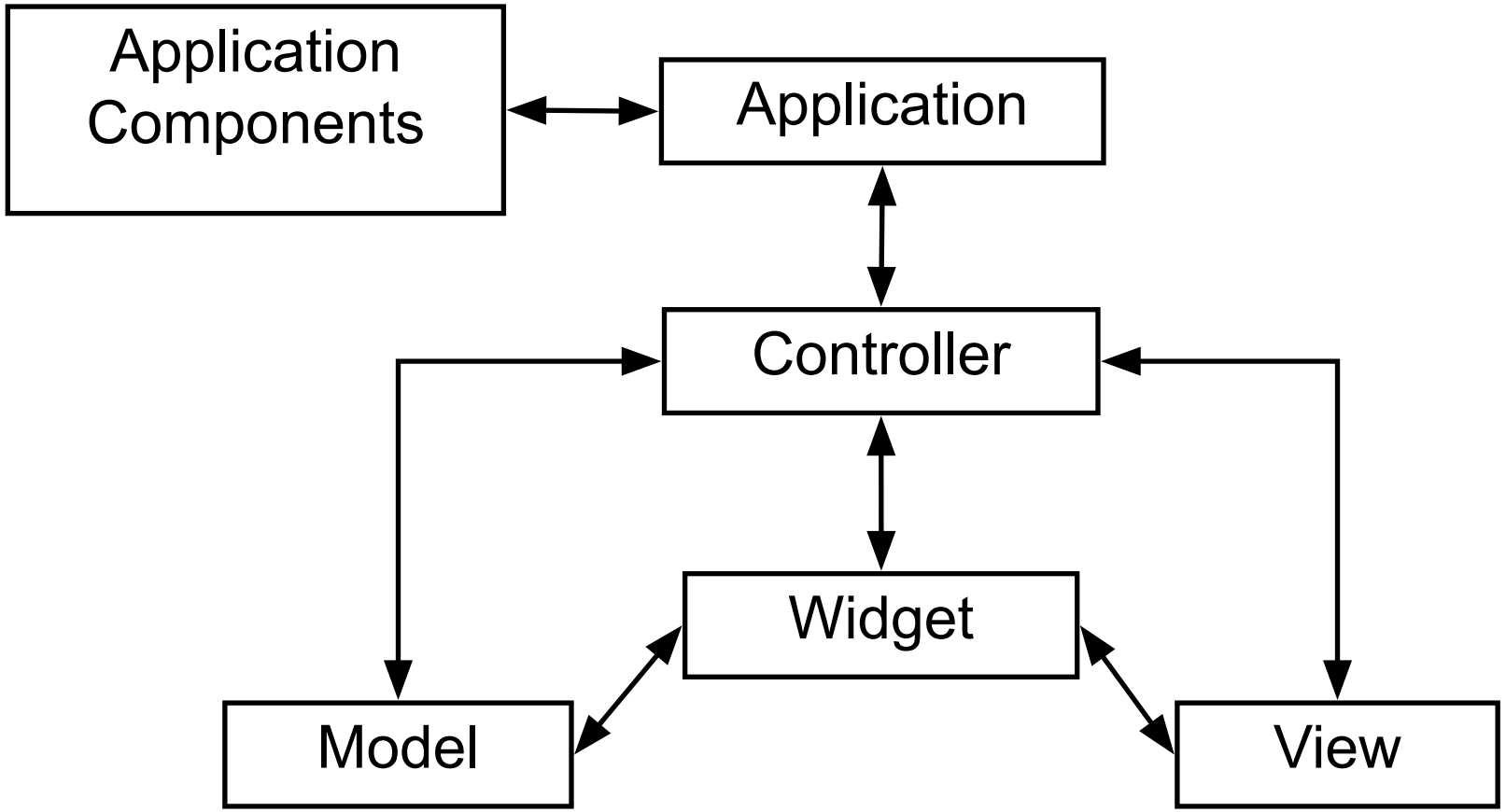
Одиночка (Singleton)

=

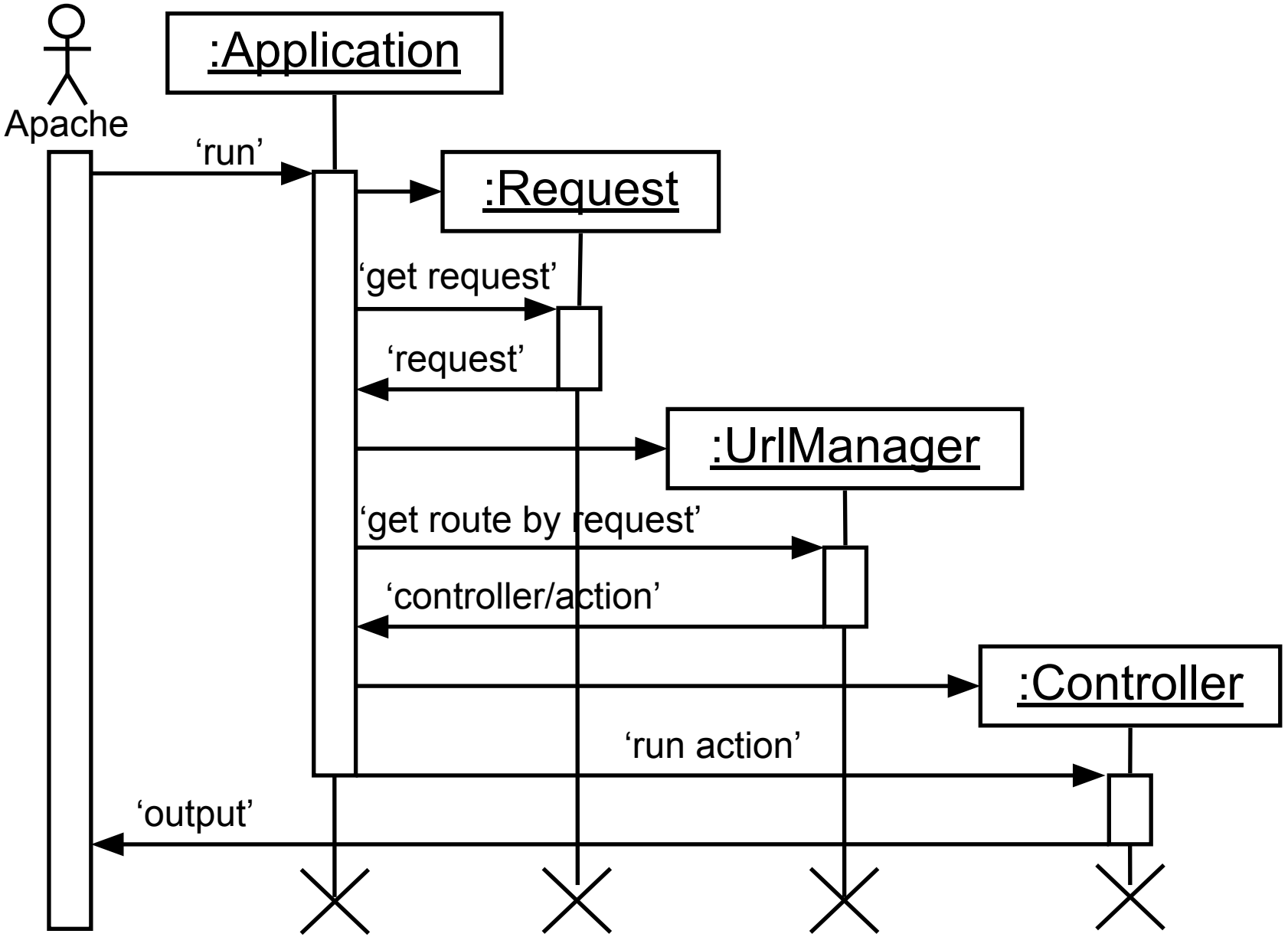
Приложение Yii (Yii Application)

```
$config = array(
    'name'=>'My Web Application',
    …
    'components'=>array(
     'user'=>array(
            'allowAutoLogin'=>true,
     ),
          …
    ),
);
Yii::createWebApplication($config)->run();
…
$application = Yii::app();
$user = Yii::app()->getComponent(‘user’);
```

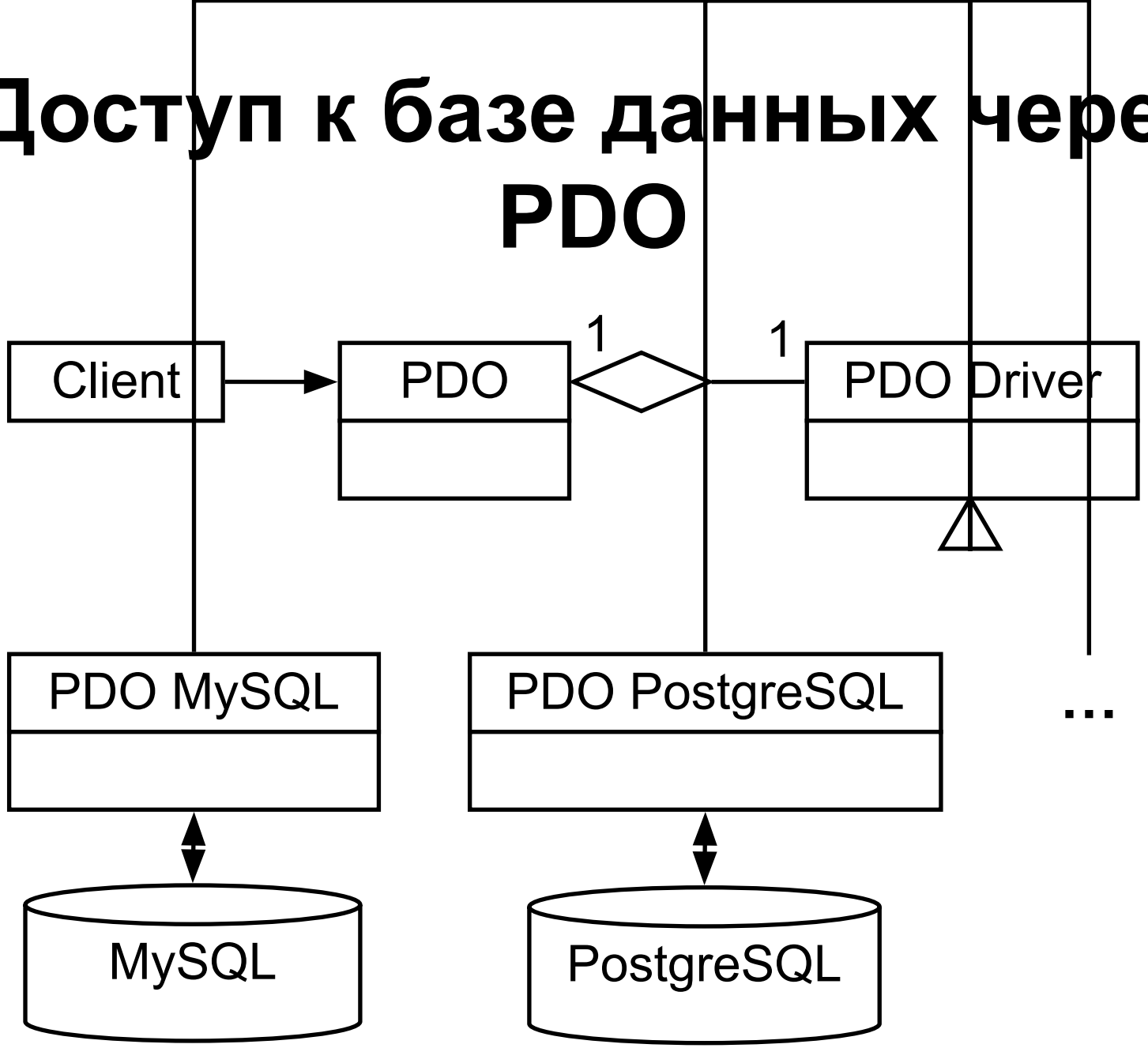
MVC в Yii


Application Components
Application
Controller
Widget
Model
View

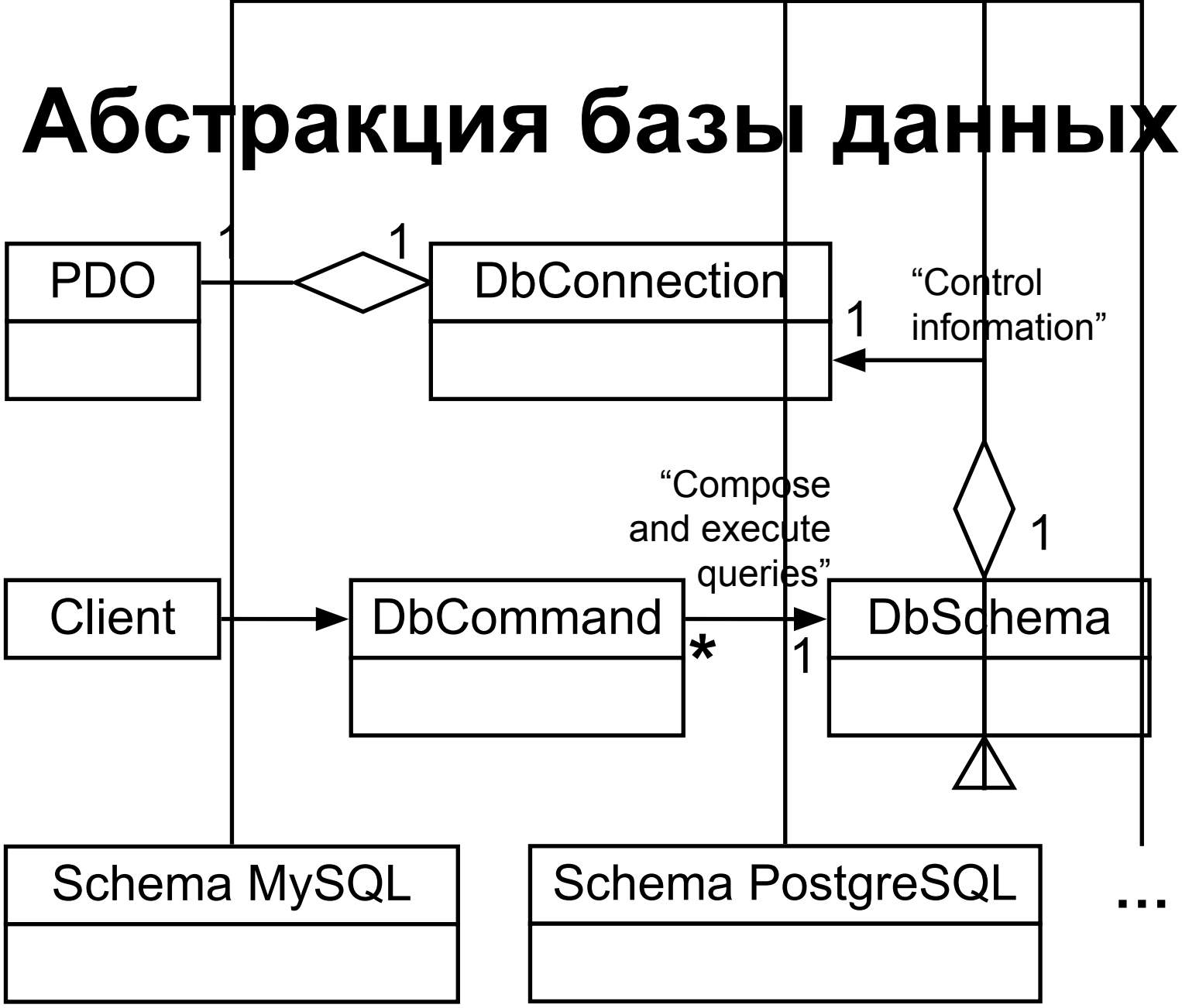
Маршрутизация web запроса



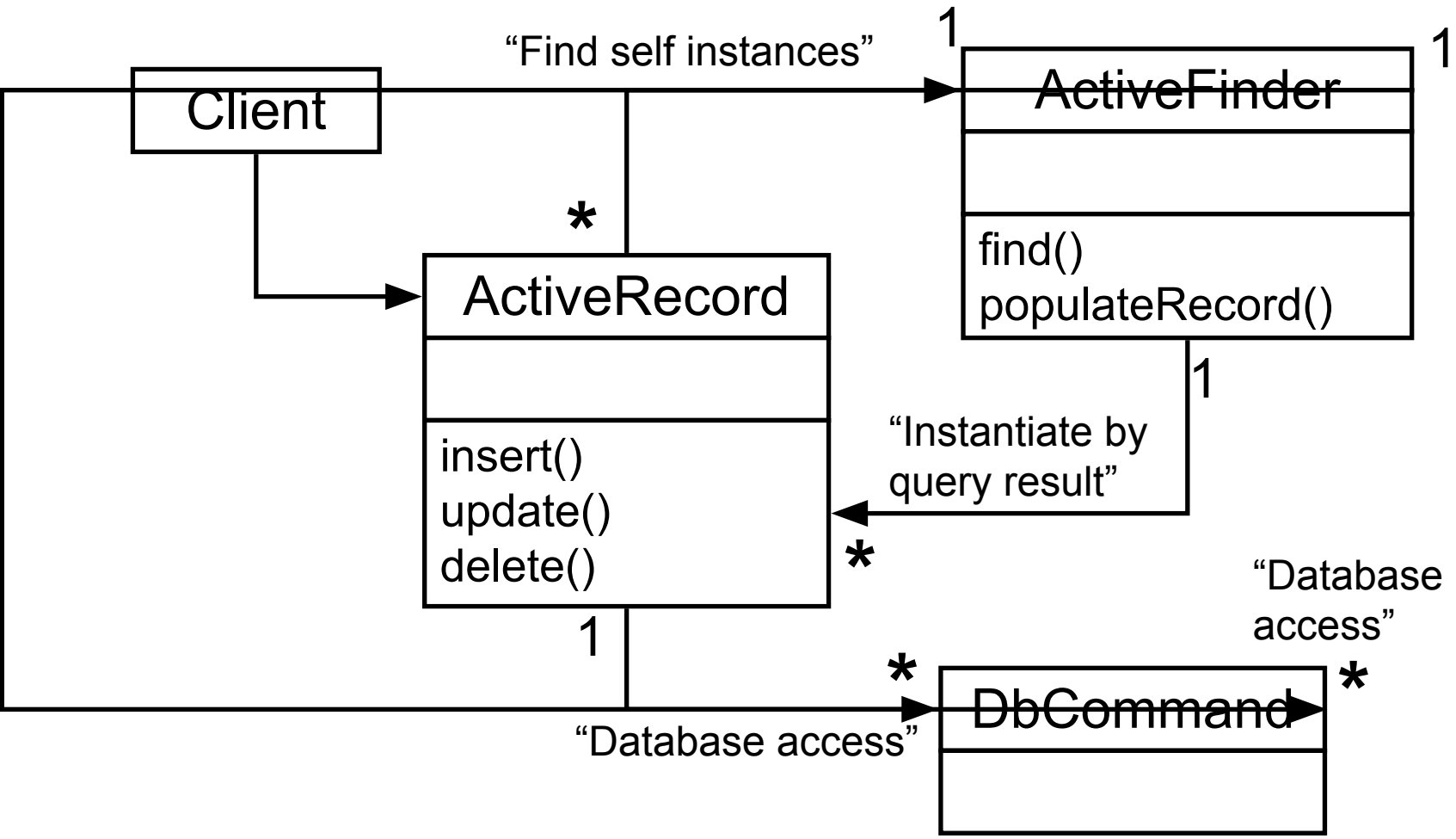
Доступ к базе данных через PDO



Абстракция базы данных



Active Record

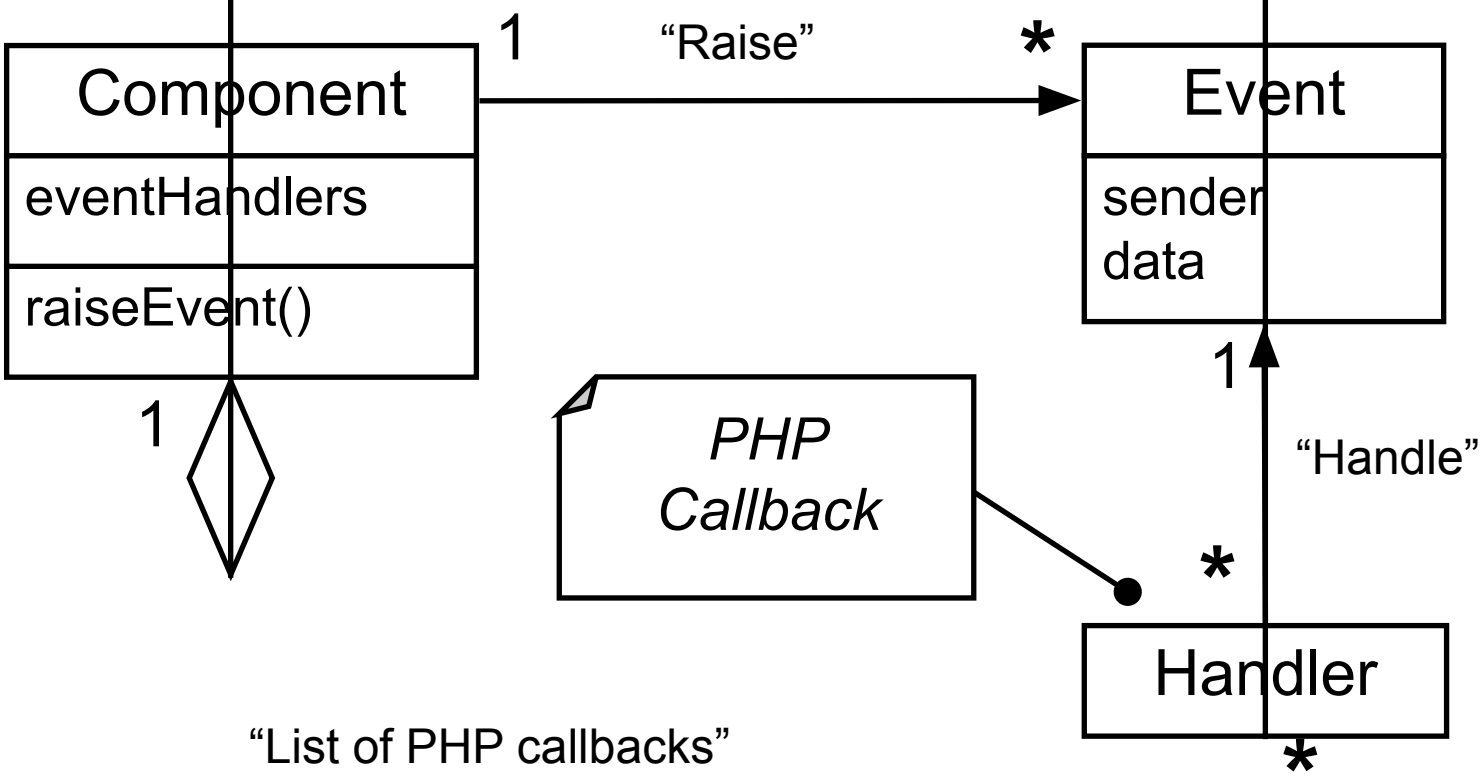


```
$allUsers = User::model()->findAll();

$newUser = new User();
$newUser->name = ‘new user’;
$newUser->save();

$existingUser = User::model()->findByName(‘testuser’);
$existingUser->email = ‘newemail@domain.com’;
$existingUser->save();
```

События (Events) в Yii



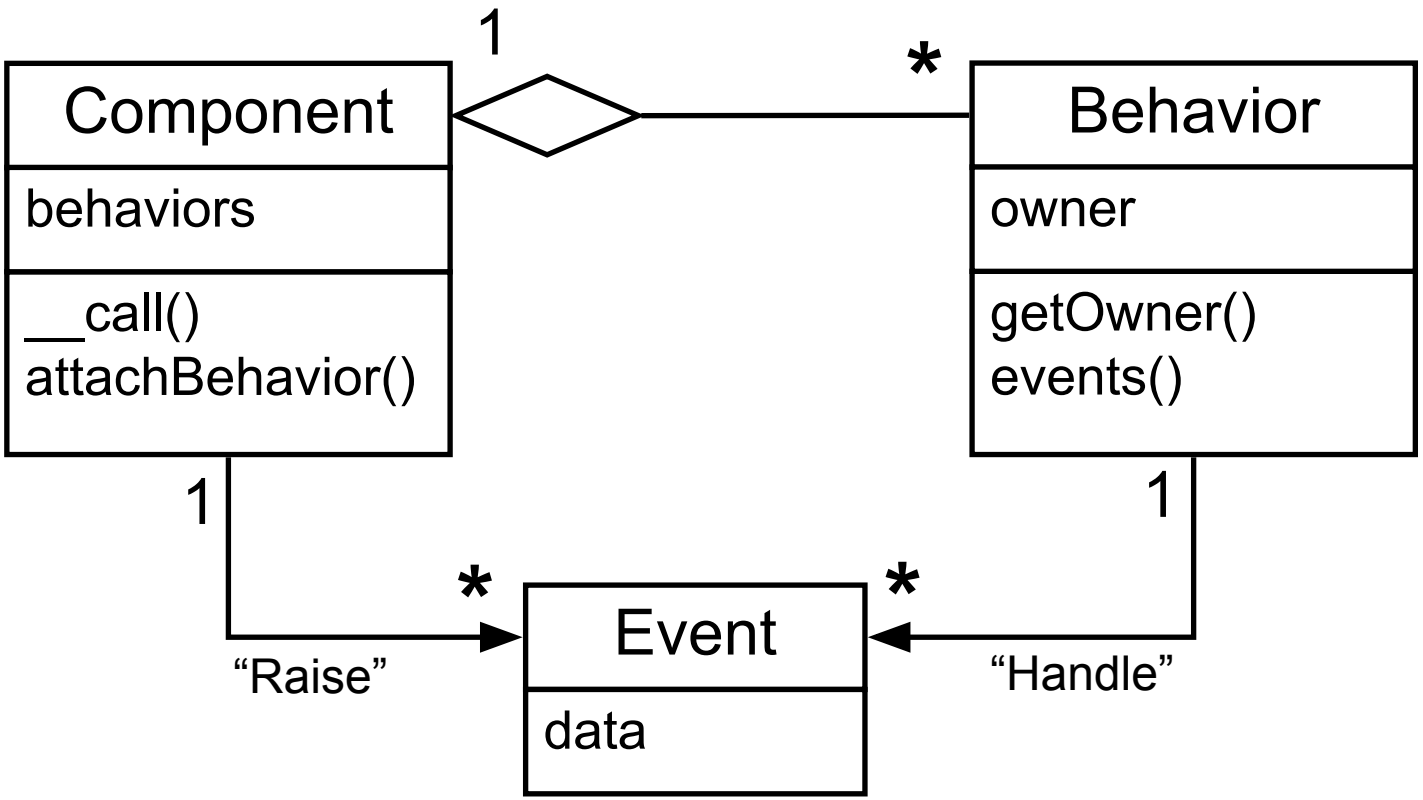
```
function handleBeforeSave(CEvent $event) {
    $sender = $event->sender;
    // Изменяем состояние отправителя события:
    $sender->create_date = date('Y-m-d H:i:s', strtotime('NOW'));
}

$user = new User();
// Назначаем обработчик события:
$user->onBeforeSave = ‘handleBeforeSave’;
$user->name = ‘test name’;
$user->save();

echo $user->create_date; // Вывод: ‘2012-03-22 16:42’
```

Проблема множественного наследования

Поведение (Behavior)



```php
class ArBehaviorExample extends CBehavior {
    public function behaviorMethod() {
        $owner = $this->getOwner();
        $owner->create_date = date('Y-m-d H:i:s', strtotime('NOW'));
    }
}

$user = new User();
// Добавляем поведение:
$behavior = new ArBehaviorExample();
$user->attachBehavior($behavior);

// Вызываем метод поведения:
$user->behaviorMethod();
echo $user->create_date; // Вывод: '2012-03-22 16:46'
```

Yii

- Динамический код
- Компонентная структура
- Приложение = «одиночка» + «фабрика»
- Отложенная загрузка и создание объектов
- MVC
- «PDO» и «Active Record»
- События
- Поведения