

# **Занятие №5**

## **MVC, Symfony**

# О себе

- Александр Козлов
  - в Симбирсофт с декабря 2010 г.
  - отдел веб-разработок
  - тимлид, руководитель проектов
- Образование:
  - УлГУ, Прикладная математика и информатика, 2009 г. (диплом специалиста)
  - МИМ ЛИНК, курс «Управление организацией и персоналом», 2013 г. (сертификат)
- Опыт использования технологий
  - LAMP – 5 лет
  - Symfony 1.x/2.x – 2.5 года
  - Yii framework 1.x – 0.5 года
- Проекты (PHP): dzangocart, dzangowiki, travelrecords, teamogen, practicethai, fuseinsurance

# Что такое фреймворк?

# Фреймворк

Фреймворк (framework) - это каркас, облегчающий разработку и объединение разных компонентов большого программного проекта.

# Фреймворк

- готовые компоненты и решения
- основы безопасности
- стандартизация
- сообщество

# PHP фреймворки

# PHP фреймворки

- Symfony
- Zend
- Yii
- ...

# Фреймворк

- Когда использовать?
- “Простые” и “сложные” фреймворки
- CMF и CMS



# MVC

# MVC

- Модель (Model)
- Представление (View)
- Контроллер (Controller)

# MVC

MVC предоставляет возможность отделения друг от друга слоя данных, представления и логики поведения.

# Модель

# Модель

- данные и методы работы с этими данными
- реагирует на запросы, изменяя своё состояние
- не содержит информации о том, как можно визуализировать данные

# Модель

Что внутри?

- описание структуры данных
- взаимодействие с хранилищем (СУБД)
- бизнес-логика\*

# Модель

```
class Attachment extends BaseEntity
{
    private $id;
    private $attachmentType;
    private $post;

    public function __construct(User $user, Post $post) {
        $this->setUploadDir('/uploads/'. $user->getId(). '/' . $post->getId(). '/');
    }

    public function setPost(Post $post = null) {
        $this->post = $post;

        return $this;
    }

    public function getPost() {
        return $this->post;
    }

    ...
}
```

# **Представление**



# Представление

Отвечает за отображение информации  
(визуализацию)

# Представление

Что внутри?

- Каркас формата (html, xml, json)
- Вывод данных
- Минимальный набор операторов  
(шаблонизатор)

# Представление

```
<div class="all-comments">
  <? foreach ($comments as $comment): ?>
    <div class="comment">
      <a href="/user/<? echo $comment->authorId; ?>">
        
        <div class="name"><? echo $comment->authorName; ?></div>
      </a>

      <span class="date"><? echo $comment->createdAt; ?></span>
      <div class="text"><? echo $comment->message; ?></div>
    </div>
  <? endforeach; ?>
</div>
```

# Контроллер

# Контроллер

- контролирует ввод данных пользователем
- использует модель и представление для реализации необходимой реакции

# Контроллер

Что внутри?

- получение данных от пользователя
- передача данных в модель
- получение представления на основе  
текущего состояния модели, вывод  
результатов пользователю

# Контроллер

```
class UserController extends BaseController
{
    public function indexAction() {
        $params = $this->_getParams();
        $params['activeMenuItem'] = self::USER_MENU_ITEM;

        return $params;
    }

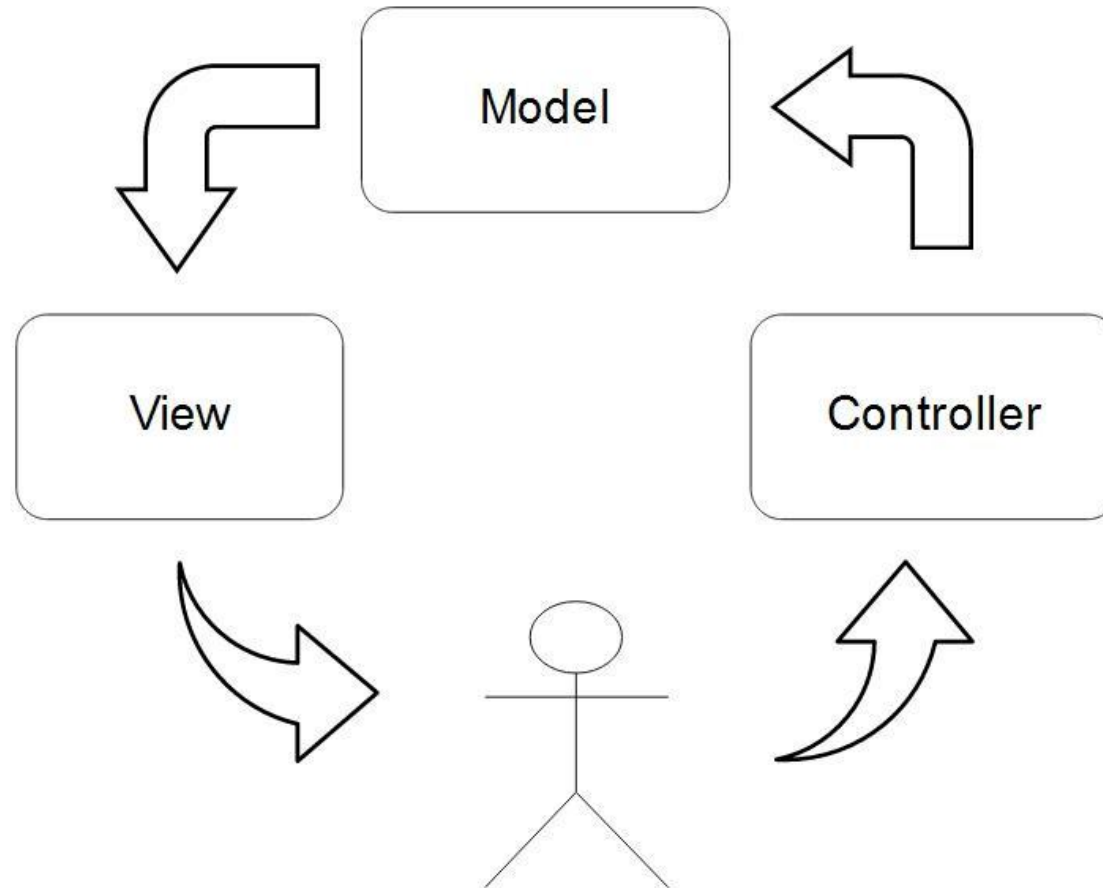
    public function showAction($id) {
        $user = $this->getRepository('user')->find($id);

        if (!$user || $user->isDeleted()) {
            return $this->generateError('user.notFound');
        }

        return $this->render('Main:index', array('user' => $user));
    }

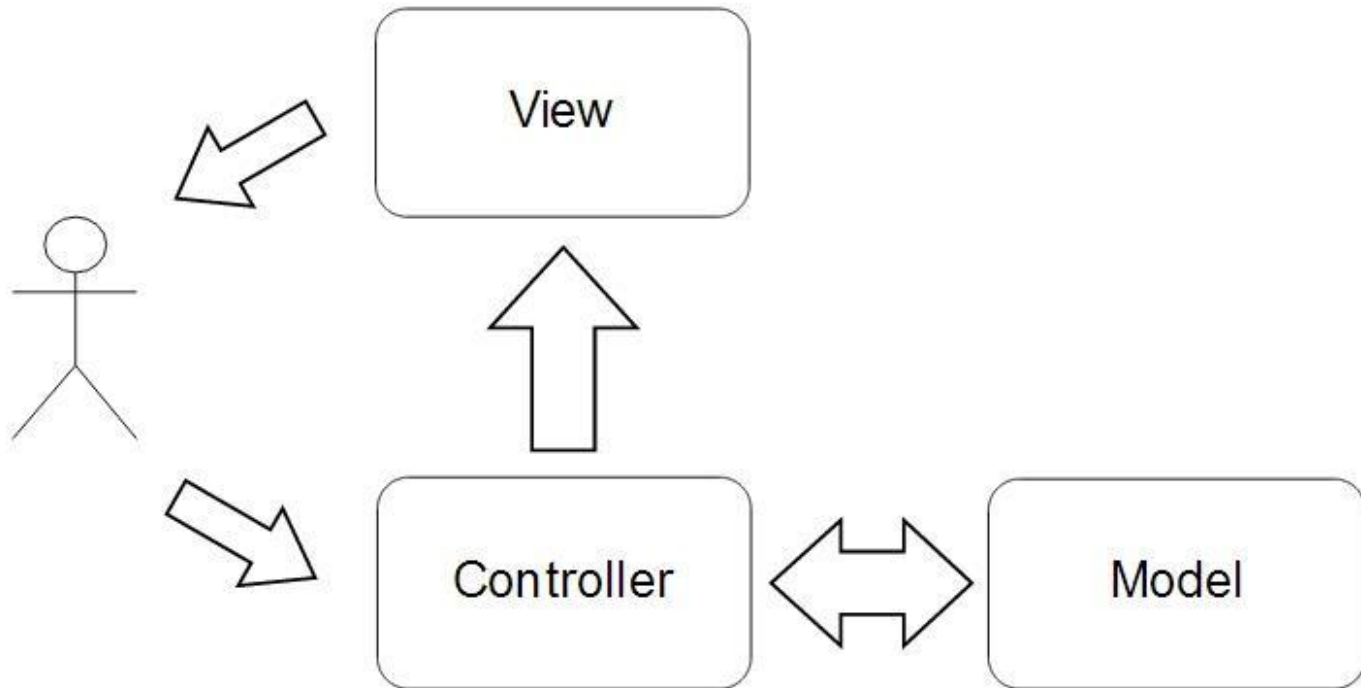
    ...
}
```

# Варианты реализации MVC





# Варианты реализации MVC



# Вспомогательные компоненты

- Маршрутизатор
- Сервисы

# Маршрутизатор

# Маршрутизатор

Основное назначение - определение контроллера, которому будет передано управление.

# Маршрутизатор

```
user_list:  
  pattern: /user  
  defaults: {_controller: MainBundle::User:index}  
  
user_show:  
  pattern: /user/{id}  
  defaults: {_controller: MainBundle::User:show}
```

# Сервисы

# Сервисы

- выполняют “глобальные” задачи
- как правило, доступны внутри контроллеров

# Сервисы

```
class UserController extends BaseController
{
    public function sendAction($id, $message) {
        $user = $this->getRepository('user')->find($id);
        if (!$user || $user->isDeleted()) {
            return $this->generateError('user.notFound');
        }

        $mailer = $this->get('mailer');

        return $mailer->send($user->getEmail(), $message);
    }

    ...
}
```



# Типы контроллеров

- frontend
- command (CLI)
- API

# Frontend

Типовые задачи:

- вывод html-страниц (или отдельных блоков)
- обработка запросов отправки форм

# Frontend

```
class UserController extends BaseController
{
    public function showAction($id) {
        $user = $this->getRepository('user')->find($id);

        if (!$user || $user->isDeleted()) {
            return $this->generateError('user.notFound');
        }

        return $this->render('Main:index', array('user' => $user));
    }

    ...
}
```

# Command

Типовые задачи:

- Импорт/экспорт данных
- Фоновые службы
- ?

# Command

```
class LdapImportUsersCommand extends ContainerAwareCommand
{
    protected function execute() {
        $userRepository = $this->getContainer()->get('user.repository');
        $ldapManager = $this->getContainer()->get('ldap.ldap_manager');

        $ldapUsers = $ldapManager->getUsers();

        foreach ($ldapUsers as $ldapUser) {
            $user = new User();
            $user->setUsername($ldapUser['username'][0]);
            $user->setName($ldapUser['name'][0]);
            $user->setSurname($ldapUser['surname'][0]);

            $userRepository->save($user);

            $output->writeln($ldapUser['username'][0]);
        }

        $output->writeln('done');
    }
}
```

# API

Типовые задачи:

- Взаимодействие с внешними приложениями
  - Интеграция с компонентами приложения
- через AJAX

# API

```
class APIController extends BaseAPIController
{
    public function userAction($id) {
        $user = $this->getRepository('user')->find($id);

        if (!$user || $user->isDeleted()) {
            return $this->generateError('user.notFound');
        }

        return json_encode(array('user' => $user));
    }

    ...
}
```

# Варианты реализации модели

- ActiveRecord
- ORM



# Шаблонизаторы

- PHP
- Twig
- Smarty
- XSLT

# Symfony

# Почему symfony?

- КОМПОНЕНТНЫЙ ПОДХОД
- MVC, DI, IoC, ORM, ...
- сообщество
- composer
- Twig

# Symfony: инструментарий

- LAMP (php 5.3+)
- git
- composer

# Symfony: установка

```
composer create-project symfony/framework-standard-  
edition /projects/sf/ "2.5.*"
```

# Symfony: установка

```
Would you like to install Acme demo bundle? [y/N] y
Installing the Acme demo bundle.
Creating the "app/config/parameters.yml" file
Some parameters are missing. Please provide them.
database_driver (pdo_mysql):
database_host (127.0.0.1):
database_port (null):
database_name (symfony):
database_user (root): symfony
database_password (null): 123
mailer_transport (smtp):
mailer_host (127.0.0.1):
mailer_user (null):
mailer_password (null):
locale (en):
secret (ThisTokenIsNotSoSecretChangeIt):
debug_toolbar (true):
debug_redirects (false):
use_assetic_controller (true):
```

# Symfony: установка

```
<VirtualHost *:80>
    ServerName symfony
    ServerAlias symfony

    DocumentRoot /projects/sf/web
    <Directory /projects/sf/web>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
        Order allow,deny
        allow from all
    </Directory>
</VirtualHost>
```

# Symfony: установка

```
cd /projects/sf/
```

```
chmod -R 777 app/logs/
```

```
chmod -R 777 app/cache/
```

**Добавляем в файл /etc/hosts запись:**

```
127.0.0.1    symfony
```

**Перезапускаем apache:**

```
sudo service apache2 restart
```



# **Symfony: установка**

[http://symfony/app\\_dev.php](http://symfony/app_dev.php)

# Symfony: установка


The screenshot shows the Symfony 2.5.3 installation welcome screen. At the top left is the Symfony logo and name. At the top right is a search bar with the text "Search on Symfony website" and an "OK" button. The main content area has a large "Welcome!" heading, followed by the message "Congratulations! You have successfully installed a new Symfony application." Below this are three large buttons: "READ THE QUICK TOUR" (with a book icon), "CONFIGURE" (with a laptop and wrench icon), and "RUN THE DEMO" (with a video camera icon). At the bottom, there are three columns of links: "Documentation" (The Book, The Cookbook, The Components, Reference, Glossary), "Sensio" (Trainings, Books), and "Community" (IRC channel, Mailing lists, Forum, Contributing). The bottom status bar shows various system icons and information: Symfony 2.5.3, PHP, 398b1d app dev, 200 WelcomeController::indexAction on \_welcome, 898 ms, 13.8 MB, 0 errors, and 0 warnings.


**Symfony**


Search on Symfony website OK

## Welcome!

Congratulations! You have successfully installed a new Symfony application.

 **READ THE QUICK TOUR**

 **CONFIGURE**

 **RUN THE DEMO**

**Documentation**

- [The Book](#)
- [The Cookbook](#)
- [The Components](#)
- [Reference](#)
- [Glossary](#)

**Sensio**

- [Trainings](#)
- [Books](#)

**Community**

- [IRC channel](#)
- [Mailing lists](#)
- [Forum](#)
- [Contributing](#)

Symfony 2.5.3 PHP 398b1d app dev 200 WelcomeController::indexAction on \_welcome 898 ms 13.8 MB 0 0

# Symfony: структура

- app/ (конфигурация приложения)
- src/ (код приложения)
- vendor/ (сторонние библиотеки)
- web/ (корневая директория приложения)

# Symfony: структура

```
▼ app
  ▼ Resources
    ▼ views
      base.html.twig
  ► cache
  ▼ config
    config.yml
    config_dev.yml
    config_prod.yml
    config_test.yml
    parameters.yml
    parameters.yml.dist
    routing.yml
    routing_dev.yml
    security.yml
  ► logs
    .htaccess
    AppCache.php
    AppKernel.php
    SymfonyRequirements.php
    autoload.php
    bootstrap.php.cache
    check.php
    console
    phpunit.xml.dist
```

# Symfony: структура

```
▼ src
  ▼ Acme
    ▼ DemoBundle
      ► Command
      ► Controller
      ► DependencyInjection
      ► EventListener
      ► Form
      ► Resources
      ► Tests
      ► Twig
      AcmeDemoBundle.php
    .htaccess
```

# Symfony: структура

```
▼ vendor
  ► composer
  ► doctrine
  ► incenteev
  ► jdorn
  ► kriswallsmith
  ► monolog
  ► psr
  ► sensio
  ► sensiolabs
  ► swiftmailer
  ► symfony
  ► twig
  autoload.php
```

# Symfony: структура

```
▼ web
  ▼ bundles
    ► acmedemo
    ► framework
    ► sensiodistribution
  .htaccess
  app.php
  app_dev.php
  apple-touch-icon.png
  config.php
  favicon.ico
  robots.txt
```

# Symfony: bundles

- bundle - пакет/плагин
- свои контроллеры, свои шаблоны, свои модели, своя конфигурация и т.д.
- Acme/DemoBundle



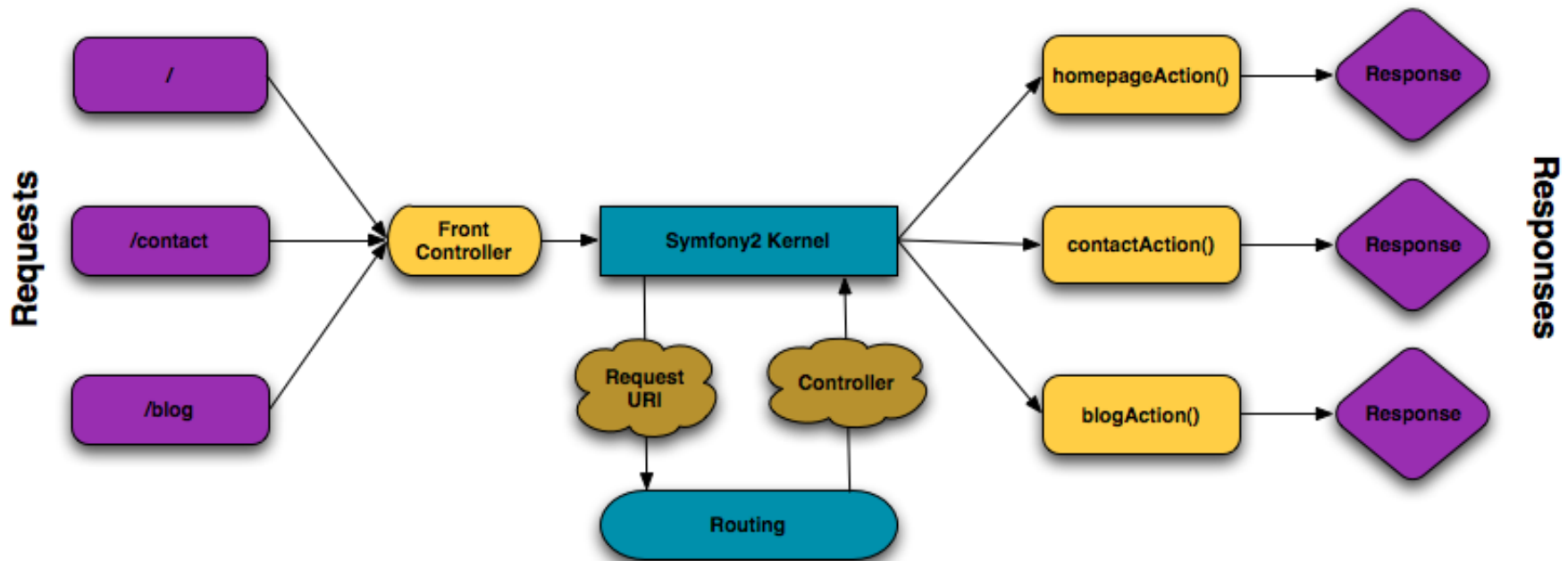
# Symfony: bundles

- Command
- Controller
- DependencyInjection
- EventListener
- Form
- Resources
- Tests
- Twig

# Symfony: окружение

- app.php VS app\_dev.php
- config\_prod.yml VS config\_dev.yml
- панель отладки

# Symfony: маршрутизация



# Symfony: маршрутизация

- `/src/Acme/DemoBundle/Resources/config/routing.yml`
- формат YAML
- аннотации

# Symfony: маршрутизация

```
_welcome:
    path:      /
    defaults: { _controller: AcmeDemoBundle>Welcome:index }

_demo_secured:
    resource: "@AcmeDemoBundle/Controller/SecuredController.php"
    type:     annotation

_demo:
    resource: "@AcmeDemoBundle/Controller/DemoController.php"
    type:     annotation
    prefix:   /demo
```

# Symfony: маршрутизация

```
class DemoController extends Controller
{
    /**
     * @Route("/hello/{name}", name="_demo_hello")
     * @Template()
     */
    public function helloAction($name)
    {
        return array('name' => $name);
    }

    ...
}
```

# Symfony: модель

- `/src/Acme/DemoBundle/Entity/` (Doctrine)
- `/src/Acme/DemoBundle/Model/` (Propel)

# Symfony: контроллер

```
namespace Acme\DemoBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class WelcomeController extends Controller
{
    public function indexAction()
    {
        /*
         * The action's view can be rendered using render() method
         * or @Template annotation as demonstrated in DemoController.
         */

        return $this->render('AcmeDemoBundle:Welcome:index.html.twig');
    }
}
```



# Symfony: ресурсы

/src/Acme/DemoBundle/Resources/



```
▼ Resources
  ▼ config
    routing.yml
    services.xml
  ▼ public
    ► css
    ► images
  ▼ views
    ► Demo
    ► Secured
    ► Welcome
    layout.html.twig
```

A file explorer view showing the structure of the Resources directory. The directory is expanded, revealing subdirectories 'config', 'public', and 'views'. 'config' contains 'routing.yml' and 'services.xml'. 'public' contains 'css' and 'images'. 'views' contains 'Demo', 'Secured', 'Welcome', and 'layout.html.twig'.

# Symfony: шаблоны

- `/src/Acme/DemoBundle/Resources/views/`
- `/app/Resources/views/`
- Twig

# Symfony: шаблоны

```
{% extends "AcmeDemoBundle::layout.html.twig" %}

{% block title "Symfony - Contact form" %}

{% block content %}
    <form action="{{ path('_demo_contact') }}" method="POST" id="contact_form">
        {{ form_errors(form) }}
        {{ form_row(form.email) }}
        {{ form_row(form.message) }}
        {{ form_rest(form) }}

        <input type="submit" value="Send" class="symfony-button-grey" />
    </form>
{% endblock %}
```

# Symfony: команды

```
/src/Acme/DemoBundle/Command/HelloWorldCommand.php
```

```
cd /projects/sf/  
php app/console acme:hello
```

Hello World!

# Материалы

[https://github.com/hiend/simbirsoft\\_examples/tree/ch05/symfony](https://github.com/hiend/simbirsoft_examples/tree/ch05/symfony)

# Задание

- Установить в свой LAMP фреймворк Symfony.
- Сгенерировать каркас приложения.
- Изменить вывод на странице (например, заменить логотип symfony в верхней части страницы).

**THE END**