Symfony: Начните с микро!

"require": {

"php": ">=7.2.5",

"ext-ctype": "\*",

"ext-iconv": "\*",

"sensio/framework-extra-bundle": "^6.2",

"symfony/console": "5.3.\*",

"symfony/dotenv": "5.3.\*",

"symfony/flex": "^1.3.1",

"symfony/framework-bundle": "5.3.\*",

"symfony/runtime": "5.3.\*",

"symfony/yaml": "5.3.\*"

},

composer require twig

<?php

namespace App\Controller;

use Symfony\Component\Routing\Annotation\Route;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class DefaultController extends AbstractController

{

/\*\*

\* @Route("/hello/{name}")

\*/

public function index($name)

{

return $this->render('default/index.html.twig', [

'name' => $name,

]);

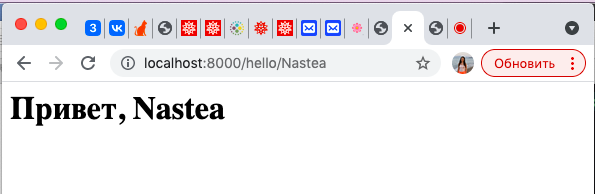
}

}

Расширив AbstractController, у вас теперь есть доступ к перечню методов сокращения и инструментов, вроде render(). Создайте новый шаблон:

{# templates/default/index.html.twig #}

<h1>Привет, {{ name }}</h1>



Добавляем:

{# templates/default/index.html.twig #}

{% extends 'base.html.twig' %}

{% block body %}

<h1>Привет, {{ name }}</h1>

{% endblock %}



Профилировщик: Рай отладки

composer require profiler

## Богатая поддержка API[¶](https://symfony.com.ua/doc/current/quick_tour/flex_recipes.html" \l "api" \o "Permalink to this headline)

Вы создаете API? Вы можете уже с лёгкостью вернуть JSON из любого контроллера:

<?php

// src/Controller/DefaultController.php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\Routing\Annotation\Route;

class DefaultController extends AbstractController

{

// ...

/\*\*

\* @Route("/api/hello/{name}")

\*/

public function apiExample($name)

{

return $this->json([

'name' => $name,

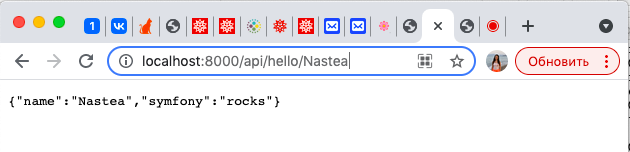
'symfony' => 'rocks',

]);

}

}

Путь: <http://localhost:8000/api/hello/Nastea>



Но для действительно богатого API, попробуйте установить [Платформу Api](https://api-platform.com/):

composer require api

Создайте сущность Product и дайте её аннотацию @ApiResource():

<?php

// src/Entity/Product.php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;

use Doctrine\ORM\Mapping as ORM;

/\*\*

\* @ORM\Entity()

\* @ApiResource()

\*/

class Product

{

/\*\*

\* @ORM\Id

\* @ORM\GeneratedValue(strategy="AUTO")

\* @ORM\Column(type="integer")

\*/

private $id;

/\*\*

\* @ORM\Column(type="string")

\*/

private $name;

/\*\*

\* @ORM\Column(type="int")

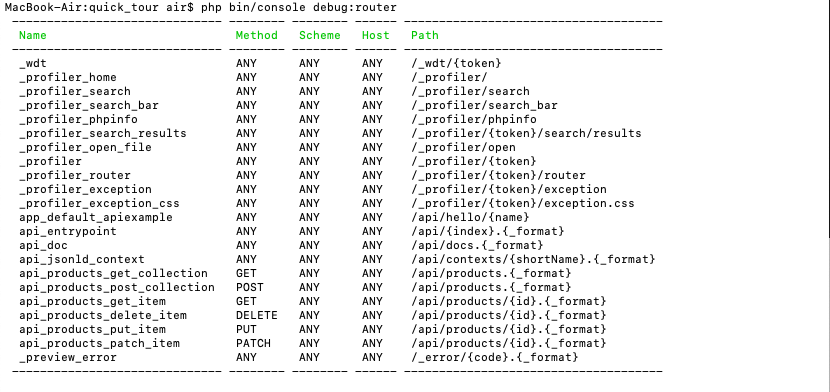
\*/

private $price;

}

Готово! У вас теперь есть конечные точки для перечисления, добавления, обновления и удаления продуктов!

php bin/console debug:router



## Удаление рецептов

Всё ещё не убеждены? Не проблема: удалите библиотеку:

composer remove api

Flex удалит рецепты: удаление файлов и отмена изменений для возвращения вашего приложения обратно в исходное состояние.

