Symfony lab\_5

В командной строке перейти в эту папку

Создать проект **composer create-project symfony/skeleton my\_db**

Запустить сервер

Затребовать:

composer require twig

composer require annotations

1 composer require symfony/orm-pack

* composer require --dev symfony/maker-bundle

2 внутри .env:

DATABASE\_URL="mysql://air:123@127.0.0.1:3306/symfony?serverVersion=mariadb-10.4.11"

3

* php bin/console doctrine:database:create

**Создание класса сущности**

* **php bin/console make:entity**

$ php bin/console make:entity

Class name of the entity to create or update:

> Product

New property name (press <return> to stop adding fields):

> name

Field type (enter ? to see all types) [string]:

> string

Field length [255]:

> 255

Can this field be null in the database (nullable) (yes/no) [no]:

> no

New property name (press <return> to stop adding fields):

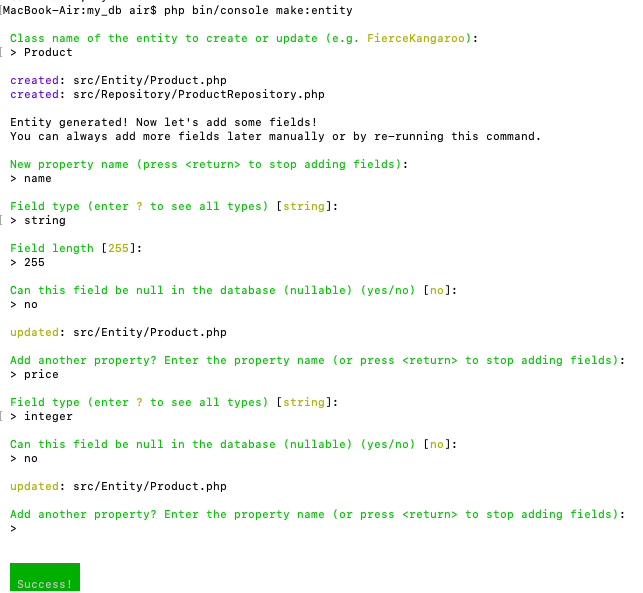
> price

Field type (enter ? to see all types) [string]:

> integer

Can this field be null in the database (nullable) (yes/no) [no]:

> no



* php bin/console make:migration
* php bin / console doctrine: migrations: migrate

**Сохранение объектов в базе данных**

<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\Response;

use Symfony\Component\Routing\Annotation\Route;

use App\Entity\Product;

use Doctrine\ORM\EntityManagerInterface;

class ProductController extends AbstractController

{

/\*\*

\* @Route("/product", name="create\_product")

\*/

public function createProduct(): Response

{

$entityManager = $this->getDoctrine()->getManager();

$product = new Product();

$product->setName('Keyboard');

$product->setPrice(1999);

//$product->setDescriptions('Ergonomic and stylish!');

// tell Doctrine you want to (eventually) save the Product (no queries yet)

$entityManager->persist($product);

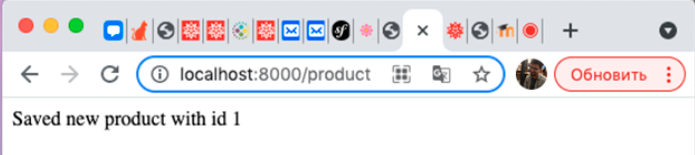
// actually executes the queries (i.e. the INSERT query)

$entityManager->flush();

return new Response('Saved new product with id '.$product->getId());

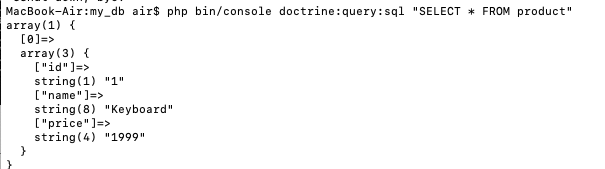
}

}



**вы можете напрямую запросить базу данных:**

php bin/console doctrine:query:sql "SELECT \* FROM product"

****

**Получение объектов из базы данных**

/\*\*

\* @Route("/product/{id}", name="product\_show")

\*/

public function show($id)

{ $product = $this->getDoctrine()

->getRepository(Product::class)

->find($id);

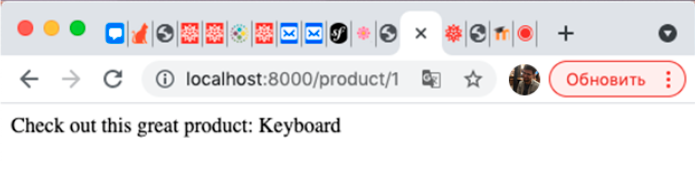
if (!$product) {

throw $this->createNotFoundException(

'No product found for id '.$id ); }

return new Response('Check out this great product: '.$product->getName());

}

****

**Обновление объекта**

/\*\*

\* @Route("/product/edit/{id}")

\*/

public function update($id)

{

$entityManager = $this->getDoctrine()->getManager();

$product = $entityManager->getRepository(Product::class)->find($id);

if (!$product) {

throw $this->createNotFoundException(

'No product found for id '.$id

);

}

$product->setName('New product name!');

$entityManager->flush();

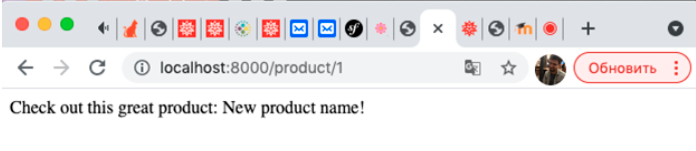
return $this->redirectToRoute('product\_show', [

'id' => $product->getId()

]);

}

<http://localhost:8000/product/edit/1>



**Удаление объекта**

/\*\*

\* @Route("/product/delete/{id}")

\*/

public function delete($id)

{

$entityManager = $this->getDoctrine()->getManager();

$product = $entityManager->getRepository(Product::class)->find($id);

if (!$product) {

throw $this->createNotFoundException(

'No product found for deleting id '.$id

);

}

$entityManager->remove($product);

$entityManager->flush();

// return new Response('delete product: '.$product->getName());

return new Response('Product with id= '.$id.' has been deleted!');

}