Министерство Образования, Культуры и Исследований

Молдавский Государственный Университет

Факультет Математики и Информатики

Департамент Информатики

**Лабораторная работа №1**

По курсу: Frameworks

На тему: Основы Symfony.

Выполнил:Voronin Alexandr,

группа IA2102

Проверил: **Nartea Nikita**,

lector universitar

Кишинев, 2023

Оглавление

[Введение 2](#_Toc147651729)

[Цель работы 2](#_Toc147651730)

[Подготовка 3](#_Toc147651731)

[Выполнение 3](#_Toc147651732)

[Ответы на вопросы: 4](#_Toc147651733)

[Ссылки 5](#_Toc147651734)

# Введение

Данная лабораторная работа представляет увлекательное погружение в мир Symfony и создание важного веб-приложения - ToDo. Symfony - это мощный инструмент для разработки веб-приложений на PHP, и в ходе работы мы исследовали его ключевые концепции: маршрутизацию, контроллеры, шаблоны и работу с базой данных. Затем мы применили наши знания, создав приложение ToDo, продемонстрировав, как Symfony делает процесс разработки увлекательным и эффективным. Эта лабораторная работа стала захватывающим введением в мир Symfony и его важные возможности

# Цель работы

Данная лабораторная работа знакомит с основами бэкенд фреймворков, используя в качестве примера создание небольшого `ToDo` приложения которое включает в себя:

***Маршрутизация и контроллеры***

1. Создание класс-контроллера с методами, для обработки соответствующих запросов:

1. `list`: просмотр списка задач

2. `view`: просмотр одной задачи

3. `create`: создание задачи

4. `update`: редактирование существующей задачи

5. `delete`: удаление задачи

**Создание сущностей**

Создание сущности \*\*"Task"\*\* и \*\*"Category"\*\*

Сущность \*\*Category\*\* включает в себя :

1. `id`: уникальный идентификатор категории

2. `name`: название категории

Сущность \*\*Task\*\* включает в себя :

1. `id`: уникальный идентификатор задачи

2. `title`: заголовок задачи.

3. `description`: описание задачи.

4. `dueDate`: срок выполнения задачи.

5. `createdAt`: дата и время создания задачи (заполняется автоматически)

В проекте определена связь между сущностями `Task` и `Category`. Одной задачи может соответсовать одна категория, одной категории может соотвествовать множество задач.

**Работа с шаблонами и данными**

1. Создание представления с использованием шаблонизатора Twig для каждой страницы приложения 'ToDo'.

1. Страница: список задач

2. Страница: просмотр одной задачи

3. Страница: создание задачи и редактирование задачи

2. Для страницы списка задач (list) отобразить данные всех задач, включая заголовок, срок выполнения и категорию

3. Для страницы одной задачи отобразить полное описание задачи.

**Обработка и сохранение данных из формы**

1. Создание соответсвующей формы для создания и редактирования задачи.

2. Обработать отправку данных из формы, провести валидацию и сохранить данные в базу данных.

1. Мин. длина названия: \*\*4\*\* символов.

2. Макс. длина названия: \*\*20\*\* символов.

3. Реализация удаление задачи.

# Подготовка

Для создания приложения, я скачиваю и устанавливаю Symfony с помощью Composer следуя официальной документации и дополнительные комплектующие

# Выполнение

**Создание сущностей**

Для создания наших сущностей мы используем команду:

php bin/console make:entity

далее мы выбираем название поля(поле id создано по умолчанию), их типы и связаны ли они с данными из другой таблицы

После этого мы выбираем как связаны Task и Category и какая у них связь в нашем случае (Одной задачи может соответствовать одна категория, одной категории может соотвествовать множество задач)

После этого мы запускаем migration в cmd

$ php bin/console make:migration

$ php bin/console doctrine:migration:migrate

После данных шагов у нас создались две сущности и связь между ними для нашего приложения

***Маршрутизация и контроллеры***

Для создания TaskController, я использовал в cmd команду:

$ symfony console make:controller

Для разделения моих задач я использую разные маршруты, которые помогают эффективнее использовать приложения

Для просмотра всех задач у меня используется **Route**:

#[Route('/task', name: 'app\_task\_')]

Для использования каких-либо действий мы добавляем /ключевое слово /динамическое id задачи

Для взаимодействия с нашей бд мы используем **TaskRepository и EntityManagerInterface** который имеет методы для получения, добавления и изменения наших данных.

Далее мы используем **render** на наши шаблоны Twig

**Работа с шаблонами и данными**

Шаблон автоматически создается с контроллером и для каждого нового маршрута, я создал отдельный шаблон.

Для использования метода в нашем приложение, мы используем:

return $this->render("task/list.html.twig", [

            'tasks'=>$task,]

Где возвращаем наш метод и его рендерим в файле нашего шаблона, а именно task/list.html.twig

Далее мы передаем нашу переменную метода $task в шаблон

Каждый мой шаблон имеет наследование от base.html.twig, что помогает добавить основные стили и структуру а далее просто наследовать в шаблоны

Для использования переменных мы используем специальный синтаксис, а именно: {{имя переменной для шаблона}}, циклы для вывода данных заключаются в {%название цикла %} и {% endназвание цикла %}

Далее я создал для всех задач отдельные шаблоны для представления данных пользователю

**Обработка и сохранение данных из формы**

Для создания формы нашего ToDo приложения я использовал в cmd

php bin/console make:form

После нужно дать имя нашей форме и с какой бд она будет взаимодействовать, в нашем случае это Task.

Далее в нашем методе в TaskController мы прописываем

$form=$this->createForm(TaskType::class,$task);

Это позволяет взамодействовать с нашей формой

В нашем файле форме мы добавляем все нужные поля при помощи

->add('name', TextType::class,

Где вместо name название формы а TextType тип для формы

Далее мы добавляем Валидацию в нашу форму, а именно: new Length(),new NotBlank(),new New Regex()

Также мы можем выбрать категорию для нашей созданной задаче это реализовано при помощи:

>add('category', EntityType::class, [

                'class' => Category::class,

                'choice\_label' => 'name',

Где у нас есть выпадающее меню, которое берет информацию из сущности Категории.

Также при помощи маршрута edit/{id} мы можем изменять нашу задачу находя ее по id и изменяя ее с помощью нашей формы.

## Ответы на вопросы:

**Какие есть преимущества использования фреймворка Symfony?**

Основные преимущества — Symfony-это упрощение процесса создания кода, быстрое создание и легкое использование Базы данных, различных форм, а также гибкое использование маршрутов и создание шаблонов для отображения проектов. Также использование Фреймворка в разработке более безопасно и делает код более читабельнее за счёт связи с ООП.

**Какие есть способы определения маршртутов в Symfony?**

Основной способ определения маршрута в проекте это-перед нашим методом в начале, например:

#[Route('/task', name: 'app\_task\_')]

Где вместо ‘/task’ и ‘app\_task’ может стоять наше любое значение нужного для нашего проекта

Также мы может возвращать рендерный метод по маршруту делая так:

return $this->redirectToRoute('app\_task\_list');

**Какая связь между таблицами баз данных была использована и как именно это было реализовано?**

В моем проекте была реализована связь одной задачи может соответствовать одна категория, одной категории может соответствовать множество задач. Для реализации этого, я при создание сущности создал поле category\_id которое имеет тип relation и определил желаемую связь и все.

**Что такое миграции базы данных, и как они применяются в Symfony?**

Миграции в Symfony помогают лучше и удобнее создавать и изменять базу данных, что для проекта является полезно. Благодаря контролю миграций в Symfony мы можем в случае ошибки при создании БД откатиться на предыдущею версию БД или изменить какие-либо поля, благодаря этому процесс разработки становиться более надежным.

# Ссылки

[Curs: Frameworks for Web Application Development (usm.md)](https://moodle.usm.md/course/view.php?id=6440)

[Symfony Documentation](https://symfony.com/doc/current/index.html)

[Form controls · Bootstrap v5.3 (getbootstrap.com)](https://getbootstrap.com/docs/5.3/forms/form-control/)

[AlexVor550/ToDo (github.com)](https://github.com/AlexVor550/ToDo)=ссылка на мой github