**Лабораторна робота 22.**

**Встановлення та налаштування Symfony. Підключення до БД та виведення Hello world!**

**Мета роботи:** Познайомитись із фреймворком Symfony, навчитись його встановлювати, настроювати середовища робробки PHPStorm для роботи із Symfony. Створити новий проект Symfony у PHPStorm та виконати підключення до БД.

**Задачі:**

* Познайомитись із фреймворком Symfony та встановити його із використанням composer
* Виконати налаштування PHPStorm та створити проект Symfony
* Виконати підключення до БД «студенти»

1. Як визначає себе сам Symfony, це «набір повторюваних компонентів PHP» та «PHP-фреймворк для веб-проектів». Звернувшись до вікіпедії, можемо побачити наступне визначення: Symfony — відкритий PHP-фреймворк, що реалізує концепцію модель-представлення-контролер (MVC) та автоматизує найзагальніші веб-задачі, являє собою широконалаштовну систему пов'язаних класів і призначений для розробки та керування веб-застосунками.

Symfony Framework добре відомий тим, що він дійсно гнучкий і використовується для створення мікро-сайтів, корпоративних програм, які обробляють мільярди з'єднань і навіть є основою для інших фреймворків.

Основа будь якого web-застосунку – робота із HTTP. На простому рівні HTTP (протокол передачі гіпертекст) - це текстова мова, яка дозволяє двом машинам спілкуватися одній із одною. HTTP - це термін, який використовується для опису цієї простої текстової мови. Незалежно від того, що і як ви розробляєте в Інтернеті, мета вашого сервера - розуміти прості текстові запити та повертати прості текстові відповіді. Symfony побудовано з нуля саме навколо цієї концепції.

Крок 1: Клієнт надсилає запит. Крок 2. Сервер повертає відповідь (рис.1).

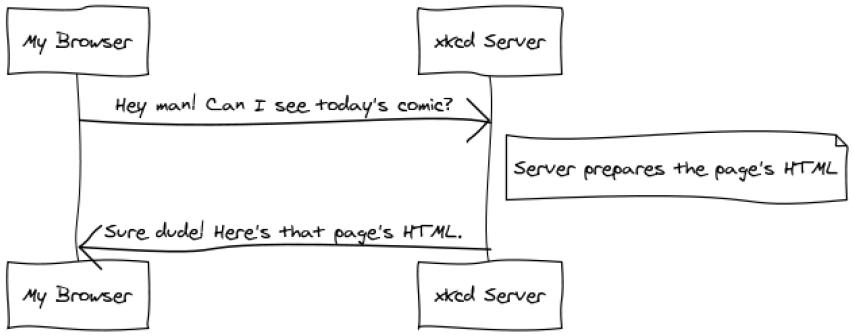


Рис.1.Запит-відповідь при роботі із веб-сервером.

Відповідь може повертатись у різних форматах, таких як HTML, XML або

JSON, і заголовок Content-Type використовуєьтся для повідомлення клієнту про цей формат.

Symfony пропонує альтернативий до «сирого» PHP-підходу, що полягає у роботі із двома класами, які дозволяють вам взаємодіяти із HTTP-запитом та відповіддю простіше. Клас Request - це просте об'єктно-орієнтоване представлення повідомлення HTTP-запиту. З ним у вас є вся інформація про запит. Symfony також надає клас Response, що дозволяє вашій програмі використовувати об'єктно-орієнтований інтерфейс для побудови відповіді для повернення клієнту.

1. Як і HTTP, об'єкти запиту та відповіді є досить прості. Складна частина написання застосунку – визначити, що відбувається між ними. Інакше кажучи, реальна робота полягає в написанні коду що інтерпретує інформацію про запит і створює відповідь. Традиційно додатки були побудовані так, щоб кожна "сторінка" сайту мала свій власний фізичний файл. Набагато кращим рішенням є використання переднього контролера: єдиного файлу PHP, який обробляє всі запити.

Усередині контролера ми повинні з'ясувати, який код повинен бути виконаний і який контент повинен повернутися. Щоб це зрозуміти, нам потрібно буде перевірити вхідний URI та виконати різні частини вашого коду залежно від цього значення. Symfony працює за єдиним шаблоном із кожним запитом (рис.2)

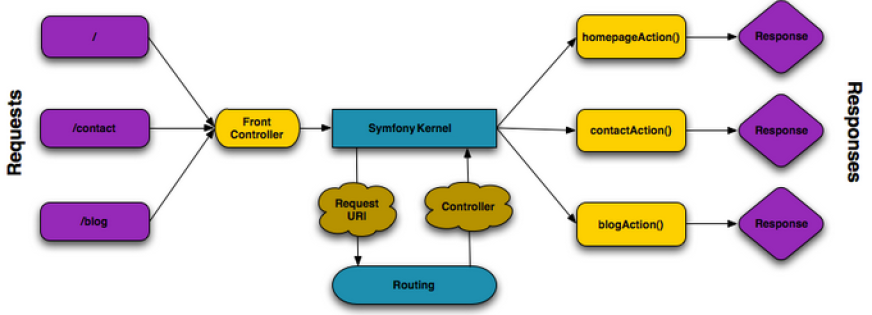


Рис.2. Шаблон обробки запиту.

Потужна та швидка система маршрутизації Symfony дозволяє нам пов’звувати певний URI до інформації про те, як потрібно обробляти цей запит.

1. Одним з найпоширеніших і складних завдань для будь-якої програми є читання інформації із бази даних і запис змінених даних у неї. Symfony Full-Stack Framework не інтегрує жодної ORM за замовчуванням. Але найбільш широко поширений дистрибутив Symfony Standard Edition інтегрований із Doctrine.

Doctrine повністю відокремлена від Symfony, і використовувати її необов'язково. Якщо ви віддаєте перевагу користуватися «сирими» запитами бдо ази даних, це також може бути реалізовано. Але використання ORM Doctrine має цілий ряд переваг, що частково були розглянуті у попередніх розділах та ще будуть розглядатись у подальшому. Використовуючи Doctrine ODM бібліотеку, можливо зберігати дані у більшості найпопулярніших реляційних (і не тільки) СУБД, що підтримують роботу через PDO.

1. Виконаємо встановлення фреймворку із використанням composer. Для цього, перебуваючи у потрібній папці, запустимо командний рядок та виконаємо наступну команду (рис.3)

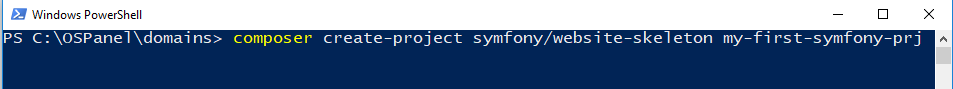


Рис.3. Встановлення Symfony із використанням composer.

Після установки, перезапустимо OSPanel та наберемо в адресному рядку браузера «http://my-first-symfony-prj/public/» для перевірки роботи встановленого фреймворку (рис.4)

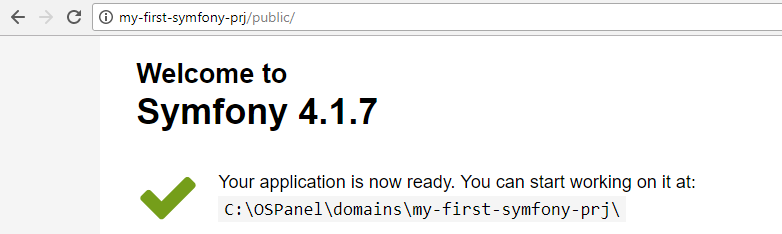


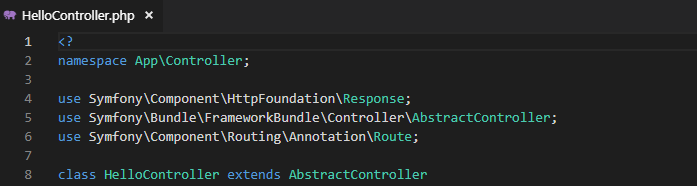
Рис.4. Перевірка роботи встановленого фреймворку.

Далі відкриємо наш проект за допомогою Visual Studio Code та реалізуємо вивід повідомлення «Hello world !» за адресою «http://my-first-symfony-prj/public/hello».

Створення нової сторінки - будь то HTML-сторінка або JSON дані – двоетапний процес:

* Створення маршруту: маршрут - це URL-адреса на вашу сторінку, що вказує на контролер;
* Створення контролера: контролер - це функція на PHP, яка створює сторінку-відповідь. Ви берете інформацію про вхідний запит та використовуєте її для створення об'єкта Symfony Response, який може містити HTML, JSON або навіть бінарний файл, такий як зображення чи PDF.

У папці «src/Controller» створимо новий файл HelloController.php наступного вмісту (рис.5)



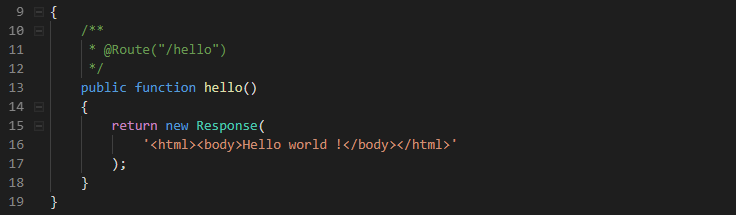


Рис.5. Файл контролера HelloController.php

Перевіримо роботу тестової сторінки. У командному рядку наберемо «php bin/console server:run» для запуску Symfony PHP Web-серверу (рис.6). Далі у браузері перейдемо за посиланням «http://127.0.0.1:8000/hello» (рис.7)

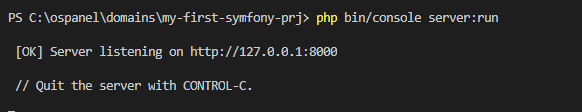


Рис.6. Запуск Symfony PHP Web-серверу.

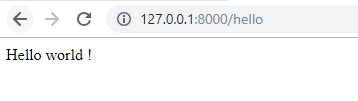


Рис.7. Завантаження сторінки Hello world

Для можливості використання web-серверу apache для подальшої роботи, необхідно виконати відповідні його налаштування. Самий простий спосіб це зробити – виконати команду, наведену на рис.8.



Рис.8. Створення . htaccess у папці public/

Після цього зможемо скористатись також посланням «http://my-first-symfony-prj/public/hello» (рис.9).

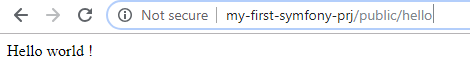


Рис.9. Завантаження Hello world із використанням apache web-server.

1. Використаємо окремий файл та шаблонізатор для генерації відповіді на запит hello world. Для цього у папці «templates» створимо файл «hello.html.twig» (рис.10).

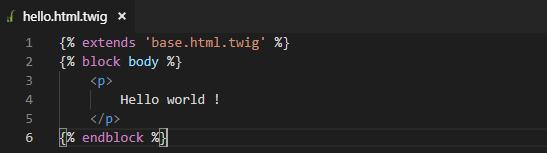


Рис.10. Вміст файлу hello.html.twig.

Даний курс не присвячений детальному огляду шаблонізатора twig, тому зазначимо лише, що наведений шаблон наслідує від base.html.twig, що знаходиться у тій же директорії, та перевизначає у ньому блок «body», виводячи у нього «Hello world !». Тепер змінимо наш контролер для роботи із створеним шаблоном (рис.11)

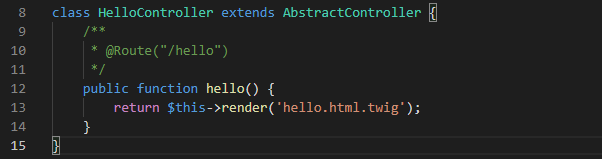
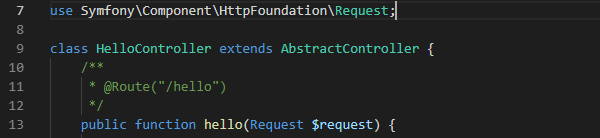


Рис.11. Змінений контролер, що використовує шаблон «hello.html.twig»

Одна із переваг шаблонів – можливість передати у них параметри. Використаємо її для передачі імені особи, з якою слід привітатися через масив $\_GET. Спочатку внесемо зміни у контролер (рис.12)



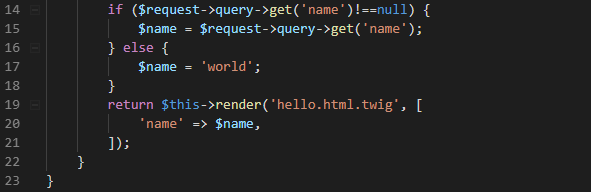


Рис.12. Зміни у контролері для передачі параметра до шаблону.

Також змінимо сам шаблон для обробки переданого параметру (рис.13)

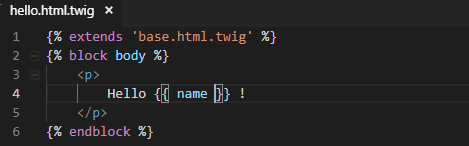


Рис.13. Обробка параметру в шаблоні.

Передамо у запит параметр name=Вася та переглянемо результат (рис.14)

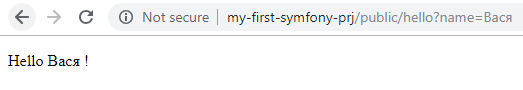


Рис.14. Вигляд сторінки hello при передачі параметру name

1. Далі продовжимо нашу роботу у середовищі розробки phpStorm. Для зручності роботи рекомендується встановити також розширення «PHP Annotation» (рис.15) та «Symfony Plugin» (рис.16)

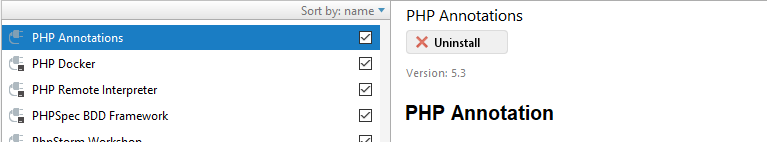


Рис.15. Плагін PHP Annotation

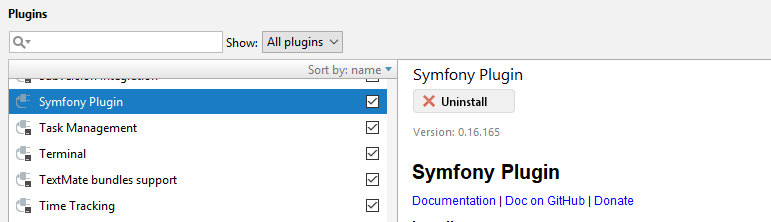
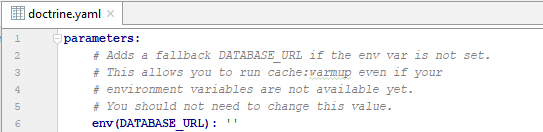


Рис.16. Плагін Symfony Plugin.

Далі виконаємо налаштування для підключення до нашої БД «студентів». Для цього у папці «config» відкриємо файл «doctrine.yaml» та у розділ «parameters» додамо перелік змінних, необхідних для збереження даних про наше підключення (рис.17), а у розділі doctrine -> dbal використаємо їх для налаштування підключення до нашої БД (рис.18).



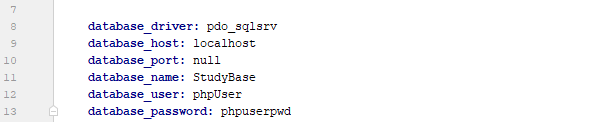


Рис.17. Додавання нових параметрів

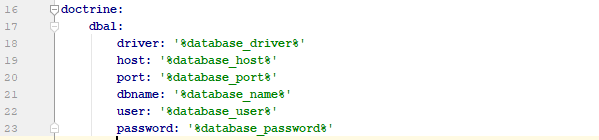


Рис.18. Налаштування з’єднання.

Після виконання вказаних налаштувань перевіримо з’єднання із БД, виконавши у командному рядку запит на вибірку даних (рис.19). Виконуємо вибірку поля «name» з таблички «кафедри».

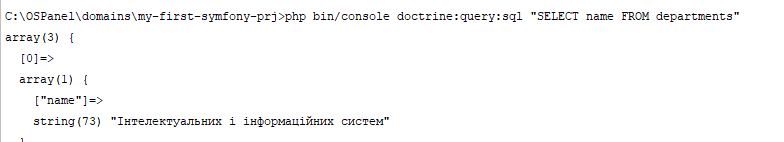


Рис.19. Вибірка списку кафедр.

Спробуємо припустити помилку у назві БД або сервера та повторити вибірку (рис.20-21)



Рис.20. Встановлення помилкових параметрів підключення.

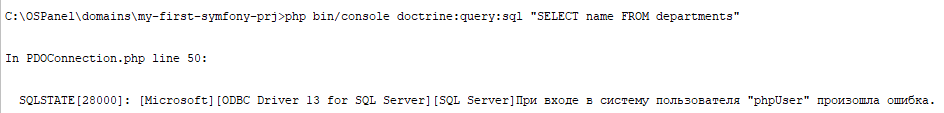
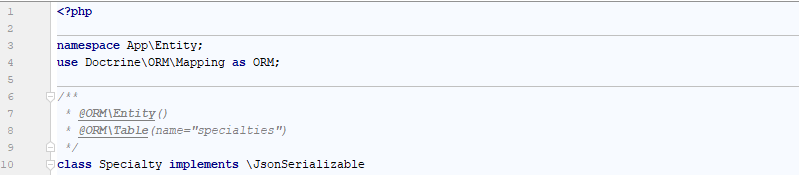
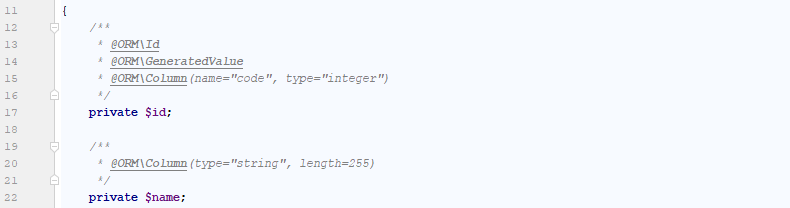


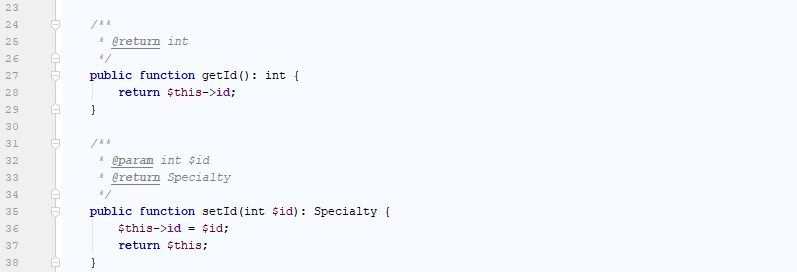
Рис.21. Помилка підключення до БД.

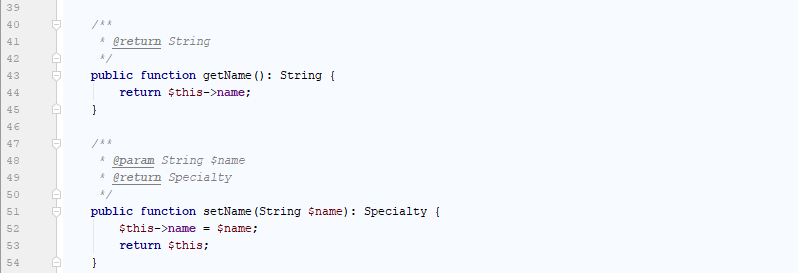
https://symfony.com/doc/current/doctrine.html

1. До папки «src/Entity» додамо новий файл «Specialty.php» із класом Specialty, що відповідатиме за роботу із табличкою «спеціальності» БД «студенти» (рис.22). У даному класі через анотації прописується відповідність між класом та табличкою БД, та між полями об’єкту та полями таблички. Також маємо гетери та сетери для полів та метод jsonSerialize() для повернення об’єкту у форматі JSON.









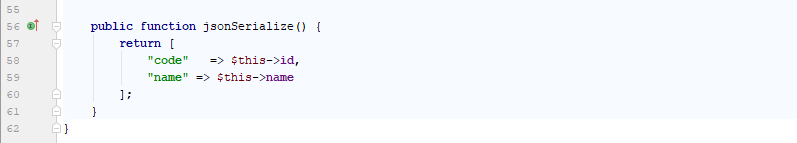


Рис.22. Клас-сутність Specialty.

На цьому основна робота по реалізації взаємодії із табличкою завершена. Тепер, якщо нам потрібно наприклад виконати вибірку всіх даних із таблиці, це може бути реалізовано за допомогою методу findAll(). Так, на рис.23 наведемо реалізацію сервісу, що повертає список спеціальностей у json форматі. Додаємо новий метод до класу контролера «HelloController».

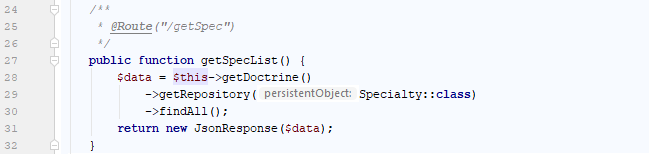


Рис.23. Повернення спеціальностей у json форматі.

Перевіримо роботу нашого сервісу, перейшовши за посиланням «http://my-first-symfony-prj/public/getSpec» (рис.24).

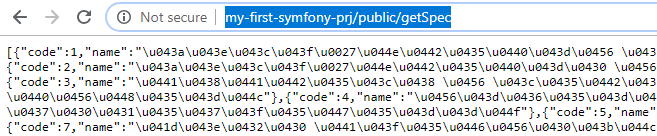


Рис.24. Отримання списку спеціальностей у json форматі.

1. Кінець роботи.

**Завдання для індивідуального виконання.**

Установити фреймворк Symfony та реалізувати сторінку «Hello world» із використанням шаблонізатору twig. Виконати підключення до власної БД та повернення у JSON-форматі вмісту однієї з таблиць а вибір.