**Лабораторна робота 23.**

**Реалізація класів-сутностей та контролерів для роботи із таблицями БД**

**Мета роботи:** Вивчити різні способи створення класів сутностей за завантаження змін структури до БД. Розібрати створення та роботу із об’єктами класів, що зберігаються у декількох табличках БД.

**Задачі:**

* Для створеного класу-сутності розібрати операції вибірки, додавання, зміни та видалення даних;
* Створити клас-сутність із готової таблиці БД
* Створити новий клас-сутність та виконати застосування змін в БД – автоматичне створення нової таблиці
* Створити складний клас-сутність для роботи із декількома таблицями.
* Виконати операції читання та модифікації для цього класу.

1. Наведемо реалізацію операції вибірки даних із використанням доданого у попередній роботі класу-ситності «Спеціальність». Для початку створимо новий клас контролеру. Дана операція може бути виконана вручну, як у попередній роботі, але можна також скористатись інтерфейсом командного рядку Symfony. Зробимо це, та виконаємо команду, наведену на рис.1. Після введення команди у командному рядку необхідно також ввести ім’я нового контролера.

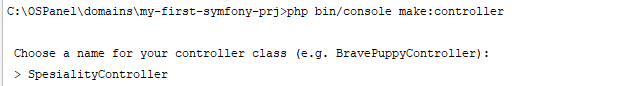
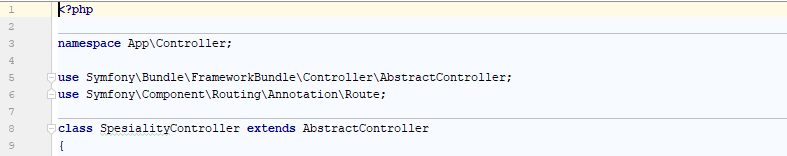


Рис.1. Створення нового контролеру за допомогою командного рядку Symfony.

Переглянувши вміст папки «src/Controller» знаходимо новий файл «SpesialityController.php» наступного вмісту (рис.2)



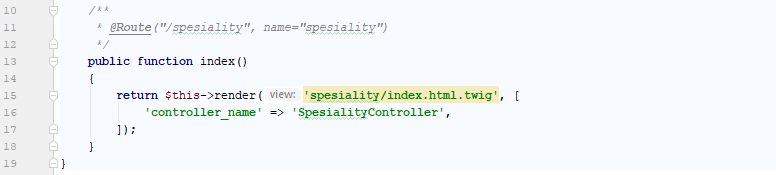


Рис.2. Вміст згенерованого контролеру SpesialityController.

Також можемо звернути увагу, що у методі index() використовується шаблон «index.html.twig», що був також автоматично згенерований у папці «templates/spesiality». Перевіимо роботу нашого контролеру, набравши у браузері адресу «http://my-first-symfony-prj/public/spesiality» (рис.3).

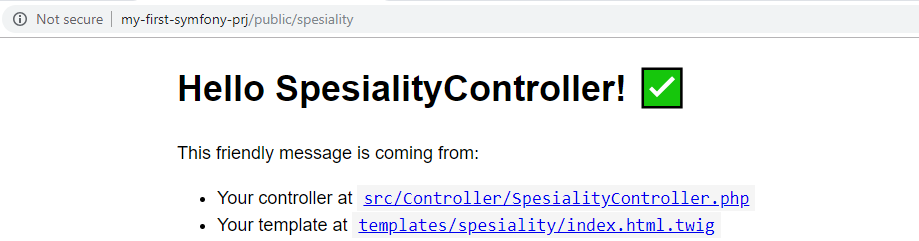


Рис.3. Перевірка роботи створеного контролера.

Виконаємо в index() повернення списку спеціальностей та використаємо для цього метод dump(), що повертає дані на панель розробника Symfony (рис.4-6).



Рис.4. Перенесення списку спеціальностей за допомогою dump.



Рис.5. Панель інструментів Symfony.

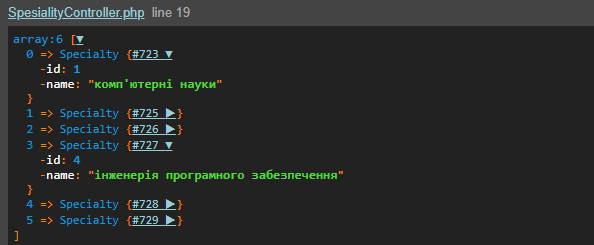


Рис.6. Перегляд даних, виведених через dump().

Dump() є потужним інструментом, що може бути використано на стадії відлагодження коду. Даний інструмент також може бути використано всередині шаблону для виведення даних для перегляду безпосередньо на сторінку. Для цього трохи змінимо метод index() нашого контролеру та шаблон «index.html.twig» у «templates/spesiality» (рис.7-9)

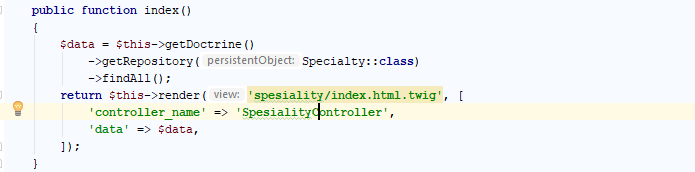


Рис.7. Зміни у методі index().

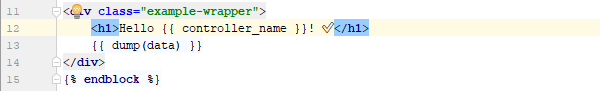


Рис.8. Зміни у шаблоні index.html.twig.

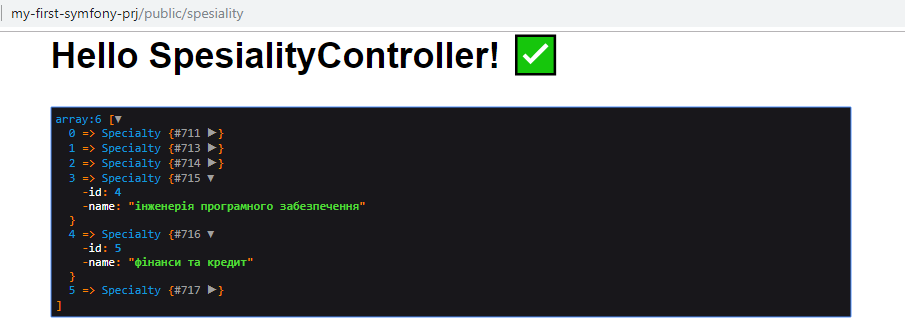


Рис.9. Результат виводу dump у шаблоні.

Додамо метод getSpesiality() для можливості пошуку та вибірки спеціальності за кодом (рис.10)

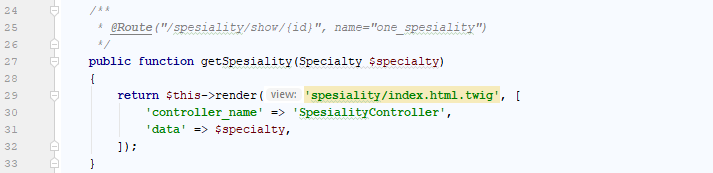


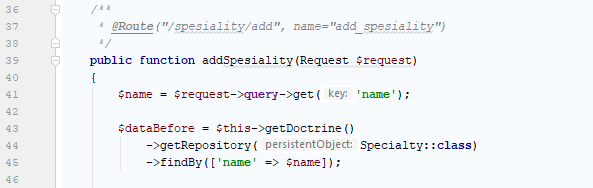
Рис.10. Вибірка спеціальності за кодом.

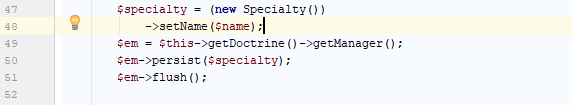
Перевіримо його роботу, перейшовши за посиланням «http://my-first-symfony-prj/public/spesiality/2», щоб відобразити спеціальність із кодом 2 (рис.11).



Рис.11. Результат вибірки спеціальності за кодом.

1. Реалізуємо додавання нової спеціальності із використанням масиву $\_GET. Метод addSpeciality() матиме наступний вигляд (рис.12). У методі також виведемо вибірку спеціальності, що додається, до та після команди для демонстрації того що спеціальність було додано.





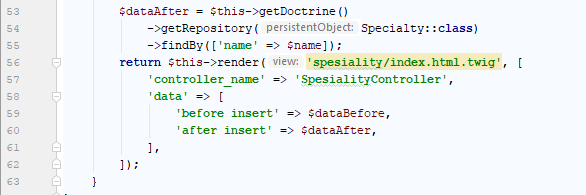


Рис.12. Метод для додавання нової спеціальності.

Виконаємо перевірку додавання даних (рис.13).

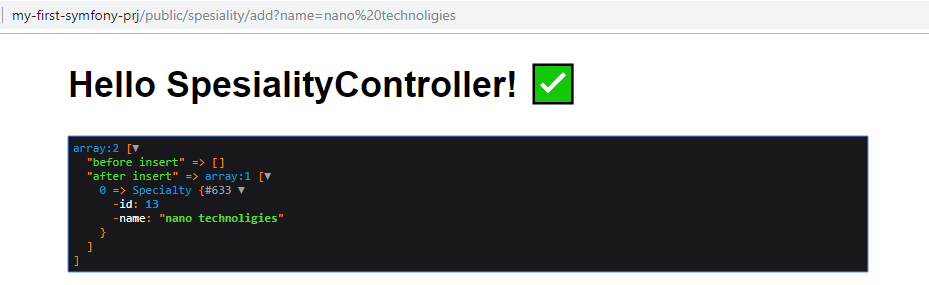
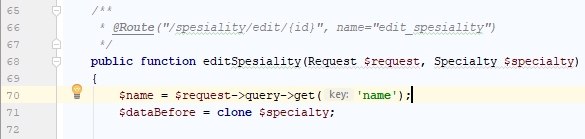
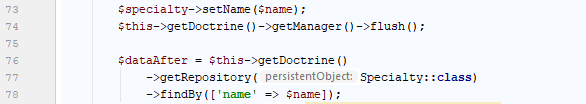


Рис.13. Перевірка методу додавання даних.

1. Аналогічно реалізуємо метод зміни даних. Код спеціальності будемо передавати, як частину шляху url, а нову назву – у масиві $\_GET (рис.14-15).





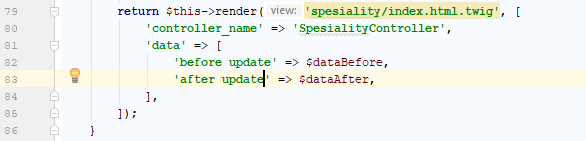


Рис.14. Реалізація методу для зміни назви спеціальності.



Рис.15. Перевірка роботи зміни даних.

Перевірити зміни також можна, виконавши відповідний запит на вибірку даних у ssms (рис.16).

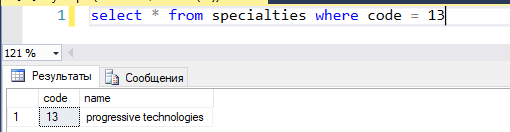
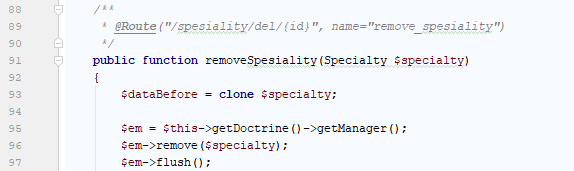
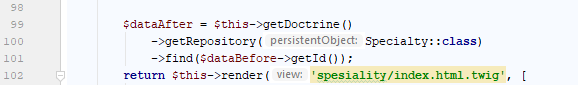


Рис.16. Вибірка змінених даних через SSMS.

1. Реалізуємо також і видалення даних спеціальності. Для цього додамо до нашого контролеру метод removeSpeciality() (рис.17-18).





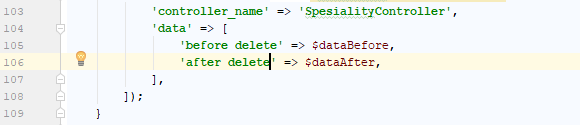


Рис.17. Метод для видалення спеціальності.

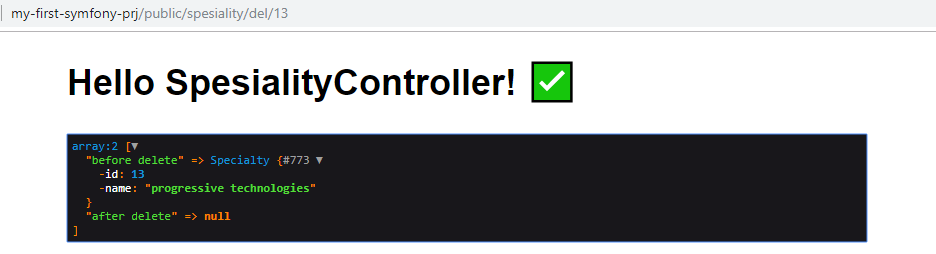


Рис.18. Перевірка видалення даних.

1. Виконаємо автоматичне створення нового класу-сутності на базі існуючої у БД таблиці. Для цього у командному рядку виконаємо наступну команду (рис.19)

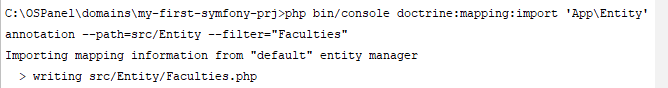
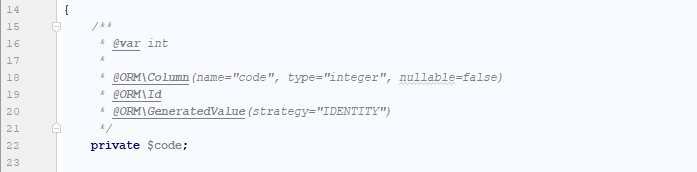


Рис.19. Команда генерації нового класу-сутності за прототипом готової таблиці БД.

Після виконання даної команди у папці «src\Entity» з’являється новий файл Faculties.php, що містить клас-сутність Faculties (рис.20), в якому вже прописані всі відповідності щодо таблиці та полів БД.





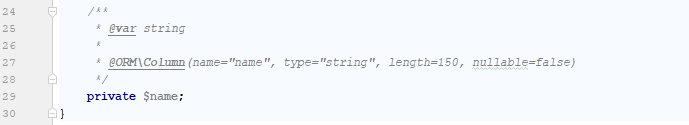


Рис.20. Згенерований клас Faculties

Далі у разі необхідності, клас доповнюється необхідними гетерами та сетерами і може бути використаний у контролері та зразок попереднього класу «Specialty».

Наведена на рис.19 команда також може бути використана для генерації класів для всіх таблиць, що присутні у БД. Для цього фільтр --filter="Faculties" має бути опущений.

*! Увага.* ORM Doctrine дозволяє генерацію класів із таблиць БД тільки за умови, що у БД не буде таблиць, у яких не визначено обмеженя цілісності «первинний ключ».

1. Заготовка для нового класу-сутності також може бути створена із використанням можливостей командного рядку Symfony. Для цього виконуємо команду «» та далі відповідаємо на питання щодо полів нашого класу, їх типів даних, та обмежень цілісності, пов’язаних із ними. Створимо у такий спосіб клас «Аудиторія» (рис.21)

















Рис.21. Створення класу-сутності «Аудиторія» за допомогою інтерфейсу командного рядку.

В результаті у папці «src\Entity» створено файл «Classroom.php», що містить клас-сутність для роботи із табличкою «сlassroom». Але у нашій БД на даний момент немає такої таблиці, тому при спробі виконання опеацій читання-запису отримаємо помилку. Є 2 шляхи вирішення проблеми – або створити таблицю в БД вручну відповідно до полів нашого класу, або скористатись інтерфейсом командного рядку Symfony для автоматичної генерації змін у нашу БД. Для цього виконаємо наступну команду, що спочатку виконає вивід на екран SQL-коду для змін структури нашої БД (рис.22).

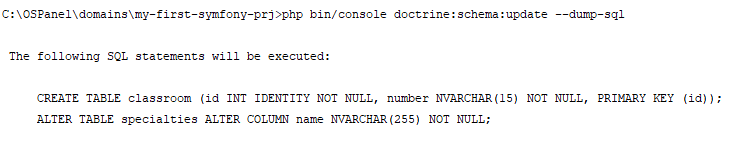


Рис.22. Зміни структури БД.

Як бачимо, doctrine ідентифікував 2 зміни у структурі – створення нової таблиці «classroom» та зміна типу даних для поля name таблиці specialties. Друга запропонована зміна пов’язана із тим, що ми не уточнили довжину при створенні поля name у класі Specialty. Тому пакетна зміна нас не зовсім влаштовує. Через це можемо скопіювати скрипт на створення таблички та виконати його в ssms (рис.23). Або змінити налаштування для поля name та виконати повторно команду рис.22, але тепер уже без параметру --dump-sql (і у разі необхідності з параметром –force).

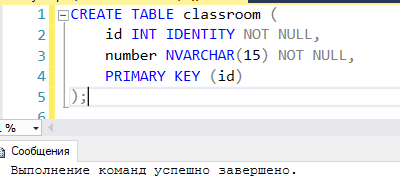
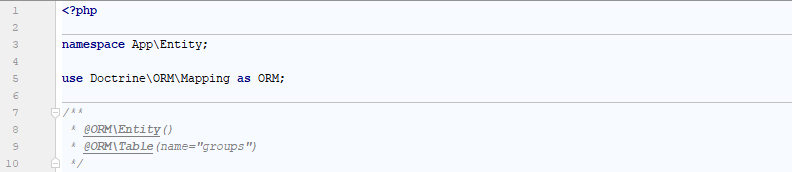
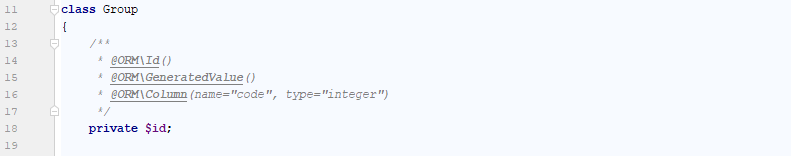
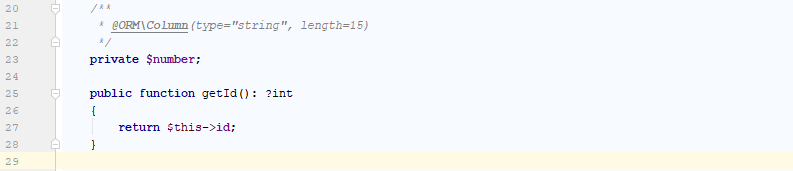


Рис.23. Створення таблички classroom.

1. Далі розглянемо механізми створення полів із обмеженням цілісності зовнішній ключ та включення табличних частин до класів-сутностей. Одним із наведених вище способів створимо спрощений клас-сутність студентська група, що буде працювати із лише із кодом на номером групи (рис.24)







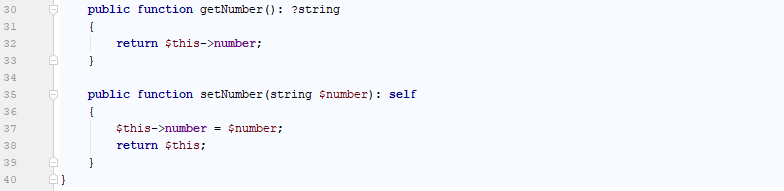
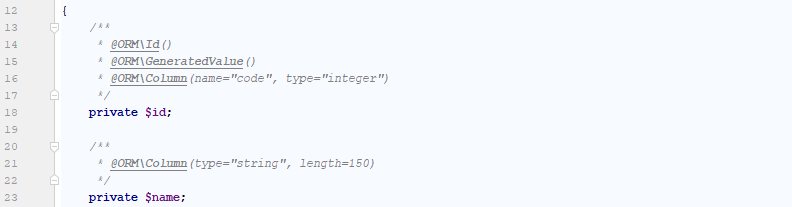
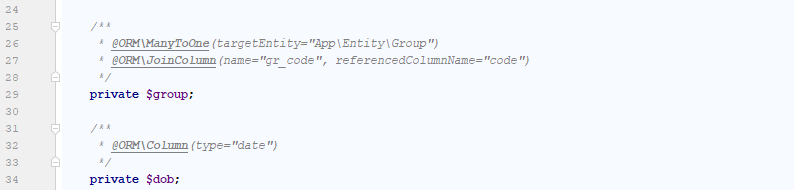


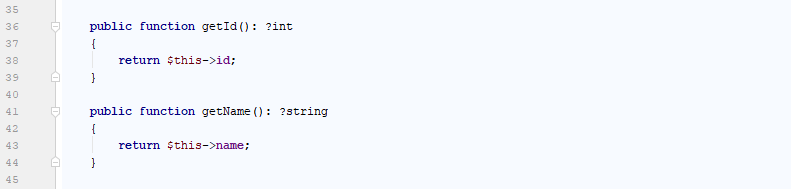
Рис.24. Спрощений клас-сутність для роботи із студентськими групами.

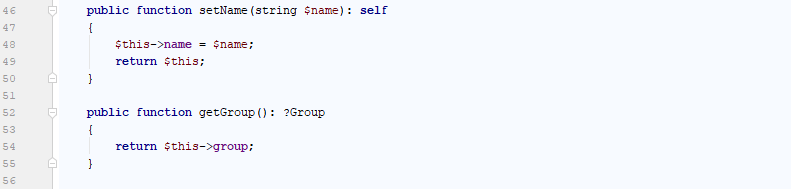
Далі створимо клас «студент», що буде мати поле-посилання на студентську групу (рис.25).

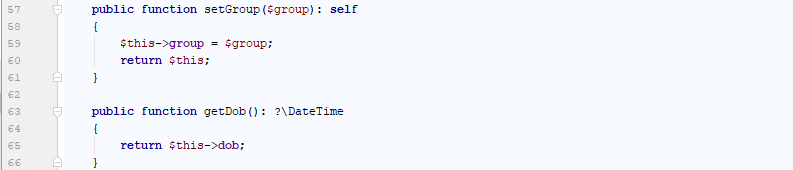












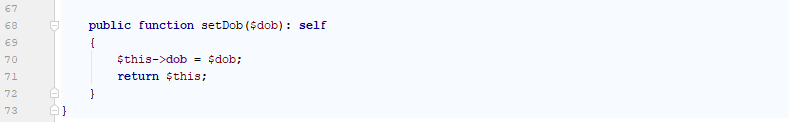
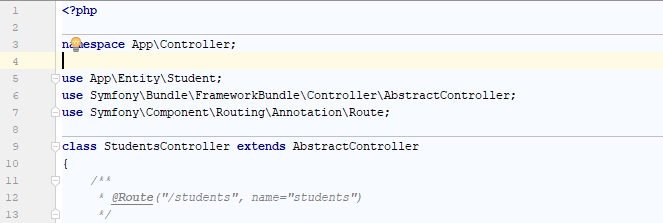


Рис.25. Клас-сутність «студент».

У переважній більшості коду клас «Студент» схожий на вже попередньо реалізовані, окрім поля «група» (строки коду 25-29), що є посиланням на інший клас-сутність та відповідно таблицю БД. Створимо контролер із методом, що читає 2-х студентів (рис.26) та шаблон для їх відображення (рис.27). На рис.28 наведено результат.



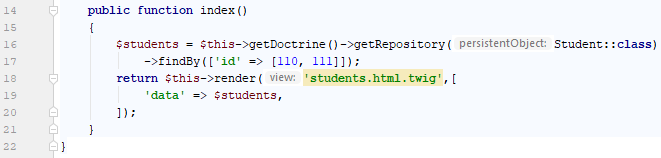


Рис.26. Метод для виведення 2-х студентів.

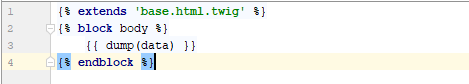


Рис.27. Шаблон students.html.twig.

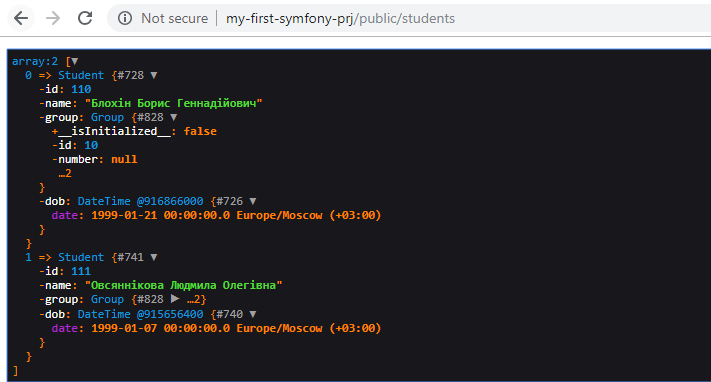


Рис.28. Зовнішній вигляд сторінки «http://my-first-symfony-prj/public/students»

Звернімо увагу на той факт, що при розгортанні детальних даних по групі, до якої належить студент, бачимо номер групи = null, хоч насправді це не так. Дана картина має місце через використання технології LazyLoad, що використовує Doctrine за замовленням. Тобто оскільки ми не зверталися до атрибутів студентської групи, Doctrine вважає, що на даному етапі вони нам не потрібні, а отже не виконує їх читання із БД. Перевірити це можна переглянувши список запитів для БД використовуючи панель інструментів Symfony (рис.29-30).



Рис.29. Панель інструментів Symfony. Database queries.

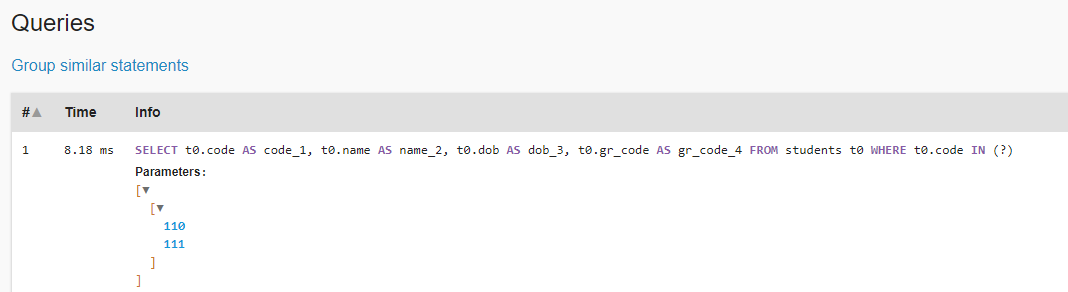


Рис.30. Symfony Profiler. Queries.

Якщо ж ми виконаємо читання даних щодо групи у контролері (рис.31), картина зміниться і ми вже будемо бачити реальний номер групи (рис.32). Також можна звернути увагу, що додався новий запит до БД, що виконує читання необхідних даних (рис.33).

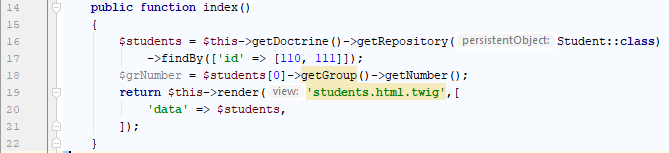


Рис.31. Зміни у контролері.



Рис.32. Відображення реального номера групи.

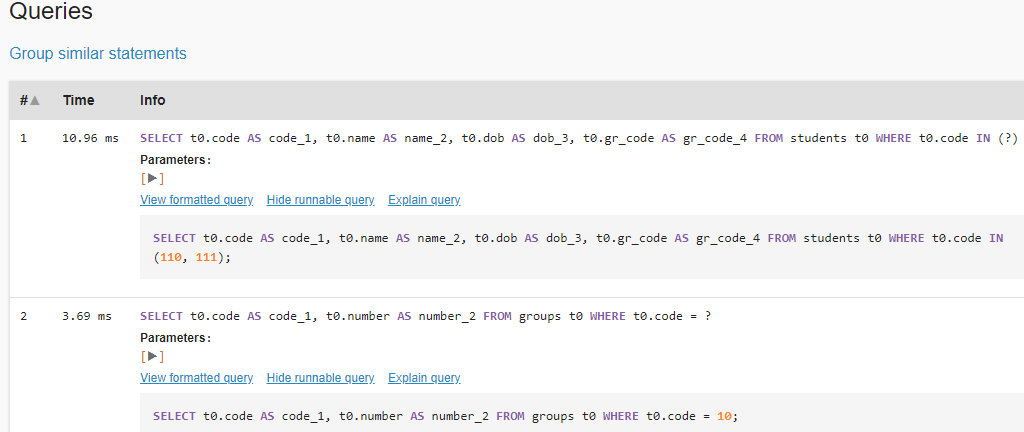


Рис.33. Новий запит до БД.

1. Доповнимо клас-сутність «групи» так, щоб використовуючи об’єкт даного класу вже містив у собі список студентів даної групи. Для виконання даної задачі слід просто додати до класу «група» нове поле-колекцію та вказати його зв’язок із полем «група» класу «студент» (рис.34).



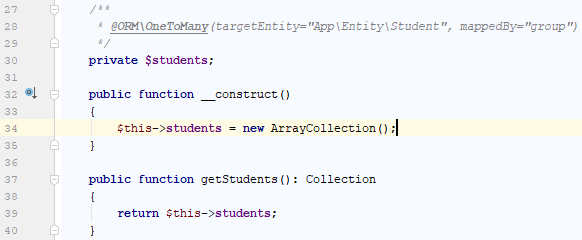


Рис.34. Зміни у класі «група».

У класі «студенти також бажано встановити зворотній зв’язок із цим полем (рис.35)



Рис.35. Зміни у класі «студент».

Тепер у контролері виконаємо читання групи та всіх її студентів (рис.36) та переглянемо результат (рис.37).

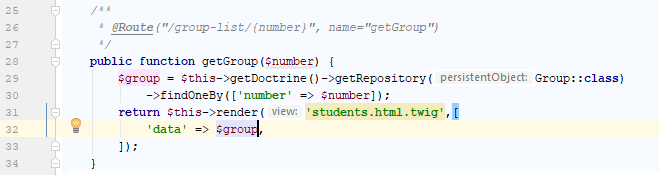


Рис.36. Читання даних щодо студентської групи

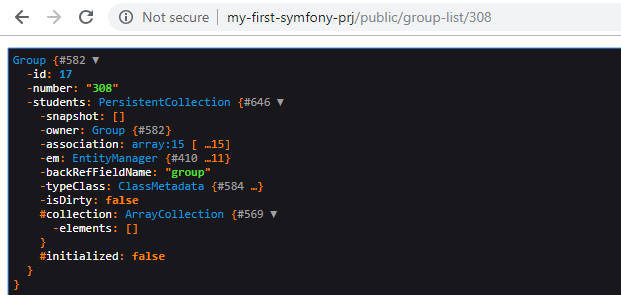


Рис.37. Студентська група.

Звернемо увагу, що колекція студентів та даний момент не містить елементів, як у попередньому випадку із порожнім номером групи. Причина цьому та ж сама – ми не звертались до елементів колекції, отже Doctrine вважає, що вони поки нам не потрібні та не читає їх із БД. Пересвідчимось у цьому, переглянувши список запитів до БД (рис.38)

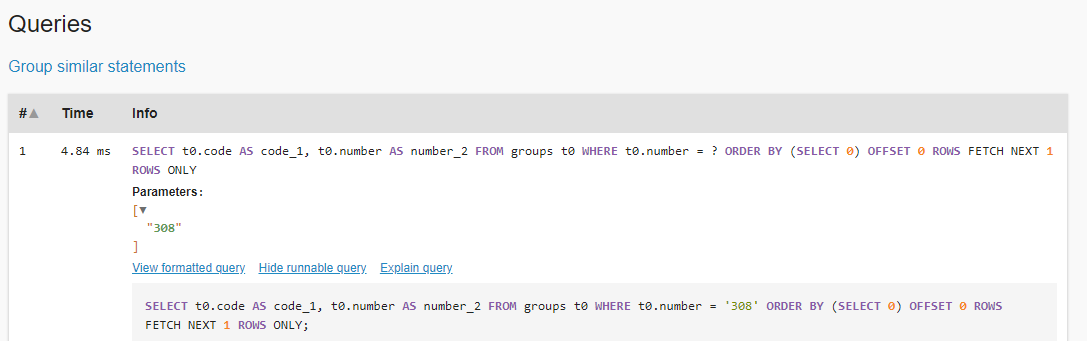


Рис.38. Запити на читання студентської групи.

Виправимо картину та виконаємо звернення у контролері до кількості елементів колекції студентів групи (рис.39). Переглянемо результат (рис.40) та список запитів до БД (рис.41)

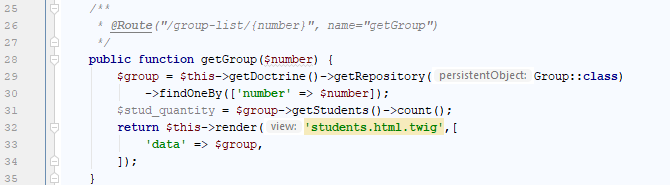


Рис.39. Читання кількості студентів із колекції.

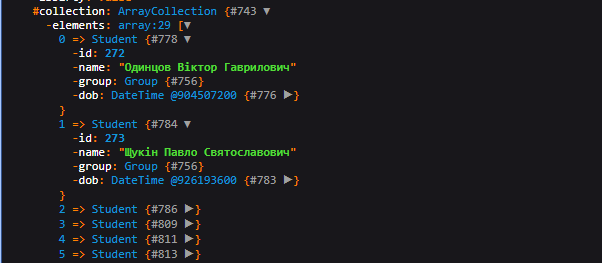


Рис.40. Результат – колекція із списком студентів.

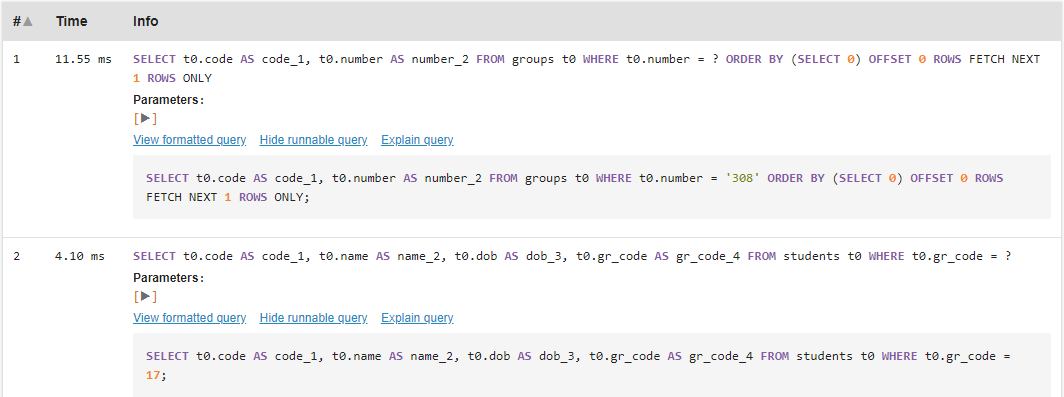


Рис.41. Список запитів до БД.

1. Кінець роботи.

**Завдання для індивідуального виконання.**

Автоматично створити клас-сутність на базі існуючої таблиці власної БД. Створити новий клас-сутність, якому не відповідає жодна з таблиць БД та імплементувати зміни у БД. Реалізувати контролер, що виконує операції читання, додавання, зміни та видалення даних.

Виконати реалізацію 2-х класів-сутностей для роботи із зв’язаними табличками, та виконати із ними операції читання даних.