**Лабораторна робота 21.**

**Використання ORM Doctrine при роботі із SQL Server.**

**Мета роботи:** Познайомитись із поняттям ORM та навчитись використовувати ORM Doctrine при роботі із SQL Server

**Задачі:**

* Познайомитись із поняттям ORM та Doctrine
* Навчитись встановлювати ORM Doctrine, використовуючи Composer
* Створити тестове підключення до БД
* Виконати операції вибірки, зміни, додавання та видалення інформації
* Виконання запитів DQL
* Реалізувати виконання нативного запиту SQL із перетворенням результату до об’єктного вигляду
* Використання драйверу з’єднання із БД для реалізації роботи через PDO
* Виконати індивідуальне завдання згідно свого варіанту

1. На даний момент найбільш розповсюджене та популярне рішення проблеми зберігання даних - це реляційні СУБД. Однак використання реляційної БД для зберігання об'єктно-орієнтованих даних призводить до семантичного розриву, змушуючи програмістів писати програмне забезпечення, яке повинно вміти як обробляти дані в об'єктно-орієнтованому вигляді, так і вміти зберегти ці дані в реляційної формі.

Ця постійна необхідність в перетворенні між двома різними формами даних не тільки сильно знижує продуктивність, але і створює труднощі для програмістів, так як обидві форми даних накладають обмеження один на одного.

ORM (Object-Relational Mapping – об'єктно-реляційне відображення, або перетворення) - технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних».

Розроблено багато пакетів, що усувають необхідність в перетворенні об'єктів для зберігання в реляційних базах даних. З точки зору програміста система повинна виглядати як постійне сховище об'єктів. Він може просто створювати об'єкти і працювати з ними як зазвичай, а вони автоматично будуть зберігатися в реляційній базі даних.

Doctrine - об'єктно-реляційний проектор (ORM) для PHP, який базується на шарі абстракції доступу до БД (DBAL). Однією з ключових можливостей Doctrine є можливість реалізації запитів до БД на власному об'єктно-орієнтованому діалекті SQL, що має назву DQL (Doctrine Query Language) і базується на ідеях HQL (Hibernate Query Language).

1. Установку ORM Doctrine зручніше за все виконувати із використанням програмної утиліти composer. Composer - це пакетний менеджер рівня застосунків для мови програмування PHP, який надає засоби для управління залежностями в PHP-застосунку. Composer працює через інтерфейс командного рядка і встановлює залежності (наприклад бібліотеки) для програми.

!УВАГА. Якщо даний пункт викликає складності або немає наявного з’єднання із мережею internet, всі необхідні файли можуть бути скопійовані до проекту із папки з лабораторними роботами.

Для перевірки, чи встановлено та налаштовано composer на вашому комп’ютері, виконаємо у командному рядку наступну команду (рис.1)

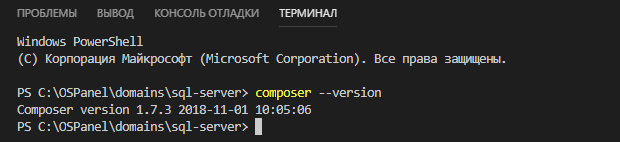


Рис.1. Виведення версії composer

Інструкцію щодо встановлення та налаштування composer можна отримати на офіційному сайті «https://getcomposer.org/».

Далі, знаходячись у папці із нашим проектом «sql-server», виконаємо команду для установки ORM Doctrine (рис.2)

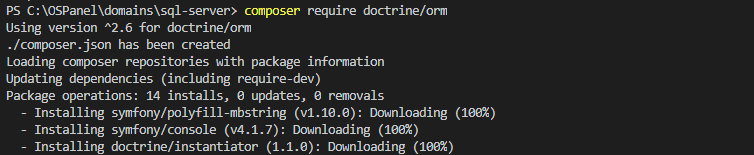


Рис.2. Установка ORM Doctrine через composer

Після установки виконаємо деякі налаштування для подальшого автоматичного підключення класів, що будуть описувати наші сутності (таблички). Виконаємо зміну composer.json у корньовій папці проекту (рис.3) та запустимо команду для оновлення autoload.php (рис.4)

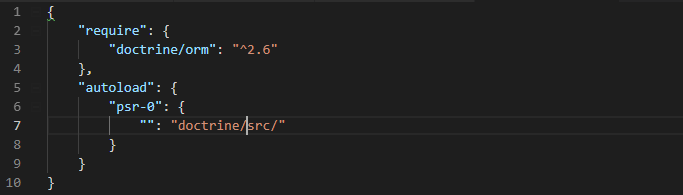
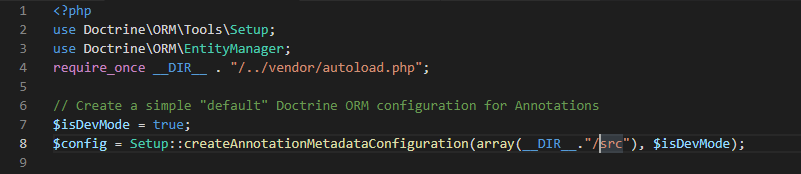


Рис.3. Зміни у composer.json.



Рис.4. Оновлення autoload.php

1. Далі реалізуємо стартові файли для завантаження ORM Doctrine та виконання первинних налаштувань. У папці із проектом створимо директорію «doctrine» а у ній «doctrine/src». У папці «doctrine» створимо файл «doctrine-bootstrap.php» (рис.5)



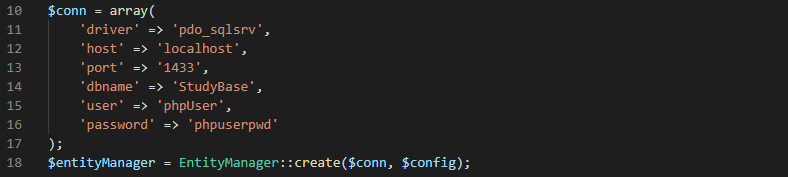
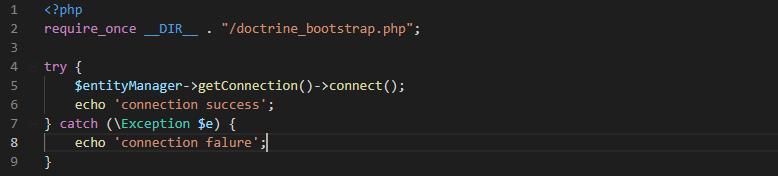


Рис.5. Вміст файлу «doctrine-bootstrap.php»

Далі виконаємо тестове підключення до бази даних. У папці «doctrine» створимо файл «test-doctrine.php» (рис.6)



Перевіримо підключення, набравши у командному рядку «php test-doctrine.php» (рис.7)



Рис.7. Перевірка підключення до БД

Перевірку також можна виконати без командного рядку із використанням інтернет-браузера (рис.8)

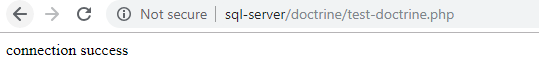


Рис.8. Перевірка підключення до БД із використанням інтернет-браузера

Далі виконаємо помилку у імені БД (рис.9) та виконаємо перевірку підключення повторно (рис.10)

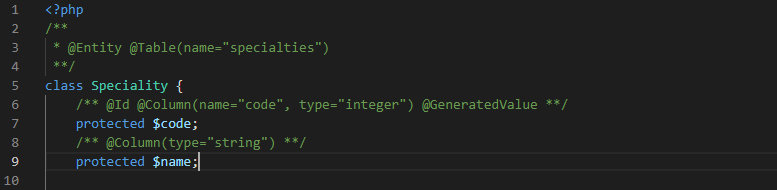


Рис.9. Вказання неіснуючої БД



Рис.10. Помилка підключення до БД

1. Отже, ми встановили ORM Doctrine та налаштували підключення до нашої БД. Далі створимо клас, що буде посередником між реляційною та об’єктною моделями при роботі із сутністю «спеціальність». Дана сутність має 2 поля – код на назва та зберігається у БД у табличці «specialties». Для цього у папці «doctrine/data» створимо файл «Speciality.php» наступного змісту (рис.11)



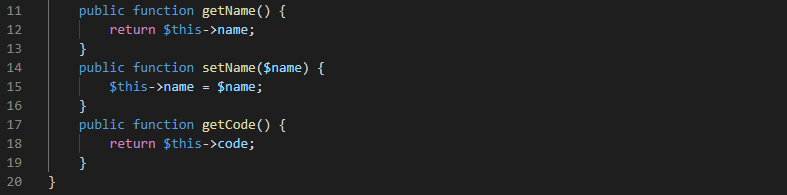


Рис.11. Клас, що описує сутність «спеціальності»

Далі, доповнимо файл «test-doctrine.php» кодом, що буде використовуючи новостворений клас виконувати читання із БД спеціальності за значенням коду. Код спеціальності передамо, як параметр командного рядку (рис.12-13)

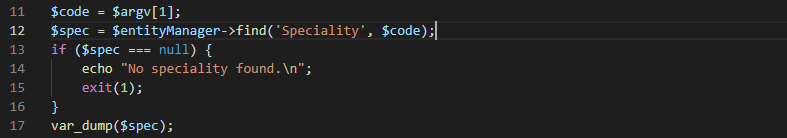


Рис.12. Доповнення «test-doctrine.php» для читання спеціальності за кодом

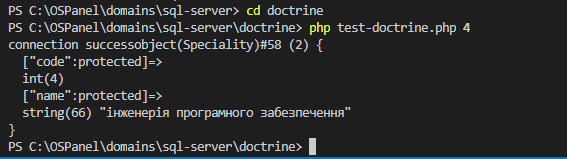


Рис.13. Перевірка читання спеціальності за кодом.

Як бачимо, вся інформація, що стосується БД була передана у «Speciality.php», і тепер ORM Doctrine бере на себе роботу по генерацію SQL-коду для отримання даних із таблиці або таблиць. Даний приклад також може бути реалізований у браузері без використання командного рядку. Для цього змінимо метод отримання коду із параметру командного рядку на get-запит (рис.14), та наберемо у командному рядку браузера «http://sql-server/doctrine/test-doctrine.php?code=4» (рис.15)



Рис.14. Зміна параметру командного рядку на get-запит

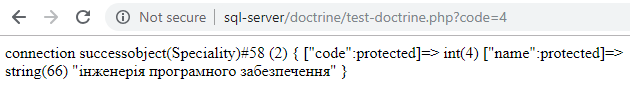


Рис.15. Перевірка вибірки даних у браузері

1. Тепер виконаємо вибірку не за кодом, а за назвою. Для пошуку за первинним ключем використовувався метод find(), але є більш загальний метод findBy() (рис.16-17).

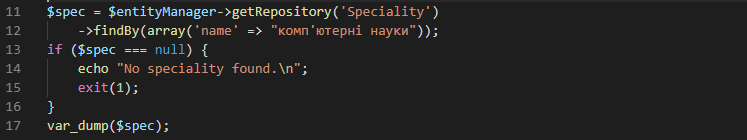


Рис. 16. Код для вибірки спеціальностей за назвою

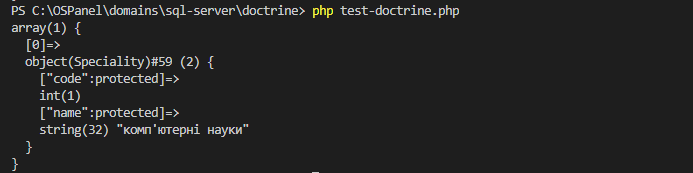


Рис.17. Результат вибірки спеціальностей за назвою

Слід пам’ятати, що findBy(), на відміну від find(), повертає масив об’єктів, оскільки результатом вибірки може бути декілька елементів. Якщо потрібно повернути один елемент, можна скористатись методом findOneBy() (рис.18-19).



Рис. 18. Код для вибірки однієї спеціальності за назвою

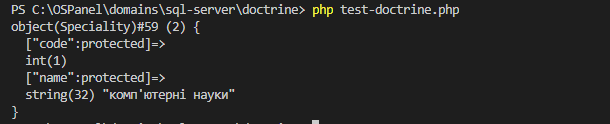


Рис.19. Результат вибірки однієї спеціальності за назвою

Виконаємо більш цікаву вибірку – спеціальності, у назву яких входить слово «комп'ютер». На цьому прикладі також буде проілюстровано використання мови DQL (doctrine query language) (рис.20-21).

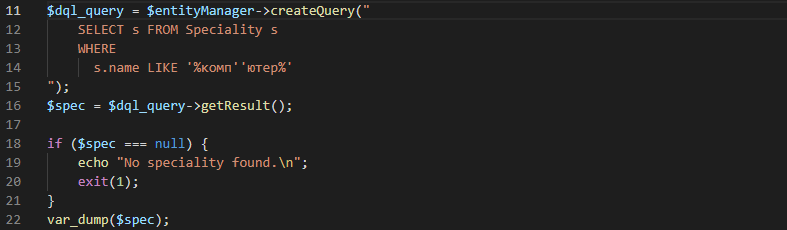


Рис.20. Вибірка спеціальностей, у назву яких входить слово «комп'ютер» із використанням DQL

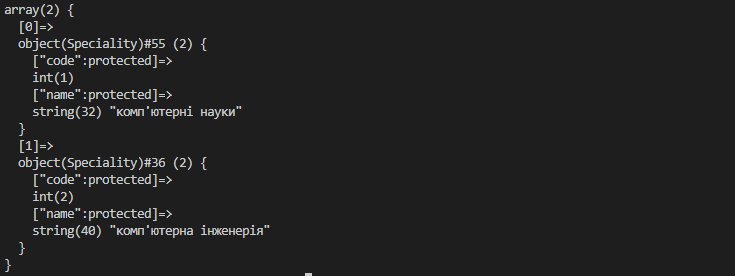


Рис.21. Результат виконання вибірки із використанням DQL

На перший погляд DQL дуже схожий на SQL, ключова відмінність у даному прикладі у тому, що DQL може повертати результат у вигляді об’єктів.

1. Далі виконаємо команди додавання, зміни та видалення даних із використанням ORM Doctrine. Додамо до БД нову спеціальність (рис.22)

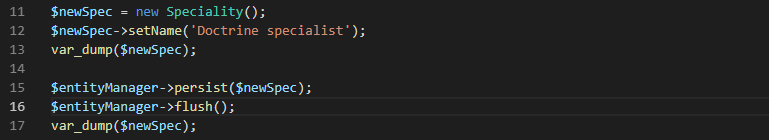


Рис.22. Додавання нової спеціальності «Doctrine specialist»

Для додавання спочатку створюється екземпляр класу Speciality та заповнюється його атрибут name. Далі використовується метод persist, що вказує doctrine на те, що цей об’єкт є частиною БД, і його зміни мають бути перенесені у БД. Метод flush() виконує запис у БД всіх змін, що були виконані протягом сеансу. Як бачимо на рис.32, новостворений об’єкт не мав значення у полі code, але воно з’являється після запису об’єкту до БД.

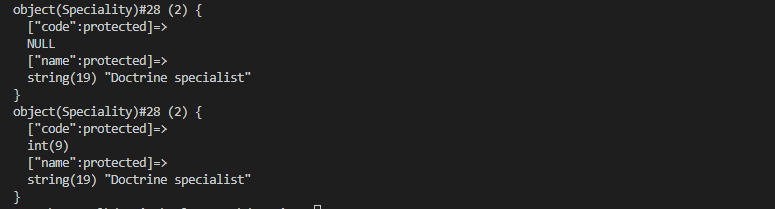


Рис.23. Новостворений об’єкт до та після запису до БД.

Тепер виконаємо зміну даних (рис.24). Можна звернути увагу, що на вірміну від попереднього прикладу, тут ми не використали метод persist. У даному випадку цього робити не обов’язково, тому що об’єкт був отриманий читанням із БД, і тому Doctrine автоматично відслідковуватиме для нього всі виконані зміни та запише їх до БД при виконанні наступної операції flush(). Результат зміни наведено на рис.24.

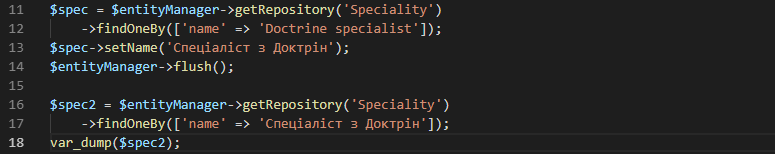


Рис.24. Зміна об’єкту із наступним записом до БД.

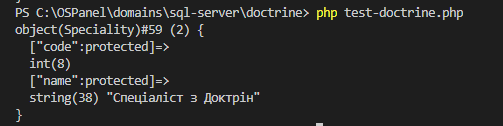


Рис.25. Результат зміни даних

І остання операція – видалення об’єкту із БД. Її реалізація навориться на рис.26. Наступна вибірка повертає NULL, оскільки у БД вже намає такого рядку (рис.27)

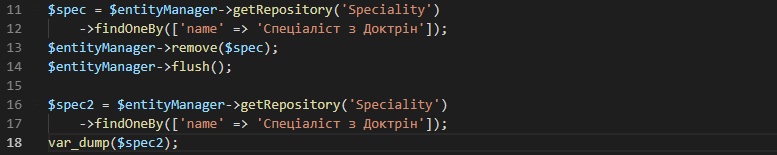
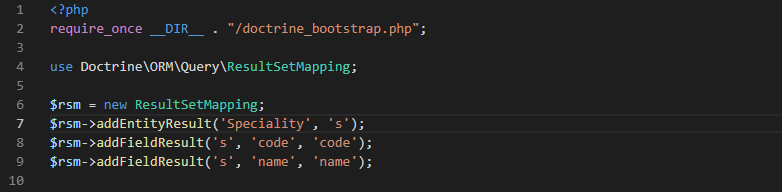


Рис.26. Видалення об’єкту із БД.



Рис.27. Результат роботи скрипту видалення об’єкту із БД

1. Тепер розглянемо можливість виконання нативних SQL-запитів із використанням менеджера сутностей Doctrine. Для цього виконаємо вибірку спеціальностей, на яких є всього одна студентська група та повернення їх у вигляді списку об’єктів. Реалізація даної задачі наводиться на рис.28. На рис.29 наведено результат вибірки.



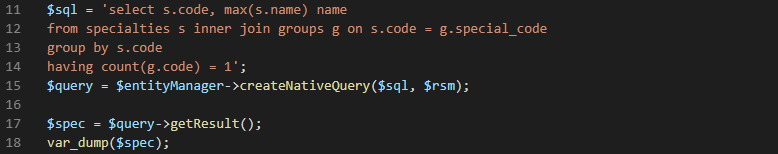


Рис.28. Нативний SQL-запит із перетворенням результату до об’єктного виду

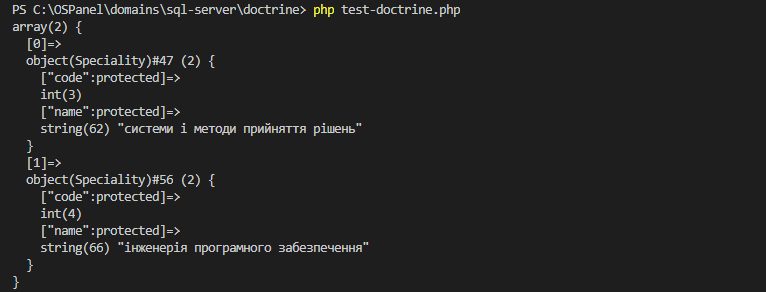


Рис.29. Результат роботи нативного SQL-запиту

1. Далі продемонструємо можливість роботу із БД напряму через драйвер з’зднання, що використовує Doctrine. Для цього можна використати методи getConnection() та getWrappedConnection() менеджера сутностей. У якості прикладу виконаємо збережену процедуру із вхідними та вихідними параметрами та декількома результуючими наборами, що було реалізовано у попередніх роботах (рис.30)

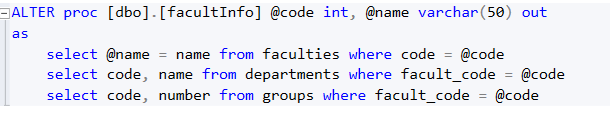
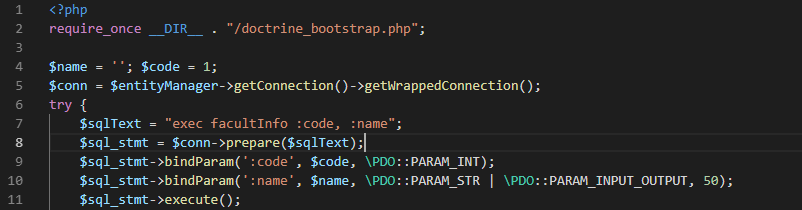


Рис.30. Процедура, що має вхідні та вихідні параметри та декілька результуючих наборів.

Код прикладу, що отримує з’єднання із БД та далі реалізовує виконання процедури через звичайний PDO, наведено на рис.31). Як бачимо, реалізація є дуже схожою на виконану у попередній роботі.



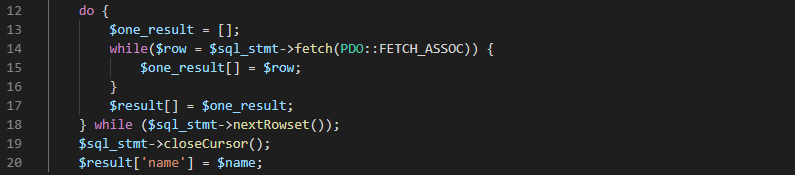


Рис.31. Робота напряму через драйвер з’єднання із БД

1. Кінець роботи.

**Завдання для індивідуального виконання.**

Для однієї з таблиць БД свого варіанту реалізувати скрипти із використанням ORM Doctrine, що реалізують вибірку, додавання, зміну та видалення даних.

Також для власної БД реалізувати виконання запиту мовою DQL, нативного SQL запиту та запиту із використанням драйверу з’єднання та PDO.