**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра <Інтелектуальних інформаційних систем>**

**ЗВІТ**

*з лабораторної роботи №  3*

**Варіант №21**

Дисципліна "**Java EE**"

Спеціальність: 122

122-ЛР.ПЗ.03-1810121

***Cтудент***

*\_\_\_\_\_\_\_\_\_\_\_\_\_ М.О. Салютiн*

*(підпис)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

***Викладач*** *\_\_\_\_\_\_\_\_\_\_\_\_\_К.О.Антiпова*

*(підпис)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Миколаїв – 2020**

## **Лабораторна робота №3**

**Persistence і Validation API в Java EE 7.0**

Завдання:

1. За допомогою Persistence API створити CRUD функції для роботи з базою даних згідно з варіантом.
2. Розділити прототип програми для моделі MVC згідно з варіантом на шари – контролери, робота з бд, інтерфейс.
3. Продемонструвати використання анотацій @Entity, @Table, @SecondaryTable, @Id, @GeneratedValue, @Basic, @Column, @Enumerated.
4. Задіяти не менше 5 JPQL-інструкцій в проекті, створити кілька іменованих запитів.
5. За допомогою Validation API продемонструвати використання та застосування валідаційних обмежень, при цьому задіяти, як вбудовані @Pattern, @Size, @NotNull, так і власні обмеження.

1. JPA надає інтерфейс javax.persistence.EntityManager, який використовується для взаємодії з базою даних. Екземпляр EntityManager відтворює контекст стійкості, а EntityManagerFactory взаємодіє з фабрикою менеджера сутності.

Створюємо файл persistence.xml і додаємо до нього такий код:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<persistence version=*"2.1"*

xmlns=*"http://xmlns.jcp.org/xml/ns/persistence"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd"*>

<persistence-unit name=*"JSFJPACrud"*>

<class>com.jsf.jpa.crud.SchoolEntityManager</class>

<!-- Configuring JDBC Connection -->

<properties>

<property name=*"javax.persistence.jdbc.driver"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"javax.persistence.jdbc.url"* value=*"jdbc:mysql://localhost:3306/tutorialDb"* />

<property name=*"javax.persistence.jdbc.user"* value=*"root"* />

<property name=*"javax.persistence.jdbc.password"* value=*""* />

</properties>

</persistence-unit>

</persistence>

Використовуємо нашу стару базу даних hospitalbd:

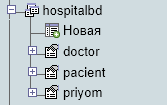


Рис.1 – база даних hospitalbd

Тут, у doctorList.xhtml, ми матимемо компонент інтерфейсу jsf, який відображатиме список докторiв, завантажений безпосередньо зі шкільного столу при завантаженні програми. Тут компоненти bootstrap реалізують класи CSS, які ми вдосконалимо в інтерфейсі програми. Кнопки Додати нового доктору, Оновити та Видалити відображатимуть відповідні результати на основі написаної внутрішньої логіки. Додаемо файли doctorList.xhtml, doctorEdit.xhtml, newDoctor.xhtml.

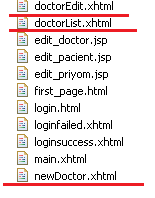


Рис.2 – доданi файли

Створення класiв Java:

Для цього ми створюємо два package в src:



Рис.3 – package LB3

В них знаходяться 3 класи:

DoctorBean.java, DoctorEntityManager.java, DataBaseOperations.java.

Впровадження класу EntityManager

У цьому класі є всі геттери / сеттери для таблицi доктору. Угорі цього класу ми повідомляємо компілятору, що назва таблиці - doctor. Ми оголошуємо клас Entity за допомогою анотації @Entity, а потім використовуємо анотацію @Id для створення ідентифікатора стовпця id таблиці.

Впровадження керованого класу Bean

Цей клас має методи, які взаємодіють із подіями дій, натиснутими на сторінках користувальницького інтерфейсу, і відображають результат на вихідній сторінці на основі логіки навігації, повернутої з класу DAO.

Впровадження класу DAO

Цей клас має методи, які взаємодіють з базою даних для різних операцій, тобто створюють новий шкільний запис, редагують або оновлюють існуючий шкільний запис або видаляють запис із бази даних.

Результат роботи:

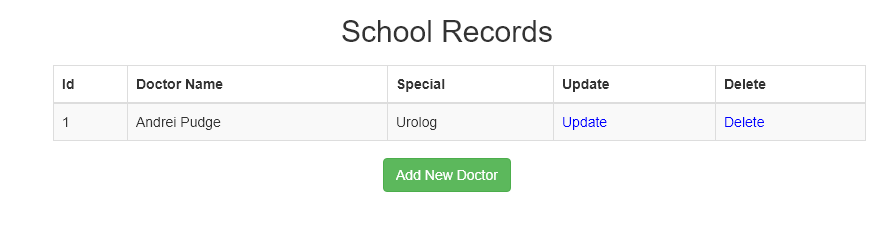


Рис. 4 – таблиця doctor

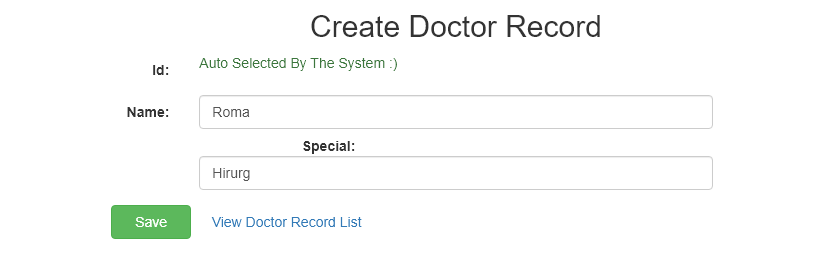


Рис. 5 – додавання нового доктору.

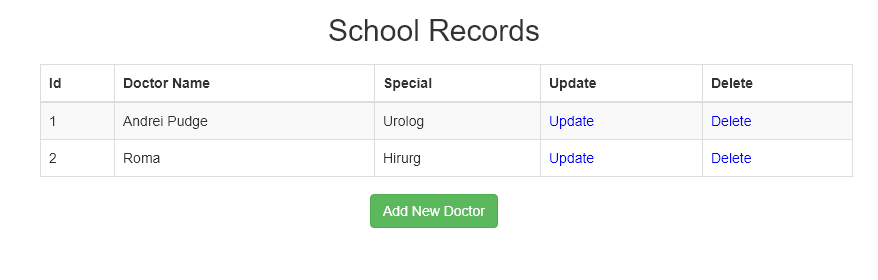


Рис.6 – результат додавання доктору.

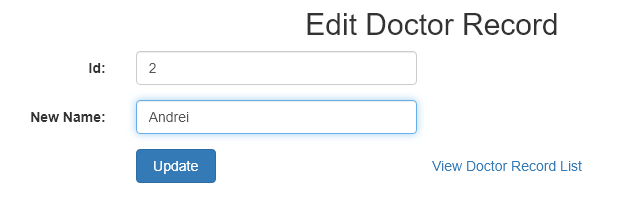


Рис.7 – редагування доктору.

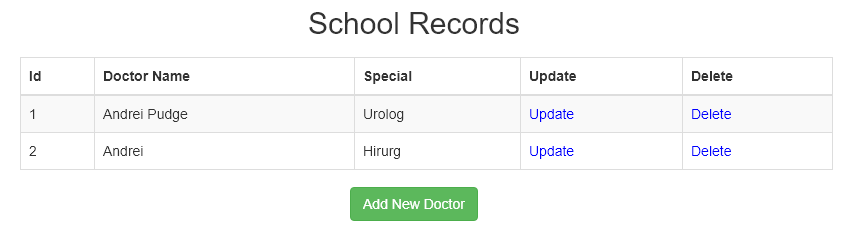


Рис.8 – результат редагування доктору.

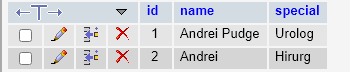
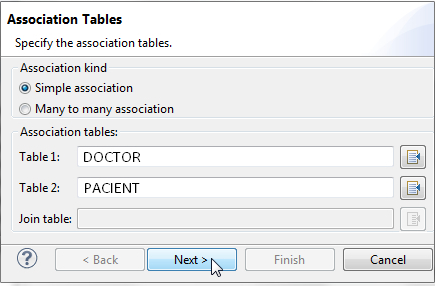
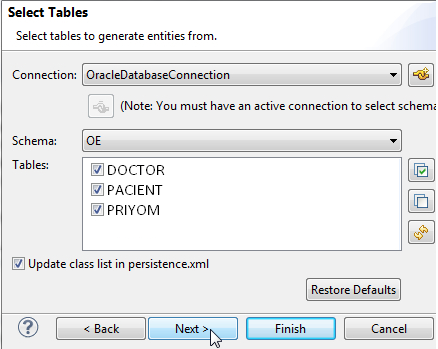


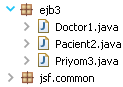
Рис.9 – результат в phpMyAdmin.

Продемонструвати використання анотацій @Entity, @Table, @SecondaryTable, @Id, @GeneratedValue, @Basic, @Column, @Enumerated.

Пiдключення JPA Entity



Створюємо 3 класу Java:



Створення файлу конфігурації зберігання JPA

У файлі конфігурації persistence.xml вказується постачальник зберігання, який буде використовуватися для збереження сутностей в базі даних. Одиниця збереження стану вказується за допомогою елемента одиниці збереження стану. Тип транзакції за замовчуванням встановлений на JTA.

Файл persistence.xml:

<?xml version="1.0" encoding="UTF-8"?>

<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence\_1\_0.xsd">

<persistence-unit name="em" transaction-type="JTA">

<provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>

<jta-data-source>jdbc/OracleDS</jta-data-source>

<class>ejb3.Doctor1</class>

<class>ejb3.Pacient2</class>

<class>ejb3.Priyom3</class>

<properties>

<property name="eclipselink.target-server" value="WebLogic\_10"/>

<property name="eclipselink.target-database" value="Oracle"/>

<property name="eclipselink.ddl-generation" value="create-tables"/>

<property name="eclipselink.logging.level" value="FINEST"/>

</properties>

</persistence-unit>

</persistence>

У класах використовуємо анотації:

@Entity - JPA вказує що вона може працювати як з властивостями класів (property), оформлені в стилі JavaBeans, або з полями (field), тобто змінними класу (instance variables).

@NamedQueries - це статично визначений запит із заздалегідь визначеним незмінним рядком запиту. Використання іменованих запитів замість динамічних запитів може поліпшити організацію коду, відокремлюючи рядки запитів JPQL від коду Java.

@NamedQuery - анотація @NamedQuery містить чотири елементи - два з яких є обов’язковими, а два необов’язковими. Два обов’язкові елементи, ім’я та запит, визначають назву запиту та сам рядок запиту і продемонстровано вище. Два додаткові елементи, lockMode та підказки, забезпечують статичну заміну

@Id - Призначений стовпець первинного ключа сутності вважається первинним ключем первинної таблиці. Якщо анотація стовпця не вказана, ім'ям стовпця первинного ключа вважається ім'я властивості або поля первинного ключа.

@Column - Використовується для вказівки відображеного стовпця для постійного властивості або поля. Якщо анотація стовпця не вказана, застосовуються значення за замовчуванням.

@SequenceGenerator - Визначає генератор первинного ключа, на який може посилатися по імені, коли елемент генератора вказаний для анотації GeneratedValue.

@GeneratedValue - GeneratedValue анотація може бути застосована до властивості первинного ключа або полю об'єкта або відображеного суперкласу в з'єднанні з Id анотацією.

@JoinTable - Використовується в відображенні асоціацій. Це визначається на стороні володіння асоціаціями.

@Table - Визначає первинну таблицю для коментованої сутності. Додаткові таблиці можуть бути вказані за допомогою анотації @SecondaryTable або @SecondaryTables.

**Задіяти не менше 5 JPQL-інструкцій в проекті, створити кілька іменованих запитів.**

Створюємо package jpql, в якому будемо створювати класи для управління БД:

****

Для роботи нам потрібно встановити javax.persistence-2.0:

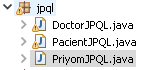
****

****

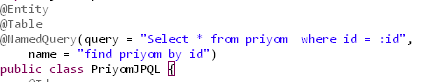
Результат:



Так само створюємо три класи: DoctorJPQL, PacientPJPQL, PriyomJPQL.



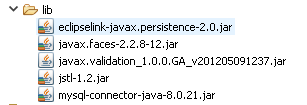
При створенні використовувалося @NamedQuery:

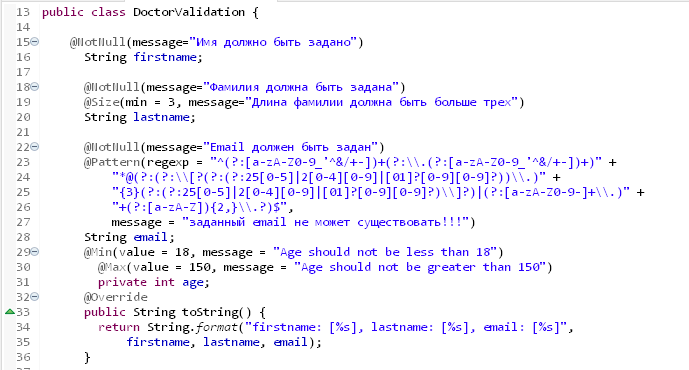


Так само були реалізовані @NamedQueries у класi NamedQueries.java:

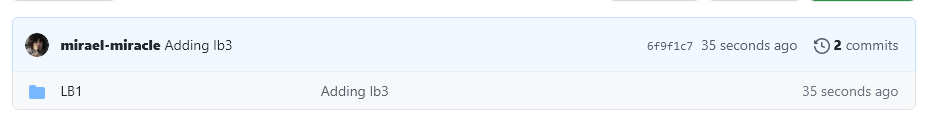


Для виконання Validation Bean треба завантажити бiблiотеку та закинути до папки lib:



Після цього створюємо package "validation" і завантажуємо туди клас DoctorValidation, в якому є анотації: @NotNull, @Size, @Pattern з регулярним виразом, @Min, @Max. 

Завантажимо проект на github:



Посилання: <https://github.com/mirael-miracle/javaEE.git>