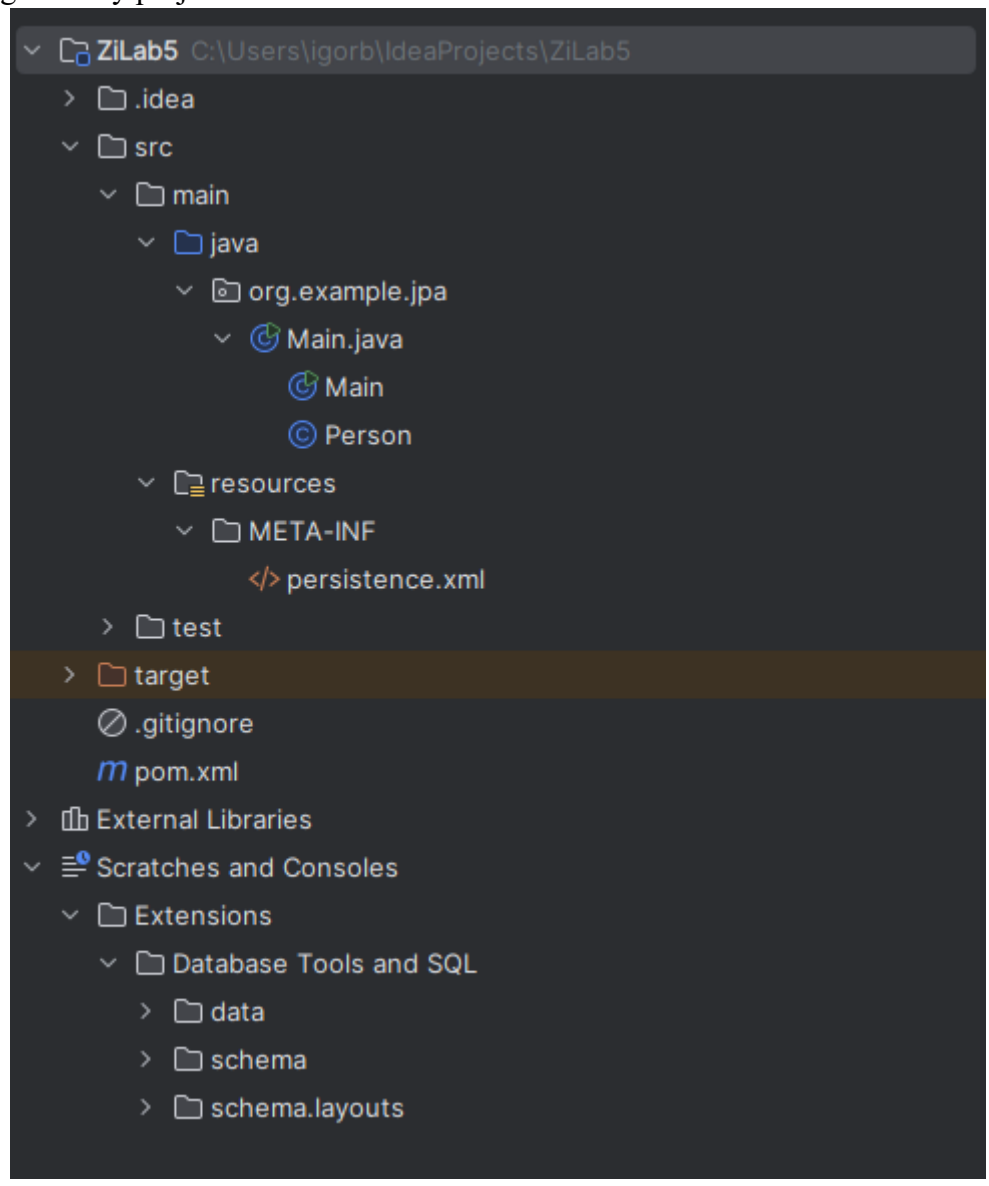
Zarządzanie Informacją 2

Sprawozdanie z Laboratorium 5
Igor Bębenek
Grupa 332

Skonfigurowany projekt



Klasa encji z danymi osobowymi

Stworzyliśmy klasę Person z adnotacją @Entity dzięki czemu od razu stworzyliśmy tabelę w bazie danych. Dzięki hibernate.hbm2ddl.auto w persistence.xml ustawionym value na update tabela tworzona jest automatycznie. @Id określa pole id jako klucz

główny a @GeneratedValue(strategy = GenerationType.Identity) nadaje id dla każdej nowej osoby. Gettery i setery wymagane są żeby mieć dostęp do pól podczas zapisu, odczytu, aktualizacji w bazie.

```java
package org.example.jpa;

import jakarta.persistence.*;
import java.util.List;
import java.util.Random;

@Entity
class Person {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String familyName;
    private Integer age;

    public String getFirstName() { return firstName; }
    public void setFirstName(String firstName) { this.firstName = firstName; }
    public String getFamilyName() { return familyName; }
    public void setFamilyName(String familyName) { this.familyName = familyName; }
    public Integer getAge() { return age; }
    public void setAge(Integer age) { this.age = age; }
}
```

W metodzie main tworzymy obiekt EntityManagerFactory, a następnie EntityManager do wykonywania operacji na bazie.

Rozpoczynamy potem transakcję i tworzymy w pętli 5 obiektów Person ustawiając dla nich Imię, Nazwisko, ustawiamy wiek na losową wartość z przedziału od 1 do 50. Na końcu obiekt jest zapisywany w bazie danych dzięki em.persist(person).

Po pętli zatwierdzamy transakcje i zapisujemy w bazie danych zmiany.

```java
public class Main {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory( persistenceUnitName: "xd");
        EntityManager em = emf.createEntityManager();

        em.getTransaction().begin();
        for (int i = 0; i < 5; i++) {
            Person person = new Person();
            person.setFirstName("Imie" + i);
            person.setFamilyName("Nazwisko" + i);
            person.setAge(new Random().nextInt( bound: 50) + 1);
            em.persist(person);
        }
        em.getTransaction().commit();
```

Tworzymy zapytanie SELECT p FROM Person p żeby wczytać obiekty Person z bazy. Korzystamy z getResultList() pobierając wyniki jako liste obiektow Person. Potem wyświetlamy foreachem na konsoli listę persons z imiennie, nazwiskiem oraz wiekiem.

Rozpoczynamy transakcje żeby zacząć modyfikacje obiektow Person. Iterujemy przez liste i zmieniamy wiek na 18 jeżeli jest on mniejszy niż 18. Zapisujemy obiekt za pomocą em.merge(p). Zatwierdzamy transakcje metodą .commit()

```
List<Person> persons = em.createQuery( s: "SELECT p FROM Person p", Person.class).getResultList();
for (Person p : persons) {
    System.out.println(p.getFirstName() + " " + p.getFamilyName() + ", Wiek: " + p.getAge());
}

em.getTransaction().begin();
for (Person p : persons) {
    if (p.getAge() < 18) {
        p.setAge(18);
        em.merge(p);
    }
}
em.getTransaction().commit();
```

Wczytujemy wszystkie obiekty Person z bazy selectem, używamy getResultList żeby zaktualizować listę persons.

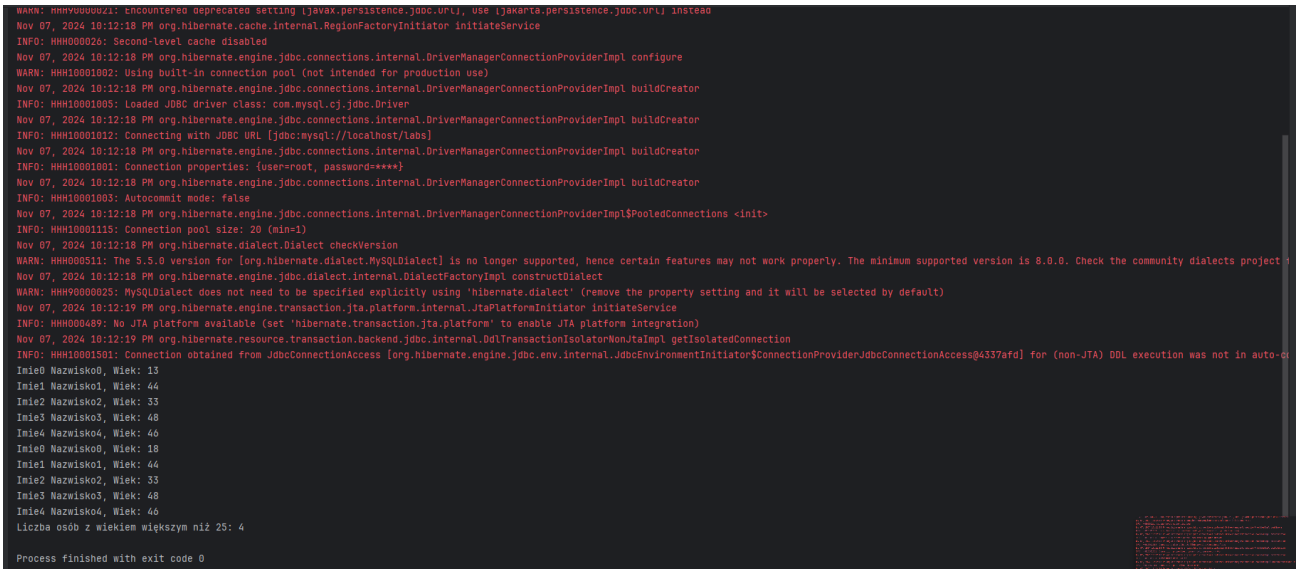
Iterujemy przez listę foreachem i wypisujemy Imie, Nazwisko oraz Wiek.

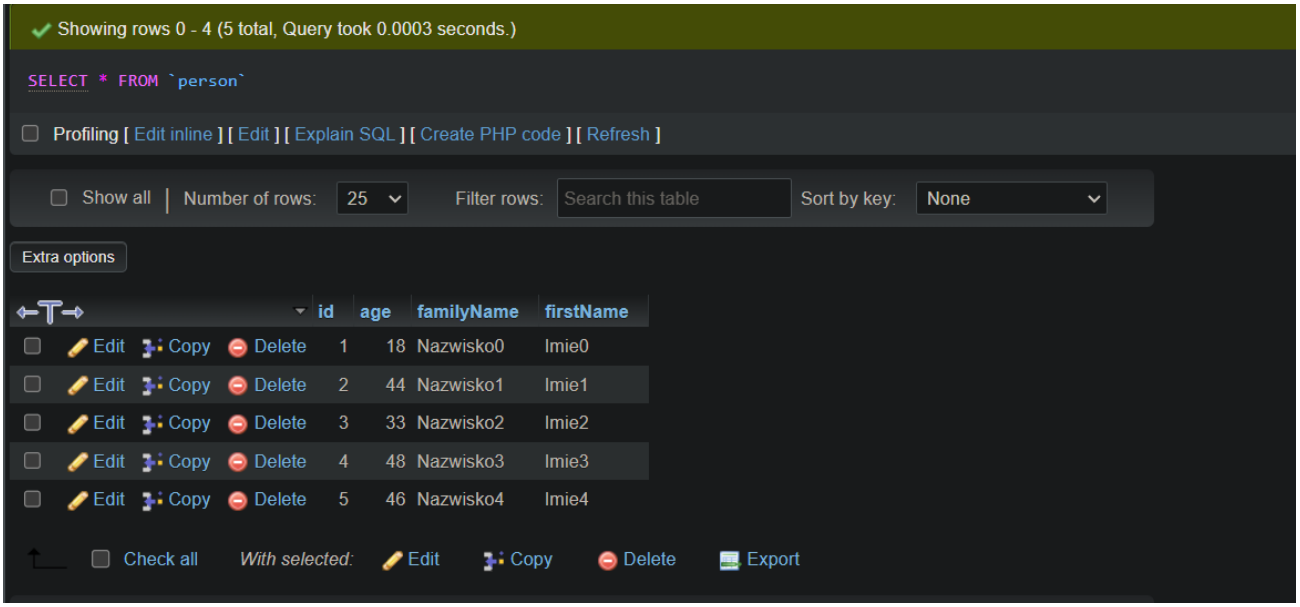
Na koniec tworzymy zapytanie w którym zliczamy obiekty Person, które maja wiek większy niż 25, Count(p) zwraca liczbę spełniających warunek wyników. getSingleResult zwraca jeden wynik typu obiekt Long, który rzutujemy na long.

```
    persons = em.createQuery( s: "SELECT p FROM Person p", Person.class).getResultList();
    for (Person p : persons) {
        System.out.println(p.getFirstName() + " " + p.getFamilyName() + ", Wiek: " + p.getAge());
    }

    long count = (long) em.createQuery( s: "SELECT COUNT(p) FROM Person p WHERE p.age > 25").getSingleResult();
    System.out.println("Liczba osób z wiekiem większym niż 25: " + count);
}
```

Wyniki z konsoli:


WARN: HHH90000021: Encountered deprecated setting [javax.persistence.jdbc.url], use [jakarta.persistence.jdbc.url] instead
Nov 07, 2024 10:12:18 PM org.hibernate.cache.internal.RegionFactoryInitiator initiateService
INFO: HHH000026: Second-level cache disabled
Nov 07, 2024 10:12:18 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl configure
WARN: HHH10001002: Using built-in connection pool (not intended for production use)
Nov 07, 2024 10:12:18 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001005: Loaded JDBC driver class: com.mysql.cj.jdbc.Driver
Nov 07, 2024 10:12:18 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001012: Connecting with JDBC URL [jdbc:mysql://localhost/labs]
Nov 07, 2024 10:12:18 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001001: Connection properties: {user=root, password=****}
Nov 07, 2024 10:12:18 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl buildCreator
INFO: HHH10001003: Autocommit mode: false
Nov 07, 2024 10:12:18 PM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections <init>
INFO: HHH10001115: Connection pool size: 20 (min=1)
Nov 07, 2024 10:12:18 PM org.hibernate.dialect.Dialect checkVersion
WARN: HHH000511: The 5.5.0 version for [org.hibernate.dialect.MySQLDialect] is no longer supported, hence certain features may not work properly. The minimum supported version is 8.0.0. Check the community dialects project
Nov 07, 2024 10:12:18 PM org.hibernate.engine.jdbc.dialect.internal.DialectFactoryImpl constructDialect
WARN: HHH90000025: MySQLDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)
Nov 07, 2024 10:12:19 PM org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
Nov 07, 2024 10:12:19 PM org.hibernate.resource.transaction.backend.jdbc.internal.DdlTransactionIsolatorNonJtaImpl getIsolatedConnection
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator$ConnectionProviderJdbcConnectionAccess@4337afd] for (non-JTA) DDL execution was not in auto-c
Imie0 Nazwisko0, Wiek: 13
Imie1 Nazwisko1, Wiek: 44
Imie2 Nazwisko2, Wiek: 33
Imie3 Nazwisko3, Wiek: 48
Imie4 Nazwisko4, Wiek: 46
Imie0 Nazwisko0, Wiek: 18
Imie1 Nazwisko1, Wiek: 44
Imie2 Nazwisko2, Wiek: 33
Imie3 Nazwisko3, Wiek: 48
Imie4 Nazwisko4, Wiek: 46
Liczba osób z wiekiem większym niż 25: 4
Process finished with exit code 0


Showing rows 0 - 4 (5 total, Query took 0.0003 seconds.)
SELECT * FROM `person`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options
id age familyName firstName
Edit Copy Delete 1 18 Nazwisko0 Imie0
Edit Copy Delete 2 44 Nazwisko1 Imie1
Edit Copy Delete 3 33 Nazwisko2 Imie2
Edit Copy Delete 4 48 Nazwisko3 Imie3
Edit Copy Delete 5 46 Nazwisko4 Imie4
Check all With selected: Edit Copy Delete Export