# Konfiguracja bazy danych H2

To baza danych przechowywana w pamięci. Resetuje się za każdym razem gdy uruchamiamy nasz program od nowa.

1)dodanie do POM xml odpowiedniej zależności

<!-- https://mvnrepository.com/artifact/com.h2database/h2 -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>1.0.60</version>

</dependency>

2)podłączenie do bazy danych

**Klasa AppH2**

**private static** String *h2Driver*=**"org.h2.Driver"**;  
**private static** String *h2Address*=**"jdbc:h2:mem:test\_mem;DB\_CLOSE\_DELAY=-1"**;  
*//domyślne wartości dla bazy danych w pamięci***private static** String *user*=**""**;  
**private static** String *password*=**""**;

By default, closing the last connection to a database closes the database. For an in-memory database, this means the content is lost. To keep the database open, add ;DB\_CLOSE\_DELAY=-1 to the database URL. To keep the content of an in-memory database as long as the virtual machine is alive, use jdbc:h2:mem:test;DB\_CLOSE\_DELAY=-1.

**public static** Connection getConnection() **throws** ClassNotFoundException, SQLException {  
 Connection connection=**null**;  
 Class.*forName*(*h2Driver*);  
 connection=DriverManager.*getConnection*(*h2Address*,*user*,*password*);  
 **return** connection;

}

**public static void** main(String[] args) **throws** SQLException, ClassNotFoundException {  
 Connection connection=*getConnection*();  
 **if** (connection!=**null**){  
 System.***out***.println(**"SUKCES"**);  
 }  
}

# JDBC

JDBC polega na przesyłaniu zapytań do bazy.

Będziemy chcieli wrzucić obiekty klasy Student do bazy danych.

@AllArgsConstructor  
@Getter  
@Setter  
@NoArgsConstructor

@ToString  
**public class** Student {  
 **private int id**;  
 **private** String **name**;  
}

Utworzenie tabeli USERS w bazie danych w pamięci (H2)

**private static void** createTableForStudent() **throws** SQLException, ClassNotFoundException {  
 Connection connection=AppH2.*getConnection*();  
 Statement statement=connection.createStatement();  
 String query=**"CREATE TABLE STUDENTS(id int primary key,name varchar(255))"**;  
 statement.execute(query);  
 connection.commit();  
}

Wstawienie do bazy danych obiektów klasy Student

**private static void** insertStudent(Student student) **throws** SQLException, ClassNotFoundException {  
 Connection connection=AppH2.*getConnection*();  
 Statement statement=connection.createStatement();  
 String query=**"INSERT INTO STUDENTS VALUES("**+  
 student.getId()+**",\'"**+student.getName()+**"\')"**;  
 statement.execute(query);  
 connection.commit();  
}

Pobranie studentów

**public static** List<Student> getStudent() **throws** SQLException, ClassNotFoundException {  
 List<Student> students=**new** ArrayList<>();  
 Connection connection=AppH2.*getConnection*();  
 Statement statement=connection.createStatement();  
 String query=**"SELECT** *\** **FROM STUDENTS"**;  
 ResultSet resultSet=statement.executeQuery(query);  
 **while** (resultSet.next()){  
 **int** id=resultSet.getInt(**"id"**);  
 String name=resultSet.getString(**"name"**);  
 students.add(**new** Student(id,name));  
 }  
 **return** students;  
}

Jak widzimy musi tu być **ResultSet resultSet=statement.executeQuery(query);**  
bo odbieramy wyniki

# JPA

JPA/Hibernate

Encja

Crud

Embedded

OneToOne

Generowanie unikalnego Id

Operacje kaskadowe