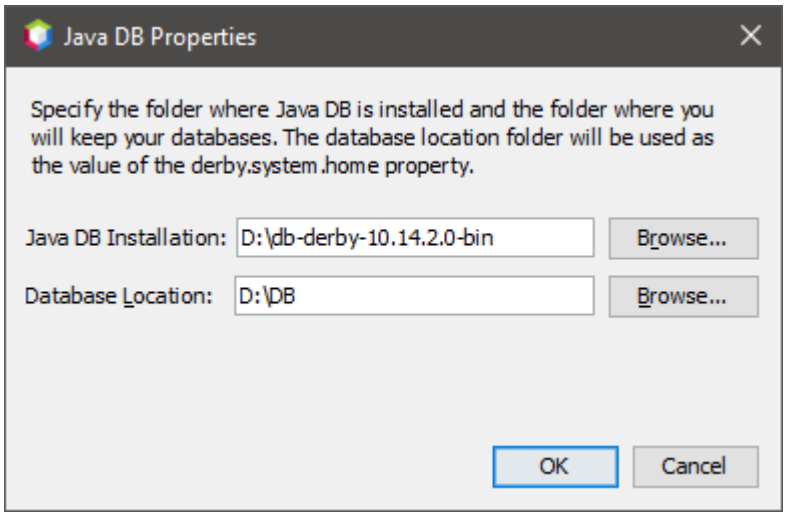
Komentarze dla prowadzących do ćwiczenia nr 5 „Aplikacje bazodanowe”

Przykłady znajdują się w pliku *exercise5_v.2021b.zip*

1. Przygotowania - uruchomienie serwera bazy danych Java Derby, ponieważ niestety nie jest dołączony do JDK, jak kiedyś bywało.

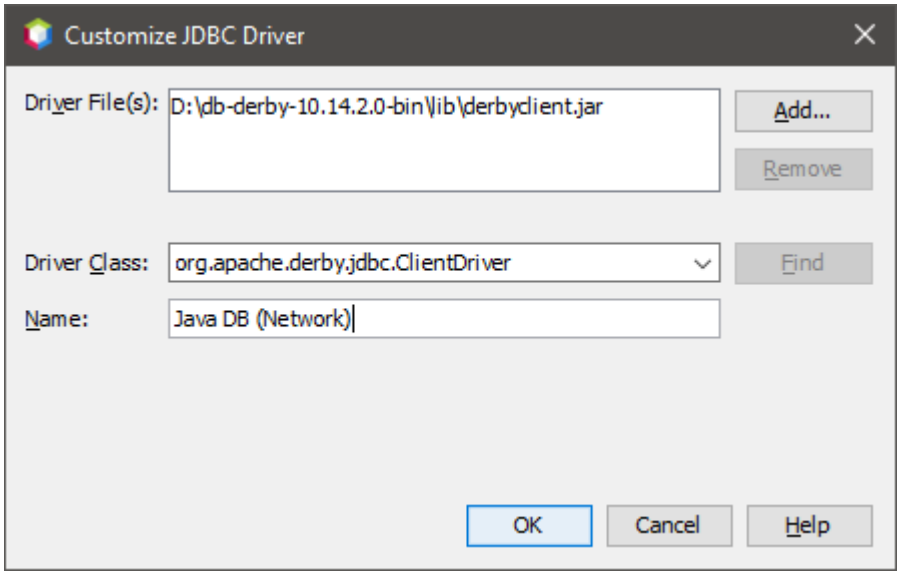
   a. Pobranie: ściągamy serwer Apache Derby z adresu: https://dlcdn.apache.org//db/derby/db-derby-10.14.2.0/db-derby-10.14.2.0-bin.zip Rozpakowujemy w dowolnym miejscu. Nie używamy nowszych wersji!

   b. Uruchomienie serwera: w oknie projektu wybieramy zakładkę *Services*, a w niej *Java DB*. Po naciśnięciu prawego przycisku myszki wybieramy *Properties*. Podajemy folder z serwerem bazy danych oraz miejsce gdzie będzie ulokowana sama baza.



   Klikamy prawym przyciskiem na *Java DB* i wybieramy *Start Server*.

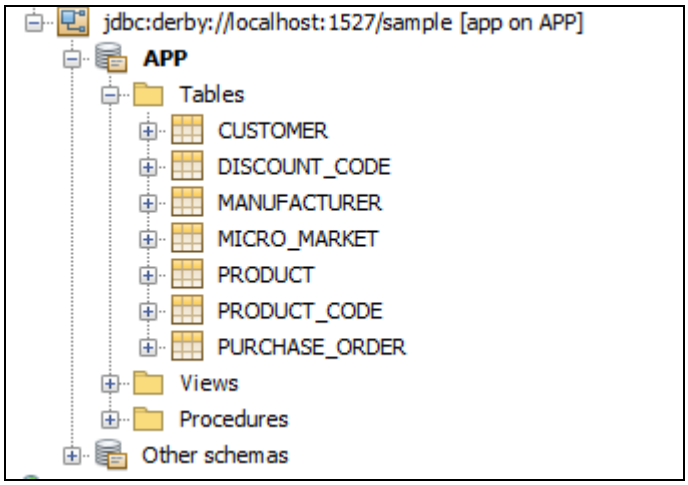
   c. Konfiguracja sterownika: Wciąż w tym samy oknie projektu, zakładka *Services* a w niej *Databases/Drivers* klikamy w Java DB (Network) prawym przyciskiem i wybieramy *Customize*



   Dodajemy odpowiedni plik sterownika z folderu gdzie rozpakowaliśmy serwer, a potem klikamy przycisk *Find*. Powinien znaleźć się sterownik taki, jak na zrzucie ekranu.

d. Otwarcie przykładowej bazy: przykładowa baza *sample* powinna być od razu dostępna. Jeśli nie, to na tej samej opcji (*Java DB*) ponownie naciskamy prawy przycisk myszy i wybieramy *Create Sample Databse* (Serwer BD musi być uruchomiony) i nadajemy jej nazwę *sample*. Następnie klikamy prawym przyciskiem i wybieramy *Connect…*

Następnie rozwijając okno połączenia można zobaczyć tabele w przykładowej bazie:



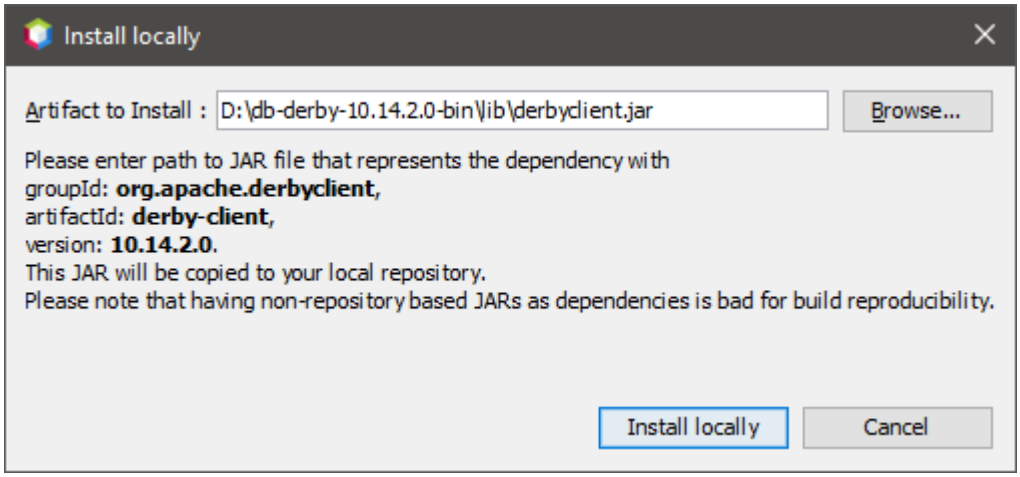
Na każdej z tabel można kliknąć „View data” i zobaczyć jakie zawiera rekordy.

Uwaga! Netbeans IDE może mieć czasem kłopot wyświetlaniem jakie aktualnie bazy są założone na serwerze. Pomaga restart Netbeans IDE.

---

2. Przykład JDBC – projekt z użyciem *Mavena*

a. Aktualizacja zależności.
Apache Derby może być zainstalowany w dowolnym miejscu, dlatego prawdopodobna będzie aktualizacja zależności *derby-client-10.14.2.0* Klikamy na niej prawym przyciskiem i wybieramy z menu opcję *Manually install artifact*. W otwartym okienku wskazujemy położenie pliku *jar*.



b. Projekt *JDBC* obejmuje pięć małych aplikacji. Uruchamiamy je (wybierając opcję *Run File* na każdym z nich) w kolejności:
CreateTablesApp,
InsertDataApp,
SelectDataApp,
DeleteDataApp,
SelectDataApp, (po raz drugi)

UpdateDataApp,
SelectDataApp, (po raz trzeci)
i obserwujemy wyniki na konsoli. Wyniki można również oglądać w zakładce:
Services/ jdbc:derby://localhost:1527/lab [app on APP]/ APP/ Tables/DANE
najpierw wykonując opcje *Connect* a potem naciskając prawy przycisk myszy na DANE wybierając *View Data*..

Każdy z przykładów składa się z następujących elementów:
- załadowanie sterownika,
- nawiązanie połączenia z serwerem BD
- wykonanie operacji na bazie
- zamknięcie połączenia.

Zwracamy uwagę na obsługę wyjątków.

---

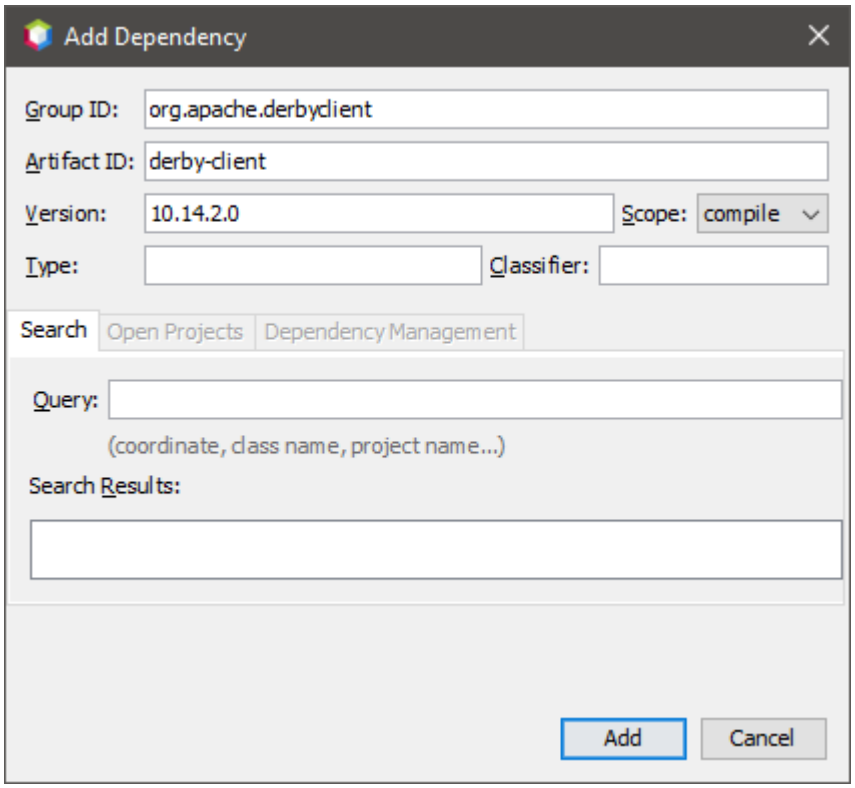
3. Przykład JPA

Prosty przykład pokazujący jak zapisywać i wyszukiwać dane z użyciem specyfikacji JPA w implementacji w bibliotece EclipseLink. W razie potrzeby aktualizujemy zależności ze sterownikiem bazodanowym jw.
Zwracamy uwagę, że to tylko demonstracja i nie tworzy się menadżera encji za każdym razem oraz że należy przechwytywać wyjątek typu *PersistenceException*.
Poza tym mieli już JPA na bazach danych, więc wszelkie operacje z użyciem mapowania obiektowo-relacyjnego powinni mieć w małym palcu.

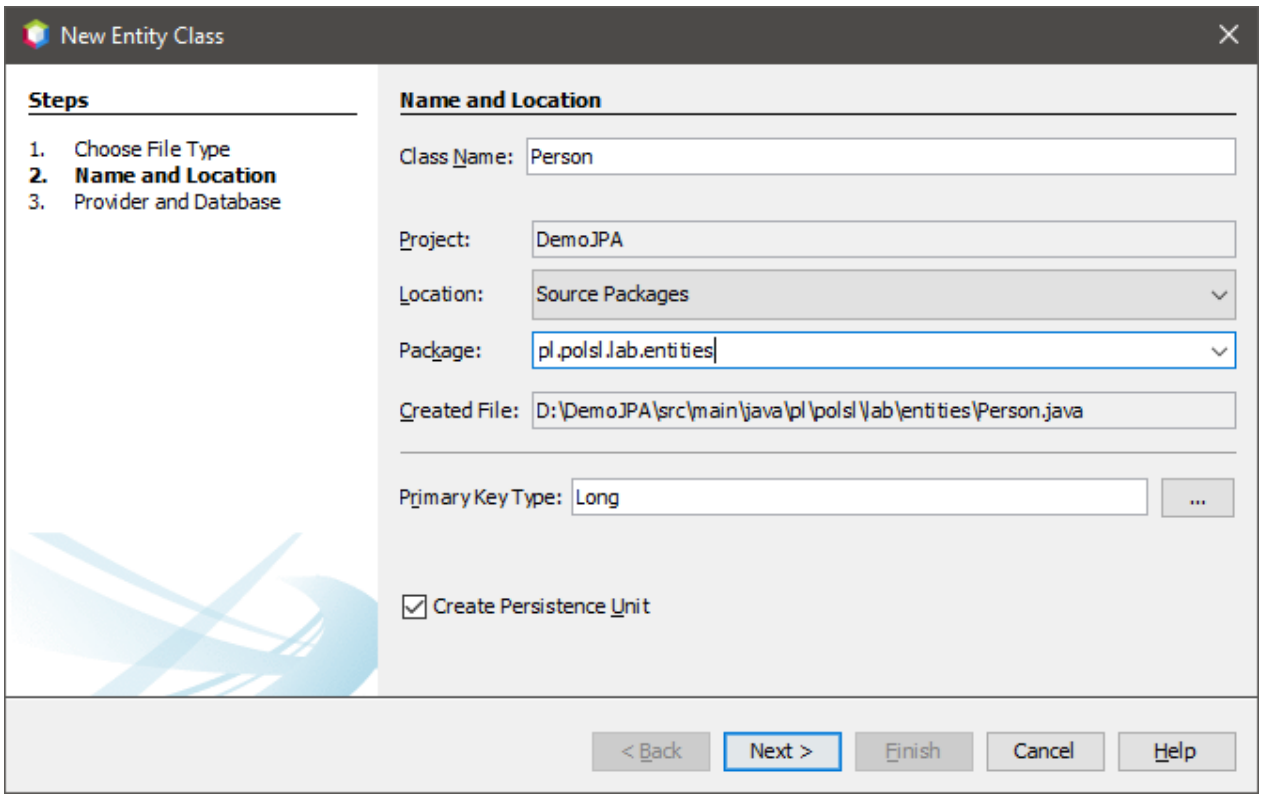
---

4. Składamy prosty projekt z użyciem JPA

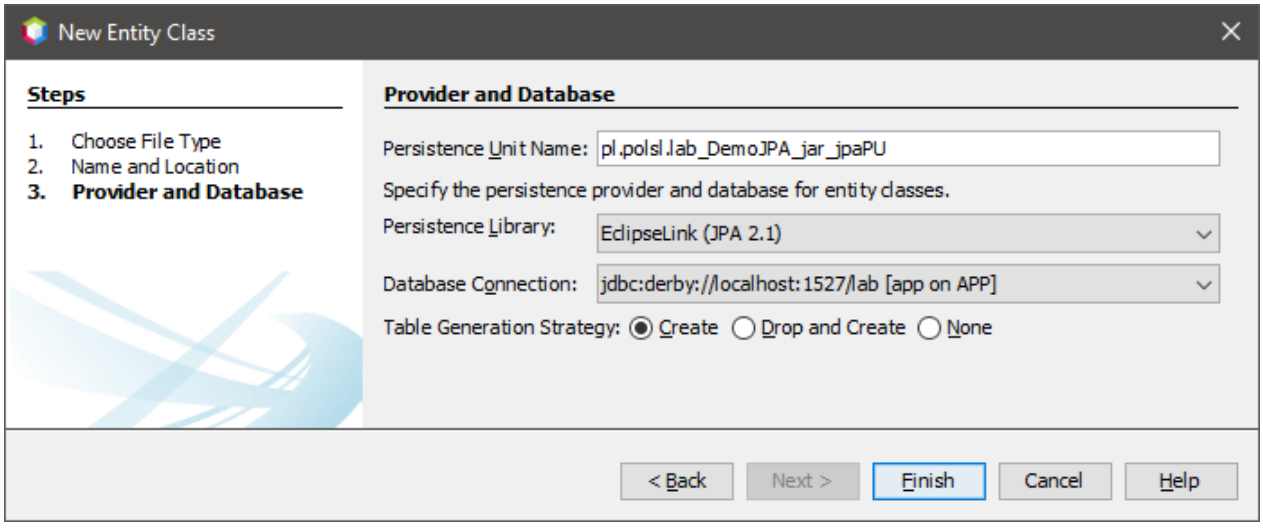
a) Tworzymy pusty projekt z *Mavenem*
b) Dodajemy zależność *Add Dependency*...



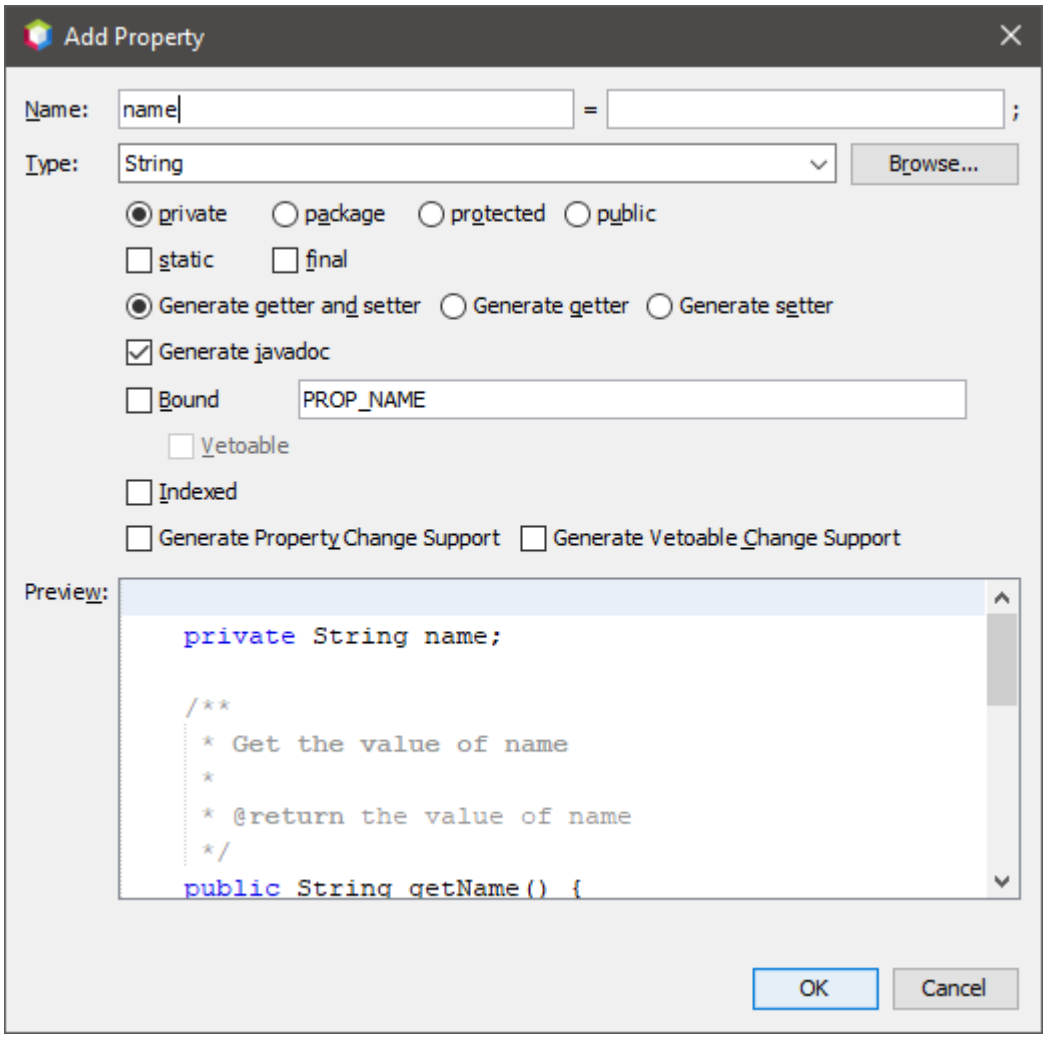
c) Dodajemy do projektu klasę encji (*Entity Class*…. w kategorii *Persistence*) nazywając ją np. *Person*. Uzupełnimy o jakieś dodatkowe pola wraz z metodami *set* i *get* (przypominam można takie elementy dodawać potem klikając w oknie edytora prawym przyciskiem i wybierając *Insert Code…*).



d) Dodajemy do projektu jednostkę trwałości (*Persistence Unit*) wybierając zdefiniowane wcześniej połączenie.



e) Dodajemy do klasy *Person* pole korzystając z opcji *Insert code../Add Property*…. dostępnej po kliknięciu prawym przyciskiem w oknie edycji pliku Person.java

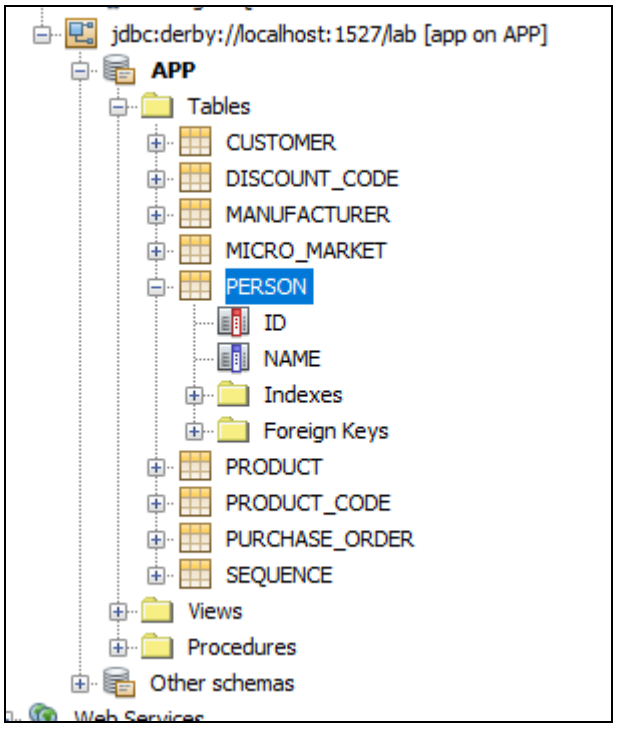


f) Dodajemy główną klasę projektu, a w niej metodę *main* z prostym tworzeniem obiektu i sprawdzeniem, czy zadziała (Serwer rzecz jasna musi być uruchomiony).

```
public static void main(String[] args) {
     Person person = new Person();
     person.setName("Mary");
     Main main = new Main();
     main.persist(person);
 }
```

W oknie edytora głównej klasy korzystając z opcji *Insert code.../Use Entity Manager…* wstawiamy definicję metody *persist* zawierającą przykład użycia zarządcy trwałości. Zwracamy uwagę że generator kodu nie jest dokonały, ponieważ nie należy łapać zbyt ogólnego wyjątku *Exception*, lecz *PersistenceException*

g) Następnie podglądamy bazę danych na serwerze i sprawdzamy czy tabela się utworzyła, a nasz rekord został dopisany (*View Data*).


jdbc:derby://localhost:1527/lab [app on APP]
APP
Tables
CUSTOMER
DISCOUNT_CODE
MANUFACTURER
MICRO_MARKET
PERSON
ID
NAME
Indexes
Foreign Keys
PRODUCT
PRODUCT_CODE
PURCHASE_ORDER
SEQUENCE
Views
Procedures
Other schemas