Lab 4

Persistenta in aplicatii java

Pregatire: dezarhivati arhiva lab4_files.zip local.

Specificatia JPA (Java Persistence API) este un mecanism de mapare obiecte – date relationale in vederea folosirii datelor relationale in aplicatii java. Maparea dintre obiectele java si datele dintr-o baza de date relationala este realizata cu ajutorul anotarilor (annotations) si/sau fisiere xml (xml deployment descriptors).

Desi JPA face parte din stiva de specificatii EJB 3, ea poate fi folosita si in aplicatii Java SE, in afara containerului Java EE.

4.1 JPA in aplicatii desktop (folosind Netbeans)

Vom defini mai intai baza de date folosind Derby Db.

Derby DB este o baza de date open-source, implementata integral in java, de catre fundatia Apache (https://db.apache.org/derby/)

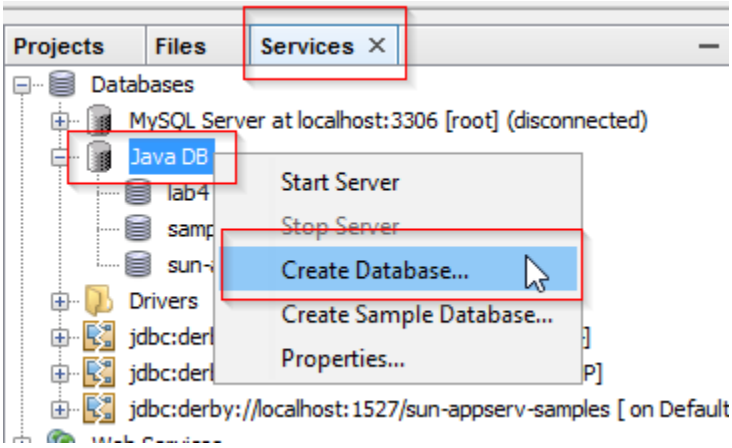
Are un foot-print mic (3.5 MB) in care este inclus atat motorul de baza de date cat si driverul embedded JDBC.

Datorita acestui driver Derby poate fi utilizata prin 'embeddare' in orice aplicatie java .

Derby este inclusa in distributia NetBeans astfel incat nu este nevoie sa instalam suplimentar nimic.

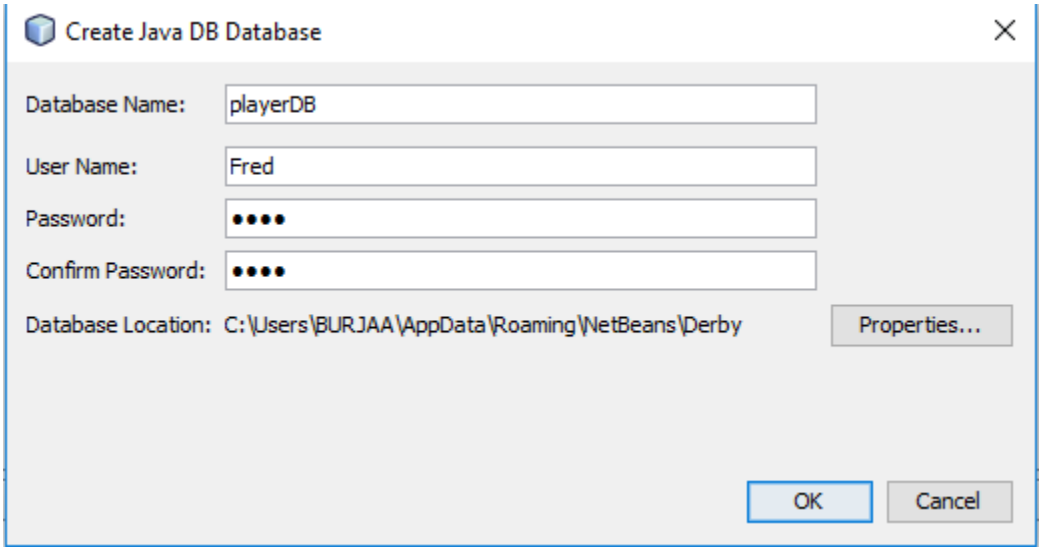
4.1.1 Crearea unei baze de date

Pentru a crea o noua baza de date selectati tab-ul 'Services' (stanga sus):

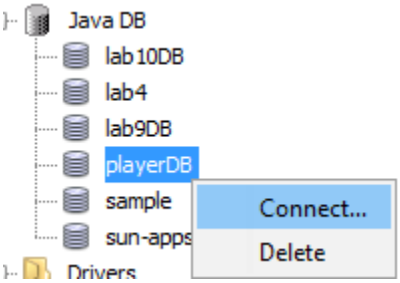


Creati baza de date cu numele **playerDB,** user**: Fred,** pass**: Fred :**

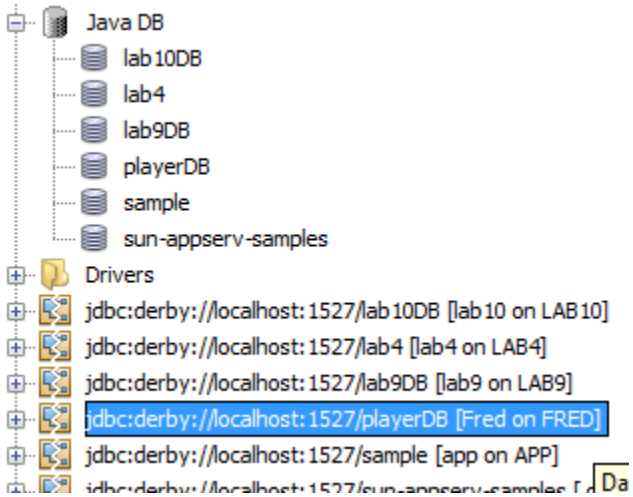
Lab 4



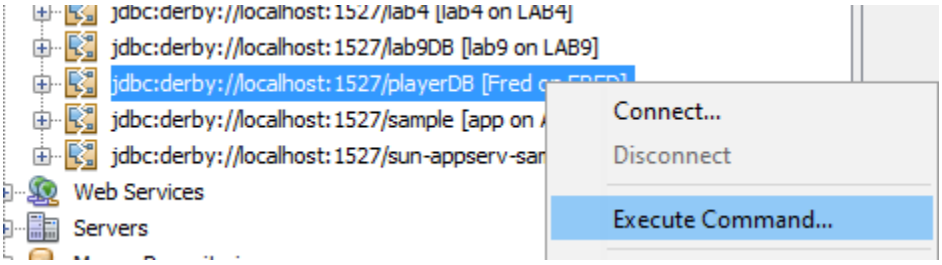
Apoi porniti aceasta baza de date:



Dupa acest pas avem disponibila conexiunea la baza de date:



Iar acum putem executa comenzi SQL:

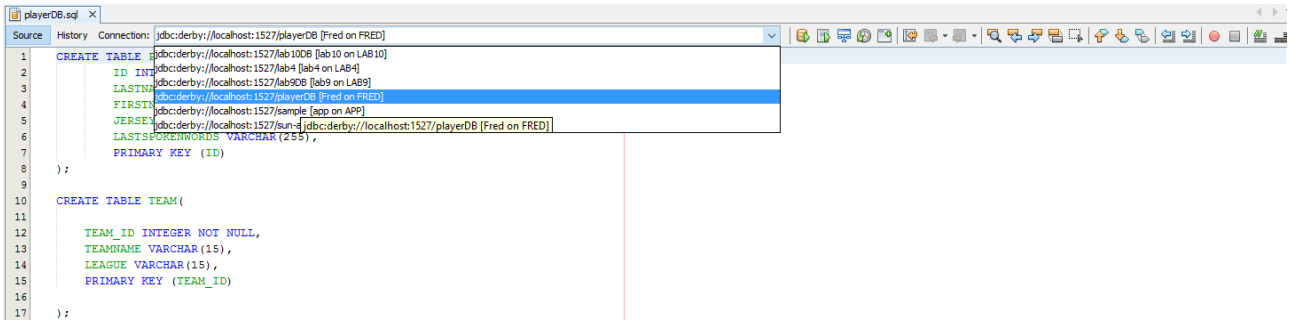


4.1.2 Crearea tabelelor si a datelor

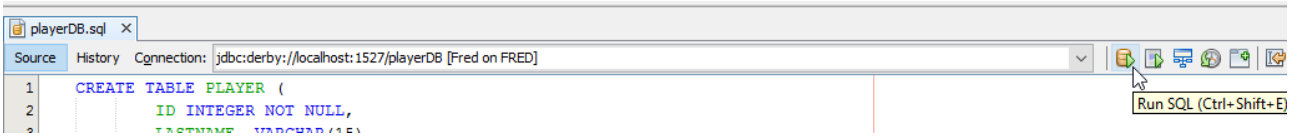
Netbeans -> File -> Open File -> alegeti `playersDB.sql` din directorul unde ati dezarhivat arhiva de la inceput.

-> Open
Fisierul se deschide automat in editorul SQL.

Selectati conexiunea `jdbc:derby://localhost:1527/playerDB`:



Apasati iconita ‘Run SQL’ pentru executia scripurilor:

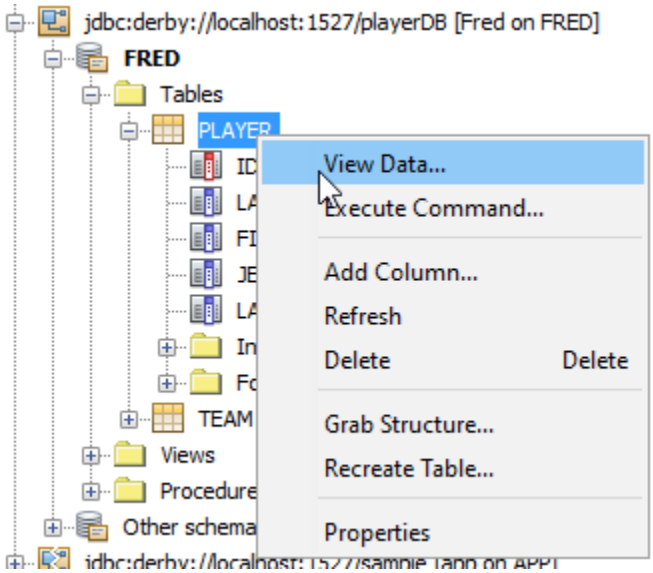


4.1.3 Examinarea continutului tabelelor

In tab-ul 'Services' expandati conexiunea `jdbc:derby://localhost:1527/playerDB`

Click dreapta -> Refresh

Expandati schema 'Fred' -> Tables -> PALYER -> right click: View Data



4.2.1 Generarea entitatilor din baza de date

Cream un nou proiect java:

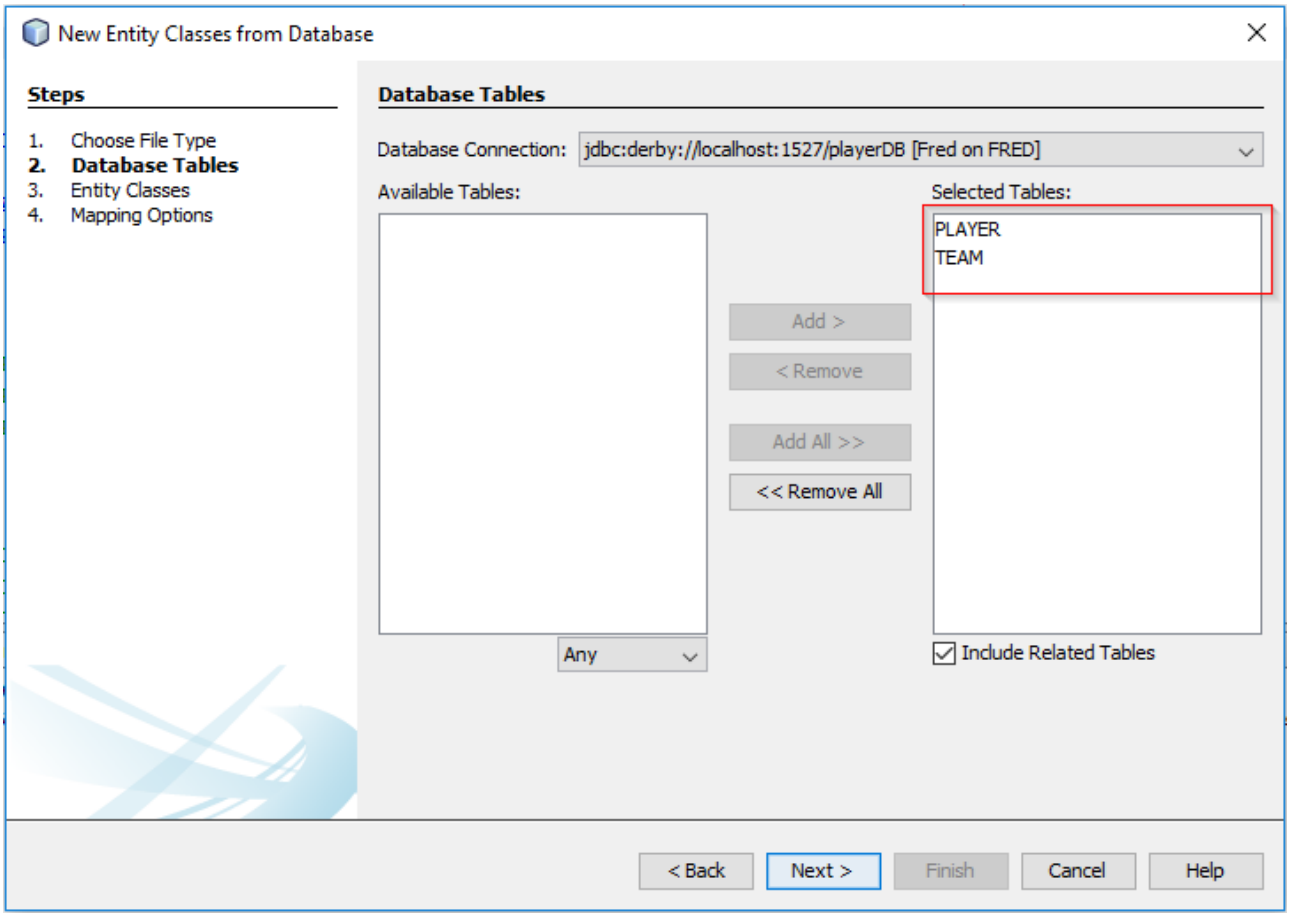
File -> New Project -> Java -> Java Application -> Next

Nume: **PersistenceDemo**
Create Main Class: debifati casuta
Finish

PersistenceDemo Project -> Right-click: ***New -> Other -> Persistence -> Entity Classes From Database***.

Database Connection: `jdbc:derby://localhost:1527/playerDB`

Selectati ambele tabele: PLAYER si TEAM si adaugati-le in tabelul din dreapta:



Apoi adaugati un packet (ex. demo) -> Finish

Verificati clasele generate: Player si Team

Setul anterior de entitati create se cheama unitate de persistenta (persistence unit). Acestea se configureaza in persistence.xml localizat in subdirectorul META-INF. Acest xml contine numele furnizorului de persistenta (persistence provider), entitatile si proprietatile conexiunii la baza de date: URL, jar-ul pt db (driver-ul), userul, parola.

Editati fisierul persistence.xml redenumind unitatea de persistenta: **PersistenceDemoPU**

4.2.2 Operatiile CRUD (Create, Read, Update, Delete)

Create

Selectati proiectul **PersistenceDemo** -> click dreapta: **New -> Java Class**

Folositi numele **CreatePlayers (**packet demo)

Lab 4

Editati clasa CreatePlayers:

Adaugati importurile:

```
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
```

Adaugati metoda:

```
public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("PersistenceDemoPU");
    EntityManager em = emf.createEntityManager();
}
```

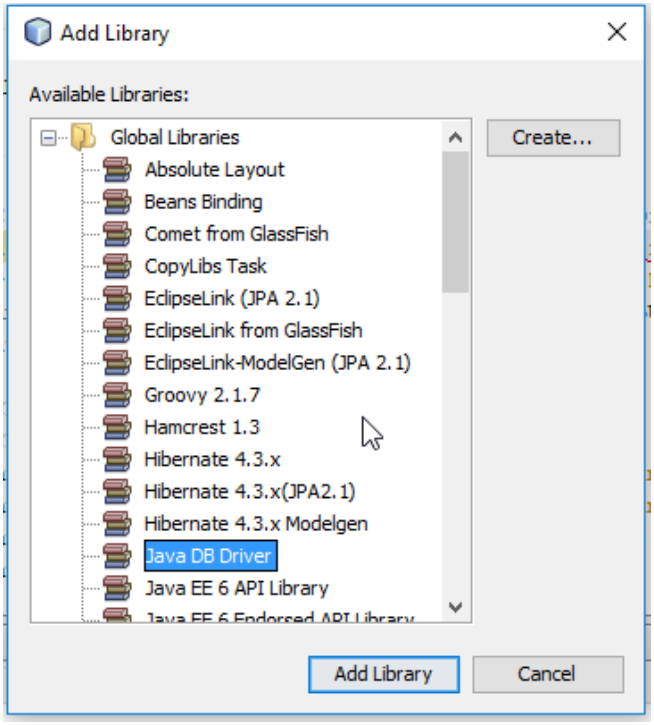
Observati instantierea unui EntityManager . Cu ajutorul acestei instante sunt efectuate operatiile CRUD.

Adaugati urmatorul cod in main:

```
    em.getTransaction().begin();

    Player p1 = new Player();
    p1.setId(5);
    p1.setFirstname("Ian");
    p1.setJerseynumber(30);
    p1.setLastname("Thorpe");
    p1.setLastspokenwords("I am in the best form");
    em.persist(p1);

    Player p2 = new Player();
    p2.setId(6);
    p2.setFirstname("Diego");
    p2.setJerseynumber(40);
    p2.setLastname("Maradona");
    p2.setLastspokenwords("I will be back");
    em.persist(p2);

    em.getTransaction().commit();

    em.close();
    emf.close();
```

Adaugati driverul de baza de date la proiect:

Selectati proiectul **PersistenceDemo** -> **Libraries -> Add library:**



Alternativa:

Cautati derby.jar pe disc. Rezultatul ar putea fi: "C:\Program Files\Java\jdk1.8.0_144\db\lib\derbyclient.jar"

Selectati proiectul **PersistenceDemo** -> click dreapta: **Properties -> Libraries -> Add jar/folder**

Adaugati calea de mai sus.

Selectati clasa **CreatePlayers.java** -> click dreapta -> Run File

Verificati rezultatul in consola. Verificati si noile date din tabela PLAYER.

Nu uitati ca dupa terminarea operatiunilor sa inchideti conexiunea !!! (ex conn.close)

Mai multe la:
https://www.tutorialspoint.com/jpa/index.htm

https://www.infoworld.com/article/3373652/java-persistence-with-jpa-and-hibernate-part-1-entities-and-relationships.html

https://www.infoworld.com/article/3387643/java-persistence-with-jpa-and-hibernate-part-2-many-to-many-relationships.html

https://download.oracle.com/otn-pub/jcp/persistence-2_2-mrel-spec/JavaPersistence.pdf

http://www.javawebtutor.com/articles/jpa/jpa-overview.php

https://docs.oracle.com/javaee/6/tutorial/doc/bnbpy.html

https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/SettingUpJPA/SettingUpJPA.htm

http://db.apache.org/derby/integrate/plugin_help/derby_app.html

4.3 Probleme

4.3.1 Similar clasei CreatePlayers realizati clasa ReadPlayers

4.3.2 Similar clasei CreatePlayers realizati clasa UpdatePlayers

4.3.3 Similar clasei CreatePlayers realizati clasa Delete Players

4.3.4 Realizati clasa/clase care sa realizeze operatii CRUD pt tabela TEAM.