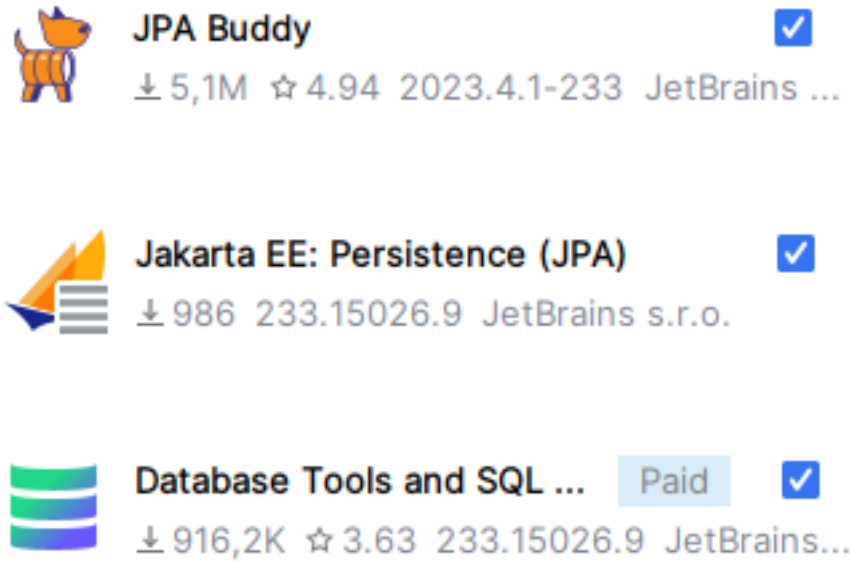
JPA aplikace

Nastavení IntelliJ IDEA

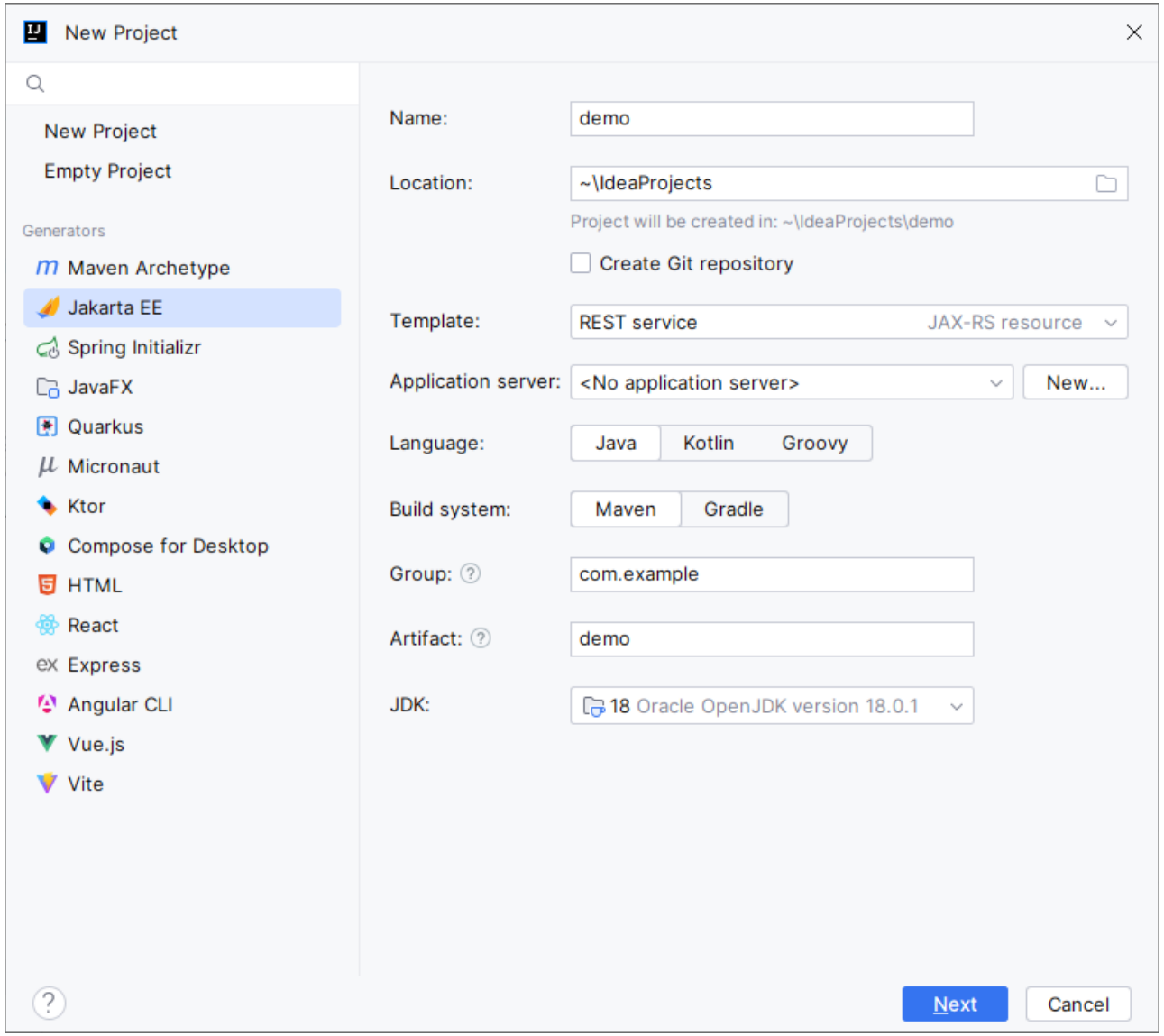
Settings… -> Plugins



Nastavení IntelliJ IDEA

New Project

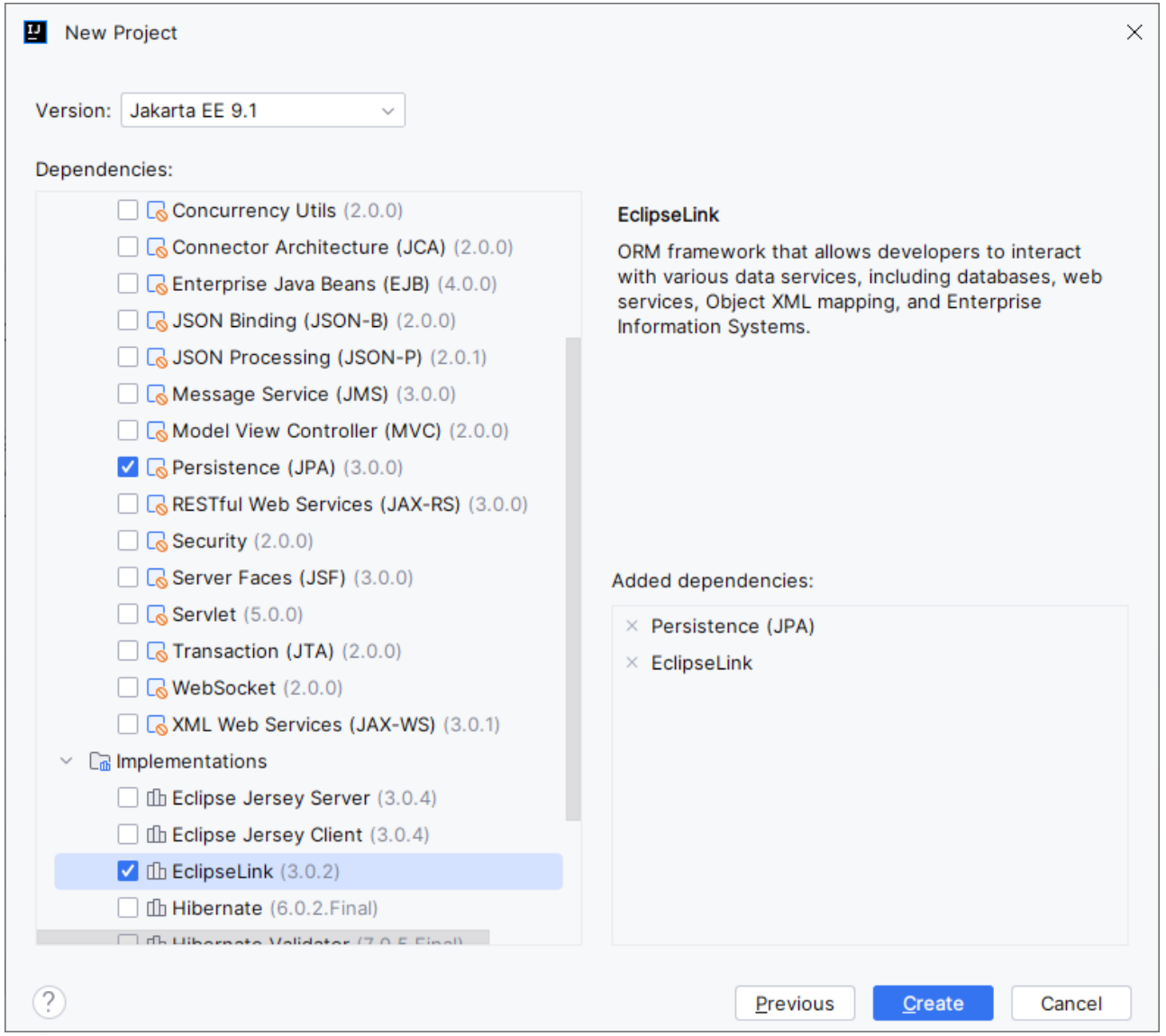
Důležité je to NEXT svítíci



Nastavení IntelliJ IDEA

New Project

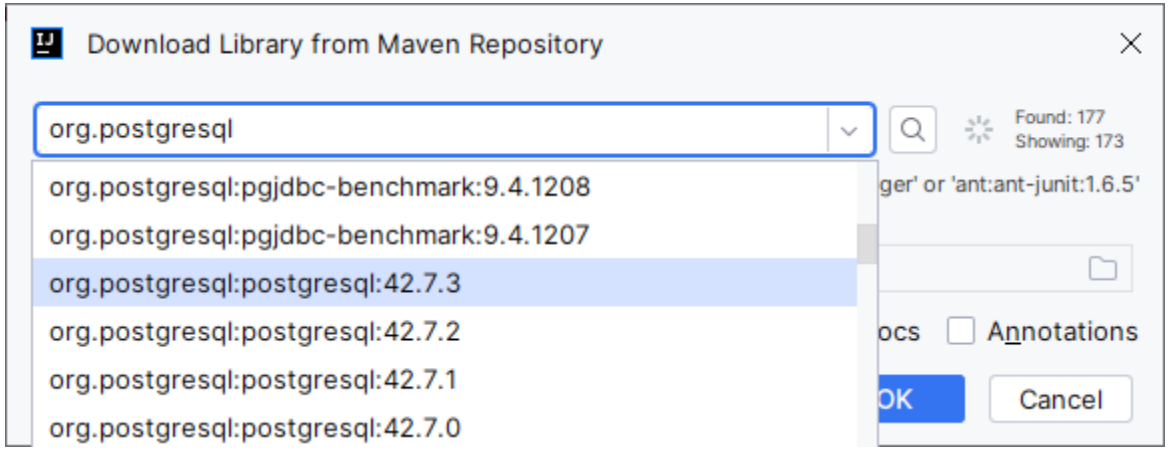
- Persistence (JPA)
- EclipseLink



Nastavení IntelliJ IDEA

Project Structure…

- Libraries -> From Maven…



- **pom.xml**

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.7.3</version>
    <scope>test</scope>
</dependency>
```

Nastavení IntelliJ IDEA

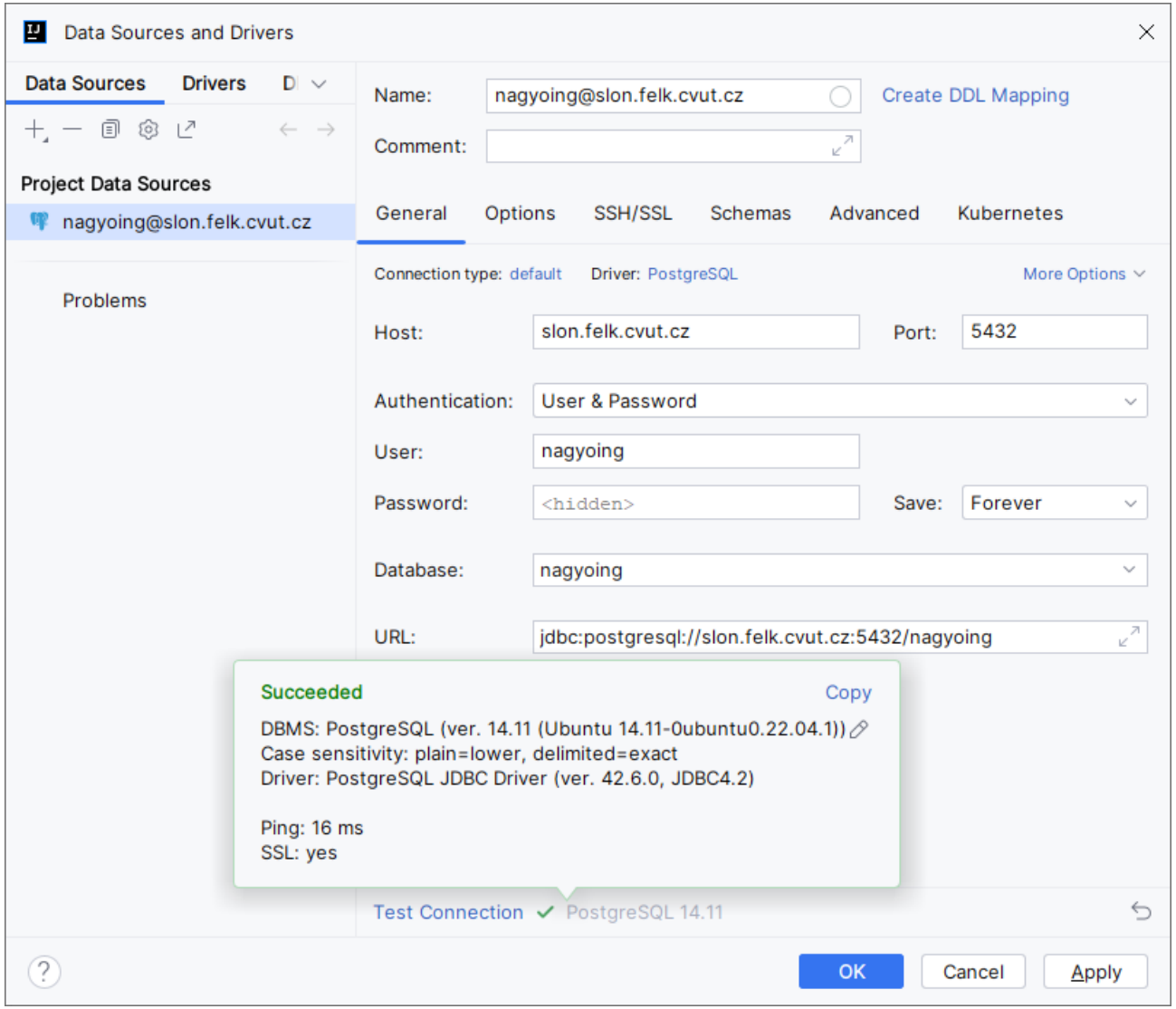
- **persistence.xml**

```
<persistence-unit name="ApplicationPU" transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>tables.Kniha</class>
    <properties>
        <property
                name="jakarta.persistence.jdbc.url"
                value="jdbc:postgresql://slon.felk.cvut.cz:5432/nagyoing"/>
        <property
                name="jakarta.persistence.jdbc.user"
                value="nagyoing"/>
        <property
                name="jakarta.persistence.jdbc.driver"
                value="org.postgresql.Driver"/>
        <property
                name="jakarta.persistence.jdbc.password"
                value="heslo"/>
        <property
                name="jakarta.persistence.schema-generation.database.action"
                value="create"/>
    </properties>
</persistence-unit>
```

Nastavení IntelliJ IDEA

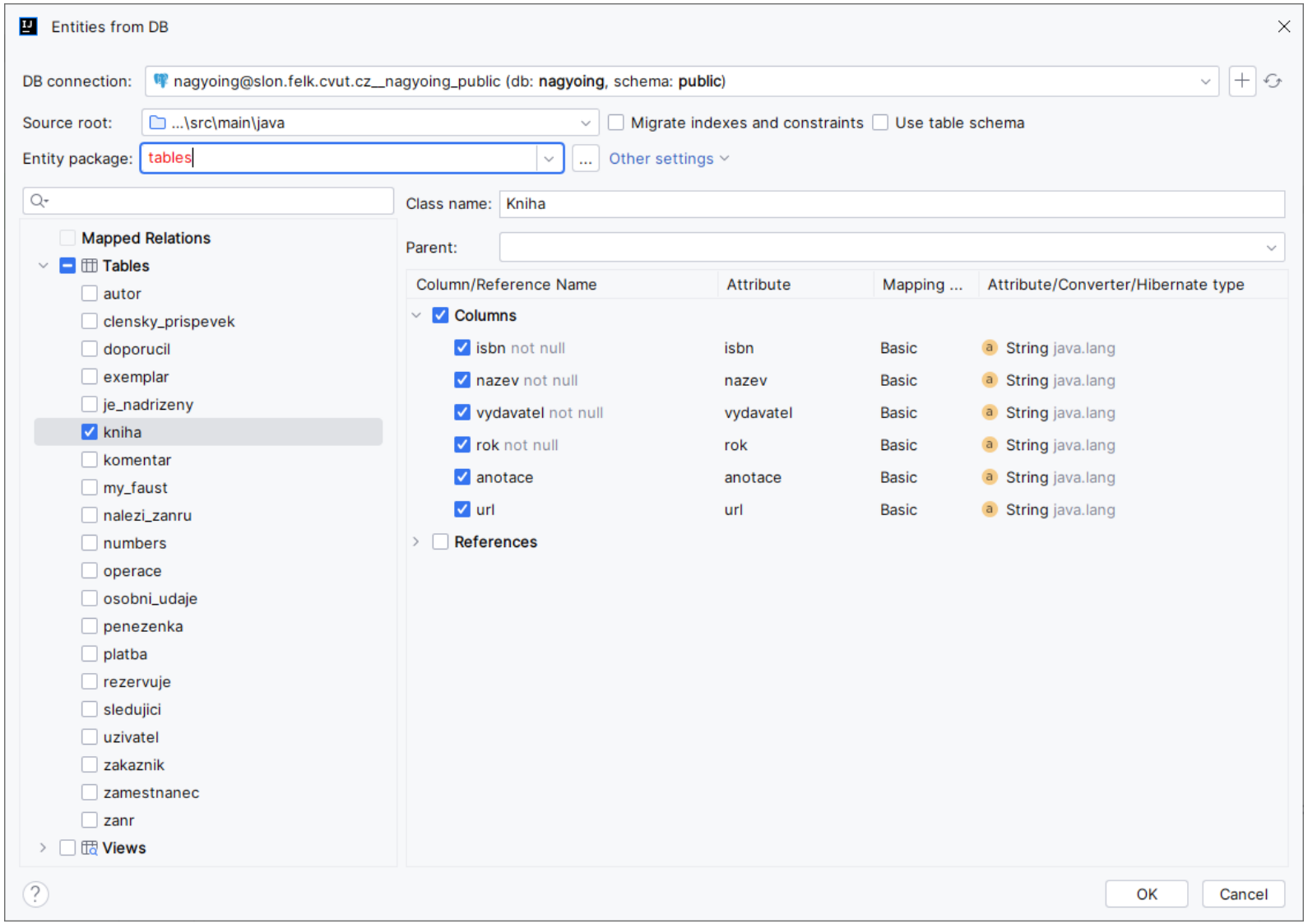
View -> Tool Windows -> Database

- New -> Data Source
  -> PostgreSQL



Vytvoření třídy podle tabulky

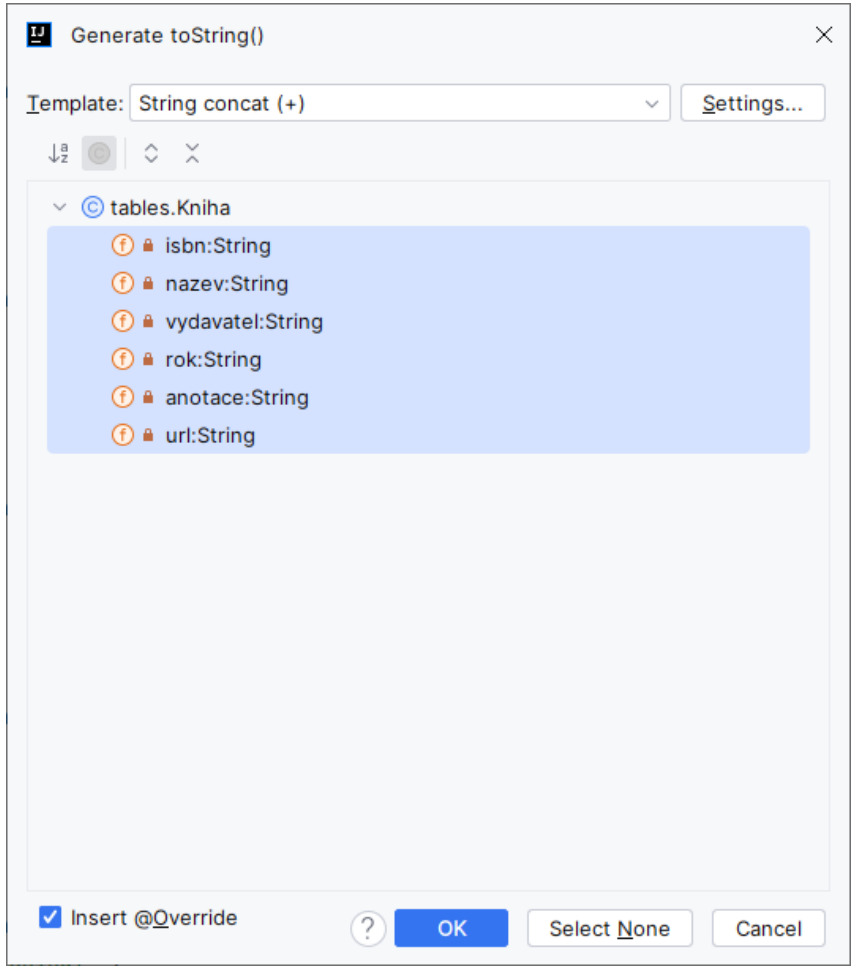
JPA Entities from DB



Vytvoření třídy podle tabulky

Přidáme výpis

- Generate… -> toString()



Vytvoření hlavního programu – třída Main.java

```
import jakarta.persistence.*;

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

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

Vložení věty do databáze

```
EntityTransaction et = em.getTransaction();

Kniha k = new Kniha();
k.setIsbn("978-0-349-43702-6");
k.setNazev("Iron Flame");
k.setVydavatel("Little, Brown and Company");
k.setRok("2023");

et.begin();
em.persist(k);
et.commit();
```

Výběr vět z databáze

```
TypedQuery<Kniha> q =
          em.createQuery("SELECT k FROM Kniha AS k", Kniha.class);
List<Kniha> kk = q.getResultList();
for (Kniha k : kk) {
    System.out.println(k);
}

q = em.createQuery(
    "SELECT k FROM Kniha AS k WHERE (k.vydavatel = :idVydal)",
    Kniha.class);
q.setParameter("idVydal", "Kontrast");
kk = q.getResultList();
for (Kniha k : kk) {
    System.out.println(k);
}
```

Změna věty v databázi

```
Kniha b = em.find(Kniha.class, "978-80-277-2018-7");
b.setRok("2020");
em.merge(b);
```

find

- Hledá podle ID

merge

- Změní knihu, jak ji podle ID najde

Mazání věty v databázi

```
EntityTransaction et = em.getTransaction();

Kniha b = em.find(Kniha.class, "978-80-277-2018-7");
et.begin();
em.remove(b);
et.commit();
```

DAO vrstva

```
public class KnihaDAO {
    private EntityManager em;
    public KnihaDAO(EntityManager em) {
        this.em = em;
    }
    public Kniha insertKniha(String isbn, String nazev, String vydav, String rok) {
        ...
        return k;
    }
    public List<Kniha> getAll() {
        return em.createQuery("SELECT k FROM Kniha AS k", Kniha.class).getResultList();
    }
    public Kniha getKniha(String isbn) {
        return em.find(Kniha.class, isbn);
    }
    public Kniha updateKniha(Kniha k) {
        return em.merge(k);
    }
    public void deleteKniha(Kniha k){
        ...
    }
}
```

DAO vrstva

```
public Kniha insertKniha(String isbn, String nazev, String vydav, String rok) {
    EntityTransaction et = em.getTransaction();
    Kniha k = new Kniha();
    k.setIsbn(isbn);
    k.setNazev(nazev);
    k.setVydavatel(vydav);
    k.setRok(rok);
    et.begin();
    em.persist(k);
    et.commit();
    return k;
}

public void deleteKniha(Kniha k){
    EntityTransaction et = em.getTransaction();
    et.begin();
    em.remove(k);
    et.commit();
}
```

DAO vrstva

Nebo to, co ještě potřebujeme

- Knihy, které jsou vydány daným vydavatelem
- Nebo jiné potřebné dotazy, transakce

```
public List<Kniha> getAllVydavatel(String idVydal) {
    TypedQuery<Kniha> q = em.createQuery(
            "SELECT k FROM Kniha AS k WHERE (k.vydavatel = :idVydal)",
            Kniha.class);
    q.setParameter("idVydal", idVydal);
    return q.getResultList();
}
```

DAO vrstva

Upravíme hlavní program s využitím DAO vrstvy

- Do databáze zapíšeme novou knihu
- Vybranou knihu upravíme
- Vypíšeme seznam knih v databázi
- Změny na vybrané knize zrušíme
- Vypíšeme seznam knih v databázi – knihy vydané vydavatelem Kontrast
- Vymažeme nově zapsanou knihu

Žánry a knihy, které žánru náleží

```
List<Zanr> zz = em.createQuery(
    "SELECT z FROM Zanr AS z",
    Zanr.class).getResultList();
for (Zanr z : zz) {
    TypedQuery<NaleziZanru> q = em.createQuery(
            "SELECT n FROM NaleziZanru AS n WHERE (n.nazev = :idZanr)",
            NaleziZanru.class);
    q.setParameter("idZanr", z);
    System.out.println(z);
    List<NaleziZanru> nz = q.getResultList();
    for (NaleziZanru nnz : nz) {
        System.out.println("\t" + nnz.getIsbn());
    }
    System.out.println();
}
```