Duomenų bazės su *Spring*

Code Academy

Programuok savo ateitį!

Code Academy
Programuok savo ateitį!

Kas yra *ORM*

ORM arba *Object relational mapping* yra būdas rašyti duomenų bazės užklausas naudojant Object-oriented paradigmą, naudojant pageidaujamą programavimo kalbą.

Tai technika, surišanti programavimo kalbos objektus su duomenų bazės modeliais, tam skirtų bibliotekų pagalba

Tai leidžia dirbti su duomenų bazės duomenimis programavimo kalbos lygmenyje, nesirūpinant duomenų bazės valdymo detalėmis.

Code Academy
Programuok savo ateitį

JPA

Java Persistence API (JPA) - sąsajų specifikacija, apibūdinanti reliacinių duomenų valdymą Java platformoje.

Bibliotekos sąsajos apibūdina standartą, kaip reikėtų įgyvendinti duomenų valdymą ORM būdu Java platformoje.

Biblioteka sudaryta iš 3 dalių:

- Pačios bibliotekos, esančios *javax.persistence* pakete
- *Java Persistence Query Language (JPQL)*
- meta informacijos apie objektus/reliacinius duomenis

Code Academy
Programuok savo ateitį

Entity klasės

Klasės, kurių paskirtis yra reprezentuoti duomenų bazės lenteles yra vadinamos *Entity*. Kiekvienas *Entity* klasės objekto pavyzdys reprezentuoja vieną eilę duomenų bazės lentelėje.

Code Academy
Programuok savo ateitį

Reikalavimai *Entity* klasėms

- Klasė turi turėti *javax.persistence.Entity* anotaciją
- Klasė turi turėti *public* arba *protected* konstruktorių be argumentų
- Klasė ir jokie jos metodai negali būti *final*
- *Entity* klasės gali paveldėti / būti paveldėtos, tiek *Entity* tiek paprastų klasių
- Klasės laukai turi būti private ir pasiekiami tik per *setter* / *getter* metodus

Code Academy
Programuok savo ateitį

Spring Data

Spring Data yra dar vienas abstrakcijos lygis virš *JPA* bibliotekos.

Spring data:

- Paslepia tam tikras duomenų bazės detales, leidžiant, iki tam tikro lygio, dirbti su duomenimis neatsižvelgiant, kokia duomenų bazė yra naudojama
- Sumažina *boilerplate* kodo kiekį, pateikiant standartinius metodus, pasiekti duomenims
- Suteikia galimybę lengvai kurti reikalingas užklausas

Code Academy
Programuok savo ateitį!

Prisijungimas prie duomenų bazės

Norint prisijungti prie *H2* duomenų bazės užtenka nurodyti kelis konfiguracijos parametrus, konfigūracijos faile:

```
spring.datasource.url=jdbc:h2:mem:db
spring.datasource.username=sa
spring.datasource.password=sa
```

Norint prisijungti prie kitokių duomenų bazių (pvz. *MySQL*) gali reikėti pridėti papildomus konfiguracijos parametrus, tačiau nepaisant to, konfigūravimo procesas išlieka labai greitas ir nesudėtingas

Code Academy
Programuok savo ateitį!

Spring Data

Naudojant *Hibernate* biblioteką, vartotojui nereikia rūpintis duomenų gavimo subtilybėmis - jam tereikia sukurti *Data Access Object (DAO)* naudojant *Hiberntate* teikiamus įrankius, kuris parūpina reikalingus duomenis iš duomenų bazės.

Spring Data dar labiau supaprastina darbą su duomenų saugyklomis, pašalinant būtinybę įgyvendinti *DAO* klases, atsakingas už duomenų pasiekimą.

Šių objektų įgyvendinimus pateikia *Spring Data* biblioteka

Code Academy
Programuok savo ateitį

Spring Data

Norint pradėti dirbti su *JPA* biblioteka naudojant *Spring Data,* užtenka sukurti sąsają, paveldinčią *JpaRepository* sąsają.

Spring data biblioteka automatiškai suranda klases, paveldinčias Spring repozitorijos sąsajas (kaip pvz. *JpaRepository*), ir automatiškai aprūpina jų įgyvendinimus.

Paveldint sąsają, naudotojas gauna aktualiausius *CRUD* metodus, reikalingus dirbti su duomenų saugyklų duomenimis

Code Academy
Programuok savo ateitį!

Spring Data repozitorijos

Norint sukurti repozitoriją klasėje T, kur T yra Entity, užtenka sukurti šią sąsają:

```
public interface TRepository extends JpaRepository<T, ID>  {
}
```

Spring Data automatiškai įgyvendins šią sąsają ir jį paveldi rinkinį metodų, skirtų dirbti su duomenų baze iš *JpaRepository* sąsajos

Paveldimų metodų pavyzdžiai: *findAll(), findById(ID id), deleteAll, save(T t),* ...

Code Academy
Programuok savo ateitį

Spring Data užklausos

Jei repozitorijų pateikiamos užklausos netenkina sistemos reikalavimų, Spring Data suteikia tris būdus, kaip rašyti reikalingas užklausas

Code Academy
Programuok savo ateitį!

Automatic Custom Queries

```
public interface FooDAO extends JpaRepository<Foo, Long> {
   Foo findByName(String name);
}
```

Spring Data analizuoja abstrakčius repozitorijų metodus ir pagal jų pavadinimus įgyvendina reikalingas užklausas.
Šiuo atveju metodo implementacija ieškos *Foo* objektų, kurių *name* lauko reikšmė bus lygi gautam *name* parametrui

Instrukcijas, kaip rašyti tokias užklausas galite rasti čia:
https://docs.spring.io/spring-data/data-jpa/docs/current/reference/html/#jpa.query-methods.query-creation

Code Academy
Programuok savo ateitį!

Manual Custom Queries

```
@Query("SELECT f FROM Foo f WHERE LOWER(f.name) = LOWER(:name)")
Foo retrieveByName(@Param("name") String name);
```

Užklausos, turinčios *Query* anotaciją, nėra įgyvendinamos pagal metodo pavadinimą. Metodų grąžinami duomenys yra aprašomi JPQL kalbos užklausomis

Instrukcijas, kaip rašyti tokias užklausas galite rasti čia:
https://docs.spring.io/spring-data/data-jpa/docs/current/reference/html/#jpa.named-parameters

Code Academy
Programuok savo ateitį

Native Queries

```
@Query(value = "SELECT * FROM Users u WHERE u.name = :name",
nativeQuery = true)
User findUserByName(@Param("name") String name);
```

Jei kartais prireiktų, *Query* anotacijoje nurodžius parametrą *nativeQuery* = *true*, užklausas galima rašyti standartine *SQL* kalba, nenaudojant *JPQL* abstrakcijos

Code Academy
Programuok savo ateitį!

Spring Data abstrakcijos

Skirtingos repozitorijų sąsajos paslepia nuo naudotojų tam tikrą informacijos kiekį apie naudojamas duomenų saugyklas.

Skirtingos repozitorijos teikia skirtingo lygio abstrakcijas.

Pavyzdžiui *CrudRepository* sąsaja gali būti naudojama su bet kokia duomenų baze, nežinant jos subtilybių, o *JpaRepository* sąsajos teikiami metodai gali būti naudojami su bet kokia reliacine duombaze.

Code Academy
Programuok savo ateitį!

Spring Data repozitorijų abstrakcijos hierarchija