Hibernate - Övning: Mappning

Mål

I denna övning ska du:

- Lägga till nya fält i en Hibernate-entitet och uppdatera databasen.
- Använda olika namn för kolumner i Java-klassen och databasen.
- Förstå skillnaden mellan fält- och egenskapsaccess i Hibernate.
- Använda @Transient för att exkludera fält från databasen.

Steg 1: Förberedelser

- Se till att du har ditt Hibernate-projekt konfigurerat.
- Starta Derby-databasen:
    - Windows (cmd)

      ```
      NetworkServerControl.bat -p 50000 start
      ```

    - Mac/Linux:

      ```
      ./NetworkServerControl -p 50000 start
      ```

- Öppna en ny terminal och starta ij-verktyget:

  Windows: ij.bat

  Mac/Linux: ./ij
- Anslut till databasen:

  ```
  connect 'jdbc:derby://localhost:50000/Hibernate; create=true' ;
  ```

Tips:

- Kontrollera att student tabellen finns genom att skriva:

```
describe student;
```

Steg 2: Lägg till ett nytt fält i Student klassen

- Lägg till ett nytt fält numberOfCourses:Integer i klassen Student.

- Se till att fältet är privat.
- Uppdatera hibernate.cfg.xml och sätt hbm2ddl.auto till *update*.
- Skapa en ny student i main metoden och spara den i tabellen.
- Kör programmet och kontrollera i databasen att fältet har lagts till.
- Använd *describe student;* i ij för att se om kolumnen lagts till.
- Kontrollera om befintliga rader har NULL i numberOfCourses genom att köra:
  `select * from student;`

Tips:

Hibernate hanterar automatiskt att fält läggs till, men gamla rader får NULL som standard.

Steg 3: Använda ett annat kolumnnamn i databasen

Ibland vill vi ha ett annat namn för kolumnerna i tabellen än namnen på fälten i klassen.

- Ta bort numberOfCourses-kolumnen i databasen (ij)
- Lägg till en ny kolumn NUM_COURSES i tabellen student.

```
alter table student drop column numberOfCourses;
alter table student add column NUM_COURSES integer;
```

- Mappa numberOfCourses i Java-klassen till NUM_COURSES i databasen.

Tips: Använd annoteringen @Column(name="NUM_COURSES") i Java-koden.

- Lägg till denna rad i konstruktorn: `public Student(String name){}`

```
this.numberOfCourses = 10;
```

- Kör nu koden och kontrollera resultatet i ij för att se om du har: 10 numberOfCourses.

Steg 4: Byta tabellnamn

- Ta bort tabellen student från databasen.
- Uppdatera Student-klassen så att den mappar till *TBL_STUDENT* tabell.
- Kör programmet och kontrollera att tabellen har bytt namn.

Instruktioner:

- Använd drop table student; i ij för att ta bort tabellen.
- Lägg till @Table(name="TBL_STUDENT") ovanför Student-klassen.
- Kör programmet och kontrollera med *show tables*; i ij att tabellen nu heter TBL_STUDENT.

Tips:

Hibernate kan automatiskt skapa om tabellen om hbm2ddl.auto är satt till update eller create.

Steg 5: Field access vs. Property access

- Identifiera var @Id-annoteringen sitter i Student-klassen.

OBS. För att kunna använda detta alternativ (property access) behöver vi get och set metoder för alla egenskaper i vår klass.

Kontrollera att dina get- och set-metoder följer standardnamnet i JavaEE (Jakarta EE):

Till exempel:

getName() eller getEnrollmentID()

setName() eller setEnrollmentID()

- Flytta @Id ovanför getId()-metoden istället för fältet id.
- Se till att alla andra annoteringar också flyttas till getter-metoderna.
- Kör programmet och kontrollera att det fortfarande fungerar.

Steg 6: Exkludera ett fält från databasen

- Lägg till ett nytt fält *temporaryData* i Student-klassen.
- Markera fältet med @Transient.om du använder fält access, annars placera @Transient ovanför dess getter-metod.
- Kör programmet och verifiera att kolumnen inte lagts till i databasen.

Tips:

@Transient kan vara användbart för att lagra temporär data som inte behöver sparas i databasen.