Spring - Connection Pools - Övningar

Mål

I denna övning kommer vi att:

- Konfigurera JdbcTemplate i en XML-fil.
- Använda *SimpleDriverDataSource* för att skapa och hantera databaskopplingar.
- Byta ut *SimpleDriverDataSource* mot en mer effektiv connection pool.
- Konfigurera *DBCP* som en connection pool i Spring.
- Implementera bättre kodhantering med init- och destroy-metoder.

Steg 1: Konfigurera JdbcTemplate i XML-filen

- Gå till application.xml och lägg till en bean för JdbcTemplate.
    - klassen för JdbcTemplate: org.springframework.jdbc.core.JdbcTemplate
    - Hämta paketnamnet och egenskaperna från Spring-dokumentationen och konfigurera dessa: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/jdbc/core/JdbcTemplate.html
- Ändra implementeringen av bookDao bean till *BookDaoSpringJdbcImpl*
    - Se till att den tar emot JdbcTemplate som beroende
        - <constructor-arg ref="jdbcTemplate"/>

Steg 2: Skapa en DataSource med SimpleDriverDataSource

JdbcTemplate behöver en DataSource för att fungera. Vi behöver definiera en DataSource-bean.

- Skapa en ny bean (dataSource) för *SimpleDriverDataSource* i application.xml.
- Konfigurera de fyra egenskaperna:
    - driverClass
    - url
    - username
    - password

- Hämta drivernamnet och databasens anslutningsuppgifter från *BookDaoJdbcImpl* klass.
- Injectera dataSource i JdbcTemplate.
    - <constructor-arg ref="dataSource" />
- Kör koden och verifiera att databaskopplingen fungerar.

Tips:

- Du kan hitta den fullständiga dokumentationen för SimpleDriverDataSource på Spring-dokumentationens hemsida.
- Kontrollera att du använder rätt JDBC-driver.

Steg 3: Använda en Connection Pool (DBCP)

Vi ska nu byta ut SimpleDriverDataSource mot en connection pool för att få bättre prestanda.

- Lägg till följande beroende i pom.xml:

```
<!-- https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp -->
<dependency>
     <groupId>commons-dbcp</groupId>
     <artifactId>commons-dbcp</artifactId>
     <version>1.4</version>
</dependency>
```

- Ändra klassen i dataSource-beanen från SimpleDriverDataSource till *BasicDataSource*.
- Ändra egenskapen driverClass till driverClassName eftersom DBCP använder ett annat namn.
- Konfigurera connection poolen genom att ange:
    - driverClassName
    - url
    - username
    - password
- Ta bort de gamla databasfilerna och kör programmet för att verifiera att det fungerar med DBCP

Steg 4: Stänga databaskopplingen korrekt

- För att undvika att databasen blir låst, lägg till destroy-method="close" i dataSource-beanen.
- Radera eventuella lås-filer i projektmappen innan du kör om koden.
- Verifiera att databasen inte genererar nya lås-filer efter att programmet har körts.

Tips:

- Om filen database.dat.lck skapas, betyder det att databasen inte har stängts korrekt.
- destroy-method="close" ser till att anslutningen stängs när Spring Container avslutas.

Steg 5: Förbättra koden med init-metoder

- I *BookDaoSpringJdbcImpl*, flytta SQL-koden som skapar tabellen till en egen metod *createTables().*
- Uppdatera application.xml och lägg till en init-metod i bookDao som anropar createTables() vid uppstart.
- Kör programmet och verifiera att databasen skapas automatiskt vid uppstart

Varför init-metoder?

- Separerar kod och gör *BookDaoSpringJdbcImpl* mer lättläslig.
- Spring Container kan automatiskt anropa metoden vid uppstart.