Übung Book Service Teil 1

Setup Projekt

Analog dem jumpstart Projekt erstellen wir ein Spring Boot Projekt mit Rest Support. Die Screenshots zeigen die Variante mit Eclipse, Spring Initializer und Maven:

JPA Entity Book

Der Book Service liest Bücher aus der Datenbank. Das Modell der Datenbank soll über die JPA Entity Book wie folgt definiert werden:package ch.std.book.jpa;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "book")
public class Book {

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Long id;

 @Column(name = "isbn", unique = true, length = 32)
 private String isbn;

 @Column(name = "title", length = 256)
 private String title;

 @Column(name = "description", length = 1024)
 private String description;

 @Column(name = "publisher", length = 256)
 private String publisher;

 public Book() {
 }

 public Book(String isbn) {
  this.id = null;
  this.isbn = isbn;
  this.title = null;
 this.description = null;
 }

 @Override
 public int hashCode() {
  final int prime = 31;
  int result = 1;
  result = prime * result + ((isbn == null) ? 0 : isbn.hashCode());
  return result;
 }

 @Override
 public boolean equals(Object obj) {
  if (this == obj)
  return true;
  if (obj == null)
  return false;
  if (getClass() != obj.getClass())
  return false;
  Book other = (Book) obj;
  if (isbn == null) {
  if (other.isbn != null)
  return false;
  } else if (!isbn.equals(other.isbn))
  return false;
  return true;
 }

 public Long getId() {
  return id;
 }

 public String getIsbn() {
  return isbn;
 }

 public void setIsbn(String isbn) {
  this.isbn = isbn;
 }

 public String getTitle() {
  return title;
 }

 public void setTitle(String title) {
 this.title = title;
 }

 public String getDescription() {
  return description;
 }

 public String getPublisher() {
  return publisher;
 }

 public void setPublisher(String publisher) {
  this.publisher = publisher;
 }

 public void setDescription(String description) {
  this.description = description;
 }

 @Override
 public String toString() {
  return "Book [id=" + id + ", isbn=" + isbn + ", title=" + title + ", description=" + description
  + ", publisher=" + publisher + "]";
 }

}Integrieren Sie die Klasse Book in Ihr Projekt. In Eclipse können Sie den Programmcode direkt aus dem Clipboard via Paste hineinkopieren.Für den JPA Support benötigen wir noch das folgende Spring Starter Package als Maven Dependency:<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

mysql database

Wir arbeiten mit dieser Übung mit einer mysql Datenbank. Wir setzen die Datenbank mit dem folgenden Schema und Credentials auf:mysql –u root –p
create database jumpstart;
create user 'jumpstart'@'%' identified by 'jumpstart';
grant all on jumpstart.* to 'jumpstart'@'%';Die notwendigen Treiber für mysql definieren wir via Maven Dependency:<dependency>
 <groupId>mysql</groupId>
 <artifactId>mysql-connector-java</artifactId>
 <scope>runtime</scope>
</dependency>

application.properties

Damit unsere Anwendung mit der mysql Datenbank arbeitet, definieren wir die application.properties:spring.jpa.hibernate.ddl-auto=create-drop
spring.datasource.url=jdbc:my

sql://localhost:3306/jumpstart?serverTimezone=UTC
spring.datasource.username=jumpstart
spring.datasource.password=jumpstart
spring.jpa.hibernate.dialect-platform=org.hibernate.dialect.MySQL5InnoDBDialectMit der create-drop ddl-auto Anweisung erstellen wir das Schema mit jedem Start vollständig neu. In der Praxis macht die create-drop Option keinen Sinn, für unsere Übung ist es aber hilfreich.Das SQL Debugging belassen wir wie bisher und erweitern es noch wie folgt:logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.use_sql_comments=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.type=traceDamit zeigen wir alle angewendeten SQL Statements im Log schön formatiert an.

Verify Startup und DB Schema

In der Praxis sollte man die Aufgabe in kleine einfach verifzierbare Steps unterteilen. Wir starten deshalb die Spring Boot Appliktion und verifzieren das Startup Log (ein Auszug):...
 :: Spring Boot :: (v2.3.5.RELEASE)

2020-11-08 11:55:04.833 INFO 4385 --- [main] ch.std.book.BookserviceApplication : Starting BookserviceApplication on ubuntu with PID 4385 (/home/user/sbrs2/bookservice/target/classes started by user in /home/user/sbrs2/bookservice)
2020-11-08 11:55:04.836 INFO 4385 --- [main] ch.std.book.BookserviceApplication : No active profile set, falling back to default profiles: default
2020-11-08 11:55:05.583 INFO 4385 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFERRED mode.
2020-11-08 11:55:05.603 INFO 4385 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 12ms. Found 0 JPA repository interfaces.
2020-11-08 11:55:06.411 INFO 4385 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-11-08 11:55:06.424 INFO 4385 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-11-08 11:55:06.424 INFO 4385 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.39]
2020-11-08 11:55:06.542 INFO 4385 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-11-08 11:55:06.543 INFO 4385 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1661 ms
2020-11-08 11:55:06.872 INFO 4385 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-11-08 11:55:06.945 INFO 4385 --- [task-1] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2020-11-08 11:55:06.988 INFO 4385 --- [task-1] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.4.22.Final
2020-11-08 11:55:07.114 INFO 4385 --- [task-1] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2020-11-08 11:55:07.214 INFO 4385 --- [task-1] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2020-11-08 11:55:07.358 INFO 4385 --- [task-1] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2020-11-08 11:55:07.377 INFO 4385 --- [task-1] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL55Dialect
2020-11-08 11:55:07.919 DEBUG 4385 --- [task-1] org.hibernate.SQL :

 drop table if exists book
Hibernate:

 drop table if exists book
2020-11-08 11:55:07.943 DEBUG 4385 --- [task-1] org.hibernate.SQL :

 create table book (
 id bigint not null auto_increment,
 description varchar(1024),
 isbn varchar(32),
 publisher varchar(256),
 title varchar(256),
 primary key (id)
) engine=InnoDB
Hibernate:

 create table book (
 id bigint not null auto_increment,
 description varchar(1024),
 isbn varchar(32),
 publisher varchar(256),
 title varchar(256),
 primary key (id)
) engine=InnoDB
2020-11-08 11:55:07.948 DEBUG 4385 --- [task-1] org.hibernate.SQL :

 alter table book
 add constraint UK_ehpdfjpu1jm3hijhj4mm0hx9h unique (isbn)
Hibernate:

 alter table book
 add constraint UK_ehpdfjpu1jm3hijhj4mm0hx9h unique (isbn)
2020-11-08 11:55:07.959 INFO 4385 --- [task-1] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2020-11-08 11:55:07.969 INFO 4385 --- [task-1] j.LocalContainerEntityManagerFactoryBean : Initialized JPA

EntityManagerFactory for persistence unit 'default'
2020-11-08 11:55:08.307 WARN 4385 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2020-11-08 11:55:08.957 INFO 4385 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2020-11-08 11:55:08.959 INFO 4385 --- [main] DeferredRepositoryInitializationListener : Triggering deferred initialization of Spring Data repositories…
2020-11-08 11:55:08.959 INFO 4385 --- [main] DeferredRepositoryInitializationListener : Spring Data repositories initialized!
2020-11-08 11:55:08.969 INFO 4385 --- [main] ch.std.book.BookserviceApplication : Started BookserviceApplication in 4.524 seconds (JVM running for 5.385)Weiter prüfen wir ob das DB Schema mit der Tabelle Book korrekt erstellt wurde:sudo mysql -u root -p
show databases;
use jumpstart;
describe book;Das book Schema sollte wie folgt angezeigt werden:MariaDB [jumpstart]> describe book;
+----------+----------+------+-----+---------+----------+
| Field | Type | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+----------+
id	bigint(20)	NO	PRI	NULL	auto_increment
description	varchar(1024)	YES		NULL	
isbn	varchar(32)	YES	UNI	NULL	
publisher	varchar(256)	YES		NULL	
title	varchar(256)	YES		NULL	
+----------+----------+------+-----+---------+----------+

PostConstruct

Nach dem Startup möchten wir Dateien aus einer CSV Datei einlesen. In einem ersten Schritt speichern wir die Datei books.csv im main resources Verzeichnis:Das Einlesen der CSV Datei programmieren wir mit den folgenden Java Klassen:Taskable.javaInitialTask.javaRunnableTask.javaInitialScheduledTask.javaIntegrieren Sie die Java Klassen in Ihr Projekt. Für das Lesen der CSV Datei verwenden wir die Jackson Dataformat Library. Solche bedingt die folgende Maven Dependency:<dependency>
 <groupId>com.fasterxml.jackson.dataformat</groupId>
 <artifactId>jackson-dataformat-csv</artifactId>
</dependency>Das Laden der book csv Datei wird über @ConditionalProperties und damit das Property book.scheduler.initial aktiviert. Definieren Sie für die Aktivierung das folgende Property in der Datei application.properties: book.scheduler.initial.delay=1000
book.scheduler.initial.enabled=trueDas initiale Delay von 1s ist wichtig um nach dem Drop Table ausgeführt zu werden.

Final Run

Nun folgt der finale Run mit dem erneuten Start der Applikation. Die Bücher sollten mit dem Startup geladen werden (siehe Log):...
2020-11-08 12:31:31.724 INFO 6248 --- [pool-1-thread-1] ch.std.book.tasks.impl.InitialTask : IntialTask.run start
2020-11-08 12:31:46.537 INFO 6248 --- [pool-1-thread-1] ch.std.book.tasks.impl.InitialTask : InitialTask.run, insert book done using key = 1
2020-11-08 12:31:50.960 INFO 6248 --- [pool-1-thread-1] ch.std.book.tasks.impl.InitialTask : InitialTask.run, insert book done using key = 2
2020-11-08 12:31:51.007 INFO 6248 --- [pool-1-thread-1] ch.std.book.tasks.impl.InitialTask : InitialTask.run, insert book done using key = 3
2020-11-08 12:31:51.064 INFO 6248 --- [pool-1-thread-1] ch.std.book.tasks.impl.InitialTask : InitialTask.run, insert book done using key = 4
2020-11-08 12:31:51.121 INFO 6248 --- [pool-1-thread-1] ch.std.book.tasks.impl.InitialTask : InitialTask.run, insert book done using key = 5
2020-11-08 12:31:51.181 INFO 6248 --- [pool-1-thread-1] ch.std.book.tasks.impl.InitialTask : InitialTask.run, insert book done using key = 6
2020-11-08 12:31:51.183 INFO 6248 --- [pool-1-thread-1] ch.std.book.tasks.impl.InitialTask : IntialTask.run doneVerifzieren Sie via mysql Konsole ob die Bücher auch wirklich in der Datenbank gespeichert sind:MariaDB [jumpstart]> select id, title from book;
+----+----------+
| id | title |
+----+----------+
1	Spring Boot in Action
2	Cloud Native Java
3	Spring Microservices in Action
4	Spring Boot 2: Moderne Softwareentwicklung mit Spring 5
5	Learning Spring Boot 2.0
6	Mastering Spring Boot

2.0 |
+----+----------+
6 rows in set (0.001 sec)Damit ist dieser 1. Teil der Book Service Übung beendet.

Lösung

Eine mögliche Lösung finden Sie als Maven Projekt bookservice1.zip

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsauschluss.
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

https://www.simtech-ag.ch/46