Übung Book Service Teil 3

JPA Entity Author

Wir erweitern die Book Anwendung mit der JPA Author Entity Klasse: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 = "author")
public class Author {

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

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

  public Author() {
  this(null);
  }

  public Author(String name) {
    this.name = name;
  }

  @Override
  public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((name == null) ? 0 : name.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;
  Author other = (Author) obj;
  if (name == null) {
   if (other.name != null)
    return false;
  } else if (!name.equals(other.name))
   return false;
  return true;
 }

 public Long getId() {
  return id;
 }

 public String getName() {
  return name;
 }

 public void setName(String name) {
  this.name = name;
 }

 @Override
 public String toString() {
  return "Author [id=" + id + ", name=" + name + "]";
 }

}Integrieren Sie die Klasse Book in Ihr Projekt. In Eclipse können Sie den Programmcode direkt aus dem Clipboard via Paste hineinkopieren.Starten Sie die Applikation und verifizeren Sie ob die Entity Author in der Datenbank erstellt wurde:MariaDB [jumpstart]> show tables;
+--------------------+
| Tables_in_jumpstart |
+--------------------+
| author |
| book |
+--------------------+
2 rows in set (0.000 sec)

Book Author

Jetzt fehlt noch die Bidirektionale M:N Relation zwischen der Entity Book und Author. JPA deckt solche Relationen mit der @ManyToMany Relation, wobei eine Seite das Mapping übernimmt.In der Klasse Author definieren wir die @ManyToMany Relation wie folgt:@ManyToMany(mappedBy="authorList")
private List<Book> bookList;Definieren Sie die Getter-/Setter Methoden und initalisieren Sie die bookList Instanz im Konstruktor der Author Klasse. Seitens Book erfolgt der Gegenpart, jedoch mit der Definition der Relation wie folgt:@ManyToMany
@JoinTable(name = "book_author", joinColumns = { @JoinColumn(name = "fk_book") }, inverseJoinColumns = {
 @JoinColumn(name = "fk_author") })
private List<Author> authorList;Definieren Sie die Getter-/Setter Methoden und initalisieren Sie die bookList Instanz im Konstruktor der Author Book. Starten Sie die Applikation und verifizeren Sie ob die Tabelle book_author in der Datenbank erstellt wurde:
MariaDB [jumpstart]> show tables;
+--------------------+
| Tables_in_jumpstart |
+--------------------+
| author |
| book |
| book_author |
+--------------------+

Author Import

Die Datei books.csv enthält neben Büchern auch die Autoren. Solche wollen wir nun mit dem Start der Applikation in die Datenbank einlesen.Hierzu ersetzen wir die aktuelle InitialTask Klasse mit der neuen Variante:InitialTask.javaDie InitialTask Klasse parsed das CSV Dokument und speichert die Daten via JDBC Statements in der Datenbank und nicht via JPA Instanzen.Starten Sie die Applikation und verifizeren Sie ob die Tabelle author in der Datenbank die Autoren importierte:MariaDB [jumpstart]> select id, name from author;
+----+------------------+
| id | name |
+----+------------------+
1	Craig Walls
2	Josh Long
3	Kenny Bastani
4	John Carnell
5	Michael Simons
6	Greg L. Turnquist
7	Dinesh Rajput
+----+------------------+
7 rows in set (0.001 sec)Testen Sie die Einträge in der book_author Relationstabelle:MariaDB [jumpstart]> select * from book_author;
+---------+-----------+
| fk_book | fk_author |
+---------+-----------+
1	2
1	3
2	4

3 | 5 |
| 4 | 6 |
| 5 | 7 |
+---------+-----------+
6 rows in set (0.000 sec)Verifizieren Sie mit dem Maven Konsolenbefehl ob die Unit Tests noch funktionieren:mvn clean installDer Test Run sollte keine Fehler zeigen, wir testen ja nur die einzelne Book Entity auf der JPA Ebene.

Exception

Mit dem Neustart der Applikation und dem Anwenden des http://localhost:8080/book/rest/books REST Endpoints ergibt sich leider ein Fehler oder Exception:Das Problem liegt an der bidirektionalen Verlinkung zwischen Book und Author Instanzen, welche eine Rekursion im JSON Mapper auslöst. Das JSON Mapping basiert auf Jackson Rules und damit Annotations. Um den Konflikt zu lösen müssen wir eine Seite vom Renderung abkoppeln und dies via @JsonIgnoreProperties Annotation, Klasse Book@JsonIgnoreProperties("bookList")
public List<Author> getAuthorList() {
 return authorList;
}Klasse Author@JsonIgnoreProperties("authorList")
public List<Book> getBookList() {
 return bookList;
}Mit dem Restart der Applikation funktioniert der REST Endpoint wieder korrekt.

AuthorRepository

Programmiernen Sie die AuthorRepository JPA Klasse. Erstellen Sie die Unit Test Klasse AuthorRepositoryJPATest und testen Sie die Author Entity analog der Book Entity (siehe BookRepositoryJPATest) aus:package ch.std.book.repositories;
...
@ExtendWith(SpringExtension.class)
@DataJpaTest
@ActiveProfiles("unittest") // alternativ kann mit der @TestPropertySource Annotation gearbeitet werden
public class AuthorRepositoryJPATest {

 @Autowired
 private AuthorRepository authorRepository;

 private Author author;
 private List<Author> authorList;

 @BeforeEach
 public void setup() { ... }

 @AfterEach
 public void tearDown() { ... }

 @Test
 public void testFindById() { ... }

 @Test
 public void testFindByName() { ... }

 @Test
 public void testFindAll() { ... }

 @Test
 public void testFindAllSorted() { ... }

}Testen Sie via mvn clean install auf der Konsole.

BookAuthorRepositoryJPATest

Im BookRepository definieren wir die Spring Data Join Methode:List<Book> findByAuthorList_Name(String authorName);Alsdann erstellen wir die folgende Unit Test Klasse:package ch.std.book.repositories;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;

import java.util.List;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
import org.springframework.test.context.ActiveProfiles;
import org.springframework.test.context.junit.jupiter.SpringExtension;

import ch.std.book.jpa.Author;
import ch.std.book.jpa.Book;

@ExtendWith(SpringExtension.class)
@DataJpaTest
@ActiveProfiles("unittest")
public class BookAuthorRepositoryJPATest {

 @Autowired
 private BookRepository bookRepository;
 @Autowired
 private AuthorRepository authorRepository;
 private Book book;
 private Author author;

 @BeforeEach
 public void setup() {
 book = new Book("978-1617292545", "this is the title", "that's the description", "my publisher");
 author = new Author("my author");
 author.getBookList().add(book);
 book.getAuthorList().add(author);
 this.bookRepository.save(book);
 this.authorRepository.save(author);
 }

 @Test
 public void testFindByAuthorName() { ... }

}Testen Sie via mvn clean install auf der Konsole.

Lösung

Eine mögliche Lösung finden Sie als Maven Projekt bookservice3.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/getbooklist