REPORT JAVA

Dependencies

Το build management tool που χρησιμοποιήθηκε είναι το maven και το framework, το Spring Boot

Πληροφορίες Έργου:

Όνομα: booksapi

Περιγραφή: REST εφαρμογή που διεξάγει ερωτήματα στο Google Books API

Έκδοση: 0.0.1-SNAPSHOT

Έκδοση Java: 17

Εξαρτήσεις:

Spring Boot Starter Data JPA

GroupId: org.springframework.boot

ArtifactId: spring-boot-starter-data-jpa

Έκδοση: Προέρχεται από τον γονέα (3.1.2)

Σκοπός: Παρέχει τις λειτουργίες του Spring Data JPA για ευκολία στην αλληλεπίδραση με βάσεις δεδομένων χρησιμοποιώντας το Spring.

Spring Boot Starter Web

GroupId: org.springframework.boot

ArtifactId: spring-boot-starter-web

Έκδοση: Προέρχεται από τον γονέα (3.1.2)

Σκοπός: Κύριες βιβλιοθήκες για τη δημιουργία web εφαρμογών, συμπεριλαμβανομένων των RESTful, χρησιμοποιώντας το Spring MVC.

Gson

GroupId: com.google.code.gson

ArtifactId: gson

Έκδοση: 2.10.1

Σκοπός: Μια βιβλιοθήκη σε Java για σειριοποίηση και αντισειριοποίηση για τον μετασχηματισμό των Java Objects σε JSON και αντίστροφα.

OkHttp

GroupId: com.squareup.okhttp3

ArtifactId: okhttp

Έκδοση: 5.0.0-alpha.11

Σκοπός: Ένας αποτελεσματικός πελάτης HTTP και HTTP/2 για εφαρμογές Android και Java.

Lombok

GroupId: org.projectlombok

ArtifactId: lombok

Έκδοση: Προέρχεται από τον γονέα (3.1.2)

Σκοπός: Μια βιβλιοθήκη Java που συνδέεται αυτόματα με τον επεξεργαστή και τα εργαλεία δημιουργίας για την παροχή λειτουργιών μείωσης του boilerplate κώδικα.

H2 Database

GroupId: com.h2database

ArtifactId: h2

Εμβέλεια: runtime

Σκοπός: Μια SQL βάση δεδομένων σε Java που βρίσκεται στη μνήμη και είναι χρήσιμη για την ανάπτυξη και τον έλεγχο.

Spring Boot Starter Test

GroupId: org.springframework.boot

ArtifactId: spring-boot-starter-test

Έκδοση: Προέρχεται από τον γονέα (3.1.2)

Εμβέλεια: test

Σκοπός: Έναρξη για τον έλεγχο των εφαρμογών του Spring Boot με βιβλιοθήκες όπως JUnit, Hamcrest, και Mockito.

# Architecture

Χρησιμοποιείται το model view controller architecture.

Η main έχει το command line text για τον χρήστη και προσομειώνει τον client που στέλνει request στο Controller class GoogleBooksApiController, όπου έχουμε endpoint methods τα οποία χρησίμευσαν και για testing μέσω POSTMAN.

Ύστερα από το endpoint method πέφτουμε μέσα στο service layer όπου έχουμε τη λογική μας και εκτελούμε τα requests στο Books API της Google. Ένα method ανά request. Το OkHttp library χρησιμοποιήθηκε για την εκτέλεση των HTTP requests.

Καθώς χρησιμοποιούμε Spring Boot, dependency injection υπάρχει σε διάφορα σημεία στον κώδικα καθώς και άλλα annotations όπως @Service, @RestController, etc

Ενδεικτικά, ο σχολιασμός μιας από τις service layer methods:

@Override  
*public* VolumeList getBookshelfVolumes(String bookshelfId) {  
 OkHttpClient client = *new OkHttpClient*();  
  
 String url = String.*format*("https://www.googleapis.com/books/v1/users/%s/bookshelves/%s/volumes?key=%s", userId, bookshelfId, API\_KEY);  
  
 Request request = *new* Request.*Builder*()  
 .*url*(url)  
 .*build*();  
  
 *try* (Response response = client.*newCall*(request).*execute*()) {  
 *if* (!response.*isSuccessful*()) *throw new IOException*("Unexpected code " + response);  
  
 String responseBody = response.*body*().*string*();  
 *return new Gson*().*fromJson*(responseBody, VolumeList.*class*);  
 } *catch* (IOException e) {  
 *throw new RuntimeException*("Failed to fetch volumes for the bookshelf: " + e.*getMessage*());  
 }  
}

Η μέθοδος getBookshelfVolumes είναι υπεύθυνη για την ανάκτηση των τόμων από μια συγκεκριμένη βιβλιοθήκη βάσει της δοθείσας ταυτότητας bookshelfId. Χρησιμοποιείται ο πελάτης HTTP OkHttpClient για τη διεξαγωγή του αιτήματος. Η διεύθυνση URL για το αίτημα δημιουργείται με βάση μια δοθείσα ταυτότητα χρήστη, την ταυτότητα της βιβλιοθήκης και ένα κλειδί API. Εάν η απάντηση είναι επιτυχής, η μέθοδος αναλύει την απάντηση JSON σε ένα αντικείμενο τύπου VolumeList. Σε περίπτωση αποτυχίας, η μέθοδος πετάει μια εξαίρεση, δηλώνοντας ότι απέτυχε η προσπάθεια ανάκτησης των τόμων για τη συγκεκριμένη βιβλιοθήκη.

Επίσης έγινε overloading της toString στα απαραίτητα data classes, με το παρακάτω

@Override  
*public* String toString() {  
 *return new Gson*().*toJson*(*this*);  
}

Προκειμένου να αναπαρίσταται σωστά το επιστρεφόμενο αποτέλεσμα, σε JSON format.

# Model

Για την δημιουργία του model ακολουθήθηκε η εξής τεχνική; Στάλθηκαν requests απευθείας στο Google Books API από το POSTMAN και ύστερα έκανα τα JSON responses curate και μετά τα χρησιμοποίησα ως βάση για να φτιάξω τα POJO classes. Για οικονομία χώρου χρησιμοποίησα το Lombok library.

# Resources

Από το application.properties διαβάζω την πόρτα από την οποία θα ακούει ο tomcat server, το API KEY και το userid

server.port = 9092  
google.books.api.key=AIzaSyCaJxCnoSV9Aawlms-ZCfkGYdN39GJCsns  
google.books.api.userid=102701940585560677579

και χρησιμοποιώ τα @Value annotations για να τα διαβάσω στον κώδικα

@Value("${google.books.api.key}")  
*private* String API\_KEY;  
  
@Value("${google.books.api.userid}")  
*private* String userId;