LU02c - Funktionale Implementationen 3

Konfigurationsvariabeln

URLs, Benutzernamen und Passwörter sollten nicht hartcodiert werden, da sonst jeder mit Read-Rechten die Zugangsdaten lesen und somit auf die Datenbank zugreifen kann.

Es gibt verschiedene Arten, um solche Variabeln einzulesen, wie zum Beispiel: Command-line Argumente, System Properties, eine `.env`-Datei, usw.

Für native Java eignet sich `java.util.Properties`

- https://www.baeldung.com/java-properties

1. Erstellen Sie die Datei `config.properties` anhand von `config.properties.template`.
2. Initialisieren Sie ein `Properties`-Objekt:

   ```
   new Properties()
   ```

3. Laden Sie die Datei aus dem Root-Verzeichnis:

   ```
   new FileInputStream("config.properties")
   ```

4. Holen Sie die Werte der Konfigurationen von `DB_URL`, `DB_USER` und `DB_PASSWORD` aus den Properties.

`config.properties` selbst wird nicht im Git-Repository eingecheckt, sondern existiert nur lokal. So ist sichergestellt, dass sensible Angaben nicht „geleakt“ werden.

Führen Sie die Tests erneut aus und korrigieren Sie ihre Implementation falls nötig.

Für die Pipeline wird eine separate Testdatenbank benötigt. Studieren Sie die Ergänzungen der Pipeline und die beiden SQL-Dateien, um den Einsatz der Postgres-Service zu verstehen.

I/O

Stellen Sie sich vor, dass die Betreiber einer Bibliothek bisher eine Excel-Datei für die Verwaltung der Bücher genutzt haben .

Anstatt alle darin enthaltenen Bücher abzutippen und manuell in die Datenbank einzutragen, möchten sie eine Import-Funktionalität. Da die Buchtitel teilweise Kommas und Semikolons enthalten, verwenden wir `\t` als Trennzeichen für den Export. Dieser ist zudem in UTF-8, was auch die Standardcodierung in Git ist.

Anforderung 3: Mit dem Befehl `importBooks <FILE_PATH>` soll die angegebene TSV-Datei eingelesen werden und in die Datenbank gespeichert werden. `<FILE_PATH>` steht dabei natürlich für einen Pfad und Dateinamen.

1. Erweitern Sie die Befehlslogik, so dass der Befehl `importBooks` mit dem Pfad als Argument

erkannt wird.
2. Schreiben Sie die Logik, um die Datei ohne zusätzliche Dependencies einzulesen und eine `List<Book>` daraus zu erstellen. Beispielcode für einen solchen Reader finden Sie u.a. auf https://www.baeldung.com/java-csv-file-array
3. Speichern Sie die resultierende Liste in die Datenbank. Einträge mit derselben `id` sollen überschrieben werden, damit man auch Korrekturen mit dieser Funktion einlesen kann. https://www.baeldung.com/java-jdbc
4. Importieren Sie die Datei `books.tsv` im Ordner `data`

```
importBooks data/books.tsv
```

5. Überprüfen Sie die Anforderung mit den Tests im Commit `670f42e`. Aufgrund eines Fehlers in der Testdatei benötigen Sie zudem den Commit `a3f9cea`.

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/ffit/3-jahr/java/learningunits/lu02/c?rev=1756462889

Last update: **2025/08/29 12:21**