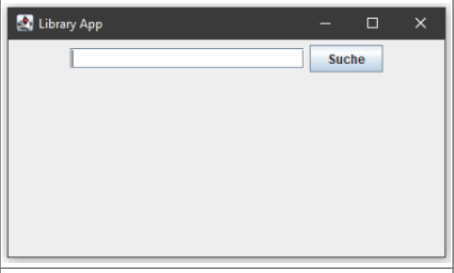
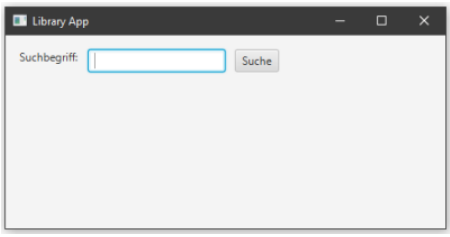
LU04a - GUI & API

Bisher haben wir die Library App der Einfachheit halber via Konsole bedient, aber eine grafische Benutzeroberfläche ist gewünscht.

Java GUI

Es gibt verschiedene Java-GUI-Frameworks, die teilweise historisch aufeinander aufbauen. Beispiele sind AWT (Abstract Window Toolkit), Swing und JavaFX. Sie ermöglichen die Entwicklung von klassischen Desktop-Anwendungen.

Java AWT	Java Swing	JavaFX
		
1996 eingeführt mit Java Version 1.0	1998 eingeführt mit Java Version 1.2	2008 eigenständig, bei Java Version 8 und ab 11 inklusive

In der heutigen Praxis sind Desktop-Applikationen jedoch im Unternehmensumfeld eher unüblich geworden. Stattdessen dominieren webbasierte Benutzeroberflächen, da sie unabhängig vom Betriebssystem im Browser laufen und einfacher verteilt werden können.

Heutzutage werden Benutzeroberflächen meist als Web-GUIs umgesetzt. Das Frontend ist dabei vom Backend entkoppelt und kommuniziert über eine REST-API oder andere Schnittstellen.

- Das Backend (z. B. in Java) stellt Daten und Geschäftslogik bereit.
- Das Frontend (z. B. in Angular, React oder Vue) übernimmt Darstellung und Interaktion.

Durch diese Trennung können beide Seiten unabhängig voneinander weiterentwickelt werden.

JSF

Es gibt aber auch Frameworks wie Jakarta Faces (JSF), bei denen der Java-Code direkt HTML-Seiten generiert. Die Anwendung läuft dann sowohl als Backend als auch als Webserver, der die Oberfläche bereitstellt. Dieses Modell koppelt Frontend und Backend enger zusammen und ist vor allem in klassischen Unternehmensanwendungen verbreitet.

REST API

Das Architekturmuster REST (Representational State Transfer) ist heutzutage am gebräuchlichsten.

REST nutzt die Standardmethoden des HTTP-Protokolls für den Datenaustausch:

- **GET** → Anfrage nach Informationen (Lesen von Daten)
- **POST** → Übermittlung neuer Daten (Anlegen eines Datensatzes)
- **PUT** → Übermittlung von Daten zur vollständigen Aktualisierung (Überschreiben eines Datensatzes)
- **PATCH** → Übermittlung von Daten zur teilweisen Aktualisierung
- **DELETE** → Aufforderung zur Löschung eines Datensatzes

Javalin

Javalin ist ein leichtgewichtiges Web Framework (ähnlich wie Flask für Python).

Anforderung 4: Die Logik des Befehls `listBooks` soll via API-Aufruf ausgeführt werden können.

Die notwendige Abhängigkeit zu `io.javalin:javalin` wurde mit dem Commit `a091fe7` in `build.gradle` ergänzt.

Erstellen Sie eine neue Klasse `ch.bzz.JavalinMain` mit einer Main-Methode und implementieren Sie Abfrage der Book-Objekte gemäss https://javalin.io/documentation.

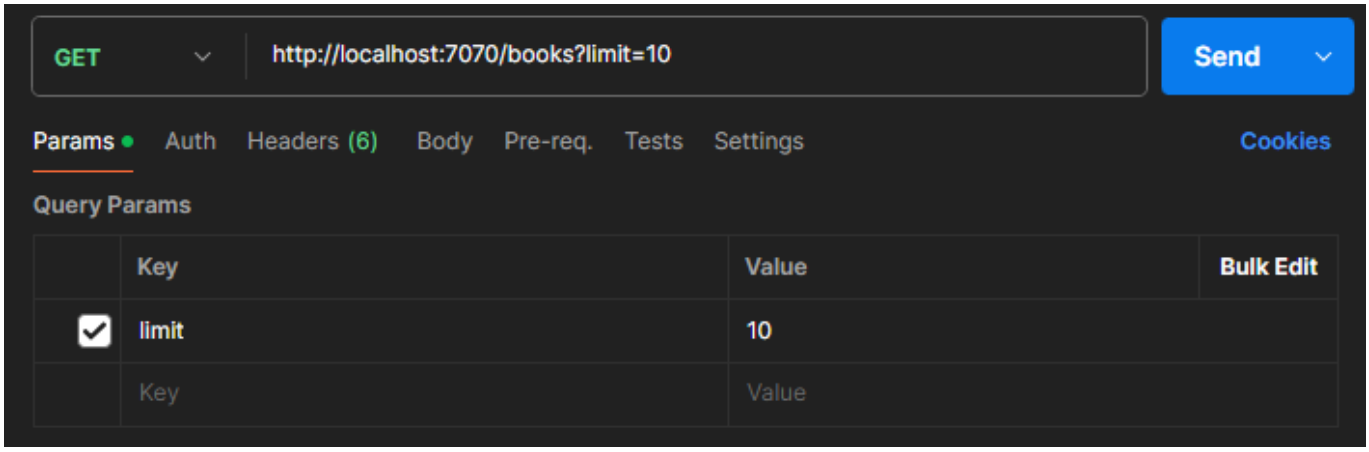
Die Route soll `/books` lauten und `7070` soll als Port verwendet werden. Das Limit wird via QueryParam mitgegeben. Die Bücherliste soll als JSON-Objekt serialisiert zurückgegeben werden.

Testen Sie Ihre Applikation manuell unter: http://localhost:7070/books?limit=10

Führen Sie zudem die Tests in der Klasse `JavalinMainTest` aus und korrigieren Sie Ihre Implementation bei Bedarf.

Postman

Um POST,PUT etc. zu testen, empfiehlt sich Postman. Installieren Sie Postman (https://www.postman.com/downloads/) und testen Sie die Funktionsweise mit dem bereits implementierten API.



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

Permanent link:
https://wiki.bzz.ch/de/modul/ffit/3-jahr/java/learningunits/lu04/a

Last update: **2025/09/09 00:10**