2. Microservices with API gateway

Creating Microservices for account and loan

In this hands on exercises, we will create two microservices for a bank. One microservice for handing accounts and one for handling loans.

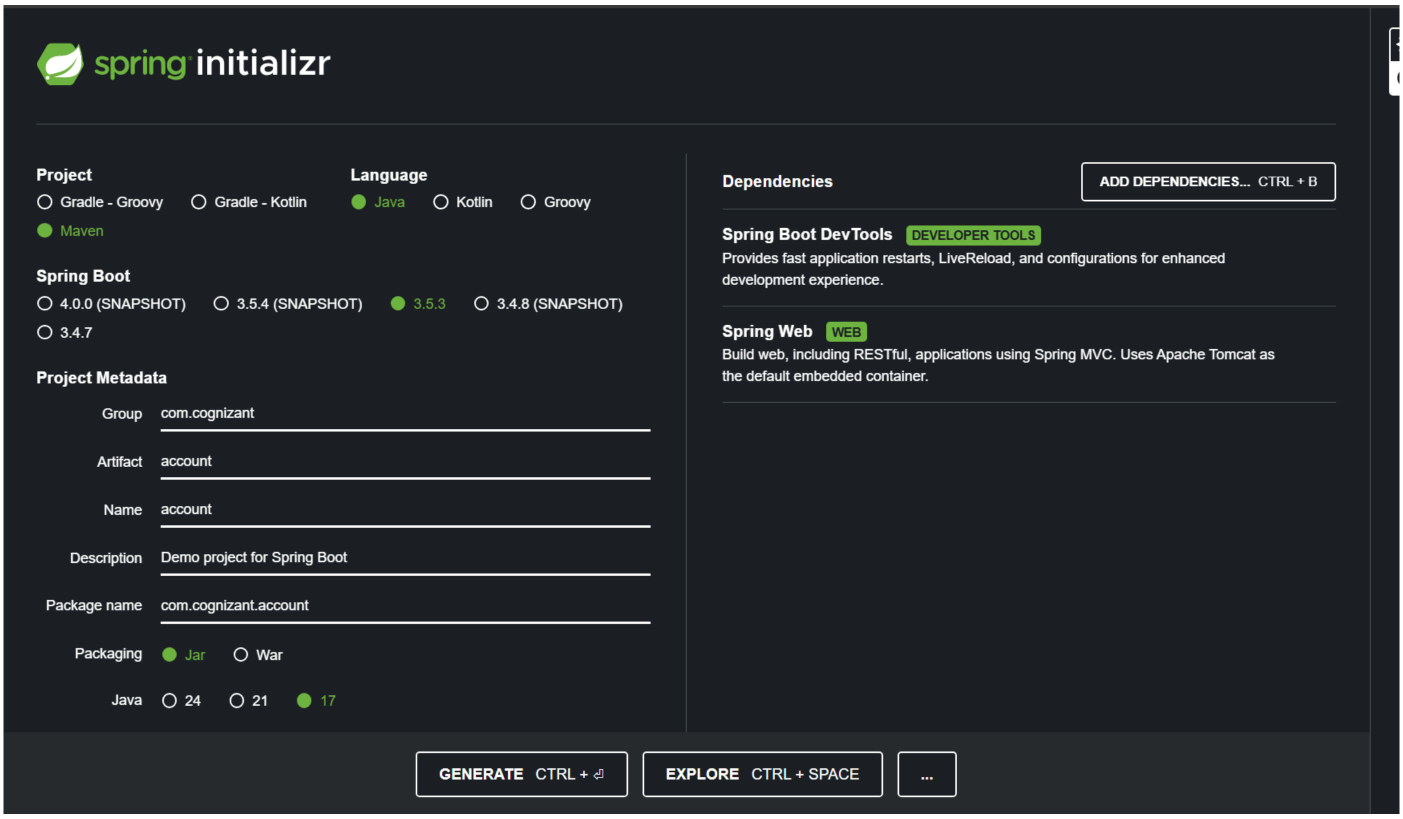
Each microservice will be a specific independent Spring RESTful Webservice maven project having it's own pom.xml. The only difference is that, instead of having both account and loan as a single application, it is split into two different applications. These webservices will be a simple service without any backend connectivity.

Follow steps below to implement the two microservices:

Account Microservice

- Create folder with employee id in D: drive
- Create folder named 'microservices' in the new folder created in previous step. This folder will contain all the sample projects that we will create for learning microservices.
- Open https://start.spring.io/ in browser
- Enter form field values as specified below:
    - **Group:** com.cognizant
    - **Artifact:** account
- Select the following modules
    - Developer Tools > Spring Boot DevTools
    - Web > Spring Web
- Click generate and download the zip file
- Extract 'account' folder from the zip and place this folder in the 'microservices' folder created earlier
- Open command prompt in account folder and build using mvn clean package command
- Import this project in Eclipse and implement a controller method for getting account details based on account number. Refer specification below:
    - Method: GET
    - Endpoint: /accounts/{number}
    - Sample Response. Just a dummy response without any backend connectivity.

```
{ number: "00987987973432", type: "savings", balance: 234343 }
```



Project: Gradle - Groovy, Gradle - Kotlin [✓ Maven]

Language: [✓ Java] Kotlin, Groovy

Spring Boot: 4.0.0 (SNAPSHOT), 3.5.4 (SNAPSHOT), [✓ 3.5.3] 3.4.8 (SNAPSHOT), 3.4.7

Project Metadata

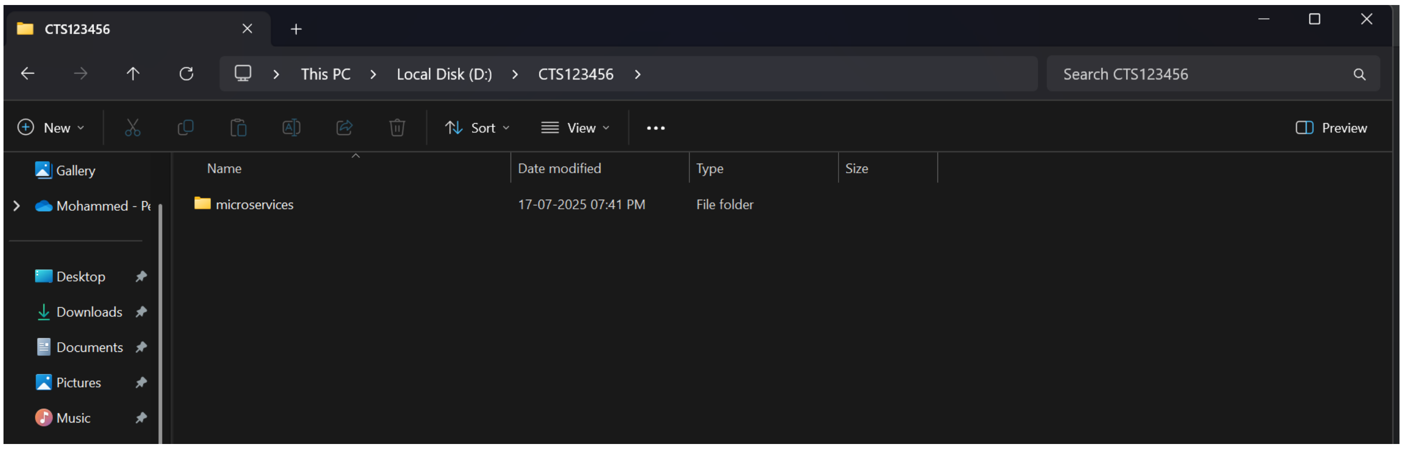
Group	com.cognizant
Artifact	account
Name	account
Description	Demo project for Spring Boot
Package name	com.cognizant.account
Packaging	[✓ Jar] War
Java	24, 21 [✓ 17]

Dependencies

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

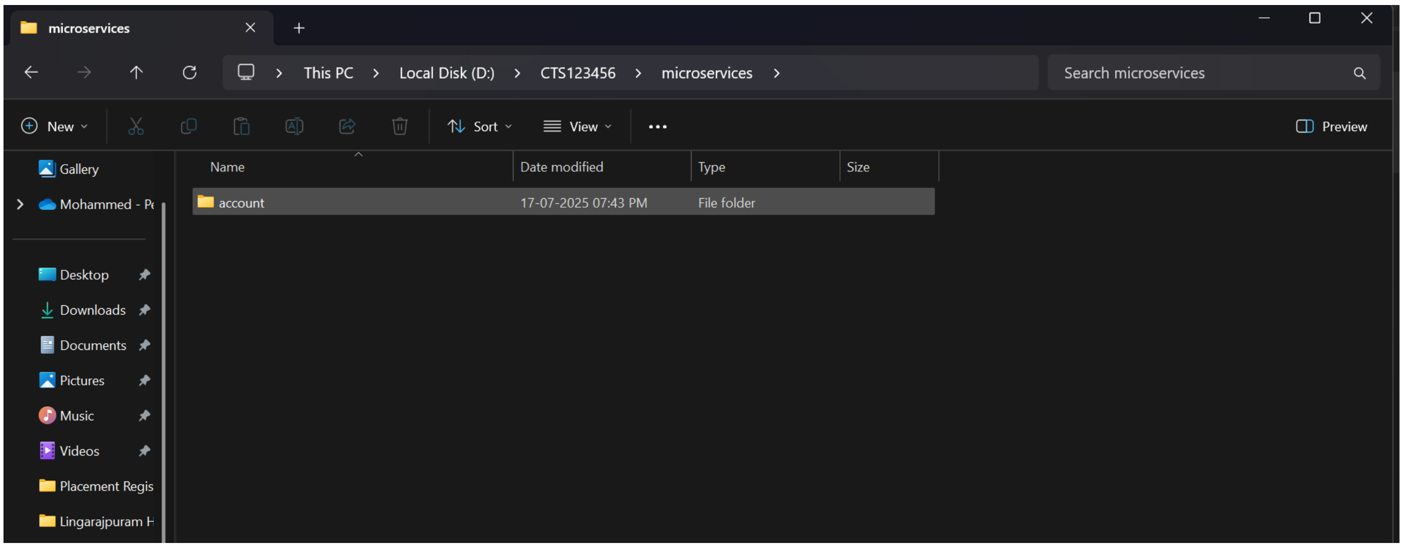
Create a folder in D drive



Name	Date modified	Type	Size
microservices	17-07-2025 07:41 PM	File folder	

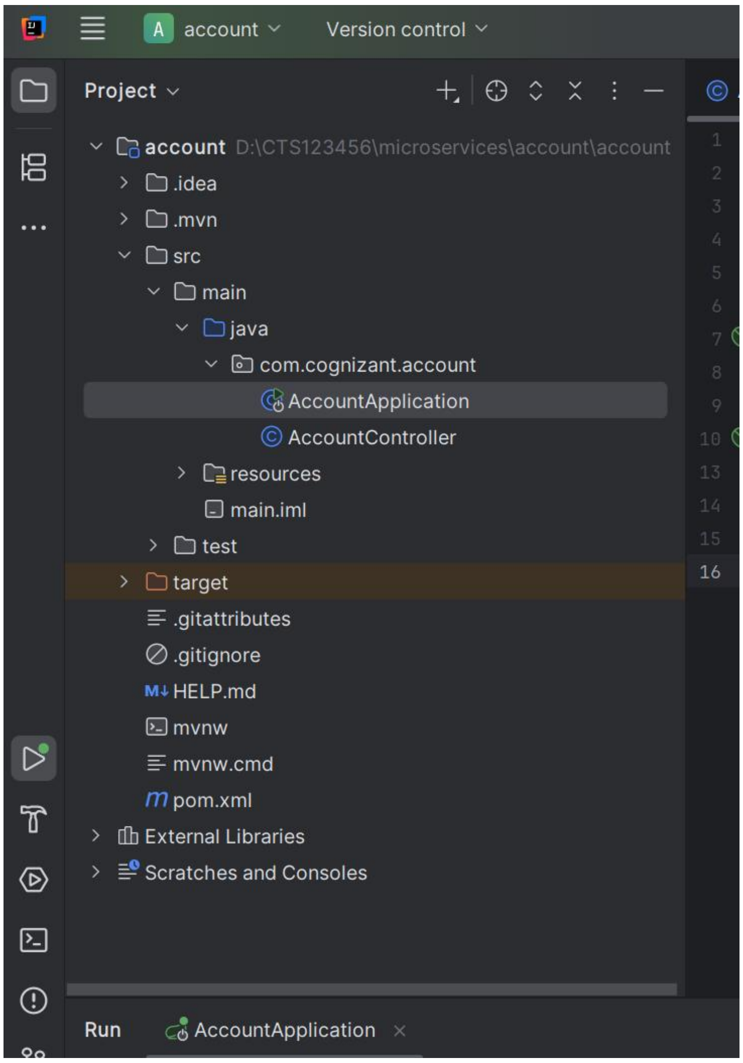
Create a folder named microservices inside the created folder

Then, extract the spring initializr file and paste it inside the “microservices” folder



Name	Date modified	Type	Size
account	17-07-2025 07:43 PM	File folder	

Folder Structure:

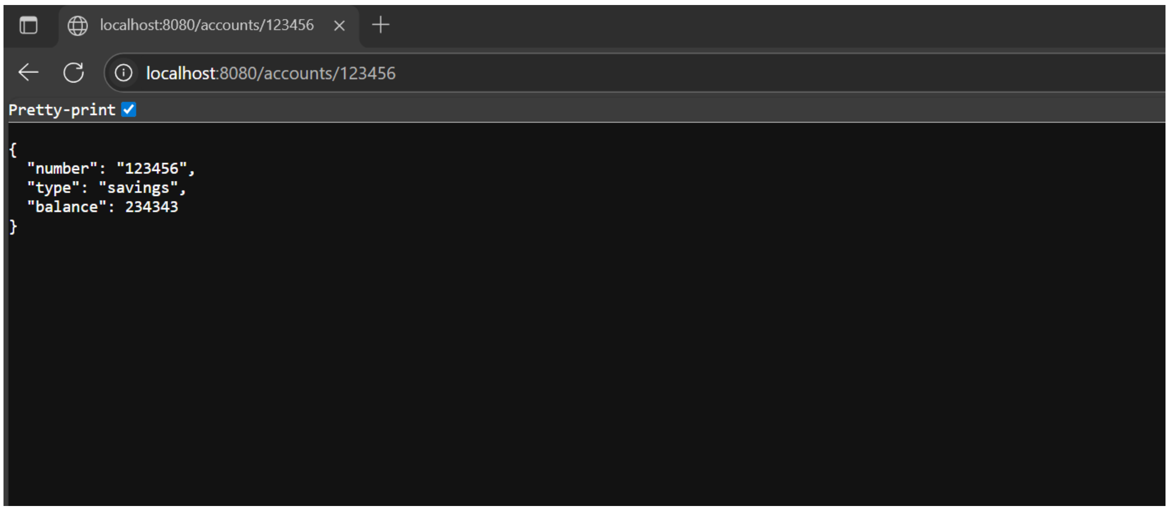


```
account D:\CTS123456\microservices\account\account
  .idea
  .mvn
  src
    main
      java
        com.cognizant.account
          AccountApplication
          AccountController
      resources
      main.iml
    test
  target
  .gitattributes
  .gitignore
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

AccountController.java

```
package com.cognizant.account;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/accounts")
public class AccountController {

    @GetMapping("/{number}")
    public Account getAccount(@PathVariable String number) {
        return new Account(number, "savings", 234343);
    }

    record Account(String number, String type, double balance) {}
}
```

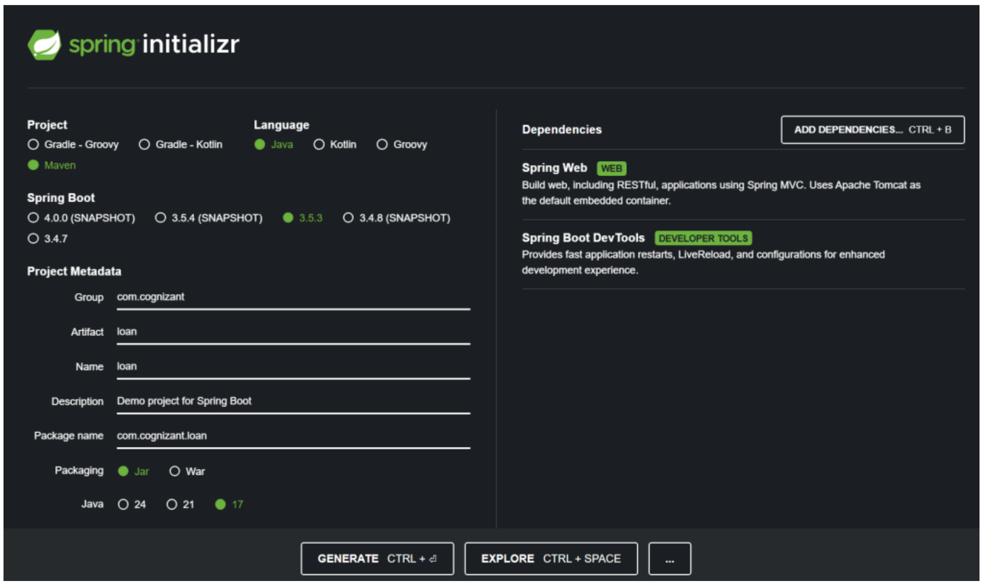
Then, run AccountApplication.java and go to localhost:8080/accounts/123456


localhost:8080/accounts/123456

Pretty-print ✓

```
{
  "number": "123456",
  "type": "savings",
  "balance": 234343
}
```

Creating Loan microservice:



Project: ○ Gradle - Groovy ○ Gradle - Kotlin [✓ Maven]

Language: [✓ Java] ○ Kotlin ○ Groovy

Spring Boot: ○ 4.0.0 (SNAPSHOT) ○ 3.5.4 (SNAPSHOT) [✓ 3.5.3] ○ 3.4.8 (SNAPSHOT) ○ 3.4.7

Project Metadata

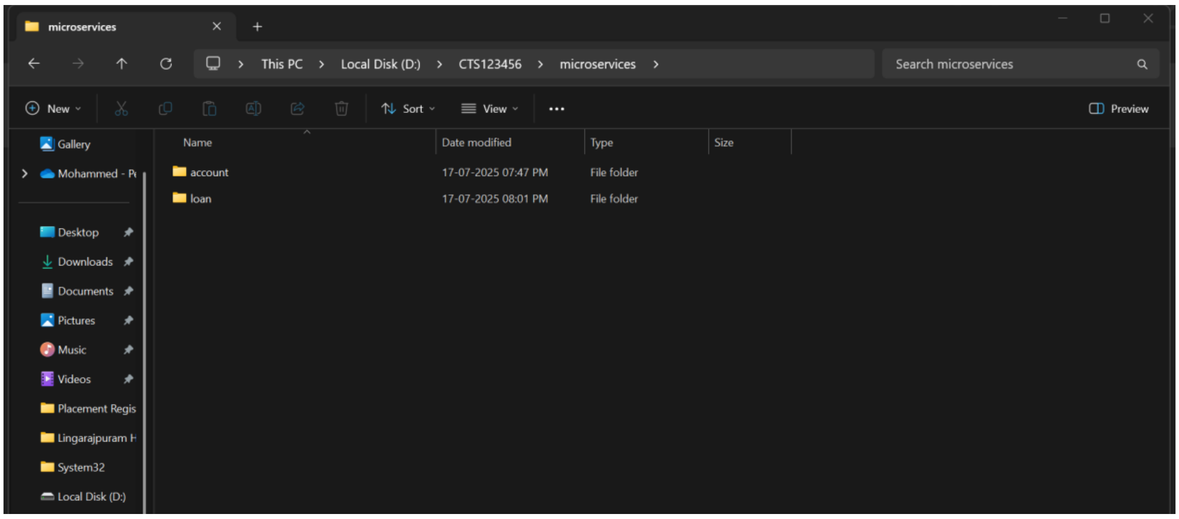
Group	com.cognizant
Artifact	loan
Name	loan
Description	Demo project for Spring Boot
Package name	com.cognizant.loan
Packaging	[✓ Jar] ○ War
Java	○ 24 ○ 21 [✓ 17]

Dependencies

Spring Boot DevTools WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

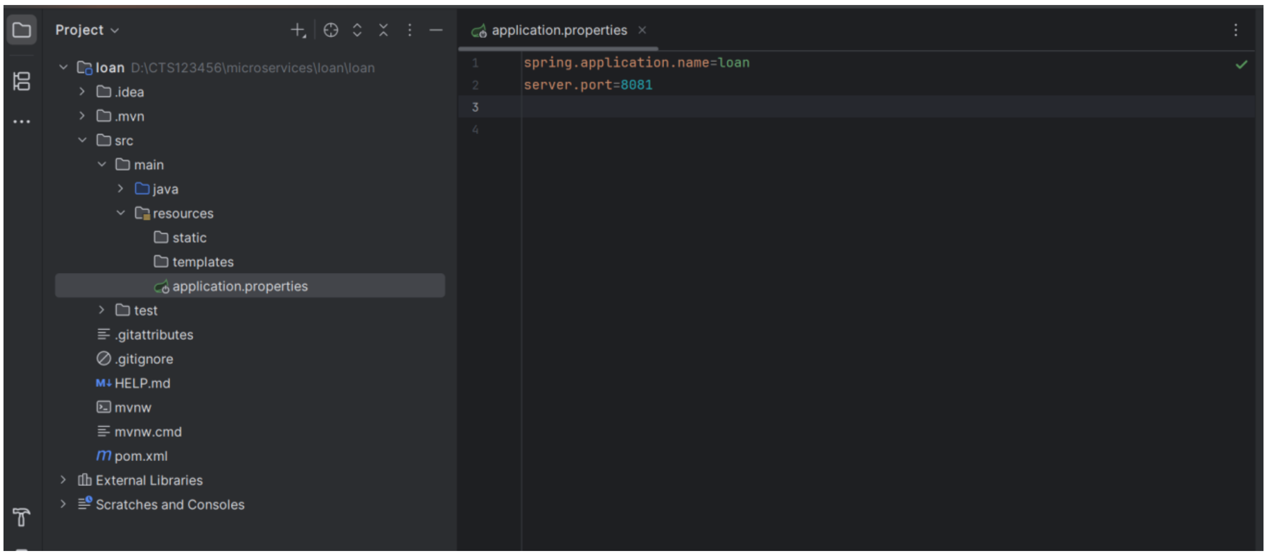
Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Then move it to the same old folder,



Name	Date modified	Type	Size
account	17-07-2025 07:47 PM	File folder	
loan	17-07-2025 08:01 PM	File folder	

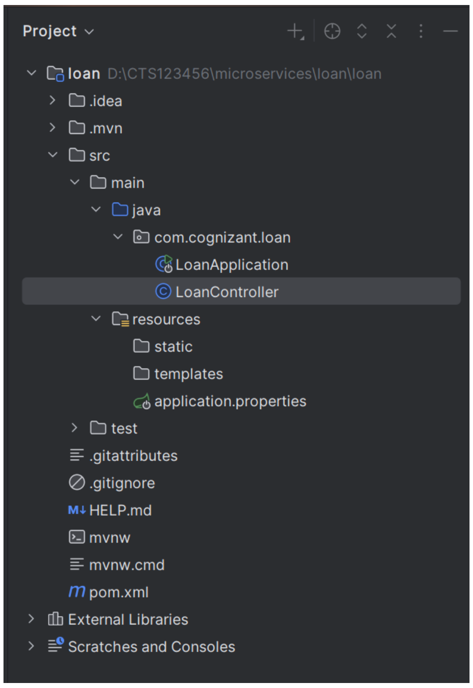
Set custom port for the loans project by adding → server.port=8081



application.properties

```
spring.application.name=loan
server.port=8081
```

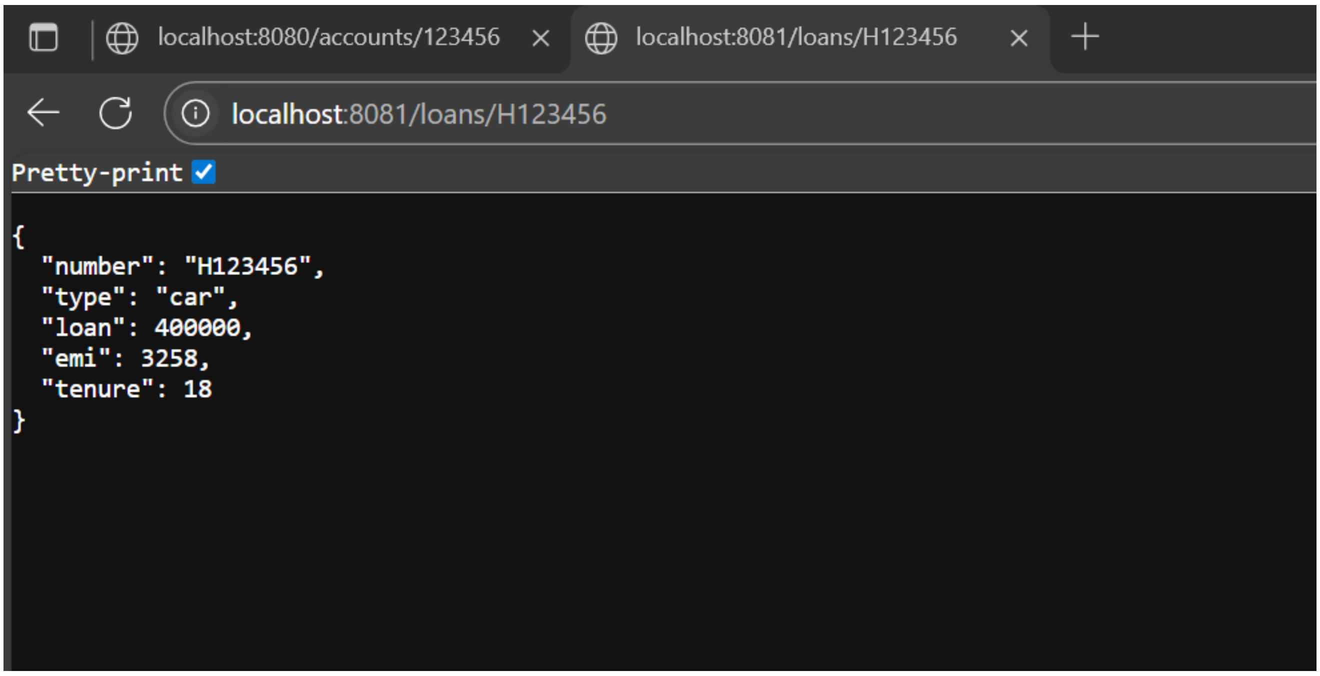
Folder structure:



LoanController.java

```
package com.cognizant.loan;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/loans")
public class LoanController {

    @GetMapping("/{number}")
    public Loan getLoan(@PathVariable String number) {
        return new Loan(number, "car", 400000, 3258, 18);
    }

    record Loan(String number, String type, double loan, double emi, int tenure) {}
}
```

Run LoanApplication.java and go to http://localhost:8081/loans/H123456


localhost:8080/accounts/123456
localhost:8081/loans/H123456
localhost:8081/loans/H123456
Pretty-print ✓

```
{
  "number": "H123456",
  "type": "car",
  "loan": 400000,
  "emi": 3258,
  "tenure": 18
}
```

I now have two microservices,

1) Account on port 8080
2) Loan on port 8081

Thank You