**Digital Nurture Java FSE**

**Week 5 – Microservices**

**Mandatory Exercise**

**1. Creating Microservices for account and loan**

* **Account:**

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 i
* 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 }**

Launch by running the application class and test the service in browser.

**Account.Controller:**

package com.cognizant.account.controller;

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

*@RestController*

*@RequestMapping*("/accounts")

public class AccountController {

*@GetMapping*("/{number}")

public Account getAccountDetails(*@PathVariable* String number) {

return new Account(number, "savings", 234343);

}

static class Account {

public String number;

public String type;

public double balance;

public Account(String number, String type, double balance) {

this.number = number;

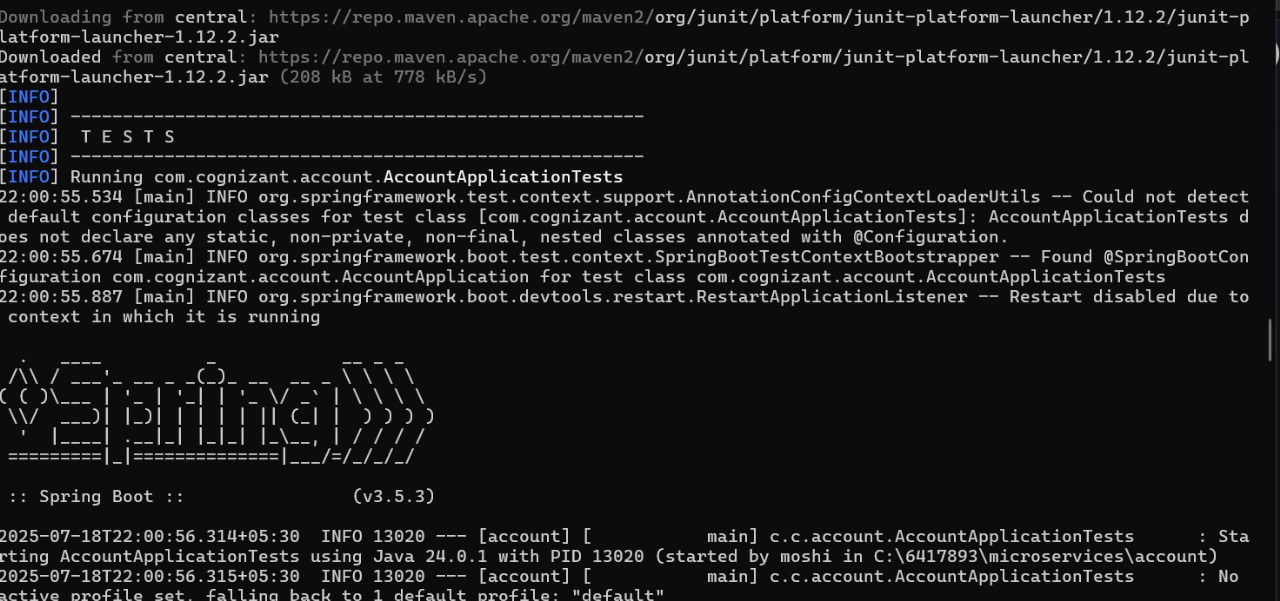
this.type = type;

this.balance = balance;

}

}

}



**AccountApplicationTest.java:**

package com.cognizant.account;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

*@SpringBootTest*

class AccountApplicationTests {

*@Test*

void contextLoads() {

}

}

**Pom.xml:**

?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>account</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>account</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

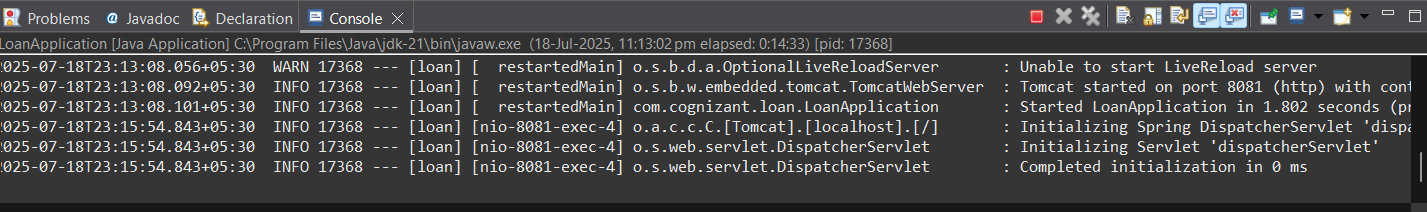
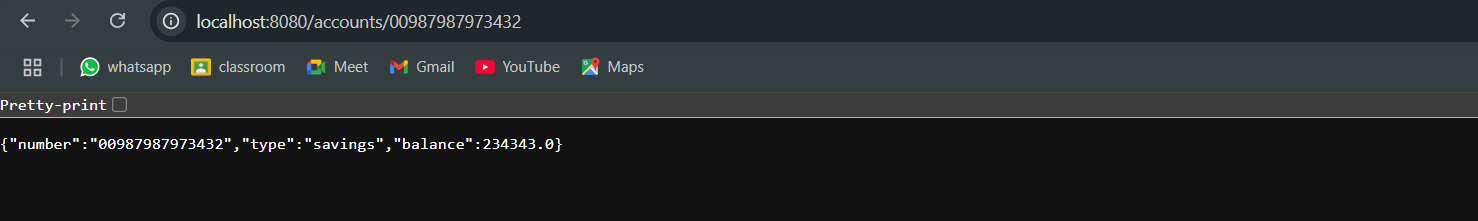
**Application.Properties:**

spring.application.name=account

**URL:**

[**http://localhost:8080/accounts/00987987973432**](http://localhost:8080/accounts/00987987973432)

**Output:**





* **Loans:**

1. Follow similar steps specified for Account Microservice and implement a service API to get loan account details
2. Method: GET o Endpoint: /loans/{number}
3. Sample Response. Just a dummy response without any backend connectivity.

**{ number: "H00987987972342", type: "car", loan: 400000, emi: 3258, tenure: 18 }**

1. Launching this application by having account service already running
2. This launch will fail with error that the bind address is already in use
3. The reason is that each one of the service is launched with default port number as 8080. Account service is already using this port and it is not available for loan service.
4. Include "server.port" property with value 8082 and try launching the application
5. Test the service with 8082 port Now we have two microservices running on different ports
6. NOTE: The console window of Eclipse will have both the service console running. To switch between different consoles use the monitor icon within the console view.

**Loan.Controller:**

package com.cognizant.loan.controller;import org.springframework.web.bind.annotation.\*;

*@RestController*

*@RequestMapping*("/loans")

public class LoanController {

*@GetMapping*("/{number}")

public Loan getLoanDetails(*@PathVariable* String number) {

return new Loan(number, "car", 400000, 3258, 18);

}

static class Loan {

public String number;

public String type;

public double loan;

public double emi;

public int tenure;

public Loan(String number, String type, double loan, double emi, int tenure)

{

this.number = number;

this.type = type;

this.loan = loan;

this.emi = emi;

this.tenure = tenure;

}

}

}

**LoanApplicationTests.java:**

package com.cognizant.loan;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

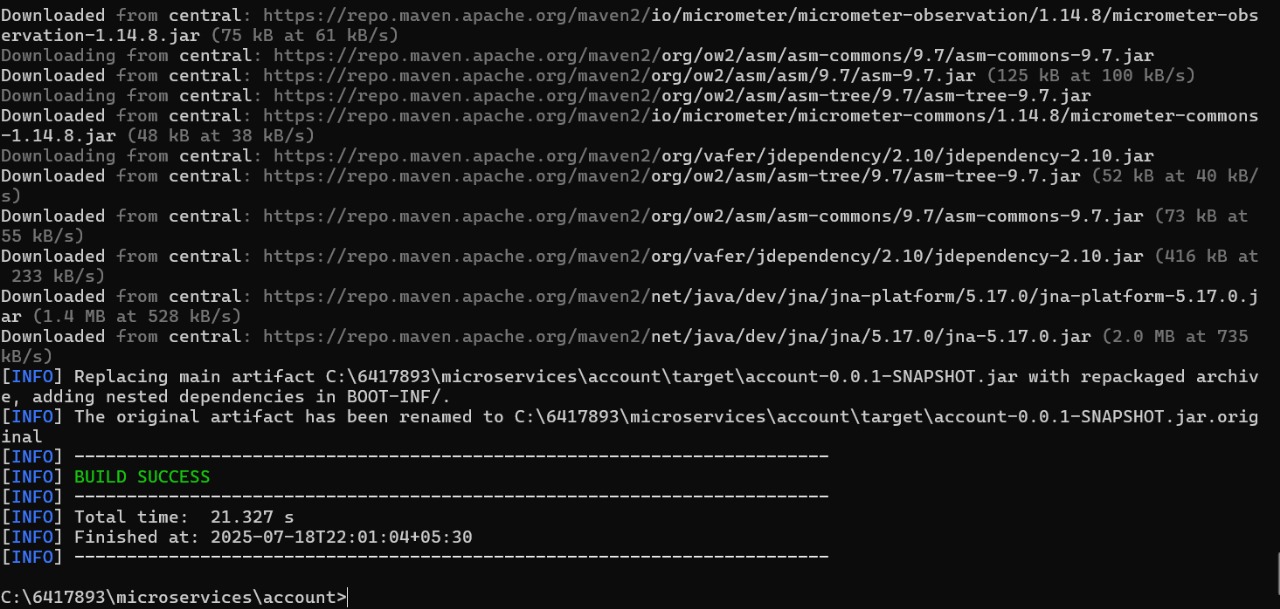
*@SpringBootTest*

class LoanApplicationTests {

*@Test*

void contextLoads() {

}

}

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>loan</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>loan</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

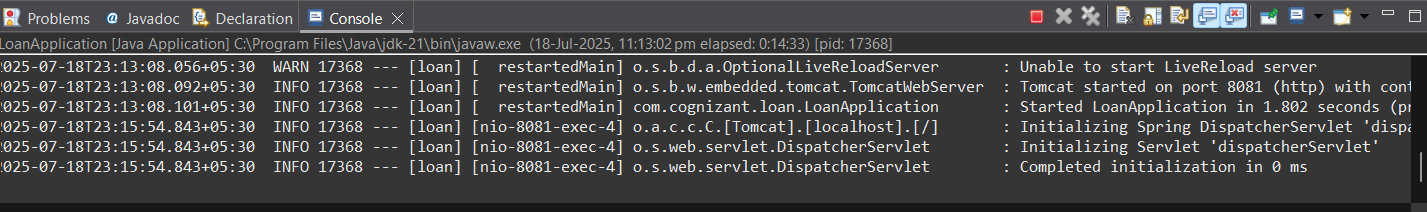
**Appilcation.Properties:**

spring.application.name=loan

server.port=8081

**URL:**

[**http://localhost:8081/loans/H00987987972342**](http://localhost:8081/loans/H00987987972342)

**Output:**



