**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 } •

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

**Loan Microservice**

• Follow similar steps specified for Account Microservice and implement a

service API to get loan account details

Method: GET

Endpoint: /loans/{number}

Sample Response. Just a dummy response without any backend

connectivity.

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

18 }

• Launching this application by having account service already running

• This launch will fail with error that the bind address is already in use

• 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.

• Include "server.port" property with value 8081 and try launching the

application

• Test the service with 8081 port

Now we have two microservices running on different ports.

**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

**LoanApplication.java**

package com.cognizant.loan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LoanApplication {

public static void main(String[] args) {

SpringApplication.run(LoanApplication.class, args);

}

}

**LoanController.java**

package com.cognizant.loan.controller;

import com.cognizant.loan.model.Loan;

import com.cognizant.loan.service.LoanService;

import org.springframework.beans.factory.annotation.Autowired;

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

@RestController

@RequestMapping("/loan")

public class LoanController {

@Autowired

private LoanService loanService;

@GetMapping("/status")

public String status() {

return "Loan service is running...";

}

@GetMapping("/details")

public Loan getLoanDetails() {

return loanService.getLoanDetails();

}

}

**Loan.java**

package com.cognizant.loan.model;

public class Loan {

private String loanNumber;

private String type;

private double amount;

public Loan(String loanNumber, String type, double amount) {

this.loanNumber = loanNumber;

this.type = type;

this.amount = amount;

}

public String getLoanNumber() { return loanNumber; }

public void setLoanNumber(String loanNumber) { this.loanNumber = loanNumber; }

public String getType() { return type; }

public void setType(String type) { this.type = type; }

public double getAmount() { return amount; }

public void setAmount(double amount) { this.amount = amount; }

}

**LoanService.java**

package com.cognizant.loan.service;

import com.cognizant.loan.model.Loan;

import org.springframework.stereotype.Service;

@Service

public class LoanService {

public Loan getLoanDetails() {

return new Loan("LN123456", "Home Loan", 750000.00);

}

}

**application.properties**

server.port=8081

**pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>loan</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>loan</name>

<description>Loan Microservice</description>

<parent>

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

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

<version>3.2.5</version>

<relativePath/>

</parent>

<properties>

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

</properties>

<dependencies>

<!-- Web Starter -->

<dependency>

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

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

</dependency>

<!-- Test (optional) -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot plugin -->

<plugin>

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

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

<version>3.2.5</version>

</plugin>

</plugins>

</build>

</project>

**OUTPUT:**

**Account Microservice**

A screenshot of a computer

AI-generated content may be incorrect.

**Loan Microservice**

**A screen shot of a computer

AI-generated content may be incorrect.**