**DN 4.0 WEEK 5 MANDATORY PROBLEMS**

**MODULE 9: 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.**

In a monolithic application will all the methods it will become a hectic task for us to maintain that codebase and make changes it that instead this concept of Microservices will give way to new method where we will break the whole codebase into smaller code blocks just like divide and conquer by this we can assign a port to different microservices and maintain easily make changes to the code whenever needed.

The main advantage here is independence where if one microservice stops working then other microservice will not stop working.

**PROGRAM:**

First let’s see the accounts microservice

This is the directory structure I used in replit



**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.account</groupId>

<artifactId>account</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<properties>

<java.version>1.8</java.version>

<spring.boot.version>2.7.18</spring.boot.version>

</properties>

<dependencies>

<dependency>

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

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

<version>${spring.boot.version}</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

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

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

<version>${spring.boot.version}</version>

</plugin>

</plugins>

</build>

</project>

**AccountApplication.java**

package com.cognizant.account;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AccountApplication {

public static void main(String[] args) {

SpringApplication.run(AccountApplication.class, args);

}

}

**AccountController.java**

package com.cognizant.account.controller;

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

import java.util.Map;

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

public Map<String, Object> getAccountDetails(@PathVariable String number) {

return Map.of(

"number", number,

"type", "savings",

"balance", 234343

);

}

}

Now in the shell if we just give this command

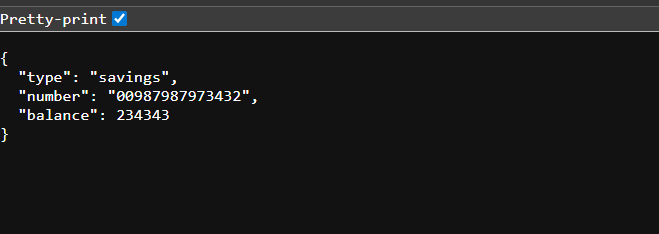
mvn clean spring-boot:run

we get output like this:



In the web we get output like this we add this accounts and account number in the end of the url accounts/00987987973432

Output will beL



**Now lets see the Loan MicroService:**

**We have to follow the same steps as before**

**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.loan</groupId>

<artifactId>loan</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<properties>

<java.version>1.8</java.version>

<spring.boot.version>2.7.18</spring.boot.version>

</properties>

<dependencies>

<dependency>

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

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

<version>${spring.boot.version}</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

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

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

<version>${spring.boot.version}</version>

</plugin>

</plugins>

</build>

</project>

**LoanController.java**

package com.cognizant.loan.controller;

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

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

public Map<String, Object> getLoanDetails(@PathVariable String number) {

Map<String, Object> loan = new HashMap<>();

loan.put("number", number);

loan.put("type", "car");

loan.put("loan", 400000);

loan.put("emi", 3258);

loan.put("tenure", 18);

return loan;

}

}

**LoanApplication.java**

package com.cognizant.loan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LoanApplication {

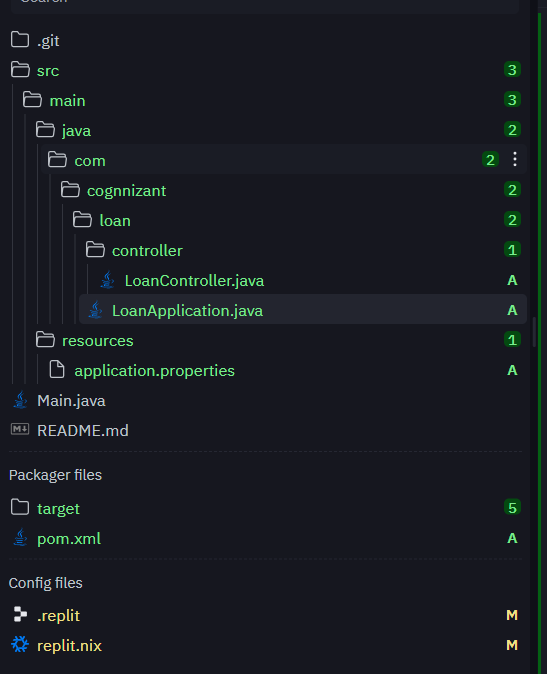
public static void main(String[] args) {

SpringApplication.run(LoanApplication.class, args);

}

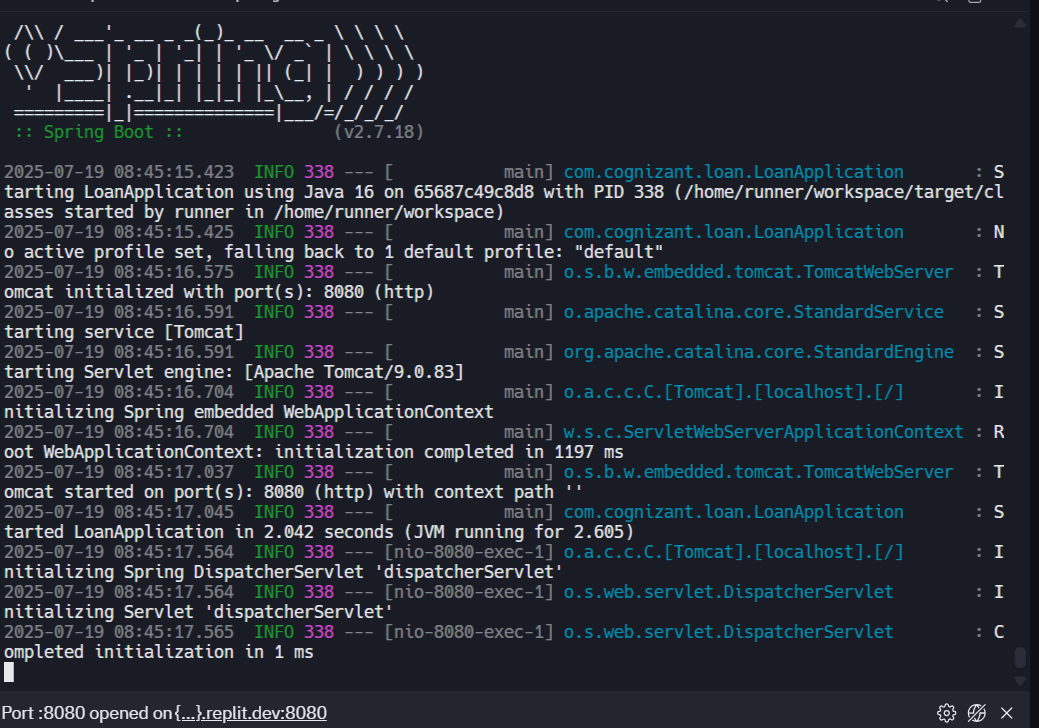
}

The directory Structure is



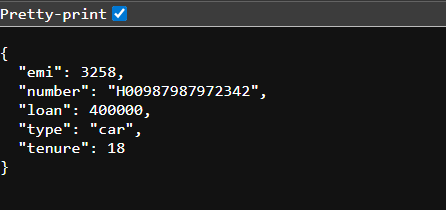
After giving the same command in the shell that we gave for accounts microservice

We get like this



Now add the path in the link and search that in the browser /loans/H00987987972342

We will get the output like this

****

**Here now we just divided the whole bank and loan application into microservices because by doing this we get smaller codebases and by this the testing part will be easy and if any error occurs then it will be easy to identify and debug that error in short period of time and also the different teams can work on different microservices and develop the final product**