Microservices Assignment – Account and Loan Services using Spring Boot

This document provides the detailed procedure for creating two independent Spring Boot microservices – one for handling account details and another for loan details. Each service is a separate Maven project without any backend connectivity.

# Step 1: Setup Directory Structure

1. Create a folder with your Employee ID in the D: drive (e.g. C:\Users\likhi\OneDrive\Desktop\CTS DN\Week5).  
2. Inside that folder, create another folder named 'microservices' (e.g. C:\Users\likhi\OneDrive\Desktop\CTS DN\Week5\microservices, ).

# Step 2: Create Account Microservice

1. Open https://start.spring.io in your browser.  
2. Fill the form:  
 - Group: com.cognizant  
 - Artifact: account  
3. Add Dependencies:  
 - Spring Boot DevTools  
 - Spring Web  
4. Click Generate to download the zip file.  
5. Extract the 'account' folder and move it to the 'microservices' folder.  
6. Open Command Prompt in the 'account' folder and run:  
 mvn clean package  
7. Import the project into Eclipse as an existing Maven project.

# Step 3: Implement Account Controller

Create a class AccountController.java inside the package com.cognizant.account:

@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;  
 }  
 }  
}

Run the application and test in browser: http://localhost:8080/accounts/00987987973432

# Step 4: Create Loan Microservice

Repeat the same steps as in the account microservice but with the following differences:  
1. Artifact: loan  
2. Move it to 'microservices' folder  
3. Use a different controller with the following details:

@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 int loan;  
 public int emi;  
 public int tenure;  
  
 public Loan(String number, String type, int loan, int emi, int tenure) {  
 this.number = number;  
 this.type = type;  
 this.loan = loan;  
 this.emi = emi;  
 this.tenure = tenure;  
 }  
 }  
}

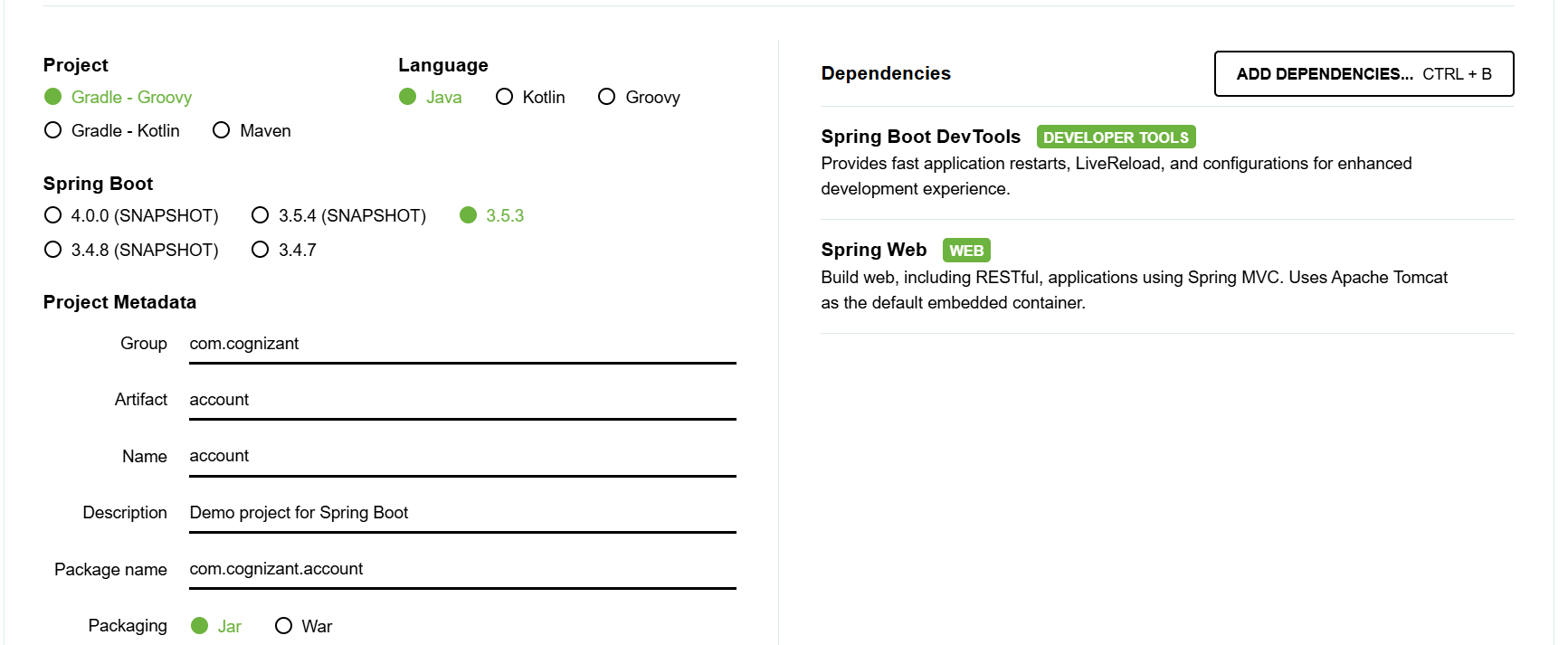
# Step 5: Set Different Port for Loan Microservice

Open src/main/resources/application.properties in the loan project and add:  
 server.port=8081  
This ensures that the loan service runs on port 8081 while account service runs on 8080.

# Step 6: Run and Test Both Services

1. Run AccountApplication.java and access:  
 http://localhost:8080/accounts/00987987973432  
2. Run LoanApplication.java and access:  
 http://localhost:8081/loans/H00987987972342  
In Eclipse, use the monitor icon to switch between both running consoles.

## Spring Initializr - Account Setup



Filling form to generate the Account microservice

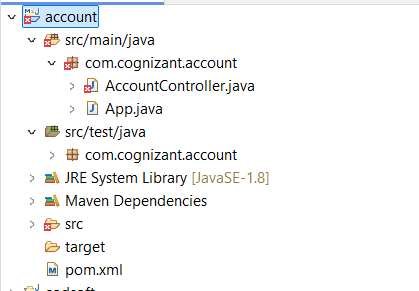
## Command Prompt - mvn -v

Verifying Maven installation

## Command Prompt - mvn package

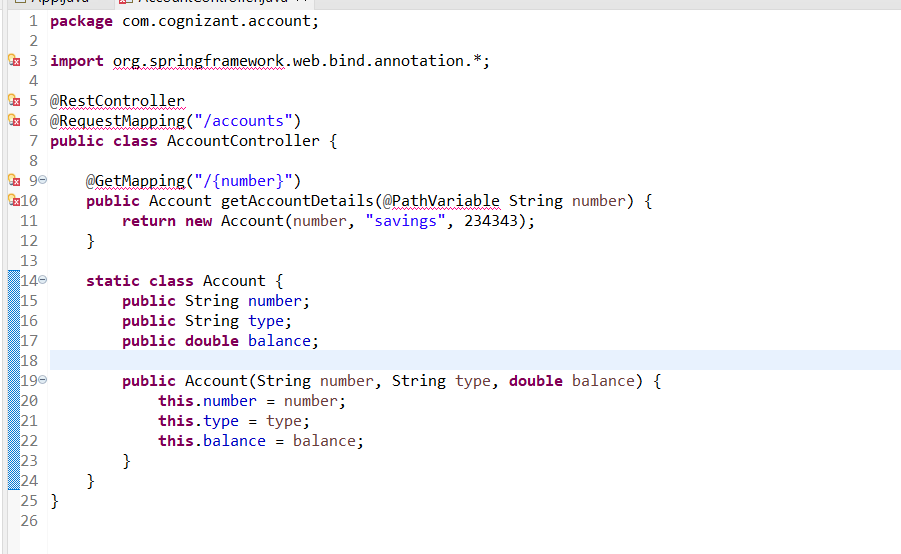
Running mvn clean package in the account project

## Eclipse - Import Project



Importing the Maven project into Eclipse

## AccountController.java in Eclipse

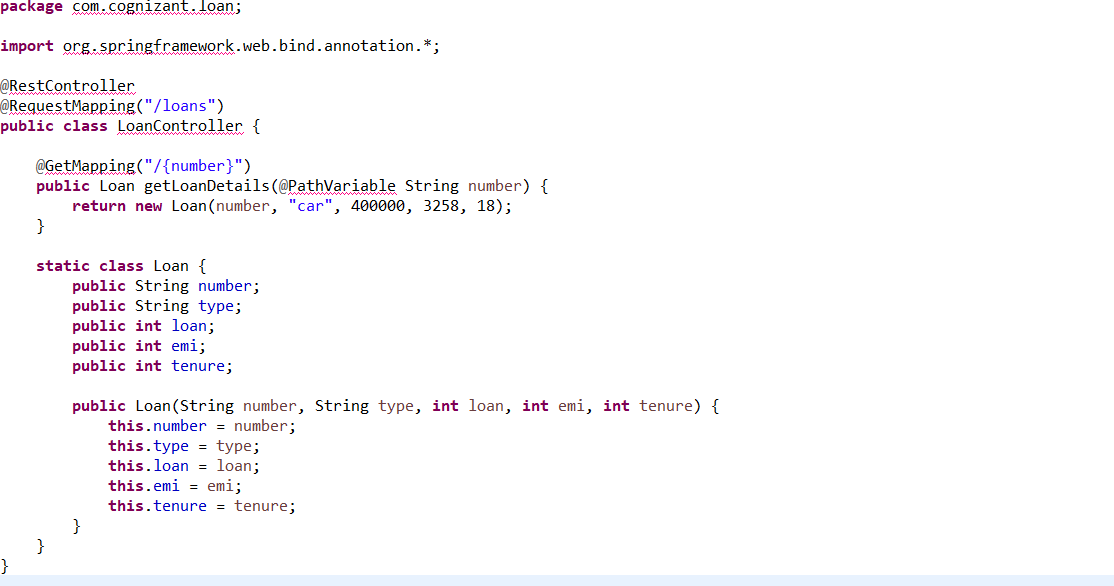


Implemented AccountController class in Eclipse

## Account API in Browser

Accessed account microservice via browser

## LoanController.java in Eclipse



Implemented LoanController class in Eclipse

## Loan Port Config

Configured port 8081 for loan microservice

## Loan API in Browser

Accessed loan microservice via browser

## Switch Console in Eclipse

Switching between service consoles in Eclipse