**SKILL LEARNT: MICROSERVICES WITH SPRING BOOT 3 AND SPRING CLOUD (WEEK 5)**

**EXERCISE 1: CREATING MICROSERVICES FOR ACCOUNT AND LOAN**

STEP 1: Firstly, we created folder named ‘16012004’ in our D drive.

STEP 2: Next we created folder named ‘microservices’ inside the folder named ‘16012004’.

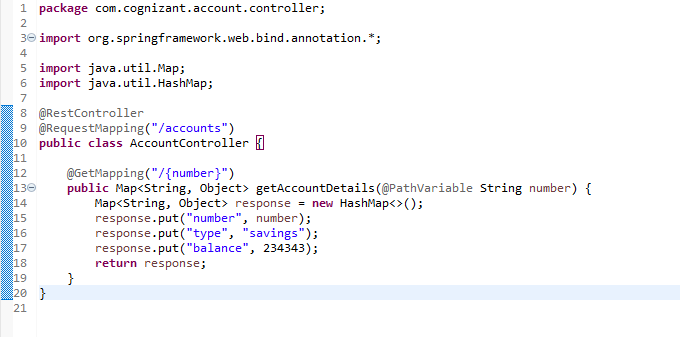
STEP 3: Then, we went to Spring Initializr to create a Spring Boot Project setting Group as com.cognizant and Artifact as account. We also set two dependencies one is Spring Boot DevTools and another is Spring Web. Then we generated and downloaded the zip file.

STEP 4: Then we unzipped the account folder and placed it under the microservices folder created by us.

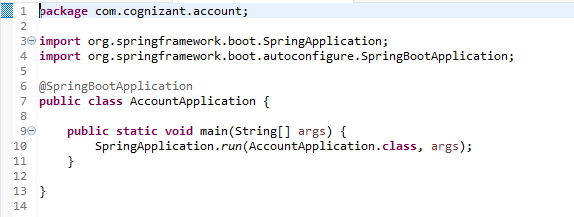
STEP 5: Then we navigated to the folder location using command prompt and build using mvn clean package command.

STEP 6: Then we imported the project in the Eclipse IDE.

STEP 7: Under com.cognizant.account, I created a new package named com.cognizant.account.controller under which I created a new class named AccountController. The code for AccountController is shown below



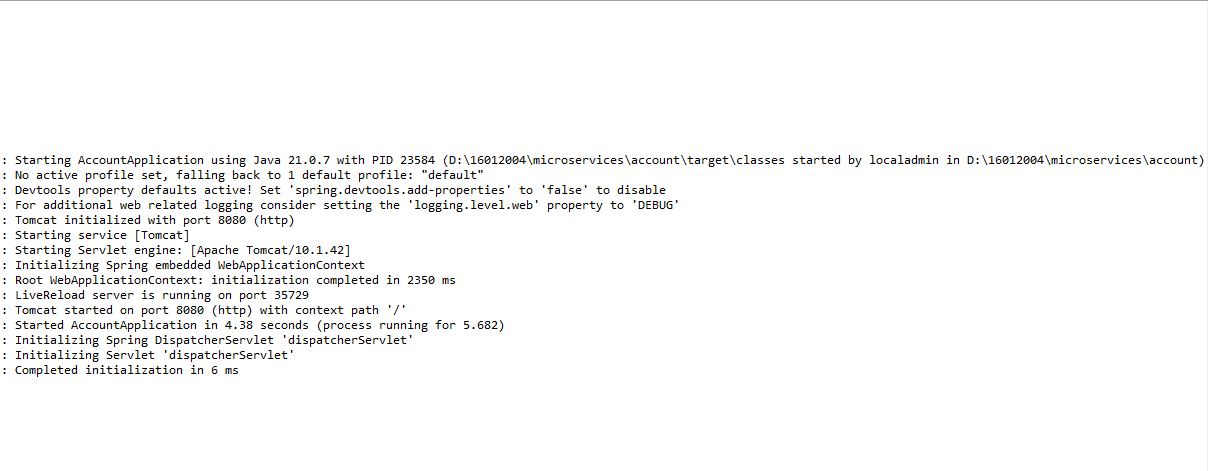
STEP 8: Under package com.cognizant.account lies the main Java class named AccountApplication. The code for it is shown below



STEP 9: Running the main java class produces the output



Continued….



STEP 10: When we open the browser and test it, it gives us something like this



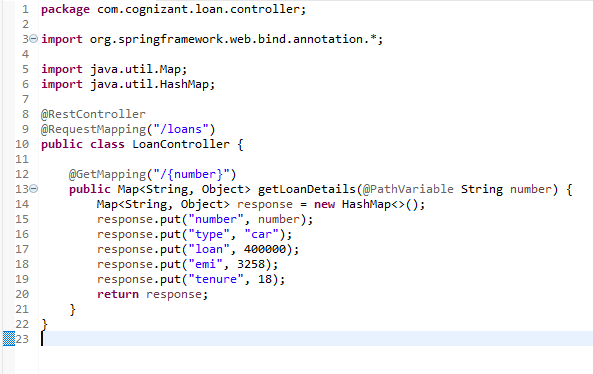
STEP 11: Again, we went to Spring Initializr to create a Spring Boot Project setting Group as com.cognizant and Artifact as loan. We also set two dependencies one is Spring Boot DevTools and another is Spring Web. Then we generated and downloaded the zip file.

STEP 12: Then we unzipped the account folder and placed it under the microservices folder created by us.

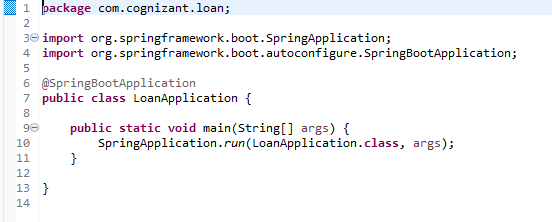
STEP 13: Then we navigated to the folder location using command prompt and build using mvn clean package command.

STEP 14: Then we imported the project in the Eclipse IDE.

STEP 15: Under com.cognizant.loan, I created a new package named com.cognizant.loan.controller under which I created a new class named LoanController. The code for LoanController is shown below



STEP 16: Under package com.cognizant.loan lies the main Java class named LoanApplication. The code for it is shown below



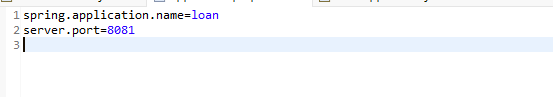
STEP 17: Running the main java class produces the output



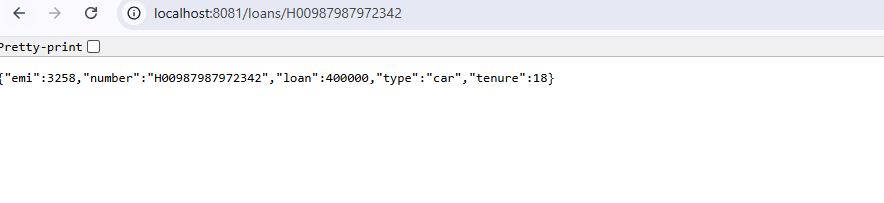
Continued…



STEP 18: Under src/main/resources, we have application.properties. We set the server port to 8081 for the application to run at that browser.



STEP 19: When we open the browser and test it, it gives us something like this



STEP 20: Thus, we have two different microservices running at two different ports, port 8080 and port 8081.

By doing this exercise, we understood how to develop and execute two standalone Spring Boot microservices—Account and Loan—each providing REST APIs with sample data. We created two individual Maven projects with Spring Initializr, arranged them in an organized folder structure, and developed them with the Maven Wrapper without any global installation. By setting various server ports (8080 for Account and 8081 for Loan), we did not face any conflicts and were able to run the two services simultaneously. This practical learning provided us with a basic understanding of microservices architecture, RESTful API implementation, port handling, and isolated service deployment by using Spring Boot.