**Cognizant Digital Nurture 4.0**

***WEEK - 5 Module 8 – Microservices with Spring Boot 3 and Spring Cloud***

**Creating Microservices for Account and Loan (MANDATORY)**

In this hands-on exercise, I created two independent microservices using Spring Boot: one for handling **accounts** and another for **loans**. Each service was implemented as a separate Maven project with its own pom.xml. These services return **dummy data** without any backend/database connectivity.

**Step 1: Setting Up the Folder Structure**

1. First, I created a folder named with my **employee ID** in the H: drive.  
   Example: H:\ 6439983
2. Inside that folder, I created another folder named **microservices**.  
   Final path: H:\ 6439983\microservices

**Step 2: Creating the Account Microservice**

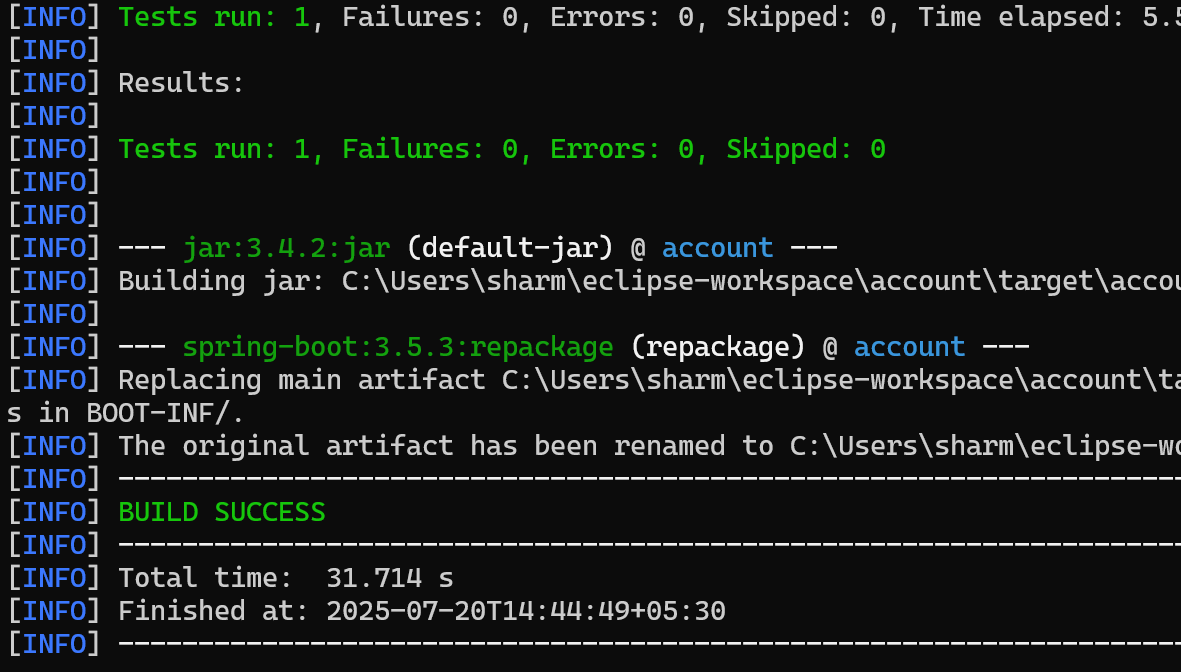
**a. Project Initialization**

1. I opened <https://start.spring.io/>.
2. I filled in the following:
   * **Group**: com.cognizant
   * **Artifact**: account
3. I selected these dependencies:
   * **Spring Boot DevTools** (under Developer Tools)
   * **Spring Web** (under Web)
4. After that, I clicked **Generate** and downloaded the ZIP file.

**b. Extracting and Building the Project**

1. I extracted the ZIP file and placed the account folder inside the microservices directory.
2. I opened **Command Prompt** inside the account folder and ran:

mvn clean package



**c. Importing and Coding in Eclipse**

1. I imported the account project into **Eclipse**.
2. Then, I created a **REST controller** to handle account details.

**Here’s what I implemented:**

* **Method**: GET
* **Endpoint**: /accounts/{number}
* **Sample Response**:

{

"number": "00987987973432",

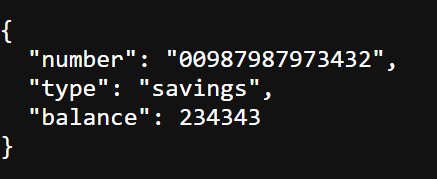
"type": "savings",

"balance": 234343

}



1. I ran the application and tested it in my browser:  
   <http://localhost:8080/accounts/00987987973432>



**Step 3: Creating the Loan Microservice**

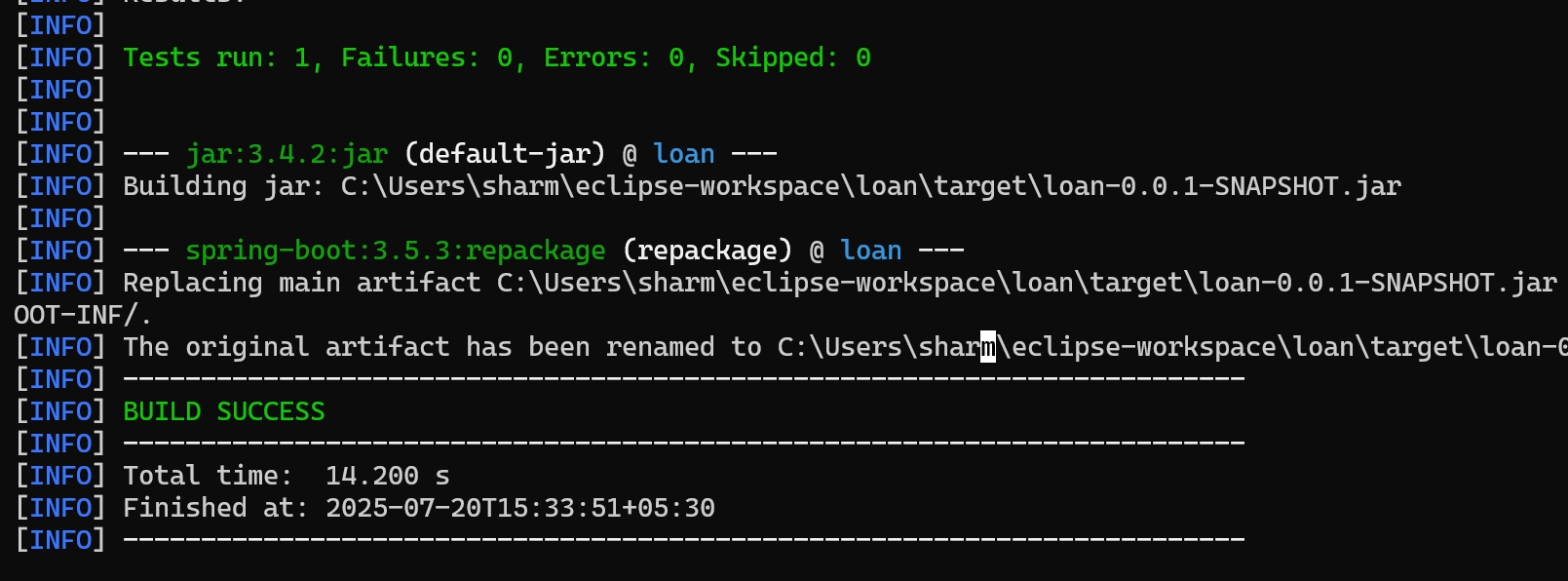
**a. Project Setup**

1. Again, I visited [start.spring.io](https://start.spring.io/) and filled in:
   * **Group**: com.cognizant
   * **Artifact**: loan
2. I added the same dependencies as I did for the account project.
3. I downloaded and extracted the project, placing it inside the microservices folder.

**b. Building and Importing**

1. In Command Prompt, I navigated to the loan folder and ran:

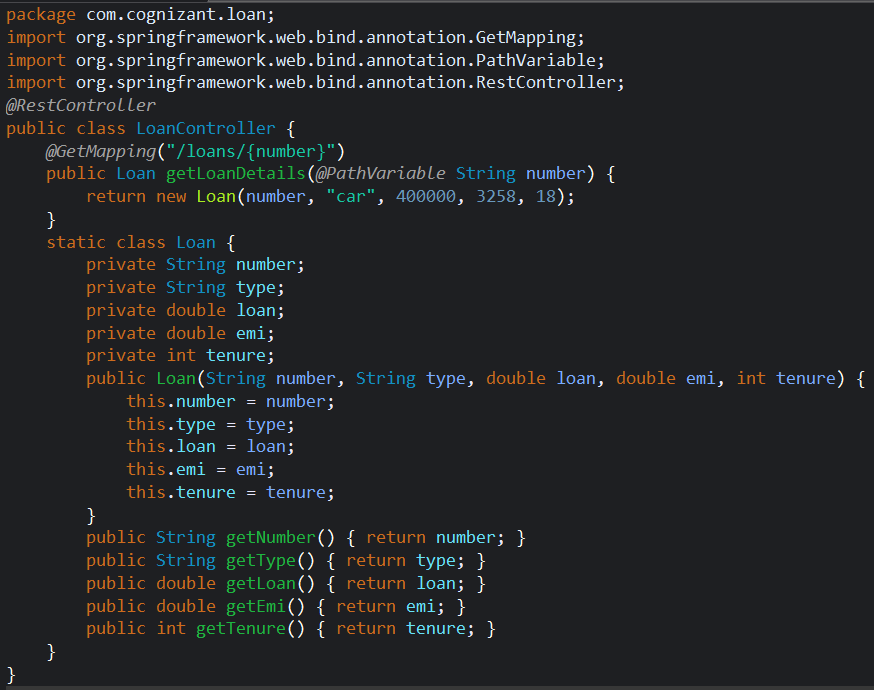
mvn clean package



1. I imported the loan project into Eclipse.

**c. Implementing the Loan API**

I created a REST controller to return loan account details.



**Details of the endpoint I created:**

* **Method**: GET
* **Endpoint**: /loans/{number}
* **Sample Response**:

{

"number": "H00987987972342",

"type": "car",

"loan": 400000,

"emi": 3258,

"tenure": 18

}

**d. Handling Port Conflict**

When I tried to run the Loan Microservice while the Account Microservice was still running, I got a **port conflict error** (default port 8080 already in use).

To solve this:

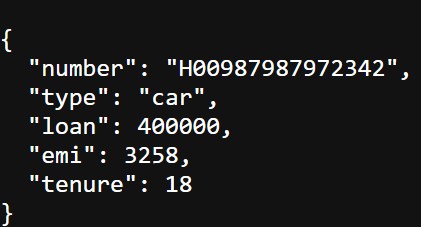
1. I opened the application.properties file in the Loan Microservice.
2. I added the following line:

ini

server.port=8081



After this change, I re-ran the Loan Microservice and tested it successfully at:  
<http://localhost:8081/loans/H00987987972342>



**Final Setup**

At this point, I had **two microservices running independently**:

* **Account Microservice** on port 8080
* **Loan Microservice** on port 8081

**ADDITIONAL IMPORTANT HANDS-ON**

**Create Eureka Discovery Server and register microservices**

Eureka Discovery Server holds a registry of all the services that are available for immediate consumption. Anybody whom wants to consume a RESTful Web Service can come to the discovery server and find out what is available and ready for consumption. Eureka Discovery Server is part of spring cloud module.

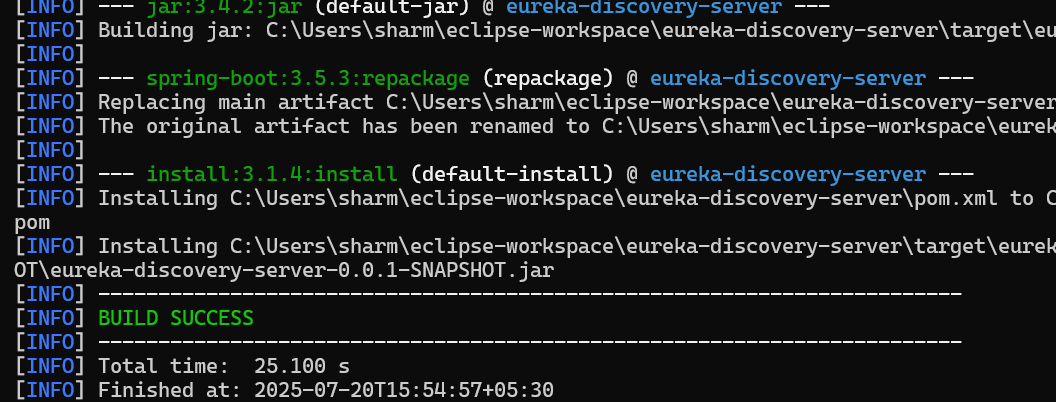
**Steps Followed:**

1. I went to <https://start.spring.io> and generated a new Spring Boot project with the following configuration:

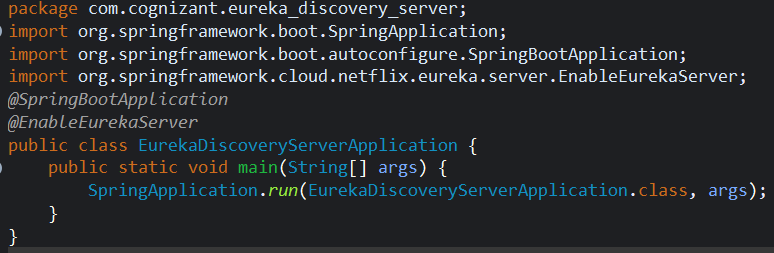
* **Group**: com.cognizant
* **Artifact**: eureka-discovery-server
* **Dependencies**: Spring Cloud Discovery > Eureka Server

1. I downloaded the generated project and used the terminal to build it with Maven:

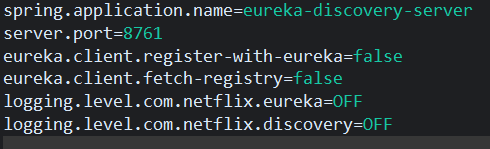
mvn clean install



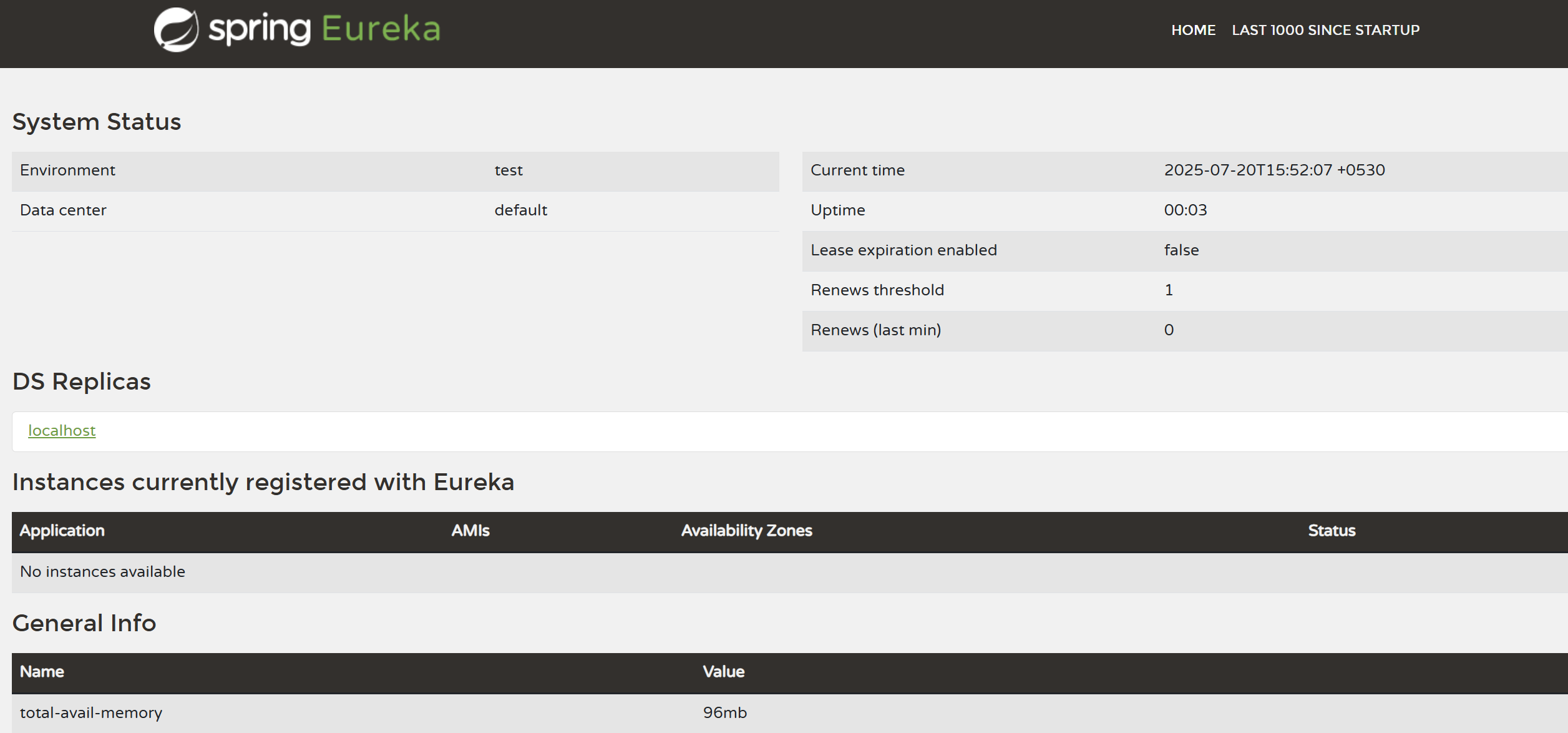
1. I imported the project into Eclipse as a Maven project.
2. In the main class EurekaDiscoveryServerApplication, I added the @EnableEurekaServer annotation to enable the Eureka Server:



1. I updated the application.properties file with the following configuration:



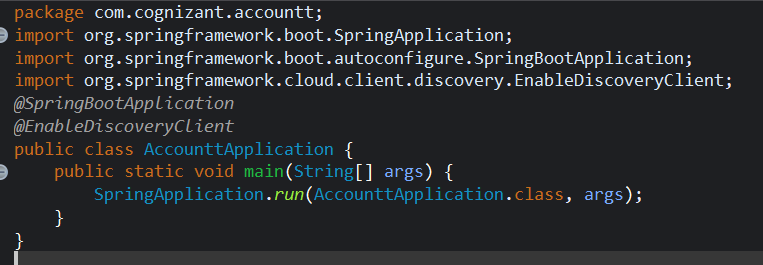
1. I ran the application and opened my browser to http://localhost:8761. The Eureka Dashboard opened with an empty registry.



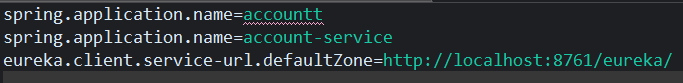
1. I went back to <https://start.spring.io> and generated another project:

* **Group**: com.cognizant
* **Artifact**: account
* **Dependencies**: Spring Web, Eureka Discovery Client, Spring Boot DevTools

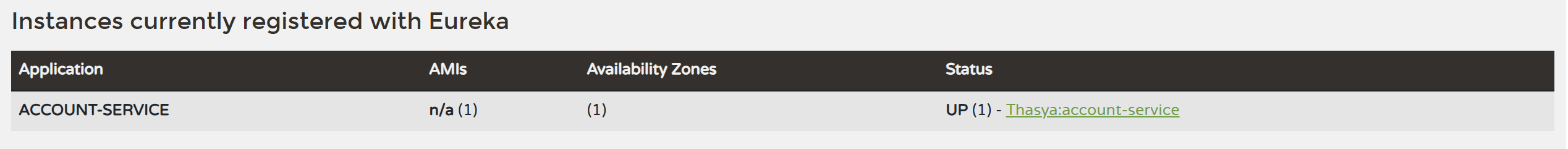
1. I downloaded the project and replaced its pom.xml with the one from the online preview (Explore > Copy).
2. I built the project using Maven:
3. I added the @EnableDiscoveryClient annotation in the main class:



1. I updated the application.properties:



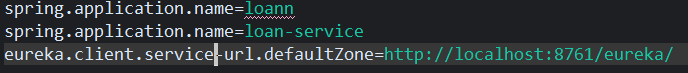
1. I stopped all running applications from Eclipse, then started eureka-discovery-server first.
2. Once the Eureka Dashboard was live (still empty), I started the account service.
3. After refreshing the Eureka Dashboard, I could now see account-service listed in the registry!



1. I went back to <https://start.spring.io> and generated another project:

* **Group**: com.cognizant
* **Artifact**: loan
* **Dependencies**: Spring Web, Eureka Discovery Client, Spring Boot DevTools

1. I downloaded the project and replaced its pom.xml with the one from the online preview (Explore > Copy).
2. I built the project using Maven
3. I added the @EnableDiscoveryClient annotation in the main class:
4. I updated the application.properties



1. Refreshed the Eureka Dashboard — now both account-service and loan-service were listed!
2. Repeat the **same steps** as Account:
3. **Artifact:** loan
4. **Dependencies:** Same
5. **Main Class:** Annotate with @EnableDiscoveryClient
6. **application.properties:**

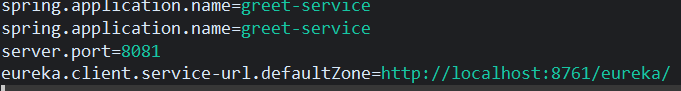
spring.application.name=loan-service

eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

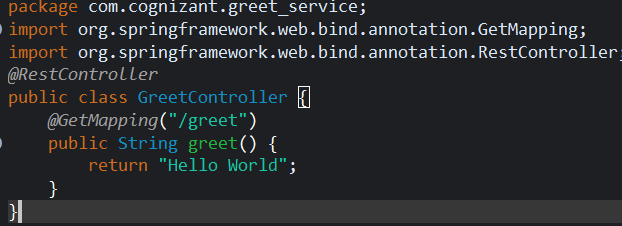
1. **Start after Eureka Server**
   * Both account-service and loan-service should appear in dashboard.

**Create greet-service Microservice**

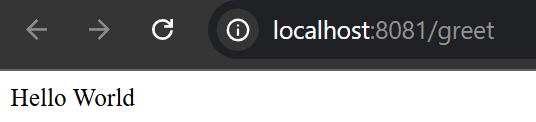
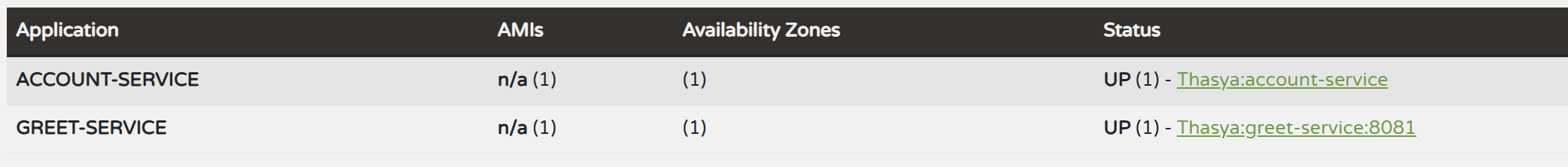
1. **Generate Project**
   * Artifact: greet-service
   * Dependencies: Spring Web, Eureka Discovery Client
2. **Configure application.properties**



1. **Add REST Controller**



1. **Add @EnableDiscoveryClient** to main class.
2. **Run the Service**
   * Test URL: <http://localhost:8081/greet>
   * Eureka Dashboard will show greet-service



 **Generate Project**

* Artifact: api-gateway
* Dependencies: Spring Cloud Gateway, Eureka Discovery Client, Spring Boot DevTools

 **Configure application.properties**

 **Run the API Gateway**

