**Digital Nurture 4.0**

**Week 5- Microservices with Spring Boot 3 and Spring Cloud**

# **Mandatory HandsOn**

# **File name: 2. Microservices with API gateway**

**1).Creating Microservices for account and loan**

**Objective:**

To create two independent Spring Boot microservices — one for handling Account details and another for Loan details — using Spring Web and DevTools dependencies. Each service should:

* Have its own endpoint (/accounts/{number}, /loans/{number})
* Run on different ports (8080, 8081)
* Return static/dummy JSON responses
* Be implemented and tested inside IntelliJ IDEA

**ACCOUNT MICROSERVICE**

**Step 1: Spring Initializr Setup**

1. Open the browser and go to: <https://start.spring.io> fill the details:

|  |  |
| --- | --- |
| Field | Value |
| Project | Maven |
| Language | Java |
| Group | com.cognizant |
| Artifact | account |
| Name | account |
| Package | com.cognizant.account |
| Java | 17 (or latest) |

1. Click "Add Dependencies" and select:

* Spring Web
* Spring Boot DevTools

1. Click Generate → ZIP file will download

**Step 2: Extract and Open in IntelliJ**

1. Extract the downloaded zip → Folder named account will be there
2. Open IntelliJ IDEA
3. File > Open → Select the account folder → Click OK
4. IntelliJ will import it as a Maven Project

**Step 3: Modify application.properties (if needed)**

* Open this file:  
  src/main/resources/application.properties

**application.properties**

server.port=8080

**Step 4: Create Controller**

1. Right-click on src/main/java/com/cognizant/account
2. Create new package: controller
3. Inside that, create new Java class: AccountController

**AccountController.java**

package com.cognizant.account.controller;

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

import java.util.\*;

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

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

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

response.put("number", number);

response.put("type", "savings");

response.put("balance", 234343);

return response;

}

}

**Step 5: Run Account Application**

1. Go to AccountApplication.java
2. Right-click → Run 'AccountApplication'
3. Open browser → test URL:

<http://localhost:8080/accounts/00987987973432>

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**LOAN MICROSERVICE**

**Step 1: Spring Initializr Setup**

* 1. Open the browser and go to: <https://start.spring.io> fill the details:

|  |  |
| --- | --- |
| Field | Value |
| Project | Maven |
| Language | Java |
| Group | com.cognizant |
| Artifact | loan |
| Name | loan |
| Package | com.cognizant. loan |
| Java | * 1. or latest) |

* 1. Click "Add Dependencies" and select:
* Spring Web
* Spring Boot DevTools
  1. Click Generate → ZIP file will download

**Step 2: Extract and Open in IntelliJ**

Extract the downloaded zip → Folder named account will be there

Open IntelliJ IDEA

File > Open → Select the account folder → Click OK

IntelliJ will import it as a Maven Project

**Step 3: Modify application.properties**

* Open this file:  
  src/main/resources/application.properties

**application.properties**

server.port=8081

**Step 4: Create Controller**

1. Right-click on src/main/java/com/cognizant/ loan
2. Create new package: controller
3. Inside that, create new Java class: loanController

**loanController.java**

package com.cognizant.loan.controller;

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

import java.util.\*;

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

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

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

response.put("number", number);

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

response.put("loan", 400000);

response.put("emi", 3258);

response.put("tenure", 18);

return response;

}

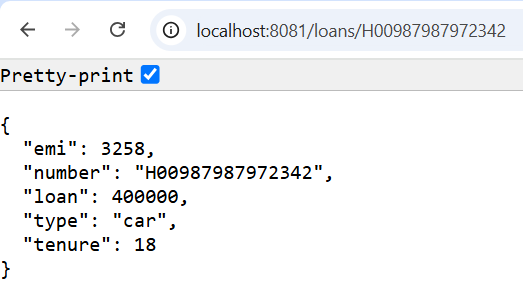
}

**Step 5: Run loan Application**

1. Go to LoanApplication.java
2. Right-click → Run 'LoanApplication'
3. Test in browser:

<http://localhost:8080/accounts/00987987973432>

**Output:**

****

# **Additional -Important HandsOn**

# **File name: 2. Microservices with API gateway**

**2).Create Eureka Discovery Server and register microservices**

**OBJECTIVE:**

To create and run 3 Spring Boot applications:

1. **Eureka Discovery Server** (Port: 8761)
2. **Account Microservice** registered with Eureka (Port: 8080)
3. **Loan Microservice** registered with Eureka (Port: 8081)

After setup, all services should be accessible and visible in the Eureka Dashboard.

**EUREKA DISCOVERY SERVER**

**STEP 1: Generate Project using Spring Initializr**

1. Go to: <https://start.spring.io>
2. Fill details:
   1. Group: com.cognizant
   2. Artifact: eureka-discovery-server
3. Select dependencies:
   1. Spring Boot DevTools
   2. Spring Cloud Discovery → Eureka Server
4. Click Generate
5. Extract the ZIP and open the folder in IntelliJ IDEA

**STEP 2: Modify application.properties**

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

**STEP 3: Add @EnableEurekaServer**

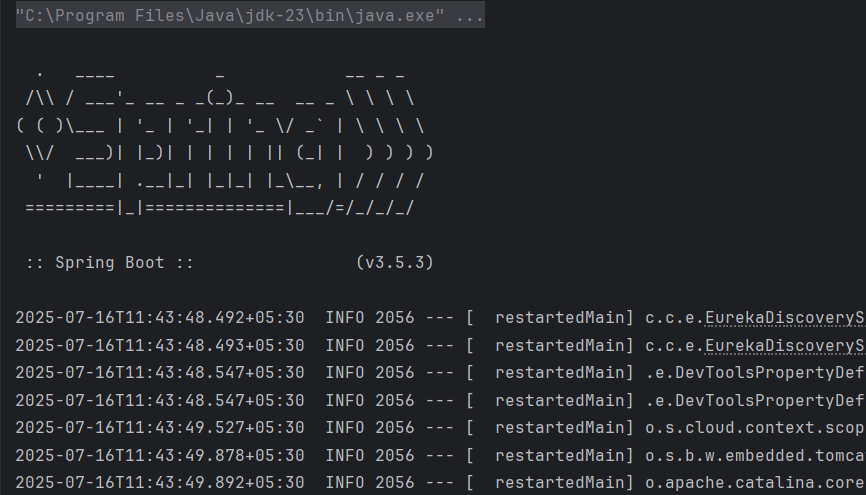
**EurekaDiscoveryServerApplication.java:**

package com.cognizant.eureka\_discovery\_server;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  
  
@SpringBootApplication  
@EnableEurekaServer  
public class EurekaDiscoveryServerApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(EurekaDiscoveryServerApplication.class, args);  
 }  
}

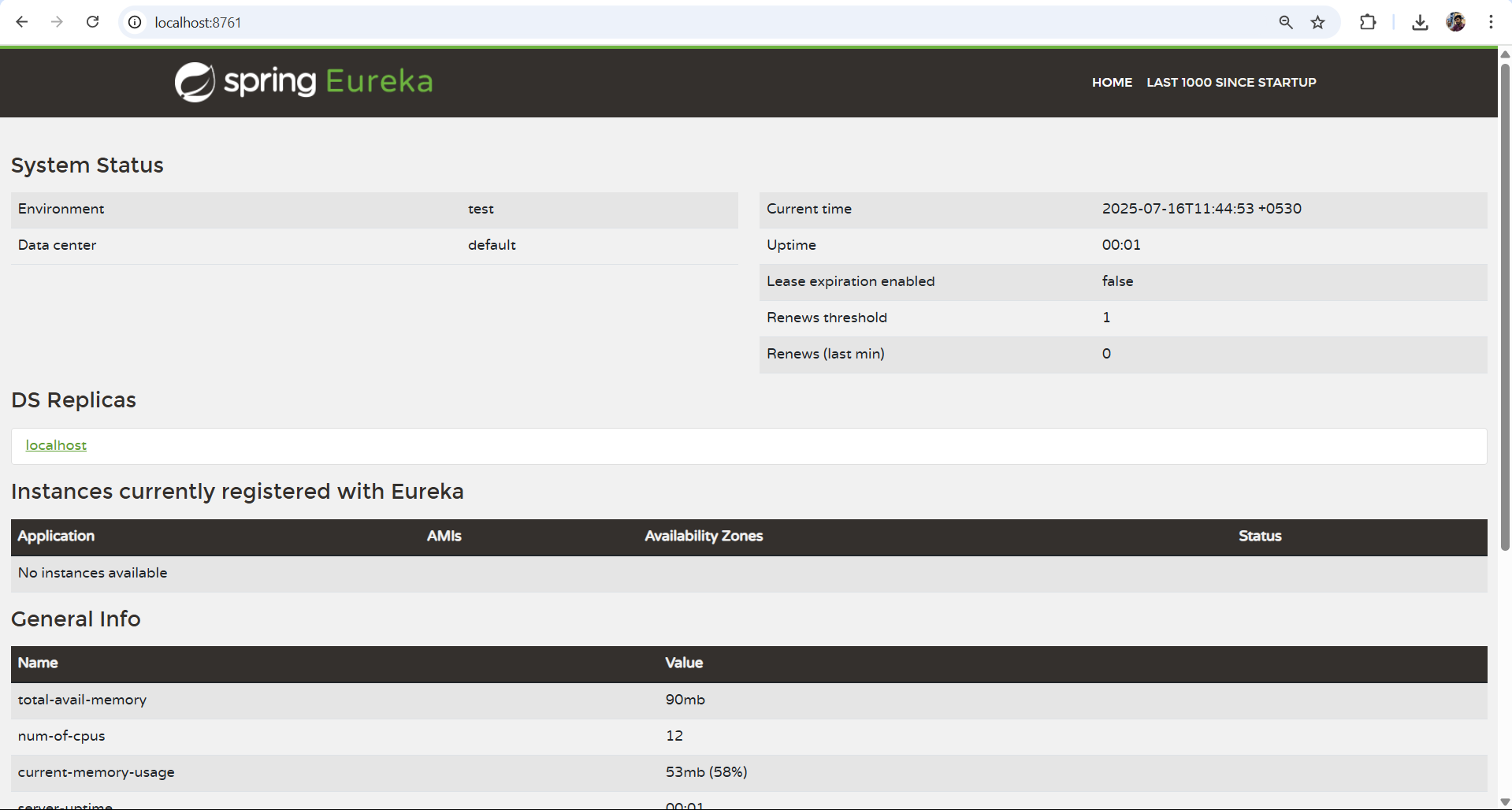
**STEP 4: Run & Test Eureka**

* Right-click project → Run
* Open in browser:  
   <http://localhost:8761>

**Output:**



**In Browser:**



**ACCOUNT MICROSERVICE**

**STEP 1: Create New Spring Boot Project**

* Go to: <https://start.spring.io>
* Fill details:
  + Group: com.cognizant
  + Artifact: account-service
* Dependencies:
  + Spring Web
  + Spring Boot DevTools
  + Spring Cloud Discovery → Eureka Discovery Client
* Generate, unzip, and open in IntelliJ

**STEP 2: application.properties**

server.port=8080

spring.application.name=account-service

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

**STEP 3: Add @EnableDiscoveryClient**

**AccountServiceApplication.java:**

package com.cognizant.account\_service;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
  
@SpringBootApplication  
@EnableDiscoveryClient  
public class AccountServiceApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(AccountServiceApplication.class, args);  
 }  
}

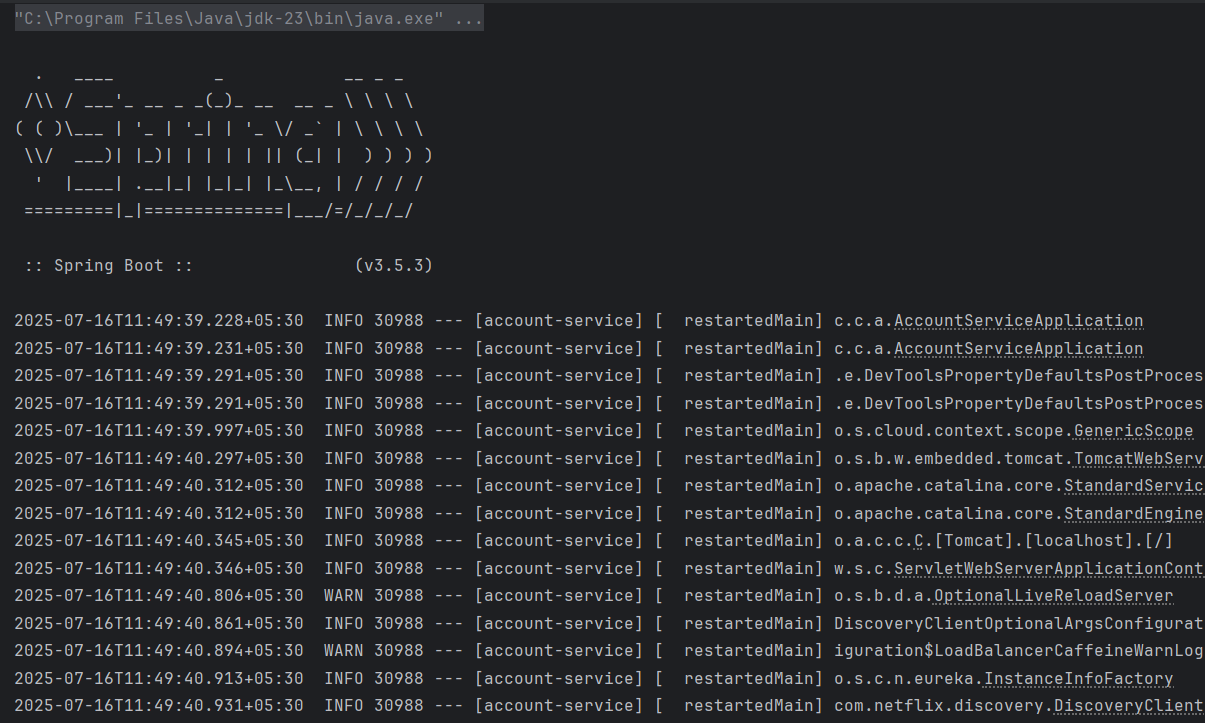
**STEP 4: Create Controller**

Create a file: AccountController.java under package com.cognizant.account\_service.controller;  
  
import org.springframework.web.bind.annotation.\*;  
import java.util.\*;  
  
@RestController  
@RequestMapping("/accounts")  
public class AccountController {  
 @GetMapping("/{number}")  
 public Map<String, Object> getAccount(@PathVariable String number) {  
 Map<String, Object> data = new HashMap<>();  
 data.put("number", number);  
 data.put("type", "savings");  
 data.put("balance", 234343);  
 return data;  
 }  
}

**STEP 5: Run & Test**

* Right-click → Run AccountServiceApplication.java
* Test in browser:  
  <http://localhost:8080/accounts/12345>
* Also refresh:  
  <http://localhost:8761> — To see ACCOUNT-SERVICE

**Output:**

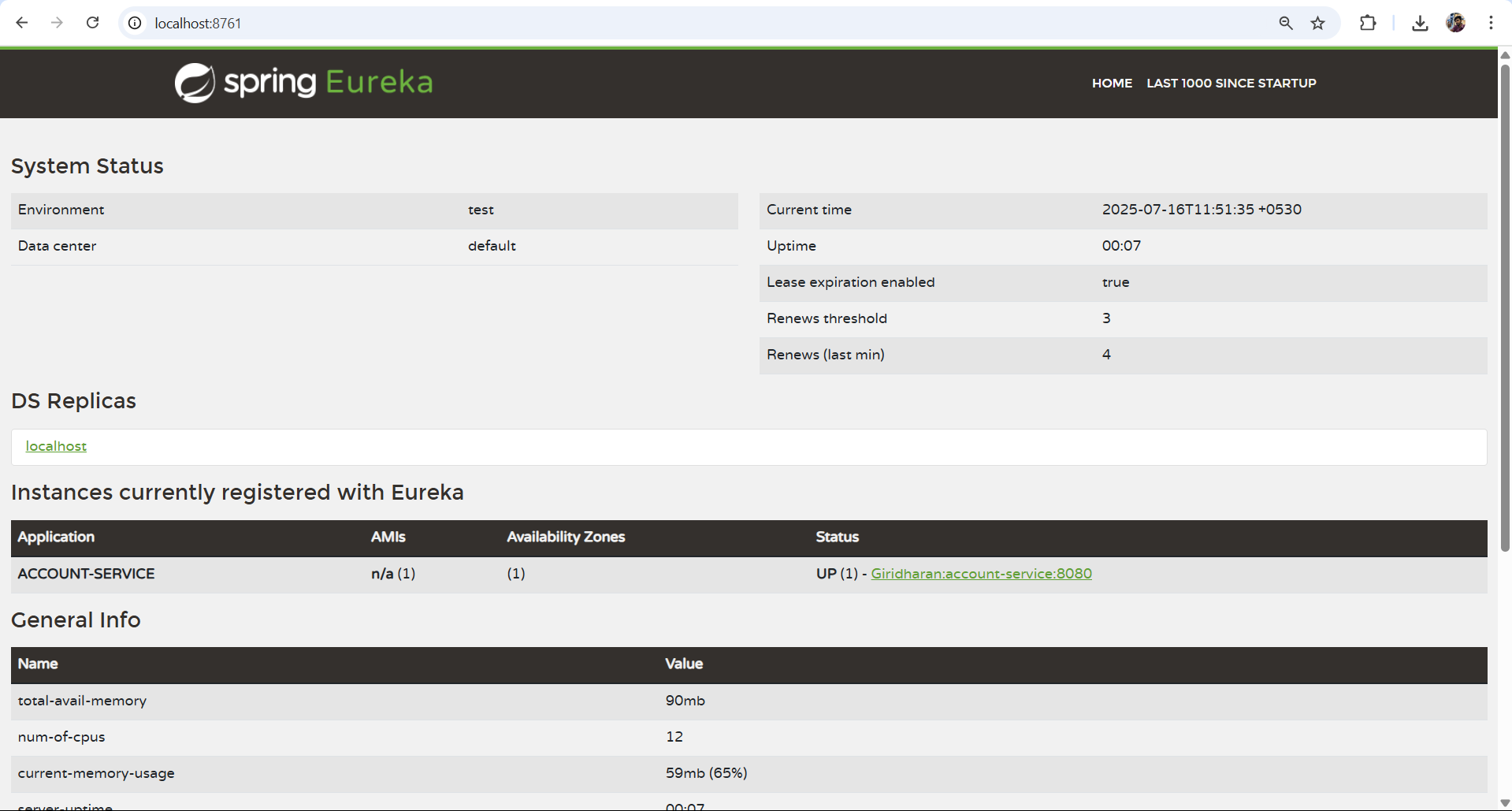
****

**In browser:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**In Eureka Server:**

****

**LOAN MICROSERVICE**

**STEP 1: Create New Spring Boot Project**

1. Go to: <https://start.spring.io>
2. Fill:

* Group: com.cognizant
* Artifact: loan-service

1. Dependencies:

* Spring Web
* Spring Boot DevTools
* Spring Cloud Discovery → Eureka Discovery Client

1. Generate and open in IntelliJ

**STEP 2: application.properties**

server.port=8081

spring.application.name=loan-service

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

**STEP 3: Add @EnableDiscoveryClient**

**LoanServiceApplication.java:**

package com.cognizant.loan\_service;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
  
@SpringBootApplication  
@EnableDiscoveryClient  
public class LoanServiceApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(LoanServiceApplication.class, args);  
 }  
}

**STEP 4: Create Controller**

Create a file LoanController.java under com.cognizant.loanservice.controller:

package com.cognizant.loan\_service.controller;  
  
import org.springframework.web.bind.annotation.\*;  
import java.util.\*;  
  
@RestController  
@RequestMapping("/loans")  
public class LoanController {  
 @GetMapping("/{number}")  
 public Map<String, Object> getLoan(@PathVariable String number) {  
 Map<String, Object> data = new HashMap<>();  
 data.put("number", number);  
 data.put("type", "car");  
 data.put("loan", 400000);  
 data.put("emi", 3258);  
 data.put("tenure", 18);  
 return data;  
 }  
}

**STEP 5: Run & Test**

* Right-click → Run LoanServiceApplication.java
* Test in browser:  
  <http://localhost:8081/loans/H12345>
* Also refresh:  
  <http://localhost:8761> — To see LOAN-SERVICE

**Output:**

****

**In browser:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**In Eureka Server:**

**A screenshot of a computer

AI-generated content may be incorrect.**