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.

Follow steps below to implement:

# Create and Launch Eureka Discovery Server

* Using https://start.spring.io generate a project with following configuration:
  + Group: com.cognizant
  + Artifact: eureka-discovery-server
  + Module: Spring Cloud Discovery > Eureka Server
* Download the project, build it using maven in command line
* Import the project in Eclipse
* Include @EnableEurekaServer in

class EurekaDiscoveryServerApplication

* Include the following configurations in application.properties:

server.port=8761

eureka.client.register-with-eureka=false eureka.client.fetch-registry=false

logging.level.com.netflix.eureka=OFF logging.level.com.netflix.discovery=OFF

* The above configuration runs the discovery service in port 8761
* The eureka properties prohibits direct registration of services, instead discovery server will find available services and register them.
* Launch the service by running the application class
* The discovery service can be view by launching [http://locahost:8761](http://locahost:8761/) in the browser.
* This will display the discover server details
* Look into the section "Instances currently registered with Eureka", which will have an empty list
* Follow steps below to add account and loan service to this discovery server.

# Register Account REST API to eureka discovery

* Go to https://start.spring.io and provide the following configuration:
  + Group: com.cognizant
  + Artifact: account
  + Modules:
    - Spring Boot DevTools
    - Eureka Discovery Client
    - Spring Web
* Click "Explore", which will open pom.xml
* Use copy option in the opened window to copy the pom.xml and overwrite the pom.xml in account project
* Build the project using maven in console
* Include @EnableDiscoveryClient annotation to application class of account project
* Include application name for account application as specified below in application.properties. This is the name that will be displayed in the eureka discovery registry.

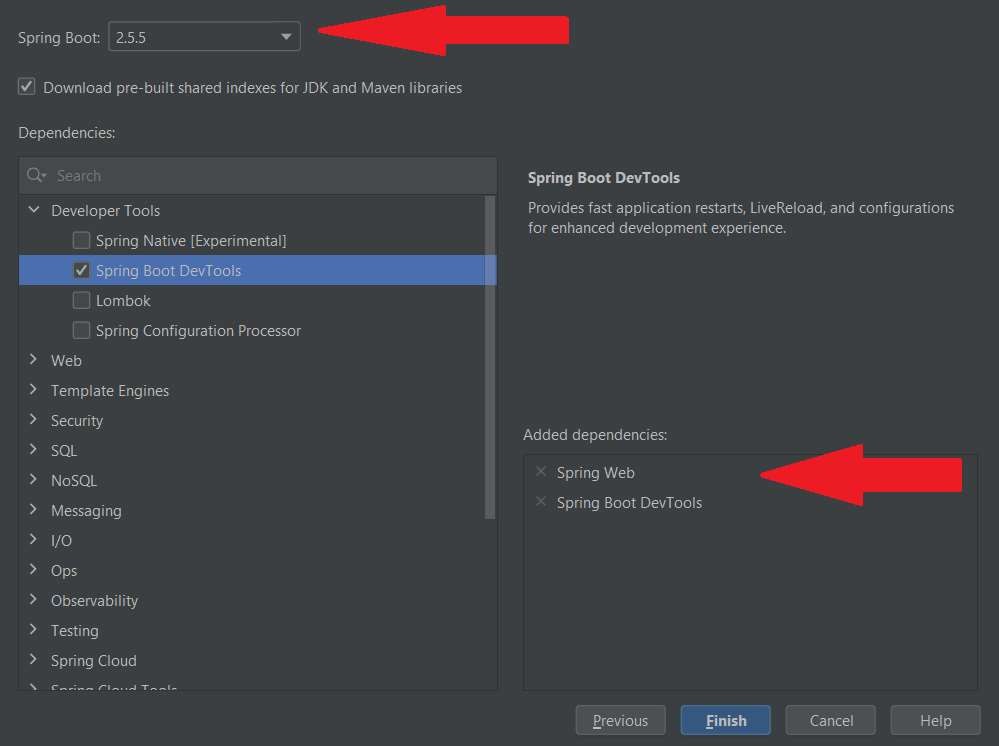
spring.application.name=account-service

* Stop all services (account, loan, eureka-discovery-server) using the console window of Eclipse. Use the monitor icon in console view to switch between applications and use the Terminate button to stop the server.
* First start eureka-discovery-server and wait till the application starts completely. Then open [http://locahost:8761](http://locahost:8761/) in browser. The service list should be empty.
* Then start account application and wait till the application starts.
* Refresh the eureka-discovery-server web page in browser, the account- service will be listed in the registry
* Perform similar steps for loan application and have it registered with eureka-discovery-server.

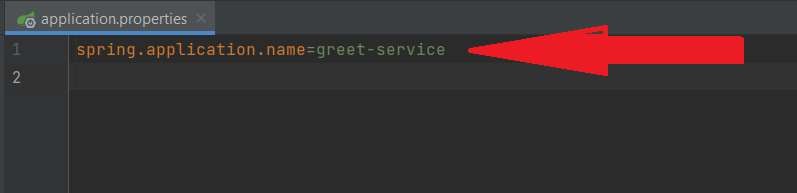
Create a Spring Cloud API Gateway and call one microservice thru the API gateway. Configure a global filter to log each request targeting the microservice using Spring Cloud API Gateway.

Steps.

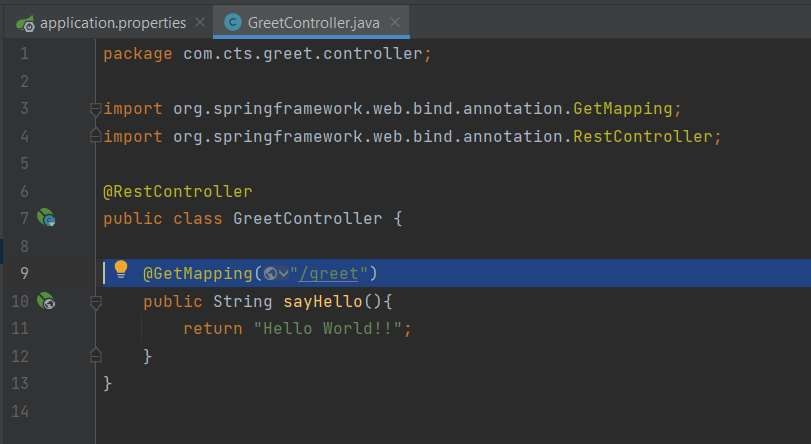
1. Create Simple Microservice **greet-service** that returns “Hello World” using Spring Initializer.
2. Select the latest version of Spring Boot and the dependencies as shown in the image below.



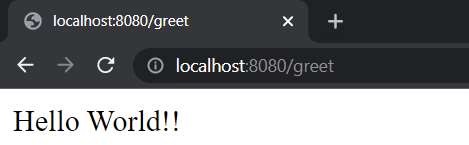
1. Configure the application name in “application.properties” as shown below



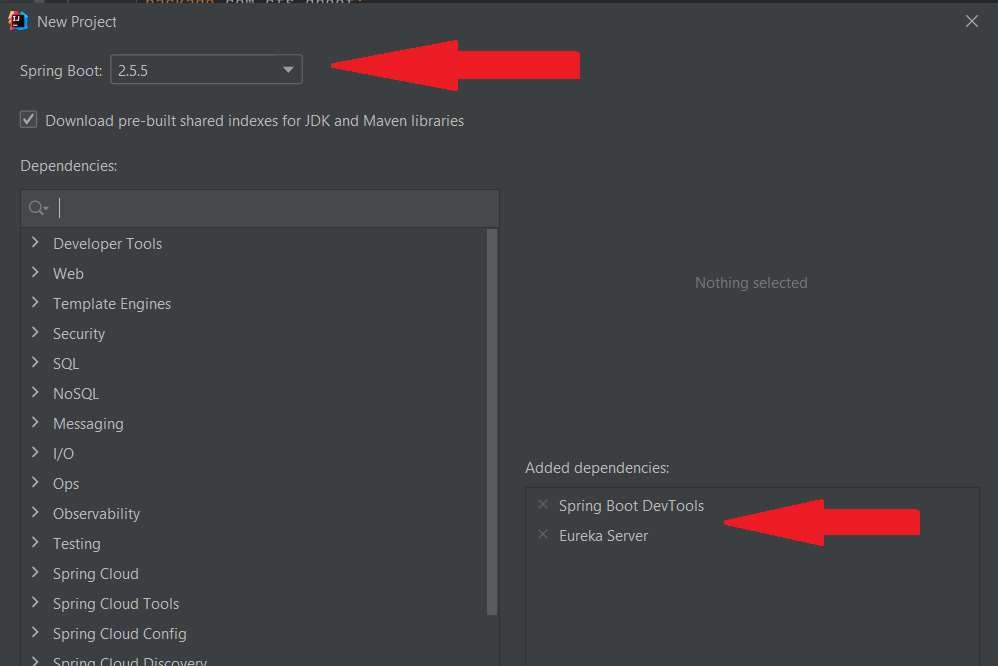
1. Create a controller as shown below.



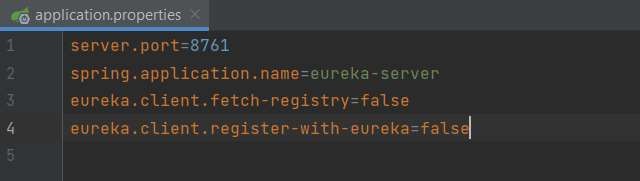
1. Run the microservice and make sure that it is working fine as shown below



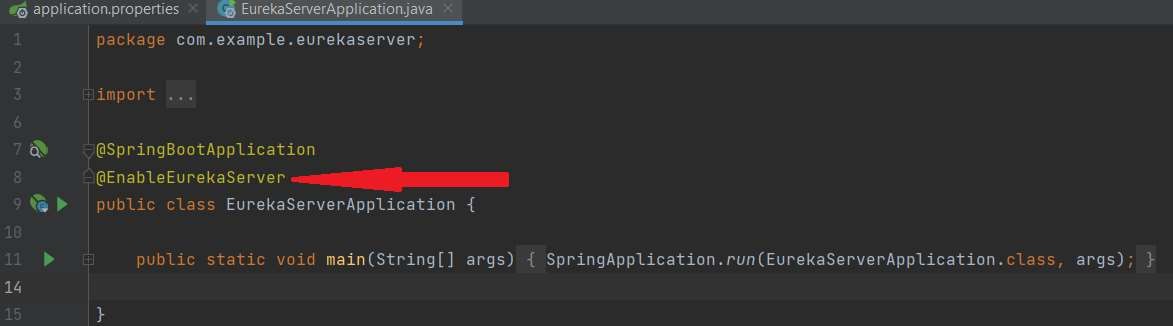
1. Create a second microservice that is acting as the Discovery Server using Spring Cloud Eureka Server.
2. Select the latest version of Spring Boot and the required dependencies as shown below



1. Include the following configurations in the “application.properties” of the Eureka server

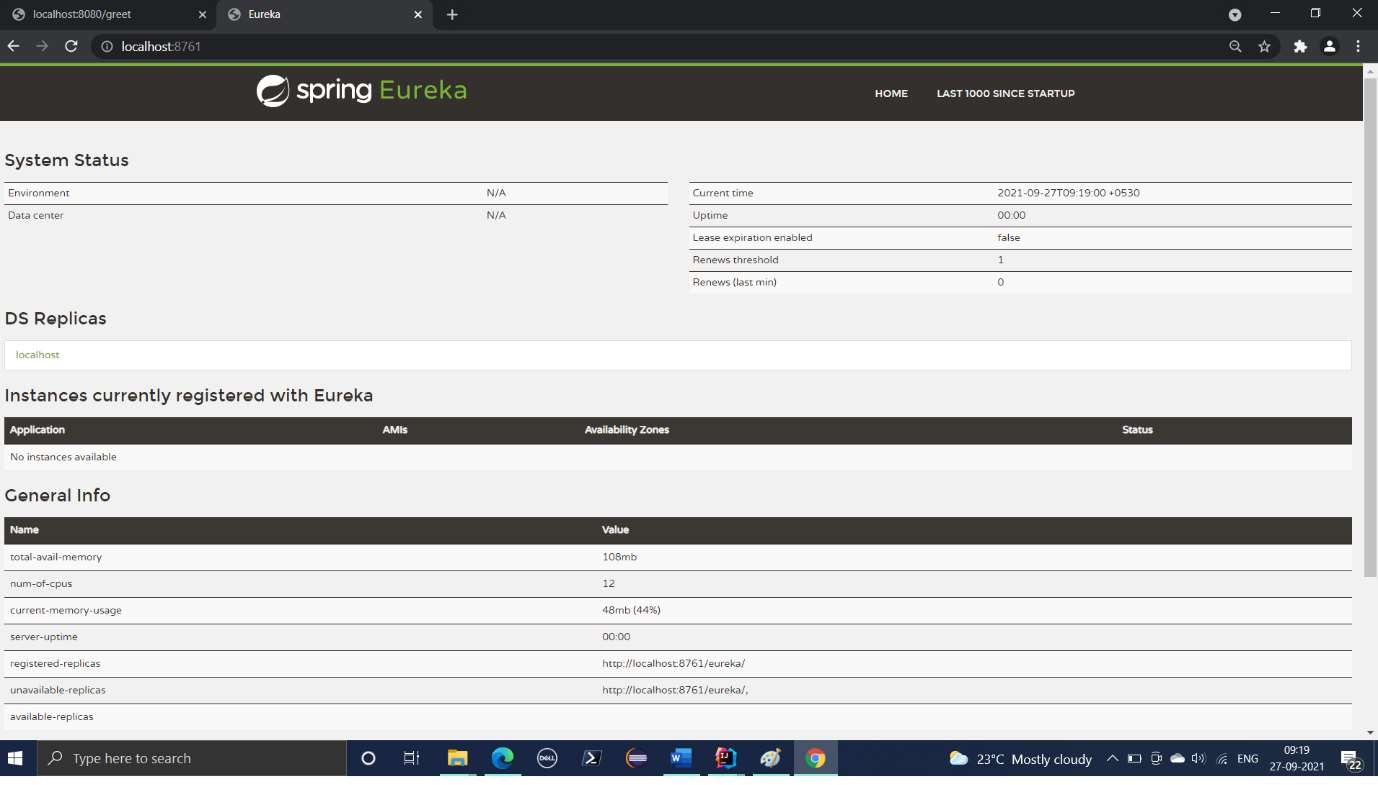


1. Annotate the main class in Eureka Server with @EnableEurekaServer as shown below.

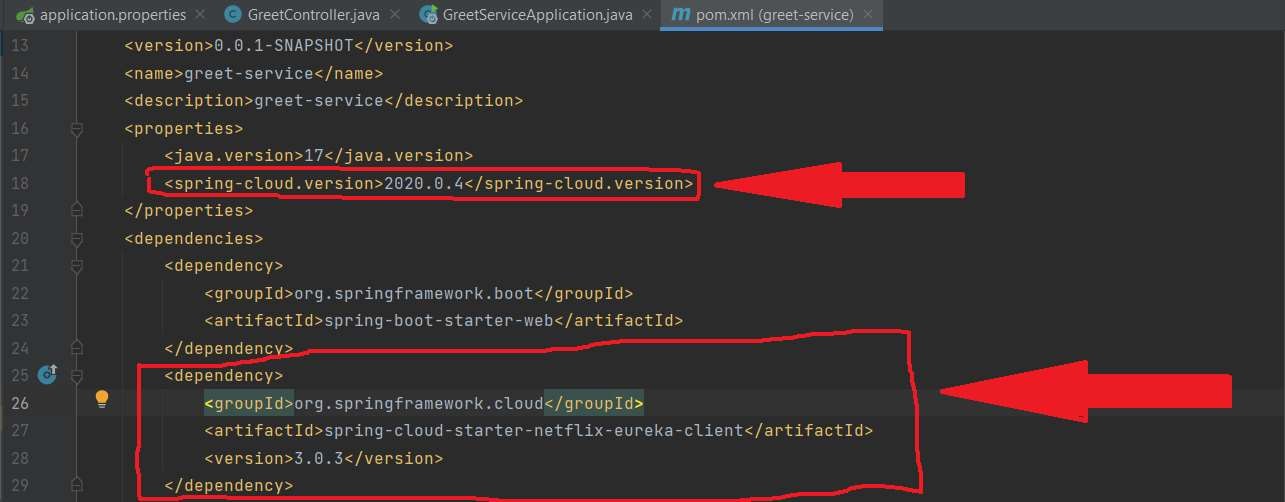


1. Run the Eureka Server and ensure that the server is functioning properly by entering the below URL at any of the browser.

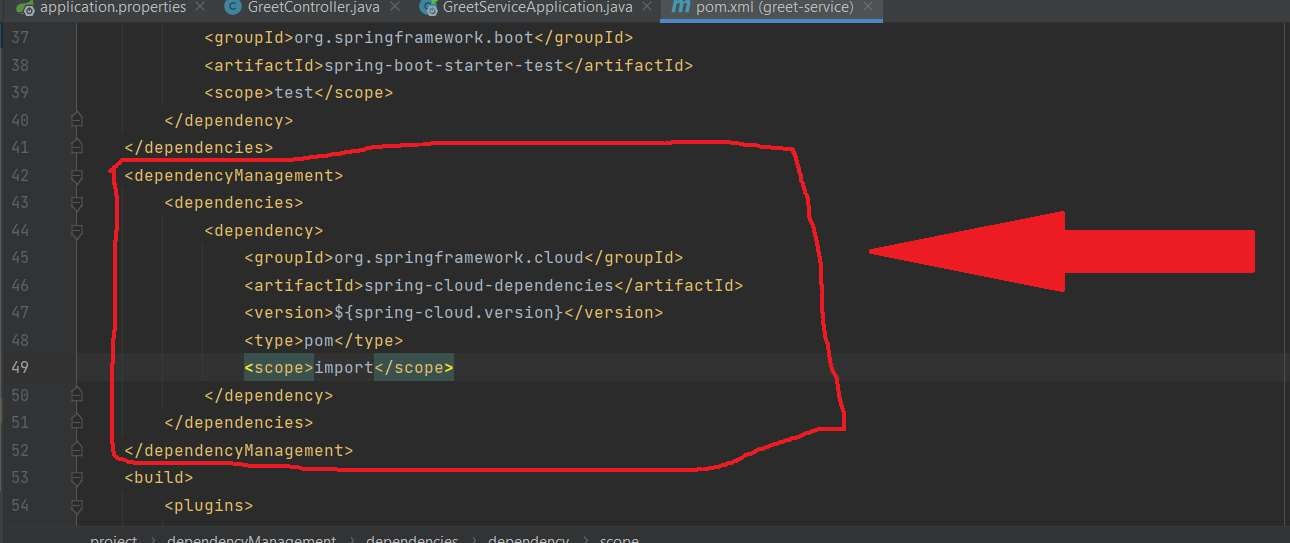
[http://localhost:8761](http://localhost:8761/)



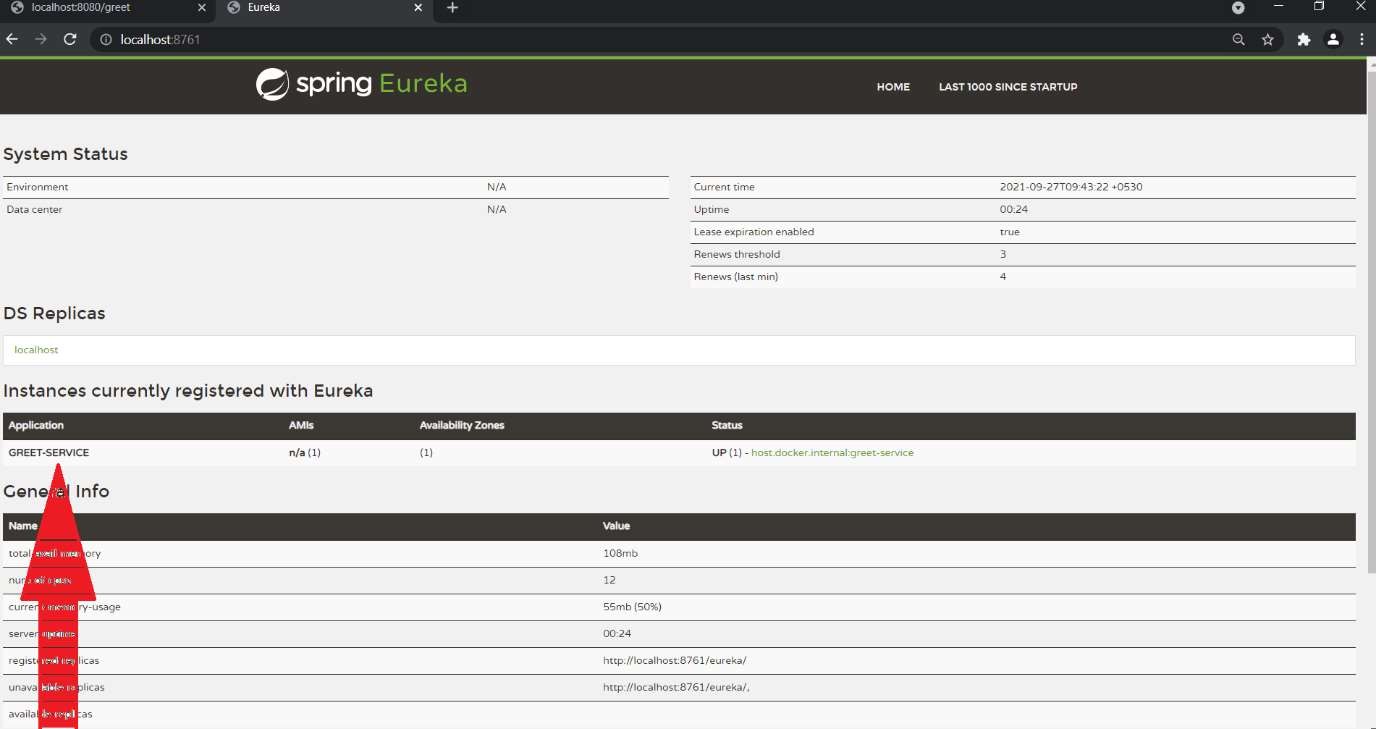
1. Open the pom.xml of greet-service microservice and add the eureka client dependencies as shown in the image below.
2. Copy the spring cloud version from pom.xml of eureka-server and paste it at the appropriate location in the pom.xml of greet-service as shown below.



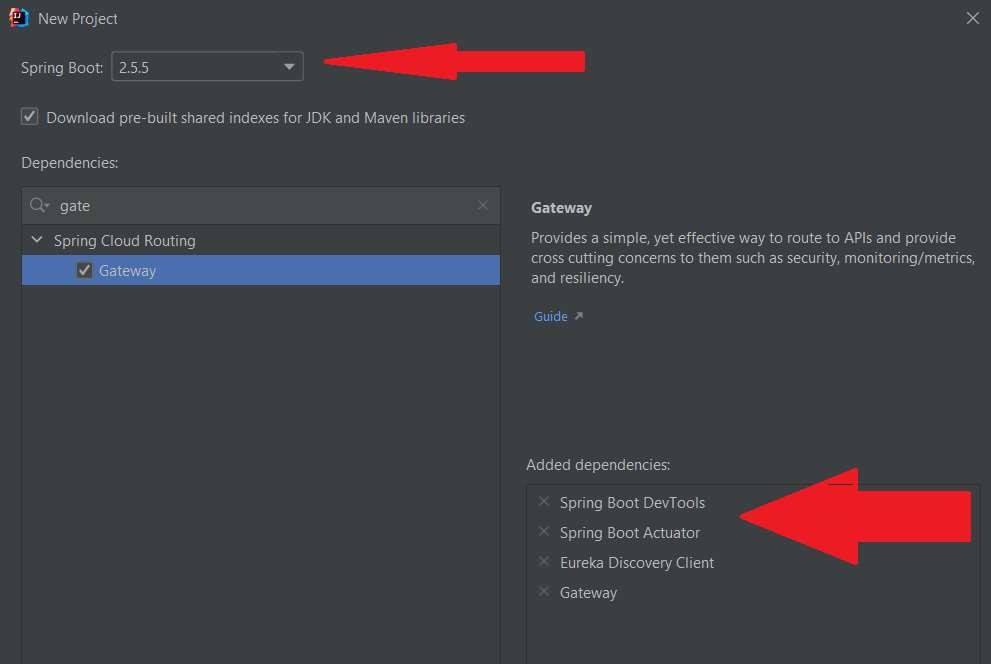
1. Copy the “dependency-management” session in the pom.xml of the eureka-server and paste it immediately after the dependencies session of the pom.xml of the greet-service as shown below.



1. Restart greet-service and refresh the [http://localhost:8761](http://localhost:8761/) to see whether the name of the greet-service figuring in the eureka-server console as shown below.



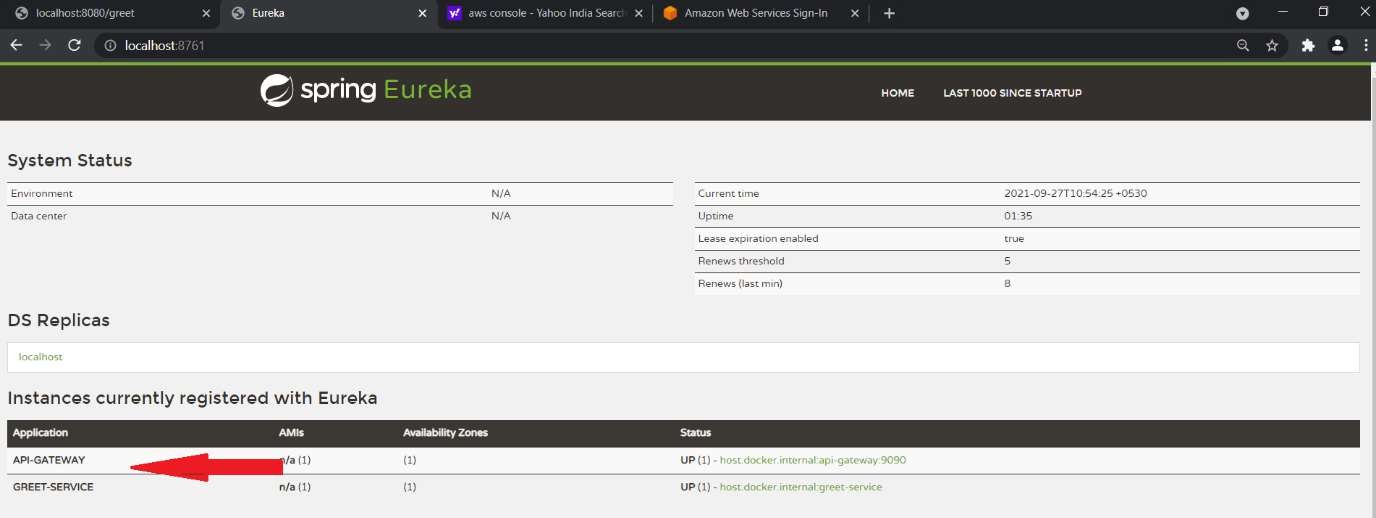
1. Create another microservice “api-gateway” using spring initializer.
2. Select the latest version of Spring Boot and add the required dependencies as shown below.



1. Make the necessary configurations in the “application.properties” file as shown below.

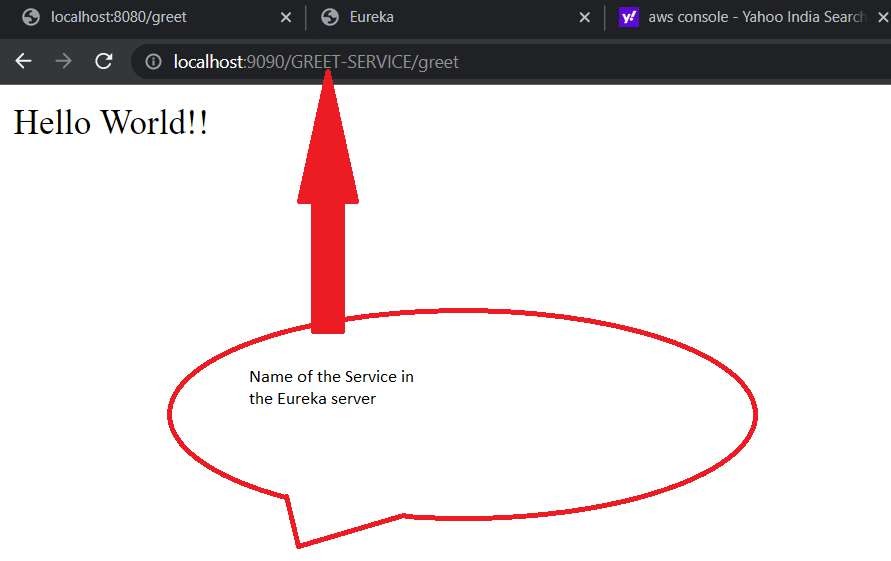


1. Run the api-gateway service and check if it is getting registered with the eureka-server.



1. Try to access the following URL from any of the browser and see if you are able to access the greet-service thru the api-gateway.

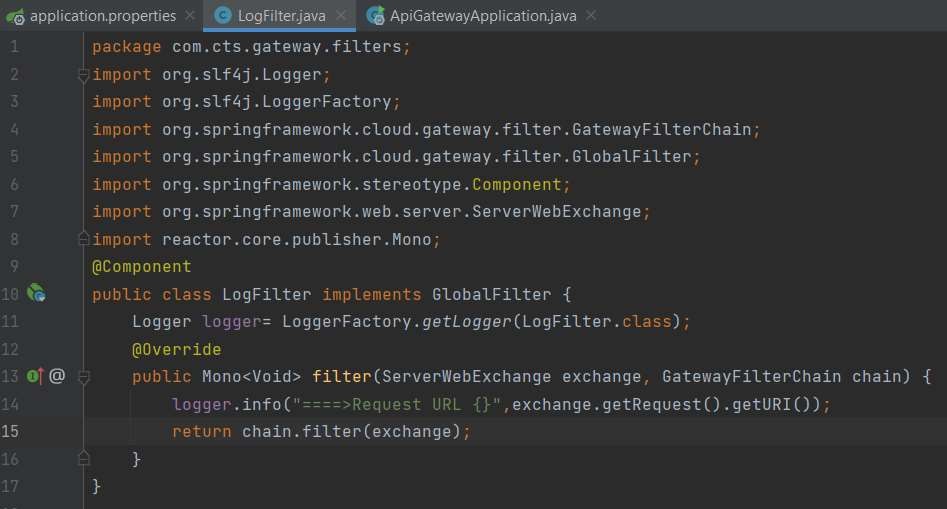
<http://localhost:9090/GREET-SERVICE//greet>



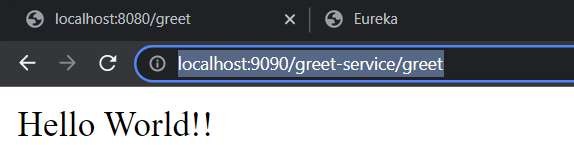
1. Include the following configuration in the “ application.properties” of the api-gateway to specify the service name in lower case.



1. Implement a global filter which logs all incoming requests.
   1. Create a LogFilter class as shown below.



1. Try to access the url [http://localhost:9090/greet-service/greet.](http://localhost:9090/greet-service/greet) You will get the output as shown below.



1. Check the console of the api-gateway service and you should be getting the log shown below.

**Source Code:**

**Eur.java:**

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class Eur {

public static void main(String[] args) {

SpringApplication.run(EurekaDiscoveryServerApplication.class, args);

    }

}

**application.properties:**

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF

**Greet.java:**

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class Greet {

public static void main(String[] args) {

SpringApplication.run(GreetServiceApplication.class, args);

    }

}

**Con.java:**

package com.cognizant;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class Con {

@GetMapping("/greet")

public String greet() {

return "Hello World";

    }

}

**application.properties:**

server.port=8081

spring.application.name=greet-service

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

**AccountServiceApplication.java:**

package com.cognizant;

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);

}

}

**AccountController.java:**

package com.cognizant;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class AccountController {

@GetMapping("/account")

public String getAccountDetails() {

return "Account Service - Account details fetched successfully!";

}

}

**application.properties:**

server.port=8082

spring.application.name=account-service

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

**Loan.java:**

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class Loan {

public static void main(String[] args) {

SpringApplication.run(LoanServiceApplication.class, args);

}

}

**LoanController.java:**

package com.cognizant;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class LoanController {

@GetMapping("/loan")

public String getLoanDetails() {

return "Loan Service - Loan details fetched successfully!";

}

}

**application.properties:**

server.port=8083

spring.application.name=loan-service

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

**ApiGatewayApplication.java:**

package com.cognizant;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

import org.springframework.cloud.gateway.config.EnableGateway;

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**LogFilter.java:**

package com.cognizant;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.cloud.gateway.filter.GlobalFilter;

import org.springframework.core.Ordered;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

@Component

public class LogFilter implements GlobalFilter, Ordered {

private static final Logger logger = LoggerFactory.getLogger(LogFilter.class);

@Override

public Mono<Void> filter(ServerWebExchange exchange, org.springframework.cloud.gateway.filter.GatewayFilterChain chain) {

logger.info("Incoming request path: " + exchange.getRequest().getPath());

return chain.filter(exchange);

}

@Override

public int getOrder() {

return -1;

}

}

**application.properties:**

server.port=9090

spring.application.name=api-gateway

spring.cloud.gateway.discovery.locator.enabled=true

spring.cloud.gateway.discovery.locator.lower-case-service-id=true

spring.cloud.gateway.routes[0].id=greet-service

spring.cloud.gateway.routes[0].uri=lb://greet-service

spring.cloud.gateway.routes[0].predicates[0]=Path=/greet-service/\*\*

spring.cloud.gateway.routes[1].id=account-service

spring.cloud.gateway.routes[1].uri=lb://account-service

spring.cloud.gateway.routes[1].predicates[0]=Path=/account-service/\*\*

spring.cloud.gateway.routes[2].id=loan-service

spring.cloud.gateway.routes[2].uri=lb://loan-service

spring.cloud.gateway.routes[2].predicates[0]=Path=/loan-service/\*\*

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

**OUTPUT:**

http://localhost:9090/account-service/account Account Service - Account details fetched successfully!

http://localhost:9090/loan-service/loan Loan Service - Loan details fetched successfully!

http://localhost:9090/greet-service/greet Hello World

**API Gateway Console Log Output**

Incoming request path: /greet-service/greet

Incoming request path: /account-service/account

Incoming request path: /loan-service/loan