**Digital Nurture 4.0 Java FSE Additional Mandatory**

**Microservices with API Gateway – Assignment Instructions**

**1. Create and Launch Eureka Discovery Server**

Eureka Discovery Server holds a registry of all the services available for consumption. Any client or consumer can query this server to discover available services. Eureka is part of the Spring Cloud module.

**Steps:**

1. **Generate the Eureka Server Project:**
   * Go to: <https://start.spring.io>
   * Configuration:
     + **Group**: com.cognizant
     + **Artifact**: eureka-discovery-server
     + **Dependencies**:
       - Spring Cloud Discovery > **Eureka Server**
2. **Build and Import:**
   * Download the generated ZIP project.
   * Extract and build it via Maven command line:

bash

CopyEdit

mvn clean install

* + Import the project into **Eclipse IDE**.

1. **Code Setup:**
   * In EurekaDiscoveryServerApplication.java, add the annotation:

java

CopyEdit

@EnableEurekaServer

1. **Configure application.properties:**

properties

CopyEdit

server.port=8761

spring.application.name=eureka-discovery-server

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF

1. **Run the Eureka Discovery Server**
   * Launch the application from the main class.
   * Open <http://localhost:8761> in your browser.
   * The **"Instances currently registered with Eureka"** section should be empty initially.

**2. Register Microservices to Eureka**

**Server Register Account Service**

1. Go to: <https://start.spring.io>
   * Configuration:
     + **Group**: com.cognizant
     + **Artifact**: account
     + **Dependencies**:
       - Spring Boot DevTools
       - Eureka Discovery Client
       - Spring Web
2. Copy the generated pom.xml into your existing account project.
3. In the AccountApplication.java, add:

java

CopyEdit

@EnableDiscoveryClient

1. Add to application.properties:

properties

CopyEdit

spring.application.name=account-service

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

1. **Start the applications in order:**
   * Stop all running services.
   * Start eureka-discovery-server first.
   * Then start account service.
   * Visit <http://localhost:8761> to verify account-service is registered.

**Register Loan Service**

Repeat the same steps for loan microservice:

* Artifact: loan
* Use the same dependencies.
* Update:

properties

CopyEdit

spring.application.name=loan-service

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

* Once running, verify both account-service and loan-service appear on Eureka dashboard.

**3. Create Spring Cloud API Gateway**

**⚙️ Step-by-Step Setup**

1. **Create greet-service Microservice**
   * Use Spring Initializer:
     + **Group**: com.cognizant
     + **Artifact**: greet-service
     + **Dependencies**:
       - Spring Boot DevTools
       - Eureka Discovery Client
       - Spring Web
2. **Configure application.properties:**

properties

CopyEdit

server.port=8082

spring.application.name=greet-service

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

1. **Create Controller:**

java

CopyEdit

@RestController

public class GreetController {

@GetMapping("/greet")

public String greet() {

return "Hello World";

}

}

1. **Verify:**
   * Run the service and visit:

bash

CopyEdit

http://localhost:8082/greet

1. **Register greet-service to Eureka**
   * Ensure it's listed at <http://localhost:8761>

**4. Create and Configure API Gateway**

1. **Generate API Gateway Project**
   * Go to Spring Initializer:
     + **Group**: com.cognizant
     + **Artifact**: api-gateway
     + **Dependencies**:
       - Eureka Discovery Client
       - Spring Cloud Gateway
2. **Update application.properties:**

properties

CopyEdit

server.port=9090

spring.application.name=api-gateway

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

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

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

1. **Test Gateway Routing:**
   * Run api-gateway
   * Access via browser:

bash

CopyEdit

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

1. **Implement Global Filter for Logging Requests**
   * Create class LogFilter.java:

java

CopyEdit

@Component

public class LogFilter implements GlobalFilter {

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

@Override

public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

logger.info("Incoming request: {}", exchange.getRequest().getURI());

return chain.filter(exchange);

}

}

1. **Verify Output:**
   * Visit:

bash

CopyEdit

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

* + You should see "Hello World" as response.
  + Console should log:

bash

CopyEdit

Incoming request: /greet-service/greet

**Output:**



