

# Assignment No. 2

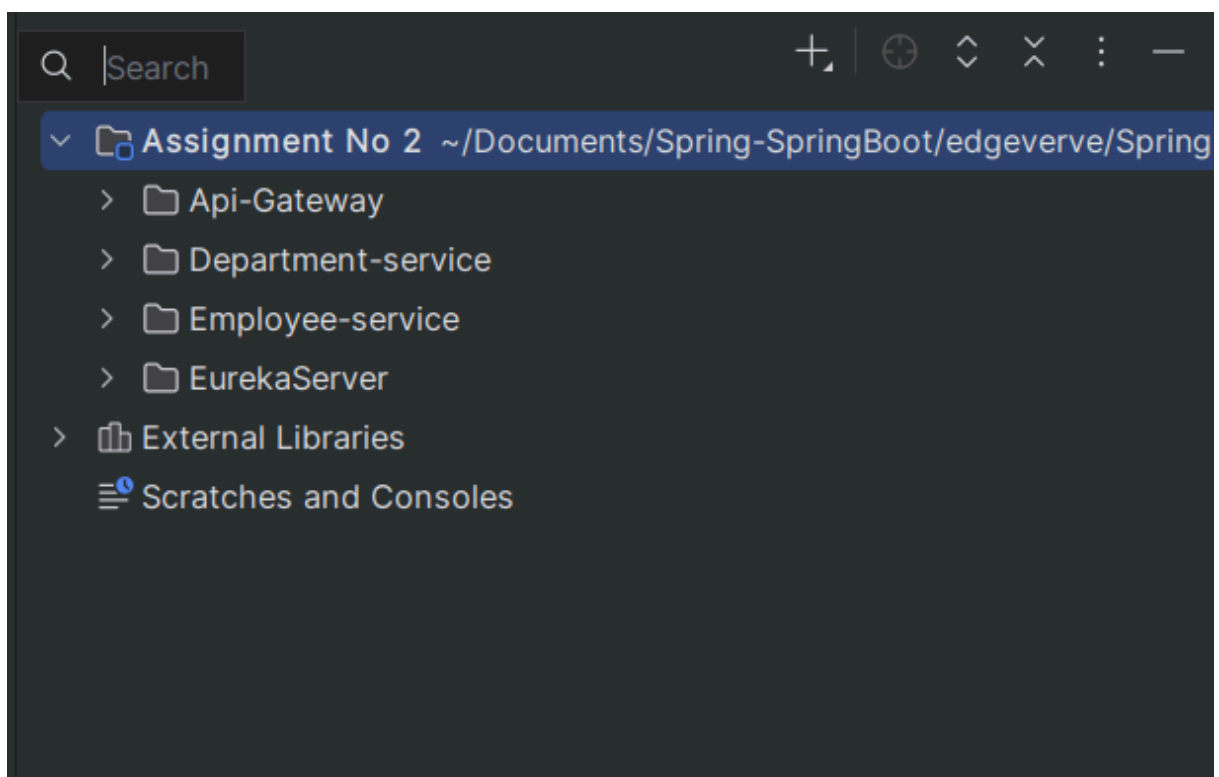
---

Name - Aditya Pawar

USN - 72233061J

---

Folder Structure:



---

## Department-service

Dependencies

<b>Spring Web</b>	WEB	—
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.		
<b>Spring Data JPA</b>	SQL	—
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.		
<b>PostgreSQL Driver</b>	SQL	—
A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.		
<b>Lombok</b>	DEVELOPER TOOLS	—
Java annotation library which helps to reduce boilerplate code.		
<b>Validation</b>	I/O	—
Bean Validation with Hibernate validator.		
<b>Eureka Discovery Client</b>	SPRING CLOUD DISCOVERY	—
A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.		
<b>Cloud Bootstrap</b>	SPRING CLOUD	—
Non-specific Spring Cloud features, unrelated to external libraries or integrations (e.g. Bootstrap context and @RefreshScope).		

## DepartmentController.java

```
package com.aditya.Department_service.controller;

import com.aditya.Department_service.model.Department;
import com.aditya.Department_service.repository
.DepartmentRepo;
import org.springframework.beans.factory.annotation
.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/departments")
public class DepartmentController {

    @Autowired
    private DepartmentRepo departmentRepo;

    @GetMapping("/allDepartments")
    public List<Department> getAllDepartments(){
```

```

        return departmentRepo.findAll();
    }

    @PostMapping("/addDepartment")
    public Department addDepartment(@RequestBody
    Department department){
        return departmentRepo.save(department);
    }

    @GetMapping("/{deptId}")
    public ResponseEntity<Department>
    getDepartmentById(@PathVariable("deptId")
    Long deptId){
        Department department = departmentRepo.
        findById(deptId).orElseThrow(() → new
        RuntimeException("Department ID Not Found:
        " +deptId ));
        return ResponseEntity.ok(department);
    }
}

```

## Department.java

```

package com.aditya.Department_service.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Entity
@Table(name = "department-info")
public class Department {

```

```
@Id
@GeneratedValue(strategy = GenerationType
.IDENTITY)
private Long deptId;
private String deptName;
private Long managerId;
}
```

## DepartmentRepo.java

```
package com.aditya.Department_service.repository;

import com.aditya.Department_service.model.Department;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface DepartmentRepo extends JpaRepository<Department, Long>
{
}
```

## DepartmentServiceApplication.java

```
package com.aditya.Department_service;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class DepartmentServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(DepartmentServiceApplication.class, args);
    }

}
```

## Application.properties

```
spring.application.name=Department-service

spring.datasource.username=postgres
spring.datasource.password=root
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/department_db
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.data.jpa.repositories.bootstrap-mode=default
spring.jpa.defer-datasource-initialization=true
server.port=8082
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format-sql=true

eureka.client.service-url-defaultZone=http://localhost:8761/eureka/
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.instance.prefer-ip-address=true
```

## Output:

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'Department-service' with a 'main' directory containing a 'java' package. The package structure includes 'com.aditya.DepartmentServiceApplication', 'controller', 'DepartmentController', 'Department', 'DepartmentRes', and 'EmployeeDTO'.
- Editor:** Displays the 'DepartmentServiceApplication.java' file. The code is as follows:

```
1 package com.aditya.Department_service;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 @EnableDiscoveryClient
8 public class DepartmentServiceApplication {
9
10     public static void main(String[] args) { SpringApplication.run(DepartmentServiceApplication.class, args); }
11 }
12
13
14
15
16
```
- Run Console:** Shows the execution logs for 'DepartmentServiceApplication'. The logs indicate that the application started successfully on port 8082. The logs also show that the application is registered with the Eureka service and that the response status is 200.

## Employee-service

### Dependencies

<b>Spring Web</b> <span>WEB</span>	Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.	—
<b>Spring Data JPA</b> <span>SQL</span>	Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.	—
<b>PostgreSQL Driver</b> <span>SQL</span>	A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.	—
<b>Lombok</b> <span>DEVELOPER TOOLS</span>	Java annotation library which helps to reduce boilerplate code.	—
<b>Validation</b> <span>I/O</span>	Bean Validation with Hibernate validator.	—
<b>Eureka Discovery Client</b> <span>SPRING CLOUD DISCOVERY</span>	A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.	—
<b>Cloud Bootstrap</b> <span>SPRING CLOUD</span>	Non-specific Spring Cloud features, unrelated to external libraries or integrations (e.g. Bootstrap context and @RefreshScope).	—
<b>Spring Reactive Web</b> <span>WEB</span>	Build reactive web applications with Spring WebFlux and Netty.	—

## WebClientConfig.java

```
package com.aditya.Employee_service.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.reactive.function.client.WebClient;

@Configuration
public class WebClientConfig {

    @Bean
    public WebClient.Builder webClientBuilder(){
        return WebClient.builder();
    }
}
```

```
}  
}
```

## EmployeeController.java

```
package com.aditya.Employee_service.controller;  
  
import com.aditya.Employee_service.model.DepartmentDTO;  
import com.aditya.Employee_service.model.Employee;  
import com.aditya.Employee_service.model.EmployeeResponseDTO;  
import com.aditya.Employee_service.repository.EmployeeRepo;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.*;  
import org.springframework.web.reactive.function.client.WebClient;  
import reactor.core.publisher.Mono;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/employees")  
public class EmployeeController {  
  
    @Autowired  
    private EmployeeRepo employeeRepo;  
  
    @Autowired  
    private WebClient.Builder webClientBuilder;  
  
    @GetMapping("/allEmployees")  
    public List<Employee> allEmployees(){  
        return employeeRepo.findAll();  
    }  
  
    @GetMapping("/{empId}")  
    public ResponseEntity<Employee> getEmployeeById(@PathVariable("empId")  
        empId){  
        Employee employee = employeeRepo.findById(empId).orElseThrow(() ->
```



```

        new RuntimeException("employee ID Not Found: "+empld ));
        return ResponseEntity.ok(employee);
    }

    @PostMapping("/addEmployee")
    public Mono<ResponseEntity<EmployeeResponseDTO>> addDepartment(@
    Employee employee){
        return webClientBuilder.build().get()
            .uri("http://localhost:8082/departments/" + employee.getDeptId())
            .retrieve()
            .bodyToMono(DepartmentDTO.class)
            .map(departmentDTO → {
                employeeRepo.save(employee);

                EmployeeResponseDTO responseDTO = new EmployeeResponseDTO();
                responseDTO.setEmpld(employee.getEmpld());
                responseDTO.setEmpName(employee.getEmpName());
                responseDTO.setEmpSalary(employee.getEmpSalary());

                responseDTO.setDeptId(departmentDTO.getDeptId());
                responseDTO.setDeptName(departmentDTO.getDeptName());
                responseDTO.setManagerId(departmentDTO.getManagerId());
                return ResponseEntity.ok(responseDTO);
            });
    }
}

```

## Employee.java

```

package com.aditya.Employee_service.model;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@AllArgsConstructor

```

```

@NoArgsConstructor
@Getter
@Setter
@Entity
@Table(name = "employee-info")
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long empId;
    private String empName;
    private Long deptId;
    private Double empSalary;
}

```

## EmployeeResponseDTO.java

```

package com.aditya.Employee_service.model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class EmployeeResponseDTO {
    private Long empId;
    private String empName;
    private Double empSalary;

    private Long deptId;
    private String deptName;
    private Long managerId;
}

```

## DepartmentDTO.java

```
package com.aditya.Employee_service.model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class EmployeeResponseDTO {
    private Long empld;
    private String empName;
    private Double empSalary;

    private Long deptId;
    private String deptName;
    private Long managerId;
}
```

## EmployeeRepo.java

```
package com.aditya.Employee_service.repository;

import com.aditya.Employee_service.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EmployeeRepo extends JpaRepository<Employee, Long> {
}
```

## EmployeeServiceApplication.java

```

package com.aditya.Employee_service;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class EmployeeServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(EmployeeServiceApplication.class, args);
    }
}

```

## Application.properties

```

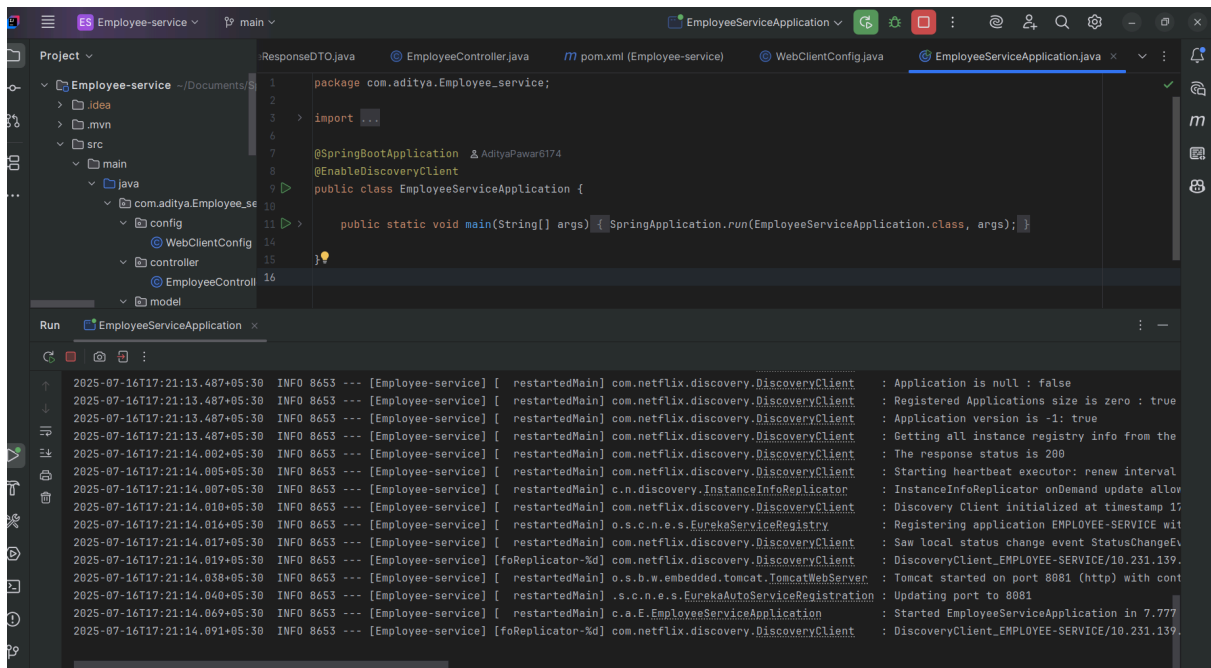
spring.application.name=Employee-service

spring.datasource.username=postgres
spring.datasource.password=root
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/employee_db
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.data.jpa.repositories.bootstrap-mode=default
spring.jpa.defer-datasource-initialization=true
server.port=8081
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format-sql=true

eureka.client.service-url-defaultZone=http://localhost:8761/eureka/
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
eureka.instance.prefer-ip-address=true

```

## Output:



The screenshot shows an IDE with the `EmployeeServiceApplication.java` file open. The code is as follows:

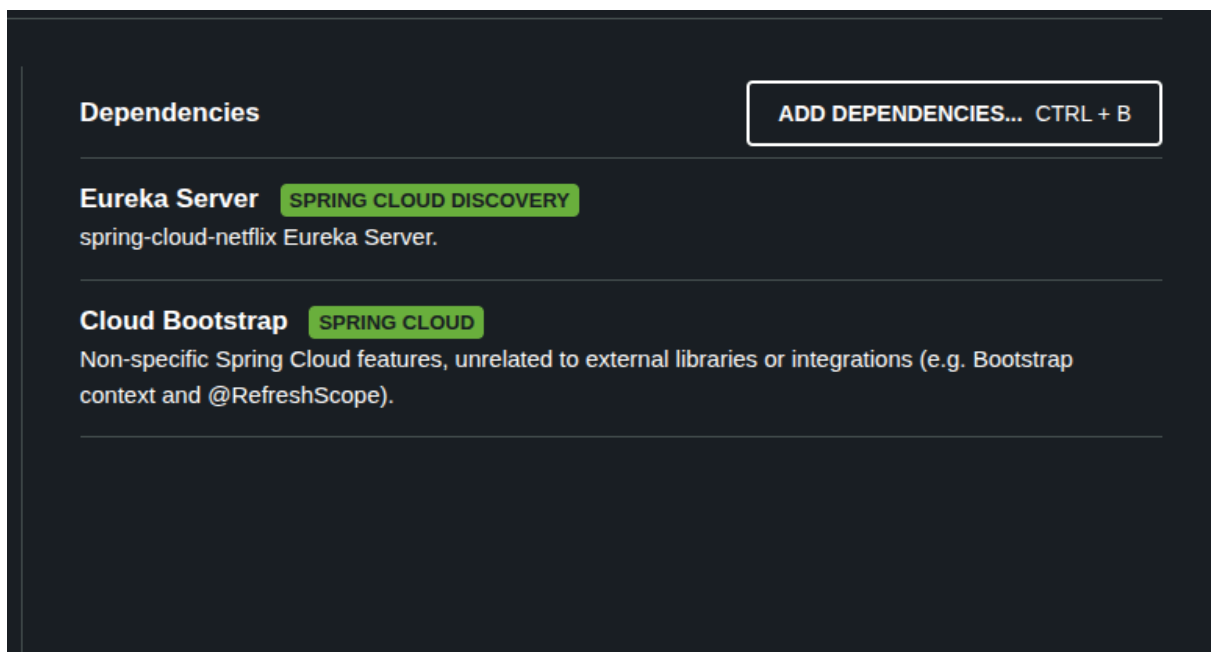
```
1 package com.aditya.Employee_service;
2
3 import ...
4
5
6
7 @SpringBootApplication & AdityaPawar6174
8 @EnableDiscoveryClient
9 public class EmployeeServiceApplication {
10
11     >
12     public static void main(String[] args) { SpringApplication.run(EmployeeServiceApplication.class, args); }
13
14
15 }
16
```

The output console shows the following logs:

```
2025-07-16T17:21:13.487+05:30 INFO 8653 --- [Employee-service] [ restartedMain] com.netflix.discovery.DiscoveryClient : Application is null : false
2025-07-16T17:21:13.487+05:30 INFO 8653 --- [Employee-service] [ restartedMain] com.netflix.discovery.DiscoveryClient : Registered Applications size is zero : true
2025-07-16T17:21:13.487+05:30 INFO 8653 --- [Employee-service] [ restartedMain] com.netflix.discovery.DiscoveryClient : Application version is -1: true
2025-07-16T17:21:13.487+05:30 INFO 8653 --- [Employee-service] [ restartedMain] com.netflix.discovery.DiscoveryClient : Getting all instance registry info from the
2025-07-16T17:21:14.082+05:30 INFO 8653 --- [Employee-service] [ restartedMain] com.netflix.discovery.DiscoveryClient : The response status is 200
2025-07-16T17:21:14.085+05:30 INFO 8653 --- [Employee-service] [ restartedMain] com.netflix.discovery.DiscoveryClient : Starting heartbeat executor: renew interval
2025-07-16T17:21:14.087+05:30 INFO 8653 --- [Employee-service] [ restartedMain] c.n.discovery.InstanceInfoReplicator : InstanceInfoReplicator onDemand update allow
2025-07-16T17:21:14.087+05:30 INFO 8653 --- [Employee-service] [ restartedMain] com.netflix.discovery.DiscoveryClient : Discovery Client initialized at timestamp 17
2025-07-16T17:21:14.016+05:30 INFO 8653 --- [Employee-service] [ restartedMain] o.s.c.n.e.s.EurekaServiceRegistry : Registering application EMPLOYEE-SERVICE wit
2025-07-16T17:21:14.017+05:30 INFO 8653 --- [Employee-service] [ restartedMain] com.netflix.discovery.DiscoveryClient : Saw local status change event StatusChangeEx
2025-07-16T17:21:14.019+05:30 INFO 8653 --- [Employee-service] [foReplicator-%d] com.netflix.discovery.DiscoveryClient : DiscoveryClient_EMPLOYEE-SERVICE/10.231.139.
2025-07-16T17:21:14.038+05:30 INFO 8653 --- [Employee-service] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8081 (http) with cont
2025-07-16T17:21:14.040+05:30 INFO 8653 --- [Employee-service] [ restartedMain] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8081
2025-07-16T17:21:14.069+05:30 INFO 8653 --- [Employee-service] [ restartedMain] c.a.E.EmployeeServiceApplication : Started EmployeeServiceApplication in 7.777
2025-07-16T17:21:14.091+05:30 INFO 8653 --- [Employee-service] [foReplicator-%d] com.netflix.discovery.DiscoveryClient : DiscoveryClient_EMPLOYEE-SERVICE/10.231.139.
```

## EurekaServer

### Dependencies



The screenshot shows the Spring IDE Dependencies view. It displays the following dependencies:

- Eureka Server** (SPRING CLOUD DISCOVERY)  
spring-cloud-netflix Eureka Server.
- Cloud Bootstrap** (SPRING CLOUD)  
Non-specific Spring Cloud features, unrelated to external libraries or integrations (e.g. Bootstrap context and @RefreshScope).

A button labeled "ADD DEPENDENCIES... CTRL + B" is visible in the top right corner.

## EurekaServerApplication.java

```
package com.aditya.EurekaServer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(EurekaServerApplication.class, args);
    }

}
```

## Application.properties

```
spring.application.name=EurekaServer

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

eureka.instance.prefer-ip-address=true
```

## Output:

```

1 package com.aditya.EurekaServer;
2
3 import ...
4
5 @SpringBootApplication & AdityaPawar6174
6 @EnableEurekaServer
7 public class EurekaServerApplication {
8
9     public static void main(String[] args) { SpringApplication.run(EurekaServerApplication.class, args); }
10
11 }
12
13
14
15
16

```

```

2025-07-16T17:15:43.654+05:30 INFO 7130 --- [EurekaServer] [ Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl : Renew threshold is: 1
2025-07-16T17:15:43.655+05:30 INFO 7130 --- [EurekaServer] [ Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl : Changing status to UP
2025-07-16T17:15:43.657+05:30 INFO 7130 --- [EurekaServer] [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8761 (http) with context
2025-07-16T17:15:43.659+05:30 INFO 7130 --- [EurekaServer] [ main] s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8761
2025-07-16T17:15:43.660+05:30 INFO 7130 --- [EurekaServer] [ Thread-9] e.s.EurekaServerInitializerConfiguration : Started Eureka Server
2025-07-16T17:15:43.724+05:30 INFO 7130 --- [EurekaServer] [ main] c.a.E.EurekaServerApplication : Started EurekaServerApplication in 6.134 seconds
2025-07-16T17:16:31.889+05:30 INFO 7130 --- [EurekaServer] [nio-8761-exec-1] o.a.c.c.f.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcher
2025-07-16T17:16:31.902+05:30 INFO 7130 --- [EurekaServer] [nio-8761-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2025-07-16T17:16:31.932+05:30 INFO 7130 --- [EurekaServer] [nio-8761-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 29 ms
2025-07-16T17:16:43.657+05:30 INFO 7130 --- [EurekaServer] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2025-07-16T17:17:43.657+05:30 INFO 7130 --- [EurekaServer] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2025-07-16T17:18:43.656+05:30 INFO 7130 --- [EurekaServer] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
2025-07-16T17:19:29.698+05:30 INFO 7130 --- [EurekaServer] [nio-8761-exec-3] c.n.e.registry.AbstractInstanceRegistry : Registered instance DEPARTMENT-SERVICE/10.231.11
2025-07-16T17:19:30.481+05:30 INFO 7130 --- [EurekaServer] [nio-8761-exec-2] c.n.e.registry.AbstractInstanceRegistry : Registered instance DEPARTMENT-SERVICE/10.231.11

```

(<https://localhost:8761/>)

HOME LAST 1000 SINCE STARTUP

### System Status

Environment	test	Current time	2025-07-16T17:16:32 +0530
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

### DS Replicas

localhost

### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

### General Info

Name	Value
total-avail-memory	107mb
num-of-cpus	16
current-memory-usage	69mb (64%)
server-uptime	00:00
registered-replicas	<a href="http://localhost:8761/eureka/">http://localhost:8761/eureka/</a>

## Api-Gateway

## Dependencies

Dependencies

ADD DEPENDENCIES... CTRL + B

Eureka Discovery Client

SPRING CLOUD DISCOVERY

A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.

Reactive Gateway

SPRING CLOUD ROUTING

Provides a simple, yet effective way to route to APIs in reactive applications. Provides cross-cutting concerns to those APIs such as security, monitoring/metrics, and resiliency.

## ApiGatewayApplication.java

```
package com.aditya.Api_Gateway;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication
@EnableDiscoveryClient
public class ApiGatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(ApiGatewayApplication.class, args);
    }

}
```

## Application.properties

```
spring.application.name=Api-Gateway

server.port=8080
```



```
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
spring.cloud.gateway.discovery.locator.enabled=false
spring.cloud.netflix.eureka.discovery.enabled=true
```

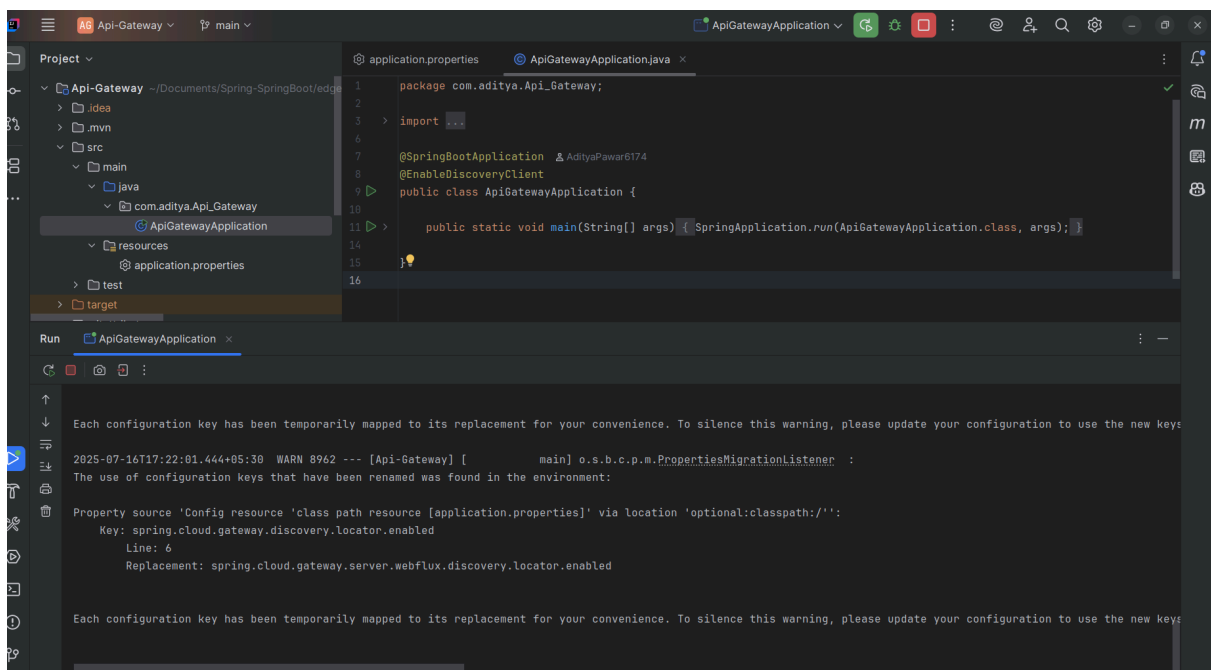
#Define the static router for order service

```
spring.cloud.gateway.routes[0].id=Department-service
spring.cloud.gateway.routes[0].uri=lb://DEPARTMENT-SERVICE
spring.cloud.gateway.routes[0].predicates[0]=Path=/departments/**
```

#Define the static router for Employee service

```
spring.cloud.gateway.routes[1].id=Employee-service
spring.cloud.gateway.routes[1].uri=lb://EMPLOYEE-SERVICE
spring.cloud.gateway.routes[1].predicates[0]=Path=/employees/**
```

## Output:



## Output:

### 1. Eureka Server

localhost:8761

spring Eureka

HOME LAST 1000 SINCE STARTUP

### System Status

Environment	test	Current time	2025-07-16T17:22:59 +0530
Data center	default	Uptime	00:07
		Lease expiration enabled	true
		Renews threshold	6
		Renews (last min)	10

### DS Replicas

localhost

### Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - 10.231.139.142-Api-Gateway:8080
DEPARTMENT-SERVICE	n/a (1)	(1)	UP (1) - 10.231.139.142-Department-service:8082
EMPLOYEE-SERVICE	n/a (1)	(1)	UP (1) - 10.231.139.142-Employee-service:8081

### General Info

Name	Value
total-avail-memory	107mb
num-of-cpus	16
current-memory-usage	71mb (66%)

## 2. Postman

(using Api-gateway port → 8080)

Get Mapping →

GET http://localhost:8080/departments/allDepartments

Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (3) Test Results

200 OK 3.17 s 203 B Save Response

JSON Preview Visualize

```
1 [
2   {
3     "deptId": 1,
4     "deptName": "IT",
5     "managerId": 3
6   },
7   {
8     "deptId": 2,
9     "deptName": "IT",
10    "managerId": 3
11  }
12 ]
```

GET http://localhost:8080/employees/allEmployees Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (3) Test Results 200 OK • 1.10 s • 181 B Save Response

{ } JSON Preview Visualize

```
1 [
2   {
3     "empId": 1,
4     "empName": "Srushti",
5     "deptId": 2,
6     "empSalary": 150000.0
7   }
8 ]
```

Post Mapping →

POST http://localhost:8080/departments/addDepartment Send

Params Authorization Headers (8) Body Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   "deptName" : "HR",
3   "managerId" : "2"
4 }
```

Body Cookies Headers (3) Test Results 200 OK • 264 ms • 158 B Save Response

{ } JSON Preview Visualize

```
1 {
2   "deptId": 3,
3   "deptName": "HR",
4   "managerId": 2
5 }
```

POST http://localhost:8080/employees/addEmployee Send

Params Authorization Headers (8) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "empName" : "Aditya",
3   "deptId" : 3,
4   "empSalary" : 65000
5 }
```

Body Cookies Headers (3) Test Results 200 OK • 1.69 s • 207 B Save Response

{ JSON Preview Visualize

```
1 {
2   "empId": 2,
3   "empName": "Aditya",
4   "empSalary": 65000.0,
5   "deptId": 3,
6   "deptName": "HR",
7   "managerId": 2
8 }
```