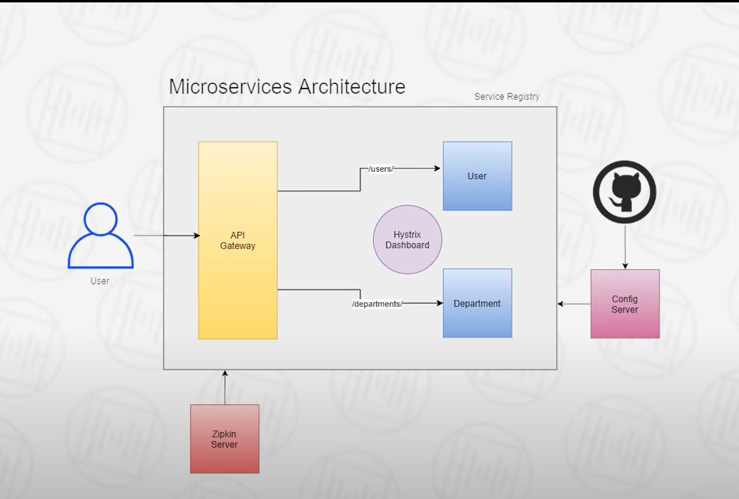
**Spring Microservices**



<https://github.com/shabbirdwd53/Springboot-Microservice/>

[Microservices using SpringBoot | Full Example - YouTube](https://www.youtube.com/watch?v=BnknNTN8icw)

Daily code buffer tutorial in you tube.

Sequence of development

1. Service
2. Service Registry
3. Hystix Dahsboard
4. Api Gateway
5. Config Server

# **1) Service:**

## **Controller**

* **@RestController** - tag as Rest service.
* **Http method -** mapping using one of the following @RequestMapping, @PostMapping, @GetMapping, @Putmapping @DeleteMapping
* **@RequestBody, @Patthvariable** for Input reading.

**Code Snippet ->**

@RestController

@RequestMapping("/departments")

@Slf4j

@Autowired

private DepartmentService departmentService;

@PostMapping("/")

public Department saveDepartment(@RequestBody Department department) {

@GetMapping("/{id}")

public Department findDepartmentById(@PathVariable("id") Long departmentId) {

## **Service**

* **@Service** – Tag the class as Service.
* **@Autowire** -- for linking the repository.
* **@Bean, RestTemplate** – for calling other service with the Bean defined in the Spring boot application.
* **@LoadBalanced** – Load balancing multiple instances is needed.

**Code Snippet ->**

@Service

@Slf4j

@Autowired

private DepartmentRepository departmentRepository;

departmentRepository.save(department);

departmentRepository.findByDepartmentId(departmentId);

**Code Snippet -> Bean Creation**

@SpringBootapplicaiton

@Bean

@LoadBalanced

public RestTemplate restTemplate() {

return new RestTemplate();

}

**Code Snippet -> Invoking Rest service**

@Autowired

private RestTemplate restTemplate;

Department department =restTemplate.getForObject("http://DEPARTMENT-SERVICE/departments/" + user.getDepartmentId(),Department.class);

## **Repository**

* **@Repository –** Declare as repository
* **JpaRepository –** extends for making as repository.

**Code Snippet ->**

@Repository

public interface DepartmentRepository extends JpaRepository<Department, Long> {

Department findByDepartmentId(Long departmentId);

}

## **Entity**

* **H2, jpa, Lombok –** Dependency for Database**s**
* **@Entity –** Table mapping
* **@Data – Lombok Getter and setter**
* **@AllArgsConstructor, @NoArgsConstructor -** Lombok constructor
* **@Id, @GeneratedType – Primary key**

**Code Snippet ->**

@Entity

@Data

@AllArgsConstructor

@NoArgsConstructor

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

**Application.yml/application.properties**

server:

port: 9001

spring:

application:

name: DEPARTMENT-SERVICE

# **Service Registry**

**Server application**

* **Eureka server –** Dependency for server.
* **Eureka Discover Clint -**Dependency for client.

**Code Snippet ->**

@SpringBootApplication

@EnableEurekaServer

**Application.yml/application.properties**

server:

port: 8761

http://localhost:8761

eureka:

client:

register-with-eureka: false

fetch-registry: false

**Client application (Eureka Discover client)**

**Code Snippet ->**

@SpringBootApplication

@EnableEurekaClient

**Application.yml/application.properties**

eureka:

client:

register-with-eureka: true

fetch-registry: true

service-url:

defaultZone:http://localhost:8761/eureka

Instance:

Hostname:localhost

# **Hystix Dashboard**

* **@EnableHystrixDashboard** – enabling dashboard application.
* http://apigateway/actuator/hystrix.stream

**Code Snippet ->**

@SpringBootApplication

@EnableHystrixDashboard

@EnableEurekaClient

**Application.yml/application.properties**

server:

port: 9295

spring:

application:

name: HYSTRIX-DASHBOARD

hystrix:

dashboard:

proxy-stream-allow-list: "\*"

**http://apigateway/actuator/hystrix.stream**

# **Api Gateway**

(Port will be common for all Services Registered in API Gateway,Hystirx will be enabled here )

* **Define Routes in application.yml for Routes**.
* **Filters for fall back url against circuit breaker**
* **Time out in Millisecond for timeout.**
* **Actuator for Hystrix dashboard streams**

**Code Snippet ->**

@SpringBootApplication

@EnableEurekaClient

@EnableHystrix

**Code Snippet Hystrix Fall back->**

@RestController

public class FallBackMethodController {

@GetMapping("/userServiceFallBack")

public String userServiceFallBackMethod() {

return "User Service is taking longer than Expected." +

" Please try again later";

}

@GetMapping("/departmentServiceFallBack")

public String departmentServiceFallBackMethod() {

return "Department Service is taking longer than Expected." +

" Please try again later";

}

}

**Application.yml/application.properties**

server:

port: 9191

spring:

application:

name: API-GATEWAY

cloud:

gateway:

routes:

- id: USER-SERVICE

uri: lb://USER-SERVICE

predicates:

- Path=/users/\*\*

filters: **<!—circuit breaker for fall back 🡪**

- name: CircuitBreaker

args:

name: USER-SERVICE

fallbackuri: forward:/userServiceFallBack

- id: DEPARTMENT-SERVICE

uri: lb://DEPARTMENT-SERVICE

predicates:

- Path=/departments/\*\*

filters:

- name: CircuitBreaker

args:

name: DEPARTMENT-SERVICE

fallbackuri: forward:/departmentServiceFallBack

**<!—4 second timeout 🡪**

hystrix:

command:

fallbackcmd:

execution:

isolation:

thread:

timeoutInMilliseconds: 4000

**<!—Actuators 🡪**

management:

endpoints:

web:

exposure:

include: hystrix.stream

eureka:

# **Config Server**

* **Git Url for config servers**
* **Git repo content having application.yml contains Eureka configuration for Each application.**
* **Client Application To be enabled with URL server app,in bootstrap.yml**

**Code Snippet ->**

@SpringBootApplication

@EnableEurekaClient

@EnableConfigServer

**Application.yml/application.properties**

Git repo content having application.yml contains Eureka configuration

server:

port: 9296

spring:

application:

name: CONFIG-SERVER

cloud:

config:

server:

git:

uri: https://github.com/shabbirdwd53/config-server

clone-on-start: true

**Client application (Spring config)**

**bootstrap.yml**

spring:

cloud:

config:

enabled: true

uri: <http://localhost:9296>

**Distributed Tracing**

* **Client application (Zipkinclient,sleuth) dependency should be added.**
* **Download** Zipkin.io (JAR) -> java -jar zipkin-serverdfdsf.jar run in local

**Client application (Zipkinclient,sleuth)**

spring:

application:

name: USER-SERVICE

zipkin:

base-url: <http://127.0.0.1:9411/> <! – default downloaded address 🡪

Service Name ,Trace ID,