Edge Server/API Gateway

Challenges

1. How do we maintain single entry point into microservices environment.
2. How do we routes based on custom requirements.
3. Cross Cutting Concerns

Why should we have to create separate edge server.

1. Request Validation
2. Include & Exclude list
3. Auth & Authrozatiion
4. Rate Limit
5. Dynamic Routing
6. Modify Request
7. Protocol Conversion
8. Exception Handling
9. Circuit Breaker

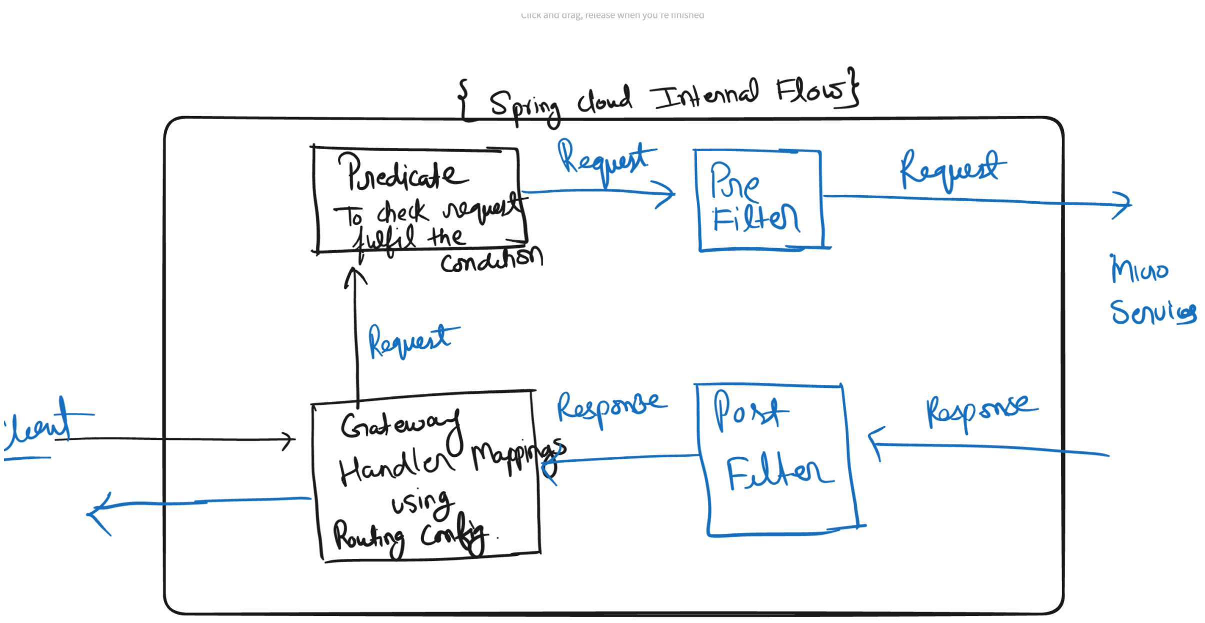
Spring Cloud Gateway[Reactive]

This project provides a libraries for building an API Gateway on top of Spring WebFlux or Spring WebMVC.

Spring Cloud Gateway aims to provide a simple, yet effective way to route to APIs and provide cross cutting concerns to them such as: security, monitoring/metrics, and resiliency.

Zuul is another option for api gateway.

Internal working of spring cloud



Click and drag, release when you're finished
{ Spring cloud Internal Flow }
Predicate
To check request fulfil the condition
Request
Pre Filter
Request
Micro Services
Request
lient
Gateway Handler Mappings using Routing Config.
Response
Post Filter
Response

Lets create api gateway.

1. Create spring boot project
2. Add dependencies
    a. Spring Cloud Gateway
    b. Eureka Discovery client
    c. acutator
    d. dev tools
3. Download and open with intellij

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

Check actuator urls for routes information

```
/actuator/gateway/routes
```

Make gateway to accept lower case services name

```
spring.cloud.gateway.discovery.locator.lowerCaseServiceId=true
```

Custom Routing in Cloud Gateway

create bean

```
@Bean
public RouteLocator routeLocator(RouteLocatorBuilder builder){
    return builder.routes()
    .route(p→ p.path("/elarn/category/**")
    .filters(
    f→ f.rewritePath("/elaern/category/(?<segment>.*)","/${segment}"))
    .uri("lb://category")
    )
}
```