**List of Configuration for Spring Boot Microservices**

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

Default Eureka Discovery Service URL

server.port=${PORT:0}

Assign Random Port Number to Microservice

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

enable discovery locator for API gateway so whenever request will come it will go to discovery service search for appropriate service with port number and map the URL and route to destination URL. This process will happen automatically

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

It allows you to locate your backend microservice in lower case whenever you are passing your backend microservice spring application name in API gateway URL

eureka.instance.prefer-ip-address=true

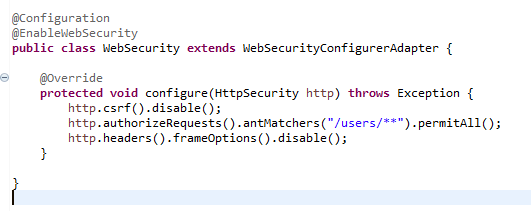
prefer ip address add this in your microservice so that api gateway can locate your microservice based on ip address

eureka.instance.instance-id=${spring.application.name}:${spring.application.instance\_id:${random.value}}

Assign Random value to your microservice instance assign random instance id and random port number to your microservice

Spring Security

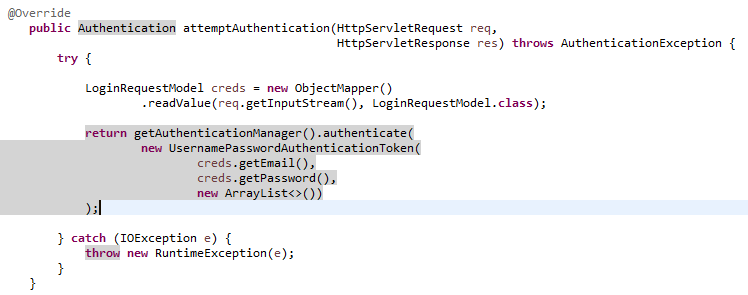
To permit all request for a particular URL you can use permitAll() method of Spring Security by extending WebSecurityConfigurerAdapter



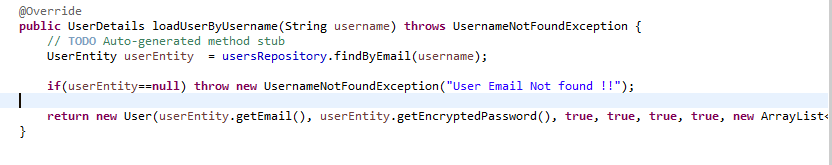
How to validate User using Spring Security

1. Create a Authentication Filter which extend UsernamePasswordAuthenticationFilter and override a method attemptAuthentication()

Get the value from HttpServletRequest like email and password and pass it to authenticationManager



1. AuthenticationManager will create AuthenticationManagerbuilder and internally it will call loadUserName() method it will fetch value from database we have to write custom logic to fetch details regrading the user from database



Understanding of concept

1. Internal Working of Spring Cloud API Gateway, Filters in api gateway, route, predicate,