Profiles in SpringBoot :: 6thApril video
Video topic: SpringSecurity and Junit with Mockito(Will be uploaded soon)

Today Evening Topic:
a. CloudConfig server(springprofiles)
b. CircuitBreaker(hsytrix circuit breaker)
c. Distubuted logging in Microservices

Tommo Last session
a. Morning Sesion[RedisCache integration]
b. Evening Session[Webflux:ReactiveProgramming]
c. Apache-Kafka Integration

Actuators

=> We are developing our applications using springboot and those applications will be deployed in servers.
=> After deploying our applications in server, those applciation will be used by the clients(users).
=> when our application is running in production, It is very important to monitor the application.
=> Monitoring the application corresponds to how the application is responding the clients,

What is the meaning of Monitoring the application?
a. HealthCheck
b. BeansCheck
c. configProps check
d. Heap dump
e. Thread dump(information about threads)
f. Http trace etc.....

To monitor our applications,Spring boot has provided actuators.
Actuators are used to provide "Production ready features" of our application.

Working with actuators

1. use the following dependancies in our application

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

2. Actuators provided several predefined endpoints to monitor our application
   a. health
   b. info
   c. beans[To know all the beans loaded in our application]
   d. heapdump
   e. Threaddump
   f. shutdown(It is a special endpoints)
   g. configProps
   h. mappings[To know all the url patterns of our application]

3. To access actuator endpoints we should use /actuator as prefix(It is introduced from SpringBoot2.X)
   eg: http://localhost:9999/actuator/health

```
4. In SpringBoot 1.x 'actuator' is optional in URL to access the endpoints.
5. In SpringBoot 2.x 'autuator' is compulsory in URL to access the endpoints.
6. By default 1 endpoints will be available(health)


Note: In order to include all the inbuilt endpoints
            management.endpoints.web.exposure.include=*
            management.endpoints.web.exposure.exclude=health,mappings,beans


heapdump   => It is used to download heap details, To analyse heapdump we use 
analyzer tools called MAT[Memory Analyzer Tool].
threaddump => It is used to give information about the threads available in our 
application.


shutdown
========
   It is a special endpoint.
   It is used to stop the application.
   This endpoint by default is in disable state.
   This endpoint is binded to HTTPPOST request method(we can send the request from 
browser due to security reasons)
            management.endpoint.shutdown.enabled=true

For any application we can send the request in the following ways
   POST => http://localhost:9999/actuator/shutdown   in postman tool then the 
application would be shutdown
                             |
                             |output
                             |
                  {
                         "message": "Shutting down, bye..."
                  }

SpringBoot Admin Server and Admin Client
========================================
  In Microservices architecture based project, we will have several services they 
are RestApi's.
  To monitor RestApi's we will enable and expose actuators endpoints
  If we have more no of services, it will be difficult to monitor and manage all 
our services.
  To overcome this problem,SpringBoot has provided "Admin Server" and "AdminClient" 
concept.
  If we use Admin Server,it will provide beautiful user interface to monitor and 
manage our RestApi's.

Note: Our RestApi's should be register with Admin Server then our RestApi's is 
called "Admin Client".


Steps to create AdminServer for our application
===============================================
  1. Create a SpringBoot application with the following dependencies
                  a. spring-boot-starter-web
                  b. spring-boot-starter-admin-server

  2. Configure @EnableAdminServer in SpringBoot starter class
  3. Configure embeded container port not in application.properties/yml file
  4. Run application


Steps to create AdminClient for our application
================================================
  1. Create a SpringBoot application with the following dependencies
                    a. spring-boot-starter-web
                    b. spring-boot-starter-admin-client
                    c. spring-boot-starter-actuator

  2. Configure below properties in application.properties/yml file
                    a. portno
                    b. applicationname
                    c. register with admin server
                    d. expose actuator endpoints

application.properties
======================
server.port= 1111
spring.application.name=CLIENT-1
spring.boot.admin.client.url=http://localhost:9999/
management.endpoints.web.exposure.include=*

  3. Create ReSt controller with required method
  4. Run the application and verify the AdminServer dashboard.

Note: ClientApplication should be displayed in the AdminServer DashBoard.


SpringCloud Config Server
=========================
 => Spring Cloud provided Config server
 => Config Server is used to externalize the configuration properties from our
application.

Scenario
      Microservice -------> Config Server -----> Git Repo
```