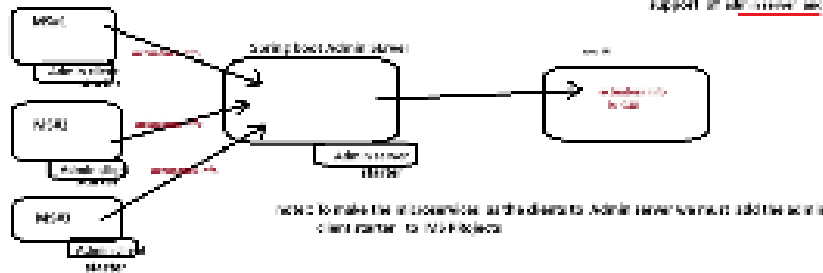


spring boot Admin server

→ Spring boot Admin server is a spring boot app that acts as proxy server for multiple micro services of the project. to collect and show spring boot instances in the GUI view.

note: for we have used annotations in every MDP project and we are able to open the application with each MDP project. If multiple Admin Servers are applied with annotations. In get there and we have in single MDP once we need to have the support of Admin server and Admin client.



→ instead of getting activation info about each MDP each instance of the assembly we can get the same information about all the instances and instances from single place called web UI through Spring boot Admin server.

we have 3 types of activation info from each MDP to get the info, and the thing is Spring boot Admin server proxy collecting their information and giving the info UI.

Example app

→ [Example app](#)

step1) Create Admin Server

step2) Create Spring boot admin server

1) Maven: Add ... Admin server (code version)

→ place Spring boot Admin Server on the top of admin client

spring boot Admin Server
spring boot Admin Client
spring boot Admin Server (code version)

profile: use that MDP (Spring boot Admin Server)

```

<code>
</code>

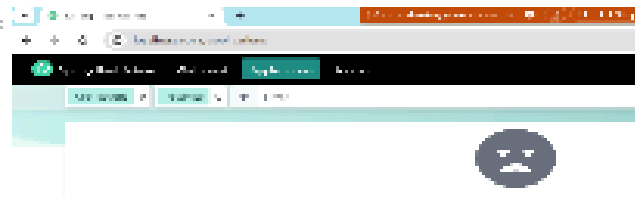
```

→ Run the Application and observe the Basic web UI

application properties

we have added Admin Server (Spring boot Admin Server)

Admin server and
spring boot Admin Server



step3) provide correct microservices by making clients to Admin server (spring boot Admin client)

(Take the copy from ... and paste it in the Admin client)

Make sure that these Microservices are
already added with spring boot Admin server

also, we can see that the following screen is added to the web UI in the Microservices Admin client

[Spring boot Admin Server \(code version 3.4.0\) / Spring boot Admin Server](#)

step4) add additional starter in the application and server file of the both MDP projects

→ To make the Admin server as Admin client
spring boot Admin client (code version 3.4.0) (code version 3.4.0)
→ To make the Admin client as Admin server
spring boot Admin server (code version 3.4.0) (code version 3.4.0)
→ To make the Admin client as Admin server
spring boot Admin server (code version 3.4.0) (code version 3.4.0)
→ To make the Admin client as Admin server
spring boot Admin server (code version 3.4.0) (code version 3.4.0)
→ To make the Admin client as Admin server
spring boot Admin server (code version 3.4.0) (code version 3.4.0)

step5) Run the Project in the following order

- 1) Admin server
- 2) Admin client
- 3) MDP1
- 4) MDP2

step6) Refresh the following components in the web UI of the Admin server

→ To make the Admin client as Admin server

→ To make the Admin client as Admin server

→ To make the Admin client as Admin server

→ To make the Admin client as Admin server

→ To make the Admin client as Admin server

Spring Boot Admin Server

=====

=====

=> Spring boot admin server is a spring boot app that acts proxy server for multiple micro services of the project to collect and show spring boot actuators

in the GUI env...

=> So far we have used actuators in every MS project and we are able to operate the actuators w.r.t each MS project if multiple MicroServices are applied with actuators to get them and use them in single GUI env.. we need to take the support of admin server and admin client

MS#1

actuators info

Spring boot Admin Server

Admin client

starter

MS#2

actuator info

Admin client

actuators info

Starter

MS#3

Admin client

starter

Admin server

starter

web UI

actuators info in GUI

note:: To make the microservices as the clients to Admin server we must add the admin client starter to MS Projects

each instance of MS separately we can get the

=> Instead of getting actuators info about each MS

same information about all the MSs and instances from single place called web UI through Spring Boot Admin server

instance of

=> web UI gets actuators info from each MS or each MS separately.. only the thing is

as

Spring boot admin acts proxy collecting that information and giving to web UI

Example App

=====

step1) Create Eureka Server

step2) Create Spring boot admin server

starters :: web, admin server (codecentric)

=> place `@EnableAdminServer` on the top of main class

`@SpringBootApplication`

`@EnableAdminServer`

```
public class BootMsProj11AdminServerApplication {
```

```
    application.properties
```

```
    spring.application.name=BootMSProj11-AdminServer
```

```
    # admin server port server.port=9091
```

```
    public static void main(String[] args) {
```

```
        Spring Boot Admin
```

```
        * +
```

```
        SpringApplication.run(BootMsProj11AdminServerApplication.class, args);
```

```
    }
```

```
    c
```

```
    O localhost:9091/applications
```

```
}
```

Spring Boot Admin

Wallboard

Applications

Journal

Applications o

Instances U Y Filter

=> **Run the Application and observe the Basic web UI**

You're sharing your screen

):

Q 00

Stop

No applications registered.

step3)

(Take the copy from any old projects

Develop one or microServices by making clients to Admin server (nothing but admin clients) (add Adminclient as the extra starter)

Make sure that these MicroServices are already added with Spring boot Actuator starters

:)

step4) add additional entries in the application.properties file of the both MS projects

#Connect to Admin server as admin client spring.boot.admin.client.url=http://localhost:9091 #activate all the actuators

management.endpoints.web.exposure.include=*

management.endpoint.health.show-details-always

management.endpoint.env.enabled=true management.endpoint.env.show-values-always

step5) Run the Projects in the following order

i) Eureka server

ii) Admin server

iii) MS#1

vi) MS#2

step6) Perform the following operations on webUI of the admin server

a) see the active applications count

http://localhost:9091/applications

b) add ur choice properoties to get their metrics using metrics actuators

Admin server web ui home page ---> wallboard ---> choose ms or ms instance select metrics ---->

Metrics

http.server.requests

http.server.requests

method:GET

uri:/customer-api/report

outcome:SUCCESS

status:200

exception

method

GET

error

uri

/customer api/report

outcome

SUCCESS

status

200

note:: make sure that the following property is added to the pom.xml file MS projects (admin clients)

<spring-boot-admin.version>3.4.5</spring-boot-admin.version>

COUNT

TOTAL_TIME

MAX

43

0.2883377

0

Add Metric