https://github.com/karrasankar158

ADMIN CLIENT AND SERVER(UI)+(ACTUATOR SERVICES)

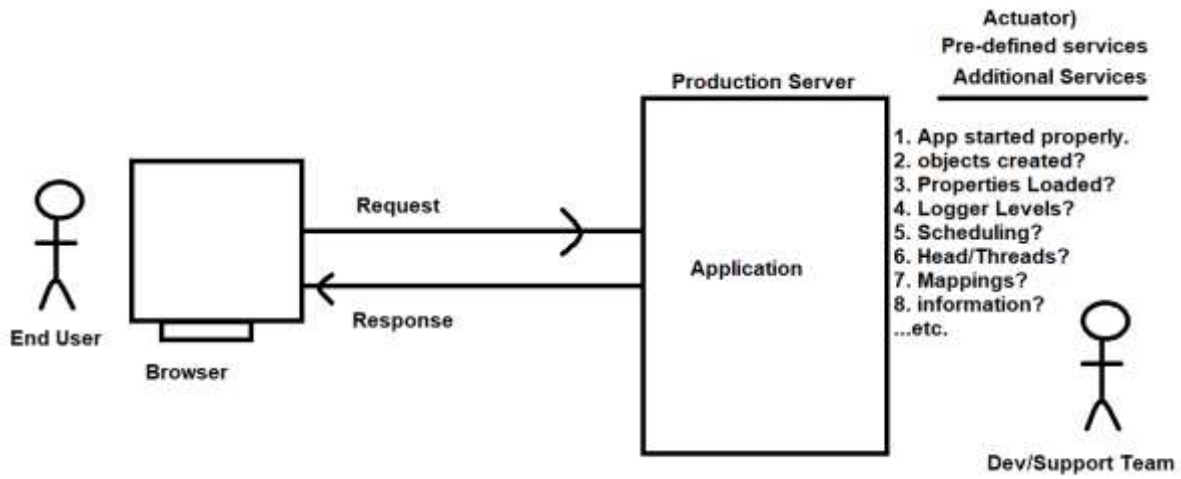
=>Actuator is used to Monitor your application/microservices using endpoints.

⇨ Admin server is used for Monitoring your apps.

⇨ Our application almost works fine in Developer machine. But once it is moved to Production server(Actual Server Deployed to give service to endusers).
⇨ To find problems and monitor issues we have lot of tools at Production server.
⇨ One of such service/tool is “Actuator”.
⇨ Admin Server add dependency in only one application.
⇨ Admin client add dependency in remaining all microservices.
⇨ Codecentric vendor has provided this Open Source API to implement Admin Client/Server.

Q) What is Actuator?

A) Production ready endpoints.

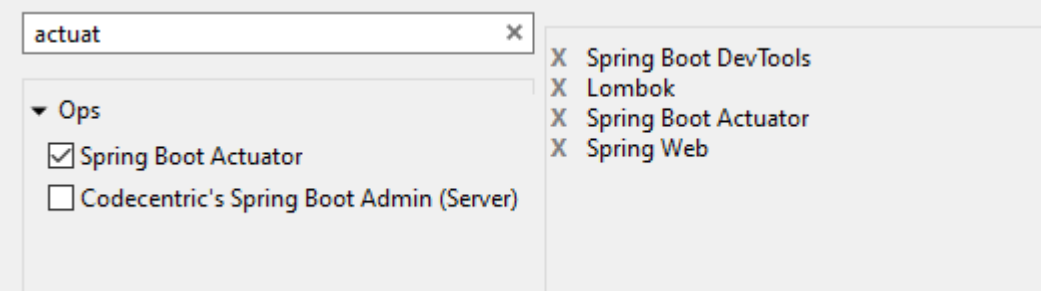


Endpoint: It is a pre-defined service that is used to find/execute a work like,

a. App started properly or not?
b. Objects created or not?
c. Properties data loaded or not?
…etc.

⇨ These endpoints can be used with any Spring Boot application(web dependency) required.
----------manual process----------

Dependencies:



Step1: Application.properties:

```
# Server port
server.port=9800
```

Step2: ActuatorRestController:

```
package com.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/app")
public class ActuatorRestController {
	/**Actuator: http://localhost:9800/actuator
	 * URL: http://localhost:9800/app/message?message=welcome to rest api
	 * @param message
	 * @return
	 */
	@GetMapping("/message")
	public String showMessage(@RequestParam String message) {
		return message;
	}

}
```

Step3: Starter class:

```
package com;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ActualTestApplication {

	public static void main(String[] args) {
		SpringApplication.run(ActualTestApplication.class, args);
	}

}
```

Step4: Run app:

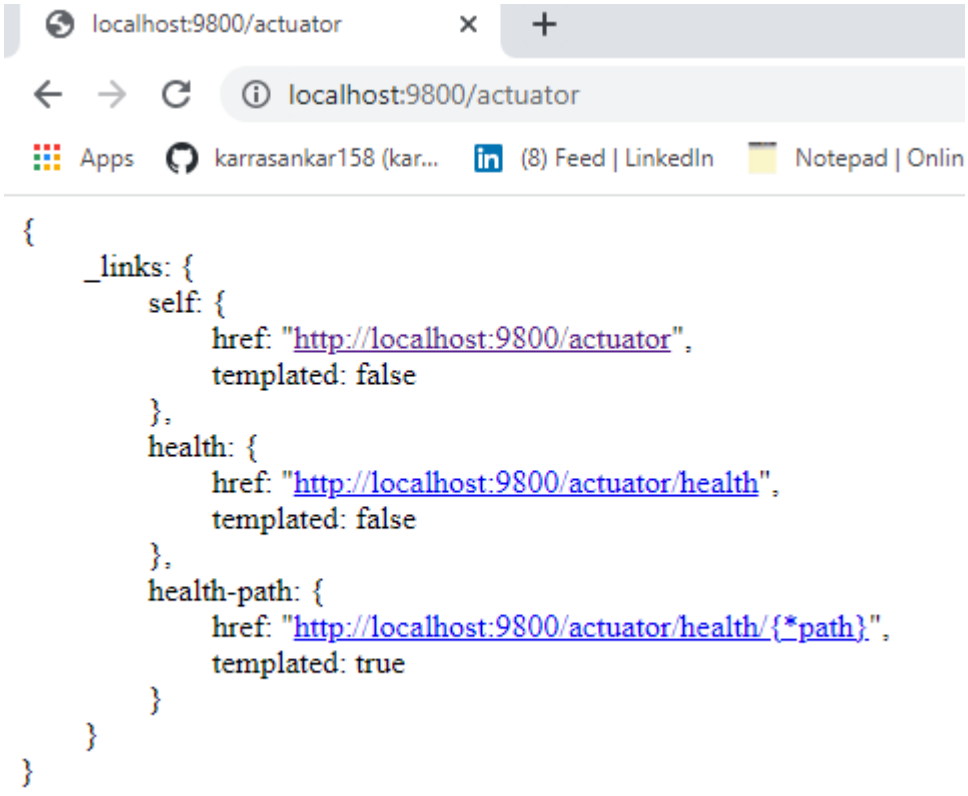
Q) which endpoint by default exposes spring boot2.5.8v actuator?

A) /actuator

Q) which endpoint by default exposes spring boot 2.1.3 actuator?

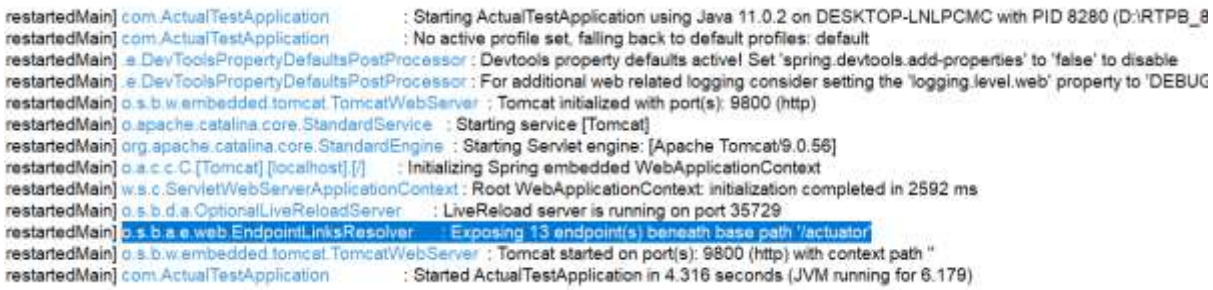
A) /actuator, /info

/actuator end point url: http://localhost:9800/actuator:



Q) How to expose all spring boot actuator endpoints?

A) management.endpoints.web.exposure.include=*



Q) How many pre-defined endpoints are there in Spring Boot Actuator?

A) 13+base /actuator.

http://localhost:9800/actuator/health

{

- status: "UP"

}

Q) Which endpoint is used to check application up or down?

A) http://localhost:9800/actuator/health

Q) All actuator production ready end points?

A)

1. /actuator
2. /actuator/beans
3. /actuator/caches
4. /actuator/health
5. /actuator/info
6. /actuator/conditions
7. /actuator/configprops
8. /actuator/env
9. /actuator/loggers
10. /actuator/heapdump
11. /actuator/threaddump
12. /actuator/metrics
13. /actuator/scheduledtasks
14. /actuator/mappings

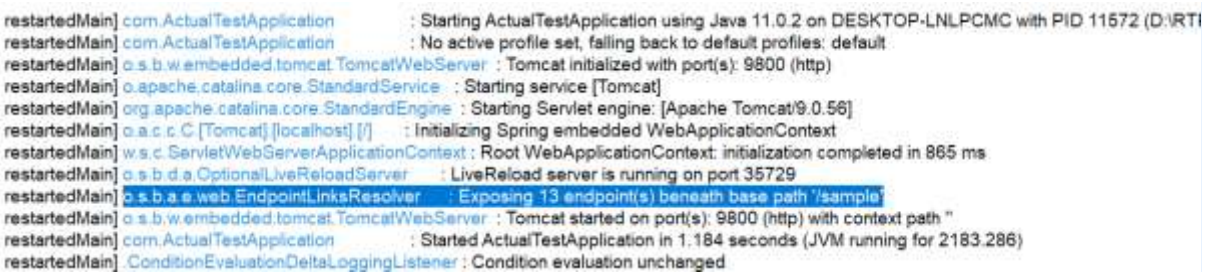
Q) How to expose specific spring boot actuator endpoints?

A) management.endpoints.web.exposure.include=bean,info

Q) can we modify basePath of Actuator services?

A) yes, by using key management.endpoints.web.base-path=/sample

Now url is: http://localhost:9800/sample



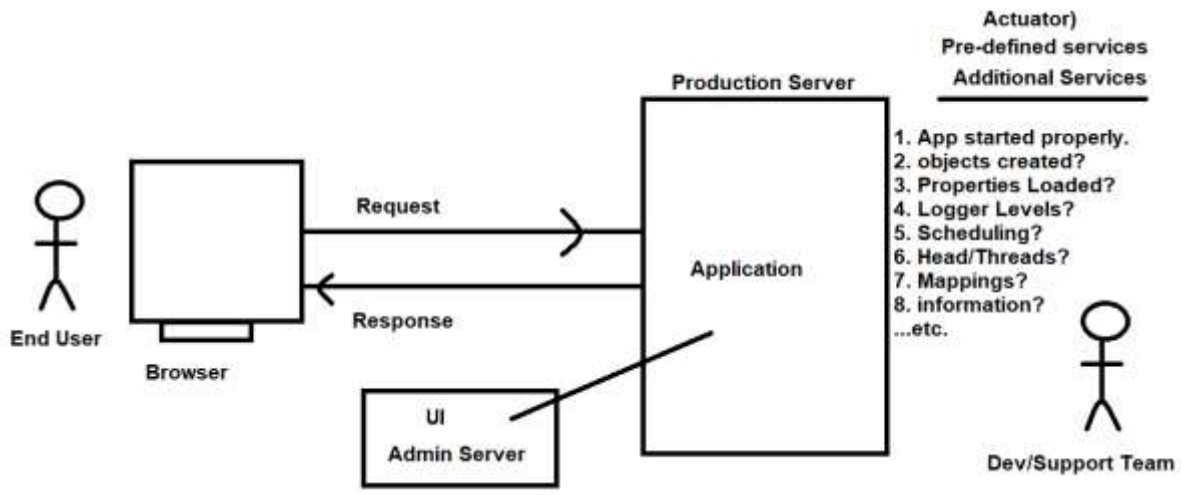
Few application.properties:

```
# Server port
server.port=9800

#Expose all spring boot actuator endpoints
management.endpoints.web.exposure.include=*

#actuator base path change
management.endpoints.web.base-path=/sample

#Expose specific spring boot actuator endpoints
#management.endpoints.web.exposure.include=bean,info

# to see more health endpoint details
management.endpoint.health.show-details=always

# to disable particular endpoint
management.endpoint.health.enabled=false

#exclude specific spring boot actuator endpoints
management.endpoints.web.exposure.exclude=bean,health
```

----------Automated process----------

In realtime, there can be multiple microservices running. If we check all endpoints manually then it takes lot of time even complex process. So, Spring Boot has provided ADMIN SERVER UI.

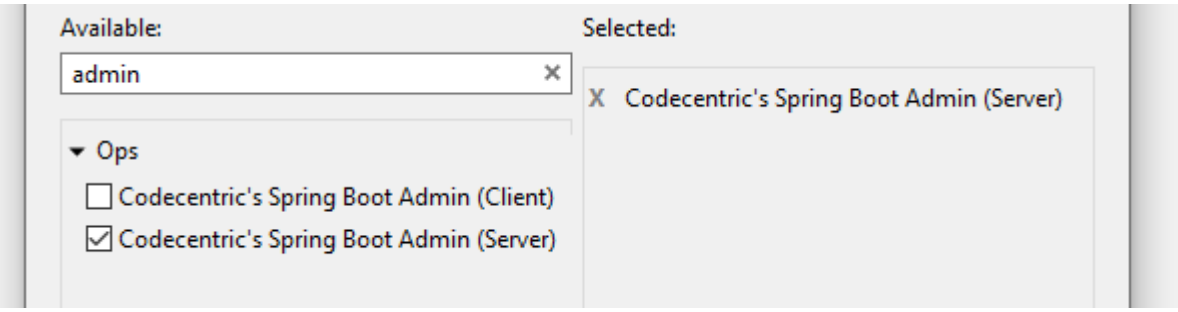
This Admin Server UI gets all microservices Actuator details into one place and display as UI (Easy observation).



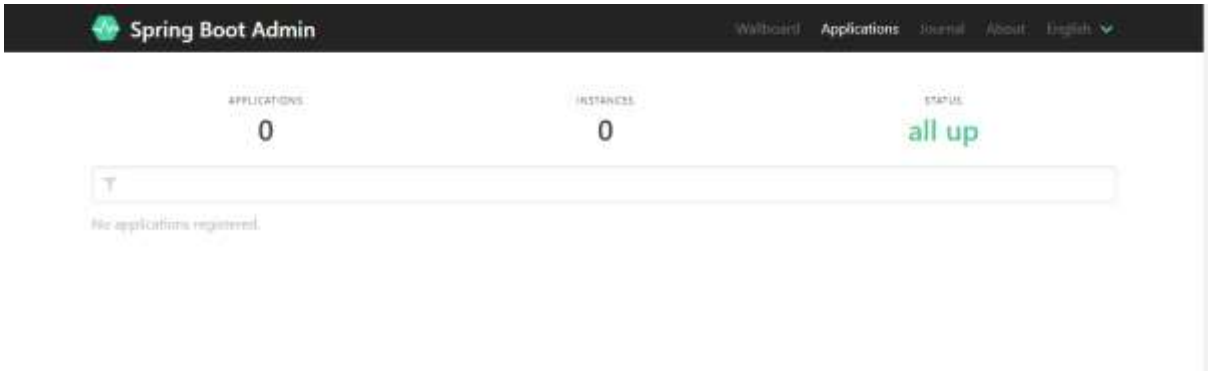
Admin Server: it is central Server for All microservices used to execute Actuator Services and gets result into one UI Format.

⇨ Codecentric has developed this API as opensource and Integrated by Spring boot.

Dependencies:



Admin Server url: http://localhost:9999/applications



application.properties:

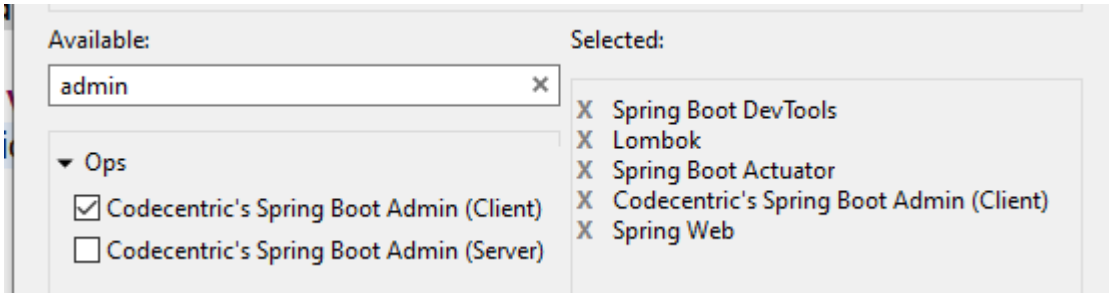
#Recommended port number
server.port=9999

Starter class:

```
package com;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import de.codecentric.boot.admin.server.config.EnableAdminServer;

@SpringBootApplication
@EnableAdminServer
public class AdminServerApplication {

	public static void main(String[] args) {
		SpringApplication.run(AdminServerApplication.class, args);
	}

}
```

For every micro service application:
Step1: add two dependencies

1. Actuator
2. Admin Client



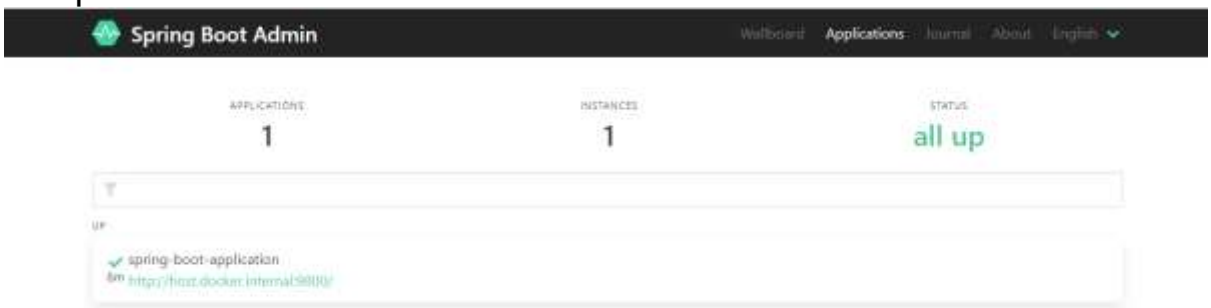
Step2: Activate all end points

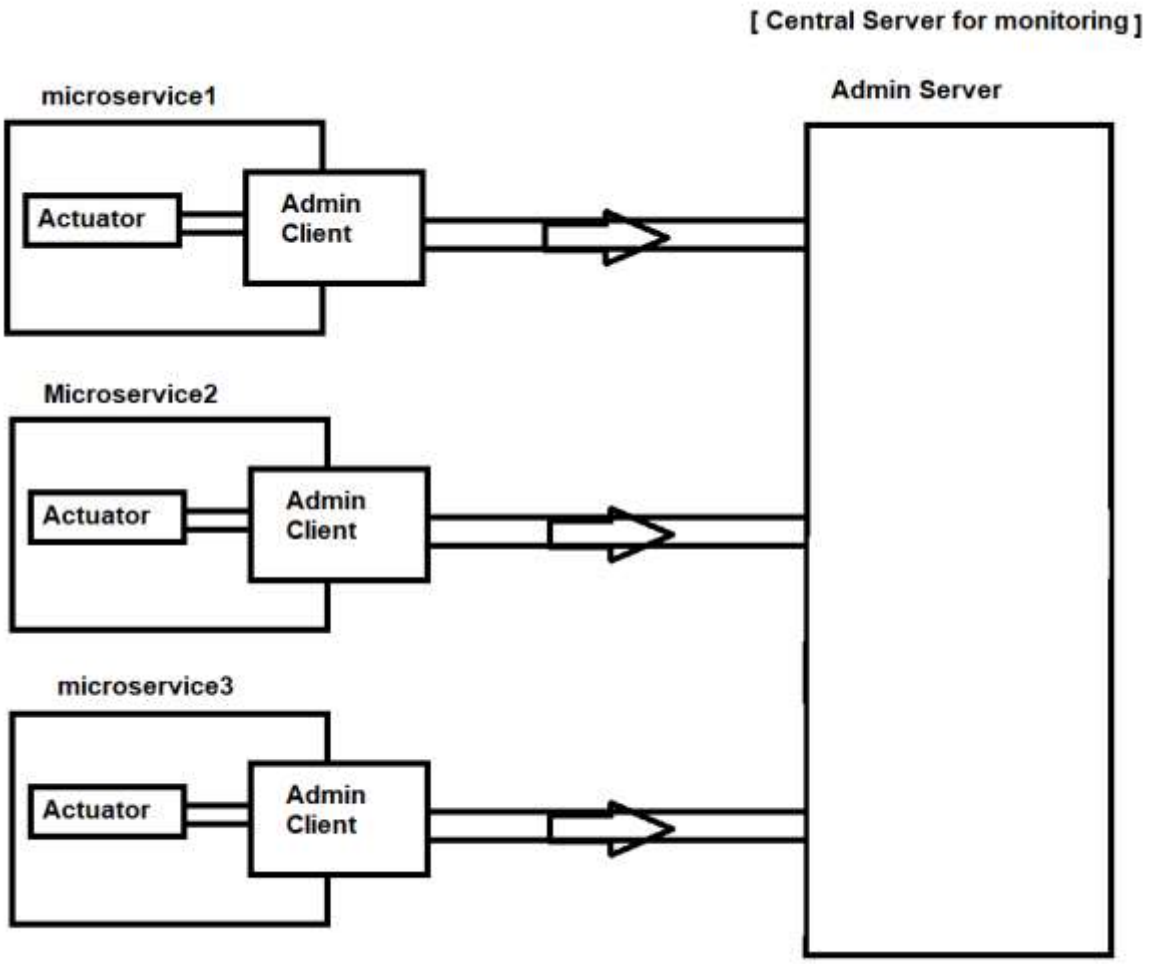
```
#Expose all spring boot actuator endpoints
management.endpoints.web.exposure.include=*
```

Step3: Link to admin server URL.

```
#Admin Server Connecting URL
spring.boot.admin.client.url=http://localhost:9999
```

sample Admin Server screen:





Q) How can we create multiple instances?
A) Just run application multiple times with different port numbers.
Q) How Admin server is different from eureka server?
A) Admin server is checking all pre-defined services like
Beans- objects created or not?, key-value loaded or not?, ….etc.

Eureka Server is store Microservices instance data when gateway request comes select an instance that has less load factor and return same to gateway.

----------THE END----------

https://github.com/karrasankar158