**Spring Boot Admin - Admin UI For Monitoring Microservices**

In this post we develop a Spring Boot Admin Server and Client module.

What is Spring Boot Admin? Need for it?

Spring Boot provides actuator endpoints to monitor metrics of individual microservices. These endpoints are very helpful for getting information about applications like if they are up, if their components like database etc are working good. But a major drawback or difficulty about using actuator enpoints is that we have to individually hit the enpoints for applications to know their status or health. Imagine microservices involving 50 applications, the admin will have to hit the actuator endpoints of all 50 applications. To help us deal with this situation, we will be using open source project located at <https://github.com/codecentric/spring-boot-admin>.  
Built on top of Spring Boot Actuator, it provides a web UI to enable us visualize the metrics of multiple applications.

The implementation part is divided into 2 parts-

* Develop Spring Boot Admin Module
* Modify code of the microservices developed in our previous tutorial to act as spring boot admin client. We will be modifying the [Microservice Registration and Discovery with Spring cloud using Netflix Eureka](https://www.javainuse.com/spring/spring_eurekaregister4).

Spring Boot Admin Module

Maven Project will be as follows-

In the Maven we need the spring boot WebSocket dependency.Maven will be as follows-

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

**<dependency>**

**<groupId>de.codecentric</groupId>**

**<artifactId>spring-boot-admin-server-ui-login</artifactId>**

**<version>1.5.1</version>**

**</dependency>**

**<dependency>**

**<groupId>de.codecentric</groupId>**

**<artifactId>spring-boot-admin-server</artifactId>**

**<version>1.5.1</version>**

**</dependency>**

**<dependency>**

**<groupId>de.codecentric</groupId>**

**<artifactId>spring-boot-admin-server-ui</artifactId>**

**<version>1.5.1</version>**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

Next we create the SpringBoot Bootstrap class. Since using the Spring Boot Admin we will be accessing sensitive information, we will also be configuring spring security here.

package com.javainuse;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

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

@EnableAdminServer

@Configuration

@SpringBootApplication

public class SpringBootAdminApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootAdminApplication.class, args);

}

@Configuration

public static class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http.formLogin().loginPage("/login.html").loginProcessingUrl("/login").permitAll();

http.logout().logoutUrl("/logout");

http.csrf().disable();

http.authorizeRequests().antMatchers("/login.html", "/\*\*/\*.css", "/img/\*\*", "/third-party/\*\*").permitAll();

http.authorizeRequests().antMatchers("/\*\*").authenticated();

http.httpBasic();

}}

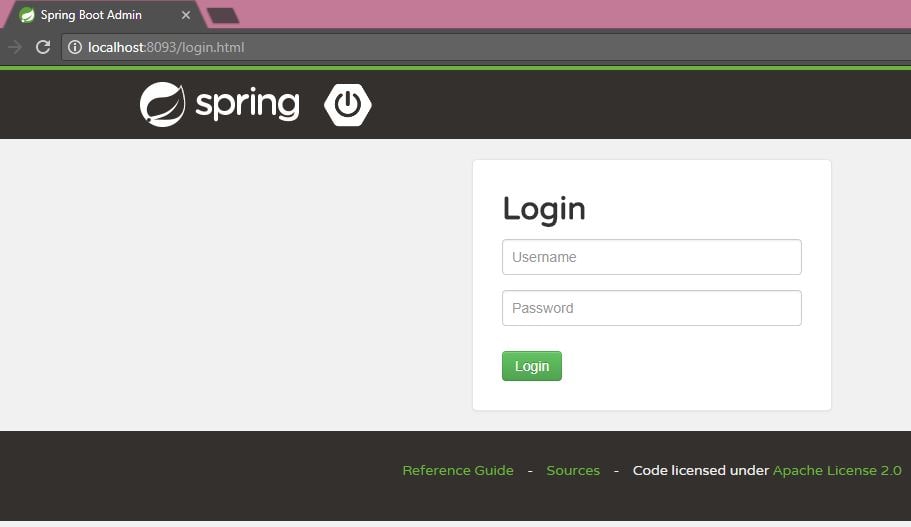
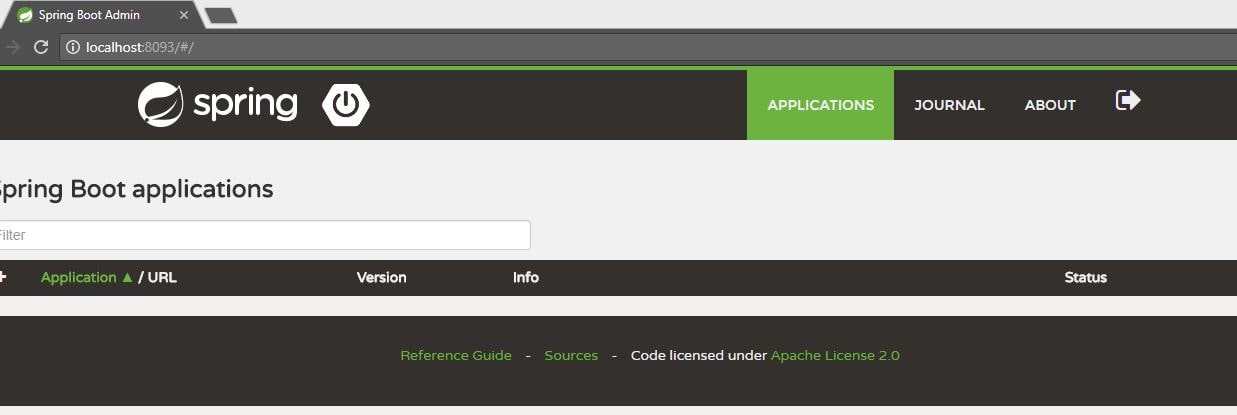
The properties file will be as follows-

spring.application.name=Boot-Admin

server.port=8093

security.user.name=admin

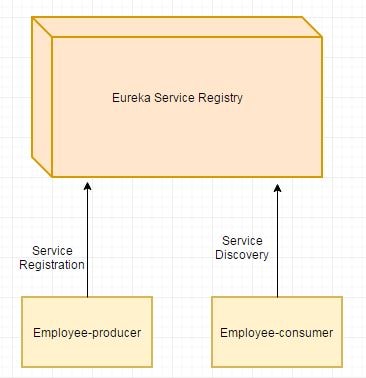
security.user.password=admin

These are the only changes required for the spring boot admin module. Start the application and got to localhost:8093  
  
Enter the username and password as admin and admin respectively.  


Spring Boot Admin Client Modules

In previous tutorial [Microservice Registration and Discovery with Spring cloud using Netflix Eureka](https://www.javainuse.com/spring/spring_eurekaregister4) we had created three modules.

* Eureka Server
* Employee Producer
* Employee Consumer

  
We fill first add the spring boot actuator and the spring boot admin dependencies in all the three modules. So add the following dependencies in all 3 modules.

<dependency>

<groupId>de.codecentric</groupId>

<artifactId>spring-boot-admin-starter-client</artifactId>

<version>1.5.1</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

Next in the application.properties of all the 3 modules add the following properties-

spring.boot.admin.url=http://localhost:8093

spring.boot.admin.username=admin

spring.boot.admin.password=admin

These are the only changes required. Now start all the three applications. They will be reflected in the spring boot admin UI.  
