# admin\_server for monitoring

admin-server + notification service

dependency for server -->

<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>

Add @EnableAdminServer on top of the class which contains main method.

@EnableAdminServer

@SpringBootApplication

public class SpringBootAdminApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootAdminApplication.class, args);

}

}

Admin Configuration file for monitoring annotated with @Configuration.

@Configuration

public class AdminConfiguration 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();

}

}

Admin Configuration file for notification when any service will go down or stopped , it will notify to registered user via mail.

@Configuration

@EnableScheduling

public class NotificationService {

@Autowired

private Notifier delegate;

@Bean

public FilteringNotifier filteringNotifier() {

return new FilteringNotifier(delegate);

}

@Bean

@Primary

public RemindingNotifier remindingNotifier() {

RemindingNotifier notifier = new RemindingNotifier(filteringNotifier());

notifier.setReminderPeriod(TimeUnit.SECONDS.toMillis(10));

return notifier;

}

@Scheduled(fixedRate = 1\_000L)

public void remind() {

remindingNotifier().sendReminders();

}

@Bean

public LoggingNotifier loggerNotifier() {

return new LoggingNotifier();

}

}

properties file :

spring.application.name=Admin-Server

server.port=1111

management.security.enabled=false

security.user.name=admin

security.user.password=admin123

spring.mail.host=--------

spring.mail.username=----------

spring.mail.password=--------

spring.mail.port=-------

spring.boot.admin.notify.mail.to=email@example.com

spring.boot.admin.notify.mail.enabled=true

spring.boot.admin.notify.mail.ignore-changes=UNKNOWN:UP

spring.boot.admin.notify.mail.from= email@example.com

spring.boot.admin.notify.mail.text = Hi User\_Name, \n\n The Service is goes down or stopped please insure that it is up \n\n Regards\n Sender\_Name

spring.mail.default-encoding=UTF-8

for client end dependency -->

<dependency>

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

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

</dependency>

<dependency>

<groupId>de.codecentric</groupId>

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

<version>1.5.1</version>

</dependency>

// Optional dependency

<dependency>

<groupId>org.jolokia</groupId>

<artifactId>jolokia-core</artifactId>

</dependency>

Jolokia library is used for more advanced features like JMX mbeans and log level management.

Client Properties file : client username & password should be match with

admin\_server.

server.port=7705

spring.application.name= application\_name

spring.boot.admin.url=http://localhost:1111 // admin server running url

management.security.enabled=false

spring.boot.admin.username=admin // admin server user\_name

spring.boot.admin.password=admin123 // admin server password