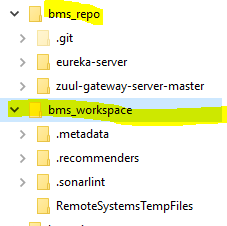
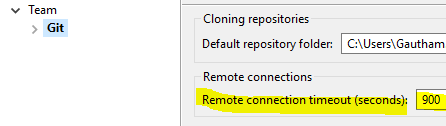
1. Main repository: <https://github.com/nithinsreeraj/bookmyshow>
2. Have one folder as your local repository and have use another location as the eclipse workspace.



1. Git bash inside the repo folder(bms\_repo) and execute the following:
   1. git init
   2. git remote add main <https://github.com/nithinsreeraj/bookmyshow>
   3. git pull main master
   4. git config --global core.autocrlf true
2. Launch the eclipse workspace and import each of the pulled project as “existing maven project”.
3. Verify whether the java code formatter is set to “Eclipse built in” for all our projects.
   1. windows -> preferences
   2. ->java -> code style -> formatter
   3. ->configure project specific settings
   4. -> (select the microservice project) ->OK
   5. -> check "enable project specific settings"
   6. ->choose "eclipse built in" as the active profile ->apply
4. Increase the remote connection timeout (IF you are using Eclipse to push/pull etc.)
   1. Windows -> Preferences
   2. Team -> Git

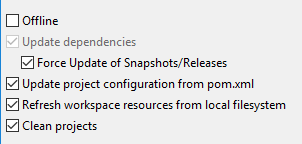


WHEN CREATING A NEW PROJECT

1. Make sure to use the spring snapshot "2.1.2(snapshot)" when creating the project from start.spring.io
2. Extract the zip to repo(bms\_repo)
3. To push your changes
   1. Git bash @ bms\_repo
   2. git add <folder-name>/\*
   3. git commit –m “<commit-message>”
   4. git push main master
4. Pull all the changes from the main before performing a push.

If there are any errors after importing a maven project, then try the following method:

* 1. right click on project
  2. -> maven
  3. ->update project
  4. ->check "force update of snapshots/releases" -> OK



Running Kafka server

Inorder to run kafka server we need to first run zookeeper and then run kafka server.

Please make sure that environment variable JAVA\_HOME is configured with path C:\Program Files\Java\jdk1.8.0\_144

1. To run zookeeper

checkout folder kafka\_2.11-2.1.0 and copy to C:\

Open CMD and navigate to C:\kafka\_2.11-2.1.0

Type the following command to run zookeeper

.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

1. To run kafka

open another CMD and navigate to C:\kafka\_2.11-2.1.0

type the following command

.\bin\windows\kafka-server-start.bat .\config\server.properties

1. Add topic to kafka

The topics “zipkin” and “notification” where already added in kafka.

open another CMD and navigate to C:\kafka\_2.11-2.1.0

Make sure the topics are available by typing following command after running zookeeper and kafka

.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic zipkin

.\bin\windows\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic notification

Running zipkin server

Open CMD and navigate to folder containing

Type command

set KAFKA\_BOOTSTRAP\_SERVERS=localhost:9092

To start zipkin server type

java -jar zipkin-server-2.11.12-exec.jar

Dependencies for zipkin and kafka

Add the following dependencies in all service to support zipkin and kafka

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-sleuth</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-zipkin</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.kafka</groupId>

<artifactId>spring-kafka</artifactId>

</dependency>

To support tracing in zipkin add the following method in service’s main class

@Bean

**public** Sampler defaultSampler() {

**return** Sampler.***ALWAYS\_SAMPLE***;

}

Sending message to notification service

Notification service is reading notification message from kafka bus. So all other service should push the notification message to kafka as JSON.

Open cmd and navigate to kafka folder and run following command to view “notification” topic

.\bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic notification --from-beginning

In the service create a class Message with following attributes

1. userId
2. msg

Create a configuration class KafkaConfiguration as follows

@Configuration

**public** **class** KafkaConfiguration {

@Bean

**public** ProducerFactory<String, Message> producerFactory() {

Map<String, Object> config = **new** HashMap<>();

config.put(ProducerConfig.***BOOTSTRAP\_SERVERS\_CONFIG***, "localhost:9092");

config.put(ProducerConfig.***KEY\_SERIALIZER\_CLASS\_CONFIG***, StringSerializer.**class**);

config.put(ProducerConfig.***VALUE\_SERIALIZER\_CLASS\_CONFIG***, JsonSerializer.**class**);

**return** **new** DefaultKafkaProducerFactory<>(config);

}

@Bean

**public** KafkaTemplate<String, Message> kafkaTemplate() {

**return** **new** KafkaTemplate<>(producerFactory());

}

}

To push a notification message to kafka, do the following

@Autowired

**private** KafkaTemplate<String, Message> kafkaTemplate;

kafkaTemplate.send("notification", **new** Message(1, "Hello"));

Follow the link to know more

<https://www.youtube.com/watch?v=NjHYWEV_E_o>

To add Authorization for service

Add following dependency in the service

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.micro</groupId>

<artifactId>auth-config-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

Add following class in the service

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Bean;

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

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

**import** org.springframework.security.config.http.SessionCreationPolicy;

**import** org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

**import** com.micro.authconfigservice.security.JwtConfig;

**import** com.micro.authconfigservice.security.JwtTokenAuthenticationFilter;

@EnableWebSecurity

**public** **class** SecurityTokenConfigurerAdapter **extends** WebSecurityConfigurerAdapter {

@Autowired

**private** JwtConfig jwtConfig;

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.***STATELESS***)

.and().exceptionHandling().authenticationEntryPoint((req, rsp, e) -> rsp.sendError(HttpServletResponse.***SC\_UNAUTHORIZED***))

.and().addFilterAfter(**new** JwtTokenAuthenticationFilter(jwtConfig), UsernamePasswordAuthenticationFilter.**class**)

.authorizeRequests().anyRequest().authenticated();

}

@Bean

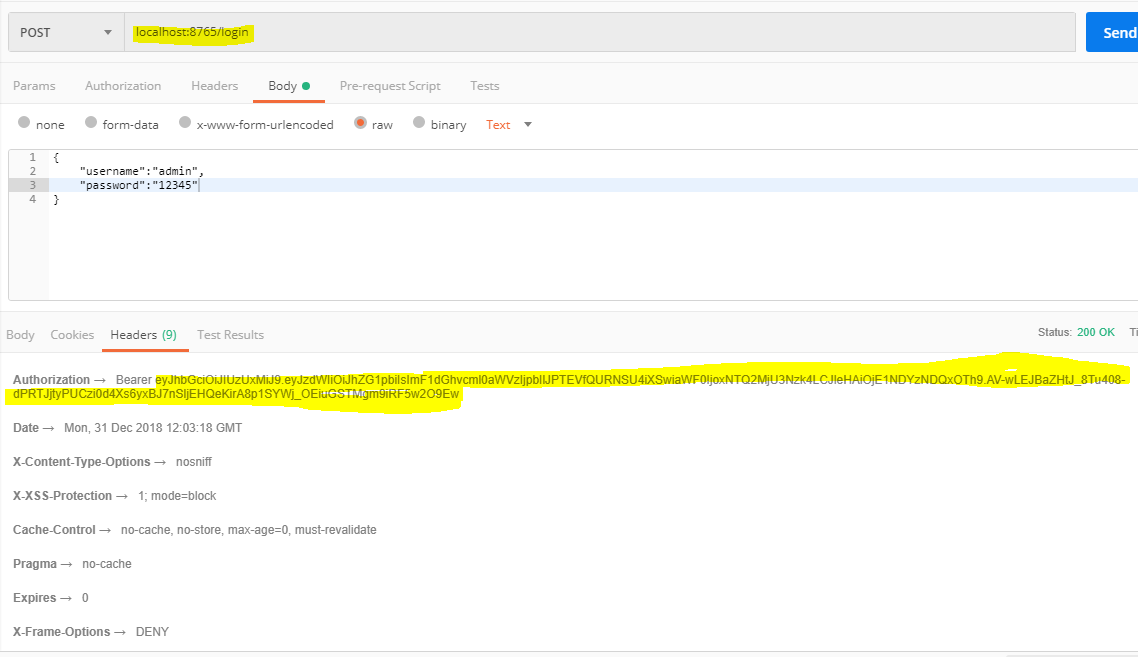
**public** JwtConfig jwtConfig() {

**return** **new** JwtConfig();

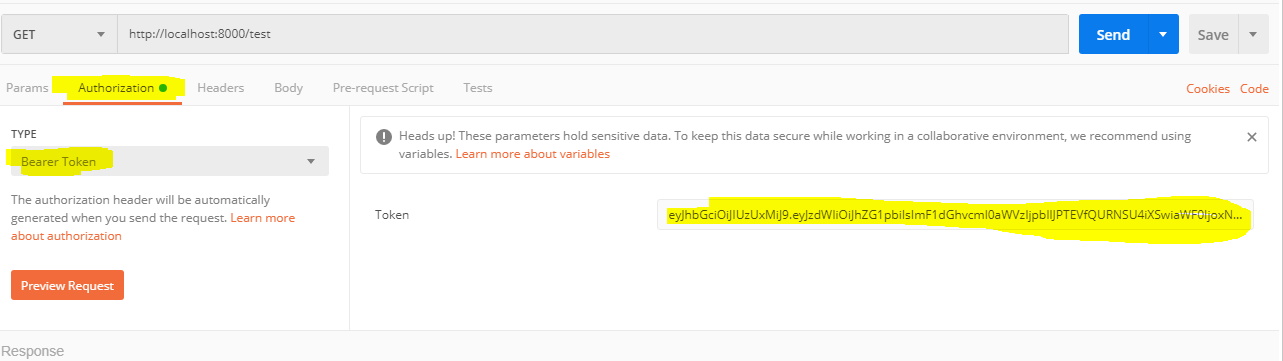
}

}

Before accessing your service run postman and do a post request with following



Copy the bearer token and add it to your service’s GET/POST request authorization



For transmitting token in inter-service call add the following classes in the consumer service

1. Request context

**public** **class** RequestContext {

**public** **static** **final** String ***REQUEST\_HEADER\_NAME*** = "Authorization";

**private** **static** **final** ThreadLocal<RequestContext> ***CONTEXT*** = **new** ThreadLocal<>();

**private** String token;

**public** **static** RequestContext getContext()

{

RequestContext result = ***CONTEXT***.get();

**if** (result == **null**) {

result = **new** RequestContext();

***CONTEXT***.set(result);

}

**return** result;

}

**public** String getToken() {

**return** token;

}

**public** **void** setToken(String token) {

**this**.token = token;

}

}

1. Request filter

**import** java.io.IOException;

**import** javax.servlet.Filter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.http.HttpServletRequest;

**public** **class** RequestFilter **implements** Filter {

@Override

**public** **void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

**throws** IOException, ServletException {

HttpServletRequest httpServletRequest = (HttpServletRequest) request;

String token = httpServletRequest.getHeader(RequestContext.***REQUEST\_HEADER\_NAME***);

**if** (token == **null** || token.equals("")) {

**throw** **new** IllegalArgumentException("Can't retrieve JWT Token");

}

RequestContext.*getContext*().setToken(token);

chain.doFilter(request, response);

}

}

1. Rest Template Interceptor

**import** java.io.IOException;

**import** org.springframework.http.HttpRequest;

**import** org.springframework.http.client.ClientHttpRequestExecution;

**import** org.springframework.http.client.ClientHttpRequestInterceptor;

**import** org.springframework.http.client.ClientHttpResponse;

**public** **class** RestTemplateInterceptor **implements** ClientHttpRequestInterceptor {

@Override

**public** ClientHttpResponse intercept(HttpRequest request, **byte**[] body, ClientHttpRequestExecution execution)

**throws** IOException {

String token = RequestContext.*getContext*().getToken();

request.getHeaders().add(RequestContext.***REQUEST\_HEADER\_NAME***, token);

**return** execution.execute(request, body);

}

}

Also add the following method in the service’s application class

@Bean

**public** FilterRegistrationBean getFilterRegistration()

{

FilterRegistrationBean registration = **new** FilterRegistrationBean();

registration.setFilter(**new** RequestFilter());

registration.addUrlPatterns("/\*");

registration.setName("requestFilter");

**return** registration;

}

Before requesting a service using RestTemplate, do the following

RestTemplate restTemplate = **new** RestTemplate();

List<ClientHttpRequestInterceptor> interceptors = restTemplate.getInterceptors();

interceptors.add(**new** RestTemplateInterceptor());

restTemplate.setInterceptors(interceptors);