Multi module project in maven

Create both below

WedServer

WebServerClient

And deploy in server

Create a maven project

Group id :org.develop

Artifact id: wheatherforcast

Packaging : pom (virtual grp of project)

Finish

Now to Add mutlti module projects

Now Maven Module in maven menu

Create module name add parent project in browse

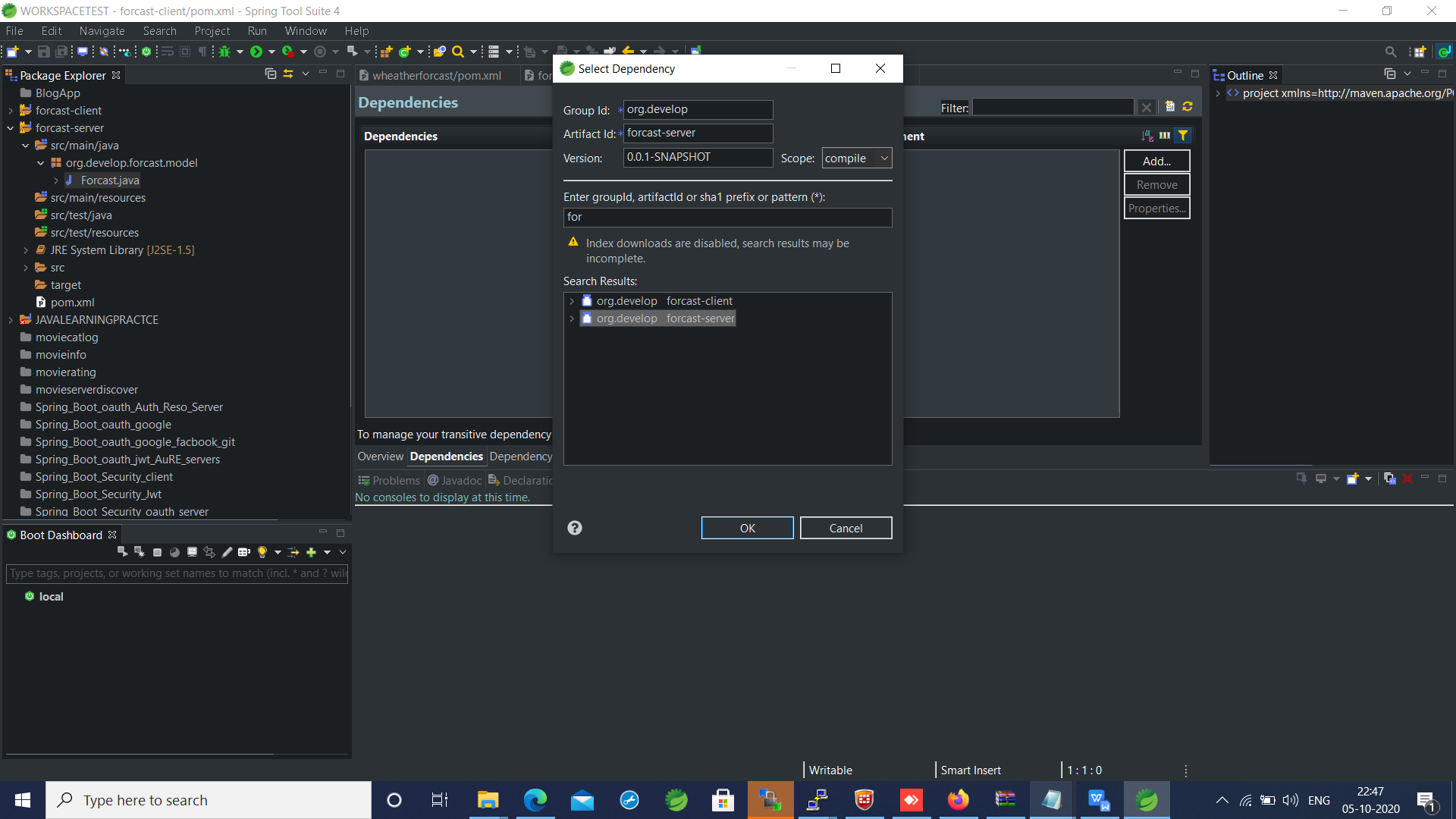
Create forcast-server : maven module project

Crate forcast-cleint : maven module project

In forcast-cleint add dependency of forcast-server

In forcast-client.pom in depedncies

This is noting but I am adding that jar file to to another project



So one project depend on another

This is to add dedency of project to another

Same USIng commands

#### Step 1 : Create Parent project

|  |
| --- |
| mvn archetype:generate -DgroupId=com.websystique.multimodule -DartifactId=parent-project |

#### Step 3 : Create sub-modules

Via command line, navigate to the directory of parent-project, and fire following commands

|  |
| --- |
| cd parent-project  mvn archetype:generate -DgroupId=com.websystique.multimodule  -DartifactId=model-lib  mvn archetype:generate -DgroupId=com.websystique.multimodule  -DartifactId=webapp1  mvn archetype:generate -DgroupId=com.websystique.multimodule  -DartifactId=webapp2 |

Now if you open the parent-project pom.xml, you will find all three modules being added in there.

|  |
| --- |
| <modules>    <module>model-lib</module>    <module>webapp1</module>    <module>webapp2</module>  </modules> |

To install a maven genrated jar r war to currect project and import that jar to m2 directory

1 Install using giving full detail path of file group id artiface id version pacaking

Mvn install:install-file -Dfile=<path-to-file> -DgroupId=<group-id> -DartifactId=<artifact-id> -Dversion=<vesrion> -Dpackaging=<pacaking>

2 using pom file also we can insatll like below one

Mvn install:install-file -Dfile=<path-to-file> -DpomFile=<path-to-pom-file>

3 if jar is built by apache maven with 2.5 version of maven-install-plugin it contain pom.xml in sub folder of MET-INF it will read from there in that case

Mvn install:install-file -Dfile=<path to file>

example: install:install-file -Dfile=src/main/resources/library/ReportProvider-1.0.jar -DgroupId=com.bcits.bsmartcore -DartifactId=ReportProvider -Dversion=1.0 -Dpackaging=jar

example: install:install-file -Dfile=src/main/resources/library/BsmartLogWriter-1.0.jar -DgroupId=com.bcits.bsmartframework -DartifactId=BsmartLogWriter -Dversion=1.0 -Dpackaging=jar

SPRING BOOT MAIL

SpringBoot doc for mail refer : <https://docs.spring.io/spring-framework/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/html/mail.html>

Using JavaMailSender interface , Springboot provide autoconfiguration and stater model

We have two class to create a mail

1 SimpleMailMessage

2 MimeMailMessage

Dependency

<dependency>

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

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

</dependency>

spring.mail.host=smtp.gmail.com

spring.mail.username=bcitsiworknotification@gmail.com

spring.mail.password=Bcits@123

spring.mail.port=587

#spring.mail.port=465

spring.mail.properties.mail.smtp.starttls.enable=true

@Override

public void sendNotification(Author author)

{

SimpleMailMessage mail=new SimpleMailMessage();

mail.setTo(author.getEmail());

mail.setSubject("Spring Boot Mail");

mail.setText("Spting Boot Test Mail Service");

javaMailSender.send(mail);

}

Spring BOOT GUIDE for Integration reference kit

<https://spring.io/guides/>

@SpringBootApplication

@EnableScheduling

public class BlogAppApplication {

public static void main(String[] args) {

SpringApplication.run(BlogAppApplication.class, args);

}

@Component

public class ScheduledTasks {

private static final Logger log = LoggerFactory.getLogger(ScheduledTasks.class);

private static final SimpleDateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");

@Scheduled(fixedRate = 5000)

public void reportCurrentTime() {

log.info("The time is now {}", dateFormat.format(new Date()));

}

}

Spring BOOT CronSequenceGenerator reference:

<https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/scheduling/support/CronSequenceGenerator.html>

public class CronSequenceGenerator

extends [Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html?is-external=true" \o "class or interface in java.lang)

Date sequence generator for a [Crontab pattern](https://www.manpagez.com/man/5/crontab/), allowing clients to specify a pattern that the sequence matches.

The pattern is a list of six single space-separated fields: representing second, minute, hour, day, month, weekday. Month and weekday names can be given as the first three letters of the English names.

Example patterns:

* "0 0 \* \* \* \*" = the top of every hour of every day.
* "\*/10 \* \* \* \* \*" = every ten seconds.
* "0 0 8-10 \* \* \*" = 8, 9 and 10 o'clock of every day.
* "0 0 6,19 \* \* \*" = 6:00 AM and 7:00 PM every day.
* "0 0/30 8-10 \* \* \*" = 8:00, 8:30, 9:00, 9:30, 10:00 and 10:30 every day.
* "0 0 9-17 \* \* MON-FRI" = on the hour nine-to-five weekdays
* "0 0 0 25 12 ?" = every Christmas Day at midnight

Spring BOOT GUIDE for Integration reference kit

## Swagger 2 in Spring Boot

Swagger 2 is an open-source project used to describe and document RESTful APIs. Swagger 2 is language-agnostic and is extensible into new technologies and protocols beyond HTTP. The current version defines a set HTML, JavaScript, and CSS assets to dynamically generate documentation from a Swagger-compliant API. These files are bundled by the [Swagger UI](http://swagger.io/swagger-ui/" \t "https://dzone.com/articles/_blank) project to display the API on the browser. Besides rendering documentation, Swagger UI allows other API developers or consumers to interact with the API’s resources without having any of the implementation logic in place.

The Swagger 2 specification, which is known as [OpenAPI specification](https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md" \t "https://dzone.com/articles/_blank), has several implementations. Currently, [Springfox](https://springfox.github.io/springfox/" \t "https://dzone.com/articles/_blank) that has replaced Swagger-SpringMVC (Swagger 1.2 and older) is popular for Spring Boot applications. Springfox supports both Swagger 1.2 and 2.0.

POM.xml dependency

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.6.1</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.6.1</version>

<scope>compile</scope>

</dependency>

Issue with :

Swagger2 enable and jars cause comiple error below issue while building app fix it

Correct the classpath of your application so that it contains a single, compatible version of org.springframework.plugin.core.PluginRegistry

Remove above dependency and Add new Depedency like below

<!-- io.springfox setup -->

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>3.0.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>3.0.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-data-rest</artifactId>

<version>3.0.0-SNAPSHOT</version>

</dependency>

<repositories>

<repository>

<id>jcenter-snapshots</id>

<name>jcenter</name>

<url>http://oss.jfrog.org/artifactory/oss-snapshot-local/</url>

</repository>

</repositories>

@Configuration

@EnableSwagger2WebMvc

public class SwaggerConfig {

@Bean

public Docket applcationApi() {

return new Docket(DocumentationType.SWAGGER\_2).select().apis(RequestHandlerSelectors.any())

.paths(PathSelectors.any()).build();

}

}

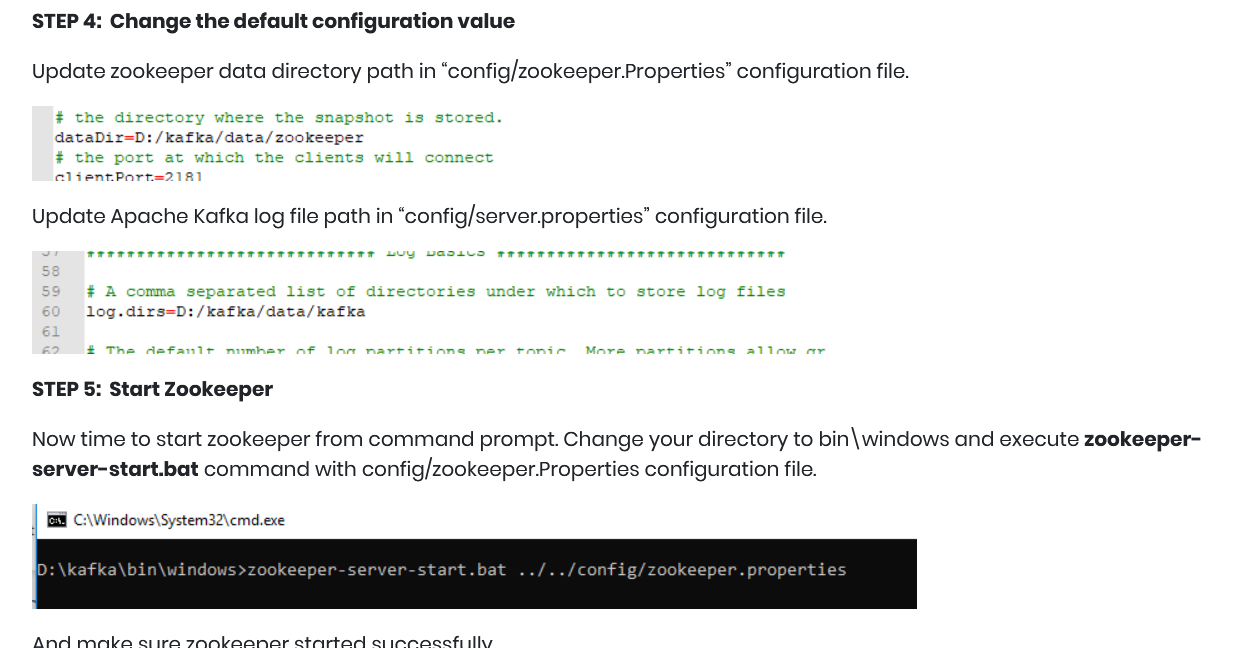
This line will fix swagger dependency

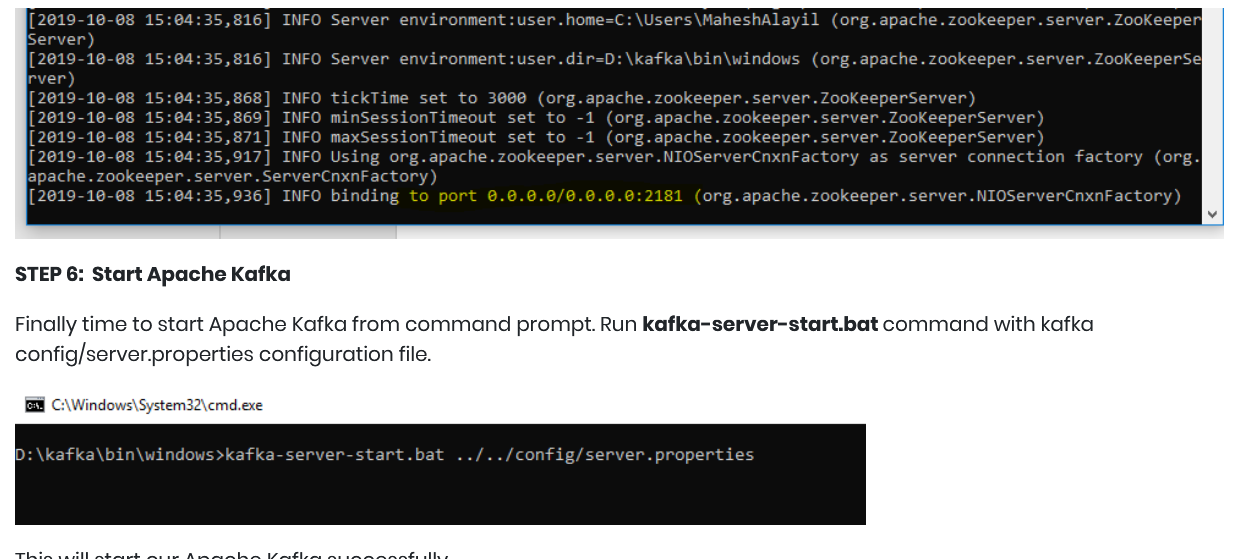
Spring Boot Kafka :

<https://www.goavega.com/install-apache-kafka-on-windows/>

Follow above steps :

To config





D:\kafka\bin\windows>zookeeper-server-start.bat ../../config/zookeeper.properties

D:\kafka\bin\windows>kafka-server-start.bat ../../config/server.properties

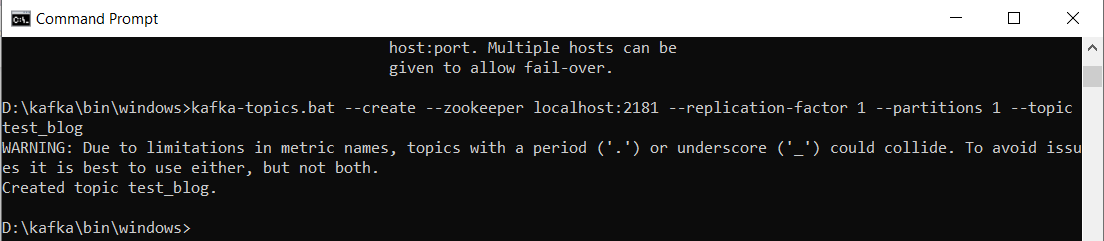
D:\kafka\bin\windows>kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test\_blog

Def port in local system will be 2181 ie zookepper node if we have multiple we can can config it

Replication factor can be increased

Partition also can be increased

Topic name test\_blog



Now we can publish but before that consumer need to configur for that topic to check if message being posted or not

Consumer for particular topic

D:\kafka\bin\windows>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test\_blog --from-beginning

SPRING BOOT ELK

The ELK Stack consists of three open-source products - Elasticsearch, Logstash, and Kibana from Elastic.

* Elasticsearch is a NoSQL database that is based on the Lucene search engine.
* Logstash is a log pipeline tool that accepts inputs from various sources, executes different transformations, and exports the data to various targets. It is a dynamic data collection pipeline with an extensible plugin ecosystem and strong Elasticsearch synergy
* Kibana is a visualization UI layer that works on top of Elasticsearch.

These three projects are used together for log analysis in various environments. So Logstash collects and parses logs, Elastic search indexes and store this information while Kibana provides a UI layer that provide actionable insights.

**Use Cases-**

* Consider you have a single application running and it produces logs. Now suppose you want analyze the logs generated. One option is to manually analyze them. But suppose these logs are large, then manually analyzing them is not feasible.
* Suppose we have multiple Application running and all these applications produce logs. If we have to analyze the logs manually we will need to go through all the log files. These may run into hundreds.

We can use ELK here to analyze the logs more efficiently and also using more complex search criterias. It provides log aggregation and efficient searching.

Elastic Search

-Run the elasticsearch.bat using the command prompt. Elasticsearch can then be accessed at localhost:9200

<http://localhost:9200/>

{

"name": "SYSTEM-92",

"cluster\_name": "elasticsearch",

"cluster\_uuid": "yfk119weS7GbIFpMLJp3Zg",

"version": {

"number": "7.8.0",

"build\_flavor": "default",

"build\_type": "zip",

"build\_hash": "757314695644ea9a1dc2fecd26d1a43856725e65",

"build\_date": "2020-06-14T19:35:50.234439Z",

"build\_snapshot": false,

"lucene\_version": "8.5.1",

"minimum\_wire\_compatibility\_version": "6.8.0",

"minimum\_index\_compatibility\_version": "6.0.0-beta1"

},

"tagline": "You Know, for Search"

}

Kibana

Kibana folder=>config=>kibana.yml => uncomment or add below line path of elasticsearch path

# The URLs of the Elasticsearch instances to use for all your queries.

elasticsearch.hosts: "http://localhost:9200"

Kibana floder=> bin=> kibana.bat

Run the kibana.bat using the command prompt. kibana UI can then be accessed at localhost:5601

LOGSTASH

Create a logstash.conf file in the root directory of the Logstash installation and copy the following code into it.

Add data inside

input {

kafka {

bootstrap\_servers => "localhost:9092"

topics => ["test\_blog"]

}

}filter {

grok {

match => [ "message", "%{GREEDYDATA}" ]

}

}output {

elasticsearch {

hosts => ["localhost:9200"]

index => "test\_blog-%{+YYYY.MM.dd}"

workers => 1

}

}

To start LOGSTASH

D:\devtools\logstash-7.8.0\bin>logstash.bat -f logstash.conf

Successfully started Logstash API endpoint {:port=>9600}

Go To Kibana UI - ImportElastic Search Data

go to kibana UI console- localhost and create an index pattern logstash-\* to see the indexed data-