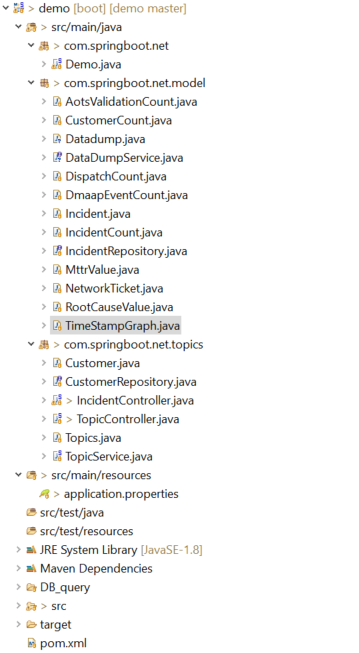
Spring Boot + MongoDB (Mongo Template).



Pom.xml

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-data-mongodb</artifactId>

</dependency>

</dependencies>

Application Properties:

server.port=8088

#mongodb

spring.data.mongodb.host=localhost

spring.data.mongodb.port=27017

spring.data.mongodb.database=test

#logging

logging.level.org.springframework.data=debug

logging.level.=error

Controller:

**@RequestMapping(method=RequestMethod.POST,value="/incident")**

**public** List<TimeStampGraph> getCustomerCount(Boolean historic){

mongoTemplate.setReadPreference(ReadPreference.*secondaryPreferred*());

TimeStampGraph data = **new** TimeStampGraph();

data.setTarget("volume of customer effected");

List<List<Long>> points = **new** ArrayList<>();

LookupOperation lookupOperation = LookupOperation.*newLookup*()

.from("sSOTIncidentEntity")

.localField("incident\_id")

.foreignField("incident\_id")

.as("incident\_entity");

Aggregation agg=**null**;

**if**(historic) {

agg = *newAggregation*(

lookupOperation,

*unwind*("incident\_entity"),

Aggregation.*project*().and("incident\_entity.created\_timestamp").as("incident\_entity.created\_timestamp").and("incident\_id").as("incident\_id").and("customers").as("customers").and("\_id").as("\_id"),

*unwind*("customers"),

Aggregation.*sort*(Sort.Direction.***ASC***,"customers"),

Aggregation.*group*("customers.customerIdValue").first("incident\_entity.created\_timestamp").as("timestamp"),

Aggregation.*project*().and("\_id").as("\_id").and("timestamp").as("timestamp").and("customers.customerIdValue").as("cust\_cnt"),

Aggregation.*group*("timestamp").sum("\_id").as("cust\_cnt").count().as("timestamp"),

Aggregation.*sort*(Sort.Direction.***ASC***,"\_id")

).withOptions(*newAggregationOptions*().allowDiskUse(**true**).build());

}**else** {

agg = *newAggregation*(

lookupOperation,

*unwind*("incident\_entity"),

Aggregation.*match*(**new** Criteria().andOperator(Criteria.*where*("incident\_entity.created\_timestamp").gt((Instant.*now*().toEpochMilli()-(86400000\*7)) + "").lte(Instant.*now*().toEpochMilli() + ""))),

Aggregation.*project*().and("incident\_entity.created\_timestamp").as("incident\_entity.created\_timestamp").and("incident\_id").as("incident\_id").and("customers").as("customers").and("\_id").as("\_id"),

*unwind*("customers"),

Aggregation.*sort*(Sort.Direction.***ASC***,"customers"),

Aggregation.*group*("customers.customerIdValue").first("incident\_entity.created\_timestamp").as("timestamp"),

Aggregation.*project*().and("\_id").as("\_id").and("timestamp").as("timestamp").and("customers.customerIdValue").as("cust\_cnt"),

Aggregation.*group*("timestamp").sum("\_id").as("cust\_cnt").count().as("timestamp"),

Aggregation.*sort*(Sort.Direction.***ASC***,"\_id")

);

}

/\*Aggregation agg = newAggregation(

lookupOperation,

project("tags"),

unwind("tags"),

group("tags").count().as("n"),

project("n").and("tag").previousOperation()

);\*/

AggregationResults<CustomerCount> results = mongoTemplate.aggregate(agg,"sSOTCustomersEntity",CustomerCount.**class**);

List<CustomerCount> customerCount = results.getMappedResults();

System.***out***.println(customerCount.get(4).get\_id());

System.***out***.println(customerCount.get(4).getTimestamp());

**for**(CustomerCount inc : customerCount ) {

List<Long> point = **new** ArrayList<>();

**long** timestp = inc.get\_id().isEmpty() ? 0 :Long.*parseLong*(inc.get\_id());

System.***out***.println(timestp);

//point.add(incidentId++);

point.add(Long.*parseLong*(inc.getTimestamp()));

point.add(timestp);

points.add(point);

}

data.setDatapoints(points);

List<TimeStampGraph> dataset = **new** ArrayList<>();

dataset.add(data);

**return** dataset;

}

Pojo/Module

**public** **class** TimeStampGraph {

**public** String target;

**public** List<List<Long>> datapoints = **null**;

**public** String getTarget() {

**return** target;

}

**public** **void** setTarget(String target) {

**this**.target = target;

}

**public** List<List<Long>> getDatapoints() {

**return** datapoints;

}

**public** **void** setDatapoints(List<List<Long>> points) {

**this**.datapoints = points;

}

}