ushur

Tool to analyze status of SMS messages

A tool to help debug flow of SMS messages within the Ushur Platform

ushur

Summary

Ushur provides a platform for enterprises to engage with their customers. The engagements can happen over a variety of channels - SMS being one of them. A log of these messages along with their delivery status and other information are stored for analysis. Today the analysis is manual and there is scope for a tool to be developed that can aid and make it easy to debug a flow.

Use Case

Below is the list of use cases where this tool will come in handy. This list can continue to grow and hence the tool is made to be extensible

- For a provided date range provide data on the count of SMS sent and their status
- For a provided session ID provide a chronological listing of SMS and their status with details of failure if any

Scope

The scope of this project was to develop a tool that can work with a configurable Mongo DB collection of a predefined structure and provide APIs for querying data based on commands that can be configured. The tool is developed in Java as a **SpringBoot Application** and uses **Dependency Injection** to deliver output based on the command given. The application provides **REST** endpoints for querying the information listed above. This API responds to POST requests and asks for payload as a Request Body.

Requirements

The requirement was to develop an application that can work independently with a configurable source of data. Configuration is allowed via an **application.properties** file using which the user can configure the following

- ➔ DB Host and Port
- ➔ DB and Collection name
- ➔ Default duration queried for, in case any of the limitation is null

The application delivers a REST endpoint that takes “command” and additional parameters as a JSON payload. The endpoint can be

Endpoint: http://host:port/query

As of now, there can be two types of payload:

Payload 1:

{

command: SMS_STATUS,

fromDate: <optional>,

toDate: <optional>

}

As stated in Requirements, if the value of “fromDate” or “toDate” is empty then it will deliver the output for a default duration of MAX_DURATION.

The Output of this payload will be like :


```
{
  "command": "SMS_STATUS",
  "fromDate": "",
  "toDate": "2019-04-17T08:42:54.493Z",
  "responseData": {
```

```
    "delivered": 17,
    "sent": 1
  }
}
```


Payload 2:

{

command : SESSION_SMS_STATUS,

SessionId : <sid>

}

In this case, if the SessionId is given empty, it will return the response with the value of responseData to be empty. Otherwise, the response will look like this


```
{
  "command": "SESSION_SMS_STATUS",
  "SessionId": "a42aacd2-2ef8-4fdb-96bd-a4fe1127c9e0-uxidxyz-ab795954-8f3c-44dd-bbf9-448645d72858",
  "responseData": [
    {
      "UshurMsgId": "dN5E4FObQ6",
      "DateCreated": "Thu Apr 11 17:48:24 IST 2019",
      "Status": "delivered",
```

ushur


```
    "PartialText": "[deepti]:Hi team.. welcome to Ushur",
    "Vendor": "nexmo",
    "ToPhNo": "919900961301",
    "VirtNo": "447937946597"
  },
  {
    "UshurMsgId": "YLK9HKyWFB",
    "DateCreated": "Thu Apr 11 17:48:34 IST 2019",
    "Status": "delivered",
    "PartialText": "Have you taken part in voting polls before?\nPlease respond with a number.\n1) yes\n2) no",
    "Vendor": "nexmo",
    "ToPhNo": "919900961301",
    "VirtNo": "447937946597"
  },
  {
    "UshurMsgId": "YLK9HKyWFB",
    "DateCreated": "Thu Apr 11 17:48:34 IST 2019",
    "Status": "delivered",
    "PartialText": "Have you taken part in voting polls before?\nPlease respond with a number.\n1) yes\n2) no]]",
    "Vendor": "nexmo",
```

```
      "ToPhNo": "919900961301",

      "VirtNo": "447937946597"

    }

  ]

}
```

Developer's Manual

The tool consists of a single project, **SMSStatusTracker** (the main project). An application.properties file is saved in the /conf directory.

In case, one is installing ("mvn clean install") the project from the start,

- Run it in **SMSStatusTracker**, A SMSStatusTracker.war file and a SMSStatusTracker-0.0.1-ush.jar will be generated in the /target folder so as to be used as a dependency for external commands (if the project gets extended).

Now, these **.war** and **application.properties** files can be deployed.

The command used to run the project will be,

```
java -classpath application.properties -jar SMSStatusTracker.war
```

How can the project be Extended ?

To add classes for other commands there are three ways, you can either use the **commands** project or you can just add classes to the com.ushur.commands package of the main project, in both the cases you can skip to Step 4, otherwise follow the steps:

1. Initiate a simple SpringBoot project, one way could be using https://start.spring.io/ .
   Project : Maven, **Language** : Java, **Spring Boot** : 2.3.3, **Packaging** : Jar, **Java** : 8, **Project Metadata** : GroupId : com.ushur .
2. Install the SMSStatusTracker project, so as to generate the SMSStatusTracker-0.0.1-ush. jar in .m2/repository.

3. Add these dependencies,

```
<dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
</dependency>
<dependency>
         <groupId>org.mongodb</groupId>
         <artifactId>mongo-java-driver</artifactId>
         <version>3.12.6</version>
 </dependency>
 <dependency>
        <groupId>com.ushur</groupId>
        <artifactId>SMSStatusTracker</artifactId>
        <version>0.0.1</version>
        <classifier>ush</classifier>
</dependency>
```

4. Create a class with the same name as the command required and it has to **implement CommandInterface**. For example, I might need to make a new command USHURMSG_SMS_STATUS which takes “UshurMsgId” as payload, then I would have to make a class like this:

   @Component
   public class USHURMSG_SMS_STATUS implements CommandInterface .

   This @Component annotation is required to declare it as a Bean to be scanned by SMSStatusTracker.

5. Now, the CommandInterface has a method “respond”, and we have to Override that. Like,

   @Override
   public JsonObject respond(JsonObject payload, Environment env, MongoClient mongoClient)

   The return type is JsonObject and it asks for arguments (JsonObject, Environment, MongoClient).

6. The database and collection name are set in application.properties of SMSStatusTracker and can be accessed as,

   MongoDatabase db =
   mongoClient.getDatabase(env.getProperty("spring.data.mongodb.database"));
   MongoCollection<Document> collection =
   db.getCollection(env.getProperty("spring.data.mongodb.collection"));

   The names can be changed by using String with the required name instead of **env.getProperty("spring.data.mongodb.database")** or **env.getProperty("spring.data.mongodb.collection")**.

7. A JsonObject must be returned as it will be added to the Response Json for the POST request.
8. After this, go to Developer's Manual for further execution of the project.

Date	Author	Description