API performance testing Tool

## Spring-boot based RESTful APIs

The following RESTful API has been developed in the Spring Boot framework.

Error/ exception handling has also been included in the API code.

Class : Flights

1. Id - id

2. Departure Airport - DeptApt

3. Arrival Airport - ArrApt

4. Departure Time - DeptTime

5. Arrival Time - ArrTime

6. Distance - Distance

7. Price - Price

8. Type - International/Domestic - type

Airports have been named by their three letter codes. For eg. DEL, GUW etc.

GET Methods: “{}” represents the input variable in the path.

1. Search by Flight id - **/flightInfo/{FlightId}**
2. Search by Departure and arrival (returns a List of flights with the given arrival and departure airports) - **/flightInfo/departure/{depart}/arrival/{arrival}**
3. Search by Type (Domestic/International) (returns a List of flights with the given type i.e. International or Domestic) - **/flightInfo/airport/{I || D}**
4. All Flights on a Particular Airport (returns a List of flights departing from a particular airport) -  **/flightInfo/departure/{airport}**
5. All Flights on a Particular Airport (returns a List of flights arriving on a particular airport)- **/flightInfo/arrival/{airport}**
6. All flights in the database - **/flightInfo**

POST Method:

1. Add a new Flight-  **/addFlight**
2. Add a list of flights - **/addList**

**NOTE: Duplicate flight Ids are not allowed. i.e. each flight requires a unique id.**

PUT Methods: The flight id of the required flight is fetched so that its particulars can be edited.

“{}” represents the input variable in the path.

1. Edit Arrival time -  **/edit/{id}/arr/{time}**
2. Edit Departure time -  **/edit/{id}/dept/{time}**
3. Edit Arrival Airport - **/edit/{id}/arrapt/{Airport}**
4. Edit Departure Airport -  **/edit/{id}/deptapt/{Airport}**
5. Edit Price -  **/edit/{id}/price/{price}**

DELETE Method: The required flight fetched by its Id is deleted.

1. By flight ID - **/delflight/byid/{id}**

Logging functionality configuration-

1. Two appenders are added in the logging configuration- One to append log data to the console, the other to append log data to external log file.
2. The external log file properties are also set in the logging configuration.We’ll have a current log file which will be having the most recent log data, when the size of this file exceeds 50KB, it will pushed into the archived folder and will be stored in the creation date and serial number format.When the size of this archived folder exceeds 10MB, automatically the contents of the archived folder are deleted.
3. We’ve also set the max history for log file storage to be 7 days, log files will be automatically deleted after 7 days of creation.
4. This is done to avoid overload when we deploy the application on external servers.
5. All the above configurations are made in the logback.xml file and they can be modified according to the usage from there.

Following classes were used so as to handle the requests and deal with the MySQL database.

1. FlightRepository : deals with the database
2. Get/Put/Post/DeleteResource : handles all the HTTP requests.

MySQL Database:

* Hosted locally
* Database name: flights\_schema
* Table name: flights

## 2. Docker containerization

### How to create Docker image and run it:

**While using docker, since the API will use localhost, please use localhost:8080/FlightApi/ as the base url.**

**For ex. localhost:8080/FlightApi/flightInfo/1 for a sample GET request.**

* docker pull sarthvitekar/imgflights
* Then open cmd in the directory containing the docker-compose.yaml file. Enter the command “docker-compose up -d”. The API should be accessible at localhost port 8080.
* The docker container for the MySQL database is also automatically made by the above command.

1. Jmeter Test Plan Configuration:

This is a java-based application to configure a test-plan for performance and load testing of RESTful APIs.

Key components used in the Application:

* Java SDK 1.8
* Open-source JMeter Libraries
* Maven plugins
* Jmeter Dependencies

Java classes used:

1. JMeterFromScratch
2. Allsource
3. DelInput
4. FileIn
5. GetInput
6. Inputs
7. PostInput
8. PutInput
9. Read

Input text file format:

Domain

Port (keep 80 for Default)

Thread

Loop

RampTime

No of Get Requests: N

N line of get request path

No of Post Requests: M

1st Post request path

1st post request body

same for M times (Path followed by Body for that Path)

No of Put Requests: K

K line of Put request path

No of Del Requests: L

L line of Del request path