**Coding Assessment**

Distributed event persistence platform

Domenico Rosario Caldesi

**Producer**

The producer application exposes a service to upload the event file on the path**: /event/send**. This accept a file that will be parsed line-by-line and for each line a JSON object that represents the event is built and then queued into an in-memory queue.

Periodically, a queue consumer thread looks into the queue and if any event is present, send them one by one to the configured endpoint of the consumer application.

Is possible to configure some parameters on this side editing the file **WEB-INF/conf/config.xml**. Following, a configuration example:

<configuration>

<eventEndpoint>http://localhost:8080/consumer-rest/event</eventEndpoint>

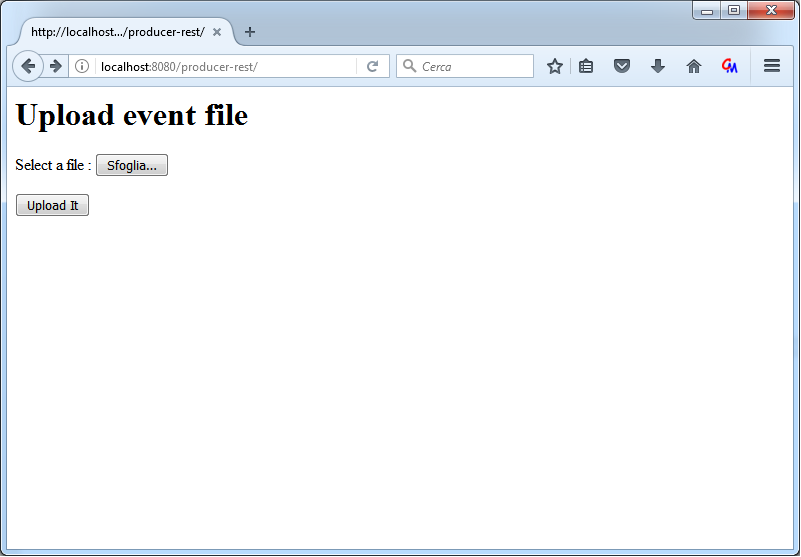
<queueConsumerThreads>2</queueConsumerThreads>

<queueConsumerThreadSleepTime>500</queueConsumerThreadSleepTime>

</configuration>

* *eventEndpoint*: is the endpoint on the consumer application that will be used to send the collected events.
* *queueConsumerThread*: the number of queue consumer thread to be instantiated.
* *queueConsumerThreadSleepTime*: sleep time in milliseconds of the queue consumer threads.

In order to test in an easy way the Producer, a very simple page for the event file upload is provided on the root of the web application that, usually will be: <http://localhost:8080/producer-rest/>



This upload page allow to select a file and then submit with a form that **POST** to the **/event/send** endpoint.

**Consumer**

The consumer application interface with a MongoDB instance and use it as repository to store the events. Expose a POST call on the path **/event** that receive a JSON object that represent the event into the HTTP body and then persists it on the MongoDB instance.

Is possible to configure some parameters into the file **/WEB-INF/conf/config.xml** as in the following example:

<configuration>

<host>localhost</host>

<port>27017</port>

<username></username>

<password></password>

<databaseName>event\_repo</databaseName>

<collectionName>events</collectionName>

</configuration>

All of these are parameters of configuration needed for the connection with the Repository component.

* *host*: hostname or ip address of the MongoDB instance;
* *port*: port of the MongoDB instance;
* *username*: username used to connect to the DB. Could be leaved blank or removed if no login parameters are needed;
* *password*: the authentication password for the given username. Could be leaved blank or removed;
* *databaseName*: the database name;
* *collectionName*: the collection name;

**Testing with Docker**

Before to build the Producer and Consumer dockerfiles, it is necessary to copy **lib/** folder into the Producer and Consumer folders.

**Repository**

Instantiate the MongoDB repository using the following commands:

docker-machine rm repository

docker-machine create --driver virtualbox repository

docker-machine env --shell cmd repository

@FOR /f "tokens=\*" %i IN ('docker-machine env --shell cmd repository') DO @%i

Now the “repository” machine should be created. We need to take note of the machine IP, with the following command:

docker-machine ip repository

Build and launch the image:

docker build -t repository-app .

docker run -p 27017:27017 –name mongo\_instance -d repository-app

Now the MongoDB instance should be up and running on local port 27017.

**Consumer**

As the repository instance, create the docker machine:

docker-machine rm consumer

docker-machine create --driver virtualbox consumer

docker-machine env --shell cmd consumer

@FOR /f "tokens=\*" %i IN ('docker-machine env --shell cmd consumer') DO @%i

Take note of the IP:

docker-machine ip consumer

IMPORTANT: Before build the Dockerfile, copy the lib folder at the same level of the Dockerfile:

xcopy ..\lib .\lib\

docker build -t consumer .

docker run -it --rm -p 8181:8080 consumer

In order to communicate with the MongoDB instance, you should edit the **/conf/config.xml** file filling the host and the port elements and then restart the Tomcat.

**Producer**

Analogously to the Consumer instance, create a new docker machine:

docker-machine rm producer

docker-machine create --driver virtualbox producer

docker-machine env --shell cmd producer

@FOR /f "tokens=\*" %i IN ('docker-machine env --shell cmd producer') DO @%i

Take note of the IP, will be the entry point to test all the distributed platform:

docker-machine ip producer

IMPORTANT: Before build the Dockerfile, copy the lib folder at the same level of the Dockerfile:

xcopy ..\lib .\lib\

docker build -t producer .

docker run -it --rm -p 8282:8080 producer

Configure in **/conf/config.xml** the eventEndpoint: http://<ip-of-consumer>:8181/consumer-rest/event and restart the Tomcat instance.

Now it is possible to call the via HTTP POST the endpoint (http://<ip-of-producer>:8282/producer-rest/event/send) providing in multipart the file containing the events (source.txt) or use the simple page located at the root of the project:

http://<ip-of-producer>:8282/producer-rest/

After the invocation of the **/event/send**, the MongoDB collection “events” will be filled with all the data provided by the given file.