UUID Generation

# Problem Statement

Provider a software solution to generator global unique identifier.

# Challenge

Provide a solution and prove it will work world wide.

# UUID from rfc4122

rfc4122 provides 128 bits long UUID, this can be represented on 26 alpha-numeric long. This will be greater than 20 characters long. Using this mechanism will be dropped since it does not achieve the 20 characters UUID long requirement.

We can change the implementation of the rfc4122 to support only 20 characters, but we will then face the problem of proving the generated number is globally unique.

# REST Service UUID

Another solution is to use a REST service that generate sequence numbers. Since this service is state full due to the sequence number, scalability will be an issue. To solve that we will use two digest from the guid number to be used for scalability. These two digest will be called systemID. This will allow the guid service to be running on 1295(35\*36+35) different systems

Since this system uses sequence numbers then it is for sure unique. More implementation needs to be done to make sure of the systemId uniqueness, and no other system is using that system id.

This is very easy system to implement, and easy to scale.

# Solution

This solution will discuss in details the implementation of REST service UUID. This service will uses three integer numbers, every number needs up to 6 characters to represent it. So this means we will need 18 characters. The other two characters will be used for SystemID.

An increment method is needed to increment the sequence number. As soon as the first number reaches its maximum limit, we reset it to zero and increment the second number. The increment to second or third number will do the same. For the third number the increment may through an out of boundary exception, and in this case the system will not be reused again guid number generation.

# Deploy

This solution consists of three services, Guid, ServiceA, and ServiceB. Every one of these has application.properties file which sets the REST service port.

ServiceA comes with two files application.properties and application-dev.properties. Use the application.properties to set the active profile. You need to use dev profile will you deploy to docker. The dev profile sets URL regarding docker ip address, which you need to change if needed.

### Build

The first step is you need to do maven build for every system.

Use: mvn clean install to build every system of these.

The solution can be enhance to be build using single maven command, but this can be done later.

### Deploy to docker

Every one of these systems has its own Docker folder which includes a Docker file. The docker file needs a \*-0.0.1-SNAPSHOT.jar file where star can be guid, servicea, or serviceb. You need to copy \*-0.0.1-SNAPSHOT.jar to its Docker folder. The \*-0.0.1-SNAPSHOT.jar will be generated from the build step you did above.

#### Guid Docker deploy

1. Move to Guid directory.
2. mvn clean install
3. copy guid-0.0.1-SNAPSHOT.jar to Docker folder.
4. Change to Docker directory under Guid project
5. docker build -t shalabi67/guid .
6. do not forget the dot ☺ in the above command
7. docker run -p 9092:9092 -d --name guid shalabi67/guid
8. use url to get guid numbers: <http://192.168.99.100:9092/guid>

note: 192.168.99.100 will be used in servicea application-dev.properties file.

#### ServiceB Docker deploy

1. Move to ServiceB directory.
2. mvn clean install
3. copy serviceb-0.0.1-SNAPSHOT.jar to Docker folder.
4. Change to Docker directory under ServiceB project
5. docker build -t shalabi67/serviceb .
6. do not forget the dot ☺ in the above command
7. docker run -p 9091:9091 -d --name serviceb shalabi67/serviceb
8. use url to list exiting metadata: <http://192.168.99.100:9091/metadata>

#### ServiceA Docker deploy

1. Move to ServiceB directory.
2. Checkthe active profile is dev properties in application.properties
3. Fix application-dev.properties properties if needed
4. mvn clean install
5. copy servicea-0.0.1-SNAPSHOT.jar to Docker folder.
6. Change to Docker directory under ServiceA project
7. docker build -t shalabi67/servicea .
8. do not forget the dot ☺ in the above command
9. docker run -p 9090:9090 -d --name servicea shalabi67/servicea
10. use url to post metadata: http://192.168.99.100:9090/uuids