TOPICS

* Service and details
* Setup
* How to run services
* How to access service
* Swagger
* Other details

Service and details

City-map-provider(service)

It has logic to build a graph, adj matrix. It provides the APIs to get the connectivity between cities.

It has different strategies to fetch the connectivity result.

Different strategies(DISJOINT, FAST\_ADJ\_MATRIX, ADJ\_MATRIX) based on different scenarios -

1. ADJ\_MATRIX - We have lots of cities and having strong connectivity between them. So here we can have matrix two dimensional array with boolean entries for showing connectivity. This approach can handle future requirement of showing the path(traceability between cities, those are connected). Storing boolean inside array will cost less compare to storing object reference inside collection like map/(list |set).
2. FAST\_ADJ\_MATRIX - Suppose we have map, where cities are well connected with good density, we prefer using matrix but with linked list to traverse connected cities. Instead of using map as a cluster of cities we prefer to use 2 dimensional array. Less memory and good performance. This approach can handle future requirement of showing the path(traceability between cities, those are connected) Option 2 is very similar to option 1, difference is while traversing, no need to go one by one index inside array, we created linked list between the connected nodes so we can jump and save iteration time.
3. DISJOINT - We have disjoint set, this provides simplest implementation to find connectivity between cities.

This approach is best considering memory and response time.

1. [TODO] – having lots of cities with less possibilities of connectivity. (here we can go with map having collection of list/set as entry) BUT this option is not better compare to option 3, in case of finding connectivity.

This service(city-map-provider) is the heart of complete application. Finding the connectivity, this is solved inside this module.

City-map-selector (service)

This service is like a façade pattern simply gives interface to call city-map-provider service with specific strategy configured and fetched from config server.

Config server(service)

Gets file from git repo and make it available to all services. For now config server address is hard coded but when multiple config servers are running on cloud, we can use DiscoveryClient and get instance of `config server` by `application name`.

Inside application we have configured “city-map-provider” and “city-map-selector” to get property values from config server. Config server is also registered as Eureka client. ( we are now simply fetching config properties using @value annotation.

Discovery-service

Eureka service makes sure all register services are available to access by application name. This discovery feature helps in cloud environment. We can access the services by “application name”, no need to provide hostname details, eureka will take care of it. Eureka collects (uri, port) of all registered services w.r.t. application name.

Api-gateway

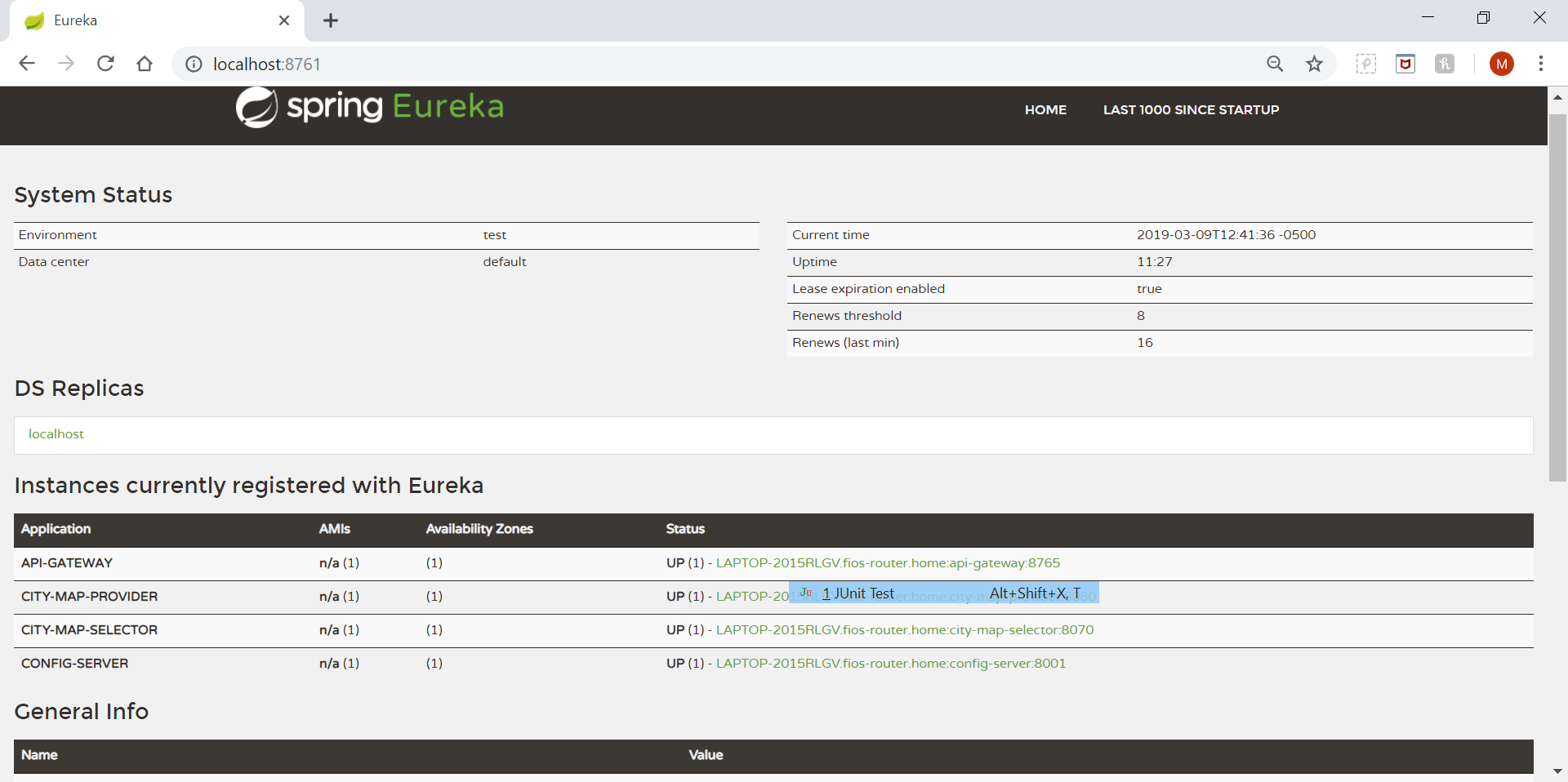
This provides the interceptor feature, we added simple filter in our example that filters the service request (log file shows that request has been filtered). We can find it’s been called inside city-map-selector service.

Ribbon

It is used inside project, for load balancing. Eureka, from “application name” figures out the registered services and using that information from eureka, ribbon balanced the load in round robin fashion.

Inside the service(city-map-selector) we provided feign client as “api-gateway” (i.e. Zuul service). We added Zuul reference so now feign is referring to gateway service meaning we need to add the actual service name in URI, i.e. /city-map-provider/connected-by-disjoint.(Actual rest call mapping inside city-map-provider was ‘/connected-by-disjoint’ but to provide target reference to Zuul we added ‘/city-map-provider’ as a starting point in uri.) Hence we api-gateway gets the request it reads the first part of URL and it knows the request belongs to which service.

Please find below Eureka page, shows all registered clients. All those clients are registered because we added eureka.client.service-url.default-zone entry inside properties file.



Setup

1. Input file location for cities inside <https://github.com/mangeshh/config>

We need to modify the properties file based on the application name. Meaning all properties for application `city-map-provider` resides inside city-map-provider.properties file.

1. All strategy for building data model for city-to-city connection configured inside city-map-selector properties file. Parameter is service.strategy.
2. Input file for city pairs is available inside

<https://github.com/mangeshh/config/blob/master/city-map-provider.properties>

parameter (city.map.input.file.location)

How to run services?

|  |
| --- |
| Clone the code. Either directly run them directly from IDE OR make a build(maven) and execute following command. |
| * java -jar api-gateway-0.0.1-SNAPSHOT.jar Application.java * java -jar discovery-service-0.0.1-SNAPSHOT.jar Application.java * java -jar config-server-0.0.1-SNAPSHOT.jar Application.java * >> wait for 10 seconds * java -jar city-map-provider-0.0.1-SNAPSHOT.jar Application.java * java -jar city-map-selector-0.0.1-SNAPSHOT.jar Application.java |

How to access the service?

|  |
| --- |
| Start services:  api-gateway service, discovery-service, config-server  Wait for 10-12 seconds till services complete registration with Eureka.  Then start city-map-provider, city-map-selector services.  URL to access the problem statement.  <http://localhost:8765/city-map-selector/connected?origin=L&destination=N>  sample data for city pair, inside file.  P,K  L,M  N,M  Boston, New York  Philadelphia, Newark  Newark, Boston  Trenton, Albany  8765 is the port for gateway service.  properties file for city-map-provider is city-map-provider.properties. please find parameter city.map.input.file.location |

Swagger

|  |
| --- |
| Swagger for city-map-selector and city-map-provider. |
| <http://localhost:8070/swagger-ui.html>  <http://localhost:8080/swagger-ui.html> |

Other Details

We can directly access connectivity from city-map-provider using following 3 rest calls.

|  |
| --- |
| <http://localhost:8080/connected-by-disjoint?origin=L&destination=N> |
| [http://localhost:8080/connected-by-adj-matrix?origin=L&destination=N](http://localhost:8080/connected-by-disjoint?origin=L&destination=N) |
| [http://localhost:8080/connected-by-fast-adj-matrix?origin=L&destination=N](http://localhost:8080/connected-by-disjoint?origin=L&destination=N) |

We are using `gateway api` and `load balancing` on top of `city-map-provider` service.

**TODO** - These above 3 calls would be prohibited from outside world using Zuul and profiler.

|  |  |
| --- | --- |
| service | port |
| Config server | 8001 |
| Eureka | 8761 |
| city-map-selector | 8070 |
| city-map-provider | 8080 |
| api-gateway | 8765 |