SpotLocationService

Description:

SpotLocationService is a microservice which is built on Spring Cloud Platform and locates the closest distance to a specific spot like airport, beach and return the details to the client. Currently this service only supports for the Florida State and also with limited location spots.

- This uses Eureka, a Spring based Naming server to register and discover all the microservices so that there are referred by the service names rather than the hard-coded URIs.
- This also uses Zuul, Spring API Gateway to route the request through a single channel.
- This also uses Ribbon, spring's version of load balancer so that multiple instances of the AirportLocateService, BeachLocateService can be started in various http-ports. Again none of the URIs are hardcoded even when run multiple instances of the services.
- This also uses H2, in-momory database to store the Beach and aiport location coordinates and uses JPA to fetch the data. Please follow the 'Server Start order' to run this application.

Services Details:

1. Eureka-Naming-Server: Spring Cloud Naming Server running in port #8761

URI : http://localhost:8761/
Server Starting Order - #1

2. Zuul-API-Gateway : Spring cloud API Gateway running in port #8765

URI : http://localhost:8765
Server Starting Order - #2

3. Airport-Locate-Service: Finds closest airport to the given State, latitude and longitude.

Direct URI : http://localhost:8300/locate-airport/state/FL/latitude/28.4810971/longitude/-81.508834
API Gateway URI :
http://localhost:8765/airport-locate-service/locate-airport/state/FL/latitude/28.4810971/longitude/-81.508834
Server Starting Order : #3

4. Beach-Locate-Service - Finds closest beach to the given State, latitude and longitude

Direct URI : http://localhost:8400/locate-beach/state/FL/latitude/28.4810971/longitude/-81.508834
API Gateway URI :
http://localhost:8765/beach-locate-service/locate-beach/state/FL/latitude/28.4810971/longitude/-81.508834
Server Starting Order - #4

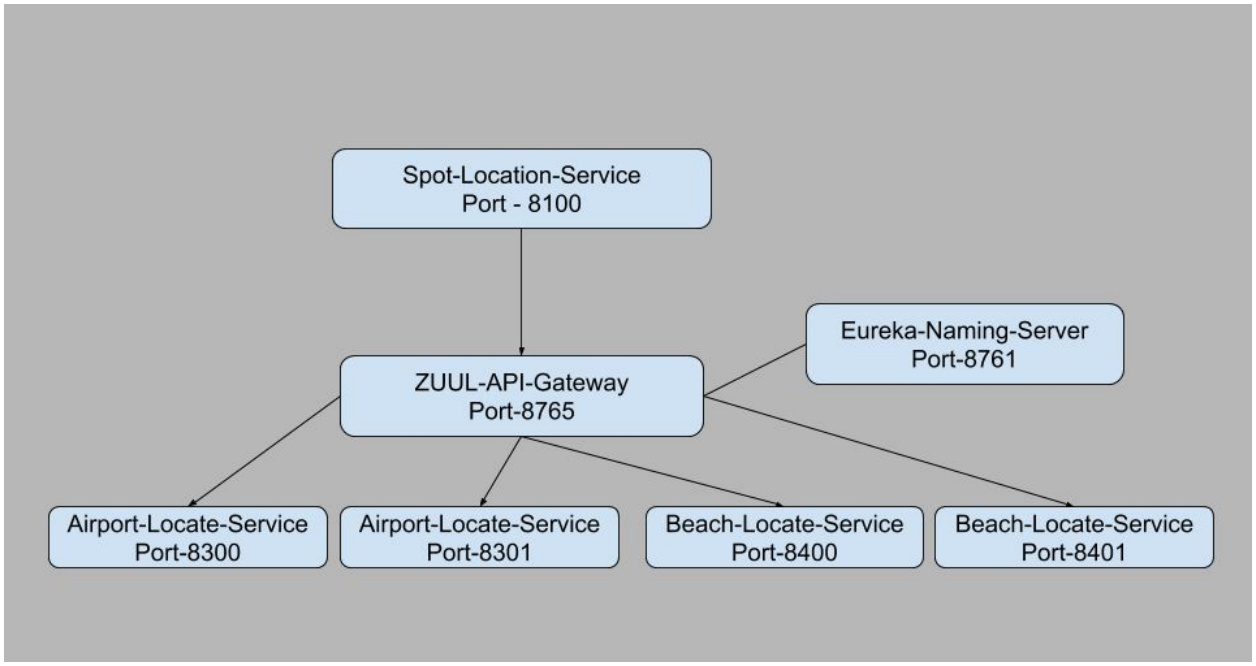
5. Spot-Location-Service: This is the main service which calls the appropriate microservice (whether to find the Airport or Beach)

Direct URI to find Airport : http://localhost:8100/locate-spot/airport/state/FL/latitude/28.4810971/longitude/-81.508834
Direct URI to find Beach : http://localhost:8100/locate-spot/beach/state/FL/latitude/28.4810971/longitude/-81.508834
API Gateway URI to find Airport :
http://localhost:8765/spot-location-service/locate-spot/airport/state/FL/latitude/28.4810971/longitude/-81.508834

API Gateway URI to find Beach :
http://localhost:8765/spot-location-service/locate-spot/beach/state/FL/latitude/28.4810971/longitude/-81.508834
Server Starting Order - #5

Spot-Location-Serivice System Spec:

Please see below the detail design specification of this application with the name of each microservices and on which port it is running and also describes the load balancing.



Technical Things need to be incorporated:

- Enable Swagger2
- Write Unit Test and Integration Test
- Spring Security
- JWT token based Authorization
- Logging and Exception Handling
- Common error
- Input Validation
- Enhance to accept XML/JSON payload
- Dockerize the microservices for easier deployment.

Functional Things, need to be Incorporated:

- Currently this application only supports state of Florida
- Does have very limited data so polling against limited location coordinates
- The distance calculation logic is not working as expected, the results are not accurate
- This can be enhanced for various other spots, with added features and provide more information to the client.