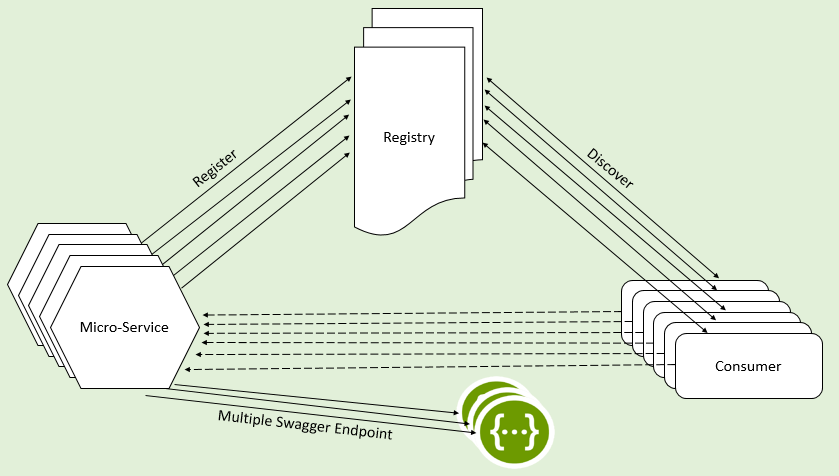
**Centralized API Documentation in Microservices**

**Using Spring Boot, SpringFox Swagger-UI, and Eureka**

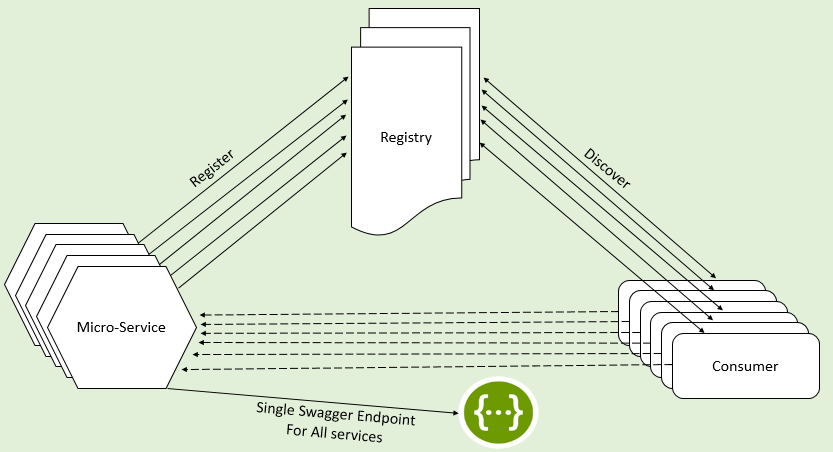
**The Problem**

As we already know, it is very easy to document REST applications using the SpringFox Swagger-UI library, but a problem arises when we are working in an environment where we have multiple REST-based applications. Typically, we face this issue in a microservices environment. Most of us end up managing a separate Swagger-UI for each application, which means that each service will have its own endpoint and to access the Swagger-UI and we have to use a different URL for different applications.

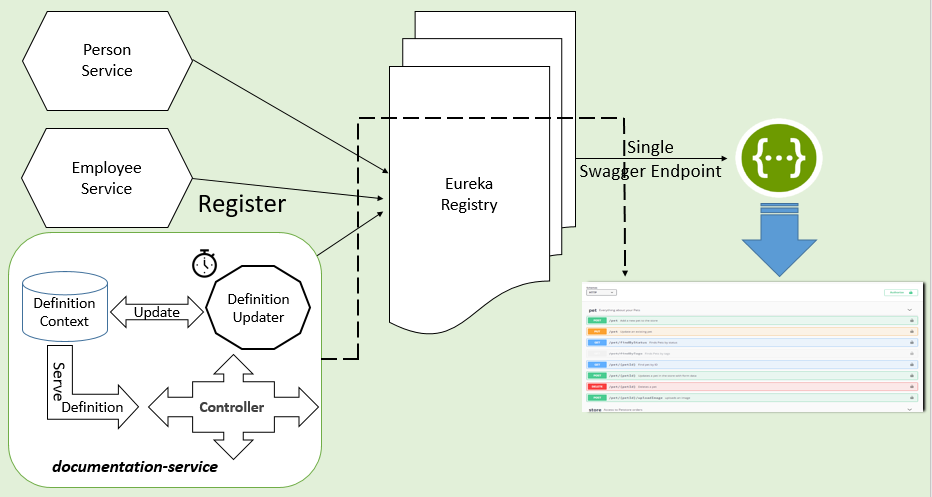


**The Solution**

To access all this API documentation from a single URL, a solution can be implemented using the below steps:

1. Get the list of registered service instances from the service registry.
2. For each registered service instance, pull the Swagger definition JSON from the instance and store it locally. In our case, we are putting this JSON in the in-memory documentation context backed by a concurrent map.
3. Refresh the in-memory context at regular intervals to automatically remove/add the definitions as they are updated in the service registry.
4. Provide a single endpoint to serve Swagger definitions from our in-memory store on the basis of service instance name.

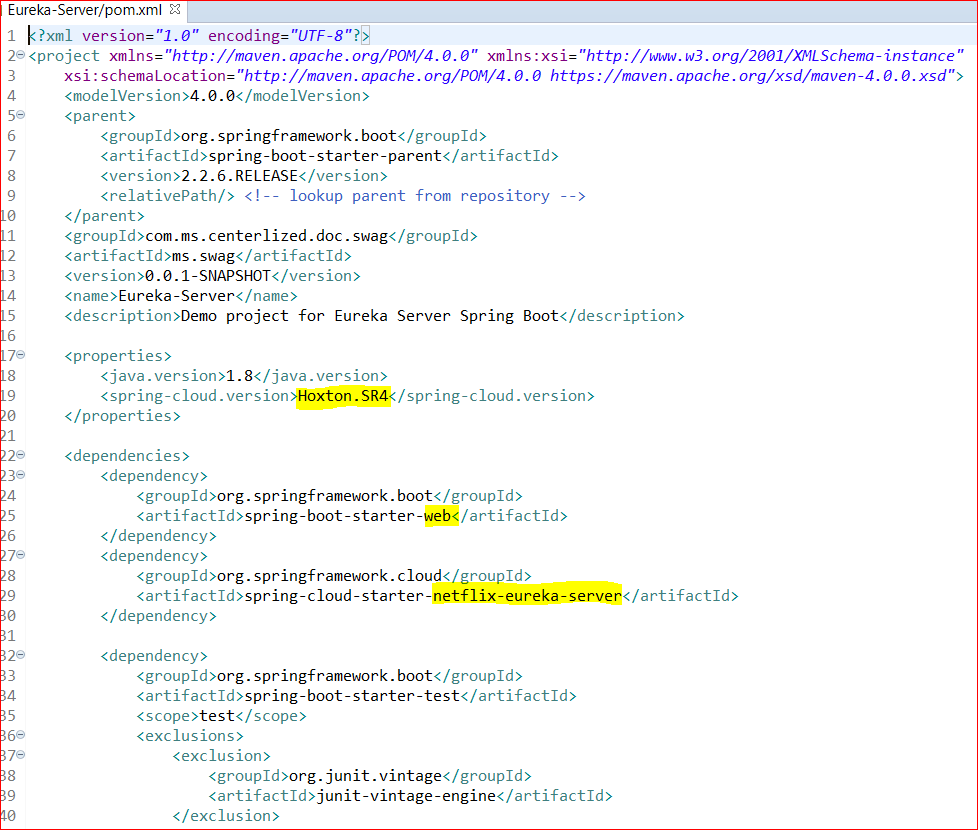
**Implementation**

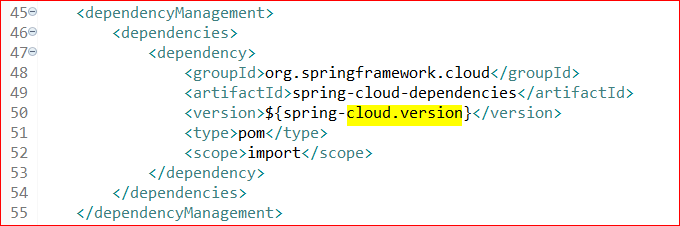
Thanks to the folks at [SpringFox](https://github.com/springfox" \t "_blank), Swagger-UI offers the great functionality of [extending](http://springfox.github.io/springfox/docs/current/#aggregating-multiple-swagger-specifications-in-the-same-swagger-ui) the documentation by providing an implementation of the bean SwaggerResourcesProvider.  Let us implement this using a hypothetical microservices enviroment setup. Our environment has the below services:

1. central-docs-eureka-server: Service registry powered by Netflix Eureka
2. employee-application and person-application: REST applications with Swagger-UI enabled. You can follow [this article](https://dzone.com/articles/monitoring-using-spring-boot-2-prometheus-amp-graf) for a step-by-step guide.
3. documentation-service: Spring Boot-based REST application consolidating all the Swagger JSON and offering it in a single endpoint.  Please note that this component can be part of a gateway or the registry itself, but I have chosen to keep it separate. The final documentation shall be available at http://localhost:9093/swagger-ui.html.

**1. Setup Eureka server**

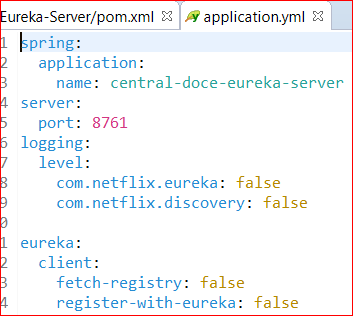
\* dependencies eureka server, web , spring boot version 2.2.6, spring cloud version:Hoston



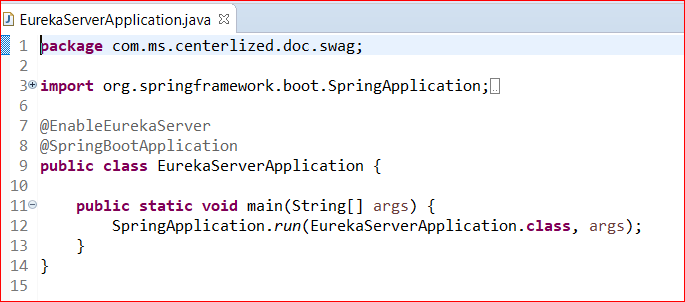


This dependency will be automatically added when you add eureka server dependency.

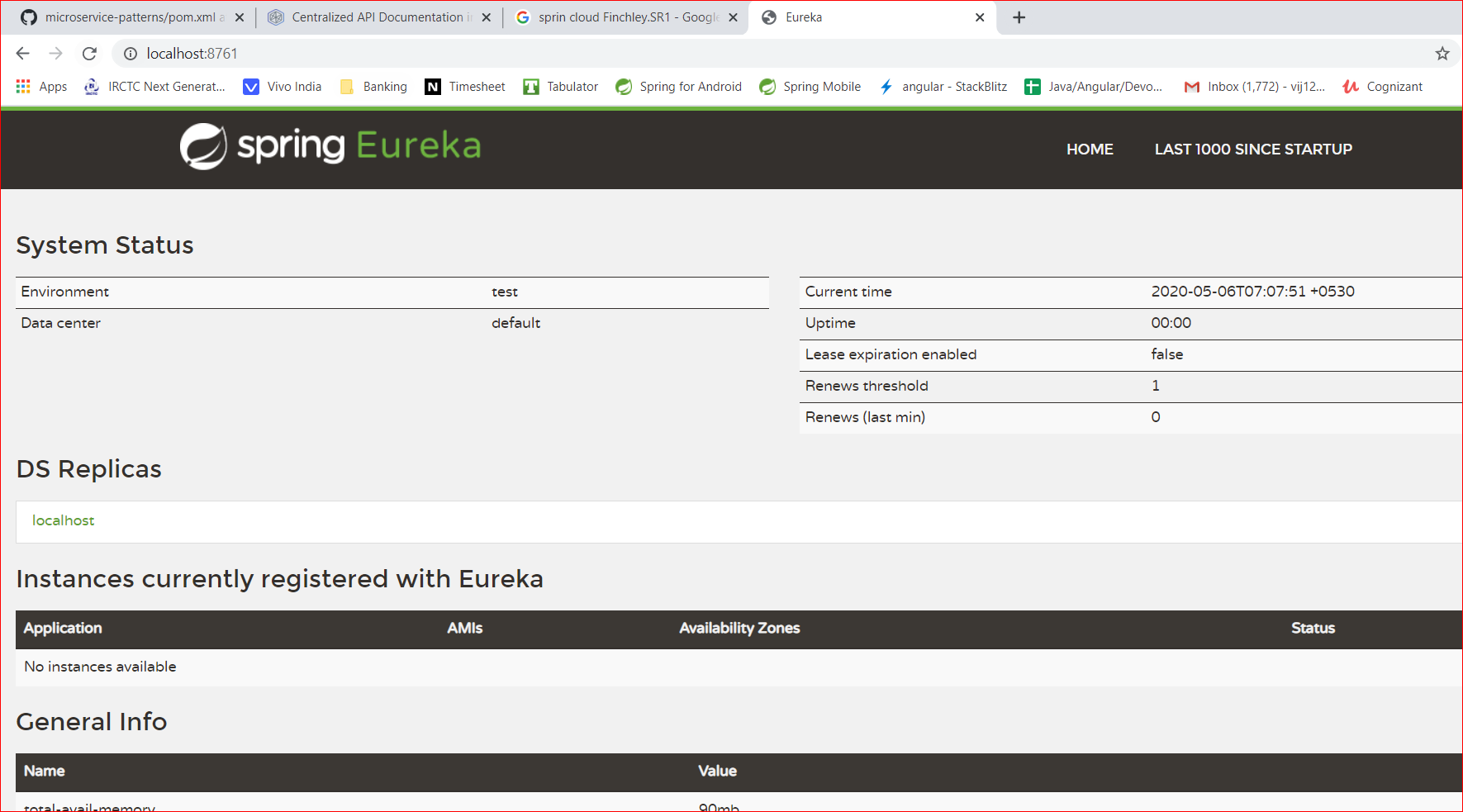
Setup application.yml inside resources (classpath),. Which overrides the default application.properties



Enable eureka server in boot loader file. @EnableEurekaServer



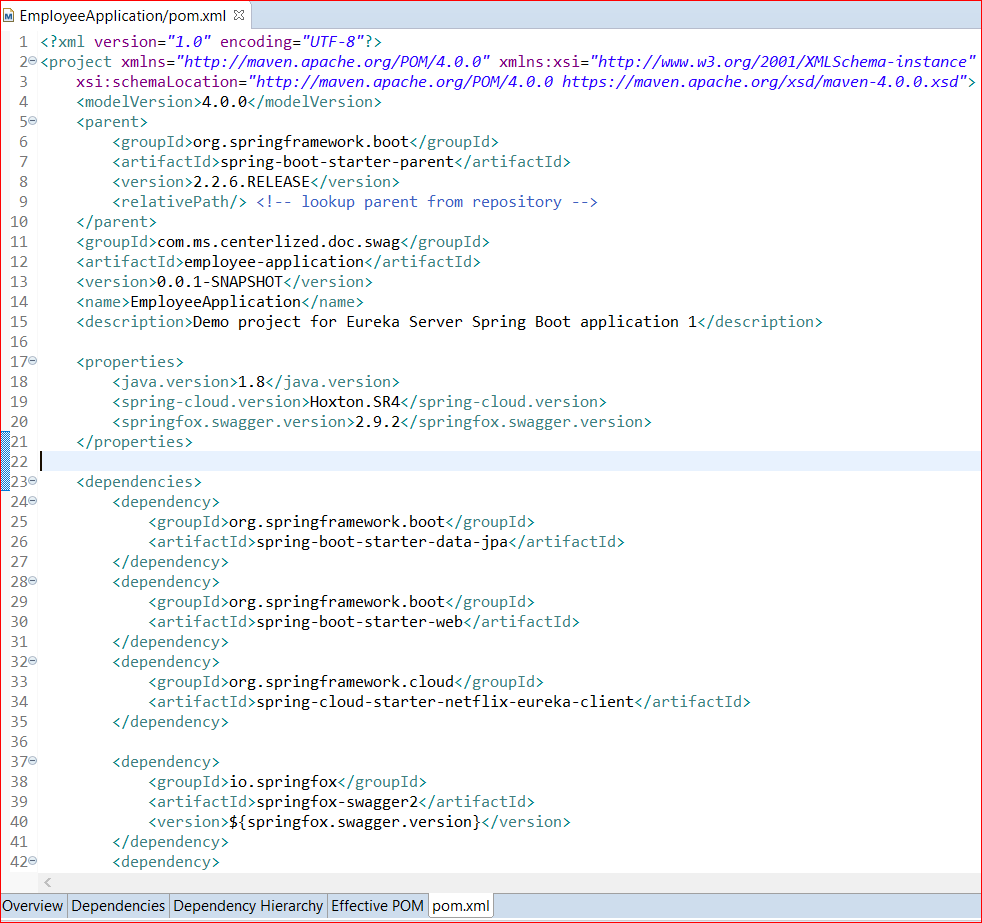
Now start the server and access the page.



**2. Setup the applications**

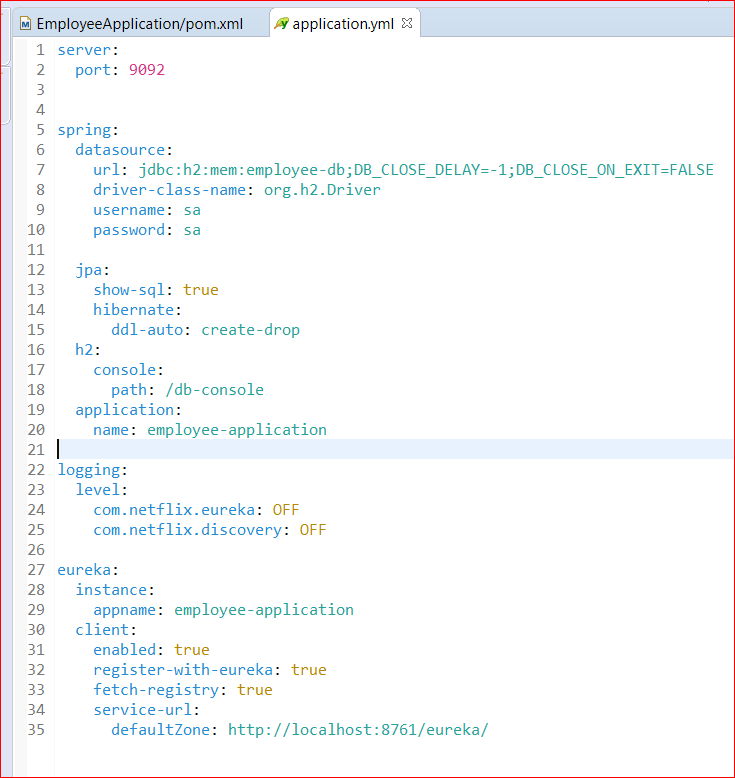
**Employee-application**

Dependencies required are web, eureka client, jpa, h2 database, swagger2, Lombok



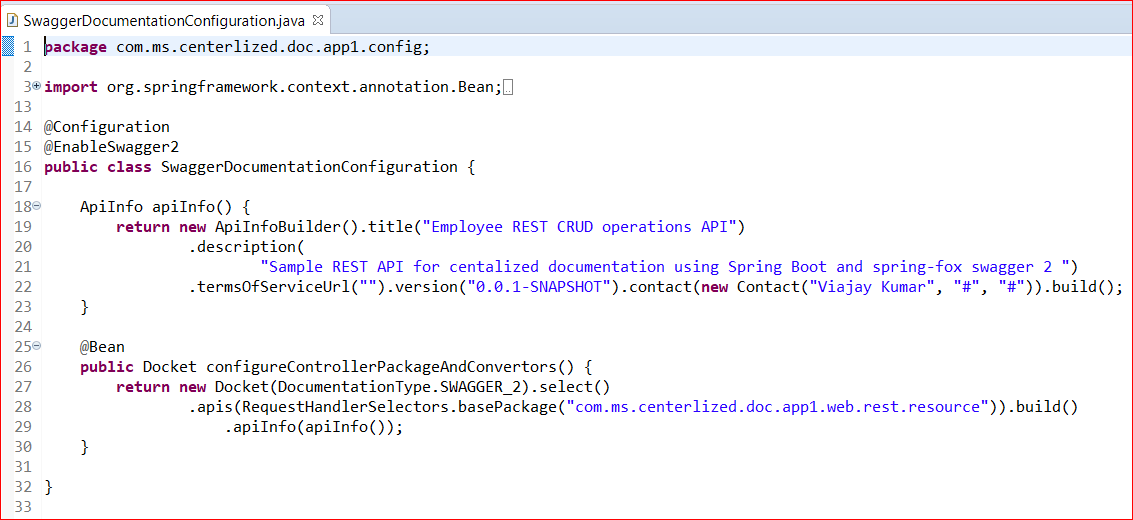


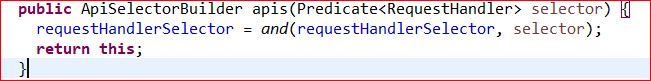
Setup your application.yml

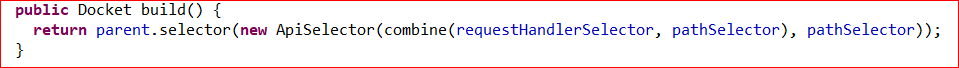


27 – 35 we are providing the configuration for creating the employee application instance and make it register in eureka server by providing the server running url which is 8761

Configure the swagger documentation





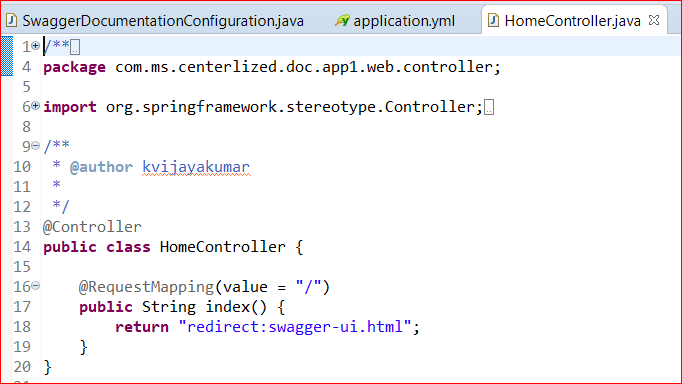


Construct the Api info and pass it to the Docket api builder.

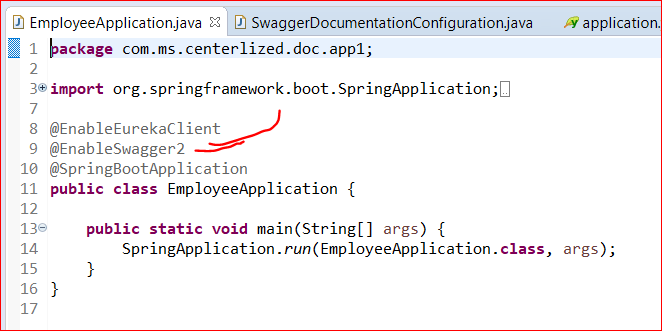
Docket: A builder which is intended to be the primary interface into the Springfox framework.

Provides sensible defaults and convenience methods for configuration.

In Line 28 we are providing the base package rest.resource which has all the endpoint implementations.



Redirects to swgger-ui.html



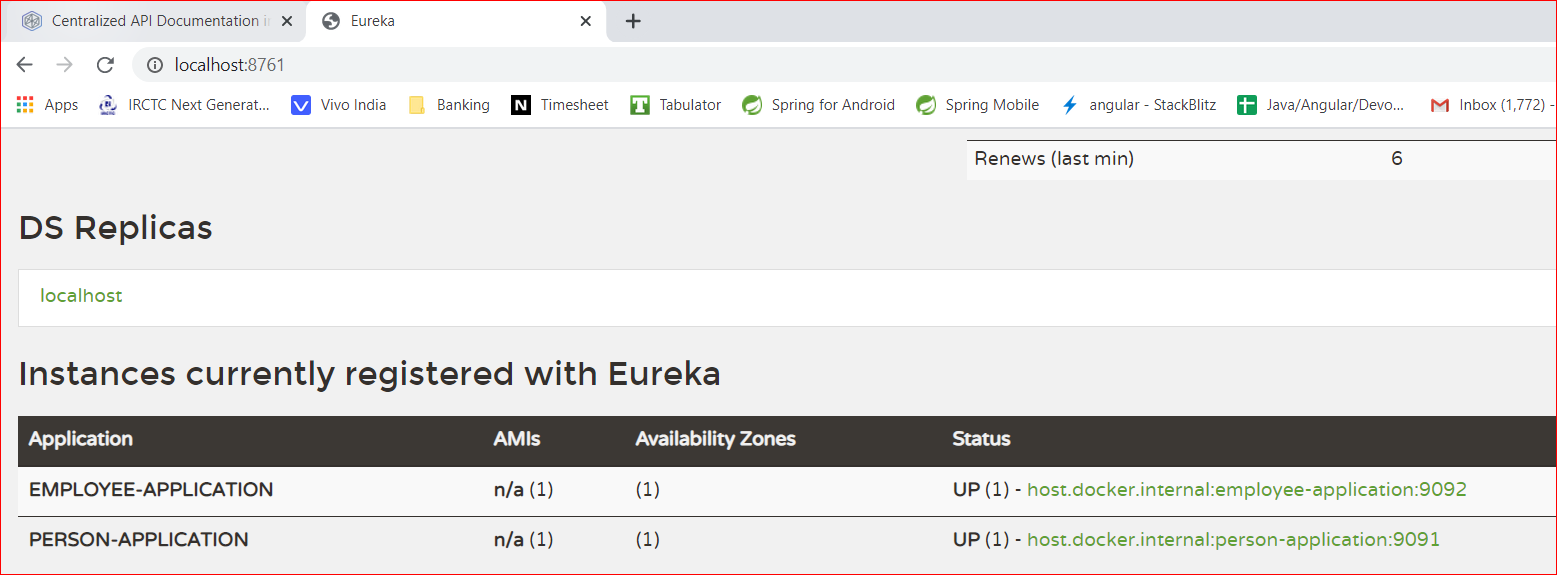
Have to use the annotation EnableEurekaClient and EnableSwagger2

EmployeeResource – Have implemented all the CURD operations



The same steps can be implemented for Person Application as well.

Run the both person and employee micro services and see the Eureka service instances were registered.

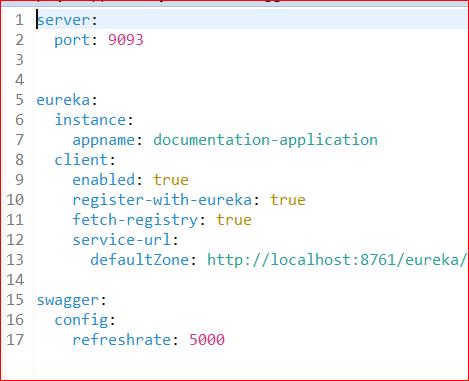


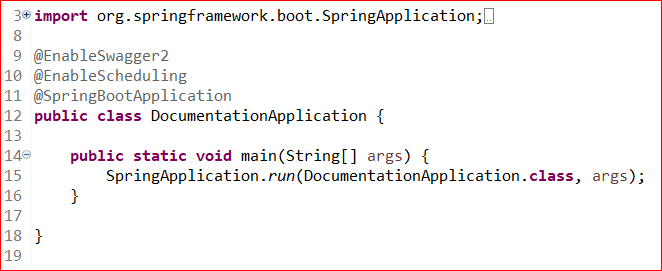
**3.Documentation Application**

Pom.xml

Dependencies: Eureka client and swagger2

Application.yml





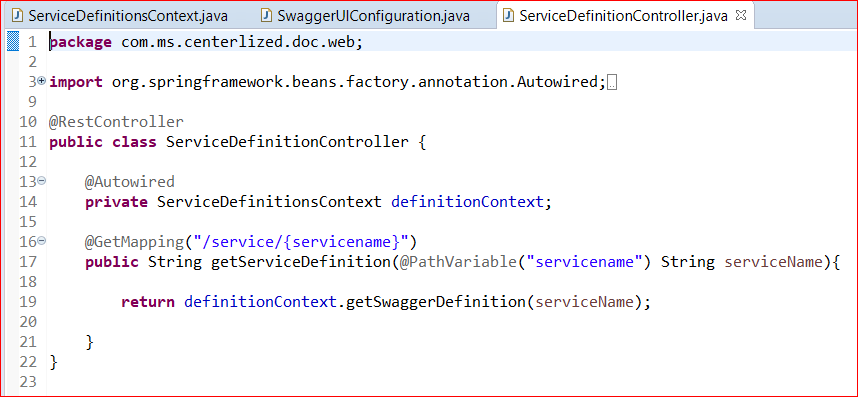
**SwaggerUIConfiguration**

The Spring configuration class registers the instance of SwaggerResourcesProvider, which reads the swagger-api JSON files from our ServiceDefinitionsContext.



**ServiceDefinitionController**

Override the default behavior of the call to the service id and return JSON from ServiceDefinitionsContext as a response.



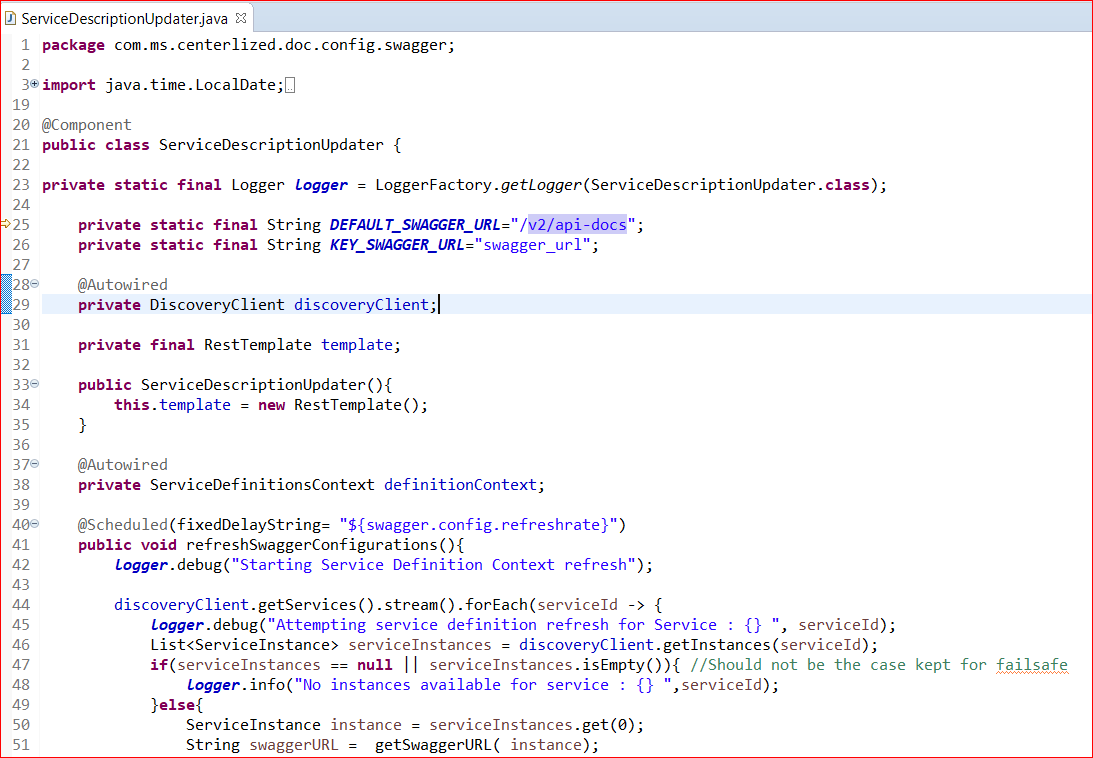
**ServiceDefinitionsContext**

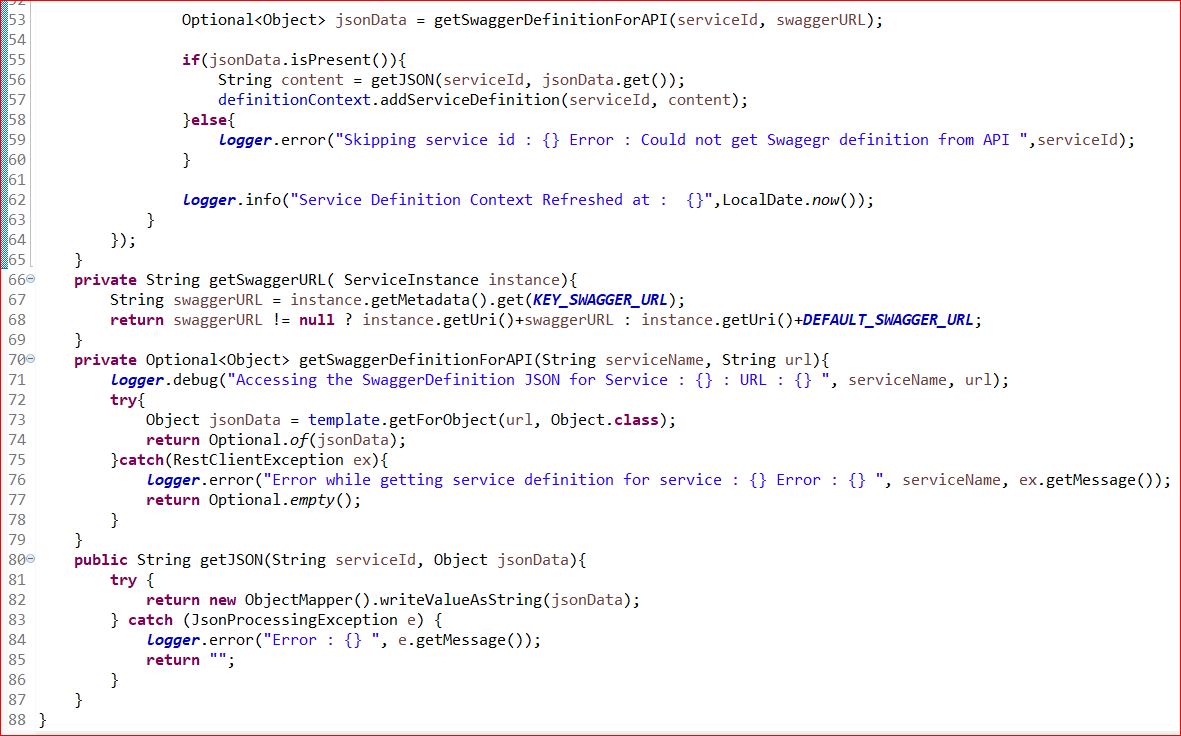
This component serves as an in-memory store for all the Swagger JSON files.



**ServiceDescriptionUpdater**

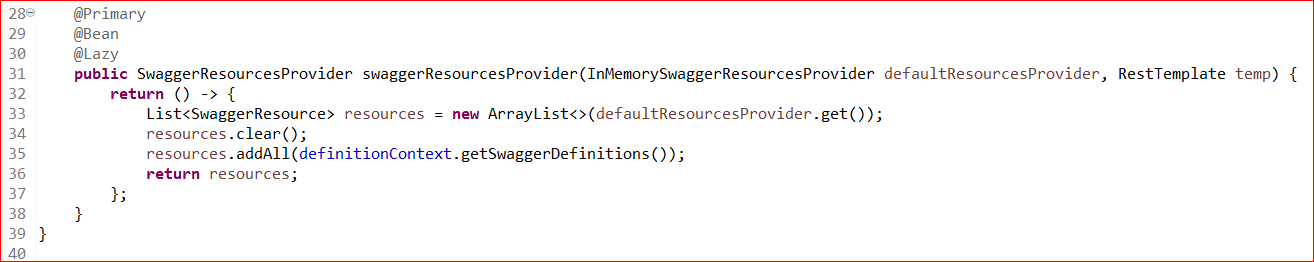
This is the most important component which reads all the registered service instances on Eureka server, polls them for the Swagger definition, and stores them in ServiceDefinitionsContext. By default, the puller will expect the Swagger definitions JSON to be available on the path "http://<Host: IP>:<Port>/v2/api-docs". If you have changed the path of the Swagger JSON URL, you can configure the path as Eureka metadata with the key "swagger\_url" and the updater will look up that path.





Now go to the browser and access documentation application.

Localhost:9093/swagger-ui.html : On load of the application the swagger resources will be loaded.



<http://localhost:9093/swagger-resources>

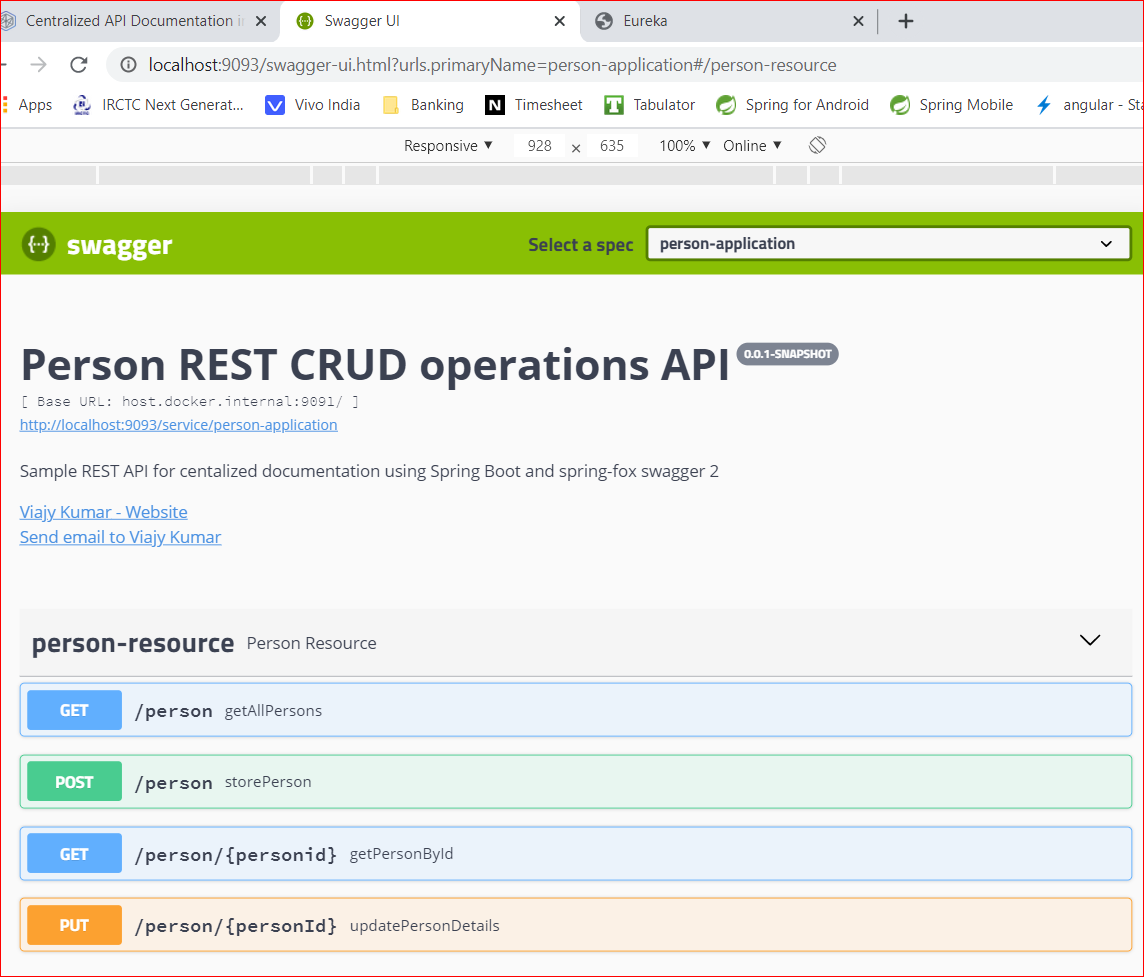
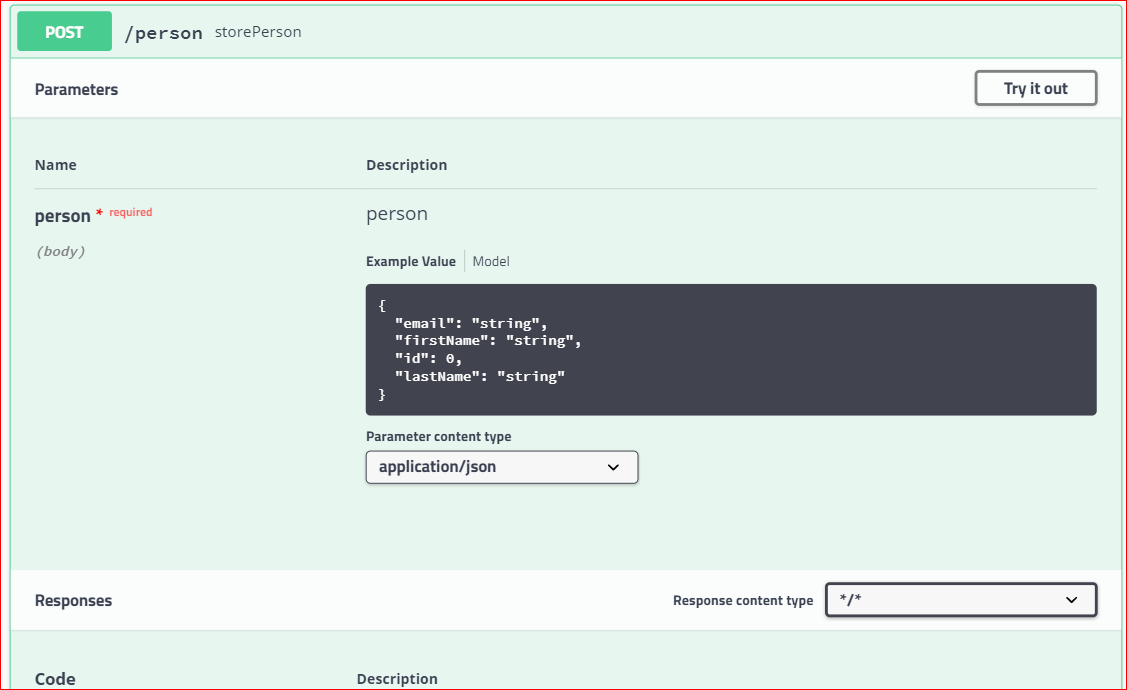
Response:

[{"name":"employee-application","url":"/service/employee-application","swaggerVersion":"2.0","location":"/service/employee-application"},{"name":"person-application","url":"/service/person-application","swaggerVersion":"2.0","location":"/service/person-application"}]

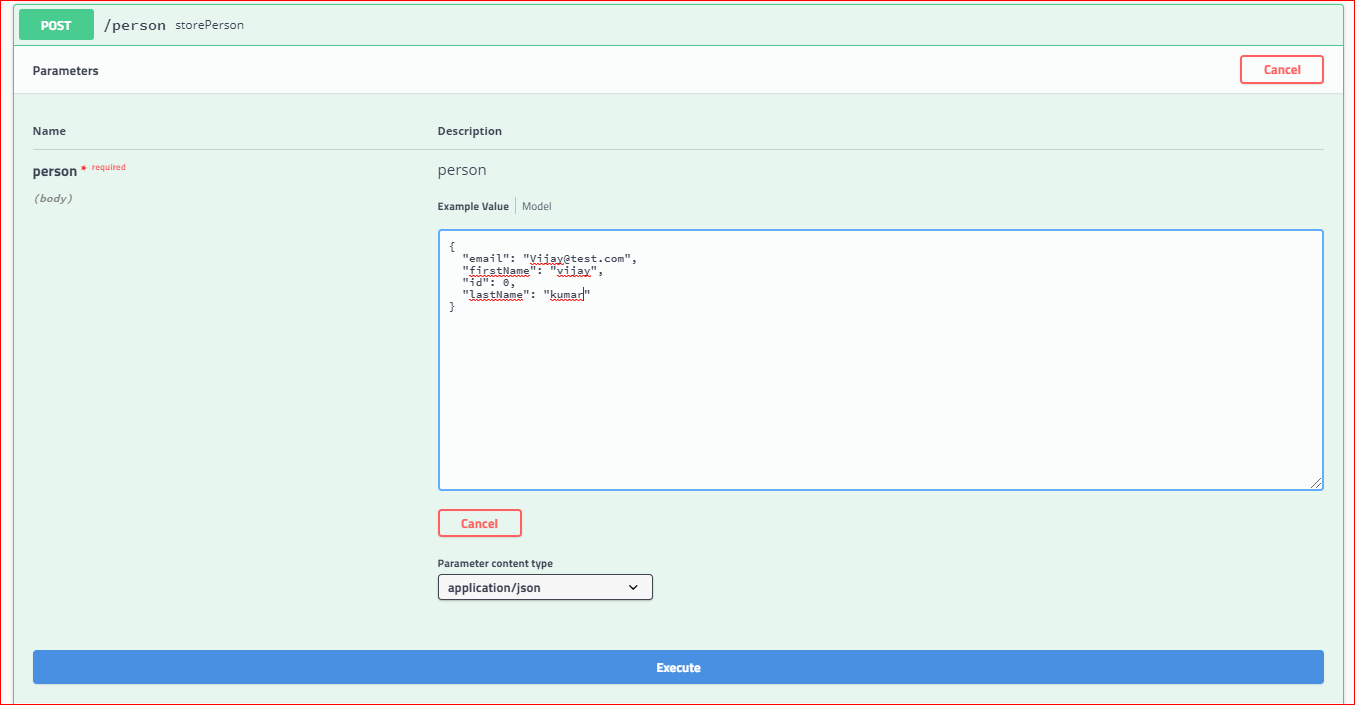
<http://localhost:9093/service/employee-application>

{"swagger":"2.0","info":{"description":"Sample REST API for centalized documentation using Spring Boot and spring-fox swagger 2 ","version":"0.0.1-SNAPSHOT","title":"Employee REST CRUD operations API","contact":{"name":"Viajay Kumar","url":"#","email":"#"}},"host":"host.docker.internal:9092","basePath":"/","tags":[{"name":"employee-resource","description":"Employee Resource"}],"paths":{"/employee":{"get":{"tags":["employee-resource"],"summary":"getAllEmployees","operationId":"getAllEmployeesUsingGET","produces":["\*/\*"],"responses":{"200":{"description":"OK","schema":{"type":"array","items":{"$ref":"#/definitions/Employee"}}},"401":{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}},"deprecated":false},"post":{"tags":["employee-resource"],"summary":"createNewEmployee","operationId":"createNewEmployeeUsingPOST","consumes":["application/json"],"produces":["\*/\*"],"parameters":[{"in":"body","name":"person","description":"person","required":true,"schema":{"$ref":"#/definitions/Employee"}}],"responses":{"200":{"description":"OK","schema":{"$ref":"#/definitions/Employee"}},"201":{"description":"Created"},"401":{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}},"deprecated":false}},"/employee/{employeeid}":{"get":{"tags":["employee-resource"],"summary":"getEmployeeByEmployeeId","operationId":"getEmployeeByEmployeeIdUsingGET","produces":["\*/\*"],"parameters":[{"name":"employeeid","in":"path","description":"employeeid","required":true,"type":"integer","format":"int32"}],"responses":{"200":{"description":"OK","schema":{"$ref":"#/definitions/Employee"}},"401":{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}},"deprecated":false},"put":{"tags":["employee-resource"],"summary":"updateEmployeeById","operationId":"updateEmployeeByIdUsingPUT","consumes":["application/json"],"produces":["\*/\*"],"parameters":[{"in":"body","name":"employeeDataToBeUpdated","description":"employeeDataToBeUpdated","required":true,"schema":{"$ref":"#/definitions/Employee"}},{"name":"employeeid","in":"path","description":"employeeid","required":true,"type":"integer","format":"int32"}],"responses":{"200":{"description":"OK","schema":{"$ref":"#/definitions/Employee"}},"201":{"description":"Created"},"401":{"description":"Unauthorized"},"403":{"description":"Forbidden"},"404":{"description":"Not Found"}},"deprecated":false}}},"definitions":{"Employee":{"type":"object","properties":{"email":{"type":"string"},"employeeFirstName":{"type":"string"},"employeeId":{"type":"integer","format":"int32"},"employeeLastName":{"type":"string"}},"title":"Employee"}}}

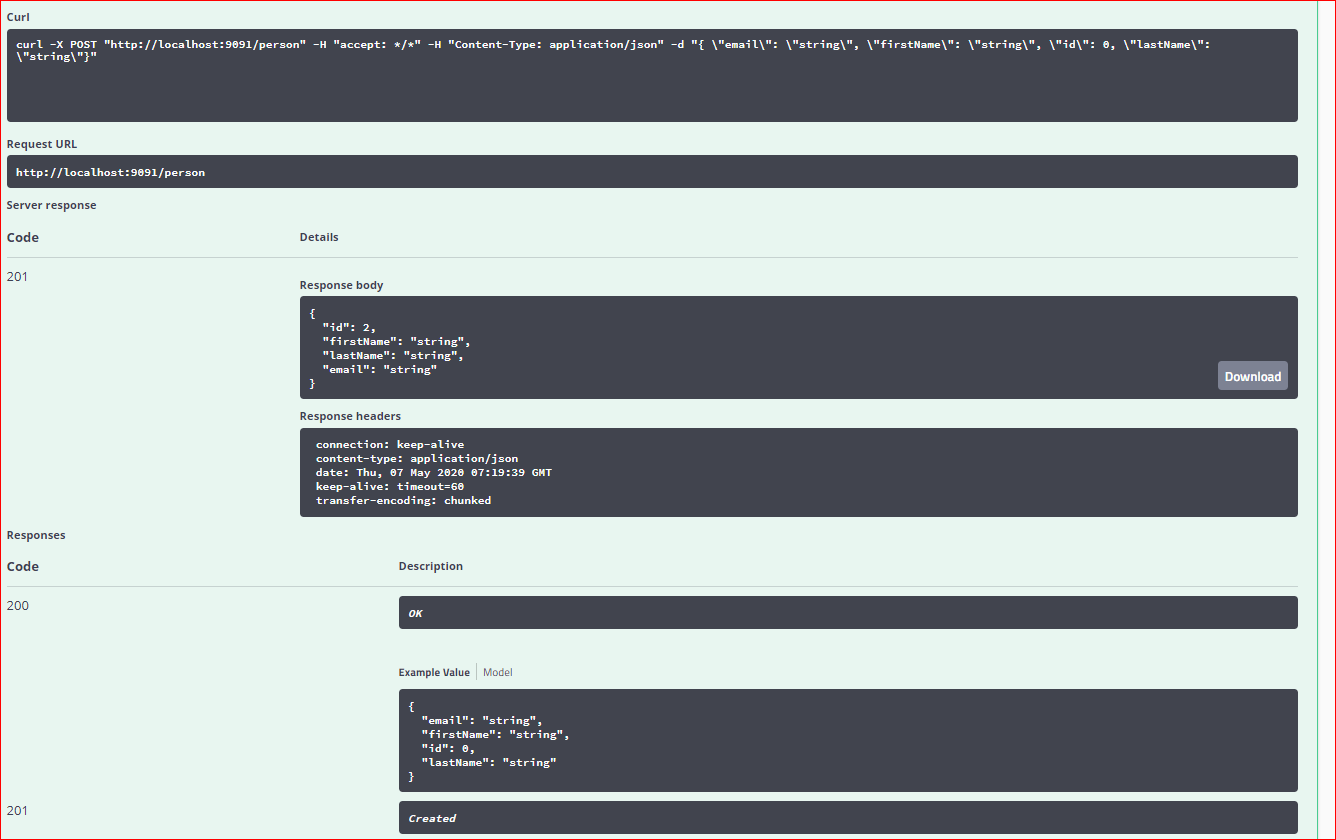
The response is stored in the InMemorySwaggerResourcesProvider as a Swagger resource.

Try it out

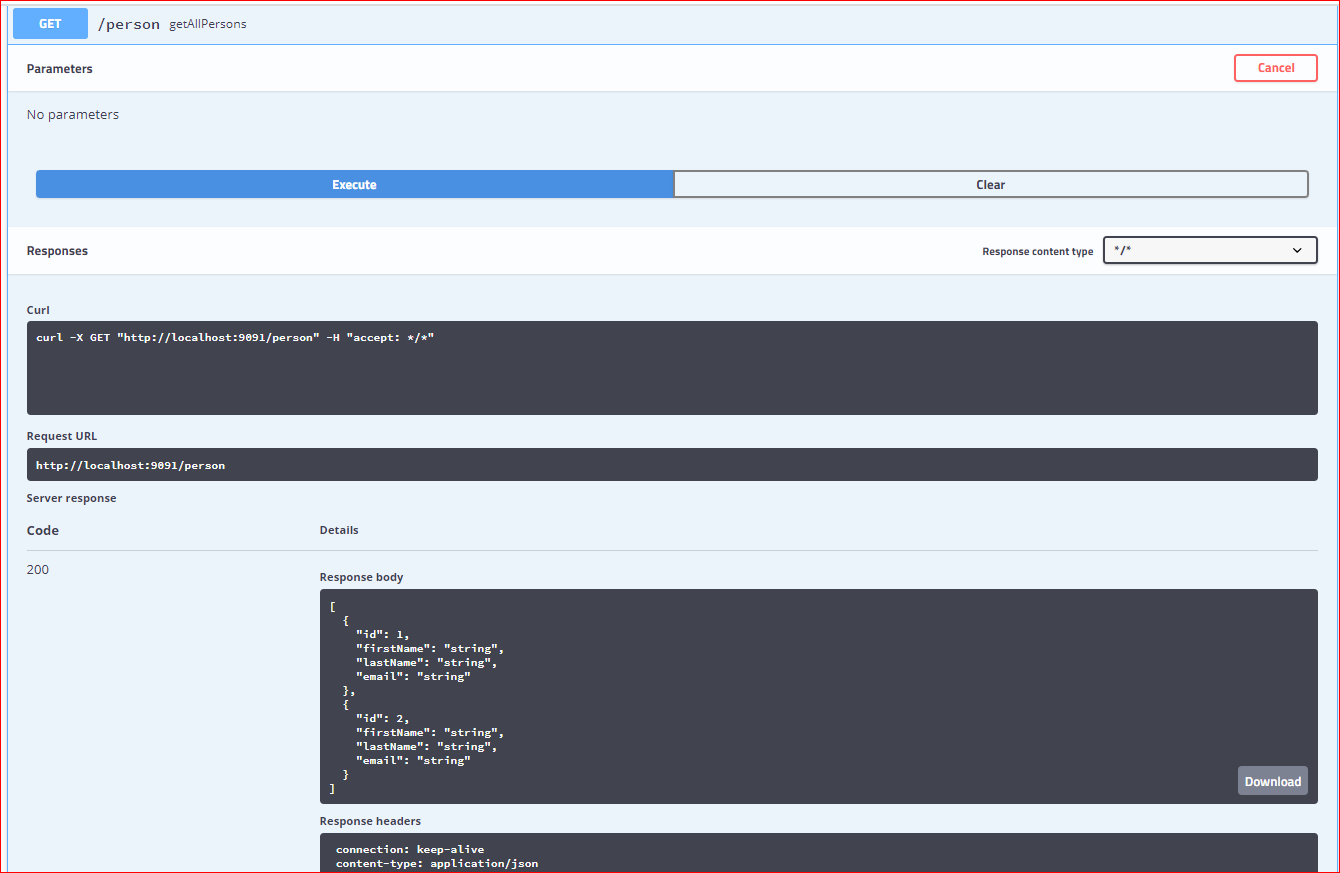


Execute



Person has created successfully

Fetch it from the same swagger ui



Conclusion: As part Microservices pattern the implementation of the centralized documentation using swagger2 was done.

Using the centralized ui we can read the whole information about the all end points were there as part of the individual Microservices in one place. Also we can test each endpoint of it purpose by providing the required inputs in the request body. As shown in the above screenshots.