My project is based on Spring Framework the best document to follow is **https://docs.spring.io/spring/docs/current/spring-framework-reference/**

Now I have implemented the basic structure for the 3 microservices. In Registration microservice a basic Get and Post rest service is implemented as well.

Now I should implement login service.

I will use spring security, OAuth and JWT for authentication, authorization and exception handling.

Note: Do not do the mistake of putting Boot Application java class in package other than the maven or gradle group name.

In config(HttpSecurity http) method where you extend WebSecurityConfigrerAdapator if you do not want default spring’s login page the do not add .formLogin() in the http.. chain of antMatchers().

Hey “Optional” in java 8 is basically to avoid NullPointerException. It is just a wrapper around the class it refers. With helper methods and support of functional programming abilities of java8.

Note about @Autowired and @Bean

By @Bean we tell the spring to manage this class so that whenever I ask for it give it to me.

@Autowired is asking for a managed class or bean.

Struggled to get Authorization part of spring working.

Steps

1: extend for the @Configuration class were we have @WebsecurityConfigurer with WebSecurityConfigurerAdaptar.

2: here in this class we can use the overloaded config method for HttpSecurity as argument were we give antMatchers.

3: to tell spring what roles are available for a user we use argument in one of the config methods AuthenticationManagerBuilder and we call our Custom UserService which has extends Springs provided UserService. There the important method is loadByUserName, in here we actually return the particular user.

4: important note is the overridden getAuthorities method need to be provded with the actuall Role text appended with “Role\_”

Now my plan is to start working on other 2 microservices. The registration and login microsrvice structure is almost complete. This service will be used by the other two services. The login is a token mechanism, so the registration of donor and host will create respective roles for them. First time when they register will create their login id and password that will be send to them on their mobile number and email. After this when they try to login, they actually interact with the login microservice, which authenticate and generate token for them and redirect thm to the home page of their respective microservice. The token will also be send to the respective microservice so that the request and response cycle between the user and microservices remains seamless.

I plan to store the token info in cache usng redis cache.

Note : The messaging model is Publish/Subscriber model.

I will use RabbitMQ for inter-microservices communication, also will use RabbitMQ container on docker.

The thing to note is that spring provide this under their project of spring cloud stream it is not spring cloud bus. Spring cloud bus is a different project that is used for changing configuration of the microservices remotely. It has @refresh annotation for that purpose.

I also plan to use different db for each microservice.

Follow the link for inter-microservice communication

<https://piotrminkowski.wordpress.com/2018/06/15/building-and-testing-message-driven-microservices-using-spring-cloud-stream/>

I need to upscale the messaging service because I will be using multiple instances of each microservice.

To run RabbitMQ execute following steps,

Open an elevated command line (Run as Administrator)

Navigate to the sbin directory of the RabbitMQ Server installation directory. In my case the path is C:\Program Files (x86)\RabbitMQ Server\rabbitmq\_server-3.3.4\sbin

Run the following command to enable the plugin rabbitmq-plugins.bat enable rabbitmq\_management

Then, re-install the RabbitMQ service using the commands below:

rabbitmq-service.bat stop

rabbitmq-service.bat install

rabbitmq-service.bat start

to run on docker just run the following command,

docker run -d --hostname my-rabbit-msg --name majnu-rabbit micdenny/rabbitmq-windows

check once RabbitMQ server is up <http://127.0.0.1:15672>

Starting with the microservices first we need to build config server.

Config server is working and client service is able to communicate.

The use of spring config server is a central versioned control over the configuration need by client microservices. We use to maintain configuration in properties files or environment variables. These was not versioned controlled and main drawback was that the system needed to be restarted.

As per microservice principals : **Build just once and deploy it in any environment.**

This is a violation of the principal.

Spring cloud provide as a way to store the configuration in a central place which can be changed without any need to restart the system.

The donor microservice will have the functionality of send request of donating toys. It will be a submit request with no. of toys and the pickup location.

This will go into donor-request table. This table will have a reference of donor’s user id. As per microservice principle I am planning to have separate db for each service. Once the request is submitted the request will appear on donors home page with the status as pending. This request will be received by Admin service, he can approve it and the status should change to pickup confirmed.

The records on the Admin’s home page will be fetched through RabbitMQ.

Spring Authentication Provider: how it works,

In the configuration file we override configure method with AuthenticationManagerBuilder, this class can be used to add custom authentication builders like jwt.

AuthenticationProvider has authenticate method that returns Authenticate object. Which in turn has getAuthorities and other methods to be verified for authentication.

I have got token detokenize and able to save the in the local dv of donor microservice. Now I should be able to store in cache.

Redis port while setting up on windows is 6379. To run this we need to go to the folder Redis in Program Files. Execute command >redis-srve.exe /redis.windows-service.conf

Then execute >redis-cli.exe

Then say >KEYS pattern

Refer for the above instruction this link <http://googleda.com/2018/02/16/redis-server-and-cli-for-windows/>

There is no need to write code to fetch from cache, it happens automatically.

When I add an Authorization header for token then follow pattern as suggested by W3C in HTTP 1.0.

Authorization: <type> <credentials>

So as per OAuth 2.0 Authorization framework we must include “Bearer” as type.

Now I start building admin microservice there are couple of things involved, connection to RabbitMQ so that it can talk with other microservices, token based authorization logic, etc.,

Swagger if implemented on Donor services. It is a good tool for documenting apis. It provides its own ui page. There is swagger-editor for designing your apis and import in json format.

@PostConstruct method called after dependency injection

@PreDestroy method called prior to destroying the bean

The above beans must not take any parameter and must return void.

Now, I need to start with registering services in eureka. Then ribbon, feign and hystrix need to be implemented.

On hiting from my ui-project I noticed the CORS error. It is basically the resource not allowing to access any request from outside(SAME-ORIGIN POLICY). This issue emerges in distributed applications. To deal with it we need to let resource allow access to a request from outside. We can restrict calls from outside based on the http protocol methods i.e., ‘GET’, ‘POST’, ‘PUT’, etc.,

Using Zuul proxy we can provide configuration for cors for all of the microservices registered to it.

The token can also be stored in cookies. It is safer to store tokens in cookies than to store in web-storage or session. Because in the later case it becomes vulnerable to XSS (cross-site scripting attack), however even with cookies it can be attacked by CSRF attack. In session or web-storage javascript can be used to access the data. So I decide to use cookies for token storage.

We got app request hit reaching the apis. For this e have to make couple of changes in zuul proxy.

This is regarding getting the Authorization header passed in response from the apis.

First **zuul**:  
 **sensitiveHeaders**: Cookie,Set-Cookie,Authorization

Second: in cors filter it self config. setExposedHeaders

V V Imp!

So I had some gaps in my knowledge regarding how cookies work. Putting it here so that others can benefit from this.

1. HTTP ONLY (Secure) cookies cannot be accessed in JavaScript. If you try to read some token, etc from a secure cookie it's not going to work.
2. Cross-domain cookies cannot be accessed. In case you are building a single page application and your server is on a different domain. You cannot access the cookies on your SPA. The withCredentials flag will just make sure that the network calls made include the cookies and accept any incoming cookies.

Now, I am really stuck at the overall architecture of microservices. How the user registration and login process should work?

There are some good read regarding this <https://medium.com/tech-tajawal/microservice-authentication-and-authorization-solutions-e0e5e74b248a>

Still mere the axios call I make to get asynch data is not sending the cookie back to the service.

Caught the train too late, though for CORS Mozilla has good documentation read this,

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

so finally I got the cookie header relayed back to the resource server. This I did in zuul pre filter. In future versions of spring this should be resolved.

There is confusion between zuul proxy and ribbon. From this link there is some clarity I can get <https://stackoverflow.com/questions/43538030/zuul-and-ribbon-integration>

That ribbon is built in the zuul proxy. Ribbon is a client side load balancer but does not directly interact with your UI component. Zuul proxy server-side load balancer from UI perspective.

Zuul uses RibbonRoutingFilter to choose from the list of services provided from Eureka. RibbonRoutingFilter internally uses Ribbon.

Still you need to make Ribbon client configuration if you want to make call between microservices.

Now come across a new pattern called Circuit Breaker Pattern. In microservices scenario we can come across a suitation where one of the services is down for long time. In this case we need to handle by calling a fall back method. In Spring cloud, we rely on the Netflix’s Hystrix implementation. In this pattern we can give the configuration about the number of calls that failed to reach threshold to open the circuit and subsequent call fallsback without calling failing service. Also we can set the threshold of wait time and try again to check if the service is up and running.

In spring cloud, we wrap the calling code with @HystrixCommand annotation so that in case of serval failure attempts the fallback method is called. We also need to @EnableCircuitBreaker to tell spring that we want to implement the circuit breaker pattern.

Start from here reading this

<https://dzone.com/articles/microservices-part-4-spring-cloud-circuit-breaker>

all the configuration options can be found here <https://github.com/Netflix/Hystrix/wiki/Configuration>

the method called from @HystrixCommand is called on different thread, to make the threadlocal variable available we can configure it as **execution.isolation.strategy=SEMAPHORE**

**HystrixConcurrencyStrategy can be used to pass any data to @HytrixCommand.**

**We can also implement Hystrix dashboard as a separate spring boot project to monitor the calls success and failure. Now instead of having separate service for each service we can make use of Turbine to have unified view of all the service enabled with circuit-breaker.**

**Start with feign**