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.