**API Documentation Guidelines**

Objective

The main purpose of API documentation is to facilitate the integration effort both within and also outside of the organization. With the adoption of microservices, there will be an increase in the number of APIs interacting with one and another, therefore, a proper API documentation practice is required so each of our API comes with accurate and up-to-date documentation that enables seamless integration work.

Swagger and Swagger UI

Swagger is an open source framework that contains a set of tools to help developers design, build, and document RESTful APIs. Automatically generating and maintaining API documentation allows developers to spend more effort on writing good code instead of manually updating API documentation for every code change. Additionally, Swagger UI allows any party to visually examine and interact with a live API, all without the need of writing a single line of code.

Set up in Spring Boot

The following steps show how to enable Springfox Swagger and Springfox Swagger UI in a Spring Boot application:

1. Import dependency

* Maven dependency

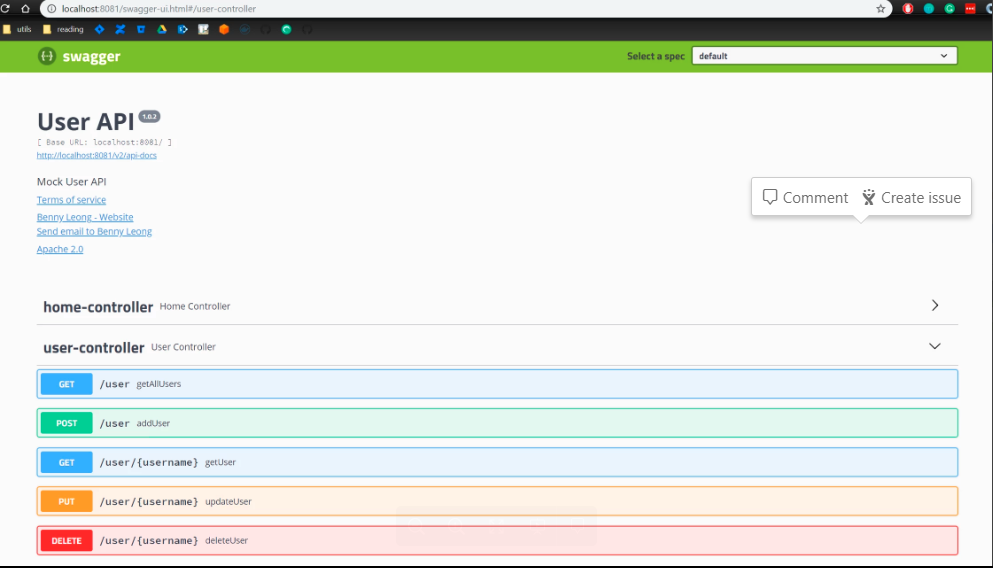
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->  
<dependency>  
 <groupId>io.springfox</groupId>  
 <artifactId>springfox-swagger2</artifactId>  
 <version>2.9.2</version>  
</dependency>  
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->  
<dependency>  
 <groupId>io.springfox</groupId>  
 <artifactId>springfox-swagger-ui</artifactId>  
 <version>2.9.2</version>  
</dependency>

1. Create a new class to configure Swagger

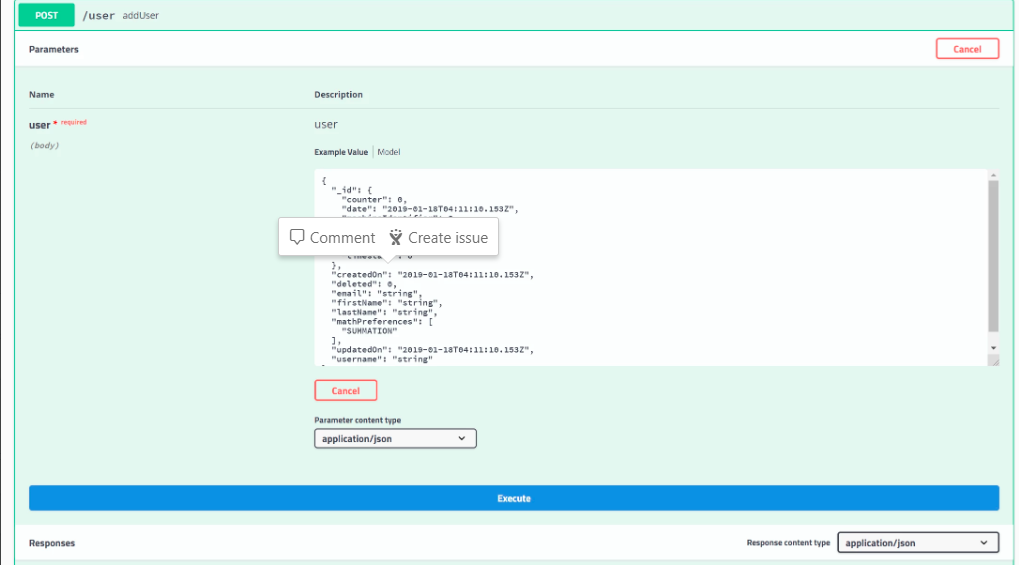
* Swagger configuration

@Configuration  
@EnableSwagger2  
public class SwaggerConfig extends WebMvcConfigurationSupport {  
 public static final Contact DEFAULT\_CONTACT = new Contact(  
 "Benny Leong", "https://amtrustmobilesolutions.asia", "benny.leong@amtrustmobilesolutions.asia");  
  
 public static final ApiInfo DEFAULT\_API\_INFO = new ApiInfo(  
 "User API", "Mock User API", "1.0.2",  
 "urn:tos", DEFAULT\_CONTACT,  
 "Apache 2.0", "http://www.apache.org/licenses/LICENSE-2.0", new ArrayList<>());  
  
 private static final Set<String> DEFAULT\_PRODUCES\_AND\_CONSUMES =  
 new HashSet<>(ImmutableList.of("application/json"));  
  
 @Bean  
 public Docket productApi() {  
 return new Docket(DocumentationType.SWAGGER\_2)  
 .apiInfo(DEFAULT\_API\_INFO)  
 .produces(DEFAULT\_PRODUCES\_AND\_CONSUMES)  
 .consumes(DEFAULT\_PRODUCES\_AND\_CONSUMES)  
 .select().apis(RequestHandlerSelectors.basePackage("com.amtrust.userapi.controller"))  
 .build();  
  
 }  
  
 @Override  
 protected void addResourceHandlers(ResourceHandlerRegistry registry) {  
 registry.addResourceHandler("swagger-ui.html")  
 .addResourceLocations("classpath:/META-INF/resources/");  
  
 registry.addResourceHandler("/webjars/\*\*")  
 .addResourceLocations("classpath:/META-INF/resources/webjars/");  
 }  
}

1. With the configuration ins Step 2, Swagger knows the base package ("com.naman.userapi.controller") where controllers are located and will generate the corresponding API specification for each of the controller found in the package https://<application-url>//swagger-ui.html.
2. Once the application is up and running, the API documentation can be located at:-



1. Users will be able to interact with each of the controller endpoints directly through Swagger UI page



1. The API documentation will automatically reflect any latest code changes once the application is redeployed (i.e.: zero effort required by the developers to maintain the documentation).

**Non-Java API**

Both Swagger and Swagger UI are not restricted to applications built on Java environment only. All other APIs written in non-Java language should include the related Swagger dependencies for API documentation