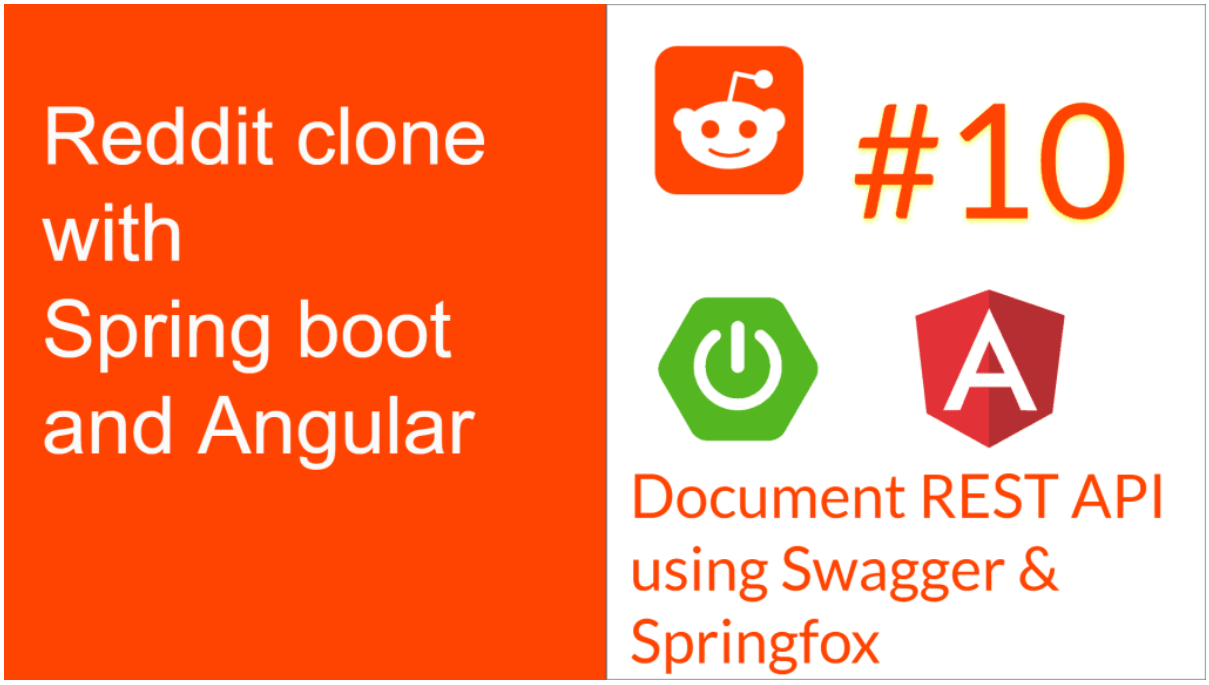
PROGRAMMING TECHIE

Build a Full Stack Reddit Clone with – Spring boot and Angular – Part 10

0 Comments

BY SAI UPADHYAYULA | March 23, 2020



Share Tweet Pin

Welcome to Part 10 of Build a Full Stack Reddit Clone with Spring boot and Angular series, and in this article, we will see how to document the REST API using Swagger and Springfox.

In Part 9, we already started developing our frontend application using Angular. Our frontend application will be consuming the REST API's which we developed in the previous tutorials.

If you are a visual learner like me, check out also the Video Tutorial:

Download Source Code

> **Source code for Angular application** – https://github.com/SaiUpadhyayula/angular-reddit-clone
>
> **Source code for Spring boot backend** – https://github.com/SaiUpadhyayula/spring-reddit-clone

Why should we document our REST APIs?

In the real world, consumers of an API should have a good understanding of the REST APIs they are using. Having good documentation is vital in helping the users to use the API effectively.

In our scenario, as a frontend developer using a framework like Angular, we need to build components in our application that communicates with the REST API frequently.

In this case, having good documentation for our REST APIs is necessary. On the other hand, maintaining the documentation manually is tiresome and error-prone.

Generating REST API Documentation using Swagger and Springfox

Swagger and Springfox makes this process of generating REST API documentation quick and painless. Using these tools, we can automate the process of documentation.

What is Swagger?

So what is Swagger? It is an OPEN API specification that is created as a standard to describe your REST API.

As we are using Springboot to develop our REST API we can use a library called as Springfox to automatically create JSON Documentation.

Adding Springfox Dependencies to project

Inside our pom.xml file, add the following maven dependencies. This should download the required springfox dependencies to our project.

```
<!-- Spring Fox Dependencies -->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
```

```
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>
```

This is how our complete **pom.xml** file should look like after adding the dependencies

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.8.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>spring-reddit-clone</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>spring-reddit-clone</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
        <org.mapstruct.version>1.3.1.Final</org.mapstruct.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.8</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
```

```
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <!-- JWT related dependencies-->
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-api</artifactId>
            <version>0.10.5</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-impl</artifactId>
            <scope>runtime</scope>
            <version>0.10.5</version>
        </dependency>
        <dependency>
            <groupId>io.jsonwebtoken</groupId>
            <artifactId>jjwt-jackson</artifactId>
            <scope>runtime</scope>
            <version>0.10.5</version>
        </dependency>
        <dependency>
            <groupId>org.mapstruct</groupId>
            <artifactId>mapstruct</artifactId>
            <version>${org.mapstruct.version}</version>
            <scope>compile</scope>
        </dependency>
        <!-- Spring Fox Dependencies -->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>
        <!-- For Displaying time as Relative Time Ago ("Posted 1 Day ago"),
         as this is a Kotlin library, we also need Kotlin runtime libraries-->
        <dependency>
            <groupId>com.github.marlonlom</groupId>
            <artifactId>timeago</artifactId>
            <version>4.0.1</version>
        </dependency>
        <dependency>
            <groupId>org.jetbrains.kotlin</groupId>
            <artifactId>kotlin-stdlib-jdk8</artifactId>
            <version>${kotlin.version}</version>
        </dependency>
        <dependency>
            <groupId>org.jetbrains.kotlin</groupId>
            <artifactId>kotlin-test-junit</artifactId>
            <version>${kotlin.version}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
```

```
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.5.1</version> <!-- or newer version -->
                <configuration>
                    <source>1.8</source> <!-- depending on your project -->
                    <target>1.8</target> <!-- depending on your project -->
                    <annotationProcessorPaths>
                        <path>
                            <groupId>org.mapstruct</groupId>
                            <artifactId>mapstruct-processor</artifactId>
                            <version>${org.mapstruct.version}</version>
                        </path>
                        <path>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                            <version>1.18.8</version>
                        </path>
                    </annotationProcessorPaths>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.jetbrains.kotlin</groupId>
                <artifactId>kotlin-maven-plugin</artifactId>
                <version>${kotlin.version}</version>
                <executions>
                    <execution>
                        <id>compile</id>
                        <phase>process-sources</phase>
                        <goals>
                            <goal>compile</goal>
                        </goals>
                        <configuration>
                            <sourceDirs>
                                <source>src/main/java</source>
                                <source>target/generated-sources/annotations</source>
                            </sourceDirs>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>

</project>
```

Configure Swagger and Springfox

Now it's time to configure Swagger and Springfox in our project, for that we will create a configuration class called **SwaggerConfiguration**.

This class is marked with annotation **@Configuration** and **@EnableSwagger2**

SwaggerConfiguration.java

```
package com.example.springredditclone.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;
```

```
@Configuration
@EnableSwagger2
public class SwaggerConfiguration {

    @Bean
    public Docket redditCloneApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.any())
                .paths(PathSelectors.any())
                .build()
                .apiInfo(getApiInfo());
    }

    private ApiInfo getApiInfo() {
        return new ApiInfoBuilder()
                .title("Reddit Clone API")
                .version("1.0")
                .description("API for Reddit Clone Application")
                .contact(new Contact("Sai Upadhyayula", "http://programmingtechie.com", "xyz@email.com"))
                .license("Apache License Version 2.0")
                .build();
    }
}
```

Inside the **SwaggerConfiguration.java** class, we created a Bean with the name **redditCloneApi**, this can be anything you like.

Inside the bean, we are creating a new **Docket** which is a Springfox internal class and we are specifying the Documentation Type as Swagger2, everything which is returned inside the redditCloneAPI() method is the standard defaults for **Springfox**.

We have also some API information through the **apiInfo()** method.

Now let's import this configuration file to our **RedditCloneApplication**

```
package com.example.springredditclone;

import com.example.springredditclone.config.SwaggerConfiguration;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Import;
import org.springframework.scheduling.annotation.EnableAsync;

@SpringBootApplication
@EnableAsync
@Import(SwaggerConfiguration.class)
public class SpringRedditCloneApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringRedditCloneApplication.class, args);
    }

}
```

We have added **@Import(SwaggerConfiguration.class)** to our **SpringRedditCloneApplication**. This should enable Swagger and Springfox in our application. Now let's see how this all works.

How Springfox works?

Springfox scans our backend application and looks for all the Controllers and related com

application and it automatically generates the documentation for our REST API.

Using **springfox-swagger-ui**, it constructs a webpage where we can see the documentation for our REST API.

Once we start the application, we can see the documentation at **http://localhost:8080/swagger-ui.html**

But if you do open that URL, you can just see a 403 error page because we have to bypass the **/swagger-ui.html** path inside Spring Security, or else we have to provide an access token. For now, let's exclude this path and some other paths which are responsible to display us the documentation from our **SecurityConfig** class. Let's add the below code to our **SecurityConfig**

```
.antMatchers("/v2/api-docs",
        "/configuration/ui",
        "/swagger-resources/**",
        "/configuration/security",
        "/swagger-ui.html",
        "/webjars/**")
.permitAll()
```

The **SecurityConfig.java** class should look like below after adding the above code snippet.

```
package com.example.springredditclone.config;

import com.example.springredditclone.security.JwtAuthenticationFilter;
import lombok.AllArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.BeanIds;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticati

@EnableWebSecurity
```

```
@AllArgsConstructor
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private final UserDetailsService userDetailsService;
    private final JwtAuthenticationFilter jwtAuthenticationFilter;

    @Bean(BeanIds.AUTHENTICATION_MANAGER)
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Override
    public void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity.csrf().disable()
                .authorizeRequests()
                .antMatchers("/api/auth/**")
                .permitAll()
                .antMatchers(HttpMethod.GET, "/api/subreddit")
                .permitAll()
                .antMatchers("/v2/api-docs",
                        "/configuration/ui",
                        "/swagger-resources/**",
                        "/configuration/security",
                        "/swagger-ui.html",
                        "/webjars/**")
                .permitAll()
                .anyRequest()
                .authenticated();
        httpSecurity.addFilterBefore(jwtAuthenticationFilter,
                UsernamePasswordAuthenticationFilter.class);
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder authenticationManagerBuilder) throws Exception {
        authenticationManagerBuilder.userDetailsService(userDetailsService)
                .passwordEncoder(passwordEncoder());
    }

    @Bean
    PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

Now if you restart the application and go to **http://localhost:8080/swagger-ui.html** you can see something like below:

Subscribe now to get the latest updates!

Name Email Sign Up

As you can see we can find very detailed documentation of our API, with just adding som
Configuration class. Springfox provides very good defaults so that without writing much code you have a very good

documentation setup in place already.

You can also get the documentation in the raw JSON format by accessing the http://localhost:8080/v2/api-docs URL

Conclusion

So that's it for this short article, in the next article as promised we will start working on the Frontend of our application, by implementing the Login and Signup functionalities.

I hope you are enjoying this Fullstack Reddit Clone with Springboot and Angular series, I will see you in the next article, until then **Happy Coding Techies** 🙂

About the author
Sai Upadhyayula