What is REST and RESTFUL

REST represents REpresentational State Transfer
RESTFUL web services are web services that follows REST architectural concept (stateless client-server architecture).
Its is an architectural style for developing applications that can be accessed over the network

What is a REST Resource?

Every content in the REST architecture is considered a resource.

The resource == object in the object-oriented programming world.
Can be represented as text files, HTML pages, images, or any other dynamic data.

The REST Server provides access to these resources whereas the REST client consumes (accesses and modifies) these resources. Every resource is identified globally by means of a URI

What is URI?

Uniform Resource Identifier is the full form of URI which is used for identifying each resource of the REST architecture. URI is of the format:

<protocol>://<service-name>/<ResourceType>/<ResourceID>

https://en.wikipedia.org/wiki/Representational_state_transfer

They are of2 types:
URN
URL

What are the HTTP Methods?

HTTP Methods are also known as HTTP Verbs. They form a major portion of uniform interface restriction followed by the REST that specifies what action has to be followed to get the requested resource. Below are some examples of HTTP Methods:

GET: This is used for fetching details from the server and is basically a read-only operation.

POST: This method is used for the creation of new resources on the server.

PUT: This method is used to update the old/existing resource on the server or to replace the resource.

DELETE: This method is used to delete the resource on the server.

PATCH: This is used for modifying the resource on the server.

OPTIONS: This fetches the list of supported options of resources present on the server.

The POST, GET, PUT, DELETE corresponds to the create, read, update, delete operations which are most commonly called CRUD Operations.

GET, HEAD, OPTIONS are safe and idempotent methods whereas PUT and DELETE methods are only idempotent. POST and PATCH methods are neither safe nor idempotent.

What are the features of RESTful Web Services?

The service is based on the Client-Server model.

The service uses HTTP Protocol for fetching data/resources, query execution, or any other functions.

The medium of communication between the client and server is called “Messaging”.

Resources are accessible to the service by means of URIs.

It follows the statelessness concept where the client request and response are not dependent on others and thereby provides total assurance of getting the required data.

These services also use the concept of caching to minimize the server calls for the same type of repeated requests.

What is the concept of statelessness in REST?

As per REST architecture, a RESTful web service should not keep a client state on server. This restriction is called statelessness. It is responsibility of the client to pass its context to server and then server can store this context to process client's further request. For example, session maintained by server is identified by session identifier passed by the client.

How Do you share REST API contracts you created with Front end team?

There are multiple ways to share the created Rest API contract with front end or any other backend team who wants to access your service.

1) You can share the CURL from postman.

2) Best practice is to Implement the Swagger and configure it in such a way that as soon as you start your server the APIs must be automatically documented in Swagger UI

What is SWAGGER

Documentation From Your API Design

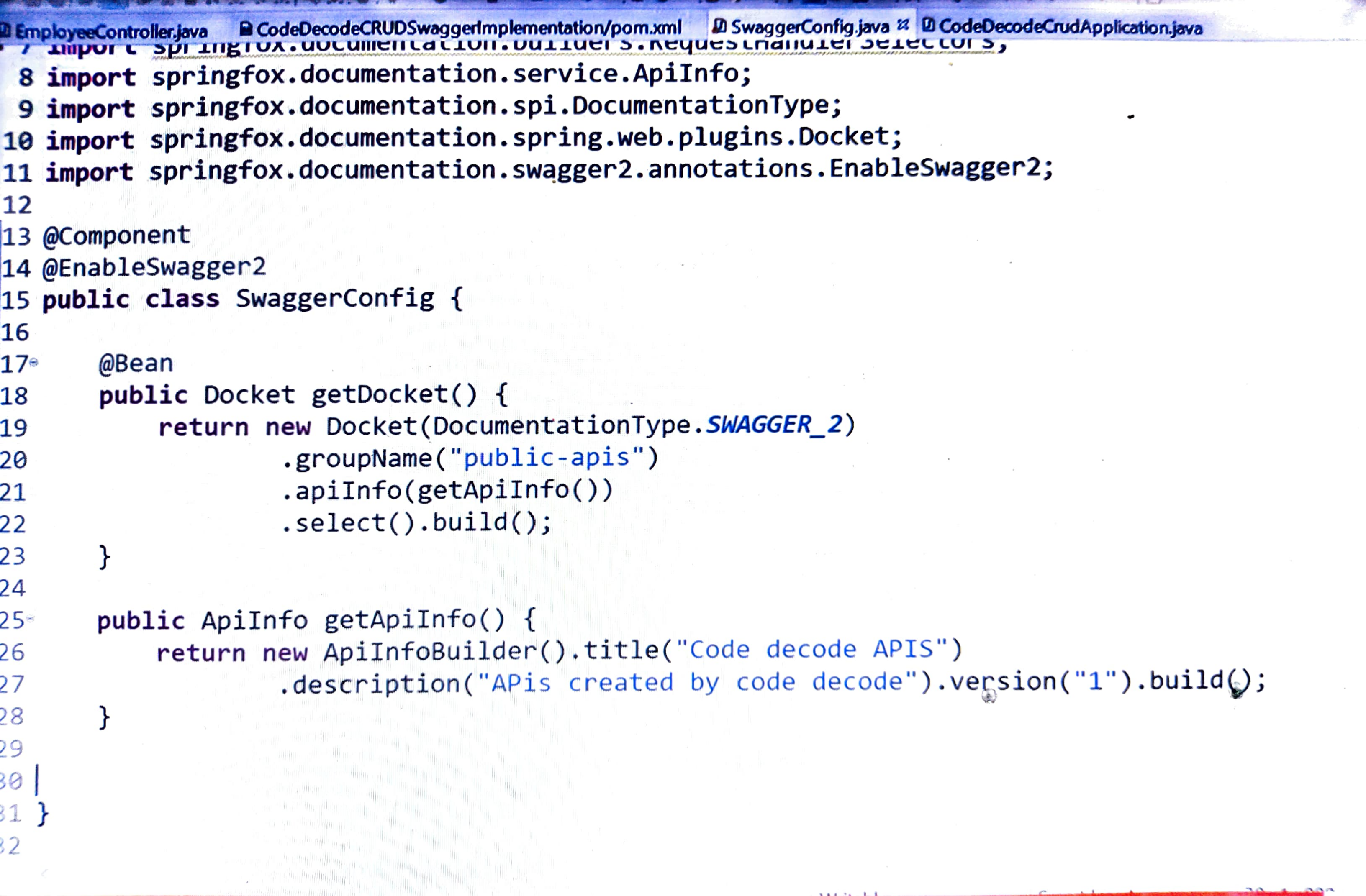
As your project grows with the time, The number of rest end points to fulfilment new functionalities also increases. And so does the headache of maintaining API docs

Swagger tools takes the hard work out of generating and maintaining your API docs, ensuring your documentation stays up-to-date as your API evolves.

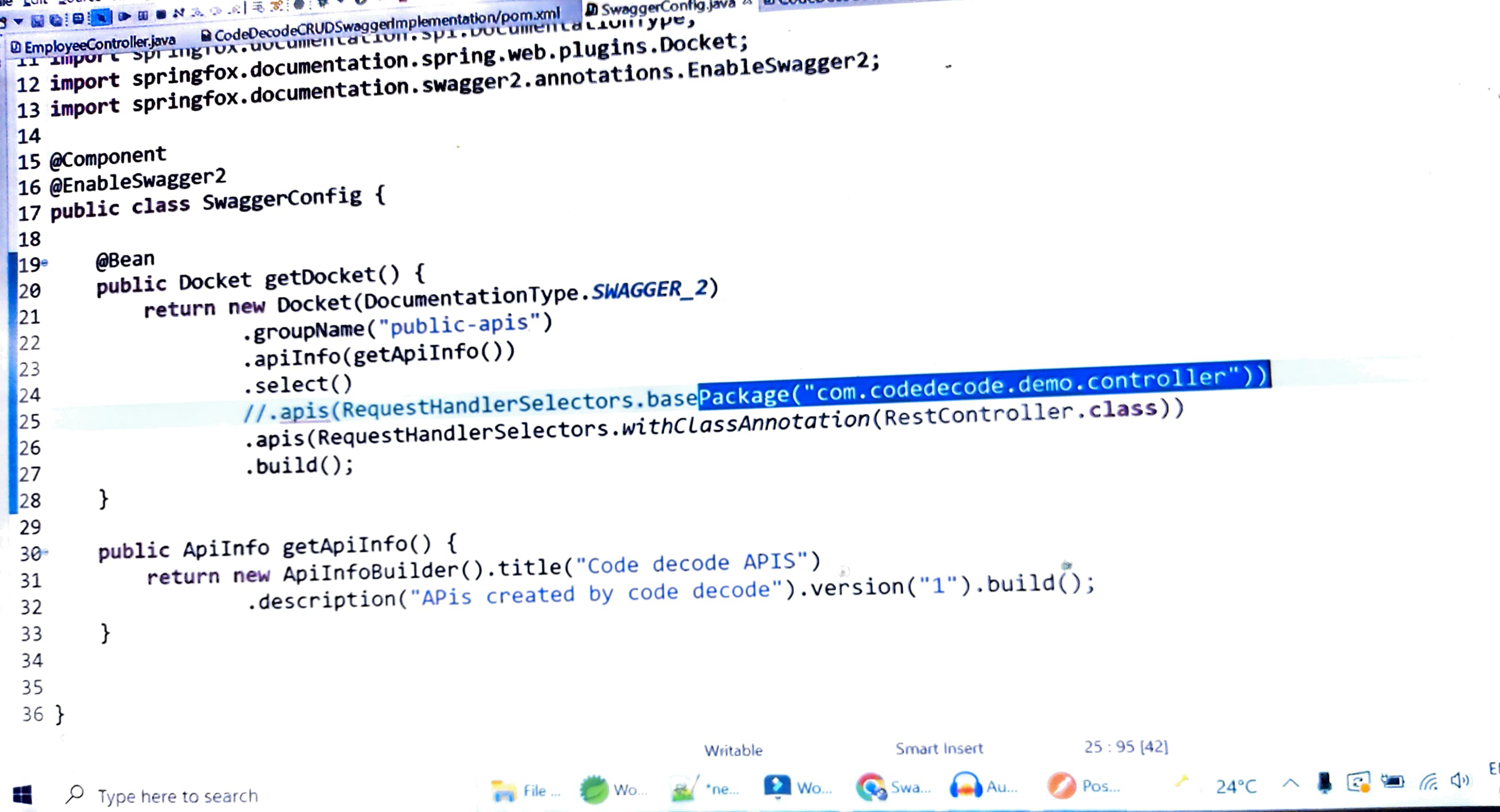
For existing APIs: Documentation can be auto-generated from an API definition.
For new APIs - on startup that too will be auto generated.

You can maintain multiple versions too.

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.4.0</version>
</dependency>

<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.4.0</version>
</dependency>
```


EmployeeController.java
CodeDecodeCRUDSwaggerImplementation/pom.xml
SwaggerConfig.java
CodeDecodeCrudApplication.java

```java
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Component
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket getDocket() {
        return new Docket(DocumentationType.SWAGGER_2)
                .groupName("public-apis")
                .apiInfo(getApiInfo())
                .select().build();
    }

    public ApiInfo getApiInfo() {
        return new ApiInfoBuilder().title("Code decode APIS")
                .description("APis created by code decode").version("1").build();
    }


}

```



```java
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Component
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket getDocket() {
        return new Docket(DocumentationType.SWAGGER_2)
                .groupName("public-apis")
                .apiInfo(getApiInfo())
                .select()
                //.apis(RequestHandlerSelectors.basePackage("com.codedecode.demo.controller"))
                .apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))
                .build();
    }

    public ApiInfo getApiInfo() {
        return new ApiInfoBuilder().title("Code decode APIS")
                .description("APis created by code decode").version("1").build();
    }


}
```

What is SWAGGER UI

Swagger UI allows

- Developers
- Front end team members
- Other Service users who calls your Rest endpoints for some data / functionality

to visualize and interact with the API's resources without having any of the implementation logic in place

Understanding Code

- @EnableSwagger2 - Enables Springfox swagger 2
- return new Docket(DocumentationType.SWAGGER_2) - Springfox's, primary api configuration mechanism is initialized for swagger specification 2.0
- groupName("public-api") - Docket helps configure a subset of the services to be documented and groups them by name. Like - public-api , private-apis, business-api, admin-specific-apis
- apiInfo(apiInfo()) - return ApiInfoBuilder - Builds the api information - like title, description, version, license etc.

Understanding Code

- select() - select() returns an instance of ApiSelectorBuilder to give fine grained control over the endpoints exposed via swagger.
- apis() allows selection of RequestHandler's using a predicate. The example here uses an any predicate (default). Out of the box predicates provided are any, none, withClassAnnotation, withMethodAnnotation and basePackage.
-