**Description:**

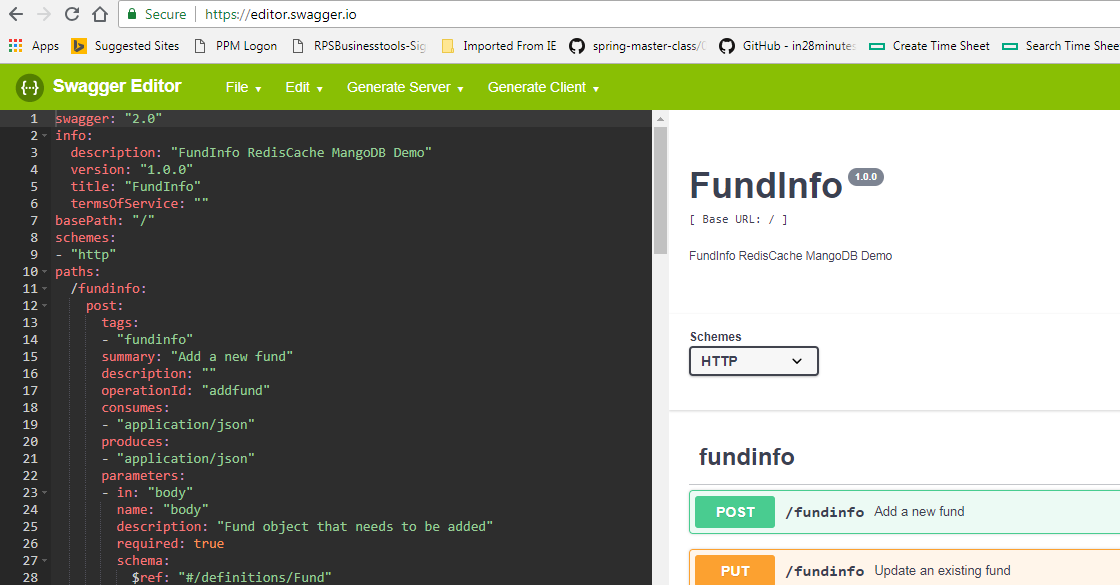
This POC is to create a Spring Boot application to cache the data for CURD Endpoints to Redis-Cache and to update the MYSQL database using JPA Repository. Also integrate the functional testing with “Cucumber-Rest Assured” testing frame work.

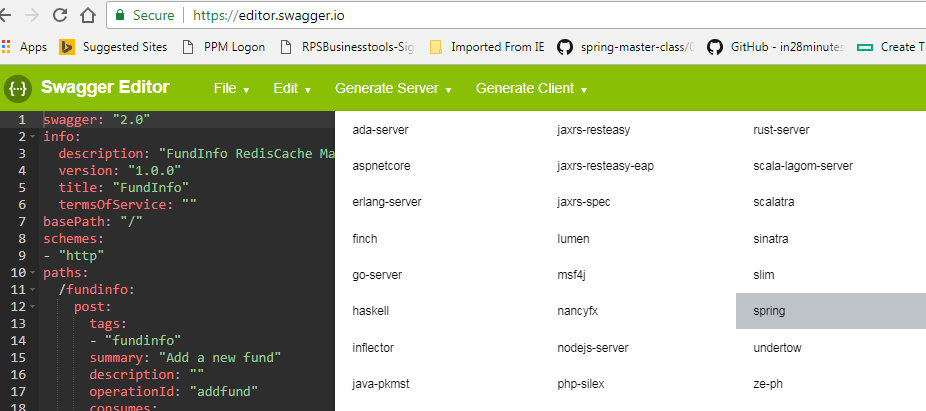
Git link **:** https://git.platform.manulife.io/jhrps/pcf-rediscache-poc

**STEPS:**

1. **SWAGGER :**

We used “Contract First” approach to create this REST service. We defined the API-EndPoints & DB Model in <https://editor.swagger.io/> [Please refer [fund-Swagger.yml](https://git.platform.manulife.io/jhrps/pcf-rediscache-poc/blob/master/fund-Swagger.yml) ] and auto generated Spring Code.





1. **Maven Dependency**:

Import the Auto-Generated Code as Maven Project. In Pom.xml, add or update the following dependencies.

**Spring-Boot**

<parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.0.5.RELEASE</version>  
</parent>

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
</dependency>

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
</dependency>

**Redis-Cache**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-redis</artifactId>  
 <version>1.4.7.RELEASE</version>  
</dependency>

**MY-SQL**

<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
</dependency>

**Spring JPA**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>

**H2 In-Memory Database**

<dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
</dependency>

**Junit [Testing Framework]**

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <scope>test</scope>  
</dependency>

**Rest-Assured [Testing Framework]**

<dependency>  
 <groupId>com.jayway.restassured</groupId>  
 <artifactId>rest-assured</artifactId>  
 <version>${rest-assured.version}</version>  
 <scope>test</scope>  
</dependency>

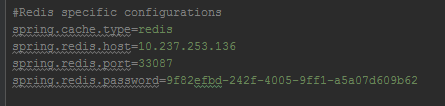
<!-- Json path library -->  
<dependency>  
 <groupId>com.jayway.jsonpath</groupId>  
 <artifactId>json-path</artifactId>  
 <version>${jsonpath-version}</version>  
</dependency>

**Cucumber [Testing Framework]**

<!-- Cucumber java implementation -->  
<dependency>  
 <groupId>info.cukes</groupId>  
 <artifactId>cucumber-spring</artifactId>  
 <version>${cucumber-version}</version>  
 <scope>test</scope>  
</dependency>  
  
<dependency>  
 <groupId>info.cukes</groupId>  
 <artifactId>cucumber-junit</artifactId>  
 <version>${cucumber-version}</version>  
 <scope>test</scope>  
</dependency>  
<dependency>  
 <groupId>info.cukes</groupId>  
 <artifactId>cucumber-java</artifactId>  
 <version>${cucumber-version}</version>  
 <scope>test</scope>  
</dependency>

1. **RedisCache Service:**

From PCF market place, search for Redis Service and create “Shared-VM” Instance for development. For unit testing configure the Redis parameters in Application properties file. While Pushing the application to PCF bind the Spring application with Redis Service Instance, so that the Redis Parameter will be dynamically passed through VCAP.



1. **Caching - Spring Implementation**

Implement Caching logic in Service Layer. Following are some important spring cache annotations,

@**Cacheable** - The simplest way to enable caching behavior for a method is to demarcate it with @Cacheable and parameterize it with the name of the cache where the results would be stored:

@**CacheConfig** - With this annotation, you can streamline some of the cache configuration into a single place – at the class level – so that you don’t have to declare things multiple times:

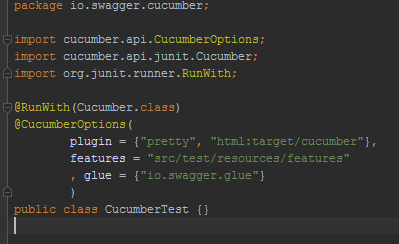
@**CacheEvict** – this annotation is used to indicate the removal of one or more/all values – so that fresh values can be loaded into the cache again:

@**CachePut** - With this annotation, you can update the content of the cache without interfering the method execution. That is, the method would always be executed and the result cached.

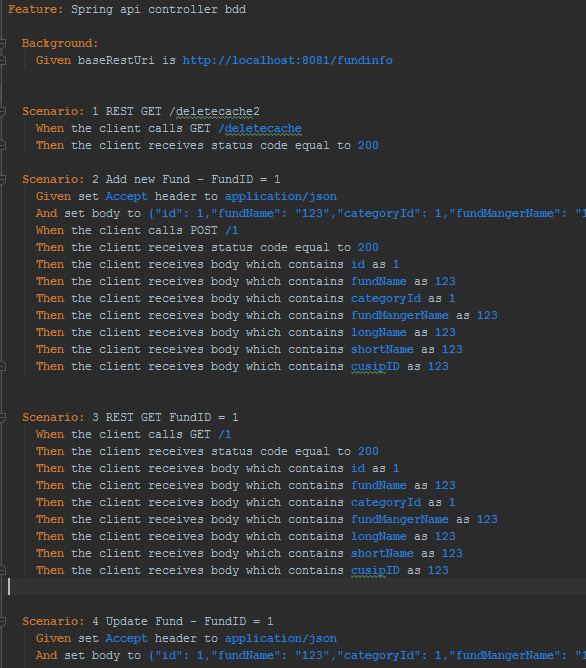
1. **Cucumber- Rest Assured Functional Testing:**

Use “**MVN Test**” Command to run the Cucumber test

1. Define cucumber test class.



1. Define the Feature file with Rest API Test secnerios



1. Define the cucumber & Rest Assured Java Implementation in GLUE Java Package,

