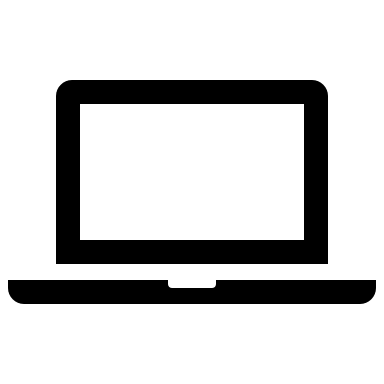
Investment Recommendation REST API

# Objective

To develop an application **“investment-recommendation-api”** that exposes a REST API, where the API would give Relationship Manager (RM) information about recommended investment products for a given RM’s client with specific risk criteria.

Client info with specific risk criteria



Recommended investment products

**RM**

**REST API**

# Assumptions

For this application to be able to serve its users (relationship managers) and advise on recommended investment products, this application must have knowledge/information about the following:

* Client and the risk score indicating the risk they are willing to take.
* Investment product and the risk information advised by the investment committee.

In this application design, the above information is stored in the database in the following 2 tables.

|  |
| --- |
| **CLIENT\_PORTFOLIO** |
| ID INT PRIMARY KEY |
| CLIENT\_NAME VARCHAR(100) |
| RISK\_SCORE INT |

|  |
| --- |
| **INVESTMENT\_RECOMMENDATION** |
| ID INT PRIMARY KEY |
| INVESTMENT\_NAME VARCHAR(100) |
| RISK\_SCORE INT |
| INVESTMENT\_RETURN\_SCORE INT |

# Project and Application Structure

The newly created project is a maven java project. The project information is available in the pom.xml file.

## Dependencies / Libraries / Frameworks

* JDK 8
* Maven
* Spring Boot (web and jdbc)
* H2 database
* commons-lang3
* log4j

## Spring Boot

This application is a Spring Boot application. It means that the REST API is available online as soon as the application is started up. There is no need for deployment in any container (e.g. Tomcat). Please see Spring Boot documentation for more information.

## H2 Database

As per the Assumptions section above, H2 database is used for this purpose. The database configuration can be found in the **src/main/resources/application.properties** file.

Two SQL files below are provided in **src/main/resources** and they will be executed every time the application is being started up:

* schema-h2.sql (SQL file for creating tables)
* data-h2.sql (SQL file for inserting sample data)

For development/debugging convenience purpose, H2 provides web console that allows developers to browse the database. This setting (spring.h2.console.enabled=true) is configurable in application.properties. The console can then be accessed from http://localhost:8080/h2-console.

Lastly, it is also important to note that H2 database can easily be replaced with any other kind of database (like Oracle, etc) for production environment. The configuration file application.properties can be reconfigured by splitting it into application-dev.properties, application-prod.properties, etc. The database configuration in prod would then be configured differently from dev in the individual files.

## Available REST API

Following the requirement, the REST API would be available at the following URL as soon as the application is started:

* **http://host:8080/api/client/{id}**
  + host would be the host of the machine where the application is started. E.g. localhost
  + 8080 is the port number defined in application.properties
  + {id} would be the client id (primary key)

Additional API is also available to list all clients in the database:

* **http://host:8080/api/clients**
  + host would be the host of the machine where the application is started. E.g. localhost
  + 8080 is the port number defined in application.properties
  + {id} would be the client id (primary key)

## User Interface

A very simple and plain user interface to get data from api/clients is also available at the following URL:

* **http://host:8080/**

## Unit Tests and Integration Test

The following Unit and Integration tests are written. They will be executed as part of the test phase in maven build.

|  |  |
| --- | --- |
| **Class Name** | **Type** |
| DataAccessServiceTest | Unit Test |
| WebControllerTest | Unit Test |
| InvestmentRecommendationApiApplicationTests | Integration Test |

# How to Build and Run the Application

To build this application, please navigate to the root directory of this project and then run:

**mvn clean package**

Maven will automatically download the required libraries from Maven repository for this purpose. Upon successful build, you would find the following jar file in the target directory:

* investment-recommendation-api-1.0.0-SNAPSHOT.jar

To run the application, please navigate to the target directory and then run the following:

**java -jar investment-recommendation-api-1.0.0-SNAPSHOT.jar**

The REST API application is now online, and the following API are available for access:

* **http://localhost:8080/api/client/{id}**
* **http://localhost:8080/api/clients**

# Code Coverage Test

Code coverage test plugin cobertura is configured in Maven configuration file pom.xml.

To generate code coverage test report, please navigate to the root directory of this project and then run:

**mvn cobertura:cobertura**

The generated report can then be found in **target/site/cobertura/index.html** file.