PCS Version : 0.0.1
Document version : 1.0
Author: Jalal Sordo.
Date : 22-05-2019

Postal Code Service: Technical Documentation

Contents

Scope of the document 1
Project Description 1
Development Environment Setup 2
1/ Java JDK 8 (**64bit**) 2
2/ Eclipse STS (**64bit**) 2
3/ Import project and Git repository 2
5/ Running the project 3
Logging 3
Database 3
Tables 3
Project Structure and technology 3
Framework 3
Project Structure: 4
Containerization: 5
Docker 5
Cloud Deployment 5
Azure 5
“What else could have been done?” 5

Scope of the document

This document aims to provide instructions and guidelines on the technical aspects of **Postal Code Service** project.

Project Description

Postal Code Service is a web service that allows clients to do the following operations (Up to the moment of writing this document):

- Updates a postal code's coordinates of a location.
- Deletes a postal code along with their latitude and longitude.
- Returns the geographic (straight line) distance between two postal codes in the UK.

Development Environment Setup

In order to set up the project you will need to follow the steps below:

1/ Java JDK 8 (**64bit**)

Download and install Java JDK 8 (**64bit**) from the link
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

2/ Eclipse STS (**64bit**)

1. Download and extract Eclipse STS to `c:/sts` from http://download.springsource.com/release/STS/3.9.5.RELEASE/dist/e4.8/spring-tool-suite-3.9.5.RELEASE-e4.8.0-win32-x86_64.zip
2. Create a folder in `c:/postal-code-service`
3. Run Eclipse STS.
4. Use `c:/workspace` as default workspace.

3/ Import project and Git repository

The project source code is hosted on **GitHub** under the URL below:

```
https://github.com/JalalSordo/postal-code-service.git
```

1. In eclipse STS click **File> Import…**
2. In the new Window Select **Git> Projects from Git** and click **Next.**
3. Select **Clone URI** and click **Next.**
4. Paste the Git repository URL above in **URI Field**.
5. In the **Authentication** section put your **GitHub** Login and Password and click **Next**.
6. Select **master** branch or **another** branch if you are interested in, click **Next**.
7. In the Destination/Directory use `c:/postal-code-service`
8. Keep **remote name** as **origin** and click **Next**.
9. Select Import as **general project** click **Next**.
10. In this last window you can define a **name** for the project (If you are targeting a specific branch it is recommended to append the branch name to the **name of the project** e.g.: *postal-code-service-branchName*), click **Next**.
11. Now in the Eclipse Projects tree (far left) right click on the newly imported *postal-code-service* project.
12. Select **Configure**> C**onvert to Maven project**

5/ Running the project

In order to create a run configuration please follow the steps below:

1. Start the web server application (*Run As…* > *Spring Boot App*).
2. Open your Web Browser and go to http://localhost:8080/swagger-ui.html. Now you should see the Swagger API Documentation of Postal Code service RESTful API.

Logging

The application will log automatically to `<user.home>` folder under `postal-code-service\logs,` the actual trigger for the **Rolling Policy** is 10MB.

It can be changed in `src/main/reources/logback.xml`

```
<triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
      <MaxFileSize>10MB</MaxFileSize>
</triggeringPolicy>
```

Database

The project uses H2 in memory database, you may access `<Host>/h2-console` to check the Postal Code Service database, use the credentials below to connect to the database.

- **JDBC URL** : `jdbc:h2:mem:postalcodes;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE`
- **User Name** : sa
- **Password** : postalcodeservice

Tables

API_REQUEST (*ID, TIMESTAMP, FROM_POSTAL_CODE, TO_POSTAL_CODE*)
Contains the log of all request sent to `/api/postal-codes/distance`
LOCATION (ID*, LATITUDE, LONGITUDE, POSTAL_CODE*)
Contains all the locations in the UK (limited to 1000 Rows for Brevity and keeping the project as light-weight as possible.)

Project Structure and technology

Framework

Spring boot is a module of spring framework which is used to create stand-alone, production-grade Spring based Applications with minimum programmer's efforts. It is developed on top of core spring framework. The main concept behind spring boot is to avoid lot of boilerplate code and configuration to improve development, unit test etc. As in case of creating a new spring application, we have write many XML configurations, server setting, adding dependencies etc. These configuration files are one of the example of boilerplate code. Spring boot avoids all these boilerplate codes.

PCS Version : 0.0.1
Document version : 1.0
Author: Jalal Sordo.
Date : 22-05-2019

Project Structure:

```
Postal Code Service/
|   Dockerfile
|   Documentation.docx
|   pom.xml
|   README.md
+---src
|   +---main
|   |   +---java
|   |   |   \---com
|   |   |       \---worth
|   |   |           \---postalcodeservice
|   |   |                   Application.java
|   |   |
|   |   |                   +---configuration
|   |   |                   |       HttpRequestInterceptor.java
|   |   |                   |       HttpRequestInterceptorConfig.java
|   |   |                   |       SwaggerConfig.java
|   |   |                   +---controller
|   |   |                   |       PostalCodesController.java
|   |   |                   +---dto
|   |   |                   |       LocationDTO.java
|   |   |                   |       Result.java
|   |   |                   +---exceptions
|   |   |                   |       LocationNotFoundException.java
|   |   |                   +---model
|   |   |                   |       ApiRequest.java
|   |   |                   |       Location.java
|   |   |                   +---repositories
|   |   |                   |       ApiRequestRepository.java
|   |   |                   |       LocationRepository.java
|   |   |                   +---service
|   |   |                   |   |   ApiRequestsService.java
|   |   |                   |   |   PostalCodesService.java
|   |   |                   |   |
|   |   |                   |   \---impl
|   |   |                   |           ApiRequestsServiceImpl.java
|   |   |                   |           PostalCodesServiceImpl.java
|   |   |                   \---utils
|   |   |                       |   GeoUtils.java
|   |   |                       |
|   |   |                       \---constants
|   |   |                               GeoConstants.java
|   |   \---resources
|   |       |   application.properties
|   |       |   data.sql
|   |       |   logback.xml
|   |       +---static
|   |       \---templates
|   \---test
|       +---java
|       |   \---com
|       |       \---worth
|       |           \---postalcodeservice
|       |                   ApplicationTests.java
|       |                   +---controller
|       |                   |       PostalCodesControllerIntegrationTest.java
|       |                   \---service
|       |                           PostalCodesServiceIntegrationTest.java
|       |
\---\---resources
            logback-test.xml
```

PCS Version : 0.0.1
Document version : 1.0
Author: Jalal Sordo.
Date : 22-05-2019

Containerization:

Docker

A docker image this project is available in docker hub and can be pulled with the command below:

```
docker pull javco/postal-code-service
```

Cloud Deployment

Azure

The above mentioned docker image is deployed in Microsoft azure, you access it via the URL below:

https://postal-code-service.azurewebsites.net/swagger-ui.html

“What else could have been done?”

- More elaborated unit tests by mockito for every @Service and @Repositry and @RestController.
- Basic angular front end that send requests to the back end.
- New API endpoint to generate reports of postal codes with highest hits.
- Securing the API by using Simple JWT token or full oAuth implementation.
- Spring batch to read from the csv and insert into a fully-fledged the database (Mysql).
- Regression campaign : Analysis and Automated Tests with Postman
- Sonar/Code coverage stats.
- Continuous Integration Pipelines in Jenkins.