User Guide

Java Backend Developer Technical Test

Bill Huang

Version : V1.0

Date : Dec 12, 2018

Copyright© Aequilibrium.

All Rights Reserved.

Content

Installation The Project

Install JDK and IDE

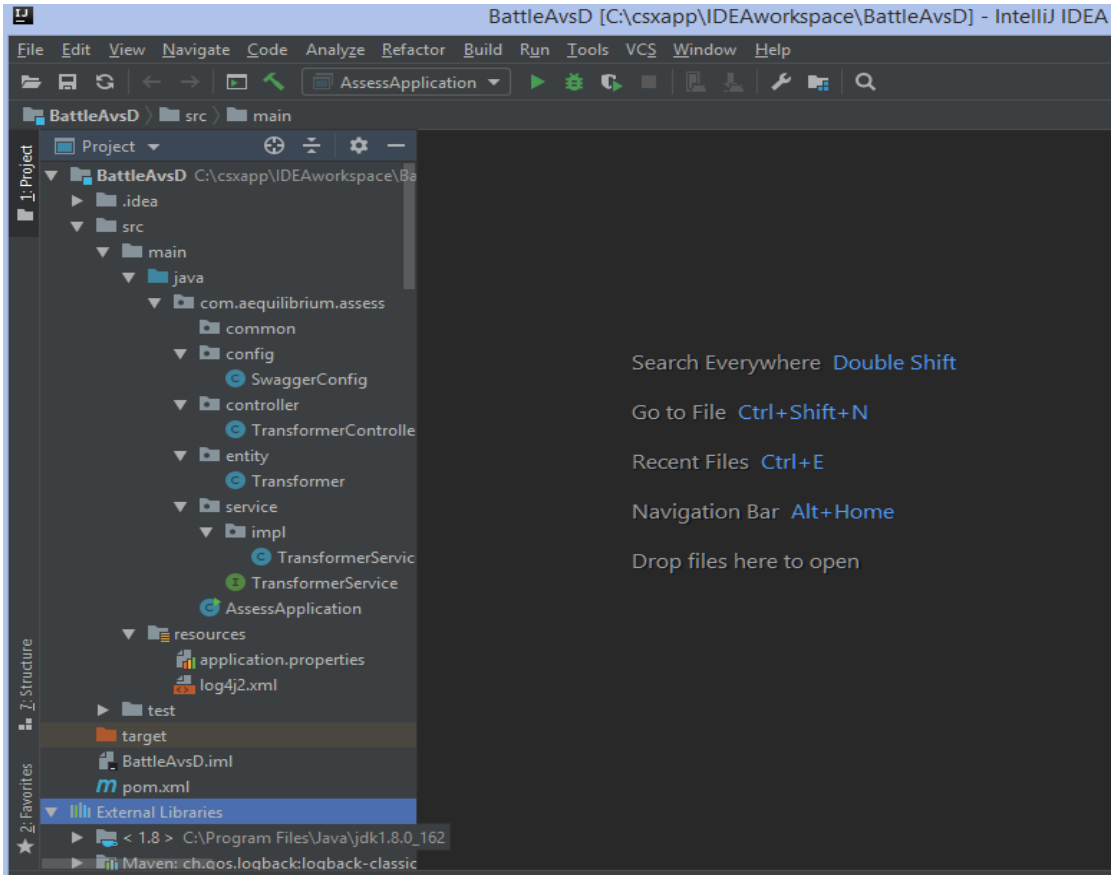
I use the IDEA as development tools. Before start IDEA, make sure the Java JDK is already installed.

The following is the steps how to operate it:

1) The test project was zipped with 7-zip, it need be unzipped before it is loaded to IDEA;

2) Check files structure, then locate to the folder that test project was unzipped there;

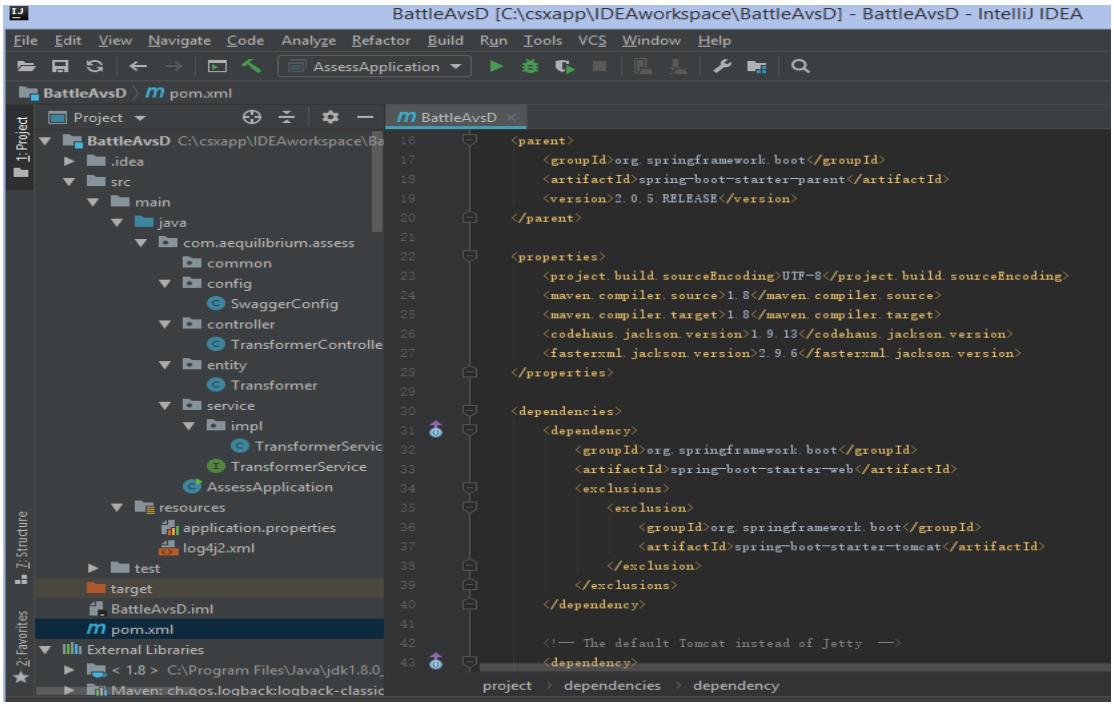
3) After the BattleAvsD (the IDEA project name of the test project) is opened, right click the project name BattleAvsD, select Maven, and select Download Sources. It looks like this.



4) If there is not messages displayed, it can be gone to the next step.

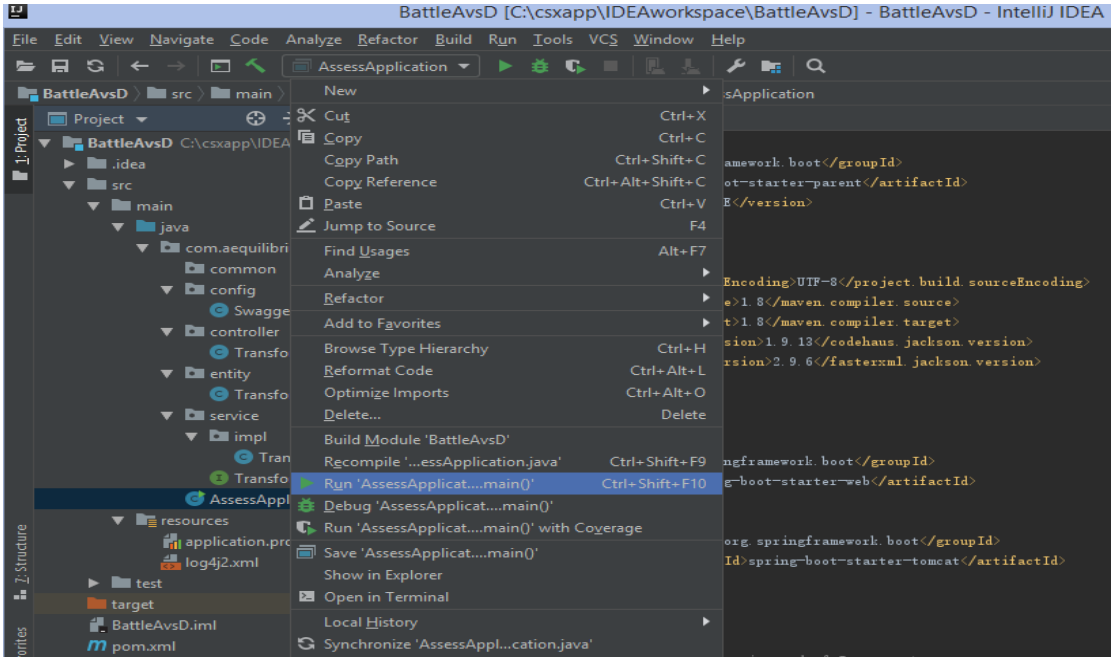
Run test project BattleAvsD from IDEA

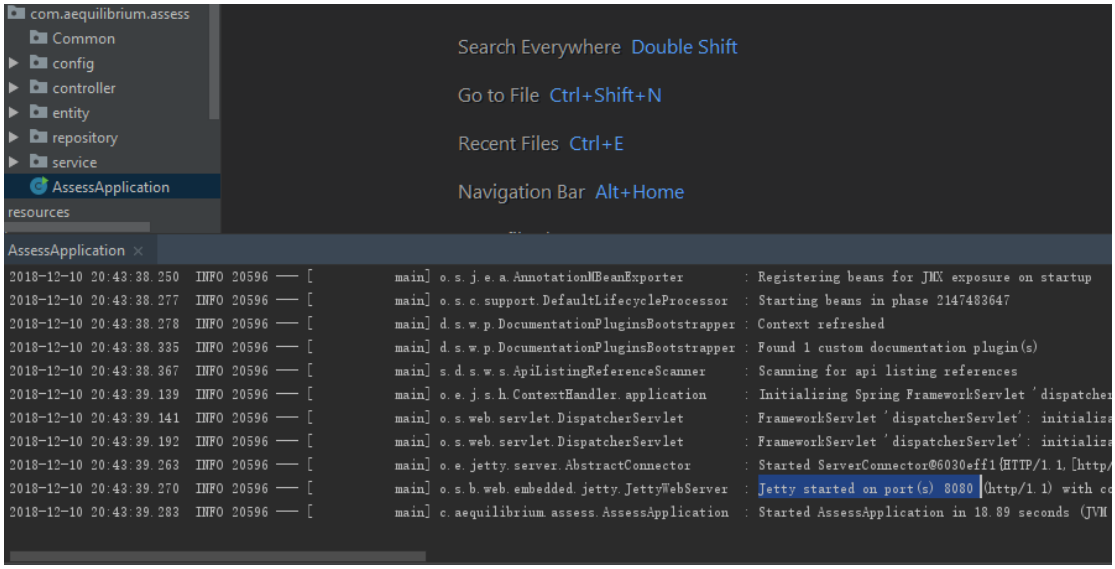
Make sure all dependencies have been downloaded.



Right click AssessApplication under package com.aequilibrium.assess, select Run

'AssessApplication' or press Ctrl+shift+F10. Then it can be seen running from window Console.



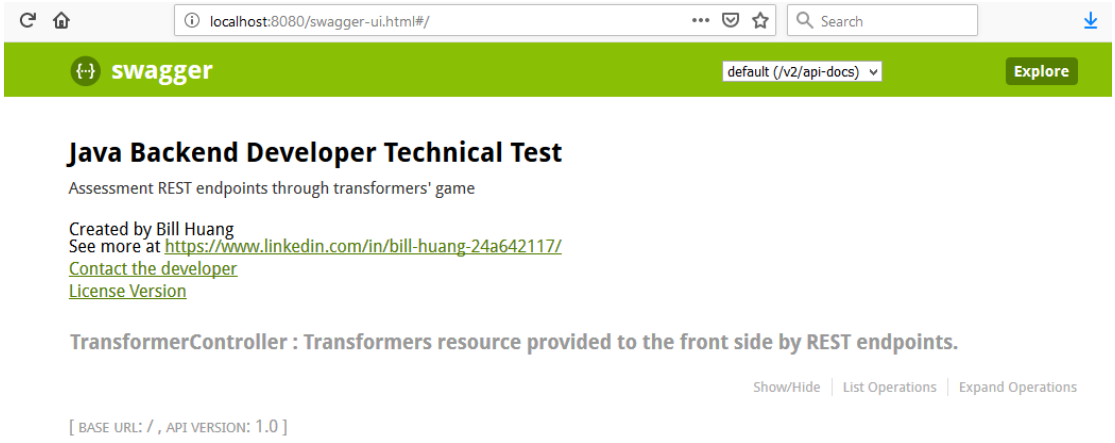


If Jetty started on port(s) 8080 is shown in above window. That means the project BattlesAvsD started correctly. The result can be checked from explorer.

Check the result from Explorer

Start Explorer by Manually

It can be used any explorers, e.g. IE, Firefox, Google Chrome because it has not any html, JavaScript, CSS etc. It is used only to check the REST provided result to client side.

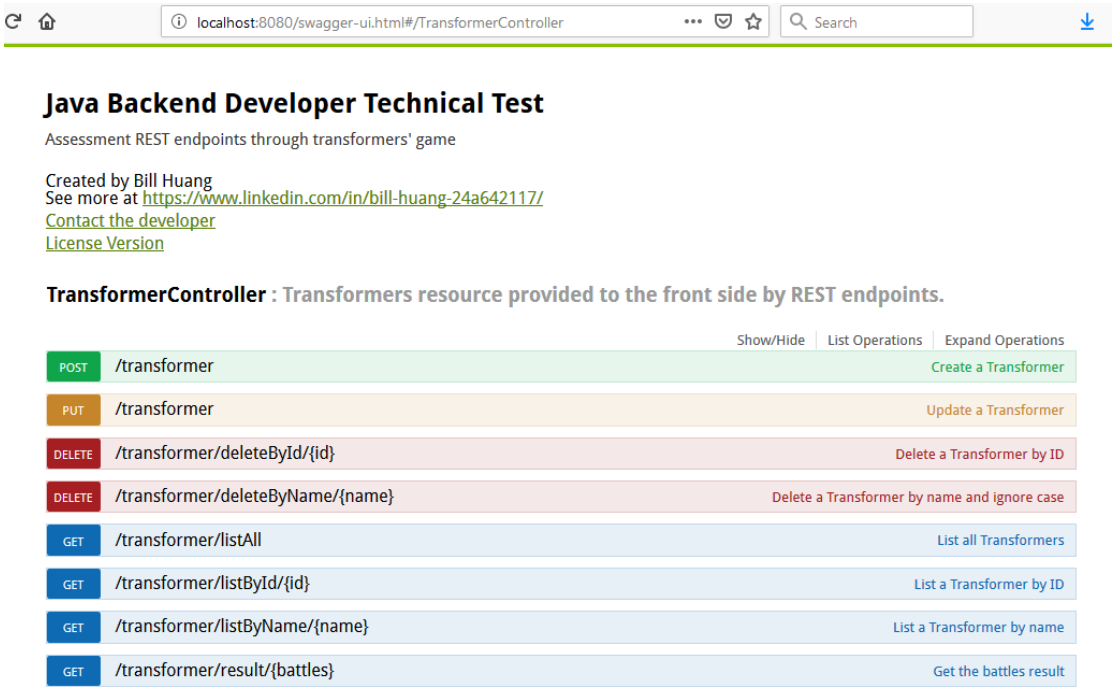


Input the URL address: http://localhost:8080/swagger-ui.html, the message above can be seen.

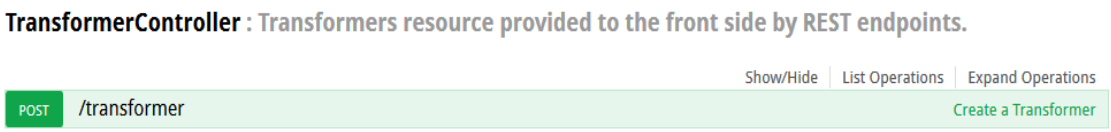
Check the REST endpoints

According to the Java Backend Developer Technical Test requirement, the API should allow the following main functionality:

- ✧ Create a Transformer;
- ✧ Update a Transformer;
- ✧ Delete a Transformer;
- ✧ List Transformers;
- ✧ Given a list of Transformer IDs, determine the winning team.

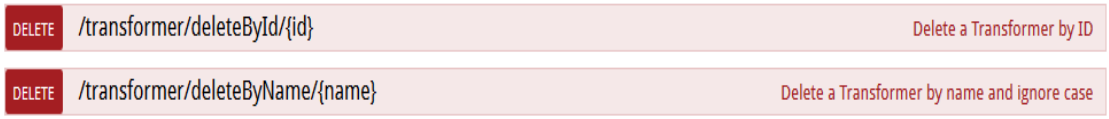


The list of all endpoints is above.

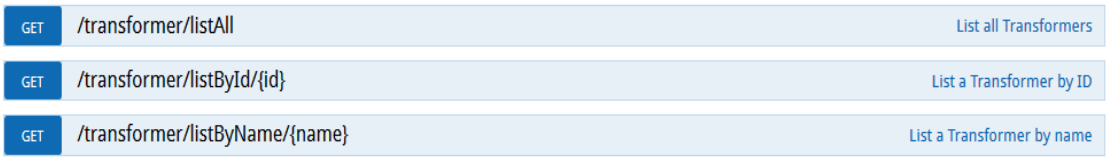


Create a new transformer, click /transformer to check it.

Update a transformer, click /transformer to check it.



Delete a transformer, click /transformer/deleteByID/{id} or /transformer/deleteByName/{name} to delete it by ID or name.



Query a or all by click the above addresses, it is filtered by ID, name or all.

Click /transformer/result/{battles}, and input the battles, for example 1, it will be shown the default result. The project already set the default three transformers as follows:

Soundwave, D, 8, 9, 2, 6, 7, 5, 6, 10

Bluestreak, A, 6, 6, 7, 9, 5, 2, 9, 7

Hubcap, A, 4, 4, 4, 4, 4, 4, 4, 4

You can use these data to test the battle result first, then delete transformers, add a new one or more, try run it under different cases.

Check the transformer definition

According to the Java Backend Developer Technical Test requirement, each transformer has the following criteria (ranked from 1 to 10) on their tech spec:

- Strength
- Intelligence
- Speed
- Endurance
- Rank
- Courage
- Firepower
- Skill

It can be checked from any REST endpoint, e.g. /transformer, create a transformer as follows:

POST /transformer

Response Class (Status 200)
OK

Model | Example Value


```
{
  "courage": 0,
  "endurance": 0,
  "firepower": 0,
  "flag": "string",
  "id": 0,
  "intelligence": 0,
  "name": "string",
  "overallRating": 0,
  "rank": 0,
```


POST /transformer

Response Class (Status 200)
OK

Model | Example Value

Transformer {
courage (integer, *optional*): Ranked from 1 to 10,
endurance (integer, *optional*): Ranked from 1 to 10,
firepower (integer, *optional*): Ranked from 1 to 10,
flag (string, *optional*): It is work field. To flag status in the game,
id (integer, *optional*): ID is created automatically, it is unique,
intelligence (integer, *optional*): Ranked from 1 to 10,
name (string, *optional*): The transformer's name is not blank,
overallRating (integer, *optional*): It is work field. To hold the overall rating,
rank (integer, *optional*): Ranked from 1 to 10,
skill (integer, *optional*): Ranked from 1 to 10,
speed (integer, *optional*): Ranked from 1 to 10,
strength (integer, *optional*): Ranked from 1 to 10,
type (string, *optional*): Fill type as A/D
}

When it was checked, the format is the same with Java Backend Developer Technical Test requirement.

Using Swagger Notes

As the following shown, I use swagger version 2.6.1, the project can be used the newest verion2.9.2, but the interference has a little differences.

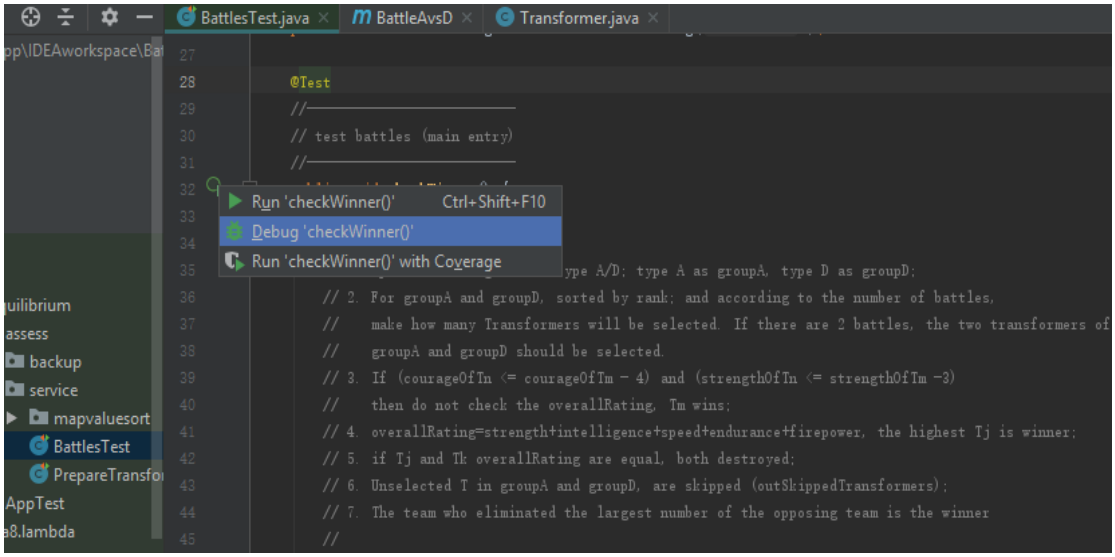
```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui 2.9.2-->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.6.1</version>
    <scope>compile</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 2.9.2-->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.6.1</version>
    <scope>compile</scope>
</dependency>
```

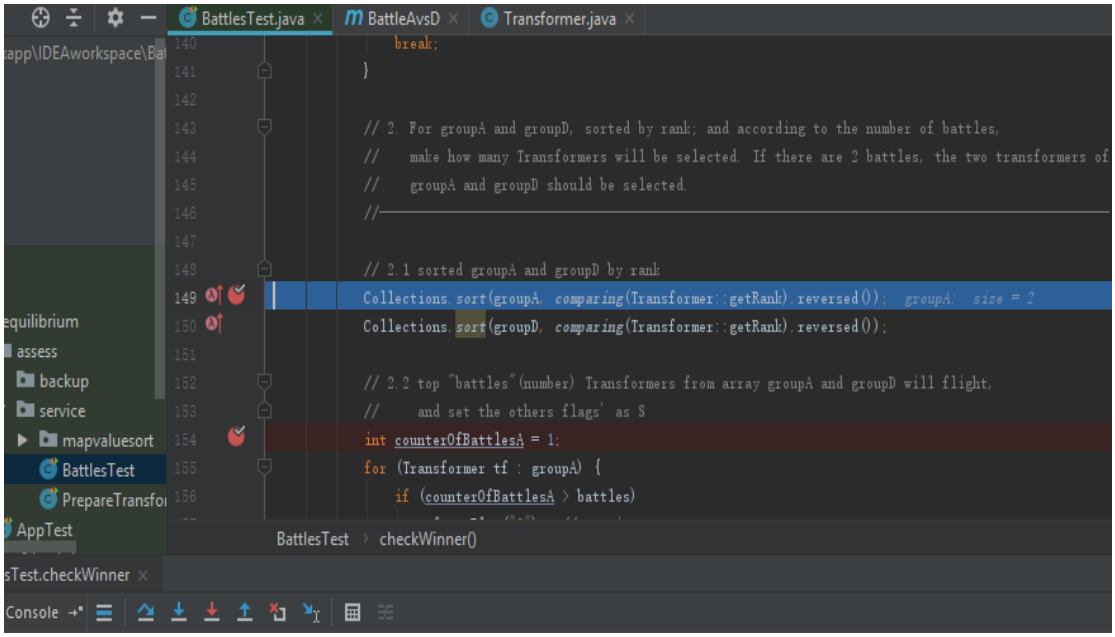
I also give it a scope limited because we need only check the API endpoint result to help front side development. It should not bring to the product environment.

Unit Test Notes

As the following figure, if I want to unit test BattlesTest' s function checkWinner(), it can be done as follows:



Click the small green circle, then select Debug 'checkWinner()' .



Run the test step by step, and check the changes of the variables.

It is done here. Enjoyed it!

Thanks!