**Project Title**

Top Score Ranking

## Getting Started

## This instruction will give some details insight of how to install, run and test the Top Score Ranking sample project. Both Unit tests and Integration tests have been executed on Postman and curl. Please have a look deployment note to know how to import and run the project and conduct test cases locally.

Tools and Technologies

1. Java 8 (1.8.0\_211)
2. Gradle 6 (6.7.1)
3. IntelliJ ultimate edition 2020.3
4. Spring boot 2.3.2
5. MySQL DB.
6. JUnit 5
7. Postman (v.7.36.1)
8. MacOS (Big Sur)

Deployment

1. Download / Clone “Score Project” from GitHub.
2. Run IntelliJ and “open” project from IntelliJ
3. Right click on “build.gradle” and click on “build”. If any problem, please close and open the project to ensure build is performed properly.
4. Create a MySQL DB “playscore” and create a table “score”. Table columns details are described on “DB Table” section.
5. Right click on “GameApplication.java” class under “src -> main -> java -> com -> sample -> play” packages and click on “run”. Application will be started on local tomcat with port 8080.
6. To run Unit test, please right click on “ScoreServiceTests” class under “test -> java -> com -> sample -> play” packages and click on “run”. Also, individual test case can be run inside the “ScoreServiceTests” file.
7. To run Controller Integration test, please right click on “ScoreControllerIntegrationTest” class under “test -> java -> com -> sample -> play” packages and click on “run”.

DB Table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| DB Name | playscore | | | | | | |
| Table Name | score | | | | | | |
| Column Name | Data Type | Size | Null | Default | Auto Increment | Primary key | Comments |
| score\_id | Int | 11 | Checkbox Crossed with solid fill |  |  |  |  |
| score | int |  | Checkbox Crossed with solid fill | 0 |  |  |  |
| name | Varchar | 100 |  |  |  |  |  |
| created\_at | datetime |  | Checkbox Crossed with solid fill | CURRENT\_TIMESTAMP |  |  |  |
| delete\_flg | tinyInt | 1 | Checkbox Crossed with solid fill | 0 |  |  |  |

“application.property” files contains the configuration information for DB setup. Please change password if needed. Property file resides inside resource package.

API Documentation

1. GET Score by ID

|  |  |
| --- | --- |
| **METHOD** | POST |
| **URL** | http://localhost:8080/play/score/ |
| **Comment** | |
| Requirement document describes that request and response should be delivered on payload. So, method is POST and request id will be given in payload. | |

**Request Body**

|  |  |
| --- | --- |
| **Object** | ScoreReqParam |
| **Physical Item** | **Data Type** |
| Id | Long |

**Response**

|  |  |
| --- | --- |
| **Object** | ScoreReponseDto |
| **Physical Object Item** | **Data Type** |
| scoreId | Long |
| name | String |
| score | int |

|  |
| --- |
| {  "code": 200,  "errCode": **null**,  "messages": [  "OK"  ],  "errItems": **null**,  "result": {  "scoreId": 1,  "name": "Rashed",  "score": 500  }  } |

**Response Json**

1. Create Score

|  |  |
| --- | --- |
| **METHOD** | POST |
| **URL** | http://localhost:8080/play/score/create |

**Request Body**

|  |  |
| --- | --- |
| **Object** | CreateReqParam |
| **Physical Object Item** | **Data Type** |
| player | String |
| score | int |
| createdAt | String |

|  |
| --- |
| {  "code": 200,  "errCode": **null**,  "messages": [  "OK"  ],  "errItems": **null**,  "result": **null**  } |

**Response Json**

1. Delete Score

|  |  |
| --- | --- |
| **METHOD** | DELETE |
| **URL** | http://localhost:8080/play/score/delete |

**Request Body**

|  |  |
| --- | --- |
| **Object** | DeleteParam |
| **Physical Object Item** | **Data Type** |
| id | Long |

|  |
| --- |
| {  "code": 200,  "errCode": **null**,  "messages": [  "OK"  ],  "errItems": **null**,  "result": **null**  } |

**Response Json**

1. GET Score List

|  |  |
| --- | --- |
| **METHOD** | POST |
| **URL** | http://localhost:8080/play/score/list?pageNum=1&pageSize=10 |

**Request Body**

|  |  |  |
| --- | --- | --- |
| **Object** | ScoreListReqParam | |
| **Physical Object Item Name** | **Data Type** | **Comments** |
| player | List<String> |  |
| beforeTime | String | Date format : yyyy-MM-dd |
| afterTime | String | Date format : yyyy-MM-dd |

Response

|  |  |  |
| --- | --- | --- |
| **Object** | ScoreResponseDto | |
| **Physical Object Item Name** | **Data Type** | **Comments** |
| scoreId | Long |  |
| name | String |  |
| score | Integer |  |

**Response Json**

|  |
| --- |
| {  "code": 200,  "errCode": **null**,  "messages": [  "OK"  ],  "errItems": **null**,  "result": {  "pageNum": 1,  "pageSize": 10,  "totalPage": 1,  "total": 6,  "list": [  {  "scoreId": 1,  "name": "Rashed",  "score": 500  },  {  "scoreId": 3,  "name": "Rashed",  "score": 300  },  {  "scoreId": 5,  "name": "Rashed",  "score": 270  }  ] } } |

1. GET score history

|  |  |
| --- | --- |
| **METHOD** | POST |
| **URL** | http://localhost:8080/play/score/history |

**Request Body**

|  |  |  |
| --- | --- | --- |
| **Object** | NameReqParam | |
| **Physical Object Item Name** | **Data Type** | **Comments** |
| player | String | Not Null |

Response

|  |  |  |
| --- | --- | --- |
| **Object** | PlayerHistoryResponseDto | |
| **Physical Object Item Name** | **Data Type** | **Comments** |
| player | String |  |
| topScore | Integer |  |
| topScoreTime | String | Date format : yyyy-MM-dd |
| lowScore | Integer |  |
| lowScoreTime | String | Date format : yyyy-MM-dd |
| averageScore | Integer |  |
| allScores | List<Integer> |  |

**Response Json**

|  |
| --- |
| {  "code": 200,  "errCode": **null**,  "messages": [  "OK"  ],  "errItems": **null**,  "result": {  "topScore": 500,  "topScoreTime": "2020-12-25 00:00:00",  "lowScore": 10,  "lowScoreTime": "2020-12-28 12:40:50",  "averageScore": 218,  "allScores": [  500,  300,  270,  10,  10  ],  "player": "rashed"  }  } |

Note

1. To keep simplicity auto increment number is used as id starting from 1.
2. Try to keep simplicity of code to follow K.I.S.S
3. DateTime format “yyyy-MM-dd hh:mm:ss” is implemented.
4. REST API convention has been implemented.
5. Integration test conducted on existing score table. No separate configuration files or tables are used due to sample project.