**Match and Match Odds REST CRUD Operations Documentation**

At application.properties file have been configured these properties:

spring.datasource.url=jdbc:postgresql://localhost:5432/springdemodb  
spring.datasource.username=postgres  
spring.datasource.password=password  
spring.jpa.hibernate.ddl-auto=create-drop  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect  
spring.jpa.properties.hibernate.format\_sql=true  
  
server.error.include-message=always

1. There is the need of a postrgres database at port 5432 with name dbname= springdemodb.
2. Username and password of the database (now they are username=postgres password=password).
3. The property spring.jpa.hibernate.ddl-auto=create-drop is used, which means that every time the application is stops will drop the schema and every time it starts will create it again.
4. At application startup there are added two Matches through MatchConfiguration Class and commandLineRunner.

We have two entities (1. Match & 2. MatchOdds).

They have a relationship of 1 to Many -> One Match may have Many Match\_Odds.

I use a bidirectional mapping through the two entities.

Also the entity Match has the field “sport” which is an Enum saved in the database with values 1. for Football & 2. for Basketball.

I have implemented the bellow Rest API entry points.

1. Get All Matches: http://localhost:8080/api/v1/match
2. Get Match by Id: [http://localhost:8080/api/v1/match/{id}](http://localhost:8080/api/v1/match/%7bid%7d)

example: http://localhost:8080/api/v1/match/2

1. Post a new Match: http://localhost:8080/api/v1/match

Body payload JSON like that:

{

        "description": "description test insert",

        "match\_date": "2022-04-28",

        "team\_a": "TRIPOLI",

        "team\_b": "ASTERAS",

        "sport": "Football"

 }

1. Put a Match (update a Match):  
   [http://localhost:8080/api/v1/match/{id}](http://localhost:8080/api/v1/match/%7bid%7d)

Body payload JSON like that:

{

        "description": "description test insert",

        "match\_date": "2022-04-28",

        "team\_a": "TRIPOLI",

        "team\_b": "ASTERAS",

        "sport": "Football"

 }

1. Delete a Match: [http://localhost:8080/api/v1/match/{id}](http://localhost:8080/api/v1/match/%7bid%7d)

example: <http://localhost:8080/api/v1/match/2>

1. Patch a Match. Updated partially.

At this example we update only the description of the match:

Patch [http://localhost:8080/api/v1/match/{id}](http://localhost:8080/api/v1/match/%7bid%7d)

example: <http://localhost:8080/api/v1/match/2>

Body payload:

Anything we want for description like “test description”.

1. Post Match Odd: http://localhost:8080/api/v1/matchOdds

Body payload JSON like that (the match id must be already in the database):

{

    "match":{"id":2},

    "specifier":"X",

    "odd":1.5

}

1. Get All Match Odds: <http://localhost:8080/api/v1/matchOdds>
2. Get Match Odd by Id: [http://localhost:8080/api/v1/matchOdds/{id}](http://localhost:8080/api/v1/matchOdds/%7bid%7d)

example: <http://localhost:8080/api/v1/matchOdds/2>

1. Put Match Odd (Update a Match Odd):

[http://localhost:8080/api/v1/match/{id}](http://localhost:8080/api/v1/match/%7bid%7d)

Body payload JSON like that:

{

    "match":{"id":1},

    "specifier":"X",

    "odd":1.5

}

1. Delete Match Odd: : [http://localhost:8080/api/v1/matchOdds/{id}](http://localhost:8080/api/v1/matchOdds/%7bid%7d)

example: <http://localhost:8080/api/v1/matchOdds/2>

1. Patch a Match Odd. Updated partially.

At this example we update only the odd percentage of the match odd:

Patch [http://localhost:8080/api/v1/matchOdss/{id}](http://localhost:8080/api/v1/matchOdss/%7bid%7d)

example: <http://localhost:8080/api/v1/match/3>

Body payload:

Any double value like: 3.1