

## DICE SIMULATION Project

### INSTRUCTION BEFORE RUNNING THE PROJECT:

Please configure the database as per your requirement in "*application.properties*" file.

Contents:

- A) About REST endpoints
- B) About Data base Design
- C) About SimulationService.java

### **A) ABOUT REST ENDPOINTS:**

As described in the assignment required endpoints are configured with *query parameters*.

For the given Java assignment, making sensible assumptions, total 3 REST end points are created.

All the REST end points are being exposed in the "ApplicationController" file.

Dice\src\main\java\com\synpulse\controller\ApplicationController.java

1st end point:

**@GetMapping("/simulate")**

**To this endpoint we need to send 3 values as query parameters i.e number of dice, number of sides and number of rolls.**

This endpoint is for the assignment->Create a REST endpoint to execute a dice distribution simulation.

Here we can configure noOfDice, noOfSides and noOfRolls and Validations has been written as required.

A sample call from Postman:

<http://localhost:8080/simulate?dice=9&sides=12&rolls=100>

This returns a JSON of SUM-COUNT pairs.

**sample output:**

```
{  
  "18": 1,  
  "20": 1,  
  "21": 2,  
  "22": 2,  
  "23": 5,  
  "24": 4,  
  "25": 6,  
  "26": 7,  
  "27": 8,  
  "28": 3,  
  "29": 13,  
  "30": 18,  
  "31": 7,  
  "32": 8,  
  "33": 9,  
  "34": 5,  
  "35": 12,  
  "36": 9,  
  "37": 9,  
  "38": 6,  
  "39": 11,  
  "40": 4,  
  "41": 2,  
  "42": 5,  
  "43": 5,  
  "44": 1,  
  "45": 3,  
  "46": 1,  
  "47": 2,  
  "48": 1  
}
```

2nd end point:

**@GetMapping("/totaldetails")**

This endpoint is for the assignment->Return the total number of simulations and total rolls made, grouped by all existing dice number–dice side combinations.

A sample call from Postman:

<http://localhost:8080/totaldetails>

This returns the required details of all the existing combinations as a JSON.

**Sample Output:**

```
[
  {
    "simulationId": 1,
    "simulationCount": 1,
    "totalRolls": 10,
    "diceNumber_diceSides": "3-6"
  },
  {
    "simulationId": 2,
    "simulationCount": 1,
    "totalRolls": 100,
    "diceNumber_diceSides": "6-10"
  },
  {
    "simulationId": 3,
    "simulationCount": 4,
    "totalRolls": 550,
    "diceNumber_diceSides": "8-12"
  }
]
```

3rd end point:

**@GetMapping("/relativedistribution")**

***To this endpoint we need to send 2 values as query parameters i.e number of dice and number of sides.***

This endpoint is for the assignment->For a given dice number–dice side combination, return the relative distribution, compared to the total rolls, for all the simulations.

A sample call from Postman:

<http://localhost:8080/relativedistribution?dice=5&sides=8> This returns the required details for the given diceNumber–diceSide combination as a JSON.

**Sample Output:**

```
[
  {
    "simulationId": 2,
    "sumOnDice": 5,
    "count": 1,
    "relativeDistribution": 1.25
  },
  {
    "simulationId": 2,
    "sumOnDice": 4,
    "count": 2,
    "relativeDistribution": 1.25
  },
  {
    "simulationId": 2,
    "sumOnDice": 16,
    "count": 2,
    "relativeDistribution": 1.25
  },
  {
    "simulationId": 2,
    "sumOnDice": 8,
    "count": 8,
    "relativeDistribution": 8.75
  },
  {
    "simulationId": 2,
    "sumOnDice": 9,
    "count": 7,
    "relativeDistribution": 7.5000005
  },
  {
    "simulationId": 2,
    "sumOnDice": 7,
    "count": 7,
```

```

    "relativeDistribution": 7.5000005
  },
  {
    "simulationId": 2,
    "sumOnDice": 14,
    "count": 4,
    "relativeDistribution": 3.7500002
  },
  {
    "simulationId": 2,
    "sumOnDice": 13,
    "count": 4,
    "relativeDistribution": 3.7500002
  },
  {
    "simulationId": 2,
    "sumOnDice": 15,
    "count": 4,
    "relativeDistribution": 3.7500002
  },
  {
    "simulationId": 2,
    "sumOnDice": 17,
    "count": 2,
    "relativeDistribution": 1.25
  },
  {
    "simulationId": 2,
    "sumOnDice": 6,
    "count": 4,
    "relativeDistribution": 3.7500002
  },
  {
    "simulationId": 2,
    "sumOnDice": 12,
    "count": 9,
    "relativeDistribution": 10.0
  },
  {
    "simulationId": 2,
    "sumOnDice": 11,
    "count": 14,
    "relativeDistribution": 16.25
  },
  {
    "simulationId": 2,
    "sumOnDice": 10,
    "count": 12,
    "relativeDistribution": 13.75
  }
]

```

## B) About Database Design:

Based on the requirement for the assignment, I have created 2 tables. Hence there are 2 Models and 2 repositories in the Project.

First table is SIMULATION and the other is DISTRIBUTION and these two tables are linked by a "SimulationId".

**Simulation table:** This table is to store the total number of simulations and total number of rolls for a specific diceNumber-diceSide combination. This table has the following columns

1. *simulation\_id* (Id given for a specific DiceNumber- diceSide combination).
2. *diceNumberDiceSides* (This is a string which is specific DiceNumber- diceSides combination).
3. *simulation\_count* (This is total number of simulations for the specific dice\_side combination)
4. *total\_rolls* (Total number of rolls for the specific dice\_side combination).

**Distribution table :** This table is to store the Sum on dice and the number of times it occurs for a specific diceNumber-diceSide combination. This table has following columns:

1. *Id* ( It's just a generated id)
2. *count* (Stores the count of a a specific Sum on the dice for a specific diceNumber-diceSide combination)
3. *sum\_on\_dice* ( Represents Sum on the dice for a specific diceNumber-diceSide combination)
4. *simulation\_id* ( Id that is taken from Simulation table. Basically each diceNumber-diceSide combination will have a simulationId).
5. *relativeDistribution* (stores the relativeDistribution compared to totalRolls for the specific diceNumber-diceSides combination)

C)**SimulationService.java:** It is the important class where entire functionalities are written.

**Three different functions** are written here, each corresponding to a REST endpoint.

- *createSimulation* Function is to create a simulation, return the SUM-COUNT pairs as JSON and Store all the related information in 2 Tables.
- *getAllSimulationDetails* Function Returns Total Simulations and Total number of rolls for all the existing diceNumber-diceSides combinations.
- *getRelativeDistribution* Function returns the Relative Distribution compared to Total number of rolls for a given diceNumber-diceSides combination.