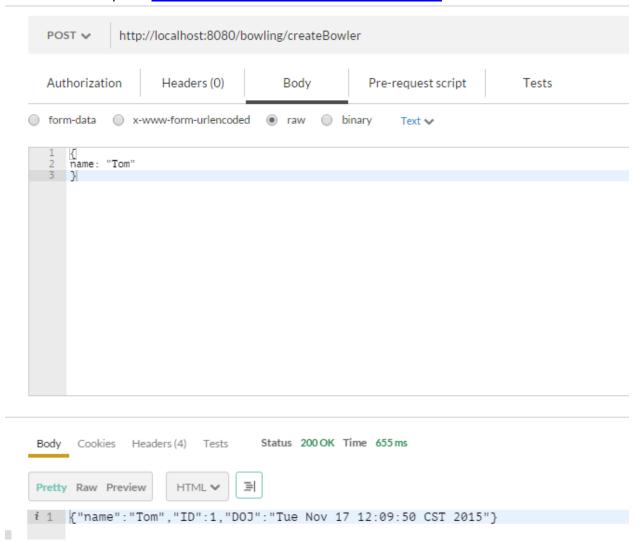
Bowling API Documentation:

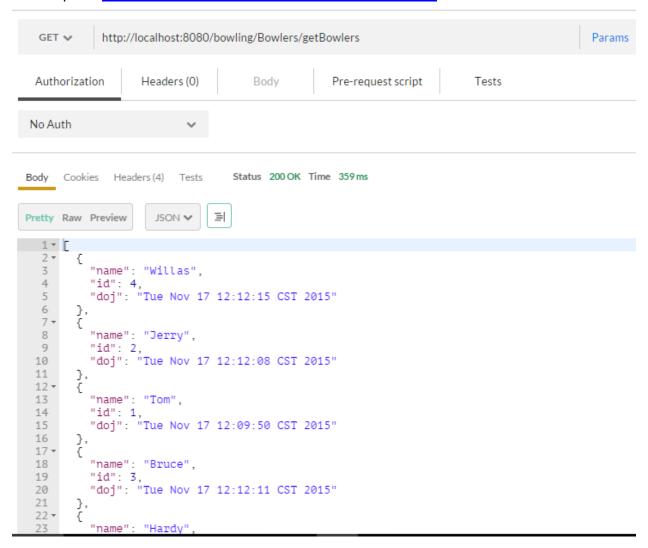
1) Create New Bowlers:

POST Request: http://localhost:8080/bowling/createBowler



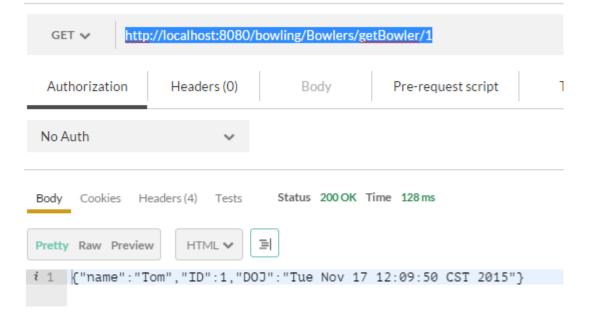
2) Get all Bowlers:

GET Request: http://localhost:8080/bowling/Bowlers/getBowlers



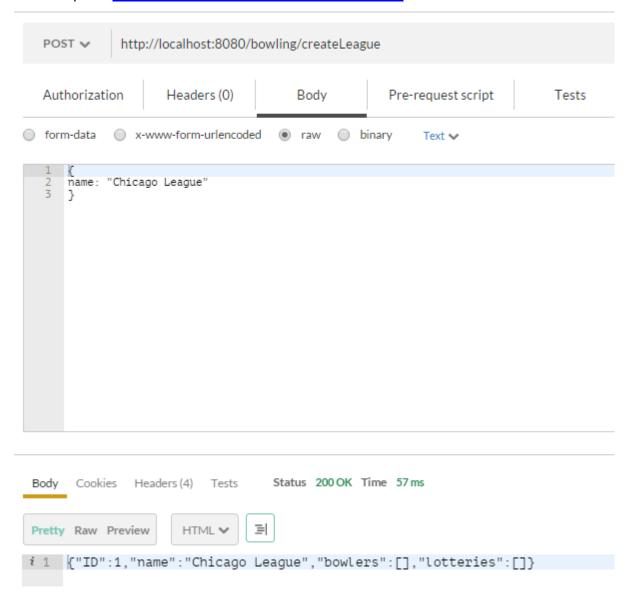
3) Get Bowlers by ID:

GET Request: http://localhost:8080/bowling/Bowlers/getBowler/{bowlerID}



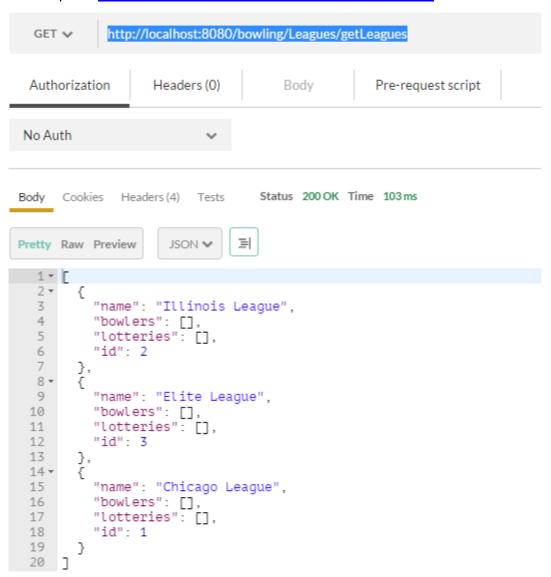
4) Create League:

POST Request: http://localhost:8080/bowling/createLeague



5) Get all Leagues:

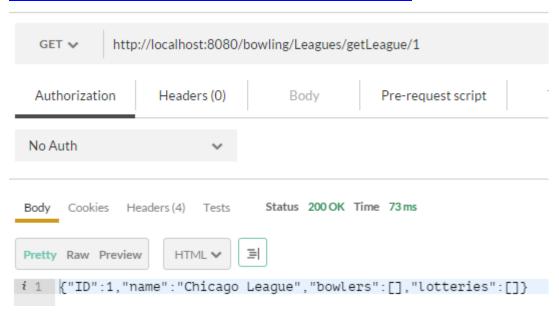
GET Request: http://localhost:8080/bowling/Leagues/getLeagues



6) Get League by ID:

GET Request:

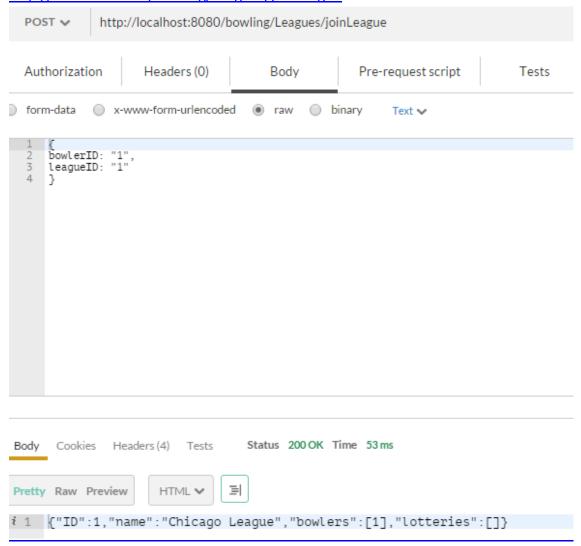
http://localhost:8080/bowling/Leagues/getLeague/{leagueID}



7) Join League:

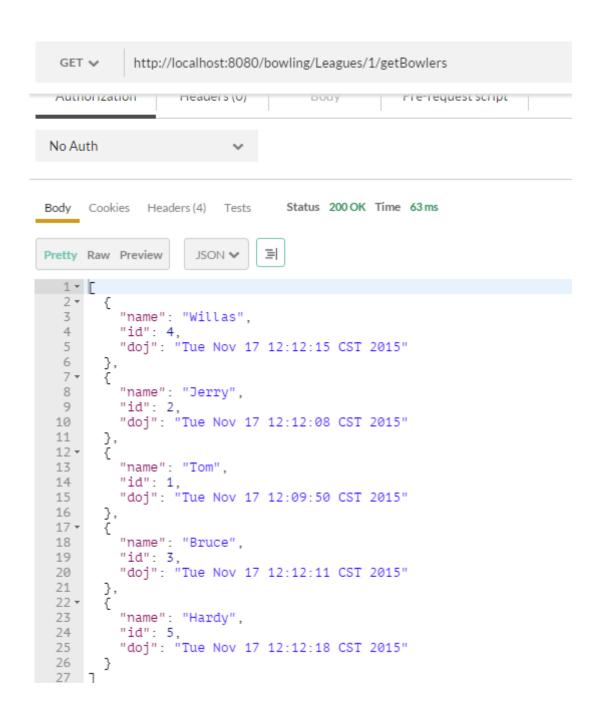
POST Request:

http://localhost:8080/bowling/Leagues/joinLeague



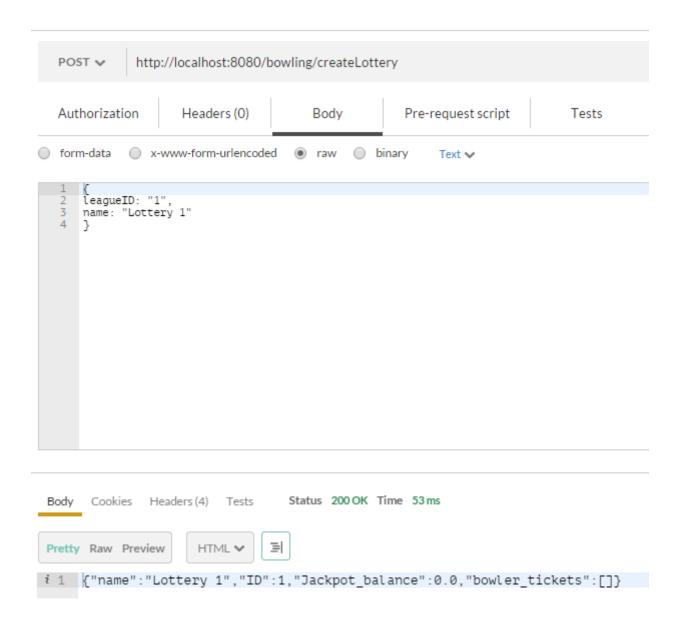
8) Get Bowlers who Joined a League:

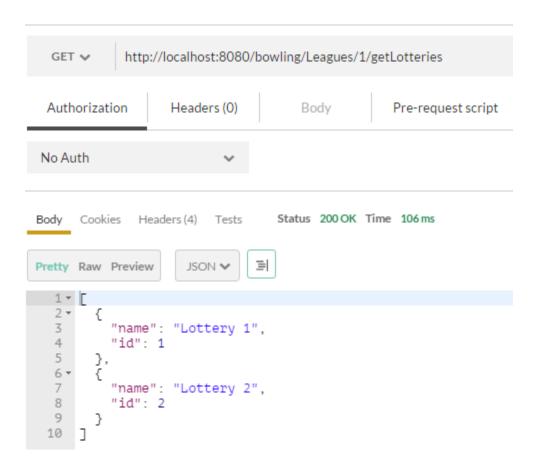
GET Request: http://localhost:8080/bowling/Leagues/{leagueID}/getBowlers



9) Create Lottery:

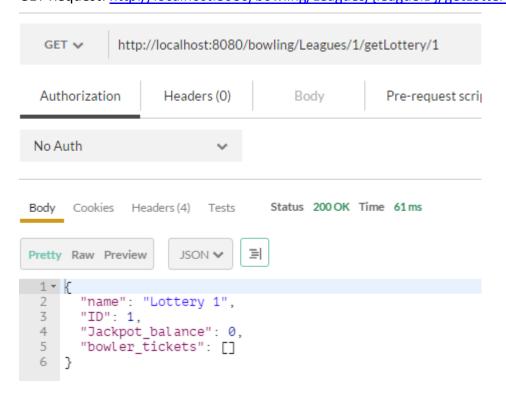
POST Request: http://localhost:8080/bowling/createLottery





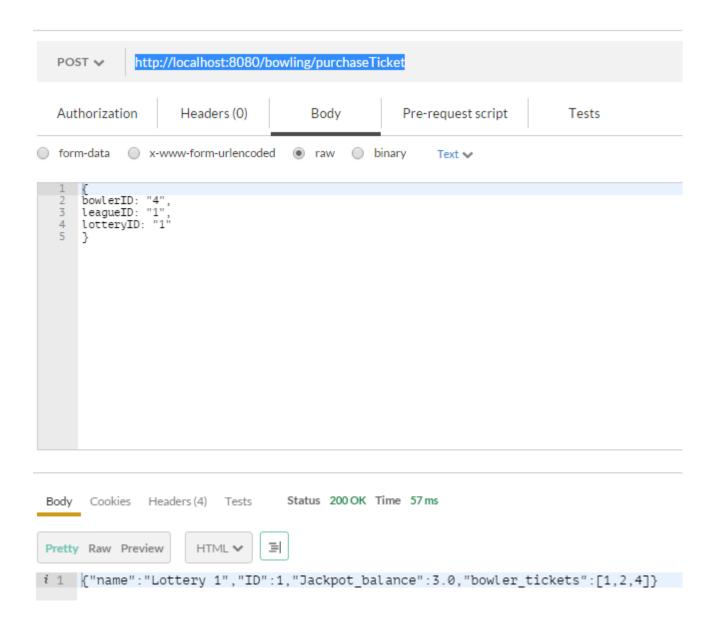
11) Get a specific lottery based on Lottery ID and League ID:

GET Request: http://localhost:8080/bowling/Leagues/{leagueID}/getLottery/{lotteryID}



12) Purchase Ticket:

POST Request: http://localhost:8080/bowling/purchaseTicket

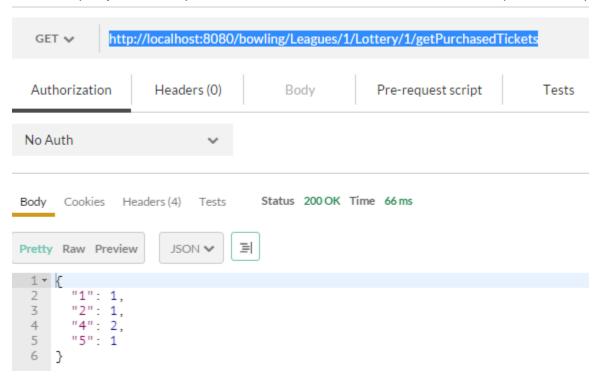


GET Request:

http://localhost:8080/bowling/Leagues/{leagueID}/Lottery/{lotteryID}/getPurchasedTickets

Output:

A HashMap object with Key as BowlerID and value as the number of tickets purchased by that bowler.

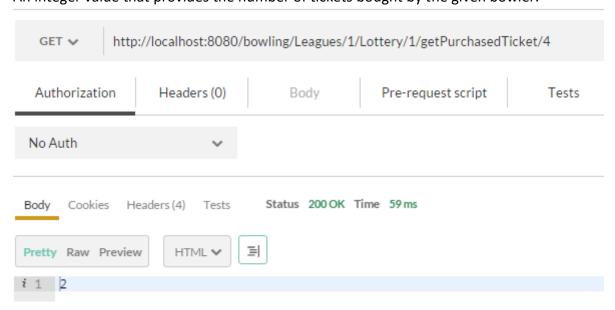


14) Get purchased Ticket for a particular bowler ID based on Lottery ID and League ID:

GET Request:

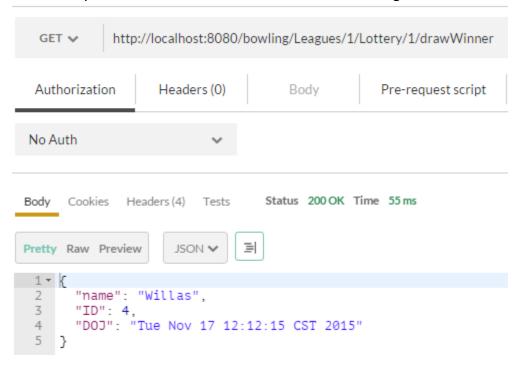
http://localhost:8080/bowling/Leagues/{leagueID}/Lottery/{lotteryID}/getPurchasedTicket/{BowlerID} Output:

An integer value that provides the number of tickets bought by the given bowler.



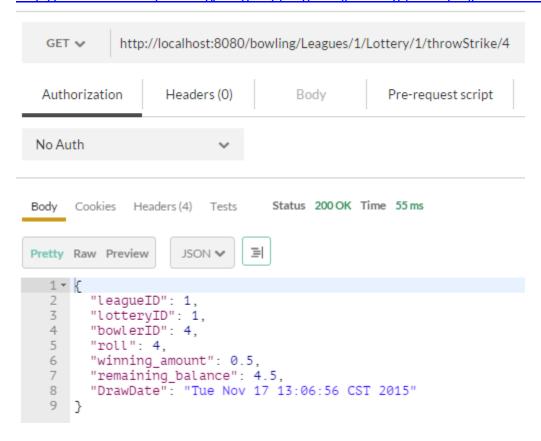
GET Request: http://localhost:8080/bowling/Leagues/{leagueID}/Lottery/{lotteryID}/drawWinner
Output:

A Randomly selected Bowler from all the bowlers who bought ticket for the draw.

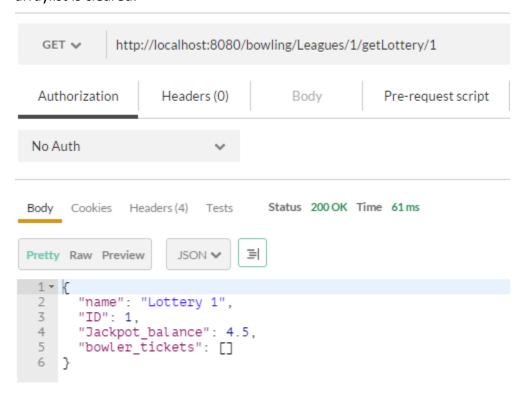


16) Providing a chance to the winner of the draw to claim Jackpot by throwing a strike for the given lottery: GET Request:

http://localhost:8080/bowling/Leagues/{leagueID}/Lottery/{lotteryID}/throwStrike/{winnerID}



Note: After throwStrike request, balance gets carry forward for the next weeks draw. Also the tickets arraylist is cleared.



Data Storage:

All data is getting stored in HashSet as shown below. As serialization is not performed at start of the server the HashSet objects get reinitialized.

```
package com.shouvik.bean;
import java.util.HashSet;
public class BowlingClub{
   public static HashSet<Bowler> bowlers = new HashSet<Bowler>();
   public static HashSet<League> leagues = new HashSet<League>();
   public static HashSet<Payment> payments = new HashSet<Payment>();
```

Things remaining:

- Fixing a day for the draw and error handling if DrawWinner request is called before that day.
- Allowing ticket sale till a day before the draw date. Freezing the ticket sale on the draw date.
- Displaying the Jackpot payout history data which is stored in the data model named payments.
- Serializing Datastore into a local file when the server stops to maintain persistence.