Sports Betting Application

Application is written in Spring Boot / Gradle 4.3.1 or higher version

Enterprise Application for registering Bookies and Players in the System, and taking action as Bookie for offering new Bets for the Players, and the Players for placing those Bets.

3 Users in system: AdminSport, Bookie, Player

Admin can add / update / remove Bookies and view Booki / all Bookies, can add / update / remove Players and view Player / all Players Bookie can add / update / remove Bet and view Bet / view All His Bets Player can place Beet and view All his Bets

The dependencies for the application are:

JPA – allows us using Hibernate technology implements **JPA**, **ORM,** and **JPQL** queries.

H2 DB – Embedded Data Base, creates the 5 tables when application get started up and drops them after application ends.

Web – **RestFull Web services** exposed by using **Single Page Applicaion**, **AngularJs**.

Architecture responsibilities and layers are as follows:

Application	Main Application	
Controller	Rest Controllers	RestFull Web Services – application URLs
Service	Business Logic	Implementations of methods with business logic
DAL	Date Access Layer	JPA Repositories for managing DB CRUD transactions

We create and drop 5 tables behind the scenes, using the **javax.persistence** for managing them and the relationships between them, as shown here:

All the information in the **application.properties** file as the **port:8081** and **SQL** information.

All 5 Tables will be created on Application StartUp and will be droped on Shutdown

Bookie	**Player**	**Bet**
bookieId (PK) Long	playerId (PK) Long	betId (PK) Long
bookieName string	playerName string	betTitle string
bookiePassword string	playerPassword string	sport enum
bookieEmail string	playerEmail string	eventDate date
		betWager double
		betOdds double
		betImage string
		betReceipt string

Player_Bet	**Bookie_Bet**
PlayerId (FK) Long	bookieId (FK) Long
betId (FK) Long	betId (FK) Long

Bookie contains also a Collection of Bets (bookieBets), and Player contains also Collection of Bets (playerBets),we use **ORM – Object Relationships Mapping** between the tables:

@OneToMany annotation between Bookie and Bet, One Bookie can offer Many Bets.

And cascade type to declare what is happening to Bets related to Bookie (bookieBets)

cascade = { CascadeType.PERSIST, CascadeType.MERGE }, fetch = FetchType.LAZY)

@**JoinTable**(name = "BOOKIE_BETS", joinColumns = @JoinColumn(name = "BOOKIE_ID"), inverseJoinColumns = @JoinColumn(name = "BET_ID"))

@ManyToMany annotation between Player and Bet, Many Players can take Many Bets.

And cascade type to declare what is happening to Bets related to Player (playerBets)

cascade = { CascadeType.PERSIST, CascadeType.MERGE }, fetch = FetchType.LAZY)

@**JoinTable**(name = "PLAYER_BETS", joinColumns = @JoinColumn(name = "PLAYER_ID"), inverseJoinColumns = @JoinColumn(name = "BET_ID"))

Bet related to Bookie - @**ManyToOne** , Many Bets can related to One Bookie.

Bet is related to Player - @**ManyToMany** (**mappedBy = "playerBets"**),

Many Bets can be placed by Many Players – mappedBy playerBets.

FetchType is **Lazy** here so to get Join Query to find bookieBets and playerBets in the BetRepository we use custom queries for it

JPQL @**Query** for getting bookieBets by bookieId:

Value = "SELECT b FROM Bet b WHERE b.betId IN (SELECT b.betId FROM Bookie bk WHERE bk.bookieId=?1)");

JPQL @**Query** for getting playerBets by playerId:

Value = "SELECT b FROM Bet b WHERE b.betId IN (SELECT b.betId FROM b.players p WHERE p.playerId=?1)")";

Enter to AdminSport username "admin", password "1111" and start using the Application.