Demo UserEvents project:

Technology : Spring data-JPA, SpringBoot, MySQL , SpringBoot Security, Thymeleaf

1.Important Test cases :

Cancel: **Did not use cascade on delete as it can sometimes delete not only associations but other table's entities.**

U0 creates 3 events E1, E2, E3
U1 joins E1, E2 :
U2 joins E1, E3 :

U1 cancels E1, U2 still has joining status for E1 , U3 has joining status

Delete: Again did not use Cascade type REMOVE for same reason as above, instead manually deleting using JPQL

When deleted from user_ event, entries are not deleted from user table.

2.Mappings:
a) User_Event - ManyToMany - User is the owning side of relationship, which User joining which Event
b) Event has Many to One with User : User can create many events
c) One event has many messages
d) One user has many messages

3.JPA design:

1.Did not use cascade REMOVE as it can delete other table entities
2.@Transactional - For performance used default lazy fetch ToMany and ManyToMany,
3.@Transactional - when there is a read after write - session closed in write and throws Exception - DAORepositoyService
4.Delete implemented in 2 ways - Done manually as no cascade REMOVE, in one query for performance using JPQL, otherwise too many queries are fired if delete by single row

3.. Code of addJoiningEvent and deleteJoiningEvent, deleteEvent

4.. **Create queries and all constraints**

QUERIES:

1. **User:**

```
CREATE TABLE IF NOT EXISTS user(
    id INT NOT NULL AUTO_INCREMENT,
    first_Name VARCHAR(255) NOT NULL,
    last_Name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    location VARCHAR(255)  NOT NULL ,
    state VARCHAR(20)   NOT NULL,
    password VARCHAR(80) NOT NULL,
    PRIMARY KEY (id),
    UNIQUE KEY (email)
)  ENGINE=INNODB;
```

2. **Event**

```
CREATE TABLE IF NOT EXISTS Event(
    id INT NOT NULL AUTO_INCREMENT,
    event_Name VARCHAR(255) NOT NULL,
    event_Location VARCHAR(255) NOT NULL,
    event_Date Date NOT NULL,
    state VARCHAR(20)  NOT NULL,
    user_Created_id INT NOT NULL,
    PRIMARY KEY (id)
)  ENGINE=INNODB;
```

```
ALTER TABLE event
ADD CONSTRAINT FK_created
FOREIGN KEY (user_Created_id) REFERENCES user(id)
```

3. **User_Event**

```
CREATE TABLE IF NOT EXISTS user_event(
    users_id INT NOT NULL ,
    events_id INT NOT NULL ,
    PRIMARY KEY (users_id,events_id)

)  ENGINE=INNODB;


ALTER TABLE user_event
ADD CONSTRAINT FK_ue1
FOREIGN KEY (users_id) REFERENCES user(id);



ALTER TABLE user_event
ADD CONSTRAINT FK_ue2
FOREIGN KEY (events_id) REFERENCES event(id)
```

4. **Message**

```
CREATE TABLE IF NOT EXISTS Message(
        id INT NOT NULL AUTO_INCREMENT,
         message VARCHAR(300) NOT NULL,
         message_Time TIMESTAMP NOT NULL,
         event_message_id INT  NOT NULL,
         user_sent_id INT NOT NULL,
         PRIMARY KEY (id)
    )  ENGINE=INNODB;


ALTER TABLE Message
ADD CONSTRAINT FK_message1
FOREIGN KEY (user_sent_id) REFERENCES user(id)

ALTER TABLE Message
ADD CONSTRAINT FK_message2
FOREIGN KEY (event_message_id) REFERENCES Event(id)


USE INFORMATION_SCHEMA;
SELECT *
FROM KEY_COLUMN_USAGE
```

```
WHERE TABLE_SCHEMA = "<eventdb>"
    AND TABLE_NAME = "<user>"
    AND REFERENCED_COLUMN_NAME IS NOT NULL;
```

5. Code Design :

No business logic in controller

No @Transactional in controller

Package structure