Pawan Gupta

Spring Data JPA with Hibernate Part 3

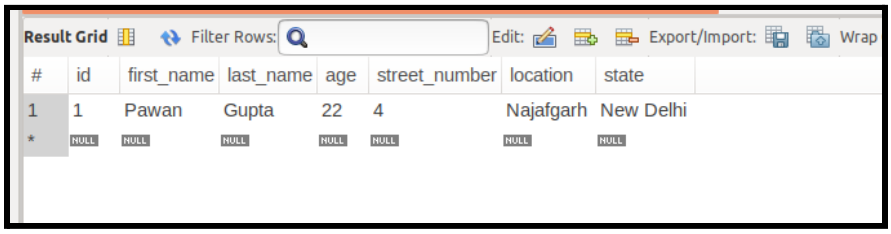
1. Create a class Address for Author with instance variables streetNumber, location, State.

```java
@Embeddable
public class Address {
//Author with instance variables streetNumber, location, State.    /
    private int streetNumber;
    private String location;
    private String state;

    public int getStreetNumber() {
        return streetNumber;
    }

    public void setStreetNumber(int streetNumber) {
        this.streetNumber = streetNumber;
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public String getState() {
        return state;
    }
```

```java
import javax.persistence.*;

@Entity
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String firstName;
    private String lastName;
    private int age;

    public void setAge(int age) {
        this.age = age;
    }

    @Embedded
    private Address address;

    public int getId() {
        return id;
    }

    public String getFirstName() {
        return firstName;
    }
```

2. Create instance variable of Address class inside Author class and save it as embedded object.

```
public void createRecord()
{
    Author author=new Author();
    author.setAge(22);
    author.setFirstName("Pawan");
    author.setLastName("Gupta");

    Address address=new Address();
    address.setStreetNumber(4);
    address.setLocation("Najafgarh");
    address.setState("New Delhi");
    author.setAddress(address);

    repository.save(author);
}
```



#	id	first_name	last_name	age	street_number	location	state
1	1	Pawan	Gupta	22	4	Najafgarh	New Delhi
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Introduce a List of subjects for author.

```java
import javax.persistence.*;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String firstName;
    private String lastName;
    private int age;
    @OneToMany(mappedBy = "author",cascade =CascadeType.ALL)//t
    private Set<Subject> subjects;

    public Set<Subject> getSubjetcs() {
        return subjects;
    }

    public void setSubjetcs(Set<Subject> subjetcs) {
        this.subjects= subjetcs;
    }

    public void setAge(int age) {
```

```java
@Entity
public class Subject {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String subjectName;


    @ManyToOne
    @JoinColumn(name="author_id")
    private Author author;

    public String getSubjectName() {
        return subjectName;
    }

    public void setSubjectName(String subjectName) {
        this.subjectName = subjectName;
    }

    public Author getAuthor() {
        return author;
    }
```

4. Persist 3 subjects for each author.

```java
public void createRecord()
{
    Author author=new Author();
    author.setAge(22);
    author.setFirstName("Pawan");
    author.setLastName("Gupta");

    Address address=new Address();
    address.setStreetNumber(4);
    address.setLocation("Najafgarh");
    address.setState("New Delhi");
    author.setAddress(address);
    //Add Subject here
    Subject subject1=new Subject();
    subject1.setSubjectName("Java");

    Subject subject2=new Subject();
    subject2.setSubjectName("C++");

    Subject subject3=new Subject();
    subject3.setSubjectName("DataBase");

    author.addSubject(subject1);
    author.addSubject(subject2);
    author.addSubject(subject3);
    repository.save(author);
}
```

#	id	first_name	last_name	age	street_number	location	state
1	1	Pawan	Gupta	22	4	Najafgarh	New Delhi
2	4	Pawan	Gupta	22	4	Najafgarh	New Delhi
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

#	id	subject_name	author_id
1	1	C++	4
2	2	DataBase	4
3	3	Java	4
*	NULL	NULL	NULL

5. Create an Entity book with an instance variable bookName.

```java
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String bookName;


    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @OneToOne(cascade=CascadeType.ALL)
    @JoinColumn(name = "author_id")
    private Author author;

    public String getBookName() { return bookName; }

    public void setBookName(String bookName) { this.bookName = b

    public Author getAuthor() { return author; }

    public void setAuthor(Author author) { this.author = author;
```

AssociationExcerciseApplication

6. Implement One to One mapping between Author and Book.

```
Hibernate: insert into author (location, state, street_number, age, first_name, last_name) values (?, ?, ?, ?, ?, ?)
Hibernate: insert into subject (author_id, subject_name) values (?, ?)
Hibernate: insert into subject (author_id, subject_name) values (?, ?)
Hibernate: insert into subject (author_id, subject_name) values (?, ?)
Hibernate: insert into book (author_id, book_name) values (?, ?)
Hibernate: select author0_.id as id1_0_1_, author0_.location as location2_0_1_, author0_.state as state3_0_1_, author0_.str
Hibernate: select book0_.id as id1_1_1_, book0_.author_id as author_i3_1_1_, book0_.book_name as book_nam2_1_1_, author1_.1
2021-03-16 16:49:16.692  INFO 9686 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManag
```

#	id	first_name	last_name	age	street_number	location	state
1	1	Pawan	Gupta	22	4	Najafgarh	New Delhi
2	4	Pawan	Gupta	22	4	Najafgarh	New Delhi
3	5	Pawan	Gupta	22	4	Najafgarh	New Delhi
4	6	Pawan	Gupta	22	4	Najafgarh	New Delhi
5	7	Pawan	Gupta	22	4	Najafgarh	New Delhi
6	9	Gaurav	Rawat	22	4	Najafgarh	New Delhi
7	10	Gaurav	Rawat	22	4	Najafgarh	New Delhi
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

id	book_name	author_id
1	Master in hibernet	NULL
2	Master in hibernet	NULL
4	Master in hibernet	10
NULL	NULL	NULL

```java
Book book=new Book();
book.setBookName("Master in hibernet");
book.setAuthor(author);

bookRepository.save(book);
repository.save(author);
```

7. Implement One to Many Mapping between Author and Book(Unidirectional, BiDirectional and without additional table) and implement cascade save.

```
Hibernate: insert into author (location, state, street_number, age, first_name, last_name) values (?, ?, ?, ?, ?, ?)
Hibernate: insert into book (author_id, book_name) values (?, ?)
Hibernate: insert into book (author_id, book_name) values (?, ?)
Hibernate: insert into book (author_id, book_name) values (?, ?)
Hibernate: insert into subject (author_id, subject_name) values (?, ?)
Hibernate: insert into subject (author_id, subject_name) values (?, ?)
Hibernate: insert into subject (author_id, subject_name) values (?, ?)
```

#	id	book_name	author_id
2	2	NULL	13
3	3	NULL	13
4	4	Master in Java	14
5	5	Master in database	14
6	6	NULL	14
7	7	Master in Java	15
8	8	Master in database	15
9	9	Master in c++	15
*	NULL	NULL	NULL

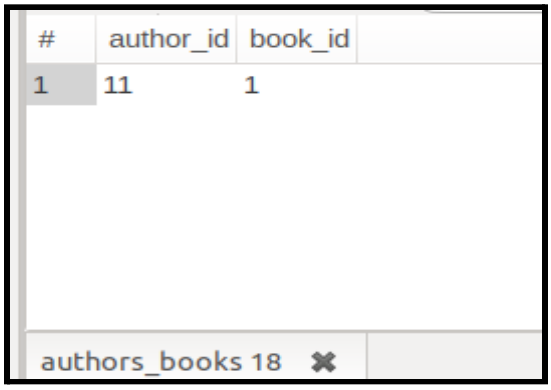
#	id	first_name	last_name	age	street_number	location	state
4	6	Pawan	Gupta	22	4	Najafgarh	New Delhi
5	7	Pawan	Gupta	22	4	Najafgarh	New Delhi
6	9	Gaurav	Rawat	22	4	Najafgarh	New Delhi
7	10	Gaurav	Rawat	22	4	Najafgarh	New Delhi
8	11	Gaurav	Rawat	22	4	Najafgarh	New Delhi
9	13	Shubham	Pandey	22	4	Shyam Vihar	New Delhi
10	14	Shubham	Pandey	22	4	Shyam Vihar	New Delhi
11	15	Shubham	Pandey	22	4	Shyam Vihar	New Delhi
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
60
61      Book book1=new Book();
62      book1.setBookName("Master in Java");
63
64      Book book2=new Book();
65      book2.setBookName("Master in c++");
66
67      Book book3=new Book();
68      book3.setBookName("Master in database");
69
60      author.addBooks(book1);
61      author.addBooks(book2);
62      author.addBooks(book3);
63  //    HashSet<Book> books =  new HashSet<Book>();
64  //    Book book=new Book();
65  //    book.setBookName("Master IN Java");
66  //    books.add(book);
67
```

```java
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String bookName;


    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name="author_id")
    private Author author;
```

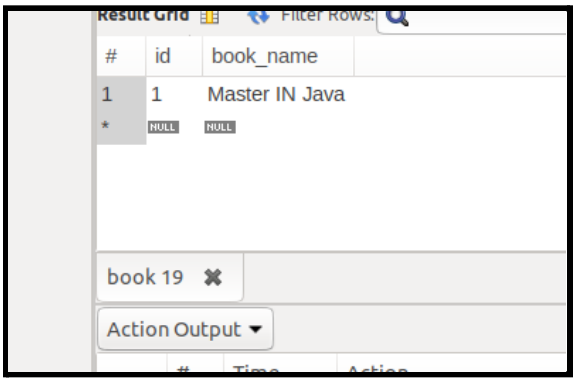
```java
    @OneToMany(mappedBy = "author",cascade =CascadeType.ALL)//table name(entity
    private Set<Subject> subjects;

    @OneToMany(mappedBy = "author",cascade =CascadeType.ALL)
    private Set<Book> books;
    public Set<Book> getBooks() {
        return books;
    }

    public void setBooks(Set<Book> books) {
        this.books = books;
    }
```

8. Implement Many to Many Mapping between Author and Book.



#	author_id	book_id
1	11	1

authors_books 18

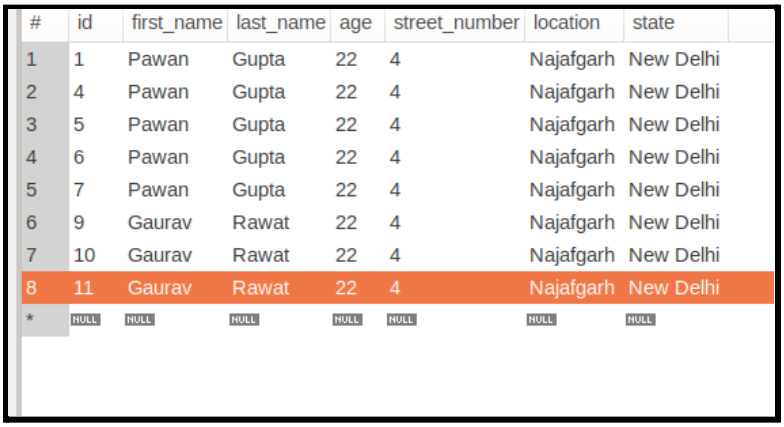


Result Grid Filter Rows:

#	id	book_name
1	1	Master IN Java
*	NULL	NULL

book 19

Action Output



#	id	first_name	last_name	age	street_number	location	state
1	1	Pawan	Gupta	22	4	Najafgarh	New Delhi
2	4	Pawan	Gupta	22	4	Najafgarh	New Delhi
3	5	Pawan	Gupta	22	4	Najafgarh	New Delhi
4	6	Pawan	Gupta	22	4	Najafgarh	New Delhi
5	7	Pawan	Gupta	22	4	Najafgarh	New Delhi
6	9	Gaurav	Rawat	22	4	Najafgarh	New Delhi
7	10	Gaurav	Rawat	22	4	Najafgarh	New Delhi
8	11	Gaurav	Rawat	22	4	Najafgarh	New Delhi
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

9. Which method on the session object can be used to remove an object from the cache?

   Evict() is used to remove an object.

10. What does @transactional annotation do?

The *@Transactional* annotation is the metadata that specifies the semantics of the transactions on a method. We have two ways to rollback a transaction: declarative and programmatic.In the declarative approach, we annotate the methods with the *@Transactional* annotation. The *@Transactional* annotation makes use of the attributes *rollbackFor* or *rollbackForClassName* to rollback the transactions, and the attributes *noRollbackFor* or *noRollbackForClassName* to avoid rollback on listed exceptions.