

Spring Data JPA with Hibernate Part 3

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

```
package com.bootcamp.mappings.Entities;
```

```
import javax.persistence.Embeddable;
```

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

```
@Embeddable
```

```
public class Address {
```

```
    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;
```

```
    }
```

```
    public void setState(String state) {
```

```
        this.state = state;
```

```
    }
```

```
}
```

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

```
package com.bootcamp.mappings.Entities;
```

```
import javax.persistence.Embedded;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;
```

```
public class Author {
```

```
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private int id;  
    private String name;
```

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

```
    @Embedded  
    private Address address;
```

```
    public int getId() {  
        return id;  
    }
```

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

```
    public String getName() {  
        return name;  
    }
```

```
    public void setName(String name) {  
        this.name = name;  
    }
```

```
    public Address getAddress() {  
        return address;  
    }
```

```
    public void setAddress(Address address) {  
        this.address = address;  
    }  
}
```

3. Introduce a List of subjects for author.
4. Persist 3 subjects for each author.
5. Create an Entity book with an instance variable bookName.
6. Implement One to One mapping between Author and Book.
7. Implement One to Many Mapping between Author and Book(Unidirectional, BiDirectional and without additional table) and implement cascade save.

File=Book.java

```
package com.bootcamp.mappings.Entities;
```

```
import javax.persistence.*;
```

```
@Entity
```

```
public class Book {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private int id;
```

```
//Ques 5 Create an Entity book with an instance variable bookName.
```

```
    private String bookname;
```

```
//Ques6 Implement One to One mapping between Author and Book.
```

```
//@OneToOne
```

```
//Ques7 Unidirectional- many to one without using mapped by
```

```
//author_id is a foreign key
```

```
    @ManyToOne
```

```
    @JoinColumn(name = "author_id")
```

```
    private Author author;
```

```
public int getId() {
```

```
    return id;
```

```
}
```

```
public void setId(int id) {
```

```
    this.id = id;
```

```
}
```

```
public String getBookname() {
```

```
    return bookname;
```

```
}
```

```
public void setBookname(String bookname) {
```

```
    this.bookname = bookname;
```

```
}}
```

File=Author.java

```
package com.bootcamp.mappings.Entities;
```

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

```
@Entity
```

```
public class Author {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private int id;
```

```
    private String name;
```

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

```
    @Embedded
```

```
    private Address address;
```

```
//Ques6 Implement One to One mapping between Author and Book.
```

```
// @OneToOne(cascade = CascadeType.ALL)
```

```
// *****
```

```
//Ques7 Implement One to Many Mapping between Author and Book(Unidirectional, BiDirectional and without additional table ) and implement cascade save.
```

```
//Cascade means whatever operations are performed on Author ,reflect or perform it to Book also
```

```
    @OneToMany(mappedBy = "author",cascade = CascadeType.ALL)
```

```
    private Set<Book> books;
```

```
//Ques3 Introduce a List of subjects for author.
```

```
    @ElementCollection
```

```
    private List<String> subjects;
```

```
    public List<String> getSubjects() {
```

```
        return subjects;
```

```
    }
```

```
    public void setSubjects(List<String> subjects) {
```

```
        this.subjects = subjects;
```

```
    }
```

```
    public int getId() {
```

```
        return id;
```

```
}

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

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public Address getAddress() {
    return address;
}

public void setAddress(Address address) {
    this.address = address;
}

public Set<Book> getBooks() {
    return books;
}

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

}
```

File=MappingsApplicationTests.java

```
package com.bootcamp.mappings;
```

```
import com.bootcamp.mappings.Entities.Address;  
import com.bootcamp.mappings.Entities.Author;  
import com.bootcamp.mappings.Entities.Book;  
import com.bootcamp.mappings.repositories.AuthorRepo;  
import org.junit.jupiter.api.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.context.SpringBootTest;
```

```
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.List;
```

```
@SpringBootTest
```

```
class MappingsApplicationTests {
```

```
    @Autowired
```

```
    AuthorRepo authorRepo;
```

//this method will not reflect changes in the book table ,because we haven't tell the hibernate to save the changes in the book table ,To do so we need to cascade

```
    @Test
```

```
    public void testCreate(){
```

```
        Author author = new Author();  
        author.setName("Ruskin Bond");
```

```
        Address address=new Address();  
        address.setLocation("GTB Nagar");  
        address.setState("Delhi");  
        address.setStreetNumber(10);
```

```
        author.setAddress(address);  
        HashSet<Book> b1=new HashSet<Book>();
```

```
        Book book1=new Book();  
        book1.setBookname("Delhi is not far");  
        b1.add(book1);
```

```
        Book book2=new Book();  
        book2.setBookname("Angry river");  
        b1.add(book2);
```

```
        author.setBooks(b1);  
//Ques4 Persist 3 subjects for each author  
        List<String> subjects=new ArrayList<>();
```

```
        subjects.add("Maths");
```

```

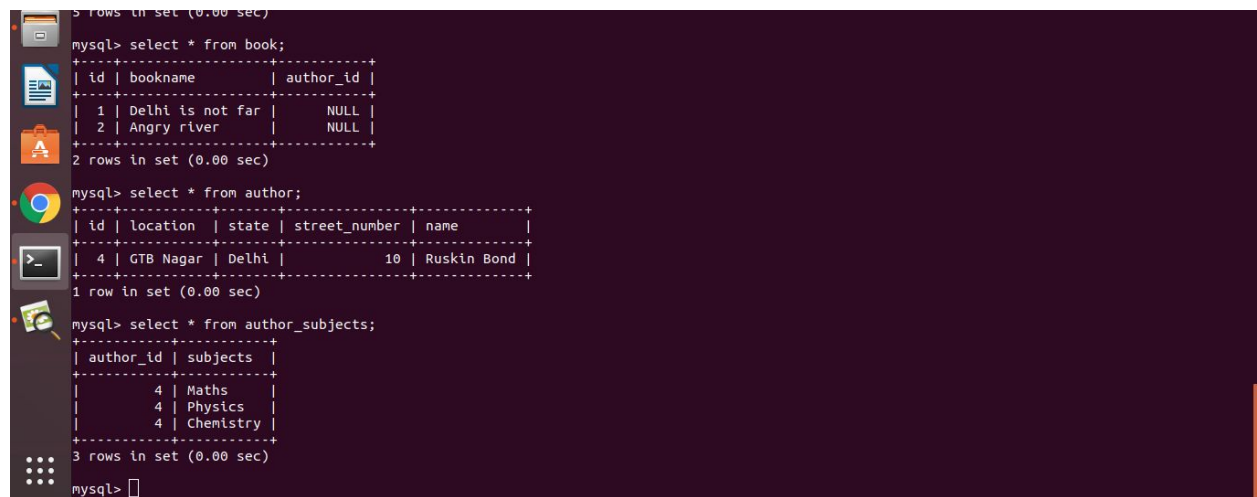
subjects.add("Physics");
subjects.add("Chemistry");

author.setSubjects(subjects);

authorRepo.save(author);
}

}

```



The screenshot shows a MySQL terminal window with the following queries and results:

```

mysql> select * from book;
+-----+
| id | bookname      | author_id |
+-----+
| 1  | Delhi is not far | NULL      |
| 2  | Angry river    | NULL      |
+-----+
2 rows in set (0.00 sec)

mysql> select * from author;
+-----+
| id | location | state | street_number | name      |
+-----+
| 4  | GTB Nagar | Delhi | 10             | Ruskin Bond |
+-----+
1 row in set (0.00 sec)

mysql> select * from author_subjects;
+-----+
| author_id | subjects |
+-----+
| 4         | Maths    |
| 4         | Physics  |
| 4         | Chemistry|
+-----+
3 rows in set (0.00 sec)

mysql>

```

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

File=Book.java

```
package com.bootcamp.mappings.Entities;
```

```
import javax.persistence.*;
```

```
@Entity
```

```
public class Book {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private int id;
```

```
//Ques 5 Create an Entity book with an instance variable bookName.
```

```
    private String bookname;
```

```
//Ques6 Implement One to One mapping between Author and Book.
```

```
//@OneToOne
```

```
//Ques7 Unidirectional- many to one without using mapped by
```

```
//author_id is a foreign key
```

```
// @ManyToOne
```

```
// @JoinColumn(name = "author_id")
```

```
//Ques 8 Implement Many to Many Mapping between Author and Book.
```

```
    @ManyToMany(mappedBy = "book")
```

```
    private Author author;
```

```
    public int getId() {
```

```
        return id;
```

```
    }
```

```
    public void setId(int id) {
```

```
        this.id = id;
```

```
    }
```

```
    public String getBookname() {
```

```
        return bookname;
```

```
    }
```

```
    public void setBookname(String bookname) {
```

```
        this.bookname = bookname;
```

```
    }}
```


File=Author.java

```
package com.bootcamp.mappings.Entities;
```

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

```
@Entity
```

```
public class Author {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private int id;
```

```
    private String name;
```

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

```
    @Embedded
```

```
    private Address address;
```

```
//Ques6 Implement One to One mapping between Author and Book.
```

```
// @OneToOne(cascade = CascadeType.ALL)
```

```
// *****
```

```
//Ques7 Implement One to Many Mapping between Author and Book(Unidirectional, BiDirectional and without additional table ) and implement cascade save.
```

```
//Cascade means whatever operations are performed on Author ,reflect or perform it to Book also
```

```
// @OneToMany(mappedBy = "author",cascade = CascadeType.ALL)
```

```
//Ques 8 Implement Many to Many Mapping between Author and Book.
```

```
    @ManyToMany(cascade = CascadeType.ALL,fetch = FetchType.EAGER)
```

```
    @JoinTable(name="author_book",joinColumns = @JoinColumn(name = "author_id",referencedColumnName = "id"),inverseJoinColumns = @JoinColumn(name = "book_id",referencedColumnName = "id"))
```

```
    private Set<Book> books;
```

```
//Ques3 Introduce a List of subjects for author.
```

```
    @ElementCollection
```

```
    private List<String> subjects;
```

```
public List<String> getSubjects() {
```

```
    return subjects;
```

```
}

public void setSubjects(List<String> subjects) {
    this.subjects = subjects;
}

public int getId() {
    return id;
}

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

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public Address getAddress() {
    return address;
}

public void setAddress(Address address) {
    this.address = address;
}

public Set<Book> getBooks() {
    return books;
}

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

}
```

File=MappingsApplicationTests.java

```
package com.bootcamp.mappings;
```

```
import com.bootcamp.mappings.Entities.Address;  
import com.bootcamp.mappings.Entities.Author;  
import com.bootcamp.mappings.Entities.Book;  
import com.bootcamp.mappings.repositories.AuthorRepo;  
import org.junit.jupiter.api.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.context.SpringBootTest;
```

```
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.List;
```

```
@SpringBootTest
```

```
class MappingsApplicationTests {
```

```
    @Autowired
```

```
    AuthorRepo authorRepo;
```

```
//this method will not reflect changes in the book table ,because we haven't tell the hibernate to save the changes in  
the book table , To do so we need to cascade
```

```
    @Test
```

```
    public void testCreate(){
```

```
        Author author = new Author();  
        author.setName("Ruskin Bond");
```

```
        Address address=new Address();  
        address.setLocation("GTB Nagar");  
        address.setState("Delhi");  
        address.setStreetNumber(10);
```

```
        author.setAddress(address);  
        HashSet<Book> b1=new HashSet<Book>();
```

```
        Book book1=new Book();  
        book1.setBookname("Delhi is not far");  
        b1.add(book1);
```

```
        Book book2=new Book();  
        book2.setBookname("Angry river");  
        b1.add(book2);
```

```
        author.setBooks(b1);  
//Ques4 Persist 3 subjects for each author  
        List<String> subjects=new ArrayList<>();
```

```
    subjects.add("Maths");
    subjects.add("Physics");
    subjects.add("Chemistry");

    author.setSubjects(subjects);

    authorRepo.save(author);
}

}
```

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

evict() method is used to remove the entry from cache.

```
Session session=entityManager.unwrap(Session.class);  
session.evict();
```

10. What does @transactional annotation do?

This is much more convenient and readable, and is currently the recommended way to handle transactions in Spring.

By using @Transactional, many important aspects such as transaction propagation are handled automatically. In this case if another transactional method is called by businessLogic, that method will have the option of joining the ongoing transaction.

