Session : Spring Data JPA with Hibernate Part 3

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

Address.java

```
@Embeddable
public class Address {
  private String streetnumber;
  private String location;
  private String state;

  public String getStreetnumber() {
     return streetnumber;
  }

  public void setStreetnumber(String 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.**

CODE

Author.java

```
@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String name;
  @Embedded
  private Address address;

  public Integer getId() {
     return id;
  }

  public void setId(Integer 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;
  }
}
```

OUTPUT

```
mysql> use SpringDataJPA3;
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_SpringDataJPA3 |
+--------------------------+
| author                   |
| hibernate_sequence       |
+--------------------------+
2 rows in set (0.00 sec)

mysql> desc author;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| id           | int(11)      | NO   | PRI | NULL    |       |
| location     | varchar(255) | YES  |     | NULL    |       |
| state        | varchar(255) | YES  |     | NULL    |       |
| streetnumber | varchar(255) | YES  |     | NULL    |       |
| name         | varchar(255) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

3. **Introduce a List of subjects for author.**
4. **Persist 3 subjects for each author.**

CODE

Author.java

```
@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String name;
  @Embedded
  private Address address;
  @OneToMany(mappedBy = "author",cascade = CascadeType.ALL)  //all changes you do on author , do them in subject table as well.
  private List<Subject> subjects;

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

```
        return subjects;
    }

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

    public Integer getId() {
        return id;
    }

    public void setId(Integer 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 void addSubject(Subject subject)
    {
        if(subject != null)
        {
            if(subjects == null)
                subjects = new ArrayList<>();
        }
        subject.setAuthor(this);
        subjects.add(subject);
    }
}
```

Subject.java

```
@Entity
public class Subject {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String subjectname;
  @ManyToOne
  @JoinColumn(name = "authorid")
  private Author author;

  public Author getAuthor() {
     return author;
  }

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

  public Integer getId() {
     return id;
  }

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

  public String getSubjectname() {
     return subjectname;
  }

  public void setSubjectname(String subjectname) {
     this.subjectname = subjectname;
  }
}
```

AuthorController.java

```
@GetMapping("/create")
public String createAuthor()
```

```
{
  authorService.createAuthor();
  return "Author Added";
}
```

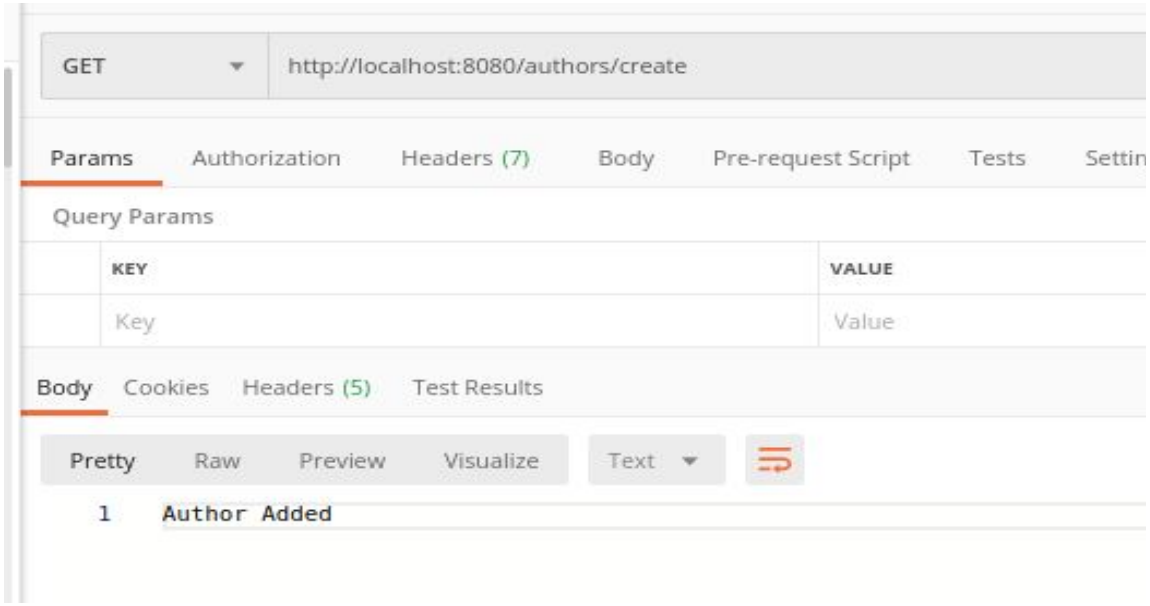
AuthorService.java

```
@Autowired
AuthorRepository authorRepository;

public void createAuthor()
{
  Address address = new Address();
  address.setStreetnumber("AL76");
  address.setLocation("Pitampura");
  address.setState("Delhi");

  Subject subject1 = new Subject();
  subject1.setSubjectname("Java");
  Subject subject2 = new Subject();
  subject2.setSubjectname("Python");
  Subject subject3 = new Subject();
  subject3.setSubjectname("Groovy");


  Author author = new Author();
  author.setName("Aayushi");
  author.setAddress(address);

  author.addSubject(subject1);
  author.addSubject(subject2);
  author.addSubject(subject3);

  authorRepository.save(author);
}
```

AuthorRepository.java

```
@Repository
public interface AuthorRepository extends CrudRepository<Author,Integer> {
}
```

OUTPUT


GET
http://localhost:8080/authors/create
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settin
Query Params
KEY
VALUE
Key
Value
Body
Cookies
Headers (5)
Test Results
Pretty
Raw
Preview
Visualize
Text
1
Author Added

```
mysql> select * from author;
Empty set (0.00 sec)

mysql> select * from subject;
Empty set (0.00 sec)

mysql> select * from address;
ERROR 1146 (42S02): Table 'SpringDataJPA3.address' doesn't exist
mysql> select * from author;
+----+-----------+-------+--------------+---------+
| id | location  | state | streetnumber | name    |
+----+-----------+-------+--------------+---------+
|  1 | Pitampura | Delhi | AL76         | Aayushi |
+----+-----------+-------+--------------+---------+
1 row in set (0.00 sec)

mysql> select * from subject;
+----+-------------+----------+
| id | subjectname | authorid |
+----+-------------+----------+
|  2 | Java        |        1 |
|  3 | Python      |        1 |
|  4 | Groovy      |        1 |
+----+-------------+----------+
3 rows in set (0.00 sec)
```

5. **Create an Entity book with an instance variable bookName.**
6. **Implement One to One mapping between Author and Book.**

CODE

Author.java

```
@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String name;
  @Embedded
  private Address address;
  @OneToMany(mappedBy = "author",cascade = CascadeType.ALL)  //all changes you do on
author , do them in subject table as well.
  private List<Subject> subjects;

  @OneToOne(mappedBy = "author",cascade = CascadeType.ALL)
  private Book book;

  public Book getBook() {
     return book;
  }

  public void setBook(Book book) {
     this.book = book;
  }

  public List<Subject> getSubjects() {
     return subjects;
  }

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

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

```
  public void setId(Integer 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 void addSubject(Subject subject)
  {
      if(subject != null)
      {
          if(subjects == null)
              subjects = new ArrayList<>();
      }
      subject.setAuthor(this);
      subjects.add(subject);
  }
}
```

Book.java

```
@Entity
public class Book {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String bookname;

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

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

  public Integer getId() {
     return id;
  }

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

  public String getBookname() {
     return bookname;
  }

  public void setBookname(String bookname) {
     this.bookname = bookname;
  }

  public Author getAuthor() {
     return author;
  }

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

AuthorService.java

```
public void createAuthorBook()
{
  Address address = new Address();
  address.setStreetnumber("AL76");
  address.setLocation("Pitampura");
  address.setState("Delhi");

  Subject subject1 = new Subject();
  subject1.setSubjectname("Java");
  Subject subject2 = new Subject();
  subject2.setSubjectname("Python");
  Subject subject3 = new Subject();
```

```
  subject3.setSubjectname("Groovy");


  Book book = new Book();
  book.setBookname("Java Advanced Guide");

  Author author = new Author();
  author.setName("Aayushi");
  author.setAddress(address);

  author.addSubject(subject1);
  author.addSubject(subject2);
  author.addSubject(subject3);
  author.setBook(book);

  book.setAuthor(author);
  authorRepository.save(author);
}
```

AuthorController.java

```
@GetMapping("/create-book")
public String createAuthorBook()
{
  authorService.createAuthorBook();
  return "Records Added to Author & Books";
}
```

AuthorRepository.java

```
@Repository
public interface AuthorRepository extends CrudRepository<Author,Integer> {
}
```

OUTPUT

```
mysql> show tables;
+--------------------------+
| Tables_in_SpringDataJPA3 |
+--------------------------+
| author                   |
| book                     |
| hibernate_sequence       |
| subject                  |
+--------------------------+
4 rows in set (0.00 sec)

mysql> desc author;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| id           | int(11)      | NO   | PRI | NULL    |       |
| location     | varchar(255) | YES  |     | NULL    |       |
| state        | varchar(255) | YES  |     | NULL    |       |
| streetnumber | varchar(255) | YES  |     | NULL    |       |
| name         | varchar(255) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> desc book;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | int(11)      | NO   | PRI | NULL    |       |
| bookname  | varchar(255) | YES  |     | NULL    |       |
| author_id | int(11)      | YES  | MUL | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> desc subject;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id          | int(11)      | NO   | PRI | NULL    |       |
| subjectname | varchar(255) | YES  |     | NULL    |       |
| authorid    | int(11)      | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

GET http://localhost:8080/authors/create-book

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

Query Params

KEY	VALUE
Key	Value

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | Text

```
1   Records Added to Author & Books
```

```
mysql> select * from author;
+----+-----------+-------+--------------+---------+
| id | location  | state | streetnumber | name    |
+----+-----------+-------+--------------+---------+
|  1 | Pitampura | Delhi | AL76         | Aayushi |
+----+-----------+-------+--------------+---------+
1 row in set (0.00 sec)

mysql> select * from subject;
+----+-------------+----------+
| id | subjectname | authorid |
+----+-------------+----------+
|  2 | Java        |        1 |
|  3 | Python      |        1 |
|  4 | Groovy      |        1 |
+----+-------------+----------+
3 rows in set (0.00 sec)

mysql> select * from book;
+----+---------------------+-----------+
| id | bookname            | author_id |
+----+---------------------+-----------+
|  5 | Java Advanced Guide |         1 |
+----+---------------------+-----------+
1 row in set (0.00 sec)
```

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

A. **<u>Unidirectional</u>**

<u>Author.java</u>

```
@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String name;
  @Embedded
  private Address address;
  @OneToMany(mappedBy = "author",cascade = CascadeType.ALL)  //all changes you do on
author , do them in subject table as well.
  private List<Subject> subjects;

@OneToMany(cascade = CascadeType.ALL)
@JoinColumn(name = "book_id")
private List<Book> books;

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

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

  public List<Subject> getSubjects() {
     return subjects;
  }

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

  public Integer getId() {
     return id;
  }

  public void setId(Integer 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 void addSubject(Subject subject)
    {
        if(subject != null)
        {
            if(subjects == null)
                subjects = new ArrayList<>();
        }
        subject.setAuthor(this);
        subjects.add(subject);
    }
}
```

Book.java

```
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    private String bookname;
    public Integer getId() {
        return id;
    }
```

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

    public String getBookname() {
        return bookname;
    }

    public void setBookname(String bookname) {
        this.bookname = bookname;
    }

}
```

AuthorService.java

```
public void createAuthorBookManyToOne()
{
    Address address = new Address();
    address.setStreetnumber("AL76");
    address.setLocation("Pitampura");
    address.setState("Delhi");

    Subject subject1 = new Subject();
    subject1.setSubjectname("Java");
    Subject subject2 = new Subject();
    subject2.setSubjectname("Python");
    Subject subject3 = new Subject();
    subject3.setSubjectname("Groovy");


    List<Book> books = new ArrayList<Book>();
    Book book1 = new Book();
    book1.setBookname("Book1");
    Book book2 = new Book();
    book2.setBookname("Book2");
    Book book3 = new Book();
    book3.setBookname("Book3");
    books.add(book1);books.add(book2);books.add(book3);

    Author author = new Author();
```

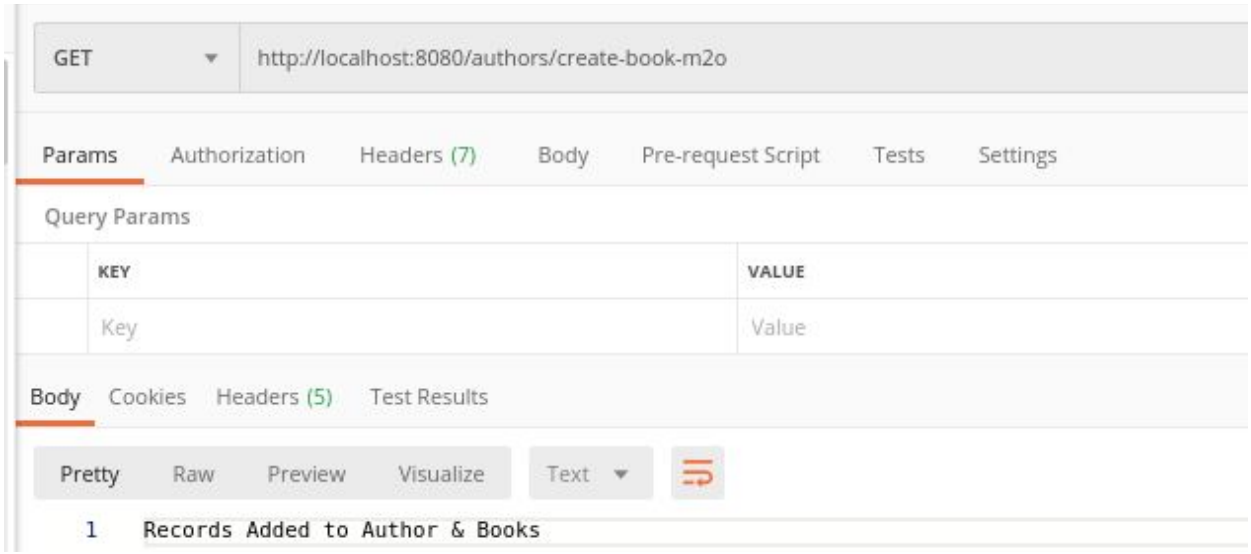
```
  author.setName("Aayushi");
  author.setAddress(address);
  author.setBooks(books);

  author.addSubject(subject1);
  author.addSubject(subject2);
  author.addSubject(subject3);

  authorRepository.save(author);
}
```

AuthorController.java

```
@GetMapping("/create-book-m2o")
private String createAuthorBookManyToOne()
{
  authorService.createAuthorBookManyToOne();
  return "Records Added to Author & Books";
}
```

OUTPUT

```
mysql> show tables;
+--------------------------+
| Tables_in_SpringDataJPA3 |
+--------------------------+
| author                   |
| book                     |
| hibernate_sequence       |
| subject                  |
+--------------------------+
4 rows in set (0.00 sec)

mysql> desc author;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| id           | int(11)      | NO   | PRI | NULL    |       |
| location     | varchar(255) | YES  |     | NULL    |       |
| state        | varchar(255) | YES  |     | NULL    |       |
| streetnumber | varchar(255) | YES  |     | NULL    |       |
| name         | varchar(255) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> desc book;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int(11)      | NO   | PRI | NULL    |       |
| bookname | varchar(255) | YES  |     | NULL    |       |
| book_id  | int(11)      | YES  | MUL | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```



```
mysql> select * from author;
+----+-----------+-------+--------------+---------+
| id | location  | state | streetnumber | name    |
+----+-----------+-------+--------------+---------+
|  1 | Pitampura | Delhi | AL76         | Aayushi |
+----+-----------+-------+--------------+---------+
1 row in set (0.01 sec)

mysql> select * from book;
+----+----------+---------+
| id | bookname | book_id |
+----+----------+---------+
|  2 | Book1    |       1 |
|  3 | Book2    |       1 |
|  4 | Book3    |       1 |
+----+----------+---------+
3 rows in set (0.00 sec)
```

B. <u>BiDirectional</u>

<u>Author.java</u>

```
@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String name;
  @Embedded
  private Address address;
  @OneToMany(mappedBy = "author",cascade = CascadeType.ALL)  //all changes you do on
author , do them in subject table as well.
  private List<Subject> subjects;

  @OneToMany(mappedBy = "author",cascade = CascadeType.ALL)  //all changes you
do on author , do them in subject table as well.
  private List<Book> books;

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

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

```
}

public List<Subject> getSubjects() {
    return subjects;
}

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

public Integer getId() {
    return id;
}

public void setId(Integer 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 void addSubject(Subject subject)
{
    if(subject != null)
    {
        if(subjects == null)
            subjects = new ArrayList<>();
    }
    subject.setAuthor(this);
    subjects.add(subject);
```

```
  }
}
```

Book.java

```
@Entity
public class Book {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String bookname;

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

  public Author getAuthor() {
     return author;
  }

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

  public Integer getId() {
     return id;
  }

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

  public String getBookname() {
     return bookname;
  }

  public void setBookname(String bookname) {
     this.bookname = bookname;
  }
}
```

AuthorService.java

```
public void createAuthorBookManyToOne()
```

```
{
   Address address = new Address();
   address.setStreetnumber("AL76");
   address.setLocation("Pitampura");
   address.setState("Delhi");

   Subject subject1 = new Subject();
   subject1.setSubjectname("Java");
   Subject subject2 = new Subject();
   subject2.setSubjectname("Python");
   Subject subject3 = new Subject();
   subject3.setSubjectname("Groovy");


   List<Book> books = new ArrayList<Book>();
   Book book1 = new Book();
   book1.setBookname("Book1");
   Book book2 = new Book();
   book2.setBookname("Book2");
   Book book3 = new Book();
   book3.setBookname("Book3");
   books.add(book1);books.add(book2);books.add(book3);

   Author author = new Author();
   author.setName("Aayushi");
   author.setAddress(address);
   author.setBooks(books);

   author.addSubject(subject1);
   author.addSubject(subject2);
   author.addSubject(subject3);

   book1.setAuthor(author);
   book2.setAuthor(author);
   book3.setAuthor(author);
   authorRepository.save(author);
}
```

AuthorController.java

```
@GetMapping("/create-book-m2o")
private String createAuthorBookManyToOne()
```

```
{
  authorService.createAuthorBookManyToOne();
  return "Records Added to Author & Books";
}
```

OUTPUT

```
mysql> show tables;
Empty set (0.00 sec)

mysql> show tables;
+---------------------------+
| Tables_in_SpringDataJPA3 |
+---------------------------+
| author                    |
| book                      |
| hibernate_sequence        |
| subject                   |
+---------------------------+
4 rows in set (0.00 sec)
```

```
mysql> desc author;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| id           | int(11)      | NO   | PRI | NULL    |       |
| location     | varchar(255) | YES  |     | NULL    |       |
| state        | varchar(255) | YES  |     | NULL    |       |
| streetnumber | varchar(255) | YES  |     | NULL    |       |
| name         | varchar(255) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> desc book;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int(11)      | NO   | PRI | NULL    |       |
| bookname | varchar(255) | YES  |     | NULL    |       |
| authorid | int(11)      | YES  | MUL | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> desc subject;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id          | int(11)      | NO   | PRI | NULL    |       |
| subjectname | varchar(255) | YES  |     | NULL    |       |
| authorid    | int(11)      | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

GET http://localhost:8080/authors/create-book-m2o

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

Query Params

KEY	VALUE
Key	Value

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | Text

```
1  Records Added to Author & Books
```

```
mysql> select * from author;
+----+-----------+-------+--------------+---------+
| id | location  | state | streetnumber | name    |
+----+-----------+-------+--------------+---------+
|  1 | Pitampura | Delhi | AL76         | Aayushi |
+----+-----------+-------+--------------+---------+
1 row in set (0.00 sec)

mysql> select * from book;
+----+----------+----------+
| id | bookname | authorid |
+----+----------+----------+
|  2 | Book1    |        1 |
|  3 | Book2    |        1 |
|  4 | Book3    |        1 |
+----+----------+----------+
3 rows in set (0.00 sec)

mysql> select * from subject;
+----+-------------+----------+
| id | subjectname | authorid |
+----+-------------+----------+
|  5 | Java        |        1 |
|  6 | Python      |        1 |
|  7 | Groovy      |        1 |
+----+-------------+----------+
3 rows in set (0.00 sec)
```

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

Author.java

```
@Entity
public class Author {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String name;
  @Embedded
  private Address address;

  @ManyToMany(cascade = CascadeType.ALL)
  @JoinTable(name = "author_books",
        joinColumns = @JoinColumn(name = "author_id" ,referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(name = "book_id",referencedColumnName =
"id"))
  private List<Book> books;

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

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


  public Integer getId() {
     return id;
  }

  public void setId(Integer 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;
    }
}
```

Book.java

```
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;
    private String bookname;

    @ManyToMany(mappedBy = "books")
    private List<Author> authors;
    public List<Author> getAuthors() {
        return authors;
    }

    public void setAuthors(List<Author> authors) {
        this.authors = authors;
    }

    public Integer getId() {
        return id;
    }

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

    public String getBookname() {
        return bookname;
    }
```

```
  public void setBookname(String bookname) {
      this.bookname = bookname;
  }
}
```

AuthorService.java

```
public void createAuthorBookManyToMany()
{
  Address address = new Address();
  address.setStreetnumber("AL76");
  address.setLocation("Pitampura");
  address.setState("Delhi");

  List<Book> books = new ArrayList<Book>();
  Book book1 = new Book();
  book1.setBookname("Book1");
  Book book2 = new Book();
  book2.setBookname("Book2");
  Book book3 = new Book();
  book3.setBookname("Book3");
  books.add(book1);books.add(book2);books.add(book3);

  Author author = new Author();
  author.setName("Aayushi");
  author.setAddress(address);
  author.setBooks(books);

  authorRepository.save(author);

}
```

AuthorController.java

```
@GetMapping("/create-book-m2m")
public String createAuthorBookManyToMany()
{
  authorService.createAuthorBookManyToMany();
  return "Records Added to Author & Books";
}
```

OUTPUT

```
mysql> show tables;
+---------------------------+
| Tables_in_SpringDataJPA3 |
+---------------------------+
| author                    |
| author_books              |
| book                      |
| hibernate_sequence        |
+---------------------------+
4 rows in set (0.00 sec)
```

```
mysql> desc author;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| id           | int(11)      | NO   | PRI | NULL    |       |
| location     | varchar(255) | YES  |     | NULL    |       |
| state        | varchar(255) | YES  |     | NULL    |       |
| streetnumber | varchar(255) | YES  |     | NULL    |       |
| name         | varchar(255) | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> desc book;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int(11)      | NO   | PRI | NULL    |       |
| bookname | varchar(255) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc author_books;
+-----------+---------+------+-----+---------+-------+
| Field     | Type    | Null | Key | Default | Extra |
+-----------+---------+------+-----+---------+-------+
| author_id | int(11) | NO   | MUL | NULL    |       |
| book_id   | int(11) | NO   | MUL | NULL    |       |
+-----------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> select * from author_books;
+-----------+---------+
| author_id | book_id |
+-----------+---------+
|         1 |       2 |
|         1 |       3 |
|         1 |       4 |
+-----------+---------+
3 rows in set (0.00 sec)

mysql> select * from author;
+----+-----------+-------+--------------+---------+
| id | location  | state | streetnumber | name    |
+----+-----------+-------+--------------+---------+
|  1 | Pitampura | Delhi | AL76         | Aayushi |
+----+-----------+-------+--------------+---------+
1 row in set (0.00 sec)

mysql> select * from book;
+----+----------+
| id | bookname |
+----+----------+
|  2 | Book1    |
|  3 | Book2    |
|  4 | Book3    |
+----+----------+
3 rows in set (0.00 sec)
```

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

A. <u>LEVEL 1 CACHING: The FindById is executed only once.</u>

<u>AuthorService.java</u>

```
public Optional<Author> testCaching()
{
  Optional<Author> author = authorRepository.findById(1);

  authorRepository.findById(1);

  authorRepository.findById(1);
  return author;
}
```

AuthorController.java

```
@GetMapping("/caching")
public void testCaching()
{
  authorService.testCaching();
}
```

OUTPUT

```
AuthorController › testCaching()
Run:  SpringDataJPA3 [SpringDataJpa3A...
2020-03-19 12:22:27.498  INFO 31663 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat sta
2020-03-19 12:22:27.500  INFO 31663 --- [           main] c.A.S.SpringDataJpa3Application          : Started Sp
2020-03-19 12:22:32.015  INFO 31663 --- [nio-8080-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializi
2020-03-19 12:22:32.015  INFO 31663 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet        : Initializi
2020-03-19 12:22:32.021  INFO 31663 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet        : Completed
Hibernate: select author0_.id as id1_0_0_, author0_.location as location2_0_0_, author0_.state as state3_0_0_,
```

B. **EVICT METHOD: the method on the session object that can be used to remove an object from the cache.**

AuthorSession.java

```
public Optional<Author> testCaching()
{
   Session session = entityManager.unwrap(Session.class);
   Optional<Author> author = authorRepository.findById(1);

   authorRepository.findById(1);
   session.evict(author.get());
   authorRepository.findById(1);
   return author;
}
```

AuthorController.java

```
@GetMapping("/caching")
public void testCaching()
{
```

```
  authorService.testCaching();
}
```

OUTPUT

```
Run:  SpringDataJPA3 [SpringDataJpa3A...
2020-03-19 12:35:36.186  WARN 32391 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.
2020-03-19 12:35:36.312  INFO 32391 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initial
2020-03-19 12:35:36.510  INFO 32391 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat
2020-03-19 12:35:36.512  INFO 32391 --- [           main] c.A.S.SpringDataJpa3Application          : Started
2020-03-19 12:35:39.429  INFO 32391 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initial
2020-03-19 12:35:39.429  INFO 32391 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initial
2020-03-19 12:35:39.436  INFO 32391 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Complet
Hibernate: select author0_.id as id1_0_0_, author0_.location as location2_0_0_, author0_.state as state3_0_0
Hibernate: select author0_.id as id1_0_0_, author0_.location as location2_0_0_, author0_.state as state3_0_0
```

10. What does @transactional annotation do?

With programmatic transactions, transaction management code needs to be explicitly written so as to commit when everything is successful and rolling back if anything goes wrong. The transaction management code is tightly bound to the business logic in this case.

A. WITHOUT TRANSACTION

BankAccount.java

```
@Entity
@Table(name = "bankaccount")
public class BankAccount {
  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Integer id;
  private String name;
  private Integer balance;

  public Integer getId() {
     return id;
  }

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

  public String getName() {
```

```
        return name;
    }

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

    public Integer getBalance() {
        return balance;
    }

    public void setBalance(Integer balance) {
        this.balance = balance;
    }
}
```

BankAccountService.java

```
public List<BankAccount> bankTransfer(Integer amount)
{
    BankAccount bankAccount1 = ((bankAccountRepository.findById(1)).get());
    bankAccount1.setBalance(bankAccount1.getBalance()-amount);
    bankAccountRepository.save(bankAccount1);

    if(true)
    {
        throw new RuntimeException();
    }

    BankAccount bankAccount2 = ((bankAccountRepository.findById(2)).get());
    bankAccount2.setBalance(bankAccount2.getBalance()+amount);
    bankAccountRepository.save(bankAccount2);
    List<BankAccount> bankAccountList = (List<BankAccount>) bankAccountRepository.findAll();

    return bankAccountList;
}
```

BankAccountController.java

```
@RestController
@RequestMapping("/bank")
public class BankAccountController {
```

```
  @Autowired
  BankAccountService bankAccountService;

  @GetMapping("/tranfer/{amount}")
  public List<BankAccount> bankTranfer(@PathVariable Integer amount)
  {
     List<BankAccount> bankAccountList = bankAccountService.bankTransfer(amount);
     return bankAccountList;
  }
}
```

<u>OUTPUT : The balance is subtracted from Aayushi's account but Not Added to Pragya's account.</u>

```
mysql> select * from bankaccount;
+----+---------+---------+
| id | name    | balance |
+----+---------+---------+
|  1 | Aayushi |    4500 |
|  2 | Pragya  |    3500 |
+----+---------+---------+
2 rows in set (0.01 sec)

mysql> select * from bankaccount;
+----+---------+---------+
| id | name    | balance |
+----+---------+---------+
|  1 | Aayushi |    4000 |
|  2 | Pragya  |    3500 |
+----+---------+---------+
2 rows in set (0.00 sec)
```

B. <u>WITH TRANSACTION</u>

<u>BankAccountService.java</u>

```
@Transactional
public List<BankAccount> bankTransfer(Integer amount)
{
  BankAccount bankAccount1 = ((bankAccountRepository.findById(1)).get());
```

```
    bankAccount1.setBalance(bankAccount1.getBalance()-amount);
    bankAccountRepository.save(bankAccount1);

    if(true)
    {
        throw new RuntimeException();
    }

    BankAccount bankAccount2 = ((bankAccountRepository.findById(2)).get());
    bankAccount2.setBalance(bankAccount2.getBalance()+amount);
    bankAccountRepository.save(bankAccount2);
    List<BankAccount> bankAccountList = (List<BankAccount>) bankAccountRepository.findAll();

    return bankAccountList;
}
```

<u>OUTPUT : The balance is not changed from both the accounts.</u>

```
mysql> select * from bankaccount;
+----+---------+---------+
| id | name    | balance |
+----+---------+---------+
|  1 | Aayushi |    4000 |
|  2 | Pragya  |    3500 |
+----+---------+---------+
2 rows in set (0.00 sec)

mysql> select * from bankaccount;
+----+---------+---------+
| id | name    | balance |
+----+---------+---------+
|  1 | Aayushi |    4000 |
|  2 | Pragya  |    3500 |
+----+---------+---------+
2 rows in set (0.00 sec)
```