One-to-One mapping relationship using JPA and Spring Boot

We will create two JPA entities: the Book and the Story. The Book and the Story entities have one to one relationship, which means that the Book entity has a Story entity, and the Story entity also contains a Book entity.

Because we will be using Spring Boot Data JPA, there is no need to create database tables manually. The framework will create all database tables based on how we annotate our JPA entities.

After running this example, you will be able to save the Book and the Story object in a MySQL database by sending an HTTP request using the Postman HTTP client.

Default Fetch types

To implement the one-to-one relationship, we will use a special annotation called @OneToOne. This and other similar annotations have a default fetch type. I feel it is useful to mention their default values here.

@OneToOne – The default fetch type is EAGER.
@OneToMany – The default fetch type is LAZY.
@ManyToOne – The default fetch type is EAGER.
@ManyToMany – The default fetch type is LAZY.

One-to-One Mapping in Hibernate/JPA Spring Boot

Prerequisites:

- JDK 1.8
- Eclipse
- Maven
- MySQL
- Postman HTTP Client

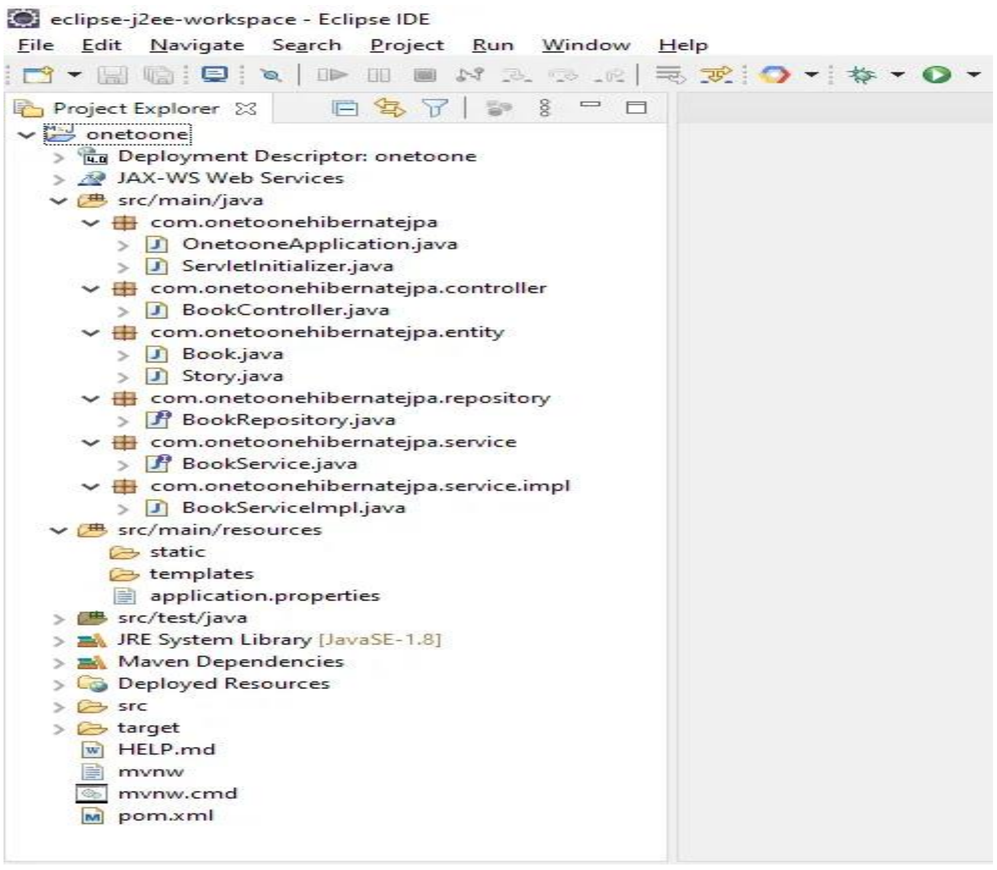
Open eclipse and create a maven project; replace the pom.xml with the code below.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.5.3</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.onetoonehibernatejpa</groupId>
<artifactId>onetoone</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>
<name>onetoone</name>
<description>Demo project for Spring Boot</description>
<properties>
<java.version>1.8</java.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-devtools</artifactId>
<scope>runtime</scope>
<optional>true</optional>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-tomcat</artifactId>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
```

```
</dependency>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<scope>runtime</scope>
</dependency>
</dependencies>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
</project>
```

Let maven download all necessary jars. Once it is done, you will be able to see the maven dependency folder that contains different jar files.

Now you can start writing our controller classes, ServiceImpl and Repository. The directory structure of the application looks as below.


eclipse-j2ee-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer

```
onetoone
  > Deployment Descriptor: onetoone
  > JAX-WS Web Services
  v src/main/java
    v com.onetoonehibernatejpa
      > OnetooneApplication.java
      > ServletInitializer.java
    v com.onetoonehibernatejpa.controller
      > BookController.java
    v com.onetoonehibernatejpa.entity
      > Book.java
      > Story.java
    v com.onetoonehibernatejpa.repository
      > BookRepository.java
    v com.onetoonehibernatejpa.service
      > BookService.java
    v com.onetoonehibernatejpa.service.impl
      > BookServiceImpl.java
  v src/main/resources
      static
      templates
      application.properties
  > src/test/java
  > JRE System Library [JavaSE-1.8]
  > Maven Dependencies
  > Deployed Resources
  > src
  > target
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
```

Define JPA Entities: Book and Story

Book.java

```
package com.onetoonehibernatejpa.entity;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import com.fasterxml.jackson.annotation.JsonManagedReference;
@Entity
@Table(name = "book")
public class Book {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int bookId;
@Column(name = "book_name")
private String bookName;
@OneToOne(cascade = CascadeType.ALL, mappedBy = "book")
@JsonManagedReference
private Story story;
public Story getStory() {
return story;
}
public void setStory(Story story) {
this.story = story;
}
public int getBookId() {
return bookId;
}
public void setBookId(int bookId) {
this.bookId = bookId;
}
public String getBookName() {
return bookName;
```

```java
}
public void setBookName(String bookName) {
this.bookName = bookName;
}
}
```

Story.java

```java
package com.onetoonehibernatejpa.entity;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import com.fasterxml.jackson.annotation.JsonBackReference;
@Entity
@Table(name = "story")
public class Story {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int storyId;
@Column(name = "story_name")
private String storyName;
@OneToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "book_id")
@JsonBackReference
private Book book;
public int getStoryId() {
return storyId;
}
public void setStoryId(int storyId) {
this.storyId = storyId;
}
public String getStoryName() {
return storyName;
}
public void setStoryName(String storyName) {
this.storyName = storyName;
}
public Book getBook() {
return book;
}
public void setBook(Book book) {
this.book = book;
}
}
```

Define the Repository Interface Extending JPARepository.

BookRepository.java

```
package com.onetoonehibernatejpa.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.onetoonehibernatejpa.entity.Book;
@Repository
public interface BookRepository extends JpaRepository<Book, String> {
public Book findByBookId(int bookId);
}
```

Define service interface, i.e. BookService.java

```
package com.onetoonehibernatejpa.service;
import org.springframework.stereotype.Component;
import com.onetoonehibernatejpa.entity.Book;
@Component
public interface BookService {
public Book saveBook(Book book);
public Book findByBookId(int bookId);
}
```

Define the Service Implementation Class

BookServiceImpl.java

```
package com.onetoonehibernatejpa.service.impl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.onetoonehibernatejpa.entity.Book;
import com.onetoonehibernatejpa.entity.Story;
import com.onetoonehibernatejpa.repository.BookRepository;
import com.onetoonehibernatejpa.service.BookService;
@Service
public class BookServiceImpl implements BookService {
@Autowired
private BookRepository bookRepository;
public Book saveBook(Book book) {
Story story = book.getStory();
story.setBook(book);
book = bookRepository.save(book);
return book;
}
public Book findByBookId(int bookId) {
Book book = bookRepository.findByBookId(bookId);
return book;
}
}
```

Define the Controller class

BookController.java

```
package com.onetoonehibernatejpa.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;
import com.onetoonehibernatejpa.entity.Book;
import com.onetoonehibernatejpa.service.BookService;
@RestController
@RequestMapping(value = "/book")
public class BookController {
@Autowired
private BookService bookService;
@RequestMapping(value = "/savebook", method = RequestMethod.POST)
@ResponseBody
public Book saveBook(@RequestBody Book book) {
Book bookResponse = bookService.saveBook(book);
return bookResponse;
}
@RequestMapping(value = "/{bookId}", method = RequestMethod.GET)
@ResponseBody
public Book getBookDetails(@PathVariable int bookId) {
Book bookResponse = bookService.findByBookId(bookId);
return bookResponse;
}
}
```

And finally, you have an application.properties file where you have database details.

application. properties

```
spring.jpa.hibernate.ddl-auto=create
spring.datasource.url=jdbc:mysql://localhost:3306/db_test
spring.datasource.username=test
spring.datasource.password=test@123
spring.datasource.driver-class-name =com.mysql.jdbc.Driver
#spring.jpa.show-sql: true
```

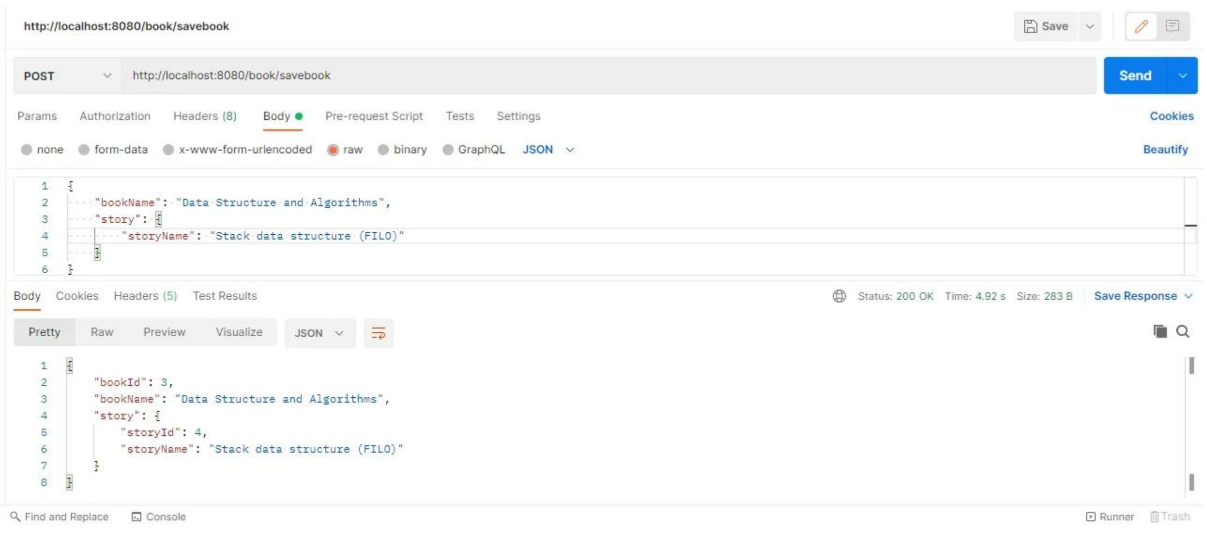
You are almost done. Just build the project once running the main method.

Let's deploy the application running *OnetooneApplication* class as a java application.

Now you will prepare JSON data and try to save it in the database.

After running the example, you will have a database entry like the one below.

Test with Post man



POST http://localhost:8080/book/savebook

Body (raw, JSON):

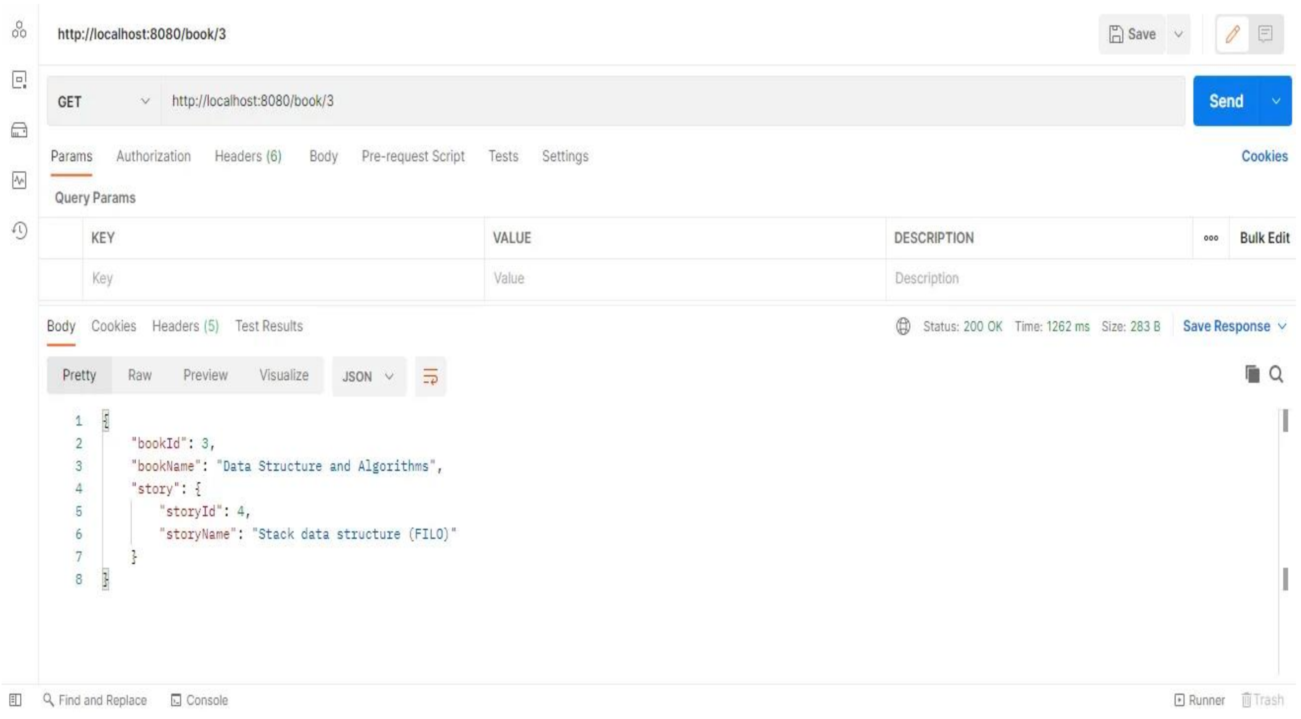
```
{
    "bookName": "Data Structure and Algorithms",
    "story": {
        "storyName": "Stack data structure (FILO)"
    }
}
```

Response:

```
{
    "bookId": 3,
    "bookName": "Data Structure and Algorithms",
    "story": {
        "storyId": 4,
        "storyName": "Stack data structure (FILO)"
    }
}
```

You should now have records in the book and the story table.
Notice *book_id* is the column in the story table is a foreign key. This is the primary key for the book table.

Test the get URL, i.e. http://localhost:8080/book/3



GET http://localhost:8080/book/3

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
{
    "bookId": 3,
    "bookName": "Data Structure and Algorithms",
    "story": {
        "storyId": 4,
        "storyName": "Stack data structure (FILO)"
    }
}
```

That’s all about One-to-One Mapping in Hibernate/JPA Spring Boot.