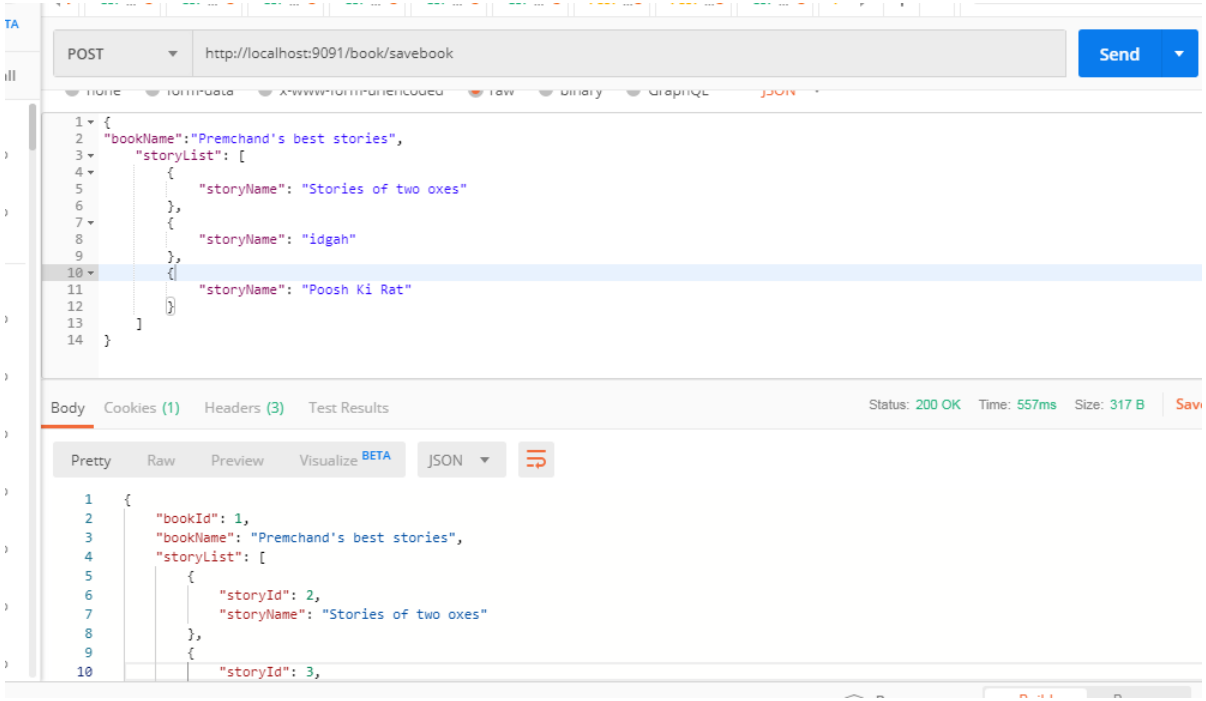
One To Many Mapping Annotation Example In Hibernate/JPA Using Spring Boot And Oracle

POSTED BY: RAKESH OCTOBER 27, 2019

In this tutorial, we will see One to Many Annotation Mapping Example in Hibernate/JPA using Spring Boot and Oracle. We are going to use maven, embedded tomcat, postman and oracle database. Here we will have some rest endpoint which will be used to save and retrieve data in the database.

We have two entity Book.java and Story.java which are mapped in One To Many relationships. In this example, we are assuming one Book can have multiple stories. After running the below example we will be able to save the Book and Story entity which is in One To Many relationships.

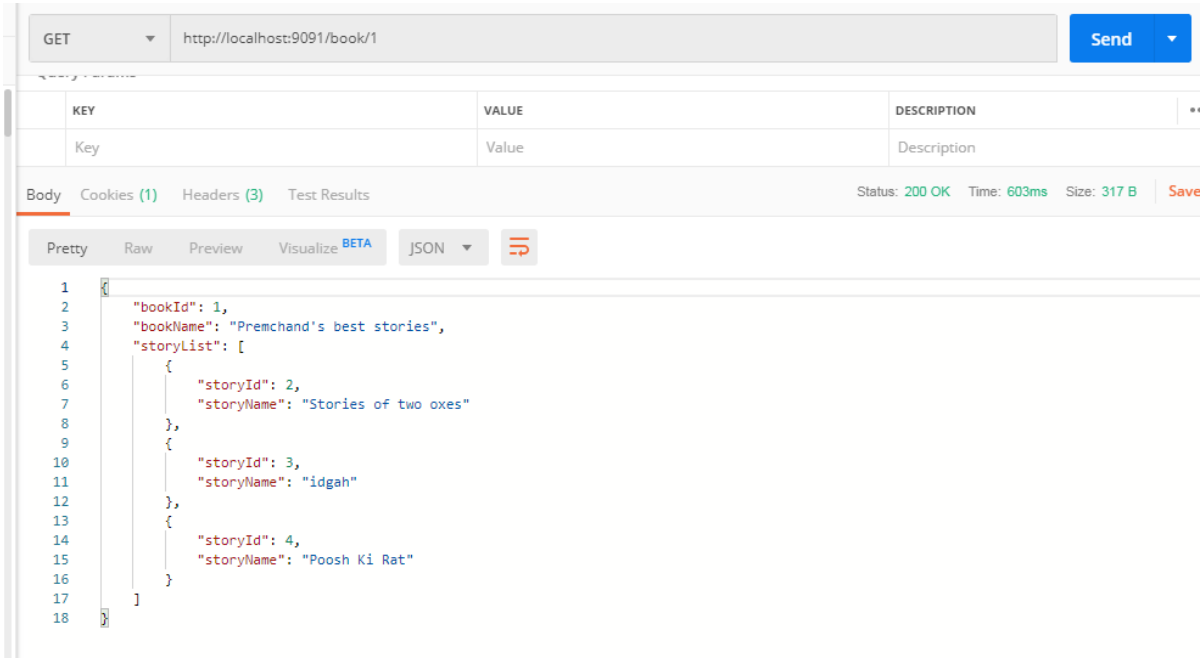
Save URL – http://localhost:9091/book/savebook



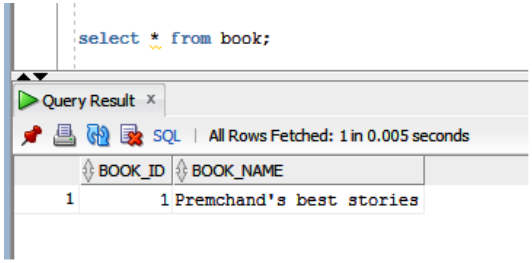
Get URL- http://localhost:9091/book/1

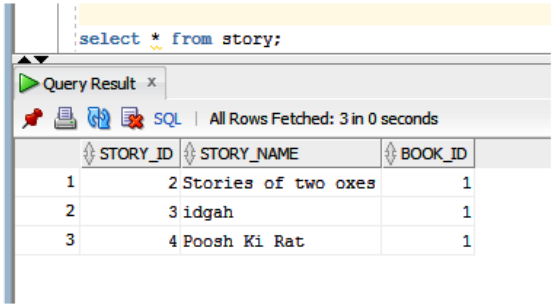
Spring Boot Examples

Spring restful web services example using spring boot.
How to get all loaded beans in Spring Boot application.
Spring Boot interceptor example.
Filter example in Spring Boot.
Content negotiation example using Spring Boot.
Deploy Spring boot war in JBoss EAP server.
Jboss 7 EPA datasource configuration using oracle and spring boot.
Deploy Spring Boot application on external Tomcat.
Deploy multiple war files in JBoss to different port.
Spring boot datasource configuration using tomcat



Database entry –





Recent Posts

JPA CascadeType PERSIST Example Using Spring Boot
@ElementCollection Example in Hibernate/JPA Using Spring Boot
JPA EntityManager CRUD example Using Spring Boot
JPA EntityManager remove() example Using Spring Boot
JPA EntityManager merge() example Using Spring Boot
Hibernate First Level Cache example using Spring Boot
JPA EntityManager persist() method Example
JPA EntityManager find() method
How to write custom method in repository in Spring Data JPA

Let’s see complete example One To Many Mapping Annotation Example In Hibernate/JPA Using Spring Boot And Oracle from scratch.

Before going ahead, let’s see some points.

- We are going to use two annotations @OneToMany and @JoinColumn for mapping.
- We will not create a table manually, let’s hibernate do this job.

Note – Default Fetch type in case of the below annotations.

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

We will use the spring boot library (will provide dependency in pom.xml) to make sure all necessary jar/library is easily available for our application. Spring boot will take care of all jars. Le’s see One To Many Mapping Annotation Example In Hibernate/JPA Using Spring Boot And Oracle from scratch.

Prerequisites –

How to create a custom repository in Spring Data JPA
Sorting And Pagination in Spring Data JPA
Spring Data JPA Interview Questions and Answers
Spring Data JPA @NamedNativeQuery
Spring Data JPA @NamedQuery
Spring Data JPA @Query Annotation
Spring Data JPA Named Parameters
Define multiple Rest API with the same name
Spring Data JPA JPQL and Native Query Example
Difference between Repository and CrudRepository
CrudRepository Methods Example
Difference between CrudRepository and JpaRepository in Spring Data JPA
Spring Data JPA Nested Property Query Method
Spring Data JPA Projection Example
Spring Data JPA Query Methods/Repository Methods
@Min And @Max Javax Validation Hibernate Example
Spring Data JPA StartingWith And EndingWith Example
Spring Data JPA Is and Equals Example
Spring Data JPA In and NotIn Example
Spring Data JPA between Example
Spring Data JPA And Or Example Using Spring Boot

- JDK 1.8
- Oracle 10g
- Eclipse
- maven
- postman

Open eclipse and create maven project, Don't forget to check 'Create a simple project (skip)'click on next. Fill all details(GroupId – onetomanyhibernatejpa, ArtifactId – onetomanyhibernatejpa and name – onetomanyhibernatejpa) and click on finish. Keep packaging as the jar.

Modify the pom.xml with the below code.

```
<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 http://mav
    <modelVersion>4.0.0</modelVersion>
    <groupId>onetoonehibernatejpa</groupId>
    <artifactId>onetoonehibernatejpa</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>onetomanyhibernatejpa</name>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.2.RELEASE</version>
    </parent>
    <dependencies>
        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <dependency>
        <groupId>com.oracle</groupId>
        <artifactId>ojdbc6</artifactId>
```

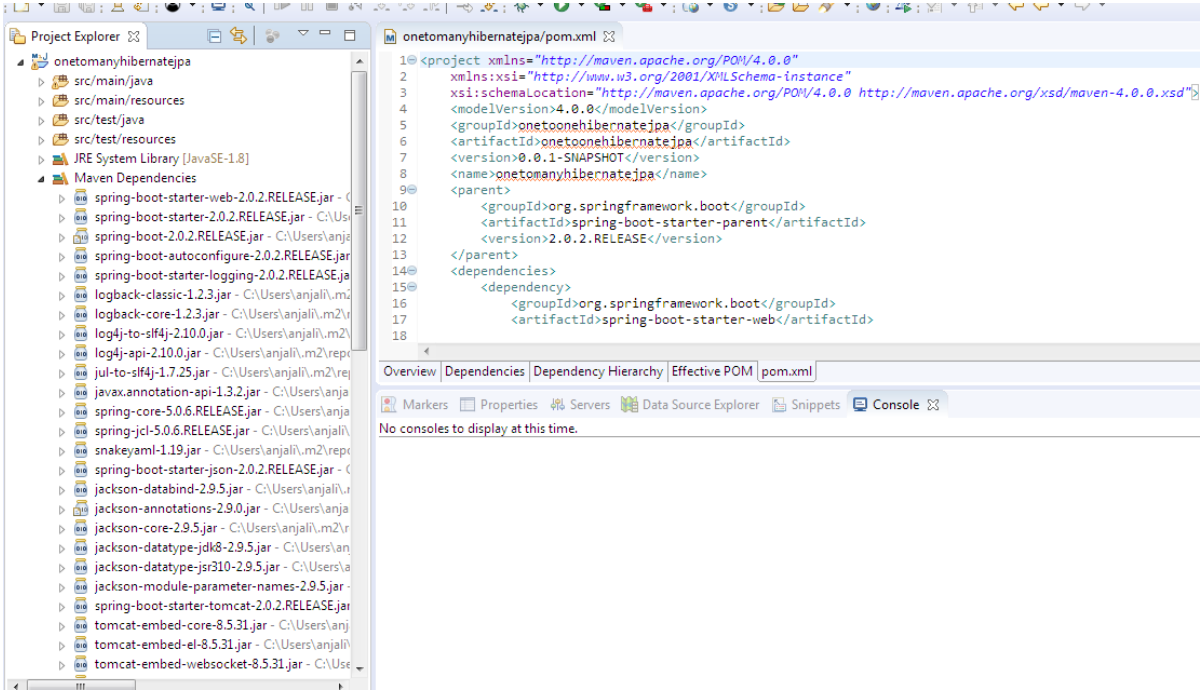
Spring Data JPA Not Example Using Spring Boot
Spring Data JPA contains ignorecase Example
Spring Data JPA Like and Containing Example
Spring Data JPA greater than Example
Spring Data JPA less than Example
Spring Data JPA IsNull Example Using Spring Boot
@Transactional timeout Using Spring Boot
Spring Data Case Insensitive Search Example
Spring Data findById() Vs getOne()
Spring Data JPA JpaRepository getOne()
Spring Data JPA delete() vs deleteInBatch()
Spring Data JPA deleteAll() Vs deleteAllInBatch()
Spring Data JPA JpaRepository deleteAllInBatch()
Spring Data JPA deleteInBatch() Example
Spring Data JPA JpaRepository saveAndFlush() Example
Spring Data JPA CrudRepository count() Example
Spring Data JPA CrudRepository delete() and deleteAll()
Spring Data JPA CrudRepository deleteById() Example
CrudRepository findAllById() Example Using Spring Boot
How to Sort ArrayList in Descending Order in Java

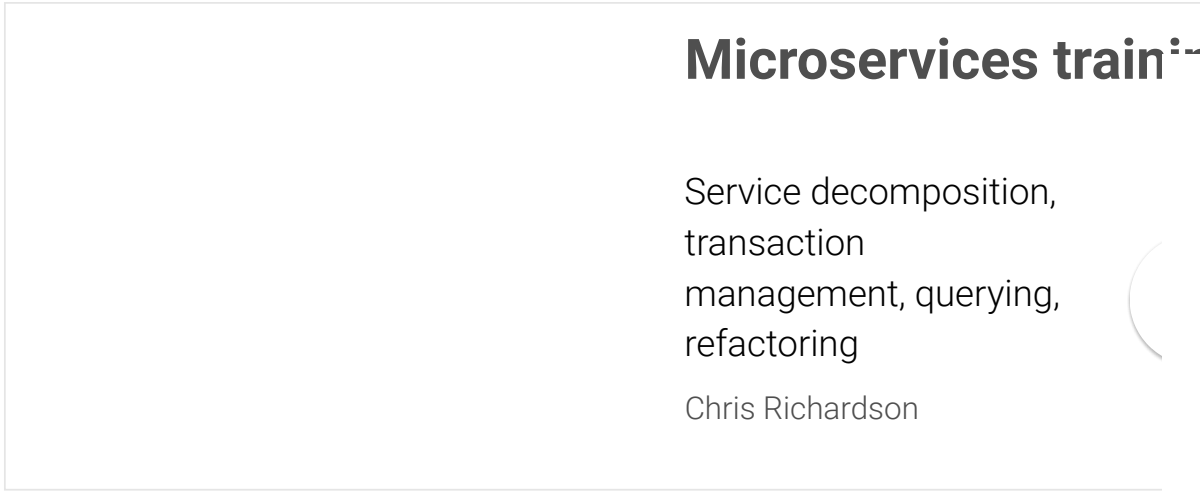
```
        <version>11.2.0.3</version>
        </dependency>

    </dependencies>

    <build>
        <finalName>${project.artifactId}</finalName>
        <plugins>

        <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
        <fork>true</fork>
        <executable>C:\Program Files\Java\jdk1.8.0_131\bin\javac.exe
        </configuration>
        </plugin>


        </plugins>
    </build>
</project>
```

Note – In pom.xml we have defined javac.exe path in configuration tag. You need to change accordingly i.e where you have installed JDK.

If you see any error for oracle dependency then follow **these** steps.

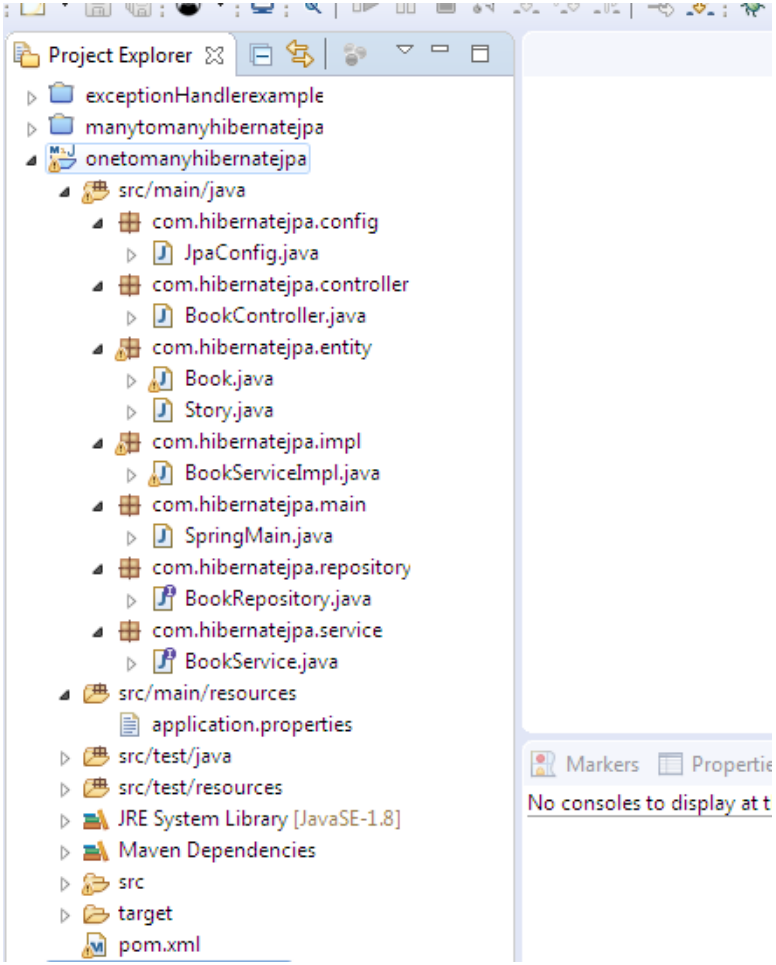
Let maven download all necessary jar. Once it is done we will able to see the maven dependency folder which contains different jar files.





We are good now. We can start writing our controller classes, ServiceImpl and Repository. The directory structure of the application looks as below.

@ControllerAdvice Global Error Handling Example in Spring Boot
@ExceptionHandler Example in Spring Boot
How To Load Sql Script On Application Startup Using H2 Database
One To Many Mapping Annotation Example In Hibernate/JPA Using Spring Boot And Oracle
Junit Test Cases For Exception Example In Java
One to One Bidirectional Mapping Example In Hibernate/JPA Using Spring Boot and Oracle
One to One Mapping in Hibernate/JPA using Spring Boot and Oracle
Synchronization In Java With Example
Difference between sleep() and wait() method in Java
Runnable interface in java
Daemon Thread In Java With Example
Thread Priority in Java with example
Loop detection in a linked list
Spring boot datasource configuration using tomcat
wait() notify() and notifyAll() method in Java
Thread sleep() method in Java
Thread join() method in Java
Thread yield() method in Java with Example
Thread getId() method in Java
Thread getName() method in java
Thread dumpStack() method in Java
Thread currentThread() method in Java
Thread checkAccess() method in Java
Thread activeCount() method in Java



Define entity class i.e Book.java and Story.java.

Book.java

```
1  package com.hibernatejpa.entity;
2
3  import java.util.ArrayList;
4  import java.util.List;
```

Thread life cycle in Java
Doubly Circular Linked List in Java
Circular Linked List in Java
Iterator for doubly linked list java
Deploy Spring Boot application on external Tomcat
Deploy multiple war files in JBoss to different port
How to deploy multiple war files in Jboss in same port
How run() method Works internally in Java
Thread class constructors and methods in Java
How to create a thread in Java
Multithreading in Java with example
Reverse a Doubly Linked list
Delete operation in doubly linked list
How HashMap works internally in Java
Doubly Linked List example in Java
Iterator for singly linked list java
Junit test for private methods reflection example
Accessing private class members using reflection
Get class members information using reflection
Reflection in Java
Intersection of two linked lists
Rotation of Linked List
Remove duplicate nodes from linked list
@Transactional noRollbackForClassName example using spring boot

```
5
6  import javax.persistence.CascadeType;
7  import javax.persistence.Column;
8  import javax.persistence.Entity;
9  import javax.persistence.FetchType;
10 import javax.persistence.GeneratedValue;
11 import javax.persistence.GenerationType;
12 import javax.persistence.Id;
13 import javax.persistence.JoinColumn;
14 import javax.persistence.OneToMany;
15 import javax.persistence.OneToOne;
16
17 import com.fasterxml.jackson.annotation.JsonManagedReference;
18
19 @Entity
20 public class Book {
21
22     @Id
23     @GeneratedValue(strategy = GenerationType.AUTO)
24     private int bookId;
25
26     @Column(name = "book_name")
27     private String bookName;
28
29     @OneToMany(cascade = CascadeType.ALL, fetch= FetchType.LAZY)
30     @JoinColumn(name = "book_id",referencedColumnName="bookId")
31     private List<Story> storyList = new ArrayList<>();
32
33     public int getBookId() {
34         return bookId;
35     }
36
37     public void setBookId(int bookId) {
38         this.bookId = bookId;
39     }
40
41     public String getBookName() {
42         return bookName;
43     }
44
45     public void setBookName(String bookName) {
46         this.bookName = bookName;
47     }
48
49     public List<Story> getStoryList() {
```

@Transactional rollbackForClassName example using spring boot
@Transactional readonly true example in spring boot
@Transactional noRollbackFor example using spring boot
@Transactional rollbackFor example using spring boot
Sorting of linked list in Java
Middle of a Singly Linked List
Reverse a Singly Linked List
Implementation of Index based Linked List
Delete operation in Linked list
Singly Linked List Implementation using generics in Java
Singly Linked List Implementation in Java
@Transactional REQUIRED vs REQUIRES_NEW example in spring boot
Spring Boot interceptor example
Filter example in Spring Boot
@ComponentScan example in spring boot
Content negotiation example using Spring Boot
@Configuration annotation example using spring boot
Jboss 7 EPA datasource configuration using oracle and spring boot
Spring security inMemoryAuthentication and authorization example using spring boot
Spring security default authorization example using spring boot

```
50          return storyList;
51      }
52
53      public void setStoryList(List<Story> storyList) {
54          this.storyList = storyList;
55      }
56
57
58
59  }
```

Story.java

```
1   package com.hibernatejpa.entity;
2
3   import javax.persistence.Column;
4   import javax.persistence.Entity;
5   import javax.persistence.GeneratedValue;
6   import javax.persistence.GenerationType;
7   import javax.persistence.Id;
8
9   @Entity
10  public class Story {
11
12      @Id
13      @GeneratedValue(strategy = GenerationType.AUTO)
14      private int storyId;
15
16      @Column(name = "story_name")
17      private String storyName;
18
19      public int getStoryId() {
20          return storyId;
21      }
22
23      public void setStoryId(int storyId) {
24          this.storyId = storyId;
25      }
26
27      public String getStoryName() {
28          return storyName;
29      }
30
31      public void setStoryName(String storyName) {
32          this.storyName = storyName;
```

@Component @Controller @Service and @Repository annotations example using spring boot

How to get all loaded beans in Spring Boot application

@SpringBootApplication annotation example in Spring Boot

@RequestHeader annotation example using Spring Boot

Deploy Spring boot war in JBOSS EAP server

How to create war file using maven

@PathVariable and @RequestParam annotations in Spring Boot

@RequestBody and @ResponseBody annotation example in Spring Boot

@RequestMapping annotation example In Spring Boot

@RestController and @Controller annotation example in Spring Boot

Fail fast and fail safe example in java

Difference between HashMap and Hashtable in java

Difference between Comparable and Comparator in java

Difference between Iterator and ListIterator in Java

Difference between Iterator and Enumeration in java

User defined class as key in HashMap in Java

instanceof operator in java

Spring batch task scheduling example using spring boot

```
33      }
34
35  }
```

Define the repository interface extending CrudRepository.

BookRepository.java

```
1  package com.hibernatejpa.repository;
2
3  import java.io.Serializable;
4
5  import org.springframework.data.repository.CrudRepository;
6  import org.springframework.stereotype.Repository;
7
8  import com.hibernatejpa.entity.Book;
9  @Repository
10 public interface BookRepository extends CrudRepository<Book,Seriali
11     public Book findByBookId(int bookId);
12 }
13
```

Define service interface i.e BookService.java

```
1  package com.hibernatejpa.service;
2
3  import org.springframework.stereotype.Component;
4
5  import com.hibernatejpa.entity.Book;
6
7  @Component
8  public interface BookService {
9      public Book saveBook(Book book);
10     public Book findByBookId(int bookId);
11 }
12
```

Define service implementation class.

BookServiceImpl.java

```
1  package com.hibernatejpa.impl;
```

Spring batch basic
Equals and hashcode contract in java
Spring transaction management example using spring boot
Spring transaction management basic
Difference between HashSet and HashMap in Java
Difference between HashSet and TreeSet in java
How HashSet works internally in java
Difference between List and Set in Java
Custom exception in java and its use
Difference between ArrayList and LinkedList in java
Program to count number of object created in java
Spring restful web services example using spring boot
Performance in case of HashMap and Hashtable
How get method of ArrayList work internally in java
ArrayList vs Vector in java

```
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.stereotype.Service;
8
9  import com.hibernatejpa.entity.Book;
10 import com.hibernatejpa.entity.Story;
11 import com.hibernatejpa.repository.BookRepository;
12 import com.hibernatejpa.service.BookService;
13
14 @Service("bookServiceImpl")
15 public class BookServiceImpl implements BookService {
16
17     @Autowired
18     private BookRepository bookRepository;
19
20     public Book saveBook(Book book) {
21
22         book = bookRepository.save(book);
23         return book;
24
25     }
26
27     public Book findByBookId(int bookId) {
28         Book book = bookRepository.findByBookId(bookId);
29         return book;
30     }
31 }
```

Note – See here more about @Component, @Controller, @Service and @Repository annotations **here**.

Define the controller class or endpoint.

BookController.java

```
1  package com.hibernatejpa.controller;
2
3  import org.springframework.beans.factory.annotation.Autowired;
4  import org.springframework.web.bind.annotation.PathVariable;
5  import org.springframework.web.bind.annotation.RequestBody;
6  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.hibernatejpa.entity.Book;
import com.hibernatejpa.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;
    }

}
```

Note – See more details about @Controller and RestController **here**.

Define the JpaConfig.java

```
package com.hibernatejpa.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.data.jpa.repository.config.EnableJpaRepos

@Configuration
@EnableJpaRepositories(basePackages = "com.hibernatejpa.repository")
public class JpaConfig {

}
```

```
11
```

Step 12 – Define the SpringMain.java

```
1  package com.hibernatejpa.main;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.boot.autoconfigure.domain.EntityScan;
6  import org.springframework.context.annotation.ComponentScan;
7
8  @SpringBootApplication
9  @ComponentScan(basePackages="com.hibernatejpa.*")
10 @EntityScan("com.hibernatejpa.*")
11 public class SpringMain {
12     public static void main(String[] args) {
13
14         SpringApplication.run(SpringMain.class, args);
15     }
16
17 }
18
```

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

application.properties

```
1  # Connection url for the database
2  spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE
3  spring.datasource.username=SYSTEM
4  spring.datasource.password=oracle2
5  spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
6  # Show or not log for each sql query
7  spring.jpa.show-sql = true
8
9
10 spring.jpa.hibernate.ddl-auto =create
11 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.Orac
12
13 server.port = 9091
```

We are almost done. Just build the project once running the main method. Open git bash or cmd and Run mvn clean install.

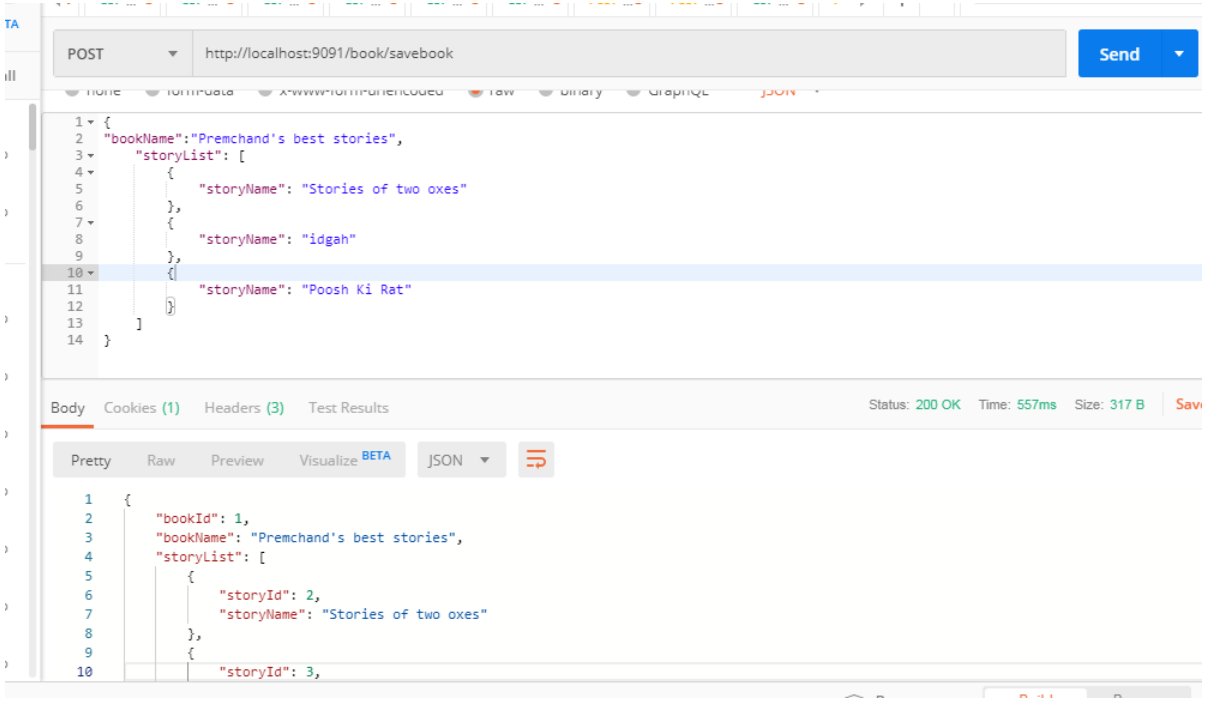
Let's deploy the application running SpringMain class as a java application.

Now we will prepare json data and will try save in database.

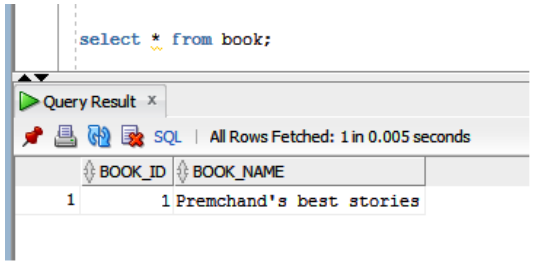
Sample request JSON data-

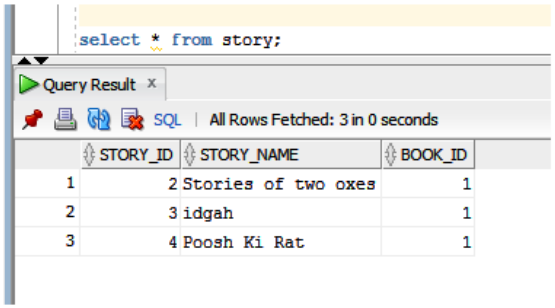
```
{
    "bookName": "Premchand's best stories",
    "storyList": [{
        "storyName": "Stories of two oxes"
        },
        {
        "storyName": "idgah"
        },
        {
        "storyName": "Poosh Ki Rat"
        }
    ]
}
```

Let's test the save url.



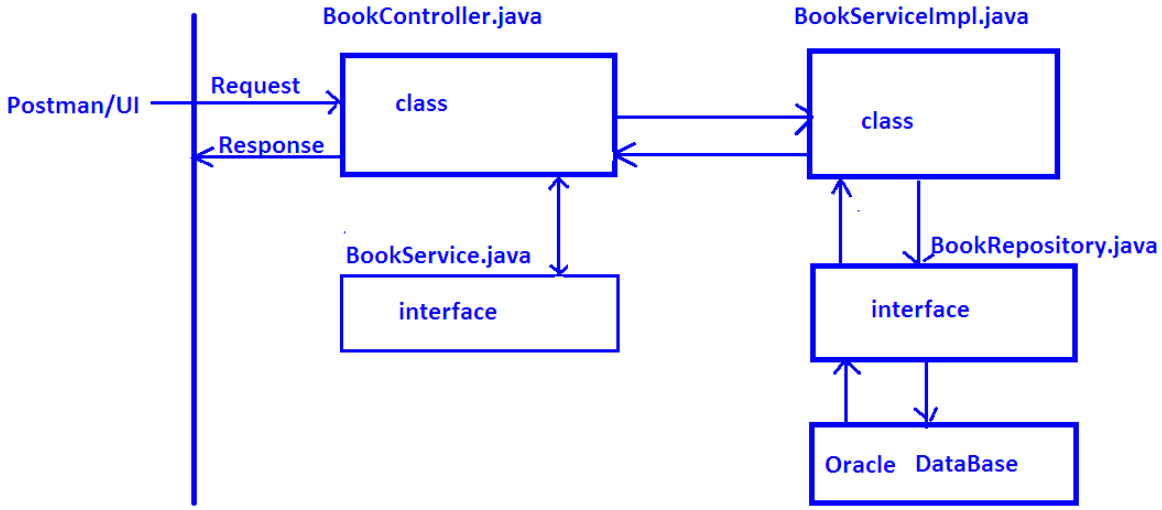
Let's check the database.





Yes, we have data in the book as well as story table. Did you notice in the book table book_id is a foreign key?

Let’s see in the below diagram which will give us a brief about flow.



That’s all about One To Many Mapping Annotation Example In Hibernate/JPA Using Spring Boot And Oracle.

One to One Mapping Annotation Example in Hibernate/JPA using Spring Boot and Oracle.
One to One Bidirectional Mapping Example In Hibernate/JPA Using Spring Boot and Oracle.
One To Many Bidirectional Mapping In Hibernate/JPA Annotation Example Using Spring Boot and Oracle
Many To One Unidirectional Mapping In Hibernate/JPA Annotation Example Using Spring Boot and Oracle.
Many To Many Mapping Annotation Example In Hibernate/JPA Using Spring Boot And Oracle.

@OneToMany **Docs**.

Summary – We have seen One To Many Mapping Annotation Example In Hibernate/JPA Using Spring Boot And Oracle.

Previous post **Next post**

BE THE FIRST TO COMMENT

ON "ONE TO MANY MAPPING ANNOTATION EXAMPLE IN HIBERNATE/JPA USING SPRING BOOT AND ORACLE"

Leave a comment

Your email address will not be published.

Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT