Spring Boot

1. Spring Boot is a java based framework.
2. Spring boot is use for a faster development and also it reduce the development efforts.
3. Using a Spring Boot you can create a production ready application.
4. Spring Boot provide embedded servers like Tomcat server, Data Base server such as H2 Database.
5. Spring boot is a flexible framework where you can customize as per your requirement.
6. Spring Boot is based on Spring Framework.
7. Spring Boot is majorly used for Full stack application and microservices.
8. Spring framework is divided into multiple module. You can use a module as per you project requirement.
   1. Spring IOC/Core
   2. Spring JDBC
   3. Spring ORM
   4. Spring REST
   5. Spring batch
   6. Spring MVC
   7. Spring Cloud
   8. Spring JMS
   9. Spring Security
9. How to implement Module in spring boot project
   1. Add the dependencies of Spring Module into project.
   2. Configure the model into project using predefine configuration.
   3. Use a Spring APIs to implement the functionality.
10. Spring Documentation

<https://docs.spring.io/spring-framework/docs/current/reference/html/>

<https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

1. Spring boot is a wrapper of spring framework. Spring framework issue such as setup, configuration and dependency management etc. has been resolve in this spring boot.
   1. Spring Boot provides a starter project which contains the required dependency for the modules.
   2. Spring Boot provides the auto configuration feature using which the module configuration will be happened automatically. You can customize the configuration as per your requirement.
   3. Spring provides the embedded tomcat server and H2 server by default.

**Spring Boot Project Creation**

1. Spring CLI (Command Line Interface)
   1. In this approach you have to install the Spring CLI tool to create spring boot application.

<https://docs.spring.io/spring-boot/docs/current/reference/html/cli.html>

1. Spring STS (Spring Tool Suit)
   1. This is the IDE provided by Spring community. This is an extension for eclipse, IntelliJ etc. IDE.

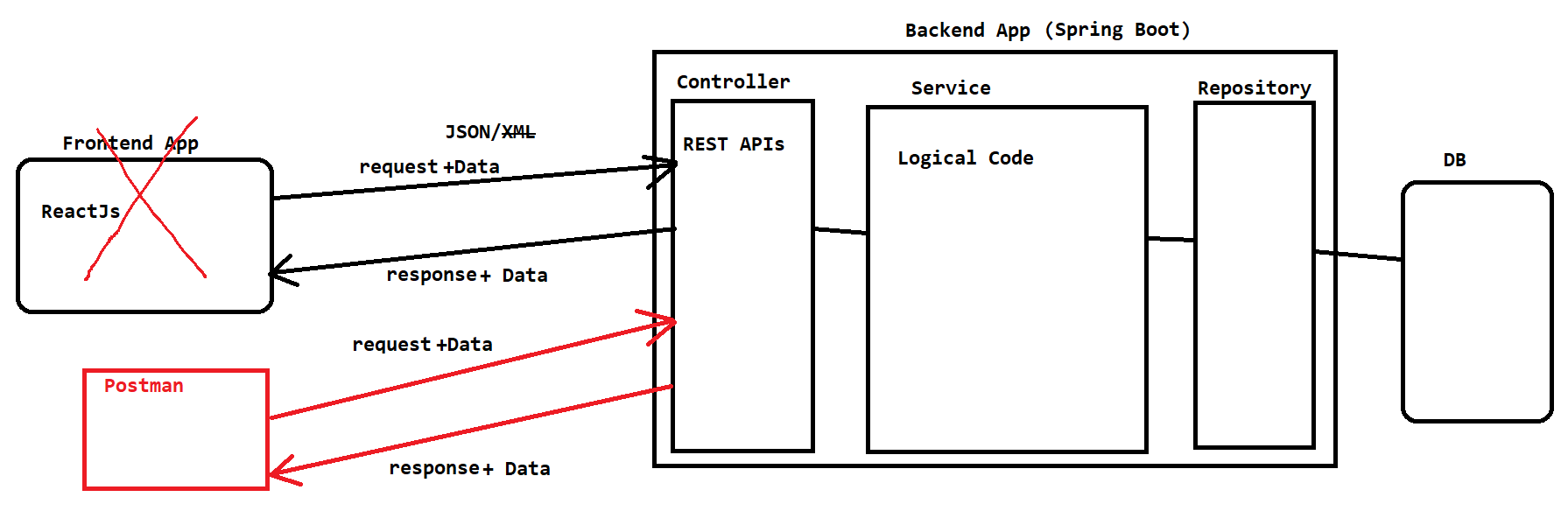
<https://spring.io/tools>

1. Spring Initializer
   1. Is a web application which is use to create a spring boot project. Using this you can create a spring project which will be downloaded as a zip file.

<https://start.spring.io/>



**Full Stack Application development**



**Create Spring Boot Project**

1. Go to Web Application to create spring boot project.

<https://start.spring.io/>



1. Get the Project Zip file and extract it into specific location.
2. Import the project into eclipse.
   1. Open an Eclipse workspace.
   2. Go To “File” Menu - > Select “Import…” option
   3. You can search for “Maven” option into the wizard of the new window



* 1. Select “Existing Maven Project” option in the list and click on “Next”
  2. Select the extracted folder as a Root Directory (Make sure that select the folder which has pom.xml)
  3. Click on “Finish”

**Spring Core**

**@SpringBootApplication annotation**

1. It is use to define the initial activated in the spring application.
2. This annotation is a combination of 3 annotations
3. @Configuration
   1. It denotes the class is a configuration class which can have the configuration for the application.
   2. Configuration classes will be loaded inside the spring container at the initial stage.
4. @EnableAutoConfiguration
   1. This is use to enable the auto configuration feature of the spring boot.
   2. This configuration will be perform by spring boot internally by looking into the dependencies included into project.
5. @ComponentScan
   1. To scan the spring bean class from the project.
   2. This will be scan and the object will be created and added inside spring container.

**Spring Bean classes**

1. Spring bean classes are the java classes for which spring creates an object and manage the life cycle of the object.
2. This classes can be the build-in or custom classes.
3. To create and maintain the java object spring internally spring use a container which is an ApplicationContext container.
4. To declare any java class as a spring bean class you can use the following annotation on class level.
   1. @Component
   2. @Controller
   3. @RestController
   4. @Service
   5. @Repository
   6. @ControllerAdvice etc.

**Spring Container**

1. It is use to store and manage the spring bean classes.
2. The object will be store inside container and will be available whenever required.
3. In Spring ApplicationContext container is used to manage the object life cycle.
4. You can get the object of any bean class by using getBean() method

Dependency Injection(DI)

1. It is a process in which you can inject the object of one class into another class.
2. To fulfill the HAS-A relation of java you can perform the DI.
3. This process can be achieve manually by providing the Object of another.
4. Also this can be automate by java using @Autowire concept.
5. Autowire concept is use to automate the DI process.

**JavaScript Object Notation (JSON)**

1. Json is use as a common language between a different applications based on different platform developed in different programming languages.
2. JSON is use to exchange the data between the different application.
3. JSON is use as a data transmission languages.
4. JSON represent Key and Value pair.
5. JSON can be use in 2 forms
   1. JSON Object
   2. JSON Array
6. JSON supports the text/String, Numeric, Boolean data type.

**JSON Object**

1. JSON Object is a combination of JSON elements.
2. One JSON element will have a Key=value pair.
3. Key and value pair is called as JSON element.
4. Following syntax use to represent JSON

{

“Key” : Value,

“Key” : Value

}

1. Example

{

“name” : “Abcd”,

“salary” : 4645.43,

“active” : true,

“contact” : “9988776633”

}

**JSON Array**

1. JSON Array is a group of values or group of JSON Object
2. JSON array will be represent as follows
3. Syntax

[ value1, value2, value3]

[{}, {}, {}]

1. Example

[“Abc”, “Xyz”, “Pqr”, “Lmn”]

[

{

“name” : “Abc”,

“contact” : “998876554”

},

{

“name” : “Xyz”,

“contact” : “778876554”

},

{

“name” : “LMN”,

“contact” : “818876554”

}

]

Example of JSON

{

“eid” : 2342,

“name” : “Abcd”,

“contact” : “978967867”,

“salary” : 54645.45,

“address” : {

“city” : “Pune”,

“pincode” : 546451

“state” : “MH”

},

“skills” : [“Core Java”,”Spring Boot”,”SQL”,”HTML”, “CSS”, “ReactJS”],

“experience” : [

{

“companyName” : “Abc Pvt.Ltd”,

“years” : 5.6

},

{

“companyName” : “Xyz Pvt.Ltd”,

“years” : 3.1

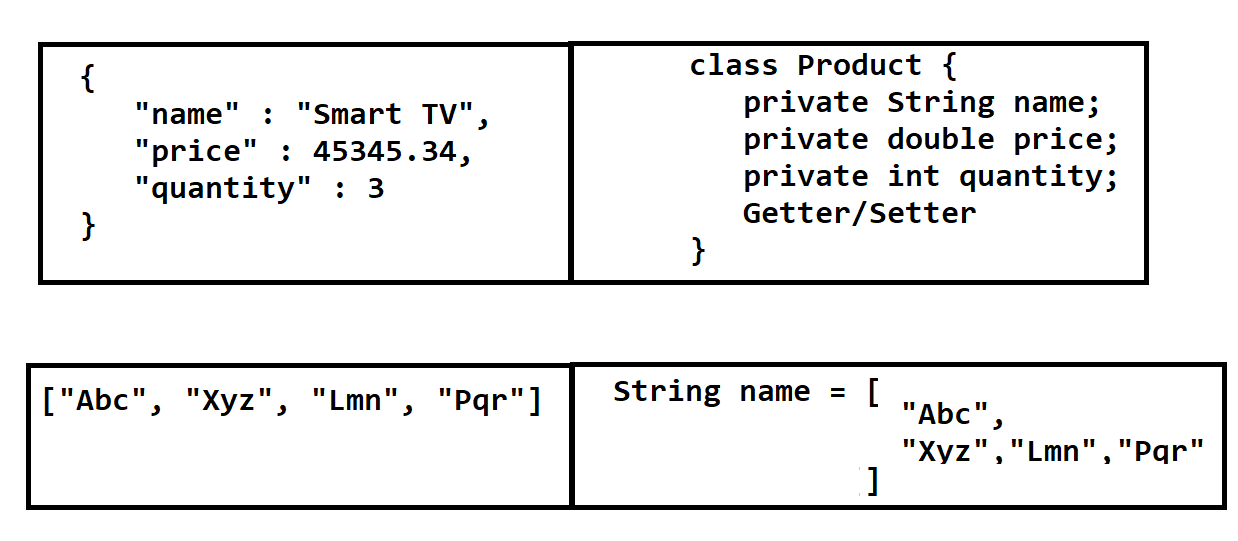
}

]

}

**JSON To Java conversion and wise versa**

1. JSON has to convert into Java Object once it is received in the Java Application.
2. The converting JSON into Java object or Java Object JSON you can use following libraries
   1. GSON
   2. Jackson
3. Spring Boot by default internally use the Jackson library to achieve this conversion.



Postman Tool.

1. This tool is use to test the REST APIs.
2. It is use to generate the requests and get the response back.
3. To download it use the following URL

<https://www.postman.com/downloads/>

**REST API using Spring Boot**

1. REST stans for **Re**presentational **S**tate **T**ransport.
2. REST API is also known as Restful API.
3. REST API is use to expose the services (Web services) of one application to the another application.
4. REST APIs accepts a JSON/~~XML~~ and return the JSON/~~XML~~
5. Create REST API in Spring Boot Application
   1. Create a java class and convert it into Rest Controller in Spring boot application. Using @RestController annotation.
   2. Create a method which return the values as a Rest API output(response).
   3. Annotate the method using an annotation such as @GetMapping, @PostMapping, @PutMapping, @DeleteMapping etc.
   4. Provide the URL of the REST API which is also known as End Point

**Accept the user data from the REST API**

1. User data can be accept using different option.
2. Accepting Data using **Request Parameter**
   1. It is use to get the user data using an request parameter
   2. These parameters are pass from the URL. Which is appended inside request after ‘?’
   3. To accept the parameter value into the spring controller you have to create an input parameter of the method.
   4. And use **@RequestParam** annotation to receive the parameter values inside the input variable.

URL : localhost:8080/user-name**?fn=Abcd&mn=Pqr&ln=Xyz**

`

1. Accept data using **path variable**
   1. It is use to get the user data using a path variable
   2. This is use to pass the Values as a part of URL. Which is also known as dynamic part of the URL
   3. To receive the values in the spring boot application you can use **@PathVariable** annotation.

URL : localhost:8080/user-name/**abc/pqr/xyz**



1. Accept Data using **JSON (Request Body)**
   1. It is use to get the user data using a request body in the form of JSON
   2. This data will be pass as the request body from the Internal of the URL.
   3. To receive this data into sprig boot application you can use **@RequestBody** annotation

URL : localhost:8080/user-data



**Http Methods for REST API**

1. GET
   1. GET method is recommended to get/retrieve the records from the backend application.
   2. To Create **HTTP GET** method you can use a **@GetMapping annotation**
   3. If you use the same method multiple time then it **return an same result** and hence it is also known as **idempotent method**.
2. POST
   1. POST method is use to create/insert a new resources at the backend application.
   2. To create **HTTP POST** method you can use **@PostMapping annotation**
   3. If you use the same method multiple time then it **return an different result** and hence it is also known as **non-idempotent method**.
3. PUT
   1. PUT method is use to update the existing resources from the backend application.
   2. To create **HTTP PUT** method you can use **@PutMapping annotation**
   3. If you use the same method multiple time then it **return an same result** and hence it is also known as **idempotent method**.
4. DELETE
   1. DELETE method is use to delete the existing resources from the backend application.
   2. To create **HTTP DELETE** method you can use **@DeleteMapping annotation**
   3. If you use the same method multiple time then it **return an same result** and hence it is also known as **idempotent method**.

**Swagger**

1. Is use to generate the REST API documentation
2. It is also known as OPEN API in the latest version.
3. Swagger is use to get the API documentation from the code written in the backend/spring boot application.
4. It generates an HTML document which can be access to get the API documentation.
5. Steps to add swagger tool into application
   1. Add the swagger dependency into application

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-ui</artifactId>

<version>1.6.12</version>

</dependency>

* 1. Execute the Application and access the HTML document using URL

<http://localhost:8080/swagger-ui.html>

1. Swagger Documentation Details

<https://springdoc.org/>

Task 1:

Create a Spring Boot Application Which Manage the Employee Details

Employee Details: id, name, salary, city

Create a REST API to Insert the Employee, Update the employee, delete the employee, get list of employee.

Task 2:

Create a Spring Boot Application Which Manage the Books Details

Books Details: id, name, author, publishingdate

Create a REST API to Insert the book, Update the book, delete the book, get list of books.

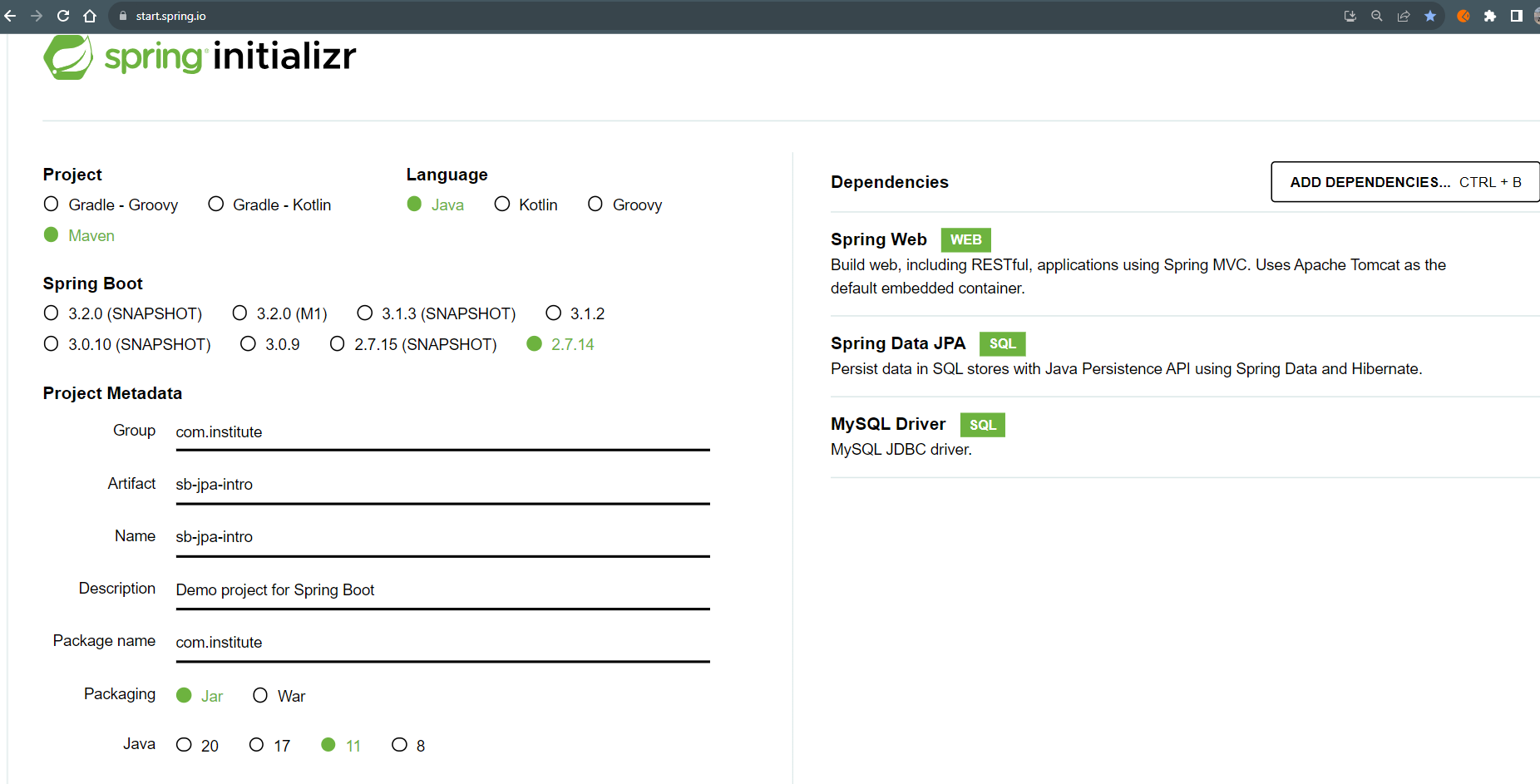
**Spring ORM**

1. This module is use to connect Spring applications with the Database using ORM tool.
2. ORM Tool
   1. ORM stands for **O**bject **R**elational **M**apping
   2. These tools are use to connect with databases and also use to perform Db operation more efficiently.
   3. Examples of ORM tools are JPA, EJB, Hibernate.
   4. In these tools you can connect with DB using the configuration instead of writing a code.
   5. It can generate the DDL and DML queries internally. And also execute them to get the results.
   6. The SQL checked exception has been converted into unchecked in ORM tools.
   7. It also provided the caching to improve the performance of the DB operations.

****

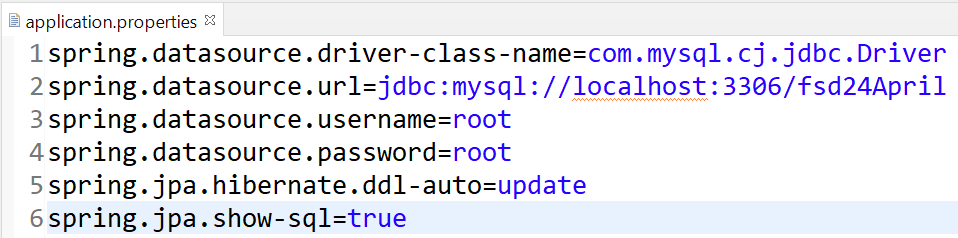
**Steps to create and Implement Spring JPA in Project**

1. Create a Project



1. Add JPA/DB configuration
   1. DB Configuration is to connect Spring Boot application with Database using JPA.
   2. This configuration includes the UserName, Password, url, driver etc.
   3. Configuration has to add inside the **application.properties** file present inside src/main/resources
   4. Refer Link to find all Spring Boot Properties:

<https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html>



1. Create Entity class
   1. You have to create entity class with private variable and getter setter.
   2. There are two mandatory annotation you have to use inside entity class @Entity and @Id.
   3. Entity class will be connect with the table using hibernate.
2. Using APIs execute Db Operations

**JpaRepository interface**

* 1. Using JpaRepository you can perform the DML operations of the DB.
  2. Using this Api you can execute the queries and get result back.
  3. Steps for create JpaRepository
     1. Create a custom interface and extends JpaRepository interface on it.
     2. Add a generic type as the Name of the entity class and the data type of primary key.
     3. Use @Repository annotation on the entity class.
  4. There are multiple pre-define methods inside repository which is use to perform the different Db operation such as insert, update, delete, select etc.

**Using Query in Spring Boot JPA**

1. To perform the Db operation other than primary key, you must have to write the query manually.
2. There are 2 ways to provide the query in the JPA
   1. HQL/JPQL
      1. In this type you will use the entity class name and variable name instead of column and table name.
      2. This queries are independent to the database, so the changes happened in the table will not effect you queries.
      3. This will gets converted into the SQL internally.
   2. SQL
      1. In this you will use the table and column name like an normal SQL query.
      2. This query is also known as native queries.
      3. While execution hibernate will not apply any conversion on it.
3. Steps To execute manual query
   1. Create a method inside Repository.
   2. Annotate that method with @Query annotation. And define your query.
   3. If query is of DML type then also use @Modifying and @Transactional annotation.
   4. If query is SQL type then set the a native query flag.

**JPA Relational Mapping (HAS-A)**

1. Using this you can achieve a relational mapping between the table. And also can add the primary and foreign key relation between tables.
2. You can create a reference of one class into another class to create a foreign key inside the Database table.
3. Create a reference of a class for which you wanted to create a foreign key, in the class where you wanted to create a foreign key column.
4. And then annotate the reference with the any one of the following annotation
   1. @OneToOne
   2. @OneToMany
   3. @ManyToOne
   4. @ManyToMany

