Hibernate - One-to-Many Mapping

Last Updated : 26 Aug, 2025

Hibernate is an ORM (Object-Relational Mapping) framework in Java that allows developers to map Java objects to database tables. One of the common relationships in databases is One-to-Many, where one record in a parent table can be associated with multiple records in a child table. Hibernate makes it easy to implement this relationship in Java.

Real-Life Example

Consider bike manufacturers:

- A manufacturer can produce multiple bike models.
- The same bike model cannot be produced by multiple manufacturers.

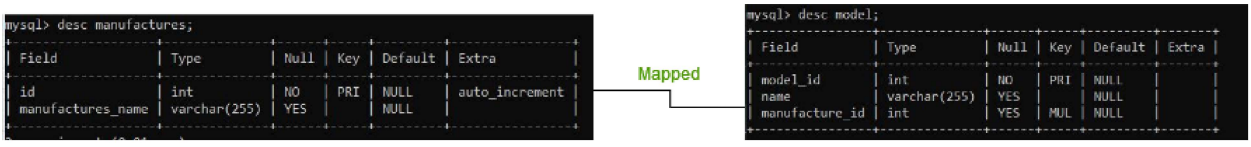
This is a classic one-to-many relationship, where one manufacturer>many models.

Syntax:

This mapped variable of the other tables is responsible for mapping between two tables.

@oneToMany(mappedby=”nameofmappedvariable”)

For this example, we will map the *id* field of *m* to the *manufacture_id* of the *model* entity.



Step by Step Implementation

Step 1. Create the MySQL database

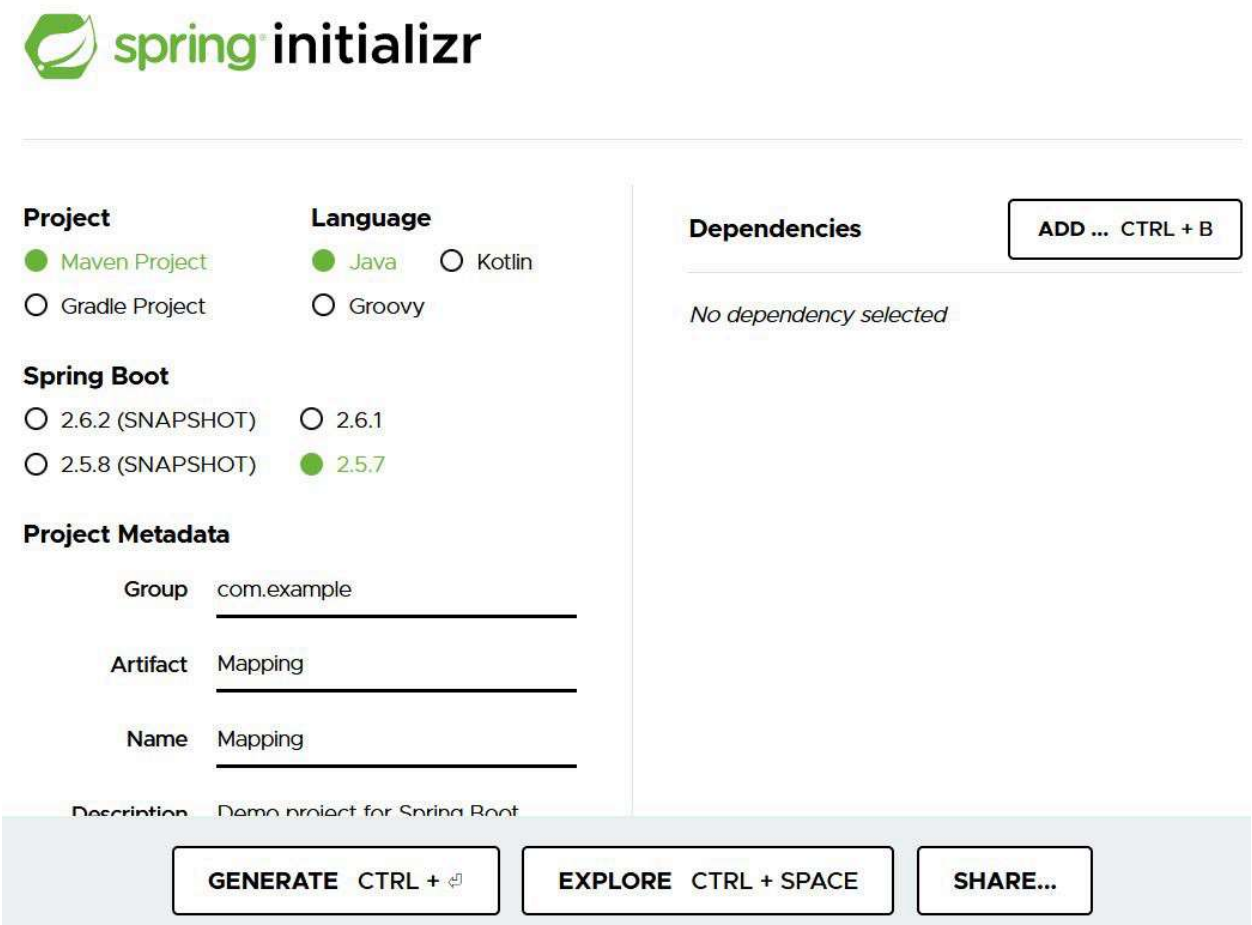
```
CREATE DATABASE mapping;
```

Step 2. Generate the starter project (spring Initializr)

Project: Maven
Language: Java
Spring Boot: 2.5.7
Packaging: JAR
Java: 11
Dependencies:

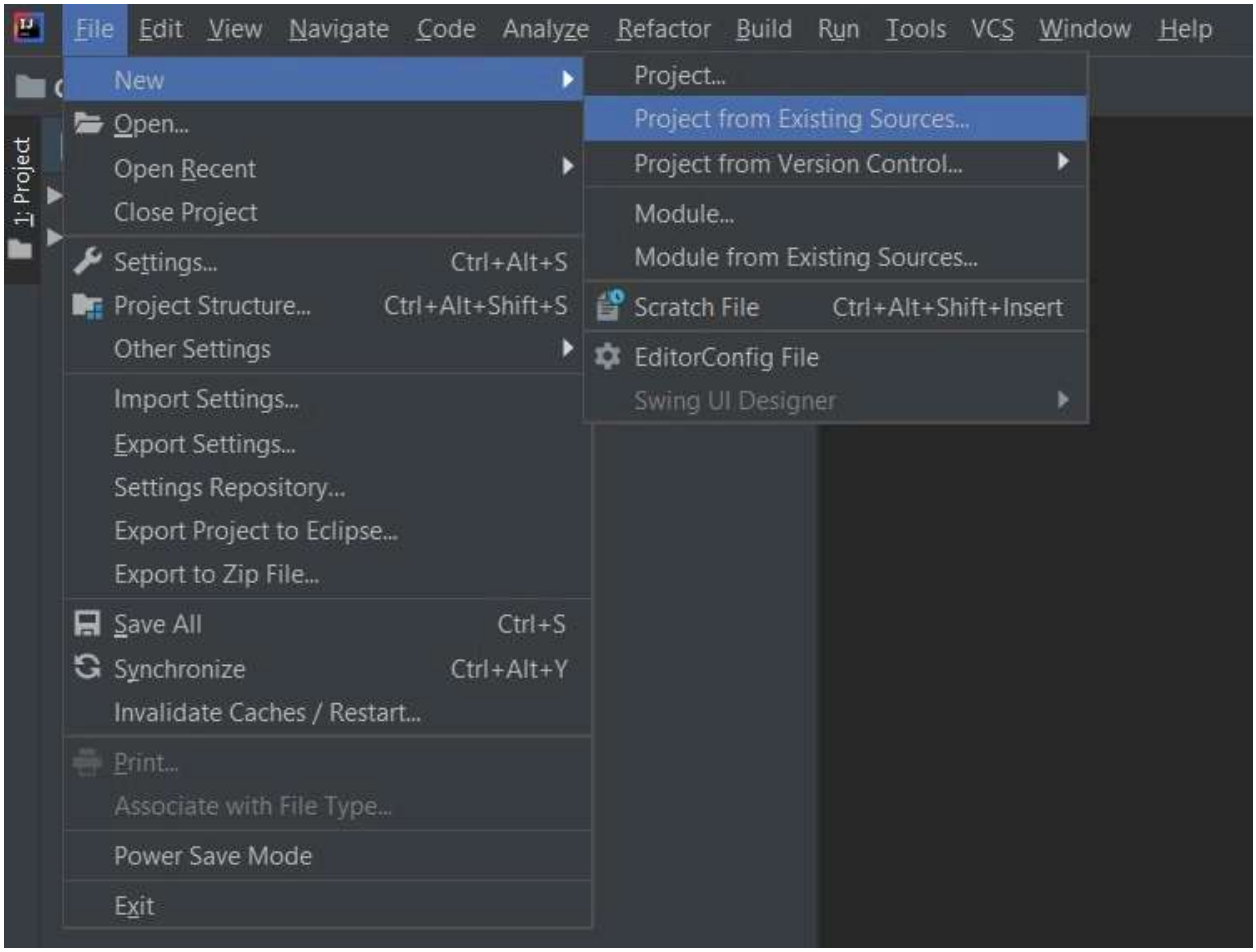
- Spring Web
- Spring Data JPA
- MySql Driver

Click Generate



Step 3. Importing the Project

1. Open your IDE.
2. **File > New > Project from Existing Sources**.
3. Select `pom.xml` and import the project.



Step 4: Database configuration

Database configuration required for connection of Mysql database to the Spring Boot Application.

application.properties

spring.datasource.username=root

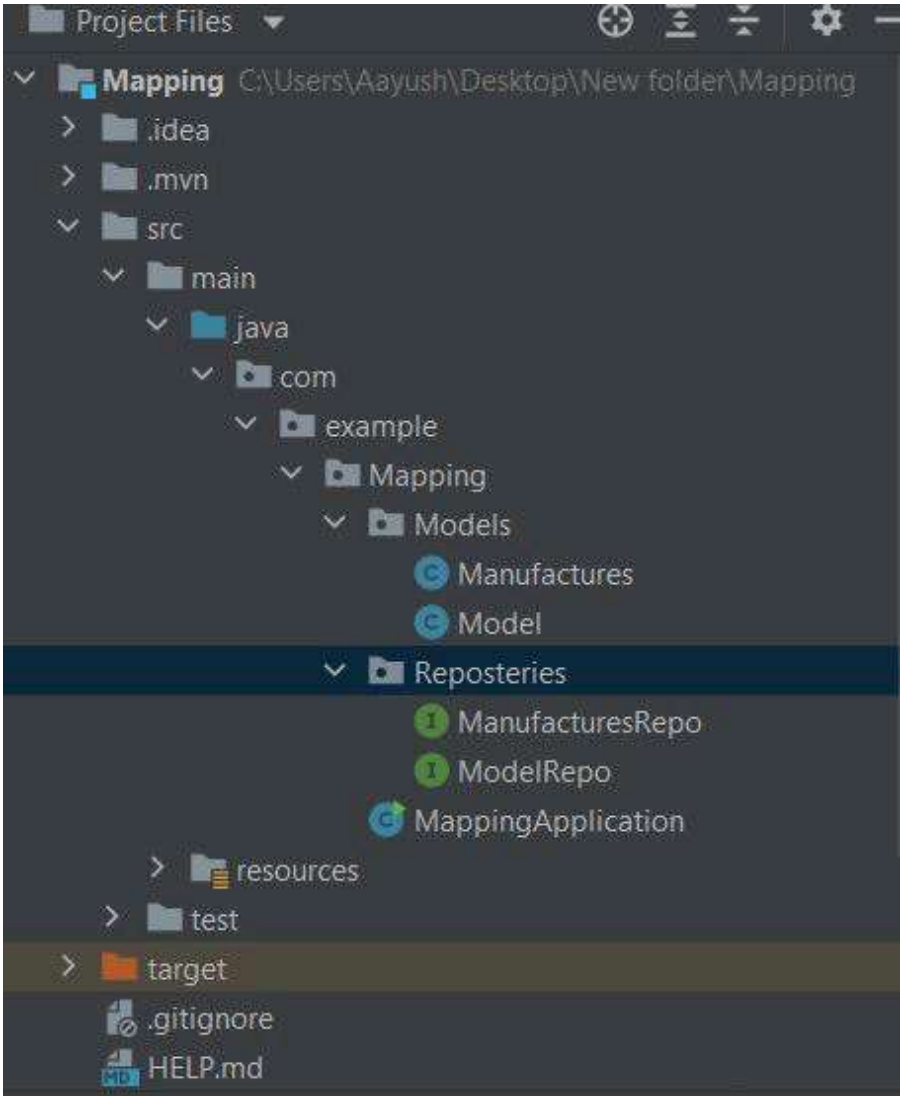
spring.datasource.password=Aayush

spring.datasource.url=jdbc:mysql://localhost:3306/mapping

spring.jpa.hibernate.ddl-auto=update

Step 5: Create folder structure

Go to src-> main -> java -> com -> example -> Mapping and create two files in the models folder i.e *Manufactures.java* and *Model.java*



Step 6: Add entity

1. Create Manufactures.java class : This contains the entity Manufactures, which contains the required associated fields.

Manufactures.java class

```
package com.example.Mapping.Models;

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

@Entity
@Table(name = "manufactures")
public class Manufactures {

    // Unique identifier for the manufacturer
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
```

```
    // Name of the manufacturer
    private String manufactures_name;

    // A manufacturer can have many models
    @OneToMany(mappedBy = "manufacturer", cascade = CascadeType.ALL)
private List<Model> models;

    // Constructor with both id and name
    public Manufactures(String manufactures_name) {
    this.manufactures_name = manufactures_name;
}

    // Default constructor
    public Manufactures() {
    }

    // Getters and setters for id and name

    public int getId() {
        return id;
    }

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

    public String getManufactures_name() {
        return manufactures_name;
    }

    public void setManufactures_name(String manufactures_name) {
        this.manufactures_name = manufactures_name;
    }

    // Getter for models
    public List<Model> getModels() {
        return models;
    }

    // Setter for models
    public void setModels(List<Model> models) {
        this.models = models;
    }
}
```

2. Create Model.java

This contains the entity Model with various fields and methods-

```
package com.example.Mapping.Models;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
```

```java
import javax.persistence.ManyToOne;

@Entity
@Table(name = "models")
public class Model {

    // Unique identifier for the model
    @Id
    private int model_id;

    // Name of the model
    private String name;

    // A model belongs to one manufacturer Foreign key referencing the
manufacturer table
    @ManyToOne
    @JoinColumn(name = "manufacture_id")
    private Manufactures manufacturer;

    // Constructor with all fields
    public Model(int model_id, String name, Manufactures manufacturer) {
        this.model_id = model_id;
        this.name = name;
        this.manufacturer = manufacturer;
    }

    // Default constructor
    public Model() {
    }

    // Getters and setters

    public int getModel_id() {
        return model_id;
    }

    public void setModel_id(int model_id) {
        this.model_id = model_id;
    }

    public String getName() {
        return name;
    }

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

    public Manufactures getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(Manufactures manufacturer) {
        this.manufacturer = manufacturer;
    }
}
```

Step 7. Add repositories

1. Create ManufactureRepo Class: This Interface contains the forms the bridge of entity Manufactures to the Database

ManufacturesRepo.java

```
package com.example.Mapping.Repositories;

import com.example.Mapping.Models.Manufactures;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ManufacturesRepo extends JpaRepository<Manufactures,
Integer> {
}
```

2. Create ModelRepo.java class: This Interface contains the forms the bridge of entity *Model* to the Database

ModelRepo.java

```
package com.example.Mapping.Repositories;

import com.example.Mapping.Models.Model;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ModelRepo extends JpaRepository<Model, Integer> {
}
```

Step 8. Application drive

Driver class for Spring Boot application to test the One-to-Many Mapping of Hibernate.

- This Spring Boot application initializes and saves data using JPA repositories.
- It creates a Manufacturer *("Honda")* entity and two associated Model (*"AYZ" and "ZET")* entities with a one-to-many relationship.
- The data is then stored in corresponding repositories during the application's run.

MappingApplication.java

```java
package com.example.Mapping;

import com.example.Mapping.Models.Manufactures;
import com.example.Mapping.Models.Model;
import com.example.Mapping.Repositories.ManufacturesRepo;
import com.example.Mapping.Repositories.ModelRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * Spring Boot application for mapping relationships between Manufactures and
Models.
 */
@SpringBootApplication
public class MappingApplication implements CommandLineRunner {

    // Injecting Manufactures and Model repositories using autowiring
    @Autowired
    private ManufacturesRepo manufacturesRepo;
    @Autowired
    private ModelRepo modelRepo;

    public static void main(String[] args) {
        SpringApplication.run(MappingApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {

        // Creating a new Manufactures object with ID 1 and name "Honda"
        Manufactures data = new Manufactures("Honda");

        // Saving the Manufactures record
        manufacturesRepo.save(data);

        // Creating two new Model objects associated with the "Honda"
manufacturer
        Model model1 = new Model(1, "AYZ", data);
        Model model2 = new Model(2, "ZET", data);

        // Saving the Model records
        modelRepo.save(model1);
        modelRepo.save(model2);
    }
}
```

Run the main application:

Console log of project running.

```
: HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
: HikariPool-1 - Starting...
: HikariPool-1 - Start completed.
: HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transacti
: Initialized JPA EntityManagerFactory for persistence unit 'default'
: spring.jpa.open-in-view is enabled by default. Therefore, database queries m
: Tomcat started on port(s): 8080 (http) with context path ''
: Started MappingApplication in 6.117 seconds (JVM running for 6.762)
```

Step 9. Verify the DB content

Database view of *manufactures* table:

*SELECT * FROM manufactures;*

```
mysql> select * from manufactures;
+----+--------------------+
| id | manufactures_name  |
+----+--------------------+
|  1 | Honda              |
+----+--------------------+
```

Model Table

Database view of *model* table:

*SELECT * FROM model;*

```
mysql> select * from model;
+----------+------+-----------------+
| model_id | name | manufacture_id  |
+----------+------+-----------------+
|        1 | AYZ  |               1 |
|        2 | ZET  |               1 |
+----------+------+-----------------+
2 rows in set (0.02 sec)
```

The below image depicts rows in manufactures and model. model will contain manufacture_id values linking to manufactures.

```
mysql> desc manufactures;
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| id               | int         | NO   | PRI | NULL    | auto_increment |
| manufactures_name | varchar(255) | YES |     | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
```

Mapped

```
mysql> desc model;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| model_id       | int         | NO   | PRI | NULL    |       |
| name           | varchar(255) | YES |     | NULL    |       |
| manufacture_id | int         | YES  | MUL | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```