Hibernate - Many-to-One Mapping

Last Updated : 26 Aug, 2025

In relational databases, a Many-to-One relationship occurs when multiple records in one table are associated with a single record in another table.

For example, in a workplace scenario:

- Many employees work in the same company.
- Each employee is linked to a single company address.

This type of relationship helps to avoid data redundancy and maintain consistency.

Syntax:

```
@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "Foreign key column")
```

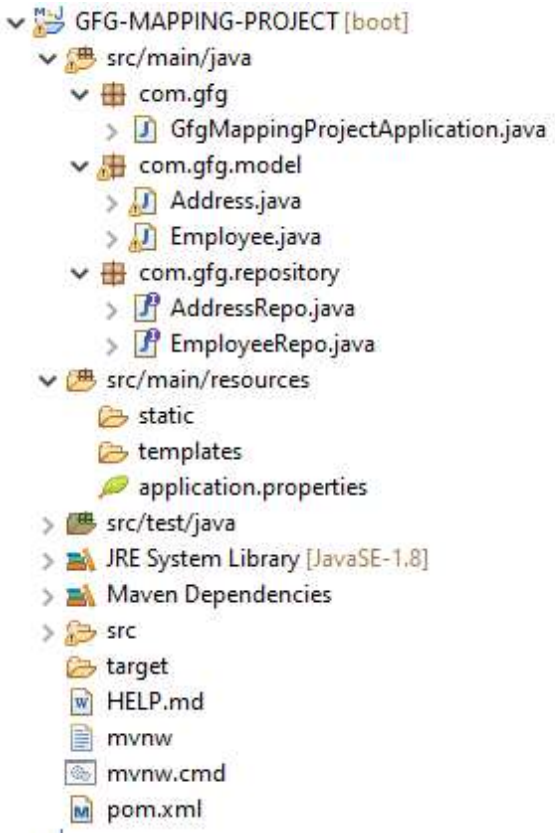
Step-by-Step Implementation

Step 1: Create a Spring Boot project using STS

Create a project using STS and enter project details in the New Spring Starter Project window:

- **Name / Artifact:** GFG-MAPPING-PROJECT
- **Group / Package:** com.gfg
- **Type:** Maven Project
- **Packaging:** Jar
- **Java Version:** 8
- **Description:** Demo project for Hibernate Mapping

Click Next to select Spring Boot version and dependencies (Spring Data JPA, MySQL Driver), then click **Finish.**



Dependencies

Step 2: Configure application.properties

Go to src/main/resources open **application.properties** in that adding the necessary properties as below

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

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

spring.datasource.username=root

spring.datasource.password=password

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

Step 3: Create Model Classes

Go to src/main/java create a package for model classes(ex : com.gfg.model) and create one package for repository (ex :com.gfg.repository). Then create two model classes under the model package

- Employee.java
- Address.java

Employee.java

```
package com.gfg.model;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
public class Employee {

    @Id private int empId;
    private String empName;

    // Many employees has one company address
    @ManyToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "address_id")
    private Address address;

    public Employee(int empId, String empName,
                    Address address)
    {
        super();
        this.empId = empId;
        this.empName = empName;
        this.address = address;
    }

    public Employee() { super(); }

    @Override public String toString()
    {
        return "Employee []";
    }

    public int getEmpId() { return empId; }

    public void setEmpId(int empId) { this.empId = empId; }
```

```
    public String getEmpName() { return empName; }

    public void setEmpName(String empName)
    {
        this.empName = empName;
    }

    public Address getAddress() { return address; }

    public void setAddress(Address address)
    {
        this.address = address;
    }
}
```

Address.java

```
package com.gfg.model;

import java.util.ArrayList;
import java.util.List;
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

@Entity
public class Address {

    @Id private int addressId;
    private String location;

    @OneToMany(cascade = CascadeType.ALL,
               mappedBy = "address")
    private List<Employee> employee
        = new ArrayList<>();

    public Address(int addressId, String location)
    {
        super();
        this.addressId = addressId;
        this.location = location;
    }

    public Address() { super(); }

    public int getAddressId() { return addressId; }

    public void setAddressId(int addressId)
    {
        this.addressId = addressId;
    }
```

```
    public String getLocation() { return location; }

    public void setLocation(String location)
    {
        this.location = location;
    }

    public List<Employee> getEmployee() { return employee; }

    public void setEmployee(List<Employee> employee)
    {
        this.employee = employee;
    }
}
```

Step 4: Create Repository Interfaces

Go to repository package adding the JPA repository of both model classes

EmployeeRepo.java

```
package com.gfg.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.gfg.model.Employee;

public interface EmployeeRepo extends JpaRepository<Employee,Integer>{

}
```

AddressRepo.java

```
package com.gfg.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.gfg.model.Address;

public interface AddressRepo extends JpaRepository<Address,Integer>{

}
```

Step 5: Main Application & Data Insertion

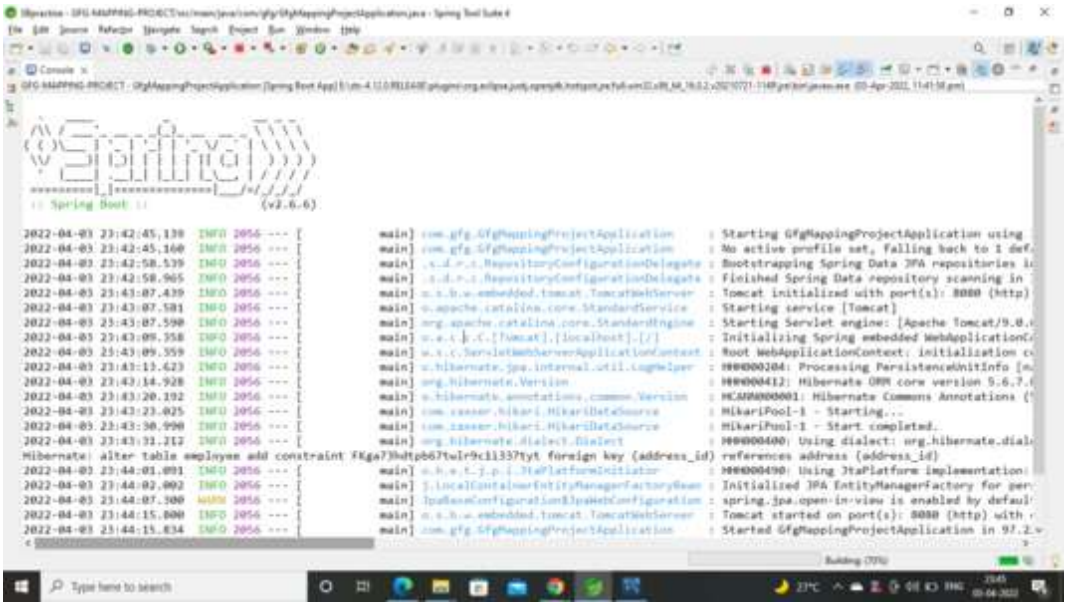
Go to starter class Autowire two repository interfaces and create objects for model classes

GfgMappingProjectApplication.java

```
package com.gfg;

import com.gfg.model.Address;
import com.gfg.model.Employee;
import com.gfg.repository.AddressRepo;
import com.gfg.repository.EmployeeRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class GfgMappingProjectApplication
    implements CommandLineRunner {

    @Autowired AddressRepo addessRepo;
    @Autowired EmployeeRepo empRepo;

    public static void main(String[] args)
    {
        SpringApplication.run(
            GfgMappingProjectApplication.class, args);
    }
    @Override
    public void run(String... args) throws Exception
    {

        Address addr = new Address(1, "Bangloor");
        addessRepo.save(addr);

        Employee emp1 = new Employee(1, "Alpha", addr);
        Employee emp2 = new Employee(2, "Beeta", addr);

        empRepo.save(emp1);
        empRepo.save(emp2);
    }
}
```

Step 6: Run Application & Verify

- Right-click GfgMappingProjectApplication -> Run as Spring Boot Application



Run Application & Verify

Then go to Database then check for tables.

Address Table:

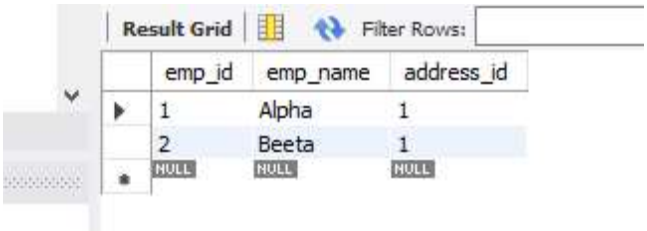
*select * from address;*

address_id	location
1	Bangloor
NULL	NULL

Address Table

Employee Table:

*select * from employee;*



emp_id	emp_name	address_id
1	Alpha	1
2	Beeta	1
NULL	NULL	NULL

Employee Table

- One address record in address table
- Two employee records linked to that address in employee table