Room

//Room.class

//Model

```
package edu.cseju.seatplan.models;

import org.springframework.stereotype.Component;

import javax.persistence.Column;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;


@Entity
@Table(name = "tblRoom")
public class Room {
    @Id
    @Column(name = "room_no")
    @NotNull(message = "This field Can Not be NULL")
    private String roomNo;
    @Column(name = "no_of_row")
    private int noOfRow;

    @Column(name = "bench_per_row")
    private int benchPerRow;

    @Column(name = "bench_capacity")
    private int benchCapacity;

    public String getRoomNo() {
        return roomNo;
    }

    public void setRoomNo(String room_no) {
        this.roomNo = room_no;
    }

    public int getNoOfRow() {
        return noOfRow;
    }

    public void setNoOfRow(int noOfRow) {
        this.noOfRow = noOfRow;
```

```
    }

    public int getBenchPerRow() {
        return benchPerRow;
    }

    public void setBenchPerRow(int benchPerRow) {
        this.benchPerRow = benchPerRow;
    }

    public int getBenchCapacity() {
        return benchCapacity;
    }

    public void setBenchCapacity(int benchCapacity) {
        this.benchCapacity = benchCapacity;
    }

    @Override
    public String toString() {
        return "Room{" +
                "roomNo='" + roomNo + '\'' +
                ", noOfRow=" + noOfRow +
                ", benchPerRow=" + benchPerRow +
                ", benchCapacity=" + benchCapacity +
                '}';
    }

    public Room() {
        roomNo="";
        noOfRow=1;
        benchPerRow=1;
        benchCapacity=1;

    }
}
```

```
//Room Repository
//Interface Class

package edu.cseju.seatplan.repository;

import edu.cseju.seatplan.models.Room;
import org.springframework.data.jpa.repository.JpaRepository;


import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

@Repository
@Transactional
public interface RoomRepository extends
JpaRepository<Room,String> {

}
```

```
//Room Service
//Interface Class


package edu.cseju.seatplan.services;

import edu.cseju.seatplan.models.Room;

import java.util.List;

public interface RoomService {
    public List<Room> getAllRooms();
    public Room getRoomById(String roomNo);
    public void saveOrUpdate(Room room);
    public void removeRoom(String roomNo);
}
```

```java
//Room Service Implement Class

package edu.cseju.seatplan.services;

import edu.cseju.seatplan.models.Room;
import edu.cseju.seatplan.repository.RoomRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service
@Transactional
public class RoomServiceImpl implements RoomService {

    @Autowired
    RoomRepository roomRepository;

    @Override
    public List<Room> getAllRooms() {
        return roomRepository.findAll();
    }

    @Override
    public Room getRoomById(String roomNo) {
        return roomRepository.getOne(roomNo);
    }

    @Override
    public void saveOrUpdate(Room room) {
        roomRepository.save(room);

    }

    @Override
    public void removeRoom(String roomNo) {
        roomRepository.deleteById(roomNo);

    }
}
```

//ROOM Controller

```
package edu.cseju.seatplan.controllers;

import edu.cseju.seatplan.models.Room;
import edu.cseju.seatplan.repository.RoomRepository;
import edu.cseju.seatplan.services.RoomService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;

import java.util.Collection;
import java.util.List;
import java.util.Map;

@RequestMapping("/rooms")
@Controller
public class RoomController {

    @Autowired
    private RoomService roomService;

    @RequestMapping(value = "",method = RequestMethod.GET)
    public ModelAndView showAllRoom(){

        ModelAndView allRooms=new ModelAndView();
        List<Room> list= roomService.getAllRooms();
        allRooms.addObject("rooms",list);
        allRooms.setViewName("Home");

        return allRooms;
    }

    @RequestMapping(value = "/addNew",method = RequestMethod.GET)
    public ModelAndView addNewRoom(){
        ModelAndView allRooms=new ModelAndView();
        Room room=new Room();
        allRooms.addObject("room",room);
        allRooms.setViewName("roomForm");
        return allRooms;
    }

    @RequestMapping(value = "/save",method = RequestMethod.POST)
    public ModelAndView saveRoom(@ModelAttribute("room") Room room){
        System.out.println(room.toString());

        roomService.saveOrUpdate(room);
        return new ModelAndView("redirect:/rooms");
    }

    @RequestMapping(value = "/edit/{roomNo}",method = RequestMethod.GET)
    public ModelAndView updateRoom(@PathVariable("roomNo") String
roomN){
```

controller command service class

th:each = "a : @{rooms}"

hello

```
        ModelAndView mv=new ModelAndView();
        Room room=roomService.getRoomById(roomN);
        mv.addObject(room);
        mv.setViewName("roomForm");
        return mv;
    }

    @RequestMapping(value = "/remove/{roomNo}",method =
RequestMethod.GET)
    public ModelAndView removeRoom(@PathVariable("roomNo") String
roomN){

        roomService.removeRoom(roomN);

        return new ModelAndView("redirect:/rooms");
    }


}
```

mv.addObject("hello",room)
"h". object

localhost:8080/rooms

//Room.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>Display Rooms</h1>
<table border="1">
    <thead>
    <tr>
        <th> Room Number</th>
        <th> Number of Row</th>
        <th> Bench Per Row</th>
        <th> Bench Capacity</th>
        <th>Total Capacity</th>
        <th>Action</th>
    </tr>

    </thead>
    <tbody>
    <a th:href="@{/rooms/addNew}" >Add New Room</a>
    <tr th:each="room: ${rooms}">
        <td th:text="${room.roomNo}"></td>
        <td th:text="${room.noOfRow}"></td>
        <td th:text="${room.benchPerRow}"></td>
        <td th:text="${room.benchCapacity}"></td>
        <td
th:text="${room.getBenchCapacity()*room.getNoOfRow()*room.getBenchPerRow
()}"></td>
        <td>
            <a role="button" th:href="@{'/rooms/edit/'+
${room.roomNo}}">EDIT </a>
             <a role="button" th:href="@{'/rooms/remove/' +
${room.roomNo}}">Delete</a>
        </td>


    </tr>
    </tbody>
    <tfoot>

    </tfoot>
</table></body>
</html>
```

updateroom

href

localhost: 8080 /rooms

//Room Form Add / Edit Form

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Form</title>
</head>
<body>
<form method="post" th:object="${room}" th:action="@{/rooms/save}">
    <label>Enter Room Number : </label>
    <input  type="text" th:field="*{roomNo}" th:value="${room.getRoomNo()}"  placeholder="Enter Room Number">
    <br><hr><label>Enter Number of Row : </label>
    <input  type="text" th:field="*{noOfRow}" th:value="${room.getNoOfRow()}" placeholder="Enter Number of Row">
    <br><hr><label>Enter Bench Per Row : </label>
    <input  type="text" th:field="*{benchPerRow}" th:value="${room.getBenchPerRow()}"  placeholder="Enter Bench Per Row">
    <br><hr><label>Enter Bench Capacity : </label>
    <input  type="text" th:field="*{benchCapacity}" th:value="${room.getBenchCapacity()}" placeholder="Enter Bench Capacity
    <br><hr>
    <input type="submit" value="Save">
</form>
</body>
</html>
```

hello
hello.getRoomNo()
hello.
hello.
hello.

//Application.Properties Configuration

```
//MySql Data Configuration

spring.datasource.driver = com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/db_seatplan?serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=

//Thymeleaf Configuration
spring.thymeleaf.suffix=.html


spring.thymeleaf.cache = false


//DAO Configuration

spring.jpa.show-sql = true
# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto = update

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

MySQL5InnoDBDialect

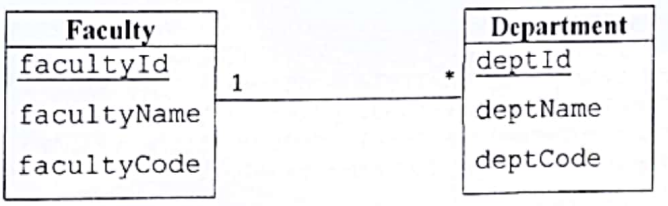
Dependencyje's :
Web, Dev Tool, JPA, MySql, Thymeleaf

Spring Boot CRUD Application with JPA, MySQL and Thymeleaf

(One-To-Many)

Overview:

In this experiment we will develop a CRUD web application using Spring Boot, JPA and MySQL as a database. Here "**db_seatplan**" is used as MySQL database and we will be using Room entity/model as a resource in this experiment.



Step 1. Create a spring boot web application with the following dependencies: Web, Thymeleaf, Dev Tool, JPA and MySQL

Step 2. Write the following code in the application.properties file

```
#Application.Properties Configuration

#MySql Data Configuration
spring.datasource.driver = com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/db_seatplan
spring.datasource.username=root
spring.datasource.password=

#Thymeleaf Configuration
spring.thymeleaf.suffix=.html
spring.thymeleaf.cache = false
#DAO Configuration
spring.jpa.show-sql = true
# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto = update

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

Step 3. With all the project dependencies already in place, let's now implement a naive domain layer.

Faculty.java

```
package edu.cseju.onetomany.model;

import javax.persistence.*;

import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
public class Faculty {
    @Id
```

```
    private String facultyId;
    private String facultyName;
    private String facultyCode;

    @OneToMany(fetch = FetchType.LAZY, orphanRemoval = true,mappedBy =
"faculty",cascade = CascadeType.ALL)
    private Set<Department> departments=new HashSet<Department>();

    public Set<Department> getDepartments() {
        return departments;
    }

    @Override
    public String toString() {
        return "Faculty{" +
                "facultyId='" + facultyId + '\'' +
                ", facultyName='" + facultyName + '\'' +
                ", facultyCode='" + facultyCode + '\'' +
                ", departments=" + departments +
                '}';
    }

    public void setDepartments(Set<Department> departments) {
        this.departments = departments;
    }

    public String getFacultyId() {
        return facultyId;
    }

    public void setFacultyId(String facultyId) {
        this.facultyId = facultyId;
    }

    public String getFacultyName() {
        return facultyName;
    }

    public void setFacultyName(String facultyName) {
        this.facultyName = facultyName;
    }

    public String getFacultyCode() {
        return facultyCode;
    }

    public void setFacultyCode(String facultyCode) {
        this.facultyCode = facultyCode;
    }
}
```

Department.java

```
package edu.cseju.onetomany.model;

import org.springframework.stereotype.Controller;

import javax.persistence.*;

@Entity
public class Department {
    @Id
```

```
    private String deptId;
    private String deptName;
    private String deptCode;

    @ManyToOne(fetch = FetchType.LAZY)
    private Faculty faculty;

    public String getDeptId() {
        return deptId;
    }

    @Override
    public String toString() {
        return "Department{" +
                "deptId='" + deptId + '\'' +
                ", deptName='" + deptName + '\'' +
                ", deptCode='" + deptCode + '\'' +
                ", faculty=" + faculty +
                '}';
    }

    public void setDeptId(String deptId) {
        this.deptId = deptId;
    }

    public String getDeptName() {
        return deptName;
    }

    public void setDeptName(String deptName) {
        this.deptName = deptName;
    }

    public String getDeptCode() {
        return deptCode;
    }

    public void setDeptCode(String deptCode) {
        this.deptCode = deptCode;
    }

    public Faculty getFaculty() {
        return faculty;
    }

    public void setFaculty(Faculty faculty) {
        this.faculty = faculty;
    }
}
```

EAGER

Step 4. Spring Data JPA allows us to implement JPA-based repositories

FacultyRepo

```
package edu.cseju.onetomany.repo;

import edu.cseju.onetomany.model.Faculty;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface FacultyRepo extends JpaRepository<Faculty,String> {
}
```

DepartmentRepo

```
package edu.cseju.onetomany.repo;

import edu.cseju.onetomany.model.Department;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface DepartmentRepo extends JpaRepository<Department,String> {
    public Department findDepartmentByFacultyFacultyCode(String facultyId);
}
```

Step 5. Service Interface

Faculty

```
package edu.cseju.onetomany.service;

import edu.cseju.onetomany.model.Faculty;

import java.util.List;

public interface FacultyService {
    public List<Faculty> getAllFaculty();
    public Faculty getFacultyById(String facultyId);
    public void saveOrUpdate(Faculty faculty);
    public void removeFaculty(String facultyId);
}
```

Department

```
package edu.cseju.onetomany.service;

import edu.cseju.onetomany.model.Department;

import java.util.List;

public interface DeptService {
    public List<Department> getAllDepartment();
    //public List<Department> getDepartmentByFaculty(String facultyId);
    public Department getDepartmentById(String departmentId);
    public void saveOrUpdate(Department department);
    public void removeDepartment(String departmentId);
}
```

Step 6. Service Implementation

Faculty.java

```
package edu.cseju.onetomany.service;

import edu.cseju.onetomany.model.Faculty;
import edu.cseju.onetomany.repo.FacultyRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
```

```
public class FacultyServiceImpl implements FacultyService {

    @Autowired
    private FacultyRepo facultyRepo;
    @Override
    public List<Faculty> getAllFaculty() {
        return facultyRepo.findAll();
    }

    @Override
    public Faculty getFacultyById(String facultyId) {
        return facultyRepo.getOne(facultyId);
    }

    @Override
    public void saveOrUpdate(Faculty faculty) {
        facultyRepo.save(faculty);
    }

    @Override
    public void removeFaculty(String facultyId) {
        facultyRepo.deleteById(facultyId);
    }
}
```

Department.java

```
package edu.cseju.onetomany.service;

import edu.cseju.onetomany.model.Department;
import edu.cseju.onetomany.repo.DepartmentRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class DeptServiceImpl implements DeptService {

    @Autowired
    private DepartmentRepo departmentRepo;

    @Override
    public List<Department> getAllDepartment() {
        return departmentRepo.findAll();
    }

    @Override
    public Department getDepartmentById(String departmentId) {
        return departmentRepo.getOne(departmentId);
    }

    @Override
    public void saveOrUpdate(Department department) {
        departmentRepo.save(department);
    }

    @Override
    public void removeDepartment(String departmentId) {
        departmentRepo.deleteById(departmentId);
    }
```

Step 7. Controller

Faculty Controller

```
package edu.cseju.onetomany.controller;

import edu.cseju.onetomany.model.Faculty;
import edu.cseju.onetomany.service.FacultyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;

import java.util.List;

@Controller
@RequestMapping("/faculty")
public class FacultyController {

    @Autowired
    private FacultyService facultyService;

    @RequestMapping("/user")
    public ModelAndView userView()
    {
        int accessTypeFlag=0; //accessTypeFlag=0 means user access
        ModelAndView modelAndView=new ModelAndView();
        List<Faculty> list=facultyService.getAllFaculty();

        modelAndView.addObject("listOfModel",list);
        modelAndView.addObject("accessTypeFlag",accessTypeFlag);

        modelAndView.setViewName("faculty");
        return modelAndView;
    }

    @RequestMapping("/admin")
    public ModelAndView adminView()
    {
        int MODE=1; // 1 means view mode
                    // 0 means no data
        Faculty faculty=new Faculty();
        int accessTypeFlag=1 ; //accessTypeFlag=1 means admin
        ModelAndView modelAndView=new ModelAndView();
        List<Faculty> list=facultyService.getAllFaculty();
        if(list.isEmpty()) MODE=0;
        modelAndView.addObject("listOfModel",list);
        modelAndView.addObject("model",faculty);
        modelAndView.addObject("MODE",MODE);
        modelAndView.addObject("accessTypeFlag",accessTypeFlag);

        modelAndView.setViewName("faculty");
        return modelAndView;
    }


@RequestMapping(value = "/admin/save",method = RequestMethod.POST)
public ModelAndView save(@ModelAttribute("model") Faculty faculty)
{
    facultyService.saveOrUpdate(faculty);
```

/faculty/user

int MODE = 1
if (list.isEmpty) mode=0
mv.addObject("MODE",mode)

/faculty/admin

model - faculty obj
list - faculty list

```
        return adminView();
    }

    @RequestMapping(value = "/admin/remove/{modelId}",method = RequestMethod.GET)
    public ModelAndView save(@RequestParam("modelId") String modelId)
    {
        facultyService.removeFaculty(modelId);

        return new ModelAndView("redirect:/faculty/admin");
    }

    @RequestMapping(value = "/admin/edit/{modelId}",method = RequestMethod.GET)
    public ModelAndView edit(@RequestParam("modelId") String modelId)
    {
        int MODE=2; //MODE=2 means edit/update mode
        Faculty faculty=facultyService.getFacultyById(modelId);

        int accessTypeFlag=1 ; //accessTypeFlag=1 means admin
        ModelAndView modelAndView=new ModelAndView();

        //modelAndView.addObject("listOfModel",faculty);
        modelAndView.addObject("model",faculty);
        modelAndView.addObject("accessTypeFlag",accessTypeFlag);
        modelAndView.addObject("MODE",MODE);
        modelAndView.setViewName("faculty");
        return modelAndView;

    }


}
```

Department Controller

```
package edu.cseju.onetomany.controller;

import edu.cseju.onetomany.model.Department;
import edu.cseju.onetomany.model.Faculty;
import edu.cseju.onetomany.service.DeptService;
import edu.cseju.onetomany.service.FacultyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import java.util.List;

@Controller
@RequestMapping("/department")
public class DepartmentController {


    @Autowired
    private FacultyService facultyService;

    @Autowired
    private DeptService deptService;


    @RequestMapping("/user")
    public ModelAndView userView()
```

```
{
    int accessTypeFlag=0; //accessTypeFlag=0 means user access
    ModelAndView modelAndView=new ModelAndView();
    List<Department> list=deptService.getAllDepartment();

    modelAndView.addObject("listOfModel",list);
    modelAndView.addObject("accessTypeFlag",accessTypeFlag);

    modelAndView.setViewName("department");
    return modelAndView;
}

@RequestMapping("/admin")
public ModelAndView adminView()
{
    int MODE=1; // 1 means view mode
    // 0 means no data
    Department department=new Department();
    int accessTypeFlag=1 ; //accessTypeFlag=1 means admin
    ModelAndView modelAndView=new ModelAndView();

    List<Department> list=deptService.getAllDepartment();
    List<Faculty> fList=facultyService.getAllFaculty();

    if(list.isEmpty()) MODE=0;

    modelAndView.addObject("listOfModel",list);
    modelAndView.addObject("fList",fList);
    modelAndView.addObject("model",department);
    modelAndView.addObject("MODE",MODE);
    modelAndView.addObject("accessTypeFlag",accessTypeFlag);

    modelAndView.setViewName("department");
    return modelAndView;
}


@RequestMapping(value = "/admin/save",method = RequestMethod.POST)
public ModelAndView save(@ModelAttribute("model") Department department)
{
    System.out.println(department.toString());
    Faculty faculty=new Faculty();

faculty=facultyService.getFacultyById(department.getFaculty().getFacultyId());

    department.setFaculty(faculty);
    deptService.saveOrUpdate(department);

    return adminView();
}

@RequestMapping(value = "/admin/remove/{modelId}",method = RequestMethod.GET)
public ModelAndView save(@RequestParam("modelId") String modelId)
{
    deptService.removeDepartment(modelId);

    return new ModelAndView("redirect:/department/admin");
}

@RequestMapping(value = "/admin/edit/{modelId}",method = RequestMethod.GET)
public ModelAndView edit(@RequestParam("modelId") String modelId)
{
    int MODE=2; //MODE=2 means edit/update mode
```

int mode=1
if (list.isEmpty)
mv.addObject("mode",--)

@pathvariable

remove

```
        Department model=deptService.getDepartmentById(modelId);

        int accessTypeFlag=1 ; //accessTypeFlag=1 means admin
        ModelAndView modelAndView=new ModelAndView();
        List<Faculty> fList=facultyService.getAllFaculty();

        modelAndView.addObject("fList",fList);
        modelAndView.addObject("model",model);
        modelAndView.addObject("accessTypeFlag",accessTypeFlag);
        modelAndView.addObject("MODE",MODE);
        modelAndView.setViewName("department");
        return modelAndView;

    }

}
```

Step 8. View layer

~~Faculty.html~~ faculty.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Faculties</title>
</head>
<body>

<div th:if="${accessTypeFlag}==1">
    <form th:object="${model}" th:action="@{/faculty/admin/save}" method="post">
        <table>
            <h2 th:if="${MODE}!=2">Add Faculty</h2>
            <h2 th:if="${MODE}==2">Edit Faculty</h2>
            <tbody>
            <tr>
                <td>Enter Faculty ID:</td>
                <td><input type="text" th:field="*{facultyId}"
th:value="${model.facultyId}"  placeholder="Enter faculty id"></td>
            </tr>
            <tr>
                <td>Enter Faculty Name:</td>
                <td><input type="text" th:field="*{facultyName}"
th:value="${model.facultyName}" placeholder="Enter faculty Name"></td>
            </tr>
            <tr>
                <td>Enter Faculty Code: </td>
                <td><input type="text" th:field="*{facultyCode}"
th:value="${model.facultyCode}" placeholder="Enter faculty code"></td>
            </tr>

            </tbody>

        </table>

        <input type="submit" value="Save">
```

→ Faculty ~~list~~ object

```
    </form>
</div>
<br>


<div th:if="${MODE}==1">
    <h2> List of Faculty</h2>


    <table th:object="${listOfModel}" border="1">

        <thead>
        <tr>
            <td><strong>Faculty Id</strong></td>
            <td><strong>Faculty Name</strong></td>
            <td><strong>Code</strong></td>
            <td th:if="${accessTypeFlag}==1"><strong>Action</strong></td>
        </tr>
        </thead>
        <tbody>
        <tr th:each="f: ${listOfModel}">
            <td th:text="${f.facultyId}" th:value="${f.facultyId}"></td>
            <td th:text="${f.facultyName}" th:value="${f.facultyName}"></td>
            <td th:text="${f.facultyCode}" th:value="${f.facultyCode}"></td>
            <td th:if="${accessTypeFlag}==1">
                <span><a role="button" th:href="@{'/faculty/admin/edit/'+
${f.facultyId}(modelId=${f.facultyId})}">Edit</a></span>
                <span><a role="button" th:href="@{'/faculty/admin/remove/'+
${f.facultyId}(modelId=${f.facultyId})}">Remove</a></span>
            </td>
        </tr>
        </tbody>
    </table>
</div>



</body>
</html>
```

Faculty list

department.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Departments</title>
</head>
<body>


<div th:if="${accessTypeFlag}==1">
    <form th:object="${model}" th:action="@{/department/admin/save}"
method="post">
        <table>
```

```
                <h2 th:if="${MODE}!=2">Add Department</h2>
                <h2 th:if="${MODE}==2">Edit Department</h2>
                <tbody>
                <tr>
                    <td>Enter Department ID:</td>
                    <td><input type="text" th:field="*{deptId}"
   th:value="${model.deptId}"  placeholder="Enter dept id"></td>
                </tr>
                <tr>
                    <td>Enter Department Name:</td>
                    <td><input type="text" th:field="*{deptName}"
   th:value="${model.deptName}" placeholder="Enter Dept Name"></td>
                </tr>
                <tr>
                    <td>Enter Department Code: </td>
                    <td><input type="text" th:field="*{deptCode}"
   th:value="${model.deptCode}" placeholder="Enter dept code"></td>
                </tr>
                <tr>
                    <td>Enter Faculty: </td>
                    <td><select th:field="*{faculty.facultyId}">
                        <option th:each="f:${fList}" th:text="${f.facultyName}"
   th:value="${f.facultyId}"></option>
                    </select>

                </tr>

                </tbody>

            </table>

            <input type="submit" value="Save">


        </form>
    </div>
    <br>


    <div th:if="${MODE}==1">
        <h2> List of Department</h2>


        <table th:object="${listOfModel}" border="1">

            <thead>
            <tr>
                <td><strong>Dept Id</strong></td>
                <td><strong>Dept Name</strong></td>
                <td><strong>Code</strong></td>
                <td th:if="${accessTypeFlag}==1"><strong>Action</strong></td>
            </tr>
            </thead>
            <tbody>
            <tr th:each="f: ${listOfModel}">
                <td th:text="${f.deptId}" th:value="${f.deptId}"></td>
                <td th:text="${f.deptName}" th:value="${f.deptName}"></td>
                <td th:text="${f.deptCode}" th:value="${f.deptCode}"></td>
                <td th:if="${accessTypeFlag}==1">
                    <span><a role="button" th:href="@{'/department/admin/edit/'+
${f.deptId}(modelId=${f.deptId})}">Edit</a></span>
                    <span><a role="button" th:href="@{'/department/admin/remove/'+
${f.deptId}(modelId=${f.deptId})}">Remove</a></span>
                </td>
```

<td> faculty </td>

<td th:text = "${f.getFaculty().getFacultyName()}"
th:value = "${f.getFaculty().getFacultyName()}" ></td>

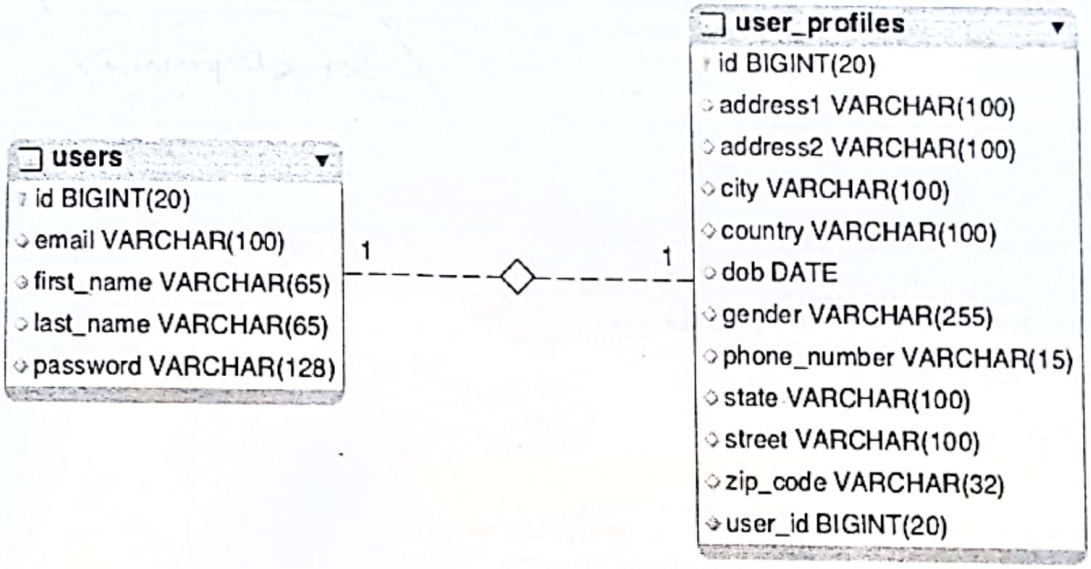
```
        </tr>
        </tbody>
    </table>
</div>


</body>
</html>
```

One to One

Let's consider an application that stores a lot of information about its users so that it can provide a personalized experience to them.

In such cases, it makes sense to store user's primary details like name, email, password in a `USERS` table and store all the other secondary details in a separate table called `USER_PROFILES`, and have a one-to-one relationship between `USERS` and `USER_PROFILES` table.

Following is a visual of such a database schema -



The `users` and `user_profiles` tables exhibit a one-to-one relationship between each other. The relationship is mapped using a foreign key called `user_id` in the `user_profiles` table.

In this article, we'll create a project from scratch and learn how to go about implementing such one-to-one relationship at the object level using JPA and Hibernate.

User Entity

```
package com.example.jpa.model;

import javax.persistence.*;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.validation.constraints.Email;
import java.io.Serializable;

@Entity
@Table(name = "users")
public class User implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @Size(max = 65)
    @Column(name = "first_name")
    private String firstName;
```

User class

```
    @Size(max = 65)
    @Column(name = "last_name")
    private String lastName;

    @NotNull
    @Email
    @Size(max = 100)
    @Column(unique = true)
    private String email;

    @NotNull
    @Size(max = 128)
    private String password;

    @OneToOne(fetch = FetchType.LAZY,
              cascade = CascadeType.ALL,
              mappedBy = "user")
    private UserProfile userProfile;
```

@One to Many

Set <Department>

UserProfile Entity

```
package com.example.jpa.model;

import javax.persistence.*;
import javax.validation.constraints.Size;
import java.io.Serializable;
import java.util.Date;

@Entity
@Table(name = "user_profiles")
public class UserProfile implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "phone_number")
    @Size(max = 15)
    private String phoneNumber;

    @Enumerated(EnumType.STRING)
    @Column(length = 10)
    private Gender gender;

    @Temporal(TemporalType.DATE)
    @Column(name = "dob")
    private Date dateOfBirth;

    @Size(max = 100)
    private String address1;

    @Size(max = 100)
    private String address2;

    @Size(max = 100)
    private String street;

    @Size(max = 100)
    private String city;

    @Size(max = 100)
    private String state;

    @Size(max = 100)
    private String country;
```

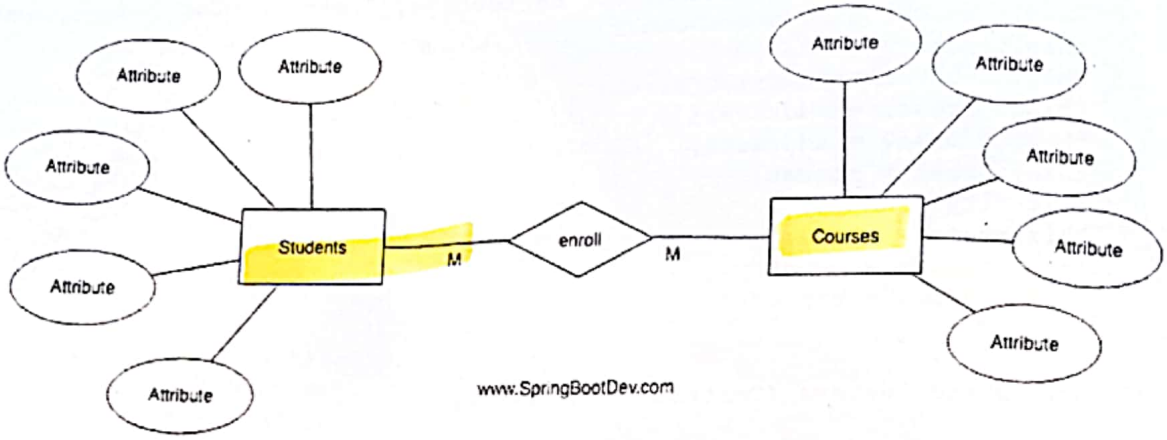
```
    @Column(name = "zip_code")
    @Size(max = 32)
    private String zipCode;

    @OneToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

    public UserProfile() {

    }

    public UserProfile(String phoneNumber, Gender gender, Date dateOfBirth,
                       String address1, String address2, String street, String city,
                       String state, String country, String zipCode) {
        this.phoneNumber = phoneNumber;
        this.gender = gender;
        this.dateOfBirth = dateOfBirth;
        this.address1 = address1;
        this.address2 = address2;
        this.street = street;
        this.city = city;
        this.state = state;
        this.country = country;
        this.zipCode = zipCode;
    }

    // Getters and Setters (Omitted for brevity)
}
```

User profile

Many to Many

In this article, i am going to discuss how to map **many to many** relationship using **@ManyToMany** annotation available in **JPA** (Java Persistence Api).

Lets look at the following ER-Diagram.

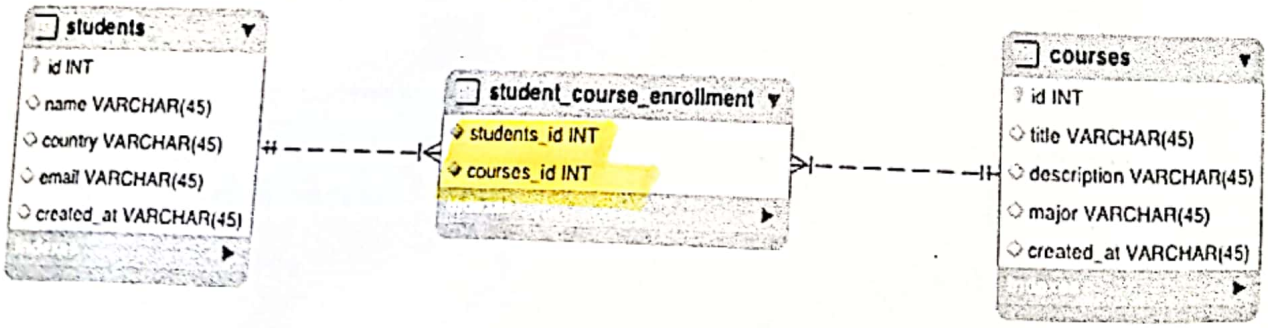


This ER-Diagram describes the relationship between **Students** and **Courses**. According to the diagram, the relationship can be described as below.

- A student can enroll for many courses.
- A course can be enrolled by many students.

Therefore the **relationship** between Students and Courses is **many to many**.

If we convert this relationship into **relational database model**, the table structure should looks like below.



Student Class

```
import javax.persistence.*;
import java.io.Serializable;
import java.util.List;

@Entity
@Table(name = "students")
public class Student implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;

    private String country;

    private String email;

    @Column(name = "created_at")
    private String createdAt;

    @ManyToMany
    @JoinTable(
            name = "student_courses",
            joinColumns = {@JoinColumn(name = "student_id", referencedColumnName =
"id")},
            inverseJoinColumns = {@JoinColumn(name = "course_id",
referencedColumnName = "id")}
    )
    private List<Course> courses;

    //TODO add getters and setters for all properties
}
```

In Student class we are going the map the relationship from **Student** to **Course**. As you are already aware, it is a many to many relationship and there should be an intermediate table (**joined table**) to map the relationship. Therefore we have used the **@JoinTable** annotation to define the joined/intermediate table with its properties.

The name of the joined table (newly created intermediate table) will be "**student_courses**" as declared. In addition, there will be two join columns.

- **student_id** – This will refer the "**id**" column of the "**students**" table.
- **course_id** – This will refer the "**id**" column of the "**courses**" table.

Here is the **Course** class

```
import javax.persistence.*;
import java.io.Serializable;
import java.util.List;

@Entity
@Table(name = "courses")
public class Course implements Serializable {
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String title;

    private String description;

    private String major;

    @Column(name = "created_at")
    private String createdAt;

    @ManyToMany(mappedBy = "courses")
    private List<Student> students;

    //TODO getters and setters for all properties
}
```

In Course class, we just declare only the relationship from **Course** to **Student**. Since the relationship is **many to many,** it is just declared with **@ManyToMany** annotation. We are not going to map the relationship as it is already mapped by **Student** class and it is denoted with **mappedBy** attribute.

What is "mappedBy" attribute here?

mappedBy attributes defines the owner of the relationship. As you can see that relationship has been defined and owned by the **courses** property of the **Student** class. Therefore in the Course class it is declared with **mappedBy="courses"**.

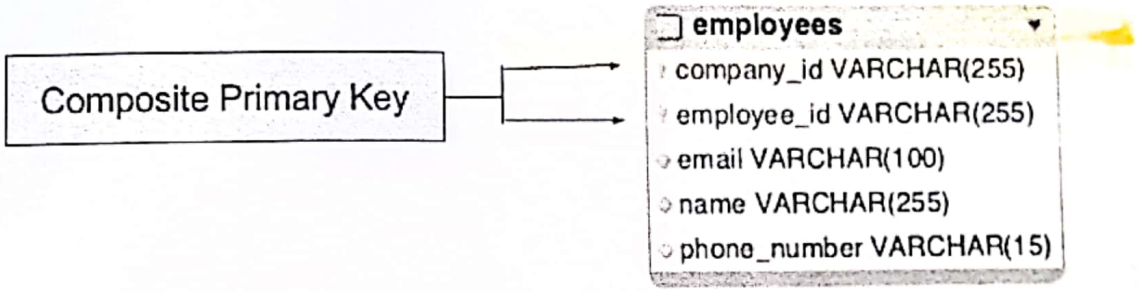
This will tells that the relationship has been mapped and owned by the **courses** property of the **Student** class.

Composite primary key in Hibernate

In this article, You'll learn how to map a composite primary key in Hibernate using JPA's `@Embeddable` and `@EmbeddedId` annotations.

Let's say that We have an application that manages Employees of various companies. Every employee has a unique employeeId within his company. But the same employeeId can be present in other companies as well. So we can not uniquely identity an employee just by his employeeId.

To identify an employee uniquely, we need to know his employeeId and companyId both. Check out the following `Employees` table that contains a composite primary key which includes both the employeeId and companyId columns -



Defining the Domain model

A composite primary key is mapped using an Embeddable type in hibernate. We'll first create an Embeddable type called `EmployeeIdentity` containing the employeeId and companyId fields, and then create the `Employee` entity which will embed the `EmployeeIdentity` type.

Create a new package named `model` inside `com.example.jpa` package and then add the following classes inside the `model` package -

1. EmployeeIdentity - Embeddable Type

```
package com.example.jpa.model;

import javax.persistence.Embeddable;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import java.io.Serializable;

@Embeddable
public class EmployeeIdentity implements Serializable {
    @NotNull
    @Size(max = 20)
    private String employeeId;

    @NotNull
    @Size(max = 20)
    private String companyId;
```

2. Employee - Domain model

```
package com.example.jpa.model;

import org.hibernate.annotations.NaturalId;
import javax.persistence.Column;
import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Email;
import javax.validation.constraints.Size;

@Entity
@Table(name = "employees")
public class Employee {

    @EmbeddedId
    private EmployeeIdentity employeeIdentity;

    @NotNull
    @Size(max = 60)
    private String name;

    @NaturalId
    @NotNull
    @Email
    @Size(max = 60)
    private String email;

    @Size(max = 15)
    @Column(name = "phone_number", unique = true)
    private String phoneNumber;

}
```