=pojo :

baseentity

railway

station

dao

i/f

dto

service

i/f

modelmapper autowire

impl

custom exc

global Exception handler

controller

1.serviceDao autowire

method call using Service

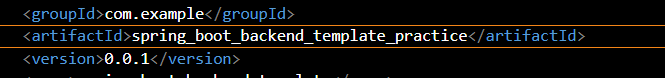
2.dao autowire

using dao method call and return back

Copy template project

Rename as practice

Open in vs code and rename it in artifact idIn pom.xml



Import that project

Force update

Import if error in application.java

In **resource application.properties**

Change password or

Get Caused by: javax.persistence.PersistenceException: [PersistenceUnit: default] Unable to build Hibernate SessionFactory; nested exception is org.hibernate.exception.GenericJDBCException: Unable to open JDBC Connection for DDL execution

Make package **pojo**

In that make **base entity** class and id field

@MappedSuperclass: Means it's not an actual table in the database but a blueprint for other tables.

Megst

Ig

@MappedSuperclass

@Entity dont take entity here as this is not table

@Getter

@Setter

@ToString

public class BaseEntity {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

//type field nam efirst then add annotations for intelisance

In class **railway**

1)Egst t n

@ToString(callSuper = true,exclude = {"clist"})

considering the superclass and excluding a specific field (clist).

@Entity

@Getter

@Setter

@ToString(callSuper = true,exclude = {"clist"})

@Table(name = "railways")

@NoArgsConstructor

public class Railway extends BaseEntity {

2)extends BaseEntity dont forget this or else eroor

For fielsd @column(length=20,nullable =false

@Enumerated(EnymType.String)

public class Railway extends BaseEntity {

@Column(length=20,nullable=false)

private String name;

@Enumerated(EnumType.STRING)

private Category category;

@Column(nullable = false)

private LocalDate startTime;

@Column(nullable = false)

private LocalDate endtime;

@Column(length=20,nullable=false)

private String source ;

@Column(length=20,nullable=false)

private String destination;

3)

If you make changes to the train (Rail), like adding, updating, or deleting, those changes will also apply to the associated stations (Station). If a station is removed from the train's list of stations (clist), it will also be deleted to keep things clean.

train can have many stations, and managing them is made easier with these annotations and the clist list.

private List<Station> clist = new ArrayList<>();: Creates a list named clist to store the associated Station objects.

* mappedBy = "rail": Means the relationship is managed by the "rail" field in the Station class.
* cascade = CascadeType.ALL: Ensures that changes to Rail (like adding, updating, or deleting) are applied to the associated Station objects.
* orphanRemoval = true: Means if a Station is removed from the clist, it will be deleted from the database to keep things tidy.

@OneToMany(mappedBy = "rails" ,orphanRemoval = true,cascade = CascadeType.ALL)

private List<Station> list=new ArrayList<>;

Dont forget that we should add field name not database column name when adding mappedby value

In Station class

No tostring super rest same

Extends base entity

NO NEED TO TAKE ID IN STATION AND RAILWAY AS TAKEN IN BASE ENTITY

2)@ManyToOne(fetch = FetchType.LAZY)

many stations (Station) can be linked to one railway (Railway)

Fetch only when req

@JoinColumn(name = "railway\_id", nullable = false)

In station table railawy\_id column connect to railway table i.e f.k And it's required (nullable = false)

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name="railway\_id",nullable=false)

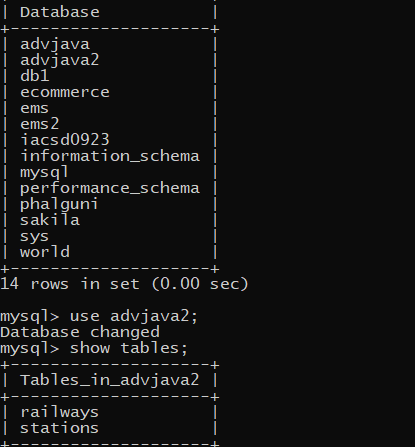
private int railwayId;

Private Railway railwayId;

**Run spring boot**

**Open database**

U will see db and tables created with name in application property

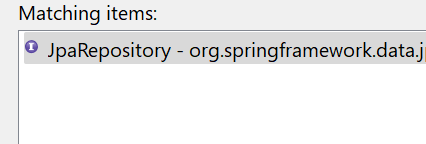
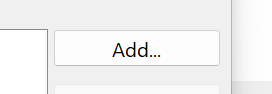


**In dao package**

**In railwaydao and station dao**

Dont make it class but interface

And extend fron jparepository



Make t id as <Railway, Long>

In dto package

Make class

Add fiel name from pojo

Gsn

@NotBlank: no blank (null or whitespace). It's used for validation.

On name ,destination

Dont add list

@Getter

@Setter

public class RailDto {

@NotBlank

private String name;

private Category category;

private LocalDate startTime;

private LocalDate endtime;

private String source ;

@NotBlank

private String destination;

}

Dont forget getter setter in dto or else get empty schema in swagger

In **stationdto**

gs

Add id too

@notblank on code and name

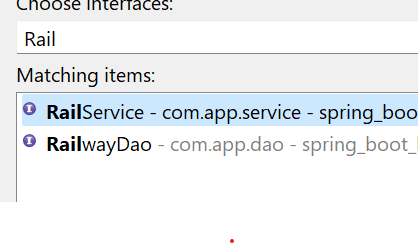
Include railayId too

In **service package**

Make interface not class **railSevice,StationService i/f**

And its **impl class**

Make impl class by add



1)@Service: Marks class as a service component

@Transactional: methos in class execute as single unit

@Service

@Transactional

public class RailServiceImpl implements RailService {

2)depcy(both dao modelmapper )

@autowire and field with class

3)In r**ailService**

Initaialize method to get list of railways in db

List<RailDto> railwayList();

Implement it in **railServiceImpl**

Dfs mc

a)rdao.findAll() -get all railways from the database.

b).straem()-Applies a stream to the result,

c).map -convert one type of obj to other

i/P obj and class of obj u want to convert to

o/p obj u want to conv to

DestType destObject = modelMapper.map(srcObject, DestType.class);

Here dao to dto

Rdao to RailwayDto.class

* collect(Collectors.toList()Collects the converted DTOs into a list.

@Override

public List<RailDto> railwayList() {

return rdao.findAll()

.stream()

.map((s)->mm.map(s,RailDto.class))

.collect(Collectors.toList());

}

Do we need to run after dao

Dto or service

Shortcut to make dto

Why not add list in dto

Which fields to take and which not

Which in/f which class

In **controller** package

Make **railContoler** and **Stationcontroller**

Rr n

1)@RestController:

* class is a controller, and the methods return the response directly to the client (rather than relying on a view resolver).

@RequestMapping("/railway"):remember to addd/

* if the application is running at http://localhost:8080, this controller would handle requests at http://localhost:8080/railway.

@NoArgsConstructor:

2)depcy

@Autowired

private RailService rs;

3)method

@GetMapping:

* Handles HTTP GET requests for the specified URL (/railway in this case).

No this @GetMapping() i.e ()

ResponseEntity in class represents everything that goes back to the client in an HTTP response—whether it's the status code (like 200 OK), any headers, or the actual data (body) being sent back

The generic type (<?>) indicates that the body can be of any type.

* ResponseEntity.status(HttpStatus.OK) is used to set a specific HTTP status code (in this case, 200 OK). You can replace OK with other HTTP status codes like BAD\_REQUEST, NOT\_FOUND, etc.
* .body(/\* your data \*/): This is where you provide the actual data (body) that you want to send in the response.
* ResponseEntity.ok(/\* your data \*/): If you're returning a 200 OK response and don't need to set a specific status code, you can use the ok() factory method for simplicity.

Remember to replace /\* your data \*/ with the actual data you want to send back in the response. This could be a string, an object, or any other type depending on your use case.

Responseentity.ok(rail.getrailingo())

@GetMapping

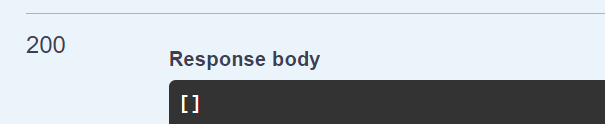
public ResponseEntity<?> getList(){

return ResponseEntity.ok(rs.railwayList());

}

4)run

Open sagger



2) **add railway method**

**a)In interface railService**

String addRail(RailDto rdto); //not railway

**b)In impl**

i/p railDto obj

o/p string msg that obj added

Use map i.e convert dto(IDTO) to obj(r1) and store it in railway class obj

Add that obj in db(rdao)(depcy)

@Override

public String addRail(RailDto rdto) {

Railway r1=mm.map(rdto, Railway.class);

rdao.save(r1);

return "railway is added";

}

* Mapping:
  + Uses ModelMapper to convert a RailwayDTO object (r) to a Railway object (r1).
* Database Save:
  + Saves the mapped Railway object (r1) to the database using rdao.save(r1).
* Return Message:
  + Returns the message "Railway has been added" to indicate the success of the operation.

In a nutshell, this method takes railway information in a DTO, converts it, saves it to the database, and returns a success message.

**c)In controller**

o/p responseentity with ?

i/p @reuestbody @valid and dto

@RequestBody Take the data from the request body, convert it into a RailwayDTO object @Valid RailwayDTO rail1 make sure it follows the validation rules specified in the RailwayDTO class."

@PostMapping("/rail")

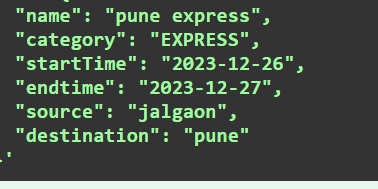
public ResponseEntity<?> postAddRail(@RequestBody@Valid RailDto idto){

return ResponseEntity.ok(rs.addRail(idto));

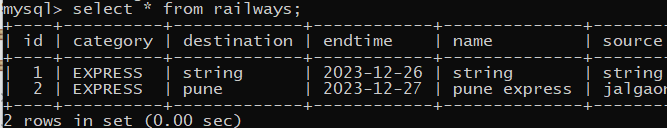
}

d) run on swagger

Put input here and execute



And see in db



U can also see it in swagger get

**3)getById**

**a)in interface**

i/p long id

o/p RailDto(obj from db)

**b)make exception package and resournf class**

1)Extends RuntimeExcaption

2) constructor with data msg

public class ResourceNotFound extends RuntimeException {

public ResourceNotFound(String msg) {

super(msg);

}

**c)in impl**

In db find entity by id or else throw exception in string msg

Sabe answer in return class obj

Convert this class obj in dto

And return it

(atat h dto main jata hai dto main)(pr opr k liye entity main conv karo)

@Override

public RailDto getById(Long id) {

Railway r1=rdao.findById(id).orElseThrow(()-> new ResourceNotFound("rail not found"));

RailDto rdto=mm.map(r1, RailDto.class);

return rdto;

}

**d)in controller**

i/p id variable so in getmapping put it in {} and add @pathvarible

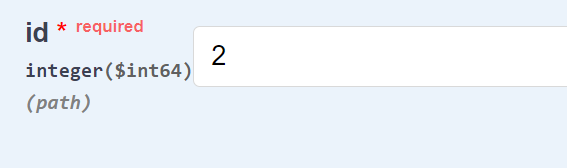
As that id is variable an rest url same

@GetMapping("/{id}")

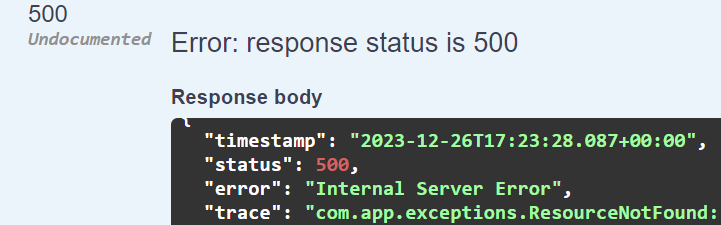
public ResponseEntity<?> getEntity(@PathVariable Long id){

return ResponseEntity.ok(rs.getById(id));

}





If i/p id 6 

**4)deletebyid**

In infterfcaei/p -long id

o/p -String for msg

In impl

**Railway r1=rdao.deleteById(id).orElseThrow(()->new ResourceNotFound("rail not found"));**

rdao.deleteById(r1);

@Override

public String deleteByid(Long id) {

Railway r1=rdao.findById(id).orElseThrow(()->new ResourceNotFound("rail not found"));

rdao.deleteById(id);

return "rail deleted";

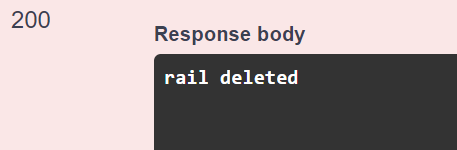
}

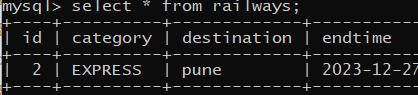
@DeleteMapping("/{id}")

public ResponseEntity<?> deleteEntity(@PathVariable Long id){

return ResponseEntity.ok(rs.deleteByid(id));

}





**Station**

Dont forget

@Service

@Transactional

Or else bean not define error

@Autowired

private StationDao sdao;

@Autowired

private ModelMapper mm;

List<Station> getAll();

@Override

public List<StationDto> getAll() {

// TODO Auto-generated method stub

return sdao.findAll().stream()

.map((s)->mm.map(s,StationDto.class))

.collect(Collectors.toList());

}

@RestController

@RequestMapping("/stations")

public class StationController {

@Autowired

private StationService ss;

**Addstation in fk table**

**In p.k table entity(railway)**

1)add method in pojo

2)in db find that railwayid by

public void addStation(Station s) {

this.list.add(s);

s.setRailwayId(this);

}

public void removeStation(Station s) {

this.list.remove(s);

s.setRailwayId(null);

}

Station s1=mm.map(sdto,Station.class);

sdao.save(s1);

return "added";

sdao.findById(sdto.getRailwayId())

**In foreignKey table to add**

Find by tile implement erro i dao

Course findByTitle(String courseTitle);

1)Find course in db by getter

2) if c1==null or else throe

2)validate -id stud dto has greater marks that corses entiy obj min score

3)conv stud dto to entity

4)add helper method in course

public void addStudents(Student s ) {

list.add(s);

s.setCourseId(this);

}

5)add s1 in c1

6)add s1 in db

7)return string

***Update***

@Override

public String UpdateCourse(Long id, double fees) {

Course c1 =cdao.findById(id).orElseThrow(()->new ResourceNotFound("course Not found"));

c1.setFees(fees);

return "updated " +c1.getTitle()+" to fees "+c1.getFees();

}

@PutMapping("/{id}/{fees}")

public ResponseEntity<?> updateEntity(@PathVariable Long id,@PathVariable double fees){

return ResponseEntity.ok(cs.UpdateCourse(id, fees));

}