This Project has 4 Entity (1) User (2) Category (3) Post (4) Comment

How to Create Entity Relationship?

1. First of all create User entity (id, name, email, password, about). This entity will not contain any code for entity mapping. Hence it is independent entity.
2. After that create Category entity (categoryId, categoryTitle). This is also an independent entity

Now Main Part starts

1. Create Post Entity (postId, title, content, imageUrl, createdAt, User, Category).

Post can not exist without User and Category. Every post should belongs to an User and some Category. One User can do multiple post and one Category can have multiple Post. Hence inside Post class

@ManyToOne //means many post can belong to one user

@JoinColumn(name = "user\_id", nullable = **false**) // give name to that column which is automatically created to contain foreign key relation, this column should not contain null

@OnDelete(action = OnDeleteAction.***CASCADE***)// means if we delete User then all his Post will be automatically deleted.

**private** User user;

@ManyToOne

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

@OnDelete(action = OnDeleteAction.***CASCADE***)

**private** Category category;

1. Create Comment entity (id, content, User, Post).

One Post can have any number of Comments. Comment can not exist without user and post. If user is deleted or post is deleted then all the comments will be automatically deleted. Hence inside Comment class

// -------Relationship with User and Post-------------

@ManyToOne

@JoinColumn(name = "user-id",nullable = **false**)

@OnDelete(action = OnDeleteAction.***CASCADE***)

**public** User user;

@ManyToOne

@JoinColumn(name = "post-id",nullable = **false**) //this field should not null

@OnDelete(action = OnDeleteAction.***CASCADE***)

@JsonIgnore //means this field will not send or receive information while talking to Postman Client. Hence inside service class we need to use PostService/repository to get Post from database and set into this field.

**public** Post post;

Now we need to again modify (3) Post Entity. We need to and new field

@OneToMany(fetch = FetchType.***EAGER***,mappedBy = "post")//one post can have any number of comments hence post class is having List<Comment> , mappedBy field says go to Comment entity, it has field name post which is containing Foreign key to establish relationship.

**private** List<Comment> comments=**new** ArrayList<>();

VVI:

(Q) Do you really understand how @oneToMany and @ManyToOne relationship is established?

Answer: Yes. @OneToMany 🡪 first word represent current class. Second word represent it’s field. Hence here a field should be a Collection.

@ManyToOne 🡪 then field should be a single object. Not collection

@JoinColumn 🡪 It is used to keep foreign key

(Q) @OneToMany(fetch = FetchType.***EAGER***,mappedBy = "post")

**private** List<Comment> comments=**new** ArrayList<>();

Explain everything about it?

Answer: see the above explanation. + understand the Database Table

Post Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Post\_id | content | Created\_at | Image\_url | title | Category\_id | User\_id |
| 101 | Java | 5-5-2020 | Java.jpg | OOP | programming | 1 |

One column can contain only one field/data not a list of data

Comment Table

|  |  |  |  |
| --- | --- | --- | --- |
| id | content | Post-id | User-id |
| 1001 | This is very nice tutorial | 101 | 5 |
| 1005 | I love Java | 101 | 8 |

Post entity is having a field which is List<Comment> comments=**new** ArrayList<>(); but collection can not fit inside a column. It needs many column hence mappedBy = "post" property is saying go to comment entity, it has field name post, that will keep foreign key relationship.

Custom Finder Methods

Optional<User> findByEmail(String email);

Optional<Category> findByCategoryTitle(String categoryTitle);

**boolean** existsByCategoryTitle(String categoryTitle);

List<Post> findByUserId(**long** id);

List<Post> findByCategoryCategoryId(**int** categoryId);

List<Post> findByCategoryCategoryTitle(String title);

Optional<Post> findByUserIdAndPostId(**long** userId,**int** postId);

Optional<Comment> findByUserIdAndId(**long** userId, **int** commentId);