71. Resolver - post.ts / Query - post()

#graphql #resolver #authentication #query #backend #typeorm

Update post() Query in post Resolver

- The graphql schema in post,graphql expects a creator object, but right now our post() query is not returning that, it's just returning the post:

```
@Query(() => Post, { nullable: true })
  post(
    @Arg("id", () => Int) id: number
  ): Promise<Post | undefined> {
    return Post.findOne(id);
  }
```

- So we update the query to return the creator:
- In TypeOrm we can use `{ relations }` object to have it create the join for us
- We named the many-to-one field in the Post entity "creator", so we use that in the `{ relations }`

/resolvers/post.ts

```
@Query(() => Post, { nullable: true })
  post(
    @Arg("id", () => Int) id: number // 'id' is just a name for using in GraphQL schema, id is the actual field in database
  ): Promise<Post | undefined> {
    return Post.findOne(id, { relations: ["creator"] });
  }
```

- Now the SQL query looks like this:

```
SELECT "Post"."id" AS "Post_id", "Post"."title" AS "Post_title", "Post"."text" AS "Post_text", "Post"."points" AS "Post_points", "Post"."creatorId" AS "Post_creatorId", "Post"."createdAt" AS "Post_createdAt", "Post"."updatedAt" AS "Post_updatedAt", "Post__creator"."id" AS "Post__creator_id", "Post__creator"."username" AS "Post__creator_username", "Post__creator"."email" AS "Post__creator_email", "Post__creator"."password" AS "Post__creator_password", "Post__creator"."createdAt" AS "Post__creator_createdAt",
```

```
"Post__creator"."updatedAt" AS "Post__creator_updatedAt" FROM "post" "Post" LEFT
JOIN "user" "Post__creator" ON "Post__creator"."id"="Post"."creatorId" WHERE
"Post"."id" IN ($1) -- PARAMETERS: [300]
```