43. Middleware authentication check - isAuth()

#middleware #authentication #typescript #graphql #backend #error-handling

- We want only authenticated users to be able to create posts. We will write a middleware function isAuth() to check if the user is logged on or not
- The middleware function runs before the resolver. It has access to args, context, info and root
- We can pass MyContext to it so that it knows the type of the context object
- Note that the error that is thrown here does **NOT** end up in response.data.createPost.errors like the errors returned from within the mutation. It ends up in response.error
- See 47. Page - create-post for receiving the errors (`const { error } = await createPost({ input: values })`)

/middleware/isAuth.ts

```
import { MyContext } from "src/types";
import { MiddlewareFn } from "type-graphql";

// MiddlewareFn runs before the resolver
export const isAuth: MiddlewareFn<MyContext> = ({ context }, next) => {
  if (!context.req.session.userId) {
    // if user is not logged in
    throw new Error("not authenticated");
  }

  // if user is logged in continue with resolver
  return next();
};
```

- Then we wrap the createPost() mutation with this middleware as follows:

/resolvers/post.ts

```
@UseMiddleware(isAuth)
@Mutation(() => Post)
async createPost(
  @Arg("input") input: PostInput,
  @Ctx() { req }: MyContext
): Promise<Post> {
   return Post.create({
      ...input,
      creatorId: req.session.userId,
   }).save();
}
```

- We want to handle this case on the front-end so we will first implement a create-post page and then handle this error there