06. 회원 엔티티 작성

1. 엔티티 폴더링

```
src/
|-- entities/
|   |-- user.entity.ts
|-- ...
```

2. 엔티티 작성

```
import { Entity, PrimaryGeneratedColumn, Column } from 'typeorm';

@Entity()
export class User {
  @PrimaryGeneratedColumn()
  id: number;

  @Column({ unique: true })
  username: string;

  @Column({ unique: true })
  email: string;

  @Column()
  password: string;

  @Column()
  role: string; // 'MEMBER' 또는 'ADMIN'
}
```

3. 엔티티 등록

`app.module.ts`

```
// app.module.ts
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { User } from './entities/user.entity'; // User 엔티티 임포트
import { AuthService } from './auth.service';
import { JwtModule } from '@nestjs/jwt';
import { PassportModule } from '@nestjs/passport';
import { JwtStrategy } from './jwt.strategy';

@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: 'mysql',
      host: 'localhost',
      port: 3306,
      username: 'root',
      password: '1234',
      database: 'opstest',
      entities: [User], // User 엔티티 등록
      synchronize: true,
    }),
    PassportModule,
    JwtModule.register({
      secret: '1234',
      signOptions: { expiresIn: '60s' },
    }),
  ],
  controllers: [AppController],
  providers: [AppService, AuthService, JwtStrategy],
})
export class AppModule {}
```

✅ 4. synchronize: true

- 이거 키면 jpa에서 테이블 만드는 거랑 똑같은 옵션임
- 실제 서비스에선 false로 바꿔줘야함

5. 테이블 생성 확인

- jpa에서 테이블 생성하는 것처럼 잘 되는 것을 볼 수 있다.

```
mysql> SELECT * FROM opstest.user;
Empty set (0.01 sec)
```