

## TypeORM and CRUD operations

```
import { todo } from "node:test";
import { Column, Entity, PrimaryGeneratedColumn }
from "typeorm";

@Entity({name: 'todo'})
export class TodoEntity {

    @PrimaryGeneratedColumn()
    id:number

    @Column()
    title:string
}
```

```
import { Injectable, NotFoundException } from
'@nestjs/common';
import { InjectRepository } from
'@nestjs/typeorm';
import { TodoEntity } from
'src/todo/todo/todo/todo.entity';
import { Repository } from 'typeorm';

@Injectable()
export class TodoService {

    constructor(
```

```
    @InjectRepository(TodoEntity) private
readonly todoRepository: Repository<TodoEntity>,
    ) {}

    async create(dto: {title:string}) {
        const todo
=this.todoRepository.create(dto)

        return await
this.todoRepository.save(todo)
    }

    async findOne(id: number): Promise<TodoEntity
| null> {

        return this.todoRepository.findOneBy({ id
});
    }

    async updateFull(
        id: number,
        dto: { title: string },
    ): Promise<TodoEntity> {
        const todo = await
this.todoRepository.preload({ id, ...dto });
        if (!todo) {
            throw new NotFoundException(`Todo #${id}
not found`);
        }
        return this.todoRepository.save(todo);
    }
}
```

```

    }

    async updatePartial(
        id: number,
        dto: Partial<{ title: string }>,
    ): Promise<TodoEntity> {
        const result = await
this.todoRepository.update(id, dto);
        if (result.affected === 0) {
            throw new NotFoundException(`Todo #${id}
not found`);
        }

        const todo = await this.findOne(id);
        if (!todo) {
            throw new NotFoundException(`Todo #${id}
not found after update`);
        }
        return todo;
    }
}

```

```

import { Body, Controller, Get, Param,
ParseIntPipe, Patch, Post, Put } from
'@nestjs/common';
import { TodoService } from '../todo.service';
import { TodoEntity } from
'src/todo/todo/todo/todo.entity';

```

```
@Controller('todo')
export class TodoController {

    constructor(private readonly
todoService:TodoService) {}

    @Post()
    create(@Body() dto) {
        return this.todoService.create(dto)
    }

    @Get('/:id')
    findOne(
        @Param('id', ParseIntPipe) id: number,
    ): Promise<TodoEntity | null> {
        return this.todoService.findOne(id);
    }

    @Put('/:id')
    updateFull(
        @Param('id', ParseIntPipe) id: number,
        @Body()dto: {title:string},
    ): Promise<TodoEntity> {
        return this.todoService.updateFull(id, dto);
    }

    @Patch('/:id')
```

```

    updatePartial(
      @Param('id', ParseIntPipe) id: number,
      @Body() dto:{title:string},
    ): Promise<TodoEntity> {
      return this.todoService.updatePartial(id,
dto);
    }
  }
}

```

```

import { Module } from '@nestjs/common';
import { AppController } from '../app.controller';
import { AppService } from '../app.service';
import { TypeOrmModule } from '@nestjs/typeorm';

import { TodoModule } from
'../todo/todo/todo/todo.module';

@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: 'mysql',
      host: 'localhost',
      port: 3306,
      username: 'root',
      password: 'root123',
      database: 'test',
      autoLoadEntities: true,
      synchronize: true,

```

```

    )) ,
    TodoModule
  ],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}

```

The screenshot shows the VS Code REST Client interface. The active tab is 'localhost:3000/books'. A POST request to 'http://localhost:3000/todo' has been executed. The status is '201 Created', size is '32 Bytes', and time is '50 ms'. The response body is a JSON object: `{ "id": 1, "title": "Buy groceries" }`. The 'Body' tab is selected, showing the request body: `{ "title": "Buy groceries" }`.

## GET

The screenshot shows the VS Code REST Client interface. The active tab is 'localhost:3000/books'. A GET request to 'http://localhost:3000/todo/1' has been executed. The status is '200 OK', size is '32 Bytes', and time is '17 ms'. The response body is a JSON object: `{ "id": 1, "title": "Buy groceries" }`. The 'Body' tab is selected, showing the request body: `{ "title": "Buy groceries" }`.

## PUT

The screenshot shows the VS Code REST Client interface. The top bar has tabs for various TypeScript files. The main editor area is divided into two panes. The left pane shows the request details: the method is **PUT**, the URL is `http://localhost:3000/todo/1`, and the body is a JSON object `{ "title": "Pay bills" }`. The right pane shows the response details: the status is **200 OK**, the size is **28 Bytes**, and the time is **7 ms**. The response body is a JSON object `{ "id": 1, "title": "Pay bills" }`. The bottom status bar shows the **TERMINAL** tab is active.

```
PUT http://localhost:3000/todo/1

{ "title": "Pay bills" }
```

Status: 200 OK Size: 28 Bytes Time: 7 ms

```
{
  "id": 1,
  "title": "Pay bills"
}
```

## Patch

The screenshot shows the VS Code REST Client interface. The top bar has tabs for various TypeScript files. The main editor area is divided into two panes. The left pane shows the request details: the method is **PATCH**, the URL is `http://localhost:3000/todo/1`, and the body is a JSON object `{ "title": "Pay the bills" }`. The right pane shows the response details: the status is **200 OK**, the size is **32 Bytes**, and the time is **7 ms**. The response body is a JSON object `{ "id": 1, "title": "Pay the bills" }`. The bottom status bar shows the **TERMINAL** tab is active.

```
PATCH http://localhost:3000/todo/1

{ "title": "Pay the bills" }
```

Status: 200 OK Size: 32 Bytes Time: 7 ms

```
{
  "id": 1,
  "title": "Pay the bills"
}
```

The screenshot shows the VS Code REST Client interface. The top bar has tabs for various TypeScript files. The main editor area is divided into two panes. The left pane shows the request details: the method is **PATCH**, the URL is `http://localhost:3000/todo/17`, and the body is a JSON object `{ "title": "Pay the bills" }`. The right pane shows the response details: the status is **404 Not Found**, the size is **69 Bytes**, and the time is **5 ms**. The response body is a JSON object `{ "message": "Todo #17 not found", "error": "Not Found", "statusCode": 404 }`. The bottom status bar shows the **TERMINAL** tab is active.

```
PATCH http://localhost:3000/todo/17

{ "title": "Pay the bills" }
```

Status: 404 Not Found Size: 69 Bytes Time: 5 ms

```
{
  "message": "Todo #17 not found",
  "error": "Not Found",
  "statusCode": 404
}
```