In **NestJS**, the main building blocks of an application are **Modules**, **Services**, and **Controllers**. Each has a clear role in the architecture, following the principles of **Separation of Concerns** and **Dependency Injection**.

□ 1. Module

□ What is it?

A **Module** is a class annotated with the `@Module()` decorator. It groups related components like **controllers**, **services**, **providers**, and even other modules.

□ Purpose:

- Organize code into cohesive blocks.
- Define the **scope** of services/controllers.
- Allow **feature-based** or **domain-driven** architecture.

□ Example:

```
// user.module.ts
import { Module } from '@nestjs/common';
import { UserService } from './user.service';
import { UserController } from './user.controller';

@Module({
  controllers: [UserController],
  providers: [UserService],
})
export class UserModule {}
```

□ 2. Service

□ What is it?

A **Service** is a class annotated with the `@Injectable()` decorator. It contains **business logic** and can be injected into controllers or other services.

□ Purpose:

- Handle data access and business rules.
- Reusable logic (e.g., fetching from a database, performing calculations, etc).

□ Example:

```
// user.service.ts
import { Injectable } from '@nestjs/common';

@Injectable()
export class UserService {
  private users = [{ id: 1, name: 'Alice' }];

  findAll() {
    return this.users;
  }

  findOne(id: number) {
    return this.users.find(user => user.id === id);
  }
}
```

□ 3. Controller

□ What is it?

A **Controller** is a class annotated with the `@Controller()` decorator. It handles **incoming HTTP requests**, delegates tasks to services, and returns responses.

□ Purpose:

- Map **routes** to functions.
- Receive and respond to HTTP requests (GET, POST, etc).

□ Example:

```
// user.controller.ts
import { Controller, Get, Param } from '@nestjs/common';
import { UserService } from './user.service';

@Controller('users')
export class UserController {
  constructor(private readonly userService: UserService) {}

  @Get()
  getAllUsers() {
    return this.userService.findAll();
  }

  @Get(':id')
  getUser(@Param('id') id: string) {
    return this.userService.findOne(Number(id));
  }
}
```

□ How They Work Together

```
[Client] --> [Controller] --> [Service] --> [Business Logic/Data]
                                 ⬆□
                            Injected by NestJS
```

Example Flow:

- A client makes a request to `GET /users`.
- Nest routes it to `UserController.getAllUsers()`.
- The controller calls `UserService.findAll()`.
- The service returns the user data.
- The controller sends it back to the client.

□ Summary

Component	Decorator	Role
Module	`@Module()`	Organizes components
Service	`@Injectable()`	Handles business logic
Controller	`@Controller()`	Handles HTTP requests