معماری سه لایه در سی شارپ چیست؟ (Three Layer Architecture) (نسخه PDF)

در این مقاله قصد داریم به یکی از سوالات بسیار رایج برنامه نویس ها که خیلی پرسیده میشه پاسخ بدیم و این سوال درباره معماری سه لایه هست. در این مقاله ابتدا توضیحاتی در مورد مفاهیم داده و بعد با یک مثال خیلی ساده با لایه بندی نرم افزار آشنا خواهیم شد.

لایه بندی چیست؟

لایه بندی یعنی تقسیم کردن اجزای برنامه. برای مثال، زمانی که شما برنامه ای می نویسید ممکنه این برنامه شامل اجزای مختلفی باشه، مثل بخشی که اطلاعات به کاربر نمایش داده میشه، بخش ارتباط با بانک اطلاعاتی، بخش سرویس ها و غیره. در بحث لایه بندی شما این بخش ها رو از هم جدا می کنید و اصطلاحاً گفته میشه که نرم افزار نوشته شده لایه بندی شده. لایه بندی باعث میشه که کار Maintenance یا نگه داری برنامه برای شما راحت تر بشه و در زمان ایجاد تغییر نیازی نباشه که کل برنامه رو تغییر بدید و تنها بخشی که مشکل داره رو به روز رسانی کنید.

تفاوت Layer و Tier

زمانی که شما قصد جداکردن اجزای برنامه رو دارید، با دو مفهوم روبرو می شید. Layer و Tier. اما این دو واژه چه تفاوت هایی با هم دارن؟ اگر شما اجزای برنامه رو در قالب فایل های DLL جداگانه از هم جدا کنید (جداسازی فیزیکی یا Physical separation)، شما یک Tier ایجاد کردید:

همانطور که در تصویر بالا مشاهده می کنید، ارتباط مستقیم میان لایه Presentation و لایه Data وجود ندارد و این ارتباط به واسطه لایه Business انجام می شود.اما اگر جداسازی در قالب Namepsace ها و Class ها انجام شود (جداسازی منطقی یا Logical separation)، شما یک Layer ایجاد کردید که می توانید در تصویر این جداسازی را مشاهده کنید:

با توجه به مباحث بالا، می توانیم از تعاریف زیر برای هر Layer یا Tier استفاده کنیم:

۱. Presentation: وظیفه این قسمت نمایش اطلاعات به کاربر هست که می تواند به صورت برنامه تحت وب، ویندوز یا موبایل باشد.
۲. Business: در این لایه کلیه محاسبات و منطق برنامه پیاده سازی می شود. مواردی مثل گرفتن اطلاعات از بانک اطلاعاتی، انجام تغییرات یا محاسبات بر روی داده ها و ارسال اطلاعات به بخش نمایش در این لایه انجام می شود.
۳. Data Access: کلیه ارتباطات برنامه با بانک اطلاعاتی در این بخش انجام می شود.

یک مثال ساده در سی شارپ

در این بخش با یک مثال ساده، به بررسی لایه بندی برنامه ها در زبان سی شارپ می پردازیم. دقت کنید که در اینجا ما از مفهوم Layer استفاده می کنیم. برای پیاده سازی این بخش یک پروژه جدید از نوع Windows Application با نام CSharpLayered ایجاد می کنیم. داخل این پروژه سه پوشه با نام های Business و Data و Forms ایجاد می کنیم. فایل Form۱ رو به MainForm تغییر داده و این فایل رو به پوشه Forms منتقل می کنیم. ساختار پروژه باید به صورت زیر بشه:

در قـدم بعـدی بایـد برای ارتباط با بانک اطلاعاتی کـدهای مربوط رو بنویسـیم. برای اینکار از Entity Framework اسـتفاده می کنیم. ابتدا بوسیله دستور زیر Entity Framework رو به پروژه اضافه می کنیم:

```
install-package EntityFramework
```

و کلاس های زیر رو داخل پوشه Data تعریف می کنیم:

```
public class ShopDbContext : DbContext
{
    public DbSet Customers { get; set; }
}
public class Customer
{
    public long Id { get; set; }
    public string Firstname { get; set; }
    public string Lastname { get; set; }
    public DateTime AddDate { get; set; }
    public string Address { get; set; }
}
```

بعد از تعریف کلاس های بالا، باید لایه Business که وظیفه اصـلی اون گرفتن اطلاعات و انتقال اون به سمت لایه Presentation هست رو ایجاد می کنیم. در قـدم اول کلاسـی ایجاد می کنیم که جدای Entity تعریف شده در قسمت Data هست و تنهـا وظیفه انتقال اطلاعات از لایه Data به Presentation رو داره (اصطلاحاً به این کلاس DTO یا Data Transfer Object گفته میشه). برای Customer کلاس رو به صورت زیر تعریف می کنیم:

```
public class CustomerDTO
{
    public long Id { get; set; }
    public string Firstname { get; set; }
    public string Lastname { get; set; }
    public DateTime AddDate { get; set; }
    public string Address { get; set; }
}
```

قدم بعدی تعریف کلاسی هست که وظیفه ارتباط با دیتابیس و گرفتن اطلاعات و انتقال این اطلاعات به لایه Business رو داره. یک کلاس با نام CustomersRepository به صورت زیر تعریف می کنیم:

```
public class CustomersRepository
{
```

```
{
    private ShopDbContext dbContext;
    public CustomersRepository()
    {
        this.dbContext = new ShopDbContext();
    }
    public CustomerDTO GetByKey(long id)
    {
        var entity = dbContext.Customers.Find(id);
        if (entity == null)
            return null;
        return new CustomerDTO()
        {
            Id = entity.Id,
            AddDate = entity.AddDate,
            Address = entity.Address,
            Firstname = entity.Firstname,
            Lastname = entity.Lastname
        };
    }
    public IQueryable All
    {
        get
        {
            return dbContext.Customers.Select(c => new CustomerDTO()
            {
                Id = c.Id,
                Lastname = c.Lastname,
                Address = c.Address,
                AddDate = c.AddDate,
                Firstname = c.Firstname
            });
        }
    }
    public CustomerDTO Add(string firstname, string lastname, string address)
    {
        var customer = new Data.Customer()
        {
            Firstname = firstname,
            Lastname = lastname,
            Address = address,
            AddDate = DateTime.Now
        };
        dbContext.Customers.Add(customer);
```

```
        dbContext.SaveChanges();
        return new CustomerDTO()
        {
            Id = customer.Id,
            Firstname = firstname,
            Lastname = lastname,
            Address = address,
            AddDate = customer.AddDate
        };
    }
}
```

این یک کلاس ساده هست و باید بر اساس نیاز آیتم های دیگه ای بهش اضافه بشه، مثل عملیات Delete. همچنین ما از IoC و DI در این کد استفاده نکردیم که بهتره از ابزارهایی مثل StructureMap استفاده بشه. برای این مورد می تونید از دوره پیشرفته سی شارپ که مفاهیم مربوطه توضیح داده شده استفاده کنید.

بعـد از اینجاد کلاس Repository داخل لایه Presentation که همون Form های ما هسـتند از کلاس Repository اسـتفاده می کنیم. برای اینکار یک Grid داخل MainForm اضافه کرده و کد زیر رو می نویسیم:

```
public partial class MainForm : Form
{
    public MainForm()
    {
        InitializeComponent();
        var repo = new Business.CustomersRepository();
        CustomersGrid.DataSource = repo.All.ToList();
        CustomersGrid.ResetBindings();
    }
}
```

کـدی که در بالا نوشـته شد تنها بخش مربوط به بارگذاری و نمایش اطلاعات مشتری رو شامل میشه. اما مفهوم کلی به این صورت شد که هر لایه که در اینجا ما از پوشه بندی و Namespace های جداگانه استفاده کردیم وظیفه مربوط به خودش رو انجام میده. لایه Business از لایه Data و لایه Presentation از لایه Business اسـتفاده می کنه و ارتباط مستقیم بین لایه Data و Presentation وجود نداره. اگر بخوایید به جای Layer از Tier اسـتفاده کنیـد، کافیه کـدهای مربوط به هر پوشه رو داخل یک پروژه جداگانه بنویسـید و بین پروژه ها Reference ایجاد کنید.

نویسنده: حسین احمدی

منبع: جزیره برنامه نویسی وب سایت توسینسو

مطلب اصلی