عنوان: **اصول طراحی شی گرا SOLID - #بخش دوم اصل OCP**
نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۷/۰۶ ۲۰:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: OOP, SOLID Principals, Object Oriented Design, Open Closed

در قسمت قبل در مورد اصل Single responsibility Principle یا به اختصار SRP صحبت شد. در این قسمت قصد داریم اصل دوم از اصول SOLID را مورد بررسی قرار دهیم.

O - OCP – Open Close Principle (2 اصل

فرض میکنیم که شما میخواهید یک طبقه بین طبقه‌ی اول و دوم خانه‌ی 2 طبقه‌ی خود اضافه کنید. فکرمیکنید امکان پذیر است؟

راه حل هایی که ممکن است به ذهن شما خطور کنند :

-1 زمانی که برای اولین بار در حال ساخت خانه هستید آن را 3 طبقه بسازید و طبقه‌ی وسط را خالی نگه دارید.اینطوری هر زمان که شما بخواهید میتوانید از آن استفاده کنید. به هر حال این هم یک راه حل است.

-2 خراب کردن طبقه دوم و ساخت دو طبقه‌ی جدید که خوب اصلا معقول نیست.

کد زیر را مشاهده کنید :

```
public class EmployeeDB
{
    public void Insert(Employee e)
    {
        //Database Logic written here
    }
    public Employee Select()
    {
        //Database Logic written here
    }
}
```

متد Select در کلاس EmployeeDB توسط کاربران و مدیر مورد استفاده قرار میگیرد. در این بین ممکن است مدیر نیاز داشته باشد تغییراتی را در آن انجام دهد. اگر مدیر این کار را برای برآورده کردن نیاز خود انجام دهد ،روی دیگر قسمت‌ها نیز تاثیر میگذارد، به علاوه ایجاد تغییرات در راه حل‌های تست شده‌ی موجود ممکن است موجب خطاهای غیر منتظره ای شود. چگونه ممکن است که رفتار یک برنامه تغییر کند بدون اینکه کد آن ویرایش شود؟ چگونه می‌توانیم بدون تغییر یک موجودیت نرم افزاری کارکرد آن را تغییر دهیم؟

اصل **OCP** میگوید : **"ماژول‌های نرم افزار باید برای تغییرات بسته و برای توسعه باز باشند."**

راه حل هایی که OCP را نقض نمیکنند : **1- استفاده از وراثت (inheritance):**

ایجاد یک کلاس جدید به نام EmployeeManagerDB که از کلاس EmployeeDB ارث بری کند و متد Select آن را جهت نیاز خود بازنویسی کند.

```
public class EmployeeDB
{
    public virtual Employee Select()
    {
        //Old Select Method
    }
}
public class EmployeeManagerDB : EmployeeDB
{
    public override Employee Select()
    {
        //Select method as per Manager
        //UI requirement
    }
}
```

این انتخاب خیلی خوبی است در صورتی که این تغییرات در زمان طراحی اولیه پیش بینی شده باشد و همکنون قابل استفاده باشند.

کد UI هم به شکل زیر خواهد بود :

```
//Normal Screen
EmployeeDB objEmpDb = new EmployeeDB();
Employee objEmp = objEmpDb.Select();

//Manager Screen
EmployeeDB objEmpDb = new EmployeeManagerDB();
Employee objEmp = objEmpDb.Select();
```

2- متدهای الحاقی (Extension Method):

اگر شما از NET 3.5. یا بالاتر از آن استفاده میکنید، دومین راه استفاده از متدهای الحاقی است که به شما اجازه میدهد بدون هیچ دست زدنی به نوع‌های موجود، متدهای جدیدی را به آنها اضافه کنید.

```
Public static class MyExtensionMethod{
    public static Employee managerSelect(this EmployeeDB employeeDB) {
//Select method as per Manager
    }
}

//Manager Screen
Employee objEmp = EmployeeDB.managerSelect();
```

البته ممکن است راه‌های دیگری هم برای رسیدن به این منظور وجود داشته باشد.

درقسمت‌های بعدی قانون‌های دیگر را بررسی خواهیم کرد.