عنوان: **اصول طراحی شی گرا SOLID - #بخش پنجم اصل DIP**
نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۷/۰۶ ۲۰:۵۰
آدرس: www.dotnettips.info
گروه‌ها: OOP, SOLID Principals, Dependency Inversion, Object Oriented Design

بخش‌های پیشین : اصول طراحی شی گرا SOLID - #بخش اول اصل SRP اصول طراحی شی گرا SOLID - #بخش دوم اصل OCP
اصول طراحی شی گرا SOLID - #بخش سوم اصل LSP
اصول طراحی شی گرا SOLID - #بخش چهارم اصل ISP

اصل 5) D – DIP– Dependency Inversion principle

مقایسه با دنیای واقعی:

همان مثال کامپیوتر را دوباره در نظر بگیرید.این کامپیوتر دارای قطعات مختلفی مانند RAM ، هارد دیسک، CD ROM و ... است که هر کدام به صورت مستقل به مادربرد متصل شده اند. این به این معنی است که اگر قسمتی از کار بیفتد میتوان آن را با یک قطعه‌ی جدید به آسانی تعویض کرد . حالا فقط تصور کنید که تمامی قطعات شدیداً به یکدیگر متصل شده اند آنوقت دیگر نمیتوانستیم قطعه ای را از مادربرد برداریم و به همین خاطر اگر مثلا RAM از کار بیفتد ، باید یک مادربرد جدید خریداری کنید که برای شما گران تمام می‌شود.

به مثال زیر توجه کنید :

```
public class CustomerBAL
{
    public void Insert(Customer c)
    {
        try
        {
            //Insert logic
        }
        catch (Exception e)
        {
            FileLogger f = new FileLogger();
            f.LogError(e);
        }
    }
}

public class FileLogger
{
    public void LogError(Exception e)
    {
        //Log Error in a physical file
    }
}
```

در کد بالا کلاسCustomerBAL مستقیما به کلاس FileLogger وابسته است که استثناءهای رخ داده را بر روی یک فایل فیزیکی لاگ میکند. حالا فرض کنید که چند روز بعد مدیریت تصمیم میگیرد که از این به بعد استثناءها بر روی یک Event Viewer لاگ شوند.
اکنون چه میکنید؟ با تغییر کدها ممکن است با خطاهای زیادی روبرو شوید(درصورت تعداد بالای کلاسهایی که از کلاس FileLogger استفاده میکنند و فقط تعداد محدودی از آنها نیاز دارند که بر روی Event Viewer لاگ کنند.) **DIP** به ما میگوید : **"ماژول‌های سطح بالا نباید به ماژولهای سطح پایین وابسته باشند، هر دو باید به انتزاعات وابسته باشند. انتزاعات نباید وابسته به جزئیات باشند، بلکه جزئیات باید وابسته به انتزاعات باشند. ".**

در طراحی ساخت یافته، ماژولهای سطح بالا به ماژولهای سطح پایین وابسته بودند. این مسئله دو مشکل ایجاد می‌کرد:
1- ماژول‌های سطح بالا (سیاست گذار) به ماژول‌های سطح پایین (مجری) وابسته هستند. در نتیجه هر تغییری در ماژول‌های سطح پایین ممکن است باعث اشکال در ماژول‌های سطح بالا گردد.
2- استفاده مجدد از ماژول‌های سطح بالا در جاهای دیگر مشکل است، زیرا وابستگی مستقیم به ماژول‌های سطح پایین دارند. **راه حل با توجه به اصل DIP :**

```
public interface ILogger
{
    void LogError(Exception e);
}

public class FileLogger:ILogger
```

```
{
    public void LogError(Exception e)
    {
        //Log Error in a physical file
    }
}
public class EventViewerLogger : ILogger
{
    public void LogError(Exception e)
    {
        //Log Error in a Event Viewer
    }
}
public class CustomerBAL
{
    private ILogger _objLogger;
    public CustomerBAL(ILogger objLogger)
    {
        _objLogger = objLogger;
    }

    public void Insert(Customer c)
    {
        try
        {
            //Insert logic
        }
        catch (Exception e)
        {
            _objLogger.LogError(e);
        }
    }
}
```

در اینجا وابستگی‌های کلاس CustomerBAL از طریق سازنده آن در اختیارش قرار گرفته است. یک اینترفیس ILogger تعریف شده‌است به همراه دو پیاده سازی مختلف از آن مانند FileLogger و EventViewerLogger.
یکی از انواع فراخوانی آن نیز می‌تواند به شکل زیر باشد:

```
var customerBAL = new CustomerBAL (new EventViewerLogger());
customerBAL.LogError();
```

اطلاعات بیشتر در دوره آموزشی " بررسی مفاهیم معکوس سازی وابستگی‌ها و ابزارهای مرتبط با آن ".

نظرات خوانندگان

نویسنده: سعید سلیمانی فر
تاریخ: ۱۳۹۲/۰۷/۰۹ ۹:۳۱

خیلی مطلب خوبی بود! لذت بردیم متشکرم :)

نویسنده: بهزاد علی محمدزاده
تاریخ: ۱۳۹۲/۰۷/۱۹ ۱۶:۲۲

اقای طاهری با تشکر . امکان داره منبع رو معرفی کنید . به دنبال یه کتاب یا منبع آموزشی خوب در این زمینه هستم که البته نمونه‌ها رو با C# انجام داده باشه .

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۷/۱۹ ۱۷:۴۱

چند مقاله ای که من اونها رو مطالعه کردم : اصول طراحی SOLID SOLIDify your software design concepts through SOLID Articles by Christian Vos Object Oriented Design Principles SOLID by example SOLID Agile Development Articles Of SOLID SOLID Principles in C# - An Overview