عنوان: **آشنایی با الگوی طراحی Template Method**
نویسنده: وحید فرهمندیان
تاریخ: ۱۳۹۳/۱۲/۰۹ ۱۸:۲۵
آدرس: www.dotnettips.info
گروه‌ها: C#, OOP, DesignPatterns

سناریویی وجود دارد که در آن شما می‌خواهید تنها یک کار را انجام دهید، ولی برای انجام آن n روش وجود دارد. برای مثال قصد مرتب سازی دارید و برای اینکار روش‌های مختلفی وجود دارند. برای حل این مساله پیشتر از الگوی طراحی استراتژی استفاده نمودیم.(مطالعه بیشتر در مورد الگوی طراحی استراتژی)
حال به سناریویی برخورد کردیم که بصورت زیر است:
می‌خواهیم یک کار را انجام دهیم ولی برای انجام این کار تنها برخی بخش‌های کار با هم متفاوت هستند. برای مثال قصد تولید گزارش و چاپ آن را داریم. در این سناریو خواندن اطلاعات و پردازش آن‌ها رخدادهایی ثابت هستند. ولی اگر بخواهیم گزارش را چاپ کنیم به مشکل می‌خوریم؛ چرا که چاپ گزارش به فرمت اکسل، فرمت و روش خود را دارد و چاپ به فرمت PDF شرایط خود را دارد.
در این سناریو دیگر الگوی طراحی استراتژی جواب نخواهد داد و نیاز داریم با یک الگوی طراحی جدید آشنا بشویم. این الگوی طراحی Template Method نام دارد.
در این الگو یک کلاس انتزاعی داریم به صورت زیر:

```
public abstract class DataExporter
    {
        public void ReadData()
        {
            Console.WriteLine("Data is reading from SQL Server Database");
        }

        public void ProcessData()
        {
            Console.WriteLine("Data is processing...!");
        }

        public abstract void PrintData();

        public void GetReport()
        {
            ReadData();
            ProcessData();
            PrintData();
        }
    }
```

این کلاس abstract، یک متد بنام GetReport دارد که نحوه‌ی انجام کار را مشخص می‌کند. متدهای ReadData و ProcessData نشان می‌دهند که انجام این دو عمل همیشه ثابت هستند (منظور در این سناریو همیشه ثابت هستند). متد PrintData همانطور که مشاهده می‌شود بصورت انتزاعی تعریف شده است، چرا که چاپ عملی است که در هر فرمت دارای خروجی متفاوتی می‌باشد.
لذا در ادامه داریم:

```
public class ExcelExporter : DataExporter
    {
        public override void PrintData()
        {
            Console.WriteLine("Data exported to Microsoft Excel!");
        }
    }

    public class PDFExporter : DataExporter
    {
        public override void PrintData()
        {
            Console.WriteLine("Data exported to PDF!");
        }
    }
```

کلاس ExcelExporter برای چاپ به فرمت اکسل می‌باشد. همانطور که مشاهده می‌شود این کلاس از کلاس انتزاعی DataExporter ارث بری کرده است. این بدین معنا است که کلاس ExcelExporter کارهای ReadData و ProcessData را از کلاس

پدر خود می‌گیرد و در ادامه نحوه‌ی چاپ مختص به خود را پیاده می‌کند. همین توضیحات در مورد PDFExporter نیز صادق است.
حال برای استفاده‌ی از این کدها داریم:

```
DataExporter dataExporter = new ExcelExporter();
            dataExporter.GetReport();
            Console.WriteLine("**************************");
            dataExporter = new PDFExporter();
            dataExporter.GetReport();
```

شما شاید بخواهید متدهای ReadData و ExportData و ProcessData را با سطح دسترسی متفاوتی از public تعریف نمایید که در این مقاله به این دلیل که خارج از بحث بود به آنها اشاره نشد و بصورت پیش فرض public در نظر گرفته شد.