عنوان: **اصول طراحی شی گرا SOLID - #بخش چهارم اصل ISP**
نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۷/۰۶ ۲۰:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: OOP, SOLID Principals, Object Oriented Design, Interface Segregation

**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

**اصل 4) I – ISP- Interface Segregation principle**

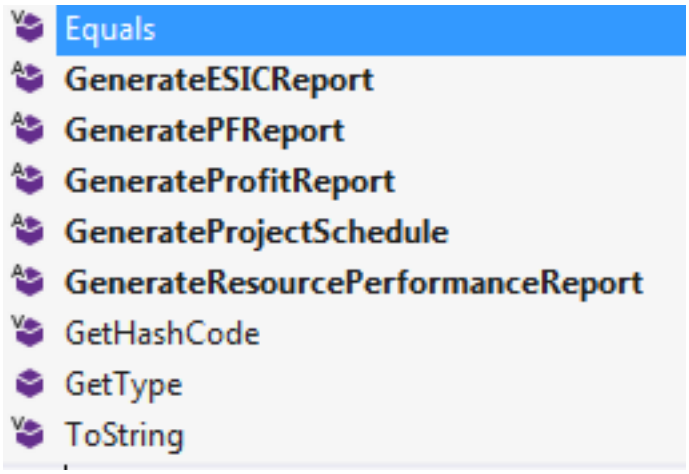
مقایسه با دنیای واقعی:

بیایید فکر کنیم شما یک کامپیوتر دسکتاپ جدید خریداری کرده اید.شما یک زوج پورت USP، چند پورت سریال، یک پورت VGA و ... را پیدا میکنید. اگر شما بدنه‌ی کیس خود را باز کنید، تعدادی اسلات بر روی مادربرد خود که برای اتصال قطعات مختلف با یکدیگر هستند را مشاهده خواهید کرد که عمدتا مهندسان سخت افزار در زمان منتاژ از آنها استفاده میکنند.
این اسلات‌ها تا زمانی که بدنه کیس را باز نکنید قابل مشاهده نخواهند بود. به طور خلاصه تنها رابطه‌های مورد نیازی که برای شما ساخته شده اند، قابل مشاهده خواهند بود.
و فرض کنید شما به یک توپ فوتبال نیاز دارید. به یک فروشگاه وسایل ورزشی میروید و فروشنده شروع به نشان دادن انواع توپ ها، کفش ها،لباس و گرم کن‌های ورزشی،لباس شنا و کاراته ،زانوبند و ... کرده است. در نهایت ممکن است شما چیزی را خریداری کنید که اصلا مورد نیازتان نبوده است یا حتی ممکن است فراموش کنیم که ما چرا اینجا هستیم.! **ارائه مشکل:**
ما میخواهیم یک سیستم مدیریت گزارشات را توسعه بدهیم. اولین کاری که انجام می‌دهیم ایجاد یک لایه منطقی که توسط سه UI مختلف دیگر مورد استفاده قرار میگیرد.
1- EmployeeUI : نمایش گزارش‌های مربوط با کارمند وارد شده‌ی جاری در سایت.
2- ManagerUI : نمایش گزارش‌های مربوط به خود و تیمی که به او تعلق دارد.
3- AdminUI : نمایش گزارش‌های مربوط به کارمندان، مربوط به تیم و مربوط به شرکت مانند گزارش سود و زیان و ...

```
public interface IReportBAL
{
    void GeneratePFReport();
    void GenerateESICReport();

    void GenerateResourcePerformanceReport();
    void GenerateProjectSchedule();

    void GenerateProfitReport();
}
```

```
public class ReportBAL : IReportBAL
{
    public void GeneratePFReport()
    {/*..............*/}

    public void GenerateESICReport()
    {/*..............*/}

    public void GenerateResourcePerformanceReport()
    {/*..............*/}

    public void GenerateProjectSchedule()
    {/*..............*/}

    public void GenerateProfitReport()
    {/*..............*/}
}
public class EmployeeUI
{
    public void DisplayUI()
    {
        IReportBAL objBal = new ReportBAL();
        objBal.GenerateESICReport();
        objBal.GeneratePFReport();
    }
}
public class ManagerUI
{
    public void DisplayUI()
    {
        IReportBAL objBal = new ReportBAL();
        objBal.GenerateESICReport();
        objBal.GeneratePFReport();
        objBal.GenerateResourcePerformanceReport ();
        objBal.GenerateProjectSchedule ();
    }
}
public class AdminUI
{
    public void DisplayUI()
    {
        IReportBAL objBal = new ReportBAL();
        objBal.GenerateESICReport();
        objBal.GeneratePFReport();
        objBal.GenerateResourcePerformanceReport();
        objBal.GenerateProjectSchedule();
        objBal.GenerateProfitReport();
    }
}
```

حال زمانی که توسعه دهنده در هر UI عبارت objBal را تایپ کند، در لیست بازشوی آن (intellisense ) به صورت زیر نمایش داده خواهد شد :

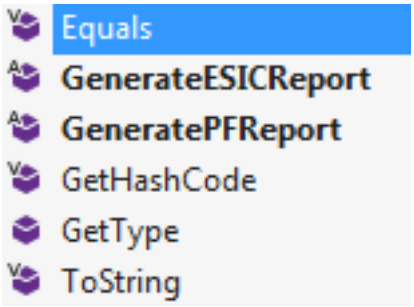


اما مشکل چیست؟ مشکل این است شخصی که روی لایه‌ی EmployeeUI کار میکند به تمام متدهاها به خوبی دسترسی دارد و این متدهای غیرضروری ممکن است باعث سردرگمی او شود.
این مشکل به این دلیل اتفاق می‌افتد که وقتی کلاسی یک واسط را پیاده سازی می‌کند، باید همه متدهای آن را نیز پیاده سازی کند. حالا اگر خیلی از این متدها توسط این کلاس استفاده نشود، می‌گوییم که این کلاس از مشکل واسط چاق رنج می‌برد.
اصل **ISP** میگوید : **" کلاینت‌ها نباید وابسته به متدهایی باشند که آنها را پیاده سازی نمی‌کنند. "**
برای رسیدن به این امر در مثال بالا باید آن واسط را به واسطهای کوچکتر تقسیم کرد. این تقسیم بندی باید بر اساس استفاده کنندگان از واسطها صورت گیرد.
ویرایش مثال بالا با در نظر گرفتن اصل ISP :

```
public interface IEmployeeReportBAL
{
    void GeneratePFReport();
    void GenerateESICReport();
}
public interface IManagerReportBAL : IEmployeeReportBAL
{
    void GenerateResourcePerformanceReport();
    void GenerateProjectSchedule();
}
public interface IAdminReportBAL : IManagerReportBAL
{
    void GenerateProfitReport();
}
public class ReportBAL : IAdminReportBAL
{
    public void GeneratePFReport()
    {/*..............*/}

    public void GenerateESICReport()
    {/*..............*/}

    public void GenerateResourcePerformanceReport()
    {/*..............*/}

    public void GenerateProjectSchedule()
    {/*..............*/}

    public void GenerateProfitReport()
    {/*..............*/}
}
```

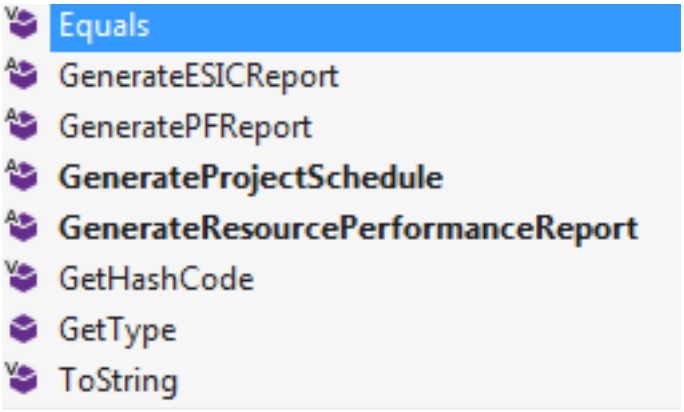
وضعیت نمایش :

```
public class EmployeeUI
{
    public void DisplayUI()
    {
        IEmployeeReportBAL objBal = new ReportBAL();
        objBal.GenerateESICReport();
        objBal.GeneratePFReport();
    }
}
```

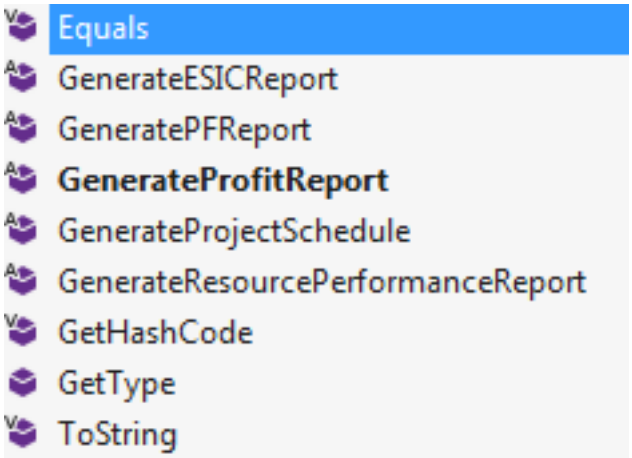



```
public class ManagerUI
{
    public void DisplayUI()
```

```
    {
        IManagerReportBAL  objBal = new ReportBAL();
        objBal.GenerateESICReport();
        objBal.GeneratePFReport();
        objBal.GenerateResourcePerformanceReport ();
        objBal.GenerateProjectSchedule ();
    }
}
```




```
public class AdminUI
{
    public void DisplayUI()
    {
        IAdminReportBAL  objBal = new ReportBAL();
        objBal.GenerateESICReport();
        objBal.GeneratePFReport();
        objBal.GenerateResourcePerformanceReport();
        objBal.GenerateProjectSchedule();
        objBal.GenerateProfitReport();
    }
}
```




```
DIP   :
```

```
در قسمت بعدی آخرین اصل
```

را بررسی خواهیم کرد