设计模式之六大原则

前言：

为了使不同的开发人员都可以写出规范简洁、可读性高、耦合度低、逻辑清晰的代码，技术人员总结出了以下六大设计原则。

一、里氏替换原则LSP

定义：任何使用基类（父类）的地方都可以安全的使用继承它的子类代替；

安全：不会出现行为不一致的情况；

继承：子类拥有父类的一切属性和行为。

需要注意的是：

1、对于继承来说，如果父类出现了子类不需要的成员，那么就应该断掉这种强类间关系，不要强行继承。

2、子类可以拥有属于自己的属性和行为，所以子类出现的地方，父类不一定可以替代。

3、父类不强制子类必须遵从自己所有的行为，比如子类可以重写父类的抽象方法、虚方法等。

4、如果子类非要重写父类的普通方法，那么比较通用的方式是：使原来的父类和子类都继承一个更加通俗的基类，把原有的继承关系去掉，而采用依赖、聚合、组合等关系代替；

二、单一职责原则SRP

定义：一个类/接口/方法只负责一件事。

说明：例如类T负责两个不同的职责P1、P2,当由于职责P1需求发生改变而需要修改类T时，就有可能导致原本运行正常的P2功能发生故障。

解决：通过遵循单一职责原则，分别建立新的类来对应相应的职责；这样就能避免修改类时影响到其他的职责；

优点：简单意味着稳定，并且类的复杂性将会降低，可读性提高，维护性也会提高。

缺点：拆分导致代码量增加，类多了导致管理成本提高。

例外：在遇到类中职责扩散、逻辑足够简单、方法数量足够少的情况下，可以考虑在代码级别上面违反单一职责原则。

三、依赖倒置原则DIP

定义：高层模块不应该依赖低层模块，二者都应该通过抽象实现依赖，而不应该是高层依赖底层的细节；

说明：比如类A直接依赖类B，现在要将类A改为依赖类C，则必须通过修改类A的代码来完成；这种场景下，类A一般是高层模块，负责复杂的业务逻辑；类B和类C是低层模块，负责基本的原则操作；假如修改类A，会给程序带来不必要的风险。

实现：将类A修改为依赖接口I，类B和类C各自实现接口I，类A通过接口I来间接与类B和类C发生联系，则会降低修改类A的几率；

在实际中，我们一般需要做到以下三点：

1.低层模块尽量都要有抽象类或者接口，或者两者都有；

2.变量的声明类型尽量是抽象类或者接口；

3.使用继承时遵循里氏替换原则；

缺点：面向抽象后，就不能使用子类的特殊内容，这种情况就不应该使用面向抽象；

四、接口隔离原则ISP

定义：一个类对另一个类的依赖应该建立在最小的接口上，否则将会造成接口污染；如类A通过接口I依赖类B，如果接口I对于类A来说不是最小接口，则类B必须去实现它们不需要的方法；

说明：建立功能单一的接口，不要建立庞大臃肿的接口，尽量细化接口，接口中的方法尽量少；就是说，不要试图去建立一个庞大的接口供所有依赖它的类去调用，这样会导致强迫它的实现类实现多余的行为；

注意：接口尽量小，但是要有限度，对接口进行细化拆分可以提高程序设计灵活性，但是如果过小，则会导致接口数量过多，使设计复杂化，失去面向对象的意义。所以一定要适度，为依赖接口的类定制服务，只定义给调用的类需要的方法；

已经被污染了的接口，尽量修改，如果变更风险太大，则用适配器模式进行转化；

五、开闭原则OCP

定义：一个软件实体如类、模版和函数应该对扩展开放，对修改关闭；

实现：当软件需要变化时，尽量通过扩展软件实体的行为来实现变化，而不是修改已有的代码来实现变化；

核心思想就是用扩展替代修改，降低修改带来的风险，提高代码的稳定性；

六、迪米特原则LOD

定义：一个对象应该对其他对象保持最少的了解，简单来说就是一个对象应该对自己需要关联调用的类知道的少；这会降低类与类之间的耦合度，每个类都应尽量减少对其他类细节的依赖；

实现：

1.减少内部依赖；

2.降低访问修饰符权限；

3.逻辑尽量不要集中，要拆分开；

举例：

这个例子描述的是一个人用咖啡机煮咖啡的过程。例子中只有两个类，一个类是人 -- Class Person，一个类是咖啡机 -- Class CoffeeMachine，类中包含三个方法AddCoffeeBean()、AddWater()、MakeCoffee()。

当CoffeeMachine将三个方法全都暴露出来(访问修饰符设置为public)，那么Person就先要获得一个CoffeeMachine对象，然后按顺序依次调用以上三个方法才能得到咖啡；

但是如果由于业务需要，现在要求调换加水和加咖啡豆的顺序，就要修改Person的方法内部，就有可能影响到这个方法整体的功能；

而且站在Person的角度看，它根本就不需要关心咖啡的制作流程和具体细节，它只是想得到一杯咖啡。所以合理的做法是，只需CoffeeMachie暴露一个Work()的方法，将其他方法都应对外隐藏，具体实现由其内部决定，上层只调用这一个方法去获得想要的结果；当业务发生改变也只需要修改这一个类，调用它的类则不用做出调整，从而减少不必要的错误，提高系统的稳定性；

七、项目架构优化

```
// PUT: api/Purse/close/5
[HttpPut("admin/close/{gid}")]
[ApiFilter(ApiType = ApiType.Admin, ApiPath = "/api/Purse/admin/close/{gid}")]
0 个引用 | cary.hu，290 天前 | 2 名作者，6 项更改
public void Close([FromRoute] Guid gid)
{
    var purseBo = _boProvider.GetPurseBo(gid);
    purseBo.Close();
}
```

```
#region PurseBo

3 个引用 | Theodore，68 天前 | 1 名作者，1 项更改
internal PurseBo GetPurseBo(Guid guid)
{
    Purse purse = _purseRepo.GetPurseByGid(guid);
    return new PurseBo(purse)
    {
        _boProvider = this,
    };
}

4 个引用 | Theodore，69 天前 | 1 名作者，1 项更改
internal PurseBo GetPurseBoOrNull(PurseTypeEnum typeEnum, string userId)
{
    Purse purse = _purseRepo.GetPurseOrNull(typeEnum, userId);
    if (purse.IsNull())
        return null;

    return new PurseBo(purse)
    {
        _boProvider = this,
    };
}


2 个引用 | Theodore，69 天前 | 1 名作者，1 项更改
internal PurseBo GetPurseHrBoOrNull(string customerId)
{
    Purse purse = _purseRepo.GetPurseOrNullByCustomerId(customerId);
    if (purse.IsNull())
        return null;
    return new PurseBo(purse)
    {
        _boProvider = this,
    };
}

8 个引用 | Theodore，69 天前 | 1 名作者，1 项更改
internal PurseBo GetPurseHrBo(string customerId)
{
    var bo = GetPurseHrBoOrNull(customerId);
    if (bo.IsNull())
        throw ExceptionHelper.DataNotFoundException($"企业账户找不到{customerId}");
    return bo;
}
```

```
4 个引用 | Theodore，69 天前 | 1 名作者，1 项更改
internal PurseBo GetPurseBoOrNull(PurseTypeEnum typeEnum, string userId)
{
    Purse purse = _purseRepo.GetPurseOrNull(typeEnum, userId);
    if (purse.IsNull())
        return null;

    return GetPurseBoFactory(purse);
}


1 个引用 | 0 项更改 | 0 名作者，0 项更改
internal PurseBo GetPurseBoFactory(Purse purse)
{
    return new PurseBo(purse)
    {
        _boProvider = this,
    };
}
```

```
21 个引用 | cary.hu，36 天前 | 4 名作者，14 项更改
public class BoBase
{
    99+ 个引用 | Wellis Sirh，322 天前 | 1 名作者，1 项更改
    public BoProvider _boProvider { get; set; }

    1 个引用 | 0 项更改 | 0 名作者，0 项更改
    public BoBase(BoProvider boProvider)
    {
        _boProvider = boProvider;
    }

    //public Tmodel Model { get; internal set; }

    36 个引用 | Wellis Sirh，301 天前 | 1 名作者，1 项更改
    public void SaveChanges()
    {
        _boProvider._context.SaveChanges();
    }

    6 个引用 | Wellis Sirh，195 天前 | 1 名作者，2 项更改
    public IDbContextTransaction BeginTransaction()
    {
        return _boProvider.BeginTransaction();
    }

    3 个引用 | Wellis Sirh，195 天前 | 1 名作者，1 项更改
    public void Commit()
    {
        _boProvider.Commit();
    }

    1 个引用 | Wellis Sirh，195 天前 | 1 名作者，1 项更改
    public void Rollback()
    {
        _boProvider.Rollback();
    }
```

```
6 个引用 | 0 项更改 | 0 名作者，0 项更改
public PurseBo(Purse purse, BoProvider boProvider) :base(boProvider)
{
    Purse = purse;
}
```

```
1 个引用 | 0 项更改 | 0 名作者，0 项更改
internal PurseBo GetPurseBoFactory(Purse purse)
{
    return new PurseBo(purse, this);
}
```

```
4 个引用 | 0 项更改 | 0 名作者，0 项更改
internal R PublicBoFactory<R, T>(T input) where R : BoBase where T : BaseModel
{
    Type t = typeof(R);
    var ci = t.GetConstructors();
    foreach (ConstructorInfo c in ci)
    {
        ParameterInfo[] ps = c.GetParameters();
        foreach (ParameterInfo pi in ps)
        {
            if (pi.ParameterType == typeof(T))
            {
                R result = Activator.CreateInstance(t, new object[] { input,this }) as R;
                result._boProvider = this;
                return result;
            }
        }
    }
    return null;
}
```

```
4 个引用 | Theodore，69 天前 | 1 名作者，1 项更改
internal PurseBo GetPurseBoOrNull(PurseTypeEnum typeEnum, string userId)
{
    Purse purse = _purseRepo.GetPurseOrNull(typeEnum, userId);
    if (purse.IsNull())
        return null;

    return PublicBoFactory<PurseBo, Purse>(purse);
}
```

```
#region chargeBo
1 个引用 | Theodore，69 天前 | 1 名作者，1 项更改
public ChargeLogHrBo GetChargeHrLogBo(Guid hrChargeLogGuid)
{
    ChargeLogHr chargeLogHr = _categoryRepo.GetChargeHrLog(hrChargeLogGuid);
    return PublicBoFactory<ChargeLogHrBo, ChargeLogHr>(chargeLogHr);
}
```