

虚方法的使用

张逸

www.agiledon.com

C#的语法脱胎于 C++，因而保留了 `virtual` 关键字，可以定义一个虚方法（或虚属性）。一个类的成员被定义为 `virtual`，就意味着它在告诉自己的子类：我准备了一笔遗产，你可以全盘接受，也可以完全拒绝或者修改我的遗嘱。显然，虚方法授予子类的权利甚至大于抽象方法。子类面对抽象方法只有重写（`override`）的权利，而对于虚方法，它还可以选择完全继承。

毫无疑问，虚方法破坏了对对象的封装性。如果不加约束的使用，会对调用方造成破坏，至少它有可能破坏子类与父类之间在外在行为上的一致性。因此，当我们在重写虚方法时，务必遵循 Liskov 替换原则。我们要保证对于调用方而言，子类对于父类是完全可以替换的。这里所谓的“替换”，是指子类不能破坏调用方对父类行为的期待。准确地说，子类在重写父类的虚方法时，必须遵循调用该方法的前置条件与后置条件。这也是“契约式设计”的思想。最理想的状态是让使用对象甚至无法知道是否存在派生类^[1]。即类的继承体系对于调用者而言，必须体现外部接口的一致性，这样才能做到调用者对派生类无知。

如果确实需要重写父类的方法，最好的方式是扩展而不是修改。这实际上也是开放-封闭原则的体现。例如在 Decorator 模式中，我们重写父类方法的目的，是为了实现对该方法的装饰。Proxy 模式的实现同样如此。Michael C. Feathers 对此给出的忠告是^[2]：

- 1) 尽可能避免重写具体方法。
- 2) 倘若真的重写了某个具体方法，那么看看能否在重写方法中调用被重写的那个方法。

Feathers 的忠告是针对 Java 语言，因为在 C#中我们无法重写具体方法，只能利用 `new` 关键字在子类中新建一个相同方法签名的具体方法，而这样的方法并不具备多态性。这里涉及到一个有趣的话题，是关于 Java 和 C#的比较。在 Java 语言中，如果没有添加任何关键字，则方法默认就是虚方法，任何子类都可以重写它。C#则相反，它对虚方法给予了显式的定义。Java 语言的缔造者显然是“性本善”论者，他认为所有子类的实现者均抱着善意的态度来对待父类的方法，因而他赋予了子类相当程度的自由，但却可能被别有用心者偷偷打开封装的后门。如果确有非常重要的隐私防止被篡改，则可以利用 `final` 关键字来强制保护。C#语言的发明者则持有“性本恶”的论调，他恶意地揣测子类总是会不怀好意，所以提供了一套默认的强权，来保护父类的隐私。如果需要子类开放，则明确地声明为 `virtual`，这就牢牢地把控制权攥紧在父类的手中。

C#保守的做法使得语言的特质更加安全(当然,Java会更加自由),我们可以使用virtual的自由性,搭配方法的访问限制,搭建一个安全合理的白盒框架。virtual关键字的含意本身就是面向子类的,所以,我们应该尽可能地将其放在protected方法中使用。如果该方法代表的行为确实需要公开给调用者,我们可以定义一个公开的具体方法,在其中调用一个受保护的虚方法。

在Template Method模式中,体现了C#这种划分具体方法和虚方法的好处。Template Method模式要求子类只能部分地替换父类的实现,整个骨架则必须保持固定不变。在父类中,我们将模板方法定义为具体方法,将基本方法定义为抽象方法。模板方法规定了基本方法的调用顺序,如果我们可以在子类中重写模板方法,就可能破坏基本方法的调用顺序,从而对整个策略造成影响。Strategy模式就不存在这个问题,因为它的策略是整体的。Template Method模式在模板方法中规定的骨架,实际上就是为调用者制订的前置条件和后置条件。

有一种说法是不要在虚方法中访问私有字段^[3]。这存在一定的合理性。因为一旦我们在父类的虚方法中访问了私有字段,那么在子类重写该虚方法时,由于无法获得父类的私有字段值,就可能会导致该字段值的缺失。但这种说法并不完全准确。一方面,我们认为Liskov替换原则主要是为了约束Is-A关系在行为上的一致性^[4],如果该字段对行为不会造成影响,则无大碍。另一方面,这也说明我们在重写虚方法时,最佳实践还是需要同时在重写的同时,调用父类的虚方法,如Decorator模式的实现方式。

[1] Alan Shalloway, James R. Trott Design Patterns Explained

[2] Michael C. Feathers Working Effectively with Legacy Code

[3] Dino Esposito, Andrea Saltarello Microsoft.NET Architecting Applications for the Enterprise

[4] Robert C. Martin Agile Software Development:Principles,Patterns and Practices