[**C#设计模式(14)——模板方法模式（Template Method）**](https://www.cnblogs.com/zhili/p/TemplateMethodPattern.html)

**一、引言**

提到模板，大家肯定不免想到生活中的“简历模板”、“论文模板”、“Word中模版文件”等，在现实生活中，模板的概念就是——有一个规定的格式，然后每个人都可以根据自己的需求或情况去更新它，例如简历模板，下载下来的简历模板的格式都是相同的，然而我们下载下来简历模板之后我们可以根据自己的情况填充不同的内容要完成属于自己的简历。在设计模式中，模板方法模式中模板和生活中模板概念非常类似，下面让我们就详细介绍模板方法的定义，大家可以根据生活中模板的概念来理解模板方法的定义。

**二、模板方法模式详细介绍**

**2.1 模板方法模式的定义**

模板方法模式——在一个抽象类中定义一个操作中的算法骨架（对应于生活中的大家下载的模板），而将一些步骤延迟到子类中去实现（对应于我们根据自己的情况向模板填充内容）。模板方法使得子类可以不改变一个算法的结构前提下，重新定义算法的某些特定步骤，模板方法模式把不变行为搬到超类中，从而去除了子类中的重复代码。

**2.2 模板方法模式的实现**

理解了模板方法的定义之后，自然实现模板方法也不是什么难事了，下面以生活中炒蔬菜为例来实现下模板方法模式。在现实生活中，做蔬菜的步骤都大致相同，如果我们针对每种蔬菜类定义一个烧的方法，这样在每个类中都有很多相同的代码，为了解决这个问题，我们一般的思路肯定是把相同的部分抽象出来到抽象类中去定义，具体子类来实现具体的不同部分，这个思路也正式模板方法的实现精髓所在，具体实现代码如下：

[复制代码](javascript:void(0);)

// 客户端调用

class Client

{

static void Main(string[] args)

{

// 创建一个菠菜实例并调用模板方法

Spinach spinach = new Spinach();

spinach.CookVegetabel();

Console.Read();

}

}

public abstract class Vegetabel

{

// 模板方法，不要把模版方法定义为Virtual或abstract方法，避免被子类重写，防止更改流程的执行顺序

public void CookVegetabel()

{

Console.WriteLine("抄蔬菜的一般做法");

this.pourOil();

this.HeatOil();

this.pourVegetable();

this.stir\_fry();

}

// 第一步倒油

public void pourOil()

{

Console.WriteLine("倒油");

}

// 把油烧热

public void HeatOil()

{

Console.WriteLine("把油烧热");

}

// 油热了之后倒蔬菜下去，具体哪种蔬菜由子类决定

public abstract void pourVegetable();

// 开发翻炒蔬菜

public void stir\_fry()

{

Console.WriteLine("翻炒");

}

}

// 菠菜

public class Spinach : Vegetabel

{

public override void pourVegetable()

{

Console.WriteLine("倒菠菜进锅中");

}

}

// 大白菜

public class ChineseCabbage : Vegetabel

{

public override void pourVegetable()

{

Console.WriteLine("倒大白菜进锅中");

}

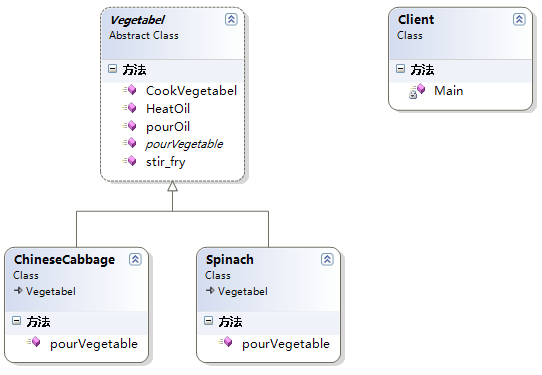
}

[复制代码](javascript:void(0);)

在上面的实现中，具体子类中重写了导入蔬菜种类的方法，因为这个真是烧菜方法中不同的地方，所以由具体子类去实现它。

**2.3 模板方法模式的类图**

实现完模板方法模式之后，让我们看看模板方法的类图结构，以理清该模式中类之间的关系，具体类图如下：



模板方法模式中涉及了两个角色：

* **抽象模板角色（Vegetable扮演这个角色）**：定义了一个或多个抽象操作，以便让子类实现，这些抽象操作称为基本操作。
* **具体模板角色（ChineseCabbage和Spinach扮演这个角色）**：实现父类所定义的一个或多个抽象方法。

**三、模板方法模式的优缺点**

下面让我们继续分析下模板方法的优缺点。

**优点：**

1. 实现了代码复用
2. 能够灵活应对子步骤的变化，符合开放-封闭原则

**缺点**：因为引入了一个抽象类，如果具体实现过多的话，需要用户或开发人员需要花更多的时间去理清类之间的关系。

 附：在.NET中模板方法的应用也很多，例如我们在开发自定义的Web控件或WinForm控件时，我们只需要重写某个控件的部分方法。

**四、总结**

到这里，模板方法的介绍就结束了，模板方法模式在抽象类中定义了算法的实现步骤，将这些步骤的实现延迟到具体子类中去实现，从而使所有子类复用了父类的代码，所以模板方法模式是基于继承的一种实现代码复用的技术。