简单工厂模式和策略模式的区别

设计模式

（10354）

最近一直在抽时间研究设计模式，之前对设计模式也有一定的了解，但是都没有平心静气的去研究过，只是了解了一些皮毛，最近打算再深入研究一下，重新打开了设计模式的数据，对之前的疑问一个个的刨根问底，今天看了简单工厂模式和策略模式看的人有点小晕，两个的大概思路没有怎么变，都是通过多态去减少代码的耦合度，怎么看两个都是如出一辙，最后终于找到了两个的本质区别，在此和大家分享下：

先上代码：

简单工厂模式：

```
//抽象类
abstract class AbsClass
{
  //抽象方法：提供一些列的算法操作
  public abstract void acceptCash(string org);
}

//继承自抽象类
class A:AbsClass
{
  //具体方法：提供一些列的算法操作
  public override double acceptCash(string org)

      Console.WriterLine("A类方法");

//继承自抽象类
class B:AbsClass
{
  //具体方法：提供一些列的算法操作
  public override double acceptCash(string org)

      Console.WriterLine("B类方法");

... .... ....
```

简单工厂类

```
//现金收取工厂
class CashFactory
{
  //根据条件返回相应的对象
  public static AbsClass createCashAccept(string type)
  {
    AbsClass cs = null;
```

```
        switch (type)
        {
            case "A":
                cs = new Al();
                break;
            case "B":
                cs = new B();
                break;
            case "...":
                ..... ....
                break;
        }
        return cs;
    }
}
```

客户端调用:

```
/利用简单工厂模式根据下拉选择框，生成相应的对象
AbsClass csuper = CashFactory.createCashAccept("A");
```

策略模式:

前面的类没有任何变化，只是把Factory变成了CaseContext策略类

```
//策略Context
class CashContext
{
    //声明一个现金收费父类对象
    private AbsClass cs;

    //设置策略行为，参数为具体的现金收费子类（正常，打折或返利）
    public CashContext(AbsClass csuper)
    {
        this.cs = csuper;
    }

    //得到现金促销计算结果（利用了多态机制，不同的策略行为导致不同的结果）
    public double GetResult(double money)
    {
        return cs.acceptCash(money);
    }
}
```

客户端调用:

```
AbsClass cc = null;
    switch (cbxType.SelectedItem.ToString())
        {
            case "A":
                cc = new CashContext(new A());
                break;
            case "B":
                cc = new CashContext(new B());
```

```
            break;
        case "...":
            ... ....
            break;
    }
```

最后概括总结一下:

策略模式和简单工厂模式看起来非常相似，都是通过多态来实现不同子类的选取，这种思想应该是从程序的整体来看得出的。如果从使用这两种模式的角度来看的话，我们会发现在简单工厂模式[1]中我们只需要传递相应的条件就能得到想要的一个对象，然后通过这个对象实现算法的操作。而策略模式[2]，使用时必须首先创建一个想使用的类对象，然后将该对象最为参数传递进去，通过该对象调用不同的算法。在简单工厂模式中实现了通过条件选取一个类去实例化对象，策略模式则将选取相应对象的工作交给模式的使用者，它本身不去做选取工作。

结合上面的代码和下面的释义不难看出，其实两个的差别很微妙,Factory是直接创建具体的对象并用该对象去执行相应的动作，而Context将这个操作给了Context类，没有创建具体的对象，实现的代码的进一步封装，客户端代码并不需要知道具体的实现过程。

今天已经晚了就写到这，感谢你对Darren作品的支持。如果有什么不懂的可以提问。

Links

1. http://www.tianboo.net/
2. http://www.tianboo.net/