常见的六种设计模式以及应用场景

设计模式是对设计原则的具体化。用江湖话说就是武林秘籍，总结出来的一些固定套路，可以帮助有根基的程序员迅速打通任督二脉，从此做什么都特别快。常用的模式及其场景如下。

1) 单例模式。

单例模式是一种常用的软件设计模式。

在它的核心结构中只包含一个被称为单例类的特殊类。通过单例模式可以保证系统中一个类只有一个实例而且该实例易于外界访问，从而方便对实例个数的控制并节约系统资源。

应用场景：如果希望在系统中某个类的对象只能存在一个，单例模式是最好的解决方案。

2) 工厂模式。

工厂模式主要是为创建对象提供了接口。

应用场景如下：

a、 在编码时不能预见需要创建哪种类的实例。

b、 系统不应依赖于产品类实例如何被创建、组合和表达的细节。

3) 策略模式。

策略模式：定义了算法族，分别封装起来，让它们之间可以互相替换。此模式让算法的变化独立于使用算法的客户。

应用场景如下。

a、 一件事情，有很多方案可以实现。

b、我可以在任何时候，决定采用哪一种实现。

c.、未来可能增加更多的方案。

d、 策略模式让方案的变化不会影响到使用方案的客户。

举例业务场景如下。

系统的操作都要有日志记录，通常会把日志记录在数据库里面，方便后续的管理，但是在记录日志到数据库的时候，可能会发生错误，比如暂时连不上数据库了，那就先记录在文件里面。日志写到数据库与文件中是两种算法，但调用方不关心，只负责写就是。

4) 观察者模式。

观察者模式又被称作发布/订阅模式，定义了对象间一对多依赖，当一个对象改变状态时，它的所有依赖者都会收到通知并自动更新。

应用场景如下：

a、对一个对象状态的更新，需要其他对象同步更新，而且其他对象的数量动态可变。

b、对象仅需要将自己的更新通知给其他对象而不需要知道其他对象的细节。

5) 迭代器模式。

迭代器模式提供一种方法顺序访问一个聚合对象中各个元素，而又不暴露该对象的内部表示。

应用场景如下：

当你需要访问一个聚集对象，而且不管这些对象是什么都需要遍 历的时候，就应该考虑用迭代器模式。其实 stl 容器就是很好的迭代器模式的例子。

6) 模板方法模式。

模板方法模式定义一个操作中的算法的骨架，将一些步骤延迟到子类中，模板方法使得子类可以不改变一个算法的结构即可重定义该算法的某些步骤。

应用场景如下：

对于一些功能，在不同的对象身上展示不同的作用，但是功能的框架是一样的。