**生成器模式(建造者模式)**

建造者模式（Builder Pattern）使用多个简单的对象**一步一步**构建成一个复杂的对象。这种类型的设计模式属于创建型模式。

一个 Builder 类会一步一步构造最终的对象。该 Builder 类是独立于其他对象的。

**介绍**

**意图：** 将一个复杂的构建与其表示相分离，使得同样的构建过程可以创建不同的表示。

**主要解决：** 主要解决在软件系统中，有时候面临着**"一个复杂对象"的创建工作**，其通常由各个部分的子对象用一定的算法构成；由于需求的变化，**这个复杂对象的各个部分经常面临着剧烈的变化，但是将它们组合在一起的算法却相对稳定。**

**何时使用：** 1**.**一些基本部件不会变，而其组合经常变化的时候。2.需要一个有弹性的数据结构,需要遵照一系列的复杂顺序，创建这样的规划。

**如何解决：** 将变与不变分离开。将创建的过程封装到生成器(Builder)中。

**关键代码：** 建造者：创建和提供实例，导演：管理建造出来的实例的依赖关系。

**应用实例：**  1、去肯德基，汉堡、可乐、薯条、炸鸡翅等是不变的，而其组合是经常变化的，生成出所谓的"套餐"。 2、JAVA 中的 StringBuilder。

**优点：**

1、建造者独立，易扩展。

2、便于控制细节风险。

3.将一个复杂对象的创建过程封装起来。

4.允许对象通过多个步骤来创建，并且可以改变过程。

5.向客户隐藏产品的内部表现

6.产品的实现可以被替换，因为客户只看到一个抽象的接口。

**缺点：**  1、产品必须有共同点，范围有限制。 2、如内部变化复杂，会有很多的建造类。

**使用场景：**  **1、需要生成的对象具有复杂的内部结构。** **2、需要生成的对象内部属性本身相互依赖。(经常被用来创建组合结构)**

**注意事项：**

**与工厂模式的区别是：**

**1.建造者模式更加关注与零件装配的顺序。**

**2.采用生成器模式创建对象的客户，需要具备更多的领域知识。**