## 一句话概括：

**使用多个简单对象一步一步构建成复杂对象，将复杂对象的构建与表示相分离。**

## 补充介绍：

建造者模式（Builder Pattern）中有一个 Builder 类，这个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。

建造者模式的目的是**将复杂对象的构建与表示相分离**，使同样的构建过程可以构建不同的表示。将复杂对象的**变与不变相分开**。

建造高楼大厦时，需要打地基，搭建框架，浇灌混泥土，一层一层盖起来。建造高楼大厦就好比用Builder模式构建一个复杂对象。

肯德基的炸鸡汉堡薯条可乐，可以拼凑在一起成为一个套餐。

## 参与角色:

1）抽象建造类

2）建造类的实现类（多个不同实现类）

3）复杂对象类

## 优点：

1）建造者独立，易于扩展；

2）便于控制细节风险

## 缺点：

1）产品必须有共同点，范围有限制；

2）如果内部变化复杂，会有很多建造类

## 使用案例或场景：

使用场景：一个类的基本部件不会变，但是其组合一直变化的时候。

需要生成的对象具有复杂的内部结构，需要生成的对象内部本身相互依赖。

案例：肯德基麦当劳的基本食品：汉堡，薯条，炸鸡，可乐…这些东西是不会变得，但是套餐却各种玩法都有，所以可以通过这些基本对象去构建一个复杂对象。Java中的StringBuilder，是根据基本字符串构建出一个复杂字符串。还有Spring Boot 启动类中的SpringApplication对象也是通过builder构建出来的，Spring Cloud Stream中的消息对象，也是builder出来的。

## 示例程序

需要源码的朋友可以前往github下载：

<https://github.com/aharddreamer/chendong/tree/master/design-patterns/demo-code/design-patterns>

**程序简介**：

我们来看一段使用Builder模式编写文档的程序，编写出的文档具有以下结构：

1）含有一个标题；

2）含有几个字符串

3）含有条目项目

Builder类种定义了决定文档结构的方法，然后Director类使用该方法编写一个具体的文档。

Builder类是抽象类，它并没有进行任何实际的处理，仅仅声明了抽象方法。Builder类的子类决定了用来编写文档的具体处理。

在示例程序中，我们定义了以下Builder类的子类。

TextBuilder类：使用纯文本（普通字符串）编写文档

HTMLBuilder类： 使用HTML编写文档

类清单：

Builder 定义了决定文档结构的方法的抽象类

Director 编写一个文档的类

TextBuilder 使用纯文本编写文档的类

HTMLBuilder 使用HTML编写文档的类

BuilderPatternTest 测试类

**代码：**

**public abstract class** Builder {  
 **public abstract void** makeTitle(String title);  
 **public abstract void** makeString(String str);  
 **public abstract void** makeItems(String[] items);  
 **public abstract void** close();  
}

**public class** Director {  
 **private** Builder **builder**;  
 **public** Director (Builder builder) {  
 **this**.**builder** = builder;  
 }  
 **public void** construct() {  
 *//编写文档* **builder**.makeTitle(**"美好的一天"**);  
 **builder**.makeString(**"从早上到下午"**);  
 **builder**.makeItems(**new** String[]{  
 **"早上好"**,  
 **"下午好"**,  
 });  
 **builder**.makeString(**"晚上"**);  
 **builder**.makeItems(**new** String[]{  
 **"晚上好"**,  
 **"晚安"**,  
 });  
 *//完成文档* **builder**.close();  
 }  
}

**public class** TextBuilder **extends** Builder {  
 **private** StringBuilder **buffer** = **new** StringBuilder();  
 @Override  
 **public void** makeTitle(String title) {  
 **buffer**.append(**"==============Start Text Document===============\n"**);  
 **buffer**.append(**"《"**);  
 **buffer**.append(title);  
 **buffer**.append(**"》"**);  
 **buffer**.append(**"\n"**);  
 }  
 @Override  
 **public void** makeString(String str) {  
 **buffer**.append(str);  
 **buffer**.append(**"\n"**);  
 }  
 @Override  
 **public void** makeItems(String[] items) {  
 **for** (**int** i = 0; i < items.**length** ; i++) {  
 **buffer**.append(**" ·"** + items[i] + **"\n"**);  
 }  
 **buffer**.append(**"\n"**);  
 }  
 @Override  
 **public void** close() {  
 **buffer**.append(**"==============End Text Document===============\n"**);  
 }  
 **public** String getResult() {  
 **return buffer**.toString();  
 }  
}

**public class** HTMLBuilder **extends** Builder {  
 **private** StringBuilder **buffer** = **new** StringBuilder();  
 @Override  
 **public void** makeTitle(String title) {  
 **buffer**.append(**"==============Start HTML Document===============\n"**);  
 **buffer**.append(**"<h1>"**);  
 **buffer**.append(title);  
 **buffer**.append(**"</h1>"**);  
 **buffer**.append(**"\n"**);  
 }  
 @Override  
 **public void** makeString(String str) {  
 **buffer**.append(**"<p>"**);  
 **buffer**.append(str);  
 **buffer**.append(**"</p>"**);  
 **buffer**.append(**"\n"**);  
 }  
 @Override  
 **public void** makeItems(String[] items) {  
 **buffer**.append(**"<ur>"**);  
 **buffer**.append(**"\n"**);  
 **for** (**int** i = 0; i < items.**length** ; i++) {  
 **buffer**.append(**"<li>"** + items[i] + **"</li>\n"**);  
 }  
 **buffer**.append(**"</ur>"**);  
 **buffer**.append(**"\n"**);  
 }  
 @Override  
 **public void** close() {  
 **buffer**.append(**"==============End HTML Document===============\n"**);  
 }  
 **public** String getResult() {  
 **return buffer**.toString();  
 }  
  
}

**public class** BuilderPatternTest {  
 **public static void** main(String[] args) {  
 *//Text Builder* TextBuilder textBuilder = **new** TextBuilder();  
 Director director = **new** Director(textBuilder);  
 director.construct();  
 String result = textBuilder.getResult();  
 System.***out***.println(result);  
 System.***out***.println();  
 System.***out***.println();  
  
 *//HTML Builder* HTMLBuilder htmlBuilder = **new** HTMLBuilder();  
 director = **new** Director(htmlBuilder);  
 director.construct();  
 String htmlResult = htmlBuilder.getResult();  
 System.***out***.println(htmlResult);  
 }  
}

运行结果：

==============Start Text Document===============

《美好的一天》

从早上到下午

·早上好

·下午好

晚上

·晚上好

·晚安

==============End Text Document===============

==============Start HTML Document===============

<h1>美好的一天</h1>

<p>从早上到下午</p>

<ur>

<li>早上好</li>

<li>下午好</li>

</ur>

<p>晚上</p>

<ur>

<li>晚上好</li>

<li>晚安</li>

</ur>

==============End HTML Document===============

参考：

《建造者模式》菜鸟教程网站

《图解设计模式》【日】结城浩著