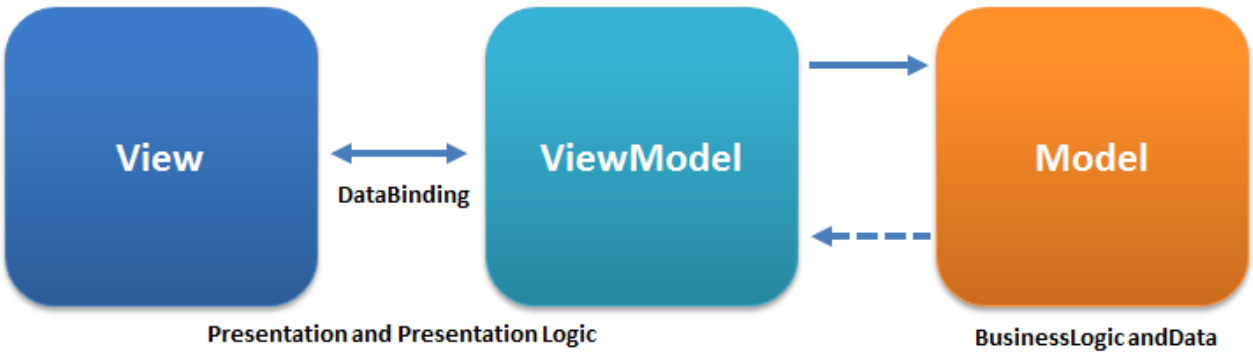
WinUI 3 | Ep.4 MVVM架构



MVVM，全称Model-View-ViewModel，是一种软件架构模式。MVVM架构通过将项目分层，有助于将图形界面的开发与业务逻辑和后端代码分离开来。

MVVM的组成部分

MVVM架构由三大部分组成：

- Model：模型，是业务逻辑存放的地方，也会为ViewModel提供各种数据。
- **View：视图**，也就是用户在屏幕上看到的结构、布局和外观（UI）。
- **ViewModel：视图模型**，是连接业务逻辑和视图的中间层。

View 视图

视图是很简单的一层，负责定义用户在屏幕上看到的结构、布局和外观。在WinUI 3中，我们可以使用XAML轻松的定义一个视图。上一节的`MainWindow`的XAML部分就是一个视图。

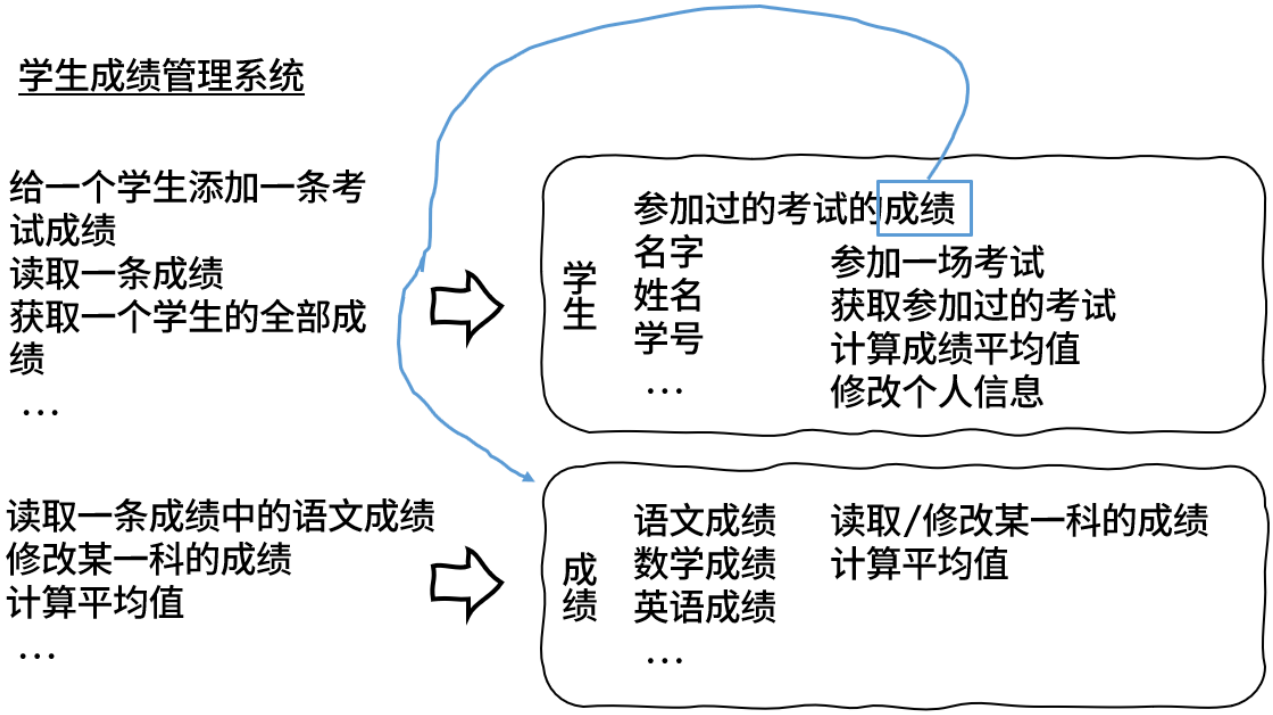
除了屏幕上看到的内容，视图也负责接受用户的输入。比如按下按钮后，视图里的按钮会触发其`Click`事件。

在MVVM中，视图是十分“薄”的一层，也就是说它不会包含任何业务代码，只是简单的定义结构和布局。而视图的后台代码一般没有内容，或者只包含在XAML里表达起来很麻烦很啰嗦的内容，如动画。

Model 模型

视图用来展示数据，但是它并不会拥有逻辑。零散在每一个事件处理函数里的逻辑很混乱，面向对象编程通过对业务逻辑进行建模，并且抽象成一个一个具有行为的对象，将零散的逻辑聚合起来。而这些对象就是模型。

这是一个典型的建模过程。建模完成后，我们可以轻松地将这个模型转换成代码。



模型是应用的核心。如上图所示的管理系统，无论你用什么形式的视图来完成用户交互和数据展示，它都叫学生成绩管理系统。因为真正的业务逻辑处于模型之中，是应用最有价值的部分。因此，我们需要保证模型的高可维护性，不要让它被复杂的UI界面给污染。

模型与视图无关，彼此相隔离。这是MVVM架构的核心思想。这种隔离使得无论你准备以何种形式与用户交互和展示数据，是通过网页，桌面应用，手机app，还是控制台命令行，都不会影响你的模型。

ViewModel 视图模型

我们了解了MVVM中的视图与模型，知道了它们彼此间需要完全隔离。很显然，隔离的两部分不能构成一个整体，我们需要有一个“胶水”将二者粘起来。这层胶水是MVVM架构的独特之处，叫做——ViewModel 视图模型。从名字上就不难猜测，它就是为了连接视图与模型而被设计出来的。

视图与模型的连接是双向的。这不难理解：

- **用户交互：** 视图到模型。用户在视图上所做的操作（按下一个按钮，输入一段文字……）需要最终传递给模型做处理；
- **数据展示：** 模型到视图。模型中的数据需要及时的传递给视图进行展示。

Recap | 计数器

在继续之前，我们先回顾一下，在上一节的计数器中，不使用数据绑定时，是如何把视图与模型连接起来的。

> 虽然上一节中的计数器并没有视图和模型的概念。这里指的是更抽象的视图与模型，可以理解为用户界面和业务逻辑

用户交互：监听按钮的`Click`事件，并且由模型中的事件处理函数处理，完成视图到模型的连接。
数据展示：事件处理函数里直接更改视图的控件的文字，实现模型到视图的连接。

ViewModel处于视图与模型之间，它负责协调视图与模型之间的连接。

ViewModel从模型中提取数据，转换成便于视图展示的形式，提供给视图；并且ViewModel也将模型中业务逻辑抽象成一个个命令，同样提供给视图。因此，ViewModel其实起到的是一种转换器的作用，协调视图与所需的任何模型类的双向交互。

视图则通过一种叫做数据绑定的机制，实现从ViewModel里拿到数据和命令，而不需要在ViewModel中编写与视图相关的代码。

通过转换和数据绑定，ViewModel成功为视图和模型搭建了桥梁。

在上一节中，我们已经初步了解了什么是数据绑定。可以说，数据绑定是MVVM架构的核心技术。接下来我们将更深入的了解它。