# 重构，MVC，单元测试

## 综述：

**本章我们讨论和研究重构代码，让代码变的更易于维护和修改；讨论MVC模式，这在ui设计中应用非常广泛；最后使用一个模块进行单元测试，会有一个单元测试的例子**

## 重构代码：

**重构代码，通俗意义上指的是减少重复的操作，用通用的操作来处理类似的事情**

**05.py中的第一个例子显示了一个简单的前端实例，但结构性很差，它作为后续修改的一个原型；**

**05.py的第二个例子列出了所有按钮的共性 操作，这样的代码就可以被重构，用一个通用的方式来解决类似的问题，可以减少重复操作；**

**05.py的第三个例子使用了一个公用的方法，将每个按钮的共同部分归纳起来，交给一个方法处理；这样做可以增加代码的可读性，减少使用局部变量。**

**05.py的第四个例子使用了分离数据，把按钮名称和事件处理器放在一个容器内，按钮位置可以自动判断，确保新添加的按钮会被放在合适的位置，因为数据会被分离出去，所以方法变成了公用的，在以后的项目中可以重用代码**

**05.py的第五个例子，整合了上面所有的特性，完整的呈现了重构后的代码示例**

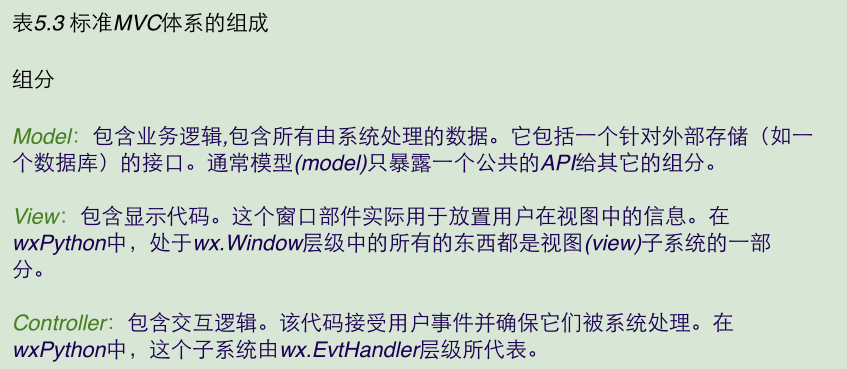
**05是一个完整的代码结构，从最初的啰嗦，重复代码，到一步步优化，最后构成一个完整的代码。它显示了这个完整的过程，也是编写代码从计算机发展开始到今天一步步累积的经验和进步的缩影；**

**（这个例子划重点研究）**

## 视图和模型的分离

### 什么是MVC模式

**MVC模式，它被专门用于面向对象的编程；M表示model，模型负责你的系统所要处理的数据或者是经常被调用的业务逻辑，V表示view，视图负责可显示的对象，展现给用户；C表示controller，控制器负责管理与用户的交互，它位于M和V之间**



**在现代GUI工具包中，view和controller是被集成在一起的，因为显示对象同时也要捕捉用户产生的事件，在wx中，这种关系已经被放置进去了（wx.Window对象也是wx.EvtHandler的子类）。这意味着他们同时具有view元素和controller元素的功能；相比之下，大部分web应用架构的view和controller有更严格的分离，因为它的交互逻辑发生在后台**

**工作原理图如下：**

New display

model

controller

view

event

dispatch

Updated data

Updata notification

**在一个严格的MVC模式中，处理器函数被声明在一个单独的控制器当中，而不是在框架类中；**

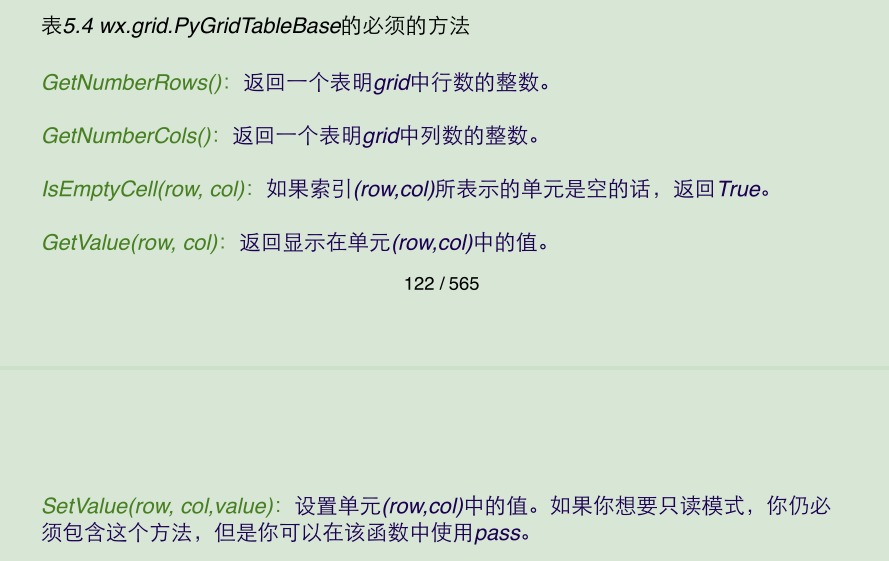
**对于事件的响应，模型对象处理之后返回一个update data,然后交给控制器，控制器通知视图进行自我更新；在一些小型的系统中，通知会直接被视图接受；在wx中，当模型处理数据后，你可以选择让哪一个接受处理的数据，是模型维护的对象的列表接受数据，或是模型关联的视图接受通知**

**对于一个好的MVC模式，尽量不要修改model中的内容，而且，model一定要和view，controler分离；最重要的，尽可能少的在三者之间进行交互，这样保证方法之间明确的界定；**

## 在MVC中使用一个表格空间grid

**代码05.1的第一个程序显示了不使用模型方法的grid控件的使用；这个例子中网格中存放的数据就是model类要处理的数据，网格本身就是显示的view类，现在这两个类放在一起，使得程序更难处理；**

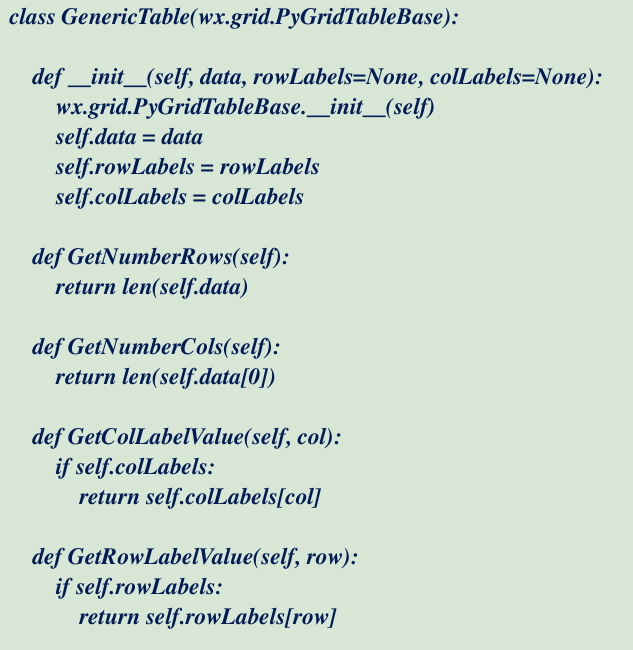
**如果使用模型，要把model和view分离出去，PyGridTableBase类是网格的model，它是一个抽象类，不能被直接实例化，每次使用它时，有五个方法必须被定义，如下图：**

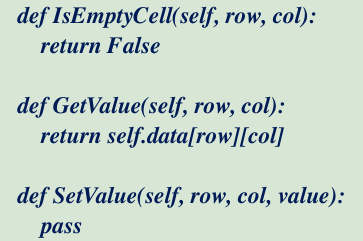


**有两种方法来使用这个类，一种是让你的模型类成为这个类的子类，这种方法适用于数据量不太大时；另一种是单独创建一个PyGridTableBase的子类，然后让这个子类关联你的模型类，这种方法需要模型和视图很好的分离，适用于复杂的数据**

**05.1.py的第二个程序展示了如果将数据与模型分离，数据被分离到一个单独的结构中，模型代码放在一块，通常数据可以放在外部文件，数据库也可以放在这里**

**下面的图片描述了一个通用的处理二维列表的类，它可以处理任何二维列表，包含行标签和列标签**





**05.1.py的第三个程序显示了如何将模型分离出代码单独组成一个模块，把功能性的代码放在一起，显示的代码放到另一边，**

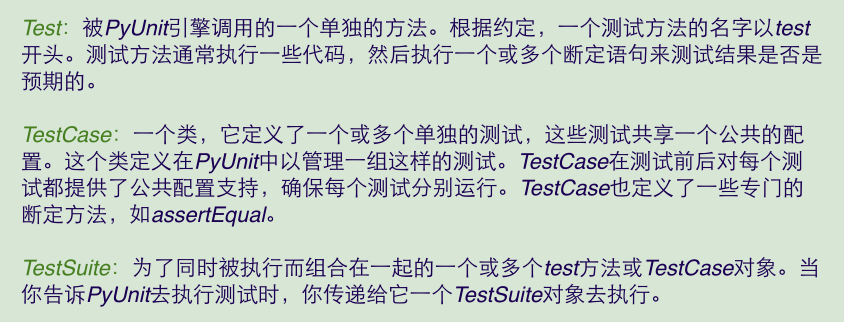
## 自定义模型与编写测试（重难点）

**如何自定义对象呢？首先构造你的模型，不要担心他们如何被显示，然后为数据类做一个公共接口，这个接口提供给视图进行访问；**

**下面的05.1的第四个例子，展示了这种典型的MVC思想**

**单元测试，unittest模块**

**这个模块中包含一个pyunit的测试框架，它由三个部分组成，如下图所示**



**单个的pyunit测试可能有三种结果：success,failure,error,成功表示测试完成，代码执行了你想要做的事，failure表示其中一个断言返回了false，表明代码执行成功了，但没有做你预期的事，error意味着测试执行到某处，引发了一个python异常，表明代码没有执行成功。在单个测试中，failure或error一出现，整个测试就终止了，即使代码中还有多个断言要测试。然后测试的执行将移到下一个单个测试中**

**05.2.py的第一个程序展示了一个unitest的示例**

**End**