结构型模式-模型 视图 控制器模式

概述：
关注点分离(Separation of Concerns,SoC)原则是软件工程相关的设计原则之一，SoC原则背后的思想是将一个应用切分成不同的部分，每个部分解决一个单独的关注点.分层设计中的层次(数据访问层、业务逻辑层和表示层等)即是关注点的例子，使用SoC原则能简化软件应用的开发和维护.

1)模型是核心的部分，代表着应用的信息本源，包含业务逻辑、数据、状态以及应用的规则.
2)视图是模型的可视化表现，视图的例子有，计算机图形用户界面、计算机终端的文本输出、智能手机的应用图形界面、PDF文档、饼 图和柱状图等. 视图只是展示数据，并不处理数据.
3)控制器是模型与视图之间的链接/粘附，模型与视图之间的所有通信都通过控制器进行.
.

MVC的优势：
无需修改模型就能使用多个视图的能力(甚至可以根据需要同时使用多个视图)，为了实现模型与其表现之间的解耦，每个视图通常都需要属于它的控制器. 如果模型直接与特定视图通信，我们将无法对同一个模型使用多个视图(或者至少无法以简洁模块化的方式实现)

示例代码：

In [1]:
```
#coding:utf8

from random import randint
import time

class Model:
    text = ['A man is not complete until he is married. Then he is finished.', 'As I said before, I never repeat myself.',
          'Behind a successful man is an exhausted woman.', 'Black holes really suck...', 'Facts are stubborn things.']

    def get_msg(self, n):
        try:
            value = self.text[n]
        except:
            raise IndexError("index not found")
        else:
            return value

class View:

    def show(self, msg):
        print('the message is %s' %msg)


    def error(self, msg):
        print('found error %s' %msg)


    def select(self):
        x = randint(1, 5)
        return x

class Control:
    def __init__(self):
        self.m = Model()
        self.v = View()

    def run(self):
        n = 10
        while n > 0:
            time.sleep(0.2)
            try:
                n = int(self.v.select())
                msg = self.m.text[n]
            except Exception as e:
                self.v.error(str(e))
            else:
                self.v.show(msg)
            n = n - 1

if __name__ == '__main__':
    c = Control()
    c.run()
```

```
the message is Facts are stubborn things.
the message is Facts are stubborn things.
found error list index out of range
the message is Behind a successful man is an exhausted woman.
the message is As I said before, I never repeat myself.
```

由于在MVC模式中每个部分都有明确的职责，模型负责访问数据，管理应用的状态；视图是模型的外在表现，控制器是模型与视图之间的连接，MVC的恰当使用能确保终产出的应用易于维护、易于扩展.