视图  
  
视图是用户看到并与之交互的界面。对Web应用程序来说，视图就是由HTML元素组成的界面。  
如何处理应用程序的界面变得越来越有挑战性。MVC一个大的好处是它能为你的应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。

模型  
  
模型表示数据和业务规则。在MVC的三个部件中，模型拥有最多的处理任务。例如它可能用象EJBs和ColdFusion Components这样的构件对象来处理数据库。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

控制器  
  
控制器接受用户的输入并调用模型和视图去完成用户的需求。所以当单击Web页面中的超链接和发送HTML表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后用确定用哪个视图来显示模型处理返回的数据。

优点

首先，最重要的一点是多个视图能共享一个模型，正如我所提及的，现在需要用越来越多的方式来访问你的应用程序。对此，其中一个解决之道是使用MVC，无论你的用户想要Flash界面或是 WAP 界面；用一个模型就能处理它们。由于你已经将数据和业务规则从表示层分开，所以你可以最大化的重用你的代码了。  
  
由于模型返回的数据没有进行格式化，所以同样的构件能被不同界面使用。例如，很多数据可能用HTML来表示，但是它们也有可能要用Macromedia Flash和WAP来表示。模型也有状态管理和数据持久性处理的功能，例如，基于会话的购物车和电子商务过程也能被Flash网站或者无线联网的应用程序所重用。  
  
因为模型是自包含的，并且与控制器和视图相分离，所以很容易改变你的应用程序的数据层和业务规则。如果你想把你的数据库从MySQL移植到Oracle，或者改变你的基于RDBMS数据源到LDAP，只需改变你的模型即可。  
  
一旦你正确的实现了模型，不管你的数据来自数据库或是LDAP服务器，视图将会正确的显示它们。由于运用MVC的应用程序的三个部件是相互对立，改变其中一个不会影响其它两个，所以依据这种设计思想你能构造良好的松偶合的构件。  
  
对我来说，控制器的也提供了一个好处，就是可以使用控制器来联接不同的模型和视图去完成用户的需求，这样控制器可以为构造应用程序提供强有力的手段。给定一些可重用的模型和视图，控制器可以根据用户的需求选择模型进行处理，然后选择视图将处理结果显示给用户。

缺点

MVC的缺点是由于它没有明确的定义，所以完全理解MVC并不是很容易。使用MVC需要精心的计划，由于它的内部原理比较复杂，所以需要花费一些时间去思考。  
你将不得不花费相当可观的时间去考虑如何将MVC运用到你的应用程序，同时由于模型和视图要严格的分离，这样也给调试应用程序到来了一定的困难。每个构件在使用之前都需要经过彻底的测试。一旦你的构件经过了测试，你就可以毫无顾忌的重用它们了。  
根据我个人经验，由于我们将一个应用程序分成了三个部件，所以使用MVC同时也意味着你将要管理比以前更多的文件，这一点是显而易见的。这样好像我们的工作量增加了，但是请记住这比起它所能带给我们的好处是不值一提。  
MVC并不适合小型甚至中等规模的应用程序，花费大量时间将MVC应用到规模并不是很大的应用程序通常会得不偿失。