ＪＡＶＡ基础：MVC减少编程复杂性 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022__EF_BC_AA_EF_BC_A1_EF_BC_B6_EF_c104_145418.htm 随着面向对象的语言（如Java）的迅速发展和普及，越来越多的编程人员开始在应用开发中使用这些语言。然而原有的开发语言（即面向操作的开发语言如C＋＋等）在短时间内还不可能退出历史舞台，因此现在就出现了面向对象的语言和传统的面向操作的语言共存的局面。在设计应用中同时使用两类不同的语言（混合语言设计）比过去只使用一类语言会带来许多新的问题，其中复杂性就是混合语言设计中最经常遇到的问题。下面我们探讨混合语言设计中可能导致复杂性增加的地方，以及如何减少以至消除这些复杂性。 复杂性 复杂性是应用开发过程中最令人头疼的一个问题。每当在一个应用中增加一个功能时，它的复杂性通常呈几何级的增长。这种复杂性往往导致程序的开发无法再继续下去。这也是现在为什么许多应用只有Beta版本而没有正式版的原因。 专家将应用开发过程产生的复杂性分为两类，即非本质的（accidental）和本质的（essential）。本质的复杂性是对于解决目标问题所必然产生的复杂性，非本质的复杂性是由于选择了不适当的开发工具和设计工具而产生的复杂性。对于一个功能确定的程序来讲，本质的复杂性是确定的，而非本质的复杂性则是没有限制的。因此，一个应用的开发要想较顺利地取得成功，就需要尽可能地减少非本质的复杂性。 OOD的特点 面向对象的设计（OOD）将一个程序分解成根据具体的对象而设计的一系列元素。这些具体对象的行为和数据以一种叫做“ 类（class

）”的编程单元进行打包。应用程序创建一个或多个这些类的例示，也称为“对象（object）”。类的行为是通过创建对象之间的关系组合在一起的。OOD允许开发者用两种主要的方法来控制复杂性的增加。第一，OOD定义严格的出口语义，这允许开发者隐藏实现的细节，并且明确说明什么方法是其它的对象可以访问的。这个信息隐藏使得可以对大部分的代码进行修改而不影响其它的对象。第二，OOD将对象之间的关系分为四类：继承、包容、使用和协调。适当地使用这些关系可以大大减少应用开发过程中本质的和非本质的复杂性。如，继承是产生面向对象设计中可再使用的主要因素。这个再使用性是通过代码共享和多态性获得的。这种再使用可以大大减少应用的本质的复杂性。包容允许一个类的用户在使用包容器时忽略被包容的类（class）。这个简化使设计者能够大大减少应用的非本质的复杂性。可视化接口在OOD方面的不足 许多程序都需要可视化接口，这些接口由对话框、选单、工具条等组成。这些可视化接口的增加会引进OOD设计的不足，使得一个好的面向对象的设计走向反面。可视化接口有三个属性可能会给应用开发带来麻烦。第一，可视化接口提高了传统的面向操作的拓扑结构。用户产生接口事件，如开关按键和列表框选择等，受到程序的一个模块的驱动并且用来对静态的数据进行操作。在设计中将这面向操作的拓扑结构同一个面向对象的设计混合在一起将导致对象之间的大量的杂合。第二，用户接口通常对于同样的信息经常会需要许多不同的显示。如，一个客户选择列表框可以包含一个客户的名字和电话号码以及许多其它客户的名字。当用户选择某个特定的客户后，他／她的名字和电话号码及其它

全部相关的信息都会详细地显示出来。 除此之外，一个简单的程序可能具有不同的用户接口。如一个银行账户系统有一个接口用于出纳员来访问账户平衡、存款和取款，而监督者的接口则包含另外的信息并加上账号管理的功能。这些不同的接口很容易导致类的扩展。 最后，可视化接口在整个设计阶段还会进行较大的改变。这些改变包括完全重新安排用户与系统的交互操作等。可视化接口的这些改变即使在最好的设计中也会增加应用开发的复杂性。 