解决方案结构考试：逻辑设计 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/330/2021_2022__E8_A7_A3_E5_86_B3_E6_96_B9_E6_c100_330675.htm 作为刚刚出现的MCSD.NET认证考试的一部分，解决方案结构考试（70-100）已经经过了修改。新的考试“ 分析要求和定义Microsoft .NET解决方案结构 ”（70-300）于2003年2月正式启动了，它涵盖了软件开发中最困难的一些内容，它会带你超越设计平台的代码，深入开发应用程序的原因和过程。 本篇学习指导是根据我参加这门考试测试版之后的印象而写成的，因此你会了解到如何实现各种设计任务包括需求的收集、技术结构、概念和逻辑设计、数据造型、用户界面和物理设计一切都是和Microsoft .NET相关的。 设计解决方案的一个重要部分是把你原始的概念转变成用于系统的逻辑结构。这个转变过程涉及辨别出逻辑设计的要点、在各种类型的应用程序接口里进行选择，然后再使用这些信息创建一个宽泛的设计准则，这些准则会区分构成解决方案的组件和模块。最后，你需要回顾一下你所主张的解决方案，以确保商业条例被正确地集成进了逻辑设计，还能测试这个逻辑设计对性能、可访问性、安全、可伸缩性、可维护性和可扩展性等目标的影响。 应用程序接口的类型 当一个客户提出需要为他们设计能够完成特定任务的应用程序时，例如，“ 我需要能够让我管理联络信息的东西。你能帮我制作一个吗？ ”，我的第一反应就是回答：“ 当然了，我有144种方法来实现这个目的。但是我应该选择哪一种呢？ ”当你在把概念设计转变成逻辑形式的时候，你就需要考虑应用程序接口的类型，以及你

能够使用的多种方法里哪些最适合于你的解决方案。在大多数情况下，你最终会把下面这些应用程序类型结合起来： 桌面应用程序 Web应用程序 分层应用程序 协作应用程序 桌面应用程序 桌面应用程序是标准的基于窗体的应用程序，所有图形用户界面（GUI）的用户对这样的应用程序都很熟悉。这些类型的应用程序常常被分为下面这几类： 单文档界面（SDI）应用程序是最简单的类型，它能够允许用户在应用程序单个实例里打开一个活动的窗口。想一想：Word（原来）的版本只能让你一次编辑一个文档。 多文档界面（MDI）应用程序能够同时打开多个活动窗口，常常是在一个主要的父窗口中。父窗口菜单的可用选项会根据活动窗口的可用功能而改变。Word和Excel都是MDI应用程序的经典范例。 控制台应用程序在命令提示行运行。这些常常都是几乎不需用户交互操作的系统工具或者服务。 基于对话框的应用程序 在Windows里常常指的是向导。这些应用程序作为工具来运行，允许用户回答一些问题或者完成一些步骤，从而执行复杂的线性任务。 Web应用程序 Web和桌面应用程序相比有很多优势，包括管理集中、升级容易，以及客户端统一的特性。其唯一的不足之处是，如果你无法访问服务器，它们常常就无法工作。但是，就是这个问题也被.NET涉及到了，包括对连接断开的Web 应用程序甚至是连接断开数据库的支持。 分层应用程序 在分层的解决方案中，组件按照功能被分层，不同的层常常位于不同的计算机上。在设计分层应用程序的过程中，当你把概念设计转换成逻辑设计时，你会希望区分开它所需要的各种层。分层应用程序的巨大优势包括其可伸缩性和易维护性。其不足之处就是复杂性的问题。 协作应用

程序 最后也是最复杂的应用程序类型是协作应用程序。这种类型应用程序的一个例子是微软NetMeeting的白板特性，它能够允许多个用户同时在白板上写字画画。白板的所有用户都能够实时地看到任何变化。Visual Studio .NET能够允许许多开发人员在同一时间开发同一个应用程序，因此它是协作应用程序的另一个例子。 逻辑、扩块和基于组件的设计 一旦建立好了你概念设计的要点并辨别了按照这一概念所构建的应用程序类型，你就有了解决方案的基础。然后你就可以开始设计组成可交付产品的真正组件和服务了。就和所有的设计元素一样，组件和服务的定义在编写代码以前都应该能够被仔细地归档和考虑。 组件 如果你告诉某个程序员你需要一个组件，那么你会获得一个ActiveX控件、Java的类或者是.NET的组件，具体是哪者要依据平台的不同。但是如果你告诉一个软件设计师同样的事情，你会获得不同的东西。对于软件设计师而言，组件就意味着任何一小块解决方案，这个方案被作为一组功能和特性分离开来，并和这一解决方案的其他组件相关。例如，结构组件可以是一组数据存储过程或者是能够放在CD上的可重新分布的客户端程序。它也可以是用户需要用来使用最终解决方案的一组技巧。你需要像设计师一样思考组件，这样才能充实你的组件设计。 服务 程序员把服务理解为Web服务、COM 或者协议服务器。但是，对于软件设计师而言，服务常常意味着向个人或者机构支付费用，用以获得对某些需求的支持。这笔费用可以是每年支付给VeriSign的500美元，用以获得代码签名的认证；也可以是每月支付给Internet服务提供商（ISP）的250美元，用于获得窄带的静态IP连接。如果你的解决方案需要这种类型的支持

，你就希望确保这一点在解决方案的提案中得到了充分的叙述。 把商业条例集成到对象设计里 不论你的解决方案是一组组件、一套服务或者两者的组合，你都希望确保它集成进了机构的商业条例：这些条例会定义或者限制你解决方案所关注商业的特定方面。这一过程很容易就会成为导致整个解决方案失败的地方。无论这个解决方案设计得有多稳固多周密，如果它违反了重要的商业条例例如“ 每笔购买订单都必须由管理层的一员认可 ”它就会被认为是失败的。 逻辑设计对PASS ME目标的影响 逻辑、模块和基于组件解决方案的设计的最后一部分会发生在设计过程的最后，并会向后延续一段时间。你有了一个可以接受的解决方案，但是现在你会希望确切地知道这个方案能够在多大程度上满足你的期望，甚至超过你的预期。这一阶段就是你收集其优势确定自己的解决方案是否满足了PASS ME目标的时侯。它运行得是否良好？它是否能够被需要访问的人轻易地访问到？它是否稳定和安全？它是否易于维护？它是否能够满足未来的要求？(BUILDER.COM) 