有关Office的对象模型 PDF转换可能丢失图片或格式，建议阅读原文

https://www.100test.com/kao_ti2020/221/2021_2022__E6_9C_89_E5_85_B3Offi_c98_221255.htm 计算机等级考试训练软件《百宝箱》 Office 应用程序可以公开 COM 接口。例如，虽然您可以专注于 Word Document 对象并以逻辑方式独立地实例化该对象，但在实际中会执行 Word 进程来提供此对象。但如果您确实需要一个开发平台，那么虽然在某种特定情况下您只使用一个组件，您仍然会希望该平台的其余部分保持可用。您希望该组件使用所需的任何其他平台服务，而不必亲自手动实例化这些服务。 在 Office 解决方案环境中，哪里适合进行组件化？许多客户构建其自己的特定于域、可重复使用的组件，并将其放在 Office 的上层。例如，特定于某种投资银行业务功能的组件可能会使用 Excel 计算引擎。此组件的解决方案端接口是特定于域的，而平台端接口则特定于 Excel。这种抽象允许该组件针对 Excel 的多个版本进行工作，而这通常恰恰是客户所要求的。 或者，您可以构建一个可重复使用、不影响宿主应用程序的通用组件。例如，看一看 UserControl，它使用 Web Services 在自定义任务窗格中显示数据。这个控件可以在多个 Office 应用程序中重复使用。 Office 应用程序随着时间的推移已经发生了演变，它们具有非常不同的功能集，因此不可避免地提供不同的对象模型。尽管如此，但应用程序之间仍然存在一些一致性。例如，对象模型大多具有层次结构，并且根位置通常具有一个 Application 对象。在 Word 中，您可以从该 Application 对象开始查找 Documents 集合，深化各个 Document 对象，然后深化各个 Range 对象。同样，

在 Excel 中，您可以从 Application 对象开始下移到 Workbooks 集合，找到该集合中的 Workbook 对象，然后获取工作簿中的各个 Range 对象。 此外，Visual Studio Tools for Office 还在一致的强类型化对象模型上进行分层。您只需看看两个截然不同的 Visual Studio Tools for Office 解决方案（例如基于 Excel 文档的解决方案和基于 Outlook 应用程序的解决方案），就可以了解其一致性。两个解决方案都具有简单的 Startup 和 Shutdown 方法，因此，您可以集中精力于业务需求，而无需考虑宿主应用程序的各个特征。在仍然可以完全访问基础对象模型的同时，您还可以选择在更高的抽象级别下进行工作。 Office 的每个新版本都保持了非常高的向后兼容性。Office 应用程序是 COM 服务器程序，COM 提供了多个规则（如接口永久性），从而防止出现版本不兼容问题。Office 通常按照这些规则运行。所有 Office 接口都要么为双接口，要么为纯调度接口。纯 vtable 接口（通过扩展，也包括双接口）为永久性接口，但调度接口不是，因为可以在运行时（因而也在后期绑定）发现其可用功能集和签名。在发布新版本时，Office 团队通常会利用这一特点向接口的末尾添加额外的方法和属性。他们通常还会向现有的方法中添加新的可选参数。通过这些技术，Office 可以保持向后兼容性。 保持版本复原能力的另一种技术是使用松散类型化。典型的示例为 COM 加载项必须实现的 IDTExtensibility2 接口。当 Office 应用程序加载某个加载项时，该应用程序将调用 IDTExtensibility2::OnConnection，并传入表示宿主应用程序的松散类型化对象。在 .NET 代码中，此参数的类型为 System.Object，而在 C 中，它是一个通用的 IDispatch 指针。

在运行时，它实际上是指向由宿主应用程序所提供的 Application 对象的指针。在构建传统的共享加载项时，向导生成的代码对宿主没有任何影响。这具有两方面的好处：它允许该加载项在支持 COM 加载项的任何应用程序中运行，并允许该加载项在任何这类应用程序的任何版本中运行。 这种对宿主无影响的模型的缺点是松散类型化和后期绑定通常所具有的缺点：不能为所涉及的特定类型提供设计时或编译时支持，因为这些类型在运行时以前是未知的。这极易使编写的代码在运行时失败。另外，后期绑定会带来性能降低，因为每个方法调用都必须完成发现过程，以确定是否确实能够在运行时找到匹配的方法。 在使用 .NET Framework 开发加载项时，可以继续使用此松散类型化，但除非加载项本身只提供对宿主无影响的功能，否则所具有的优势将十分有限。只要加载项需要提供特定于宿主的功能，您就有可能使用宿主的特定于版本的主互操作程序集 (PIA)。另外，您还可以使用 Visual Studio Tools for Office 通过 .NET 代码构建加载项。Visual Studio Tools for Office 强制要求强类型化，不但可以为您带来编译时类型检查的好处，还可以为您带来设计时 IntelliSense&reg. 和自动完成的好处。强类型化可避免后期绑定所带来的性能开销。 强类型化是否意味着丧失版本的复原能力？不一定。您可以针对某一版本的 Office 构建一个松散类型化的加载项，该加载项几乎肯定可以在更高版本的 Office 中完全正常地工作。使用 .NET Framework 时，虽然行为会稍稍复杂一些，但您可以得到同样的结果。针对特定版本的 Office PIA 构建的 .NET 加载项也应能在更高版本中正常工作。实际上有两种方法可以实现这一点：加载项使用构建时所

针对的 PIA 版本，或者为 PIA 部署绑定重定向，以便加载项在运行时使用对应于所安装的 Office 版本的更高版本的 PIA。