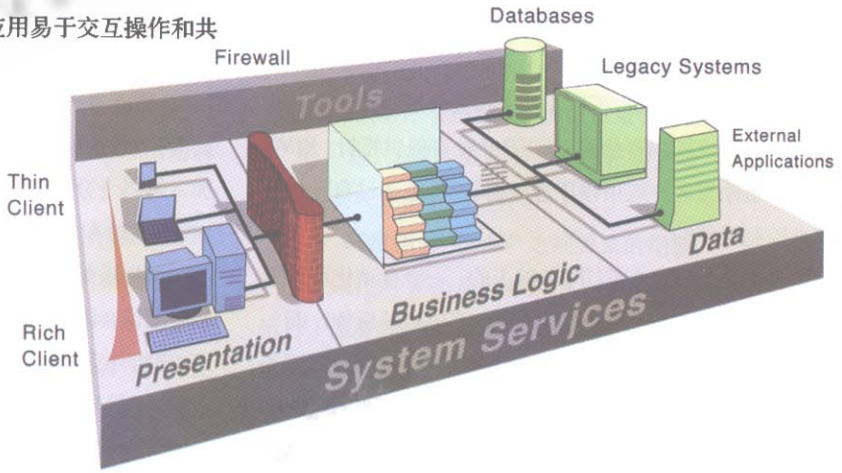
以组件对象为中心的动态Web开发技术问题

北京工业大学计算机学院 软件学科部 陈文博 夏长虹

以可重用组件的思想构造大型应用软件系统的思想方法由来已久，但在技术和工艺方面趋于成熟则是近几年的事情。在以组件对象为中心的动态Web开发中，通过定制的组件进行特定事务处理是最可行的方法。本文着重介绍了组件对象的开发方法、组件在Web环境中的使用，并简略介绍了两种实用的软件开发辅助工具。

DNA思想与组件对象COM

以可重用组件的思想构造大型应用软件系统的思想方法由来已久，但在技术和工艺方面 趋于成熟则是近几年的事情。根植于组件方法的Web开发技术正日益受到业界的重视。本文 将从工艺技术的视角讨论组件对象在动态Web开发方面的几个值得关注的问题。微软提出的DNA概念是借助生命科学中脱氧核糖核酸（DNA Deoxyribonucleic Acid）的寓意来诠释现代企业信息开发的真谛，并将其称之为数字神经系统。Microsoft提出DNA还有其具体的技术考虑，Windows DNA(Windows Distributed InterNet Applications Architecture)在技术层面是指分布式互连网应用结构。随着互联网络技术的发展，传统的客户机/服务器两层结构愈发表现出明显的局限性。为适应更快更复杂的事务处理任务和快速开发的需要，一种新的分布式应用设计方法便应运而生。Windows DNA是以“表现层／事务逻辑层／数据服务层”三层体系结构为构架的，并将COM概念应用于Internet。系统利用COM组件在中间层进行事务逻辑服务，处理各种复杂的商务逻辑计算和规则演算。这些服务都通过COM以一种统一的方式展示出来，使诸多应用易于交互操作和共享组件。系统结构如下图所示



开发方法与技术细节

COM组件是遵循COM规范编写、以Win32动态链接库（DLLs）或可执行文件（EXEs）的形式发布的二进制代码。它具有不依赖特定开发工具、重用性高、运行效率高、便于使用和管理等特点，由组件“组装”的应用系统具有高度的灵活性和强大的生命力。近几年来，组件在软件开发中得到了广泛的应用。尤其是以Windows DNA思想架构为基础，将组件应用于Internet进行各种事务处理，摆脱了以简单脚本为主的开发局面，使组件在Web开发领域显示了强大的功力。人们既可以十分方便地使用已有的COM组件，也可以根据特定的商务逻辑需要创建自己的COM组件。本文将从开发方法和技术细节的角度来讨论定制组件的有关问题，涉及COM组件的规划、接口设定、详细设计、具体实现、测试和发布等方面内容。

1．组件的规划原则

创建自己的COM组件之前，首先必须从功能方面弄清楚组件的应用需求。一般来说，COM组件的规划应遵循以下原则：

(1)用户对于数据库的访问请求应通过COM组件实现。将访问权限授予COM组件，可有效保证访问数据库的安全性；把对每个用户的连接变成和COM组件的连接，以避免数据库资源的浪费和系统崩溃的危险。

(2)组件粒度不宜过大，争取每一个COM组件实现某一个或一类相似的应用内容，而不必追求其功能的过分庞大。保证每个组件对象完成的商务逻辑功能相对单一，有助于重用机制的发挥和调节DNA的适应性。

(3)COM组件与用户的接口应尽量简单、友好。COM组件如果是可视的(ActiveX控件)，则只能有一个可视化界面。

2. 组件接口设计的考虑

组件的接口在整个应用系统中起决定性作用。一般接口应具有较高的通用性，以提高整个应用系统的复用能力，同时还要兼顾简单和实用性。举例说，如果我们需要定制一个组件访问特定的数据库并将结果返回给用户。若只是为该系统设计，可以将库名、表名、访库语句全部封装在组件内部，这样组件的接口十分简单，但组件本身的重用性很小。若将库名、表名等作为组件的接口，组件的重用性大大提高了，但接口复杂了，不利于组件的拆换。建议的做法是：如果希望系统有再次开发或移植的潜力，可以将库表名作为接口，访库语句应封装在组件内部，否则将访库权限交给用户，既不利于数据库的安全性，也会给应用程序带来很大的负担。

组件的内部实现细节不能反映到接口中，接口同内部实现细节的隔离程度越高，组件或应用发生变化对接口的影响将越小。

3. 用VB实现组件的原理简介

对于Internet应用而言，完成事务逻辑处理计算任务的组件以MS Visual Basic进行开发是常用的选择方案。其开发迅速，调试方便，编译之后的组件以二进制的形式发布，来构建Windows DNA。

在严格意义上讲，COM组件的接口是一个包含一个函数指针数组的内存结构，而VB并没有提供指针，用VB开发的组件与客户应用程序进行通信时，是通过自动化方式来控制组件的。

自动化是建立在COM基础上的，一个自动化服务器实际上是一个实现了IDispatch接口的COM组件；而一个自动化控制器是一个通过IDispatch接口同自动化服务器进行通信的COM客户。COM接口提供的任何服务都可以通过一个标准的IDispatch接口实现，该接口可以接受一个函数的名称并执行它。接口有两个函数GetIDsOfNames和Invoke。GetIDsOfNames将读取一个函数名并返回其调度标识DISPID(每个函数的唯一标识)；自动化控制程序将DISPID作为函数指针数组的索引传给Invoke成员函数，以找到该索引对应的函数地址。

值得注意的是，接口的所有成员都是函数，VB制作的组件却可以支持“属性”的概念。事实上，COM接口是通过“Set”和“Get”类函数模拟对其成员函数的访问的，VB中支持的属性实际上就是VB程序员可以当成变量对待的“Set/Get”函数。当然对于一般的组件开发人员，以上组件的接口实现是透明的，我们可以暂且搁置底层细节。

4. 利用VB进行组件制作的步骤

在动态Web的实际应用中，一般常用的COM组件有两类：ActiveX DLL和ActiveX Controls。ActiveX DLL用于服务器端中间层处理商务逻辑，ActiveX Controls用于在客户端表现结果。

若需要创建在服务器端的ActiveX DLL，可按以下步骤进行：在新建工程中选择新建ActiveX DLL；然后在工具菜单中选择“添加过程”，若该DLL具有某方法，则选择添加“子程序”或“函数”，若该DLL具有某属性，则选择添加“属性”；最后编写具体实现的代码。需要注意，ActiveX DLL工程和普通的EXE工程不同，它无法通过公共变量与外部应用程序进行通信，而只能在内部通过方法的调用实现“属性”。

开发客户端的ActiveX控件和ActiveX DLL的过程类似，可按以下步骤进行：在新建工程中选择新建ActiveX控件；在UserControl的窗体中绘制用户界面。然后，在“应用程序向导”中，选择ActiveX控件向导并选择控件中支持的事件、方法和属性，为控件添加自己的事件、方法和属性。接下来，将控件的事件、方法和属性映射到组成控件的成员中。最后在代码区域编写代码，ActiveX控件向导自动为用户控件初始化属性，并自动生成部分代码，我们只需在此基础上进行部分的修改和添加，编码十分简便。

5. 对ActiveX DLL组件进行设计时测试

当构造好自己的ActiveX组件后，并不要忙于在应用环境测试，因为在Web环境中测试ActiveX组件实为下策。当组件工作出错时，系统不能对组件内部的错误进行定位，而只是简单地提示组件对象创建不成功。我们可以利用VB所提供的配套开发工具对组件进行本地测试。当用VB完成ActiveX DLL的制作后，在该工程中添加一

个标准EXE工程组成工程组，将此标准EXE工程设置为启动工程。编写代码，在标准EXE工程中引用ActiveX DLL，调用组件的方法和属性，与之进行通信。

在制作组件时进行设计时调试是一个非常好的方法。如果组件构造中有错误，可以利用VB提供的单步跟踪、断点设置、立即窗口显示等功能很快的发现并改正错误。它简单有效，大大降低了开发过程的返工率，而且避免了反复注册或打包而造成的注册表信息混乱。最后可以编写一段简单的ASP脚本，通过下载页面的形式对其进行全面的测试。

6．ActiveX控件的测试与发布

要保证ActiveX控件能正常工作，必须对它进行全面测试，包括测试其每一个属性，方法和事件。首先，对ActiveX控件进行本地测试，在标准EXE工程的窗体中嵌入该ActiveX控件，并添加另外一些控件对象，如文本框和命令按钮来显示、激活或调用ActiveX控件中开放的属性、事件和方法。然后可再利用Visual C++中的控件测试容器（Control Test Container)对控件的属性、事件和方法进行全面测试，它支持对控件的注册、注销及重注册。当我们发现某个已注册的控件发生故障或不再适用时，应及时将其从注册表中删除，利用ActiveX控件测试容器删除控件比直接运行Regedit进行删除要方便和安全。

制作好的ActiveX控件编译为*.ocx后，应使用Setup Wizard建立一个包含所有所需文件的可发行版本。在本地使用成功的ActiveX控件，不见得按下载方式工作也能成功，很可能因伴随的支持文件不足而失败。

Setup Wizard可以对应用软件进行创建、设置和发布，对控件所在项目进行分析，确定该控件必须包含的支持文件，使经初始化后带有*.cab文件的页面能自动下载到客户端创建对象。*.cab文件是一种特殊的文件，其内部包括ActiveX控件及所需的所有必要的支持文件。当用户第一次申请含有该控件的主页时，Web浏览器从*.cab文件中读取这些文件并和已有的文件进行比较，以决定需要下载的文件，实现按需下载。

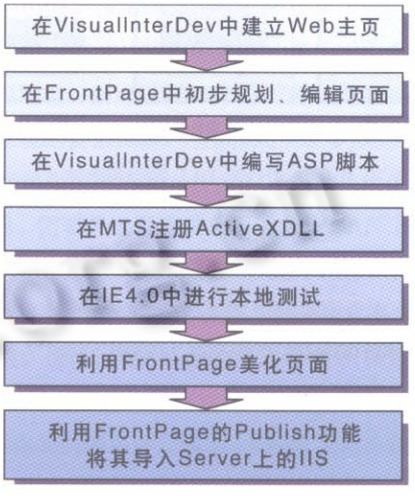
一个容易被忽视的细节是控件安全性的设置。可使用Safety功能来设置项目中每个ActiveX控件的等级，以确定控件对初始化和脚本运行都是安全的。如果不设置控件的安全性，用户的Web浏览器可能会提出警告。将控件设置为“安全”，是以开发者的信誉作为保证的。这就要求每个有责任心的开发人员反复、多方面地对控件进行测试。由于控件是发布在Internet上的，难免一些不道德的编程人员制作一些恶意的控件危害用户计算机。一般来讲，若要将所制的控件在Internet上发布，应到微软等站点进行数字认证，用以昭告世人：该ActiveX的安全性由创建者负责。

组件对象COM在Web环境下的部署

ActiveX DLL形式的COM组件在Web环境下应用时，它通过ASP脚本的调用而显现自身的价值，通过MTS的管理而发挥其作用。在最后实现阶段，把经过测试的COM组件集成到动态Web的ASP页面脚本中，并进行系统测试。

1．集成步骤

在集成过程中，一般先通过Visual InerDev建立Web站点，连接服务器并在Web服务器的根目录下建立站点主页，然后交替使用Visual InterDev和FrontPage工具编写ASP脚本，引用事务逻辑组件并修饰页面。最后在FontPage Explorer下利用其提供的Publish功能将其导入IIS服务器，从而使客户用前端的浏览器从服务器上下载这些应用程序。具体过程如下图描述：



由于应用系统由若干个封装好的组件组装而成，只需着重测试组件与应用系统的接口是否将它们良好地“对接”在一起。若某个处理要求不能正确完成，则只需修改或调换实现该功能的组件。

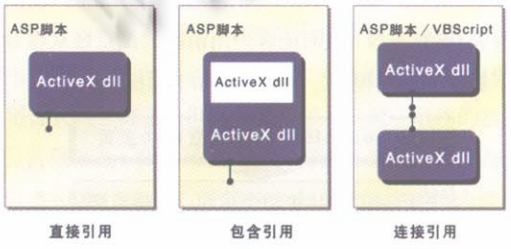
2．COM组件 与ASP之间关系的把握

在以往的Web应用系统中，比较流行的办法是编写运行于服务器端的ASP脚本进行事务逻辑处理。Active Server Pages(ASP)是服务器端的脚本编写环境，用来创建动态Web页面应用。虽然不用定制COM组件，仅使用ASP也能实现绝大部分的服务器端事务处理，但相对组件而言，仅凭ASP脚本来构建应用系统存在着明显的

局限性：首先，解释脚本比运行一个对象要慢得多，不利于向大范围的用户推广；其次，脚本不能从功能中分离出来，用脚本编写的事务逻辑划分不明晰，增大了发现错误的难度，无形中提高了排错的开销；而且组件是可重用的，而脚本则谈不上真正意义上的重用性。

在现今的三层体系结构中，事务逻辑单元是以一个个COM组件的形式在中间层执行，大大减少了服务器端运行的ASP脚本。但这并不意味着ASP将失去其原有的光彩。我们不仅可以通过编写ASP脚本动态生成HTML，ASP还可以作为“黏合剂”将各个COM组件“黏合”在一起，并负责应用系统和COM组件间参数的传递。

引用组件ActiveX DLL的方法很多。它既可以在ASP脚本中被直接引用，也可在另一ActiveX DLL中引用，还可通过编写VBScript及ASP脚本将若干个ActiveX DLL连接起来。



组件ActiveX DLL的引用方法

直接引用：

```
<%
...
Set obj=Server.CreateObject("ObjectName")
...
%>
```

包含引用：

在定制的ActiveX DLL的VB程序中使用以下代码引用另一个ActiveX DLL（ObjectName）

```
Dim obj As Object
Set obj=New ObjectName
```

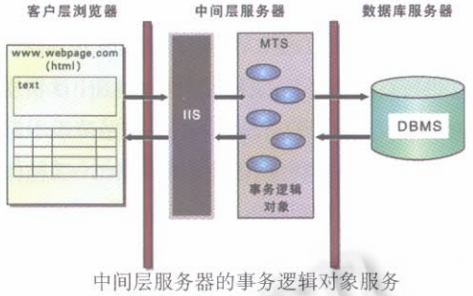
连接引用

在VBScript或ASP脚本中使用以下代码：

```
<%
...
Set Obj_1=Server.CreateObject("Object1_Name")
...
Result=Obj_1.Method
...
Set Obj_2=Server.CreateObject("Object2_Name")
Obj_2.Property=Result
...
%>
```

3. 利用MTS改善Web应用的管理

随着组件技术的逐渐成熟，企业会经常使用COM组件建立独立的事务逻辑处理单元，并将其嵌入到页面文件中，开发出具有Internet通信能力和强大商业计算能力的Web应用系统。但在网页中嵌入ActiveX DLL时，由于多个用户会同时访问该主页，势必容易造成服务器网络的拥塞，从而使系统性能下降。由于IIS只提供了对页面的多线程支持，而不负责对组件本身的服务和管理。MTS服务恰好克服了这一缺陷，使IIS只担负页面的服务和管理工作；MTS则负责页面所嵌入组件的服务和管理工作。两者结合起来组成中间层的应用服务器，原理如图所示：



中间层服务器的事务逻辑对象服务

MTS是一个基于组件的事务处理系统，用于配置及管理高性能、可测和可靠的Internet及Intranet企业级应用。MTS的强大功能主要表现在其对组件和访库的支持上，它提供了包括数据连接缓冲、线程管理、事务服务等多项服务。所有ActiveX DLL组件置于MTS的统一管理之下，很好地解决了多客户端利用组件频繁访问后台数据库等一系列问题。

MTS对组件的管理是通过Transaction Server Explorer完成的。所有运行于服务器端的ActiveX DLL都应在MTS中注册、创建软件包。所谓软件包是在同一进程中运行的组件集合，不同软件包中的组件以进程隔离的方式运行在各自的进程中。包装组件的依据应当是：应尽量把共享资源的组件分配在同一软件包内；考虑到软件包中各个组件所共享的资源的类型，可以把那些共享“昂贵”资源（如对某个特定数据库的连接）的组件编为一组。无论何时重新编译注册过的ActiveX DLL，都需要及时刷新Transaction Server中组件的信息。■