Microsoft ActiveX Data Objects (ADO)

Table of Contents

1. Microsoft ActiveX Data Objects (ADO)

Microsoft® ActiveX® Data Objects (ADO) 使您的客户端应用程序能够通过 OLE DB 提供者访问和操作在数据库服务器中的数据。

ADO 程序员参考

ADO 支持用于建立基于客户端/服务器和 Web 的应用程序的主要功能。其主要优点是易于使用、高速度、低内存支出和占用磁盘空间较少。ADO 同时具有远程数据服务 (RDS) 功能，通过 RDS 可以在一次往返过程中实现将数据从服务器移动到客户端应用程序或 Web 页、在客户端对数据进行处理然后将更新结果返回服务器的操作。

有关 ADO 的详细信息，请参阅 ADO 概述(See 1.1)。请在 http://www.microsoft.com/data/ado 中参阅 Microsoft 有关 ADO 发布说明的 Web 页。

ADO Extensions for DDL and Security (ADOX) 程序员参考

ActiveX Data Objects Extensions for DDL and Security (ADOX) 将 ADO 扩展为包括创建、修改和删除模式对象，如表格和过程。它还包括安全对象，用于维护用户和组，以及授予和撤消对象的权限。

有关 ADOX 的详细信息，请参阅 ADOX 概述(See 1.2)。请在 http://www.microsoft.com/data/ado 中参阅 Microsoft 有关 ADOX 发布说明的 Web 页。

ADO MD 程序员参考

ActiveX Data Objects (Multidimensional) (ADO MD) 将 ADO 扩展为包括指定到多维数据的对象，并允许浏览多维模式、查询立方和检索结果。

有关 ADO MD 的详细信息，请参阅 ADO MD 概述(See 1.3)。请在 http://www.microsoft.com/data/ado 中参阅 Microsoft 有关 ADO MD 发布说明的 Web 页。

1.1 Microsoft ADO 程序员参考

Microsoft® ActiveX® Data Objects (ADO) 使您能够编写通过 OLE DB 提供者对在数据库服务器中的数据进行访问和操作的应用程序。其主要优点是易于使用、高速度、低内存支出和占用磁盘空间较少。ADO 支持用于建立基于客户端/服务器和 Web 的应用程序的主要功能。

ADO 同时具有远程数据服务 (RDS) 功能，通过 RDS 可以在一次往返过程中实现将数据从服务器移动到客户端应用程序或 Web 页、在客户端对数据进行处理然后将更新结果返回服务器的操作。RDS 以前的版本是 Microsoft Remote Data Service 1.5，现在，RDS 已经与 ADO 编程模型合并,以便简化客户端数据的远程操作。

有关 ADO 及 RDS 如何与之集成的详细信息，请参阅 ADO 入门(See 1.1.2)。

1.1.1 ADO 的新增内容

在该版本中包含多个新功能和附加文档。

新功能

查找和索引

Seek(See 1.1.4.4.42) 方法和 Index(See 1.1.4.6.34) 属性在记录中实现快速并基于索引的行定位。

行更新和同步

针对通过 JOIN 操作创建的 **Recordset** 对象，可以自定义对行更新和同步的控制。七个新的动态属性(See 1.1.4.7)控制五个现有方法的行为。

附加文档

Microsoft OLE DB Persistence Provider

Microsoft OLE DB Persistence Provider(See 1.1.5.8) 与记录集对象的 Save(See 1.1.4.4.41) 和 Open(See 1.1.4.4.33) 方法一起，可以将 **Recordset** 保存和恢复到文件。在该版本中，**Recordset** 可使用 Extensible Markup Language (XML) 格式保存，该格式是 Internet 上定义用于数据传输的用户指定标记的标准。

Microsoft Data Shaping Service for OLE DB

Microsoft Data Shaping Service for OLE Db(See 1.1.5.7) 支持数据构形(See 1.1.3.7)，现在可以：

- 重新构形以前已构形的记录集。为了支持该功能，**Recordset** 对象现在具有在连接期间存在的 **Name** 属性。
- 对构形后 **Recordset** 的任何等级上的列执行合计运算，而不仅仅针对父的直接子。该功能使用完整的子集名来形成到预定等级和列的路径。
- 参数化的 COMPUTE 命令在 PARAMETER 子句和参数之间可以插入任意数量的 COMPUTE 子句。

Microsoft OLE DB Remoting Provider

Microsoft OLE DB Remoting Provider(See 1.1.5.9) 服务提供者已成为标准服务提供者，并拥有新的动态属性：增强的性能、更多的用户控制和向后兼容 ADO 2.0。

Microsoft Cursor Service for OLE DB

Microsoft Cursor Service for OLE DB(See 1.1.5.10) 服务组件增补了数据提供者的游标功能。结果，用户可从所有数据提供者处获得相对统一的功能性。

1.1.2 ADO 入门

本节内容包含按预定顺序排列的系列主题。只需单击每个主题结尾处的链接即可转到下一个主题。这些主题包括：

- 本地数据访问的解决方案(See 1.1.2.1)
- 基本的 ADO 编程模型(See 1.1.2.2)
- ADO 编程模型详细资料(See 1.1.2.3)

下一页 本地数据访问的解决方案(See 1.1.2.1)。

1.1.2.1 本地数据访问的解决方案

问题

您需要的是简单、一致的应用程序编程接口 (API)，使应用程序能够访问和修改各种各样的数据源。数据源可能象文本文件一样简单，也可能象一堆异构数据库那样复杂，或者是尚未定型的某种数据。此外，API 不应该预先设定访问和操作数据源的方式。

虽然提出了这些要求，典型的数据源依然需要支持开放式数据库连接 (ODBC) 标准的关系型数据库，并可通过用结构化查询语言 (SQL) 编写的命令对它进行操作。

Microsoft 对该问题提供的总体解决方案是 OLE DB，这是一套组件对象模型 (COM) 接口，可提供对存储在不同信息源进行统一访问的能力。但是 OLE DB 应用程序编程接口的设计目的是为了为多种多样的应用程序提供优化功能，它无法满足对简单化的要求。

您需要的 API 应该是一座连接应用程序和 OLE DB 的桥梁，这就是 ActiveX® Data Objects (ADO)。

解决方案

ADO 定义编程模型，即访问和更新数据源所必需的活动顺序。编程模型概括了 ADO 的全部功能。

编程模型意味着对象模型，即响应并执行编程模型的“对象”组。对象拥有“方法”，方法执行对数据进行的操作；对象拥有“属性”，属性指示数据的某些特性或控制某些对象方法的行为。

与对象关联的是“事件”，事件是某些操作已经发生或将要发生的通知。

下一页 基本的 ADO 编程模型(See 1.1.2.2)。

1.1.2.2 基本的 ADO 编程模型

ADO 提供执行以下操作的方式：

1. 连接到数据源。同时，可确定对数据源的所有更改是否已成功或没有发生。
2. 指定访问数据源的命令，同时可带变量参数，或优化执行。
3. 执行命令。
4. 如果这个命令使数据按表中的行的形式返回，则将这些行存储在易于检查、操作或更改的缓存中。

5. 适当情况下，可使用缓存行的更改内容来更新数据源。

6. 提供常规方法检测错误（通常由建立连接或执行命令造成）。

在典型情况下，需要在编程模型中采用所有这些步骤。但是，由于 ADO 有很强的灵活性，所以最后只需执行部分模块就能做一些有用的工作。例如：将数据从文件直接存储到缓存行，然后仅用 ADO 资源对数据进行检查。

下一页 ADO 编程模型详细资料(See 1.1.2.3)。

1.1.2.3 ADO 编程模型详细资料

以下元素是 ADO 编程模型中的关键部分：

- 连接
- 命令
- 参数
- 记录集
- 字段
- 错误
- 属性
- 集合
- 事件

连接

通过“连接”可从应用程序访问数据源，连接是交换数据所必需的环境。通过如 Microsoft® Internet Information Server 作为媒介，应用程序可直接（有时称为双层系统）或间接（有时称为三层系统）访问数据源。

对象模型使用 Connection(See 1.1.4.2.2) 对象使连接概念得以具体化。

“事务”用于界定在连接过程中发生的一系列数据访问操作的开始和结束。ADO 可明确事务中的操作造成的对数据源的更改或者成功发生，或者根本没有发生。

如果取消事务或它的一个操作失败，则最终的结果将仿佛是事务中的操作均未发生，数据源将会保持事务开始以前的状态。

对象模型无法清楚地体现出事务的概念，而是用一组 **Connection** 对象方法来表示。

ADO 访问来自 OLE DB 提供者(See 1.1.5)的数据和服务。**Connection** 对象用于指定专门的提供者和任意参数。例如，可对远程数据服务 (RDS) 进行显式调用，或通过“Microsoft OLE DB Remoting Provider(See 1.1.5.9)”进行隐式调用。（请参阅 RDS 教程(See 1.1.6.1.2)通过“MS Remote Provider”调用 RDS 第二步的范例）

命令

通过已建立的连接发出的“命令”可以某种方式来操作数据源。一般情况下，命令可以在数据源中添加、删除或更新数据，或者在表中以行的格式检索数据。

对象模型用 Command(See 1.1.4.2.1) 对象来体现命令概念。**Command** 对象使 ADO 能够优化对命令的执行。

参数

通常，命令需要的变量部分即“参数”可以在命令发布之前进行更改。例如，可重复发出相同的数据检索命令，但每一次均可更改指定的检索信息。

参数对执行其行为类似函数的命令非常有用，这样就可知道命令是做什么的，但不必知道它如何工作。例如，可发出一项银行过户命令，从一方借出贷给另一方。可将要过户的款额设置为参数。

对象模型用 Parameter(See 1.1.4.2.8) 对象来体现参数概念。

记录集

如果命令是在表中按信息行返回数据的查询（行返回查询），则这些行将会存储在本地。

对象模型将该存储体现为 Recordset(See 1.1.4.2.10) 对象。但是，不存在仅代表单独一个 **Recordset** 行的对象。

记录集是在行中检查和修改数据最主要的方法。**Recordset** 对象用于：

- 指定可以检查的行。
- 移动行。
- 指定移动行的顺序。
- 添加、更改或删除行。
- 通过更改行更新数据源。
- 管理 **Recordset** 的总体状态。

字段

一个记录集行包含一个或多个“字段”。如果将记录集看作二维网格，字段将排列构成“列”。每一字段（列）都分别包含有名称、数据类型和值的属性，正是在该值中包含了来自数据源的真实数据。

对象模型以 Field(See 1.1.4.2.7) 对象体现字段。

要修改数据源中的数据，可在记录集行中修改 **Field** 对象的值，对记录集的更改最终被传送给数据源。作为选项，**Connection** 对象的事务管理方法能够可靠地保证更改要么全部成功，要么全部失败。

错误

错误随时可在应用程序中发生，通常是由于无法建立连接、执行命令或对某些状态（例如，试图使用没有初始化的记录集）的对象进行操作。

对象模型以 Error(See 1.1.4.2.6) 对象体现错误。

任意给定的错误都会产生一个或多个 **Error** 对象，随后产生的错误将会放弃先前的 **Error** 对象组。

属性

每个 ADO 对象都有一组唯一的“属性”来描述或控制对象的行为。

属性有两种类型：内置(See 1.1.4.6)和动态(See 1.1.4.7)。内置属性是 ADO 对象的一部分并且随时可用。动态属性则由特别的数据提供者添加到 ADO 对象的属性集合中，仅在提供者被使用时才能存在。

对象模型以 Property(See 1.1.4.2.9) 对象体现属性。

集合

ADO 提供“集合”，这是一种可方便地包含其他特殊类型对象的对象类型。使用集合方法可按名称（文本字符串）或序号（整型数）对集合中的对象进行检索。

ADO 提供四种类型的集合：

- **Connection** 对象具有 Errors(See 1.1.4.3.1) 集合，包含为响应与数据源有关的单一错误而创建的所有 **Error** 对象。
- **Command** 对象具有 Parameters(See 1.1.4.3.3) 集合，包含应用于 **Command** 对象的所有 **Parameter** 对象。
- **Recordset** 对象具有 Fields(See 1.1.4.3.2) 集合，包含所有定义 **Recordset** 对象列的 **Field** 对象。

- 另外，**Connection**、**Command**、**Recordset** 和 **Field** 对象都具有 Properties(See 1.1.4.3.4) 集合。它包含所有属于各个包含对象的 **Property** 对象。

ADO 对象拥有可在其上使用的诸如“整型”、“字符型”或“布尔型”这样的普通数据类型来设置或检索值的属性。然而，有必要将某些属性看成是数据类型“COLLECTION OBJECT”的返回值。相应的，集合对象具有存储和检索适合该集合的其他对象的方法。

例如，可认为 **Recordset** 对象具有能够返回集合对象的 **Properties** 属性。该集合对象具有存储和检索描述 **Recordset** 性质的 **Property** 对象的方法。

事件

“事件”是对将要发生或已经发生的某些操作的通知。一般情况下，可用事件高效地编写包含几个异步任务的应用程序。

对象模型无法显式体现事件，只能在调用事件处理程序(See 1.1.3.5.1)例程时表现出来。

在操作开始之前调用的事件处理程序便于对操作参数进行检查或修改，然后取消或允许操作完成。

操作完成后调用的事件处理程序在异步操作完成后进行通知。多个操作经过增强可以有选择地异步执行。例如，用于启动异步 **Recordset.Open** 操作的应用程序将在操作结束时得到执行完成事件的通知。

- 有关事件的详细信息，请参阅 ADO 事件模型和异步操作(See 1.1.3.5)。

下一页 使用对象的 ADO 编程模型(See 1.1.2.4)。

1.1.2.4 使用对象的 ADO 编程模型

ADO 的目标是访问、编辑和更新数据源，而编程模型体现了为完成该目标所必需的系列动作的顺序。ADO 提供类和对象以完成以下活动：

- 连接到数据源 (**Connection**)，并可选择开始一个事务。
- 可选择创建对象来表示 SQL 命令 (**Command**)。
- 可选择在 SQL 命令中指定列、表和值作为变量参数 (**Parameter**)。
- 执行命令 (**Command**、**Connection** 或 **Recordset**)。

- 如果命令按行返回，则将行存储在缓存中 **(Recordset)**。
- 可选择创建缓存视图，以便能对数据进行排序、筛选和定位 **(Recordset)**。
- 通过添加、删除或更改行和列编辑数据 **(Recordset)**。
- 在适当情况下，使用缓存中的更改内容来更新数据源 **(Recordset)**。
- 如果使用了事务，则可以接受或拒绝在完成事务期间所作的更改。结束事务 **(Connection)**。

下一页 ADO 对象模型总结(See 1.1.2.5)。

1.1.2.5 ADO 对象模型总结

ADO 对象总结

对象	说明
Connection(See 1.1.4.2.2)	启用数据的交换。
Command(See 1.1.4.2.1)	体现 SQL 语句。
Parameter(See 1.1.4.2.8)	体现 SQL 语句参数。
Recordset(See 1.1.4.2.10)	启用数据的定位和操作。
Field(See 1.1.4.2.7)	体现 **Recordset** 对象列。
Error(See 1.1.4.2.6)	体现连接错误。
Property(See 1.1.4.2.9)	体现 ADO 对象特性。

ADO 集合总结

集合	说明
Errors(See 1.1.4.3.1)	为响应单个连接错误而创建的所有 **Error** 对象。
Parameters(See 1.1.4.3.3)	与 **Command** 对象关联的所有 **Parameter** 对象。
Fields(See 1.1.4.3.2)	与 **Recordset** 对象关联的所有 **Field** 对象。
Properties(See 1.1.4.3.4)	与 **Connection**、**Command**、**Recordset** 或 **Field** 对象关联的所有 **Property** 对象。

ADO 事件处理程序总结

ConnectionEvents	说明
BeginTransComplete 、 CommitTransComplete 、 RollbackTransComplete(See 1.1.4.5.1)	**事务管理** — 通知连接中的当前事务已开始、提交或回卷。
WillConnect(See 1.1.4.5.13)、ConnectComplete、Disconnect(See 1.1.4.5.2)	**连接管理** — 通知当前连接将要开始、已经开始或结束。
WillExecute(See 1.1.4.5.14)、ExecuteComplete(See 1.1.4.5.4)	**命令执行管理** — 通知连接中的当前命令将要开始或已经结束。
InfoMessage(See 1.1.4.5.7)	**信息** — 通知获得与当前操作相关的附加信息。

RecordsetEvents	说明
FetchProgress(See 1.1.4.5.6) 、 FetchComplete(See 1.1.4.5.5)	**检索状态** — 通知数据检索操作的进程或检索操作已经完成。
WillChangeField 、 FieldChangeComplete(See 1.1.4.5.10)	**字段更改管理** — 通知当前字段的值将要更改或已经更改。
WillMove 、 MoveComplete(See 1.1.4.5.15) 、 EndOfRecordset(See 1.1.4.5.3)	**定位管理** — 通知在 **Recordset** 中当前行的位置将要更改、已经更改或已到达 **Recordset** 的结尾。
WillChangeRecord 、 RecordChangeComplete(See 1.1.4.5.11)	**行更改管理** — 通知有关 **Recordset** 当前行中某些内容将要更改，或已经更改。
WillChangeRecordset 、 RecordsetChangeComplete(See 1.1.4.5.12)	**记录集更改管理** — 通知当前 **Recordset** 中某些内容将要更改，或已经更改。

下一页 远程数据访问的解决方案(See 1.1.2.6)。

1.1.2.6 远程数据访问的解决方案

问题

ADO 可让应用程序直接访问并修改数据源（有时称为双层系统）。例如，如果要连接到包含所需数据的数据源，则该连接即是在双层系统中的直接连接。

然而，也需要通过象 Microsoft Internet Information Server (IIS) 这样的媒介间接地访问数据源。这种方法称为三层系统。IIS 采用客户端/服务器系统，该系统可帮助本地（或客户端）的应用程序通过 Internet 或 Intranet 高效地调用远程（或服务器）程序。服务器程序访问数据源，并可有选择地处理已获得的数据。

例如，您的 Intranet Web 页包含有用 Microsoft® Visual Basic® Scripting Edition (VBScript) 编写的应用程序，该程序连接 IIS。IIS 相应地连接实际数据源，检索数据，以某种方式处理数据，然后将已处理的信息返回给应用程序。

在这个例子中，应用程序从未直接连接数据源，该工作由 IIS 完成。而 IIS 利用 ADO 来访问数据。

> **注意** 客户端/服务器应用程序不一定必须基于 Internet 或 Intranet （即基于 Web），它可以仅由局域网上的编译程序组成。但是，典型的范例是基于 Web 的应用程序。

因为一些可视化控件诸如网格、复选框或列表可以使用返回信息，所以返回信息必须易于被可视化控件使用。

用户需要的是简单有效的应用程序编程接口，该程序应支持三层系统，同时能够象在双层系统上检索信息一样容易返回信息。“远程数据服务” (RDS) 即是这样的接口。

解决方案

RDS 定义的编程模型（访问和更新数据源必需的系列活动）通过如 Internet Information Server 这样的中间媒介来访问数据。编程模型总结了 RDS 的全部功能。

编程模型通过对象模型，即“对象”集来表达并实现编程模型。对象拥有能操作数据的“方法”，以及能够表示数据属性或控制某些对象方法行为的“属性”。

与对象关联的是“事件”，事件用于通知某些操作已经发生，或将要发生。

下一页 基本的 RDS 编程模型(See 1.1.2.7)。

1.1.2.7 基本的 RDS 编程模型

RDS 为在如下环境中存在的应用程序确定地址：客户端应用程序指定将在服务器上执行的程序，并指定用来返回相应信息的参数。服务器上被调用的程序访问指定的数据源，检索信息，对数据进行相应处理，然后将结果信息按易于使用的格式返回给客户端应用程序。 RDS 为您提供了执行以下系列操作的方式：

1. 指定在服务器上被调用的程序，并得到从客户端引用该程序的途径。（该引用有时称为“代理”，它代表远程服务器程序。客户端应用程序象调用本地程序一样“调用”代理，但实际上调用的是远程服务器程序。）
2. 调用服务器程序。将参数传送到标识数据源及所要发布命令的服务器程序。（服务器程序实际上使用 ADO 访问数据源。ADO 与所给参数中的一个建立连接，然后发布在其他参数中指定的命令）。
3. 服务器程序从数据源获得了 **Recordset** 对象。可以选择在服务器上处理 **Recordset** 对象。
4. 服务器程序将最终的 **Recordset** 对象返回客户端应用程序。
5. 在客户端，**Recordset** 对象被转换成为便于可视化控件使用的格式。
6. 任何对 **Recordset** 对象所作的修改都将返回给服务器程序，服务器程序用这些修改来更新数据源。

该编程模型包含了某些便利功能。如果不需要复杂的服务器程序访问数据源，并提供所需的连接和命令参数，RDS 将自动使用简单的默认服务器程序来检索指定数据。

如果仍需要进行复杂处理，可指定自定义的服务器程序。例如，由于自定义服务器程序有足够的 ADO 处置能力，所以能连接几个不同的数据源，并把这些数据以某种复杂的方式结合起来，然后将简化的、经过处理的结果返回给客户端应用程序。

最后，如果在这两者之间还需要别的方式，ADO 支持自定义默认服务器程序的行为。

下一页 RDS 编程模型详细资料(See 1.1.2.8)。

1.1.2.8 RDS 编程模型详细资料

下列元素是 RDS 编程模型中的关键部分：

- RDS.DataSpace
- RDSServer.DataFactory
- RDS.DataControl
- 事件

RDS.DataSpace

客户端应用程序必需指定服务器和要调用的服务器程序。相应的，应用程序接收对服务器程序的引用，并且将此引用当作实际的服务器程序。

RDS 对象模型通过 RDS.DataSpace(See 1.1.4.2.5) 对象来体现该功能。

使用程序标识符（即 ProgID）来指定服务器程序。服务器使用 ProgID 和服务器计算机的注册表来定位需初始化的程序的信息。根据服务器程序是在由 Internet 或 Intranet 连接的远程服务器上，还是在连接局域网的服务器上，或者根本不在服务器而在本地动态连接库 (DLL) 上，RDS 可在内部进行划分。该划分决定在客户与服务器之间交换数据的方式，并对于返回给客户端应用程序的“引用”类型中有着实际的区别。然而，从用户的角度来看，该划分没有特殊的意义。所有这些只是令您接收到可用的程序引用。

RDSServer.DataFactory

RDS 提供的默认服务器程序可对数据源执行 SQL 查询并返回 **Recordset** 对象，或获得 **Recordset** 对象并更新数据源。

RDS 对象模型用 RDSServer.DataFactory(See 1.1.4.2.4) 对象来体现该功能。

此外，该对象具有创建空的 **Recordset** 对象的方法，可用编程的方式对该空 **Recordset** 进行填写。它还有另一种方法可将 **Recordset** 转换为文本串来建立 Web 页。

使用 ADO，可以利用 **DataFactory** 处理程序和包含连接、命令和安全参数的自定义文件覆盖一些标准连接和 **RDSServer.DataFactory** 的命令行为。

服务器程序有时可称作“业务对象”。您可以编写自己的自定义业务对象，它可以执行复杂数据访问，有效性检查等。甚至在编写自己的自定义业务对象时，可创建 RDSServer.DataFactory 对象的实例并且使用它的一些方法完成自己的任务。

RDS.DataControl

RDS 提供了可将 **RDS.DataSpace** 和 **RDSServer.DataFactory** 的功能结合在一起的方法。RDS 也能让可视化控件容易地使用查询数据源所返回的 **Recordset** 对象。大多数情况下，RDS 总是尽可能多的自动访问服务器上的信息，并且将信息显示在可视化控件中。

RDS 对象模型用 RDS.DataControl(See 1.1.4.2.3) 对象来体现该功能。

RDS.DataControl 有两个方面。一个方面与数据源有关。如果设置 **RDS.DataControl** 的命令和连接属性，它将会自动使用 **RDS.DataSpace** 创建对默认 **RDSServer.DataFactory** 对象的引用。然后 **RDSServer.DataFactory** 将使用连接属性的值连接到数据源，并使用命令属性的值从数据源获得 **Recordset**，最后将 **Recordset** 对象返回到 **RDS.DataControl**。

第二个方面涉及在可视化控件中显示被返回的 **Recordset** 信息。可以使可视控件与 **RDS.DataControl** 相关联（在称为绑定的过程中），并访问关联的 **Recordset** 对象中的信息，查询结果显示在 Internet Explorer 的 Web 页上。每个 **RDS.DataControl** 对象将一个表示单个查询结果的 **Recordset** 对象绑定到一个或多个可视控件（例如文本框、组合框和网格控件等）上。在每页上可以有多个 **RDS.DataControl** 对象。每个 **RDS.DataControl** 对象都可连接不同的数据源，并且包含各自的查询结果。

RDS.DataControl 对象也有其自己的方法用于定位、排序和筛选相关联的 **Recordset** 对象的行。这些方法虽然相似，但与 ADO **Recordset** 对象所用的方法不同。

事件

RDS 支持两个独立于 ADO 事件模型的自身事件。无论 **RDS.DataControl** ReadyState(See 1.1.4.6.54) 属性何时更改均调用 onReadyStateChange(See 1.1.4.5.9) 事件，以此对异步操作的完成、结束或出现错误等发出通知。无论何时发生错误，即使发生在异步操作执行的过程中，均调用 onError(See 1.1.4.5.8) 事件。

> **注意** Microsoft Internet Explorer 环境提供给 RDS 两个附加事件：**onDataSetChanged**（**Recordset** 在起作用但还在检索行）和 **onDataSetComplete**（ **Recordset** 已结束检索行）。

下一页 使用对象的 RDS 编程模型(See 1.1.2.9)。

1.1.2.9 使用对象的 RDS 编程模型

RDS 的目的是通过 Internet Information Server 这样的媒介来访问和更新数据源。编程模型则指定为完成这个目的所必需的活动序列。对象模型指定其方法和属性影响编程模型的对象。

RDS 提供执行以下动作序列的途径：

1. 指定在服务器上被调用的程序，并获得通过客户端调用该程序的方式（代理）。 **(RDS.DataSpace)**
2. 调用服务器程序。将参数传递给标识数据源及所要发布的命令的服务器程序。**(**代理或 **RDS.DataControl)**
3. 服务器程序从数据源获得 **Recordset** 对象（一般通过使用 ADO）。可选择在服务器上处理 **Recordset** 对象。**(RDSServer.DataFactory)**
4. 服务器程序将最终的 **Recordset** 对象返回客户端应用程序。（代理）
5. 在客户端，**Recordset** 对象被转换成可视控件能方便使用的格式。（可视控件和 **RDS.DataControl**）
6. 对 **Recordset** 对象所作的更改被返回服务器并用于更新数据源。（**RDS.DataControl** 或 **RDSServer.DataFactory**）

下一页 RDS 对象模型总结(See 1.1.2.10)。

1.1.2.10 RDS 对象模型总结

对象	**说明**
RDS.DataSpace(See 1.1.4.2.5)	该对象只包含获得服务器代理的方法。代理可以是默认或自定义的服务器程序（业务对象）。在 Internet、Intranet、局域网或本地动态连接库上均可调用服务器程序。
RDSServer.DataFactory(See 1.1.4.2.4)	该对象表示默认的服务器程序。它执行默认的 RDS 数据检索和更新行为。

RDS.DataControl(See 1.1.4.2.3)	• 该对象可自动调用 **RDS.DataSpace** 和 **RDSServer.DataFactory** 对象。 • 使用该对象调用默认的 RDS 数据检索或更新行为。 • 该对象也为可视控件提供访问被返回的 **Recordset** 对象的途径。

1.1.3 ADO 特性

下面这些主题描述了在 ADO 和 RDS 中各种可用的特性。

- 创建 Recordset 的捷径(See 1.1.3.1)
- Recordset 持久性(See 1.1.3.2)
- 索引支持和查找、排序以及过滤(See 1.1.3.3)
- ADO for Windows Foundation Classes(See 1.1.3.4)
- ADO 事件模型和异步操作(See 1.1.3.5)
- VC++ Extensions for ADO(See 1.1.3.6)
- 数据构形(See 1.1.3.7)
- DataFactory 自定义(See 1.1.3.8)

1.1.3.1 创建 Recordset 的捷径

ADO 提供便捷方法创建 **Recordset**：将新的 **Field** 对象添加到 **Recordset** 的 **Field** 集合。随后，可以打开 **Recordset** 并插入来自任意源（不必是数据库）的数据。还可以通过程序产生数据。

新 **Recordset** 可以使用对任意 **Recordset** 均为可用的所有数据操作方法。使用 **Recordset** 将信息提供给可视控件，甚至更新实际数据源。

1.1.3.2 Recordset 持久性

使用记录集持久性，可以将 **Recordset** 数据和元数据保存为文件。随后，使用持久文件来重新建立 **Recordset** 对象。持久文件可以保存在本地驱动器、网络服务器上或者作为 URL 保存在 Web 站点上。

另外，**GetString** 方法将 **Recordset** 对象转换成表单，在表单中列和行使用指定的字符分界。

详细资料

Microsoft OLE DB Persistence Provider 支持使用 **Recordset** 对象 **Save** 方法将 **Recordset** 对象保存在文件中。随后，使用

Recordset 对象的 **Open**、或 **Connection** 对象的 **Execute** 方法可恢复持久文件。

Recordset 对象被转换为能被保存在文件中的表单。**Recordset** 对象可以按所拥有的高级数据图表 **(ADTG)** 格式保存，或者按打开的可扩展标记语言 **(XML)** 格式保存。

挂起更改保存在持久文件中。因此，可以发布查询返回 **Recordset** 对象、编辑记录集、保存该记录集和挂起变化、以后恢复该记录集、然后使用保存的挂起变化更新数据源。

用法

保存记录集:

```
Dim rs as New ADODB.Recordset
rs.Save "c:\yourFile.adtg", adPersistADTG
```

使用 Recordset.Open 打开持久文件:

```
dim rs as New ADODB.Recordset
rs.Open "c:\yourFile.adtg", "Provider=MSPersist",,,adCmdFile
```

可选地，如果 Recordset 没有活动的连接，则都可以接受所有的默认值和简单的代码：

```
dim rs as New ADODB.Recordset
rs.Open "c:\yourFile.adtg"
```

使用 Connection.Execute 打开持久文件:

```
dim conn as New ADODB.Connection
dim rs as New ADODB.Recordset
conn.Open "Provider=MSPersist"
set rs = conn.execute("c:\yourFile.adtg")
```

使用 RDS.DataControl 打开持久文件:

在这种情况下，没有设置 Server 属性。

```
Dim dc as New RDS.DataControl
dc.Connection = "Provider=MSPersist"
dc.SQL = "c:\yourFile.adtg"
dc.Refresh
```

1.1.3.3 索引支持和查找、排序以及过滤

对字段进行索引可以极大增强 **Recordset** 对象的 **Find** 方法、以及 **Sort** 和 **Filter** 属性的性能。通过设置 **Field** 对象的动态 **Optimize** 属性，可以创建该对象的内部索引。当将 **CursorLocation** 属性设置为 **adUseClient** 时，该动态属性被添加给 **Field** 对象的 **Properties** 集合。请记住该索引对于 ADO 是内部的，即无法访问该索引或者将该索引用于其他的目的。

Sort 属性决定穿越 **Recordset** 行的顺序，**Filter** 属性决定当穿越行时哪些行是可访问的，**Find** 方法在 **Recordset** 的索引列（字段）中快速定位值。

1.1.3.4 ADO for Windows Foundation Classes

通过 Microsoft® Visual J++™ (Java 编程语言)，可使用 ADO 应用程序编程接口访问数据。

ADO for Windows Foundation Classes (ADO/WFC) 支持所有标准 ADO 方法、属性、对象和事件。但是，需要 VARIANT 作为参数并使用类似 Microsoft® Visual Basic® 的语言可显示优良性能的操作，在如 Visual J++ 的语言中却仅显示较低的性能。因为该原因，ADO/WFC 同时提供了 **Field** 对象的访问者函数，用本身 Java 数据类型代替了 VARIANT 数据类型。

1.1.3.5 ADO 事件模型和异步操作

ADO 事件模型支持某些同步和异步的 ADO 操作，在这些操作开始之前或完成之后会产生“事件”(即通知)。事件实际上是对在应用程序中定义的事件处理程序例程的调用。

在操作开始之前被调用的事件处理程序允许您检查或修改操作参数，然后取消操作或让它完成。

由于 ADO 支持异步操作，在操作完成之后被调用的事件处理程序尤其重要。例如，启动异步 **Recordset.Open** 操作的应用程序在操作结束时会接收到执行完成事件的通知。

有关 ADO 中事件的详细信息，请参阅如下主题：

- ADO 事件处理程序总结(See 1.1.3.5.1)
- 事件类型(See 1.1.3.5.2)
- 事件参数(See 1.1.3.5.3)
- 事件处理程序如何共同工作(See 1.1.3.5.4)
- ADO/WFC 中的 ADO 事件(See 1.1.3.5.5)
- 不同语言的 ADO 事件实例(See 1.1.3.5.6)

1.1.3.5.1 ADO 事件处理程序总结

事件分成两类。**ConnectionEvent** 类从属于 **Connection** 对象的操作，而 **RecordsetEvent** 类则从属于 **Recordset** 对象的操作。

- **ConnectionEvents** — 在连接的事务开始、提交或回卷时，或在 **Command** 被执行、连接开始或结束时，将引发该类事件。
- **RecordsetEvents** — 在 **Recordset** 对象行中定位、更改 **Recordset** 行中的字段、更改 **Recordset** 的行、或是在 **Recordset** 作任何修改时，将引发该类事件。

ADO 事件处理程序概述

ConnectionEvent	说明
BeginTransComplete 、	**事务管理** — 关于连接上的当前事务已经开始、已经提交、或者已经回卷的通知。

CommitTransComplete、RollbackTransComplete(See 1.1.4.5.1)	
WillConnect(See 1.1.4.5.13)、ConnectComplete, Disconnect(See 1.1.4.5.2)	**连接管理** — 关于当前连接即将开始、已经开始、或者已经结束的通知。
WillExecute(See 1.1.4.5.14)、ExecuteComplete(See 1.1.4.5.4)	**命令执行管理** — 关于连接的当前命令的执行即将开始、或者已经结束的通知。
InfoMessage(See 1.1.4.5.7)	**信息管理** — 关于当前操作有附加信息的通知。

RecordsetEvent	说明
FetchProgress(See 1.1.4.5.6), FetchComplete(See 1.1.4.5.5)	**检索状态** — 关于数据检索操作进度、或者检索操作已经结束的通知。
WillChangeField, FieldChangeComplete(See 1.1.4.5.10)	**字段更改管理** — 关于当前字段值即将更改、或者已经更改的通知。
WillMove, MoveComplete(See 1.1.4.5.15), EndOfRecordset(See 1.1.4.5.3)	**定位管理** — 关于当前行在 **Recordset** 中的位置即将更改、已经更改、或者已经到达 **Recordset** 结尾的通知。
WillChangeRecord, RecordChangeComplete(See 1.1.4.5.11)	**行更改管理** — 关于 **Recordset** 当前行的某些地方即将更改、或者已经更改的通知。
WillChangeRecordset, RecordsetChangeComplete(See 1.1.4.5.12)	**Recordset 更改管理** — 关于当前 **Recordset** 的某些地方即将更改、或者已经更改的通知。

1.1.3.5.2 事件类型

Will 事件

操作开始之前调用的事件处理程序使您有机会检查或修改操作参数，然后取消操作或允许完成该操作。这些事件处理程序例程通常具有形如 **WillEvent**（**Will** 事件）的名称。

Complete 事件

操作完成之后调用的事件处理程序将向应用程序发出操作已经结束的通知。当挂起的操作被 **Will** 事件处理程序取消时，该事件处理程序也会收到通知。这些事件处理程序例程通常具有如象 **EventComplete**（**Complete** 事件）的名称。
Will 和 **Complete** 事件一般成对使用。

其他事件

其他事件处理程序（事件名不使用 **WillEvent** 或 **EventComplete** 格式）仅在操作完成后被调用。

1.1.3.5.3 事件参数

每个事件处理程序都有一个控制它的状态参数。大部分 **Complete** 事件都有错误参数用于报告引起事件发生的操作是否成功。另外还有一个对象参数，用于标识操作所针对的 ADO 对象。

参数同时被传递到 Will 事件以便用于挂起操作。这样您就有机会检查参数并确定操作是否应该完成。

某些事件处理程序有原因参数，该参数可以提供有关事件发生原因的详细信息。

状态参数

当事件处理程序例程被调用时，状态参数将被设置为下列信息值之一。

值	说明
adStatusOK	引发事件的操作已成功发生。
adStatusErrorsOccurred	引发事件的操作未成功发生，或 **Will** 事件取消了操作。有关细节，请单击错误参数。
adStatusCantDeny	**Will** 事件无法请求取消即将开始的操作。

在事件处理程序例程返回之前，应当保持状态参数不变，或将其设置为下列请求值之一。

值	说明
adStatusUnwantedEvent	请求该事件处理程序不接收以后的通知。
adStatusCancel	请求取消即将开始的操作。

由事件类型所决定，当事件处理程序被调用时，状态参数可以取下列值之一。

事件类型	值
Will	**adStatusOK**, **adStatusCantDeny**
Complete	**adStatusOK**, **adStatusErrorsOccurred**

取决于事件类型，在事件处理程序返回时，状态参数可以取下列值之一。

事件类型	值
Will	**adStatusOK**, **adStatusCancel**, **adStatusUnwantedEvent**
Complete	**adStatusOK**, **adStatusUnwantedEvent**

错误参数

错误参数是对 ADO Error(See 1.1.4.2.6) 对象的引用。状态参数为 **adStatusErrorsOccurred** 时，该对象包含操作失败的详细信息。

对象参数

对象参数是对所操作的 ADO 对象的引用。例如，可以同时打开几个 **Connection** 对象，但一次只能打开一个 **Disconnect** 事件处理程序。如果所有连接关闭，将对象参数设置为关闭的 **Connection** 对象，即可调用 **Disconnect** 事件处理程序。

原因参数

原因参数（**adReason**）提供有关事件发生原因的附加信息。带有 **adReason** 参数的方法可以被多次调用 — 甚至被相同的操作调用，但每次调用都有不同原因。

例如，**WillChangeRecord** 事件处理程序被某些操作调用，这些操作将进行或撤消记录的插入、删除或修改。**adReason** 参数被作为只处理特殊事件的筛选器来使用。

必须在 **adStatus** 参数中返回 **adStatusUnwantedEvent**，要求不带 **adReason** 参数的事件处理程序停止接收事件通知。但是，带有 **adReason** 参数的事件处理程序可能会接收几个通知，每个通知有不同原因。所以，对每个不同原因产生的通知，必须返回 **adStatusUnwantedEvent**。

例如，假设 **WillChangeRecord** 事件处理程序是用 Microsoft® Visual Basic® 编写的。如果您以后不想接收任何进一步的通知，只需编写如下代码：

```
Set adStatus = adStatusUnwantedEvent
```

但是，如果要处理其行将被删除的事件，但取消所有其他原因所带来的通知时，请编写如下程序：

```
if (adReason = adRsnDelete)
'处理此原因引起的事件。
...
else
'停止接收任何其他原因引起的事件。
Set adStatus = adStatusUnwantedEvent
...
```

1.1.3.5.4 事件处理程序如何共同工作

无论是否实际使用事件，都必须执行 **ConnectionEvent** 和 **RecordsetEvent** 类中的所有事件处理程序。必须执行的工作量取决于所用的编程语言。某些语言，如 Microsoft Visual Basic 可以为您完成所有工作。其他语言如 Microsoft® Visual C++® 则需要您做所有工作。而 Microsoft® Visual J++™ with ADO/WFC 则介于两者之间，该语言为您处理大部分工作。详细信息，请参阅不同语言的 ADO 事件实例(See 1.1.3.5.6)。

虽然由您自己实现事件处理程序的工作量较大，但可以完成用 Visual Basic 这类语言无法进行的工作。例如，在 Microsoft Visual C++ 中，一个 **RecordsetEvent** 处理程序可以处理在操作多个 **Recordset** 对象时所产生的通知。

Will 和 **Complete** 事件处理程序可以成对或分开使用。

成对的事件处理程序

- 如下设定说明 **Will** 事件成功时会出现什么情况。

  Recordset 对象拥有成对事件 **WillChangeField** 和 **FieldChangeComplete**。在应用程序中开始更改字段值时，将调用 **WillChangeField** 事件处理程序；可以返回一个用于更改字段的标志。操作完成时，**FieldChangeComplete** 事件将通知应用程序操作已经结束，而事件处理程序状态参数则报告操作成功。

- 如下设定说明 **Will** 事件将操作取消时会出现什么情况。

  在同一应用程序中，更改另一个字段。将调用 **WillChangeField** 事件处理程序。可能因为某种原因您决定不接受更改字段，所以您在状态参数中返回 **adStatusCancel**。结果，操作未能完成。

  FieldChangeComplete 事件通知您操作已经结束。事件处理程序状态参数被设为 **adStatusErrorsOccurred**；错误参数引用 **Error** 对象，而 **Error** 对象的 **Number** 属性被设为 ADO 值或提供者，从而指示操作已被取消。

- 多个 **Will** 和 **Complete** 事件处理程序可被同一操作调用。如下设定说明当多个 **will** 事件成功时会出现了什么情况。

  Recordset 对象拥有成对事件 WillChangeField、FieldChangeComplete、WillChangeRecord 和 RecordChangeComplete。开始更改字段值时将调用 WillChangeField 事件处理程序；可以返回一个用于更改字段的标志。

  下一步，调用 WillChangeRecord 事件处理程序，再次指示操作应该结束。

  注意 所有从属于 ADO 对象的特定实例的 will 事件处理程序通常都将被调用。但是，调用过程没有特定次序。

  操作结束时，将调用 FieldChangeComplete 和 RecordChangeComplete 事件处理程序。

- 多个 **Will** 和 **Complete** 事件处理程序可被同一操作调用，但是可以取消挂起的操作。如下设定说明在多个 **will** 事件的最后一个取消操作时，将出现什么情况。

  再一次，Recordset 拥有成对的 WillChangeField、FieldChangeComplete、WillChangeRecord 和 RecordChangeComplete 事件。开始更改字段值时将调用 WillChangeField 事件处理程序；可以返回一个用于更改字段的标志。

  下一步，将调用 WillChangeRecord 事件处理程序。也许您认为字段更改本身没有问题，但是它将在记录中创建一个整体错误。您返回 adStatusCancel，指示不接受更改字段。WillChangeField 事件处理程序已经允许进行操作。

  操作被 WillChangeRecord 事件处理程序取消，因此未能完成。调用 FieldChangeComplete 事件处理程序，将状态参数设置为 adStatusErrorsOccurred；并对错误参数进行适当的设置。

  下一步，RecordChangeComplete 事件处理程序被设置为 adStatusErrorsOccurred 的状态参数调用。匹配的 Complete 事件被 will 事件调用。

- 多个 **Will** 和 **Complete** 事件处理程序可被同一个操作调用，但是可以取消挂起操作。如下设定说明当操作被事件处理程序、而不是多个 will 事件的最后一个事件取消时，会出现什么情况。

  再一次，Recordset 拥有成对的 WillChangeField、FieldChangeComplete、WillChangeRecord 和 RecordChangeComplete 事件。在开始更改字段值时将调用 WillChangeField 事件处理程序，返回 adStatusCancel，从而指示更改字段是不可接受的。操作没有完成；FieldChangeComplete 事件处理程序通知您操作已经结束，状态和错误参数设置正确。

  然而，由于第一个 will 事件取消操作，WillChangeRecord（还有 RecordChangeComplete）事件处理程序未被调用。通常，如果 will 事件取消操作，其他 will 事件处理程序将不会被调用。

不成对的事件处理程序

通过返回 **status** 参数的 **adStatusUnwantedEvent** 可以关闭任何事件的事件通知。例如，第一个 **Complete** 事件处理程序被调用时，将返回 **adStatusUnwantedEvent**，并且您随后只收到 **will** 事件。

检查操作中将要使用的参数时，单个 **will** 事件处理程序会很有帮助。您可以修改这些操作参数或者取消操作。

另外一种情况是，打开 **Complete** 事件通知，当第一个 **will** 事件处理程序被调用时，返回 **adStatusUnwantedEvent**。您随后将只

收到 **Complete** 事件。

单个 **Complete** 事件处理程序是管理异步操作的有效工具。每个异步操作都具有相应的 **Complete** 事件。

例如，充填很大的 **Recordset** 对象要花很长时间。如果应用程序编写恰当，则可以启动 Recordset.Open(...,adAsyncExecute) 操作并继续其他处理程序。最终您一定会在 **ExecuteComplete** 事件充填 **Recordset** 时收到通知。

单个事件处理程序和多个对象

Microsoft Visual C++ 这类程序语言的灵活性使得一个事件处理程序可以处理多个对象的事件。例如，应用一个 **Disconnect** 事件处理程序可以处理多个 **Connection** 对象的事件。如果一个连接结束，**Disconnect** 事件处理程序即会被调用。因为事件处理程序参数 **object** 被设置到相应的 **Connection** 对象，您可以判断出引起事件的连接。

注意 因为 Visual Basic 只能使一个对象与一个事件相关联，此技术不能应用于该语言。

多个事件处理程序和单个操作

要让一个 ADO 对象及其操作与多组事件相关联是可能的，但是不太有用。例如，每次执行专门的字段有效性编辑时，可以创建多个 **WillChangeField** 事件。如果字段将要更改，某个 **will** 事件可能会使字段值的某个部分有效，而另一个 **will** 事件则会使另一部分有效。

此技术之所以不太有用，是因为通过单个事件处理程序易于执行或调用所有的编辑程序。但为了完整起见，才在此提到它。

1.1.3.5.5 ADO/WFC 中的 ADO 事件

ADO for Windows Foundation Classes (ADO/WFC) 建立在 ADO 事件模型之上，提供简化的应用程序接口。通常，ADO/WFC 截取 ADO 事件，将事件参数合并到单个事件类之中，然后调用事件处理程序。

使用 ADO/WFC 中的 ADO 事件

1. 定义自己的事件处理程序方法以处理事件。例如，当您要处理 **ConnectionEvent** 类中的 **ConnectComplete** 事件时，可使用下列程序：

```
public void onConnectComplete(Object sender,ConnectionEvent e)
{
System.out.println("onConnectComplete:" + e);
}
```

6. 定义对象处理程序以表示您的事件处理程序方法。对 **ConnectionEvent** 类型的事件，处理程序对象数据类型应该是 **ConnectEventHandler**，而对 **RecordsetEvent** 类型事件则应为 **RecordsetEventHandler** 数据类型。例如，请为 **ConnectComplete** 事件处理程序编写如下程序：

```
ConnectionEventHandler handler =
new ConnectionEventHandler(this, "onConnectComplete");
```

ConnectionEventHandler 构造函数的第一个参数是对某个类的引用，该类含有第二个参数命名的方法。

Microsoft Visual J++ 编译器也支持相同的语法：

```
ConnectionEventHandler handler =
new ConnectionEventHandler(this.onConnectComplete);
```

单个参数是对所需类（即 this）及其方法的引用（即 onConnectComplete）。

9. 在指定用来处理特殊类型事件的处理程序列表中添加事件处理程序。使用带有 **addOn**EventName(handler) 这类名称的方法。

10. ADO/WFC 在内部执行所有的 ADO 事件处理程序。所以，**Connection** 或 **Recordset** 操作引起的事件都由 ADO/WFC 事件处理程序截取。

    ADO/WFC 事件处理程序传送 ADO/WFC **ConnectionEvent** 类实例的 ADO **ConnectionEvent** 参数，或 ADO/WFC **RecordsetEvent** 类实例的 ADO **RecordsetEvent** 参数。这些 ADO/WFC 类将合并 ADO 事件参数；也就是说，对所有 ADO **ConnectionEvent** 或 **RecordsetEvent** 方法的每个特有的参数，每个 ADO/WFC 都含有一个相应的数据成员。

11. 然后 ADO/WFC 通过 ADO/WFC 的事件对象调用事件处理程序。例如，**onConnectComplete** 事件处理程序有下列签名：

```
12. public void onConnectComplete(Object sender,ConnectionEvent e)
```

    第一个参数是传送事件（**Connection** 或 **Recordset**）的对象类型，而第二个参数则是 ADO/WFC 事件对象（**ConnectionEvent** 或 **RecordsetEvent**）。

    事件处理程序的签名比 ADO 事件简单。但是，您还必须要了解 ADO 事件模型才能知道适用于事件的参数以及响应的方法。

13. 从事件处理程序返回到 ADO 事件的 ADO/WFC 处理程序。ADO/WFC 将复制相关的 ADO/WFC 事件数据成员到 ADO 事件参数，然后返回 ADO 事件处理程序。

14. 结束处理的时候，从 ADO/WFC 事件处理程序列表上删除处理程序。使用带有 **removeOn**EventName(handler) 这类名称的方法。

1.1.3.5.6 不同语言的 ADO 事件实例

不同的编程语言以不同的方式创建 ADO 事件实例。如下所有范例均创建 **ConnectComplete** 事件处理程序。

Visual Basic

```
Dim WithEvent connEvent as Connection
Dim conn as New Connection
set connEvent = conn      '打开事件支持。
conn.Open(...)
...
set connEvent = Nothing   '关闭事件支持。
...
Private Sub connEvent_ConnectComplete(ByVal err as ADODB.Error, & _
adStatus as ADODB.EventStatus, ByVal pConnectionas ADODB.Connection)
'仅当 adStatus 等于 adStatusErrorsOccurred 时检查错误对象。
...
End Sub
```

Visual C++

这是关于在 VC++ 中如何实例化事件的示意性说明。

创建在 adoint.h 文件中由 **ConnectionEventsVt** 和 **RecordsetEventsVt** 接口所派生的类。

```
class CConnEvent : public ConnectionEventsVt
{
  public:
  STDMETHODIMP InfoMessage(
       ADOError *pError,
       EventStatusEnum *adStatus,
       _ADOConnection *pConnection);
...
}

class CRstEvent : public RecordsetEventsVt
{
  public:
     STDMETHODIMP WillChangeField(
          LONG cFields,
          VARIANT Fields,
          EventStatusEnum *adStatus,
          _ADORecordset *pRecordset);
...
}
```

在两个类中执行每个事件处理程序方法。每个方法只返回一个 HRESULT of S_OK 就够了。但是，一旦事件处理程序有效，这些事件处理程序就会在默认状态下被连续调用。为改变这种情况，您可通过将 **adStatus** 设置为 **adStatusUnwantedEvent**，以便在第一次调用之后不再请求其他通知。

```
STDMETHODIMP CConnEvent::ConnectComplete(
          ADOError *pError,
          EventStatusEnum *adStatus,
           _ADOConnection *pConnection)
     {
     *adStatus = adStatusUnwantedEvent;
     return S_OK;
     }
```

事件类继承了 **Iunknown**，所以也必须执行 **QueryInterface**、**AddRef** 和 **Release** 方法。如要执行类构造函数和析构函数，请选择最易于使用并能简化这部分任务的 VC++ 工具。

通过在 **Recordset** 和 **Connection** 对象上向 **IConnectionPointContainer** 和 **IConnectionPoint** 接口发出 **QueryInterface**，通知事件处理程序现在已可以使用，然后向各类发出 **IConnectionPoint::Advise**。

例如，假设您正在使用布尔型函数，该函数在成功地向 Recordset 对象发出可以使用事件处理程序的通知时返回“真”值。

```
HRESULT   hr;
DWORD     dwEvtClass;
IConnectionPointContainer   *pCPC = NULL;
IConnectionPoint          *pCP = NULL;
CRstEvent               *pRStEvent = NULL;
...
_RecordsetPtr   pRs();
pRs.CreateInstance(__uuidof(Recordset));
pRStEvent = New CRstEvent();
```

```
if (pRStEvent == NULL) return FALSE;
...
hr = pRs->QueryInterface(IID_IConnectionPointContainer, &pCPC);
if (FAILED(hr)) return FALSE;
hr = pCPC->FindConnectionPoint(IID_ADORecordsetEvents, &pCP);
pCPC->Release();  // Always Release now, even before checking.
if (FAILED(hr)) return FALSE;
hr = pCP->Advise(pRstEvent, &dwEvtClass);  //Turn on event support.
pCP->Release();
if (FAILED(hr)) return FALSE;
...
return TRUE;
...
```

在此，**RecordsetEvent** 类的事件会被打开并且在 **Recordset** 事件出现时方法将被调用。

此后，如果您想使事件处理程序无效，可以再次获取连接点并发出 **IConnectionPoint::UnAdvise** 方法。

```
...
hr = pCP->UnAdvise(dwEvtClass);  //Turn off event support.
pCP->Release();
if (FAILED(hr)) return FALSE;
...
```

当然，必须在适当的时候释放接口并毁掉类的对象。

Visual J++

```
import wfc.data.*;
public class MyClass
{
ConnectionEventHandler handler =
   new ConnectionEventHandler(this,"onConnectComplete");

public void onConnectComplete(Object sender,ConnectionEvent e)
{
   if (e.adStatus == AdoEnums.EventStatus.ERRORSOCCURRED)
      System.out.println("Connection failed");
   else
      System.out.println("Connection completed");
   return;
}

void main( void )
{
   Connection conn = new Connection();

   conn.addOnConnectComplete(handler);      //打开事件支持。
   conn.open("DSN=pubs");
   conn.close();
```

```
    conn.removeOnConnectComplete(handler);    //关闭事件支持。
}
}
```

VBScript

VBScript 不支持事件。

1.1.3.6 VC++ Extensions for ADO

用 ADO 恢复数据时，Visual C++ 程序员所面对的一个最冗长而乏味的工作是必须将以 VARIANT 数据类型返回的数据转换为 C++ 数据类型，然后将转换后的数据存入类或结构中。除繁琐外，通过 VARIANT 数据类型恢复 C++ 数据会降低性能。

ADO 提供的接口支持直接将数据恢复到 C/C++ 自有数据类型中，而不通过 VARIANT 的转换。并提供预处理宏简化对接口的使用。由此获得一个高效易用的灵活工具。

普通 C/C++ 客户端方案将 **Recordset** 中的记录绑定到包含自有 C/C++ 类型的 C/C++ 结构/类上。使用 VARIANT 时，将涉及编写从 VARIANT 到 C/C++ 自有类型的转换代码。ADO VC++ Extensions 的目的便是使 VC++ 程序员更容易地实现该方案。

有关 ADO VC++ Extensions 的详细信息，请参阅如下主题：

- 使用 ADO VC++ Extensions(See 1.1.3.6.1)
- VC++ Extensions 头文件的详细资料(See 1.1.3.6.2)
- 范例：无 Extensions 的 ADO(See 1.1.3.6.3)
- 范例：带 Extensions 的 ADO(See 1.1.3.6.4)

1.1.3.6.1 使用 ADO VC++ Extensions

定义绑定条目

ADO VC++ Extensions 可将 **Recordset** 对象的字段映射到 C/C++ 变量，对字段与变量之间映射关系的定义称为“绑定条目”。预处理宏用来定义数值、定长和变长变量的绑定条目。

将 **BEGIN_ADO_BINDING** 和 **END_ADO_BINDING** 宏之间的绑定条目用括号括起。不要在绑定条目结尾使用逗号或分号，这些定界符仅限在宏中使用。

为每个将被转换为 C/C++ 变量的字段指定一个绑定条目。使用适当的 **ADO_FIXED_LENGTH_BINDING_ENTRY**、**ADO_NUMERIC_BINDING_ENTRY** 或 **ADO_VARIABLE_LENGTH_BINDING_ENTRY** 宏。

在宏的参数中，用序数指定将被提出的 **Recordset** 字段 — 0 标识第一字段，1 标识第二字段，依此类推。

使用数据类型声明 C/C++ 变量。如果变量为数值，也可指定精度和范围。如果变量为变长变量（如字符串），则必须以字节指定变量的最大尺寸。如果需要，**Recordset** 字段值可被强制为该数据类型。

指定临时的工作缓冲区，用来将字段值从 VARIANT 转换为 C/C++ 变量。缓冲区应至少与此 C/C++ 变量一样大。

将布尔型修改参数设置为 TRUE 使 ADO 可更新绑定的字段，如只检查字段而不将其更改，可设置为 FALSE。VC++ Extensions 不保留有关字段的状态信息，因此必须指定 ADO 是否更改字段值（例如，由数据源保留的自动增值字段的值）。因此该字段的修改参数应设置为 FALSE。

状态参数可告诉您从 **Recordset** 字段到 C 或 C++ 变量的转换是否成功以及变量的内容是否有效。该参数的两个最重要的值是 **adFldOK**（意味着转换成功）和 **adFldNull**（意味着字段是 NULL—无值可供转换）。

首先检测该参数以决定 C 或 C++ 变量是否有效。例如，如果字段具有有效的行内容，状态将会是 **adFldOK**；如果移动到另一个字段为 NULL 的行，状态则将是 **adFldNull**。然而，C 或 C++ 变量的内容将不被更改 — 该变量将仍然包含上一行的字段值。

将 Recordset 绑定到变量

在应用程序中，调用 **BindToRecordset** 接口方法可使 **Recordset** 字段关联（或绑定）到 C/C++ 变量，无论何时更改 **Recordset** 对象的当前行，C/C++ 变量都将自动更新。

头文件

要使用 VC++ Extensions，请在应用程序中包含如下文件：

- #include <icrsint.h>

接口方法

IADORecordBinding 接口具有使 **Recordset** 字段与 C/C++ 变量关联、添加新行和执行更新的方法。所有这三个方法都可使指针指向由 **CADORecordBinding** 派生的类，该 **CADORecordBinding** 定义每个字段和变量之间的绑定。
接口方法是：

- **BindToRecordset**(*&binding*)

调用该方法可使变量与字段相关联。

- **AddNew**(&*binding*)

调用该方法可间接调用 ADO **AddNew** 方法。

- **Update**(&*binding*)

调用该方法可间接调用 ADO **Update** 方法。

预处理宏

- BEGIN_ADO_BINDING(*cls*)
- ADO_FIXED_LENGTH_BINDING_ENTRY(*Ordinal*, *DataType*, *Buffer*, *Status*, *Modify*)
- ADO_NUMERIC_BINDING_ENTRY(*Ordinal*, *DataType*, *Buffer*, *Precision*, *Scale*, *Status*, *Modify*)
- ADO_VARIABLE_LENGTH_BINDING_ENTRY(*Ordinal*, *DataType*, *Buffer*, *Size*, *Status*, *Modify*)
- END_ADO_BINDING()

参数	**说明**
cls	类，定义绑定条目，缓冲区，和 **Recordset** 对象。
Ordinal	按顺序的字段号码，0 标识第一字段，1 标识第二字段，依此类推。

DataType	储存已转换字段的变量的数据类型。
Buffer	缓冲区，用于将字段转换为变量。
Status	指示字段转换是否成功。
Modify	布尔标志；如果为 TRUE，则表明 ADO 可以更新关联的字段。
Precision	在数值变量中可被表现出的数字位数。
Scale	位于数值变量中的小数点后的位数。
Size	变长变量所需的字节数，诸如：字符串。

状态参数值	**说明**
adFldOK	返回非 NULL 字段值。
adFldBadAccessor	绑定无效。
adFldCantConvertValue	由于符号不匹配和数据溢出以外的原因，值不能转换。
adFldNull	返回 NULL。
adFldTruncated	变长数据或数值型数字被截短。
adFldSignMismatch	值有符号，而变量数据类型无符号。
adFldDataOverFlow	值大于在变量数据类型中的存储大小。
adFldCantCreate	已打开未知列类型和字段。
adFldUnavailable	无法确定字段值 — 例如在无默认值的新建、未指定的字段中。
adFldPermissionDenied	更新时，不允许写入数据。
adFldIntegrityViolation	更新时，字段值将破坏列的完整性。
adFldSchemaViolation	更新时，字段值将破坏列模式。
adFldBadStatus	更新时，无效的状态参数。
adFldDefault	更新时，使用了默认值。

1.1.3.6.2 VC++ Extensions 头文件的详细资料

下列头文件 (icrsint.h) 详细说明了允许客户提取一行数据并直接送至类数据成员的接口。客户程序需要在其类中包含绑定条目，以指定 **Recordset Field** 对象和类数据成员之间的关联。

```
#ifndef _ICRSINT_H_
#define _ICRSINT_H_
```

```
#include <olectl.h>
#include <stddef.h>

// forwards
class CADORecordBinding;

#define classoffset(base, derived) \
   ((DWORD)(static_cast<base*>((derived*)8))-8)

enum FieldStatusEnum
{
   adFldOK = 0,
   adFldBadAccessor = 1,
   adFldCantConvertValue = 2,
   adFldNull = 3,
   adFldTruncated = 4,
   adFldSignMismatch = 5,
   adFldDataOverFlow = 6,
   adFldCantCreate = 7,
   adFldUnavailable = 8,
   adFldPermissionDenied = 9,
   adFldIntegrityViolation = 10,
   adFldSchemaViolation = 11,
   adFldBadStatus = 12,
   adFldDefault = 13
};

typedef struct stADO_BINDING_ENTRY
{
   ULONG        ulOrdinal;
   WORD        wDataType;
   BYTE        bPrecision;
   BYTE        bScale;
   ULONG        ulSize;
   ULONG        ulOffSet;
   ULONG        ulIADOBindingEntriesOffSet;
   ULONG        ulFldStatusOffset;
   BOOL        fModify;
} ADO_BINDING_ENTRY;

#define BEGIN_ADO_BINDING(cls) public: \
   typedef cls ADORowClass; \
   const ADO_BINDING_ENTRY* STDMETHODCALLTYPE GetADOBindingEntries() { \
   static const ADO_BINDING_ENTRY rgADOBindingEntries[] = {
```

```
#define ADO_FIXED_LENGTH_BINDING_ENTRY(Ordinal, DataType, Buffer, \    Status,
Modify)\
   {Ordinal, \
   DataType, \
   0, \
   0, \
   0, \
   offsetof(ADORowClass, Buffer), \
   classoffset(CADORecordBinding, ADORowClass), \
   offsetof(ADORowClass, Status), \
   Modify},

#define ADO_NUMERIC_BINDING_ENTRY(Ordinal, DataType, \
   Buffer, Precision, Scale, Status, Modify)\
   {Ordinal, \
   DataType, \
   Precision, \
   Scale, \
   0, \
   offsetof(ADORowClass, Buffer), \
   classoffset(CADORecordBinding, ADORowClass), \
   offsetof(ADORowClass, Status), \
   Modify},

#define ADO_VARIABLE_LENGTH_BINDING_ENTRY(Ordinal, DataType, \
   Buffer, Size, Status, Modify)\
   {Ordinal, \
   DataType, \
   0, \
   0, \
   Size, \
   offsetof(ADORowClass, Buffer), \
   classoffset(CADORecordBinding, ADORowClass), \
   offsetof(ADORowClass, Status), \
   Modify},

#define END_ADO_BINDING() {0, adEmpty, 0, 0, 0, 0, 0, adFldOK, FALSE}};\
   return rgADOBindingEntries;}

//
// 客户端“record”类需要支持的接口。
// ADO 绑定条目提供执行该接口。
//
class CADORecordBinding
```

```
{
public:
   STDMETHOD_(const ADO_BINDING_ENTRY*, GetADOBindingEntries)
   (VOID) PURE;
};

//
// 接口允许客户程序获取数据记录并传递给类数据成员。
//
DEFINE_GUID(IID_IADORecordBinding,
          0x00000544, 0, 0x10, 0x80,0,0,0xAA,0,0x6D,0x2E,0xA4);
DECLARE_INTERFACE_(IADORecordBinding, IUnknown)
{
public:
   STDMETHOD(BindToRecordset)(CADORecordBinding *pAdoRecordBinding)PURE;
   STDMETHOD(AddNew)(CADORecordBinding *pAdoRecordBinding) PURE;
   STDMETHOD(Update)(CADORecordBinding *pAdoRecordBinding) PURE;
};

#endif // !_ICRSINT_H_
```

1.1.3.6.3 范例：无 Extensions 的 ADO

```
#import "c:\Program Files\Common Files\System\ADO\msado15.dll"
   no_namespace rename("EOF", "EndOfFile")
#include <stdio.h>

Class CEmployee
{
public:
   FetchEmployeeData();

   char m_szFirstName[30];
   char m_szLastName[30];
   int nAge;
};

CEmployee::FetchEmployeeData()
{
_ConnectionPtr   pCon();
_RecordsetPtr    pRs();
FieldPtr        pfldFirstName, pfldLastName, pfldAge;
_variant_t      vFirstName, vLastName, vAge;
```

```
pCon.CreateInstance(__uuidof(Connection));
pCon->Open("pubs", "sa", "");

pRs.CreateInstance(__uuidof(Recordset));
pRs->Open("select FirstName, LastName, Age from Employees", pCon,
   adOpenForwardOnly, adLockReadOnly, adCmdUnknown);

pfldFirstName = pRs->Fields->GetItem(0);
pfldLastName = pRs->Fields->GetItem(1);
pfldAge = pRs->Fields->GetItem(2);

while (VARIANT_FALSE == pRs->EndOfFile)
   {
   vFirstName.Clear();
   vLastName.Clear();
   vAge.Clear();

   vFirstName = pfldFirstName->Value;
   WideCharToMultiByte(CP_ACP, 0, vFirstName.bstrVal, -1,
      m_szFirstName, sizeof(m_szFirstName), NULL, NULL);

   vLastName = pfldLastName->Value;
   WideCharToMultiByte(CP_ACP, 0, vLastName.bstrVal, -1,
      m_szLastName, sizeof(m_szLastName), NULL, NULL);

   nAge = vAge.iVal;

   pRs->MoveNext();
   }
}
```

1.1.3.6.4 范例：带 Extensions 的 ADO

该程序说明了如何从字段检索数值并将数值转换为 C++ 变量。它包括了在程序段（范例：无 Extensions 的 ADO(See 1.1.3.6.3)）中所描述的功能。

```
#define INITGUID
#import "c:\Program Files\Common Files\System\ADO\msado15.dll"
   no_namespace rename("EOF", "EndOfFile")
#include <stdio.h>
#include "icrsint.h"

void dump_com_error(_com_error &e)
   {
printf("Error\n");
```

```
printf("\a\tCode = %08lx\n", e.Error());
printf("\a\tCode meaning = %s", e.ErrorMessage());
_bstr_t bstrSource(e.Source());
_bstr_t bstrDescription(e.Description());
printf("\a\tSource = %s\n", (LPCSTR) bstrSource);
printf("\a\tDescription = %s\n", (LPCSTR) bstrDescription);
   }

class CCustomRs :
   public CADORecordBinding
{
BEGIN_ADO_BINDING(CCustomRs)
   ADO_VARIABLE_LENGTH_BINDING_ENTRY(1, adVarChar, m_szau_lname,
        sizeof(m_szau_lname), lau_lnameStatus, FALSE)
   ADO_VARIABLE_LENGTH_BINDING_ENTRY(2, adVarChar, m_szau_fname,
        sizeof(m_szau_fname), lau_fnameStatus, TRUE)
END_ADO_BINDING()

public:
   CHAR   m_szau_lname[41];
   ULONG   lau_lnameStatus;
   CHAR   m_szau_fname[41];
   ULONG   lau_fnameStatus;
};

VOID   main()
   {
   HRESULT hr;
   IADORecordBinding   *picRs = NULL;

   ::CoInitialize(NULL);

   try
      {
      _RecordsetPtr pRs.CreateInstance(__uuidof(Recordset));     CCustomRs rs;

      pRs->Open("select FirstName, LastName, Age from Employees",
         "dsn=pubs;uid=sa;pwd=;",
         adOpenStatic, adLockOptimistic, adCmdUnknown);

      if (FAILED(hr = pRs->QueryInterface(__uuidof(IADORecordBinding),
           (LPVOID*)&picRs)))
         _com_issue_error(hr);

      if (FAILED(hr = picRs->BindToRecordset(&rs)))
```

```
      _com_issue_error(hr);

   while (VARIANT_FALSE == pRs->EndOfFile)
      {
   // 处理 CCustomRs C++ 实例变量中的数据。

      printf("\a\tName = %s \t%s",
         (lau_fnameStatus == adFldOK ? m_szau_fname : "<NULL>"),
         (lau_lnameStatus == adFldOK ? m_szau_lname): "<NULL>"));

   // 更改 Recordset 的当前行。
   // 新当前行的 Recordset 数据将被
   // 自动取出并防止在 CCustomRs C++ 实例变量中。

      pRs->MoveNext();
      }
   }
catch (_com_error &e)
   {
   dump_com_error(e);
   }

if (picRs)
   picRs->Release();

CoUninitialize();
};
```

1.1.3.7 数据构形

ADO 使您能够回答其答案可表示为 **Recordset** 的提问。例如，假设您要公司客户的列表，而您有包含名为 Customers 的表的数据库。对表发出查询命令，则 ADO 将返回 **Recordset**，在 **Recordset** 中每个行表示一个客户，并且每个行的列拥有能够包含客户的名称、地址、客户 ID 等内容的数据类型。

“数据构形”使您能够回答其答案可由成形的 **Recordset** 表示的提问。数据构形定义成形 **Recordset** 的列、由列表示条目之间的关系和数据充填到 **Recordset** 的方式。

成形 **Recordset** 的列可以包含来自数据提供者（如 SQL Server）的数据、对另一个 **Recordset** 的引用、对 **Recordset** 一个行进行计算得到的值、对整个 **Recordset** 的列进行操作所得到的值，或者可以是新虚构的空列。

在检索包含对另一个 **Recordset** 的引用的列的值时，ADO 将自动返回由引用表示的实际的 **Recordset**。包含另一个 **Recordset** 的 **Recordset** 称为“分级 Recordset”或“分级游标”。

例如，假定您要得到由公司每个客户发出的所有定单的列表。可对包含名为 Customers 和 Orders 的数据库表发出数据构形命令。如同上例一样，ADO 将返回客户记录 **Recordset**。但是，每行将同时有附加的列，该列引用包含所有该客户定单的 **Recordset**。改进该形状即可获得按州排列的销售总计的列表。在 Customers 表中的地址列包含每个客户的州，而 Orders 表的数量列则包含每个定单的数量。发出不同的数据构形命令，则 ADO 将返回 **Recordset**，该 **Recordset** 包含每个州一个行（在此，每个行包含标识州的列）、该州所有定单的销售量总计和对开列该州的所有客户的 **Recordset** 的引用。访问该客户 **Recordset** 将会发现与前面的

范例一样，每个客户行均含有对包含所有该客户定单的 **Recordset** 的引用。

Shape 命令语法使您能够通过编程创建成形的 **Recordset**。然后就可以通过编程或适当的可视化控件，访问 **Recordset** 的组件。

有关数据构形的详细信息，请参阅如下主题：

1.1.3.7.1 数据构形总结

ADO 2.0 推出了数据构形功能、分级记录集和 Shape 命令语法。

ADO 2.1 通过插入 COMPUTE 命令推出重构形、孙子合计和参数化命令。

数据构形

数据构形使您能够定义成形 **Recordset** 的列、由列表示的条目之间的关系和数据充填到 **Recordset** 的方式。

成形 **Recordset** 的列可以包含数据、对另一个 **Recordset** 的引用、对 **Recordset** 一个行进行计算得到的值、对整个 **Recordset** 的列进行操作所得到的值，或者可以是新虚构的空列。

在检索包含对另一个 **Recordset** 的引用的列的值时，ADO 将自动返回由引用表示的实际的 **Recordset**。包含另一个 **Recordset** 的 **Recordset** 被称为“分级 **Recordset**”。分级 **Recordsets** 展示的是父-子关系，其中“父”是包含的 **Recordset**，而“子”是被包含的 **Recordset**。对 **Recordset** 的引用实际是对子的子集合（即“子集”）的引用。单个父可以包含多个子 **Recordset**。

ADO 2.0 同时推出了新的 Shape 命令语法，能够通过编程创建成形的 **Recordset** 对象。Shape 命令可以象其他任何 ADO 命令文本一样发出。

使用 Shape 命令语法，可通过两种途径创建分级的 **Recordset** 对象。其一是将子 **Recordset** 追加到父 **Recordset**，一般，父和子至少必需有一个列：在父的行中列的值与子的所有行中列的值相同。

其二则是从子 **Recordset** 产生父 **Recordset**。在引用子 **Recordset** 的父中，必须有子集列。创建其他父列的途径是：对子列的合计运算，运算 **Recordset** 行的表达式，使用 **BY** 关键字指定分组的列，或追加新的空列。

可将分级 **Recordset** 对象嵌套到所需的任何深度（即创建子 **Recordset** 对象的子 **Recordset** 对象，如此继续）。

通过程序或相应的可视控件，可以访问成形的 **Recordset** 的 **Recordset** 组件。

Microsoft 提供了能够生成 Shape 命令的可视化工具（请参阅 Visual Basic 主题，“数据环境设计者”）以及另一个能够显示分级游标的可视化工具（请参阅 Visual Basic 主题，“使用 Microsoft Hierarchical Flexgrid Control”）。

重构形

由 Shape 命令的子句创建的 **Recordset** 可以被赋值为“别名”（一般使用 **AS** 关键字）。在 ADO 2.1 中，成形 **Recordset** 的别名可以在完全不同的命令中引用。就是说，可以发出新的 Shape 命令来更改（即重构形）以前构形的 **Recordset**。为了支持该功能，

ADO 提供了新的 **Recordset** 对象 Name(See 1.1.4.6.41) 属性。

所受限制为不可以将列追加到现有的 **Recordset** 中，或在任何插入 COMPUTE 的子句中对参数化的 **Recordset** 或 **Recordset** 对象进行重构形，或对除正在被构形的 **Recordset** 以外的任何 **Recordset** 执行合计操作。

孙子合计

用 Shape 命令的子句创建的子集列可以得到“子集-别名名称”（一般使用 **AS** 关键字）。可以使用用以标识包含了列的子的完整名称，在子集中标识成形 **Recordset** 的任何列。例如，如果父子集 chap1 包含拥有数量列 amt 的子子集 chap2，那么完整名称将是 chap1.chap2.amt。然后，完整名称可以用作参数在合计函数 (SUM, AVG, MAX, MIN, COUNT, STDEV, or ANY) 中使用。

使用插入 COMPUTE 命令的参数化命令

典型的参数化形状 *APPEND* 命令含有通过查询命令创建父 **Recordset** 的子句，以及通过参数化查询命令（即包含参数占位符：问号“？”的命令）创建子 **Recordset** 的另一个子句。最终得到的成形 **Recordset** 有两层，父在上层而子占据下层。

创建子 **Recordset** 的子句现在可以是任意数量的嵌套形状 *COMPUTE* 命令，在最深的嵌套命令中包含了参数化查询。所得到的成形 **Recordset** 有多层，父占据最上层，子占据最下层，而由形状 *COMPUTE* 命令生成的任意数量 **Recordsets** 则占据插入的层。该特性典型的用法是调用合计函数和形状 *COMPUTE* 命令的分组功能，创建带有有关子 **Recordset** 的分析信息的插入 **Recordset** 对象。总之，因为这是参数化 Shape 命令，每当父的子集列被访问时，就可检索新的子 **Recordset**。因为插入层来源于子，它们也将被重新计算。

1.1.3.7.2 数据构形所需的提供者

数据构形一般需要两个提供者。服务提供者 OLE DB 的数据构形服务(See 1.1.5.7)提供数据构形功能，而数据提供者，例如 SQL Server 的 OLE DB 提供者，则提供充填成形 **Recordset** 的数据行。

服务提供者的名称可以指定为 **Connection** 对象 **Provider** 属性的值，或连接字符串关键字“**Provider=**”。

数据提供者的名称可以指定为“**Data Provider**”动态属性的值，该动态属性由 Data Shaping Service for OLE DB 添加到 **Connection** 对象 **Properties** 集合。或者也可以指定为连接字符串关键字“**Data Provider=**”。

如果没有充填 **Recordset**（例如，如果成形 **Recordset** 的所有列均使用 **NEW** 关键字创建），则不需要数据提供者。在这种情况下指定“**Data Provider=none**”。

范例

```
Dim cnn As New ADODB.Connection
cnn.Provider = "MSDataShape"
cnn.Open "Data Provider=MSDASQL;DSN=vfox;uid=sa;pwd=vfox;database=pubs”
```

1.1.3.7.3 常规 Shape 命令

“数据构形”定义了成形 **Recordset** 的列、由列代表的条目之间的关系以及数据充填到 **Recordset** 的方式。

成形的 **Recordset** 可以由如下类型的列组成：

列类型	说明
数据	由对数据提供者、表或以前成形 **Recordset** 使用查询命令所返回的 **Recordset** 的字段。

子集	对另一个 **Recordset** 的引用，称为“子集”。子集列使定义“父-子”关系成为可能，在这种关系中“父”是包含子集列的 **Recordset**，“子”是由子集代表的 **Recordset**。
合计	列的值通过对所有行执行“合计函数”获得，或者是子 **Recordset** 的所有行的列。（请参阅下表中的合计函数。）
计算表达式	列的值通过对在 **Recordset** 的相同行中的列进行 Visual Basic for Applications 表达式的计算而获得。表达式是 CALC 函数的参数。（请参阅下表中的计算表达式。）
新建	空的、虚构的字段，可在随后充填数据。列使用 NEW 关键字定义。（请参阅下表中的 NEW 关键字。）

Shape 命令可以包含子句，指定针对基本数据提供者并将返回 **Recordset** 对象的查询命令。查询的语法取决于对基本数据提供者的要求。虽然 ADO 并不要求使用任何指定的查询语言，但通常是使用结构化查询语言 (SQL)。

您可以使用 SQL **JOIN** 子句关联两个表，但是，分级 **Recordset** 可以更有效地表达信息。由 **JOIN** 创建的 **Recordset** 的每行会多余地重复一个表中的信息。分级 **Recordset** 的多个子 **Recordset** 对象中，每个对象仅有一个父 **Recordset**。

Shape 命令可以仅由 **Recordset** 对象发出。

Shape 命令可以嵌套，即父命令或子命令本身可以是另一个 Shape 命令。

有关定位分级 **Recordset** 的详细信息，请参阅访问分级 Recordset 中的行(See 1.1.3.7.6)。

有关语法正确的 Shape 命令的详细信息，请参阅形状语法格式(See 1.1.3.7.7)。

合计函数、CALC 函数和 NEW 关键字

数据构形支持如下函数。*chapter-alias* 是指定给包含了将被操作列的子集名称。

chapter-alias（子集-别名）可以是完整的，由指向包含 *column-name* 的子集的每个子集列名称组成，全部用句号分隔。例如，如果父子集 chap1 包含拥有数量列 amt 的子子集，则完整名即是 chap1.chap2.amt。

合计函数	**说明**
SUM(*chapter-alias.column-name*)	计算指定列中所有值的和。
AVG(*chapter-alias.column-name*)	计算指定列中所有值的平均值。
MAX(*chapter-alias.column-name*)	计算指定列中的最大值。
MIN(*chapter-alias.column-name*)	计算指定列中的最小值。
COUNT(*chapter-alias*[.*column-name*])	计算指定别名或列中行的数量。
STDEV(*chapter-alias.column-name*)	计算指定列中的标准偏差。
ANY(*chapter-alias.column-name*)	列的值（列的值在所有行均相同）。

计算表达式	**说明**
CALC(*expression*)	计算任意表达式，但仅针对包含 CALC 函数的 **Recordset** 行。可以是任何 Visual Basic for Applications (VBA) 函数或表达式。

NEW 关键字	说明
NEW (field type [(width | scale [,precision])]	将指定类型的空列添加到 **Recordset**。

1.1.3.7.4 Shape Append 命令

Shape APPEND 命令将子 **Recordset** 分配给父 **Recordset** 中 **Field** 对象的 **Value** 属性。

语法

SHAPE **{***parent-command***}** [[AS] *parent-alias*]
APPEND **({***child-command***}** [AS] *child-alias*
RELATE *parent-column* TO *child-column***...)** [[AS] *chapter-alias*] **...**

组成说明

该命令的组成部分为：
parent-command, *child-command* 如下之一。

- 在尖括号（“**{}**”）中的查询命令，返回 **Recordset** 对象。命令发布给基本数据提供者，其语法取决于该提供者的要求。虽然 ADO 并不要求使用任何指定的查询语言，但通常是使用结构化查询语言(SQL)。圆括号（“**()**”）是必需的关键字，它们将子集列追加到引用由查询命令返回的 **Recordset** 的父。
- 以前成形的 **Recordset** 的名称。
- 另一个 Shape 命令。
- TABLE 关键字，后跟表的名称。

parent-column 由 *parent-command* 返回的 **Recordset** 中的列。
child-column 由 *child-command* 返回的 **Recordset** 中的列。
... “*parent-column* **TO** *child-column*”子句实际上是列表，并用逗号将每个定义关系分隔开。
chapter-alias 别名，对追加到父的列的引用。
parent-alias 别名，对父 **Recordset** 的引用。
child-alias 别名，对子 **Recordset** 的引用。
... 在 APPEND 关键字后面的子句实际上是列表（每个子句使用逗号分隔），定义被追加到父的另一个列。

操作

发出 *parent-command* 并返回父 **Recordset**。然后发出 *child-command* 并返回子 **Recordset**。
例如，*parent-command* 可以从客户表返回公司的客户 **Recordset**，而 *child-command* 从定货表返回所有客户的定单 **Recordset**。一般，父和子 **Recordset** 对象必须各自拥有用于关联父和子的列。列在 **RELATE** 子句中命名，*parent-column* 在先，*child-column* 在后。在各自的 **Recordset** 中，列可以有不同名称，但必须引用相同信息以便指定有意义的关系。例如，***Customers*** 和 ***Orders***

Recordsets 可以同时拥有 ***customerID*** 字段。

数据构形将子集列追加到父 **Recordset**。子集列中的值是对子 **Recordset** 中列的引用，子 **Recordset** 满足 **RELATE** 子句。即在给定父行中的 *parent-column* 与在子集子的所有行中的 *child-column* 具有相同的值。

当您访问在子集列中的引用时，ADO 将自动检索由引用表示的 **Recordset**。注意尽管已经检索了全部子 **Recordset**，但子集（*chapter*）仅表示行的子集。

如果追加的列没有 *chapter-alias*，则会自动生成其名称。列的 **Field** 对象将被追加到 **Recordset** 对象的 **Fields** 集合，其数据类型将是 **adChapter**。

有关定位分级 **Recordset** 的详细信息，请参阅访问分级 Recordset 中的行(See 1.1.3.7.6)。

参数化命令

如果您正在处理大的子 **Recordset**（尤其是比父 **Recordset** 大），却只需要访问部分子子集，那么，使用参数化命令会更有效。

non-parameterized command（非参数化命令）同时检索整个父和子 **Recordsets**，并将子集列追加到父，然后为每个父行指定相关子子集的引用。

parameterized command（参数化命令）检索整个父 **Recordset**，但在访问子集列时仅检索子集 **Recordset**。这种检索策略的差别可以有益的性能好处。

例如，可以指定如下：

```
"SHAPE {SELECT * FROM customer}
APPEND ({SELECT * FROM orders WHERE cust_id = ?}
RELATE cust_id TO PARAMETER 0)"
```

父和子表通常拥有列名 *cust_id*。*child-command* 有占位符（即“？”），受 **RELATE** 子句引用（即“...PARAMETER 0”）。关系在于显性标识的 **customer** 表 *parent-column*（即 **cust_id**）和隐性标识的 **orders** 表 *child-column*（即 **cust_id**）之间，由占位符和“PARAMETER 0”指定。

注意 PARAMETER 子句仅属于 Shape 命令语法。与 ADO **Parameter** 属性和 **Parameters** 集合均无关联。

在执行 Shape 命令时，发生如下情形：

1. 执行 *parent-command*，并返回 **customer** 表的父 **Recordset**。
2. 子集列被追加到父 **Recordset**。
3. 在访问父行的子集列时，**customer.cust_id** 列的值将替换 **orders.cust_id** 的占位符，并执行 *child-command*。
4. **orders** 表（在此，**orders.cust_id** 列的值与 **customer.cust_id** 列的值相匹配）的所有行被检索。
5. 对检索到的子行（即子 **Recordset** 的 *chapter*）的引用被放置在父 **Recordset** 当前行的子集列。
6. 当访问另一个行的子集列时，重复步骤 3-5。

插入 Shape *COMPUTE* 命令

现在将参数化 Shape 命令的参数化命令嵌入任意嵌套数量的形状 *COMPUTE* 命令中是有效的。例如：

```
SHAPE {select au_lname, state from authors} APPEND
((SHAPE
```

```
(SHAPE
{select * from authors where state = ?} rs
COMPUTE rs, ANY(rs.state) state, ANY(rs.au_lname) au_lname
BY au_id) rs2
COMPUTE rs2, ANY(rs2.state) BY au_lname)
RELATE state TO PARAMETER 0)
```

1.1.3.7.5 Shape Compute 命令

Shape COMPUTE 命令生成父 **Recordset**（其列由对子 **Recordset** 的引用组成）、可选的列（其内容是对子 **Recordset** 或以前成形的 **Recordset** 执行合计函数的结果）和在可选的 BY 子句中开列出的任何子 **Recordset** 的列。

语法

"SHAPE **{***child-command***}** [AS] *child-alias*
COMPUTE *child-alias* [,*aggregate-command-field-list*]
[BY *grp-field-list*]"

组成说明

该命令的组成是：
child-command 如下之一。

- 在尖括号（“**{}**”）中的查询命令，返回 **Recordset** 对象。命令发布给基本数据提供者，其语法取决于该提供者的要求。虽然 ADO 并不要求使用任何指定的查询语言，但通常是使用结构化查询语言 (SQL)。
- 以前成形的 **Recordset** 的名称。
- 另一个形状（**Shape**）命令。
- TABLE 关键字，后跟表的名称。

child-alias 别名，用于引用由 *child-command* 返回的 **Recordset**。在 COMPUTE 子句的列的列表中需要 *child-alias*，用于定义父和子 **Recordset** 对象的关系。
aggregate-command-field-list 列表，定义在生成的父中的列，含有对子 **Recordset** 执行合计函数所产生的值。
grp-field-list 在父和子 **Recordset** 对象中的列的列表，指定在子中的行如何分组。
对在 *grp-field-list* 中的每个列，在父和子 **Recordset** 对象中有对应的列。对父 **Recordset** 的每个行，*grp-field-list* 列有唯一的值，并且由父行引用的子 **Recordset** 由子行（其 *grp-field-list* 列含有与父行相同的值）单独组成。
如果 **COMPUTE** 子句包含合计函数，但没有 **BY** 子句，那么，只有一个父行含有整个子 **Recordset** 的合计值。如果有 **BY** 子句，那么，有多个父行均分别含有引用和子 **Recordset** 的合计值。

操作

child-command 被发布给提供者，并返回子 **Recordset**。

COMPUTE 子句指定父 **Recordset** 的列，该 **Recordset** 可以是对子 **Recordset** 的引用、一个或多个合计、计算表达式或新列。如果有 **BY** 子句，那么，它定义的列同时被追加到父 **Recordset** 中。**BY** 子句指定子 **Recordset** 的行分组的方式。

例如，假定有一个人口统计表，包括 State、City 和 Population 字段（人口数字单独说明）。

State	City	Population
WA	Seattle	700,000
OR	Medford	200,000
OR	Portland	600,000
CA	Los Angeles	900,000
CA	San Diego	400,000
WA	Tacoma	500,000
OR	Corvallis	300,000

现在，发出该 Shape 命令：

```
rst.Open     "SHAPE  {select * from demographics} AS rs
           COMPUTE SUM(rs.population), rs
           BY state",
           connection
```

该命令打开具有两个层次的成形 **Recordset**。父层是生成的 **Recordset**，有合计列 (SUM(rs.population))、引用子 **Recordset** (rs) 的列和分组 **Recordset** （州）的列。子层是由查询命令 (select * from demographics) 返回的 **Recordset**。

子 **Recordset** 具体行将由 State 分组，但不按照特定的顺序。即分组将不采用字母或数字顺序。

现在，您可以定位打开的父 **Recordset**，并访问具体的子 **Recordset** 对象。请参阅访问分级 Recordset 中的行(See 1.1.3.7.6)。

所得到的父和子具体的 Recordsets

父

SUM (rs.Population)	rs	State
1,300,000	Reference to child1	CA
1,200,000	Reference to child2	WA
1,100,000	Reference to child3	OR

子 1

State	City	Population
CA	Los Angeles	900,000

CA	San Diego	400,000

子 2

State	City	Population
WA	Seattle	700,000
WA	Tacoma	500,000

子 3

State	City	Population
OR	Medford	200,000
OR	Portland	600,000
OR	Corvallis	300,000

1.1.3.7.6 访问分级 Recordset 中的行

以下范例说明了访问分级 **Recordset** 中的行的所需步骤：

1. authors 和 titleauthors 表中的 **Recordset** 对象通过 author ID 进行关联。
2. 外循环显示每个作者的姓名、州/省别和身份。
3. 每行所追加的 **Recordset** 都从 **Fields** 集合进行检索并分配给 rstTitleAuthor。
4. 内循环显示追加的 **Recordset** 中每行的四个字段。

（**StayInSync** 属性是为了说明而设置为 FALSE 的，以便您可以在每次外循环中显性地看见子集更改。但是，如果在步骤 3 中的赋值被移动到步骤 2 第一行之前，范例将会更有效，所以赋值只执行一次。然后将 **StayInSync** 属性设为 TRUE，这样无论 rst 何时移动到新行，rstTitleAuthor 都将隐性和自动地更改为相应的子集。）

范例

```
Sub datashape()
    Dim cnn As New ADODB.Connection
    Dim rst As New ADODB.Recordset
    Dim rstTitleAuthor As New ADODB.Recordset
```

```
    cnn.Provider = "MSDataShape"
    cnn.Open    "Data Provider=MSDASQL;" & _
              "DSN=vfox;uid=sa;pwd=vfox;database=pubs”
‘步骤 1
    rst.StayInSync = FALSE
    rst.Open    "SHAPE  {select * from authors}
              APPEND ({select * from titleauthor}
              RELATE au_id TO au_id) AS chapTitleAuthor",
              cnn
‘步骤 2
    While Not rst.EOF
        Debug.Print    rst("au_fname"), rst("au_lname"),
                  rst("state"), rst("au_id")
‘步骤 3
        Set rstTitleAuthor = rst("chapTitleAuthor").Value
‘步骤 4
        While Not rstTitleAuthor.EOF
            Debug.Print rstTitleAuthor(0), rstTitleAuthor(1),
                     rstTitleAuthor(2), rstTitleAuthor(3)
            rstTitleAuthor.MoveNext
        Wend
        rst.MoveNext
    Wend
End Sub
```

1.1.3.7.7 形状语法格式

以下是创建 Shape 命令的规范语法。

- 必需的语法项为尖括号（“<>”）界定的文本字符串。
- 可选项由方括号（“[]”）界定。
- 任选其一项用竖线（“|”）隔开。
- 重复的可选项用省略号（“…”）表示。
- Alpha 表示字母字符串。
- Digit 表示数字字符串。
- Unicode-digit 表示由 unicode 数字组成的字符串。

所有其他项目均为文字。

项目	定义
<shape-command>	SHAPE [<table-exp> [[AS] <alias>]][<shape-action>]
<table-exp>	{<native-sql-statement>} | (<shape-command>) | TABLE <quoted-name> | <quoted-name>
<shape-action>	APPEND <aliased-field-list> | COMPUTE <aliased-field-list> [BY <field-list> [[AS] <alias>]]
<aliased-field-list>	<aliased-field> [, <aliased-field...]
<aliased-field>	<field-exp> [[AS] <alias>]
<field-exp>	(<relation-exp>) | <calculated-exp>
<relation_exp>	<table-exp> [[AS] <alias>] RELATE <relation-cond-list>
<relation-cond-list>	<relation-cond> [, <relation-cond>...]
<relation-cond>	<field-name> TO <child-ref>
<child-ref>	<field-name> | PARAMETER <param-ref>
<param-ref>	<number>
<field-list>	<field-name [, <field-name>]
<calculated-exp>	SUM(<qualified-field-name>) | AVG(<qualified-field-name>) | MIN(<qualified-field-name>) | MAX(<qualified-field-name>) | COUNT(<alias> | <qualified-field-name>) | STDEV(<qualified-field-name>) | ANY(<qualified-field-name>) | CALC(<expresion>)
<qualified-field-name>	<alias>.[<alias>...]<field-name>

<alias>	<quoted-name>
<field-name>	<quoted-name>
<quoted-name>	"<string>" | '<string>' | [<string>] | <name>
<name>	*alpha* [*alpha* | *digit* | _ | # ...]
<number>	*digit* [*digit*...]
<string>	*unicode-char* [*unicode-char*...]
<expression>	Visual Basic for Applications 表达式，其操作数是相同行中其他非 CALC 列。

1.1.3.8 DataFactory 自定义

“远程数据服务”(RDS) 提供在三层客户端/服务器系统中进行数据访问的简便方法。客户数据控制程序指定连接和命令字符串参数以便进行远程数据源查询，或者指定连接字符串和 **Recordset** 对象参数以便进行更新。

参数被传送到服务器程序，该服务器程序在远程数据源中实际执行数据访问操作。RDS 提供被称为 **RDSServer.DataFactory** 对象的默认服务器程序。**RDSServer.DataFactory** 对象返回由客户查询而产生的 **Recordset** 对象。

然而，**RDSServer.DataFactory** 仅限于执行查询和更新。它不能对连接或命令字符串执行任何验证和处理。

有了 ADO 2.0，就可以指定 **DataFactory** 与称为“处理程序”的另一类服务器程序一起使用。在客户连接和命令字符串被用于访问数据源之前，处理程序可以对其进行修改。此外，处理程序可以加强访问权限，该权限决定客户读写数据源数据的能力。

处理程序用来修改客户端参数并访问权限的参数在自定义文件的节中指定。

有关自定义 **DataFactory** 对象的详细信息，请参阅如下主题：

- 了解自定义文件(See 1.1.3.8.1)
- 自定义文件 Connect 节(See 1.1.3.8.2)
- 自定义文件 SQL 节(See 1.1.3.8.3)
- 自定义文件 userlist 节(See 1.1.3.8.4)
- 自定义文件 logs 节(See 1.1.3.8.5)
- 所需客户端设置(See 1.1.3.8.6)
- 编写自己的自定义处理程序(See 1.1.3.8.7)

1.1.3.8.1 了解自定义文件

自定义文件中的每个节标头均由方括号 ([ ]) 及方括号内的类型和参数。4 个节类型由文字字符串 connect、sql、userlist 或 logs 表示。参数为文字字符串、默认值、用户指定的标识符或无。

因此，每节都必须标记有以下节标头之一：

```
[ connect   default     ]
[ connect   identifier   ]
[ sql        default     ]
[ sql        identifier   ]
[ userlist   identifier   ]
[ logs                ]
```

以下是节标头的组成部分：

组成部分	说明
connect	文字字符串 — 该节修改连接字符串。
sql	文字字符串 — 该节修改命令字符串。
userlist	文字字符串 — 该节修改特定用户的访问权限。
logs	文字字符串 — 该节指定记录操作错误的日志文件。
default	文字字符串 — 如果没有指定或找到标识符，则使用该节。
identifier	与连接或命令字符串相匹配的字符串。 • 如果节标头包含 **connect** 而且连接字符串中有标识符字符串，则使用该节。 • 如果节标头包含 **sql** 而且命令字符串含有标识符字符串，则使用该节。 • 如果节标头含有 **userlist** 而且标识符字符串与 **Connect** 节标识符相匹配，则使用该节。

-

DataFactory 调用处理程序，传递客户端参数。处理程序在客户端参数中搜索与相应节标头的标识符匹配的完整字符串。如果找到匹配的字符串，该节的内容将应用于客户端的参数。

下列情况可以使用特定的节：

- 如果客户端连接字符串关键字“**Data Source**=value”中的值与 **Connect** 节的标识符相匹配，则使用 **Connect** 节。
- 如果客户命令字符串含有与 **sql** 节标识符匹配的字符串，则使用 **sql** 节。

- 如果没有匹配的标识符，则使用带有默认参数的 **connect** 或 **sql** 节。
- 如果 **userlist** 节标识符与 **connect** 节标识符匹配，则使用 **userlist** 节。如果有匹配的，**userlist** 节的内容将被应用于 **connect** 节决定的连接。
- 如果连接字符串或命令的字符串与任何 **connect** 或 **sql** 节标头的标识符不匹配，并且没有带默认参数的 **connect** 或 **sql** 节标头，那么未经修改即可使用客户字符串。
- 无论何时操作 **DataFactory** 都可以使用 **logs** 节。

1.1.3.8.2 自定义文件 Connect 节

Connect 节可包含：

- 新的连接字符串，用于替代客户端连接字符串。
- 默认访问条目，用于指定该连接允许的默认读写操作。如果没有默认访问项目，该节会被忽略。

语法

替代连接字符串条目的格式：

Connect=*connectionString*

默认访问条目的格式：

Access=*accessRight*

组成部分	说明
Connect	文字字符串 — 指明这是连接字符串条目。
connectionString	字符串，用于替代整个客户端连接字符串。
Access	文字字符串 — 指明这是访问条目。
accessRight	以下访问权限之一： • **NoAccess** — 用户不能访问数据源。 • **ReadOnly** — 用户可以读取数据源。 • **ReadWrite** — 用户可读取或写入数据源。

1.1.3.8.3 自定义文件 SQL 节

SQL 节可包含新的 SQL 字符串，用于替代客户端命令字符串。如果节内没有 SQL 字符串，则该节将被忽略。

新的 SQL 字符串可以“参数化”。也就是说，**sql** 节的 SQL 字符串（由‘？’标明）中的参数，可以用客户端命令字符串（由一个用括号括起的、以逗号分隔的列表标明）的标识符中的相应参数替代。标识符和括号内的参数列表可以象函数调用一样操作。

例如，假设客户端命令字符串为“CustomerByID(4)”，SQL 节标头为 [SQL CustomerByID]，而新的 SQL 节的字符串为“SELECT * FROM Customers WHERE CustomerID = ?”。处理程序将生成“SELECT * FROM Customers WHERE CustomerID = 4”并使用该字符串查询数据源。

如果新的 SQL 语句为空字符串("")，则该节将被忽略。

如果新的 SQL 语句字符串无效，则该语句执行会失败，客户端参数实际上被忽略。使用如下指定可由此“故意”关闭客户端的所有 SQL 命令：

```
[SQL default]
SQL = " "
```

语法

替代 SQL 字符串条目的格式：

SQL=*sqlString*

组成部分	说明
SQL	文字字符串 — 指示这是 SQL 节条目。
sqlString	SQL 字符串，用于替代客户端字符串。

1.1.3.8.4 自定义文件 userlist 节

该 **userlist** 节与带有相同节标识符参数的 **connect** 节有关。

该节可包含“用户访问条目”，该条目为指定的用户指定访问权限并且覆盖匹配的 **connect** 节的“默认访问条目”。

语法

用户访问条目的格式：

userName=***accessRights***

组成部分	说明
userName	使用该连接的“用户名”。有效的用户名是通过 IIS **Service Manager** 对话框创建的。
accessRights	以下访问权限之一： • **NoAccess** — 用户不能访问数据源。 • **ReadOnly** — 用户可以读取数据源。 • **ReadWrite** — 用户可以读取或写入数据源。

1.1.3.8.5 自定义文件 logs 节

logs 节包含日志文件条目，用于指定在 **DataFactory** 的操作过程中记录错误的文件名。

语法

日志文件条目格式：

err=*FileName*

组成部分	说明
err	文字字符串 — 指明这是日志文件条目。
FileName	完整的路径和文件名。典型的文件名是 **c:\msdfmap.log**。

日志文件包含用户名、HRESULT、日期和每次出错的时间。

1.1.3.8.6 所需客户端设置

指定下列设置以便使用自定义 DataFactory 处理程序。

- 在 **Connection** 对象的 **Provider** 属性或者 **Connection** 对象的连接字符串 “**Provider=**” 的关键字中指定 “Provider=MS Remote”。
- 设置 **CursorLocation** 属性为 **adUseClient**。
- 指定处理程序的名称以便在 **RDS.DataControl** 对象的 **Handler** 属性、或者 **Recordset** 对象的连接字符串“**Handler=**”的关键字中使用。（不能在 **Connection** 对象连接字符串中设置处理程序）

RDS 在称为 **MSDFMAP.Handler** 的服务器上提供默认处理程序。（默认自定义文件被称为 **MSDFMAP.INI**。）

范例

假设有下列 **MSDFMAP.INI** 节以及数据源名称 AdvWorks 事先已有定义：

```
[connect CustomerDataBase]
Access=ReadWrite
Connect="DSN=AdvWorks"

[sql CustomerById]
SQL="SELECT * FROM Customers WHERE CustomerID = ?"
```

如下代码片段使用 Visual Basic 编写：

RDS.DataControl 版本

```
Dim dc as New RDS.DataControl
Set dc.Handler = "MSDFMAP.Handler"
```

```
Set dc.Server = "http://YourServer"
Set dc.Connect = "Data Source=CustomerDatabase"
Set dc.SQL = "CustomerById(4)"
dc.Refresh
```

Recordset 版本

```
Dim rs as New ADODB.Recordset
rs.CursorLocation = adUseClient
```

指定 **Handler** 属性或关键字；**Provider** 属性或关键字；**CustomerById** 和 **CustomerDatabase** 标识符。然后打开 **Recordset** 对象。

```
rs.Open "CustomerById(4)", "Handler=MSDFMAP.Handler;" & _
   "Provider=MS Remote;Data Source=CustomerDatabase;" & _
   "Remote Server=http://YourServer"
```

1.1.3.8.7 编写自己的自定义处理程序

如果您是需要默认 RDS 支持的 IIS 服务器管理员，但需要进一步控制用户请求和访问权限，则可能需要编写自己的处理程序。

MSDFMAP.Handler 实现 **IDataFactoryHandler** 接口。

IDataFactoryHandler 接口

该接口有两种方法，即 **GetRecordset** 和 **Reconnect**。两种方法都要求将 **CursorLocation** 属性设置为 **adUseClient**。

两种方法都取“**Handler**=”关键词的第一个逗号后面出现的参数。例如，“Handler=progid,arg1,arg2;”将传递“arg1,arg2”的参数字符串，而“Handler=progid”将传递参数 NULL。

GetRecordset 方法

该方法使用提供的参数查询数据源并创建新的 **Recordset** 对象。**Recordset** 必须使用 **adLockBatchOptimistic** 打开，不能异步打开。

参数

conn 连接字符串。

args 处理程序参数。

query 产生查询所用的命令文本。

ppRS 指向返回 **Recordset** 的位置。

Reconnect 方法

该方法更新数据源。它创建新的 **Connection** 对象，并附加给定的 **Recordset**。

参数

conn 连接字符串。

args 处理程序参数。

pRS **Recordset** 对象。

msdfhdl.idl

这是出现在 **msdfhdl.idl** 文件中对 **IDataFactoryHandler** 的接口定义。

```
[
  uuid(D80DE8B3-0001-11d1-91E6-00C04FBBBFB3),
```

```
  version(1.0)
]
library MSDFHDL
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    // TLib : Microsoft ActiveX Data Objects 2.0 Library
    // {00000200-0000-0010-8000-00AA006D2EA4}
    #ifdef IMPLIB
    importlib("implib\\x86\\release\\ado\\msado15.dll");
    #else
    importlib("msado20.dll");
    #endif

    [
      odl,
      uuid(D80DE8B5-0001-11d1-91E6-00C04FBBBFB3),
      version(1.0)
    ]
    interface IDataFactoryHandler : IUnknown
    {
HRESULT _stdcall GetRecordset(
     [in] BSTR conn,
     [in] BSTR args,
     [in] BSTR query,
     [out, retval] _Recordset **ppRS);

// DataFactory 将在调用 Reconnect 后
// 使用记录集的 ActiveConnection 属性。
   HRESULT _stdcall Reconnect(
      [in] BSTR conn,
      [in] BSTR args,
      [in] _Recordset *pRS);
    };
};
```

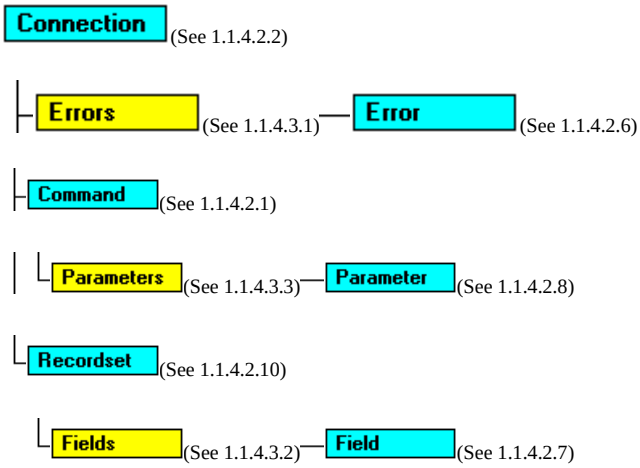
1.1.4 ADO API 参考

本节 ADO 文档包含各个 ADO 对象、集合、方法、事件和属性的主题。详细信息，请在索引中搜索指定主题或参考如下主题：

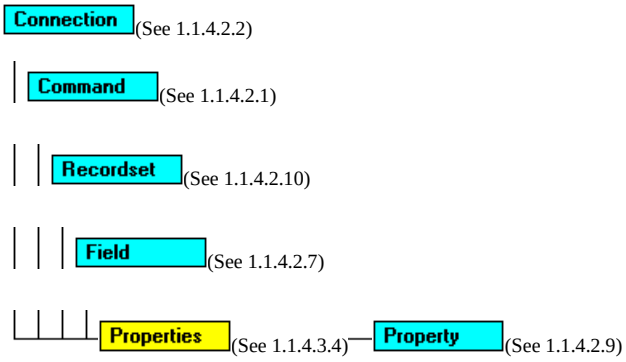
- ADO 对象(See 1.1.4.2)
- ADO 集合(See 1.1.4.3)

- ADO 方法(See 1.1.4.4)
- ADO 事件(See 1.1.4.5)
- ADO 属性(See 1.1.4.6)

1.1.4.1 ADO 对象模型



每个 **Connection**、**Command**、**Recordset** 和 **Field** 对象都有 **Properties** 集合。



1.1.4.2 ADO 对象

ADO 对象总结

对象	说明
Command(See 1.1.4.2.1)	**Command** 对象定义了将对数据源执行的指定命令。
Connection(See 1.1.4.2.2)	代表打开的、与数据源的连接。
DataControl (RDS)(See 1.1.4.2.3)	将数据查询 **Recordset** 绑定到一个或多个控件上（例如，文本框、网格控件或组合框），以便在 Web 页上显示 **ADOR.Recordset** 数据。

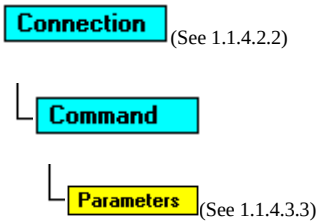
DataFactory (RDS Server)(See 1.1.4.2.4)	实现对客户端应用程序的指定数据源进行读/写数据访问的方法。
DataSpace (RDS)(See 1.1.4.2.5)	创建客户端代理以便自定义位于中间层(See 2.32)的业务对象(See 2.7)。
Error(See 1.1.4.2.6)	包含与单个操作（涉及提供者）有关的数据访问错误的详细信息。
Field(See 1.1.4.2.7)	代表使用普通数据类型的数据的列。
Parameter(See 1.1.4.2.8)	代表与基于参数化查询或存储过程的 **Command** 对象相关联的参数或自变量。
Property(See 1.1.4.2.9)	代表由提供者定义的 ADO 对象的动态特性。
RecordSet(See 1.1.4.2.10)	代表来自基本表或命令执行结果的记录的全集。任何时候，**Recordset** 对象所指的当前记录均为集合内的单个记录。

这些对象之间的关系在 ADO 对象模型(See 1.1.4.1)中表现。

每个对象可以包含在对应的集合中。例如，**Error** 对象可以包含在 Errors(See 1.1.4.3.1) 集合中。详细信息，请参阅 ADO 集合(See 1.1.4.3)或指定的集合主题。

1.1.4.2.1 Command 对象 (ADO)

Command 对象定义了将对数据源执行的指定命令。



(See 1.1.4.2.2)

(See 1.1.4.3.3)

说明

使用 **Command** 对象查询数据库并返回 Recordset(See 1.1.4.2.10) 对象中的记录，以便执行大量操作或处理数据库结构。取决于提供者的功能，某些 **Command** 集合、方法或属性被引用时可能会产生错误。

可以使用 **Command** 对象的集合、方法、属性进行下列操作：

- 使用 **CommandText** 属性定义命令（例如，SQL 语句）的可执行文本。
- 通过 Parameter(See 1.1.4.2.8) 对象和 **Parameters** 集合定义参数化查询或存储过程参数。
- 可使用 **Execute** 方法执行命令并在适当的时候返回 **Recordset** 对象。
- 执行前应使用 **CommandType** 属性指定命令类型以优化性能。

- 使用 **Prepared** 属性决定提供者是否在执行前保存准备好（或编译好）的命令版本。
- 使用 **CommandTimeout** 属性设置提供者等待命令执行的秒数。
- 通过设置 **ActiveConnection** 属性使打开的连接与 **Command** 对象关联。
- 设置 **Name** 属性将 **Command** 标识为与 **Connection** 对象关联的方法。
- 将 **Command** 对象传送给 **Recordset** 的 **Source** 属性以便获取数据。

注意 如果不想使用 Command 对象执行查询，请将查询字符串传送给 Connection 对象的 Execute 方法或 Recordset 对象的 Open 方法。但是，当需要使命令文本具有持久性并重新执行它，或使用查询参数时，则必须使用 Command 对象。

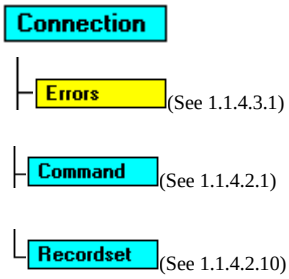
要独立于先前已定义的 Connection(See 1.1.4.2.2) 对象创建 **Command** 对象，请将它的 **ActiveConnection** 属性设置为有效的连接字符串。ADO 仍将创建 **Connection** 对象，但它不会将该对象赋给对象变量。但是，如果正在将多个 **Command** 对象与同一个连接关联，则必须显式创建并打开 **Connection** 对象，这样即可将 **Connection** 对象赋给对象变量。如果没有将 **Command** 对象的 **ActiveConnection** 属性设置为该对象变量，则即使使用相同的连接字符串，ADO 也将为每个 **Command** 对象创建新的 **Connection** 对象。

要执行 **Command**，只需通过它所关联的 **Connection** 对象的 **Name** 属性，将其简单调用即可。必须将 **Command** 的 **ActiveConnection** 属性设置为 **Connection** 对象。如果 **Command** 带有参数，则将这些参数的值作为参数传送给方法。

如果在相同连接上执行两个或多个 **Command** 对象，并且某个 **Command** 对象是带输出参数的存储过程，这时会发生错误。要执行各个 **Command** 对象，请使用独立的连接或将所有其他 **Command** 对象的连接断开。

1.1.4.2.2 Connection 对象 (ADO)

Connection 对象代表打开的、与数据源的连接。



说明

Connection 对象代表与数据源进行的唯一会话。如果是客户端/服务器数据库系统，该对象可以等价于到服务器的实际网络连接。取决于提供者所支持的功能，**Connection** 对象的某些集合、方法或属性有可能无效。

使用 **Connection** 对象的集合、方法、和属性可执行下列操作：

- 在打开连接前使用 **ConnectionString**、**ConnectionTimeout** 和 **Mode** 属性对连接进行配置。
- 设置 **CursorLocation** 属性以便调用支持批更新的“客户端游标提供者”。
- 使用 **DefaultDatabase** 属性设置连接的默认数据库。

- 使用 **IsolationLevel** 属性为在连接上打开的事务设置隔离级别。
- 使用 **Provider** 属性指定 OLE DB 提供者。
- 使用 **Open** 方法建立到数据源的物理连接。使用 **Close** 方法将其断开。
- 使用 **Execute** 方法执行对连接的命令，并使用 **CommandTimeout** 属性对执行进行配置。
- 可使用 **BeginTrans**、**CommitTrans** 和 **RollbackTrans** 方法以及 **Attributes** 属性管理打开的连接上的事务（如果提供者支持则包括嵌套的事务）。
- 使用 **Errors** 集合检查数据源返回的错误。
- 通过 **Version** 属性读取使用中的 ADO 执行版本。
- 使用 **OpenSchema** 方法获取数据库模式信息。

注意 如果不使用 **Command** 对象执行查询，请向 **Connection** 对象的 **Execute** 方法传送查询字符串。但是，当需要使命令文本具有持久性并重新执行，或使用查询参数的时候，则必须使用 **Command** 对象。

可以创建独立于先前定义的其他任何对象的 **Connection** 对象。

注意 可以象执行 **Connection** 对象的本地方法一样执行命令或存储过程。

如果要执行命令，可以使用 **Command** 对象的 **Name** 属性给命令指定一个名称。将 **Command** 对象的 **ActiveConnection** 属性设置为该连接。然后，象发出 **Connection** 对象的方法一样发出使用命令名称的语句，后面可带任何参数（如果有返回行，则后面带 **Recordset** 对象）。设置 **Recordset** 属性以便自定义所产生的记录集。例如：

```
Dim cnn As New ADODB.Connection
Dim cmd As New ADODB.Command
Dim rst As New ADODB.Recordset
...
cnn.Open "..."
cmd.Name = "yourCommandName"
cmd.ActiveConnection = cnn
...
'命令名称、任意参数、以及可选记录集。
cnn.yourCommandName "parameter", rst
```

要执行存储过程，可以如同发出 **Connection** 对象的方法一样发出使用存储过程名称的语句，后面可带任何参数。ADO 将对参数类型进行“最佳判断”。例如：

```
Dim cnn As New ADODB.Connection
...
'存储过程名称及任意参数。
cnn.sp_yourStoredProcedureName "parameter"
```

1.1.4.2.3 DataControl 对象 (RDS)

RDS.DataControl 对象将数据查询 **Recordset** 绑定到一个或多个控件上（例如，文本框、网格控件或组合框），以便在 Web 页显示 **ADOR.Recordset** 数据。

语法

<OBJECT CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID="*DataControl***"**
<PARAM NAME="Connect" VALUE="DSN=*DSNName***;UID=***usr***;PWD=***pw***;">**
<PARAM NAME="Server" VALUE="http://*awebsrvr***">**
<PARAM NAME="SQL" VALUE="*QueryText***">**
</OBJECT>

说明

RDS.DataControl 对象的类 ID 是 BD96C556-65A3-11D0-983A-00C04FC29E33。
对基本应用方案，仅需设置 **RDS.DataControl** 对象的 **SQL**、**Connect** 和 **Server** 属性，从而自动调用默认业务对象 RDSServer.DataFactory(See 1.1.4.2.4)。
RDS.DataControl 中所有的属性都是可选的，因为自定义业务对象(See 2.7)可以替代它们的功能。
使用一个 **RDS.DataControl** 对象可将单个查询结果链接到一个或多个可视控件。例如，假设您编写请求客户数据查询的代码，如姓名、住所、出生地、年龄及“优先级客户状态”，可使用单个 **RDS.DataControl** 对象分别在三个文本框中显示客户姓名、年龄及居住区，并在复选框中显示“优先级客户状态”，而在网格控件中显示所有的数据。
使用不同的 **RDS.DataControl** 对象可将多个查询结果链接到不同的可视控件上。例如，假设您使用一个查询获取客户信息，又使用第二个查询获取客户所购买商品的信息。如果要在三个文本框和一个复选框中显示第一个查询的结果，在网格控件中显示第二个查询的结果，则在使用默认业务对象 (RDSServer.DataFactory) 时必须进行如下操作：

- 将两个 **RDS.DataControl** 对象添加到 Web 页。
- 针对两个 **RDS.DataControl** 对象的每个 **SQL** 属性，分别编写两个查询。一个 **RDS.DataControl** 对象包含请求客户信息的 SQL 查询，而另一个则包含请求客户所买商品列表的查询。
- 在每个绑定控件的 OBJECT 标记中，指定 DATAFLD 值以便设置将要显示在每个可视控件中的数据值。

对可通过 OBJECT 标记嵌入单个 Web 页的 **RDS.DataControl** 对象，不存在数目上的限制。
在 Web 页上定义 **RDS.DataControl** 对象时，可使用非零 **Height** 和 **Width** 值，如 1（以避免包含多余空间）。
远程数据服务客户端组件已经是 Internet Explorer 4.0 安装程序的一部分，因此， **RDS.DataControl** 对象标记不必包含 CODEBASE 参数。

已测试控件

下表列出了与 **RDS.DataControl** 对象以及关联的客户(See 2.11)端的组件一起使用并且已经过测试的数据识别控件(See 2.16)。其他控件虽然也可与远程数据服务一起使用，但尚未经过测试。

控件名	文件名	类 ID (CLSID)

SSDBGrid	SSDATB32.ocx (Sheridan)	AC05DC80-7DF1-11d0-839E-00A024A94B3A

使用 Internet Explorer 4.0，只有在 HTML 控件和 ActiveX® 控件被标记为房间模型控件时，才可以使用这两种控件绑定到数据。以上所列控件不通过远程数据服务发布，但可以作为 Microsoft® Visual Basic®, Enterprise Edition 的一部分或从 Sheridan Systems, Inc. 购买。

重要事项 不能重新分发以任何方式部署为远程数据服务范例组成部分的 ActiveX 控件。它们是作为范例应用程序的组成部分提供的，任何情况下都不可重新分发到其他方。

Microsoft Visual Basic 用户 RDS.DataControl 仅用于基于 Web 的应用程序。Visual Basic 客户端应用程序不需要它。

1.1.4.2.4 DataFactory 对象 (RDSServer)

该默认服务器端业务对象实现对客户端应用程序的指定数据源进行读/写数据访问的方法。

说明

RDSServer.DataFactory 对象被设计为接收客户端请求的服务器端自动化对象。在 Internet 的实现过程中，该对象驻留于 Web 服务器(See 2.51)并且通过 ADISAPI 组件实例化。**RDSServer.DataFactory** 对象提供对指定数据源的读写访问，但不包含任何有效性或业务规则(See 2.8)逻辑。

如果使用的方法在 **RDSServer.DataFactory** 和 RDS.DataControl(See 1.1.4.2.3) 对象中都有效，则远程数据服务在默认状态下使用 **RDS.DataControl** 版本。默认采用基本的编程应用方案，在此方案中 **RDSServer.DataFactory** 用作常规服务器端业务对象。

如果要让 Web 应用程序处理特定任务的服务器端进程，则可以用自定义业务对象(See 2.7)替代 **RDSServer.DataFactory**。

可以创建调用 **RDSServer.DataFactory** 方法的服务器端业务对象，如 **Query** 和 **CreateRecordset**。这将有助于利用现有的远程数据服务技术向业务对象添加功能

RDSServer.DataFactory 对象的类 ID(See 2.10) 为 9381D8F5-0288-11D0-9501-00AA00B911A5。

1.1.4.2.5 DataSpace 对象 (RDS)

RDS.DataSpace 对象创建客户端代理以自定义位于中间层(See 2.32)的业务对象(See 2.7)。

说明

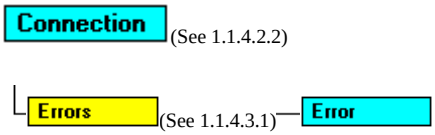
远程数据服务需要使用业务对象代理以便客户端(See 2.11)组件与位于中间层(See 2.32)的业务对象进行通讯。代理便于对跨越进程或计算机边界的应用程序记录集数据进行打包、拆包和传输（调度(See 2.29)）。

远程数据服务使用 **RDS.DataSpace** 对象的 **CreateObject** 方法创建业务对象代理。无论其相应的中间层业务对象的实例何时创建，业务对象代理(See 2.40)均被动态创建。远程数据服务支持以下协议：HTTP、HTTPS（HTTP 安全套接字）、DCOM 以及进程内（客户端组件及业务对象驻留于同一台计算机上）。

RDS.DataSpace 对象的类 ID(See 2.10) 为 BD96C556-65A3-11D0-983A-00C04FC29E36。

1.1.4.2.6 Error 对象 (ADO)

Error 对象包含与单个操作（涉及提供者）有关的数据访问错误的详细信息。



说明

任何涉及 ADO 对象的操作都会生成一个或多个提供者错误。每个错误出现时，一个或多个 **Error** 对象将被放到 Connection(See 1.1.4.2.2) 对象的 **Errors** 集合中。当另一个 ADO 操作产生错误时，**Errors** 集合将被清空，并在其中放入新的 **Error** 对象集。

注意 每个 **Error** 对象都代表特定的提供者错误而不是 ADO 错误，ADO 错误被记载到运行时的例外处理机制中。例如，在 Microsoft Visual Basic 中，产生特定 ADO 的错误将触发 **On Error** 事件并出现在 **Err** 对象中。关于 ADO 错误的完整列表，请参阅 ADO 错误代码(See 1.1.8.1)主题。

通过 **Error** 对象的属性可获得每个错误的详细信息，其中包括以下内容：

- **Description** 属性，包含错误的文本。
- **Number** 属性，包含错误常量的**长整型**整数值。
- **Source** 属性，标识产生错误的对象。在向数据源发出请求之后，如果 **Errors** 集合中有多个 **Error** 对象，则将会用到该属性。
- **SQLState** 和 **NativeError** 属性，提供来自 SQL 数据源的信息。

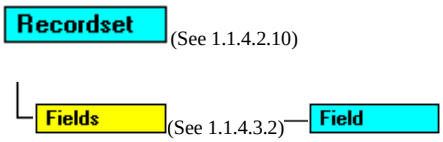
出现提供者错误时，**Error** 对象将被放在 Connection(See 1.1.4.2.2) 对象的 Errors(See 1.1.4.3.1) 集合中。ADO 支持由单个 ADO 操作返回多个错误，以便显示特定提供者的错误信息。要在错误处理程序中获得丰富的错误信息，可使用相应的语言或所在工作环境下的错误捕获功能，然后使用嵌套循环枚举出 **Errors** 集合的每个 **Error** 对象的属性。

Microsoft Visual Basic 及 VBScript 如果没有有效的 **Connection** 对象，则需要检索 **Err** 对象的错误信息。

与提供者一样，ADO 在进行可能引发新的提供者错误的调用前将清除 **OLE Error Info** 对象。但是，只有当提供者产生新的错误或 **Clear** 方法被调用时，才能清空并充填 **Connection** 对象的 **Errors** 集合。
某些属性和方法返回的警告以 **Errors** 集合中的 **Error** 对象的方式出现，但并不中止程序的执行。在调用 Recordset(See 1.1.4.2.10) 对象的 Resync(See 1.1.4.4.40)、UpdateBatch(See 1.1.4.4.46)、或 CancelBatch(See 1.1.4.4.7) 方法，或 **Connection** 对象的 Open(See 1.1.4.4.32) 方法，或者在设置 **Recordset** 对象的 Filter(See 1.1.4.6.28) 属性之前，可通过调用 **Errors** 集合的 **Clear** 方法。这样就可以读取 **Errors** 集合的 Count(See 1.1.4.6.16) 属性，以测试返回的警告。

1.1.4.2.7 Field 对象 (ADO)

Field 对象代表使用普通数据类型的数据的列。



说明

Recordset(See 1.1.4.2.10) 对象含有由 **Field** 对象组成的 Fields(See 1.1.4.3.2) 集合。每个 **Field** 对象对应于 **Recordset** 中的一列。使用 **Field** 对象的 **Value** 属性可设置或返回当前记录的数据。取决于提供者具有的不同功能，**Field** 对象的某些集合、方法或属性有可能无效。

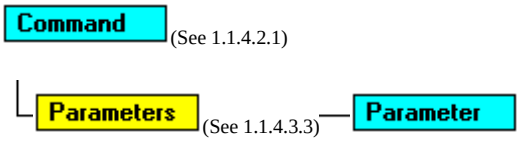
使用 **Field** 对象的集合、方法、和属性可进行如下操作：

- 使用 **Name** 属性可返回字段名。
- 使用 **Value** 属性可查看或更改字段中的数据。
- 使用 **Type**、**Precision** 和 **NumericScale** 属性可返回字段的基本特性。
- 使用 **DefinedSize** 属性可返回已声明的字段大小。
- 使用 **ActualSize** 属性可返回给定字段中数据的实际大小。
- 使用 **Attributes** 属性和 **Properties** 集合可决定对于给定字段哪些类型的功能受到支持。
- 使用 **AppendChunk** 和 **GetChunk** 方法可处理包含长二进制或长字符数据的字段值。
- 如果提供者支持批更新，可使用 **OriginalValue** 和 **UnderlyingValue** 属性在批更新期间解决字段值之间的差异。

在打开 **Field** 对象的 **Recordset** 前，所有元数据属性（**Name**、**Type**、**DefinedSize**, **Precision** 和 **NumericScale**）都是可用的。在此时设置这些属性将有助于动态构造其格式。

1.1.4.2.8 Parameter 对象 (ADO)

Parameter 对象代表与基于参数化查询或存储过程的 **Command** 对象相关联的参数或自变量。



说明

许多提供者都支持参数化的命令。需要进行的操作在这些命令中只定义一次，但可以使用变量（或参数）改变命令的某些细节。例如，SQL SELECT 语句可使用参数定义 WHERE 子句的匹配条件，而使用另一个参数来定义 SORT BY 子句的列的名称。

Parameter 对象代表与参数化查询关联的参数，或进/出参数以及存储过程的返回值。取决于提供者的功能，**Parameter** 对象的某些集合、方法或属性有可能无效。

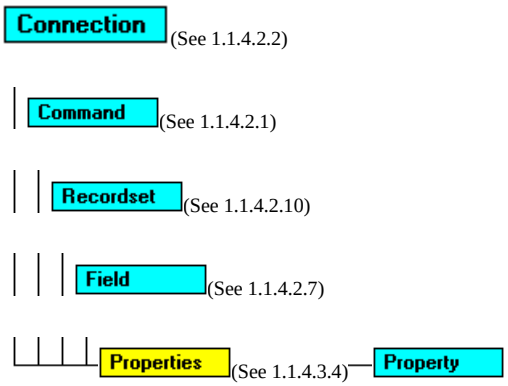
使用 **Parameter** 对象的集合、方法、和属性可进行如下操作：

- 使用 **Name** 属性可设置或返回参数名称。
- 使用 **Value** 属性可设置或返回参数值。
- 使用 **Attributes** 和 **Direction**、**Precision**、**NumericScale**、**Size** 以及 **Type** 属性可设置或返回参数特性。
- 使用 **AppendChunk** 方法可将长整型二进制或字符数据传递给参数。

如果知道与想要调用的存储过程或参数化查询相关联的参数属性和名称，则可使用 CreateParameter(See 1.1.4.4.16) 方法创建带有相应属性设置的 **Parameter** 对象，并使用 **Append** 方法将它们添加到 Parameters(See 1.1.4.3.3) 集合。这样就可以设置并返回参数值，而无需调用 **Parameters** 集合的 Refresh(See 1.1.4.4.36) 方法来检索提供者的参数信息，即潜在的、资源集中的操作。

1.1.4.2.9 Property 对象 (ADO)

Property 对象代表由提供者定义的 ADO 对象的动态特征。



说明

ADO 对象有两种类型的属性：内置属性和动态属性。

内置属性是在 ADO 中实现并立即可用于任何新对象的属性，此时使用 MyObject.Property 语法。它们不会作为 **Property** 对象出现在对象的 Properties(See 1.1.4.3.4) 集合中，因此，虽然可以更改它们的值，但无法更改它们的特性。

动态属性由基本的数据提供者定义，并出现在相应的 ADO 对象的 **Properties** 集合中。例如，指定给提供者的属性可能会指示 Recordset(See 1.1.4.2.10) 对象是否支持事务或更新。这些附加的属性将作为 **Property** 对象出现在该 **Recordset** 对象的 **Properties** 集合中。动态属性只能通过集合使用 MyObject.Properties(0) 或 MyObject.Properties("Name") 语法来引用。

两种属性都无法删除。

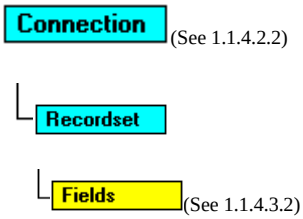
动态 **Property** 对象有四个自己的内置属性：

- **Name** 属性是标识属性的字符串。
- **Type** 属性是用于指定属性数据类型的整数。
- **Value** 属性是包含属性设置的变体型。

- **Attributes** 属性是指示特定于提供者的属性特征的长整型值。

1.1.4.2.10 Recordset 对象 (ADO)

Recordset 对象表示的是来自基本表或命令执行结果的记录全集。任何时候，**Recordset** 对象所指的当前记录均为集合内的单个记录。



说明

可使用 **Recordset** 对象操作来自提供者的数据。使用 ADO 时，通过 **Recordset** 对象可对几乎所有数据进行操作。所有 **Recordset** 对象均使用记录（行）和字段（列）进行构造。由于提供者所支持的功能不同，某些 **Recordset** 方法或属性有可能无效。

ADOR.Recordset 和 **ADODB.Recordset** 是用来创建 **Recordset** 对象的 ProgID。由此产生的 **Recordset** 对象行为相同，与 ProgID 无关。**ADOR.Recordset** 随 Microsoft® Internet Explorer 安装，而 **ADODB.Recordset** 则随 ADO 安装。**Recordset** 对象的行为受环境（即客户端、服务器、Internet Explorer 等）的影响。这些差异将在属性、方法和事件的“帮助”主题中加以说明。

在 ADO 中定义了四种不同的游标类型：

- **动态游标** — 用于查看其他用户所作的添加、更改和删除，并用于不依赖书签的 **Recordset** 中各种类型的移动。如果提供者支持，可使用书签。

- **键集游标** — 其行为类似动态游标，不同的只是禁止查看其他用户添加的记录，并禁止访问其他用户删除的记录，其他用户所作的数据更改将依然可见。它始终支持书签，因此允许 **Recordset** 中各种类型的移动。

- **静态游标** — 提供记录集合的静态副本以查找数据或生成报告。它始终支持书签，因此允许 **Recordset** 中各种类型的移动。其他用户所作的添加、更改或删除将不可见。这是打开客户端 (ADOR) **Recordset** 对象时唯一允许使用的游标类型。

- **仅向前游标** — 除仅允许在记录中向前滚动之外，其行为类似静态游标。这样，当需要在 **Recordset** 中单程移动时就可提高性能。

在打开 **Recordset** 之前设置 **CursorType** 属性来选择游标类型，或使用 Open(See 1.1.4.4.33) 方法传递 **CursorType** 参数。部分提供者不支持所有游标类型。请检查提供者的文档。如果没有指定游标类型，ADO 将默认打开仅向前游标。

如果 CursorLocation(See 1.1.4.6.17) 属性被设置为 **adUseClient** 后打开 **Recordset**，则在返回的 **Recordset** 对象中，Field(See 1.1.4.2.7) 对象的 **UnderlyingValue** 属性不可用。对部分提供者（例如 Microsoft ODBC Provider for OLE DB 连同 Microsoft SQL Server），可以通过使用 **Open** 方法传递连接字符串，根据以前定义的 Connection(See 1.1.4.2.2) 对象独立地创建 **Recordset** 对象。ADO 仍然创建 **Connection** 对象，但它不将该对象赋给对象变量。不过，如果正在相同的连接上打开多个 **Recordset** 对象，就应该显式创建和打开 **Connection** 对象，由此将 **Connection** 对象赋给对象变量。如果在打开 **Recordset** 对象时没有使用该对象变量，即使在传递相同连接字符串的情况下，ADO 也将为每个新的 **Recordset** 创建新的 **Connection** 对象。

可以创建所需数量的 **Recordset** 对象。

打开 **Recordset** 时，当前记录位于第一个记录（如果有），并且 BOF(See 1.1.4.6.7) 和 EOF(See 1.1.4.6.7) 属性被设置为 **False**。如果没有记录，**BOF** 和 **EOF** 属性设置是 **True**。

假设提供者支持相关的功能，可以使用 **MoveFirst**、**MoveLast**、**MoveNext** 和 **MovePrevious** 方法以及 **Move** 方法，和 **AbsolutePosition**、**AbsolutePage** 和 **Filter** 属性来重新确定当前记录的位置。仅向前 **Recordset** 对象只支持 **MoveNext** 方法。当使用 **Move** 方法访问每个记录（或枚举 **Recordset**）时，可使用 **BOF** 和 **EOF** 属性查看是否移动已经超过了 **Recordset** 的开始或结尾。

Recordset 对象可支持两类更新：立即更新和批更新。使用立即更新，一旦调用 **Update** 方法，对数据的所有更改将被立即写入基本数据源。也可以使用 **AddNew** 和 **Update** 方法将值的数组作为参数传递，同时更新记录的若干字段。

如果提供者支持批更新，可以使提供者将多个记录的更改存入缓存，然后使用 **UpdateBatch** 方法在单个调用中将它们传送给数据库。这种情况应用于使用 **AddNew**、**Update** 和 **Delete** 方法所做的更改。调用 **UpdateBatch** 方法后，可以使用 **Status** 属性检查任何数据冲突并加以解决。

注意 要执行不使用 Command 对象的查询，应将查询字符串传递给 Recordset 对象的 Open 方法。但是，在想要保持命令文本并重复执行或使用查询参数时，仍然需要 Command 对象。

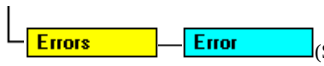
1.1.4.3 ADO 集合

集合	说明
Errors(See 1.1.4.3.1)	包含为响应涉及提供者的单个错误而创建的所有 **Error** 对象。
Fields(See 1.1.4.3.2)	包含 **Recordset** 对象的所有 **Field** 对象。
Parameters(See 1.1.4.3.3)	包含 **Command** 对象的所有 **Parameter** 对象。
Properties(See 1.1.4.3.4)	包含指定对象实例的所有 **Property** 对象。

1.1.4.3.1 Errors 集合 (ADO)

包含在响应涉及提供者的单个失败时产生的所有 **Error** 对象。

(See 1.1.4.2.2)



(See 1.1.4.2.6)

说明

任何涉及 ADO 对象的操作都可以产生一个或多个提供者错误。产生错误时，可以将一个或多个 **Error** 对象置于 Connection(See 1.1.4.2.2) 对象的 **Errors** 集合中。其他 ADO 操作产生错误时，将清空 **Errors** 集合，并且将新的 **Error** 对象置于 **Errors** 集合中。

每个 **Error** 对象代表特定的提供者错误，而不是 ADO 错误。ADO 错误被记载在运行时的异常处理机制中。例如，在 Microsoft Visual Basic 中，出现特定 ADO 的错误出现将引发 **On Error** 事件并且该错误将显示在 **Err** 对象中。

没有产生错误的 ADO 操作对 **Errors** 集合没有影响。使用 **Clear** 方式可手工清除 **Errors** 集合。

Errors 集合中的 **Error** 对象集合描述响应单个语句时产生的所有错误。列举 **Errors** 集合中指定错误可使错误处理例程更精确地确定产生错误的原因及错误来源，并执行适当还原步骤。

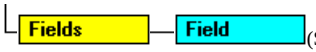
某些属性和方法将返回作为 **Errors** 集合中的 **Error** 对象显示的警告，但不会中止程序的执行。在调用 Recordset(See 1.1.4.2.10) 对象上的 Resync(See 1.1.4.4.40)、UpdateBatch(See 1.1.4.4.46) 或 CancelBatch(See 1.1.4.4.7) 方法，**Connection** 的 Open(See 1.1.4.4.32) 方法或者设置 **Recordset** 对象上的 Filter(See 1.1.4.6.28) 属性前，请调用 **Errors** 集合上的 Clear(See 1.1.4.4.10) 方法。这样您就可以阅读 **Errors** 集合的 Count(See 1.1.4.6.16) 属性以测试返回的警告。

注意 有关单个 ADO 操作可产生多个错误的方式的详细说明，请参阅 **Error** 对象主题。

1.1.4.3.2 Fields 集合 (ADO)

包含 **Recordset** 对象的所有 **Field** 对象。

(See 1.1.4.2.10)



(See 1.1.4.2.7)

说明

Recordset 对象包括 **Field** 对象组成的 **Fields** 集合。每个 **Field** 对象对应 **Recordset** 集中的一列。在通过调用集合上的 Refresh(See 1.1.4.4.36) 方法打开 **Recordset** 前可以填充 **Fields** 集合。

注意 有关如何使用 **Field** 对象的详细说明，请参阅 **Field** 对象主题。

1.1.4.3.3 Parameters 集合 (ADO)

包含 **Command** 对象的所有 **Parameter** 对象。

(See 1.1.4.2.1)

(See 1.1.4.2.8)

说明

Command 对象具有由 **Parameter** 对象组成的 **Parameters** 集合。

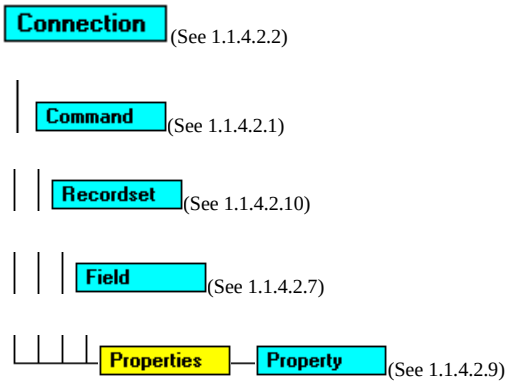
使用 **Command** 对象的 **Parameters** 集合上的 Refresh(See 1.1.4.4.36) 方法，可以获取有关 **Command** 对象中指定的存储过程或参数化查询的提供者的参数信息。某些提供者不支持存储过程进行调用或参数化查询，使用这样的提供者时调用 **Parameters** 集合上的 **Refresh** 方法将返回错误。

如果调用 **Refresh** 方法前没有定义自己的 **Parameter** 对象而访问 **Parameters** 集合，ADO 将自动调用方法并填充该集合。

如果知道与要调用的存储过程或参数化查询相关联的参数的属性，可以最小化对提供者的调用以提高性能。使用 CreateParameter(See 1.1.4.4.16) 方法可以创建具有适当属性设置的 **Parameters** 对象，使用 Append(See 1.1.4.4.2) 方法可以将其添加到 **Parameters** 集合。这将允许您设置并返回参数值而不必调用参数信息的提供者。如果正在写入不提供参数信息的提供者，则必须使用此方法手工填充 **Parameters** 集合才能使用参数。如果必要可以使用 Delete(See 1.1.4.4.18) 方法将 **Parameters** 对象从 **Parameters** 集合中删除。

1.1.4.3.4 Properties 集合 (ADO)

包含特定对象实例的所有 **Property** 对象。



说明

某些 ADO 对象包含由 **Property** 对象组成的 **Property** 集合。每个 **Property** 对象与指定给提供者的 ADO 对象的特性相对应。

注意 有关如何使用 **Property** 对象的详细说明，请参阅 Property(See 1.1.4.2.9) 对象主题。

1.1.4.4 ADO 方法

方法	说明
AddNew(See 1.1.4.4.1)	创建可更新的 **Recordset** 对象的新记录。
Append(See 1.1.4.4.2)	将对象追加到集合中。如果集合是 **Fields**，可以先创建新的 **Field** 对象然后再将其追加到集合中。
AppendChunk(See 1.1.4.4.3)	将数据追加到大型文本、二进制数据 **Field** 或 **Parameter** 对象。
BeginTrans(See 1.1.4.4.4) 、CommitTrans(See 1.1.4.4.4) 和 RollbackTrans(See 1.1.4.4.4)	按如下方式管理 **Connection** 对象中的事务进程： **BeginTrans** – 开始新事务。 **CommitTrans** – 保存任何更改并结束当前事务。它也可能启动新事务。 **RollbackTrans** – 取消当前事务中所作的任何更改并结束事务。它也可能启动新事务。
Cancel(See 1.1.4.4.5)	取消执行挂起的、异步 **Execute** 或 **Open** 方法调用。
Cancel (RDS)(See 1.1.4.4.6)	取消当前运行的异步执行或获取。
CancelBatch(See 1.1.4.4.7)	取消挂起的批更新。
CancelUpdate(See 1.1.4.4.8)	取消在调用 **Update** 方法前对当前记录或新记录所作的任何更改。
CancelUpdate (RDS)(See 1.1.4.4.9)	放弃与指定 **Recordset** 对象关联的所有挂起更改，从而恢复上一次调用 **Refresh** 方法之后的值。

Clear(See 1.1.4.4.10)	删除集合中的所有对象。
Clone(See 1.1.4.4.11)	创建与现有 **Recordset** 对象相同的复制 **Recordset** 对象。可选择指定该副本为只读。
Close(See 1.1.4.4.12)	关闭打开的对象及任何相关对象。
CompareBookmarks(See 1.1.4.4.13)	比较两个书签并返回它们相差值的说明。
ConvertToString(See 1.1.4.4.14)	将 **Recordset** 转换为代表记录集数据的 MIME 字符串。
CreateObject (RDS)(See 1.1.4.4.15)	创建目标业务对象的代理并返回指向它的指针。
CreateParameter(See 1.1.4.4.16)	使用指定属性创建新的 **Parameter** 对象。
CreateRecordset (RDS)(See 1.1.4.4.17)	创建未连接的空 **Recordset**。
Delete (ADO Parameters Collection)(See 1.1.4.4.18)	从 **Parameters** 集合中删除对象。
Delete (ADO Fields Collection)(See 1.1.4.4.19)	从 **Fields** 集合删除对象。
Delete (ADO Recordset)(See 1.1.4.4.20)	删除当前记录或记录组。
Execute (ADO Command)(See 1.1.4.4.21)	执行在 **CommandText** 属性中指定的查询、SQL 语句或存储过程。
Execute (ADO Connection)(See 1.1.4.4.22)	执行指定的查询、SQL 语句、存储过程或特定提供者的文本等内容。
Find(See 1.1.4.4.23)	搜索 **Recordset** 中满足指定标准的记录。
GetChunk(See 1.1.4.4.24)	返回大型文本或二进制数据 **Field** 对象的全部或部分内容。
GetRows(See 1.1.4.4.25)	将 **Recordset** 对象的多个记录恢复到数组中。
GetString(See 1.1.4.4.26)	将 **Recordset** 按字符串返回。
Item(See 1.1.4.4.27)	根据名称或序号返回集合的特定成员。
Move(See 1.1.4.4.28)	移动 **Recordset** 对象中当前记录的位置。
MoveFirst、MoveLast、MoveNext 和 MovePrevious(See 1.1.4.4.29)	移动到指定 **Recordset** 对象中的第一个、最后一个、下一个或前一个记录并使该记录成为当前记录。
MoveFirst、MoveLast、MoveNext、MovePrevious (RDS)(See 1.1.4.4.30)	移动到显示的 **Recordset** 中的第一个、最后一个、下一个或前一个记录。
NextRecordset(See 1.1.4.4.31)	清除当前 **Recordset** 对象并通过提前命令序列返回下一个记录集。
Open (ADO Connection)(See 1.1.4.4.32)	打开到数据源的连接。
Open (ADO Recordset)(See 1.1.4.4.33)	打开游标。
OpenSchema(See 1.1.4.4.34)	从提供者获取数据库模式信息。

Query (RDS)(See 1.1.4.4.35)	使用有效的 SQL 查询字符串返回 **Recordset**。
Refresh(See 1.1.4.4.36)	更新集合中的对象以便反映来自提供者的可用对象以及特定于提供者的对象。
Refresh (RDS)(See 1.1.4.4.37)	对在 **Connect** 属性中指定的 ODBC 数据源进行再查询并更新查询结果。
Requery(See 1.1.4.4.38)	通过重新执行对象所基于的查询，更新 **Recordset** 对象中的数据。
Reset (RDS)(See 1.1.4.4.39)	根据指定的排序和筛选属性对客户端 **Recordset** 执行排序或筛选操作。
Resync(See 1.1.4.4.40)	从基本数据库刷新当前 **Recordset** 对象中的数据。
Save (ADO Recordset)(See 1.1.4.4.41)	将 **Recordset** 保存（持久）在文件中。
Seek(See 1.1.4.4.42)	搜索 **Recordset** 的索引以便快速定位与指定值相匹配的行，并将当前行的位置更改为该行。
SubmitChanges (RDS)(See 1.1.4.4.43)	将本地缓存的可更新 **Recordset** 的挂起更改提交到在 **Connect** 属性中指定的 ODBC 数据源中。
Supports(See 1.1.4.4.44)	确定指定的 **Recordset** 对象是否支持特定类型的功能。
Update(See 1.1.4.4.45)	保存对 **Recordset** 对象的当前记录所做的所有更改。
UpdateBatch(See 1.1.4.4.46)	将所有挂起的批更新写入磁盘。

1.1.4.4.1 AddNew 方法 (ADO)

创建可更新 Recordset(See 1.1.4.2.10) 对象的新记录。

语法

recordset.**AddNew** ***FieldList, Values***

参数

FieldList 可选。新记录中字段的单个名称、一组名称或序号位置。

Values 可选。新记录中字段的单个或一组值。如果 ***Fields*** 是数组，那么 ***Values*** 也必须是有相同成员数的数组，否则将发生错误。字段名称的次序必须与每个数组中的字段值的次序相匹配。

说明

使用 **AddNew** 方法创建和初始化新记录。通过 **adAddNew** 使用 Supports(See 1.1.4.4.44) 方法可验证是否能够将记录添加到当前的 **Recordset** 对象。

在调用 **AddNew** 方法后，新记录将成为当前记录，并在调用 Update(See 1.1.4.4.45) 方法后继续保持为当前记录。如果 **Recordset** 对象不支持书签，当移动到其他记录时将无法对新记录进行访问。是否需要调用 Requery(See 1.1.4.4.38) 方法访问新记录则取决于所使用的游标类型。

如果在编辑当前记录或添加新记录时调用 **AddNew**，ADO 将调用 **Update** 方法保存任何更改并创建新记录。

AddNew 方法的行为取决于 **Recordset** 对象的更新模式以及是否传送 ***Fields*** 和 ***Values*** 参数。

在立即更新模式（调用 **Update** 方法时提供者会立即将更改写入基本数据源）下，调用不带参数的 **AddNew** 方法可将 **EditMode** 属性设置为 **adEditAdd**。提供者将任何字段值的更改缓存在本地。调用 **Update** 方法可将新记录传递到数据库并将 **EditMode** 属性重置为 **adEditNone**。如果传送了 **Fields** 和 **Values** 参数，ADO 则立即将新记录传递到数据库（无须调用 **Update**），且 **EditMode** 属性值没有改变 (**adEditNone**)。

在批更新模式(提供者缓存多个更改并只在调用 UpdateBatch(See 1.1.4.4.46) 时将其写入基本数据源)下，调用不带参数的 **AddNew** 方法可将 **EditMode** 属性设置为 **adEditAdd**。提供者将任何字段值的更改缓存在本地。调用 **Update** 方法可将新的记录添加到当前记录集并将 **EditMode** 属性重置为 **adEditNone**，但在调用 **UpdateBatch** 方法之前提供者不将更改传递到基本数据库。如果传送 ***Fields*** 和 ***Values*** 参数，ADO 则将新记录发送给提供者以便缓存；需要调用 **UpdateBatch** 方法将新记录传递到基本数据库。

如果 Unique Table(See 1.1.4.7.2) 动态属性被设置，并且 **Recordset** 是对多个表执行 JOIN 操作的结果，那么，**AddNew** 方法只能将字段插入到由 **Unique Table** 属性所命名的表中。

1.1.4.4.2 Append 方法 (ADO)

将对象追加到集合中。如果集合是 Fields(See 1.1.4.3.2)，可以先创建新的 Field(See 1.1.4.2.7) 对象然后再将其追加到集合中。

语法

collection.**Append** ***object***

fields.**Append** ***Name, Type, DefinedSize, Attrib***

参数

collection 集合对象。

fields **Fields** 集合。

object 对象变量，代表所要追加对象。

Name **字符串**，新 **Field** 对象的名称，不得与 *fields* 中的任何其他对象同名。

Type **DataTypeEnum** 类型，其默认值为 **adEmpty**。新字段的数据类型。

DefinedSize 可选，**长整型**，指示新字段的定义大小(以字符或字节为单位)。该参数的默认值源于 ***Type***（默认的 ***Type*** 为 **adEmpty**，默认的 **DefinedSize** 未指定）。

Attrib 可选，**FieldAttributeEnum**，其默认值是 **adFldDefault**。指定新字段的属性。如果该值未指定，字段将包含源于 ***Type*** 的属性。

参数

在集合上使用 **Append** 方法可将对象添加到该集合，此方法仅在 Command(See 1.1.4.2.1) 对象的 Parameters(See 1.1.4.3.3) 集合上有效。在将 Parameter(See 1.1.4.2.8) 对象追加到 **Parameters** 集合中之前必须设置其 Type(See 1.1.4.6.67) 属性。如果选定了变长数据类型，则必须将 Size(See 1.1.4.6.56) 属性设置为大于零的值。

通过对参数作出说明，可以最大程度地减少对提供者的调用，进而在使用存储过程或参数化查询时提高性能，但必须了解与所要调用的存储过程或参数化查询相关联的参数属性。使用 CreateParameter(See 1.1.4.4.16) 方法可创建具有适当属性设置的 **Parameter** 对象，而使用 **Append** 方法则可将它们添加到 **Parameters** 集合。这样可以不必调用参数信息的提供者而设置和返回参数值。如果写到不提供参数信息的提供者，则必须使用该方法手工填写 **Parameters** 集合以便能够完全使用参数。

字段

如果在调用 *fields*.**Append** 方法前您没有设置 **CursorLocation** 属性，当使用 **Open** 方法打开 **Recordset** 时，**CursorLocation** 将被自动设置为 **adUseClient**。

对打开的 **Recordset** 或已设置 **ActiveConnection** 属性的 **Recordset**，调用其 *fields*.**Append** 方法将引发运行时错误。只能将字段追加到没有打开并且尚未连接到数据源的 **Recordset**。一般地，通过 **CreateRecordset** 方法或通过将新 **Recordset** 对象显式赋给对象变量所创建的都是新 **Recordset** 对象。

1.1.4.4.3 AppendChunk 方法 (ADO)

将数据追加到大型文本、二进制数据 Field(See 1.1.4.2.7) 或 Parameter(See 1.1.4.2.8) 对象。

语法

object.**AppendChunk** ***Data***

参数

object **Field** 或 **Parameter** 对象

Data **变体型**，包含追加到对象中的数据。

说明

使用 **Field** 或 **Parameter** 对象的 **AppendChunk** 方法可将长二进制或字符数据填写到对象中。在系统内存有限的情况下，可以使用 **AppendChunk** 方法对长整型值进行部分而非全部的操作。

字段

如果 **Field** 对象 Attributes(See 1.1.4.6.6) 属性中的 **adFldLong** 位被设置为真，则可以对该字段使用 **AppendChunk** 方法。

在 **Field** 对象上的第一个 **AppendChunk** 调用将数据写入字段，覆盖任何现有的数据，随后的 **AppendChunk** 调用则添加到现有数据。如果将数据追加到一个字段，然后设置或读取当前记录中另一个字段的值，ADO 则认为已将数据追加到第一个字段。如果在第一个字段上再次调用 **AppendChunk** 方法，那么 ADO 将调用解释为新的 **AppendChunk** 操作并覆盖现有数据。访问其他 Recordset(See 1.1.4.2.10) 对象（并非第一个 **Recordset** 对象的复制品）中的字段将不会破坏 **AppendChunk** 操作。

调用 **Field** 对象的 **AppendChunk** 时如果没有当前记录，将发生错误。

参数

如果 **Parameter** 对象 Attributes(See 1.1.4.6.6) 属性中的 **adFldLong** 位被设置为真，则可以对该参数使用 **AppendChunk** 方法。**Parameter** 对象上的第一个 **AppendChunk** 调用将数据写入参数，覆盖任何现有数据，随后 **Parameter** 对象上的 **AppendChunk** 调用可添加到现有的参数数据中。传送空值的 **AppendChunk** 调用则放弃所有的参数数据。

1.1.4.4.4 BeginTrans、CommitTrans 和 RollbackTrans 方法 (ADO)

这些事务方法按如下方式管理 Connection(See 1.1.4.2.2) 对象中的事务进程：

- **BeginTrans** - 开始新事务。
- **CommitTrans** - 保存任何更改并结束当前事务。它也可能启动新事务。
- **RollbackTrans** - 取消当前事务中所作的任何更改并结束事务。它也可能启动新事务。

语法

level = *object*.**BeginTrans()**
object.**BeginTrans**
object.**CommitTrans**
object.**RollbackTrans**

返回值

BeginTrans 可以作为函数调用，用于返回指示事务嵌套层次的长整型变量。

参数

object **Connection** 对象。

Connection

如果希望以独立单元保存或取消对源数据所做的一系列更改，请使用这些具有 **Connection** 对象的方法。例如在货币转帐时，必须从帐户中减去某个数额并将其对等数额添加到另一个帐户。无论其中的哪个更新失败，都将导致帐户收支不平衡。在打开的事务中进行这些更改可确保只能选择进行全部更改或不作任何更改。

> **注意** 并非所有提供者都支持事务。需验证提供者定义的属性“Transaction DDL”是否出现在 **Connection** 对象的 **Properties** 集合中，如果在则表示提供者支持事务。如果提供者不支持事务，调用其中的某个方法将返回错误。

一旦调用了 **BeginTrans** 方法，在调用 **CommitTrans** 或 **RollbackTrans** 结束事务之前，提供者将不再立即提交所作的任何更改。对于支持嵌套事务的提供者来说，调用已打开事务中的 **BeginTrans** 方法将开始新的嵌套事务。返回值将指示嵌套层次：返回值为 1 表示已打开顶层事务（即事务不被另一个事务所嵌套），返回值为 2 表示已打开第二层事务（嵌套在顶层事务中的事务），依次类推。调用 **CommitTrans** 或 **RollbackTrans** 只影响最新打开的事务；在处理任何更高层事务之前必须关闭或回卷当前事务。
调用 **CommitTrans** 方法将保存连接上打开的事务中所做的更改并结束事务。调用 **RollbackTrans** 方法还原打开事务中所做的更改并结束事务。在未打开事务时调用其中任何一种方法都将引发错误。
取决于 **Connection** 对象的 Attributes(See 1.1.4.6.6) 属性，调用 **CommitTrans** 或 **RollbackTrans** 方法都可以自动启动新事务。如果 **Attributes** 属性设置为 **adXactCommitRetaining**，提供者在 **CommitTrans** 调用后会自动启动新事务。如果 **Attributes** 属性设置为 **adXactAbortRetaining**，提供者在调用 **RollbackTrans** 之后将自动启动新事务。

远程数据服务

BeginTrans、**CommitTrans** 和 **RollbackTrans** 方法在客户端 **Connection** 对象上无效。

1.1.4.4.5 Cancel 方法 (ADO)

取消执行挂起的异步 **Execute** 或 **Open** 方法的调用。

语法

object.**Cancel**

说明

使用 **Cancel** 方法终止执行异步 **Execute** 或 **Open** 方法调用（即通过 **adAsyncConnect**、**adAsyncExecute** 或 **adAsyncFetch** 选项调用该方法）。如果在试图终止的方法中没有使用 **adAsyncExecute**，则 **Cancel** 将返回运行时错误。

下表显示在特定类型对象上使用 **Cancel** 方法时将终止的任务。

如果对象为	将终止对该方法的上一次异步调用
Command	Execute(See 1.1.4.4.21)
Connection	Execute(See 1.1.4.4.22) 或 Open(See 1.1.4.4.32)
Recordset	Open(See 1.1.4.4.33)

1.1.4.4.6 Cancel 方法 (RDS)

取消当前运行的异步执行或获取。

语法

RDS.DataControl.**Cancel**

说明

调用 **Cancel** 时，ReadyState(See 1.1.4.6.54) 将自动设置为 **adcReadyStateLoaded**，并且 **Recordset** 将为空。

1.1.4.4.7 CancelBatch 方法 (ADO)

取消挂起的批更新。

语法

recordset.**CancelBatch** ***AffectRecords***

参数

AffectRecords 可选的 **AffectEnum** 值，决定 **CancelBatch** 方法所影响记录的数目，可为下列常量之一：

常量	说明
AdAffectCurrent	仅取消当前记录的挂起更新。
AdAffectGroup	对满足当前 Filter(See 1.1.4.6.28) 属性设置的记录取消挂起更新。使用该选项时，必须将 **Filter** 属性设置为合法的预定义常量之一。
AdAffectAll	默认值。取消 **Recordset** 对象中所有记录的挂起更新，包括由当前 **Filter** 属性设置所隐藏的任何记录。

说明

使用 **CancelBatch** 方法可取消批更新模式下记录集中所有挂起的更新。如果记录集处于立即更新模式，调用不带 **adAffectCurrent** 的 **CancelBatch** 将产生错误。

如果调用 **CancelBatch** 时正在编辑当前记录或添加新记录，则 ADO 首先调用 CancelUpdate(See 1.1.4.4.8) 方法取消所有已被缓存的修改，然后取消记录集中挂起的所有更改。

有可能在 **CancelBatch** 调用后，特别是在添加新记录时无法确定当前记录。为此，在 **CancelBatch** 调用后将当前记录位置设置为记录集中的已知位置是明智的。例如可调用 MoveFirst(See 1.1.4.4.29) 方法。

如果由于与基本数据冲突（如记录已被其他用户删除）而导致取消挂起更新失败，则提供者将向 Errors(See 1.1.4.3.1) 集合返回警告，但不终止程序的执行。只有当所有请求的记录都发生冲突时才发生运行时错误。使用 **Filter** 属性 (**adFilterAffectedRecords**) 和 Status(See 1.1.4.6.65) 属性可以对冲突记录进行定位。

1.1.4.4.8 CancelUpdate 方法 (ADO)

取消在调用 Update(See 1.1.4.4.45) 方法前对当前记录或新记录所作的任何更改。

语法

recordset.**CancelUpdate**

说明

使用 **CancelUpdate** 方法可取消对当前记录所作的任何更改或放弃新添加的记录。在调用 **Update** 方法后将无法撤消对当前记录或新记录所做的更改，除非更改是可以用 RollbackTrans(See 1.1.4.4.4) 方法回卷的事务的一部分，或者是可以用 CancelBatch(See 1.1.4.4.7) 方法取消的批更新的一部分。

如果在调用 **CancelUpdate** 方法时添加新记录，则调用 AddNew(See 1.1.4.4.1) 之前的当前记录将再次成为当前记录。

如果尚未更改当前记录或添加新记录，调用 **CancelUpdate** 方法将产生错误。

1.1.4.4.9 CancelUpdate 方法 (RDS)

放弃与指定 **Recordset** 对象关联的所有挂起更改，从而恢复上一次调用 Refresh(See 1.1.4.4.37) 方法之后的值。

语法

DataControl.**CancelUpdate**

参数

DataControl 对象变量(See 2.34)，代表 **RDS.DataControl** 对象。

说明

“OLE DB 的游标服务”将原值副本以及更改缓存都加以保留。当调用 **CancelUpdate** 时，更改缓存被重置为空，并使用原始数据对绑定控件(See 2.16)进行刷新。

1.1.4.4.10 Clear 方法 (ADO)

删除集合中的所有对象。

语法

Errors.Clear

说明

对 **Errors** 集合使用 **Clear** 方法，以删除集合中全部现有 **Error** 对象。发生错误时，ADO 将自动清空 **Errors** 集合，并用基于新错误的 **Error** 对象填充集合。

某些属性和方法返回的警告显示为 **Errors** 集合中的 **Error** 对象，但并不终止程序的执行。在调用 Recordset(See 1.1.4.2.10) 对象的 Resync(See 1.1.4.4.40)、UpdateBatch(See 1.1.4.4.46) 或 CancelBatch(See 1.1.4.4.7) 方法；Connection(See 1.1.4.2.2) 对象上的 Open(See 1.1.4.4.32) 方法；或设置 **Recordset** 对象上的 Filter(See 1.1.4.6.28) 属性之前，调用 **Errors** 集合上的 **Clear** 方法。这样，就可以读取 **Errors** 集合的 Count(See 1.1.4.6.16) 属性，测试所返回的警告。

1.1.4.4.11 Clone 方法 (ADO)

创建与现有 **Recordset** 对象相同的复制 Recordset(See 1.1.4.2.10) 对象。可选择指定该副本为只读。

语法

Set *rstDuplicate* = *rstOriginal*.**Clone (*LockType*)**

返回值

返回 **Recordset** 对象引用。

参数

rstDuplicate 对象变量，标识正在创建的复制 **Recordset** 对象。

rstOriginal 对象变量，标识要被复制的 **Recordset** 对象。

LockType 可选，**LockTypeEnum** 值，指定原始 **Recordset** 的锁定类型或只读 **Recordset**。

常量	说明
AdLockUnspecified	默认值。使用与原始类型相同的锁定类型创建副本。
AdLockReadOnly	副本创建为只读。

说明

使用 **Clone** 方法可创建多个 **Recordset** 对象副本，这对于希望在给定的记录组中保留多个当前记录十分有用。使用 **Clone** 方法比使用与初始定义相同的定义创建和打开新 **Recordset** 对象要有效得多。

新创建副本的当前记录将设置为首记录。

无论游标类型如何，对某个 Recordset 对象所作的修改在其所有副本中都是可见的。不过一旦在原始 **Recordset** 上执行了 Requery(See 1.1.4.4.38)，副本将不再与原始 **Recordset** 同步。

关闭原始 **Recordset** 时并不关闭它的副本，而关闭某个副本也将不关闭原始 **Recordset** 或任何其他副本。

用户只允许复制支持书签的 **Recordset** 对象。书签值是可交换的，也就是说，来自一个 **Recordset** 对象的书签引用可引用其任何副本中的相同记录。

1.1.4.4.12 Close 方法 (ADO)

关闭打开的对象及任何相关对象。

语法

object.**Close**

说明

使用 **Close** 方法可关闭 **Connection** 对象或 **Recordset** 对象以便释放所有关联的系统资源。关闭对象并非将它从内存中删除，可以更改它的属性设置并且在此后再次打开。要将对象从内存中完全删除，可将对象变量设置为 *Nothing*。

Connection

使用 **Close** 方法关闭 **Connection** 对象的同时，也将关闭与连接相关联的任何活动 **Recordset** 对象。与正在关闭的 **Connection** 对象相关联的 Command(See 1.1.4.2.1) 对象将被持久保留，但不再与 **Connection** 对象关联，即它的 ActiveConnection(See 1.1.4.6.4) 属性将被设置为 **Nothing**，同时，**Command** 对象的 Parameters(See 1.1.4.3.3) 集合将清除任何提供者定义的参数。

可以随后调用 Open(See 1.1.4.4.32) 方法重新建立与相同数据源或其他数据源的连接，关闭 **Connection** 对象后，调用任何需要打开与对数据源连接的方法都将产生错误。

当连接上有打开的 **Recordset** 对象时，关闭 **Connection** 对象将回卷所有 **Recordset** 对象的挂起更改。在事务进行过程中显式关闭 **Connection** 对象（调用 **Close** 方法）将产生错误。如果在事务进行过程中 **Connection** 对象超出范围，那么 ADO 将自动回卷事务。

Recordset

使用 **Close** 方法关闭 **Recordset** 对象的同时，将释放关联的数据和可能已经通过该特定 **Recordset** 对象对数据进行的独立访问。随后可调用 Open(See 1.1.4.4.33) 方法重新打开具有相同属性或已修改属性的 **Recordset**。在 **Recordset** 对象关闭后，调用任何需要活动游标的方法将产生错误。

如果正在立即更新模式下进行编辑，调用 **Close** 方法将产生错误，应首先调用 Update(See 1.1.4.4.45) 或 CancelUpdate(See 1.1.4.4.8) 方法。如果在批更新期间关闭 **Recordset** 对象，则自上次 UpdateBatch(See 1.1.4.4.46) 调用以来所做的修改将全部丢失。

如果使用 Clone(See 1.1.4.4.11) 方法创建已打开的 **Recordset** 对象的副本，关闭原始 **Recordset** 或其副本将不影响任何其他副本。

1.1.4.4.13 CompareBookmarks 方法 (ADO)

比较两个书签并返回它们相差值的说明。

语法

result = *recordset*.**CompareBookmarks(*Bookmark1*, *Bookmark2*) As CompareEnum**

返回值

返回的值指示书签所代表的两个记录的相对行位置。可返回下列值。

常量	说明
AdCompareLessThan	第一个书签在第二个之前。
AdCompareEqual	书签相同。
AdCompareGreaterThan	第一个书签在第二个之后。
AdCompareNotEqual	书签不相同而且未排序。
AdCompareNotComparable	无法比较书签。

参数

Bookmark1 第一行的书签。

Bookmark2 第二行的书签。

说明

书签必须应用于相同的 **Recordset** 对象，或者是 **Recordset** 对象及其 clone(See 1.1.4.4.11)。无法可靠地比较来自不同 **Recordset** 对象的书签，即使它们的创建来源及创建所使用的命令相同。同时，对于其基本提供者不支持比较的 **Recordset** 对象，也无法比

较书签。

书签唯一标识 **Recordset** 对象中的行。使用当前行的 Bookmark(See 1.1.4.6.8) 属性可以获得它的书签。

无效或形式错误的书签将引发错误。

1.1.4.4.14 ConvertToString 方法 (RDS)

将 **Recordset** 转换为代表记录集数据的 MIME(See 2.33) 字符串。

语法

DataFactory.**ConvertToString**(*Recordset*)

参数

DataFactory 对象变量(See 2.34)，代表 **RDSServer.DataFactory** 对象。

Recordset 对象变量，代表 **Recordset** 对象。

说明

通过 .asp 文件，使用 **ConvertToString** 可将 **Recordset** 嵌入到服务器所生成的 HTML 页面以便将它传输到客户计算机。**ConvertToString** 首先将 **Recordset** 加载到“游标服务”表，然后生成 MIME 格式的流。

在客户端，Remote Data Service 可以将 MIME 字符串转换回完全有效的 **Recordset**，它能很好地处理小于 400 行，每行不超过 1024 字节的数据，但它无法用于 BLOB 数据流和 HTTP 上的大型结果集。由于无法对字符串执行线压缩，因此与 Remote Data Service 所定义并用于其本地传输协议格式的线优化格式相比较，庞大的数据集在 HTTP 传输上将花费可观的时间。

> **注意** 如果使用 Active Server Pages 将 MIME 字符串的结果嵌入客户 HTML 页面，请注意早于 2.0 版本的 VBScript 将限制串的大小为 32K，一旦超过该限制，将返回“超出字符串空间”错误。通过 .asp 文件使用 MIME 嵌入对象时请使用较小的查询范围。如要解决此问题，请从 http://www.microsoft.com/vbscript 下载最新版本的 VBScript。

1.1.4.4.15 CreateObject 方法 (RDS)

创建目标业务对象的代理并返回指向它的指针。代理(See 2.40)将数据打包并调度(See 2.29)到服务器端的通讯模块(See 2.47)，以便与业务对象通讯来通过 Internet 发送请求和数据。对于进程内的组件对象则不使用代理，而只提供指向对象的指针。

语法

远程数据服务支持下列协议：HTTP、HTTPS（通过安全套接字层的 HTTP)、DCOM(See 2.19) 和进程内。

协议	语法
HTTP	**Set** *object* = *DataSpace*.**CreateObject**("*ProgId*", "**http://***awebsrvr*")
HTTPS	**Set** *object* = *DataSpace*.**CreateObject**("*ProgId*", "**https://***awebsrvr*")
DCOM	**Set** *object* = *DataSpace*.**CreateObject**("*ProgId*", "*machinename*")

进程内	**Set** *object* = *DataSpace*.**CreateObject("***ProgId***", "")**

参数

Object 对象变量(See 2.34)，用于计算在 *ProgID* 中指定类型的对象。

DataSpace 对象变量，代表用于创建新对象实例的 **RDS.DataSpace** 对象。

ProgID **字符串**，编程 ID， 用于标识实现应用程序业务规则(See 2.8)的服务器端业务对象(See 2.7)。

awebsrvr 或 *machinename* **字符串**，代表标识 Internet Information Server (IIS) Web 服务器的 URL，服务器业务对象实例在该服务器上创建。

说明

HTTP 协议是标准的 Web 协议；*HTTPS* 是安全 Web 协议。在运行无 HTTP 的局域网时可使用 *DCOM* 协议。**进程内**协议是本地的动态链接库 (DLL)，不使用网络。

1.1.4.4.16 CreateParameter 方法 (ADO)

使用指定属性创建新的 Parameter(See 1.1.4.2.8) 对象。

语法

Set *parameter* = *command*.**CreateParameter (*Name*, *Type*, *Direction*, *Size*, *Value*)**

返回值

返回 **Parameter** 对象。

参数

Name 可选，**字符串**，代表 **Parameter** 对象名称。

Type 可选，**长整型**值，指定 **Parameter** 对象数据类型。关于有效设置请参见 Type(See 1.1.4.6.67) 属性。

Direction 可选，**长整型**值，指定 **Parameter** 对象类型。关于有效设置请参见 Direction(See 1.1.4.6.24) 属性。

Size 可选，**长整型**值，指定参数值最大长度（以字符或字节数为单位）。

Value 可选，**变体型**，指定 **Parameter** 对象的值。

说明

使用 **CreateParameter** 方法可用指定的名称、类型、方向、大小和值创建新的 **Parameter** 对象。在参数中传送的所有值都将写入相应的 **Parameter** 属性。

该方法无法自动将 **Parameter** 对象追加到 **Command** 对象的 **Parameter** 集合，这样就可以设置附加属性。如果将 **Parameter** 对象追加到集合，则 ADO 将使该附加属性的值生效。

如果在 ***Type*** 参数中指定可变长度的数据类型，那么在将它追加到 **Parameters** 集合之前必须传送 ***Size*** 参数或者设置 **Parameter** 对象的 Size(See 1.1.4.6.56) 属性；否则将产生错误。

1.1.4.4.17 CreateRecordset 方法 (RDS)

创建未连接的空记录集。

语法

object.**CreateRecordset(***ColumnInfos***)**

参数

Object 对象变量(See 2.34)，代表 **RDSServer.DataFactory** 或 RDS.DataControl(See 1.1.4.2.3) 对象。

ColumnsInfos 数组的**变体型**数组，用于定义所创建的 **Recordset** 的每列。每列的定义都包含具有以下四个所需属性的数组。

属性	说明
Name	列标头的名称。
Type	整型数据类型。
Size	以字符为单位的整型宽度，与数据类型无关。
Nullability	布尔值。

随后列数组的集合被组合为一个数组，用以定义 **Recordset**。

说明

服务器端业务对象可以使用来自非 OLE DB 数据提供者的数据充填所产生的 **ADODB.Recordset**，例如包含股票份额的操作系统文件。

下表列出了 **RDSServer.DataFactory** 对象的 **CreateRecordset** 方法支持的数据类型，所列编号为用于定义字段的引用编号。

每种数据类型可以是固定长度或可变长度。固定长度类型的大小应定义为 -1，因为其大小已预先确定而此处仍需要其大小的定义。

可变长度数据类型大小的允许范围从 1 到 32767。

对于某些可变数据类型，其类型可以强制为在“替换”列中注明的类型。只有在创建和填写 **Recordset** 之后才能看到替换情况，此后如有必要，可以检查实际数据类型。

长度	常量	编号	替换
固定	**adTinyInt**	16	
固定	**adSmallInt**	2	
固定	**adInteger**	3	
固定	**adBigInt**	20	
固定	**adUnsignedTinyInt**	17	
固定	**adUnsignedSmallInt**	18	

固定	**adUnsignedInt**	19	
固定	**adUnsignedBigInt**	21	
固定	**adSingle**	4	
固定	**adDouble**	5	
固定	**adCurrency**	6	
固定	**adDecimal**	14	
固定	**adNumeric**	131	
固定	**adBoolean**	11	
固定	**adError**	10	
固定	**adGuid**	72	
固定	**adDate**	7	
固定	**adDBDate**	133	
固定	**adDBTime**	134	
固定	**adDBTimestamp**	135	7
可变	**adBSTR**	8	130
可变	**adChar**	129	200
可变	**adVarChar**	200	
可变	**adLongVarChar**	201	200
可变	**adWChar**	130	
可变	**adVarWChar**	202	130
可变	**adLongVarWChar**	203	130
可变	**adBinary**	128	
可变	**adVarBinary**	204	
可变	**adLongVarBinary**	205	204

1.1.4.4.18 Delete 方法（ADO Parameters 集合）

从 **Parameters** 集合中删除对象。

语法

Parameters.Delete ***Index***

参数

Index **字符串**，代表将要删除对象名称，或者对象在集合中的顺序位置（索引）。

说明

使用集合上的 **Delete** 方法可删除集合中的某个对象。该方法只对 Command(See 1.1.4.2.1) 对象的 Parameters(See 1.1.4.3.3) 集合有效。在调用 **Delete** 方法时必须使用 Parameter(See 1.1.4.2.8) 对象的 Name(See 1.1.4.6.41) 属性或它的集合索引 — 对象变量是无效参数。

1.1.4.4.19 Delete 方法（ADO Fields 集合）

从 **Fields** 集合中删除对象。

语法

Fields.Delete ***Field***

参数

Field **变体型**，指定将要删除的 **Field** 对象。该参数必须是 **Field** 对象名而不能是序号位置或 **Field** 对象本身。

说明

对打开的 **Recordset** 调用 **Fields.Delete** 方法将引起运行时错误。

1.1.4.4.20 Delete 方法（ADO Recordset）

删除当前记录或记录组。

语法

recordset.**Delete** ***AffectRecords***

参数

AffectRecords **AffectEnum** 值，确定 **Delete** 方法所影响的记录数目，该值可以是下列常量之一。

常量	说明
AdAffectCurrent	默认。仅删除当前记录。

AdAffectGroup	删除满足当前 Filter(See 1.1.4.6.28) 属性设置的记录。要使用该选项，必须将 **Filter** 属性设置为有效的预定义常量之一。
adAffectAll	删除所有记录。
adAffectAllChapters	删除所有子集记录。

说明

使用 **Delete** 方法可将 **Recordset** 对象中的当前记录或一组记录标记为删除。如果 **Recordset** 对象不允许删除记录将引发错误。

使用立即更新模式将在数据库中进行立即删除，否则记录将标记为从缓存删除，实际的删除将在调用 UpdateBatch(See 1.1.4.4.46) 方法时进行。（使用 **Filter** 属性可查看已删除的记录）。

从已删除的记录中检索字段值将引发错误。删除当前记录后，在移动到其他记录之前已删除的记录将保持为当前记录。一旦离开已删除记录，则无法再次访问它。

如果在事务中嵌套删除，可用 RollbackTrans(See 1.1.4.4.4) 方法恢复已删除的记录。如果处于批更新模式，则可用 CancelBatch(See 1.1.4.4.7) 方法取消一个或一组挂起删除。

如果因与基本数据冲突而导致删除记录失败（如记录已被其他用户删除），则提供者向 Errors(See 1.1.4.3.1) 集合返回警告但不终止程序执行，只有在所有提出请求的记录上发生冲突时才会产生运行时错误。

如果 Unique Table(See 1.1.4.7.2) 动态属性被设置，并且 **Recordset** 是对多个表执行 JOIN 操作的结果，那么，**Delete** 方法将仅删除 **Unique Table** 属性所命名的表中的行。

1.1.4.4.21 Execute 方法 (ADO Command)

执行在 CommandText(See 1.1.4.6.10) 属性中指定的查询、SQL 语句或存储过程。

语法

对于按行返回的 **Command**：

Set *recordset* = *command*.**Execute(*RecordsAffected*, *Parameters*, *Options*)**

对于非按行返回的 **Command**：

command.**Execute** ***RecordsAffected***, ***Parameters***, ***Options***

返回值

返回 **Recordset** 对象引用。

参数

RecordsAffected 可选，**长整型**变量，提供者向其返回操作所影响的记录数目。***RecordsAffected*** 参数仅应用于操作查询或存储过程。***RecordsAffected*** 不返回由返回结果的查询或存储过程所返回的记录数目。详细信息，请使用 RecordCount(See 1.1.4.6.52) 属性。

Parameters 可选，**变体型**数组，使用 SQL 语句传送的参数值。（用该参数传送时输出参数将不返回正确值。）

Options 可选，**长整型**值，指示提供者如何计算 **Command** 对象的 CommandText(See 1.1.4.6.10) 属性。该值可为下列常量之一：

常量	**说明**

AdCmdText	指示提供者应按命令的文本定义（如 SQL 语句）来计算 **CommandText**。
AdCmdTable	指示 ADO 应生成 SQL 查询以便从 **CommandText** 命名的表中返回所有行。
AdCmdTableDirect	指示提供者应从 **CommandText** 命名的表中返回所有行。
AdCmdStoredProc	指示提供者应按存储过程计算 **CommandText**。
AdCmdUnknown	指示 **CommandText** 中的命令类型未知。
adAsyncExecute	指示命令应异步执行。
adAsyncFetch	指示对由 **CacheSize** 属性指定的初始数量之后的剩余行应使用异步提取。

本列表中前 4 个常量的详细说明请参见 CommandType(See 1.1.4.6.12) 属性。

说明

使用 **Command** 对象的 **Execute** 方法可执行在对象的 **CommandText** 属性中指定的查询。如果 **CommandText** 属性指定按行返回查询，执行所产生的任何结果都将存储在新的 **Recordset** 对象中。如果该命令不是按行返回查询，则提供者返回关闭的 **Recordset** 对象。某些应用程序语言允许忽略该返回值（如果不需要任何 **Recordset**）。

如果查询带有参数，将使用 **Command** 对象中参数的当前值，除非通过 **Execute** 调用传送的参数覆盖它们。可以在调用 **Execute** 方法时通过省略某些参数的新值来覆盖参数子集。指定参数的次序与其在方法中被传送的次序相同。例如，如果有 4 个（或更多）参数并且希望只为第一个和第四个参数传送新值，则可以将 Array(var1,,,var4) 作为 **Parameters** 参数传送。

注意 在 Parameters 参数中传送时输出参数将不返回正确的值。

该操作结束后将发出 **ExecuteComplete** 事件。

1.1.4.4.22 Execute 方法 (ADO Connection)

执行指定的查询、SQL 语句、存储过程或特定提供者的文本等内容。

语法

对于非按行返回的命令字符串：

connection.**Execute** ***CommandText***, ***RecordsAffected***, ***Options***

对于按行返回的命令字符串：

Set *recordset* = *connection*.**Execute (*CommandText*, *RecordsAffected*, *Options*)**

返回值

返回 **Recordset** 对象引用。

参数

CommandText **字符串**，包含要执行的 SQL 语句、表名、存储过程或特定提供者的文本。

RecordsAffected 可选，**长整型**变量，提供者向其返回操作所影响的记录数目。

Options 可选，**长整型**值，指示提供者应如何计算 **CommandText** 参数，可为下列值：

常量	说明
AdCmdText	指示提供者应按命令的文本定义计算 **CommandText**。
AdCmdTable	指示 ADO 应生成 SQL 查询以便从 **CommandText** 命名的表中返回所有行。
AdCmdTableDirect	指示提供者应从 **CommandText** 命名的表中返回所有行。
AdCmdTable	指示提供者应按表名计算 **CommandText**。
AdCmdStoredProc	指示提供者应按存储过程计算 **CommandText**。
AdCmdUnknown	指示 **CommandText** 参数中的命令类型未知。
adAsyncExecute	指示命令应该异步执行。
adAsyncFetch	指示对在 **CacheSize** 属性指定的初始数量之后的剩余行使用异步提取。

本列表中前 4 个常量的详细说明请参见 CommandType(See 1.1.4.6.12) 属性。

说明

使用 **Connection** 对象的 **Execute** 方法，可执行任何在指定连接的 **CommandText** 参数中传送给方法的查询。如果 **CommandText** 参数指定按行返回的查询，执行产生的任何结果将存储在新的 **Recordset** 对象中。如果命令不是按行返回的查询，则提供者返回关闭的 **Recordset** 对象。

返回的 **Recordset** 对象始终为只读、仅向前的游标。如需要具有更多功能的 **Recordset** 对象，应首先创建具有所需属性设置的 **Recordset** 对象，然后使用 **Recordset** 对象的 **Open** 方法执行查询并返回所需游标类型。

CommandText 参数的内容对提供者是特定的，并可以是标准的 SQL 语法或提供者支持的任何特殊命令格式。

该操作完成后将产生 ExecuteComplete(See 1.1.4.5.4) 事件。

1.1.4.4.23 Find 方法 (ADO)

搜索 **Recordset** 中满足指定标准的记录。如果满足标准，则记录集位置设置在找到的记录上，否则位置将设置在记录集的末尾。

语法

Find ***(criteria, SkipRows, searchDirection, start)***

参数

criteria **字符串**，包含指定用于搜索的列名、比较操作符和值的语句。

SkipRows 可选，**长整型**值，其默认值为零，它指定当前行或 **start** 书签的位移以开始搜索。

searchDirection 可选的 **SearchDirectionEnum** 值，指定搜索应从当前行还是下一个有效行开始。其值可为 **adSearchForward** 或 **adSearchBackward**。搜索是在记录集的开始还是末尾结束由 **searchDirection** 值决定。

start 可选，**变体型**书签，用作搜索的开始位置。

说明

criteria 中的“比较操作符”可以是“>”（大于）、“<”（小于）、“=”（等于）、“>=”（大于或等于）、“<=”（小于或等于）、“<>”（不等于）或“like”（模式匹配）。

criteria 中的值可以是字符串、浮点数或者日期。字符串值以单引号分界（如“state = 'WA'”）。日期值以“#”（数字记号）分界（如“start_date > #7/22/97#”）。

如“比较操作符”为“like”，则字符串“值”可以包含“*”（某字符可出现一次或多次）或者“_”（某字符只出现一次）。（如“state like M_*”与 Maine 和 Massachusetts 匹配。）。

1.1.4.4.24 GetChunk 方法 (ADO)

返回大型文本或二进制数据 Field(See 1.1.4.2.7) 对象的全部或部分内容 。

语法

variable = *field*.**GetChunk(*Size*)**

返回值

返回**变体型**。

参数

Size **长整型**表达式，等于所要检索的字节或字符数。

说明

使用 **Field** 对象的 **GetChunk** 方法检索其部分或全部长二进制或字符数据。在系统内存有限的情况下，可使用 **GetChunk** 方法处理部分而非全部的长整型值。

GetChunk 调用返回的数据将赋给“变量”。如果 **Size** 大于剩余的数据，则 **GetChunk** 仅返回剩余的数据而无需用空白填充“变量”。如果字段为空，则 **GetChunk** 方法返回 Null。

每个后续的 **GetChunk** 调用将检索从前一次 **GetChunk** 调用停止处开始的数据。但是，如果从一个字段检索数据然后在当前记录中设置或读取另一个字段的值，ADO 将认为已从第一个字段中检索出数据。如果在第一个字段上再次调用 **GetChunk** 方法，ADO 将把调用解释为新的 **GetChunk** 操作并从记录的起始处开始读取。如果其他 **Recordset** 对象不是首个 **Recordset** 对象的副本，则访问其中的字段不会破坏 **GetChunk** 操作。

如果 **Field** 对象的 Attributes(See 1.1.4.6.6) 属性中的 **adFldLong** 位设置为 **True**，则可以对该字段使用 **GetChunk** 方法。

如果在 **Field** 对象上使用 **Getchunk** 方法时没有当前记录，将产生错误 3021（无当前记录）。

1.1.4.4.25 GetRows 方法 (ADO)

将 Recordset 对象的多个记录恢复到数组中。

语法

array = recordset.**GetRows(*Rows*, *Start*, *Fields*)**

返回值

返回二维数组。

参数

Rows 可选，**长整型**表达式，指定要检索记录数。默认值为 **adGetRowsRest** (-1)。

Start 可选，**字符串**或**长整型**，计算得到在 **GetRows** 操作开始处的记录的书签。也可使用下列 **BookmarkEnum** 值。

常量	说明
AdBookmarkCurrent	从当前记录开始。
AdBookmarkFirst	从首记录开始。
AdBookmarkLast	从尾记录开始。

Fields 可选，**变体型**，代表单个字段名、顺序位置、字段名数组或顺序位置号。ADO 仅返回这些字段中的数据。

说明

使用 **GetRows** 方法可将记录从 **Recordset** 复制到二维数组中。第一个下标标识字段，第二个则标识记录号。当 **GetRows** 方法返回数据时数组变量将自动调整到正确大小。

如果不指定 **Rows** 参数的值，**GetRows** 方法将自动检索 **Recordset** 对象中的所有记录。如果请求的记录比可用记录多，则 **GetRows** 仅返回可用记录数。

如果 **Recordset** 对象支持书签，则可以通过传送该记录的 Bookmark(See 1.1.4.6.8) 属性值，来指定 **GetRows** 方法将从哪个记录开始检索数据。

如要限制 **GetRows** 调用返回的字段，则可以在 **Fields** 参数中传送单个字段名/编号或者字段名/编号数组。

在调用 **GetRows** 后，下一个未读取的记录成为当前记录，或者如果没有更多的记录，则 EOF(See 1.1.4.6.7) 属性设置为 **True**。

1.1.4.4.26 GetString 方法 (ADO Recordset)

将 **Recordset** 作为字符串返回。

语法

Set *Variant = recordset*.**GetString*(StringFormat, NumRows, ColumnDelimiter, RowDelimiter, NullExpr)***

返回值

将 **Recordset** 按字符串值的**变体型** (BSTR) 返回。

参数

StringFormat 指定 **Recordset** 应转换为下列格式。

常量	说明
adClipString	行由 **RowDelimiter** 分界，列由 **ColumnDelimiter** 分界，NULL 值由 **NullExpr** 分界。这三个参数只有在与 **adClipString** 一起时才有效。

NumRows 可选。记录集要转换的行数。如果没有指定 **NumRows**，或者它大于记录集的总行数，则记录集的所有行都要转换。

ColumnDelimiter 可选。如果指定则为列与列之间的分界符，否则为 TAB 字符。

RowDelimiter 可选。如果指定则为行与行之间的分界符，否则为 CARRIAGE RETURN 字符。

NullExpr 可选。如果指定则为 NULL 值处的表达式，否则为空字符串。

说明

行数据（但不是模式数据）保存在串中。因此不能使用该字符串重新打开记录集。

该方法等价于 RDO **GetClipString** 方法。

1.1.4.4.27 Item 方法 (ADO)

根据名称或序号返回集合的特定成员。

语法

Set object = collection.**Item (*Index*)**

返回值

返回对象引用。

参数

Index **变体型**，计算集合中对象的名称或顺序号。

说明

使用 **Item** 方法返回集合中的特定对象。如果方法无法在对应于 **Index** 参数的集合中找到对象，将产生错误。同时，某些集合不支持已命名的对象，对于这些集合，必须使用顺序号引用。

Item 方法是所有集合的默认方法；因此，下列语法形式是可互换的：

collection.**Item (*Index*)**

collection **(*Index*)**

1.1.4.4.28 Move 方法 (ADO)

移动 Recordset(See 1.1.4.2.10) 对象中当前记录的位置。

语法

recordset.**Move** ***NumRecords, Start***

参数

NumRecords 带符号**长整型**表达式，指定当前记录位置移动的记录数。

Start 可选，**字符串**或**变体型**，用于计算书签。也可为下列 **BookmarkEnum** 值之一：

常量	说明
AdBookmarkCurrent	默认。从当前记录开始。
AdBookmarkFirst	从首记录开始。
AdBookmarkLast	从尾记录开始。

说明

所有 **Recordset** 对象都支持 **Move** 方法。

如果 **NumRecords** 参数大于零，则当前记录位置将向前移动（向记录集的末尾）。如果 **NumRecords** 小于零，则当前记录位置向后移动（向记录集的开始）。

如果 **Move** 调用将当前记录位置移动到首记录之前，则 ADO 将当前记录放置在记录集（BOF(See 1.1.4.6.7) 为 **True**）的首记录之前。在 **BOF** 属性已经为 **True** 时试图向后移动将产生错误。

如果 **Move** 调用将当前记录位置移动到尾记录之后，则 ADO 将当前记录放置在记录集（EOF(See 1.1.4.6.7) 为 **True**）的尾记录之后。在 **EOF** 属性已经为 **True** 时试图向前移动将产生错误。

从空的 **Recordset** 对象调用 **Move** 方法将产生错误。

如果传送 **Start** 参数，则移动相对于该书签的记录（假定 **Recordset** 对象支持书签）。如果没有指定，则移动相对于当前记录。

如果使用 CacheSize(See 1.1.4.6.9) 属性在本地缓存来自提供者的记录，则在传送将当前记录位置移动到当前缓存的记录组之外的 **NumRecords** 参数时， ADO 将不得不从目标记录开始检索新的记录组。**CacheSize** 属性决定新检索记录组的大小，而目标记录是检索到的第一个记录。

如果 **Recordset** 对象是仅向前的，则用户仍然可以传送小于零的 **NumRecords** 参数（只要目标在已缓存记录的当前集合中）。如果 **Move** 调用将当前记录位置移动到第一个已缓存记录的前一个记录，将产生错误。因此可使用记录缓存，它在支持仅向前滚动的提供者上支持完全滚动。由于缓存的记录将加载到内存，因此应避免不必要地缓存过多记录。即使仅向前 **Recordset** 对象支持这种方式的向后移动，在任何仅向前的 **Recordset** 对象上调用 MovePrevious(See 1.1.4.4.29) 方法仍将产生错误。

1.1.4.4.29 MoveFirst 、 MoveLast 、 MoveNext 和 MovePrevious 方法 (ADO)

在指定 Recordset(See 1.1.4.2.10) 对象中移动到第一个、最后一个、下一个或前一个记录并使该记录成为当前记录。

语法

recordset.{**MoveFirst** | **MoveLast** | **MoveNext** | **MovePrevious**}

说明

使用 **MoveFirst** 方法将当前记录位置移动到 **Recordset** 中的第一个记录。

使用 **MoveLast** 方法将当前记录位置移动到 **Recordset** 中的最后一个记录。**Recordset** 对象必须支持书签或向后光标移动；否则调用该方法将产生错误。

使用 **MoveNext** 方法将当前记录向前移动一个记录（向 **Recordset** 的底部）。如果最后一个记录是当前记录并且调用 **MoveNext** 方法，则 ADO 将当前记录设置到 **Recordset**（EOF(See 1.1.4.6.7) 为 **True**）的尾记录之后。当 **EOF** 属性已经为 **True** 时试图向前移动将产生错误。

使用 **MovePrevious** 方法将当前记录位置向后移动一个记录（向记录集的顶部）。Recordset 对象必须支持书签或向后游标移动；否则方法调用将产生错误。如果首记录是当前记录并且调用 **MovePrevious** 方法，则 ADO 将当前记录设置在 **Recordset**（BOF(See 1.1.4.6.7) 为 **True**）的首记录之前。而 **BOF** 属性为 **True** 时向后移动将产生错误。如果 **Recordset** 对象不支持书签或向后游标移动，则 **MovePrevious** 方法将产生错误。

如果记录集是仅向前的，但是用户希望支持向前和向后滚动，则可以使用 CacheSize(See 1.1.4.6.9) 属性创建记录缓存，通过 Move(See 1.1.4.4.28) 方法支持向后游标移动。由于缓存记录是加载到内存中的，所以应避免不必要地缓存太多记录。可以调用仅向前 **Recordset** 对象的 **MoveFirst** 方法；这样做可使提供者重新执行生成 **Recordset** 对象的命令。

1.1.4.4.30 MoveFirst 、 MoveLast 、 MoveNext 、 MovePrevious 方法 (RDS)

在显示的记录集中移动到第一个、最后一个、下一个或前一个记录。

语法

DataControl.**Recordset.**{**MoveFirst** | **MoveLast** | **MoveNext** | **MovePrevious**}

参数

DataControl 对象变量(See 2.34)，代表 **RDS.DataControl** 对象。

说明

可以使用 **RDS.DataControl** 对象的 **Move** 方法在 Web 页的数据绑定控件的数据记录中定位。例如，假设通过绑定到 **RDS.DataControl** 对象来显示网格中的记录集。然后可以加入“第一个”、“最后一个”、“下一个”或“上一个”按钮，以便用户可以单击这些按钮移动到记录集的第一个、最后一个、下一个和上一个记录。对于“第一个”、“最后一个”、“下一个”和“上一个”按钮，可分别通过调用 onClick 过程中 **RDS.DataControl** 对象的 **MoveFirst**、**MoveLast**、**MoveNext** 和 **MovePrevious** 方法来实现。地址簿范例(See 1.1.6.1.4.11)将说明其具体方法。

1.1.4.4.31 NextRecordset 方法 (ADO)

清除当前 Recordset(See 1.1.4.2.10) 对象并通过提前执行命令序列返回下一个记录集。

语法

Set *recordset2* = *recordset1*.**NextRecordset(*RecordsAffected*)**

返回值

返回 **Recordset** 对象。在语法模型中，recordset1 和 recordset2 可以是相同的 **Recordset** 对象，或者可以使用不同的对象。

参数

RecordsAffected 可选，**长整型**变量，提供者向其返回当前操作所影响的记录数目。

注意 该参数仅返回受操作影响的记录的数目；它不会从用于生成 **Recordset** 的选择语句返回记录的计数。

说明

使用 **NextRecordset** 方法返回复合命令语句中下一条命令的结果，或者是返回多个结果的已存储过程结果。如果使用 **Command** 的 Execute(See 1.1.4.4.21) 方法或者 **Recordset** 的 Open(See 1.1.4.4.33) 方法打开基于复合命令语句（例如“SELECT * FROM table1;SELECT * FROM table2”）的 **Recordset** 对象，则 ADO 仅执行第一条命令并将结果返回到“记录集”。要访问语句中后续命令的结果，请调用 **NextRecordset** 方法。

只要有其他的结果，并且包含复合语句的 **Recordset** 未被跨进程边界调度，则 **NextRecordset** 方法将继续返回 **Recordset** 对象。如果行返回命令没有返回记录，则返回的 **Recordset** 对象将为空；在确认 BOF(See 1.1.4.6.7) 和 EOF(See 1.1.4.6.7) 都为 **True** 的情况下可验证这种情况。如果非按行返回命令成功执行，则返回的 Recordset 对象将关闭，通过测试 **Recordset** 的 State(See 1.1.4.6.64) 属性可以测试这种情况。如果没有其他的结果，“记录集”将设置为 *Nothing*。

远程数据服务用法 **NextRecordset** 方法对客户端的 **Recordset** 对象不可用。

如果在立即更新模式下进行编辑，调用 **NextRecordset** 方法将产生错误。应首先调用 Update(See 1.1.4.4.45) 或 CancelUpdate(See 1.1.4.4.8) 方法。

如果需要通过填写 Parameters(See 1.1.4.3.3) 集合或者通过使用原有的 **Open** 或 **Execute** 调用传送数组为复合语句中的多个命令传送参数，则参数在集合或数组中的次序必须与它们在命令序列中各自命令的次序相同。在读取输出参数值之前必须读取所有结果。

在调用 **NextRecordset** 方法时，ADO 仅执行语句中的下一条命令。如果在单步执行整个命令语句之前显式关闭 **Recordset** 对象，则 ADO 不执行其余的命令。

1.1.4.4.32 Open 方法 (ADO Connection)

打开到数据源的连接。

语法

connection.**Open *ConnectionString, UserID, Password, Options***

参数

ConnectionString 可选，**字符串**，包含连接信息。参阅 ConnectionString(See 1.1.4.6.14) 属性可获得有效设置的详细信息。

UserID 可选，**字符串**，包含建立连接时所使用用户名。

Password 可选，**字符串**，包含建立连接时所使用密码。

Options 可选，**ConnectOptionEnum** 值。决定该方法是在连接建立之后（异步）还是连接建立之前（同步）返回。可以是如下某个常量：

常量	说明
adConnectUnspecified	（默认）同步打开连接。
adAsyncConnect	异步打开连接。ConnectComplete(See 1.1.4.5.2) 事件可以用于决定连接何时可用。

说明

使用 **Connection** 对象的 **Open** 方法可建立到数据源的物理连接。在该方法成功完成后连接是活跃的，可以对它发出命令并且处理结果。

使用可选的 ***ConnectionString*** 参数指定连接字符串，它包含由分号分隔的一系列 *argument* = *value* 语句。**ConnectionString** 属性自动继承用于 ***ConnectionString*** 参数的值，因此可在打开之前设置 **Connection** 对象的 **ConnectionString** 属性，或在 **Open** 方法调用时使用 ***ConnectionString*** 参数设置或覆盖当前连接参数。

如果在 ***ConnectionString*** 参数和可选的 ***UserID*** 及 ***Password*** 参数中传送用户和密码信息，那么 ***UserID*** 和 ***Password*** 参数将覆盖 ***ConnectionString*** 中指定的值。

在对打开的 **Connection** 的操作结束后，可使用 Close(See 1.1.4.4.12) 方法释放所有关联的系统资源。关闭对象并非将它从内存中删除；可以更改它的属性设置并在以后再次使用 **Open** 方法打开它。要将对象完全从内存中删除，可将对象变量设置为 *Nothing*。

> **远程数据服务用法** 当在客户端的 Connection 对象上使用 Open 方法时，在 Connection 对象上打开 Recordset 之前 Open 方法其实并未建立到服务器的连接。

1.1.4.4.33 Open 方法 (ADO Recordset)

打开游标。

语法

recordset.**Open** ***Source, ActiveConnection, CursorType, LockType, Options***

参数

Source 可选，变体型，计算 Command(See 1.1.4.2.1) 对象的变量名、SQL 语句、表名、存储过程调用或持久 **Recordset** 文件名。

ActiveConnection 可选。变体型，计算有效 **Connection** 对象变量名；或字符串，包含 ConnectionString(See 1.1.4.6.14) 参数。

CursorType 可选，**CursorTypeEnum** 值，确定提供者打开 **Recordset** 时应该使用的游标类型。可为下列常量之一（参阅 CursorType(See 1.1.4.6.18) 属性可获得这些设置的定义）。

常量	说明
AdOpenForwardOnly	（默认值）打开仅向前类型游标。
AdOpenKeyset	打开键集类型游标。
AdOpenDynamic	打开动态类型游标。
AdOpenStatic	打开静态类型游标。

LockType 可选。确定提供者打开 **Recordset** 时应该使用的锁定（并发）类型的 **LockTypeEnum** 值，可为下列常量之一（参见 LockType(See 1.1.4.6.37) 属性可获得详细信息）。

常量	说明
AdLockReadOnly	（默认值）只读 — 不能改变数据。
AdLockPessimistic	保守式锁定（逐个） — 提供者完成确保成功编辑记录所需的工作，通常通过在编辑时立即锁定数据源的记录。
AdLockOptimistic	开放式锁定（逐个） — 提供者使用开放式锁定，只在调用 Update(See 1.1.4.4.45) 方法时才锁定记录。
AdLockBatchOptimistic	开放式批更新—用于批更新模式（与立即更新模式相对）。

Options 可选，**长整型**值，用于指示提供者如何计算 Source 参数（如果它代表的不是 **Command** 对象），或从以前保存 **Recordset** 的文件中恢复 **Recordset**。可为下列常量之一（参见 CommandType(See 1.1.4.6.12) 属性可获得该列表中前五个常量的详细说明）。

常量	说明
adCmdText	指示提供者应该将 ***Source*** 作为命令的文本定义来计算。
adCmdTable	指示 ADO 生成 SQL 查询以便从在 ***Source*** 中命名的表中返回所有行。
adCmdTableDirect	指示提供者更改从在 ***Source*** 中命名的表中返回所有行。
adCmdStoredProc	指示提供者应该将 ***Source*** 视为存储过程。
adCmdUnknown	指示 ***Source*** 参数中的命令类型为未知。
adCmdFile	指示应从在 ***Source*** 中命名的文件中恢复保留（保存的）**Recordset**。
adAsyncExecute	指示应异步执行 ***Source***。
adAsyncFetch	指示在提取 **Initial Fetch Size** 属性中指定的初始数量后，应该异步提取所有剩余的行。如果所需的行尚未提取，主要的线程将被堵塞直到行重新可用。
adAsyncFetchNonBlocking	指示主要线程在提取期间从未堵塞。如果所请求的行尚未提取，当前行自动移到文件末尾。

说明

使用 **Recordset** 对象的 **Open** 方法可打开代表基本表、查询结果或者以前保存的 **Recordset** 中记录的游标。

使用可选的 ***Source*** 参数指定使用下列内容之一的数据源：**Command** 对象变量、SQL 语句、存储过程、表名或完整的文件路径名。

如果 ***Source*** 是文件路径名，它可以是完整路径（“c:\dir\file.rst”）、相对路径（“..\file.rst”）或 URL（“http://files/file.rst”）。

ActiveConnection 参数对应于 ActiveConnection(See 1.1.4.6.4) 属性，并指定在哪个连接中打开 **Recordset** 对象。如果传送该参数的连接定义，则 ADO 使用指定的参数打开新连接。可以在打开 **Recordset** 之后更改该属性的值以便将更新发送到其他提供者。或者可以将该属性设置为 **Nothing**（在 Microsoft Visual Basic 中）以便将 **Recordset** 与所有提供者断开。

对于直接对应于 **Recordset** 对象属性的参数（***Source***、***CursorType*** 和 ***LockType***），参数和属性的关系如下：

- 在 **Recordset** 对象打开之前属性是读/写。
- 除非在执行 **Open** 方法时传送相应的参数，否则将使用属性设置。如果传送参数，则它将覆盖相应的属性设置，并且用参数值更新属性设置。
- 在打开 **Recordset** 对象后，这些属性将变为只读。

注意 对于其 Source(See 1.1.4.6.61) 属性被设置为有效 **Command** 对象的 **Recordset** 对象，即使 **Recordset** 对象没有打开，**ActiveConnection** 属性也是只读的。

如果在 ***Source*** 参数中传送 **Command** 对象并且同时传递 ***ActiveConnection*** 参数，那么将产生错误。**Command** 对象的 **ActiveConnection** 属性必须已经设置为有效的 **Connection** 对象或者连接字符串。

如果在 ***Source*** 参数中传送的不是 **Command** 对象，那么可以使用 ***Options*** 参数优化对 ***Source*** 参数的计算。如果没有定义 ***Options*** 则性能将会降低，原因是 ADO 必须调用提供者以确定参数是否为 SQL 语句、存储过程或表名。如果已确定所用的 ***Source*** 类型，则可以设置 ***Options*** 参数以指示 ADO 直接跳到相关的代码。如果 ***Options*** 参数不匹配 ***Source*** 类型，将产生错误。

如果不存在与 **Recordset** 关联的连接，***Options*** 参数的默认值将为 **adCmdFile**。这是持久 **Recordset** 对象的典型情况。

如果数据源没有返回记录，那么提供者将 BOF(See 1.1.4.6.7) 和 EOF(See 1.1.4.6.7) 属性同时设置为 **True**，并且不定义当前记录位置。如果游标类型允许，仍然可以将新数据添加到该空 **Recordset** 对象。

在打开的 **Recordset** 对象上完成操作时，可使用 Close(See 1.1.4.4.12) 方法释放任何相关的系统资源。关闭对象并非将它从内存中删除，可以更改它的属性设置并且在以后使用 **Open** 方法再次将其打开。要将对象从内存中完全删除，可将对象变量设置为 *Nothing*。

在设置 **ActiveConnection** 属性之前调用不带操作数的 **Open**，可通过将字段追加到 **Recordset Fields** 集合创建 **Recordset** 的实例。

如果已经将 **CursorLocation** 属性设置为 **adUseClient**，就可以采用两种途径之一异步检索行。建议使用的方法是将 ***Options*** 设置为 **adAsyncFetch**。或者，可以使用在 **Properties** 集合中的“异步行集合处理”动态属性，但如果未将 **Options** 参数设置为 **adAsyncFetch**，则可能丢失相关的被检索事件。

注意 在 MSRemote 提供者中的背景提取仅能通过 **Open** 方法的 ***Options*** 参数得到支持。

1.1.4.4.34 OpenSchema 方法 (ADO)

从提供者获取数据库模式信息。

语法

Set *recordset* = *connection*.**OpenSchema (*QueryType*, *Criteria*, *SchemaID*)**

返回值

返回包含模式信息的 **Recordset** 对象。**Recordset** 将以只读、静态游标打开。

参数

QueryType 所要运行的模式查询类型，可以为下列任意常量。

Criteria 可选。每个 ***QueryType*** 选项的查询限制条件数组，如下所列：

QueryType **值**	***Criteria*** **值**

AdSchemaAsserts	CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME
AdSchemaCatalogs	CATALOG_NAME
AdSchemaCharacterSets	CHARACTER_SET_CATALOG CHARACTER_SET_SCHEMA CHARACTER_SET_NAME
AdSchemaCheckConstraints	CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME
AdSchemaCollations	COLLATION_CATALOG COLLATION_SCHEMA COLLATION_NAME
AdSchemaColumnDomainUsage	DOMAIN_CATALOG DOMAIN_SCHEMA DOMAIN_NAME COLUMN_NAME
AdSchemaColumnPrivileges	TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME GRANTOR GRANTEE
adSchemaColumns	TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME
adSchemaConstraintColumnUsage	TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME
adSchemaConstraintTableUsage	TABLE_CATALOG TABLE_SCHEMA TABLE_NAME
adSchemaForeignKeys	PK_TABLE_CATALOG PK_TABLE_SCHEMA PK_TABLE_NAME FK_TABLE_CATALOG

	FK_TABLE_SCHEMA FK_TABLE_NAME
adSchemaIndexes	TABLE_CATALOG TABLE_SCHEMA INDEX_NAME TYPE TABLE_NAME
adSchemaKeyColumnUsage	CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME TABLE_CATALOG TABLE_SCHEMA TABLE_NAME COLUMN_NAME
adSchemaPrimaryKeys	PK_TABLE_CATALOG PK_TABLE_SCHEMA PK_TABLE_NAME
adSchemaProcedureColumns	PROCEDURE_CATALOG PROCEDURE_SCHEMA PROCEDURE_NAME COLUMN_NAME
adSchemaProcedureParameters	PROCEDURE_CATALOG PROCEDURE_SCHEMA PROCEDURE_NAME PARAMTER_NAME
adSchemaProcedures	PROCEDURE_CATALOG PROCEDURE_SCHEMA PROCEDURE_NAME PROCEDURE_TYPE
adSchemaProviderSpecific	参见说明
adSchemaProviderTypes	DATA_TYPE BEST_MATCH
adSchemaReferentialConstraints	CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME
adSchemaSchemata	CATALOG_NAME SCHEMA_NAME SCHEMA_OWNER

adSchemaSQLLanguages	<无>
adSchemaStatistics	TABLE_CATALOG TABLE_SCHEMA TABLE_NAME
adSchemaTableConstraints	CONSTRAINT_CATALOG CONSTRAINT_SCHEMA CONSTRAINT_NAME TABLE_CATALOG TABLE_SCHEMA TABLE_NAME CONSTRAINT_TYPE
adSchemaTablePrivileges	TABLE_CATALOG TABLE_SCHEMA TABLE_NAME GRANTOR GRANTEE
adSchemaTables	TABLE_CATALOG TABLE_SCHEMA TABLE_NAME TABLE_TYPE
adSchemaTranslations	TRANSLATION_CATALOG TRANSLATION_SCHEMA TRANSLATION_NAME
adSchemaUsagePrivileges	OBJECT_CATALOG OBJECT_SCHEMA OBJECT_NAME OBJECT_TYPE GRANTOR GRANTEE
adSchemaViewColumnUsage	VIEW_CATALOG VIEW_SCHEMA VIEW_NAME
adSchemaViewTableUsage	VIEW_CATALOG VIEW_SCHEMA VIEW_NAME
adSchemaViews	TABLE_CATALOG TABLE_SCHEMA TABLE_NAME

SchemaID OLE DB 规范没有定义用于提供者模式查询的 GUID。如果 ***QueryType*** 设置为 **adSchemaProviderSpecific**，则需要该参数，否则不使用它。

说明

OpenSchema 方法返回与数据源有关的信息，例如关于服务器上的表以及表中的列等信息。

Criteria 参数是可用于限制模式查询结果的值数组。每个模式查询有它支持的不同参数集。实际模式由 **IDBSchemaRowset** 接口下的 OLE DB 规范定义。ADO 中所支持的参数集已在上面列出。

如果提供者定义未在上面列出的非标准模式查询，则常量 **adSchemaProviderSpecific** 将用于 **QueryType** 参数。在使用该常量时需要 ***SchemaID*** 参数传递模式查询的 GUID 以用于执行。如果 ***QueryType*** 设置为 **adSchemaProviderSpecific** 但是没有提供 **SchemaID**，将导致错误。

提供者不需要支持所有的 OLE DB 标准模式查询，只有 **adSchemaTables**、**adSchemaColumns** 和 **adSchemaProviderTypes** 是 OLE DB 规范需要的。但是对于这些模式查询，提供者不需要支持上面列出的 ***Criteria*** 条件约束。

远程数据服务用法 OpenSchema 方法在客户端 Connection 对象上无效。

注意 在 Visual Basic 中，在由 Connection 对象的 OpenSchema 方法所返回的 Recordset 中有 4 字节无符号整型 (DBTYPE UI4) 的列无法与其他变量比较。有关 OLE DB 数据类型的详细信息，请参阅“Microsoft OLE DB 程序员参考”的第十章和附录 A。

1.1.4.4.35 Query 方法 (RDS)

使用有效的 SQL 查询字符串返回 **Recordset**。

语法

Set *Recordset* = *DataFactory*.**Query** *Connection, Query*

参数

Recordset 对象变量(See 2.34)，代表 **Recordset** 对象。

DataFactory 对象变量，代表 **RDSServer.DataFactory** 对象。

Connection **字符串**，包含服务器连接信息。它类似于 **Connect** 属性。

Query **字符串**，包含 SQL 查询。

说明

查询应该使用数据库服务器的特定 SQL 语言。如果执行的查询发生错误，则返回结果状态。**Query** 方法不对 **Query** 字符串进行任何语法检查。

1.1.4.4.36 Refresh 方法 (ADO)

更新集合中的对象以便反映来自提供者的可用对象和特定于提供者的对象。

语法

collection.**Refresh**

说明

Refresh 方法根据从中调用的不同集合而完成不同的任务。

参数

使用 Command(See 1.1.4.2.1) 对象的 **Parameters** 集合上的 **Refresh** 方法可为在 **Command** 对象中指定的存储过程或者参数化查询检索提供者端参数信息。对于不支持存储过程调用或参数化查询的提供者来说，集合将为空。
在调用 **Refresh** 方法之前应该将 **Command** 对象的 ActiveConnection(See 1.1.4.6.4) 属性设置为有效 Connection(See 1.1.4.2.2) 对象，将 CommandText(See 1.1.4.6.10) 属性设置为有效命令，并且将 CommandType(See 1.1.4.6.12) 属性设置为 **adCmdStoredProc**。如果在调用 **Refresh** 方法之前访问 **Parameters** 集合，ADO 将自动调用方法并填充集合。

> **注意** 如果使用 **Refresh** 方法从提供者获取参数信息而它返回一个或多个变长数据类型 Parameter(See 1.1.4.2.8) 对象，则 ADO 可能根据其大小的最大可能值为参数分配内存，这在执行期间将会导致错误。在调用 Execute(See 1.1.4.4.21) 方法之前应显式设置这些参数的 Size(See 1.1.4.6.56) 属性以防止错误发生。

`Fields`

在 **Fields** 集合上使用 **Refresh** 方法没有可见的效果。要从基本数据库结构中对更改进行检索，必须使用 Requery(See 1.1.4.4.38) 方法；如果 **Recordset** 对象不支持书签，则使用 MoveFirst(See 1.1.4.4.29) 方法。

`Properties`

在某些对象的 **Properties** 集合上使用 **Refresh** 方法可使用提供者提供的动态属性填写集合，这些属性只将功能性信息提供给 ADO 支持的内置属性之外的提供者。

1.1.4.4.37 Refresh 方法 (RDS)

对在 Connect(See 1.1.4.6.13) 属性中指定的 ODBC(See 2.35) 数据源(See 2.18) 进行再查询并更新查询结果。

语法

DataControl.**Refresh**

参数

DataControl 对象变量(See 2.34)，代表 **RDS.DataControl** 对象。

说明

在使用 Refresh 方法之前必须设置 Connect(See 1.1.4.6.13)、Server(See 1.1.4.6.55) 和 SQL(See 1.1.4.6.62) 属性。与

RDS.DataControl 对象关联的表中的所有数据绑定控件(See 2.16)反映新的记录集，所有先前存在的 **Recordset** 对象将被释放，并且放弃所有未保存的更改。**Refresh** 方法可使首记录自动成为当前记录。

在处理数据时最好定期调用 **Refresh** 方法。如果检索数据并将其在客户机上放置一段时间，数据很有可能会过期，这时所做的所有更改都将失效，原因是其他人可能在您之前更改了记录并先于您将更改提交。

1.1.4.4.38 Requery 方法 (ADO)

通过重新执行对象所基于的查询，更新 Recordset(See 1.1.4.2.10) 对象中的数据。

语法

recordset.**Requery** ***Options***

参数

Options 可选。指示影响该操作选项的位屏蔽。如果该参数设置为 **adAsyncExecute**，则该操作将异步执行并在它结束时产生 RecordsetChangeComplete(See 1.1.4.5.12) 事件。

说明

通过重新发出原始命令并再次检索数据，可使用 **Requery** 方法刷新来自数据源的 **Recordset** 对象的全部内容。调用该方法等于相继调用 Close(See 1.1.4.4.12) 和 Open(See 1.1.4.4.33) 方法。如果正在编辑当前记录或者添加新记录将产生错误。

在 **Recordset** 对象打开期间，定义游标性质（CursorType(See 1.1.4.6.18)、LockType(See 1.1.4.6.37)、MaxRecords(See 1.1.4.6.39) 等）的属性为只读，因此 **Requery** 方法只能刷新当前游标。要更改某个游标属性并查看结果，必须使用 **Close** 方法使属性再次成为读/写。然后可以更改属性设置并且调用 **Open** 方法重新打开游标。

1.1.4.4.39 Reset 方法 (RDS)

根据指定的排序和筛选属性对客户端 **Recordset** 执行排序或筛选操作。

语法

DataControl.**Reset(***value***)**

参数

DataControl 对象变量(See 2.34)，代表 **RDS.DataControl** 对象。

value 可选，**布尔型**值，如果希望在当前的“已筛选”行集合上执行筛选操作，则它是值为 **True** （默认值）；**False** 表示在原始行集合上执行筛选操作，并删除所有以前的筛选操作选项。

说明

SortColumn(See 1.1.4.6.58)、SortDirection(See 1.1.4.6.59)、FilterValue(See 1.1.4.6.31)、FilterCriterion(See 1.1.4.6.30) 和 FilterColumn(See 1.1.4.6.29) 属性提供客户端缓存上的排序和筛选功能。排序功能根据某列的值将记录排序。当全部 **Recordset** 保留在缓存中时，筛

选功能根据查找标准显示记录子集。**Reset** 方法将执行查找标准并用可更新的 **Recordset** 替换当前 **Recordset**。

如果还存在没有提交的对原始数据所做的更改，那么 **Reset** 方法将失效。首先使用 SubmitChanges(See 1.1.4.4.43) 方法将所有改动保存在读/写 **Recordset** 中，然后使用 **Reset** 方法排序或筛选记录。

如果希望对行集合执行多项筛选，可使用 **Reset** 方法中可选的 ***Boolean*** 参数。下例将说明这一过程：

```
ADC.SQL = "Select au_lname from authors"
ADC.Refresh      '获得新的行集合。

ADC.FilterColumn = "au_lname"
ADC.FilterCriterion = "<"
ADC.FilterValue = "'M'"
ADC.Reset        '记录集现在包含所有姓氏首字母小于“M”的记录。

ADC.FilterCriterion = ">"
ADC.FilterValue = "'F'"
'不必传送 True 值，因为它是对当前的
'“已筛选”记录集的默认筛选条件。
ADC.Reset(TRUE)       '记录集现在包含所有姓氏首字母
                  '小于“M”且大于“F”的记录。

ADC.FilterCriterion = ">"
ADC.FilterValue = "'T'"
'对原始记录集进行筛选，放弃以前的筛选选项。
ADC.Reset(FALSE)    '记录集现在包含所有姓氏首字母
                '大于 "T" 的记录。
```

1.1.4.4.40 Resync 方法 (ADO)

从基本数据库刷新当前 Recordset(See 1.1.4.2.10) 对象中的数据。

语法

recordset.**Resync** ***AffectRecords, ResyncValues***

参数

AffectRecords 可选，**AffectEnum** 值，决定 **Resync** 方法所影响的记录数目，可以为下列常量之一。

常量	说明
AdAffectCurrent	只刷新当前记录。
AdAffectGroup	刷新满足当前 Filter(See 1.1.4.6.28) 属性设置的记录。只有将 **Filter** 属性设置为有效预定义常量之一才能使用该选项。
AdAffectAll	默认值。刷新 **Recordset** 对象中的所有记录，包括由于当前 **Filter** 属性设置而隐藏的记录。

adAffectAllChapters	刷新所有子集记录。

ResyncValues 可选，**ResyncEnum** 值。指定是否覆盖基本值。可为下列常量之一。

常量	说明
AdResyncAllValues	默认值。覆盖数据，取消挂起的更新。
AdResyncUnderlyingValues	不覆盖数据，不取消挂起的更新。

说明

使用 **Resync** 方法将当前 **Recordset** 中的记录与基本的数据库重新同步。这在使用静态或仅向前的游标但希望看到基本数据库中的改动时十分有用。

如果将 CursorLocation(See 1.1.4.6.17) 属性设置为 **adUseClient**，则 **Resync** 仅对非只读的 **Recordset** 对象可用。

与 Requery(See 1.1.4.4.38) 方法不同，**Resync** 方法不重新执行 **Recordset** 对象的基本的命令，基本的数据库中的新记录将不可见。

如果由于与基本的数据冲突（如其他用户已将记录删除）而使重新同步的尝试失败，则提供者将警告返回到 Errors(See 1.1.4.3.1) 集合并且产生运行时错误。使用 **Filter** 属性 (**adFilterConflictingRecords**) 和 Status(See 1.1.4.6.65) 属性可以找到发生冲突的记录。

远程数据服务用法 Resync 方法在客户端 Recordset 上无效。

如果设置了 Unique Table(See 1.1.4.7.2) 和 Resync Command(See 1.1.4.7.3) 动态属性，并且 Recordset 是对多个表执行 JOIN 操作的结果，那么，Resync 方法将仅对 Unique Table 属性中命名的表执行在 Resync Command 属性中所给定的操作。

1.1.4.4.41 Save 方法 (ADO Recordset)

将 **Recordset** 保存（持久）在文件中。

语法

recordset.**Save** ***FileName, PersistFormat***

参数

FileName 可选。文件的完整路径名，用于保存 **Recordset**。

PersistFormat 可选。**PersistFormatEnum** 值，指定保存 **Recordset** 所使用的格式。可以是如下的某个常量：

常量	说明
adPersistADTG	（默认）使用专用的“Advanced Data Tablegram”格式保存。
adPersistXML	使用 XML 格式保存。

说明

只能对打开的 **Recordset** 调用 **Save** 方法。随后使用 Open(See 1.1.4.4.33) 方法可由 ***FileName*** 恢复 **Recordset**。

如果 Filter(See 1.1.4.6.28) 属性影响 **Recordset**，将只保存经过筛选的行。如果 **Recordset** 是分级结构的，那么将保存当前子 **Recordset** 和它的子 **Recordset**，但不保存上一级 **Recordset**。

在第一次保存 **Recordset** 时指定 ***FileName***。如果随后调用 **Save** 时，应忽略 ***FileName***，否则将产生运行时错误。如果随后使用新的 ***FileName*** 调用 **Save**，那么 **Recordset** 将保存到新的文件中，但新文件和原始文件都是打开的。

Save 不关闭 **Recordset** 或 ***FileName***，从而可以继续使用 **Recordset** 并保存最新的更改。在 **Recordset** 关闭之前 ***FileName*** 将保持打开，在这段时间其它应用程序可以读取但不能写入 ***FileName***。

出于安全的原因，对由 Microsoft Internet Explorer 执行的脚本，**Save** 方法仅允许使用低的和自定义的安全设置。有关安全问题的详细解释，请参阅 在 http://www.microsoft.com/data/techmat.htm 上的白皮书标题“Security Issues in the Microsoft Internet Explorer”。

如果正在进行异步 **Recordset** 获取、执行或更新操作时调用 **Save** 方法，则 **Save** 将进入等待状态，直到异步操作完成。

在 **Save** 方法完成后，当前行位置将成为 **Recordset** 的首行。

要得到最佳结果，应使用 **Save** 将 **CursorLocation** 属性设置为 **adUseClient**。如果您的提供者不支持用于保存 **Recordset** 对象的所需功能，则客户端游标将提供该功能。

1.1.4.4.42 Seek 方法

搜索 **Recordset** 的索引，快速定位与指定值相匹配的行，并将当前行更改为该行。

语法

recordset.**Seek** ***KeyValues, SeekOption***

参数

KeyValues VARIANT 值的数组。索引由一个或多个列组成，而数组包含与每个对应列进行比较的值。

SeekOption **SeekEnum** 只值，指定在索引的列和对应的 ***KeyValues*** 之间进行的比较的类型。可以是如下某个比较常量：

常量	说明
AdSeekAfterEQ	查找等于 ***KeyValues*** 的关键字，或仅在已经匹配过的位置之后进行查找。
AdSeekAfter	仅在已经有过与 ***KeyValues*** 匹配的位置之后进行查找。
AdSeekBeforeEQ	查找等于 ***KeyValues*** 的关键字，或仅在已经匹配过的位置之前进行查找。
AdSeekBefore	仅在已经有过与 ***KeyValues*** 匹配的位置之前进行查找。
AdSeekFirstEQ	查找等于 ***KeyValues*** 的第一个关键字。
AdSeekLastEQ	查找等于 ***KeyValues*** 的最后一个关键字。

说明

如果基本提供者支持对 **Recordset** 对象使用索引，请结合 Index(See 1.1.4.6.34) 属性使用 **Seek** 方法。请使用 Supports(See

1.1.4.4.44) **(adIndex)** 方法判断基本提供者是否支持索引。

如果 **Seek** 没有找到想要的行，则不发生错误，并且行被定位于 EOF。请在执行该方法之前，将 **Index** 属性设置为所需索引。

该方法只能用于当 **Recordset** 对象的 CursorLocation(See 1.1.4.6.17) 属性的值不是 **adUseClient** 时。

1.1.4.4.43 SubmitChanges 方法 (RDS)

将本地缓存的可更新 **Recordset** 的挂起更改提交到在 Connect(See 1.1.4.6.13) 属性中指定的 ODBC(See 2.35) 数据源(See 2.18)中。

语法

DataControl.**SubmitChanges**

DataFactory.**SubmitChanges** *Connection, Recordset*

参数

DataControl 对象变量(See 2.34)，代表 **RDS.DataControl** 对象。

DataFactory 对象变量，代表 **RDSServer.DataFactory** 对象。

Connection **字符串**值，代表用 **RDS.DataControl** 对象的 **Connect** 属性创建的连接。

Recordset 对象变量，代表 **Recordset** 对象。

说明

在使用 **RDS.DataControl** 对象的 **SubmitChanges** 方法之前必须设置 Connect(See 1.1.4.6.13)、Server(See 1.1.4.6.55) 和 SQL(See 1.1.4.6.62) 属性。

如果在已经调用相同 **Recordset** 对象的 **SubmitChanges** 后调用 CancelUpdate(See 1.1.4.4.9) 方法，那么 **CancelUpdate** 调用将由于更改已提交而失败。

仅发送已更改的记录用于修改，更改或者全部成功或者全部失败。

只能通过默认的 **RDSServer.DataFactory** 对象使用 **SubmitChanges**。自定义业务对象(See 2.7)不能使用该方法。

1.1.4.4.44 Supports 方法 (ADO)

确定指定的 Recordset(See 1.1.4.2.10) 对象是否支持特定类型的功能。

语法

boolean = recordset.**Supports(*CursorOptions*)**

返回值

返回布尔型值，指示提供者是否支持 ***CursorOptions*** 参数所标识的所有功能。

参数

CursorOptions **长整型**表达式，包括一个或多个下列 **CursorOptionEnum** 值。

常量	说明
adAddNew	可使用 AddNew(See 1.1.4.4.1) 方法添加新记录。
adApproxPosition	可读取并设置 AbsolutePosition(See 1.1.4.6.2) 和 AbsolutePage(See 1.1.4.6.1) 的属性。
adBookmark	可使用 Bookmark(See 1.1.4.6.8) 属性获得对特定记录的访问。
adDelete	可以使用 Delete(See 1.1.4.4.20) 方法删除记录。
AdHoldRecords	可以检索多个记录或者更改下一个检索位置而不必提交所有挂起的更改。
AdMovePrevious	可使用 MoveFirst(See 1.1.4.4.29) 和 MovePrevious(See 1.1.4.4.29) 方法，以及 Move(See 1.1.4.4.28) 或 GetRows(See 1.1.4.4.25) 方法将当前记录位置向后移动而不必使用书签。
AdResync	通过 Resync(See 1.1.4.4.40) 方法，使用在基本的数据库中可见的数据更新游标。
AdUpdate	可使用 Update(See 1.1.4.4.45) 方法修改现有的数据。
AdUpdateBatch	可以使用批更新（UpdateBatch(See 1.1.4.4.46) 和 CancelBatch(See 1.1.4.4.7) 方法）将更改组传输给提供者。
AdIndex	可以使用 Index(See 1.1.4.6.34) 属性命名索引。
AdSeek	可以使用 Seek(See 1.1.4.4.42) 方法定位 Recordset 中的行。

说明

使用 **Supports** 方法确定 **Recordset** 对象所支持的功能类型。如果 **Recordset** 对象支持其相应常量在 ***CursorOptions*** 中的功能，那么 **Supports** 方法返回 **True**。否则返回 **False**。

> **注意** 尽管 Supports 方法可对给定的功能返回 True，但它不能保证提供者可以使功能在所有环境下均有效。Supports 方法只返回提供者是否支持指定的功能（假定符合某些条件）。例如，Supports 方法可能指示 Recordset 对象支持更新（即使游标基于多个表的合并），但并且某些列仍然无法更新。

1.1.4.4.45 Update 方法 (ADO)

保存对 Recordset(See 1.1.4.2.10) 对象的当前记录所做的所有更改。

语法

recordset.**Update** ***Fields, Values***

参数

Fields 可选。**变体型**，代表单个名称；或**变体型**数组，代表需要修改的字段（一个或多个）名称及序号位置。

Values 可选。**变体型**，代表单个值；或**变体型**数组，代表新记录中字段（单个或多个）值。

说明

使用 **Update** 方法保存自调用 AddNew(See 1.1.4.4.1) 方法或更改现有记录中任何字段值以来所作的所有更改。**Recordset** 对象必须支持更新。

要设置字段值，请进行下列某项操作：

- 为 Field(See 1.1.4.2.7) 对象的 Value(See 1.1.4.6.69) 属性赋值，并调用 **Update** 方法。
- 在 **Update** 调用中传送字段名和值作为参数。
- 在 **Update** 调用中传送字段名数组和值数组。

在使用字段和值的数组时，两个数组中必须有相等数量的元素，同时字段名的次序必须匹配字段值的次序。字段和值的数量及次序不匹配将产生错误。

如果 **Recordset** 对象支持批更新，那么可以在调用 UpdateBatch(See 1.1.4.4.46) 方法之前将一个或多个记录的多个改动缓存在本地。如果在调用 **UpdateBatch** 对象时正在编辑当前记录或者添加新的记录，那么 ADO 将自动调用 **Update** 方法以便在将批更改传送到提供者之前保存挂起的更改。

如果在调用 **Update** 方法之前移动出正在添加或编辑的记录，那么 ADO 将自动调用 **Update** 以便保存更改。如果希望取消对当前记录所做的任何更改或者放弃新添加的记录，则必须调用 CancelUpdate(See 1.1.4.4.9) 方法。

在调用 **Update** 方法后当前记录仍为当前记录。

如果设置了 Unique Table(See 1.1.4.7.2) 动态属性，并且 **Recordset** 是对多个表执行 JOIN 操作的结果，那么，**Update** 方法将无法更新多个表的任何主要关键字。此外，**Update** 只能更新在 **Unique Table** 属性中指定的表中的字段。

1.1.4.4.46 UpdateBatch 方法 (ADO)

将所有挂起的批更新写入磁盘。

语法

recordset.**UpdateBatch** ***AffectRecords***

参数

AffectRecords 可选，**AffectEnum** 值。决定 **UpdateBatch** 方法所影响的记录数目。可以为如下常量之一。

常量	说明
adAffectCurrent	只写入当前记录的挂起更改。
adAffectGroup	写入满足当前 Filter(See 1.1.4.6.28) 属性设置的记录所发生的挂起更改。必须将 **Filter** 属性设置为某个有效的预定义常量才能使用该选项。
adAffectAll	（默认值）。写入 **Recordset** 对象中所有记录的挂起更改，包括由于当前 **Filter** 属性设

	置而隐藏的任何记录。
adAffectAllChapters	写入所有子集的挂起更改。

说明

按批更新模式修改 **Recordset** 对象时，使用 **UpdateBatch** 方法可将 **Recordset** 对象中的所有更改传递到基本数据库。

如果 **Recordset** 对象支持批更新，那么可以将一个或多个记录的多重更改缓存在本地，然后再调用 **UpdateBatch** 方法。如果在调用 **UpdateBatch** 方法时正在编辑当前记录或者添加新的记录，那么在将批更新传送到提供者之前，ADO 将自动调用 Update(See 1.1.4.4.45) 方法保存对当前记录的所有挂起更改。

注意 只能对键集或静态游标使用批更新。

如果由于与基本的数据冲突而导致对所有或任意记录的传送更改失败（如其他用户已将记录删除），那么提供者将把警告返回给 Errors(See 1.1.4.3.1) 集合，并发生运行时错误。使用 **Filter** 属性 (**adFilterAffectedRecords**) 和 Status(See 1.1.4.6.65) 属性可以找到发生冲突的记录。

要取消所有挂起的批更新，请使用 CancelBatch(See 1.1.4.4.7) 方法。

如果设置了 Unique Table(See 1.1.4.7.2) 和 Update Resync(See 1.1.4.7.4) 动态属性，并且 **Recordset** 是对多个表执行 JOIN 操作的结果，那么，取决于 **Update Resync** 属性，执行 **UpdateBatch** 方法会隐性导致 Resync(See 1.1.4.4.40) 方法。

1.1.4.5 ADO 事件

事件	**说明**
BeginTransComplete 、CommitTransComplete 和 RollbackTransComplete (ConnectionEvent) 方法(See 1.1.4.5.1)	以下 **Event** 处理方法将在 **Connection** 对象的关联操作执行完成后进行调用。 **BeginTransComplete** 在 **BeginTrans** 操作后调用。 **CommitTransComplete** 在 **CommitTrans** 操作后调用。 **RollbackTransComplete** 在 **RollbackTrans** 操作后调用。
ConnectComplete 和 Disconnect (Connection Event) 方法(See 1.1.4.5.2)	在连接开始后调用 **ConnectComplete** 方法。 在连接结束后调用 **Disconnect** 方法。
EndOfRecordset (RecordsetEvent) 方法 (See 1.1.4.5.3)	当试图移动到超过 **Recordset** 末尾行时，调用 **EndOfRecordset** 方法。
ExecuteComplete (Connection Event) 方法(See 1.1.4.5.4)	命令执行完成之后，调用 **ExecuteComplete** 方法。
FetchComplete (RecordsetEvent) 方法 (See 1.1.4.5.5)	当在长异步操作中所有记录已经被恢复（获取）到 **Recordset** 之后，调用 **FetchComplete** 方法。
FetchProgress (Recordset Event) 方法 (See 1.1.4.5.6)	在长异步操作期间定期调用 **FetchProgress** 方法，以便报告当前有多少行已经被恢复（获取）到 **Recordset** 中。
InfoMessage (Connection Event) 方法	在 **ConnectionEvent** 操作期间一旦出现警告，则调用 **InfoMessage** 方法。

(See 1.1.4.5.7)	
onError (Event) 方法 (RDS)(See 1.1.4.5.8)	在操作期间一旦发生错误，则调用 **onError** 方法。
onReadyStateChange (Event) 方法 (RDS)(See 1.1.4.5.9)	一旦 **ReadyState** 属性的值发生更改，则调用该方法。
WillChangeField 和 FieldChangeComplete (RecordsetEvent) 方法(See 1.1.4.5.10)	在挂起操作更改 **Recordset** 中一个或多个 **Field** 对象的值之前，则调用 **WillChangeField** 方法。 在挂起操作更改一个或多个 **Field** 对象的值之后，则调用 **FieldChangeComplete** 方法。
WillChangeRecord 和 RecordChangeComplete (RecordsetEvent) 方法(See 1.1.4.5.11)	在 **Recordset** 中一个或多个记录（行）发生更改之前，将调用 **WillChangeRecord** 方法。 在一个或多个记录发生更改之后，将调用 **RecordChangeComplete** 方法。
WillChangeRecordset 和 RecordsetChangeComplete (RecordsetEvent) 方法(See 1.1.4.5.12)	在挂起操作更改 **Recordset** 之前调用 **WillChangeRecordset** 方法。 在 **Recordset** 已经更改之后，将调用 **RecordsetChangeComplete** 方法。
WillConnect (ConnectionEvent) 方法 (See 1.1.4.5.13)	在连接开始之前调用 **WillConnect** 方法。在挂起连接中使用的参数作为输入参数提供，并可以在方法返回之前更改。该方法可以返回取消挂起连接的请求。
WillExecute (ConnectionEvent) 方法(See 1.1.4.5.14)	**WillExecute** 方法在对该连接执行挂起命令之前调用，使用户能够检查和修改挂起执行的参数。该方法可以返回取消挂起连接的请求。
WillMove 和 MoveComplete (RecordsetEvent) 方法(See 1.1.4.5.15)	在挂起操作更改 **Recordset** 中的当前位置之前，调用 **WillMove** 方法。 **Recordset** 中的当前位置发生更改之后，调用 **MoveComplete** 方法。

1.1.4.5.1 BeginTransComplete、CommitTransComplete 和 RollbackTransComplete (ConnectionEvent) 方法 (ADO)

在 **Connection** 对象的关联操作完成执行之后，将调用这些方法。

- **BeginTransComplete** 在 BeginTrans(See 1.1.4.4.4) 操作之后调用。
- **CommitTransComplete** 在 CommitTrans(See 1.1.4.4.4) 操作之后调用。
- **RollbackTransComplete** 在 RollbackTrans(See 1.1.4.4.4) 操作之后调用。

语法

BeginTransComplete ***TransactionLevel, pError, adStatus, pConnection***

CommitTransComplete ***pError***, ***adStatus***, ***pConnection***

RollbackTransComplete ***pError***, ***adStatus***, ***pConnection***

参数

TransactionLevel 长整型，包含引发该事件的 **BeginTrans** 新事务级别。

pError **Error** 对象，说明当 ***EventStatusEnum*** 的值为 **adStatusErrorsOccurred** 时发生的错误；否则将不对它进行设置。

adStatus **EventStatusEnum** 状态值，调用这些方法中的任何一种方法时，如果引发事件的操作成功，则该参数设置为 **adStatusOK**。如果操作失败，则设置为 **adStatusErrorsOccurred**。

通过在方法返回前将该参数设置为 **adStatusUnwantedEvent**，这些方法可以避免后续通知。

pConnection 发生该事件所针对的 **Connection** 对象。

说明

在 Visual C++ 中多个 **Connections** 可以共享相同的事件处理方法。方法使用返回的 **Connection** 对象以确定引发事件的对象。

如果 Attributes(See 1.1.4.6.6) 属性设置为 **adXactCommitRetaining** 或 **adXactAbortRetaining**，那么在提交或回卷事务后将启动新的事务。使用 **BeginTransComplete** 事件处理程序例程可忽略除第一个以外的所有事务启动事件。

1.1.4.5.2 ConnectComplete 和 Disconnect (ConnectionEvent) 方法 (ADO)

ConnectComplete 方法在连接开始后调用。**Disconnect** 方法在连接结束后调用。

语法

ConnectComplete ***pError, adStatus, pConnection***

Disconnect ***adStatus, pConnection***

参数

pError **Error** 对象。它表示当 **adStatus** 的值为 **adStatusErrorsOccurred** 时发生的错误，否则将不对它进行设置。

adStatus ***EventStatusEnum*** 值，始终返回 **adStatusOK**。

当调用 **ConnectComplete** 时，如果 **WillConnect** 方法已经请求了取消挂起的连接，那么该参数设置为 **adStatusCancel**。

在返回任何方法之前，将该参数设置为 **adStatusUnwantedEvent** 以避免后续通知。但是，关闭和重新打开 **Connection** 会导致这些事件再次发生。

pConnection 该事件所针对的 **Connection** 对象。

1.1.4.5.3 EndOfRecordset (RecordsetEvent) 方法 (ADO)

如果试图移动到超过 **Recordset** 结尾的行时，将调用 **EndOfRecordset** 方法。

语法

EndOfRecordset ***fMoreData***, ***adStatus***, ***pRecordset***

参数

fMoreData **VARIANT_BOOL**，在处理该事件期间有可能将新记录追加到 ***pRecordset***。在 **EndOfRecordset** 返回前添加数据，然后将该参数设置为 **True** 以指示 **Recordset** 的新结尾。

adStatus **EventStatusEnum** 状态值。

当调用 **EndOfRecordset** 时，如果引发事件的操作成功，该参数设置为 **adStatusOK**。如果该方法无法请求取消引发该事件的操作，则设置为 **adStatusCantDeny**。

在 **EndOfRecordset** 返回前，将该参数设置为 **adStatusUnwantedEvent** 可避免后续的通知。

pRecordset **Recordset** 对象，发生该事件所针对的 **Recordset**。

说明

如果 **Recordset.MoveNext** 操作失败，则可能发生 **EndOfRecordset** 事件。

当用户可能因调用 **MoveNext** 而移过 ***pRecordset*** 末尾时，将调用该事件的处理程序。使用该方法用户可以从数据库中检索到更多记录并将其追加到 ***pRecordset*** 的结尾，在这种情况下，用户要将 ***fMoreData*** 设置为 VARIANT_TRUE，并从 **EndofRecordset** 返回。在此之后用户可以再次调用 **MoveNext** 以访问新检索到的记录。

1.1.4.5.4 ExecuteComplete (ConnectionEvent) 方法 (ADO)

该方法在命令执行完成后调用。

语法

ExecuteComplete ***RecordsAffected***, ***pError***, ***adStatus***, ***pCommand***, ***pRecordset***, ***pConnection***

参数

RecordsAffected 长整型，命令所影响的记录数目。

pError **Error** 对象，说明当 **adStatus** 值为 **adStatusErrorsOccured** 时所发生的错误，否则将不对它进行设置。

adStatus **EventStatusEnum** 状态值，当调用该方法时，如果引发事件的操作成功，该参数设置为 **adStatusOK**。如果操作失败，则设置为 **adStatusErrorsOccurred**。

在该方法返回前，将该参数设置为 **adStatusUnwantedEvent** 可避免后续的通知。

pCommand 被执行的 **Command** 对象（如果有）。

pRecordset **Recordset** 对象，执行的结果。该记录集可以为空。

pConnection **Connection** 对象，通过该连接执行命令。

说明

ExecuteComplete 事件可因 **Connection.Execute**、**Command.Execute**、**Recordset.Open** 或 **Recordset.NextRecordset** 而发生。

1.1.4.5.5 FetchComplete (RecordsetEvent) 方法 (ADO)

该方法在长异步操作中所有记录已经恢复（获取）到记录集之后调用。

语法

FetchComplete ***pError***, ***adStatus***, ***pRecordset***

语法

pError **Error** 对象，说明当 **adStatus** 值为 **adStatusErrorsOccurred** 时所发生的错误，否则将不对它进行设置。
adStatus **EventStatusEnum** 状态值，当调用该方法时，如果引发事件的操作成功，则参数设置为 **adStatusOK**，如果操作失败，则设置为 **adStatusErrorsOccurred**。
在此方法返回前，将该参数设置为 **adStatusUnwantedEvent** 可避免后续的通知。
pRecordset **Recordset** 对象，恢复记录的对象。

1.1.4.5.6 FetchProgress (RecordsetEvent) 方法 (ADO)

该方法在长时间的异步操作期间定期调用，以便报告当前已经恢复（获取）到 **Recordset** 中的行的数目。

语法

FetchProgress ***Progress***, ***MaxProgress***, ***adStatus***, ***pRecordset***

参数

Progress 长整型，当前已经恢复的记录数。
MaxProgress 长整型，可能恢复的最大记录数。
adStatus **EventStatusEnum** 状态值。
pRecordset **Recordset** 对象，正在恢复记录的对象。

1.1.4.5.7 InfoMessage (ConnectionEvent) 方法 (ADO)

在 **ConnectionEvent** 操作期间，一旦出现警告将调用该方法。

语法

InfoMessage ***pError***, ***adStatus***, ***pConnection***

参数

pError **Error** 对象。该参数包含被返回的任何错误。如果返回多个错误，则枚举 **Errors** 结合以查找它们。
adStatus **EventStatusEnum** 状态值。如果出现警告，*adStatus* 被设置为 **adStatusOK**，而 *pError* 包含警告。

在该方法返回前，将该参数设置为 **adStatusUnwantedEvent** 将避免后续的通知。

pConnection **Connection** 对象。发生警告所针对的连接。例如，在打开 **Connection** 对象或对 **Connection** 执行 **Command** 时，可能出现警告。

1.1.4.5.8 onError (Event) 方法 (RDS)

一旦在操作过程中发生错误，则调用该方法。

语法

onError ***SCode, Description, Source, CancelDisplay***

参数

SCode 整型，包含错误的状态代码。

Description 字符串，包含错误说明。

Source 字符串，包含引发错误的查询或命令。

CancelDisplay 布尔型，如果设置为 **True**，可使错误不在对话框中显示。

1.1.4.5.9 onReadyStateChange (Event) 方法 (RDS)

一旦 ReadyState(See 1.1.4.6.54) 属性值发生更改，则调用该方法。

语法

onReadyStateChange

参数

无

说明

当 **RDS.DataControl** 对象将数据异步获取到它的 **Recordset** 对象中时，**ReadyState** 属性反映该对象的进程。使用 **onReadyStateChange** 方法可随时监视 **ReadyState** 属性中所发生的更改。使用它比定期查看属性值更为有效。

1.1.4.5.10 WillChangeField 和 FieldChangeComplete (RecordsetEvent) 方法 (ADO)

WillChangeField 方法在挂起操作对 **Recordset** 中一个或多个 Field(See 1.1.4.2.7) 对象的值进行更改前调用。**FieldChangeComplete** 方法在一个或多个 **Field** 对象的值已经更改后调用。

语法

WillChangeField ***cFields***, ***Fields***, ***adStatus***, ***pRecordset***

FieldChangeComplete ***cFields***, ***Fields***, ***pError***, ***adStatus***, ***pRecordset***

参数

cFields 长整型，**Fields** 中的 **Field** 对象数目。

Fields 变体型数组，包含带有挂起发生更改的 **Field** 对象。

pError **Error** 对象，说明当 **adStatus** 值为 **adStatusErrorsOccured** 时所发生的错误，否则将不对它进行设置。

adStatus **EventStatusEnum** 状态值。

当调用 **WillChangeField** 时，如果引发事件的操作成功，该参数设置为 **adStatusOK**；如果该方法无法请求取消挂起操作，则设置为 **adStatusCantDeny**。

当调用 **FieldChangeComplete** 时，如果引发事件的操作成功，该参数设置为 **adStatusOK**；如果操作失败，则设置为 **adStatusErrorsOccurred**。

在 **WillChangeField** 返回前，将该参数设置为 **adStatusCancel** 可请求取消挂起的操作。

在 **FieldChangeComplete** 返回前，将该参数设置为 **adStatusUnwantedEvent** 避免后续的通知。

pRecordset **Recordset** 对象，发生该事件所针对的 **Recordset**。

说明

WillChangeField 或 **FieldChangeComplete** 事件可因下列 **Recordset** 操作而发生： **Value** 和带有字段及数组参数值的 **Update**。

1.1.4.5.11 WillChangeRecord 和 RecordChangeComplete (RecordsetEvent) 方法 (ADO)

WillChangeRecord 方法在 **Recordset** 中的一个或多个记录（行）更改之前调用。**RecordChangeComplete** 方法在一个或多个记录更改之后调用。

语法

WillChangeRecord ***adReason***, ***cRecords***, ***adStatus***, ***pRecordset***

RecordChangeComplete ***adReason***, ***cRecords***, ***pError***, ***adStatus***, ***pRecordset***

参数

adReason **EventReasonEnum** 值，指定该事件的原因。它的值可以是 **adRsnAddNew**、**adRsnDelete**、**adRsnUpdate**、**adRsnUndoUpdate**、**adRsnUndoAddNew**、**adRsnUndoDelete** 或 **adRsnFirstChange**。

cRecords 长整型，更改（影响）的记录数目。

pError **Error** 对象，说明当 **adStatus** 值为 **adStatusErrorsOccurred** 时所发生的错误，否则将不对它进行设置。

adStatus **EventStatusEnum** 状态值。

当调用 **WillChangeRecord** 时，如果引发事件的操作成功，该参数设置为 **adStatusOK**。如果该方法无法请求取消挂起的操作，则设置为 **adStatusCantDeny**。

当调用 **RecordChangeComplete** 时，如果引发事件的操作成功，则该参数设置为 **adStatusOK**。如果操作失败，则设置为

adStatusErrorsOccurred。

在 **WillChangeRecord** 返回前，将该参数设置为 **adStatusCancel** 可请求取消引发该事件的操作。

在 **RecordChangeComplete** 返回前，将该参数设置为 **adStatusUnwantedEvent** 可防止后续的通知。

pRecordset **Recordset** 对象，发生该事件所针对的 **Recordset**。

说明

WillChangeRecord 或 **RecordChangeComplete** 事件可因下列 **Recordset** 操作并针对行中第一个更改的字段发生：**Update**、**Delete**、**CancelUpdate**、**AddNew**、**UpdateBatch** 和 **CancelBatch**。**Recordset CursorType** 的值决定了是哪一个操作导致事件发生。

在 **WillChangeRecord** 事件期间，**Recordset Filter** 属性设置为 **adFilterAffectedRecords**。在处理事件时更改该属性是不合法的。

1.1.4.5.12 WillChangeRecordset 和 RecordsetChangeComplete (RecordsetEvent) 方法 (ADO)

WillChangeRecordset 方法在挂起操作更改 **Recordset** 之前调用。**RecordsetChangeComplete** 方法在 **Recordset** 更改后调用。

语法

WillChangeRecordset ***adReason***, ***adStatus***, ***pRecordset***

RecordsetChangeComplete ***adReason***, ***pError***, ***adStatus***, ***pRecordset***

参数

adReason **EventReasonEnum** 值，指定该事件的原因。它的值可以是 **adRsnReQuery**、**adRsnReSynch**、**adRsnClose**、**adRsnOpen**。

adStatus **EventStatusEnum** 状态值。

当调用 **WillChangeRecordset** 时，如果引发事件的操作成功，则该参数设置为 **adStatusOK**。如果该方法无法请求取消挂起的操作，则设置为 **adStatusCantDeny**。

当调用 **RecordsetChangeComplete** 时，如果引发事件的操作成功，则该参数设置为 **adStatusOK**；如果操作失败，则设置为 **adStatusErrorsOccurred**；如果与以前接受的 **WillChangeRecordset** 事件关联的操作已经取消，则设置为 **adStatusCancel**。

在 **WillChangeRecordset** 返回前，将该参数设置为 **adStatusCancel** 以请求取消挂起操作。

在 **WillChangeRecordset** 或 **RecordsetChangeComplete** 返回前，将该参数设置为 **adStatusUnwantedEvent** 可避免后续的通知。

pError **Error** 对象，说明当 **adStatus** 值为 **adStatusErrorsOccurred** 时所发生的错误，否则将不对它进行设置。

pRecordset **Recordset** 对象，发生该事件所针对的记录集。

说明

WillChangeRecordset 或 **RecordsetChangeComplete** 事件可因下列 **Recordset** 操作而发生：**Requery** 和 **Open**。

如果提供者不支持书签，则每次从提供者处检索新行时发生 **RecordsetChange** 事件通知。该事件的频率取决于 **RecordsetCacheSize** 属性。

1.1.4.5.13 WillConnect (ConnectionEvent) 方法 (ADO)

该方法在连接启动前调用。在挂起连接中使用的参数将作为输入参数提供，并可在方法返回之前进行更改。该方法可以返回取消挂起连接的请求。

语法

WillConnect ***ConnectionString, UserID, Password, Options, adStatus, pConnection***

参数

ConnectionString **字符串**，包含有关挂起连接的连接信息。

UserID **字符串**，包含挂起连接的用户名。

Password **字符串**，包含挂起连接的密码。

Options **长整型**值，指明提供者应如何计算 ***ConnectionString***。只能选择 **adAsyncOpen**。

adStatus ***EventStatusEnum*** 状态值。

当调用该方法时，如果引发事件的操作成功，则该参数设置为 **adStatusOK**。如果该方法无法请求取消挂起的操作，那么该参数设置为 **adStatusCantDeny**。

在该方法返回前，将该参数设置为 **adStatusUnwantedEvent** 以避免后续的通知。将该参数设置为 **adStatusCancel** 以请求引起取消该通知的连接操作。

pConnection 该事件通知所针对的 **Connection** 对象。该 **Connection** 无法由 **WillConnect** 事件更改。

说明

当调用该方法时，**ConnectionString**、**UserID**、**Password** 和 **Options** 参数设置为引发该事件的操作所建立的值。

当取消该方法时，将调用 **ConnectComplete**，并且它的 ***adStatus*** 参数将设置为 **adStatusErrorsOccurred**。

1.1.4.5.14 WillExecute (ConnectionEvent) 方法 (ADO)

该方法在对该连接执行挂起命令之前调用，并允许用户检查和修改挂起的执行参数。该方法可返回取消挂起命令的请求。

语法

WillExecute ***Source, CursorType, LockType, Options, adStatus, pCommand, pRecordset, pConnection***

参数

Source **字符串**，包含 SQL 命令或存储的过程名称。

CursorType **CursorTypeEnum**，包含用于将要打开的记录集的游标类型。使用该参数，可以在 **Recordset Open** 操作期间，将游标更改为任何类型。对于其他任何操作，将忽略 CursorType。

LockType **LockTypeEnum**，包含将要打开的记录集的锁定类型。使用该参数，可以在 **Recordset Open** 操作期间，更改对任何类型的锁定。对于其他任何操作，将忽略 LockType。

Options 长整型选项，可用于执行命令或打开记录集。

adStatus **EventStatusEnum** 状态值，在调用该方法时它可以是 **adStatusCantDeny** 或 **adStatusOK**。如果它是 **adStatusCantDeny**，

该方法可能无法请求取消挂起操作。

在此方法返回前，将该参数设置为 **adStatusUnwantedEvent** 可以避免后续的通知，或者设置为 **adStatusCancel** 以请求取消引发该事件的操作。

pCommand 该事件通知所针对的 **Command** 对象。

pRecordset 该事件通知所针对的 **Recordset** 对象。

pConnection 该事件通知所针对的 **Connection** 对象。

说明

WillExecute 事件可能因 **Connection.Execute**、**Command.Execute** 或 **Recordset.Open** 而发生。**pConnection** 参数应当始终包含对 **Connection** 对象的有效引用。如果事件的原因是 **Connection.Execute**，则 ***pRecordset*** 和 ***pCommand*** 参数被设置为 *Nothing*。如果事件的原因是 **Recordset.Open**，则 **pRecordset** 参数将引用 **Recordset** 对象并且 ***pCommand*** 参数被设置为 *Nothing*。如果事件的原因是 **Command.Execute**，则 ***pCommand*** 参数将引用 **Command** 对象并且 ***pRecordset*** 参数被设置为 *Nothing*。

1.1.4.5.15 WillMove 和 MoveComplete (RecordsetEvent) 方法 (ADO)

WillMove 方法在挂起操作更改 **Recordset** 中的当前位置前调用。**MoveComplete** 方法则在 **Recordset** 的当前位置更改后调用。

语法

WillMove ***adReason, adStatus, pRecordset***

MoveComplete ***adReason, pError, adStatus, pRecordset***

参数

adReason **EventReasonEnum** 值，指定该事件的原因。它的值可以是 **adRsnMoveFirst**、**adRsnMoveLast**、**adRsnMoveNext**、**adRsnMovePrevious**、**adRsnMove** 或 **adRsnRequery**。

pError **Error** 对象，说明当 **adStatus** 值为 **adStatusErrorsOccurred** 时所发生的错误，否则将不对它进行设置。

adStatus **EventStatusEnum** 状态值。

当调用 **WillMove** 时，如果引发事件的操作成功，则该参数设置为 **adStatusOK**。如果该方法无法请求取消挂起的操作，则设置为 **adStatusCantDeny**。

当调用 **MoveComplete** 时，如果引发事件的操作成功，则该参数设置为 **adStatusOK**。如果操作失败，则设置为 **adStatusErrorsOccurred**。

在 **WillMove** 返回前，将该参数设置为 **adStatusCancel** 可请求取消挂起的操作。在 **MoveComplete** 返回前，将该参数设置为 **adStatusUnwantedEvent** 可避免后续的通知。

pRecordset **Recordset** 对象。发生该事件所针对的记录集。

说明

WillMove 或 **MoveComplete** 事件可因下列 **Recordset** 操作而发生：**Open**、**Move**、**MoveFirst**、**MoveLast**、**MoveNext**、**MovePrevious**、**Bookmark**、**AddNew**、**Delete**、**Requery** 和 **Resync**。这些事件可能因下列属性而发生：**Filter**、**Index**、**AbsolutePage** 和 **AbsolutePosition**。如果子 **Recordset** 使 **Recordset** 事件被连接并且父 **Recordset** 被移动，则也会发生这些事件。

1.1.4.6 ADO 属性

属性	说明
AbsolutePage(See 1.1.4.6.1)	指定当前记录所在的页。
AbsolutePosition(See 1.1.4.6.2)	指定 **Recordset** 对象当前记录的序号位置。
ActiveCommand(See 1.1.4.6.3)	指示创建关联的 **Recordset** 对象的 **Command** 对象。
ActiveConnection(See 1.1.4.6.4)	指示指定的 **Command** 或 **Recordset** 对象当前所属的 **Connection** 对象。
ActualSize(See 1.1.4.6.5)	指示字段的值的实际长度。
Attributes(See 1.1.4.6.6)	指示对象的一项或多项特性。
BOF 和 EOF(See 1.1.4.6.7)	**BOF** 指示当前记录位置位于 **Recordset** 对象的第一个记录之前。 **EOF** 指示当前记录位置位于 **Recordset** 对象的最后一个记录之后。
Bookmark(See 1.1.4.6.8)	返回唯一标识 **Recordset** 对象中当前记录的书签，或者将 **Recordset** 对象的当前记录设置为由有效书签所标识的记录。
CacheSize(See 1.1.4.6.9)	指示缓存在本地内存中的 **Recordset** 对象的记录数。
CommandText(See 1.1.4.6.10)	包含要根据提供者发送的命令文本。
CommandTimeout(See 1.1.4.6.11)	指示在终止尝试和产生错误之前执行命令期间需等待的时间。
CommandType(See 1.1.4.6.12)	指示 **Command** 对象的类型。
Connect(See 1.1.4.6.13)	设置或返回对其运行查询和更新操作的数据库名称。
ConnectionString(See 1.1.4.6.14)	包含用于建立连接数据源的信息。
ConnectionTimeout(See 1.1.4.6.15)	指示在终止尝试和产生错误前建立连接期间所等待的时间。
Count(See 1.1.4.6.16)	指示集合中对象的数目。
CursorLocation(See 1.1.4.6.17)	设置或返回游标服务的位置。
CursorType(See 1.1.4.6.18)	指示在 **Recordset** 对象中使用的游标类型。
DataMember(See 1.1.4.6.19)	指定要从 **DataSource** 属性所引用的对象中检索的数据成员的名称。
DataSource(See 1.1.4.6.20)	指定所包含的数据将被表示为 **Recordset** 对象的对象。
DefaultDatabase(See 1.1.4.6.21)	指示 **Connection** 对象的默认数据库。
DefinedSize(See 1.1.4.6.22)	指示 **Field** 对象所定义的大小。
Description(See 1.1.4.6.23)	描述 **Error** 对象。
Direction(See 1.1.4.6.24)	指示 **Parameter** 表示的是输入参数、输出参数还是既是输出又是输入参数，或

	该参数是否为存储过程返回的值。
EditMode(See 1.1.4.6.25)	指示当前记录的编辑状态。
ExecuteOptions (RDS)(See 1.1.4.6.26)	指示是否启用异步执行。
FetchOptions(See 1.1.4.6.27)	设置或返回异步获取的类型。
Filter(See 1.1.4.6.28)	指示 **Recordset** 的数据筛选条件。
FilterColumn (RDS)(See 1.1.4.6.29)	设置或返回计算筛选条件的列。
FilterCriterion (RDS)(See 1.1.4.6.30)	设置或返回在筛选值中使用的计算操作符。
FilterValue (RDS)(See 1.1.4.6.31)	设置或返回用于筛选记录的值。
Handler (RDS)(See 1.1.4.6.32)	设置或返回包含扩展 **RDSServer.DataFactory** 功能的服务器端自定义程序（处理程序）的名称的字符串，以及处理程序所用的任何参数，它们均由逗号 (",") 分隔。
HelpContext(See 1.1.4.6.33) 和 HelpFile(See 1.1.4.6.33)	指示与 **Error** 对象关联的帮助文件和主题。 **HelpContextID** - 返回帮助文件中主题的、按**长整型**值返回的上下文 ID。 **HelpFile** - 返回字符串，用于计算帮助文件的完整分解路径。
Index(See 1.1.4.6.34)	指示对 **Recordset** 对象当前生效的索引的名称。
InternetTimeout (RDS)(See 1.1.4.6.35)	指示请求超时前将等待的毫秒数。
IsolationLevel(See 1.1.4.6.36)	指示 **Connection** 对象的隔离级别。
LockType(See 1.1.4.6.37)	指示编辑过程中对记录使用的锁定类型。
MarshalOptions(See 1.1.4.6.38)	指示要被调度返回服务器的记录。
MaxRecords(See 1.1.4.6.39)	指示通过查询返回 **Recordset** 的记录的最大数目。
Mode(See 1.1.4.6.40)	指示用于更改 **Connection** 中数据的可用权限。
Name(See 1.1.4.6.41)	指示对象的名称。
NativeError(See 1.1.4.6.42)	指示针对给定 **Error** 对象的特定提供者的错误代码。
Number(See 1.1.4.6.43)	指示用于唯一标识 **Error** 对象的数字。
NumericScale(See 1.1.4.6.44)	指示 **Parameter** 或 **Field** 对象中数字值的范围。
Optimize(See 1.1.4.6.45)	指示是否应该在该字段上创建索引。
OriginalValue(See 1.1.4.6.46)	指示发生任何更改前已在记录中存在的 **Field** 的值。
PageCount(See 1.1.4.6.47)	指示 **Recordset** 对象包含的数据页数。
PageSize(See 1.1.4.6.48)	指示 **Recordset** 中一页所包含的记录数。

Precision(See 1.1.4.6.49)	指示在 **Parameter** 对象中数字值或数字 **Field** 对象的精度。
Prepared(See 1.1.4.6.50)	指示执行前是否保存命令的编译版本。
Provider(See 1.1.4.6.51)	指示 **Connection** 对象提供者的名称。
RecordCount(See 1.1.4.6.52)	指示 **Recordset** 对象中记录的当前数目。
Recordset and SourceRecordset (RDS)(See 1.1.4.6.53)	指示从自定义业务对象中返回的 **ADOR.Recordset** 对象。
ReadyState (RDS)(See 1.1.4.6.54)	在 **RDS.DataControl** 对象获取数据到它的 **Recordset** 对象中时反映其进度。
Server (RDS)(See 1.1.4.6.55)	设置或返回 Internet Information Server (IIS) 名称和通讯协议。
Size(See 1.1.4.6.56)	指示 **Parameter** 对象的最大大小（按字节或字符）。
Sort(See 1.1.4.6.57)	指定一个或多个 **Recordset** 以之排序的字段名，并指定按升序还是降序对字段进行排序。
SortColulmn (RDS)(See 1.1.4.6.58)	设置或返回记录以之排序的列。
SortDirection (RDS)(See 1.1.4.6.59)	设置或返回用于指示排序顺序是升序还是降序的布尔型值。
Source (ADO Error)(See 1.1.4.6.60)	指示产生错误的原始对象或应用程序的名称。
Source (ADO Recordset) (See 1.1.4.6.61)	指示 **Recordset** 对象（**Command** 对象、SQL 语句、表的名称或存储过程）中数据的来源。
SQL (RDS)(See 1.1.4.6.62)	设置或返回用于检索 **Recordset** 的查询字符串。
SQLState(See 1.1.4.6.63)	指示给定 **Error** 对象的 SQL 状态。
State(See 1.1.4.6.64)	对所有可应用对象，说明其对象状态是打开或是关闭。 对执行异步方法的 **Recordset** 对象，说明当前的对象状态是连接、执行或是获取。
Status(See 1.1.4.6.65)	指示有关批更新或其他大量操作的当前记录的状态。
StayInSync(See 1.1.4.6.66)	在分级 **Recordset** 对象中，指示当父行位置更改时，对基本子记录（即“子集”）的引用是否更改。
Type(See 1.1.4.6.67)	指示 **Parameter**、**Field** 或 **Property** 对象的操作类型或数据类型。
UnderlyingValue(See 1.1.4.6.68)	指示数据库中 **Field** 对象的当前值。
Value(See 1.1.4.6.69)	指示赋给 **Field**、**Parameter** 或 **Property** 对象的值。
Version(See 1.1.4.6.70)	指示 ADO 版本号。

1.1.4.6.1 AbsolutePage 属性 (ADO)

指定当前记录所在的页。

设置和返回值

设置或返回从 1 到 Recordset(See 1.1.4.2.10) 对象 (PageCount(See 1.1.4.6.47)) 所含页数的**长整型**值，或者返回以下常量。

常量	说明
AdPosUnknown	Recordset 为空，当前位置未知，或者提供者不支持 AbsolutePage 属性。
AdPosBOF	当前记录指针位于 BOF（即 BOF 属性为 True）。
AdPosEOF	当前记录指针位于 EOF（即 EOF 属性为 True）。

说明

使用 **AbsolutePage** 属性可识别当前记录所在的页码。使用 PageSize(See 1.1.4.6.48) 属性可将 **Recordset** 对象逻辑划分为一系列的页，每页的记录数等于 **PageSize**（最后页除外，该页记录数较少）。提供者必须支持该属性的相应功能才能使用该属性。

与 AbsolutePosition(See 1.1.4.6.2) 属性一样，**AbsolutePage** 从 1 开始并在当前记录为 **Recordset** 中的第一个记录时等于 1。设置该属性可移动到特定页的第一个记录。从 **PageCount** 属性中可获得总页数。

1.1.4.6.2 AbsolutePosition 属性 (ADO)

指定 Recordset(See 1.1.4.2.10) 对象当前记录的序号位置。

设置和返回值

设置或返回从 1 到 **Recordset** 对象 (PageCount(See 1.1.4.6.47)) 所含页数的**长整型**值，或者返回以下常量。

常量	说明
AdPosUnknown	Recordset 为空，当前位置未知，或者提供者不支持 AbsolutePage 属性。
AdPosBOF	当前记录指针位于 BOF（即 BOF 属性为 True。）
adPosEOF	当前记录指针位于 EOF（即 EOF 属性为 True。）

说明

使用 **AbsolutePosition** 属性可根据其在 **Recordset** 中的序号位置移动到记录，或确定当前记录的序号位置。提供者必须支持该属性的相应功能才能使用该属性。

同 AbsolutePage(See 1.1.4.6.1) 属性一样，**AbsolutePosition** 从 1 开始，并在当前记录为 **Recordset** 中的第一个记录时等于 1。从 **RecordCount** 属性可获得 **Recordset** 对象的总记录数。

设置 AbsolutePosition 属性时，即使该属性指向位于当前缓存中的记录，ADO 也将使用以指定的记录开始的新记录组重新加载缓存。CacheSize(See 1.1.4.6.9) 属性决定该记录组的大小。

注意 不能将 AbsolutePosition 属性作为替代的记录编号使用。删除前面的记录时，给定记录的当前位置将发生改变。如果 Recordset 对象被重新查询或重新打开，则无法保证给定记录有相同的 AbsolutePosition。书签仍然是保持和返回给定位置的推荐方式，并且在所有类型的 Recordset 对象的定位时是唯一的方式。

1.1.4.6.3 ActiveCommand 属性 (ADO)

指示创建关联的 Recordset(See 1.1.4.2.10) 对象的 Command(See 1.1.4.2.1) 对象。

返回值

返回包含 **Command** 对象的**变体型**。默认为 **Null** 对象引用。

说明

ActiveCommand 属性为只读。
如果没有使用 **Command** 对象创建当前 **Recordset**，将返回 **Null** 对象引用。
如果您只有结果 **Recordset** 对象，则可使用该属性查找关联的 **Command** 对象。

1.1.4.6.4 ActiveConnection 属性 (ADO)

指示指定的 Command(See 1.1.4.2.1) 或 Recordset(See 1.1.4.2.10) 对象当前所属的 Connection(See 1.1.4.2.2) 对象。

设置和返回值

设置或返回包含了定义连接或 **Connection** 对象的字符串。默认情况下为 **Null** 对象引用。

说明

使用 **ActiveConnection** 属性可确定在其上将执行指定 **Command** 对象或打开指定 **Recordset** 的 **Connection** 对象。

命令

对于 **Command** 对象，**ActiveConnection** 属性为读/写。
在将该属性设置为打开的 **Connection** 对象或有效连接字符串之前，试图调用 **Command** 对象的 Execute(See 1.1.4.4.21) 方法将产生错误。

Microsoft Visual Basic 将 ActiveConnection 属性设置为 Nothing 可使 Command 对象与当前 Connection 脱离关联，并使提供者释放数据源上所有关联的资源。然后，可以使 Command 对象与相同的 Connection 对象或其他 Connection 对象关联。某些提供者允许将该属性设置从一个 Connection 更改到另一个 Connection，而不必首先将该属性设置为 Nothing。

如果 **Command** 对象的 Parameter(See 1.1.4.3.3) 集合包含提供者提供的参数，那么假如将 **ActiveConnection** 属性设置为 Nothing 或设置为其他 **Connection** 对象，将清除集合。如果手工创建 Parameter(See 1.1.4.2.8) 对象并使用这些参数填充 **Command** 对象的 **Parameters** 集合，则将 **ActiveConnection** 属性设置为 Nothing 或其他 **Connection** 对象不会影响 **Parameters** 集合。

关闭与 **Command** 对象相关联的 **Connection** 对象将把 **ActiveConnection** 属性设置为 **Nothing**。将该属性设置为已关闭的 **Connection** 对象将产生错误。

Recordset

对于打开的 **Recordset** 对象或其 Source(See 1.1.4.6.61) 属性被设置为有效 **Command** 对象的 **Recordset** 对象，**ActiveConnection** 属性为只读。否则，该属性为读/写。

可以将该属性设置为有效的 **Connection** 对象，或设置为有效的连接字符串。这时，提供者可使用该定义创建新的 **Connection** 对象，并打开连接。另外，提供者可以将该属性设置为新的 **Connection** 对象，以便向您提供访问扩展错误信息的 **Connection** 对象或执行其他命令。

如果使用 Open(See 1.1.4.4.33) 方法的 ***ActiveConnection*** 参数打开 **Recordset** 对象，**ActiveConnection** 属性将继承该参数的值。

如果将 **Recordset** 对象的 **Source** 属性设置为有效 **Command** 对象变量，**Recordset** 的 **ActiveConnection** 属性将继承 **Command** 对象的 **ActiveConnection** 属性的设置。

远程数据服务用法 在客户端 (ADOR) **Recordset** 对象上使用时，只能将该属性设置为连接字符串或（Microsoft Visual Basic 或 VBScript 中）**Nothing**

1.1.4.6.5 ActualSize 属性 (ADO)

指示字段的值的实际长度。

设置和返回值

返回**长整型**值。某些提供者允许设置该属性以便为 BLOB 数据预留空间，在此情况下默认值为 0。

说明

使用 **ActualSize** 属性可返回 Field(See 1.1.4.2.7) 对象值的实际长度。对于所有字段，**ActualSize** 属性为只读。如果 ADO 无法确定 Field 对象值的实际长度，**ActualSize** 属性将返回 **adUnknown**。

如以下范例所示，**ActualSize** 和 DefinedSize(See 1.1.4.6.22) 属性有所不同：**adVarChar** 声明类型且最大长度为 50 个字符的 **Field** 对象将返回为 50 的 **DefinedSize** 属性值，但是返回的 **ActualSize** 属性值是当前记录的字段中存储的数据的长度。

1.1.4.6.6 Attributes 属性 (ADO)

指示对象的一项或多项特性。

设置和返回值

设置或返回**长整型**值。

对于 Connection(See 1.1.4.2.2) 对象，**Attributes** 属性为读/写，并且其值可能为以下任意一个或多个 **XactAttributeEnum** 值的和（默认为零）。

常量	说明
AdXactCommitRetaining	执行保留的提交，即通过自动调用 CommitTrans(See 1.1.4.4.4) 启动新事务。并非所有提供者都支持该常量。
AdXactAbortRetaining	执行保留的中止，即通过自动调用 RollbackTrans(See 1.1.4.4.4) 启动新事务。并非所有提供者都支持该常量。

对于 Parameter(See 1.1.4.2.8) 对象，**Attributes** 属性为读/写，并且其值可能为以下任意一个或多个 **ParameterAttributesEnum** 值的和。

常量	说明
AdParamSigned	默认值。指示该参数接受带符号的值。
AdParamNullable	指示该参数接受 **Null** 值。
AdParamLong	指示该参数接受长二进制数据。

对于 Field(See 1.1.4.2.7) 对象，**Attributes** 属性为只读，其值可能为以下任意一个或多个 **FieldAttributeEnum** 值的和。

常量	说明
adFldMayDefer	指示字段被延迟，即不从拥有整个记录的数据源检索字段值，仅在显式访问这些字段时才进行检索。
adFldUpdatable	指示可以写入该字段。
adFldUnknownUpdatable	指示提供者无法确定是否可以写入该字段。
adFldFixed	指示该字段包含定长数据。
adFldIsNullable	指示该字段接受 **Null** 值。
adFldMayBeNull	指示可以从该字段读取 **Null** 值。
adFldLong	指示该字段为长二进制字段。并指示可以使用 AppendChunk(See 1.1.4.4.3) 和 GetChunk(See 1.1.4.4.24) 方法。
adFldRowID	指示字段包含持久的行标识符，该标识符无法被写入并且除了对行进行标识（如记录号、唯一标识符等）外不存在有意义的值。
adFldRowVersion	指示该字段包含用来跟踪更新的某种时间或日期标记。
adFldCacheDeferred	指示提供者缓存了字段值，并已完成随后对缓存的读取。

对于 Property(See 1.1.4.2.9) 对象，**Attributes** 属性为只读，并可能是以下任意一个或多个 **PropertyAttributesEnum** 值的和：

常量	说明

adPropNotSupported	指示提供者不支持该属性。
adPropRequired	指示数据源初始化之前用户必须指定该属性的值。
adPropOptional	指示数据源初始化之前用户不必为该属性指定值。
adPropRead	指示用户可以读取该属性。
adPropWrite	指示用户可以设置该属性。

说明

使用 **Attributes** 属性可设置或返回 **Connection** 对象、**Paramete** 对象、**Field** 对象或 **Property** 对象的特性。
设置多个属性时，可以将相应的常量相加。如果将属性值设置为包括不兼容常量的总和，那么将产生错误。

远程数据服务用法 该属性在客户端的 Connection 对象上无效。

1.1.4.6.7 BOF、EOF 属性 (ADO)

- **BOF** 指示当前记录位置位于 **Recordset** 对象的第一个记录之前。
- **EOF** 指示当前记录位置位于 **Recordset** 对象的最后一个记录之后。

返回值

BOF 和 **EOF** 属性返回布尔型值。

说明

使用 **BOF** 和 **EOF** 属性可确定 Recordset(See 1.1.4.2.10) 对象是否包含记录，或者从一个记录移动到另一个记录时是否超出 **Recordset** 对象的限制。
如果当前记录位于第一个记录之前，**BOF** 属性将返回 **True** (-1)，如果当前记录为第一个记录或位于其后则将返回 **False** (0)。
如果当前记录位于 **Recordset** 对象的最后一个记录之后 **EOF** 属性将返回 **True**，而当前记录为 **Recordset** 对象的最后一个记录或位于其前，则将返回 **False**。
如果 **BOF** 或 **EOF** 属性为 **True**，则没有当前记录。
如果打开没有记录的 **Recordset** 对象，**BOF** 和 **EOF** 属性将设置为 **True**，而 **Recordset** 对象的 RecordCount(See 1.1.4.6.52) 属性设置为零。打开至少包含一条记录的 **Recordset** 对象时，第一条记录为当前记录，而 **BOF** 和 **EOF** 属性为 **False**。
如果删除 **Recordset** 对象中保留的最后记录，**BOF** 和 **EOF** 属性将保持 **False**，直到重新安排当前记录。
以下表格说明不同 **BOF** 和 **EOF** 属性组合所允许的 **Move** 方法。

	MoveFirst, MoveLast	MovePrevious, Move < 0	Move 0	MoveNext, Move > 0
BOF=True, EOF=False	允许	错误	错误	允许

BOF=False, EOF=True	允许	允许	错误	错误
同时为 **True**	错误	错误	错误	错误
同时为 **False**	允许	允许	允许	允许

允许使用 **Move** 方法并不能保证该方法成功定位记录，只是意味着调用指定的 **Move** 方法不会产生错误。

下表说明当调用各种 **Move** 方法但未成功定位记录时 **BOF** 和 **EOF** 属性设置所发生的情况。

	BOF	**EOF**
MoveFirst, MoveLast	设置为 **True**	设置为 **True**
Move 0	没有变化	没有变化
MovePrevious, Move < 0	设置为 **True**	没有变化
MoveNext, Move > 0	没有变化	设置为 **True**

1.1.4.6.8 Bookmark 属性 (ADO)

返回唯一标识 Recordset(See 1.1.4.2.10) 对象中当前记录的书签，或者将 **Recordset** 对象的当前记录设置为由有效书签所标识的记录。

设置和返回值

设置或返回计算有效书签的**变体型**表达式。

说明

使用 **Bookmark** 属性可保存当前记录的位置并随时返回到该记录。书签只能在支持书签功能的 **Recordset** 对象中使用。

打开 **Recordset** 对象时，其每个记录都有唯一的书签。要保存当前记录的书签，请将 **Bookmark** 属性的值赋给一个变量。移动到其他记录后要快速返回到该记录，请将该 **Recordset** 对象的 **Bookmark** 属性设置为该变量的值。

用户可能无法查看书签的值，也同样无法对书签直接进行比较（指向同一记录的两个书签的值可能不同）。

如果使用 Clone(See 1.1.4.4.11) 方法创建 **Recordset** 的一个副本，则原始的和复制的 **Recordset** 对象 **Bookmark** 属性设置相同并可以替换使用。但是，无法替换使用不同 **Recordset** 对象的书签，即使这些书签是通过同一数据源或命令创建的。

远程数据服务用法 在客户端 (ADOR) **Recordset** 对象上使用时，**Bookmark** 属性始终有效。

1.1.4.6.9 CacheSize 属性 (ADO)

指示缓存在本地内存中的 Recordset(See 1.1.4.2.10) 对象的记录数。

设置和返回值

设置或返回必须大于 0 的**长整型**值。默认值为 1。

说明

使用 **CacheSize** 属性可控制提供者在缓存中所保存的记录的数目，并可控制一次恢复到本地内存的记录数。例如，如果 **CacheSize** 为 10，首次打开 **Recordset** 对象后，提供者将前面 10 个记录调入本地内存。当在 **Recordset** 对象中移动时，提供者返回本地内存缓冲区中的数据；一旦移动超过缓存中最后的记录，提供者便将数据源中随后的 10 个记录恢复到缓存。

可以在 **Recordset** 对象的存活期调整该属性的值，但是更改该值只影响随后从数据源调入缓存的记录数。只更改属性值将不会更改缓存中的当前内容。

如果要检索的记录较 **CacheSize** 指定的少，提供者将返回其余的记录，不会产生错误。

不允许将 **CacheSize** 设置为零，否则将返回错误。

从缓存恢复的记录不反映其他用户对数据源同时所作的更改。如需强行对所有缓存数据进行更新，请使用 Resync(See 1.1.4.4.40) 方法。

1.1.4.6.10 CommandText 属性 (ADO)

包含要根据提供者发送的命令的文本。

设置和返回值

设置或返回包含提供者命令（如 SOL 语句、表格名称或存储的过程调用）的字符串值。默认值为 ""（零长度字符串）。

说明

使用 **CommandText** 属性可设置或返回 Command(See 1.1.4.2.1) 对象的文本。通常，该属性为 SQL 语句，但也可以是提供者识别的任何其他类型的命令语句（如存储的过程调用）。SQL 语句必须是提供者查询处理程序支持的特定语法或版本。

如果设置 **CommandText** 属性时将 **Command** 对象的 Prepared(See 1.1.4.6.50) 属性设置为 **True**，并将 **Command** 对象绑定到打开的连接，则在调用 **Execute** 或 **Open** 方法时 ADO 将准备查询（即，提供者保存已编译的查询格式）。

取决于 CommandType(See 1.1.4.6.12) 属性的设置，ADO 可能改变 **CommandText** 属性。请随时阅读 **CommandText** 属性查看在执行过程中 ADO 将要使用的实际命令文本。

1.1.4.6.11 CommandTimeout 属性 (ADO)

指示在终止尝试和产生错误之前执行命令期间需等待的时间。

设置和返回值

设置或返回**长整型**值，该值指示等待命令执行的时间（单位为秒）。默认值为 30。

说明

使用 Connection(See 1.1.4.2.2) 对象或 Command(See 1.1.4.2.1) 上的 **CommandTimeout** 属性，允许由于网络拥塞或服务器负载过重产生的延迟而取消 **Execute** 方法调用。如果在 **CommandTimeout** 属性中设置的时间间隔内没有完成命令执行，将产生错误，然后 ADO 将取消该命令。如果将该属性设置为零，ADO 将无限期等待直到命令执行完毕。请确保正在写入代码的提供者和数据

源支持 **CommandTimeout** 功能。

Connection 对象的 **CommandTimeout** 设置不会对相同 **Connection** 上 **Command** 对象的 **CommandTimeout** 设置产生影响，即 **Command** 对象的 **CommandTimeout** 属性不继承 **Connection** 对象的 **CommandTimeout** 的值。

在 **Connection** 对象上，打开 **Connection** 后，**CommandTimeout** 属性将保持读/写。

1.1.4.6.12 CommandType 属性 (ADO)

指示 Command(See 1.1.4.2.1) 对象的类型。

设置和返回值

设置或返回以下某个 **CommandTypeEnum** 值。

常量	说明
AdCmdText	将 CommandText(See 1.1.4.6.10) 作为命令或存储过程调用的文本化定义进行计算。
AdCmdTable	将 **CommandText** 作为其列全部由内部生成的 SQL 查询返回的表格的名称进行计算。
AdCmdTableDirect	将 **CommandText** 作为其列全部返回的表格的名称进行计算。
AdCmdStoredProc	将 **CommandText** 作为存储过程名进行计算。
AdCmdUnknown	默认值。**CommandText** 属性中的命令类型未知。
adCmdFile	将 **CommandText** 作为持久 Recordset(See 1.1.4.2.10) 文件名进行计算。
AdExecuteNoRecords	指示 **CommandText** 为不返回行的命令或存储过程（例如，插入数据的命令）。如果检索任意行，则将丢弃这些行且并不返回。它总是与 **adCmdText** 或 **adCmdStoredProc** 进行组合。

说明

使用 **CommandType** 属性可优化 **CommandText** 属性的计算。

如果 **CommandType** 属性的值等于 **adCmdUnknown**（默认值），系统的性能将会降低，因为 ADO 必须调用提供者以确定 **CommandText** 属性是 SQL 语句、还是存储过程或表格名称。如果知道正在使用的命令的类型，可通过设置 **CommandType** 属性指令 ADO 直接转到相关代码。如果 **CommandType** 属性与 **CommandText** 属性中的命令类型不匹配，调用 Execute(See 1.1.4.4.21) 方法时将产生错误。

adExecuteNoRecords 常量通过最小化内部处理来提高性能。该常量不独立使用，它总是与 **adCmdText** 或 **adCmdStoredProc** 组合（如 adCmdText+adExecuteNoRecords）一起使用。如果与 **Recordset.Open** 一起使用 **adExecuteNoRecords**，或者该方法使用 **Command** 对象都将产生错误。

1.1.4.6.13 Connect 属性 (RDS)

设置或返回对其运行查询和更新操作的数据库名称。

可以在设计时(See 2.20)在 **RDS.DataControl** 对象的 OBJECT 标记中设置 **Connect** 属性，或者在运行时(See 2.43)在脚本代码（如

VBScript）中设置 **Connect** 属性。

语法

设计时：**<PARAM NAME="Connect" VALUE="DSN=***DSNName***;UID=***usr***;PWD=***pw***;">**

运行时：*DataControl*.**Connect** = **"DSN=***DSNName***;UID=***usr***;PWD=***pw***;"**

参数

DSNName 指定系统数据源(See 2.18)名称的**字符串**，该系统数据源名称标识指定的数据库。

usr 表示服务器上合法用户帐号的**字符串**。

pw 表示用户帐号合法密码的**字符串**。

DataControl 表示 **RDS.DataControl** 对象的对象变量(See 2.34)。

1.1.4.6.14 ConnectionString 属性 (ADO)

包含用于建立连接数据源的信息。

设置和返回值

设置或返回**字符串**值。

说明

使用 **ConnectionString** 属性，通过传递包含一系列由分号分隔的 *argument = value* 语句的详细连接字符串可指定数据源。

ADO 支持 **ConnectionString** 属性的四个参数，任何其他参数将直接传递到提供者而不经过 ADO 处理。ADO 支持的参数如下：

参数	说明
Provider=	指定用来连接的提供者名称。
File Name=	指定包含预先设置连接信息的特定提供者的文件名称（例如，持久数据源对象）。
Remote Provider=	指定打开客户端连接时使用的提供者名称。（仅限于远程数据服务）
Remote Server=	指定打开客户端连接时使用的服务器的路径名称。（仅限于远程数据服务）

设置 **ConnectionString** 属性并打开 Connection(See 1.1.4.2.2) 对象后，提供者可以更改属性的内容，例如通过将 ADO 定义的参数名映射到其提供者等价项来更改属性的内容。

ConnectionString 属性将自动继承用于 Open(See 1.1.4.4.32) 方法的 ***ConnectionString*** 参数的值，以便在 **Open** 方法调用期间覆盖当前的 **ConnectionString** 属性。

由于 ***File Name*** 参数使得 ADO 加载关联的提供者，因此无法传递 ***Provider*** 和 ***File Name*** 参数。

连接关闭时 **ConnectionString** 属性为读/写，打开时其属性为只读。

远程数据服务用法 在客户端 Connection 对象上使用该服务时，ConnectionString 属性只能包括 *Remote Provider* 和 *Remote Server* 参数。

1.1.4.6.15 ConnectionTimeout 属性 (ADO)

指示在终止尝试和产生错误前建立连接期间所等待的时间。

设置和返回值

设置或返回指示等待连接打开的时间的**长整型**值（单位为秒）。默认值为 15。

说明

如果由于网络拥塞或服务器负载过重导致的延迟使得必须放弃连接尝试时，请使用 Connection(See 1.1.4.2.2) 对象的 **ConnectionTimeout** 属性。如果打开连接前所经过的时间超过 **ConnectionTimeout** 属性上设置的时间，将产生错误，并且 ADO 将取消该尝试。如果将该属性设置为零，ADO 将无限等待直到连接打开。请确认正在对其编写代码的提供者会支持 **ConnectionTimeout** 功能。

连接关闭时 **ConnectionTimeout** 属性为读/写，而打开时其属性为只读。

1.1.4.6.16 Count 属性 (ADO)

指示集合中对象的数目。

返回值

返回**长整型**值。

说明

使用 **Count** 属性可确定给定集合中对象的数目。

因为集合成员的编号从零开始，因此应该始终以零成员开头且以 Count 属性的值减 1 结尾而进行循环编码。如果正在使用 Microsoft® Visual Basic® 并想在不使用 **Count** 属性的情况下在集合的成员中循环操作，请使用 **For Each...Next** 命令。

如果 **Count** 属性为零，集合中将不存在对象。

1.1.4.6.17 CursorLocation 属性 (ADO)

设置或返回游标服务的位置。

设置和返回值

设置或返回可设置为以下某个常量的**长整型**值。

常量	说明
adUseNone	没有使用游标服务。（该常量已过时并且只为了向后兼容才出现）。
adUseClient	使用由本地游标库提供的客户端游标。本地游标服务通常允许使用的许多功能可能是驱动程

	序提供的游标无法使用的，因此使用该设置对于那些将要启用的功能是有好处的。**AdUseClient** 具有向后兼容性，也支持同义的 **adUseClientBatch**。
adUseServer	默认值。使用数据提供者的或驱动程序提供的游标。这些游标有时非常灵活，对于其他用户对数据源所作的更改具有额外的敏感性。但是，Microsoft Client Cursor Provider（如已断开关联的记录集）的某些功能无法由服务器端游标模拟，通过该设置将无法使用这些功能。

说明

该属性允许在可用于提供者的各种游标库中进行选择。通常，可以选择使用客户端游标库或位于服务器上的某个游标库。

该属性设置仅对属性已经设置后才建立的连接有影响。更改 **CursorLocation** 属性不会影响现有的连接。

对于 **Connection** 或关闭的 **Recordset** 该属性为读/写，而对打开的 **Recordset** 该属性为只读。

由 **Execute** 方法返回的游标继承该设置。**Recordset** 将自动从与之关联的连接中继承该设置。

远程数据服务用法 当用于客户端 (ADOR) Recordset 或 Connection 对象时，只能将 CursorLocation 属性设置为 adUseClient。

1.1.4.6.18 CursorType 属性 (ADO)

指示在 Recordset(See 1.1.4.2.10) 对象中使用的游标类型。

设置和返回值

设置或返回以下某个 **CursorTypeEnum** 值。

常量	说明
AdOpenForwardOnly	仅向前游标，默认值。除了只能在记录中向前滚动外，与静态游标相同。当只需要在记录集中单向移动时，使用它可提高性能。
AdOpenKeyset	键集游标。尽管从您的记录集不能访问其他用户删除的记录，但除无法查看其他用户添加的记录外，键集游标与动态游标相似。仍然可以看见其他用户更改的数据。
AdOpenDynamic	动态游标。可以看见其他用户所作的添加、更改和删除。允许在记录集中进行所有类型的移动，但不包括提供者不支持的书签操作。
AdOpenStatic	静态游标。可以用来查找数据或生成报告的记录集合的静态副本。另外，对其他用户所作的添加、更改或删除不可见。

说明

使用 **CursorType** 属性可指定打开 **Recordset** 对象时应该使用的游标类型。**Recordset** 关闭时 **CursorType** 属性为读/写，而 **Recordset** 打开时该属性为只读。

如果将 CursorLocation(See 1.1.4.6.17) 属性设置为 **adUseClient** 则只支持 **adUseStatic** 的设置。如果设置了不支持的值，不会导致

错误，将使用最接近支持的 **CursorType**。

如果提供者不支持所请求的游标类型，提供者可能会返回其他游标类型。打开 **Recordset** 对象时，将更改 **CursorType** 属性使之与实际使用的游标匹配。要验证返回游标的指定功能，请使用 Supports(See 1.1.4.4.44) 方法。关闭 **Recordset** 后，**CursorType** 属性将恢复为最初的设置。

下表说明每个游标类型所需的提供者功能（由 **Supports** 方法常量标识）。

对于该 CursorType 的某 Recordset	对于所有这些常量， Supports 方法必须返回 True
AdOpenForwardOnly	无
AdOpenKeyset	AdBookmark、adHoldRecords、adMovePrevious、adResync
AdOpenDynamic	adMovePrevious
AdOpenStatic	adBookmark, adHoldRecords, adMovePrevious, adResync

注意 尽管对于动态游标和仅向前游标 Supports(adUpdateBatch) 可能是真，但对于批处理更新应使用键集游标或静态游标。请将 LockType 属性设置为 adLockBatchOptimistic，然后将 CursorLocation 属性设置为 adUseClient 以启用批更新需要的 OLE DB 游标服务。

远程数据服务用法 当用于客户端 (ADOR) Recordset 对象时，只能将 CursorType 属性设置为 adOpenStatic。

1.1.4.6.19 DataMember 属性 (ADO)

指定要从 **DataSource** 属性所引用的对象中检索的数据成员的名称。

设置和返回值

设置或返回字符串值。名称不区分大小写。

说明

该属性用于通过“数据环境”创建数据绑定控件。“数据环境”保存着数据集合（数据源），而数据集合包含将被表示为 **Recordset** 对象的已命名对象（数据成员）。

DataMember 和 **DataSource** 属性必须连同使用。

DataMember 属性决定将把 **DataSource** 属性所指定的哪个对象作为 **Recordset** 对象提取出来。设置该属性前必须关闭 **Recordset** 对象。如果在设置 **DataSource** 属性前没有设置 **DataMember** 属性，或者在 **DataSource** 属性中指定的对象不能识别 **DataMember** 名称，都将产生错误。

详细信息，请参阅“Data Access SDK”的“Control Writer”部分内容。

用法

```
Dim rs as New ADODB.Recordset
rs.DataMember = "Command"       '需绑定的行集合名称
Set rs.DataSource = myDE        '包含 IRowset 的对象名称
```

1.1.4.6.20 DataSource 属性 (ADO)

指定所包含的数据将被表示为 **Recordset** 对象的对象。

说明

该属性用于通过“数据环境”创建数据绑定控件。“数据环境”保存着数据集合（数据源），而数据集合包含将被表示为 **Recordset** 对象的已命名对象（数据成员）。

DataMember 和 **DataSource** 属性必须连同使用。

所引用的对象必须执行 **IDataSource** 接口，并且必须包含 **IRowset** 接口。

详细信息，请参阅“Data Access SDK”的“Control Writer”部分内容。

用法

```
Dim rs as New ADODB.Recordset
rs.DataMember = "Command"       '要绑定到的行集合的名称
Set rs.DataSource = myDE        '包含 IRowset 的对象的名称
```

1.1.4.6.21 DefaultDatabase 属性 (ADO)

指示 Connection(See 1.1.4.2.2) 对象的默认数据库。

设置和返回值

设置或返回**字符串**来计算出从提供者处可用的数据库的名称。

说明

使用 **DefaultDatabase** 属性可设置或返回指定 **Connection** 对象上默认数据库的名称。

如果有默认数据库，SQL 字符串可使用非限定语法访问该数据库中的对象。如要访问 **DefaultDatabase** 属性中指定数据库以外的数据库中的对象，对象名必须与所需的数据库名称匹配。连接时，提供者将默认数据库信息写入 **DefaultDatabase** 属性。某些提供者对于每个连接只允许一个数据库，在此情况下将不能更改 **DefaultDatabase** 属性。

某些数据源和提供者可能不支持此项功能，这时将返回错误或空的字符串。

远程数据服务用法 该属性在客户端的 Connection 对象上无效。

1.1.4.6.22 DefinedSize 属性 (ADO)

指示 Field(See 1.1.4.2.7) 对象所定义的大小。

返回值

返回反映某个字段定义大小（按字节数）的**长整型**值。

说明

使用 **DefinedSize** 属性可确定 **Field** 对象的数据容量。

DefinedSize 属性与 ActualSize(See 1.1.4.6.5) 属性不同。例如，假设有一个 **Field** 对象，其声明类型为 **adVarChar**，**DefinedSize** 属性值为 50，并包含单个字符。该对象返回的 **ActualSize** 属性值为单个字符的字节长度。

1.1.4.6.23 Description 属性 (ADO)

描述 Error(See 1.1.4.2.6) 对象。

返回值

返回**字符串**值。

说明

使用 **Description** 属性可获得错误的简要说明。显示该属性可针对您无法或不想处理的错误向用户提出警告。该字符串可源于 ADO 或提供者。

提供者应负责将特定的错误文本传递到 ADO。ADO 将 Error(See 1.1.4.2.6) 对象添加到所收到的每个提供者错误或警告的 Error(See 1.1.4.3.1) 集合中。枚举 **Errors** 集合可跟踪提供者传递的错误。

1.1.4.6.24 Direction 属性 (ADO)

指示 Parameter(See 1.1.4.2.8) 所标明的是输入参数、输出参数还是既是输出又是输入参数，或该参数是否为存储过程返回的值。

设置和返回值

设置或返回以下某个 **ParameterDirectionEnum** 值。

常量	说明
AdParamUnknown	指示参数方向未知。
AdParamInput	默认值。指示输入参数。
AdParamOutput	指示输出参数。
AdParamInputOutput	同时指示输入参数和输出参数。
AdParamReturnValue	指示返回值。

说明

使用 **Direction** 属性可指定向过程传递参数或从过程传递参数的方式。**Direction** 属性为读/写；该属性允许使用不返回该信息的提供者，或者在不希望 ADO 为了获取参数信息而附加调用提供者时设置该信息。

并非所有的提供者都可以在其存储过程中确定参数方向。在此情况下，在执行查询前必须设置 **Direction** 属性。

1.1.4.6.25 EditMode 属性 (ADO)

指示当前记录的编辑状态。

返回值

返回如下某个 **EditModeEnum** 值。

常量	说明
AdEditNone	指示当前没有编辑操作。
AdEditInProgress	指示当前记录中的数据已被修改但未保存。
AdEditAdd	指示 AddNew(See 1.1.4.4.1) 方法已被调用，且复制缓冲区中的当前记录是尚未保存到数据库中的新记录 。
AdEditDelete	指示当前记录已被删除。

说明

ADO 维护与当前记录关联的编辑缓冲区。该属性指示是否对该缓冲进行了更改，或是否已创建了新的记录。使用 EditMode 属性可确定当前记录的编辑状态。如果编辑进程被中断，可以对挂起的更改进行测试，并确定是否需要使用 Update(See 1.1.4.4.45) 方法或 CancelUpdate(See 1.1.4.4.8) 方法。

有关不同编辑条件下 **EditMode** 属性的详细说明，请参考 **AddNew** 方法。

1.1.4.6.26 ExecuteOptions 属性 (RDS)

指示是否启用异步执行。

设置和返回值

设置或返回如下值。

常量	说明
AdcExecSync	同步执行 **Recordset** 的下一个刷新操作。
AdcExecAsync	默认值。异步执行 **Recordset** 的下一个刷新操作。

注意 使用这些常量的每个客户端的可执行文件必须为其提供声明。可以从位于 C:\Program Files\Common Files\System\MSADC 文件夹的文件 Adcvbs.inc 中剪切并粘贴常量声明。

说明

如果将 **ExecuteOptions** 设置为 **adcExecAsync**，则对 **RDS.DataControl** 对象的 **Recordset** 上异步执行下一个 **Refresh** 调用。当正在执行可能更改 RDS.**DataControl** 对象中 **Recordset** 的另一个异步操作时，如果试图调用 Reset(See 1.1.4.4.39)、Refresh(See 1.1.4.4.37)、SubmitChanges(See 1.1.4.4.43)、CancelUpdate(See 1.1.4.4.8) 或 Recordset(See 1.1.4.6.53)，将产生错误。

如果在异步操作中发生错误，**RDS.DataControl** 对象的 ReadyState(See 1.1.4.6.54) 值将由 **adcReadyStateLoaded** 更改为 **adcReadyStateComplete**，而 **Recordset** 属性的值仍保持为 **Nothing**。

1.1.4.6.27 FetchOptions 属性 (RDS)

指示异步获取的类型。

设置和返回值

设置或返回如下值。

常量	说明
AdcFetchUpFront	控件返回到应用程序前将获取 **Recordset** 的所有记录。在取到完整的 **Recordset** 之后才允许应用程序使用它进行其他操作。
AdcFetchBackground	只要获取首批记录，控件即可返回应用程序。对试图访问未在首批获取的记录而进行的后续读取将延迟到已实际获取待查的记录，此时控件将返回应用程序。
AdcFetchAsync	默认值。后台取记录时，控件将立即返回应用程序。如果应用程序试图读取尚未获取的记录，则它将读取最接近所发现记录的记录，并且控件会立即返回，以表明已到达 **Recordset** 的当前的尾部。例如，对 **MoveLast** 的调用将使当前记录位置移动到实际获取的上一条记录，即使已有其他记录会继续充填 **Recordset**。

注意 使用这些常量的每个客户端可执行文件必须为这些常量提供声明。可以从位于 C:\Program Files\Common Files\System\MSADC 文件夹的文件 Adcvbs.inc 中剪切并粘贴所需的常量声明。

说明

在 Web 应用程序中，通常使用 **adcFetchAsync**（默认值），因为该值提供更佳的性能。在编译的客户端应用程序中，通常使用 **adcFetchBackground**。

1.1.4.6.28 Filter 属性 (ADO)

为 Recordset(See 1.1.4.2.10) 中的数据指定筛选条件。

设置和返回值

设置或返回**变体型值**，该值包含以下某项内容：

- 条件字符串 — 由一个或多个用 **AND** 或 **OR** 操作符连接的子句组成的字符串。
- 书签数组 — 指向 **Recordset** 对象中记录的唯一书签值数组。
- 以下某个 **FilterGroupEnum** 值。

常数	说明
AdFilterNone	删除当前筛选条件并恢复查看的所有记录。
AdFilterPendingRecords	允许只查看已更改且尚未发送到服务器的记录。只能应用于批更新模式。
AdFilterAffectedRecords	允许只查看上一次 Delete(See 1.1.4.4.20)、Resync(See 1.1.4.4.40)、UpdateBatch(See 1.1.4.4.46) 或 CancelBatch(See 1.1.4.4.7) 调用所影响的记录。
AdFilterFetchedRecords	允许查看当前缓冲区中的记录，即上一次从数据库中检索记录的调用结果。
AdFilterConflictingRecords	允许查看在上一次批更新中失败的记录。

-

说明

使用 **Filter** 属性可选择性地屏蔽 **Recordset** 对象中的记录，已筛选的 **Recordset** 将成为当前游标。这将影响基于当前游标返回值的其他属性，如 AbsolutePosition(See 1.1.4.6.2)、AbsolutePage(See 1.1.4.6.1)、RecordCount(See 1.1.4.6.52) 和 PageCount(See 1.1.4.6.47)，因为将 Filter 属性设置为特定值可将当前记录移动到满足新值的第一个记录。

条件字符串由 *FieldName-Operator-Value* 格式（如“LastName = 'Smith'”）的子句组成。可以创建用单独的 **AND**（如“LastName = 'Smith' AND FirstName = 'John'”）或 **OR**（如“LastName = 'Smith' OR LastName = 'Jones'”）子句连接而成的混合子句。对于条件字符串请遵循以下规则：

- **FieldName** 必须为 **Recordset** 中的有效字段名。如果字段名包含空格，必须用方括号将字段名括起来。
- **Operator** 必须使用的操作符为：<、>、<=、>=、<>、= 或 **LIKE**。
- **Value** 是用于与字段值（如 'Smith'、#8/24/95#、12.345 或 $50.00）进行比较的值。字符串使用单引号而日期使用井号 (#)，对于数字，可以使用小数点、货币符号和科学记数法。如果 **Operator** 为 **LIKE**，**Value** 则可使用通配符。只允许使用星号 (*) 和百分号 (%) 通配符，而且必须为字符串的尾字符。**Value** 不可为 **Null**。
- **AND** 和 **OR** 在级别上没有先后之分。可使用括号将子句分组。但不能象以下示例那样先将由 **OR** 联接的子句分组，然后将该组用 **and** 联接到其他子句。

```
(LastName = 'Smith' OR LastName = 'Jones') AND FirstName = 'John'
```

- 与之相反，可以构造如下形式的筛选：

```
(LastName = 'Smith' AND FirstName = 'John') OR (LastName = 'Jones' AND
FirstName = 'John')
```

- 在 **LIKE** 子句中，可在样式的开头和结尾使用通配符（如 LastName Like '*mit*'），或者只在结尾使用通配符（如，LastName Like 'Smit*'） 。

通过仅允许查看（例如）上次调用 **UpdateBatch** 方法时受到影响的记录，筛选常量使得在批更新模式时所发生的单个记录冲突更易于解决。

设置 **Filter** 属性本身可能会因与基本数据发生冲突（如某记录已被其他用户删除）而失败。在此情况下，提供者将返回对 Errors(See 1.1.4.3.1) 集合的警告但不停止程序执行。只有在所有需要的记录上发生冲突时才产生运行时错误。使用 Status(See 1.1.4.6.65) 属性可定位发生冲突的记录。

将 **Filter** 属性设置为零长度字符串 ("") 与使用 **adFilterNone** 常量具有同样效果。

一旦设置 **Filter** 属性，当前记录位置将移动到 **Recordset** 中已筛选记录子集中的第一个记录。类似地，清除 **Filter** 属性后，当前记录位置将移动到 **Recordset** 的第一个记录。

有关可与 **Filter** 属性一起使用创建数组的书签值的解释，请参考 Bookmark(See 1.1.4.6.8) 属性。

1.1.4.6.29 FilterColumn 属性 (RDS)

指示计算筛选条件的列。

语法

DataControl.**FilterColumn** = *String*

参数

DataControl 表示 **RDS.DataControl** 对象的对象变量(See 2.34)。

String 指定计算筛选条件的列的**字符串**值。筛选条件在 FilterCriterion(See 1.1.4.6.30) 属性中指定。

说明

SortColumn(See 1.1.4.6.58)、SortDirection(See 1.1.4.6.59)、FilterValue(See 1.1.4.6.31)、FilterCriterion(See 1.1.4.6.30) 和 **FilterColumn** 属性提供客户端缓冲区上的排序和筛选功能。排序功能按照某列的值对记录进行排序。缓冲区中有完整的 **Recordset** 时，筛选功能将根据查找条件显示记录子集。**Reset** 方法将执行条件并用可更新的 **Recordset** 替换当前的 **Recordset**。

1.1.4.6.30 FilterCriterion 属性 (RDS)

设置或返回在筛选值中使用的计算操作符。

语法

DataControl.**FilterCriterion** = *String*

参数

DataControl 表示 **RDS.DataControl** 对象的对象变量(See 2.34)。

String 将 FilterValue(See 1.1.4.6.31) 的计算操作符指定到记录的字符串值。可以为以下操作符：<、<=、>、>=、= 或 <>。

说明

SortColumn(See 1.1.4.6.58)、SortDirection(See 1.1.4.6.59)、FilterValue(See 1.1.4.6.31)、**FilterCriterion** 和 FilterColumn(See 1.1.4.6.29) 属性提供客户端缓冲区上的排序和筛选功能。排序功能按照某列的值对记录进行排序。缓冲区中有完整的 **Recordset** 时，筛选功能将根据查找条件显示记录子集。**Reset** 方法将执行条件并用可更新的 **Recordset** 替换当前的 **Recordset**。

“!=”操作符对 **FilterCriterion** 无效，应使用“<>”。

如果同时设置筛选和排序属性并且调用 **Reset** 方法，将首先筛选行集合然后对其进行排序。对于升序，NULL 位于顶部；对于降序，NULL 位于底部（升序为默认方式）。

1.1.4.6.31 FilterValue 属性 (RDS)

设置或返回用于筛选记录的值。

语法

DataControl.**FilterValue** = *String*

参数

DataControl 表示 **RDS.DataControl** 对象的对象变量(See 2.34)。

String **字符串**值，表示用于筛选记录的数据值（如 'Programmer' 或 125）。

说明

SortColumn(See 1.1.4.6.58)、SortDirection(See 1.1.4.6.59)、**FilterValue**、FilterCriterion(See 1.1.4.6.30) 和 FilterColumn(See 1.1.4.6.29) 和 **FilterColumn** 属性提供客户端缓冲区上的排序和筛选功能。排序功能按照某列的值对记录进行排序。缓冲区中有完整的 **Recordset** 时，筛选功能将根据查找条件显示记录子集。**Reset** 方法将执行条件并用可更新的 **Recordset** 替换当前的 **Recordset**。空值会导致类型不匹配错误。

1.1.4.6.32 Handler 属性 (RDS)

设置或返回包含扩展 **RDSServer.DataFactory** 功能的服务器端自定义程序（**处理程序**）的名称和**处理程序**所用的任何参数的字符串，它们均由逗号 (",") 分隔。

语法

DataControl.**Handler** = *String*

参数

DataControl 表示 **RDS.DataControl** 对象的对象参数(See 2.34)。

String 包含**处理程序**和任何参数名称的**字符串**值，均由逗号分隔（如“handlerName,parm1,parm2,...,parm*N*”)。

说明

该属性所支持的功能称为自定义(See 1.1.3.8)。对自定义的支持需要将 CursorLocation(See 1.1.4.6.17) 属性设置为 **adUseClient**。

任何处理程序及其参数名称都用逗号 (",") 分隔。如果**字符串**中任何位置出现分号 (";")，将导致不可预料的行为。如果它支持 **IdataFactoryHandler** 接口，则可以编写自己的处理程序。

默认的处理程序名称为 **MSDFMAP.Handler**，其默认参数为自定义文件 **MSDFMAP.INI**。使用该属性可调用由服务器管理员创建的可替换自定义文件。

设置 **Handler** 属性的可选方式是在 ConnectionString(See 1.1.4.6.14) 属性中指定处理程序和参数；即“**Handler=***handlerName,parm1,parm2,...;*”。

1.1.4.6.33 HelpContext、HelpFile 属性 (ADO)

指示与 Error(See 1.1.4.2.6) 对象关联的帮助文件和主题。

返回值

- **HelpContextID** — 返回帮助文件中主题的、按**长整型**值返回的环境 ID。
- **HelpFile** — 返回字符串，用于计算帮助文件的完整分解路径。

说明

如果在 **HelpFile** 属性中指定了帮助文件，则 **HelpContext** 属性被用于自动显示它所标识的帮助主题。如果没有得到相关主题，则 **HelpContext** 属性返回零，并且 **HelpFile** 属性返回零长度字符串 ("")。

1.1.4.6.34 Index 属性 (ADO)

指示对 Recordset(See 1.1.4.2.10) 对象当前生效的索引的名称。

设置和返回值

设置或返回字符串值，该值为索引名。

说明

由 **Index** 属性命名的索引必须针对基本表基本 **Recordset** 对象已在前面声明过。即索引必须已在程序中声明为 ADOX Index(See 1.2.1.2.4) 对象，或在创建基本表时声明。

如果无法设置索引，则会发生运行时错误。无法在 **WillRecordsetChange** 或 **RecordsetChangeComplete** 事件处理程序内设置 **Index** 属性。如果 **Recordset** 正在执行操作，也无法对它进行设置。如果 **Recordset** 是关闭的，则总能成功设置 **Index** 属性，但

如果基本提供者不支持索引，则 **Recordset** 将无法成功打开，或者索引将无法使用。

如果可以设置索引，则可以更改当前行的位置。这将导致对 AbsolutePosition(See 1.1.4.6.2) 属性的更新，并产生 WillRecordsetChange(See 1.1.4.5.12)、RecordsetChangeComplete(See 1.1.4.5.12)、WillMove(See 1.1.4.5.15) 和 MoveComplete(See 1.1.4.5.15) 事件。

如果可以设置索引，而 LockType(See 1.1.4.6.37) 属性是 **adLockPessimistic** 或 **adLockOptimistic**，那么，将执行隐式 UpdateBatch(See 1.1.4.4.46) 操作，并释放当前的和受影响的组。任何现有的 filter(See 1.1.4.6.28) 被释放，并且当前行位置更改为重排序后 **Recordset** 的第一行。

Index 属性与 Seek(See 1.1.4.4.42) 方法连通使用。如果基本提供者不支持 **Index** 属性和 **Seek** 方法，请考虑使用 Find(See 1.1.4.4.23) 方法替代。使用 Supports(See 1.1.4.4.44)**(adIndex)** 方法可判定 **Recordset** 对象是否支持索引。

尽管二者均处理索引，但内置的 **Index** 属性与动态的 Optimize(See 1.1.4.6.45) 属性无关。

1.1.4.6.35 InternetTimeout 属性 (RDS)

指示请求超时前等待的毫秒数。

设置和返回值

设置或返回**长整型**值。

说明

该属性只适用于通过 HTTP 或 HTTPS 协议发送的请求。

执行三层环境中的请求可能需要几分钟。使用该属性可为长时间运行的请求指定附加时间。

1.1.4.6.36 IsolationLevel 属性 (ADO)

指示 Connection(See 1.1.4.2.2) 对象的隔离级别。

设置和返回值

设置或返回以下某个 **IsolationLevelEnum** 值。

常数	说明
adXactUnspecified	指示提供者正在使用非指定的 IsolationLevel，但其级别无法确定。
adXactChaos	默认值。指示无法从更高级隔离事务覆盖挂起的更改。
adXactBrowse	指示从某事务中可以查看其他事务中未提交的更改。
adXactReadUncommitted	同 adXactBrowse。
adXactCursorStability	默认值。表明只有在事务提交后才能从某事务中查看它们的更改。
adXactReadCommitted	同 adXactCursorStability。
adXactRepeatableRead	指示无法从某事务中查看其他事务中所作的更改，但通过查询可以得到新记录

	集。
adXactIsolated	指示该事务在与其他事务隔离的情况下执行。
adXactSerializable	同 **adXactIsolated**。

说明

使用 **IsolationLevel** 属性可设置 **Connection** 对象的隔离级别。**IsolationLevel** 的属性为读/写。直到下次调用 BeginTrans(See 1.1.4.4.4) 方法时，该设置才可以生效。如果您请求的隔离级别不可用，提供者可能返回下一个更高的隔离级别。

远程数据服务用法 当用于客户端 Connection 对象时，只能将 IsolationLevel 属性设置为 adXactUnspecified。

由于用户正在使用客户端缓冲区中已断开的 Recordset 对象，所以可能会出现多用户问题。例如，当两个不同的用户对同一记录进行更新时，Remote Data Service 只允许首先更新该记录的用户实现更新操作。第二个用户的更新请求将失败，并产生错误。

1.1.4.6.37 LockType 属性 (ADO)

指示编辑过程中对记录使用的锁定类型。

设置和返回值

设置或返回以下某个 **LockTypeEnum** 的值。

常量	说明
adLockReadOnly	默认值，只读。无法更改数据。
adLockPessimistic	保守式记录锁定（逐条）。提供者执行必要的操作确保成功编辑记录，通常采用编辑时立即锁定数据源的记录的方式。
adLockOptimistic	开放式记录锁定（逐条）。提供者使用开放式锁定，只在调用 Update(See 1.1.4.4.45) 方法时锁定记录。
adLockBatchOptimistic	开放式批更新。用于与立即更新模式相反的批更新模式。

说明

打开 **Recordset** 前设置 **LockType** 属性可指定打开时提供者应该使用的锁定类型。读取该属性可返回在打开的 **Recordset** 对象上正在使用的锁定类型。**Recordset** 关闭时 **LockType** 属性为读/写，打开时该属性为只读。

提供者可能不支持所有的锁定类型。如果某提供者不支持所需的 **LockType** 设置，则将替换为其他类型的锁定。要确定 **Recordset** 对象可用的实际锁定功能，请通过 **adUpdate** 和 **adUpdateBatch** 使用 Supports(See 1.1.4.4.44) 方法。

如果 CursorLocation(See 1.1.4.6.17) 属性被设置为 **adUseClient**，将不支持 **adLockPessimistic** 设置。设置不支持的值不会产生错误，因为此时将使用支持的最接近的 **LockType** 的值。

远程数据服务用法 当在客户端 (ADOR) 的 **Recordset** 对象上使用时，**LockType** 属性只能设置为 **adLockOptimisticBatch**。

1.1.4.6.38 MarshalOptions 属性 (ADO)

指示要被调度返回服务器的记录。

设置和返回值

设置或返回以下某常量的**长整型**值。

常量	说明
AdMarshalAll	默认值。表明所有行将返回到服务器。
AdMarshalModifiedOnly	表明只有已修改的行返回到服务器。

说明

当使用客户端 (ADOR) **Recordset** 时，已在客户端被修改的记录将通过称作“调度”的技术写回中间层或 Web 服务器。调度是指越过线程或进程边界包装和发送接口方法参数的过程。当已修改的远程数据通过调度被更新回中间层或 Web 服务器时，设置 **MarshalOptions** 属性可以提高性能。

远程数据服务用法 该属性只能在客户端 (ADOR) **Recordset** 上使用。

1.1.4.6.39 MaxRecords 属性 (ADO)

指示通过查询返回 Recordset(See 1.1.4.2.10) 的记录的最大数目。

设置和返回值

设置或返回**长整型**值。默认值为零（没有限制）。

说明

使用 **MaxRecords** 属性可对提供者从数据源返回的记录数加以限制。该属性的默认设置为零，表明提供者返回所有所需的记录。**Recordset** 关闭时，**MaxRecords** 属性为读/写，打开时为只读。

1.1.4.6.40 Mode 属性 (ADO)

指示在 Connection(See 1.1.4.2.2) 中修改数据的可用权限。

设置和返回值

设置或返回以下某个 **ConnectModeEnum** 的值。

常量	说明
AdModeUnknown	默认值。表明权限尚未设置或无法确定。
AdModeRead	表明权限为只读。
AdModeWrite	表明权限为只写。
AdModeReadWrite	表明权限为读/写。
AdModeShareDenyRead	防止其他用户使用读权限打开连接。
AdModeShareDenyWrite	防止其他用户使用写权限打开连接。
AdModeShareExclusive	防止其他用户打开连接。
AdModeShareDenyNone	防止其他用户使用任何权限打开连接。

说明

使用 **Mode** 属性可设置或返回当前连接上提供者正在使用的访问权限。**Mode** 属性只能在关闭 **Connection** 对象时方可设置。

远程数据服务用法 当用于客户端 Connection 对象时，Mode 属性只能设置为 adModeUnknown。

1.1.4.6.41 Name 属性 (ADO)

指示对象的名称。

设置和返回值

设置或返回**字符串**值。该值在 **Command** 或 **Parameter** 对象上为读/写，在 **Property** 或 **Field** 对象上为只读。

说明

使用 **Name** 属性可将名称赋给 **Command**、**Field**、**Parameter** 或 **Property** 对象，或检索它们的名称。

对于还没有追加到 Parameters(See 1.1.4.3.3) 集合中的 **Parameter** 对象，**Name** 属性为读/写。对于已经追加的 **Parameter** 对象以及所有其他对象，**Name** 属性为只读。在集合中名称不必唯一。

可以按照序列索引检索对象的 **Name** 属性，在此之后则可以直接使用名称引用对象。例如，如果 **rstMain.Properties(20).Name** 为 **Updatability**，可以随后将该属性引用为 **rstMain.Properties(**"Updatability")。

1.1.4.6.42 NativeError 属性 (ADO)

指示给定 Error(See 1.1.4.2.6) 对象的、特定提供者的错误代码。

返回值

返回**长整型**值。

说明

使用 **NativeError** 属性可对特殊 **Error** 对象检索特定数据库的错误信息。例如，当通过 Microsoft® SQL Server® 数据库使用 Microsoft ODBC Provider for OLE DB 时，从 SQL 服务器产生的本地错误代码将通过 ODBC 和 ODBC 提供者传递到 ADO **NativeError** 属性。

1.1.4.6.43 Number 属性 (ADO)

指示用于唯一标识 Error(See 1.1.4.2.6) 对象的数字。

返回值

返回**长整型**值。

说明

使用 **Number** 属性可确定发生了哪个错误。属性的值是唯一对应错误条件的数字。

1.1.4.6.44 NumericScale 属性 (ADO)

指出 Parameter(See 1.1.4.2.8) 或 Field(See 1.1.4.2.7) 对象中数字值的范围。

设置和返回值

设置或返回**字节**值，指示数字值所精确到的小数点位数。

说明

使用 **NumericScale** 属性可确定用于表明数字型 **Parameter** 或 **Field** 对象的值的小数位数。

对于 **Parameter** 对象，**NumericScale** 属性为读/写。对于 **Field** 对象，**NumericScale** 属性为只读。

1.1.4.6.45 Optimize 属性 (RDS)

指示是否应该在该字段上创建索引。

设置和返回值

设置或返回布尔型值。

说明

索引可提高在 **Recordset** 中查找或排序值的操作性能。索引对于 ADO 是内部的，您无法在应用程序中显式访问或使用索引。**Optimize** 属性是“动态的”；该属性不是 **Field** 对象接口的一部分。如果将 **CursorLocation** 属性设置为 **adUseClient**，则该属性只在 **Field** 对象的 **Properties** 集合中存在。

要创建字段的索引，请将 **Optimize** 属性设置为 **True**。要删除索引，请将该属性设置为 **False**。

用法

```
Dim rs As New Recordset
Dim fld As Field
rs.CursorLocation = adUseClient       '启用索引创建
rs.Fields.Append "Field1", adChar, 35, adFldIsNullable
rs.Open
Set fld = rs.Fields(0)
fld.Properties("Optimize") = True     '创建索引
fld.Properties("Optimize") = False    '删除索引
```

1.1.4.6.46 OriginalValue 属性 (ADO)

指示发生任何更改前已在记录中存在的 Field(See 1.1.4.2.7) 的值。

返回值

返回 **Variant** 值。

说明

使用 **OriginalValue** 属性可返回当前记录中某字段的原始字段值。

在立即更新模式（一旦您调用 Update(See 1.1.4.4.45) 方法，提供者会将更改写入下一级数据源）中，OriginalValue 属性将返回在发生任何更改前（即自从上次调用 Update 方法）已经存在的字段值。该值与 CancelUpdate(See 1.1.4.4.8) 方法用来替换 Value(See 1.1.4.6.69) 属性的值相同。

在批更新模式（只有在调用 UpdateBatch(See 1.1.4.4.46) 方法时提供者才缓存多个更改并将其写入下一级数据源）下，**OriginalValue** 属性返回在发生任何更改前（即自从调用 **UpdateBatch** 方法）已经存在的字段值。该值与 CancelUpdate(See 1.1.4.4.8) 方法用来替换 **Value** 属性的值相同。与 UnderlyingValue(See 1.1.4.6.68) 属性一起使用该属性时，可以解决由批更新引起的冲突。

1.1.4.6.47 PageCount 属性 (ADO)

指示 Recordset(See 1.1.4.2.10) 对象包含的数据页数。

返回值

返回**长整型**值。

说明

使用 **PageCount** 属性可确定 **Recordset** 对象中数据的页数。“页”是大小等于 PageSize(See 1.1.4.6.48) 属性设置的记录组。即使最后页不完整，由于记录数比 **PageSize** 值少，该页也会作为 **PageCount** 值中的附加页进行计数。如果 **Recordset** 对象不支持该属性，该值为 -1 以表明 **PageCount** 无法确定。

有关页的功能的详细信息，请参阅 **PageSize** 和 AbsolutePage(See 1.1.4.6.1) 属性。

1.1.4.6.48 PageSize 属性 (ADO)

指示 Recordset(See 1.1.4.2.10) 中一页所包含的记录数。

设置和返回值

设置或返回**长整型**值，该值指示某页上的记录数。默认值为 10。

说明

使用 **PageSize** 属性可确定组成逻辑数据页的记录数。建立页的大小允许使用 AbsolutePage(See 1.1.4.6.1) 属性移动到特定页的第一个记录。在您希望允许用户对数据进行分页时，该属性在 Web-服务器方案中非常有用，可用来在某一时刻查看一定数量的记录。随时可以设置该属性，其值将用来计算特定页第一个记录的位置。

1.1.4.6.49 Precision 属性 (ADO)

指示在 Parameter(See 1.1.4.2.8) 对象中数字值或数字 Field(See 1.1.4.2.7) 对象的精度。

设置和返回值

设置或返回 **Byte** 值，用来表示值的最大位数。该值在 **Parameter** 对象上为读/写，而在 **Field** 对象上为只读。

说明

使用 **Precision** 属性可确定表示数字 **Parameter** 或 **Field** 对象值的最大位数。

1.1.4.6.50 Prepared 属性 (ADO)

指示执行前是否保存命令的编译版本。

设置和返回值

设置或返回布尔型值。

说明

使用 **Prepared** 属性可使提供者在首次执行 Command(See 1.1.4.2.1) 对象前保存 CommandText(See 1.1.4.6.10) 属性中指定的已准备好（已编译）的查询版本。该属性会降低命令首次执行的速度，但提供者对命令进行编译后，在后继的命令执行中提供者可使用

已编译好命令版本，这样可以提高执行性能。

如果该属性为 **False**，提供者将直接执行 **Command** 对象而不创建编译版本。

如果提供者不支持命令准备，将该属性设置为 **True** 时也许将返回错误。如果没有返回错误，则只需要忽略准备命令的请求并将 **Prepared** 属性设置为 **False** 即可。

1.1.4.6.51 Provider 属性 (ADO)

指示 Connection(See 1.1.4.2.2) 对象提供者的名称。

设置和返回值

设置或返回**字符串**值。

说明

使用 **Provider** 属性可设置或返回连接提供者的名称。也可以通过 ConnectionString(See 1.1.4.6.14) 属性的内容或 Open(See 1.1.4.4.32) 方法的 **ConnectionString** 参数设置该属性。但是，调用 Open 方法时在多处指定提供者可能会产生无法预料的后果。如果没有指定提供者，该属性将默认为 MSDASQL (Microsoft OLE DB Provider for ODBC)。

关闭连接时 **Provider** 属性为读/写，打开连接时该属性为只读。打开 **Connection** 对象或访问 **Connection** 对象的 Properties(See 1.1.4.3.4) 集合后该设置才生效。如果该设置无效，将产生错误。

1.1.4.6.52 RecordCount 属性 (ADO)

指示 Recordset(See 1.1.4.2.10) 对象中记录的当前数目。

返回值

返回**长整型**值。

说明

使用 **RecordCount** 属性可确定 **Recordset** 对象中记录的数目。ADO 无法确定记录数时，或者如果提供者或游标类型不支持 **RecordCount**，则该属性返回 －1。读已关闭的 **Recordset** 上的 **RecordCount** 属性将产生错误。

如果 **Recordset** 对象支持近似定位或书签（即 **Supports (adApproxPosition)** 或 **Supports (adBookmark)** 各自返回 **True**），不管是否完全填充该值，该值将为 **Recordset** 中记录的精确数目。如果 **Recordset** 对象不支持近似定位，该属性可能由于必须对所有记录进行检索和计数以返回精确 **RecordCount** 值而严重消耗资源。

Recordset 对象的游标类型会影响是否能够确定记录的数目。对仅向前游标，**RecordCount** 属性将返回 -1，对静态或键集游标返回实际计数，对动态游标取决于数据源返回 -1 或实际计数。

1.1.4.6.53 Recordset、SourceRecordset 属性 (RDS)

指示从自定义业务对象(See 2.7)中返回的 **ADOR.Recordset** 对象。

可以在运行时(See 2.43)在脚本代码（如 VBScript）中设置 **SourceRecordset** 属性或读取 **Recordset** 属性。

语法

DataControl.**SourceRecordset** = *Recordset*

Recordset = *DataControl*.**Recordset**

参数

DataControl 表示 **RDS.DataControl** 对象的对象变量(See 2.34)。

Recordset 表示 **ADOR.Recordset** 对象的对象变量。

说明

SourceRecordset 为只写属性，相反 **Recordset** 为只读属性。

可以将 **SourceRecordset** 属性设置为从自定义业务对象(See 2.7)中返回的 **Recordset**。

这些属性允许应用程序通过自定义进程的方式处理绑定进程。这些属性接受 **Recordset** 中包装的行集合(See 2.42)以便与 **Recordset** 直接交互作用，执行如设置属性或在 **Recordset** 中重复等操作。

1.1.4.6.54 ReadyState 属性 (RDS)

在 RDS.DataControl(See 1.1.4.2.3) 对象获取数据到它的 **Recordset** 对象中时反映其进度。

设置或返回值

设置或返回以下某值。

值	说明
adcReadyStateLoaded	当前查询仍在执行并且没有获取到任何行。不能使用 **RDS.DataControl** 对象的 **Recordset**。
adcReadyStateInteractive	已将当前查询检索到的行的初始集合保存到 **RDS.DataControl** 对象的 **Recordset** 并可供使用。正在获取其余的行。
adcReadyStateComplete	已将当前查询检索到的所有的行保存到 **RDS.DataControl** 对象的 **Recordset** 中并可供使用。 如果由于错误而终止操作，或者没有初始化 **Recordset** 对象时，也会存在该状态。

注意 使用这些常量的每个客户端可执行文件必须提供这些变量的声明。可以从位于 C:\Program Files\Common Files\System\MSADC 文件夹中的 Adcvbs.inc 文件中剪切并粘贴所需的常量声明

说明

使用 onReadyStateChange(See 1.1.4.5.9) 事件方法可监视异步查询操作中 **ReadyStateChange** 属性中发生的更改。这是比定期检查属性值更有效的方式。

如果异步操作过程中产生错误，**ReadyState** 属性将更改为 **adcReadyStateComplete**，而 State(See 1.1.4.6.64) 属性将从

adStateExecuting 更改为 **adStateClosed**，**Recordset** 对象的 **Value** 属性为 Nothing。

1.1.4.6.55 Server 属性 (RDS)

设置或返回 Internet Information Server (IIS) 名称和通讯协议。

Server 属性可以在设计时(See 2.20)在 **RDS.DataControl** 对象的 OBJECT 标记中设置，或在运行时(See 2.43)在脚本代码（例如 VBScript）中设置。

语法

协议	设计时语法
HTTP	**<PARAM NAME="Server" VALUE="http://***awebsrvr:port***">**
HTTPS	**<PARAM NAME="Server" VALUE="https://***awebsrvr:port***">**
DCOM	**<PARAM NAME="Server" VALUE="***machinename***">**
In-process	**<PARAM NAME="Server" VALUE="">**

协议	运行时语法
HTTP	*DataControl*.**Server="https://***awebsrvr:port***"**
HTTPS	*DataControl*.**Server="https://***awebsrvr:port***"**
DCOM	*DataControl*.**Server="***machinename***"**
In-process	*DataControl*.**Server=""**

部件	说明
Awebsrvr 或 *machinename*	包含有效的 Internet 或 Intranet 路径和服务器名的字符串。
Port	可选项。用于与 IIS 服务器连接的端口。端口号在 Internet Explorer（在“查看”菜单上，单击“选项”，然后选择“连接”选项卡）或在 IIS 中设置。
DataControl	表示 **RDS.DataControl** 对象的对象变量(See 2.34)。

说明

服务器位置是创建服务器端对象(See 2.7)的地方（如有不同，则与数据位置相反）。

1.1.4.6.56 Size 属性 (ADO)

表示 **Parameter** 对象的最大大小（按字节或字符）。

设置和返回数值

设置或返回表示 **Parameter** 对象的最大大小（按字节或字符）的**长整型**值。

说明

使用 **Size** 属性确定对 **Parameter** 对象的 Value(See 1.1.4.6.69) 属性写入（或读出）值的最大大小。**Size** 属性为可读/写。

如果要指定 **Parameter** 对象为变长数据类型（例如任何字符串类型，如 **adVarChar**），则必须先设置对象的 **Size** 属性，然后再将该对象追加到 Parameters(See 1.1.4.3.3) 集合。否则，将会出现错误。

如果已经将 **Parameter** 对象追加到 Command(See 1.1.4.2.1) 对象的 **Parameter** 集合，并已经将它的类型更改为变长数据类型，那么必须先设置 **Parameter** 对象的 **Size** 属性然后再执行 **Command** 对象。否则，将出现错误。

如果使用 Refresh(See 1.1.4.4.36) 方法从提供者获取参数信息，并且它返回一个或多个变长数据类型 **Parameter** 对象，ADO 可能会根据其最大可能的大小为其分配内存空间，这样在执行中可能会导致错误。为避免出错，应该在执行命令之前显式设置这些参数的 **Size** 属性。

1.1.4.6.57 Sort 属性 (ADO)

指定一个或多个 **Recordset** 以之排序的字段名，并指定按升序还是降序对字段进行排序。

设置和返回值

设置或返回以之排序的用逗号分隔的字段名字符串，其中的每个名称是 **Recordset** 中的 Field(See 1.1.4.2.7)。可以选择后跟空格和用于指定字段排序顺序的关键字 **ASC** 或 **DESC**。

说明

实际上数据并没有重新排列，只是简单地按排序的顺序进行访问。

如果 CursorLocation(See 1.1.4.6.17) 属性被设置为 **adUseClient**，并且索引尚不存在，则将为 **Sort** 属性中指定的每个字段创建临时索引。

将 **Sort** 属性设置为空字符串将使行重置为原始顺序，并删除临时索引。现存索引将不被删除。

由于与关键字 **ASC** 和 **DESC** 发生冲突，字段无法命名为“ASC”或“DESC”。请通过在返回 **Recordset** 的查询中使用 **AS** 关键字，来给使用发生名称冲突的字段设置别名。

1.1.4.6.58 SortColumn 属性 (RDS)

设置或返回记录以之排序的列。

语法

DataControl.**SortColumn** = *String*

参数

DataControl 对象变量(See 2.34)，表示 **RDS.DataControl** 对象。

String 字符串值，表示记录以之排序的列的名称或别名。

说明

SortColumn、SortDirection(See 1.1.4.6.59)、FilterValue(See 1.1.4.6.28)、FilterCriterion(See 1.1.4.6.30) 和 FilterColumn(See 1.1.4.6.29) 属性提供客户端缓存的排序和筛选功能。排序功能依据某个列的值将记录排序。筛选功能依据查找条件显示记录子集，而整个 **Recordset** 则保留在缓存中。**Reset** 方法则将执行条件，并用可更新的 **Recordset** 替代当前 **Recordset**。

要对 **Recordset** 进行排序，必须首先保存所有挂起的更改。如果使用 **RDS.DataControl**，则可以使用 **SubmitChanges** 方法。例如，如果 **RDS.DataControl** 名为 ADC1，那么代码将是 ADC1.SubmitChanges。如果正在使用 ADO **Recordset**，则可以使用它的 **UpdateBatch** 方法。对于用 **CreateRecordset** 方法创建的 **Recordset**，建议使用 **UpdateBatch** 方法。例如，代码可以是 myRS.UpdateBatch 或 ADC1.Recordset.UpdateBatch。

1.1.4.6.59 SortDirection 属性 (RDS)

设置或返回用于指示排序顺序是升序还是降序的布尔型值。

语法

DataControl.**SortDirection** = *value*

参数

DataControl 表示 **RDS.DataControl** 对象的对象变量(See 2.34)。

Value 可以设置或返回下列数值之一的布尔值：

- **True** — 升序
- **False** — 降序

说明

- SortColumn(See 1.1.4.6.58)、**SortDirection**、FilterValue(See 1.1.4.6.28)、FilterCriterion(See 1.1.4.6.30) 和 FilterColumn(See 1.1.4.6.29) 属性提供客户端缓存的排序和筛选功能。排序功能依据某个列值将记录排序。筛选功能依据查找条件显示记录的子集，而整个 **Recordset** 则保留在高速缓存中。**Reset** 方法则将执行条件，并用可更新的 **Recordset** 替代当前的 **Recordset**。

1.1.4.6.60 Source 属性 (ADO Error)

指示产生错误的原始对象或应用程序的名称。

返回值

返回**字符串**值。

说明

使用 **Error** 对象的 **Source** 属性来确定产生错误的原始对象或应用程序的名称。该名称可以是对象的类名或编程 ID。对于 ADODB 的错误，属性值将是 **ADODB**.*ObjectName*，**ObjectName** 是触发错误的对象名称。**Error** 对象的 **Source** 属性是只读的。根据来自 **Error** 对象的 **Source**、Number(See 1.1.4.6.43) 和 Description(See 1.1.4.6.23) 属性的出错文档，可以编写对错误进行相应处理的代码。

1.1.4.6.61 Source 属性 (ADO Recordset)

指示 **Recordset** 对象中数据的来源（Command(See 1.1.4.2.1) 对象、SQL 语句、表的名称或存储过程）。

设置和返回值

设置**字符串**值或 **Command** 对象引用。仅返回**字符串**值。

说明

使用 **Source** 属性指定 **Recordset** 对象的数据源，该 **Recordset** 对象使用下列项之一：**Command** 对象变量、SQL 语句、存储过程或表的名称。**Source** 属性对于关闭的 **Recordset** 是可读/写的，对于打开的 **Recordset** 是只读的。

如果设置 **Source** 属性为 **Command** 对象，**Recordset** 对象的 ActiveConnection(See 1.1.4.6.4) 属性将继承指定 **Command** 对象的 **ActiveConnection** 属性的值。但是，读取 **Source** 属性将不返回 **Command** 对象，而是将 **Command** 对象的 CommandText(See 1.1.4.6.10) 属性返回到设置 **Source** 属性的地方。

如果 **Source** 属性是一个 SQL 语句、存储过程或表的名称，则可以通过调用 Open(See 1.1.4.4.33) 方法传递相应的 **Options** 参数来优化性能。

1.1.4.6.62 SQL 属性 (RDS)

设置或返回用于检索 **Recordset** 的查询字符串。

SQL 属性可以在设计时(See 2.20)在 **RDS.DataControl** 对象的 OBJECT 标记中设置，或在运行时(See 2.43)在脚本代码（例如 VBScript）中设置。

语法

设计时:<**PARAM NAME="SQL" VALUE="***QueryString*">

运行时: *DataControl*.**SQL** = "*QueryString*"

参数

QueryString **字符串**，包含有效 SQL 数据请求。

DataControl 对象变量(See 2.34)，表示 **RDS.DataControl** 对象。

说明

通常情况下，这是一个 SQL 语句（使用数据库服务器的特定语言），例如 “Select * from NewTitles”。为确保记录的精确匹配和准确更新，可更新的查询必须包含的字段不能是“长二进制”字段或可计算字段。
如果自定义服务器端业务对象(See 2.7)检索客户端数据，那么 **SQL** 属性是可选的。

1.1.4.6.63 SQLState 属性 (ADO)

指示给定 **Error** 对象的 SQL 状态。

返回值

返回符合 ANSI SQL 标准的 5 个字符的字符串。

说明

使用 **SQLState** 属性读取由提供者在处理 SQL 语句过程中出现错误时返回的 5 个字符的错误代码。例如，在使用带有 Microsoft® SQL Server™ 数据库的 Microsoft OLE DB Provider for ODBC 时，SQL 状态错误代码将从基于特定 ODBC 错误或 Microsoft SQL Server 错误的 ODBC 产生并映射到 ODBC 错误。这些错误代码可在 ANSI SQL 标准中找到，但随着数据源的不同会以不同方式实现。

1.1.4.6.64 State 属性 (ADO)

对所有可应用对象，说明其对象状态是打开或是关闭。
对执行异步方法的 **Recordset** 对象，说明当前的对象状态是连接、执行或是获取。

返回值

返回下列常量之一的**长整型**值。

常量	说明
AdStateClosed	默认，指示对象是关闭的。
AdStateOpen	指示对象是打开的。
AdStateConnecting	指示 **Recordset** 对象正在连接。
AdStateExecuting	指示 **Recordset** 对象正在执行命令。
AdStateFetching	指示 **Recordset** 对象的行正在被读取。

说明

可以随时使用 **State** 属性确定指定对象的当前状态。该属性是只读的。

Recordset 对象的 **State** 属性可以是组合值。例如，如果正在执行语句，该属性将是 **adStateOpen** 和 **adStateExecuting** 的组合值。

1.1.4.6.65 Status 属性 (ADO)

指示有关批更新或其他大量操作的当前记录的状态。

返回值

返回下列一个或多个 **RecordStatusEnum** 值之和。

常量	说明
AdRecOK	成功地更新记录。
AdRecNew	记录是新建的。
AdRecModified	记录被修改。
AdRecDeleted	记录被删除。
AdRecUnmodified	记录没有修改。
AdRecInvalid	由于书签无效，记录没有保存。
AdRecMultipleChanges	由于影响多个记录，因此记录未被保存。
AdRecPendingChanges	由于记录引用挂起的插入，因此未被保存。
AdRecCanceled	由于操作被取消，未保存记录。
AdRecCantRelease	由于现有记录锁定，没有保存新记录。
AdRecConcurrencyViolation	由于开放式并发在使用中，记录未被保存。
AdRecIntegrityViolation	由于用户违反完整性约束，记录未被保存。
AdRecMaxChangesExceeded	由于存在过多挂起更改，记录未被保存。
AdRecObjectOpen	由于与打开的储存对象冲突，记录未被保存。
AdRecOutOfMemory	由于计算机内存不足，记录未被保存。
AdRecPermissionDenied	由于用户没有足够的权限，记录未被保存。
AdRecSchemaViolation	由于记录违反基本数据库的结构，因此未被保存。
AdRecDBDeleted	记录已经从数据源中删除。

说明

使用 **Status** 属性查看在批更新中被修改的记录有哪些更改被挂起。也可使用 **Status** 属性查看大量操作时失败记录的状态。例如，

调用 **Recordset** 对象的 Resync(See 1.1.4.4.40)、UpdateBatch(See 1.1.4.4.46) 或 CancelBatch(See 1.1.4.4.7) 方法，或者设置 **Recordset** 对象的 Filter(See 1.1.4.6.28) 属性为书签数组。使用该属性，可检查指定记录为何失败并将问题解决。

1.1.4.6.66 StayInSync 属性 (ADO)

在分级 **Recordset** 对象中指示当父行位置更改时，对基本子记录（即“子集”）的引用是否会更改。

设置和返回值

设置或返回布尔型值。如果设置为 **True**，则父 **Recordset** 对象更改行位置时子集将更新；如果设置为 **False**，则子集将继续引用先前子集的数据而不管父 **Recordset** 对象是否更改行位置。

说明

该属性应用于分级记录集，例如由 **MSDataShape** 提供者支持的记录集，而且必须在父 **Recordset** 上设置才能检索子 **Recordset**。此属性简化分级记录集的定位。

1.1.4.6.67 Type 属性 (ADO)

指示 **Parameter**、**Field** 或 **Property** 对象的操作类型或数据类型。

设置和返回值

设置或返回下列 **DataTypeEnum** 值之一。相应的 OLE DB 类型标识符在下表的说明栏的括号中给出。有关 OLE DB 数据类型的详细信息，请参阅第 10 章和《OLE DB 程序员参考》的附录 A。

常量	说明
AdArray	与其他类型一起加入逻辑 **OR** 以指示该数据是那种类型的安全数组 (DBTYPE_ARRAY)。
AdBigInt	8 字节带符号的整数 (DBTYPE_I8)。
AdBinary	二进制值 (DBTYPE_BYTES)。
AdBoolean	布尔型值 (DBTYPE_BOOL)。
adByRef	与其他类型一起加入逻辑 **OR** 以指示该数据是其他类型数据的指针 (DBTYPE_BYREF)。
adBSTR	以空结尾的字符串 (Unicode) (DBTYPE_BSTR)。
adChar	**字符串**值 (DBTYPE_STR)。
adCurrency	货币值 (DBTYPE_CY)。货币数字的小数点位置固定、小数点右侧有四位数字。该值保存为 8 字节范围为 10,000 的带符号整型值。
adDate	**日期**值 (DBTYPE_DATE)。日期按双精度型数值来保存，数字全部表示从 1899 年 12 月 30 开始的日期数。小数部分是一天当中的片段时间。

adDBDate	日期值 (*yyyymmdd*) (DBTYPE_DBDATE)。
adDBTime	时间值 (*hhmmss*) (DBTYPE_DBTIME)。
adDBTimeStamp	时间戳（*yyyymmddhhmmss* 加 10 亿分之一的小数）(DBTYPE_DBTIMESTAMP).
adDecimal	具有固定精度和范围的精确数字值 (DBTYPE_DECIMAL)。
adDouble	双精度浮点值 (DBTYPE_R8)。
adEmpty	未指定值 (DBTYPE_EMPTY)。
adError	32 - 位错误代码 (DBTYPE_ERROR)。
adGUID	全局唯一的标识符 (GUID) (DBTYPE_GUID)。
adIDispatch	OLE 对象上 **Idispatch** 接口的指针 (DBTYPE_IDISPATCH)。
adInteger	4 字节的带符号整型 (DBTYPE_I4)。
adIUnknown	OLE 对象上 **IUnknown** 接口的指针 (DBTYPE_IUNKNOWN)。
adLongVarBinary	长二进制值（仅用于 **Parameter** 对象）。
adLongVarChar	长**字符串**值（仅用于 **Parameter** 对象）。
adLongVarWChar	以空结尾的长字符串值（仅用于 **Parameter** 对象）。
adNumeric	具有固定精度和范围的精确数字值 (DBTYPE_NUMERIC)。
adSingle	单精度浮点值 (DBTYPE_R4)。
adSmallInt	2 字节带符号整型 (DBTYPE_I2)。
adTinyInt	1 字节带符号整型 (DBTYPE_I1)。
adUnsignedBigInt	8 字节不带符号整型 (DBTYPE_UI8)。
adUnsignedInt	4 字节不带符号整型 (DBTYPE_UI4)。
adUnsignedSmallInt	2 字节不带符号整型 (DBTYPE_UI2)。
adUnsignedTinyInt	1 字节不带符号整型 (DBTYPE_UI1)。
adUserDefined	用户定义的变量 (DBTYPE_UDT)。
adVarBinary	二进制值（仅 **Parameter** 对象）。
adVarChar	**字符串**值（仅 **Parameter** 对象）。
adVariant	自动**变体型** (DBTYPE_VARIANT)。
adVector	与其他类型一起加入逻辑 **OR** 中，指示数据是 DBVECTOR 结构（由 OLE DB 定义）。该结构含有元素的计数和其他类型 (DBTYPE_VECTOR) 数据的指针。

adVarWChar	以空结尾的 Unicode 字符串（仅 **Parameter** 对象）。
adWChar	以空结尾的 Unicode 字符串 (DBTYPE_WSTR)。

说明

对 **Parameter** 对象，**Type** 属性是读/写。对其他所有对象，**Type** 属性是只读。

1.1.4.6.68 UnderlyingValue 属性 (ADO)

指示数据库中 **Field** 对象的当前值。

返回值

返回**变体型**值。

说明

使用 **UnderlyingValue** 属性从数据库返回当前字段的值。**UnderlyingValue** 属性中的字段值可见于事务并可能是另一个事务最近更新的结果。这一点不同于 OriginalValue(See 1.1.4.6.46) 属性，该属性反映的是最初返回到 Recordset(See 1.1.4.2.10) 的值。

与使用 Resync(See 1.1.4.4.40) 方法类似，但 **UnderlyingValue** 属性仅从当前记录返回指定字段的值。该值与 **Resync** 方法用于替换 Value(See 1.1.4.6.69) 属性的值相同。

当与 **OriginalValue** 属性一起使用该属性时，可以解决批更新时出现的冲突。

1.1.4.6.69 Value 属性 (ADO)

指示赋给 **Field**、**Parameter** 或 **Property** 对象的值。

设置和返回值

设置或返回**变体型**值。默认值取决于 Type(See 1.1.4.6.67) 属性。

说明

使用 **Value** 属性可以设置或返回来自 **Field** 对象的数据、**Parameter** 对象的参数值、或者 **Property** 对象的属性设置。**Value** 属性是否为读/写或只读取决于许多因素 — 详细信息，请参阅有关对象主题。

ADO 允许使用 **Value** 属性设置和返回长二进制的数据。

注意 对于 **Parameter** 对象，ADO 从提供者读取 **Value** 属性仅一次。如果命令包含其 **Value** 属性为空的 **Parameter**，并且您是通过命令创建 **Recordset**，那么请确保在恢复 **Value** 属性之前首先关闭 **Recordset**。否则，对某些提供者 **Value** 属性可能为空，因而不包含正确的值。

1.1.4.6.70 Version 属性 (ADO)

指示 ADO 版本号。

返回值

返回**字符串**值。

说明

使用 **Version** 属性返回 ADO 执行的版本号。

提供者的版本作为动态属性可从 Properties(See 1.1.4.3.4) 集合中获得。

1.1.4.7 ADO 动态属性

属性	说明
Name(See 1.1.4.7.1)	指定 **Recordset** 对象的名称。
Unique Table 、 Unique Schema 、 Unique Catalog(See 1.1.4.7.2)	**Unique Table** 指定一个允许进行更新、插入和删除的基本表的名称。 **Unique Schema** 指定模式，即表的所有者的名称。 **Unique Catalog** 指定目录，即包含表的数据库的名称。
Resync Command(See 1.1.4.7.3)	指定用户提供的命令字符串，**Resync** 方法发出该字符串用于刷新在由 **Unique Table** 动态属性所命名的表中的数据。
Update Resync(See 1.1.4.7.4)	指定在 **UpdateBatch** 方法之后是否进行隐式 **Resync** 方法操作，以及如果这样，该操作的范围。

1.1.4.7.1 Name 属性--动态 (ADO)

指定 **Recordset** 对象的名称。

返回值

返回作为 **Recordset** 名称的字符串值。

说明

名称在连接期间或在 **Recordset** 关闭之前持久存在。

Name 属性主要为了用于 Microsoft Data Shaping Service for OLE DB(See 1.1.5.7) 服务提供者的重构形特性。要参与重构形，名称必须是唯一的。

该属性是只读的，但在创建 **Recordset** 时可以间接设置。例如，如果使用 SHAPE 命令短语创建 **Recordset**，并使用“AS”关键

字命名别名，则别名将被赋给 **Name** 属性。如果别名未声明，或别名与现有名称相冲突，则 **Name** 属性将由 Shape 提供者生成的名称确定。

当要在 SHAPE 命令中引用 **Recordset** 时，请使用 **Name** 属性，但由于它是由 **MSDataShape** 生成的，所以用户不知道其名称。这时，应使用 **Name** 属性返回的字符串来组成命令，以此生成 SHAPE 命令。

当 CursorLocation(See 1.1.4.6.17) 属性设置为 **adUseClient** 时，**Name** 是追加到 **Recordset** 对象的 **Properties** 集合中的动态属性。

1.1.4.7.2 Unique Table、Unique Schema、Unique Catalog 属性--动态 (ADO)

使用户能够直接控制在通过对多个基本表执行 JOIN 操所得到的 **Recordset** 中的特定基本表的修改。

- **Unique Table** 指定一个允许进行更新、插入和删除的基本表的名称。
- **Unique Schema** 指定模式，即表的所有者的名称。
- **Unique Catalog** 指定目录，即包含表的数据库的名称。

设置和返回值

设置或返回字符串值，该值是表、模式或目录的名称。

说明

所要的基本表通过其目录、模式和表名唯一标识。设置 **Unique Table** 属性后，可使用 **Unique Schema** 或 **Unique Catalog** 属性的值查找基本表。在设置 **Unique Table** 属性之前需要设置 **Unique Schema** 和 **Unique Catalog** 属性的其中一个属性或同时设置两个属性，但这不是必须的。

Unique Table 的主键被用作整个 **Recordset** 的主键。该键可用于任何需要主键的方法。

设置 **Unique Table** 后，AddNew(See 1.1.4.4.1)、Delete(See 1.1.4.4.20)、Resync(See 1.1.4.4.40)、Update(See 1.1.4.4.45) 和 UpdateBatch(See 1.1.4.4.46) 方法只能影响到所命名的表。

如果找不到唯一的基本表，将出现运行时错误。

当把 CursorLocation(See 1.1.4.6.17) 属性设置为 **adUseClient** 时，这些动态属性均被追加到 **Recordset** 对象的 **Properties** 集合中。

1.1.4.7.3 Resync Command 属性--动态 (ADO)

指定用户提供的命令字符串，Resync(See 1.1.4.4.40) 方法发出该字符串用于刷新在由 Unique Table(See 1.1.4.7.2) 动态属性所命名的表中的数据。

设置和返回值

设置或返回字符串，是命令字符串。

说明

Recordset 对象是对多个基本表执行 JION 操作的结果。受影响的行取决于 **Resync** 方法的 **AffectRecords** 参数。如果没有设置 **Unique Table** 和 **Resync Command** 属性，将执行标准的 **Resync** 方法。

Resync Command 属性的命令字符串是唯一标识正在被刷新的行的参数化命令或存储过程，并返回包含相同列的数目和顺序的单个行作为要刷新的行。命令字符串包含在 **Unique Table** 中每个主键列的参数，否则将返回运行时错误。参数将以要刷新行的主键值自动填充。

两个基于 SQL 的范例如下：

1) **Recordset** 由命令定义：

```
SELECT * FROM Customers JOIN Orders ON
Customers.CustomerID = Orders.CustomerID
WHERE city = ‘Seattle’
ORDER BY CustomerID
```

Resync Command 属性设置为：

```
"SELECT * FROM
(SELECT * FROM Customers JOIN Orders
ON Customers.CustomerID = Orders.CustomerID
city = ‘Seattle’ ORDER BY CustomerID)
WHERE Orders.OrderID = ?"
```

Unique Table 为 ***Orders***，其主键 ***OrderID*** 被参数化。子选择提供简单的方法，在程序中确保返回的列具有与使用原始命令相同的数目和顺序。

2) **Recordset** 由存储过程定义：

```
CREATE PROC Custorders @CustomerID char(5) AS
SELECT * FROM Customers JOIN Orders ON
Customers.CustomerID = Orders.CustomerID
WHERE Customers.CustomerID = @CustomerID
```

Resync 方法应该执行如下存储过程：

```
CREATE PROC CustordersResync @ordid int AS
SELECT * FROM Customers JOIN Orders ON
Customers.CustomerID = Orders.CustomerID
WHERE Orders.ordid  = @ordid
```

Resync Command 属性设置为：

```
"{call CustordersResync (?)}"
```

又一次，**Unique Table** 为 ***Orders***，其主键 ***OrderID*** 被参数化。

当把 CursorLocation(See 1.1.4.6.17) 属性设置为 **adUseClient** 时，**Resync Command** 是追加到 **Recordset** 对象的 **Properties** 集合的动态属性。

1.1.4.7.4 Update Resync 属性--动态 (ADO)

指定在 UpdateBatch(See 1.1.4.4.46) 方法之后是否进行隐式 Resync(See 1.1.4.4.40) 方法操作，以及如果这样，该操作的范围。

设置和返回值

设置或返回如下 **CEResyncEnum** 值之一：

常量	说明

adResyncNone	不调用 **Resync**。
AdResyncAutoIncrement	（默认）对所有成功插入的行调用 **Resync**，包括它们自动递加列的值。
adResyncConflicts	对所有因并发冲突导致更新或删除操作失败的行调用 **Resync**。
adResyncUpdates	对所有成功更新的行调用 **Resync**。
adResyncInserts	对所有成功插入的行调用 **Resync**，包括它们相同列的值。
adResyncAll	对出现挂起更改的每行调用 **Resync**。

说明

只有已设置 **Unique Table** 动态属性，该属性才可以应用。

adResyncAutoIncrement 和 **adResyncConflicts** 可以同时使用。**adResyncAll**、**adResyncUpdates**、**adResyncInserts** 和 **adResyncConflicts** 可以同时使用。

常量 **adResyncConflicts** 将 resync 值作为基本值存储，但不覆盖挂起的更改。

当把 CursorLocation(See 1.1.4.6.17) 属性设置为 **adUseClient** 时，**Update Resync** 是追加到 **Recordset** 对象 **Properties** 集合的动态属性。

1.1.5 通过 ADO 使用 OLE DB 提供者

本节内容说明三种提供者：数据提供者、服务提供者和服务组件。提供者分为两类，提供数据的提供者和提供服务的提供者。数据提供者拥有其自己的数据并将数据以表的格式显露给应用程序。服务提供者通过产生和消费数据将服务封装，使 ADO 应用程序中的功能得以扩大。服务提供者也可以进一步定义为服务组件，服务组件必须连同其他服务提供者或组件一起工作。

数据提供者

ADO 之所以具有强大的功能和灵活性，是由于它可以连接到不同的数据提供者并仍能使用相同的编程模式，而不管给定提供者的特定特性。

然而，由于每个提供者都是唯一的，所以应用程序与 ADO 交互作用的方式在不同的提供者之间略有差别。需要注意的差别通常归于以下三种类型之一：

- **ConnectionString** 属性中的连接参数。
- **Command** 对象的用法。
    - 特定提供者的 **Recordset** 行为。

以下为当前每个可用的 Microsoft Provider 列出了三个区域中特定提供者的详细资料。

区域	**主题**
ODBC 数据库	Microsoft OLE DB Provider for ODBC(See 1.1.5.1)

Microsoft® Index Server	Microsoft OLE DB Provider for Microsoft Index Server(See 1.1.5.2)
Microsoft® Active Directory Service	Microsoft OLE DB Provider for Microsoft Active Directory Service(See 1.1.5.3)
Microsoft® Jet 数据库	OLE DB Provider for Microsoft Jet(See 1.1.5.4)
Microsoft® SQL Server	Microsoft OLE DB Provider for SQL Server(See 1.1.5.5)
Oracle 数据库	Microsoft OLE DB Provider for Oracle(See 1.1.5.6)

特定提供者的动态属性

Connection、**Command** 和 **Recordset** 对象的 **Properties** 集合包括特定提供者的动态属性。除了 ADO 所支持的内置属性之外，这些属性提供提供者的特定功能信息。

建立连接和创建这些对象后，使用对象的 **Properties** 集合的 **Refresh** 方法可获得提供者特定属性。有关这些动态属性的详细信息，请查阅提供者文档和 OLE DB 程序员手册。

服务提供者

要使用服务提供者，必须提供关键字。同时，应当知道与每个服务提供者相关联的、特定提供者的动态属性。当前可从 Microsoft 获得的每个服务提供者的特定提供者详细资料开列如下：

- Microsoft Data Shaping Service for OLE DB(See 1.1.5.7)
- Microsoft OLE DB Persistence Provider(See 1.1.5.8)
- Microsoft OLE DB Remoting Provider(See 1.1.5.9)

服务组件

Cursor Service for OLE DB 服务组件补充了数据提供者的游标支持功能。它也需要关键字并具有动态属性。

Microsoft Cursor Service for OLE DB(See 1.1.5.10)

参阅 有关 OLE DB Provider 的详细信息，请查阅数据 Data Access SDK 中的 Microsoft OLE DB 文档或访问 http://www.microsoft.com/data 处的 Microsoft Data Access Web 页。

1.1.5.1 Microsoft OLE DB Provider for ODBC

对于 ADO 或 RDS 的程序员来说，理想的环境是每个数据源都具有一个 OLE DB 接口，以便 ADO 可以直接调用该数据源。虽然越来越多数据库厂商提供 OLE DB 接口，但某些数据源仍无法以这种方式提供。然而，当前使用的所有 DBMS 系统实际上都可以通过 ODBC 进行访问。

Microsoft® ODBC Provider 允许 ADO 连接到任何 ODBC 的数据源。ODBC 驱动程序对于当今使用的各种主要 DBMS 都有效，包括 Microsoft® SQL Server®、Microsoft Access（Microsoft Jet 数据库引擎）和 Microsoft FoxPro® 以及诸如 Oracle 等非 Microsoft 数据库产品。提供者将不受线程控制并允许使用 unicode。

提供者将支持事务，尽管不同的 DBMS 引擎提供不同类型的事务支持。例如，Microsoft Access 支持五级或五级以下的嵌套事务。该提供者是 ADO 的默认提供者，当与 Microsoft SQL Server 6.5 一起使用时，除 ADO 语言手册主题中注释之外，所有依赖于提供者的 ADO 属性和方法都受到支持。

连接字符串参数

要连接该提供者，将（**ConnectionString** 属性的参数）“**Provider**=”设置为：

MSDASQL

读取 **Provider** 属性也将返回该字符串。

由于这是 ADO 的默认提供者，所以如果省略连接字符串的 **Provider**= 参数，ADO 将试图建立与该提供者的连接。

除了 ADO 所定义的参数外，提供者不支持任何特定连接参数。但是，提供者将把任何非 ADO 连接参数传递给 ODBC 驱动程序管理器。

由于可以省略 Provider 参数，因此使用与撰写 ODBC 连接字符串时用的相同参数名（DRIVER=、DATABASE=、DSN= 等等）、值和语法，可以撰写与同一数据源的 ODBC 连接字符串相同的 ADO 连接字符串。可以使用或不使用预定义的数据源名 (DSN) 或 FileDSN 进行连接。

带有 DSN 或 FileDSN 的语法：

"[**Provider=MSDASQL;**] { **DSN=***name* | **FileDSN=***filename* } **;** [**DATABASE=***database***;**] **UID=***user***; PWD=***password***"**

无 DSN（非 DSN 连接）的语法：

"[**Provider=MSDASQL;**] **DRIVER=***driver***; SERVER=***server***; DATABASE=***database***; UID=***user***; PWD=***password***"**

如果使用 **DSN** 或 **FileDSN**，则必须通过“Windows 控制面板”中的“ODBC 管理器”进行定义。作为设置 **DSN** 的替换方法，可以指定 ODBC 的驱动程序 (**DRIVER=**)，诸如“SQLServer”、服务器名 (**SERVER=**) 和数据库名 (**DATABASE=**)。

也可以在特定 ODBC 的参数或标准 ADO 定义的 **User ID** 和 **Password** 参数中为用户帐号 (**PWD=**) 指定用户帐号名 (**UID=**) 和密码。如果这些值中同时包括了 ADO 和特定 ODBC 的参数，则 ADO 参数优先。

即使 **DSN** 定义已经指定了数据库，也可以在 **DSN** 之外指定 **DATABASE** 参数以便连接到不同的数据库。这同时更改了 **DSN** 定义以包括指定的数据库。使用 **DSN** 时始终包括 **DATABASE** 参数是一种好办法。这样将保证能连接到正确的数据库，因为其他用户可能会在上一次检查 **DSN** 定义后更改默认的数据库参数。

Command 文本

如何使用 **Command** 对象很大程度上取决于数据源和它所接受的查询类型或命令语句。

ODBC 对于调用存储的过程提供特定的语法。对于 **Command** 对象的 **CommandText** 属性、**Connection** 对象上 **Execute** 方法的 **CommandText** 参数、或 **Recordset** 对象上 **Open** 方法的 **Source** 参数，请传递下列语法的字符串：

"{ [**?** =] **call** *procedure* [**(** **?** [**,** **?** [**,** _]] **)**] **}"**

此处每个 **?** 引用 **Parameters** 集合中的一个对象。第一个 **?** 引用 **Parameters**(0)，下一个 **?** 引用 **Parameters**(1)，依次类推。

参数引用是可选的并取决于存储过程的结构。如果要调用未定义参数的存储过程，则字符串如下：

"{ call *procedure* **}"**

如果有两个查询参数，则字符串如下：

"{ call *procedure* **(** **?, ?** **) }"**

如果存储过程要返回一个值，返回值将被作为另一个参数来对待。如果无查询参数而有返回值，则字符串如下：

"{ ? = call *procedure* **}"**

最后，如果有返回值和两个查询参数，则字符串如下：

"{ ? = call *procedure* **(** **?, ?** **) }"**

Recordset 行为

下表列出了由该提供者打开的 **Recordset** 对象上可用的标准 ADO 方法和属性。

要获得有关提供者配置的 **Recordset** 行为的详细信息，请运行 **Supports** 方法并枚举 **Recordset** 的 **Properties** 集合以确定特定提供者的动态属性是否存在。

标准 ADO **Recordset** 属性的可用性：

属性	仅向前	动态	键集	静态
AbsolutePage	不可用	不可用	读/写	读/写

AbsolutePosition	不可用	不可用	读/写	读/写
ActiveConnection	读/写	读/写	读/写	读/写
BOF	只读	只读	只读	只读
Bookmark	不可用	不可用	读/写	读/写
CacheSize	读/写	读/写	读/写	读/写
CursorLocation	读/写	读/写	读/写	读/写
CursorType	读/写	读/写	读/写	读/写
EditMode	只读	只读	只读	只读
EOF	只读	只读	只读	只读
Filter	读/写	读/写	读/写	读/写
LockType	读/写	读/写	读/写	读/写
MarshalOptions	读/写	读/写	读/写	读/写
MaxRecords	读/写	读/写	读/写	读/写
PageCount	不可用	不可用	只读	只读
PageSize	读/写	读/写	读/写	读/写
RecordCount	不可用	不可用	只读	只读
Source	读/写	读/写	读/写	读/写
State	只读	只读	只读	只读
Status	只读	只读	只读	只读

当 ADO 与 ODBC 的 Microsoft OLE DB Provider 1.0 版本一起使用时，**AbsolutePosition** 和 **AbsolutePage** 属性是只写的。

标准 ADO **Recordset** 方法的可用性：

方法	**仅向前**	**动态**	**键集**	**静态**
AddNew	是	是	是	是
Cancel				
CancelBatch	是	是	是	是
CancelUpdate	是	是	是	是
Clone	否	否	是	是
Close	是	是	是	是

Delete	是	是	是	是
GetRows	是	是	是	是
Move	是	是	是	是
MoveFirst	是	是	是	是
MoveLast	否	是	是	是
MoveNext	是	是	是	是
MovePrevious	否	是	是	是
NextRecordset*	是	是	是	是
Open	是	是	是	是
Requery	是	是	是	是
Resync	否	否	是	是
Supports	是	是	是	是
Update	是	是	是	是
UpdateBatch	是	是	是	是

*不支持 Microsoft Access 数据库。

参阅 有关 ODBC 的 Microsoft OLE DB Provider 的详细实现资料和功能信息，请查阅 ODBC 的 Microsoft OLE DB Provider 文档和《Microsoft OLE DB 程序员手册》，可以在 Data Access SDK 中得到。也可查阅 http://www.microsoft.com/data 处的数据访问 Web 页。

1.1.5.2 Microsoft OLE DB Provider for Microsoft Index Server

Microsoft OLE DB Provider for Microsoft Index Server 提供了对文件系统和由 Microsoft® Index Server 2.0 版编写索引的 Web 数据的可编程只读访问。ADO 应用程序可以发布 SQL 查询以检索内容和文件属性信息。

提供者将不受线程控制并允许使用 unicode。

连接字符串参数

要连接到该提供者，请将（**ConnectionString** 属性的参数）“**Provider**=”设置为：

```
MSIDXS
```

读取 **Provider** 属性也将返回该字符串。

命令文本

索引服务器 SQL 查询语法由 SQL92 **SELECT** 语句的扩展以及 **FROM** 和 **HERE** 子句组成。查询的结果将通过 OLE DB 行集合返回，这些结果可以被 ADO 使用并作为 **Recordset** 对象进行操作。

可以搜索准确的词或短语，或使用通配符搜索词的模式或出处。搜索逻辑可以基于布尔结果，即确定重要项与其他词之间的相近性。也可以通过“自由文本”进行搜索，该文本可根据意思而不是准确词找到匹配之处。

特定命令语法完全归档于“对索引服务器数据的 SQL 访问”下的《Microsoft Index Server 手册》当中。

提供者不接受存储过程调用或简单的表名（例如，**CommandType** 属性将总是 **adCmdText**）。

Recordset 行为

下表列出了由该提供者打开的 **Recordset** 对象的可用功能。只有静态游标类型 **(adOpenStatic)** 可用。

要获得有关提供者配置的 **Recordset** 行为的详细信息，请运行 **Supports** 方法并枚举 **Recordset** 的 **Properties** 集合以确定特定提供者的动态属性是否存在。

标准 ADO **Recordset** 属性的可用性：

属性	可用性
AbsolutePage	读/写
AbsolutePosition	读/写
ActiveConnection	只读
BOF	只读
Bookmark*	读/写
CacheSize	读/写
CursorLocation	总是 adUseServer
CursorType	总是 adOpenStatic
EditMode	总是 adEditNone
EOF	只读
Filter	读/写
LockType	读/写
MarshalOptions	不可用
MaxRecords	读/写
PageCount	只读
PageSize	读/写
RecordCount	只读
Source	读/写
State	只读
Status	只读

* 必须在提供者中激活书签以便 **Recordset** 具备该特性。

标准 ADO **Recordset** 方法的可用性：

方法	可用性
AddNew	否
Cancel	
CancelBatch	否
CancelUpdate	否
Clone	是
Close	是
Delete	否
GetRows	是
Move	是
MoveFirst	是
MoveLast	是
MoveNext	是
MovePrevious	是
NextRecordset	是
Open	是
Requery	是
Resync	是
Supports	是
Update	否
UpdateBatch	否

参阅 有关 Microsoft OLE DB Provider for Microsoft Index Server 的详细实现资料和功能信息，请参阅《Microsoft OLE DB 程序员手册》和 Microsoft Index Server 文档。在 http://www.microsoft.com/iis 处的 Microsoft Internet Information Server Web 页还包含了有关 Microsoft Index Server 的信息。

1.1.5.3 Microsoft OLE DB Provider for Microsoft Active Directory Service

Microsoft® Active Directory Service Interface (ADSI) 提供者允许 ADO 通过 ADSI 连接到不同种类的目录服务。它向 ADO 应用

程序提供对 Microsoft Windows NT® 4.0 目录服务以及任何适合 LDAP 目录服务和 Novell 目录服务的只读访问权。ADSI 本身基于一种提供者模式，所以如果有新的提供者提供对其他目录的访问权，ADO 应用程序将可以对其进行无缝访问。ADSI 提供者将不受线程控制并允许使用 unicode。

连接字符串参数

要连接到该提供者，请将（**ConnectionString** 属性的参数）“**Provider**=”设置为：ADSDSOObject

读取 **Provider** 属性也将返回该字符串。

命令文本

提供者识别下列语法中由四部分组成的命令文本字符串：

"*Root*; *Filter*; *Attributes*[; *Scope*]"

值	说明
Root	搜索启动（即搜索的根）处的 **ADsPath** 对象。
Filter	RFC 960 格式的搜索筛选。
Attributes	要返回的用逗号分隔的属性列表
Scope	可选。指定搜索范围的字符串。可以是以下的一种： • Base — 只搜索基本对象（搜索的根）。 • OneLevel — 只搜索一级 • Subtree — 搜索整个子目录树。

提供者不接受存储的过程调用或简单的表名（例如，**CommandType** 属性将总是 **adCmdText**）。要获得命令文本元素的完整说明，请参阅 Active Directory Service 文档。

Recordset 行为

以下表格列出了由该提供者打开的 **Recordset** 对象的可用功能。只有静态游标类型 (**adOpenStatic**) 是可用的。

要获得有关提供者配置的 **Recordset** 行为的详细信息，请运行 **Supports** 方法并列举 **Recordset** 的 **Properties** 集合以确定特定提供者的动态属性是否存在。

标准 ADO **Recordset** 属性的可用性：

属性	可用性
AbsolutePage	读/写
AbsolutePosition	读/写
ActiveConnection	只读
BOF	只读
Bookmark	读/写
CacheSize	读/写
CursorLocation	总是 **adUseServer**

CursorType	总是 **adOpenStatic**
EditMode	总是 **adEditNone**
EOF	只读
Filter	读/写
LockType	读/写
MarshalOptions	不可用
MaxRecords	读/写
PageCount	只读
PageSize	读/写
RecordCount	只读
Source	读/写
State	只读
Status	只读

标准 ADO **Recordset** 方法的可用性：

方法	**可用性**
AddNew	否
Cancel	
CancelBatch	否
CancelUpdate	否
Clone	是
Close	是
Delete	否
GetRows	是
Move	是
MoveFirst	是
MoveLast	是
MoveNext	是
MovePrevious	是

NextRecordset	是
Open	是
Requery	是
Resync	是
Supports	是
Update	否
UpdateBatch	否

参阅 有关常规 ADSI 和提供者说明的详细信息，请查阅 Active Directory Service Interface SDK 中提供的文档。可以从 http://www.microsoft.com/ntserver/info/adsi.htm 处的 ADSI Web 页安装 SDK。

1.1.5.4 OLE DB Provider for Microsoft Jet

OLE DB Provider for Microsoft® Jet 允许 ADO 访问 Microsoft Jet 数据库。

连接字符串参数

要连接到该提供者，请将 **ConnectionString** 属性的 **Provider** 参数设置为：

Microsoft.Jet.OLEDB.4.0

读取 Provider 属性也将返回该字符串。

除了由 ADO 定义的连接参数外，OLE DB Provider for Microsoft Jet 支持几个特定提供者的连接参数。正如所有其他连接参数一样，这些参数可以通过 **Connection** 对象的 **Properties** 集合进行设置或设置为连接字符串的一部分。

参数	**说明**
Jet OLEDB:System Database	工作组信息文件的路径和文件名。
Jet OLEDB:Registry Path	包含 Microsoft Jet 数据库引擎值的 Windows 注册表键。
Jet OLEDB:Database Password	数据库密码。

默认情况下，OLE DB Provider for Microsoft Jet 以读/写模式打开 Microsoft Jet 数据库。要以只读模式打开数据库，请将 ADO **Connection** 对象中的 **Mode** 属性设置为 **adModeRead**。

Command 对象的用法

Command 对象中的命令文本使用 Jet SQL 语言。可以在命令文本中指定行返回查询、操作查询和表名；但不支持存储过程，因此不应进行指定。

Recordset 行为

Microsoft Jet 数据库引擎不支持动态游标。因此，OLE DB Provider for Microsoft Jet 不支持 **adLockDynamic** 游标类型。当请求动态游标时，提供者将返回键集游标并重新设置 **CursorType** 属性以指明返回的 **Recordset** 类型。进一步说，如果请求可更新的 **Recordset**（**LockType** 是 **adLockOptimistic**、**adLockBatchOptimistic** 或 **adLockPessimistic**），提供者也将返回键集游标并重新设置 **CursorType** 属性。

参阅 有关 OLE DB Provider for Microsoft Jet 的详细实现资料和功能信息，请查阅 Data Access SDK 中的 OLE DB Provider for Microsoft Jet 文档。

1.1.5.5 Microsoft OLE DB Provider for SQL Server

The Microsoft® OLE DB Provider for SQL Server (SQLOLEDB) 允许 ADO 访问 Microsoft® SQL Server™。

连接字符串参数

要连接到该提供者，请将 **ConnectionString** 属性的 **Provider** 参数设置为：

```
SQLOLEDB
```

也可以使用 **Provider** 属性设置或读取该值。

除了 ADO 定义的连接参数外，提供者支持几个特定提供者的连接参数。与 ADO 连接属性一样，这些特定提供者的属性可以通过 **Connection** 的 **Properties** 集合设置或者设置为 **ConnectionString** 的一部分。

参数	说明
Trusted Connection	用户身份验证模式。可以设置为 **True** 或 **False**。默认值是 **False**。如果将属性设置为 **True**，则 SQLOLEDB 将使用 Microsoft® Windows NT® 身份验证模式授权用户访问由 **Location** 和 **Datasource** 属性值指定的 SQL 服务器数据库。如果将该属性设置为 **False**，则 SQLOLEDB 将使用“混合模式”授权用户访问 SQL 服务器数据库。SQL 服务器的登录和密码在 **User Id** 和 **Password** 属性中指定。
Current Language	SQL 服务器语言名称。识别系统信息选择和格式化所使用的语言。该语言必须安装在 SQL 服务器上，否则打开连接时会失败。
Network Address	由 **Location** 属性指定的 SQL 服务器的网络地址。
Network Library	用来与 SQL 服务器进行通讯的网络库 (DLL) 名。名称不应包括路径或 .dll 文件名的扩展名。默认名由 SQL 服务器客户配置来提供。
Use Procedure for Prepare	SQL 服务器存储过程使用。准备命令时定义 SQL 服务器临时存储过程的使用。
Auto Translate	OEM/ANSI 字符转换。该属性可设置为 **True** 或 **False**。默认值是 **True**。如果将该属性设置为 **True**，则从 SQL 服务器提取多字节字符串或将其发送到 SQL 服务器时，SQLOLEDB 将执行 OEM/ANSI 字符转换。如果将该属性设置为 **False**，则 SQLOLEDB 不在多字节字符串数据上执行 OEM/ANSI 字符转换。
Packet Size	以字节表示的网络包的大小。包大小的属性值必须在 512 和 32767 之间。默认的 SQLOLEDB 网络包大小是 4096。
Application Name	客户应用程序名。
Workstation ID	标识工作站的字符串。

Command 对象的用法

SQLOLEDB 将 ODBC、ANSI 和特定 SQL 服务器的 Transact-SQL 的混合体作为有效的语法。例如，以下的 SQL 语句使用 ODBC SQL Esc 转义序列来指定 LCASE 字符串函数：

```
SELECT customerid={fn LCASE(CustomerID)} FROM Customers
```

LCASE 返回字符串，将所有大写字符转换成相应的小写字符。ANSI SQL 的字符串函数 LOWER 执行相同的操作，因此，以下的 SQL 语句与上述 ODBC 语句的 ANSI 等价：

```
SELECT customerid=LOWER(CustomerID) FROM Customers
```

当被指定为命令的文本时，SQLOLEDB 将成功地处理任何一种语句的窗体。

存储过程

当使用 SQLOLEDB 命令执行 SQL 服务器的存储过程时，请使用命令文本的 ODBC 过程调用 Esc 转义序列。而后，SQLOLEDB 将使用 SQL 服务器的远程过程调用机制来优化命令处理。例如，以下的 ODBC SQL 语句是 Transact-SQL 窗体上的首选命令文本：

ODBC SQL

```
{call SalesByCategory('Produce', '1995')}
```

Transact-SQL

```
EXECUTE SalesByCategory 'Produce', '1995'
```

Recordset 行为

SQLOLEDB 不能使用 SQL 服务器游标支持由许多命令生成的多行集合结果。如果客户请求需要 SQL 服务器游标支持的记录集，则所使用的命令文本在其结果中生成多个记录集时将产生错误。

SQL 服务器游标支持可滚动的 SQLOLEDB 记录集。SQL 服务器限制对其他数据库用户所做的更改敏感的游标。特别是一些游标中的行不能排序，试图使用包含 SQL ORDER BY 子句的命令创建行集合会造成失败。

参阅 有关 Microsoft SQL Server OLE DB Provider 的特定执行的详细资料和功能信息，请查阅 Data Access SDK 中的 Microsoft SQL Server OLE DB Provider 文档。

1.1.5.6 Microsoft OLE DB Provider for Oracle

Microsoft OLE DB Provider for Oracle 允许 ADO 访问 Oracle 数据库。

连接字符串参数

要连接到该提供者，将 **ConnectionString** 属性的 **Provider** 参数设置为：

```
MSDAORA
```

读取 **Provider** 属性也将返回该字符串。

如果在 Oracle 数据库中使用键集或动态游标执行 join 查询，则会发生错误。Oracle 仅支持静态只读游标。

1.1.5.7 Microsoft Data Shaping Service for OLE DB (ADO Service Provider)

Microsoft Data Shaping Service for OLE DB 服务提供者支持来自一个或多个数据提供者的分级（成形）**Recordset** 对象的结构。

提供者关键字

要调用 Data Shaping Service for OLE DB，请在连接字符串中指定如下关键字和值。

```
"Provider=MSDataShape"
```

动态属性

当调用该服务提供者时，将把如下动态属性添加到 **Connection** 对象的 **Properties** 集合中。

动态属性名称	说明
Unique Reshape Names	指示赋给 **Recordset** 的 **Name** 属性的值，是否会与现有名称相冲突。如果该属性是 **True**，那么将生成唯一名称；否则，两个名称共存。
Data Provider	指示将提供要被构形行的提供者的名称。

通过在连接字符串中将可写动态属性的名称指定为关键字，也可设置这些可写动态属性。例如，在 Visual Basic 中，可通过如下指定将 **Data Provider** 动态属性设置成“MSDASQL”：

```
Dim cn as New ADODB.Connection
cn.Open "Provider=MSDataShape;Data Provider=MSDASQL"
```

通过将动态属性的名称指定为 **Properties** 属性的索引，也可设置或检索动态属性。例如，获得和打印 **Data Provider** 动态属性的当前值，然后设置新值，如：

```
Debug.Print cn.Properties("Data Provider")
cn.Properties("Data Provider") = "MSDASQL"
```

有关数据构形的详细信息，请参阅数据构形(See 1.1.3.7.1)。

1.1.5.8 Microsoft OLE DB Persistence Provider (ADO Service Provider)

Microsoft OLE DB Persistence Provider 允许将 **Recordset** 对象保存到文件中，并在随后从文件恢复 **Recordset** 对象。模式信息、数据和挂起更改将被保存下来。

用于保存 **Recordset** 的格式既可以是高级数据图表 (ADTG) 格式，也可以是可扩展标记语言 (XML) 格式。

提供者关键字

要调用该提供者，请在连接字符串中指定如下关键字和值。

"Provider=**MSPersist**"

错误

在应用程序中可检测到由该提供者发出的如下错误。

常量	说明
E_BADSTREAM	正在打开的文件没有使用有效格式（即，格式不是 ADTG 或 XML）。
E_CANTPERSISTROWSET	正在保存的 **Recordset** 对象设置有防止被保存的特性。

说明

Microsoft OLE DB Persistence Provider 不显露动态属性。

当前，不能保存参数化分级 **Recordset** 对象。

有关持久 **Recordset** 对象的详细信息，请参阅记录集持久性(See 1.1.3.2)。

1.1.5.9 Microsoft OLE DB Remoting Provider (ADO Service Provider)

Microsoft OLE DB Remoting Provider 允许在客户端机器上的本地用户调用远程计算机上的数据提供者。如果您是远程计算机上的本地用户，可指定远程计算机的数据提供者参数。然后指定由远程数据提供者使用的参数来访问远程计算机。最终结果是使您象本地用户一样访问远程计算机。

提供者关键字

要调用 OLE DB Remoting Provider，请在连接字符串中指定如下的关键字和值。（注意提供者名称中的空格。）

"Provider=**MS Remote**"

其他关键字

当调用该服务提供者时，将涉及如下关键字。

关键字	说明
Data Source	指定远程数据源的名称。它将被传递到 OLE DB Remoting Provider 进行处理。 该关键字等同于 **RDS.DataControl** 对象 Connect(See 1.1.4.6.13) 属性。

动态属性

当调用该服务提供者时，会将如下动态属性添加到 **Connection** 对象的 **Properties** 集合中。

动态属性名称	说明
DFMode	指示 DataFactory Mode。该字符串指定服务器上所需的 **DataFactory** 对象版本。在打开请求特定版本的 **DataFactory** 的连接之前，请设置该属性。如果无法得到请求的版本，将尝试使用以前版本。如果没有以前版本，将出现错误。在连接之后，该属性是只读的。 可以是下列有效字符串值： • “21”—2.1 版（默认） • “20”—2.0 版 • “15”—1.5 版
Current DFMode	指示服务器上 **DataFactory** 的实际版本号。检查该属性以便查看在 **DFMode** 属性中请求的版本是否可以得到。 可以是如下可用的 Long 整数值中的一个： • 21—2.1 版（默认） • 20—2.0 版 • 15—1.5 版
Handler	字符串值。指示服务器端用于扩展 **RDSServer.DataFactory** 函数的自定义程序（即处理程序）的名称，以及处理程序所使用的任意参数，各项使用逗号分隔(",")。
Internet Timeout	指示等待请求传输到服务器并返回所需的最长时间。（按毫秒计算，默认值是 5 分钟。）
Remote Provider	指示用于远程服务器的数据提供者的名称。
Remote Server	指示要用于该连接的服务器名和通信协议。该属性等同于 **RDS.DataControl** 对象 Server(See 1.1.4.6.55) 属性。

通过在连接字符串中将可写入动态属性的名称指定为关键字，也可以设置可写入动态属性。例如，通过以下指定将 **Internet Timeout** 动态属性设置为 5 秒钟：

```
Dim cn as New ADODB.Connection
cn.Open "Provider=MS Remote;Internet Timeout=5000"
```

通过将动态属性的名称指定为 **Properties** 属性的索引，也可设置或检索动态属性。例如，获得和打印 **Internet Timeout** 动态属性的当前值，然后设置新值，如：

```
Debug.Print cn.Properties("Internet Timeout")
cn.Properties("Internet Timeout") = 5000
```

说明

在 ADO 2.0 中，OLE DB Remoting Provider 仅能在 **Recordset** 对象 **Open** 方法的 **ActiveConnection** 参数中指定。从 ADO 2.1 开始，也可以在 **Connection** 对象 **Open** 方法的 **ConnectionString** 参数中指定。

没有 **RDS.DataControl** 对象的 **SQL** 属性的等同设置。由 **Recordset** 对象 **Open** 方法 **Source** 参数代替。

举例

这个例子在称为 YourServer 的服务器上，执行对公用数据库创建者表的查询。在 **Connection** 对象的 **Open** 方法中提供远程数据源和远程服务器的名称，并在 **Recordset** 对象的 **Open** 方法中指定 SQL 查询。**Recordset** 对象将被返回、编辑、并用来更新数据源。

```
Dim rs as New ADODB.Recordset
Dim cn as New ADODB.Connection
cn.Open    "Provider=MS Remote;Data Source=pubs;" & _
       "Remote Server=http://YourServer"
rs.Open "SELECT * FROM authors", cn
...                    'Edit the recordset
rs.UpdateBatch             'Equivalent of RDS SubmitChanges
...
```

1.1.5.10 Microsoft Cursor Service for OLE DB (ADO Service Component)

Microsoft Cursor Service for OLE DB 服务组件补充了数据提供者的游标支持功能。其结果，用户可以从所有数据提供者处获得相对统一的功能。

Cursor Service for OLE DB 服务组件使动态属性可用，并增强了某些方法的性能。例如，**Optimize** 动态属性允许创建临时索引来方便某些操作，如 **Find** 方法。

Cursor Service for OLE DB 允许支持在各种情况下的批更新。当数据提供者只能提供功能较少的游标（如静态游标）时，批更新可以模拟功能较多的游标类型（如动态游标）。

关键字

要调用该组件，请将 **Recordet** 或 **Connection** 对象的 **CursorLocation** 属性设置为 **adUseClient**。

```
connection.CursorLocation=adUseClient
recordset.CursorLocation=adUseClient
```

动态属性

当调用 Cursor Service for OLE DB 时，如下动态属性将被添加到 **Recordset** 对象的 **Properties** 集合中。

动态属性名称	说明
Handler(See 1.1.4.6.32) (RDS)	指示是否启用对 **RDSServer.DataFactory** 的服务器端自定义支持。

Name(See 1.1.4.7.1)	指示 **Recordset** 的名称。可能在当前（或随后）的数据形状命令中被引用。
Optimize(See 1.1.4.6.45)	指示是否应创建索引。当设置为 **True** 时，将临时创建索引，以便改善某些操作的执行。
Resync Command(See 1.1.4.7.3)	指定当 **Unique Table** 属性生效时由 **Resync** 方法使用的自定义命令字符串。
Unique Catalog(See 1.1.4.7.2)	指示数据库的名称，该数据库包含在 **Unique Table** 属性中被引用的表。
Unique Schema(See 1.1.4.7.2)	指示在 **Unique Table** 属性中被引用的表的拥有者名称。
Unique Table(See 1.1.4.7.2)	指示在 **Recordset** 中的一个表的名称，该 **Recordset** 由多个可以通过插入、更新或删除操作进行更改的表创建。
Update Resync(See 1.1.4.7.4)	指定当 **Unique Table** 属性生效时，是否在 **UpdateBatch** 方法（和它的行为）后隐式调用 **Resync** 方法。

通过将动态属性的名称指定为 **Properties** 属性的索引，也可以设置或检索该动态属性。例如，获得并打印 **Optimize** 动态属性的当前值，然后设置新值，如：

```
Debug.Print rs.Properties("Optimize")
rs.Properties("Optimize") = True
```

内置属性行为

Cursor Service for OLE DB 也影响某些内置属性的行为。

属性名称	**说明**
CursorType(See 1.1.4.6.18)	补充 **Recordset** 可用的游标类型。
LockType(See 1.1.4.6.37)	补充 **Recordset** 可用的锁定类型。允许批更新。
Sort(See 1.1.4.6.57)	指定 **Recordset** 以之排序的一个或更多个字段名，以及每个字段是否按升序或降序排序。

方法行为

Cursor Service for OLE DB 启用或影响 **Field** 对象的 Append(See 1.1.4.4.2) 方法的行为；以及 **Recordset** 对象的 Open(See 1.1.4.4.33)、Resync(See 1.1.4.4.40)、UpdateBatch(See 1.1.4.4.46) 和 Save(See 1.1.4.4.41) 方法的行为。

1.1.6 学习 ADO

使用如下各节内容学习如何使用 ADO 和 RDS：

- ADO 和 RDS 教程(See 1.1.6.1)
- 远程数据服务的范例应用程序(See 1.1.6.2)
- 远程数据服务开发人员指南(See 1.1.6.3)
- ADO 代码范例(See 1.1.6.4)

1.1.6.1 ADO 和 RDS 教程

- ADO 教程(See 1.1.6.1.1)
- RDS 教程(See 1.1.6.1.2)。该教程说明如何使用 RDS 编程模型来查询和更新数据源。
- 建立简单的远程数据服务应用程序(See 1.1.6.1.3)。该教程教您建立简单的远程数据服务应用程序，实现对 Microsoft® Access 数据库中活动数据的访问。
- 地址簿(See 1.1.6.1.4)。该教程教您如何创建简单的远程数据服务应用程序，实现对 Microsoft® SQL Server® 数据库的访问和更新。

1.1.6.1.1 ADO 教程

本教程说明如何使用 ADO 编程模型对数据源进行查询及更新。教程首先讲述了完成该任务的必要步骤，然后分别通过 Microsoft® Visual Basic®、具有 VC++ Extensions 特性的 Microsoft® Visual C++®、Microsoft® Visual Basic®, Scripting Edition 以及具有 ADO for Windows Foundation Classes (ADO/WFC) 特性的 Microsoft® Visual J++® 进行重复说明。
本教程使用了不同语言的代码，主要有以下两个原因：

- 假设 ADO 文档的读者使用 Visual Basic 编码。这样使得文档对 Visual Basic 编程人员有用，但对于使用其他语言的编程人员则没有多少用处。
- 如果您对特定的 ADO 功能不十分熟悉，但对于其他语言有所了解，那么可以通过在其他语言中寻求相同的功能来达到目的。

教程是如何编写的

本教程基于 ADO 编程模型，并对该模式的每一个步骤分别进行讨论，另外，使用了 Visual Basic 代码段举例讲述每个步骤。最后对整个过程进行重述，并将代码段整合为一个完整的 Visual Basic 范例。
代码范例使用其他语言重复演示，但未予讨论。给定的编程语言教程中的每个步骤以编程模型和叙述性教程中的相应步骤加以注明，使用步骤编号以便在叙述性教程中查询有关问题的讨论。
由于本教程由若干个小的代码段组成，因此无法按照所述说明执行这些代码。
ADO 编程模型将在后面重新叙述，可以在阅读教程时将其作为路标。

ADO 对象编程模型

- 连接数据源 (**Connection**)，可选择开始事务。

- 可选择创建表示 SQL 命令的对象 (**Command**)。
- 可选择指定列、表以及 SQL 命令中的值作为变量参数 (**Parameter**)。
- 执行命令（**Command**、**Connection** 或 **Recordset**）。
- 如果命令以行返回，将行存储在存储对象中 (**Recordset**)。
- 可选择创建存储对象的视图以便进行排序、筛选和定位数据 (**Recordset**)。
- 编辑数据。可以添加、删除或更改行、列 (**Recordset**)。
- 在适当情况下，可以使用存储对象中的变更对数据源进行更新 (**Recordset**)。
- 在使用事务之后，可以接受或拒绝在事务中所做的更改。结束事务 (**Connection**)。

下一页 步骤 1(See 1.1.6.1.1.1)

1.1.6.1.1.1 步骤 1：打开连接（ADO 教程）

您所在的步骤...

- **连接数据源。**
  - 可选择创建表示 SQL 查询命令的对象。
  - 可选择在 SQL 命令中将值指定为变量参数。
  - 执行命令。如果命令以行返回，将行存储在存储对象中。
  - 可选择对数据进行定位、检查、操作和编辑。
  - 适当情况下，可以使用存储对象中的变更对数据源进行更新。可选择在事务处理中嵌入更新数据。
  - 在使用事务之后，可以接受或拒绝在事务中所做的更改。结束事务。

讨论

如果需要一种途径以建立交换数据所必须的条件，那就是“连接”。所连接的数据源可在“连接字符串”中指定，但是对于不同的提供者和数据源而言，连接字符串中指定的参数会有所不同。

ADO 打开连接的主要方法是使用 **Connection.Open** 方法。另外也可在同一个操作中调用快捷方法 **Recordset.Open** 打开连接并在该连接上发出命令。以下是 Visual Basic 中用于两种方法的语法：

```
connection.Open ConnectionString, UserID, Password, OpenOptions
recordset.Open Source, ActiveConnection, CursorType, LockType, Options
```

比较这两种方法将有益于加深对 ADO 方法操作数的总体了解。

ADO 提供了多种指定操作数的简便方式。例如：**Recordset.Open** 带有 **ActiveConnection** 操作数，该操作数可以是文字字符串（表示字符串的变量），或者是代表一个已打开的连接的 **Connection** 对象。

对象中的多数方法具有属性，当操作数缺省时属性可以提供参数。使用 **Connection.Open**，可以省略显式 **ConnectionString** 操作数并通过将 **ConnectionString** 的属性设置为“DSN=pubs;uid=sa;pwd=;database=pubs”隐式地提供信息。

与此相反，连接字符串中的关键字操作数 **uid** 和 **pwd** 可为 **Connection** 对象设置 **UserID** 和 **Password** 参数。

本教程使用显式连接字符串调用 **Connection.Open** 方法，数据源是“开放式数据库连接”(ODBC) **pubs** 数据库，它作为测试数据库与 Microsoft SQL Server 一同发布。（数据源的实际位置，如本地驱动器或远程服务器，在定义“数据源名称”(DSN) 时进行指定。）

```
connection.Open "DSN=pubs;uid=sa;pwd=;database=pubs"
```

下一页 步骤 2(See 1.1.6.1.1.2)

1.1.6.1.1.2 步骤 2：创建命令（ADO 教程）

您所在的步骤...

- 连接数据源。
- **可选择创建表示 SQL 查询命令的对象。**
- **可选择在 SQL 命令中将值指定为变量参数。**
- 执行命令。如果命令以行返回，将行存储在存储对象中。
- 可选择对数据进行定位、检查、操作和编辑。
- 适当情况下，可以使用存储对象中的变更对数据源进行更新。可选择在事务处理中嵌入更新数据。
- 在使用事务之后，可以接受或拒绝在事务中所做的更改。结束事务。

讨论

查询命令要求数据源返回含有所要求信息行的 **Recordset** 对象。命令通常使用 SQL 编写。

1. 如上所述，“命令字符串”之类的操作数可表示为：
   - 代表字符串的文字串或变量。本教程可使用命令字符串“SELECT * from authors”查询 **pubs** 数据库中的 **authors** 表中的所有信息。
   - 代表命令字符串的对象。在这种情况下，**Command** 对象的 **CommandText** 属性的值设置为命令字符串。
   - `Command cmd = New ADODB.Command;`
   - `cmd.CommandText = "SELECT * from authors"`
2. 使用占位符‘?’指定参数化命令字符串。

   尽管 SQL 字符串的内容是固定的，您也可以创建“参数化”命令，这样在命令执行时占位符‘?’子字符串将被参数所替代。

   使用 **Prepared** 属性可以优化参数化命令的性能，参数化命令可以重复使用，每次只需要改变参数。

例如，执行以下命令字符串将对所有姓“Ringer”的作者进行查询：

```
Command cmd = New ADODB.Command
cmd.CommandText = "SELECT * from authors WHERE au_lname = ?"
```

3. 指定 **Parameter** 对象并将其追加到 **Parameter** 集合。

每个占位符‘?’将由 **Command** 对象 **Parameter** 集合中相应的 **Parameter** 对象值替代。 可将“Ringer”作为值来创建 **Parameter** 对象，然后将其追加到 **Parameter** 集合：

```
Parameter prm = New ADODB.Parameter
prm.Name = "au_lname"
prm.Type = adVarChar
prm.Direction = adInput
prm.Size = 40
prm.Value = "Ringer"
cmd.Parameters.Append prm
```

4. 使用 **CreateParameter** 方法指定并追加 **Parameter** 对象。

ADO 现在可提供简易灵活的方法在单个步骤中创建 **Parameter** 对象并将其追加到 **Parameter** 集合。

```
cmd.Parameters.Append cmd.CreateParameter _
"au_lname", adVarChar, adInput, 40, "Ringer"
```

本教程将不使用参数化命令，因为需要使用 **Command.Execute** 方法以参数替代占位符‘?’，但该方法不允许指定 **Recordset** 游标类型和锁定选项。为此将使用如下代码：

```
Command cmd = New ADODB.Command;
cmd.CommandText = "SELECT * from authors"
```

下面列出表 **authors** 的模式以供查阅。

列名称	数据类型（长度）	是否可为空
au_id	ID (11)	否
au_lname	varchar(40)	否
au_fname	varchar(20)	否
Phone	char(12)	否
Address	varchar(40)	是
City	varchar(20)	是
State	char(2)	是
Zip	char(5)	是
Contract	bit	否

下一步 步骤 3(See 1.1.6.1.1.3)

1.1.6.1.1.3 步骤 3：执行命令（ADO 教程）

您所在的步骤...

- 连接数据源。
- 可选择创建表示 SQL 查询命令的对象。
- 可选择在 SQL 命令中将值指定为变量参数。

- **执行命令。如果命令以行返回，将行存储在存储对象中。**
  - 可选择对数据进行定位、检查、操作和编辑。
  - 适当情况下，可以使用存储对象中的变更对数据源进行更新。可选择在事务处理中嵌入更新数据。
  - 在使用事务之后，可以接受或拒绝在事务中所做的更改。结束事务。

讨论

返回 **Recordset** 的方法有三种：**Connection.Execute**、**Command.Execute** 以及 **Recordset.Open**。以下是它们的 **Visual Basic** 语法：

```
connection.Execute(CommandText, RecordsAffected, Options)
command.Execute(RecordsAffected, Parameters, Options)
recordset.Open Source, ActiveConnection, CursorType, LockType, Options
```

通过优化这些方法可发挥特定对象的优势。

必须在发出命令之前打开连接，每个发出命令的方法分别代表不同的连接：

- **Connection.Execute** 方法使用由 **Connection** 对象自身表现的连接。
- **Command.Execute** 方法使用在其 **ActiveConnection** 属性中设置的 **Connection** 对象。
- **Recordset.Open** 方法所指定的或者是连接字符串，或者是 **Connection** 对象操作数；否则使用在其 **ActiveConnection** 属性中设置的 **Connection** 对象。

另一个不同点是命令在三种方法中的指定方式：

- 在 **Connection.Execute** 方法中，命令是字符串。
- 在 **Command.Execute** 方法中，命令是不可见的，它在 **Command.CommandText** 属性中指定。另外，此命令可含有参数符号 ('?')，它可以由“参数”VARIANT 数组参数中的相应参数替代。
- 在 **Recordset.Open** 方法中，命令是 *Source* 参数，它可以是字符串或 **Command** 对象。

每种方法可根据性能需要替换使用：

    - **Execute** 方法针对（但不局限）于执行不返回数据的命令。

- 两种 **Execute** 方法都可返回快速只读、仅向前 **Recordset** 对象。

    - **Command.Execute** 方法允许使用可高效重复利用的参数化命令。

- 另一方面，**Open** 方法允许指定 **CursorType** （用于访问数据的策略及对象）和 **LockType** （指定其他用户的 **isolation** 级别以及游标是否在 **immediate** 或 **batch modes** 中支持更新）。

- 请深入了解这些选项，它们体现了很多 Recordset(See 1.1.4.2.10) 的功能。

本教程使用动态游标对 **Recordset** 的所有变更进行批处理，为此请使用以下方法：

```
Recordset rs = New ADODB.Recordset
rs.Open cmd, conn, adOpenDymanic, adLockBatchOptimistic
```

下一步 步骤 4(See 1.1.6.1.1.4)

1.1.6.1.1.4 步骤 4：操作数据（ADO 教程）

您所在的步骤...

- 连接数据源。

- 可选择创建表示 SQL 查询命令的对象。

- 可选择在 SQL 命令中将值指定为变量参数。

- 执行命令。如果命令以行返回，将行存储在存储对象中。

- **可选择对数据进行定位、检查、操作和编辑。**

    - 适当情况下，可以使用存储对象中的变更对数据源进行更新。可选择在事务处理中嵌入更新数据。

    - 在使用事务之后，可以接受或拒绝在事务中所做的更改。结束事务。

讨论

大量 **Recordset** 对象方法和属性可用于对 **Recordset** 数据行进行检查、定位以及操作。

Recordset 可看作行数组，在任意给定时间可进行测试和操作的行为“当前行”，在 **Recordset** 中的位置为“当前行位置”。每次移动到另一行时，该行将成为新的当前行。

有多种方法可在 **Recordset** 中显式移动或“定位”（**Move** 方法）。一些方法（**Find** 方法）在其操作的附加效果中也能够做到。此外，设置某个属性（**Bookmark** 属性）同样可以更改行的位置。

Filter 属性用于控制可访问的行（即这些行是“可见的”）。**Sort** 属性用于控制所定位的 **Recordset** 行中的顺序。

Recordset 有一个 **Fields** 集合，它是在行中代表每个字段或列的 **Field** 集，可从 **Field** 对象的 **Value** 属性中为字段赋值或检索数据。作为选项，可访问大量字段数据（**GetRows** 和 **Update** 方法）。

在本教程中，您将要：

- 假定将区号为“**415**”局号并以“**5**”开头的电话号码更改为虚构的区号“**777**”。

- 在 **au_lname Field** 对象的 **Properties** 集合中设置 **Optimize** 属性以提高存储和筛选性能。
- 按作者的姓对 **Recordset** 使用 **Sort**（排序）操作。
- **Filter**（筛选）**Recordset**，使作者电话区号为“**415**”、局号为“**5**”的行成为仅可访问（即可见的）行。

使用 **Move** 方法从头至尾对经过排序和筛选的 **Recordset** 定位。当 **Recordset EOF** 属性表明已经到达最后一行时停止。在 **Recordset** 中移动时，显示作者的姓和名以及原始电话号码，然后将 **phone** 字段中的区号改为“**777**”。（**phone** 字段中的电话号码格式为“**aaa xxx-yyyy**”，其中 **aaa** 为区号，**xxx** 为局号。）

```
rs("au_lname").Properties("Optimize") = TRUE
rs.Sort = "au_lname ASC"
rs.Filter = "phone LIKE '415 5*'"
rs.MoveFirst
Do While Not rs.EOF
   Debug.Print "Name: " & rs("au_fname") & " " rs("au_lname") & _
      "Phone: " rs("phone") & vbCr
   rs("phone") = "777" & Mid(rs("phone"), 5, 11)
   rs.MoveNext
Loop
```

下一页 步骤 5(See 1.1.6.1.1.5)

1.1.6.1.1.5 步骤 5：更新数据（ADO 教程）

您所在的步骤...

- 连接数据源。
- 可选择创建表示 SQL 查询命令的对象。
- 可选择在 SQL 命令中将值指定为变量参数。
- 执行命令。如果命令以行返回，将行存储在存储对象中。
- 可选择对数据进行定位、检查、操作和编辑。

- **适当情况下，可以使用存储对象中的变更对数据源进行更新。可选择在事务处理中嵌入更新数据。**
  - 在使用事务之后，可以接受或拒绝在事务中所做的更改。结束事务。

讨论

您刚刚对 **Recordset** 若干行中的数据进行了更改。对于添加、删除和修改数据行，ADO 有两个基本概念。

第一个概念是不立即更改 **Recordset** 而是将更改写入内部“复制缓冲区”。如果您不想进行更改，复制缓冲区中的更改将被放弃；如果想保留更改，复制缓冲区中的改动将应用到 **Recordset**。

第二个概念是只要您声明行的工作已经完成则将更改立刻传播到数据源（即“立即”模式）。或者只是收集对行集合的所有更改，直到您声明该行集合的工作已经完成（即“批”模式）。这些模式将由 **CursorLocation** 和 **LockType** 属性控制。

在“立即”模式中，每次调用 **Update** 方法都会将更改传播到数据源。而在“批”模式中，每次调用 **Update** 或移动当前行位置时，更改都被保存到 **Recordset** 中，只有 **UpdateBatch** 方法才可将更改传送给数据源。使用批模式打开 **Recordset**，因此更新也使用批模式。

注意 Update 可采用简捷的形式将更改用于单个字段或将一组更改用于一组字段，然后再进行更改，这样可以一步完成更新操作。

可选择在“事务”中进行更新。实际上，您可以使用事务来确保多个相互关联的操作或者全部成功执行，或者全部取消。在此情况下，事务不是必需的。

事务可在一段相当长的时间内分配和保持数据源上的有限资源，因此建议事务的存在时间越短越好。（这便是本教程不在进行连接之初就开始事务的原因。）

为使用教程，将您的批更新括在事务中：

```
conn.BeginTrans
rs.UpdateBatch
...
```

下一步 步骤 6(See 1.1.6.1.1.6)

1.1.6.1.1.6 步骤 6：结束更新（ADO 教程）

您所在的步骤...

- 连接数据源。
- 可选择创建表示 SQL 查询命令的对象。
- 可选择在 SQL 命令中将值指定为变量参数。
- 执行命令。如果命令以行返回，将行存储在存储对象中。
- 可选择对数据进行定位、检查、操作和编辑。
- 适当情况下，可以使用存储对象中的变更对数据源进行更新。可选择在事务处理中嵌入更新数据。
- **在使用事务之后，可以接受或拒绝在事务中所做的更改。结束事务。**

讨论

假设批更新结束时发生错误，如何解决将取决于错误的性质和严重性以及应用程序的逻辑关系。如果数据库是与其他用户共享的，典型的错误则是他人在您之前更改了数据字段，这种类型的错误称为“冲突”。ADO 将检测到这种请况并报告错误。

本教程中的该步骤分为两部分：如果不存在更新错误则“提交”事务，结束更新。

如果错误存在，它们会被错误处理例程捕获。可使用 **adFilterConflictingRecords** 常数对 **Recordset** 进行筛选，将冲突行显示出来。要纠正错误只需打印作者的姓和名（**au_fname** 和 **au_lname**），然后回卷事务，放弃成功的更新。由此结束更新。

```
...
conn.CommitTrans
...
On Error
```

```
rs.Filter = adFilterConflictingRecords
rs.MoveFirst
Do While Not rs.EOF
   Debug.Print "Conflict: Name: " & rs("au_fname") " " & rs("au_lname")
   rs.MoveNext
Loop
conn.Rollback
Resume Next
...
```

本教程到此结束。

1.1.6.1.1.7 ADO 教程 (VB)

```
Public Sub main()

Dim conn As New ADODB.Connection
Dim cmd As New ADODB.Command
Dim rs As New ADODB.Recordset
 ‘步骤 1
conn.Open "DSN=pubs;uid=sa;pwd=;database=pubs"
 ‘步骤 2
Set cmd.ActiveConnection = conn
cmd.CommandText = "SELECT * from authors"
 ‘步骤 3
rs.CursorLocation = adUseClient
rs.Open cmd, , adOpenStatic, adLockBatchOptimistic
 ‘步骤 4
rs("au_lname").Properties("Optimize") = True
rs.Sort = "au_lname"
rs.Filter = "phone LIKE '415 5*'"
rs.MoveFirst
Do While Not rs.EOF
    Debug.Print "Name: " & rs("au_fname") & " "; rs("au_lname") & _
        "Phone: "; rs("phone") & vbCr
    rs("phone") = "777" & Mid(rs("phone"), 5, 11)
    rs.MoveNext
Loop

 ‘步骤 5
conn.BeginTrans

 ‘步骤 6 - A
On Error GoTo ConflictHandler
rs.UpdateBatch
On Error GoTo 0
```

```
conn.CommitTrans

Exit Sub

‘ 步骤 6 - B
ConflictHandler:

rs.Filter = adFilterConflictingRecords
rs.MoveFirst
Do While Not rs.EOF
    Debug.Print "Conflict: Name: " & rs("au_fname"); " " & rs("au_lname")
    rs.MoveNext
Loop
conn.Rollback
Resume Next

End Sub
```

1.1.6.1.1.8 ADO 教程 (VC++)

本教程以新版 Microsoft Visual C++ Extensions 为特征。VC++ Extensions 删除了对繁琐的 VARIANT 数据类型的使用。

本教程还使用了 **#import** 伪指令，它将 ADO Typelib 转换到头文件中，这个头文件使一些 ADO 的功能在使用和外观上类似 Microsoft Visual Basic 中相应的用法。

> **注意** 在该范例中发布和使用的绑定宏已经过更改，不再与原始文档相同。有关当前宏定义的详细信息，请参阅头文件 icrsint.h。

```
#define INITGUID
#import "c:\Program Files\Common Files\System\ADO\msado15.dll" \
  no_namespace rename("EOF", "EndOfFile")
#include <stdio.h>
#include <icrsint.h>

void dump_com_error(_com_error &e)
  {
  printf("Error\n");
  printf("\a\tCode = %08lx\n", e.Error());
  printf("\a\tCode meaning = %s", e.ErrorMessage());
  _bstr_t bstrSource(e.Source());
  _bstr_t bstrDescription(e.Description());
  printf("\a\tSource = %s\n", (LPCSTR) bstrSource);
  printf("\a\tDescription = %s\n", (LPCSTR) bstrDescription);
  }
```

```
class CCustomRs :
   public CADORecordBinding
{
BEGIN_ADO_BINDING(CCustomRs)
   ADO_VARIABLE_LENGTH_ENTRY2(1, adVarChar, m_szau_lname,
        sizeof(m_szau_lname), lau_lnameStatus, false)
   ADO_VARIABLE_LENGTH_ENTRY2(2, adVarChar, m_szau_fname,
        sizeof(m_szau_fname), lau_fnameStatus, false)
   ADO_VARIABLE_LENGTH_ENTRY2(3, adVarChar, m_szphone,
        sizeof(m_szphone),    lphoneStatus,    true)
END_ADO_BINDING()

public:
   CHAR   m_szau_lname[41];
   ULONG   lau_lnameStatus;
   CHAR   m_szau_fname[41];
   ULONG   lau_fnameStatus;
   CHAR   m_szphone[12];
   ULONG   lphoneStatus;
};

VOID   main()
   {
   HRESULT          hr;
   IADORecordBinding   *picRs = NULL;

   ::CoInitialize(NULL);

   try
      {
      _ConnectionPtr pConn("ADODB.Connection.1.5");
      _RecordsetPtr pRs("ADODB.Recordset.1.5");
      CCustomRs rs;

// 步骤 1 — 打开连接

      pConn->Open("dsn=pubs;", "sa", "", adConnectUnspecified);

// 步骤 2 — 创建命令
// 步骤 3 — 执行命令

      pRs->Open("select * from authors",
         _variant_t(pConn),
         adOpenDynamic, adLockOptimistic, adCmdText);
```

```
     if (FAILED(hr = pRs->QueryInterface(__uuidof(IADORecordBinding),
          (LPVOID*)&picRs)))
        _com_issue_error(hr);

     if (FAILED(hr = picRs->BindToRecordset(&rs)))
        _com_issue_error(hr);

// 步骤 4 — 操作数据

     pRs->Fields->GetItem("au_lname")->Properties->GetItem("Optimize")->Value
= true;
     pRs->Sort = "au_lname ASC";
     pRs->Filter = "phone LIKE '415 5*'";

     pRs->MoveFirst();
     while (VARIANT_FALSE == pRs->EndOfFile)
        {
        printf("\a\tName: %s\t %s\tPhone: %s\n",
           (rs.lau_fnameStatus == adFldOK ? rs.m_szau_fname : ""),
           (rs.lau_lnameStatus == adFldOK ? rs.m_szau_lname : ""),
           (rs.lphoneStatus == adFldOK ? rs.m_szphone   : ""));

        if (rs.lphoneStatus == adFldOK)
          memcpy(rs.m_szphone, "777", 3);

        if (FAILED(hr = picRs->Update(&rs)))
          _com_issue_error(hr);

     // Change the current row of the Recordset.
     // Recordset data for the new current row will automatically be
     // extracted and placed in the CCustomRs C++ instance variables.

        pRs->MoveNext();
        }
     pRs->Filter = (long) adFilterNone;

// 步骤 5 — 更新数据

     pConn->BeginTrans();
     try
        {
        pRs->UpdateBatch(adAffectAll);

        // 步骤 6-A — 结束更新
        pConn->CommitTrans();
```

```
        }

      catch (_com_error)
        {
        // 步骤 6-B  — 结束更新
        pRs->Filter = (long) adFilterConflictingRecords;
        pRs->MoveFirst();
        while (VARIANT_FALSE == pRs->EndOfFile)
           {
           printf("\a\tConflict: Name = %s\t %s\n",
              (rs.lau_fnameStatus == adFldOK ? rs.m_szau_fname : ""),
              (rs.lau_lnameStatus == adFldOK ? rs.m_szau_lname : ""));
           pRs->MoveNext();
           }
        pConn->RollbackTrans();
        }
      }

   catch (_com_error &e)
      {
      dump_com_error(e);
      }

   CoUninitialize();
   }
```

VC++ 教程到此结束。

1.1.6.1.1.9 ADO 教程 (VJ++)

```
ASCimport com.ms.wfc.data.*;

/**
* ADOTutorial:
* Purpose: Demonstrates the usage of Ado in Java.
*          opens a recordset through a command object
*          and illustrates update within a transaction
*/
public class ADOTutorial
{
   public        static        String        strConn        =        "Driver={SQL
Server};SERVER=JDO_ODIN;DATABASE=JetQA;UID=testmod;PWD=testmod;";//odbc    type
conn string

   public static void main(String args[] )
   {
```

```
     try
     {
        Connection conn = new Connection();
        Command cmd = new Command();
        Recordset rs = new Recordset();

        int actErrorNum = 0;
        Field fld;
        AdoProperties fldProps;

// 步骤 1—打开连接
        conn.open(strConn);

// 步骤 2—创建命令
        cmd.setActiveConnection(conn);
        cmd.setCommandText("SELECT * from authors");

// 步骤 2—将具有源的记录集作为命令对象打开
        rs.setCursorLocation(AdoEnums.CursorLocation.CLIENT);
        rs.setCursorType(AdoEnums.CursorType.DYNAMIC);
        rs.setLockType(AdoEnums.LockType.BATCHOPTIMISTIC);
        rs.open (cmd);

// 步骤 4—操作数据
        fldProps  = rs.getField("au_lname").getProperties();

                        fldProps.getItem("Optimize").setBoolean(true);

        rs.setSort("au_lname");

        rs.setFilter("phone like '415*'");

        rs.moveFirst();

        while ( !rs.getEOF())

        {
           StringBuffer strBuf = new StringBuffer();
           System.out.println( " Name: " + rs.getField("au_fname").getString() +
              "  " + rs.getField("au_lname").getString() +
              "   Phone : " + rs.getField("phone").getString() );

           //将字段的区域代码 415 更改为 779
           fld =  rs.getField("phone");
```

```
          strBuf.append( fld.getString());

          strBuf.setCharAt(0, '7');
          strBuf.setCharAt(1, '7');
          strBuf.setCharAt(2,  '9');


          //将字段设置为新值
          fld.setString(strBuf.toString());

          rs.moveNext();

        }

// 步骤 5—更新设置的字段值
        conn.beginTrans();

                 //STEP6 Part A: Conclude the Update
        try
        {

          rs.updateBatch();
          conn.commitTrans();

        }
// 步骤 6—结束更新


        catch(com.ms.wfc.data.AdoException ex)
        {

          //出现错误，必须回卷事务
          rs.setFilter(new                                          Integer
(AdoEnums.FilterGroup.CONFLICTINGRECORDS));
          rs.moveFirst();

          while(!rs.getEOF())
          {
            //打印冲突记录
            System.out.println("      Conflict      :      Name      :      "+
rs.getField("au_fname").getString() + " " +
               rs.getField("au_lname").getString() );

            rs.moveNext();
          }
```

```
            conn.rollbackTrans();
         }
             System.out.println("type any character to continue...");
         System.in.read();

      }
      catch(Exception ex)
      {
         ex.printStackTrace();
      }
   }
}
```

1.1.6.1.2 RDS 教程

本教程讲述如何使用 RDS 编程模型对数据源进行查询和更新。首先说明的是完成此项任务的必要步骤，然后使用以 ADO for Windows Foundation Classes (ADO/WFC) 为特征的 Microsoft® Visual Basic® Scripting Edition 以及 Microsoft® Visual J++® 重复教程。

本教程使用了不同的语言代码，这基于以下两个原因：

- 假设 RDS 文档的读者使用 Visual Basic 编码。这样使得文档对 Visual Basic 编程人员十分方便，但对于使用其他语言的编程人员则没有多少用处。
- 如果您对特定的 RDS 功能不十分熟悉，但对于其他语言有所了解，那么可以通过在其他语言中寻求相同的功能来达到目的。

关于教程的说明方法

本教程基于 RDS 编程模型，对该模型的每一个步骤都进行了分别的讨论，并在每个步骤的讲述中使用了 Visual Basic 代码段。使用其他语言重复代码范例的讨论较少。给定编程语言教程中的每个步骤，均以编程模型和描述性教程中的相应步骤加以注明。请使用步骤标号来参考在描述性教程中的有关讨论。

RDS 编程模型说明如下，在使用教程时可将其作为参考路线图。

RDS 对象编程模型

- 指定在服务器上调用的程序，并得到从客户端引用该程序的途径（代理）。
- 调用服务器程序，将参数传递给标识数据源和所发命令的服务器程序。
- 一般是通过使用 ADO，服务器程序从数据源获得 **coRecordset** 对象。可选择在服务器上处理 **Rerdset** 对象。
- 服务器程序将最终的 **Recordset** 对象返回客户端应用程序。
- 在客户端，可选择将 **Recordset** 对象置为易于可视控件使用的形式。

- 将 **Recordset** 对象的更改返回服务器并用于更新数据源。

下一页 步骤 1(See 1.1.6.1.2.1)

1.1.6.1.2.1 步骤 1: 指定服务器程序（RDS 教程）

您所在的步骤...

- **指定在服务器上被调用的程序，并获得代理。**
  - 调用服务器程序，将参数传递给标识数据源和所发命令的服务器程序。
  - 典型情况下，通过使用 ADO，服务器程序从数据源获得 **Recordset** 对象。
  - 服务器程序将最终的 **Recordset** 对象返回客户端应用程序。
  - 在客户端，可选择将 **Recordset** 对象置为易于可视控件使用的形式。
  - 将对 **Recordset** 对象的更改返回服务器并用来更新数据源。

讨论

大多数情况下，使用 **RDS.DataSpace** 对象 CreateObject(See 1.1.4.4.15) 方法来指定默认服务器程序、**RDSServer.DataFactory** 或自定义服务器程序（业务对象）。服务器程序在服务器上是实例化的，而返回的是对服务器程序的引用即代理。

本教程使用默认服务器程序：

```
Sub RDSTutorial1()
Dim DS as New RDS.DataSpace
Dim DF as Object
Set DF = DS.CreateObject("RDSServer.DataFactory", "http://yourServer")
...
```

下一页 步骤 2(See 1.1.6.1.2.2)

1.1.6.1.2.2 步骤 2: 调用服务器程序（RDS 教程）

您所在的步骤...

- 指定在服务器上调用的程序，并获得代理。
- **调用服务器程序，将参数传递给表明数据源和所发命令的服务器程序。**
  - 典型情况下通过使用 ADO，服务器程序从数据源获得 **Recordset** 对象。
  - 服务器程序将最终的 **Recordset** 对象返回客户端应用程序。
  - 在客户端，可选择将 **Recordset** 对象设置为可视控件易于使用的形式。
  - 将对 **Recordset** 对象的更改返回服务器，并用来更新数据源。

讨论

当调用客户端代理程序的方法时，服务器上的实际程序将执行该方法。在该步骤中，将在服务器上执行查询。

A 部分 在本教程中，如果不使用 **RDSServer.DataFactory**，那么执行该步骤最简便方法是使用 **RDS.DataControl** 对象。**RDS.DataControl** 将该步骤和上一步骤中的创建代理合并，用于发出查询。

设置 **RDS.DataControl** 对象 Server(See 1.1.4.6.55) 属性以标识服务器程序被实例化的位置；设置 Connect(See 1.1.4.6.13) 属性以指定访问数据源的连接字符串；设置 SQL(See 1.1.4.6.62) 属性以指定查询命令文本。然后发出 Refresh(See 1.1.4.4.37) 方法使服务器程序与数据源相连接，检索查询指定的行，并将 **Recordset** 对象返回客户端。

该教程不使用 **RDS.DataControl**，仅在这里给出它的形式：

```
Sub RDSTutorial2A()
Dim DC as New RDS.DataControl
DC.Server = "http://yourServer"
DC.Connect = "DSN=pubs"
DC.SQL = "SELECT * FROM authors"
DC.Refresh
...
```

本教程同样不使用 ADO 对象调 RDS，我们仍在这里给出它的形式：

```
Dim rs as New ADODB.Recordset
rs.Open "SELECT * FROM authors", "Provider=MS Remote;Data Source=pubs;Remote Server=http://YourServer"
```

B 部分 执行该步骤的一般方法是调用 **RDSServer.DataFactory** 对象的 **Query** 方法。该方法使用用于连接数据源的连接字符串，以及用于指定从数据源返回行的命令文本。

该教程使用 **RDSServer.DataFactory Query** 方法：

```
Sub RDSTutorial2B()
Dim DS as New RDS.DataSpace
Dim DF
Dim RS as ADODB.Recordset
Set DF = DS.CreateObject("RDSServer.DataFactory", "http://yourServer")
Set RS = DF.Query ("DSN=pubs", "SELECT * FROM authors")
...
```

下一页 步骤 3(See 1.1.6.1.2.3)

1.1.6.1.2.3 步骤 3: 服务器获得 Recordset（RDS 教程）

您所在的步骤...

- 指定在服务器上调用的程序，并获得代理。
- 调用服务器程序，将参数传递给标识数据源和所发命令的服务器程序。

- **典型情况下通过使用 ADO，服务器程序从数据源获得 Recordset 对象。**
  - 服务器程序将最终的 **Recordset** 对象返回客户端应用程序。
  - 在客户端，可选择将 **Recordset** 对象设置为可视控件易于使用的形式。
  - 将对 **Recordset** 对象的更改返回服务器，并用来更新数据源。

讨论

服务器程序使用连接字符串和命令文本在数据源查询所需的行。尽管也可以使用其他 Microsoft 数据访问接口如 OLE DB 等，但一般使用 ADO 检索该 **Recordset**。有关执行查询操作的详细信息，请参阅 ADO 教程(See 1.1.6.1.1)。

自定义服务器程序可参照如下：

```
Public Function ServerProgram(conn as String, qry as String) as Object
Dim rs as New ADODB.Recordset
rs.CursorLocation = adUseClient
rs.Open cn, qry, adOpenUnspecified, adLockUnspecified, _
  adCmdUnspecified
Set ServerProgram = rs
End Function
```

下一页 步骤 4(See 1.1.6.1.2.4)

1.1.6.1.2.4 步骤 4: 服务器返回 Recordset（RDS 教程）

您所在的步骤...

- 指定在服务器上调用的程序，并获得代理。
- 调用服务器程序，将参数传递给表明数据源和所发命令的服务器程序。
- 典型情况下通过使用 ADO，服务器程序从数据源获得 **Recordset** 对象。

- **服务器程序将最终的 Recordset 对象返回客户端应用程序。**
  - 在客户端，可选择将 **Recordset** 对象设置为可视控件易于使用的形式。
  - 将对 **Recordset** 对象的更改返回服务器并用于数据源的更新。

讨论

RDS 将被检索的 **Recordset** 对象转换为可返回客户端的形式（即整理 **Recordset**）。实际的转换形式以及发送方法取决于服务器是否位于 **Internet**、**Intranet** 或局域网上，或者服务器是动态链接库。不过该细节并不是关键。总之，RDS 将 **Recordset** 返回到客户端。

在客户端，**Recordset** 对象被返回并赋给本地变量。

```
Sub RDSTutorial4()
Dim DS as New RDS.DataSpace
Dim RS as New ADODB.Recordset   'Optionally, ADOR.Recordset
Dim DF as Object
Set DF = DS.CreateObject("RDSServer.DataFactory", "http://yourServer")
Set RS = DF.Query("DSN=pubs", "SELECT * FROM authors")
...
```

下一页 步骤 5(See 1.1.6.1.2.5)

1.1.6.1.2.5 步骤 5: 使用 DataControl（RDS 教程）

您所在的步骤...

- 指定在服务器上调用的程序，并获得代理。
- 调用服务器程序，将参数传递给表明数据源和所发命令的服务器程序。
- 较为典型的是通过使用 ADO，服务器程序从数据源获得 **Recordset** 对象。
- 服务器程序将最终的 **Recordset** 对象返回客户端应用程序。

- **在客户端，可选择将 Recordset 对象设置为可视控件易于使用的形式。**
  - 将对 **Recordset** 对象的更改返回服务器，并用来更新数据源。

讨论

返回的 **Recordset** 对象已经可以使用。可以对它进行同其他记录集一样的检查、定位或编辑。对记录集进行的操作取决于相应的环境。Microsoft Visual Basic 和 Visual C++ 都具有可直接地、或通过启用数据控件间接地使用 **Recordset** 的可视控件。

例如，如果正在使用 Microsoft® Internet Explorer 显示 Web 页，可能希望在可视控件中显示 **Recordset** 对象。Web 页上的可视控件无法直接访问 **Recordset** 对象。但是，可以通过 **RDS.DataControl** 访问 **Recordset** 对象（即绑定(See 1.1.6.3.1.3.6)）。当 **RDS.DataControl** 的 SourceRecordset(See 1.1.4.6.53) 属性设置为 **Recordset** 对象时，**RDS.DataControl** 便可被可视控件使用。

使用可视控件对象须将其 **DATASRC** 参数设置为 **RDS.DataControl**，并将 **DATAFLD** 属性设置为 **Recordset** 对象字段（列）。

在本教程中，设置 **SourceRecordset** 属性。

```
Sub RDSTutorial5()
Dim DS as New RDS.DataSpace
Dim RS as New ADODB.Recordset   '可选择 ADOR.Recordset
Dim DC as New RDS.DataControl
Dim DF as Object
Set DF = DS.CreateObject("RDSServer.DataFactory", "http://yourServer")
Set RS = DF.Query ("DSN=pubs", "SELECT * FROM authors")
DC.SourceRecordset = RS         '可视控件现在可绑定到 DC。
...
```

下一页 步骤 6(See 1.1.6.1.2.6)

1.1.6.1.2.6 步骤 6: 将更改返回服务器（RDS 教程）

您所在的步骤...

- 指定在服务器上调用的程序，并获得代理。
- 调用服务器程序，将参数传递给表明数据源和所发命令的服务器程序。
- 较为典型的是通过使用 ADO，服务器程序从数据源获得 **Recordset** 对象。
- 服务器程序将最终的 **Recordset** 对象返回客户端应用程序。
- 在客户端，可选择将 **Recordset** 对象置为易于可视控件使用的形式。

- **将对 Recordset 对象的更改返回服务器并用来更新数据源。**

讨论

如果对 **Recordset** 对象进行编辑，任何更改（即对行的增加、修改或删除）都可以返回服务器。

> **注意** TRDS 的默认行为可通过 ADO 对象和 Microsoft OLE DB Remoting Provider 隐式调用。查询可返回记录集，而被编辑的记录集则可更新数据源。本教程不通过 ADO 对象调用 RDS，但在这里给出它的形式。

```
Dim rs as New ADODB.Recordset
rs.Open "SELECT * FROM authors", "Provider=MS Remote;Data Source=pubs;Remote Server=http://YourServer"
...          '编辑记录集
rs.UpdateBatch    '等值于 SubmitChanges
...
```

A 部分 假设在这里只使用了 **RDS.DataControl** 并且 **Recordset** 对象现在已与 **RDS.DataControl** 关联。如果 Server(See 1.1.4.6.55) and Connect(See 1.1.4.6.13) 属性已设置，**SubmitChanges** 方法将把对 **Recordset** 对象的任何改动更新到数据源。

```
Sub RDSTutorial6A()
Dim DC as New RDS.DataControl
Dim RS as New ADODB.Recordset   '可选择 ADOR.Recordset
DC.Server = "http://yourServer"
DC.Connect = "DSN=pubs"
DC.SQL = "SELECT * FROM authors"
DC.Refresh
...
Set RS = DC.Recordset
...                        '编辑 Recordset
...
DC.SubmitChanges
...
```

B 部分 另外，您也可以通过指定连接和 **Recordset** 对象，使用 **RDSServer.DataFactory** 对象更新服务器。

```
Sub RDSTutorial6B()
Dim DS as New RDS.DataSpace
Dim RS as New ADODB.Recordset   '可选择 ADOR.Recordset
Dim DC as New RDS.DataControl
Dim DF as Object
Set DF = DS.CreateObject("RDSServer.DataFactory", "http://yourServer")
Set RS = DF.Query ("DSN=pubs", "SELECT * FROM authors")
DC.SourceRecordset = RS        '可视控件现在可绑定到 DC.
...                        '编辑 Recordset
blnStatus = DF.SubmitChanges "DSN=pubs", RS
```

本教程到此结束。

1.1.6.1.2.7 RDS 教程 (VBScript)

该部分是使用 Microsoft Visual Basic，Scripting Edition 编写对“RDS 教程”的重新说明。

本教程中，**RDS.DataControl** 和 **RDS.DataSpace** 是在设计时创建的，就是说它们通过对象标记进行定义，如 <OBJECT>...</OBJECT>。此外，它们也可在运行时通过 **Server.CreateObject** 方法创建。例如，**RDS.DataControl** 对象的创建可

以是：

Set DC = **Server.CreateObject**("RDS.DataControl")

```
<!-- RDS.DataControl -->
<OBJECT
ID="DC1" CLASSID="CLSID:BD96C556-65A3-11D0-983A-00C04FC29E33">
</OBJECT>

<!-- RDS.DataSpace -->
<OBJECT
ID="DS1" WIDTH=1 HEIGHT=1
CLASSID="CLSID:BD96C556-65A3-11D0-983A-00C04FC29E36">
</OBJECT>

<SCRIPT LANGUAGE="VBScript">

Sub RDSTutorial()
Dim DF1 as Object
```

步骤 1 — 指定服务器程序

VBScript 可以发现它运行其上的 IIS Web 服务器的名称，方法是访问可用于 Active Server Pages 的 **VBScript Request.ServerVariables** 方法:

```
"http://<%=Request.ServerVariables("SERVER_NAME")%>"
```

不过对于本教程，将使用假设的服务器“yourServer”。

> **注意** 请留意 **ByRef** 参数的数据类型。**VBScript** 不允许指定变量类型，因此必须始终传递变体型。使用 HTTP 时，RDS 允许将变体型传递给希望使用非变体型的方法，以便使用 **RDS.DataSpace** 对象的 **CreateObject** 方法进行调用。当使用 DCOM 或过程中服务器时，必须使客户端与服务器端的数据类型相匹配，否则将会产生“类型不匹配”错误。

```
Set DF1 = DS1.CreateObject("RDSServer.DataFactory", "http://yourServer")
```

步骤 2a — 通过 RDS.DataControl 调用服务器程序

该范例只是注释，说明 **RDS.DataControl** 的默认行为是执行指定的查询。

```
<OBJECT CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID="DC1">
<PARAM NAME="SQL" VALUE="SELECT * FROM authors">
<PARAM NAME="Connect" VALUE="DSN=Pubs;">
<PARAM NAME="Server" VALUE="http://YourServer/">
</OBJECT>
...
<SCRIPT LANGUAGE="VBScript">

Sub RDSTutorial2A()
Dim RS as New ADODB.Recordset
DC1.Refresh
Set RS = DC1.Recordset
```

```
...
```

步骤 2b — 通过 RDSServer.DataFactory 调用服务器程序

步骤 3 — 服务器获得 Recordset

步骤 4 — 服务器返回 Recordset

```
Set RS = DF1.Query("DSN=pubs", "SELECT * FROM authors")
```

步骤 5 — 使 DataControl 能被可视控件使用

```
' 将返回的记录集指定到 DataControl。
DC1.SourceRecordset = RS
```

步骤 6a — 使用 RDS.DataControl 将更改返回服务器

该范例只是注释，说明 是如何执行更新的。

```
<OBJECT CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID="DC1">
<PARAM NAME="SQL" VALUE="SELECT * FROM authors">
<PARAM NAME="Connect" VALUE="DSN=Pubs;">
<PARAM NAME="Server" VALUE="http://YourServer/">
</OBJECT>
...
<SCRIPT LANGUAGE="VBScript">

Sub RDSTutorial6A()
Dim RS as New ADODB.Recordset
DC1.Refresh
...
Set RS = DC1.Recordset
' 编辑记录集对象.
' SERVER 和 CONNECT 属性已经在步骤 2A 中设置。
Set DC1.SourceRecordset = RS
...
DC1.SubmitChanges
```

步骤 6b — 通过 RDSServer.DataFactory 将更改返回服务器

```
DF.SubmitChanges "DSN=pubs", RS

End Sub
</SCRIPT>
</BODY>
</HTML>
```

本教程到此结束。

1.1.6.1.2.8 RDS 教程 (VJ++)

ADO/WFC 不完全遵循 RDS 对象模型的地方是它不执行 DataControl 对象。ADO/WFC 仅执行客户端类 DataSpace。DataSpace 类将执行一个方法即 createObject，该方法返回 ObjectProxy 对象。DataSpace 类还执行 InternetTimeout 属性。

ObjectProxy 类将执行一个方法即 call，该方法可以调用任何服务器端业务对象。

本教程由此开始。

```
import com.ms.wfc.data.*;
public class RDSTutorial
{
public void tutorial()
{
// 步骤 1 — 指定服务器程序
ObjectProxy obj =
DataSpace.createObject(
"RDSServer.DataFactory",
"http://YourServer");

// 步骤 2 — 服务器返回 Recordset
Recordset rs = (Recordset) obj.call(
"Query",
new Object[] {"DSN=pubs", "SELECT * FROM authors"});

// 步骤 3 — 将更改传送至服务器
...                              // 编辑 Recordset
obj.call(
"SubmitChanges",
new Object[] {"DSN=pubs", rs});
return;
}
}
```

本教程到此结束。

1.1.6.1.3 建立简单的远程数据服务应用程序

可以使用 RDS.DataControl(See 1.1.4.2.3) 对象创建可访问活动数据库的简单 Active Server Pages 文件。

该教程采用“最小代码”法，通过开发人员指南(See 1.1.6.3)提供使用 **RDS.DataControl** 的详细信息。在该教程中，您将连接到 Microsoft® Access Database，显示在数据绑定网格中设置的结果，并添加在所显示的 **Recordset** 中定位的功能。如下过程说明创建该简单应用程序的步骤：

1. 标识数据库。(See 1.1.6.1.3.1)
2. 插入网格和 RDS.DataControl 对象。(See 1.1.6.1.3.2)
3. 添加 HTML 控件。(See 1.1.6.1.3.3)
4. 添加代码向数据库发送查询。(See 1.1.6.1.3.4)
5. 添加代码向数据库提交更改。(See 1.1.6.1.3.5)

6. 添加在所显示的记录集中移动的代码。(See 1.1.6.1.3.6)

7. 查看操作中的代码。(See 1.1.6.1.3.7)

如果要查看该示范应用程序的完整版本，请访问 http://<webservername>/MSADC/Samples/ADCTest.asp。其中，<webservername> 是 Web 服务器名。

1.1.6.1.3.1 标识数据库 (RDS)

在创建 .asp 文件之前，必须先标识在控制面板 ODBC applet 中的数据库。在该范例中，将使用提供 Remote Data Service 的 Microsoft® Access 数据库。

1. 在运行 Web 服务器的计算机上，打开“控制面板”。

   双击 ODBC 图标，再单击“系统 DSN”。

   有两类数据源：“用户”，只有您才能访问；“系统”，所有使用计算机的人都可以访问。用于 Web 服务器的数据源必须是“系统”数据源。

2. 单击“添加”，选定“**Microsoft** 访问驱动程序”，再单击“完成”。

3. 在“数据源名称”框中，键入“**AdvWorks**”并单击“选择”。再选择文件“C:\Program Files\Common Files\System\MSADC\Samples\AdvWorks.mdb”，单击“确定”。

4. 单击“确定”关闭对话框。

下一页(See 1.1.6.1.3.2)

1.1.6.1.3.2 插入网格和 RDS.DataControl 对象 (RDS)

1. 使用文本编辑器打开文件 Tutorial.asp，该文件位于 C:\Program Files\Common Files\System\MSADC\Samples\Tutorial 文件夹。

2. 请查找字串“教程：RDS.DataControl 对象”。复制如下脚本并粘贴在该注释后面。

```
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
  ID=ADC HEIGHT=1 WIDTH = 1>
</OBJECT>
```

6. 请查找字串“教程：数据绑定网格控件”。复制如下脚本并粘贴在该注释后面。

```
<BR>
<Center>
<OBJECT ID="GRID" WIDTH=600 HEIGHT=200 Datasrc="#ADC"
CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/ssdatb32.cab"
 CLASSID="CLSID:AC05DC80-7DF1-11d0-839E-00A024A94B3A">
```

```
<PARAM NAME="_Version"      VALUE="131072">
<PARAM NAME="BackColor"    VALUE="-2147483643">
<PARAM NAME="BackColorOdd"  VALUE="-2147483643">
<PARAM NAME="ForeColorEven" VALUE="0">
<PARAM NAME="AllowAddNew" VALUE="TRUE">
<PARAM NAME="AllowDelete" VALUE="TRUE">
<PARAM NAME="AllowUpdate" VALUE="TRUE">
</OBJECT>
<BR>
```

在网格控件的 OBJECT 标记中的参数 **DATASRC** 指示数据源将是 **RDS.DataControl** 对象 ADC。它将网格绑定到返回的 **Recordset**，在 **RDS.DataControl** 获得 **Recordset** 时使网格显示数据。参数 **AllowAddNew**，**AllowDelete**，和 **AllowUpdate** 启用网格记录来自用户的添加、变更和删除。

下一页(See 1.1.6.1.3.3)

1.1.6.1.3.3 添加 HTML 控件 (RDS)

1. 请查找字串“教程：获得记录集的 HTML 控件”。复制如下脚本并粘贴在该注释后面。

```
<BR>
<table>
<tr><td>RDS Server:<td><INPUT NAME=Server SIZE=70>
<tr><td>Connection:<td><INPUT NAME=Connect SIZE=70>
<tr><td>Query:<td><INPUT NAME=SQL SIZE=70>
</table>
```

由此添加一些用户的文本框来指定发送到数据库的 SQL 查询(See 1.1.6.1.3.4)(在数据源名称中指定），以便向客户返回已断开的 **Recordset**.

8. 请查找字串“教程：定位记录集并运行查询的 HTML 控件”。复制如下脚本并粘贴在该注释后面。

```
<BR>
<BR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run!" >
<INPUT TYPE=BUTTON NAME="First" VALUE="First" >
<INPUT TYPE=BUTTON NAME="Next" VALUE="Next" >
<INPUT TYPE=BUTTON NAME="Prev" VALUE="Previous" >
<INPUT TYPE=BUTTON NAME="Last" VALUE="Last" >
<INPUT TYPE=BUTTON NAME="Submit" VALUE="Submit Changes" >
</CENTER>
```

“第一”、“下一个”、“上一个”和“最后”按钮使用户总是能够在显示的记录集中定位(See 1.1.6.1.3.6)。

1.1.6.1.3.4 添加代码向数据库发送查询 (RDS)

请查找字串“教程：向数据库发送查询”。复制如下脚本并粘贴在注释后面。

```
SUB Run_OnClick
```

```
   ADC.Server = Server.Value
   ADC.Connect = Connect.Value
   ADC.SQL = SQL.Value

   ADC.Refresh
END SUB
```

注意　　通过 **Connect** 属性指定数据源时，通常需要提供数据源名称、用户 ID 和密码（例如，“DSN=SalesDB;UID=Manager;PWD=secret;”）。使用 Microsoft Access 数据库时，不必指定用户 ID (UID) 或密码 (PWD)。

下一页(See 1.1.6.1.3.5)

1.1.6.1.3.5 添加代码向数据库提交更改

请查找字串“教程：向数据库提交更改的代码”。复制如下脚本并粘贴在注释后面。

```
SUB Submit_OnClick  ' 已单击“提交更改”按钮。
  ADC.SubmitChanges ' 将更改发送到 DBMS.
  ADC.Refresh       ' 刷新记录集。
End Sub
```

执行 ADC.SubmitChanges 时，Remote Data Service 将所有更新信息打包后通过 HTTP 发送到服务器。只能更新全部或不更新。即如果部分更新不成功，则不进行更改，并返回状态信息。

下一页(See 1.1.6.1.3.6)

1.1.6.1.3.6 添加代码在显示的记录集 (RDS) 中移动

可对 **RDS.DataControl** 对象中的 **Recordset** 使用 **Move** 方法，以此定位显示在 **Web** 页中数据绑定控件的数据记录。例如，假设通过绑定 **RDS.DataControl** 对象来显示网格中的 **Recordset**。那么，就可以将“第一”、“最后”、“下一个”和“上一个”按钮包括进来，使用户可以单击这些按钮移动到所显示的 **Recordset** 中的第一、最后、下一个和上一个记录。实现该功能的方法是在这些按钮的 **OnClick** 过程中，分别地调用 **RDS.DataControl** 对象的 **MoveFirst**, **MoveLast**, **MoveNext**, 和 **MovePrevious** 方法。请查找字串“教程：在显示的记录集中移动”。复制如下脚本并粘贴在注释后面。

```
SUB First_OnClick
   ADC.Recordset.MoveFirst
END SUB

SUB Next_OnClick
   If ADC.Recordset.EOF Then   '移动无法超出末端记录
      ADC.Recordset.MoveFirst
      Exit Sub
   End If

   ADC.Recordset.MoveNext
END SUB

SUB Prev_OnClick
```

```
   If ADC.Recordset.BOF Then   '移动无法超出顶端记录
      ADC.Recorset.MoveFirst   '移出 BOF 缓冲区
      ADC.Recordset.MoveLast
      Exit Sub
   End If
   ADC.Recordset.MovePrevious
END SUB

SUB Last_OnClick
   ADC.Recordset.MoveLast
END SUB
```

下一页(See 1.1.6.1.3.7)

1.1.6.1.3.7 查看操作中的代码 (RDS)

1. 请按文本文件保存文件 Tutorial.asp 的更改，并退出文本编辑器。确定文本编辑器没有替换 .asp 文件扩展名。.asp 文件是 Active Server Pages 脚本文件。

2. 如需验证已创建的 ASP 页有效，请将浏览器指向 http://<webservername>/MSADC/Samples/Tutorial/Tutorial.asp，这里 <webservername> 是 Web 服务器的实际名称。

注意 必须通过指向地址访问 ASP 页。如果使用“文件”菜单中的“打开”命令或双击 .asp 文件，均不能正确把它打开。

加载网页时，请单击“运行!”将数据填入网格。可更改查询或其他参数，再次单击“运行!”可查看其他数据。

1.1.6.1.4 教程：地址簿

该范例应用程序用于说明怎样使用远程数据服务(See 2.30)来建立简单的、数据识别的 Web 应用程序 — 联机的企业地址簿。本教程适用于想学习使用带有远程数据服务的数据识别 ActiveX 控件(See 2.16)的 Microsoft® Visual Basic scripting Edition (VBScript) 和 ActiveX®(See 2.1) 程序员，以及想建立以数据为中心的 Web 应用程序而且更有经验的应用程序开发人员。

本教程假定您已了解如何使用基本的 HTML 布局标记和带 ActiveX 控件的程序。

该地址簿教程将指导您学习以下内容：

- 确定满足运行该应用程序(See 1.1.6.1.4.2)的系统要求(See 1.1.6.1.4.1)。
- 设置连接数据库(See 1.1.6.1.4.4)的环境，并运行 SQL 查询来复制范例数据(See 1.1.6.1.4.3)。
- 在 Web 页上建立 HTML 框架(See 1.1.6.1.4.6)并编写 VBScript 代码处理用户事件。

地址簿简介

地址簿范例应用程序是一个用于 Intranet 发布目录的联机地址簿。地址簿使用户可以在一个或多个字段中输入搜索字符串，以便请求有关雇员的信息。为了说明远程数据服务的基本功能，范例应用程序应尽量短小，并使用最少量的对象和搜索字段。

应用程序接口包含如下部分：

- 文本框(See 1.1.6.1.4.7)作为有关雇员的不同种类信息的搜索字段使用。
- 命令按钮(See 1.1.6.1.4.9)用于建立查询，清除搜索字段，更新雇员信息数据库，或取消挂起的更改。
- 类似电子数据表的网格(See 1.1.6.1.4.10)用于显示搜索结果，并通过不可见的 RDS.DataControl 数据绑定对象(See 1.1.6.1.4.8).绑定到后端数据库。
- 按钮(See 1.1.6.1.4.11)用于定位网格中显示的数据行。

1.1.6.1.4.1 运行地址簿范例应用程序

1. 确定满足系统要求(See 1.1.6.1.4.2)。
2. 设置 ODBC 系统 DSN(See 1.1.6.1.4.4)。
3. 运行 SQL 脚本(See 1.1.6.1.4.3) 将数据加载到 Microsoft SQL Server 数据库。
4. 确定 Microsoft® SQL Server® 正在运行。单击“开始”，指向“程序”，再指向“Microsoft SQL Server 6.5”，然后单击“SQL 服务管理器”。如果停止信号为绿色，则 SQL Server 正在运行。如果不是绿色，请单击“开始/继续”。
5. 在 Microsoft® Internet Explorer 4.0 或更新的版本中，键入如下地址：

   http://*webserver*/msadc/samples/AddressBook/AddrBook.asp

   此处 webserver 是安装远程数据服务服务器组件的 Web 服务器名，这也是创建系统 DSN(See 1.1.6.1.4.4). 的地方。

6. 可在地址簿范例应用程序中尝试各种应用方案，如按他或她的电子邮件地址查找人员，列表显示职位为“程序管理员”的所有人，或编辑和更新现存的记录。单击“查找”将所有可用名称填入数据网格。

1.1.6.1.4.2 地址簿应用程序的系统要求

要安装地址簿范例应用程序，需要满足以下软件和数据库要求。

软件要求

运行该 Web 应用程序的服务器软件必须包括：

- Microsoft® Windows NT® Server x86，版本 4.0 Service Pack 3 或更新版本。
- Microsoft® nternet Information Server (IIS) 3.0 或更新版本，带 Active Server Pages。

- 远程数据服务的服务器端 DLLs(See 2.22)。（远程数据服务安装程序可自动为您实现该项目。）

运行该 Web 应用程序的客户端软件必须包括：

- Microsoft® Internet Explorer 4.0 或更新版本。
- Microsoft® Windows NT® Workstation 或 Microsoft® Windows® 95。
- 安装在客户端计算机上的 远程数据服务客户端 DLL。它们是 Internet Explorer 4.0 安装程序中的一部分。

数据库要求

要使用该范例，必须有：

- 可操作的 Microsoft SQL Server 版本 6.5 数据库。
- 数据库访问权限。
- 将范例数据充填数据库的写入特权。
- 通过 Enterprise Manager 或 ISQL 工具充填数据的验证。

如果没有权限，则需数据库管理员设置系统，并授予您数据库的访问权限，或为您建立数据库。
一旦满足这些数据库要求，请参考如下文档所述内容“运行地址簿 SQL 脚本(See 1.1.6.1.4.3)”和“建立地址簿的 ODBC 连接(See 1.1.6.1.4.4)。”

1.1.6.1.4.3 运行地址簿 SQL 脚本

必须使用 ISQL 命令行工具或 SQL Server Enterprise Manager 来运行如下操作的 SQL 脚本：

- 在默认设备上创建新数据库 (AddrBookDB)。
- 连接正确的数据库（AddrBookDB；如果重新命名数据库，请注意是在 ODBC 连接 DSN 安装程序(See 1.1.6.1.4.4)中更改“数据库名称”）。
- 创建“雇员”表。
- 将范例数据充填表中。
- 运行简单的 SELECT 语句以验证数据库表的数量。
- 设置带有“ADCDemo”密码的名为“ADCDemo”的用户帐号。

必须运行所给的 SQL 脚本文件 (Sampleemp.sql) 以创建和充填“雇员”表。默认情况下，“雇员”表被创建在本地 SQL 服务器上新建立的数据库 AddrBookDB 中。

运行 Sampleemp.sql 脚本

1. 单击“开始”，指向“程序”，再指向“Microsoft SQL Server 6.5”，然后单击“SQL Enterprise Manager”。
2. 从“工具”菜单，单击“SQL 查询工具”。
3. 单击“加载 SQL 脚本”按钮（在工具栏上打开的文件夹），定位到：Program Files\Common Files\System\MSADC\Samples\AddressBook。
4. 选择文件 Sampleemp.sql。单击“打开”。
5. 单击“执行查询”按钮（工具栏上的绿色箭头）。
6. 执行该文件后，请关闭“查询”和“Enterprise Manager”窗口。

1.1.6.1.4.4 建立地址簿的 ODBC 连接

必须在 Web 服务器上安装 ODBC 系统数据源名称 (DSN)(See 2.18) 在该 Web 服务器上将安装远程数据服务服务器组件。该安装程序假定“雇员”表已创建在本地 SQL 服务器上的 AddrBookDB 数据库中，该 SQL 服务器通过运行 SQL 脚本(See 1.1.6.1.4.3).在 Web 服务器上运行。如果要使用运行在不同的服务器上的另一个数据库或另一个 SQL 服务器，则必须更改“ODBC SQL Server 安装程序”对话框中的“服务器”和“数据库名称”。如果这样做了，还必须在所包括的地址簿页中稍作改动。

设置 Windows NT® 客户端的 DSN

1. 在 Web 服务器上，单击“开始”，指向“设置”，再单击“控制面板”。
2. 在“控制面板”中，双击 ODBC 图标。
3. 在“系统 DSN”选项卡上，单击“添加”。
4. 在“创建新数据源”框中，选择“SQL Server”，再单击“完成”。
5. 执行“创建新的 SQL 服务器数据源”向导中的步骤。如需帮助，请单击向导对话框中的“帮助”。
6. 完成后，单击“确定”返回“ODBC 数据源管理器”对话框。单击“关闭”退出 ODBC 安装程序。

设置 Windows 95 客户端的 DSN

7. 在 Web 服务器上，单击“开始”，指向“设置”，再单击“控制面板”。
8. 在“控制面板”中，双击“ODBC”图标。
9. 单击“系统 DSN”。
10. 在“系统数据源”对话框中，单击“添加”，选择“SQL Server”，再单击“确定”。
11. 执行前面的 Windows NT 客户程序过程的步骤 5 和步骤 6。

1.1.6.1.4.5 地址簿范例应用程序代码概述

地址簿应用程序的程序代码依次安排在下面各节中。每一节突出一组应用程序的主要组件，显示组件的源代码，并描述代码的工作方式。

每节同时包括用于响应用户事件（如单击）时执行的 VBScript 代码。地址簿应用程序的 VBScript 部分可完成下列操作：将搜索文本装配到查询语句，向服务器发送请求，判断在何处显示从服务器接收到的结果，并处理所有按钮单击事件。

代码段	说明
HTML 框架(See 1.1.6.1.4.6)	包含含应用程序代码的 HTML 框架。
文本框(See 1.1.6.1.4.7)	作为有关雇员的不同种类信息的搜索字段。
命令按钮(See 1.1.6.1.4.9)	建立查询，清除查询字段，更新雇员信息数据库，或取消挂起的变更。
类似电子表格的网格(See 1.1.6.1.4.10)	通过不可见 RDS.DataControl 数据绑定对象(See 1.1.6.1.4.8)。显示搜索结果并绑定到后端数据库。
RDS.DataControl 数据绑定对象(See 1.1.6.1.4.8)	将来自 Microsoft SQL Server 数据库的数据绑定到客户程序 HTML 页的网格中。
定位按钮(See 1.1.6.1.4.11)	用于定位在网格中显示的数据行。
VBScript 初始化代码(See 1.1.6.1.4.12)	初始化数据网格的列标题和网格标题。

地址簿范例应用程序的构造便于学习如何在增加的步骤中建立简单的远程数据服务，并在每一步添加功能。为了便于说明，该教程略去重复的代码且并不总是根据其在程序中的顺序来对它进行描述。如果要查看地址簿应用程序的完整源代码，请单击此处(See 1.1.6.1.4.13)。

使用文本编辑器（如记事本）复制教程主题中的代码并粘贴到工程中。将工程按“Tutorial2.asp”保存到目录：Program Files\Common Files\System\Msadc\Samples\Tutorial directory 。 如 需 在 Internet Explorer 4.0 中 查 看 ， 请 键 入 ：**http://***webserver*/**msadc/samples/Tutorial/Tutorial2.asp**

此处，webserver 是运行 Internet Information Server 和 ASP 的 Windows NT® Web 服务器名称。

1.1.6.1.4.6 地址簿 HTML 框架

建立远程数据服务应用程序的第一步是建立标准的 HTML Web 页面框架。HTML 框架包含所使用的应用程序对象的程序代码，如按钮、文本框和数据识别控件(See 2.16)。

如下代码段显示地址簿应用程序的 HTML 框架。该部分包括标准的 HTML 标头和页结构标记，标识应用程序的注释，和 Web 页文本、字体以及颜色信息。

注释被添加到应用程序对象（如按钮，文本框，和数据识别的控件）所在位置。在这些位置可以顺便访问自己的应用程序组件。单击这些占位符注释，可获得应用程序的这些部分的说明。

注意 大部分标准 HTML 标记，以及定义 Web 页背景属性的标记将不在这里定义。SCRIPT 标记表示控件被传递到脚本语言 — 此处是 VBScript。

```
<HTML>
<HEAD>
```

```
<TITLE>Corporate Address Book</TITLE>
</HEAD>

<!--
   目的:    为 Web 用户提供公司目录搜索服务。
   编写者:    Microsoft Remote Data Service 工作组，Microsoft Corp.
   日期:      1997 年 4 月
-->

<BODY BACKGROUND="Arcadia.gif" LANGUAGE="VBScript" onload="Load">
<tr>
  <td align="center" width="40%">
   <table border="2" cellpadding="7" cellspacing="7">
   <tr>
     <td width="100%"><font color="#160B5A"><font
            size="4"><strong>Arcadia        Bay        Corporate        Phone
Directory</strong></font></font></td>
   </tr>
   </table>
   </td>
</tr>

<hr>
<h2><font color = "#160B5A">Search Parameters</h2>
<h5><font color = "#160B5A">Please enter one or more search patterns and press
FIND to search.</h5>


<PRE>First Name
<!-- 由此转到 HTML 文本框代码 (See 1.1.6.1.4.7) -->

<!-- 由此转到 HTML 命令按钮代码 (See 1.1.6.1.4.9) -->
<hr>
<h2><font color = "#400040">Search Results</h2>
</B>
<br>
<!-- 由此转到 HTML 数据网格代码 (See 1.1.6.1.4.10) -->
<!-- 由此转到 HTML 定位按钮代码 (See 1.1.6.1.4.11) -->

<!-- 由此转到 RDS.DATACONTROL 对象代码 (See 1.1.6.1.4.8) -->

<SCRIPT Language="VBScript">

<!--  由此转到初始化(See  1.1.6.1.4.12)、清除文本框(See  1.1.6.1.4.9)、提交查询(See
```

```
1.1.6.1.4.9)、通过显示的记录集定位(See 1.1.6.1.4.11)、将更改保存到记录(See 1.1.6.1.4.9)
和 取消索引挂起的更改(See 1.1.6.1.4.9)的 VBSCRIPT 代码    -->
</SCRIPT>

<BR>
<font color = "#400040">This site powered by Microsoft Remote Data Service.
</font>
</BODY>
</HTML>
```

1.1.6.1.4.7 地址簿文本框

地址簿应用程序包含 4 个文本框，用于输入搜索短语（例如，“名字”框）。下列 HTML 代码定义文本框对象：

```
<PRE> First Name   <INPUT NAME=SFirst SIZE=30> </PRE>

<PRE> Last Name  <INPUT NAME=SLast  SIZE=30> </PRE>

<PRE> Title    <INPUT NAME=STitle SIZE=30> </PRE>

<PRE> E-mail Alias <INPUT NAME=SEmail SIZE=30> </PRE>
```

固有的 INPUT 控件被建立到 HTML 中。参数 **NAME** 定义它在代码中的名称。例如，VBScript 代码可检查在“名字”框中文本的 SFirst.Value。参数 SIZE 按字符来调整文本框长度。

1.1.6.1.4.8 地址簿数据绑定对象

地址簿应用程序使用 RDS.DataControl(See 1.1.4.2.3) 对象将 SQL 服务器数据库的数据绑定到应用程序的客户 HTML 页中的可视化对象（此处为网格显示(See 1.1.6.1.4.10)）。事件驱动的 VBScript 程序逻辑使用 RDS.DataControl 可以：

- 查询数据库，将更新发送到数据库，并刷新数据网格(See 1.1.6.1.4.9)。
- 让用户在数据网格中移动到第一、下一个、上一个或最后记录(See 1.1.6.1.4.11)。

如下代码定义 RDS.DataControl 组件：

```
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=SControl Width=1 Height=1>
   <PARAM                                                    NAME="SERVER"
VALUE="http://<%=Request.ServerVariables("SERVER_NAME")%>">
   <PARAM NAME="CONNECT" VALUE="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
</OBJECT>
```

程序中的标记 OBJECT 定义 RDS.DataControl 组件。该标记包括两种参数：

- 与一般的 OBJCT 标记关联的参数。
- RDS.DataControl 对象的特定参数。

一般的 OBJECT 标记参数

下表描述与 OBJECT 标记关联的参数。

参数	说明
CLASSID	唯一的 128 位数，用于标识系统嵌入对象的类型。该标识符保留在本地计算机的系统注册表中。（关于 **RDS.DataControl** 对象的类 ID，请参见 RDS.DataControl 对象(See 1.1.4.2.3)。）
ID	定义嵌入对象的文档宽度标识符，该标识符在代码中用于标识此嵌入对象。

RDS.DataControl 标记参数

下表描述 **RDS.DataControl** 对象的特定参数。（如果要查看 **RDS.DataControl** 对象参数的完整列表，或者要执行这些参数，请参见 RDS.DataControl 对象(See 1.1.4.2.3).)

参数	说明
SERVER(See 1.1.4.6.55)	如果使用 HTTP，该值以 HTTP:// 开头后跟服务器名称。
CONNECT(See 1.1.4.6.13)	VALUE 的第一部分表示指向数据源的系统 DSN(See 2.18) 第二和第三部分指定用于访问数据源的用户 ID 和密码（例如，“DSN=ADCDEMO;UID=adcdemo;PWD=adcdemo;”）。如需帮助，可向系统数据库管理员查询。
SQL(See 1.1.4.6.62)	设置或返回用于检索 **Recordset** 的查询字符串。

1.1.6.1.4.9 地址簿命令按钮

地址簿应用程序包括下列命令按钮：

- “查找”按钮，用于向数据库提交查询。
- “清除”按钮，用于开始进行新搜索前清除文本框
- “更新配置文件”按钮，用于保存对雇员记录的更改。
- “取消更改”按钮，用于放弃更改。

“查找”按钮

下列 HTML 语句定义“查找”按钮。该 HTML 语句出现在程序的 VBScript 节之前。请将该控件复制并粘贴在 HTML 命令按钮的注释后面。

```
<INPUT TYPE=BUTTON NAME="Find"    VALUE="Find">
```

Find_OnClick 子过程

单击“查找”按钮可激活 VBScript Find_OnClick 子过程，该过程建立并发送 SQL 查询。在工程完成之后，请单击该按钮填充数据网格。

建立 SQL 查询

Find_OnClick 子过程的第一部分建立 SQL 查询（一次一个短语），方法是向全局 SQL SELECT 语句追加文本字符串。该字符串以设置变量 myQuery 为 SQL SELECT 语句开始，SQL SELECT 从数据源表中请求所有数据行。请将此代码复制并粘贴到打开 SCRIPT 标记的后面。

```
Sub Find_OnClick

myQuery = "Select FirstName, LastName, Title, Type, Email, " _
   & "ManagerEmail, Building, Room, Phone from Employee"
```

然后，子过程扫描地址簿的四个输入框(See 1.1.6.1.4.7)。请复制并粘贴该代码到“查找”子程序中。

```
   If (SFirst.Value <> "") Then
      myQuery = myQuery + " where FirstName like '" + SFirst.Value + "%'"
   End If

   IF (SLast.Value <> "") Then
      myQuery = myQuery + " where LastName like '" + SLast.Value + "%'"
   End If

   If (STitle.Value <> "") Then
      myQuery = myQuery + " where Title like '" + STitle.Value + "%'"
   End If

   If (SEmail.Value <> "") Then
      myQuery = myQuery + " where Email like '" + SEmail.Value + "%'"
   End If
```

每个“If”语句检查相应的文本框中的内容。如果文本框包含文本，则执行“Then”语句，将引号内的文本追加到包含在变量 myQuery 中的全局 SELECT 语句。由于程序在建立 SQL 语句时使用单词“like”，查询将采用子字符串搜索，而不是完全匹配。

例如，如果“姓”框包含条目“Berge”并且“标题”框包含条目“程序管理员”，则 SQL 语句（myQuery 的值）将读取：

```
Select FirstName, LastName, Title, Email, Building, Room, Phone from Employee
where lastname like 'Berge%' and title like 'Program Manager%'
```

如查询成功，则所有姓中包含“Berge”的人（如 Berge 和 Berger），以及职务包含“程序管理员”的人（例如，“程序管理员”，“高级技术”）都被显示在数据网格(See 1.1.6.1.4.10)中。

准备和发送查询

子过程 Find_OnClick 的最后部分包含两个语句。第一个语句将 SQL(See 1.1.4.6.62) 对象的 RDS.DataControl(See 1.1.4.2.3) 查询属性赋给动态建立的 SQL 查询。第二个语句使 **RDS.DataControl** 对象 (SControl) 查询数据库，然后显示网格中查询的新结果。请复制并粘贴该代码到查找子程序中。

```
   SControl.SQL = myQuery
   SControl.Refresh
End Sub
```

“清除”按钮

下列 HTML 语句定义“清除”按钮。该 HTML 语句出现在程序的 VBScript 节之前。请复制并粘贴该代码到 Find HTML 按钮之后。

```
<INPUT TYPE=BUTTON NAME="Clear"    VALUE="Clear">
```

标记 INPUT 定义如按钮，选项按钮，复选框，或文本之类的元素。使用 TYPE 参数可指定元素，在这里即是按钮。参数 NAME 定义按钮在代码中被调用的内容。参数 VALUE 指定与显示在网页中的按钮 (Clear) 相关联的标签。

Clear_OnClick 子过程

单击 Clear 按钮可激活 VBScript Clear_OnClick 子过程。请复制并粘贴该代码到标记 SCRIPT 和 /SCRIPT 之间。

```
Sub Clear_OnClick
   SFirst.Value=""
   SLast.Value=""
   STitle.Value=""
   SEmail.Value=""
End Sub
```

执行子过程时，通过 ID 标记的参数 NAME 标识的四个输入框(See 1.1.6.1.4.7), 都被初始化。属性 .Value 指示显示在 Web 页中的文本框对象的字符。该过程用 0 长度串 ("") 替换所有文本，为新的查找做准备。

“更新配置文件”按钮

下列代码定义“更新配置文件”按钮。该 HTML 语句出现在程序的 VBScript 节之前。请复制并粘贴该 HTML 控件在“清除”按钮之后。

```
<INPUT TYPE=BUTTON NAME="Update"    VALUE="Update Profile">
```

标记 INPUT 定义如按钮，选项按钮，复选框或文本之类的元素。参数 NAME 定义代码中按钮被调用的内容。参数 TYPE 指定窗体元素的类型 — 在这里即是按钮。参数 VALUE 指定与按钮（“更新配置文件”）关联的标签。

Update_OnClick 子过程

单击“更新配置文件”按钮可激活 VBScript Update_OnClick 子过程，该子过程执行 RDS.DataControl(See 1.1.4.2.3) 对象 (SControl) 的 SubmitChanges(See 1.1.4.4.43) 和 Refresh(See 1.1.4.4.37) 方法。请复制并粘贴该代码到标记 SCRIPT 和 /SCRIPT 之间。

```
Sub Update_OnClick
   SControl.SubmitChanges
   SControl.Refresh
End Sub
```

执行 SControl.SubmitChanges 时，程序将所有更新信息打包，通过 HTTP 发送到服务器。更新要么全部更新要么不更新。如果部分更新不成功，将不做任何变更，并返回状态信息。在远程数据服务中，SControl.Refresh 不一定必需跟在 **SubmitChanges** 的后面，但这样可确保数据得以刷新。

“取消更改”按钮

下列代码定义“取消更改”按钮。该 HTML 语句出现在程序的 VBScript 节之前。请复制并粘贴该 HTML 控件到“更新”按钮之后。

```
<INPUT TYPE=BUTTON NAME="Cancel"    VALUE="Cancel Changes">
```

Cancel_OnClick 子过程

单击“取消更改”可激活 VBScript Cancel_OnClick 子过程，执行 **RDS.DataControl** 对象 (SControl) 的 CancelUpdate(See 1.1.4.4.9) 方法。请复制并粘贴该代码到标记 SCRIPT 和 /SCRIPT 之间。

```
Sub Cancel_OnClick
    SControl.CancelUpdate
End Sub
```

执行 SControl.CancelUpdate 时，它将放弃自从上一次查询或更新以来用户对数据网格(See 1.1.6.1.4.10)上雇员记录所作的任何编辑。由此恢复初始值。

1.1.6.1.4.10 地址簿数据网格

网格控件是类似电子表格的对象。地址簿应用程序用它显示由查询返回的数据。

下列 HTML 代码定义网格控件。请复制并粘贴该代码到下列引用它的注释标记的后面。

```
<OBJECT CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/ssdatb32.cab"
   ID=GRID1
DATASRC=#SControl
HEIGHT=125
WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="BackColor"   VALUE="-2147483643">
   <PARAM NAME="BackColorOdd"  VALUE="-2147483643">
   <PARAM NAME="ForeColorEven" VALUE="0">
</OBJECT>
<HR>
```

OBJECT 标记定义程序的网格控件部分。该标记包括用于下列操作的参数：设置网格大小、更改在信息更新或删除时显示的数据、并根据查询结果动态地更改行数。其他参数设置网格控件的颜色属性。

下表描述与 OBJECT 标记关联的参数。

参数	说明
CLASSID	唯一的 128 位数，标识系统的嵌入对象的类型。该标识符保留在本地计算机的 Windows 注册表中。它是 Sheridan 网格控件专有的。
CODEBASE	如果被调用对象的源文件不在客户机上，则该参数将指定它的位置。 在这里，CODEBASE 指定包含 Sheridan ActiveX® 控件的 .cab 文件的位置。程序使用 Active Server Pages 脚本（代码在 <% 和 %> 中间）来请求 URL 的服务器名。
ID	定义嵌入对象的文档宽度标识符。
DATASRC	标识用于绑定数据网格到后端数据（在这里即是 SControl(See 1.1.6.1.4.8) 它是 **RDS.DataControl** 对象）的数据源。
WIDTH 和 ***HEIGHT***	标识控件的尺寸，以像素为单位。

对地址簿应用程序，网格对象使用若干其他参数来启用编辑。如果没有这些参数，网格只能提供可编辑数据的静态只读显示。下表列出在该范例应用程序中网格控件的编辑参数设置。

参数	值	说明
AllowAddNew	TRUE	用于显示现有数据以添加新记录
AllowDelete	TRUE	用于显示现有数据以删除记录。

AllowUpdate	TRUE	用于对网格单元的更改。

参数 **BackColor**、**BackColorEven** 和 **BackColorOdd** 设置网格为白底黑字。

1.1.6.1.4.11 地址簿定位按钮

地址簿应用程序在 Web 页的底部显示定位按钮。通过选定数据的第一行或最后一行，或者通过选定与当前所选行相邻的行，可以使用定位按钮在网格显示(See 1.1.6.1.4.10)中给数据定位。

下列代码定义定位按钮。这些 HTML 语句出现在程序的 VBScript 节之前。请复制并粘贴这些控件到引用它们的注释标记的后面。

```
<INPUT TYPE=BUTTON NAME="First"       VALUE="First">
<INPUT TYPE=BUTTON NAME="Prev"       VALUE="Previous">
<INPUT TYPE=BUTTON NAME="Next"      VALUE="Next">
<INPUT TYPE=BUTTON NAME="Last"      VALUE="Last">
```

HTML 使用标记 INPUT 定义窗体元素，例如按钮、选项按钮、复选框或文本。参数 **TYPE** 指定窗体元素的类型，在这里即是按钮。参数 **NAME** 定义代码中按钮被调用的内容。参数 **VALUE** 指定与页面中显示的按钮(“第一”、“上一个”、“下一个”和“最后”)关联的标签。

用户单击按钮时，生成事件，VBScript 激活相应的定位子过程。

定位子过程

地址簿应用程序包含若干过程，使用户可以单击“第一”、“下一个”、“上一个”和“最后”按钮以便在数据中移动。要启用移动，可指定 RDS.DataControl(See 1.1.4.2.3) 对象 (Scontrol) 的方法为所需的移动类型。定位按钮的方法各不相同。

例如，单击“第一”按钮激活 VBScript First_OnClick 子过程。该过程调用 MoveFirst(See 1.1.4.4.30) 方法，使数据的第一行成为当前行。单击“最后”按钮激活 Last_OnClick 子过程，该过程调用 **MoveLast** 方法，使数据的最后行成为当前行。其余的定位按钮类似。请复制并粘贴该代码到标记 SCRIPT 和 /SCRIPT 之间。

```
' 在绑定的记录集中移动到第一个记录。
Sub First_OnClick
   SControl.Recordset.MoveFirst
End Sub

' 在绑定的记录集中从当前位置移动到下一个记录。
Sub Next_OnClick
   If SControl.Recordset.EOF Then   'cannot move beyond bottom record
      SControl.Recordset.MoveFirst
      SControl.Recordset.MoveNext
      Exit Sub
   End If

   SControl.Recordset.MoveNext

End Sub

' 在绑定的记录集中从当前位置移动到前一个记录。
Sub Prev_OnClick
   If SControl.Recordset.BOF Then   '移动无法超出顶端记录
      SControl.Recordset.MoveLast   '移出 BOF 缓冲区
```

```
      SControl.Recordset.MovePrevious
      Exit Sub
   End If
   SControl.Recordset.MovePrevious

End Sub

' 移动到绑定记录集的最后一个记录。
   SControl.Recordset.MoveLast
End Sub
```

1.1.6.1.4.12 VBScript 初始化代码

地址簿应用程序的 VBScript 节首先声明变量 myQuery。该变量将包含由搜索字段文本动态建立并发送到数据库的 SQL 查询。然后，Find_OnClick 子过程(See 1.1.6.1.4.9)设置该变量的值。

```
Dim myQuery
```

其余的初始化内容被包含在 Load 子过程中。其中包括给数据网格(See 1.1.6.1.4.10)添加标题。这样的 WHERE 子句使记录无法取出，从而只有列标题可以提供。请复制并粘贴该代码到标记 SCRIPT 和 /SCRIPT 之间。

```
Sub Load
   Grid1.Caption = "Arcadia Bay Corporate Phone Directory"

   ' 仅用列名称对数据网格进行初始化。
   SControl.SQL = "Select FirstName, LastName, Title, Email, " _
      & "Building, Room, Phone from Employee where 2 < 1"
   SControl.Refresh
End Sub
```

1.1.6.1.4.13 地址簿应用程序源代码

```
<HTML>
<HEAD>
<TITLE>Corporate Address Book</TITLE>
</HEAD>

<!--
   目的:    为 Web 用户提供公司目录搜索服务。
   编写者:    Microsoft Remote Data Service 工作组，Microsoft Corp.。
   日期:    1997 年 4 月
-->

<BODY BACKGROUND="Arcadia.gif" LANGUAGE="VBScript" onload="Load">
<tr>
   <td align="center" width="40%">
      <table border="2" cellpadding="7" cellspacing="7">
         <tr>
```

```
          <td width="100%"><font color="#160B5A"><font
          size="4"><strong>Arcadia Bay Corporate Phone
          Directory</strong></font></font></td>
        </tr>
     </table>
     </td>
</tr>

<hr>
<h2><font color = "#160B5A">Search Parameters</h2>
<h5><font color = "#160B5A">Please enter one or more search patterns and press
FIND to search.</h5>

<FONT COLOR = "#160B5A"><B>

<PRE> First Name   <INPUT NAME=SFirst SIZE=30> </PRE>

<PRE> Last Name  <INPUT NAME=SLast  SIZE=30> </PRE>

<PRE> Title    <INPUT NAME=STitle SIZE=30> </PRE>

<PRE> E-mail Alias <INPUT NAME=SEmail SIZE=30> </PRE>

<!--
   “命令”按钮选项：
   -----------------------
   查找      将搜索请求提交给数据库。
   清除      清除 QBE 字段（仅为任务保存功能）。
   更新配置文件   发送已更新的“地址配置文件”回数据库。
   取消更改   取消自从上次“更新配置文件”以来的所有更改。
-->

<INPUT TYPE=BUTTON NAME="Find"    VALUE="Find">
<INPUT TYPE=BUTTON NAME="Clear"    VALUE="Clear">
<INPUT TYPE=BUTTON NAME="Update"    VALUE="Update Profile">
<INPUT TYPE=BUTTON NAME="Cancel"    VALUE="Cancel Changes">

<hr>
<h2><font color = "#400040">Search Results</h2>
</B>
<br>

<!--
   This Sheridan DataGrid control (SGrid) is initialized to
   allow changes to the data - these changes will be saved
```

```
   to the database when the Update Profile button is pressed.
-->

<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/ssd
atb32.cab"
   ID=GRID1
    datasrc=#SControl
    HEIGHT=125
    WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="BackColor"   VALUE="-2147483643">
   <PARAM NAME="BackColorOdd"  VALUE="-2147483643">
   <PARAM NAME="ForeColorEven" VALUE="0">
</OBJECT>

<br>
<br>
<INPUT TYPE=BUTTON NAME="First"    VALUE="First">
<INPUT TYPE=BUTTON NAME="Prev"    VALUE="Previous">
<INPUT TYPE=BUTTON NAME="Next"   VALUE="Next">
<INPUT TYPE=BUTTON NAME="Last"   VALUE="Last">
<hr>


<!-- Non-visual controls - RDS.DataControl -->

<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=SControl
   WIDTH=1 HEIGHT=1>
   <PARAM                                                          NAME="SERVER"
VALUE="http://<%=Request.ServerVariables("SERVER_NAME")%>">
   <PARAM NAME="CONNECT" VALUE="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
</OBJECT>


<!-- VBS 脚本：撰写查询，更新配置文件，并检索搜索结果。 -->

<SCRIPT LANGUAGE="VBScript">

Dim myQuery
```

```
Sub Load
   Grid1.CAPTION = "Arcadia Bay Corporate Phone Directory"

   ' 仅使用列名称初始化数据网格。
   SControl.SQL = "Select FirstName, LastName, Title, Email, Building, Room,
Phone from Employee where 2 < 1"
   SControl.Refresh

End Sub

' 执行“清除”按钮 – 清除所有 QBE 字段以准备新的 “查找”。

Sub Clear_OnClick
   SFirst.Value=""
   SLast.Value=""
   STitle.Value=""
   SEmail.Value=""
End Sub


' 执行“查找”按钮 – 撰写由数据库处理的动态 SQL 查询，并返回将要绑定到 Sgrid 对象的匹配记录。

Sub Find_OnClick

   myQuery = "Select FirstName, LastName, Title, Type, Email, ManagerEmail,
Building, Room, Phone from Employee"

   ' 检查 QBE 字段并撰写动态的 SQL 查询。

   IF (SFirst.Value <> "") THEN
      myQuery = myQuery + " where FirstName like '" + SFirst.Value + "%'"
   End IF

   IF (SLast.Value <> "") THEN
      myQuery = myQuery + " where LastName like '" + SLast.Value + "%'"
   End IF

   IF (STitle.Value <> "") THEN
      myQuery = myQuery + " where Title like '" + STitle.Value + "%'"
   End IF

   IF (SEmail.Value <> "") THEN
      myQuery = myQuery + " where Email like '" + SEmail.Value + "%'"
   End IF
```

```
   ' 设置新查询然后刷新 SControl 以便显示新结果。

   SControl.SQL = myQuery
   SControl.Refresh

End Sub

' 定位子程序 - 基于 RDS.DataControl (SControl) 的货币变更。
' 在绑定的记录集中移动到第一个记录。

Sub First_OnClick
    SControl.Recordset.MoveFirst
End Sub

' 在绑定的记录集中从当前位置移动到下一个记录。Sub Next_OnClick
   If SControl.Recordset.EOF Then   ' 移动无法超出末端记录
      SControl.Recordset.MoveFirst
      SControl.Recordset.MoveNext
      Exit Sub
   End If

   SControl.Recordset.MoveNext

End Sub

' 在绑定的记录集中从当前位置移动到上一个记录.

Sub Prev_OnClick
If SControl.Recordset.BOF Then   ' 移动无法超出顶端记录
     SControl.Recordset.MoveLast   ' 移出 BOF 缓冲区
     SControl.Recordset.MovePrevious
     Exit Sub
   End If
   SControl.Recordset.MovePrevious
End Sub

' 在绑定的记录集中移动到最后一个记录。

Sub Last_OnClick
    SControl.Recordset.MoveLast
End Sub

' 提交所做编辑并获取新数据的干净拷贝。

Sub Update_OnClick
```

```
    SControl.SubmitChanges
  SControl.Refresh     ' SubmitChanges 之后的 ADC 1.5 不要求刷新，但它保证刷新的数据。
End Sub

' 取消编辑并恢复初始值。

Sub Cancel_OnClick
    SControl.CancelUpdate
End Sub

</SCRIPT>

<BR>
<font color = "#400040">This site powered by Microsoft Remote Data Service.
</font>
</BODY>
</HTML>
```

1.1.6.2 远程数据服务的范例应用程序

远程数据服务中包括多个范例应用程序，它们是十分有价值的学习工具。在整个文档中，您将会找到用以说明编程技术的范例应用程序及代码。您可以将其中的任何一部分复制到自己的应用程序中并进行必要的修改。

远程数据服务提供以下范例应用程序以便说明如何使用此技术构造 Intranet 和 Internet 应用程序。如果您已安装了应用程序，这些范例则可从以下位置获得；如果您尚未安装应用程序，可以从 MSDN® CD-ROM 中进行安装。

范例应用程序	**路径**	**说明**
ADCTest	安装到 *<device>*:\Program Files\Common Files\System\MSADC\ Samples\adctest.asp	可用于校验远程数据服务安装的简单 .asp 文件。
Address Book	安装到 ***<device>***:\Program Files\Common Files\System\MSADC\Samples\ AddressBook\addrbook.asp.	简单的公司目录应用程序。 关于此应用程序的详细信息，请参阅地址簿范例应用程序(See 1.1.6.1.4)。
Selector	安装到 ***<device>***:\Program Files\Common Files\System\MSADC\Samples\ Selector\SampStart.asp	演示如何在客户端、中间层以及数据源层面实现不同的远程数据服务应用程序配置。 **注意** Selector 不能在 Alpha 平台上运行。

这些范例以 .asp 格式提供（主机服务器名称将自动提供），不过您可以很方便地修改范例以便作为 .htm 页运行。

修改范例，使其以 .htm 格式运行

1. 在文本编辑器（如记事本）中打开 .asp 文件。
2. 将字符串 <%=Request.ServerVariables("SERVER_NAME")%> 替换为您的 Web 服务器名称。

注意 范例中所用公司、姓名以及数据除特别注明外均为虚构。

注意 安装范例应用程序时，将创建“adcdemo”帐号。在运行范例应用程序时，可能会出现登录失败消息，如果发生这种情况，只需重置用户名和密码以便测试帐号信息。

1.1.6.3 远程数据服务 (RDS) 开发人员指南

设计远程数据服务 (RDS) 的目的是为了构造以数据为中心的 Web 应用程序。它不仅结合了现有的技术，如 ActiveX®; Microsoft® Internet Information Server (IIS)、Microsoft® Visual Basic® Scripting Edition Script® 和 OLE DB，并且，它引进了创新技术，例如 Web 页面的增强数据绑定、客户端缓存以及调用自定义业务对象的功能。

在 RDS 应用程序中可以使用许多相关的技术。“开发人员指南”说明了三层 RDS 应用程序的不同部分，并且提供了一些代码范例，以说明如何使用这些技术。

本节包含以下内容的有关信息：

- 了解远程数据服务应用程序(See 1.1.6.3.1)
- 开发远程数据服务应用程序(See 1.1.6.3.2)
- 远程数据服务疑难解答(See 1.1.6.3.3)

1.1.6.3.1 了解远程数据服务应用程序

下面各节说明远程数据服务技术的体系结构和组件，并详细说明 RDS 客户端和服务器端如何协同工作以便：

- 通过 Internet 或 Intranet 将数据传送到您的 Web 页。
- 通过网络将已更新的信息发送回来，并合并到您的数据库中。

本节包含与以下内容有关的信息：

- 三层应用程序(See 1.1.6.3.1.1)
- 远程数据服务应用程序如何工作(See 1.1.6.3.1.2)
- 相关技术(See 1.1.6.3.1.3)

1.1.6.3.1.1 三层应用程序

要使用远程数据服务技术，必须要了解三层的客户端/服务器模型。该模型将客户端/服务器系统的不同组件分成三“层”：

- **客户端层** — Web 浏览器在其上显示 Web 页面的本地计算机，Web 页面可以显示并处理来自远程数据源的数据，或者（在不基于 Web 的应用程序中）是单独编译的前端应用程序。

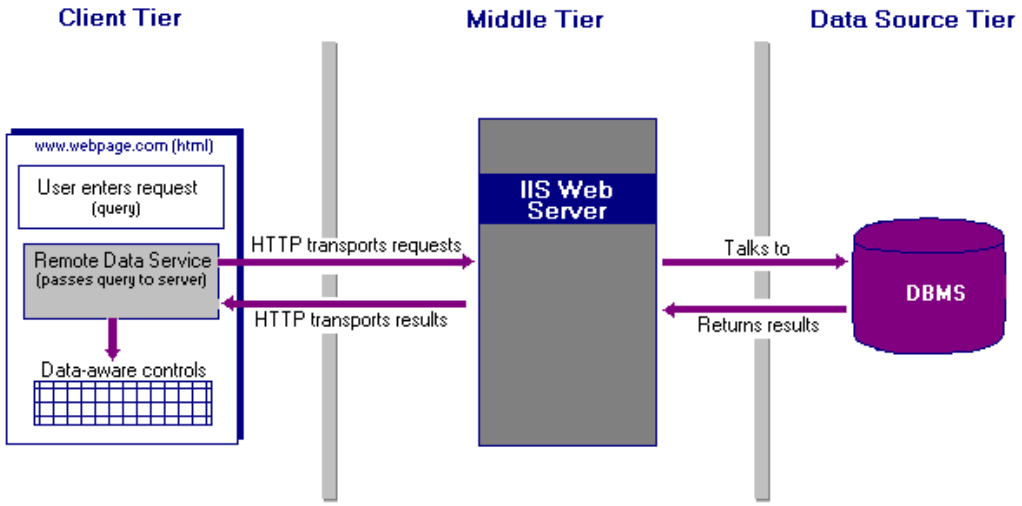
- **中间层** — 其组件封装了组织业务规则的 Microsoft® Windows NT® Server 主机。中间层组件可以是在 Internet Information Server 上执行的 Active Server Pages 脚本，或者是（在不基于 Web 的应用程序中）编译的可执行文件。

- **数据源层** — 宿主数据库管理系统 (DBMS) 计算机，该管理系统可以是 Microsoft® SQL Server® 数据库。（在两层应用程序中，中间层与数据源层组合在一起。）

这些层没有必要对应于网络上的物理位置。例如，所有三层可能只存在于两台计算机上。一台计算机可能是 Microsoft® Windows® 95 计算机，该计算机将 Microsoft® Internet Explorer 4.0 作为它的浏览器来运行。第二台计算机可能是运行 Internet Information Server 和 Microsoft SQL Server 的 Windows NT Server 计算机。为获得最高性能和维护的方便性而在网络上分配进程和数据时，用这种方式设计将使应用程序具有更大的灵活性。

1.1.6.3.1.2 远程数据服务应用程序的工作方式

在创建远程数据服务应用程序时，可以将应用程序分为两个或三个逻辑层。下面的图示说明基于 Web 的 RDS 应用程序如何处理用户请求显示数据库的信息。客户端组件一般情况下包含在 Internet Explorer 浏览器中，并使用 HTTP 与服务器组件进行通讯。Internet Explorer 4.0 已经包括 RDS 客户组件，因此简化了开发过程。



一旦用户输入请求，客户端 RDS 组件即发送查询给 Web 服务器。服务器端 RDS 组件处理请求并将其发送给 DBMS。DBMS 响应请求，发回数据。Web 服务器上的 RDS 组件将数据转换为 ADO Recordset(See 1.1.4.2.10) 对象。转换数据的目的是为了传输到客户端并通过网络发送回客户端计算机。它可以显示在数据识别(See 2.16)控件中，例如由 RDS.DataControl(See 1.1.4.2.3) 对象绑定到数据的文本框或窗格。一个 **RDS.DataControl** 可以将数据传送到许多数据识别的控件中。

结果数据将缓存在客户计端算机上，从而减少了到 Web 的连接次数并且使用户处理数据更方便。要求访问服务器的唯一调用是调用业务对象（例如对数据服务器的更新或者请求新数据）。

1.1.6.3.1.3 相关技术

许多其他技术与远程数据服务相互作用，或者在其实现过程中使用。

OLE DB 关于使用 OLE DB 的详细信息，请参阅 OLE DB Web 站点 http://www.microsoft.com/data/。

Internet Information Server 关于 Microsoft® Internet Information Server (IIS) 或 Active Server Pages (ASP) 的详细信息，请参阅 IIS Web 站点 http://www.microsoft.com/iis。以下是连接缓冲池的有关信息：

- 连接缓冲池选项(See 1.1.6.3.1.3.1)

Microsoft Transaction Server 关于 Microsoft® Transaction Server® 的详细信息，请参阅 MTS Web 站点 http://www.microsoft.com/transaction/。以下是 Microsoft Transaction Server 中的业务对象的有关信息：

- Microsoft Transaction Server 资源分配器(See 1.1.6.3.1.3.2)

Microsoft SQL Server 关于 Microsoft SQL Server 的详细信息，请参阅 SQL Server Web 站点 http://www.microsoft.com/sql/。以下是关于如何使用 SQL Server 与 RDS 的信息：

- 连接缓冲池的性能和稳定性(See 1.1.6.3.1.3.3)
- 保证足够的 TempDB 空间(See 1.1.6.3.1.3.4)
- 最小化日志文件空间的使用(See 1.1.6.3.1.3.5)

Microsoft Internet Explorer 关于 Microsoft Internet Explorer 的详细信息，请参阅 Internet Explorer Web 页：http://www.microsoft.com/ie/，以及 Microsoft Internet Client SDK Web 页：http://www.microsoft.com/msdn/sdk/inetsdk/asetup/。以下是关于数据绑定和绑定控件的信息：

- 通过绑定的控件显示数据(See 1.1.6.3.1.3.6)

Microsoft Windows NT Server 关于 Microsoft Windows NT Server 和 NT 安全性的详细信息，请参阅 NT Web 页：http://www.microsoft.com/ntserver/。以下是关于 NT 安全性和 RDS 的信息：

- 安全性和 Web 服务器 (See 1.1.6.3.1.3.7)

1.1.6.3.1.3.1 连接缓冲池选项

如果使用 ODBC 数据源，则可使用 Internet Information Server (IIS) 中的“连接缓冲池”选项来完成对客户端负载的高性能处理。“连接缓冲池”是用于连接的资源管理器，使频繁使用的连接保持打开状态。

要启用“连接缓冲池”，请参考 Internet Information Server 文档。

启用连接缓冲池可能会使 Web 服务器(See 2.51)受到其他限制，详细信息请参阅 Microsoft Internet Information Server 文档。

1.1.6.3.1.3.2 Microsoft Transaction Server 资源分配器

在三层环境中，如果在中间层上使用 Microsoft Transaction Server，那么客户可以共享数据库连接。可以使用 RDSServer.DataFactory(See 1.1.4.2.4) 对象或创建可为客户端共享设置 ODBC 连接的 ActiveX 组件 DLL。在 Transaction Server 运行时环境中运行 **RDSServer.DataFactory** 或自定义业务对象时，共享机制产生作用。只使用数百个而不是上千个数据库连接仍然可以支持上千个客户端。这是 Microsoft Transaction Server 中 ODBC 资源分配器的功能。

在 Microsoft Transaction Server 中运行业务对象

业务对象可以是可执行文件 (.exe) 或动态链接库 (.dll)。用于运行业务对象的配置取决于对象是 .dll 还是 .exe 文件：

- 创建为 .exe 文件的业务对象可以通过 DCOM(See 2.19) 调用。如果通过 Internet Information Server (IIS) 使用这些业务对象，那么它们将受到附加数据调度(See 2.29)的限制，这将降低客户端性能。
- 创建为 .dll 文件的业务对象可以通过 IIS（从而通过 HTTP）使用，并且只能通过 Microsoft Transaction Server(See 2.31) 在 DCOM 上使用。业务对象 DLL 需要在 IIS 计算机上注册以便用户

可通过 IIS 获得访问权限。RDSServer.DataFactory(See 1.1.4.2.4) 对象是远程数据服务提供的默认业务对象的 DLL，并且受到本节中的条件限制。（关于如何配置 DLL 以便在 DCOM 上运行的步骤，请参阅下一节“使 DLL 在 DCOM 上运行(See 1.1.9.5)”。）

通过在 MTS 运行时环境中运行 **RDSServer.DataFactory** 或自定义业务对象，也可以使用 MTS 资源分配器提高性能和缩放性。由于这些业务对象调用 ADO，而 ADO 间接调用 ODBC，因此可以使用 MTS ODBC 资源分配器。

资源分配器自动汇集和重复利用资源。因此，当 **RDSServer.DataFactory** 或者自定义业务对象释放数据库连接时，连接将返回到缓冲池中。当再次调用方法创建连接时，将会请求相同的数据库连接。ODBC 资源分配器可重复利用缓冲池中的连接，而不用创建新的连接，这样可以节省时间和服务器资源。

注意 当中间层上的业务对象作为 Microsoft Transaction Server 组件实现时（使用 **GetObjectContext**、**SetComplete** 和 **SetAbort**），通过多个客户端调用，该业务对象可以使用 Transaction Server 上下文对象来维护自身状态。使用 DCOM 可以实现这个方案，并且通常在委托的客户端和服务器（Intranet）之间实现。在这种情况下，客户端的 **RDS.DataSpace** 对象和 **CreateObject** 方法由事务的上下文对象和 **CreateInstance** 方法（由 **ITransactionContext** 接口提供）代替，并通过 Microsoft Transaction Server 实现。

1.1.6.3.1.3.3 连接缓冲池的性能和稳定性

要确保连接缓冲池的稳定性并且提供其他的性能，必须配置 Microsoft SQL 服务器以便使用 TCP/IP 套接字网络库。

为此，需要：

- 配置 SQL 服务器计算机以便使用 TCP/IP 套接字。
- 配置 Web 服务器以便使用 TCP/IP 套接字。

配置 SQL 服务器计算机以便使用 TCP/IP 套接字

在 SQL 服务器计算机上运行 SQL 服务器安装程序，以便在与数据源的交互操作中可以使用 TCP/IP 套接字网络库。

在 SQL 服务器计算机上指定 TCP/IP 套接字网络库

1. 从“开始”菜单，指向“程序”，再指向“Microsoft SQL Server 6.5”，然后单击“SQL 安装程序”。
2. 双击“继续”，屏幕上将出现“Microsoft SQL Server 6.5 — 选项”对话框。
3. 选择“更改网络支持”，然后单击“继续”。
4. 确认“TCP/IP 套接字”复选框被选中，然后单击“确定”。
5. 单击“继续”以完成指定过程，然后退出安装程序。

配置 Web 服务器以便使用 TCP/IP 套接字

有两个选项用于配置 Web 服务器以使用 TCP/IP 套接字。用户的行为取决于是从 Web 服务器访问所有 SQL 服务器，还是从 Web 服务器访问特定的 SQL 服务器。

如果从 Web 服务器访问所有 SQL 服务器，那么需要在 Web 服务器计算机上运行 SQL 服务器客户端配置实用程序。下列步骤将所有通过该 IIS Web 服务器创建 SQL 服务器连接所用的默认网络库更改为使用 TCP/IP 套接字网络库。

配置 Web 服务器（所有 SQL 服务器）

1. 从“开始”菜单，指向“程序”，再指向“Microsoft SQL Server 6.5”，然后单击“SQL 客户端配置实用程序”。
2. 选择“网络库”选项卡。
3. 在“默认网络”框中，选择“TCP/IP 套接字”。
4. 单击“完成”保存更改并退出实用程序。

如果从 Web 服务器访问特定的 SQL 服务器，则需要在 Web 服务器计算机上运行 SQL 服务器客户端配置实用程序。要更改特定 SQL 服务器连接的网络库，请在 Web 服务器计算机上按如下步骤配置 SQL 服务器客户端软件。

配置 Web 服务器（特定的 SQL 服务器）

1. 从“开始”菜单，指向“程序”，再指向“Microsoft SQL Server 6.5”，然后单击“SQL 客户端配置实用程序”。
2. 选择“高级”选项卡。
3. 在“服务器”框中，键入为使用 TCP/IP 套接字而连接的服务器名称。
4. 在“DLL 名称”框中，选择“TCP/IP 套接字”。
5. 单击“添加/修改”。现在所有指向该服务器的数据源都将使用 TCP/IP 套接字。
6. 单击“完成”。

1.1.6.3.1.3.4 保证足够的 TempDB 空间

如果在处理需要 SQL 服务器处理空间的 **Recordset** 对象时发生错误，则可能需要增加 TempDB 的大小。（某些查询需要临时处理空间。例如，带 ORDER BY 子句的查询需要一种 **Recordset**，而这需要部分临时空间。）

要点 请在执行操作之前阅读该步骤，因为设备对象一旦展开就难以折叠。

增加 SQL 服务器上的 TempDB 空间

1. 启动 Microsoft® SQL Server Enterprise Manager，打开“服务器”树，然后打开“数据库设备”树。
2. 选定要扩展的（物理）设备，如 Master，然后双击该设备打开“编辑数据库设备”对话框。

   该对话框显示当前数据库使用的空间大小。
3. 在“大小”滚动框中，将设备增加到所需量（例如，50 MB 的硬盘空间）。
4. 单击“立即更改”增加（逻辑）TempDB 可以扩展的空间大小。

5. 打开服务器上的“数据库”树，然后双击“TempDB”打开“编辑数据库”对话框。“数据库”选项卡列出当前分配给 TempDB 的空间大小（“数据大小”）。默认情况下是 2 MB。

6. 在“大小”组下，单击“扩展”。图形显示在每个物理设备上可用的和已分配的空间。栗色的条代表可用的空间。

7. 选定“日志设备”，例如 Master，在“大小 (MB)”框中显示可用的空间。

8. 单击“立即扩展”将空间分配给 TempDB 数据库。

“编辑数据库”对话框显示新分配的 TempDB 大小。

参阅 关于该主题的详细信息，请参阅 Mcirosoft SQL Server Enterprise Manager 帮助文件中的“扩展数据库对话框”。

1.1.6.3.1.3.5 最小化日志文件空间的使用

如果在 SQL 服务器数据库上有大量的活动，那么日志文件很快就会写满（从而中断服务器）。可以将日志文件设置为“在检查点截断”从而在很大程度上延长数据库日志文件的寿命。

启用“在检查点截断”

1. 运行 Microsoft SQL Server Enterprise Manager。

2. 打开“服务器”树。

3. 打开“数据库”树。

4. 双击要在其上启用该功能的数据库名称。

5. 在“数据库”选项卡上，单击“截断”。

6. 在“选项”选项卡上，选定“在检查点截断日志”，然后单击“确定”。

参见 关于“在检查点截断”功能的详细信息，请参阅 Microsoft SQL Server 文档。

1.1.6.3.1.3.6 通过绑定控件显示数据

绑定控件是一种数据识别控件，通过它可以访问数据库中的信息。当控件绑定到 RDS.DataControl(See 1.1.4.2.3) 对象时，远程数据服务将来自当前数据库记录的字段值应用到该控件。接着，控件显示数据并接受用户的更改。

用 Internet Explorer 4.0 可以绑定到简单的 HTML 控件。其他 ActiveX 数据识别控件可以从第三方得到，例如 Sheridan。

用 Internet Explorer 4.0 绑定数据的关键是两个关系：来源和使用者。数据使用者通过引用它们绑定的数据源来指定绑定。为此，数据使用者（诸如文本框或窗格的控件）指定数据源和绑定的数据类型。

多数绑定控件以下列 <OBJECT> 标记的数据识别参数为特征。这些参数是可用于 Internet Explorer 4.0 的 HTML 数据绑定扩展。

参数	说明
DATASRC	指示用于数据绑定的数据源（**RDS.DataControl** 对象）。

DATAFLD	指示数据集的列。
DATAFORMATAS	（可选）指示绑定的数据类型。

对于特定的字段可以有多个绑定控件，但是不需要对表中的每个字段都提供绑定控件。

在访问 Web 页前端时，**RDS.DataControl** 与数据库一起为用户提供对 **Recordset** 的访问，然后可以通过定位按钮将 **Recordset** 定位，定位按钮使用 **RDS.DataControl** 对象的 MoveFirst、MoveLast、MoveNext 和 MovePrevious(See 1.1.4.4.30) 方法。也可以使用 **RDS.DataControl** 对象的 SubmitChanges(See 1.1.4.4.43) 方法将更新内容发送到数据库。

1.1.6.3.1.3.7 安全性和 Web 服务器

如果在 Internet Web 服务器上使用 **RDSServer.DataFactory**，请记住这会引起潜在的安全性问题。获得有效数据源名称 (DSN)、用户标识符和密码信息的外部用户可以编写将任何查询发送到该数据源的页面。如果希望对数据源有更多的限制访问，一种选择是取消注册并删除 **RDSServer.DataFactory** 对象 (msadcf.dll)，并且使用带硬代码查询的自定义业务对象替代。下一节将说明怎么做。

客户端模拟和安全性

如果 IIS Web 服务器的“密码验证”属性设置为“Windows NT 请求/响应”，那么业务对象将在客户端的安全性上下文中被调用。这是 RDS 1.5 中的新功能，允许通过 HTTP 进行客户端模拟。在这种模式下工作时，对 Web 服务器 (IIS) 的登录不是匿名的，而是使用客户端计算机运行时使用的用户 ID 和密码。如果 ODBC DSN 设置为使用“委托的连接”，那么对诸如 SQL Server 的数据库的访问也发生在客户端的安全性上下文中。但只有在数据库与 IIS 在相同的计算机上时才可行。不能将客户端身份证转移到其他计算机。

例如，客户 John Doe 要登录到客户计算机，他的用户标识符=“JohnD”，密码=“secret”。他运行基于浏览器的应用程序，该应用程序需要访问 **RDSServer.DataFactory** 对象并通过在运行 IIS 的“MyServer”计算机上执行 SQL 查询来创建 **ADOR.Recordset**。MyServer 设置为使用“Windows NT 请求/响应”用户密码验证，它的 ODBC DSN 已经选定了“使用委托的连接”，同时服务器包含 SQL Server 数据源。当在 Web 服务器上接收到请求时，它要求客户输入用户 ID 和密码。因此登录到 MyServer 上的请求来自“JohnD”/“Secret”而不是 IUSER_MyServer（“匿名密码验证”打开时的默认情况）。类似地，当登录到 SQL Server 时将使用“JohnD”/“Secret”。

因此，在没有明确提示用户输入登录到数据库所需的用户 ID 和密码信息的情况下,IIS NT 请求/响应验证模式允许创建 HTML 页面。如果使用“IIS 基本验证”，也要求使用该模式。

密码验证

RDS 可以与在任何下列三种密码验证模式下运行的 IIS Web 服务器通讯：匿名、基本或 NT 请求/响应。这些设置定义 Web 服务器如何通过它控制访问，例如请求客户计算机在 NT Web 服务器上有明确的访问特权。

1.1.6.3.2 开发远程数据服务应用程序

为基于 Web 的数据访问应用程序编写代码时，如果希望使用远程数据服务的数据绑定和远程调用功能，则需要了解所涉及的不同组件。

前一节了解远程数据服务应用程序(See 1.1.6.3.1)介绍了三层 RDS 应用程序的体系结构，并且解释了客户端和服务器端组件如何通过相互通讯在客户和服务器之间发送数据。

本节包含与如下内容有关的信息：

- 将 **Recordset** 返回客户端(See 1.1.6.3.2.1)
- 将更新的 **Recordset** 对象传送到中间层(See 1.1.6.3.2.2)
- 定义 **Recordset**(See 1.1.6.3.2.3)

注意 请将整个代码范例粘贴到您的代码编辑器中。否则，如果使用的是部分范例或丢失了段落格式，范例可能无法正确运行。

1.1.6.3.2.1 将 Recordset 返回客户端

使用 **RDS** 可以有三种方式将 **Recordset** 从服务器返回给客户端。这三种方式分别为：

- 使用自动调用 **RDSServer.DataFactory** 对象的 **RDS.DataControl** 方法和属性。(See 1.1.6.3.2.1.1)
- 手工调用 **RDSServer.DataFactory** 对象。(See 1.1.6.3.2.1.2)
- 创建执行数据访问功能的自定义 **ActiveX DLL**。(See 1.1.6.3.2.1.3)

本节说明如何使用以上方法传送 **Recordset** 对象。

1.1.6.3.2.1.1 用 DataControl 对象获得 Recordset

可以通过设置 RDS.DataControl(See 1.1.4.2.3) 对象的 Connect(See 1.1.4.6.13)、Server(See 1.1.4.6.55) 和 SQL(See 1.1.4.6.62) 属性来打开未连接的 **Recordset**。

下面的代码显示如何在设计时设置这些属性：

```
<OBJECT CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID="ADC1">
   <PARAM NAME="SQL" VALUE="Select * from Products">
   <PARAM NAME="Connect" VALUE="DSN=AdvWorks;">
   <PARAM NAME="Server" VALUE="http://SalesWeb/">
</OBJECT>
```

在 **RDS.DataControl** 上设置这些属性后调用 **Refresh** 方法，将在“幕后”自动调用 RDSServer.DataFactory(See 1.1.4.2.4) 对象，并且远程数据服务将 **Recordset** 对象返回给客户端。实际上不需要编写代码来使用 **RDSServer.DataFactory**，但如果确实需要，请参阅“用 RDSServer.DataFactory 对象获得 Recordset(See 1.1.6.3.2.1.2)”。

1.1.6.3.2.1.2 用 DataFactory 对象获得 Recordset

Remote Data Service 包含服务器端的业务对象 (ActiveX® DLL)，称为 RDSServer.DataFactory(See 1.1.4.2.4)，可将命令发送到数据源并将结果通过 Internet 或 Intranet 传回给用户。**RDSServer.DataFactory** 是默认的 ActiveX DLL，使用它可以通过少量的编程在 Web 页面上提供活动数据。

下面的范例显示如何从 **VBScript Web** 页面调用 **RDSServer.DataFactory** 对象。使用客户的 RDS.DataSpace(See 1.1.4.2.5) 对象在服务器上创建 **RDSServer.DataFactory** 对象的实例。

```
<HTML>
<HEAD></HEAD>
<BODY>
```

```
<!-- RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID=ADC1>
</OBJECT>
<!-- RDS.DataSpace -->
<OBJECT ID="ADS1" WIDTH=1 HEIGHT=1
  CLASSID="CLSID:BD96C556-65A3-11D0-983A-00C04FC29E36">
</OBJECT>
.
.
.
<SCRIPT LANGUAGE="VBScript">
Option Explicit
Sub Window_OnLoad()
  Dim ADF1, myRS
  Set ADF1 = ADS1.CreateObject("RDSServer.DataFactory", _
  "http://<%=Request.ServerVariables("SERVER_NAME")%>")
  Set myRS = _
  ADF1.Query("DSN=pubs;UID=sa;PWD=permission;", _
  "Select * From Authors")
  ' Assign the returned recordset to SourceRecordset.
  ADC1.SourceRecordset = myRS
End Sub
</SCRIPT>
</BODY>
</HTML>
```

1.1.6.3.2.1.3 用自定义业务对象获得 Recordset

如果不想使用 RDSServer.DataFactory(See 1.1.4.2.4) 对象将 **Recordset** 对象返回给客户端，可以创建自己的自定义业务对象并且在服务器上运行。DLL(动态链接库) 可以是用 Microsoft® Visual Basic 和 Microsoft® Visual C++® 等创建的任何种类的 Automation 对象，或者是带 Active Server Pages 脚本代码的服务器端的 HTML 页面。客户应用程序和 **Web** 首尾应用程序调用业务对象以实现特定的功能，并且这些 中间层(See 2.32) 业务对象随之可以与后端数据库通讯。

自定义 DLL 也包含简单的 **RDSServer.DataFactory** ActiveX DLL 没有提供的方法。这些方法不必与数据访问相关 — 它们可以只包含业务规则。

本节包含与如下内容有关的信息：

- 编写代码以便用自定义的 ActiveX DLL 传送 Recordset 对象(See 1.1.6.3.2.1.4)

1.1.6.3.2.1.4 编写代码以便用自定义的 ActiveX DLL 传送 Recordset 对象

除了使用自定义业务对象以外，下列客户端 Microsoft® Visual Basic® Scripting Edition 代码实现的操作与前面的 RDSServer.DataFactory 代码(See 1.1.6.3.2.1.2)相同。您仍然可以在客户端上使用 RDS.DataSpace(See 1.1.4.2.5) 对象创建服务器中业务对象的实例（在这种情况下是 MyCustomBusinessObject）。

```
<HTML>
<HEAD></HEAD>
<BODY>

<!-- RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID=ADC1>
</OBJECT>
<!-- RDS.DataSpace -->
<OBJECT ID="ADS1" WIDTH=1 HEIGHT=1
  CLASSID="CLSID:BD96C556-65A3-11D0-983A-00C04FC29E36">
</OBJECT>
.
.
.
<SCRIPT LANGUAGE="VBScript">
Option Explicit
Sub GetRecords()
  Dim objMyCustomBusinessObject, myRS
  Set objMyCustomBusinessObject = _
  ADS1.CreateObject("MyCustomBusinessObject", _
  "http://<%=Request.ServerVariables("SERVER_NAME")%>")
  ' 假定 MyCustomBusinessObject 具有
  ' 称为 GetData 的方法，该方法可获得连接字符串和 SQL 参数。
  ' parameters.
  Set myRS = _
  objCustomBusinessObject.GetData _
  ("DSN=pubs;UID=sa;PWD=permission;", _
  "Select * From Authors")
  ' 将返回的记录集赋给  SourceRecordset。
  ADC1.SourceRecordset = myRS
End Sub
</SCRIPT>
</BODY>
</HTML>
```

假定使用 Visual Basic 创建位于中间层上的 MyCustomBusinessObject ActiveX DLL，那么 MyCustomBusinessObject 类中 **GetData** 方法的代码应该与如下代码类似。注意您可以直接使用 ActiveX® 数据对象 (ADO)。

```
'返回 ADO 结果集。
Public Function GetData(szCnStr As String, szSQL _
  As String) As Object

  Dim cn As New ADODB.Connection
  Dim rs As New ADODB.Recordset

  cn.Open szCnStr
```

```
  ' ADODB.Recordset 应该生成 Recordset
  ' 对象，该对象可以被断开，随后
  ' 再恢复连接以处理批更新。
  rs.CursorLocation = adUseClientBatch
  ' 使用非指定参数，ADO/R
  '记录集被返回。
  rs.Open szSQL, cn, _
  adOpenUnspecified, adLockUnspecified, _
  adCmdUnspecified
  Set GetData = rs
End Function
```

提示

- 在试图将 Recordset 对象返回之前，把一个简单的方法放在服务器组件中来测试最小的功能。
- 在用 Internet Explorer 部署和测试服务器组件之前，构造简单的客户应用程序对它进行测试。
- 在本地测试 Web 服务器上开发应用程序要更方便一些。在每次编译后需要在测试服务器上复制并注册 .dll。
- 传送到业务对象的 DSN 应该是服务器上的“系统 DSN”。如果它不存在或没有正确设置，那么用户组件将失败。最好用其它的 ODBC 应用程序（例如 MSQuery）来测试服务器上的 DSN 以确保正确设置 DSN。
- 自定义业务对象(See 2.7)上的方法名不能超过 255 个字符，以确保能兼容 RDS 支持的所有协议（HTTP、HTTPS、DCOM 和进程内）。
- 如果使用 Visual Basic 创建的自定义业务对象使用以前与 ADOR 1.0 类型库的绑定，那么应该将自定义业务对象重新构造为使用 ADOR 2.0 类型库。

1.1.6.3.2.2 将更新的 Recordset 对象传送给中间层

使用下列对象可以将更新的 **Recordset** 对象从客户端计算机传送到中间层和数据库服务器：

- RDS.DataControl 对象(See 1.1.6.3.2.2.1)
- ADO 与中间层自定义业务对象(See 1.1.6.3.2.2.2)

1.1.6.3.2.2.1 使用 DataControl 将更新的未连接 Recordset 对象传送回中间层

数据绑定控件允许用户进行可视化编辑、添加或删除记录。用户在显式提交或取消更新之前所做的所有更改将在本地保存。

一般情况下，用户将窗格控件绑定到 RDS.DataControl(See 1.1.4.2.3) 对象，然后通过用户接口添加、编辑和删除客户端 **Recordset** 中的记录。在更新客户端记录后，需要使用 **RDS.DataControl** 对象中的 SubmitChanges(See 1.1.4.4.43) 方法将更改的信息保存到

数据库中。**SubmitChanges** 方法将把在本地缓存中可更新的、被挂起的 **Recordset** 的更改提交给 **OLE DB** 数据源，该数据源在 **RDS.DataControl** 对象 Connect(See 1.1.4.6.13) 属性中指定。

下面的代码范例显示如何完成这项工作：

```
Sub Update_OnClick
   ADC1.SubmitChanges
End Sub
```

只有更改的记录才被发送以便进行修改，所有改动只能同时成功或同时失败。

也可以将“取消”按钮包括进来以取消对 **Recordset** 的更改：

```
Sub Cancel_OnClick
   ADC1.CancelUpdate
End Sub
```

注意

- 在使用 **RDS.DataControl** 对象的 **SubmitChanges** 方法之前必须设置 **Connect**、Server(See 1.1.4.6.55) 和 SQL(See 1.1.4.6.62) 属性。这些属性用于重新连接到数据源。
- 如果在调用相同 **Recordset** 对象的 **SubmitChange** 后调用 CancelUpdate(See 1.1.4.4.6) 方法，那么 **CancelUpdate** 调用将失败，因为此时更改结果已经提交。

1.1.6.3.2.2.2 使用 ADO 将 Recordset 对象传送到中间层

可以使用 **ADOR.Recordset** 对象将 Recordset(See 2.41) 对象从客户端 Web 页面调度(See 2.29)到中间层业务对象(See 2.32)。例如，假定用户连接到虚拟购物中心并选定了要购买的物品。选定的物品将显示在虚拟购物车中，该购物车是用 RDS.DataControl(See 1.1.4.2.3) 对象实现的，并且缓存在行集合中。当客户单击购买按钮时，**ADOR.Recordset** 对象将被创建并且作为对业务函数 (ApplyUpdates) 的输入参数传送到应用程序服务器。这将使 **Recordset** 被调度到服务器。然后 ApplyUpdates 业务函数连接到 Sales 数据库并应用更新。

```
' 客户端 Web 页面的代码。
Sub PurchaseItem_OnClick
   Set rst = ADC1.Recordset

' 以下选项通知记录集在更新时
' 仅返回已更改的记录。由此使得往返更轻松。
   rst.MarshalOptions = adMarshalModifiedOnly

' 调用 MyObj 业务对象的 ApplyUpdates 功能
' 并将 ADOR.Recordset 对象作为输入参数传递。
   MyObj.ApplyUpdates rst

End Sub


' 业务对象中的 VB 代码
' ApplyUpdates 是中间层业务对象中的方法。
Sub ApplyUpdates(rst As ADOR.Recordset)
```

```
   rst.ActiveConnection =  _
      "DSN=SalesDB;UID=SMgr;PWD=password"

' 保存更改记录。
   rst.UpdateBatch

End Sub
```

1.1.6.3.2.3 定义 Recordset

用户可以创建 **ADODB.Recordset** 对象并指定列信息，然后可以将数据插入到 **Recordset** 对象中。基本行集合(See 2.42) 将插入的数据缓存。

下面的代码范例显示如何使用 RDSServer.DataFactory(See 1.1.4.2.4) 对象定义 **Recordset**。也可以用 RDS.DataControl(See 1.1.4.2.3) 对象进行同样的定义。

```
Sub RsDefineShape()

   Dim vntRecordShape(3)
   Dim vntField1Shape(3)
   Dim vntField2Shape(3)
   Dim vntField3Shape(3)
   Dim vntField4Shape(3)

   '  给每个字段指定名称、类型、大小和空属性。
   vntField1Shape(0) = "Name"   ' Column name.
   vntField1Shape(1) = CInt(129)   ' Column type.
   vntField1Shape(2) = CInt(40)   ' Column size.
   vntField1Shape(3) = False     ' Nullable?

   vntField2Shape(0) = "Age"
   vntField2Shape (1) = CInt(3)
   vntField2Shape (2) = CInt(-1)
   vntField2Shape (3) = True

   vntField3Shape (0) = "DateOfBirth"
   vntField3Shape (1) = CInt(7)
   vntField3Shape (2) = CInt(-1)
   vntField3Shape (3) = True

   vntField4Shape (0) = "Balance"
   vntField4Shape (1) = CInt(6)
   vntField4Shape (2) = CInt(-1)
   vntField4Shape (3) = True

   ' 将所有字段放入一个数组中。
   vntRecordShape(0) = vntField1Shape
```

```
vntRecordShape(1) = vntField2Shape
vntRecordShape(2) = vntField3Shape
vntRecordShape(3) = vntField4Shape

' 使用 RDSServer.DataFactory 创建空记录集，
' 该记录集是一个变体型数组，其中
' 每个元素本身又是另一个变体型数组
' 一个变体型数组是记录集中的一列。
' 内部数组的元素是列的名称、类型、大小和空属性。

Dim NewRs

' 可以使用 RDS.DataControl 对象
' 代替 RDSServer.DataFactory 对象。
' 在这种情况下，如下代码将被设置为 Set NewRS。
' = ADC1.CreateRecordset(vntRecordShape)
Set NewRS = ADF.CreateRecordset(vntRecordShape)

Dim fields(3)
fields(0) = vntField1Shape(0)
fields(1) = vntField2Shape (0)
fields(2) = vntField3Shape (0)
fields(3) = vntField4Shape (0)

' 给新记录集填充新数据值。
Dim fieldVals(3)

' 使用 AddNew 添加记录。
fieldVals(0) = "Joe"
fieldVals(1) = 5
fieldVals(2) = CDate(#1/5/96#)
fieldVals(3) = 123.456
NewRS.AddNew fields, fieldVals

fieldVals(0) = "Mary"
fieldVals(1) = 6
fieldVals(2) = CDate(#6/5/96#)
fieldVals(3) = 31
NewRS.AddNew fields, fieldVals

fieldVals(0) = "Alex"
fieldVals(1) = 13
fieldVals(2) = CDate(#1/6/96#)
fieldVals(3) = 34.0001
NewRS.AddNew fields, fieldVals
```

```
    fieldVals(0) = "Susan"
    fieldVals(1) = 13
    fieldVals(2) = CDate(#8/6/96#)
    fieldVals(3) = 0.0
    NewRS.AddNew fields, fieldVals

    NewRS.MoveFirst

    ' 将新创建和填充的记录集设置为
    ' RDS.DataControl 的 SourceRecordset 属性
    ' 来绑定可视控件。
    Set ADC1.SourceRecordset = NewRS

End Sub
```

1.1.6.3.3 远程数据服务疑难解答

以下各节介绍指定错误的解决方案。

- “Internet 服务器错误：拒绝访问”(See 1.1.6.3.3.1)
- 运行范例应用程序时出现“未知错误”消息(See 1.1.6.3.3.2)
- 使用带 Sheridan 组合框控件的 RDS.DataControl(See 1.1.6.3.3.3)
- 可重复读取隔离级出现死锁(See 1.1.6.3.3.4)
- RDS.DataControl 和多个记录集请求(See 1.1.6.3.3.5)
- 过期的类 ID(See 1.1.6.3.3.6)

1.1.6.3.3.1 Internet 服务器错误：拒绝访问

如果发生该错误，通常意味着 Microsoft® Internet Information Server (IIS) 返回以下状态：

HTTP_STATUS_DENIED 401

请确保 IIS 访问的目录有正确的权限。RDS 1.5 可以与在三种密码验证模式（匿名、基本或 NT 请求/响应）中任何一种模式下运行的 IIS Web 服务器通讯。与以前版本不同的是不再需要“允许匿名”设置。同时，如果 Web 服务器(See 2.51)是 Windows NT® 计算机，则必须对数据源计算机有访问权限。

1.1.6.3.3.2 运行范例应用程序时出现“未知错误”消息

- 如果重新安装 IIS，则必须随后重新安装远程数据服务。

- 如果使用 Microsoft SQL Server 作为后端数据源，请使用 Microsoft SQL Server 6.5、Service Pack 2 或更高版本。
- 请确保 ODBC DSN 是“系统 DSN(See 2.18)”而不是“用户 DSN”。通过 IIS 登录的匿名用户只能查看“系统 DSN”。
- 请检查 WWW 服务使用的匿名帐号是否是来自对所有相关服务器和目录拥有 NTFS 权限的域中的用户。最后，请确认该匿名帐号的密码没有过期。
- 由于远程数据服务安装在 C:\Program Files\Common Files\System\MSADC 目录中，请检查 IIS Service Manager 的用户是否可以访问该目录。它应该可以进行 **Read** 和 **Execute** 访问。
- 有关远程数据服务 DLL 和范例应用程序的问题，请使用 Regsvr32.exe 重新注册 DLL。
- 如果正在编写自己的业务对象，请将 Active Server Pages 脚本文件 (.asp) 放在“匿名”用户（在 IIS 中）有 **Execute** 权限的目录中。

1.1.6.3.3.3 使用带 Sheridan 组合框控件的 DataControl

Sheridan 组合框有两个属性可以绑定到 RDS.DataControl(See 1.1.4.2.3) 数据。只将组合框绑定到 **RDS.DataControl** 并不能自动填充“Text”部分（显示当前所选项目的不可绑定属性）。结果在调用 Refresh(See 1.1.4.4.37) 方法后仍然显示旧的数据。

解决办法是使用类似以下两个范例 Microsoft® Visual Basic®, Scripting Edition Sub 过程的代码。二者均可在调用 **Refresh** 方法时，清除 **Text** 属性。这些范例中 Sheridan 组合框控件的名称是 SSCombo。

```
Sub SSCombo_InitColumnProps()
   SSCombo.Text=""
End Sub

Sub Refresh_OnClick
   ADC1.Refresh
   SSCombo.Text=""
End Sub
```

1.1.6.3.3.4 可重复读取隔离级出现死锁

如果自定义业务对象(See 2.7)使用可重复读取隔离级访问 SQL 服务器，并且两个在相同事务中发送查询和更新的客户端同时调用业务对象，那么可能发生死锁。远程数据服务允许其中一个进程超时以便释放死锁，但这会使客户端的更新失败。

在 Web 服务器上添加注册表项可以修改超时的长度。默认为 30 秒（或者 30,000 毫秒）。

修改超时值

1. 从“开始”菜单，单击“运行”。
2. 键入“RegEdit”并单击“确定”。
3. 在注册表编辑器中，定位到：
   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\W3SVC\Parameters 键。

4. 选择“参数”键。从“编辑”菜单，指向“新建”，然后选定“DWORD 值”。

5. 在“名称”列中，键入“ADCUpdateTimeout”并按 Enter。

6. 双击新条目。在“编辑 DWORD 值”对话框中，将新值（以毫秒为单位）输入“值数据”框中。该值必须大于或等于 0。

1.1.6.3.3.5 DataControl 和多个记录集请求

如果查询多个结果，那么仅返回第一个 Recordset。如果需要多个结果集，请将它们分配给各自的 RDS.DataControl(See 1.1.4.2.3)。多个结果的查询范例可以是“Select * from Authors, Select * from Topics”。

1.1.6.3.3.6 过期的类 ID

如果获得没有加载 **RDS.DataSpace** 或 **RDS.DataControl** 对象的错误，请确认是否使用了正确的类 ID。这些对象的类 ID 已经从版本 1.0 更改到 1.1。正确的类 ID 如下所列。

对象	类 ID
RDS.DataControl(See 1.1.4.2.3)	BD96C556-65A3-11D0-983A-00C04FC29E33
RDS.DataSpace(See 1.1.4.2.5)	BD96C556-65A3-11D0-983A-00C04FC29E36

1.1.6.4 ADO 代码范例

使用如下代码范例学习如何使用 ADO 对象、方法和属性。

注意 请将整个代码范例（从 Sub 到 End Sub）粘贴到您的代码编辑器中。如果使用部分范例或丢失了段落格式，范例可能无法正确运行。

- ADO 对象范例(See 1.1.6.4.1)
- ADO 方法范例(See 1.1.6.4.2)
- ADO 属性范例(See 1.1.6.4.3)

1.1.6.4.1 ADO 对象范例

使用如下代码范例学习如何使用 ADO 对象。

注意 请将整个代码范例（从 Sub 到 End Sub）粘贴到您的代码编辑器中。如果使用部分范例或丢失了段落格式，范例可能无法正确运行。

- RDS.DataControl 对象范例 (VBScript)(See 1.1.6.4.1.1)

- RDS.DataSpace 对象和 CreateObject 方法范例 (VBScript)(See 1.1.6.4.1.2)
- RDSServer.DataFactory 对象、Query 方法和 CreateObject 方法范例 (VBScript)(See 1.1.6.4.1.3)

1.1.6.4.1.1 DataControl 对象范例 (VBScript)

以下代码显示如何在设计时设置 **RDS.DataControl** 参数以及如何使用称为 ADCDemo 的 ODBC 数据源将其绑定到数据标识的控件当中。如果您按照指导操作，ADCDemo 将作为 SQL Server ODBC 数据源安装到服务器上。请将其剪切并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为 **ADCapi1.asp**。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples</H2>
<HR><BR>
<H3>Remote Data Service</H3>
<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/Sheridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
</OBJECT>
<!-- 设计时设置的具有参数的 Remote Data Service -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
   <PARAM NAME="SQL" VALUE="Select * from Employee for browse">
   <PARAM                                                    NAME="SERVER"
VALUE="http://<%=Request.ServerVariables("SERVER_NAME")%>">
   <PARAM NAME="CONNECT" VALUE="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
</OBJECT><BR><HR></Center>
```

以下范例显示如何在运行时设置 **RDS.DataControl** 的必要参数。要测试该范例，请将其剪切并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为 **ADCapi2.asp**。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples </H2>
<HR><BR>
<H3>Remote Data Service Run Time</H3>
<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/Sheridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
```

```
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="Remote Data Service Run Time">
</OBJECT>

<!-- 设计时设置的不带参数的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
   </OBJECT>
<HR>
<Input                          Size=70                          Name="txtServer"
Value="http://<%=Request.ServerVariables("SERVER_NAME")%>"><BR>
<Input Size=70 Name="txtConnect" Value="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
<BR>
<Input Size=70 Name="txtSQL" Value="Select * from Employee">
<HR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run"><BR>
<H4>Fill Grid with these values or change them to see data from another ODBC
data source on your server</H4>
</Center>
<Script Language="VBScript">
<!--
' 在运行时设置 RDS.DataControl 的参数
Sub Run_OnClick
   ADC.Server = txtServer.Value
   ADC.SQL = txtSQL.Value
   ADC.Connect = txtConnect.Value
   ADC.Refresh
End Sub
-->
</Script>
```

1.1.6.4.1.2 DataSpace 对象和 CreateObject 方法范例 (VBScript)

以下范例显示如何使用具有默认业务对象 **RDSServer.DataFactory** 的 **RDS.DataSpace** 的 **CreateObject** 方法。要测试该范例，请将其剪切并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为 **ADCapi8.asp**。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples </H2>
<HR><H3>Using Query Method of RDSServer.DataFactory</H3>

<!-- RDS.DataSpace  ID ADS1-->
<OBJECT ID="ADS1" WIDTH=1 HEIGHT=1
CLASSID="CLSID:BD96C556-65A3-11D0-983A-00C04FC29E36">
```

```
</OBJECT>

<!-- 运行时设置的具有参数的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
</OBJECT>

<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/She
ridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="RDSServer.DataFactory Run Time">
</OBJECT>
<HR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run"><BR>
<H4>Click Run. The CreateObject Method of the RDS.DataSpace Object Creates an
instance of the RDSServer.DataFactory.
The Query Method of the RDSServer.DataFactory is used to bring back a Recordset.
</H4>
</Center>
<Script Language="VBScript">
<!--
Dim ADF
Dim strServer
Dim strConnect
Dim strSQL
strServer = "http://<%=Request.ServerVariables("SERVER_NAME")%>"
strConnect = "dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;"
strSQL = "Select * from Employee"

Sub Run_OnClick()
   Dim objADORs      '创建 Recordset 对象
Set ADF = ADS1.CreateObject("RDSServer.DataFactory", strServer)           '获取
Recordset
Set objADORs = ADF.Query(strConnect, strSQL)
' 使用 RDS.DataControl 将 Recordset 绑定到网格敏感控件

   ADC.SourceRecordset = objADORs
```

```
End Sub
-->
</Script>
```

以下范例显示如何使用 **CreateObject** 方法创建自定义业务对象 VbBusObj.VbBusObjCls 的实例。该范例还使用 Active Server Pages 脚本标识 Web 服务器名。要查看完整的范例，请通过范例应用程序的选择器，在“客户端层”列中选择“VBScript in Internet Explorer”，并在“中间层”列中选择“自定义 Visual Basic 业务对象”。

```
Sub Window_OnLoad()
   strServer = "http://<%=Request.ServerVariables("SERVER_NAME")%>"
   Set BO = ADS1.CreateObject("VbBusObj.VbBusObjCls", strServer)
   txtConnect.Value = "dsn=pubs;uid=sa;pwd=;"
   txtGetRecordset.Value = "Select * From authors for Browse"
End Sub
```

1.1.6.4.1.3 DataFactory Object 、 Query 方 法 和 CreateObject 方法范例 (VBScript)

该范例使用 **RDS.DataSpace** 对象的 **CreateObject** 方法创建 **RDSServer.DataFactory** 对象。要测试该范例，请将该代码剪切并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，并命名为 **ADCapi7.asp**。ASP 脚本将标识您的服务器。

```
<Center><H2>RDS API Code Examples</H2>
<HR><H3>Using Query Method of RDSServer.DataFactory</H3>

<!-- RDS.DataSpace  ID ADS1-->
<OBJECT ID="ADS1" WIDTH=1 HEIGHT=1
CLASSID="CLSID:BD96C556-65A3-11D0-983A-00C04FC29E36">
</OBJECT>

<!-- 运行时设置的具有参数的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
</OBJECT>

<Object classid ="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"
   CODEBASE="http://<%=Request.ServerVariables _
   ("SERVER_NAME")%>/MSADC/Samples/Sheridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE=" RDSServer.DataFactory Run Time">
</OBJECT>
```

```
<HR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run"><BR>
<H4>Click Run. The CreateObject Method of the
RDS.DataSpace Object Creates an instance of the
RDSServer.DataFactory.
The Query Method of the RDSServer.DataFactory is used
to bring back a Recordset. </H4>
</Center>
<Script Language="VBScript">
<!--
Dim ADF
Dim strServer
Dim strConnect
Dim strSQL

strServer = "http://<%=Request.ServerVariables _
("SERVER_NAME")%>"
strConnect = "dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;"
strSQL = "Select * from Employee"

Sub Run_OnClick()
' 创建 RDSServer.DataFactory 对象
   Dim objADORs
' 获得记录集
Set ADF = ADS1.CreateObject("RDSServer.DataFactory", strServer)
Set objADORs = ADF.Query(strConnect, strSQL)
' 设置 RDS.DataControl 运行时参数
   ADC.Server = strServer
   ADC.SQL = strSQL
   ADC.Connect = strConnect
   ADC.Refresh
End Sub
-->
</Script>
```

1.1.6.4.2 ADO 方法范例

请使用如下主题了解如何使用 ADO 方法。

注意 请从 Sub 到 End Sub 将代码范例整体粘贴到您的代码编辑器。如果使用部分范例或丢失段落格式，范例可能无法正确运行。

1.1.6.4.2.1 AddNew 方法范例

该范例使用 AddNew 方法创建具有指定名称的新记录。

```
Public Sub AddNewX()

   Dim cnn1 As ADODB.Connection
   Dim rstEmployees As ADODB.Recordset
   Dim strCnn As String
   Dim strID As String
   Dim strFirstName As String
   Dim strLastName As String
   Dim booRecordAdded As Boolean

   ' 打开连接。
   Set cnn1 = New ADODB.Connection
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=;"
   cnn1.Open strCnn

   ' 打开雇员表。
   Set rstEmployees = New ADODB.Recordset
   rstEmployees.CursorType = adOpenKeyset
   rstEmployees.LockType = adLockOptimistic
   rstEmployees.Open "employee", cnn1, , , adCmdTable

   ' 从用户获取数据，雇员 ID 的格式必须为：
   ' 名、中间名和姓的三个首字母，
   ' 五位数字，以及性别标识 M 或 F。
   ' 例如，Bill Sornsin 的雇员 ID 为：B-S55555M。
   strID = Trim(InputBox("Enter employee ID:"))
   strFirstName = Trim(InputBox("Enter first name:"))
   strLastName = Trim(InputBox("Enter last name:"))

   ' P 只在用户输入姓和名之后进行。
   If (strID <> "") And (strFirstName <> "") _
      And (strLastName <> "") Then

      rstEmployees.AddNew
      rstEmployees!emp_id = strID
      rstEmployees!fname = strFirstName
      rstEmployees!lname = strLastName
      rstEmployees.Update
      booRecordAdded = True
```

```
    ' 显示新添加的数据。
    MsgBox "New record: " & rstEmployees!emp_id & " " & _
       rstEmployees!fname & " " & rstEmployees!lname

  Else
    MsgBox "Please enter an employee ID, " & _
       "first name, and last name."
  End If

  ' 删除新记录，因为这只是演示。
  cnn1.Execute "DELETE FROM employee WHERE emp_id = '" & strID & "'"

  rstEmployees.Close
  cnn1.Close

End Sub
```

VBScript 版本

下面是使用 VBScript 编写、并用于 Active Server Page (ASP) 的相同范例。如需查看该完整功能范例，请使用与 IIS 一同安装并位于 C:\InetPub\ASPSamp\AdvWorks 的数据源 AdvWorks.mdb，来创建名为 AdvWorks 的系统“数据源名称”(DSN)。这是 Microsoft Access 数据库文件。请使用“查找”命令定位文件 Adovbs.inc，并将其放入计划使用的目录中。请将以下代码剪切并粘贴到“记事本”或其他文本编辑器中，另存为 **AddNew.asp**。这样，便可在任何客户端浏览器中查看结果。

如要执行此范例，请按 HTML 格式添加虚构的新记录，单击“添加新记录”。查看 **Delete** 方法范例可删除不需要的记录。

```
<!-- #Include file="ADOVBS.INC" -->
<% Language = VBScript %>
<HTML><HEAD><TITLE>ADO Open Method</TITLE>
</HEAD><BODY>
<FONT FACE="MS SANS SERIF" SIZE=2>
<Center><H3>ADO AddNew Method</H3>
<!-- ADO Connection Object used to create recordset-->
<%
'创建并打开 Connection 对象。
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "AdvWorks"
'创建并打开 Recordset 对象。
Set RsCustomerList = Server.CreateObject("ADODB.Recordset")
RsCustomerList.ActiveConnection = OBJdbConnection
RsCustomerList.CursorType = adOpenKeyset
RsCustomerList.LockType = adLockOptimistic
RsCustomerList.Source = "Customers"
RsCustomerList.Open

%>
<!-- 如果这是第一次打开页面，则输入数据时 Form 集合将为空。
```

```
请运行 AddNew 方法-->
<% If Not IsEmpty(Request.Form) Then
   If Not Request.Form("CompanyName") = "" Then
      RsCustomerList.AddNew
      RsCustomerList("CompanyName") = Request.Form("CompanyName")
      RsCustomerList("ContactLastName") = Request.Form("LastName")
      RsCustomerList("ContactFirstName") = Request.Form("FirstName")
      RsCustomerList("PhoneNumber") = Request.Form("PhoneNumber")
      RsCustomerList("City") = Request.Form("City")
      RsCustomerList("StateOrProvince") = Request.Form("State")
      RsCustomerList.Update
      RsCustomerList.MoveFirst

   End If
End If
%>

<TABLE COLSPAN=8 CELLPADDING=5 BORDER=0>

<!-- Customer 表的 BEGIN 列标头行-->

<TR><TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Company Name</FONT></TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Contact Name</FONT></TD>
<TD ALIGN=CENTER WIDTH=150 BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Phone Number</FONT></TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>City</FONT></TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT              STYLE="ARIAL              NARROW"              COLOR="#ffffff"
SIZE=1>State/Province</FONT></TD></TR>
<!--每通过一遍记录集，显示一行 Customer 表的 ADO 数据-->
<% Do While Not RsCustomerList.EOF %>
  <TR><TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RSCustomerList("CompanyName")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("ContactLastName") & ", " %>
  <%= RScustomerList("ContactFirstName") %>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
```

```
  <FONT STYLE="ARIAL NARROW" SIZE=1>
   <%= RScustomerList("PhoneNumber")%>
 </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("City")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("StateOrProvince")%>
  </FONT></TD></TR>
<!-- Next Row = Record Loop 并添加到行 html 表-->
<%
RScustomerList.MoveNext
Loop
%>

</TABLE><HR>
<!-- 输入新记录的窗体将变量返回该页 -->
<Table>
<Form Method = Post Action="AddNew.asp" Name=Form>
<TR><TD><P>Company Name:</TD>
<TD><Input Type="Text" Size="50" Name="CompanyName" Value = ""></P></TD>
<TR><TD><P>Contact First Name:</TD>
<TD><Input Type="Text" Size="50" Name="FirstName" Value = ""></P></TD>
<TR><TD><P>Contact Last Name:</TD>
<TD><Input Type="Text" Size="50" Name="LastName" Value = ""></P></TD>
<TR><TD><P>Contact Phone:</TD>
<TD><Input Type="Text" Size="50" Name="PhoneNumber" Value = ""></P></TD>
<TR><TD><P>City:</TD>
<TD><Input Type="Text" Size="50" Name="City" Value = ""></P></TD>
<TR><TD><P>State / Province:</TD>
<TD><Input Type="Text" Size="5" Name="State" Value = ""></P></TD>
<TR><TD><Input Type="Submit" Value="Add New "><Input Type="Reset" Value="Reset
Form">
</Form></Table></Center></FONT>
<%'Show location of DSN data source
Response.Write(OBJdbConnection)
%>
<Script Language = "VBScript">
Sub Form_OnSubmit
   MsgBox "Sending New Record to Server",,"ADO-ASP _Example"
End Sub
</Script>
</BODY></HTML>
```

1.1.6.4.2.2 Append 和 CreateParameter 方法范例

```
Public Sub AppendX()

   Dim cnn1 As ADODB.Connection
   Dim cmdByRoyalty As ADODB.Command
   Dim prmByRoyalty As ADODB.Parameter
   Dim rstByRoyalty As ADODB.Recordset
   Dim rstAuthors As ADODB.Recordset
   Dim intRoyalty As Integer
   Dim strAuthorID As String
   Dim strCnn As String

   ' 打开连接。
   Set cnn1 = New ADODB.Connection
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   cnn1.Open strCnn
   cnn1.CursorLocation = adUseClient

   ' 使用一个参数打开命令对象。
   Set cmdByRoyalty = New ADODB.Command
   cmdByRoyalty.CommandText = "byroyalty"
   cmdByRoyalty.CommandType = adCmdStoredProc

   ' 获取参数值并追加参数。
   intRoyalty = Trim(InputBox("Enter royalty:"))
   Set prmByRoyalty = cmdByRoyalty.CreateParameter("percentage", _
      adInteger, adParamInput)
   cmdByRoyalty.Parameters.Append prmByRoyalty
   prmByRoyalty.Value = intRoyalty

   ' 通过执行命令创建记录集。
   Set cmdByRoyalty.ActiveConnection = cnn1
   Set rstByRoyalty = cmdByRoyalty.Execute

   ' 打开 Authors 表以获取作者姓名进行显示。
   Set rstAuthors = New ADODB.Recordset
   rstAuthors.Open "authors", cnn1, , , adCmdTable

   ' 打印记录集中的当前数据，从 Authors 表中添加作者姓名。
   Debug.Print "Authors with " & intRoyalty & " percent royalty"
   Do While Not rstByRoyalty.EOF
      strAuthorID = rstByRoyalty!au_id
```

```
    Debug.Print "   " & rstByRoyalty!au_id & ", ";
    rstAuthors.Filter = "au_id = '" & strAuthorID & "'"
    Debug.Print rstAuthors!au_fname & " " & rstAuthors!au_lname
    rstByRoyalty.MoveNext
  Loop

  rstByRoyalty.Close
  rstAuthors.Close
  cnn1.Close

End Sub
```

1.1.6.4.2.3 AppendChunk 和 GetChunk 方法范例

```
Public Sub AppendChunkX()

  Dim cnn1 As ADODB.Connection
  Dim rstPubInfo As ADODB.Recordset
  Dim strCnn As String
  Dim strPubID As String
  Dim strPRInfo As String
  Dim lngOffset As Long
  Dim lngLogoSize As Long
  Dim varLogo As Variant
  Dim varChunk As Variant

  Const conChunkSize = 100

  ' 打开连接。
  Set cnn1 = New ADODB.Connection
    strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  cnn1.Open strCnn

  ' 打开 pub_info 表。
  Set rstPubInfo = New ADODB.Recordset
  rstPubInfo.CursorType = adOpenKeyset
  rstPubInfo.LockType = adLockOptimistic
  rstPubInfo.Open "pub_info", cnn1, , , adCmdTable

  ' 提示复制徽标。
  strMsg = "Available logos are : " & vbCr & vbCr
  Do While Not rstPubInfo.EOF
    strMsg = strMsg & rstPubInfo!pub_id & vbCr & _
      Left(rstPubInfo!pr_info, InStr(rstPubInfo!pr_info, ",") - 1) & _
```

```
      vbCr & vbCr
   rstPubInfo.MoveNext
Loop
strMsg = strMsg & "Enter the ID of a logo to copy:"
strPubID = InputBox(strMsg)

' 将徽标大块复制到变量中。
rstPubInfo.Filter = "pub_id = '" & strPubID & "'"
lngLogoSize = rstPubInfo!logo.ActualSize
Do While lngOffset < lngLogoSize
   varChunk = rstPubInfo!logo.GetChunk(conChunkSize)
   varLogo = varLogo & varChunk
   lngOffset = lngOffset + conChunkSize
Loop

' 从用户处得到数据。
strPubID = Trim(InputBox("Enter a new pub ID:"))
strPRInfo = Trim(InputBox("Enter descriptive text:"))

' 添加新记录，大块复制徽标。
rstPubInfo.AddNew
rstPubInfo!pub_id = strPubID
rstPubInfo!pr_info = strPRInfo

lngOffset = 0 ' 重置位移。
Do While lngOffset < lngLogoSize
   varChunk = LeftB(RightB(varLogo, lngLogoSize - lngOffset), _
      conChunkSize)
   rstPubInfo!logo.AppendChunk varChunk
   lngOffset = lngOffset + conChunkSize
Loop
rstPubInfo.Update

 ' 显示新添加的数据。
MsgBox "New record: " & rstPubInfo!pub_id & vbCr & _
   "Description: " & rstPubInfo!pr_info & vbCr & _
   "Logo size: " & rstPubInfo!logo.ActualSize

' 删除新记录，因为这只是演示。
rstPubInfo.Requery
cnn1.Execute "DELETE FROM pub_info " & _
   "WHERE pub_id = '" & strPubID & "'"

rstPubInfo.Close
cnn1.Close
```

```
End Sub
```

1.1.6.4.2.4 BeginTrans、CommitTrans 和 RollbackTrans 方法范例

该范例更改数据库的 Titles 表中所有心理学书籍的书籍类型。在 **BeginTrans** 方法启动事务将所有对 Titles 表的更改隔离后，**CommitTrans** 方法将保存更改。可使用 **Rollback** 方法撤销用 **Update** 方法保存的更改。

```
Public Sub BeginTransX()

   Dim cnn1 As ADODB.Connection
   Dim rstTitles As ADODB.Recordset
   Dim strCnn As String
   Dim strTitle As String
   Dim strMessage As String

   ' 打开连接。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set cnn1 = New ADODB.Connection
   cnn1.Open strCnn

   ' 打开 Titles 表。
   Set rstTitles = New ADODB.Recordset
   rstTitles.CursorType = adOpenDynamic
   rstTitles.LockType = adLockPessimistic
   rstTitles.Open "titles", cnn1, , , adCmdTable

   rstTitles.MoveFirst
   cnn1.BeginTrans

   ' 在记录集中循环并询问是否想要更改指定标题的类型。
   Do Until rstTitles.EOF
      If Trim(rstTitles!Type) = "psychology" Then
         strTitle = rstTitles!Title
         strMessage = "Title: " & strTitle & vbCr & _
         "Change type to self help?"

         ' 更改指定雇员的标题。
         If MsgBox(strMessage, vbYesNo) = vbYes Then
            rstTitles!Type = "self_help"
            rstTitles.Update
         End If
      End If
```

```
      rstTitles.MoveNext
   Loop

   ' 询问用户是否想提交以上所做的全部更改。
   If MsgBox("Save all changes?", vbYesNo) = vbYes Then
      cnn1.CommitTrans
   Else
      cnn1.RollbackTrans
   End If

   ' 打印记录集中的当前数据。
   rstTitles.Requery
   rstTitles.MoveFirst
   Do While Not rstTitles.EOF
      Debug.Print rstTitles!Title & " - " & rstTitles!Type
      rstTitles.MoveNext
   Loop

   ' 恢复原始数据，因为这只是演示。
   rstTitles.MoveFirst
   Do Until rstTitles.EOF
      If Trim(rstTitles!Type) = "self_help" Then
         rstTitles!Type = "psychology"
         rstTitles.Update
      End If
      rstTitles.MoveNext
   Loop

   rstTitles.Close
   cnn1.Close

End Sub
```

1.1.6.4.2.5 Cancel 方法范例

```
Public Sub CancelX()

   Dim cnn1 As ADODB.Connection
   Dim strCnn As String
   Dim strCmdChange As String
   Dim strCmdRestore As String
   Dim booChanged As Boolean

   ' 打开连接。
```

```
   Set cnn1 = New ADODB.Connection
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   cnn1.Open strCnn

   ' 定义命令字符串。
   strCmdChange = "UPDATE titles SET type = 'self_help' " & _
      "WHERE type = 'psychology'"
   strCmdRestore = "UPDATE titles SET type = 'psychology' " & _
      "WHERE type = 'self_help'"

   ' 开始事务，然后异步执行命令。
   cnn1.BeginTrans
   cnn1.Execute strCmdChange, , adAsyncExecute

   ' 做一会其他的事情（可以将其更改）。
      For i = 1 To 10
         i = i + i
         Debug.Print i
      Next i

   ' 如果命令没有完成，取消执行并回卷事务。否则提交事务。
   If CBool(cnn1.State And adStateExecuting) Then
      cnn1.Cancel
      cnn1.RollbackTrans
      booChanged = False
      MsgBox "Update canceled."
   Else
      cnn1.CommitTrans
      booChanged = True
      MsgBox "Update complete."
   End If

   ' 如果已经更改，则恢复数据，因为这只是演示。
   If booChanged Then
      cnn1.Execute strCmdRestore
      MsgBox "Data restored."
   End If

   cnn1.Close

End Sub
```

1.1.6.4.2.6 Cancel 方法范例 (VBScript)

以下范例说明如何使用 VBScript 代码在运行时读取 **Cancel** 方法。

```
<Script Language="VBScript">
<!--
Sub cmdCancelAsync
ADC.Cancel
' 终止当前运行的、设置为 adcReadyStateLoaded 的
' AsyncExecute 和 ReadyState 属性，
' 并终止当前运行的、设置为 Nothing 的 Recordset。
End Sub
-->
</Script>
```

1.1.6.4.2.7 CancelUpdate 方法范例 (VBScript)

如要测试该范例，请剪切该代码并粘贴到规范的 HTML 文档的 <Body> 和 </Body> 标记之间，将其命名为 **ADCapi6.asp**。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples</H2>
<HR><H3>Remote Data Service SubmitChanges and CancelUpdate Methods</H3>

<!-- 在设计时设置的带有参数的 RDS.DataControl   -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
   <PARAM NAME="SQL" VALUE="Select * from Employee for browse">
   <PARAM                                                    NAME="SERVER"
VALUE="http://<%=Request.ServerVariables("SERVER_NAME")%>">
   <PARAM NAME="CONNECT" VALUE="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
</OBJECT>

<Object classid="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/She
ridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="Remote Data Service Run Time">
</OBJECT>
<HR>
```

```
<INPUT    TYPE=BUTTON    NAME="SubmitChange"    VALUE="Submit-Changes"><INPUT
TYPE=BUTTON NAME="CancelChange" VALUE="Cancel-Update"><BR>
<H4>Add a new person or alter a current entry on the grid. Move off that Row.
<BR>
Submit the Changes to your DBMS or cancel the updates. </H4>
</Center>
<Script Language="VBScript">
<!--
' 在运行时设置 RDS.DataControl 的参数。
Sub SubmitChange_OnClick
   ADC.SubmitChanges
   ADC.Refresh
End Sub

Sub CancelUpdate_OnClick
   ADC.CancelUpdate
End Sub
-->
</Script>
```

1.1.6.4.2.8 Clone 方法范例 (Visual Basic)

该范例使用 Clone 方法创建 **Recordset** 的副本并让用户独立地为每个副本的记录指针定位。

```
Public Sub CloneX()

   Dim arstStores(1 To 3) As ADODB.Recordset
   Dim intLoop As Integer
   Dim strSQL As String
   Dim strCnn As String
   Dim strMessage As String
   Dim strFind As String

   ' 将 SQL 语句和连接字符串赋值给变量。
   strSQL = "SELECT stor_name FROM Stores " & _
      "ORDER BY stor_name"
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "

   ' 将记录集作为静态游标类型记录集打开。
   Set arstStores(1) = New ADODB.Recordset
   arstStores(1).CursorType = adOpenStatic
   arstStores(1).LockType = adLockBatchOptimistic
   arstStores(1).Open strSQL, strCnn, , , adCmdText

   ' 创建原始 Recordset 的两个副本。
```

```
   Set arstStores(2) = arstStores(1).Clone
   Set arstStores(3) = arstStores(1).Clone

   Do While True

      ' 在数组中循环以使用户每一遍都搜索相同 Recordset 的不同副本。
      For intLoop = 1 To 3

         ' 要求在显示每个 Recordset 当前记录集指针位置时搜索字符串。
         strMessage = _
            "Recordsets from stores table:" & vbCr & _
            "  1 - Original - Record pointer at " & _
            arstStores(1)!stor_name & vbCr & _
            "  2 - Clone - Record pointer at " & _
            arstStores(2)!stor_name & vbCr & _
            "  3 - Clone - Record pointer at " & _
            arstStores(3)!stor_name & vbCr & _
            "Enter search string for #" & intLoop & ":"
         strFind = Trim(InputBox(strMessage))
         If strFind = "" Then Exit Do

         ' 查找搜索字符串，如果没有匹配的，请跳转到最后的记录。
         arstStores(intLoop).Filter = "stor_name >= '" & strFind & "'"
         If arstStores(intLoop).EOF Then
            arstStores(intLoop).Filter = adFilterNone
            arstStores(intLoop).MoveLast
         End If

      Next intLoop

   Loop

   arstStores(1).Close
   arstStores(2).Close
   arstStores(3).Close

End Sub
```

VBScript 版本

下面是使用 VBScript 编写、并用于 Active Server Page (ASP) 的相同范例。如需查看该完整功能范例，请使用与 IIS 一同安装并位于 C:\InetPub\ASPSamp\AdvWorks 的数据源 AdvWorks.mdb，来创建名为 AdvWorks 的系统“数据源名称”(DSN)。这是 Microsoft Access 数据库文件。请使用“查找”命令定位文件 Adovbs.inc，并将其放入计划使用的目录中。请将以下代码剪切并粘贴到“记事本”或其他文本编辑器中，另存为“Clone.asp”。这样，便可在任何客户端浏览器中查看结果。

如要执行该范例，请将行 RsCustomerList.Source = "Customers" 改为 RsCustomerList.Source = "Products" 以便为更大的表计数。

```
<!-- #Include file="ADOVBS.INC" -->
<% Language = VBScript %>
<HTML><HEAD>
<TITLE>ADO Clone Method</TITLE>
</HEAD><BODY> <Center>
<H3>ADO Clone Method</H3>
<!--- A用于创建记录集的 ADO Connection 对象 -->
<%
' 创建并打开 Connection 对象。
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "AdvWorks"
' 创建并打开 Recordset 对象。
Set RsCustomerList = Server.CreateObject("ADODB.Recordset")
RsCustomerList.ActiveConnection = OBJdbConnection
RsCustomerList.CursorType = adOpenKeyset
RsCustomerList.LockType = adLockOptimistic
RsCustomerList.Source = "Customers"
RsCustomerList.Open
%>
<HR>
<!-- 在 Customers 表中循环，每循环一次将 Counter 变量加 1 -->
<%
   Set MyRecordset = RSCustomerList.Clone
   Counter = 0
   Do Until MyRecordset.EOF
      Counter = Counter + 1
      MyRecordset.MoveNext
   Loop
%>
<!-- 显示结果 -->
<H3>There Are <%=Counter %> Records in the Customers Table</H3>
<BR><HR>
<H4>Location of DSN Datbase</H4>
<%' Show location of DSN data source
Response.Write(OBJdbConnection)
%>
<HR></Center></BODY></HTML>
```

1.1.6.4.2.9 ConvertToString 方法范例 (VBScript)

```
Sub QueryRecordset_OnClick()
   Dim objADORs
      strServer = "http://<%=Request.ServerVariables("SERVER_NAME")%>"
      Set ADF = ADS1.CreateObject _
      ("RDSServer.DataFactory", strServer)
```

```
   Set objADORs = ADF.Query _
   (txtConnect.Value,txtQueryRecordset.Value)

ADF.ConvertToString(objADORs)

End Sub
```

1.1.6.4.2.10 CreateRecordset 方法范例 (VBScript)

```
Sub CreateARecordSet
   Dim ColInfo(1), c0(3), c1(3)

   c0(0) = "Name"         ' 列名称。
   c0(1) = CInt(129)      ' 列类型 (129 = adChar)。
   c0(2) = CInt(40)       ' 列大小。
   c0(3) = False          ' 列可否为空?

   c1(0) = "Age"          ' 列名称。
   c1(1) = CInt(3)        ' 列类型 (3 = adInteger)。
   c1(2) = CInt(-1)       ' 列大小。
   c1(3) = True           ' 列可否为空?

   ' 将列添加到记录集定义。
   ColInfo(0) = c0
   ColInfo(1) = c1

   ADC1.SourceRecordset = ADF1.CreateRecordset(ColInfo)
End Sub
```

1.1.6.4.2.11 Delete 方法范例

该范例使用 **Delete** 方法从 **Recordset** 删除指定的记录。

```
Public Sub DeleteX()

   Dim rstRoySched As ADODB.Recordset
   Dim strCnn As String
   Dim strMsg As String
   Dim strTitleID As String
   Dim intLoRange As Integer
   Dim intHiRange As Integer
   Dim intRoyalty As Integer

   ' 打开 RoySched 表。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
```

```
Set rstRoySched = New ADODB.Recordset
rstRoySched.CursorLocation = adUseClient
rstRoySched.CursorType = adOpenStatic
rstRoySched.LockType = adLockBatchOptimistic
rstRoySched.Open "SELECT * FROM roysched " & _
   "WHERE royalty = 20", strCnn, , , adCmdText

' 提示删除记录。
strMsg = "Before delete there are " & _
   rstRoySched.RecordCount & _
   " titles with 20 percent royalty:" & vbCr & vbCr
Do While Not rstRoySched.EOF
   strMsg = strMsg & rstRoySched!title_id & vbCr
   rstRoySched.MoveNext
Loop
strMsg = strMsg & vbCr & vbCr & _
   "Enter the ID of a record to delete:"
strTitleID = UCase(InputBox(strMsg))

' 移动到记录并保存数据以使其可被恢复。
rstRoySched.Filter = "title_id = '" & strTitleID & "'"
intLoRange = rstRoySched!lorange
intHiRange = rstRoySched!hirange
intRoyalty = rstRoySched!royalty

' 删除记录。
rstRoySched.Delete
rstRoySched.UpdateBatch

' 显示结果。
rstRoySched.Filter = adFilterNone
rstRoySched.Requery
strMsg = ""
strMsg = "After delete there are " & _
   rstRoySched.RecordCount & _
   " titles with 20 percent royalty:" & vbCr & vbCr
Do While Not rstRoySched.EOF
   strMsg = strMsg & rstRoySched!title_id & vbCr
   rstRoySched.MoveNext
Loop
MsgBox strMsg

' 恢复数据，因为这只是演示。
rstRoySched.AddNew
rstRoySched!title_id = strTitleID
```

```
   rstRoySched!lorange = intLoRange
   rstRoySched!hirange = intHiRange
   rstRoySched!royalty = intRoyalty
   rstRoySched.UpdateBatch

   rstRoySched.Close

End Sub
```

VBScript 版本

下面是使用 VBScript 编写、并用于 Active Server Page (ASP) 的相同范例。如需查看该完整功能范例，请使用与 IIS 一同安装并位于 C:\InetPub\ASPSamp\AdvWorks 的数据源 AdvWorks.mdb，来创建名为 AdvWorks 的系统“数据源名称”(DSN)。这是 Microsoft Access 数据库文件。请使用“查找”命令定位文件 Adovbs.inc，并将其放入计划使用的目录中。请将以下代码剪切并粘贴到“记事本”或其他文本编辑器中，另存为“Delete.asp”。这样，便可在任何客户端浏览器中查看结果。

如要执行该范例，请先使用 **AddNew** 范例添加一些记录，然后可将其删除。并在任一客户端浏览器中查看结果。

```
<!-- #Include file="ADOVBS.INC" -->
<% Language = VBScript %>

<HTML>

<HEAD><TITLE>ADO Delete Method</TITLE>
</HEAD><BODY>
<FONT FACE="MS SANS SERIF" SIZE=2>
<Center><H3>ADO Delete Method</H3>

<!--- 用于创建记录集的 ADO Connection 对象 -->
<%
' 创建并打开 Connection 对象。
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "AdvWorks"
' 创建并打开 Recordset 对象。
Set RsCustomerList = Server.CreateObject("ADODB.Recordset")
RsCustomerList.ActiveConnection = OBJdbConnection
RsCustomerList.CursorType = adOpenKeyset
RsCustomerList.LockType = adLockOptimistic
RsCustomerList.Source = "Customers"
RsCustomerList.Open
%>
<!-- 移动到指定记录并将其删除 -->
<%

If Not IsEmpty(Request.Form("WhichRecord")) Then
   ' 获取值以便从 Form Post 方法移动
   Moves = Request.Form("WhichRecord")
```

```
   RsCustomerList.Move CInt(Moves)
   If Not RsCustomerList.EOF or RsCustomerList.BOF Then
      RsCustomerList.Delete 1
      RsCustomerList.MoveFirst

   Else
      Response.Write "Not a Valid Record Number"
      RsCustomerList.MoveFirst
   End If
End If

%>
<!-- Customer 表的 BEGIN 列标头行 -->

<TABLE COLSPAN=8 CELLPADDING=5 BORDER=0><TR>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Company Name</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Contact Name</FONT>
</TD>
<TD ALIGN=CENTER WIDTH=150 BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Phone Number</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>City</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>State/Province</FONT>
</TD></TR>

<!-- 显示在记录集中 Customer Table Loop 的 ADO 数据,
每循环一次向 HTML 表添加一行 -->
<% Do While Not RsCustomerList.EOF %>
  <TR><TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RSCustomerList("CompanyName")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("ContactLastName") & ", " %>
  <%= RScustomerList("ContactFirstName") %>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
```

```
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("PhoneNumber")%>
 </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("City")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("StateOrProvince")%>
  </FONT></TD>
  </TR>
<!-Next Row = Record Loop 并添加至 html 表 -->
<%
RScustomerList.MoveNext
Loop
%>
</Table></Center></FONT>
<!-- 对命名的记录执行客户端 Input Data Validation  Move 并将其删除 -->

<Center>
<H4>Clicking Button Will Remove Designated Record</H4>
<H5>There are <%=RsCustomerList.RecordCount - 1%> Records in this Set</H5>
<Form Method = Post Action = "Delete.asp" Name = Form>
<Input Type = Text Name = "WhichRecord" Size = 3></Form>
<Input Type = Button Name = cmdDelete Value = "Delete Record"></Center>

</BODY>

<Script Language = "VBScript">

Sub cmdDelete_OnClick
If IsNumeric(Document.Form.WhichRecord.Value) Then
   Document.Form.WhichRecord.Value = CInt(Document.Form.WhichRecord.Value)
Dim Response
   Response = MsgBox("Are You Sure About Deleting This Record?", vbYesNo,  "ADO-
ASP

Example")

   If Response = vbYes Then

   Document.Form.Submit

   End If
```

```
Else
   MsgBox "You Must Enter a Valid Record Number",,"ADO-ASP Example"
End If
End Sub


</Script>
</HTML>
```

1.1.6.4.2.12 Execute、Requery 和 Clear 方法范例

该范例演示运行来自 **Command** 对象和 **Connection** 对象的 **Execute** 方法。同时使用 **Requery** 方法检索记录集中的当前数据，并用 **Clear** 方法清除 **Errors** 集合的内容。运行该过程需要 ExecuteCommand 和 PrintOutput 过程。

```
Public Sub ExecuteX()

   Dim strSQLChange As String
   Dim strSQLRestore As String
   Dim strCnn As String
   Dim cnn1 As ADODB.Connection
   Dim cmdChange As ADODB.Command
   Dim rstTitles As ADODB.Recordset
   Dim errLoop As ADODB.Error

   ' 定义两个 SQL 语句作为命令文本执行。
   strSQLChange = "UPDATE Titles SET Type = " & _
      "'self_help' WHERE Type = 'psychology'"
   strSQLRestore = "UPDATE Titles SET Type = " & _
      "'psychology' WHERE Type = 'self_help'"

   ' 打开连接。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set cnn1 = New ADODB.Connection
   cnn1.Open strCnn

   ' 创建命令对象。
   Set cmdChange = New ADODB.Command
   Set cmdChange.ActiveConnection = cnn1
   cmdChange.CommandText = strSQLChange

   ' 打开标题表。
   Set rstTitles = New ADODB.Recordset
   rstTitles.Open "titles", cnn1, , , adCmdTable

   ' 打印原始数据报告。
```

```
   Debug.Print _
      "Data in Titles table before executing the query"
   PrintOutput rstTitles

   ' 清除 Errors 集合的外部错误。
   cnn1.Errors.Clear

   ' 调用 ExecuteCommand 子例程执行 cmdChange 命令。
   ExecuteCommand cmdChange, rstTitles

   ' 打印新数据报告。
   Debug.Print _
      "Data in Titles table after executing the query"
   PrintOutput rstTitles

   ' 使用 Connection 对象的 execute 方法执行 SQL 语句以恢复数据。
   ' 捕获错误，必要时检查 Errors 集合。
   On Error GoTo Err_Execute
   cnn1.Execute strSQLRestore, , adExecuteNoRecords
   On Error GoTo 0

   ' 通过再查询记录集检索当前数据。
   rstTitles.Requery

   ' 打印已恢复数据的报告。
   Debug.Print "Data after executing the query " & _
      "to restore the original information"
   PrintOutput rstTitles

   rstTitles.Close
   cnn1.Close

   Exit Sub

Err_Execute:

   ' 将任何由执行查询引起的错误通知用户。
   If Errors.Count > 0 Then
      For Each errLoop In Errors
         MsgBox "Error number: " & errLoop.Number & vbCr & _
            errLoop.Description
      Next errLoop
   End If

   Resume Next
```

```
End Sub

Public Sub ExecuteCommand(cmdTemp As ADODB.Command, _
   rstTemp As ADODB.Recordset)

   Dim errLoop As Error

   ' 运行指定的 Command 对象。捕获错误，必要时检查 Errors 集合。
   On Error GoTo Err_Execute
   cmdTemp.Execute
   On Error GoTo 0

   ' 通过再查询记录集检索当前数据。
   rstTemp.Requery

   Exit Sub

Err_Execute:

   ' 将任何由执行查询引起的错误通知用户。
   If Errors.Count > 0 Then
      For Each errLoop In Errors
         MsgBox "Error number: " & errLoop.Number & vbCr & _
            errLoop.Description
      Next errLoop
   End If

   Resume Next

End Sub

Public Sub PrintOutput(rstTemp As ADODB.Recordset)

   ' 枚举 Recordset。
   Do While Not rstTemp.EOF
      Debug.Print "  " & rstTemp!Title & _
         ", " & rstTemp!Type
      rstTemp.MoveNext
   Loop

End Sub
```

VBScript 版本

下面是使用 VBScript 编写、并用于 Active Server Page (ASP) 的相同范例。如需查看该完整功能范例，请使用与 IIS 一同安装并位于 C:\InetPub\ASPSamp\AdvWorks 的数据源 AdvWorks.mdb，来创建名为 AdvWorks 的系统“数据源名称”(DSN)。这是 Microsoft Access 数据库文件。请使用查找命令定位文件 Adovbs.inc，并将其放入计划使用的目录中。请将以下代码剪切并粘贴到记事本或其他文本编辑器中，另存为“Execute.asp”。这样，便可在任何客户端浏览器中查看结果。

```
<!-- #Include file="ADOVBS.INC" -->
<HTML><HEAD>
<TITLE>ADO Execute Method</TITLE></HEAD>

<BODY>
<FONT FACE="MS SANS SERIF" SIZE=2>
<Center><H3>ADO Execute Method</H3><H4>Recordset Retrieved Using Connection Object</H4>
<TABLE WIDTH=600 BORDER=0>
<TD VALIGN=TOP ALIGN=LEFT COLSPAN=3><FONT SIZE=2>

<!--- Recordsets 使用 Connection 和 Command 对象的 Execute 方法进行检索 -->
<%
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "AdvWorks"
SQLQuery = "SELECT * FROM Customers"
' 第一个 Recordset RSCustomerList
Set RSCustomerList = OBJdbConnection.Execute(SQLQuery)

Set OBJdbCommand = Server.CreateObject("ADODB.Command")
OBJdbCommand.ActiveConnection = OBJdbConnection
SQLQuery2 = "SELECT * From Products"
OBJdbCommand.CommandText = SQLQuery2
Set RsProductList = OBJdbCommand.Execute

%>
<TABLE COLSPAN=8 CELLPADDING=5 BORDER=0>

<!-- Customer 表的 BEGIN 列标头行 -->

<TR><TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Company Name</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Contact Name</FONT>
</TD>
<TD ALIGN=CENTER WIDTH=150 BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>E-mail address</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>City</FONT>
```

```
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>State/Province</FONT>
</TD></TR>

<!-- 显示 Customer 表的 ADO 数据 -->
<% Do While Not RScustomerList.EOF %>
  <TR>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RSCustomerList("CompanyName")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("ContactLastName") & ", " %>
  <%= RScustomerList("ContactFirstName") %>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>

  <%= RScustomerList("ContactLastName")%>
 </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("City")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("StateOrProvince")%>
  </FONT></TD>
  </TR>
<!-Next Row = Record Loop 并添加到 html 表 -->
<%
RScustomerList.MoveNext
Loop
RScustomerList.Close

%>

</TABLE><HR>
<H4>Recordset Retrieved Using Command Object</H4>
<TABLE COLSPAN=8 CELLPADDING=5 BORDER=0>

<!-- Product List 表的 BEGIN 列标头行 -->
```

```
<TR><TD ALIGN=CENTER BGCOLOR="#800000">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Product Type</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#800000">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Product Name</FONT>
</TD>
<TD ALIGN=CENTER WIDTH=350 BGCOLOR="#800000">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Product Description</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#800000">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Unit Price</FONT>
</TD></TR>

<!-- 显示 ADO 数据的 Product List -->
<% Do While Not RsProductList.EOF %>
  <TR>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RsProductList("ProductType")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RsProductList("ProductName")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
   <%= RsProductList("ProductDescription")%>
 </FONT></TD>

  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RsProductList("UnitPrice")%>
  </FONT></TD>

<!--  Next Row = Record -->
<%
RsProductList.MoveNext
Loop
' 从内存删除对象以释放资源。
RsProductList.Close
OBJdbConnection.Close
Set ObJdbCommand = Nothing
Set RsProductList = Nothing
Set OBJdbConnection = Nothing
%>
```

```
</TABLE></FONT></Center></BODY></HTML>
```

1.1.6.4.2.13 GetRows 方法范例

该范例使用 **GetRows** 方法从 **Recordset** 中检索指定数目的行，并将结果数据填充到数组。在两种情况下 **GetRows** 方法返回的行将少于所需的数目：一种情况是因为达到了 **EOF**，另一种情况是因为 **GetRows** 试图检索已被其他用户删除的数据。仅当第二种情况发生时函数将返回 **False**。运行该过程需要使用 GetRowsOK 函数。

```
Public Sub GetRowsX()

   Dim rstEmployees As ADODB.Recordset
   Dim strCnn As String
   Dim strMessage As String
   Dim intRows As Integer
   Dim avarRecords As Variant
   Dim intRecord As Integer

   ' 使用雇员表中的姓名和受雇日期打开记录集。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstEmployees = New ADODB.Recordset
   rstEmployees.Open "SELECT fName, lName, hire_date " & _
      "FROM Employee ORDER BY lName", strCnn, , , adCmdText

   Do While True
      ' 得到用户输入的行数。
      strMessage = "Enter number of rows to retrieve."
      intRows = Val(InputBox(strMessage))

      If intRows <= 0 Then Exit Do

      ' 如 GetRowsOK 成功则打印结果，请注意是否达到文件末端。
      If GetRowsOK(rstEmployees, intRows, _
           avarRecords) Then
         If intRows > UBound(avarRecords, 2) + 1 Then
            Debug.Print "(Not enough records in " & _
               "Recordset to retrieve " & intRows & _
               " rows.)"
         End If
         Debug.Print UBound(avarRecords, 2) + 1 & _
            " records found."

         ' 打印已检索的数据。
         For intRecord = 0 To UBound(avarRecords, 2)
            Debug.Print "  " & _
               avarRecords(0, intRecord) & " " & _
```

```
            avarRecords(1, intRecord) & ", " & _
            avarRecords(2, intRecord)
        Next intRecord
      Else
        ' 假定 GetRows 错误源于其他用户对数据的更改，
        ' 使用 Requery 刷新 Recordset 并重新开始。
        If MsgBox("GetRows failed--retry?", _
            vbYesNo) = vbYes Then
          rstEmployees.Requery
        Else
          Debug.Print "GetRows failed!"
          Exit Do
        End If
      End If

      ' 由于使用 GetRows 使当前记录指针指向访问过的最后一个记录，
      ' 所以，在循环回到另一次搜索前将记录指针移回 Recordset 的开始。
      rstEmployees.MoveFirst
    Loop

    rstEmployees.Close

End Sub

Public Function GetRowsOK(rstTemp As ADODB.Recordset, _
    intNumber As Integer, avarData As Variant) As Boolean

    ' 将 GetRows 方法的结果保存在数组中。
    avarData = rstTemp.GetRows(intNumber)
    ' 仅当返回的行数少于所需的行数而非由于到达了 Recordset 末端时才返回 False。
    If intNumber > UBound(avarData, 2) + 1 And _
        Not rstTemp.EOF Then
      GetRowsOK = False
    Else
      GetRowsOK = True
    End If

End Function
```

1.1.6.4.2.14 Move 方法范例

该范例使用 Move 方法定位基于用户输入的记录指针。

```
Public Sub MoveX()

    Dim rstAuthors As ADODB.Recordset
```

```
Dim strCnn As String
Dim varBookmark As Variant
Dim strCommand As String
Dim lngMove As Long

' 打开 Authors 表的记录集。
   strCnn = "Provider=sqloledb;" & _
   "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
Set rstAuthors = New ADODB.Recordset
rstAuthors.CursorType = adOpenStatic
' 使用客户端游标以允许使用 AbsolutePosition 属性。
rstAuthors.CursorLocation = adUseClient
rstAuthors.Open "SELECT au_id, au_fname, au_lname, city, state " & _
   "FROM Authors ORDER BY au_lname", strCnn, , , adCmdText


rstAuthors.MoveFirst

Do While True
   ' 显示有关当前记录的信息并询问要移动的记录数。

   strCommand = InputBox( _
      "Record " & rstAuthors.AbsolutePosition & _
      " of " & rstAuthors.RecordCount & vbCr & _
      "Author: " & rstAuthors!au_fname & _
      " " & rstAuthors!au_lname & vbCr & _
      "Location: " & rstAuthors!City & _
      ", " & rstAuthors!State & vbCr & vbCr & _
      "Enter number of records to Move " & _
      "(positive or negative).")

   If strCommand = "" Then Exit Do

   ' 保存书签以防 Move 向前或向后移动太远。
   varBookmark = rstAuthors.Bookmark

   ' Move 方法需要数据类型为长整型的参数。
   lngMove = CLng(strCommand)
   rstAuthors.Move lngMove

   ' 捕获 BOF 或 EOF。
   If rstAuthors.BOF Then
      MsgBox "Too far backward! " & _
         "Returning to current record."
      rstAuthors.Bookmark = varBookmark
```

```
    End If
    If rstAuthors.EOF Then
       MsgBox "Too far forward! " & _
          "Returning to current record."
       rstAuthors.Bookmark = varBookmark
    End If
  Loop
  rstAuthors.Close

End Sub
```

VBScript 版本

下面是使用 VBScript 编写、并用于 Active Server Page (ASP) 的相同范例。如需查看该完整功能范例，请使用与 IIS 一同安装并位于 C:\InetPub\ASPSamp\AdvWorks 的数据源 AdvWorks.mdb，来创建名为 AdvWorks 的系统“数据源名称”(DSN)。这是 Microsoft Access 数据库文件。请使用查找命令定位文件 Adovbs.inc，并将其放入计划使用的目录中。请将以下代码剪切并粘贴到记事本或其他文本编辑器中，另存为“Move.asp”。这样，便可在任何客户端浏览器中查看结果。

请输入字母或非整数查看错误处理的运作。

```
<!-- #Include file="ADOVBS.INC" -->
<% Language = VBScript %>
<HTML><HEAD>
<TITLE>ADO Move Methods</TITLE></HEAD>
<BODY>
<FONT FACE="MS SANS SERIF" SIZE=2>
<Center>
<H3>ADO Move Methods</H3>

<%
 ' 创建并打开 Connection 对象。
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "AdvWorks"
' 创建并打开 Recordset 对象。
Set RsCustomerList = Server.CreateObject("ADODB.Recordset")
RsCustomerList.ActiveConnection = OBJdbConnection
RsCustomerList.CursorType = adOpenKeyset
RsCustomerList.LockType = adLockOptimistic
RsCustomerList.Source = "Customers"

RsCustomerList.Open

 ' 检查该会话中用户移动的数目，以窗体中的数量作为增量。
Session("Clicks") = Session("Clicks") + Request.Form("MoveAmount")
Clicks = Session("Clicks")
' 移动到上次已知的记录集位置加上由 Form Post 方法传递的数量。
RsCustomerList.Move CInt(Clicks)
```

```
' 出错处理。
   If RsCustomerList.EOF Then
        Session("Clicks") = RsCustomerList.RecordCount
        Response.Write "This is the Last Record"
        RsCustomerList.MoveLast
      Else If RsCustomerList.BOF Then
        Session("Clicks") = 1
        RsCustomerList.MoveFirst
        Response.Write "This is the First Record"
      End If
   End If

%>

<H3>Current Record Number is <BR>
<% If Session("Clicks") = 0 Then
Session("Clicks") = 1
End If
Response.Write(Session("Clicks") )%> of <%=RsCustomerList.RecordCount%></H3>
<HR>


<Center><TABLE COLSPAN=8 CELLPADDING=5 BORDER=0>

<!-- Customer 表的 BEGIN 列标头行 -->

<TR>

<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Company Name</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Contact Name</FONT>
</TD>
<TD ALIGN=CENTER WIDTH=150 BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Phone Number</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>City</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>State/Province</FONT>
</TD>
```

```
</TR>

<!-- 显示 Customer 表的 ADO 数据 -->

  <TR>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RSCustomerList("CompanyName")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("ContactLastName") & ", " %>
  <%= RScustomerList("ContactFirstName") %>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>

  <%= RScustomerList("PhoneNumber")%>
 </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("City")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("StateOrProvince")%>
  </FONT></TD>
  </TR> </Table></FONT>


<HR>
<Input Type = Button Name = cmdDown  Value = "<  ">
<Input Type = Button Name = cmdUp Value = " >">
<H5>Click Direction Arrows for Previous or Next Record
<BR> Click Move Amount to use Move Method
Enter Number of Records to Move + or - </H5>

<Table>

<Form Method = Post Action="Move.asp" Name=Form>

<TR><TD><Input Type="Button" Name = Move Value="Move Amount "></TD><TD></TD><TD>
<Input Type="Text" Size="4" Name="MoveAmount" Value = 0></TD><TR>
</Form></Table></Center>
```

```
</BODY>

<Script Language = "VBScript">

Sub Move_OnClick
' 确认输入的移动值为整型。
If IsNumeric(Document.Form.MoveAmount.Value)Then
   Document.Form.MoveAmount.Value = CInt(Document.Form.MoveAmount.Value)
   Document.Form.Submit
Else
   MsgBox "You Must Enter a Number", ,"ADO-ASP Example"
   Document.Form.MoveAmount.Value = 0
End If

End Sub

Sub cmdDown_OnClick

Document.Form.MoveAmount.Value = -1
Document.Form.Submit

End Sub

Sub cmdUp_OnClick

Document.Form.MoveAmount.Value = 1
Document.Form.Submit

End Sub

</Script>
</HTML>
```

1.1.6.4.2.15 MoveFirst 、 MoveLast 、 MoveNext 和 MovePrevious 方法范例

该范例使用 **MoveFirst**、**MoveLast**、**MoveNext** 以及 **MovePrevious** 方法，按照所提供的命令移动 **Recordset** 的记录指针。运行该过程需要 MoveAny 过程。

```
Public Sub MoveFirstX()

   Dim rstAuthors As ADODB.Recordset
   Dim strCnn As String
   Dim strMessage As String
   Dim intCommand As Integer
```

```
  ' 打开 Authors 表的记录集。
    strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  Set rstAuthors = New ADODB.Recordset
  rstAuthors.CursorType = adOpenStatic
  ' 使用客户端游标激活 AbsolutePosition 属性。
  rstAuthors.CursorLocation = adUseClient
  rstAuthors.Open "authors", strCnn, , , adCmdTable

  ' 显示当前记录信息并获得用户的方法选择。
  Do While True

    strMessage = "Name: " & rstAuthors!au_fName & " " & _
      rstAuthors!au_lName & vbCr & "Record " & _
      rstAuthors.AbsolutePosition & " of " & _
      rstAuthors.RecordCount & vbCr & vbCr & _
      "[1 - MoveFirst, 2 - MoveLast, " & vbCr & _
      "3 - MoveNext, 4 - MovePrevious]"
    intCommand = Val(Left(InputBox(strMessage), 1))
    If intCommand < 1 Or intCommand > 4 Then Exit Do

    ' 根据用户输入调用方法。
    MoveAny intCommand, rstAuthors
  Loop
  rstAuthors.Close

End Sub

Public Sub MoveAny(intChoice As Integer, _
  rstTemp As Recordset)

  ' 使用指定方法捕获 BOF 和 EOF。
  Select Case intChoice
    Case 1
      rstTemp.MoveFirst
    Case 2
      rstTemp.MoveLast
    Case 3
      rstTemp.MoveNext
      If rstTemp.EOF Then
        MsgBox "Already at end of recordset!"
        rstTemp.MoveLast
      End If
    Case 4
```

```
        rstTemp.MovePrevious
        If rstTemp.BOF Then
          MsgBox "Already at beginning of recordset!"
          rstTemp.MoveFirst
        End If
  End Select

End Sub
```

VBScript 版本

下面是使用 VBScript 编写、并用于 Active Server Page (ASP) 的相同范例。如需查看该完整功能范例，请使用与 IIS 一同安装并位于 C:\InetPub\ASPSamp\AdvWorks 的数据源 AdvWorks.mdb，来创建名为 AdvWorks 的系统“数据源名称”(DSN)。这是 Microsoft Access 数据库文件。请使用查找命令定位文件 Adovbs.inc，并将其放入计划使用的目录中。请将以下代码剪切并粘贴到记事本或其他文本编辑器中，另存为“MoveOne.asp”。这样，便可在任何客户端浏览器中查看结果。

可尝试在记录集的上限和下限之外移动以查看错误处理的运作。

```
<!-- #Include file="ADOVBS.INC" -->
<% Language = VBScript %>
<HTML><HEAD>
<TITLE>ADO MoveNext MovePrevious MoveLast MoveFirst Methods</TITLE></HEAD>
<BODY>
<FONT FACE="MS SANS SERIF" SIZE=2>
<Center>
<H3>ADO Methods<BR>MoveNext MovePrevious MoveLast MoveFirst</H3>
<!-- 在服务器上创建 Connection 和 Recordset 对象 -->
<%
 ' 创建并打开 Connection 对象。
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "AdvWorks"
' 创建并打开 Recordset 对象。
Set RsCustomerList = Server.CreateObject("ADODB.Recordset")
RsCustomerList.ActiveConnection = OBJdbConnection
RsCustomerList.CursorType = adOpenKeyset
RsCustomerList.LockType = adLockOptimistic
RsCustomerList.Source = "Customers"

RsCustomerList.Open

' 检查 Request.Form 集合以查看所记录的任何移动。

If Not IsEmpty(Request.Form("MoveAmount")) Then
' 跟踪该会话的移动数目和方向。

   Session("Moves") = Session("Moves") + Request.Form("MoveAmount")
```

```
   Clicks = Session("Moves")
' 移动到上一个已知位置。
   RsCustomerList.Move CInt(Clicks)
' 检查移动为 + 还是 - 并进行错误检查。
      If CInt(Request.Form("MoveAmount")) = 1 Then

         If RsCustomerList.EOF Then
            Session("Moves") = RsCustomerList.RecordCount
            RsCustomerList.MoveLast
         End If

         RsCustomerList.MoveNext
      End If

      If Request.Form("MoveAmount") < 1 Then

         RsCustomerList.MovePrevious
      End If
' 检查有无单击 First Record 或 Last Record 命令按钮。
      If Request.Form("MoveLast") = 3 Then
         RsCustomerList.MoveLast
         Session("Moves") = RsCustomerList.RecordCount
      End If
      If Request.Form("MoveFirst") = 2 Then
         RsCustomerList.MoveFirst
         Session("Moves") = 1
      End If


End If
' 对 Move Button 单击组合进行错误检查。
      If RsCustomerList.EOF Then
         Session("Moves") = RsCustomerList.RecordCount
         RsCustomerList.MoveLast
         Response.Write "This is the Last Record"
         End If

      If RsCustomerList.BOF Then
         Session("Moves") = 1
         RsCustomerList.MoveFirst
         Response.Write "This is the First Record"
      End If


%>
```

```
<H3>Current Record Number is <BR>
<!-- 显示当前记录数目和记录集大小 -->
<% If IsEmpty(Session("Moves"))  Then
Session("Moves") = 1
End If
%>

<%Response.Write(Session("Moves") )%> of <%=RsCustomerList.RecordCount%></H3>
<HR>


<Center><TABLE COLSPAN=8 CELLPADDING=5 BORDER=0>

<!-- Customer 表的 BEGIN 列标头行 -->

<TR><TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Company Name</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Contact Name</FONT>
</TD>
<TD ALIGN=CENTER WIDTH=150 BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Phone Number</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>City</FONT>
</TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>State/Province</FONT>
</TD></TR>

<!-- 显示 Customer 表的 ADO 数据 -->

  <TR>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RSCustomerList("CompanyName")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("ContactLastName") & ", " %>
  <%= RScustomerList("ContactFirstName") %>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
```

```
  <FONT STYLE="ARIAL NARROW" SIZE=1>

  <%= RScustomerList("PhoneNumber")%>
 </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("City")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("StateOrProvince")%>
  </FONT></TD>
  </TR> </Table></FONT>



<HR>
<Input Type = Button Name = cmdDown  Value = "<  ">
<Input Type = Button Name = cmdUp Value = "  >">
<BR>
<Input Type = Button Name = cmdFirst Value = "First Record">

<Input Type = Button Name = cmdLast Value = "Last Record">
<H5>Click Direction Arrows to Use MovePrevious or MoveNext
<BR> </H5>

<!-- 使用隐含窗体字段将值发送到服务器 -->

<Form Method = Post Action="MoveOne.asp" Name=Form>
<Input Type="Hidden" Size="4" Name="MoveAmount" Value = 0>
<Input Type="Hidden" Size="4" Name="MoveLast" Value = 0>
<Input Type="Hidden" Size="4" Name="MoveFirst" Value = 0>
</Form></BODY>

<Script Language = "VBScript">

Sub cmdDown_OnClick
' 在 Input Boxes 窗体和 Submit 窗体中设置值。
   Document.Form.MoveAmount.Value = -1
   Document.Form.Submit
End Sub

Sub cmdUp_OnClick

   Document.Form.MoveAmount.Value = 1
   Document.Form.Submit
```

```
End Sub

Sub cmdFirst_OnClick

   Document.Form.MoveFirst.Value = 2
   Document.Form.Submit

End Sub

Sub cmdLast_OnClick

   Document.Form.MoveLast.Value = 3
   Document.Form.Submit

End Sub
</Script></HTML>
```

1.1.6.4.2.16 NextRecordset 方法范例

```
Public Sub NextRecordsetX()

   Dim rstCompound As ADODB.Recordset
   Dim strCnn As String
   Dim intCount As Integer

   ' 打开复合记录集。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "

   Set rstCompound = New ADODB.Recordset
   rstCompound.Open "SELECT * FROM authors; " & _
      "SELECT * FROM stores; " & _
      "SELECT * FROM jobs", strCnn, , , adCmdText

   ' 显示每一个 SELECT 语句的结果。
   intCount = 1
   Do Until rstCompound Is Nothing
      Debug.Print "Contents of recordset #" & intCount
      Do While Not rstCompound.EOF
         Debug.Print , rstCompound.Fields(0), _
            rstCompound.Fields(1)
         rstCompound.MoveNext
      Loop
```

```
    Set rstCompound = rstCompound.NextRecordset
    intCount = intCount + 1
  Loop

End Sub
```

1.1.6.4.2.17 Open 和 Close 方法范例

该范例使用已经打开的 **Recordset** 和 **Connection** 对象的 **Open** 和 **Close** 方法。

```
Public Sub OpenX()

  Dim cnn1 As ADODB.Connection
  Dim rstEmployees As ADODB.Recordset
  Dim strCnn As String
  Dim varDate As Variant

  ' 打开连接。
    strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  Set cnn1 = New ADODB.Connection
  cnn1.Open strCnn

  ' 打开雇员表。
  Set rstEmployees = New ADODB.Recordset
  rstEmployees.CursorType = adOpenKeyset
  rstEmployees.LockType = adLockOptimistic
  rstEmployees.Open "employee", cnn1, , , adCmdTable

  ' 将第一个雇员记录的受雇日期赋值给变量，然后更改受雇日期。
  varDate = rstEmployees!hire_date
  Debug.Print "Original data"
  Debug.Print "  Name - Hire Date"
  Debug.Print "  " & rstEmployees!fName & " " & _
    rstEmployees!lName & " - " & rstEmployees!hire_date
  rstEmployees!hire_date = #1/1/1900#
  rstEmployees.Update
  Debug.Print "Changed data"
  Debug.Print "  Name - Hire Date"
  Debug.Print "  " & rstEmployees!fName & " " & _
    rstEmployees!lName & " - " & rstEmployees!hire_date

  ' 再查询 Recordset 并重置受雇日期。
  rstEmployees.Requery
  rstEmployees!hire_date = varDate
  rstEmployees.Update
```

```
   Debug.Print "Data after reset"
   Debug.Print "  Name - Hire Date"
   Debug.Print "  " & rstEmployees!fName & " " & _
      rstEmployees!lName & " - " & rstEmployees!hire_date

   rstEmployees.Close
   cnn1.Close

End Sub
```

VBScript 版本

下面是使用 VBScript 编写、并用于 Active Server Page (ASP) 的相同范例。如需查看该完整功能范例，请使用与 IIS 一同安装并位于 C:\InetPub\ASPSamp\AdvWorks 的数据源 AdvWorks.mdb，来创建名为 AdvWorks 的系统“数据源名称”(DSN)。这是 Microsoft Access 数据库文件。请使用查找命令定位文件 Adovbs.inc，并将其放入计划使用的目录中。请将以下代码剪切并粘贴到记事本或其他文本编辑器中，另存为“ADOOpen.asp”。这样，便可在任何客户端浏览器中查看结果。

```
<!-- #Include file="ADOVBS.INC" -->
<HTML><HEAD>
<TITLE>ADO Open Method</TITLE>
</HEAD><BODY>
<FONT FACE="MS SANS SERIF" SIZE=2>
<Center><H3>ADO Open Method</H3>
<TABLE WIDTH=600 BORDER=0>
<TD VALIGN=TOP ALIGN=LEFT COLSPAN=3><FONT SIZE=2>
<!--- 用于创建 2 个记录集的 ADO 连接 -->
<%
Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
OBJdbConnection.Open "AdvWorks"
SQLQuery = "SELECT * FROM Customers"
' 第一个记录集 RSCustomerList
Set RSCustomerList = OBJdbConnection.Execute(SQLQuery)
' 第二个记录集 RsProductist
Set RsProductList = Server.CreateObject("ADODB.Recordset")
RsProductList.CursorType = adOpenDynamic
RsProductList.LockType = adLockOptimistic
RsProductList.Open "Products", OBJdbConnection
%>
<TABLE COLSPAN=8 CELLPADDING=5 BORDER=0>

<!-- Customer 表的 BEGIN 列标头行 -->

<TR><TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Company Name</FONT></TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Contact Name</FONT></TD>
```

```
<TD ALIGN=CENTER WIDTH=150 BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>E-mail address</FONT></TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>City</FONT></TD>
<TD ALIGN=CENTER BGCOLOR="#008080">
<FONT             STYLE="ARIAL             NARROW"             COLOR="#ffffff"
SIZE=1>State/Province</FONT></TD></TR>

<!-- 显示 Customer 表的 ADO 数据 -->
<% Do While Not RScustomerList.EOF %>
  <TR><TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RSCustomerList("CompanyName")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("ContactLastName") & ", " %>
  <%= RScustomerList("ContactFirstName") %>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("ContactLastName")%>
 </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("City")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RScustomerList("StateOrProvince")%>
  </FONT></TD></TR>
<!-Next Row = Record Loop 并添加到 html 表 -->
<%
RScustomerList.MoveNext
Loop
RScustomerList.Close
OBJdbConnection.Close
%>
</TABLE>
<HR>
<TABLE COLSPAN=8 CELLPADDING=5 BORDER=0>

<!-- Product List 表的 BEGIN 列标头行 -->

<TR><TD ALIGN=CENTER BGCOLOR="#800000">
```

```
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Product Type</FONT></TD>
<TD ALIGN=CENTER BGCOLOR="#800000">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Product Name</FONT></TD>
<TD ALIGN=CENTER WIDTH=350 BGCOLOR="#800000">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Product
Description</FONT></TD>
<TD ALIGN=CENTER BGCOLOR="#800000">
<FONT STYLE="ARIAL NARROW" COLOR="#ffffff" SIZE=1>Unit Price</FONT></TD></TR>
<!-- 显示 ADO Data 的 Product List -->
<% Do While Not RsProductList.EOF %>
  <TR>  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RsProductList("ProductType")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RsProductList("ProductName")%>
  </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
   <%= RsProductList("ProductDescription")%>
 </FONT></TD>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT STYLE="ARIAL NARROW" SIZE=1>
  <%= RsProductList("UnitPrice")%>
  </FONT></TD>

<!--  Next Row = Record -->
<%
RsProductList.MoveNext
Loop
' 从 Memory Freeing 删除对象。
Set RsProductList = Nothing
Set OBJdbConnection = Nothing
%>
</TABLE></FONT></Center></BODY></HTML>
```

1.1.6.4.2.18 OpenSchema 方法范例

该范例使用 **OpenSchema** 方法显示 Pubs 数据库内每个表的名称和类型。

```
Public Sub OpenSchemaX()

   Dim cnn1 As ADODB.Connection
   Dim rstSchema As ADODB.Recordset
   Dim strCnn As String
```

```
   Set cnn1 = New ADODB.Connection
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   cnn1.Open strCnn

   Set rstSchema = cnn1.OpenSchema(adSchemaTables)

   Do Until rstSchema.EOF
      Debug.Print "Table name: " & _
         rstSchema!TABLE_NAME & vbCr & _
         "Table type: " & rstSchema!TABLE_TYPE & vbCr
      rstSchema.MoveNext
   Loop
   rstSchema.Close

   cnn1.Close

End Sub
```

该范例在 **OpenSchema** 方法的 **Criteria** 参数中指定 TABLE_TYPE 查询约束。因此，只返回在 Pubs 数据库中指定视图的模式信息。然后该范例显示每个表的名称和类型。

```
Public Sub OpenSchemaX2()

   Dim cnn2 As ADODB.Connection
   Dim rstSchema As ADODB.Recordset
   Dim strCnn As String

   Set cnn2 = New ADODB.Connection
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   cnn2.Open strCnn

   Set rstSchema = cnn2.OpenSchema(adSchemaTables, Array(Empty, Empty, Empty,
"VIEW"))

      Do Until rstSchema.EOF
         Debug.Print "Table name: " & _
            rstSchema!TABLE_NAME & vbCr & _
            "Table type: " & rstSchema!TABLE_TYPE & vbCr
         rstSchema.MoveNext
      Loop
   rstSchema.Close

   cnn2.Close
```

```
End Sub
```

1.1.6.4.2.19 Refresh 方法范例 (Visual Basic)

```
Public Sub RefreshX()

   Dim cnn1 As ADODB.Connection
   Dim cmdByRoyalty As ADODB.Command
   Dim rstByRoyalty As ADODB.Recordset
   Dim rstAuthors As ADODB.Recordset
   Dim intRoyalty As Integer
   Dim strAuthorID As String
   Dim strCnn As String

   ' 打开连接。
   Set cnn1 = New ADODB.Connection
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   cnn1.Open strCnn

   ' 使用一个参数打开已存储过程的命令对象。
   Set cmdByRoyalty = New ADODB.Command
   Set cmdByRoyalty.ActiveConnection = cnn1
   cmdByRoyalty.CommandText = "byroyalty"
   cmdByRoyalty.CommandType = adCmdStoredProc
   cmdByRoyalty.Parameters.Refresh

   ' 获取参数值并执行命令，将结果存储到记录集中。
   intRoyalty = Trim(InputBox("Enter royalty:"))
   cmdByRoyalty.Parameters(1) = intRoyalty
   Set rstByRoyalty = cmdByRoyalty.Execute()

   ' 打开 Authors 表获取作者姓名以用于显示。
   Set rstAuthors = New ADODB.Recordset
   rstAuthors.Open "authors", cnn1, , , adCmdTable

   ' 打印记录集中的当前数据，添加 Authors 表的作者姓名。
   Debug.Print "Authors with " & intRoyalty & " percent royalty"
   Do While Not rstByRoyalty.EOF
      strAuthorID = rstByRoyalty!au_id
      Debug.Print "   " & rstByRoyalty!au_id & ", ";
      rstAuthors.Filter = "au_id = '" & strAuthorID & "'"
      Debug.Print rstAuthors!au_fname & " " & _
         rstAuthors!au_lname
      rstByRoyalty.MoveNext
```

```
   Loop

   rstByRoyalty.Close
   rstAuthors.Close
   cnn1.Close

End Sub
```

1.1.6.4.2.20 Refresh 方法范例 (VBScript)

以下范例举例说明如何在运行时设置必要的 **RDS.DataControl** 参数。使用 **Refresh** 方法检索 **Recordset** 的方式取决于 **ExecuteOptions** 和 **FetchOptions** 属性的设置。如要测试该范例，请剪切该代码并粘贴至规范的 HTML 文本的 <Body></Body> 标记之间，并将其命名为“ADCapi2.asp”。ASP 脚本将标识服务器。

```
<HTML>
<HEAD>
<TITLE>Refresh / ExecuteOptions</TITLE>
</HEAD>
<BODY>
<Center><H2>RDS API Code Examples </H2>
<HR>
<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/She
ridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="Remote Data Service Run Time">
</OBJECT>

<!-- 在设计时设置的不带参数的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
   </OBJECT>
<HR>
<Input                          Size=70                          Name="txtServer"
Value="http://<%=Request.ServerVariables("SERVER_NAME")%>"><BR>
<Input           Size=70           Name="txtConnect"           Value           =
"dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;"><BR>
<Input Size=70 Name="txtSQL" Value="Select * from Employee">
<BR>
```

```
Choose if you want the Recordset brought back Synchronously on the current
calling thread or Asynchronously on another thread
<Input          Type="Radio"          Name="optExecuteOptions"          Checked
OnClick="SetExO('adcExecSync')">
<Input                                                                Type="Radio"
Name="optExecuteOptions"  OnClick="SetExO('adcExecAsync')"><BR>
Fetch Up Front, Background Fetch with Blocking or Background Fetch without
Blocking
<Input Type="Radio" Name="optFetchOptions"  OnClick="SetFO('adcFetchUpFront')">
<Input           Type="Radio"           Name="optFetchOptions"           Checked
OnClick="SetFO('adcFetchBackground')">
<Input Type="Radio" Name="optFetchOptions"  OnClick="SetFO('adcFetchAsync')">


<HR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run"><BR>
<H4>Fill Grid with these values or change them to see data from another ODBC
data source on your server</H4>
</Center>
<Script Language="VBScript">
<!--
Dim EO         'ExecuteOptions
Dim FO        'FetchOptions
EO = "adcExecSync"   ' 默认值
FO = "adcFetchBackground"   ' 默认值
Sub SetExO(NewEO)
  EO = NewEO
End Sub
Sub SetFO(NewFO)
  FO = NewFO
End Sub


' 在运行时设置 RDS.DataControl 参数。
Sub Run_OnClick
  ADC.Server = txtServer.Value
  ADC.SQL = txtSQL.Value
  ADC.Connect = txtConnect.Value
If EO = "adcExecSync" Then     ' 选择 ExecuteOption
  ADC.ExecuteOptions = adcExecSync
    MsgBox "Recordset brought in on current calling thread Syncronously"
Else
  ADC.ExecuteOptions = adcExecAsync
      MsgBox "Recordset brought in on another thread Asyncronously"
End If
```

```
If FO = "adcFetchBackground" Then      ' 选择 FetchOption
   ADC.FetchOptions = adcFetchBackground
      MsgBox "Control goes back to user after first batch of records returned"
ElseIf FO = " adcFetchUpFront" Then
   ADC.FetchOptions = adcFetchUpFront
        MsgBox "All records returned before control goes back to user"
Else
   ADC.FetchOptions = adcFetchAsync
        MsgBox "Control goes back to user immediately"
End If

ADC.Refresh

End Sub
-->
</Script>
</BODY>
</HTML>
```

1.1.6.4.2.21 Resync 方法范例

```
Public Sub ResyncX()

   Dim strCnn As String
   Dim rstTitles As ADODB.Recordset

   ' 打开连接。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "

   ' 打开标题表的记录集。
   Set rstTitles = New ADODB.Recordset
   rstTitles.CursorType = adOpenStatic
   rstTitles.LockType = adLockBatchOptimistic
   rstTitles.Open "titles", strCnn, , , adCmdTable

   ' 更改记录集中第一个标题的类型。
   rstTitles!Type = "database"

   ' 显示更改结果。
   MsgBox "Before resync: " & vbCr & vbCr & _
      "Title - " & rstTitles!Title & vbCr & _
      "Type - " & rstTitles!Type
```

```
   ' 再次与数据库同步并重新显示结果。
   rstTitles.Resync
   MsgBox "After resync: " & vbCr & vbCr & _
      "Title - " & rstTitles!Title & vbCr & _
      "Type - " & rstTitles!Type

   rstTitles.CancelBatch
   rstTitles.Close

End Sub
```

1.1.6.4.2.22 SubmitChanges 方法范例 (VBScript)

以下代码段说明如何使用 **RDS.DataControl** 对象的 **SubmitChanges** 方法。

如要测试该范例，请剪切该代码并粘贴至规范的 HTML 文本中的 <Body></Body> 标记之间，并将其命名为“ADCapi6.asp”。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples </H2>
<HR>
<H3>Remote Data Service SubmitChanges and CancelUpdate Methods</H3>
<!-- 在设计时设置的具有参数的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
   <PARAM NAME="SQL" VALUE="Select * from Employee for browse">
   <PARAM                                                    NAME="SERVER"
VALUE="http://<%=Request.ServerVariables("SERVER_NAME")%>">
   <PARAM NAME="CONNECT" VALUE="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;"> </OBJECT>

<Object classid ="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/She
ridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="Remote Data Service Run Time">
</OBJECT>
<HR>
<INPUT     TYPE=BUTTON     NAME="SubmitChange"     VALUE="Submit-Changes"><INPUT
TYPE=BUTTON NAME="CancelChange" VALUE="Cancel-Update"><BR>
<H4>Add a new person or alter a current entry on the grid. Move off that Row.
<BR>
```

```
Submit the Changes to your DBMS or cancel the updates. </H4>
</Center>
<Script Language="VBScript">
<!--
' 在“运行时”设置 RDS.DataControl 参数。
Sub SubmitChange_OnClick
   ADC.SubmitChanges
   ADC.Refresh
End Sub

Sub CancelUpdate_OnClick
   ADC.CancelUpdate
End Sub
-->
</Script>
```

以下代码段说明如何使用 **RDSServer.DataFactory** 对象的 **SubmitChanges** 方法。如果您正在进行某个不使用 **RDS.DataControl** 的 Visual Basic 项目，则可能需要向 **DRSServer.DataFactory** 提交更改。要创建该项目，请打开 Visual Basic 窗体并在其上放置列表框 (List1)、两个文本框（txtConnect 和 txtGetRecordset），以及两个命令按钮（cmdGetRecordset 和 cmdSubmitChanges）。

```
Dim ads As Object   ' RDS.DataSpace 对象
Dim adf As Object   ' RDSServer.DataFactory 对象

Private Sub UserDocument_Initialize()
   Call Form_Load
End Sub

Private Sub Form_Load()
   txtConnect.Text = "Dsn=pubs;Uid=sa;Pwd=;"
   txtGetRecordset.Text = "Select au_lname from authors"
End Sub

Private Sub cmdGetRecordset_Click()
   Dim Server As String
   Server = txtServer.Text

   ' 使用 RDS.DataSpace 的 CreateObject 方法创建 RDS.DataSpace。
   Set ads = CreateObject("RDS.DataSpace")
   Set adf = ads.CreateObject("RDSServer.DataFactory", Server)

   ' 使用 Recordset 充填 ListBox。
   MousePointer = vbHourglass
   Dim objADORs As Object
   Set objADORs = adf.Query(CStr(txtConnect.Text), CStr(txtGetRecordset.Text))
   List1.Clear
   While Not objADORs.EOF
      List1.AddItem objADORs(0).Value
```

```
      objADORs.MoveNext
   Wend
   MousePointer = vbNormal
End Sub
Sub cmdSubmitChanges_OnClick
   adf.SubmitChanges " Dsn=pubs;Uid=sa;Pwd=;",_
   objADORs
End Sub
```

1.1.6.4.2.23 Supports 方法范例

该范例使用 **Supports** 方法，显示用不同游标类型打开的记录集所支持的选项。运行该过程需要 DisplaySupport 过程。

```
Public Sub SupportsX()

   Dim aintCursorType(4) As Integer
   Dim rstTitles As ADODB.Recordset
   Dim strCnn As String
   Dim intIndex As Integer

   ' 打开连接。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "

   ' 使用 CursorType 常量填充数组。
   aintCursorType(0) = adOpenForwardOnly
   aintCursorType(1) = adOpenKeyset
   aintCursorType(2) = adOpenDynamic
   aintCursorType(3) = adOpenStatic

   ' 使用每个 CursorType 和优化锁定打开记录集，
   ' 然后调用 DisplaySupport 过程显示所支持的选项。
   For intIndex = 0 To 3
      Set rstTitles = New ADODB.Recordset
      rstTitles.CursorType = aintCursorType(intIndex)
      rstTitles.LockType = adLockOptimistic
      rstTitles.Open "titles", strCnn, , , adCmdTable

      Select Case aintCursorType(intIndex)
         Case adOpenForwardOnly
            Debug.Print "ForwardOnly cursor supports:"
         Case adOpenKeyset
            Debug.Print "Keyset cursor supports:"
         Case adOpenDynamic
            Debug.Print "Dynamic cursor supports:"
         Case adOpenStatic
```

```
            Debug.Print "Static cursor supports:"
        End Select

        DisplaySupport rstTitles
        rstTitles.Close
    Next intIndex

End Sub

Public Sub DisplaySupport(rstTemp As ADODB.Recordset)

    Dim alngConstants(11) As Long
    Dim booSupports As Boolean
    Dim intIndex As Integer

    ' 用游标选项常量填充数组。
    alngConstants(0) = adAddNew
    alngConstants(1) = adApproxPosition
    alngConstants(2) = adBookmark
    alngConstants(3) = adDelete
    alngConstants(4) = adFind
    alngConstants(5) = adHoldRecords
    alngConstants(6) = adMovePrevious
    alngConstants(7) = adNotify
    alngConstants(8) = adResync
    alngConstants(9) = adUpdate
    alngConstants(10) = adUpdateBatch

    For intIndex = 0 To 10
        booSupports = _
            rstTemp.Supports(alngConstants(intIndex))
        If booSupports Then
            Select Case alngConstants(intIndex)
                Case adAddNew
                    Debug.Print "   AddNew"
                Case adApproxPosition
                    Debug.Print "   AbsolutePosition and AbsolutePage"
                Case adBookmark
                    Debug.Print "   Bookmark"
                Case adDelete
                    Debug.Print "   Delete"
                Case adFind
                    Debug.Print "   Find"
                Case adHoldRecords
                    Debug.Print "   Holding Records"
```

```
            Case adMovePrevious
               Debug.Print "   MovePrevious and Move"
            Case adNotify
               Debug.Print "   Notifications"
            Case adResync
               Debug.Print "   Resyncing data"
            Case adUpdate
               Debug.Print "   Update"
            Case adUpdateBatch
               Debug.Print "   batch updating"
         End Select
      End If
   Next intIndex

End Sub
```

1.1.6.4.2.24 Update 和 CancelUpdate 方法范例

该范例连同 CancelUpdate 方法说明 Update 方法。

```
Public Sub UpdateX()

   Dim rstEmployees As ADODB.Recordset
   Dim strOldFirst As String
   Dim strOldLast As String
   Dim strMessage As String

   ' 使用雇员表中的姓名打开记录集。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstEmployees = New ADODB.Recordset
   rstEmployees.CursorType = adOpenKeyset
   rstEmployees.LockType = adLockOptimistic
   rstEmployees.Open "SELECT fname, lname " & _
      "FROM Employee ORDER BY lname", strCnn, , , adCmdText

   ' 储存原始数据。
   strOldFirst = rstEmployees!fname
   strOldLast = rstEmployees!lname
   ' 更改编辑缓冲区中的数据。
   rstEmployees!fname = "Linda"
   rstEmployees!lname = "Kobara"

   ' 显示缓冲区的内容并获取用户输入。
   strMessage = "Edit in progress:" & vbCr & _
      "  Original data = " & strOldFirst & " " & _
```

```
    strOldLast & vbCr & "  Data in buffer = " & _
    rstEmployees!fname & " " & rstEmployees!lname & vbCr & vbCr & _
    "Use Update to replace the original data with " & _
    "the buffered data in the Recordset?"

  If MsgBox(strMessage, vbYesNo) = vbYes Then
    rstEmployees.Update
  Else
    rstEmployees.CancelUpdate
  End If

  ' 显示结果数据。
  MsgBox "Data in recordset = " & rstEmployees!fname & " " & _
    rstEmployees!lname

  ' 恢复原始数据，因为这只是演示。
  If Not (strOldFirst = rstEmployees!fname And _
       strOldLast = rstEmployees!lname) Then
    rstEmployees!fname = strOldFirst
    rstEmployees!lname = strOldLast
    rstEmployees.Update
  End If

  rstEmployees.Close

End Sub
```

该范例连同 **AddNew** 方法说明 **Update** 方法。

```
Public Sub UpdateX2()

  Dim cnn1 As ADODB.Connection
  Dim rstEmployees As ADODB.Recordset
  Dim strEmpID As String
  Dim strOldFirst As String
  Dim strOldLast As String
  Dim strMessage As String

  ' 打开连接。
  Set cnn1 = New ADODB.Connection
    strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  cnn1.Open strCnn

  ' 使用雇员表中的数据打开记录集。
  Set rstEmployees = New ADODB.Recordset
  rstEmployees.CursorType = adOpenKeyset
```

```
   rstEmployees.LockType = adLockOptimistic
   rstEmployees.Open "employee", cnn1, , , adCmdTable

   rstEmployees.AddNew
   strEmpID = "B-S55555M"
   rstEmployees!emp_id = strEmpID
   rstEmployees!fname = "Bill"
   rstEmployees!lname = "Sornsin"

   ' 显示缓冲区内容并获取用户输入。
   strMessage = "AddNew in progress:" & vbCr & _
      "Data in buffer = " & rstEmployees!emp_id & ", " & _
      rstEmployees!fname & " " & rstEmployees!lname & vbCr & vbCr & _
      "Use Update to save buffer to recordset?"

   If MsgBox(strMessage, vbYesNoCancel) = vbYes Then
      rstEmployees.Update
      ' 转到新记录并显示结果数据。
      MsgBox "Data in recordset = " & rstEmployees!emp_id & ", " & _
          rstEmployees!fname & " " & rstEmployees!lname
   Else
      rstEmployees.CancelUpdate
      MsgBox "No new record added."
   End If

   ' 删除新数据，因为这只是演示。
   cnn1.Execute "DELETE FROM employee WHERE emp_id = '" & strEmpID & "'"

   rstEmployees.Close

End Sub
```

1.1.6.4.2.25 UpdateBatch 和 CancelBatch 方法范例

```
Public Sub UpdateBatchX()

   Dim rstTitles As ADODB.Recordset
   Dim strCnn As String
   Dim strTitle As String
   Dim strMessage As String

   ' 将连接字符串赋值给变量。
      strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
```

```
Set rstTitles = New ADODB.Recordset
rstTitles.CursorType = adOpenKeyset
rstTitles.LockType = adLockBatchOptimistic
rstTitles.Open "titles", strCnn, , , adCmdTable

rstTitles.MoveFirst

' 对记录集执行循环并询问用户是否要更改指定标题的类型。
Do Until rstTitles.EOF
   If Trim(rstTitles!Type) = "psychology" Then
      strTitle = rstTitles!Title
      strMessage = "Title: " & strTitle & vbCr & _
         "Change type to self help?"

      If MsgBox(strMessage, vbYesNo) = vbYes Then
         rstTitles!Type = "self_help"
      End If
   End If

   rstTitles.MoveNext
Loop

' 询问用户是否要提交以上所作的全部更改。
If MsgBox("Save all changes?", vbYesNo) = vbYes Then
   rstTitles.UpdateBatch
Else
   rstTitles.CancelBatch
End If

' 打印记录集中的当前数据。
rstTitles.Requery
rstTitles.MoveFirst
Do While Not rstTitles.EOF
   Debug.Print rstTitles!Title & " - " & rstTitles!Type
   rstTitles.MoveNext
Loop

' 恢复原始值，因为这只是演示。
rstTitles.MoveFirst
Do Until rstTitles.EOF
   If Trim(rstTitles!Type) = "self_help" Then
      rstTitles!Type = "psychology"
   End If
   rstTitles.MoveNext
Loop
```

```
   rstTitles.UpdateBatch

   rstTitles.Close

End Sub
```

1.1.6.4.3 ADO 属性范例

使用如下范例学习如何使用 ADO 属性。

注意 请从 Sub 到 End Sub 将代码范例整体粘贴到您的代码编辑器。如果使用部分范例或丢失段落格式，范例可能无法正确运行。

- AbsolutePage、PageCount 和 PageSize 属性范例(See 1.1.6.4.3.1)
- AbsolutePosition 和 CursorLocation 属性范例(See 1.1.6.4.3.2)
- ActiveConnection、CommandText、CommandTimeout、CommandType、Size 和 Direction 属性范例(See 1.1.6.4.3.3)
- ActualSize 和 DefinedSize 属性范例(See 1.1.6.4.3.4)
- Attributes 和 Name 属性范例(See 1.1.6.4.3.5)
- BOF、EOF 和 Bookmark 属性范例(See 1.1.6.4.3.6)
- CacheSize 属性范例(See 1.1.6.4.3.7)
- Connect 属性范例(See 1.1.6.4.3.8)
- ConnectionString、ConnectionTimeout 和 State 属性范例(See 1.1.6.4.3.9)
- Count 属性范例(See 1.1.6.4.3.10)
- CursorType、LockType 和 EditMode 属性范例(See 1.1.6.4.3.11)
- Description、NativeError、Number、Source 和 SQLState 属性范例(See 1.1.6.4.3.12)
- ExecuteOptions 和 FetchOptions 属性范例(See 1.1.6.4.3.13)
- Filter 和 RecordCount 属性范例(See 1.1.6.4.3.14)
- FilterColumn、FilterCriterion、FilterValue、SortColumn 和 SortDirection Properties 和 Reset 属性范例(See 1.1.6.4.3.15)
- IsolationLevel 和 Mode 属性范例(See 1.1.6.4.3.16)
- MarshalOptions 属性范例(See 1.1.6.4.3.17)

- MaxRecords 属性范例(See 1.1.6.4.3.18)
- NumericScale 和 Precision 属性范例(See 1.1.6.4.3.19)
- OriginalValue 和 UnderlyingValue 属性范例(See 1.1.6.4.3.20)
- Prepared 属性范例(See 1.1.6.4.3.21)
- Provider 和 DefaultDatabase 属性范例(See 1.1.6.4.3.22)
- Recordset 和 SourceRecordset 属性范例(See 1.1.6.4.3.23)
- ReadyState 属性范例(See 1.1.6.4.3.24)
- Server 属性范例(See 1.1.6.4.3.25)
- Source 属性范例(See 1.1.6.4.3.26)
- SQL 属性范例(See 1.1.6.4.3.27)
- State 属性范例(See 1.1.6.4.3.28)
- Status 属性范例(See 1.1.6.4.3.29)
- Type 属性范例(See 1.1.6.4.3.30)
- Type 属性范例(See 1.1.6.4.3.31)
- Version 属性范例(See 1.1.6.4.3.32)

1.1.6.4.3.1 AbsolutePage、PageCount 和 PageSize 属性范例

该范例使用 **AbsolutePage**、**PageCount** 和 **PageSize** 属性，以每次五个记录的方式显示雇员表中的姓名和受雇日期。

```
Public Sub AbsolutePageX()

  Dim rstEmployees As ADODB.Recordset
  Dim strCnn As String
  Dim strMessage As String
  Dim intPage As Integer
  Dim intPageCount As Integer
  Dim intRecord As Integer

  ' 使用客户端游标为雇员表打开一个记录集。
  strCnn = "Provider=sqloledb;" & _
     "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  Set rstEmployees = New ADODB.Recordset
  ' 使用客户端游标激活 AbsolutePosition 属性。
  rstEmployees.CursorLocation = adUseClient
```

```
   rstEmployees.Open "employee", strCnn, , , adCmdTable

   ' 显示姓名和受雇日期，每次五个记录。
   rstEmployees.PageSize = 5
   intPageCount = rstEmployees.PageCount
   For intPage = 1 To intPageCount
      rstEmployees.AbsolutePage = intPage
      strMessage = ""
      For intRecord = 1 To rstEmployees.PageSize
         strMessage = strMessage & _
            rstEmployees!fname & " " & _
            rstEmployees!lname & " " & _
            rstEmployees!hire_date & vbCr
         rstEmployees.MoveNext
         If rstEmployees.EOF Then Exit For
      Next intRecord
      MsgBox strMessage
   Next intPage
   rstEmployees.Close

End Sub
```

1.1.6.4.3.2 AbsolutePosition 和 CursorLocation 属性范例

该范例说明 **AbsolutePosition** 属性如何对枚举所有 **Recordset** 记录的循环进程进行跟踪。它通过将游标设置为客户端游标，使用 **CursorLocation** 属性激活 **AbsolutePosition** 属性。

```
Public Sub AbsolutePositionX()

   Dim rstEmployees As ADODB.Recordset
   Dim strCnn As String
   Dim strMessage As String

   ' 使用客户端游标为雇员表打开一个记录集。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstEmployees = New ADODB.Recordset
   ' 使用客户端游标激活 AbsolutePosition 属性。
   rstEmployees.CursorLocation = adUseClient
   rstEmployees.Open "employee", strCnn, , , adCmdTable

   ' 枚举记录集。
   Do While Not rstEmployees.EOF
      ' 显示当前记录信息。
      strMessage = "Employee: " & rstEmployees!lName & vbCr & _
         "(record " & rstEmployees.AbsolutePosition & _
```

```
      " of " & rstEmployees.RecordCount & ")"
    If MsgBox(strMessage, vbOKCancel) = vbCancel _
      Then Exit Do
    rstEmployees.MoveNext
  Loop

  rstEmployees.Close

End Sub
```

1.1.6.4.3.3 ActiveConnection 、CommandText 、CommandTimeout、CommandType、Size 和 Direction 属性范例

该范例使用 **ActiveConnection**、**CommandText**、**CommandTimeout**、**CommandType**、**Size** 和 **Direction** 属性执行存储过程。

```
Public Sub ActiveConnectionX()

  Dim cnn1 As ADODB.Connection
  Dim cmdByRoyalty As ADODB.Command
  Dim prmByRoyalty As ADODB.Parameter
  Dim rstByRoyalty As ADODB.Recordset
  Dim rstAuthors As ADODB.Recordset
  Dim intRoyalty As Integer
  Dim strAuthorID As String
  Dim strCnn As String

  ' 定义存储过程的命令对象。
  Set cnn1 = New ADODB.Connection
  strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  cnn1.Open strCnn
  Set cmdByRoyalty = New ADODB.Command
  Set cmdByRoyalty.ActiveConnection = cnn1
  cmdByRoyalty.CommandText = "byroyalty"
  cmdByRoyalty.CommandType = adCmdStoredProc
  cmdByRoyalty.CommandTimeout = 15

  ' 定义存储过程的输入参数。
  intRoyalty = Trim(InputBox( _
    "Enter royalty:"))
  Set prmByRoyalty = New ADODB.Parameter
  prmByRoyalty.Type = adInteger
  prmByRoyalty.Size = 3
```

```
   prmByRoyalty.Direction = adParamInput
   prmByRoyalty.Value = intRoyalty
   cmdByRoyalty.Parameters.Append prmByRoyalty

   ' 通过执行该命令创建记录集。
   Set rstByRoyalty = cmdByRoyalty.Execute()

   ' 打开作者表以便显示作者姓名。
   Set rstAuthors = New ADODB.Recordset
   rstAuthors.Open "authors", strCnn, , , adCmdTable

   ' 打印记录集中的当前数据，从作者表中添加作者姓名。
   Debug.Print "Authors with " & intRoyalty & _
      " percent royalty"
   Do While Not rstByRoyalty.EOF
      strAuthorID = rstByRoyalty!au_id
      Debug.Print , rstByRoyalty!au_id & ", ";
      rstAuthors.Filter = "au_id = '" & strAuthorID & "'"
      Debug.Print rstAuthors!au_fname & " " & _
         rstAuthors!au_lname
      rstByRoyalty.MoveNext
   Loop

   rstByRoyalty.Close
   rstAuthors.Close
   cnn1.Close

End Sub
```

1.1.6.4.3.4 ActualSize 和 DefinedSize 属性范例

```
Public Sub ActualSizeX()

   Dim rstStores As ADODB.Recordset
   Dim strCnn As String

   ' 打开存储表的记录集。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstStores = New ADODB.Recordset
   rstStores.Open "stores", strCnn, , , adCmdTable

   ' 在显示 stor_name 字段内容、字段定义
   ' 大小和实际大小的记录集中进行循环。
   rstStores.MoveFirst
```

```
  Do Until rstStores.EOF
    MsgBox "Store name: " & rstStores!stor_name & _
    vbCr & "Defined size: " & _
    rstStores!stor_name.DefinedSize & _
    vbCr & "Actual size: " & _
    rstStores!stor_name.ActualSize & vbCr
    rstStores.MoveNext
  Loop

  rstStores.Close

End Sub
```

1.1.6.4.3.5 Attributes 和 Name 属性范例

该范例显示 **Connection**、**Field** 和 **Property** 对象的 **Attributes** 属性值。它使用 **Name** 属性显示每个 **Field** 和 **Property** 对象的名称。

```
Public Sub AttributesX

  Dim cnn1 As ADODB.Connection
  Dim rstEmployees As ADODB.Recordset
  Dim fldLoop As ADODB.Field
  Dim proLoop As ADODB.Property
  Dim strCnn As String

  ' 打开连接和记录集。
  strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  Set cnn1 = New ADODB.Connection
  cnn1.Open strCnn
  Set rstEmployees = New ADODB.Recordset
  rstEmployees.Open "employee", cnn1, , , adCmdTable

  ' 显示连接的属性。
  Debug.Print "Connection attributes = " & _
    cnn1.Attributes

  ' 显示雇员表字段的属性。
  Debug.Print "Field attributes:"
  For Each fldLoop In rstEmployees.Fields
    Debug.Print "   " & fldLoop.Name & " = " & _
      fldLoop.Attributes
  Next fldLoop
```

```
  ' 显示是 NULLABLE 的雇员表的字段。
  Debug.Print "NULLABLE Fields:"
  For Each fldLoop In rstEmployees.Fields
    If CBool(fldLoop.Attributes And adFldIsNullable) Then
      Debug.Print "   " & fldLoop.Name
    End If
  Next fldLoop

  ' 显示雇员表属性的属性。
  Debug.Print "Property attributes:"
  For Each proLoop In rstEmployees.Properties
    Debug.Print "   " & proLoop.Name & " = " & _
      proLoop.Attributes
  Next proLoop

  rstEmployees.Close
  cnn1.Close

End Sub
```

1.1.6.4.3.6 BOF、EOF 和 Bookmark 属性范例

该范例使用 **BOF** 和 **EOF** 属性，在用户试图移过 **Recordset** 的第一个和最后一个记录时显示一条信息。它通过 **Bookmark** 属性使用户对 **Recordset** 中的记录进行标记，稍后再返回给它。

```
Public Sub BOFX()

  Dim rstPublishers As ADODB.Recordset
  Dim strCnn As String
  Dim strMessage As String
  Dim intCommand As Integer
  Dim varBookmark As Variant

  ' 使用来自出版商表的数据打开记录集。
  strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  Set rstPublishers = New ADODB.Recordset
  rstPublishers.CursorType = adOpenStatic
  ' 使用客户端游标启用 AbsolutePosition 属性。
  rstPublishers.CursorLocation = adUseClient
  rstPublishers.Open "SELECT pub_id, pub_name FROM publishers " & _
    "ORDER BY pub_name", strCnn, , , adCmdText

  rstPublishers.MoveFirst

  Do While True
```

```
    ' 显示关于当前记录的信息并让用户输入。
    strMessage = "Publisher: " & rstPublishers!pub_name & _
       vbCr & "(record " & rstPublishers.AbsolutePosition & _
       " of " & rstPublishers.RecordCount & ")" & vbCr & vbCr & _
       "Enter command:" & vbCr & _
       "[1 - next / 2 - previous /" & vbCr & _
       "3 - set bookmark / 4 - go to bookmark]"
    intCommand = Val(InputBox(strMessage))

    Select Case intCommand
       ' 向前或向后移动，捕获 BOF 或 EOF。
       Case 1
          rstPublishers.MoveNext
          If rstPublishers.EOF Then
             MsgBox "Moving past the last record." & _
                vbCr & "Try again."
             rstPublishers.MoveLast
          End If
       Case 2
          rstPublishers.MovePrevious
          If rstPublishers.BOF Then
             MsgBox "Moving past the first record." & _
                vbCr & "Try again."
             rstPublishers.MoveFirst
          End If

       ' 保存当前记录的书签。
       Case 3
          varBookmark = rstPublishers.Bookmark

       ' 转到由存储的书签所指示的记录。
       Case 4
          If IsEmpty(varBookmark) Then
             MsgBox "No Bookmark set!"
          Else
             rstPublishers.Bookmark = varBookmark
          End If

       Case Else
          Exit Do
    End Select

Loop

rstPublishers.Close
```

```
End Sub
```

该范例使用 **Bookmark** 和 **Filter** 属性创建记录集的限定视图，将只允许访问书签数组所引用的记录。

```
Public Sub BOFX2()

Dim rs As New ADODB.Recordset
Dim bmk(10)

rs.CursorLocation = adUseClient
rs.ActiveConnection = "Provider=sqloledb;" & _
     "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=;"

rs.Open "select * from authors", , adOpenStatic, adLockBatchOptimistic
Debug.Print "Number of records before filtering: ", rs.RecordCount

ii = 0
While rs.EOF <> True And ii < 11
    bmk(ii) = rs.Bookmark
    ii = ii + 1
    rs.Move 2
Wend
rs.Filter = bmk
Debug.Print "Number of records after filtering: ", rs.RecordCount

rs.MoveFirst
While rs.EOF <> True
    Debug.Print rs.AbsolutePosition, rs("au_lname")
    rs.MoveNext
Wend

End Sub
```

1.1.6.4.3.7 CacheSize 属性范例

```
Public Sub CacheSizeX()

  Dim rstRoySched As ADODB.Recordset
  Dim strCnn As String
  Dim sngStart As Single
  Dim sngEnd As Single
  Dim sngNoCache As Single
  Dim sngCache As Single
  Dim intLoop As Integer
  Dim strTemp As String
```

```
' 打开 RoySched 表。
strCnn = "Provider=sqloledb;" & _
   "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
Set rstRoySched = New ADODB.Recordset
rstRoySched.Open "roysched", strCnn, , , adCmdTable

' 枚举记录集对象两次并记录经历的时间。
sngStart = Timer

For intLoop = 1 To 2
   rstRoySched.MoveFirst

   Do While Not rstRoySched.EOF
      ' Execute a simple operation for the
      ' performance test.
      strTemp = rstRoySched!title_id
      rstRoySched.MoveNext
   Loop
Next intLoop

sngEnd = Timer
sngNoCache = sngEnd - sngStart

' 以 30 个记录为一组缓存记录。
rstRoySched.MoveFirst
rstRoySched.CacheSize = 30
sngStart = Timer

' 枚举记录集对象两次并记录经历的时间。
For intLoop = 1 To 2

   rstRoySched.MoveFirst
   Do While Not rstRoySched.EOF
      ' 执行一个简单操作，进行性能测试。
      strTemp = rstRoySched!title_id
      rstRoySched.MoveNext
   Loop
Next intLoop

sngEnd = Timer
sngCache = sngEnd - sngStart

' 显示性能结果。
MsgBox "Caching Performance Results:" & vbCr & _
   "   No cache: " & Format(sngNoCache, _
```

```
      "##0.000") & " seconds" & vbCr & _
      "   30-record cache: " & Format(sngCache, _
      "##0.000") & " seconds"
   rstRoySched.Close

End Sub
```

1.1.6.4.3.8 Connect 属性范例

该代码演示如何在设计时设置 **Connect** 属性：

```
<OBJECT CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID="ADC1">
.
.
.
   <PARAM NAME="SQL" VALUE="Select * from Sales">
   <PARAM NAME="Connect" VALUE="DSN=pubs;UID=sa;PWD=;">
   <PARAM NAME="Server" VALUE="http://MyWebServer">
.
.
.
</OBJECT>
```

以下范例演示如何以 VBScript 代码在运行时设置 **Connect** 属性。

要测试该范例，请剪切该代码并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为“ADCapi3.asp”。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples </H2>
<HR>
<H3>Set Connect Property at Run Time</H3>
<!-- 在设计时设置无参数 RDS.DataControl  -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
   </OBJECT>
<Object classid="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/She
ridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="Remote Data Service Connect Property at Run Time">
</OBJECT>
<HR>
```

```
<Input                          Size=70                          Name="txtServer"
Value="http://<%=Request.ServerVariables("SERVER_NAME")%>"><BR>
<Input Size=70 Name="txtConnect"><BR>
<Input Size=70 Name="txtSQL" Value="Select * from Employee">
<HR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run"><BR>
<H4>To Fill Grid enter Connect String in middle text box<BR>
Try dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;</H4>
</Center>
<Script Language="VBScript">
<!--
' 在运行时设置 RDS.DataControl 的参数。
Sub Run_OnClick
   ADC.Server = txtServer.Value
   ADC.SQL = txtSQL.Value
   ADC.Connect = txtConnect.Value
   ADC.Refresh
End Sub
-->
</Script>
```

1.1.6.4.3.9 ConnectionString、ConnectionTimeout 和 State 属性范例

该范例说明了使用 **ConnectionString** 属性打开 **Connection** 对象的不同方法。同时还使用 **ConnectionTimeout** 属性设置连接超时时间，并使用 **State** 属性检查连接的状态。该过程运行时需要 **GetState** 函数。

```
Public Sub ConnectionStringX()

   Dim cnn1 As ADODB.Connection
   Dim cnn2 As ADODB.Connection
   Dim cnn3 As ADODB.Connection
   Dim cnn4 As ADODB.Connection

   ' 不使用数据源名 (DSN) 打开连接。
   Set cnn1 = New ADODB.Connection
   cnn1.ConnectionString = "driver={SQL Server};" & _
      "server=bigsmile;uid=sa;pwd=pwd;database=pubs"
   cnn1.ConnectionTimeout = 30
   cnn1.Open

   ' 使用 DSN 和 ODBC 标记打开连接。
   Set cnn2 = New ADODB.Connection
   cnn2.ConnectionString = "DSN=Pubs;UID=sa;PWD=pwd;"
   cnn2.Open
```

```
   ' 使用 DSN 和 OLE DB 标记打开连接。
   Set cnn3 = New ADODB.Connection
   cnn3.ConnectionString = "Data Source=Pubs;User ID=sa;Password=pwd;"
   cnn3.Open

   ' 使用 DSN 和单个参数而非连接字符串打开连接。
   Set cnn4 = New ADODB.Connection
   cnn4.Open "Pubs", "sa", "pwd"

   ' 显示连接的状态。
   MsgBox "cnn1 state: " & GetState(cnn1.State) & vbCr & _
      "cnn2 state: " & GetState(cnn2.State) & vbCr & _
      "cnn3 state: " & GetState(cnn3.State) & vbCr & _
      "cnn4 state: " & GetState(cnn4.State)

   cnn4.Close
   cnn3.Close
   cnn2.Close
   cnn1.Close

End Sub

Public Function GetState(intState As Integer) As String

   Select Case intState
      Case adStateClosed
         GetState = "adStateClosed"
      Case adStateOpen
         GetState = "adStateOpen"
   End Select

End Function
```

1.1.6.4.3.10 Count 属性范例

该范例使用雇员数据库中的两个集合说明 **Count** 属性。该属性获得每个集合中的对象数并对枚举这些集合的循环设置上限。另一种不通过使用 **Count** 属性来枚举这些集合的方法是使用 For Each...Next 语句。

```
Public Sub CountX()

   Dim rstEmployees As ADODB.Recordset
   Dim strCnn As String
   Dim intloop As Integer

   ' 使用雇员表中的数据打开记录集。
```

```
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstEmployees = New ADODB.Recordset
   rstEmployees.Open "employee", strCnn, , , adCmdTable

   ' 打印有关字段集合的信息。
   Debug.Print rstEmployees.Fields.Count & _
      " Fields in Employee"
   For intloop = 0 To rstEmployees.Fields.Count - 1
      Debug.Print "   " & rstEmployees.Fields(intloop).Name
   Next intloop

   ' 打印属性集合的信息。
   Debug.Print rstEmployees.Properties.Count & _
      " Properties in Employee"
   For intloop = 0 To rstEmployees.Properties.Count - 1
      Debug.Print "   " & rstEmployees.Properties(intloop).Name
   Next intloop

   rstEmployees.Close

End Sub
```

1.1.6.4.3.11 CursorType、LockType 和 EditMode 属性范例

该范例说明如何在打开 **Recordset** 之前设置 **CursorType** 和 **LockType** 属性。同时还显示不同情况下 **EditMode** 的属性值。该过程运行时需要 **EditModeOutput** 函数。

```
Public Sub EditModeX()

   Dim cnn1 As ADODB.Connection
   Dim rstEmployees As ADODB.Recordset
   Dim strCnn As String

   ' 使用雇员表中的数据打开记录集。
   Set cnn1 = New ADODB.Connection
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   cnn1.Open strCnn

   Set rstEmployees = New ADODB.Recordset
   Set rstEmployees.ActiveConnection = cnn1
   rstEmployees.CursorType = adOpenKeyset
   rstEmployees.LockType = adLockBatchOptimistic
   rstEmployees.Open "employee", , , , adCmdTable
```

```
   ' 显示不同编辑状态下的 EditMode 属性。
   rstEmployees.AddNew
   rstEmployees!emp_id = "T-T55555M"
   rstEmployees!fname = "temp_fname"
   rstEmployees!lname = "temp_lname"
   EditModeOutput "After AddNew:", rstEmployees.EditMode
   rstEmployees.UpdateBatch
   EditModeOutput "After UpdateBatch:", rstEmployees.EditMode
   rstEmployees!fname = "test"
   EditModeOutput "After Edit:", rstEmployees.EditMode
   rstEmployees.Close

   ' 删除新记录，因为这只是演示。
   cnn1.Execute "DELETE FROM employee WHERE emp_id = 'T-T55555M'"

End Sub

Public Function EditModeOutput(strTemp As String, _
   intEditMode As Integer)

   ' 打印基于 EditMode 属性值的报表。
   Debug.Print strTemp
   Debug.Print "  EditMode = ";

   Select Case intEditMode
      Case adEditNone
         Debug.Print "adEditNone"
      Case adEditInProgress
         Debug.Print "adEditInProgress"
      Case adEditAdd
         Debug.Print "adEditAdd"
   End Select

End Function
```

1.1.6.4.3.12 Description、NativeError、Number、Source 和 SQLState 属性范例

该范例触发并捕获错误，同时显示产生的 **Error** 对象的 **Description**、**HelpContext**、**HelpFile**、**NativeError**、**Number**、**Source** 和 **SQLState** 属性。

```
Public Sub DescriptionX()

   Dim cnn1 As ADODB.Connection
   Dim errLoop As ADODB.Error
```

```
   Dim strError As String

   On Error GoTo ErrorHandler

   ' 有意触发错误。
   Set cnn1 = New ADODB.Connection
   cnn1.Open "nothing"

   Exit Sub

ErrorHandler:

   ' 枚举错误集合并显示每个 Error 对象的属性。
   For Each errLoop In cnn1.Errors
      strError = "Error #" & errLoop.Number & vbCr & _
         "   " & errLoop.Description & vbCr & _
         "   (Source: " & errLoop.Source & ")" & vbCr & _
         "   (SQL State: " & errLoop.SQLState & ")" & vbCr & _
         "   (NativeError: " & errLoop.NativeError & ")" & vbCr
      If errLoop.HelpFile = "" Then
         strError = strError & _
            "   No Help file available" & _
            vbCr & vbCr
      Else
         strError = strError & _
            "   (HelpFile: " & errLoop.HelpFile & ")" & vbCr & _
            "   (HelpContext: " & errLoop.HelpContext & ")" & _
            vbCr & vbCr
      End If


   Debug.Print strError
   Next

   Resume Next

End Sub
```

1.1.6.4.3.13 ExecuteOptions 和 FetchOptions 属性范例

以下代码演示如何在设计时设置 **ExecuteOptions** 和 **FetchOptions** 属性。如果不进行设置，**ExecuteOptions** 将默认为 **adcExecSync**。该设置表明在调用 **ADC.Refresh** 方法时，该方法将在当前调用的线程上执行（同步）。

```
<OBJECT CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID="ADC1">
.
.
```

```
.
<PARAM NAME="SQL" VALUE="Select * from Sales">
<PARAM NAME="Connect" VALUE="DSN=pubs;UID=sa;PWD=;">
<PARAM NAME="Server" VALUE="http://MyWebServer">
<PARAM NAME="ExecuteOptions" VALUE="1">
<PARAM NAME="FetchOptions" VALUE="3">


.
.
</OBJECT>
```

以下范例演示如何以 VBScript 代码在运行时设置 **ExecuteOptions** 和 **FetchOptions** 属性。请参阅这些属性工作范例的 **Refresh** 方法。

```
<Script Language="VBScript">
<!--
Sub ExecuteHow
' 对调用的线程异步执行下一个记录集刷新。
ADC1.ExecuteOptions = 1
ADC1.FetchOptions = 3
ADC.Refresh
End Sub
-->
</Script>
```

1.1.6.4.3.14 Filter 和 RecordCount 属性范例

该范例使用 **Filter** 属性打开一个新的 **Recordset**，它基于适用于已有 **Recordset** 的指定条件。它使用 **RecordCount** 属性显示两个 **Recordsets** 中的记录数。该过程运行时需要 FilterField 函数。

```
Public Sub FilterX()

   Dim rstPublishers As ADODB.Recordset
   Dim rstPublishersCountry As ADODB.Recordset
   Dim strCnn As String
   Dim intPublisherCount As Integer
   Dim strCountry As String
   Dim strMessage As String

   ' 使用出版商表中的数据打开记录集。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstPublishers = New ADODB.Recordset
   rstPublishers.CursorType = adOpenStatic
   rstPublishers.Open "publishers", strCnn, , , adCmdTable

   ' 充填记录集。
```

```
   intPublisherCount = rstPublishers.RecordCount

   ' 获得用户输入。
   strCountry = Trim(InputBox( _
      "Enter a country/region to filter on:"))

   If strCountry <> "" Then
      ' 打开已筛选的记录集对象。
      Set rstPublishersCountry = _
         FilterField(rstPublishers, "Country", strCountry)

      If rstPublishersCountry.RecordCount = 0 Then
         MsgBox "No publishers from that country/region."
      Else
         ' 打印原始记录集和已筛选记录集对象的记录数。
         strMessage = "Orders in original recordset: " & _
            vbCr & intPublisherCount & vbCr & _
            "Orders in filtered recordset (Country = '" & _
            strCountry & "'): " & vbCr & _
            rstPublishersCountry.RecordCount
         MsgBox strMessage
      End If
      rstPublishersCountry.Close

   End If

End Sub

Public Function FilterField(rstTemp As ADODB.Recordset, _
   strField As String, strFilter As String) As ADODB.Recordset

   ' 在指定的记录集对象上设置筛选操作并打开一个新的记录集对象。
   rstTemp.Filter = strField & " = '" & strFilter & "'"
   Set FilterField = rstTemp

End Function
```

注意　　当已知要选择的数据时，使用 SQL 语句打开 **Recordset** 通常更为有效。该范例说明了如何创建唯一的 **Recordset** 并从特定的国家（地区）获得记录。

```
Public Sub FilterX2()

   Dim rstPublishers As ADODB.Recordset
   Dim strCnn As String
```

```
   ' 使用出版商表中的数据打开记录集。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstPublishers = New ADODB.Recordset
   rstPublishers.CursorType = adOpenStatic
   rstPublishers.Open "SELECT * FROM publishers " & _
      "WHERE Country = 'USA'", strCnn, , , adCmdText

   ' 打印记录集中的当前数据。
   rstPublishers.MoveFirst
   Do While Not rstPublishers.EOF
      Debug.Print rstPublishers!pub_name & ", " & _
         rstPublishers!country
      rstPublishers.MoveNext
   Loop

   rstPublishers.Close

End Sub
```

1.1.6.4.3.15 FilterColumn、FilterCriterion、FilterValue、SortColumn 和 SortDirection 属性 和 Reset 方法范例

以下代码说明了如何在设计时设置 **RDS.DataControl Server** 参数并使用称为 Pubs 的 ODBC 数据源将其绑定在数据识别控件之上。Pubs 是随 SQL Server 6.5 带来的。要尝试该范例，需要名为 txtSortcolumn、txtSortdirection、txtFiltercolumn、txtCriterion 和 txtFilterValue 的超文本控件以及名为 SortFilter 的 HTML 表单输入按钮。

```
<OBJECT CLASSID="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
ID=ADC HEIGHT=10 WIDTH=10>
<PARAM NAME="SQL" VALUE="Select * from Sales ">
<PARAM NAME="SERVER" VALUE="http://MyWebServer">
<PARAM NAME="CONNECT" VALUE="dsn=pubs;UID=sa;PWD=;">
</OBJECT>

<!-- Sheridan Grid -->
<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"
   CODEBASE="http://MyWebServer
/MSADC/Samples/Sheridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
```

```
   <PARAM NAME="BackColor"   VALUE="-2147483643">
   <PARAM NAME="BackColorOdd"  VALUE="-2147483643">
   <PARAM NAME="ForeColorEven" VALUE="0">
</OBJECT>

<Script Language="VBScript">
<!--
Sub SortFilter_OnClick

   ' 设置数值。txtSortcolumn 是一个超文本框控件。
   ' SortColumn 的值将是用户在 txtSortcolumn 框中指定的文本值。
   If(txtSortcolumn.text <> "") then
      ADC.SortColumn = txtSortcolumn.text
   End If

   ' txtSortdirection 是一个超文本框控件。
   ' SortDirection 的值将是用户在 txtSortdirection 框中指定的文本值。
   Select Case UCASE(txtSortDirection.text)
   Case "TRUE"
      ADC.SortDirection = TRUE
   Case "FALSE"
      ADC.SortDirection = FALSE
   Case Else
      MsgBox "Only true or false are accepted for sort direction"
   End Select

   ' txtFiltercolumn 是一个超文本框控件。
   ' FilterColumn 的值将是用户在 txtFiltercolumn 框中指定的文本值。
   If (txtFiltercolumn.text <> "") Then
      ADC.FilterColumn = txtFiltercolumn.text
   End If

   ' txtCriterion 是一个超文本框控件。
   ' FilterCriterion 的值将是用户在 txtCriterion 框中指定的文本值。
   If (txtCriterion.text <> "") Then
      ADC.FilterCriterion = txtCriterion.text
   End If

   ' txtFilterValue 是一个超文本框控件。
   ' FilterValue 的值将是用户在 txtFilterValue 框中指定的文本值。
   If (txtFilterValue.text <> "") Then
      ADC.FilterValue = txtFilterValue.text
   End If
```

```
   ' 在基于指定排序和筛选属性的客户端记录集上执行排序和筛选。
   ' 调用重新设置将显示在数据绑定控件中的结果设置集刷新以显示
   ' 已筛选和已排序的记录集。
   ADC.Reset

End Sub
-->
</Script>
```

1.1.6.4.3.16 IsolationLevel 和 Mode 属性范例

```
Public Sub IsolationLevelX()

   Dim cnn1 As ADODB.Connection
   Dim rstTitles As ADODB.Recordset
   Dim strCnn As String

   ' 将连接字符串赋给变量。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "

   ' 打开连接和标题表。
   Set cnn1 = New ADODB.Connection
   cnn1.Mode = adModeShareExclusive
   cnn1.IsolationLevel = adXactIsolated
   cnn1.Open strCnn

   Set rstTitles = New ADODB.Recordset
   rstTitles.CursorType = adOpenDynamic
   rstTitles.LockType = adLockPessimistic
   rstTitles.Open "titles", cnn1, , , adCmdTable

   cnn1.BeginTrans

   ' 显示连接模式。
   If cnn1.Mode = adModeShareExclusive Then
      MsgBox "Connection mode is exclusive."
   Else
      MsgBox "Connection mode is not exclusive."
   End If

   ' 显示独立级别。
   If cnn1.IsolationLevel = adXactIsolated Then
      MsgBox "Transaction is isolated."
   Else
```

```
    MsgBox "Transaction is not isolated."
  End If

  ' 更改 psychology 标题的类型。
  Do Until rstTitles.EOF
    If Trim(rstTitles!Type) = "psychology" Then
      rstTitles!Type = "self_help"
      rstTitles.Update
    End If
    rstTitles.MoveNext
  Loop

  ' 打印记录集中的当前数据。
  rstTitles.Requery
  Do While Not rstTitles.EOF
    Debug.Print rstTitles!Title & " - " & rstTitles!Type
    rstTitles.MoveNext
  Loop

  ' 恢复原始数据。
  cnn1.RollbackTrans
  rstTitles.Close

  cnn1.Close

End Sub
```

1.1.6.4.3.17 MarshalOptions 属性范例

```
Public Sub MarshalOptionsX()

  Dim rstEmployees As ADODB.Recordset
  Dim strCnn As String
  Dim strOldFirst As String
  Dim strOldLast As String
  Dim strMessage As String
  Dim strMarshalAll As String
  Dim strMarshalModified As String

  ' 使用雇员表的姓名打开记录集。
  strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  Set rstEmployees = New ADODB.Recordset
  rstEmployees.CursorType = adOpenKeyset
  rstEmployees.LockType = adLockOptimistic
```

```
rstEmployees.CursorLocation = adUseClient
rstEmployees.Open "SELECT fname, lname " & _
   "FROM Employee ORDER BY lname", strCnn, , , adCmdText

' 存储原始数据。
strOldFirst = rstEmployees!fname
strOldLast = rstEmployees!lname

' 更改编辑缓冲区中的数据。
rstEmployees!fname = "Linda"
rstEmployees!lname = "Kobara"

' 显示缓冲区中的内容并让用户输入。
strMessage = "Edit in progress:" & vbCr & _
   " Original data = " & strOldFirst & " " & _
   strOldLast & vbCr & " Data in buffer = " & _
   rstEmployees!fname & " " & rstEmployees!lname & vbCr & vbCr & _
   "Use Update to replace the original data with " & _
   "the buffered data in the Recordset?"
strMarshalAll = "Would you like to send all the rows " & _
          "in the recordset back to the server?"
strMarshalModified = "Would you like to send only " & _
          "modified rows back to the server?"

If MsgBox(strMessage, vbYesNo) = vbYes Then
   If MsgBox(strMarshalAll, vbYesNo) = vbYes Then
      rstEmployees.MarshalOptions = adMarshalAll
      rstEmployees.Update
   ElseIf MsgBox(strMarshalModified, vbYesNo) = vbYes Then
      rstEmployees.MarshalOptions = adMarshalModifiedOnly
      rstEmployees.Update
   End If
End If

' 显示结果数据。
MsgBox "Data in recordset = " & rstEmployees!fname & " " & _
   rstEmployees!lname

' 恢复原始数据，因为这只是演示。
If Not (strOldFirst = rstEmployees!fname And _
      strOldLast = rstEmployees!lname) Then
   rstEmployees!fname = strOldFirst
   rstEmployees!lname = strOldLast
   rstEmployees.Update
End If
```

```
   rstEmployees.Close

End Sub
```

1.1.6.4.3.18 MaxRecords 属性范例

```
Public Sub MaxRecordsX()

   Dim rstTemp As ADODB.Recordset
   Dim strCnn As String

   ' 打开包含标题表中 10 个最有价值标题的记录集。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstTemp = New ADODB.Recordset
   rstTemp.MaxRecords = 10
   rstTemp.Open "SELECT Title, Price FROM Titles " & _
      "ORDER BY Price DESC", strCnn, , , adCmdText

   ' 显示记录集内容。
   Debug.Print "Top Ten Titles by Price:"

   Do While Not rstTemp.EOF
      Debug.Print "  " & rstTemp!Title & " - " & rstTemp!Price
      rstTemp.MoveNext
   Loop
   rstTemp.Close

End Sub
```

1.1.6.4.3.19 NumericScale 和 Precision 属性范例

该范例使用 **NumericScale** 和 **Precision** 属性来显示在 Pubs 数据库 Discounts 表中字段的数值范围和精度。

```
Public Sub NumericScaleX()

   Dim rstDiscounts As ADODB.Recordset
   Dim fldTemp As ADODB.Field
   Dim strCnn As String

   ' 打开记录集。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstDiscounts = New ADODB.Recordset
   rstDiscounts.Open "discounts", strCnn, , , adCmdTable
```

```
  ' 显示数字和小整数字段的数值范围和精度。
  For Each fldTemp In rstDiscounts.Fields
     If fldTemp.Type = adNumeric _
        Or fldTemp.Type = adSmallInt Then
        MsgBox "Field: " & fldTemp.Name & vbCr & _
           "Numeric scale: " & _
              fldTemp.NumericScale & vbCr & _
           "Precision: " & fldTemp.Precision
     End If
  Next fldTemp

  rstDiscounts.Close

End Sub
```

1.1.6.4.3.20 OriginalValue 和 UnderlyingValue 属性范例

该范例通过当某记录的基本数据在 **Recordset** 批更新过程中被更改时显示有关信息，来说明 **OriginalValue** 和 **UnderlyingValue** 属性。

```
Public Sub OriginalValueX()

  Dim cnn1 As ADODB.Connection
  Dim rstTitles As ADODB.Recordset
  Dim fldType As ADODB.Field
  Dim strCnn As String

  ' 打开连接。
  Set cnn1 = New ADODB.Connection
  strCnn = "Provider=sqloledb;" & _
     "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
  cnn1.Open strCnn

  ' 打开批更新的记录集。
  Set rstTitles = New ADODB.Recordset
  Set rstTitles.ActiveConnection = cnn1
  rstTitles.CursorType = adOpenKeyset
  rstTitles.LockType = adLockBatchOptimistic
  rstTitles.Open "titles"

  ' 设置 Type 字段的字段对象变量。
  Set fldType = rstTitles!Type

  ' 更改 psychology 标题的类型。
  Do Until rstTitles.EOF
```

```
    If Trim(fldType) = "psychology" Then
      fldType = "self_help"
    End If
    rstTitles.MoveNext
  Loop

  ' 通过使用命令字符串更新数据的方法模拟另一用户所做的更改。
  cnn1.Execute "UPDATE titles SET type = 'sociology' " & _
    "WHERE type = 'psychology'"

  ' 检查更改。
  rstTitles.MoveFirst
  Do Until rstTitles.EOF
    If fldType.OriginalValue <> _
      fldType.UnderlyingValue Then

      MsgBox "Data has changed!" & vbCr & vbCr & _
        "  Title ID: " & rstTitles!title_id & vbCr & _
        "  Current value: " & fldType & vbCr & _
        "  Original value: " & _
        fldType.OriginalValue & vbCr & _
        "  Underlying value: " & _
        fldType.UnderlyingValue & vbCr
    End If
    rstTitles.MoveNext
  Loop

  ' 取消更新，因为这只是演示。
  rstTitles.CancelBatch
  rstTitles.Close

  ' 恢复原始值。
  cnn1.Execute "UPDATE titles SET type = 'psychology' " & _
    "WHERE type = 'sociology'"

  cnn1.Close

End Sub
```

1.1.6.4.3.21 Prepared 属性范例

```
Public Sub PreparedX()

  Dim cnn1 As ADODB.Connection
  Dim cmd1 As ADODB.Command
```

```
Dim cmd2 As ADODB.Command
Dim strCnn As String
Dim strCmd As String
Dim sngStart As Single
Dim sngEnd As Single
Dim sngNotPrepared As Single
Dim sngPrepared As Single
Dim intLoop As Integer

' 打开连接。
strCnn = "Provider=sqloledb;" & _
   "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
Set cnn1 = New ADODB.Connection
cnn1.Open strCnn

' 为相同命令（一个是准备好的，另一个是未准备好的）创建两个命令对象。
strCmd = "SELECT title, type FROM titles ORDER BY type"

Set cmd1 = New ADODB.Command
Set cmd1.ActiveConnection = cnn1
cmd1.CommandText = strCmd

Set cmd2 = New ADODB.Command
Set cmd2.ActiveConnection = cnn1
cmd2.CommandText = strCmd
cmd2.Prepared = True

' 设置一个计时器，而后执行未准备好的命令 20 次。
sngStart = Timer
For intLoop = 1 To 20
   cmd1.Execute
Next intLoop
sngEnd = Timer
sngNotPrepared = sngEnd - sngStart

' 重置计时器，而后执行准备好的命令 20 次。
sngStart = Timer
For intLoop = 1 To 20
   cmd2.Execute
Next intLoop
sngEnd = Timer
sngPrepared = sngEnd - sngStart

' 显示执行结果。
MsgBox "Performance Results:" & vbCr & _
```

```
      "   Not Prepared: " & Format(sngNotPrepared, _
      "##0.000") & " seconds" & vbCr & _
      "   Prepared: " & Format(sngPrepared, _
      "##0.000") & " seconds"

   cnn1.Close

End Sub
```

1.1.6.4.3.22 Provider 和 DefaultDatabase 属性范例

该范例通过打开三个使用不同提供者的 **Connection** 对象演示 **Provider** 属性。还使用 **DefaultDatabase** 属性设置 Microsoft ODBC 提供者的默认数据库。

```
Public Sub ProviderX()

   Dim cnn1 As ADODB.Connection
   Dim cnn2 As ADODB.Connection
Dim cnn3 As ADODB.Connection

   ' 使用 Microsoft ODBC 提供者打开连接。
   Set cnn1 = New ADODB.Connection
   cnn1.ConnectionString = "driver={SQL Server};" & _
      "server=bigsmile;uid=sa;pwd=pwd"
   cnn1.Open strCnn
   cnn1.DefaultDatabase = "pubs"

   ' 显示提供者。
   MsgBox "Cnn1 provider: " & cnn1.Provider

   ' 使用 Microsoft Jet 提供者打开连接。
   Set cnn2 = New ADODB.Connection
   cnn2.Provider = "Microsoft.Jet.OLEDB.3.51"
   cnn2.Open "C:\Samples\northwind.mdb", "admin", ""

   ' 显示提供者。
   MsgBox "Cnn2 provider: " & cnn2.Provider

   ' 使用 Microsoft SQL 服务器提供者打开连接。
   Set cnn3 = New ADODB.Connection
   cnn3.Provider = "sqloledb"
   cnn3.Open "Data Source=srv;Initial Catalog=pubs;", "sa", ""

   ' 显示提供者。
   MsgBox "Cnn3 provider: " & cnn3.Provider
```

```
   cnn1.Close
   cnn2.Close
   cnn3.Close

End Sub
```

1.1.6.4.3.23 Recordset 和 SourceRecordset 属性范例

该范例演示如何在运行时设置 **RDSServer.DataFactory** 默认业务对象所必需的参数。要测试该范例，请剪切代码并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为“ADCapi4.asp”。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples</H2>
<HR>
<H3>Using SourceRecordset and Recordset with RDSServer.DataFactory</H3>
<!-- RDS.DataSpace ID ADS1 -->
<OBJECT ID="ADS1" WIDTH=1 HEIGHT=1
CLASSID="CLSID:BD96C556-65A3-11D0-983A-00C04FC29E36">
</OBJECT>

<!--参数在运行时设置的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
</OBJECT>

<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/She
ridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="RDSServer.DataFactory Run Time">
</OBJECT>
<HR>
<Input                          Size=70                          Name="txtServer"
Value="http://<%=Request.ServerVariables("SERVER_NAME")%>"><BR>
<Input                         Size=70                         Name="txtConnect"
Value="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;"><BR>
<Input Size=70 Name="txtSQL" Value="Select * from Employee">
<HR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run"><BR>
<H4>Fill Grid with these values or change them to see data from another ODBC
```

```
data source on your server.
<BR>Try dsn=pubs;uid=sa;pwd=; and Select * From Authors</H4>
</Center>
<Script Language="VBScript">
<!--
Dim ADF
Dim strServer
strServer = "http://<%=Request.ServerVariables("SERVER_NAME")%>"

Sub Run_OnClick()
   Dim objADORs        ' 创建 RDSServer.DataFactory 对象。
Set ADF = ADS1.CreateObject("RDSServer.DataFactory", strServer)                '
Get Recordset
Set objADORs = ADF.Query(txtConnect.Value,txtSQL.Value)

' 在运行时设置 RDS.DataControl 的参数。
   ADC.Server = txtServer.Value
   ADC.SQL = txtSQL.Value
   ADC.Connect = txtConnect.Value
   ADC.Refresh
End Sub
-->
</Script>
```

1.1.6.4.3.24 ReadyState 属性范例

以下范例说明如何在运行时以 VBScript 代码读取 **RDS.DataControl** 对象的 **ReadyState** 属性。**ReadyState** 为只读属性。

要测试该范例，请剪切该代码并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为“ADCapi9.asp”。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples </H2>
<HR>
<H3> RDS.DataControl ReadyState property</H3></Center>
<!-- 在运行时设置参数的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
   <PARAM NAME="SQL" VALUE="Select * from Employee for browse">
   <PARAM                                                    NAME="SERVER"
VALUE="http://<%=Request.ServerVariables("SERVER_NAME")%>">
   <PARAM NAME="CONNECT" VALUE="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
<PARAM NAME="ExecuteOptions" VALUE="adcExecAsync">
<PARAM NAME="FetchOptions" VALUE="adcFetchAsync">
</OBJECT>

<Script Language="VBScript">
```

```
Sub Window_OnLoad
Select Case ADC1.ReadyState
   case 2: MsgBox "Executing Query"
   case 3: MsgBox "Fetching records in background"
   case 4: MsgBox "All records fetched"
End Select
End Sub
</Script>
```

1.1.6.4.3.25 Server 属性范例

下列代码说明如何在设计时设置 **RDS.DataControl** 参数并使用称为 ADCDemo 的 ODBC 系统数据源将其绑定在数据识别控件上。依照指示，ADCDemo 将作为一个 SQL Server ODBC 系统数据源安装在服务器上。请剪切该代码并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为“ADCapi1.asp”。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples </H2>
<HR><BR>
<H3>Remote Data Service</H3>
<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/Sheridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
</OBJECT>
<!-- 参数在设计时设置的远程数据服务 -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
   <PARAM NAME="SQL" VALUE="Select * from Employee for browse">
   <PARAM                                                    NAME="SERVER"
VALUE="http://<%=Request.ServerVariables("SERVER_NAME")%>">
   <PARAM NAME="CONNECT" VALUE="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
</OBJECT><BR><HR></Center>
```

以下的范例说明如何在运行时设置 **RDS.DataControl** 所必需的参数。要测试该范例，请剪切该代码并粘贴到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为 “ADCapi4.asp”。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples </H2>
<HR><H3>Remote Data Service Server Property Set at Run Time</H3>
<!-- 在设计时没有设置参数的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
</OBJECT>
```

```
<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/She
ridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="Remote Data Service Run Time">
</OBJECT><HR>
<Input Size=70 Name="txtServer"><BR>
<Input Size=70 Name="txtConnect" Value="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
<BR>
<Input Size=70 Name="txtSQL" Value="Select * from Employee">
<HR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run"><BR>
<H4>Fill top text box with your Web server name in the form http://Myserver</H4>
</Center>
<Script Language="VBScript">
<!--
' 在运行时设置 RDS.DataControl 的参数。
Sub Run_OnClick
   ADC.Server = txtServer.Value
   ADC.SQL = txtSQL.Value
   ADC.Connect = txtConnect.Value
   ADC.Refresh
End Sub
-->
</Script>
```

1.1.6.4.3.26 Source 属性范例

```
Public Sub SourceX()

   Dim cnn1 As ADODB.Connection
   Dim rstTitles As ADODB.Recordset
   Dim rstPublishers As ADODB.Recordset
   Dim rstPublishersDirect As ADODB.Recordset
   Dim rstTitlesPublishers As ADODB.Recordset
   Dim cmdSQL As ADODB.Command
   Dim strCnn As String
   Dim strSQL As String
```

```
' 打开连接。
Set cnn1 = New ADODB.Connection
strCnn = "Provider=sqloledb;" & _
   "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
cnn1.Open strCnn

' 打开基于命令对象的记录集。
Set cmdSQL = New ADODB.Command
Set cmdSQL.ActiveConnection = cnn1
cmdSQL.CommandText = "Select title, type, pubdate " & _
   "FROM titles ORDER BY title"
Set rstTitles = cmdSQL.Execute()

' 打开基于表的记录集。
Set rstPublishers = New ADODB.Recordset
rstPublishers.Open "publishers", strCnn, , , adCmdTable

' 打开基于表的记录集。
Set rstPublishersDirect = New ADODB.Recordset
rstPublishersDirect.Open "publishers", strCnn, , , adCmdTableDirect


' 打开基于 SQL 字符串的记录集。
Set rstTitlesPublishers = New ADODB.Recordset
strSQL = "SELECT title_ID AS TitleID, title AS Title, " & _
   "publishers.pub_id AS PubID, pub_name AS PubName " & _
   "FROM publishers INNER JOIN titles " & _
   "ON publishers.pub_id = titles.pub_id " & _
   "ORDER BY Title"
rstTitlesPublishers.Open strSQL, strCnn, , , adCmdText

' 使用 Source 属性显示每个记录集的资源。
MsgBox "rstTitles source: " & vbCr & _
   rstTitles.Source & vbCr & vbCr & _
   "rstPublishers source: " & vbCr & _
   rstPublishers.Source & vbCr & vbCr & _
   "rstPublishersDirect source: " & vbCr & _
   rstPublishersDirect.Source & vbCr & vbCr & _
   "rstTitlesPublishers source: " & vbCr & _
   rstTitlesPublishers.Source

rstTitles.Close
rstPublishers.Close
rstTitlesPublishers.Close
```

```
   cnn1.Close

End Sub
```

1.1.6.4.3.27 SQL 属性范例

以下代码说明如何在设计时设置 **RDS.DataControl** 参数并使用随 SQL Server 6.5 带来的称为 Pubs 的 ODBC 数据源将其绑定在数据识别控件上。

```
<!—- RDS.DataControl -- >
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33" ID=ADC HEIGHT=10
WIDTH=10>
   <PARAM NAME="SQL" VALUE="Select * from Authors">
   <PARAM NAME="SERVER" VALUE="http://MyWebServer">
<PARAM NAME="CONNECT" VALUE="dsn=pubs;UID=sa;PWD=;">
</OBJECT>

<!-- 数据网格 -- >
<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"
   CODEBASE="http://MyWebServer
/MSADC/Samples/Sheridan.cab"
   ID=GRID1
      datasrc=#ADC
      HEIGHT=125
      WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="BackColor"   VALUE="-2147483643">
   <PARAM NAME="BackColorOdd"  VALUE="-2147483643">
   <PARAM NAME="ForeColorEven" VALUE="0">
</OBJECT>
```

以下的范例说明如何在运行时设置 RDS.DataControl 所必需的参数。要测试该范例，请剪切该代码到标准 HTML 文档的 <Body></Body> 标记之间，然后将其命名为 “ADCapi5.asp”。ASP 脚本将标识服务器。

```
<Center><H2>RDS API Code Examples </H2>
<HR><H3>Remote Data Service SQL Property Set at Run Time</H3>

<!-- 在设计时没有设置参数的 RDS.DataControl -->
<OBJECT classid="clsid:BD96C556-65A3-11D0-983A-00C04FC29E33"
   ID=ADC>
</OBJECT>
<Object CLASSID="clsid:AC05DC80-7DF1-11d0-839E-00A024A94B3A"

CODEBASE="http://<%=Request.ServerVariables("SERVER_NAME")%>/MSADC/Samples/She
ridan.cab"
   ID=GRID1
```

```
     datasrc=#ADC
     HEIGHT=125
     WIDTH=495>
   <PARAM NAME="AllowAddNew" VALUE="TRUE">
   <PARAM NAME="AllowDelete" VALUE="TRUE">
   <PARAM NAME="AllowUpdate" VALUE="TRUE">
   <PARAM NAME="Caption" VALUE="Remote Data Service Run Time">
</OBJECT>
<HR>
<Input               Size=70              Name="txtServer"               Value=
"http://<%=Request.ServerVariables("SERVER_NAME")%>">
<BR>
<Input Size=70 Name="txtConnect" Value="dsn=ADCDemo;UID=ADCDemo;PWD=ADCDemo;">
<BR>
<Input Size=70 Name="txtSQL">
<HR>
<INPUT TYPE=BUTTON NAME="Run" VALUE="Run"><BR>
<H4>Fill bottom text box with an SQL string<BR>
Try ... Select * from Employee</H4>
</Center>
<Script Language="VBScript">
<!--
' 在运行时设置 RDS. 的参数。
Sub Run_OnClick
   ADC.Server = txtServer.Value
   ADC.SQL = txtSQL.Value
   ADC.Connect = txtConnect.Value
   ADC.Refresh
End Sub
-->
</Script>
```

1.1.6.4.3.28 State 属性范例

```
Public Sub StateX()

   Dim cnn1 As ADODB.Connection
   Dim cnn2 As ADODB.Connection
   Dim cmdChange As ADODB.Command
   Dim cmdRestore As ADODB.Command
   Dim strCnn As String

   ' 打开两个异步连接，在连接时显示消息。
   Set cnn1 = New ADODB.Connection
   Set cnn2 = New ADODB.Connection
```

```
  strCnn = "Provider=sqloledb;" & _
    "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "

  cnn1.Open strCnn, , , adAsyncConnect
  While (cnn1.State = adStateConnecting)
    Debug.Print "Opening first connection...."
  Wend

  cnn2.Open strCnn, , , adAsyncConnect
  While (cnn2.State = adStateConnecting)
    Debug.Print "Opening second connection...."
  Wend

  ' 创建两个命令对象。
  Set cmdChange = New ADODB.Command
  cmdChange.ActiveConnection = cnn1
  cmdChange.CommandText = "UPDATE titles SET type = 'self_help' " & _
    "WHERE type = 'psychology'"

  Set cmdRestore = New ADODB.Command
  cmdRestore.ActiveConnection = cnn2
  cmdRestore.CommandText = "UPDATE titles SET type = 'psychology' " & _
    "WHERE type = 'self_help'"

  ' 执行命令，在正在执行时显示消息。
  cmdChange.Execute , , adAsyncExecute
  While (cmdChange.State = adStateExecuting)
    Debug.Print "Change command executing...."
  Wend

  cmdRestore.Execute , , adAsyncExecute
  While (cmdRestore.State = adStateExecuting)
    Debug.Print "Restore command executing...."
  Wend

  cnn1.Close
  cnn2.Close

End Sub
```

1.1.6.4.3.29 Status 属性范例

```
Public Sub StatusX()

  Dim rstTitles As ADODB.Recordset
```

```
   Dim strCnn As String

   ' 为批更新打开记录集。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstTitles = New ADODB.Recordset
   rstTitles.CursorType = adOpenKeyset
   rstTitles.LockType = adLockBatchOptimistic
   rstTitles.Open "titles", strCnn, , , adCmdTable

   ' 更改 psychology 标题的类型。
   Do Until rstTitles.EOF
      If Trim(rstTitles!Type) = "psychology" Then
         rstTitles!Type = "self_help"
      End If
      rstTitles.MoveNext
   Loop

   ' 显示标题 ID 和状态。
   rstTitles.MoveFirst
   Do Until rstTitles.EOF
      If rstTitles.Status = adRecModified Then
         Debug.Print rstTitles!title_id & " - Modified"
      Else
         Debug.Print rstTitles!title_id
      End If
      rstTitles.MoveNext
   Loop

   ' 取消更新，因为这只是演示。
   rstTitles.CancelBatch
   rstTitles.Close

End Sub
```

1.1.6.4.3.30 Type 属性范例

该范例通过显示与雇员表所有 **Field** 对象的 **Type** 属性值对应的常量名说明 **Type** 属性。该过程运行时需要 **FieldType** 函数。

```
Public Sub TypeX()

   Dim rstEmployees As ADODB.Recordset
   Dim fldLoop As ADODB.Field
   Dim strCnn As String

   ' 使用雇员表中的数据打开记录集。
```

```
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstEmployees = New ADODB.Recordset
   rstEmployees.Open "employee", strCnn, , , adCmdTable

   Debug.Print "Fields in Employee Table:" & vbCr

   ' 枚举雇员表的字段集合。
   For Each fldLoop In rstEmployees.Fields
      Debug.Print "  Name: " & fldLoop.Name & vbCr & _
         "  Type: " & FieldType(fldLoop.Type) & vbCr
   Next fldLoop

End Sub

Public Function FieldType(intType As Integer) As String

   Select Case intType
      Case adChar
         FieldType = "adChar"
      Case adVarChar
         FieldType = "adVarChar"
      Case adSmallInt
         FieldType = "adSmallInt"
      Case adUnsignedTinyInt
         FieldType = "adUnsignedTinyInt"
      Case adDBTimeStamp
         FieldType = "adDBTimeStamp"
   End Select

End Function
```

1.1.6.4.3.31 Value 属性范例

```
Public Sub ValueX()

   Dim rstEmployees As ADODB.Recordset
   Dim fldLoop As ADODB.Field
   Dim prpLoop As ADODB.Property
   Dim strCnn As String

   ' 使用雇员表的数据打开记录集。
   strCnn = "Provider=sqloledb;" & _
      "Data Source=srv;Initial Catalog=pubs;User Id=sa;Password=; "
   Set rstEmployees = New ADODB.Recordset
```

```
   rstEmployees.Open "employee", strCnn, , , adCmdTable

   Debug.Print "Field values in rstEmployees"
   ' 枚举雇员表的字段集合。
   For Each fldLoop In rstEmployees.Fields
      ' 由于值是字段对象的默认属性，
      ' 所以此处实际关键字的使用是可选的。
      Debug.Print "   " & fldLoop.Name & " = " & fldLoop.Value
   Next fldLoop

   Debug.Print "Property values in rstEmployees"
   ' 枚举记录集对象的属性集合。
   For Each prpLoop In rstEmployees.Properties
      ' 由于值是属性对象的默认属性，
      ' 所以此处实际关键字的使用是可选的。
      Debug.Print "   " & prpLoop.Name & " = " & prpLoop.Value
   Next prpLoop

   rstEmployees.Close

End Sub
```

1.1.6.4.3.32 Version 属性范例

该范例使用 **Connection** 对象的 **Version** 属性显示当前的 ADO 版本。同时还使用几个动态属性显示当前的 DBMS 名称和版本、OLE DB 的版本、提供者名称和版本、驱动程序名称和版本以及驱动程序 ODBC 的版本。

```
Public Sub VersionX()

   Dim cnn1 As ADODB.Connection

   ' 打开连接。
   Set cnn1 = New ADODB.Connection
   strCnn = “driver={SQL Server};server=srv;” & _
      “user id=sa;password=;database=pubs;”
   cnn1.Open strCnn

   strVersionInfo = "ADO Version: " & cnn1.Version & vbCr & _
   "DBMS Name: " & cnn1.Properties("DBMS Name") & vbCr & _
   "DBMS Version: " & cnn1.Properties("DBMS Version") & vbCr & _
   "OLE DB Version: " & cnn1.Properties("OLE DB Version") & vbCr & _
   "Provider Name: " & cnn1.Properties("Provider Name") & vbCr & _
   "Provider Version: " & cnn1.Properties("Provider Version") & vbCr & _
   "Driver Name: " & cnn1.Properties("Driver Name") & vbCr & _
   "Driver Version: " & cnn1.Properties("Driver Version") & vbCr & _
   "Driver ODBC Version: " & cnn1.Properties("Driver ODBC Version")
```

```
   MsgBox strVersionInfo

   cnn1.Close

End Sub
```

1.1.7 ADO 语法索引

使用这些索引参考在 Microsoft Visual J++ 或 Microsoft Visual C++ 环境中的 ADO 对象、方法、属性和事件主题。

- 语法索引 (ADO for VC++)(See 1.1.7.1)
- 语法索引 (ADO/WFC)(See 1.1.7.2)

1.1.7.1 语法索引 (ADO for VC++)

“ADO 语言参考”使用 Microsoft® Visual Basic® 编程语言来演示说明 ADO 方法和属性的语法。该索引是对基于 Microsoft® Visual C++® 的“ADO 语言参考”的交叉引用。

如果在应用程序中使用 **#import** 伪指令，将生成头文件以便用户能够使用与 Microsoft Visual Basic 类似的语法。可以将形式为 **get**_PropertyName 和 **put**_PropertyName 的属性名当作已被简单声明为 PropertyName，属性由此可被看作是数据成员而非函数。

所有方法、属性和事件都是返回 **HRESULT** 的函数，对 **HRESULT** 进行测试可确定函数是否执行成功。

下列元素的方法和属性的语法已被列出：

- 集合(See 1.1.7.1.7)
- Command 对象(See 1.1.7.1.2)
- Connection 对象(See 1.1.7.1.1)
- Error 对象(See 1.1.7.1.6)
- Field 对象(See 1.1.7.1.5)
- Parameter 对象(See 1.1.7.1.3)
- Recordset 对象(See 1.1.7.1.4)

1.1.7.1.1 _Connection (ADO for VC++ 语法)

方法

BeginTrans(See 1.1.4.4.4)(long **TransactionLevel*)

CommitTrans(See 1.1.4.4.4)(void)

RollbackTrans(See 1.1.4.4.4)(void)

Cancel(See 1.1.4.4.5)(void)

Close(See 1.1.4.4.12)(void)

Execute(See 1.1.4.4.22)(BSTR *CommandText*, VARIANT **RecordsAffected*, long *Options*,

_ADORecordset ***ppiRset*)

Open(See 1.1.4.4.32)(BSTR *ConnectionString*, BSTR *UserID*, BSTR *Password*, long *Options*)

OpenSchema(See 1.1.4.4.34)(SchemaEnum *Schema*, VARIANT *Restrictions*, VARIANT *SchemaID*,

_ADORecordset ***pprset*)

属性

get_Attributes(See 1.1.4.6.6)(long **plAttr*)

put_Attributes(long *lAttr*)

get_CommandTimeout(See 1.1.4.6.11)(LONG **plTimeout*)

put_CommandTimeout(LONG *lTimeout*)

get_ConnectionString(See 1.1.4.6.14)(BSTR **pbstr*)

put_ConnectionString(BSTR *bstr*)

get_ConnectionTimeout(See 1.1.4.6.15)(LONG **plTimeout*)

put_ConnectionTimeout(LONG *lTimeout*)

get_CursorLocation(See 1.1.4.6.17)(CursorLocationEnum **plCursorLoc*)

put_CursorLocation(CursorLocationEnum *lCursorLoc*)

get_DefaultDatabase(See 1.1.4.6.21)(BSTR **pbstr*)

put_DefaultDatabase(BSTR *bstr*)

get_IsolationLevel(See 1.1.4.6.36)(IsolationLevelEnum **Level*)

put_IsolationLevel(IsolationLevelEnum *Level*)

get_Mode(See 1.1.4.6.40)(ConnectModeEnum **plMode*)

put_Mode(ConnectModeEnum *lMode*)

get_Provider(See 1.1.4.6.51)(BSTR **pbstr*)

put_Provider(BSTR *Provider*)

get_State(See 1.1.4.6.64)(LONG **plObjState*)

get_Version(See 1.1.4.6.70)(BSTR **pbstr*)

get_Errors(See 1.1.4.3.1)(ADOErrors ***ppvObject*)

事件

BeginTransComplete(See 1.1.4.5.1)(LONG *TransactionLevel*, ADOError **pError*,

EventStatusEnum **adStatus*, _ADOConnection **pConnection*)

CommitTransComplete(See 1.1.4.5.1)(ADOError **pError*, EventStatusEnum **adStatus*,

_ADOConnection **pConnection*)

ConnectComplete(See 1.1.4.5.2)(ADOError **pError*, EventStatusEnum **adStatus*,

_ADOConnection **pConnection*)

Disconnect(See 1.1.4.5.2)(EventStatusEnum **adStatus*, _ADOConnection **pConnection*)

ExecuteComplete(See 1.1.4.5.4)(LONG *RecordsAffected*, ADOError **pError*,

EventStatusEnum **adStatus*, _ADOCommand **pCommand*,

_ADORecordset **pRecordset*, _ADOConnection **pConnection*)

InfoMessage(See 1.1.4.5.7)(ADOError **pError*, EventStatusEnum **adStatus*,

_ADOConnection **pConnection*)

RollbackTransComplete(See 1.1.4.5.1)(ADOError **pError*, EventStatusEnum **adStatus*,

_ADOConnection **pConnection*)

WillConnect(See 1.1.4.5.13)(BSTR **ConnectionString*, BSTR **UserID*, BSTR **Password*,

long *Options*, EventStatusEnum **adStatus*,

_ADOConnection **pConnection*)

WillExecute(See 1.1.4.5.14)(BSTR **Source*, CursorTypeEnum **CursorType*,

LockTypeEnum **LockType*, long **Options*,

EventStatusEnum **adStatus*, _ADOCommand **pCommand*,

_ADORecordset **pRecordset*, _ADOConnection **pConnection*)

1.1.7.1.2 _Command (ADO for VC++ 语法)

方法

Cancel(See 1.1.4.4.5)(void);

CreateParameter(See 1.1.4.4.16)(BSTR *Name*, DataTypeEnum *Type*,

ParameterDirectionEnum *Direction*, long *Size*, VARIANT *Value*,

_ADOParameter ***ppiprm*);

Execute(See 1.1.4.4.21)(VARIANT **RecordsAffected*, VARIANT **Parameters*, long *Options*,

_ADORecordset ***ppirs*);

属性

get_ActiveConnection(See 1.1.4.6.4)(_ADOConnection ***ppvObject*);

put_ActiveConnection(VARIANT *vConn*);

putref_ActiveConnection(_ADOConnection **pCon*);

get_CommandText(See 1.1.4.6.10)(BSTR **pbstr*);

put_CommandText(BSTR *bstr*);

get_CommandTimeout(See 1.1.4.6.11)(LONG **pl*);

put_CommandTimeout(LONG *Timeout*);

get_CommandType(See 1.1.4.6.12)(CommandTypeEnum **plCmdType*);

put_CommandType(CommandTypeEnum *lCmdType*);

get_Name(See 1.1.4.6.41)(BSTR **pbstrName*);

put_Name(BSTR *bstrName*);

get_Prepared(See 1.1.4.6.50)(VARIANT_BOOL **pfPrepared*);

put_Prepared(VARIANT_BOOL *fPrepared*);

get_State(See 1.1.4.6.64)(LONG **plObjState*);

get_Parameters(See 1.1.4.3.3)(ADOParameters ***ppvObject*);

1.1.7.1.3 _Parameter (ADO for VC++ 语法)

方法

AppendChunk(See 1.1.4.4.3)(VARIANT *Val*)

属性

get_Attributes(See 1.1.4.6.6)(LONG **plParmAttribs*)

put_Attributes(LONG *lParmAttribs*)

get_Direction(See 1.1.4.6.24)(ParameterDirectionEnum **plParmDirection*)

put_Direction(ParameterDirectionEnum *lParmDirection*)

get_Name(See 1.1.4.6.41)(BSTR **pbstr*)

put_Name(BSTR *bstr*)

get_NumericScale(See 1.1.4.6.44)(BYTE **pbScale*)

put_NumericScale(BYTE *bScale*)

get_Precision(See 1.1.4.6.49)(BYTE **pbPrecision*)

put_Precision(BYTE *bPrecision*)

get_Size(See 1.1.4.6.56)(long **pl*)

put_Size(long *l*)

get_Type(See 1.1.4.6.67)(DataTypeEnum **psDataType*)

put_Type(DataTypeEnum *sDataType*)

get_Value(See 1.1.4.6.69)(VARIANT **pvar*)

put_Value(VARIANT *val*)

1.1.7.1.4 _Recordset (ADO for VC++ 语法)

方法

AddNew(See 1.1.4.4.1)(VARIANT *FieldList*, VARIANT *Values*)

Cancel(See 1.1.4.4.5)(void)

CancelBatch(See 1.1.4.4.7)(AffectEnum *AffectRecords*)

CancelUpdate(See 1.1.4.4.8)(void)

Clone(See 1.1.4.4.11)(LockTypeEnum *LockType*, _ADORecordset ***ppvObject*)

Close(See 1.1.4.4.12)(void)

CompareBookmarks(See 1.1.4.4.13)(VARIANT *Bookmark1*, VARIANT *Bookmark2*,

CompareEnum **pCompare*)

Delete(See 1.1.4.4.20)(AffectEnum *AffectRecords*)

Find(See 1.1.4.4.23)(BSTR *Criteria*, LONG *SkipRecords*, SearchDirectionEnum *SearchDirection*,

VARIANT *Start*)

GetRows(See 1.1.4.4.25)(long *Rows*, VARIANT *Start*, VARIANT *Fields*, VARIANT **pvar*)

GetString(See 1.1.4.4.26)(StringFormatEnum *StringFormat*, long *NumRows*, BSTR *ColumnDelimeter*,

BSTR *RowDelimeter*, BSTR *NullExpr*, BSTR **pRetString*)

Move(See 1.1.4.4.28)(long *NumRecords*, VARIANT *Start*)

MoveFirst(See 1.1.4.4.29)(void)

MoveLast(See 1.1.4.4.29)(void)

MoveNext(See 1.1.4.4.29)(void)

MovePrevious(See 1.1.4.4.29)(void)

NextRecordset(See 1.1.4.4.31)(VARIANT **RecordsAffected*, _ADORecordset ***ppiRs*)

Open(See 1.1.4.4.33)(VARIANT *Source*, VARIANT *ActiveConnection*, CursorTypeEnum *CursorType*,

LockTypeEnum *LockType*, LONG *Options*)

Requery(See 1.1.4.4.38)(LONG *Options*)

Resync(See 1.1.4.4.40)(AffectEnum *AffectRecords*, ResyncEnum *ResyncValues*)

Save(See 1.1.4.4.41)(BSTR *FileName*, PersistFormatEnum *PersistFormat*)

Supports(See 1.1.4.4.44)(CursorOptionEnum *CursorOptions*, VARIANT_BOOL **pb*)

Update(See 1.1.4.4.45)(VARIANT *Fields*, VARIANT *Values*)

UpdateBatch(See 1.1.4.4.46)(AffectEnum *AffectRecords*)

属性

get_AbsolutePage(See 1.1.4.6.1)(PositionEnum *pl*)

put_AbsolutePage(PositionEnum *Page*)

get_AbsolutePosition(See 1.1.4.6.2)(PositionEnum *pl*)

put_AbsolutePosition(PositionEnum *Position*)

get_ActiveCommand(See 1.1.4.6.3)(IDispatch ***ppCmd*)

get_ActiveConnection(See 1.1.4.6.4)(VARIANT **pvar*)

put_ActiveConnection(VARIANT *vConn*)

putref_ActiveConnection(IDispatch **pconn*)

get_BOF(See 1.1.4.6.7)(VARIANT_BOOL **pb*)

get_Bookmark(See 1.1.4.6.8)(VARIANT **pvBookmark*)

put_Bookmark(VARIANT *vBookmark*)

get_CacheSize(See 1.1.4.6.9)(long **pl*)

put_CacheSize(long *CacheSize*)

get_CursorLocation(See 1.1.4.6.17)(CursorLocationEnum **plCursorLoc*)

put_CursorLocation(CursorLocationEnum *lCursorLoc*)

get_CursorType(See 1.1.4.6.18)(CursorTypeEnum **plCursorType*)

put_CursorType(CursorTypeEnum *lCursorType*)

get_DataMember(See 1.1.4.6.19)(BSTR **pbstrDataMember*)

put_DataMember(BSTR *bstrDataMember*)

get_DataSource(See 1.1.4.6.20)(IUnknown ***ppunkDataSource*)

putref_DataSource(IUnknown **punkDataSource*)

get_EditMode(See 1.1.4.6.25)(EditModeEnum **pl*)

get_EOF(See 1.1.4.6.7)(VARIANT_BOOL **pb*)

get_Filter(See 1.1.4.6.28)(VARIANT **Criteria*)

put_Filter(VARIANT *Criteria*)

get_LockType(See 1.1.4.6.37)(LockTypeEnum **plLockType*)

put_LockType(LockTypeEnum *lLockType*)

get_MarshalOptions(See 1.1.4.6.38)(MarshalOptionsEnum **peMarshal*)

put_MarshalOptions(MarshalOptionsEnum *eMarshal*)

get_MaxRecords(See 1.1.4.6.39)(long **plMaxRecords*)

put_MaxRecords(long *lMaxRecords*)

get_PageCount(See 1.1.4.6.47)(long **pl*)

get_PageSize(See 1.1.4.6.48)(long **pl*)

put_PageSize(long *PageSize*)

get_RecordCount(See 1.1.4.6.52)(long **pl*)

get_Sort(See 1.1.4.6.57)(BSTR **Criteria*)

put_Sort(BSTR *Criteria*)

get_Source(See 1.1.4.6.61)(VARIANT **pvSource*)

put_Source(BSTR *bstrConn*)

putref_Source(IDispatch **pcmd*)

get_State(See 1.1.4.6.64)(LONG **plObjState*)

get_Status(See 1.1.4.6.65)(long **pl*)

get_StayInSync(See 1.1.4.6.66)(VARIANT_BOOL **pbStayInSync*)

put_StayInSync(VARIANT_BOOL *bStayInSync*)

get_Fields(See 1.1.4.3.2)(ADOFields ***ppvObject*)

杂项

get_Collect(VARIANT *Index*, VARIANT **pvar*)

put_Collect(VARIANT *Index*, VARIANT *value*)

事件

EndOfRecordset(See 1.1.4.5.3)(VARIANT_BOOL **fMoreData*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

FetchComplete(See 1.1.4.5.5)(ADOError **pError*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

FetchProgress(See 1.1.4.5.6)(long *Progress*, long *MaxProgress*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

FieldChangeComplete(See 1.1.4.5.10)(LONG *cFields*, VARIANT *Fields*, ADOError **pError*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

MoveComplete(See 1.1.4.5.15)(EventReasonEnum *adReason*, ADOError **pError*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

RecordChangeComplete(See 1.1.4.5.11)(EventReasonEnum *adReason*, LONG *cRecords*, ADOError **pError*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

RecordsetChangeComplete(See 1.1.4.5.12)(EventReasonEnum *adReason*, ADOError **pError*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

WillChangeField(See 1.1.4.5.10)(LONG *cFields*, VARIANT *Fields*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

WillChangeRecord(See 1.1.4.5.11)(EventReasonEnum *adReason*, LONG *cRecords*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

WillChangeRecordset(See 1.1.4.5.12)(EventReasonEnum *adReason*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

WillMove(See 1.1.4.5.15)(EventReasonEnum *adReason*, EventStatusEnum **adStatus*, _ADORecordset **pRecordset*)

1.1.7.1.5 _Field (ADO for VC++ 语法)

方法

AppendChunk(See 1.1.4.4.3)(VARIANT *Data*)

GetChunk(See 1.1.4.4.24)(long *Length*, VARIANT **pvar*)

属性

get_ActualSize(See 1.1.4.6.5)(long **pl*)

get_Attributes(See 1.1.4.6.6)(long **pl*)

put_Attributes(long *lAttributes*)

get_DataFormat(IUnknown ***ppiDF*)

put_DataFormat(IUnknown **piDF*)

get_DefinedSize(See 1.1.4.6.22)(long **pl*)

put_DefinedSize(long *lSize*)

get_Name(See 1.1.4.6.41)(BSTR **pbstr*)

get_NumericScale(See 1.1.4.6.44)(BYTE **pbNumericScale*)

put_NumericScale(BYTE *bScale*)

get_OriginalValue(See 1.1.4.6.46)(VARIANT *pvar)

get_Precision(See 1.1.4.6.49)(BYTE *pbPrecision)

put_Precision(BYTE *bPrecision*)

get_Type(See 1.1.4.6.67)(DataTypeEnum *pDataType)

put_Type(DataTypeEnum *DataType*)

get_UnderlyingValue(See 1.1.4.6.68)(VARIANT *pvar)

get_Value(See 1.1.4.6.69)(VARIANT *pvar)

put_Value(VARIANT *Val*)

1.1.7.1.6 Error (ADO for VC++ 语法)

属性

get_Description(See 1.1.4.6.23)(BSTR *pbstr)

get_NativeError(See 1.1.4.6.42)(long *pl)

get_Number(See 1.1.4.6.43)(long *pl)

get_Source(See 1.1.4.6.60)(BSTR *pbstr)

get_SQLState(See 1.1.4.6.63)(BSTR *pbstr)

1.1.7.1.7 集合（ADO for VC++ 语法）

Parameters

方法

Append(See 1.1.4.4.2)(IDispatch *Object)

Delete(See 1.1.4.4.18)(VARIANT *Index*)

get_Item(See 1.1.4.4.27)(VARIANT *Index*, _ADOParameter **ppvObject)

Refresh(See 1.1.4.4.36)(void)

属性

get_Count(See 1.1.4.6.16)(long *c)

Fields

方法

Append(See 1.1.4.4.2)(BSTR *bstrName*, DataTypeEnum *Type*, long *DefinedSize*,

FieldAttributeEnum *Attrib*)

Delete(See 1.1.4.4.19)(VARIANT *Index*)

get_Item(See 1.1.4.4.27)(VARIANT *Index*, ADOField **ppvObject)

Refresh(See 1.1.4.4.36)(void)

属性

get_Count(See 1.1.4.6.16)(long *c)

Errors

方法

Clear(See 1.1.4.4.10)(void)

get_Item(See 1.1.4.4.27)(VARIANT *Index*, ADOError **ppvObject)

Refresh(See 1.1.4.4.36)(void)

属性

get_Count(See 1.1.4.6.16)(long *c*)

1.1.7.2 语法索引 (ADO/WFC)

“ADO 语言参考”使用 Microsoft® Visual Basic® 编程语言来说明 ADO 方法和属性的语法。该索引是对基于 Microsoft® Visual J++™ 和 ADO for Windows Foundation Classes (ADO/WFC) 的“ADO 语言参考”主题的交叉引用。当出现语法差异时，请使用该索引中的函数签名对照语言参考主题中的语法列表。

如下元素的方法和属性语法已被列出：

ActiveX 数据对象

- 集合(See 1.1.7.2.7)（Parameters、Fields、Errors）
- Command 对象(See 1.1.7.2.2)
- Connection 对象(See 1.1.7.2.1)
- Error 对象(See 1.1.7.2.6)
- Field 对象(See 1.1.7.2.5)
- Parameter 对象(See 1.1.7.2.3)
- Recordset 对象(See 1.1.7.2.4)

远程数据服务

- DataSpace(See 1.1.7.2.8)
- ObjectProxy(See 1.1.7.2.9)

1.1.7.2.1 Connection (ADO/WFC 语法)

包 com.ms.wfc.data

构造函数

public Connection(See 1.1.4.2.2)()

public **Connection**(String *connectionstring*)

方法

public int beginTrans(See 1.1.4.4.4)()

public void commitTrans(See 1.1.4.4.4)()

public void rollbackTrans(See 1.1.4.4.4)()

public void cancel(See 1.1.4.4.5)()

public void close(See 1.1.4.4.12)()

public com.ms.wfc.data.Recordset execute(See 1.1.4.4.22)(String *commandText*)

public com.ms.wfc.data.Recordset **execute**(String *commandText*, int *options*)

public int executeUpdate(See 1.1.4.4.22)(String *commandText*)

public int **executeUpdate**(String *commandText*, int *options*)

public void open(See 1.1.4.4.32)()

public void **open**(String *connectionString*)

public void **open**(String *connectionString*, String *userID*)

public void **open**(String *connectionString*, String *userID*, String *password*)

public void **open**(String *connectionString*, String *userID*, String *password*, int *options*)

public Recordset openSchema(See 1.1.4.4.34)(int schema, Object[] *restrictions*, String *schemaID*)

public Recordset **openSchema**(int schema)

public Recordset **openSchema**(int schema, Object[] *restrictions*)

属性

public int getAttributes(See 1.1.4.6.6)()

public void **setAttributes**(int *attr*)

public int getCommandTimeout(See 1.1.4.6.11)()

public void **setCommandTimeout**(int *timeout*)

public String getConnectionString(See 1.1.4.6.14)()

public void **setConnectionString**(String *con*)

public int getConnectionTimeout(See 1.1.4.6.15)()

public void **setConnectionTimeout**(int *timeout*)

public int getCursorLocation(See 1.1.4.6.17)()

public void **setCursorLocation**(int *cursorLoc*)

public String getDefaultDatabase(See 1.1.4.6.21)()

public void **setDefaultDatabase**(String *db*)

public int getIsolationLevel(See 1.1.4.6.36)()

public void **setIsolationLevel**(int *level*)

public int getMode(See 1.1.4.6.40)()

public void **setMode**(int *mode*)

public String getProvider(See 1.1.4.6.51)()

public void **setProvider**(String *provider*)

public int getState(See 1.1.4.6.64)()

public String getVersion(See 1.1.4.6.70)()

public AdoProperties getProperties(See 1.1.4.3.4)()

public com.ms.wfc.data.Errors getErrors(See 1.1.4.3.1)()

事件

有关 ADO/WFC 事件的详细信息，请参阅 ADO/WFC 中的 ADO 事件(See 1.1.3.5.5)。

public void addOnBeginTransComplete(See 1.1.4.5.1)(ConnectionEventHandler *handler*)

public void **removeOnBeginTransComplete**(ConnectionEventHandler *handler*)

public void addOnCommitTransComplete(See 1.1.4.5.1)(ConnectionEventHandler *handler*)

public void **removeOnCommitTransComplete**(ConnectionEventHandler *handler*)

public void addOnConnectComplete(See 1.1.4.5.2)(ConnectionEventHandler *handler*)

public void **removeOnConnectComplete**(ConnectionEventHandler *handler*)

public void addOnDisconnect(See 1.1.4.5.2)(ConnectionEventHandler *handler*)

public void **removeOnDisconnect**(ConnectionEventHandler *handler*)

public void addOnExecuteComplete(See 1.1.4.5.4)(ConnectionEventHandler *handler*)

public void **removeOnExecuteComplete**(ConnectionEventHandler *handler*)

public void addOnInfoMessage(See 1.1.4.5.7)(ConnectionEventHandler *handler*)

public void **removeOnInfoMessage**(ConnectionEventHandler *handler*)

public void addOnRollbackTransComplete(See 1.1.4.5.1)(ConnectionEventHandler *handler*)

public void **removeOnRollbackTransComplete**(ConnectionEventHandler *handler*)

public void addOnWillConnect(See 1.1.4.5.13)(ConnectionEventHandler *handler*)

public void **removeOnWillConnect**(ConnectionEventHandler *handler*)

public void addOnWillExecute(See 1.1.4.5.14)(ConnectionEventHandler *handler*)

public void **removeOnWillExecute**(ConnectionEventHandler *handler*)

1.1.7.2.2 Command (ADO/WFC 语法)

包 com.ms.wfc.data

构造函数

public Command(See 1.1.4.2.1)()

public **Command**(String *commandtext*)

方法

public void cancel(See 1.1.4.4.5)()

public com.ms.wfc.data.Parameter createParameter(See 1.1.4.4.16)(String *Name*, int *Type*, int *Direction*, int *Size*, Object *Value*)

public Recordset execute(See 1.1.4.4.21)()

public Recordset **execute**(Object[] *parameters*)

public Recordset **execute**(Object[] *parameters*, int *options*)

public int executeUpdate(See 1.1.4.4.22)(Object[] *parameters*)

public int **executeUpdate**(Object[] *parameters*, int *options*)

public int **executeUpdate**()

属性

public com.ms.wfc.data.Connection getActiveConnection(See 1.1.4.6.4)()

public void **setActiveConnection**(com.ms.wfc.data.Connection *con*)

public void **setActiveConnection**(String *conString*)

public String getCommandText(See 1.1.4.6.10)()

public void **setCommandText**(String *command*)

public int getCommandTimeout(See 1.1.4.6.11)()

public void **setCommandTimeout**(int *timeout*)

public int getCommandType(See 1.1.4.6.12)()

public void **setCommandType**(int *type*)

public String getName(See 1.1.4.6.41)()

public void **setName**(String *name*)

public boolean getPrepared(See 1.1.4.6.50)()

public void **setPrepared**(boolean *prepared*)

public int getState(See 1.1.4.6.64)()

public com.ms.wfc.data.Parameter getParameter(See 1.1.4.2.8)(int *n*)

public com.ms.wfc.data.Parameter **getParameter**(String *n*)

public com.ms.wfc.data.Parameters getParameters(See 1.1.4.3.3)()

public AdoProperties getProperties(See 1.1.4.3.4)()

1.1.7.2.3 Parameter (ADO/WFC 语法)

包 com.ms.wfc.data

构造函数

public Parameter(See 1.1.4.2.8)()

public **Parameter**(String *name*)

public **Parameter**(String *name*, int *type*)

public **Parameter**(String *name*, int *type*, int *dir*)

public **Parameter**(String *name*, int *type*, int *dir*, int *size*)

public **Parameter**(String *name*, int *type*, int *dir*, int *size*, Object *value*)

方法

public void appendChunk(See 1.1.4.4.3)(byte[] *bytes*)

public void **appendChunk**(char[] *chars*)

public void **appendChunk**(String *chars*)

属性

public int getAttributes(See 1.1.4.6.6)()

public void **setAttributes**(int *attr*)

public int getDirection(See 1.1.4.6.24)()

public void **setDirection**(int *dir*)

public String getName(See 1.1.4.6.41)()

public void **setName**(String *name*)

public int getNumericScale(See 1.1.4.6.44)()

public void **setNumericScale**(int scale)

public int getPrecision(See 1.1.4.6.49)()

public void **setPrecision**(int prec)

public int getSize(See 1.1.4.6.56)()

public void **setSize**(int *size*)

public int getType(See 1.1.4.6.67)()

public void **setType**(int *type*)

public com.ms.com.Variant getValue(See 1.1.4.6.69)()

public void **setValue**(Object *v*)

public AdoProperties getProperties(See 1.1.4.3.4)()

Parameter 存取方法

Parameter 对象的 **Value** 属性可获得或设置对象的内容。内容以变体型表示，变体型是可被赋以值和若干数据类型的对象类型。ADO/WFC 使用 **getValue** 方法和 **setValue** 方法实现 **Value** 属性，**getValue** 返回 VARIANT 对象，**setValue** 则把 VARIANT 当作参数使用。虽然在某些语言(如 Microsoft Visual Basic) 中 VARIANT 的效率已经很高。但仍可以在 Microsoft Visual J++ 中通过使用本地 Java 数据类型获得更高的性能。

除 **Value** 属性外，ADO/WFC 还提供使用 Java 数据类型获得并设置 **Parameter** 对象内容的存取方法。大多数这些方法都具有名称，其形式为 **Get**DataType 或 **Set**DataType。

有一点例外须加以注意，这就是不存在 **getNull** 属性。但存在 **isNull** 属性，该属性返回的布尔值可指明字段是否为空。

public boolean **getBoolean**()

public void **setBoolean**(boolean *v*)

public byte **getByte**()

public void **setByte**(byte *v*)

public double **getDouble**()

public void **setDouble**(double *v*)

public float **getFloat**()

public void **setFloat**(float *v*)

public int **getInt**()

public void **setInt**(int *v*)

public long **getLong**()

public void **setLong**(long *v*)

public short **getShort**()

public void **setShort**(short *v*)

public String **getString**()

public void **setString**(String *v*)

public boolean **isNull**()

public void **setNull**()

1.1.7.2.4 Recordset (ADO/WFC 语法)

包 com.ms.wfc.data

构造函数

public Recordset(See 1.1.4.2.10)()

public **Recordset**(Object *r*)

方法

public void addNew(See 1.1.4.4.1)(Object[] *fieldList*, Object[] *valueList*)

public void **addNew**(Object[] *valueList*)

public void **addNew**()

public void cancel(See 1.1.4.4.5)()

public void cancelBatch(See 1.1.4.4.7)(int *affectRecords*)

public void **cancelBatch**()

public void cancelUpdate(See 1.1.4.4.8)()

public Object clone(See 1.1.4.4.11)()

public Object **clone**(int *lockType*)

public void close(See 1.1.4.4.12)()

public int compareBookmarks(See 1.1.4.4.13)(Object *bookmark1*, Object *bookmark2*)

public void delete(See 1.1.4.4.20)(int *affectRecords*)

public void **delete**()

public void find(See 1.1.4.4.23)(String *criteria*)

public void **find**(String *criteria*, int *SkipRecords*)

public void **find**(String *criteria*, int *SkipRecords*, int *searchDirection*)

public void **find**(String *criteria*, int *SkipRecords*, int *searchDirection*, Object *bmkStart*)

public Object[][] getRows(See 1.1.4.4.25)(int *Rows*, Object *bmkStart*, Object[] *fieldList*)

public void move(See 1.1.4.4.28)(int *numRecords*)

public void **move**(int *numRecords*, Object *bmkStart*)

public void moveFirst(See 1.1.4.4.29)()

public void moveLast(See 1.1.4.4.29)()

public void moveNext(See 1.1.4.4.29)()

public void movePrevious(See 1.1.4.4.29)()

public Recordset nextRecordset(See 1.1.4.4.31)()

public Recordset nextRecordset(int[] *recordsAffected*)

public void open(See 1.1.4.4.33)()

public void **open**(Object *source*)

public void **open**(Object *source*, Object *activeConnection*)

public void **open**(Object *source*, Object *activeConnection*, int *cursorType*)

public void **open**(Object *source*, Object *activeConnection*, int *cursorType*,

int *lockType*)

public void **open**(Object *source*, Object *activeConnection*, int *cursorType*,

int *lockType*, int *options*)

public void requery(See 1.1.4.4.38)()

public void **requery**(int *options*)

public void resync(See 1.1.4.4.40)()

public void **resync**(int *affectRecords*, int *resyncValues*)

public void save(See 1.1.4.4.41)(String *fileName*)

public void **save**(String *fileName*, int *persistFormat*)

public boolean supports(See 1.1.4.4.44)(int *cursorOptions*)

public void update(See 1.1.4.4.45)()

public void **update**(Object[] *valueList*)

public void **update**(Object[] *fieldList*, Object[] *valueList*)

public void updateBatch(See 1.1.4.4.46)()

public void **updateBatch**(int *affectRecords*)

属性

public int getAbsolutePage(See 1.1.4.6.1)()

public void **setAbsolutePage**(int *page*)

public int getAbsolutePosition(See 1.1.4.6.2)()

public void **setAbsolutePosition**(int *pos*)

public Command getActiveCommand(See 1.1.4.6.3)()

public Connection getActiveConnection(See 1.1.4.6.4)()

public void **setActiveConnection**(String *conn*)

public void **setActiveConnection**(com.ms.wfc.data.Connection *c*)

public boolean getBOF(See 1.1.4.6.7)()

public boolean getEOF(See 1.1.4.6.7)()

public Object getBookmark(See 1.1.4.6.8)()

public void **setBookmark**(Object *bmk*)

public int getCacheSize(See 1.1.4.6.9)()

public void **setCacheSize**(int *size*)

public int getCursorLocation(See 1.1.4.6.17)()

public void **setCursorLocation**(int *cursorLoc*)

public int getCursorType(See 1.1.4.6.18)()

public void **setCursorType**(int *cursorType*)

public String getDataMember(See 1.1.4.6.19)()

public void **setDataMember**(String *pbstrDataMember*)

public Iunknown getDataSource(See 1.1.4.6.20)()

public void **setDataSource**(IUnknown *dataSource*)

public int getEditMode(See 1.1.4.6.25)()

public Object getFilter(See 1.1.4.6.28)()

public void **setFilter**(Object *filter*)

public int getLockType(See 1.1.4.6.37)()

public void **setLockType**(int *lockType*)

public int getMarshalOptions(See 1.1.4.6.38)()

public void **setMarshalOptions**(int *options*)

public int getMaxRecords(See 1.1.4.6.39)()

public void **setMaxRecords**(int *maxRecords*)

public int getPageCount(See 1.1.4.6.47)()

public int getPageSize(See 1.1.4.6.48)()

public void **setPageSize**(int *pageSize*)

public int getRecordCount(See 1.1.4.6.52)()

public String getSort(See 1.1.4.6.57)()

public void **setSort**(String *criteria*)

public String getSource(See 1.1.4.6.61)()

public void **setSource**(String *query*)

public void **setSource**(com.ms.wfc.data.Command *command*)

public int getState(See 1.1.4.6.64)()

public int getStatus(See 1.1.4.6.65)()

public boolean getStayInSync(See 1.1.4.6.66)()

public void **setStayInSync**(boolean *pbStayInSync*)

public com.ms.wfc.data.Field getField(See 1.1.4.2.7)(int *n*)

public com.ms.wfc.data.Field **getField**(String *n*)

public com.ms.wfc.data.Fields getFields(See 1.1.4.3.2)()

public AdoProperties getProperties(See 1.1.4.3.4)()

事件

有关 ADO/WFC 事件的详细信息，请参阅 ADO/WFC 中的 ADO 事件(See 1.1.3.5.5)。

public void addOnEndOfRecordset(See 1.1.4.5.3)(RecordsetEventHandler *handler*)

public void **removeOnEndOfRecordset**(RecordsetEventHandler *handler*)

public void addOnFetchComplete(See 1.1.4.5.5)(RecordsetEventHandler *handler*)

public void **removeOnFetchComplete**(RecordsetEventHandler *handler*)

public void addOnFetchProgress(See 1.1.4.5.6)(RecordsetEventHandler *handler*)

public void **removeOnFetchProgress**(RecordsetEventHandler *handler*)

public void addOnFieldChangeComplete(See 1.1.4.5.10)(RecordsetEventHandler *handler*)

public void **removeOnFieldChangeComplete**(RecordsetEventHandler *handler*)

public void addOnMoveComplete(See 1.1.4.5.15)(RecordsetEventHandler *handler*)

public void **removeOnMoveComplete**(RecordsetEventHandler *handler*)

public void addOnRecordChangeComplete(See 1.1.4.5.11)(RecordsetEventHandler *handler*)

public void **removeOnRecordChangeComplete**(RecordsetEventHandler *handler*)

public void addOnRecordsetChangeComplete(See 1.1.4.5.12)(RecordsetEventHandler *handler*)

public void **removeOnRecordsetChangeComplete**(RecordsetEventHandler *handler*)

public void addOnWillChangeField(See 1.1.4.5.10)(RecordsetEventHandler *handler*)

public void **removeOnWillChangeField**(RecordsetEventHandler *handler*)

public void addOnWillChangeRecord(See 1.1.4.5.11)(RecordsetEventHandler *handler*)

public void **removeOnWillChangeRecord**(RecordsetEventHandler *handler*)

public void addOnWillChangeRecordset(See 1.1.4.5.12)(RecordsetEventHandler *handler*)

public void **removeOnWillChangeRecordset**(RecordsetEventHandler *handler*)

public void addOnWillMove(See 1.1.4.5.15)(RecordsetEventHandler *handler*)

public void **removeOnWillMove**(RecordsetEventHandler *handler*)

1.1.7.2.5 Field (ADO/WFC 语法)

包 com.ms.wfc.data

方法

public void appendChunk(See 1.1.4.4.3)(byte[] *bytes*)

public void **appendChunk**(char[] *chars*)

public void **appendChunk**(String *chars*)

public byte[] **getByteChunk**(int *len*)

public char[] **getCharChunk**(int *len*)

public String **getStringChunk**(int *len*)

属性

public int getActualSize(See 1.1.4.6.5)()

public int getAttributes(See 1.1.4.6.6)()

public void **setAttributes**(int *pl*)

public com.ms.com.IUnknown **getDataFormat**()

public void **setDataFormat**(com.ms.com.IUnknown *format*)

（详细信息，请参阅 Microsoft Visual J++ WFC 参考文档中有关 com.ms.wfc.data.IDataFormat 接口的内容。）

public int getDefinedSize(See 1.1.4.6.22)()

public void **setDefinedSize**(int *pl*)

public String getName(See 1.1.4.6.41)()

public int getNumericScale(See 1.1.4.6.44)()

public void **setNumericScale**(byte *pbNumericScale*)

public Variant getOriginalValue(See 1.1.4.6.46)()

public int getPrecision(See 1.1.4.6.49)()

public void **setPrecision**(byte *pbPrecision*)

public int getType(See 1.1.4.6.67)()

public void **setType**(int *pDataType*)

public Variant getUnderlyingValue(See 1.1.4.6.68)()

public Variant getValue(See 1.1.4.6.69)()

public void **setValue**(Variant *value*)

public AdoProperties getProperties(See 1.1.4.3.4)()

Field 存取方法

Field 对象的 **Value** 属性可获得或设置该对象的内容。内容以 VARIANT 表示，VARIANT 是可被赋以值和多种数据类型的对象类型。

ADO/WFC 使用 **getValue** 方法和 **setValue** 方法实现 **Value** 属性，**getValue** 返回 VARIANT 对象，**setValue** 则把 VARIANT 当作参数使用。虽然在某些语言(如 Microsoft Visual Basic) 中变体型的效率已经很高。但仍可以在 Microsoft Visual J++ 中通过使用

本地 Java 数据类型获得更高的性能。

除 **Value** 属性外，ADO/WFC 还提供使用 Java 数据类型获得并设置 **Field** 对象内容的存取方法。大多数这些方法都具有名称，其形式为 **Get**DataType 或 **Set**DataType。

有两点例外须加以注意。**getObject** 方法之一可返回强制为指定类的对象；不存在 **getNull** 属性。但 **isNull** 属性是存在的，它返回的布尔值可指明字段是否为空。

public native boolean **getBoolean**();

public void **setBoolean**(boolean *v*)

public native byte **getByte**();

public void **setByte**(byte *v*)

public native byte[] **getBytes**();

public void **setBytes**(byte[] *v*)

public native double **getDouble**();

public void **setDouble**(double *v*)

public native float **getFloat**();

public void **setFloat**(float *v*)

public native int **getInt**();

public void **setInt**(int *v*)

public native long **getLong**();

public void **setLong**(long *v*)

public native short **getShort**();

public void **setShort**(short *v*)

public native String **getString**();

public void **setString**(String *v*)

public native boolean **isNull**();

public void **setNull**()

public Object **getObject**()

public Object **getObject**(Class *c*)

public void **setObject**(Object *value*)

1.1.7.2.6 Error (ADO/WFC 语法)

包 com.ms.wfc.data

Properties

public String getDescription(See 1.1.4.6.23)()

public int getNativeError(See 1.1.4.6.42)()

public int getNumber(See 1.1.4.6.43)()

public String getSource(See 1.1.4.6.60)()

public String getSQLState(See 1.1.4.6.63)()

1.1.7.2.7 集合 (ADO/WFC 语法)

包 com.ms.wfc.data

Parameters

方法

public void append(See 1.1.4.4.2)(com.ms.wfc.data.Parameter *param*)

public void delete(See 1.1.4.4.18)(int *n*)

public void **delete**(String *s*)

public void refresh(See 1.1.4.4.36)()

public Parameter getItem(See 1.1.4.4.27)(int *n*)

public Parameter **getItem**(String *s*)

属性

public int getCount(See 1.1.4.6.16)()

Fields

方法

public void append(See 1.1.4.4.2)(String *name*, int *type*)

public void **append**(String *name*, int *type*, int *definedSize*)

public void **append**(String *name*, int *type*, int *definedSize*, int *attrib*)

public void delete(See 1.1.4.4.19)(int *n*)

public void **delete**(String *s*)

public void refresh(See 1.1.4.4.36)()

public com.ms.wfc.data.Field getItem(See 1.1.4.4.27)(int *n*)

public com.ms.wfc.data.Field **getItem**(String *s*)

属性

public int getCount(See 1.1.4.6.16)()

Errors

方法

public void clear(See 1.1.4.4.10)()

public void refresh(See 1.1.4.4.36)()

public com.ms.wfc.data.Error getItem(See 1.1.4.4.27)(int *n*)

public com.ms.wfc.data.Error **getItem**(String *s*)

属性

public int getCount(See 1.1.4.6.16)()

1.1.7.2.8 DataSpace (ADO/WFC 语法)

ADO/WFC **DataSpace** 类有一个方法 **createObject**，用于指定服务器和通讯协议；服务器对象处理客户端应用程序请求，并返回代表服务器的 ObjectProxy(See 1.1.7.2.9) 对象。

包 com.ms.wfc.data

构造函数

public DataSpace()

方法

public static ObjectProxy *DataSpace*.createObject(See 1.1.4.4.15)(String *progid*, String *connection*)

属性

public static int getInternetTimeout(See 1.1.4.6.35)()

public static void **setInternetTimeout**(int *plInetTimeout*)

1.1.7.2.9 ObjectProxy (ADO/WFC 语法)

ObjectProxy 对象代表服务器，并由 DataSpace(See 1.1.4.2.5) 对象的 **createObject** 方法返回。ObjectProxy 类有一个方法 **call**，可以调用服务器的方法并返回由该调用产生的对象。

包 com.ms.wfc.data

方法

Call 方法(ADO/WFC 语法)

调用由 ObjectProxy 代表的服务器上的方法。可选地，方法参数可以作为对象数组传递。

语法

public Object *ObjectProxy*.**call**(String *method*)

public Object *ObjectProxy*.**call**(String *method*, Object[] *args*)

返回

Object 由调用方法所产生的对象。

参数

ObjectProxy 代表服务器的 ObjectProxy 对象。

method 字符串，包含在服务器上调用的方法的名称。

args 可选。对象数组，是服务器上方法的参数。Java 数据类型被自动转换成适合用于服务器的数据类型。

1.1.7.2.10 AdoEnums (ADO/WFC 语法)

要使用 AdoEnums，请指定枚举常量的完整合法名称。例如，在 **AdoEnums.Schema** 类中的第一个常量是 AdoEnums.Schema.PROVIDERSPECIFIC。

每一个主题都列出了常量的值。但请使用常量名，而不是值。这些值根据版本不同会有更改。

下表包含三个从属于核心 ADO 文档的标题。

- **主题** 该标题链接到描述枚举常量的 ADO 文档主题。
- **前缀** ADO/WFC 枚举常量的名称来自对应的 ADO 常量。但在读取 ADO 文档时，哪一个是对应的 ADO 常量可能并不是显而易见的。

  ADO/WFC 常量的最后限定符是“标识符”，用大写字母书写。ADO 常量包括“前缀”，后跟大小写混合的标识符。

  例如，ADO/WFC 常量 AdoEnums.Schema.PROVIDERSPECIFIC 对应的是 ADO 常量 adSchemaProviderSpecific。这时，前缀 adSchema 被追加到标识符 ProviderSpecific 之前。

  该标题列出了应该追加到 ADO/WFC 标识符来生成 ADO 枚举常量名称的前缀。

- **声明** ADO COM 接口文件对每个集枚举常量进行声明。该标题显示该声明的名称。

通过给第二个限定符添加后缀 Enum，可以从 ADO/WFC 常量得到 COM 声明的名称。例如，与 AdoEnums.Schema.* 关联的 COM 声明名称为 SchemaEnum。

在 ASO/WFC 中定义了下列的枚举量。

AdcPropAsyncThreadPriority(See 1.1.7.2.10.1)	DataType(See 1.1.7.2.10.13)	ObjectState(See 1.1.7.2.10.25)
AdcPropUpdateCriteria(See 1.1.7.2.10.2)	EditMode(See 1.1.7.2.10.14)	ParameterAttributes(See 1.1.7.2.10.26)
Affect(See 1.1.7.2.10.3)	ErrorValue(See 1.1.7.2.10.15)	ParameterDirection(See 1.1.7.2.10.27)
Bookmark(See 1.1.7.2.10.4)	EventReason(See 1.1.7.2.10.16)	PersistFormat(See 1.1.7.2.10.28)
CommandType(See 1.1.7.2.10.5)	EventStatus(See 1.1.7.2.10.17)	Position(See 1.1.7.2.10.29)
Compare(See 1.1.7.2.10.6)	ExecuteOption(See 1.1.7.2.10.18)	PropertyAttributes(See 1.1.7.2.10.30)
ConnectMode(See 1.1.7.2.10.7)	FieldAttribute(See 1.1.7.2.10.19)	RecordStatus(See 1.1.7.2.10.31)
ConnectOption(See 1.1.7.2.10.8)	FilterGroup(See 1.1.7.2.10.20)	Resync(See 1.1.7.2.10.32)
ConnectPrompt(See 1.1.7.2.10.9)	GetRowsOption(See 1.1.7.2.10.21)	Schema(See 1.1.7.2.10.33)
CursorLocation(See 1.1.7.2.10.10)	IsolationLevel(See 1.1.7.2.10.22)	SearchDirection(See 1.1.7.2.10.34)
CursorOption(See 1.1.7.2.10.11)	LockType(See 1.1.7.2.10.23)	StringFormat(See 1.1.7.2.10.35)
CursorType(See 1.1.7.2.10.12)	MarshalOptions(See 1.1.7.2.10.24)	XactAttribute(See 1.1.7.2.10.36)

1.1.7.2.10.1 AdoEnums.AdcPropAsyncThreadPriority.*

包 **com.ms.wfc.data**

主题 对于 RDS **Recordset** 对象，指定获取数据的异步线程的执行优先级。通过记录集的“**Background Thread Priority**”动态属性使用这些常量。

前缀 adPriority

声明 ADCPROP_ASYNCTHREADPRIORITY_ENUM

常量	值
LOWEST	1
BELOWNORMAL	2
NORMAL	3
ABOVENORMAL	4
HIGHEST	5

1.1.7.2.10.2 AdoEnums.AdcPropUpdateCriteria.*

包 com.ms.wfc.data

主题 指定哪些字段可用来监测在数据源的行的优化更新过程中，与 RDS **Recordset** 对象的冲突。常量指定如果关键字有更改（即，已经删除数据源行）、数据源行中的任何列有更改、与记录集的更新列相同的数据源行的列有更改、或者如果数据源行只是被访问过，则应当检查冲突。通过记录集的 **Update Criteria** 动态属性使用这些常量。

前缀 adCriteria

声明 ADCPROP_UPDATECRITERIA_ENUM

常量	值
KEY	0
ALLCOLS	1
UPDCOLS	2
TIMESTAMP	3

1.1.7.2.10.3 AdoEnums.Affect.*

包 com.ms.wfc.data

主题 UpdateBatch(See 1.1.4.4.46) 方法

前缀 adAffect

声明 AffectEnum

常量	值
CURRENT	1
GROUP	2
ALL	3
ALLCHAPTERS	4

1.1.7.2.10.4 AdoEnums.Bookmark.*

包 com.ms.wfc.data

主题 Find(See 1.1.4.4.23)、GetRows(See 1.1.4.4.25) 和 Move(See 1.1.4.4.29) 方法

前缀 adBookmark

声明 BookmarkEnum

常量	值
CURRENT	0

FIRST	1
LAST	2

1.1.7.2.10.5 AdoEnums.CommandType.*

包 com.ms.wfc.data

主题 CommandType(See 1.1.4.6.12) 属性

前缀 adCmd

声明 CommandTypeEnum

常量	值
UNSPECIFIED	-1
UNKNOWN	8
TEXT	1
TABLE	2
STOREDPROC	4
FILE	256
TABLEDIRECT	512

1.1.7.2.10.6 AdoEnums.Compare.*

包 com.ms.wfc.data

主题 CompareBookmarks(See 1.1.4.4.13) 方法

前缀 adCompare

声明 CompareEnum

常量	值
LESSTHAN	0
EQUAL	1
GREATERTHAN	2
NOTEQUAL	3
NOTCOMPARABLE	4

1.1.7.2.10.7 AdoEnums.ConnectMode.*

包 **com.ms.wfc.data**

主题 Mode(See 1.1.4.6.40) 属性

前缀 adMode

声明 ConnectModeEnum

常量	值
UNKNOWN	0
READ	1
WRITE	2
READWRITE	3
SHAREDENYREAD	4
SHAREDENYWRITE	8
SHAREEXCLUSIVE	12
SHAREDENYNONE	16

1.1.7.2.10.8 AdoEnums.ConnectOption.*

包 **com.ms.wfc.data**

主题 Connect.Open(See 1.1.4.4.32) 方法（按 OpenOptionEnum 引用）

前缀 ad

声明 ConnectOptionEnum

常量	值
CONNECTUNSPECIFIED	-1
ASYNCCONNECT	16

1.1.7.2.10.9 AdoEnums.ConnectPrompt.*

包 **com.ms.wfc.data**

主题 指定是否应该显示对话框，来提示在打开到 ODBC 数据源的连接时丢失参数。这些常量将指定是否始终提示、如果需详细信息时提示、如果需要详细信息但不允许可选参数时提示或者始终不提示。通过连接的“**Prompt**”动态属性使用这些常量。

前缀 adPrompt

声明 ConnectPromptEnum

常量	值

ALWAYS	1
COMPLETE	2
COMPLETEREQUIRED	3
NEVER	4

1.1.7.2.10.10 AdoEnums.CursorLocation.*

包 **com.ms.wfc.data**

主题 CursorLocation(See 1.1.4.6.17) 属性

前缀 adUse

声明 CursorLocationEnum

常量	值
NONE	1
SERVER	2
CLIENT	3

1.1.7.2.10.11 AdoEnums.CursorOption.*

包 **com.ms.wfc.data**

主题 Supports(See 1.1.4.4.44) 方法

前缀 ad

声明 CursorOptionEnum

常量	值
HOLDRECORDS	256
MOVEPREVIOUS	512
ADDNEW	16778240
DELETE	16779264
UPDATE	16809984
BOOKMARK	8192
APPROXPOSITION	16384
UPDATEBATCH	65536
RESYNC	131072

NOTIFY	262144
FIND	524288

1.1.7.2.10.12 AdoEnums.CursorType.*

包 **com.ms.wfc.data**

主题 CursorType(See 1.1.4.6.18) 属性

前缀 adOpen

声明 CursorTypeEnum

常量	值
UNSPECIFIED	-1
FORWARDONLY	0
KEYSET	1
DYNAMIC	2
STATIC	3

1.1.7.2.10.13 AdoEnums.DataType.*

包 **com.ms.wfc.data**

主题 Append(See 1.1.4.4.2) 方法

前缀 ad

声明 DataTypeEnum

常量	值	常量	值	常量	值
EMPTY	0	DECIMAL	14	DBTIMESTAMP	135
TINYINT	16	NUMERIC	131	BSTR	8
SMALLINT	2	BOOLEAN	11	CHAR	129
INTEGER	3	ERROR	10	VARCHAR	200
BIGINT	20	USERDEFINED	132	LONGVARCHAR	201
UNSIGNEDTINYINT	17	VARIANT	12	WCHAR	130
UNSIGNEDSMALLINT	18	IDISPATCH	9	VARWCHAR	202
UNSIGNEDINT	19	IUNKNOWN	13	LONGVARWCHAR	203
UNSIGNEDBIGINT	21	GUID	72	BINARY	128

SINGLE	4	DATE	7	VARBINARY	204
DOUBLE	5	DBDATE	133	LONGVARBINARY	205
CURRENCY	6	DBTIME	134	CHAPTER	136

1.1.7.2.10.14 AdoEnums.EditMode.*

包 **com.ms.wfc.data**

主题 EditMode(See 1.1.4.6.25) 属性

前缀 adEdit

声明 EditModeEnum

常量	值
NONE	0
INPROGRESS	1
ADD	2
DELETE	4

1.1.7.2.10.15 AdoEnums.ErrorValue.*

包 **com.ms.wfc.data**

主题 ADO 错误代码(See 1.1.8.1)

前缀 adErr

声明 ErrorValueEnum

常量	值	常量	值
INVALIDARGUMENT	3001	OBJECTOPEN	3705
NOCURRENTRECORD	3021	PROVIDERNOTFOUND	3706
ILLEGALOPERATION	3219	BOUNDTOCOMMAND	3707
INTRANSACTION	3246	INVALIDPARAMINFO	3708
FEATURENOTAVAILABLE	3251	INVALIDCONNECTION	3709
ITEMNOTFOUND	3265	NOTREENTRANT	3710
OBJECTINCOLLECTION	3367	STILLEXECUTING	3711
OBJECTNOTSET	3420	OPERATIONCANCELLED	3712
DATACONVERSION	3421	STILLCONNECTING	3713

OBJECTCLOSED	3704	NOTEXECUTING	3715
		UNSAFEOPERATION	3716

1.1.7.2.10.16 AdoEnums.EventReason.*

包 com.ms.wfc.data

主题 WillMove(See 1.1.4.5.15)、WillChangeRecord(See 1.1.4.5.11) 和 WillChangeRecordset(See 1.1.4.5.12) 方法

前缀 adRsn

声明 EventReasonEnum

常量	值	常量	值
ADDNEW	1	CLOSE	9
DELETE	2	MOVE	10
UPDATE	3	FIRSTCHANGE	11
UNDOUPDATE	4	MOVEFIRST	12
UNDOADDNEW	5	MOVENEXT	13
UNDODELETE	6	MOVEPREVIOUS	14
REQUERY	7	MOVELAST	15
RESYNCH	8		

1.1.7.2.10.17 AdoEnums.EventStatus.*

包 com.ms.wfc.data

主题 所有 ADO 事件(See 1.1.4.5)

前缀 adStatus

声明 EventStatusEnum

常量	值
OK	1
ERRORSOCCURRED	2
CANTDENY	3
CANCEL	4
UNWANTEDEVENT	5

1.1.7.2.10.18 AdoEnums.ExecuteOption.*

包 **com.ms.wfc.data**

主题 Execute(See 1.1.4.4.21) 方法

前缀 ad

声明 ExecuteOptionEnum

常量	值
UNSPECIFIED	-1
ASYNCEXECUTE	16
ASYNCFETCH	32
ASYNCFETCHNONBLOCKING	64

1.1.7.2.10.19 AdoEnums.FieldAttribute.*

包 **com.ms.wfc.data**

主题 Attribute(See 1.1.4.6.6) 属性

前缀 adFld

声明 FieldAttributeEnum

常量	值
UNSPECIFIED	-1
MAYDEFER	2
UPDATABLE	4
UNKNOWNUPDATABLE	8
FIXED	16
ISNULLABLE	32
MAYBENULL	64
LONG	128
ROWID	256
ROWVERSION	512
CACHEDEFERRED	4096

1.1.7.2.10.20 AdoEnums.FilterGroup.*

包 **com.ms.wfc.data**

主题 Filter(See 1.1.4.6.28) 属性

前缀 adFilter

声明 FilterGroupEnum

常量	值
NONE	0
PENDINGRECORDS	1
AFFECTEDRECORDS	2
FETCHEDRECORDS	3
PREDICATE	4
CONFLICTINGRECORDS	5

1.1.7.2.10.21 AdoEnums.GetRowsOption.*

包 **com.ms.wfc.data**

主题 GetRows(See 1.1.4.4.25) 方法

前缀 adGetRows

声明 GetRowsOptionEnum

常量	值
REST	-1

1.1.7.2.10.22 AdoEnums.IsolationLevel.*

包 **com.ms.wfc.data**

主题 IsolationLevel(See 1.1.4.6.36) 属性

前缀 adXact

声明 IsolationLevelEnum

常量	值
UNSPECIFIED	-1
CHAOS	16
READUNCOMMITTED	256
BROWSE	256

CURSORSTABILITY	4096
READCOMMITTED	4096
REPEATABLEREAD	65536
SERIALIZABLE	1048576
ISOLATED	1048576

1.1.7.2.10.23 AdoEnums.LockType.*

包 com.ms.wfc.data

主题 LockType(See 1.1.4.6.37) 属性

前缀 adLock

声明 LockTypeEnum

常量	值
UNSPECIFIED	-1
READONLY	1
PESSIMISTIC	2
OPTIMISTIC	3
BATCHOPTIMISTIC	4

1.1.7.2.10.24 AdoEnums.MarshalOptions.*

包 com.ms.wfc.data

主题 MarshalOptions(See 1.1.4.6.38) 属性

前缀 adMarshal

声明 MarshalOptionsEnum

常量	值
ALL	0
MODIFIEDONLY	1

1.1.7.2.10.25 AdoEnums.ObjectState.*

包 com.ms.wfc.data

主题 State(See 1.1.4.6.64) 属性

前缀 adState

声明 ObjectStateEnum

常量	值
CLOSED	0
OPEN	1
CONNECTING	2
EXECUTING	4
FETCHING	8

1.1.7.2.10.26 AdoEnums.ParameterAttributes.*

包 **com.ms.wfc.data**

主题 Attribute(See 1.1.4.6.6) 属性

前缀 adParam

声明 ParameterAttributesEnum

常量	值
SIGNED	16
NULLABLE	64
LONG	128

1.1.7.2.10.27 AdoEnums.ParameterDirection.*

包 **com.ms.wfc.data**

主题 Direction(See 1.1.4.6.24) 属性

前缀 adParam

声明 ParameterDirectionEnum

常量	值
UNKNOWN	0
INPUT	1
OUTPUT	2
INPUTOUTPUT	3
RETURNVALUE	4

1.1.7.2.10.28 AdoEnums.PersistFormat.*

包 **com.ms.wfc.data**

主题 Save(See 1.1.4.4.41) 方法

前缀 adPersist

声明 PersistFormatEnum

常量	值
ADTG	0
XML	1

1.1.7.2.10.29 AdoEnums.Position.*

包 **com.ms.wfc.data**

主题 AbsolutePage(See 1.1.4.6.1) 和 AbsolutePosition(See 1.1.4.6.2) 属性

前缀 adPos

声明 PositionEnum

常量	值
UNKNOWN	-1
BOF	-2
EOF	-3

1.1.7.2.10.30 AdoEnums.PropertyAttributes.*

包 **com.ms.wfc.data**

主题 Attribute(See 1.1.4.6.6) 属性

前缀 adProp

声明 PropertyAttributesEnum

常量	值
NOTSUPPORTED	0
REQUIRED	1
OPTIONAL	2
READ	512
WRITE	1024

1.1.7.2.10.31 AdoEnums.RecordStatus.*

包 **com.ms.wfc.data**

主题 Status(See 1.1.4.6.65) 属性

前缀 adRec

声明 RecordStatusEnum

常量	值	常量	值
OK	0	CANTRELEASE	1024
NEW	1	CONCURRENCYVIOLATION	2048
MODIFIED	2	INTEGRITYVIOLATION	4096
DELETED	4	MAXCHANGESEXCEEDED	8192
UNMODIFIED	8	OBJECTOPEN	16384
INVALID	16	OUTOFMEMORY	32768
MULTIPLECHANGES	64	PERMISSIONDENIED	65536
PENDINGCHANGES	128	SCHEMAVIOLATION	131072
CANCELED	256	DBDELETED	262144

1.1.7.2.10.32 AdoEnums.Resync.*

包 **com.ms.wfc.data**

主题 Resync(See 1.1.4.4.40) 方法

前缀 adResync

声明 ResyncEnum

常量	值
UNDERLYINGVALUES	1
ALLVALUES	2

1.1.7.2.10.33 AdoEnums.Schema.*

包 **com.ms.wfc.data**

主题 OpenSchema(See 1.1.4.4.34) 方法

前缀 adSchema

声明 SchemaEnum

常量	值	常量	值

PROVIDERSPECIFIC	-1	STATISTICS	19
ASSERTS	0	TABLES	20
CATALOGS	1	TRANSLATIONS	21
CHARACTERSETS	2	PROVIDERTYPES	22
COLLATIONS	3	VIEWS	23
COLUMNS	4	VIEWCOLUMNUSAGE	24
CHECKCONSTRAINTS	5	VIEWTABLEUSAGE	25
CONSTRAINTCOLUMNUSAGE	6	PROCEDUREPARAMETERS	26
CONSTRAINTTABLEUSAGE	7	FOREIGNKEYS	27
KEYCOLUMNUSAGE	8	PRIMARYKEYS	28
REFERENTIALCONTRAINTS	9	PROCEDURECOLUMNS	29
TABLECONSTRAINTS	10	DBINFOKEYWORDS	30
COLUMNSDOMAINUSAGE	11	DBINFOLITERALS	31
INDEXES	12	CUBES	32
COLUMNPRIVILEGES	13	DIMENSIONS	33
TABLEPRIVILEGES	14	HIERARCHIES	34
USAGEPRIVILEGES	15	LEVELS	35
PROCEDURES	16	MEASURES	36
SCHEMATA	17	PROPERTIES	37
SQLLANGUAGES	18	MEMBERS	38

1.1.7.2.10.34 AdoEnums.SearchDirection.*

包 **com.ms.wfc.data**

主题 Find(See 1.1.4.4.23) 方法

前缀 adSearch

声明 SearchDirectionEnum

常量	值
FORWARD	1
BACKWARD	-1

1.1.7.2.10.35 AdoEnums.StringFormat.*

包 **com.ms.wfc.data**

主题 GetString(See 1.1.4.4.26) 方法

前缀 ad

声明 StringFormatEnum

常量	值
CLIPSTRING	2

1.1.7.2.10.36 AdoEnums.XactAttribute.*

包 **com.ms.wfc.data**

主题 Attribute(See 1.1.4.6.6) 属性

前缀 adXact

声明 XactAttributeEnum

常量	值
COMMITRETAINING	131072
ABORTRETAINING	262144
ASYNCPHASEONE	524288
SYNCPHASEONE	1048576

1.1.8 错误代码

该主题正在建设。

有关特定错误消息的详细信息，请参阅如下主题：

- ADO 错误代码(See 1.1.8.1)
- RDS.DataControl 错误代码(See 1.1.8.2)
- Internet Explorer 错误代码(See 1.1.8.3)
- Internet Information Server 错误代码(See 1.1.8.4)

1.1.8.1 ADO 错误代码

除了在 **Error** 对象和 **Errors** 集合中说明的提供者错误之外，ADO 本身也将错误返回到运行时环境的异常处理机制之中。使用编程语言的错误捕获机制（如 Microsoft® Visual Basic® 中的 **On Error** 语句）可捕获及处理下列错误。下表将同时显示十进制和十六进制错误代码值。

常量名称	编号	说明
adErrInvalidArgument	3001 0x800A0BB9	应用程序使用的参数其类型错误、超出可接受的范围或者与其他参数冲突。
adErrNoCurrentRecord	3021 0x800A0BCD	BOF(See 1.1.4.6.7) 或 EOF(See 1.1.4.6.7) 为 **True**，或者当前记录已经删除。应用程序请求的操作需要当前记录。
adErrIllegalOperation	3219 0x800A0C93	应用程序请求的操作不允许出现在该上下文中
adErrInTransaction	3246 0x800A0CAE	在事务中应用程序无法显式关闭 Connection(See 1.1.4.2.2) 对象。
adErrFeatureNotAvailable	3251 0x800A0CB3	提供者不支持应用程序请求的操作。
adErrItemNotFound	3265 0x800A0CC1	ADO 无法在对应于应用程序请求的名称或顺序引用的集合中找到对象。
adErrObjectInCollection	3367 0x800A0D27	无法追加，对象已经在集合中。
adErrObjectNotSet	3420 0x800A0D5C	应用程序引用的对象不再指向有效的对象。
adErrDataConversion	3421 0x800A0D5D	应用程序使用了不符合对当前操作的值类型。
adErrObjectClosed	3704 0x800A0E78	如果对象关闭，则不允许应用程序请求的操作。
adErrObjectOpen	3705 0x800A0E79	如果对象打开，则不允许应用程序请求的操作。
adErrProviderNotFound	3706 0x800A0E7A	ADO 找不到指定的提供者。
adErrBoundToCommand	3707 0x800A0E7B	应用程序无法用 Command(See 1.1.4.2.1) 对象将 Recordset(See 1.1.4.2.10) 对象的 ActiveConnection(See 1.1.4.6.4) 属性更改为它的来源数据。
adErrInvalidParamInfo	3708 0x800A0E7C	应用程序错误地定义了 Parameter(See 1.1.4.2.8) 对象。

adErrInvalidConnection	3709 0x800A0E7D	应用程序通过引用关闭或无效的 Connection(See 1.1.4.2.2) 对象来请求对对象的操作。

1.1.8.2 DataControl 错误代码

RDS.DataControl 错误代码	编号	说明
IDS_UpdatesFailed	4098 0x800A1002	无法更新数据库。
IDS_CantConnect	4099 0x800A1003	无法连接到服务器。
IDS_CantCreateObject	4100 0x800A1004	无法创建业务对象。
IDS_CantInvokeMethod	4101 0x800A1005	无法调用业务对象的方法。
IDS_CantFindDataspace	4102 0x800A1006	**Dataspace** 属性无效。
IDS_ObjectNotSafe	4103 0x800A1007	对象对于脚本和初始化不安全。
IDS_RowsetNotUpdateable	4104 0x800A1008	行集合无法更新。
IDS_BadInlineTablegram	4105 0x800A1009	错误的内嵌表格语法。
IDS_InvalidADCClientVersion	4106 0x800A1010	无效的 RDS 客户端版本：客户端比服务器新。
IDS_AsyncPending	4107 0x800A1011	无法用异步操作挂起执行操作。
IDS_ResetInvalidField	4108 0x800A1012	在 **SortColumn** 或 **FilterColumn** 中指定的列不存在。
IDS_RecordsetNotOpen	4109 0x800A1013	记录集不处于打开状态。
IDS_InvalidParam	4110 0x800A1014	一个或多个参数无效。
IDS_UnexpectedError	4351	未知错误。

	0x800A10FF	

1.1.8.3 Internet Explorer 错误代码

Internet Explorer (Wininet) 错误	编号	说明
IDS_WinInet_Error	8193 0x800A2001	Internet 客户端错误。
IDS_WinInet_Timeout	8194 0x800A2002	Internet 客户端错误：请求超时。
IDS_WinInet_CantConnect	8195 0x800A2003	Internet 客户端错误：无法连接到服务器。
IDS_WinInet_SSLPostLimitation	8196 0x800A2004	Internet 客户端错误：SSL 错误（可能是由于 32K 数据上载限制）。
IDS_WinInet_InvalidServerResponse	8430 0x800A20EE	Internet 客户端错误：无效的服务器响应。
IDS_WinInet_ConnectionReset	12031 0x800A2EFF	Internet 客户端错误：连接复位。

1.1.8.4 Internet Information Server 错误代码

下表列出与“远程数据服务”的使用有关的 Internet Information Server 十进制和十六进制错误代码。

Internet Information Server 错误	编号	说明
IDS_IIS_AccessDenied	8208 0x800A2010	Internet 服务器错误：拒绝访问。
IDS_IIS_ObjectNotFound	8209 0x800A2011	Internet 服务器错误：找不到对象/模型。
IDS_IIS_RequestForbidden	8210 0x800A2012	Internet 服务器错误：禁止请求。
IDS_IIS_UnexpectedError	8447 0x800A20FF	Internet 服务器错误。

1.1.9 ADO 配置信息

- 授予 Web 服务器计算机客户特权(See 1.1.9.1)
- 注册自定义业务对象(See 1.1.9.2)
- 将业务对象标记为“脚本安全”(See 1.1.9.3)
- 在客户端注册业务对象以便用于 DCOM(See 1.1.9.4)
- 使 DLL 能够在 DCOM 上运行(See 1.1.9.5)
- 了解 Microsoft Internet Explorer 安全问题(See 1.1.9.6)

1.1.9.1 授予 Web 服务器计算机客户特权

授予 Web 服务器计算机客户特权

1. 在 Windows NT® 数据源计算机上，单击“开始”，指向“程序”，然后单击“管理工具”。
2. 运行“域用户管理器”。
3. 在“用户”菜单中，单击“选择域”。键入当前计算机名称，然后单击“确定”。
4. 在下边的窗口中，双击“Guests”组，然后单击“添加”加入计算机名称帐号。默认情况下，该帐号为 IUSR_MYWEBSVR（MYWEBSVR 是 Web 服务器名）。

1.1.9.2 注册自定义业务对象

要成功地通过 Web 服务器启动自定义业务对象(See 2.7)（.dll 或 .exe)，必须按如下过程的说明，将业务对象的 ProgID 输入注册表。该 RDS 功能通过只运行经过认可的可执行程序来保证 Web 服务器的安全。默认的业务对象 RDSServer.DataFactory(See 1.1.4.2.4) 已完全注册。

注册自定义业务对象

1. 指向“开始”并单击“运行”。
2. 键入“RegEdit”，然后单击“确定”。
3. 在“注册表编辑器”中，定位到“HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\W3SVC\Parameters\ADCLaunch”注册表键。
4. 选择“ADCLaunch”键，指向“编辑”菜单中的“新建”，然后单击“主键”。

5. 键入自定义业务对象的 ProgID 并单击“输入”，将“值”项置空。

1.1.9.3 将业务对象标记为“脚本安全”

为确保 Internet 环境的安全，需要将由 RDS.DataSpace(See 1.1.4.2.5) 对象的 CreateObject(See 1.1.4.4.15) 方法事例化的业务对象(See 2.7)标记为“脚本安全”。在 DCOM(See 2.19) 中使用之前，应确保在系统注册表的“许可”区域中对该业务对象进行了同样的标记。

要手工标记业务对象为“脚本安全”，需要创建包含以下文本的具有 .reg 扩展名的文本文件。下面的两组编号可以启用“脚本安全”功能：

[HKEY_CLASSES_ROOT\CLSID\<*MyActiveXGUID*>\Implemented Categories\{7DD95801-9882-11CF-9FA9-00AA006C42C4}]

[HKEY_CLASSES_ROOT\CLSID\<*MyActiveXGUID*>\Implemented Categories\{7DD95802-9882-11CF-9FA9-00AA006C42C4}]

其中 <MyActiveXGUID> 是业务对象的十六进制 GUID 号码，将其保存并通过使用“注册表编辑器”或双击“Windows 资源管理器”中的 .reg 文件将其合并到注册表中。

在 Microsoft® Visual Basic® 中创建的业务对象可通过“打包和展开向导”被自动标记为“脚本安全”。当向导要求指定安全设置时，请选择“初始化安全”和“脚本安全”。

在最后的步骤中，“应用程序安装向导”创建了一个 .htm 和一个 .cab 文件，可以将这两个文件复制到目标计算机并双击 .htm 文件以加载页面并正确注册服务器。

由于在默认情况下业务对象将安装在 Windows\System32\Occache 目录中，因此要将其移动到 Windows\System32 目录并改变 **HKEY_CLASSES_ROOT\CLSID**\<*MyActiveXGUID*>**InprocServer32** 注册表键使之与正确的路径相匹配。

1.1.9.4 在客户端注册业务对象以便用于 DCOM

自定义业务对象(See 2.7)需要确保客户端能将其程序名称 (ProgId) 映射到可用于 DCOM 的标识符 (CLSID(See 2.10))。为此 DCOM 对象的 ProgID 必须位于客户端注册表并映射到服务器端业务对象的类标识码。不过，这对于所支持的其他协议（HTTP、HTTPS 以及进程内）则没有必要。

例如，如果要显示具有特定类标识码的名为 MyBObj 的服务器端业务对象如“{00112233-4455-6677-8899-00AABBCCDDEE}”，就需要确认以下条目已经添加到了客户端注册表：

```
[HKEY_CLASSES_ROOT]
\MyBObj
   \Clsid
   (Default) "{00112233-4455-6677-8899-00AABBCCDDEE}"
```

DCOM 流调度

使用 RDS 1.5 或此前组件的客户端计算机与使用 RDS 2.0 组件的服务器不兼容。新的 RDS 2.0 支持在传送 **Recordset** 对象方面效率更高。如果遇到这种情况，可以设置服务器，使之可与先前的 RDS 支持（称为 RDS 1.0）或新的 RDS 支持（称为 RDS 2.0）一同工作。请选择设置以下某个注册表项：

```
[HKEY_CLASSES_ROOT]
\CLSID
```

```
   \[58ECEE30-E715-11CF-B0E3-00AA003F000F}
      \ADTGOptions]"MarshalFormat"="RDS10"
```

-或者-

```
[HKEY_CLASSES_ROOT]
\CLSID
   \[58ECEE30-E715-11CF-B0E3-00AA003F000F}
      \ADTGOptions]"MarshalFormat"="RDS20"
```

1.1.9.5 使 DLL 能够在 DCOM 上运行

以下步骤概述如何通过 Microsoft® Transaction Server(See 2.31) 使业务对象(See 2.7) .dll(See 2.22) 能够同时使用 DCOM 以及 Microsoft® Internet Information Server(HTTP)。

1. 在 Transaction Server Explorer 中创建新的软件包。

   可以使用 Transaction Server Explorer 创建软件包并将 DLL 加入其中。这样通过 DCOM 便可以访问 .dll，但将无法通过 IIS 对其访问。（如果在注册表中检查 .dll，则 **Inproc** 键现在为空；请设置 Activation 属性（我们在稍后给出相应的解释），并在 **Inproc** 键中添加值。）

2. 将业务对象安装到软件包中。

   -或者-

   将 **RDSServer.DataFactory** 对象导入软件包。

3. 将组件的 Activation 属性设置为“在创建者的进程中”。

   为确保可通过同一台计算机上的 DCOM 和 IIS 访问 .dll，必须在 Microsoft Transaction Server Explorer 中设置组件的 Activation 属性。将属性设置为“在创建者的进程中” 之后，会发现注册表中的 **Inproc** 服务器键已被添加，并指向 Microsoft Transaction Server 替代 .dll。

另请参阅 关于 Transaction Server 以及如何执行这些步骤的详细信息，请访问位于 http://www.microsoft.com/transaction/ 的 Transaction Server 站点或参考 Microsoft Transaction Server 文档。

1.1.9.6 Microsoft Internet Explorer 安全问题

使用 Microsoft® Internet Explorer 新加入的安全增强功能，一些 ADO 和 RDS 对象会被限制为只能在“安全”模式环境中运行。这需要用户知道这些问题，包括不同区域、安全等级、限制性行为、不安全操作和自定义安全设置。

有关这些问题的详细信息，请参阅在 Web 站点 www.microsoft.com/data/techmat.htm 上名为“Security Issues in Microsoft Internet Explorer”的文档。

1.1.10 ADO 词汇表

服务提供者(See 2.45)

单线程控件(See 2.46)

通讯模块(See 2.47)

图表(See 2.49)

variant(See 2.50)

Web 服务器(See 2.51)

1.2 Microsoft ADO Extensions for DDL and Security (ADOX) 程序员参考

Microsoft® ActiveX® Data Objects Extensions for Data Definition Language and Security (ADOX) 是对 ADO 对象和编程模型的扩展。ADOX 包括用于模式创建和修改的对象，以及安全性。由于它是基于对象实现模式操作，所以用户可以编写对各种数据源都能有效运行的代码，而与它们原始语法中的差异无关。

ADOX 是核心 ADO 对象的扩展库。它显露的其他对象可用于创建、修改和删除模式对象，如表格和过程。它还包括安全对象，可用于维护用户和组，以及授予和撤消对象的权限。

要通过开发工具使用 ADOX，需要建立对 ADOX 类型库的引用。对 ADOX 库的说明为“Microsoft ADO Ext. for DDL and Security.”。ADOX 库文件名为“Msadox.dll”，程序 ID (ProgID) 为“ADOX”。有关建立库引用的详细信息，请参阅开发工具的文档。

用于 Microsoft Jet Database Engine 的 Microsoft OLE DB Provider 完全支持 ADOX。取决于数据提供者，某些 ADOX 的功能可能不被支持。有关 Microsoft OLE DB Provider for ODBC、Microsoft OLE DB Provider for Oracle 或 Microsoft SQL Server OLE DB Provider 的被支持功能的详细信息，请参阅 ADOX 自述文件。

本文档假定读者懂得并能使用 Microsoft® Visual Basic® 编程语言，并了解 ADO 的基本知识。有关 ADO 的详细信息，请参阅 ADO 程序员参考(See 1.)。有关 ADOX 的概述信息，请参阅：

- ADOX 对象模型(See 1.2.1.1)
- ADOX 对象(See 1.2.1.2)
- ADOX 集合(See 1.2.1.3)
- ADOX 属性(See 1.2.1.5)
- ADOX 方法(See 1.2.1.4)
- ADOX 范例(See 1.2.1.6)

1.2.1 ADOX API 参考

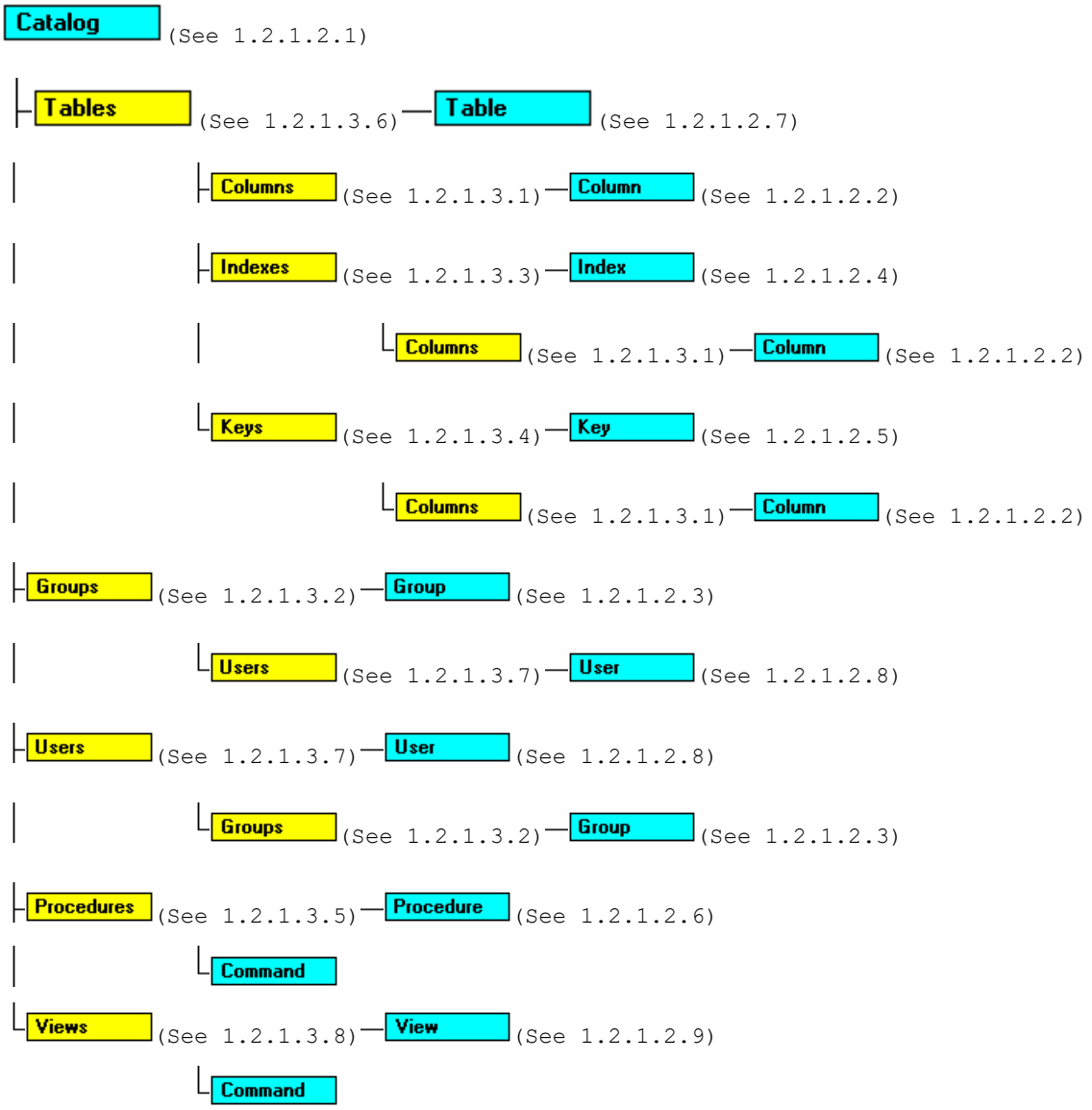
本节 ADOX 文档包含每个 ADOX 对象、集合、方法和属性的主题，以及相应的范例代码。详细信息，请在索引中搜索指定主题或参考如下主题：

- ADOX 对象(See 1.2.1.2)
- ADOX 集合(See 1.2.1.3)

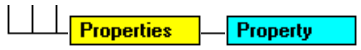
- ADOX 方法(See 1.2.1.4)
- ADOX 属性(See 1.2.1.5)
- ADOX 范例(See 1.2.1.6)

1.2.1.1 ADOX 对象模型

如下图表说明在 ADOX 中如何表示这些对象及其相互关系。有关指定对象或集合的详细信息，请参阅指定的参考主题，或 ADOX 对象(See 1.2.1.2)和 ADOX 集合(See 1.2.1.3)。



每个 Table(See 1.2.1.2.7)、Index(See 1.2.1.2.4) 和 Column(See 1.2.1.2.2) 对象均有标准的 ADO Properties(See 1.1.4.3.4) 集合。



1.2.1.2 ADOX 对象

ADOX 对象总结

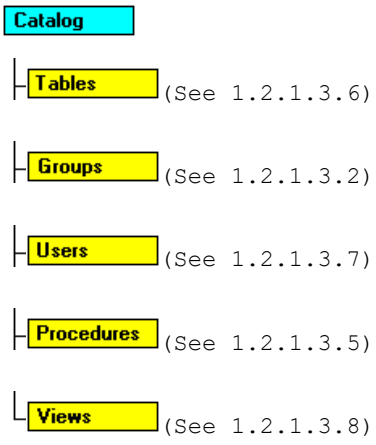
对象	说明
Catalog(See 1.2.1.2.1)	包含描述数据源模式目录的集合。
Column(See 1.2.1.2.2)	表示表、索引或关键字的列。
Group(See 1.2.1.2.3)	表示在安全数据库内有访问权限的组帐号。
Index(See 1.2.1.2.4)	表示数据库表中的索引。
Key(See 1.2.1.2.5)	表示数据库表中的主关键字、外部关键字或唯一关键字。
Procedure(See 1.2.1.2.6)	表示存储的过程。
Table(See 1.2.1.2.7)	表示数据库表，包括列、索引和关键字。
User(See 1.2.1.2.8)	表示在安全数据库内具有访问权限的用户帐号。
View(See 1.2.1.2.9)	表示记录或虚拟表的过滤集。

这些对象之间的关系以图的形式表示在 ADOX 对象模型(See 1.2.1.1)中。

每个对象均可以包含在其相应的集合中。例如，**Table** 对象可以包含在 Tables(See 1.2.1.3.6) 集合中。详细信息，请参阅 ADOX 集合(See 1.2.1.3)或指定的集合主题。

1.2.1.2.1 Catalog 对象 (ADOX)

包含描述数据源模式目录的集合（Tables(See 1.2.1.3.6)、Views(See 1.2.1.3.8)、Users(See 1.2.1.3.7)、Groups(See 1.2.1.3.2) 和 Procedures(See 1.2.1.3.5)）。



说明

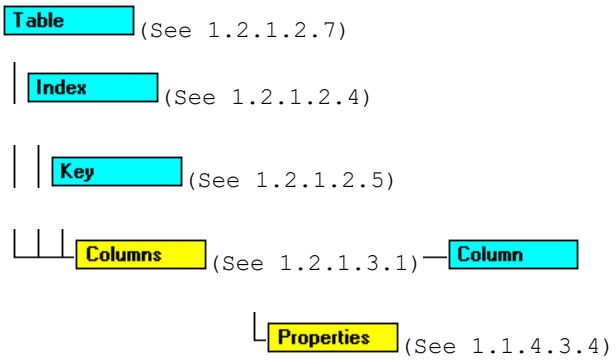
可以通过添加或删除对象、或修改现有的对象来修改 **Catalog** 对象。有些提供者可能不支持所有 **Catalog** 对象，或可能只支持查看模式信息。

使用 **Catalog** 对象的属性和方法，可以：

- 通过将 ActiveConnection(See 1.2.1.5.1) 属性设置为 ADO Connection(See 1.1.4.2.2) 对象或有效的连接字符串来打开目录。
- 使用 Create(See 1.2.1.4.10) 方法创建新目录。
- 使用 GetObjectOwner(See 1.2.1.4.12) 和 SetObjectOwner(See 1.2.1.4.14) 方法确定 **Catalog** 中对象的所有者。

1.2.1.2.2 Column 对象 (ADOX)

表示表、索引或关键字的列。



说明

如下代码创建新的 **Column**：

```
Dim obj As New Column
```

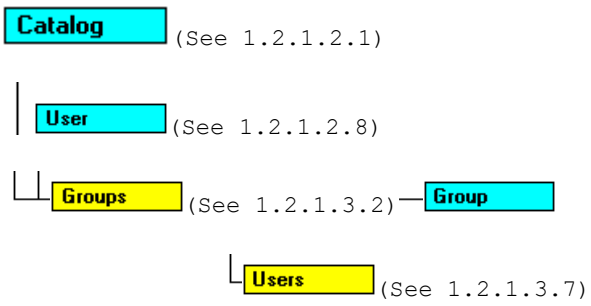
使用 **Column** 对象的属性和集合，可以：

- 使用 Name(See 1.2.1.5.10) 属性标识列。
- 使用 Type(See 1.2.1.5.18) 属性指定列的数据类型。
- 使用 Attributes(See 1.2.1.5.2) 属性确定是否列是固定长度或包含空值。
- 使用 DefinedSize(See 1.2.1.5.7) 属性指定列的最大大小。
- 对于数字数据值，使用 NumericScale(See 1.2.1.5.11) 方法指定范围。
- 对于数字数据值，使用 Precision(See 1.2.1.5.13) 属性指定最大精度。
- 使用 ParentCatalog(See 1.2.1.5.12) 属性指定拥有列的 Catalog(See 1.2.1.2.1)。

- 对于关键字列，使用 RelatedColumn(See 1.2.1.5.15) 属性指定在相关表中相关列的名称。
- 对于索引列，使用 SortOrder(See 1.2.1.5.17) 属性指定排序顺序是升序还是降序。
- 使用 Properties(See 1.1.4.3.4) 集合访问特定提供者的属性。

1.2.1.2.3 Group 对象 (ADOX)

表示在安全数据库内具有访问权限的组帐号。



说明

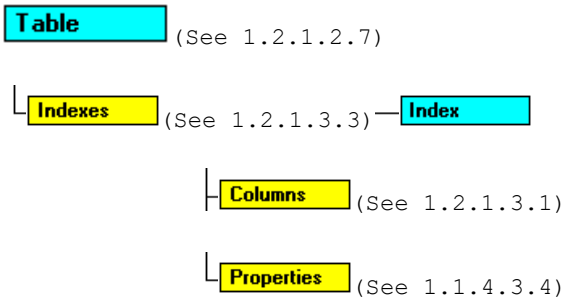
Catalog(See 1.2.1.2.1) 的 Groups(See 1.2.1.3.2) 集合代表所有目录的组帐号。User(See 1.2.1.2.8) 的 **Groups** 集合仅代表用户所属的组。

使用 **Group** 对象的属性、集合和方法，可以：

- 使用 Name(See 1.2.1.5.10) 属性标识组。
- 使用 GetPermissions(See 1.2.1.4.13) 和 SetPermissions(See 1.2.1.4.15) 方法，确定组是否有读、写或删除的权限。
- 使用 Users(See 1.2.1.3.7) 集合访问在组中具有成员身份的用户帐号。

1.2.1.2.4 Index 对象 (ADOX)

表示数据库表的索引。



说明

如下代码创建新的 **Index**：

```
Dim obj As New Index
```

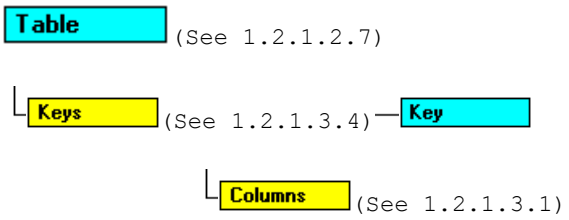
使用 **Index** 对象的属性和集合，可以：

- 使用 Name(See 1.2.1.5.10) 属性标识索引。
- 使用 Columns(See 1.2.1.3.1) 集合访问索引的数据库列。
- 使用 Unique(See 1.2.1.5.21) 属性指定索引关键字是否必须唯一。
- 使用 PrimaryKey(See 1.2.1.5.14) 属性指定索引是否为表的主关键字。
- 使用 IndexNulls(See 1.2.1.5.9) 属性指定索引字段中有 **Null** 值的记录是否有索引项目。
- 使用 Clustered(See 1.2.1.5.3) 属性指定索引是否分簇。
- 使用 ParentCatalog(See 1.2.1.5.12) 属性指定拥有索引的 **Catalog**。
- 使用 Properties(See 1.1.4.3.4) 集合访问特定提供者的索引属性。

注意 如果在已追加到 Tables(See 1.2.1.3.6) 集合的 Table(See 1.2.1.2.7) 对象中不存在 **Column**，则会在将 Column(See 1.2.1.2.2) 追加到 **Index** 的 **Columns** 集合时出现错误。

1.2.1.2.5 Key 对象 (ADOX)

表示数据库表中的主关键字、外部关键字或唯一关键字字段。



说明

如下代码创建新的 **Key**。

```
Dim obj As New Key
```

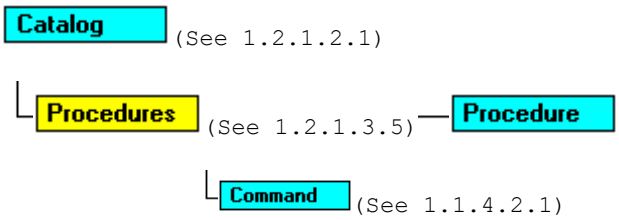
使用 **Key** 对象的属性和集合，可以：

- 使用 Name(See 1.2.1.5.10) 属性标识关键字。
- 使用 Type(See 1.2.1.5.19) 属性确定关键字是否为主关键字、外部关键字或唯一关键字。
- 使用 Columns(See 1.2.1.3.1) 集合访问关键字的数据库列。
- 使用 RelatedTable(See 1.2.1.5.16) 属性指定相关表的名称。

- 使用 DeleteRule(See 1.2.1.5.8) 和 UpdateRule(See 1.2.1.5.22) 属性确定在删除或更新主关键字时所执行的操作。

1.2.1.2.6 Procedure 对象 (ADOX)

表示存储过程。当连同 ADO Command(See 1.1.4.2.1) 对象使用时，**Procedure** 对象可用于添加、删除或修改存储过程。



说明

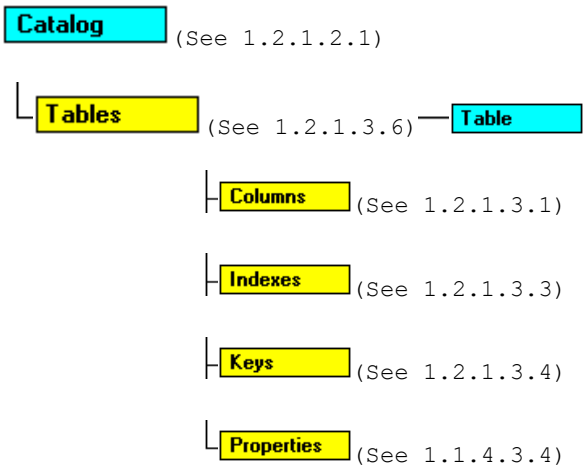
Procedure 对象允许创建存储过程，而无须知道或使用提供者的“CREATE PROCEDURE”语法。

使用 **Procedure** 对象的属性，可以：

- 使用 Name(See 1.2.1.5.10) 属性标识过程。
- 使用 Command(See 1.2.1.5.4) 属性指定可用于创建或执行该过程的 ADO **Command** 对象。
- 使用 DateCreated(See 1.2.1.5.5) 和 DateModified(See 1.2.1.5.6) 属性返回日期信息。

1.2.1.2.7 Table 对象 (ADOX)

表示包括列、索引和关键字的数据库表。



说明

如下代码创建新的 **Table**：

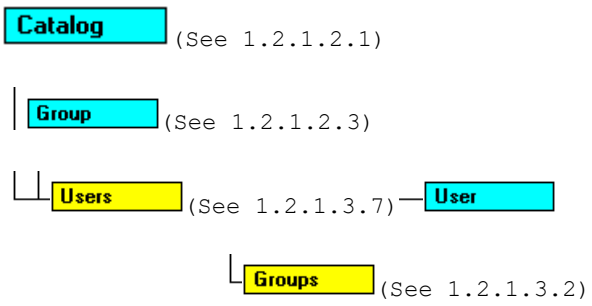
```
Dim obj As New Table
```

使用 **Table** 对象的属性和集合，可以：

- 使用 Name(See 1.2.1.5.10) 属性标识表。
- 使用 Type(See 1.2.1.5.20) 属性确定表的类型。
- 使用 Columns(See 1.2.1.3.1) 集合访问表的数据库列。
- 使用 Indexes(See 1.2.1.3.3) 集合访问表的索引。
- 使用 Keys(See 1.2.1.3.4) 集合访问表的关键字。
- 使用 ParentCatalog(See 1.2.1.5.12) 属性指定拥有表的 Catalog(See 1.2.1.2.1)。
- 使用 DateCreated(See 1.2.1.5.5) 和 DateModified(See 1.2.1.5.6) 属性返回日期信息。
- 使用 Properties(See 1.1.4.3.4) 集合访问特定提供者的表属性。

1.2.1.2.8 User 对象 (ADOX)

表示在安全数据库中具有访问权限的用户帐号。



说明

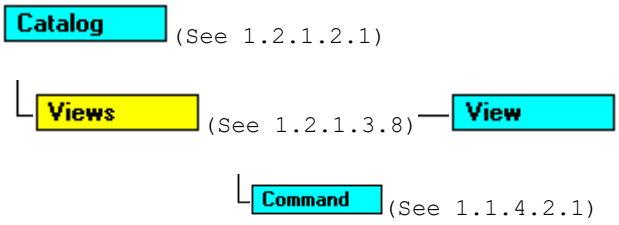
Catalog(See 1.2.1.2.1) 的 Users(See 1.2.1.3.7) 集合代表所有目录的用户。Group(See 1.2.1.2.3) 的 **Users** 集合只代表指定组的用户。

使用 **User** 对象的属性、集合和方法，可以：

- 使用 Name(See 1.2.1.5.10) 属性标识用户。
- 使用 ChangePassword(See 1.2.1.4.9) 方法更改用户的密码。
- 使用 GetPermissions(See 1.2.1.4.13) 和 SetPermissions(See 1.2.1.4.15) 方法确定用户拥有读、写或删除的权限。
- 使用 Groups(See 1.2.1.3.2) 集合访问用户所属的组。

1.2.1.2.9 View 对象 (ADOX)

表示记录或虚拟表的过滤集。当连同 ADO Command(See 1.1.4.2.1) 对象使用时，**View** 对象可用于添加、删除或修改视图。



说明

视图是虚拟表，是通过其它数据库表或视图创建的。**View** 对象允许创建视图，而无须知道或使用提供者的“CREATE VIEW”语法。

使用 **View** 对象的属性，可以：

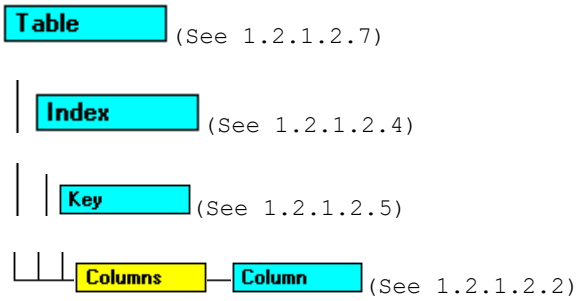
- 使用 Name(See 1.2.1.5.10) 属性标识视图。
- 使用 Command(See 1.2.1.5.4) 属性指定可用于添加、删除或修改视图的 ADO **Command** 对象。
- 使用 DateCreated(See 1.2.1.5.5) 和 DateModified(See 1.2.1.5.6) 属性返回日期信息。

1.2.1.3 ADOX 集合

集合	说明
Columns(See 1.2.1.3.1)	包含表、索引或关键字的所有 **Column** 对象。
Groups(See 1.2.1.3.2)	包含目录或用户的所有存储 **Group** 对象。
Indexes(See 1.2.1.3.3)	包含表的所有 **Index** 对象。
Keys(See 1.2.1.3.4)	包含表的所有 **Key** 对象。
Procedures(See 1.2.1.3.5)	包含目录的所有 **Procedure** 对象。
Tables(See 1.2.1.3.6)	包含目录的所有 **Table** 对象。
Users(See 1.2.1.3.7)	包含目录或组的所有存储 **User** 对象。
Views(See 1.2.1.3.8)	包含目录的所有 **View** 对象。

1.2.1.3.1 Columns 集合 (ADOX)

包含表、索引或关键字的所有 Column(See 1.2.1.2.2) 对象。



说明

Columns 集合的 **Append** 方法对于 ADOX 是唯一的。可以：

- 使用 Append(See 1.2.1.4.1) 方法将新列添加到集合中。

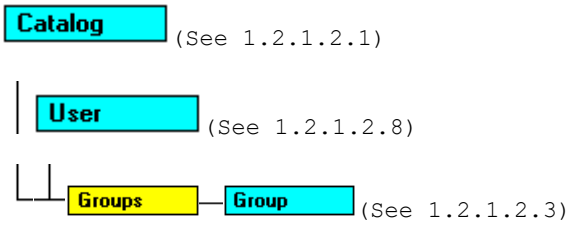
其余的属性和方法对于 ADO 集合是标准的。可以：

- 使用 Item(See 1.1.4.4.27) 方法访问集合中的列。
- 使用 Count(See 1.1.4.6.16) 属性返回包含在集合中的列的数目。
- 使用 Delete(See 1.2.1.4.11) 方法从集合中删除列。
- 使用 Refresh(See 1.1.4.4.36) 方法更新集合中的对象，以反映当前数据库的模式。

注意 如果 **Column** 尚未存在于已追加到 Tables(See 1.2.1.3.6) 集合的 Table(See 1.2.1.2.7) 中，则在将 **Column** 追加到 Index(See 1.2.1.2.4) 的 **Columns** 集合时会出现错误。

1.2.1.3.2 Groups 集合 (ADOX)

包含目录或用户的所有存储 Group(See 1.2.1.2.3) 对象。



说明

Catalog(See 1.2.1.2.1) 的 **Groups** 集合代表所有目录的组帐号。User(See 1.2.1.2.8) 的 **Groups** 集合只代表用户所属的组。

Groups 集合的 **Append** 方法对于 ADOX 是唯一的。可以：

- 使用 Append(See 1.2.1.4.2) 方法将新的安全组添加到集合中。

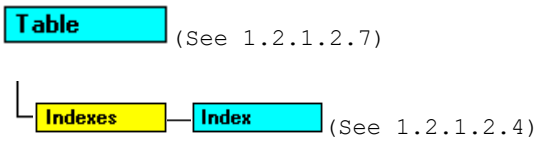
其余的属性和方法对于 ADO 集合是标准的。可以：

- 使用 Item(See 1.1.4.4.27) 方法访问集合中的组。
  - 使用 Count(See 1.1.4.6.16) 属性返回包含在集合中的列的数目。
  - 使用 Delete(See 1.2.1.4.11) 方法从集合中删除组。
  - 使用 Refresh(See 1.1.4.4.36) 方法更新集合中的对象，以反映当前数据库的模式。

说明 将 **Group** 对象追加到 **User** 对象的 **Group** 集合之前，与被追加对象具有相同 Name(See 1.3.5.5.13) 的 **Group** 对象必须已存在于 **Catalog** 的 **Groups** 集合中。

1.2.1.3.3 Indexes 集合 (ADOX)

包含表的所有 Index(See 1.2.1.2.4) 对象。



说明

Indexes 集合的 **Append** 方法对于 ADOX 是唯一的。可以：

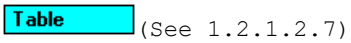
- 使用 Append(See 1.2.1.4.3) 方法将新索引添加到集合中。

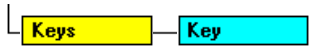
其余的属性和方法对于 ADO 集合是标准的。可以：

- 使用 Item(See 1.1.4.4.27) 方法访问集合中的索引。
- 使用 Count(See 1.1.4.6.16) 属性返回包含在集合中的索引的数目。
- 使用 Delete(See 1.2.1.4.11) 方法从集合中删除索引。
- 使用 Refresh(See 1.1.4.4.36) 方法更新集合中的对象，以反映当前数据库的模式。

1.2.1.3.4 Keys 集合 (ADOX)

包含表的所有 Key(See 1.2.1.2.5) 对象。





(See 1.2.1.2.5)

说明

Keys 集合的 **Append** 方法对于 ADOX 是唯一的。可以：

- 使用 Append(See 1.2.1.4.4) 方法将新关键字添加到集合中。

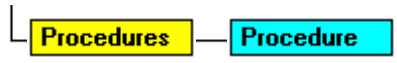
其余的属性和方法对于 ADO 集合是标准的。可以：

- 使用 Item(See 1.1.4.4.27) 方法访问集合中的关键字。
- 使用 Count(See 1.1.4.6.16) 属性返回包含在集合中的关键字的数目。
- 使用 Delete(See 1.2.1.4.11) 方法从集合中删除关键字。
- 使用 Refresh(See 1.1.4.4.36) 方法更新集合中的对象，以反映当前数据库的模式。

1.2.1.3.5 Procedures 集合 (ADOX)

包含目录的所有 Procedure(See 1.2.1.2.6) 对象。

(See 1.2.1.2.1)



(See 1.2.1.2.6)

说明

Procedures 集合的 **Append** 方法对于 ADOX 是唯一的。可以：

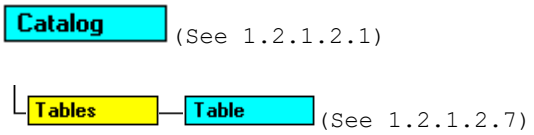
- 使用 Append(See 1.2.1.4.5) 方法将新过程添加到集合中。

其余的属性和方法对于 ADO 集合是标准的。可以：

- 使用 Item(See 1.1.4.4.27) 方法访问集合中的过程。
- 使用 Count(See 1.1.4.6.16) 属性返回包含在集合中的过程的数目。
- 使用 Delete(See 1.2.1.4.11) 方法从集合中删除过程。
- 使用 Refresh(See 1.1.4.4.36) 方法更新集合中的对象，以反映当前数据库的模式。

1.2.1.3.6 Tables 集合 (ADOX)

包含目录的所有 Table(See 1.2.1.2.7) 对象。



说明

Tables 集合的 **Append** 方法对于 ADOX 是唯一的。可以：

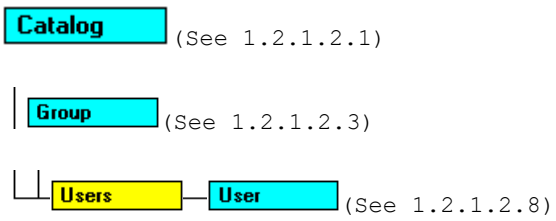
- 使用 Append(See 1.2.1.4.6) 方法将新表添加到集合中。

其余的属性和方法对于 ADO 集合是标准的。可以：

- 使用 Item(See 1.1.4.4.27) 方法访问集合中的表。
- 使用 Count(See 1.1.4.6.16) 属性返回包含在集合中的表的数目。
- 使用 Delete(See 1.2.1.4.11) 方法从集合中删除表。
- 使用 Refresh(See 1.1.4.4.36) 方法更新集合中的对象，以反映当前数据库的模式。

1.2.1.3.7 Users 集合 (ADOX)

包含目录或组的所有存储 User(See 1.2.1.2.8) 对象。



说明

Catalog(See 1.2.1.2.1) 的 **Users** 集合代表所有目录的用户。Group(See 1.2.1.2.3) 的 **Users** 集合仅代表具有特定组成员身份的用户。**Users** 集合的 **Append** 方法对于 ADOX 是唯一的。可以：

- 使用 Append(See 1.2.1.4.7) 方法将新用户添加到集合中。

其余的属性和方法对于 ADO 集合是标准的。可以：

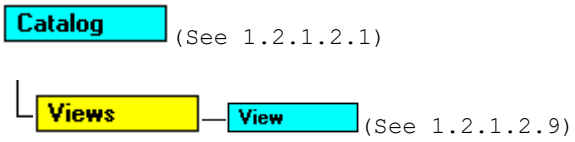
- 使用 Count(See 1.1.4.6.16) 属性返回包含在集合中的用户的数目。

- 使用 Item(See 1.1.4.4.27) 方法访问集合中的用户。
- 使用 Delete(See 1.2.1.4.11) 方法从集合中删除用户。
- 使用 Refresh(See 1.1.4.4.36) 方法更新集合中的对象，以反映当前数据库的模式。

注意 在将 **User** 对象追加到 **Group** 对象的 **Users** 集合之前，与被追加对象具有相同 Name(See 1.3.5.5.13) 的 **User** 对象必须已存在于 **Catalog** 的 **Users** 集合中。

1.2.1.3.8 Views 集合 (ADOX)

包含目录的所有 View(See 1.2.1.2.9) 对象。



说明

Views 集合的 **Append** 方法对于 ADOX 是唯一的。可以：

- 使用 Append(See 1.2.1.4.8) 方法将新视图添加到集合中。

其余的属性和方法对于 ADO 集合是标准的。可以：

- 使用 Item(See 1.1.4.4.27) 方法访问集合中的视图。
- 使用 Count(See 1.1.4.6.16) 属性返回包含在集合中的视图的数目。
- 使用 Delete(See 1.2.1.4.11) 方法从集合中删除视图。
- 使用 Refresh(See 1.1.4.4.36) 方法更新集合中的对象，以反映当前数据库的模式。

1.2.1.4 ADOX 方法

方法	说明
Append(Columns)(See 1.2.1.4.1)	将新的 **Column** 对象添加到 **Columns** 集合。
Append(Groups)(See 1.2.1.4.2)	将新的 **Group** 对象添加到 **Groups** 集合。
Append(Indexes)(See 1.2.1.4.3)	将新的 **Index** 对象添加到 **Indexes** 集合。
Append(Keys)(See 1.2.1.4.4)	将新的 **Key** 对象添加到 **Keys** 集合。
Append(Procedures)(See	将新的 **Procedure** 对象添加到 **Procedures** 集合。

1.2.1.4.5)	
Append（Tables）(See 1.2.1.4.6)	将新的 **Table** 对象添加到 **Tables** 集合。
Append（Users）(See 1.2.1.4.7)	将新的 **User** 对象添加到 **Users** 集合。
Append（Views）(See 1.2.1.4.8)	将新的 **View** 对象添加到 **Views** 集合。
ChangePassword(See 1.2.1.4.9)	更改用户帐号的密码。
Create(See 1.2.1.4.10)	创建新的目录。
Delete(See 1.2.1.4.11)	删除集合中的对象。
GetObjectOwner(See 1.2.1.4.12)	返回目录中对象的拥有者。
GetPermissions(See 1.2.1.4.13)	获得对象上组或用户的权限。
Item(See 1.1.4.4.27)	按名称或序号返回集合的指定成员。
Refresh(See 1.1.4.4.36)	更新集合中的对象，以反映针对提供者可用的和指定的对象。
SetObjectOwner(See 1.2.1.4.14)	指定目录中对象的拥有者。
SetPermissions(See 1.2.1.4.15)	设置对象上组或用户的权限。

1.2.1.4.1 Append 方法 (ADOX Columns)

将新的 Column(See 1.2.1.2.2) 对象添加到 Columns(See 1.2.1.3.1) 集合。

语法

Columns.**Append** *Column* [, *Type*] [, *DefinedSize*]

参数

Column 要追加的 **Column** 对象，或要创建和追加的列的名称。

Type 可选，长整型值。指定列的数据类型。*Type* 参数与 **Column** 对象的 Type(See 1.2.1.5.18) 属性相对应。

DefinedSize 可选，长整型值。指定列的大小。*DefinedSize* 参数与 **Column** 对象的 DefinedSize(See 1.2.1.5.7) 属性相对应。

> **注意** 如果在已追加到 Tables(See 1.2.1.3.6) 集合的 Table(See 1.2.1.2.7) 中还没有 **Column**，则会在将 **Column** 追加到 Index(See 1.2.1.2.4) 的 **Columns** 集合时出现错误。

1.2.1.4.2 Append 方法 (ADOX Groups)

将新的 Group(See 1.2.1.2.3) 对象添加到 Groups(See 1.2.1.3.2) 集合。

语法

Groups.**Append** *Group*

参数

Group 要追加的 **Group** 对象，或要创建和追加的组的名称。

说明

Catalog(See 1.2.1.2.1) 的 **Groups** 集合代表所有目录的组帐号。User(See 1.2.1.2.8) 的 **Groups** 集合仅代表用户所属的组。
如果提供者不支持创建组，将出现错误。

> **说明** 在将 **Group** 对象追加到 **User** 对象的 **Groups** 集合之前，与被追加的对象具有相同 Name(See 1.3.5.5.13) 的 **Group** 对象必须已存在于 **Catalog** 的 **Groups** 集合中。

1.2.1.4.3 Append 方法 (ADOX Indexes)

将新的 Index(See 1.2.1.2.4) 对象添加到 Indexes(See 1.2.1.3.3) 集合。

语法

Indexes.**Append** *Index* [, *Columns*]

参数

Index 要追加的 **Index** 对象，或要创建和追加的索引的名称。
Columns 可选，变体型值。指定要进行索引的列的名称。*Columns* 参数与 Column(See 1.2.1.2.2) 对象的 Name(See 1.2.1.5.10) 属性值相对应。

说明

Columns 参数可使用列名或列名数组。
如果提供者不支持创建索引，将出现错误。

1.2.1.4.4 Append 方法 (ADOX Keys)

将新的 Key(See 1.2.1.2.5) 对象添加到 Keys(See 1.2.1.3.4) 集合。

语法

Keys.**Append** *Key* [, *KeyType*] [, *Column*] [, *RelatedTable*]

参数

Key 要追加的 **Key** 对象，或要创建和追加的关键字的名称。

KeyType 可选，长整型值。指定关键字类型。*Key* 参数与 **Key** 对象的 Type(See 1.2.1.5.19) 属性相对应。

Column 可选，字符串值。指定进行索引的列的名称。*Columns* 参数与 Column(See 1.2.1.2.2) 对象的 Name(See 1.2.1.5.10) 属性值相对应。

RelatedTable 可选，字符串值。指定有关的表的名称。*RelatedTable* 参数与 Table(See 1.2.1.2.7) 对象的 Name(See 1.2.1.5.10) 属性值相对应。

说明

Columns 参数可使用列名或列名数组。

1.2.1.4.5 Append 方法 (ADOX Procedures)

将新的 Procedure(See 1.2.1.2.6) 对象添加到 Procedures(See 1.2.1.3.5) 集合。

语法

Procedures.**Append** *Name*, *Command*

参数

Name 字符串值。指定要创建和追加的过程的名称。

Command ADO Command(See 1.1.4.2.1) 对象。表示要创建和追加的过程。

说明

使用 **Command** 对象中指定的名称和属性，在数据源中创建新的过程。

如果用户指定的命令文本代表的是视图而不是过程，将出现错误，并且命令不会保存在数据源中或添加到集合中。

如果提供者不支持持久命令，**Append** 将失败。

1.2.1.4.6 Append 方法 (ADOX Tables)

将新的 Table(See 1.2.1.2.7) 对象添加到 Tables(See 1.2.1.3.6) 集合。

语法

Tables.**Append** *Table*

参数

Table 变体型值。包含对要追加的 **Table** 的引用，或要创建和追加的表的名称。

说明

如果提供者不支持创建表，将出现错误。

1.2.1.4.7 Append 方法 (ADOX Users)

将新的 User(See 1.2.1.2.8) 对象添加到 Users(See 1.2.1.3.7) 集合。

语法

Users.**Append** *User*[, *Password*]

参数

User 变体型值。包含要追加的 **User** 对象，或要创建和追加的用户的名称。

Password 可选，字符串值。包含用户的密码。*Password* 参数对应于 **User** 对象的 ChangePassword(See 1.2.1.4.9) 方法所指定的值。

说明

Catalog(See 1.2.1.2.1) 的 **Users** 集合代表所有目录的用户。Group(See 1.2.1.2.3) 的 **Users** 集合仅代表在特定组中具有成员身份的用户。

如果提供者不支持创建用户，将出现错误。

> **说明** 在将 **User** 对象追加到 **Group** 对象的 **Users** 集合之前，与被追加的对象具有相同 Name(See 1.3.5.5.13) 的 **User** 对象必须已存在于 **Catalog** 的 **Users** 集合中。

1.2.1.4.8 Append 方法 (ADOX Views)

将新的 View(See 1.2.1.2.9) 对象添加到 Views(See 1.2.1.3.8) 集合。

语法

Views.**Append** *Name*, *Command*

参数

Name 字符串值。指定要创建的视图的名称。

Command ADO Command(See 1.1.4.2.1) 对象。代表要创建的视图。

说明

使用 **Command** 对象中指定的名称和属性，在数据源中创建新的视图。

如果用户指定的命令文本代表过程而不是视图，将出现错误，并且该命令不会保存在数据源中，或添加到集合中。

如果提供者不支持持久命令，**Append** 将失败。

1.2.1.4.9 ChangePassword 方法 (ADOX)

更改用户帐号的密码。

语法

User.**ChangePassword** *OldPassword*, *NewPassword*

参数

OldPassword 字符串值。指定用户现有密码。如果用户当前没有密码，请对 *OldPassword* 使用空字符串 ("")。

NewPassword 字符串值。指定新的密码。

说明

由于安全原因，除新密码外必须指定旧密码。

如果提供者不支持受托者属性的管理，将出现错误。

1.2.1.4.10 Create 方法 (ADOX)

创建新目录。

语法

Catalog.**Create** *ConnectString*

参数

ConnectString 字符串值。用于连接数据源。

说明

Create 方法创建并打开在 *ConnectString* 中指定的数据源的新 ADO Connection(See 1.1.4.2.2)。如果成功，新的 **Connection** 对象将被赋给 ActiveConnection(See 1.2.1.5.1) 属性。

如果提供者不支持创建新目录，将出现错误。

1.2.1.4.11 Delete 方法 (ADOX 集合)

从集合中删除对象。

语法

Collection.**Delete** *Name*

参数

Name 变体型。指定要删除的对象的名称或序号位置（索引）。

说明

如果集合中不存在 *Name*，将出现错误。

对于 Tables(See 1.2.1.3.6) 和 Users(See 1.2.1.3.7) 集合，如果提供者不支持各自删除表或用户，将出现错误。对于 Procedures(See 1.2.1.3.5) 和 Views(See 1.2.1.3.8) 集合，如果提供者不支持持久命令，删除将失败。

1.2.1.4.12 GetObjectOwner 方法 (ADOX)

返回 Catalog(See 1.2.1.2.1) 中对象的拥有者。

语法

Owner = *Catalog*.**GetObjectOwner(***ObjectName***,** *ObjectType* [**,** *ObjectTypeId*]**)**

返回值

返回字符串值，该值指定拥有对象的 **User** 或 **Group** 的 **Name**。

参数

ObjectName 字符串值。指定返回其拥有者的对象的名称。

ObjectType 枚举型值。指定得到其拥有者的对象类型。下列常量是 *ObjectType* 的有效值：

常量	说明
adPermObjProviderSpecific	对象属于提供者定义的类型。如果 *ObjectType* 是 **adPermObjProviderSpecific**，并且未提供 *ObjectTypeId*，将出现错误。
adPermObjTable	对象是表。
adPermObjColumn	对象是列。
adPermObjDatabase	对象是数据库。
adPermObjProcedure	对象是过程。
adPermObjView	对象是视图。
adPermObjSchema	对象是模式。
adPermObjDomain	对象是域。
adPermObjCollation	对象是序列。

adPermObjSchemaRowset	对象是模式行集合。
adPermObjCharacterSet	对象是字符集。
adPermObjTranslation	对象是转换。

ObjectTypeId 可选，变体型值。指定 OLE DB 规范未定义的提供者对象类型的 GUID。如果 *ObjectType* 设置为 **adPermObjProviderSpecific**，则需要该参数；否则，将不使用它。

说明

如果提供者不支持返回对象拥有者，将出现错误。

1.2.1.4.13 GetPermissions 方法 (ADOX)

获得对象上组或用户的权限。

语法

ReturnValue = *GroupOrUser*.**GetPermissions(***Name***,** *ObjectType*
[**,** *ObjectTypeId*]**)**

返回值

返回的长整型值，该值指定包含组或用户对对象具有的权限的位掩码。

参数

Name 字符串值。指定用于设置权限的对象的名称。

ObjectType 长整型值。指定用于得到权限的对象类型。下列常量是 *ObjectType* 的有效值：

常量	说明
adPermObjProviderSpecific	对象是提供者定义的类型。如果 *ObjectType* 是 **adPermObjProviderSpecific**，并且未提供 *ObjectTypeId*，将出现错误。
adPermObjTable	对象是表。
adPermObjColumn	对象是列。
adPermObjDatabase	对象是数据库。
adPermObjProcedure	对象是过程。
adPermObjView	对象是视图。
adPermObjSchema	对象是模式。

adPermObjDomain	对象是域。
adPermObjCollation	对象是序列。
adPermObjSchemaRowset	对象是模式行集合。
adPermObjCharacterSet	对象是字符集。
adPermObjTranslation	对象是转换。

ObjectTypeId 可选，变体型值。指定 OLE DB 规范未定义的提供者对象类型的 GUID。如果 *ObjectType* 设置为 **adPermObjProviderSpecific**，则需要该参数；否则，将不使用它。

1.2.1.4.14 SetObjectOwner 方法 (ADOX)

指定 **Catalog** 中对象的拥有者。

语法

Catalog.**SetObjectOwner** *ObjectName*, *ObjectType* , *OwnerName* [, *ObjectTypeId*]

参数

ObjectName 字符串值。指定需指定其拥有者的对象的名称。

ObjectType 枚举型值。指定要指定其拥有者的对象类型。下列常量是 *ObjectType* 的有效值：

常量	说明
adPermObjProviderSpecific	对象是提供者定义的类型。如果 *ObjectType* 是 **adPermObjProviderSpecific**，并且未提供 *ObjectTypeId*，将出现错误。
adPermObjTable	对象是表。
adPermObjColumn	对象是列。
adPermObjDatabase	对象是数据库。
adPermObjProcedure	对象是过程。
adPermObjView	对象是视图。
adPermObjSchema	对象是模式。
adPermObjDomain	对象是域。
adPermObjCollation	对象是序列。
adPermObjSchemaRowset	对象是模式行集合。

adPermObjCharacterSet	对象是字符集。
adPermObjTranslation	对象是转换。

OwnerName 字符串值。指定将拥有对象的 **User** 或 **Group** 的名称。

ObjectTypeId 可选，变体型值。指定 OLE DB 规范未定义的提供者对象类型的 GUID。如果 *ObjectType* 设置为 **adPermObjProviderSpecific**，则需要该参数；否则，将不使用它。

说明

如果提供者不支持指定对象的拥有者，将出现错误。

1.2.1.4.15 SetPermissions 方法 (ADOX)

设置对象上组或用户的权限。

语法

GroupOrUser.**SetPermissions** *Name*, *ObjectType*, *Action*, *Rights* [, *Inherit*]
[, *ObjectTypeId*]

参数

Name 字符串值。指定需设置其权限的对象的名称。

ObjectType 长整型值。指定需得到其权限的对象的类型。下列常量是 *ObjectType* 的有效值：

常量	说明
adPermObjProviderSpecific	对象是提供者定义的类型。如果 *ObjectType* 是 **adPermObjProviderSpecific**，并且未提供 *ObjectTypeId*，将出现错误。
adPermObjTable	对象是表。
adPermObjColumn	对象是列。
adPermObjDatabase	对象是数据库。
adPermObjProcedure	对象是过程。
adPermObjView	对象是视图。
adPermObjSchema	对象是模式。
adPermObjDomain	对象是域。
AdPermObjCollation	对象是序列。
AdPermObjSchemaRowset	对象是模式行集合。

adPermObjCharacterSet	对象是字符集。
adPermObjTranslation	对象是转换。

Action 长整型值。指定设置权限时所执行的操作。下列常量是 *Action* 的有效值：

常量	说明
adAccessGrant	组或用户将至少拥有所请求的权限。
adAccessSet	组或用户恰好具有所请求的权限。
adAccessDeny	组或用户被拒绝所指定的权限。
adAccessRevoke	组或用户具有的任何显式访问权限将被撤消。
adAccessAuditSuccess	当成功打开使用所请求的权限的对象时，将对组进行审核。
AdAccessAuditFailure	当未能打开使用所请求的权限的对象时，将对组进行审核。

Rights 长整型值。包含指示要设置的权限的位掩码。值可以组合。下列常量是 *Rights* 的有效值：

常量	说明
adRightExecute	组具有执行对象的权限。
adRightRead	组具有读取对象的权限。
adRightUpdate	组具有更新对象的权限。
adRightInsert	组具有插入对象的权限。
adRightDelete	组具有删除对象的权限。
adRightReference	组具有引用对象的权限。
adRightCreate	组具有创建对象的权限。
adRightWithGrant	组具有授予对象上权限的权限。
adRightDesign	组具有设计对象的权限。
adRightAll	组具有所有对象上的权限。

Inherit 可选，枚举型值。指示对象如何继承这些权限。默认值为 **adInheritNone**。下列常量是 *Inherit* 的有效值：

常量	说明
adInheritNone	无继承。
adInheritObjects	容器中的无容器对象继承权限。
adInheritContainers	主对象包含的其它容器继承输入项。

adInheritBoth	主对象包含的对象和其它容器继承输入项。
adInheritNoPropagate	**adInheritObjects** 和 **adInheritContainers** 标志不传递给继承输入项。
adInheritOnly	权限不应用于访问控制列表 (ACL) 与之相连的主对象，但主对象包含的对象继承输入项。

ObjectTypeId 可选，变体型值。指定 OLE DB 规范未定义的提供者对象类型的 GUID。如果 *ObjectType* 设置为 **adPermObjProviderSpecific**，则需要该参数；否则，将不使用它。

说明

如果提供者不支持为组或用户设置访问权限，将出现错误。

1.2.1.5 ADOX 属性

属性	说明
ActiveConnection(See 1.2.1.5.1)	指示目录所属的 ADO **Connection** 对象。
Attributes(See 1.2.1.5.2)	描述列特性。
Clustered(See 1.2.1.5.3)	指示索引是否被分簇。
Command(See 1.2.1.5.4)	指定可用于创建或执行过程的 ADO **Command** 对象。
Count(See 1.1.4.6.16)	指示集合中的对象数量。
DateCreated(See 1.2.1.5.5)	指示创建对象的日期。
DateModified(See 1.2.1.5.6)	指示上一次更改对象的日期。
DefinedSize(See 1.2.1.5.7)	指示列的规定最大大小。
DeleteRule(See 1.2.1.5.8)	指示主关键字被删除时将执行的操作。
IndexNulls(See 1.2.1.5.9)	指示在索引字段中有 **Null** 值的记录是否有索引项。
Name(See 1.2.1.5.10)	指示对象的名称。
NumericScale(See 1.2.1.5.11)	指示列中数值的范围。
ParentCatalog(See 1.2.1.5.12)	指定表或列的父目录以便访问特定提供者的属性。
Precision(See 1.2.1.5.13)	指示列中数据值的最高精度。

PrimaryKey(See 1.2.1.5.14)	指示索引是否代表表的主关键字。
RelatedColumn(See 1.2.1.5.15)	指示相关表中相关列的名称（仅关键字列）。
RelatedTable(See 1.2.1.5.16)	指示相关表的名称。
SortOrder(See 1.2.1.5.17)	指示列的排序顺序（仅索引列）。
Type(列)(See 1.2.1.5.18)	指示列的数据类型。
Type（关键字）(See 1.2.1.5.19)	指示关键字的数据类型。
Type（表）(See 1.2.1.5.20)	指示表的类型。
Unique(See 1.2.1.5.21)	指示索引关键字是否必须是唯一的。
UpdateRule(See 1.2.1.5.22)	指示主关键字被更新时会执行的操作。

1.2.1.5.1 ActiveConnection 属性 (ADOX)

指示目录(See 1.2.1.2.1)所属的 ADO Connection(See 1.1.4.2.2) 对象。

设置和返回值

设置包含连接定义的 **Connection** 对象或字符串。返回活动 **Connection** 对象。

说明

默认值是 **Null** 对象引用。

1.2.1.5.2 Attributes 属性 (ADOX)

描述列特性。

设置和返回值

设置或返回长整型值。该值指定由 Column(See 1.2.1.2.2) 对象代表的表的特性，并可以是如下常量的组合：

常量	说明
adColFixed	列长度是固定的。
AdColNullable	列可以包含 null 值。

说明

默认值是零（0），既不是 **adColFixed** 也不是 **adColNullable**。

1.2.1.5.3 Clustered 属性 (ADOX)

指示索引是否被分簇。

设置和返回值

设置和返回布尔值。

说明

默认值是 **False**。
该属性在已追加到集合的 Index(See 1.2.1.2.4) 对象上是只读的。

1.2.1.5.4 Command 属性 (ADOX)

指定可用于创建或执行过程的 ADO Command(See 1.1.4.2.1) 对象。

设置和返回值

设置或者返回有效的 ADO **Command** 对象。

说明

如果**提供者**不支持持久命令，当获得并设置该属性时将发生错误。

1.2.1.5.5 DateCreated 属性 (ADOX)

指示创建对象的日期。

返回值

返回变体型值，指定创建日期。如果**提供者**不支持 **DateCreated**，那么该值将为 **Null**。

1.2.1.5.6 DateModified 属性 (ADOX)

指示上次修改对象的日期。

返回值

返回指定修改日期的变体型值。如果提供者不支持 **DateModified**，那么该值将为 **Null**。

1.2.1.5.7 DefinedSize 属性 (ADOX)

指示列的规定最大大小。

设置和返回值

设置并返回长整型值，该值是数据值按字符计算的最大长度。

说明

默认值是零（0）。

对于已追加到集合中的 Column(See 1.2.1.2.2) 对象，该属性是只读的。

1.2.1.5.8 DeleteRule 属性 (ADOX)

指示主关键字被删除时将执行的操作。

设置和返回值

设置并返回长整型值，该值为如下常量之一：

常量	说明
adRINone	无操作。
AdRICascade	层叠更改。
AdRISetNull	将外部关键字值设置为 null。
adRISetDefault	将外部关键字值设置为默认值。

说明

默认值是 **adRINone**。

该属性在已追加到集合中的 Key(See 1.2.1.2.5) 对象上是只读的。

1.2.1.5.9 IndexNulls 属性 (ADOX)

指示在索引字段中具有 **Null** 值的记录是否有索引项。

设置和返回值

设置并返回枚举型值，该值为如下常量之一：

常量	说明
adIndexNullsDisallow	索引不允许存在关键字列为 **Null** 的项目。如果在关键字列中输入 **Null** 值，将会发生错误。
AdIndexNullsIgnore	索引没有插入包含 **Null** 关键字的项目。如果在关键字列中输入 **Null** 值，则会忽略该项目，但不会发生错误。
AdIndexNullsIgnoreAny	索引没有在某个关键字列有 **Null** 值的地方插入项目。对于具有多列关键字的索引，如果在某个列中输入 **Null** 值，则将忽略该项目，但不会发生错误。

说明

默认值是 **adIndexNullsDisallow**。

该属性在已追加到集合中的 Index(See 1.2.1.2.4) 对象上是只读的。

1.2.1.5.10 Name 属性 (ADOX)

指示对象的名称。

设置和返回值

设置或者返回字符串值。

说明

在集合内名称不必唯一。

在 Column(See 1.2.1.2.2)、Group(See 1.2.1.2.3)、Key(See 1.2.1.2.5)、Index(See 1.2.1.2.4)、 Table(See 1.2.1.2.7) 和 User(See 1.2.1.2.8) 对象上，**Name** 属性是可读/写的。在 Catalog(See 1.2.1.2.1)、Procedure(See 1.2.1.2.6) 和 View(See 1.2.1.2.9) 对象上，**Name** 属性是只读的。

对于读/写对象（**Column**、**Group**、**Key**、**Index**、**Table** 和 **User** 对象），默认值是空字符串 ("")。

注意

- 对于关键字，该属性在已追加到集合中的 **Key** 对象上是只读的。
- 对于表，该属性对于已追加到集合中的 **Table** 对象是只读的。

1.2.1.5.11 NumericScale 属性 (ADOX)

指示列中数值的范围。

设置和返回值

设置并返回字节值，该值是当 Type(See 1.2.1.5.18) 属性为 **adNumeric** 或 **adDecimal** 时列中数值的范围。对于所有其他数据类型，将忽略 **NumericScale**。

说明

默认值是零（0）。
对于已追加到集合中的 Column(See 1.2.1.2.2) 对象，**NumericScale** 是只读的。

1.2.1.5.12 ParentCatalog 属性 (ADOX)

指定表或列的父目录以便访问特定提供者的属性。

设置和返回值

设置并返回 Catalog(See 1.2.1.2.1) 对象。将 **ParentCatalog** 设置为打开的 **Catalog** 将允许在将表或列追加到 **Catalog** 集合之前，访问特定**提供者**的属性。

说明

有些数据提供者只在创建时（当表或列被追加到它的 **Catalog** 集合时）允许写入特定提供者的属性值。要在将这些对象追加到 **Catalog** 之前访问这些属性，请先在 **ParentCatalog** 属性中指定 **Catalog**。
将表或列追加到非 **ParentCatalog** 的 **Catalog** 时，将发生错误。

1.2.1.5.13 Precision 属性 (ADOX)

指示列中数值的最高精度。

设置和返回值

设置并返回长整型值，该值是当 Type(See 1.2.1.5.18) 属性是数值型时列中数值的最高精度。对于所有其他数据类型，将忽略 **Precision**。

说明

默认值是零（0）。
对于已追加到集合的 Column(See 1.2.1.2.2) 对象，该属性是只读的。

1.2.1.5.14 PrimaryKey 属性 (ADOX)

指示索引是否代表表的主关键字。

设置和返回值

设置并返回布尔值。

说明

默认值是 **False**。

该属性在已追加到集合中的 Index(See 1.2.1.2.4) 对象上是只读的。

1.2.1.5.15 RelatedColumn 属性 (ADOX)

指示相关表中相关列的名称（仅关键字列）。

设置和返回值

设置并返回长整型值，该值是相关表中相关列的名称。

说明

默认值是空字符串 ("")。

对于已追加到集合中的 Column(See 1.2.1.2.2) 对象，该属性是只读的。

1.2.1.5.16 RelatedTable 属性 (ADOX)

指示相关表的名称。

设置和返回值

设置并返回字符串值。

说明

默认值是空字符串 ("")。

如果关键字是外部关键字，那么 **RelatedTable** 是包含该关键字的表的名称。

1.2.1.5.17 SortOrder 属性 (ADOX)

指示列的排序顺序（仅索引列）。

设置和返回值

设置并返回长整型值，该值是如下常量之一：

常量	说明

adSortAscending	列的排序顺序为升序。
AdSortDescending	列的排序顺序为降序。

说明

默认值是 **adSortAscending**。

该属性仅应用于在 Index(See 1.2.1.2.4) 的 Columns(See 1.2.1.3.1) 集合中的 Column(See 1.2.1.2.2) 对象。

1.2.1.5.18 Type 属性（列）(ADOX)

指示列的数据类型

设置和返回值

设置或返回长整型值，该值是如下常量之一：

常量	说明
adTinyInt	精确的数字值，精度为小数点后 3 位。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdSmallInt	精确数字值，精度为小数点后 5 位。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdInteger	精确数字值，精度为小数点后 10 位。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdBigInt	精确数字值，精度为小数点后 19 位。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdUnsignedTinyInt	无符号的 **adTinyInt**。
AdUnsignedSmallInt	无符号的 **adSmallInt**。
AdUnsignedInt	无符号的 **adInteger**。
AdUnsignedBigInt	无符号的 **adBigInt**。
AdSingle	单精度浮点数。
AdDouble	双精度浮点数。
AdCurrency	货币类型。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdDecimal	变体型十进制类型。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdNumeric	数值类型。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。

AdBoolean	变体布尔类型。0 为假而 ~0 为真。
AdUserDefined	用户定义的变量长度数据类型。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdVariant	自动变体型。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdGuid	全域唯一标识符。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdDate	自动日期。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdDBDate	数据库日期数据结构。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdDBTime	数据库时间数据结构。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdDBTimestamp	数据库时间戳结构。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdBSTR	BSTR 的指针。关于该类型的详细资料，请参阅“OLE DB 程序员参考”。
AdChar	定长字符串。
AdVarChar	变长字符串。
AdLongVarChar	长变长字符串。
AdWChar	宽定长字符串。
AdVarWChar	宽变长字符串。
AdLongVarWChar	长、宽变长字符串。
AdBinary	定长二进制数据。
AdVarBinary	变长二进制数据。
AdLongVarBinary	长变长二进制数据。

说明

默认值是 **adVarWChar**。

在 Column(See 1.2.1.2.2) 对象追加到集合或到另一个对象之前，该属性是可读/写的，而在追加之后是只读的。

1.2.1.5.19 Type 属性（关键字）(ADOX)

指示关键字的数据类型。

设置和返回值

设置或返回长整型值，该值是如下常量之一：

常量	说明
adKeyPrimary	关键字是主关键字。
AdKeyForeign	关键字是外部关键字。
AdKeyUnique	关键字是唯一的。

说明

默认值是 **adKeyPrimary**。

在已追加到集合的 Key(See 1.2.1.2.5) 对象上，该属性是只读的。

1.2.1.5.20 Type 属性（表）(ADOX)

指示表的类型。

返回值

返回指定表的类型的字符串值；例如，“TABLE”、“SYSTEM TABLE”或“GLOBAL TEMPORARY”值。

说明

该属性是只读的。

1.2.1.5.21 Unique 属性 (ADOX)

指示索引关键字是否必须是唯一的。

设置和返回值

设置并返回布尔值。

说明

默认值是 **False**。

在已追加到集合中的 Index(See 1.2.1.2.4) 对象上，该属性是只读的。

1.2.1.5.22 UpdateRule 属性 (ADOX)

指示主关键字被更新时会执行的操作。

设置和返回值

设置和返回长整型值，该值是如下常量之一：

常量	说明
adRINone	无操作。
AdRICascade	层叠更改。
AdRISetNull	将外部关键字值设置为 null。
adRISetDefault	将外部关键字值设置为默认值。

说明

默认值是 **adRINone**。

在已追加到集合中的 Key(See 1.2.1.2.5) 对象上，该属性是只读的。

1.2.1.6 ADOX 范例

这些主题提供了帮助用户了解如何使用 ADOX 的范例代码。所有代码范例均使用 Visual Basic 编写。范例根据功能区分组并根据复杂性排序如下。

> **注意** 请将整个代码范例（从 Sub 到 End Sub）粘贴到您的代码编辑器中。如果使用部分范例或丢失了段落格式，则范例可能无法正确运行。

数据库连接范例

目录 ActiveConnection 范例(See 1.2.1.6.8)

关闭连接范例(See 1.2.1.6.10)

模式创建范例

创建数据库范例(See 1.2.1.6.2)

创建表范例(See 1.2.1.6.6)

创建索引范例(See 1.2.1.6.3)

创建关键字范例(See 1.2.1.6.4)

指定提供者的属性范例

ParentCatalog 范例(See 1.2.1.6.16)

视图（SELECT 查询）范例

创建视图范例(See 1.2.1.6.7)

删除视图范例(See 1.2.1.6.9)

视图文本范例(See 1.2.1.6.15)

视图字段范例(See 1.2.1.6.14)

存储过程范例

创建过程范例(See 1.2.1.6.5)

删除过程范例(See 1.2.1.6.11)

刷新过程范例(See 1.2.1.6.17)

过程文本范例(See 1.2.1.6.13)

过程参数范例(See 1.2.1.6.12)

用户和组许可权范例

授予许可权范例(See 1.2.1.6.1)

1.2.1.6.1 授予许可权范例 (ADOX)

```
Sub GrantPermissions()

   Dim cnn As New ADODB.Connection
   Dim cat As New ADOX.Catalog

   ' 使用 Jet 4.0 提供者打开与 northwind 数据库的连接
   cnn.Provider = "Microsoft.Jet.OLEDB.4.0"
   cnn.Open "data source=" & _
   "c:\Program Files\Microsoft Office\Office\Samples\Northwind.mdb;" & _
   "jet oledb:system database=mysystem.mdb"

   Set cat.ActiveConnection = cnn

   ' 将对 orders 对象的全部权利授予管理员用户
   cat.Users("Admin").SetPermissions "Orders", adPermObjTable, _
      adAccessSet, adRightAll

End Sub
```

1.2.1.6.2 创建数据库范例 (ADOX)

如下代码显示如何通过 Create(See 1.2.1.4.10) 方法创建新的 Jet 数据库。

```
Sub CreateDatabase()

   Dim cat As New ADOX.Catalog
   cat.Create "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\new.mdb"

End Sub
```

1.2.1.6.3 创建索引范例 (ADOX)

```
Sub CreateIndex()

   Dim tbl As New Table
```

```
   Dim idx As New ADOX.Index
   Dim cat As New ADOX.Catalog

' 打开目录。
   ' 打开目录。
   cat.ActiveConnection = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   ' 定义表并将其追加到目录
   tbl.Name = "MyTable"
   tbl.Columns.Append "Column1", adInteger
   tbl.Columns.Append "Column2", adInteger
   tbl.Columns.Append "Column3", adVarWChar, 50
   cat.Tables.Append tbl

   ' 定义多列索引
   idx.Name = "multicolidx"
   idx.Columns.Append "Column1"
   idx.Columns.Append "Column2"

   ' 将索引追加到表上
   tbl.Indexes.Append idx

End Sub
```

1.2.1.6.4 创建关键字范例 (ADOX)

```
Sub CreateKey()

   Dim kyForeign As New ADOX.Key
   Dim cat As New ADOX.Catalog

   cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   kyForeign.Name = "CustOrder"
   kyForeign.Type = adKeyForeign
   kyForeign.RelatedTable = "Customers"
   kyForeign.Columns.Append "CustomerId"
   kyForeign.Columns("CustomerId").RelatedColumn = "CustomerId"
   kyForeign.UpdateRule = adRICascade
```

```
   cat.Tables("Orders").Keys.Append kyForeign

End Sub
```

1.2.1.6.5 创建过程范例 (ADOX)

```
Sub CreateProcedure()

   Dim cnn As New ADODB.Connection
   Dim cmd As New ADODB.Command
   Dim prm As ADODB.Parameter
   Dim cat As New ADOX.Catalog

   ' 打开连接
   cnn.Open _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   ' 创建参数化命令（Jet 指定）
   Set cmd.ActiveConnection = cnn
   cmd.CommandText = "PARAMETERS [CustId] Text;" & _
      "Select * From Customers Where CustomerId = [CustId]"

   ' 打开目录
   Set cat.ActiveConnection = cnn

   ' 创建新过程
   cat.Procedures.Append "CustomerById", cmd

End Sub
```

1.2.1.6.6 创建表范例 (ADOX)

```
Sub CreateTable()

   Dim tbl As New Table
   Dim cat As New ADOX.Catalog

'打开目录。
   ' 打开目录。
   cat.ActiveConnection = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
```

```
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   tbl.Name = "MyTable"
   tbl.Columns.Append "Column1", adInteger
   tbl.Columns.Append "Column2", adInteger
   tbl.Columns.Append "Column3", adVarWChar, 50
   cat.Tables.Append tbl

End Sub
```

1.2.1.6.7 创建视图范例 (ADOX)

```
Sub CreateView()

   Dim cmd As New ADODB.Command
   Dim cat As New ADOX.Catalog

   ' 打开目录
   cat.ActiveConnection = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   ' 创建代表视图的命令。
      cmd.CommandText = "Select * From Customers"

   ' 创建新视图
   cat.Views.Append "AllCustomers", cmd

End Sub
```

1.2.1.6.8 目录 ActiveConnection 范例 (ADOX)

将 ActiveConnection(See 1.2.1.5.1) 属性设置为有效的、打开的连接即可“打开”目录。通过打开的目录，可以访问包含在该目录中的模式对象。

```
Sub OpenConnection()

   Dim cnn As New ADODB.Connection
   Dim cat As New ADOX.Catalog

      cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source= c:\Program Files\Microsoft Office\" & _
```

```
      "Office\Samples\Northwind.mdb;"
   Set cat.ActiveConnection = cnn
   Debug.Print cat.Tables(0).Type

End Sub
```

将 **ActiveConnection** 属性设置为有效连接字符串也可以“打开”目录。

```
Sub OpenConnectionWithString()

   Dim cat As New ADOX.Catalog

   cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"
   Debug.Print cat.Tables(0).Type

End Sub
```

1.2.1.6.9 删除视图范例 (ADOX)

如下代码说明如何使用 Delete(See 1.2.1.4.11) 方法从目录中删除视图。

```
Sub DeleteView()

   Dim cat As New ADOX.Catalog

   ' 打开目录
   cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   '删除视图
   cat.Views.Delete "AllCustomers"

End Sub
```

1.2.1.6.10 关闭连接范例 (ADOX)

把 ActiveConnection(See 1.2.1.5.1) 属性设置为 **Nothing** 将“关闭”目录。关联的集合将被置空。目录中任何通过模式对象创建的对象都将被孤立。这些已缓存对象的任何属性依然可用，但读取属性时如果该属性需要调用提供者，则此操作将会失败。

```
Sub CloseConnectionByNothing()

   Dim cnn As New Connection
   Dim cat As New Catalog
   Dim tbl As Table
```

```
   cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source= c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"
   Set cat.ActiveConnection = cnn
   Set tbl = cat.Tables(0)
   Debug.Print tbl.Type     ' 缓存 tbl.Type 信息
   Set cat.ActiveConnection = Nothing
   Debug.Print tbl.Type     ' tbl 被孤立
   ' 如果它被缓存，前面的行将成功
   Debug.Print tbl.Columns(0).DefinedSize
   ' 如果该信息未被缓存，前面的行将失败

End Sub
```

关闭用于“打开”目录的 **Connection** 对象，与将 **ActiveConnection** 属性设置为 **Nothing** 效果相同。

```
Sub CloseConnection()

   Dim cnn As New Connection
   Dim cat As New Catalog
   Dim tbl As Table

   cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source= c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"
   Set cat.ActiveConnection = cnn
   Set tbl = cat.Tables(0)
   Debug.Print tbl.Type     ' 缓存 tbl.Type 信息
   cnn.Close
   Debug.Print tbl.Type     ' tbl 被孤立
   ' 如果它被缓存，前面的行将成功
   Debug.Print tbl.Columns(0).DefinedSize
   ' 如果该信息未被缓存，前面的行将失败

End Sub
```

1.2.1.6.11 删除过程范例 (ADOX)

如下代码演示如何使用 Procedures(See 1.2.1.3.5) 集合 Delete(See 1.2.1.4.11) 方法删除过程。

```
Sub DeleteProcedure()

Dim cat As New ADOX.Catalog

   ' 打开目录
   cat.ActiveConnection = _
```

```
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   ' 删除过程。
   cat.Procedures.Delete "CustomerById"

End Sub
```

1.2.1.6.12 过程参数范例 (ADOX)

```
Sub ProcedureParameters()

   Dim cnn As New ADODB.Connection
   Dim cmd As ADODB.Command
   Dim prm As ADODB.Parameter
   Dim cat As New ADOX.Catalog

   ' 打开连接
   cnn.Open _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   ' 打开目录
   Set cat.ActiveConnection = cnn

   ' 获得命令对象
   Set cmd = cat.Procedures("CustomerById").Command

   ' 检索参数信息
   cmd.Parameters.Refresh
   For Each prm In cmd.Parameters
      Debug.Print prm.Name & ":" & prm.Type
   Next

End Sub
```

1.2.1.6.13 过程文本范例 (ADOX)

```
Sub ProcedureText()

   Dim cnn As New ADODB.Connection
```

```
   Dim cat As New ADOX.Catalog
   Dim cmd As New ADODB.Command

   ' 打开连接
   cnn.Open _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   ' 打开目录
   Set cat.ActiveConnection = cnn

   ' 获得 Command
   Set cmd = cat.Procedures("CustomerById").Command

   ' 更新 CommandText
   cmd.CommandText = "Select CustomerId, CompanyName, ContactName " & _
      "From Customers " & _
      "Where CustomerId = [CustId]"

   ' 更新过程
   Set cat.Procedures("CustomerById").Command = cmd

End Sub
```

1.2.1.6.14 视图字段范例 (ADOX)

```
Sub ViewFields()

   Dim cnn As New ADODB.Connection
   Dim rst As New ADODB.Recordset
   Dim fld As ADODB.Field
   Dim cat As New ADOX.Catalog

   ' 打开连接
   cnn.Open _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   ' 打开目录
   Set cat.ActiveConnection = cnn

   ' 设置记录集的源
```

```
   Set rst.Source = cat.Views("AllCustomers").Command

   ' 检索字段信息
   rst.Fields.Refresh
   For Each fld In rst.Fields
      Debug.Print fld.Name & ":" & fld.Type
   Next

End Sub
```

1.2.1.6.15 视图文本范例 (ADOX)

```
Sub ViewText()

   Dim cnn As New ADODB.Connection
   Dim cat As New ADOX.Catalog
   Dim cmd As New ADODB.Command

   ' 打开连接
   cnn.Open _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\Microsoft Office\" & _
      "Office\Samples\Northwind.mdb;"

   ' 打开目录
   Set cat.ActiveConnection = cnn

   ' 获得 Command
   Set cmd = cat.Views("AllCustomers").Command

   ' 更新 Command 的 CommandText
   cmd.CommandText = _
      "Select CustomerId, CompanyName, ContactName From Customers"

   ' 更新视图
   Set cat.Views("AllCustomers").Command. = cmd

End Sub
```

1.2.1.6.16 ParentCatalog 范例 (ADOX)

```
Sub SetAllowZeroLength()
```

```
   Dim cnn As New ADODB.Connection
   Dim cat As New ADOX.Catalog
   Dim tbl As New ADOX.Table
   Dim col As New ADOX.Column

   cnn.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source= c:\Program Files\" & _
      "Microsoft Office\Office\Samples\Northwind.mdb;"
   Set cat.ActiveConnection = cnn
   tbl.Name = "MyTable"
   tbl.Columns.Append "Column1", adInteger
   Set col.ParentCatalog = cat
   col.Name = "Column2"
   col.Type = adVarWChar
   col.Properties("Jet OLEDB:Allow Zero Length") = True
   tbl.Columns.Append col
   cat.Tables.Append tbl

End Sub
```

1.2.1.6.17 过程刷新范例 (ADOX)

如下代码演示如何刷新 Catalog(See 1.2.1.2.1) 的 Procedures(See 1.2.1.3.5) 集合。

```
Sub ProcedureRefresh()

   Dim cat As New ADOX.Catalog

   ' 打开目录
   cat.ActiveConnection = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=c:\Program Files\" & _
      "Microsoft Office\Office\Samples\Northwind.mdb;"

   ' 刷新 Procedures 集合
   cat.Procedures.Refresh

End Sub
```

1.2.1.6.18 AutoIncrement Column 范例 (ADOX)

```
Sub CreateAutoIncrColumn()

     Dim cat     As New ADOX.Catalog
```

```
    Dim tbl     As New ADOX.Table
    Dim col     As New ADOX.Column

  ' 打开目录
  cat.ActiveConnection = "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=C:\Program Files\Microsoft Office\" & _
    "Office\Samples\Northwind.mdb;"

  With tbl
    .Name = "MyContacts"
    Set .ParentCatalog = cat
    ' 创建字段并将它们追加到新的 Table 对象中。
    .Columns.Append "ContactId", adInteger
    ' 产生 ContactId 列和自动递加列
    .Columns("ContactId").Properties("AutoIncrement") = True
    .Columns.Append "CustomerID", adVarWChar
    .Columns.Append "FirstName", adVarWChar
    .Columns.Append "LastName", adVarWChar
    .Columns.Append "Phone", adVarWChar, 20
    .Columns.Append "Notes", adLongVarWChar
  End With

  cat.Tables.Append tbl

  Set cat = Nothing

End Sub
```

1.3 Microsoft ADO MD 程序员参考

Microsoft® ActiveX® Data Objects (Multidimensional) (ADO MD) 用于通过多种语言，如 Microsoft® Visual Basic®、Microsoft® Visual C++® 和 Microsoft® Visual J++™，来实现对多维数据的便捷访问。ADO MD 通过对 Microsoft® ActiveX® Data Objects (ADO) 的扩展，使之包括了特定于多维数据的对象，如 CubeDef(See 1.3.5.2.5) 和 Cellset(See 1.3.5.2.4) 对象。使用 ADO MD，用户能够浏览多维模式、查询立方并检索结果。

与 ADO 一样，ADO MD 使用基本 OLE DB 提供者来实现对数据的访问。要使用 ADO MD，提供者必须是按照 OLE DB for OLAP 规范定义的多维数据提供者 (MDP)。与使用表格式视图来呈现数据的表格式数据提供者 (TDP) 相反，MDP 使用多维视图呈现数据。有关提供者所支持的指定语法和行为的详细信息，请参阅 OLAP OLE DB 提供者的文档。

本文档假设您已掌握 Visual Basic 编程语言的工作知识，并对 ADO 和 OLAP 已有基本了解。详细信息，请参阅 ADO 程序员参考(See 1.)。有关 ADO MD 的详细概述信息，请参阅如下主题：

- 多维模式和数据的概述(See 1.3.1)

- 使用多维数据(See 1.3.2)
- 通过 ADO MD 使用 ADO (See 1.3.3)
- ADO MD 编程(See 1.3.4)
- ADO MD 对象模型(See 1.3.5.1)

1.3.1 多维模式和数据的概述

了解多维模式

在 ADO MD 中，中心元数据对象是“立方”。立方由相关的维、分级结构、级别和成员所构造的集合组成。

“维”是多维数据库中数据的独立目录，由业务实体产生。通常，维包含用作查询标准以度量数据库的项目。

“分级结构”是合计维的路径。维可以有多个间隔级别，级别具有父子关系。分级结构定义这些级别之间的关系。

“级别”是分级结构中进行合计的一个步骤。对具有多层信息的维，每一层就是一个级别。

“成员”是维中的数据项目。通常，使用成员来创建标题或描述数据库的度量。

立方由 ADO MD 中的 CubeDef(See 1.3.5.2.5) 对象表示。维、分级结构、级别和成员也由它们相应的 ADO MD 对象表示：Dimension(See 1.3.5.2.6)、Hierarchy(See 1.3.5.2.7)、Level(See 1.3.5.2.8) 和 Member(See 1.3.5.2.9)。

维

立方的维取决于要在数据库中模型化的业务实体及数据类型。通常，对于选定数据，每个维均是独立的条目点或机制。

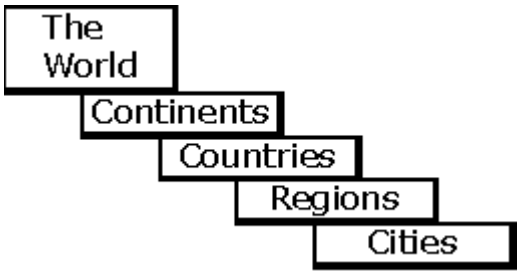
例如，包含销售数据的立方有如下五个维：Salesperson、Geography、Time、Products 和 Measures。Measures 维包含实际的销售数据值，其它维代表对销售数据值进行分类和分组的方法。

Geography 维具有如下成员集：

```
{All, North America, Europe, Canada, USA, UK, Germany, Canada-West,
Canada-East, USA-NW, USA-SW, USA-NE, USA-SE, England, Scotland,
Wales,Ireland, Germany-North, Germany-South, Ottawa, Toronto,
Vancouver, Calgary, Seattle, Boise, Los Angeles, Houston,
Shreveport, Miami, Boston, New York, London, Dover, Glasgow,
Edinburgh, Cardiff, Pembroke, Belfast, Londonderry, Berlin,
Hamburg, Munich, Stuttgart}
```

分级结构

分级结构定义使维的级别能被“卷起”或分组的方法。维可以有多个分级结构。在 Geography 维中，存在天然的分级结构：



级别

在上面的图表所描述的范例 Geography 维中，每个方框均代表分级结构中级别。

每个级别均有成员集合，如下所示：

- The World = {All}

- Continents = {North America, Europe}
- Countries = {Canada, USA, UK, Germany}
- Regions = {Canada-East, Canada-West, USA-NE, USA-NW, USA-SE, USA-SW, England, Ireland, Scotland, Wales, Germany-North, Germany-South}
- Cities = {Ottawa, Toronto, Vancouver, Calgary, Seattle, Boise, Los Angeles, Houston, Shreveport, Miami, Boston, New York, London, Dover, Glasgow, Edinburgh, Cardiff, Pembroke, Belfast, Londonderry, Berlin, Hamburg, Munich, Stuttgart}

成员

位于分级结构叶级别处的成员没有子，位于根级别处的成员则没有父。所有其它成员均至少有一个父，并至少有一个子。例如，对 Geography 维分级结构树进行部分截断产生如下父子关系：

- {All}（是）{Europe, North America}（的父）
- {North America}（是）{Canada, USA}（的父）
- {USA}（是）{USA-NE, USA-NW, USA-SE, USA-SW}（的父）
- {USA-NW}（是）{Boise, Seattle}（的父）

每个维沿一个或多个分级结构即可固定成员。假设有一个 Time 维，有两种方法从 Days 级别卷起到 Year 级别：

Year	Year
Quarter	Week
Month	Day of Week
Day of Month	

该范例也说明了另一个特性：Year-Week 分级结构的 Week 级别上的一些成员没有在 Year-Quarter 分级结构的任何级别中出现。即分级结构不必包括维的所有成员。

1.3.2 使用多维数据

“单元集”是查询多维数据得到的结果。它由“轴”的集合组成，通常不超过四个轴，一般仅二或三个。“轴”是来自一个或多个维的成员的集合，用于在立方中定位或过滤指定的值。

“位置”是轴上的点。对于由单个维组成的轴，这些位置即维成员的子集。如果轴由多个维组成，那么每个位置均是复合实体，该实体有 n 部分，n 是在该轴方向上的维的数量。位置的每个部分均是一个组成维中的成员。

例如，如果在包含销售数据的立方中 Geography 和 Product 维处在单元集的 x 轴方向上，则该轴上的某位置可能包含成员“USA”和“Computers”。在该范例中，在 x 轴上决定某位置需要每个维的成员均朝向该轴。

“单元”是位于轴坐标交叉位置的对象。每个单元均有与它关联的多条信息，包括数据本身、格式化字符串（单元数据的可显示格式）和单元的序号值。（每一个单元均是单元集中唯一的序号值。单元集中的第一个单元的序号值是 0，而在有八列的单元集的第二行中最左边单元的序号值为 8。）

例如，一个立方具有如下六个维（注意，该立方模式与在多维模式和数据(See 1.3.1)中的范例略微不同）：

- Salesperson
- Geography（天然的分级结构）—Continents、Countries、States 等

- Quarters—Quarters、Months、Days
- Years
- Measures—Sales、PercentChange、BudgetedSales
- Products

如下单元集代表所有产品 1991 年的销售情况：

Sales for 1991, All Products

		Valentine				Nash			
		USA			Japan	USA			Japan
		USA_North		USA_South		USA_North		USA_South	
		Seattle	Boston			Seattle	Boston		
Qtr1	Jan	00	10	20	30	40	50	60	70
	Feb	01	11	21	31	41	51	61	71
	Mar	02	12	22	32	42	52	62	72
Qtr2		03	13	23	33	43	53	63	73
Qtr3		04	14	24	34	44	54	64	74
Qtr4	Oct	05	15	25	35	45	55	65	75
	Nov	06	16	26	36	46	56	66	76
	Dec	07	17	27	37	47	57	67	77

注意 该范例中的单元值可以被看作是一对排序后的轴位置序数，其中，第一个数字代表 x 轴的位置，而第二个数字则代表 y 轴的位置。

该单元集的特性如下：

- 轴维：Quarters、Salesperson、Geography
- 过滤维：Measures、Years、Products
- 两个轴：COLUMN（x 或 轴 0）和 ROW（y 或轴 1）
- x 轴：两个嵌套维，Salesperson 和 Geography
- y 轴：Quarters 维

x 轴有两个嵌套维：Salesperson 和 Geography。从 Geography 中，选择了四个成员：Seattle、Boston、USA-South 和 Japan。从 Salesperson 中选择的两个成员是：Valentine 和 Nash。因此，在该轴上总共产生八个位置 (8 = 4*2)。

使用两个成员将每个坐标表示为位置：一个来自于 Salesperson 维，另一个来自于 Geography 维：

```
(Valentine, Seattle), (Valentine, Boston), (Valentine, USA_North),
(Valentine, Japan), (Nash, Seattle), (Nash, Boston), (Nash, USA_North),
(Nash, Japan)
```

y 轴仅有一个维，包含如下八个位置：

```
Jan, Feb, Mar, Qtr2, Qtr3, Oct, Nov, Dec
```

在 ADO MD 中，单元集、单元、轴和位置均由相应的对象表示：Cellset(See 1.3.5.2.4)、Cell(See 1.3.5.2.3)、Axis(See 1.3.5.2.1) 和

Position(See 1.3.5.2.10)。

1.3.3 通过 ADO MD 使用 ADO

ADO 和 ADO MD 是相关但又独立的对象模型。ADO 提供用于连接数据源、执行命令、按表格式检索表格式数据和模式元数据、和查看提供者错误信息的对象。ADO MD 提供用于检索多维数据和查看多维模式元数据的对象。

当使用 MDP 工作时，您可能会在应用程序中选择使用 ADO、ADO MD 或同时选择二者。通过在项目中引用两个库，将能够实现对 MDP 所提供功能的完全访问。

对于用户来说，获得一个多维数据库的平铺、表格式的视图，通常是有用的。使用 ADO Recordset(See 1.1.4.2.10) 对象即可实现该操作。这时，请将 Cellset(See 1.3.5.2.4) 的源指定为 **Recordset** Open(See 1.1.4.4.33) 方法的 **Source** 参数，而不是 ADO MD **Cellset** 的源。

将模式元数据作为用表格式视图、而不是对象的分级结构来查看，也是有用的。ADO Connection(See 1.1.4.2.2) 对象的 OpenSchema(See 1.1.4.4.34) 方法允许用户打开包含模式信息的 **Recordset**。**OpenSchema** 方法的 **QueryType** 参数有几个与 MDP 关系特殊的值。这些值是：

- **adSchemaCubes**
- **adSchemaDimensions**
- **adSchemaHierarchies**
- **adSchemaLevels**
- **adSchemaMeasures**
- **adSchemaMembers**

要通过 ADO MD 属性或方法使用 ADO enum 值，您的项目必须同时引用 ADO 和 ADO MD 两个库。例如，通过 ADO MD State(See 1.3.5.5.19) 属性可以使用 ADO **adState** enum 值。有关建立库引用的详细信息，请参阅开发工具的相关文档。

有关 ADO 对象和方法的详细信息，请参阅 ADO 程序员参考(See 1.)。

1.3.4 ADO MD 编程

要通过开发工具使用 ADO MD，应当建立对 ADO MD 类型库的引用。ADO MD 库的说明是 Microsoft ActiveX Data Objects (Multi-dimensional) Library。ADO MD 库的文件名是 msadomd.dll，程序 ID (ProgID) 是“ADOMD”。有关建立库引用的详细信息，请参阅开发工具的相关文档。

1.3.5 ADO MD API 参考

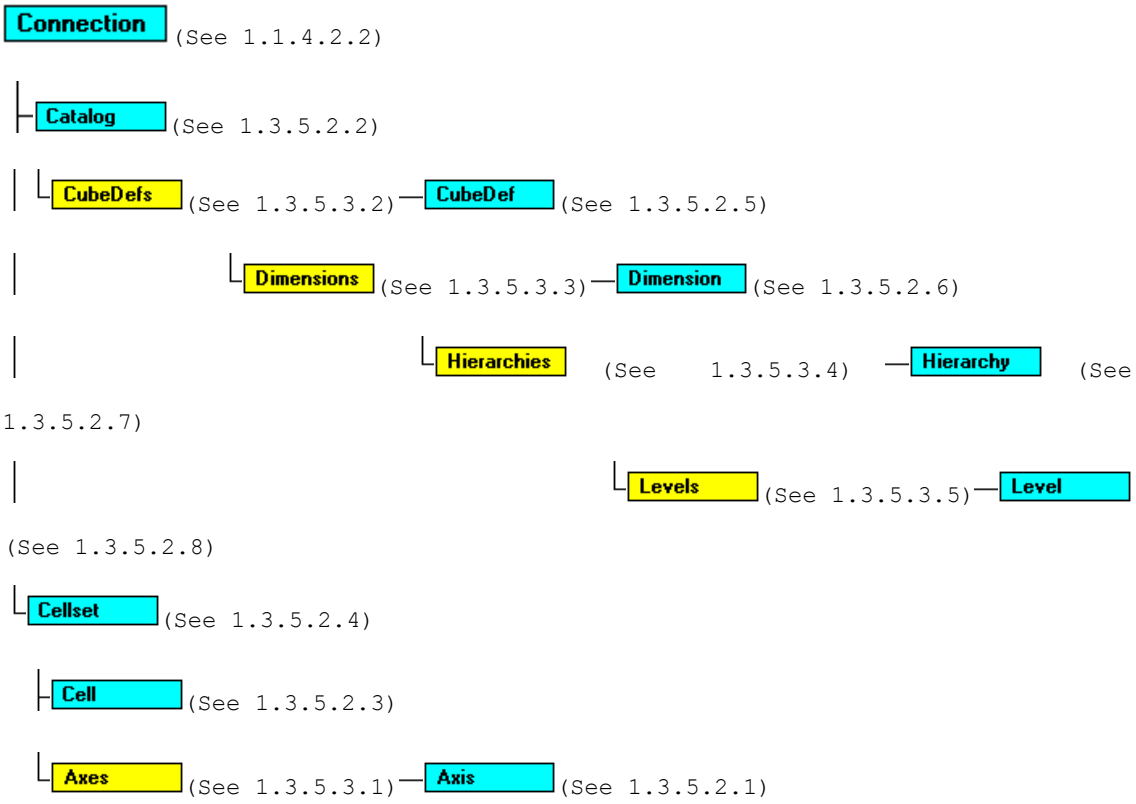
本节 ADO MD 文档的内容包含每个 ADO MD 对象、集合、方法和属性的主题，同时还适时提供范例代码。详细信息，请在索引中搜索特定主题或参考如下主题：

- ADO MD 对象(See 1.3.5.2)

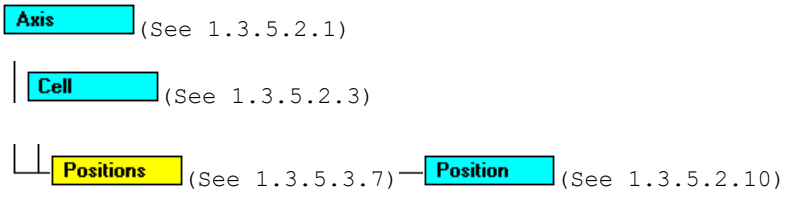
- ADO MD 集合(See 1.3.5.3)
- ADO MD 方法(See 1.3.5.4)
- ADO MD 属性(See 1.3.5.5)
- ADO MD 范例(See 1.3.5.6)

1.3.5.1 ADO MD 对象模型

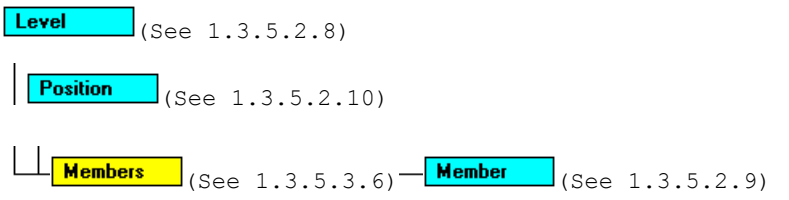
如下图表说明如何表示这些对象以及它们在 ADO MD 中的关系。



Axis(See 1.3.5.2.1) 和 Cell(See 1.3.5.2.3) 对象各自拥有 Positions(See 1.3.5.3.7) 集合。

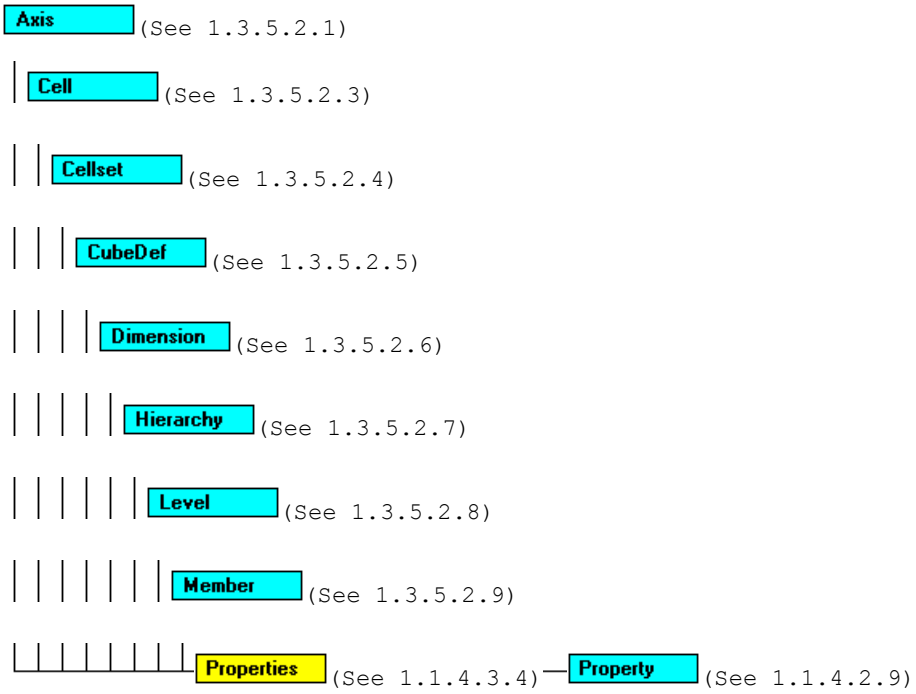


Level(See 1.3.5.2.8) 和 Position(See 1.3.5.2.10) 对象各自拥有 Members(See 1.3.5.3.6) 集合。



Axis(See 1.3.5.2.1)、Cell(See 1.3.5.2.3)、Cellset(See 1.3.5.2.4)、CubeDef(See 1.3.5.2.5)、Dimension(See 1.3.5.2.6)、Hierarchy(See 1.3.5.2.7)、

Level(See 1.3.5.2.8) 和 Member(See 1.3.5.2.9) 对象各自拥有标准的 ADO Properties(See 1.1.4.3.4) 集合。



1.3.5.2 ADO MD 对象

ADO MD 对象总结

对象	说明
Axis(See 1.3.5.2.1)	表示单元集的位置轴或过滤轴，包含所选的一维或多维成员。
Catalog(See 1.3.5.2.2)	包含特定多维数据提供者 (MDP) 的多维模式信息（即立方和基本维、分级结构、级别和成员）。
Cell(See 1.3.5.2.3)	代表位于轴坐标交叉点上的数据，包含在单元集中。
Cellset(See 1.3.5.2.4)	表示多维查询的结果。它是从立方或其它单元集中选出的单元的集合。
CubeDef(See 1.3.5.2.5)	表示多维模式中的一个立方，包含有关维的集合。
Dimension(See 1.3.5.2.6)	表示多维立方的一个维，包含成员的一个或多个分级结构。
Hierarchy(See 1.3.5.2.7)	表示一种方式，按此方式维的成员可以被合计或“卷起”。维可以按照一个或多个分级结构进行合计。
Level(See 1.3.5.2.8)	包含一个成员集，其中每个成员在分级结构内均有相同的等级。
Member(See 1.3.5.2.9)	代表立方中级别的成员、级别成员的子或位于单元集的轴上某位置的成员。

Position(See 1.3.5.2.10)	代表定义轴上点的不同维的一个或多个成员的集合。

同时，**Catalog** 对象与 ADO **Connection** 对象相连接，**Connection** 对象包括在标准 ADO 库中：

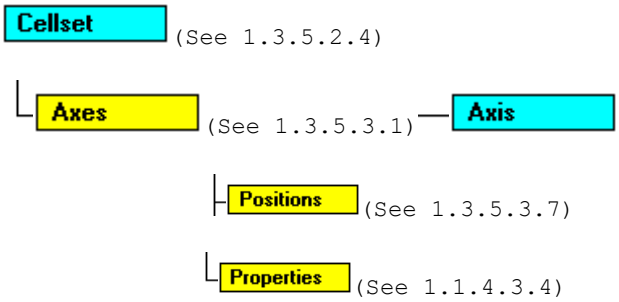
对象	说明
Connection(See 1.1.4.2.2)	**Connection** 对象，代表打开的、与数据源的连接。

这些对象间的关系以图形方式表示在 ADO MD 对象模型(See 1.3.5.1) 中。

很多 ADO MD 对象可以包含在相应的集合中。例如，**CubeDef** 对象可以包含在 **Catalog** 的 CubeDefs(See 1.3.5.3.2) 集合中。详细信息，请参阅 ADO MD 集合(See 1.3.5.3)。

1.3.5.2.1 Axis 对象 (ADO MD)

表示单元集的位置轴或过滤轴，包含所选的一维或多维成员。



说明

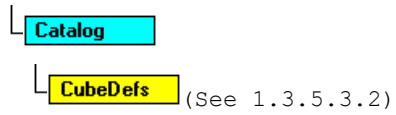
Axis 对象可以被 Axes(See 1.3.5.3.1) 集合包含，或由 Cellset(See 1.3.5.2.4) 的 FilterAxis(See 1.3.5.5.9) 属性返回。

通过 **Axis** 对象的集合和属性，可以：

- 使用 Name(See 1.3.5.5.13) 属性标识 **Axis**。
- 使用 Positions(See 1.3.5.3.7) 集合对在 **Axis** 上的每个位置进行迭代。
- 使用 DimensionCount(See 1.3.5.5.7) 属性获得 **Axis** 的维数。
- 使用标准 ADO Properties(See 1.1.4.3.4) 集合获得特定提供者 **Axis** 的属性。

1.3.5.2.2 Catalog 对象 (ADO MD)

包含特定多维数据提供者 (MDP) 的多维模式信息（即立方和基本维、分级结构、级别和成员）。



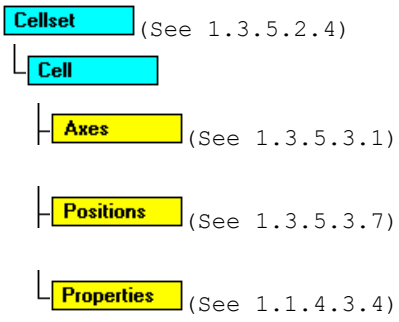
说明

使用 **Catalog** 对象的集合和属性，可以：

- 通过将 ActiveConnection(See 1.3.5.5.1) 属性设置为标准 ADO Connection(See 1.1.4.2.2) 对象或有效连接字符串，来打开目录。
- 使用 Name(See 1.3.5.5.13) 属性标识 **Catalog**。
- 使用 CubeDefs(See 1.3.5.3.2) 集合对目录中的立方进行迭代。

1.3.5.2.3 Cell 对象 (ADO MD)

代表位于轴坐标交叉点上的、包含在单元集中的数据。



说明

通过 Cellset(See 1.3.5.2.4) 对象的 Item(See 1.3.5.4.2) 方法返回 **Cell** 对象。

使用 **Cell** 对象的集合和属性，可以：

- 使用 Value(See 1.3.5.5.22) 属性返回 **Cell** 中的数据。
- 使用 FormattedValue(See 1.3.5.5.10) 属性返回表示 **Value** 属性格式化显示的字符串。
- 使用 Ordinal(See 1.3.5.5.14) 属性返回 **Cellset** 中 **Cell** 的序号值。
- 使用 Positions(See 1.3.5.3.7) 集合确定 CubeDef(See 1.3.5.2.5) 中 **Cell** 的位置。
- 使用标准 ADO Properties(See 1.1.4.3.4) 集合检索有关 **Cell** 的其它信息。

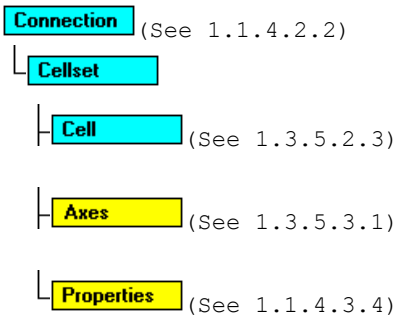
Properties 集合包含由提供者提供的属性。下表列出了可能可用的属性。实际属性的列表可能会由于提供者的具体情况而不同。有关可用属性更为完整的列表，请参阅提供者文档。

名称	说明

BackColor	显示单元时使用的背景色。
FontFlags	字体的位掩码详细效果。
FontName	用于显示单元值的字体。
FontSize	用于显示单元值的字体大小。
ForeColor	显示单元时使用的前景色。
FormatString	格式化字符串的值。

1.3.5.2.4 Cellset 对象 (ADO MD)

表示多维查询的结果。它是从立方或其它单元集中选出的单元的集合。



说明

使用直接、类似数组的访问来实现对 **Cellset** 中数据的检索。可以“深入”到特定的成员以获得该成员的数据。例如，下列代码将返回名为 cst 的单元集第一坐标轴第一位置中第一个成员的标题：

cst.Axes(0).Positions(0).Members(0).Caption

在单元集中没有当前单元的概念，而是使用 Item(See 1.3.5.4.2) 方法从单元集中检索指定的 Cell(See 1.3.5.2.3) 对象。**Item** 方法的参数确定检索哪个单元。可以指定单元的唯一序号值。也可通过使用单元集每个轴上的位置号来检索单元。有关检索单元的详细信息，请参阅 Item(See 1.3.5.4.2) 方法。

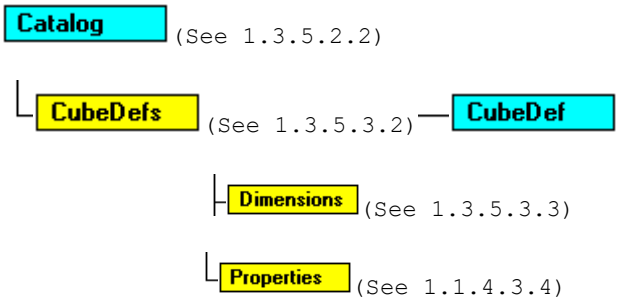
使用 **Cellset** 对象的集合、方法和属性，可以：

- 通过设置它的 ActiveConnection(See 1.3.5.5.1) 属性，使打开的连接与 **Cellset** 对象相关联。
- 使用 Open(See 1.3.5.4.3) 方法执行并检索多维查询的结果。
- 使用 Item(See 1.3.5.4.2) 方法从 **Cellset** 中检索 **Cell**。
- 使用 Axes(See 1.3.5.3.1) 集合返回定义 **Cellset** 的 Axis(See 1.3.5.2.1) 对象。
- 使用 FilterAxis(See 1.3.5.5.9) 属性检索有关用于过滤 **Cellset** 中数据的维的信息。
- 使用 Source(See 1.3.5.5.18) 属性返回或指定用于定义 **Cellset** 的查询。

- 使用 State(See 1.3.5.5.19) 属性返回 **Cellset** 的当前状态（打开、关闭、正在执行或正在连接）。
- 使用 Close(See 1.3.5.4.1) 方法关闭打开的 **Cellset**。
- 使用标准的 ADO Properties(See 1.1.4.3.4) 集合检索特定提供者的 **Cellset** 信息。

1.3.5.2.5 CubeDef 对象 (ADO MD)

代表多维模式中的立方，包含相关维的集合。



说明

使用 **CubeDef** 对象的集合和属性，可以：

- 使用 Name(See 1.3.5.5.13) 属性标识 **CubeDef**。
- 使用 Description(See 1.3.5.5.6) 属性返回描述立方的字符串。
- 使用 Dimensions(See 1.3.5.3.3) 集合返回构成立方的维。
- 使用标准 ADO Properties(See 1.1.4.3.4) 集合获得 **CubeDef** 的其他信息。

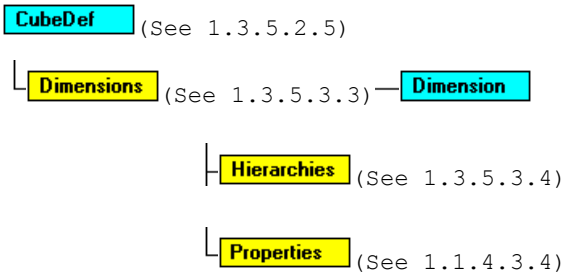
Properties 集合包含由提供者提供的属性。下表列出了可能可用的属性。实际属性列表可能会由于提供者的具体情况而有所不同。有关可用属性完整的列表，请参阅提供者的文档。

名称	说明
CatalogName	该立方所属的目录的名称。
CreatedOn	创建立方的日期和时间。
CubeGUID	立方 GUID。
CubeName	立方的名称。
CubeType	立方的类型。
DataUpdatedBy	最后进行数据更新的人的用户 ID。
Description	有意义的立方说明。

LastSchemaUpdate	最后模式更新的日期和时间。
SchemaName	该立方所属的模式名称。
SchemaUpdatedBy	最后进行模式更新的人的用户 ID。

1.3.5.2.6 Dimension 对象 (ADO MD)

代表多维立方中的一个维，包含一个或多个成员的分级结构。



说明

使用 **Dimension** 对象的集合和属性，可以：

- 使用 Name(See 1.3.5.5.13) 和 UniqueName(See 1.3.5.5.21) 属性标识 **Dimension**。
- 使用 Description(See 1.3.5.5.6) 属性返回描述 **Dimension** 的有意义的字符串。
- 使用 Hierarchies(See 1.3.5.3.4) 集合返回构成 **Dimension** 的 Hierarchy(See 1.3.5.2.7) 对象。
- 使用标准的 ADO Properties(See 1.1.4.3.4) 集合获得 **Dimension** 对象的其他信息。

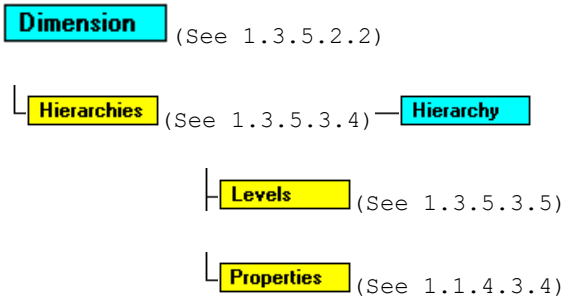
Properties 集合包含由提供者提供的属性。下表列出了可能可用的属性。实际属性列表可能会由于提供者的具体情况而有所不同。有关可用属性的完整列表，请参阅提供者的文档。

名称	说明
CatalogName	该立方所属的目录的名称。
CubeName	立方的名称。
DefaultHierarchy	默认分级结构的唯一名称。
Description	有意义的立方说明。
DimensionCaption	与维关联的标签或标题。
DimensionCardinality	维中的成员数。
DimensionGUID	维的 GUID。

DimensionName	维的名称。
DimensionOrdinal	在形成立方的维的组中维的序号。
DimensionType	维类型。
DimensionUniqueName	维的明确名称。
SchemaName	该立方所属的模式的名称。

1.3.5.2.7 Hierarchy 对象 (ADO MD)

表示一种方式，按此方式维的成员可以被合计或“卷起”。维可以按照一个或多个分级结构进行合计。



说明

使用 **Hierarchy** 对象的集合和属性，可以：

- 使用 Name(See 1.3.5.5.13) 和 UniqueName(See 1.3.5.5.21) 属性标识 **Hierarchy**。
- 使用 Description(See 1.3.5.5.6) 属性返回描述 **Hierarchy** 的有意义的字符串。
- 使用 Levels(See 1.3.5.3.5) 集合返回构成 **Hierarchy** 的 Level(See 1.3.5.2.8) 对象。
- 使用标准的 ADO Properties(See 1.1.4.3.4) 集合获得有关 **Hierarchy** 对象的其他信息。

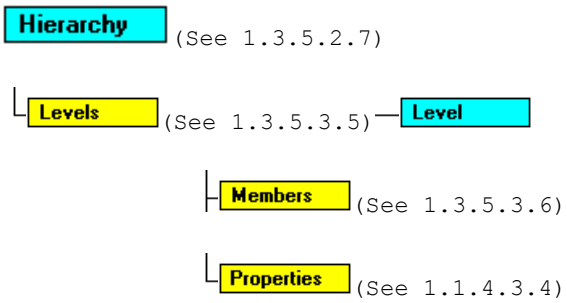
Properties 集合包含由提供者提供的属性。下表列出了可能可用的属性。实际属性列表可能会由于提供者的具体情况而有所不同。有关可用属性的完整列表，请参阅提供者的文档。

名称	说明
AllMember	在分级机构中位于最高卷起级别的成员。
CatalogName	该立方所属的目录的名称。
CubeName	立方的名称。
DefaultMember	该分级结构的默认成员的唯一名称。

Description	有意义的分级结构说明。
DimensionType	该分级结构所属的维的类型。
DimensionUniqueName	维的明确名称。
HierarchyCaption	与分级结构关联的标签或标题。
HierarchyCardinality	分级结构中的成员数。
HierarchyGUID	分级结构的 GUID。
HierarchyName	分级结构的名称。
HierarchyUniqueName	分级结构的明确名称。
SchemaName	该立方所属的模式的名称。

1.3.5.2.8 Level 对象 (ADO MD)

包含一个成员集，其中每个成员在分级结构内均有相同的等级。



说明

使用 **Level** 对象的集合和属性，可以：

- 使用 Name(See 1.3.5.5.13) 和 UniqueName(See 1.3.5.5.21) 属性标识 **Level**。
- 使用 Caption(See 1.3.5.5.2) 属性返回显示 **Level** 时所使用的字符串。
- 使用 Description(See 1.3.5.5.6) 属性返回描述 **Level** 的有意义的字符串。
- 使用 Members(See 1.3.5.3.6) 集合返回构成 **Level** 的 Member(See 1.3.5.2.9) 对象。
- 使用 Depth(See 1.3.5.5.5) 属性返回从 **Level** 的根起算的级别号。
- 使用标准的 ADO Properties(See 1.1.4.3.4) 集合获得有关 **Level** 对象的其他信息。

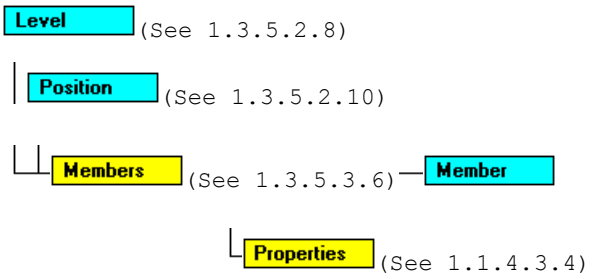
Properties 集合包含由提供者提供的属性。下表列出了可能可用的属性。实际属性列表可能会由于提供者的具体情况而有所不同。

有关可用属性的完整列表，请参阅提供者的文档。

名称	说明
CatalogName	该立方所属的目录的名称。
CubeName	立方的名称。
Description	有意义的级别说明。
DimensionUniqueName	维的明确名称。
HierarchyUniqueName	分级结构的明确名称。
LevelCaption	与级别关联的标签或标题。
LevelCardinality	级别中的成员数。
LevelGUID	级别的 GUID。
LevelName	级别的名称。
LevelNumber	分级结构的根和级别间的距离。
LevelType	级别的类型。
LevelUniqueName	级别的明确名称。
SchemaName	该立方所属的模式的名称。

1.3.5.2.9 Member 对象 (ADO MD)

代表立方中级别的成员、级别成员的子或位于单元集的轴上某位置的成员。



说明

根据使用环境，**Member** 的属性各不相同。CubeDef(See 1.3.5.2.5) 中 Level(See 1.3.5.2.8) 的 **Member** 有一个 Children(See 1.3.5.5.4) 属性，该属性可以由当前 **Member** 返回分级结构中位于下一个较低级别的 **Members**。对于 Position(See 1.3.5.2.10) 的 **Member**，**Children** 集合始终为空。同样，Type(See 1.3.5.5.20) 属性仅适用于 **Level** 的 **Members**。

Position 的 **Member** 具有两个属性：DrilledDown(See 1.3.5.5.8) 和 ParentSameAsPrev(See 1.3.5.5.17)，在显示 Cellset(See 1.3.5.2.4) 时使用。如果对 **Level** 的 **Members** 访问这些属性，将出现错误。

使用 **Level** 的 **Member** 对象的集合和属性，可以：

- 使用 Name(See 1.3.5.5.13) 和 UniqueName(See 1.3.5.5.21) 属性标识 **Member**。
- 使用 Caption(See 1.3.5.5.2) 属性返回显示 **Member** 时所使用的字符串。
- 使用 Description(See 1.3.5.5.6) 属性返回描述措施或公式 **Member** 的有意义的字符串。
- 使用 Description(See 1.3.5.5.6) 属性返回描述措施或公式 **Member** 的有意义的字符串。
- 使用 Type(See 1.3.5.5.20) 属性确定 **Member** 的属性。
- 使用 LevelDepth(See 1.3.5.5.11) 和 LevelName(See 1.3.5.5.12) 属性获得 **Member** 的 **Level** 信息。
- 使用 Parent(See 1.3.5.5.16) 和 Children(See 1.3.5.5.4) 属性获得 Hierarchy(See 1.3.5.2.7) 中的相关 **Members**。
- 使用 ChildCount(See 1.3.5.5.3) 属性对 **Member** 的子进行计数。
- 使用标准的 ADO Properties(See 1.1.4.3.4) 集合获得 **Level** 对象的其他信息。

使用 Axis(See 1.3.5.2.1) 上 **Position** 的 **Member** 的集合和属性，可以：

- 使用 Name(See 1.3.5.5.13) 和 UniqueName(See 1.3.5.5.21) 属性标识 **Member**。
- 使用 Caption(See 1.3.5.5.2) 属性返回显示 **Member** 时所使用的字符串。
- 使用 Description(See 1.3.5.5.6) 属性返回描述措施或公式 **Member** 的有意义的字符串。
- 使用 LevelDepth(See 1.3.5.5.11) 和 LevelName(See 1.3.5.5.12) 属性获得 **Member** 的 **Level** 信息。
- 使用 ChildCount(See 1.3.5.5.3) 属性对 **Member** 的子进行计数。
- 使用 DrilledDown(See 1.3.5.5.8) 属性确定是否在 **Axis** 上该 **Member** 之后至少有一个直接的子。
- 使用 ParentSameAsPrev(See 1.3.5.5.17) 属性确定是否该 **Member** 的父与直接在前的 **Member** 的父相同。
- 使用标准的 ADO Properties(See 1.1.4.3.4) 集合可获得 **Level** 对象的其他信息。

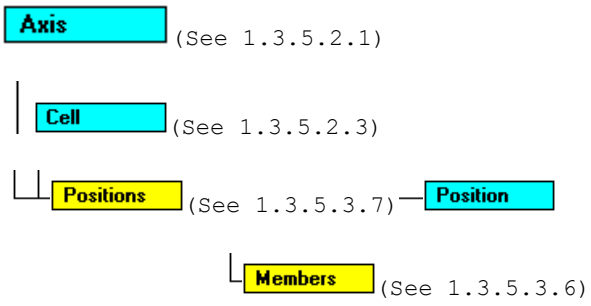
Properties 集合包含由提供者提供的属性。下表列出了可能可用的属性。实际属性列表可能会根据提供者的具体情况而有所不同。有关可用属性的完整列表，请参阅提供者的文档。

名称	说明
CatalogName	该立方所属的目录的名称。
ChildrenCardinality	成员拥有的子的数目。
CubeName	立方的名称。
Description	有意义的成员说明。

DimensionUniqueName	维的明确名称。
HierarchyUniqueName	分级结构的明确名称。
LevelNumber	分级结构的级别和根之间的距离。
LevelUniqueName	级别的明确名称。
MemberCaption	与成员关联的标签或标题。
MemberGUID	成员的 GUID。
MemberName	成员的名称。
MemberOrdinal	成员的序号。
MemberType	成员的类型。
MemberUniqueName	成员的明确名称。
ParentCount	对该成员所具有的父的数目的计数。
ParentLevel	成员的父的级别号。
ParentUniqueName	成员的父的明确名称。
SchemaName	该立方所属的模式的名称。

1.3.5.2.10 Position 对象 (ADO MD)

代表定义轴上点的不同维的一个或多个成员的集合。



说明

使用 **Position** 对象的集合和属性，可以：

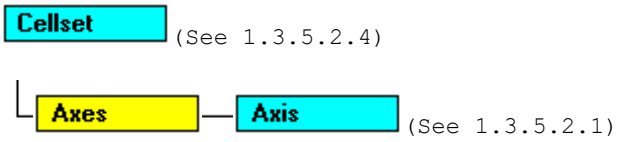
- 使用 **Ordinal** 属性返回 Axis(See 1.3.5.2.1) 上 **Position** 的序号位置。
- 使用 Members(See 1.3.5.3.6) 集合返回构成 **Axis** 上位置的成员。

1.3.5.3 ADO MD 集合

集合	说明
Axes(See 1.3.5.3.1)	包含定义单元集的 Axis 对象。
CubeDefs(See 1.3.5.2.5)	包含代表多维目录中的立方的 CubeDef 对象。
Dimensions(See 1.3.5.2.6)	包含构成立方的 Dimension 对象。
Hierarchies(See 1.3.5.2.7)	包含对应于维的 Hierarchy 对象集。
Levels(See 1.3.5.2.8)	包含构成分级结构的 Level 对象。
Members(See 1.3.5.3.6)	包含对应于轴上的级别或位置的 Member 对象。
Positions(See 1.3.5.3.7)	包含定义轴上的点的 Position 对象。

1.3.5.3.1 Axes 集合 (ADO MD)

包含定义单元集的 Axis(See 1.3.5.2.1) 对象。



说明

Cellset(See 1.3.5.2.4) 对象包含 **Axes** 集合。一旦打开 **Cellset**，该集合将至少包含一个 **Axis**。有关如何使用 **Axis** 对象的详细说明，请参阅 Axis(See 1.3.5.2.1) 对象。

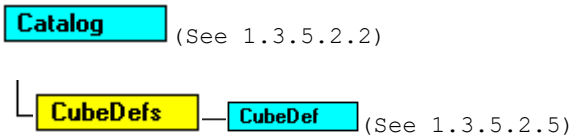
注意 **Axes** 集合中不包含 **Cellset** 的过滤轴。详细信息，请参阅 FilterAxis(See 1.3.5.5.9) 属性。

Axes 是标准的 ADO 集合。使用集合的属性和方法，可以：

- 使用 Count(See 1.1.4.6.16) 属性获得集合中对象的数量。
- 使用默认的 Item(See 1.1.4.4.27) 方法返回集合中的对象。
- 使用 Refresh(See 1.1.4.4.36) 方法更新来自提供者的集合中的对象。

1.3.5.3.2 CubeDefs 集合 (ADO MD)

包含代表多维目录中的立方的 CubeDef(See 1.3.5.2.5) 对象。



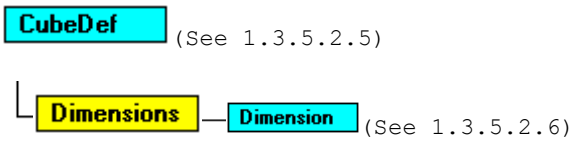
说明

CubeDefs 是标准的 ADO 集合。使用集合的属性和方法，可以：

- 使用 Count(See 1.1.4.6.16) 属性获得集合中对象的数量。
- 使用默认的 Item(See 1.1.4.4.27) 方法返回集合中的对象。
- 使用 Refresh(See 1.1.4.4.36) 方法更新来自提供者的集合中的对象。

1.3.5.3.3 Dimensions 集合 (ADO MD)

包含构成立方的 Dimension(See 1.3.5.2.6) 对象。



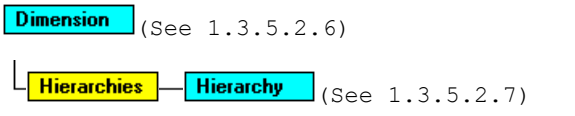
说明

Dimensions 是标准的 ADO 集合。使用集合的属性和方法，可以：

- 使用 Count(See 1.1.4.6.16) 属性获得集合中对象的数量。
- 使用默认的 Item(See 1.1.4.4.27) 方法返回集合中的对象。
- 使用 Refresh(See 1.1.4.4.36) 方法更新来自提供者的集合中的对象。

1.3.5.3.4 Hierarchies 集合 (ADO MD)

包含对应于维的 Hierarchy(See 1.3.5.2.7) 对象集。



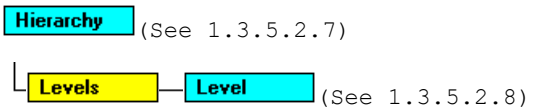
说明

Hierarchies 是标准的 ADO 集合。使用集合的属性和方法，可以：

- 使用 Count(See 1.1.4.6.16) 属性获得集合中对象的数量。
- 使用默认的 Item(See 1.1.4.4.27) 方法返回集合中的对象。
- 使用 Refresh(See 1.1.4.4.36) 方法更新来自提供者的集合中的对象。

1.3.5.3.5 Levels 集合 (ADO MD)

包含构成分级结构的 Level(See 1.3.5.2.8) 对象。



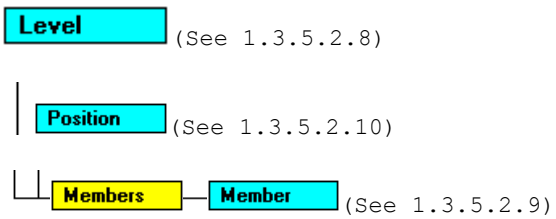
说明

Levels 是标准的 ADO 集合。使用集合的属性和方法，可以：

- 使用 Count(See 1.1.4.6.16) 属性获得集合中对象的数量。
- 使用默认的 Item(See 1.1.4.4.27) 方法返回集合中的对象。
- 使用 Refresh(See 1.1.4.4.36) 方法更新来自提供者的集合中的对象。

1.3.5.3.6 Members 集合 (ADO MD)

包含对应于轴上的级别或位置的 Member(See 1.3.5.2.9) 对象。



说明

Members 集合用于包含如下成员类型：

- 立方中组成级别的成员。这些成员包含在 Level(See 1.3.5.2.8) 对象的 **Members** 集合中。例如，使用多维模式和数据概述(See 1.3.1)中的范例，Countries 级别的四个成员是 Canada、USA、UK 和 Germany。
- 是分级结构中指定成员的子的成员。这些成员由父 **Member** 对象的 Children(See 1.3.5.5.4) 属性返回。例如，再次使用相同的范例，Canada 成员的两个子成员为 Canada-East 和 Canada-West。

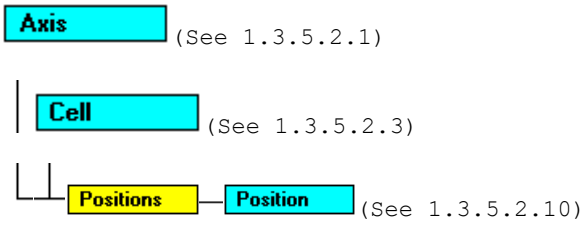
- 定义单元集的轴上指定位置的成员。将使用多维数据(See 1.3.2)中的单元集作为范例，位于 x 轴上的第一位置处的两个成员是 Valentine 和 Seattle。这些成员包含在 Position(See 1.3.5.2.10) 对象的 **Members** 集合中。

Members 是标准的 ADO 集合。使用集合的属性和方法，可以：

- 使用 Count(See 1.1.4.6.16) 属性获得集合中对象的数量。
- 使用默认的 Item(See 1.1.4.4.27) 方法返回集合中的对象。
- 使用 Refresh(See 1.1.4.4.36) 方法更新来自提供者的集合中的对象。

1.3.5.3.7 Positions 集合 (ADO MD)

包含定义轴上的点的 Position(See 1.3.5.2.10) 对象。



说明

Positions 是标准的 ADO 集合。使用集合的属性和方法，可以：

- 使用 Count(See 1.1.4.6.16) 属性获得集合中对象的数量。
- 使用默认的 Item(See 1.1.4.4.27) 方法返回集合中对象。
- 使用 Refresh(See 1.1.4.4.36) 方法更新来自提供者的集合中的对象。

1.3.5.4 ADO MD 方法

方法	说明
Close(See 1.3.5.4.1)	关闭打开的单元集。
Item(See 1.3.5.4.2)	使用其坐标检索单元集中的单元。
Item(See 1.1.4.4.27)	按名称或序号返回集合的指定成员。
Open(See 1.3.5.4.3)	检索多维查询的结果并将其返回到单元集。
Refresh(See	更新集合中的对象，以反映可从提供者处获得的对象和特定于提供者的对象。

1.1.4.4.36)	

1.3.5.4.1 Close 方法 (ADO MD)

关闭打开的单元集。

语法

Cellset.**Close**

说明

使用 **Close** 方法关闭 Cellset(See 1.3.5.2.4) 对象，以释放关联的数据，包括在任何 Cell(See 1.3.5.2.3)、Axis(See 1.3.5.2.1)、Position(See 1.3.5.2.10) 或 Member(See 1.3.5.2.9) 对象中的数据。关闭 **Cellset** 并不将其从内存中删除，随后可以更改其属性设置并再次将其打开。要将对象从内存中彻底清除，请将对象变量设置为 **Nothing**。

可以稍后调用 Open(See 1.3.5.4.3) 方法、使用相同或其他源字符串重新打开 **Cellset**。在 **Cellset** 对象关闭时，进行引用基本数据或图元数据的任何操作，如检索属性或调用方法，都将产生错误。

1.3.5.4.2 Item 方法（ADO MD 单元集）

使用其坐标检索单元集中的单元。

语法

Set *Cell* = *Cellset*.**Item (** *Positions* **)**

参数

Positions 变体型数组。唯一指定单元的值。*Positions* 可以是：

- 位置号数组
- 成员名数组
- 序号位置

说明

使用 **Item** 方法返回 Cellset(See 1.3.5.2.4) 对象中的 Cell(See 1.3.5.2.3) 对象。如果 **Item** 方法找不到与 *Positions* 参数对应的单元，将出现错误。

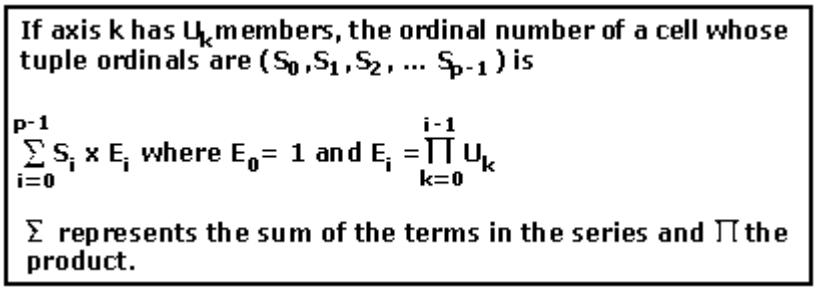
Item 方法是 **Cellset** 对象的默认方法。如下语法格式等价：

Cellset.Item (*Positions*)

Cellset (*Positions*)

Positions 参数指定返回哪个单元。可以按序号位置或按在每个轴上的位置来指定单元。当按在每个轴上的位置指定单元时，可以指定位置的数字值或每个位置的成员名。

序号位置是唯一标识 **Cellset** 中的单元的数字。从概念上讲，**Cellset** 中单元的编号如同是将 **Cellset** 看作一个 *p* 维数组，此处 *p* 是轴号。单元按以行为主的顺序定址。下面是计算单元序号的公式：



如果成员名按字符串传给 **Item**，则成员必须按数字轴标识符的升序排列。在轴中，成员必须按维嵌套的升序排列，即首先是最外部维的成员，后面的是内部维的成员。每个维应由单独的字符串表示，成员字符串的列表应以逗号分隔。

注意 数据提供者可能不支持按成员名检索单元。详细信息，请参阅提供者的文档。

1.3.5.4.3 Open 方法 (ADO MD)

检索多维查询的结果并将其返回到单元集。

语法

Cellset.**Open** *Source*, *ActiveConnection*

参数

Source 可选，变体型。计算出有效的多维查询，如多维表达式 (MDX) 查询。*Source* 参数与 Source(See 1.3.5.5.18) 属性相对应。有关 MDX 的详细信息，请参阅 Microsoft Data Access SDK 中的相关文档。

ActiveConnection 可选，变体型。计算出一个字符串，用于指定有效的 ADO Connection(See 1.1.4.2.2) 对象变量名或连接的定义。*ActiveConnection* 参数指定在其中打开 Cellset(See 1.3.5.2.4) 对象的连接。如果传递该参数的连接定义，ADO 将使用指定参数打开新连接。*ActiveConnection* 参数与 ActiveConnection(See 1.3.5.5.1) 属性相对应。

说明

如果其某个参数被省略，并且在打开 **Cellset** 之前尚未设置其相应的属性值，则 **Open** 方法将产生错误。

1.3.5.5 ADO MD 属性

属性	说明
ActiveConnection(See 1.3.5.5.1)	指示当前单元集或目录当前属于哪个 ADO **Connection** 对象。

Caption(See 1.3.5.5.2)	指示显示 **Level** 或 **Member** 对象时所使用的文本标题。
ChildCount(See 1.3.5.5.3)	指示在分级结构中当前 **Member** 对象是其父的成员的数目。
Children(See 1.3.5.5.4)	返回在分级结构中当前 **Member** 是其父的 **Members** 的集合。
Count(See 1.1.4.6.16)	指示集合中的对象数目。
Depth(See 1.3.5.5.5)	指示在 **Level** 和分级结构的根级别之间的级别数目。
Description(See 1.3.5.5.6)	返回当前对象的文本说明。
DimensionCount(See 1.3.5.5.7)	指示轴上的维数。
DrilledDown(See 1.3.5.5.8)	指示成员是否已“到头”。
FilterAxis(See 1.3.5.5.9)	指示当前单元集的过滤器信息。
FormattedValue(See 1.3.5.5.10)	指示单元值的格式化显示。
LevelDepth(See 1.3.5.5.11)	指示在分级结构的根和成员之间的级别数。
LevelName(See 1.3.5.5.12)	指示成员的级别名。
Name(See 1.3.5.5.13)	指示对象的名称。
Ordinal（单元）(See 1.3.5.5.14)	通过单元集中的位置唯一标识单元。
Ordinal（位置）(See 1.3.5.5.15)	唯一标识在轴上的位置。
Parent(See 1.3.5.5.16)	指示在分级结构中是当前成员的父的成员。
ParentSameAsPrev(See 1.3.5.5.17)	指示该位置成员的父是否与直接在前的成员的父相同。
Source(See 1.3.5.5.18)	指示单元集中数据的源。
State(See 1.3.5.5.19)	指示单元集的当前状态。
Type(See 1.3.5.5.20)	指示当前成员的类型。
UniqueName(See 1.3.5.5.21)	指示当前对象的明确名称。
Value(See 1.3.5.5.22)	指示当前单元的值。

1.3.5.5.1 ActiveConnection 属性 (ADO MD)

指示当前单元集或目录当前属于哪个 ADO Connection(See 1.1.4.2.2) 对象。

设置和返回值

设置或返回变体型值，包含定义连接或 **Connection** 对象的字符串。默认为空。

说明

可以将该属性设置为有效的 ADO **Connection** 对象或有效的连接字符串。当把该属性设置为连接字符串时，提供者将使用该定义创建新的 **Connection** 对象并打开该连接。

如果使用 Open(See 1.3.5.4.3) 方法的 **ActiveConnection** 参数打开 Cellset(See 1.3.5.2.4) 对象，则 **ActiveConnection** 属性将继承该参数的值。

把 Catalog(See 1.3.5.2.2) 对象的 **ActiveConnection** 属性设置为 **Nothing**，将释放关联的数据，包括在 CubeDefs(See 1.3.5.3.2) 集合和任何相关的 Dimension(See 1.3.5.2.6)、Hierarchy(See 1.3.5.2.7)、Level(See 1.3.5.2.8) 和 Member(See 1.3.5.2.9) 对象中的数据。关闭被用来打开 **Catalog** 的 **Connection** 对象，与将 **ActiveConnection** 属性设置为 **Nothing** 效果相同。

如果试图更改已打开的 **Cellset** 对象的 **ActiveConnection** 属性，将出现错误。

> **注意** 在 Visual Basic® 中，在将 **ActiveConnection** 属性设置为 **Connection** 对象时，请记住使用 **Set** 关键字。如果省略了 **Set** 关键字，则实际上是在把 **ActiveConnection** 属性设置为 **Connection** 对象的默认属性：**ConnectionString**。代码将工作；但是，需要为数据源创建额外连接，这样做可能产生不利结果。

当使用 MSOLAP 数据提供者时，请将连接字符串中的数据源设置为服务器名，并把初始目录设置为数据源的目录名。要连接到与服务器连接断开的立方文件，请将位置设置为 .CUB 文件的完整路径。在任一种情况下，均请将提供者设置为提供者名。例如，如下字符串使用 MSOLAP 提供者连接到名为 Servername 的服务器上的名为 Bobs Video Store 的目录：

```
"Data Source=Servername;Initial Catalog=Bobs Video Store;Provider=msolap"
```

如下字符串连接到位于 C:\MSDASDK\samples\oledb\olap\data\bobsvid.cub 的本地立方文件：

```
"Location=C:\MSDASDK\samples\oledb\olap\data\bobsvid.cub;Provider=msolap"
```

1.3.5.5.2 Caption 属性 (ADO MD)

指示在显示 Level(See 1.3.5.2.8) 或 Member(See 1.3.5.2.9) 对象时使用的文本标题。

返回值

返回只读的字符串。

1.3.5.5.3 ChildCount 属性 (ADO MD)

指示在分级结构中当前 Member(See 1.3.5.2.9) 对象是其父的成员的数目。

返回值

返回只读的长整型整数。

说明

使用 **ChildCount** 属性返回 **Member** 所拥有的子的估算值。**Member** 实际的子可由 Children(See 1.3.5.5.4) 属性返回。

对于 Position(See 1.3.5.2.10) 对象的 **Member** 对象，返回的最大值为 65536。如果实际子的数目超过 65536，返回值仍为 65536。因此，应用程序将把 65536 的 **ChildCount** 解释为等于或大于 65536 个子。

对于 Level(See 1.3.5.2.8) 对象的 **Member** 对象，使用 **Children** 集合上的 ADO 集合 Count(See 1.1.4.6.16) 属性确定子的确切数目。如果集合中的子数目很大，则确定子的确切数目会比较慢。

1.3.5.5.4 Children 属性 (ADO MD)

返回在分级结构中当前 Member(See 1.3.5.2.9) 是其父的 Members(See 1.3.5.3.6) 的集合。

返回值

返回只读的 **Members** 集合。

说明

Children 属性包含当前 **Member** 是其分级结构中的父的 **Members** 集合。叶级别 **Member** 对象在 **Members** 集合中无子成员。仅在属于 Level(See 1.3.5.2.8) 对象的 **Member** 对象中支持该属性。当从属于 Position(See 1.3.5.2.10) 对象的 **Member** 对象引用该属性时，将出现错误。

1.3.5.5.5 Depth 属性 (ADO MD)

指示在 Level(See 1.3.5.2.8) 和分级结构的根级别之间的级别数目。

返回值

返回只读整数。

说明

在分级结构根位置的 **Level** 其 **Depth** 值为零 (0)。

1.3.5.5.6 Description 属性 (ADO MD)

返回当前对象的文本说明。

返回值

返回只读的字符串。

说明

对于 Member(See 1.3.5.2.9) 对象，**Description** 只应用于尺度和公式成员。**Description** 对所有其它类型的成员均返回空字符串 ("")。

有关各种类型成员的详细信息，请参阅 Type(See 1.3.5.5.20) 属性。

仅在属于 Level(See 1.3.5.2.8) 对象的 **Member** 对象中支持该属性。当从属于 Position(See 1.3.5.2.10) 对象的 **Member** 对象引用该属性时，将出现错误。

1.3.5.5.7 DimensionCount 属性 (ADO MD)

指示轴上的维数。

返回值

返回只读的长整型整数。

1.3.5.5.8 DrilledDown 属性 (ADO MD)

指示成员是否已“到头”。

返回值

返回只读的布尔值。如果轴上无当前成员的子成员，**DrilledDown** 将返回 **True**。如果在轴上有当前成员的一个或多个子成员，**DrilledDown** 将返回 **False**。

说明

使用 **DrilledDown** 属性确定在轴上紧随该成员是否至少还有一个该成员的子。该信息在显示成员时有用。

只在属于 Position(See 1.3.5.2.10) 对象的 Member(See 1.3.5.2.9) 对象中支持该属性。当从属于 Level(See 1.3.5.2.8) 对象的 **Member** 对象引用该属性时，将出现错误。

1.3.5.5.9 FilterAxis 属性 (ADO MD)

指示有关当前单元集的过滤器信息。

返回值

返回只读的 Axis(See 1.3.5.2.1) 对象。

说明

使用 **FilterAxis** 属性返回有关用于切割数据的维的信息。**Axis** 的 DimensionCount(See 1.3.5.5.7) 属性返回切割器维的数目。该轴通常只有一行。

通过 FilterAxis 所返回的 **Axis** 不包含在 Cellset(See 1.3.5.2.4) 对象的 Axes(See 1.3.5.3.1) 集合中。

1.3.5.5.10 FormattedValue 属性 (ADO MD)

指示单元值的格式化显示。

返回值

返回只读的字符串。

说明

使用 **FormattedValue** 属性获得 Cell(See 1.3.5.2.3) 对象中 Value(See 1.3.5.5.22) 属性的格式化显示值。例如，如果单元的值为 1056.87，并且该值表示的是美元数值，则 **FormattedValue** 将是 $1,056.87。

1.3.5.5.11 LevelDepth 属性 (ADO MD)

指示在分级结构的根和成员之间的级别数。

返回值

返回只读的长整型整数。

说明

使用 **LevelDepth** 属性确定 Member(See 1.3.5.2.9) 对象到分级结构根级别之间的距离。根级别处成员的 **LevelDepth** 为 0。它与 Level(See 1.3.5.2.8) 对象的 Depth(See 1.3.5.5.5) 属性相对应。

1.3.5.5.12 LevelName 属性 (ADO MD)

指示成员的级别名称。

返回值

返回只读的字符串。

说明

使用 **LevelName** 属性检索成员所属的级别名称。它与 Level(See 1.3.5.2.8) 对象的 Name(See 1.3.5.5.13) 属性相对应。

1.3.5.5.13 Name 属性 (ADO MD)

指示对象的名称。

返回值

返回只读的字符串。

说明

可以按序号引用来检索对象的 **Name** 属性，然后可以直接按名称引用该对象。例如，如果 cdf.CubeDefs(0).Name 产生“Bobs Video Store”，则可以按 cdf.CubeDefs("Bobs Video Store") 引用该 CubeDef(See 1.3.5.2.5)。

1.3.5.5.14 Ordinal 属性（ADO MD 单元）

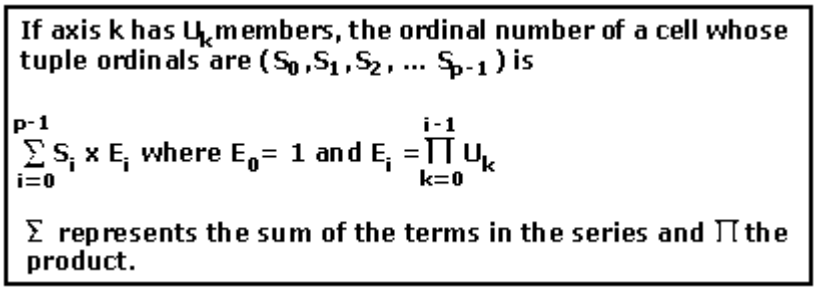
通过其在单元集中的位置唯一标识单元。

返回值

返回只读的长整型整数。

说明

单元的序号值用于唯一标识单元集中的单元。从概念上讲，单元集中单元的编号如同将单元集作为一个 p 维的数组，这里 p 是轴的数目。单元按以行为主的顺序从零开始编号。计算单元序号的公式为：



单元的序号值可用于 Cellset(See 1.3.5.2.4) 对象的 Item(See 1.3.5.4.2) 方法，来快速检索 Cell(See 1.3.5.2.3)。

1.3.5.5.15 Ordinal 属性（ADO MD 位置）

唯一标识在轴上的位置。

返回值

返回只读的长整型整数。

说明

Position(See 1.3.5.2.10) 对象的 **Ordinal** 与 Positions(See 1.3.5.3.7) 集合中的 **Position** 的索引相对应。

在 Cellset(See 1.3.5.2.4) 对象的 Item(See 1.3.5.4.2) 方法中，使用轴上 **Position** 的 **Ordinal** 可以快速地检索单元。

1.3.5.5.16 Parent 属性 (ADO MD)

指示在分级结构中是当前成员的父的成员。

返回值

返回只读的 Member(See 1.3.5.2.9) 对象。

说明

位于分级结构顶层的成员（根）无父。仅在属于 Level(See 1.3.5.2.8) 对象的 **Member** 对象中支持该属性。当从属于 Position(See 1.3.5.2.10) 对象的 **Member** 对象引用该属性时，会出现错误。

1.3.5.5.17 ParentSameAsPrev 属性 (ADO MD)

指示该位置成员的父是否与直接在前的成员的父相同。

返回值

返回只读的布尔值。

说明

仅在属于 Position(See 1.3.5.2.10) 对象的 Member(See 1.3.5.2.9) 对象中支持该属性。当从属于 Level(See 1.3.5.2.8) 对象的 **Member** 对象引用该属性时，会出现错误。

1.3.5.5.18 Source 属性 (ADO MD)

指示单元集中数据的源。

设置和返回值

设置或返回变体型。对于关闭的 Cellset(See 1.3.5.2.4) 对象该值是可读/写的，而对于打开的 **Cellset** 对象则是只读的。该变体型包含有效的字符串，如 MDX 查询。

1.3.5.5.19 State 属性 (ADO MD)

指示单元集的当前状态。

返回值

返回只读的长整型整数，指示 Cellset(See 1.3.5.2.4) 对象的当前状态。如下值有效：

常量	值	说明
adStateClosed	0	Cellset 关闭。
adStateOpen	1	Cellset 打开。

说明

要使用常量名，必须在工程中引用 ADO 类型库。详细信息，请参阅通过 ADO MD 使用 ADO(See 1.3.3)。

1.3.5.5.20 Type 属性 (ADO MD)

指示当前成员的类型。

返回值

返回只读的枚举型值。

Type 属性可以是如下值：

常量	值	说明
adMemberlRegular	1	默认。成员代表业务实体的实例。
AdMemberMeasure	2	成员属于尺度维。代表数量属性。
adMemberFormula	4	使用公式表达式计算成员。
adMemberAll	8	成员代表级别的所有成员。
adMemberUnknown	16	类型无法确定。

说明

仅在 Level(See 1.3.5.2.8) 对象的 Member(See 1.3.5.2.9) 对象中支持该属性。当从 Position(See 1.3.5.2.10) 对象的 **Member** 对象引用该属性时，会出现错误。

1.3.5.5.21 UniqueName 属性 (ADO MD)

指示当前对象的明确名称。

返回值

返回只读的字符串。

1.3.5.5.22 Value 属性 (ADO MD)

指示当前单元的值。

返回值

返回只读的变体型。

1.3.5.6 ADO MD 范例

使用如下范例主题可以了解部分 ADO MD 编程元素。

连接多维据源

Connection 范例(See 1.3.5.6.1)演示如何连接 OLAP（联机分析过程）数据库服务器或本地 OLAP 立方文件。

访问 ADO MD 集合

CubeDef 范例(See 1.3.5.6.2)演示如何显示 Catalog(See 1.3.5.2.2) 中 CubeDef(See 1.3.5.2.5) 的维的名称。

获得单元集

单元集范例(See 1.3.5.6.3)演示如何执行返回单元集的 MDX 查询。该范例显示单元集中成员的标题和单元的格式化值。

访问单元

单元范例(See 1.3.5.6.4)演示检索指定单元的三种方法：通过序号位置、通过轴数值位置和通过成员名称。

1.3.5.6.1 Connection 范例 (ADO MD)

如下代码演示如何使用连接字符串设置 ActiveConnection(See 1.3.5.5.1) 属性。该代码假定在名为 Servername 的 OLAP 服务器上有一个名为 Bob Video Store 的数据源，并且使用的是 MSOLAP 数据提供者。

```
Dim cat As New ADOMD.Catalog
cat.ActiveConnection = "Data Source=Servername;" + _
   "Initial Catalog=Bobs Video Store;Provider=msolap;"
```

如下代码也演示如何使用连接字符串设置 **ActiveConnection** 属性。但是，该代码使用 MSOLAP 数据提供者连接到本地立方文件，而不是连接到名为 Servername 的 OLAPT 服务器。

```
Dim cat As New ADOMD.Catalog
cat.ActiveConnection = _
"Location=C:\MSDASDK\samples\oledb\olap\data\BobsVid.cub;" + _
"Provider=msolap;"
```

如下代码演示如何将 **ActiveConnection** 属性设置为标准的 ADO Connection(See 1.1.4.2.2) 对象。要使用 ADO **Connection** 对象，必须在工程中引用 ADO 类型库。

```
Dim cnn  As New ADODB.Connection
Dim cat  As New ADOMD.Catalog
Cnn.Open "Data Source=Servername;" + _
   "Initial Catalog=Bobs Video Store;Provider=msolap;"
Set cat.ActiveConnection = cnn
```

注意　　在将 **ActiveConnection** 属性设置为 **Connection** 对象时，记住要使用 **Set** 关键字。如果没有使用 **Set** 关键词，其结果是将 **ActiveConnection** 属性设置成 **Connection** 的默认属性：ConnectionString(See 1.1.4.6.14)。代码将会工作，但将创建与数据源的其他连接，最终得到相反的结果。

1.3.5.6.2 CubeDef 范例 (ADO MD)

如下代码范例演示如何访问 CubeDefs(See 1.3.5.3.2) 和 Dimensions(See 1.3.5.3.3) 集合。该范例打印立方中每个维的名称。该代码假定在名为 Servername 的 OLAP 服务器上存在名为 Bobs Video Store 的数据源。

```
Dim cat      As New ADOMD.Catalog
Dim cdf      As ADOMD.CubeDef
Dim I        As Integer
cat.ActiveConnection = "Data Source=Servername;" + _
   "Initial Catalog=Bobs Video Store;Provider=msolap;"
Set cdf = cat.CubeDefs("Bobs Video Store")
For i = 0 To cdf.Dimensions.Count - 1
   Debug.Print cdf.Dimensions(i).Name
Next
```

1.3.5.6.3 Cellset 范例 (ADO MD)

如下代码演示如何打开 Cellset(See 1.3.5.2.4)、打印每个位置的 Members(See 1.3.5.3.6) 的标题和打印 Cell(See 1.3.5.2.3) 的格式化值。该代码假定在名为 Servername 的 OLAP 服务器上存在名为 Bobs Video Store 的数据源。

```
Dim cat      As New ADOMD.Catalog
Dim cst      As New ADOMD.Cellset
Dim axs      As ADOMD.Axis
Dim I        As Integer
Dim j        As Integer
cat.ActiveConnection = "Data Source=Servername;" + _
   "Initial Catalog=Bobs Video Store;Provider=msolap;"
cst.Source = "SELECT [Product Category].MEMBERS ON ROWS," + _
                  "[Store State].MEMBERS ON COLUMNS" + _
             "FROM  [Bobs Video Store]" + _
             "WHERE ([Quantity])"
Set cst.ActiveConnection = cat.ActiveConnection
cst.Open
For i = 0 To cst.Axes(0).Positions.Count - 1
   Debug.Print cst.Axes(0).Positions(i).Members(0).Caption & " ";
```

```
   Set axs = cst.Axes(1)
   For j = 0 To axs.Positions.Count - 1
      Debug.Print "     " & axs.Positions(j).Members(0).Caption & _
         " " & cst(i, j).FormattedValue
   Next
Next
cst.Close
```

1.3.5.6.4 Cell 范例 (ADO MD)

```
Dim cat     As New ADOMD.Catalog
Dim cst     As New ADOMD.Cellset
Dim cll     As ADOMD.Cell
cat.ActiveConnection = "Data Source=Servername;" + _
   "Initial Catalog=Bobs Video Store;Provider=msolap;"
cst.Source = "SELECT [Product Category].MEMBERS ON ROWS," + _
                 "[Store State].MEMBERS ON COLUMNS" + _
             "FROM  [Bobs Video Store]" + _
             "WHERE ([Quantity])"
Set cst.ActiveConnection = cat.ActiveConnection
cst.Open
' 使用序号位置检索单元。
Set cll = cst(10)
' 使用轴数值位置检索单元。
Set cll = cst(2,2)
' 使用成员名检索单元。
' MSOLAP 提供者不支持该方法，但是对于支持由成员名
' 访问单元的提供者，该方法可以包括在内。
' Set cll = cst("Childrens", "New York State")
```

2. Addenda

2.1 ActiveX?

使软件组件能够在网络环境中交互作用而与创建组件的语言无关的一套技术。实现 ActiveX? 的基础是“组件对象模型”(COM)。

2.2 ADISAPI

高级数据 Internet 服务器应用程序编程接口。ISAPI DLL 可提供语法分析、自动控制、记录集调度和 MIME 打包。ADISAPI 组件通过由 Internet Information Server (IIS) 提供的 API 工作。

2.3 合计函数 (aggregate function)

函数，如 Count、Avg 和 Var，用于创建计算总数的查询。在编写表达式和编程时，可以使用 SQL 合计函数（包括上面所列的三个函数）和域合计函数进行各种统计。

2.4 别名 (Alias)

在 SQL SELECT 语句中用户指定给列或表达式的可选的名称，通常较短并有一定含义。例如 BobSales 是如下 SELECT 语句中的别名：“Select wr-Sales as BobSales from SalesDB”。可以用别名动态分派列，以便控制 DataControl 对象的绑定。

2.5 房间线程 (apartment thread)

用于执行对配置为“房间线程”的组件对象的调用。每个对象在其生命期内“生活在房间”（线程）中。所有对该对象的调用均在房间线程上执行。

2.6 异步操作 (Asynchronous operation)

由代码（如查询）启动的操作。在操作完成之前，代码执行将继续。请参阅同步操作(See 2.48)。

2.7 业务对象 (Business Object)

对数据进行检索和处理的 ActiveX® 组件。业务对象通常位于中间层。

Remote Data Service 提供默认的中间层业务对象 **RDSServer.DataFactory**，用于接收客户端请求并提供对指定数据源的读写访问，但不包含任何验证或业务规则逻辑。

用户可以创建能够提供与 **RDSServer.DataFactory** 功能相同的自定义业务对象，并且对应用程序的业务规则进行封装。

2.8 业务规则 (Business Rule)

验证编辑、登录确认、数据库查找、策略以及算法转换的组合，共同构成企业的业务行为方式。也称为“业务逻辑”。

2.9 子 (child)

子是分级结构中另有一个节点位于其上（靠近根）的节点。

2.10 类 ID (CLSID)

“通用唯一标识符”(UUID)，用于标识 COM 组件。每个 COM 组件在 Windows 注册表中都有自己的 CLSID，以便让其他应用程序加载。

2.11 客户端层 (Client Tier)

逻辑层，代表本地计算机。客户端层是将数据呈现给用户或处理用户输入的应用程序或系统一部分。客户端也称为前端，它并不执行数据函数，而是通过输入向服务器请求数据，然后以一定的格式显示结果。参见中间层(See 2.32)、数据源层(See 2.23)。

2.12 组件对象模型 (COM)

一种跨平台的开放结构，用于开发基于面向对象技术的客户端/服务器应用程序。客户端可以通过在对象上实现的接口访问对象。COM 与语言无关，因此任何生成 ActiveX® 组件的语言都可以生成 COM 应用程序。

2.13 组件 (Component)

建立在 ActiveX® 技术上的代码的独立单元，用于通过特定的接口提供特定的一组服务。组件提供客户端在运行时所请求的对象。在 Remote Data Service 中，当组件包括支持业务进程的关键字逻辑时也被称为“业务对象”。

2.14 连接缓冲池 (Connection pooling)

基于使用预分配资源的集合的性能优化，例如对象或数据库连接。缓冲池可产生更有效率的资源分配。

2.15 游标 (cursor)

返回记录并定义 ADO 如何访问和使用这些记录的机制。游标控制由其他用户对数据库所作的记录定位、数据的可更新性和更改的可见性。

2.16 数据识别控件 (Data-aware Control)

能够使用数据库中数据的控件。一旦通过 **RDS.DataControl** 对象将该控件绑定到数据库，则该控件称为“绑定控件”。

2.17 数据提供者 (data provider)

拥有数据并使用表格格式直接将其显露给应用程序的软件。请参阅服务提供者(See 2.45)、服务组件(See 2.44)。

2.18 数据源 (Data Source)

应用程序用来请求连接系统 ODBC 数据源的名称。它指定数据源名称 (DSN) 所映射的计算机名和（可选）数据库。
系统数据源是任何使用计算机的人都可以使用的数据源。用于 Web 服务器的数据源应为系统数据源。

2.19 DCOM（分布式组件对象模型）

使 ActiveX® 组件能通过网络直接互相通讯的对象协议。DCOM 与语言无关，因此任何可以生成 ActiveX 组件的语言都可以生成 DCOM 应用程序。

2.20 设计时 (Design Time)

是指在开发环境中通过添加控件、设置控件或窗体属性等方法，建立应用程序的时间。与此相应的“运行时”，是指可以象用户那样与应用程序交互作用的时间。
在制作 Web 页时，可以在对象的 OBJECT 标记中设置设计时属性。可以在脚本代码中设置运行时属性（例如 VBScript）。

2.21 未连接记录集 (Disconnected recordset)

在客户端缓冲区中不再活动连接服务器的记录集。如果需要使用原始数据源，例如更新数据，则需要重新建立连接。

2.22 DLL（动态链接库）

包含一个或多个函数的文件，这些函数的编译、链接和存储独立于使用它们的进程。在进程启动或运行时，操作系统将 DLL 映射到调用进程的地址空间。

2.23 数据源层 (Data Source Tier)

逻辑层，代表运行 DBMS（例如 SQL Server 数据库）的计算机。参见客户端层(See 2.11)、中间层(See 2.32)。

2.24 动态属性 (dynamic property)

特定于数据提供者或游标服务的属性。在适当的时候（“动态地”），对象的 **Properties** 集合被它们自动充填。对象直到通过特定数据提供者连接到数据源后，才具有动态属性。

2.25 事件 (event)

由对象识别的操作，可以针对它编写作出响应的代码。事件可以在执行其他操作时由命令执行、事务完成、记录集定位和数据更新等产生。要执行的代码被附加到相应的事件处理程序。

2.26 孙子合计 (grandchild aggregate)

合计计算在子集的记录中的值、或在子集中记录的子集的记录的值，等等。

2.27 事件处理程序 (event handler)

事件处理程序可以包含当事件发生时被执行的代码。

2.28 ISAPI

Internet 服务器应用程序编程接口。Internet 服务器（如运行 Microsoft® Internet Information Server (IIS) 的 Windows NT® Server）的函数集合。

2.29 调度 (Marshaling)

跨越线程或进程边界将接口方法参数打包并发送的过程。

2.30 远程数据服务(Remote Data Service)

一种基于 Web 的高性能技术，可将数据库连接和企业数据发布功能应用到 Internet 和 Intranet 应用程序中。

2.31 Microsoft Transaction Server

基于组件的事务处理系统，用于开发部署并管理高性能、可缩放和健壮的企业、Internet 和 Intranet 的服务器应用程序。Microsoft® Transaction Server 定义了应用程序编程模型，用于开发分布式、基于组件的应用程序。它也提供了用于部署和管理这些应用程序的运行时基础结构。

2.32 中间层 (Middle Tier)

也称作“应用程序服务器层”，是用户接口或 Web 客户端与数据库之间的逻辑层。典型情况下 Web 服务器位于该层，业务对象在此实例化。中间层是生成并操作接收信息的业务规则和函数的集合。它们通过业务规则（可以频繁更改）完成该任务，并由此被封装到在物理上与应用程序程序逻辑本身相独立的组件中。请参见客户端层(See 2.11)、数据源层(See 2.23)。

2.33 MIME

在 Internet 上实现发布和读取二进制数据的标准。二进制数据文件的文件头包含数据的 MIME 类型，由此通知客户端程序（例如 Web 浏览器和邮件软件包）需要按与处理纯文本不同的方式来处理该数据。例如，包含 JPEG 图形的 Web 文档的文件头包含对应于 JPEG 文件格式的 MIME 类型，由此通知浏览器使用其 JPEG 查看程序来显示文件（如果有的话）。

2.34 对象变量 (Object Variable)

包含对对象进行引用的变量。例如，objCustomObject 是指向 CustomObject 类型对象的变量：

```
Set objCustomObject = ADS1.CreateObject("CustomObject", _
   "http://SalesWeb")
```

2.35 ODBC（开放式数据库连接）

用于连接不同数据源的标准编程语言接口。一般通过“控制面板”进行访问，并在此分配数据源名称 (DSN) 以使用特定的 ODBC 驱动程序。

2.36 参数化命令 (parameterized command)

带有参数的命令，该参数是可以由您的代码设置的值、并可在更改后从一个命令执行进入下一个命令执行。这些值由命令传递给提供者。

2.37 父 (parent)

表关系的控制端。

2.38 父 - 子关系 (parent-child relationship)

在分级结构中，父是位置较高并与一个或多个子直接关联的一个等级。子则是位置较低并必须有一个父的一个等级。

2.39 持久 (Persist)

保存 COM 对象的当前状态，如保存到文件或记录集。

2.40 代理 (Proxy)

特定接口的对象，用于提供客户端在调用不同执行环境中运行（例如在不同线程或其他进程中）的应用程序对象时所需的参数调度和通讯。代理位于客户端，并与位于正在被调用的应用程序对象处的、相应的通讯模块进行通讯。

2.41 记录集 (Recordset)

通过 ActiveX® Data Objects (ADO) 所提供的行集合。定位行和编辑数据的主要机制。它是包含了行集合以及用于访问和更新数据的方法和属性的对象。

2.42 行集合 (rowset)

数据库中具所有有相同字段模式的行的集合。行集合可以代表表格中所有或部分字段。行集合也可以代表由查询或由多个表格的合并所创建的虚拟表格。

2.43 运行时 (Run Time)

正在运行应用程序的时间，在运行时可以与应用程序交互作用。与此对应的设计时，是可以创建对象和修改其设计的时间。
在制作 Web 页时，可以用 HTML 在对象的 OBJECT 标记中设置设计时属性。可以在脚本代码（例如 VBScript）中设置运行时属性。

2.44 服务组件 (service component)

服务组件同时与服务提供者和数据提供者一起工作，以便进一步将服务增加到您的应用程序中。请参阅数据提供者(See 2.17)、服务提供者(See 2.45)。

2.45 服务提供者 (service provider)

通过产生和消费数据对服务进行封装的软件，由此可增加 ADO 应用程序中的功能。提供者并不直接显露数据，而是提供服务，例如查询处理。服务提供者可以处理由数据提供者提供的数据。请参阅数据提供者(See 2.17)、服务组件(See 2.44)。

2.46 单线程控件 (Single-threaded control)

使所有对象均按单线程执行的模型。

2.47 通讯模块 (Stub)

特定接口的对象，它提供应用程序对象所需的参数调度和通讯，以接受运行于不同执行环境（如不同的线程或进程）的调用。通讯模块与应用程序对象处于同一位置并且与相应的代理通讯，该代理位于调用它的客户端处。

2.48 同步操作 (Synchronous operation)

由代码（如查询）启动并在另一个操作开始之前完成的操作。请参阅异步操作(See 2.6)。

2.49 表图 (tablegram)

用于将结果集合传输到客户端（或从客户端传输出来）的专用格式。

2.50 variant

在 Automation 中作为 VARIANT 数据类型的实例，可以表示很多不同类型的值，如整型、浮点型、布尔型、字符串型、指针型等等。在 Visual Basic 中是数据类型。

2.51 Web 服务器

向 Intranet 和 Internet 用户提供 Web 服务和网页的计算机。Internet Information Server 被安装在 Web 服务器上，服务器端的业务对象一般在此实例化。