11.3创建SQLServer2005服务器段物件 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/140/2021_2022_113_E5_88_9B_E5_BB_BA_c100_140294.htm 通过 Visual Studio 2005创建新的 SQL Server 2005 对象项目后，编辑环境会自动参照相关的组件，如 sqlaccess.dll 和 System.Data.dll 等。同时包含存储过程（Stored Procedure）、触发器（Trigger）、用户自定义函数（User-Defined Function）、用户自定义数据类型（User-Defined Type）以及集合函数（Aggregation）等五种对象的程序编写模板。并辅助将编写与编译好的程序集部署到 SQL Server 2005 内，它同时也会上载原始程序代码与 .pdb 文件，以支持集成开发环境调试的功能。SQL Server 2005 与 Visual Studio 2005 提供了很好的协同合作以方便调试(调试的方式可以参照上一节)，不管你是通过 TDS（Tabular Data Stream）还是 HTTP 协议调用 SQL Server 2005，都可以用往熟悉的方式完成调试。在此就先来看如何建立存储过程。 11.3.1 创建存储过程在项目中加入类型后，定义public static（VB.NET 的声明是 Shared）函数，也就是 SQL Server 在调用这个函数时，无须先建立该类型的实例(instance)，就可以直接调用该函数。在函数前通过 SqlProcedure 特性(Attribute)告知该函数是用作创建存储过程。在同一个类型中，可以声明多个函数，注册该函数到 SQL Server 时，会用到程序集在 SQL Server 注册的名称、命名空间(namespace)、类型，以及函数名称，以指定函数当作某个存储过程。当你通过 Visual Studio 2005 创建“ SQL Server Project ”项目时，可以通过在项目中新建“ Stored Procedure ”模板的方式加入含有相关定义

的类，你只须将函数的主体编写完毕即可。在此示范一个通过网络访问数据的简单存储过程范例，内容如程序代码列表11-3所示：程序代码列表11-3 通过.NET的XPathNavigator 解析 XML 文件，并以记录的方式返回Partial Public Class myStoredProcedures _ Public Shared Sub RetrieveRSS(ByVal strURL As SqlString) Try Dim strRSS As String 通过 MSDN 提供的 RSS(Really Simple Syndication) 取得关于 SQL Server 各种技术信息的列表 将该 XML 的文件转成一条条的记录，通过 Pipe 返回到前端 If strURL.IsNull Then strRSS = "http://msdn.microsoft.com/sql/rss.xml" Else strRSS = strURL.Value End If ********** 通过 XPathDocument 类型 通过 Internet 访问XML文件 Dim doc As New XPathDocument(strRSS) ********** Dim nav As XPathNavigator = doc.CreateNavigator Dim i As XPathNodeIterator = nav.Select("//item") Dim rss_results(4) As SqlMetaData rss_results(0) = New SqlMetaData("名称", SqlDbType.NVarChar, 250) rss_results(1) = New SqlMetaData("发布日期", SqlDbType.DateTime) rss_results(2) = New SqlMetaData("说明", SqlDbType.NVarChar, 2000) rss_results(3) = New SqlMetaData("连接", SqlDbType.NVarChar, 300) rss_results(4) = New SqlMetaData("RSS 原始节点", SqlDbType.Xml) 定义存放结果的每一条记录的数据结构 Dim record As New Microsoft.SqlServer.Server.SqlDataRecord(rss_results)

********** 将SQL Server收到的数据结果返回到前端 Dim pipe As SqlPipe = SqlContext.Pipe 将 Record 结构传回到前端 pipe.SendResultsStart(record) While i.MoveNext 将 XML 数据通过 XPath 语法取出，并放入记录中 record.SetString(0, CType(i.Current.Evaluate("string(title[1]/text())"), String)) record.SetDateTime(1, DateTime.Parse(_ CType(i.Current.Evaluate("string(pubDate[1]/text())"), String))) record.SetString(2, _ CType(i.Current.Evaluate("string(description[1]/text())"), String)) record.SetString(3, CType(i.Current.Evaluate("string(link[1]/text())"), String)) record.SetString(4, CType(i.Current.InnerXml, String)) 将结果通过 Pipe 传回到调用端 pipe.SendResultsRow(record) End While ********** pipe.SendResultsEnd() Catch ex As Exception Dim pipe As SqlPipe = SqlContext.Pipe pipe.Send(ex.ToString()) End Try End SubEnd Class在上述程序代码中，我们通过 XPathDocument 对象实例，通过网络取回微软放在 MSDN 网站上关于各产品的技术文件，在此取得的是与 SQL Server 有关的技术资料。由于该文件是 XML 格式，所以可通过 XPathNavigator 和XPathNodeIterator 对象实例查找文件内所需的节点元素 。以 SqlMetaData 对象实例定义每条记录各字段的结构后，赋予 SqlMetaData 实例所形成的数组到 SqlDataRecord 对象的构造函数中，据此创建出符合该结构的单录记录实例，接着以 XPathNodeIterator 实例取得整篇 XML

文件中所有的 item 节点，逐笔赋予节点内子节点的数据到 SqlDataRecord 实例，并直接通过 SqlPipe 对象实例将该条记录返回到用户手上。你可以通过 Visual Studio 2005 所提供的部署功能将程序代码列表 11-3 所建立的函数注册成 SQL Server 内的存储过程，也可以通过以下的 T-SQL 语法完成注册。程序代码列表11-4 通过T-SQL 注册程序集和存储过程--让该数据库的 .NET 对象可以访问 SQL Server 之外的资源ALTER DATABASE AdventureWorks SET TRUSTWORTHY ON--将 Assembly 加入到 SQL ServerCREATE ASSEMBLY YukonCLR FROM C:\BookSamples\SQL 2005 Dev\Ch11_Clr\YukonCLR\bin\YukonCLR.dll WITH PERMISSION_SET=external_accessGO --创建一个名为 RetrieveRSS 的存储过程CREATE PROC RetrieveRSS @strURL NVARCHAR(100)=NULLASEXTERNAL NAME YukonCLR.[YukonCLR.myStoredProcedures].RetrieveRSS GO注册完成后，使用下列T-SQL语法调用存储过程，测试执行结果。EXEC RetrieveRSS由于范例中的存储过程会读取因特网上的 RSS XML 文件，此为 SQL Server 外部的资源，所以该组件注册时要设置 PERMISSION_SET 为 external_access。但若要使用 PERMISSION_SET 为 external_access 或 unsafe 等级的数据集时，该数据库的 TRUSTWORTHY 属性必须设为 ON。默认这个属性设置为 OFF，以此来防护 DBA 附加数据库时，如果该数据库已经存在了功能强大而且可以访问外部资源的 .NET 程序，这将会危害到数据库服务器的安全。由于附加数据库时，该数据库的 TRUSTWORTHY 属性为 OFF，所以这些用 .NET 编写的数据库对象将没有权限来执行。若 DBA 确认这

些对象没有问题，再自行打开该选项，以减少 .NET 所带来的风险。在指定组件中函数当作存储过程的完整名称结构如程序代码列表11-4：程序代码列表11-4注册到 SQL Server 2005 的 Assembly 名称 >.命名空间.类型名称] >.函数名称 >事实上，Visual Studio 2005 就是引用我们在各函数或类定义的 Attribute，先移除 SQL Server 服务器上已经存在的同名对象，然后用与我们手动注册相似的 T-SQL 语法完成注册。布署完毕后，通过 Sql 语法调用上述存储过程的结果如图11-13所示：在图11-13 中可以看到逐条返回来的记录结构与一般的SQL语法执行结果无异。前端应用程序调用时，并不知道该存储过程是以 T-SQL 实际操作的，有可能是通过 .NET 语言编写的。而在此我们通过 .NET CLR 扩展了 SQL Server 2005 的功能，让它可以访问网络上的数据。图11-13 通过存储过程显示 MSDN RSS 的记录若你要删除已注册的存储过程与程序集，记得要先删除存储过程，所有依附该程序集相关的对象全部删除后，最后才可以删除该程序集。语法如下：DROP PROCEDURE RetrieveRSSDROP ASSEMBLY YukonClr看完了存储过程后，接着来创建系统在某个数据表添加、修改、删除记录后，自动调用的触发器(Trigger)。11.3.2 创建触发器与前述创建存储过程的方式相同，在类中声明 public static 方法。通过 SqlTrigger 属性告知该方法是在某个数据表在添加、删除或修改时，SQL Server 需自动调用的触发器。并利用 SqlPipe 与 SqlCommand 两种对象分别将结果显示给前端应用程序，最后记录更改过程到我们自定义的数据表 tblAuditSalesPerson 中。简单的程序范例如程序代码列表11-5：程序代码列表11-5 通过 .NET的 SqlTriggerContext 对象取得

触发器的相关数据Public Class Trigger Target:="Sales.SalesPerson")> _Public Shared Sub TriggerEx() 通过 .NET Framework 2.0 版 ADO.NET 之 SQLCommand 对象， 可以更详尽地沟通与操作 SQL Server Using cnn As New SqlConnection("Context Connection=true") cnn.Open() Using sqlCmd As New SqlCommand sqlCmd.Connection = cnn 通过 SqlPipe 将运行结果接回原来 SQL Server 的输出 Dim sqlP As SqlPipe = SqlContext.Pipe Dim trigContext As SqlTriggerContext = SqlContext.TriggerContext 判断是哪一种行为触发 Trigger，纯粹示范 Trigger 状态与 SqlPipe 通过 SqlPipe 将结果直接与前端应用程序沟通。 但实际操作时，最好不要这么做，因为前端应用程序不会预期到 更新数据还会返回状态记录 Select Case trigContext.TriggerAction Case TriggerAction.Delete sqlCmd.CommandText = "SELECT * FROM DELETED" sqlP.ExecuteAndSend(sqlCmd) sqlP.Send("读出删除的数据") Case TriggerAction.Insert sqlCmd.CommandText = "SELECT * FROM INSERTED" sqlCmd.CommandText = "SELECT SalesPersonID,CommissionPct " amp. _ "d.CommissionPct [删除的数据], " amp. _ "FROM DELETED d JOIN INSERTED i ON d.SalesPersonID=i.SalesPersonID" sqlP.ExecuteAndSend(sqlCmd) sqlP.Send("读出删除和插入的数据") End Select 示范用，仅能 Auditing 单一条记录的更新 创建数据记录的 XML Dim strDelRow, strInsRow As String Dim strPK As String ‘ 判断触发 Trigger 的原因 Select Case trigContext.TriggerAction Case TriggerAction.Delete sqlCmd.CommandText = "0select * from 0deleted for xml auto,elements" strDelRow =

sqlCmd.ExecuteScalar().ToString() strInsRow = "没有数据" 抓取被更新数据记录的主键 strPK = "SELECT SalesPersonID FROM DELETED" Case TriggerAction.Insert sqlCmd.CommandText = "0select * from inserted for xml auto,elements" strInsRow = sqlCmd.ExecuteScalar().ToString() strDelRow = "没有数据" strPK = "SELECT SalesPersonID FROM INSERTED" Case Else sqlCmd.CommandText = "0select * from 0deleted for xml auto,elements" strDelRow = sqlCmd.ExecuteScalar().ToString() sqlCmd.CommandText = "0select * from inserted for xml auto,elements" strInsRow = sqlCmd.ExecuteScalar().ToString() strPK = "SELECT SalesPersonID FROM INSERTED" End Select sqlCmd.CommandText = strPK Dim strInsPK As String = sqlCmd.ExecuteScalar().ToString() 将 Auditing 数据记录加入到 Auditing 数据表 Dim strSQLAudit As String strSQLAudit = "INSERT tblAuditSalesPerson VALUES(" amp. ",N" amp. ",N" amp. _ ",SUSER_SNAME(),GETDATE())" sqlCmd.CommandText = strSQLAudit sqlCmd.ExecuteNonQuery() End Using cnn.Close() End UsingEnd Sub End Class在程序代码列表 11-5 中，我们分别访问 SQL Server 在触发程序环境中自动创建的 Inserted 和 Deleted 临时数据表。也就是如果用户删除记录，则被删除的记录会放到 Deleted 临时数据表，同理，添加的记录会放到 Inserted 临时数据表，而修改记录则等同先删除再添加，所以 Deleted 和 Inserted 两个临时数据表都可以访问。我们通过 SqlTriggerContext 对象来判断用户做的是添加、删除还是修改，以确定要访问 Inserted 还是 Deleted 数据表，或者是两者都访问。在此为了简化范例的内容，我们仅针对用户进行单一

条记录的添加、删除或修改的操作，。若一次更改多条记录，则须循环读取变更的内容，这一部分就让你自行发挥了J 而部署 trigEx 触发器的 T-SQL 语法如下：CREATE TRIGGER trigExON Sales.SalesPersonFOR INSERT,DELETE,UPDATEASEXTERNAL NAME YukonCLR.[YukonCLR.Trigger].TriggerEx通过 T-SQL 语法将 trigEx 部署到数据库后，你可以在 SQL Server Management Studio 的“ 对象资源管理器 ”窗口内的 Sales.SalesPerson 数据表节点下发现多一项 trigEx 节点。如图11-14 所示：图11-14 Sales.SalesPerson数据表底下的触发器 trigEx 对象部署完毕后，还需要创建记录更新所使用的数据表 tblAuditSalesPerson，内容如下所示：CREATE TABLE tblAuditSalesPerson(id int identity(1,1) primary key, PK int, Del xml, Ins xml, UserID nvarchar(100), ActionTime datetime)接下来通过下列 T-SQL语法，对 Sales.SqlesPerson 数据表添加、修改记录。INSERT Sales.SalesPerson(SalesPersonID,TerritoryID,CommissionPct) VALUES(123,2,0.2) --修改 SalesPerson 数据表以触发 TriggerUPDATE Sales.SalesPerson SET CommissionPct=0.8 WHERE SalesPersonID=268当添加、修改数据表时便会调用到 trigEx 触发器，通过 .NET 的 SqlPipe 对象将跟踪的结果返回 SQL Server 的查询引擎(Query Engine)执行，我们可以在 Management Studio 的“ 结果(Result) ”窗口查看触发器执行的结果。如图11-15、11-16 所示：图11-15仅显示修改语法所触发的程序执行结果：图11-15 通过“ 结果 ”窗口可查看通过 SqlPipe.ExecuteAndSend 方法执行 T-SQL 语法的结果切换到信息页签后可以显示通过 SqlPipe 类型实例 Send 方法送回来的

信息，显示如图11-16的结果：图11-16 通过“ 信息 ”窗口可查看通过 SqlPipe.Send 方法所送回的信息程序代码列表11-5 最后一段范例程序通过 SqlCommand 对象，取回我们想要的数据内容，重新组成 Insert 语法后，再通过 SqlCommand 对象交由数据库引擎执行，将监控的数据插入到我们事先建立的监控数据表。通过以下的语法查看刚刚我们新建立的 tblAuditSalesPerson 数据表：Select * from tblAuditSalesPerson结果如图11-17：图11-17 在更新记录后，查看tblAuditSalesPerson数据表的结果在图11-17中可以看到我们添加和修改记录时，通过上述的触发器所记录的数据细节。创建触发器的方式就介绍到此，我们接着再来看看如何编写用户自定义函数。