访问数据：ADO.NET

访问数据：ADO.NET

- 概述
- .NET数据供应器
- 直接访问数据
- 以DataSet访问数据
- LINQ to ADO.NET 概述

ADO.NET - 概述

- ADO.NET 是重要的应用程序级接口，用于在 Microsoft .NET 平台中提供数据访问服务。
- ADO.NET既能用来访问关系型数据，也能用来访问XML定义的数据。
- ADO.NET的类在Syste.Data命名空间中

ADO.NET - 概述

- ADO.NET 结构

  以前，数据处理主要依赖于基于连接的双层模型。随着数据处理越来越多地使用多层体系结构，程序员正在向断开方法转换，以便为他们的应用程序提供更好的可伸缩性。

  ADO.NET 3.0 用于访问和操作数据的两个主要组件是 .NET Framework 数据提供程序和 DataSet。

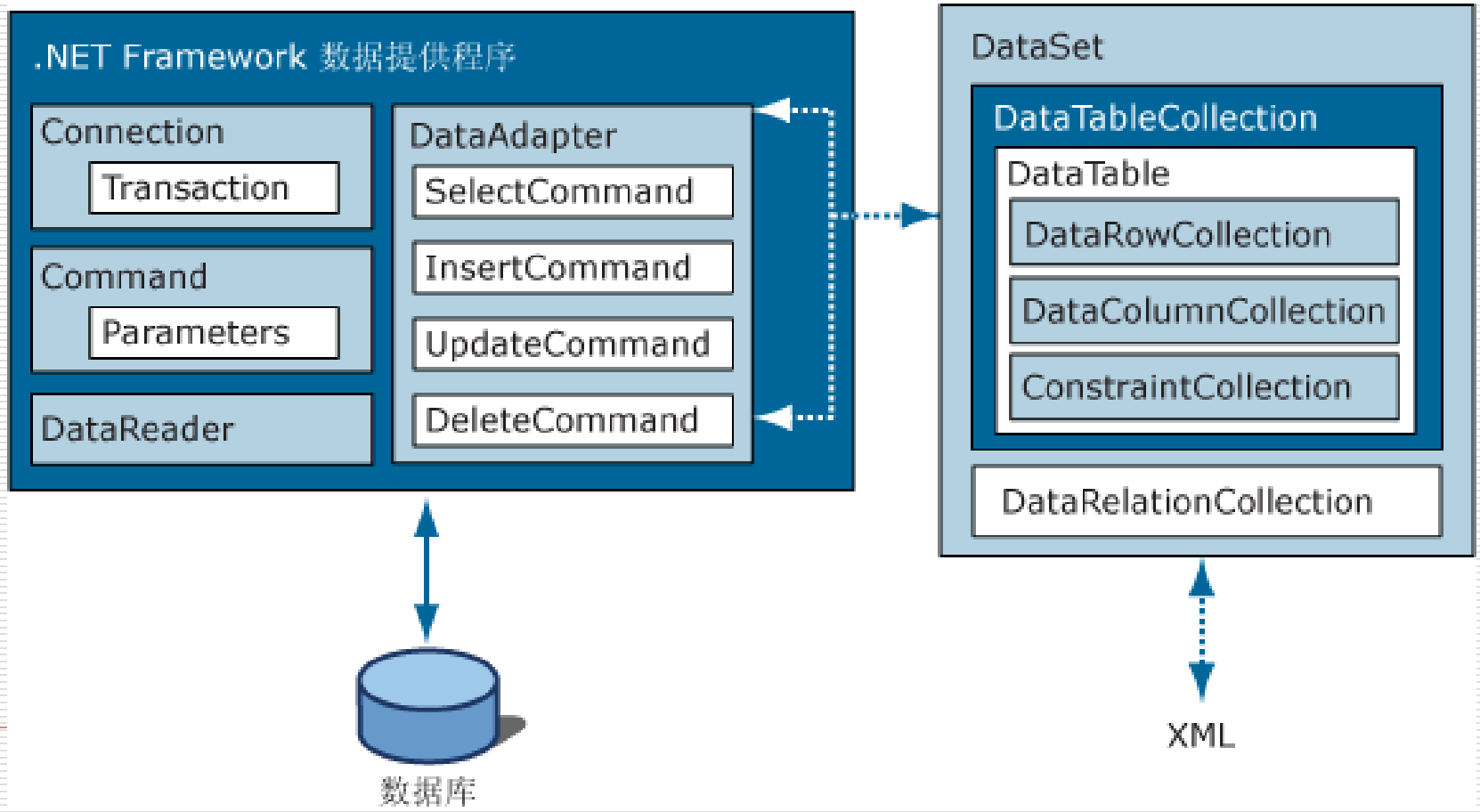
ADO.NET - 概述

- .NET Framework 数据提供程序用于连接到数据库、执行命令和检索结果。这些结果可直接处理、也可放置在 DataSet 中以便进一步处理。
- DataSet 是专门为独立于任何数据源的数据访问而设计的。因此，它可用于多种不同的数据源：用于 XML 数据，或数据库。

  DataSet 包含一个或多个 DataTable 对象的集合，这些对象由数据行和数据列以及有关 DataTable 对象中数据的主键、外键、约束和关系信息组成。

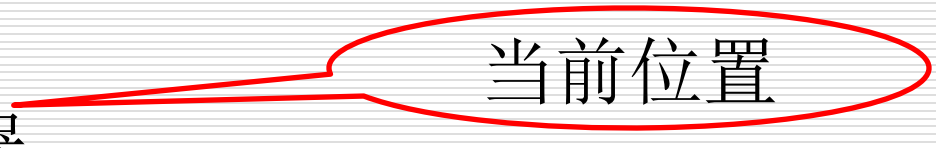
ADO.NET - 概述

- .NET Framework 数据提供程序和 DataSet 之间的关系



访问数据：ADO.NET

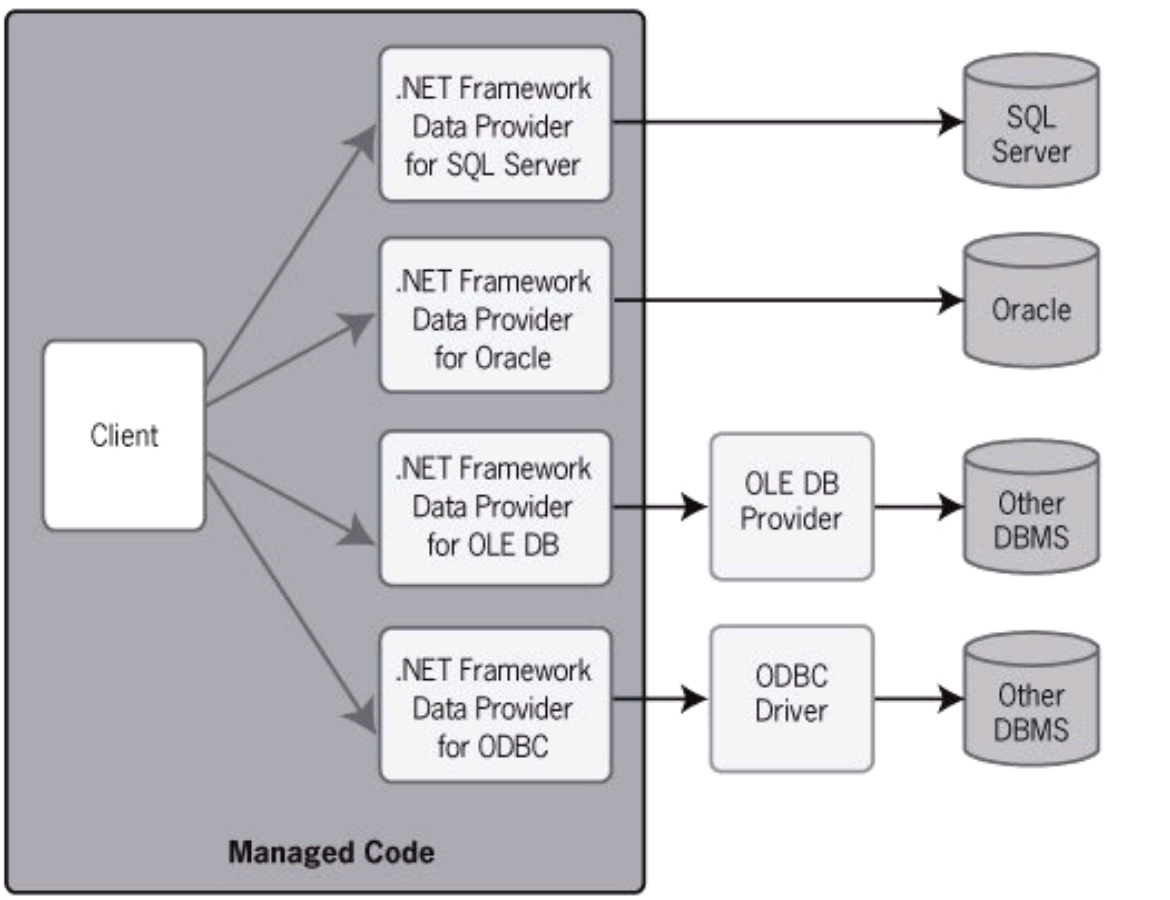
- 概述
- .NET数据供应器
- 直接访问数据
- 以DataSet访问数据
- LINQ to ADO.NET 概述



ADO.NET - 数据供应器

- ADO.NET访问数据，依赖于.NET供应器。 ADO.NET提供了多种数据供应器来访问数据源。
  - SQL Server.NET ：位于System.Data.SqlClient命名空间，用于访问Microsoft® SQL Server™ 7.0 或更高版本。
  - OLE DB.NET ：位于System.Data.OleDb命名空间，用于访问所有类型的数据的开放式标准，这些数据既包括关系数据又包括非关系数据。（ OLE DB：对象链接和嵌入数据库）
  - ODBC.NET ：位于System.Data.Odbc命名空间，用于访问ODBC 公开的数据源 （ ODBC：开放数据库互连标准 ）
  - Oracle.NET ：位于System.Data.OracleClient命名空间，适用于 Oracle 数据源。Oracle .NET Framework 数据提供程序支持 Oracle 客户端软件 8.1.7 版和更高版本。

ADO.NET - 数据供应器



ADO.NET - 数据供应器

- 用.NET数据供应器访问数据源，性能较佳。
- 每一种.NET数据供应器都有一套同样的核心类
  - Connection：建立与特定数据源的连接。
  - Command：对数据源执行命令。
  - DataReader：从数据源中读取循序、只读的数据流。
  - DataAdapter：用数据源填充 DataSet 并解析更新。

ADO.NET - 数据供应器

- 除上表列出的核心类之外，.NET Framework 数据提供程序还包含下表列出的类。
  - Transaction：使您能够在数据源的事务中登记命令。
  - CommandBuilder：帮助器对象将自动生成 DataAdapter 的命令属性或将从存储过程导出参数信息并填充 Command 对象的 Parameters 集合。
  - Parameter：定义命令和存储过程的输入、输出和返回值参数。
  - Exception：在数据源中遇到错误时返回。对于在客户端遇到的错误，.NET Framework 数据提供程序会引发 .NET Framework 异常。
  - Error：公开数据源返回的警告或错误中的信息。
  - ClientPermission：为 .NET Framework 数据提供程序代码访问安全属性而提供。

ADO.NET - 数据供应器

- 用户可通过DataReder或DataSet访问数据。

  ADO.NET提供访问数据的二个可选方案，二方案都使用Connections和commands与DBMS互动，但对“查询结果”的处理有重大差别。

- 选择 DataReader 或 DataSet

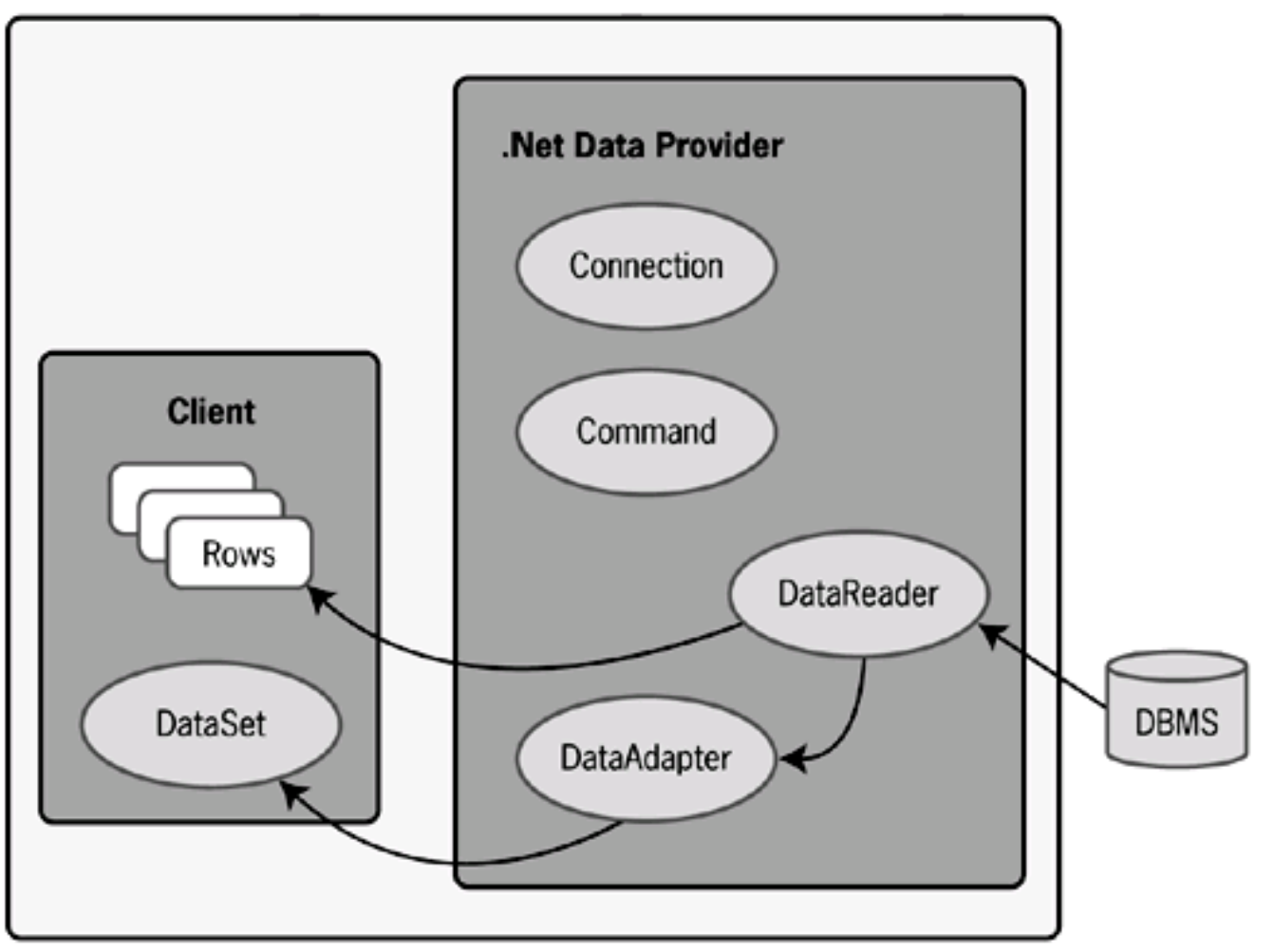
  决定应用程序应使用 DataReader还是应使用DataSet时，应考虑应用程序所需的功能类型。使用 DataSet 可执行以下操作：

ADO.NET - 数据供应器

- 在应用程序中将数据缓存在本地，以便对数据进行处理。 如只需读取查询结果，则 DataReader 是更好的选择。
- 在层间或从 XML Web 服务对数据进行远程处理。
- 与数据进行动态交互，如绑定到 Windows 窗体控件或组合并关联来自多个源的数据。
- 对数据执行大量的处理，而不需要与数据源保持打开的连接，从而将该连接释放给其他客户端使用。

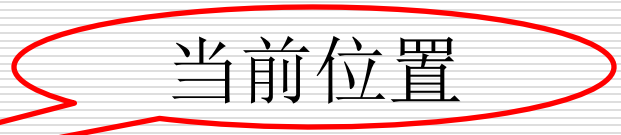
如不需要 DataSet 所提供的功能，则可以通过使用 DataReader 以只进、只读方式返回数据，从而提高应用程序的性能。

ADO.NET - 数据供应器



访问数据：ADO.NET

- 概述
- .NET数据供应器
- 直接访问数据
- 以DataSet访问数据
- LINQ to ADO.NET 概述



ADO.NET – 直接访问数据（ DataReader ）

- 用户可使用**Connection** 对象来连接到指定的数据源。 (对象的ConnectionString 属性设置连接数据库字符串，Open() 打开数据库连接。)
  - 若要连接到 Microsoft SQL Server 7.0 版或更高版本，请使用 SQL Server.NET数据提供器的 SqlConnection 对象。
  - 若要使用OLE DB 数据源或连接到 Microsoft SQL Server 6.*x* 版或较早版本，请使用 OLE DB.NET数据提供器的 OleDbConnection 对象。
  - 若要连接到 ODBC 数据源，请使用 ODBC .NET 数据提供器的 OdbcConnection 对象。
  - 若要连接到 Oracle 数据源，请使用 Oracle.NET 数据提供器的 OracleConnection 对象。

ADO.NET – 直接访问数据（ DataReader ）

- 用户依赖Command对象发送SQL操作(SQL操作放在属性CommandText中)， 与Connection类似，不同的数据源，有不同的Command类。
- Command对象提供了几个不同的选项来执行命令：
  - ExecuteReader：执行查询，将返回结果集生成一个DataReader。
  - ExecuteScalar：执行查询，并返回结果集中第一行的第一列或空引用（如果结果集为空）。 忽略额外的列或行。
  - ExecuteNonQuery：执行查询，并返回受影响的行数。
  - SqlCommand还提供ExecuteXmlReader：执行查询，将返回结果集生成一个 XmlReader。

ADO.NET – 直接访问数据（ DataReader ）

- Command对象还可用于执行存储过程。
  - 实例化Command对象时，用存储过程名代替SQL语句，另将Command对象的CommandType设置成StoredProcedure。
  - 若存储过程有参数，可用Command对象的Parameters.Add()方法添加Parameter参数。

ADO.NET – 直接访问数据（ DataReader ）

- 数据库访问完后，应显示关闭连接：
  Connection.Close()
- 用户可启动和结束事务
  - 启动事务：用 ObjTrans = Connection.BeginTransaction()
  - 结束事务：
    - 提交： ObjTrans.Commit()
    - 放弃(回滚)： ObjTrans.Rollback()

ADO.NET – 直接访问数据（ DataReader ）

- 可以使用 ADO.NET DataReader 从数据库中检索只读、只进的数据流。
- 举例：(DBExam)

```
Dim SqlConnT As SqlConnection = New
    SqlConnection("Data Source=localhost; Initial
  Catalog=ExamDB“; password=;user id=sa)
Dim SqlCmdT As SqlCommand =
    SqlConnT.CreateCommand()
SqlCmdT.CommandText = "SELECT * FROM
  ExamTable"
SqlConnT.Open()
```

ADO.NET – 直接访问数据

```
Dim myReader As SqlDataReader =
    SqlCmdT.ExecuteReader()
Dim intCt As Integer, intCols As Integer =
    myReader.FieldCount
Do While myReader.Read()
   For intCt = 0 To intCols – 1
      Console.Write("{0}" & vbTab, myReader(intCt))
   Next
   Console.WriteLine()
Loop
myReader.Close()
SqlConnT.Close()
```

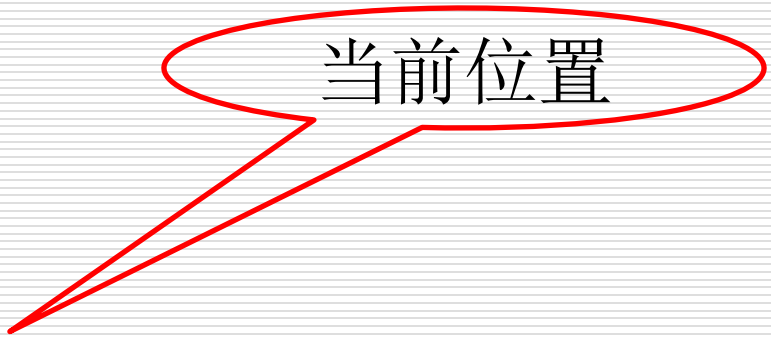
以XML形式读入(DBExam)

```
Dim SqlConnT As SqlConnection = New SqlConnection("data
    source=.;initial catalog=sq;password=;user id=sa")
Dim SqlCmdT As SqlCommand = SqlConnT.CreateCommand()
SqlCmdT.CommandText = "SELECT top 10 * FROM 基本信息
    FOR XML AUTO“
SqlConnT.Open()
Dim myReader As Xml.XmlReader=SqlCmdT.ExecuteXmlReader
Do While myReader.Read()
    Console.WriteLine(vbTab & "{0}" & vbTab & "{1}",
    myReader(0), myReader(1))
Loop
myReader.Close()
SqlConnT.Close()
```

注： AUTO：XML模式。可是RAW,AUTO,Explicit决定所得的XML形式

访问数据：ADO.NET

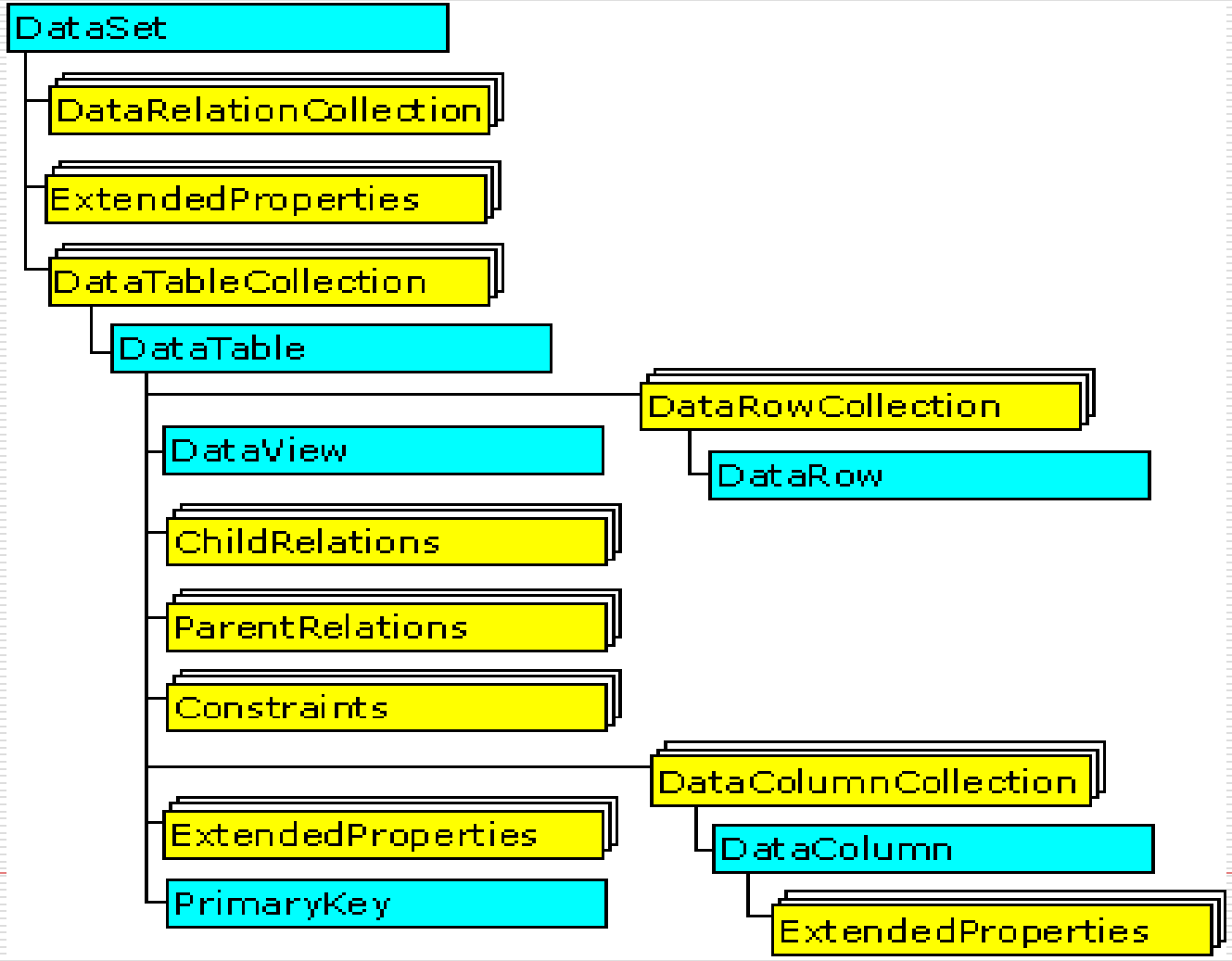
- 概述
- .NET数据供应器
- 直接访问数据
- 以DataSet访问数据
- LINQ to ADO.NET 概述



ADO.NET – 以DataSet访问数据

- DataSet 对象是支持 ADO.NET 的断开式、分布式数据方案的核心对象。是数据的内存驻留表示形式，无论数据源是什么，它都会提供一致的关系编程模型。它可以用于多个不同的数据源，用于 XML 数据，或用于管理应用程序本地数据。
- DataSet 的结构类似于关系数据库的结构；DataSet 表示包括相关表、行、列、约束和表间关系在内的整个数据集。(见下页图)
- 每一个DataSet都有一个结构纲目(Schema)

ADO.NET – 以DataSet访问数据



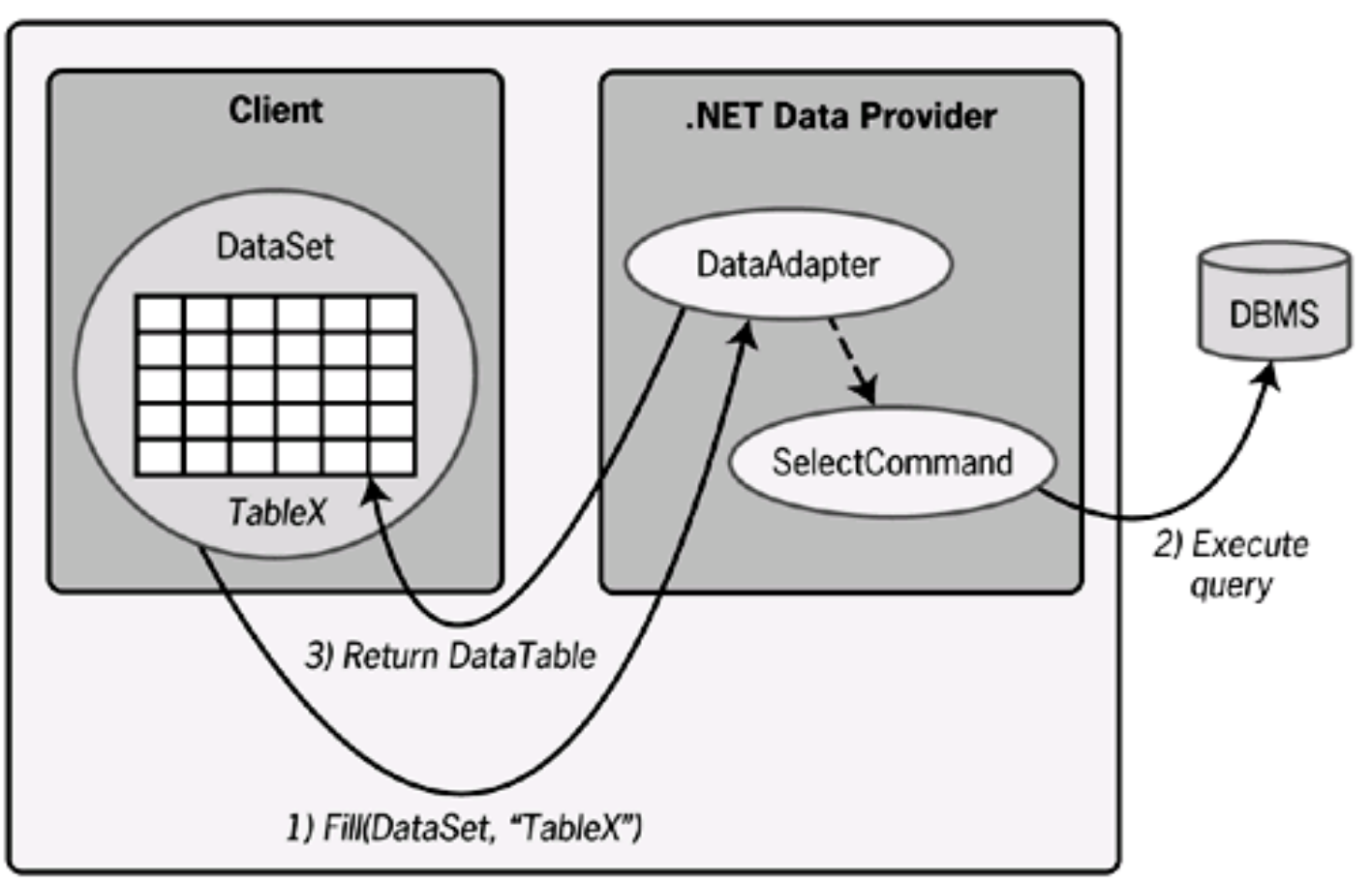
ADO.NET – 创建和使用DataSet

- 可用 Adapter 填充和更新 DataSet
  - 默认情况下， DataSet不包含任何实际数据。填充DataSet指的是将数据加载到组成DataSet的 DataTable 对象中。
  - 可以通过表适配器TableAdapte或数据适配器DataAdapter命令来填充数据表。对数据集填充数据时，将引发各种事件，并对约束进行检查，等等。
  - xxxAdapter表示一组数据命令和一个数据库连接，用作 DataSet 和数据源之间的桥接器以便检索和保存数据，通过映射 Fill 和 Update来提供这一桥接器。
  - TableAdapter是对DataAdapter功能改进了的组件。是具有内置连接对象并能够包含多个查询的 DataAdapter，即可以在 TableAdapter 上根据需要拥有任意多的查询，只要这些查询返回符合同一架构的数据即可。

ADO.NET – 创建和使用DataSet

- 可用 Adapter 填充和更新 DataSet(续)
  - DataAdapter中有一些重要属性，它们获取或设置一个 SQL 语句或存储过程，完成相应的功能。这些属性有：
    - SelectCommand：在数据源中选择记录。
    - InsertCommand：在数据源中插入新记录。
    - UpdateCommand：更新数据源中的记录。
    - DeleteCommand：从数据集删除记录。
  - 可以通过将 DataAdapter 与其关联的 Command 和 Connection 对象一起使用，访问数据源。

ADO.NET – 创建和使用DataSet



ADO.NET – 创建和使用DataSet

- 示例

  以下代码示例创建 DataAdapter 的一个实例，该实例使用与 Microsoft SQL Server Northwind 数据库的 Connection 并使用客户列表来填充 DataSet 中的 DataTable。向 DataAdapter 构造函数传递的 SQL 语句和 Connection 参数用于创建 DataAdapter 的 SelectCommand 属性。

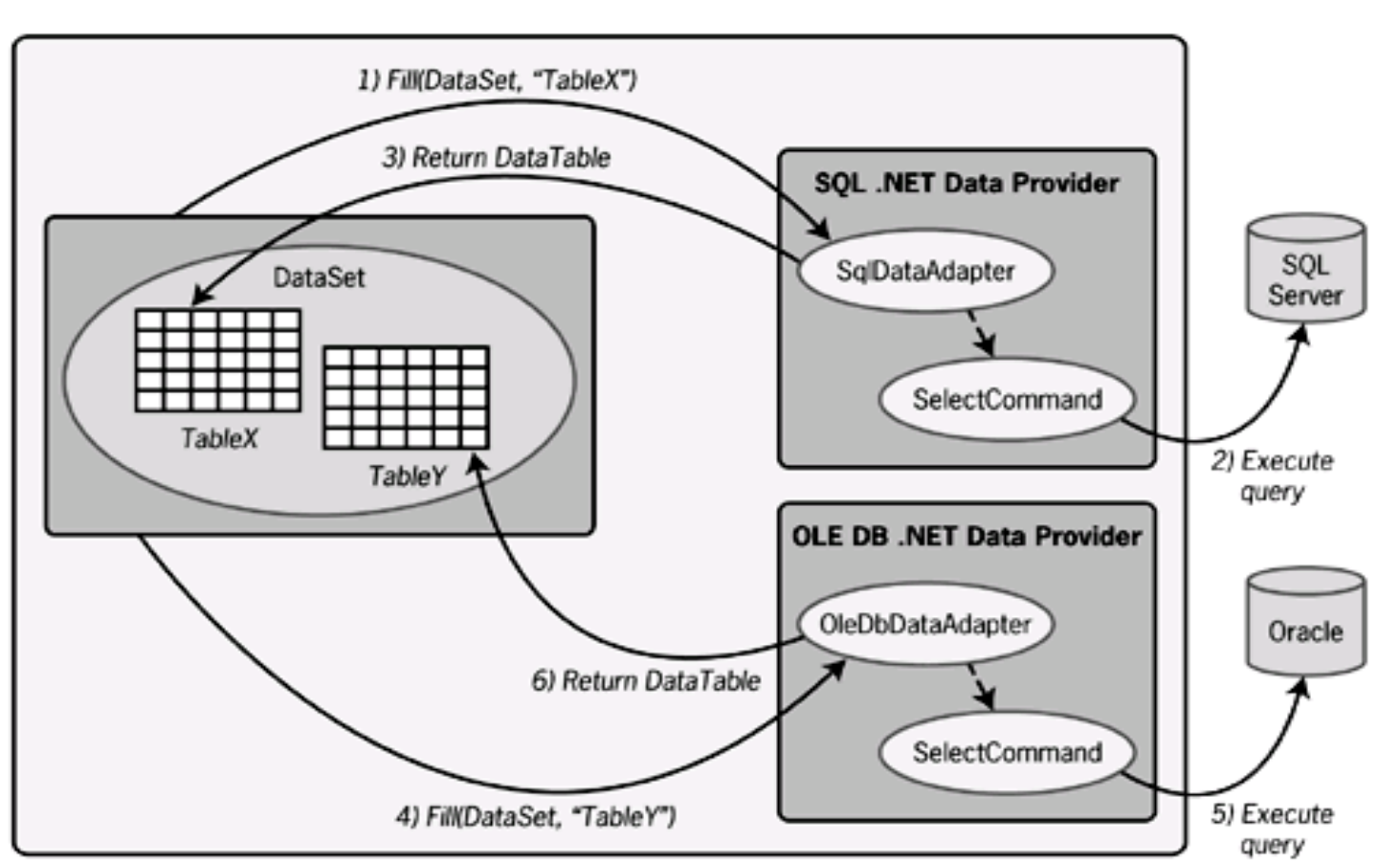
ADO.NET – 创建DataSet举例

```
SqlConnection nwindConn = new SqlConnection(
    "Data Source=localhost; Initial Catalog=northwind");
SqlCommand selectCMD = new SqlCommand(
    "SELECT CustomerID, CompanyName
    FROM Customers", nwindConn);
selectCMD.CommandTimeout = 30;
SqlDataAdapter custDA = new SqlDataAdapter();
custDA.SelectCommand = selectCMD;
nwindConn.Open();
DataSet custDS = new DataSet();
custDA.Fill(custDS, "Customers");
nwindConn.Close();
```

ADO.NET – 创建和使用DataSet

- 一个DataSet可以包含源于不同数据库的数个DataTables

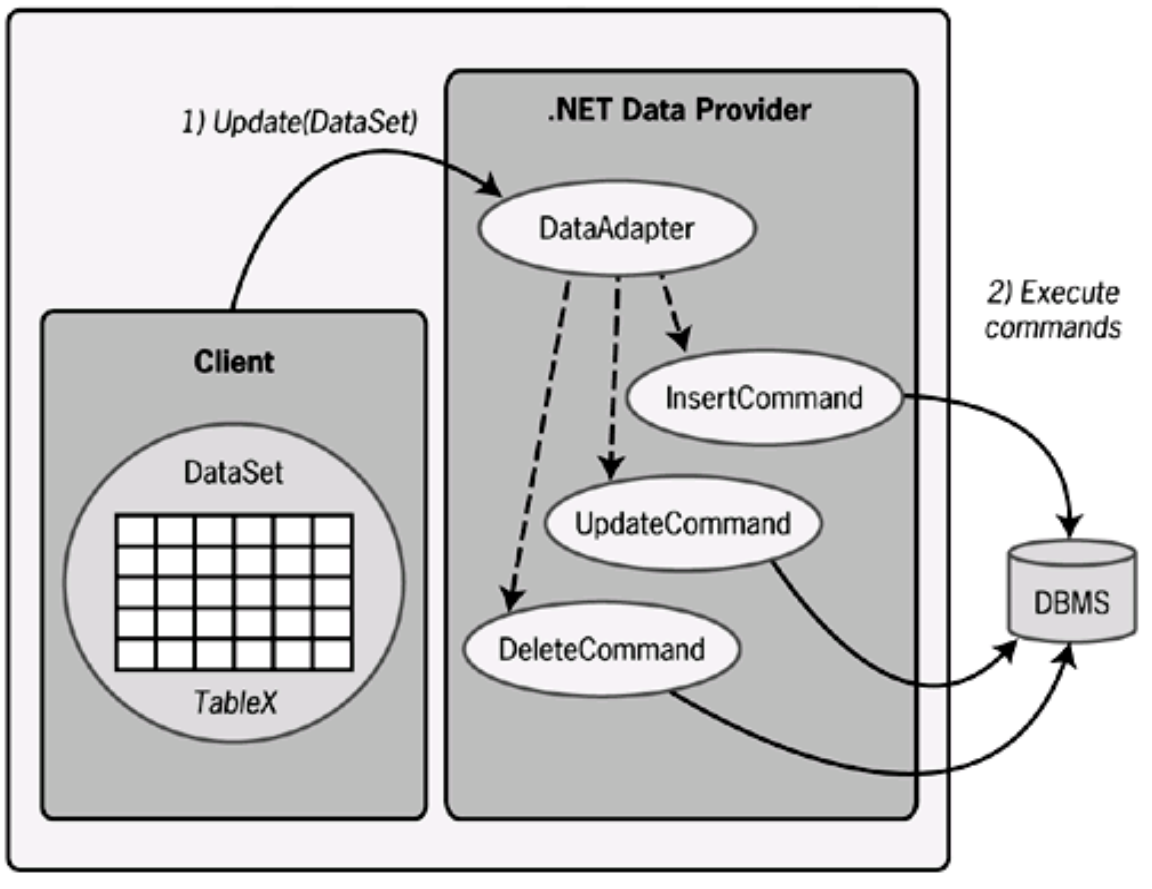
ADO.NET – 创建和使用DataSet



ADO.NET – 创建和使用DataSet

- DataSet可修改DBMS内的数据
  - 可用DataAdapter.Update()方法
  - 当应用程序调用 Update 方法时，DataAdapter 根据 DataSet 中配置的索引顺序为每一行检查 RowState 属性，并迭代执行所需的 INSERT、UPDATE 或 DELETE 语句。

ADO.NET – 创建和使用DataSet



ADO.NET – 创建和使用DataSet

- DataAdapter 不会自动生成 DataSet 更改 SQL Server 实例所需的 SQL 语句。但是，如果设置了DataAdapter 的 SelectCommand 属性，则可以创建一个 CommandBuilder 对象来自动生成用于单表更新的SQL 语句。然后，CommandBuilder 将生成其他任何未设置的SQL 语句。

  如：

ADO.NET – 创建和使用DataSet

```
Dim myConn As New OleDbConnection(连接串)
Dim myDataAdapter As New OleDbDataAdapter()
myDataAdapter.SelectCommand = New
        OleDbCommand(Sel查询, myConn)
Dim custCB As OleDbCommandBuilder = New
        OleDbCommandBuilder(myDataAdapter)
myConn.Open()
Dim custDS As DataSet = New DataSet
myDataAdapter.Fill(custDS)
' Code to modify data in DataSet here
' Without the OleDbCommandBuilder this line would fail.
myDataAdapter.Update(custDS)
myConn.Close()
CreateCmdsAndUpdate = custDS
```

（DBExample -- Update）

ADO.NET – 创建和使用DataSet

- DataSet自身不能对数据源锁定。

  数据源内容变动后，可能会带来一些问题。

- 程序可通过DataSet包含的DataTables来访问和修改DataSet的内容
  - DataRow：表示 DataTable 中的一行数据。
  - DataColumn：表示 DataTable 中列的架构。

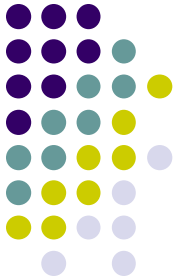
  如：

ADO.NET – 创建和使用DataSet

```
Dim SqlConnT As SqlConnection = New SqlConnection("data
    source=.;initial catalog=sq;password=;user id=sa")
Dim SqlCmdT As SqlCommand = SqlConnT.CreateCommand()
SqlCmdT.CommandText = "SELECT top 100 * FROM 基本信息"
SqlConnT.Open()
Dim dst As DataSet = New DataSet
Dim da As SqlDataAdapter = New SqlDataAdapter(SqlCmdT)
da.Fill(dst, "jbxx")
Dim intCt, intCt2 As Integer
For intCt2 = 0 To dst.Tables("jbxx").Columns.Count - 1
   Console.Write("{0}" & vbTab,
     dst.Tables("jbxx").Columns(intCt2).Caption)
Next
For intCt = 0 To dst.Tables("jbxx").Rows.Count - 1
   Dim dr As DataRow = dst.Tables("jbxx").Rows(intCt)
   For intCt2 = 0 To dst.Tables("jbxx").Columns.Count - 1
     Console.Write("{0}" & vbTab, dr(intCt2))
   Next
   Console.WriteLine()
 Next
```

ADO.NET – 创建和使用DataSet

- DataTable的Select方法可按照指定顺序获取与筛选条件相匹配的所有DataRow对象的数组。 Select()中有三个参数：
  - 要用来筛选行的条件。
  - 一个字符串，它指定列和排序方向。
  - DataViewRowState，描述 DataRow 中数据的版本，有：
    - **Added**：一个新行。
    - **CurrentRows**：包括未更改行、新行和已修改行的当前行。
    - **Deleted**：已删除的行。
    - **ModifiedCurrent**：当前数据，原始数据的修改版本。
    - **ModifiedOriginal**：原始数据（尽管它后来已被修改并以 **ModifiedCurrent** 形式存在）。
    - **None**：无。
    - **OriginalRows**：包括未更改行和已删除行的原始行。
    - **Unchanged**：未更改的行。



```
 Dim SqlConnT As SqlConnection = New SqlConnection("data
     source=.;initial catalog=sq;password=;user id=sa")
Dim SqlCmdT As SqlCommand = SqlConnT.CreateCommand()
SqlCmdT.CommandText = "SELECT top 100 * FROM 基本信息 “
SqlConnT.Open()
Dim dst As DataSet = New DataSet
Dim da As SqlDataAdapter = New SqlDataAdapter(SqlCmdT)
da.Fill(dst, "jbxx")
Dim strTj = "空船吨位=135"
Dim myRows() As DataRow = dst.Tables("jbxx").Select(strTj, "ID",
     DataViewRowState.CurrentRows)
Dim intCt, intCt2 As Integer
Console.WriteLine()
Dim dr As DataRow
For Each dr In myRows
   For intCt2 = 0 To 10
      Console.Write("{0}" & vbTab, dr(intCt2))
   Next
   Console.WriteLine()
Next
```

修改数据库存盘

```
Dim SqlConnT As SqlConnection = New SqlConnection("data source=.;initial catalog=sq;password=;user id=sa")
Dim SqlCmdT As SqlCommand = SqlConnT.CreateCommand()
SqlCmdT.CommandText = "SELECT top 10 * FROM 基本信息"
SqlConnT.Open()
Dim dst As DataSet = New DataSet
Dim da As SqlDataAdapter = New SqlDataAdapter(SqlCmdT)
Dim custCB As SqlCommandBuilder = New SqlCommandBuilder(da)
da.Fill(dst, "jbxx")
dst.Tables("jbxx").Rows(0)(1) = "test"
da.Update(dst, "jbxx")
```

ADO.NET – 创建和使用DataSet

- DataSet可用于处理以XML格式定义的数据
- 可以将一个XML文档直接读入一个DataSet中
  用DataSet的ReadXml
- DataSet schema可以自动从一个XML Document构建而来
- DataSet也可显示读取一个XSD schema
- DataSet中的内容可被序列化为一个XML文档
- 一个DataSet可以和一个XmlDataDocument对象保持同步

例：将一个XML文档直接读入一个DataSet中并输出(DBExam—ReadXml)

```
 Dim dst As DataSet = New DataSet
dst.ReadXml("Books.xml", XmlReadMode.Auto)
Console.WriteLine("表名:" & dst.Tables(0).TableName)    ‘输出表名
 ‘GetXml方法返回存储在 DataSet 中的数据的 XML 表示形式（字符串）。
Console.WriteLine(dst.GetXml())
Dim intCt, intCt2 As Integer
’输出字段名
For intCt2 = 0 To dst.Tables(0).Columns.Count - 1
    Console.Write("{0}" & vbTab, dst.Tables(0).Columns(intCt2).ColumnName)
Next
‘输出内容
For intCt = 0 To dst.Tables(0).Rows.Count - 1
      Dim dr As DataRow = dst.Tables(0).Rows(intCt)
      For intCt2 = 0 To dst.Tables(0).Columns.Count - 1
        Console.Write("{0}" & vbTab, dr(intCt2))
      Next
      Console.WriteLine()
    Next
```

例： 将DataSet中的内容写入一个XML文档(DBExam—Write Xml)

```
Dim SqlConnT As SqlConnection = New SqlConnection("data
    source=.;initial catalog=sq;password=;user id=sa")
Dim SqlCmdT As SqlCommand = SqlConnT.CreateCommand()
SqlCmdT.CommandText = "SELECT top 10 * FROM 基本信息 “
SqlConnT.Open()
Dim dst As DataSet = New DataSet
Dim da As SqlDataAdapter = New SqlDataAdapter(SqlCmdT)
da.Fill(dst, "jbxx")

Dim fs As FileStream = File.Create("SzTest.xml")
Dim fs2 As FileStream = File.Create("SzTest2.xml")
‘从 DataSet 写 XML 数据，第二个参数可以选择写架构。
dst.WriteXml(fs)
dst.WriteXmlSchema(fs2) ‘写 XML 架构形式的 DataSet 结构
fs.Close()
```

LINQ to ADO.NET 概述

- LINQ to ADO.NET 包括两种独立的技术：LINQ to DataSet 和 LINQ to SQL。

  使用 LINQ to DataSet 可以对 DataSet 执行丰富而优化的查询;

  使用 LINQ to SQL 可以直接查询 SQL Server 数据库架构。

DataGrid 类

- 在可滚动的网格中显示 ADO .NET 数据。

  DataGrid 显示指向子表的类似 Web 的链接。您可以单击链接定位到子表。在显示子表时，标题中显示后退按钮，可以单击它向后定位到父表。父行的数据显示在标题下面、列标头的上面。可以通过单击后退按钮右边的按钮来隐藏父行信息。
- 若要在运行时在**DataGrid** 中显示表，可将 DataSource 和 DataMember 属性设置为有效的数据源。
- 若要确定所选择的单元格，使用 CurrentCell 属性。
- 可使用 Item 属性访问任意单元格

DataGrid 类

- 若要管理控件的外观，有多个可用于设置颜色和标题属性的属性，包括 CaptionForeColor、CaptionBackColor、CaptionFont 等。
- 可进一步修改所显示网格的外观，方法是创建 DataGridTableStyle 对象，并将它们添加到 GridTableStylesCollection 中，可通过 TableStyles 属性访问后者。例如，如果 **DataSource** 设置为包含三个 **DataTable** 对象的 **DataSet**，则可以向该集合添加三个 **DataGridTableStyle** 对象，每个表一个。若要将每个 **DataGridTableStyle** 对象与一个 **DataTable** 同步，请将 **DataGridTableStyle** 的 MappingName 设置为 **DataTable** 的 TableName。

DataView 类

- 表示用于排序、筛选、搜索、编辑和导航的 DataTable 的可绑定数据的自定义视图。
  - 创建 DataView ：有两种：
    - 使用 DataView 构造函数。如：

```
DataView dvT = new DataView(dsT.Tables[“jbxx"]);
```

    - 创建对 DataTable 的 DefaultView 属性的引用。如：

```
DataView dvT = dsT.Tables["jbxx"].DefaultView;
```

DataView 类

- 使用 DataView 对数据排序和筛选
  - 使用 Sort 属性，可指定单个或多个列排序顺序并包含 ASC（升序）和 DESC（降序）参数。
  - 使用 RowFilter 属性，可根据行的列值来指定行的子集。
  - 使用 RowStateFilter 属性，可指定要查看的行版本。
    - DataViewRowState 选项

      CurrentRows：所有行版本。（默认选项）

      Added：所有增加行的当前行版本。

      Deleted：所有删除行的原行版本。

      ModifiedCurrent：所有修改行的当前行版本。

      ModifiedOriginal：所有修改行的原始行版本。

      None：没有行。

      OriginalRows：所有 Unchanged、Modified 和 Deleted 行的 Original 行版本。

      Unchanged：所有 Unchanged 行的 Current 行版本。

```
DataView catView = new DataView(catDS.Tables[“jbxx"]);
Console.WriteLine("Current Values:");
WriteView(catView);

Console.WriteLine("Original Values:");
catView.RowStateFilter = DataViewRowState.ModifiedOriginal;
WriteView(catView);

public static void WriteView(DataView myView)
{
  foreach (DataRowView myDRV in myView)
  {
    for (int i = 0; i < myView.Table.Columns.Count; i++)
      Console.Write(myDRV[i] + "\t");
    Console.WriteLine();
  }
}
```

DataView 类

- 搜索 DataView
  - 使用 DataView 的 Find 和 FindRows 方法，可按照行的排序关键字值来对行进行搜索。
    - 搜索值是否区分大小写取决于基础DataTable的CaseSensitive属性。
    - Find 方法返回一个整数，表示匹配搜索条件的 DataRowView 的第一个索引。如果未找到匹配项，Find 返回 -1。 如：

```
dvT.RowStateFilter =DataViewRowState.CurrentRows;
dvT.Sort="船舶主人";
int p=dvT.Find("张文付");
```

    - FindRows 返回引用 DataView 中所有匹配行的 DataRowView 数组。如果未找到匹配项，DataRowView 数组将为空。

```
dvT.Sort="船舶主人";
DataRowView[] foundRows = dvT.FindRows("150");
if (foundRows.Length == 0)
        Console.WriteLine("No match found.");
else
        foreach (DataRowView myDRV in foundRows)
        Console.WriteLine("{0}, {1}", myDRV["ID"].ToString(),
                myDRV["空船吨位"].ToString());
```

DataView 类

- 使用 DataView 修改数据

  可使用 DataView 在基础表中添加、删除或修改数据行。通过设置 DataView 的三个属性，可以控制使用 DataView 修改基础表数据的能力。这些属性为：（默认为 true ）

  - AllowNew：如果为 true，则可使用 DataView 的 AddNew 方法。
  - AllowEdit：如果为 true，可以通过 DataRowView 来修改 DataRow 的内容。可使用 DataRowView.EndEdit 确认对基础行的更改，或使用 DataRowView.CancelEdit 拒绝更改。
  - AllowDelete：如果为 true，则可以使用 DataView 或 DataRowView 对象的 Delete 方法删除 DataView 中的行，这些行也将从基础 DataTable 中删除。

ASP.NET 2.0中数据绑定的变化