

Bài 10: ADO.NET

Lương Trần Hy Hiến

FIT, HCMUP

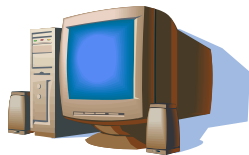
Lập trình Windows Form với C#

Nội dung

- **ADO.NET**
- **Sơ lược lịch sử phát triển**
- **Kiến trúc ADO.NET**
- **.NET Data Provider**
- **DataSet**
- **Hỏi & Đáp**

Giới thiệu ADO.NET

- **ActiveX Data Object .NET (ADO.NET)**
 - Công nghệ của MS
 - Phát triển từ nền tảng ADO
 - Cung cấp các **lớp đối tượng** và **hàm thư viện** phục vụ cho việc kết nối và xử lý dữ liệu



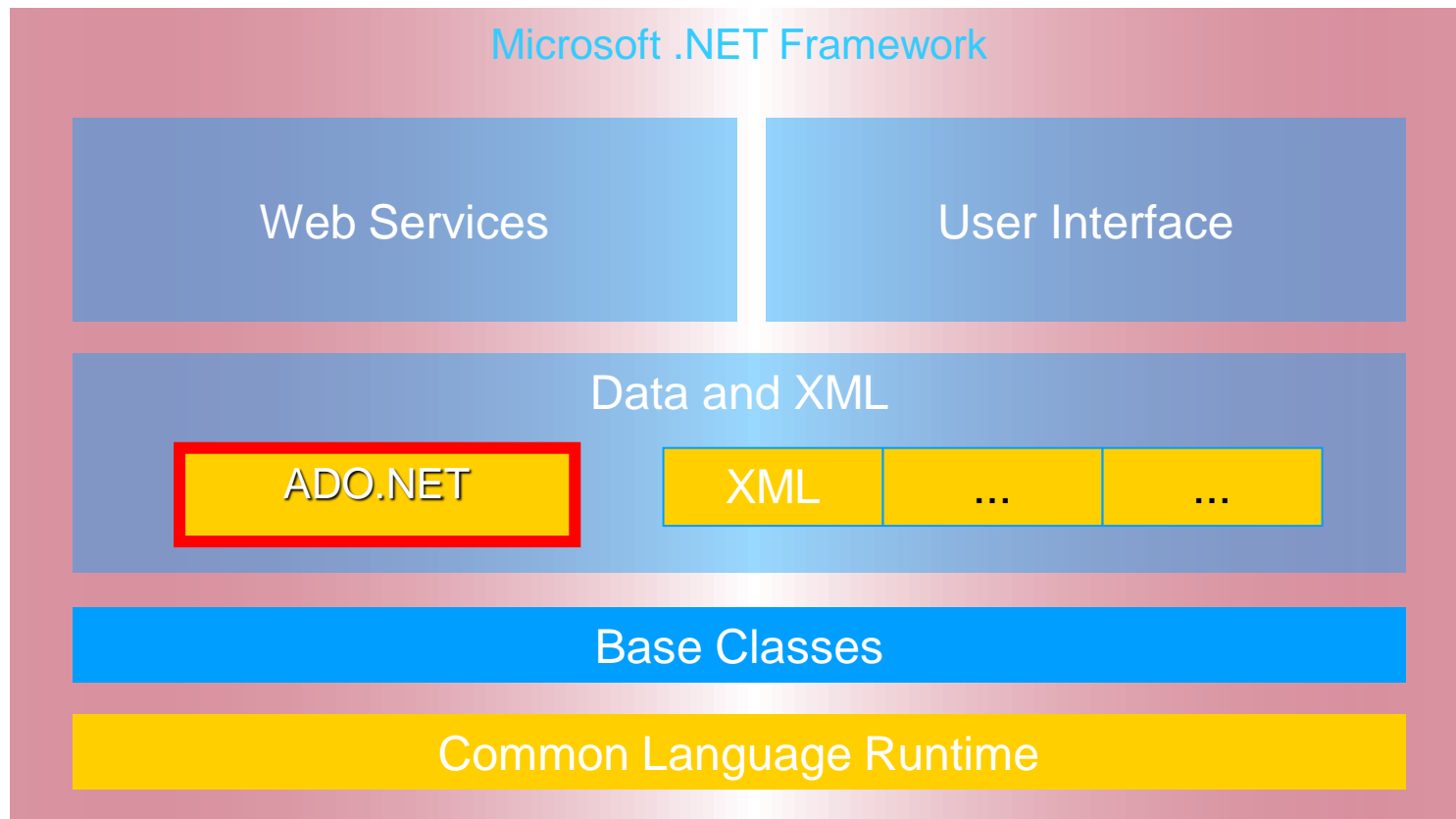
.NET Application

ADO.NET



Giới thiệu ADO.NET

■ Mô hình .NET Framework



Nội dung

- **ADO.NET**

- **Sơ lược lịch sử phát triển**

- **Kiến trúc ADO.NET**

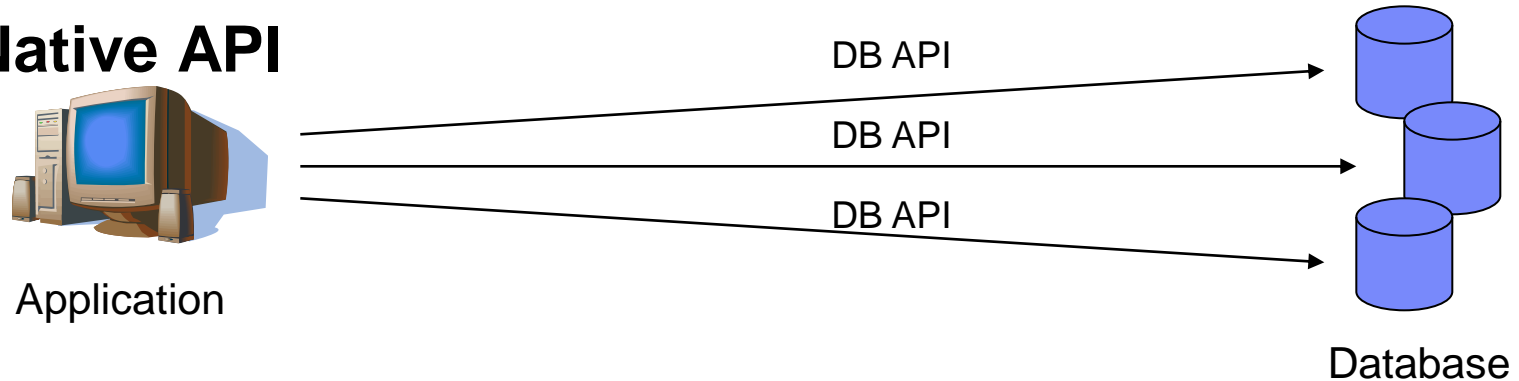
- **.NET Data Provider**

- **DataSet**

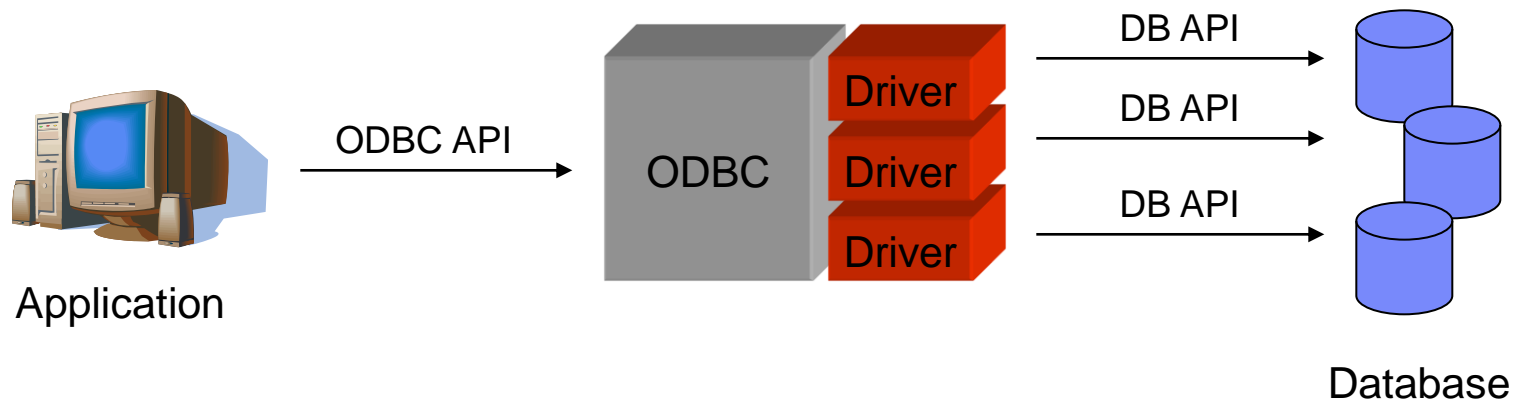
- **Hỏi & Đáp**

Sơ lược lịch sử phát triển

■ Native API

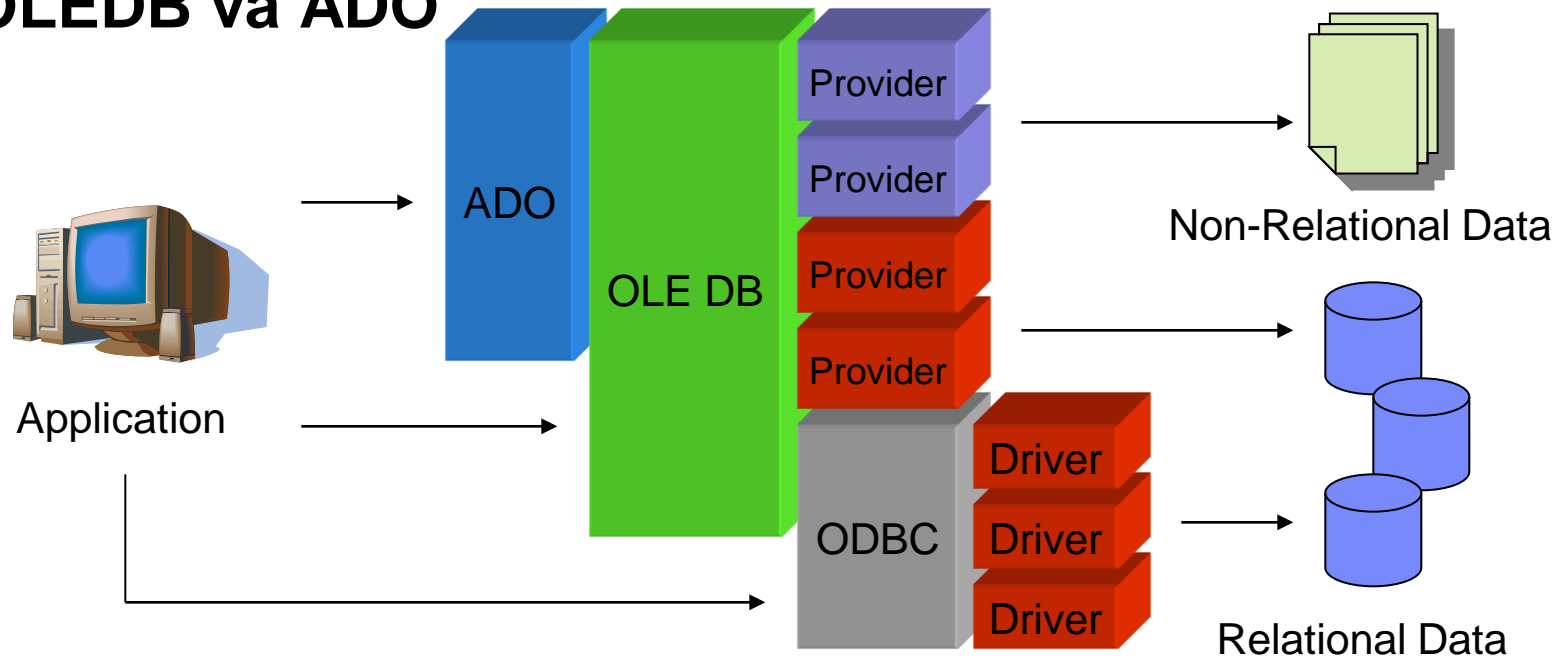


■ Open DataBase Connectivity



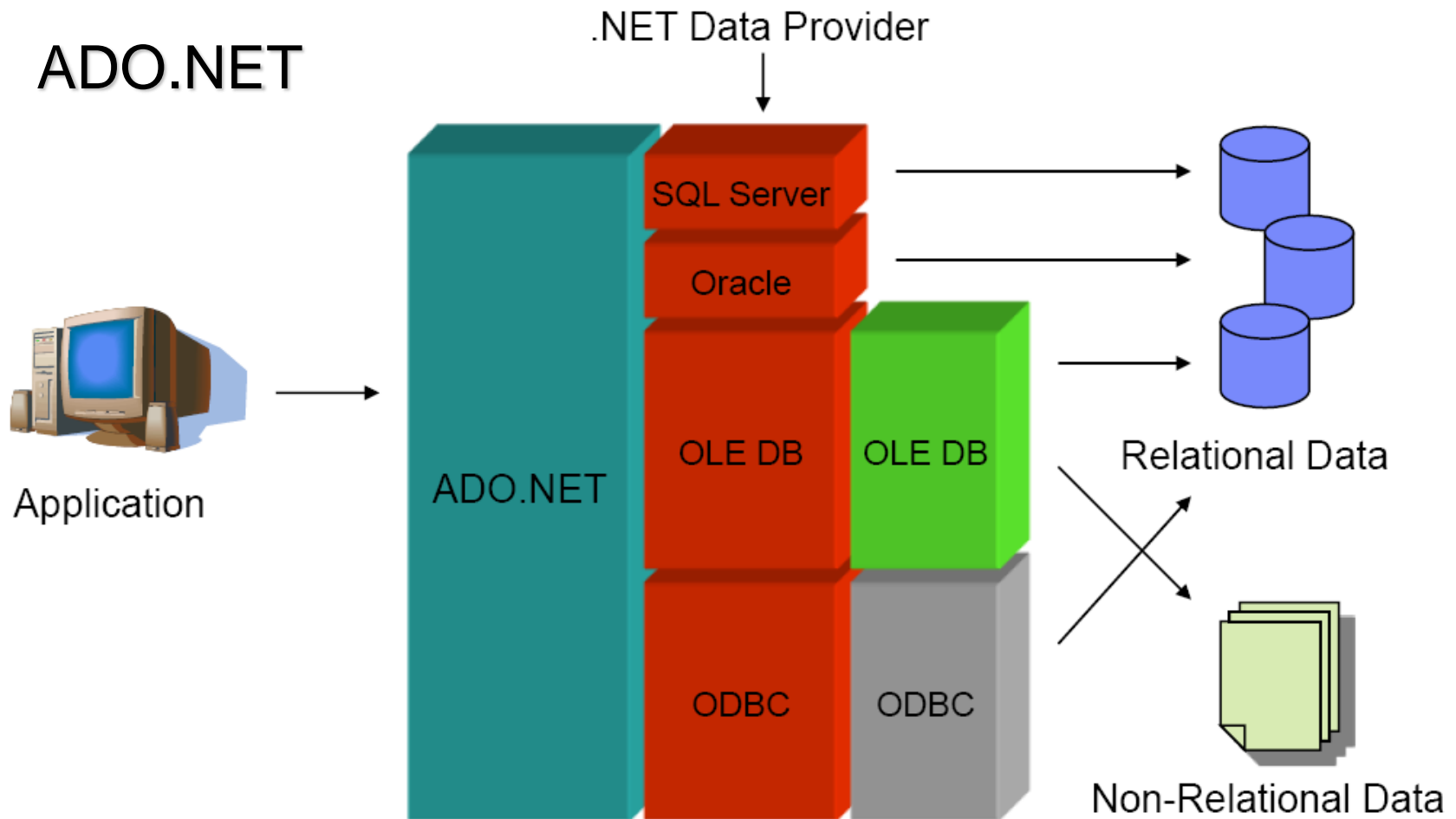
Sơ lược lịch sử phát triển (cont)

OLEDB và ADO



OLE: Object Linking and Embedding

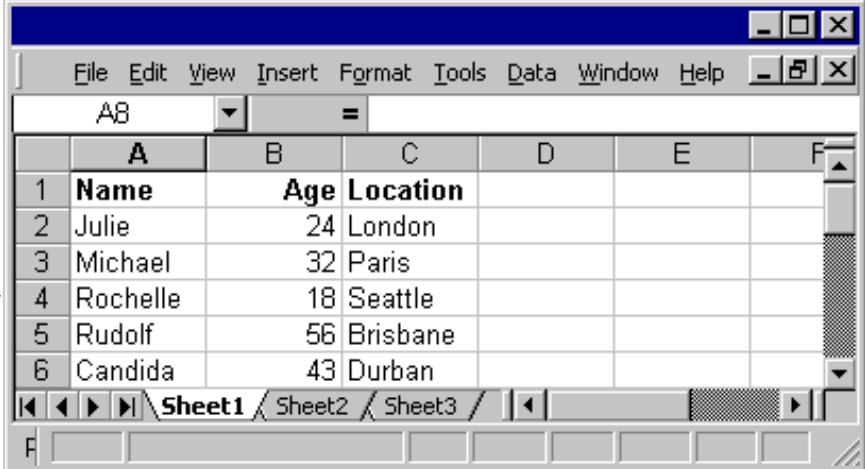
Sơ lược lịch sử phát triển (cont)



Introduction

Data

Stored
into



	A	B	C	D	E	F
1	Name	Age	Location			
2	Julie	24	London			
3	Michael	32	Paris			
4	Rochelle	18	Seattle			
5	Rudolf	56	Brisbane			
6	Candida	43	Durban			

Database (Oracle, SQL Server)

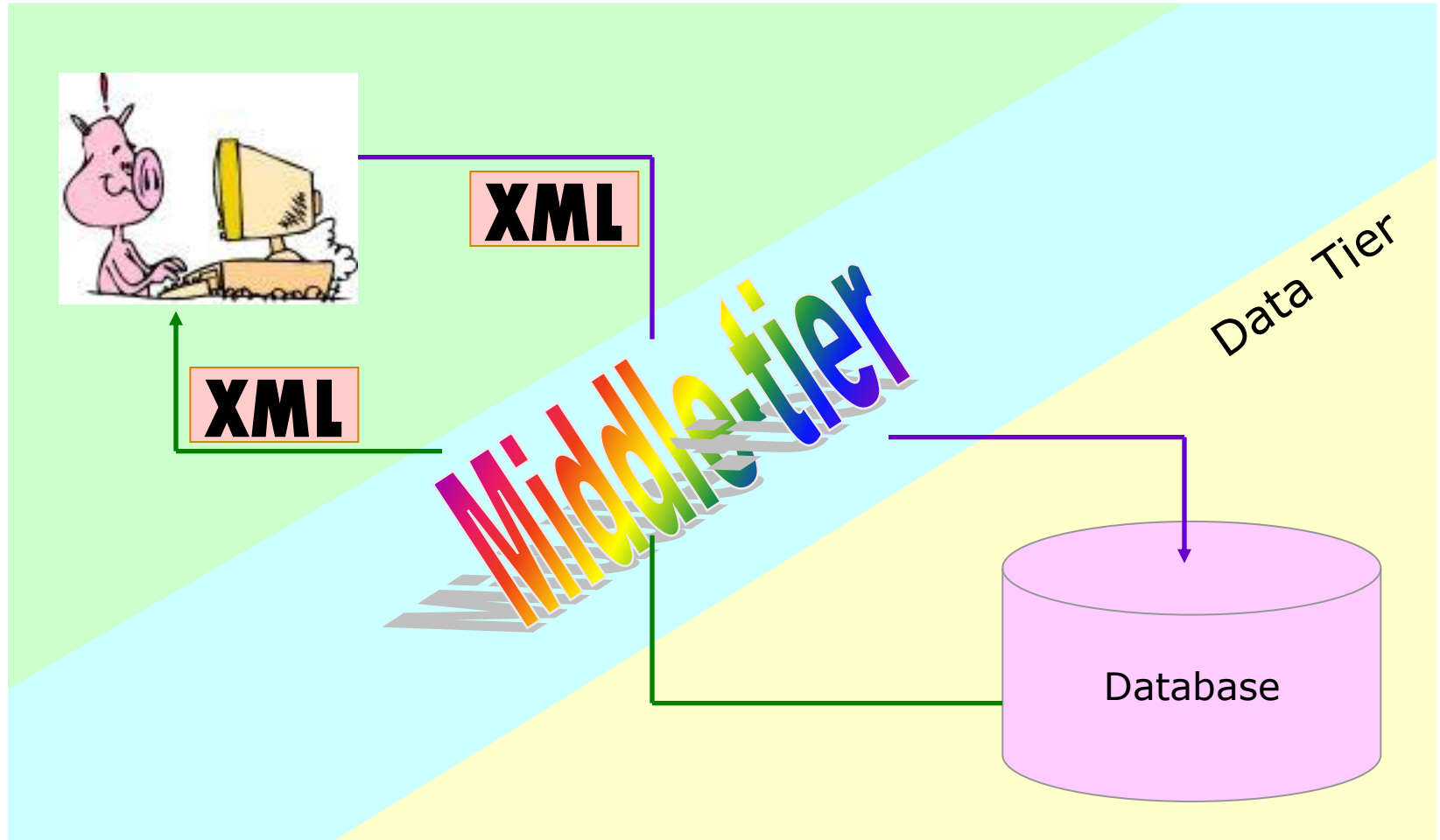


Client

ADO.NET

Data access technology

ADO.NET architecture



ADO.NET (tt)

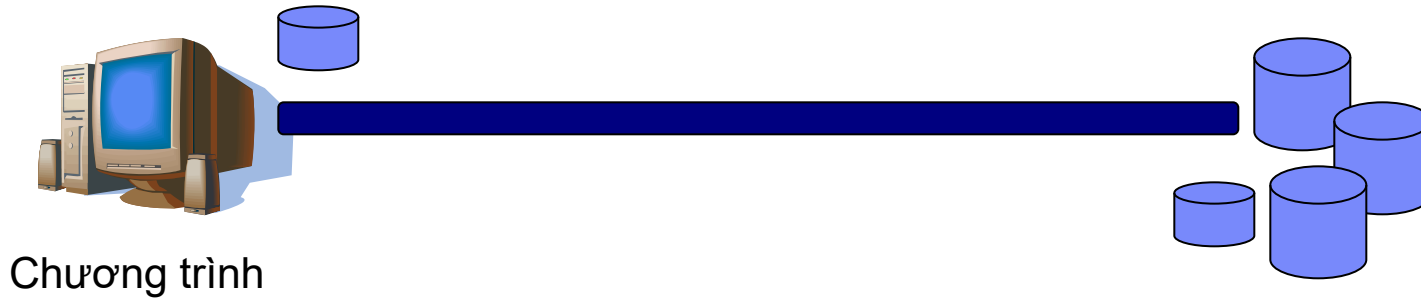
- **Hỗ trợ bởi .Net Platform**
- **Sử dụng công nghệ XML để chuyển đổi dữ liệu.**
- **Tương tác với tất cả các loại cơ sở dữ liệu.**
- **Khả năng thực thi nhanh.**
- **Sử dụng cho các loại ứng dụng client-server.**

Nội dung

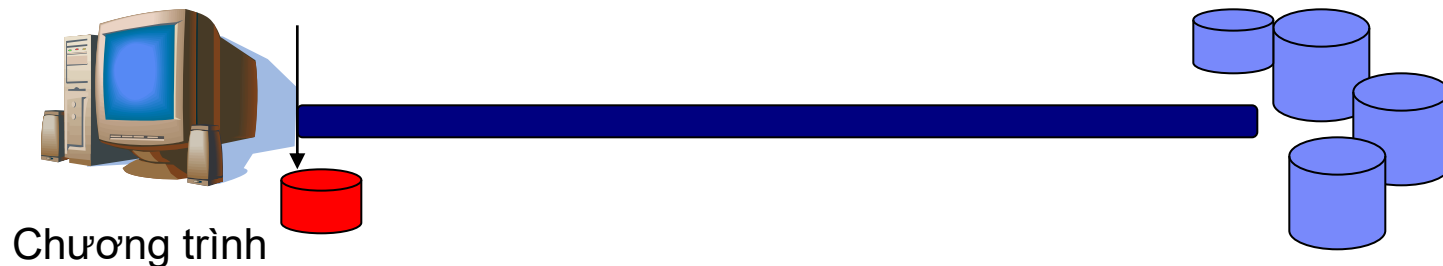
- **ADO.NET**
- **Sơ lược lịch sử phát triển**
- **Kiến trúc ADO.NET**
- **.NET Data Provider**
- **DataSet**
- **Hỏi & Đáp**

Kiến trúc

■ Connected Model



■ Disconnected Model



ADO.NET

- **ADO.NET là một phần của .NET Framework**
 - Thư viện lớp có chức năng **thao tác dữ liệu** trong ngôn ngữ MS.NET
- **ADO.NET là dạng “*Disconnected*”**
 - Cho phép lấy cả một cấu trúc phức tạp của DL từ CSDL, sau đó ngắt kết nối rồi mới thực hiện thao tác xử lý!
 - Trước đây ADO luôn phải duy trì kết nối trong quá trình làm việc.

Môi trường “connected”

- **Mỗi user có một kết nối cố định tới data source**
- **Ưu điểm**
 - Môi trường được bảo vệ tốt
 - Kiểm soát được sự đồng bộ
 - Dữ liệu luôn được mới
- **Nhược**
 - Phải có một kết nối mạng cố định
 - Scalability

Môi trường “disconnected”

- Một tập con của dữ liệu trung tâm được sao chép và bổ sung độc lập, sau đó sẽ được merge lại vào dữ liệu trung tâm.
- **Ưu điểm**
 - Có thể làm việc bất cứ lúc nào, cũng như có thể kết nối bất kỳ vào Data Source
 - Cho phép user khác có thể kết nối
 - Nâng cao hiệu suất thực hiện của ứng dụng
- **Khuyết**
 - Dữ liệu không được cập nhật một cách nhanh nhất
 - Sự tranh chấp có thể xuất hiện và phải giải quyết

ADO.NET

- **ADO.NET mạnh mẽ**

- Kế thừa các ưu điểm của ADO
- Kết hợp với ý tưởng thiết kế hoàn toàn mới

- **Đặc điểm nổi bật**

- Thiết kế hoàn toàn dựa vào XML
 - Chuẩn giao tiếp dữ liệu tốt nhất trên môi trường Internet hiện nay
- Thiết kế hoàn toàn hướng đối tượng
 - Đặc trưng của thư viện .NET Framework

ADO.NET vs. ADO

Đặc Điểm	ADO	ADO.NET
DL xử lý được đưa vào bộ nhớ dưới dạng	Recordset : tương đương 1 bảng dữ liệu trong database	Dataset : tương đương 1 database
Duyệt dữ liệu	Recordset chỉ cho phép duyệt tuần tự, từng dòng một.	Dataset : duyệt “tự do, ngẫu nhiên”, truy cập thẳng tới bảng, dòng ,cột mong muốn.
Dữ liệu ngắt kết nối	Recordset thiên về hướng kết nối, nên việc hỗ trợ ngắt kết nối không mạnh	Dataset hỗ trợ hoàn toàn ngắt kết nối
Trao đổi dữ liệu qua Internet	Khả năng trao đổi dữ liệu ADO qua Internet thường có nhiều hạn chế. Do dùng chuẩn COM	ADO.NET trao đổi dữ liệu qua Internet rất dễ dàng vì ADO.NET được thiết kế theo chuẩn XML, là chuẩn dữ liệu chính được sử dụng để trao đổi trên Internet.

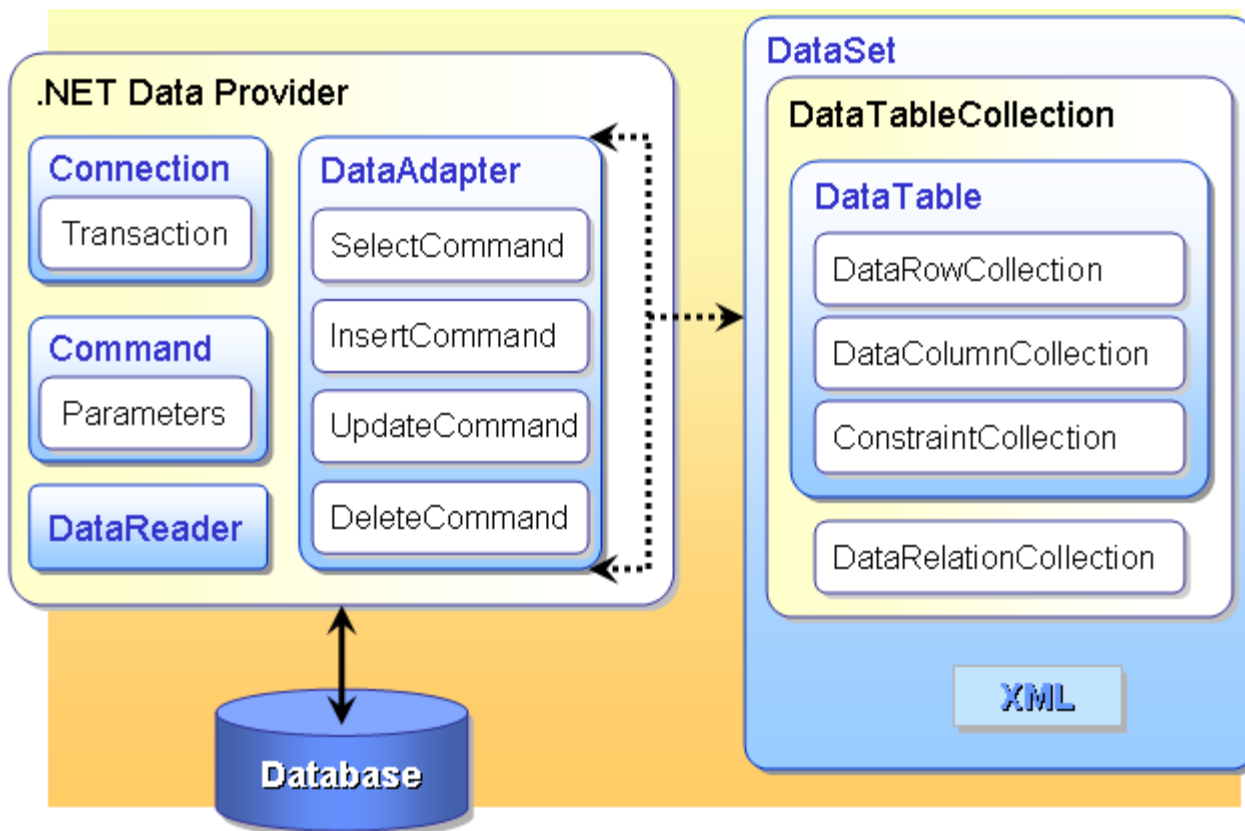
ADO.NET

- Kiến trúc của ADO.NET gồm 2 phần chính
- **Phần kết nối**: sử dụng khi kết nối CSDL và thao tác dữ liệu, phải thực hiện kết nối khi thao tác
 - **Connection**: quản lý việc đóng mở DB
 - **???Connection**: `SqlConnection`, `OleDbConnection`
 - **Command**: lệnh truy vấn, tương tác dữ liệu khi đang lập kết nối
 - **???Command**: `SqlCommand`, `OleDbCommand`
 - **DataReader**: đọc dữ liệu, chỉ xử lý 1 dòng dữ liệu tại một thời điểm
 - **???DataReader**: `SqlDataReader`, `OleDbDataReader`
 - **DataAdapter**: cầu nối giữa DataBase và DataSet

ADO.NET

- **Phần ngắt kết nối: là DataSet**
 - DataSet không quan tâm đến Database thuộc kiểu gì, mà lấy dữ liệu từ DataAdapter để xử lý
 - DataSet xem như một Database trong bộ nhớ: bảng, quan hệ...
 - DataSet có các thành phần con như
 - DataTable
 - DataRow
 - DataColumn
 - DataRelation
 - Các đối tượng nhóm: DataTableCollection, DataRowCollection, DataColumnCollection

Mô hình đối tượng ADO.NET



Namespace

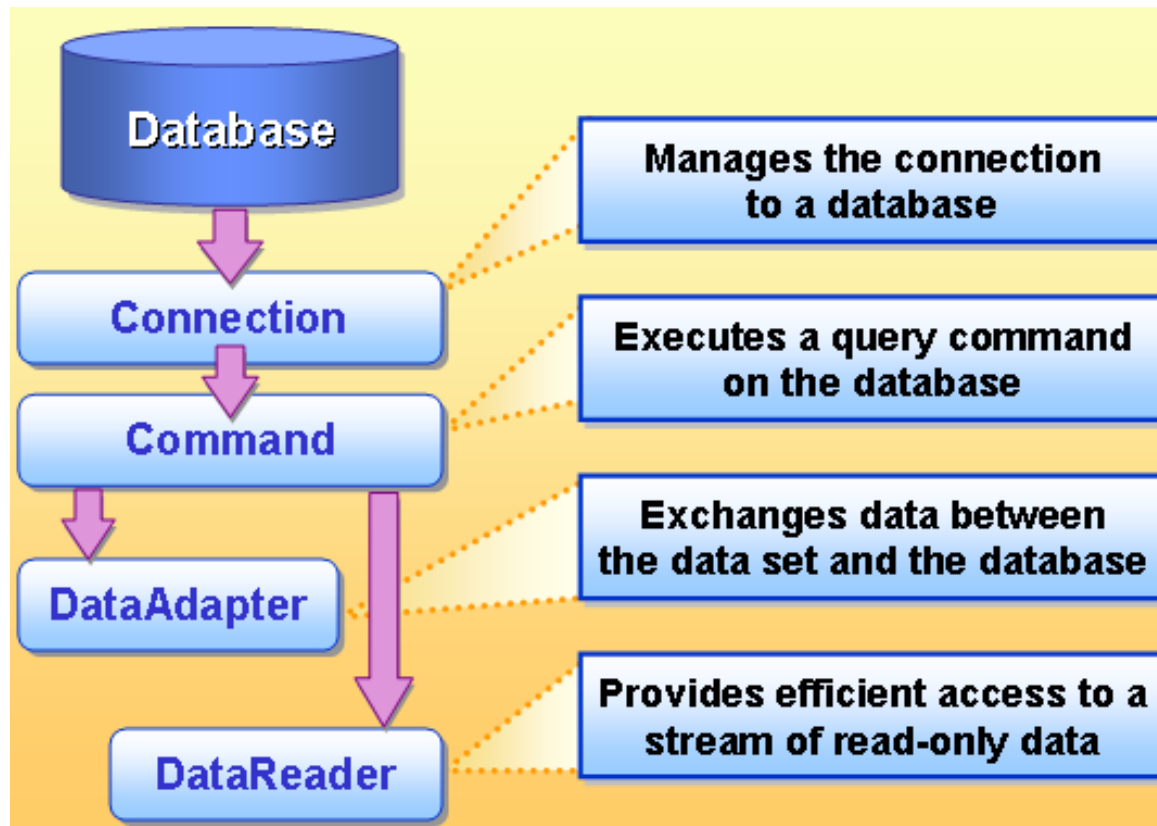
- **System.Data** — All generic data access classes
- **System.Data.Common** — Classes shared (or overridden) by individual data providers
- **System.Data.Odbc** — ODBC provider classes
- **System.Data.OleDb** — OLE DB provider classes
- **System.Data.ProviderBase** — New base classes and connection factory classes
- **System.Data.Oracle** — Oracle provider classes
- **System.Data.Sql** — New generic interfaces and classes for SQL Server data access
- **System.Data.SqlClient** — SQL Server provider classes
- **System.Data.SqlTypes** — SQL Server data types

Các lớp thư viện ADO.NET

- **System.Data.OleDb**: Access, SQL Server, Oracle
- **System.Data.SqlClient**: SQL Server
- **System.Data.OracleClient**: Oracle
- **Đặc điểm:**
 - Cả ba thư viện trên về giao tiếp lập trình là giống nhau
 - Dùng thư viện SqlConnection truy xuất SQL Server nhanh hơn OleDb
 - Tương tự cho OracleClient

.NET Data Provider

■ Các thành phần .NET Data Provider



Connected Model

- **Kết nối vào CSDL**
- **Thực hiện lệnh**
 - Thêm/Xóa/Sửa dữ liệu
 - Đọc dữ liệu từ CSDL

Connected Model

■ Kết nối vào CSDL

— Các lớp phụ trách kết nối

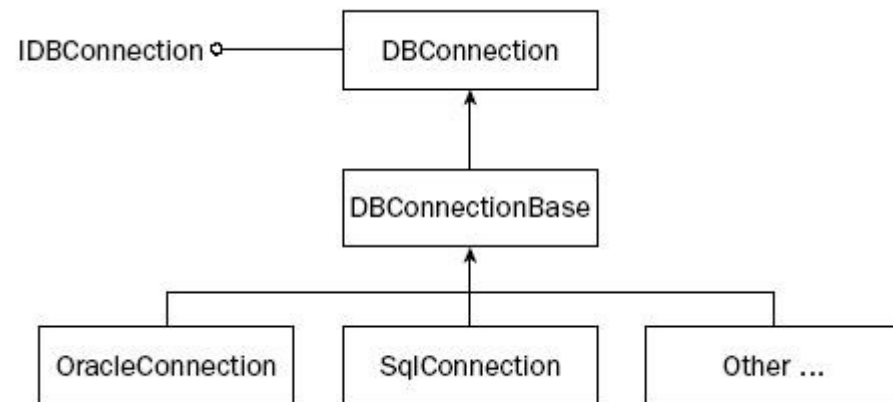
- ODBCConnection
- OleDbConnection
- SqlConnection
- OracleConnection

— Thuộc tính quan trọng

- **ConnectionString**: xác định nguồn dữ liệu cần kết nối

— Phương thức quan trọng

- **Open**: mở kết nối
- **Close**: đóng kết nối



.NET Data Provider - Connection



■ Các đối tượng Connection tuân thủ IDbConnection interface.

- **ConnectionString**: loại Data Source cần kết nối.
- **Open()**: thiết lập kết nối đến Data Source.
- **Close()**: ngắt kết nối đến Data Source.

.NET Data Provider – Connection (cont)

- **Kết nối vào CSDL SQL Server:**

```
SqlConnection cnn = new SqlConnection("server=  
HIENLTH-PC\\SQLEXPRESS; database = QLHS;  
user id=sa; password=sa");
```

```
cnn.Open();
```

```
// Thực hiện truy vấn dữ liệu
```

```
cnn.Close();
```

SQL2005ConnectionString

- Theo đặc quyền của SQL Server (có user, pass):

server= HIENLTH-PC\SQLEXPRESS; database = QLHS; user id=sa; password=sa

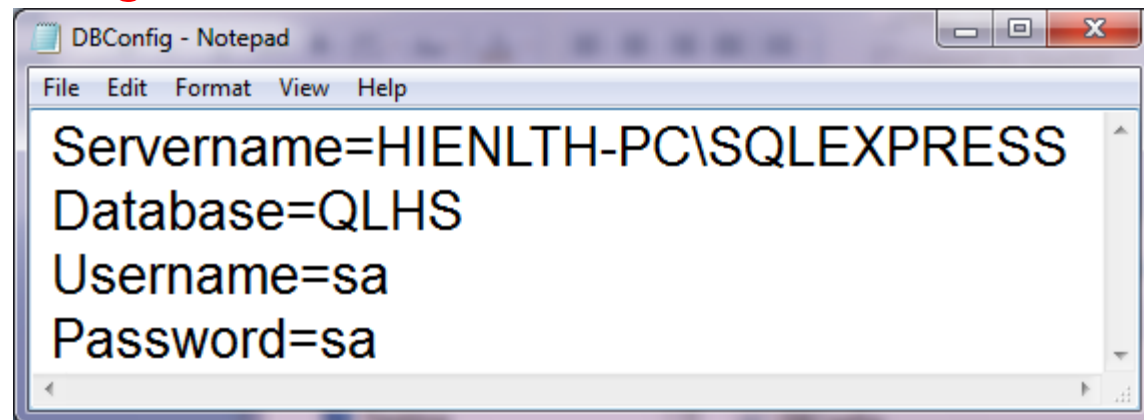
- Theo đặc quyền của hệ điều hành:

server= HIENLTH-PC\SQLEXPRESS; database = QLHS; Trusted_Connection = True;

server= HIENLTH-PC\SQLEXPRESS; database = QLHS; Integrated Security = True;

Tập tin lưu trữ kết nối

- Cho phép người dùng có thể cấu hình các giá trị thuộc tính cho chuỗi kết nối CSDL.
- Thường có dạng **.ini*, **.txt*. Từ .NET 1.0 trở lên có thể sử dụng tập tin *App.config* (định dạng XML) chứa khai báo các tham số cùng giá trị và các chỉ thị khác.
- Ví dụ: Tập tin *DBConfig.ini*



Đọc thông tin tập tin lưu trữ kết nối

- Loại *.ini, *.txt sử dụng StreamReader (using System.IO)

```
StreamReader docfile = new  
StreamReader(@"DBConfig.ini");  
servername = Tach(docfile.ReadLine());  
databasename = Tach(docfile.ReadLine());  
username = Tach(docfile.ReadLine());  
password = Tach(docfile.ReadLine());
```

Đọc thông tin tập tin lưu trữ kết nối

■ Loại App.config

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<configuration>
```

```
  <connectionStrings>
```

```
    <add name="SqlServer" connectionString="server=HIENLTH-PC\squlexpress;database=SEQLHS;integrated security=true" providerName="System.Data.SqlClient"/>
```

```
  </connectionStrings>
```

```
</configuration>
```


Đọc thông tin tập tin lưu trữ kết nối

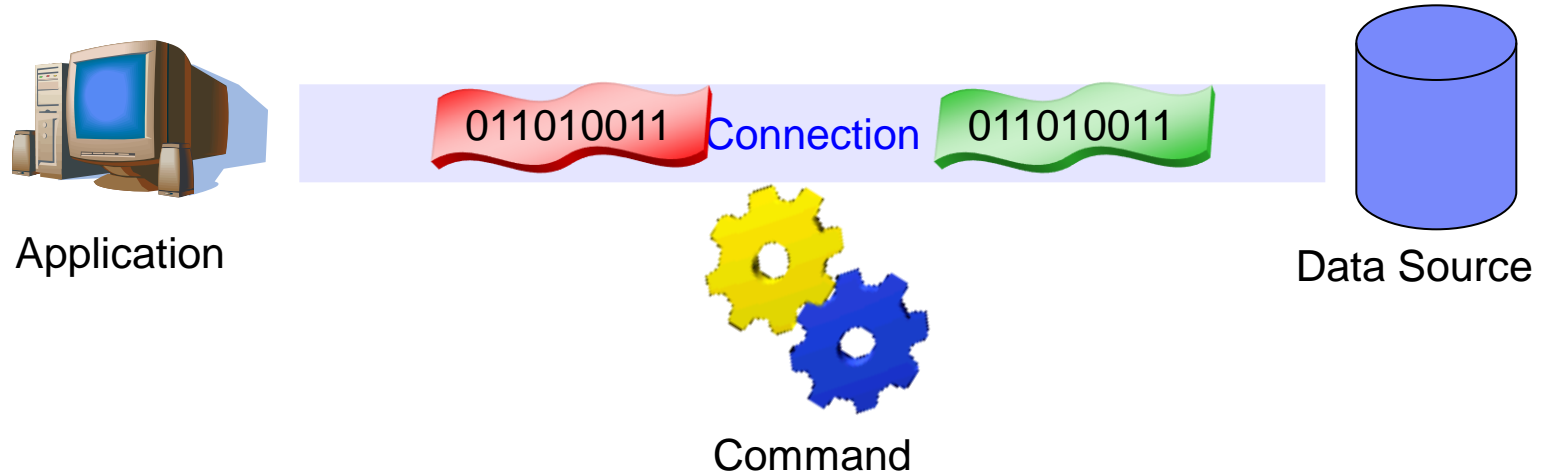
- Loại **App.config**:

```
connectionstring =  
    ConfigurationManager.ConnectionStrings[tên].  
        ConnectionString;
```

Ví dụ:

```
con.ConnectionString =  
    ConfigurationManager.ConnectionStrings["SqlServer"].  
        ConnectionString;
```

.NET Data Provider - Command



.NET Data Provider - Command

- **Các đối tượng Command tuân thủ IDbCommand interface.**
 - **Connection**: kết nối dùng để thực hiện câu lệnh.
 - **CommandText**: câu lệnh SQL cần thực hiện trên Data Source.
 - **CommandType**: loại câu lệnh trong CommandText (Text, TableDirect, StoredProc).
 - **ExecuteScalar()**: thực hiện câu lệnh trong CommandText, kết quả trả về là một giá trị đơn.
 - **ExecuteNonQuery()**: thực hiện câu lệnh trong CommandText và không có kết quả trả về.
 - **ExecuteReader()**: thực hiện câu lệnh trong CommandText, kết quả trả về là một DataReader.

Command (cont) – SQL Server

```
SqlConnection cnn = new SqlConnection("server=HIENLTH-PC\\SQLEXPRESS; database=QLHS; user id=sa; password=sa");  
SqlCommand cmd = new SqlCommand();  
  
cmd.Connection = cnn;  
cmd.CommandText = "SELECT COUNT(*) FROM HocSinh";  
cmd.CommandType = CommandType.Text;  
cnn.Open();  
int count = (int)cmd.ExecuteScalar();  
cnn.Close();
```

Command (cont) – SQL Server

```
SqlConnection cnn = new SqlConnection("server=HIENLTH-PC\\SQLEXPRESS; database=QLHS; user id=sa; password=sa");  
SqlCommand cmd = new SqlCommand();
```

```
cmd.Connection = cnn;
```

```
cmd.CommandText =
```

```
    "INSERT INTO HocSinh(Ho, Ten, DienThoai)  
    VALUES(N'Nguyễn Văn', N'Trường', '0989366990')";
```

```
cmd.CommandType = CommandType.Text;
```

```
cnn.Open();
```

```
cmd.ExecuteNonQuery();
```

```
cnn.Close();
```

.NET Data Provider - Parameter

■ Mục đích sử dụng:

- Một vài giá trị trong câu lệnh chỉ biết khi thực hiện câu lệnh.
- Cần thực hiện câu lệnh nhiều lần với các giá trị khác nhau.

■ Các bước thực hiện:

- Tham số hóa câu lệnh: **?** hoặc **@[tên tham số]**.
- Tạo các parameters tương ứng cho command.
- Đặt giá trị cho các parameter mỗi khi dùng command thực hiện câu lệnh.

.NET Data Provider – Parameter(cont)

■ Tham số hóa

➤ SQL Data Provider:

```
cmd.CommandText =  
    "SELECT * FROM HocSinh WHERE tenhocsinh = @ten";
```

```
cmd.CommandText =  
    "INSERT INTO HocSinh(id_hocsinh, tenhocsinh, dtb)" +  
    " VALUES(@id, @ten, @dtb)";
```

➤ Các provider khác:

```
cmd.CommandText =  
    "SELECT * FROM HocSinh WHERE tenhocsinh = ?";
```

```
cmd.CommandText =  
    "INSERT INTO HocSinh(id_hocsinh, tenhocsinh, dtb)" +  
    "VALUES(?, ?, ?)";
```

.NET Data Provider – Parameter(cont)

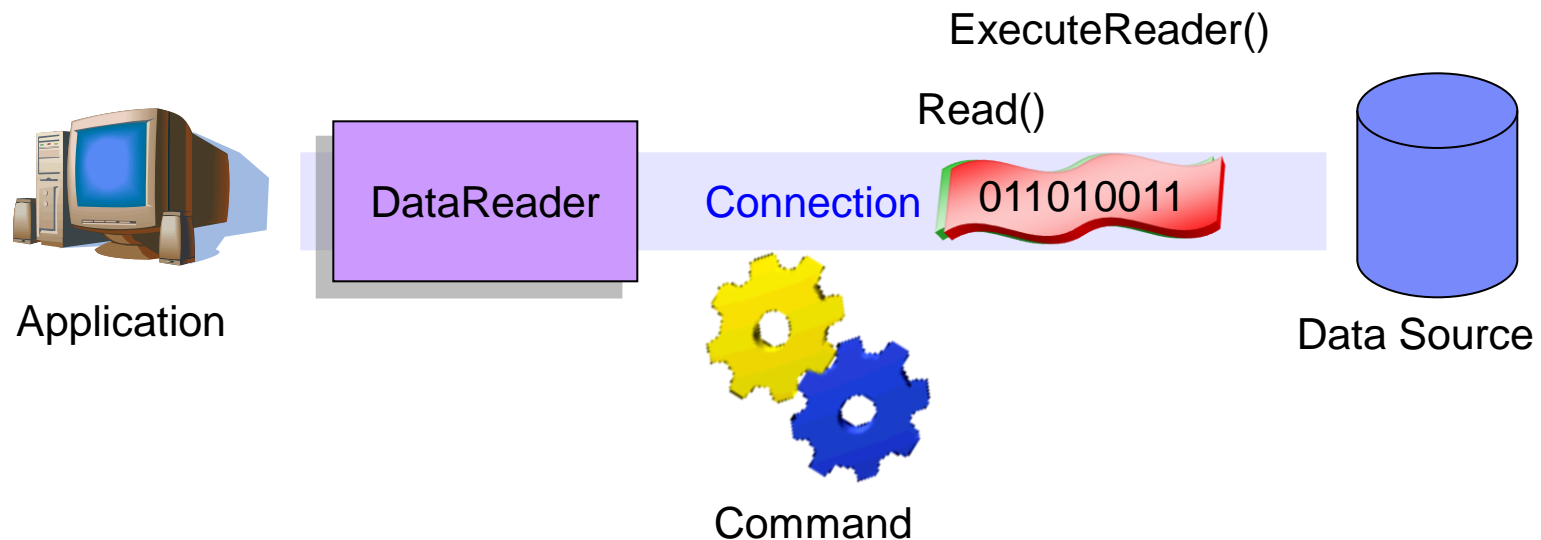
Tạo các parameter

```
cmd.Parameters.Add("@id", 5);  
cmd.Parameters.Add("@ten", "Nguyễn Văn A");  
cmd.Parameters.Add("@dtb", 8.5);
```

Đặt giá trị cho các parameter

```
foreach (Student s in studentList)  
{  
    cmd.Parameters["@id"] = i;  
    cmd.Parameters["@ten"] = s.studentName;  
    cmd.Parameters["@dtb"] = s.studentMarks;  
    cmd.ExecuteNonQuery();  
}
```


.NET Data Provider – DataReader



MaHS	HoTen	DiaChi
i	HocSinh i	DiaChi i

.NET Data Provider – DataReader

■ DataReader là gì?

- Chỉ di chuyển tới phía trước, chỉ đọc
- truy cập dữ liệu nhanh,
- kết nối đến nguồn dữ liệu (data source)
- Quản lý dữ liệu, hoặc ràng buộc vào điều khiển list-bound
- Sử dụng tài nguyên

.NET Data Provider - DataReader

- **Các đối tượng DataReader tuân thủ interface IDataReader.**
 - **HasRow**: cho biết còn dữ liệu để đọc nữa không.
 - **Read()**: đọc một mẫu tin vào DataReader.
 - **Toán tử [i]**: truy xuất đến cột i trong mẫu tin đọc được.
 - **Close()**: đóng DataReader.
- **Lưu ý:**
 - Truy xuất tuần tự và không quay lui.
 - Không cập nhật dữ liệu.
 - Cơ chế kết nối.

.NET Data Provider – DataReader (cont)

```
SqlConnection cnn = new SqlConnection("server=localhost;  
database=Northwind; user id=sa; password=sa");
```

```
SqlCommand cmd = new SqlCommand();  
cmd.Connection = cnn;  
cmd.CommandText = "select * from Orders";  
cmd.CommandType = CommandType.Text;
```

```
cnn.Open();  
SqlDataReader dr = cmd.ExecuteReader();
```

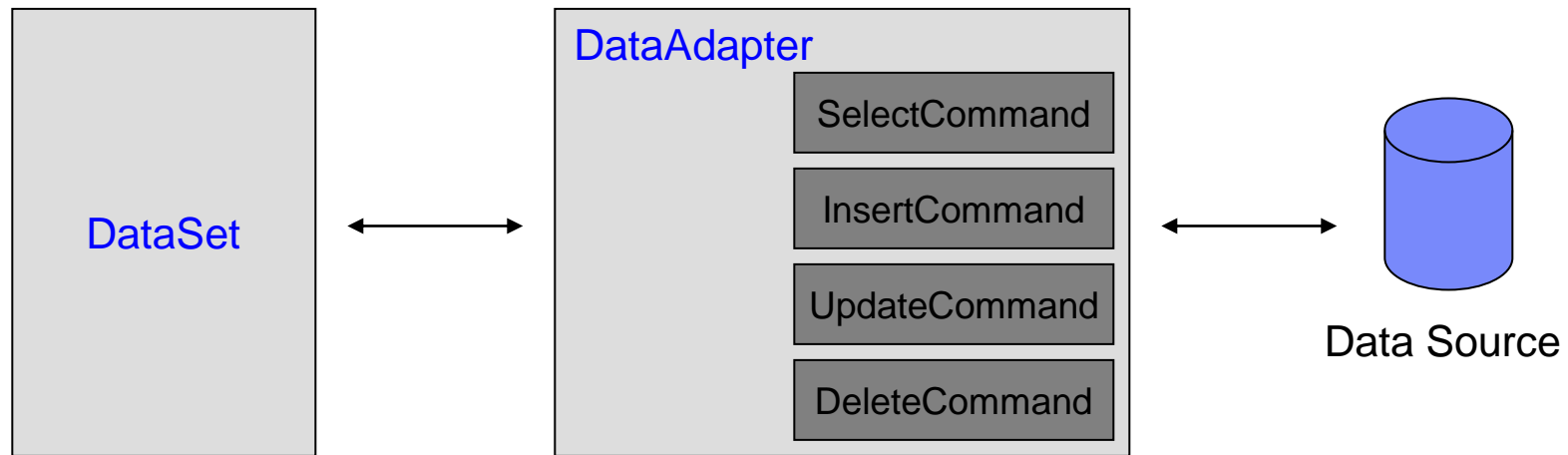
```
while (dr.Read())  
{  
    MessageBox.Show(dr["CustomerID"].ToString());  
}
```

```
dr.Close();  
cnn.Close();
```

.NET Data Provider - SqlDataAdapter

- **Bộ đọc dữ liệu từ CSDL nguồn (SQLServer) và đổ vào đối tượng DataSet hay DataTable.**
- **Phân trang dữ liệu**
- **Phương thức:**
 - Fill(datasource)
 - Fill(datasource, start, number, “tablename”)

.NET Data Provider - DataAdapter



■ Các đối tượng DataAdapter tuân thủ interface IDbDataAdapter.

- **Fill(DataSet)**: dùng SelectCommand lấy dữ liệu từ Data Source đổ vào DataSet.
- **Update(DataSet)**: dùng InsertCommand, UpdateCommand và DeleteCommand cập nhật dữ liệu trong DataSet vào Data Source.

.NET Data Provider – DataAdapter (cont)

```
SqlConnection cnn = new  
    SqlConnection("server=localhost;  
    database=Northwind; user id=sa; password=sa");  
  
SqlDataAdapter da = new SqlDataAdapter("select *  
    from Orders", cnn)  
  
DataSet ds = new DataSet();  
  
da.Fill(ds);  
  
// Does something on the DataSet.  
da.Update(ds);
```

Nội dung

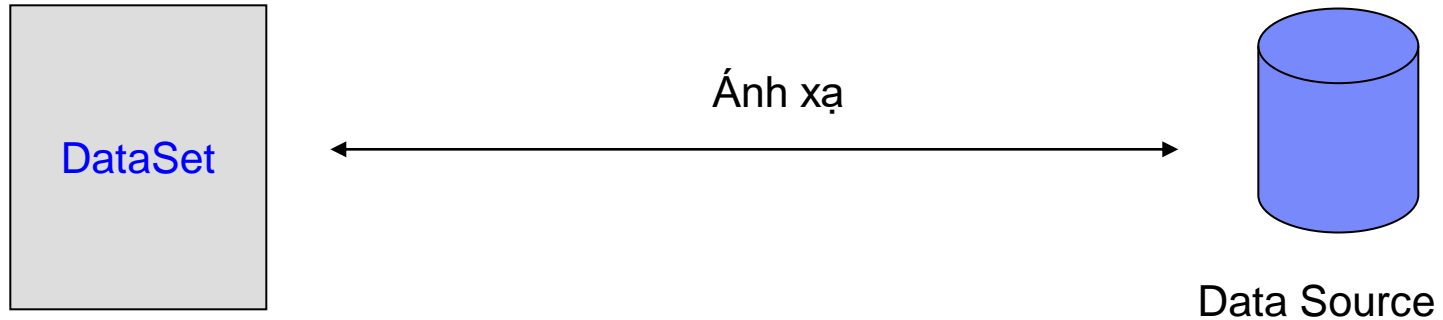
- Sơ lược lịch sử phát triển
- Kiến trúc ADO.NET
- .NET Data Provider
- DataSet
- Hỏi & Đáp

System.Data namespace

- **DataSet** — This object is designed for disconnected use and can contain a set of DataTables and include relationships between these tables.
- **DataTable** — A container of data that consists of one or more DataColumnns and, when populated, will have one or more DataRows containing data.
- **DataRow** — A number of values, akin to a row from a database table, or a row from a spreadsheet.
- **DataColumn** — This object contains the definition of a column, such as the name and data type.
- **DataRelation** — A link between two DataTable classes within a DataSet class. Used for foreign key and master/detail relationships.
- **Constraint** — This class defines a rule for a DataColumn class (or set of data columns), such as unique values.

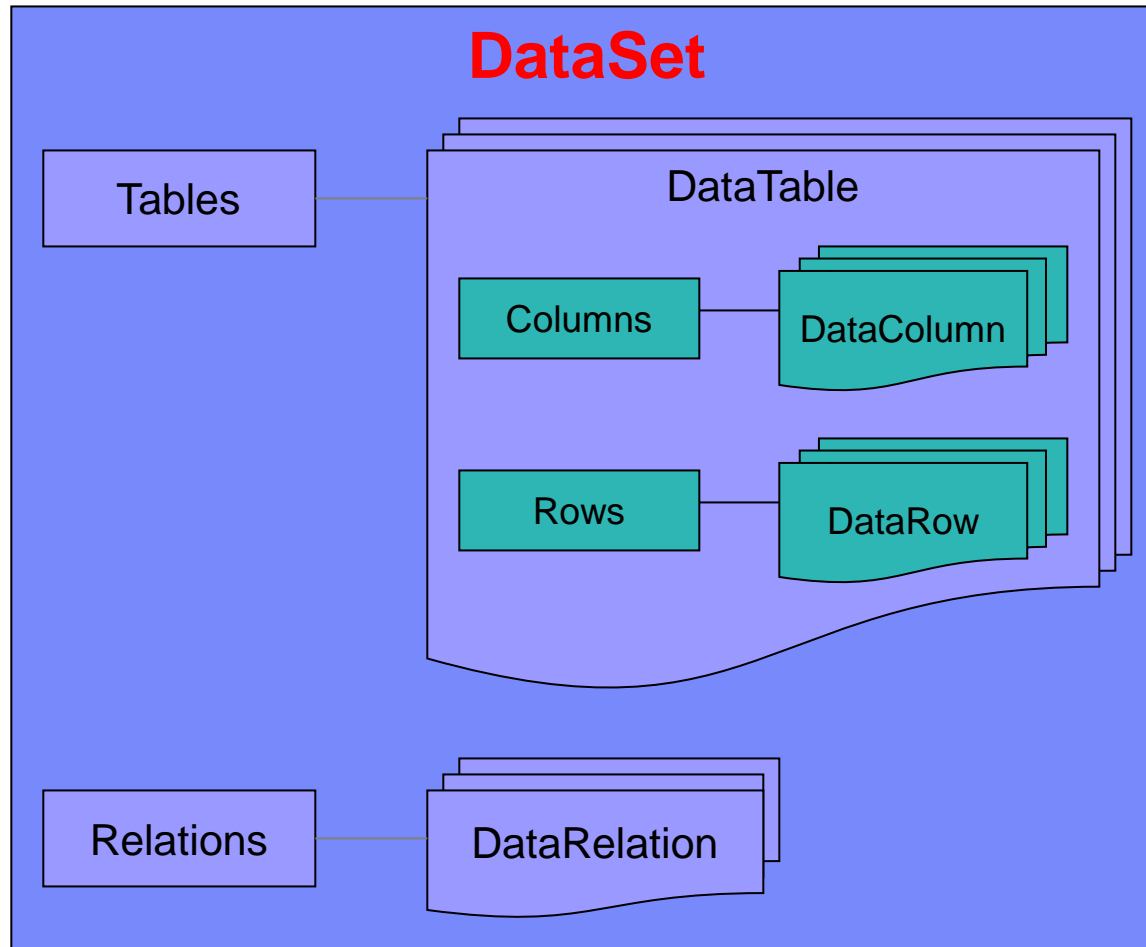
DataSet

■ DataSet là gì?



- **DataSet là cơ sở dữ liệu được lưu trữ trong bộ nhớ chính (in-memory database).**
- **Cơ chế không kết nối.**
- **Gồm các đối tượng**
 - DataTable
 - DataRelation

DataSet (cont)



Các phương thức của DataSet

- Thêm 1 đối tượng DataTable:
`ds.Tables.Add(datatable_name);`
- Xóa 1 đối tượng:
 - `ds.Tables.Remove(datatable_name);`
 - `ds.Tables.RemoveAt(datatable_index);`
- Loại bỏ tất cả DataTable: `ds.Tables.Clear();`
- Kiểm tra tồn tại:
`ds.Tables.Contains(datatable_name);`
- Kiểm tra tồn tại và có thể xóa được:
`ds.Tables.CanRemove(datatable_name);`
- Đếm số lượng DataTable: `ds.Tables.Count` (thuộc tính)

Các phương thức của DataSet

- Ghi ra file XML: `ds.WriteXml(ten_file_xml);`
- Đọc từ file XML: `ds.ReadXml(ten_file_xml);`

DataSet - DataTable

- **DataTable** thể hiện một bảng trong cơ sở dữ liệu.
- **Gồm các đối tượng:**
 - DataColumn
 - DataRow
- **Các thuộc tính và phương thức:**
 - **TableName**: tên bảng.
 - **Columns**: danh sách các cột (DataColumn).
 - **Rows**: danh sách các mẫu tin (DataRow).
 - **PrimaryKey**: danh sách các cột làm khóa chính (DataColumn).
 - **NewRow()**: tạo một mẫu tin mới.

DataSet - DataColumn

- **DataColumn** thể hiện một cột trong bảng.
- **Các thuộc tính và phương thức:**
 - **ColumnName**: tên cột.
 - **DataType**: kiểu dữ liệu cột.

DataSet - DataRow

- **DataRow** thể hiện một mẫu tin trong bảng.
- **Các thuộc tính và phương thức:**
 - **RowState**: trạng thái của mẫu tin (Added, Modified, Deleted, Unchanged, Detach).
 - **Toán tử [i]**: truy xuất đến cột i của mẫu tin.
 - **Delete()**: đánh dấu xóa mẫu tin.

DataSet (cont)

```
DataTable table = new DataTable("SinhVien");  
  
table.Columns.Add(new DataColumn("MSSV", Type.GetType("Int32")));  
table.Columns.Add(new DataColumn("HoTen", Type.GetType("string")));  
  
table.PrimaryKey = new DataColumn[] { table.Columns["MSSV"] };  
  
DataRow row = table.NewRow();  
row["MSSV"] = 123;  
row["HoTen"] = "Nguyễn Văn A";  
table.Rows.Add(row);
```

DataSet (cont)

```
SqlConnection cnn = new  
    SqlConnection("server=HIENLTH-PC\SQLEXPRESS;  
    database=Northwind; user id=sa; password=sa");  
SqlDataAdapter da = new SqlDataAdapter("select * from  
    Orders", cnn)  
DataSet ds = new DataSet();  
  
da.Fill(ds);  
foreach (DataRow row in ds.Tables[0].Rows)  
    row["OrderDate"] = DateTime.Now;  
da.Update(ds);
```

DataSet (cont)

```
SqlConnection cnn = new SqlConnection("server=HIENLTH-PC\\SQLEXPRESS; database=Northwind; user id=sa; password=sa");
```

```
SqlDataAdapter da = new SqlDataAdapter("select * from Orders",  
cnn)
```

```
DataSet ds = new DataSet();
```

```
da.Fill(ds);
```

```
foreach (DataRow row in ds.Tables[0].Rows)
```

```
    if (row["CustomerID"].ToString() == "Nguyễn Văn A")
```

```
        row.Delete();
```

```
da.Update(ds);
```

DataSet (cont)

■ So sánh DataSet và DataReader:

- Tốc độ truy xuất.
- Bộ nhớ lưu trữ.
- Thuận tiện trong thao tác.
- Cơ chế kết nối và không kết nối.

Trình bày dữ liệu

Lập trình Windows Form với C#

Nhắc lại SqlDataAdapter

- **SqlDataAdapter** được dùng để điền dữ liệu vào đối tượng **DataSet**, **DataTable** và cập nhật dữ liệu từ hai đối tượng này trở lại dữ liệu nguồn.
- **Các trường hợp tạo đối tượng SqlDataAdapter:**
 - `SqlDataAdapter(string SQL, SqlConnection cnn);`
 - `SqlDataAdapter(SqlCommand cmd);`
- **Điền dữ liệu:**
 - `sqlDataAdapter.Fill(datasetname);`
 - `sqlDataAdapter.Fill(datatable);`
 - `sqlDataAdapter(DataSet dataset, int start, int end);`
 - `sqlDataAdapter(int start, int end, DataTable datatable);`

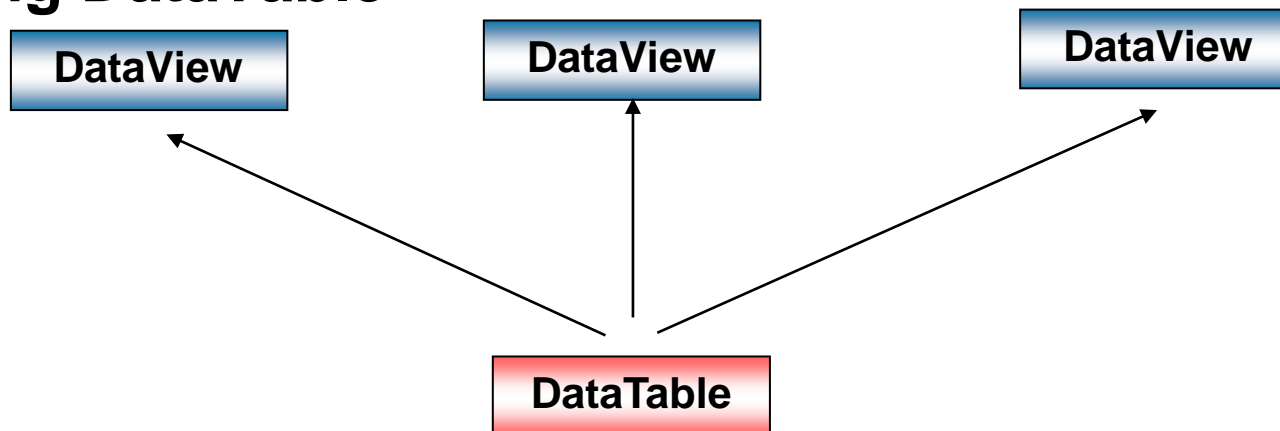


Cập nhật dữ liệu từ DataAdapter

- Trong Form khai báo các thuộc tính là đối tượng DataSet (hay DataTable)
- Lấy dữ liệu: Đổ dữ liệu vào DataSet
`dataAdapter.Fill(dataset);`
- Cập nhật dữ liệu: Cập nhật dữ liệu từ DataSet vào CSDL
`dataAdapter.Update(dataset);`

DataView

- Thể hiện của 1 DataTable.
- Đóng vai trò quan trọng trong DataBinding
- 1 DataTable có thể có nhiều View khác nhau
- Dùng để trình bày dữ liệu dưới hình thức lọc, sắp xếp, tìm kiếm, hiệu chỉnh và điều hướng dữ liệu trong DataTable



Khai báo DataView

- **Có thể sử dụng các cú pháp sau:**
 - `dataView = new DataView();
dataView.Table = dataTable;`
 - `dataView = new DataView(dataTable);`
 - `dataView = new DataView(dataTable, sortFilter,
sortString, DataViewRowState);`

DataView

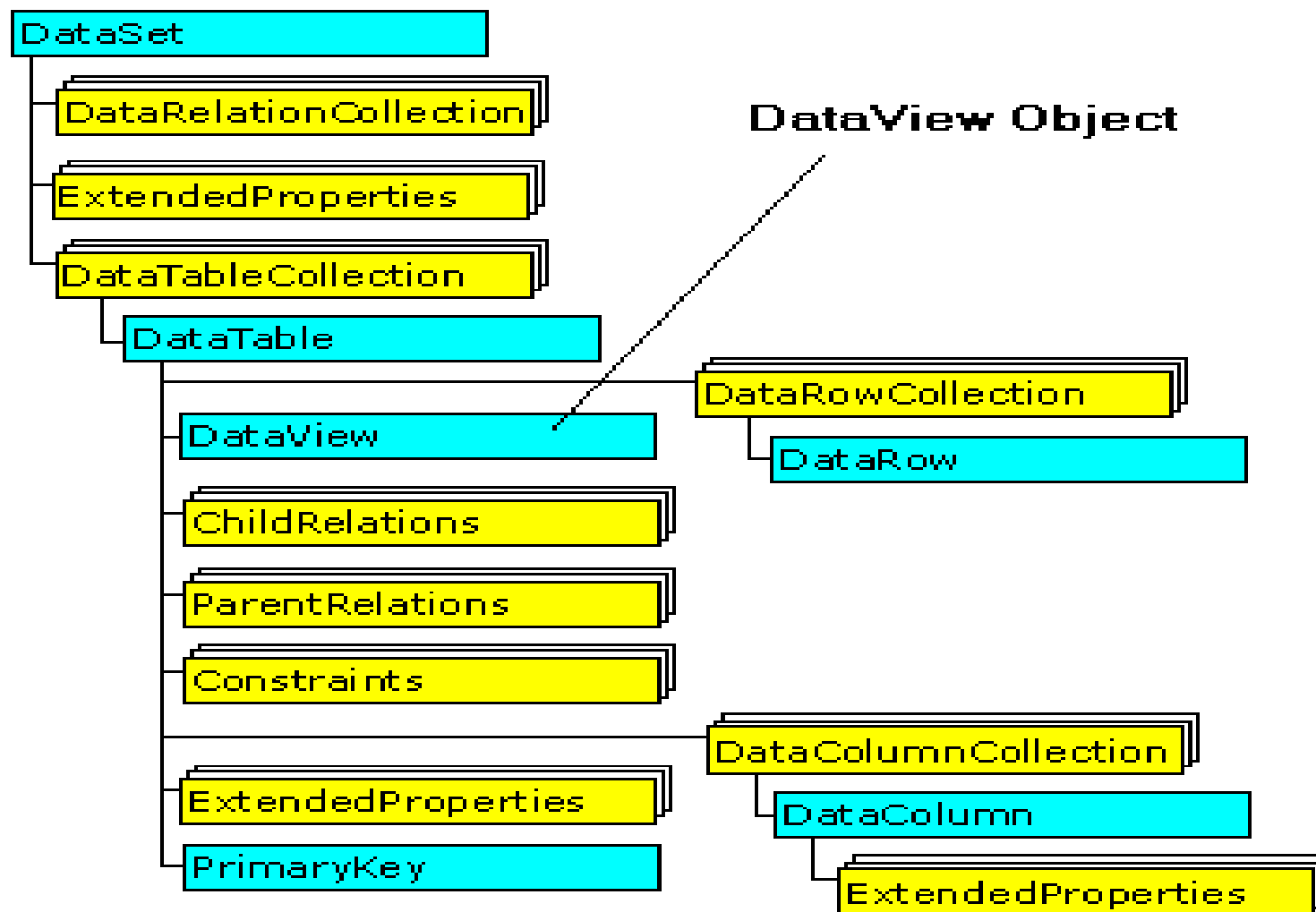
Danh sách các thuộc tính

Tên	Ý nghĩa
AllowDelete	Cho phép xóa trên thể hiện
AllowEdit	Cho phép chỉnh sửa
AllowNew	Cho phép thêm mới
Item (index)	Lấy giá trị value tại column có chỉ số index
RowFilter	Thiết lập Expression dùng để lọc row
Sort	Sắp xếp tăng hoặc giảm theo column
Table	Cho biết view này được tạo bởi table nào
Count	Lấy số lượng mẫu tin có trong DataView

DataView

Danh sách các phương thức

Tên	Ý nghĩa
AddNew	Thêm mới 1 mẫu tin
Delete(index)	Xóa mẫu tin thứ index



DataBinding

- **Hiển thị dữ liệu trong DataTable, ... vào các control (TextBox, ComboBox, DataGrid, ...)**
- **Gồm 2 loại:**
 - Simple Binding
 - Complex Binding
- **Ví dụ: Điền nội dung bảng TonGiao vào ComboBox Tôn Giáo**

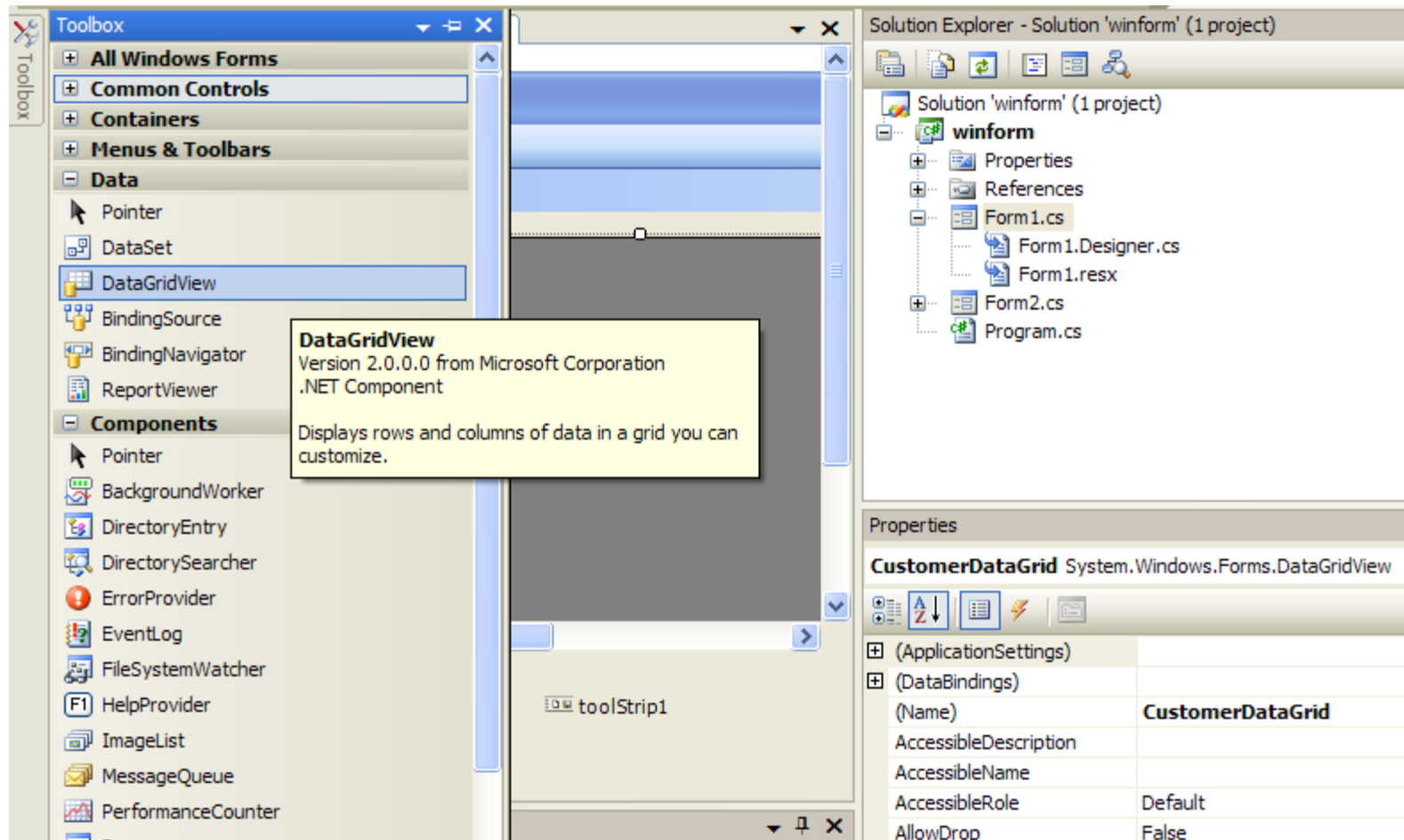
```
DataTable bang = ketnoi.GetDataTable("SELECT * FROM  
TonGiao");
```

```
cboTonGiao.DataSource = bang;
```

```
cboTonGiao.DisplayMember =  
    bang.Columns[1].ToString();//TenTG
```

```
cboTonGiao.SelectedValue =  
    bang.Columns[0].ToString();//MaTG
```

Đưa dữ liệu vào lưới DataGridView



Điền dữ liệu vào DataGridView

- `datagridview.DataSource = datatable;`
- `datagridview.DataSource = dataset.Tables[index];`

DataGridView (tt)

- **Các thuộc tính của DataGridView:**
 - **DataSource**: điền dữ liệu vào DataGridView
 - **CurrentRow**: lấy mẫu tin đang chọn
 - **CurrentCell**: lấy ô đang chọn
- **Các biến cố (Event) của DataGridView:**
 - **SelectionChanged**: xảy ra khi người dùng di chuyển vào vùng dữ liệu
 - **DoubleClick**

Nội dung

- **Sơ lược lịch sử phát triển**
 - **Kiến trúc ADO.NET**
 - **.NET Data Provider**
 - **DataSet**
- **Hỏi & Đáp**

Tham khảo

- **Bài giảng ADO.NET của thầy Nguyễn Minh Huy, ĐH KHTN TpHCM**
- **MSDN**

