Lập trình và Thiết kế Web 2

Bài 7

# Thao tác CSDL với ADO .NET

Ths. Trần Thị Bích Hạnh

*Khoa CNTT – ĐH.KHTN*





## Nội dung

- Giới thiệu ADO .NET
- Kiến trúc ADO .NET
- Mô hình sử dụng ADO .NET
- .NET Data Provider






## Nội dung

- Giới thiệu ADO .NET
- Kiến trúc ADO.NET
- Mô hình sử dụng ADO .NET
- .NET Data Provider



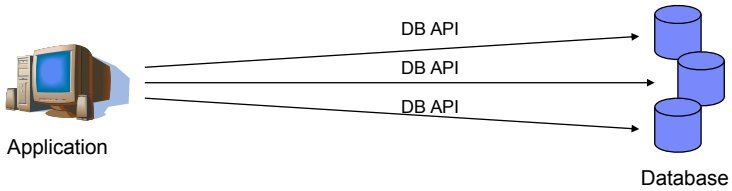



## ADO.NET là gì?

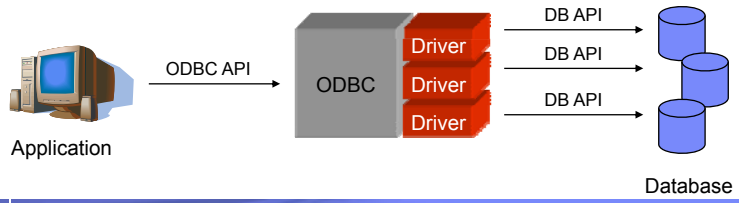
- ADO.NET
  - ActiveX Data Object .NET
  - Công nghệ của Microsoft
  - Phát triển từ ADO
  - Cung cấp các đối tượng và hàm thư viện dùng để kết nối và xử lý trên CSDL

## Sơ lược lịch sử phát triển

- **Native API**




- **Open DataBase Connectivity**



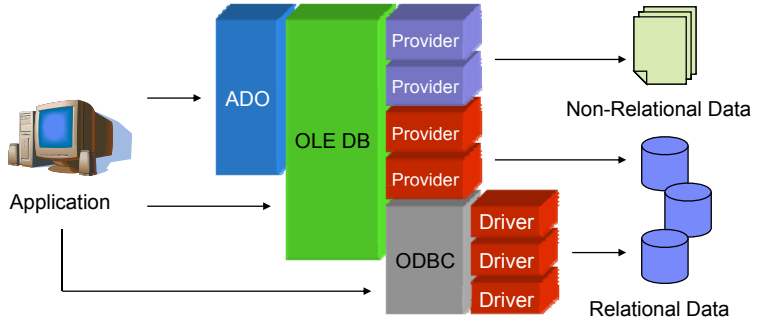






## Sơ lược lịch sử phát triển (tt)

- **OLEDB và ADO**



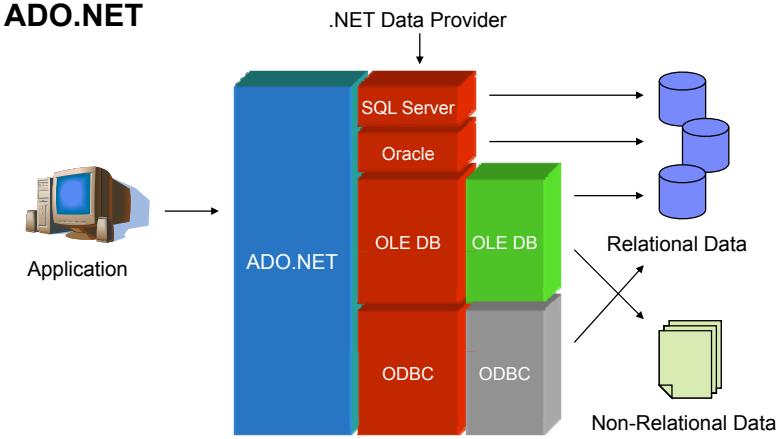






## Sơ lược lịch sử phát triển (tt)

- **ADO.NET**





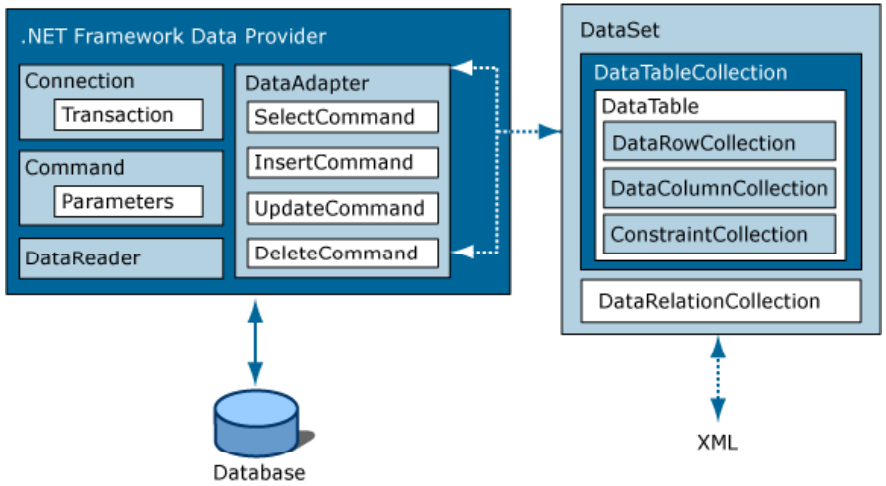




## Nội dung

- Giới thiệu ADO.NET
- Kiến trúc ADO .NET
- Mô hình sử dụng ADO .NET
- .NET Data Provider

## Kiến trúc của ADO.NET









## Namespaces

- Nếu ứng dụng .NET cần truy xuất dữ liệu → Phải khai báo namespace ADO.NET tương ứng với dữ liệu cho ứng dụng
- Đối với dữ liệu **OLE**
  - **using System.Data;**
  - **using System.Data.OleDB;**
- Đối với dữ liệu **SQL Server**
  - **using System.Data;**
  - **using System.Data.SqlClient;**





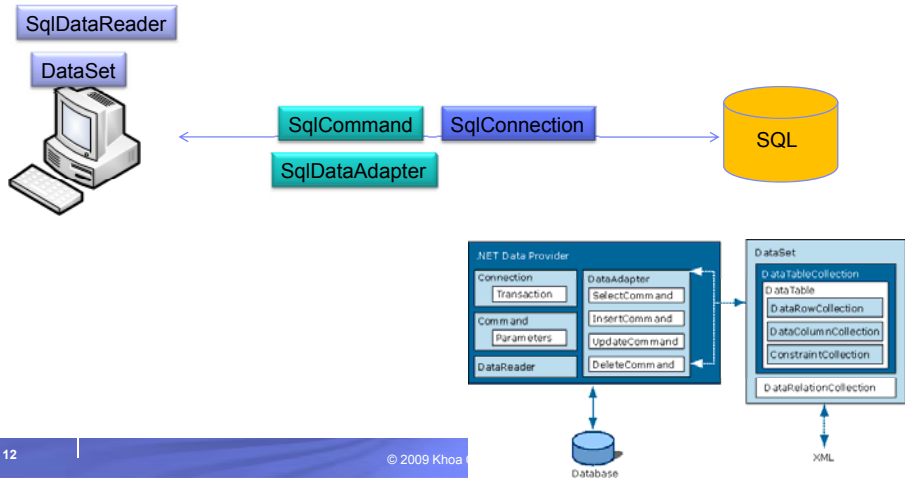

## Nội dung

- Giới thiệu ADO.NET
- Kiến trúc ADO .NET
- ➢ Mô hình sử dụng ADO .NET
- .NET Data Provider

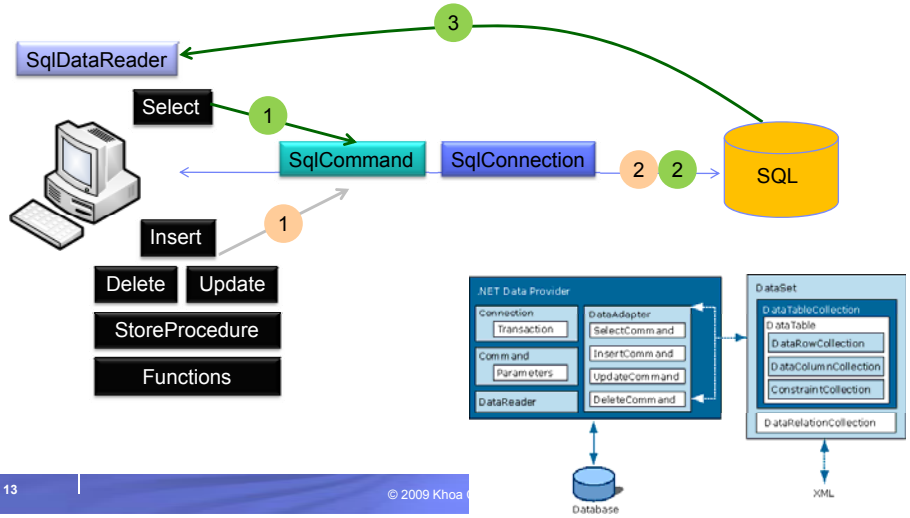





## Mô hình sử dụng ADO.NET với Phần mềm



## Mô hình sử dụng ADO.NET với Phần mềm

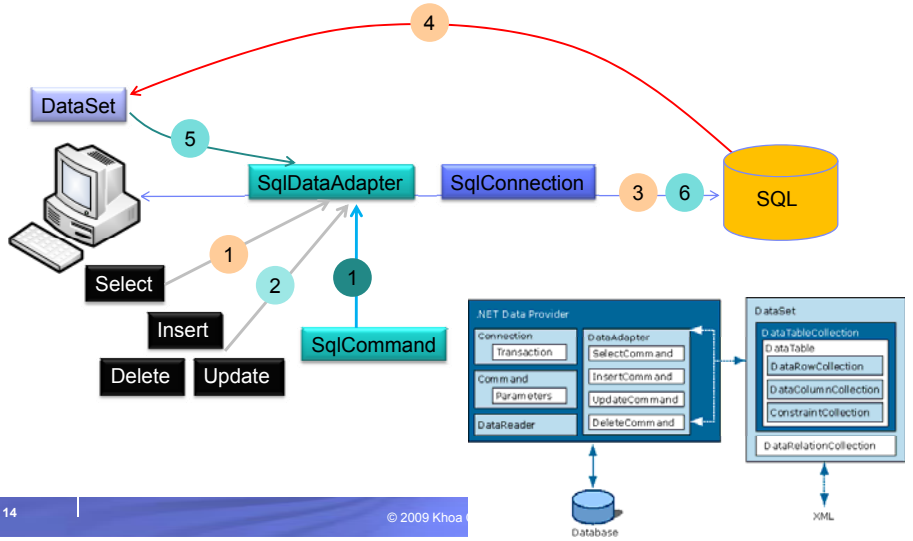








## Mô hình sử dụng ADO.NET với Phần mềm









## Nội dung

- Giới thiệu ADO.NET
- Kiến trúc ADO .NET
- Mô hình sử dụng ADO .NET
- .NET Data Provider






## .NET Data Provider

- Connection
- Command & Parameter
- DataReader
- DataSet & DataAdapter
- Transaction

# .NET Data Provider - Connection


- Thiết lập kết nối đến Data Source
- Thuộc tính
  - **ConnectionString**: Lưu chuỗi kết nối đến Data Source
  - **State**: cho biết tình trạng của kết nối
- Phương thức
  - **Open()**: thiết lập kết nối đến Data Source.
  - **Close()**: ngắt kết nối đến Data Source.






## Ví dụ

```
using System.Data.SqlClient;

string sConnectionString =
  "Initial Catalog=Northwind;
   Data Source=localhost;
   user=sa;
   password=sa;";

SqlConnection cnn = new SqlConnection();
cnn.ConnectionString = sConnectionString;

cnn.Open();
// do somethings
cnn.Close();
```






## Connection string (Access, SQL Server,…)

- Tạo connection string

| Database | ODBC/OLEDB Connection String |
|---|---|
| Microsoft Access | `Provider=Microsoft.Jet.OLEDB.4.0; Data Source=ĐườngDẫnĐếnFileAccess` |
| Microsoft SQL | `Provider=SQLOLEDB;Data Source=ServerName; Initial Catalog=DatabaseName; UserId=Username; Password=Password;` |

```
String strConn = string.Format("Provider=Microsoft.Jet.OLEDB.4.0; Data Source={0}", HttpContext.Current.Server.MapPath("database/mydb.mdb"));
```

**REF: http://www.connectionstrings.com/**






## Đường dẫn tới tập tin Access

- HttpContext.Current.Server.MapPath(StringPath)
  - ánh xạ đường dẫn tương đối StringPath thành đường dẫn đến thư mục vật lý trên Server
  - Ví dụ: Giả sử tập tin aspx sử dụng hàm Server.MapPath được lưu tại **D:\MyWebsite**

| | Kết quả |
|---|---|
| `Server.MapPath("myDB.mdb");` | `D:\MyWebsite\myDB.mdb` |
| `Server.MapPath("Database/myDB.mdb");` | `D:\MyWebsite\Database\myDB.mdb` |
| `Server.MapPath("../myDB.mdb");` | `D:\myDB.mdb` |

## Cấu hình lưu ConnectionString trong Web.Config

```
// Web.Config
<configuration>
<connectionStrings>
        <add    name="OleDbConnectionString"
        connectionString="Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=|DataDirectory|\DatabasePath" />
        <add    name="SqlConnectionString"
        connectionString="SQLOLEDB;Data Source=ServerName; Initial
Catalog=DatabaseName; UserId=Username; Password=Password" />
</connectionStrings>
<system.web>
…
</system.web>
</configuration>
```

```
// WebForm.aspx.cs
using System.Configuration;
string strConn =
ConfigurationManager.ConnectionStrings["SqlConnectionString"].ToString();
```






## Connection Pooling

```
SqlConnection conn = new SqlConnection();
conn.ConnectionString =
  "Integrated Security=SSPI;Initial Catalog=northwind";
conn.Open();  // Pool A is created.
```

```
SqlConnection conn = new SqlConnection();
conn.ConnectionString =
  "Integrated Security=SSPI;Initial Catalog=pubs";
conn.Open();
// Pool B is created because the connection strings differ.
```

```
SqlConnection conn = new SqlConnection();
conn.ConnectionString =
  "Integrated Security=SSPI;Initial Catalog=northwind";
conn.Open(); // The connection string matches pool A.
```





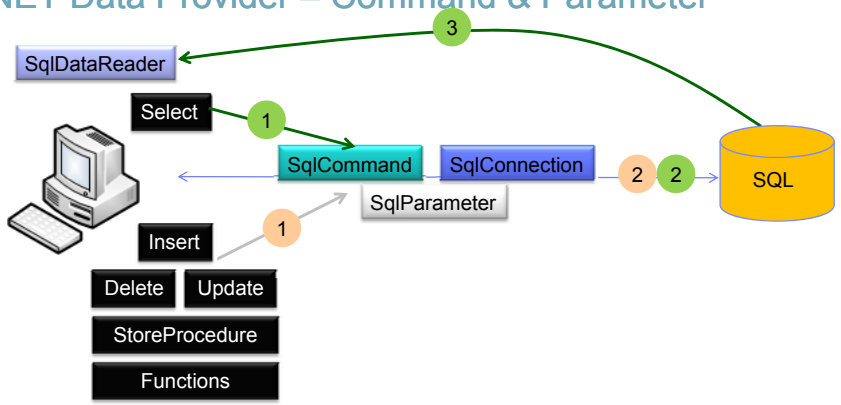

## .NET Data Provider

- Connection
- Command & Parameter
- DataReader
- DataSet & DataAdapter
- Transaction






## .NET Data Provider – Command & Parameter




- Thực thi câu truy vấn
- Hỗ trợ tham số vào, tham số ra, và giá trị trả về

## Command - Hàm khởi tạo và Thuộc tính

| Các hàm khởi tạo |
|---|
| `new ???Command()` |
| `new ???Command(cmdText)` |
| `new ???Command(cmdText, connection)`<br>`new ???Command(cmdText, connection, transaction)` |

| Thuộc tính | Ý nghĩa |
|---|---|
| `.Connection` | Trỏ đến đối tượng kết nối |
| `.CommandType` | `CommandType.Text` (mặc định)<br>`CommandType.StoreProcedure`<br>`CommandType.TableDirect` |
| `.CommandText` | Câu truy vấn SQL hoặc tên Store, tên Bảng |
| `.CommandTimeOut` | Thời gian chờ đợi thực thi 1 câu sql |
| `.Parameters` | Danh sách các tham số truyền vào |






## Command - Phương thức

| Phương thức | Ý nghĩa |
|---|---|
| `.ExecuteReader()` | Trả về một DataReader |
| `.ExecuteNonQuery()` | Trả về số lượng dòng bị ảnh hưởng trên CSDL |
| `.ExecuteScaler()` | Trả về 1 giá trị đầu tiên (VD: giá trị tính tổng) |
| `.ExecuteXMLReader()` | Trả về 1 XMLReader |






## Ví dụ - OleDbCommand

```
using System.Data.OleDb;
OleDbConnection cnn = new OleDbConnection();
cnn.ConnectionString =
string.Format("Provider=Microsoft.Jet.OLEDB.4.0; Data
     Source={0}", Server.MapPath("~/App_Data/QLHS.mdb"));
OleDbCommand cmd = new OleDbCommand();
cmd.Connection = cnn;
cmd.CommandText = "INSERT INTO HocSinh(id_hocsinh,
     tenhocsinh, dtb)VALUES(5, 'Nguyễn Văn A', 8.5)";
cmd.CommandType = CommandType.Text;
cnn.Open();
cmd.ExecuteNonQuery();
cnn.Close();
```






## Ví dụ - SqlCommand

```
using System.Data.SqlClient;

SqlConnection cnn = new SqlConnection();
cnn.ConnectionString = "Initial Catalog=Northwind;
   Data Source=localhost; user=sa; password=sa;";
SqlCommand cmd = new SqlCommand();
cmd.Connection = cnn;
cmd.CommandText = "SELECT COUNT(*) FROM Orders";
cmd.CommandType = CommandType.Text;
cnn.Open();
int count = (int)cmd.ExecuteScalar();
cnn.Close();
```

## .NET Data Provider - Parameter

- Định nghĩa tham số truyền vào cho đối tượng Command
- Có các thuộc tính sau :

| Thuộc tính | Ý nghĩa |
|---|---|
| `ParameterName` | Tên tham số |
| `SqlDbType` | Kiểu dữ liệu của tham số tương ứng với kiểu dữ liệu của SqlServer |
| `Direction` | Input, Output, InputOutput, ReturenValue, … |
| `Size` | Kích thước tối đa của dữ liệu |
| `Value` | Giá trị của tham số (input / Output) |






## Parameter – Cách sử dụng

- 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 truy vấn: **?** 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.






## Parameter – Tham số hóa câu truy vấn

**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(?, ?, ?)";
```






## Parameter – Tạo các tham số cho Command

**Sql Data Provider:**

```
cmd.Parameters.Add("id", SqlDbType.Int);
cmd.Parameters.Add("ten", SqlDbType.NVarChar);
cmd.Parameters.Add("dtb", SqlDbType.Float);
```

**OleDb Data Provider:**

```
cmd.Parameters.Add("id", OleDbType.Integer);
cmd.Parameters.Add("ten", OleDbType.VarWChar);
cmd.Parameters.Add("dtb", OleDbType.Numeric);
```

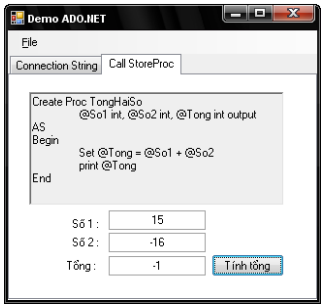
## Parameter – Đặt giá trị cho các tham số và thực thi

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






## Ví dụ - Gọi StoredProcedure




```
private void buttonTinhTong_Click(object sender, EventArgs e)
{
    _cnn.Open();

    // Tao doi tuong SqlCommand
    SqlCommand cmd = new SqlCommand();
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "TongHaiSo";
    cmd.Connection = _cnn;

    // Tao tham so cho SqlCommand
    SqlParameter para;
    para = new SqlParameter("So1", SqlDbType.Int, 4);
    para.Direction = ParameterDirection.Input;
    para.Value = int.Parse(textBoxSo1.Text);
    cmd.Parameters.Add(para);
    para = new SqlParameter("So2", SqlDbType.Int, 4);
    para.Direction = ParameterDirection.Input;
    para.Value = int.Parse(textBoxSo2.Text);
    cmd.Parameters.Add(para);
    para = new SqlParameter("Tong", SqlDbType.Int, 4);
    para.Direction = ParameterDirection.Output;
    cmd.Parameters.Add(para);

    // Thuc thi viec goi Storeproc
    cmd.ExecuteNonQuery();

    // Xuat ket qua
    textBoxTong.Text = cmd.Parameters["Tong"].Value.ToString();

    // Dong ket noi
    _cnn.Close();
}
```





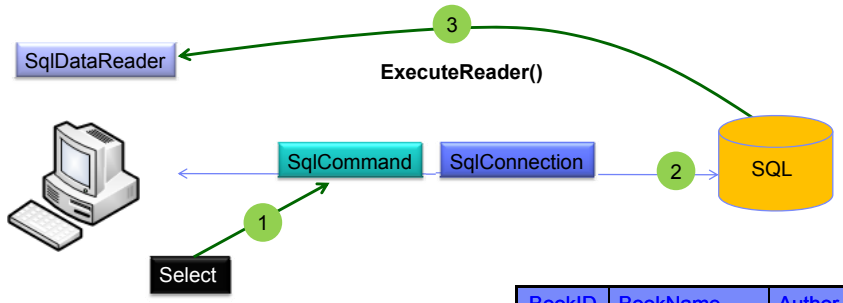

## .NET Data Provider

- Connection
- Command & Parameter
- DataReader
- DataSet & DataAdapter
- Transaction






## .NET Data Provider - DataReader




| BookID | BookName | Author |
|---|---|---|
| i | Book i | Author i |

- Truy xuất tuần tự và không quay lui
- Chỉ đọc, Không cập nhật dữ liệu
- Chỉ lưu lại 1 record kết quả trong bộ nhớ với mỗi lần truy xuất

## DataReader

- Một số thuộc tính & phương thức :

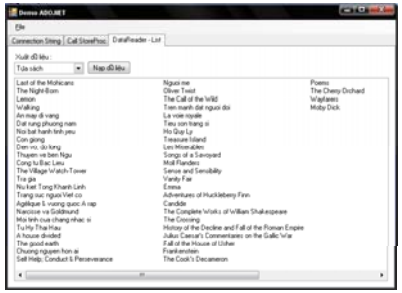
| Thuộc tính | Ý nghĩa |
|---|---|
| `HasRows` | Trả về xem DataReader có đọc được dữ liệu nào không. |
| `FieldCount` | Trả về số lượng thuộc tính trong dòng hiện tại (đang đọc) |
| `[int/string]` | Trả về giá trị của thuộc tính đang yêu cầu |

| Phương thức | Ý nghĩa |
|---|---|
| `Read()` | Đọc record dữ liệu kế tiếp |
| `IsDBNull(i)` | Kiểm tra xem giá trị cột i có bị null không |
| `Close()` | Đóng DataReader |






## Ví dụ: Load dữ liệu vào List



```
private void buttonNapDuLieu_Click(object sender, EventArgs e)
{
    listViewDS.Items.Clear();
    string sql;
    if (comboBoxBang.SelectedIndex == 0) // Doc gia
        sql = "Select ho+' '+ten+' '+tenlot as Hoten From Docgia";
    else // Tua sach
        sql = "Select Tuasach From Tuasach";

    _cnn.Open();

    SqlCommand cmd = new SqlCommand(sql, _cnn);
    SqlDataReader reader = cmd.ExecuteReader();

    if (reader.HasRows == true)
    {
        // Doc du lieu
        if (comboBoxBang.SelectedIndex == 0) // Doc gia
        {
            while (reader.Read())
                listViewDS.Items.Add(reader["Hoten"].ToString());
        }
        else // Tua sach
        {
            while (reader.Read())
                listViewDS.Items.Add(reader["Tuasach"].ToString());
        }
    }

    _cnn.Close();
}
```





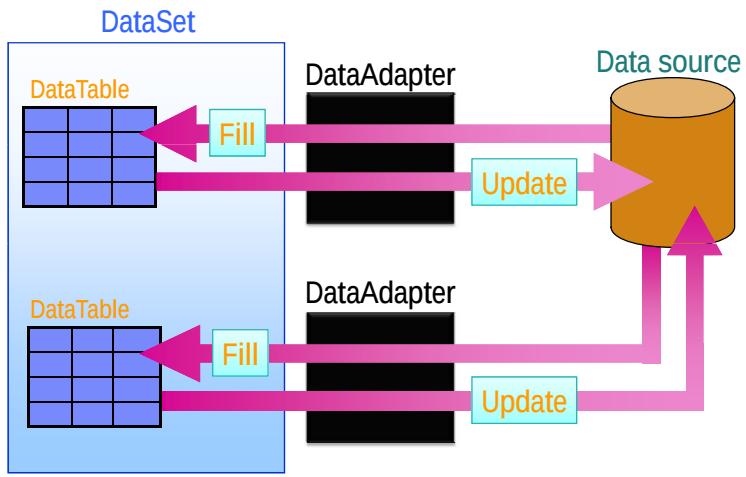

## .NET Data Provider

- Connection
- Command & Parameter
- DataReader
- DataSet & DataAdapter
- Transaction






## Đối tượng DataSet và DataAdapter



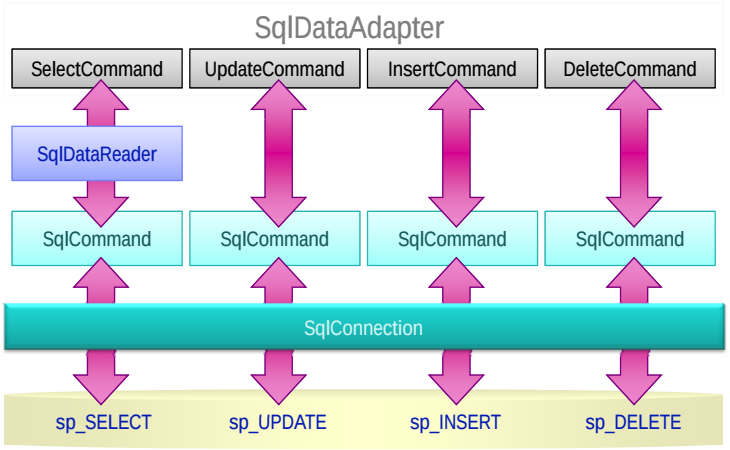
## Đối tượng DataSet và DataAdapter

- DataSet
  - Là cơ sở dữ liệu được lưu trữ trong bộ nhớ chính (in-memory database)
  - Mọi thao tác thay đổi dữ liệu được thực hiện trên DataSet, không làm ảnh hưởng đến CSDL
- DataAdapter
  - Fill: Lấy dữ liệu từ CSDL đổ vào DataSet
  - Update: Theo vết các thay đổi trên dữ liệu trên DataSet và cập nhật dữ liệu ngược vào CSDL






## Mô hình đối tượng DataAdapter









## Đối tượng SqlDataAdapter

- Một số thuộc tính và phương thức

| Thuộc tính | Ý nghĩa |
|---|---|
| `SelectCommand` | |
| `UpdateCommand` | |
| `InsertCommand` | |
| `DeleteCommand` | |

| Phương thức | Ý nghĩa |
|---|---|
| `Fill(Dataset)` | Lấy dữ liệu từ CSDL và đổ vào Dataset |
| `FillSchema()` | |
| `Update(…)` | Tiến hành cập nhật dữ liệu trên DataSet với CSDL |






## Đối tượng DataSet

- Một số thuộc tính và phương thức

| Thuộc tính | Ý nghĩa |
|---|---|
| `DataSetName` | |
| `Relations` | |
| `Tables` | Danh sách các table có trong Dataset |

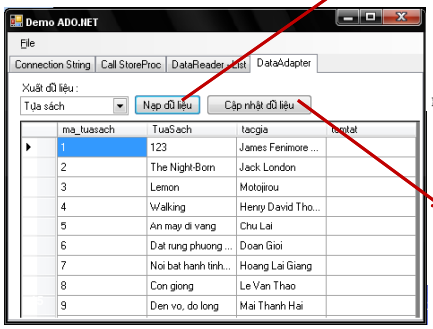
| Phương thức | Ý nghĩa |
|---|---|
| `GetChange()` | Trả về DataSet chứa các thay đổi trên DataSet đang xét |
| `RejectChanges()` | |
| `AcceptChanges()` | |
| `GetXML(), ReadXML(), WriteXML()` | |

## Ví dụ

```
using System.Data.SqlClient;

namespace DemoADO
{
    public partial class DemoADO : Form
    {
        private SqlConnection _cnn;
        private DataSet _ds;
        private SqlDataAdapter _da;

        public DemoADO()
```




```
private void buttonNapDuLieu2_Click(object sender, EventArgs e)
{
    dataGridViewData.DataSource = null;
    string sql;
    if (comboBoxChonBang.SelectedIndex == 0) // Doc gia
        sql = "Select * From Docgia";
    else // Tua sach
        sql = "Select * From Tuasach";

    _cnn.Open();
    _da = new SqlDataAdapter(sql, _cnn);
    SqlCommandBuilder sqlcb = new SqlCommandBuilder(_da);

    _ds = new DataSet();
    _da.Fill(_ds);

    dataGridViewData.DataSource = _ds.Tables[0];
    dataGridViewData.DataBind();

    _cnn.Close();
}

private void buttonUpdateData_Click(object sender, EventArgs e)
{
    if (_ds.HasChanges() == false)
    {
        MessageBox.Show("Nothing Change");
        return;
    }

    DataSet ds = _ds.GetChanges();
    _da.Update(ds);
}
```






## .NET Data Provider

- Connection
- Command & Parameter
- DataReader
- DataSet & DataAdapter
- Transaction



## Một số vấn đề truy xuất dữ liệu đồng thời

- Lost Update
- Dirty Reads
- Non-repeatable Reads
- Phantoms

| Accounts | |
|---|---|
| Number | int |
| Name | nvarchar(50) |
| Balance | double |






## Lost Update

**Transaction 1**

```
SELECT balance
FROM Accounts
WHERE number = 123;
```

```
UPDATE Accounts
SET balance = balance - 100
WHERE number = 123;
```

**Transaction 2**

```
SELECT balance
FROM Accounts
WHERE number = 123;
```

```
UPDATE Accounts
SET balance = balance - 500
WHERE number = 123;
```

## Dirty Reads

**Transaction 1**

```
SELECT balance
FROM Accounts
WHERE number = 123;
```

```
SELECT balance
FROM Accounts
WHERE number = 123
```

**Transaction 2**

```
UPDATE Accounts
SET balance = balance - 100
WHERE number = 123;
```

```
ROLLBACK
```






## Non-repeatable Reads

**Transaction 1**

```
SELECT balance
FROM Accounts
WHERE number = 123;
```

```
SELECT balance
FROM Accounts
WHERE number = 123
```

**Transaction 2**

```
UPDATE Accounts
SET balance=balance - 100
WHERE number = 123;
COMMIT
```






## Phantoms

**Transaction 1**

```
SELECT balance
FROM Accounts
```

```
SELECT balance
FROM Accounts
```

**Transaction 2**

```
INSERT INTO Accounts
VALUES(456, 'ABC', 5000);
COMMIT
```






## Mức độ cô lập - IsolationLevel

| IsolationLevel | Dirty Read | Non-Repeatable Read | Phantoms |
|---|---|---|---|
| ReadUncommitted | x | x | x |
| ReadCommitted | | x | x |
| RepeatableRead | | | x |
| Serializable | | | |

## Thực hiện giao tác với ứng dụng CSDL

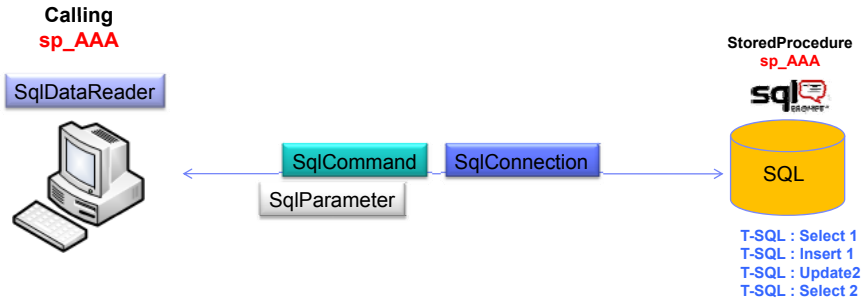
- Có 2 cách để thực hiện 1 giao tác (nhiều lệnh T-SQL) cho ứng dụng.
  - **Cách 1 :** Viết storeproc ở HQTSQL + Gọi thực hiện store thông qua đối tượng SqlCommand + SqlParamenter
  - **Cách 2 :** Quản lý Giao tác ở phía ứng dụng, chỉ gọi thực hiện từng câu truy vấn T-SQL






## Thực hiện giao tác với ứng dụng CSDL

- **Cách 1 :**



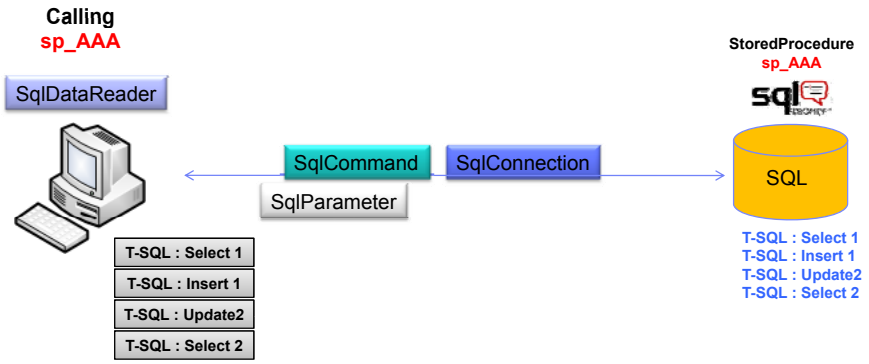






## Thực hiện giao tác với ứng dụng CSDL

- **Cách 2 :**



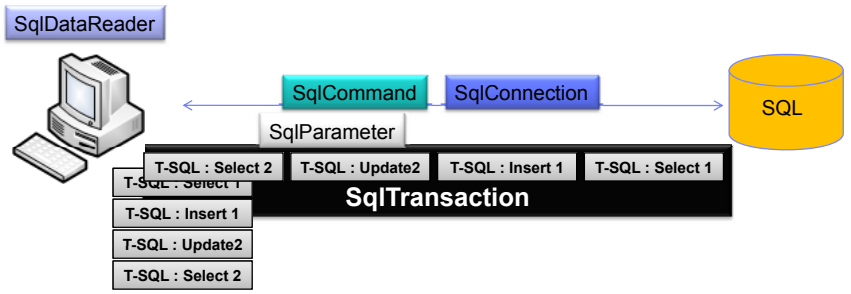






## Thực hiện giao tác với ứng dụng CSDL

- **Cách 2 :**



## Đối tượng SqlTransaction

- Một số thuộc tính và phương thức

| Thuộc tính | Ý nghĩa |
|---|---|
| `Connection` | Trỏ đến đối tượng SqlConnection cần tạo transaction |
| `IsolationLevel` | Enum Thiết lập mức cô lập cho Transaction |

| Phương thức | Ý nghĩa |
|---|---|
| `ConnObj.BeginTransaction()` | Khởi tạo Đối tượng SqlTransaction bằng Đối tượng SqlConnection |
| `Commit(…)` | |
| `RollBack(…)` | |
| `Save(SavePointName)` | Tạo save point để có thể Rollback |






## Ví dụ

```
// Create the connection and transaction objects.
SqlConnection cnn = new SqlConnection( strConnectString );
SqlTransaction trans = null;
try {
        cnn.Open();
        trans = cnn.BeginTransaction();
        // Do some stuff here...
        SqlCommand cmd = new SqlCommand( strQuery, cnn, trans );
        cmd.ExecuteNonQuery();
        // Commit the transaction.
        trans.Commit();
} catch {
        if( trans != null )
                trans.Rollback();
} finally {
        // Close the connection.
        if( cnn.State == ConnectionState.Open )
                cnn.Close();
}
```






## Ví dụ - Thiết lập IsolationLevel

```
namespace DemoADO
{
    public partial class DemoADO : Form
    {
        private SqlConnection _cnn;
        private DataSet _ds;
        private SqlDataAdapter _da;
        private SqlTransaction _trans;
```

```
private void buttonBeginTrans_Click(object sender, EventArgs e)
{
    IsolationLevel iso;
    switch (comboBoxMucCoLap.SelectedIndex)
    {
        case 0:
            iso = IsolationLevel.ReadUncommitted;
            break;
        case 1:
            iso = IsolationLevel.ReadCommitted;
            break;
        case 2:
            iso = IsolationLevel.RepeatableRead;
            break;
        case 3 :
            iso = IsolationLevel.Serializable;
            break;
        default :
            iso = IsolationLevel.ReadCommitted;
            break;
    }
    _cnn.Open();
    _trans = _cnn.BeginTransaction(iso);
}
```






## Tổng kết

- Phân biệt ODBC, OLEDB, ADO, ADO.NET
- Mô hình sử dụng ADO .NET
  - Connection
  - Command & Parameter
  - DataReader
  - DataAdapter & DataSet
  - Transaction
    - Các vấn đề truy xuất dữ liệu đồng thời
    - Các mức độ cô lập - IsolationLevel