

XÂY DỰNG THƯ VIỆN DATA ACCESS

Đỗ Ngọc Cường – ITDLU
Email: dnc.dlu@gmail.com


Mục lục


Xây dựng tầng Data	2
Xây dựng thư viện Data Access.....	3
Viết code cho lớp Post (Post.cs).....	5
Viết code cho lớp DataProvider (DataProvider.cs)	6
Viết code cho lớp SqlDataProvider (SqlDataProvider.cs).....	7
Bổ sung code vào lớp DataProvider	9
Test	10
Luyện tập	15
Làm tiếp các thủ tục còn lại.....	16
Thủ tục và các hàm bổ sung Post_Count	17
Thủ tục và các hàm bổ sung Post_All.....	17
Thủ tục và hàm bổ sung Post_Single	18
Thủ tục và hàm bổ sung Post_Find	19
Sử dụng lớp SqlHelper để tối ưu lớp SqlDataProvider	20
Sử dụng lớp CBO để tự động chuyển dữ liệu từ DataReader sang đối tượng	24
Cách sử dụng lớp CBO	25
Xây dựng thư viện Core ở mức tổng quát.....	27
Bổ sung code cho lớp DataProvider	29
Bổ sung code cho lớp SqlDataProvider	29
Bổ sung tiếp code cho lớp DataProvider	31
Xây dựng thư viện DataAccess sử dụng thư viện Core.dll	32




Xây dựng tầng Data

Tạo cơ sở dữ liệu đặt tên là **Sample**. Sau đó tạo một bảng như hình và thiết lập giá trị tự tăng cho cột PostID

Post			
	Column Name	Data Type	Allow Nulls
	PostID	int	<input type="checkbox"/>
	Title	nvarchar(50)	<input type="checkbox"/>
	Body	nvarchar(4000)	<input type="checkbox"/>
	Publish	datetime	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

	Column Name	Data Type	Allow Nulls
	PostID	int	<input type="checkbox"/>
	Title	nvarchar(50)	<input type="checkbox"/>
	Body	nvarchar(4000)	<input type="checkbox"/>
	Publish	datetime	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Column Properties	
	Identity Specification
	(Is Identity)
	Identity Increment
	Identity Seed

Yes
Yes
1
1

Quy tắc đặt tên thủ tục (Stored Procedure – SP) như sau: **X_Y**

Trong đó:

- X: Tên của bảng
- Y: Chức năng mà thủ tục thực hiện

Ví dụ:

Tên thủ tục

Post_All
Post_Single
Post_Find
Post_Add
Post_Update

Mô tả

Lấy tất cả record
Lấy **một** record theo PostID
Lấy nhiều record theo điều kiện nào đó
Thêm một record vào bảng Post
Cập nhật một record theo PostID



Post_Delete	Xóa một record theo PostID
Post_Count	Đếm tất cả record hoặc đếm theo điều kiện nào đó
Post_Paging	Lấy các record theo trang (phân trang)
...	...

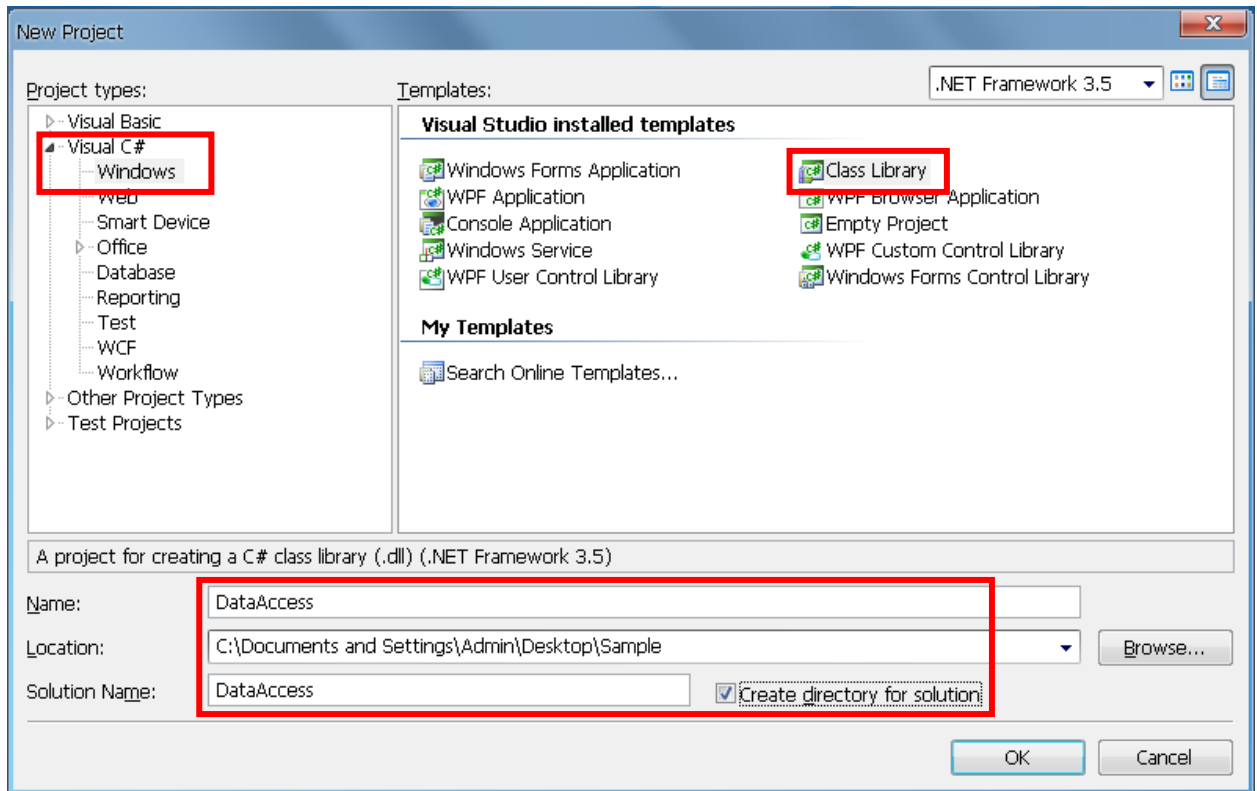
Tạo thủ tục Post_Add như sau: chú ý **tên** và **dữ liệu** của các tham số

```
CREATE PROCEDURE [Post_Add]
    @PostID int output, -- output: tương tu như truyền tham chiếu trong OOP
    @Title nvarchar(50),
    @Body nvarchar(4000),
    @Publish datetime
AS
INSERT INTO [Post]
(
    [Title],
    [Body],
    [Publish]
)
VALUES
(
    @Title,
    @Body,
    @Publish
)
-- Lay gia tri tu tang cua record vua moi them vao
SET @PostID = @@IDENTITY
```

Xây dựng thư viện Data Access

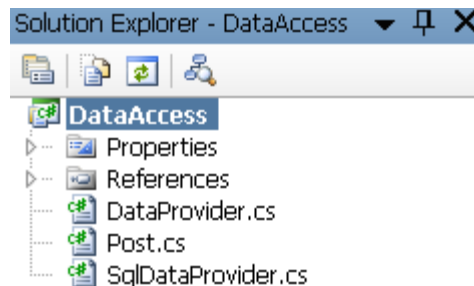
1. Tạo một thư mục ở Desktop đặt tên là **Sample**
2. Mở Microsoft Visual Studio (VS), tạo một project thư viện (Class Library Project) và đặt tên là **DataAccess**






3. Tạo 3 class:


Tên Class	Mô tả
DataProvider	Lớp trừu tượng chứa các phương thức abstract
SqlDataProvider	Lớp kế thừa từ lớp DataProvider, dùng cho SQL Server
Post	Lớp ánh xạ từ bảng Post



Bảng ánh xạ kiểu dữ liệu SQL Server sang kiểu dữ liệu trong C#

DBType		C# Type
...char		String
...text		String
bit		Bool
datetime		DateTime
smalldatetime		DateTime
smallint		Int16
int		Int32 (int)
bigint		Int64 (long)
float		float
decimal		Decimal
money		Double
real		Double
...		...

Viết code cho lớp Post (Post.cs)

Tên cột	DBType		C# Property	C# Type
PostID	int		PostID	int hoặc Int32
Title	Nvarchar(50)		Title	String
Body	Nvarchar(max)		Body	String
Publish	Datetime (được phép null)		Publish	DateTime?

Phiên bản .NET 2.0 (VS 2005) code như sau

```
using System;

namespace DataAccess
{
    public class Post
    {
        private int _PostID;
        public int PostID
        {
            get { return _PostID; }
            set { _PostID = value; }
        }

        private string _Title;
        public string Title
        {
            get { return _Title; }
            set { _Title = value; }
        }

        private string _Body;
```



```

        public string Body
        {
            get { return _Body; }
            set { _Body = value; }
        }

        private DateTime? _Publish;
        public DateTime? Publish
        {
            get { return _Publish; }
            set { _Publish = value; }
        }

        public Post()
        {
        }
    }
}

```

Các phiên bản từ .NET 3.5 về sau (từ VS 2008 về sau) có thể code rút gọn như sau hoặc theo cách code của phiên bản .NET 2.0

```

using System;

namespace DataAccess
{
    public class Post
    {
        public int PostID { get; set; }
        public string Title { get; set; }
        public string Body { get; set; }
        public DateTime? Publish { get; set; }
        public Post()
        {
        }
    }
}

```

Viết code cho lớp DataProvider (DataProvider.cs)

```

using System;
using System.Data;
namespace DataAccess
{
    public abstract class DataProvider
    {
        public abstract int PostAdd(Post post);
    }
}

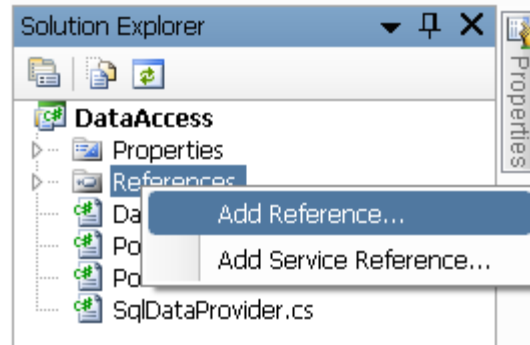
```



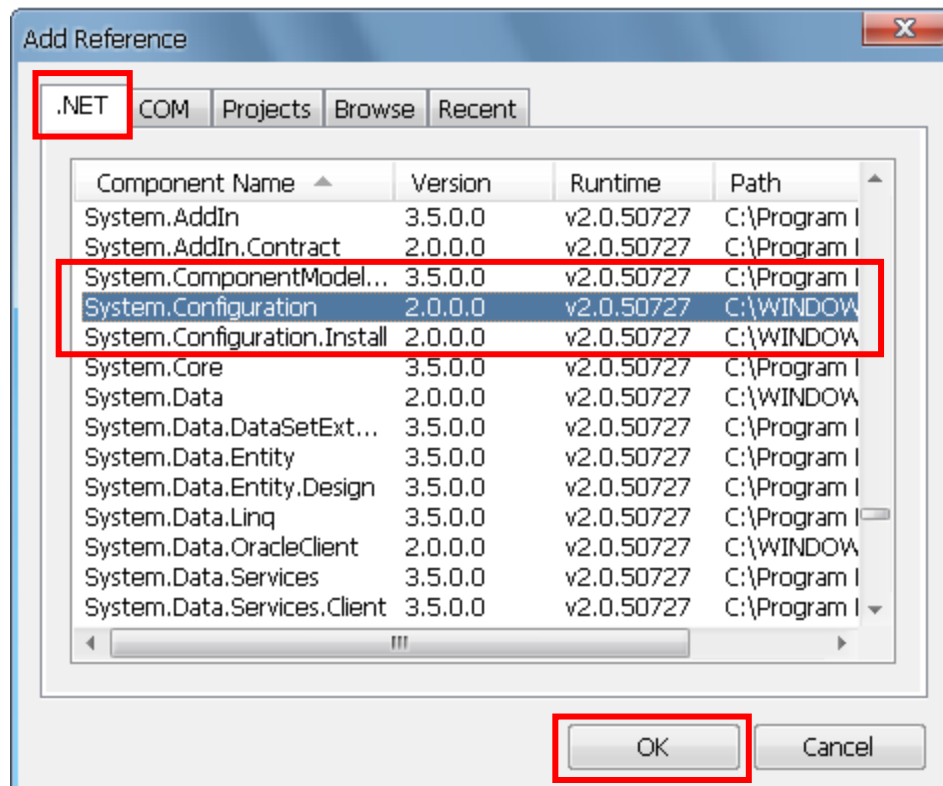
Viết code cho lớp SqlDataProvider (SqlDataProvider.cs)

Trong lớp SqlDataProvider có sử dụng lớp **ConfigurationManager** để lấy chuỗi kết nối từ **web config/app config**. Muốn sử dụng được lớp này thì phải thêm một thư viện System.Configuration vào project hiện tại. Cách thêm như sau:

Bấm chuột phải vào mục References (bên cửa sổ Solution) chọn Add Reference



Chọn thẻ .NET trong cửa sổ vừa mới hiện ra, sau đó chọn dòng **System.Configuration** và bấm OK




```

using System;
using System.Data.SqlClient;
using System.Configuration;
using System.Data;

namespace DataAccess
{
    public class SqlDataProvider : DataProvider
    {
        private string _ConnectionString;

        public SqlDataProvider(string connectionStringName)
        {
            // Lay chuỗi ket noi tu web config / app config
            _ConnectionString =
                ConfigurationManager.ConnectionStrings[connectionStringName]
                    .ConnectionString;
        }

        // Ham tao mot ket noi den CSDL

        protected SqlConnection GetSqlConnection()
        {
            try
            {
                return new SqlConnection(_ConnectionString);
            }
            catch
            {
                throw new Exception("SqlConnection");
            }
        }

        public override int PostAdd(Post post)
        {
            // 1. Tao doi tuong SqlConnection
            using (SqlConnection cnn = GetSqlConnection())
            {
                // 2. Tao doi tuong SqlCommand
                SqlCommand cmd = cnn.CreateCommand();
                // 2.1 Dat loai command la SP va ten thu tuc

                // Thuc hien truy van tu SP
                cmd.CommandType = CommandType.StoredProcedure;
                // Truyen ten cua SP
                cmd.CommandText = "Post_Add";

                // 2.2 Truyen ten, kieu du lieu va gia tri tham so
                // Tuong ung voi phan PostID output trong SP
                // Do cot PostID tu tang nen khong can truyen gia tri
                cmd.Parameters.Add("@PostID", SqlDbType.Int)
                    .Direction = ParameterDirection.Output;

                cmd.Parameters.Add("@Title", SqlDbType.NVarChar, 50)

```



```

        .Value = post.Title;
cmd.Parameters.Add("@Body", SqlDbType.NVarChar, 4000)
        .Value = post.Body;

// chỉ có kiểu dữ liệu nullable
// (thêm dấu ? sau kiểu dữ liệu)
// mỗi phải kiểm tra HasValue
if (post.Publish.HasValue)
    cmd.Parameters.Add("@Publish", SqlDbType.DateTime)
        .Value = post.Publish.Value;
else
    cmd.Parameters.Add("@Publish", SqlDbType.DateTime)
        .Value = DBNull.Value;

// 3. Mở kết nối
cnn.Open();

// Thực hiện thêm một record với các giá trị
// được truyền thông qua các Parameter
// Kết quả của hàm ExecuteNonQuery là
// số record được thêm vào CSDL
// > 0: thêm vào thành công
// = 0: không có hàng nào được thêm ~ thất bại
// 4. Gọi hàm ExecuteNonQuery của đối tượng SqlCommand
int rs = cmd.ExecuteNonQuery();
if (rs > 0)
    // 5. Lấy giá trị id tự tăng của record vừa thêm vào
    return (int)cmd.Parameters["@PostID"].Value;
return 0;
    }
}
}

```

Bổ sung code vào lớp DataProvider

```

private static DataProvider _Instance = null;
public static DataProvider Instance
{
    get
    {
        if (_Instance == null)
            _Instance = new SqlDataProvider("ConnectionString");
        return _Instance;
    }
}

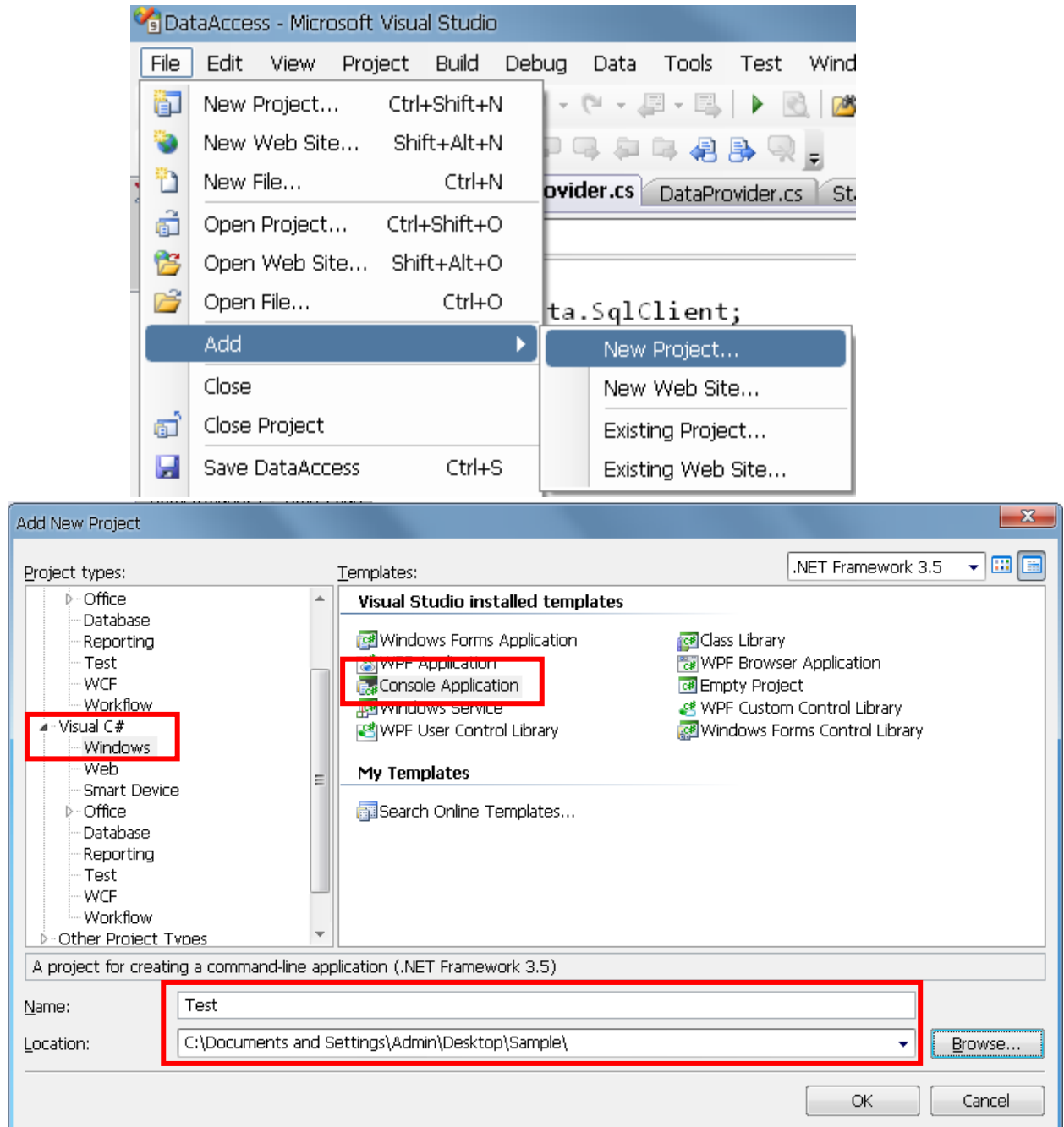
```

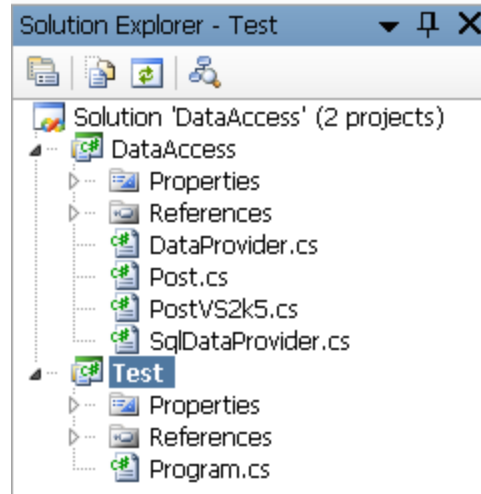
Bấm **Ctrl + Shift + B** hoặc chọn menu **Build > Build Solution** để build project



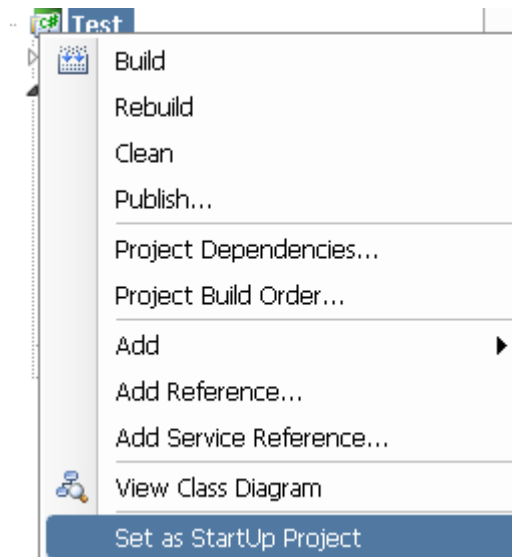
Test

1. Thêm project kiểu Console (Console Application) trong thư mục Sample và đặt tên là **Test**



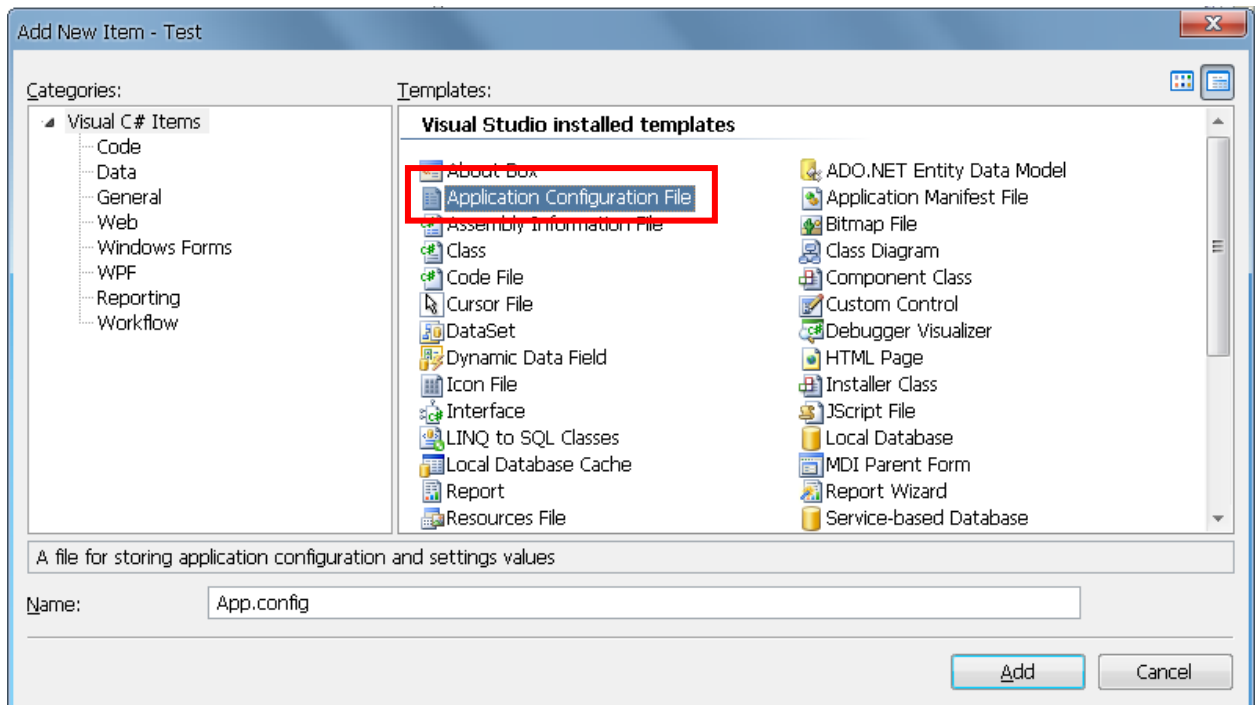


2. Set Startup Project: bấm chuột phải vào Project Test và chọn **Set as StartUp Project**



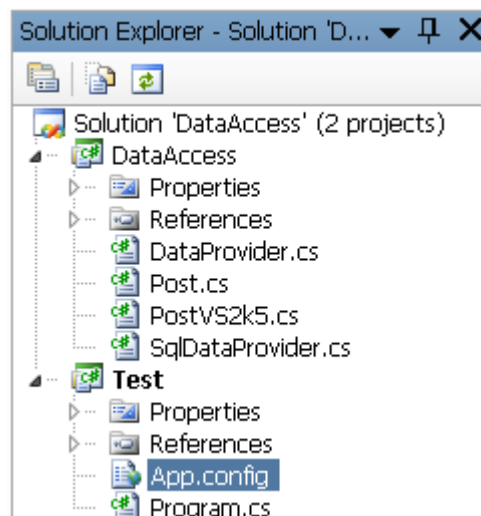
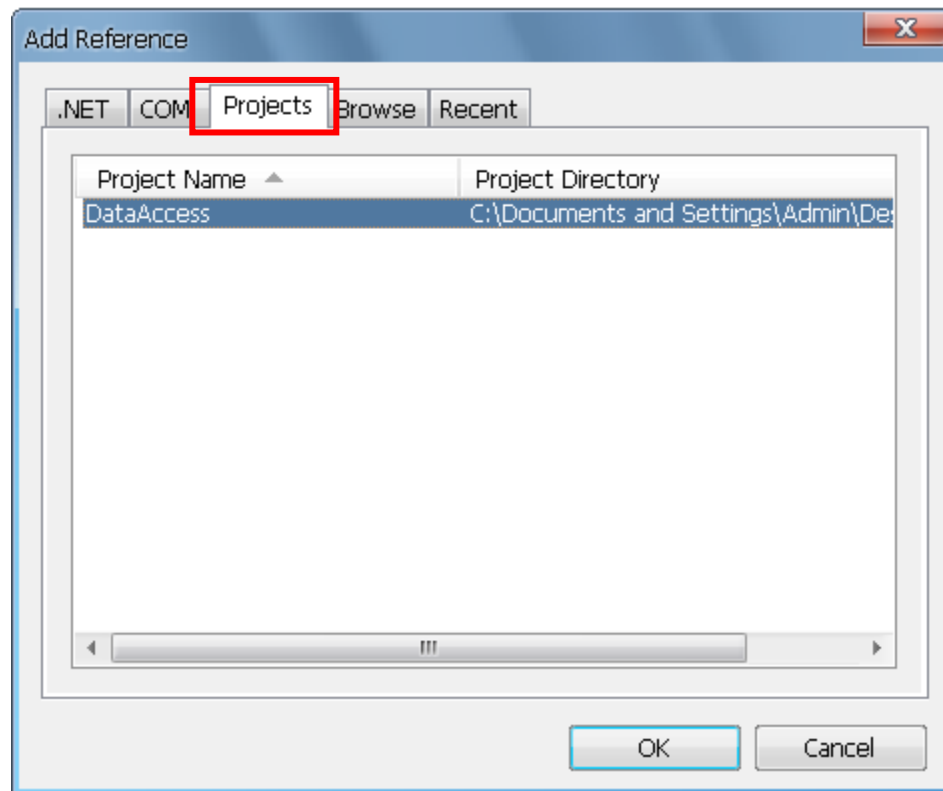
3. Thêm file App.Config project Console





4. Thêm thư viện DataAccess vào project Console





5. Mở tập tin App.config và thêm vào đoạn mã sau

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="ConnectionString"
      connectionString="server=.\\sqlexpress;database=sample;integrated
security=true"/>
  </connectionStrings>
</configuration>
```



```
</configuration>
```

Name: tương ứng với chuỗi **ConnectionString** trong thuộc tính **Instance** của lớp **DataProvider**

```
public static DataProvider Instance
{
    get
    {
        if (_Instance == null)
            _Instance = new SqlDataProvider("ConnectionString");
        return _Instance;
    }
}
```

ConnectionString: chuỗi kết nối (có thể khác nhau tùy từng máy, dấu “chấm” trong **.sqlexpress** tương đương với chữ **(local)\sqlexpress**)

6. Thêm đoạn code sau vào hàm **Main** trong lớp **Program.cs**

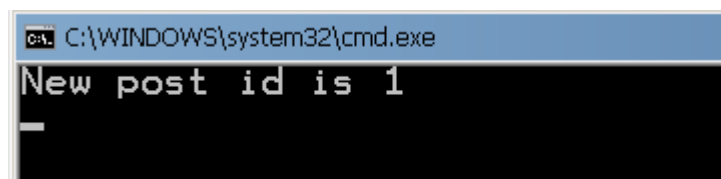
```
static void Main(string[] args)
{
    // Tao mot doi tuong de them vao CSDL
    Post p = new Post();
    p.Title = "A Title";
    p.Body = "Lorem Ip sum";

    // Them du lieu
    int rs = DataProvider.Instance.PostAdd(p);

    if (rs > 0)
        Console.WriteLine("New post id is " + rs);
    else
        Console.WriteLine("Insert failed!");

    Console.Read();
}
```

7. Chạy và xem kết quả



```
C:\WINDOWS\system32\cmd.exe
New post id is 1
```

8. Tương tự thêm một record khác với thuộc tính **Publish** đặt bằng **DateTime.Now**. Sau đó dùng **SQL Server** để kiểm dữ liệu vừa được thêm vào.



	PostID	Title	Body	Publish
1	1	A Title	Lorem Ip sum	NULL
2	2	A Title	Lorem Ip sum	2011-02-11 23:38:25.470

Luyện tập

1. Tương tự như trên tạo thủ tục Post_Update và bổ sung thêm phương thức PostUpdate vào thư viện DataAccess để gọi thủ tục trên.
2. Tạo thủ tục Post_Delete và phương thức PostDelete

Hướng dẫn:

Tạo thủ tục Post_Update

```
CREATE PROCEDURE [Post_Update]
    @PostID int,
    @Title nvarchar(50),
    @Body nvarchar(4000),
    @Publish datetime
AS
UPDATE [Post] SET
    [Title] = @Title,
    [Body] = @Body,
    [Publish] = @Publish
WHERE [PostID] = @PostID
```

Bổ sung thêm phương thức vào lớp DataProvider

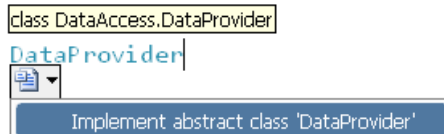
```
public abstract int PostUpdate(Post post);
```

Bổ sung code vào lớp SqlDataProvider

Để tiết kiệm thời gian bạn có thể rê chuột vào chữ DataProvider trong lớp SqlDataProvider và bấm vào nút mũi tên xuống > chọn **Implement abstract class** như hình sau để VS tự động sinh 1 phần mã

```
namespace DataAccess
{
    public class SqlDataProvider : DataProvider
    {

```



The screenshot shows the Visual Studio IDE. On the left, a code snippet shows the start of the SqlDataProvider class inheriting from DataProvider. On the right, a context menu is open, showing the option 'Implement abstract class 'DataProvider'', which is highlighted.

Sau khi bấm vào đó thì VS sẽ sinh ra đoạn mã như sau




```
public override int PostUpdate(Post post)
{
    throw new NotImplementedException();
}
```

Xóa dòng `throw new NotImplementedException()`. **Copy & Paste** từ hàm **PostAdd**, sau đó sửa lại những phần cần thiết để phù hợp với SP Post_Update. Phần in đậm là phần được thay đổi

```
public override int PostUpdate(Post post)
{
    using (SqlConnection cnn = GetSqlConnection())
    {
        SqlCommand cmd = cnn.CreateCommand();
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "Post_Update";
        cmd.Parameters.Add("@PostID", SqlDbType.Int)
            .Value = post.PostID;
        cmd.Parameters.Add("@Title", SqlDbType.NVarChar, 50)
            .Value = post.Title;
        cmd.Parameters.Add("@Body", SqlDbType.NVarChar, 4000)
            .Value = post.Body;
        if (post.Publish.HasValue)
            cmd.Parameters.Add("@Publish", SqlDbType.DateTime)
                .Value = post.Publish.Value;
        else
            cmd.Parameters.Add("@Publish", SqlDbType.DateTime)
                .Value = DBNull.Value;
        cnn.Open();
        return cmd.ExecuteNonQuery();
    }
}
```

Kiểm tra kết quả

Tạo thủ tục Post_Delete và làm tương tự. **Chú ý:** thủ tục này chỉ có 1 tham số

```
CREATE PROCEDURE [Post_Delete]
    @PostID int
AS
DELETE [Post]
WHERE [PostID] = @PostID
```

Làm tiếp các thủ tục còn lại

Post_Count	Đếm tất cả bài post
Post_All	Lấy tất cả bài post (sắp xếp theo một PostID giảm dần)
Post_Single	Tìm bài post theo PostID
Post_Find	Tìm kiếm các bài post theo Title



Hướng dẫn:

Thủ tục và các hàm bổ sung Post_Count

SP Post_Count

```
CREATE PROCEDURE [Post_Count]
AS
SELECT COUNT(PostID) FROM [Post]
```

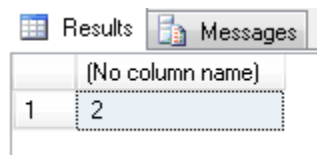
DataProvider.cs

```
public abstract int PostCount();
```

SqlDataProvider.cs

```
public override int PostCount()
{
    using (SqlConnection cnn = GetSqlConnection())
    {
        SqlCommand cmd = cnn.CreateCommand();
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "Post_Count";
        cnn.Open();
        object rs = cmd.ExecuteScalar();
        return Convert.ToInt32(rs);
    }
}
```

Khi truy vấn SP Post_Count thì bảng kết quả trả về như sau, chỉ có 1 hàng và 1 cột. Để lấy được ô dữ liệu đó 1 cách nhanh chóng ngoài cách dùng DataReader thì bạn dùng phương thức **ExecuteScalar**, sau đó ép về kiểu mà bạn muốn.



(No column name)	
1	2

Thủ tục và các hàm bổ sung Post_All

SP Post_All

```
CREATE PROCEDURE [Post_All]
AS
SET NOCOUNT ON -- sv tu tìm hiểu tác dụng của lệnh này
SELECT * FROM [Post] ORDER BY [PostID] DESC
```



DataProvider.cs

```
public abstract List<Post> PostAll();
```

SqlDataProvider.cs

```
public override List<Post> PostAll()
{
    using (SqlConnection cnn = GetSqlConnection())
    {
        SqlCommand cmd = cnn.CreateCommand();
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "Post_All";
        cnn.Open();
        // Tao doi tuong SqlDataReader de doc du lieu tuan tu tu csdl
        using (SqlDataReader reader =
            cmd.ExecuteReader(CommandBehavior.CloseConnection))
        {
            // Tao mot danh sach de chua du lieu doc duoc
            List<Post> list = new List<Post>();

            // Do kq tra ve la mot ds nhieu record nen phai dung
            // vong lap while de doc tung reader
            // reader.Read() => doc mot record tu csdl
            while (reader.Read())
            {
                // Trich du lieu tu 1 record va day vao doi tuong Post
                Post p = new Post();

                // reader[ten cot] => doc gia tri cua mot o^ du lieu
                // Kieu du lieu tra ve la object => ep ve kieu du lieu tuong ung voi
                // tung property cua doi tuong
                p.PostID = Convert.ToInt32(reader["PostID"]);
                p.Title = reader["Title"].ToString();
                p.Body = reader["Body"].ToString();
                // Do cot Publish duoc phep null nenkiem tra truoc khi ep kieu du lieu
                // DBNull.Value: gia tri NULL trong database
                if (reader["Publish"] != DBNull.Value)
                    p.Publish = DateTime.Parse(reader["Publish"].ToString());
                list.Add(p);
            }
            return list;
        }
    }
}
```

Thủ tục và hàm bổ sung Post_Single

SP Post_Single

```
CREATE PROCEDURE [dbo].[Post_Single]
    @PostID int
AS
```



```
SET NOCOUNT ON
SELECT * FROM [Post] WHERE [PostID]=@PostID
```

DataProvider.cs

```
public abstract Post PostSingle(int postId);
```

SqlDataProvider.cs

```
public override Post PostSingle(int postId)
{
    using (SqlConnection cnn = GetSqlConnection())
    {
        SqlCommand cmd = cnn.CreateCommand();
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.CommandText = "Post_Single";
        cmd.Parameters.Add("@PostID", SqlDbType.Int).Value = postId;
        cnn.Open();
        using (SqlDataReader reader =
            cmd.ExecuteReader(CommandBehavior.CloseConnection))
        {
            // kết quả chỉ là 1 record nên không dùng vòng while

            reader.Read();

            Post p = new Post();
            p.PostID = Convert.ToInt32(reader["PostID"]);
            p.Title = reader["Title"].ToString();
            p.Body = reader["Body"].ToString();
            if (reader["Publish"] != DBNull.Value)
                p.Publish = DateTime.Parse(reader["Publish"].ToString());

            return p;
        }
    }
}
```

Thủ tục và hàm bổ sung Post_Find

SP Post_Find

```
CREATE PROCEDURE [dbo].[Post_Find]
    @Title nvarchar(50)
AS
SET NOCOUNT ON
SELECT * FROM [Post] WHERE [Title] LIKE N'%' + @Title + '%'
```

DataProvider.cs

```
public abstract List<Post> PostFind(string title);
```

SqlDataProvider.cs



```
public override List<Post> PostFind(string title)
{
    // sinh vien tu lam
}
```

Nhân xét:

Các phương thức trong lớp SqlDataProvider sẽ được ứng dụng gọi rất nhiều lần. Để tránh việc tạo đi tạo lại các đối tượng SqlParameter, thì chúng ta sẽ cache (lưu vào bộ nhớ) để sau này có thể dùng lại.

Cách làm tham khảo trong cuốn ebook **ScaleNet** trang 543 mục **Cache Stored Procedure SqlParameter Objects** hoặc sử dụng lớp **SqlHelper**

Sử dụng lớp SqlHelper để tối ưu lớp SqlDataProvider

Copy tập tin **Files/ DataTools/SQLHelper.cs** vào project DataAccess.

Tạo 1 class SqlDataProviderV2 cho kế thừa từ lớp DataProvider và bổ sung phần thân cho các phương thức được kế thừa. Chèn thêm namespace **Microsoft.ApplicationBlocks.Data** để sử dụng lớp SqlHelper

Đối với phương thức PostUpdate, ta thêm như sau:

```
public override int PostUpdate(Post post)
{
    return SqlHelper.ExecuteNonQuery(_ConnectionString, "Post_Update",
        post.PostID, post.Title, post.Body, post.Publish);
}
```

Giải thích:

Phương thức PostUpdate ở trên dùng để gọi SP Post_Update. Ý nghĩa lần lượt của các tham số truyền vào như sau

_ConnectionString: chuỗi kết nối

"Post_Update": tên của SP muốn gọi

post.PostID, post.Title, post.Body, post.Publish: ở cách làm trước thì đây chính là giá trị của các tham số mà bạn muốn truyền vào 4 tham số theo thứ tự là: @PostID, @Title, @Body, @Publish

```
SqlCommand cmd = cnn.CreateCommand();
cmd.CommandType = CommandType.StoredProcedure;
cmd.CommandText = "Post_Update";
cmd.Parameters.Add("@PostID", SqlDbType.Int).Value = post.PostID;
```



```

cmd.Parameters.Add("@Title", SqlDbType.NVarChar, 50).Value = post.Title;
cmd.Parameters.Add("@Body", SqlDbType.NVarChar, 4000).Value = post.Body;
if (post.Publish.HasValue)
    cmd.Parameters.Add("@Publish", SqlDbType.DateTime)
        .Value = post.Publish.Value;
else
    cmd.Parameters.Add("@Publish", SqlDbType.DateTime)
        .Value = DBNull.Value;

```

Khi sử dụng SqlHelper thì bạn không cần phải viết chi tiết như trên mà chỉ cần truyền giá trị đúng với thứ tự các tham số trong SP, có nghĩa là

Giá trị post.PostID → giá trị của tham số @PostID

post.Title → giá trị của tham số @Title

....

Tương tự như cách làm trên, phần code bổ sung cho phương thức PostDelete như sau

```

public override int PostDelete(Post post)
{
    return SqlHelper.ExecuteNonQuery(_ConnectionString,
        "Post_Delete", post.PostID);
}

```

Do 2 thủ tục Update và Delete không có tham số nào là output nên cách viết trên không có vấn đề gì. Bạn tiếp tục bổ sung code cho phương thức PostAdd thì một sự cố nhỏ xảy ra là bạn muốn lấy giá trị output từ SP. Cách làm như sau:

```

// Gan tu dong cac gia tri cho cac doi tuong SqlParameter
private void AssignParameterValues(SqlParameter[] commandParameters, object[]
parameterValues)
{
    if ((commandParameters == null) || (parameterValues == null))
        return;

    if (commandParameters.Length != parameterValues.Length)
        throw new ArgumentException("Parameter count does not match Parameter Value
count.");

    for (int i = 0, j = commandParameters.Length; i < j; i++)
        commandParameters[i].Value = parameterValues[i];
}

public override int PostAdd(Post post)
{
    string spName = "Post_Add";
    // Lay cac tham so da luu trong bo nho
    SqlParameter[] parameters =
    SqlHelperParameterCache.GetSpParameterSet(_ConnectionString, spName);
    // Gan cac gia tri cho cac tham so do theo dung thu tu trong SP

```



```

AssignParameterValues(parameters,
    new object[] { post.PostID, post.Title, post.Body, post.Publish });
// Lay cac tham so da luu trong bo nho
int rs = SqlHelper.ExecuteNonQuery(_ConnectionString,
    CommandType.StoredProcedure, spName, parameters);

if (rs > 0)
{
    // Lay gia tri cua tham so @PostID
    foreach (SqlParameter p in parameters)
    {
        if (String.Compare(p.ParameterName, "@PostID", true) == 0)
            return (int)p.Value;
    }
}
return rs;
}

```

Phần code cho các phương thức còn lại

```

public override Post PostSingle(int postId)
{
    using (SqlDataReader reader = SqlHelper.ExecuteReader(_ConnectionString,
        "Post_Single", postId))
    {
        reader.Read();

        Post p = new Post();
        p.PostID = Convert.ToInt32(reader["PostID"]);
        p.Title = reader["Title"].ToString();
        p.Body = reader["Body"].ToString();
        if (reader["Publish"] != DBNull.Value)
            p.Publish = DateTime.Parse(reader["Publish"].ToString());
        return p;
    }
}

public override List<Post> PostFind(string title)
{
    using (SqlDataReader reader = SqlHelper.ExecuteReader(_ConnectionString,
        "Post_Find", title))
    {
        List<Post> list = new List<Post>();
        while (reader.Read())
        {
            Post p = new Post();
            p.PostID = Convert.ToInt32(reader["PostID"]);
            p.Title = reader["Title"].ToString();
            p.Body = reader["Body"].ToString();
            if (reader["Publish"] != DBNull.Value)
                p.Publish = DateTime.Parse(reader["Publish"].ToString());
            list.Add(p);
        }
    }
}

```



```

        return list;
    }
}

public override List<Post> PostAll()
{
    using (SqlDataReader reader = SqlHelper.ExecuteReader(_ConnectionString,
        "Post_All"))
    {
        List<Post> list = new List<Post>();
        while (reader.Read())
        {
            Post p = new Post();
            p.PostID = Convert.ToInt32(reader["PostID"]);
            p.Title = reader["Title"].ToString();
            p.Body = reader["Body"].ToString();
            if (reader["Publish"] != DBNull.Value)
                p.Publish = DateTime.Parse(reader["Publish"].ToString());
            list.Add(p);
        }
        return list;
    }
}

public override int PostCount()
{
    object rs = SqlHelper.ExecuteScalar(_ConnectionString, "Post_Count");
    return Convert.ToInt32(rs);
}

```

Sửa lại code trong lớp DataProvider như sau, sau đó chạy, kiểm tra kết quả và các bạn sẽ thấy nó vẫn chạy bình thường

```

public static DataProvider Instance
{
    get
    {
        if (_Instance == null)
            _Instance = new SqlDataProviderV2("ConnectionString");
        return _Instance;
    }
}

```

Nhận xét:

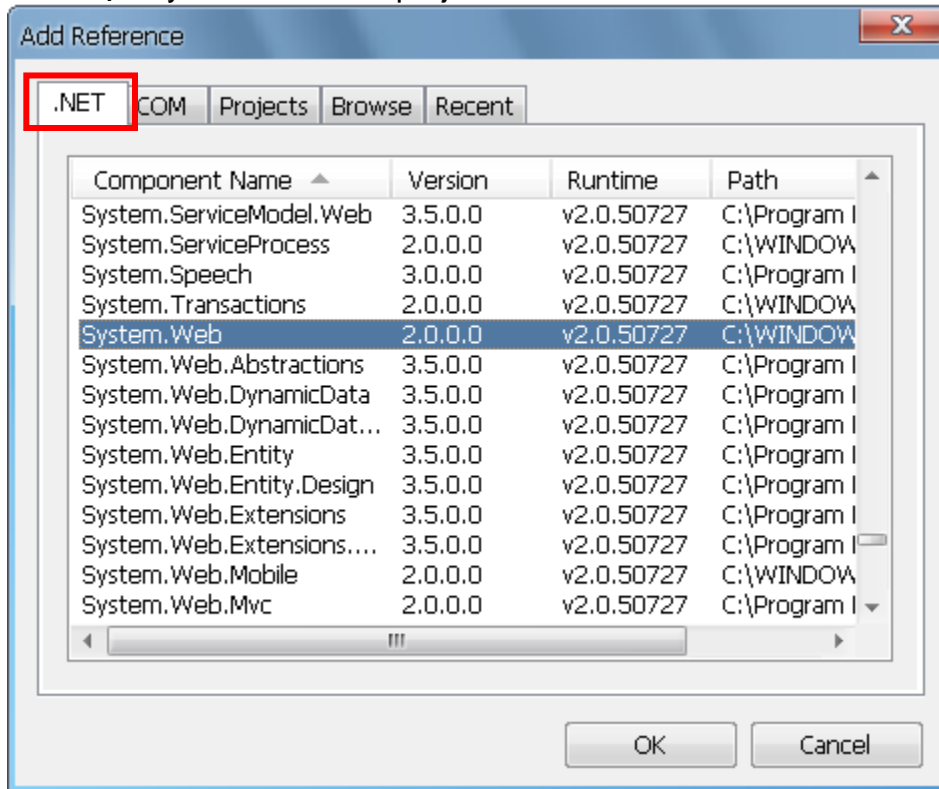
Đối với 3 phương thức Single, Find và All phần code tương đối giống nhau, chỉ khác ở phần `SqlHelper.ExecuteReader` (truyền tên thủ tục và các giá trị cho các tham số của thủ tục).

Phần code tô đậm chính là phần trích dữ liệu từ `DataReader` và chuyển vào đối tượng.



Sử dụng lớp CBO để tự động chuyển dữ liệu từ DataReader sang đối tượng

Chèn thêm thư viện System.Web vào project DataAccess



Copy 3 tập tin CBO.cs, Null.cs và DataCache.cs trong thư mục **Files/DataTools** vào project DataAccess

Sửa lại code của 3 phương thức Single, All, Find trong lớp SqlDataProviderV2 như sau (chèn thêm namespace **Core** để sử dụng lớp CBO)

```
public override Post PostSingle(int postId)
{
    return CBO.FillObject<Post>(SqlHelper.ExecuteReader(_ConnectionString,
"Post_Single", postId));
}

public override List<Post> PostFind(string title)
{
    return CBO.FillCollection<Post>(SqlHelper.ExecuteReader(_ConnectionString,
"Post_Find", title));
}

public override List<Post> PostAll()
{
    return CBO.FillCollection<Post>(SqlHelper.ExecuteReader(_ConnectionString,
"Post_All"));
}
```



Cách sử dụng lớp CBO

```
public override Post PostSingle(int postId)
{
    return CBO.FillObject<Post>(SqlHelper.ExecuteReader(_ConnectionString,
"Post_Single", postId));
}
```

Phương thức **FillObject<kiểu dữ liệu đối tượng>(DataReader)**: lấy dữ liệu từ DataReader và trả về một đối tượng

```
public override List<Post> PostAll()
{
    return CBO.FillCollection<Post>(SqlHelper.ExecuteReader(_ConnectionString,
"Post_All"));
}
```

Phương thức **FillCollection< kiểu dữ liệu đối tượng>(DataReader)**: lấy dữ liệu từ DataReader và trả về một danh sách kiểu List

Toàn bộ code của lớp SqlDataProviderV2 được viết như sau:

```
using System;
using System.Collections.Generic;
using System.Configuration;
using Microsoft.ApplicationBlocks.Data; // namespace của lớp SqlHelper
using System.Data;
using System.Data.SqlClient;
using Core; // namespace của lớp CBO

namespace DataAccess
{
    public class SqlDataProviderV2 : DataProvider
    {
        private string _ConnectionString;
        public SqlDataProviderV2(string connectionStringName)
        {
            _ConnectionString =
ConfigurationManager.ConnectionStrings[connectionStringName].ConnectionString;
        }

        private void AssignParameterValues(SqlParameter[] commandParameters, object[]
parameterValues)
        {
            if ((commandParameters == null) || (parameterValues == null))
                return;

            if (commandParameters.Length != parameterValues.Length)
                throw new ArgumentException("Parameter count does not match Parameter
Value count.");
        }
    }
}
```



```

        for (int i = 0, j = commandParameters.Length; i < j; i++)
            commandParameters[i].Value = parameterValues[i];
    }

    public override Post PostSingle(int postId)
    {
        return CBO.FillObject<Post>(SqlHelper.ExecuteReader(_ConnectionString,
"Post_Single", postId));
    }

    public override List<Post> PostFind(string title)
    {
        return
CBO.FillCollection<Post>(SqlHelper.ExecuteReader(_ConnectionString, "Post_Find",
title));
    }

    public override List<Post> PostAll()
    {
        return
CBO.FillCollection<Post>(SqlHelper.ExecuteReader(_ConnectionString, "Post_All"));
    }

    public override int PostAdd(Post post)
    {
        string spName = "Post_Add";
        // Lay cac tham so tu bo nho
        SqlParameter[] parameters =
SqlHelperParameterCache.GetSpParameterSet(_ConnectionString, spName);
        AssignParameterValues(parameters,
            new object[] { post.PostID, post.Title, post.Body, post.Publish });
        int rs = SqlHelper.ExecuteNonQuery(_ConnectionString,
CommandType.StoredProcedure, spName, parameters);
        if (rs > 0)
        {
            foreach (SqlParameter p in parameters)
            {
                if (String.Compare(p.ParameterName, "@PostID", true) == 0)
                    return (int)p.Value;
            }
        }
        return rs;
    }

    public override int PostUpdate(Post post)
    {
        return SqlHelper.ExecuteNonQuery(_ConnectionString, "Post_Update",
post.PostID, post.Title, post.Body, post.Publish);
    }

    public override int PostDelete(Post post)
    {

```



```

        return SqlHelper.ExecuteNonQuery(_ConnectionString, "Post_Delete",
post.PostID);
    }
}

```

Nhận xét:

- Các phương thức Update, Delete tương đối giống nhau chỉ khác phần spName và các giá trị tham số
- Các phương thức Find, All, Single tương đối giống nhau chỉ khác phần spName và các giá trị tham số

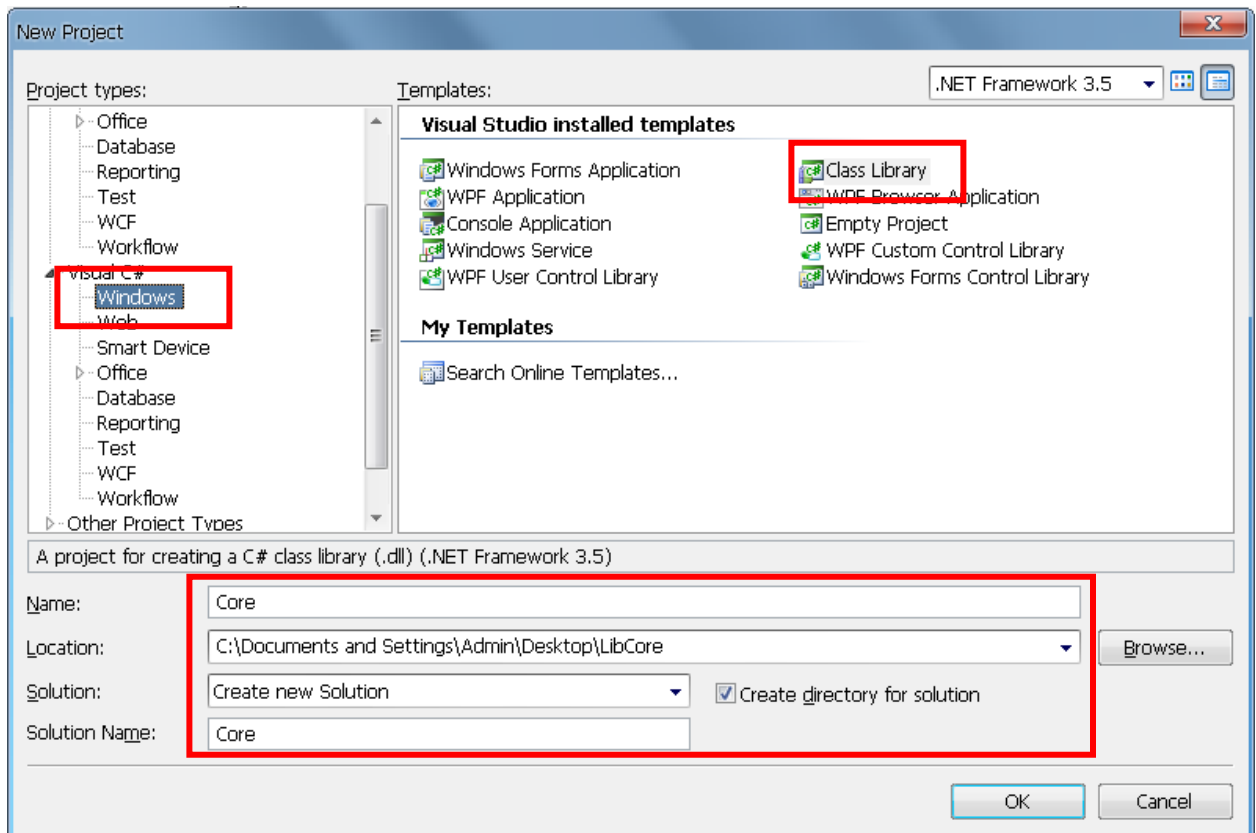
Đặt vấn đề:

Giả sử bạn có thêm một bảng nữa trong CSDL, thì các thao tác đọc/ghi cũng tương tự như các trên. Bạn chỉ cần “cóp dán” và sửa lại những phần sai khác. Nhưng bạn phải làm gì khi bạn muốn xây dựng một ứng dụng khác với CSDL hoàn toàn khác. Bạn sẽ phải viết lại hoặc copy và sửa lại một số phần trong project DataAccess để phù hợp với project mới → mất thời gian. Do đó để tiết kiệm thời gian xây dựng ứng dụng, bạn nên xây dựng một thư viện bao gồm các lớp thực hiện thao tác đọc, ghi dữ liệu ở mức tổng quát (tạm đặt là Core) có thể dùng cho nhiều project khác nhau.

Xây dựng thư viện Core ở mức tổng quát

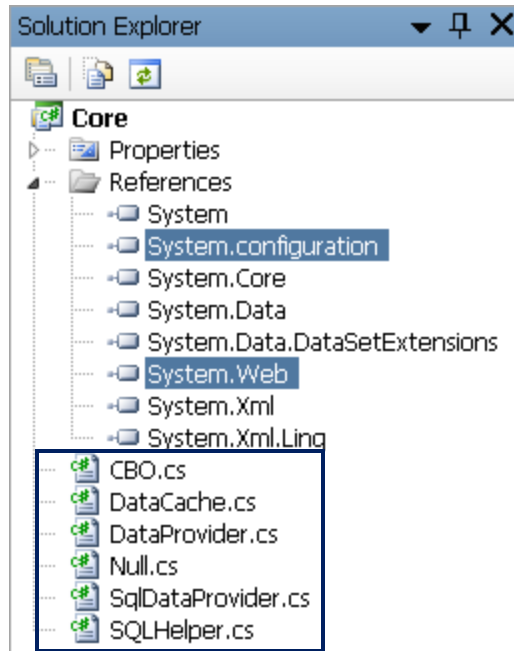
1. Tạo thư mục ở Desktop và đặt tên là LibCore
2. Mở Microsoft Visual Studio (VS), tạo một project thư viện (Class Library Project) và đặt tên là **Core**





3. Bấm chuột phải References chọn Add References, sau đó vào thẻ .NET để thêm 2 thư viện sau
 - System.Configuration
 - System.Web
4. Xóa Class1 và copy 4 file sau vào project Core
 - SqlHelper.cs
 - CBO.cs
 - Null.cs
 - DataCache.cs
5. Thêm 2 class là DataProvider và SqlDataProvider vào project
6. Solution của bạn sẽ có 2 thư viện và các tập tin như hình sau:





Bổ sung code cho lớp DataProvider

```
using System;
using System.Data;

namespace Core
{
    public abstract class DataProvider
    {
        public abstract object ExecuteNonQueryWithOutput(string outputParam, string
spName, params object[] parameterValues);

        public abstract int ExecuteNonQuery(string spName, params object[]
parameterValues);

        public abstract DataSet ExecuteDataset(string spName, params object[]
parameterValues);

        public abstract IDataReader ExecuteReader(string spName, params object[]
parameterValues);

        public abstract object ExecuteScalar(string spName, params object[]
parameterValues);
    }
}
```

Bổ sung code cho lớp SqlDataProvider

```
using System;
using System.Configuration;
using Microsoft.ApplicationBlocks.Data;
using System.Data;
```



```

using System.Data.SqlClient;

namespace Core
{
    public class SqlDataProvider : DataProvider
    {
        private string connectionString;

        public SqlDataProvider(string connectionStringName)
        {
            this.connectionString =
            ConfigurationManager.ConnectionStrings[connectionStringName].ConnectionString;
        }

        private void AssignParameterValues(SqlParameter[] commandParameters, object[]
parameterValues)
        {
            if ((commandParameters == null) || (parameterValues == null))
                return;

            if (commandParameters.Length != parameterValues.Length)
                throw new ArgumentException("Parameter count does not match Parameter
Value count.");
            for (int i = 0, j = commandParameters.Length; i < j; i++)
                commandParameters[i].Value = parameterValues[i];
        }

        public override int ExecuteNonQuery(string spName, params object[]
parameterValues)
        {
            return SqlHelper.ExecuteNonQuery(connectionString, spName,
parameterValues);
        }

        public override object ExecuteNonQueryWithOutput(string outputParam, string
spName, params object[] parameterValues)
        {
            if (string.IsNullOrEmpty(outputParam))
                throw new ArgumentException("OutputParam is null or empty");

            SqlParameter[] parameters =
            SqlHelperParameterCache.GetSpParameterSet(connectionString, spName);
            SqlParameter sqlParameter = null;
            foreach (SqlParameter item in parameters)
            {
                if (String.Compare(item.ParameterName, outputParam, true) == 0)
                {
                    sqlParameter = item;
                    break;
                }
            }
            if (sqlParameter == null)
                throw new Exception("Parameter not found");

            AssignParameterValues(parameters, parameterValues);
        }
    }
}

```



```

        int rs = SqlHelper.ExecuteNonQuery(connectionString,
CommandType.StoredProcedure, spName, parameters);
        if (rs > 0)
            return sqlParameter.Value;
        return null;
    }

    public override IDataReader ExecuteReader(string spName, params object[]
parameterValues)
    {
        return SqlHelper.ExecuteReader(connectionString, spName,
parameterValues);
    }

    public override DataSet ExecuteDataset(string spName, params object[]
parameterValues)
    {
        return SqlHelper.ExecuteDataset(connectionString, spName,
parameterValues);
    }

    public override object ExecuteScalar(string spName, params object[]
parameterValues)
    {
        return SqlHelper.ExecuteScalar(connectionString, spName,
parameterValues);
    }
}

```

Bổ sung tiếp code cho lớp DataProvider

```

private static DataProvider instance = null;
public static DataProvider Instance
{
    get
    {
        if (instance == null)
            instance = new SqlDataProvider("ConnectionString");
        return instance;
    }
}

```

Toàn bộ code của lớp DataProvider như sau

```

using System;
using System.Data;

namespace Core
{
    public abstract class DataProvider
    {
        private static DataProvider instance = null;
        public static DataProvider Instance

```




```
{
    get
    {
        if (instance == null)
            instance = new SqlDataProvider("ConnectionString");
        return instance;
    }
}

public abstract object ExecuteNonQueryWithOutput(string outputParam, string
spName, params object[] parameterValues);

public abstract int ExecuteNonQuery(string spName, params object[]
parameterValues);

public abstract DataSet ExecuteDataset(string spName, params object[]
parameterValues);

public abstract IDataReader ExecuteReader(string spName, params object[]
parameterValues);

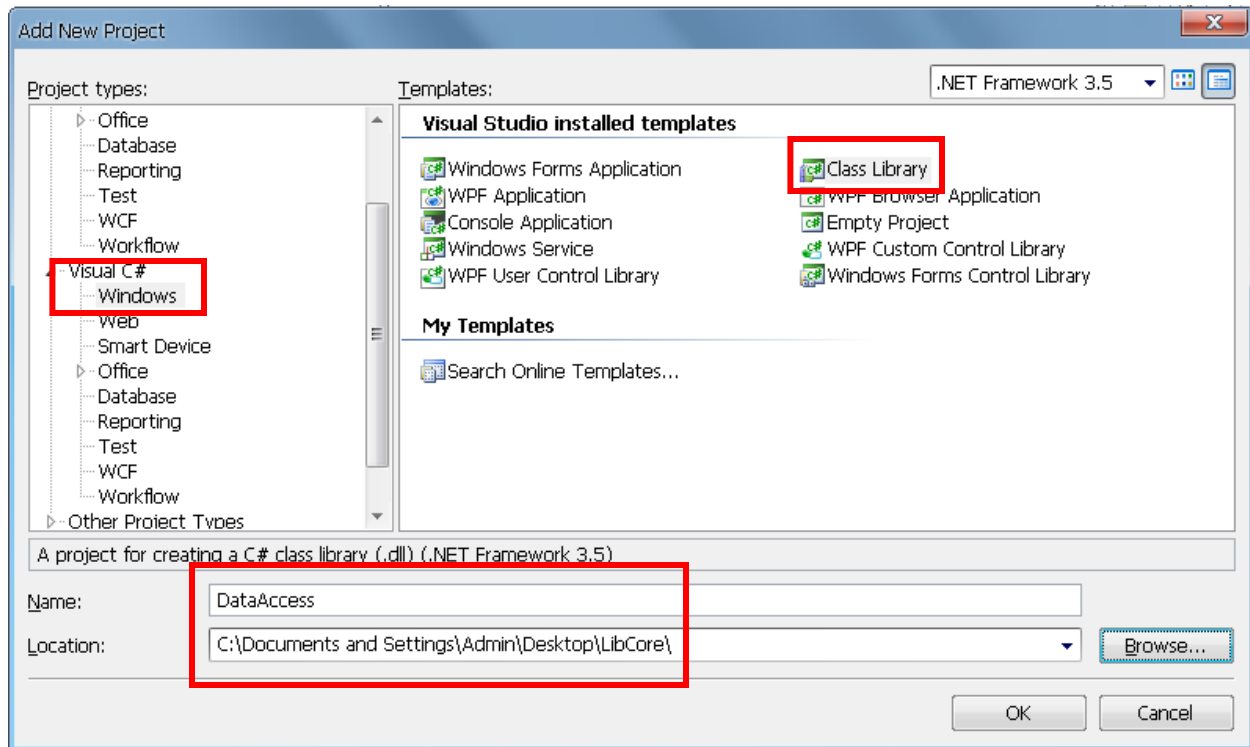
public abstract object ExecuteScalar(string spName, params object[]
parameterValues);
}
```

Bấm Ctrl + Shift + B để build project, sau đó vào thư mục bin để lấy file thư viện Core.dll

Xây dựng thư viện DataAccess sử dụng thư viện Core.dll

1. Vào menu File > Add > New Project vào tạo một project thư viện (Class Library Project) và đặt tên là **DataAccess**





2. Xóa Class1 và thêm thư viện Core vào project DataAccess.

3. Tạo lớp Post như sau

```
using System;
using System.Collections.Generic;
using Core;

namespace DataAccess
{
    public class Post
    {
        public int PostID { get; set; }
        public string Title { get; set; }
        public string Body { get; set; }
        public DateTime? Publish { get; set; }

        public Post() { }

        public static List<Post> All()
        {
            return CB0.FillCollection<Post>(
                DataProvider.Instance.ExecuteReader("Post_All"));
        }

        public static List<Post> Find(string title)
        {
            return CB0.FillCollection<Post>(
                DataProvider.Instance.ExecuteReader("Post_Find", title));
        }
    }
}
```

dnc.dlu@gmail.com



```

    }

    public static Post Single(string postId)
    {
        try
        {
            return CBO.FillObject<Post>(
                DataProvider.Instance.ExecuteReader("Post_Single",
                    Convert.ToInt32(postId)));
        }
        catch (Exception)
        {
            return null;
        }
    }

    public static int Count()
    {
        object rs = DataProvider.Instance.ExecuteScalar("Post_Count");
        return Convert.ToInt32(rs);
    }

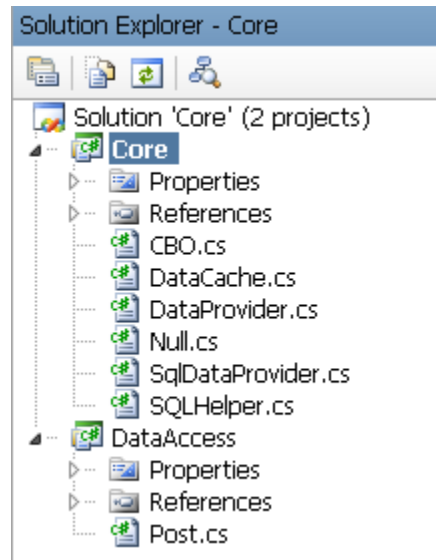
    public static int Add(Post data)
    {
        object rs =
        DataProvider.Instance.ExecuteNonQueryWithOutput("@PostID", "Post_Add",
            data.PostID, data.Title, data.Body, data.Publish);
        int identity = rs != null ? Convert.ToInt32(rs) : 0;
        return identity;
    }

    public static bool Update(Post data)
    {
        int rs = DataProvider.Instance.ExecuteNonQuery("Post_Update",
            data.PostID, data.Title, data.Body, data.Publish);
        return rs > 0;
    }

    public static bool Delete(string postId)
    {
        try
        {
            int rs = DataProvider.Instance.ExecuteNonQuery("Post_Delete",
                Convert.ToInt32(postId));
            return rs > 0;
        }
        catch (Exception)
        {
            return false;
        }
    }
}
}
}

```





4. Tạo tiếp project Console và test như ví dụ trên

Luyện tập

1. Tạo bảng như sau, thiết lập cột Id có giá trị id tự tăng, đặt tên là Movie

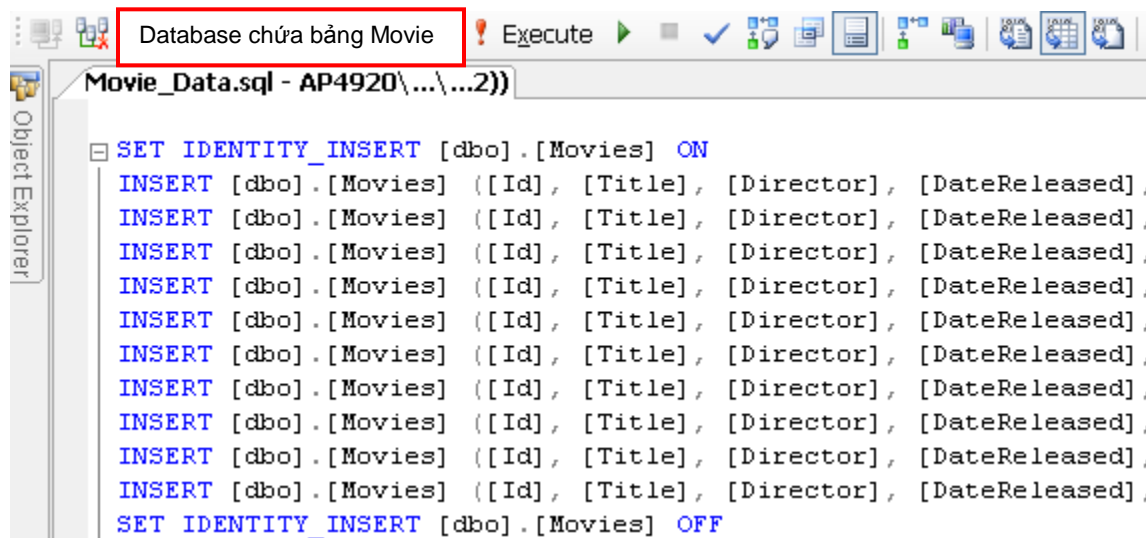
	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Title	nvarchar(100)	<input type="checkbox"/>
	Director	nvarchar(100)	<input type="checkbox"/>
	DateReleased	datetime	<input type="checkbox"/>
	InTheaters	bit	<input type="checkbox"/>
	BoxOfficeTotals	money	<input type="checkbox"/>
	Description	nvarchar(4000)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Column Properties	
	Identity Specification
	(Is Identity)
	Identity Increment
	Identity Seed

2. Thêm dữ liệu cho bảng **Movie**



Mở file **SrcriptSQL/Movie_Data.sql**, chọn database chứa bảng Movie sau đó bấm **Execute**



3. Tạo thư viện Data Access như phần trước để gọi các thủ tục sau (tạo tiếp lớp Movie và bổ sung các phương thức vào project ở trên). Sau mỗi phương thức phải test kết quả trong project Console

- Movie_All
- Movie_Single
- Movie_Find
- Movie_Add
- Movie_Update
- Movie_Delete
- Movie_Count
- **Movie_Paging**

Hướng dẫn làm thủ tục Movie_Paging

1. Bấm vào **New Query**, gõ lần lượt các câu truy vấn sau và xem kết quả



Câu 1: thêm một cột số thứ tự vào bảng kết quả (hàm **ROW_NUMBER()** chỉ sử dụng ở các bản SQL Server 2005 trở lên)

```

SELECT ROW_NUMBER() OVER (ORDER BY Id) AS Row, Title
FROM Movie
    
```



	Row	Title
1	1	Titanic
2	2	Star Wars II
3	3	Jurassic Park
4	4	Jaws
5	5	Ghost
6	6	Forrest Gump
7	7	Ice Age
8	8	Shrek
9	9	Independence Day
10	10	The Ring

Câu 2: lấy các kết quả từ hàng thứ 6 → 10

```
SELECT Row, Title
FROM
(
    SELECT ROW_NUMBER() OVER (ORDER BY Id) AS Row, Title
    FROM Movie
) AS MoviesWithRowNumbers
WHERE Row >= 6 AND Row <= 10
```

	Row	Title
1	6	Forrest Gump
2	7	Ice Age
3	8	Shrek
4	9	Independence Day
5	10	The Ring

Câu 3: tạo một bảng tạm (@Movies) có kèm theo cột thứ tự để lưu dữ liệu tạm vào đó, sau đó lấy các hàng từ 6 → 10 trong bảng tạm → cách làm này gọi là phân trang (Paging)

Sử dụng SQL 2005 trở đi

```
-- declare a new TABLE variable
DECLARE @Movies TABLE
(
    RowNumber INT,
    Id INT,
    Title NVARCHAR(100)
)

-- populate the table variable with the complete list of products
INSERT INTO @Movies
```



```
SELECT ROW_NUMBER() OVER (ORDER BY Movie.Id) AS Row,
       Id, Title
FROM Movie
```

```
-- extract the requested page of products
SELECT * FROM @Movies
WHERE RowNumber >= 6 AND RowNumber <= 10
```

Sử dụng SQL Server 2000

```
DECLARE @Movies TABLE
(
    RowNumber INT IDENTITY (1,1),
    Id INT,
    Title NVARCHAR(100)
)

-- populate the table variable with the complete list of products
INSERT INTO @Movies
SELECT Id, Title
FROM Movie

-- extract the requested page of products
SELECT * FROM @Movies
WHERE RowNumber >= 6 AND RowNumber <= 10
```

Kết quả từ câu truy vấn 3 (chỉ lấy 3 cột)

	RowNumber	Id	Title
1	6	7	Forrest Gump
2	7	8	Ice Age
3	8	9	Shrek
4	9	10	Independence Day
5	10	22	The Ring

Tương tự như trên, viết thêm vào câu truy vấn 3 để lấy đầy đủ các cột của bảng Movie

Câu 4: Lấy dữ liệu từ dòng 1 → 5 (trang 1, số record tối đa trong 1 trang là 5)

```
DECLARE @PageNumber INT
DECLARE @PageSize INT

SET @PageNumber = 1 -- trang hiện tại
SET @PageSize = 5 -- số record tối đa của một trang

-- tạo bảng tạm để chứa dữ liệu
DECLARE @Movies TABLE
(
    RowNumber INT,
```



```

        Id INT,
        Title NVARCHAR(100),
        Director NVARCHAR(100),
        DateReleased DATETIME,
        InTheaters BIT,
        BoxOfficeTotals MONEY,
        Description NVARCHAR(4000)
    )

    -- dua du lieu vao bang movie kem theo cot stt
    INSERT INTO @Movies
    SELECT ROW_NUMBER() OVER (ORDER BY Movie.Id) AS Row,
           Id, Title, Director, DateReleased, InTheaters, BoxOfficeTotals,
           Description
    FROM Movie

    -- tong so record cua bang tam @Movies
    SELECT COUNT(Id) FROM @Movies

    -- lay cac record can thiet
    SELECT * FROM @Movies
    WHERE RowNumber > (@PageNumber - 1) * @PageSize
           AND RowNumber <= @PageNumber * @PageSize
    
```

Sử dụng SQL Server 2000

```

    DECLARE @PageNumber INT
    DECLARE @PageSize INT

    SET @PageNumber = 1 -- trang hien tai
    SET @PageSize = 5 -- so record toi da cua mot trang

    -- tao bang tam de chua du lieu
    DECLARE @Movies TABLE
    (
        RowNumber INT IDENTITY (1,1),
        Id INT,
        Title NVARCHAR(100),
        Director NVARCHAR(100),
        DateReleased DATETIME,
        InTheaters BIT,
        BoxOfficeTotals MONEY,
        Description NVARCHAR(4000)
    )

    -- dua du lieu vao bang movie kem theo cot stt
    INSERT INTO @Movies
    SELECT Id, Title, Director, DateReleased, InTheaters, BoxOfficeTotals,
           Description
    FROM Movie

    -- tong so record cua bang tam @Movies
    SELECT COUNT(Id) FROM @Movies
    
```




```
-- lay cac record can thiet
SELECT * FROM @Movies
WHERE RowNumber > (@PageNumber - 1) * @PageSize
AND RowNumber <= @PageNumber * @PageSize
```

Kết quả truy vấn như sau

Results		Messages						
		{No column name}						
1	10							
	RowNumber	Id	Title	Director	DateReleased	InTheaters	BoxOfficeTotals	De
1	1	1	Titanic	James Cameron	1997-06-21 00:00:00.000	0	600000000.00	Or
2	2	2	Star Wars II	George Lucas	1977-06-01 00:00:00.000	1	500000000.00	Or
3	3	3	Jurassic Park	Steven Spielberg	1993-06-17 00:00:00.000	1	400000000.00	Or
4	4	4	Jaws	Steven Spielberg	1975-05-30 00:00:00.000	0	300000000.00	Or
5	5	5	Ghost	Jerry Zucker	1990-06-14 00:00:00.000	0	200000000.00	Or

- Bảng 1: tổng số record. Từ tổng số record này, bạn có thể tính được tổng số trang theo công thức

Số Trang = Làm tròn lên(TổngRecord / PageSize)

- Bảng 2: phần dữ liệu của trang 1

Câu 5: Tương tự thay @PageNumber = 2 và xem kết quả

```
SET @PageNumber = 2 -- trang hiện tại
```

Áp dụng viết thủ tục **Movie_Paging** như sau

```
CREATE PROCEDURE Movie_Paging
    @PageNumber INT,
    @PageSize INT
AS
-- tao bang tam de chua du lieu
DECLARE @Movies TABLE
(
    RowNumber INT,
    Id INT,
    Title NVARCHAR(100),
    Director NVARCHAR(100),
    DateReleased DATETIME,
    InTheaters BIT,
    BoxOfficeTotals MONEY,
    Description NVARCHAR(4000)
)

-- dua du lieu vao bang movie kem theo cot stt
INSERT INTO @Movies
SELECT ROW_NUMBER() OVER (ORDER BY Movie.Id) AS Row,
```



```

        Id, Title, Director, DateReleased, InTheaters, BoxOfficeTotals,
        Description
    FROM Movie

-- tong so record cua bang tam @Movies
SELECT COUNT(Id) FROM @Movies

-- lay cac record can thiet
SELECT * FROM @Movies
WHERE RowNumber > (@PageNumber - 1) * @PageSize
    AND RowNumber <= @PageNumber * @PageSize

```

SQL Server 2000

```

CREATE PROCEDURE Movie_Paging
    @PageNumber INT,
    @PageSize INT
AS
-- tao bang tam de chua du lieu
DECLARE @Movies TABLE
(
    RowNumber INT IDENTITY (1,1),
    Id INT,
    Title NVARCHAR(100),
    Director NVARCHAR(100),
    DateReleased DATETIME,
    InTheaters BIT,
    BoxOfficeTotals MONEY,
    Description NVARCHAR(4000)
)

-- dua du lieu vao bang movie kem theo cot stt
INSERT INTO @Movies
SELECT Id, Title, Director, DateReleased, InTheaters, BoxOfficeTotals,
        Description
FROM Movie

-- tong so record cua bang tam @Movies
SELECT COUNT(Id) FROM @Movies

-- lay cac record can thiet
SELECT * FROM @Movies
WHERE RowNumber > (@PageNumber - 1) * @PageSize
    AND RowNumber <= @PageNumber * @PageSize

```

Viết phương thức Paging trong lớp Movie như sau

```

public static List<Movie> Paging(int page, int pageSize, out int howManyPages)
{
    IDataReader reader = null;
    try
    {

```



```
reader = DataProvider.Instance.ExecuteReader(
    "Movie_Paging", page, pageSize);

//lay du lieu tu bang 1
reader.Read();
//tinh so trang
howManyPages = (int)Math.Ceiling((double)reader.GetInt32(0) /
(double)pageSize);
//lay du lieu tu bang 2
reader.NextResult();
return CBO.FillCollection<Movie>(reader);
}
catch
{
    // dam bao ket noi duoc dong neu co loi xay ra
    if (reader != null && reader.IsClosed == false)
        reader.Close();
    // so trang = 0
    howManyPages = 0;
    // tra ve ket qua la mot ds rong
    return new List<Movie>();
}
}
```

