مذكرة استخدام الاجراءات المخزنة في سي شارب بخطوات سهلة وميسرة وبالمثال العملي البسيط

باستخدام سيكوال سيرفر ٢٠١٧ + فيجول ستديو ٢٠١٧

يناير ٢٠١٩

تأليف: أ. محمد صبحي الحلو

غزة – فلسطين
٢٠١٩

Un_Moh@yhaoo.co.uk

استدعاء الاجراء المخزن في قاعدة البيانات

An introduction to executing SQL Server stored procedures and how to retrieve Data.

مقدمـــة:

الإجراءات المخزنة هي مجموعة من أزامر SQL التي تنفذ و تخزن بداخل قاعدة البيانات، كل مرة نقوم بها بتنفيذ جملة أو أمر SQL ، يتم تمرير الامر و اجراء التحسين و الملائمة و من ثم التنفيذ، و تكرار هذه العملية كل مرة تنفذ فيها الامر هو أمر مكلف و مستنفذ لمصادر الجهاز ، و لحل هذه المشكلة لدينا مجموعة من الأوامر تنفذ و تسمي "الاجراء المخزن" و التي هي تم تمريرها و فحصها و تنفيذها مسبقاً في قاعدة البيانات، و في هذا المقال سنتحدث عن طريقة استدعائها من خلال ADO.net و كيف سنتعامل مع المخرجات من الاجراء المخزن بعد استدعائه.

١- في البداية نقوم بإنشاء كائن (SqlConnection) و هو موجود ضمن النطاق **(System.Data.SqlClient namespace) ، وهكذا نكون** قد وفرنا نص الاتصال كمتغير تمرير (parameter) يحتوي القيم
(Data Source name, database name , authentication credentials) و هي القيم اللازمة لتأسيس الاتصال بقاعدة البيانات ثم نقوم بفتح الاتصال كما يلي.

```
SqlConnection con = new SqlConnection("Data Source=  ;
          initial catalog= Northwind ;
            User Id=  ; Password=  '");

con.open();
```

٢- قم بإنشاء الاجراء المخزن التالي على الجدول المطلوب في قاعدة البيانات (Region table in the Northwind database) و الذي يقبل متغيرين و لا يقوم بإرجاع أي متغيرات.

```
CREATE PROCEDURE RegionUpdate (@RegionID INTEGER,
@RegionDescription NCHAR(50)) AS
SET NOCOUNT OFF
UPDATE Region
SET RegionDescription = @RegionDescription
```

٣- قم بإنشاء كائن أمر (SqlCommand) بمتغير نصي هو اسم الاجراء المخزن المنوي استدعائه و تنفيذه و اسم كائن الاتصال (con) لتعريف أي أمر (command) سوف يتم ارساله للتنفيذ.

```
SqlCommand command = new
SqlCommand("RegionUpdate",con);
```

٤- قم بتغيير قيمة خاصية نوع الامر (CommandType) الي اجراء مخزن ("stored procedure").

```
command.CommandType = CommandType.StoredProcedure;
```

٥- قم بإضافة متغيرات لكائن الأمر (command) باستخدام مجموعة متغيرات مررة و كائن (SqlParameter class) يحتوي نوع الحقول التي سنتعامل معها.

```
command.Parameters.Add(new
SqlParameter("@RegionID",SqlDbType.Int,0,"RegionID"));

command.Parameters.Add(new
            SqlParameter("@RegionDescription",SqlDbType.NChar,
            50,"RegionDescription"));
```

٦- قم بتعريف قيم لاستخدام المتغيرات باستخدام خاصية (Value) التابعة للمتغيرات المررة للإجراء التنفيذي.

```
command.Parameters[0].Value=4;
command.Parameters[1].Value="SouthEast";
```

٧- قم بتنفيذ الاجراء المخزن باستخدام خاصية (ExecuteNonQuery) والتي تعيد عدد الصفوف التي سوف يطبق عليها الاجراء المخزن.

```
Int i=command.ExecuteNonQuery();
```

٨- الآن دعنا نري كيف ينفذ الاجراء المخزن – الذي لديه مخرجات – وكيف نصل للنتائج باستخدام متغيرات الإخراج (output parameters.).

٩- قم بإنشاء الاجراء المخزن التالي والذي له متغير إخراج واحد:

```
ALTER PROCEDURE RegionFind(@RegionDescription NCHAR(50) OUTPUT,
@RegionID INTEGER )

AS

SELECT @RegionDescription = RegionDescription from Region
where <A href="mailto:RegionID=@RegionID">RegionID=@RegionID</A>
```

الاجراء المخزنة السابق يقبل المتغير (RegionID) كمتغيرات ادخال ويبحث عن قيمة الحقل للخاصية المدخلة (RegionDescription) وارجاعها كمتغير اخراج.

```
SqlCommand command1 = new SqlCommand("RegionFind",con);
command1.CommandType = CommandType.StoredProcedure;
```

١٠- قم بإضافة متغيرات تم تمريرها للأمر(command1).

```
command1.Parameters.Add(new SqlParameter ("@RegionDescription",SqlDbType.NChar
,50,ParameterDirection.Output,false,0,50,"RegionDescription",DataRowVersion.Default,null));

command1.Parameters.Add(new SqlParameter("@RegionID" , SqlDbType.Int,0 , "RegionID" ));
```

لاحظ المتغير (RegionDescription) الذي أضيف مع (ParameterDirection) كمخرجات.

١١- قم بتعريف قيمة لمتغير الادخال (.RegionID) كما يلي.

```
command1.Parameters["@RegionID"].Value = 4;
```

١٢- قم بتعيين قيمة لخاصية (UpdatedRowSource) التابعة للكائن (SqlCommand) ل (UpdateRowSource.OutputParameters) لنتأكد من أن البيانات سوف يتم ارجاعها من الاجراء المخزن من خلال متغير الإخراج.

```
command1.UpdatedRowSource = UpdateRowSource.OutputParameters;
```

١٣- قم باستدعاء الاجراء المخزن وقم بالوصول الي الحقل (RegionDescription) الخاصة بالحقل (RegionID) باستخدام الخاصية (value) التابعة للمتغير.

```
command1.ExecuteNonQuery();
string newRegionDescription =(string)
command1.Parameters["@RegionDescription"].Value;
```

١٤- قم بإغلاق الاتصال.

```
con.Close();
```

و بنفس الطريقة تستطيع أن تستدعي الاجراء المخزن الذي يعيد مجموعة من الحقول بتعريف المتغيرات كما هو مناسب و تنفيذ الامر (command) باستخدام الخاصية (ExecuteReader()) التي تحول السجلات المعادة بالأمر(command).

التعامل مع الإجراءات المخزنة

قم بإنشاء قاعدة بيانات كالتالي:

RIX\SQL_EXPRESS.C....MDF - dbo.Person | RIX\SQL_EXPRESS.C...T

Column Name	Data Type	Allow Nulls
inc	int	☐
id	int	☐
Name	varchar(50)	☑
Memo	varchar(200)	☑
job_id	int	☐
		☐

RIX\SQL_EXPRESS.C...TA.MDF - dbo.Jobs

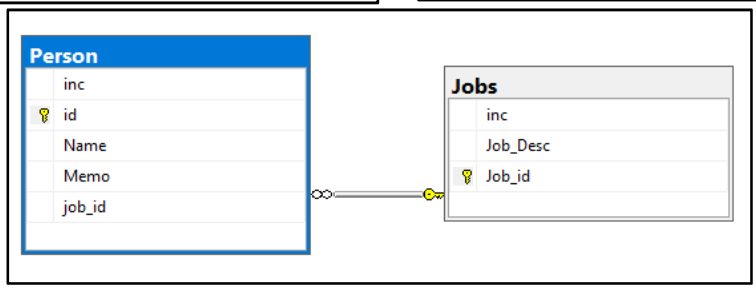
Column Name	Data Type	Allow Nulls
inc	int	☐
Job_Desc	nchar(10)	☑
Job_id	int	☐
		☐

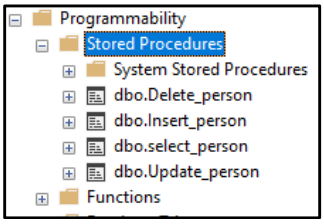
RIX\SQL_EXPRESS.E...TA.MDF - dbo.Jobs | RIX\SQL_EXPRES

inc	Job_Desc	Job_id
1	Manager	100
2	Employee	200
3	HandWroker	300
NULL	NULL	NULL

RIX\SQL_EXPRESS.E....MDF - dbo.Person | RIX\SQL_EXPRESS.E...A.MDF - Diagram_0*

inc	id	Name	Memo	job_id
1	1111	Moh	jhkjkjkjk	100
2	2222	sami	kjhjhkjkj	200
3	3333	shadi	jhklllll	300
8	444	yos	fdfdf	200
NULL	NULL	NULL	NULL	NULL






```
USE [master]
GO
/****** Object:  Database [C: \DATA.MDF]    Script Date: 01/12/2019 10:12:51 ص ******/
DROP DATABASE [C:\DATA.MDF]
GO

/****** Object:  Database [C:\DATA.MDF]    Script Date: 01/12/2019 10:12:51 ص ******/
CREATE DATABASE [C:\DATA.MDF]

 CONTAINMENT = NONE
 ON  PRIMARY

( NAME = N'Data', FILENAME = N'C:\Data.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )
 LOG ON
( NAME = N'Data_log', FILENAME = N'C:\ Data_log.ldf' , SIZE = 768KB , MAXSIZE = UNLIMITED, FILEGROWTH = 10%)

GO
ALTER DATABASE [C:\ DATA.MDF] SET COMPATIBILITY_LEVEL = 140
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [C:\ DATA.MDF].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
```

```
ALTER DATABASE [C:\DATA.MDF] SET ANSI_NULL_DEFAULT OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET ANSI_NULLS OFF   GO

ALTER DATABASE [C:\DATA.MDF] SET ANSI_PADDING OFF    GO

ALTER DATABASE [C:\DATA.MDF] SET ANSI_WARNINGS OFF   GO

ALTER DATABASE [C:\DATA.MDF] SET ARITHABORT OFF   GO

ALTER DATABASE [C:\DATA.MDF] SET AUTO_CLOSE ON   GO

ALTER DATABASE [C:\DATA.MDF] SET AUTO_SHRINK OFF   GO

ALTER DATABASE [C:\DATA.MDF] SET AUTO_UPDATE_STATISTICS ON   GO

ALTER DATABASE [C:\DATA.MDF] SET CURSOR_CLOSE_ON_COMMIT OFF   GO

ALTER DATABASE [C:\DATA.MDF] SET CURSOR_DEFAULT  GLOBAL   GO

ALTER DATABASE [C:\DATA.MDF ] SET CONCAT_NULL_YIELDS_NULL OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET NUMERIC_ROUNDABORT OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET QUOTED_IDENTIFIER OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET RECURSIVE_TRIGGERS OFF GO

ALTER DATABASE [C:\DATA.MDF] SET  DISABLE_BROKER  GO

ALTER DATABASE [C:\DATA.MDF] SET AUTO_UPDATE_STATISTICS_ASYNC OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET DATE_CORRELATION_OPTIMIZATION OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET TRUSTWORTHY OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET ALLOW_SNAPSHOT_ISOLATION OFF GO

ALTER DATABASE [C:\DATA.MDF] SET PARAMETERIZATION SIMPLE  GO

ALTER DATABASE [C:\DATA.MDF] SET HONOR_BROKER_PRIORITY OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET RECOVERY SIMPLE  GO

ALTER DATABASE [C:\DATA.MDF] SET  MULTI_USER  GO

ALTER DATABASE [C:\DATA.MDF] SET PAGE_VERIFY CHECKSUM   GO

ALTER DATABASE [C:\DATA.MDF] SET DB_CHAINING OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )  GO

ALTER DATABASE [C:\DATA.MDF] SET TARGET_RECOVERY_TIME = 60 SECONDS  GO

ALTER DATABASE [C:\DATA.MDF] SET DELAYED_DURABILITY = DISABLED  GO

ALTER DATABASE [C:\DATA.MDF] SET QUERY_STORE = OFF GO

ALTER DATABASE [C:\DATA.MDF] SET  READ_WRITE  GO
```

```
ALTER DATABASE [C:\DATA.MDF] SET READ_COMMITTED_SNAPSHOT OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET HONOR_BROKER_PRIORITY OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET RECOVERY SIMPLE    GO

ALTER DATABASE [C:\DATA.MDF] SET  MULTI_USER  GO

ALTER DATABASE [C:\DATA.MDF] SET PAGE_VERIFY CHECKSUM    GO

ALTER DATABASE [C:\DATA.MDF] SET DB_CHAINING OFF  GO

ALTER DATABASE [C:\DATA.MDF] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )  GO

ALTER DATABASE [C:\DATA.MDF] SET TARGET_RECOVERY_TIME = 60 SECONDS  GO

ALTER DATABASE [C:\DATA.MDF] SET DELAYED_DURABILITY = DISABLED  GO

ALTER DATABASE [C:\DATA.MDF] SET QUERY_STORE = OFF GO

ALTER DATABASE [C:\DATA.MDF] SET  READ_WRITE   GO
```

نقوم بإنشاء الاجراءات المخزنة في قاعدة البيانات لنقوم باستدعائها فيما بعد:

١. أجراء الاختيار وعرض محتويات الجدول كلها:

```
Create Proc select_person

AS
Select * From person
```

٢. إجراء إضافة سجل جديد.

```
Create Proc Insert_person
@ID int, @name VarChar(50), @Memo Varchar(200), @job_id int

AS
Insert Into person(ID, Name, Memo,job_id)
Values (@ID, @Name, @Memo, @job_id)
```

٣. إجراء حذف سجل حسب رقم الشخص.

```
Create Proc Delete_person
@ID int

AS
Delete  From person Where ID = @ID
```

٤. إجراء تعديل سجل شخص حسب رقمه.

```
Create Proc Update_person
@ID int, @name varchar(50), @Memo varchar(200), @job_id int

AS
Update person
Set ID=@ID, Name= @Name, Memo=@Memo , job_id@job_id

Where ID = @ID
```

- صمم شاشة (فورم) كما بالصورة التالية:

- بالنقر المزدوج على الفورم التي قمت بتصميمها كما بالصورة السابقة و استخدم المكتبة التالية
- ثم نذهب الي جزء كلاس الفورم ونكتب التالي:

```
using System.Data.SqlClient;
```

```
namespace DataAccessLAyer
{
    public partial class Person_Frm : Form
    {
        SqlConnection cn = new SqlConnection(
                           @"SERVER=.\SQL_ExPRESS;
                           AttachDBFilename= E:\C# Prog++++\Win_App\MyDB_Test\Data DB\Data.mdf;
                           User ID= sa;
                           Password = cocacolax2; ");

        // InitialCatalog = @" E:\C# Prog++++\Win_App\MyDB_Test\Data DB\Data.mdf "; //If DB Online
        //AttachDBFilename = @" E:\C# Prog++++\Win_App\MyDB_Test\Data DB\Data.mdf "; // If DB Offline

        SqlCommand cmd;
        SqlDataAdapter Da;
        DataTable Dt = new DataTable();


        public Person_Frm() //الفورم الرئيسية
        {
            InitializeComponent();
            cmd = new SqlCommand("Select_Person", cn); //-- Select_Person => Name of stored Procedure
            cmd.CommandType = CommandType.StoredProcedure; //-- Kind of query is stored Procedure

            Da = new SqlDataAdapter(cmd);
            Da.Fill(Dt); // -- Reads The Data

            //---------------------------------------

            //-- Put Data In DataGrid View (DataGV)
            this.DataGV.DataSource = Dt;
        }


       private void Person_Frm_Load(object sender, EventArgs e)
        {

        }

    }
}
```

-

- لتسهيل إعادة استخدام تحديث DataGV لعرض البيانات محدثة بعد كل اجراء حذف او إضافة ... الخ نضع التعليمات التالية في دالة يتم استدعائها بدل تكرار الأوامر كل مرة و ذلك كالتالي:

```
public void Refresh_DataGV()
        {
            Dt.Clear(); //-- افراغ الداتا تيبل قبل ملئه مرة اخرى
            cmd = new SqlCommand("Select_Person", cn);          //-- Select_Person => Name of stored Procedure
            cmd.CommandType = CommandType.StoredProcedure;      //-- Kind of query is stored Procedure

            Da = new SqlDataAdapter(cmd);
            Da.Fill(Dt); // -- Reads The Data

            //---------------------------------------

            //-- Put Data In DataGrid View (DataGV)
            this.DataGV.DataSource = Dt;
        }
```

لبرمجة زر الإضافة قم بالنقر المزدوج عليه ثم انسخ هذين السطرين:

```
cmd = new SqlCommand("Insert_Person", cn); //-- Select_Person => Name of stored Procedure
cmd.CommandType = CommandType.StoredProcedure; //-- Kind of query is stored Procedure
```

- الاجراء المخزن الخاص بإضافة سجل جديد في الجدول Person هو Person_Insert و اذا نظرنا داخلة سنجده يتعامل مع ثلاثة متغيرات + متغير رابع من نوع "زيادة تلقائية" و تتحكم به قاعدة البيانات لذلك سنقوم بالإعلان عن ثلاثة متغيرات ممررة (Parameters) في زر الإضافة للتعامل معها لنقل القيم للإجراء المخزن علي شكل مصفوفة متغيرات كما يلي.

```
SqlParameter[] param = new SqlParameter[4];
```

ثم نقوم بتعريف باقي المتغيرات من أعضاء المصفوفة كما يلي:

```
param[0] = new SqlParameter("@ID", SqlDbType.Int);
param[1] = new SqlParameter("@Name", SqlDbType.VarChar,50);
param[2] = new SqlParameter("@Memo", SqlDbType.VarChar, 200);
param[3] = new SqlParameter("@Job_id", SqlDbType.Int);
```

- ثم نقود بربط القيم المأخوذة من مربعات النصوص بهذه المتغيرات كالتالي:

```
param[0].Value = ID_txt.Text;
param[1].Value = Name_txt.Text;
param[2].Value = Memo_txt.Text;
param[3].Value = "100";
```

- ثم إضافة هذه المتغيرات للأمر (Command) المرتبط بتنفيذ الاجراء المخزن وتمريرها له وفتح الاتصال ثم تنفيذ الامر ثم الاغلاق و يكون كالتالي:

```
//-- add parameters to Command
cmd.Parameters.AddRange(param);
cn.Open();
cmd.ExecuteNonQuery();
cn.Close();
MessageBox.Show("لقد تم اضافة سجل جديد بنجاح ","نجاح الاتصال", MessageBoxButtons.OK, MessageBoxIcon.Information);
Refresh_DataGV();
```

لبرمجة زر الحذف:

- نقوم بالنقر المزدوج على زر الحذف ثم نكتب الكود التالي فيه:

```
cmd = new SqlCommand("Delete_person", cn);          //-- Select_Person => اسم الاجراء المخزن
cmd.CommandType = CommandType.StoredProcedure;     //-- تحيد نوع الاستعلام الي اجراء مخزن

//-- سنقوم بتعريف متغير واحد و ليس مصفوفة ID لان عملية الحذف تحتاج فقط الحقل
SqlParameter param = new SqlParameter();

param = new SqlParameter("@ID", SqlDbType.Int);

//-- هنا تحديد مربع النص الذي سنحصل منه علي قيمة الحقل ID
param.Value = ID_txt.Text;


//-- نمرر المتغيرات لكائن الامر
cmd.Parameters.Add(param);              //-- لقد استخدمنا هنا AddRange بدل ADD  لأننا نتعامل مع متغير واحد فقط نستخدم
cn.Open();
cmd.ExecuteNonQuery();
cn.Close();
MessageBox.Show("لقد تم حذف سجل بنجاح ","نجاح العملية", MessageBoxButtons.OK, MessageBoxIcon.Information);
Refresh_DataGV();
```

لبرمجة زر التعديل على سجل موجود:

- نقوم بنسخ الكود من زر إضافة جديد كما هو ثم نجري عليه بعض التعديلات (اسم الاجراء المخزن + الرسالة) كما يلي:

```
            cmd = new SqlCommand("Update_Person", cn); //-- Select_Person => Name of stored Procedure
            cmd.CommandType = CommandType.StoredProcedure; //-- Kind of query is stored Procedure

            //-- Defining Parameters Needed to Be Passed to Stored Procedure
            SqlParameter[] param = new SqlParameter[4];


            param[0] = new SqlParameter("@ID", SqlDbType.Int);
            param[1] = new SqlParameter("@Name", SqlDbType.VarChar, 50);
            param[2] = new SqlParameter("@Memo", SqlDbType.VarChar, 200);
            param[3] = new SqlParameter("@Job_id", SqlDbType.Int);

            //-- Linking Form Elements With Paramaeters (Give Them Values)

            param[0].Value = ID_txt.Text;
            param[1].Value = Name_txt.Text;
            param[2].Value = Memo_txt.Text;
            param[3].Value = "100";

            //-- add parameters to Command
            cmd.Parameters.AddRange(param);
            cn.Open();

            cmd.ExecuteNonQuery();
            cn.Close();

            Refresh_DataGV(); //--- Refresh Data Grid
            MessageBox.Show("لقد تم تعديل سجل  بنجاح" ,"التعديل", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

لعمل التعديل سنختار ان ينقر المستخدم على السجل من شبكة البيانات DataGV ليتم ملء مربعات النصوص بالبيانات الخاصة بهذا السجل ثم يقوم المستخدم بالتعديل على ما يريد منها ثم يضغط زر "تعديل" فيتم حفظ التعديلات الجديدة.

١. **ننقر نقراً مفرداً على DataGV لتحديدها ثم نذهب لخانة الخصائص ونختار الحدث Click ثم ننقر عليه نقر مزدوج فيتم فتح شاشة البرمجة الخاصة بالحدث ونكتب فيه التالي:**

```
int pos = DataGV.CurrentRow.Index; //-- هذا يعيد الاندكس الخاص بالصف (السجل) المختار
```

```
//-- نضع قيم السجل في مربعات النص المناسبة لكل واحد
 INC_txt.Text  = DataGV.Rows[pos].Cells[0].Value.ToString();
 ID_txt.Text   = DataGV.Rows[pos].Cells[1].Value.ToString();
 Name_txt.Text = DataGV.Rows[pos].Cells[2].Value.ToString();
 Memo_txt.Text = DataGV.Rows[pos].Cells[3].Value.ToString();
```

كتابة أوامر زر جديد (تفريغ):

```
 //-- This Button Just Clear all Text Boxes
 ID_txt.Clear();
 Name_txt.Clear();
 Memo_txt.Clear();
```

وهو يقوم فقط بتفريغ الحقول تمهيداً لكتابة قيم جديدة ثم النقر على زر إضافة لحفظ الملف الجديد.

كتابة أوامر زر إغلاق (وقف عمل البرنامج):

نكتب فيه جملة واحدة فقط هي `Application.Exit();`

لعمل تحديث للشبكة DataGV بالبيانات نكتب الجمل التالية في اجراء خاص ونستدعيه بعد تنفيذ كل أمر وكذلك في بداية عمل البرنامج:

```
 public void Refresh_DataGV()
        {
            Dt.Clear();

            cmd = new SqlCommand("Select_Person", cn); //-- Select_Person => Name of stored Procedure
            cmd.CommandType = CommandType.StoredProcedure; //-- Kind of query is stored Procedure

            Da = new SqlDataAdapter(cmd);
            Da.Fill(Dt); // -- Reads The Data

            //-- Put Data In DataGrid View (DataGV)
            this.DataGV.DataSource = Dt;
        }
```

تعديلات

إلي هنا كل شي جيد و لكن هناك نقطة تم تأجيلها ألا و هي اننا نتعامل مع جدولين (Person , Jobs) مرتبطين معاً بمعني أنه لكي يتم الإضافة في جدول Person يجب تعيين نوع الوظيفة للشخص قبل الحفظ أو التعديل و نحن كنا نستخدم عند التعديل او حفظ السجل الجديد جملة

```
param[3].Value = "100";
```

لتحديد رقم الوظيفة ١٠٠ من جدول الوظائف و لكي يعمل برنامجنا بشكل واقعي و طبيعي فيجب الاختيار عند الإضافة و التعديل من قائمة الوظائف المخزنة في جدول الوظائف Jobs عن طريق ComboBox و لذلك يجب إضافة كمبوبوكس للفورم و سيكون لها ارتباط بالجدول الثاني (Jobs) و رقم الوظيفة المختارة سيحفظ في جدول (Person) لأنه مفتاح اجنبي و لا يتم الحفظ بدون كتابته، لذلك سنجري بعض التغييرات في الفورم فقط و لن نغير شيئاً في جمل الإجراءات المخزنة لأنها كنا نتعامل معها مسبقاً لكننا كنا نمرر لها القيمة ١٠٠ كما في الجملة السابقة تفادياً لحصول خطأ عند الحفظ.

لذلك قم بالتغييرات التالية:

١. أولا أضف comboBox للفورم واجعل خاصية Enabled= False لمنع التعديل فيها فهي للقراءة فقط.

٢. قم بإنشاء دالة (Method) أو سميها اجراء يتم استدعائه للاتصال بالجدول (Jobs) و استدعاء البيانات منه و مل أسماء الوظائف من الحقل المسمى (Job_Desc) للكمبوبوكس كالتالي:

```
public void Fill_Jobs_Cmo()
        {
            SqlDataAdapter Da2;
            DataTable Dt2 = new DataTable();
            Da2 = new SqlDataAdapter("select * from Jobs", cn);
            Da2.Fill(Dt2);

            //Jobs_Combo.Dispose();              //--- Clear Combo First

            Jobs_Combo.DataSource = Dt2;       //-- put Da2 as DataSource of Data For Jobs Combo Box
            Jobs_Combo.DisplayMember = "Job_Desc";    //-- Spicify The Field Of data To Be desplayed
            Jobs_Combo.ValueMember = "Job_id";        //-- The Value To Tracked On selection
        }
```

وذلك بالاعتماد على جملة الاتصال بقاعدة البيانات cn نفسها المستخدمة من قبل في البرنامج ثم استبدال الجملة المذكورة سابقاً في الحفظ والتعديل وهي:

```
param[3].Value = "100";  ==➔    param[3].Value = Jobs_Combo.SelectedValue.ToString();      //-- Takes Job_id Value from Combo
```

وذلك في حالة الحفظ الجديد وحالة التعديل

مثال على استخدام الإجراءين السابقين عند تحميل الفورم:

```
    public Person_Frm()
        {
            InitializeComponent();

            Refresh_DataGV();
            Fill_Jobs_Cmo(); //-- Fill Combob of Jobs Names
        }
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DataAccessLAyer
{
    public partial class Person_Frm : Form
    {
        SqlConnection cn = new SqlConnection(
                           @"SERVER=.\SQL_ExPRESS;
                           AttachDBFilename= E:\C# Prog++++\Win_App\MyDB_Test\Data DB\Data.mdf;

                           User ID= sa;
                           Password = cocacolax2;

                           ");
        // InitialCatalog = @"C:\Users\un_mo\Desktop\My Test DB\Data.mdf"; //If DB Online
        //AttachDBFilename = @"C:\Users\un_mo\Desktop\My Test DB\Data.mdf"; // If DB Offline

        SqlCommand cmd;
        SqlDataAdapter Da;
        DataTable Dt = new DataTable();



        public void Fill_Jobs_Cmo()
        {
            SqlDataAdapter Da2;
            DataTable Dt2 = new DataTable();
            Da2 = new SqlDataAdapter("select * from Jobs", cn);
            Da2.Fill(Dt2);

            //Jobs_Combo.Dispose();              //--- Clear Combo First

            Jobs_Combo.DataSource = Dt2;       //-- put Da2 as DataSource of Data For Jobs Combo Box
            Jobs_Combo.DisplayMember = "Job_Desc";    //-- Spicify The Field Of data To Be desplayed
            Jobs_Combo.ValueMember = "Job_id";        //-- The Value To Tracked On selection
        }




        public void Refresh_DataGV()
        {
            Dt.Clear();

            cmd = new SqlCommand("Select_Person", cn); //-- Select_Person => Name of stored Procedure
            cmd.CommandType = CommandType.StoredProcedure; //-- Kind of query is stored Procedure

            Da = new SqlDataAdapter(cmd);
            Da.Fill(Dt); // -- Reads The Data

            //-- Put Data In DataGrid View (DataGV)
            this.DataGV.DataSource = Dt;
        }




        public Person_Frm()
        {
            InitializeComponent();
            Refresh_DataGV();
            Fill_Jobs_Cmo(); //-- Fill Combob of Jobs Names
        }



        private void Person_Frm_Load(object sender, EventArgs e)
        {
```

```
}



private void Add_btn_Click(object sender, EventArgs e)
{
    cmd = new SqlCommand("insert_Person", cn); //-- Select_Person => Name of stored Procedure
    cmd.CommandType = CommandType.StoredProcedure; //-- Kind of query is stored Procedure


    //-- Defining Parameters Needed to Be Passed to Stored Procedure
    SqlParameter[] param = new SqlParameter[4];


    param[0] = new SqlParameter("@ID", SqlDbType.Int);
    param[1] = new SqlParameter("@Name", SqlDbType.VarChar,50);
    param[2] = new SqlParameter("@Memo", SqlDbType.VarChar, 200);
    param[3] = new SqlParameter("@Job_id", SqlDbType.Int);

    //-- Linking Form Elements With Paramaeters (Give Them Values)

    param[0].Value = ID_txt.Text;
    param[1].Value = Name_txt.Text;
    param[2].Value = Memo_txt.Text;
    param[3].Value = Jobs_Combo.SelectedValue.ToString();  //-- Takes Job_id Value from Combo

    //-- add parameters to Command
    cmd.Parameters.AddRange(param);
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();

    Refresh_DataGV(); //--- Refresh Data Grid
    //Fill_Jobs_Cmo(); //-- Fill Combob of Jobs Names

    MessageBox.Show("لقد تم اضافة سجل جديد بنجاح" ,"نجاح الاتصال", MessageBoxButtons.OK, MessageBoxIcon.Information);
}




private void Delete_btn_Click(object sender, EventArgs e)
{

    cmd = new SqlCommand("Delete_person", cn); //-- Select_Person => Name of stored Procedure
    cmd.CommandType = CommandType.StoredProcedure; //-- Kind of query is stored Procedure

    //-- Defining1 parameter Needed to Be Passed to Stored Procedure
    //-- Here No Need to define array => []
    SqlParameter param = new SqlParameter();


    param = new SqlParameter("@ID", SqlDbType.Int);

    //-- Linking Element With Paramaeter (Give it Value)
     param.Value = ID_txt.Text;


    //-- add parameters to Command
    cmd.Parameters.Add(param); //-- Here Used .Add not .AddRange Because Its just One Parameter
    cn.Open();
    cmd.ExecuteNonQuery();
    cn.Close();
    MessageBox.Show("لقد تم حذف سجل بنجاح" ,"نجاح العملية", MessageBoxButtons.OK, MessageBoxIcon.Information);

    Refresh_DataGV(); //--- Refresh Data Grid
    Fill_Jobs_Cmo(); //-- Fill Combob of Jobs Names

}




private void Update_btn_Click(object sender, EventArgs e)
{

    cmd = new SqlCommand("Update_person", cn); //-- Select_Person => Name of stored Procedure
    cmd.CommandType = CommandType.StoredProcedure; //-- Kind of query is stored Procedure
```

```
            //-- Defining Parameters Needed to Be Passed to Stored Procedure
            SqlParameter[] param = new SqlParameter[4];


            param[0] = new SqlParameter("@ID", SqlDbType.Int);
            param[1] = new SqlParameter("@Name", SqlDbType.VarChar, 50);
            param[2] = new SqlParameter("@Memo", SqlDbType.VarChar, 200);
            param[3] = new SqlParameter("@Job_id", SqlDbType.Int);

            //-- Linking Form Elements With Paramaeters (Give Them Values)

            param[0].Value = ID_txt.Text;
            param[1].Value = Name_txt.Text;
            param[2].Value = Memo_txt.Text;
            param[3].Value = Jobs_Combo.SelectedValue.ToString(); //-- Takes Job_id Value from Combo

            //-- add parameters to Command
            cmd.Parameters.AddRange(param);
            cn.Open();
            cmd.ExecuteNonQuery();
            cn.Close();

            Refresh_DataGV(); //--- Refresh Data Grid
            Fill_Jobs_Cmo(); //-- Fill Combob of Jobs Names

            MessageBox.Show( "نجاح التعديل" ,"لقد تم تعديل سجل  بنجاح" ,MessageBoxButtons.OK, MessageBoxIcon.Information);
        }




        private void DataGV_Click(object sender, EventArgs e)
        {
            int pos = DataGV.CurrentRow.Index; //-- هذا يعيد الاندكس الخاص بالصف (السجل) المختار

            //-- نضع قيم السجل في مربعات النص المناسبة لكل  واحد
            INC_txt.Text = DataGV.Rows[pos].Cells[0].Value.ToString();

            ID_txt.Text = DataGV.Rows[pos].Cells[1].Value.ToString();
            Name_txt.Text = DataGV.Rows[pos].Cells[2].Value.ToString();
            Memo_txt.Text = DataGV.Rows[pos].Cells[3].Value.ToString();


        }




        private void New_btn_Click(object sender, EventArgs e)
        {
            //-- This Button Just Clear all Text Boxes
            ID_txt.Clear();
            Name_txt.Clear();
            Memo_txt.Clear();

            Fill_Jobs_Cmo(); //-- Fill Combob of Jobs Names
        }

        private void Exit_btn_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }



        //private void Jobs_Combo_SelectedIndexChanged(object sender, EventArgs e)
        //{
        //    MessageBox.Show(Jobs_Combo.SelectedValue.ToString());
        //}
    }
}
```

الكود البرمجي للبرنامج الرئيسي **Programm.cs**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

using System.Text;
using System.Data.SqlClient;
using System.Data;


namespace DataAccessLAyer
{

    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]

        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Person_Frm());
        }
    }
}
```

هذا كل ما لدي حتي اللحظة بهذا الخصوص، ارجو أن يستفيد الجميع من هذا الجهد المتواضع
لا تبخلوا علّى بالدعاء لوالدي فهذا العمل صدقة جارية عن روحه (رحمه الله).