



```
import 'react'; 8.23 kB
import 'next/head';
import { AppProps } from 'next/app'; 5.11 kB (gzip)
import { ApolloProvider } from '@apollo/client'; 123.67 kB
import { ThemeProvider } from '@material-ui/core/styles'; 2.45 kB
import CssBaseline from '@material-ui/core/CssBaseline'; 61.61 kB (gzip)
import { Container } from '@material-ui/core'; 63.32 kB (gzip: 20.38 kB)
import { useApollo } from '../graphql/client';

import { lightTheme, darkTheme } from '../utils/theme';
import useLocalStorage from '../hooks/useLocalStorage';

import NavBar from '../components/NavBar';

function App({ Component, pageProps }: AppProps) {
  const [currentTheme, setCurrentTheme] = useLocalStorage({ key: 'theme-value', initialValue: 'light' });
  const apolloClient = useApollo(pageProps.initialApolloState);

  useEffect(() => {
    const jssStyles = document.querySelector(selectors: '#jss-server-side');
    if (jssStyles) {
      jssStyles.parentElement.removeChild(jssStyles);
    }
  }, []);

  return (
    <>
    <Head>
    <title>For
    </Head>
  )
}
```

2023

الثانوية المهنية الصناعية في سوريا

القسم النظري

البرمجة 3

الثالث الثانوي تقنيات الحاسوب

إعداد : رامي سلطان المسالمة

هندسة نظم المعلومات

الجامعة الافتراضية

الوحدة الأولى (البرمجة غرضية التوجه)

ما الهدف من البرمجة غرضية التوجه ؟

الهدف هو إتمام عملية تطوير البرامج المعقدة بشكل اسرع ، واستخدامها بشكل اسرع واوسع واسهل ، وبعدد غير محدد من المرات .

تعريف البرمجة غرضية التوجه OOP ؟

هي البرمجة التي تعتمد على الأغراض والعلاقات فيما بينها لإنشاء البرامج بشكل متكامل.

على ماذا تعتمد البرمجة غرضية التوجه ؟

تعتمد البرمجة غرضية التوجه على المفاهيم التالية :

١ - الأغراض Objects	٢ - الخصائص Properties	٣ - القيم Values	٤ - الطرق Methods	٥ - الأحداث Events
---------------------	------------------------	------------------	-------------------	--------------------

تعريف الأغراض Objects :

هي وحدات مستقلة يتم ضمنها تغليف البيانات وإخفاؤها ، ولكل غرض خصوصيته ، فلا يمكنه الاطلاع على بيانات الأغراض الأخرى ، والغرض هو نسخة من الصف Class .

تعريف الخصائص Properties :

تحدد صفات تميز الغرض ، يتم من خلالها الدخول الى المعطيات المغلفة ضمن الأغراض ، لكل غرض مجموعة من الخصائص المختلفة عن خصائص غرض اخر كما يوجد بعض الخصائص المشتركة (كالاسم Name).

تعريف القيم Values :

هي قيم الخصائص وتكون نصية أو عددية أو قيماً لونية أو منطقية .

تعريف الطرق Methods :

هي توابع وإجراءات تطبق على الغرض ، مثال : اجراء مسح النموذج من النص المكتوب عليه ; `this.Clear()` .

أحداث الأدوات Events :

هي كل فعل ينتج عن استخدام الماوس أو لوحة المفاتيح مع الأداة ، أو النقر المزدوج أو السحب والافلات أو ضغط مفتاح من لوحة المفاتيح وغيرها من الأحداث .

عند الانتقال الى نافذة الشيفرة لأي أداة ، يظهر إجراء مكون من : |الصيغة٢| مما يتكون إجراء حدث الاداة ؟

١ - الاسم البرمجي للأداة ٢ - الحدث المطبق على الاداة ٣ - وسطاء الاجراء مثال (Object sender , EventArgs e)

الوسيط **Sender** : يدل على الأداة التي طبق عليها الحدث ، كما قد يكون لدينا أكثر من أداة لها الحدث نفسه.
الوسيط **e** : يدل على المعلومات التي تخص الحدث المطبق على الأداة كحدث النقر **MouseDown** .

عدد خطوات انشاء مشروع جديد بلغة VC# :

١. فتح برنامج Microsoft Visual Studio 2010 .
٢. من صفحة البداية نختار **New Project** أو من القائمة **File** ننقر الأمر **New** ثم ينقر الأمر **Project** .
٣. نختار من واجهة تحديد نوع المشروع التي ظهرت نوع المشروع فنختار **Windows Form Application** .
٤. نكتب اسم المشروع في الحقل **Name** .
٥. نختار المكان المطلوب حفظ المشروع فيه من الحقل **Location** (بالنقر على الزر **Browse** أو من خلال القائمة المنسدلة).
٦. يظهر في الحقل **Solution Name** الاسم نفسه الذي تمت كتابته في الحقل **Name** للدلالة على أن **Solution** له اسم المشروع نفسه. **ملاحظة** : يمكن للـ **Solution** أن يحوي أكثر من مشروع.
٧. النقر على الزر **OK** لتظهر واجهة المشروع .

عند انشاء مشروع باسم **MyFirstProgram** تظهر ملفات في المكان **Location** الذي حفظ فيه المشروع .
عددها مع الشرح .

١. **الملف Form1.cs** : هو ملف يحفظ الشيفرات الخاصة بالنموذج **Form1** ويظهر في نافذة مستكشف المشاريع وامتداد هذا الملف هو **cs** اختصار لاسم لغة البرمجة **C Sharp** .
٢. **الملف MyFirstProgram.csproj** : هو ملف يحفظ المشروع المنشأ ، وامتداد هذا الملف هو **csproj** اختصاراً للعبارة **C Sharp Project** .
٣. **الملف program.cs** : يحفظ الشيفرة الخاصة بالمشروع ، حيث يحوي هذا الملف في شيفرته التابع **Main()** الذي يبدأ تنفيذ البرنامج منه ، وامتداد هذا الملف هو **cs** .
٤. **الملف Form1.Designer.cs** : هو ملف يحوي الشيفرات المسؤولة عن انشاء عناصر الواجهة وتنظيمها .

عدد طرق الانتقال الى نافذة تصميم النموذج .

١. النقر المزدوج على **Form1.cs** في نافذة مستكشف المشروع .
٢. تحديد **Form1.cs** ثم ينقر زر **View Design** أعلى نافذة مستكشف المشاريع .

عدد طرق الانتقال الى نافذة الشيفرة .

١. النقر المزدوج على **Form1.Designer.cs** من مستكشف المشروع .
٢. اختيار الامر **Code** من القائمة **View** .

٣. النقر المزدوج على أي أداة من الأدوات المضافة الى النموذج .
٤. تحديد النموذج أو الأداة ، من نافذة الخصائص ينقر زر الاحداث **Events** ، ثم ينقر نقراً مزدوجاً في الحقل يمين الحدث المطلوب .
٥. الضغط على مفتاح **F7** من لوحة المفاتيح .

أدوات بناء التطبيقات في VC# وخصائصها

تعريف الأداة .

تستخدم الأدوات في بناء المشروع ، والأداة هي غرض **Object** تقوم بوظيفة معينة في البرنامج ، وتختلف الأدوات عن بعضها بالوظيفة والخصائص **Properties** والاحداث **Events** .

تقسم الأدوات في Visual Studio ضمن مجموعات مبوبة ، عددها .

١- الأدوات الأكثر استخداماً Common Controls	٢- الأدوات الحاوية لغيرها من الأدوات Containers
٣- الأدوات الخاصة بالقوائم Menus & Toolbars	٤- الأدوات الخاصة بقواعد البيانات Data الخ

عدد طرق إظهار صندوق الأدوات في برنامج Visual Studio .

١ - نقر الأمر ToolBox من القائمة View .	٢. نقر تبويب ToolBox الموجود يسار نافذة VC#
---	---

تضاف الأدوات الى النموذج بطريقتين :

- ١- **الرسم بالماوس** : يمكن التحكم بأبعاد و موقع الأداة على النموذج .
- ٢- **النقر المزدوج** : تضاف الأداة اعلى ويسار النموذج ، ويمكن سحب الأداة من مكانها ووضعها في أي مكان من النموذج .

مما تتألف نافذة الخصائص ؟

١. قائمة منسدلة تحوي أسماء ادوات المشروع الحالي .
٢. شريط الأدوات  للتحكم بما تعرضه نافذة الخصائص ، ويحوي الازرار :
 - ❖ زر **Categorized** لعرض الخصائص بشكل مجموعات . 
 - ❖ زر **Alphabetical** لعرض الخصائص مرتبة ابجدياً . 
 - ❖ زر **Properties** لعرض الخصائص وقيمها ويكون مفعلاً بشكل تلقائي . 
 - ❖ زر **Events** لعرض احداث الأداة الحالية . 
٣. جدول مؤلف من عمودين ، يعرض العمود الأول أسماء الخصائص والثاني قيمها، وتكون الخاصية وقيمتها في صف واحد.
٤. شريط المعلومات ، يعرض معلومات نصية بسيطة عن الخاصية المحددة من جدول الخصائص .

تغيير قيم خصائص الأدوات كالتالي :

١. تحديد الأداة من النموذج بالنقر عليها ، أو نقر القائمة المسندلة لنافذة الخصائص واختيار اسم الأداة المطلوب تغيير قيم خصائصها .

٢. النقر في العمود الثاني من جدول الخصائص في صف الخاصية التي نريد تغيير قيمتها .

- إذا كانت قيمة الخاصية نصية أو عددية نكتب النص أو العدد المطلوب .
- إذا كانت القيم على شكل قائمة منسدلة نختار القيمة المطلوبة من القائمة .
- إذا كانت القيم لونية ، ينقر مربع اللون المطلوب أو يكتب اسم اللون مباشرة في صندوق قيم الخاصية .
- اما إذا كانت على شكل زر يظهر عليه ثلاث نقاط (بمعنى استعراض) ، عند نقره يظهر مربع حوار لتحديد تنسيقات معينة للخاصية كمربع حوار تنسيق النص أو مربع حوار تحديد مسار خلفية للنموذج .

تحذف الأدوات من النموذج بالطرق التالية :

الطريقة الاولى	الطريقة الثانية	الطريقة الثالثة
١. تحديد الأداة من النموذج .	١. تحديد الأداة من النموذج .	١. نقر الأداة بالزر الأيمن للماوس لتظهر القائمة المختصرة .
٢. ضغط Delete من لوحة المفاتيح.	٢. نقر الامر Delete من القائمة Edit .	٢. اختيار الامر Delete .

يمكن التراجع عن أي امر تم تطبيقه على الأداة بإحدى الطرق التالية :

- ★ بالضغط على المفاتيح **Ctrl + Z** من لوحة المفاتيح .
- ★ نقر زر التراجع **Undo** من شريط الأدوات القياسي لنافذة **VC#** .
- ★ نقر الامر **Undo** من القائمة **Edit** .

يمكن إعادة تطبيق الأمر على الأداة بإحدى الطرق التالية : (التراجع عن التراجع)

- ★ بالضغط على المفاتيح **Ctrl + Y** من لوحة المفاتيح .
- ★ نقر زر التراجع **Redo** من شريط الأدوات القياسي لنافذة **VC#** .
- ★ نقر الامر **Redo** من القائمة **Edit** .

ملاحظات هامة جداً

- ☺ تدل الكلمة المحجوزة **this** على النسخة الحالية أو الغرض الذي تم انشاءه من الصف الحالي (أي في **منهاجنا تدل على الاداة Form1**).
- ☺ يوجد بعض الخصائص المشتركة كالاسم **Name** لكن مع اختلاف قيم هذه الخصائص .
- ☺ (ما) الهدف من البرمجة غرضية التوجه ؟ إتمام عملية تطوير البرامج المعقدة بشكل اسرع و اوسع واسهل و بعدد غير محدد من المرات .
- ☺ تعد البرمجة غرضية التوجه **OOP** تطويراً للبرمجة البنوية حيث البرمجة البنوية تعتمد في أساسها على تقسيم البرنامج الى مجموعة من الاجرائيات يتم استدعاؤها عند الحاجة اليها .
- ☺ السطر البرمجي الاتي يبين رأس اجراء النقر على الزر **button1** :

Private void button1_Click(object sender, EventArgs e)

- ☺ يطلق مصطلح المشروع Project على البرنامج Program اثناء تصميمه وتطويره وكتابة شيفرته .
- ☺ يطلق مصطلح التطبيق Application على البرنامج Program اثناء تنفيذه .
- ☺ يمكن أن يحوي الـ Solution على أكثر من مشروع كما يكون اسمه هو اسم المشروع نفسه .
- ☺ (لا يمكن) إضافة ادوات من أداة صندوق الأدوات Toolbox إلا اذا كنا في نافذة التصميم (النموذج) لأن صندوق الأدوات يكون فارغاً في حال كنا في نافذة الشيفرة .
- ☺ لإظهار أي نافذة في برنامج VC# من القائمة View ثم النافذة المطلوبة .

تعريف أداة النموذج Form :

هي الأرضية التي تستخدم لوضع الأدوات عليها اثناء التصميم ، وتمثل النافذة التي ستظهر أثناء التنفيذ وتعد من أهم الأدوات .

من أهم خصائص أداة النموذج Form

١- الخاصية Name :

تمثل اسم الأداة البرمجي المستخدم عند كتابة الشيفرة (كما هي خاصية مشتركة بين جميع الادوات).

٢- الخاصية AutoScroll :

تسمح بعرض أو عدم عرض أشرطة التمرير (الأفقية والعمودية أو الاثنين معاً) وذلك عندما تكون إحدى الأدوات خارج حدود النموذج ولها قيمتان :

- ❖ True : تسمح بعرض أشرطة التمرير على النموذج ، وتكتب تعليماتها برمجياً كالتالي :
`this.AutoScroll = true ;`
- ❖ False : لاتسمح بعرض أشرطة التمرير على النموذج ، وتكتب تعليماتها برمجياً كالتالي :
`this.AutoScroll = false ;`

3- الخاصية BackColor :

تستخدم لتلوين خلفية النموذج (وهي خاصية مشتركة بين عدة ادوات).

مثال : لتلوين خلفية النموذج باللون الأزرق برمجياً ، نكتب التعليمات كالتالي :

`this.BackColor = Color.Blue;`
`this.BackColor = Color.FromName("Blue");`

4- الخاصية BackgroundImage :

تستخدم لاختيار صورة كخلفية للنموذج ، وذلك بتحديد مسار الصورة .

ويتم تحديد مسار الصورة كالتالي :

١- باستخدام نافذة الخصائص :

- ✂ ينقر على النموذج ثم من نافذة الخصائص ينقر المربع المنقط (استعراض) للخاصية BackgroundImage فيظهر مربع حوار تحديد المصادر ،
- ✂ يحدد أحد الخيارين Local Resource أو Project Resource ثم ينقر الزر Import لفتح مربع حوار فتح الملفات ومن خلاله يتم تحديد الصورة المطلوبة .
- ✂ ثم ينقر الزر Open فتضاف الصورة الى قائمة مصادر الصور المستخدمة في المشروع (إن تم تحديد الخيار Project Resource) ثم تحدد الصورة فتظهر معاينة لها في النافذة ثم ينقر الزر OK لتظهر كخلفية للنموذج .

٢- برمجياً :

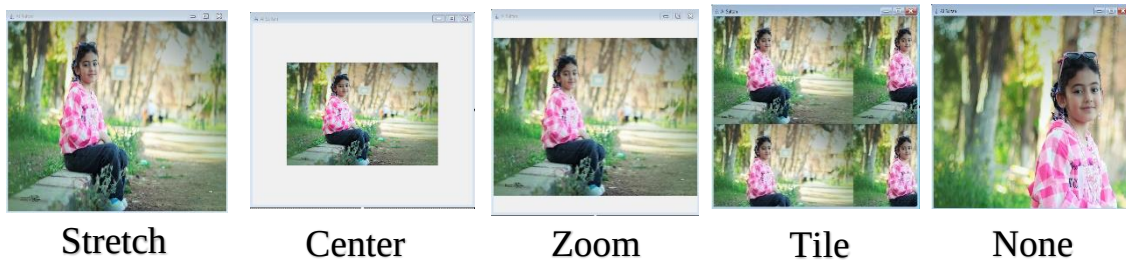
تكتب التعليمة كالتالي :

this.BackgroundImage=Image.FromFile("D:\\Ramy\\Sultan\\Almasalmh.Png");

5- الخاصية BackgroundImageLayout :

تحدد شكل عرض الصورة في خلفية النموذج ولها القيم التالية :

تعرض الصورة في الخلفية بحيث تنطبق زاويتها العليا اليسرى على الزاوية العليا اليسرى للنموذج ولا تتغير ابعاد الصورة أو ابعاد النموذج .	None
تتكرر الصورة بشكل متجانس لتملأ خلفية كامل النموذج ، ويظهر ذلك بوضوح اذا كانت أبعاد الصورة أصغر من أبعاد النموذج ، اما اذا كانت أبعاد الصورة اكبر من أبعاد النموذج ، فلن يكون لهذا الخيار أي تأثير حيث تعرض الصورة كما القيمة None .	Tile
يتم ضبط مركز الصورة على مركز النموذج ، هذا في حال كانت ابعاد الصورة اصغر من ابعاد النموذج ، أما اذا كانت ابعاد النموذج اكبر من ابعاد الصورة ، تظهر الصورة كما القيمة None .	Center
يتم تغيير ابعاد الصورة لتصبح مساوية لأبعاد النموذج .	Stretch
يتم تغيير أبعاد الصورة نسبياً (نسبة الطول الى العرض) لتعرض كاملة .	Zoom



٦- الخاصية Enabled :

تستخدم لتفعيل وإلغاء تفعيل النموذج ، ويقصد بذلك تفعيل وإلغاء تفعيل كل الأدوات الموجودة على النموذج وهي خاصية مشتركة بين عدة أدوات ، لها قيمتان :

١ - True : لتفعيل النموذج	٢ - False : إلغاء تفعيل النموذج
---------------------------	---------------------------------

٧- الخاصية ForeColor :

تستخدم لتحديد لون الامامية أو لون الكتابة فوق النموذج ، كما تمثل لون الكتابة للعناوين الظاهرة للأدوات التي توضع على النموذج ، مع إمكانية تغيير لون كل أداة بشكل منفرد (وهي خاصية مشتركة بين عدة أدوات).

مثال : لتلوين الامامية للنموذج باللون الذهبي برمجياً ، نكتب التعليمة كالتالي :

```
this.BackColor = Color.Gold;
```

```
this.BackColor = Color.FromName("Gold");
```

٨- الخاصية MaximizeBox :

تستخدم للسماح أو عدم السماح بتكبير حجم النافذة للحد الأقصى **أثناء التنفيذ** ولها قيمتان :

١ - True : تسمح بتكبير النافذة للحجم الأقصى	٢ - False : لا تسمح بتكبير النافذة للحجم الأقصى
---	---

٩- الخاصية MinimizeBox :

تستخدم للسماح أو عدم السماح بتصغير حجم النافذة للحد الأقصى **أثناء التنفيذ** ولها قيمتان :

١ - True : تسمح بتصغير النافذة لشريط المهام	٢ - False : لا تسمح بتصغير النافذة لشريط المهام
---	---

١٠- الخاصية RightToLeft :

تستخدم للسماح أو عدم السماح بتعريب النموذج (وهي خاصية مشتركة بين عدة أدوات) ولها القيم التالية :

Yes : **تسمح** بتعريب النموذج أي العنوان يظهر على اليمين `this.RightToLeft=RightToLeft.Yes;`

No : **لا تسمح** بتعريب النموذج أي العنوان يظهر على اليسار `this.RightToLeft=RightToLeft.No;`

١١- الخاصية Size :

تحدد ابعاد النموذج (عرضه Width ، ارتفاعه Height) على التوالي (وهي خاصية مشتركة بين عدة أدوات).

❖ لا يمكن اسناد قيم لهذه الخاصية برمجياً إلا باستخدام التابع البناء **Size** على النحو التالي :

```
this.Size = new Size(150 , 120);
```

حيث : 150 هو عرض النموذج Width و 120 هو ارتفاع النموذج Height .

١٢- الخاصية StartPosition :

تحدد موقع ظهور نافذة النموذج على شاشة الحاسوب عند أول تنفيذ **ولها قيم** :

يحدد موقع النموذج على الشاشة حسب الخاصية Location(x,y) للنموذج .	Manual
يحدد موقع النموذج في مركز الشاشة حيث ينطبق مركز نافذة النموذج على مركز الشاشة .	CenterScreen
يحدد موقع النموذج على الشاشة حسب الإعدادات الافتراضية لنظام التشغيل ، وتكون أبعاده حسب خاصية ارتفاعه Height وعرضه Width .	WindowsDefaultLocation
يحدد موقع النموذج على الشاشة حسب الإعدادات الافتراضية لنظام التشغيل ، وتكون أبعاده حسب إعدادات نظام التشغيل وليست حسب أبعاد النموذج .	WindowsDefaultBounds
يحدد موقع النموذج على الشاشة في مركز النافذة الأب للنموذج عند استخدام مشروع متعدد النماذج .	CenterParent

١٣- الخاصية Tag :

تستخدم لتخزين معلومات عن الأداة دون أن تكون ظاهرة ، حيث تستخدم هذه التعليمة برمجياً **(وهي خاصية مشتركة بين عدة ادوات)**.

١٤- الخاصية Text :

تمثل اسم الأداة الظاهري أو عنوان الأداة ، وفي حالة النموذج تمثل عنوان النافذة الذي يظهر في شريط العنوان أثناء التنفيذ **(وهي خاصية مشتركة بين عدة ادوات)**.

مثال عن الخاصيتين السابقتين :

this.Tag = "Ramy";

this.Text = this.Tag.ToString();

في التعليمة الأولى تخزن السلسلة النصية **"Ramy"** كمعلومات عن النموذج ، وفي التعليمة الثانية تعرض السلسلة النصية المخزنة كعنوان للنافذة ، بعد تحويلها للنمط النصي باستخدام التابع ToString().

١٥- الخاصية WindowState :

تستخدم لتحديد حجم النافذة أثناء التنفيذ **ولها ثلاث قيم** :

✂ Normal : تكون النافذة بالحجم الافتراضي . **وتسند برمجياً كالآتي :**

this.WindowState=FormWindowState.Normal;

✂ Minimized : تصغر النافذة لشكل ايقونة في شريط المهام . **وتسند برمجياً كالآتي :**

this.WindowState=FormWindowState.Minimized;

✂ Maximized : تكبر النافذة لأقصى حجم . **وتسند برمجياً كالآتي :**

this.WindowState=FormWindowState.Maximized;

١٦- الخاصية Opacity :

تستخدم لتغيير قيمة عتامة النموذج ، قيمتها محصورة بين 0 و 1 .
تكتب برمجياً كالآتي :

```
this.Opacity = 0.8f;
```

١٧- الخاصية AllowDrop :

تستخدم للسماح أو عدم السماح بإفلات الأدوات المسحوبة بالماوس ضمن منطقة النموذج أثناء تنفيذ البرنامج (وهي خاصية مشتركة بين عدة أدوات) ولها قيمتان :

١. True : تسمح بإفلات الأدوات المسحوبة ضمن منطقة النموذج .
٢. False : لا تسمح بإفلات الأدوات المسحوبة ضمن منطقة النموذج .

لتغييرها برمجياً نكتب :

```
This.AllowDrop = True;
```

ملاحظة : في بعض الأحيان عند كتابة الخاصية AllowDrop لا تظهر ضمن قائمة التنبؤ عندها يجب كتابة الخاصية يدوياً مع مراعاة حالة الحروف .

ملاحظات هامة

- عندما تكون كل من الخاصيتين MaximizeBox و MinimizeBox تساوي False لا يظهر في شريط عنوان النموذج سوى أيقونة الاغلاق .
- اما اذا كانت احدهما فقط False تظهر كل من إشارة الاغلاق والتصغير والتكبير مع الغاء تفعيل الايقونة التي قيمتها False .
- عندما تكون الخاصية RightToLeft.Yes هذا يعني انها مطبقة على جميع الأدوات داخل النموذج مع إمكانية تغيير هذه الخاصية لكل أداة على حدا .
- يمكن كتابة تعليمية تغيير الحجم بطريقة أخرى :

```
this.Width = 150;
```

```
this.Height = 120;
```

تعريف أداة العنوان Label

هي أداة تستخدم لكتابة النصوص ضمنها مع (عدم) إمكانية تحريرها (إثناء التنفيذ) من لوحة المفاتيح .

من أهم خصائص أداة العنوان Label

١- الخاصية AutoSize :

تحدد فيما إذا كان حجم الأداة بحجم النص المكتوب فيها أم لا ولها قيمتان :

١. True : تصبح الأداة بحجم النص المكتوب فيها ، ولا يمكن تغيير حجمها يدوياً .
٢. False : تحافظ الأداة على حجمها الافتراضي ، مع إمكانية تغيير حجمها يدوياً .

٢- الخاصية Text :

تمثل النص المكتوب ضمن أداة العنوان ، مع إمكانية كتابة أسطر متعددة باستخدام مفتاح الإدخال ENTER (وهي خاصية مشتركة بين عدة أدوات).

مثال عن الخاصيتين السابقتين :

1- Label1.Text = " My Name Is Ramy ";

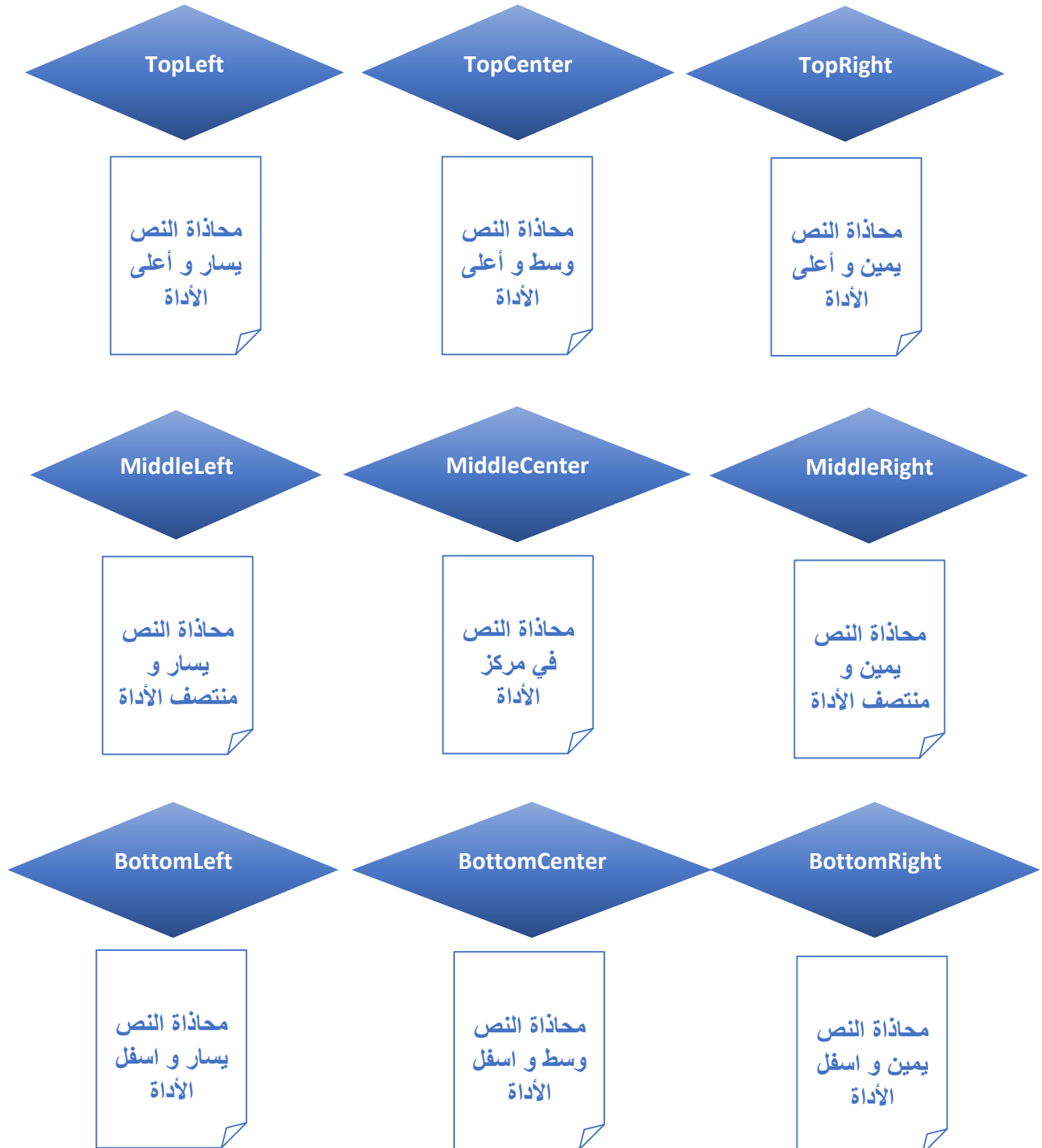
2- Label1.AutoSize = true ;

شرح التعليمات :

- ١- يعرض النص (My Name Is Ramy) ضمن أداة العنوان Label1 .
- ٢- يكون حجم الأداة بحجم النص المكتوب ضمنها .

٣- الخاصية TextAlign :

تحدد محاذاة النص المكتوب ضمن أداة العنوان . ولها القيم التالية :



لتغيير قيمة هذه الخاصية **برمجياً** نكتب :

```
Label1.TextAlign = ContentAlignment.BottomCenter;
```

نشرح هذه التعليمة كالتالي :

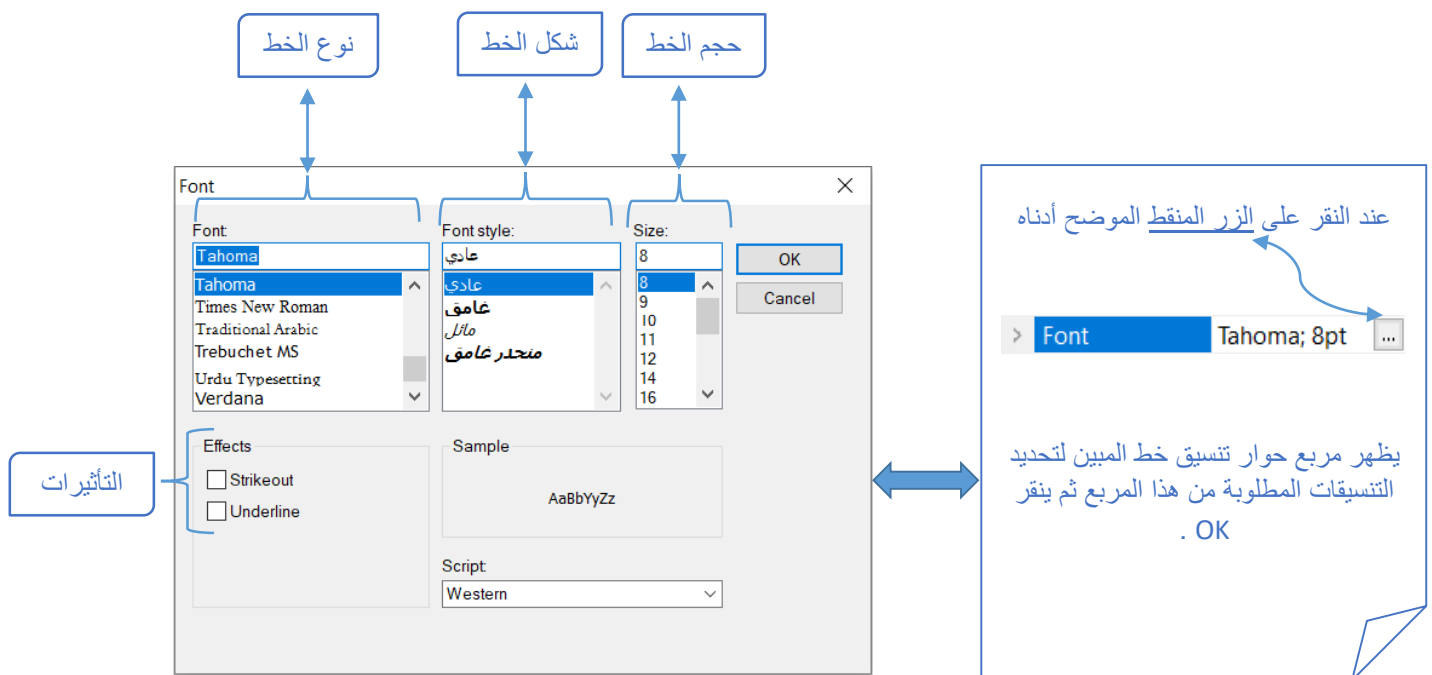
اجعل محاذاة النص وسط و اسفل أداة العنوان label1 .

ملاحظة :

إن الوسيط **ContentAlignment** يستخدم للوصول لقيم المحاذاة الـ (٩) المذكورة سابقاً لخاصية المحاذاة لأداة العنوان .

٤- الخاصية Font :

تحدد نوع الخط لنص الأداة وحجمه (**Size**) وشكله (**Style**) من حيث السماكة (**Bold**) وغيرها من التأثيرات (**Effects**) كالـتسطير (**UnderLine**) أو الخط المنصف (**Strikeout**) . (وهي خاصية مشتركة بين عدة أدوات) .



لتنسيق الخط **برمجياً** يستخدم (**الشكل العام**) التالي:

```
ControlName.Font = new font ("FontName" , Size , FontStyle.Effects , GraphicsUnit.Unit ) ;
```

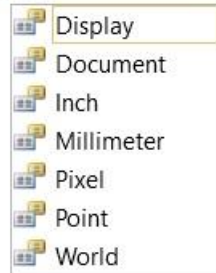
حيث :

- ★ ControlName : اسم الأداة المراد تغيير تنسيق الخط لها .
- ★ FontName : يمثل اسم الخط المراد تطبيقه .
- ★ Size : يستخدم لتحديد حجم الخط .
- ★ FontStyle : (تعريف) هو صف من التأثيرات يستخدم لتحديد التأثير المطلوب تطبيقه على النص .
- ★ GraphicsUnit : (تعريف) هو صف من وحدات القياس ، يستخدم لتحديد وحدة القياس المراد تحديد حجم النص بها .

مثال على ما سبق :

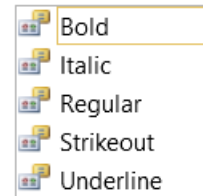
```
textBox1.Text = new Font ("Arial" , 18 , FontStyle.Bold , GraphicsUnit.Inch) ;
```

GraphicsUnit.



قائمة وحدات القياس في الصف
GraphicsUnit

FontStyle.



قائمة التأثيرات في الصف **FontStyle**

تعريف أداة الزر Button :

تستخدم في بناء معظم التطبيقات ، لتنفيذ وظيفة معينة عند النقر عليها ، وتحدد هذه الوظيفة بحسب الشيفرات المكتوبة في الزر .

من اهم خصائص أداة الزر Button

١- الخاصية FlatStyle :

تحدد شكل الزر . **لها اربع قيم :**

- ❖ **Flat** : يظهر الزر بشكل مستطيل مسطح ، ثنائي البعد
 - ❖ **Popup** : يظهر الزر بشكل مستطيل مسطح أيضاً ، لكن عند مرور مؤشر الفأرة على الزر يتحول الى الشكل البارز .
 - ❖ **Standard** : يظهر الزر بشكل ثلاثي الأبعاد ، وهي القيمة الافتراضية .
 - ❖ **System** : يظهر الزر بالشكل نفسه الذي يظهر به أي زر في النظام عند ظهور أي رسالة أو مربع حوار بالنظام .
- مثال** على ماسبق :

```
button1.FlatStyle = FlatStyle.Popup ;
```

ملاحظة هامة :

إن الوسيط **FlatStyle** يستخدم لإسناد قيم اشكال الزر السابقة الى الخاصية FlatStyle لأداة الزر .

٢- الخاصية Locked :

تسمح بفتح أو عدم الأداة و المقصود بعملية قفل الأداة (عدم السماح بتحريكها أو تغيير أبعادها أثناء عملية التصميم) ، تستخدم في مرحلة التصميم فقط و لا يمكن استخدامها برمجياً (وهي خاصية مشتركة بين عدة أدوات)

لها قيمتان :

١. True : تفعل الأداة وتظهر إشارة القفل عليها (أثناء عملية التصميم) .
٢. False : تكون الأداة غير مقفولة .

٣- الخاصية TabIndex :

تحديد ترتيب الأداة عند الضغط على مفتاح الجدولة Tab لنقل التركيز إليها .

شرح لبعض قيمها :

- إذا كانت قيمة هذه الخاصية (0) فإن التركيز سيكون على الأداة عند بداية تنفيذ المشروع .
- إذا كانت قيمة هذه الخاصية (1) سينتقل التركيز إليها عند ضغط على مفتاح Tab لمرة واحدة .
- إذا كانت قيمة هذه الخاصية (2) سينتقل التركيز إليها عند ضغط على مفتاح Tab لمرتين وهكذا... وتكتب برمجياً كالتالي :

button1.TabIndex = 1 ;

نقل التركيز الى الأداة button1 عند ضغط مفتاح Tab مرة فقط .

٤- الخاصية TabStop :

تستخدم للسماح أو عدم السماح بنقل التركيز الى الأداة عند ضغط مفتاح الجدولة Tab .

لها قيمتان :

١. True : تسمح بنقل التركيز الى الأداة .
٢. False : لا تسمح بنقل التركيز الى الأداة مهما تم ضغط مفتاح الجدولة Tab .

ملاحظة :

تعمل الخاصية TabIndex إذا كانت قيمة الخاصية TabStop = true فقط .

الحدث الافتراضي للأداة Button هو Click والشكل العام له

تضاف الأداة Button لتنفيذ تعليمة أو مجموعة من التعليمات البرمجية ضمن إجراء حدث النقر على الزر .

والشكل العام لإجراء النقر على الزر Button كالتالي :

```
private void ButtonName_Click ( object sender , EventArgs e)
{
    Statements;
}
```

حيث يتألف رأس الاجراء من اسم أداة الزر **ButtonName** والحدث **Click()** .

مثال ١ : عند النقر على الزر button 3 يتم إنهاء / إغلاق / الخروج من التطبيق .

```
private void button3_Click ( object sender , EventArgs e)
{
    Application.Exit() ;
}
```

مثال ٢ : عند النقر على الزر button 4 يتم إنهاء / إغلاق / الخروج من النموذج .

```
private void button4_Click ( object sender , EventArgs e)
{
    this.Close();
    // أو
    // Close();
}
```

ملاحظات هامة

عندما يبدأ الطلب بـ (عند النقر) هذا يعني ان التعليمة ستكتب ضمن إجراء حدث النقر **Click()** على الزر **Button** كما المثالين السابقين .
يجب التفريق بين طلب (إغلاق التطبيق) و (إغلاق النموذج) .
الفرق بين إغلاق التطبيق **Application.Exit()** وإغلاق النموذج **this.Close()** أنه اذا كان لدينا مشروع مؤلف من عدة واجهات فإن الامر **Close()** سيغلق الواجهة نفسها اما الامر **Exit()** سيغلق جميع نماذج البرنامج .

ملاحظة :

في مناهجنا العمل كله على نموذج واحد لذلك لن تلاحظوا الفرق بين الأمرين ، لكن وجب التوضيح .

تعريف أداة مربع النص **TextBox** :

تستخدم لكتابة النصوص ضمنها ، مع إمكانية تحرير هذه النصوص أثناء التنفيذ باستخدام لوحة المفاتيح .

من أهم خصائص أداة مربع النص TextBox

١ – الخاصية Lines :

لإدخال نص مؤلف من سطر أو أكثر ، ويعد كل سطر عنصراً من عناصر مصفوفة من نمط السلسلة المحرفية String .

مثال :

```
textbox2.Text = textbox1.Lines[1];
```

شرح التعليمة :

عرض السطر الثاني المدخل عن طريق الخاصية Lines لأداة مربع النص textbox1 ضمن مربع النص textbox2 .

هام : (تم عرض السطر الثاني لأنه يبدأ العد في المصفوفة من [0] لذلك إن [1] هي السطر الثاني)

٢ – الخاصية Text :

لإدخال نص ضمن مربع النص ويعد النص المدخل سلسلة محرفية واحدة .

مثال :

```
textbox3.Text = " RamyAlmasalmh ";
```

شرح التعليمة :

عرض العبارة **RamyAlmasalmh** في مربع النص textbox3 .

٣ – الخاصية MultiLine :

تحدد إمكانية إدخال أسطر متعددة ضمن مربع النص وذلك عند ضغط المفتاح Enter ولها قيمتان :

١. True : تسمح بإدخال أسطر متعددة .
٢. False : لا تسمح بإدخال أسطر متعددة .

ملاحظات

✂ إذا كانت قيمة الخاصية `MultiLine = true` وتم إدخال نص بأسطر متعددة باستخدام الخاصية `Lines` ، فإن الأسطر المدخلة سيظهر كل منها ضمن سطر مستقل في أداة مربع النص ، حيث يتم الانتقال بين هذه الأسطر باستخدام المفاتيح (أعلى ، أسفل) من لوحة المفاتيح ، ذلك في حال كان حجم مربع النص اصغر من عدد الاسطر المدخلة ضمنه باستخدام الخاصية `Lines` .

✂ إذا كانت قيمة الخاصية `MultiLine = false` وتم إدخال نص بأسطر متعددة باستخدام الخاصية `Lines` ، فإن الاسطر المدخلة ستظهر جميعها على سطر واحد ، أي ستبدو ظاهرياً سلسلة حرفية واحدة ، لكنها في الواقع مصفوفة سلاسل حرفية .

٤ – الخاصية `MaxLength` :

تحدد عدد المحارف الأعظمي الممكن إدخاله ضمن مربع النص .

٥ – الخاصية `ReadOnly` :

تحدد إمكانية التعديل على محتوى مربع النص أثناء التنفيذ . لها قيمتان :

١. <code>True</code> : لا تسمح بتعديل محتوى مربع النص أثناء التنفيذ ، مع إمكانية تعديل محتواه برمجياً .
٢. <code>False</code> : تسمح بتعديل محتوى مربع النص أثناء التنفيذ .

٦ – الخاصية `TextAlign` :

تحدد محاذاة النص ضمن مربع النص . لها ثلاث قيم :



وتكتب برمجياً كالتالي :

```
textBox2 . TextAlign = HorizontalAlignment . Center ;
textBox3 . TextAlign = HorizontalAlignment . Right ;
textBox1 . TextAlign = HorizontalAlignment . Left ;
```

ملاحظة :

لدينا وسطين للوصول للمحاذاة على حسب الأداة المطلوبة :

الوسيط **HorizontalAlignment** : وهو وسيط يستخدم للوصول لقيم المحاذاة الـ (3) السابقة لأداة مربع النص **TextBox** .

الوسيط **ContentAlignment** : وهو وسيط يستخدم للوصول لقيم المحاذاة الـ (٩) المذكورة سابقاً لخاصية المحاذاة لأداة العنوان **Label** .

٧ - الخاصية **AllowDrop** :

تستخدم للسماح بإفلات المعطيات ضمن منطقة مربع النص (وهي خاصية مشتركة) .
وتكتب برمجياً كالتالي :

`textBox1.AllowDrop = true ;`

عبارة معالجة الاستثناءات Exception Handling

عند حدوث بعض الأخطاء في البرنامج ، يمكن تجاوزها وإيقاف كتلة التعليمات التي حدث فيها الخطأ ، باستخدام العبارة **try** وعبارة معالجة الخطأ **catch** .

الشكل العام لها و مبدأ عملها :

```
try
{
    statements1;
}
catch
{
    statements2;
}
```

عند حدوث خطأ في مجموعة التعليمات **Statement1** يتم إيقاف تنفيذها والخروج منها وتنفيذ مجموعة تعليمات **Statement2**

الشكل العام لمربع حوار الرسالة **MessageBox**

`MessageBox.Show("نص الرسالة", "عنوان الرسالة", MessageBoxButtons.الزر, MessageBoxIcon.الرمز);`

مثال :

```
MessageBox.Show("الرجاء ادخال رقم", "خطأ", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Error);
```



ملاحظات

يمكن فحص القيمة المدخلة في مربع النص باستخدام التابع Equal () الذي يعيد قيمة منطقية true \ false كالآتي :

```
if(textBox1.Text.Equals(""))
MessageBox.Show("الرجاء ادخال رقم", "خطأ", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

تشير هذه التعليمة الى أن الشرط لن يتحقق الا اذا كانت textBox1 تساوي " " أي الفراغ بمعنى اخر اذا كان محتوى مربع النص textBox1 فارغاً يتحقق الشرط بعرض مربع حوار رسالة خطأ نصها "الرجاء ادخال رقم" وعنوانها "خطأ" وازرارها (موافق) ورمزها (استفهام) .

يمكن كتابة الشرط السابق بطريقة أخرى من خلال تبديل التابع Equals () بـ (=) كالآتي :

```
if(textBox1.Text=="")
MessageBox.Show("الرجاء ادخال رقم", "خطأ", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

تعريف أداة الاختيار CheckBox :

تستخدم لتحديد أو عدم تحديد (خيار واحد أو أكثر) من مجموعة اختيارات .

من اهم خصائص أداة الاختيار CheckBox

١ – الخاصية Appearance :

تحدد مظهر أداة مربع الاختيار . لها قيمتان :

Normal : الشكل الافتراضي للأداة .	Button : يظهر مربع الاختيار على شكل زر .
-----------------------------------	--

checkBox1

☐ checkBox1

مع ملاحظة يمكن التمييز بين أداة الاختيار CheckBox في حال كان شكلها كالزر Button وبين أداة الزر Button أن أداة الاختيار CheckBox في حال كانت شكلها Button عند اختيارها يبقى الزر مضغوطاً وعند إزالة اختيارها تظهر الأداة بالشكل البارز .

checkBox1

checkBox1

في حالة عدم الاختيار

في حالة الاختيار

ونكتب برمجياً كالتالي :

checkBox1.Appearance = Appearance.Button;

يتم استخدام الوسيط Appearance للوصول للقيمتين (Button , Normal) .

٢ – الخاصية AutoCheck :

تحدد إمكانية اختيار الأداة أو عدم اختيارها . لها قيمتان :

true : تسمح باختيار مربع الاختيار .	false : لا تسمح باختيار مربع الاختيار .
-------------------------------------	---

٣ – الخاصية Checked :

تحدد فيما إذا كانت أداة مربع الاختيار مختارة ام لا . لها قيمتان :

true : الأداة مختارة .	false : الأداة غير مختارة .
------------------------	-----------------------------

٤ - الخاصية CheckAlign :

تحدد موقع (مربع الاختيار) بالنسبة لنص الأداة . ولها (9) قيم :

☐ RamyAlmasalmh	☐ RamyAlmasalmh	☐ RamyAlmasalmh
TopLeft : يكون مربع الاختيار يسار و اعلى النص	TopCenter : يكون مربع الاختيار وسط و اعلى النص	TopRight : يكون مربع الاختيار يمين و اعلى النص
☐ RamyAlmasalmh	☐ RamyAlmasalmh	☐ RamyAlmasalmh
MiddleLeft : يكون مربع الاختيار يسار النص وعلى محاذاته	MiddleCenter : يكون مربع الاختيار وسط النص وعلى محاذاته	MiddleRight : يكون مربع الاختيار يمين النص وعلى محاذاته
☐ RamyAlmasalmh	☐ RamyAlmasalmh	☐ RamyAlmasalmh
BottomLeft : يكون مربع الاختيار يسار و اسفل النص	BottomCenter : يكون مربع الاختيار وسط و اسفل النص	BottomRight : يكون مربع الاختيار يمين و اسفل النص

وتسند برمجياً بإستخدام الوسيط **ContentAlignment** كالآتي :

```
checkBox1.CheckAlign = ContentAlignment.BottomRight;
```

ملاحظة :

كما مر معنا سابقاً نستخدم الوسيط **ContentAlignment** مع اداتي **Label** و **CheckBox** لأن لهن 9 قيم للمحاذاة بينما نستخدم الوسيط **HorizontalAlignment** مع أداة **textbox** لانه لها 3 قيم للمحاذاة .

٥ – الخاصية CheckState :

تحدد حالة مربع الاختيار ولها ثلاث قيم :

Unchecked : الأداة غير مفعلة
Checked : الأداة مفعلة
Indeterminate : تكون حالة الأداة غير محددة ويظهر مربع أزرق دون ان تظهر إشارة الصح .

٦ – الخاصية ThreeState :

تحدد فيما اذا كان مربع الاختيار سيمر بحالتي (الاختيار وعدم الاختيار فقط) أم بثلاث حالات وهي (الاختيار وعدم الاختيار وعدم التحديد) ولها قيمتان :

True : تسمح بمرور الأداة بثلاث حالات من خلال النقر على الأداة .
False : لا تسمح بمرور مربع الاختيار الا بحالتي الاختيار وعدم الاختيار فقط .

٧ – الخاصية Cursor :

تحدد شكل مؤشر الفأرة عند مروره فوق الأداة (وهي خاصية مشتركة بين عدة ادوات) .

٨ – الخاصية Location :

تحدد موقع الأداة بالنسبة للزاوية العليا اليسرى للنموذج ولها قيمتان :

(X) : بعد الأداة عن الحافة اليسرى للنموذج.	(Y) : بعد الأداة عن الحافة العليا للنموذج .
--	---

ملاحظة : لا يمكن تغيير قيم (X , Y) برمجياً بشكل مباشر انما تستخدم الخاصيتين Top , Left للأداة لتغيير قيمها برمجياً .

مثال :

checkBox1.Top = 350 ;

checkBox.Left = 250;

الشرح :

تحديد بعد الأداة CheckBox1 عن الحافة العليا للنموذج بمقدار ٣٥٠ وعن الحافة اليسرى بمقدار ٢٥٠ .

٩ – الخاصية Visible :

تسمح بإظهار أو إخفاء الأداة (وهي خاصية مشتركة بين عدة أدوات) . لها قيمتان :

True : تكون الأداة ظاهرة .
False : تكون الأداة مخفية .

الحدث الافتراضي للأداة CheckBox هو CheckedChanged والشكل العام له

Private void checkBoxName_CheckedChanged (object sender , EventArgs e)

```
{
    Statements ;
}
```

تعريف أداة مربع النص RadioButton :

هي أداة تستخدم لاختيار خيار واحد فقط من مجموعة خيارات تنتمي للمجموعة نفسها .

ملاحظة :

إذا كان هناك اداتي RadioButton وكل أداة تنتمي لمجموعة ، فيمكن اختيار الاداتين معاً دون أن يؤثر احدهما على الأخرى .

من اهم خصائص الأداة RadioButton

١ – الخاصية Appearance :

تحدد مظهر الاداة. لها قيمتان :

Normal : الشكل الافتراضي للأداة .	Button : تظهر الأداة على شكل زر .
-----------------------------------	-----------------------------------

checkBox1

☐ radioButton1

تكتب برمجياً كالتالي :

radioButton1.Apperance = Apperance.Button;

لتظهر الأداة بشكل زر اثناء التنفيذ .

٢ – الخاصية Checked :

تحدد فيما إذا كانت الاداة مختارة ام لا . لها قيمتان :

false : الاداة غير مختارة .

true : الاداة مختارة .

الحدث الافتراضي للاداة RadioButton هو CheckedChanged والشكل العام له

Private void radioButtonName_CheckedChanged (object sender , EventArgs e)

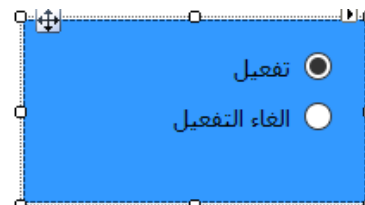
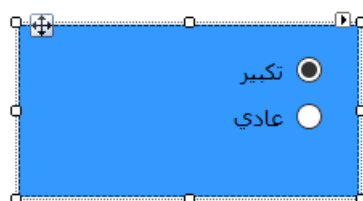
```
{
    Statements ;
}
```

تعريف الاداة Panel :

تستخدم لتجميع مجموعة أدوات مع بعضها البعض بحيث تعد أداة واحدة وتضاف الى النموذج وترسم الأدوات ضمنها وعند حذف الاداة تحذف جميع الأدوات التي تحويها .

ملاحظة :

بحسب تعريف أداة RadioButton يمكننا اختيار خيار واحد فقط من مجموعة اختيارات لكن بالمثال التالي لدينا اداتي RadioButton تم اختيارهم وذلك لأن كل أداة تعتبر في مجموعة منفصلة عن الاداة الأخرى وذلك لوجود الادتين في أداة Panel مختلفة عن الأخرى ونستنتج من ذلك انه يمكن اختيار خيار واحد فقط من المجموعة نفسها أما اذا كانت الأدوات كل منها في مجموعة فيمكنني اختيار اكثر من أداة على حسب عدد المجموعات .



تعريف أداة القائمة المنسدلة ComboBox :

تستخدم لعرض مجموعة من البنود على شكل قائمة منسدلة ، بحيث نختار بنداً واحداً فقط من مجموعة البنود .

من أهم خصائص الأداة ComboBox

١ – الخاصية Items :

تحدد البنود المضافة للأداة .

تتم عملية إضافة البنود الى الأداة بطريقتين :

١ – **من الخصائص** : وذلك بالنقر على المربع المنقط في الخاصية **Items** فيظهر مربع حوار لكتابة البنود فيه بحيث يفصل بين كل بند وآخر بضغط مفتاح **Enter** ثم ينقر الزر **OK** .

٢ – **برمجياً** : وذلك باستخدام التابع **Add** للخاصية **Items** كالتالي :

```
ComboBox1.Items.Add("Ramy");
```

٢ – الخاصية Sorted :

تستخدم لترتيب البنود ضمن القائمة . **لها قيمتان** :

True : تكون البنود ضمن الأداة **مرتبة تصاعدياً** .

False : تكون البنود **غير مرتبة** .

٣ – الخاصية MaxLength :

تحدد العدد الاعظمي للمحارف الممكن إدخاله ضمن الأداة اثناء التنفيذ .

وتكتب برمجياً كالتالي :

```
ComboBox1.MaxLength = 4 ;
```

٤ – الخاصية DropDownStyle :

تحدد شكل الأداة . **لها ثلاث قيم** :

تظهر الأداة بشكلها البسيط ، بحيث تعرض البنود كافة على شكل قائمة ظاهرة دون النقر على سهم الأداة و لاختيار أي بند ينقر عليه بالفأرة .	: Simple
تظهر الأداة بالشكل الافتراضي ، حيث ينقر سهم الأداة ، لعرض البنود واختيار البند المطلوب منها ، أو يكتب الحرف الأول من البند ضمن الأداة ثم يضغط احد مفاتيح الأسهم من لوحة المفاتيح لعرض البند المطلوب .	: DropDown
تظهر الأداة بشكل قائمة منسدلة ، دون إمكانية الكتابة في الأداة ، و لاختيار أي بند ينقر على سهم الأداة فتعرض البنود ويختار البند المطلوب او بالضغط على الحرف الأول من لوحة المفاتيح فيعرض البند مباشرة .	:DropDownList

٥ - الخاصية SelectedIndex :

تستخدم برمجياً فقط ، لتحديد رقم العنصر المختار من القائمة المنسدلة مع ملاحظة أن ترتيب البنود يبدأ من الصفر .

تكتب برمجياً كالتالي :

ComboBox1.SelectedIndex = 0 ;

٦ - الخاصية SelectedItem :

تستخدم برمجياً فقط ، لتحديد العنصر المختار الذي تم اختياره من القائمة المنسدلة .

تكتب برمجياً كالتالي :

textBox1.Text = ComboBox1.SelectedItem.ToString();

لعرض البند المختار من القائمة المنسدلة ضمن مربع النص TextBox1 بعد تحويله الى نص .

٧ - اجراء حذف بند () Remove:

يستخدم لحذف بند من القائمة المنسدلة حسب قيمة البند .

الشكل العام لإجراء حذف البند :

ComboBoxName.Items.Remove(Item);

حيث Item : هو الوسيط لقيمة البند ويمكن أن يكون قيمة مباشرة أو خاصية أو متحول .

مثال :

```
comboBox1.Items.Remove("String"); // قيمة مباشرة
comboBox1.Items.Remove(textBox1.Text); // قيمة خاصية
comboBox1.Items.Remove(Strx); // قيمة متحول نصي
```

٨ - إجراء حذف بند (RemoveAt) :

يستخدم لحذف بند من القائمة المنسدلة حسب ترتيب البند .

الشكل العام لإجراء حذف البند :

```
ComboBoxName.Items.RemoveAt(ItemIndex);
```

حيث Item : هو الوسيط ترتيب البند ويمكن أن يكون قيمة مباشرة أو خاصية أو متحول يبدأ ترتيبه من الصفر.

مثال :

```
comboBox1.Items.RemoveAt(7); // حذف البند الثامن // قيمة مباشرة
comboBox1.Items.RemoveAt(comboBox1.SelectedIndex); // حذف البند المحدد حسب ترتيبه // قيمة خاصية
comboBox1.Items.RemoveAt(IntX); // قيمة متحول رقمي
```

الحدث الافتراضي للأداة ComboBox هو SelectedIndexChanged والشكل العام له

حيث ينفذ هذا الإجراء عند اختيار بند أو تغيير البند المحدد من القائمة المنسدلة .

شكله العام :

```
private void comboBoxName_SelectedIndexChanged ( object sender , EventArgs e)
{
    Statements ;
}
```

تعريف أداة لائحة البنود ListBox :

تستخدم لعرض عدد من البنود على شكل قائمة مؤلفة من عمود واحد أو عدة أعمدة .

من أهم خصائص الأداة ListBox

١ - الخاصية HorizontalScrollbar :

تسمح بظهور شريط التمرير الأفقي على الأداة ، وذلك في حال كان طول (عدد المحارف) أحد البنود على الأقل يزيد عن عرض القائمة . **لها قيمتان :**

True : تسمح بإظهار شريط التمرير الأفقي .
False : لا تسمح بإظهار شريط التمرير الأفقي .

٢ - الخاصية MultiColumn :

تسمح بعرض بنود القائمة في أكثر من عمود ، وذلك عند ازدياد عدد بنود القائمة عن طول الأداة ، كما تتفاعل هذه الخاصية مع الخاصية HorizontalScrollbar . **لها قيمتان :**

True : تسمح بعرض البنود في أكثر من عمود .
False : لا تسمح بعرض البنود في أكثر من عمود .

ملاحظات هامة

- في حال كان عدد بنود القائمة أصغر من طول الأداة ، فلا يظهر شريط التمرير الأفقي الا اذا زاد عدد احد محارف البنود عن عرض الأداة ، وكانت قيمة الخاصية HorizontalScrollbar = true
- في حال زاد عدد بنود القائمة عن طول الأداة فإننا **نميز حالتين :**
 - اذا كانت قيمة الخاصية MultiColumn = true يتم تلقائياً إضافة شريط التمرير الأفقي للأداة ، ولا يتم إضافة شريط التمرير الشاقولي ، بغض النظر عن قيمة الخاصية HorizontalScrollbar
 - اذا كانت قيمة الخاصية MultiColumn = false يتم إضافة شريط التمرير الشاقولي للأداة ولا يتم إضافة شريط التمرير الأفقي الا اذا كانت الخاصية HorizontalScrollbar = true وكان طول احد البنود على الأقل يزيد عن عرض القائمة .

٣ - الخاصية SelectionMode :

تحدد عدد البنود الممكن اختيارها معاً من القائمة ، وطريقة تحديد هذه البنود . **لها القيم التالية :**

لا يختار أي بند من بنود القائمة .	: None
يختار بنوداً واحداً فقط من القائمة .	: One
يختار أكثر من بند من القائمة ، حيث يتم اختيار كل بند بالنقر عليه فقط .	: MultiSimple
يختار أكثر من بند من القائمة ، حيث يتم اختيار كل بند بالنقر عليه بالفأرة بالإضافة الى مفتاح Ctrl .	: MultiExtended

وتكتب برمجياً كالتالي :

```
listBox1.SelectionMode = SelectionMode.MultiSimple;
```

شرح التعليمة :

جعل حالة الاختيار للأداة **listBox1** تسمح باختيار أكثر من بند من القائمة ، بنقر كل بند بالفأرة فقط .

٤ - الخاصية SelectedIndex :

تستخدم برمجياً فقط ، لتحديد رقم العنصر المختار من القائمة مع ملاحظة أن ترتيب البنود يبدأ من الصفر .

تكتب برمجياً كالتالي :

```
listBox1.SelectedIndex = 0 ;
```

٥ - الخاصية SelectedItem :

تستخدم برمجياً فقط ، لتحديد العنصر المختار الذي تم اختياره من القائمة.

تكتب برمجياً كالتالي :

```
textBox1.Text = listBox1.SelectedItem.ToString();
```

لعرض البند المختار من القائمة ضمن مربع النص TextBox1 بعد تحويله الى نص .

٦ - اجراء إضافة البنود Add() :

يستخدم لإضافة بند الى نهاية القائمة .

الشكل العام لإجراء إضافة بند الى اللائحة // القائمة :

```
listBox1.Items.Add(Item);
```

حيث Item هو وسيط عن قيمة البند المدخل وقد تكون قيمة نصية أو قيمة عددية .

ملاحظة : اجراء إضافة البند مشترك بين الاداتين ListBox و ComboBox .

```
listBox1.Items.Add("Sultan"); // قيمة مباشرة
```

```
listBox1.Items.Add(textBox.Text); // قيمة خاصة
```

```
listBox1.Items.Add(R) // قيمة متحول
```

٧ - اجراء إضافة مجال من البنود AddRange() :

يستخدم لإضافة مجموعة من البنود الى نهاية القائمة .

الشكل العام لإجراء إضافة مجال من البنود الى اللائحة :

```
listBoxName.Items.AddRange(Range) ;
```

حيث Range هي وسيط مجال القيم التي نريد اضافتها الى القائمة ، كما يمكن أن تكون مصفوفة أحادية البعد .

مثال :

```
string[ ] Color = new string[ ] { "Blue" , "Green" , "Black" , "Cyan" };
```

```
listBox1.Items.AddRange(Color);
```

شرح التعليمات :

انشاء مصفوفة اسمها Color من النوع string تحوي عدداً من العناصر ثم إضافة هذه العناصر الى أداة listBox1 .

٨ - حساب عدد عناصر لائحة البنود باستخدام Count :

يستخدم لحساب عدد عناصر (بنود) أداة لائحة البنود .

مثال :

```
textBox1.Text = listBox1.Items.Count;
```

شرح التعليمة :

عرض عدد عناصر لائحة البنود listBox1 في مربع النص textBox1 .

٩ - التابع Contains() :

يستخدم للبحث في أداة لائحة البنود عن بند ما .

مثال :

```
if ( listBox1.Items.Contains (textBox1.Text) )
```

```
listBox1.SelectedItem = textBox1.Text ;
```

شرح التعليمة :

إذا كانت أداة لائحة البنود listBox1 تحتوي في بنودها على بند قيمته تساوي القيمة المكتوبة في أداة صندوق النص textBox1 سيتم تحديد هذا البند .

٧ - إجراء حذف بند () Remove:

يستخدم لحذف بند من القائمة حسب قيمة البند .

الشكل العام لإجراء حذف البند :

```
listBoxName.Items.Remove(Item);
```

حيث Item : هو الوسيط لقيمة البند ويمكن أن يكون قيمة مباشرة أو خاصية أو متحول .

مثال :

```
listBox1.Items.Remove("Ramy"); // قيمة مباشرة
listBox1.Items.Remove(textBox1.Text); // قيمة خاصية
listBox1.Items.Remove(strx); // قيمة متحول نصي
```

٨ - إجراء حذف بند () RemoveAt:

يستخدم لحذف بند من القائمة حسب ترتيب البند .

الشكل العام لإجراء حذف البند :

ComboBoxName.Items.RemoveAt(ItemIndex);

حيث Item : هو الوسيط ترتيب البند ويمكن أن يكون قيمة مباشرة أو خاصية أو متحول يبدأ ترتيبه من الصفر.

مثال :

```
listBox1.Items.RemoveAt(7); // حذف البند الثامن // قيمة مباشرة
listBox1.Items.RemoveAt(comboBox1.SelectedIndex); // حذف البند المحدد حسب ترتيبه // قيمة خاصية
listBox1.Items.RemoveAt(IntX); // قيمة متحول رقمي
```

الحدث الافتراضي للأداة listBox هو SelectedIndexChanged والشكل العام له

حيث ينفذ هذا الإجراء عند اختيار بند أو تغيير البند المحدد من لائحة البنود.

شكله العام :

private void listBoxName_SelectedIndexChanged (object sender , EventArgs e)

```
{
    Statements ;
}
```

ملاحظة

يوجد الكثير من الخصائص المشتركة بين ListBox و ComboBox .

تعريف أداة صندوق الصورة PictureBox:

هي عبارة عن صندوق ليتم عرض الصورة فيه .

من أهم خصائص الأداة PictureBox

١ - الخاصية BorderStyle :

تحدد شكل إطار صندوق الصورة (وهي خاصية مشتركة بين عدة أدوات) . لها ثلاث قيم :

يظهر صندوق الأداة دون إطار.	: None
يظهر صندوق الأداة بإطار ثنائي البعد بخط مفرد وحيد .	: FixedSingle
يظهر صندوق الأداة بإطار ثلاثي الأبعاد .	: Fixed3D

٢- الخاصية Image :

تحدد الصورة التي ستعرض ضمن صندوق الأداة (وهي خاصية مشتركة بين عدة أدوات) .

وتكتب برمجياً لعرض الصورة ضمن الصندوق كالتالي :

```
pictureBox1.Image = Image.FromFile("D:\\Ramy\\Ms.Png");
```

شرح التعليمة :

تحميل الصورة من المسار "D:\\Ramy\\Ms.Png" وإسنادها للخاصية Image لعرض الصورة ضمن صندوق الصورة .

٣- الخاصية SizeMode :

تحدد طريقة عرض الصورة ضمن صندوق الأداة . ولها القيم الآتية :

لعرض اعلا ويسار الصورة حسب حجم صندوق الأداة وعند تكبير الأداة يظهر جزء اكبر من الصورة المعروضة .	Normal
لتغيير ابعاد الصورة ليصبح لها ابعاد صندوق الأداة نفسه .	StretchImage
لتغيير ابعاد صندوق الصورة ليصبح له ابعاد الصورة نفسها .	AutoSize
لضبط مركز الصورة على مركز صندوق الأداة ولا تتغير ابعاد الصورة أو ابعاد الصندوق.	CenterImage
لضبط الابعاد نسبياً (نسبة الطول الى العرض) وعرضها بشكل كامل ضمن صندوق الأداة ولا تتغير ابعاد صندوق الأداة .	Zoom

ملاحظة

لإزالة الصورة من أداة صندوق الصورة تكتب التعليمة التالية :

```
PictureBox1.Image = null;
```

القيمة **null** تعني لاشيء .

تعريف أداة الشريط التقدمي ProgressBar:



يحدد مقدار العمل المنجز من العمل الكلي .

من اهم خصائص الأداة ProgressBar

١ - الخاصية Maximum :

تحدد القيمة العظمى لمؤشر الأداة . تكتب برمجياً كالتالي :

```
progressBar1.Maximum = 100;
```

٢- الخاصية Minimum :

تحدد القيمة الدنيا لمؤشر الأداة . تكتب برمجياً كالآتي :

progressBar1.Minimum = 0;

٣- الخاصية Value :

تحدد القيمة الحالية لمؤشر الأداة وهذه القيمة محصورة بين القيمة الدنيا Minimum والقيمة العليا Maximum .

٤- الخاصية Step :

تحدد قيمة الخطوة التي ستتغير بها الخاصية Value .

تعريف أداة المؤقت Timer:

تسمح بتكرار تنفيذ عمل معين عند مرور فترة زمنية معينة ولاتوضع هذه الأداة على أرضية النموذج وإنما تظهر اسفل النموذج ولاتظهر اثناء التنفيذ .

من اهم خصائص الأداة ProgressBar

١- الخاصية Interval :

تحدد الفترة الزمنية اللازم مرورها لتكرار تنفيذ عمل المؤقت وتقدر قيمتها بالملي ثانية .
حيث كل 1 ثانية تساوي 1000 ملي ثانية .

مثال :

إذا كانت قيمة الخاصية Interval تساوي 3000 هذا يعني ان الزمن اللازم مروره لتكرار تنفيذ العملية هو 3 ثوان .

٢- الخاصية Enabled :

تحدد فيما إذا كانت الأداة مفعلة ام لا . لها قيمتان :

True : يكون المؤقت مفعلاً .

False : يكون المؤقت غير مفعّل .

اجراء حدث المؤقت الافتراضي Tick

إن الحدث الافتراضي لهذه الأداة هو الحدث **Tick** الذي ينفذ مجموعة من التعليمات الخاصة بالمؤقت عند مرور فترة زمنية معينة محددة بالخاصية **Interval** .

الشكل العام له :

```
Private void TimerName_Tick (object sender , EventArgs e )
```

```
{  
    Statements ;  
}
```

تعريف أداة استعراض المجلدات FolderBrowserDialog :

تستخدم لعرض مربع حوار استعراض المجلدات وتضاف الى المشروع من تبويب **Dialog** من صندوق الأدوات .

من اهم خصائص وإجراءات الأداة FolderBrowserDialog

١- الخاصية RootFolder :

تحدد جذر شجرة المجلدات التي سيتم التعامل معها . من قيمها :

Windows و **MyPicture** و **Desktop** و **MyMusic** وغيرها .

٢- الخاصية SelectedPath :

تحدد مجلدًا ما بشكل تلقائي عند فتح مربع حوار استعراض المجلدات وتكون قيمتها (مسار) وعند تغيير المجلد المحدد من مربع حوار استعراض المجلدات تصبح قيمة هذه الخاصية مسار المجلد المحدد .

٣- الخاصية ShowNewFolderButton :

لعرض زر انشاء مجلد جديد ضمن مربع حوار استعراض المجلدات .

٤- الاجراء ShowDialog() :

يستخدم لفتح مربع حوار استعراض المجلدات .

ويكتب برمجياً كالتالي :

```
FolderBrowserDialog1.ShowDialog();
```

بما ان مربع الحوار يحوي الزرين OK و Cancel فلا بد من معرفة أي هذه الازرار تم نقره لذلك يستخدم الصف DialogResult في عبارة شرطية كالتالي :

```
If ( folderBrowserDialog1.ShowDialog() == DialogResult.OK )
{
    textBox1.Text = folderBrowserDialog.SelectedPath;
}
```

شرح التعليمة :

إذا كان الزر الذي تم نقره هو الزر OK عندها يسند مسار المجلد الذي تم تحديده من مربع حوار استعراض المجلدات FolderBrowserDialog1 الى الأداة textBox1 .

أو :

```
If ( folderBrowserDialog1.ShowDialog() == DialogResult.Cancel )
{
    textBox1.Text = "" ;
}
```

شرح التعليمة :

إذا كان الزر الذي تم نقره هو الزر Cancel عندها يسند قيمة فارغة الى الأداة textBox1 للدلالة على انه لم يتم تحديد مجلد .

تعريف الصف DialogResult : يستخدم في مختلف أنواع مربعات الحوار منها مربع حوار الرسالة MessageBox() لمعرفة أي من الازرار تم نقرها في مربع الحوار .

تعريف أداة استعراض الملفات OpenFileDialog :

تستخدم لعرض مربع حوار استعراض الملفات وتضاف الى المشروع من تبويب Dialog من صندوق الأدوات حيث يمكن تحديد ملف أو عدة ملفات .

من أهم خصائص وإجراءات الأداة OpenFileDialog

١- الخاصية MultiSelect:

تستخدم للسماح بتحديد عدة ملفات. ولها قيمتان :

True :	تسمح بتحديد عدة ملفات معاً .
False :	لا تسمح بتحديد أكثر من ملف واحد .

٢- الخاصية InitialDirectory:

لتحديد المجلد الافتراضي الذي يعرض محتواه عند فتح مربع حوار استعراض الملفات .

مثال :

عند كتابة المسار ("C:\\Ramy") يتم الانتقال الى المجلد (Ramy) الموجود في السوافة (C) لعرض مجلداته وملفاته .

٣- الخاصية Filter:

تستخدم لترشيح أنواع الملفات المطلوب عرضها في مربع حوار استعراض الملفات .

مثال :

لعرض ملفات الصور فقط في مربع حوار فتح الملفات تُكتب التعليمة الآتية :

```
OpenFileDialog1.filter = "image files|.jpg;*.png;*.icon;*.gif;";
```

او كتابة العبارة ("image files|.jpg;*.png;*.icon;*.gif;") مباشرة في الخاصية Filter من نافذة الخصائص أثناء تصميم البرنامج .

٤- الاجراء ShowDialog() :

يستخدم لفتح مربع حوار استعراض الملفات .

ويكتب **برمجياً** كالتالي :

```
OpenFileDialog1.ShowDialog() ;
```

٥- الخاصية FileName :

تستخدم للحصول على اسم ومسار الملف المحدد ضمن أداة مربع حوار فتح الملفات .

وتكتب **برمجياً** كالتالي :

```
string Ramy = openFileDialog1.FileName;
```

شرح التعليمة :

تخزين اسم ومسار الملف المحدد من أداة مربع حوار فتح الملفات في المتحول النصي Ramy .

٦- الخاصية FileNames :

تستخدم للحصول على أسماء ومسارات الملفات المحددة ضمن أداة مربع حوار فتح الملفات .

وتكتب **برمجياً** كالتالي :

```
List <string> lstFiles = new List <string> () ;  
lstFiles.AddRange(openFileDialog1.FileNames);
```

شرح التعليمة :

إضافة أسماء ومسارات الملفات المحددة من مربع حوار فتح الملفات openFileDialog1 الى اللائحة lstFiles .

الصف Directory

تعريف الصف **Directory** : يستخدم للتعامل مع العمليات على المجلدات كعملية انشاء مجلد وحذفه ونقله والحصول على أسماء ملفاته ومجلداته الفرعية وغيرها من العمليات .

يستدعى هذا الصف من المكتبة **System.IO** لذلك **يجب إضافة** هذه المكتبة الى المكتبات داخل المشروع باستخدام العبارة **using** على الشكل التالي :

```
using System.IO;
```

إجراءات الصف Directory

Directory.CreateDirectory(path)	انشاء مجلد جديد
Directory.Delete(path)	حذف مجلد فارغ
Directory.Delete(path,true)	حذف مجلد غير فارغ
Directory.Exist(path)	فحص هل المجلد موجود ام لا ويعيد قيمة منطقية True / False
Directory.GetCreationTime(path)	الحصول على تاريخ انشاء المجلد
Directory.GetDirectories(path)	الحصول على أسماء المجلدات الفرعية ضمن المجلد
Directory.Move(path1,path2)	نقل المجلد المحدد بالمسار path1 الى المجلد المحدد بالمسار path2
Directory.GetFiles(path)	الحصول على مسارات وأسماء الملفات في المجلد

مثال :

```
if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)  
{  
    string path = folderBrowserDialog1.Selectedpath;  
    listBox1.Items.AddRange (Directory.GetFiles(Path));  
}
```

شرح التعليمات :

عند عرض مربع حوار استعراض المجلدات ونقر الزر **OK** تضاف أسماء ومسارات ملفات المجلد المحدد من مربع حوار استعراض المجلدات الى لائحة البنود **listBox1** .

الصف Path

تعريف الصف Path : يستخدم للتعامل مع الأدلة أي مسارات المجلدات والملفات.
يستدعى هذا الصف من المكتبة System.IO لذلك **يجب إضافة** هذه المكتبة الى مكتبات المشروع .

إجراءات الصف Path

١ – الاجراء **Path.GetFileName(path)** :

يستخدم للحصول على اسم المجلد او الملف الموجود في نهاية الدليل **path** .

مثال :

```
textBox1.Text = Path.GetFileName("D:\\Ramy\\Almasalmh\\Pro");
```

تنفيذ السطر البرمجي السابق سيعيد القيمة "Pro" ويسننها الى أداة مربع النص textBox1 .

ملاحظة :

نلاحظ أن "Pro" هو مجلد لانه لا يحوي على لاحقة .

مثال ٢ :

```
textBox1.Text = Path.GetFileName("D:\\Ramy\\Almasalmh\\Pro.jpg");
```

تنفيذ السطر البرمجي السابق سيعيد القيمة "Pro.jpg" ويسننها الى أداة مربع النص textBox1 .

ملاحظة ٢ :

نلاحظ أن "Pro" هو ملف لانه يحوي على اللاحقة **.jpg** وهي من نوع صورة .

٢ – الاجراء **Path.GetFileNameWithoutExtension(path)** :

يستخدم للحصول على اسم المجلد او الملف الموجود في نهاية الدليل **path** دون امتداد (لاحقة) الملف.

مثال :

```
textBox1.Text = Path.GetFileName("D:\\Ramy\\Almasalmh\\Pro.jpg ");
```

تنفيذ السطر البرمجي السابق سيعيد القيمة النصية "Pro" ويسننها الى صندوق النص textBox1 .

٣ – الاجراء **Path.Combine (path , str1 ,str2 ,....)** :

يستخدم للجمع بين دليل (مسار) وعدة سلاسل نصية ويوضع بينها الرمز \ لتشكيل دليل منها .

مثال :

```
textBox1.Text = Path.Combine("D:\\Ramy\\Almasalmh , C# , Test1");
```

وتظهر نتيجة الكود البرمجي السابق في أداة مربع النص كالآتي :

C:\Ramy\Almasalmh\C#\Test1

نلاحظ انه تم جمع المسار مع السلاسل النصية ووضع بينها الرمز \ .

٤ - الاجراء `Path.GetDirectoryName(path)` :

يستخدم للحصول على الدليل (المسار) للمجلد او الملف المحدد في الدليل `path`.

مثال :

```
textBox1.Text = Path.GetDirectoryName("D:\\Ramy\\Almasalmh\\Pro");
```

تنفيذ السطر البرمجي السابق سيعيد القيمة " D:\\Ramy\\Almasalmh " ويسننها الى أداة مربع النص textBox1 .

مثال ٢ :

```
textBox1.Text = Path.GetFileName("D:\\Ramy\\Almasalmh\\C#\\Pro.jpg");
```

تنفيذ السطر البرمجي سيعيد القيمة " D:\\Ramy\\Almasalmh\\C#" ويسننها الى أداة مربع النص textBox1 .

٤ - الاجراء `Path.GetPathRoot(path)` :

يستخدم للحصول على جذر الدليل (المسار) للمجلد او الملف المحدد في الدليل `path`.

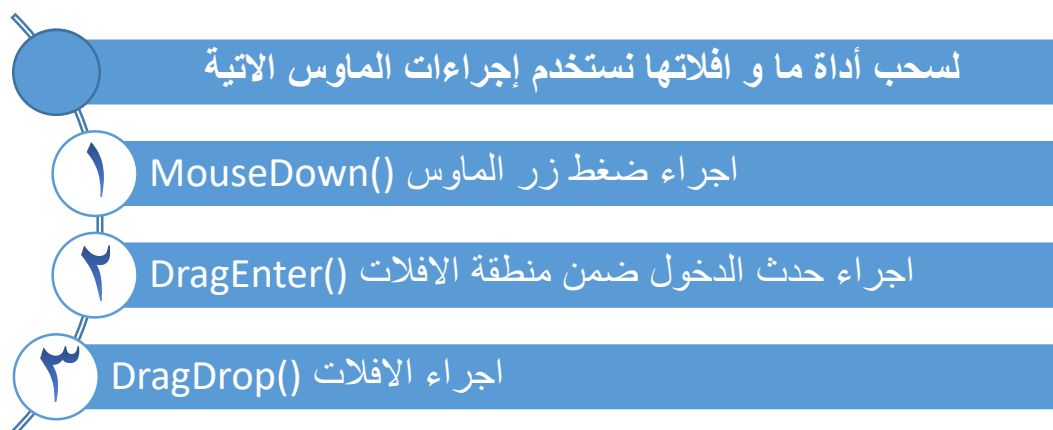
مثال :

```
textBox1.Text = Path.GetDirectoryName("D:\\Ramy\\Almasalmh\\Pro");
```

تنفيذ السطر البرمجي سيعيد القيمة " D:\\ " ويسننها الى أداة مربع النص textBox1 .

السحب والافلات Drag And Drop

السحب والافلات `DragDrop` : هي عملية الضغط على احد ازرار الماوس والاحتفاظ به مضغوطاً وتحريك الماوس ثم تحرير زر الماوس المضغوط .



١- اجراء ضغط زر الماوس (MouseDown) :

تتفد شيفرة هذا الاجراء عند ضغط احد ازرار الماوس .

الشكل العام له :

```
private void objectName_MouseDown ( object sender , MouseEventArgs e )
{
    Statements ;
}
```

هام :

تعريف الوسيط e : هو وسيط من نوع صف المعطيات **MouseEventArgs** يستخدم للوصول الى المعلومات التي تخص الماوس من حيث موقع مؤشر الماوس (**e.X , e.Y**) وزر الماوس المضغوط (**Button**) والمعطيات (**Data**) التي يحملها مؤشر الماوس اثناء السحب .

ملاحظات :

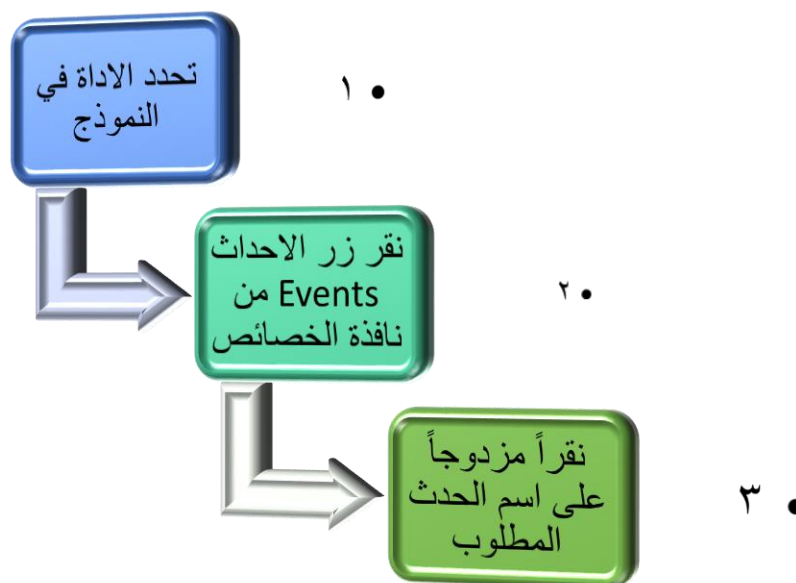
١ - **ObjectName** هو اسم الاداة المراد سحب المعطيات منها والتي سيتم تطبيق حدث ضغط زر الماوس عليها .

٢ - المقصود بـ (**e.X , e.Y**) هو موقع مؤشر الماوس حيث :

X : هو موقع مؤشر الماوس افقياً على اعتبار أن الشاشة مقسومة على مستوي احداثيات و أن العد يبدأ من الزاوية العليا اليسرى للنموذج .

Y : هو موقع مؤشر الماوس عامودياً على اعتبار أن الشاشة مقسومة على مستوي احداثيات و أن العد يبدأ من الزاوية العليا اليسرى للنموذج .

طريقة كتابة شيفرة إجراء الحدث للأداة



مثال :

```
private void pictureBox1_MouseDown ( object sender , MouseEventArgs e )
{
    if ( e.Button == MouseButton.Left )
        pictureBox1.DoDragDrop ( pictureBox1.Image , DragDropEffects.All );
}
```

شرح التعليمات :

اختبار فيما اذا كان الزر المضغوط للماوس هو الزر الايسر للماوس باستخدام الخاصية `Button` للوسيط `e`.
استخدام القائمة `MouseButtons` التي تحوي على مجموعة ازرار الماوس الثلاث (`Left , Middle, Right`) للمقارنة مع زر الماوس المضغوط.
استخدام الاجراء `DoDragDrop` للبدء بعملية سحب صندوق الصورة `pictureBox1` وتحديد المعطيات المطلوب سحبها وتضمينها في وسيط حدث الماوس `e` حيث حددت الخاصية `Image` لصندوق الصورة وللسماع بتطبيق العمليات كافة على المعطيات المسحوبة من القائمة `DragDropEffects` حيث تم تحديد الخيار `All`.

ملاحظة :

إن الشكل العام السابق هو ثابت لكل الأدوات مع اختلاف في اسم الأداة ونوع المعطيات المراد سحبها اما نص او صورة .
حيث في المثال السابق تم سحب صورة من صندوق الصورة لذلك لناخذ أدوات اكثر لفهم الصورة بشكل كامل كالتالي :
اذا كان السحب من أداة مربع النص `TextBox` نوع البيانات الذي سيسحب منه هو `Text` لذلك سيصبح الشكل العام للاجراء كالتالي :

```
private void TextBox1_MouseDown ( object sender , MouseEventArgs e )
{
    if ( e.Button == MouseButton.Left )
        TextBox1.DoDragDrop ( TextBox1.Text , DragDropEffects.All );
}
```

و اذا كان السحب من أداة لائحة البنود `listBox1` نوع البيانات الذي سيسحب منه هو `Items` لذلك سيصبح الشكل العام للاجراء كالتالي :

```
private void listBox1_MouseDown ( object sender , MouseEventArgs e )
{
    if ( e.Button == MouseButton.Left )
        listBox1.DoDragDrop ( listBox1.Items , DragDropEffects.All );
}
```

ملاحظة ٢ :

في اجراء ضغط زر الماوس (MouseDown) يجب مراعاة كتابة نوع صف المعطيات e MouseEventArgs بدلاً من EventArgs في رأس الاجراء كما هو **موضح** في المثال التالي :

```
private void listBox1_MouseDown ( object sender , MouseEventArgs e )
{
    if ( e.Button == MouseButton.Left )
        listBox1.DoDragDrop ( listBox1.Items , DragDropEffects.All );
}
```



٢- اجراء حدث دخول منطقة الإفلات (DragEnter) :

تتفد شيفرة هذا الاجراء عند دخول مؤشر الماوس منطقة الإفلات اثناء سحب المعطيات .

والشكل العام له :

```
Private void ObjectName_DragEnter ( object sender , DragEventArgs e )
{
    Statements ;
}
```

مثال :

```
Private void pictureBox1_DragEnter ( object sender , DragEventArgs e )
{
    e.Effect = DragDropEffects.Move ;
}
```

شرح التعليمات :

- ✓ إن نوع معطيات الوسيط e هو الصف MouseEventArgs وهو صف وسطاء حدث السحب والافلات التي استخدم منها الخاصية Effect .
- ✓ استخدام قائمة أفعال السحب والافلات DragDropEffects واختيار فعل النقل Move منها .

ملاحظة ١ :

إن الشكل العام في هذا المثال مع الشيفرة التي بداخله ثابت لكل أداة نريد تطبيق السحب والافلات عليها وفي حال طلب كتابة كود إضافي يتم تنفيذه عند دخول المؤشر لمنطقة الإفلات يكتب هذا الكود اسفل الكود السابق مباشرةً .

ملاحظة ٢ :

في اجراء دخول منطقة الإفلات (DragEnter) يجب مراعاة كتابة نوع صف المعطيات e DragEventArgs بدلاً من EventArgs في رأس الاجراء كما هو **موضح** في المثال التالي :

```
private void pictureBox1_DragEnter ( object sender , DragEventArgs e)
{
    e.Effect = DragDropEffects.Move ;
}
```

٣- اجراء حدث دخول منطقة الإفلات (DragDrop) :

تتفد شيفرة هذا الاجراء عند إفلات المعطيات أو الأداة المسحوبة .
يمكن أن تكتب فيه التعليمات التي تحقق الهدف من عملية السحب والافلات .

الشكل العام له :

```
private void ObjectName_DragDrop ( object sender , DragEventArgs e)
{
    Statements ;
}
```

مثال :

إجراء الإفلات في منطقة النموذج ونقل أداة PictureBox1 الى نقطة الإفلات في النموذج :

```
private void Form1_DragDrop ( object sender , DragEventArgs e)
{
    pictureBox1.Location = this.PointToClient( new point(e.X,e.Y));
}
```

تعرف الاجراء PointToClient : يستخدم للقيام بتحويل إحداثيات نقطة الإفلات (e.X,e.Y) ضمن الشاشة الاحداثيات ضمن منطقة النموذج .

ملاحظة ١ :

في لغة VC# يتم استخدام الاجراء PointToClient لأن مبدأ الاحداثيات هو الزاوية العليا اليسرى للشاشة وليس للنموذج اثناء عملية السحب والافلات .

ملاحظة ٢ :

لإتمام عملية السحب والافلات يجب كتابة تعليمة السماح بالافلات في المكان المناسب بحسب السؤال المطلوب .

مثال :

```
private void Form1_Load ( object sender , EventArgs e)
{
    pictureBox1.AllowDrop = true ;
}
```

تعريف لبعض المصطلحات السابقة

- ✓ القائمة **MouseButtons** : تحوي مجموعة ازرار الماوس (**Left** , **Middle** , **Right**) وتستخدم للمقارنة مع ازرار الماوس المضغوط .
- ✓ الاجراء **DoDragDrop** : يستخدم للبدء بعملية السحب وتحديد المعطيات المطلوب سحبها وتضمينها في وسيط حدث الماوس **e** .
- ✓ الصف **DragDropArgs** : هو صف وسطاء حدث السحب والافلات ، والذي يستخدم منه الخاصية **Effect** .
- ✓ القائمة **DragDropEffects** : قائمة أفعال السحب والافلات .

بفضل الله تعالى

تم الانتهاء من الوحدة الاولى



للاستفسار على الرابط التالي :
<https://www.facebook.com/ramy.almasalmh.3/>

الوحدة الثانية لغة الاستعلامات البنوية SQL

ما الهدف من قواعد المعطيات ؟

الهدف هو تنظيم المعطيات ، واختصار الوقت والجهد أثناء التعامل معها .

تعريف قاعدة المعطيات Database :

هي مجموعة من المعطيات غير المكررة ، تحفظ ضمن جدول أو مجموعة جداول مرتبطة مع بعضها البعض بروابط لها أنواع متعددة تمكننا من استرجاع المعطيات والبحث فيها واستخلاص المعلومات منها بشكل سهل وسريع .

عدد أنواع قواعد المعطيات :

١ - قواعد معطيات تحفظ في مخدّم قواعد المعطيات .

٢ - قواعد معطيات تحفظ كملف على قرص الذاكرة .

١- قواعد معطيات تحفظ في مخدّم قواعد المعطيات :

لا يمكن الدخول الى قاعدة المعطيات المحفوظة في مخدّم قواعد معطيات الا عن طريق الاتصال مع المخدم باستخدام اسم المستخدم وكلمة المرور .

مثال :

١ - **SQL Server** : يكون امتداد قاعدة المعطيات عند حفظها **mdf** .

٢ - **Oracle** : يكون امتداد قاعدة المعطيات عند حفظها **dmp** .

٢- قواعد معطيات تحفظ كملف على قرص الذاكرة :

لاتحفظ قاعدة المعطيات على المخدم ، انما تحفظ كملف يفتح مباشرة واستخلاص المعطيات من قاعدة المعطيات (اذا كان الملف محمياً لايمكن الدخول اليه الا بإدخال كلمة المرور) .

مثال :

١ - **Access** : يكون امتداد قاعدة المعطيات عند حفظها **mdb** .

ملاحظة هامة :

يجب حفظ أنواع قواعد المعطيات مع اللاحقة المناسبة لكل نوع .

تعريف نظام إدارة قواعد المعطيات DBMS :

هو النظام المسؤول عن عملية إدارة قواعد المعطيات وضمان الامن والسلامة لها ، حيث يتم من خلاله انشاء عدد من المستخدمين ، وإعطاء كلّ منهم صلاحيات محددة ، للسماح بالوصول الى المعطيات والاستفادة منها ، أو إجراء العمليات عليها (كإضافة و الحذف و التعديل) وذلك بإنشاء ملفات فيزيائية في الذاكرة **RAM** او في الذاكرة الظاهرية على القرص الصلب للتعامل مع المعطيات المحفوظة في قاعدة المعطيات .

تعريف الجداول Tables :

الجدول هي المكون الأساسي لقاعدة المعطيات ، حيث يحوي الجدول مجموعة من الاعمدة والاسطر تخزن المعطيات غير المكررة فيها حيث يسمى كل عمود (Column) في الجدول بالحقول (Field) ويحدد نوع المعطيات (Data Type) لكل حقول عند انشاء الجدول ويسمى كل سطر (Rows) في الجدول بالسجل (Record) ويتكون السجل من قيم حقول الجدول الموجودة في سطر واحد وبالتالي الجدول يتكون من مجموعة سجلات ومجموعة الجداول تكون قاعدة المعطيات .

ماهي المعلومات اللازم تحديدها لكل حقول من حقول جدول عند انشائه في قاعدة المعطيات ؟

عند انشاء أي جدول في قاعدة المعطيات تحدد المعلومات التالية في كل حقول من حقول الجدول :

- اسم الحقول (العمود) Column Name .
- نوع أو نمط الحقول (Data Type) .
- تحديد إمكانية / عدم إمكانية أن تكون قيمة الحقول فارغة Allow Nulls .

من أنماط الحقول Data Type نذكر منها

int : لتخزين عدد صحيح .
float : لتخزين عدد حقيقي .
image : لتخزين صورة بشكل ثنائي ، والحجم الأعلى المسموح تخزينه 2GB .
date : لتخزين المعطيات على شكل تاريخ .
money : لتخزين المعطيات بصيغة عملة .
ntext : لتخزين المعطيات بشكل نص يحوي عدد كبير من المحارف .
varchar(50) : لتخزين معطيات بشكل نص طوله الاعظمي 50 حرف .
varchar(Max) : لتخزين معطيات بشكل نص طوله الاعظمي Max حرف .
nvarchar(50) : لتخزين معطيات بشكل نص طوله الاعظمي 50 حرف مع إمكانية ادخال قيم بلغات أخرى كالعربية .
nvarchar(Max) : لتخزين معطيات بشكل نص طوله الاعظمي Max حرف مع إمكانية ادخال قيم بلغات أخرى كالعربية .

تعريف المفتاح الرئيسي Primary Key ؟

هو رقم يستخدم لتمييز السجلات عن بعضها ويجب تحديد قيمة هذا المفتاح في كل سجل من السجلات و **لايسمح** أن تكون قيمة **المفتاح الرئيسي** فارغة Null .

ما الحاجة من وجود المفتاح الرئيسي Primary Key ؟

بما ان الجدول يحوي سجلات غير مكررة اذ لا يمكن أن يتكرر السجل نفسه في جدول واحد لذلك يجب تعريف مفتاح رئيسي **Primary Key** .

تعريف المفتاح الاجنبي Foreign Key ؟

هو مفتاح ينتج عن علاقة جدولين مع بعضهما حيث يكون المفتاح الرئيسي لأحد هذين الجدولين مفتاحاً أجنبياً للجدول الآخر ويأخذ المفتاح الأجنبي قيمة من الجدول الذي يكون فيه مفتاحاً رئيسياً .

ملاحظات هامة

امتدادات قواعد المعطيات :

Access هو .mdb	Oracle هو .dmp	SqlServer هو .mdf
----------------	----------------	-------------------

- الجدول **لا يمكن** أن يحوي الا مفتاحاً رئيسياً واحداً.
- يعد نظام إدارة قواعد المعطيات **DBMS** صلة الوصل وواجهة التخاطب بين المستخدم وقاعدة المعطيات .
- يتكون الجدول من **أعمدة واسطر** تخزن داخله معطيات **غير مكررة** .
- يسمى كل سطر **Row** في الجدول بالسجل **Record** .
- يسمى كل عمود **Column** في الجدول بالحقل **Field** .
- لا يمكن** لقيمة المفتاح الرئيسي أن تكون فارغة **Null** .
- يتكون السجل من قيم **حقول** الجدول الموجودة في سطر واحد .
- يتألف **الجدول** من مجموعة **سجلات** .
- تتألف **قاعدة المعطيات** من مجموعة **جداول** .
- يمكن للجدول أن يحوي أكثر من مفتاح اجنبي وهو ينتج عن علاقة جدولين مع بعضهما بأي شكل من الاشكال العلاقات .

فيما تستخدم العلاقات بين الجداول ؟

تستخدم العلاقات للحصول على جداول أخرى تحوي معطيات مستخلصة من الجداول المرتبطة مع بعضها بهذه العلاقات .

عدد اشكال العلاقات بين الجداول :

١ - علاقة واحد الى واحد One To One .
٢ - علاقة واحد الى كثير One To Many .
٣ - علاقة كثير الى كثير Many To Many .

علاقة واحد الى واحد **One To One** :

يرتبط كل سجل من الجدول الأول **بسجل واحد** فقط من الجدول الثاني وكل سجل من الجدول الثاني مرتبط **بسجل واحد** فقط من الجدول الأول .

مثال :

العلاقة بين جدول أسماء رؤساء الاقسام و جدول الأقسام حيث أن لكل مدير قسم واحد فقط ، ولكل قسم رئيس واحد فقط .

لذلك يجب أن يحوي الجدول الثاني مفتاحاً يمثل المفتاح الرئيسي للجدول الأول أو بالعكس أي يجب أن يكون في جدول رؤساء الأقسام حقل يمثل رقم القسم أو أن يكون في جدول الأقسام حقل يمثل رقم رئيس القسم .

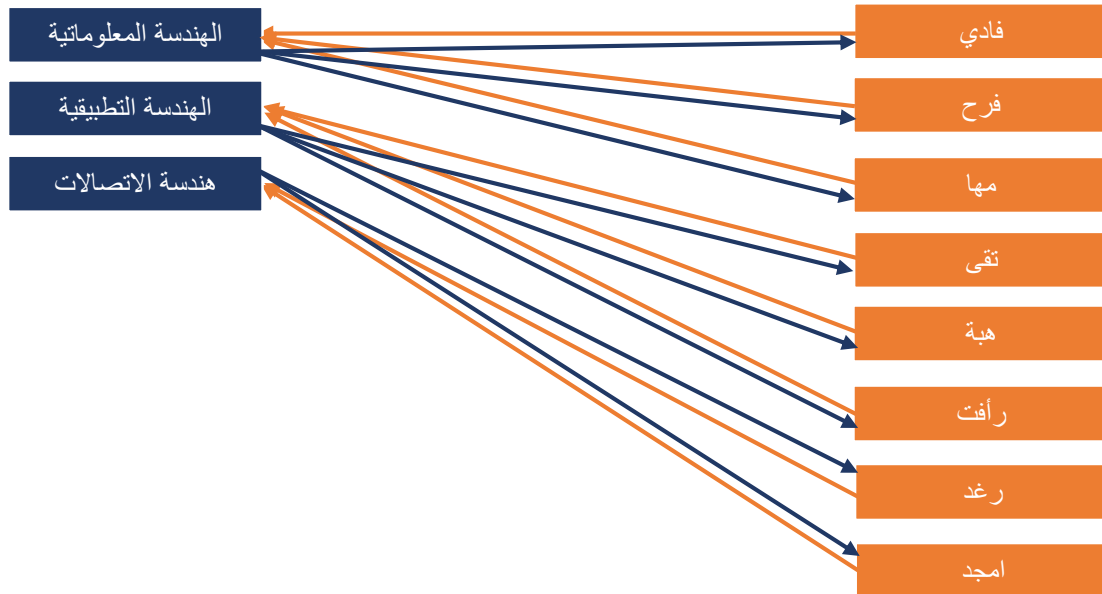


علاقة واحد الى كثير One To Many :

يرتبط كل سجل من الجدول الأول **بسجل واحد أو أكثر** من الجدول الثاني وكل سجل من الجدول الثاني يرتبط **بسجل واحد فقط** من الجدول الأول .

مثال :

العلاقة بين جدول الكليات وجدول الطلاب ، كل كلية تحوي أكثر من طالب بينما كل طالب مسجل بكلية واحدة فقط . في هذه الحالة يجب أن يحوي الجدول الثاني مفتاحاً اجنبياً يمثل المفتاح الرئيسي لجدول الكليات حيث يخزن في سجل كل طالب رقم الكلية التي قام بالتسجيل فيها .

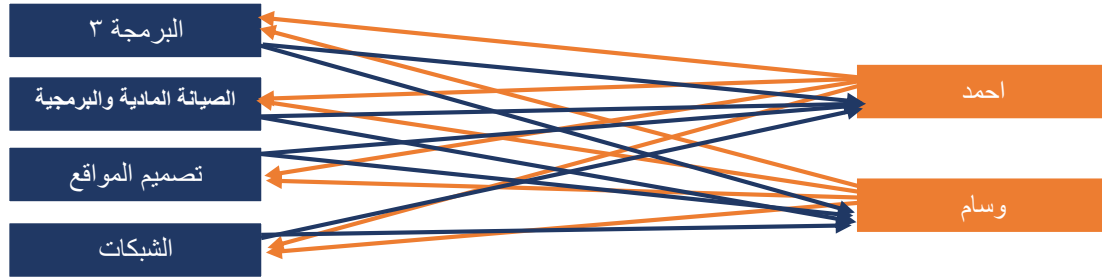


علاقة كثير الى كثير Many To Many :

يرتبط كل سجل من الجدول الأول **بأكثر** من سجل في الجدول الثاني وكل سجل من الجدول الثاني يرتبط **بأكثر** من سجل في الجدول الأول .

مثال :

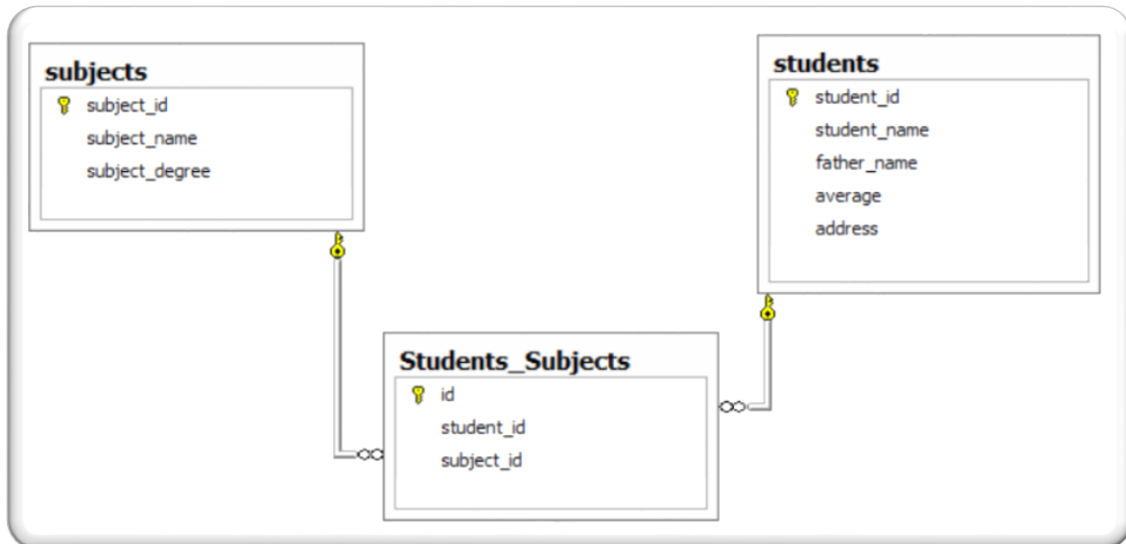
العلاقة بين جدول الطلاب وجدول المواد حيث كل طالب يدرس أكثر من مادة وكل مادة يدرسها أكثر من طالب .



ملاحظة هامة :

في حالة كثير الى كثير **يجب** إنشاء جدول ثالث وسيط يحوي مفتاحاً مركباً بحيث يتألف من مفتاحين أجنبيين هما المفتاحين الرئيسيين لجدولي الطلاب والمواد ويحوي معلومات مرتبطة بكل من الجدولين مثل علامة الطالب.

أي يتم تحويل علاقة الكثير لكثير **ManyToMany** الى علاقتي واحد لكثير **OneToMany** بين جدول الطلاب والجدول الوسيط وجدول المواد والجدول الوسيط كالتالي :



الجدول **Students_Subjects** هو الجدول الوسيط نلاحظ بين الجدولين والجدول الوسيط علاقة واحد لكثير **OneToMany** من خلال العلاقتين تتشكل لدينا علاقة كثير لكثير **ManyToMany** .

ملاحظات هامة

- ❖ العلاقة بين جدول الطلاب وجدول المواد هي علاقة **كثير لكثير** لأن كل طالب يرتبط بأكثر من مادة وكل مادة مرتبطة بأكثر من طالب .
- ❖ العلاقة بين جدول الطلاب وجدول البطاقة الامتحانية هي علاقة **واحد لواحد** لأن لكل طالب بطاقة واحدة والبطاقة تعود لطالب واحد فقط (كما تعمم هذه الحالة على العلاقة بين الطالب ورقم الاكتتاب)
- ❖ العلاقة بين جدول الطلاب وجدول المدارس **واحد لكثير** لأن كل طالب مرتبط بمدرسة فقط وكل مدرسة مرتبطة بالكثير من الطلاب .

ما المتطلبات الواجب توافرها في الحاسوب لتثبيت برنامج SqlServer2008 عليه ؟

- ✓ ذاكرة بحجم 1GB على الأقل ويفضل ان تكون بسعة 4GB ليعمل البرنامج بشكل جيد.
- ✓ سرعة وحدة المعالجة المركزية CPU على الأقل 1.0GHz في حال كان المعالج X86 و 1.4GHz في حال كان المعالج X64 ويفضل أن يكون لكلا النوعين 2GHz ليعمل البرنامج بأداء جيد.

خطوات انشاء قاعدة معطيات في برنامج SQL Server Management Studio

- ✚ فتح برنامج SQL Server Management Studio .
- ✚ النقر على الزر Connect للاتصال مع مخدم قواعد المعطيات .
- ✚ النقر بالزر الأيمن للفأرة على المجلد Databases ثم انقر الامر New DataBase .
- ✚ يدخل اسم قاعدة المعطيات ضمن الحقل DataBase Name ثم ينقر الزر OK فيتم انشاء قاعدة المعطيات بالاسم الذي تم إدخاله .

خطوات انشاء جداول قاعدة معطيات في برنامج SQL Server Management Studio

- ✓ نقر الإشارة + بجوار المجلد DataBase فتظهر قاعدة المعطيات التي تم انشاؤها سابقاً بالإضافة الى قواعد معطيات جاهزة مبنية ضمن مخدم قواعد المعطيات المتصل به .
- ✓ تنقر الإشارة + بجانب قاعدة المعطيات التي تم انشاؤها مسبقاً .
- ✓ النقر بالزر الأيمن للفأرة على المجلد Tables ويختار منه الامر New Table لإنشاء جدول جديد ضمن قاعدة المعطيات .
- ✓ يدخل اسم كل حقل من حقول الجدول ونمط كل حقل وتحديد إمكانية أن تكون قيمة الحقل فارغة ام لا .
- ✓ تحديد حقل من الجدول ليكون مفتاح رئيسي بالنقر على الزر الأيمن للفأرة على السهم بجوار اسم الحقل ويختار منه Set Primary Key فتتم تلقائياً إزالة إشارة الاختيار Allow Nulls لأن المفتاح الرئيسي لا يمكن أن تكون قيمته فارغة .
- ✓ يحفظ الجدول بعد ادخال أسماء وانماط الحقول بضغطة المفاتيح Ctrl + S أو من القائمة File الامر Save Table_1 فيتم ادخال اسم الجدول المطلوب حفظه .
- ✓ تنشأ كل جداول قاعدة المعطيات بالطريقة نفسها .

خطوات انشاء العلاقات بين الجداول في قواعد المعطيات

- ❖ نقر الإشارة + بجانب قاعدة المعطيات المنشأة ثم نقر الزر الأيمن للماوس على المجلد Database Diagram ويختار الخيار New Database Diagram .
- ❖ تظهر رسالة عند انشاء أول مخطط في قاعدة المعطيات فينقر الزر Yes لإنشاء المكونات اللازمة لإنشاء المخطط.
- ❖ يتم تحديد الجداول المطلوب اضافتها الى المخطط من النافذة التي ظهرت من خلال تحديده ونقر الزر Add وفي النهاية Close .

- ❖ لإنشاء العلاقة يوضع المؤشر على المفتاح الرئيسي من الجدول الأول ويسحب حتى الوصول الى الحقل الذي سيكون مفتاحاً اجنياً للجدول الأول في الجدول الثاني فتظهر نافذة تحديد نوع العلاقة .
- ❖ يحدد نوع العلاقة من القائمة المنسدلة أي تحديد اسم المفتاح الأجنبي في الجدول الثاني ثم ينقر **OK** .
- ❖ يحفظ المخطط الذي تم إنشاؤه بضغطة المفاتيح **Ctrl + S** او من القائمة **File** الامر **Save Diagram_1** فيدخل اسم المخطط المطلوب حفظه من ثم **OK** .
- ❖ لفتح المخطط ينقر عليه نقرأ مزدوجاً .
- ❖ لحذف العلاقة بين الجدولين ينقر بالزر الأيمن للماوس على العلاقة ثم ينقر الامر **Delete Realtionships From Database** .

(هام)

عند حفظ قاعدة المعطيات يحفظ ملفان خاصان بقاعدة البيانات المحفوظة وهما

- DatabaseName.mdf** : يمثل النسخة الاصلية لقاعدة المعطيات التي تم حفظها ولها الامتداد **.mdf**.
- DatabaseName_log.ldf** : يمثل النسخة الاحتياطية لقاعدة المعطيات التي تم حفظها وامتدادها **.ldf**.
- حيث أن **DatabaseName** هو اسم قاعدة البيانات التي تم حفظها .
- اما الملف ذو اللاحقة **.ldf** في حال حدوث أي عطب لقاعدة المعطيات الاصلية تسترجع من قاعدة المعطيات الاحتياطية .

خطوات تعديل قاعدة المعطيات

- ١- تنقر الإشارة (+) بجانب قاعدة المعطيات المطلوب تعديل جداولها .
- ٢- تنقر الإشارة (+) بجوار المجلد **Tables** .
- ٣- لتعديل اسم الجدول ينقر بالزر الأيمن للفأرة عليه ويختار الخيار **Rename** ويدخل الاسم المناسب له ثم يضغطة المفتاح **Enter** .
- ٤- لتعديل محتوى الجدول تتبع إحدى الطريقتين :
 - a) ينقر بالزر الأيمن للفأرة على اسم الجدول ويختار الخيار **Design** فيفتح تصميم الجدول وتجرى التعديلات أو الإضافات اللازمة عليه .
 - b) ينقر الإشارة (+) بجوار اسم الجدول المطلوب تعديله فتظهر قائمة المجلدات لإجراء التعديلات اللازمة عليها مثل إضافة حقول جديدة الى الجدول أو تغيير أنماط الحقول التي تم انشاؤها مسبقاً أو تغيير الحقول التي تم اعتبارها مفاتيح أو حذف الحقول من الجدول.
- ٥- بعد الانتهاء من إجراء التعديلات على الجدول يحفظ بإحدى الطرائق السابقة ، فتظهر رسالة تفيد **" لا يمكن حفظ التعديلات على الجدول "** .
- ٦- ينقر الزر **Cancel** لإغلاق النافذة السابقة ، ومن القائمة **Tools** في شريط القوائم ، ثم يختار منها الخيار **Options** .
- ٧- ينقر التبويب **Designers** من القائمة اليسارية من النافذة الظاهرة .
- ٨- يلغى تفعيل الخيار **Prevent Saving Changes That Require Table Re-Creation** بإزالة الإشارة من مربع الاختيار للسماح بحفظ التعديلات على الجداول ثم ينقر **OK** .
- ٩- يحفظ الجدول الذي أجريت التعديلات عليه بإحدى طرائق الحفظ التي سبق ذكرها لحفظ الجداول .

١٠- للتعديل على أي مخطط في القاعدة تنقر الإشارة (+) بجوار المجلد Database Diagram ثم ينقر نقرًا مزدوجاً على اسم المخطط لفتحه وإجراء التعديلات اللازمة عليه ، ويمكن إضافة جدول جديد مبني في قاعدة المعطيات الى المخطط بطريقتين :

(a) النقر على القائمة Database Diagram من شريط القوائم ثم ينقر الزر Add Table .

(b) النقر على الأداة Add Table في شريط الأدوات .

ففي كلتا الحالتين يظهر مربع حوار لإضافة الجدول المطلوب كما الطريقة سابقاً لإضافة الجدول الى المخطط ثم يحفظ كما ذكرنا سابقاً .

١١- لتعديل اسم المخطط ينقر بالزر الأيمن للفأرة عليه ويختار الخيار Rename ويدخل الاسم المناسب ثم يضغط المفتاح Enter .

١٢- لتعديل اسم قاعدة المعطيات ينقر بالزر الأيمن للفأرة عليها ويختار الخيار Rename ويدخل الاسم الجديد لها ثم يضغط المفتاح Enter .

خطوات الحذف من قاعدة المعطيات

لحذف أي (جدول - قاعدة معطيات - مخطط) ينقر الزر الأيمن على اسم (جدول - قاعدة معطيات - مخطط) المطلوب حذفه ويختار الأمر Delete ثم الزر OK لتأكيد العملية ، أو بتحديد (جدول - قاعدة معطيات - مخطط) المطلوب حذفه وضغط الرز Delete من لوحة المفاتيح ثم نقر الزر OK .

عدد طرائق تعبئة معطيات في قاعدة معطيات

(هام) يمكن تعبئة المعطيات في قائمة المعطيات بطريقتين :

١ - إدخال المعطيات مباشرة من خلال البرنامج الذي تم تصميم قاعدة المعطيات فيه كبرنامج Sql Server Management Studio .

٢ - عن طريق الربط مع برنامج صمم بإحدى لغات البرمجة المتقدمة كلغة VC# .

خطوات تعبئة معطيات في قاعدة المعطيات باستخدام SqlServer2008 بالطريقة المباشرة

١- فتح برنامج SQL Server Management Studio .

٢- الاتصال مع مخدم قواعد المعطيات .

٣- تنقر الإشارة (+) بجانب المجلد Databases ثم تنقر (+) بجانب قاعدة المعطيات المطلوب تعبئتها ثم تنقر الزر (+) بجانب المجلد Tables .

٤- النقر بالزر الأيمن للماوس على إحدى جداول قاعدة المعطيات ثم ينقر الأمر Edit Top 200 Rows .

٥- تظهر نافذة إدخال المعطيات في الجدول فتُدخل المعطيات في حقول كل سجل .

لغة الاستعلامات البنوية SQL

(هام (تعريف) : هي لغة تحوي مجموعة من التعليمات التي تهدف للبحث في قاعدة المعطيات عن معطيات معينة توافق الشروط التي يضعها المبرمج في برنامجه ، وكذلك تعديل قاعدة المعطيات أو الإضافة إليها أو الحذف منها ويستخدم برنامج SQL Server Management Studio لكتابة تعليمات الاستعلام عن المعطيات في قاعدة المعطيات .

تعريف الاستعلامات

تستخدم الاستعلامات للتعامل مع المعطيات في قاعدة المعطيات بطريقة سهلة وسريعة ، **(هام)** فالإستعلام : هو طلب يستخدم لإخبار نظام إدارة قواعد المعطيات **DBMS (DataBase Management System)** **(يعرف نطيتو عن ترجمة نظام إدارة قواعد المعطيات بس حطيتو زيادة ثقافة ☺)** عن المعطيات المطلوب الحصول عليها من قاعدة المعطيات المحددة حسب الشروط التي تم تحديدها ضمنه .

عدد وظائف الاستعلام & عدد استخدامات الاستعلام

١- الحصول على المعطيات المطلوبة .	٣- تحديد طريقة عرض المعطيات المستخلصة في جدول.
٢- إضافة أو حذف أو تعديل المعطيات في قاعدة المعطيات	٤ - إضافة أو حذف الجداول من قاعدة المعطيات.

هام :

علل تعد لغة الاستعلامات البنوية من اللغات السهلة المستخدمة لكتابة الاستعلامات **(لأنها لغة قريبة من لغة المستخدم)**.

ملاحظات :

🔗 لغة الاستعلامات البنوية SQL لغة غير حساسة لحالة الاحرف .

مثال : select - SELECT - Select .

🔗 يمكن عدم كتابة الفاصلة المنقوطة في نهاية تعليمة الاستعلام .

كيف يتم انشاء استعلام باستخدام برنامج SQL Server :

يتم انشاء الاستعلام بطريقتين :

١ - بالنقر الزر الأيمن للفأرة على قاعدة المعطيات المطلوب انشاء استعلام عليها وينقر الامر New Query .

٢ - بالنقر على الأداة New Query من شريط الأدوات القياسي .

التعليمة SELECT

تعريف : تستخدم هذه التعليمة لاختيار وعرض المعطيات المحددة من قاعدة المعطيات ، ولتحديد طريقة عرض هذه المعطيات بعد استخلاصها .

الشكل العام لتعليمة الاختيار :

Select * From Table_Name ;

حيث :

***** : تستخدم لاختيار الحقول كافة من الجدول .

From : كلمة محجوزة تستخدم لتحديد اسم الجدول أو الجداول المطلوب استخلاص المعطيات منها .

Table_Name : اسم الجدول أو الجداول المطلوب استخلاص المعطيات منها .

ملاحظة : لإستخلاص حقول معينة دون أخرى تذكر أسماء الحقول مع وضع فواصل بينها بدلاً من (*) .

أمثلة :

(١)

Select * From Doctors ;

لعرض كافة حقول جدول الأطباء .

(٢)

Select Doctor_Id , Doctor_Name From Doctors ;

لعرض معطيات الحقليين (رقم الطبيب ، اسم الطبيب) من جدول الأطباء وبذلك سيتم عرض الجدول فقط بعمودين اثنين الأول هو رقم الطبيب والثاني اسمه .

ملاحظات :

- ❖ لتنفيذ الاستعلام من خلال الضغط على الزر F5 أو بنقر الأداة Excute من شريط الأدوات .
- ❖ لتنفيذ تعليمة واحدة من مجموعة تعليمات يتم تحديد هذه التعليمة ثم الضغط على الزر F5 أو Excute .
- ❖ تظهر نتائج الاستعلامات في نافذة النتائج Results اسفل نافذة الاستعلامات .

العبرة WHERE

تعريف : تستخدم لإضافة شرط الى تعليمة الاختيار .

الشكل العام للتعليمة Select بعد إضافة العبرة Where :

Select * From Table_Name Where Condition ;

Condition : هي الشروط المفروضة على نتيجة الاستعلام حيث يحوي الشرط على عمليات المقارنة و التوابع المنطقية .

جدول عمليات المقارنة المستخدمة مع العبرة الشرطية Where

العملية	المعنى	مثال
=	يساوي	Select * From Teachers Where Teacher_Name = 'Ola Alsharo3' ;
<	اصغر	Select * From Students Where Average < 85 ;
>	اكبر	Select * From Students Where Average > 85 ;
<=	اصغر أو يساوي	Select * From Students Where Average <= 85 ;
>=	اكبر أو يساوي	Select * From Students Where Average >= 85 ;
<>	لا يساوي	Select * From Students Where Average <> 85 ;
Between	ضمن المجال	Select * From Students Where Average Between 60 and 90 ;

جدول لبعض التوابع المنطقية المستخدمة مع العبرة الشرطية Where

التابع المنطقي	المعنى	مثال
AND	و	Select * From Teachers Where (Teacher_Name='Majdolen Mutlek') AND (Teacher_Name='Nour Mhamed');
OR	أو	Select * From Teachers Where (Teacher_Name='Imad Almstrehe') OR (Teacher_Name='Waleed Ghazal');
NOT	نفي	Select * From Students Where NOT (Teacher_Name='Razan');

العبارة Order By

تعريف : تستخدم لترتيب عرض السجلات في جدول النتائج حسب حقل أو حقول معينة .

الشكل العام للعبارة Order By

Select * From table_name order by fieldName1 , fieldName2 , ... , fieldNameN ;

حيث :

field_name1 , field_name2 , ... : هي أسماء الحقول التي ستترتب سجلات الجدول حسبها ، حيث يتم ترتيب أولاً حسب field_name1 فإذا تساوت هذا الحقل لسجلين في جدول فيتم الترتيب حسب قيمة الحقل الثاني field_name2 .
علماً انه يتم ترتيب السجلات تصاعدياً بشكل افتراضي **أما** لترتيبها تنازلياً توضع كلمة **DESC** في نهاية التعليمة .

مثال :

Select * From Doctors Order By Doctor_Name , Address ;

شرح التعليمة:

عرض حقول الجدول Doctors كافة ويتم ترتيب هذا الجدول **تصاعدياً** حسب الحقل Doctor_Name فإذا تساوت قيمة هذا الحقل لسجلين في الجدول يتم ترتيبهما حسب الحقل الثاني وهو Address .

مثال ٢ :

Select * From Students Order By Average Desc ;

شرح التعليمة:

عرض حقول الجدول Students كافة ويتم ترتيب هذا الجدول **تنازلياً** حسب الحقل Average .

التابع () SUM

تعريف : يستخدم لإيجاد مجموع قيم عمود ما من الجدول .

الشكل العام للتابع () SUM

Select SUM(field_name) as alias_name From table_name;

Field_name : اسم الحقل المطلوب جمع قيمه .

as : كلمة مفتاحية تستخدم لعرض المجموع بالاسم الذي بعدها .

alias_name : اسم الحقل الذي سيظهر فيه نتيجة الجمع .

table_name : اسم الجدول المطلوب إيجاد مجموع قيم عمود من أعمدته .

التعليمة INSERT INTO

تعريف : تستخدم لاضافة سجلات جديدة الى الجدول .

الشكل العام للتعليمة INSERT INTO

Insert Into table_name (fieldName1, fieldName2, ... ,fieldNameN) **Values** (Value1 , Value2 , ... , ValueN) ;

حيث :

Table_name : اسم الجدول المطلوب إضافة السجلات اليه .

fieldName1 , fieldName2 , ... : أسماء حقول الجدول المطلوب إضافة قيم جديدة اليها .

Values : القيم الجديدة المطلوب اضافتها الى الحقول المذكورة فمثلاً تضاف القيمة Value1 الى الحقل fieldName1 وكذلك وبالترتيب .

مثال :

Insert Into Teachers (Teachers_id , Teachers_Name , Subject) **Values** (7 , N 'رامي المسالمه' , N 'برمجة') ;

التعليمة UPDATE

تعريف : تستخدم لتعديل سجل أو عدة سجلات في جدول .

الشكل العام للتعليمة UPDATE

Update table_Name SET fieldName1 = value1 , fieldName2 = value2 , ... , fieldNameN = valueN **Where** criteria ;

Table_name : اسم الجدول المطلوب تعديل السجلات فيه .

SET : كلمة مفتاحية تستخدم لتحديد القيم الجديدة المطلوب التعديل بها في الجدول .

fieldName1 , fieldName2 , ... : أسماء حقول الجدول المطلوب تعديل قيمها .

Value1 , Value2 , ... : القيم الجديدة المطلوب التعديل بها في الحقل المذكور قبلها فمثلاً تصبح قيمة الحقل fieldName1 = Value1 وهكذا .

Criteria : تستخدم لتحديد السجلات التي ستعدل قيم حقولها في الجدول .

مثال :

Update Clinic **Set** Address = N 'درا' **Where** Clinic_name=N 'الدكتور احمد المحمد' ;

الشرح :

يتم تعديل الحقل Address في الجدول Clinic للسجل ذو الاسم "الدكتور احمد المحمد" ليصبح "درا" .

التعليمة DELETE

تعريف : تستخدم لحذف سجل أو عدة سجلات في جدول .

الشكل العام للتعليمية DELETE

Delete From table_name Where criteria ;

حيث :

From : كلمة مفتاحية تستخدم لتحديد الجدول المطلوب الحذف منه .

Table_name : اسم الجدول المطلوب حذف السجلات منه .

Where Criteria : شروط تستخدم لتحديد السجلات التي ستحذف من الجدول .

مثال :

Delete From Students Where Student_name = 'حمزة N' ;

الشرح :

يحذف سجل الطالب الذي اسمه " حمزة " من الجدول Students .

التعليمية Create Table

تعريف : تستخدم لبناء جدول جديد واضافته الى قاعدة المعطيات .

الشكل العام للتعليمية Create Table

Create Table Table_Name (Column1_name Column1_type Column1_Constraints ,
Column2_name Column2_type Column2_Constraints , ...) ;

Table_name : اسم الجدول المطلوب إنشاؤه وحفظه ضمن قاعدة المعطيات .

Column1_name : اسم الحقل الأول من الجدول المطلوب إنشاؤه .

Column2_Type : نمط معطيات الحقل الأول .

Column1_Constraints : القيود المفروضة على الحقل الأول (ان يكون مفتاحاً رئيسياً للجدول ، أو إمكانية ان تكون قيمته فارغة أم لا) .

طبعاً هذه ينطبق على كافة الحقول في الجدول المطلوب إنشاؤه ، الى الانتهاء من انشاء كل حقل من حقول الجدول .

مثال :

Create Table Players (Players_id int primary key , Players_name nvarchar(50) not null ,
Players_team nvarchar(50)) ;

الشرح :

يتم انشاء جدول جديد اسمه Players في قاعدة المعطيات يحوي الحقول التالية :

Player_id : من النمط int وهو مفتاح رئيسي للجدول .

Players_name : من النمط `nvarchar(50)` ولايسمح لقيمة هذا الحقل أن تكون فارغة .

Players_team : من النمط `nvarchar(50)` ويسمح لقيمتها أن تكون فارغة .

ملاحظات :

✚ للتأكد من انشاء الجدول وحفظه في قاعدة المعطيات نفتح قاعدة المعطيات ثم إشارة (+) بجانب Tables فيلاحظ ظهور الجدول .

فإذا لم يظهر الجدول ينقر بالزر الأيمن للفأرة على المجلد Tables ثم ينقر الامر Refresh أو من خلال الضغط ع الأداة Refresh في شريط الأدوات .

✚ للتأكد من كتابة التعليمة بالشكل الصحيح تحدد التعليمة ثم تنقر على الأداة Parse ✓ في شريط الأدوات ، فإن كانت التعليمة صحيحة الشكل تظهر لك الرسالة التالية **Commands Completed Successfully** في نافذة الرسائل اما اذا كانت خطأ فتظهر رسالة توضح الخطأ .

✚ يضاف الحرف N قبل قيمة الحقل اذا كانت القيمة باللغة العربية وذلك لإدخال القيمة الى الجدول كما هي وفي حال لم يوضع الحرف N تظهر قيمة الحقل بإشارات استفهام عوضاً عن القيمة المدخلة باللغة العربية .

✚ لايمكن حذف الجدول اذا اضيف الى مخطط قواعد المعطيات وتم انشاء علاقة أو أكثر بينه وبين واحد أو أكثر من الجداول . (لأنه سيكون هناك مفاتيح أجنبية بين الجدول المطلوب حذفه وبين هذه الجداول وبالتالي فغن حذف الجدول سيؤدي لضياع المعطيات من الجداول الأخرى لذلك يجب حذف العلاقات أولاً ثم حذف الجدول .

يتم حفظ الاستعلام بطريقتين :

١ - من القائمة File يختار الامر **Save File as** .

٢ - الضغط على المفاتيح **Ctrl + S** من لوحة المفاتيح .

بكلا الحالتين يدخل في الحقل File Name اسم الملف المطلوب حفظ الاستعلام به مع ملاحظة أن امتداد ملف الاستعلام عند الحفظ هو **(.sql)** . ثم ينقر الزر **Save** لحفظ الاستعلام .

اما لفتح الاستعلام المحفوظ سابقاً من القائمة File ينقر الامر **Open** ثم بنقر الامر **File** فتظهر نافذة يحدد منها المكان الذي حفظ الاستعلام فيه سابقاً ثم يحدد الاستعلام المطلوب ثم ينقر الامر **Open** فيتم فتحه في نافذة الاستعلامات .

بفضل الله تعالى

تم الانتهاء
من الوحدة
الثانية



للاستفسار عبر الرابط التالي :

<https://www.facebook.com/ramy.almasalmh.3/>

الوحدة الثالثة ربط البرامج مع قواعد المعطيات

ما الحاجة من ربط قواعد المعطيات مع البرامج التي صممت في لغات البرمجة كـ C#

إن عملية الاستفادة من المعطيات في قاعدة المعطيات تكون صعبة على المستخدم وطرائق عرضها ضعيفة وغير مناسبة للمستخدمين .

يتم تصميم واجهات رسومية GUI للمستخدمين وربطها مع نظام إدارة قواعد المعطيات DBMS للتقديم وعرض المعطيات الموجودة ضمن قواعد المعطيات فتقوم هذه البرامج بالدخول الى قواعد المعطيات الموجودة على الحاسوب او شبكة الانترنت .

عدد طرائق ربط برنامج VC# مع قاعدة المعطيات

١ - الربط باستخدام الكائن DataSet .
١ - الربط باستخدام SqlDataAdapter .

الربط باستخدام الكائن DataSet

اولاً يتم انشاء مشروع وليكن اسمه myprogram_connect يتم انشاء اتصال مع قاعدة المعطيات باستخدام Visual Studio كالتالي :

١ - يفتح مستكشف المشاريع Server Explorer من القائمة View من شريط القوائم ثم ينقر الامر Server Explorer.

٢ - تفتح نافذة Server Explorer تنقر منها Data Connections بالزر الأيمن للفأرة ثم تختار الخيار Add Connection .

٣ - تختار من النافذة التي ظهرت الخيار Microsoft SQL Server ثم ينقر الزر OK .

٤ - تكتب في الحقل Server Name اسم مخدم قواعد المعطيات (اذا لم يتم تغيير اسم المخدم اثناء تنصيب البرنامج فإن اسم المخدم هو localhost) اما في حال تم تغيير اسم المخدم وكلمة المرور اثناء التنصيب فيجب اختيار الخيار Use SQL Server Authentication وإدخال اسم المستخدم وكلمة المرور المناسبين والا اختيار الخيار Use Windows Authentication .

٥ - في الحقل Select or enter a database name يختار من القائمة المنسدلة اسم قاعدة المعطيات المطلوب ربط البرنامج معها .

٦ - ينقر الزر Test Connection للتأكد من إتمام عملية الاتصال مع المخدم فإذا كانت العملية صحيحة تظهر رسالة تؤكد إتمام عملية الاتصال.

٧ - ينقر الزر OK في نافذة إضافة الاتصال السابقة ليتم اتصال برنامج Visual Studio مع القاعدة المحددة .

٨ - يلاحظ ظهور قاعدة المعطيات المختارة مع جداولها ومخططاتها كافة في نافذة Server Explorer .

٩ - لإضافة قاعدة المعطيات المتصل بها الى المشروع المنشأ يجب انشاء DataSet حيث يتعامل التطبيق مع قاعدة المعطيات المخزنة ضمن DataSet في الذاكرة دون الحاجة الى استمرار الاتصال مع قاعدة المعطيات الأساسية بمعنى أن أي تعديل يحدث أولاً على DataSet ثم يتم تطبيقه على قاعدة المعطيات الأساسية بعد ان يعود اتصال التطبيق معها ، لإضافة DataSet تنقر القائمة Data من شريط القوائم ويختار منها الامر Add New Data Source .

١٠ - في نافذة تحديد مصدر المعطيات التي ظهرت يختار الخيار DataBase ثم Next .

١١ - يختار **DataSet** ثم **Next** .

١٢ - يحدد من القائمة المنسدلة اسم قاعدة المعطيات المطلوب تخزين معطياتها في **DataSet** المنشأة ثم التعامل معها ضمن المشروع ولتغيير اسم المخدم المتصل معه ينقر الزر **New Connection** وبعد تحديد اسم المخدم وقاعدة المعطيات ينقر الزر **Next** .

١٣ - تظهر نافذة محتويات قاعدة المعطيات المحددة فيضاف المطلوب منها الى **DataSet** ولتحديد جداول قاعدة المعطيات كافة وإضافتهم الى **DataSet** نفعل مربع الاختيار جانب **Tables** ولاختيار جداول معينة ينقر السهم بجانب **Tables** وتحدد الجداول المطلوب اضافتها فقط وفي حقل **DataSetName** يحدد اسم **DataSet** التي يتم انشاؤها ثم ينقر الزر **Finish** لإنشاء **DataSet** .

١٤ - يظهر ملف **DataSet** في مستكشف المشاريع وله الامتداد **(.XSD هام)**. لعرض محتوى **DataSet** ينقر الملف بالزر الأيمن للفأرة ثم يختار الأمر **Open** فتظهر نافذة تحوي جداول قاعدة المعطيات والعلاقات بين الجداول المضافة الى **DataSet** حيث **(هام)** يوجد **Adapter** في كل جدول من جداول قاعدة المعطيات ، **(هام)** يستخدم للتعامل مع قاعدة المعطيات (كقراءة المعطيات منها وتخزينها ضمن **DataSet** أو التعديل عليها) من خلال استخدام التوابع الخاصة به كالتوابع **(Fill ، GetData)** .

ملاحظة هامة جداً :

تعريف DataSet : هو نسخة عن مجموعة جداول قاعدة المعطيات تخزن في الذاكرة بشكل مؤقت ليتم التعامل معها من خلال تطبيق منشأ بلغة **VC#** .

ربط كائنات النموذج مع قاعدة المعطيات باستخدام الكائن Binding Source

ليكن لدينا قاعدة المعطيات **Clinic** :

للتعامل مع جدول المرضى **Sicks** في قاعدة المعطيات تضاف خمسة مربعات نصية **TextBox** للنموذج بحيث يكون كل مربع نص ممثلاً لحقل من حقول الجدول .

يمكن إجراء عملية الربط بين كل مربعات النصوص والحقول المكافئة لها في الجدول باتباع الخطوات الآتية :

١ - يحدد مربع النص المطلوب ربطه مع الحقل ، فمثلاً مربع النص الأول الممثل برقم المريض .

٢ - من نافذة الخصائص تختار الخاصية **GreenDataBinding** وتستخدم لربط الكائن مع حقل من احد الجداول الموجودة في **DataSet** المنشأة سابقاً ، ووظيفة **GreenDataBinding (هام)** الحصول على الجدول وحقوله من **DataSet** .

٣ - ينقر الحقل بجوار **Text** فتظهر قائمة منسدلة .

٤ - ينقر السهم بجوار **Other Data Source** ، ثم ينقر السهم بجوار **Project Data Source** ثم ينقر السهم بجوار **ClinicDataSet** وهي **DataSet** التي تم انشائها سابقاً فيظهر الجداول المحتواة ضمن **DataSet** المنشأة .

٥ - ينقر السهم بجوار الجدول **Sicks** في النافذة السابقة ثم ينقر على الحقل **Sick_id** لربط مربع النص المحدد الممثل برقم المريض مع هذا الحقل .

٦ - بعدها يتم ملاحظة ظهور المكونات الآتية في تصميم المشروع : **(هام)**

✓ **SicksDataSet** : هي **DataSet** المنشأة والمستخدمة لتخزين جداول قاعدة المعطيات **Clinic** في الذاكرة بشكل مؤقت للتعامل معها ضمن التطبيق .

- ✓ **SicksBindingSource** : هي المسؤولة عن إدارة العمليات بين التطبيق و **DataSet** المخزنة في الذاكرة (كإضافة المعطيات أو التنقل بينها أو حذف المعطيات).
- ✓ **SicksTableAdapter** : تحدد اسم **Adapter** الذي سيقوم بعمليات الإضافة أو التعديل أو جلب المعطيات من قاعدة المعطيات الى **DataSet** المنشأة .

٧ - لربط بقية مربعات النص مع الحقول الموافقة من جدول **Sicks** نتبع نفس الخطوات السابقة ، لكن في الحقل بجوار **Text** ينقر السهم بجوار **SicksDataBindings** عوضاً عن **Other Data Source** ثم يختار الحقل الموافق من الجدول .

٨ - عند التنفيذ سيظهر النموذج وتكون مربعات النص قد امتلأت من السجل الأول في قاعدة المعطيات .

٩ - يحتمل محتوى الجدول **Sicks** الى **DataSet** المنشأة بكتابة الشيفرة الآتية في حدث تحميل النموذج تلقائياً :

```
Private Void Form1_Load ( object sender , EventArgs e )
```

```
{
    //TODO: This line of code loads data into the 'ClinicDataSet
    this.SicksTableAdapter.Fill (this.ClinicDataSet.Sicks);
}
```

التابع Fill : (هام) يقوم بملء **DataSet** المخزنة في الذاكرة من الجدول **Sicks** في قاعدة المعطيات وعرض قيم حقول السجل الأول في مربعات النصوص .

التنقل بين سجلات الجدول المخزن ضمن DataSet

- ١ - يتم الانتقال الى السجل الأول في الجدول **Sicks** المخزن ضمن **DataSet** في الذاكرة كالتالي :
`this.SicksBindingSource.Position = 0 ;`
- ٢ - الانتقال الى السجل التالي في الجدول **Sicks** المخزن ضمن **DataSet** في الذاكرة كالتالي :
`this.SicksBindingSource.Position++ ;`
- ٣ - الانتقال الى السجل السابق في الجدول **Sicks** المخزن ضمن **DataSet** في الذاكرة كالتالي :
`this.SicksBindingSource.Position-- ;`
- ٤ - الانتقال الى السجل الاخير في الجدول **Sicks** المخزن ضمن **DataSet** في الذاكرة كالتالي :
`this.SicksBindingSource.Position = this.SicksBindingSource.Count - 1 ;`
- ٥ - للحصول على رقم السجل الحالي من الجدول **Sicks** المخزن ضمن **DataSet** في الذاكرة كالتالي :
`textBox6.Text = this.SicksBindingSource.Position.ToString() ;`
- ٦ - للحصول على عدد السجلات الموجودة في الجدول **Sicks** المخزن ضمن **DataSet** في الذاكرة كالتالي :
`textBox7.Text = this.SicksBindingSource.Count.ToString() ;`

ملاحظات :

- ❖ **الخاصية Position :** لتحديد رقم السجل المختار حالياً من **BindingSource** مع ملاحظة أن ترقيم السجلات يبدأ من صفر .
- ❖ **Count :** لحساب عدد السجلات في الجدول المخزن ضمن **DataSet** في الذاكرة لذلك رقم السجل الأخير هو **Count - 1** لان ترقيم السجلات يبدأ من صفر .

نتيجة تنفيذ التعليمات السابقة

ربط برنامج VC# مع قاعدة المعطيات باستخدام SqlDataAdapter

يمكن ربط برنامج VC# وقاعدة معطيات مخزنة في **SqlServer** باتباع الخطوات التالية :

أولاً : تضاف المكتبة **Data.SqlClient** الى البرنامج على النحو التالي :

using System.Data.SqlClient ;

ثانياً : انشاء اتصال جديد مع **Sql Server** يصرح عنه بكتابة التعليمتين التاليتين :

SqlConnection con ;

con = **new SqlConnection** (" Data Source = Ramy-Ms; Initial Catalog = Clinic; Integrated Security = True ") ;

تعريف هامة :

- ★ **الصف SqlConnection :** يستخدم لإنشاء اتصال جديد بين البرنامج و **SqlServer** حيث تمرر لتابعه البناء سلسلة نصية تمثل معلومات المخدم وقاعدة المعطيات اللذين سيتم الاتصال معهما .
- ★ **التابع البناء DataSource :** وسيط يمثل اسم المخدم المطلوب الاتصال به وهو الاسم نفسه الذي يظهر في نافذة الاتصال بالمخدم عند فتح برنامج **SqlServer** .
- ★ **التابع البناء Initial Catalog :** وسيط يمثل اسم قاعدة المعطيات المطلوب ربطها مع البرنامج .

★ **التابع البناء Integrated Security :** وسيط قيمته منطقية True / False :

تكون القيمة False للوسيط Integrated Security اذا اضيف اسم مستخدم وكلمة مرور اثناء تنصيب برنامج Sql Server وعندها يجب ذكر اسم المستخدم وكلمة المرور بإضافة الوسيط UserID والوسيط Password الى معلومات الاتصال المذكورة سابقاً .
تكون القيمة True للوسيط Integrated Security اذا لم يتم إضافة اسم مستخدم وكلمة مرور اثناء تنصيب برنامج SqlServer ويكون اسم المستخدم الحالي للنظام هو اسم المستخدم الذي سيتم الاتصال به مع SqlServer بواسطته .

ثالثاً : بعد انشاء الاتصال يتم فتح الاتصال والارتباط مع قاعدة المعطيات باستخدام التعليمة التالية :

con.Open();

رابعاً : انشاء تعليمة استعلام Sql جديدة وتنفيذها على قاعدة المعطيات المتصل معها من خلال التصريح عن متحول من نوع SqlDataAdapter كالتالي :

SqlDataAdapter adapter1;

adapter1 = new SqlDataAdapter (" Select * From Sicks " , con) ;

★ **الصف SqlDataAdapter :** هو صف يستخدم لإنشاء امر او تعليمة SQL جديدة ويكتب في تابعه البناء (أي التابع الموجود بعد الكلمة المفتاحية new) الوسيط الممثل لتعليمة SQL بشكل سلسلة نصية بعدها يكتب وسيط اسم متحول الاتصال con .

★ **المتحول adapter1 :** تحفظ نتيجة الاستعلام فيه ويكون صلة الوصل مع قاعدة المعطيات الاصلية ، كما يمكن له بالقيام بعمليات الإضافة والتعديل والحذف على قاعدة المعطيات الاصلية لكنه بحاجة الى الصف

SqlCommandBuilder من أجل تنفيذ العمليات بشكل تلقائي في الخلفية حيث لا يستخدم هذا الصف مباشرة .

★ **الصف SqlCommandBuilder :** هو الصف المسؤول عن ارسال التعديلات من قاعدة المعطيات المخزنة في الذاكرة الى قاعدة المعطيات الأساسية وتكتب التعليمة كالتالي :

SqlCommandBuilder combld ;

combld = new SqlCommandBuilder (adapter1) ;

★ **الصف DataTable :** هو صف يستخدم لإنشاء جدول فارغ في الذاكرة لملئه بنسخة من المعطيات ويتم كتابة التعليمة على الشكل التالي :

DataTable table_Sicks = new DataTable ();

تفيد التعليمة السابقة لإنشاء جدول معطيات فارغ في الذاكرة بالإسم table_Sicks .

لملء الجدول بالمعطيات يستخدم الإجراء Fill لمتحول الربط adapter1 كالاتي :

adapter.Fill (table_Sicks);

وتفيد هذه التعليمة للحصول على المعطيات الموجودة في adapter1 ووضعها في الجدول table_Sicks لمعالجتها .

التنقل بين سجلات جدول المعطيات وعرضها باستخدام SqlDataAdapter

للتنقل بين سجلات الجدول Sicks يجب التصريح عن متحول من النوع الصحيح int يشير الى رقم السجل الحالي في الجدول كالتالي :

int i = 0 ;

يعطي المتحول i القيمة 0 ليتم عرض السجل الأول لان الترقيم يبدأ من الصفر ويجب التصريح عنه بالتصريح العام لكافة البرنامج لاستدعائه حين الحاجة اليه .

عرض معطيات السجلات في الجدول

لعرض معطيات السجل ذو الرقم i من الجدول في مربعات النص الموافقة لكل حقل من الحقول نكتب التالي :

Private Void Showcurentrecord ()

```
{
    if ( table_sicks.Rows.Count == 0 )
    {
        textBox1.Text = "";
        textBox2.Text = "";
        textBox3.Text = "";
        textBox4.Text = "";
        textBox5.Text = "";
    }
    else
    {
        textBox1.Text = table_sicks.Rows[ i ][ " Sicks_id " ].ToString();
        textBox2.Text = table_sicks.Rows[ i ][ " Sicks_name " ].ToString();
        textBox3.Text = table_sicks.Rows[ i ][ " Sicks_gender " ].ToString();
        textBox4.Text = table_sicks.Rows[ i ][ " Sicks_Address " ].ToString();
        textBox5.Text = table_sicks.Rows[ i ][ " Sicks_Mobile " ].ToString();
    }
}
```

تفيد هذه التعليمة بالتحقق من وجود سجلات في الجدول فإذا لم يكن هناك سجلات تُعرض مربعات النصوص فارغة والا يتم تعبئة هذه المربعات بالحقول الموافقة لها .

ملاحظات هامة :

- يمكن اختيار أي اسم بدلاً من Showcurentrecord .
- ان التابع Showcurentrecord يتم تنفيذه في كل مرة يتم فيها استدعاؤه .
- بعد كتابة الاجراء showcurentrecord على مستوى النموذج Form1 يمكن استدعاؤه في أي مكان من النموذج.
- إذا كانت قيمة الحقل المقروءة قيمة عددية يجب تحويلها الى قيمة نصية لعرضها في أداة مربع النص .
- هام جداً** يتم التعامل مع جدول المعطيات على انه مؤلف من | مصفوفة ثنائية البعد | اسمها **table_Name.Rows** ودليل صفوفها هو **Index** ودليل اعمدتها هو اسم الحقل المذكور من جدول المعطيات.

الشكل العام للوصول للمعطيات من حقول جدول المعطيات (هام) :

table_Name.Rows[index]["Field_Name"] ;

حيث :

- table_Name** : هو اسم جدول المعطيات المرتبط مع الكائن **DataAdapter** .
- Rows** : كلمة مفتاحية تدل على مجموعة سجلات جدول المعطيات .
- Index** : دليل او ترتيب سجلات الجدول ويشير الى رقم السجل المطلوب .
- Field_Name** : يمثل اسم الحقل في جدول المعطيات .

كتابة شيفرات ازرار التنقل والتعديل والحذف والاضافة والبحث

١ - شيفرة النقر على الزر الأول ليعرض السجل الأول من جدول المعطيات في مربعات النصوص الموافقة :

```
Private Void button1_Click ( object sender , EventArgs e )
{
    i = 0;
    this.Showcurentrecord() ;
}
```

٢ - شيفرة النقر على الزر التالي ليعرض السجل التالي من جدول المعطيات في مربعات النصوص الموافقة :

```
Private Void button2_Click ( object sender , EventArgs e )
{
    if ( i < table_Sicks.Rows.Count - 1 )
    {
        i ++;
        this.Showcurentrecord() ;
    }
}
```

يتم في هذه التعليمة اختبار عدم الوصول الى السجل الأخير في الجدول حيث يقوم بحساب عدد السجلات باستخدام **Count** ويطرح منه **(واحد)** لأن ترقيم السجلات يبدأ من الصفر **(مثال اذا كان عدد السجلات 5 فإن رقم السجل الخامس هو 5-1 أي 4)** وزيادة قيمة فهرس السجلات **i** بمقدار واحد ثم استدعاء اجراء عرض المعطيات **Showcurentrecord** .

٣ - شيفرة النقر على الزر السابق ليعرض السجل السابق من جدول المعطيات في مربعات النصوص الموافقة :

```
Private Void button3_Click ( object sender , EventArgs e )
{
    if ( i != 0 )
    {
        i --;
        this.Showcurentrecord() ;
    }
}
```

يتم في هذه التعليمة اختبار عدم الوصول للسجل الأول فإذا لم يتم الوصول الى السجل الأول ينقص فهرس سجلات الجدول بمقدار واحد لعرض السجل السابق .

٤ - شيفرة النقر على الزر الاخير ليعرض السجل الاخير من جدول المعطيات في مربعات النصوص الموافقة :

```
Private Void button4_Click ( object sender , EventArgs e )
{
    i = table_Sicks.Rows.Count - 1 ;
    this.Showcurentrecord() ;
}
```

يتم في هذه التعليمة عرض السجل الأخير بحساب عدد السجلات ناقصاً بواحد كما شرحنا سابقاً .

٥- شيفرة النقر على الزر تعديل ليتم تعديل السجل الحالي من جدول المعطيات :

```
Private Void button5_Click ( object sender , EventArgs e )
{
    table_Siks.Rows[ i ][ " Sicks_Name " ] .ToString() = textBox2.Text ;
    table_Siks.Rows[ i ][ " Sicks_gender " ] .ToString() = textBox3.Text ;
    table_Siks.Rows[ i ][ " Sicks_Address " ] .ToString() = textBox4.Text ;
    table_Siks.Rows[ i ][ " Sicks_Mobile " ] .ToString() = textBox5.Text ;
    adapter1.Update ( table_Sicks ) ;
}
```

يتم في هذه التعليمة بأخذ قيمة احد الحقول بقيمة النص المكتوب في مربع النص الموافق له كما يتم استخدام اجراء التحديث أو التعديل (Update (table_Sicks الخاص بمتحول الربط adapter1 من اجل تحديث المعطيات جدول المعطيات في الجدول الأساسي .

ملاحظة : في تعليمة التعديل لا يتم ذكر حقل رقم المريض الممثل بـ (Sicks_id) لانه يكون مفتاح رئيسي والمفتاح الرئيسي لا يمكن التعديل عليه .

٦- شيفرة النقر على الزر حذف ليتم حذف السجل الحالي من جدول المعطيات :

```
Private Void button6_Click ( object sender , EventArgs e )
{
    if ( table_Siks.Rows.Count !=0 )
    {
        table_Siks.Rows[ i ].Delete();
        adapter1.Update ( table_Siks );
        i = 0 ;
        this.Showcurentrecord();
    }
}
```

يتم في هذه التعليمة التأكد أن الجدول غير فارغ فإن لم يكن فارغ يتم حذف السجل الحالي ذو الرقم i ثم يطبق اجراء التحديث أو التعديل (Update (table_Sicks على متحول الربط adapter1 لحفظ التغييرات التي أجريت في جدول المعطيات الأساسي ثم الانتقال الى السجل الأول وعرض معطياته .

٧- شيفرة النقر على الزر اضافة ليتم اضافة السجل الذي تم تعبئته ضمن مربعات النصوص الى جدول المعطيات :

```
Private Void button7_Click ( object sender , EventArgs e )
{
    DataRow nrow = table_Sicks.NewRow() ;
    nrow[ " Sicks_id " ] = table_Sicks.Rows.Count + 1 ;
    nrow[ " Sicks_Name " ] = textBox6.Text ;
    nrow[ " Sicks_gender " ] = textBox7.Text ;
    nrow[ " Sicks_Address " ] = textBox8.Text ;
    nrow[ " Sicks_Mobile " ] = textBox9.Text ;
    table_Sicks.Rows.Add (nrow) ;
    adapter.Update (table_Sicks);
    i = table_Sicks.Rows.Count -1;
    this.Showcurentrecord();
}
```

تم في هذه التعليمة التالي :

استخدام الاجراء **NewRow()** (هام) : يستخدم لإنشاء سجل فارغ في الذاكرة يوافق حقول جدول المعطيات ويوافق نوع معطياتها .

ثم تم تعبئة هذا السجل الفارغ **nrow** من المربعات النصية ثم تمت إضافة السجل الجديد **nrow** الى الجدول **table_Sicks** باستخدام الاجراء **Add()** لإضافة السجل لجدول المعطيات ثم تم تطبيق اجراء التحديث او التعديل **Update (table_Sicks)** على متحول الربط **adapter1** لحفظ التغييرات التي أجريت في جدول المعطيات الأساسي.

ملاحظات هامة :

- ✓ يجب تحويل القيم النصية الى نوع المعطيات الذي يوافق نوع معطيات حقل السجل على سبيل المثال (المعدل **Average** يجب تحويله الى **Float** لأنه يأخذ قيمة عشرية باستخدام اجراء التحويل **Convert.ToSingle()**).
- ✓ نلاحظ في التعليمة السابقة قمنا بإسناد عدد السجلات وزيادة واحد عليه (لأن التقييم في قاعدة المعطيات يبدأ بواحد أي ان السجل الأول هو عدد السجلات + 1 فإذا كان عدد السجلات هو 0 فإن السجل الأول هو 0 + 1 أي رقم السجل الأول في قاعدة المعطيات هو 1 .

٨- شيفرة النقر على الزر بحث للبحث عن السجل الذي تم كتابته ضمن مربع نص البحث وعرضه:

```
Private Void button8_Click ( object sender , EventArgs e )
{
    SqlDataAdapter adapter2;
    adapter2 = new SqlDataAdapter ( " Select * From Sicks Where Sicks_Name = N' "
    +textBox10.Text+ " ' " , con);
    DataTable result_Search = new DataTable();
    adapter2.Fill(result_Search);
    if (result_Search.Rows.Count == 0)
        MessageBox.Show( " لم يتم العثور على الاسم المطلوب " );
    else
    {
        textBox1.Text = result_Search.Rows[ j ][ " Sicks_id " ].ToString();
        textBox2.Text = result_Search.Rows[ j ][ " Sicks_Name " ].ToString();
        textBox3.Text = result_Search.Rows[ j ][ " Sicks_gender " ].ToString();
        textBox4.Text = result_Search.Rows[ j ][ " Sicks_Address " ].ToString();
        textBox5.Text = result_Search.Rows[ j ][ " Sicks_Mobile " ].ToString();
        if ( j != result_Search.Rows.Count - 1)
            j++;
    }
}
```

تم في هذه التعليمة الاتي :

التصريح عن متحول ربط جديد **adapter2** لكتابة تعليمة **Sql** جديدة لعرض السجلات حسب الاسم بإضافة العبارة الشرطية **Where** للتعليمة ويتم ادخال هذا الاسم في مربع النص **textBox10** و تم التصريح عن جدول جديد في الذاكرة بالاسم **result_Search** وذلك لحفظ نتيجة الاستعلام السابق فيه من خلال استخدام التابع **Fill** ثم يتم فحص وجود سجلات بالاسم الذي تم البحث عنه فإذا كان الجدول **result_Search** فارغاً هذا يعني أنه لم يتم العثور على سجلات فيظهر مربع حوار رسالة والا اذا تم العثور على الاسم المطلوب فيتم عرض السجل الموافق للاسم المطلوب في مربعات النصوص الموافقة لكن يمكن أن يكون في قاعدة المعطيات أكثر من سجل يوافق اسم المريض الذي نبحث عنه لذلك وضعنا الشرط الأخير الذي يفيد انه إن لم يتم الوصول الى اخر اسم يوافق الاسم الذي نبحث عنه فإعرض الاسم الموافق التالي وذلك من خلال الدليل **j** .

ملاحظة : يتم التصريح عن الدليل **j** في القسم العام للتصريحات من نوع **int** ليكون الدليل او الفهرس المسؤول عن عرض سجلات الأسماء التي تم العثور عليها .

إغلاق الاتصال مع قاعدة المعطيات

لإغلاق الاتصال تكتب التعليمة التالية في شيفرة انتهاء البرنامج :

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

(هام)

- ❖ الاجراء Close() : يقوم بقطع الاتصال مع قاعدة المعطيات .
- ❖ الاجراء Dispose : يقوم بتحرير مساحات الذاكرة التي يحجزها الاتصال وكائناته كالجداول وأدوات الربط .

اللهم اجعلنا قدوة حسنة لطالما حيننا ... ننتفع وينتفع الناس بنا دائماً ابداً...

مع تمنياتي لكم بالتوفيق والتفوق

بفضل الله تعالى

تم الانتهاء من الوحدة الثالثة



للاستفسار عبر الرابط التالي :

<https://www.facebook.com/ramy.almasalmh.3/>