C# .Net

حتى الإحتراف

حماة النهضة

تأليف: حماة النهضة

إعداد و تنسيق: y85.fox

الفهرس

الموضوع **الصفحة**

الموضوع	الصفحة

الموضوع	الصفحة

مقدمة:

السلام عليكم و رحمة الله و بركاته...

النتيجـــة : صنع برنامج مشغل صوتيات شبيه بالريال بلاير و الجت أوديو .

الهـــدف : إخواننا الكرام في كل المنتديات العربية .

الطريقة : دروس برمجة مشروحة و معروضة على شكل صور موضح عليها كل شيء .

ما يميزنا عن الجميـــع :

1- أننا نجيب على كل الأسئلة بإذن الله .

2- أننا سنواصل معكم حتى النهاية ..... ولو بقي معنا شخص واحد فقط.

مع أمثلـــــــة لنتائج من طبقوا تلك الدروس...

من نحن ؟

حماة النهضة مجموعة من الإخوة متحابين في الله وضعوا نهضة الأمة نصب أعينهم و يسيرون لها بشتى و مختلف الطرق سواءً أكانت إيمانية , أو تكنولوجية , أدبية , ثقافية , برمجية ..... بكل الطرق.

و في هذا الدرس نسعى جاهدين لإفادتكم ..... فأنتم إخواننا.

يا إخوة .....

هذه الدروس مبتدئة ..... لكن بنفس الوقت هيا رائعة ..... و سنتابع بعدها دروساً أكبر و أضخم إن شاء الله .

من رأى فينا خطأً ..... فلينصحنا .... فنحن بشر .....

لسنا الوحيدون ..... و لكننا نسعى للأفضل بإذن الله لخير الأمة و نهضتها

و إنه لأملٌ أمل ..... جهدٌ و عمل

صناع الثقة و التغيير

حماة النهضة

1- النتيجة:

باختصار الموضوع عبارة عن مجموعة دروس مبسطة في لغة C# 2005 للمبتدئين .بهدف إنشاء هذا البرنامج لتشغيل الصوتيات والمرئيات :

الدروس ستكون مقسمة إلى قسمين , قسم الدرس وقسم التطبيق .وفي كل درس نبدأ بشرح الفكرة أو المفهوم أو القاعدة وفي قسم التطبيق نطبق اللي تعلمانه من الدرس على مشروع البرنامج.ونستمر بهذه الطريقة كل درس إلى أن ننجز المشروع.

لكن قبل البداية يجب تجهيز برنامجين للعمل على اللغة والبرنامج الأول من ميكروسوفت وهو برنامج ملفات لغة البرمجة .NET Framework

التحميل من هنا

حجم الملف يقارب 24 ميجا بايت

عند فتح هذا اللينك ستجد زر مكتوب عليه Download

البرنامج الثاني هو بيئة التطوير , أي المكان الذي سنكتب فيه تعليمات البرنامج ونجربه .

يوجد هناك عدة بيئات تطوير مجانية متوفرة منها بيئة ميكروسوفت بإصدارها المضغوط Express

لتحميل C# 2005 Express

حجم الملف يقارب 450 ميجا بايت

إذا لم تستطع تحميل هذه البيئة فهناك بيئة أخرى مجانية

Sharp Developer

حجم الملف يقارب 4 ميجا بايت

لتنظيم العمل على الدروس ينصح عمل مجلد جديد على القرص المحلي D وتسميته CS

يجب أولاً تنصيب البرنامج الأول dotnetfx.exe

وعند الانتهاء منها يتم تنصيب البيئة SharpDevelop2_2.0.0.1591_Setup.exe أو Express C#

وبهذا تكون مستعداً لتلقي الدرس الأول

الدرس الأول

مفهوم لغات البرمجة

منذ بداية ظهور الكهرباء وتطورها , ظهرت فكرة معالجة التيار الكهربائي والاستفادة منه .فمثلاً المصباح يأخذ الكهرباء ويعالجها "يحولها" إلى ضوء .والراديو يأخذ الكهرباء ويحولها إلى صوت .. وهكذا

وبالتالي فإن أي عمل مع الكهرباء يحتاج إلى ثلاثة أشياء مهمة.

1- مدخلات "تيار كهربائي".

2-معالجة "تحويل التيار إلى ضوء , صوت...."

3-مخرجات "ضوء , صوت ...."

بالإضافة إلى جهاز التحويل "المصباح , الراديو"

الحاسوب لا يخرج عن هذه القاعدة فهو يحتاج إلى مدخلات ليعالجها ويحولها إلى مخرجات.

مثلاً عندما تضغط أي زر في لوحة المفاتيح , فإن تيار كهربائي يتولد وينطلق باتجاه المعالج.

المعالج بدوره يحول هذا التيار إلى تيار مختلف ينطلق نحو الشاشة ويظهر الحرف .قد يتساءل البعض كيف يفرق المعالج بين زر وآخر في لوحة المفاتيح , فجميع الأزرار تطلق تيار كهربائي. وهذا هو أساس العمل في الحاسوب , حيث أن معالج الكمبيوتر لا يستقبل تيار مستمر. كما في المصباح , أو في الراديو , أو في أي جهاز كهربائي وإنما معالج الكمبيوتر يستقبل تيار متغير .

للتوضيح نفرض أن العدد 1 يمثل مرور تيار والعدد 0 يمثل عدم مرور التيار , عندما يكون المصباح مطفأ يكون التيار الذي يستقبله هكذا

0000000000000000

أما عندما نضيء المصباح فإن التيار يكون هكذا

1111111111111111

وهكذا مع الراديو و المكواة و.و.و....

لكن في الحاسوب يختلف شكل التيار لأنه يكون متغير فيكون شيئاً هكذا

11000101010001111010100011101010100 1

وهذه هي الطريقة التي يفرق المعالج حرف عن حرف آخر فالتيار الذي يمثل حرف الواو مثلاً يكون هكذا 11001010 وحرف الطاء يكون هكذا 10111010 وحرف النون يكون هكذا 01010011

وبالتالي فإن كلمة وطن ستكون هكذا

010100111011101011001010

ولا ننسى أن معنى 0 عدم مرور تيار و معنى 1 مرور تيار وبافتراض أن سرعة قراءة المعالج لرقم هي ثانية واحد فأن التيار الكهربائي الذي يمثل حرف الواو 11001010 سيكون هكذا:

في البداية ينقطع التيار لفترة ثانية ثم يمر التيار لمدة ثانية, ثم ينقع لمدة ثانية ثم يمر لمدة ثانية ثم ينقطع لمدة ثانيتين ثم يمر لمدة ثانيتين ,أي أن المعالج سيستغرق ثمان ثواني لقراءة حرف الواو .

في المثال السابق افترضنا أن سرعة المعالج هي ثانية واحدة لكنها في الحقيقة أسرع بكثير ,ومن هنا أعتقد إننا فهمنا ما معنى سرعة المعالج "هي سرعة قراءته للرقم الواحد" ,والرقم الواحد سواء كان 1 أو 0 يسمى "بت" Bit

فعندما تكون سرعة المعالج 3200 يعني أنه يقرأ 3200 ميجا بايت في الثانية الواحدة , والميجا يحتوي على 1024 كيلو بايت والكيلو بايت يحتوي على 1024 بايت , والبايت يحتوي على 8 بت , يعني أن هذا المعالج يقرأ 3277832 بت في الثانية الواحدة . يعني أنو يقرأ حرف الواو في مدة 0.00000244 من الثانية .

والآن بعد أن عرفنا كيف يقرأ الكمبيوتر الكلمات بقي أن نعرف كيف ينفذ الكمبيوتر الأوامر المطلوبة منه. كان الحاسوب في بدايته يستقبل الأوامر كتيار كهربائي مثل ما رأينا سابقاً , لكن هذه الطريقة غير عملية , فإذا أردنا أن نأمر الكمبيوتر بأن يطبع كلمة وطن على الشاشة فسيكون الأمر شبيهاً بهذا :

01010011101010101011010100111011101011001010

فهذه هل اللغة التي يفهمها الحاسوب , لكنها لغة صعبة الفهم على الإنسان ,لذلك ظهر ما يسمى بلغات البرمجة

" وهي لغات قريبة من لغة الإنسان ويفهما الحاسوب"

حيث أن لغات البرمجة سهلت التعامل مع الحاسوب وأصبح من الممكن إصدار الأوامر للحاسوب هكذا:

كود

```
Print "وطن"
```

وحيث أن الحاسوب لا يفهم هذه اللغة يوجد ما يسمى بالمترجم , وهو برنامج ملحق بلغة البرمجة وظيفته قراءة الأوامر المدخلة وتحويلها إلى تيار يفهمه المعالج , لكن عملية الترجمة تتم بشكل غير ظاهر , فما على المبرمج إلا أن يكتب الأوامر ويشغل البرنامج .. والمترجم يقرأ Print "وطن" ويحولها إلى :

01010011101010101011010100111011101011001010

ويرسلها إلى المعالج..

مفهوم الكائنات

في لغات البرمجة كل شيء يعتبر كائن له صفات وله أفعال كأي كائن في الوجود , فمثلاً السيارة كائن له صفات "مثل اللون والشكل وعدد الإطارات...." وله أفعال مثل "السير , التوقف, ..."

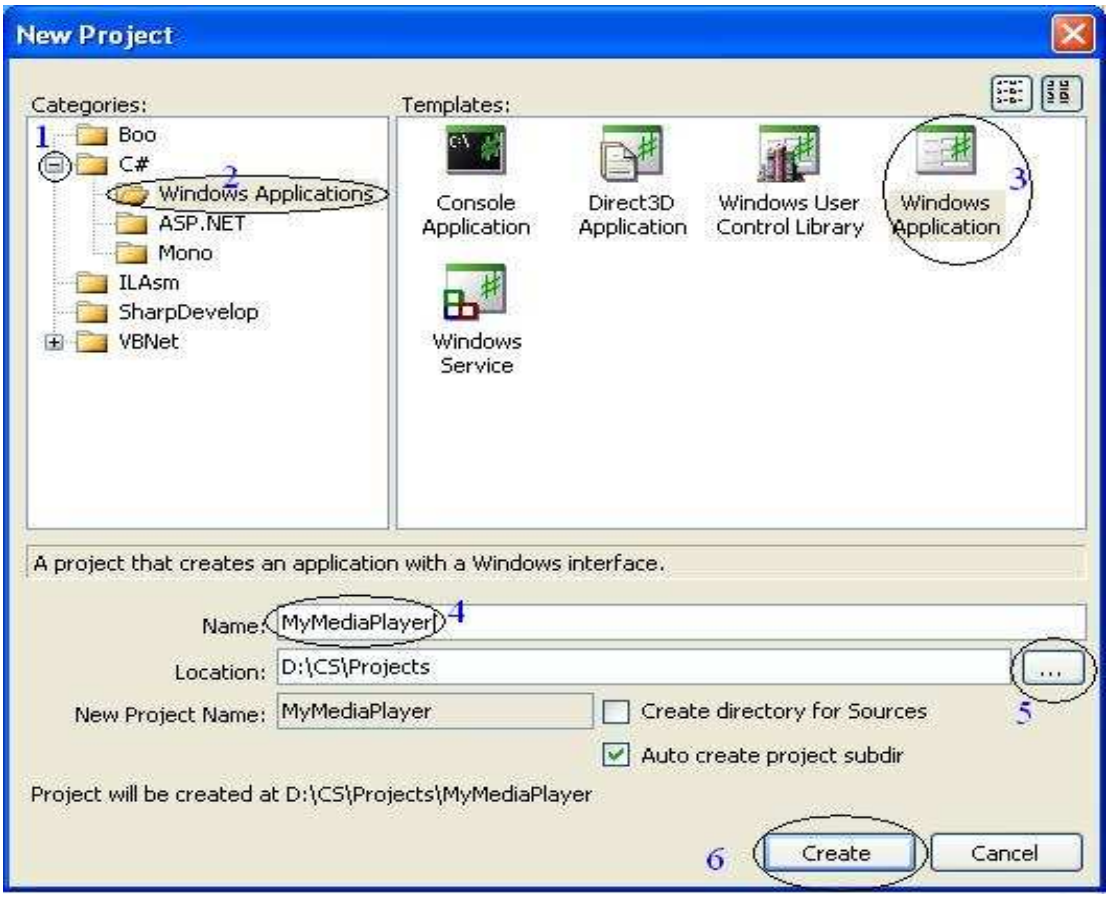
لا حظوا أن الأفعال قد تكون من الكائن نفسه "السير" و قد تكون من كائن آخر ,في الحاسوب أيضاً الأمر لا يختلف فكل شيء يعتبر كائن له صفات وأفعال ,فمثلاً هذا المنتدى كائن له صفات وأفعال , وزر"اقتباس" تحت هذه المشاركة هو كائن له صفات مثل اللون , و الكلام المكتوب عليه و الحجم... وله أفعال مثل أنه يعمل رد جديد مع اقتباس الرد الحالي...
و صندوق النص في آخر هذه الصفحة عند الرد السريع , يعتبر كائن له صفات كاللون وله أفعال كإرسال الرد إلى قاعدة بيانات المنتدى..
وزر ابدأ أو Start في الويندوز هو كائن وشريط المهام Task Bar هو كائن يحتوي على كائن آخر "زر ابدأ"
إذاً يمكن لكائن أن يحتوي على عدة كائنات فالسيارة كائن يحتوي على كائن آخر وهي الإطارات مثلاً .

تطبيق الدرس الأول:

بعد تنصيب البرامج افتح مجلد CS الذي أنشأته في القرص D وداخل المجلد CS قم بإنشاء مجلد جديد اسمه Projects من سطح المكتب لتجد إيقونة برنامج بيئة التطوير وهذا شكلها:

بعد أن تفتح البيئة سيظهر لك زرين في الوسط هكذا :

أنقر على زر New Solution المؤشر عليه بالأحمر ستظهر لك هذه النافذة:



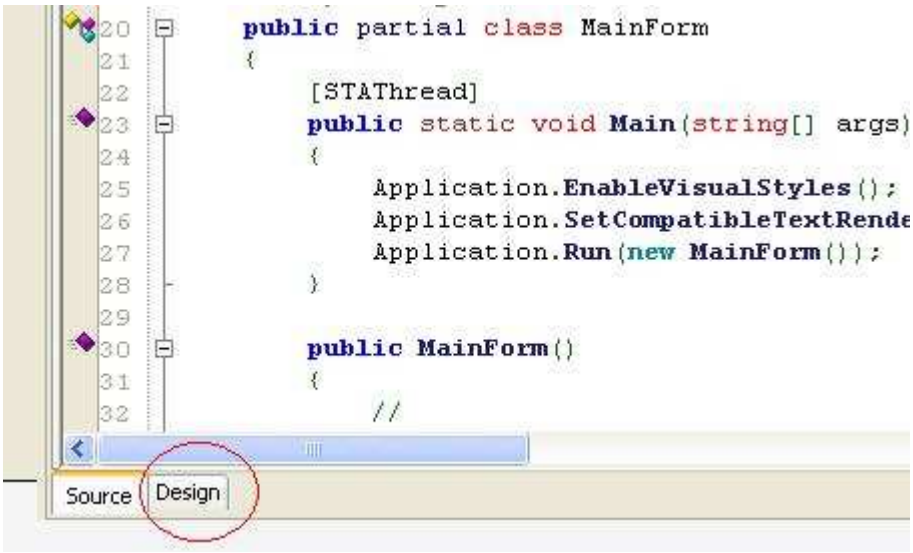
أنقر لى إشارة + الموظحه في رقم 1 ثم انقر علىWindows Applications كما في رقم 2

ثم أنقر على Windows Application كما في رقم 3 , عند صندوق اسم المشروع أكتب أسم المشروع الذي ترغب به وليكن:

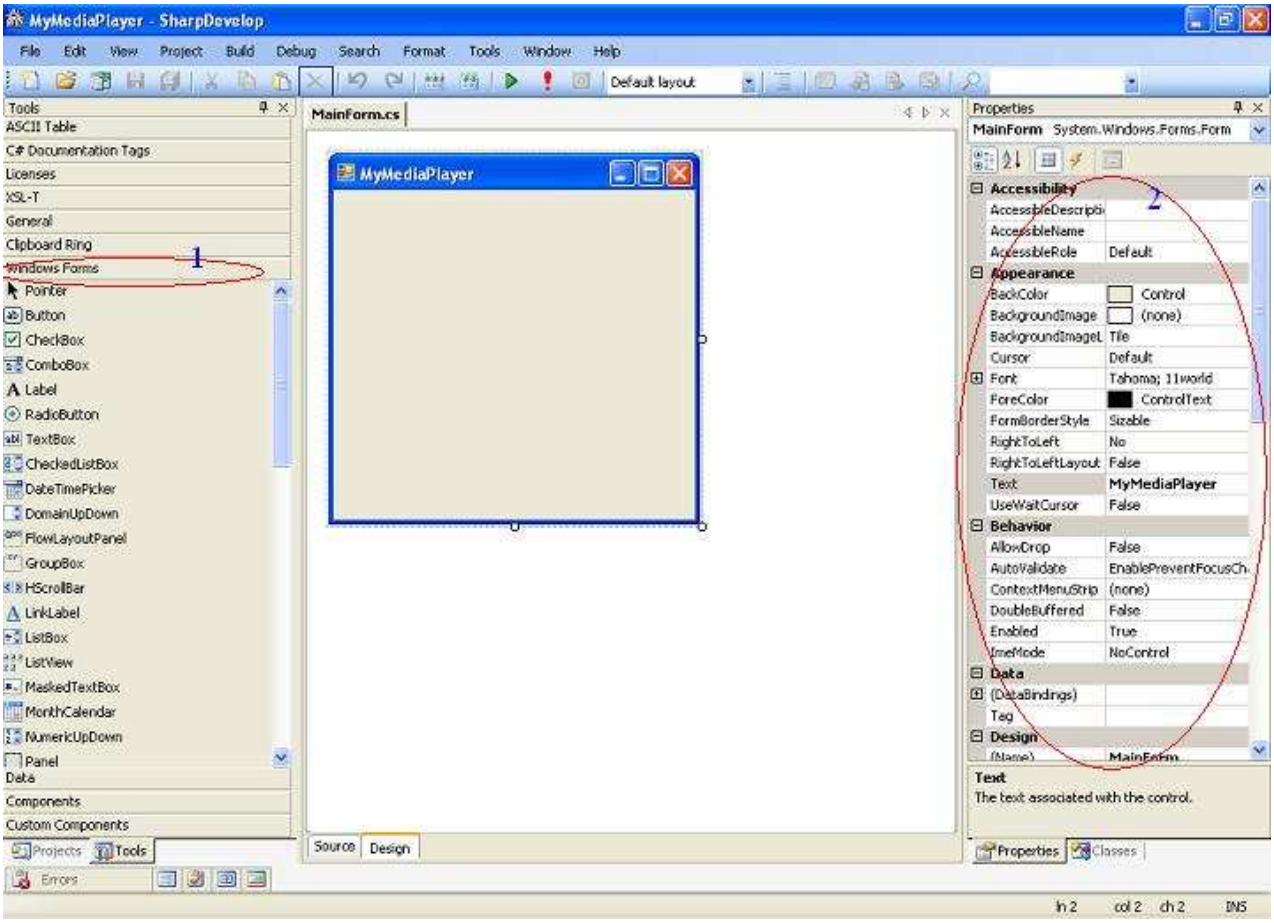
MyMediaPlayer كما في رقم 4 ثم انقر على الزر في رقم 5 وأختر مجلد Projects

الذر أنشأته سابقاً داخل مجلد CS في القرص D ثم أنقر على زر Create في رقم 6

تظهر لك نوافذ المشروع وفيها شفرة المشروع الافتراضية , تحت نافذة الشفرة انقر على زر Design كما في الصورة



ستظهر أمامك هذه النوافذ:



في اليسار انقر على Windows Forms كما في رقم 1 ستظهر لك جميع الكائنات التي نحتاجها في البرامج ,وفي الوسط ستظهر لك نافذة فارغة نسميها Form وهذه النافذة هي النافذة الرئيسية للمشروع ,ولا ننسى إن هذه النافذة تعتبر كائن لها صفات مثل أسمها "MyMediaPlayer " ولها أفعال مثل ما سنلاحظ لاحقاً , على اليسار يوجد جدول وهذا هو جدول صفات الكائنات ومن خلاله يمكن تغيير أي صفة من صفات الكائن ,ما رأيك لو قمنا بتغيير لون النافذة إلى الأزرق..

نذهب إلى جدول الخصائص ونبحث عن خاصية BackColor ,ننقر على الصندوق أمامها ستظهر قائمة ألوان , نختار منها أي لون وليكن الأزرق

نلاحظ أن لون النافذة تغير , يعني أننا غيرنا صفة من صفات هذا الكائن , الآن نرجع الكائن للونه الطبيعي

يوجد صفة أيضاً مهمة أسمها Text أبحث عنها في جدول الصفات عند رقم 2

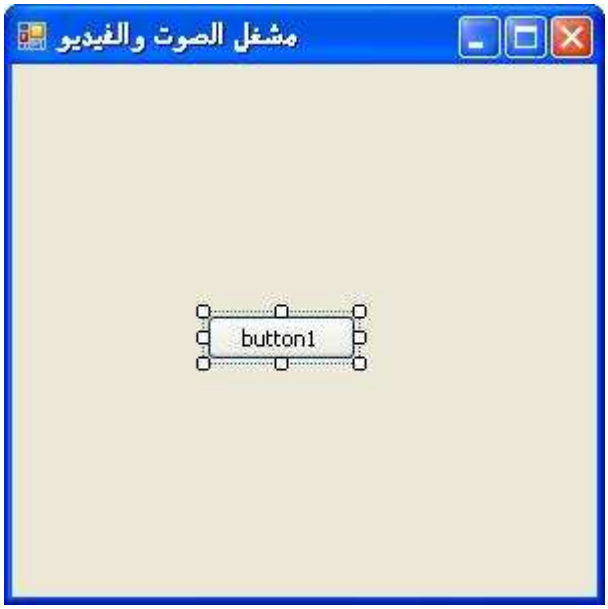
ستجد مكتوب أمامها MyMediaPlayer

أمسحها وأكتب بدلها "مشغل الصوت والفيديو".

ملاحظة: لسهولة الوصول إلى الصفات يمكن ترتيب الصفات بالترتيب الأبجدي بالنقر على زر الترتيب في أعلى جدول الصفات :

من نافذة الأدوات على اليسار انقر على صورة الزر المكتوب أمامه Button , واستمر في الضغط واسحب الماوس فوق النافذة الفارغة .

سيكون شكل النافذة هكذا:



يمكن تغيير حجم أو موقع الزر , الآن تأكد من اختيار الزر و اذهب إلى نافذة الصفات ,أبحث عن صفة اسمها Text وأكتب بدل button1 أكتب "ترحيب".

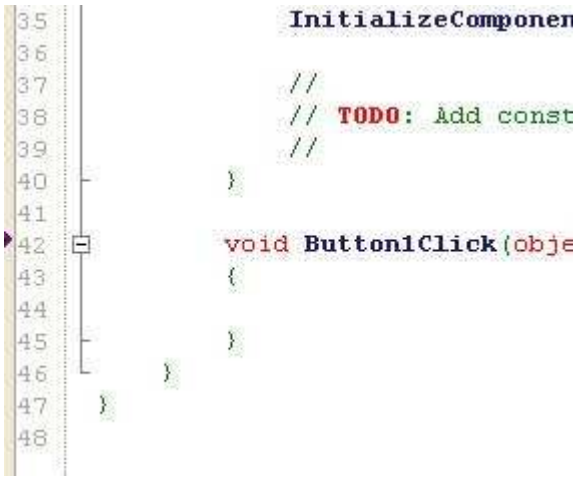
لاحظ أنا غيرنا صفة من صفات الزر , ولكن كيف نغير من أفعال الزر لأنه كائن ولا بد له من أفعال ,والزر يمكن أن يفعل أي شيء لكنا نريده أن يظهر رسالة ترحيب ..

للذهاب إلى أفعال الزر أنقر نقراً مزدوجا Double Click على الزر ,

ستظهر لك نافذة الشفرة :

.

.



وينتقل مؤشر الماوس إلى مكان بين حاصرتين مكتوب فوقة void Button1Click ومعنى هذا أن المعالج عند ضغط الزر سينفذ الأوامر المكتوبة بين الحاصرتين ,لا تهتم للشفرة المكتوبة فوق ولا تعدل فيها أبدا فهي مهمه لتشغيل البرنامج الآن بين الحاصرتين يجب أن نكتب الأمر الذي يظهر الرسالة وهذا الأمر هو:

كود

```
MessageBox.Show("مرحباً بكم إلى سي شارب");
```

لنشرح الأمر , أولا MessageBox هو كائن في لغة البرمجة يمثل إظهار رسالة و Show هو كائن أخر داخل كائن MessageBox ويقوم بطباعة الرسالة على الشاشة وكلمة "مرحباً بكم إلى سي شارب" تعتبر مدخل للكائن وهو يعالج الأمر ويخرج الرسالة على الشاشة .

لاحظ أن الأمر MessageBox.Show يعني أظهر رسالة , ولكن المعالج لن ينفذه لأنه لا يعرف ماذا يكتب في الرسالة لذلك يجب فتح قوس وكتابة الرسالة بين علامة اقتباس , والقوسين تعني أن بينهما مدخلات ,أما علامة الاقتباس فهي مهمة لأنه بدونها سيعتبر المترجم أن مرحباً بكم إلى سي شارب هي جزء من الأمر .

ولا يفهم هذا الأمر لذلك لن يعمل البرنامج ويجب أن تكون الرسالة بين علامتي اقتباس حتى يتجاوزها المترجم, في نهاية السطر يجب كتابة الفاصلة المنقوطة حتى نخبر المترجم أن هذه هي نهاية السطر وينتقل إلى السطر التالي .

ملاحظة: يجب أن تكون حذراً جداً عند كتابة الشفرة فزيادة نقطة أو حرف تجعل المترجم يرتبك ولا يشغل البرنامج وأيضاً يجب مراعاة الأحرف الكبيرة والصغيرة

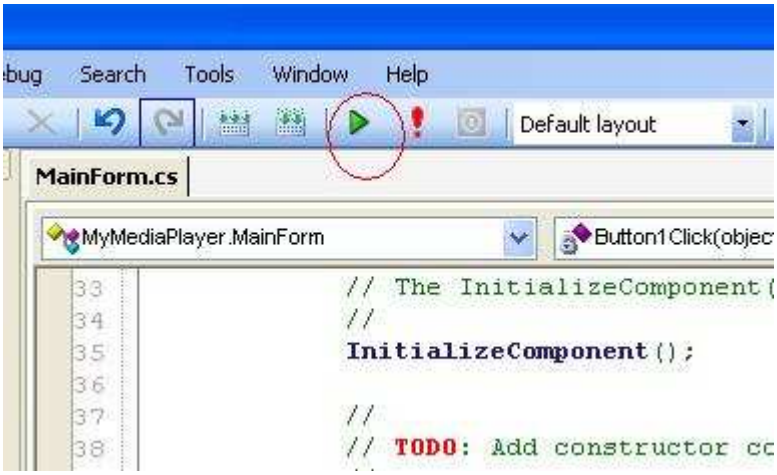
فالمترجم لن يفهم هذا الأمر messagebox بينما يفهم هذا الأمر MessageBox

.

.

.

بقي لنا أن نشغل البرنامج ونرى كيف سيعمل , لتشغيل البرنامج أنقر على زر التشغيل أعلى نافذة الشفرة:



عند تشغيل البرنامج ستظهر لك النافذة الأساسية وعند النقر على زر الترحيب ستظهر رسالة الترحيب:

مبروك لقد نجحت في تنفيذ أول برنامج بلغة C#

واجب الدرس الأول

أنشئ مشروع جديد و أنشئ فيه زر لرسالة ترحيب ,غيّر لون النافذة الأساسية , و غير حجمها , وغير لون الزر وغير حجمه ,من الممكن أيضاً إنشاء زر آخر يظهر رسالة أخرى وأرسل التطبيق .

لإرسال التطبيق:

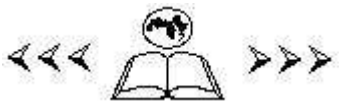
اذهب إلى مجلد CS في القرص D ثم إلى مجلد Projects ثم إلى مجلد MyMediaPlayer

ثم إلى مجلد bin ثم إلى مجلد Debug ستجد ملف تنفيذي بإسم MyMediaPlayer

هذا هو برنامجك الأول .. أضغطه ببرنامج الونزيب وأرسله لي

ملاحظة:عند عمل الواجب أو عند تجربة كائنات أخرى , أنشئ مشروع جديد للعبث به , لكن لا تغير أي شيء في المشروع الذي نطبق عليه

.نهاية الدرس الأول



الدرس الثاني

المتغيرات

في الدرس الأول تعلمنا أن كل شيء في الحاسوب يعتبر كائن وفي الحقيقة هناك نوعين من الكائنات :

كائنات حقيقية: وهي التي لها صفات وأفعال ويمكن أن يكون لها تمثيل مرئي.

المتغيرات : وهي متغيرات لا تحتوي على صفات أو أفعال وإنما تحتوي على قيمة محددة ولا تكون مرئية ,أي انه من الممكن أن يحتوي الكائن "المتغير" X على قيمة مخزنة في ذاكرة الحاسوب , ومن الممكن أن تكون قيمة المتغير أما رقم أو نص أو حرف أو رقم عشري أو....

وإذا أردنا أن ننشئ متغير في ذاكرة الحاسوب فيجب أن نحدد نوع القيمة التي سيحملها المتغير ,مع العلم أننا نختار أسم الكائن كما نشاء مادام الاسم يحقق الشروط التالية:

1-أن يكون اسم المتغير حروف أو أرقام لاتينية

2-أن لا يكون أول حروف الاسم رقم , فهذا الاسم غير مسموح 5 int بينما هذا مسموح int5

3-أن لا يحتوي الاسم على علامات أو إشارات مثل # أو $ أو * أو + , الإشارة المسموح فقط هي _

وهذه الشروط تنطبق على أسماء الكائنات أو المتغيرات التي سنقوم بإنشائها.

إذاً المتغير هو كائن يستخدم لتخزين قيمة من نوع محدد

مثلاً إذا أردنا أن نعرف متغير اسمه x من نوع رقم سيكون الكود كالتالي:

كود

```
int x;
```

وكلمة int اختصار لكلمة integer ومعناها عدد صحيح ,الكود السابق معناه أنا عرفنا متغير اسمه x سيحمل قيمة من نوع عدد صحيح ,لاحظ أنك تستطيع تغيير اسم المتغير كما تشاء المهم أن تراعي شروط التسمية الثلاثة ,وإذا أردنا تحميل هذا المتغير قيمة محدد .

يكون الكود كالتالي:

كود

```
x=20;
```

الكود السابق معناه أن المتغير x يحمل القيمة 20 ويستمر بحمل القيمة حتى يتوقف البرنامج أو نحمله قيمة أخرى

لاحظ الكود التالي:

كود

```
int Var1;
  int Var2;
  Var1=10;
  Var2=Var1;
```

قمنا بتعريف متغيرين من نوع رقم صحيح ثم حملنا الرقم الأول قيمة 10 ,وفي السطر الأخير حملنا المتغير الثاني قيمة المتغير الأول أي أن المتغير الثاني يحمل أيضاً القيمة 10

ملاحظة مهمة: في لغات البرمجة معامل التساوي ليس تبادلي يعني أن

Var2=Var1

لا تساوي أبداً

Var1=Var2

لان معامل = في لغات البرمجة يقوم بنقل قيمة المتغير في اليمين إلى المتغير في اليسار

فلو عكسنا المعاملات في السطر الأخير من الكود السابق هكذا :

كود

```
int Var1;
int Var2;
Var1=10;
  Var1=Var2;
```

فسيظهر لنا خطأ عند السطر الأخير لأن المعالج سيحاول نقل ما في المتغير Var2 إلى المتغير Var1 ,ولكن المتغير Var2 لا يحتوي على أي قيمة لذلك سيظهر لنا الخطأ .

بعض أنواع المتغيرات وكيفية تعريفها:

رقم صحيح int

نص أو كلمة string

رقم حقيقي رقم يقبل الكسور العشرية مثل" 41.51" double

حرف char

تاريخ أو وقت Date Time

قيمة بوليا نية bool

معنى القيمة البوليانية أي أن المتغير يحمل قيمة صح أو خطأ , للتوضيح:

كود

```
int k;
string w;
double m;
char n;
bool p;
k=15;
w="نص هذا";
m=12.15;
n='م';
p=true;
```

لا حظ أن المتغير من نوع رقم لا يحتاج إلى علامتي اقتباس ,بينما المتغير من نوع نص أو حرف يحتاج إلى علامتي اقتباس , المتغير من نوع bool يحمل أما قيمة true أو false

ماذا لو جربنا الكود التالي:

كود

```
k="نص";
```

سيظهر لنا خطأ لأنا نحاول أن نحمل المتغير k قيمة من نوع نص ,بينما المفروض أن يحمل قيمة من نوع رقم كما عرفناه

وأيضاً هذا الكود خاطئ لنفس السبب:

كود

```
k=w;
```

حيث أن w يحتوي على قيمة من نوع نص و k المفروض أن يحمل قيمة من نوع رقم ,ماذا لو كان لدينا هذا الكود:

كود

```
w="45";
  k=w;
```

سيظهر لنا خطأ أيضاً عند السطر الثاني بالرغم من أن w يحمل رقم ,لأن المترجم لا يعرف أن w يحمل رقم , فهو يعتبر كل ما هو بين علامتي اقتباس على أنه نص ,وبالتالي فالمترجم لن يقبل أن يحمل k قيمة نصية لأن من المفروض أن يحمل رقم ,للتغلب على هذه المشكلة يوجد في بيئة التطوير كائن للتحويل ما بين الأنواع ,وهذا الكائن يقوم بتحويل القيمة إلى أي نوع فإذا أردنا تنفيذ الكود السابق باستخدام كائن التحويل:

كود

```
w="45"; k=Convert.ToInt32(w);
```

الكائن Convert قام بأخذ قيمة w وحولها إلى رقم ثم حملها للمتغير ,لاحظ أننا للوصول إلى أفعال أو صفات الكائن نكتب اسم الكائن ثم نقطة ثم فعل أو صفة الكائن ,فالكائن Convert يحتوي على فعل ToInt32 لذلك كتبنا اسم الكائن ثم نقطه ثم فعل الكائن ,ووضعنا بين القوسين المدخلات , ومن أفعال الكائن Convert:

ToInt32 للتحويل إلى رقم صحيح كما رأينا في المثال السابق

ToString للتحويل إلى نص

ToDouble للتحويل إلى عدد عشري

ToChar للتحويل إلى حرف

ToBoolean للتحويل إلى قيمة بوليا نية

ملاحظة: الرقم الأخير من المتغير ToInt32 يعني حجمه في الذاكرة, مثلاً int16 حجمه 16 بت و int32 حجمه 32 بت ويوجد أيضاً 64, وكل نوع له مدى محدد لا يستطيع يتجاوزه.

مثلاً int16 يستطيع خزن أي رقم من – 32,768 إلى 32,767

و int32 يستطيع خزن أي رقم من – 2,147,483,648 إلى 2,147,483,647

و int64 له مدى أكبر بكثير, لذلك اخترنا int32 كحل وسط , لأنه يخزن أرقام كبيرة , ولا يستهلك مكان كبير في الذاكرة.

تطبيق الدرس الثاني:

أفتح مشروع التطبيق الذي طبقنا عليه في الدرس الماضي غير النص المكتوب في الزر من "ترحيب" إلى "إظهار قيمة x" من خلال تعديل الصفة Text من جدول الصفات , انقر نقراً مزدوجا على الزر لتنتقل إلى شفرة الزر .. ثم قم بحذف سطر الأوامر التالي:

كود

```
MessageBox.Show("مرحباً بكم إلى سي شارب");
```

في مكان الكود المحذوف أكتب الكود التالي:

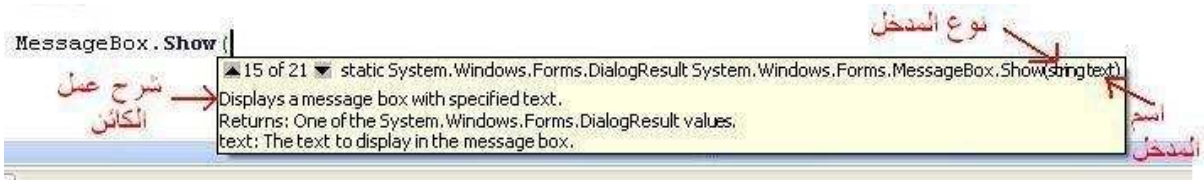
كود

```
int x;
        x=50;
        MessageBox.Show(x);
```

في السطر الأول عرفنا متغير اسمه x سيحمل قيمة من نوع رقم صحيح ,في السطر الثاني حملنا المتغير x قيمة 50 في السطر الثالث كتبنا أمر إظهار الرسالة , لاحظ أننا وضعنا x بدل نص الرسالة ,حيث أن المترجم سيقوم بإظهار رسالة ويكتب داخلها القيمة المحملة على x ,قبل تجربة البرنامج , هل تعتقد أن هناك خطأ في الشفرة السابقة؟

نعم هناك خطأ فلو جربت تشغيل البرنامج ثم الضغط على الزر ستظهر لك رسالة خطأ ,وسبب الخطأ أن كائن MessageBox.Show يحتاج إلى مدخلات من نوع نص ليظهرها في الرسالة ,وقد مررنا x على أنه المدخلات ولكنه من نوع رقم صحيح .

ربما يتساءل البعض ما أدراني أن الكائن MessageBox.Show يحتاج إلى مدخلات من نوع نص وليس من نوع رقم الأمر بسيط جداً , إذا كنت تريد إن تعرف نوع المدخلات التي يحتاجها الكائن سيظهر لك ذلك في رسالة صغيرة بمجرد أن تفتح القوس الأول كما في الصورة:



عندما انتهيت من كتابة القوس الأول ظهر صندوق المعلومات وأظهر أسماء وأنواع المدخلات التي يحتاجها الكائن السهم يشير إلى الكلمة التالية:

إقتباس

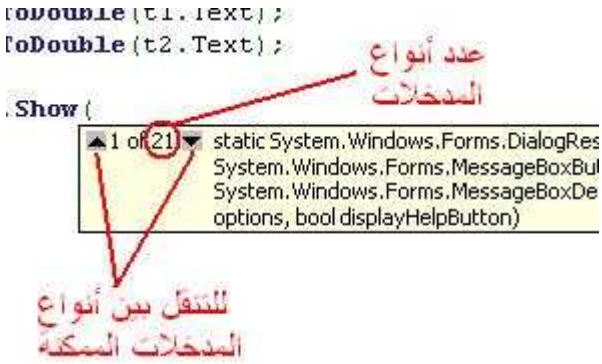
```
string text
```

الجزء الأول من الكلمة السابقة يعني نوع المدخل وهو string أي نص ,والجزء الثاني هو أسم المدخل text وهذا سنشرحه فيما بعد , ولكن ما بهمنا هو الجزء الأول لنعرف نوع المدخل الذي يحتاجه الكائن .

ملاحظة:بعض الكائنات ومنها كائن MessageBox.Show يمكن أن تقبل مدخلات متعددة فيمكن أن تقبل مدخل واحد وهو نص الرسالة كما رأينا في الدرس السابق و يمكن أن تقبل أكثر من مدخل .

الكائن MessageBox.Show يقبل 21 نوع من المدخلات.

ومن الممكن أن تتجول بين أنواع المدخلات الممكنة للكائن بالضغط على زر الأسهم فوق وتحت في لوحة المفاتيح عند ظهور صندوق معلومات المدخلات كما في الصورة:



ولكن ما بهمنا حالياً من أنواع المدخلات هو النوع رقم 15 الذي يحتاج فقط لنص الرسالة ,نعود إلى الكود السابق الذي أظهر لنا خطأ نتيجة تضارب أنواع البيانات .

لحل مشكلة التضارب نستخدم كائن التحويل لتحويل قيمة x إلى نص ثم تمريرها إلى الكائن

MessageBox.Show

لأنه كما قلنا لا يقبل إلا مدخلات من نوع نص و x من نوع رقم ,الكائن الذي يحول من رقم إلى نص هو

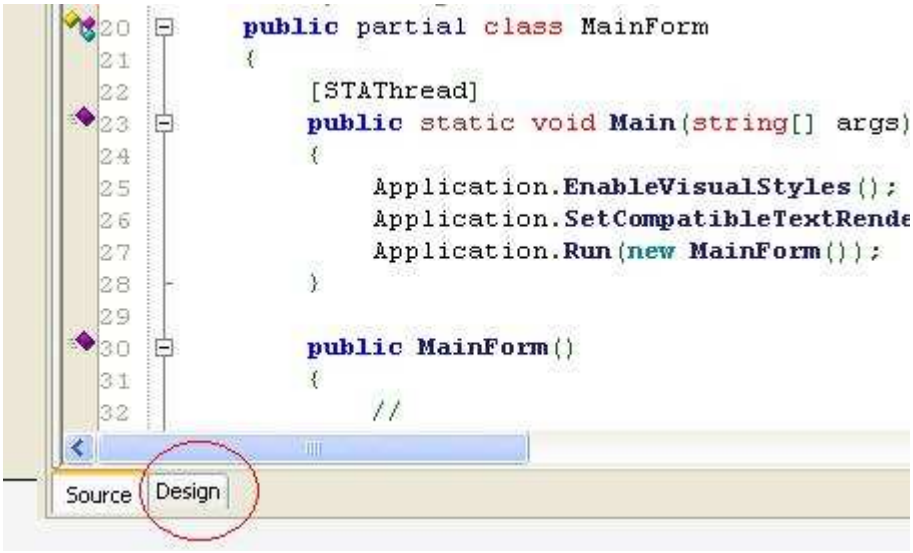
Convert.ToString

وبالتالي سيصبح الكود الصحيح هكذا:

كود

```
int x;
        x=50;
        MessageBox.Show(Convert.ToString(x));
```

جرب البرنامج الآن واضغط على الزر ستظهر لك رسالة مكتوب عليها 50 ,وهكذا تغلبنا على مشكلة تضارب البيانات لتوسيع التطبيق سنستخدم كائن آخر من كائنات البيئة وهو مربع النص ,اضغط على زر Design أسفل الشفرة لترجع إلى نافذة التصميم :



الآن قم بسحب كائن مربع النص وأفلته فوق النافذة كما سحبت الزر في الدرس الأول

ليصبح شكل النافذة هكذا:

تأكد من اختيار مربع النص واذهب إلى جدول الصفات ,أبحث عن الصفة (Name) ستجد مكتوباً أمامها textBox1 غيرها إلى t1 .

لاحظ أنا غيرنا أسم الكائن من textBox1 إلى t1 وذلك لسهولة التعامل معه في الشفرة كما سنرى لاحقاً.

الآن اضغط مزدوجاً على الزر لترجع إلى الشفرة ,أمسح السطر الثاني من الكود السابق.

وأكتب بدل منه هذا الكود

كود

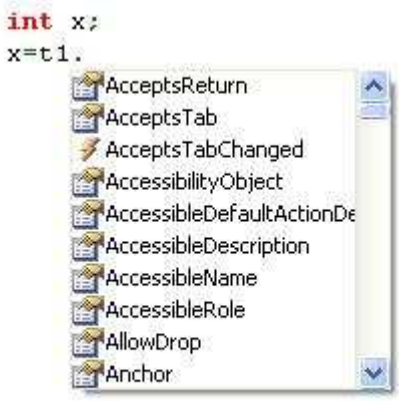
```
x=t1.Text;
```

لاحظ أن الكائن t1 الذي هو مربع النص له صفة اسمها Text وهي تمثل النص المكتوب عليه ,معنى الكود السابق أن المترجم سيقوم بأخذ ما هو مكتوب في مربع النص المسمى t1 ويحمله للمتغير x ,أي أن الكود كاملا سيصبح هكذا:

كود

```
int x;
      x=t1.Text();
      MessageBox.Show(Convert.ToString(x));
```

ملاحظة:لمعرفة صفات وأفعال الكائن أكتب أسم الكائن ثم أكتب نقطة , ستظهر لك نافذة بجميع أعمال وصفات الكائن ففي المثال السابق بمجر د كتابة تكتب t1 ثم نقطة ,سيظهر لك هذا الصندوق :



وبمجرد أن تكتب حرف t من كلمة Text سينقل الاختيار في الصندوق إلى كلمة Text:

وللاختصار اضغط على زر Tab لكي تكتب كلمة Text آلياً

لاحظ في الصورة السابقة أن نوع صفة Text هو string أي نص ,وبالتالي لو حاولت تنفيذ البرنامج وضغط الزر سيظهر خطأ لأننا حاولنا أن نحمل المتغير x قيمة نصيه ,بينما يجب أن نحمله رقم , يأتي دور كائن التحويل مرة أخرى ولكن هذه المرة من نص إلى رقم ,فيصبح الكود الكامل والصحيح هكذا:

كود

```
int x;
    x=Convert.ToInt32(t1.Text);
    MessageBox.Show(Convert.ToString(x));
```

في السطر الأول عرفنا متغير من نوع رقم ,وفي السطر الثاني حملنا هذا المتغير القيمة المكتوبة في مربع النص وحولناها إلى رقم ,وفي السطر الثالث لم نغير شيء.

الآن نفذ البرنامج , وقبل أن تضغط الزر أكتب أي رقم في مربع النص ثم اضغط الزر ,ستظهر لك رسالة بالرقم الذي كتبته تأكد انك ستكتب رقم وليس حرف وإلا ظهرت لك رسالة خطأ ,لتوسيع التطبيق أضف متغيرين آخرين y,z وأضف مربع نص آخر وسمه t2 ,غير النص المكتوب على الزر وأكتب بدلاً منه "اظهار ناتج الجمع" .

ليكون شكل النافذة هكذا:

الآن اذهب إلى الشفرة وعدلها إلى التالي:

كود

```
int x;
        int y;
        int z;
        x=Convert.ToInt32(t1.Text);
       y=Convert.ToInt32(t2.Text);
        z=x+y;
        MessageBox.Show(Convert.ToString(z));
```

عرفنا ثلاثة متغيرات من نوع رقم صحيح , في السطر الرابع حملنا x الرقم المكتوب في t1 , وفي السطر الخامس حملنا y الرقم المكتوب في t2 , وفي السطر السادس حملنا z مجموع x وy , وفي السطر السابع أظهرنا قيمة z في الرسالة , لاحظ أنا وضعنا z بدل x في السطر الأخير , الآن قم بتنفيذ البرنامج , وأكتب عدداً صحيحاً في كل مربع نص وأضغط الزر.

ستظهر لك نافذة مجموع العددين :

أغلق الرسالة وجرب أن تضع في أحد مربعات النص رقماً حقيقياً مثل 41.25 واضغط على الزر ,ستظهر لك رسالة خطأ لأن المتغيرات xوyوz تحمل متغيرات من نوع رقم صحيح وأنت حاولت أن تحملها رقم حقيقي , اضغط على زر Continue في رسالة الخطأ

ليتم إيقاف تنفيذ البرنامج:

أذهب إلى الشفرة وحاول أن تجد الحل للقيام بجمع أعداد حقيقية, وذلك هو الواجب.

تلميح:نعرف أن double تعني رقم حقيقي والكائن Convert.ToDouble يحول إلى رقم حقيقي ,ما عليك إلا أن تضع double بدل من int , وتضعConvert.ToDouble بدل من Convert.ToInt32

لحظة , لم ينتهي الواجب بعد , أضف إلى البرنامج ثلاثة أزرار إضافية للقيام بباقي العمليات الرياضية

1-إظهار ناتج الطرح

2-إظهار ناتج الضرب

3-إظهار ناتج القسمة

ليصبح شكل النافذة هكذا:

نهاية الدرس الثاني.

الدرس الثالث

الجمل الشرطية:

معظم عمل الحاسوب يقوم على المقارنة بين قيمتين لاتخاذ أي قرار ,بعبارة أدق أي عمل يقوم به الحاسوب هو عبارة عن عدد من المقارنات, بين قيم مخزنة في مسجلات المعالج .

مسجلات المعالج هي عبارة عن أجزاء من المعالج يمكن أن تخزن قيم صغيرة.

وكل مسجل يستطيع أن يخزن 16 بت , وهذه القيم المخزنة يقوم المعالج بمقارنتها, مع قيم أخرى في مسجلات أخرى من نفس المعالج لإنتاج تيار كهربائي يمثل نتيجة المقارنة.

إذاً مبدأ عمل الحاسوب هو المقارنة , عرفنا المقارنة على مستوى المعالج ,ولكن هذا لا يهمنا , وما يهمنا هو المقارنة على مستوى لغة البرمجة ,والمقارنة تعتبر أحد أهم أعمدة إي لغة من اللغات , وهناك عدة صيغ للمقارنة:

1 - إذا كان (الشرط) فـ

{التعليمات التي تنفذ في حالة توفر الشرط}

كود

```
if (x>y)
{
  MessageBox.Show("من واي إكس أكبر");
}
```

في الكود السابق سيقوم المعالج بإظهار الرسالة إذا كان x أكبر من y وإلا فلن تظهر أي رسالة.

لاحظ أننا لم نضع الفاصلة المنقوطة بعد الشرط لأن المعالج لم يكمل جملة الشرط ويجب أن ندعه يكمل الجملة إلى نهاية الحاصرة الثانية.

2-إذا كان (الشرط) فـ

{التعليمات التي تنفذ في حالة توفر الشرط}

وإلا

{التعليمات التي تنفذ في حالة عدم توفر الشرط}

كود

```
If (x<y)
{
MessageBox.Show("واي إكس أكبر من");
}
else
{
MessageBox.Show("إكس ليس أكبر من واي");
}
```

في الكود السابق سيقوم المعالج بإظهار رسالة "إكس أكبر من واي" إذا كان x أكبر من y ,أما إذا لم يكن x اكبر من y فستظهر رسالة "إكس ليس أكبر من واي" ,لاحظ أننا لم نضع فاصلة منقوطة بعد كلمة else لأن التعليمة لم تكتمل بعد.

3-إذا كان(الشرط1) فـ

{التعليمات التي تنفذ عن توفر الشرط1}

وإلا إذا كان (الشرط 2) فـ

{التعليمات التي تنفذ في حالة عدم توفر الشرط1 و توفر الشرط 2}

وإلا إذا كان ...

كود

```
if (x>5)
{
MessageBox.Show("خمسة اكس أكبر من");
}
else if(x<4)
{
MessageBox.Show("وأكس أصغر من أربعة , اكس ليس أكبر من خمسة");
}
..
```

في الكود السابق يبدأ المعالج بفحص الشرط الأول فإذا تحقق يظهر رسالة "اكس أكبر من خمسة" ,وإن لم يتحقق فإنه ينتقل إلى الشرط الثاني فإذا تحقق سيظهر رسالة "اكس ليس أكبر من خمسة , وأكس أصغر من أربعة", وهكذا حتى ينتهي من جميع الشروط .. لاحظ أنه من الممكن أن يكون في الجملة أكثر من شرطين.

هناك عدد من معاملات المقارنة بالإضافة إلى < و >:

< أكبر من

مثل ما رأينا سابقاً

> أصغر من

مثل ما رأينا سابقاً

== يساوي

كود

```
if (x == y)
{
MessageBox.Show("واي اكس يساوي");
}
```

=! لا يساوي

كود

```
if (x != y)
{
MessageBox.Show("واي لا يساوي اكس");
}
```

أحياناً نحتاج إلى شرط مزدوج , أي أننا نحتاج إلى شرط مكون من جزئيين .

مثلاً إذا كان x أكبر من y و x أصغر من z

كود

```
if (x>y & x<z)
{
MessageBox.Show("زد اكس أكبر من واي و أصغر من");
}
```

لاحظ إننا وضعنا الشرطين بينها علامة &

أما إذا أردنا شرط مزدوج من نوع آخر :

مثلاً إذا كان x أكبر من y أو x يساوي 10

كود

```
if (x>y | x==10)
{
MessageBox.Show("أكبر من واي أو يساوي 10 اكس");
}
```

لاحظ إننا وضعنا الشرطين بينها علامة |

إذا (&) تعني (و)

و (|) تعني (أو)

عند فصل الشرطين بـ | أو & فإن المترجم يقوم بفحص الشرطين مهما كانت نتيجة الشرط الأول ,وهذا يعتبر إهدار لوقت المترجم و جهده في بعض الحالات , فمثلاً الشرط : إذا كان x أكبر من y أو x يساوي 10.

في هذا الشرط المركب لو تحقق الشرط الأول , فالجملة المنطقية كلها صحيحة وبالتالي فلا داعي لأن يفحص المترجم الشرط الثاني ,ولتنفيذ هذا الفحص الانتقائي يجب تكرار علامة الشرط بين الشرطين , فالعلامة | يجب أن تكون || و & يجب أن تكون && :

كود

```
if (x>y && x<z)
{
MessageBox.Show("زد اكس أكبر من واي و أصغر من");
}
```

كود

```
if (x>y || x==10)
{
MessageBox.Show("أكبر من واي أو يساوي 10 اكس");
}
```

ملاحظة : علامة | يمكن طباعتها بالضغط على Shift مع زر \ والذي عادة ما يأتي بجانب زر المسح BackSpace وفوق زر Enter

الدوارات (الحلقات):

هناك أنواع كثيرة من الدوارات لكننا سنتعرف حالياً على أهم نوع منها ,والدوارات هي تعليمات فائدتها تكرار تعليمات محدد لعدد محدد من المرات , مثلاً إذا أردنا أن نظهر رسالة ما عشر مرات أو عشرين مرة , لن نكتب التعليمة عشر أو عشرين مرة. حيث يستخدم المعالج متغير من نوع رقم كعداد وفي كل مرة تنفذ العملية يزداد العداد رقم واحد ,وهكذا حتى يتوفر شرط وقوف لدواره , وأحياناً قد يكون عداد عكسي يعني أنه في كل مرة ينقص رقم ,إذاً هناك أربعة أشياء مهمة لجملة الدوارة For

1- متغير من نوع عدد يستخدم كعداد مع تزويده برقم ابتدائي يبدأ منه.

2- شرط استمرار الدوارة في العمل.

3- تعريف نوع العداد هل هو تصاعدي أم تنازلي.

4- التعليمات التي تنفذ.

كود

```
for (int i = 0;i < 10;i++)
{
MessageBox.Show("الدوارة من");
}
```

في الكود السابق كلمة for تستخدم لتعريف الدوارة ,بعد فتح القوس وضعنا تعريف للعداد int i=0 ووضعنا له قيمة 0 أي أن العداد سيبدأ من 0 ,وضعنا فاصلة منقوطة و ضعنا بعدها شرط الاستمرار في العمل وهو أن يكون i أصغر من عشرة, وضعنا فاصلة منقوطة ووضعنا بعدها نوع العداد وهو تصاعدي i++ أما إذا كان تنازلي نضع i– ,ثم أغلقنا القوس وفتحنا حاصرتين بينهما التعليمات التي سينفذها المعالج .

معنى الدوارة السابقة أن المعالج سيظهر الرسالة للمرة الأولى ثم يزيد i ويفحص شرط الاستمرار ,أذا تحقق شرط الاستمرار يظهر الرسالة مرة أخرى ويزيد العداد برقم ويفحص شرط الاستمرار .....

في الكود السابق سيقوم المعالج بإظهار الرسالة عشر مرات

ملاحظة : يمكن استخدام متغير العداد داخل التعليمات

كود

```
for (int i = 0;i < 10;i++)
{
MessageBox.Show(Convert.ToString(i));
}
```

في الكود السابق سيظهر المعالج الرسالة للمرة الأولى وفيها قيمة العداد الأولية وهي 0 ,وفي المرة الثانية سيظهر الرسالة وفيها قيمة العداد بعد أن زاد برقم أي 1 ,وهكذا حتى العدد 9 , وعندما يزد العداد بعدها برقم أي يصبح 10 سيفحص المعالج شرط الاستمرار .

وهو أن يكون المتغير أصغر من 10 ولكن المتغير هذه المرة ليس أصغر من عشرة لذلك سيتوقف المعالج عن تنفيذ التعليمات ويخرج من جملة For.

تطبيق الدرس الثالث:

أفتح تطبيق الدرس السابق :

وقم بحذف صندوقي النص بواسطة زر Delete

غير النص المكتوب في الزر إلى "تشغيل الدوارة"

من صندوق الأدوات قم بسحب أداة القائمة :

من خلال الخاصية (Name) قم بتغيير أسم القائمة من ListBox1 إلى l1

ليسهل التعامل معها من الشفرة.

ملاحظة: عند عمل تطبيق باللغة العربية يجب تحويل الأدوات في النافذة للترتيب العربي ,أي من اليمين إلى اليسار ويتم ذلك عن طريق تحديد النافذة وتغيير صفتين من صفاتها.

وهي صفات RightToLeft و صفة RightToLeftLayout من جدول الصفات. حيث تكون الخاصية الأولى Yes والخاصية الثانية True :

عندها ستلاحظ إن اتجاه الأدوات تغير ويصبح شكل النافذة هكذا :

الآن اضغط مزدوجاً على الزر للذهاب للشفرة قم بحذف الكود الموجود مسبقاً بين الحاصرتين

وأكتب بدلاً عنه هذا الكود:

كود

```
for (int i=0;i<10;i++)
{
l1.Items.Add("  من الدوارة  ");
}
```

قمنا بعمل دوارة من الصفر إلى 10 لإضافة كلمة "من الدوارة" إلى القائمة الكائن l1.Items.Add يقوم بإضافة ما داخل القوس للقائمة l1 .

وبالتالي فالكود السابق سيقوم بكتابه كلمة "من الدوارة" عشر مرات على القائمة ,جرب تشغيل البرنامج وأضغط على الزر

.

.

.

.

يجب أن ترى شيئاً كهذا:

ملاحظة : إذا أردنا أن نجمع كلمتين أو نصين في نص واحد فيمكننا استخدام إشارة +

مثلاً إذا أردنا أن نظر رسالة تقول مرحباً بكم فيمكن عمل ذلك هكذا

كود

```
MessageBox.Show("مرحباً" + "بكم");
```

هناك كلمتين ربطنا بينهما بعلامة +

والآن لنعدل التطبيق قليلاً لكي نجرب استخدام العداد :

قم بتعديل كود الدوارة وأضف إلى كلمة "من الدوارة" أضف العداد هكذا:

كود

```
for (int i=0;i<10;i++)
{
l1.Items.Add("  من الدوارة  " + Convert.ToString(i));
}
```

بما أن العداد من نوع رقم استخدمنا كائن التحويل لنحوله إلى نص ثم ربطناه بكلمة "من الدوارة"

عند تشغيل البرنامج سترى شيئاً كهذا :

أعتقد أن الصورة لا تحتاج للشرح ...

الآن لماذا لا نطبق جملة شرطية داخل الدوارة ..

سنكتب شفرة لطباعة كلمة "من الدوارة"

و إذا كان العداد أكبر من خمسة سنضيف كلمة "أكبر من خمسة" :

قم بتغيير كود الدوارة إلى هذا الكود:

كود

```
for (int i=0;i<10;i++)
{
if (i>5)
{
l1.Items.Add("  من الدوارة  " + Convert.ToString(i) + "  أكبر من خمسة  ");
}
else
{
l1.Items.Add("  من الدوارة  " + Convert.ToString(i) );
}
}
```

الكود باللون الأحمر سينفذ عشر مرات .

والكود بالخط المائل سينفذ إذا كان العداد i أكبر من خمسة.

والكود الذي تحته خط سينفذ إذا كان العداد i أصغر من أو يساوي خمسة .

عند تنفيذ البرنامج ستكون النافذة هكذا :

واجب الدرس الثالث:

الواجب صعب قليلاً هذه المرة ...

أستخدم الدوارة والجمل الشرطية لتنفيذ عملية إضافة نصوص للقائمة 12 مرة ,بحيث يكتب في آخر كل نص ما إذا كان العداد أكبر من 6 أو يساوي 6 أو أصغر من 6 .

لتكون النتيجة هكذا:

تلميح : ستستخدم الجمل الشرطية من النوع الثالث بثلاث شروط

أي أن جملة الشرط ستكون شبيهة بهذه:

كود

```
if (...)
{
...
}
else if(...)
{
...
}
else if(...)
{
...
}
```

أكمل الفراغات , طبعاً كل هذا داخل الدوارة

نهاية الدرس الثالث.

..

الدرس الرابع

الإجراءات Procedures

الإجراء هو مجموعة أوامر ينفذها المعالج مكتوبة بين حاصرتين { } ,ويتم تعرفها في مكان واحد في البرنامج ويمكن استدعائها عدة مرات .

والإجراء له ثلاثة أشياء مهمة :

1- تعريف الاسمDeceleration Name

2-المدخلاتParameters

3-الشفرة مكتوبة بين حاصرتين Code

فمثلاً عند الدخول إلى شفرة زر رسالة الترحيب في الدرس الأول ,سنرى هذا الكود:

كود

```
void Button1Click(object sender, System.EventArgs e)
 {
 MessageBox.Show("مرحباً بكم في السي شارب");
 }
```

الكود السابق يمثل إجراء ينفذه المعالج عند الضغط على الزر ,نلاحظ انه يحتوي على الثلاث الأشياء المهمة للإجراء وهي **تعريف** الاسم:

كود

```
void Button1Click
```

و **المدخلات**:

كود

```
void Button1Click(object sender, System.EventArgs e)
```

والشفرة مكتوبة بين حاصرتين:

كود

```
{
   MessageBox.Show("مرحبا بكم في السي شارب");
  }
```

في التعريف void تعني إجراء أي أننا سنعرف متغير من نوع إجراء ,كما عرفنا متغير من نوع رقم في الدرس السابق:

كود

```
int x;
```

بنفس الطريقة عند تعريف الإجراء:

كود

```
void x;
```

لكن هذا التعريف لا يكفي فهو لا يحتوي على الثلاث الأشياء المهمة ,مثلاً لو أردنا إجراء يظهر رسالة مكتوب عليها مثلاً "من داخل الإجراء"

فسكون التعريف كالتالي:

كود

```
void sm()
  {
  MessageBox.Show("من داخل الإجراء");
  }
```

عرفنا متغير من نوع إجراء اسمه sm اختصار لـ Show Message ..يمكنك تغيير أسم الإجراء كما تشاء .

يوجد لدينا الاسم:

كود

```
void sm
```

و المدخلات:

كود

```
()
```

لا يوجد لدينا مدخلات في هذا الإجراء ولكن يجب عمل الأقواس الفارغة ,وسنعرف فيما بعد كيف نتعامل مع المدخلات

ولدينا الشفرة:

كود

```
{
  MessageBox.Show("من داخل الإجراء");
  }
```

وبالتالي فالتعريف كامل ويمثل إجراء لإظهار رسالة , والآن إذا أردنا في أي مكان من البرنامج تنفيذ الإجراء فما علينا سوى كتابة اسمه هكذا:

كود

```
sm();
```

التعليمة السابقة تجعل المعالج يبحث عن الإجراء sm وينفذ الأوامر بداخلة. ربما يقول البعض ما الداعي من عمل إجراء وكتابة الكود داخلة ثم استدعاءه .

لماذا لا نكتب الكود مباشرة كما فعلنا في الدرس الأول؟؟؟

والجواب ببساطه ماذا لو أردت إظهار الرسالة خمس مرات؟؟

بالطريقة القديمة ستكتب هكذا:

كود

```
void Button1Click(object sender, System.EventArgs e)
 {
 MessageBox.Show("من داخل الإجراء");
 MessageBox.Show("الإجراء من داخل");
 MessageBox.Show("من داخل الإجراء");
 MessageBox.Show("داخل الإجراء من");
 MessageBox.Show("من داخل الإجراء");
 }
```

أما بطريقة الإجراءات فستكتب هكذا:

كود

```
void Button1Click(object sender, System.EventArgs e)
 {
  sm();
  sm();
  sm();
  sm();
  sm();
  }

  void sm()
 {
  MessageBox.Show("من داخل الإجراء");
 }
```

الكود بالأزرق هو كود الضغط على الزر و الكود بالأحمر هو كود إجراء إظهار الرسالة ,نلاحظ أن طريقة الإجراءات تسهل التعامل أكثر وتوفر الوقت والجهد ,ربما يقول البعض لا يوجد فرق كبير بين الطريقتين لكن ماذا إذا كانت الرسالة كبيرة جداً أو ماذا إذا كانت الأوامر أكثر من مجرد إظهار رسالة ؟

عندها بالتأكيد الإجراء سيوفر الكثير من الجهد والوقت!

وأيضا المدخلات تجعل الإجراءات أكثر فائدة :

مثلاً لو أردنا أن نطور الإجراء السابق ليظهر رسالة مختلفة في كل مرة نستدعيه .سنجعل الرسالة المراد إظهارها علي أنها مدخل متغير ويكون تعريف الإجراء كالتالي:

كود

```
void sm(string msg)
  {
  MessageBox.Show(msg);
  }
```

المدخلات هي متغيرات كما رأينا حيث عرفنا متغير مدخل اسمه msg من نوع نص string, وفي تعليمة إظهار الرسالة جعلنا الكائن MessageBox.Show يظهر ما في المتغير المدخل msg, الآن عند استدعاء الإجراء يجب تمرير قيمة من نوع نص مكان المتغير المدخل هكذا:

كود

```
sm("من داخل الإجراء");
sm("من نفس الإجراء لكن رسالة مختلفة");
```

نلاحظ أن الإجراء أظهر رسائل مختلفة بنفس الشفرة, الأمر الأول في الكود السابق سيظهر رسالة مكتوب عليها

"من داخل الإجراء"

والأمر الثاني سيظهر رسالة مكتوب عليها "من نفس الإجراء لكن رسالة مختلفة"

ملاحظة: إذا كان لدينا أكثر من مدخل فنفصل بينها بالفاصلة مثلاً:

كود

```
void sm(int x,string msg)
```

وعند الاستدعاء :

كود

```
sm(14,"رسالة");
```

Functions الدوال

الدول هي إجراءات لكن لها خاصية أنها ترجع قيم ,رأينا في الإجراءات أنا نستدعيها للقيام بعمل محدد فقط.

أما الدوال فإنا نستدعيها للقيام بعمل محدد وإرجاع قيمة لنا.

ولها خمس أشياء مهمة:

1- تعريف الاسم Name Deceleration

2-نوع القيمة التي ستحملها الدالة Result Data Type

3-المدخلات Parameters

4-الشفرة مكتوبة بين حاصرتين Code

5-تعليمة تحميل الدالة للقيمة الناتجة Phrase Return

مثلاً إذا أردنا عمل دالة تقوم بأخذ رقمين وترجع لنا مجموعهما فيكون التعريف كما يأتي:

كود

```
int sum(int x,int y)
  {
  int z;
  z=x+y;
  return z;
 }
```

في السطر الأول عرفنا دالة أسمها sum ترجع متغير من نوع رقم int ,ولها مدخلين من نوع رقم x و y ,وبين الحاصرتين لدينا التعليمات التي تقوم بجمع المتغيرين ,وفي التعليمة الأخيرة z return تعني تحميل الدالة للقيمة الموجود في z وهي مجموع xوy . وعند الاستدعاء بنفس الطريقة في الإجراءات لكن الدالة تكون محملة بقيمة :

كود

```
sum(4,5);
```

بدون الدالة ستكود شفرة جمع رقمين هكذا:

كود

```
int z;
   z=4+5;
   MessageBox.Show(Convert.ToString(z));
```

أما باستخدام الدالة فتكون الشفرة هكذا:

كود

```
int z;
   z=sum(4,5);
   MessageBox.Show(Convert.ToString(z));
```

لاحظ أنا استدعينا الدالة وحملنا ما داخلها للمتغير z ,في المثال السابق تعرفنا كيف نستخدم الدالة , رغم إننا لم نستفد منها كثيراً في المثال السابق ,ولكن كان الغرض منه توضيح استخدام الدوال .

عزل - كبسلة المتغيرات Encapsulation Variables

مار أيك بالكود التالي:

كود

```
void Button1Click(object sender, System.EventArgs e)
 {
 int x;
 x=10;
 AddFive();
 }

 void AddFive()
 {
 int y;
 y=5+x;
 MessageBox.Show(Convert.ToString(y));
 }
```

الكود بالأزرق هو إجراء الضغط على الزر , وبالأحمر إجراء إظهار قيمة x مضاف إليها 5.

لو قمت بتنفيذ الكود السابق , ما الذي ينتج , ستقول ستظهر رسالة مكتوب عليها 15؟

سأقول لا , لا ليس صحيحاً , سينتج خطأ جسيم !!

السبب أن المتغير x معرف داخل الإجراء باللون الأزرق ,وبالتالي فالمفروض أن نستخدمه في هذا الإجراء فقط , وإذا استخدمناه في إجراء آخر كما فعلنا .

فإن المعالج لن يتعرف على هذا المتغير !! .. ولكن ماذا لو أردنا أن نستخدم متغير في أكثر من إجراء؟؟

الجواب:

المتغيرات Variables تنقسم إلى نوعين رئيسيين :

1-متغيرات محلية Local وهي التي تعرف وتستخدم داخل الإجراء فقط.

2-متغيرات عامة Global وهي التي تعرف خارج الإجراء وتستخدم من قبل إجراءات كثيرة.

والمتغيرات العامة تنقسم إلى قسمين :

a- متغيرات خاصة private وهي تستخدم من قبل إجراءات ودوال النافذة الواحدة فقط ولا تراها النوافذ الأخرى:

كود

```
private int x;
```

b-متغيرات عامة public وهي تستخدم من قبل إجراءات ودوال النافذة والنوافذ الأخرى

كود

```
public int x;
```

إذن إذا أردنا أن نستخدم متغير في عدة إجراءات فإننا نعرفه خارج الإجراء ,والأفضل أن يكون تعريف المتغيرات العامة في بداية كود النافذة فوق دالة Main :

كود

```
//الدولية هنا تعريف المتغيرات
  [STAThread]
  public static void Main(string[] args)
  .
  .
```

وبالتالي فالإجراء السابق بعد التعديل سيصبح هكذا :

كود

```
private int x;
 .
 .
 void Button1Click(object sender, System.EventArgs e)
 {
 x=10;
 AddFive();
 }

 void AddFive()
 {
 int y;
 y=5+x;
 MessageBox.Show(Convert.ToString(y));
 }
```

طبعاً بفرض أننا وضعنا تعريف المتغير x فوق دالة Main ,بهذه الطريقة عرفنا x مرة واحدة ويمكن استخدامه في أكثر من إجراء من نفس النافذة .

ملاحظة: إذا أردنا أن نكتب تعليق في الشفرة أو ملاحظة بدون أن يقرأها المترجم ,فإننا نكتب في بداية السطر // والمترجم لن يقرأ الكود التي تحول لونها إلى الأخضر كما فعلنا في الكود السابق , أما إذا كانت الملاحظة من عدة أسطر ولا نريد أن نضيف // لكل سطر نتيجة الكسل أو ما شابه ,فيكفي أن نكتب*/ في بداية الملاحظة و/* في نهاية الملاحظة :

كود

```
int x;
 /* المترجم هذا السطر لن يقرأه
 وهذا السطر ايضا
 وهذا أيضاً*/
 MessageBox.....
```

ملحق 1 :

Appendix 1 : Error Handling تعقب الأخطاء

كثيراً ما تحدث أخطاء في البرامج نتيجة تعامل المستخدم مع البرنامج , وبدون تعقب للأخطاء سيغلق البرنامج بمجرد أي خطأ , لذلك يجب على المبرمج , تعقب الأخطاء ليمنع من إغلاق البرنامج وليظهر رسالة للمستخدم بالخطأ.

مثلاً في تطبيق الآلة الحاسبة في الدرس الثاني ماذا لو أدخل المستخدم حرف بدل من رقم ؟

بالتأكيد كائن التحويل Convert.ToInt32 سيطلق خطأ لأنه لن يستطيع تحويل حرف إلى رقم ,الكود كان هكذا :

كود

```
int x;
  int y;
  int z;
x=Convert.ToInt32(t1.Text);
y=Convert.ToInt32(t2.Text);
z=x+y;
  MessageBox.Show(Convert.ToString(z));
```

وعند إدخال حرف إلى مربع النص t1 أو t2 فستظهر رسالة خطأ ويتوقف البرنامج ,لتعقب الأخطاء نستخدم جملة try و catch حيث نكتب الكود الذي يمكن أن يسبب خطأ ,ضمن جملة try ونكتب الكود الذي ينفذ في حالة حدوث خطأ ضمن جملة catch .

ويصبح الكود السابق هكذا:

كود

```
int x;
int y;
int z;
try
{
x=Convert.ToInt32(t1.Text);
y=Convert.ToInt32(t2.Text);
}
catch
{
MessageBox.Show("تأكد من إدخال أرقام فقط");
}
z=x+y;
MessageBox.Show(Convert.ToString(z));
```

الكود باللون الأزرق هو مكان احتمال وجود الخطأ لأنه المكان الذي يتم تحويل المدخلات إلى رقم ,وإذا حصل إي خطأ فإن المعالج سينتقل لتنفيذ الكود باللون الأحمر. أما إذا لم يحدث أي خطأ , فالمعالج لن ينفذ الكود بالأحمر ..

ملحق 2 :

Appendix 2 : Program Termination إنهاء البرنامج

لإنهاء البرنامج نحتاج إلى كائن يتعامل مع بيئة الدوت نت وهذه هي التعليمة :

كود

```
System.Environment.Exit(0);
```

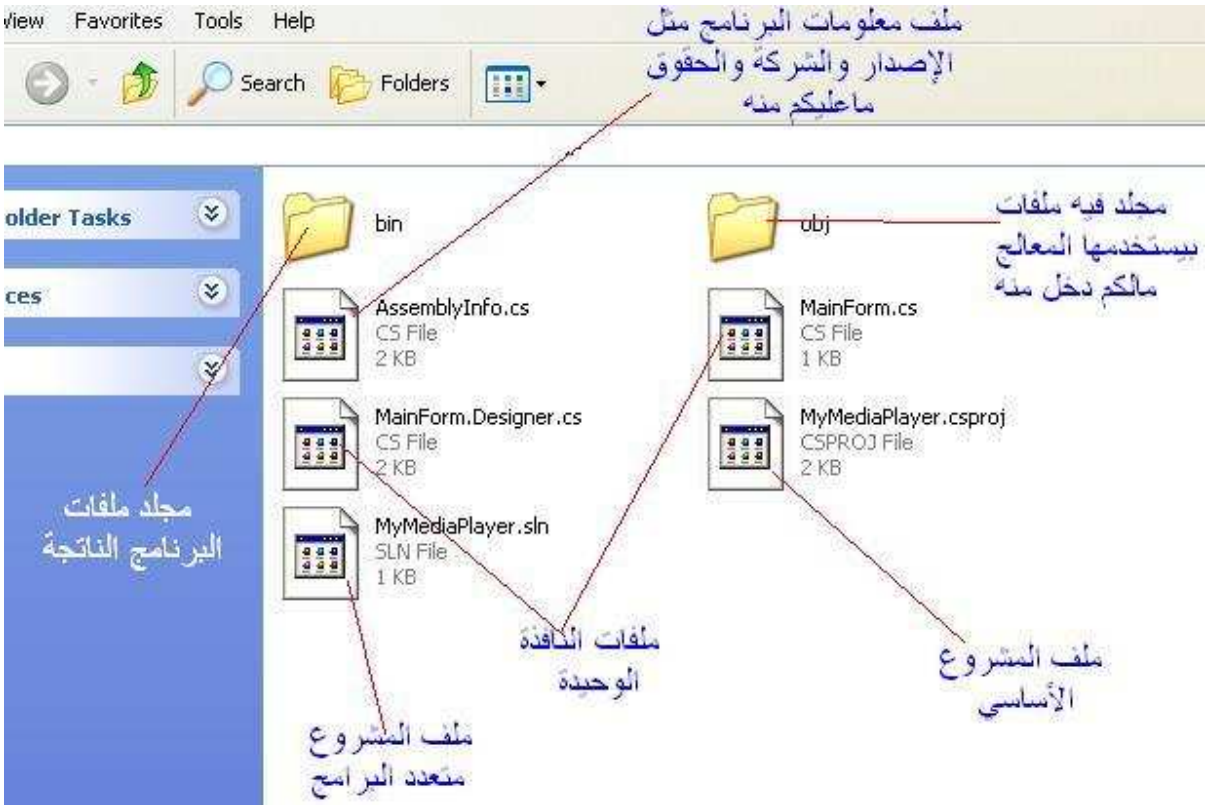
الكائن Environment يمثل بيئة الدوت نت الذي يعمل عليها البرنامج . والتعليمة Exit تجعل المعالج ينهي البرنامج ويفرغ الذاكرة References Release Memory . والمدخل 0 يعني إنهاء البرنامج تماماً و سنعرف فيما بعد مدخلات أخرى لهذه التعليمة.

ملحق 3 :

Appendix 3: Dealing with project files- التعامل مع ملفات المشروع

في لغات البرمجة المشروع ليس ملف وحيد فقط ,أنما مجموعة من الملفات المترابطة مع بعضها بواسطة ملف المشروع

وعند فتح المشروع يجب فتح الملف الأساسي للمشروع وسيقوم هو آلياً ,بتحميل جميع ملفات المشروع لبيئة التطوير :



كما في الصورة فالملف الأساسي للمشروع هو بامتداد csproj ,وإذا كان المشروع متعدد البرامج فالملف الأساسي هو بامتداد sln ,أما إذا كان المشروع برنامج واحد كما في مشروعنا فيمكن فتح أي منهما .وإذا أردت أن تنسخ المشروع فعليك نسخ المجلد بالكامل الذي يحتوي على جميع الملفات .

ثم فتح المشروع بواسطة ملف المشروع الأساسي.

تطبيق الدرس الرابع:

ربما تتساءل الكثير عن وظيفة الملفات الكثيرة التي تنتهي بامتداد dll , حان الوقت لنعرف ذلك , dll هي اختصار لـ Dynamic Link Library .

هذه الملفات كتبت بلغة برمجة و تحتوي على دوال وإجراءات وكائنات ,و تستخدم في برامج أخرى, مثلاً دالة إظهار رسالة كما استخدمناها في الدروس السابقة. موجودة في ملف اسمه System.dll ينزل مع لغة البرمجة , وبدون هذا الملف لن نستطيع إظهار الرسالة , حيث أن شفرة إظهار الرسالة ورسم صندوق الرسالة .ورسم زر موافق ورسم شريط العنوان كلها هذه الشفرة مكتوبة مسبقاً وموجود في الملف System.dll .

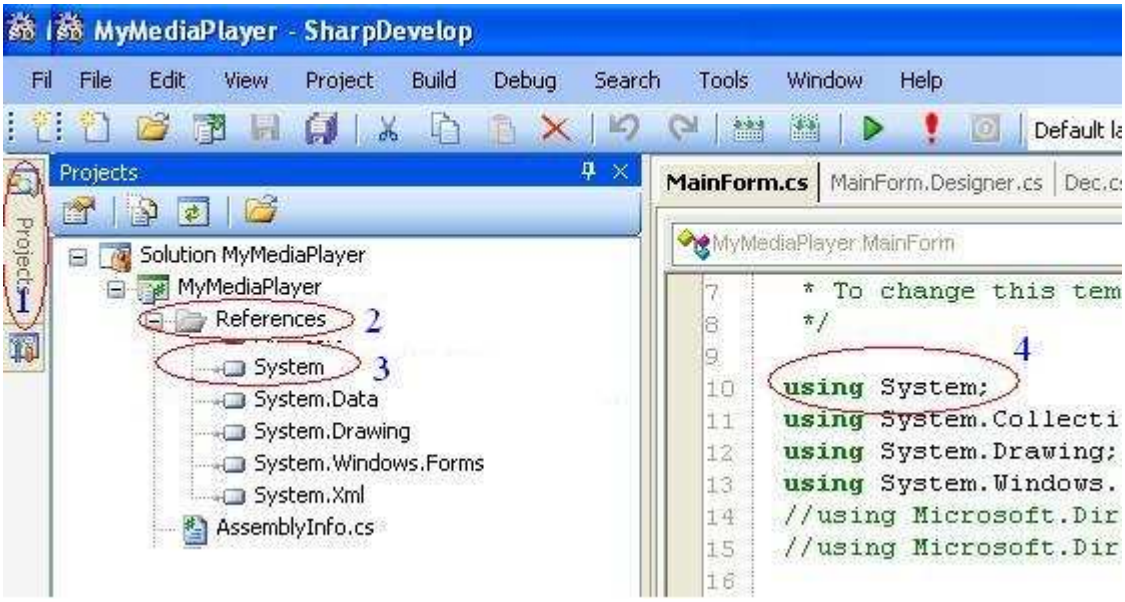
ونحن ما علينا سوى استدعائها من الملف كما نستدعي دالة أو إجراء عادي , وهناك دوال كثيرة أخري تتعلق بقواعد البيانات موجودة في الملف System.Data.dll .

وهناك دوال الرسم موجودة في System.Drawing.dll ....

إي أن الدوال و الإجراءات والكائنات في لغات البرمجة مقسمة في ملفات بغرض تنظيمها ,وعندما تريد استخدام دالة في ملف محدد يجب أن تضيف هذا الملف إلى مشروعك. حتى يتعرف علية المعالج ويستطيع استدعاء الدوال داخلة .

ربما تسأل كيف استخدمنا دالة إظهار الرسالة في الدروس السابقة ولم نظيف ملف System.dll؟

الإجابة أن لغة البرمجة تضيف هذا الملف آلياً لأنه يحتوي على دوال كثيرة الاستخدام ومهمة , ولتعرف إن هذا الملف مضاف في المشروع :



أفتح نافذة ملفات المشروع كما في رقم 1 ثم أفتح الملحقات References رقم 2 ,سترى مجموعة من الملفات مضافة للمشروع ومنها ملف System رقم 3 ,وفي بداية كود النافذة يجب استدعاء الملف باستخدام تعليمة using حتى يتم تحميل.

جميع كائنات ودوال الملف ويتمكن المعالج من معرفتها واستخدامها . أحياناً ملف dll يحتوي على كائنات داخلة وداخل كل كائن دوال مختلفة. في هذه الحالة لن نستطيع استدعاء الدالة كما فعلنا في دالة إظهار الرسالة ,لأن المعالج لا يعرف مكان الدالة في الملف فهو يفحص الدوال الموجودة في الملف ولكنه لا يفحص الكائنات , لذلك لا يجد الدوال المخزنة داخل كائنات ويجب علينا أن نحدد عنوان الدالة داخل الملف , ويتم ذلك بكتابة أسم الكائن متبوعاً بنقطة ثم أسم الكائن الداخلي ... وهكذا حتى نصل إلى مكان الدالة .

مثلاً دالة إنهاء البرنامج هي Exit , وهذه الدالة موجودة داخل كائن أسمة Environment ,وهذا الكائن موجود داخل ملف System , فإذا استدعينا هذه الدالة باسمها فقط فإن المعالج لن يعرف عنوانها ,ويظهر لنا خطأ , لذلك وجب علينا تحديد عنوانها هكذا :

كود

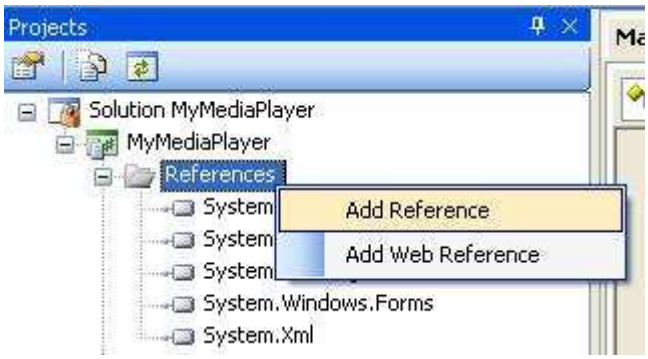
```
System.Environment.Exit(0);
```

..

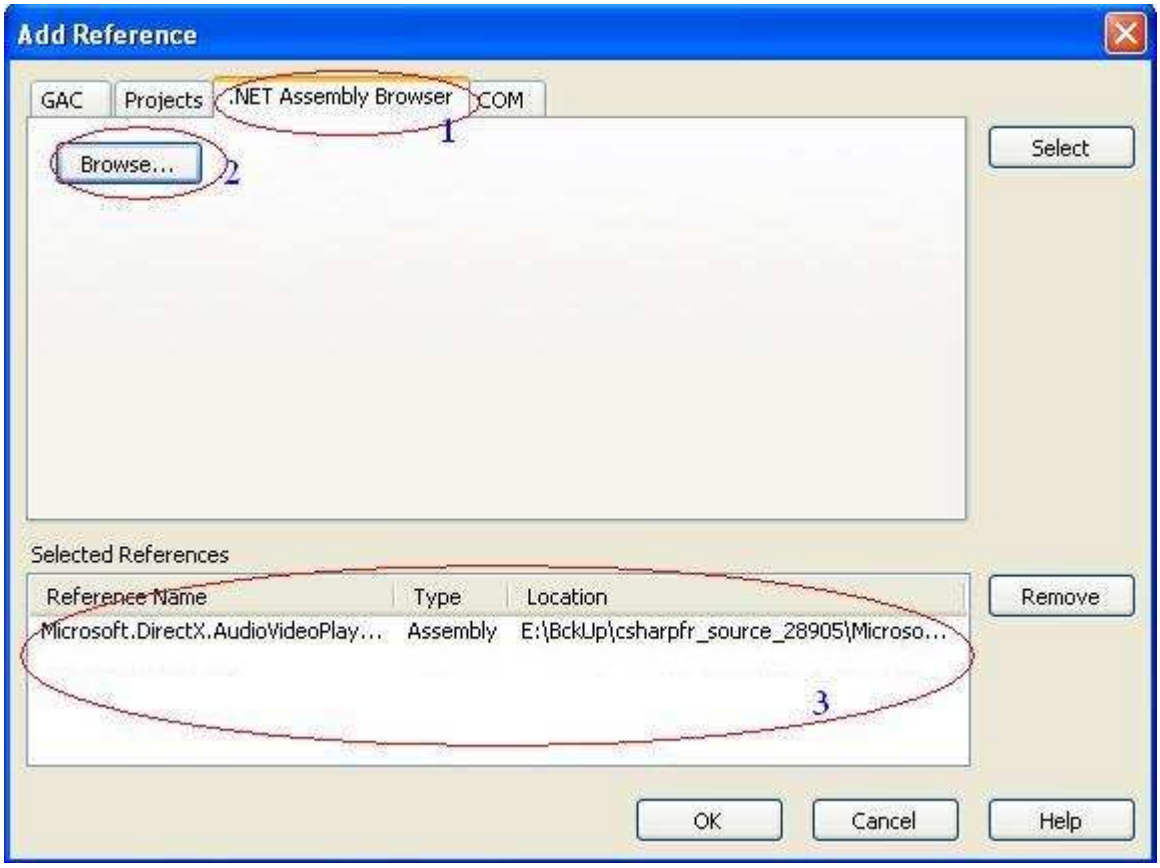
وبما أن تطبيقنا يشغل ملفات صوت وفيديو فيجب علينا إضافة الملفات التي يوجد فيها دوال الصوت والفيديو ودوال الصوت والفيديو موجود في ملفي dll ,وللأسف هذه الملفات لا ينزل مع لغة البرمجة لذلك يجب تنزيلها

تحميل من هنا

بعد تنزيل الملفات وفك الضغط , أفتح المشروع وافتح نافذة ملفات المشرع ,واضغط بالزر الأيمن على References وأختار Add Reference



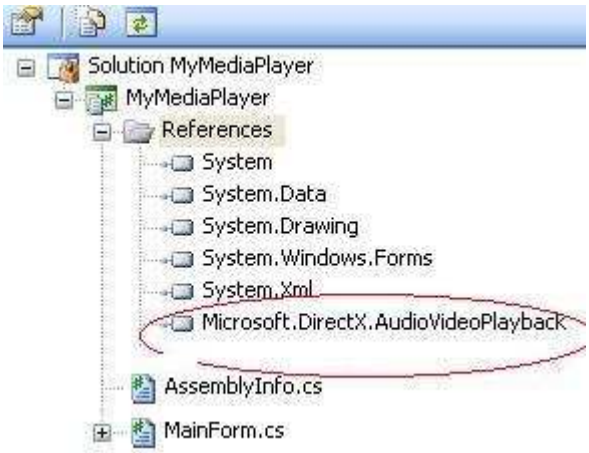
ستظهر لك هذه النافذة:



أختر **Browse** من تبويب **Net Assembly** ثم أضف ملف

Microsoft.DirectX.AudioVideoPlayback.dll

الذي قمت بتنزيله .حتى يظهر الملف في قائمة الملفات المختارة رقم **3** , ثم أختر **OK** ,سترى أن الملف تم إضافتها إلى قائمة الملفات الملحقة :

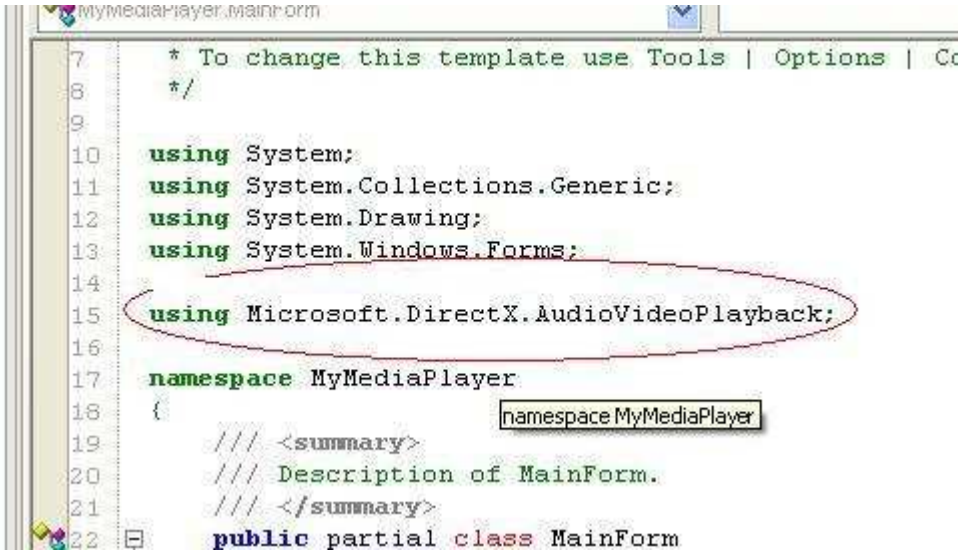


الآن في أعلى كود النافذة يجب استدعاء الملف ليتعرف عليه المعالج وذلك بإضافة الكود:

كود

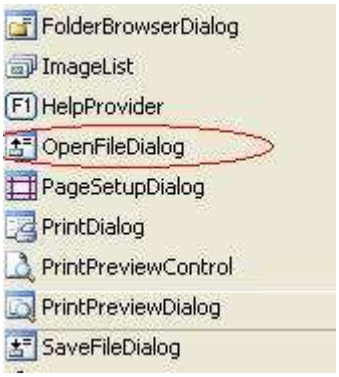
```
using Microsoft.DirectX.AudioVideoPlayback;
```

يضاف هذا الكود أسفل تعليمات الإضافة الافتراضية كما في الصورة :



الآن أذهب إلى تصميم نافذة تطبيق الدرس السابق و قم بحذف القائمة l1 ,وعدل النص في الزر من "تشغيل الدوارة" إلى "فتح ملف الصوت".

أسحب زر آخر وأكتب عليه "تشغيل ملف الصوت" ,ثم أسحب أداة OpenFileDialog إلى النافذة:



ستلاحظ أنها لن تضاف في النافذة وإنما في شريط أصفر تحت وذلك لأن هذه الأداة لا تظهر على النافذة مثل الزر أو صندوق النص.

الآن أختر الأداة من الشريط الأصفر و غير أسمها من OpenFileDialog1 إلى OF ليسهل التعامل معها من الكود

حتى يصبح شكل النافذة هكذا :

الآن سنقوم بتعريف متغير من نوع مشغل صوت Audio, وهذا النوع موجود في ملف

Microsoft.DirectX.AudioVideoPlayback

كنا نعرف متغير من نوع رقم هكذا :

كود

```
int x;
```

وبنفس الطريقة سنعرف متغير"كائن" من نوع مشغل صوت

كود

```
Audio ap;
```

عرفنا كائن أسمة ap من نوع Audio ..

لاحظ أن النوع **Audio** لا يعتبر متغير وإنما كائن لأنه يحتوي على أفعال وصفات ودوال كما سنرى ,ولكن لاحظ أننا سنستخدم هذا الكائن في إجراءين أثنين , إجراء فتح الملف و إجراء تشغيل الملف ,إذا في أي إجراء سنعرفه ؟؟ ... يجب أن يكون هذا المتغير متغير عام **Global**. لذلك يجب أن نعرفه خارج الإجراءين ونحدد هل هو عام لجميع النوافذ أم خاص لهذه النافذة .

إبحث عن مكان في الشفرة خارج الإجراءين , والمفضل أن يكون كما حددته في الدرس وقم بتعريف الكائن بهذا الكود :

كود

```
private Audio ap;
```

بما إننا سنستخدمه في هذه النافذة فقط جعلنا نوعه خاص **private** :

```
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

using Microsoft.DirectX.AudioVideoPlayback;

namespace MyMediaPlayer
{
    /// <summary>
    /// Description of MainForm.
    /// </summary>
    public partial class MainForm
    {
        private Audio ap;
        [STAThread]
        public static void Main(string[] args)
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
```

الآن افتح التصميم و اضغط مزدوجاً على زر "فتح ملف الصوت" لتذهب إلى الكود أمسح الكود من الدرس السابق :

وأضف هذا الكود :

كود

```
OF.ShowDialog();
    ap=Audio.FromFile(OF.FileName);
```

أضغط مزدوجا على زر "تشغيل ملف الصوت" وأضف هذا الكود:

كود

```
ap.Play();
```

في الكود الأول استخدمنا دالة ShowDialog في أداة فتح الملفات OF لتظهر لنا نافذة لفتح ملف الصوت :

هذة الدالة ترجع قيمة من نوع نص تحتوي على مسار الملف الذي اخترته وفي السطر الثاني استخدمنا دالة FromFile في كائن Audio لفتح الملف وتحميله للمتغير ap.

لاحظ أن الدالة FromFile تتطلب مدخل من نوع نص يحتوي على مسار الملف المراد تحميله ,ومسار هذا الملف هو القيمة التي سترجعها الدالة OF.FileName, في الكود الثاني استخدمنا دالة Play في المتغير ap لتشغيل ملف الصوت.

الآن قم بتشغيل البرنامج وأضغط على زر "فتح ملف الصوت" ,وإختر ملف صوت من نوع mp3 أو wav , لا تختر ملف من نوع rm , سنعرف فيما بعد كيف نشغل ملفات rm و ram ,بعد اختيار الملف اضغط على زر "تشغيل ملف الصوت" واستمع للصوت الناتج من برنامجك.

ملاحظة : بما إننا التطبيق أستخدم دوال داخل ملف Microsoft.DirectX.AudioVideoPlayback

فانه عند تشغيل البرنامج سيقوم المعالج بنسخ هذا الملف إلى مسار البرنامج ,

ولن يعمل البرنامج إلا إذا كان هذا الملف إلى جانبه في نفس المجلد

واجب الدرس الرابع

ماذا لو اخترت ملفاً غير صوتياً وجربت تشغيله ؟

بالطبع سيظهر لك خطأ ويتوقف البرنامج , ما رأيك لو تعقبت هذا الخطأ ,وأظهرت رسالة بالعربية تخبر المستخدم ما الذي يجري .

تلميح : ظهر الخطأ سيكون عند تعليمة

كود:

كود

```
ap.Play();
```

لذلك يجب وضع هذه التعليمة ضمن جملة try و ... , يمكن أيضاً إن يظهر خطأ عند تعليمة :

كود

```
ap=Audio.FromFile(OF.FileName);
```

وبالتالي فالأفضل وضعها ضمن جملة try أيضاً

أضف زر لإنهاء البرنامج وأرسل التطبيق ..

ملاحظة مهمة:

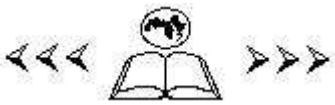
ليعمل تطبيق الصوت في أنظمة لا تحتوي على DirectX SDK

يجب إرفاق ملف DirectX الذي قمت بتنزيله في مجلد التطبيق بجانب ملف البرنامج

بمعنى أن نضع هذا الملف بجانب ملف البرنامج exe كما في الصورة

نهاية الدرس الرابع.

..



الدرس الخامس

تشغيل ملف فيديو

دوال تشغيل الفيديو موجودة في ملف Microsoft.DirectX.AudioVideoPlayback.dll

الذي أضفناه إلى التطبيق في الدرس السابق .

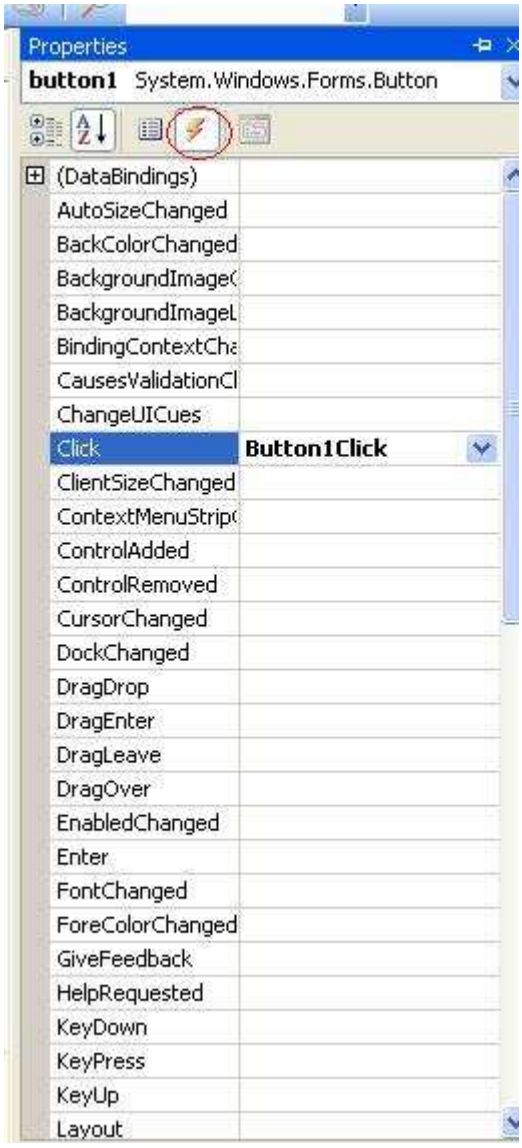
دوال الفيديو تشبه إلى حد كبير دوال الصوت عدى عن اختلافات بسيطة, من أهم هذه الاختلافات أنه يمكنك أن تحدد كائن حاضن Owner Control لشاشة الفيديو, معنى كائن حاضن إي كائن يعمل كشاشة فيديو .

أما إذا لم تحدد كائن حاضن فان التطبيق سيفتح نافذة جديدة لتكون شاشة الفيديو .

قراءة أحداث المفاتيح

كل كائن له صفات و دوال كما عرفنا سابقاً , والكائنات تحتوي أيضاً على أحداث Events ,مثلاً الزر له عدة أحداث منها عند الضغط عليه Click وعند الضغط المزدوج DoubleClick ,و عند مرور مؤشر الماوس به MouseOver وفي الدروس السابقة استخدمنا حدث Click فقط .

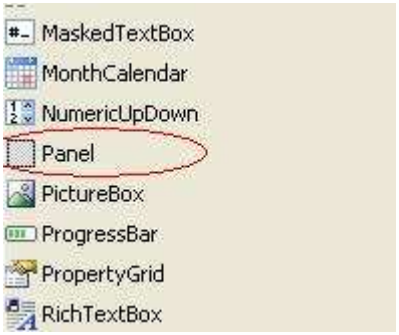
للتصفح بين أحداث الكائن , أختر الكائن ثم أفتح نافذة الصفات "الخواص" واختر زر الأحداث:



ستظهر لك قائمة بجميع الأحداث للكائن المحدد ,ولكي تضيف تعليمات عند حدوث حدث معين اضغط مزدوجاً على أسم الحدث في القائمة ,ستنتقل إلى الشفرة لتكتب التعليمات التي تنفذ عن حدوث ذلك الحدث , ومن الأحداث المهة حدث KeyPress وينطلق هذا الحدث عند الضغط على أي مفتاح ,في لوحة المفاتيح .

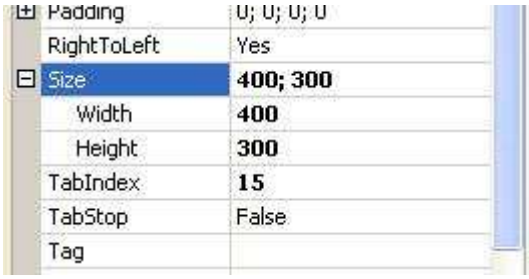
تطبيق الدرس الخامس:

أفتح تطبيق الدرس السابق وعدل النص في زر فتح ملف الصوت إلى "فتح ملف ميديا" .وفي زر تشغيل ملف الصوت إلى "تشغيل ملف ميديا" .أضف زراً جديداً وغير النص فيه إلى "ملئ الشاشة" ,قبل تشغيل ملف الفيديو يجب أن نحدد له كائن حاضن ليعمل كشاشة .افتح التصميم وأضف أداة البانيل إلى الفورم:



تأكد من اختيار البانيل و غير صفة الاسم إلى p1

غير صفة العرض Width إلى 400 وصفة الارتفاع Height إلى 300



من خلال صفة BackgroundImage إختر اي صورة لتكون خلفية الشاشة

ليصبح شكل النافذة هكذا:

أضغط مزدوجاً على أي زر للذهاب إلى الشفرة , نحتاج إلى تعريف كائن عام Global من نوع فيديو ولأننا سنستخدمه في أكثر من إجراء .

سنعرفه خارج جميع الإجراءات والدوال , أذهب إلى الشفرة حيث عرفنا متغير الصوت في الدرس السابق .وأكتب تعريف كائن الفيديو :

كود

```
private Video vp;
```

تحت تعريف كائن الصوت من الدرس السابق :

```
/// Description of MainForm.
/// </summary>
public partial class MainForm
{
    private Audio ap;
    private Video vp;
    [STAThread]
    public static void Main(string[] args)
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault
```

سنجعل التطبيق يفتح ملفات صوت أو ملفات فيديو , لكن المشكلة أن للصوت كائن وللفيديو كائن آخر , وبالتالي يجب أن نعرف نوع الملف حتى نستخدم الكائن المحدد .وبما أن أنواع الفيديو كثيرة جداً فلا يمكننا فحص نوع الملف لتأكد من نوعه وبالتالي فالحل الوحيد هو أن نحاول أولا تشغيل الملف بكائن الفيديو ,فإن ظهر لنا خطأ فنحاول نشغله بكائن الصوت , وأن ظهر لنا خطأ للمرة الثانية فالملف غير صالح .قد يسأل البعض لماذا بدأنا بتجربة كائن الفيديو مع أن معظم الملفات ستكون صوت ؟ .

والإجابة أن كائن الصوت ممكن أن يشغل ملفات فيديو بدون صورة , يعني انه يفتح ملف الفيديو ,ويشغل الصوت الذي فيه وبالتالي فربما يكون الملف فيديو ويتم تشغيله كملف صوت. لذلك يجب أولاً تجربة كائن الفيديو حتى إذا كان ملف فيديو يشغل الصوت مع الصورة . وإذا كان ملف صوت نجرب كائن الصوت ..

لعمل ذلك نحتاج إلى تنفيذ جملة تعقب الأخطاء try مرتين متداخلتين:

كود

```
try
{
//شغل ملف الفيديو
}
catch
{
   try
     {
     //شغل ملف الصوت
     }
     catch
     {
      //الملف غير صالح
       }
}
```

أولا سنجرب فتح ملف فيديو فإذا حدث خطأ فإن المترجم ينتقل لتنفيذ التعليمات داخل جملة **catch** والتي هي باللون الأزرق , وهناك جملة **try** أخرى نقوم من خلالها بتجربة فتح ملف صوت .فإذا حدث أي خطأ فان المترجم ينتقل لتنفيذ تعليمات **catch** التي تحتها خط .في زر فتح ملف الميديا أمسح الكود السابق وأكتب بدلاً عنه :

كود

```
OF.ShowDialog();
try
{
   vp = Video.FromFile(OF.FileName);
   vp.Owner = p1;
   p1.Width = 400;
   p1.Height = 300;
}
catch
{
  try
   {
   ap = Audio.FromFile(OF.FileName);
   }
   catch
   {
   MessageBox.Show("الرجاء إختيار ملف صالح");
   }

}
```

التعليمة **OF.ShowDialog** تفتح نافذة فتح ملف ميديا ,عند اختيار الملف نجرب تشغيل الفيديو عبر هذه التعليمات:

كود

```
vp = Video.FromFile(OF.FileName);
  vp.Owner = p1;
  p1.Width = 400;
  p1.Height = 300;
```

في السطر الأول نحمل الملف الموجود في كائن فتح الملف إلى كائن الفيديو ,في السطر الثاني نحدد الكائن الحاضن Owner لكائن الفيديو ,في السطر الثالث والرابع , نعيد تحديد حجم الكائن الحاضن إلى الحجم الأصلي .لأن ملفات الفيديو لها أحجام مختلفة وحجم الكائن الحاضن قد يختلف على حسب حجم الصورة ,في ملف الفيديو لذلك من الأفضل إعادة تحجيم الكائن الحاضن ,عند حدوث أي خطأ سينتقل المترجم لتنفيذ الكود باللون الأزرق وتعليمة تحميل ملف الصوت إلى كائن الصوت:

كود

```
ap = Audio.FromFile(OF.FileName);
```

عند حدوث خطأ آخر هنا سينتقل المترجم إلى تنفيذ تعليمة **catch** الداخلية وهي إظهار رسالة الخطأ .,سنستخدم نفس الطريقة في إجراء الضغط على زر تشغيل ملف الميديا .فتح التصميم وأضغط علية مزدوجاً للانتقال للشفرة أمسح الشفرة الموجودة وأكتب بدلاً منها:

كود

```
try
{
vp.Play();
}
catch
{
  try
   {
   ap.Play();
   }
   catch
   {
  MessageBox.Show("محاولة تشغيل الملف حدث خطأ اثناء ");
   }
}
```

نجرب تشغيل ملف الفيديو وإذا حدث خطا نجرب تشغيل ملف الصوت وإذا حدث خطأ آخر نظهر رسالة.

افتح التصميم واضغط مزدوجاً على زر ملئ الشاشة وأكتب تعليمة ملئ الشاشة لكائن الفيديو:

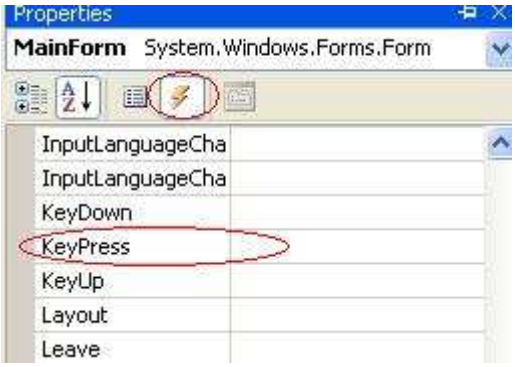
كود

```
vp.Fullscreen = true;
```

كائن الفيديو يحتوي على صفة **Fullscreen** من نوع قيمة بوليا نية إما صح أو خطأ

فإذا حددنا الصفة بقية **true** فأن تشغيل الفيديو سيكون ملى الشاشة ,و إذا حددناها **false** سيرجع تشغيل الفيديو إلى الكائن الحاضن ,عرفنا كيف نجعل التشغيل ملئ الشاشة لكن المشكل كيف نرجعه للكائن الحاضن لأنه في وضع ملى الشاشة لن نستطيع أن نرى أي شيء في شاشة الكمبيوتر سوى الفيديو ,يأتي هنا دور حدث الضغط على أي من مفاتيح لوحة المفاتيح حيث سنجعل الفيديو ,يرجع إلى الكائن الحاضن بمجرد الضغط على أي من مفاتيح لوحة المفاتيح ,أفتح التصميم وأختر النافذة الأساسية **"لا تختر أي زر أو كائن تأكد من اختيار النافذة كلها"**

أذهب إلى قائمة الإحداث وأضغط مزدوجا على الحدث **KeyPress**



ستنتقل إلى شفرة الحدث , أكتب تعليمة إعادة صورة الفيديو إلى الكائن الحاضن :

كود

```
vp.Fullscreen = false;
```

هناك مشكلة ستواجهنا هنا وهي في حالة إذا كان تركيز لوحة المفاتيح على زر محدد ,فلن ينطلق حدث الضغط على النافذة سينطلق الحدث على الزر المحدد فقط ,والشفرة السابقة على النافذة الأساسية , الحل أن هناك صفة في النافذة تمكنها من إطلاق الحدث حتى ولو حصل في أي كائن من الكائنات هذه الخاصية هي **KeyPreview** عدلها من **False** إلى **True** لتصبح كما في الصورة :

الآن شغل البرنامج وجرب تشغيل ملف صوت أو فيديو أو ملف نصي

تلميحات الأدوات

أحيانا تحتاج إلى إظهار ملاحظة بسيطة للمستخدم عن وظيفة زر في التطبيق مثلا ً :

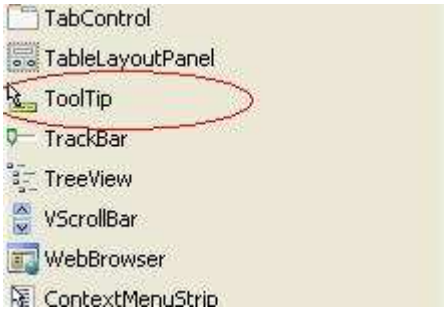
وتظهر هذا الملاحظات في مستطيل أصفر يظهر بمجرد مرور الماوس من فوق الزر

مثلاً لو تمرر الماوس فوق زر Start في شريط المهام ستظهر هذه الملاحظة

Click here to begin

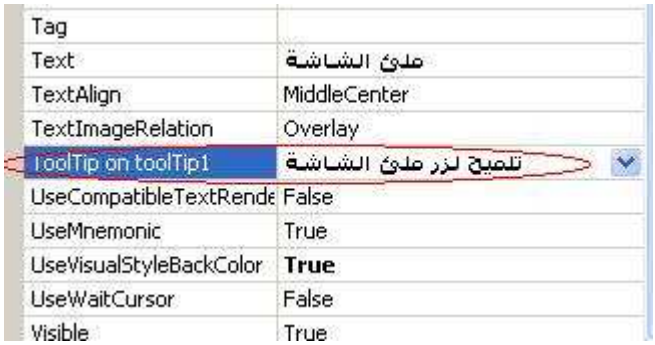
لعمل مثل هذه الملاحظات في تطبيقنا نحتاج لإضافة كائن جديد

افتح التصميم واسحب كائن تلميحات الأدوات ToolTip



الآن في أي كائن تختاره ستجد صفة جديدة أسمها ToolTip on toolTip1

أكتب فيها أي تلميح نصي للكائن :



عند تشغيل البرنامج وتمرير الماوس من فوق الزر ستظهر الملاحظة هكذا:

واجب الدرس الخامس :

أرسل التطبيق.

نهاية الدرس الخامس.

..

الدرس السادس

تشغيل ملفات الريل بلاير Playing RM Files

سنستخدم في هذا الدرس الإجراءات كثيراً , ولمزيد من التوضيح عنها أرجع إلى الدرس الرابع ,تقنية rm تم تصميمها بواسطة شركة Real Networks Inc. حيث أن لهذه التقنية ,مزايا كثيرة من أهمها حجم الملفات الصغير الذي يتيح تداول الملفات عبر الوب بسهولة .في الدروس السابقة تعاملنا مع ملفات الميديا بأي تقنية ما عدا هذه التقنية .ولأن هذه التقنية تختلف تماماً عن التقنيات الأخرى خصصنا لها هذا الدرس.

لتشغيل ملفات rm نحتاج لملفات dll تحتوي على دوال تشغيل rm

الملفات موجودة للتحميل من هنا

وهي من شركة Real تم تحويله قليلاً ليتعامل مع بيئة الدوت نت ,في التطبيق سنجد اختلاف بين أوامر rm والأوامر الأخرى لأن شركة Real تحتكر هذه التقنية .

ولا تقبل بتوزيعها على الشركات الأخرى لذلك فميكروسوفت لا تدعم تشغيل ملفات rm .فمثلاً كما رأينا في الدرس السابق فان أمر تشغيل ملف صوت أو فيديو عادي هو:

كود

```
Play();
```

أما لتشغيل ملف rm فيكون الأمر هكذا

كود

```
DoPlay();
```

هذه مقارنه بسيطة بين أوامر ميكروسوفت وأوامر **Real**

كود

```
AVI----------------Real
1-FromFile--------------------Set Source
2-Play------------------------Do Play
3-Stop------------------------Do Stop
4-Pause-----------------------Do Pause
5-Duration--------------------Get Length
6-CurrentPosition-----------Get Position /Set Position
```

الأمر الأول يستخدم لتحميل الملف

الأمر رقم 2 يستخدم لتشغيل الملف

الأمر 3 لإيقاف تشغيل الملف نهائياً

الأمر 4 لإيقاف تشغيل الملف مؤقتاً

الأمر 5 لإيجاد طول الملف بالثواني

الأمر 6 لإيجاد مكان التشغيل , أي إيجاد الموقع في الملف الذي يتم تشغيله

مثلاً إذا أنتجت الدالة 60 معنى ذلك أنه يتم الآن تشغيل الثانية رقم 60 من الملف ,في أوامر Real هناك أمرين أحدهما لقرائه الموقع والآخر لضبط الموقع ,أما في أوامر ميكروسوفت فالقراءة والضبط تستخدم نفس الأمر .سنستخدم في هذا الدرس جميع الأوامر السابقة .

تطبيق الدرس السادس:

أفتح تطبيق الدرس السابق ,أضف الملف AxInterop.RealAudioObjects.dll الذي قمت بتحميله إلى المراجع References كما فعلنا في الدرس الرابع .سنضيف في هذا التطبيق كائن عام جديد من نوع مشغل rm وليكون أسم هذا الكائن rp اختصار لـ Real Player.

سنقوم بتعريف هذا الكائن تحت تعريفات كائنات الصوت والصورة في الدرس السابق هكذا:

كود

```
private AxRealAudioObjects.AxRealAudio rp=new AxRealAudioObjects.AxRealAudio();
```

لاحظ أن كود تعريف الكائن يختلف قليلاً عن تعريف كائنات ميكروسوفت ,بعد تعريف كائن مشغل rm أصبح لدينا ثلاثة كائنات رئيسية .

مشغل صوت و مشغل فيديو ومشغل rm وعند فتح ملف جديد.

سنبدأ باختبار هل هو ملف فيديو ثم هل هو ملف صوت ثم هل هو ملف rm .وبالتالي سنستخدم جملة try ثلاث مرات متداخلة عند فتح الملف وعند تشغيل الملف .وعند إيقاف تشغيل الملف أو عند تقديم أو تأخير الملف ..

وإذا أردنا اختصار الكود لنستخدم جملة try المتداخلة فقط عند فتح الملف. سنستخدمها فقط عند فتح "تحميل الملف" ومنها سنعرف نوع الملف , ونقوم بخزن نوع الملف .في متغير عام من نوع نص , وعند تشغيل الملف بدلاً من استخدام try المتداخلة كما فعلنا في الدرس السابق .سنفحص نوع الملف من خلال المتغير من نوع نص الذي خزنا فيه نوع الملف عند الفتح .دون استخدام جملة try وذلك سيسهل كثيرا ويسرع عملية تشغيل الملف والتعامل معه .

إذاً سنعرف متغير عام من نوع نص لخزن نوع الملف وليكون أسمه type .نعرف هذا المتغير تحت تعريفات كائن الصوت والفيديو و مشغل rm

هكذا:

كود

```
private string type;
```

بحيث تصبح المتغيرات العامة هكذا :

```
/// </summary>
public partial class MainForm
{
    private Audio ap;
    private Video vp;
    private AxRealAudioObjects.AxRealAudio rp=new AxRealAudioObjects.AxRealAudio();
    private string type;

    [STAThread]
    public static void Main(string[] args)
```

اذهب إلى التصميم ,أضف للنافذة الأساسية زرين زر للإيقاف و زر للإيقاف المؤقت .أذهب إلى كود البرنامج ..

سنقوم بكتابة إجراء مستقل لتحميل ملف الميديا إلى أحد الكائنات الثلاثة....

إجراء تحميل الملف

وهذه الإجراء سيكون مستقلة لكي نستطيع استخدامه في أكثر من مكان ,لنسمي هذه الإجراء LoadFile وستكون وظيفته اختبار نوع الملف ثم ضبط قيمة المتغير type ,ليحمل حرف يرمز إلى نوع الملف و أيضا هذا لإجراء ستقوم بتحميل الملف للكائن المناسب من الكائنات الثلاثة.

الأحرف التي سيحملها المتغير type هي A إذا كان ملف صوت و V إذا كان ملف فيديو وR إذا كان ملف rm

أي أن الإجراء سيفحص نوع الملف فإذا كان ملف صوت سيحمل type الحرف A ويحمل الملف إلى كائن الصوت ap

أما إذا كان فيديو فيحمل type الحرف V ويحمل الملف لكائن الفيديو vp

أما إذا كان الملف rm فيحمل type الحرف R ويحمل الملف لكائن الريل rp

فحص نوع الملف سيكون بنفس الطريقة في الدرس السابق ولكنا سنستخدم try ثلاث مرات متداخلة

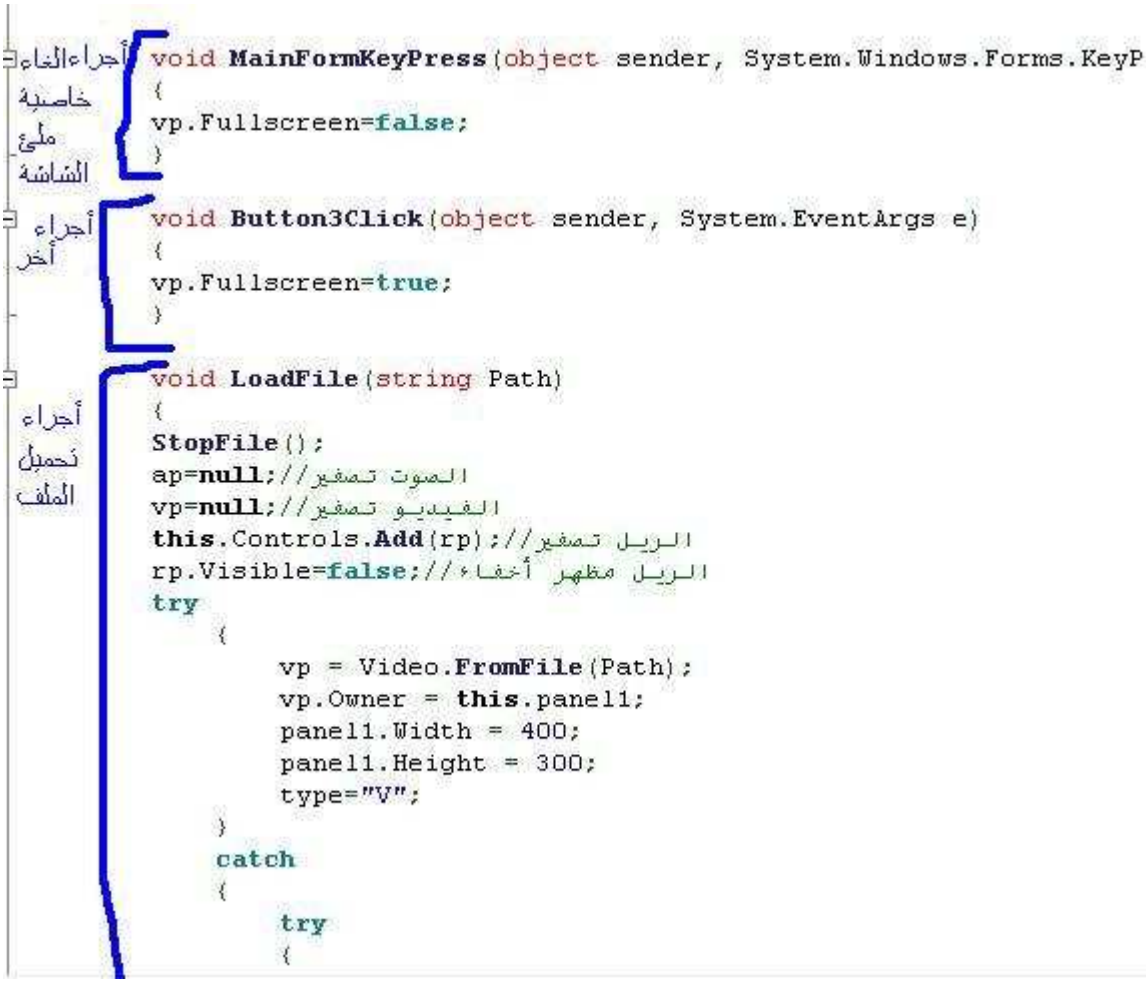
أنسخ هذا الكود إلى الشفرة وتأكد أن يكون هذا الكود خارج أي دالة أخرى:

كود

```
void LoadFile(string Path)
  {
        StopFile();
        ap=null;//تصفير الصوت
        vp=null;//الفيديو تصفير
        this.Controls.Add(rp);//تصفير الريل
              rp.Visible=false;//أخفاء مظهر الريل
        try      {
              vp = Video.FromFile(Path);
              vp.Owner = this.panel1;
              panel1.Width = 400;
              panel1.Height = 300;
              type="V";
              }
              catch  {
                  try
                  {
                  ap = Audio.FromFile(Path);
                  type="A";
                  }
                  catch  {
                    try  {
                    rp.SetSource(Path);
                    type="R";
                    }
                    catch
                    {
                       MessageBox.Show("ملف صالح الرجاء إختيار");
                       type="N";
                    }
                  }

              }
  }
```

الكود السابق يمثل إجراء فحص الملف وتحمليه إلى أحد الكائنات الإجراء هو جزء من الشفرة يمكن استخدامه أكثر من مرة من خلال استدعائه باسمه فقط. وبما أنه إجراء مستقلة يجب إن تكون خارج أي إجراء أخر:



الإجراء يتكون من مدخل واحد **Path** من نوع نص ويمثل مسار الملف الذي سيتم تحميله

أي أننا عندما نستدعي الإجراء سنكتب أسمه ونعطيه مدخل من نوع نص مثلاً:

كود

```
LoadFile("C://s.mp3");
```

الكود السابق سيقوم باستدعاء الإجراء و الإجراء سيقوم بفحص وتحميل الملف **s.mp3** الموجود في القرص **C**

ملاحظة: في لغة السي شارب يعتبر حرف \ حرفاً غير مسموح به

لذلك عند كتابة مسار الملف يتم إست بدالة بـ //

في السطر الأول من الإجراء تعلميه StopFile وهي استدعاء لإجراء أخر سنشرحه فيما بعد .يقوم إجراء StopFile بإيقاف تشغيل أي ملف لكي يتم تحميل ملف جديد بدلاً منه وإذا لم نوقف تشغيل ,الملف السابق وقمنا بتحميل ملف آخر سيحدث تداخل في الملفات ويتم تشغيل أكثر من ملف في آن واحد .في السطور 2 و3 و4 و5 كما هو مشروح في الكود يتم تصفير الكائنات ,ومعنى التصفير هو إلغاء تحميل الملفات السابقة لكي يتم تحميل ملفات جديدة .بعدها جملة try الأولى تقوم بتحميل كائن الفيديو الملف المدخل من خلال المتغير Path ,فإذا تم التحميل بنجاح فإن التعليمة "type="V تقوم بتحميل الحرف V إلى المتغير type ,دلالة على أن نوع الملف هو فيديو , أما إذا حدث إي خطأ في التحميل فسوف يقفز المعالج إلى تنفيذ الكود باللون الأخضر ز

ويتم فيها تحميل الملف لكائن الصوت فإذا تم التحميل بنجاح يتم تحميل المتغير type الحرف A دلالة على أن نوع الملف هو صوت , أما إذا حدث خطأ في التحميل فالمعالج سيقفز إلى تنفيذ الكود باللون الأحر .ويتم فيه تحميل الملف إلى كائن rm "لاحظ الاختلاف في الأمر عن أوامر الصوت والفيديو"

فإذا تم التحميل بنجاح يتم تحميل المتغير type الحرف R دلالة على أن نوع الملف هو ريل .

وإذا حدث خطأ في التحميل يقفز المعالج لتنفيذ الكود باللون البرتقالي حيث يظهر رسالة خطأ

ويحمل المتغير الحرف N دلالة على أن نوع الملف غير معروف .

إجراء تشغيل الملف

سنقوم بكتابة إجراء مستقل أخر لتشغيل الملف , إجراء التشغيل سيكون أسهل بكثير من إجراء تحميل الملف لان لدينا نوع الملف مخزن في المتغير type ولن نقوم بفحص نوع الملف مرة أخرى .حيث سنقوم باستخدام تعليمة if لنعرف الحرف الموجود في type ومن خلاله نعرف نوع الملف فنشغل الكائن المطلوب .

لنسمي هذا الإجراء PlayFile ولن تكون لهذا الإجراء أي مدخلات .

أنسخ هذا الكود إلى الشفرة وتأكد أن يكون خارج أي إجراء أخر :

كود

```
void PlayFile()
{
   if(type=="V")
   {
   vp.Play();
   }
   else if(type=="A")
   {
    ap.Play();
   }
   else if(type=="R")
   {
   rp.DoPlay();
   }
}
```

في المقارنة الأولى إذا كان المتغير يحمل الحرف V فهذا يعني أن نوع الملف هو فيديو كما في إجراء التحميل ولذلك سيتم تشغيل كائن الفيديو vp , إما إذا لم يكن V فينتقل المعالج للمقارنة الثانية وإلا للثالثة.

إجراء الإيقاف المؤقت

في هذا الإجراء أيضاً سنستفيد من محتويات المتغير type لتحديد نوع الملف والتعامل مع الكائن المحدد

أنسخ هذا الكود إلى الشفرة وتأكد أن يكون خارج أي إجراء أخر :

كود

```
void PauseFile()
{
   if (type=="V")
   {
   vp.Pause();
   }
   else if (type=="A")
   {
    ap.Pause();
   }
   else if (type=="R")
   {
   rp.DoPause();
   }
}
```

لا أظن أنها تحتاج إلى شرح..

إجراء إيقاف التشغيل

أنسخ هذا الكود إلى الشفرة وتأكد أن يكون خارج أي إجراء أخر:

كود

```
void StopFile()
  {
  if (type=="V")
     {
     vp.Stop();
     }
     else if (type=="A")
     {
      ap.Stop();
     }
     else if (type=="R")
     {
     rp.DoStop();
     }
  }
```

هذا هو الإجراء الذي استدعيناه في إجراء تحميل الملف لو قمت بتشغيل البرنامج الآن لن يتم تشغيل أي إجراء لأننا لم نستدعي أي منها .

أفتح التصميم وأضغط مزدوجاً على زر فتح ملف ميديا , أمسح الكود السابق وأكتب بدلاً عنه:

كود

```
OF.ShowDialog();
   LoadFile(OF.FileName);
   PlayFile();
```

في السطر الأول استدعينا نافذة فتح الملف

في السطر الثاني استدعينا إجراء تحميل الملف و حملنا المدخل بناتج كائن فتح الملف ,أي أنه في شفرة إجراء التحميل ستكون قيمة المدخل **Path** هي قيمة **OF.FileName** .

بعد استدعاء إجراء تحميل الملف استدعينا إجراء تشغيل الملف , أي أنه سيتم تشغيل الملف بمجرد اختياره

أذهب إلى التصميم مجددا واضغط مزدوجاً على زر التشغيل, أمسح الكود السابق وأكتب بدلاً عنه:

كود

```
PlayFile();
```

استدعاء لإجراء التشغيل لا غير

نلاحظ كيف استدعينا إجراء التشغيل في زر فتح الملف وفي زر التشغيل

اذهب للتصميم واضغط مزدوجاً على زر الإيقاف المؤقت وأكتب فيه هذا الكود:

كود

```
PauseFile();
```

اذهب الي التصميم واضغط مزدوجاً على زر الإيقاف وأكتب هذا الكود:

كود

```
StopFile();
```

نستطيع الآن أن نقول أن التطبيق يشغل جميع أنواع الملفات

ملاحظة:

عند استيراد ملف دوال الريل بلاير **AxInterop.RealAudioObjects.dll**

تقوم بيئة التطوير بإنشاء الملف الآخر كربط بين التطبيق والملف

حيث يتكون الملف **Interop.RealAudioObjects.dll**

بجانب التطبيق بمجرد تشغيل التطبيق من بيئة التطوير .وهذا الملف ضروري لأنه يعمل كوسيط بين التطبيق وملف دوال الريل .إذا لم تستطيع البيئة تكوين هذا الملف لسبب أو لآخر ,يجب ان يوضع بجانب التطبيق

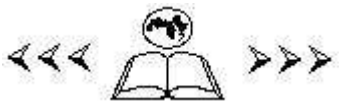
أي أن ملفات **dll** التي يحتاجها التطبيق أصبحت أربعة ملفات مع ملفات الدروس السابقة..

واجب الدرس السادس

أرسل التطبيق..

.نهاية الدرس السادس

..



الدرس السابع

التنقل في الملف

ملف الصوت أو الفيديو يحتوي على خاصية طول الملف أي أن كل ملف له طول زمني .ويتم قياس الطول في مشغلات ميكروسوفت بالثانية , وفي مشغلات rm بالملي ثانية .

يتم إيجاد طول الملف من خلال الخاصية Duration في مشغلات ميكروسوفت ,أو الخاصية GetLength في مشغلات rm

مثلا التعليمة التالية :

كود

```
ap.Duration;
```

ستنتج رقم يحدد طول الملف بالثواني مثلاً إذا أنتجت رقم 100 يعني هذا أن طول الملف 100 ثانية

أم التعليمة التالية:

كود

```
rp.GetLength();
```

فهي تنتج رقم يحدد طول الملف بالملي ثانية مثلاً إذا أنتجت 50000 يعني أن طول الملف 50 ثانية

إذاً هناك فرق كبير بين قيمة طول الملف في مشغلات ميكروسوفت ومشغلات rm .من أهم الوظائف لأي مشغل ميديا هو إمكانية التنقل عبر الملف سوءاً تقديماً أو تأخيرا ,ويتم التنقل عبر الملف من خلال تعديل خاصية من خصائص كائن التشغيل الخاصية ترجع قيمة تحتوي على رقم يمثل موقع التشغيل الحالي .مثلاً إذا أرجعت الخاصية الرقم 50 في مشغل ميكروسوفت فهذا يعني أن تشغيل الملف وصل إلى الثانية رقم 50 .أما إذا أرجعت 40000 في مشغلات rm فهذا يعني أن تشغيل الملف وصل إلى الثانية رقم 40 .الخاصية في مشغلات ميكروسوفت هي CurrentPosition سواء لقراءة القيمة أو لتعديلها .

فمثلاً التعليمة التالية:

كود

```
ap.CurrentPosition;
```

ترجع رقم يمثل رقم الثانية الذي وصل إليها التشغيل , فمثلاً لو كان طول الملف 100 ثانية

وأرجعت الرقم 50 فهذا يعني أن تشغيل الملف وصل إلى المنتصف.

أما التعليمة التالية:

كود

```
ap.CurrentPosition=60;
```

هذه التعليمة تقوم بنقل التشغيل إلى الثانية رقم 60 , أي أن التشغيل يقفز من موقعه إلى الثانية رقم 60

وفي مشغلات rm هي GetPosition لقراءة القيمة أو SetPosition لتعديل القيمة.

فمثلاً التعليمة :

كود

```
rp.GetPosition();
```

ترجع رقم يمثل رقم الملي ثانية الذي وصل إليها التشغيل فمثلاً إذا كان طول الملف 10000 ملي ثانية

وأرجعت الرقم 5000 فهذا يعني أن التشغيل وصل إلى الثانية رقم 5 اي إلى منتصف الملف.

أما التعليمة التالية:

كود

```
rp.SetPosition(60000);
```

هذه التعليمة تقوم بنقل التشغيل إلى الثانية رقم 60 , لاحظ أن جميع تعاملات مشغل rm تكون بالملي ثانية .وليست بالثانية

كما في مشغلات ميكروسوفت , والثانية الواحد = 1000 ملي ثانية .

إذاً التنقل عبر الملف يتم من خلال تعديل خاصية موقع التشغيل .

المؤقت Timer

المؤقت هو عبارة عن كائن لضبط الوقت , ويستخدم لتكرار حدث معين خلال فترة زمنية محددة.

مثلاً إظهار رسالة كل 50 ثانية , أو تنفيذ كود معين كل 10 ثوان.

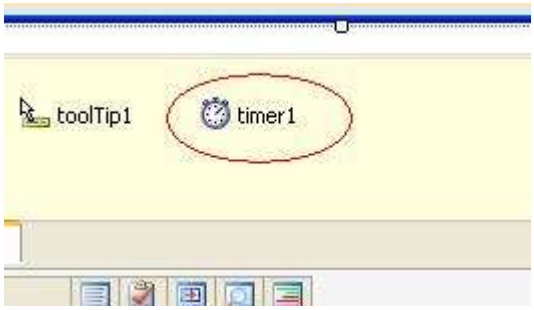
يتم تحديد طول الفترة الزمنية من خلال تعديل الخاصية "الصفة" Interval, حيث أن هذه الخاصية تأخذ الزمن بالملي ثانية أي أنه إذا أردنا أن تكون الفترة ثانية واحدة.

فإن الخاصية Interval يجب إن تكون قيمتها 1000 , وإذا أردنا ثانيتين تكون 2000 .ويتم كتابة الكود المراد تنفيذه من خلال الضغط مزدوجاً على المؤقت .

في هذا الدرس سنستخدم المؤقت ليفحص موقع تشغيل الملف كل ثانية.

التطبيق

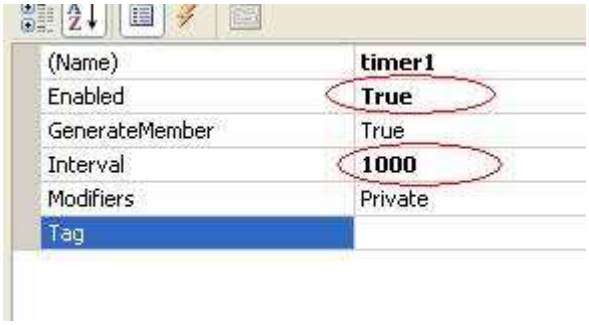
أفتح تطبيق الدرس السابق , وأضف زرين للتقديم والتأخير .أضف مؤقت Timer , ستلاحظ انه سيضاف إلى الشريط الأصفر تحت نافذة التصميم :



أختر المؤقت وقم بتعديل خاصية Interval إلى 1000 ,"ثانية واحدة" .

وعدل قيمة الخاصية Enabled إلى True , هذه الخاصية تقوم بتشغيل أو تعطيل عمل المؤقت.

فإذا كانت القيمة True فالمؤقت مفعل , أما إذا كانت False فهو معطل ولن يعمل



أضف كائن التنقل TrackBar ليستخدم كأداة تنقل عبر الملف

أختر كائن التنقل و عدل الخاصية TickFrequency إلى 0 لكي تختفي خطوط القياس

وعدل الخاصية TickStyle إلى Both إذا أردت

TabIndex	23
TabStop	True
Tag	
TickFrequency	0
TickStyle	Both
ToolTip on toolTip1	
UseWaitCursor	False
Value	0

كائن التنقل يرجع قيمة بين الخاصيتين Minimum و Maximum

فمثلاً إذا كان Minimum = 0 و Maximum=50 فإن الخاصية Value سترجع قيمة بين 0 و 50

تمثل مكان المؤشر بالنسبة للكائن فمثلاً إذا كان المؤشر في منتصف الكائن فإن Value سترجع الرقم 25

وبما إننا نحتاج للتنقل عبر الملف فالمفروض أن نعدل الخاصية Minimum إلى 0

ونعدل الخاصية Maximum إلى طول الملف حتى ترجع القيمة Value رقم بين 0 وطول الملف .وبما أن طول كل ملف يختلف عن الملف الآخر , فإننا سنعدل الخاصية Maximum عند تحميل أي ملف .ولن نحتاج لتعديل الخاصية Minimum لأن قيمتها الافتراضية هي 0

سيصبح شكل النافذة هكذا:

إجراء ضبط كائن التنقل

أنقل هذا الكود إلى شفرة البرنامج مع مراعاة أن يكون خارج أي إجراء آخر:

كود

```
void SetTrackBar()
 {
 if (type=="V")
{
 trackBar1.Maximum=Convert.ToInt32(vp.Duration);
 }
 else if (type=="A")
 {
 trackBar1.Maximum=Convert.ToInt32(ap.Duration);
 }
 else if (type=="R")
 {
 trackBar1.Maximum=Convert.ToInt32(rp.GetLength());
}
}
```

أسم الإجراء هو SetTrackBar ونقوم فيه بفحص نوع الملف كما شرحنا في الدرس السابق ,فإذا كان نوع الملف صوت أو فيديو فإنا نأخذ طول الملف عبر الخاصية Duration,ثم نحول الرقم إلى عدد صحيح عبر كائن التحويل Convert.ToInt32

والرقم الناتج نحمله لخاصية Maximum في كائن التنقل , وبهذا مهما تنقلنا في الكائن فأن القيمة الناتجة ستكون بين 0 و طول الملف ,إما إذا كان نوع الملف rm فإنا نأخذ طول الملف عبر الخاصية GetLength ونحول الرقم إلى عدد صحيح ونحمله للخاصية Maximum .

ذلك هو إجراء ضبط القيمة العظمى لكائن التنقل ,لكن أين نستدعي هذا الإجراء ؟؟؟

أفضل مكان لاستدعاء هذا الإجراء هو بعد تحميل الملف أي بعد استدعاء إجراء التحميل LoadFile

أفتح كود زر فتح الملف وأضف تعليمة الاستدعاء بعد تعليمة استدعاء إجراء تحميل الملف:

كود

```
OF.ShowDialog();
  LoadFile(OF.FileName);
SetTrackBar();
  PlayFile();
```

اللون الأزرق يمثل الزيادة التي أضفناها إلى الشفرة. بعد أن قمنا بضبط القيمة العظمى لكائن التنقل نحتاج لكود التنقل عبر الملف. إي أننا نحتاج لكود يقوم بالتنقل عبر الملف بمجرد تغيير مؤشر كائن التنقل .

أضغط مزدوجا على كائن التنقل لتنتقل إلى الكود الذي تنفذ بمجرد تغيير المؤشر **TrackBar1Scroll**

أنسخ الكود التالي :

كود

```
if (type=="V")
{
  vp.CurrentPosition=trackBar1.Value;
  }
else if (type=="A")
{
  ap.CurrentPosition=trackBar1.Value;
}
  else if (type=="R")
  {
  rp.SetPosition(trackBar1.Value);
}
```

عند تغيير مؤشر كائن التغيير سيقوم المعالج بتنفيذ الكود السابق .

trackBar1.Value ترجع رقم بين **0** و طول الملف على حسب موقع المؤشر ,حيث نقوم بفحص نوع الملف فإذا كان فيديو أو صوت نعدل موقع التشغيل عبر الخاصية **CurrenPosition**

أما إذا كان rm فنعدل موقع التشغيل عبر الخاصية **SetPosition**.

الآن نحتاج إلى كود لتحريك مؤشر كائن التنقل ليحدد الموقع الذي وصل إليه التشغيل , لعمل ذلك نحتاج لاستخدام المؤقت حيث نقوم كل ثانية بفحص موقع التشغيل ونعدل موقع المؤشر.

أضغط مزدوجاً على كائن المؤقت لتنتقل إلى الكود الذي ينفذ كل ثانية وأنسخ الكود التالي :

كود

```
if (type=="V")
  {
  trackBar1.Value=Convert.ToInt32(vp.CurrentPosition);
  }
  else if (type=="A")
  {
trackBar1.Value= Convert.ToInt32(ap.CurrentPosition);
  }
  else if (type=="R")
{
  trackBar1.Value=Convert.ToInt32(rp.GetPosition());
}
```

الكود السابق سينفذ كل ثانية لأننا حددنا خاصية **Interval** في المؤقت إلى **1000**

يقم الكود السابق بفحص نوع الملف فإذا كان صوت أو فيديو نقوم بأخذ موقع التشغيل الحالي ,عبر الخاصية **CurrentPosition** ونحولها إلى عدد صحيح ونحمل الرقم الناتج إلى الخاصية **Value**, حيث أن الخاصية **Value** تمثل موقع مؤشر كائن التنقل .

وإذا كان الملف **rm** نأخذ طول الملف عبر الخاصية **GetPosition** ونحولها إلى عدد صحيح ,ونحملها في خاصية موقع المؤشر **Value**.

الكود السابق يقوم بتغيير موقع مؤشر كائن التنقل على حسب موقع التشغيل الذي وصل إليه الملف.

بقي لنا كود زري التقديم والتأخير ..

أضغط مزدوجاً على زر التقديم وأنقل الكود التالي :

كود

```
if (type=="V")
{
  vp.CurrentPosition=vp.CurrentPosition+20;
  }
  else if (type=="A")
{
ap.CurrentPosition=ap.CurrentPosition+20;
  }
  else if (type=="R")
  {
  rp.SetPosition(rp.GetPosition()+20000);
}
```

في الكود السابق نقوم بتعديل موقع التشغيل ليكون موقع التشغيل الحالي + 20 ثانية

أي أننا إذا ضغطنا على زر التقديم فان التشغيل سيقفز 20 ثانية إلى الأمام

يمكن وضع عدد أكبر من العشرين ثانية إذا أردت أو أصغر ..

لاحظ أنه إذا كان نوع الملف rm فإننا أضفنا إلى الموقع الحالي 20000 لأن مشغل rm يتعامل بالملي ثانية

شغل البرنامج وأفتح ملف ميديا وجرب تحريك مؤشر كائن التنقل.

واجب الدرس السابع

*** أكتب كود زر التأخير ..**

*** قد يحدث خطأ في زري التقديم والتأخير إذا تم التأخير لأقل من صفر أو التقديم لأكثر من طول الملف**

لذلك يجب وضع كود الزرين داخل جملة try هكذا:

كود

```
try
{
//كود التقديم أو التأخير
}
catch
{


}
```

عند حدوث خطأ لن يتم تنفيذ أي شيء.

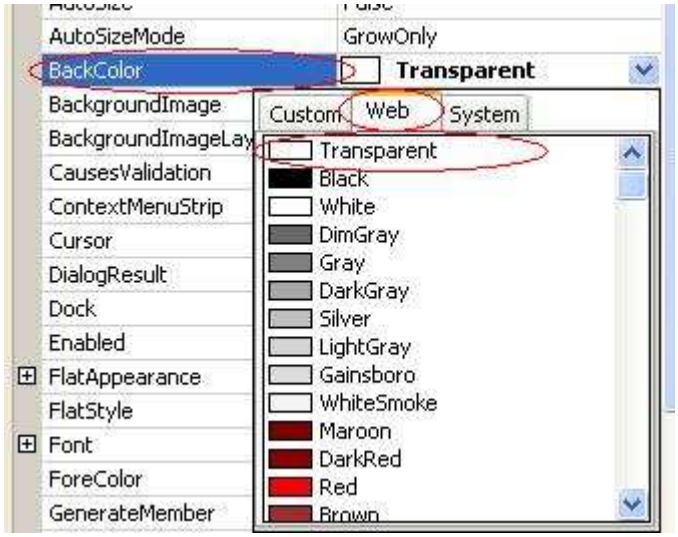
*** أرسل التطبيق.**

ملاحظات:

*** تأكد من اتجاه مؤشر كائن التنقل من خلال الخاصية RightToLeft.**

*** لجعل خلفية الزر شفافة عدل خاصية FlatStyle إلى Popup**

و خاصية BackColor إلى Transparent



- **يمكن تغيير الصور في الأزرار بواسطة الخاصية Image.**

*** يمكن تعديل لون خلفية الكائن الحاضن للفيديو إلى شفاف لتصبح كما في الصورة .**

- للأسف ليست كل الكائنات تدعم الخلفية الشفافة , فكائن التنقل مثلاً لا يدعمها ..

ملاحظة لمستخدمي بيئة Sharp Developer:

أحيانا عند الضغط المزدوج على المؤقت يحصل خطأ وتغلق بيئة التطوير .ولكتابة كود المؤقت إذا حدث هذا الخطأ دون الضغط مزدوجاً على المؤقت:

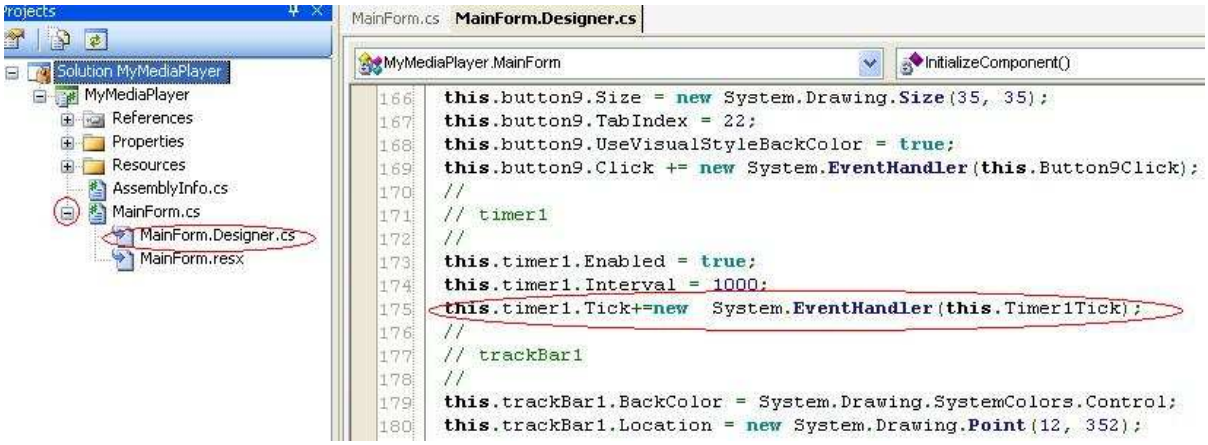
افتح شفرة التصميم من خلال فتح مستعرض الملفات ثم فتح MainForm.Designer.cs

و ابحث عن كود تعريف المؤقت وأضف بعده هذا السطر:

كود

```
this.timer1.Tick+=new System.EventHandler(this.Timer1Tick);
```

كما في الصورة :



ثم أفتح شفرة البرنامج عند الإجراءات من خلال اختيار MainForm.cs

وأضف هذا الإجراء وتأكد أنه خارج أي إجراء:

كود

```
void Timer1Tick(object sender, System.EventArgs e)
  {
  if (type=="V")
{
  trackBar1.Value=Convert.ToInt32(vp.CurrentPosition);
}
  else if (type=="A")
  {
trackBar1.Value= Convert.ToInt32(ap.CurrentPosition);
}
  else if (type=="R")
{
  trackBar1.Value=Convert.ToInt32(rp.GetPosition());
}
}
```

نهاية الدرس السابع.

..

الدرس الثامن

المصفوفات:

المصفوفات هي عبارة عن مجموعة متغيرات من نفس النوع متراصة ومترابطة داخل كائن **Array**.

ويتم الوصول لأي عنصر في هذه المجموعة من خلال ترتيبه في المجموعة **Index**.

مثلاً : مصفوفة أيام الأسبوع تعتبر مجموعة متغيرات من نوع نص, و مصفوفة أيام الشهر تعتبر مجموعة متغيرات من نوع رقم ..

يتم تعريف المصفوفة بإحدى ثلاث طرق :

1-طريقة تعريف أسم المصفوفة فقط

التعريف بهذه الطريقة نادراً ما يستخدم , ويحدد فقط أسم لكائن المصفوفة بدون أن يقوم بإنشاءه:

كود

```
string[] days;
```

الكود السابق يعرف أسم **days** لمصفوفة عناصرها من نوع نص, لاحظ القوسين المربعين بعد كلمة **string** دلالة على أن **days** تمثل مصفوفة نصوص وليس نص .

2-طريقة تعريف المصفوفة وتحديد عدد عناصرها

طريقة التعريف هذه تعرف و تنشئ كائن مصفوفة وتحدد عدد عناصر هذه المصفوفة:

كود

```
string[] days= new string[7];
```

لأن المصفوفة هي كائن وجب علينا تعريفه باستخدام عبارة **new** ,والرقم **7** بين قوسي تعريف الكائن هو عدد عناصر المصفوفة.

أي أن الكود السابق عرف مصفوفة أسمها days تحتوي على 7 عناصر من نوع نص. لكن هذه الطريقة لا تحمل قيم لعناصرها , أي أنها تحتوي على 7 نصوص فارغة.

3-طريقة تعريف المصفوفة وتحميل قيم لعناصرها

هذه الطريقة تعرف وتنشئ كائن المصفوفة وتحمل قيم لجميع عناصر المصفوفة:

كود

```
string[] days= new string[]{"السبت","الأحد","الاثنين","الثلاثاء","الأربعاء","الخميس","الجمعة"};
```

في الكود السابق قمنا بتعريف كائن المصفوفة وقمنا بتحميل القيم لجميع عناصره ,بإدخالها كمجموعة نصوص بين حاصرتين تفصل بين كل نص وآخر علامة , لاحظ أنا لم نحدد عدد عناصر المصفوفة حيث سيقوم المعالج بتحديدها آلياً من خلال عد النصوص المدخلة بين الحاصرتين .

ملاحظة : العناصر المدخلة بين حاصرتين يجب أن تكون من نفس نوع عناصر المصفوفة

الوصول إلى عناصر المصفوفة:

كما قلنا سابقاً يتم الوصول إلى عناصر المصفوفة من خلال ترتيب العنصر المراد الوصول إليه .والترتيب في المصفوفات يبدأ من الصفر , أي أن أول عنصر في المصفوفة ترتيبه صفر.

فمثلاً في مصفوفة الأيام , يوم السبت ترتيبه صفر و الأحد ترتيبه 1 ..... والجمعة ترتيبه 6.

ويتم الوصول إلى العنصر المحدد بكتابة أسم المصفوفة الموجود بها متبوعا بقوسين مربعين بينهما ترتيب العنصر.

ماذا لو أردنا إظهار رسالة تخبرنا بالعنصر الذي ترتيبه 4 في مصفوفة الأيام :

كود

```
string[] days= new string[]{"السبت","الأحد","الإثنين","الثلاثاء","الأربعاء","الخميس","الجمعة"};
MessageBox.Show(days[4]);
```

لاحظ كيف تم الوصول إلى العنصر الرابع في مصفوفة days من خلال العبارة days[4].

الكود السابق ينتج هذه الرسالة :

يمكن أيضاً تعديل أي عنصر من عناصر المصفوفة من خلال ترتيبه في المصفوفة :

كود

```
string[] days= new string[]{"السبت","الأحد","الاثنين","الثلاثاء","الأربعاء","الخميس","الجمعة"};
days[4]="بعد التعديل الأربعاء";
MessageBox.Show(days[4]);
```

في السطر الثاني غيرنا قيمة العنصر الذي ترتيبه 4 إلى "الأربعاء بعد التعديل"

الكود السابق ينتج الرسالة التالية :

القوائم

مع أن المصفوفات تعتبر من أهم هياكل البيانات التي استخدمت في لغات البرمجة منذ نشأتها .إلا أن لها عدداً من العيوب التي سعت C#2005 لحلها من خلال إدخال نوع جديد من هياكل البيانات .أهم العيوب هو حجم المصفوفة الثابت , فعند تعريف مصفوفة وإسناد قيم لها أو تعريف حجمها ,لن تستطيع أن تزيد من حجمها أو تنقص منه , وأيضاً لا توجد طريقة فعالة للبحث بين العناصر .

مثلاً في مصفوفة أيام الأسبوع لا يوجد طريقة للبحث عن يوم السبت مثلاً إلى باستخدام دوارة تمر بجميع عناصر المصفوفة:

كود

```
for(int i=0;i<days.Lehgth;i++)
{
if (days[i]=="السبت")
{
MessageBox.Show("السبت هذا يوم");
}
}
```

هيكل البيانات الجديد الذي تفادى عيوب المصفوفات هو القائمة ,حيث يتم تعريف القائمة وتعريف أنواع عناصرها هكذا:

كود

```
List<string> dayslist;
```

ولأن القائمة كائن وليست متغير , فيجب استخدام عبارة new ليصبح التعريف الصحيح هكذا:

كود

```
List<string> dayslist=new List<string>();
```

لاحظ أننا لم نحدد لها حجم أو عدد لعناصر لأن حجمها متغير , حيث تستطيع أن تضيف لها عنصر هذا:

كود

```
dayslist.Add("السبت");
```

كما أنك تستطيع أن تضيف لها عناصر مصفوفة كاملة:

كود

```
dayslist.AddRange(days);
```

الكود السابق يضيف عناصر المصفوفة **days** إلى القائمة **dayslist**

وتستطيع أن تحذف منها عنصر هكذا :

كود

```
dayslist[5].Remove();
```

الكود السابق يحذف العنصر الذي ترتيبه **5** من القائمة

كما أنك تستطيع البحث عن عنصر ما هكذا :

كود

```
dayslist.Contains("السبت");
```

الدالة السابقة سترجع القيمة **true** إذا كان العنصر موجود في القائمة ,أو ترجع القيمة **false** إذا لم يوجد العنصر في القائمة.مما سبق نستنتج أن القائمة أفضل بكثير وأسهل في الاستخدام من المصفوفة ,وهذا ما سنستخدمه في تطبيقنا.

تطبيق الدرس الثامن:

تشغيل أكثر من ملف

سنضيف خاصية جديدة للبرنامج وهي فتح أكثر من ملف وإضافتها إلى قائمة التشغيل .حيث سنستخدم قائمة عناصرها من نوع نص لتخزين مسارات الملفات في قائمة التشغيل .

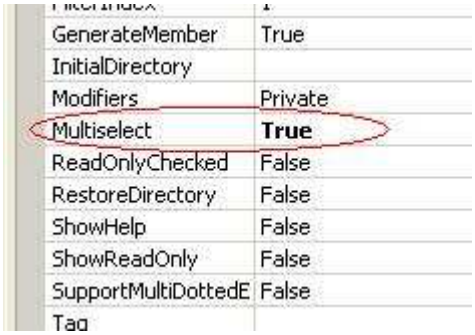
أفتح تطبيق الدرس السابق , ومن صندوق الأدوات قم بسحب أداة القائمة :

من خلال الخاصية (Name) في جدول الخصائص قم بتغيير أسم القائمة من ListBox1 إلى pl ,ليسهل التعامل معها من الشفرة , هذه هي قائمة التشغيل.

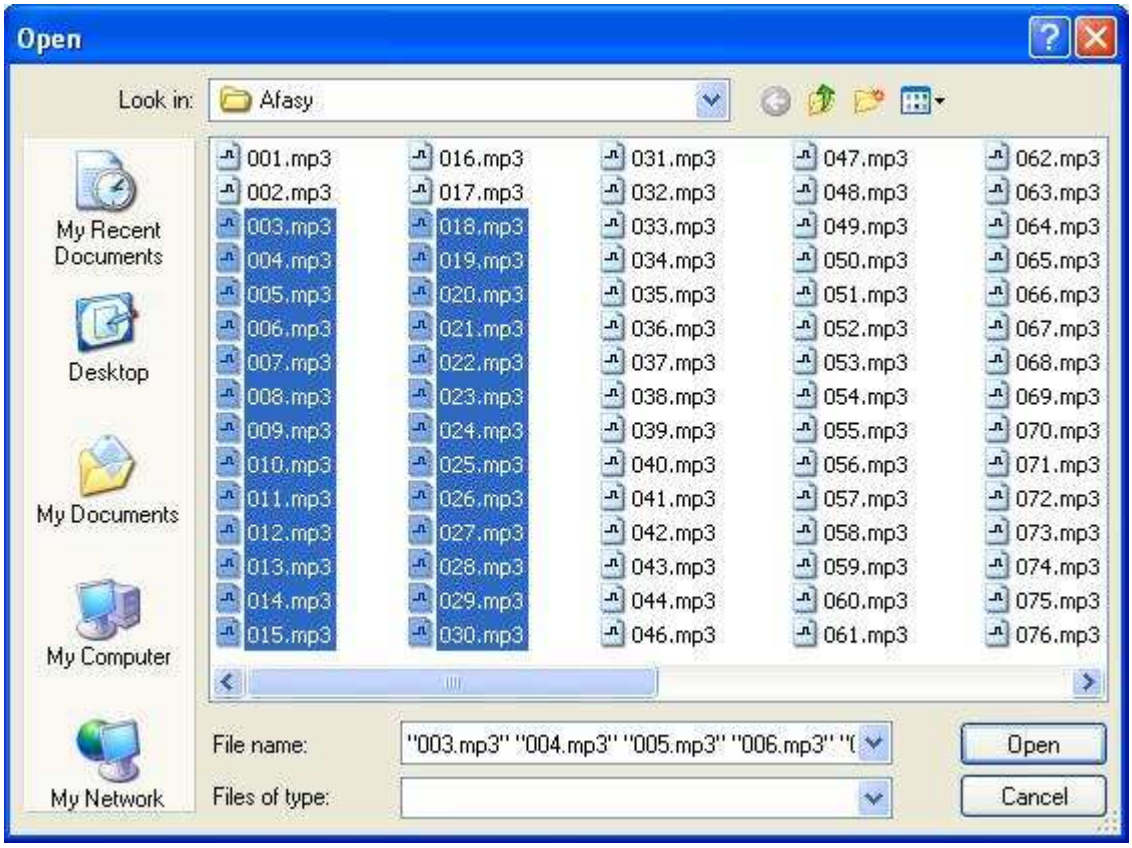
قم بإعادة ترتيب الأدوات في النافذة حتى يصبح شكلها هكذا :

قم بتحديد أداة OF في الشريط الأصفر تحت النافذة:

وأذهب إلى جدول الخصائص وقم بتغيير خاصية Multiselect إلى True



عند إسناد قيمة True إلى هذه الخاصية فإنها تسمح لأداة اختيار الملفات باختيار أكثر من ملف واحد كل مرة:



الآن سنعرف متغير عام من نوع قائمة عناصرها نصوص لخزن مسارات الملفات المختارة بأداة اختيار الملفات ,كائن القائمة موجود في فضاء الأسماء **System.Collections.Generic** لذلك يجب إضافة عبارة **using** لتضمين كائنات ودوال الفضاء .

أذهب إلى شفرة النافذة وأضف عبارة **using** التالية تحت عبارات **using** الموجودة في رأس الصفحة :

كود

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

أذهب إلى شفرة البرنامج وقم بكتابة الكود التالي تحت كود تعريف كائنات **Audio** و **Vedio** و **type** :

كود

```
public List<string> albume=new List<string>();
```

```
public partial class MainForm
{
    private Audio ap;
    private Video vp;
    private AxRealAudio rp=new AxRealAudio();
    private string type;
    public List<string> albume=new List<string>();
    [STAThread]
    public static void Main(string[] args)
```

تحميل قيم القائمة

أداة اختيار الملفات OF تحتوي على خاصية استخدمناها في الدروس السابقة .وهي خاصية FileName , وهذه الخاصية تحمل قيمة من نوع نص للملف الذي قام المستخدم باختياره ,لكن ماذا لو اختيار المستخدم أكثر من ملف واحد ؟

لحسن الحظ يوجد خاصية أخرى أسمها FileNames "بزيادة حرف s عن الخاصية الأخرى"

والخاصية FileNames تحمل مصفوفة جميع الملفات الذي قام المستخدم باختيارها ,وعناصر المصفوفة من نوع نص لخزن مسارات الملفات المختارة .

فمثلاً لو أختار المستخدم 3 ملفات سيكون عدد عناصر هذه المصفوفة 3 .

أذهب إلى التصميم وأضغط مزدوجاً على زر فتح ملف ستجد الكود التالي :

كود

```
OF.ShowDialog();
  LoadFile(OF.FileName);
  SetTrackBar();
  PlayFile();
```

وغيره إلى الكود إلى التالي :

كود

```
OF.ShowDialog();
albume.AddRange(OF.FileNames);
  LoadFile(albume[0]);
FillList();
  SetTrackBar();
  PlayFile();
```

في السطر الثاني قمنا بتحميل جميع عناصر المصفوفة OF.FileNames إلى القائمة albume ,والمصفوفة OF.FileNames كما شرحنا سابقاً تحتوي على عناصر تحمل مسارات الملفات المختارة.وقمنا بإضافة عناصرها إلى القائمة albume والتي سنستخدمها في أنحاء البرنامج ,في السطر الثالث قمنا بتحميل العنصر الذي ترتيبه صفر من قائمة albume حتى يتم تشغيله ,وفي السطر الرابع استدعينا إجراء FillList الذي يقوم بملء قائمة التشغيل بأسماء الملفات في عناصر القائمة albume كما سنشرح في الفقرة التالية .

إجراء Fill List

هذا الإجراء سيقوم بنقل جميع عناصر القائمة albume إلى قائمة التشغيل ,حتى يسهل الوصول للمقطع الصوتي المراد تشغيله , لكن القائمة albume تحتوي على مسارات الملفات كاملة , وقائمة التشغيل لا تتسع لكتابة مسار الملف كاملاً نريد فقط أسماء الملفات في قائمة التشغيل ,لحسن الحظ يوجد كائن في إطار العمل داخل الفضاء System.IO يقوم بهذا العمل .الكائن Path يحتوي على دالة GetFileName تقوم بأخذ مسار الملف كاملاً وتنتج لنا أسم الملف فقط إذا كان مسار الملف هو :

كود

```
c:\sounds\s1\track.mp3
```

فالدالة GetFileName تأخذ المسار كامل وترجع القيمة التالية :

كود

```
track.mp3
```

يتم استدعاء الكائن Path من خلال كتابة عنوناه الكامل :

كود

```
string FileName;
FileName=System.IO.Path.GetFileName("c://sounds//s1//track.mp3");
MessageBox.Show(FileName);
```

في السطر الأول عرفنا متغير من نوع نص ليحمل نتيجة الدالة GetFileName ,في السطر الثاني قمنا باستدعاء الدالة GetFileName من خلال كتابة عنوانها كاملاً ,وأسندنا ناتج الدالة للمتغير FileName ,السطر الثالث سينتج رسالة مكتوب عليها :

track.mp3

بعد أن شرحنا عمل الدالة GetFileName سنحتاج في هذا الإجراء أيضاً لدوارة تقوم بالمرور على كافة عناصر القائمة albume

يمكنك الرجوع للدرس الثالث ومراجعة الدوارات.

تبدأ الدوارة من 0 إلى عدد عناصر القائمة , حيث يمكن معرفة عدد عناصر القائمة من خلال الخاصية Count :

كود

```
albume.Count;
```

أفتح شفرة البرنامج , وأنسخ الكود التالي وتأكد أن يكون خارج أي إجراء آخر :

كود

```
        void FillList()
        {
pl.Items.Clear();
            for(int i=0;i<albume.Count;i++)
            {
            string FileName;
            FileName=System.IO.Path.GetFileName(albume[i]);
            pl.Items.Add(FileName);
            }
        }
```

الأمر pl.Items.Clear يقوم بتنظيف القائمة من أي عناصر موجودة مسبقاً حتى تستعد لإدخال العناصر الجديدة.

الكود داخل حاصرتي for سيتم تنفيذه بعدد عناصر القائمة albume, في السطر الأول من الكود داخل حاصرتي for , قمنا بتعريف الكائن الذي سيحمل نتيجة الدالة GetFileName. وفي السطر الثاني استدعينا الدالة GetFileName ومررنا لها العنصر الذي ترتيبه i من قائمة albume. طبعاً في أول مرة يتم تنفيذ الكود سيكون i يساوي صفر , والمرة الثانية 1 والثالثة 2 .

وهكذا فإن قيمة i متغيرة من مرة لأخرى .

في السطر الثالث أضفنا أسم المكلف الناتج إلى قائمة التشغيل pl

..

عند تشغيل البرنامج حالياً واختيار عدة ملفات ستنتقل جميع أسماء الملفات إلى قائمة التشغيل .ويبدأ البرنامج بتشغيل أول ملف في القائمة :

اختيار الملف من قائمة التشغيل

أذهب إلى التصميم , وأنقر مزدوجاً على قائمة التشغيل , ستنتقل إلى شفرة البرنامج ,أنسخ الكود التالي حيث مؤشر الفارة بين حاصرتي إجراء **plSelectedIndexChanged** "

كود

```
StopFile();
 LoadFile(albume[pl.SelectedIndex]);
 SetTrackBar();
 PlayFile();
```

حيث يتم تنفيذ الكود السابق كل مرة تقوم بها باختيار عنصر من قائمة التشغيل. عند اختيار عنصر من قائمة التشغيل , فإن السطر الأول يقوم بايقاف تشغيل الملف الحالي ,والسطر الثاني يأخذ ترتيب العنصر المختار من قائمة التشغيل من خلال الخاصيةSelectedIndex ,ويقوم بتحميل الملف من قائمة albume الذي ترتيبه هو نفس ترتيب العنصر المختار في قائمة التشغيل . السطر الثالث والرابع تم شرحهما سابقاً ..

الآن قم بتشغيل البرنامج وأختر أكثر من ملف , سيتم نقل أسماء الملفات لقائمة التشغيل ,ثم تشغيل أول ملف في القائمة , وعند اختيار ملف آخر من قائمة التشغيل سيتوقف تشغيل الملف الحالي .

ويبدأ تشغيل الملف المختار

خطأ شائع:

أحيانا يقم المؤقت بتغيير قيمة شريط التنقل قبل أن يتم تحميل الملف ,وهذا يسبب خطأ لأن قيم شريط التنقل الكبرى والصغرى لم يتم ضبطها بعد , ولا يجوز تغيير قيمة شريط التنقل قبل ضبط قيمه الصغرى والكبرى

لمعالجة الخطأ أذهب إلى شفرة إجراء المؤقت Timer1Tick :

كود

```
void Timer1Tick(object sender, System.EventArgs e)
{

   if (type=="V")
   {
     trackBar1.Value=Convert.ToInt32(vp.CurrentPosition);
   }
   else if (type=="A")
   {
      trackBar1.Value= Convert.ToInt32(ap.CurrentPosition);
   }
   else if (type=="R")
   {
      trackBar1.Value=Convert.ToInt32(rp.GetPosition());
   }
}
```

وأضف إلى بدايتها الكود التالي:

كود

```
if (trackBar1.Maximum == 0)
{
    SetTrackBar();
}
```

وهذا الكود يفحص ما إذا كانت قيمة شريط التنقل الكبرى قد تم ضبطها .فإن كانت قيمتها 0 يعني أنه لم يتم ضبطها بعد وبالتالي يستدعي إجراء ضبطها مرة أخرى .حيث يصبح شكل الإجراء هكذا بعد وضعه داخل عبارة try لتفادي الأخطاء الأخرى :

كود

```
void Timer1Tick(object sender, System.EventArgs e)
{
try    {
    if (trackBar1.Maximum == 0)
    {
        SetTrackBar();
    }
    if (type=="V")   {
      trackBar1.Value=Convert.ToInt32(vp.CurrentPosition);
    }
    else if (type=="A")
    {
        trackBar1.Value= Convert.ToInt32(ap.CurrentPosition);
    }
    else if (type=="R")
    {
        trackBar1.Value=Convert.ToInt32(rp.GetPosition());
    }
}
catch
{
}
}
```

واجب الدرس الثامن:

أرسل التطبيق.

نهاية الدرس الثامن.

..

الدرس التاسع

إنتاج أرقام عشوائية

ربما يتساءل البعض ما الذي يدعونا لإنتاج أرقام عشوائية وفي إي شيء سنستخدمها حسناً , يعتبر إنتاج الأرقام العشوائية من أهم الدعائم التي ترتكز عليها أنظمة الذكاء الاصطناعي والشبكات العصبية الاصطناعية .وإنتاج الأرقام العشوائية ليس بالسهولة التي تتصورها .حيث تستخدم خوارزمية شديدة التعقيد لغرض أنتاج أرقام لا معنى لها ,تكمن الصعوبة في الخوارزمية أن أنظمة الحاسوب لاستطيع أن تنتج أي شي لا معنى له .

فلا يوجد شيء "عشوائي" في أنظمة الحاسوب .

ولكن خوارزميات إنتاج الأرقام العشوائية تستخدم الكثير من المدخلات من مصادر مختلفة لإنتاج الأرقام العشوائية كأن تأخذ تاريخ ووقت النظام , وتجري عليه عددا من الدوال والحسابات في برنامجنا سنضيف اليوم خاصية اختيار طريقة تشغيل الملفات , إما تتابعيه أو عشوائية .تتابعيه يعني أن يشغل البرنامج الملف الأول في قائمة التشغيل وعند الانتهاء منه يشغل الملف الثاني وهكذا .

أما الطريقة العشوائية فهي أن يقوم البرنامج بالحصول على رقم عشوائي بين الصفر و عدد الملفات في قائمة التشغيل ويبدأ بتشغيله .وعند الانتهاء منه يحصل على رقم عشوائي ويشغل ملف آخر وهكذا.

في لغة C# يوجد كائن مسئول عن إنتاج الأرقام العشوائية هو كائن Random

وعند تعريفة ولأنه كائن وليس متغير يجب استخدام كلمة new هكذا :

كود

```
Random rnd=new Random();
```

في الكود السابق قمنا بتعريف كائن أسمه rnd من نوع Random, الدالة Next في هذا الكائن هي المسئولة عن إنتاج الأرقام العشوائية, وتستقبل هذه الدالة رقمين يمثلا الحد الأدنى والحد الأقصى للعدد العشوائي المراد إنتاجه .

مثلاً لو أردنا إنتاج رقم عشوائي بين 0 و 10 :

كود

```
rnd.Next(0,10);
```

حيث ستنتج هذه الدالة رقماً عشوائيا بين الصفر والعشرة ,ولأن هذه الدالة تعيد قيمة فإن استدعائها بالطريقة السابقة خاطئ , حيث يجب إسنادها لمتغير من نوع رقم:

كود

```
int num;
num=rnd.Next(0,10);
```

حيث أن المتغير num سيحمل القيمة الناتجة من الدالة .

تطبيق الدرس التاسع

أفتح تصميم البرنامج , ومن صندوق الأدوات قم بسحب كائن صندوق المجموعة Group Box إلى الفورم الرئيسي:

قم باختيار صندوق المجموعة ومن جدول الخصائص أذهب إلى خاصية Text

وأكتب أمامها "طريقة التشغيل" , ستلاحظ أن عنوان صندوق المجموعة قد تغير :

من صندوق الأدوات أيضاً قم بسحب كائن زر الاختيار Radio Button مرتين إلى صندوق المجموعة :

تأكد من أن زري الاختيار قد تم وضعها داخل صندوق المجموعة :

أختر زر الاختيار المسمى **radioButton1** وأذهب إلى جدول الخصائص

وغير الخاصية **(Name)** من **radioButton1** إلى **serbtn**

وغير الخاصية **Text** من **radioButton1** إلى **تتابعي**

وغير أيضاً الخاصية **Checked** إلى **true**.

أختر زر الاختيار الأخر المسمى **radioButton2** وأذهب إلى جدول الخصائص

وغير الخاصية **(Name)** من **radioButton2** إلى **rndbtn**

وغير أيضاً الخاصية **Text** من **radioButton2** إلى **عشوائي**

عدل ترتيب الأدوات في النافذة حتى يصبح شكلها هكذا :

سنحتاج إلى متغير عام من نوع رقم وليكن **current** حيث سيقوم بخزن ترتيب الملف الذي يتم تشغيله حالياً

قم بإضافة هذا المتغير أسفل المتغيرات العامة من الدروس السابقة :

```
public partial class MainForm
{
    private Audio ap;
    private Video vp;
    private AxRealAudio rp=new AxRealAudio();
    private string type;
    public List<string> albume=new List<string>();
    private int current;

    [STAThread]
    public static void Main(string[] args)
```

إجراء تشغيل الملف الحالي

الإجراء **Play Current** وظيفته هي تشغيل الملف الذي ترتيبه هو قيمة **current** من القائمة **albume**

مثلاً إذا كانت قيمة **current** هي **3** فإن هذا الإجراء يشغل الملف من قائمة التشغيل والذي ترتيبه **3**

أنسخ هذا الإجراء إلى شفرة البرنامج وتأكد أنه خارج أي إجراء آخر:

كود

```
void PlayCurrent()
{
    LoadFile(albume[current]);
    SetTrackBar();
    PlayFile();
}
```

إجراء اختيار الملف التالي

وظيفة الإجراء SetNext هي اختيار الملف التالي لتشغيله بعد الانتهاء من تشغيل الملف الحالي .حيث يقوم هذا الإجراء بفحص طريقة التشغيل المختارة من قبل المستخدم ومن خلالها تقوم باختيار الملف التالي بطريقتين مختلفتين:

كود

```
if(rndbtn.Checked)
{
  //current 1 إلى المتغير إضافة
}
else
{
  // current للمتغير إختيار رقم عشوائي وتحميلة
}
```

أنسخ الإجراء التالي إلى شفرة البرنامج وتأكد أنه خارج أي إجراء آخر:

كود

```
void SetNext()
{
   if(serbtn.Checked)
   {
       current=current+1;
   }
   else
   {
           Random rnd=new Random();
       current=rnd.Next(0,albume.Count-1);
   }
     pl.SelectedIndex = current;
     PlayCurrent();
}
```

في السطر الثالث من الكود السابق توجد عبارة if والتي تفحص زر الاختيار serbtn الذي خصصناه للتشغيل ألتتابعي فإذا كانت حالة الزر Ckecked أي أن الاختيار عليه , أي أن نوع التشغيل المختار من قبل المستخدم هو ألتتابعي عندها في السطر الخامس نقوم بإضافة 1 إلى قيمة current , فإذا كانت قيمته مثلاً 5 ستصبح قيمته 6 إما إذا كذا الاختيار على الزر الآخر وهو المخصص للتشغيل العشوائي ,ففي السطر التاسع قمنا بتعريف كائن من نوع Random, وفي السطر العاشر أسندنا للمتغير current القيمة الناتجة من دالة إنتاج الأرقام العشوائية ,حيث أن الدالة ستنتج رقماً عشوائيا بين الصفر و ترتيب آخر عنصر في قائمة التشغيل ,ونعرف أن عدد عناصر قائمة التشغيل هو طول نفس عدد عناصر القائمةalbume

ولكن عدد عناصر القائمة يختلف عن الترتيب لأنه كما قلنا في الدرس السابق فإن الترتيب في المصفوفات والقوائم يبدأ من الصفر

فإذا كان طول القائمة 5 فإن أعلى ترتيب فيها هو 4 لذلك أضفنا -1 بعد طول المصفوفة

في السطر الثاني عشر قمنا بتغيير الاختيار في قائمة التشغيل من خلال الخاصية SelectedIndex

مثلاً لو أردنا أن نختار العنصر الثالث من قائمة التشغيل سنستخدم الكود التالي :

كود

```
pl.SelectedIndex=2;
```

لاحظ 2 و ليست 3 لأن العنصر الثالث في القائمة ترتيبه 2 وليس 3 حيث إن العنصر الأول كما قلنا ترتيبه 0 والعنصر الثاني ترتيبه 1 والعنصر الثالث ترتيبه 2 و هكذا .

في السطر الثالث عشر قمنا باستدعاء الإجراءPlayCurrent والذي يقوم بتشغيل الملف الذي ترتيبه هو قيمة current

هناك ملاحظة على الكود السابق

فإذا كانت طريقة التشغيل تتابعي , ووصل ترتيب الملف إلى آخر عنصر .عندها إذا أضفنا 1 إلى المتغير current فإنه سيحمل ترتيب أعلى من ترتيب آخر عنصر في القائمة وهذا سوف يسبب خطأ .

عند محاولة تحميل ملف غير موجود أساسا في قائمة الملفات albume وقائمة التشغيل .لذلك يجب أن نفحص قيمة current أولاً فإذا كانت تساوي آخر ترتيب في القائمة فنقوم بإسناد الرقم 0 إلى المتغير current حتى يرجع ترتيب المتغير لأول عنصر في القائمة , وبالتالي عند الانتهاء من تشغيل جميع الملفات في القائمة فسيعود تشغيل الملفات من أول القائمة مرة أخرى

في كود إجراء اختيار الملف أحذف السطر التالي :

كود

```
current=current+1;
```

وأكتب مكانه هذه الأسطر :

كود

```
if(current==albume.Count-1)
{
    current=0;
}
else
{
   current=current+1;
}
```

عبارة if في السطر الأول تقوم بفحص قيمة current فإذا كانت تساوي ترتيب آخر عنصر بالقائمة ففي السطر الثالث تسند القيمة 0 إلى المتغير current

أما إذا كانت قيمة current لا تساوي ترتيب آخر عنصر ففي السطر السابع تضيف 1 إلى قيمة current

بالتعديل السابق يصبح إجراء اختيار الملف التالي هكذا :

كود

```
void SetNext()
{
   if(serbtn.Checked)
   {
      if(current==albume.Count-1)
      {
          current=0;
      }
      else
      {
         current=current+1;
      }
   }
   else
   {
      Random rnd=new Random();
      current=rnd.Next(0,albume.Count-1);
   }
   pl.SelectedIndex = current;
   PlayCurrent();
}
```

متى نستدعي إجراء اختيار الملف التالي ؟

إذا قمت بتشغيل البرنامج فلن يتم تشغيل الملفات لا عشوائيا و لا تتابعيا ,لأننا قمنا بكتابة كود إجراء الاختيار ولكنا لم نستدعه في أي مكان في البرنامج .والمفروض أن نستدعي إجراء اختيار الملف التالي عند الانتهاء من تشغيل الملف الحالي .

ولكن كيف نعرف متى سيتم الانتهاء من تشغيل الملف الحالي ؟

حسنا نعرف أن قيمة شريط التنقل عبر الملف تتغير كل ثانية بفعل المؤقت

سنستخدم هذه الميزة لنعرف وقت الانتهاء من تشغيل الملف الحال ..

حيث أنه عند الانتهاء من تشغيل الملف فإن قيمة شريط التنقل تصبح مساوية لطول الملف ,إذا سنختبر قيمة شريط التنقل في كل مرة تتغير فيها فإذا أصبحت قيمته مساوية لطول الملف يعني أنه وصل إلي نهاية الملف الحالي .

وبالتالي يجب استدعاء إجراء SetNext

لكن هناك مشكلة , فالمؤقت يغير قيمة الشريط كل ثانية واحدة , وطول الملف في ملفات الريل يقاس بالملي ثانية أي انه عند وصول الملف لنهايته فهناك احتمال كبير أن لا تكون قيمة الشريط هي طول الملف .

لذلك يجب أن نجد طريقة أخر لفحص الصول إلى نهاية الملف الطريقة هي أن نفحص نوع الملف الذي يتم تشغيله حالياً فإذا كان ملف ريل فالفحص يتم على آخر 1000 رقم ,أما إذا كان ملف صوت عادي أو ملف فيديو فالفحص يتم على أخر رقم

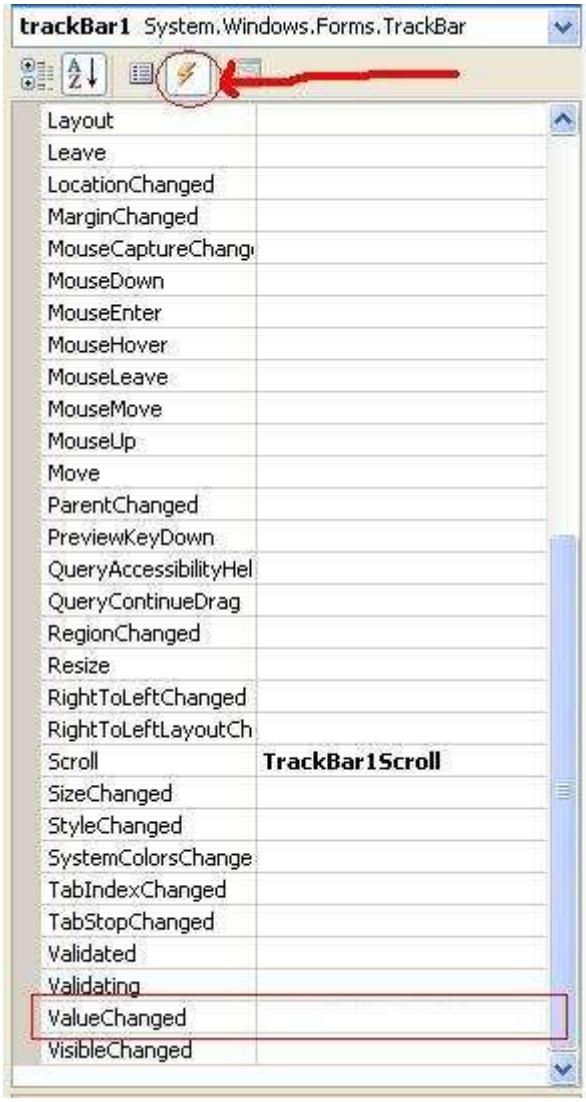
أي أن شرط الوصول إلى نهاية ملف الريل يتحقق إذا كانت قيمة شريط التنقل أكبر من طول الملف ناقص 1000

وشرط الوصول إلى نهاية ملف الصوت العادي والفيديو يتحقق إذا كانت قيمة شريط التنقل أكبر من طول الملف ناقص واحد

كما سنشرح في الفقرة التالية:

أذهب إلى تصميم البرنامج وقم باختيار كائن التنقل trackBar1

من نافذة الخصائص أختر خصائص الأحداث وانقر مزدوجاً أمام الحدث ValueChanged:



ستنتقل إلى شفرة البرنامج إلى إجراء TrackBar1ValueChanged هذا الإجراء يتم تنفيذه في كل مرة تتغير فيها قيمة شريط التنقل

حيث من المفترض إن نستدعي إجراء اختيار الملف التالي هنا :

كود

```
void TrackBar1ValueChanged(object sender, EventArgs e)
{
//استدعاء إجراء اختيار الملف التالي
}
```

أنسخ الكود التالي إلى داخل الإجراء **TrackBar1ValueChanged** :

كود

```
int last;
 if (type == "R")
 {
    last = trackBar1.Maximum - 1000;
 }
 else
 {
    last = trackBar1.Maximum - 1;
 }
 if (trackBar1.Value > last)
 {
    SetNext();
 }
```

حيث يصبح الإجراء **TrackBar1ValueChanged** هكذا :

كود

```
void TrackBar1ValueChanged(object sender, EventArgs e)
 {
    int last;
    if (type == "R")
    {
       last = trackBar1.Maximum - 1000;
    }
    else
    {
       last = trackBar1.Maximum - 1;
    }
    if (trackBar1.Value > last)
    {
       SetNext();
    }
 }
```

في هذا الإجراء عرفنا متغير أسمه last من نوع رقم يحمل قيمة تمثل أعلى قيمة لطول الملف الصوتي .وهي طول الملف الصوتي ناقص 1000 في ملفات الريل ,وهي طول الملف الصوتي ناقص واحد في ملفات الصوت العادية والفيديو

في السطر الرابع عبارة if لفحص نوع الملف الذي يتم تشغيله حالياً من خلال المتغير type الذي شرحناه في دروس سابقة .

فإذا كانت قيمة type هي R يعني هذا أنه ملف ريل , وبالتالي فإن أعلى قيمة لطول الملف الصوتي هي آخر ثانية في الملف .

وهي طول الملف ناقص 1000 , وقد وضعنا بدل طول الملف trackBar1.Maximum لأنها يحملان نفس القيمة بفعل الإجراء SetTrackBar

أما إذا لم يكن الملف ملف ريل فإن أعلى قيمة لطول الملف هي آخر ثانية وهي طول الملف ناقص واحد .في السطر الثاني عشر عبارة if أخرى تقوم بفحص قيمة شريط التنقل فإذا كانت قيمته أكبر من أعلى قيمة لطول الملف ,فهذا يعني أن البرنامج وصل إلى نهاية الملف الصوتي , ويجب استدعاء دالة اختيار الملف التالي وتشغيله ...

أذهب إلى إجراء plSelectedIndexChanged من الدرس السابق وغير شفرته حتى يصبح هكذا :

كود

```
void plSelectedIndexChanged(object sender, EventArgs e)
{
   current=pl.SelectedIndex;
   PlayCurrent();
}
```

عرفنا أن الإجراء plSelectedIndexChanged ينفذ في كل مرة تقوم باختيار ملف من قائمة التشغيل والكود السابق يقوم بتحميل المتغير current ترتيب الملف الذي اخترته ثم يستدعي إجراء تشغيله

الآن قم بتشغيل البرنامج وأختر عدة ملفات و جرب طرق التشغيل .

واجب الدرس التاسع

أرسل التطبيق

نهاية الدرس التاسع.

..

الدرس العاشر

المتغيرات العامة public

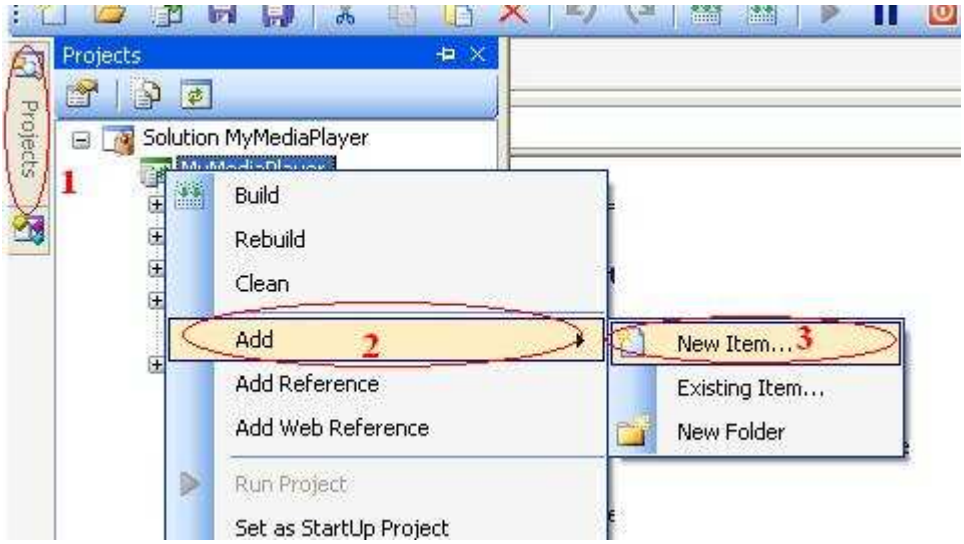
إلى الآن ما زال برنامجنا يحتوي على نافذة واحدة فقط , لذلك لم نجد الفرصة لاستخدام المتغيرات العامة .في درس اليوم سنقوم بإضافة نافذة جديدة إلى البرنامج لإدارة الألبومات "قوائم التشغيل" .
حيث سنحتاج للوصول إلى متغيرات النافذة الأساسية من نافذة الألبومات ,المتغير الأساسي الذي سنحتاج للوصول إليه من نافذة الألبومات هو قائمة التشغيل albume
حيث يجب تمريرها إلى نافذة الألبومات حتى يتم إجراء التعديلات عليها .

وهنا يجب استخدام كلمة public قبل تعريف albume حتى تستطيع النافذة الأخرى التعرف عليها كما سنرى في التطبيق.

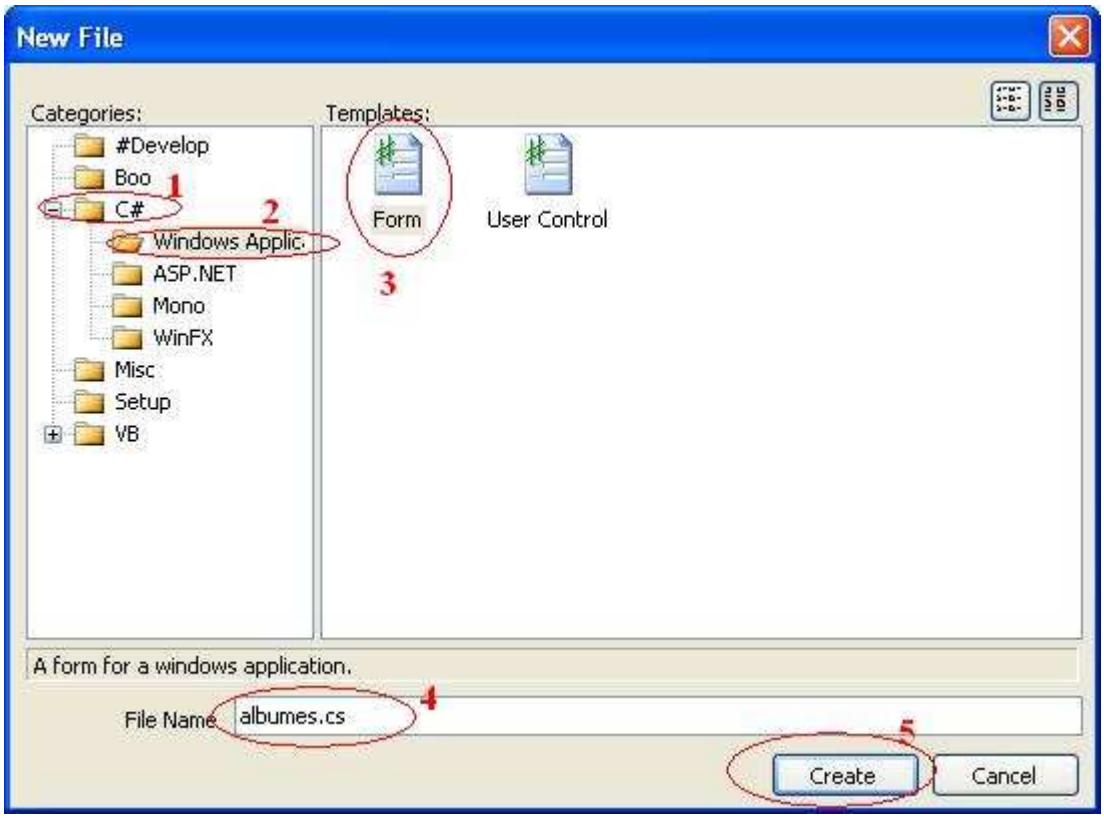
تطبيق الدرس العاشر:

أفتح تطبيق الدرس السابق وأذهب إلى نافذة Project وأنقر بزر الفارة الأيمن على أسم البرنامج MyMediaPlayer

ومن القائمة أختر Add ثم New Item

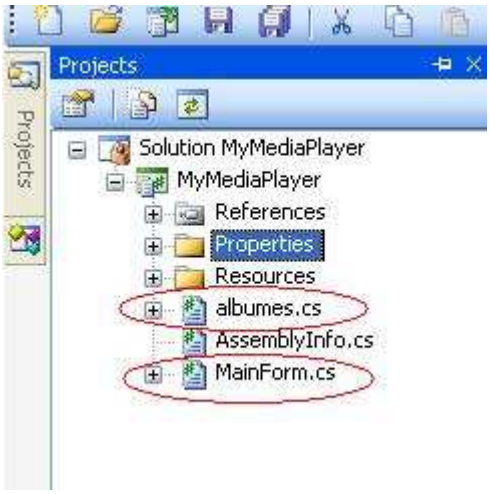


ستظهر لك القائمة التالية :

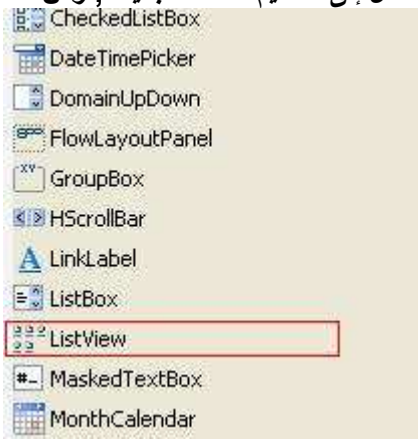


أختر C# ثم Windows Application ثم Form وأكتب في حقل أسم الفورم albumes.cs

ثم انقر على زر **Create** لكي يتم إنشاء نافذة جديدة في المشروع ,يتم التنقل عبر نوافذ البرنامج من خلال النقر المزدوج على ملف النافذة في نافذة **Projects**



أنتقل إلى تصميم النافذة الجديدة , ومن صندوق الأدوات قم بسحب كائن المستعرض **View List** إلى النافذة الجديدة .

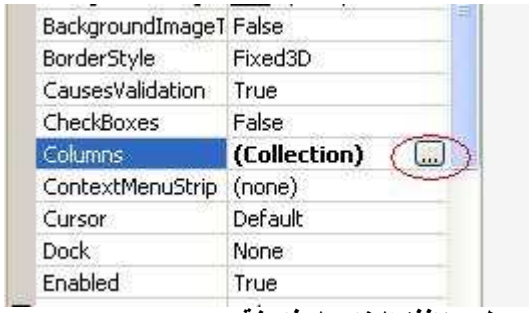


قم باختيار كائن المستعرض وأذهب إلى جدول الخصائص .

غير الخاصية **(Name)** إلى **lv**

وغير الخاصية **View** إلى **Details**

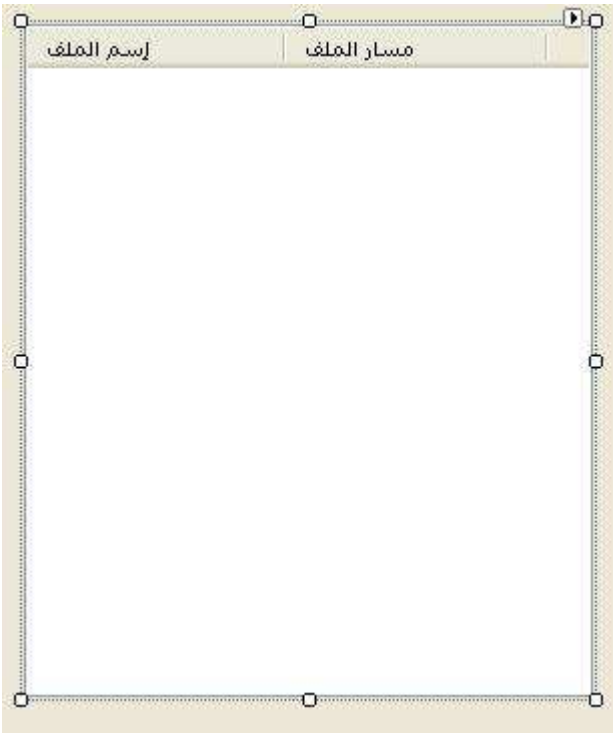
أذهب إلى خاصية **Columns** وأنقر على زر التفاصيل :



ستظهر نافذة التفاصيل فارغة .

هذه هي النافذة التي يتم من خلالها إنشاء أعمده لكائن المستعرض:

حيث سنكون عمودين , عمود لأسم الملف فقط , وعمود آخر لمسار الملف كاملاً .أنقر على زر Add وغير خاصية Text إلى "أسم الملف" .
ونقر مرة أخرى على زر Add لتنشئ العمود الثاني وغير خاصية Text فيه إلى "مسار الملف".
ثم انقر على زر الموافقة سيتكون في كائن المستعرض عمودين :



من صندوق الأدوات أسحب أربعة أزرار إلى يمين كائن المستعرض , وغير خاصية Text إلى:
(إضافة ,إلى الأعلى ,إلى الأسفل ,حذف).
أسحب زرين آخرين من صندوق الأدوات إلى أسفل كائن المستعرض وغير خاصية Text إلى:
(موافق ,إلغاء الأمر).

بحيث يصبح شكل النافذة هكذا :

أختر زر "موافق" وأذهب إلى جدول الخصائص وغير الخاصية DialogResult إلى OK
معنى هذا أنه عند النقر على هذا الزر فإن نتيجة النافذة هي الموافق كما سنشرح لاحقاً.
ثم أختر زر "إلغاء الأمر" وغير خاصية DialogResult إلى Cancel.
وهذا يعني أنه عند النقر على هذا الزر فإن نتيجة النافذة هي إلغاء الأمر.
من صندوق الأدوات أسحب كائن فتح الملفات و أختره وغير خاصية (Name) إلى OF
وغير خاصية MultiSelect إلى true
أنتقل إلى شفرة النافذة , نحتاج لتعريف متغير من نوع قائمة شبيهه بقائمة albume في النافذة الرئيسية ,حتى يتم من خلالها التعامل مع الملف وإضافتها إلى كائن المتصفح و حذفها منه ,وكما شرحنا سابقاً فإن كائن القائمة موجود في فضاء الأسماء System.Collections.Generic لذلك يجب إضافة عبارة using لتضمين كائنات ودوال الفضاء.

أذهب إلى شفرة النافذة وأضف عبارة using التالية تحت عبارات using الموجودة في رأس الصفحة :

كود

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

ثم أكتب كود تعريف القائمة تحت تعريف كائن النافذة :

كود

```
public List<string> albume=new List<string>();
```

```
namespace MyMediaPlayer
{
    /// <summary>
    /// Description of albumes .
    /// </summary>
    public partial class albumes  : Form
    {
        public List<string> albume=new List<string>();

        public albumes ()
        {
            //
```

لاحظ أنا استخدمنا الكلمة public وليس private وذلك حتى تتمكن النافذة الرئيسية من رؤية هذا المتغير كما سنرى لاحقاً .

القائمة albume في نافذة الألبومات ستكون نسخة مطابقة للقائمة albume في النافذة الرئيسية ,ويتم التعامل مع كائن المستعرض من خلال عناصر هذه القائمة .

إضافة ملفات

أنتقل إلى تصميم نافذة الألبومات وأنقر مزدوجا على زر "إضافة" للذهاب إلى شفرته ,أكتب الكود التالي بين الحاصرتين :

كود

```
OF.ShowDialog();
albume.AddRange(OF.FileNames);
FillView();
```

يقوم هذا الكود بفتح نافذة اختيار الملفات ثم إضافة الملفات المختارة إلى القائمة albume
في السطر الثالث نستدعي إجراء إضافة الملفات إلى كائن المستعرض .

الإجراء Fill View

يقوم هذا الإجراء بنسخ جميع عناصر القائمة albume إلى كائن المستعرض lv
أنسخ الإجراء التالي إلى شفرة نافذة الألبومات وتأكد من أنه خارج إلى إجراء أخر :

كود

```
void FillView()
{
    lv.Items.Clear();
    for (int i = 0; i < albume.Count; i++)
    {
        string filename=System.IO.Path.GetFileName(albume[i]);
        ListViewItem file = new ListViewItem(filename);
        file.SubItems.Add(albume[i]);
        lv.Items.Add(file);
    }
}
```

في السطر الثالث قمنا قمنا باستدعاء دالة تنظيف كائن المستعرض lv.Items.Clear

حيث تقوم هذه الدالة بتنظيف كائن المستعرض من أي عناصر موجودة مسبقاً حتى يتم إضافة العناصر الجديدة

في السطر الرابع دوارة for من الصفر وحتى عدد عناصر القائمة albume التي تحمل مسارات الملفات المختارة.

في السطر السادس قمنا بتعريف متغير من نوع نص يحمل ناتج دالة GetFileName والتي ترجع أسم الملف من المسار الموجود في قائمة albume في الترتيب I .

في السطر السابع عرفنا كائن جديد أسمه file من نوع ListViewItem

حيث إن كائن المستعرض lv هو عبارة عن مجموعة كائنات ListViewItem حيث يمثل كل كائن منها سطر واحد من سطور كائن المستعرض ,ولأن ListViewItem كائن وليس متغير فقد استخدمنا الجملة new ومررنا له أسم الملف الناتج من السطر السابق ,حيث أصبح المتغير يحمل قيمة filename من السطر السابق

وفي السطر الثامن , أضفنا كائن فرعي للكائن file من خلال الدالة file.SubItems.Add

ومررنا لها العنصر الذي ترتيبه i من قائمة albume.

الآن المتغير file أصبح يحمل قيمتين الأولى هي أسم الملف والثانية الفرعية هي مساره المخزن في قائمة albume.

في السطر التاسع أضفنا الكائن file إلى lv حيث يعتبر سطر واحد من سطور كائن المستعرض lv

وبعد تكرار هذه العملية لكافة عناصر القائمة albume يصبح الكائن lv محتوياً على جدول يوضح جميع أسماء ومسارات الملفات المختارة.

حذف ملف

أذهب إلى تصميم نافذة الألبومات وأنقر مزدوجاً على زر "حذف" لتنتقل إلى شفرته ,أكتب الكود التالي بين الحاصرتين :

كود

```
albume.RemoveAt(lv.SelectedIndices[0]);
FillView();
```

الدالة Remove At تقوم بأخذ ترتيب العنصر كمدخل وتحذفه من القائمة ,فمثلاً لو أردنا حذف العنصر الذي ترتيبه 4 سيكون الكود كالتالي :

كود

```
albume.RemoveAt(4);
```

لاحظ أن العنصر الذي ترتيبه 4 هو العنصر الخامس ,الدالة SelectedIndices من دوال الكائن lv تقوم بإرجاع مصفوفة تحتوي على ترتيبات الأسطر المختارة في الكائن.

فمثلاً لو اخترنا السطر الثالث و الرابع من القائمة سترجع الدالة مصفوفة تحتوي رقمي 2 و 3

لاحظ , أن السطر الثالثة ترتيبه 2 وليس 3 كما شرحنا مراراً.

لكنا نريد عند النقر على زر الحذف أن يقوم البرنامج بحذف سطر واحد, وهو أول سطر من السطور المختارة

كود

```
lv.SelectedIndices[0]
```

وفي السطر الثاني نقوم باستدعاء الإجراء FillView والذي يقوم بنسخ محتويات القائمةalbumeإلى الكائنlv

ماذا لو تم النقر على زر الحذف وليس هناك أسطر مختارة من قبل المستخدم ,سيظهر خطأ عندها , ولحل هذا الخطأ يجب أولاً فحص عدد العناصر المختارة فإذا كان أكبر من صفر ,فذلك يعني أن هناك ملفات مختارة و بالتالي ينفذ كود الحذف .

عدل الكود السابق حتى يصبح هكذا :

كود

```
if(lv.SelectedIndices.Count>0)
 {
 albume.RemoveAt(lv.SelectedIndices[0]);
 FillView();
 }
```

يتم فحص عدد العناصر المختارة من خلال الخاصية lv.SelectedIndices.Count

نقل ملف إلى الأعلى

وظيفة هذا الزر هو إعادة ترتيب الملفات بحيث ينقل الملف المختار في كائن المستعرض إلى أعلى خطوة واحدة

أذهب إلى تصميم نافذة الألبومات و أنقر مزدوجاً على زر "إلى الأعلى".

وأكتب الكود التالي بين الحاصرتين :

كود

```
int sel = lv.SelectedIndices[0];
 string tmp = albume[sel];
 albume[sel] = albume[sel-1];
 albume[sel - 1] = tmp;
 FillView();
 lv.Items[sel-1].Selected = true;
```

في السطر الأول عرفنا متغير من نوع رقم يقوم بخزن ترتيب السطر المختار في كائن المستعرض lv

وفي السطر الثاني عرفنا متغير آخر من نوع نص يقوم بخزن قيمة العنصر الذي ترتيبه هو ترتيب السطر المختار

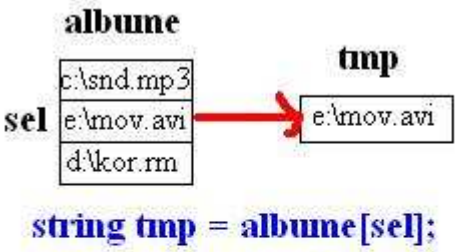
وفي السطر الثالث قمنا بتحميل العنصر الذي ترتيبه هو ترتيب السطر المختار بقيمة العنصر الذي قبله.

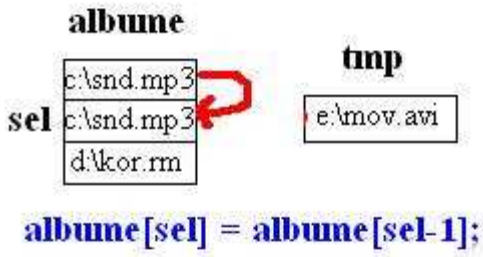
وفي السطر الرابع قمنا بتحميل السطر السابق للسطر المختار بالقيمة التي كانت مخزنة في السطر المختار

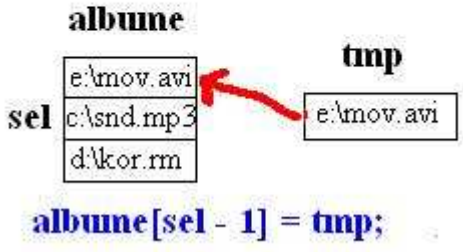
وفي السطر الخامس قمنا باستدعاء إجراء تعبئة الكائن lv

وفي السطر السادس قمنا بتحديد واختيار السطر الحالي الذي تم نقله

لتوضيح الكود السابق لاحظ هذا الشكل التوضيحي :







حيث افترضنا أن القائمة albume تحتوي على ثلاثة عناصر تم اختيار العنصر الأوسط لنقله خطوة إلى الأعلى

ماذا لو أختار المستخدم أول عنصر سطر في المستعرض , كيف سيتم نقل أول عنصر في القائمة خطوة إلى أعلى سيحدث خطأ إذا حاول المستخدم فعل ذلك ولتفادي هذا الخطأ يجب أن نفحص ترتيب السطر المختار sel فإذا كان يساوي صفر.

فمعناه أن المستخدم أختار أول عنصر في القائمة , ولا يجب تنفيذ كود النقل , إما إذا كانت قيمته أكبر من الصفر فلا مانع من تنفيذ كود النقل .

عدل الكود السابق ليصبح هكذا :

كود

```
int sel = lv.SelectedIndices[0];
 if(sel>0)
 {
 string tmp = albume[sel];
 albume[sel] = albume[sel-1];
 albume[sel - 1] = tmp;
 FillView();
 lv.Items[sel-1].Selected = true;
 }
```

ماذا لو لم يختر المستخدم أي سطر وقام بالنقر على زر النقل .. سيحدث هنا أيضاً خطأ , ويجب معالجته كما فعلنا في زر الحذف.

عدل الكود السابق ليصبح هكذا :

كود

```
if(lv.SelectedIndices.Count>0)
 {
 int sel = lv.SelectedIndices[0];
 if(sel>0)
 {
 string tmp = albume[sel];
 albume[sel] = albume[sel-1];
 albume[sel - 1] = tmp;
 FillView();
 lv.Items[sel-1].Selected = true;
 }
 }
```

نقل ملف إلى أسفل

يشبه كود نقل الملف لأعلى باختلاف بسيط جداً فبدل من sel-1 نضع sel+1

وعند المقارنة sel>0 نضع lv.Items.Count>sel لأن النقل في هذه الحالة سيكون للأسفل

ويجب أن نفحص ما إذا كان ترتيب السطر المختار أقل من عدد سطور الكائن :

أفتح تصميم نافذة الألبومات وأنقر مزدوجاً على زر "إلى الأسفل" وأنسخ الكود التالي بين الحاصرتين:

كود

```
if(lv.SelectedIndices.Count>0)
{
int sel = lv.SelectedIndices[0];
if(sel<lv.Items.Count-1)
{
string tmp = albume[sel];
albume[sel] = albume[sel+1];
albume[sel + 1] = tmp;
FillView();
lv.Items[sel+1].Selected = true;
}
}
```

بهذا نكون قد انتهينا من نافذة الألبومات ...

أنتقل إلى النافذة الرئيسية و من صندوق الأدوات أسحب كائن الزر إلى النافذة الرئيسية وغير النص إلى "التحكم بالألبومات"

اضبط ترتيب الأدوات في النافذة حتى يصبح شكلها هكذا :

إظهار نافذة الألبومات

أذهب إلى تصميم النافذة الرئيسية وأنقر مزدوجاً على زر "التحكم بالألبومات" لتنتقل إلى شفرته وانسخ الكود التالي بين الحاصرتين:

كود

```
albumes frm = new albumes();
 frm.albume = albume;
 frm.ShowDialog();
 albume = frm.albume;
 FillList();
```

في السطر الأول عرفنا كائن أسمه frm من نوع albumes , لاحظ أن albumes هو أسم نافذة الألبومات إي أنها الآن تعتبر كائن في النظام ويمكن تعريف متغيرات من نوعها كما كنا نفعل مع القوائم أو الكائنات الأخرى.

في السطر الثاني نقوم بتحميل القائمة albume من النافذة frm بقيم القائمة albume من النافذة الرئيسية أي أن القائمة albume في النافذة الجديدة frm ستصبح نسخة مطابقة لقائمة albume الموجودة في النافذة الرئيسية.

في السطر الثالث قمنا باستدعاء الإجراء ShowDialog والذي يظهر النافذة frm للمستخدم .

السطر الرابع يتم تنفيذه عند إغلاق النافذة frm , حيث يتم فيه نسخ قيمة albume من نافذة الألبومات لقائمة albume الموجودة في النافذة الرئيسية.

أي إن الكود السابق يقوم بإنشاء نافذة جديدة وينسخ لها قيم albume حتى يعدل المستخدم فيها ويغلق النافذة ثم يأخذ قيم albume من النافذة frm والتي تم تعديلها من قبل المستخدم وينسخ قيمها لقائمةalbume الموجوده في الصفحة الرئيسية.

وفي السطر الخامس نستدعي إجراء FillList الذي يقوم بنسخ قيم albume إلى قائمة التشغيل كما شرحنا في الدروس السابقة.

إذا شغلت البرنامج الآن واخترت "التحكم بالألبومات" فلن تظهر لك عناصر قائمة albume لأنك لم تقم باستدعاء إجراء تعبئة المستعرض FillView.

حيث يجب استدعاءه بمجر تحميل نافذة الألبومات.

أذهب إلى تصميم نافذة الألبومات وانقر مزدوجا على مكان خالي من الأدوات في النافذة ستنتقل إلى إجراء حدث **Load** :

كود

```
void AlbumesLoad(object sender, System.EventArgs e)
{

}
```

أكتب استدعاء إجراء تعبئة المستعرض بين الحاصرتين حتى يصبح الإجراء هكذا :

كود

```
void AlbumesLoad(object sender, System.EventArgs e)
{
    FillView();
}
```

شغل البرنامج ومن النافذة الرئيسية أختر مجموعة ملفات ثم أختر "التحكم بالألبومات" وجرب كل الوظائف.

واجب الدرس العاشر

أرسل التطبيق

نهاية الدرس العاشر.

..

الدرس الحادي عشر

دوارة While

في الدرس الرابع تعرفنا على هيكل من هياكل التكرار في لغة C # وهو دوارة for ,والحقيقة أن هناك هياكل أخرى من هياكل التكرار أهمها هي دوارة While.

وهي تشبه إلى حد كبير دوارة for باختلاف أنها لا تحتوي على عداد Counter كما كنا نعرف i في دوارة for

والهيكل الأساسي لدوارة While يشبه إلى حد كبير هيكل for باختلاف أشياء بسيطة :

كود

```
while (شرط التكرار)
{
كود الدوارة
}
```

تقوم جملة while بتكرار كود الدوارة ما دام شرط التكرار متحققاً , مثلاً :

كود

```
int n=0;
while (n<5)
{
MessageBox.Show("من داخل الدوارة ");
n++;
}
```

الدوارة السابقة تقوم بإظهار الرسالة خمس مرات

حيث عرفنا متغير من نوع رقم n وأسندنا له قيمة ابتدائية 0

وشرط التكرار في الدوارة هو أن تكون قيمة المتغير n أصغر من 5

في المرة الأولى ستكون قيمة n هي صفر , وبالتالي فشرط التكرار متحقق , لأن الصفر أقل من 5

يتم تنفيذ كود إظهار الرسالة , والكود n++ يعني إضافة واحد إلى قيمة n أي أن n سيحمل الآن القيمة 1.يعود المعالج لبداية جملة while ويختبر شرط التكرار مرة أخرى ثم ينفذ كود إظهار الرسالة ويزيد واحد لقيمة n.

وهكذا حتى تصبح قيمة n=5 عندها لن يتحقق شرط التكرار لأن 5 ليست أقل من 5 ,وبالتالي يقفز المعالج إلى نهاية كود الدوارة ويواصل تنفيذ التعليمات بعدها, يستخدم هذا النوع من الدوارات حين لا نعرف كم بالضبط سنكرر كود الدوارة.

وسنستخدمها في درس اليوم حيث لدينا معلومات الألبوم ولا نعرف عدد الملفات في الألبوم كما سنشرح بالتفصيل لا حقا.

ملفات XML

من أهم أنظمة الحاسوب التي استفادة منها الإنسانية منذ ظهور الحاسوب هي أنظمة قواعد البيانات وأنظمة قواعد البيانات هي برمجيات تستخدم مع لغات البرمجة بغرض خزن واسترجاع أنواع متعددة من البيانات ,ويتم خزن هذه البيانات بطرق مرتبة ومنهجية بحيث يسهل استرجاع بيانات محددة مهما كبير كان حجم البيانات.

حيث يتم ترتيب البيانات في وحدات متجانسة تسمى جداول Tables والتي تحتوي على حقول Fields لكل حقل نوع محدد من البيانات. ويمكن للجدول الواحد أن يحتوي على عدة حقول من أنواع بيانات مختلفة,قد ترتبط الجداول مع بعضها بواسطة علاقات Relationships حيث تعتمد بيانات من جدول على بيانات من جدول آخر

أنظمة قواعد البيانات تقسم إلى قسمين , نظام الخزن , ونظام المعالجة.

أما نظام الخزن فهو المسئول عن خزن البيانات في ملفاتها المحددة بغض النظر عن ترتيبها أو أماكنها ,ونظام المعالجة هو المسئول عن تحديد أماكن البيانات الصحيحة و استرجاعها والقيام بالعمليات عليها .

النظام الأول -نظام الخزن- يتكفل به نظام التشغيل ولا يحتاج لبرمجيات أو برامج خاصة

أما النظام الثاني فيتطلب برمجيات محددة لنظام قواعد البيانات.

معظم أنظمة قواعد البيانات تستخدم النظام الثاني , مثل قواعد الأكسس و لأوراكل و غيرها.

هناك أنواع أخرى لا تتطلب برمجيات مخصصة للتعامل معها ومنها نظام قواعد البيانات عبر ملفات XML

وهذا هو محور درسنا اليوم , حيث لا يحتاج هذا النظام لبرمجيات متخصصة للتعامل معه .وإنما ملفاته عبارة عن ملفات نصية يمكن فتحها وتعديلها بأي محرر نصوص,وسنستخدم ملفات XML في تطبيقنا لخزن بيانات ,الألبومات أو قوائم التشغيل,صيغة هذه الملفات النصية تشبه إلى حد كبير صيغة ملفات HTML

حيث يعتمد على وسوم Tags ترتب البيانات في هياكل مترابطة.

وكل جزء من البيانات يحتوي على وسم بداية ووسم نهاية وله أسم محدد.

فمثلاً هذا الكود :

كود

```
<msg>رسالة</msg>
```

هذا جزء من كود ملف XML حيث يحتوي على جزء لخزن البيانات أسمه msg ,يحتوي هذا الجزء على بيانات مخزنة هو كلمة "رسالة" لاحظ أن الجزء يحتوي على وسمين ,وسم البداية حيث يحتوي على أسم الجزء بين علامتي أصغر وأكبر من.

ووسم النهاية وهو مشابه تماما لوسم البداية بزيادة الشرطة المائلة / قبل أسم الجزء في وسم النهاية.

وبين الموسمين توجد البيانات.

وتسمى الأجزاء في ملفات XML بالعقد Nodes فالكود السابق يحتوي على عقدة أسمها msg

سميت بالعقد لأنه يمكن لأي جزء من البيانات أن يحتوي على أجزاء أخرى :

كود

```
<albumes>
   <albume1>

   </albume1>
</albumes>
```

الكود السابق يحتوي على عقدة رئيسية أسمها albumes تحتوي بدورها على عقدة فرعية أسمها albume1

يمكن أن تحتوي العقدة على عدة عقد فرعية :

كود

```
<albumes>

   <albume1>
   </albume1>

   <albume2>
   </albume2>

</albumes>
```

لعقدة الرئيسية albumes تحتوي على عقدتين فرعيتين albume1 و albume2

يمكن أيضاً للعقدة الفرعية أن تحتوي على عقد فرعية أخرى:

كود

```
<albumes>

   <albume1>
    <file>sound.mp3</file>
    <file>wave.rm</file>
   </albume1>

   <albume2>
    <file>real.wav</file>
   </albume2>

</albumes>
```

العقدة الرئيسية albumes تحتوي على عقدتين فرعيتين albume1 و albume2

العقدة الفرعية albume1 تحتوي على عقدتين فرعيتين بنفس الاسم file

كل عقدة من عقدتي file بيانات نصية sound.mp3 و wave.rm

والعقدة الفرعية albume2 تحتوي على عقدة فرعية file

الكود السابق يمثل ملف XML يحتوي على بيانات مرتبة في عقد , ولكن ينقصه شيء مهم,نوع التكويد في الملف , ورقم إصدار كود XML , والتكويد هو أسلوب خزن البيانات النصية في الملف.

عادة ما نستخدم تكويد UTF-8 , وسطر نوع التكويد يجب أن يكون في بداية الملف:

كود

```
<?xml version="1.0" encoding="utf-8"?>
<albumes>
   <albume1>
    <file>sound.mp3</file>
    <file>wave.rm</file>
   </albume1>
   <albume2>
    <file>real.wav</file>
   </albume2>
</albumes>
```

هذا السطر لا يتغير في معظم ملفات XML هذا هو هيكل الملف الذي سيقوم بخزن بيانات الألبومات وقوائم التشغيل.

يتم التعامل مع ملف XML في البرنامج من خلال كائن XmlDocument الموجود في فضاء الأسماء System.XML:

كود

```
XmlDocument doc = new XmlDocument();
```

الكود السابق عرفنا من خلاله كائن أسمه doc من نوع XmlDocument

الكائن doc يقوم بتحميل الملف عن طريق الإجراء Load

كود

```
doc.Load("c://albumes.xml");
```

الكود السابق يقوم بتحميل الملف albumes.xml الموجود في القرص c إلى الكائن doc

يتم الوصول إلى العقد الموجودة في الملف عن طريق ذكر أسم العقدة الرئيسية متبوعة بقوسين مربعين يحتويان أسم العقدة الفرعية.

أما إذا أردنا الوصول إلى العقد الرئيسية ألمسماه albumes فنكتب أسم الكائن doc متبوعاً بقوسين مربعين يحتويان إسم العقدة :

كود

```
doc["albumes"];
```

ويتم الوصول إلى البيانات في العقدة بواسطة الخاصية InnerText :

كود

```
MessageBox.Show(doc["albumes"].InnerText);
```

الكود السابق يقوم بإظهار رسالة تحتوي على البيانات المخزنة في العقدة albumes

ويتم الوصول لأسم العقدة عن طريق الخاصية **Name:**

كود

```
MessageBox.Show(doc["albumes"].Name);
```

الكود السابق يظهر رسالة تحتوي على أسم العقدة **albumes**

ويتم الوصول إلى العقد الفرعية بإضافة قوسين مربعين يحتويان أسم العقدة. فمثلاً للوصول إلى العقدة الفرعية **albume1** الموجود في العقدة **albumes** :

كود

```
doc["albumes"]["albume1"]
```

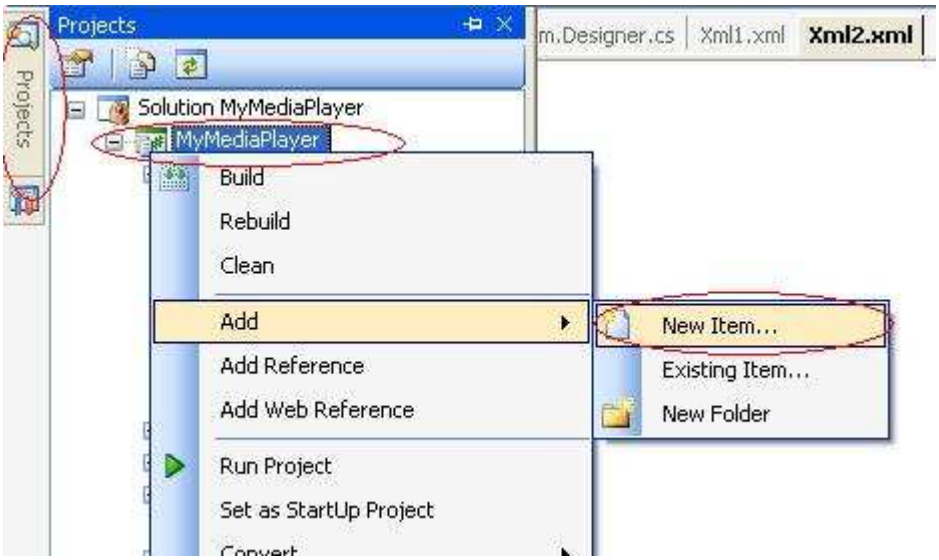
وللوصول إلى العقدة الأعمق **file** :

كود

```
doc["albumes"]["albume1"]["file"]
```

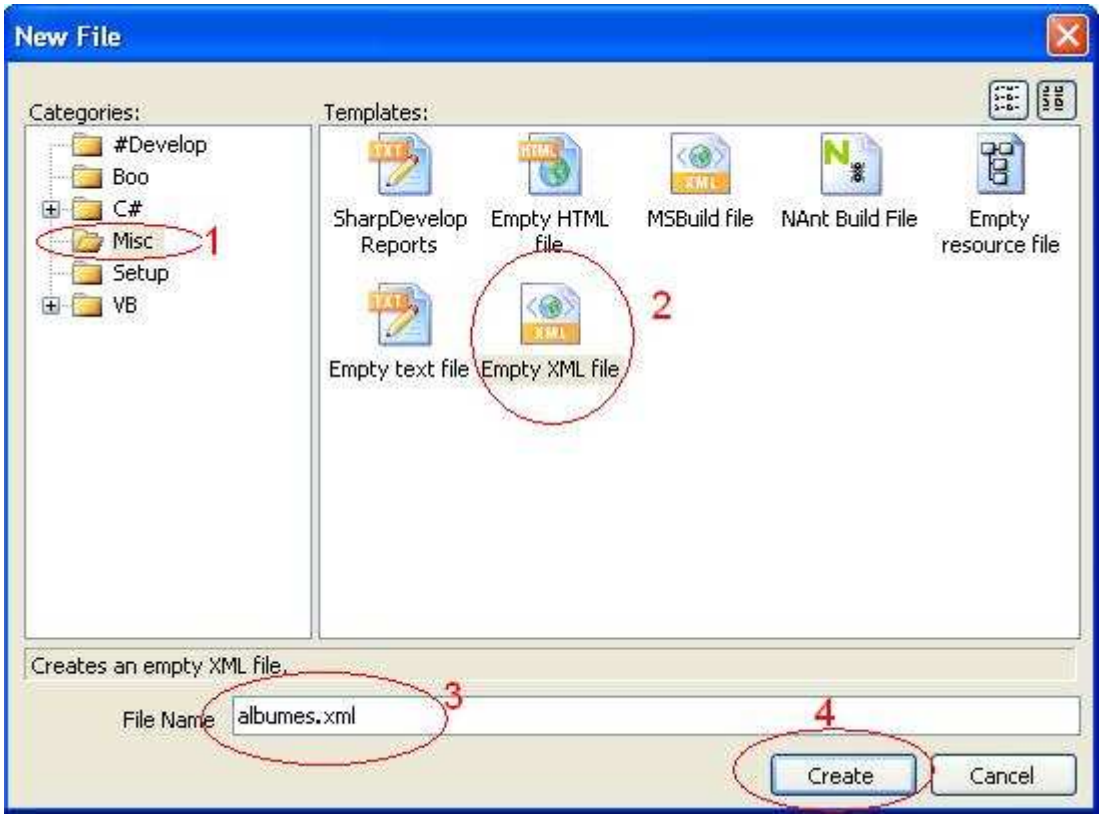
تطبيق الدرس الحادي عشر:

افتح تطبيق الدرس السابق , من نافذة مستعرض المشروع, أنقر بالزر الأيمن على أسم المشروع ,ومن القائمة الناتجة أختر Add ثم New Item :



ستظهر نافذة إضافة ملف جديدة للمشروع , من القائمة في اليسار أختر misc

ثم أختر Empty XML file , وفي صندوق أسم الملف أكتب الاسم التالي albumes.xml , وأختر OK



أذهب إلى مجلد المشروع وأنسخ الملف albumes.xml إلى مجلد Debug الموجود داخل مجلد bin

حيث يصبح الملف موجود بجانب ملفات المشروع الأخرى.

الآن قم بفتح تصميم نافذة الألبومات وسحب ثلاثة أزرار إلى النافذة وغير الأسماء فيها إلى:

ألبوم جديد

حفظ الألبوم

حذف الألبوم

ومن صندوق الأدوات أسحب كائن القائمة المنسدلة إلى نافذة الألبومات :

أختر كائن القائمة المنسدلة و من جدول الخصائص , غير الخاصية (Name) , إلى ac

وغير الخاصية Text إلى "أختر الألبوم"

قم بترتيب الأدوات في النافذة حتى تصبح هكذا :

جميع الدوال التي سنستخدمها للتعامل مع ملف XML موجودة في فضاء الأسماء XML الموجود داخل الفضاء System.

لذلك يجب إضافة عبارة using لتعريف جميع دوال وكائنات التعامل مع ملفات XML

أذهب إلى شفرة نافذة الألبومات و أكتب السطر التالي , تحت جمل using في رأس الملف:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml;

namespace MyMediaPlayer
{
```

يتم تحميل ملف XML إلى البرنامج والتعامل معه من خلال كائن XmlDocument .حيث سنعرف كائن عام من هذا النوع ليتم التعامل معه في جميع أنحاء نافذة الألبومات.

إنسخ كود التعريف تحت تعريف قائمة albume من الدرس السابق :

```
public partial class albumes : Form
{
    public List<string> albume=new List<string>();
    private XmlDocument doc = new XmlDocument();

    public albumes()
    {
        InitializeComponent();
```

قمنا هنا بتعريف كائن أسمه doc من نوع XmlDocument , ولأنه كائن استخدمنا الكلمة new

إجراء تحميل الألبومات للقائمة المنسدلة Fill Combo

يقوم هذا الإجراء بتحميل أسماء الألبومات الموجودة في ملف XML إلى القائمة المنسدلة ac

قم بنسخ الكود التالي إلى شفرة نافذة الألبومات وتأكد أنه خارج أي إجراء آخر :

كود

```
void FillCombo()
{
   doc.Load(Application.StartupPath + "//albumes.xml");
   XmlNode alnd = doc["albumes"].FirstChild;
   while (alnd != null)
   {
      ac.Items.Add(alnd.Name);
      alnd = alnd.NextSibling;
   }
}
```

السطر الأول يقوم بتحميل الملف إلى الكائن doc ,والتعليمة Application.StartupPath تعيد قيمة نصية تمثل مسار ملف exe للتطبيق ,حيث استفدنا من هذه التعليمة لمعرفة مسار ملف XML وأضفنا أسم الملف للمسار .في السطر الثاني قمنا بتعريف كائن من نوع عقدة XmlNode أسمه alnd

ثم حملنا هذا الكائن أول عقدة فرعية داخل العقدة الرئيسية albumes من خلال الخاصية FirstChild

السطر الثالث يمثل جملة while تنفذ كود الدوارة ما دامت العقدة alnd تحمل قيمة

الشرط alnd!=null يختبر الكائن alnd فإذا كان يحمل قيمة غير القيمة الفارغة null فسيتم تنفيذ كود الدوارة.

أما إذا كان الكائن alnd لا يحمل قيمة أي أنه يحمل null فعندها لن يتم تنفيذ كود الدوارة .السطر الخامس يقوم بإضافة إسم العقدة alnd إلى القائمة المنسدلة ac .

السطر السادس يقوم باختيار العقدة التالية للعقدة alnd حيث يحمل الكائن alnd العقد التالية لما كان يحمله

يتم تكرار السطرين السابقين حتى الوصول إلى آخر عقدة عندها سيحمل الكائن alnd القيمة null لأنه لا يوجد عقدة تالية للعقدة الأخيرة. وبالتالي ينتهي تكرار جملة while

لكن ماذا لو لم يكن هناك إي عقد في ملف XML , عندها سينتج خطأ من تنفيذ هذا الإجراء ولحل ذلك نستخدم عبارة try:

قم بتعديل الكود السابق حتى يصبح هكذا :

كود

```
void FillCombo()
{
    try
    {
        doc.Load(Application.StartupPath + "//albumes.xml");
        XmlNode alnd = doc["albumes"].FirstChild;
        while (alnd != null)
        {
        ac.Items.Add(alnd.Name);
        alnd = alnd.NextSibling;
        }
    }
    catch
    {
        MessageBox.Show("حدث خطأ أثناء استرجاع بيانات الألبومات ");
    }
}
```

بعد شرح إجراء **FillCombo** بقي أن نحدد مكانا مناسباً لإستدعاءه , وأفضل مكان لذلك هو عند تحميل النافذة

إذا إلى تصميم نافذة الألبومات وانقر مزدوجاً على مكان فارغ في النافذة لتذهب إلى كود تحميل النافذة

albumes_Load

أضف إستدعاء إجراء **FillCombo** تحت إجراء **FillView** من الدرس السابق , حيث يصبح كود تحميل النافذة هكذا :

كود

```
void AlbumesLoad(object sender, System.EventArgs e)
{
    FillView();
    FillCombo();
}
```

إجراء قراءة محتويات الألبوم ReadAlbume

سيقوم هذا الإجراء بقراءة محتويات ألبوم محدد من ملف XML وتخزينها في قائمة albume

ثم استدعاء إجراء FillView لنسخ محتويات القائمة إلى كائن المستعرض lv

يحتوي هذا الإجراء على مدخل من نوع نص يمثل الألبوم المختار المراد تحميل ملفاته إلى كائن المستعرض

أنسخ الكود التالي إلى شفرة نافذة الألبومات وتأكد أنه خارج أي إجراء آخر:

كود

```
void ReadAlbume(string albumename)
{
    try
    {
    doc.Load(Application.StartupPath + "//albumes.xml");
    XmlNode filend = doc["albumes"][albumename].FirstChild;
    albume.Clear();
    while (filend != null)
    {
        albume.Add(filend.InnerText);
        filend = filend.NextSibling;
    }
    FillView();
    }
    catch
    {
        MessageBox.Show("حدث خطأ أثناء استرجاع بيانات الألبومات ");
    }
}
```

السطر الأول من الكود يحتوي على أسم الإجراء وأسم المدخل ونوعه string

السطر الخامس لتحميل ملف XML إلى الكائن doc

السطر السادس عرفنا كائن من نوع عقدة يحمل أول عقدة من العقدة الفرعية التي إسمها هو المدخل والموجوده داخل العقدة الرئيسية albumes

لاحظ أنا وضعنا إسم المدخل في العقدة الفرعية فعند تشغيل البرنامج سيتم إستبدالة بقيمة المدخل كما سنرى لاحقاً

السطر السابق يقوم بتحديد عقدة الألبوم المدخل , مثلاً albume1 ثم أخذ أول عقدة منه file وتحميلها على الكائن filend

السطر السابع قمنا بتنظيف القائمة albume من أي عناصر موجودة فيها حتى ندخل إليها العناصر الجديدة من الملف.

جملة while في السطر الثامن , تنفذ كود الدوارة ما دام filend لا يحمل القيمة null

السطر العاشر نقوم بإضافة بيانات العقدة إلى القائمة عن طريق الخاصية InnerText في العقدة

في السطر الحادي عشر نقوم باختيار وتحميل العقدة التالية للعقدة الحالية , وهكذا حتى نصل لآخر عقدة ويتوقف تنفيذ كود الدوارة.

في السطر الثالث عشر نقوم باستدعاء إجراء نسخ القائمة albume إلى كائن المستعرض lv

بعد شرح هذا الإجراء بقي أن نختار المكان المناسب لإستدعائة. وأفضل مكان لذلك هو عندما يختار المستخدم أسم الألبوم من القائمة المنسدلة .

إذهب إلى تصميم نافذة الألبومات وأنقر مزدوجاً على كائن القائمة المنسدلة

ستذهب إلى الشفرة , أكتب هناك استدعاء الأجراء ReadAlbume

لا تنسى أن هذا الإجراء يتطلب مدخل من نوع نص يمثل الألبوم المختار.

والألبوم المختار في هذه الحالة هو النص المكتوب على كائن القائمة المنسدلة Text

لذلك فاستدعاء الإجراء سيكون هكذا :

كود

```
ReadAlbume(ac.Text);
```

ويصبح كود الإجراء كاملاً - اختيار أسم الألبوم من القائمة - هكذا :

كود

```
void AcSelectedIndexChanged(object sender, System.EventArgs e)
{
   ReadAlbume(ac.Text);
}
```

حيث أنه عند تشغيل البرنامج واستدعاء الإجراء سيتم تغيير albumename في الإجراء إلى قيمة ac.Text

إجراء حذف الألبوم DeleteAlbume

يقوم هذا الإجراء بحذف بيانات الألبوم المختار من ملف XML

حيث يتطلب مدخل واحد من نوع نص , يمثل أسم الألبوم المراد حذفه

انسخ الكود التالي إلى شفرة نافذة الألبومات وتأكد من أنه خارج أي إجراء آخر :

كود

```
void DeleteAlbume(string albumename)
{
    try
    {
        if (MessageBox.Show("هل أنت متأكد من حذف / تعديل الألبوم" ,"؟",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button2) == DialogResult.Yes)
        {
            doc.Load(Application.StartupPath + "//albumes.xml");
            doc["albumes"].RemoveChild(doc["albumes"][albumename]);
            doc.Save(Application.StartupPath + "//albumes.xml");
            ac.Items.RemoveAt(ac.SelectedIndex);
            lv.Items.Clear();
        }
    }
    catch
    {
    }
}
```

السطر الأول يحتوي على أسم الإجراء وأسم المدخل ونوعه string

السطر الخامس جملة if تقوم بإظهار رسالة تحذير للمستخدم بحيث يتم تأكيد الحذف , لاحظ عبارة استدعاء الدالة:

كود

```
MessageBox.Show("هل أنت متأكد من حذف / تعديل الألبوم ؟", "هل أنت متأكد", MessageBoxButtons.YesNo,
MessageBoxIcon.Question, MessageBoxDefaultButton.Button2)
```

العبارة إختلافة عن ما كنا نعرفه من قبل , ورغم أنها نفس التعليمة إلا أنها هذه المرة استقبلت مدخلات أكثر لذلك وظيفتها هذه المرة مختلفة قليلاً , المدخل الأول هو نص الرسالة كما عرفنا سابقاً,المدخل الثاني هو عنوان الرسالة الذي سيظهر في شريط العنوان.

المدخل الثالث MessageBoxButtons.YesNo هو الأزرار المطلوب إظهارها في الرسالة حيث اخترنا YesNo وهناك أيضاً OkCancel و هناك مجموعات أخرى من الأزرار.

المدخل الرابع MessageBoxIcon.Question يمثل الأيقونه المراد إظهارها في الرسالة حيث اخترنا أيقونة السؤال Question.

المدخل الخامس MessageBoxDefaultButton.Button2 يمثل الزر الافتراضي , أي الزر الذي يكون التركيز عليه عند ظهور الرسالة.

وقد اخترنا Button2 والذي سيكون في هذه الحالة زر No حيث سيكون الجواب الافتراضي للرسالة هو No

وقد اخترنا No بدلا من Yes حتى يتأكد المستخدم من إختياره قبل أن نحذف بيانات الألبوم .

وإذا كانت نتيجة هذه الرسالة هي الموافقة DialogResult.Yes سيتم تنفيذ كود الحذف وإلا فلا

في السطر السابع تعليمة تحميل الملف لكائن doc

وفي السطر الثامن تعلمية إزالة عقدة الألبوم , حيث يتم إزالة عقدة بتحديد عقدتها الرئيسية ثم إستدعاء الإجراء RemoveChild

والعقدة الرئيسية لعقد الألبومات هي albumes لذلك تم إستدعاء إجراء الحذف منها داخل قوسي إجراء الحذف العقدة المراد حذفها , وهي العقدة الذي أسمها هو المدخل albumename

الموجودة داخل العقدة الرئيسية albumes

في السطر التاسع تعليمة حفظ التغييرات على ملف XML باستدعاء الإجراء Save من كائن doc

في السطر العاشر تعليمة حذف أسم الألبوم المحذوف من القائمة المنسدلة .

وفي السطر الحادي عشر تعليمة تنظيف كائن المستعرض lv من ملفات الألبوم المحذوف .المكان المناسب لاستدعاء هذا الإجراء هو كود الزر "حذف الألبوم"

أذهب إلى تصميم نافذة الألبومات وانقر مزدوجاً على زر "حذف الألبوم" وأكتب استدعاء إجراء الحذف هناك :

كود

```
DeleteAlbume(ac.Text);
```

إجراء حفظ الألبوم WriteAlbume

هذا الإجراء سيقوم بتخزين محتويات كائن المستعرض lv إلى ملف XML

حيث يقوم الإجراء أولاً بإنشاء عقدة الألبوم , وتسميتها بالنص المكتوب على القائمة المنسدلة

ثم يقوم بإنشاء العقد الفرعية file والتي تحتوي على مسارات الملفات الموجودة في كائن المستعرض

وأيضاً يقوم الإجراء بحفظ التغيرات على ألبوم موجود أساساً , بحيث يقوم بحذفه أولاً من الملف ثم تخزين ألبوم جديد بنفس الاسم وبالبيانات المعدلة

أنسخ الكود التالي إلى شفرة نافذة الألبومات وتأكد من أنه خراج إلى إجراء آخر :

كود

```
void WriteAlbume(string albumename)
{
    try
    {
    doc.Load(Application.StartupPath + "//albumes.xml");
    if (doc["albumes"][albumename]!=null)
    {
        doc["albumes"].RemoveChild(doc["albumes"][albumename]);
    }
    else
    {
        ac.Items.Add(ac.Text);
    }
    XmlNode albumend = doc.CreateElement(albumename);
    for (int i = 0; i < albume.Count; i++)
    {
        XmlNode filend = doc.CreateElement("file");
        XmlNode path = doc.CreateTextNode(albume[i]);
        filend.AppendChild(path);
        albumend.AppendChild(filend);
    }
    doc["albumes"].AppendChild(albumend);
    doc.Save(Application.StartupPath + "//albumes.xml");
    }
    catch
```

```
        {
            MessageBox.Show("حدث خطأ أثناء محاولة حفظ بيانات الألبومات");
        }
    }
```

السطر الأول يحتوي على إسم الإجراء وإسم المدخل و نوعة string

السطر الخامس تعليمة تحميل الملف إلى كائن doc

السطر السادس عبارة if تختبر هل هناك عقدة للألبوم المحدد في ملف XML

فإذا كانت العقدة موجودة فالسطر الثامن يقوم بحذفها , حتى توضع بدلاً منها العقدة بالبيانات المعدلة والحذف هنا هو للبيانات القديمة وخزن البيانات الجديدة كأنها ألبوم جديد بنفس الاسم .أما إذا لم يكن الألبوم موجود في الملف فإن كود الحذف لن يفعل شيئاً .

لأنه ألوم جديد وبالتالي يضاف أسم هذا الألبوم إلى القائمة المنسدلة في السطر الثاني عشر

السطر الرابع عشر عرفنا متغير من نوع عقدة أسمه albumend وأسندنا له عقدة جديدة بعد استدعاء تعليمة إنشاء العقدة :

كود

```
doc.CreateElement(albumename);
```

حيث يقوم الكود السابق بإنشاء عقدة جديدة في الملف أسمها هو المدخل albumename وهو أسم الألبوم المراد حفظه.

السطر الخامس عشر عبارة for من الصفر إلى عدد عناصر القائمة albume لإنشاء عقد جديدة لكافة عناصرها.

السطر السابع عشر عرفنا متغير من نوع عقدة أسمه filend حيث سيقوم بحمل عقدة جديدة أسمها file بعد استدعاء تعليمة إنشاء العقدة

كود

```
doc.CreateElement("file");
```

لاحظ لحد الآن أن لدينا عقدتين , العقدة albumend التي تحمل أسم الألبوم , والعقدة filend التي تحمل أسم file.

بقي أن ندخل البيانات إلى العقدة file

السطر الثامن عشر عرفنا متغير من نوع عقدة وحملناه قيمة نصية من خلال استدعاء إجراء إنشاء عقدة نصية

كود

```
doc.CreateTextNode(albume[i]);
```

حيث يستقبل هذه الإجراء مدخل من نوع نص وهو البيانات المراد خزنها في العقدة , وهي العنصر الذي ترتيبه i من القائمة albume

الآن أصبح لدينا ثلاث عقد عقدة albumend التي تحمل أسم الألبوم , والعقدة filend التي تحمل أسم file وعقدة path التي تحمل مسار الملف من عنصر القائمة albume .إلى الآن لم يتم ربط العقد ببعضها البعض فهي إلى حد الآن عقد مستقلة ومنفصلة .يتم ربط عقدة فرعية بعقدة رئيسية من خلال استدعاء الإجراء AppendChild من العقدة الرئيسية.

وتمرير العقدة الفرعية كمدخل لهذا الإجراء , حيث في السطر التاسع عشر ربطنا العقدة path بعقدة filend وأصبحت العقدة file تحتوي على بيانات , ثم في السطر العشرين ربطنا العقدة filend بما فيها بالعقدة albumend.

وبتكرار العملية لكافة عناصر القائمة albume ستنتج لنا عقدة رئيسية أسمها باسم المدخل albumename تحتوي على عدة عقدة فرعية file تحتوي على مسارات الملفات في الألبوم .لاحظ أن عقدة الألبوم حتى الآن لم ترتبط بعقدة رئيسية , بل ربطناها بعقد فرعية فقط.

في السطر الثاني والعشرين ربطنا العقدة albumend بالعقدة الرئيسية في الملف albumes

وبهذا أصبح لدينا جميع العقد متصلة ومترابطة مع بعض ولم يبقى سوى حفظ التغيرات إلى الملف في السطر الثالث والعشرين.بقي أن نحدد مكان استدعاء هذا الإجراء , أذهب إلى تصميم نافذة الألبومات وأنقر مزدوجاً على زر "حفظ الألبوم"

ستنتقل إلى الشفرة , أكتب هذا :

كود

```
WriteAlbume(ac.Text);
```

بقي لدينا زر واحد وهو زر "البوم جديد" , أذهب إلى تصميم نافذة الألبومات وانقر مزوجاً على الزر "ألبوم جديد"

وأكتب داخلة هذه التعليمات :

كود

```
lv.Items.Clear();
albume.Clear();
ac.Text = "";
```

في السطر الأول قمنا بتنظيف كائن المستعرض من أي ملفات موجودة مسبقاً .

وفي السطر الثاني قمنا بتنظيف القائمة albume من أي عناصر موجودة مسبقاً.

وفي السطر الثالث قمنا بتصفير النص المكتوب على القائمة المنسدلة , بحيث أصبح كل شيء جاهز لإنشاء ألبوم جديد.

شغل البرنامج الآن وقم بتجريب حفظ ألبوم و استرجاع بياناته.

ملاحظة: عند أنشاء ألبوم جديد تأكد من أن إسم الألبوم لا يحتوي على فراغات أو رموز غريبة

لأن أسماء العقد يجب أن لا تحتوي على فراغات أو رموز غريبة وإلا ظهرت رسالة الخطأ .

واجب الدرس الحادي عشر

أرسل التطبيق

.نهاية الدرس الحادي عشر

..

الدرس الثاني عشر

تحسين وظائف التطبيق

من البديهيات في البرمجة أن يحتوي أي تطبيق على شوائب bugs, تحد من أداء التطبيق مهامه على أكمل وجه والشوائب هي الأخطاء البسيطة في البرنامج التي تنتج في ظروف محددة .والمبرمج عادة لا يكتشف هذه الأخطاء أثناء العمل في التطبيق , وإنما يكشفها اختبار التطبيق .وعملية اختبار التطبيق تعتبر من أهم مراحل تطوير البرمجيات وربما أكثر تعقيداً واستهلاكا للوقت والجهد .

ولكن لحسن الحظ هذه العملية لا تحتاج إلى مبرمجين ليقوموا بها , وإنما دورهم هنا هو التعديل في كود التطبيق الذي يقوم بهذه العملية هم المستخدمون أنفسهم , من خلال توزيع التطبيق بنسخ تجريبية Beta .وعادة ما تكون هذه النسخ مجانية, لأن مطور التطبيق هو المستفيد من اكتشاف الأخطاء .حيث يطلب من مستخدمي النسخ التجريبية إرسال أي أخطاء قد تواجههم إلى المطور .

وبهذا يوفر المطور على نفسه الكثير من المجهود والوقت في اختبار التطبيق .

والشوائب عادة ما تكون إحدى ثلاثة:

1-أخطاء الشوائب برمجية

وهي الشوائب التي تحدث بسبب خطأ في شفرة التطبيق , وتنتج هذه الأخطاء لأسباب كثيرة .وتسبب خلل في عمل البرنامج , كأن يتوقف البرنامج فجأة أو يقوم بعمليات غير منطقية .

وكمثال على هذه الشوائب :

أفتح تطبيق الدرس السابق و أختر مجموعة ملفات من النافذة الأساسية .ثم أفتح نافذة الألبومات وأختر أحد الألبومات , ثم أختر إلغاء الأمر من نافذة الألبومات, المفروض بما أنك ألغيت الأمر أن لا يتغير شيء في قائمة التشغيل في النافذة الرئيسية .

ولكن القائمة تغيره رغم أنك اخترت "إلغاء الأمر"

هذا الخطأ يعتبر من الشوائب البرمجية , وعلى المطور أن يبحث عن مكان الخطأ ويصححه, مكان الخطأ هنا هو إجراء النقر على زر "التحكم بالألبومات" .

أفتح تصميم النافذة الرئيسية وانقر مزدوجاً على زر "التحكم بالألبومات" لتنتقل إلى الشفرة ستجد

هذا الكود :

كود

```
        try
{
   albumes frm = new albumes();
    frm.albume = albume;
    frm.ShowDialog();
   albume = frm.albume;
   FillList();
   LoadFile(albume[0]);
       SetTrackBar();
   PlayFile();
}
catch
{
   MessageBox.Show("حدث خطأ أثناء محاولة تشغيل الملف المحدد");
}
```

لاحظ السطر الخامس تعليمة إظهار نافذة الألبومات , بعدها تماما تعليمات التغيير في قائمة التشغيل وهنا الخطأ , المفترض أن نختبر نتيجة نافذة الألبومات , فإذا كانت "موافق" ينفذ كود تغيير قائمة التشغيل .وبالتالي يجب أن نضيف للكود السابق عبارة if لاختبار ناتج النافذة قبل تنفيذ تعليمات التغيير في قائمة التشغيل.

هناك خطأ آخر هنا في السطر الرابع :

كود

```
frm.albume = albume;
```

هذه التعليمة تساوي بين القائمة albume في كلا النافذتين , بحيث إذا تغير شيء في قائمة albume في هذه النافذة. سيتغير نفس الشيء في قائمة albume في النافذة الألبومات والعكس صحيح أيضاً .وهذا يعتبر خطأ لأنه من المفترض أن تكون كل قائمة مستقلة عن الأخرى , والمفروض أن لا نعمل مساواة بين القائمتين .وإنما نسخ عناصر القائمة في هذه النافذة إلى القائمة في النافذة الأخرى , والنسخ غير المساواة.

المساواة تعني أن كلا النافذتين تحمل نفس القيمة دائماً ,فإذا ساويت بين قائمتين ثم عدلت في أي منهما فإن الأخرى بالضرورة ستعدل تلقائيا .أما النسخ فإذا نسخت قائمة من أخرى وعدلت في أي منها فإن الأخرى لا تتأثر لأنهما مستقلتان عن بعض .وهذا ما نحتاجه هنا , و النسخ في القوائم يتم عبر الإجراء **AddRange** كما شرحناه سابقاً .وبالتالي فالمفروض استبدال السطر الرابع بهذا السطر :

كود

```
frm.albume.AddRange(albume);
```

وبإضافة عبارة فحص نتيجة النافذة يصبح كود النقر على زر "التحكم بالألبومات" هكذا :

كود

```
try
   {
      albumes frm = new albumes();
      frm.albume.AddRange(albume);
        frm.ShowDialog();
      if (frm.DialogResult == DialogResult.OK)
      {
         albume = frm.albume;
         FillList();
         LoadFile(albume[0]);
         SetTrackBar();
         PlayFile();
      }
   }
   catch
   {
      MessageBox.Show("المحدد حدث خطأ أثناء محاولة تشغيل الملف");
   }
```

لا حظ داخل حاصرتي **if** أنا ساوينا بين قائمتين , وهنا المساواة هي المطلوبة وليس النسخ .لأن ناتج النافذة هي الموافقة وبالتالي نريد مساواة بين قائمة نافذة الألبومات التي قمنا بالتعديلات عليها والقائمة في النافذة الرئيسية.

وكمثال آخر على هذا النوع من الأخطاء شغل التطبيق وانقر على زر ملئ الشاشة دون تشغيل ملف صوتي ستظهر رسالة خطأ ويتوقف البرنامج عن العمل , لأنه لا يوجد فيديو حتى يجعله ملئ الشاشة ولحل هذه المشكلة أذهب إلى كود زر ملئ الشاشة , وأضف جملة try :

كود

```
try
{
    vp.Fullscreen = true;
}
catch
{
    MessageBox.Show("لا يوجد ملف فيديو ");
}
```

2-أخطاء نقص في الوظائف

وهي الأخطاء التي لا تنتج عن خطأ في الكود وإنما تنتج عن نقص في وظائف التطبيق ,أي أن التطبيق لا يحقق جميع الوظائف المطلوبة من المستخدم , ويتم حل هذه المشكلة بالتحديث Updating ,وهذه التحديثات عادة ما تتكرر من إصدار لآخر في التطبيق.

كمثال على هذه الأخطاء في تطبيقنا :

المستخدم عادة ما يفضل طريقة لاختيار سريع للألبوم الذي يريد تشغيله .وفي تطبيقنا لا يمكن اختيار الألبوم إلى بفتح نافذة الألبومات , وهذا يعد نقصاً في متطلبات المستخدم .لذلك يجب التعديل في البرنامج لعمل قائمة منسدلة لاختيار الألبوم في نافذة التطبيق الرئيسية .نحتاج لذلك إلى كائن من نوع XmlDocument ولذلك يجب إضافة عبارة using كما فعلنا في نافذة الألبومات

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml;

namespace MyMediaPlayer
{
```

ثم نعرف متغير عام أسمه doc من نوع XmlDocument تحت تعريف القائمة albume كما فعلنا في نافذة الألبومات .

أفتح تصميم النافذة الرئيسية , ومن صندوق الأدوات أسحب كائن القائمة المنسدلة إلى النافذة الرئيسية فوق قائمة التشغيل

وقم بتغيير الخاصية Name إلى ac و الخاصية Text إلى "أختر الألبوم" .

يجب تحميل أسماء الألبومات بمجرد فتح البرنامج , لذلك سنكتب شفرة تحميل أسماء الألبومات في حدث تحميل النافذة كما فعلنا في نافذة الألبومات .

افتح تصميم النافذة الرئيسية و انقر مزدوجاً على مكان فارغ في النافذة لتذهب إلى كو تحميل النافذة

MainForm_Load أو Form1_Load

أنسخ هذا الكود بين الحاصرتين :

كود

```
doc.Load(Application.StartupPath + "//albumes.xml");
   XmlNode albumesnd = doc["albumes"];
   XmlNode alnd = albumesnd.FirstChild;
   while (alnd != null)
   {
      ac.Items.Add(alnd.Name);
      alnd = alnd.NextSibling;
   }
```

الكود السابق هو نفسه الكود في إجراء تحميل نافذة الألبومات ,نحتاج لقراءة ملفات الألبوم المحدد إلى إجراء ReadAlbume الموجود في نافذة الألبومات ,أنسخ كود الإجراء ReadAlbume إلى شفرة النافذة الرئيسية .

وتأكد من أنه خارج أي إجراء آخر :

كود

```
void ReadAlbume(string albumename)
  {
     try
     {
        doc.Load(Application.StartupPath + "//albumes.xml");
        XmlNode filend = doc["albumes"][albumename].FirstChild;
        albume.Clear();
        while (filend != null)
        {
           albume.Add(filend.InnerText);
           filend = filend.NextSibling;
        }
        FillList();
     }
     catch
     {
        MessageBox.Show("الألبومات حدث خطأ أثناء محاولة إسترجاع بيانات");
     }
  }
```

كما نحتاج أيضا لإضافة كود استدعاء الإجراء **ReadAlbume** في حدث التغيير في القائمة المنسدلة .

أذهب إلى تصميم النافذة الرئيسية وانقر مزدوجاً على كائن القائمة المنسدلة **ac** لتذهب إلى الشفرة.

واكتب بين الحاصرتين كود استدعاء إجراء قراه لملفات الألبوم :

كود

```
try
  {
  ReadAlbume(ac.Text);
  FillList();
  LoadFile(albume[0]);
  SetTrackBar();
  PlayFile();
  }
  catch
  {
     MessageBox.Show("حدث خطأ أثناء محاولة تشغيل الملف المحدد");
  }
```

السطر الثالث يستدعي الإجراء ReadAlbume ليتم تحميل ملفات الملف المحدد إلى القائمة albume ,

والسطر الرابع يقوم بإستدعاء الإجراء FillList لتحميل عناصر albume إلى قائمة التشغيل .

والسطر الخامس يقوم بتحميل أول عنصر من عناصر القائمة ,

والسادس يضبط شريك التنقل و السابع يشغل الملف..

3- أخطاء في الواجهات

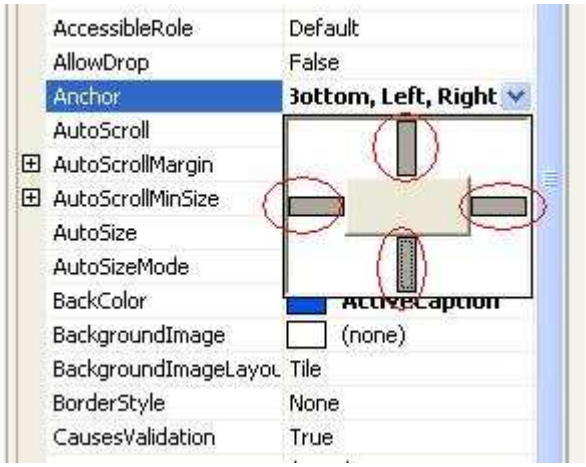
وهي الأخطاء التي لا تؤثر على عمل البرنامج ولكن تؤثر على الواجهات الصورية للنوافذ ,وهذه الأخطاء عادة ما يتم حلها بواسطة التحديثات Updating .

تطبيق الدرس الثاني عشر

أفتح التطبيق , وانقر على زر تكبير النافذة في شريط العنوان .ستلاحظ أن جميع كائنات النافذة محصورة في الزاوية العليا اليسرى من النافذة . حيث أن تكبير النافذة لم يقم بتعديل حجم و أماكن الكائنات ,ولحل هذه المشكلة سنستخدم خاصية واحدة في جميع الكائنات : Anchor

معنى الكلمة "المرساه" ومن معناها يتبين لنا وظيفتها ,حيث تقوم بربط الكائن بأحد جوانب النافذة أو بعدة جوانب فمثلاً لو ربطت كائن قائمة التشغيل بيمين النافذة , وقمت بتكبير النافذة فستلاحظ أن مكانها تغير وأنزاح نحو يمين النافذة. أما إذا ربطت أي كائن بجانبين متعاكسين , كأن تربط شريط التنقل بيمين ويسار النافذة ,وقمت بتكبير النافذة ستلاحظ أن حجم شريط التنقل قد تغير لأنه مرتبط باتجاهين متعاكسين وهذا يؤدي إلى "مط" الكائن .

أذهب إلى تصميم النافذة الرئيسية , وأختر الكائن الحاضن للفيديو "شاشة الفيديو" ,وغير خاصية Anchor إلى Top, Right ,Bottom, Left



لاحظ أنا ربطنا كائن الشاشة بجميع جوانب النافذة وبالتالي فحجمها سيتمدد في جميع الجهات ,ولكن المسافة بين نهاية الكائن وطرف النافذة من الأربع الجهات لن يتغير .

أختر صندوق "طريقة التشغيل" وغير الخاصية anchor إلى : Bottom, Right

زر التحكم بالألبومات : Bottom, Right

القائمة المنسدلة لاختيار الألبوم : Top, Right

قائمة التشغيل : Top, Bottom, Right

شريط التنقل : Bottom, Left, Right

أزرار التشغيل : Bottom, Left

الآن شغل التطبيق , وجرب تغيير حجم النافذة ستلاحظ أن الكائنات تغير من حجمها و مكانها بحسب حجم النافذة أيضاً كمثال على هذه الأخطاء , شكل شريط التنقل الكبير , والمشوه ,لحل مشكلة شكله الكبير سنلجأ إلى خدعة صغيرة :

سنقوم بتغطية جزء كبير منه فوق وتحت شريط الوسط .وذلك باستخدام كائن النص الثابت Label , أفتح تصميم النافذة الرئيسية ومن صندوق الأدوات أسحب كائن Label مرتين .وقم بتغيير الخاصية BackColor في كليهما إلى Transparent من تبويب Web وأحذف أي كلمة أمام خاصية Text

ثم قم بتغطية الأجزاء الزائدة من شريط التنقل ولا تنسى أن تغير الخاصية Anchor فيهما كما هي في شريط التنقل ,أيضاً يمكن أن تغير أيقونه التطبيق بإختيار النافذة الرئيسية وتغيير الخاصية Icon .لتضيف لمسة احترافية على التطبيق قم بتغيير الصور على الأزرار من خلال الخاصية Image

وتغيير خلفية النافذة من خلال الخاصية BackGroundImage

ملاحظة : هناك خطأ برمجي في بيئة تطوير ميكروسوفت وهو أن خلفية النافذة لا تظهر إذا كانت الخاصية RightToLeft تساوي True

سيبدو شكل التطبيق بعد هذه التعديلات هكذا :

واجب الدرس الثاني عشر

أختبر التطبيق وصحح أي أخطاء برمجية أخرى

.نهاية الدرس الثاني عشر

.نهاية الدروس

تم بحمد الله

..

الدرس الثالث عشر

التحكم بالصوت

هناك العديد من الطرق للوصول إلى أجهزة الصوت في نظام التشغيل .منها الطريقة المباشرة و التي تتعامل مباشرة مع محركات أجهزة الصور ,وهذه الطريقة معقدة نسبياً وقد تؤدي إلى إعطاب أجهزة الصوت في الحاسوب إذا ما أخطأ المبرمج في تعليمة واحدة .وهناك طرق غير مباشرة تتعامل مع أدوات تشكل وسيط بين أجهزة الصوت ولغة البرمجة ,بحيث يسهل التعامل مع الأجهزة , و تجنب أي ضرر في أجهزة الحاسوب إذا ما حدث خطأ ما ,الأداة الوسيطة التي سنستخدمها هي أداة

Homa_Audio

وهذه الأداة تستخدم إصدار حديث من مكتبة **Microsoft.DirectX.dll**

وتستخدم مكتبة جديدة **Microsoft.DirectX.DirectSound.dll**

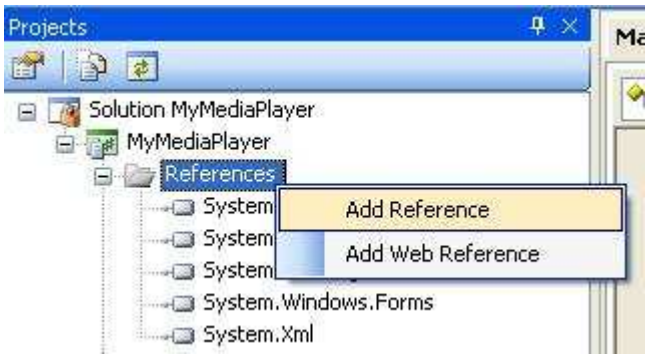
وبالتالي يجب استبدال أداة **Microsoft.DirectX.dll** من الدروس السابقة

بالإصدار الجديد الذي سيتم تنزيله مع الأداة الوسيطة

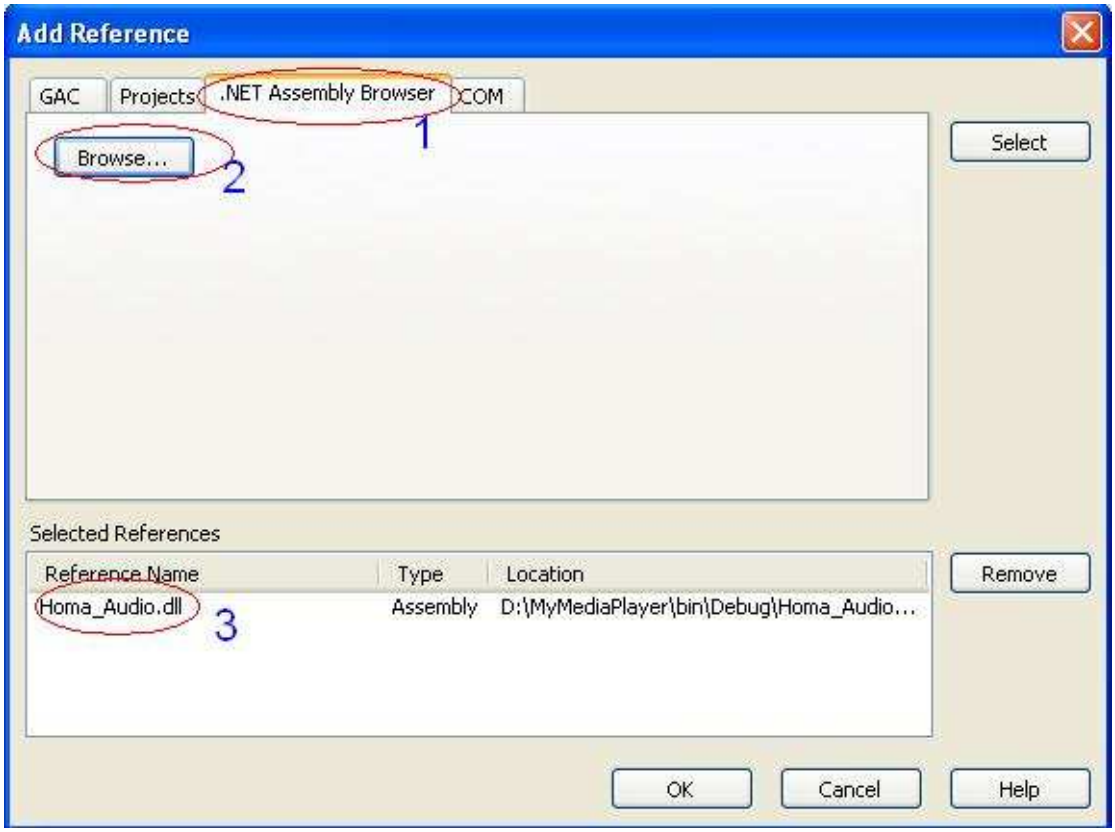
تطبيق الدرس الثالث عشر:

أنسخ هذا الملف إلى مجلد Debug داخل مجلد bin داخل مجلد المشروع .

الآن أفتح المشروع , ومن نافذة متصفح المشروع أختر Add Reference كما فعلنا في الدرس الرابع :



ستظهر لك هذه النافذة:

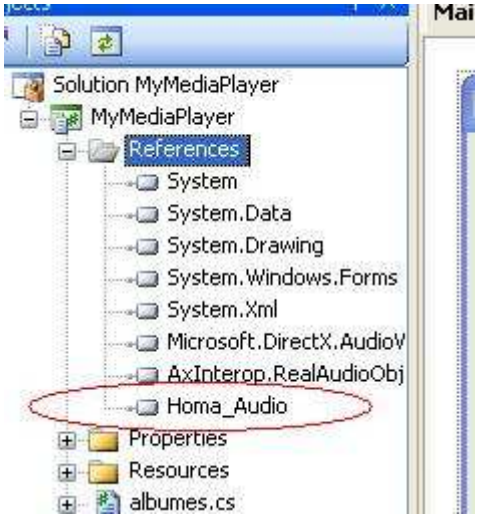


أختر Browse من تبويب .Net Assembly ثم أضف ملف

Homa_Audio.dll

الذي قمت بتنزيله ,حتى يظهر الملف في قائمة الملفات المختارة رقم 3 , ثم أختر OK

سترى أن الملف تم إضافتها إلى قائمة الملفات الملحقة :



الآن في أعلى كود النافذة الرئيسية للبرنامج .قم باستدعاء الملف ليتعرف عليه المعالج وذلك بإضافة الكود:

كود

```
using Homa_Audio;
```

يضاف هذا الكود أسفل تعليمات الإضافة السابقة كما في الصورة :

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Microsoft.DirectX.AudioVideoPlayback;
using AxRealAudioObjects;
using System.Xml;
using Homa_Audio;

namespace MyMediaPlayer
{
    /// <summary>
    /// Description of MainForm.
    /// </summary>
    public partial class MainForm
    {
```

الآن قم بتعريف كائن التحكم بالصوت :

كود

```
private HVolume hv=new HVolume();
```

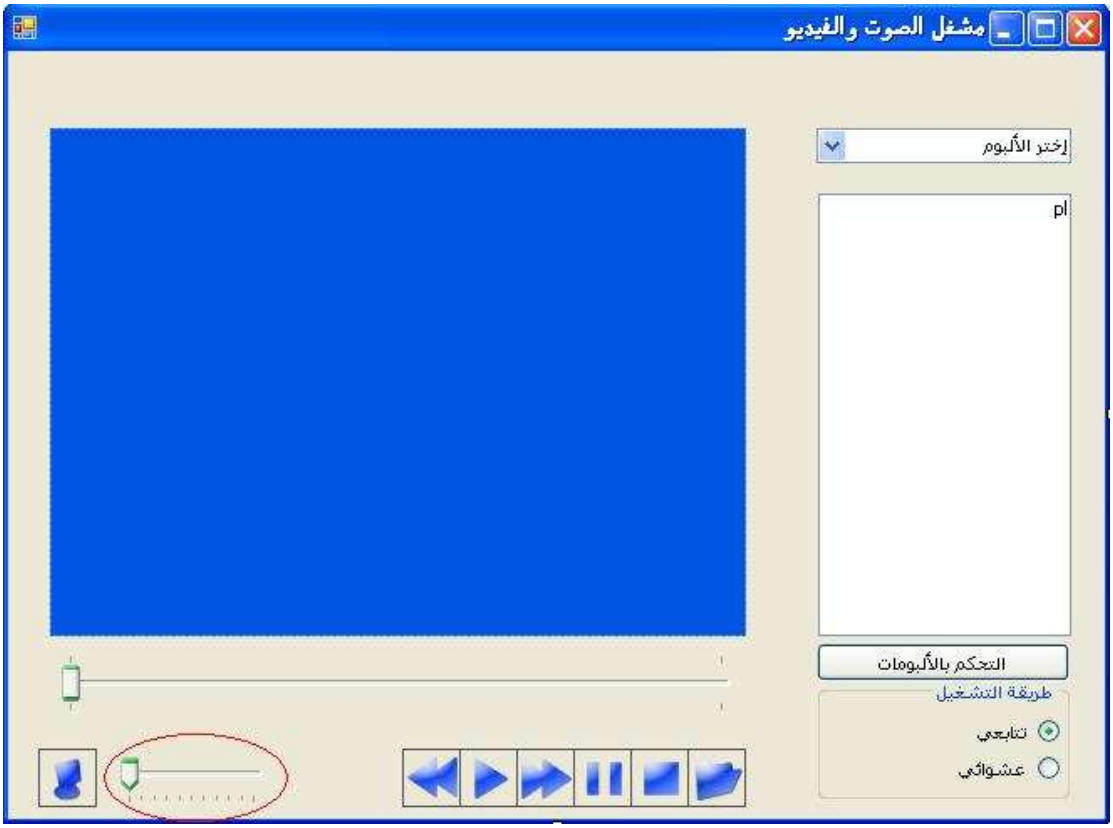
تحت المتغيرات من الدروس السابقة :

```
namespace MyMediaPlayer
{
    /// <summary>
    /// Description of MainForm.
    /// </summary>
    public partial class MainForm
    {
        private Audio ap;
        private Video vp;
        private AxRealAudio rp=new AxRealAudio();
        private string type;
        private List<string> albume=new List<string>();
        private int current=0;
        private bool userend = false;
        private XmlDocument doc = new XmlDocument();
        private HVolume hv=new HVolume();

        [STAThread]
        public static void Main(string[] args)
        {
            Application.EnableVisualStyles();
```

وأذهب إلى التصميم , وأسحب إلى الفورم أداة Track Bar

وغير ترتيب أدوات الفورم حتى يصبح شكلها هكذا :



ثم أختر الأداة التي قمت بإضافتها وغير خاصية Name إلى vt ,وغير أيضاً خاصية Value من 0 إلى 10 حيث ستكون هذه هي أداة التحكم بالصوت .من التصميم أنقر مزدوجاً على أداة التحكم بالصوت لتنتقل إلى الكود , ثم أكتب هناك الكود التالي :

كود

```
hv.Volume=vt.Value;
```

كائن التحكم بالصوت hv يحتوي على خاصية Volume تستقبل رقم من 0 إلى 10 يمثل مستوى الصوت فلو أسندنا لهذه الخاصية القيمة 0 فلن نسمع أي صوت .وبالتالي فهذه الخاصية ستأخذ أي قيمة من 0 إلى 10 من خلال كائن شريط التحكم بالصوت vt .الكود السابق يقوم بقراة قيمة شريط التحكم بالصوت vt و إسنادها إلى كائن التحكم بالصوت hv .

واجب الدرس الثالث عشر :

أضف مربع اختيار Check Box يمثل خاصية كتم الصوت ليصبح التطبيق هكذا :

مساعدة :

أسحب أداة مربع الاختيار إلى النافذة الرئيسية :

ثم أختره وغير خاصية Name إلى mute وخاصية Text إلى "كتم" .ثم أنقر عليه مزدوجاً لتنتقل إلى الكود وأكتب هذا الكود هناك :

كود

```
if(mute.Checked)
{
hv.Volume=...;
}
else
{
hv.Volume=vt.Value;
}
```

الكود السابق يفحص مربع الاختيار إذا كان مختاراً ينفذ من بين الحاصرتين الأولى ,وإن لم يكن مختاراً ينفذ ما بين الحاصرتين الثانية وهي إسناد قيمة شريط التحكم بالصوت vt.Value .إلى خاصية Volume في كائن التحكم بالصوت hv .

أكل الفراغ بقيمة محددة لكتم الصوت - تم شرحها سابقاً في الدرس -

ملاحظة :

عند تشغيل التطبيق ثم تفعيل مربع "كتم" سينقطع الصوت .ولكن عند تحريك شريط التحكم بالصوت سيرجع الصوت مع أن مربع "كتم" مفعل ,وبالتالي يجب وضع شرط في حدث تحريك كائن الصوت ليتم فحص مربع "كتم" .فإذا كان مربع "كتم" مفعلاً فلا يجب أن يتغير الصوت , ويصبح كود تحريك شريك التحكم بالصوت هكذا :

كود

```
if(mute.Checked)
{
}
else
{
hv.Volume=vt.Value;
}
```

الكود السابق يفحص إذا كان مربع "كتم" مفعل فلن ينفذ شي ,وإن لم يكن مفعلاً فسينفذ كود تغيير الصوت

يمكن كتابة الكود السابق بطريقة أخرى , باستخدام معامل العكس !

حيث يتم إضافة رمز التعجب ! قبل الجملة المنطقية ليعكس الجملة كلها.

مثلاً :

كود

```
if(mute.Checked)
```

تعني إذا كان مربع الكتم مفعل , و :

كود

```
if(!mute.Checked)
```

تعني إذا لم يكن مربع الكتم مفعلاً

وبالتالي فيمكن كتابة كود تحريك شريط التحكم بالصوت هكذا :

كود

```
if(!mute.Checked)
{
hv.Volume=vt.Value;
}
```

. نهاية الدرس الثالث عشر .

الدرس الرابع عشر

تسجيل الصوت

سنضيف في هذا الدرس خاصية مهمة للتطبيق .وهي إمكانية تسجيل الصوت سواء من الأجهزة الخارجية بواسطة لاقط الصوت .أو من الكمبيوتر نفسه باستخدام خاصية Mix in, نحتاج في هذا الدرس إلى كائن جديد من مكتبة Homa_Audio .

كائن HAudio الذي يقوم بعمليات التسجيل .ولاستخدام هذا الكائن في التطبيق لا بد من تعريفه وليكن أسمه ra .

أنسخ كود التعريف التالي تحت تعريف كائن تغيير الصوت من الدرس السابق :

كود

```
private HAudio ra=new HAudio();
```

```
public partial class MainForm
{
    private Audio ap;
    private Video vp;
    private AxRealAudio rp=new AxRealAudio();
    private string type;
    private List<string> albume=new List<string>();
    private int current=0;
    private bool userend = false;
    private XmlDocument doc = new XmlDocument();
    private HVolume hv=new HVolume();
    private HAudio ra=new HAudio();

    [STAThread]
    public static void Main(string[] args)
```

هذا الكائن يحتوي على إجراء التسجيل Record والذي يستقبل مدخل من نوع سلسلة نصية تمثل مسار الملف إلي سيتم التسجيل إليه .

مثلاً التعليمة التالي :

كود

```
ra.Record(@"c:\record.wav");
```

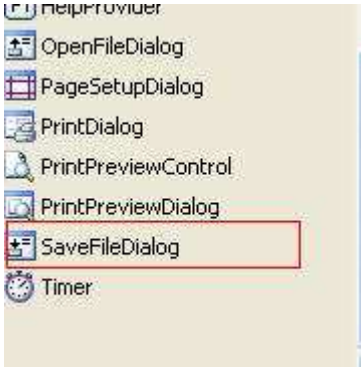
تقوم بتسجيل الصوت إلى الملف record.wav في القرص المحلي C .

أي أن كائن التسجيل يقوم بإنشاء الملف record.wav ثم تسجيل الصوت إليه .ولكن يجب أن لا يكون مسار الملف ثابت , يجب أن يختار المستخدم المكان الذي يرغب بتسجيل الصوت إليه ,لذلك سنستخدم نافذة جاهزة أخرى وهي نافذة حفظ الملفات SaveFileDialog

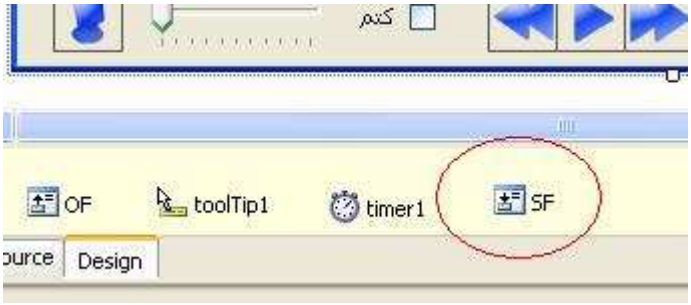
التسجيل

أذهب إلى نافذة التصميم , وأسحب زر جديد من صندوق الأدوات يمثل زر التسجيل وأضف إليه صورة مناسبة

أسحب أداة حفظ الملفات **SaveFileDialog** من صندوق الأدوات



وحددها من الشريط الأصفر في الأسفل .ثم غير خاصية **Name** إلى **SF** :



وغير خاصية **FileName** إلى **new.wav**

أنقر مزدوجا على زر التسجيل وأنسخ هذا الكود :

كود

```
SF.ShowDialog();
ra.Record(SF.FileName);
```

السطر الأول يقوم بفتح أداة حفظ الملفات **SF** كما فعلنا سابقاً في أداة فتح الملفات **OF**

السطر الثاني يستدعي إجراء التسجيل في كائن تسجيل الصوت **ra** الذي عرفناه سابقاً

واستبدلنا مدخل مسار ملف الصوت بناتج أداة حفظ الملفات حيث أن خاصية **FileName** في كائن **SF**

ترجع بقيمة سلسلة نصية تمثل المسار المختار من قبل المستخدم .

إيقاف التسجيل

كائن التسجيل يحتوي على إجراء أخر يقوم بعملية إيقاف التسجيل وحفظ الملف .

StopRecording هو إجراء إيقاف التسجيل , فالتعليمة التالية :

كود

```
ra.StopRecording();
```

هي تعليمة استدعاء إجراء إيقاف التسجيل .وهذا الإجراء يوقف عملية التقاط الصوت ثم يقوم بحفظ الملف الناتج

أي أن ملف التسجيل لن يتم إنشاؤه بشكل صحيح إلا بعد استدعاء إجراء إيقاف التسجيل ,أذهب إلى التصميم وأضف زر يمثل زر إيقاف التسجيل وأكتب داخله الكود المناسب

وذلك هو الواجب ..

ملاحظة :

يمكن للتطبيق الآن أن يسجل الصوت من نفسه , حيث أن أدوات التسجيل منفصلة عن أدوات التشغيل وهذه ميزه مهمة

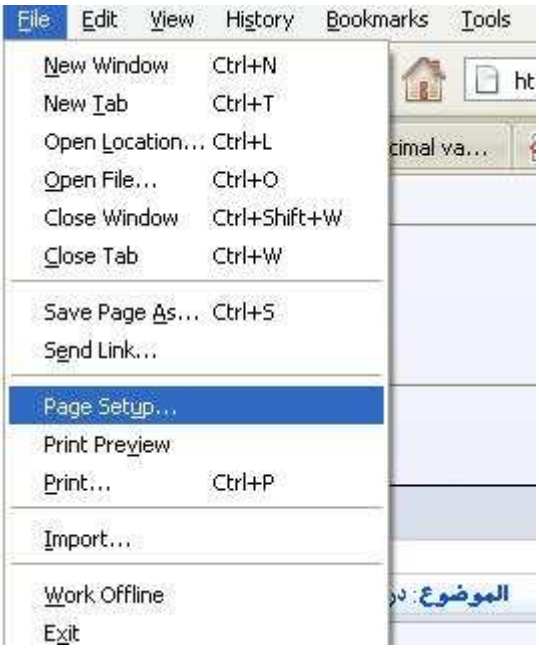
نهاية الدرس الرابع عشر .

الدرس الخامس عشر

القوائم:

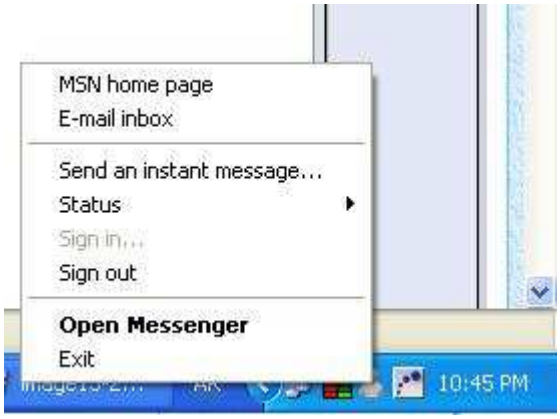
نادراً ما نجد برنامجاً يخلوا من القوائم Menus .. والقوائم تنقسم إلى ثلاث أنواع :

1 - قوائم رئيسية , وهي القوائم التي تأتي عادة أعلى النوافذ .كقائمة "ملف" و "تحرير" في أي نافذة من نوافذ ويندوز



2 - قوائم فرعية Context Menu و هي القوائم التي تنبثق عادة عند النقر على زر الفأرة الأيمن ,مثل قائمة التحرير في محررات النصوص:

3- قوائم شريط المهام وهي القوائم التي تنبثق من أيقونات البرامج التي لها اختصارات في شريط المهام كقائمة المسنجر:



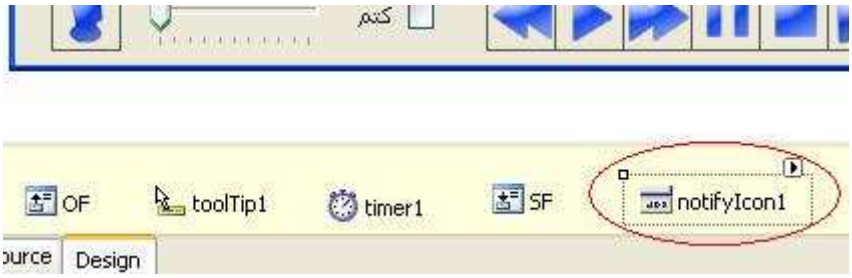
وطريقة التعامل مع القوائم بأنواعها المختلفة متشابهة جدا في معظم لغات البرمجة ومنها السي شارب.في درسنا اليوم سنضيف قائمة للتطبيق من النوع الثالث ,حيث سنضيف إليها جميع وظائف الأزرار في النافذة الرئيسية ,وسنضيف أيضاً خاصية إخفاء النافذة الرئيسية , وإضافة إيقونة للتطبيق في شريط المهام.

إيقونة شريط المهام Notify Icon

وهي الإيقونات التي تظهر عادة في شريط المهام بجانب الساعة .مثل إيقونات المسنجر و الشبكة وغيرها.

أفتح نافذة التطبيق الرئيسية , وأسحب كائن إيقونة التنبيه من صندوق الأدوات من جزء Components

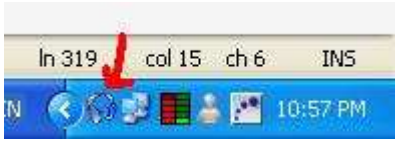
ستجد الإيقونة في الشريط الأصفر تحت النافذة , أخترها بالنقر عليها مرة واحدة بزر الفأرة الأيسر.



ثم أذهب إلى جدول الخصائص ومن خاصية Icon أختر الإيقونة المناسبة ,وغير خاصية Text إلى

"مشغل الصوت والفيديو"

وعند تشغيل التطبيق ستلاحظ إيقونة التحذير التي اخترتها موجودة في شريط المهام.



الآن سنضيف خاصية إخفاء النافذة للتطبيق وتعليمة إخفاء النافذة بسيطة جداً :

كود

```
this.Hiade();
```

حيث أن this تمثل النافذة الحالية وهي النافذة الرئيسية ,بقي أن نجد المكان المناسب لوضع تعليمة الإخفاء وأفضل مكان لهذه التعليمة هي عند تصغير النافذة الرئيسية للبرنامج ,حيث ستختفي النافذة من شريط المهام و لا يظهر منها سوى إيقونة التنبيه التي اخترناها .

ولعمل ذلك يجب أن نكتب تعليمة الإخفاء في حدث Resize للنافذة , وهذا الحدث ينطلق عند تغيير حجم النافذة ولكن هناك مشكلة , المفروض أن نخفي النافذة فقط عندما يتم تصغير النافذة , ولكن هذا الحدث ينطلق عند تصغير أو تكبير النافذة.

وبالتالي فيجب أن نضيف جملة شرط if نختبر فيها حالة النافذة فإذا كانت النافذة مصغرة يتم تنفيذ كود الإخفاء ونختبر حالة النافذة من خلال خاصية WindowsState من خصائص الكائن this والذي يمثل النافذة الرئيسية فالتعليمة التالية ترجع قيمة تمثل حالة النافذة :

كود

```
this.WindowState
```

ونتائج هذه الخاصية الشائعة هي ثلاث نتائج "حالات" :

1- الحالة العادية , وهي حالة تكون فيها النافذة بحجمها الأصلي كما في التصميم وترجع بقيمة :

كود

```
FormWindowState.Normal
```

2- الحالة المكبرة , وهي حالة تكون فيها النافذة مكبرة ملئ الشاشة من زر التكبير في أعلى النافذة وترجع بقيمة :

كود

```
FormWindowState.Maximized
```

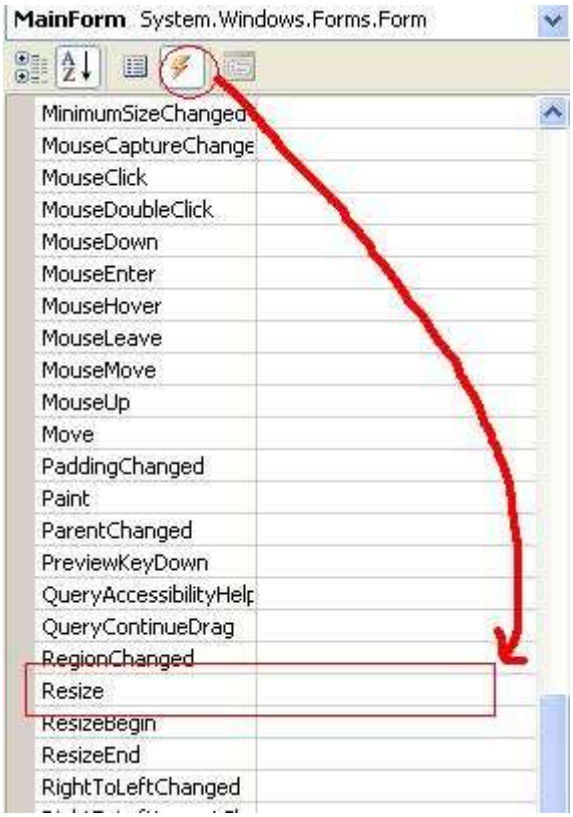
3- الحالة المصغرة , وهي حالة تكون فيها النافذة مصغرة وموجودة فقط في شريط المهام وترجع بقيمة :

كود

```
FormWindowState.Minimized
```

والحالة التي تهمنا هي آخر حالة , حيث سنختبر خاصية حالة النافذة WindowsState فإذا كانت Minimized فيتم تنفيذ كود الإخفاء.

أذهب إلى نافذة التصميم , وأختر النافذة الرئيسية بالنقر مرة واحدة على مكان فارغ بالنافذة .ثم أذهب إلى جدول الخصائص جزء الأحداث , وأنقر مزدوجاً على حدث Resize



ستنتقل إلى الكود , لنكتب هناك كود الإخفاء مع جملة if :

كود

```
if(this.WindowState == FormWindowState.Minimized)
{
   this.Hide();
}
```

السطر الأول يمثل جملة شرطية تختبر حالة النافذة إذا كانت مصغرة Minimized أم لا

فإذا كانت مصغرة يتم تنفيذ الكود بين الحاصرتين وهو تعليمة الإخفاء .

انتهينا من خاصية الإخفاء , بقي أن نضيف خاصية إعادة الإظهار.

وتعليمة الإظهار بسيطة أيضاً , وقد ذكرناها في دروس سابقة :

كود

```
this.Show();
```

حيث أن this يمثل النافذة الرئيسية .ولكن هذا الكود لا يكفي فهو يظهر النافذة على شريط المهام , ولا يقوم بتكبيرها

وبالتالي يجب إضافة كود تكبير النافذة , وهو تغيير لحالة النافذة WindowsState إلى Maximized

أي أن تعليمة تكبير النافذة كاملة هي :

كود

```
this.WindowState = FormWindowState.Normal;
```

تعليمة الإظهار والتكبير سنضعها كما قلنا في حدث النقر المزدوج على إيقونة التنبيه .

أذهب إلى التصميم وأنقر مزدوجاً على إيقونة التحذير notifyIcon1 في الشريط الأصفر أسفل النافذة ستنتقل إلى الكود

أكتب هناك تعليمات الإظهار والتكبير :

كود

```
this.Show();
this.WindowState = FormWindowState.Normal;
```

الآن شغل البرنامج وصغر النافذة , ستختفي النافذة من شريط المهام , وتبقى فقط إيقونة التحذير

وإذا نقرت عليها مزدوجاً ستظهر النافذة مرة أخرى .

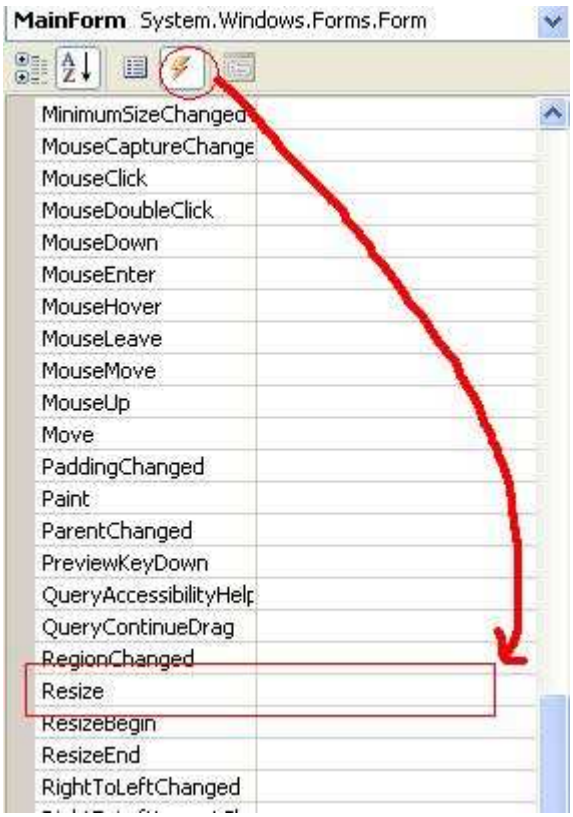
القوائم الفرعية :

أحياناً نحتاج إلى التفاعل مع البرنامج وهو مخفي , كأن نوقف التشغيل مثلاً .في حالة البرنامج الحالية , يجب أولاً إظهار النافذة الرئيسية بالنقر مزدوجاً على إيقونة التحذير .ثم عندما تظهر النافذة نختر زر الإيقاف ..

يمكن اختصار هذا الإجراء بإضافة قائمة على إيقونة التحذير , تحتوي على بند "إيقاف"

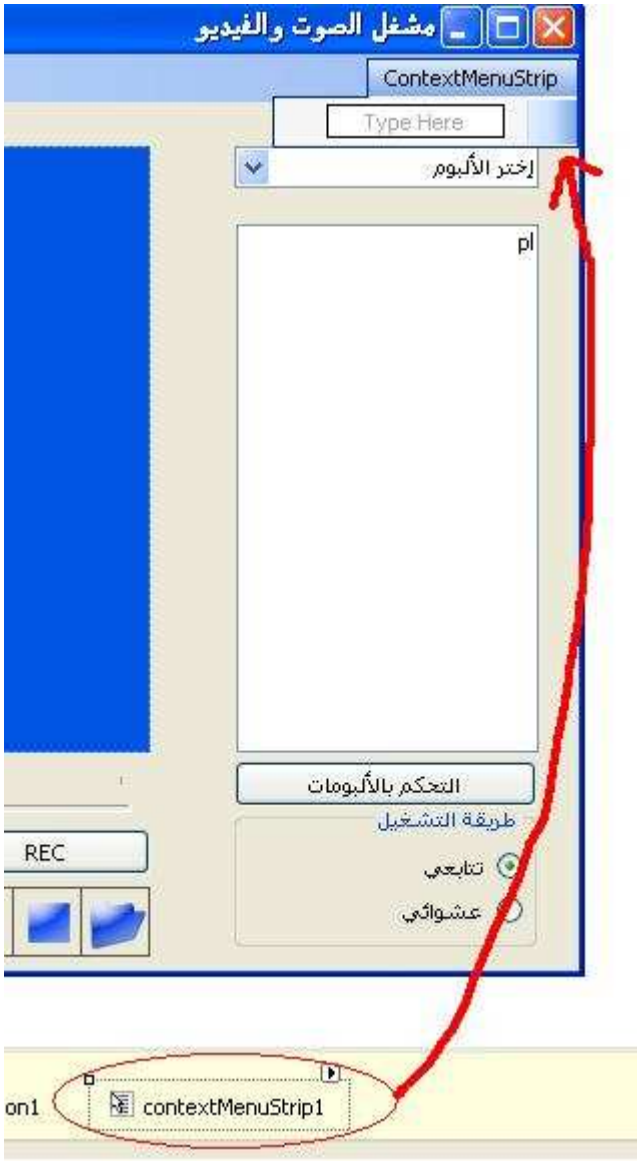
وبالتالي يمكن إيقاف التشغيل بدون إظهار النافذة الرئيسية .سنضيف إلى البرنامج كائن القائمة التي ستحتوي على جميع أوامر التطبيق :

أذهب إلى التصميم , وأسحب كائن القائمة الفرعية من صندوق الأدوات :



ستجد كائن القائمة أضيف إلى الشريط الأصفر أسفل النافذة .

أختر القائمة بالنقر عليها مرة واحدة , ستلاحظ أن شكل النافذة الرئيسية تغير و أضيفت إليها قائمة تحت شريط العنوان



هنا ستقوم بتحرير عناصر القائمة , ولإضافة عنصر جديد للقائمة أنقر مزدوجاً على المربع الأبيض المكتوب عليه

Type Here

وأكتب هناك " إيقاف التشغيل "

بهذا تكون قد أضفت عنصر جديد للقائمة وهو عنصر إيقاف التشغيل .أضف عنصر جديد تحت العنصر السابق وليكن

" إنهاء البرنامج "

الآن انقر في أي مكان فارغ في النافذة , ثم اخترت مرة أخري كائن القائمة من الشريط الأصفر في الأسفل .

ولكن هذه المرة سنقوم بإضافة كود إلى كل عنصر من عناصر القائمة , أنقر مزدوجاً على عنصر " إيقاف التشغيل"

ستنتقل إلى الكود , لتكتب هناك تعليمة إيقاف التشغيل وهي نفس التعليمة في زر إيقاف التشغيل في النافذة الرئيسية من الدروس السابق :

كود

```
StopFile();
```

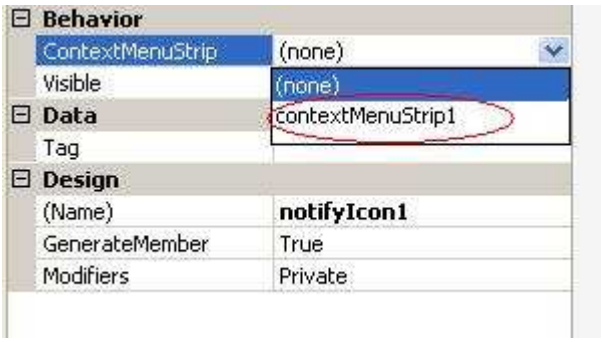
ثم أذهب إلى التصميم مرة أخرى و أنقر مزدوجاً على عنصر " إنهاء البرنامج"

وأكتب هناك كود إغلاق البرنامج الذي عرفناه من الدروس السابقة :

كود

```
System.Environment.Exit(0);
```

بقي أن نربط القائمة بإيقونة التحذير , أذهب إلى التصميم و أختر إيقونة التحذير notifyIcon1 من الشريط الأصفر بالأسفل.ومن خاصية ContextMenuStrip أختر contextMenuStrip1



الآن جرب تشغيل البرنامج , و صغر النافذة الرئيسية و أنقر بزر الفارة الأيمن على زر التحذير ,ستظهر لك القائمة الفرعية

واجب الدرس الخامس عشر

أضف عناصر أخري للقائمة تحتوي على جميع خصائص البرنامج

تشغيل , إيقاف مؤقت , تقديم , تأخير , فتح ملفات , التحكم بالألبومات , ملئ الشاشة, ...

.نهاية الدرس الخامس عشر .

..

الدرس السادس عشر

التوقيت:

من الإضافات المهمة للتطبيق إضافة طول الملف و الوقت الحالي للتشغيل بالدقيقة والثانية .ولإضافة ذلك نحتاج إلى استخدام خاصية CurrentPosition في كائنات تشغيل الصوت والفيديو أو GetPosition() في كائن تشغيل ملفات الريل.حيث إن هذه الخصائص ترجع قيمة من نوع رقم تحدد الثانية الحالية التي يتم تشغيلها في الملف .
فمثلاً إذا أعادت الخاصية CurrentPosition القيمة 120 فمعنى هذا أنه يتم الآن تشغيل الثانية رقم 120 في الملف وإذا أعادت الخاصية GetPosition() القيمة 50000 فمعنى ذلك أنه يتم الآن تشغيل الثانية رقم 50 في ملف الريل وقد سبق وشرحنا أن الاختلاف بين كائنات تشغيل الصوت والفيديو وكائن تشغيل الريل هو أن الأولى ترجع القيمة بالثانية أما كائن تشغيل الريل فيرجع القيمة بالملي ثانية و يجب قسمتها على 1000 كي نحولها إلى ثانية.

الحصول على طول الملف :

هناك خاصية أخرى عرفناها في دروس سابقة وهو خاصية Duration في كائن مشغل الصوت والفيديو
والتي ترجع قيمة من نوع رقم تمثل طول الملف بالثواني.
مثلاً التعليمة التالي :

كود

```
ap.Duration
```

لو كان نتاجها 195 فهذا يعني أن طول الملف 195 ثانية
والتعليمة :

كود

```
rp.GetLength()
```

لو كان نتاجها 195000 فهذا يعني أن طول الملف 195000 ملي ثانية أي 195 ثانية "بالقسمة على 1000"
لأن rp هو كائن مشغل ملفات الريل و يرجع القيمة بالملي ثانية لذلك وجب القسمة على 1000 لتحويلها إلى ثانية
ولكنا نريد كتابة الوقت بالدقيقة و الثانية . وعند تحويل 195 إلى دقائق تصبح ثلاث دقائق و 15 ثانية.
أي بهذه الصيغة :

كود

```
3:15
```

نحتاج إلى معادلة رياضية لتحويل الصيغة من ثواني إلى دقائق وثواني

إيجاد الدقائق :

لإيجاد الدقائق في كائن مشغلا الصوت أو الفيديو نقسم عدد الثواني على 60 ثم نحول الناتج إلى عدد صحيح حتى يتم إزالة الكسور العشرية.

فمثلاً لو قيمنا 195 على 60 سيكون الناتج 3.25 و هو عدد حقيقي لكنا نريد فقط عدد الدقائق و هو 3 وبالتالي فتحويل العدد الحقيقي إلى صحيح .

يتكفل بإزالة الكسر 25 ويصبح الناتج 3 .

كود

```
int m;
  m=vp.Duration/60;
```

السطر الأول تعريف للمتغير m من نوع رقم والذي سيحمل عدد الدقائق .

والسطر الثاني إسناد طول الملف مقسوما على 60 للتحويله إلى دقائق.

ولأنا نريد فقط الدقائق دون الكسور سنستخدم دالة جديدة Math.Floor والتي تحذف الكسر وترجع فقط العدد الصحيح

هذه الدالة لها مدخل من نوع رقم Double ودالتي Duration في كائنات الصوت والفيديو ترجع Double أيضاً

كود

```
int m;
  m =Convert.ToInt32(Math.Floor(vp.Duration/60));
```

وبنفس الطريقة لكائن تشغيل الصوت :

كود

```
int m;
  m =Convert.ToInt32(Math.Floor(ap.Duration/60));
```

أما كائن مشغل ملفات الريل فيجب كما قلنا أن نقسم الناتج على 1000 لتحويله إلى ثواني ثم نقسم على 60 لتحويله إلى دقائق.

وكما نعلم في الرياضيات فإن :

كود

```
x/y/z=x/y*z
```

وبالتالي بدل من كتابة كود التحويل هذا :

كود

```
rp.GetLength()/1000/60
```

يمكن كتابته هكذا :

كود

```
rp.GetLength()/60000
```

لأن 60000 = 1000 * 60

وأيضاً دالة GetLength في كائن مشغل الريل لا ترجع قيمة من نوع Double وهذا يسبب مشكلة في التعامل مع الدالة Math.Floor

لأنه يجب أن تستقبل مدخل من نوع Double

وبالتالي يجب تحويل ناتج الدالة GetLength إلى Double :

ويصبح كود التحويل كاملاً هكذا :

كود

```
int m;
  m = Convert.ToInt32(Math.Floor(Convert.ToDouble(rp.GetLength()/60000)));
```

إيجاد الثواني :

نقصد هنا بالثواني الكسر المتبقي من قسمة الثواني على 60

ولاستنتاج المعادلة الرياضية المستخدمة لإيجاد الثواني سنقسم عدد الثواني 195 إلى أربعة أقسام:

كود

```
60 + 60 + 60 + 15
```

الثلاثة الحدود الأولي تمثل الثلاث الدقائق التي أستخر جناها من معادلة إيجاد الدقائق وبقي 15 تمثل عدد الثواني المتبقية

يجب لإيجاد الثواني المتبقية أن نستبعد الثواني التي تم تحويلها إلى دقائق

إن أننا نقوم بطرح الثواني التي تم تحويلها إلى دقائق من عدد الثواني الإجمال :

كود

```
195 - (60 + 60 + 60 ) = 15
```

بهذه المعادلة استطعنا إيجاد الثواني المتبقية , ولكن البرنامج كيف يصيغ ما بين الأقواس

لنحول المعادلة قليلاً :

كود

```
195 - (60 * 3) = 15
```

ركز على رقم 3 بين الأقواس , أليس هو عدد الدقائق m الذي أوجدناه في الفرقة السابقة ؟

وبالتالي يمكن كتابة المعادلة هكذا :

كود

```
195 - (60 * m) = 15
```

إذاً يمكن إيجاد عدد الثواني المتبقية من خلال طرح عدد الدقائق مضروباً في 60 من العدد الإجمالي للثواني

ففي كائن تشغيل الفيديو ستكون تعليمة إيجاد الثواني المتبقية هكذا :

كود

```
vp.Duration-(60 * m)
```

وبالتحويل إلى العدد الصحيح تصبح :

كود

```
Convert.ToInt32(vp.Duration-(60 * m))
```

إذاً لإيجاد الثواني المتبقية :

كود

```
int s;
  s =Convert.ToInt32(vp.Duration-(60 * m));
```

وفي كائن تشغيل الصوت :

كود

```
int s;
  s =Convert.ToInt32(ap.Duration-(60 * m));
```

وفي كائن تشغيل ملفات الريل :

كود

```
int s;
  s =Convert.ToInt32(rp.GetLength()/1000-(60 * m));
```

لإيجاد الوقت الحالي :

المعادلتين السابقة كانت لإيجاد طول الملف بالدقيقة والثانية .ونحتاج إلى معادلات أخرى لإيجاد الموقع الذي يتم تشغيله حالياً بالدقيقة والثانية ,وهي نفس المعادلات السابقة باستثناء استبدال **Duration** بـ **CurrentPosition** في كائنات تشغيل الصوت والفيديو ,واستبدال **GetLength** بـ **GetPosition** في كائن تشغيل ملفات الريل حيث أن **CurrentPosition** و **GetPosition** ترجع رقم الثانية التي يتم تشغيلها الآن ,أو عدد الثواني التي تم تشغيلها , وبالتالي فيمكن حساب الدقائق والثواني ,من خلال تطبيق المعادلات السابقة على عدد الثواني التي تم تشغيلها

مثلاً لإيجاد الدقيقة التي يتم تشغيلها حالياً في كائن مشغل الفيديو :

كود

```
int m;
  m =Convert.ToInt32(Math.Floor(vp.CurrentPosition/60));
```

وفي كائن مشغل الصوت :

كود

```
int m;
  m =Convert.ToInt32(Math.Floor(ap.CurrentPosition/60));
```

وفي كائن مشغل ملفات الريل :

كود

```
int m;
  m = Convert.ToInt32(Math.Floor(Convert.ToDouble(rp.GetPosition()/60000)));
```

ولإيجاد الثانية التي يتم تشغيلها حالياً في كائن مشغل الفيديو :

كود

```
int s;
  s =Convert.ToInt32(vp.CurrentPosition-(60 * m));
```

وفي كائن مشغل الصوت :

كود

```
int s;
  s =Convert.ToInt32(ap.CurrentPosition-(60 * m));
```

وفي كائن مشغل ملفات الريل :

كود

```
int s;
  s =Convert.ToInt32(rp.GetPosition()/1000-(60 * m));
```

تطبيق الدرس السادس عشر :

أفتح تطبيق الدرس السابق , وأضف متغير محلي من نوع نص يمثل طول الملف بالدقيقة والثانية وليكن length

```
/// Description of MainForm.
/// </summary>
public partial class MainForm : Form
{
    private Audio ap;
    private Video vp;
    private AxRealAudioObjects.AxRealAudio rp=ne
    private string type;
    public List<string> albume=new List<string>(
    private XmlDocument doc =new XmlDocument();
    private int current;
    private HVolume hv=new HVolume();
    private HAudio ra=new HAudio();
    private string length;
```

أنسخ هذا الإجراء إلى الكود وتأكد أنه خارج أي إجراء آخر :

كود

```
void SetLength()
 {
    try
    {
    int m = 0;
    int s = 0;
       if (type == "V")
       {
          m = Convert.ToInt32(Math.Floor(vp.Duration / 60));
          s = Convert.ToInt32(vp.Duration - (60 * m));
       }
       else if (type == "A")
       {
          m = Convert.ToInt32(Math.Floor(ap.Duration / 60));
          s = Convert.ToInt32(ap.Duration - (60 * m));
       }
       else if (type == "R")
       {
```

```
                m = Convert.ToInt32(Math.Floor(Convert.ToDouble(rp.GetLength() /
60000)));
                s = Convert.ToInt32((rp.GetLength() / 1000) - (60 * m));
            }
            length = Convert.ToString(m) + ":" + Convert.ToString(s);
        }
        catch
        {
        }
    }
```

الإجراء السابق **SetLength** وظيفته إيجاد طول الملف بالدقيقة و الثانية و إسناده إلى المتغير المحلي **length**
حيث إستخدمنا جملة شرط ثلاثية لفحص نوع الملف ,و أدمجنا معادلتي إيجاد الدقائق والثواني في مكان واحد لكل نوع كما شرحنا في الدرس.
وفي السطر الأخير من كود جملة **try** حولنا الدقائق إلى نص , ثم أضفنا إليها : و الثواني بعد تحويلها أيضاً إلى نص
حيث إن المتغير **length** سيحمل قيمة من نوع نص تمثل طول الملف بهذه الصيغة :

كود

```
3:15
```

بقي أن نجد المكان المناسب لاستدعاء هذا الإجراء , حيث من المفترض أن نجد طول الملف بعد أن يتم تحميله
وبالتالي فإن إجراء تحميل الملف **LoadFile** هو المكان المناسب لاستدعاء .
أذهب إلى كود التطبيق و أضف تعليمة الاستدعاء التالية في نهاية إجراء تحميل الملف :

كود

```
SetLength();
```

حيث يصبح إجراء تحميل الملف هكذا :

كود

```
void LoadFile(string Path)
 {
 StopFile();
 ap=null;//الصوت تصفير
 vp=null;//الفيديو تصفير
 this.Controls.Add(rp);//الريل تصفير
 rp.Visible=false;//الريل أخفاء مظهر
 try
 {
    vp = Video.FromFile(Path);
      vp.Owner = this.panel1;
    panel1.Width = 400;
    panel1.Height = 300;
    type="V";
 }
 catch
 {
    try
    {
          ap = Audio.FromFile(Path);
          type="A";
     }
     catch
     {
         try
         {
            rp.SetSource(Path);
            type="R";
         }
         catch
         {
              MessageBox.Show("صالح إختيار ملف الرجاء");
```

```
                type="N";
            }
        }


    }
SetLength();
 }
```

عند تحميل أي ملف سيتم تحميل المتغير length قيمة نصية تمثل طول الملف بالثواني والدقائق ,وإجراء إيجاد الوقت الحالي أثناء التشغيل مشابه جدا لإجراء إيجاد طول الملف ,ولكن قبله نحتاج إلى مكان على النافذة نكتب عليه طول الملف و الوقت الحالي للتشغيل .

حيث سيكون بعد الصيغة :

كود

```
1:25/3:15
```

بافتراض أن طول الملف هو 3 دقائق و 15 ثانية والوقت الحالي للتشغيل هو الدقيقة الأولى والثانية الخامسة والعشرين

أفتح نافذة التصميم و أسحب كاشن اللافته Label إلى النافذة :

ثم أختره وغير خاصية Name إلى len و غير خاصية Text إلى :

كود

```
0:0/0:0
```

والآن أذهب إلى الكود وأنسخ هذا الإجراء وتأكد من أنه خارج أي إجراء آخر :

كود

```
void SetPosition()
 {
    try
    {
       int m = 0;
       int s = 0;
       if (type == "V")
       {
          m = Convert.ToInt32(Math.Floor(vp.CurrentPosition / 60));
          s = Convert.ToInt32(vp.CurrentPosition - (60 * m));
       }
       else if (type == "A")
       {
          m = Convert.ToInt32(Math.Floor(ap.CurrentPosition / 60));
          s = Convert.ToInt32(ap.CurrentPosition - (60 * m));
       }
       else if (type == "R")
       {
          m = Convert.ToInt32(Math.Floor(Convert.ToDouble(rp.GetPosition() /
60000)));
          s = Convert.ToInt32((rp.GetPosition() / 1000) - (60 * m));
       }

       len.Text = Convert.ToString(m) + ":" + Convert.ToString(s) + " / " +
length;
    }
    catch
    {
    }
 }
```

الإجراء السابق SetPosition يقوم بإيجاد الوقت الحالي للتشغيل بالدقيقة والثانية .

في السطر الأخر من جملة try حولنا الدقيقة التي يتم تشغيلها إلى نص ثم أضفنا إليها ":"

ثم أضفنا إليها الثانية التي يتم تشغيلها حالياً , ثم أضفنا إليها "/" ثم طول الملف الذي أوجدناه في الإجراء SetLength

بقي أن نجد المكان المناسب لاستدعاء هذا الإجراء .فالمفروض أن نستدعي الإجراء كل ثانية أي أنا نحتاج إلى مؤقت Timer ولكن يوجد لدينا مؤقت من الدروس السابق .لذلك سنستخدمه عوضاً عن إضافة مؤقت جديد

أذهب إلى كود المؤقت timer1 من خلال النقر مزدوجا على كائن المؤقت timer1 من نافذة التصميم.

وأضف تعلميه الاستدعاء التالية في آخر كود المؤقت :

كود

```
SetPosition();
```

حيث يصبح كود المؤقت هكذا :

كود

```
void Timer1Tick(object sender, System.EventArgs e)
 {
 try
 {
    if (trackBar1.Maximum == 0)
    {
       SetTrackBar();
    }
    if (type=="V")
    {
      trackBar1.Value=Convert.ToInt32(vp.CurrentPosition);
    }
    else if (type=="A")
    {
       trackBar1.Value= Convert.ToInt32(ap.CurrentPosition);
    }
    else if (type=="R")
    {
       trackBar1.Value=Convert.ToInt32(rp.GetPosition());
```

```
    }
   SetPosition();
}
catch
{
}

}
```

واجب الدرس السادس عشر:

أرسل التطبيق

نهاية الدرس السادس عشر.

..

الدرس السابع عشر

تغيير الواجهات الرسومية

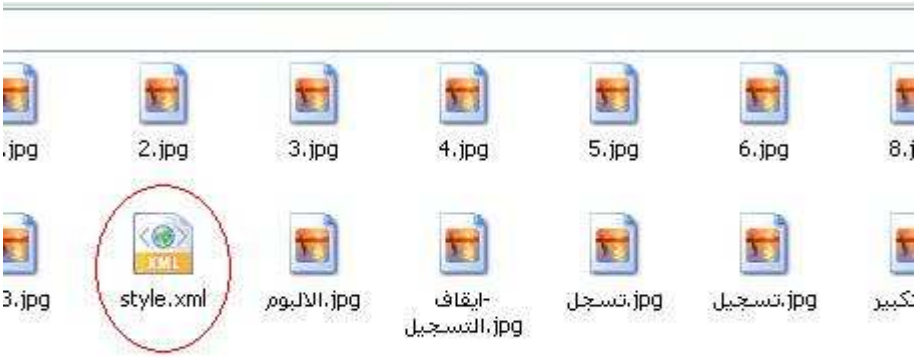
في هذا الدرس سنستخدم برنامج خاص بإنشاء واجهات رسومية للتطبيق.

قم بتحميل البرنامج من هنا:

برنامج صنع الواجهات

واجهة هذا البرنامج مبسطة وواضحة , حيث تقوم من خلالها بتحديد ألوان وخلفيات عناصر التطبيق .كتحديد خلفية النوافذ و ألوان الخطوط وغيرها.

يعتمد هذا البرنامج على تسجيل القيم المدخلة له في ملف Xml ,على أن برنامج مشغل الميديا سيقرأ القيم من هذا الملف ,والبرنامج يقوم بإنشاء مجلد جديد باسم الستايل ,ويضع داخله ملف XML الذي يحتوي على الألوان وأسماء الصور و يضع داخلة أيضاً الصور المختارة للستايل .



بعد إنشاء الستايل نقوم بنسخ المجلد كاملاً إلى مجلد جديد سنقوم بإنشاءه في مسار البرنامج باسمStyles حيث سيحتوي هذا المجلد على جميع الستايلات الناتجة من برنامج صنع الواجهات:

للتعامل مع الواجهات المتعددة , نحتاج إلى قائمة منسدلة جديدة تحتوي على جميع الستايلات

إعداد النوافذ لتقبل الستايلات

سنحتاج لتغيير أسماء معظم الكائنات وذلك لكي يسهل علينا التعامل معها بواسطة أسمائها:

أفتح تصميم النافذة الرئيسية وأختر زر التشغيل وغير أسمه من الخاصية Name إلىbtnPlay

وغير أسم زر الإيقاف إلى - btnStop

زر التسجيل - btnRecord

زر إيقاف التسجيل - btnStopRecord

زر التقديم - btnForward

زر التأخير - btnBackWard

زر ملئ الشاشة - btnFullScreen

زر الإيقاف المؤقت - btnPause

زر فتح الملفات - btnOpen

زر التحكم بالألبومات - btnAlbumes

شريط التنقل عبر الملف - barMove

شريط التحكم بالصوت - barVol

صندوق طريقة التشغيل - boxMthd

ملاحظة عند اختيار صندوق طريقة التشغيل تأكد أنك أختر الصندوق وليس أحد مكوناته حتى تظهر المربعات حول الصندوق:

ومن نافذة الألبومات قم بتغيير أسماء الكائنات أيضاً:

زر ألبوم جديد - btnNew

زر حفظ الألبوم - btnSave

زر حذف الألبوم - btnDelete

زر إضافة ملفات - btnAdd

زر حذف ملف - btnRemove

زر نقل الملف للأعلى - btnUp

زر نقل الملف للأسفل - btnDown

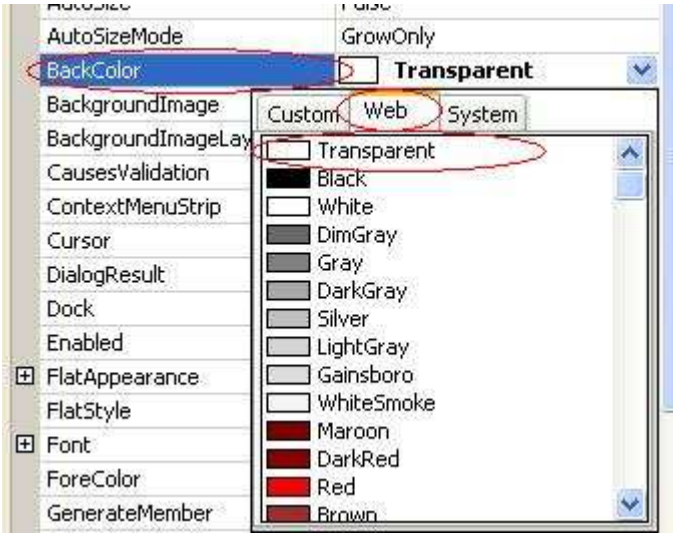
زر موافق - btnOk

زر إلغاء الأمر - btnCancel

تأكد عند نسخ الأسماء أنك نسخت الكلمة فقط دون فراغات في البداية أو النهاية .تغيير الأسماء سيساعدنا كثيرا في التعامل مع الكائنات بواسطة أسمائها .

مثلاً للتعامل مع كائن زر التشغيل من الكود نستدعيه باسمه btnPlay و للتعامل مع زر الإيقاف btnStop وهكذا ..

الآن أفتح تصميم النافذة الرئيسية و أختر كائن التشغيل ألتتابعي و غير الخاصية BackColor إلى Trasparent :

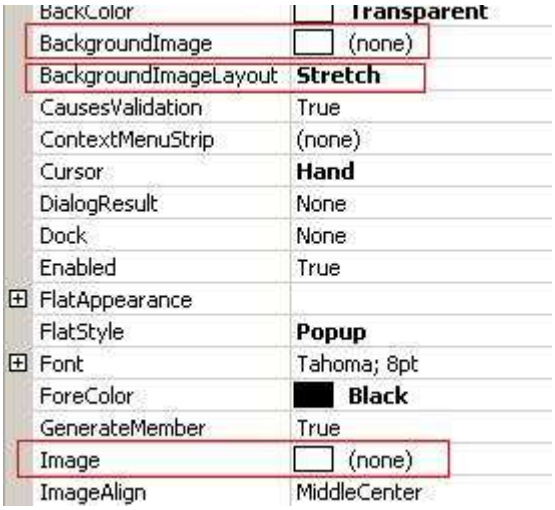


طبق نفس الخطوة السابقة لكل من كائن التشغيل العشوائي , لافتة طول الملف والوقت الحالي للتشغيل, صندوق طريقة التشغيل , جميع أزرار التحكم - تشغيل , إيقاف , تقديم....

أختر زر التشغيل وغير خاصية **size** إلى **40;40** وكذلك لجميع أزرار التحكم

وغير خاصية **BackgroundImageLayout** إلى **Streatch** لجميع أزرار التحكم

ثم أحذف القيمة الموجودة أمام خاصية **Image** وخاصية **BackgroundImage** لجميع أزرار التحكم:



ليصبح شكل النافذة هكذا:

أفتح تصميم نافذة الألبومات وكرر نفس الخطوة على جميع أزرار التحكم هناك إذا أردت إضافة صور لها
أما إذا أردت أن تبقى أزرار نافذة الألبومات عادية فلا تغير شيء من خصائصها
أختر كائن حاضن الفيديو وغير خاصية BackgroundImageLayuot إلىStreatch
الآن أفتح المجلد Debug الموجود داخل المجلد bin داخل مجلد المشروع
وأضف مجلد جديد بهذا الاسم:

كود:

```
Styles
```

هذا هو المجلد الذي سيحتفظ بجميع مجلدات الواجهات المختلفة

أفتح تصميم النافذة الرئيسية وأضف كائن القائمة المنسدلة:

غير خاصية Name فيه إلى sc وخاصية Text إلى "أختر الستايل"

تحميل أسماء الستايلات

نحتاج إلى طريقة لتحميل أسماء الستايلات إلى قائمة اختيار الستايلsc, وكما قلنا سابقاً فإن كل ستايل هو عبارة عن مجلد داخل مجلد Styles, وبالتالي فإن الستايلات المتوفرة هي جميع المجلدات داخل مجلد Style .
وهنا سنستخدم دالة لإيجاد المجلدات الفرعية دخل مجلد رئيسي .
الدالة هي System.IO.Directory.GetDirectories :
وهذه الدالة تستقبل مدخل واحد من نوع نص يمثل مسار المجلد الذي نريد استخراج مجلداته الفرعية, ولأنها دالة وليست إجراء فإنها ترجع بقيمة , هذه القيمة من نوع مصفوفة نصوص تمثل مسارات المجلدات الفرعية مثلاً:

كود:

```
String  subfld;
subfld=System.IO.Directory.GetDirectories(@"c:\");
```

السطر الأول تعريف مصفوفة نصوص , والسطر الثاني إسناد ناتج الدالة للمصفوفةsubfld
والتي تحمل عناصر من نوع نصوص كل عنصر يمثل مجلد موجود داخل القرص المحلي C
التعامل مع المصفوفات شرحناه مسبقاً في الدرس الثامن .الكود السابق ينتج المجلدات في القرص c ولكن لا نريد الحصول على المجلدات في القرص c بل نريد المجلدات في مجلد Style .
ونعلم أن هذا المجلد موجود في مسار ملف البرنامج , ومن الخطأ أن نكتب التعليمة هكذا:

كود:

```
String  subfld;
subfld=System.IO.Directory.GetDirectories(@"c:\cs\project\bin\debug\styles");
```

لأنه لو تغير مجلد البرنامج فلن تعمل التعليمة السابقة لأن مكان المجلد Style تغير لذلك سنستخدم متغير جديد Application.StartupPathيرجع قيمة من نوع نص تمثل مسار البرنامج , وعند الحصول على مسار البرنامج نضيف إليه أسم مجلدStyle, وبالتالي نحصل على مسار مجلد الستايلات مهما تغير مكان البرنامج:

كود:

```
String  subfld;
subfld=System.IO.Directory.GetDirectories(Application.StartupPath + @"\Styles");
```

التعليمة السابقة ستعمل دائما بغض النظر عن مسار البرنامج و هذا ما نحتاجه ,نحتاج الآن إلى دوارة for لتمر على جميع عناصر المصفوفة subfld وتضيف كل عنصر إلى قائمة اختيار الستايل, لكن هناك مشكلة المصفوفة subfldتحتوي على نصوص تمثل المسار الكامل لمجلد الستايل مثل:

كود:

```
c:\cs\project\bin\debug\styles\style1
```

ونحن نريد فقط أسم الستايل فقط في قائمة اختيار الستايلات:

كود:

```
style1
```

لحل هذه المشكلة سنستخدم خاصية التبديل في النص , وهي دالة تقوم بالبحث عن كلمة محددة وتبديلها بكلمة أخرى
التعليمة هي **Replace** وتوجد في أي متغير من نوع نص مثلاً:

كود:

```
string wrd;
wrd="this is a test";
wrd.Replace("test","good test");
```

السطر الأخير استدعاء لتعليمة الاستبدال حيث تبحث عن كلمة **test** في المتغير **wrd** وتحولها إلى**good test**
وتصبح الجملة داخل متغير: **wrd**

كود:

```
this is a good test
```

سنستخدم هذه الخاصية لمسح مسار مجلد الستايلات في المصفوفة **subfld** والإبقاء فقط على أسم مجلد الستايل
مثلاً لو كانت قيمة العنصر الأول من المصفوفة هي:

كود:

```
c:\cs\project\bin\debug\styles\style1
```

سنحصل على أسم المجلد فقط بهذه التعليمة:

كود:

```
subfld[0].Replace("c:\cs\project\bin\debug\styles\","");
```

حيث أن التعليمة السابقة تأخذ الجزء الغير ضروري من المسار وتكتب بدلاً منه فراغ وبالتالي يبقى فقط**style1**
نلاحظ أن الجزء المحذوف من المسار هو نفسه المسار الناتج من تعليمة:

كود:

```
Application.StartupPath + @"\Styles"
```

لذلك ولكي تنجح التعليمة مهما تغير مسار البرنامج فسنغيرها كما فعلنا سابقاً:

كود:

```
subfld[0].Replace(Application.StartupPath + @"\Styles\","");
```

التعليمة السابقة سنكررها لكل عنصر من عناصر المصفوفة **subfld** بإستخدام دوارة: **for**

كود:

```
        for (int i = 0; i < subfld.Length; i++)
        {
            sc.Items.Add(subfld[i].Replace(Application.StartupPath + @"\Styles\", ""));
        }
```

الدوارة السابقة تكرر العملية من الصفر حتى عدد عناصر المصفوفة subfld.Length كما عرفنا في الدرس الثالث .
بقي أن نجمع كود إجراء قراءة الستايلات.

إجراء قراءة الستايلات

أذهب إلى كود التطبيق وأنسخ هذا الكود وتأكد أن يكون خارج أي إجراء آخر:

كود:

```
    void ReadAllStyles()
    {
        try
        {
            string subfld;
            subfld = System.IO.Directory.GetDirectories(Application.StartupPath +
@"\Styles");
            for (int i = 0; i < subfld.Length; i++)
            {
                sc.Items.Add(subfld[i].Replace(Application.StartupPath + @"\Styles\",
""));
            }
        }
        catch
        {
            MessageBox.Show("حدث خطأ أثناء محاولة قراءة الستايلات");
        }
    }
```

السطر الأول تعريف مصفوفة من نوع نصوص , لاحظ القوسين بعد كلمة string للدلالة على أنها مصفوفة متغيرات وليست متغير واحد .
السطر الثاني لتعبئة المصفوفة subfld بجميع المجلدات الفرعية لمجلد الستايلات.
والسطر داخل دوارة for لإضافة أسم مجلد الستايل إلى قائمة اختيار الستايلات sc بعد أن نحذف منه المسار الغير ضروري .
بقي أن نجد المكان المناسب لاستدعاء الإجراء السابق , وأفضل مكان هو عند تحميل النافذة الرئيسية .
أذهب إلى تصميم النافذة الرئيسية و أنقر مزدوجاً على أي مكان فارغ فيها.

ستنتقل إلى إجراء حدث تحميل النافذة:

كود:

```
        void MainFormLoad(object sender, System.EventArgs e)
        {
                doc.Load(Application.StartupPath + "//albumes.xml");
XmlNode albumesnd = doc["albumes"];
XmlNode alnd = albumesnd.FirstChild;
while (alnd != null)
{
   ac.Items.Add(alnd.Name);
   alnd = alnd.NextSibling;
}
        }
```

أضف استدعاء الإجراء **ReadAllStyles** قبل أخر حاصرة ليصبح الإجراء هكذا:

كود:

```
        void MainFormLoad(object sender, System.EventArgs e)
        {
                doc.Load(Application.StartupPath + "//albumes.xml");
XmlNode albumesnd = doc["albumes"];
XmlNode alnd = albumesnd.FirstChild;
while (alnd != null)
{
   ac.Items.Add(alnd.Name);
   alnd = alnd.NextSibling;
}
ReadAllStyles();
        }
```

إجراء خزن الستايل المختار

عند اختيار أحد عناصر قائمة اختيار الستايلات المفروض أن يقوم البرنامج بتحميل ألوان وصور الستايل المختار ويجب أن يتذكر البرنامج الستايل المحدد حتى لو أغلقنا البرنامج و أعدنا تشغيل الكمبيوتر .
لذلك يجب أن نجد طريقة لخزن الستايل المختار بحيث يعرف البرنامج ما هو الستايل الذي استخدمناه حتى لو أعدنا تشغيل الكمبيوتر .
لذلك يجب أن نكون ملف نصي في مجلد Styles نكتب بداخله المجلد المختار ,وعند تشغيل البرنامج مرة أخرى نقرأ الملف المختار من هذا الملف , لنسمي هذا الملفCurrentStyle.txt :
ولكتابة ملفات نصية سنستخدم إجراء جديد System.IO.File.WriteAllText يستقبل هذا الإجراء مدخلين من نوع نص الأول يمثل مسار الملف المراد كتابته ,والمدخل الثاني يمثل النص المراد كتابته في الملف , مثلاً:

كود:

```
System.IO.File.WriteAllText(@"c:\newtxt.txt","السلام عليكم");
```

الكود السابق يقوم بإنشاء ملف باسم new.txt في القرص المحلي c ويكتب داخلة جملة "السلام عليكم"
بنفس الطريقة سنقوم بإنشاء ملف نصي باسم CurrentStyle.txt لخزن الستايل المختار .لكن المسار هنا سيكون مجلد الستايلات , Styles ونستدعي إجراء خزن الستايل المختار عند اختيار عنصر من عناصر قائمة اختيار الستايلات .

أفتح تصميم النافذة الرئيسية وأنقر مزدوجاً على قائمة اختيار الستايلات وفي الكود أكتب تعليمة خزن الملف المختار:

كود:

```
System.IO.File.WriteAllText(Application.StartupPath + @"\Styles\CurrentStyle.txt",
sc.Text);
```

التعليمة السابقة ستقوم بإنشاء ملف نصي باسم CurrentStyle.txt في مجلد الستايلات وتكتب داخله النص الموجود في قائمة اختيار الستايلاتsc .

إجراء تحميل الستايل المختار

كما قلنا فإن لكل ستايل ملف XML باسم Style.XML موجود داخل مجلد الستايل , وعند اختيار ستايل من القائمةsc .المفروض أن يقوم البرنامج بقراءة ملف XML الموجود داخل مجلد الستايل وتحميل قيم الألوان والصور من داخله .
أولاً يجب أن نقرأ قيمة الستايل المختار المخزنة في الملف النصي CurrentStyle.txt حتى نعرف أي ستايل يجب تحميله ,وللقراءة من الملف النصي سنستخدم إجراء جديد يستقبل مدخل واحد يمثل مسار الملف المراد قراءته وينتج قيمة من نوع نص تمثل محتويات الملف:

كود:

```
System.IO.File.ReadAllText(@"c:\newtxt.txt");
```

الإجراء السابق ينتج قيمة من نوع نص تمثل محتويات الملف newtxt.txt الموجود في القرص المحلي C
وبنفس الطريقة سنقوم بقراءة الستايل المخزن في الملف: CurrentStyle.txt

كود:

```
        string currentstyle;
        currentstyle= System.IO.File.ReadAllText(Application.StartupPath +
@"\Styles\CurrentStyle.txt");
```

عرفنا متغير من نوع نص currentstyle وأسندنا له ما داخل الملف CurrentStyle.txt وهو أسم الستايل المختار .
بمعرفة أسم الستايل المختار يمكن معرفة مكان ملف XML الذي يجب تحميله:

كود:

```
Application.StartupPath + @"\Styles\" + currentstyle + @"\Style.XML"
```

هذا هو مسار ملف XML للستايل المختار حيث أخذنا مسار البرنامج وأضفنا له مجلد Styles ثم أسم الستايل المختار currentstyle ثم أسم ملف XML ,لقراءة القيم من الملف نحتاج لتعريف كائن من نوع XMLDocument كما فعلنا في الدرس الحادي عشر ولنسمه: StyleDoc

كود:

```
        XmlDocument StyleDoc = new XmlDocument();
```

ولتحميل قيم الملف المطلوب يجب استدعاء إجراء Load بواسطة مسار الملف

كود:

```
        StyleDoc.Load(Application.StartupPath + @"\Styles\" + currentstyle +
@"\Style.XML");
```

برنامج صنع الواجهات يكتب الألوان والصور في ملف XML بدلالة كلمات مفتاحيه كل كلمة يخزن فيها قيمة وتسمى الكلمات المفتاحين بالمفاتيح لأنها تعمل كدليل على القيمة , مثلاً المفتاح MainBackColor هو دليل قيمة تمثل لون خلفية النافذة الرئيسية , ولا يمكن قراءة القيمة من ملف XML إلا بمعرفة مفتاح القيمة وهو MainBackColor

وهذه هي جميع مفاتيح القيم ودلالاتها:

لون خلفية النافذة الرئيسية - MainBackColor
لون خلفية قائمة التشغيل - PlayListBackColor
لون الخط في قائمة التشغيل - PlayListFontColor
لون خلفية قائمة اختيار الألبوم في النافذة الرئيسية - MainSelectAlbumeBackColor
لون الخط في قائمة اختيار الألبوم في النافذة الرئيسية - MainSelectAlbumeFontColor
لون خلفية قائمة اختيار الستايل - MainSelectStyleBackColor
لون الخط في قائمة اختيار الستايل - MainSelectStyleFontColor
لون الخط في لافتة طول الملف والوقت الحالي للتشغيل - TimeFontColor
لون الخط في صندوق طريقة التشغيل - MethodFontColor
لون خلفية شريط التنقل - MoveBarColor
لون خلفية شريط التحكم بالصوت - SoundBarColor
لون الخط في مربع كتم الصوت - MuteFontColor
خلفية النافذة الرئيسية - MainBackGroundImage
خلفية زر الإيقاف - StopImage
خلفية زر التشغيل - PlayImage
خلفية زر التسجيل - RecordImage
خلفية زر التقديم - ForwardImage
خلفية زر التأخير - BackwardImage
خلفية زر إيقاف التسجيل - StopRecordingImage
خلفية زر فتح الملفات - OpenImage
خلفية زر الإيقاف المؤقت - PauseImage
خلفية زر ملئ الشاشة - FullScreenImage
خلفية حاضن الفيديو - VedioImage

لون خلفية نافذة الألبومات - AlbumesBackColor
لون الخط في قائمة الملفات في نافذة الألبومات - FilesFontColor
لون خلفية قائمة اختيار الألبوم في نافذة الألبومات - AlbumesSelectAlbumeBackColor
لون الخط في قائمة اختيار الألبوم في نافذة الألبومات - AlbumesSelectAlbumeFontColor
خلفية نافذة الألبومات - AlbumesBackGroundImage
خلفية زر الإضافة في نافذة الألبومات - AddImage
خلفية زر حذف الملفات - DeleteImage
خلفية زر نقل الملف إلى أعلى - UpImage
خلفية زر نقل الملف إلى أسفل - DownImage
خلفية زر ألبوم جديد - NewAlbumeImage
خلفية زر حفظ الألبوم - SaveAlbumeImage
خلفية زر حذف الألبوم - DeleteAlbumeImage
خلفية زر موافق - OkImage
خلفية زر إلغاء الأمر - CancelImage
خلفية قائمة الملفات - FilesImage

وهذه القيم كلها مخزنة في عقدة رئيسية أسمها **Style** وبالتالي لقراءة هذه القيم يجب تعريف متغير من نوع عقدة **XMLNode**لخزن العقدة الرئيسية:

كود:

```
XmlNode StyleNode = StyleDoc["Style"];
```

الآن يمكن استدعاء أي قيمة من خلال العقدة **StyleNode** فمثلاً لاستدعاء قيمة لون خلفية النافذة الرئيسية **MainBackColor**

كود:

```
StyleNode["MainBackColor"].InnerText
```

التعليمة السابقة ستنتج نص يمثل قيمة لون خلفية النافذة الرئيسية ولتغيير لون خلفية النافذة الرئيسية:

كود:

```
this.BackColor = StyleNode["MainBackColor"].InnerText;
```

التعليمة السابقة خاطئة لأنه من المفترض أن يكون ناتج الطرف الأيمن من المعادلة هو كائن من نوع لون
لكن في التعليمة السابقة فإن القيمة من ملف **XML** هي من نوع نص يمثل رقم اللون وبالتالي يجب تغيير هذا النص إلى لون كيف ؟

لدينا دالة لتغيير رقم اللون إلى كائن لون:

كود:

```
Color.FromArgb(25698);
```

الدالة السابقة تحول الرقم **25698** إلى لون , وبنفس الطريقة سنغير القيمة الناتجة من ملف **XML** إلى لون:

كود:

```
this.BackColor = Color.FromArgb(StyleNode["MainBackColor"].InnerText);
```

التعليمة السابقة أيضاً خاطئة لأن الدالة **Color.FromArgb** يجب أن تستقبل مدخل من نوع رقم , بينما القيمة الناتجة من ملف **XML** تمثل نص ,وبالتالي يجب أيضاً تحويلها إلى رقم بواسطة كائن التحويل **Convert.ToInt32**وتصبح التعليمة الصحيحة:

كود:

```
this.BackColor =
Color.FromArgb(Convert.ToInt32(StyleNode["MainBackColor"].InnerText));
```

وهكذا لجميع قيم الألوان , فمثلا لتغيير خلفية قائمة التشغيل: **pl**
نرجع لجدول الكلمات المفتاحية ومدلولاتها ونبحث عن مفتاح خلفية قائمة التشغيل , سنجده **PlayListBackColor**

كود:

```
pl.BackColor =
Color.FromArgb(Convert.ToInt32(StyleNode["PlayListBackColor"].InnerText));
```

وهكذا يجب كتابة تعليمات تحويل لجميع الكلمات المفتاحية حسب التسميات في التطبيق لديك, ولتغيير خلفيات الأزرار أو الكائنات سنستخدم الخاصية **Image.FromFile** والتي تنتج متغير من نوع صورة وتستقبل مدخل واحد يمثل مسار ملف الصورة .
مثلاً لتغيير خلفية حاضن الفيديو نتعامل معه باسمه وهو : **p1**

كود:

```
p1.BackgroundImage=Image.FromFile(@"c:\back.jpg");
```

التعليمة السابقة تقوم بتغيير خلفية كائن الحاضن إلى الصورة **back.jpg** الموجودة في القرص المحلي **C** ولتحميل القيمة الموجودة داخل ملف **XML** يجب أن نعرف المسار الكامل لملفات الصور, حيث أن ملف **XML** يحتوي فقط على أسم الصورة ولا يحتوي على مسارها كاملاً , ونعلم أن مسارها هو نفس مسار مجلد الستايل وبالتالي يجب إنشاء متغير من نوع نص لخزن مسار الستايل الحالي الذي يحتوي على الصور:

كود:

```
        string stylepath;
            stylepath = Application.StartupPath + @"\Styles\" + currentstyle + @"\";
```

أضفنا إلى مسار مجلد البرنامج مجلد الستايلات ثم أضفنا إسم الملف المختار **currentstyle** وبالتالي فإن المتغير الجديد **stylepath** سيحتوي على المسار الكامل للصور في الستايل المختار ,بقي أن نجد أسم الصورة من ملف **XML** ,نبحث عن الكلمة المفتاحية لقيمة خلفية كائن الفيديو وهي**VedioImage**

كود:

```
p1.BackgroundImage = Image.FromFile(stylepath +
StyleNode["VediotImage"].InnerText);
```

ولكن أحياناً لا يحتوي ملف **XML** على قيمة إذا أدرنا أن تكون الخلفية فارغة وبالتالي الكود السابق سينتج خطأ لانه لا يوجد قمية عند المفتاح**VedioImage**

لذلك يجب وضع التعليمة السابقة ضمن جملة: **Try Catch**

كود:

```
        try
        {
            p1.BackgroundImage = Image.FromFile(stylepath +
StyleNode["VediotImage"].InnerText);
        }
        catch
        {
        }
```

ولاختصار المساحة يمكن كتابة الكود في سطر واحد:

كود:

```
        try { p1.BackgroundImage = Image.FromFile(stylepath +
StyleNode["VediotImage"].InnerText);}catch{}
```

وهكذا لجميع خلفيات الأزرار و الكائنات الأخرى , مثلاً زر التشغيل الذي إسمه**btnPlay**
نبحث عن مفتاح خلفية زر التشغيل في جدول المفاتيح سنجده: **PlayImage**

كود:

```
        try { btnPlay.BackgroundImage = Image.FromFile(stylepath +
StyleNode["PlayImage"].InnerText);}catch{}
```

و زر الإيقاف الذي أسمه **btnStop** سنجد أن مفتاح خلفية زر الإيقاف هو: **StopImage**

كود:

```
        try { btnStop.BackgroundImage = Image.FromFile(stylepath +
StyleNode["StopImage"].InnerText);}catch{}
```

وهكذا لجميع الكلمات المفتاحية في الجدول السابق , وبتجميع الكود في إجراء واحد يستخدم لتحميل قيم الستايل المختار ولنسمي هذا الإجراء: **LoadStyle**

أفتح كود النافذة الرئيسية وأنسخ هذا الإجراء وتأكد أنه خارج أي إجراء آخر:

كود:

```
    void LoadStyle()
    {
        try
        {
            string currentstyle;
            currentstyle    =    System.IO.File.ReadAllText(Application.StartupPath    +
@"\Styles\CurrentStyle.txt");
            sc.Text = currentstyle;

            XmlDocument StyleDoc = new XmlDocument();
            StyleDoc.Load(Application.StartupPath  +  @"\Styles\"  +  currentstyle  +
@"\Style.XML");
            XmlNode StyleNode = StyleDoc["Style"];


            this.BackColor                                                    =
Color.FromArgb(Convert.ToInt32(StyleNode["MainBackColor"].InnerText));
            pl.BackColor                                                      =
Color.FromArgb(Convert.ToInt32(StyleNode["PlayListBackColor"].InnerText));
            pl.ForeColor                                                      =
Color.FromArgb(Convert.ToInt32(StyleNode["PlayListFontColor"].InnerText));
```

```
            ac.BackColor =
Color.FromArgb(Convert.ToInt32(StyleNode["MainSelectAlbumeBackColor"].InnerT
ext));
            ac.ForeColor =
Color.FromArgb(Convert.ToInt32(StyleNode["MainSelectAlbumeFontColor"].InnerTe
xt));
            sc.BackColor =
Color.FromArgb(Convert.ToInt32(StyleNode["MainSelectStyleBackColor"].InnerText)
);
            sc.ForeColor =
Color.FromArgb(Convert.ToInt32(StyleNode["MainSelectStyleFontColor"].InnerText)
);
            len.ForeColor =
Color.FromArgb(Convert.ToInt32(StyleNode["TimeFontColor"].InnerText));
            boxMthd.ForeColor=
Color.FromArgb(Convert.ToInt32(StyleNode["MethodFontColor"].InnerText));
            serbtn.ForeColor=
Color.FromArgb(Convert.ToInt32(StyleNode["MethodFontColor"].InnerText));
            rndbtn.ForeColor=
Color.FromArgb(Convert.ToInt32(StyleNode["MethodFontColor"].InnerText));
            barMove.BackColor=
Color.FromArgb(Convert.ToInt32(StyleNode["MoveBarColor"].InnerText));

barVol.BackColor=Color.FromArgb(Convert.ToInt32(StyleNode["SoundBarColor"].In
nerText));

mute.ForeColor=Color.FromArgb(Convert.ToInt32(StyleNode["MuteFontColor"].Inne
rText));



            string stylepath;
            stylepath = Application.StartupPath + @"\Styles\" + currentstyle + @"\";

            try { this.BackgroundImage = Image.FromFile(stylepath +
StyleNode["MainBackGroundImage"].InnerText); }catch { }
            try { btnStop.BackgroundImage = Image.FromFile(stylepath +
StyleNode["StopImage"].InnerText); }catch { }
            try { btnPlay.BackgroundImage = Image.FromFile(stylepath +
StyleNode["PlayImage"].InnerText); }catch { }
            try { btnRecord.BackgroundImage = Image.FromFile(stylepath +
StyleNode["RecordImage"].InnerText); }catch { }
            try { btnForward.BackgroundImage = Image.FromFile(stylepath +
StyleNode["ForwardImage"].InnerText); }catch { }
            try { btnStopRecord.BackgroundImage = Image.FromFile(stylepath +
StyleNode["StopRecordingImage"].InnerText); }catch { }
            try { btnBackWard.BackgroundImage = Image.FromFile(stylepath +
StyleNode["BackwardImage"].InnerText); }catch { }
            try { btnOpen.BackgroundImage = Image.FromFile(stylepath +
StyleNode["OpenImage"].InnerText); }catch { }
```

```
            try { btnPause.BackgroundImage = Image.FromFile(stylepath +
StyleNode["PauseImage"].InnerText); }catch { }
            try { btnFullScreen.BackgroundImage = Image.FromFile(stylepath +
StyleNode["FullScreenImage"].InnerText); }catch { }
            try { p1.BackgroundImage = Image.FromFile(stylepath +
StyleNode["VedioImage"].InnerText); } catch { }
        }
        catch
        {
            MessageBox.Show("حدث خطأ أثناء تحميل الستايل");
        }
    }
```

الجزء الأول من الإجراء السابق لقراءة الستايل المختار من ملفCurrentStyle.txt
وإنشاء كائن XMLDocument وتحميل ملف XML للستايل المختار
والجزء الثاني لتحميل ألوان الكائنات كما شرحنا سابقاً
والجزء الثالث لتحميل الصور للكائنات كما شرحنا أيضاً
لاحظ أن خاصية الخلفية للكائنات هي:

كود:

```
BackgroundImage
```

وخاصية لون الخلفية لأي كائن هي:

كود:

```
BackColor
```

وخاصية لون الخط لأي كائن هي:

كود:

```
ForeColor
```

بقي أن نضيف مكان مناسب لاستدعاء هذا الإجراء , وأنسب مكان هو حدث اختيار ستايل من قائمة اختيار الستايلات
أذهب إلى تصميم النافذة الرئيسية و أنقر مزدوجا على قائمة اختيار الستايلات لتذهب إلى كود إجراء الحدث:

كود:

```
        System.IO.File.WriteAllText(Application.StartupPath +
@"\Styles\CurrentStyle.txt", sc.Text);
```

أضف بعده سطر استدعاء إجراء تحميل الستايل ليصبح الكود هكذا:

كود:

```
        System.IO.File.WriteAllText(Application.StartupPath +
@"\Styles\CurrentStyle.txt", sc.Text);
        LoadStyle();
```

يوجد هناك مكان آخر يجب أن نستدعي فيه هذا الإجراء وهو عند بداية تشغيل البرنامج أي عند حدث تحميل النافذة الرئيسية .
أذهب إلى الكود وأنقر مزدوجا في أي مكان فراغ من النفاذة الرئيسية لتذهب إلى الكود , أضف تعليمة الاستدعاء في نهاية الإجراء ليصبح هكذا:

كود:

```
                doc.Load(Application.StartupPath + "//albumes.xml");
        XmlNode albumesnd = doc["albumes"];
        XmlNode alnd = albumesnd.FirstChild;
        while (alnd != null)
        {
            ac.Items.Add(alnd.Name);
            alnd = alnd.NextSibling;
        }
        ReadAllStyles();
        LoadStyle();
```

بهذا أكملنا وظائف تغيير الستايل في النافذة الرئيسية , بقي أن نضيف نفس الإجراء لنافذة الألبومات:
أفتح كود نافذة الألبومات وأنسخ الكود التالي وتأكد أنه خارج أي إجراء آخر:

كود:

```
    void LoadStyle()
    {
        try
        {
            string currentstyle;
            currentstyle = System.IO.File.ReadAllText(Application.StartupPath +
@"\Styles\CurrentStyle.txt");

            XmlDocument StyleDoc = new XmlDocument();
            StyleDoc.Load(Application.StartupPath + @"\Styles\" + currentstyle +
@"\Style.XML");
            XmlNode StyleNode = StyleDoc["Style"];


            this.BackColor =
Color.FromArgb(Convert.ToInt32(StyleNode["AlbumesBackColor"].InnerText));
            lv.BackColor =
Color.FromArgb(Convert.ToInt32(StyleNode["FilesFontColor"].InnerText));
            ac.BackColor =
Color.FromArgb(Convert.ToInt32(StyleNode["AlbumesSelectAlbumeBackColor"].Inn
erText));
            ac.BackColor =
Color.FromArgb(Convert.ToInt32(StyleNode["AlbumesSelectAlbumeFontColor"].Inn
erText));


            string stylepath;
            stylepath = Application.StartupPath + @"\Styles\" + currentstyle + @"\";
```

```
            try { this.BackgroundImage = Image.FromFile(stylepath +
StyleNode["AlbumesBackGroundImage"].InnerText); }catch { }
            try { btnAdd.BackgroundImage = Image.FromFile(stylepath +
StyleNode["AddImage"].InnerText); }catch { }
            try { btnRemove.BackgroundImage = Image.FromFile(stylepath +
StyleNode["DeleteImage"].InnerText); }catch { }
            try { btnUp.BackgroundImage = Image.FromFile(stylepath +
StyleNode["UpImage"].InnerText); }catch { }
            try { btnDown.BackgroundImage = Image.FromFile(stylepath +
StyleNode["DownImage"].InnerText); }catch { }
            try { btnNew.BackgroundImage = Image.FromFile(stylepath +
StyleNode["NewAlbumeImage"].InnerText); }catch { }
            try { btnSave.BackgroundImage = Image.FromFile(stylepath +
StyleNode["SaveAlbumeImage"].InnerText); } catch { }
            try { btnDelete.BackgroundImage = Image.FromFile(stylepath +
StyleNode["DeleteAlbumeImage"].InnerText); }catch { }
            try { btnOk.BackgroundImage = Image.FromFile(stylepath +
StyleNode["OkImage"].InnerText); }catch { }
            try { btnCancel.BackgroundImage = Image.FromFile(stylepath +
StyleNode["CancelImage"].InnerText); }catch { }
            try { lv.BackgroundImage = Image.FromFile(stylepath +
StyleNode["FilesImage"].InnerText); }catch { }
        }
        catch
        {
            MessageBox.Show("حدث خطأ أثناء تحميل الستايل");
        }
    }
```

هذا الإجراء يشبه كثيراً نفس الإجراء في النافذة الرئيسية إلا أنه هنا للتعامل مع كائنات نافذة الألبومات .هنا أيضاً يجب أن نجد مكان مناسب لاستدعاء الإجراء وهو عند تحميل نافذة الألبومات .

أذهب إلى تصميم نافذة الألبومات وأنقر مزدوجاً على أي مكان فارغ فيها ,وأضف إلى الكود تعليمة استدعاء الإجراء ليصبح هكذا:

كود:

```
                    FillView();
                    FillCombo();
        LoadStyle();
```

واجب الدرس السابع عشر:

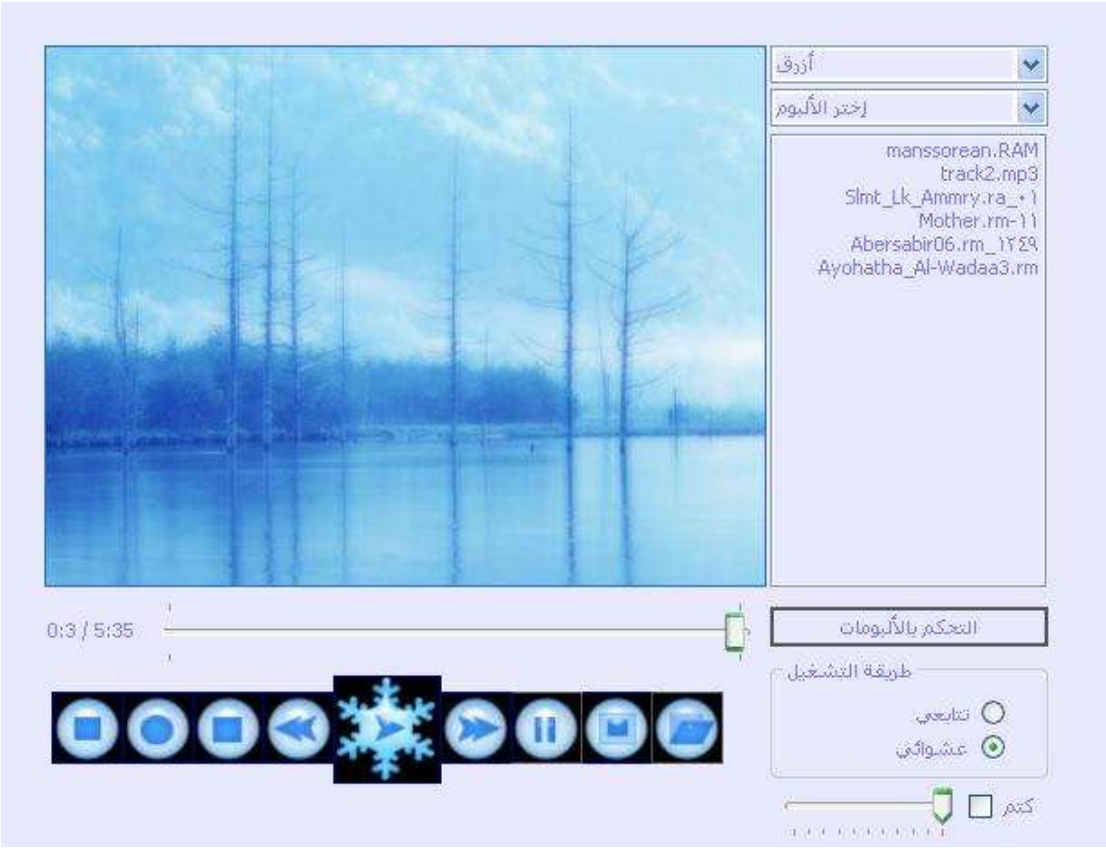
صمم ستايل جديد بواسطة برنامج صنع الستايلات وطبقة في البرنامج

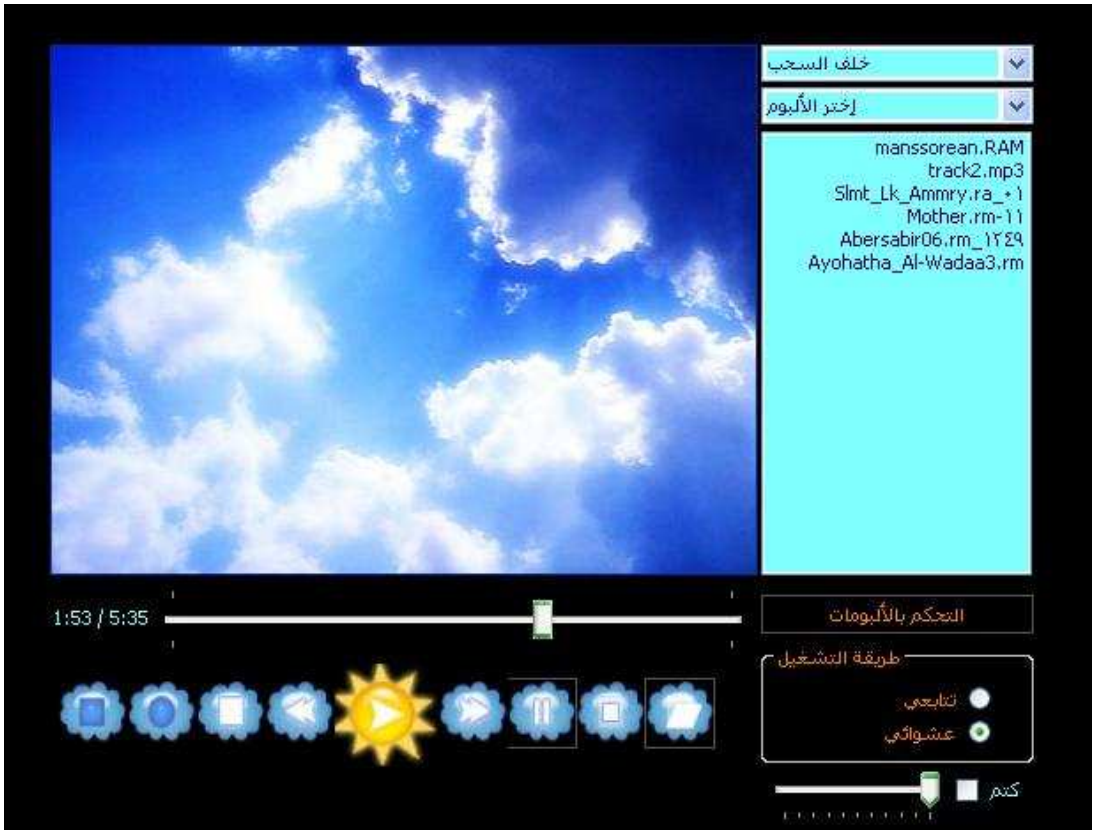
ملاحظة : برنامج مصمم الستايلات ينتج مجلد بنفس أسم الستايل الذي اخترته ,قم بنسخ المجلد كامل إلى مجلد Stylesفي مسار البرنامج , وعند تشغيل البرنامج سيضاف الستايل إلى القائمة المنسدلة.

أرسل التطبيق.

.نهاية الدرس السابع عشر

هذه عدد من الستايلات الجاهزة يتم استخدامها بنسخ جميع المجلدات إلى مجلد Styles






بحيرة ألوان
إختر الألبوم
manssorean.RAM
track2.mp3
Slmt_Lk_Ammry.ra_٠١
Mother.rm-١١
Abersabir06.rm_١٢٢٩
Ayohatha_Al-Wadaa3.rm
1:16 / 5:35
التحكم بالألبومات
طريقة التشغيل
تتابعي
عشوائي
كتم


أوراق الورد
إختر الألبوم
manssorean.RAM
track2.mp3
Slmt_Lk_Ammry.ra_٠١
Mother.rm-١١
Abersabir06.rm_١٢٢٩
Ayohatha_Al-Wadaa3.rm
0:40 / 5:35
التحكم بالألبومات
طريقة التشغيل
تتابعي
عشوائي
كتم


رياضي
إختر الألبوم
manssorean.RAM
track2.mp3
Slmt_Lk_Ammry.ra_٠١
Mother.rm-١١
Abersabir06.rm_١٢٤٩
Ayohatha_Al-Wadaa3.rm
2:34 / 5:35
التحكم بالألبومات
طريقة التشغيل
تتابعي
عشوائي
كتم


زهور
إختر الألبوم
manssorean.RAM
track2.mp3
Slmt_Lk_Ammry.ra_٠١
Mother.rm-١١
Abersabir06.rm_١٢٤٩
Ayohatha_Al-Wadaa3.rm
Beautiful Day
التحكم بالألبومات
طريقة التشغيل
تتابعي
عشوائي
كتم


ساكورا
إختر الألبوم
manssorean.RAM
track2.mp3
Slmt_Lk_Ammry.ra_٠١
Mother.rm-١١
Abersabir06.rm_١٢٤٩
Ayohatha_Al-Wadaa3.rm
التحكم بالألبومات
طريقة التشغيل
تتابعي
عشوائي
كتم


قلوب
إختر الألبوم
manssorean.RAM
track2.mp3
Slmt_Lk_Ammry.ra_٠١
Mother.rm-١١
Abersabir06.rm_١٢٤٩
Ayohatha_Al-Wadaa3.rm
4:10 / 5:35
التحكم بالألبومات
طريقة التشغيل
تتابعي
عشوائي
كتم


مرآه
إختر الألبوم
manssorean.RAM
track2.mp3
Slmt_Lk_Ammry.ra_٠١
Mother.rm-١١
Abersabir06.rm_١٢٤٩
Ayohatha_Al-Wadaa3.rm
4:36 / 5:35
التحكم بالألبومات
طريقة التشغيل
تتابعي
عشوائي
كتم


ميري
إختر الألبوم
manssorean.RAM
track2.mp3
Slmt_Lk_Ammry.ra_٠١
Mother.rm-١١
Abersabir06.rm_١٢٤٩
Ayohatha_Al-Wadaa3.rm
5:8 / 5:35
التحكم بالألبومات
طريقة التشغيل
تتابعي
عشوائي
كتم

ملف الستايلات مرفق
مع الكتاب .....

بالتوفيق للجميع

من رأى فينا خطأً .. فلينصحنا .. فنحن بشر

في حال وجود خطا في التنسيق

Y85.fox@hotmail.com