بناء حزمة تثبيت محمية برقم تسلسلي باستخدام VSI, Orca, C++ or Delphi

- عصام علي

http://issamsoft.com

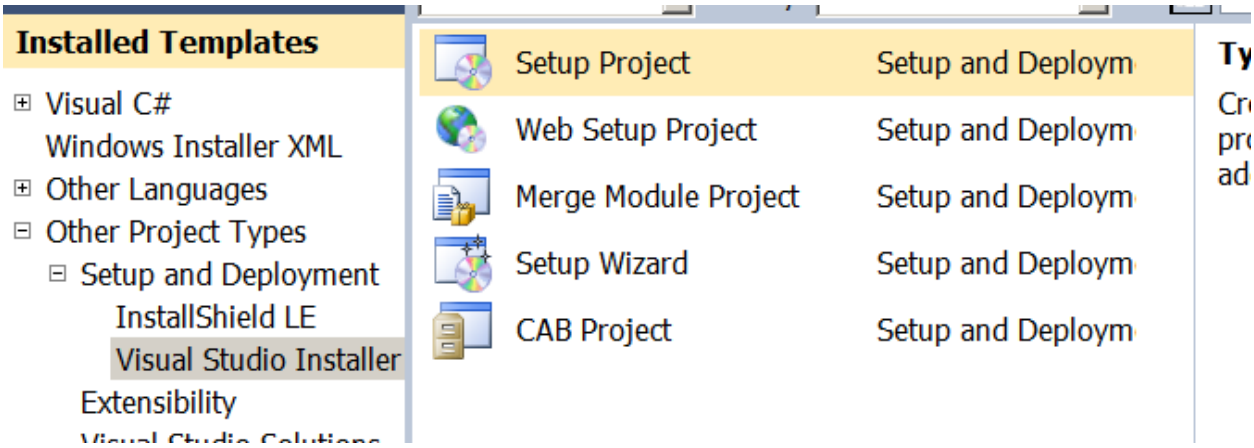
بناء حزمة تثبيت محمية برقم تسلسلي باستخدام VSI, Orca, C++ or Delphi

مقدمة

تستطيع من خلال فيجوال ستوديو بناء حزم تثبيت لبرامجك باستخدام Visual Studio Installer او أختصاراً VSI وتستطيع من خلال هذه الأداة أيضاً أن تضيف مربع حوار إلى حزمة التثبيت لتسجيل بيانات المستخدم: الاسم، الشركة، والرقم التسلسلي (السريال). يقدم VSI إمكانيات محدودة لعملية التحقق من الرقم التسلسلي مما سيجبرك على استخدام نماذج أرقام تسلسلية Serial Number Templates سهلة التخمين و ليست بحاجة حتى إلى برامج كراك Crack لكسرها في كثير من الأحيان. الحل الأمثل هنا هو الاستغناء عن خوارزمية التحقق المدمجة في VSI واستخدام خوارزمياتك الخاصة عن طريق مكاتب خارجية dll تستطيع كتابتها بأي لغة أصلية native مثل دلفي أو سي بلس بلس -سأعرض مثالاً باللغتين- وعندها ستكون حدودك هي السماء في بناء نماذج أرقام تسلسلية لحماية برامجك. وجدت مقالات قليلة تشرح هذه الطريقة وبالاضافة إلى قلتها فالبعض منها فيه أخطاء قد تؤدي إلى توجيهك باتجاه خاطئ وإضاعة وقتك الثمين خصوصاً إذا كنت جديداً على هذا المجال. لذلك قررت كتابة هذا المقال بالعربية و الانكليزية أيضاً راجياً أن يكون فيه شيء من الفائدة لمن يهتم بهذا الموضوع.

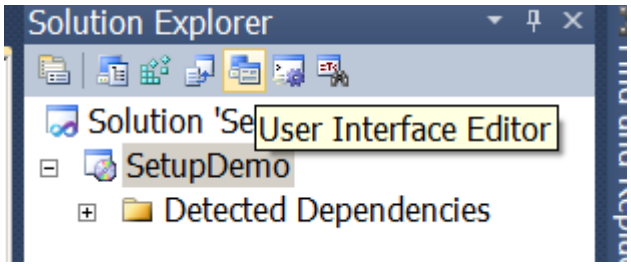
بناء حزمة تثبيت بسيطة بواسطة VSI

سأحاول أن تكون الأمثلة في هذا المقال بسيطة وواضحة قدر الإمكان. لذلك سنقوم بإنشاء مشروع حزمة تثبيت فارغ empty setup project (لا شيء ليتم تثبيته) لأننا سنركز فقط على تقنية التحقق من الرقم التسلسلي.
في فيجوال ستوديو 2010 انشئ مشروع جديد و اختر نوع المشروع Visual Studio Installer ثم اختر Setup Project وسمه 'SetupDemo'.

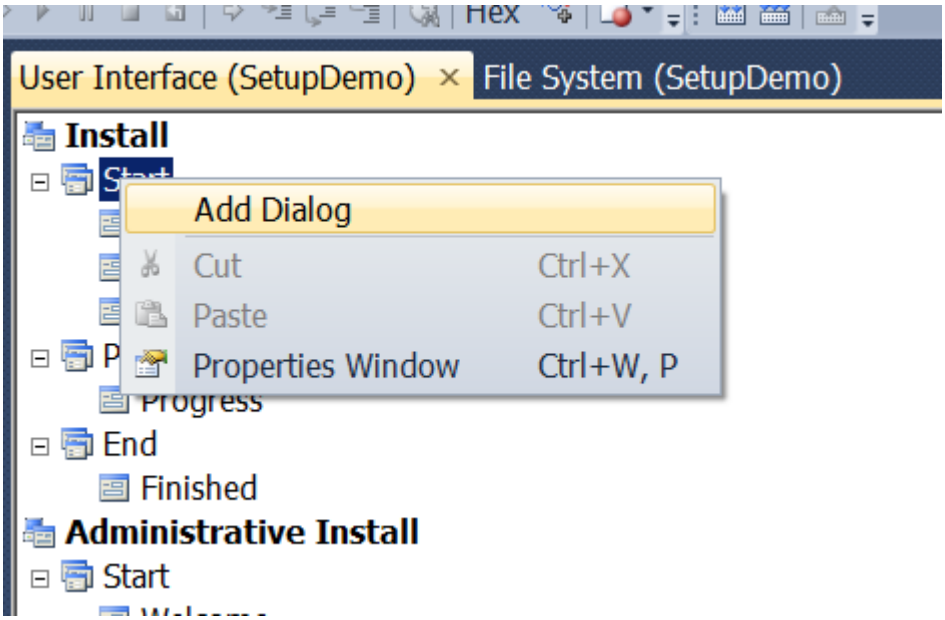


إنشاء مربع حوار معلومات المستخدم Customer Information

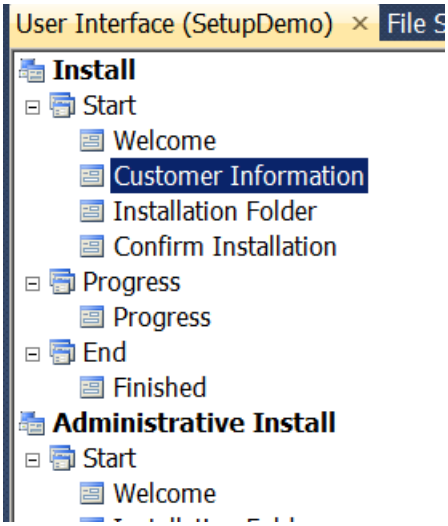
ضمن لوحة Solution Explorer اضغط على زر User Interface Editor



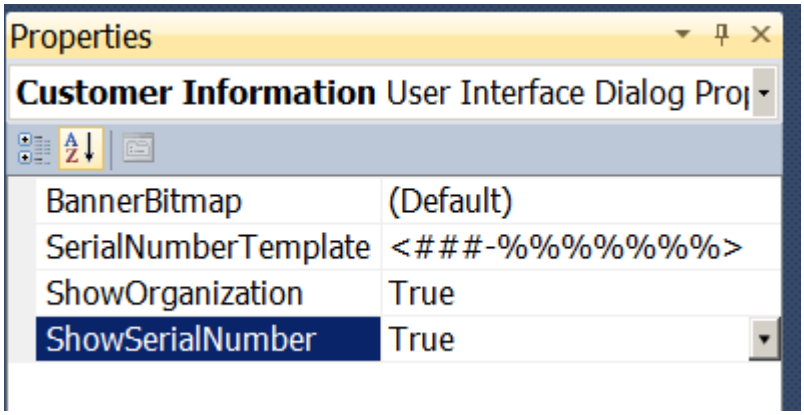
الآن من لسان تبويب User Interface اختر عقدة Install-Start ومن القائمة المنبثقة اختر Add Dialog



اختر Customer Information dialog واضغط موافق ثم قم بتغيير ترتيب صندوق الحوار الجديد ليصبح بعد صندوق حوار Welcome



لإظهار وتفعيل الرقم التسلسلي في صندوق حوار Customer Information ، من لوحة الخصائص لهذا الصندوق غير الخاصية ShowSerialNumber لتصبح True.



خاصية SerialNumberTemplate

افتراضياً تقوم هذه الخاصية بتحديد نموذج للرقم التسلسلي ليتم التحقق منه عند التثبيت، وهي عبر هذا النموذج تقوم أيضاً بتحديد كيفية ظهور مربعات النصوص TextBoxes التي ستشكل مجموعات أوخانات الرقم التسلسلي. على سبيل المثال، القيمة الافتراضية لهذه الخاصية '<###-%%%%%%%%>' ستقوم بإنشاء مربعي نص مفصولين بمحرف '-' (Dash). المجموعة الأولى (###) ستقوم بالتحقق أن المستخدم قد ادخل ثلاثة أرقام. المجموعة الثانية(%%%%%%%%) . ستقوم بالتحقق وفق الخوارزمية التالية: ستقوم بجمع هذه الأرقام المدخلة ضمن هذه المجموعة ومن ثم تقسيم الناتج على 7 فإذا كان باقي القسمة 0 يكون التحقق ناجح (الرقم التسلسلي مقبول) وإلا فالرقم التسلسلي غير مقبول. طبعاً كما تلاحظون هي خوارزمية بسيطة والنموذج السابق يمكن أن يكسر بسهولة (أو بطريق الصدفة) عن طريق ادخال الرقم التسلسلي التالي (000-0000000) وبالتأكيد ليس هذا ما تطمح إليه لحماية برنامجك. أمكانية استخدام نماذج أرقام تسلسلية أكثر تعقيداً متوفرة عن طريق استخدام خليط من المحارف التالية الخاصة ببناء النماذج (القيم) لهذه الخاصية.

(#) مطلوب رقم ولا يدخل في خوارزمية التحقق (الافتراضية).
(%) مطلوب رقم يدخل في خوارزمية التحقق (يدخل في المجموع).
(?) مطلوب رقم أو محرف لا يدخل في خوارزمية التحقق.
(^) مطلوب حرف فقط (كبير\صغير) . وجود رقم بهذه الخانة يؤدي إلى فشل التحقق (سريال غير مقبول).
(<) اي محرف إلى يسار هذه الإشارة لن يكون مرئياً.
(<) أي محرف إلى يمين هذه الاشارة لن يكون مرئياً و تستخدم لإقفال الاشارة الأولى(<).

باستخدام هذه المحارف الخاصة تستطيع بناء نماذج أكثر تعقيداً وبالتالي أكثر متانة وحصانة ضد التخمين أو الكسر من النموذج السابق، لكن ستواجهك مشكلة أخرى هنا وهي: لنفرض أننا نريد بناء نموذج لرقم تسلسلي مشابه لرقم المنتج الخاص ب شركة مكروسوفت (X2RWD - C9W64 - H269K - GGBCX - CB48T)، يمكن أن نعبّر عن هذا النموذج ضمن خاصية SerialNumberTemplate بالقيمة التالية "^^%#^ - ^^^^^ - ^%%%^ - ^%^%% - ^%^^^" المشكلة التي ستظهر هنا هي أن هذا النموذج سيقوم بإنشاء 14 صندوق نص وليس 5 صناديق كما هو مفترض أو متوقع، والسبب يعود إلى أنه هذه الخاصية SerialNumberTemplate ستقوم بإنشاء صندوق نص جديد عند كل تغيير لنوع المحرف (نص أو رقم) -بمعنى أن نموذج مثل #%? سيقوم بإنشاء صندوقي نص- وبالتأكيد أن رؤية 14 صندوق نص لإدخال رقم تسلسلي أو لادخال رقم منتج سيكون مربكاً للمستخدم وهو أمر بعيد تماماً عن الاحترافية، وبالتالي يكون الخيار الأفضل كما ذكرت في مقدمة المقال هو الاعتماد على مكتبة خارجية تضع فيها خوارزميتك الخاصة للتحقق من الرقم التسلسلي. طبعاً الحاجة إلى خاصية SerialNumberTemplate ستبقى موجودة وذلك لإنشاء صناديق النص للرقم

التسلسلي بالشكل والعدد المطلوبين. لكن الآن من أجل المثال الذي سنعمل عليه خلال هذا المقال اترك القيمة الافتراضية لهذه الخاصية كما هي دون تغيير وقم ببناء المشروع Project Build .

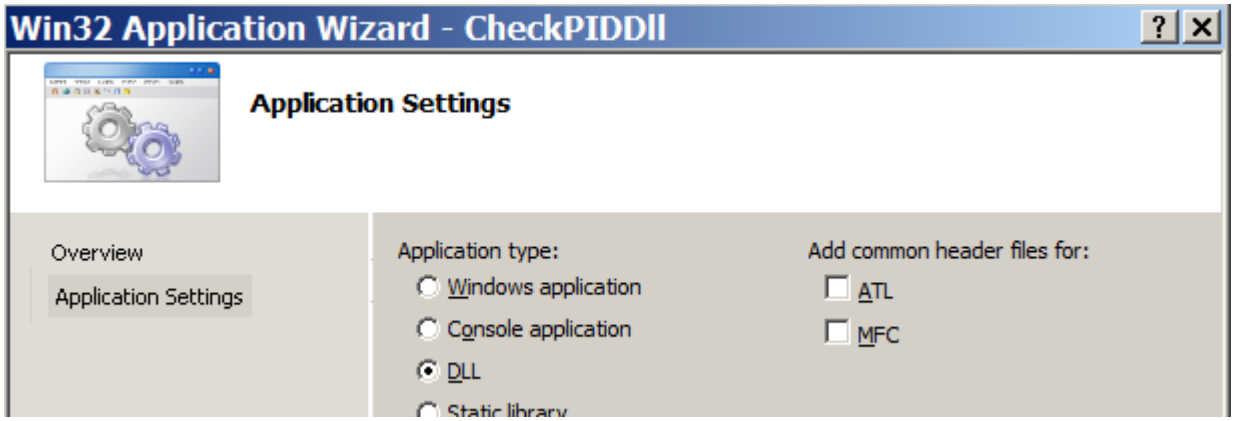
كتابة الكود

الخوارزمية التي سأعتمدها للتحقق ستكون بسيطة جداً لأن غايتي في هذا المقال ليس الخوارزمية نفسها وإنما كيفية استثمار الخوارزمية. سأقوم بجمع (الخانة) الرقم الثالث مع الرقم الأخير من الرقم التسلسلي المدخل وباقي قسمة هذا المجموع على ثلاثة يجب أن يساوي صفر ليكون رقم تسلسلي مقبول، وسأضيف أيضاً شرط آخر وهو عدم تساوي الرقمين لكي لا تمر أرقام تسلسلية من قبيل 000-0000000 أو 333-3333333. أيضاً لتبسيط الكود في هذا المثال سأفترض أن المستخدم سيقوم بإدخال رقم تسلسلي من 11 خانة وليس أقل وبالتالي لن أضيف كود خاص لفحص طول القيمة المدخلة ونوعية كل محرف وبقية (إجراءات السلامة)، لكن طبعاً في المشاريع الحقيقية يجب التأكد من كل هذه الأمور لأن آخر ما تحتاجة هو ظهور اعتراض Exception اثناء تثبيت برنامجك.
عملية التحقق تعتمد بالدرجة الأولى على الحصول على القيمة التي أدخلها المستخدم ضمن مربع الرقم التسلسلي أثناء تثبيت البرنامج وهذا الأمر يمكن تحقيقه من خلال استخدام مكتبة msi.lib. هذه المكتبة تتضمن الكثير من التوابع التي تتيح لك التخاطب مع جلسة التثبيت الحالية Installer Session وذلك عن طريق متحول من النوع MSIHandle الذي يمكن عبره تمرير ممسك handle جلسة (أو تطبيق) التثبيت، وبما أن المكتبة التي سنقوم بكتابتها سنقوم بتضمينها ضمن حزمة التثبيت وبالتالي سيقوم برنامج التثبيت هذا بشحنها load في الذاكرة عند تشغيله هذا يعني أن ممسك handle برنامج التثبيت سيكون مرئيا لمكتبتنا التي ستقوم بدورها بتمريره إلى توابع msi.lib المعدة مسبقاً للتخاطب مع برنامج التثبيت عبر ممسكه handle.
الأمر المهم الآخر في مايخص مثالنا هنا هو أننا سنعتمد على خاصية مساعدة helper property سنسميها 'PIDCHECK' وهي خاصية إضافية ليست موجودة ضمن حزمة التثبيت لكن سنقوم نحن بإضافتها لاحقاً بستخدام أداة التحرير Orca . عندما تجتاز قيمة الرقم التسلسلي لخوارزمية التحقق التي سنكتبها بنجاح سنضع قيمة TRUE ضمن PIDCHECK وعند الفشل سنضع قيمة FALSE.

قيمة الرقم التسلسلي ستكون محفوظة في خاصية متضمنة Built in اسمها PIDKEY لهذا سنستخدم الاجراء MsiGetProperty للحصول على هذه القيمة وبعدها سنقوم بفحص هذه القيمة وفق خوارزمية التحقق التي شرحتها سابقاً وفي النهاية سنستخدم إجراء MsiSetProperty لحفظ النتيجة في الخاصية المساعدة PIDCHECK.

إنشاء مكتبة تحقق من الرقم التسلسلي باستخدام C++

في فجوال ستوديو 2010 قم بانشاء مشروع Win32 dll C++ وسمه CheckPIDDll.



ضمن هذه المكتبة نحتاج فقط إلى إجراء وحيد ليقوم بفحص قيمة الرقم التسلسلي وفق الخوارزمية البسيطة التي شرحتها سابقاً لذلك قم بتعديل نص المكتبة على الشكل التالي:

```
#include    "stdafx.h"
#include    <msi.h>
#include    <msiquery.h>
#include    <tchar.h>

#pragma comment(lib, "msi.lib")

UINT __stdcall VerifyPID(MSIHANDLE hInstall)
{
   TCHAR   szPidKey[MAX_PATH];
   DWORD   dwBuffer;
   dwBuffer = MAX_PATH * sizeof(TCHAR);

   // Get the PIDKEY property
   MsiGetProperty(hInstall, TEXT("PIDKEY"), szPidKey, &dwBuffer);

   //check PIDKEY here
   int n1 = _ttoi(&szPidKey[2]);
   int n2 = _ttoi(&szPidKey[10]);
   int n3 = (n1 + n2) % 3;
   if (n3 == 0 && n1 != n2)
      MsiSetProperty(hInstall, L"PIDCHECK", L"TRUE");//PIDKEY passes check
   else
   {
      //PIDKEY doesn't pass check
      MsiSetProperty(hInstall, L"PIDCHECK", L"FALSE");
      MessageBox(NULL, L"serial number is not valid.", L"Installer", MB_OK |
MB_ICONINFORMATION);
   }

   return 0;
}
```

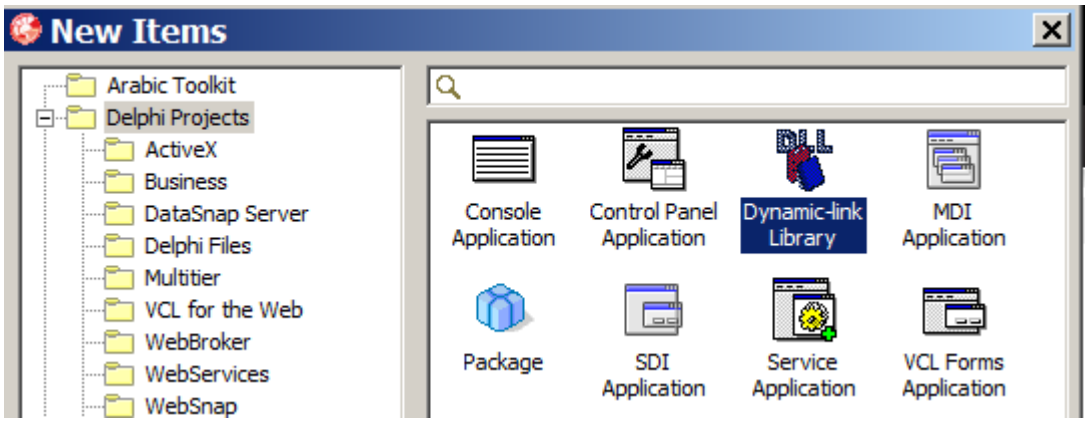
ولجعل هذا التابع مرئياً (للعالم الخارجي) قم بإضافة ملف(Module-Definition File (.def وسمه CheckPIDDll ثم قم بتعديله كما يلي:

```
LIBRARY "CheckPIDDll"
EXPORTS
    VerifyPID
```

قم بترجمة المشروع Build والملف الناتج CheckPIDDll.dll سيكون جاهزاً للاستخدام.

إنشاء مكتبة تحقق من الرقم التسلسلي باستخدام دلفي

في دلفي 2010 (أو في أي نسخة دلفي Unicode) قم بإنشاء مشروع مكتبة dll جديد



هناك تغليف دلفي Delphi Wrapper جيد لمكتبة msi.lib ضمن مشروع JEDI المفتوح المصدر يمكنك تحميلها من هذا الرابط http://sourceforge.net/project/platformdownload.php?group_id=121894 بعدها قم بتعديل كود المكتبة ليصبح كما يلي:

```
library CheckPIDDll;


uses
  SysUtils,
  Classes,
  Windows,
  JwaMsi,
  jwaMSIQuery;

{$R *.res}
function VerifyPID(hInstall: MSIHandle): Integer; stdcall;
var
  sPidKey: PChar;
  dwBuffer: Cardinal;
  sKey: string;
  n1,n2,n3: Integer;
begin
  dwBuffer := MAX_PATH;
  sPidKey := strAlloc(MAX_PATH);
  try
    // Get the PIDKEY property
    MsiGetProperty(hInstall, 'PIDKEY', sPidKey, dwBuffer);
    //check PIDKEY here
    sKey := string(sPidKey);
    n1 := StrToInt(sKey[3]);
    n2 := StrToInt(sKey[11]);
    n3 := (n1 + n2) mod 3;
    if (n3 = 0) and (n1 <> n2) then
     Begin
      MsiSetProperty(hInstall, 'PIDCHECK', 'TRUE');//PIDKEY passes check
     End
    else
     Begin
      MsiSetProperty(hInstall, 'PIDCHECK', 'FALSE');
```

```
      MessageBox(0, PChar('serial number is not valid.'), PChar('Installer'), MB_OK
or MB_ICONINFORMATION);
     End;
  finally
    StrDispose(sPidKey);
  end;
  result := 0;
end;

exports
  VerifyPID;

begin
end.
```

قم ببناء المشروع والمكتبة الناتجة ستكون جاهزة للاستخدام.

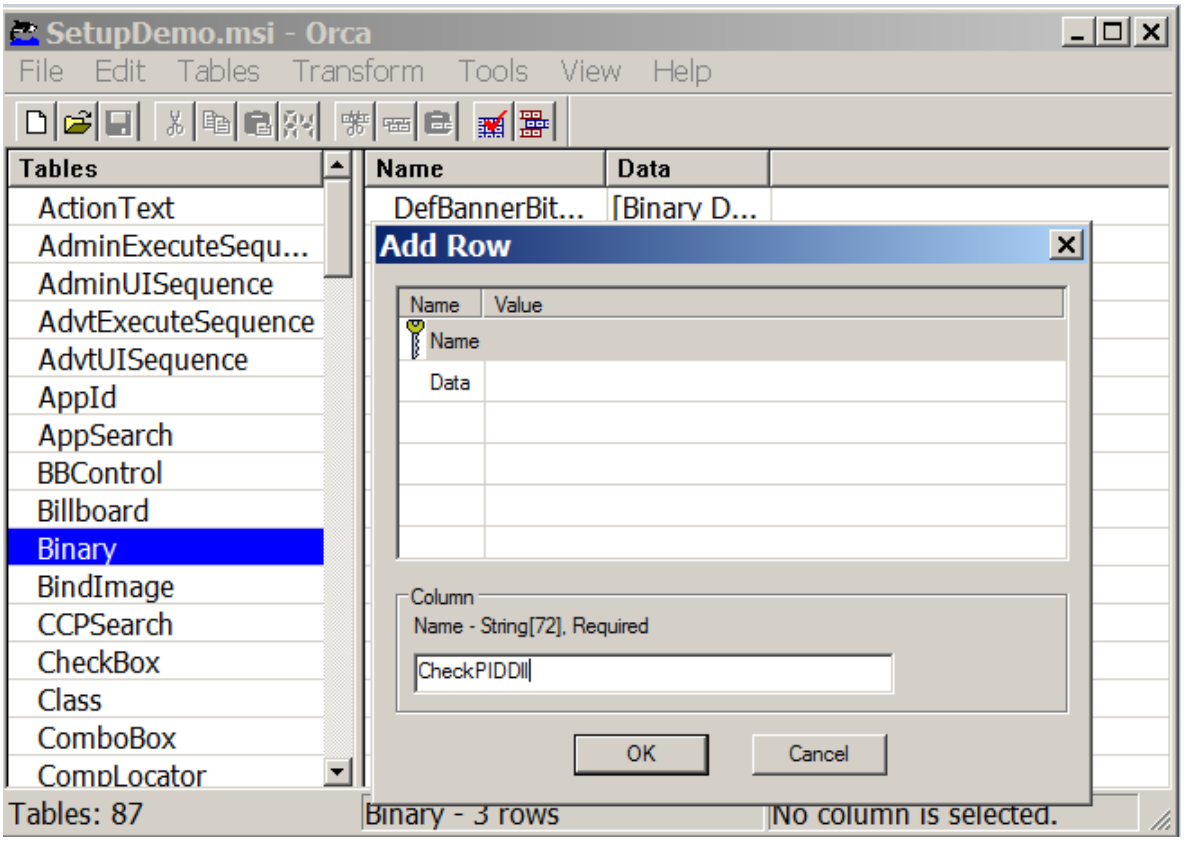
أداة Orca

Orca هي أداة لتحرير حزم التثبيت الخاصة بويندوز (*.msi) وهي أداة مصممة لتؤمن الوصول الكامل إلى كافة الجداول التي تشكل حزمة التثبيت كما تتيح لك الاستفادة من كامل إمكانيات الWindows Installer وهذه الأداة هي جزأ من ال Windows Installer SDK ، لذلك إذا لم تكن قد استخدمت هذه الأداة أو قمت بتثبيتها من قبل، فهي ربما تكون موجودة في جهازك إذا كنت قد ثبت فيجوال ستوديو. فقط ابحث عن Orca.msi في مجلد Program Files وعند العثور على الملف قم بتثبيت الأداة. يمكنك تحميل الأداة أيضاً من خلال هذا الرابط http://www.microsoft.com/en-us/download/details.aspx?id=6510

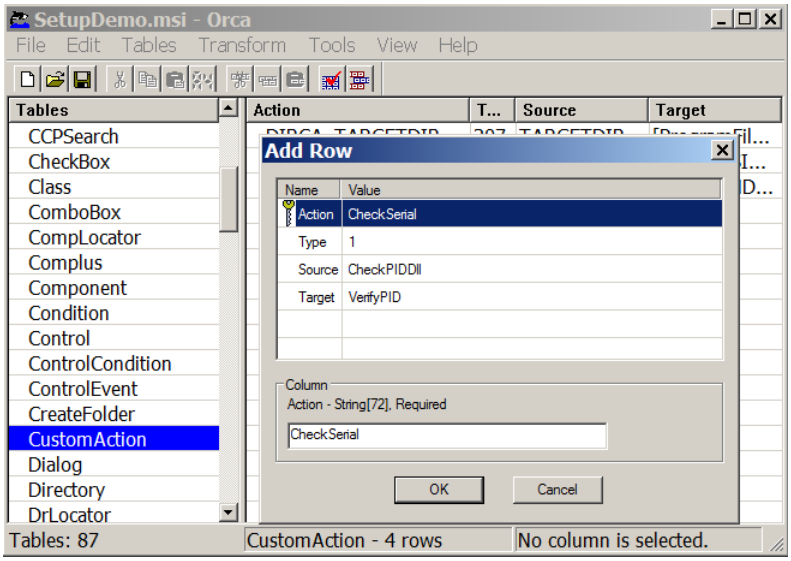
في هذا المقال أنا لن أستفيض بالشرح عن كيفية استخدام أداةOrca وإمكاناتها لأن هذا ليس موضوعنا ولأن هذا الأمر قد يحتاج إلى صفحات ومقالات كثيرة جداً. سأشرح فقط الخطوات المطلوبة لتضمين وربط مكتبة التحقق مع حزمة التثبيت.

تعديل حزمة التثبيت وإضافة مكتبة التحقق من الرقم التسلسلي

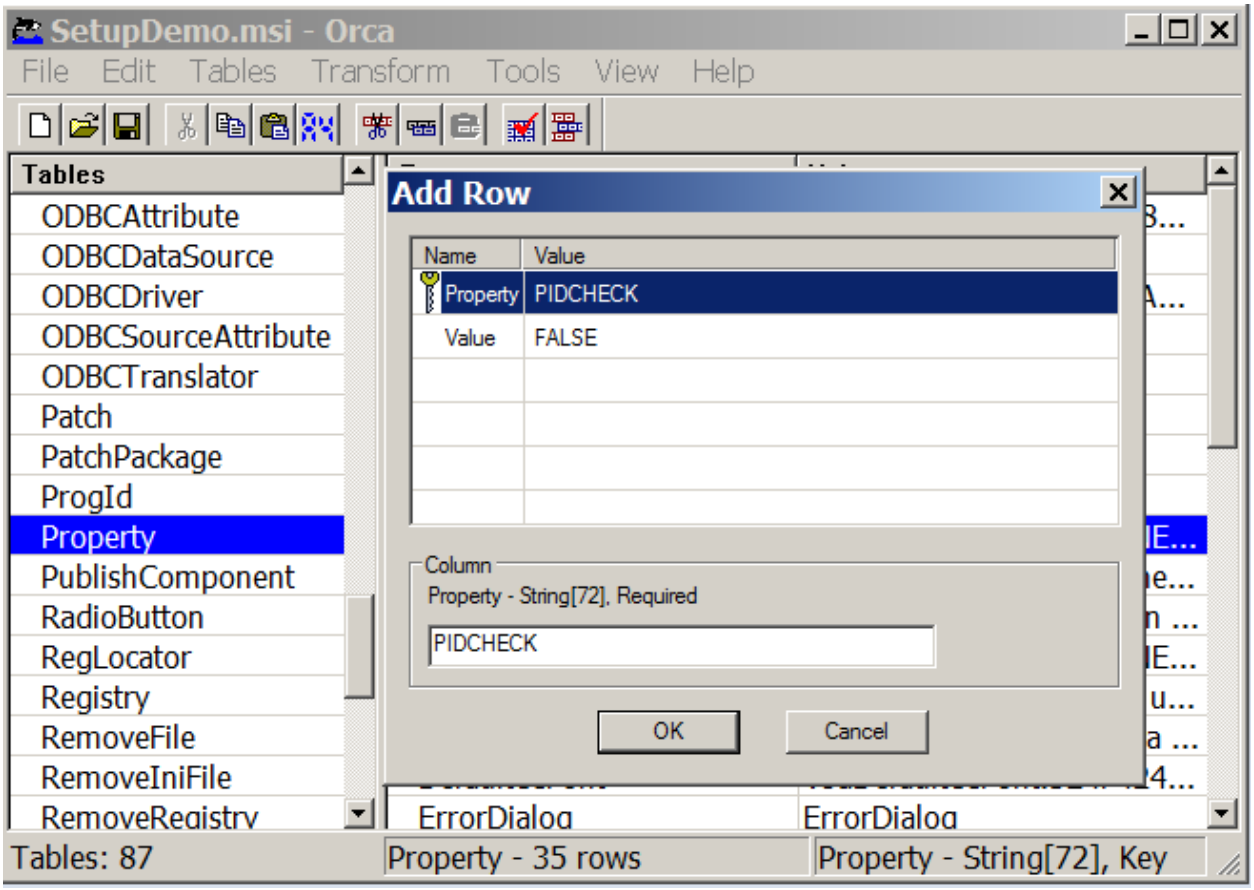
قم بتشغيل Orca وافتح ملف'SetupDemo.msi' الذي أنشئته في بداية المقال، ثم من لوحة الجداول Tables Panel إلى اليسار اختر Binary Table ثم من القائمة المنبثقة اختر Add Row سم هذا الصف الجديد 'CheckPIDDll' وفي حقل Data اضع مسار أحد مكاتب التحقق سواء نسخة دلفي أو نسخة سي++ (يمكنك التغيير لاحقاً).



اختر Custom Action من لوحة الجداول إلى اليسار وأضف Custom Action جديد سمه CheckSerial ضع 1 في حقل Type ، ضع CheckPIDDll في حقل Source وضع VerifyPID في حقل Target. أعتقد أن الأمور واضحة هنا، نحن فقط أشرنا إلى مكتبة التحقق التي إضفناها إلى الحزمة في الخطوة السابقة ومن ثم إلى إجراء التحقق.



الآن سنضيف الخاصية المساعدة PIDCHECK التي تكلمنا عنها سابقاً، اختر Property من لوحة الجداول إلى اليسار ثم قم بإضافة خاصية جديدة سمها PIDCHECK وضع FALSE كقيمة افتراضية.



الآن نحن بحاجة إلى اسناد ال Custom Action الذي أضفناه وسميناه CheckSerial إلى حدث الضغط على زر (التالي) في مربع حوار معلومات المستخدم ليتم فحص الرقم التسلسلي عندها. لتحقيق هذا، اختر ControlEvent من لوحة الجداول إلى اليسار ومن لوحة التفاصيل إلى اليمين انتقل إلى أسفل لتجد الأحداث الخاصة بـCustomerInfoForm يهمنا منها أحداث عنصر NextButton غير قيم الصف الذي يحوي ValidateProductID على الشكل التالي: Event= DoAction, Argument= CheckSerial
غير شرط الحدث NewDialog للعنصر NextButton ليصبح كمايلي: ((PIDCHECK = "TRUE") AND CustomerInfoForm_NextArgs <> "" AND CustomerInfoForm_ShowSerial <> "").

FolderForm	BrowseButton	SpawnDialog	SelectFolderDialog	1	1
FolderForm	BrowseButton	[SelectFolderDial...	TARGETDIR	1	0
CustomerInfoF...	PreviousButton	NewDialog	[CustomerInfoF...	CustomerInfoForm_PrevArgs<>""	0
CustomerInfoF...	CancelButton	SpawnDialog	Cancel		0
CustomerInfoF...	NextButton	EndDialog	Return	CustomerInfoForm_NextArgs="...	1
CustomerInfoF...	NextButton	NewDialog	[CustomerInfoF...	(PIDCHECK = "TRUE") AND Cu...	2
CustomerInfoF...	NextButton	DoAction	CheckSerial	CustomerInfoForm_ShowSerial...	0
WelcomeForm	PreviousButton	NewDialog	[WelcomeForm_...	WelcomeForm_PrevArgs<>""	0
WelcomeForm	CancelButton	SpawnDialog	Cancel		0
WelcomeForm	NextButton	EndDialog	Return	WelcomeForm_NextArgs=""	0
WelcomeForm	NextButton	NewDialog	[WelcomeForm_...	WelcomeForm_NextArgs<>""	1
ConfirmInstall...	PreviousButton	NewDialog	[ConfirmInstallF...	ConfirmInstallForm_PrevArgs<...	0
ConfirmInstall...	CancelButton	SpawnDialog	Cancel		0

هذا كل ما نحتاجه. قم بحفظ التغييرات، أغلق Orca وقم باختبار برنامج التثبيت SetupDemo.msi . يمكنك تغيير واستبدال نسخة مكتبة التحقق من دلفي إلى سي++ وبالعكس فقط عن طريق تغيير صف ال Binary الذي تضيف عبره المكتبة إلى حزمة التثبيت.

خلاصة

كان هذا مجرد مثال بسيط، ولكن بعد الفهم الجيد لهذه الآلية لن يكون صعباً عليك كتابة خوارزميات تسمح بنماذج أرقام تسلسلية بمستوى رقم المنتج الخاص بمكروسوفت.

مراجع

http://msdn.microsoft.com/en-us/library/w3xwh311%28v=vs.80%29.aspx

http://msdn.microsoft.com/en-us/library/windows/desktop/aa370557%28v=vs.85%29.aspx

http://support.microsoft.com/kb/255905/EN-US