كيف تعمل Setup

موضوع **التحزيم** أحد المواضيع المهمة، كما أنه الخطوة الأساسية التي يتم تنفيذها بعد الانتهاء من برمجة التطبيقات أو المكونات تمهيداً لتوزيعها على الجمهور، وبما أنه قد كثرت الأسئلة حول هذا الموضوع فقد حاولت أن أجمع المعلومات الأساسية عن ذلك ..

في البداية ومن خلال الـ Visual Studio .NET سنقوم بإنشاء مشروع جديد، ومن البديهي أن المشروع سيكون من نوع Setup and Deployment Projects والذي يعني أننا سنقوم بإنشاء مشروع خاص بتحزيم تطبيق أو مكون معين، ولو لاحظنا القوالب الموجودة في هذا النوع سنجد أن لدينا أربعة أنواع من برامج التحزيم، وهي كالتالي:

١. Setup Project : يقوم هذا النوع ببناء برامج تحزيم لتطبيقات الويندوز Windows Applications.
٢. Project Web Setup : يقوم هذا النوع ببنا برامج تحزيم لتطبيقات الويب Web Applications.
٣. Merge Module Project : يختص هذا النوع بتحزيم المكونات أو الملفات التي يتم مشاركتها بين أكثر من تطبيق، وينتج عن ذلك ملف من نوع msm يمكن تضمينه في أي نوع من برامج التحزيم الثلاثة الأخرى.
٤. Cab Project : يقوم هذا النوع ببناء ملف مضغوط لتحزيم مكونات الـ ActiveX، والذي يتم وضعه على سيرفر ما Web Server مع إتاحة تحميله من المستعرض الخاص بالمستخدم Web Browser، ويكمن الفرق بين هذا النوع من والنوع الثاني Web Setu Project في المكان الذي يتم تحميل برنامج التحزيم من خلاله.
بالإضافة إلى ذلك سنجد الخيار Setup Wizard والذي يوفر معالج مساعد، يقوم بإعداد واختيار برنامج التحزيم المناسب على ضوء الإجابات التي يتلقاها من المصمم (المبرمج).
لاحظ الشاشة التالية

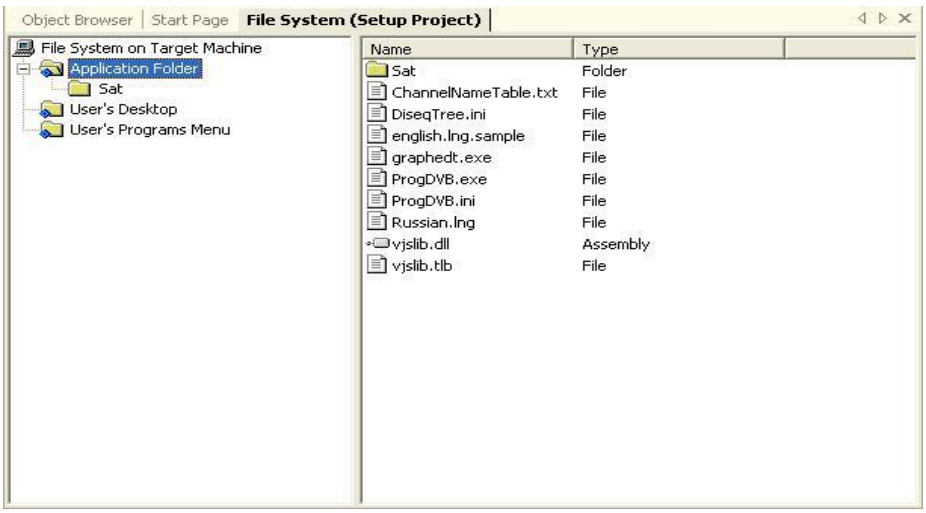
جدير بالذكر - وقبل الدخول في تفاصيل إنشاء هذه الأنواع الأربعة - أنه لا يلزم لتوزيع البرامج أن نقوم بإنشاء برنامج تحزيم من خلال الـ Visual Studio .NET - وإن كانت هذه هي الطريقة الأفضل لعملية التوزيع - حيث أنه يمكننا القيام بعملية التوزيع عن طريق استخدام النسخ اليدوي (copy project أو Xcopy)، أو استخدام برامج تحزيم خاصة بشركات أخرى غير Microsoft.
سنبدأ الآن بالتحدث عن **النوع الأول** Project Setup والخاص ببناء برنامج تحزيم لتطبيق ويندوز Windows Application، والذي يقوم بإنتاج ملف من نوع msi يحتوي على التطبيق بالإضافة إلى كل الملفات المتعلقة به، كما أنه يحتوي على المعلومات الخاصة بعملية التحميل كإدخالات الريجستري التي يتم إضافتها مثلاً، وكل ذلك يتم دمجة في ملف واحد فيما يُسمى بتقنية Microsoft Windows Installer ..
تقوم هذه التقنية (على جهاز العميل) بالاحتفاظ بقاعدة بيانات تحتوي على المعلومات الخاصة بكل التطبيقات التي تم تثبيتها على الجهاز - كل على حدة -، وتحتوي هذه المعلومات على الملفات الخاصة بالتطبيق، والمفاتيح الخاصة به في الريجستري، بالإضافة إلى الأدوات التي يتم تثبيتها معه، وعندما تكون هناك حاجة إلى إزالة برنامج ما Uninstall فإن هذه التقنية تضمن القيام بذلك بشكل آمن بدون التأثير على التطبيقات الأخرى ومكوناتها ..
من نافذة Project New ومن خلال التفرع Setup and Deployment Projects سنقوم باختيار برنامج التحزيم Setup Project، ونقوم باختيار الاسم والمكان المناسب له، ثم OK
ستقوم بيئة الـ Visual Studio .NEt بإنشاء مشروع فارغ مجهز لتحزيم تطبيق موجه إلى نظام الويندوز Windows Application، وسنبدأ بإلقاء نظرة على مكونات هذا المشروع، وتحديداً نافذة الـ Solution Explorer

في أعلى الـ Solution Explorer سنجد سبع أيقونات صغيرة، هذه الأيقونات هي أهم ما لدينا في عملية إنشاء برنامج التحزيم حيث أنها تحتوي على الخطوات الكاملة لإنشاء برنامج تحزيم مثالي، فيما يلي سنسرد هذه الخطوات ثم نناقشها - كل على حدة :-

١. File System Editor
٢. Registry Editor
٣. Types Editor File
٤. User Interface Editor
٥. Costum Actions Editor
٦. Launch Conditions Editor
٧. Properties

الخطوة الأولى: إعداد الملفات الخاصة بالتطبيق

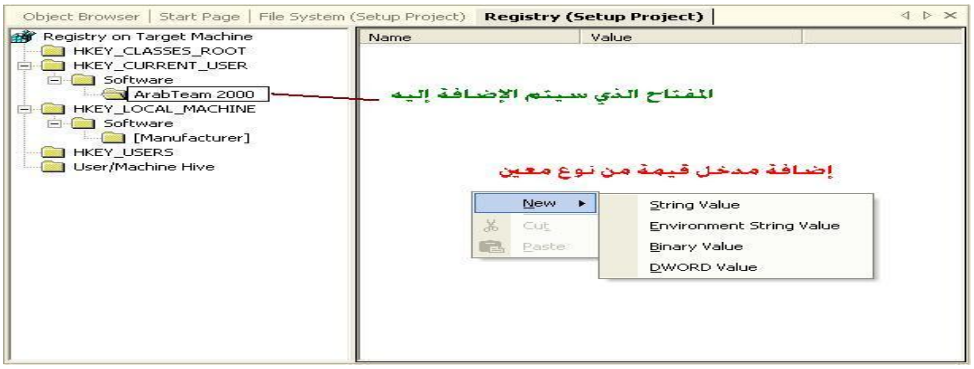
قبل كل شيء سنحدد لبرنامج التحزيم الملفات التي سيتم تحميلها عند المستخدم، ولذلك سنذهب إلى الـ File System Editor ويمكن الوصول إليه من خلال الأيقونة الأولى في نافذة الـ Solution Explorer كما ذكرت سابقاً.
سنجد أن الـ File System Editor مصمم على شكل مستعرض أو مستكشف الويندوز Explorer Windows، حيث نجد في الجانب الأيسر المجلدات الرئيسية التي سيتم التحميل إليها، وفي الجانب الأيمن الملفات التي سيتم إضافتها بداخل هذه المجلدات.
لاحظ الصورة التالية:



تقوم بيئة الـ Visual Studio .NET افتراضياً بإضافة ثلاث مجلدات رئيسية في الجانب الأيسر، وهي:
Application Folder ويحتوي على الملفات التي سيتم إضافتها إلى المجلد الخاص بالتطبيق في جهاز العميل
User's Desktop ويحتوي على الملفات التي سيتم إضافتها إلى الـ Desktop (سطح المكتب) الخاص بالمستخدم
User's Programs Menu ويحتوي على الملفات التي سيتم إضافتها إلى قائمة Start (ابدأ) في جهاز المستخدم
وغالباً ما تقتصر الإضافة في المجلدين الأخيرين على الاختصارات Shortcuts فقط.
من الممكن إضافة مجلدات أخرى إلى هذه المجلدات، على سبيل المثال يمكن إضافة مجلد لتحميل الملفات إلى مجلدات الـ System أو الـ Fonts أو الـ Favorites الخاصة بالنظام، وذلك عن طريق النقر بزر الماوس الأيمن في الجانب الأيسر من نافذة الـ File System Editor واختيار Folder Add Special ..
أما في الجانب الأيمن من نافذة الـ File System Editor فإنه يتم إضافة المحتوى الخاص بهذه المجلدات سواء كانت هذه المحتويات عبارة عن مجلدات فرعية، أو ملفات، أو اختصارات، او مكتبات تجميع Assemblies، وذلك عن طريق النقر بزر الماوس الأيمن ثم الاختيار المناسب ..

الخطوة الثانية: إعداد الإدخالات الخاصة بالتطبيق في الـ Registry (مسجل النظام):

يعتبر الـ Registry أحد الأدوات الهامة للمبرمج، والذي يمكن أن يمثل أداة لتخزين قيم أو إعدادات معينة حسب الحاجة إليها .. تتيح بيئة الـ Visual Studio .NET ومن خلال برنامج التحزيم القيام بإضافة القيم التي تريدها إلى الريجستري وذلك في المكان المخصص للتطبيق، حيث يتم إتاحة مفتاحين فرعيين بداخل المسارين التاليين: HKCU\Software و HKLM\Software، ويمكنك اختيار الأسماء المناسبة لهما، ثم بعد ذلك يمكنك إضافة القيم إلى هذه المفاتيح من خلال الجانب الأيمن وزر الماوس الأيمن
لاحظ الصورة التالية



جدير بالذكر أنه لا يمكنك التعديل على قيم الريجستري الموجودة مسبقاً والخاصة بالنظام والتطبيقات الأخرى، حيث لا يمكن الوصول إليها من هنا، وإنما يلزمك كتابة الكود الخاص بالتعامل مع الريجستري، والوصول إلى هذه القيم من خلال التطبيق الخاص بك ..

الخطوة الثالثة: تعريف أنواع الملفات الخاصة بالتطبيق

المقصود بأنواع المفات هنا الـ Extension (الامتداد) الخاص بها، كما هو الحال بالنسبة لملفات الـ Microsoft WinWord مثلاً فهي تأخذ الامتداد DOC، وهي مرتبطة ببرنامج الـ Word، فعندما يتم النقر على أحد هذه الملفات مرتين فإنه يتم تشغيل برنامج الـ Word وفتح الملف، وكذلك هو الحال بالنسبة لملفات الـ TXT وبرنامج الـ Notepad، وملفات الـ RTF وبرنامج الـ Wordpad ... الخ هل تريد أنت أيضاً أن يكون لتطبيقك ملفات من نوع معين مرتبطة به بهذا الشكل، الأمر أبسط مما تتصور، فبعد أن كان ذلك يتطلب كوداً معقداً في الإصدارات الأقدم فإن الأمر لا يتعدى نقرة بالماوس في بيئة الـ Visual Studio .NET ..

من نافذة الـ File Types Editor سنقوم بالنقر بزر الماوس الأيمن واختيار Add File Type، ثم نقوم باختيار اسم تعريفي لملفات النوع الجديد الذي سنضيفه، وبعد ذلك من نافذة الخصائص سيكون علينا تعديل الخاصية Extension إلى نوع الملف الذي نريد تخصيصه للتطبيق الخاص بنا، ويجب أيضاً تعديل الخاصية Command لتحديد مسار التطبيق المرتبط، والذي سيتم تشغيله عند محاولة تشغيل الملف ..

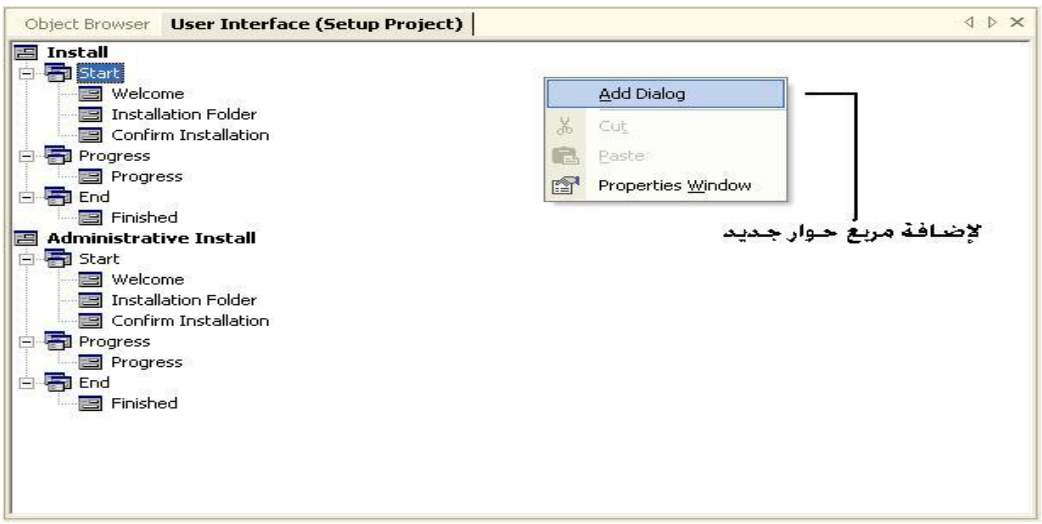
لاحظ أن هناك بند فرعي بالاسم Open وهو يمثل الإجراء الذي سيتم اتخاذه بعد فتح البرنامج المرتبط، ومن الطبيعي أن الإجراء الافتراضي سيكون هو فتح الملف الذي من خلاله تم تشغيل البرنامج، ويمكن تغيير هذا الإجراء أو إضافة أي إجراءات أخرى حسب الرغبة ..

الخطوة الرابعة: تصميم واجهات المستخدم

والمقصود بواجهات المستخدم هنا هي مربعات الحوار التي تظهر للمستخدم أثناء عملية تحميل التطبيق، ونستطيع الوصول إلى مربعات الحوار هذه والتعديل عليها عن طريق نافذة الـ User Interfaces Editor، وكما نرى فهي تنقسم إلى قسمين:

القسم الأول Install : وهو خاص بمربعات الحوار التي ستظهر للمستخدم أثناء عملية تحميل التطبيق.

القسم الثاني Install Administrative : وهو خاص بمربعات الحوار التي ستظهر عندما يقوم مدير النظام برفع برنامج التحزيم إلى مكان ما في شبكة Network، والذي يتم الوصول إليه عن طريق تمرير a/ إلى ملف التثبيت أثناء تشغيله ..



تقوم بيئة الـ Visual Studio .NET تلقائياً بإضافة خمس مربعات حوار افتراضية هي:

مربع الحوار Welcome للترحيب بالمستخدم وتقديم نبذة عن التطبيق

مربع الحوار Installation Folder لتمكين المستخدم من اختيار المجلد الذي سيتم نسخ التطبيق إليه

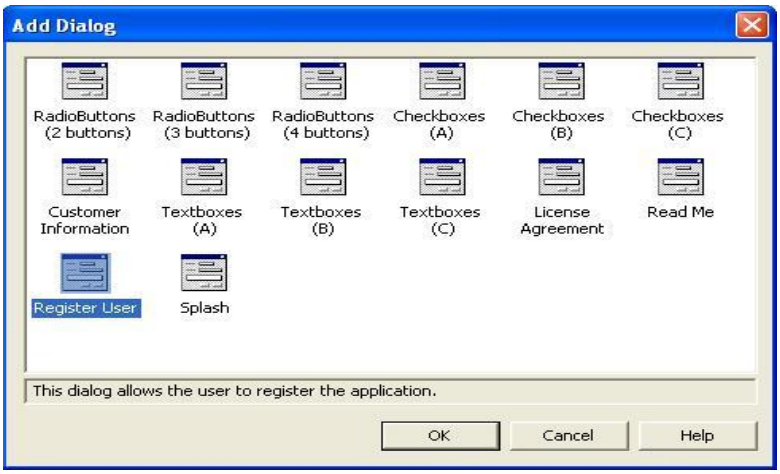
مربع الحوار Confirm Installation للتأكيد على استمرار عملية التثبيت

مربع الحوار Progress لإظهار حالة تحميل الملفات إلى جهاز المستخدم

مربع الحوار Finished لإنهاء برنامج التحزيم

لاحظ أنه يمكنك التحكم في الصور والكلمات المعروضة في هذه المربعات عن طريق نافذة الـ Properties (الخصائص)

لاحظ أيضاً أن هذه المربعات الحوارية ليست إلزامية، أي أنه يمكن الاستغناء عن بعضها والاستعانة بغيرها، فعن طريق الزر Delete يمكن حذف أي منها، كما أنه يمكن إضافة مربع حواري جديد عن طريق النقر بزر الماوس الأيمن كما يظهر في الصورة السابقة، ثم اختيار Add Dialog ليظهر لنا النموذج التالي:

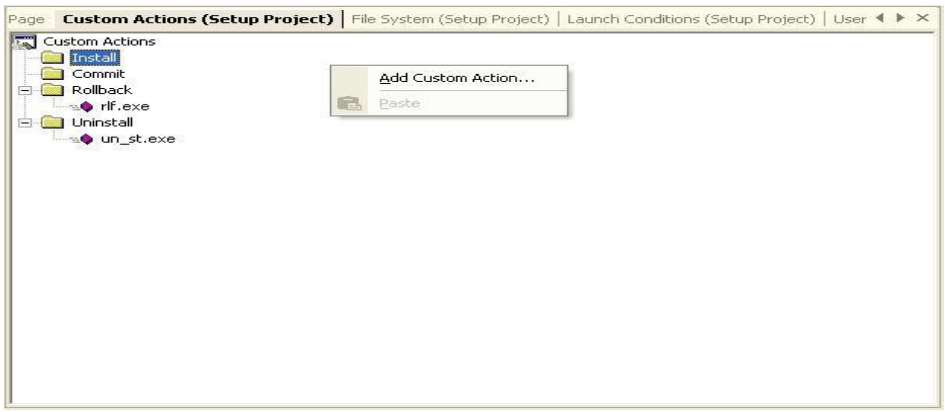


هذا النموذج يحتوي على العديد من مربعات الحوار التي يمكن الاستعانة بأي منها، وعلى حسب مربع الحوار الذي يتم إضافته فإن مربع الخصائص Properties يختلف ليضيف خاصية جديدة، أو يلغي أخرى، على سبيل المثال عند إضافة مربع الحوار License Agreement فإنه يلزم إعداد الخاصية LicenseFile وهي تستقبل ملف نصي من نوع RTF يحتوي على نص اتفاقية الترخيص، وعند إضافة مربع الحوار Register User فإنه يلزم إعداد الخاصية Executable وهي تستقبل ملف تنفيذي من نوع EXE يحتوي على البرنامج الخاص بتسجيل النسخة من التطبيق ، وهكذا الحال بالنسبة لمربعات الحوار الأخرى ...

<u>الخطوة الخامسة</u>: <u>تنفيذ إجراءات معينة بعد إتمام عملية التحميل</u>

إحدى المميزات الجميلة الموجودة في برامج التحزيم المنشأة في بيئة الـ Visual Studio .NET أنها تسمح لك بتشغيل كود معين بعد إتمام عملية التثبيت، وهو ما يُسمى بالـ Custom Action، وتظهر فائدة هذه الميزة في القيام بالعمليات التي لا يمكن لبرنامج التحزيم أن يقوم بها بنفسه، فربما تكون بحاجة إلى إنشاء قاعدة بيانات أثناء عملية تثبيت التطبيق مثلاً، وهو ما لا يمكنك عمله بواسطة برنامج التحزيم، ولكن من الممكن إضافة Custom Action معين يقوم بهذه العملية، ويتم اعتباره كجزء لا يتجزأ من عملية التثبيت بحيث إذا فشل تنفيذه يتم إزالة التطبيق تلقائياً ..

يتم إضافة الـ Custom Action من خلال نافذة الـ Custom Actions Editor، وتحتوي هذه النافذة على أربع مجلدات يمكن الإضافة إليها، وهي على التوالي Install ثم Commit ثم Rollback ثم Uninstall ..



ويمثل كل واحد من هذه الأربعة التوقيت الذي سيتم فيه تنفيذ الإجراء، وما إذا كان سيتم تنفيذه أثناء التثبيت أو بعده أو في حال فشل التثبيت أو في حال إلغاء التثبيت .. وهكذا

أبسط مثال لذلك أن نقوم بكتابة كود يقوم بإنشاء قاعدة بيانات، ونضعه في ملف تنفيذي، ثم نقوم بإضافته كـ Custom Action يتم تنفيذه بعد تثبيت البرنامج ..

كما نعلم فإنه يتم تنفيذ الإجراء بعد الانتهاء من عملية تثبيت البرنامج، وبالتالي فإنه لا يتمكن من الوصول إلى العمليات التي قام بها المستخدم أثناء عملية التثبيت، وكحل لذلك يمكن تمرير البيانات من برنامج التحزيم إلى الإجراء عن طريق الخاصية CustomActionData ..

من الممكن أيضاً إضافة شرط معين يجب تحققه لتنفيذ الإجراء عن طريق الخاصية Condition، وذلك للإجراءات التي يتم تنفيذها في حالات معينة، وأبسط مثال لذلك إجراء يتم تنفيذه في حال كون إصدارة النظام Windows9x، ولكن لا يتم تنفيذه مع الإصدارات الأحدث ... وهكذا

<u>الخطوة السادسة</u>: <u>التحقق من شرط معين</u>

أحد المزايا المهمة الموجودة في برامج التحزيم التي نقوم بإنشائها في بيئة الـ Visual Studio .NET، وهي إمكانية التحقق من شرط معين قبل التثبيت، وعلى أساس هذا الشرط يتم تحديد الموقف من إتمام التثبيت أو إلغائه ..

سنقوم بالتوجه إلى نافذة الـ Launch Conditions Editor.. نلاحظ في الجزء الأيسر من الصورة أنه يوجد قسمين:
- القسم الأول هو Search Target Machine
- القسم الثاني هو Launch Conditions

يوجد خمسة أنواع من الشروط التي يمكن التحقق منها وهي كالتالي:

١. التحقق من وجود ملف معين:
ومن خلاله نقوم بالتحقق من وجود ملف معين في جهاز المستخدم، ولذلك نقوم بإضافة باحث يقوم بالبحث عن الملف وذلك عن طريق النقر بزر الماوس الأيمن على القسم الأول المذكور سابقاً ثم اختيار Add File Search، نقوم بعدها باختيار الاسم المناسب لهذا الباحث، ثم نقوم بإعداد الخصائص اللازمة لهذا الباحث من نافذة الخصائص، وأهمها خاصية FileName والتي تحتوي على اسم الملف المراد البحث عنه، وخاصية Folder وهي عبارة عن المجلد الذي سيتم البحث بداخله، وخاصية Depth والتي تحدد مستوى البحث بالمجلدات الفرعية بداخل المجلد الذي سيتم البحث بداخله، والخاصية Property والتي تحوي اسماً تعريفياً يتم من خلاله الربط بالـ Launch Condition الذي سيتم إضافته في القسم الثاني، بالإضافة إلى بعض الخصائص الأخرى التي تساعد في تسهيل مهمة البحث عن الملف ..
بعد الانتهاء من إعداد الخصائص الخاصة بالباحث نقوم بإضافة Launch Condition في القسم الثاني، ونقوم بربطه بالباحث عن طريق الخاصية Condition والتي تحتوي على قائمة منسدلة بجميع البواحث الموجودة في برنامج التحزيم، ثم نقوم بإعداد الخاصية Message والتي تظهر للمستخدم في حال عدم تحقق الشرط، والخاصية InstallUrl والتي تقوم بتوجيهك إلى صفحة ويب خاصة بتحميل الملف الذي لم يتم إيجاده، ويمكنك تحديد هذه الصفحة أو تركها فارغة.

٢. التحقق من قيمة معينة في الريجستري:
لا يختلف الأمر كثيراً عن سابقه، فسنقوم بإضافة باحث للتحقق من قيمة معينة في الريجستري، ويمكن الاختلاف في الخصائص حيث أن الخاصية Root تحتوي على الجذر الخاص بمدخل القيمة، ثم الخاصية RegKey والذي تحتوي على مسار المفتاح الذي يحوي مدخل القيمة المراد التحقق من قيمته، وأخيراً الخاصية Value وتحتوي على اسم مدخل القيمة المراد التحقق من قيمته .. بنفس الطريقة السابقة يتم إضافة Launch Condition في القسم الثاني، وإعداد الخاصية Condition للتحقق من وجود قيمة معينة في المدخل الذي تمت الإشارة إليه..

٣. التحقق من وجود برنامج الـ Windows Installer:
وكما هو الحال بالنسبة لما سبق يتم إضافة باحث للتحقق من وجود برنامج الـ Windows Installer والذي يتم الاعتماد عليه أثناء عملية التثبيت، ومن حسن الحظ أنه يمكنك من خلال إضافة Launch Condition في القسم الثاني، ثم ربطه بالباحث، وإعداد الخاصية InstallUrl، أن تقوم بتحديد مسار معين لتحميل هذا البرنامج في حال عدم تواجده ..

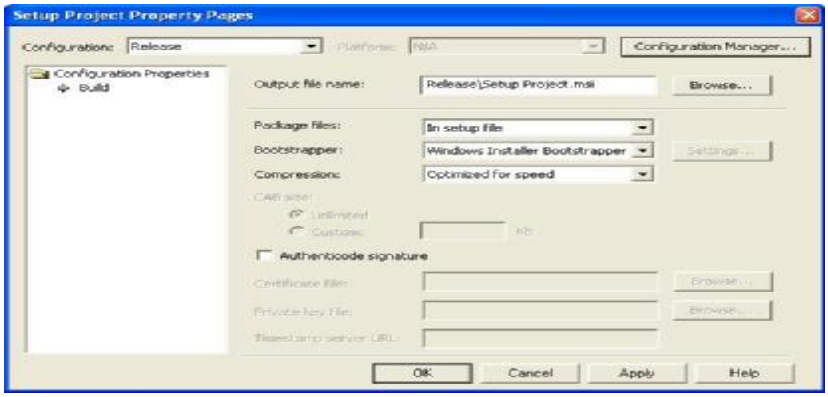
٤. التحقق من وجود مكتبة الـ Framwork NET.:
ويبدو هذا الشرط غاية في الأهمية حيث أنه لابد من وجود مكتبة الـ NET Frmaework. لتشغيل أي تطبيق يتم تصميمه بواسطة الـ Visual Studio .NET. ومن هنا تكمن أهمية وجود هذه المكتبة لأنه من دونها لن تكون هناك فائدة لتثبيت البرنامج لأنه لن يعمل ..
يتم إنشاء هذا الشرط من خلال القسم الثاني Launch Conditions، ومن خلال الخاصية InstallUrl يتم تحديد موقع الملف dotnetfx.exe الخاص بتثبيت مكتبة الـ NET Frmaework. لكي يتم تشغيله في حال عدم وجود هذه المكتبة في جهاز المستخدم

٥. التحقق من تثبت الـ Internet Information Services IIS -:
بنفس الطريقة السابقة يمكننا التحقق من وجود الـ IIS مثبتاً في جهاز المستخدم، وفي حال عدم تواجده يتم إظهار رسالة معينة واتخاذ الإجراء المناسب.

أضف إلى هذه الشروط السابقة ما قد يقوم المستخدم بإضافته من الشروط، على سبيل المثال من الممكن **التحقق من وجود الإصدارة الحديدة من الـ Microsoft Data Access Components أو MDAC** عن طريق إضافة باحث للتحقق من قيمة معينة في الريجستري، وإعداد الخصائص التالية:
Root = vsdrrHKLM
Software\Microsoft\DataAccess = RegKey
Value = FullInstallVer
MDACSEARCH = Property
ثم نقوم بإضافة Launch Condition وربطه بالباحث السابق عن طريق إعداد الخاصية Condition لكي تصبح "MDACSEARCH>=٢,٧"، وكما هو واضح فهي تتحقق من أن إصدارة الـ MDAC هي الإصدارة ٢,٧ أو الأحدث ..
في خاصية الـ Message يمكنك إضافة الرسالة التالية:

installing this application. You can install MDAC from the MDAC version 2.7 or higher must be installed prior to
/http://www.microsoft.com Microsoft Web site
أو أي رسالة أخرى تراها مناسبة ..

الخطوة السابعة والأخيرة: **الخصائص العامة**
ويمكن الوصول إليها من قائمة Project ثم Properties لتظهر لنا نافذة الخصائص والتي لا تحتوي على الكثير من التعقيد، ويمكن فهمها بكل بساطة ..



في الـ Output file name يتم تحديد المجلد الذي سيتم إنشاء برنامج التحزيم بداخله ..
في الـ Package files يتم تحديد نوعية ملفات برنامج التحزيم وما إذا كانت ملفاً واحداً بالامتداد msi، مع إمكانية جعله من نوع CAB وتقسيمه بحجم معين ..
في الـ Compression يتم تحديد تقنية الضغط، وتحديد الأولوية بين الحجم وسرعة التثبيت بالنسبة لبرنامج التحزيم ..
بذلك نكون قد انتهينا من الحديث عن النوع الأول وهو Setup Project، ولا يختلف الأمر كثيراً بالنسبة للأنواع الأخرى حيث يمكنكم اكتشاف الفروق ببساطة ..
عودة إلى السؤال الرئيسي وهو كيف يمكن تحزيم مكتبة الـ NET FrameWork. بجوار التطبيق بحيث يتم تحميلها في حال عدم تواجدها في جهاز المستخدم، وذلك بشكل تلقائي ودون تدخل المستخدم ..
لذلك سنقوم باستخدام ما يسمى بالـ BootStrapper Sample Setup.exe، وهو عبارة عن ملف تنفيذي يقوم بعملية البحث التلقائي عن مكتبة الـ NET FrameWork. وفي حال عدم تواجدها فإنه يقوم بتحميلها إلى جهاز المستخدم بشكل صامت وبدون إزعاج، بعدها يتم تحميل التطبيق نفسه، والذي قمنا بتحزيمه سابقاً من خلال الـ Microsoft Windows Installer في ملف واحد يحمل الامتداد msi ..
وفيما يلي سنشرح العمليات التي يقوم بها هذا الـ Sample Setup.exe BootStrapper بشيء من التفصيل، رغم أن معرفتها ليست بالأهمية الكبيرة:

العملية الأولى: **التحقق من تواجد مكتبة الـ** NET FrameWork.
وذلك من خلال مفتاح الريجستري التالي
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\policy\v1.0
حيث يتم التحقق من رقم الإصدارة وما إذا كانت مماثلة لتلك التي يحتاجها التطبيق أو لا، وفي حال عدم تواجد المكتبة أو عدم توافق الإصدارة فإنه يتم الانتقال إلى العملية الثانية والبدء بتحميل مكتبة الـ NET FrameWork.

لاحظ أنه من الممكن التحقق من إصدارة اللغة من خلال تحديد رمز اللغة التي نريد التحقق منها في ملف الإعدادات Setting.ini، وفي حال لم يتم تحديد أي لغة فإنه سيتم اعتبار اللغة الإنجليزية تلقائياً ..

العملية الثانية: **التحميل الهادئ لمكتبة الـ** NET FrameWork.:
وذلك من خلال استخدام الأمر التالي لبدء التحميل
CODE
dotnetfx.exe /q:a /c:"install /l /q"

ويقوم هذا الأمر بإخفاء كافة الواجهات ورسائل الخطأ التي تظهر في حال التحميل العادي لمكتبة الـ NET FrameWork. فيما يسمى بعملية التحميل الصامت أو الهادئ، ويتم تسجيل المعلومات الخاصة بالتحميل ورسائل الخطأ في الملف Netfx.log وتخزينه في مجلد الـ Temp ..

العملية الثالثة: **معالجة الأخطاء العامة أثناء عملية التحميل**

وفيما يلي توضيح لرسائل الخطأ التي يتم معالجتها

الخطأ ٣٠١٠ والخطأ ٨١٩٢ ويقعا عند الحاجة إلى إعادة تشغيل الجهاز ..

ويتم معالجته بالرسالة التالية reboot now Setup requires a reboot. Would you like to؟

الخطأ ٤١٠١ ويقع في حال تشغيل نسخة أخرى من برنامج التثبيت ..

ويتم معالجته بالرسالة التالية running Another instance of setup is already

الخطأ ٤٠٩٧ ويقع في نظام الـ Windows NT، وفي حال عدم الامتلاك لصلاحيات التحميل

ويتم معالجته بالرسالة التالية You do not have the permissions necessary to install this application. Please contact your administrator

الخطأ ١٦٣٣ ويقع في حال وجود مشاكل في منصة تشغيل التطبيق

ويتم معالجته بالرسالة التالية Your computer is not configured properly to run this application. Please contact support

بقية الأخطاء التي من الممكن أن تحدث أثناء عملية التحميل

ويتم معالجتها بالرسالة التالية Setup has encountered errors. Installation cannot proceed

العملية الرابعة: **استدعاء ملف التحميل الخاص بالتطبيق** (**ذو الامتداد** msi) **وإرجاء إعادة تشغيل الجهاز إلى ما بعد تحميل التطبيق**

وذلك من خلال استخدام الأمر التالي

msiexec /i myapp.msi REBOOT=ReallySuppress

والذي يقوم بتشغيل الملف الخاص بتثبيت التطبيق (الذي تم تحزيمه) على جهاز المستخدم ..

وهذا هو مجمل العمليات التي يقوم بها الـ Setup.exe BootStrapper Sample، وكما ذكرنا فليس من الضروري معرفتها لأنها تتم بشكل تلقائي في الخلفية ودون تدخل من المطور أو المستخدم ..

ولكن لكي تتم العملية بشكل سليم يجب أن نقوم بإنشاء ملف خاص بالإعدادات يحمل الاسم Settings.ini، ويحتوي على المعلومات المطلوبة أثناء التثبيت، كما أنه يستحسن إنشاء ملف نصي ينبه المستخدمين إلى الإعدادات المطلوبة لتثبيت التطبيق، وبطبيعة الحال لابد أن نضيف ملف الـ Dotnetfx.exe والخاص بتحميل مكتبة الـ FrameWork NET.، وكل هذه الملفات يتم تجميعها إلى جوار ملف الـ Sample Setup.exe BootStrapper، وملف تثبيت التطبيق ذو الامتداد msi. في مجلد واحد ..

كيفية إنشاء ملف الإعدادات Settings.ini:

بداية يجب أن نعرف أن ملف الـ Setup.exe BootStrapper Sample يحتاج إلى ملف إعدادات خارجي يحتوي على:

- موقع ملف الـ Dotnetfx، وموقع ملف التطبيق msi file
- إصدارة لغة مكتبة الـ NET FrameWork. المراد التحقق من وجودها
- النص المخصص الذي سيظهر في مربعات الحوار الخاصة بالملف Setup.exe

وفيما يلي سنضع الشكل التقليدي البسيط لملف الإعدادات Settings.ini والعناصر الموجودة بداخله، مع إمكانية إسناد القيم المناسبة لكل عنصر

ODE

[Bootstrap]

Msi=

LanguageDirectory=

ProductName=

DialogText=

CaptionText=

ErrorCaptionText=

FxInstallerPath=

العنصر MSI يحتوي على موقع الملف الخاص بتثبيت التطبيق MSI File، ومن الممكن كتابة الاسم مباشرة ليدل على أنه موجود في نفس مجلد الـ Setup.exe كالتالي:

E**Msi=ArabTeam.msi**

ليدل على أن الملف ArabTeam.exe هو الملف الخاص بتثبيت التطبيق، وأنه موجود في نفس مجلد الـ Setup.exe

كما أنه من الممكن أن يكون موقع ملف التثبيت في مجلد فرعي كالتالي:

Msi=Setup/ArabTeam.msi

على اعتبار أن ملف الـ ArabTeam.msi موجود في مجلد فرعي بالاسم Setup

العنصر LanguageDirectory يحتوي على إصدارة لغة مكتبة الـ NET FrameWork. المراد التحقق من وجودها، وفي حال الرغبة في تجاهل هذا العنصر فإنه يتم إضافة فاصلة (') قبل العنصر وسيتم اختيار اللغة الإنجليزية بشكل افتراضي، أما في حال اختيار لغة ما فإنه يتم إضافة رمز اللغة، مثلاً اللغة الفرنسية fr، واللغة الألمانية de وهكذا ..

العنصر ProductName ويحتوي على اسم التطبيق الذي سوف يظهر للمستخدم أثناء عملية التثبيت، وفي حال الرغبة في تجاهل هذا العنصر فإنه يتم إضافة فاصلة (') قبل العنصر وسيتم الاستعاضة عنه بـ Application ..
العنصر DialogText ويحتوي على النص الذي سيظهر مربع الحوار الذي سيظهر بعد تشغيل ملف الـ Setup.exe، وفي حال الرغبة في تجاهل هذا العنصر فإنه يتم إضافة فاصلة (') قبل العنصر وسيتم الاستعاضة عنه بـ .To start Application Setup, click OK Cancel To quit without installing, click
العنصر CaptionText ويحتوي على العنوان الذي سيظهر في نافذة التثبيت، وفي حال الرغبة في تجاهل هذا العنصر فإنه يتم إضافة فاصلة (') قبل العنصر وسيتم الاستعاضة عنه بالعنصر ProductName ..
العنصر ErrorCaptionText ويحتوي على العنوان الذي سيظهر في نوافذ الخطأ، وفي حال الرغبة في تجاهل هذا العنصر فإنه يتم إضافة فاصلة (') قبل العنصر وسيتم الاستعاضة عنه بـ Application Setup Error ..
العنصر FxInstallerPath ويحتوي على موقع الملف الخاص بتثبيت مكتبة الـ NET FrameWork. كما هو الحال بالنسبة للعنصر الأول، وفي حال تركه فارغاً فإن هذا يعني أن الملف موجود في نفس مجلد الـ Setup.exe

لا تنسونا من صالح دعائكم