

Tema 10. Creación de un autoinstalable

Formación específica, cursos verano 2008
ETS de Informática Aplicada
Universidad Politécnica de Valencia

Contenidos

- ◆ Patrones de implementación para .NET CF
- ◆ Generación de una distribución
- ◆ Ofuscación de código

Referencia básica:

- Application Security, Deployment and Management
(<http://msdn2.microsoft.com/en-us/library/aa127199.aspx>)

Empaquetado

- ◆ Primer paso necesario para poner a disposición del usuario una aplicación
- ◆ Esta fase incluye:
 - Definición de una carpeta de salida y de acciones de generación en los archivos del proyecto
 - Cambio de modo de generación del proyecto
 - Comprensión del funcionamiento de la caché de ensamblados global (GAC) en .NET CF
 - Empaquetado de la aplicación en un archivo .cab para su posterior implementación

Definición de carpeta de salida

- ◆ La carpeta de salida es la carpeta en la que se implementará el proyecto cuando se ejecute el fichero CAB
- ◆ Por defecto

The image shows a sequence of steps in Visual Studio to define the output folder for a project. On the left, the 'Dispositivos*' menu is open, with an arrow pointing to the 'Propiedades' option. In the center, the 'Opciones de implementación' dialog box is shown with the 'Carpeta de archivo de resultados' field set to '%CSIDL_PROGRAM_FILES%\TestApp'. An arrow points from this field to the 'Carpeta de archivo de resultados' dialog box on the right, which shows a list of folders with 'Carpeta Archivos de programa' selected.

Configuración: N/A Plataforma: N/A

Opciones de implementación

Dispositivo de destino: Pocket PC 2003 SE - Emulador

Carpeta de archivo de resultados: %CSIDL_PROGRAM_FILES%\TestApp

Implementar la versión más reciente de .NET Compact Framework (incluidos los Service Pack)

Firmar con Authenticode

Firmar el resultado del proyecto con este certificado

Seleccionar certificado...

Carpeta de archivo de resultados

Ubicación de resultados en el dispositivo:

(Valor predeterminado del dispositivo)

(Valor predeterminado del dispositivo)

Carpeta Datos de programa

Carpeta de datos de programas comunes

Carpeta Fuentes

Carpeta Mis documentos

Carpeta Archivos de programa

Carpeta raíz

Carpeta del menú Inicio

Carpeta Programas del menú Inicio

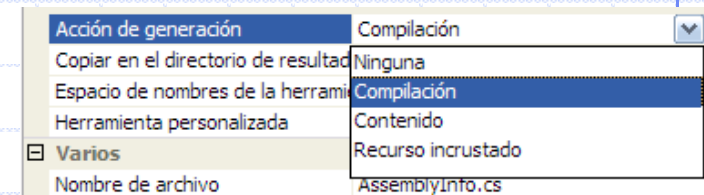
Carpeta de inicio del menú Inicio

Carpeta Windows

Acciones de generación

◆ Cada archivo se marca con una acción de generación

- Compilación (por defecto) – el archivo se compilará
- Contenido – empaquetado del archivo en el fichero cab resultante. Útil para la implementación de archivos de configuración XML y bases de datos SQL Server CE
- Ninguna – Se ignora. Útil para incluir documentación en el proyecto
- Recurso incrustado - Se incluye en el ensamblado ejecutable como un recurso. Esta acción permite que el código escrito extraiga el recurso mediante programación. Útil para empaquetar imágenes y archivos de secuencias de comandos.



```
private void btnEmbedded_Click(object sender, System.EventArgs e) {  
    Sound sound =  
        new Sound (Assembly.GetExecutingAssembly().GetManifestResourceStream("SoundSample.chimes.wav"));  
    sound.Play();  
}
```

Modos de generación

◆ Debug/Release

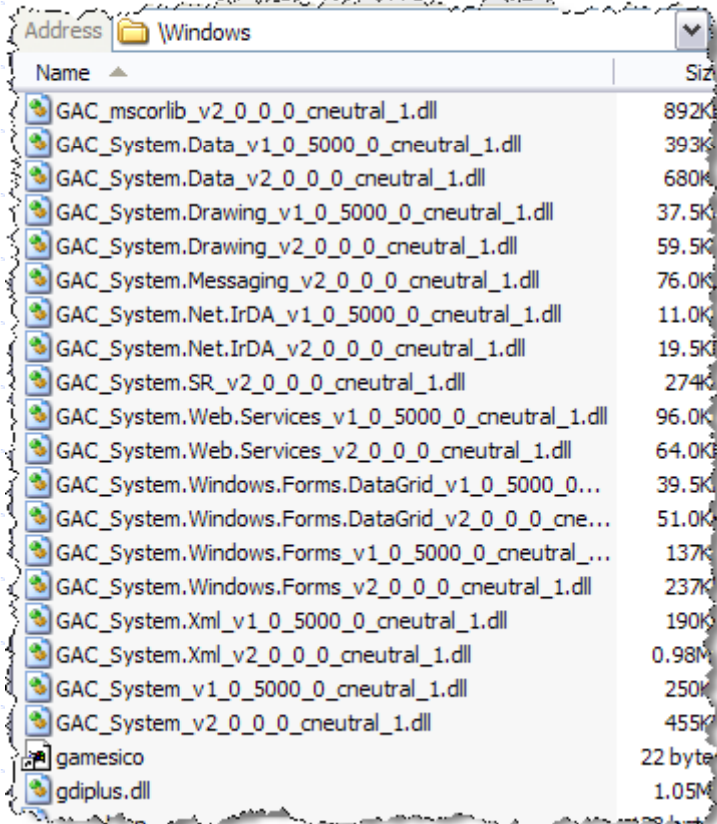
- El modo depuración sólo se utiliza durante la fase de desarrollo
- El modo Release (o de distribución) reduce el tamaño el ejecutable resultante y aumenta su velocidad de ejecución
 - ◆ Hacer siempre antes de generar el fichero CAB

◆ El modo de generación se puede cambiar en

- La barra de herramientas estándar de Visual Studio .NET
- El cuadro de diálogo Administrador de configuración accesible desde la opción Generar del menú de Visual Studio .NET

Uso de la caché GAC

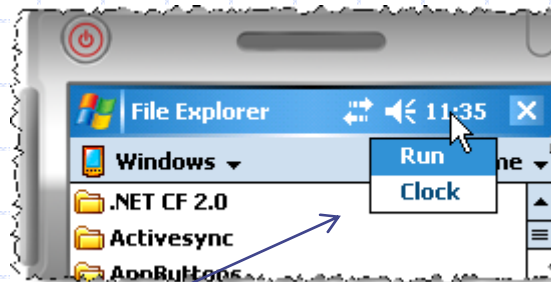
- ◆ GAC – Global Assembly Cache
 - Repositorio global que sirve para ahorrar espacio en los dispositivos móviles
- ◆ Los ensamblados contenidos en el GAC se almacenan en el directorio \Windows con el siguiente nombre:
 - GAC_<shortname>_v<maj>_<min>_<build>_<rev>_c<culture>_<ref>.dll



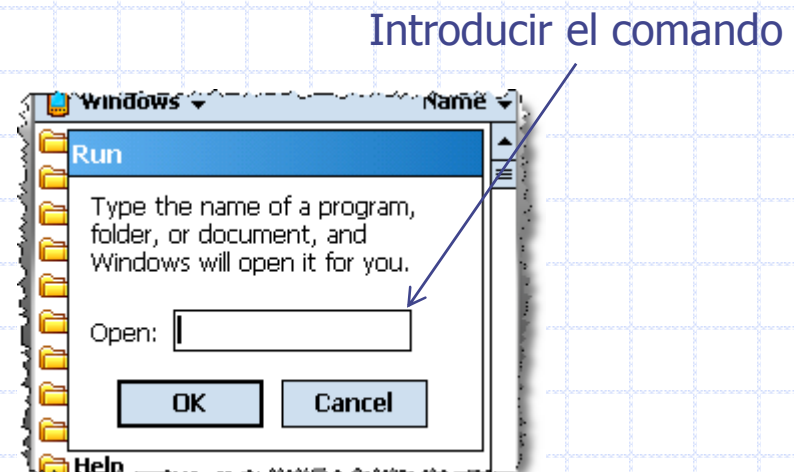
Name	Size
GAC_mscorlib_v2_0_0_0_neutral_1.dll	892K
GAC_System.Data_v1_0_5000_0_neutral_1.dll	393K
GAC_System.Data_v2_0_0_0_neutral_1.dll	680K
GAC_System.Drawing_v1_0_5000_0_neutral_1.dll	37.5K
GAC_System.Drawing_v2_0_0_0_neutral_1.dll	59.5K
GAC_System.Messaging_v2_0_0_0_neutral_1.dll	76.0K
GAC_System.Net.IrDA_v1_0_5000_0_neutral_1.dll	11.0K
GAC_System.Net.IrDA_v2_0_0_0_neutral_1.dll	19.5K
GAC_System.SR_v2_0_0_0_neutral_1.dll	274K
GAC_System.Web.Services_v1_0_5000_0_neutral_1.dll	96.0K
GAC_System.Web.Services_v2_0_0_0_neutral_1.dll	64.0K
GAC_System.Windows.Forms.DataGrid_v1_0_5000_0...	39.5K
GAC_System.Windows.Forms.DataGrid_v2_0_0_0_cne...	51.0K
GAC_System.Windows.Forms_v1_0_5000_0_neutral_...	137K
GAC_System.Windows.Forms_v2_0_0_0_neutral_1.dll	237K
GAC_System.Xml_v1_0_5000_0_neutral_1.dll	190K
GAC_System.Xml_v2_0_0_0_neutral_1.dll	0.98M
GAC_System_v1_0_5000_0_neutral_1.dll	250K
GAC_System_v2_0_0_0_neutral_1.dll	455K
gamesico	22 bytes
gdiplus.dll	1.05M

Insertar ensamblados en GAC

- ◆ Uso de la aplicación `\windows\cgacutil.exe`
 - `/i <assembly path>` (install)
 - `/u <assembly path>` (uninstall)
- ◆ Ejemplo: registrar `mylib.dll` en el GAC
 - `cgacutil /i \windows\mylib.dll`



Botón de acción + stylus en reloj
En el emulador: ctrl + stylus en reloj



Insertar ensamblados en GAC

- ◆ El fichero que aparecerá en \windows es *GAC_mylib_v1_0_0_0_cneutral_1.dll*
- ◆ A partir de ese momento cualquier aplicación podrá utilizar la librería mylib.dll sin necesidad de tenerlo en su directorio de trabajo
- ◆ Dicha librería puede compartirse entre varias aplicaciones
- ◆ Para desinstalar la librería:
 - `cgacutil /u \windows\mylib.dll`

Gestión del GAC vía programa

- ◆ Uso de System.Runtime.InteropServices
 - Permiten el uso de código no manejado (normalmente escrito en c/c++) desde .NET CF.

```
[StructLayout(LayoutKind.Sequential)]
struct PROCESS_INFORMATION
{
    public uint hProcess;
    public uint hThread;
    public uint dwProcessId;
    public uint dwThreadId;
}

[DllImport("coredll")]
private static extern int CreateProcess(
    string lpszImageName, string lpszCmdLine,
    IntPtr lpsaProcess,
    IntPtr lpsaThread, int fInheritHandles,
    uint fdwCreate,
    IntPtr lpvEnvironment, IntPtr lpszCurDir,
    IntPtr lpsiStartInfo,
    ref PROCESS_INFORMATION lppiProcInfo);
```

```
private static void StartProcess(string proc, string cmdLine)
{
    PROCESS_INFORMATION procInfo = new PROCESS_INFORMATION();
    CreateProcess(proc, cmdLine, IntPtr.Zero, IntPtr.Zero,
        0, 0, IntPtr.Zero, IntPtr.Zero, IntPtr.Zero, ref procInfo);
}

public static void GACInstall(string libName)
{
    StartProcess("cgacutil.exe", "-i " + libname);
}

public static void CUinstall(string libName)
{
    StartProcess("cgacutil.exe", "-u " + libname);
}
```

Creación de archivos CAB

- ◆ Un fichero .cab es un fichero comprimido que contiene
 - el ejecutable de la aplicación
 - los archivos que ésta requiere
 - las entradas de registro
 - Comandos para indicar al instalador del dispositivo (wceload.exe) cómo debe realizar la instalación de la aplicación
- ◆ Los ficheros cab se adecuan a la versión del SOP y de la arquitectura del dispositivo sobre el que se ejecutan
 - Pej: .NET CF v2.0 se distribuye en 7 ficheros .cab distintos

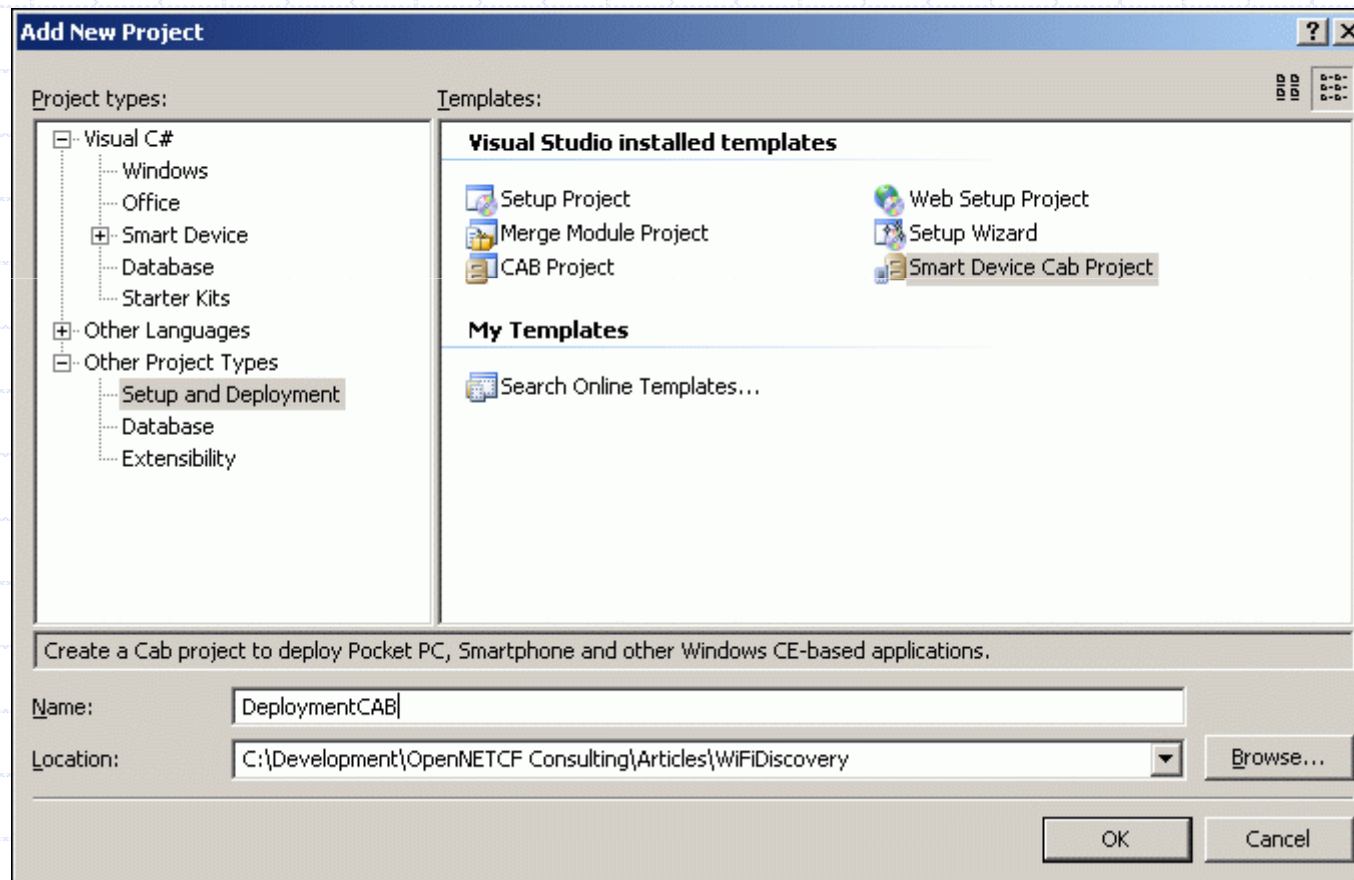
Una duda ...

- ◆ ¿No eran las aplicaciones .NET CF aplicaciones manejadas?
 - Recordad que esto significaba que las aplicaciones se distribuían en formato MSIL que era independiente del procesador
- ◆ ¿Por qué no utilizar eso mismo para los ficheros CAB?

Respuesta ...

- ◆ El fichero CAB contiene siempre un fichero llamado setup.dll con instrucciones especiales para el instalador
- ◆ Setup.dll se implementa utilizando código no manejado → debe adaptarse a las características de cada procesador y cada versión del sistema operativo
- ◆ Si la aplicación no requiere nada específico del dispositivo → un solo CAB es suficiente

Proyectos Smart Device CAB



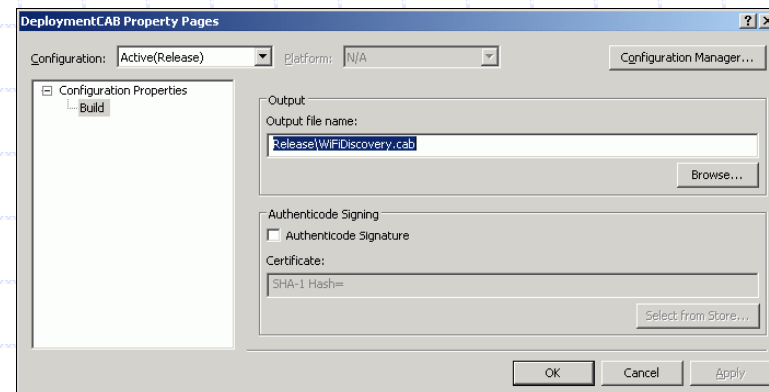
Agregar contenido al proyecto

- ◆ Es posible **Agregar** al proyecto
 - Ficheros individuales
 - Resultados de un proyecto
- ◆ Para ello en el explorador de soluciones hacer click con el botón derecho del ratón, seleccionar **Agregar**, y luego **Fichero o Resultados del proyecto**

Cambios en el CAB

- ◆ Cambiar el nombre del fichero CAB que se genera

- Botón derecho sobre nombre proyecto → Propiedades



- ◆ Cambiar las propiedades de implementación del proyecto
 - Son las que utiliza el instalador del ActiveSync

- ◆ En el proyecto CAB se dispone de 3 editores (propiedades, ficheros, registro)

The screenshot illustrates the three editors available for a CAB project in Visual Studio:

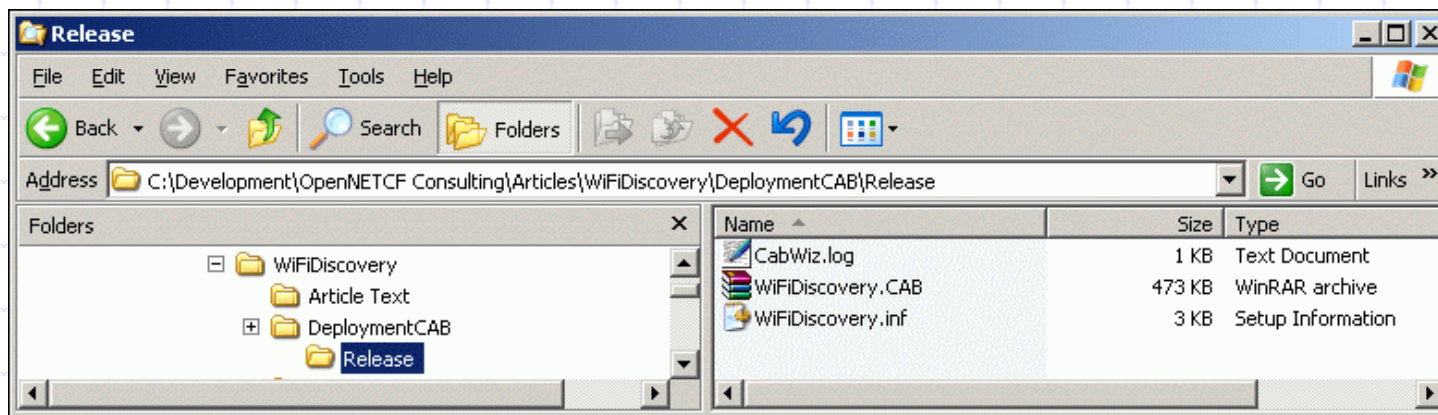
- Properties Editor:** Shows the 'Deployment Project Properties' for 'DeploymentCAB'. Key properties include:

CE Setup DLL	(None)
Compress	False
Manufacturer	OpenNETCF
NoUninstall	False
OSVersionMax	5.99
OSVersionMin	4.0
ProductName	WiFiDiscovery
- Registry Editor:** Shows the 'Registry (DeploymentCAB)' view, displaying a tree structure of registry keys such as 'HKEY_CLASSES_ROOT', 'HKEY_CURRENT_USER', and 'HKEY_LOCAL_MACHINE'. A 'Version' value is shown with a value of '1'.
- File System Editor:** Shows the 'File System (DeploymentCAB)' view, displaying a tree structure of files and folders, including 'Software', '%Manufacturer%', and 'WiFiDiscovery'.

Arrows indicate the flow of information between these editors, and a context menu is visible over the File System editor, showing options like 'Agregar carpeta especial' and 'Carpeta de la aplicación'.

Generación del fichero CAB

- ◆ La aplicación cabwiz.exe genera la siguiente salida:



¡Ojo! Las modificaciones de ficheros y registro sólo se aplican sobre el dispositivo cuando se instale la aplicación

Más información en : <http://msdn2.microsoft.com/en-us/library/ms924762.aspx>

Crear un MSI

- ◆ MSI: Microsoft installer package
- ◆ Realiza automáticamente la instalación de la aplicación siguiendo el procedimiento al que los usuarios están acostumbrados
- ◆ La instalación de las aplicaciones en los dispositivos requiere del uso de ActivenSync

Crear un MSI

- ◆ La instalación de una aplicación requiere del uso de CeAppMgr.exe
 - Forma parte del programa ActiveSync
 - Se encuentra instalado en el PC
- ◆ Registro de aplicaciones
 - Necesidad de proporcionar un fichero ini (puede ser generado manualmente)
- ◆ Más información en
 - <http://msdn.microsoft.com/en-us/library/ms838273.aspx>

Creación y prueba de ficheros .ini

- ◆ Mejor trabajar con CeAppMgr.exe en modo depuración, para ello:

```
[HKLM\Software\Microsoft\Windows CE Services\AppMgr]
"ReportErrors"=dword:1
```

- ◆ Ejemplo de fichero .ini para registrar una aplicación

Info para
CeAppMgr.exe

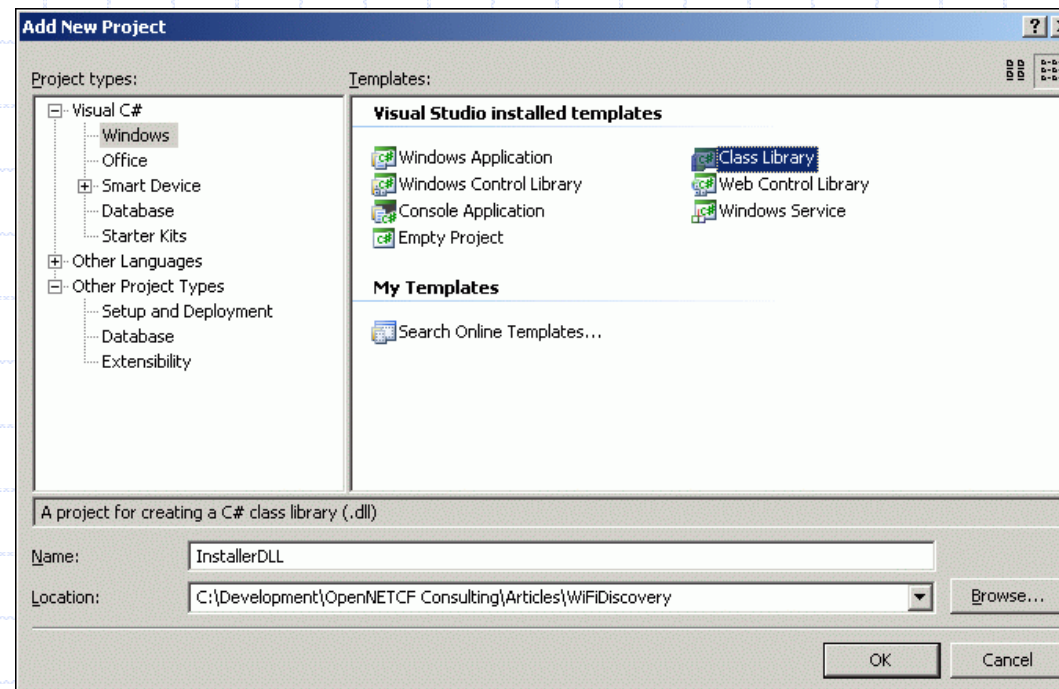
```
→ 01 [CEAppManager]
02 Version = 1.0
03 Component = OpenNETCF WiFiDiscovery
04
```

Info sobre
el componente
a instalar

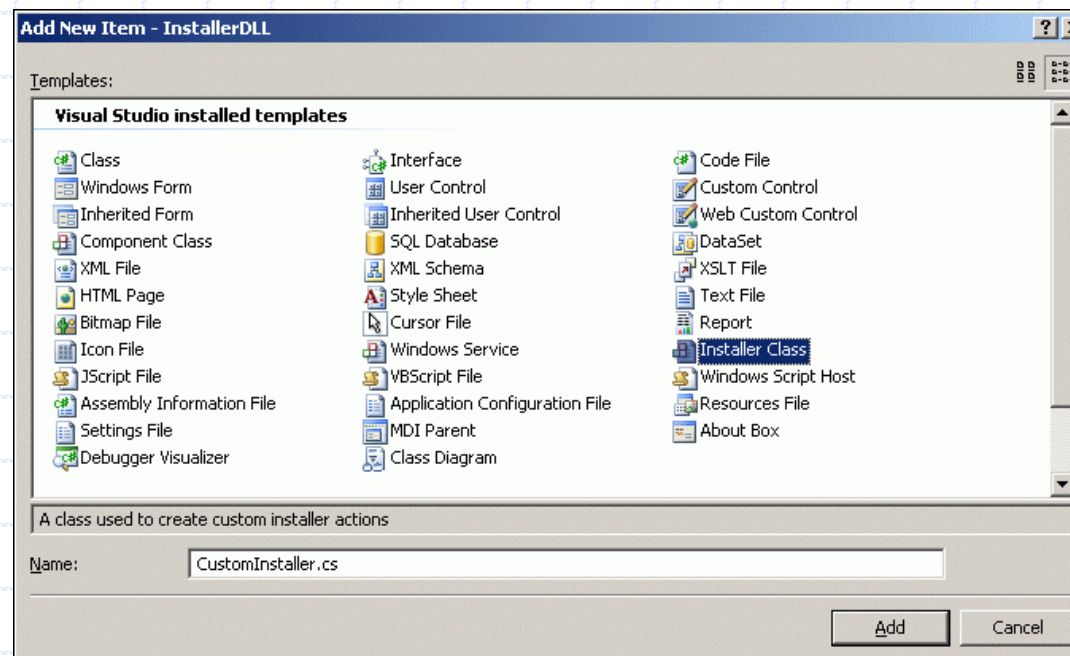
```
→ 05 [OpenNETCF WiFiDiscovery]
06 Description = Sample WiFi Network Discovery Application using the SDF
07 CabFiles = WiFiDiscovery.cab
```

- ◆ Las aplicaciones registradas en CeAppMgr.exe para instalación suelen almacenarse en un subdirectorio del directorio de instalación de ActiveSync
- ◆ Es necesario que el instalador determine dónde ha instalado el usuario el ActiveSync y guarde los ficheros a instalar dónde corresponda

- ◆ Creamos una dll que lance a ejecución la aplicación CeAppMgr.exe con el fichero .ini asociado



- ◆ La clase Class1 que se genera no se utilizará → Borrarla
- ◆ Se añadirá una clase que ejercerá de instalador (Explorador de soluciones → Añadir → Nuevo item)



◆ Eventos de instalación y desinstalación

- BeforeInstall – copiará los archivos al directorio de ActiveSync correspondiente y registrará la instalación con CeAppMgr.exe
- AfterInstall – borrar los ficheros del directorio de instalación
- BeforeUninstall – Elimina los archivos del directorio de ActiveSync

◆ Asociando manejadores a los eventos de interés

```
public CustomInstaller() //Lo hacemos en el constructor
{
    InitializeComponent();
    this.BeforeInstall +=
        new InstallEventHandler(CustomInstaller_BeforeInstall);
    this.AfterInstall +=
        new InstallEventHandler(CustomInstaller_AfterInstall);
    this.BeforeUninstall +=
        new InstallEventHandler(CustomInstaller_BeforeUninstall);
}
```

◆ Constantes que utilizaremos en el ejemplo

```
private const string CEAPPMGR_PATH =  
    @"SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\CEAPPMGR.EXE";  
private const string ACTIVESYNC_INSTALL_PATH =  
    @"SOFTWARE\Microsoft\Windows CE Services";  
private const string INSTALLED_DIR = "InstalledDir";  
private const string CEAPPMGR_EXE_FILE = @"CEAPPMGR.EXE";  
private const string CEAPPMGR_INI_FILE = @"WiFiDiscovery.ini";  
private const string APP_SUBDIR = @"\OpenNETCF WiFiDiscovery";  
private string TEMP_PATH =  
    Environment.SystemDirectory + @"\TEMP\WiFiDiscovery";
```

◆ Determinar dónde está instalado el ActiveSync

```
private string GetAppInstallDirectory()
{
    // Obtener el directori de instalación del programa ActiveSync
    RegistryKey keyActiveSync =
        Registry.LocalMachine.OpenSubKey(ACTIVESYNC_INSTALL_PATH);
    if (keyActiveSync == null)
    {
        throw new Exception("ActiveSync is not installed.");
    }
    // Definir el directorio de instalación partiendo del directorio de ActiveSync
    string activeSyncPath =
        (string)keyActiveSync.GetValue(INSTALLED_DIR);
    string installPath = activeSyncPath + APP_SUBDIR;
    keyActiveSync.Close();
    return installPath;
}
```

Manejadores de eventos

```
void CustomInstaller_BeforeInstall(object sender, InstallEventArgs e)
{
    // Determinar el directorio de instalación
    string installPath = GetAppInstallDirectory();
    // Crear el directorio destino
    Directory.CreateDirectory(installPath);
    // Copiar la aplicación al directorio destino
    foreach (string installFile in Directory.GetFiles(TEMP_PATH))
    {
        File.Copy(installFile, Path.Combine(installPath, Path.GetFileName(installFile)), true);
    }
    // Obtener el path al ceappmgr.exe
    RegistryKey keyAppMgr =
        Registry.LocalMachine.OpenSubKey(CEAPPMGR_PATH);
    string appMgrPath = (string)keyAppMgr.GetValue(null);
    keyAppMgr.Close();
    // Ejecuta CeAppMgr.exe para instalar los ficheros en el dispositivos
    System.Diagnostics.Process.Start(appMgrPath,
        "\"" + Path.Combine(installPath, CEAPPMGR_INI_FILE) + "\"");
}
```

Manejadores de eventos

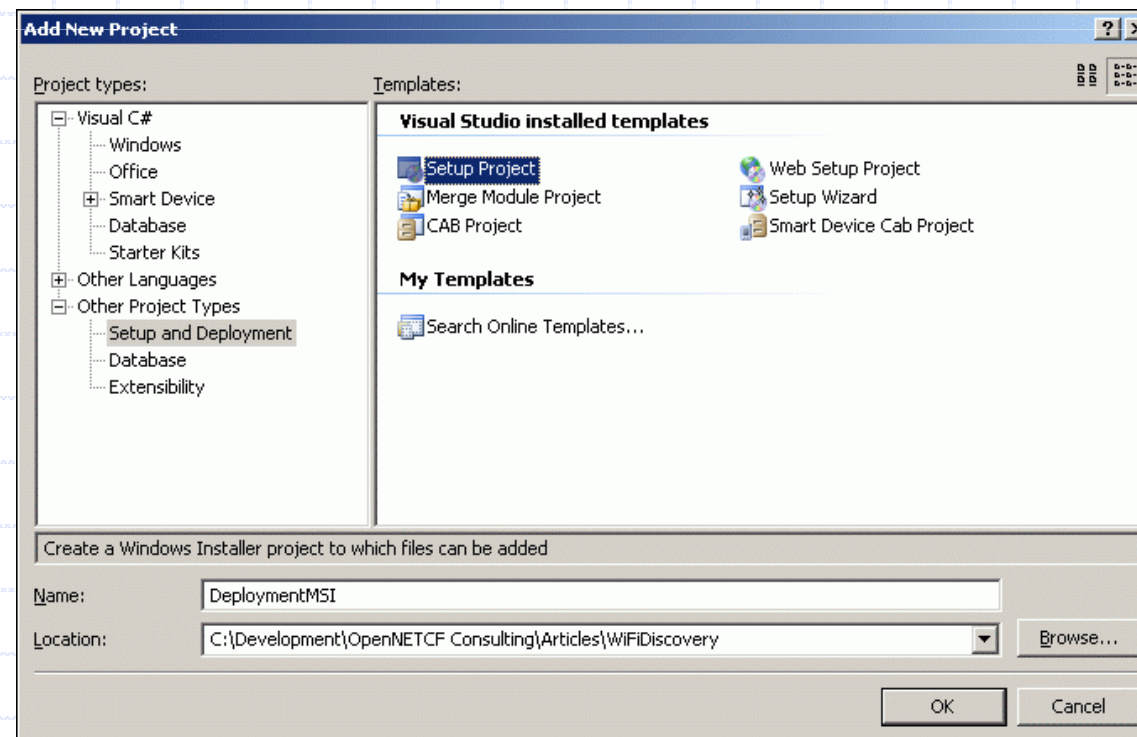
```
void CustomInstaller_AfterInstall(object sender, InstallEventArgs e)
{
    // Delete the temp files
    foreach (string tempFile in Directory.GetFiles(TEMP_PATH))
    {
        File.Delete(tempFile);
    }
}
```

Manejadores de eventos

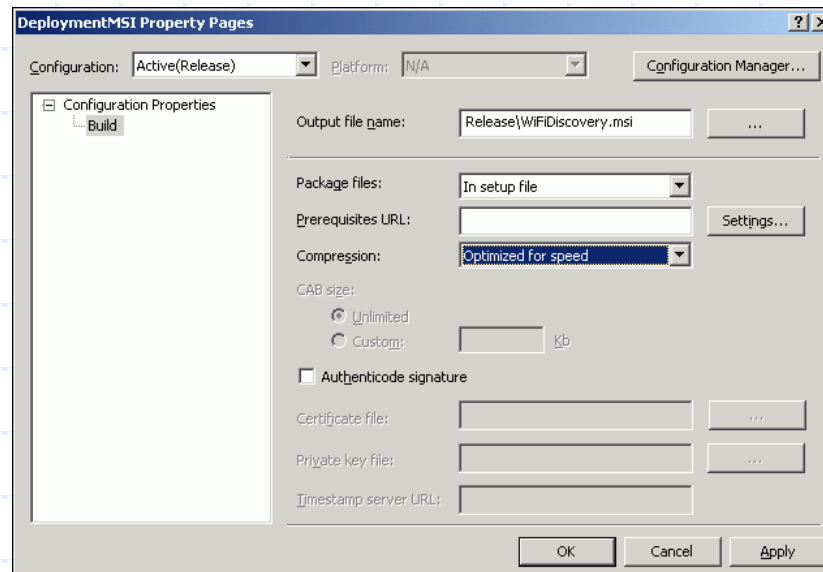
```
void CustomInstaller_BeforeUninstall(object sender, InstallEventArgs e)
{
    // Find where the application is installed
    string installPath = GetAppInstallDirectory();
    // Delete the files
    foreach (string appFile in Directory.GetFiles(installPath))
    {
        File.Delete(appFile);
    }
    // Delete the folder
    Directory.Delete(installPath);
}
```

Creación del MSI

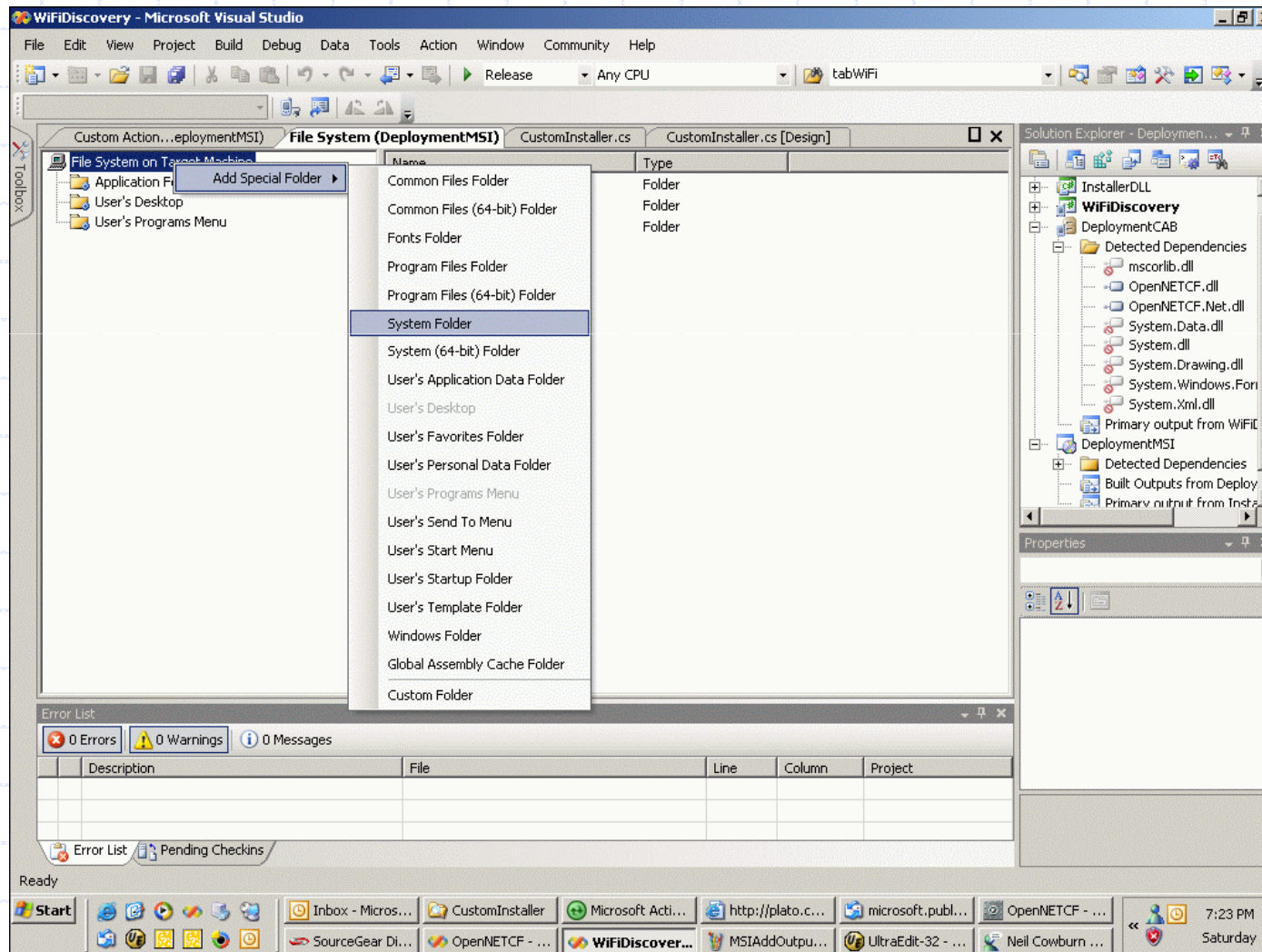
- ◆ Compilar la clase InstallerDLL implementada y generar la dll correspondiente
- ◆ Crear ahora un proyecto de instalación



- ◆ Actualizar las propiedades del proyecto
 - Pej: Nombre de la aplicación, compañía, URL de soporte
- ◆ Proporcionar al fichero MSI el nombre deseado

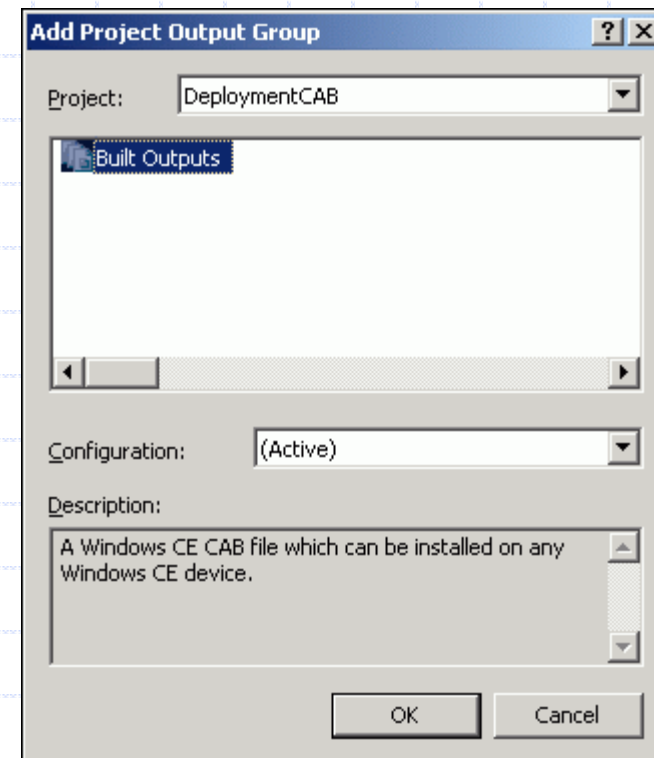


Crear un directorio temporal



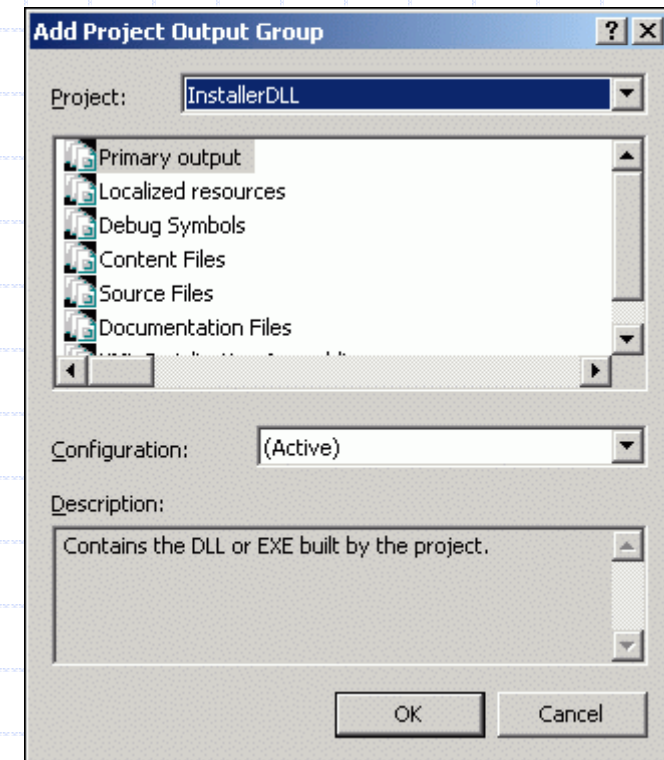
Añadir el CAB al MSI

- ◆ Se puede añadir un fichero específico o el resultado de otro proyecto



Asociar el MSI con la dll

- ◆ Abrir el editor de acciones personalizadas
- ◆ Agregar acción personalizada
- ◆ Seleccionar el directorio temporal ya creado
- ◆ Agregar resultados



Resultado

The screenshot displays the Visual Studio IDE for a project named 'WiFiDiscovery'. The main window shows the 'Custom Actions' configuration for a 'DeploymentMSI' project. The 'Custom Actions' list is expanded to show the following actions and their outputs:

- Install: Primary output from InstallerDLL (Active)
- Commit: Primary output from InstallerDLL (Active)
- Rollback: Primary output from InstallerDLL (Active)
- Uninstall: Primary output from InstallerDLL (Active)

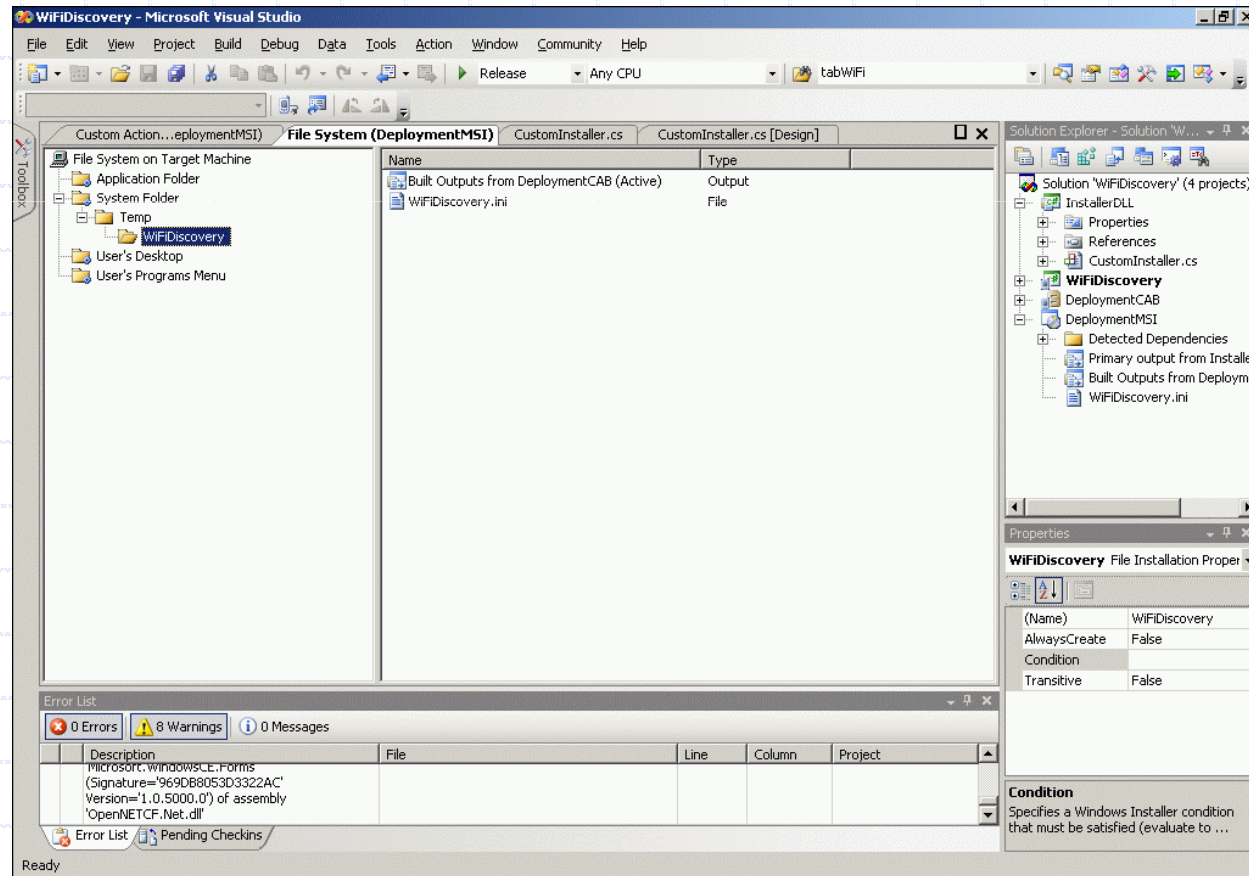
The Solution Explorer on the right shows the project structure, including 'Detected Dependencies' and 'Primary output from WiFi'. The Properties window at the bottom right shows the 'Condition' property for the 'Primary output'.

Property	Value
Condition	(Dependencies)
Exclude	False
ExcludeFilter	(None)
Folder	Temp
Hidden	False
KeyOutput	(Outputs)

The Error List at the bottom shows 0 Errors, 0 Warnings, and 0 Messages.

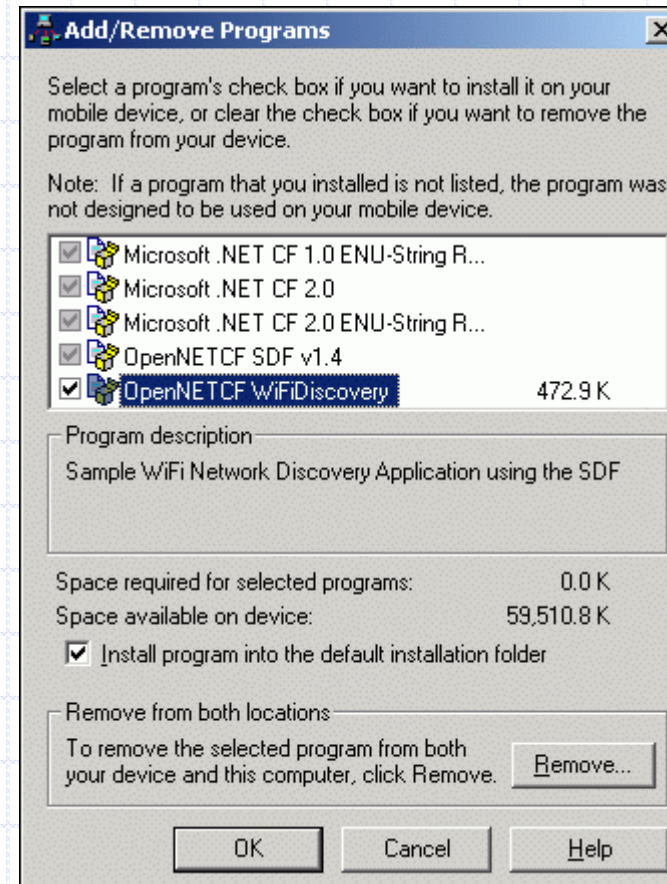
Añadir finalmente el .ini

◆ Añadir el fichero explícitamente



Probar el instalador

- ◆ La PDA o un emulador deben estar conectados al PC
- ◆ Seguir las instrucciones



Actualización de aplicaciones

- ◆ Parte de su ciclo de vida. P.ej.:
 - Corrección de errores de programación
 - Introducción de nuevas funcionalidades
 - Adecuación a nuevos requerimientos
- ◆ Posibilidades
 - Publicar nuevas versiones de la aplicación en la web (intervención del usuario)
 - La aplicación se actualiza sola

Aplicaciones auto-actualizables

- ◆ ¿Qué necesitamos?
 - Un servidor que sirva las actualizaciones (por ejemplo un servicio web + IIS)
 - Un applet de actualización implementado como una aplicación .NET CF basada en formularios

Veamos el servidor web

◆ Capacidades

- Comunicar a los clientes la disponibilidad de una actualización
 - ◆ Gestión local de las actualizaciones disponibles (en el ejemplo se utilizará un fichero XML)
- Servir el CAB con la nueva actualización

Gestión de versiones

◆ Objetivos

- Poder proporcionar actualizaciones adaptadas a cada plataforma y arquitectura
- Utilizar un mismo servicio web para todas ellas

◆ Sintaxis

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<updateinfo>
```

```
  <downloadmodule plat="PPC" arch="ARM" name="TESTAPP" action="cab">
```

```
    <version maj="1" min="0" bld="1363"/>
```

```
    download/TestApp_PPC.ARM.CAB
```

```
  </downloadmodule>
```

```
  <downloadmodule plat="PPC" arch="ARMV4" name="TESTAPP" action="cab">
```

```
    <version maj="1" min="0" bld="1363"/>
```

```
    download/TestApp_PPC.ARMV4.CAB
```

```
  </downloadmodule>
```

```
</updateinfo>
```

} Fichero CAB con MYAPP
version 1.0.1363 producida
para arquitecturas ARM.

Servicio web

◆ Estructura devuelta por el servicio

```
public class UpdateInfo {  
    public string Url;  
    public bool IsAvailable;  
    public string newVersion;  
}
```

◆ Interfaz del servicio ofrecido

```
public UpdateInfo GetUpdateInfo( string nombre_app,  
                                string arquitectura,  
                                int version_maj,  
                                int version_min,  
                                int version_bld);
```

Implementación

◆ Carga del fichero de gestión de versiones

```
UpdateInfo ui = new UpdateInfo ();
Version ver = new Version(maj, min, bld);
XmlDocument xmlUpdateCfg = new XmlDocument();
//Attempt to load xml configuration
try {
    xmlUpdateCfg.Load(Context.Request.MapPath("updatecfg.xml"));
} catch(Exception ex) {
    ui.IsAvailable = false; return ui;
}
```

Implementación

◆ Búsqueda de la entrada solicitada

```
string xq = string.Format(
    "-//downloadmodule[@arch=\"{0}\" and @name=\"{4}\" " +
    "and ( version/@maj>{1} or version/@maj={1} " +
    "and (version/@min > {2} or version/@min = {2}) " +
    "and version/@bld > {3})]]",
    arch, maj, min, bld, name);
XmlElement nodeUpdate =
    (XmlElement)xmlUpdateCfg["updateinfo"].SelectSingleNode(xq);
if ( nodeUpdate == null ) {
    ui.IsAvailable = false;
    return ui;
}
```

Implementación

◆ Búsqueda de la entrada solicitada

```
if ( nodeUpdate == null ) {
    ui.IsAvailable = false;
    return ui;
}
ui.IsAvailable = true;
ui.newVersion = new Version(string.Format("{0}.{1}.{2}",
    nodeUpdate["version"].Attributes["maj"].Value,
    int.Parse(nodeUpdate["version"].Attributes["min"].Value),
    int.Parse(nodeUpdate["version"].Attributes["bld"].Value))).ToString();
string srv = Context.Request.ServerVariables["SERVER_NAME"];
string path = Context.Request.ServerVariables["SCRIPT_NAME"];
ui.Url = string.Format("http://{0}{1}", srv, path);
ui.Url = ui.Url.Substring(0, ui.Url.LastIndexOf("/"));
ui.Url += "/" + nodeUpdate.InnerText.Trim();
return ui;
```

Parte del cliente

◆ XML de configuración del cliente

```
<?xml version="1.0" encoding="utf-8" ?>
<updateinfo>
  <checkassembly name="MyApp.EXE"/>
  <remoteapp name="MyApp"/>
  <service url="http://updates.mycompany.com/UpdateAgent/Agent.asmx"/>
</updateinfo>
```

◆ Lectura del fichero

```
<?xml version="1.0" encoding="utf-8" ?>
<updateinfo>
  <checkassembly name="MyApp.EXE"/>
  <remoteapp name="MyApp"/>
  <service url="http://updates.mycompany.com/UpdateAgent/Agent.asmx"/>
</updateinfo>
```

Búsqueda de actualizaciones

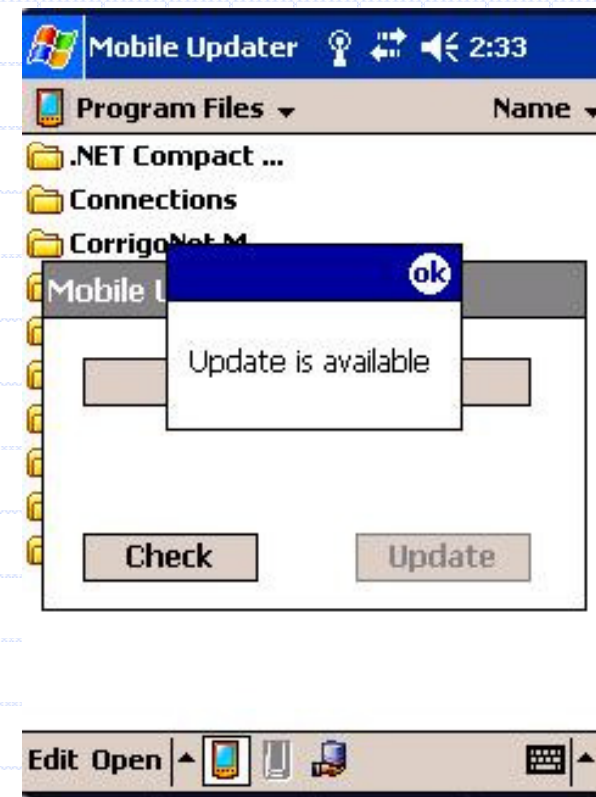
◆ Obtención de la URL del servidor

```
Agent agent = new Agent();  
agent.Url = xmlConfig["updateinfo"]["service"].GetAttribute("url");
```

◆ Obtención disponibilidad de actualización

```
string platform = Utils.GetPlatformType();  
string arch = Utils.GetInstructionSet();  
string appName = xmlConfig["updateinfo"]["remoteapp"].GetAttribute("name").ToUpper();  
UpdateInfo info = agent.GetUpdateInfo(appName, platform, arch,  
    name.Version.Major, name.Version.Minor, name.Version.Build);
```

◆ Información obtenida en UpdateInfo.IsAvailable



Descarga de la actualización

```
private HttpWebRequest m_rec;  
  
private void btnUpdate_Click(object sender, System.EventArgs e)  
{  
    m_req = (HttpWebRequest)HttpWebRequest.Create(UpdateUrl);  
    m_req.BeginGetResponse(new AsyncCallback(ResponseReceived), null);  
    btnCheck.Enabled = false;  
    Cursor.Current = Cursors.WaitCursor;  
}
```

Descarga de la actualización

```
void ResponseReceived(IAsyncResult res)
{
    try
    {
        m_resp = (HttpWebResponse)m_req.EndGetResponse(res);
    }
    catch(WebException ex)
    {
        MessageBox.Show(ex.ToString(), "Error");
        return;
    }
    // Allocate data buffer
    dataBuffer = new byte[ DataBlockSize ];
    // Set up progress bar
    maxVal = (int)m_resp.ContentLength;
    pbProgress.Invoke(new EventHandler(SetProgressMax));
    // Open file stream to save received data
    m_fs = new FileStream(GetCurrentDirectory() + @"\download.cab",
        FileMode.Create);
    // Request the first chunk
    m_resp.GetResponseStream().BeginRead(dataBuffer, 0, DataBlockSize,
        new AsyncCallback(OnDataRead), this);
}
```

Descarga de la actualización

```
void OnDataRead(IAsyncResult res){
    // How many bytes did we get this time
    int nBytes = m_resp.GetResponseStream().EndRead(res);
    // Write buffer
    m_fs.Write(dataBuffer, 0, nBytes);
    // Update progress bar using Invoke()
    pbVal += nBytes;
    pbProgress.Invoke(new EventHandler(UpdateProgressValue));
    // Are we done yet?
    if ( nBytes > 0 ){
        // No, keep reading
        m_resp.GetResponseStream().BeginRead(dataBuffer, 0,
            DataBlockSize, new AsyncCallback(OnDataRead), this);
    }
    else{
        // Yes, perform cleanup and update UI.
        m_fs.Close();
        m_fs = null;
        this.Invoke(new EventHandler(this.AllDone));
    }
}
```

Al terminar la descarga

- ◆ La instalación del fichero CAB puede lanzarse utilizando los servicios de interoperación con código c/c++

- ◆ Invocar la función ShellExecute

(http://www.opennetcf.org/Forums/topic.asp?TOPIC_ID=263)



Ofuscación de código

- ◆ Problema de MSIL => poco seguro utilizando un decompilador
 - Por ejemplo: Anakrino
- ◆ Solución: Utilizar una herramienta de ofuscación de código para .NET
 - PreEmptive Dotfuscator Community Edition (<http://www.preemptive.com/dotfuscator.html>)

Ejemplo

```
private void CalcPayroll(SpecialList employeeGroup)
{
    while(employeeGroup.HasMore())
    {
        employee = employeeGroup.GetNext(true);
        employee.UpdateSalary();
        DistributeCheck(employee);
    }
}
```

Ofuscación

Código
obtenido

Ingeniería
inversa

```
private void a(a b) {
    while(b.a()) {
        a = b.a(true);
        a.a();
        a(a);
    }
}
```