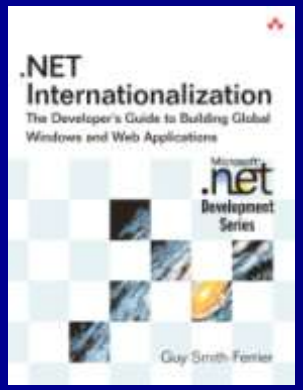
Using ClickOnce and XBAPs To Deploy Windows Forms and WPF Applications

Guy Smith-Ferrier
guy@guysmithferrier.com
Blog: http://www.guysmithferrier.com

About…

- Author of .NET Internationalization
  - Visit http://www.dotneti18n.com to download the complete source code



- The .NET Developer Network
  - http://www.dotnetdevnet.com
  - Free user group for .NET developers, architects and IT Pros based in Bristol

ClickOnce Vision

- To bring the ease and reliability of web application deployment to Windows Forms applications

Agenda

- ClickOnce and Windows Forms
- ClickOnce and WPF
- XAML Browser Applications (XBAPs)
- Windows Forms, WPF and XBAP Comparison
- XBAPs and Silverlight Comparison

Information Sources

- Smart Client Deployment with ClickOnce, Brian Noyes, Addison Wesley
- Deploying .NET Applications, Sayed Y. Hashimi and Sayed I. Hashimi, Apress
- Windows Presentation Foundation Unleashed, Adam Nathan, Sams Publishing
- Pro WPF, Mathew MacDonald, Apress
- Saurabh Pant's blog
  - blogs.msdn.com/saurabh/archive/tags/ClickOnce/default.aspx

Online Example

- Create a Windows Forms application and call it CustomerCare1
- Set Form1.Text to "Customer Care" and add a button
- Select Build | Publish CustomerCare1
  - Click Next
  - Select the "No" radio button
  - Click Next
  - Click Finish
- In the CustomerCare1 web page click on the Run CustomerCare1 button
  - In the Security Warning dialog click on Run

Online Example (continued)

- Close the application
- Click on the Run button again and observe that the application doesn't show the "Downloading" dialog and it starts up faster
- Close the application and close the web page
- Add a button to CustomerCare1, publish it again and click on the Run button
  - Observe that the application shows the "Downloading" dialog as it downloads the updated copy

How Does It Work ?

- The Publish Wizard:-
  - creates a new Virtual Directory in IIS using the name of the application (e.g. CustomerCare1)
  - creates a new folder beneath the Home Directory (e.g. C:\Inetpub\wwwroot\CustomerCare1)
    - adds a customized setup.exe (the Bootstrapper) to the new folder
    - adds a Publish.htm page which is shown to the user
    - adds a deployment manifest (e.g. CustomerCare1.application)
    - adds a version-specific deployment manifest (e.g. CustomerCare1_1_0_0_0.application)
  - creates a new folder for the latest version of the application (e.g. CustomerCare1_1.0.0.0)
    - adds the application (e.g. CustomerCare1.exe.deploy)
    - adds an application manifest (e.g. CustomerCare1.exe.manifest)
- When the application is run it is cached so it runs faster next time

How Does Publish.htm Work ?

- Publish.htm includes:-
  - a Button called InstallButton with an href of "setup.exe"
  - a section called BootstrapperSection
- Publish.htm includes JavaScript which is automatically executed to determine if the .NET Framework 2.0 is installed
  - If it is installed:-
    - BootstrapperSection.style.display is set to none
    - InstallButton.href is set to the application's deployment manifest (e.g. CustomerCare1.application)

ClickOnce Client Requirements

- For browser access the client must support URL activation e.g.
  - Internet Explorer
    - Non-Internet Explorer browsers (e.g. Mozilla) will not work because they look for the application files in the same location as the manifest files
    - For FireFox use FFClickOnce
      - http://www.softwarepunk.com/ffclickonce/
  - Outlook
  - MSN Explorer
  - AOL

ClickOnce Client Requirements (continued)

- Before the application can be run the client must have:-
  - Windows 98 or higher
  - the .NET Framework 2.0
  - The Windows Installer 2.0
  - Whatever pre-requisites have been dictated by the publisher
- ClickOnce can install all of these using the ClickOnce Bootstrapper
  - The Bootstrapper requires Administrator access to install pre-requisites (but not to install the ClickOnce application)

Online And Offline Example

- Create a Windows Forms application and call it CustomerCare2
- Set Form1.Text to "Customer Care 2" and add a button
- Select Build | Publish CustomerCare2
  - Click Next
  - Select the "Yes" radio button
  - Click Next
  - Click Finish
- In the CustomerCare2 web page click on the Install button
  - In the Security Warning dialog click on Install

Online And Offline Example (continued)

- Show that an item has been added to the Start menu
- Show that an item has been added to Add/Remove Programs
- Close the application and close the web page
- Add a button to the form and publish it again
- In Internet Information Services select the Default Web Site, right click and select Stop
- In Windows select Start | All Programs | <Company Name> | CustomerCare2
  - The original, 'cached' program is run and not the new version

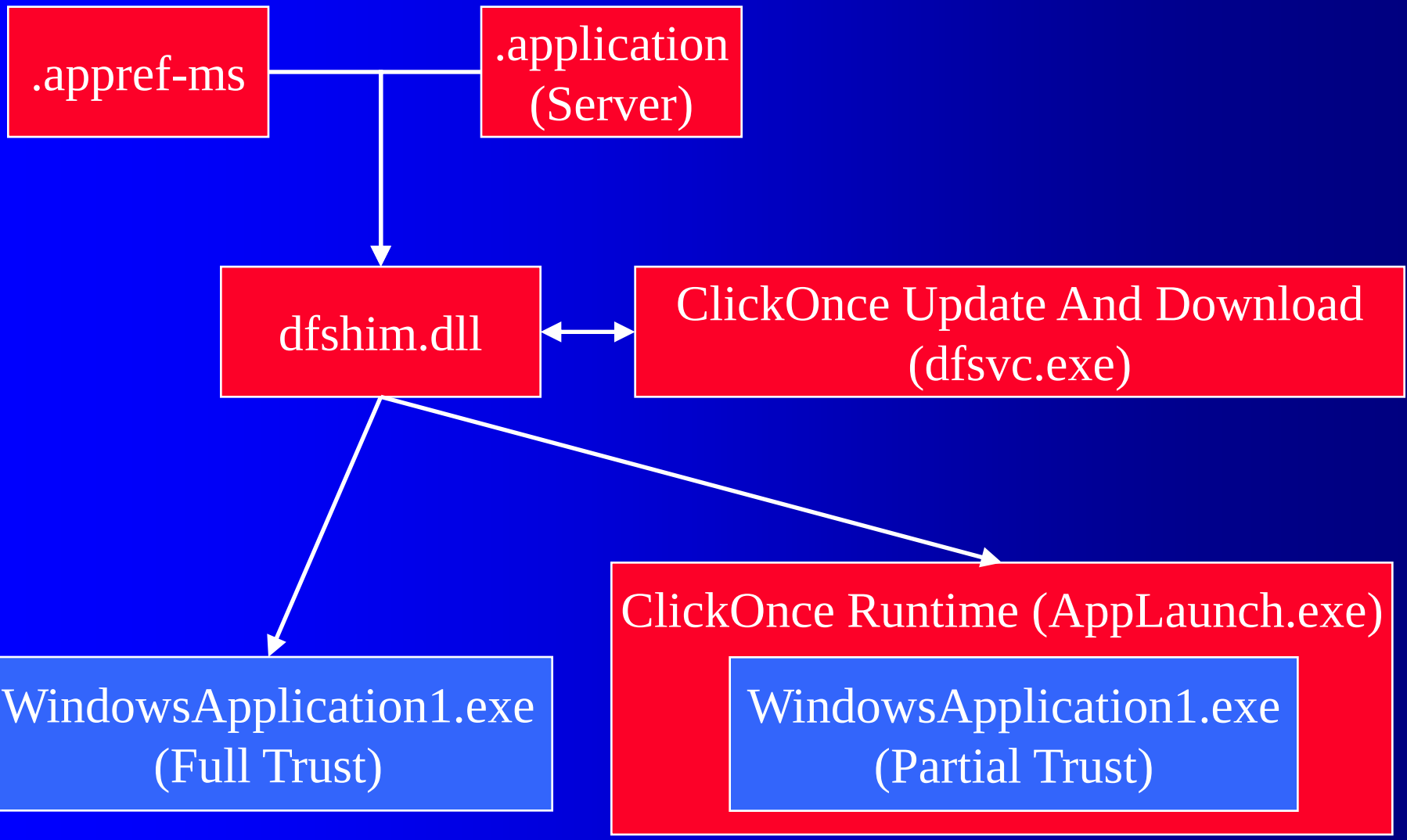
Online And Offline Example (continued)

- Close the application
- Restart the Default Web Site
- Run the application
  - In the Update Available dialog click Ok and the updated application is run

How Does It Work ?

- The Publish Wizard performs the same steps as before except:-
  - Publish.htm says "Install" instead of "Run" (but the functionality is identical)
- The shortcut added to the Start menu is a .appref-ms file
  - .appref-ms files are MIME files and are automatically executed by the operating system's MIME filter
  - The .appref-ms file contains the URL to the .application file (the deployment manifest) on the web server
  - .application and .appref-ms extensions invoke COM routines in dfshim.dll which launch the ClickOnce update and download component
  - The ClickOnce update and download component (dfsvc.exe) manages the deployment and update of the application
    - dfsvc.exe allegedly times out after 15 minutes of inactivity
  - If the application is partially trusted the ClickOnce runtime (AppLaunch.exe) is run to handle the sandboxing and execution of the application
  - Otherwise the application is run normally

How Does It Work ? (continued)



The ClickOnce Cache

- Online and installed applications are held on the client computer in the ClickOnce cache
- The cache is in <Documents>\Local Settings\Apps
  - Windows XP:-

```
\Documents and Settings\Administrator\Local Settings\Apps
```

  - Windows Vista:-

```
\Users\Guy Smith-Ferrier\Local Settings\Apps
```

- The cache contains the complete application
  - The application's assemblies, its configuration files, its data
- Applications cannot be installed outside of the cache
  - Applications cannot be installed to a specific location
- There is no publicly available "ClickOnce Cache API"
  - The cache can be managed by regular file I/O methods

The ClickOnce Cache Size

- The maximum size for all online applications is 100Mb
  - This can be changed in the registry:-
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Deployment\OnlineAppQuotaInKB
  - The value is in Kb so 200Mb is 204800
- There is no restriction on installed applications
  - Installed applications do not count towards the online applications maximum size
- The cache contains the current and previous version:-
  - Cached online applications are aged
    - When the cache is full the least recently used application(s) is/are deleted
      - Old online apps can be programmatically purged using file delete methods
  - Cached installed applications are not aged

ClickOnce Manifest Files

	Deployment Manifest	Application Manifest
Description	Contains information about the application including the current version number	Contains information about the files, dependencies and security requirements of a specific version
Responsibility	Administrator	Developer
Extension	.application	.exe.manifest
Location	The published folder e.g. C:\Inetpub\wwwroot\WindowsApplication1	The specific version of the published application e.g. \Inetpub\wwwroot\WindowsApplication1\ WindowsApplication1_1.0.0.3
Signing	Deployment manifest files are always signed (using XMLDSIG) and cannot be tampered with	Application manifest files are always signed (using XMLDSIG) and cannot be tampered with

Manifest Generation and Editing Tool (MAGE)

- The Manifest Generation and Editing Tool (MAGE) is a .NET Framework 2.0 SDK tool for creating and editing ClickOnce manifest files
  - There are two versions:-
    - Mage.exe is a command line tool
    - MageUI.exe is a GUI
- MAGE and Visual Studio 2005 offer similar functionality

Command Line Parameters

- ClickOnce applications can accept command line parameters as arguments in the application manifest's URL

  `http://localhost/WindowsApplication1.application?Culture=en-GB`

- Edit the .appref-ms file using NotePad
  - In Windows XP:-

    <Documents>\<UserName>\Start Menu\Programs\<CompanyName>\WindowsApplication1
  - In Windows Vista:-

    \Users\<UserName>\Start Menu\Programs\WindowsApplication1
- Command line parameters must be enabled
  - In the project's Publish properties click the Options… button and check "Allow URL parameters to be passed to application"

Programmatic Updating

- Turn off automatic updating
  - In the Publish properties, click the Updates… button and uncheck the "Application should check for updates" checkbox
- Add a reference to System.Deployment.dll
- Add the following directive to the code:-

```
using System.Deployment.Application;
```

- Add a button to the main form with the following code:-

Programmatic Updating (continued)

```
ApplicationDeployment applicationDeployment =
   ApplicationDeployment.CurrentDeployment;
if (applicationDeployment.CheckForUpdate())
{
    applicationDeployment.Update();
    Application.Restart();
}
```

- Publish the application and install it
- Change the application and publish it again
- Run the application and show that the new version is not automatically downloaded
- Click on the button to update the application

Programmatic Updating (continued)

- Change the button's code to:-

```
ApplicationDeployment deployment =
    ApplicationDeployment.CurrentDeployment;
UpdateCheckInfo updateCheckInfo = deployment.CheckForDetailedUpdate();
if (updateCheckInfo.UpdateAvailable)
{
    if (MessageBox.Show("An update is available" +
        System.Environment.NewLine + "Version " +
        updateCheckInfo.AvailableVersion + " (" +
        updateCheckInfo.UpdateSizeBytes.ToString() + " bytes)" +
        System.Environment.NewLine +
        "Update now ?", "", MessageBoxButtons.YesNo) ==
        DialogResult.Yes)
    {
        deployment.Update();
        Application.Restart();
    }
}
```

Uses For Programmatic Updating

- Updates on demand
  - The user can check for updates manually
- Download load balancing
  - The client could contain logic which determines which day or hour the user checks for downloads based upon data specific to the client (e.g. IP address, first letter of user name)
- Beta programs
  - The client could compare the new minor version number with the old and only download it if the client has been registered as a beta client

Manifest Signing

- Application and deployment manifests are signed using an XML Digital Signature (XMLDSIG)
  - Manifests which are altered without being signed again result in the error:-

```
The manifest XML signature is invalid
```

- Assemblies are signed using the publisher certificate (.pfx)
  - You cannot sign assemblies using a strong name (.snk)
  - Signing assemblies with a strong name would be bad in a ClickOnce application:-
    - An assembly which uses a signed assembly must be rebuilt to use the new signed assembly when the signed assembly is changed
      - Changing one assembly could mean rebuilding many assemblies and therefore cause unnecessary downloads

ClickOnce Security Zones

- ClickOnce applications adopt the security zone that they are installed from
  - Not the location they are run from

ClickOnce Application	Security Zone
Online only	Internet
Online and offline installed from web	Internet
Online and offline installed from file server	Local Intranet
Offline installed from CD-ROM	Local Computer (Full Trust)

Calculating Security Permissions

- You can calculate your applications' security permissions in two ways:-
  - In Visual Studio open the Security tab of the project's Properties
    - Click on the Calculate Permissions… button
  - In a command prompt run permcalc.exe:-

```
permcalc WindowsApplication1.exe
```

Debug In Zone

- If you build, publish and debug your application on a single machine your debugging experience will be unrealistic
  - You will be debugging with Full Trust
- Visual Studio can set the permissions which are used during debugging to a set of permissions other than full trust
  - In Solution Explorer right click the project and select Properties
    - Select the Security tab and click "Enable ClickOnce Security Settings"
    - In the "Zone your application will be installed from" combo box set the settings to either Local Intranet or Internet

Trust Licenses

- Trust licenses allow you to create ClickOnce applications which are trusted without the user being involved with the trust process
  - The benefit is that you can deploy ClickOnce applications which have elevated security without your users having to answer trust questions

The Trusted Application Deployment Process

- The user's machine is configured to accept trust licenses from the given trust license issuer
  - This is a one-time touch to the user's machine using the .NET 2.0 Configuration Tool
- The developer requests a trust license (.tlic file) from the trust issuer and gives the issuer the application manifest (which includes the requested permissions)
- The trust issuer returns a trust license to the developer
- The developer signs the application with the trust license

ClickOnce vs The Windows Installer

	ClickOnce	The Windows Installer
Specify The Installation Folder	No	Yes
Install For All Users	No	Yes
Create File Associations	No	Yes
Create Shell Extensions	No	Yes
Install .NET Components In The GAC	No	Yes
Install COM Components In The Registry	No	Yes
Run Custom Actions	No	Yes
Write To Registry	No	Yes
Administer ODBC	No	Yes
Create And Apply .msp Patches	No	Yes

Installing ClickOnce Applications On Vista The Problem

- The ClickOnce Bootstrapper is called setup.exe
- Vista treats programs called setup.exe as installation programs
  - Vista demands Administrator privileges
  - (The ClickOnce setup.exe does not need Administrator privileges)
- The UAC triggers permission elevation
  - If the account has Administrator privileges then a dialog box is displayed
  - If the account does not have Administrator privileges then the user must supply Administrator credentials
    - Consequently the application is installed under the Administrator's account

Installing ClickOnce Applications On Vista The Solution

- Either:-
  - 1. Disable the UAC (not recommended)
  - 2. Get users to invoke the .application file instead of setup.exe
    - Only works if .NET Framework 2.0 is already installed on client
  - 3. Rename setup.exe
    - Must use a resource editor to rename the setup resource inside setup.exe to the same name
  - 4. Use .NET Framework 2.0 SP2 or .NET Framework 3.5

ClickOnce And .NET Framework 2.0 SP2 And .NET Framework 3.5

- The following enhancements are in the .NET Framework 2.0 SP2 and .NET Framework 3.5 version of ClickOnce:-
  - Support for FireFox
    - API exposed to allow developers to create parity with IE
  - Published ClickOnce applications can be copied to a new location
    - The publish location is no longer fixed when the manifest is signed
  - Authentication certificates are renewable across updates
  - Works across authenticated proxies

ClickOnce And WPF Applications

- Create a WPF Application Application (select File | New | Project, in Project Types, open Visual C# and select .NET Framework 3.0, select Windows Application (WPF))
  - Set Name to WPF1 and click OK
- Open Window1.xaml
  - Add a button to the page
- Publish the application (Build | Publish WPF1)
  - Set the location to "http://localhost/WPF1"
- In Internet Explorer click on the Install button and the application will run

Creating An XBAP

- Create a XAML Browser Application (select File | New | Project, in Project Types, open Visual C# and select .NET Framework 3.0, select XAML Browser Application (WPF))
  - Set Name to XBAP1 and click OK
- Open Page1.xaml
  - Add a TextBlock to the page

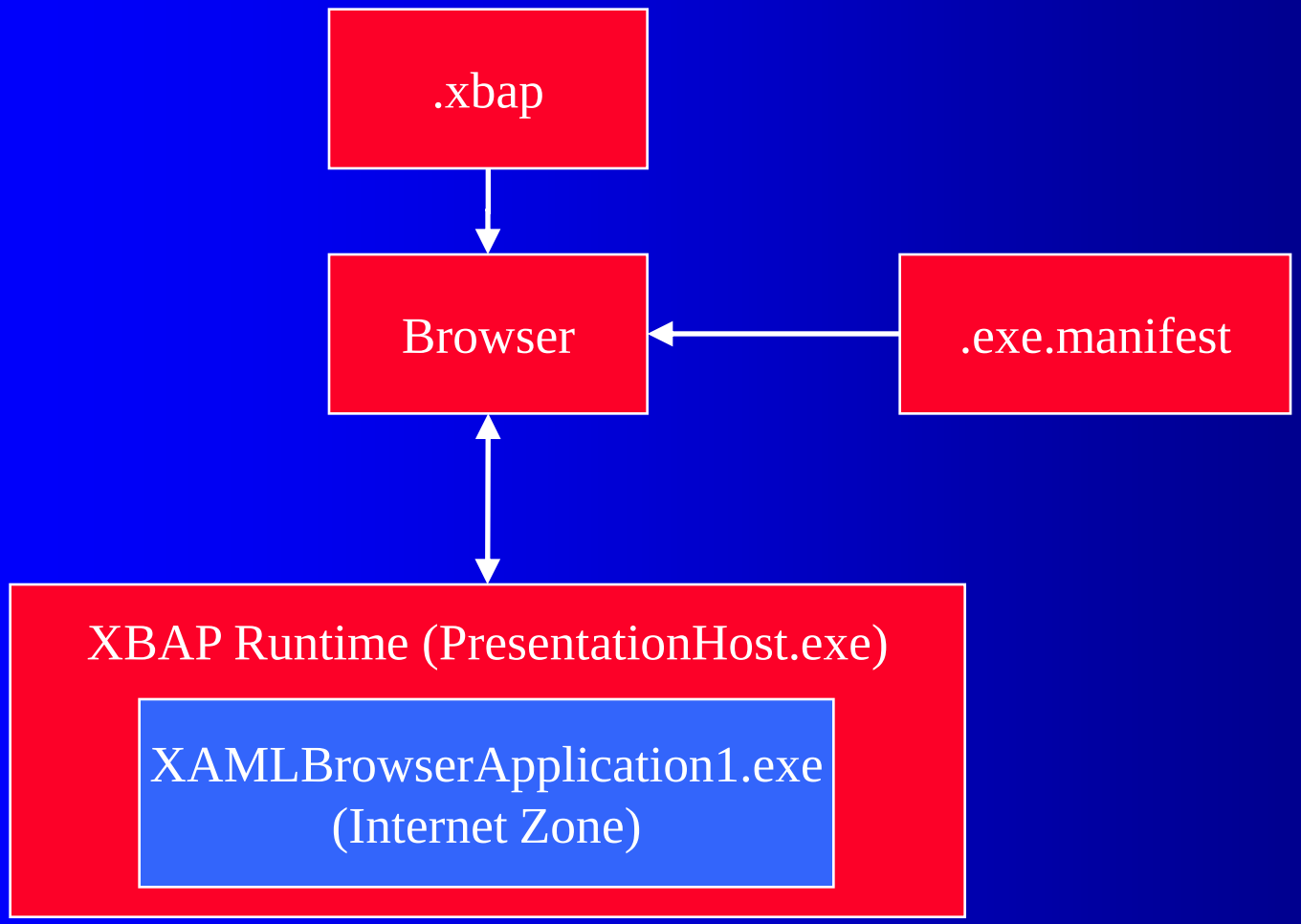
```
<TextBlock Margin="1">
        Hello World
</TextBlock>
```

- Run the application and see that it runs in the browser

XBAP Client Requirements

- Windows XP SP2 or later
- Internet Explorer 6 or later
  - For Internet Explorer 6 the .NET Framework 3.0 must be installed prior to opening the XBAP
  - Internet Explorer 7 recognises XBAPs and can offer to install the .NET Framework 3.0 prior to opening the XBAP
  - FireFox does not support XBAPs
    - Use IETab (http://ietab.mozdev.org) to launch IE automatically from FireFox

How Does It Work ?



XBAPs And The Internet Security Zone

- XBAPs:-
  - Cannot use the ClickOnce API
  - Cannot use unmanaged code
    - Including bitmap effects
  - Windows drag and drop
    - Simulate drag and drop (respond to mouse events directly)
  - Cannot read/write local files
  - Cannot use Windows Forms controls
  - Cannot use WCF services
  - Cannot use dialog windows (e.g. OpenFileDialog)
    - Use Popup element instead

Page Navigation (Internet Explorer 7 And Above)

- Add a second page to XBAP1 (in Solution Explorer right click the project, select Add | New Item…, Page (WPF) and click Add)
- Add the following hyperlink to the TextBlock on Page1.xaml:-

```
<Hyperlink NavigateUri="Page2.xaml">Page 2</Hyperlink>
```

- Run the application and click on the hyperlink
  - Click on the back button to go back to the previous page
  - Click on the forwards button to move to page 2

Converting A WPF Application To An XBAP

- Open the WPF application's .csproj or .vbproj file using a text editor and change the following elements:-

```
<HostInBrowser>True</HostInBrowser>
<Install>False</Install>
<TargetZone>Internet</TargetZone>
<ApplicationExtension>.xbap</ApplicationExtension>
```

- To debug an XBAP you also need to change these:-
  - EnableSecurityDebugging, MapFileExtensions, StartAction, StartProgram, StartArguments

Single Source WPF / XBAP Applications

- Karen Corby's Flexible Application Template
  - http://scorbs.com/2006/06/04/vs-template-flexible-application
  - Maintain a single .csproj or .vbproj file for an application that can be WPF or XBAP
  - Switch between WPF and XBAP using Configuration combo box
  - Does not support publishing yet

XBAP Examples

- To search for examples of XBAPs:-
  - http://www.google.com/search?q=filetype:xbap
- XamlXaml Examples
  - http://xamlxaml.com/category/examples/

Windows Forms, WPF And XBAPs Comparison

	Windows Forms	WPF	XBAPs
Deployment Manifest File Extension	.application	.application	.xbap
Online	Yes	Yes	Yes
Offline	Yes	Yes	No
Host	None or AppLaunch.exe	None or AppLaunch.exe	PresentationHost.exe
Top Level UI	Windows Forms	WPF	Browser
Minimum .NET Framework	2.0	3.0	3.0
Can Install Pre-Requisites	Yes	Yes	No
Can Elevate Permissions On Demand	Yes	Yes	No

XBAPs And Silverlight Comparison

	XBAPs	Silverlight
UI Runs In	Browser	Browser
Process Runs In	PresentationHost.exe	Browser
Installed In	ClickOnce Online Cache	Browser Cache
.NET Framework On Client	3.0	Not required
Operating System	Windows XP SP2 or later	Windows XP SP2 or later, Mac OS X (Linux, Unix in 1.1)
Browser	IE6 or later (FireFox 1.5 & 2 in .NET 3.5)	IE6 or later, FireFox 1.5 & 2, Safari 2
WPF Features	Restricted only by Internet Zone	Vector graphics, animations, streaming (Some controls in 1.1)
Client Side Code	Managed	JavaScript (Managed in 1.1)

Summary

- ClickOnce and Windows Forms
- ClickOnce and WPF
- XAML Browser Applications (XBAPs)
- Windows Forms, WPF and XBAP Comparison
- XBAP and Silverlight Comparison