www.migrateto.net

WHY USE WPF INSTEAD OF WINFORMS

Whitepaper on WPF Features That Take Desktop Development to the Next Level

Nellaiappan L

OVERVIEW

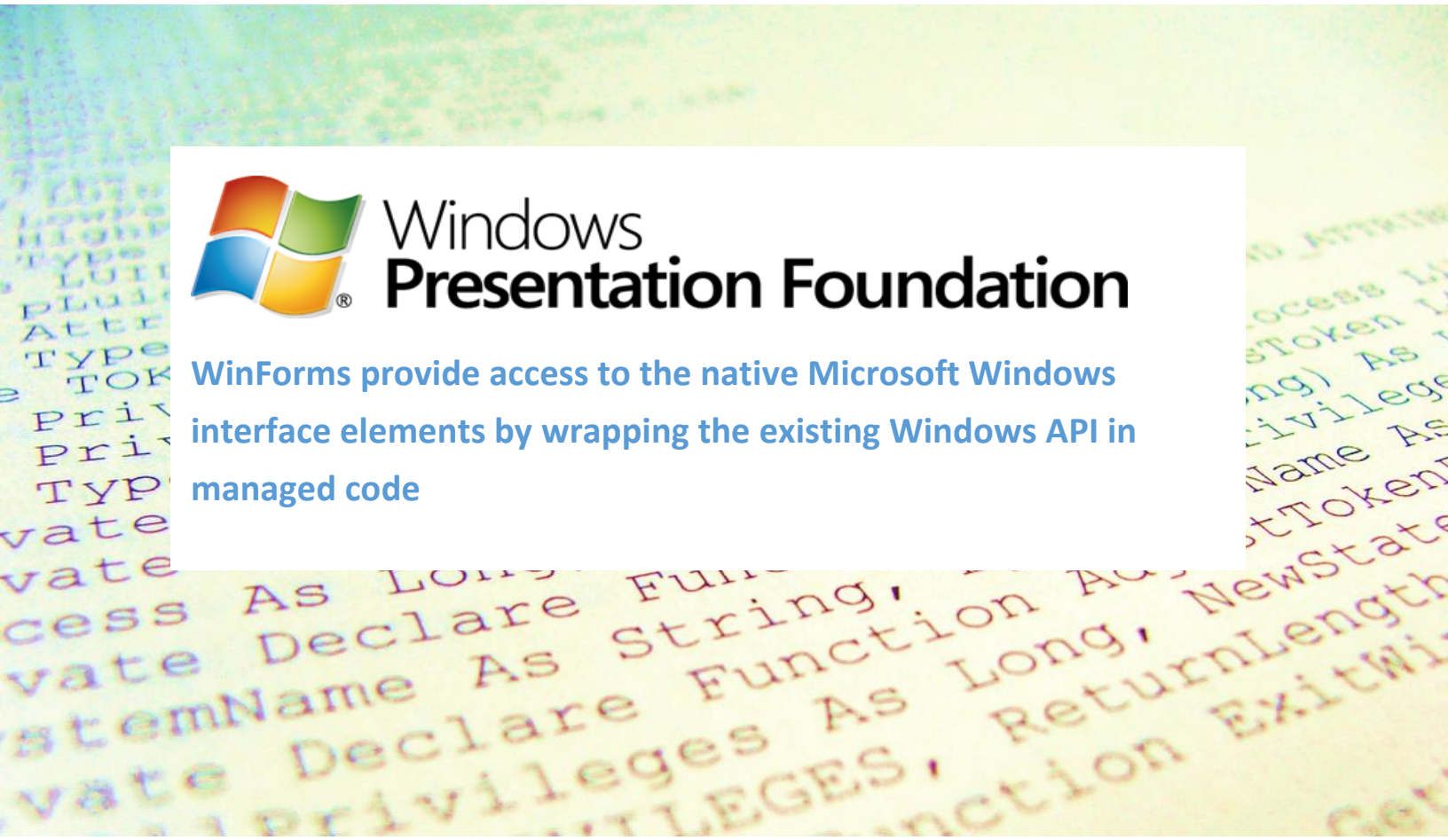
In today's technology-driven world, organizations need to strategically decide on how to develop new desktop based applications. When you are opting for Microsoft technologies, you have two choices available: Windows Forms (WinForms) and Windows Presentation Foundation (WPF). **THIS WHITEPAPER DISCUSSES KEY WPF FEATURES THAT MAKE IT A BETTER CHOICE THAN WINDOWS FORMS FOR DESKTOP APPLICATION DEVELOPMENT.**

BACKGROUND

WinForms have been used for a long time to building Desktop applications. WinForms serve as platform to write rich client applications with a graphical class library included as a part of .NET framework. WinForms provide access to the native Microsoft Windows interface elements by wrapping the existing Windows API in managed code. Modern software development goes through rapid

iteration and major user interface changes throughout the project lifetime. Changes may be driven by project stakeholders with evolving requirements, graphic designers with updated UIs, developers or marketing team who want the product to look "shiny" and animated.

TO SUPPORT THE RAPIDLY CHANGING SCENARIOS, YOU MUST CHOOSE A TECHNOLOGY THAT DECOUPLES VISUAL BEHAVIOR FROM THE UNDERLYING PROGRAM LOGIC AND SEPARATES THE USER INTERFACE FROM THE REST OF THE IMPLEMENTATION. Developers should be able to create a fully functional "ugly" application that designers can directly re-theme without requiring developers to translate their artwork. The Win32 style of programming, in which controls directly contain code to paint and repaint themselves, makes rapid user interface iteration far too difficult for most projects.

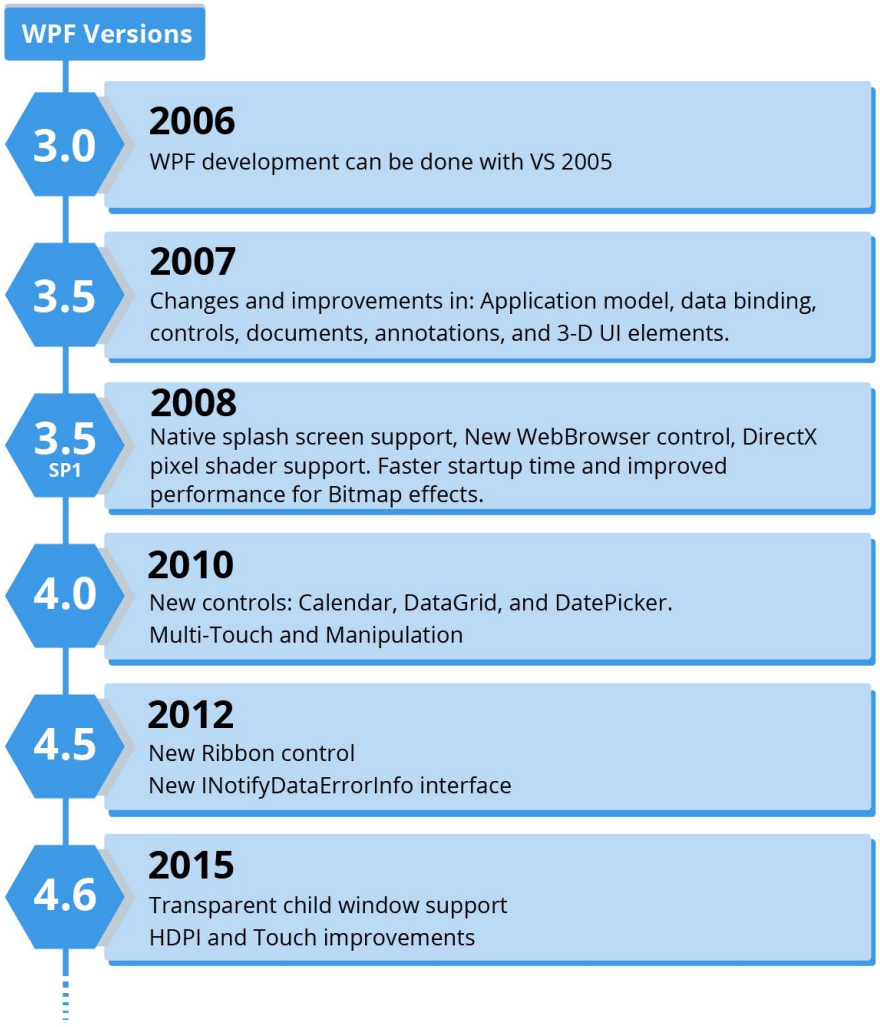


Windows Presentation Foundation (WPF)

In 2006, Microsoft released Windows Presentation Foundation to help people create cutting-edge adaptable software. With the release of WPF 4 in 2010, the technology has grown better than ever at delivering amazing results for just about any kind of software.

Almost a decade after Tom Cruise helped popularize the idea of multi-touch computer input in the movie *Minority Report*, and after successful multi-touch implementations in a variety of devices, WPF 4 and Windows 7 are bringing multi-touch to the masses.

WPF VERSIONS AND MAJOR ENHANCEMENTS



The primary technology behind many Windows-based user interface used Graphics Device Interface (GDI). GDI+ was introduced during the Windows XP time frame. It ended up being slower than GDI due to its complexity and lack of hardware acceleration. With the introduction of .NET and managed code in 2002, developers were treated to a highly productive model for creating Windows (and web) applications. In this world, Windows Forms (built on top of GDI+) became the primary way for a C# programmer to create new user interfaces on Windows.

WINDOWS FORMS HAS BEEN A SUCCESSFUL AND PRODUCTIVE TECHNOLOGY, BUT IT STILL HAS ALL THE FUNDAMENTAL LIMITATIONS OF GDI+.

Microsoft recognized that something brand new was needed that escaped the limitations of GDI+ yet provided the kind of productivity that people enjoy with frameworks like Windows Forms. And with the continual rise of cross-platform applications based on HTML and JavaScript, Windows desperately needed a technology that's as fun and easy to use as these, yet with the power to exploit the capabilities of the local computer. WPF is the answer for software developers and graphic designers who want to create modern user experiences without having to master several difficult technologies.

WPF IS THE ANSWER FOR SOFTWARE DEVELOPERS AND GRAPHIC DESIGNERS WHO WANT TO CREATE MODERN USER EXPERIENCES WITHOUT HAVING TO MASTER SEVERAL DIFFICULT TECHNOLOGIES

WPF Features That Take Desktop Development to the Next Level

1. BROAD INTEGRATION

Prior to WPF, a Windows developer who wanted to use 3D, video, speech, and rich document viewing in addition to normal 2D graphics and controls would have to learn several independent technologies. These independent technologies have number of inconsistencies and attempt to blend them together without much built-in support. But WPF covers all these areas with a consistent programming model as well as tight integration when each type of media gets composited and rendered.

2. RESOLUTION INDEPENDENCE

Imagine a world in which moving to a higher resolution or DPI setting doesn't mean that everything gets smaller; instead, graphics and text simply get crisper! Envision user interfaces that look reasonable on a small netbook as well as on a 60-inch TV! WPF makes this easy and gives you the power to shrink or enlarge elements on the screen independently from the screen's resolution. A lot of this is possible because of WPF's emphasis on vector graphics.

3. HARDWARE ACCELERATION

WPF is built on Direct3D, so content in a WPF application - whether 2D or 3D, graphics, or text is converted to 3D triangles, textures and other Direct3D objects and then rendered by hardware. This means that WPF applications get the benefits of hardware acceleration for smoother graphics and all around better performance.

4. DECLARATIVE PROGRAMMING

Declarative programming is not unique to WPF, as Win16/Win32 programs have used declarative resource scripts to define the layout of dialog boxes and menus for over 25 years. But WPF takes declarative programming to the next level with Extensible Application Markup Language (XAML) The combination of WPF and

XAML is similar to using HTML to define a user interface—but with an incredible range of expressiveness.

5. SEPARATE UI AND LOGIC

In WPF, the UI can be designed separately and the code can be structured separately. But in WinForms the code for each event and method are tightly coupled with the UI element. In WinForms, even when you used the visual designer, there was layout code generated in the language of choice (for example - C#, VB.NET, or C++) for the .Designer extension part of your form. You could not design the form separately without actually having that code – this means that you had to know the language in order to build the UI. In WPF, the UI is based on XAML – extensible and using XAML, you can design a specific UI without knowing what language it is going to be used with – either C# or VB.NET. Therefore, the entire UI design can be done (not necessarily) by a person who is not aware of the programming language that is going to be used (even a person who has no programming language knowledge whatsoever).

6. WINFORMS TOOLBOX IN WPF

When WPF just started, a lot of developers were saying that the main reason they don't like WPF is because of the lack of user controls that can be used in WPF applications. Although this was only partially true – most of the controls in the WinForms toolbox are accessible through WinAPI. So you have the flexibility to use WinForms controls as well as WPF controls.

7. ABILITY TO RUN IN BROWSER

A WPF can run inside the browser through XBAP and access a lot of Windows functionality as long as it's a full-trust application. This gives you the ability to host the application on a remote server and let your clients access it no matter where they are, as long as they have the proper version of .NET Framework installed. Now your application can not only run based on separate windows, but

also pages. This is especially useful when there are various levels segments that require user input and visualization, since pages offer easier navigation between content.

8. **MULTILINGUAL SUPPORT**

WPF application supports multilingual development. You can create your application with dynamic language change based on Resource files. The resource file can be for individual languages and can have the flexibility to change the language at runtime to view the application on their culture specific. This allows for global release of your software and allows users to interact in their language of preference.

CONCLUSION

The demand for robust user experience continues to grow with highly sophisticated consumer products. The advent of WPF as a major Microsoft technology for programming visually appealing Windows client applications is to satisfy the demands of developers for better and faster methods to designing the user experience.

The rapid growth of software applications should push industries to rethink and redesign the face of their software.