

POWERBOOTS

NEXT GENERATION UI



Joel Bennett
HuddledMasses.org

OVERVIEW

- Introduction to WPF?
- Why do you need UI?
- Building UI in PowerBoots
- Advanced User Interfaces

Intro to WPF

- Windows Presentation Framework
- Next-generation UI framework
- Resolution-Independent
- Vector-based rendering
- Hardware accelerated

Windows Presentation Foundation (WPF) is, according to Microsoft: “a next-generation presentation system for building Windows client applications with visually stunning user experiences.”

And that’s basically true.

What do we mean by “next generation”? Well, Windows Forms represents the current generation. It’s a .Net wrapper over the Win32 APIs, and almost all of the Win32 API shows up in WinForms: window handles, message loops, bitmap pixel-based drawing, and a lot of resource usage.

The core of WPF is a resolution-independent, **vector-based** rendering engine that is built to take advantage of modern graphics hardware. It doesn’t use window handles (resources) for controls, and it doesn’t rely on message passing or a message loop (although each top-level window still has to have one, for interoperability with Windows).

Instead it is event driven, and events bubble up (or down) through containers (which we’ll talk more about in a bit) and can be handled at any level.

Intro to WPF

- XAML Markup (Declarative UI)
- Styles, Templates
- Animation
- 2D & 3D graphics
- Documents with rich typography

Among the best features of WPF for developers is the fact that it uses a new declarative XML markup language called XAML (pronounced ZAMEL, rhymes with camel). This markup language leads to excellent tooling (show Expression Blend) that allows developers and designers to collaborate on user interfaces.

In addition, XAML supports templated controls (which means that you can totally change the look of a control without changing its behavior, as far as the developer is concerned), and “styles” which allow you to consistently apply custom looks to your user interfaces (with the help of a graphics designer).

The stuff that’s most exciting in terms of “selling” users on these new user interfaces is the ability to create animations and transitions easily, and to do 2D and 3D graphics (and even mix them). Documents with rich typography (OpenType fonts, with alternate glyphs, etc) are also compelling, but probably a bit out-of-scope for our talk tonight.

Why do you need GUI?

- Get Input from users
- Show Data to users
- Summarize/Overview
- Monitoring

If you're writing scripts for other people to use, a simple user interface can work wonders on their willingness to use them, and on the perception of how easy they are to use. One thing we're working on for the next version of PowerBoots is a cmdlet invoker which would prompt you graphically for any and all inputs that a cmdlet (or script) needs, provide visual feedback of validation errors, etc.

Whether you write scripts for "end users," for other system administrators or developers, or for yourself, when you need to analyze data, it's frequently best to do so in a graphical way, and PowerBoots can help with that. This is particularly true when you're trying to get an overview of large amounts of data, or monitor and track data that changes throughout the day...

So, hopefully I've convinced you that there are some areas where you need graphical user interfaces ... let's move on, and talk about what PowerBoots is, and why it should be your choice. Actually, first, let me make an announcement.

SHOW-UI

PowerShell WPF Toolkit

PowerBoots + WPK

- **Faster and Lighter**
- **More helper functions**
- **Pre-baked UI kits**

Hot News!

This is the “official” announcement that PowerBoots and WPK are merging -- I know some of you have heard about this, and there are still some details to work out, but here’s the basics:

Once WPK launched, I had started looking at all of the code from WPK and had begun copying over the features and ideas which were not already in PowerBoots ... but then James Brundage, the original WPK author, left Microsoft (he’s at Wolfram). He’s still doing PowerShell consulting through a company he’s starting called “Start-Automating”, and he’s obviously putting most of his free time into that venture.

In any case, we’ve agree to work together to improve PowerBoots, and that a new name would be a good way to convey that it’s now the only WPF module. We figured if we called it “the WPF module” you wouldn’t be able to find it on Google, so we’re going to stick with “Show-UI” and we’ve registered show-ui.org as it’s home page (for now, that will just point to the CodePlex site).

So, with that out of the way, let’s move on to showing you how to use it...

Building UI

- Start with Show-UI ("boots")
- Handle Events
- Container Model
- Learn about panels:
Stack, Wrap, Dock, Grid, Canvas

"Show" or Show-UI is the new name for New-BootsWindow, the main cmdlet to create an interface. Everything starts with Show-UI

Ok, let's do some demos. We'll walk through this series:

I'll show you how to call Show-UI, and how DataTemplates work.

Then we'll talk about event handling, and how you can run script where users interact with your UI

Then we'll get into the WPF container model (which is what makes WPF perfect for the PowerBoots syntax)

And I'll show you some of the most common containers

Advanced UI

- Charts
- Monitoring
- Gadgets and Animation

Next, I have some more advanced examples of what you'll be able to do in Show-UI with WPF's databinding and pre-built functions like: Show-Chart, Show-Gadget, and New-ScriptDataSource...

A Note: PoshConsole

- Rich WPF PowerShell host
- Flat “metro” look
- Hotkeys, Quake Mode
- New release this weekend

If we run out of things to talk about...

Most of PowerBoots works inline in PoshConsole...

Show examples:

Show { \$files }

Show { \$procs }

Samples 13

VisifireSamples 1

VisifireSamples 3