Introduction to Windows Client Technologies

...for Java developers

Introduction

A client application is an application that runs locally on a user's computer. Generally speaking, client applications fall into three categories:

- Desktop applications that are explicitly installed on the user's computer and perform common business tasks.
- Lightweight applications that are downloaded from a Web site and run in a Web browser.
- Mobile applications that run on a device such as a mobile phone.

Browser-based and phone-based applications tend to be more focused toward a specific function than the more generalized approach that desktop applications take.

On the Windows® platform, the primary technology for creating desktop-based user interfaces is Windows Presentation Foundation (WPF). This platform is comparable to the Java Swing graphical user interface (GUI) toolkit. However, WPF is more modern, so it has many features that Swing does not.

The principal client technology for creating browser-based applications is Microsoft® Silverlight®. Silverlight supports multiple browsers and can run on non-Windows computers when the appropriate Silverlight browser plug-in has been installed. This plug-in provides the Silverlight runtime environment. Silverlight 3 provides an out of browser feature to enable users to run Silverlight applications from the desktop.

Silverlight has functionality that is comparable to JavaFX (and Adobe Flash). Both can deliver rich Internet-based experiences over the Web, but Silverlight utilizes the Microsoft .NET Framework whereas JavaFX employs the Java platform.

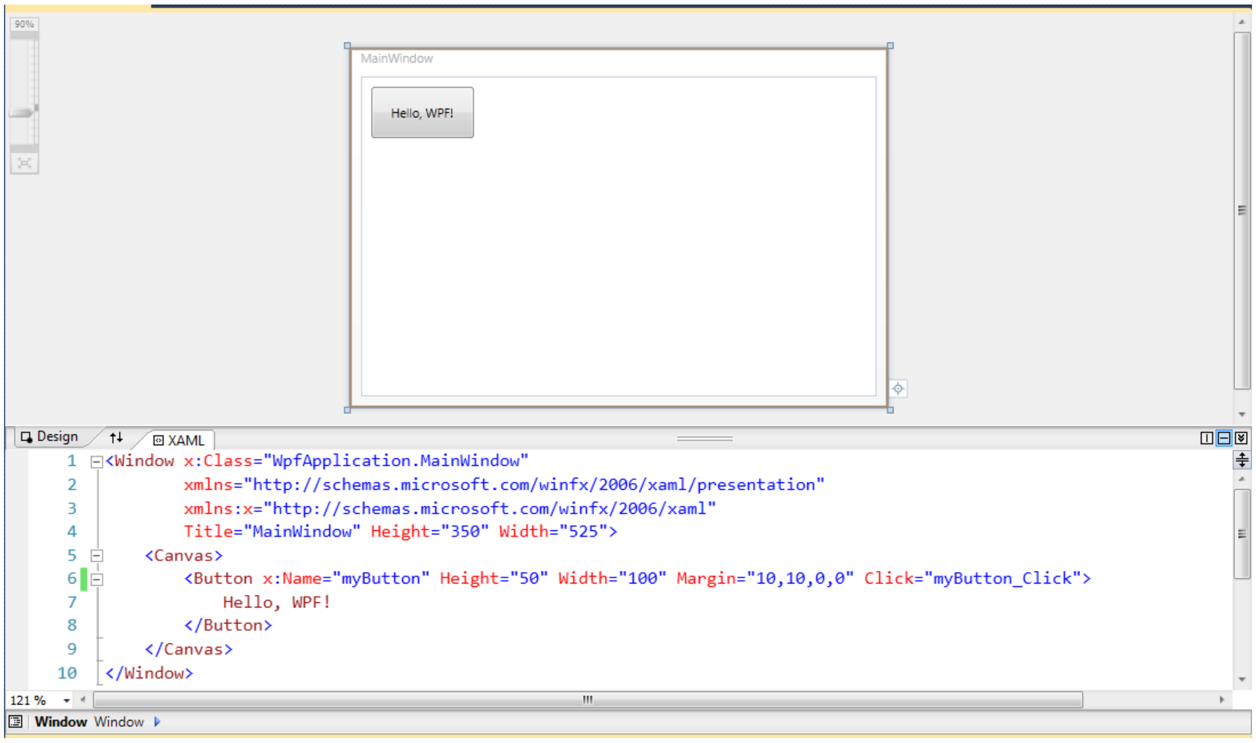
Contents

Windows Presentation Foundation

WPF is the graphical subsystem that is responsible for rendering user interfaces in Windows desktop applications. WPF was introduced in .NET Framework 3.0 with Windows Vista® and offers a modern, vector-based presentation framework.

The vector-based nature of WPF means that you can scale the user interface without a loss in quality. You can do this by using the graphical power of Microsoft DirectX® technology on the graphics processor that is provided on the graphics card. This enables a WPF application to produce interfaces that appear more modern and stunning than those in any Java user interface framework including Swing. However, note that if a WPF application is running on a computer that has an older graphics card with no DirectX support (or on an older version of Windows with.NET Framework 3.0 installed), the application still runs, but is forced into a software rendering mode that may reduce performance.

WPF provides several prebuilt controls and containers, but you can also define your own custom controls. WPF implements a programming model that separates the layout of the user interface from the user interface and business logic of an application. It uses XAML to define the user interface. XAML is a graphics markup language that is based on XML—Eclipse XWT takes a similar approach for designing Swing user interfaces. Figure 1 illustrates the Design and Code View windows of Microsoft Visual Studio® 2010, which show the layout of a simple WPF application. The application defines a **Canvas** control as its layout control, which contains a **Button** control with its various properties.

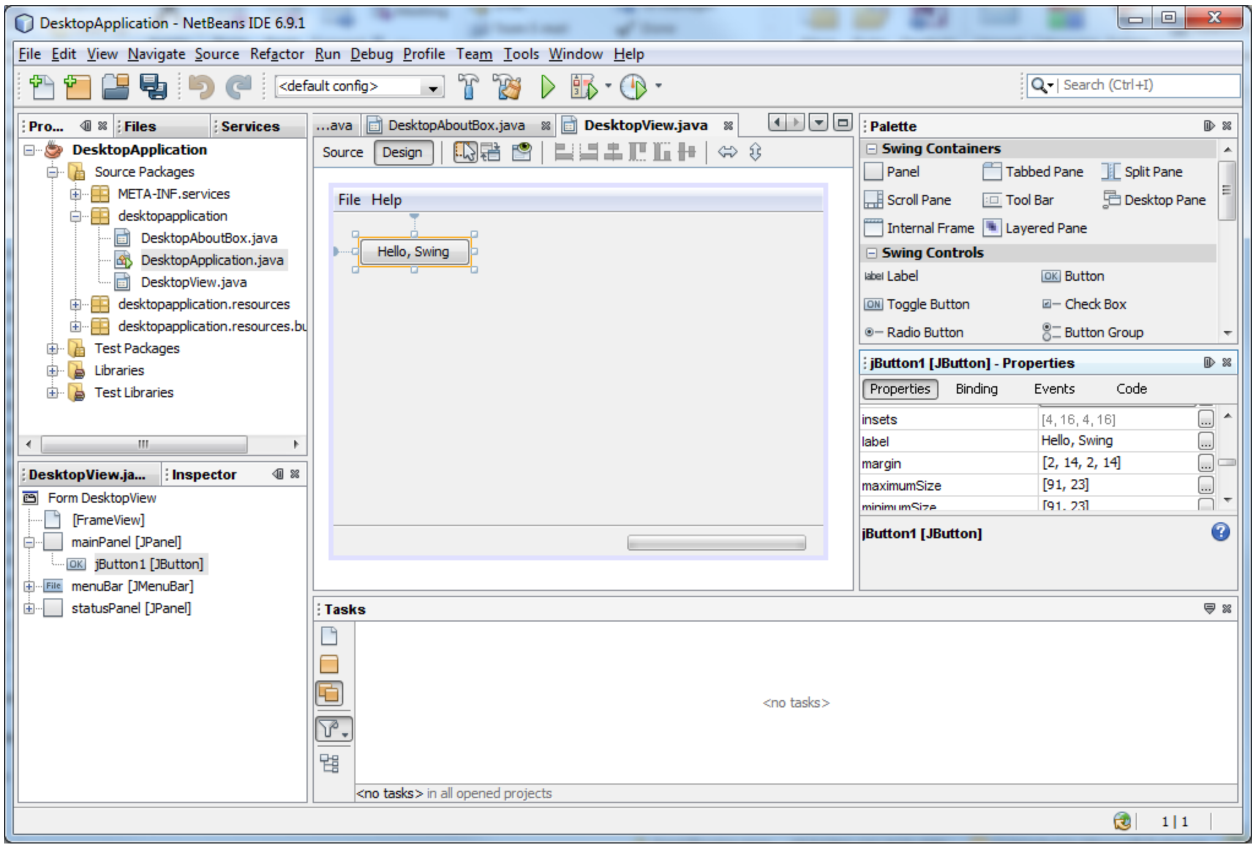


```
<Window x:Class="WpfApplication.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Canvas>
        <Button x:Name="myButton" Height="50" Width="100" Margin="10,10,0,0" Click="myButton_Click">
            Hello, WPF!
        </Button>
    </Canvas>
</Window>
```

Figure 1. Design and Code View windows in Visual Studio 2010

You define the application logic by using a programming language such as C#, and implement event handlers to connect the user interface to the application code. In Figure 1, the **Click** event is handled by the **myButton_Click** event-handler method. The code for this handler is defined in the code-behind file for this **Window** object (not shown).

Figure 2 shows the equivalent design window for a Swing application that is being developed in NetBeans. The application shows a **JButton** control on a **JPanel** object. Note that Swing does not provide the equivalent markup language to XAML. In addition, the layout is defined by setting the properties of controls in the Properties window. Unlike the WPF model, where the XAML markup entirely defines the user interface, setting the properties of a control in Swing generates Java code that configures the control when the application runs.


DesktopApplication - NetBeans IDE 6.9.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Search (Ctrl+I)
Pro... Files Services
DesktopApplication
Source Packages
META-INF.services
desktopapplication
DesktopAboutBox.java
DesktopApplication.java
DesktopView.java
desktopapplication.resources
desktopapplication.resources.bu
Test Packages
Libraries
Test Libraries
...ava DesktopAboutBox.java DesktopView.java
Source Design
File Help
Hello, Swing
Palette
Swing Containers
Panel Tabbed Pane Split Pane
Scroll Pane Tool Bar Desktop Pane
Internal Frame Layered Pane
Swing Controls
Label Button
Toggle Button Check Box
Radio Button Button Group
jButton1 [JButton] - Properties
Properties Binding Events Code
DesktopView.ja... Inspector
Form DesktopView
[FrameView]
mainPanel [JPanel]
jButton1 [JButton]
menuBar [JMenuBar]
statusPanel [JPanel]
Tasks
<no tasks>
<no tasks> in all opened projects
1 | 1

insets	[4, 16, 4, 16]
label	Hello, Swing
margin	[2, 14, 2, 14]
maximumSize	[91, 23]
minimumSize	[91, 23]

jButton1 [JButton]

Figure 2. Designing a Swing application in NetBeans

The event-handling model that WPF employs is similar in concept to that of Swing, although WPF provides a more extensive event-routing mechanism. WPF implements event tunneling or preview events, which enable a control such as a **Canvas** control or **Window** control to intercept events that are directed toward child controls that they contain. WPF also implements event bubbling, enabling events to be propagated from a child control to the container that holds the child control.

WPF also provides data binding, which is a powerful yet simple way for an application to present data. You can associate various properties of user interface elements to data items in a data model. If the values in these data items change, the corresponding elements in the user interface are updated automatically. Data binding supports a variety of data sources including data that is retrieved from a database, common language runtime (CLR) objects, and elements in an XML document.

The controls that make up a WPF application are also “lookless,” in the sense that you can define templates and styles to suit the needs of an application. If you modify a template, the look and feel of the user interface changes accordingly. This approach is similar in concept to that implemented by the pluggable look and feel that is available in Swing, and extended by the Synth package. However, in WPF, the properties of the WPF controls are changed directly instead of defining and applying **SynthStyle** style properties.

In WPF version 4, which is the current version, WPF offers features for Windows 7 shell integration, support for multitouch, and the new Visual State Manager, which offers a simple way to apply new visual states to an application.

WindowsClient.net is an excellent resource for diving into WPF development. The Web site contains training videos, presentations, and hands-on labs to help you learn about specific areas of the platform.

Silverlight

Silverlight is a development platform for creating rich media applications primarily for the Web and phone environments. However, you can also deploy and run Silverlight applications on the desktop. Silverlight integrates multimedia, graphics, and animations into a single .NET-driven platform that is compatible across browsers and platforms. If a user browses to a Web site that contains Silverlight content, the user may be asked to install the Silverlight browser plug-in before the content is rendered.

Silverlight offers similar functionality to JavaFX. Both platforms provide data binding although they implement this feature in different ways.

The Silverlight development experience is very similar to WPF. Silverlight implements the same programming model that separates the user interface from the business logic by using XAML and code-behind classes. Silverlight implements the WPF data-binding model.

Did you know?

The Prism resource on the CodePlex Web site provides guidance documentation, examples, a reusable code library, and a reference implementation that shows how to design and build rich, flexible, and easy-to-maintain WPF desktop applications, Silverlight Web client applications, and Windows Phone 7 mobile applications.

Design Patterns and Implementation Best Practices

The architecture of the WPF and Silverlight development models means that these platforms lend themselves really well to common development patterns including the Model-View-ViewModel (MVVM) pattern in particular. The MVVM pattern is a natural extension of the Model-View-Controller (MVC) pattern that enables you to build extensible, command-driven user interfaces that can totally separate the user interface layout from the rest of the application. It is facilitated by using the data binding with the Command and Messaging frameworks that are present in both WPF and Silverlight.

This separation of concerns between the user interface and application logic also means that the MVVM pattern is very suitable for a designer and developer workflow. User interface designers, who might not be familiar with writing code, can use Microsoft Expression Blend® to design a professional user interface in a visual way. Developers can then import the XAML code that Expression Blend generates into Visual Studio and combine it with the application logic.

Summary

WPF and Silverlight are the two primary technologies for targeting the desktop, Web, and phone environments. WPF and Silverlight share a common development model that has many similarities with user interface technologies that are commonly used in conjunction with Java applications, particularly Swing and JavaFX. However, they also have some significant differences both in the way that you design applications, and the strategies that are available for implementing them.

With your previous experience in Java-based technologies, you should be in a great position to hit the ground running with WPF-based and Silverlight-based development. You can benefit from key platform features such as the use of XAML with the separation of user interface and business logic, data binding, theming, templating, and a host of multimedia options.