Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

WPF with MVVM and C#

Class Duration

35 hours of live training delivered over 5 days.

Student Prerequisites

- Professional C# programming experience
- Familiarity with Git
- No prior WPF, XAML, or MVVM experience required

Target Audience

C# developers who need to build maintainable Windows desktop applications and want to learn WPF the right way from the start: with the MVVM pattern as the architectural backbone rather than an afterthought. Ideal for teams starting new WPF projects, developers moving from web or WinForms backgrounds, teams porting existing Windows Forms, MFC, or traditional Win32 applications to WPF, and organizations that want testable desktop codebases. Developers who already know WPF and MVVM basics and want the full application-lifecycle picture (custom controls, performance, packaging) should consider Building Powerful Desktop Apps with WPF. Teams who need their XAML skills to reach mobile and other desktop platforms can continue to Build Cross-Platform Apps with .NET MAUI.

Description

This hands-on course teaches WPF on its current foundation, .NET 10 with Microsoft's evolving Fluent theming, with MVVM and clean C# architecture woven in from the first hour, not bolted on at the end. WPF remains a fully supported, actively developed part of modern .NET, and the premier framework for serious Windows desktop work. Participants build up from XAML layout and WPF's uniquely powerful data binding through the full MVVM stack using CommunityToolkit.Mvvm and its source generators: observable view models, relay commands, validation, messaging, and view-model-first navigation with dependency injection. Data-heavy UI, async patterns, and styling and theming complete the picture, and the course closes with unit testing view models: the payoff MVVM promises. All work happens in realistic business applications, with agentic AI coding assistants

used throughout for generating XAML, scaffolding view models, and reviewing binding errors, with participants learning to direct and verify that work.

Learning Outcomes

- Explain WPF’s architecture, its place in the current .NET desktop landscape, and when it is the right choice.
- Build responsive layouts with WPF’s panel system and compose UIs from reusable user controls.
- Apply data binding fluently: modes, paths, converters, collection views, and binding diagnostics.
- Structure applications with MVVM: view models, commands, and strict separation from view code.
- Use CommunityToolkit.Mvvm source generators to eliminate MVVM boilerplate.
- Implement validation, messaging between view models, and view-model-first navigation with dependency injection.
- Bind and edit large data sets with collection virtualization and master-detail patterns.
- Style and theme applications with resources, styles, templates, and the Fluent theme.
- Write unit tests for view models and application logic, keeping views thin by design.
- Use agentic AI coding assistants effectively for XAML generation, view model scaffolding, and refactoring toward MVVM.

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

A Windows machine with Visual Studio 2026 (Community edition is sufficient), the .NET 10 SDK, and Git. Access to an agentic AI coding assistant is required; a free tier is sufficient.

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Training Topics

WPF and the Modern .NET Desktop

- WPF’s place in the .NET 10 desktop landscape
- Application model and startup
- Fluent theming and modern visuals
- Project structure and tooling in Visual Studio 2026
- XAML Hot Reload and designer workflow
- Agentic AI coding assistants in XAML and C# work

XAML Foundations

- XAML syntax, namespaces, and markup extensions
- Layout panels: Grid, StackPanel, DockPanel, WrapPanel
- Fine-tuning layout: margin, padding, alignment, and transforms
- Core controls and content model
- Resources and resource dictionaries
- Styles and setters

Data Binding in Depth

- Binding modes, paths, and sources
- Dependency properties and the WPF property system
- INotifyPropertyChanged and change notification
- Value converters and string formatting
- ElementName, RelativeSource, and DataContext flow
- Debugging bindings and binding diagnostics

MVVM Fundamentals

- Why MVVM: testability, separation, longevity
- View, view model, and model responsibilities
- CommunityToolkit.Mvvm and source generators
- Observable properties without boilerplate
- Design-time data and tooling support

Commands and User Interaction

- Routed events and the WPF input architecture
- ICommand and the command pattern
- RelayCommand and AsyncRelayCommand
- CanExecute logic and UI state

- Input gestures and keyboard support
- Async operations: progress, cancellation, error surfacing

Application Architecture and Navigation

- Dependency injection with the .NET Generic Host
- Service abstractions for dialogs and platform features
- View-model-first navigation strategies
- Messaging between view models
- Composition of views: user controls and data templates

Working with Collections and Data

- ObservableCollection and collection binding
- CollectionView: sorting, filtering, grouping
- Master-detail interfaces
- Virtualization for large data sets
- Async data access with EF Core and services

Validation and Forms

- Validation approaches in WPF
- ObservableValidator and data annotations
- INotifyDataErrorInfo and error templates
- Form workflows: dirty tracking, save guards

Styling, Templating, and Theming

- Control templates and lookless controls
- Data templates and template selectors
- Triggers and visual states
- Fluent theme, dark mode, and accent colors
- Application-wide theming strategies

Testing MVVM Applications

- Unit testing view models
- Faking services behind interfaces
- Testing commands, validation, and navigation logic
- What stays untested in the view, by design
- Reviewing AI-generated view models and tests

MVVM in Practice

- Anti-patterns: code-behind creep, fat view models

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Sharing state across view models
- Performance basics: binding cost, startup time
- Preparing for growth: modular application structure

