.NET Desktop Guide for Windows Forms

Learn about Windows Forms (WinForms), a graphical user interface for Windows and .NET Framework.

WinForms for .NET Framework

OVERVIEW

Windows Forms overview

GET STARTED

Get started with Windows Forms Designer

Create a WinForms app from the command-line

Controls

OVERVIEW

Developing your own controls

Developing controls

HOW-TO GUIDE

Add Controls to Windows Forms

Position controls on Windows Forms

REFERENCE

Windows Forms Controls by Function

Events

OVERVIEW

Events in Windows Forms

Event handlers in Windows Forms

Event handlers in Windows Forms

CONCEPT

Order in which events are raised

Input

OVERVIEW

About keyboard input

About mouse input

CONCEPT

Keyboard events

Mouse events

Mouse pointers

HOW-TO GUIDE

Modify keyboard input

Detect modifier keyboard keys

Distinguish between single/double clicks

Getting Started with Windows Forms

Article • 02/06/2023

With Windows Forms, you can create powerful Windows-based applications. The following topics describe in-depth how to harness the power of Windows Forms to display data, handle user input, and deploy your applications easily and with enhanced security.

In This Section

Windows Forms Overview

Contains an overview of Windows Forms and smart client applications.

Creating a New Windows Form

Contains links to topics that describe basic concepts for creating Windows Forms applications.

Creating Event Handlers in Windows Forms

Contains links to topics that describe how to create Windows Forms event handlers.

Adjusting the Size and Scale of Windows Forms

Contains links to topics that show how to adjust the size and scale of Windows Forms.

Changing the Appearance of Windows Forms

Contains links to topics that show how to change the appearance of Windows Forms applications.

Windows Forms Controls

Contains links to topics that describe and show how to use Windows Forms controls and components.

User Input in Windows Forms

Contains links to topics that describe and show how to handle input from the user in Windows Forms applications.

Dialog Boxes in Windows Forms

Contains links to topics that describe the different dialog boxes for use in Windows Forms.

Windows Forms Data Binding

Contains links to topics that describe the Windows Forms data binding architecture and how to use it in Windows Forms applications.

Windows Forms Security

Contains links to topics that describe how to build Windows Forms applications that have enhanced security.

ClickOnce Deployment for Windows Forms

Contains links to topics that describe how to easily deploy Windows Forms applications.

How to: Access Keyed Collections in Windows Forms

Demonstrates how to access collections with keys rather than indexes.

Related Sections

Enhancing Windows Forms Applications

Contains links to topics that describe more advanced concepts for creating Windows Forms applications.

Windows Forms overview

Article • 02/06/2023

The following overview discusses the advantages of smart client applications, the main features of Windows Forms programming, and how you can use Windows Forms to build smart clients that meet the needs of today's enterprises and end users.

Windows Forms and smart client apps

With Windows Forms you develop smart clients. *Smart clients* are graphically rich applications that are easy to deploy and update, can work when they are connected to or disconnected from the Internet, and can access resources on the local computer in a more secure manner than traditional Windows-based applications.

Build rich, interactive user interfaces

Windows Forms is a smart client technology for the .NET Framework, a set of managed libraries that simplify common application tasks such as reading and writing to the file system. When you use a development environment like Visual Studio, you can create Windows Forms smart-client applications that display information, request input from users, and communicate with remote computers over a network.

In Windows Forms, a *form* is a visual surface on which you display information to the user. You ordinarily build Windows Forms applications by adding controls to forms and developing responses to user actions, such as mouse clicks or key presses. A *control* is a discrete user interface (UI) element that displays data or accepts data input.

When a user does something to your form or one of its controls, the action generates an event. Your application reacts to these events by using code, and processes the events when they occur. For more information, see Creating Event Handlers in Windows Forms.

Windows Forms contains a variety of controls that you can add to forms: controls that display text boxes, buttons, drop-down boxes, radio buttons, and even Web pages. For a list of all the controls you can use on a form, see Controls to Use on Windows Forms. If an existing control does not meet your needs, Windows Forms also supports creating your own custom controls using the UserControl class.

Windows Forms has rich UI controls that emulate features in high-end applications like Microsoft Office. When you use the ToolStrip and MenuStrip control, you can create

toolbars and menus that contain text and images, display submenus, and host other controls such as text boxes and combo boxes.

With the drag-and-drop **Windows Forms Designer** in Visual Studio, you can easily create Windows Forms applications. Just select the controls with your cursor and add them where you want on the form. The designer provides tools such as gridlines and snap lines to take the hassle out of aligning controls. And whether you use Visual Studio or compile at the command line, you can use the FlowLayoutPanel, TableLayoutPanel and SplitContainer controls to create advanced form layouts in less time.

Finally, if you must create your own custom UI elements, the System.Drawing namespace contains a large selection of classes to render lines, circles, and other shapes directly on a form.

> ⓘ **Note**
>
> Windows Forms controls are not designed to be marshaled across application domains. For this reason, Microsoft does not support passing a Windows Forms control across an **AppDomain** boundary, even though the **Control** base type of **MarshalByRefObject** would seem to indicate that this is possible. Windows Forms applications that have multiple application domains are supported as long as no Windows Forms controls are passed across application domain boundaries.

Create forms and controls

For step-by-step information about how to use these features, see the following Help topics.

Description	Help topic
Using controls on forms	How to: Add Controls to Windows Forms
Using the ToolStrip Control	How to: Create a Basic ToolStrip with Standard Items Using the Designer
Creating graphics with System.Drawing	Getting Started with Graphics Programming
Creating custom controls	How to: Inherit from the UserControl Class

Display and manipulate data

Many applications must display data from a database, XML file, XML Web service, or other data source. Windows Forms provides a flexible control that is named the DataGridView control for displaying such tabular data in a traditional row and column format, so that every piece of data occupies its own cell. When you use DataGridView, you can customize the appearance of individual cells, lock arbitrary rows and columns in place, and display complex controls inside cells, among other features.

Connecting to data sources over a network is a simple task with Windows Forms smart clients. The BindingSource component represents a connection to a data source, and exposes methods for binding data to controls, navigating to the previous and next records, editing records, and saving changes back to the original source. The BindingNavigator control provides a simple interface over the BindingSource component for users to navigate between records.

You can create data-bound controls easily by using the Data Sources window. The window displays data sources such as databases, Web services, and objects in your project. You can create data-bound controls by dragging items from this window onto forms in your project. You can also data-bind existing controls to data by dragging objects from the Data Sources window onto existing controls.

Another type of data binding you can manage in Windows Forms is *settings*. Most smart client applications must retain some information about their run-time state, such as the last-known size of forms, and retain user preference data, such as default locations for saved files. The Application Settings feature addresses these requirements by providing an easy way to store both types of settings on the client computer. After you define these settings by using either Visual Studio or a code editor, the settings are persisted as XML and automatically read back into memory at run time.

Display and manipulate data

For step-by-step information about how to use these features, see the following Help topics.

Description	Help topic
Using the BindingSource component	How to: Bind Windows Forms Controls with the BindingSource Component Using the Designer
Working with ADO.NET data sources	How to: Sort and Filter ADO.NET Data with the Windows Forms BindingSource Component
Using the Data Sources window	Bind Windows Forms controls to data in Visual Studio

Description	Help topic
Using application settings	How to: Create Application Settings

Deploy apps to client computers

After you have written your application, you must send the application to your users so that they can install and run it on their own client computers. When you use the ClickOnce technology, you can deploy your applications from within Visual Studio by using just a few clicks, and provide your users with a URL pointing to your application on the Web. ClickOnce manages all the elements and dependencies in your application, and ensures that the application is correctly installed on the client computer.

ClickOnce applications can be configured to run only when the user is connected to the network, or to run both online and offline. When you specify that an application should support offline operation, ClickOnce adds a link to your application in the user's **Start** menu. The user can then open the application without using the URL.

When you update your application, you publish a new deployment manifest and a new copy of your application to your Web server. ClickOnce will detect that there is an update available and upgrade the user's installation; no custom programming is required to update old assemblies.

Deploy ClickOnce apps

For a full introduction to ClickOnce, see ClickOnce Security and Deployment. For step-by-step information about how to use these features, see the following Help topics,

Description	Help topic
Deploying an application by using ClickOnce	How to: Publish a ClickOnce Application using the Publish Wizard Walkthrough: Manually Deploying a ClickOnce Application
Updating a ClickOnce deployment	How to: Manage Updates for a ClickOnce Application
Managing security with ClickOnce	How to: Enable ClickOnce Security Settings

Other controls and features

There are many other features in Windows Forms that make implementing common tasks fast and easy, such as support for creating dialog boxes, printing, adding Help and documentation, and localizing your application to multiple languages. Additionally, Windows Forms relies on the robust security system of the .NET Framework. With this system, you can release more secure applications to your customers.

Implement other controls and features

For step-by-step information about how to use these features, see the following Help topics.

Description	Help topic
Printing the contents of a form	How to: Print Graphics in Windows Forms How to: Print a Multi-Page Text File in Windows Forms
Learn more about Windows Forms security	Security in Windows Forms Overview

See also

- Getting Started with Windows Forms
- Creating a New Windows Form
- ToolStrip Control Overview
- DataGridView Control Overview
- BindingSource Component Overview
- Application Settings Overview
- ClickOnce Security and Deployment

Creating a New Windows Form

Article • 02/06/2023

This topic contains links to topics that describe how to create your first Windows Forms application. Also, the topics in this section introduce some of the basic vocabulary and guidelines that you should understand when you start to create a Windows Forms application. To learn more about Windows Forms applications, the controls you can use on them, events and handling events, and how to handle input from the user, see the related topic list.

In This Section

Windows Forms Coordinates
Describes client and screen coordinates.

How to: Create a Windows Forms Application from the Command Line
Describes how to create a basic Windows Form and compile it from the command line.

Reference

Form
Describes this class and contains links to all its members.

Control
Describes this class and contains links to all its members.

Related Sections

Handling User Input
Contains links to topics that discuss user input and how to handle it in Windows Forms applications.

Creating Event Handlers in Windows Forms
Contains links to topics that describe how to handle events in Windows Forms applications.

Changing the Appearance of Windows Forms
Contains links to topics that show how to change the appearance of Windows Forms applications.