

Programming the Microsoft .NET Framework with C# (Prerelease)

Course No. 2349 • Five days • Instructor-led

Because some parts of the course are currently being developed, some elements of this syllabus are subject to change.

This course syllabus should be used to determine whether the course is appropriate for the student, based on current skills and technical training needs.

Course content, prices, and availability are subject to change without notice.

The goal of this course is to help application developers understand the Microsoft® .NET Framework. In addition to offering an overview of the .NET Framework and an introduction to key concepts and terminology, the course provides a series of labs, which introduce and explain .NET Framework features that are used to code, debug, tune, and deploy applications.

Audience

This course is intended for experienced, professional software developers, including those employed by independent software vendors and software companies or working on corporate internal development teams. Most students will be Microsoft WIN32®, Microsoft Visual Basic®, or Microsoft Visual C++® developers.

At Course Completion

After completing the course, students will be able to:

- List the major elements of the .NET Framework and explain how they fit into the .NET platform.
- Explain the main concepts behind the common language runtime and use the features of the .NET Framework to create a simple application.
- Create and use components in Windows Forms-based and ASP.NET-based applications.
- Use the deployment and versioning features of the .NET runtime to deploy multiple versions of a component.
- Create, use, and extend types by understanding the Common Type System architecture.
- Create classes and interfaces that are functionally efficient and appropriate for given programming scenarios.
- Use the .NET Framework class library to

efficiently create and manage strings, arrays, collections, and enumerators.

- Use delegates and events to make an event-sender object signal the occurrence of an action to an event-receiver object.
- Describe and control how memory and other resources are managed in the .NET Framework.
- Read from and write to data streams and files.
- Use the basic request/response model to send and receive data over the Internet.
- Serialize and deserialize an object graph.
- Create distributed applications by means of Web Services and Object Remoting.

Prerequisites

Before attending this course, students must be proficient in the C++ or Java programming languages and have been exposed to the C# language.

Students can meet these prerequisites by taking Course 2124A, *Introduction to C# Programming for the Microsoft .NET Platform*.

Microsoft Certified Professional Exams

There is no Microsoft Certified Professional exam associated with this course.

Student Materials

The student kit includes a comprehensive workbook and other necessary materials for this class.

The following software is provided in the student kit:

- Microsoft Visual Studio® .NET Beta 2

Module 1: Overview of the Microsoft .NET Framework

The following topics are covered in this module:

- Overview of the Microsoft .NET Framework
- Overview of Namespaces

After completing this module, you will be able to list the major elements of the .NET Framework. This includes:

- Describing the .NET Framework and its components.
- Explaining the relationship between the .NET Framework class library and namespaces.

Module 2: Introduction to a Managed Execution Environment

The following topics are covered in this module:

- Writing a .NET Application
- Compiling and Running a .NET Application

After completing this module, you will be able to explain the main concepts behind the common language runtime and use the features of the common language runtime to create a simple application. This includes:

- Creating simple console applications in C#.
- Explaining how code is compiled and executed in a managed execution environment.
- Explaining the concept of garbage collection.

Module 3: Working with Components

The following topics are covered in this module:

- An Introduction to Key .NET Framework Development Technologies
- Creating a Simple .NET Framework Component
- Creating a Simple Console Client
- Creating an ASP.NET Client

After completing this module, you will be able to create and use components in Windows Form-based and ASP.NET-based applications. This includes:

- Creating a simple .NET Framework component in C#.
- Implementing structured exception handling.

- Creating a simple .NET Framework console application that calls a component.
- Creating a .NET Framework client application by using the Windows Forms library.
- Creating an ASP.NET page that uses the previously developed .NET Framework component to create an ASP.NET application.

Module 4: Deployment and Versioning

The following topics are covered in this module:

- Introduction to Application Deployment
- Application Deployment Scenarios
- Related Topics and Tools

After completing this module, you will be able to use the deployment and versioning features of the .NET common language runtime to deploy multiple versions of a component. This includes:

- Packaging and deploying simple and componentized applications.
- Creating strong-named assemblies.
- Installing and removing assemblies from the global assembly cache.
- Configuring applications to control binding based on assembly location and version data.

Module 5: Common Type System

The following topics are covered in this module:

- An Introduction to the Common Type System
- Elements of the Common Type System
- Object-Oriented Characteristics

After completing this module, you will be able to create, use, and extend types. This includes:

- Describing the Common Type System architecture.
- Describing the difference between value types and reference types.
- Explaining the purpose of each element in the type system, including values, objects, and interfaces.
- Explaining how object-oriented programming concepts, such as abstraction, encapsulation, inheritance, and polymorphism, are implemented in the Common Type System.

Module 6: Working with Types

The following topics are covered in this module:

- **System.Object** Class Functionality
- Specialized Constructors
- Type Operations
- Interfaces
- Managing External Types

After completing this module, you will be able to create classes and interfaces that are functionally efficient and appropriate for given programming scenarios. This includes:

- Applying attributes to control visibility and inheritance in classes and interfaces.
- Creating and using interfaces that define methods and properties.
- Explaining how boxing and unboxing work and when boxing and unboxing occur.
- Using operators to determine types at run time and to cast values to different types.
- Explaining what features are available to work with unmanaged types, such as COM types.

Module 7: Strings, Arrays, and Collections

The following topics are covered in this module:

- Strings
- Terminology – Collections
- .NET Framework Arrays

- .NET Framework Collections

After completing this module, you will be able to use the .NET Framework class library to create and manage strings, arrays, collections, and enumerators. This includes:

- Parsing, formatting, manipulating, and comparing strings.
- Using the classes in the **System.Array** and **System.Collections** namespaces.
- Improving the type safety and performance of collections by using specialized collections and class-specific code.

Module 8: Delegates and Events

The following topics are covered in this module:

- Delegates
- Multicast Delegates
- Events
- When to Use Delegates, Events, and Interfaces

After completing this module, you will be able to use delegates and events to have an

Module 9: Memory and Resource Management

The following topics are covered in this module:

- Memory Management Basics
- Non-Memory Resource Management
- Implicit Resource Management
- Explicit Resource Management
- Optimizing Garbage Collection

After completing this module, you will be able to describe and control how memory and other resources are managed in the .NET Framework. This includes:

- Describing how garbage collection manages object memory.
- Implicitly managing non-memory resources by using a destructor's finalize code.
- Explicitly managing non-memory resources by using client-controlled deterministic release of resources.
- Writing code by using the temporary resource usage design pattern.
- Programmatically controlling the behavior of the garbage collection.
- Describing advanced garbage collection features.

Module 10: Data Streams and Files

The following topics are covered in this module:

- Streams
- Readers and Writers
- Basic File I/O

After completing this module, you will be able to read from and write to data streams,

© 2001 Microsoft Corporation. All rights reserved.

Some elements of this course syllabus are subject to change. This syllabus is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. Microsoft, MS-DOS, Windows, Windows NT, ActiveX, IntelliMirror, JScript, MSDN, MSN, PowerPoint, SQL Server, Visual Basic, Visual C++, Visual C#, Visual FoxPro, Visual Studio, Win32, and Windows Media are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

event sender object signal the occurrence of an action to an event receiver object. This includes:

- Using the delegate class to create type-safe callback functions and event-handling methods.
- Using the **event** keyword to simplify and improve the implementation of a class that raises events.
- Implementing events that conform to the .NET Framework guidelines.

files, and the Internet. This includes:

- Using **Stream** objects to read and write bytes to backing stores, such as strings and files.
- Using **BinaryReader** and **BinaryWriter** objects to read and write primitive types as binary values.
- Using **StreamReader** and **StreamWriter** objects to read and write characters to a stream.
- Using **StringReader** and **StringWriter** objects to read and write characters to strings.
- Using **Directory** and **DirectoryInfo** objects to create, move, and enumerate through directories and subdirectories.
- Using **FileSystemWatcher** objects to monitor and react to changes in the file system.
- Explaining the key features of the .NET Framework's isolated storage mechanism.

Module 11: Internet Access

The following topics are covered in this module:

- Internet Application Scenarios
- The WebRequest and WebResponse Model
- Application Protocols
- Handling Errors
- Security
- Best Practices

After completing this module, you will be able to use the .NET Framework classes to work with data over the Internet. This includes:

- Using the basic request/response model to send and receive data over the Internet.
- Using the **System.Net** classes to communicate with other applications by

using the Hypertext Transfer Protocol (HTTP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Socket Internet protocols.

Module 12: Serialization

The following topics are covered in this module:

- Serialization Scenarios
- Serialization Attributes
- Object Graph
- Serialization Process
- Serialization Example
- Deserialization Example
- Custom Serialization
- Custom Serialization Example
- Security Issues

After completing this module, you will be able to serialize and deserialize an object graph. This includes:

- Writing an application that serializes and deserializes an object graph by using either a binary or Simple Object Access Protocol (SOAP) XML format.

Module 13: Remoting and Web Services

The following topics are covered in this module:

- Remoting
- Remoting Configuration Files
- Web Services

After completing this module, you will be able to create distributed applications by means of Web Services and Object Remoting. This includes:

- Writing and configuring distributed applications that use .NET Remoting.
- Creating a Web Service by using Visual Studio .NET and ASP.NET.
- Consuming a Web Service by using the Web Services Description Language tool (Wsd.exe).