The Philosophy of .NET

Microsoft's .NET platform (and the related C# programming language) were formally introduced circa 2002 and have quickly become a mainstay of modern-day software development. As mentioned in the book's introduction, the goal of this text is twofold. The first order of business is to provide you with a deep and detailed examination of the syntax and semantics of C#. The second (equally important) order of business is to illustrate the use of numerous .NET APIs, including database access with ADO.NET and the Entity Framework (EF), user interfaces with Windows Presentation Foundation (WPF), service-oriented applications with Windows Communication Foundation (WCF), and web service and web site development using ASP.NET MVC. The last part of this book covers the newest member of the .NET family, .NET Core,which is the cross-platform version of the .NET platform. As they say, the journey of a thousand miles begins with a single step; and with this I welcome you to Chapter 1.

The point of this first chapter is to lay the conceptual groundwork for the remainder of the book. Here you will find a high-level discussion of a number of .NET-related topics such as assemblies, the Common Intermediate Language (CIL), and just-in-time (JIT) compilation. In addition to previewing some keywords of the C# programming language, you will also come to understand the relationship between various aspects of the .NET Framework, such as the Common Language Runtime (CLR), the Common Type System (CTS), and the Common Language Specification (CLS).

This chapter also provides you with a survey of the functionality supplied by the .NET base class libraries, sometimes abbreviated as BCLs. Here, you will get an overview of the language-agnostic and platform-independent nature of the .NET platform. As you would hope, many of these topics are explored in further detail throughout the remainder of this text.

An Initial Look at the .NET Platform

Before Microsoft released the C# language and .NET platform, software developers who created applications for the Windows family of operating system frequently made use of the COM programming model. COM (which stands for the Component Object Model) allowed individuals to build libraries of code that could be shared across diverse programming languages. For example, a C++ programmer could build a COM library that could be used by a Visual Basic developer. The language-independent nature of COM was certainly useful; however, COM was plagued by a complicated infrastructure and a fragile deployment model and was possible only on the Windows operating system.

Despite the complexity and limitations of COM, countless applications have been successfully created with this architecture. However, nowadays, the majority of applications created for the Windows family of operating systems are not created with COM. Rather, desktop applications, web sites, OS services,and libraries of reusable data access/business logic are created using the .NET platform.