Chapter 18. Reflection and Metadata

As we saw in Chapter 17, a C# program compiles into an assembly that includes metadata, compiled code, and resources. Inspecting the metadata and compiled code at runtime is called *reflection*.

The compiled code in an assembly contains almost all of the content of the original source code. Some information is lost, such as local variable names, comments, and preprocessor directives. However, reflection can access pretty much everything else, even making it possible to write a decompiler.

Many of the services available in .NET and exposed via C# (such as dynamic binding, serialization, and data binding) depend on the presence of metadata. Your own programs can also take advantage of this metadata and even extend it with new information using custom attributes. The `System.Reflection` namespace houses the reflection API. It is also possible at runtime to dynamically create new metadata and executable instructions in Intermediate Language (IL) via the classes in the `System.Reflection.Emit` namespace.

The examples in this chapter assume that you import the `System` and `System.Reflection` as well as `System.Reflection.Emit` namespaces.