

deterministic finalization was already mentioned at this time. I wonder why it didn't upset developers as much then.)

- A strong security model for the components.
- Functionality similar to `SqlDataAdapter` in ADO.NET. (An `SqlDataAdapter` can be used to register which stored procedure should be called for inserting new rows that have been added to a `DataSet`, for updating, and so on.)
- No need for a separate interface language (such as IDL).
- All languages decide on a common set of types.
- Parameterized constructors and overloading.

The good news is that .NET does deliver on all of these ideas and, even years later, it's an impressive list. Although Mary's discussion was based on the idea to build the runtime and everything else on COM, .NET isn't written on COM; it's a completely new platform.

In addition, .NET also includes the following features:

- Remoting (for example with SOAP)
- ADO.NET
- ASP.NET
- Windows Forms
- Web Forms
- XML Web services support
- Base Class Library (BCL)
- A new and a much more competent versioning schema
- A new language called C#
- Many changes to VB6 (or rather a rewrite) to make it Visual Basic .NET

It's important to note that even though the common language runtime is extremely important in .NET, it does not replace COM+. This means if your components written in managed code (that is, code managed and executed completely by the common language runtime) need component services, such as declarative transactions, you will still use our old friend COM+. We will discuss serviced components (components that use component services) further in a minute.

In the end, you may be asking yourself whether it is an advantage or a disadvantage to have prior experience with COM and COM+ when you enter the new world of .NET. In my opinion, it's definitely an advantage.

.NET Component Services

.NET isn't really an evolution, it's a revolution. Even so, writing serviced components (that is, using component services in .NET) is, in essence, evolutionary. If you have worked with writing components that use COM+ component services, you will be well prepared.

Listing 1.1 shows what a very simple serviced component would look like as a class in Visual Basic .NET. (For it to work, you must set a reference to `System.EnterpriseServices.dll`.)

Listing 1.1 A Simple Serviced Component in Visual Basic .NET

```
Imports System.EnterpriseServices

Public Class MyFirstSample
    Inherits ServicedComponent

    Public Function GetTime() As String
        Return Now.ToString()
    End Function
End Class
```

When this class is used for the first time, it will automatically be registered as a configured COM+ component, which comes in handy at development time. Still, at deployment time, this feature is most often not useful, because the user running the application must have administrator privileges to have it registered. You can use `RegSvcs` to manually register the component as a configured COM+ component.

COM+ object pooling will now be available to the majority of COM+ programmers, namely those who use VB6, when they shift to a .NET language, such as Visual Basic .NET or C#. An added plus is that the components you write with Visual Basic .NET and C# will not have the thread affinity problems—which means, for example, that an object of a VB6-written class can only be used from the same thread as the one that instantiated the object because of the heavy use of Thread Local Storage (TLS)—and Single Threaded Apartment only threading model of VB6.