

A very handy feature of .NET is that the attributes can be written in code. Listing 1.2 shows an example in which a class is decorated with attributes for object pooling.

Listing 1.2 Attributes Decorating a Class

```
<ObjectPoolingAttribute(Enabled:=True, _  
MinPoolSize:=2, MaxPoolSize:=3, _  
CreationTimeout:=20000)> Public Class MyPooledClass
```

Special Considerations

You may be wondering whether the .NET programmer has to consider anything special when writing serviced components. Indeed, several things differ between writing ordinary .NET components and writing serviced ones, including the following:

- You should call `Dispose()` when you are done with an instance of a serviced component. (Most often it's not a must, but recommended.)
- You should not use nondefault constructors.
- Static methods are not serviceable.
- Services flow between machines, but only when DCOM is used. (That means, for example, that you must use DCOM instead of .NET Remoting if you want to use a COM+ transaction that spans two machines.)
- During runtime, the user executing the COM+ application must have permission to run unmanaged code.

As usual, consumers won't have to know whether the used component is serviced or not. Of course, the consumers can be unmanaged code, such as VB6-written code. In fact, using .NET to write better COM+ components for unmanaged consumers will likely be a common scenario in the near future.

COM+ 1.5 Component Services

Windows XP and Windows .NET Server will contain COM+ 1.5 and with it a lot of new functionality, including:

- Process recycling
- Configurable isolation level
- Applications as Windows services
- Memory gates
- Pause/disable applications
- Process dumping
- Moving and copying components
- Public/private components
- Application partitions
- COM+ applications exposed as XML Web services

We will discuss each of these features in the rest of this section.

Process Recycling

A simple way of escaping the problems with memory leakage is to restart the server process now and then. With process recycling, you can declaratively tell how and when this should happen—for example, after a certain amount of requests or every night.

Note

I once wrote a Web application that had a memory leak. The customer thought it was a cheap solution for them to just reboot the machine every weekend because they had people working then anyway. I preferred to try to solve the problem instead of using a Band-Aid, but they didn't see this as a problem at all. Of course, process recycling would have been more convenient for them.

Configurable Isolation Level

In COM+ 1.0, the transaction isolation level was always `SERIALIZABLE` for COM+ transactions against SQL Server. (Oracle gets the same request to set the isolation level to `SERIALIZABLE`, but neglects this and uses `READ COMMITTED` instead. This is because Oracle uses versioning instead of locking for concurrency control, and then `READ COMMITTED` is most often enough.)³ In the past, this