CHAPTER 11

VB.NET:
Brave New World

You just finished learning all about Visual Basic 6 and now here's a chapter on the next version already. Hold on because VB.NET is much more than just a new version—it's a whole new game altogether. Let's deal with the question on everyone's mind: why a new Visual Basic—what's wrong with Visual Basic 6? Sure, we can all think of improvements to VB6 and the Visual Basic community has never been shy about suggestions for the next release. But VB.NET? There are those who claim that VB.NET is not VB at all but a whole new language that is, well, VB-like. I have to confess that I am a member of that group. That's because Microsoft really didn't take Visual Basic 6 as a base and modify it—it's entirely new. Oh, yes, many of the same capabilities are there and it looks somewhat familiar but the underlying run time has changed. VB.NET is as close to Visual Basic 6 as the new VW Beetle is to the classic Beetle. They look a bit alike but we can all tell the difference right away.

So, again, why a new Visual Basic? Well, in spite of what we in the VB community may think, we are not the whole world. Microsoft has for years been supporting vastly different kinds of programmers with fundamentally different products. This has had its benefits. Visual Basic, up to this point, has been the most successful programming language ever, drawing into the programming world quasi-technical types, like us testers for example, who would never have ventured so far if we hadn't been lured by the sexy simplicity and elegance of Visual Basic. The C++ developers have been able to do what they do best: get down pedal-to-the-metal writing the code that makes the Windows world go-round. Perhaps this has caused a kind of corporate schizophrenia that they wanted to remedy but they really do want to provide the fabulous parts of each product to the whole programming community. And trying to upgrade Visual Basic to include more complex features created workarounds that were getting increasingly rickety. To keep growing Visual Basic for an era where the Web is becoming the delivery platform of choice, Microsoft had to make a big change at some point. That time is now.

What are those fabulous new things we get with VB.NET? With each release of Visual Basic, users have demanded more object-oriented capability. Visual Basic will be a fully object-oriented language with the release of .NET, or at least...
as much so as one can be to date. We get some confusing parts of Visual Basic fixed, too. For example, having to understand why we had to use Let and Set for object variables was a pain (Chapter 9)—Let and Set are eliminated in VB.NET. Web development is vastly improved, which is something the Visual Basic community has been wanting for years. And we will get the all-important interoperability with other Windows languages since all .NET languages are the same at the core. In particular, VB.NET code will run as fast as C++ or C# code on the .NET platform: VB is no longer a second-class citizen. So, we're getting what we asked for. Conversely, C++ language developers will gain the ease of RAD (Rapid Application Development) using the best elements of Visual Basic's great IDE that we have enjoyed. ASP developers gain huge new features and performance with ASP.NET. And .NET adds much greater capacity for Intranet and Internet applications. Combining these features into a common development environment on top of a common language core—called the Common Language Runtime (CLR)—is risky and courageous. Will it pay off? I think it will. But it will be a long time before Visual Basic 6 is gone. So, for those of us in the testing community, don't throw out your VB test scripts just yet. Even though Visual Basic is scheduled to be released this year, we've got awhile. A transition this big will take a long time.

Now, for some of the changes: The VB.NET IDE will look familiar with similar windows so it should only take a little stumbling before you can drag and drop a button on the form. There are some new windows. (You'll see more on all of this later in this chapter.) As far as languages changes, readers of this book will find that since much of what we covered was general programming, we won't have to change much. There are some exceptions to that, which you will find mostly annoying but not terribly problematic; here's a sampling:

- There's no Debug.Print; it is now Debug.WriteLine and it goes to the new Output window and not the Immediate window. The Immediate window is still there but is used to enter and check values. You also can't use it to evaluate expressions except when the program is in Break mode (as was the case with early versions of VB).

- The default data type is Object, not Variant:

  ```vba
  Dim myvar as Variant 'old
  Dim myvar as Object   'new
  ```