.NET开发丛书

Microsoft .net

PEARSON Addison Wesley

ASP.NET基础教程

——Visual Basic .NET案例版

Essential ASP.NET

with Examples in Visual Basic .NET

(美) Fritz Onion 著

施 诺 译

Pearson Education

清华大学出版社

.NET 开发丛书

ASP.NET 基 础 教 程

——Visual Basic .NET 案例版

（美） Fritz Onion 著

施 诺 译

清华大学出版社

北 京

内 容 简 介

本书结合用 Visual Basic .NET 语言编写的可实际运行的示例代码，讨论了 ASP.NET 的构架、Web 窗体、配置、HTTP 管道、故障诊断和错误处理、验证、数据绑定、自定义控件、缓存、状态管理和安全性，阐述了用 Visual Basic .NET 构建基于 Web 应用程序的最佳实践。

本书既可以作为软件工程专业的学生的参考书，也可以作为 VB 程序员学习 ASP.NET 的参考书。

EISBN: 0-201-76039-8

北京市版权局著作权合同登记号 图字：01-2003-3536

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

图书在版编目（CIP）数据

ASP.NET 基础教程——Visual Basic .NET 案例版/（美）奥尼恩著；施诺译. —北京：清华大学出版社，2003
（.NET 开发丛书）
书名原文：Essential ASP.NET with Examples in Visual Basic .NET
ISBN 7-302-07313-9
Ⅰ.A… Ⅱ.①奥… ②施… Ⅲ.①主页制作—程序设计 ②BASIC 语言—程序设计 Ⅳ.①TP393.092 ② TP312
中国版本图书馆 CIP 数据核字（2003）第 086729 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969

文稿编辑：文开棋
封面设计：立日新设计公司
印 刷 者：北京牛山世兴印刷厂
发 行 者：新华书店总店北京发行所
开　　本：185×230　**印张**：21　**字数**：460 千字
版　　次：2003 年 10 月第 1 版　2003 年 10 月第 1 次印刷
书　　号：ISBN 7-302-07313-9/TP・5311
印　　数：1～3500
定　　价：39.00 元

序

让一个公司迁移到.NET 框架、迁移到公共语言运行库（Common Lauguage Runtime，CLR）有很多很好的理由。例如，最新的开发工具使代码的编写、测试和调试更加容易。FCL（Framework Class Library）提供了很多功能，多得令人难以置信，远远超出了 Visual Basic 以前的版本。CLR 使应用程序及其关联的 DLL 的部署更加容易，而且成本也更低。此外，CLR 提供了许多新的附加安全层，以防范病毒和蠕虫等恶意软件的攻击。然而，在公司认为要迁移到.NET 框架的所有理由中，ASP.NET 似乎是重要理由之一。

ASP.NET 本身是.NET 框架最成熟的、最具活力的开发工具之一。ASP.NET 是一个服务器端应用程序框架，为客户、雇员和厂商等用户创建 Web 应用程序提供了一个完整的开发平台。ASP.NET 之所以功能强大，是因为它能够使您通过 Internet 接触到运行着许多不同的浏览器和操作系统的用户。

在过去的五、六年内，软件业亲眼目睹了上百家公司用最初的 ASP 框架成功地构建了大型应用程序。然而，还是这些公司，现在有许多已经意识到迁移到 ASP.NET 具有的巨大好处。这些好处包括业绩提升、开发人员的生产率提高以及代码更好维护。ASP.NET 还引进了用于缓存和状态管理的可贵的新特性，可以进一步提高应用程序的性能及其在 Web farm 环境中的扩展能力。

虽然关于 ASP.NET 主题已有很多不同的书籍，但是我强烈地鼓励您阅读本书，彻底消化书中的材料。Fritz Onion 通过彻底而详尽地剖析 ASP.NET 的底层构架，教您学习 ASP.NET。如果像我一样，不满足于仅仅知道如何用 Visual Studio.NET 构建简单的 Web 应用程序，而是想真正地了解 ASP.NET 的工作原理，那么必须能够回答如下问题：

- 为什么 ASP.NET 应用程序的运行速度比 ASP 应用程序快？
- 为什么影子复制使应用程序的部署更容易？
- Web 窗体如何同服务器端控件进行交互？
- 关于 ASP.NET 配置，您真正需要知道什么？
- ASP.NET 管道的内部工作机制是什么？
- 如何创建应用程序范围的异常处理程序？

- 关于控件验证和数据绑定的构架，要想将这些特性集成在自定义的控件中，需要知道些什么？
- 为什么输出缓存很重要，如何有效地利用它？
- ASP.NET 提供的用于管理各请求间状态的各种选择的折衷考虑是什么？
- ASP.NET 在哪些方面增强了 Windows 和 IIS 提供的安全机制？

鼓励大家阅读《ASP.NET 基础教程——Visual Basic .NET 案例版》，因为本书不仅提供了许多实用的建议，而且还提供了 ASP.NET 的工作原理。只要掌握了本书介绍的知识，就能够做出理智的判断：ASP.NET 开发者面临哪些最重要的设计问题。掌握这些知识后，可编写出性能更优、功能更多的 Web 应用程序。

Ted Pattison
作家、讲师兼研究员
于 2002 年 11 月

前　言

这是在 2000 年 8 月的一个深夜，地点是加利福尼亚州南部的托兰斯。这天，我已经同 Mike Woodring 和 Jason Whittington 上了 12 个小时 DevelopMentor's Guerrilla COM 课程。课后 Don Box 来到我这里，同往常一样，在学生们已经入睡后，我们还在讨论有关的技术和黑客问题，一直到深夜。Microsoft 公司 7 月份在 PDC 刚刚发布了.NET 的测试版本，而这一年我们已经花了大量时间来研究“下一代 COM”，我们为它的最终发布而倍感兴奋，所以我们可以讨论它了。就是在这样的一个晚上，Don 以他特有的简洁的方式，让我第一次见识了 ASP.NET（当时称为 ASP+）。他首先在 emacs 中输入一个.aspx 文件，如下所示：

```
<%@ Page Language="C#" src="TestPage.cs"
    Inherits="TestPage" %>

<html>
<h1 runat=server id=ctl/>
</html>
```

然后，他编写了另一个文件，如下所示：

```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.HtmlControls;

public class TestPage: Page
{
    protected HtmlGenericControl ctl;
    void Page_Load(object src, EventArgs e)
    {
        ctl.InnerText = "Hello!";
    }
}
```

然后他把这两个文件放在机器的 c:\inetpub\wwwroot 子目录中，*并通过浏览器向我显*示.aspx 页面的生成过程，然后大声对我说“明白吗？”也许是因为太晚或者因上了一整天

课的缘故，但是我必须承认，尽管我“明白了”Don 所演示的问题，但是对能够从一个类中改变 h1 元素的 innerText，并没有很深刻的印象。

享受两晚的充足睡眠后，我回想起那个.aspx 例子，并开始比较详细地研究 ASP.NET。经过一天的阅读和试验以后，我终于“明白了”，而且我已经迷上了 ASP.NET。这种技术的提出，从根本上改变了人们在 Windows 上开发 Web 应用程序的方式，而且它充分利用了最新的.NET 运行库。之后，我用 6 个月时间研究和构建 ASP.NET 应用程序，并为 DevelopMentor 的 Essential ASP.NET 课程编写教材，在此后的一年半内，我一直从事 ASP.NET 的教学和讲座，以及撰写有关的教材。本书是这些活动的结晶，希望它能帮助您了解 ASP.NET。

C#与 VB.NET

在.NET 推出之前，Visual Basic 不仅仅是另一种语言——它本身就是一个平台。例如，用 Visual Basic 6.0 构建应用程序，完全不同于用 C++和 MFC 构建应用程序。推出.NET 以后，这种差别不复存在，而 Visual Basic 的确只是另一种.NET 语言而已，与所有其他语言一样使用相同的函数库、相同的开发工具和相同的运行库。因此，我们现在可以从语言中立的角度谈论 ASP.NET 等技术。然而，代码示例必须用某种特定的语言表示，因此本书推出了两个版本：一个版本用 C#编写例子，另一个版本用 VB.NET 编写例子。除了例子以外，这两本书的内容几乎是相同的。

示例代码、Web 站点、反馈

本书中的所有代码都取自可实际运行的示例，并且可以从本书的配套网站 *http://www.develop.com/books/essentialasp.net/*下载。本站点还包含本书勘误表，而且为本书中介绍的概念提供一系列更加丰富的例子供大家参考。作者欢迎大家通过该网站提供的表单提出意见、勘误和反馈。

预备知识

本书专门讲述 ASP.NET，而不再花时间复习.NET 编程、面向对象编程技术、数据库访问或者一般的 Web 应用程序开发技术。如果您已经具备这些领域的相关知识，就可以理解本书所介绍的大部分知识。

本书的结构

本书从 ASP.NET 的基础讲起，首先在第 1 章中讨论 ASP.NET 构架的核心元素，然后在第 2 章中讨论服务器端控件模型。建议读者在阅读本书的所有后续章节之前，应先熟悉第 1 章和第 2 章的内容。然而，第 2 章以后的各章都是独立的，可以按您希望的任意顺序阅读。

第 1 章“构架”，讨论 ASP.NET 构架的基础，首先解析.aspx 文件，随后将它们编译成程序集。然后详细解释 Page 类，举例说明最新的代码隐藏模型，讨论用来防止文件锁定的影子复制机制。本章最后还分析了 ASP.NET 中新增的类，这些类取代了传统 ASP 固有的对象。

第 2 章“Web 窗体”，讨论 ASP.NET 所支持的基于控件的编程模型，即所谓的 Web 窗体。还讨论使用 POST 主体数据和 ViewState 实现回送（post-back）状态保持（state retention）的细节，并描述如何有效地使用服务器端控件创建动态 Web 页面。最后还讨论了 ASP.NET 中可用的各种服务器端控件。

第 3 章“配置”，描述 ASP.NET 所用的配置模型。首先讲述所有配置文件使用的 XML 格式，以及配置设置的层次化应用。本章详细研究几个配置元素，包括 processModel 和 appSettings 元素。最后介绍了两种方法，用以举例说明如何在配置文件中添加自定义的配置节。

第 4 章“HTTP 管道”，详细研究 ASP.NET 中服务 HTTP 请求所涉及的类。首先分析 HTTP 管道中用来处理请求的所有元素，然后讨论管道中的 3 个可扩展点：自定义应用程序类、自定义处理程序和自定义模块。最后讨论管道中的线程和对象缓冲（object pooling）。

第 5 章“故障诊断和错误处理”，讨论 ASP.NET 的新的诊断特性，包括页面和应用程

序跟踪以及新的性能计数器。还讨论了用于调试 ASP.NET 应用程序和异常处理的技术。最后讨论如何为应用程序定义自定义的错误页面。

第 6 章“验证”，描述 ASP.NET 内建的新的验证构架。首先分析 Web 应用中一般是怎样执行验证的，然后说明 ASP.NET 的验证构架如何为验证用户输入问题提供一种通用方案。本章详细分析了客户端和服务器的验证工作，而且还分析了所有可用的验证控件。

第 7 章“数据绑定”，分析如何将服务器端数据与 ASP.NET 页面中的控件进行绑定。本章首先说明数据绑定在多种不同的数据源下（包括集合类、DataReader 和 DataSet）是如何工作的，然后说明如何将数据与几个控件进行绑定，包括 DataGrid 类。最后讨论了模板，以及如何结合 Repeater，DataList 和 DataGrid 类有效地使用它们。

第 8 章“自定义控件”，讨论如何构建自己的自定义控件以供 ASP.NET 应用程序使用。阐明了如何构建自定义控件、如何使用 HtmlTextWriter 类实现独立的浏览器、如何进一步支持与浏览器无关的生成、如何定义属性和子属性、如何提取一个控件标记的内部内容以及如何生成客户端脚本以及如何管理控件状态。与此同时，还讨论了如何构建复合控件、用户控件、支持验证的控件以及支持数据绑定的控件。最后还讨论了如何将控件与 Visual Studio .NET 设计器集成起来。

第 9 章“缓存”，讨论 ASP.NET 的输出缓存和数据缓存，输出缓存的机制，如何精确地控制把一个页面的哪个版本放入缓存，以及如何以用户控件的形式用页面分段高速缓存对一个页面的某些部分进行缓存。阐明了如何运用新的应用程序范围的数据缓存，并讨论了数据缓存时应关注的注意事项和准则。

第 10 章“状态管理”，讨论 ASP.NET Web 应用程序中各种类型的状态，并说明如何使用每种类型以及何时使用。首先讨论了应用程序状态，解释了为何要在 ASP.NET 中应避开它（通常如此）。然后，讨论了会话状态的改进，包括进程外存储和无 cookie 密钥管理，以及优化使用会话状态的技术。最后讨论了如何用 cookie 和视图状态来取代会话状态。

第 11 章“安全性”，描述了 ASP.NET 的安全特性，以及如何控制应用程序的客户身份验证和授权。首先回顾 Web 应用程序的安全性概念，接着解释如何构建和管理需要用 ASP.NET 提供的窗体验证构架对客户进行身份验证的应用程序。本章还讨论了如何在 Web farm 中认证 cookie 的管理、安全的密码存储、构建基于角色的身份验证系统以及如何控制 ASP.NET 运用的进程身份。

致　谢

首先感谢我的妻子 Susan，我的孩子 Zoë 和 Som，感谢他们在我撰写本书期间的全力支持。还要感谢我的父母 Pat 和 Dan Onion，感谢他们的支持和指导。

感谢我在 DevelopMentor 的所有同事，无论课程，还是本书，我与他们进行了多次讨论，并总能从他们那里得到诚恳建议。尤其要感谢 Bob Beauchemin（*Essential ADO.NET* 的作者），他总是及时地提出有用的意见；感谢 Keith Brown 告诉我如何删除无关紧要的信息，而要增强安全性方面的内容；感谢 Simon Horrell 为本书提供的详尽反馈；感谢 Dan Sullivan，为我解决了众多难题；感谢 Ted Pattison 对我写作过程的关心和始终如一的明确的意见；感谢 Stu Halloway 使本书更加简洁；感谢 Mike Woodring 与我一同考虑同步处理程序蕴含的线程问题。

感谢我的指定审校人：Justin Burtch，Amit Kalani，Daryl Richter，Martin Heller，以及 Matt Milner，他们都提出了宝贵意见。感谢 Microsoft 公司的 ASP.NET 团队的成员构建了如此有趣的一个产品，感谢他们为我解答了那么多问题。特别要感谢 Rob Howard 对缓存部分提供的建议，以及 Erik Olson 在解释管道中线程分配和对象分享方面提供的帮助。感谢 Don Box 把我带入了 ASP.NET 领域，在他的帮助下，我于 1995 年开始了写作生涯。

感谢我的编辑 Stephane Thomas，感谢他为本书付出的辛勤劳动；感谢本书的文稿编辑 Debby English，不愧为 English 先生。还要感谢上千名参与 Essential ASP.NET 培训课程的学生：本书内容取材于他们的建议。特别要感谢 2002 年 10 月在华盛顿进修 ASP.NET 课程的学生们，他们帮助我选择了本书的封面。

Fritz Onion
Wells, Maine
2002 年 10 月
http://staff.develop.com/onion

目　录

第1章 构 架

ASP.NET 为 Web 应用程序开发者引入了许多新特性，这些特性包括已编译的服务器端代码、一种将服务器端逻辑与客户端布局相分离的代码隐藏（code-behind）技术、可扩展的服务器端控件模型、设计良好且易于使用的数据绑定模型、xcopy 部署、以及客户和服务器上的窗体身份验证支持。然而，除了上述的所有特性外，ASP.NET 还为我们提供了一种统一性：语言、工具、库、部署模型、系统设计和故障诊断的统一性。Web 应用程序开发者再也不用区别对待页所用的组件与构架中其他地方所用的组件；再也不用借助于脚本调试器来诊断页中存在的问题。他们可以摆脱许多不同规格的脚本语言（这些语言的细微之处经常使人迷惑）的束缚，使用他们拿手的.NET 语言来构建页。现在，在.NET 平台上构建 Web 应用程序如同在.NET 平台上开发其他任何应用程序一样简单。

本章介绍如何用 ASP.NET 构建 Web 应用程序的构架基础；讨论用于处理请求的新的编译模型、如何构建代码隐藏类、如何用影子复制（shadow copy）机制实现 xcopy 部署，以及可用的新指令（directive）和内部对象（intrinsic）。

1.1 基础知识

ASP.NET 的核心是许多.NET 类，它们相互间密切合作，对 HTTP 请求提供服务。在这些类中，一些类在系统程序集中定义，作为基类库的一部分，并随.NET 运行库一起安装；一些类被载入全局程序集缓存（global assembly cache，GAC），还有一些类从与应用程序关联的虚目录中的本地程序集中载入。所有这些类都被载入 ASP.NET 工作者进程内的应用程序域中，而且它们相互作用对给定的请求作出响应。图 1.1 显示了这一构架。

大多数开发者在从其他编程环境转向 ASP.NET 时，都会面对基本的思想转变：要做的一切就是从一个程序集中装入一个类。类似于其他基于类的构架，在 ASP.NET 中构建应用程序的过程就是构造与基础框架中的类交互的类的过程。在用户编写的类中，有一些派生于基础框架中的基类，有一些可能会实现框架中定义的接口，而其余的类则只是简单地通过调用框架中的基类的方法来实现与它们的交互。尽管 ASP.NET 还支持 ASP 风格的语法，但是 ASP.NET 文件中的服务器端代码被转化为类定义，并在第一次访问时被装入

程序集中。因此，最终的结果是，许多类在一个进程中互相协作，对请求提供服务。

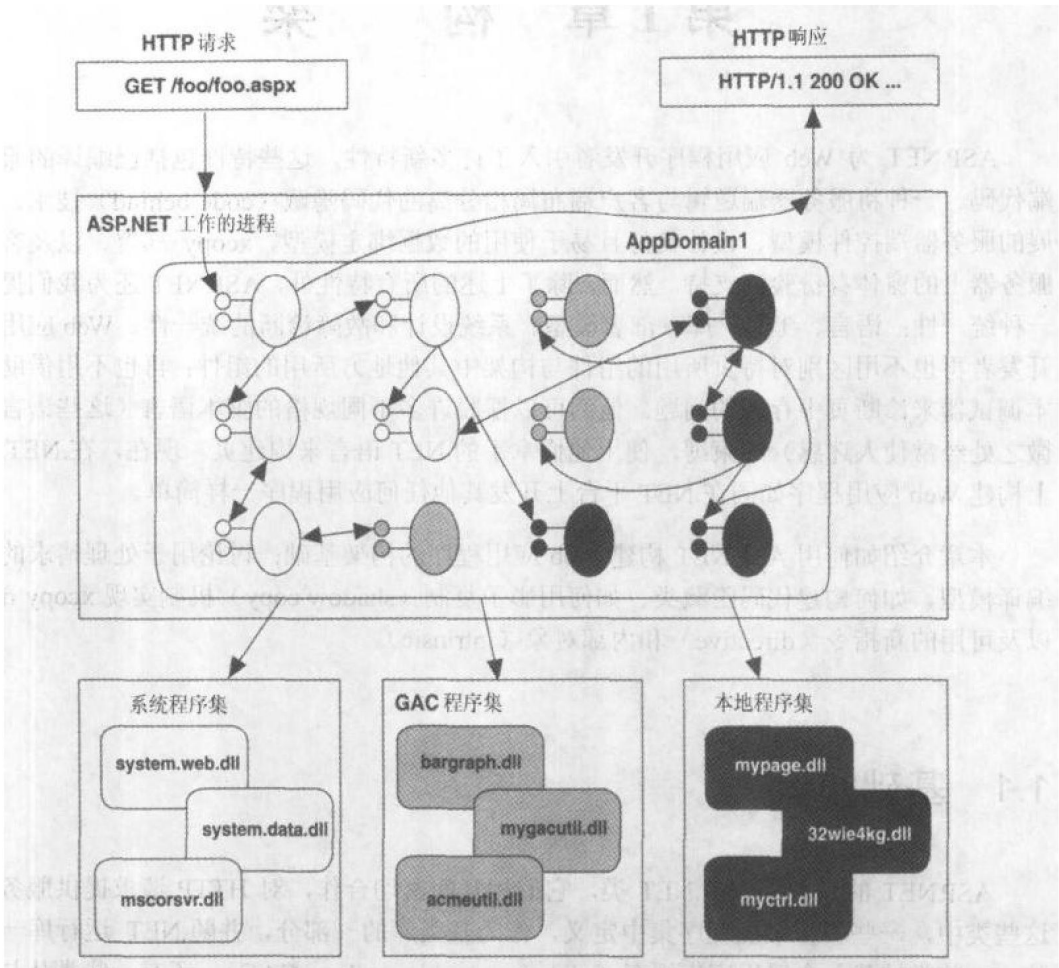

图 1.1 ASP.NET 的高层构架

另一个重大的转变是进程模型（process model）。ASP.NET 工作者进程（worker process）是从 inetinfo.exe（Internet 信息服务器进程）中分离出来的一个单独的工作者进程——aspnet_wp.exe[①]，ASP.NET 中的进程模型与 IIS 中有关的进程隔离设置无关。尽管 IIS 仍然

① 在 IIS 6.0（Windows Server 2003 带有该版本的 IIS）中，工作者进程模型有些不同，我们将在第 3 章对此进行讨论。这里值得注意的是，如果不是以 IIS 5.0 隔离模式运行 IIS 6.0，ASP.NET 将驻留在一个称为 w3wp.exe 的进程中。

是访问 ASP.NET 应用程序的典型入口点[①]，在物理上侦听相应的端口，分发请求，但是它的作用有所减弱，并且它所处理的许多任务都可以由 ASP.NET 在它自己的工作者进程中完成。

1.2　ASP 4.0

尽管 ASP.NET 在技术上不称为 ASP 4.0，但从许多方面来说，它就是 ASP 4.0，即 ASP 的下一个版本。因为大多数使用 ASP.NET 的程序员都具备 ASP 知识，所以我们先介绍这两种技术共有的特性。程序 1.1 列举了一个简单的 ASP 页，其中混合了静态 HTML 和 JavaScript。注意，该文件举例说明了几种服务器端编码技术。有一个用 runat=server 属性标记的脚本块，它包含一个 Add 函数。服务器端的求值语法“<%=”，用于调用 Add 函数，并将表达式的结果输出给 h2 标记。服务器端脚本语法“<%”，以编程方式生成一个有 10 行的表格，最后，内部对象 Response 以编程方式在页的底部添加一个 h2 标记。

程序 1.1　ASP 页示例

```
<!-- File: test.asp -->
<%@ language='javascript' %>

<script language='JScript' runat=server>
function Add(x, y)
{
  return x+y;
}
</script>

<html> <body>
<h1>Test ASP Page</h1>

<h2>2+2=<%=Add(2,2)%></h2>
<table border=2>
<%
  for (var i=0; i<10; i++) {
```

① 事实证明，可以在没有 IIS 作为前端 Web 服务器的情况下使用 ASP.NET。Cassini 是由 Microsoft 制作的一个样本 Web 服务器，其完整的源代码可以从 *http://www.asp.net* 下载。Cassini 与其他项目一起，可以使我们在 Apache 中提供 ASP.NET 服务。

```
%>
  <tr><td>Row<%=i%> Col0</td><td>Row<%=i%> Col1</td></tr>
<%
  }
%>
</table>

<%
  Response.Write("<h2>Written directly to Response</h2>");
%>

</body> </html>
```

ASP.NET 也支持所有这些服务器端编程技术。事实上，如果选择这个文件，并简单地将其后缀名由“.asp”改为“.aspx”，将会发现“.aspx”文件与“.asp”文件的执行情况完全一样。但正如后文所述，在访问这两个文件时，其背后所发生的事情却截然不同。从表面上看，许多传统的 ASP 页可以不需任何修改即可转化为 ASP.NET 页。

在 ASP.NET 中，不再局限于传统的 ASP 中使用的两种脚本语言：VBScript 和 JavaScript。[①]所有与.NET 完全兼容的语言都可用于 ASP.NET，包括 C#和 VB.NET。为了用一个例子进行说明，我们可以用 C#作为服务器端语言，重写程序 1.1 中的 ASP 页，使之成为 ASP.NET 页。我们使用 Page 指令在该页中加入一个语言选择（该语言选择指令并不是必需的），它控制大部分 ASP.NET 页的页级属性。程序 1.2 列出了用 VB.NET 重写的页，其中使用了一个 ASP.NET 的“Page”指令。

程序 1.2 .aspx 页示例

```
<!-- File: test.asp -->
<%@ Page language='VB' %>

<script runat=server>
Function Add(x As Integer, y As Integer) As Integer
  Return x+y
End Function
</script>

<html> <body>
```

① 正如我们所知道的，JScript 是一种完全支持.NET 的语言，可用于 ASP.NET 页。相反，ASP.NET 不直接支持 VBScript，尽管 ASP.NET 可以使用目前已经相当完善的 Visual Basic .NET。