第 1 章
ASP.NET 框架概述

ASP.NET是一个开源的Web框架，用于与HTML、CSS和JavaScript配合构建出色的网站和Web应用程序。此外，它还支持创建Web API，并能够使用实时技术（例如Web套接字）实现高度的双向通信。本章将详细讲解ASP.NET框架及其相关背景知识。

1.1 C/S 架构和 B/S 架构

C/S架构是一种典型的两层架构，其全称是Client/Server，即客户端/服务器端架构。其客户端包含一个或多个在用户的计算机上运行的程序。而服务器端有两种：一种是数据库服务器端，客户端通过数据库连接访问服务器端的数据；另一种是Socket服务器端，服务器端的程序通过Socket与客户端的程序通信。

B/S架构的全称为Browser/Server，即浏览器/服务器结构。Browser指的是Web浏览器，极少数事务逻辑在前端实现，主要事务逻辑在服务器端实现。Browser客户端、Web应用服务器端和数据库（Database）端构成所谓的三层架构。B/S架构的系统，B端无须特别安装，只要有Web浏览器即可。

对于C/S和B/S的区别，首先必须强调的是C/S和B/S并没有本质的区别，B/S是基于特定通信协议HTTP的C/S架构，也就是说B/S包含在C/S中，是特殊的C/S架构。

之所以在C/S架构上提出B/S架构，是为了满足瘦客户端、一体化客户端的需要，最终目的是节约客户端更新、维护的成本，以及广域资源的共享。

（1）B/S属于C/S，浏览器只是特殊的客户端。

（2）C/S可以使用任何通信协议，而B/S必须使用HTTP协议。

（3）浏览器是一个通用客户端，开发B/S应用本质上还是实现一个C/S系统。

1.2　网站开发概述

1.2.1　ASP.NET 网站的运行原理

一图胜千言，用户访问网站的过程如图1-1所示，注意图中的箭头线表示信息传递过程。

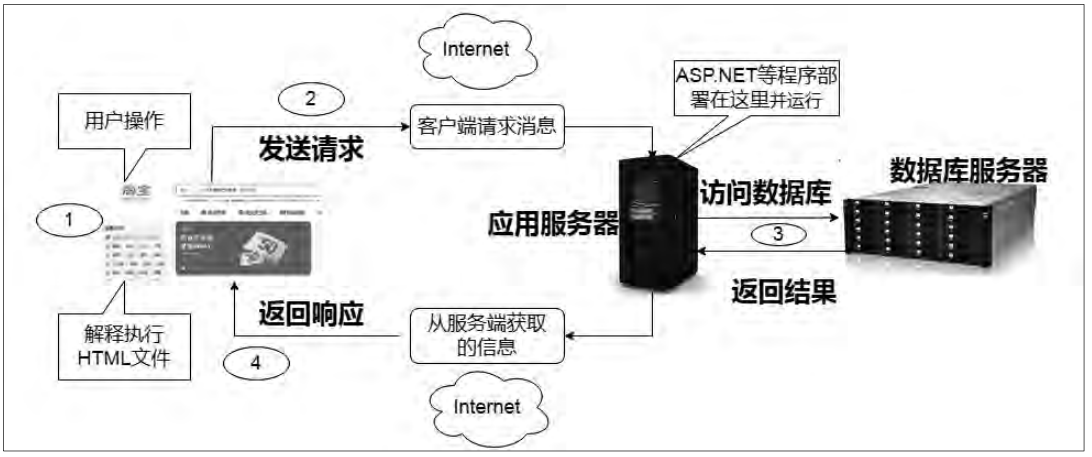

图 1-1

当请求发送至Web服务器并被其接收后，服务器会判断请求文件的类型：

（1）如果是静态文件，如HTML、JPG、GIF和TXT等，服务器会自行根据目录找到文件并发送给客户端。

（2）如果是动态文件，如aspx，服务器会通过aspnet_isapi.dll将请求转交给ASP.NET 运行时环境进行处理。ASP.NET会先检查代码是否已经被编译，如果没有，则将代码编译成MSIL（Microsoft Intermediate Language，微软中间语言），然后由JIT（Just-in-time，即时）编译器进一步编译成机器语言去执行。其中，JIT并非一次完全编译，而是调用哪部分代码就编译哪部分，这样用户的等待时间更短。同时，编译好的代码再次请求运行时不需要重新编译，极大提高了Web应用程序的性能。这种先将代码编译成中间语言，执行时再编译成机器语言的过程称为二次编译。

1.2.2　ASP.NET 的服务器

ASP.NET程序需要使用Web服务器作为发布平台，ASP.NET使用IIS（Internet Information Service，Internet信息服务）作为Web服务器。IIS是微软开发的Web服务器，它基于Windows操作系统，操作方便，功能强大。

实际上，在开发阶段并不用配置IIS，我们只需要像WinForm开发那样编码，然后单击运行就可以了。微软在Visual Studio中内置了一个轻量级的Web服务器，运行应用程序时，将会默认启动它，并在状态栏中出现图标，如图1-2所示。右击该图标，在弹出的快捷菜单中选择“显示所有应用程序”选项，可以查看当前正在运行的站点信息。

图 1-2

1.2.3　网站开发所需技能

对于网站开发各环节，我们用一幅图来表示，一目了然，如图1-3所示。乍看上去，要做好一个网站所需的技能蛮多的。网站开发前台页面技术包括页面设计（如HTML、CSS）、页面特效（JavaScript、图片图标）和前端框架。网站开发后台主要包括安全设计（比如用户认证、权限认证和防攻击）和后台框架技术（包括ASP.NET开发技术、基于WebForm开发企业网站、MVC框架学习、EF框架等）。数据存储技术包括Session、主流数据库（Redis、SQL Server或MySQL等）和数据存储框架，数据存储框架又包括数据库管理系统（DBMS）、数据模型、数据库语言、数据库引擎、数据库存储等。

一旦把这些东西全部掌握，走遍天下都不怕了。

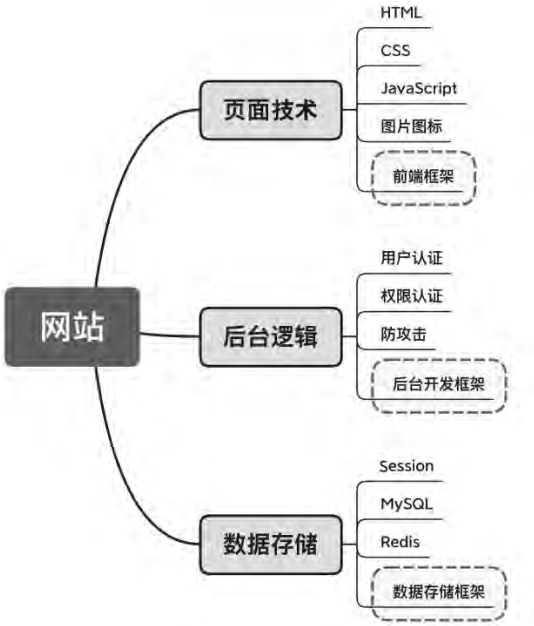

图 1-3

1.3　ASP.NET 概述

编写静态页面需要掌握HTML+CSS。静态页面的最大的优点是速度快，可以跨平台，跨服务器。早期的网站建设大多采用静态页面，静态页面的网址以.htm或.html为后缀。在这种静态网站上也可以使用动态效果，例如滚动字幕、GIF格式的动画或者FLASH。这些视觉上的动态效果并不是动态页面，它们是截然不同的概念。所谓动态页面，就是该网页不仅具有HTML标记，而且含有服务器端的脚本程序代码，能实现操作数据库、交互等功能。动态网页能根据不同的时间、不同的来访者显示不同的内容，而且动态网站更新方便，一般在后台直接更新，并不需要人工手动修改代码。

编写动态页面的主要技术有ASP.NET、JSP、PHP等，本书主要讲解ASP.NET技术。在本节中，我们初步认识ASP.NET，了解它的概念和优势。

1.3.1　ASP.NET 的概念

ASP.NET是微软公司整个.NET FrameWork的一部分，使用它可以创建动态交互的Web页面。其中，ASP的全称是Active Server Pages（动态服务器页面），是一种使嵌入在网页中的服务器脚本由服务器来执行的技术。

ASP.NET、.NET FrameWork及对应的集成开发环境 Visual Studio 一直以来都在不断地更新，这些更新包括.NET框架类库的扩充、纳入新的语言特性等。.NET FrameWork 8.0的出现标志着.NET Frame Work真正走向成熟，同时也说明了ASP.NET技术的成熟与稳定。为了支持ASP.NET的开发，Visual Studio 也在不断地升级版本，目前应用最广的版本是VS2019。

ASP.NET已经发展了多年，目前最新的版本是4.x。ASP.NET 4.x是一个成熟的框架，提供在Windows上生成基于服务器的企业级Web应用所需的服务。注意，ASP.NET只能部署在Windows系统上。

1.3.2　ASP.NET 的优势

作为微软公司.NET FrameWork的一部分，ASP.NET技术延续了Microsoft的一贯优势，即有开发效率高、功能强大的IDE（Integrated Development Environment，集成开发环境）设计工具的支持。除了这些，ASP.NET 还具备以下优势。

1）与浏览器无关

无论使用何种版本的浏览器访问ASP.NET应用程序，呈现的结果都一致。ASP.NET遵循W3C标准化组织推荐的XHTML标准生成页面代码，而XHTML标准被目前所有主流浏览器支持。

2）编译后执行，运行效率高

代码编译是指将C#或VB.NET源码“翻译”成机器语言。ASP.NET先把源码编译为微软中间语言（MSIL），然后由即时编译器（JIT）进一步编译成机器语言。编译好的代码再次运行时不需要重新编译，极大地提高了Web应用程序的运行效率。整个过程如图1-4所示。

图 1-4

3）易于部署

将必要的文件复制到Web服务器上，ASP.NET应用程序即可将其部署到该服务器上。不需要重新启动服务器，甚至在替换运行的编译代码时也不需要重新启动Web服务器。

4）丰富的可用资源

ASP.NET可利用整个.NET平台的资源，包括.NET框架类库和数据访问解决方案等。ASP.NET本身提供了大量的控件，包括与传统HTML代码对应的HTML控件和重新封装的Web控件。

5）支持多层开发

ASP.NET支持多层开发，从而改变了原来Web项目开发代码混乱且难以管理的状况，使得Web项目开发逻辑更清晰，管理维护更方便。

6）逻辑代码和设计代码分离

ASP.NET将逻辑代码放于单独的文件中，将Web界面元素和程序逻辑分开显示，使得代码结构更加清晰，从而方便维护和阅读。

1.3.3　ASP.NET 的主流开发方式

ASP.NET是一个开发框架，用于通过HTML、CSS、JavaScript以及服务器脚本来构建网页和网站。ASP.NET支持3种开发模式：Web Pages、MVC（Model View Controller）以及WebForm。

在.NET Framework 3.5 SP1发布前，ASP.NET WebForm一直是微软官方提供的唯一的ASP.NET开发框架。在.NET Framework 3.5 SP1中，微软提供了另一种ASP.NET的开发框架，即ASP.NET MVC。

ASP.NET WebForm是微软的开发团队为开发者设计的一个在可视化设计器中拖放控件、编写代码响应事件的快速开发环境。在WebForm中，微软将ASP.NET的开发模型与WinForm统一起来，提供了类似于WinForm的控件、事件驱动模型，使ASP.NET应用程序的开发体验与WinForm应用程序高度一致。WebForm设计模式也称为事件驱动模式，此种开发方式有一个特殊的事件响应模型，即拖拉一个服务器控件到设计器中并双击，就会为该控件添加事件代码，使用起来非常方便。如果你是一个新手，你可能会很感激微软，毕竟它为你节省了很多的工作。但方便的同时就是很多底层代码不透明，我们并不知道底层的原理。作为Web应用开发，请求、处理、响应是基本的流程，可是用此种开发方式开发的网站，用户请求的都是页面，处理的也是页面的后台代码，响应的还是页面，一切就是以UI为核心。笔者感觉此种开发方式虽然有点背离了Web开发，但也是一种创新和一种选择。

由于WebForm方式出来比较早，因此不少现有项目都是基于该方式开发的，而MVC更现代化，是新项目采用的主要开发方式。

1.4　ASP.NET Core 概述

ASP.NET必须部署在基于Windows系统的Web服务器上，无法跨平台，这个特点导致方便跨平台的Java系技术非常“嚣张”。为此，微软推出了ASP.NET Core。

ASP.NET Core是一个跨平台的开源框架，用于在Windows、macOS或Linux上生成基于云的新式Web应用。ASP.NET Core是ASP.NET 4.x的重新设计。

1.4.1　ASP.NET Core 的优点

ASP.NET Core具有如下优点：

- 生成Web UI和Web API的统一场景。
- 针对可测试性进行构建。
- Razor Pages可以使基于页面的编码方式更简单高效。
- Blazor允许在浏览器中使用C#和JavaScript。共享全部使用.NET编写的服务器端和客户端应用逻辑。
- 能够在Windows、macOS和Linux上进行开发和运行。
- 开放源码和以社区为中心，这对于注重安全的用户非常重要。

- 新式客户端框架和开发工作流。
- 支持使用gRPC托管远程过程调用（RPC）。
- 基于环境的云就绪配置系统。
- 内置依赖项注入。
- 轻型的高性能模块化HTTP请求管道。
- 能够托管和部署于以下各项：Kestrel、IIS、HTTP.sys、Nginx、Apache、Docker。
- 并行版本控制。
- 简化新式Web开发的工具。

ASP.NET Core的优点多多，这也是.NET开发广受欢迎的主要原因。

1.4.2 ASP.NET Core 和 ASP.NET 4.x 的比较

我们可以将ASP.NET Core和ASP.NET做个对比，如表1-1所示。

表 1-1 ASP.NET Core 和 ASP.NET 的比较

	ASP.NET Core	ASP.NET 4.x
平台	针对 Windows、macOS 或 Linux 进行生成	针对 Windows 进行生成
开发语言	C#、VB、F#	C#、VB、F#
开发环境	通过 Visual Studio、Visual Studio for Mac 或 Visual Studio Code 进行开发	通过 Visual Studio 进行开发
性能	比 ASP.NET 4.x 的性能更高	良好的性能
运行时	使用.NET Core 运行时	使用.NET Framework 运行时

经过对比，我们发现对于开发者而言，ASP.NET Core和ASP.NET所使用的主流语言是相同的，比如都是C#，而运行时对于开发者而言是感觉不到的。所以学会了ASP.NET 4.X，其实也就基本学会了ASP.NET Core，而且ASP.NET Core由于要支持跨平台，需要让代码在不同的平台上去验证和测试，因此相对初学者而言会更复杂些。

另外，如果项目明确要在Windows上开发，建议用ASP.NET，这样可以得到更多的Windows特性，而这是ASP.NET Core不具备的，因为它需要照顾跨平台通用性。

还有一点，不少跨平台的项目是基于成本考虑的，毕竟非Windows服务器基本免费，当项目的预算比较紧张时，开发环境的选择就更倾向于免费。微软提供的Visual Studio Code是一款免费开发环境，而Visual Studio则费用昂贵。

以上几点都是项目主管需要考虑的事情，初学者不用理会这个，来者不拒，统统学会！Visual Studio要会用，Visual Studio Code也要会用。在Windows上会开发部署ASP.NET，在其他平台上也要会开发和部署ASP.NET Core。唯有如此，才能走遍天下都不怕。

1.5 C#语言概述

C#是微软公司发布的一种程序设计语言，专门为.NET平台设计。它是由C和C++衍生出来的、面向对象的高级程序设计语言。由于其简单易于掌握，加之语法简洁、精确、类型安全，与Web

紧密结合，健全的错误处理机制以及版本处理技术，因此成为.NET开发的首选语言。无论是复杂的商业需求，还是系统级的应用程序，通过使用C#语言，都可以方便地转换为XML网络服务，从而使它们可以通过webservice在任何平台被任何一种语言调用。C#最大的优势是能使程序员高效地开发程序，而绝不损失C/C++原有的强大的功能。C#不再提供对指针类型的支持，从而限制了程序在运行过程中对内存地址空间的访问，使得程序更加安全、稳定。加之.NET核心为C#提供的强大的、逻辑结构一致且易用的程序设计环境，可以让程序编写人员快速、简单地编写各种基于.NET平台的手机及Web应用项目。正是由于C#面向对象的优越性，越来越多的Windows程序更偏向于使用它进行开发。

1.6　.NET Framework 框架

.NET Framework（.NET框架）是一个多语言组件开发和执行环境，是以通用语言运行库（Common Language Runtime，CLR）为基础采用系统虚拟机运行的编程平台。它提供了一个跨语言的统一编程环境，支持多种语言（C#、C++、VB、Python等）的开发。其目的是简化开发人员建立Web应用程序和Web服务的流程，使得Internet上的各应用程序可以使用Web服务进行沟通。.NET Framework是运行在Windows系列操作系统上的一个系统应用程序。它是.NET的核心部分，提供了建立和运行.NET应用程序所需要的编辑、编译等核心服务。它包括两个重要组成部分：公共语言运行时（Common Language Runtime，CLR）和.NET Framework类库（Framework Class Library，FCL）。.NET框架如图1-5所示。

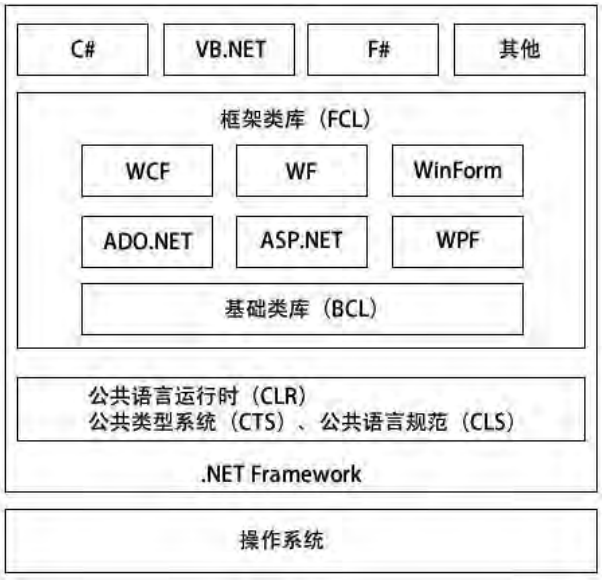

图 1-5

公共语言运行时（CLR）本质上就是.NET 虚拟机（类似Java的虚拟机JVM），算是.NET的引擎，用来执行托管.NET代码，确切地说是编译后的IL代码。提供管理内存、线程执行、代码执行、代码安全验证、异常处理、编译、垃圾回收等运行时服务。

框架类库（FCL）就是.NET Framework内置的各种组件服务，如ASP.NET、WCF和WPF等组件，以满足不同编程应用场景的需求。基础类库（Base Class Library，BCL）是FCL的一个子集，顾名思义就是一些比较基础、通用的类库，如基本数据类型、集合、线程、安全、字符串操作、网络操作、IO、XML操作等。基础类库大多包含在System命名空间下，如System.Text、System.IO。其他一些常用的名词，如核心 .NET库、框架库、运行时库、共享框架，大多指的是BCL。

1.7 HTTP 与 HTML

HTTP（Hyper Text Transfer Protocol，超文本传输协议）是一种简单的请求—响应协议，通常运行在TCP之上。它指定了客户端发送给服务器什么样的消息，以及得到什么样的响应。请求和响应消息的头采用ASCII码的形式，而消息内容则采用类似MIME的格式。

HTML为超文本标记语言，是一种标识性语言。它包括一系列标签，通过这些标签可以将网络上的文档格式统一起来，使分散的互联网资源成为一个逻辑整体。

1.7.1 TCP/IP 通信传输流

假如客户端在应用层（HTTP）发起一个想看某个Web页面的HTTP请求，那么传输层（TCP）会把从应用层收到的数据（HTTP请求报文）进行分割，并在各个报文上打上标记序号及端口号，形成网络层传输的数据包，再在添加通信目的地的MAC地址后作为链路层的数据包，这样发往服务端的网络通信请求就准备齐全了。接收端的服务器在链路层接收到数据包后，按层解包按序往上层发送，一直到应用层。只有当数据包成功抵达应用层时，才标志着服务器真正收到从客户端发送过来的HTTP请求。

1.7.2 HTTP

HTTP是Web应用中客户端和服务器之间进行交互的协议规范，完成客户端向服务端发起请求，服务端向客户端返回请求响应或请求处理结果的一系列过程，如图1-6所示。

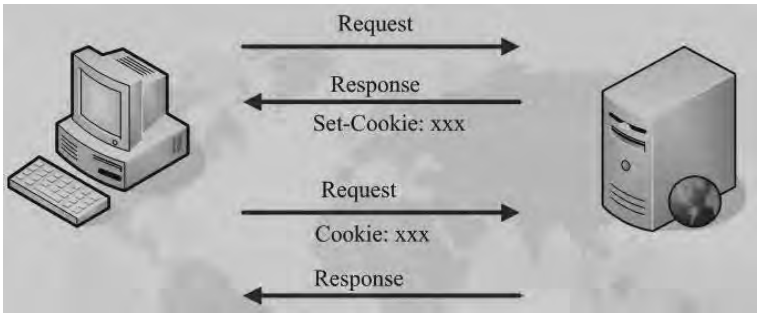

图 1-6

在HTTP交互过程中，客户端通过URI（Uniform Resource Identifier，统一资源标识符）查找并定位网络中的资源。URI通常由3部分组成：①资源的命名机制；②存放资源的主机名；③资源自身的名称。注意：这只是一般URI资源的命名方式，事实上只要是可以唯一标识资源的都被称为URI，

上面3部分合在一起是URI的充分不必要条件。URI包含协议名、登录认证信息、服务器地址、服务器端口号、带层次的文件路径、查询字符串、片段标志符。URI的表达方式如下：

```
http://user:pass@www.example.com:80/home/index.html?age=11#mask
```

其中，http为协议方案名；user:pass表示登录认证信息；www.example.com是服务器地址；80为端口号；/home/index.html表示路径和所要请求的网页文件；age=11表示查询字符串；mask表示片段标识符。

协议方案名在获取资源时要指定协议类型，包括http、https、ftp等。登录（认证）信息指定用户名和密码作为从服务器端获取资源时必要的登录信息，此项是可选的。服务器地址使用绝对URI来指定待访问的服务器地址。服务器端口号就是指定服务器连接的网络端口号，即Web服务侦听服务器的TCP端口号，此项是可选的。路径和文件名用于定位服务器上的特定资源，比如/home/ index.html。查询字符串用于定位指定文件内的资源，可以使用查询字符串传入任意参数，此项是可选的。片段标识符通常可以标记出来，以获取资源中的子资源（文档内的某一个位置），此项也是可选的。

HTTP分为请求报文和响应报文，报文由报文首部和报文主体构成。请求报文首部包含请求行、请求头部、通用头部、实体头部。响应报文首部包含状态行、响应头部、通用头部、实体头部。

- 请求行：包含客户端请求服务器资源的操作或方法（get、post、put、delete）、请求URI、HTTP的版本。
- 状态行：包含HTTP的版本、服务端响应结果编码、服务端响应结果描述。状态行为“HTTP/1/1 200 OK”代表处理成功。响应分为5种：信息响应（100~199）、成功响应（200~299）、重定向（300~399）、客户端错误（400~499）、服务器错误（500~599）。比如，200表示请求成功，请求方法为get、post、head或者trace；404表示请求失败，请求资源找不到，类似于脚本未被定义；507表示服务器有内部配置错误。
- HTTP通用头部如表1-2所示。

表 1-2　HTTP 通用头部及其说明

头部编码	头部说明
Cache-Control	控制缓存
Connection	连接管理
Transfer-Encoding	报文主体编码方式
Date	报文创建的日期时间
Upgrade	升级为其他协议
Via	代理服务器相关信息
Warning	错误通知

其他头部这里不再赘述，读者可以参考HTTP文档。使用HTTP要注意以下几点：

1）通过请求和响应的交换达成通信

应用HTTP时，必定是一端担任客户端角色，另一端担任服务器端角色。仅从一条通信线路来说，服务器端和客户端的角色是确定的。HTTP规定，请求从客户端发出，最后服务器端响应该请求并返回。换句话说，肯定是从客户端开始建立通信的，服务器端在没有接收到请求之前不会发送响应。

2）HTTP 是不保存状态的协议

HTTP是一种无状态协议。协议自身不对请求和响应之间的通信状态进行保存。也就是说，在HTTP这个级别，协议对于发送过的请求或响应都不做持久化处理。这是为了更快地处理大量事务，确保协议的可伸缩性，而特意把HTTP设计得如此简单。

随着Web技术的不断发展，很多业务都需要保存通信状态，于是引入了Cookie技术。有了Cookie，再用HTTP通信，就可以管理状态了。

3）使用 Cookie 的状态管理

Cookie技术通过在请求和响应报文中写入Cookie信息来控制客户端的状态。Cookie会根据从服务器端发送的响应报文内的一个名为Set-Cookie的首部字段信息，通知客户端保存Cookie。当下次客户端再往该服务器发送请求时，客户端会自动在请求报文中加入Cookie值后发送出去。服务器端发现客户端发送过来的Cookie后，会检查究竟是从哪一个客户端发来的连接请求，然后对比服务器上的记录，而后得到之前的状态信息。

4）请求 URI 定位资源

HTTP使用URI定位互联网上的资源。正是因为URI的特定功能，HTTP请求能访问到互联网上任意位置的资源。

5）持久连接

在HTTP的初始版本中，每进行一个HTTP通信都要断开一次TCP连接。比如，在使用浏览器浏览一个包含多幅图片的HTML页面时，在发送请求访问HTML页面资源的同时，也会请求该HTML页面中包含的其他资源。因此，每次的请求都会造成TCP连接的建立和断开，无谓地增加了通信量的开销。

为了解决上述TCP连接的问题，HTTP 1.1和部分HTTP 1.0想出了持久连接的方法。其特点是，只要任意一端没有明确提出断开连接，则保持TCP连接状态，旨在建立一次TCP连接后进行多次请求和响应的交互。在HTTP 1.1中，所有的连接默认是持久连接。

6）管线化

持久连接使得多数请求以管线化方式发送成为可能。以前发送请求后需等待并接收到响应，才能发送下一个请求。管线化技术出现后，不用等待收到前一个请求的响应也可以发送下一个请求。这样就能以并发方式发送多个请求。

比如，当请求一个包含多幅图片的HTML页面时，与挨个建立连接相比，用持久连接可以让请求一响应更快结束。而管线化技术要比持久连接的速度快，请求数越多，越能彰显每个请求所用时间短的优势。

1.7.3　HTML

HTML（Hyper Text Markup Language，超文本标记语言）是表达Web网站内容的一种语言。HTML除了用于表示文本内容，还用于描述网页的样式（如颜色、字体等），支持在网页中包含链接、图片、音乐、视频、程序。通过HTML语言描述的树状结构的文档为HTML文档，在该文档中通过标签树来表达网页的结构及各种元素。

1.8　框　　架

框架（Framework）是整个或部分系统的可重用设计，表现为一组抽象构件及构件实例间交互的方法。另一种定义认为，框架是为应用开发者定制的应用骨架或开发模板，一个框架是一个可复用的设计构件，它规定了应用的体系结构，阐明了整个设计、协作构件之间的依赖关系、责任分配和控制流程。前端开发框架和后端开发框架是基于前端开发和后端开发两种不同的开发方式来区分的。

1.8.1　为什么要使用框架

软件系统发展到今天已经很复杂了，特别是服务器端软件，涉及的知识和问题太多。在某些方面使用别人成熟的框架，就相当于让别人帮我们完成一些基础工作，而我们只需要集中精力完成系统的业务逻辑设计。框架一般是成熟、稳健的，它可以处理系统的很多细节问题，比如事物处理、安全性、数据流控制等问题。另外，框架一般经过很多人使用，所以结构很好，扩展性也很好，而且它是不断升级的，我们可以直接享受别人升级代码带来的好处。

1.8.2　Web 框架基础技术

很多做Web开发的人（本文表示读者）都有一个梦想，就是将来能开发一套可以在项目中使用的Web应用框架。刚开始做开发时，觉得这个技术好厉害，只需要写一点代码，就能够做出带有业务逻辑的网站。其实，写一个Web应用框架并不难，但是写出一个能够经受工业强度测试的软件就不容易了。

Web框架的全称为“Web应用框架”（Web Application Framework），一般负责如下几个方面的工作：MVC分层、URL过滤与分发、View渲染、HTTP参数预处理、安全控制，还有部分附加功能，如页面缓存、数据库连接管理、ORM映射等。Web框架可以用任何语言写成，几乎常见的计算机语言都有对应的Web框架，可以用来写Web程序。不要以为只有PHP、Java、C#才能做网站。

框架重要的特征是MVC分层。MVC这个概念并不是Web开发中才有的，也不是从这个方向发展出来的技术。在20世纪70年代的Smalltalk-76中就引入了这个概念。

1.8.3　分清框架和库

框架是一套架构，会基于自身的特点向用户提供一套相当完整的解决方案，控制权在框架本身，使用者需要按照框架所规定的规范着手开发。

库是一种插件，是一种封装好的特定方法的集合，提供给开发者使用，控制权在使用者手里。

框架提供了一套完整的解决方案，同时前端功能越来越强大，因而产生了前端框架，所以开发Web产品很有必要使用前端框架（前端架构）。目前流行的前端框架有Angular、Vue和React。流行的一些库有jQuery、Zepto等。使用前段框架可以降低界面的开发周期并提高界面的美观性。

1.8.4　Web 开发框架技术

Web开发框架技术，即Web开发过程中可重复使用的技术规范，它可以帮助技术人员快速开发特定的系统。Web开发框架技术分为前端开发框架技术和后端开发框架技术。

前端开发是创建Web页面或App等前端界面并呈现给用户的过程，它通过HTML、CSS及JavaScript以及衍生出来的各种技术、框架、解决方案来实现互联网产品的用户界面交互。前端框架技术的应用使前端开发变得方便快捷。目前，Web前端开发框架有Vue、Angular、Bootstrap、React等。

后端开发负责程序设计架构、数据库管理和处理相关的业务逻辑，主要考虑功能的实现以及数据的操作和信息的交互等。后端开发对开发团队的技术要求相对较高，借助后端开发框架技术，可以简化后端开发过程，使其变得相对容易。后端框架技术往往和后端功能实现所用的语言有关。目前，流行的Web后端框架技术有Laravel、Spring MVC、Spring Boot、ASP.NET MVC、MyBatis、Phoenix、Django等。

1.9　常见 Web 框架

框架一般用于简化网页设计，使用广泛的前端开发框架有Angular、ASP.NET MVC和React等。这些框架封装了一些功能，比如HTML文档操作、各种漂亮的控件（按钮、表单等），使用前端框架有助于快速建立网站。

随着互联网的快速发展，Web应用不断推陈出新，Web前端技术发挥着举足轻重的作用。如今智能化设备全面普及使得Web前端页面变得越来越复杂，从视觉体验到对用户的友好交互、技术特效等的要求越来越高，系统的维护要求不断提升。前端技术的不断演进也带来了前端开发模式的不断改进，在基于前端开发逐渐复杂的背景下，Web前端框架技术也成了人们关注的焦点。大多数的Web框架都提供了一套开发和部署的方式，实现了数据的交互和业务功能的完善。开发者使用Web框架只需要考虑业务逻辑，因此可以有效地提高开发效率。

在Web发展早期，页面的展示完全由后端PHP、JSP控制。Ajax技术的出现给用户带来了新的体验，前后端通过Ajax接口进行交互，分工逐渐清晰。同时，伴随着JavaScript技术的革新，浏览器端的JavaScript代替了服务器端的JSP页面，它可以依靠JavaScript处理前端复杂的业务逻辑，但是代码的复杂度仍然很高。因此，为了提升开发效率、简化代码、便于后期维护，在开发中应用分层的架构模型应运而生。

1.9.1　MVC 框架模式

MVC（Model-View-Controller，模型－视图－控制器）框架模式，即为模型（Model）、视图（View）和控制器（Controller）的分层模式。模型层用于处理数据，能够直接针对相关数据进行访问，针对应用程序业务逻辑的相关数据进行封装处理；视图层能够显示网页，由于视图层没有程序逻辑，因此需要对数据模型进行监视和访问；控制层主要体现在对应用程序流程的控制，以及对事件的处理和响应上。控制层获取用户事件信息后，通知模型层进行更新处理，由模型层将处理结果发送给视图层，视图层的相关显示信息随之发生改变，因此控制层对于视图层和模型层的一致性

进行了有效的调节和控制。下面以用户提交表单为例，展示MVC的设计模式。MVC模式示意图如图1-7所示。

图 1-7

在图1-7中，当用户提交表单时，控制器接收到HTTP请求并向模型发送数据，模型调用数据后将数据返回至控制器，控制器调用视图将处理结果发送至浏览器，浏览器负责网页的渲染。

前端MVC模式中广泛使用的框架为Backbone.js、Ember.js等。Backbone.js的优势在于可以较好地解决系统应用中的层次问题，同时应用层中的视图层在模型数据修改后，可以及时地对自身页面数据进行修改。此外，它还可以通过定位有效地找到事件源头，解决相关问题。Ember.js广泛应用于桌面开发中，借助于该框架的优势，能够实现模块化、标准化的页面设计与分类，保证MVC运行的效率。除此之外，Ember.js框架能够有效地结合大数据系统的优势，将整个运行过程中所产生的各种参数及时有效地记录在档案数据库中。

MVC模式最早应用于桌面应用程序中，随着Web前端的发展，其复杂程度逐渐增加，后被广泛应用于后端开发，实现数据层与表示层的分离。作为早期的框架模式，MVC模式主要的优势在于能够清晰地分离视图和业务逻辑，满足不同用户的访问需求，在一定程度上降低了设计大型Web应用的难度。但是，由于内部原理较为复杂，并且定义不够明确，因此开发者需要明确前端MVC框架的使用范围，并且需要耗费大量的时间和精力解决如何将MVC模式运用到应用程序的问题。另外，MVC严格的分离模式也导致每个构件均需经过完整的测试才能使用，使得在相当长的一段时间内，MVC模式不适用于中小型项目。随着技术的发展，部分框架能够直接对MVC提供支持，但在实现多用户界面的大型Web应用上，开发者仍需要花费大量时间，不利于开发效率的提升。

1.9.2　MVP 框架模式

MVP（Model-View-Presenter，模型－视图－表示器）框架模式是由IBM公司于2000年开发的一种模式，是MVC模式的改进，主要用来隔离UI和业务逻辑，旨在使Web应用程序分层和提高测试效率。

MVP模式和MVC模式都具有相同的分层架构设计，均由视图进行显示，由模型管理数据。它们的区别是，在MVP中，视图和模型之间的通信是通过Presenter进行的，所有的交互都发生在Presenter内部；而在MVC中，View直接从Model中读取数据，而不是通过Controller。由于MVP模式中的View和Model层之间没有关系，因此可以将View层抽离为组件，在复用性上比MVC模型具有优势。

作为MVC模式的演变，MVP模式主要是为了解决MVC模式中View对Model的依赖。MVP模式的优点在于模型和视图完全分离，开发者可以只修改视图而不影响模型，并且可以更高效地使用模型。同时，由于所有的交互都在Presenter内部完成，因此可以更高效地应用模型，另外可以脱离用户接口测试业务逻辑。其劣势在于View和Presenter的接口使用量较大，使得View和Presenter的交互过于频繁。在用户界面较为复杂的情况下，一旦View发生改变，View和Presenter之间的接口必然发生变更，导致接口群的需求量增加。因此，MVP模型适用于开发后期需要不断维护且较大型的项目。

1.9.3　MVVM 框架模式

MVVM（Model-View-ViewModel，模型－视图－视图模型）框架模式的结构如图1-8所示。

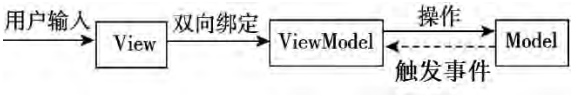

图 1-8

MVVM模式的出现，是为了解决MVP模式中由于UI种类变化频繁而导致接口不断增加的问题。其设计思想是“数据驱动界面”，以数据为核心，使视图处于从属地位。该模式只需要声明视图和模型的对应关系，数据绑定由视图模型完成，相当于MVC模式的控制器，实现了视图和模型之间的自动同步。

MVVM模式简化了MVC和MVP模式，不仅解决了MVC和MVP模式中存在的数据频繁更新的问题，同时降低了界面与业务之间的依赖程度。在该模式中，视图模型、模型和视图彼此独立，视图察觉不到模型的存在。这种设计模式具有以下优势：

- 低耦合。View可以不随Model的变化而修改，一个ViewModel可以绑定到不同的View上，当View变化时，Model可以不变；当Model变化时，View也可以不变。
- 可重用性。将视图逻辑放在ViewModel，View将重用视图逻辑。
- 独立开发。开发人员可以专注于业务逻辑和数据的开发,设计人员可以专注于界面的设计。
- 可测试性。可以针对ViewModel对View进行测试。

MVVM框架模式是MVC精心优化后的结果，适合编写大型Web应用。在开发层面，由于View与ViewModel之间的低耦合关系，使得开发团队分工明确且相互之间不受影响，从而提升了开发效率；在架构层面，由于模块间的低耦合关系，使得模块间的相互依赖性降低，项目架构更稳定，扩展性更强；在代码层面，通过合理地规划封装，可以提高代码的重用性，使整个逻辑结构更为简洁。

MVVM模式中应用较为广泛的框架有AngularJS、React、Vue.js等，我们重点对MVVM模式的主流框架进行分析。

目前，优秀的前端开发框架很多，在选择上建议：①与需求相匹配的框架；②与浏览器兼容性好的框架；③组件丰富、支持插件的框架；④文档丰富、社区大的框架；⑤高效的框架。

1.9.4　Web 框架的发展现状

早期的Web前端主要包含HTML、CSS与JavaScript三大部分，其中HTML主要负责页面结构，

CSS主要负责页面样式，JavaScript主要控制页面行为和用户交互。前端仅限于网页的设计，大部分功能需要依赖后端实现。随着Web应用的迅速发展，前端的功能性越来越强，开发难度逐渐增大。一大批优秀前端框架的出现推动了前端技术的发展，降低了开发成本，提升了开发效率。起初的JavaScript框架jQuery凭借便捷的DOM操作、支持组件选择、内部封装Ajax操作等特点占据着主导地位。随着前端的进一步发展，利用jQuery开发Web应用无法分离出业务逻辑、交互逻辑和UI设计，增加了代码的维护难度。而MVVM设计模式的出现实现了数据和视图的自动绑定，它将DOM操作从业务代码中剥离，提高了代码的可维护性和复用性。

国外前端开发起步早于国内，涌现了较多的高水平Web框架，并且能够较好地支持移动端。目前，国内知名互联网公司致力于开发高水平的开源Web前端框架，总体水平已经达到了较高的程度。百度前端团队开发的QWrap突破了jQuery的局限，提供了原型功能，为广大用户带来了便利；腾讯非侵入式的JX前端框架实现了JavaScript的扩展工具套件，于2012年切换到GitHub，具有较优的执行效率，无过度的封装，并且努力探索如何在前端使用MVP、MVC等模式构建大型Web应用；淘宝内部使用的Web框架KISSY是一款跨终端、模块化、高性能、使用简单的JavaScript框架，具备较为完整的工具集以及面向对象、动态加载、性能优化的解决方案，为移动端的适配和优化作出了巨大贡献。

在互联网快速发展的今天，前端框架被广泛应用。为了适应网站的大量需求，加快开发网站的效率，大型互联网厂商纷纷构建满足各自业务的前端框架，如Element UI和Ant Design分别是饿了么和阿里巴巴自研的前端UI组件库。工具的发展和前端的发展相辅相成，JavaScript的每次进步都会带动浏览器厂商和相关开发工具的进步，同时也为浏览器的兼容性提出了更合理的解决办法。

前端框架技术的发展日趋成熟，未来前端在已经趋向成熟的技术方向上会慢慢稳定下来，进入技术迭代优化阶段，新的Web思想也会给前端带来新的技术革新和发展机遇。

第 2 章

搭建 Web 开发环境

当前ASP.NET底层支持通常依赖两种底层框架技术，一种是.NET Framework，另外一种是.NET技术（版本5以前叫.NET Core）。.NET Framework只能运行在Windows系统上，功能比较完整，并且拥有Windows系统的特殊功能。而.NET技术可以跨平台，功能相当于.NET Framework的一个子集，更注重平台通用性。这两种底层框架技术各有千秋，各有用途。对于初学者而言，不建议一上来就进入跨平台开发，因为跨平台开发肯定要面临多平台的调试和运行，环境搭建比较复杂。因此可以先在Windows上搭建环境（这个相对简单），把精力集中在.NET Framework的学习上。当学习了.NET Framework后，.NET Core也就基本掌握了（因为它是.NET Framework的一个子集），然后可以把精力放在各平台的环境搭建和程序调试上。

这样学习后，无论以后针对Windows深度应用，还是跨平台应用，都能对付。

另外，我们把基于.NET Framework的ASP.NET学习放在Visual Studio工具中实现，以后把跨平台开发放在VSCode中实现，这样就学会了两种开发环境，能轻松应对不同的企业要求。总而言之，笔者是根据企业一线开发的实际情况来撰写本书的。

在本章中，我们将搭建基于.NET Framework的Visual Studio的开发环境，这个环境比较简单，适合初学者。后续章节还将搭建基于.NET的VSCode跨平台开发环境。

2.1 下载和安装 Visual Studio

当前企业界主流的Visual Studio（VS）开发环境是Visual Studio 2019，该开发环境适用面广、稳定、速度快且对计算机配置友好。建议读者不要盲目追求最新的开发工具，因为企业很多已有项目的维护都是基于旧开发工具开发的，而且最新的开发环境相关的资料比较少，一旦碰到问题，答案都找不到。我们可以到官网可以下载Visual Studio 2019，网址如下：

```
https://learn.microsoft.com/en-us/visualstudio/releases/2019/
```

然后可以找到3个版本，包括社区版（Community）、专业版（Professional）和企业版（Enterprise），如图2-1所示。

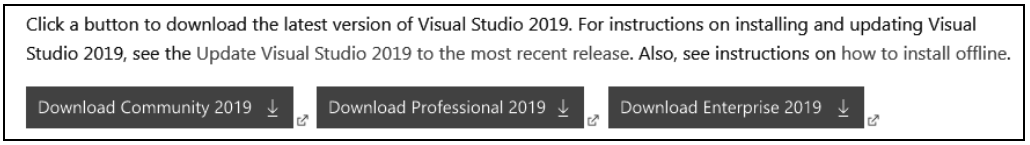

图 2-1

通常，对于学习者而言，下载社区版就够用了，而且社区版对于个人学习来说是免费的。后两者都是针对企业，需要收费。我们单击“Download Community 2019”按钮就可以下载安装包文件，下载下来的文件名是vs_community__77008ca6f8834edcb53ab2ee09aa4fb4.exe，这是个在线安装包文件，也就是说安装过程需要联网。如果不想下载，读者也可以在本书配套源码目录下的somesofts文件夹下找到该文件。

直接双击这个安装包文件就可以开始安装了。注意，如果使用的是Windows 7操作系统，可能会跳出如图2-2所示的对话框，意思是要先安装.NET Framework 4.6或更高版本。这个组件可以去微软官方网站上找，为了节省时间，笔者提供了一个.NET Framework 4.8的离线安装包文件，可以在本书配套源码目录的somesosfts\.NET Framework 4.8\下找到文件ndp48-x86-x64-allos-enu.exe，直接双击它就可以开始安装了，安装完成如图2-3所示。

图 2-2

图 2-3

等 .NET Framework 安装完毕后，再重新双击运行 vs_community__77008ca6f8834edcb53ab2ee09aa4fb4.exe，图2-4所示是安装过程中的第一个界面。

图 2-4

单击“继续”按钮，将进入“工作负荷”对话框，选择需要的组件。因为我们的开发基于.NET Framework 4.8的ASP.NET，因此勾选“ASP.NET和Web开发”和“.NET Framework 4.8开发工具”复选框，如图2-5所示。

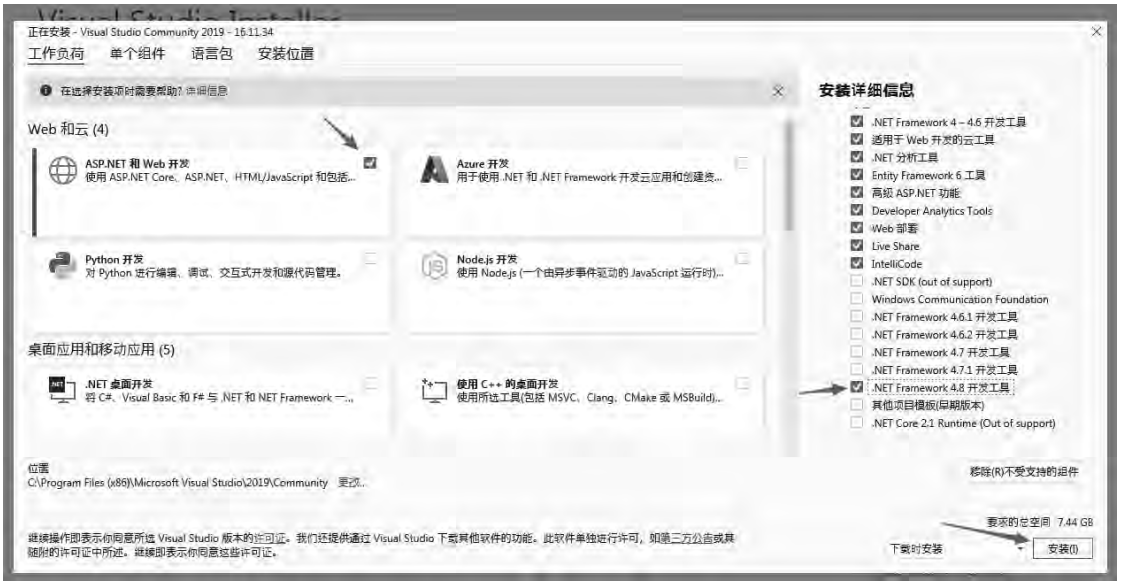

图 2-5

再打开“单个组件”选项卡，准备安装localDB数据库：在搜索框中输入local，然后按回车键，勾选要下载的“SQL Server Express 2016 LocalDB”，并勾先“SQL Server支持的数据源”，如图2-6所示。

图 2-6

案装了“SQL Server支持的数据源”后，以后Visual Studio的“视图”菜单下就有“SQL Server资源管理器”菜单项了。如果只勾选“SQL Server Express 2016 LocalDB”，是不会出现“SQL Server资源管理器”菜单项的。SQL Server是微软公司的数据库产品的旗舰产品，而SQL Server Express 2016 LocalDB（简称LocalDB）是SQL Server的精简版，为便于开发人员开发和测试而提供的。

其他保持默认即可。然后单击右下角的“安装”按钮开始安装。稍等片刻，安装完成，就能直接启动Visual Studio了。第一次启动时会让用户选择不同颜色的界面，这个根据自己的喜好选择，随后进入“开始使用”界面，如图2-7所示。

图 2-7

该界面左边“打开最近使用的内容”中将显示最近已经建立的项目，目前我们还没新建项目，因此下面为空。右边有4个选项，其中“克隆存储库”用于从GitHub等版本管理系统中复制代码；“打开项目或解决方案”用于打开本地硬盘上的Visual Studio项目或解决方案（sln）；“打开本地文件夹”可以用来查看代码，这个功能比较新，而且比较实用，此时Visual Studio就相当于一款功能强大、面向项目的编程编辑器、代码浏览器和分析器，可帮助我们在学习和开发时理解代码；最后一个选项“创建新项目”则是Visual Studio的主要功能入口，里面有不同的编程模板可供选择，开发新项目都是从这里进去。

2.2　第一个 ASP.NET 项目

现在我们趁热打铁，马上新建并运行一个ASP.NET项目，以此来验证Visual Studio是否工作正常。

【例2.1】第一个ASP.NET项目

首先打开Visual Studio，单击“创建新项目”，此时出现“创建新项目”对话框，我们在该对话框上选择C#版本的“ASP.NET Web应用程序（.NET Framework）”，如图2-8所示。

图 2-8

然后单击“下一步”按钮，此时出现“配置新项目”对话框，如图2-9所示。在这个对话框中，我们需要输入项目名称，比如helloworld；确定项目位置，这里在“位置”下面输入D:\ex\myweb，这个路径即使不存在，也会自动建立。另外，框架选择“.NET Framework 4.8”。接着单击右下角的“创建”按钮，此时出现“创建新的ASP.NET Web应用程序”对话框，如图2-10所示。

图 2-9

图 2-10

有关项目的创建过程，笔者在第一个范例中说得详细一点，后面的范例为了节省篇幅，一般就直接说“新建一个基于MVC的项目”了，默认项目名称是test，保存在D:\ex\myweb\下，并且每完成一个范例就清空该目录。因此，在开始新建项目时，就默认myweb目录下是空的。

微软为ASP.NET提供了4种开发模式，分别为Web Forms（Web窗体）、MVC（Model View Controller，模型－视图－控制器）、Web API和Web Pages（单页应用程序）。这4个框架都稳定且成熟，我们可以使用其中任何一个框架创建出色的Web应用程序。无论选择哪种框架，我们都能随时随地获得ASP.NET的所有优势和功能。

Web Forms又称Web窗体，在这个模式下，可以采用传统的拖曳事件驱动模型的方式来生成动态网站，同时利用设计图以及许多控件和组件，可以迅速生成带有数据访问的、高级的、功能强大的UI驱动型网站。但需注意，Web窗体开发模式现在已经过时，学习价值不大。

MVC模式提供了功能强大、基于模式的方法来构建完全分离关注点的动态网站。基于MVC模式的网站有着耦合度低、重用性高、部署快等优点。MVC模式是使用ASP.NET框架开发复杂网站的首选开发模式。

Web API是一个用于构建RESTful Web服务的框架，基于HTTP协议，提供灵活的路由、数据格式序列化和状态管理，能快速开发公开、可靠、高性能的API接口。通过Web API，开发者可以轻松地处理和返回数据，并与客户端通过HTTP完成通信。在现代Web开发中，构建高性能、可扩展的Web服务至关重要。ASP.NET Web API是一个强大的框架，它提供了构建RESTful Web服务的工具和功能。无论是为移动应用程序提供后端支持，还是与第三方平台集成，Web API 都能够满足各种需求，并帮助开发者轻松地构建功能丰富的应用程序。

Web Pages即单页应用程序，指的是只有一个Web页面的应用。在这个模式中，可以使用C#（或Visual Basic）结合网页的Razor标记语法将C#（Visual Basic）代码嵌入网页当中，实现C#（Visual Basic）代码和HTML、CSS、JavaScript等服务器代码的结合。值得一提的是，单页应用程序的开发模式是ASP.NET框架3种开发模式中最简单的一种，如果你是个新手，那么Web Pages单页应用程序开发模式是你入门的不错选择。

这几种技术都有优缺点和各自不同的应用，这里不做详述。由于目前最流行的开发方式是基于MVC的，因此在图2-10中选择“MVC”，然后单击右下角的“创建”按钮。稍等片刻，一个ASP.NET项目框架就自动建立起来了，并且可以在解决方案视图下看到已经生成的文件，如图2-11所示。

图 2-11

Controllers文件夹存放的是控制器相关的代码文件，用于处理来自用户、整个应用程序流以及特定应用程序逻辑的通信；Models文件夹存放的是模型相关的代码文件，用于描述要处理的数据以及修改和操作数据的业务规则；Views文件夹存放的是视图相关的代码文件，用于定义应用程序用户界面的显示方式。这样组合起来就构成了MVC架构。至于其他文件都是一些辅助功能，比如程序启动文件、程序配置文件以及路由配置等。下面对一些文件夹或文件进行解释：

- App_Data：用于存储应用程序的数据文件，例如数据库文件或其他本地文件。
- App_Start：包含应用程序的启动配置文件，例如路由配置、日志配置等。
- bn：包含应用程序的二进制文件，例如编译后的DLL文件、第三方库等。
- Content：存放应用程序的静态资源文件，如CSS、JavaScript、图像等。
- Controllers：包含控制器类，用于处理请求并生成响应。
- Models：存放应用程序的模型类，用于表示数据结构、业务逻辑等。
- Views：用于存储与应用程序的显示相关的HTML文件（用户界面）等。Views文件夹下面有Home和Shared两个子文件夹：Home文件夹用于存储诸如home页（首页）和about页（关于页）之类的应用程序页面；Shared文件夹用于存储控制器间分享的视图（母版页和布局页）。
- Scripts：存放应用程序的JavaScript文件（.js文件）。
- Web.config：Web.config是ASP.NET Web应用程序的主要配置文件，使用XML格式。它包含了应用程序的各种配置信息，如应用程序的全局设置、连接字符串、授权和身份验证设置、HTTP模块和处理程序的配置等。
- Global.asax：Global.asax是一个全局的应用程序类文件，用于处理应用程序级别的事件，如应用程序的启动和关闭、会话管理、应用程序错误处理等。它没有特定的后缀名，但通常命名为Global.asax。
- .cs文件：.cs是C#源码文件的后缀名。在ASP.NET中，.cs文件通常用于编写服务器端的逻辑代码，包括控制器、模型、业务逻辑等，用于处理请求、生成动态内容和执行服务器端的操作。
- .cshtml文件：.cshtml是Razor视图文件的后缀名。Razor是一种用于在服务器端生成动态HTML内容的视图引擎。.cshtml文件可以包含HTML标记和Razor语法，用于定义Web页面的结构和呈现逻辑。在.cshtml文件中，可以使用C#代码和Razor语法来动态生成页面内容。我们在Views文件夹下面可以找到很多.cshtml文件。
- .config文件：.config是各种配置文件的通用后缀名，如Web.config和packages.config。这些配置文件用于存储应用程序的配置信息、软件包配置等信息。

另外，在D:\ex\myweb\下可以看到生成了一个helloworld文件夹，这个文件夹就是我们的解决方案文件夹（注意是解决方案文件夹），里面包含解决方案文件、所有的项目文件夹以及软件包文件夹。进入helloworld文件夹，可以看到helloworld.sln文件、helloworld文件夹（这个文件夹是项目文件夹）、软件包文件夹packages。其中helloworld.sln表示解决方案文件，它是一个文本文件（我们不要去编辑它），文件扩展名“.sln”代表“Solution（解决方案）”，helloworld表示解决方案的名称。Visual Studio使用解决方案的概念，这是一个顶层的源码结构，它将多个项目绑定在一个持久的环境中，通过Visual Studio中的“解决方案资源管理器”处理。一个解决方案由基于文本的解决方案（.sln）文件和二进制解决方案用户选项（.suo）文件表示。一个解决方案内可以包含和管理多个项目，以后我们只要双击这个.sln文件，就可以直接打开Visual Studio并加载解决方案及其下面的所有项目了。当然，现在我们刚刚开始学，在helloworld这个解决方案下只放一个项目，这个项目就是helloworld，所以我们能看到文件夹helloworld。一般情况下，在新建项目时，解决方案的名称默认和项目名称相同，当然也可以在Visual Studio中修改。

项目文件夹helloworld里面的内容，就是实际的代码文件和项目配置文件，该文件夹下的内容和Visual Studio中“解决方案资源管理器”视图下面的内容是一一对应的。我们在Visual Studio的“解决方案资源管理器”视图下增加、删除或重命名文件，实际上就是对这个项目文件夹下的文件进行

同样的修改；我们在Visual Studio的编辑器中编辑文件，实际上也是对该项目文件夹下的文件进行同样的修改。

软件包文件夹（简称包文件夹）packages存放项目运行所需的基础软件包，我们一般不需要去改动，Visual Studio会自动进行配置。

说了这么多，现在是时候让程序运行起来了！下面我们来执行程序，按快捷键Ctrl+F5或单击菜单栏中的“调试”→“开始执行（不调试）”命令，将直接打开IE浏览器来显示网页结果，如图2-12所示。

图 2-12

这个页面是项目网站的首页，如果我们单击网页上的“关于”或“联系方式”等菜单，将会显示相应的页面。IE地址栏中的“localhost”表示访问的网站主机位于本地计算机上，因此也可以写成127.0.0.1；“44358”表示Web服务的端口。

如果要停止程序，直接关闭IE浏览器即可。至此，第一个ASP.NET项目运行成功了，我们一行代码都没有写，居然实现了一个MVC架构的ASP.NET网站！通过这个范例，也证实Visual Studio安装成功了。

通过这个范例，我们学会了新建ASP.NET工程，学会了执行程序，也了解了ASP.NET工程的文件组织架构。这里有必要再说明一下，在Visual Studio中，“解决方案”（Solution）和“项目”（Project）是两个重要的概念，它们用于组织和管理软件开发过程中的代码、文件和资源。以下是它们之间的区别：

一个解决方案是一个包含一个或多个项目的容器。它是一个顶层的组织单元，可以包含多个项目、项目文件夹、配置和设置。解决方案提供了一种组织代码的方式，使多个项目可以协同工作，共享资源，同时管理它们的构建和调试设置。解决方案文件通常有“.sln”扩展名。例如，如果你正在开发一个大型应用程序，可能会创建一个解决方案，其中包含多个项目：主应用程序项目、库项目、测试项目等。这样的结构使得不同项目可以在同一个解决方案下协同工作，而不必单独管理每个项目。

一个项目是一个独立的代码组织单元，它包含了实际的源码、资源文件、配置文件等。每个项目代表着特定的功能或模块。在解决方案中，每个项目都可以有自己的构建设置、依赖项和编译规则。项目可以是应用程序、库、控制台程序、DLL等不同类型的程序单元。例如，如果你在一个解决方案中创建了一个名为“MyApp”的项目，那么该项目就可以包含你的应用程序的源码、图像资源等。

总之，解决方案是一个容器，用于组织和管理一个或多个项目。每个项目是一个独立的代码单元，代表特定的功能或模块。通过将多个项目组织在一个解决方案中，可以更好地管理整个开发过程。

如果不想用IE浏览器显示程序结果，也选择其他浏览器，比如火狐浏览器（Firefox），但会出现报错，如图2-13所示。

图 2-13

此时降低Firefox对TLS版本的要求，网站就能正常访问了。设置步骤如下：

步骤01 打开Firefox的高级设置。在浏览器的地址栏中输入about:config，然后按回车键，打开Firefox的高级配置页面。打开页面后会看到一个警告，提示“更改这些设置可能会影响你的计算机……”，单击“接受风险并继续”按钮。

步骤02 搜索TLS版本设置。在页面的搜索框中，输入 security.tls.version.min，快速匹配与TLS版本相关的设置。在搜索结果中找到security.tls.version.min 这个设置项。默认情况下，它的值可能是2或3（笔者这里是3），代表Firefox只支持TLS 1.2或1.3协议。单击该项旁边的“编辑”按钮（或“修改”按钮，取决于Firefox版本），然后在弹出的对话框中输入数字1，以允许Firefox使用TLS 1.0及以上协议，最后单击“✓”（保存按钮）以应用更改，如图2-14所示。

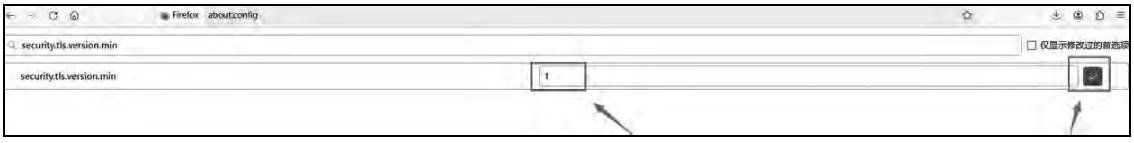

图 2-14

上面配置修改好后，Firefox将支持使用TLS 1.0及以上协议，网站就可以正常访问了。建议确保正在尝试访问的是一个可信的网站。如果是未知网站，降低TLS版本可能会使我们的连接更容易受到攻击，因此不建议降低安全标准。

2.3　生成和调试程序

2.3.1　为何要用生成

在开发和执行ASP.NET项目的过程中，一旦出现浏览器加载网页，则说明源码中已经没有语法错误了，所以通过执行程序的方式可以验证程序是否有无语法问题。但这种方式有个缺点，那就是IE加载网页的速度比较慢，如果依靠这种方式来验证程序是否有语法问题，那就比较耗时且让人心烦，因为多了一个关闭浏览器的步骤。

那如何快速检查源码中是否有语法问题呢？答案是可以用“生成”的方式，单击Visual Studio菜单栏中的“生成”→“生成解决方案”或按快捷键Ctrl+Shift+B，如图2-15所示。

图 2-15

“生成解决方案”是一个全面的操作，它会编译项目中的所有文件，包括已修改和未修改的文件。如果某个文件已经编译过，但没有发生修改，它就不会被重新编译。这个操作将确保整个解决方案处于最新的构建状态，包括所有项目和文件。“生成解决方案”只是编译而不会执行程序，因此通过“生成解决方案”可以相对快速地了解程序有无语法错误，如果没有错误，那么在Visual Studio下方的输出窗口就可以看到生成成功的提示，如图2-16所示。

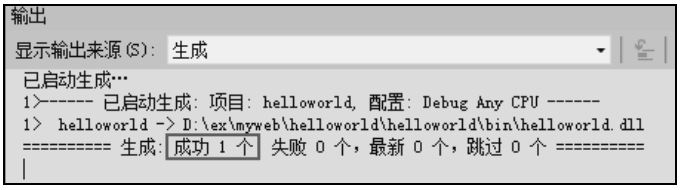

图 2-16

现在我们人为制造一点语法错误，在Visual Studio的“解决方案资源管理器”中找到并展开文件夹Controllers，然后双击该文件夹下的HomeController.cs文件，这是一个C#源码文件，此时Visual Studio编辑器中将显示HomeController.cs的内容。我们找到About函数，然后把下列代码结尾的分号去掉，如下所示：

```
    ViewBag.Message = "Your application description page."
```

学过C#的人都知道，C#中的语句必须用分号结束，因此这是个语法错误。我们马上生成解决方案，此时可以在Visual Studio下方的输出窗口中看到错误提示了，如图2-17所示。可以看到“失败1个”，而且提示“应输入;”。

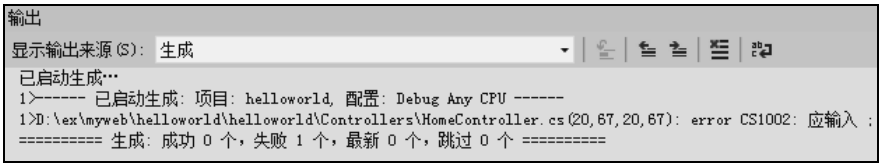

图 2-17

如果要定位错误行，可以直接双击输出窗口中的这一行：

```
    1>D:\ex\myweb\helloworld\helloworld\Controllers\HomeController.cs(20,67,20,67):
error CS1002:应输入 ;
```

这时，Visual Studio将自动在编辑器中将光标定位到出错行。

这样，我们就能通过“生成解决方案”的方式快速检查语法错误。另外，由于解决方案可能包含多个项目，因此在多项目的解决方案中，“生成解决方案”会对所有项目进行编译，而这不一定每次都有必要，因为有时我们只需要编译正在修改代码的当前项目，如果每次都把解决方案中的所有项目编译一遍，那耗时也很长。这个时候，我们单击菜单栏中的“生成”→“生成xxx”或按快捷键Ctrl+B，就可以只生成当前项目。这里的xxx表示当前项目的名称，比如图2-15中的“生成helloworld”。当然，在单项目的解决方案中，“生成解决方案”和“生成xxx”方式没有什么区别，都是在编译唯一的项目。

总之，“生成xxx”操作是相对于单个项目的。当我们在解决方案中选择一个特定的项目并执行“生成”时，只有该项目以及与之相关的依赖项会被编译。这可以用于快速测试和构建某个特定项目，而无须重新编译整个解决方案。

细心的读者可能会发现，“生成”菜单下面还有“重新生成解决方案”和“重新生成helloworld”，前者是重新生成解决方案，后者是重新生成当前项目。这里的“生成”和“重新生成”稍微有点区别：“生成”是在上次编译的基础上，只对改动过的文件重新生成，没有改动的文件不会重新生成；而“重新生成”会对解决方案或项目中的所有的文件都重新生成，如果引用了其他类库的DLL，也会重新生成其他的类库，这样速度要慢些，但可靠度要高一些。另外，在重新生成之前，还会有一

个清理中间文件的步骤。Visual Studio也单独提供了“清理解决方案”和“清理 xxx”两个菜单，但不常用。

2.3.2　增加工具栏按钮

“生成”和“生成解决方案”功能用得比较频繁，如果能把这两个功能作为按钮放在工具栏上，就会方便很多。在Visual Studio中，可以在工具栏空白处右击，在弹出的快捷菜单中选择“生成”，如图2-18所示。此时，工具栏上就会多出4个按钮，左边头两个正是“生成”和“生成解决方案”按钮，如图2-19所示。

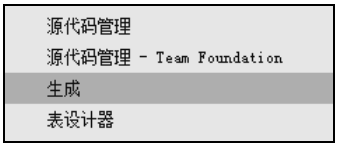

图 2-18

图 2-19

除了“生成”比较常用外，“开始执行（不调试）”也比较常用，因此我们也可以将“开始执行（不调试）”作为一个按钮放在工具栏上。在工具栏空白处右击，在弹出的快捷菜单中选择“自定义”，此时出现“自定义”对话框，在该对话框中单击“新建”按钮来新建一个工具栏，如图2-20所示。

出现“新建工具栏”对话框后，输入名称，比如“my”，单击“确定”按钮。然后在“自定义”对话框上切换到“命令”选项卡，选择“工具栏”，并在“工具栏”右边下拉列表框中选择“my”，单击“添加命令”按钮，如图2-21所示。

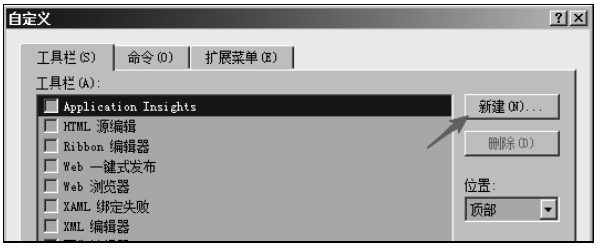

图 2-20

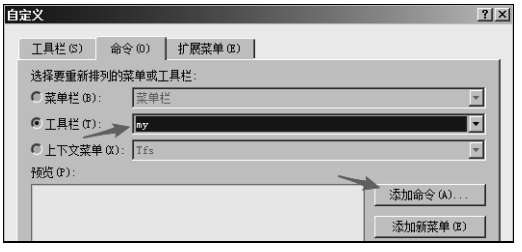

图 2-21

这个步骤就是为新工具栏my添加命令按钮。此时出现“添加命令”对话框，在左边“类别”下选择“调试”，右边选择“开始执行（不调试）”，如图2-22所示。

图 2-22

单击“确定”按钮，“开始执行（不调试）”按钮就出现在my工具栏上了，如图2-23所示。最后关闭“自定义”对话框即可。

图 2-23

2.3.3　单步调试 ASP.NET 项目

虽然通过“生成”方式可以检查出语法错误，但逻辑错误却只能在程序运行后才能察觉，比如对于1+1居然输出了3。如果逻辑错误比较复杂，我们还需要让程序在运行时在怀疑有错的代码行暂停下来，然后查看该代码行中相关的变量值是否正确。此时，就需要采用（单步）调试来达到这个效果了。

“调试”这一术语可能有很多不同的含义，但从字面上看，它是指从代码中删除bug（漏洞）。现在，可通过多种方法实现此目的。例如，可以使用性能探查器来调试代码，也可以使用“调试器”进行调试。调试器是一种非常专业的开发人员工具，它可附加到正在运行的应用中，并允许检查代码。调试器是在应用中查找和修复bug的重要工具。但是，上下文是关键所在，请务必充分利用可以使用的所有工具，以便快速消除bug或错误。

单步调试是指在程序开发中，为了找到程序的bug，一步一步跟踪程序执行的流程，根据变量的值找到错误的原因。相信读者都学过C语言了，应该对单步调试有过经验。现在使用Visual Studio开发ASP.NET，也可以单步调试，因为ASP.NET所使用的主要语言是C#。在ASP.NET项目中调试程序，主要就是调试C#代码。下面在范例中修改代码并进行单步调试。

【例2.2】单步调试ASP.NET项目

（1）打开Visual Studio，新建一个基于MVC的ASP.NET项目，项目名称是helloworld。

（2）在Visual Studio中双击打开Controllers下的HomeController.cs，然后将该文件中的About函数修改为如下：

```
public ActionResult About()
{
    double i = 0.126;  //定义一个浮点型变量
    string str = string.Format("the value is {0:p}", i); //格式化一个含百分号的字符串
    ViewBag.Message = str; //将str内容赋值给ViewBag.Message，最终显示在About页面上

    return View();
}
```

ViewBag是一个动态属性，它允许我们在控制器中动态地设置属性并在视图中使用。在这里，我们将字符串str的内容赋值给ViewBag.Message，那么在About页面上就可以看到str的内容了。除了传递字符串，ViewBag还可以传递复杂类型的数据，如对象或集合。

（3）按快捷键Ctrl+F5或单击工具栏上的“开始执行（不调试）”按钮（这个按钮前面已经添加过了）。注意，以后会直接说运行项目，不再说按快捷键Ctrl+F5或单击工具栏上的“开始执行（不调试）”按钮。此时将启动IE并显示首页，我们单击“关于”进入About页，就可以看到str的内容了，如图2-24所示。

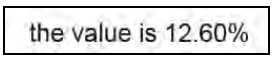

图 2-24

好了，程序没问题，关闭浏览器，下面来单步调试。

（4）在About函数中，我们在“double i = 0.126;”和“ViewBag.Message = str;”这两行的最左边分别单击，此时将出现两个红色小圈，如图2-25所示。

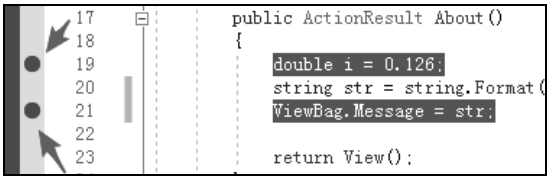

图 2-25

这个过程是为程序运行设置断点。断点的意思是当程序处于调试运行时，只要遇到有断点的代码行，就暂停在断点处，并且该代码行处于还未执行的状态。现在我们设置了两个断点，一旦程序执行进入About函数，碰到拥有断点的代码行“double i = 0.126;”，就会停在这一行。我们查看一下效果，首先让程序以调试运行的方式启动。注意，刚才按快捷键Ctrl+F5运行程序是以“不调试”方式运行，也就是会忽略所有断点，而调试运行方式则会遇断点暂停。要启动调试运行，可以按F5键或单击工具栏上的“IIS Express（Internet Explorer）”按钮，如图2-26所示。

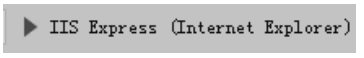

图 2-26

IIS Express是一个专为开发人员优化的轻型独立版本的IIS。借助IIS Express，可以轻松地使用最新版本的IIS开发和测试网站。它具有IIS 7及更高版本的所有核心功能，以及旨在简化网站开发的其他功能，包括：它不作为服务运行，也不需要管理员权限来执行大多数任务；IIS Express适用于ASP.NET和PHP应用程序；多个IIS Express用户可以在同一台计算机上独立工作。IIS是Internet Information Services的缩写，是微软推出的一个Web服务器。而Internet Explorer表示用IE浏览器来显示网页，相对于IIS服务端，IE就是一个客户端。Visual Studio现在很智能，不需要我们手动去架设Web服务器，做到了一键启动。

启动调试运行后，首先显示在浏览器中的也是首页，我们单击“关于”，此时将自动切换到Visual Studio的编辑窗口，并高亮“double i = 0.126;”这一行，而且该行左边红圈中还多了一个箭头，如图2-27所示。

这个箭头的意思就是告诉程序员，当前程序执行到这一行暂停了。现在我们开始按F10键单步执行，如图2-28所示。

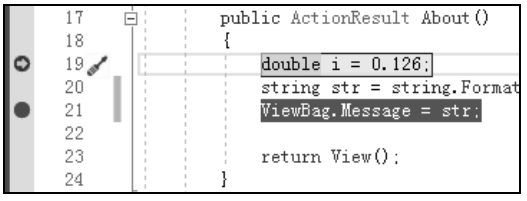

图 2-27

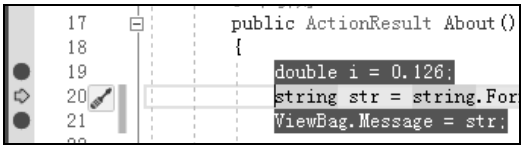

图 2-28

可以看到箭头指向“string str...”这一行，表示“double i = 0.126;”那一行执行过了，现在准备执行“string str...”了。如果我们把鼠标指针悬停在i上，此时变量i的值就会显示出来，如图2-29所示。

这就是设置断点让程序暂停下来的意义，即方便我们查看变量，以此来排查代码是否有错。另外，如果刚才不按F10键（单步前进），而是按F5键，那么将直接前进到下一个断点处，也就是“ViewBag.Message = str;”这一行。如果要停止调试运行，可以在Visual Studio中按快捷键Shift+F5或单击Visual Studio工具栏上的“停止调试”按钮，如图2-30所示。

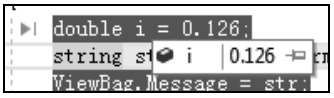

图 2-29

图 2-30

除了按F10键是单步前进外，按F11键也可以单步前进，而且碰到函数会进入函数内部，而按F10键则不会进入函数内部，它直接执行函数调用处后面的代码。读者以后可以慢慢体会。

至此，我们介绍完了Visual Studio调试ASP.NET的单步功能。其实单步调试是“重型武器”，有时杀鸡不必用牛刀。在怀疑有bug的代码处放置一句显示信息框的代码，也可以作为显示某个变量的手段，这其实是一种轻量级的调试手段。

【例2.3】在控制器的方法中显示信息框

（1）复制例2.2的项目文件到某个路径，作为本例项目，然后在Visual Studio中打开，并双击打开Controllers下的HomeController.cs，在该文件的About函数中的“return View();”前添加一句代码：

```
    System.Web.HttpContext.Current.Response.Write("<Script Language='JavaScript'>
window.alert('" + str + "');</script>");
```

其中，System.Web是一个命名空间（namespace），HttpContext是该命名空间中的一个类。类HttpContext用于封装某个HTTP请求的所有HTTP特定的信息，也称为HTTP有关的上下文信息。它的生存周期是从Web浏览器客户端用户单击并向服务器发送请求开始，到服务器处理完请求并返回到客户端为止。注意：针对不同用户的请求，服务器会创建一个新的HttpContext实例，直到请求结束才销毁这个实例。

为什么会有HttpContext类呢？在ASP（Active Server Pages）时代，人们都是在.asp页面的代码中使用Request、Response、Server等HTPP特定上下文信息的。而在ASP.NET时代，这种方式已经无法满足应用需求，于是产生了HttpContext类，它对Request、Response、Server等都进行了封装，并保证在整个请求周期内都可以随时调用。

当然，HttpContext不仅仅只有这点功能。在ASP.NET中它还提供了很多特殊的功能，例如Cache、HttpContext.Item等。通过它，我们可以在HttpContext的生存周期内提前存储一些临时数据，方便随时使用。举个例子：

运动员参加体操比赛，从进赛场的那一刻起，引导员就带领运动员到不同的项目地点参加比赛（如先比双杠，再比跳马，最后比自由体操），在整个过程中，引导员一直在运动员身边。尽管引导员不参加比赛，但运动员在比赛过程中遇到问题或有困难可以与他交流（比说要求他提供饮用水、保管衣物，或者问他一些有关比赛的问题等）。当所有的比赛项目结束后，引导员又会把运动员带出赛场，这时他的任务就完成了。这里，引导员就好比是当前请求的HttpContext，它保存了HTTP请求过程中的一些信息（如Response、Request等），我们可以通过HttpContext来访问相关的信息。

继续看Current。Current是类HttpContext的一个属性，表示当前HTTP请求的HttpContext对象（也称实例），也就是当前HttpContext实例。获取实例后，又可以调用类HttpContext中的属性了，这里调用的是Response属性，该属性用于获取当前HTTP响应的HttpResponse对象，该对象调用类HttpResponse的方法write，将信息写入HTTP响应输出流。输出流也就是服务器发送给浏览器的数据，最终输出到客户端浏览器网页上。现在我们输出的内容是一段JavaScript代码（简称JS代码）。JS代码通常用一对<script></script>包围，比如：

```
<script language='javascript'>
...
</scrip>
```

属性language用于指定具体的语言，这里指定的是JavaScript语言。省略号处就可以写具体的JS代码，这里的JS代码就是一句window.alert。window.alert是JavaScript中的一种方法，用于在浏览器窗口中弹出一个信息框，并显示指定的文本消息。在JavaScript中，所有以“window.”开始的语句都可以省略window，因为它是全局对象，可以直接调用其方法。现在我们传给alert的参数是变量str，那么就是显示str的值。这样，当程序执行到这个地方时，就可以查看这些值是否符合预期，是否出现错误。

图 2-31

（2）运行项目，然后在首页上单击“关于”，就能弹出信息框了，如图2-31所示。

其中，12.60%是double变量i的值。这就通过显示信息框的方式知道了某个变量值。这也是一种轻量级的调试手段，甚至是项目发布给用户后，在用户环境那里排错的一种有效手段，可以帮助开发人员了解程序当前运行状态，从而进行排错。因为用户环境不一定有开发环境，没办法进行单步调试。这个时候，打印日志或打印信息框就是唯一的排错利器了。注意，打印日志也是一种常用排错手段，就是把一些变量值或程序的运行状态写到某个路径下，即在C#中写文件。相信读者都会，这里不再赘述。

2.4　简要剖析项目

转眼间，我们已经完成了两个ASP.NET项目，而且还学会了调试。花架子要起来像个练武的人，现在我们提高一下内功心法，简要剖析ASP.NET项目，看看其内部运行机制。当然，本节不会讲得非常深，避免让你走火入魔，放轻松。

首先打开helloworld工程，然后运行它，此时出现IE浏览器，并显示首页。我们在浏览器地址栏的末尾加上字符串“Home/Index”，然后按回车键，发现依旧显示的是首页，如图2-32所示。

图 2-32

这里URL变成了https://localhost:44358/Home/Index。URL遵守一种标准的语法，它由协议、主机名、域名、端口、路径以及文件名这6个部分构成，其中端口可以省略。

对于地址栏的URL里包含Home/Index，按照传统网站URL的定义，应该可以在我们项目的根目录下找到Home目录，然后Home目录下找到Index的文件。但是，这里我们并不能在根目录下找到Home这个目录，只能在Views目录下找到Views/Home/Index.cshtml文件。我们在浏览器中输入https://localhost:44358/Views/Home/Index.cshtml这个地址，运行结果如图2-33所示。

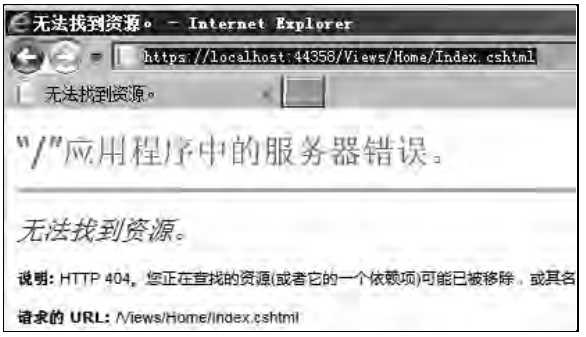

图 2-33

路径是对的，文件也存在，但为什么会出现404错误，说找不到文件呢？如果不是直接访问存在的物理文件，那么MVC又是怎样工作的呢？重点来了，原来MVC模式的工作过程是这样的，如图2-34所示。

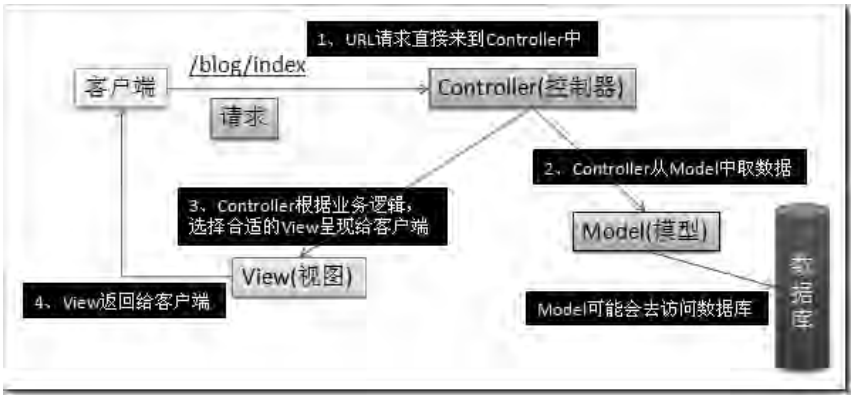

图 2-34

在MVC中，客户端所请求的URL被映射到相应的Controller（控制器）中，然后由Controller来处理业务逻辑，或许要从Model中取数据，然后由Controller选择合适的View返回给客户端。回到http://localhost:44358/Home/Index这个URL，它访问的其实是HomeController中的Index，如图2-35所示。

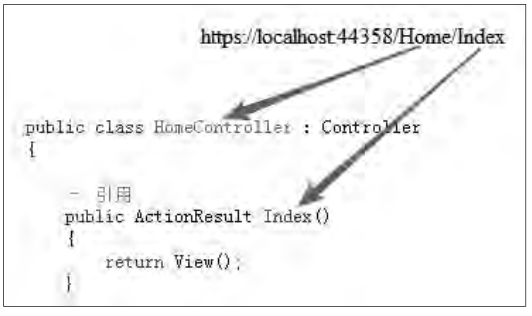

图 2-35

其中public ActionResult Index()这个方法被称为Controller的行动（Action），它返回的是ActionResult的类型。一个Controller可以有很多个Action。那么一个URL是怎样被定位到Controller中来的呢？

IIS网站的配置可以分为两个块：全局web.Config和本站Web.Config。我们在“C:\Windows\Microsoft.NET\Framework\v4.0.30319\”这个路径下可以找到全局配置文件web.config，在web.config文件的httpModules配置节中可以看到一个UrlRoutingModule：

```
    <add name="UrlRoutingModule-4.0" type="System.Web.Routing.UrlRoutingModule"/>
```

通过在全局web.Config中注册System.Web.Routing.UrlRoutingModule，IIS请求处理管道在接收到请求后，就会加载UrlRoutingModule类型的Init()方法。总之，就是这个UrlRoutingModule把URL定位（路由）到Controller中去的。而URL会被路由到哪一个Controller中，这些是完全可以自定义的。

项目启动从Global开始，我们到解决方案目录（helloworld）下的Global.asax.cs文件中去看一下：

```
    namespace helloworld
    {
        public class MvcApplication : System.Web.HttpApplication
        {
            protected void Application_Start()
            {
                AreaRegistration.RegisterAllAreas();
                FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
                RouteConfig.RegisterRoutes(RouteTable.Routes); //这里调用RegisterRoutes注册了
路由
                BundleConfig.RegisterBundles(BundleTable.Bundles);
            }
        }
    }
```

项目启动的时候，会调用Application_Start函数，在这个函数中会调用RegisterRoutes注册路由，而RegisterRoutes函数中会确定路由规则。那RegisterRoutes在哪里呢？我们到App_Start/RouteConfig.cs中查看一下：

```
    namespace helloworld
    {
        public class RouteConfig
        {
            public static void RegisterRoutes(RouteCollection routes) //定义函数
RegisterRoutes
            {
                //忽略对.axd文件的路由，也就是和WebForm一样直接去访问.axd文件
                routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
                //下面开始确定路由规则
                routes.MapRoute(
                    name: "Default",   //路由的名称
                    url: "{controller}/{action}/{id}", //带有参数的URL
                    //设置默认的参数
                    defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
                );
            }
        }
    }
```

可以看到这里定义了一个名为“Default”的路由（Route），还定义了默认的参数，也就是作为HomeController下的index。默认参数的意义在于，当我们访问例如http://localhost:44358/的URL

的时候，它会将不存在的参数用默认的参数补上，也就是相当于访问http://localhost:44358/Home/Index。

现在，我们知道一个URL是怎样定位到相应的Controller中去的，那么View又是怎么被返回给客户端的呢？从图2-34中可以看到，HomeController的Action方法index中有“return View();”语句。默认情况下，它会返回与Action同名的View。在ASP.NET MVC默认的视图引擎下，View按如下路径访问：

```
    /Views/{Controller}/{Action}.aspx
```

也就是说，对于http://localhost:44358/Home/Index这个路径，在默认情况下，在Index这个Action中用return View()来返回View时，会去寻找/Views/Home/Index.cshtml文件，如果找不到这个文件，就会抛出找不到View的异常。

那么，为什么前面我们直接访问Views/Home/Index.cshtml文件时会出现404错误，找不到文件呢？因为在MVC中，是不建议直接去访问View的，所以我们创建的ASP.NET MVC程序在默认情况下就在Views目录下加了一个Web.config文件，里面有这么一段内容：

```
    <system.webServer>
        <handlers>
          <remove name="BlockViewHandler"/>
          <add name="BlockViewHandler" path="*" verb="*" preCondition="integratedMode"
type="System.Web.HttpNotFoundHandler" />
        </handlers>
    </system.webServer>
```

也就是访问Views目录下的所有的文件都会由System.Web.HttpNotFoundHandler来处理，所以不要将资源文件（CSS、JS、图片等）放到Views目录中。如果确实要放到Views目录下，请修改Views/web.config文件。

至此，读者应该对MVC的工作原理有一个大概的了解了。