ASP.NET 打造互联网未来空间站

技术话题

如果你已经看过我们前面的那些文章，您有可能已经形成了这样一个想法:.NET 是微软公司非常重要的战略措施，微软公司正在花费大量的时间和精力努力营建它。其中，.NET 的一个不可缺少的部分是 ASP.NET。现在，我们来谈一谈 ASP.NET 的体系结构，看看它是如何与.NET 的整体架构协调工作的，以及如何利用 ASP.NET 开发大型网站那样的东西。

我们已经做 ASP.NET 三年了。我们刚开始主要是研究 ASP。众所周知，ASP 是个建立网络应用非常成功的平台，我们研究 ASP 主要看看它有些什么东西的确很值得保留，看看在哪些方面，我们可以改进它，于是我们花了一年半的时间进行头脑风暴式的集体研讨，产生想法并开始建设，这样就产生了 ASP.NET。此外，我们最近将 ASP+的名字改为 ASP.NET。

请问，你们这样做的确切原因是什么？这样做又意味着什么？

.NET 的起因是多方面的，比如将软件提升为服务；比如 XML 和 web 服务，他们用他们可做到的方式增强了互联网的性能。我们称之为 ASP+的东西是在我们有.NET 之前被命名的；一旦.NET 名字出来，我们就将它的名字和.NET 架构紧密联系在一起了。我们可以这么认为，它是 ASP 的新版本,并且紧紧地依赖于我们谈论的.NET 构架的所有东西。准确地说是这样的。

但是，我们认为关于 ASP.NET 重要的是，它并不是我们拿了来自 ASP 先前的版本的代码，将它移植到.NET 上，或者与.NET 相结合。我们真正从底层创建了它。这是一种彻底不同的代码，这是在我们花费 3 年时间所提出新想法的基础上，是在 common language runtime 基础上，是在 XML 基础上，是在所有其它.NET 技术的基础上构建的。

我们在上面讨论了 ASP 新的构架，并且命名为 ASP.NET，还重新建造了它，这难道意味着如果我是一名 ASP 程序员，我必须抛弃我学习过的一切，为了适应 ASP.NET，必须重新学习一种新思路吗？

绝对不是的。如果你是一名 ASP 开发者，你仍然可以保持你目前的全部技能，并且非常轻易地将它们应用到 ASP.NET 中。如果你想继续按照你原来的风格写，你可以继续使用。如果你想接受最新的编程

方式，你可以开始这种体验。不过，你可以渐渐地做到这一点，你可能会在许多新的特点中使用很多老的编码风格，并且获得这两种风格的最佳优点。

嗷，有那些新的特点吸引我前往 ASP.NET，为什么我应该向 ASP.NET 模式转移？

有大量的新特点。

第一，现在全部的语言都是可以编译的语言。这是，很多 ASP 用户渴望得到的事情。他们希望在 VB script 中得到 VB 的这种特征。现在你可以用纯粹的 Visual Basic 语言来编写你的 ASP。NET 程序。你也可以选择许多其他的语言，象 C 或者 JAVA SCRIPT 甚至 COBOL，我们已经证明 COBOL 可以运行在 ASP 内部，你可以使用这些更好的语言。你也可以拥有更加简明和强大的页面开发工具。因此，你可以利用很多的方法将对象行为变成 TAGS，这样，你能仅仅把 TAGS 放在你的页面上就可以实现很多行为；

第二，你可以不必写很多的代码。我们做很多试验，约 400 行的 ASP 代码我们把它减少到了 20 行代码。这样，可以提高工作效率，创造的更多的价值。

另外，从创建你的网站并且使它能被其他的计算机自动访问这个角度看，web 服务也是一个重要的新特点。例如一个以 XML 为基础的 web 服务的例子，它可以把不同的种类的信息连结到一起，就像电视节目列表。商务公司在用这种方式其他的商务公司交流信息。所以，能轻松地制作这些基于 XML 的服务是 ASP.NET 的另一个重要的部分。

不是 SOAP 也在提供同样的事情吗？

是的，但是，ASP.NET 提供了一种非常简单的方法，建立了基于 SOAP 的 WEB 服务。

啊，我知道了。SOAP 的一个方面是将你的页面那些告诉你你的过程所涉及的属性、方法、事件等小提示包装起来。因此，ASP.NET 在页面上可以自动完成这些事情。

ASP.NET 从根本上提供给这样一种自动化的方法，你用它建造的不是页面，:你实际上用它建造的是有独立的文件扩展名的 WEB 服务。在这样的页面里面，你实际上能定义使网站被调用的方法，你需要做的一切只是在页面里写你的代码，点击保存。并且，如果你用浏览器点击那个方法，你将实际上得到所有可以调用的方法的全部列表，以及一个如何通过 SOAP 调用它的方法的描述。你既可以用标准的查询字符串调用那些方法，也可以动态的创建一个 com 代理对象，使网络通信全部自动化，包括全部 http 调用，以及 XML 集合，使用 SOAP 调用这个服务就像你调用一个储存在服务器里的对象的常规方法.

就像 ASP 的产生使得编写动态的网页的工作变得轻而易举所用的方法相同，ASP.NET 的产生真的使写动态的 WEB 服务变得轻而易举。而实现这一功能，你只需要认真考虑你想写的代码和你想处理的数据。你根本不必想到 XML。（当然，如果你想考虑 XML，你能考虑 XML。）

你可以写一个提供类别名字的方法，我可以用它来查询我的库存数据库，以及我的其他信息：我的生

意怎么样了，我有多少库存，我将返还给你一个那个类别的所有产品列表；比如,如果我是食品杂货商店你向我询问牛奶的信息，我将给你我卖的牛奶的全部不同的种类信息。

你在 ASP+中做这件事（当然，他真的意思是想说 ASP.NET）的方法是:你仅仅用 VB, C#编写一个类，该类有一个方法比如说取得类别，并且他还包含类别名字。接下来，它能进行查询 SQL SERVER 数据库或任何 XML 数据源，把数据带回。这个过程实际上不必认真考虑转换成 XML 这些事情。ASP.NET 会自动完成 XML 的转换工作。

因此，我们能够自动地将一个 xml payload 转换成一个整数。然后，如果你返回一个 records set 的时候，我们将自动地处理 records set，并将他变成 XML 数据流。因此，在 ASP 中不得不做的那些事情：手工分析 XML,手工将 HTTP 的内容转换成对 XML payload 的访问并分析他,这些过程都被自动的完成了。我们在其中提供动态的维护 contract 能力。Contract 主要用来解释如何调用 web service 的。因此你可以很容易的传递这个 contract 到其他站点上了，其他站点也可以直接对你的站点进行编程。

我们的网站作了一个很有趣的例子，提供了一系列的调用 ASP+（即 ASP.NET）的例子，他们提供的 web service 是：允许其他 ASP.NET 站点列举这个站点在某一天写的所有新的例子。因此那些站点可以订阅这些服务，并且可以对这些 web service 编程，规定他们是一天一次还是一个小时一次获取网站更新的那些例子的链接列表。

这样说来，对于用户来讲，他们没有必要考虑与浏览器或者 web 页面打交道的这些事情，他们直接通过 web service 就可以访问互联网了？

是的，但是那将是 web service 的美好的远景，那时我们将会认识到，互联网实际上是所有信息的总和。我们曾经为了便于人们浏览提供了在浏览器中显示 html 的方式，但是如果您的合作伙伴或者您公司的其他工作人员的一些进程或者程序需要调用这些信息时候，目前的技术实在是在无能为力的。XML 技术的产生使得人们随心所欲表现任何数据成为可能。因此,你可以利用 xml 创建更多的 web service，以便人们可以用多种方式调用程序或者交换数据。Xml 的确是一件强有力的工具，他也可以帮助人们构建更多种类的应用程序。Xml 技术也给客户提供了更多的帮助，由于他们同样可以获得数据和信息，所以他们可以做更多的事情。

由于有了以上这些技术，你不一定非要从客户端基于浏览器的应用程序中调用WEB服务。你还可以在一个普通的VB或者C#应用程序中使用WEB服务。

这些都不是什么问题。当然，这里面还有一个问题应该强调一下。我们所使用的 SOAP 和 XML 都

是开放式协议标准。这些标准不是WINDOWS系统独有的，他们现在已经被许多家生产厂家所推崇。所以我们可以和UNIX系统，可以和基于 JAVA 的系统交流，无论是发送信息还是接受信息，我们都可以用这种方式来交换信息，而不必考虑那台机器上是否安装了ASP.NET系统。如果真的会这样，你将会看见遍及互联网的应用程序用他们感兴趣的方式开始了互联。我们使用这些新技术的结果是，那些富客户端的应用程序会渐渐被人们遗忘的。

当然从另一个方面来看，尽管ASP.NET和.NET框架的主要目标是启动任何一种类型设备上的应用程序，但这并不意味着这些设备必须运行在.NET框架上。在 ASP.NET平台上，我们已经作了许多直接支持 WML 这样协议的事情。因此，你可以直接开发出与较小的设备，例如与 WAP PHONE 或者其他可以处理HTML的小设备通信的应用程序。

因此，你用一个简单的AST.NET页面，可以指向许多类型的设备。这些设备可能具有完全不一样的特征。

是的。

这么说，是 ASP.NET 代码确定了如何到一个小设备或者大设备这样的事情吗？

是这样的。我们有一套特殊的服务器控件，我们称他们为 MOBILE CONTROL，他们可以根据设备的类型，自动地发送 WML,HDML 或者HTML信息，譬如说如果你有一个爱立信的手机，他可能只有 4、5 行显示文字的区域，如果你调用的一个页面时，我们可以自动地根据这个设备的情况提供合适的显示方式。

这是否意味着开发人员不得不考虑他们所遇到的所有不同的情况，例如，这是个彩色的浏览器那是一个 POCKET PC 的显示界面。

不会的。这实际上就是ASP.NET最迷人的部分。通过这些服务器控件，你实际上找到了一条包含你所想象的不同行为方式或者对不同类型设备的不同输出的最好的解决问题的办法。我们可以在许多不同的情况下实现这些事情。例如，我们有一套控件是用来作确认工作的，确认的工作是一个站点中非常复杂的工作。特别是当你想生成客户端的代码，并且由这些代码来做确认的工作的时候，你会遇到很多麻烦的。现在，我们这套控件自动地为你做这些事情。它们包含了最基本的确认行为。对于富浏览器端的程序，他们将发送一种东西，他们将发送 SCRIPT 代码，对于服务器端的程序他们也将履行确认这个过程，并产生相同的结果。只不过这里这时的主要工作是在服务器端进行。

服务器端控件自动的处理客户是有意按照客户端脚本运行还是故意不按照客户端脚本运行等情况，并且可以很好地控制这一切。你所做的只是在你的页面中宣布这些控件为ＴＡＧＳ，然后写上 2，3 行借用他们处理的代码，你便拥有了一个完整的确认页面了。

你说的只是一些基本的确认，他们支持对数字的确认吗？

是的，有一系列的东西我们支持。我们支持电话号码、邮政编码等相关的东西。我们会遇到很多的问题，有时候要确认添入的东西是不是数字，有时候他们添入的数字要匹配一些规则的表达式，有时候添入的数字要满足一定的界限的限制。

这些要求是可扩展的，我们由第三方厂家来从事这方面的工作。其中，第三方厂家做的一项工作是关于邮政编码的确认。你可以在一个页面的文本框里面使用这种确认，他将判断你输入的数字是否是一个有效的邮政编码，因此你可以在许多不同的场合使用这些确认功能，他们可以使您更容易编写页面程序。

实际上，完全组件化的系统允许开发人员和第三方的工作人员几乎在系统的任何一个层面插入他们的代码，这是我们一个最大的革新措施。我们常听见使用 ASP 的那些开发人员说过的一件事情：我喜欢 ASP，我很喜欢 Session State 这样的东西，但是我希望他可以工作在 WEB FARM 上。例如他们会问，我如何使得他工作在 WEB FARM 上？答案是，在 ASP 的环境中那是不可能的。但是在 ASP.NET 中，我们可以使得 Session State 工作在 WEB FARM 中，因此你可以尽情的使用了。此外，如果你想完全取代 Session State 你也可以做到；你想修改缓存输出的方法，你也可以轻松做到，因为整个系统是组件化的。如果还有人提出，尽管我们的喜欢 response 、 request、session 以及与他们相关的这些应用，但是我还想在 ISAPI 这个层面做更多的工作。我应该如何去实现呢？答案是，在 ASP 中没有办法实现，因为所有的事情都封装在 ASP 系统中。没有办法达到这么低的级别中，但是，如果你采用 ASP.NET 的话，由于它是组件构架的系统模型，所以你可以做你想做的事情。

嗷，在你提及 ISAPI 的时候，我想起了在 ISAPI 应用中有两种完全不同的模式：ISAPI 应用模式和 ISAPI 过滤器模式，ASP.NET 是否支持这两种不同模式？

是的，他支持这两种不同的模式。

我可以想象他是如何支持 ASP.NET 应用模型的，难道他也用 ISAPI 过滤器做一些事情吗？

是的，他也用它做一些事情。我们也支持一种叫做 ASP 模块扩展点的技术。他基本上允许你像一个

函数那样使用 ISAPI 过滤器。我们提及的很多例子，如替换 Session State 等工作都是在那个级别实现的。此外，重定向 URL 的工作也是在这个级别实现的。例如，我们的许多客户经常要为他们的用户提供个性化的 URL，像 financial institution 站点就可以为他的顾客提供 www.financialinstitution.com/scott 这样的 URL 为一个名叫 SCOTT 的用户，因为他们不想创建太明显的路径，所以要借用重定向技术提供各种各样的不可视的 URL，你可以在 ASP.NET 上实现这样的功能。

带有"/scott" URL，实际上被 ASP.NET 转换成他要去的地方，是吗？

是的，因此在实际运行前，你会有一些代码解析"/scott" 部分，转换成一个实际地址的页面，实际执行的是一个很普通的页面。这种方式可以使得所有的用户都得到个性化。我们有一个模块中的一个 API 可以为你解决这个问题。在这个问题上，我们也有一个扩展点。在 ASP 的时候，我们有一个叫做 Global.asa 的概念，他可以定义应用开始，应用结束，Session 开始，Session 结束等事件。在 ASP.NET 中我们允许您将 ISAPI 过滤器功能加入 Global.asa 中，因此，如果您想继续重定向 URL 时候，你甚至不需要写一个模块了，你只需要在 Global.asa 中定义就足够了。

我们做的另一点是，努力提高安全水平并且允许更为灵活的认证，在过去，您使用 ASP 的时候，您主要用 IIS 来构建您的认证功能，这种认证实际上依赖 NT SAM，在 ASP.NET 中，我们将安全模块放入 ISAPI 过滤器中，因此如果您想在一个 main frame 或者在你有一些 usernames/passwords 的数据库中，并且是你自己做这种认证时，你可以在 Global.asa 中写入相关的信息或者在一个与那个事件同步的组件中加入相关信息。

因此，我可以写一个我自己用户名的数据库了，并且，他还带有标准的认证过程。

是的，没错。如果你这样做了，你可以获得更多的东西，你不仅可以拥有了用户名字和口令，而且还会有角色（role）这样的东西。你可以将那些用户映射成角色，你可以使用这种方式而不是仅仅使用 ACL 的方式构建你的站点。你可以知道 Joe 和 Mark（用户名）访问了这个页面，相反，你也可以知道这个页面被哪个角色访问了。你在数据库的主要工作就是告诉我们 Mark 的用户名和密码是什么，他在哪一组角色中，其余的工作由安全系统来考虑，以保证它可以访问那些网页或者服务。当然，这里需要提及的是，所有的架构，安全架构、caching 架构都为网页和 XML 服务提供支持。

如果从性能角度来看，增加的这些东西，这些能力，可以提供更好的性能服务吗？

我们的确提供了更好的性能服务，当我们创建 ASP.NET 项目组时，我们就指定一个开发人员专门负责性能上的工作，我们当时使用的是早期的 common language runtime，他每日的工作就是检查

每项操作是变快了还是变慢了，并且与运行时(runtime)项目组一起查找问题，加速变慢了的操作。同时，他也指出性能上的问题。这样做的结果是 ASP.NET 比 ASP 要更快。所有的代码都是编译过的，编译成本地代码，不再用解释的方式。

是吗，那么说我在 ASP.NET 中写的程序代码再也不用解释的方式？

正确。此外，更好的是，您仍然可以像您从前那样在编辑器里面编写代码，点击 SAVE 存盘，然后剩下的事情都交给系统去处理。但是，对于剩下工作 ASP 与 ASP.NET 确有不同处理方式。ASP 采用的是分析代码然后传给脚本引擎,在程序运行时解释代码 ;而 ASP.NET 则是传给编译器,然后在 common language runtime 上运行。在 common language runtime 上执行的是本地代码，因此你可以用 VB 编制程序获得与 C++一样级别的性能。

那么，它是在什么时候编译，是在我存储文件并且 NT 文件系统发现文件变化时，还是页面第一次使用的时候？

当某人请求页面的时候发生。我们将会检查，是否某个页面已经被编译，如果没有编译，我们将会编译他。我们在 ASP.NET 中作了许多很有趣的工作，使得编译过程更有效率。因为我们编译了他，所以如果你为了使机器更可靠或者性能更好而重起机器或者 shut down 你的 WEB 服务器或者 shut down 你的进程时，你不用再编译你的那个页面了，因为我们已经检测到了该文件已经被编译和加载，这样可以获得更高的响应速度和更强的伸缩性，并且那也是因为把它放在 CACHE 的某个地方了。

下面我们在谈论一下性能。好，我们谈论一下性能，是的，从整体上看，性能是更优化了。我们也在可扩展性，可靠性，可获取性方面作了很多的工作。我们曾经有一个假设，在一个 common language runtime 上，即使有一个 Session 想要和所有与他相关的东西通信的话，应用程序的可靠性仍然变得很高。

我们还可以举些例子。例如，真正强大的类型检查。在一个可管理的环境中，如果出现数组越界的情况，系统将会抛出一个 exception 而不会产生一个垃圾内存。除此之外，我们还有更多的可靠性机制保证您的使用。例如，我们可以检测内存冲突，也就是内存在某一点徘徊。我们可以检测到死锁，你甚至可以配置系统以便在发生这样的情况的时候，你的机器可以每隔多长时间重起一次。因此，在出现这样的情况的时候，我们可以规定一小时一次、一天一次或者每隔 200 个请求一次等来启动一个应用程序的新的实例。然后允许旧的应用程序完成相关的任务后，新的程序接着进行。

保持这样的一个可靠性机制使得你不必中断任何为用户工作的服务。我们将会动态地生成一个新的进程，开始接受新的请求，老的进程完成当前运行的所有请求，然后我们将删除老的进程。在这样的过程中，

为了提高效率，系统实际上作了 RESET 的工作。但是这个过程中，你不会看见管理人员的干涉。你的客户也不会感觉到任何变化，他们会认为，嗷，每一件事情都运行的这样好，服务器还在运行。用这样的方式，你就可以获得非常高的可靠性。

此外，我们在 Session State 上也有了一些改进，Session State 不再需要和你的代码运行在同一个进程中。他可以作为一种服务运行在一台机器上，也可以作为一种服务运行在整个 WEB 服务器群中的某一台机器上，其他的前端机器可以共享该服务。

Session State 只能存放在一台机器上吗？

不，不是的，如果需要的话可以在多台机器上保留。Session State 可以在多台机器上存储，也可以有多台机器的前端去访问那些存储信息。

我是一个用户，我正在访问一台机器，我的 Session State 留在这台机器上，如果我下一次要访问另一台机器，Session State 中的数据可以共享吗？

是的，这种工作方式是 WEB 群组最典型的工作方式。一个用户你可以访问机器 A，等一会儿，你又会访问机器 B，你的 Session State 保留在另外一台独立的机器上。Session State 既可以运行在我们的 Session State provider 上面也可以运行在 SQL Server 上面，我们有一种方式可以让 SQL Server 成为 Session 信息的服务器。但是，前端的这两台机器，也可以是多台机器，都可以访问同样的 Session State。

这样看来，我们可以通过这种方式共享数据了。

是的，正确。但是你可以想象一旦你拥有这种 Session State,与你的 ASP 代码不在一个进程中运行的 Session State，无论你的 ASP 程序遇到下面什么样的问题，程序崩溃、我们终止了你的代码运行、我们检测到某种资源漏洞或者锁定、我们所配置的一些事情（例如一小时一次）遇到问题时，你所有的 Session 依然是安全的。因为，你将你的 Session State 保存在另外一台独立的机器上了。

如果你只有一台机器，你也可以利用 Session State 的这种特点，你可以让他运行的进程与你的代码运行的进程不同，因此，只要进程到来时，他们就可以访问在安全保护进程中的 Session State。

正如我们刚才提到的，Session State 最重要的几个好处是：（1）可靠性。你不用再担心你的应用程序崩溃或者一个用户数据丢失这样的事情发生。（2）第二个好处是，现在，你的应用程序不必再局限于一台机器。如果你使用了 Session State ，你的确可以将你的应用扩展到更多台的机器上，这样，

就真的没有什么条件能够限制你了，也就是说，你不必再担心你的应用程序能够在多大范围内或者能够为多少用户提供正常服务这样的问题。

听了上面的讲解，我们好像做了两种不同的事情，一个是 ASP.NET 提供了许多新的功能和性能，人们可以充分利用这些新的特点；另外一个方面是，在这些功能和性能的后面，ASP.NET 做了许多额外的工作，使得这些特征和功能发挥了更大的作用。

是这样的

对我来说，如果我已经有了一个 ASP 的 WEB 站点，如果他已经为我提供了满意的结果，我不想再改变程序中的任何东西，如果我将站点的整个框架更新到新的 ASP.NET 上或者类似的东西上，那么就意味着我的站点获得更多的鲁棒性。

是这样的，你说的的确是一个很有趣的观点。我确信，如果你是那个用户，你说那个站点的应用程序运行良好，我的确不想做任何改变，如果我需要升级到 ASP.NET 框架的时候，我只想在我的机器上安装这个 ASP.NET 即可。如果你这样做了，我们并不会替换你的机器上任何的 ASP 代码，因此 ASP 与 ASP.NET 运行在一台机器上。但是，你实际上将不得不将运行环境让位给 ASP.NET，并且重新配置你的机器。尽管作这样的事情没有什么难的，但是你需要告诉我们你想让 ASP.NET 运行 ASP 的那些程序。因为，如果一个机器里已经存在了多个应用程序的时候，你再想安装一个新的运行环境，新的编译环境，和所有新的.NET 的那些东西时，我们不能完全保证你的每个应用程序在升级后的环境中都运行良好，都带给你同样的性能。做服务器程序的开发人员将会用很长的时间调试他们的程序，以确保程序正常工作。

如果我是按照你们说的去做，那么，好，我安装了 ASP.NET，我标记了我的文件，并且告诉他们：现在你们运行在 ASP.NET 的环境中，就算是我根本没有对文件作任何改变，我是否可以获得具有更高鲁棒性和更好柔韧性的解决方案？

是的，你将获得具有更高鲁棒性和更好柔韧性的解决方案。不过，在这里我要提及的一个重要的事情是，我们当前的解决方案不是百分之百地与 ASP 兼容。众所周知，尽管我们的 ASP.NET 与 ASP 之间具有非常非常的亲密关系，但是，最近几年里，当我们开发这个平台的时候，由于若干的总总原因，我们不得不牺牲一些兼容性。但是人们根本不会感觉到这些。在 ASP.NET 中，我们获得 COOKIES 的顺序实际上稍稍发生了一些变化。这一切正如我们期望的那样，不会有百分之百的兼容。不过最重要的是，无论发生了怎样的变化，如果你安装了 ASP.NET 后，你现有的代码和你现有的网页将依然会正常的工作。

从整体上看 ASP.NET 对你是有帮助的，但是由于它本身的一些小小的变化，你将不得不随之作一些小小的调整。我们会有一些自动化的工具帮助你完成这一切。但是，对于大型的应用来讲，毫无疑问，

你要做一些改变。

让我们再一次回顾一下理解你的应用程序架构的一些关键的事情或者简单看看那些新的事物

很好，这是一个很有意思的部分。这其实是一种移植工作。你可以将一个页面，将你所掌握的技巧，将你能够你所做的一切，将你写的东西移植到 ASP.NET 上，你依然可以用同样的技巧，同样的技术，但是你将获得更高的性能。当人们开始了解这个平台的时候，我推荐给人们的是应该了解这个平台的所有特征。因为除了移植外，还有很多的新的特征。也可能不仅仅是移植的过程还有一点点重建的过程。例如，我们在前面提到了确认的过程，大多数的应用程序都可以用到不同程度的确认过程。但是，他将不再是“保留你原来的确认逻辑，并且让他运行的更快一些”，而是，完全被我们的确认组件所替代。

许多人都已经用 ASP 建立了他们自己的某种安全架构，他们可以用它来验证登录信息。现在当他们知道 ASP.NET 带有这种功能是，他们会说，这下可好了，我可以直接使用 ASP.NET 内嵌的认证功能了，在也不用我自己编的那些东西，不再用 Session State 跟踪用户信息。

Caching 是与性能相关的的一项重要的功能，我前面没有提及，许多的开发人员非常努力并且使用了很多的非常好的技术试图弄明白一个页面的局部或者服务器上的整个一个页面的运行结果，以便当两个浏览器请求同一个特殊资源时，不必重新生成两次，我只要使用 cache 中的版本就可以了。在 ASP.NET 中，我们提供了一个对 cache 的支持。

我想说的基本也就这些，但是我认为他们是值得花时间的，特别是你想快速开始时，在 SDK 中，我们有大约 900 来个例子，这是一个很大的数目，我们有一些很大的端到端的应用程序，一个电子商务的应用，一个建立类似 INTRANET 门户框架的应用。我认为，认真研究这些应用程序，并且看看他们是如何构建的是一件很值得做的事情。因为你会发现你找到了很多技巧，意想不到地减少了你写代码的数量。使你的代码更干净。我想，你将会突然发现那些的确难以管理的 50、60 行的代码变成了 5、6 行清晰的代码，几周之后，你再回来看看时，一目了然。很容易弄清楚是什么意思。

小结：

关于 ASP.NET 最重要的事情是，首先，他使得开发人员的效率大大地提高了。他使得应用程序和系统的可靠性大大地增强了，有许多东西使得应用程序真的很容易配置。例如，当你的应用程序正在运行时，如果你复制一个 DLL 时，他没有被锁定，这样你的应用程序就可以平滑地移植到新版本上。举这个例子是为了说明可以真正地提高性能、可靠性、可用性、可扩展性。我想，所有这些能力都实际上是系统

中最重要的事情。除此之外，我们还增加了大量的新的功能。我知道有很多的人看着 ASP 从 1.0，2。0，到 3.0 成长的，他们今天看到 ASP.NET 时候会说，嗷，对于 server 对象或者 response 对象以及其他类似的东西还有三种不同的方法。这就是新特征带给我们的巨大的财富。

程序员话题

下面我们谈一谈开发人员如何快速地开始 ASP.NET 的程序设计。

你发现 ASP.NET 中什么东西比 ASP 更吸引人？

有些人说是配置，ASP.NET 配置起来更容易吗？因此，我想谈论关于这个问题的一点内容。我们的确发现 ASP.NET 的一个优点是可以简单地配置。由于 ASP.NET 的应用程序是由编译过的代码组成，所以不需要注册 DLL 或者停止某些服务。例如，在今天的 ASP 环境中，如果在你的网站上，有一个非常重要的商业组件需要被一个功能更加强大的组件替换时，你将不得不暂停 WEB 站点的服务，以便使得 DLL 不被锁定。然后替换该组件，重新在系统中注册，重新启动你的 WEB 站点，这样做的结果是在某一段的时间里，你的站点将不能对外工作。然而，令人兴奋的是在 ASP.NET 中，你不需要在做上面的任何工作,你在也不必考虑那个该死的 regsrv32。你只需要将这个新的 DLL 放到与老的 DLL 系统的目录下。当前访问的新的 DLL 的请求仍然在工作，新的请求将会触发新的 DLL,这一切会持续到老的 DLL 完成所有的请求时为止。那时，新的 DLL 就会代替老的 DLL,这一切不会引发停机，这一切只需要调用 XCOPY 的配置功能，这一切使得我们的工作更有效率，更加简洁。

这样做的好处是不用再考虑哪个 DLL 需要关闭，然后再关闭它；哪个 DLL 中有 BUG。是否有一种方式可以使得当前应用的所有的 DLL 迅速的被关闭，还是必须要将 WEB 站点停掉。

对我来说，保留它前面的版本，后面的版本是一件很容易的事情。只需要利用 XCOPY 重新配置一下就可以了，那些 DLL 很快会被自己替换。

好了，除了前面所见的配置简单的这个特性以外，你们还有那些优点足以吸引 ASP 程序员快速移植到 ASP.NET 环境中？

这个问题问得好，如果我是一个 VB 程序员，当我进入 ASP 编程环境时，我会发现，他是如此熟悉，我乐意立刻在那里编制程序。我不用担心那些脚本语言，不用再考虑那种从头到尾都是线性的处理模型，我可以在 ASP.NET 中使用更新的逻辑关系，使用提供的更多新服务器组件，提供比 ASP 程序更多的功能。

确实是这样的，ASP.NET 节省了我们大量的的开发时间，你可以在同样的环境下开发供 ASP+页面调用的 DLL,并且可以安全的使用。然而,对大多数的开发者来讲,最大的收益来自于他提供的 caching 功能。

我们都理解 caching 功能对于标准的 web 页面的重要性，客户端的 caching 可以使得页面很快重现，服务器端的 caching 存储了一些已经编译过的代码，可以提访问的速度。

我们下面举一个 caching 的例子，一个在线的商店，这是一个销售 CD 的网上商店，在他的主页面上，你列举了一系列的商品。因此，你使用了数据库存储商品信息、价格信息、种类信息等。当 ASP 脚本访问数据库时，不可避免地会有时间延迟。但是，当你使用 caching 时，一些访问过的信息将会留在 caching 中。当请求的信息在 caching 之内被检索的时候，caching 中的页面作为原页面输出，因此，你不必在访问数据库就可以获得数据，因为他们已经在 caching 中了。.NET 框架会一直监视 caching 中的页面，如果这些页面相关的数据库的信息发生了变化，他就会立即更新这些页面。因此，你不必担心得不到最新的数据信息。通过 caching 的设置功能你还可以设置 caching 的时间长度，规定他在多长时间内定期更新 caching 中的内容，你也可以在 caching 中缓存页面的点击数。总之，通过 caching 技术，你实际上为你站点的用户提供了快速获得信息的可能性。在本质上，他实际上直接获取的是生成好的 HTML 页面，从而避免了一系列的页面生成的过程。

caching 将真正帮助网站的开发人员调整网站的性能以及快速的相应客户的请求。

是的。实际上你可以看到这样的情形---网站服务器每秒钟可以相应更多的点击数，因为用户大部分的请求都落在 caching 中。

从开发人员的角度来看，如果他们在编写第一个 ASP.NET 程序时，他们会遇到哪些大的困难？

我对您的话感到吃惊，从 ASP 到 ASP.NET 并没有不可逾越的障碍。他们只是在一些细节上有所不同。当然有许多的是要引起注意的。例如，我们过去常用 ASP 与 VB SCRIPT 编写程序，当我们想创建一个 record set 对象的时候，我们不得不 SET 一个变量等于 record set。在 ASP.NET 中没有 SET 这个参数，直接就是变量等于 ADO records set 对象。所以，在 ASP 与 ASP.NET 之间仅有一些小的语法的差异,这些小的语法差异根本不会影响到页面的性能,但是如果你将 ASP 程序移植到 ASP.NET 上时，你要注意这些小的差异。实际上，从 ASP 迁移到 ASP.NET 的代码量是很少的，不必考虑将整个程序代码移植，这两者实际上是可以并存的。因此你不必强制将你的网站的程序立即移植到 ASP.NET 上，你可以在新的工作中逐渐采用新的 ASP.NET 技术。

我想，你仍然可以像你从前那样声明你所有的数据类型。你仍然可以使用 Server.CreateObject，你仍然可以使用 DIM RS，定义一个 record set 对象，你仍然可以使用 DIM RS AS NEW ADO record set 这种方式，对吗？

这只是我们的一种选择，在我们最少量的移植级别里面，你实际上不用考虑 ASP 与 ASP.NET 的区别。你仍然可以使用 DIM RS 的方式，并且用这种方式创建与数据及相关的对象。这两者之间的主要区别是一个是用 VB SCRIPT 创建的对象，另一个是用 VB 创建的对象。

这么说，你们现在是已经改变到一种拥有类型的开发语言平台上了？

你说的很对，我们有数据类型了。我们现在更严格的进行应用程序的内存管理，而不是将每一件事情能够都看成是变量，我们现在可以将不同的事物看成 string、 integer、a data set，因此我们可以更好地控制内存的使用。

接下来，我们应该演示一些代码用以解释上面所说的一些东西。

好的。我们的第一个问题是移植问题。这是一个典型的 ASP 代码的页面，我们将从数据库中取出数据放到表中。我们来看一下这些代码，我们首先定义了我们的 connection 和我们的 record set， 我们创建了这些对象，我们用 SET 关键字设置我们的 record set 等于 connection 的执行。接下来是一个 DO WHILE 循环，将数据库中的信息显示在页面上。

```
<%
Dim con, rs
Set con = Server.CreateObject("ADODB.Connection")
con.Open
"Provider=SQLOLEDB;server=(local);database=Northwind;UID=sa;PWD=;"
Set rs = con.Execute("SELECT ContactName, City FROM Customers")
%>
<html>
<body>
<table border="0">
<tr>
```

```
    <td>ContactName</td>
    <td>City</td>
</tr>
<%
Do While Not rs.EOF
%>
<tr>
    <td><%=rs("ContactName")%></td>
    <td><%=rs("City")%></td>
</tr>
<%
    rs.MoveNext
Loop
%>
</table>
</body>
</html>
```

如果转到.NET 框架下，仅需要少量的移植工作。我们可以看见哪些东西已经被改变了，哪些东西不能使用了。我们在 Set con = Server.CreateObject("ADODB.Connection")和 Set rs = con.Execute("SELECT ContactName, City FROM Customers")中所使用的 set 之类的关键字一去不复返了。但是在这里我要指出的是，在下面的代码中，我们在 rs("ContactName")和 rs("City")的右边添加了一个属性 Value。他的基本含义是，我们可以获得指定行或者指定列的值。从 ASP 到 ASP.NET 也就这些改动，实际上有很少的变化，你可以看到 ASP 与 ASP.NET 代码之间几乎一样，没有什么特别大的变化。

```
<%
Dim con, rs
con = Server.CreateObject("ADODB.Connection")
con.Open("Provider=SQLOLEDB;server=(local);database=Northwind;UID=sa;PWD=;")
rs = con.Execute("SELECT ContactName, City FROM Customers")
%>
```

```
<html>
<body>
<table border="0">
<tr>
        <td>ContactName</td>
        <td>City</td>
</tr>
<%
Do While Not rs.EOF
%>
<tr>
        <td><%=rs("ContactName").Value%></td>
        <td><%=rs("City").Value%></td>
</tr>
<%
        rs.MoveNext
Loop
%>
</table>
</body>
</html>
```

这是很吸引人的，那些 SET 关键字被去掉了，VALUE 关键字被引入，代码本身实际上没有什么变化，因此只要你看看输出结果，你就会发现他们基本上是一致的。但是，ASP.NET 是运行在.NET 框架下，他的页面扩展名是.ASPX，当然我们在该页面也可以实现 CACHE 的功能，只不过在这个程序中没有用到罢了。如果我们将上面的实例进一步深入下去并且考虑应用.NET 框架和 ASP.NET 框架所提供的一些更为便利的手段。例如，用 managed providers 来获取数据，那么我们将以入下面这个事例。我们依然工作在 SQL 7.0 上的 NorthWind 数据库中。

.NET 提供了一个重要手段是 SQL managed provide，它可以直接连接到 SQL 上获得数据而

不必通过 OLEDB 这样的东西，这种方式经过实践被认定是大大提高了访问速度。我们可以看一下下面的代码,他们使用纯粹的 VB而不是 VBs编写的。但是输出的结果几乎一样读。我们要做的事情是 import 一些 namespaces ，System.Data，这样我们就会有最基本的数据处理能力以及获得对 SQL managed provider 访问能力的 SQL namespace。这些代码的风格与前面讨论的C#的风格好象很一样，那时我们使用 C#来处理基于.N E T架构的各种类库，现在我们所见到的在 VB 中的.NET 类库与 C#中的类库具有相同的风格。

这两者之间的确是同一种风格。如果我用 C#编写，代码会不同，但是 namespace（名字空间）是一致的。对于编写 ASP.NET 的程序，这两种方法都是很好的。尽管某些类可能会有一些不同，但是我们可以仅仅修改几个的地方，就可以将 VB代码快速移植成 C#代码。

实际上，许多代码将不会被改变（一直保留下来）。例如，SQL connection 字符串只需要改变一点点，SQL select statement 字符串不会有什么改变。在这里，我们强调一下，对于 connection 字符串我们不再使用 ADO provider，不再使用 SQL OLEDB 作为 ADO provider,因为我们使用了 SQL managed provider，因此我们提倡使用 SQL7.0 以上的版本，他能给我们更好的性能。我们创建一个 data set用以保存我们的数据以及后来访问的结果。我们用 SQLDataSetCommand 检索 SELECT 的结果，并返回页面，将他们插入到 data set 中，我们命名这个表为 Customers。因此，我们现在有一个 data set 对象，他包含了一个 Customers 表。如果需要的话，我们可以加入更多的表和更多的关系。与 ADO records set 不同的是，我们现在可以处理更大的事情，我们可以将整张大表分成若干个小表，而他们之间的关系去保持不变，这种方式有利于更大的程序应用开发。.

```
<head>
<script runat="server">
Sub Page_Load(Source As Object, E As EventArgs)
  Dim ds As New DataSet
  Dim dsc As SQLDataSetCommand
  dsc = New SQLDataSetCommand("SELECT ContactName, City FROM Customers",
_
                        "server=localhost;database=Northwind;UID=sa;PWD=;")
  dsc.FillDataSet(ds, "Customers")
  dgCustomers.DataSource = ds.Tables("Customers").DefaultView
  dgCustomers.DataBind()
```

```
End Sub
</script>
</head>
```

.

接下来，我们跳过几行代码，直接显示与 ASP data grid server 控件相关的代码。ASP data grid server 的作用是提供一个最基本的表。我们要做的唯一的一件事情是告诉他我们要在服务器端运行他，他是服务器端的控件。接下来我们设置 the border 宽度为零 ，因为上面的例子中 border="0"。此外，我们新加了一个属性，他用来改变 items 的背景色。他用来标识在 ASP 表中不同表格项的颜色，这样便于区分。当然，你要写很多的代码，用循环的方式改变每一个项目的背景色，在这里我们简化一下工作，只是标志这个属性。他表现的色彩是浅灰色。

```
<body>
<asp:DataGrid id="dgCustomers" runat="server"
    AlternatingItemStyle-BackColor="#CCCCCC"
    BorderWidth="0"
    />
</body>
</html>
```

我注意到这里有一个 ASP 的 tag,他的作用是标示一个控件吗？

是的，他的前缀，ASP 的前缀。表明这是一个 ASP 的控件，后面的部分表示这是一个服务器端的控件，是在.NET 框架下定义的。我么也可以形成我们自己的控件并且给他们一个独立的前缀，实际上我们已经为我们的站点定义了一些我们内部使用的控件。好，现在，我们看一下这个程序的运行结果，这个运行结果与上面的结果相似，所不同的是数据库访问依赖的是 SQL provider，另外，增加了一个小小的定义背景色的属性。我们这个程序与上面的程序的设计基本相同，但是却很容易添加这样的属性，我们每天都用这样的方式处理成吨的表格、添加许多种其他的属性。实际上，这也是使用服务器端控件的好处，他能使我们更容易在 HTML 中表现方法和属性，他使得编程工作更容易，更快捷。

因此，我们看见这段代码与前一段代码的主要区别是他不需要让程序员编写遍历整个 records set 的 WHILE循环。也不需要在 HTML 中的某个合适的位置显示结果。你可以很简单的生成一个表，然后告诉他：你的值来源于 records set。

对的，是这样的。刚才有一件事情我跳过去了，就是你说的那件事情。我们回过头来在看一下那段代码。我们用名字调用这些 data grid 并且将他们捆绑到来源于 data set 的结果以及 data set.中的一个特定的表格，接下来的工作就可以使用这些数据了。

```
dgCustomers.DataBind()
```

上面的例子是一个将一个很简单的表格显示在屏幕上的程序，这种表现方式看起来是简单有效的，但是，对于一个比较复杂的表格，还要进行很多次的复杂运算以及显示更为花哨的字符串等要求的程序仍然可以用同样的方式遍历整个 records set 吗？

当然可以，这实际上是.NET 服务器端控件的一大特点。我们很容易对 datagrid 做一些客户化的定制工作，因为，他是一个很基本的原始表格。也可以对 datalist 做一些客户化的工作，使他的每一行或者每一个单元对应一条记录。我们可以通过循环的方式用模版定制他的每一个重复的地方，有一种 repeater 数据控件，允许完全用模版定制他的每一步，因此，我们有一个头模版和一个一个尾模版分别定义了模版的起始和模版的终止。我们还有一些 item 模版和一些 alternating item 模版，他们可以为我们提供更多个性化的设计。我们还有很多的好东西放到了网站 ASPNextGen.com 上面，在那里我们用 data repeater 作了许多的工作，同时也提供了更多的客户化的输出。

太好了，太好了，我可以像现出那些激动人心的功能，刚才你提到了你可以开发你自己的控件放到页面上去，如果现在的 table grid 控件不能够满足你的需求，你可以写一些自己的代码并将改造过的控件应用到任何你想要用到的地方。

是的，不过最有意思的是，你不仅仅可以定制你自己编写的控件，而且还可以扩展服务器端组件的功能例如我可以扩展 data grid 功能，以满足我自己的特殊的需求。做这些事情，我们可以使用我们完全编译的服务器端控件，也可以使用一种另外一种控件---用户控件（user control）。用户控件是一种中间件，他介于页面控件和服务器控件之间，你可以通过用户控件显示方法和属性，也可以表现更多的功能。

开发这些额外的用户控件要比单纯写一个ＡＳＰ的程序复杂，但是可以获得更加强大的功能，是这样的吗？

你提了一个好问题。开发用户控件的复杂性会令你非常吃惊的。如果你创建一个 ASPC 类型的文件，并且写一些ＨＴML 在文件里面，你就已经拥有了一个最简单的用户控件。如果你想让一个页面内容包含在每一个页面里面，你可以将这个页面定义为一个 ＨＥＡＤＥＲ，像使用ＩNCLUＤＥ文件那样使用它。你可以很容易在你的用户控件中包含ＨＴＭＬ的结果。因此，最简单的用户控件程序就是 HTML 程序。此外，如果我想用同样的方式在这些页面中加入一些代码，我可以在用户控件中提供一些功能并且暴露一些属性。例如，我想在每个页面上设置我的用户控件背景色，我可以在我的用户控件中暴露背景色属性，

但是，从另外一个角度来看，这将会是一件相当复杂的工作。例如，你要做一个服务器的控件，一个 data grid 控件不会很容易地。因为它涉及到ＮＥＴ框架中的一些东西。有一种创建此控件的方法，必须发生在提交方法之前，发生在子控件被提交页面之前。因此，如果你要做数据绑定或者设置背景色等工作，你就应该在合适的时间内完成，否则当该页面被提交时，你将得不到你想要的东西。如果从这些方面来看，做这种控件将会是是一件复杂的事情。我曾经做过一个控件，一个 ad rotator 控件。它相当于.ＮＥＴ服务器控件的一个子控件。它可以绑定到一个 XML 文件上，该文件完成的是当页面被访问时在你的的站点上显示不同的广告。在他的存储表单里面，有一个图像的ＵRL 指向图像文件，一个导航 URL，当你点击广告时，该 URL 将你带到与该广告相关的那个页面。这个页面是不固定的，因为它涞源于不同的厂商。该页面可以是图像，也可以是文字。

到目前为止他还没有提供给我们找到那个 BANNER 显示了多少次的方法。因此，我在这个控件中加入了这个属性，就像是放入了一个倒计时器。因此你可以在ＸML 文件中放入一个默认值，设置该默认值为一个人 100 或者另一个人 200 等。你的默认值会逐渐递减直到为零。此时图像将不再显示了。我想建立这种控件还是会花费的一定的时间的。

无论你是一个经验丰富的ＡＳＰ开发人员还是一个经验不丰富的 ASP 开发人员，你都可以非常容易地做一个很基本的用户控件。但是，如果没有今天的技术，你可能要花很大的精力，可能要深入更底层去做许多工作，并且这一切还取决于你的经验丰富程度，以及系统提供的相应函数的丰富程度。

这有点像用汇编写一个 HELLO WORLD 的程序，那会是一个很复杂的过程。但是用Ｃ语言写这个程序则很容易。这并不意味着用Ｃ语言写所有程序都非常容易。但是，看起来好像这种方式扩大了开发人员的开发能力。因此，如果一个开发人员想开发一些运行在他们 WEB 服务器上的复杂的应用程序，他们要在 ASP.NET 上做很多的工作。

你认为ＡＳＰ.ＮＥＴ未来将提供什么样的功能？给予 ASP.NET 的 WEB SERVICE 将会被人们设计成什么样子的？

那些都是无止境的。不过信用卡的认证将会是一个巨大的功能需求。输入卡号、名字和地址，然后让 WEB SERVICE 返回每一部分的布尔值 YES 或者 NO，接下来要么认证成功进行交易要么由于认证失败而退出。当然也可以返回名字拼写错误、认证完成的消息或者一个邮政编码是否有效等结果。

在这个功能上就这么多的东西，我们还有一些 WEB SERVICE 的例子在我们的站点上，他们都是很基本的应用程序，你可以调用他们，他们都是我们放到网站上的一些教程程序，他们可以给你足够的你想要的信息。关于 ASP.NET 和 WEB SERVICE，我认为会有一个巨大的市场和他们相伴而行。你应该知道一个叫 Amazon.com 的公司，他们做了许多将服务连接起来的工作。人们可以在他们自己的网站上销售书，但是书的信息却通过 Amazon.com 表现出来，那些销售书的人得到了信誉。这可以看成是 WEB SERVICE 的一个典型的例子。这个例子很好的表现了当你浏览一个书的列表时候，如何让这些书显示出来？无论是神秘小说还是 StephenKing 的小说，都可以通过 WEB SERVICE 返回结果。

Amazon.com 实际上并不是你要访问的站点，但是你却可以从哪里得到图像、得到标题、得到书的列表信息。你可以用你喜欢的方式显示他们。

是的，WEB SERVICE 的优点是他是基于 xml 的，因此你可以用这种方式得到信息，并且用你喜欢的方式处理信息。只要能够分析并处理 xml 信息，你就可以使用 web SERVICE，因此你没有必要一定要跑到一个.net 的 web 站点上去使用 web SERVICE。

=---

什么是做.ＮＥＴ程序最重要的事情？如果你给我们观众一些指点，他们将会更快地接受 ASP.NET。

好的。句法这样的东西，你已经用了很多次了。你只需要学习它们之间的不同就可以了。如果你遇到完全不同的处理方式，你就不必按照过去的方式去做了，因为你做脚本工作已经很长时间了，所以会有很多不同的东西需要学习。其次，关于 assembly，namespace 这些概念都要弄清楚。接下来，是语言的问题。我曾经是一个 VB 程序员，现在我是一个 C#程序员，我喜欢 C#，它是一种很，好的语言。如果有 20 行的

VB 程序 C#5 行就可以解决问题。但是对于 COM，.NET 框架的已经为你处理了那些常见的工作，所以你只需要编写一些业务逻辑的代码就可以了。

你说的是理解“什么是 Assembly?“

是的，Assembly，namespaces、相关背景、整个.NET 架构，都要理解。

理解 namespaces 和 Assembly 的关键是理解他们的概念。你理解它们越多，你对架构的理解就越深入。主要理解它们是什么和你为什么需要它们。我认为一些 ASP 开发者，或是其他一些没有任何开发背景的人进入 ASP 领域的原因是因为 ASP 非常好理解，它很容易接受并且你可按照自己的方法去做。如果要变换到.NET 框架还是需要一些努力。如果你有开发背景就比较容易理解，因为你熟悉相关概念，熟悉数据格式，并且你知道过程调用这些东西，对于新涉入的人来说就需要做更多的工作。那些经常做脚本开发的工作人员需要做更多的努力。

ASP 编程模型在整个WEB页面上使用的都是脚本语言,无论是客户端脚本还是服务器端脚本都使得程序显得非常凌乱。你经常要更改的一些代码或者不得不四处修改一番。现在，ASP.NET 好像已经为你提供了更加规范的开发方式。

你现在能完全分离出你的代码。你可以有 HTML 代码和程序代码，能处理数据和得到动人效果。

你提到你是个 C#程序员，ASP.NET 在页面上支持 C#。我们看到 SQL Server 的那个例子，你使用的是 VB，不是 VB script，你是不是在页面上也能用 c#开发。

正确。或者是 EIFFEL、VB、COBOL 其他任何一种.NET 支持的语言。你可以利用这些语言做你想做的任何事情，但是最最重要的，你不再被什么条框限制。比如做 ASP程序，你只能选择 VB script 或者 javaScript 这些东西。现在则不同了，你熟悉什么语言？你更喜欢用什么语言开发？你就可以使用什么语言开发.net的程序。如果你喜欢 COBOL，你会感觉到在.NET 框架下做程序更顺手、更有意思。我想这种革新对很多使用其他语言的人都开启了方便之门。我曾经和很多 C++项目开发者工作过，我不想说他们恐惧 ASP，但是他们确实不喜欢使用 VBScript 这些东西，现在好了，他们不必再为此烦恼，他们可以使用他们最喜欢的语言开发。

是的。在我们即将结束我们的谈话时，你还有什么话要对我们的听众说吗？能对他们深入学习并使用

ASP.NET 有些益处。

热爱它，学习它，尽情使用它，我想这是关键问题。ASP.NET 确实是伟大的工具。它使网站的开发更为容易。并且我想，正如上面提到的，它变得更有意思，更顺利、更容易。所以我想说，如果你打算学习 ASP.Net，现在就开始，你会精通它并且感到充满快乐。