第 1 章

aesthNETics

在过去的数十年里，技术的发展日新月异。现在的最新技术可能在 6 个月后就完全过时了。您刚刚开始了解一种新技术，它就被更新了，而且您必须完全重新学习。对于.NET 开发人员来说，情况更是如此。当.NET 刚刚引入时，它对以前的 Microsoft 编程语言做了巨大的改进。只要看看这个事实就能说明一些问题：VB 和 C 语言程序员能理解彼此的代码了。但是多到令人难以置信的新控件和其他小玩意能使很多人望而生畏。在大家都还没有习惯.NET 1.0 之前，1.1 就发布了。当人们最终开始习惯 1.1 时，2.0 又发布了。当 2.0 发布不久，新的 3.0 又发布了。有些程序员喜欢.NET 这样不断地改进功能。他们努力学习所有的新功能和增强特性，以便他们的项目能跟得上技术的发展。他们还会研究这些新功能将对他们的现有应用程序产生什么样的影响。另一方面，多数人关心这种架构的“内幕”部分。他们想知道新的.NET 2.0 GridView 是否真的比.NET 1.1 DataGrid 好得多(确实好得多)。他们能立即发现在代码中实现 Try…Catch…Finally 语句的好处。

但是这些程序员似乎经常会忘记，或者至少将其放在相对不太重要的位置的是——使这些新的强大的.NET 页面看起来漂亮。他们忘记了 aesthNETics。

1.1 aesthNETics 的定义

简单地说，aesthNETics 能使.NET 页面看起来漂亮。这样解释可能有点太简单了。“使.NET 页面看起来漂亮”到底意味着什么？它仅意味着“美学上令人愉悦”吗？较短的答案是：不是。

较长的答案是，要看您是否喜欢 Web 站点。对此大多数人会有一致的看法。虽然可能难以提出当您访问 Web 站点时希望或预期的功能的权威列表，但是大多数人应该都会期望下面这些特点：

- 能够轻松地找到要找的内容。
- 访问的每个页面有一致的外观。
- 页面以直观且有逻辑性的方式布局。
- 如果使用了图形，图形不要过于突出或令人讨厌，同时也不要为了使用图形而使用图形。
- 使用的颜色不伤眼睛，否则您马上就会想离开这个页面。
- 不用一边加载页面一边想“哇，这里要发生的事太多了。”

- 页面加载的时间也不要太长，即使在较慢的连接上也是如此。
- 不会立即想“啊，以前我见过这种模板，它太普通了。”

除了上述这些外，开发人员可能还会考虑如何为他们的客户提供最佳体验。例如，要考虑到浏览器的兼容性。至少必须决定要支持哪些浏览器，此后为站点写的代码都要符合这个标准。不过，在考虑浏览器的兼容性时应当记住，有很多上网的人或访问您的站点内容的人使用的是基于文本的浏览器。您为这些人写好代码了吗？

aesthNETics 结合了所有这些思想。这意味着您没有忘记站点的外观，意味着您不仅操心站点的业务逻辑与数据，而且对于使界面看起来用户友好和给人带来愉悦的感觉给予了同等的重视。这还意味着您记得使访问站点的客户能看到的内容与他们看不到的内容同样出色。

优秀的 aesthNETics 程序员需要掌握的技能包括：

- 对 Web 设计和布局基础知识的良好理解。
- 色彩图形的鉴赏能力(有时少实际上就是多)。
- 层叠样式表的运用能力。
- 全面了解使所创建的页面具有一致性的 ASP.NET 工具。
  - 站点导航　对站点进行直观的和一致的访问的组件。
  - Master Page　用于结构化布局。
  - 主题　用于使.NET 组件保持一致。

以本书作为指南，您将学会鉴赏自己开发的 Web 站点的设计元素。您将了解关于图形的足够多的入门知识，并在此基础上学会更多内容。而且，与.NET 程序员关系最密切的是，您将学会可以帮助您在开发的站点中使用 aesthNETics 的.NET 2.0 新增功能。

不过，请不要误解本书的范围。本书不仅概述 Web 应用程序开发人员可用的特定.NET 工具，还介绍了在.NET 程序之外也可以使用的有声 Web 设计原理与工具。包括对 Worldwide Web Consortium(W3C)建立与维护的层叠样式表及其标准的讨论。本书还将概述色彩、图像及其他通用的 Web 相关设计事项。不过，随着对本书学习的深入，您将看到如何在 Visual Studio 2005 中综合运用这些基本原理，以及由.NET 2.0 Framework 提供的组件。严格来讲，本书讨论的很多概念并不只是.NET 所特有的。不过，aesthNETics 的思想是主要在.NET 应用程序中集中应用这些概念。

1.2　aesthNETics 之所以重要的原因

您可能会把 aesthNETics 看作典型 n 层应用程序的表示层。很多表示层是关于编程的，.NET 当然也是这样，它不但给出了关于如何让 Web 站点更强大的大量细节，而且充斥着最新最出色的技术。aesthNETics 不仅引入了 AJAX 以及 XML 串行化等主题，而且还介绍了在您开发的 Web 应用程序中实现业务逻辑的现实世界示例。不过，大多数表示方案最缺少的似乎是对表示层的应有关注。这种状态的风险在哪里呢？很简单，永远会存在看上去真的很无趣但又真正强大的 Web 站点。

表示层重要吗？当然重要。它与经典的三层应用程序中的业务逻辑和数据访问层一样重要吗？这个，要看您问谁了。一方面，您可能会遇到一名经验丰富的程序员，他阅读了当前大部分期刊，参加过旨在跟上最新技术趋势的技术会议。他可能会说，“ 不，它并没有那么重要。” 但是如果问的是客户呢？当然，他们可能会对使用 SQL Server 2005 的新交叉表功能所带来的访问等待时间减少 235ms 感到高兴。或者，他们可能为通过正则表达式验证 Social Security 字段的有效性然后在数据库中加密这一事实叫绝。但是，如果您告诉他们所有这些功能，而他们看到的却是只包含几个文本框的空白页面，他们会说什么呢？这通常被称为 “ 光环效应 ”，即借用最初的理解来判断一个条目的其他属性。对于 Web 设计而言，这意味着潜在客户通常会根据他们在浏览器中初次加载时所看到的内容来评判整个 Web 站点。如果在开始讨论幕后价值之前失去了客户，则需要费很大的力气才能把他们重新请回来。

为了说明这一点，假设有一个想要数据显示板应用程序的客户。他们有一些数据放在电子表中，且希望先把数据转换为某种企业解决方案，然后实现一个每天都自动将数据送到新的数据库的功能模块，最后在公司内网上用一个可浏览的 Web 应用程序显示对数据的分析。这是一个令人瞩目的项目，对负责交付这个新系统的小组意味着赞美与认可。结果，两个小组为设计新系统项目和将它呈现给领导班子而展开竞争。第一组花了大量时间进行数据转换，并创建了一个需要参数的自动更新系统。他们决定放弃花大量时间在项目界面上，而只用一些 Microsoft Office 图形组件来显示数据。他们呈现给管理层的示例如图 1-1 所示。

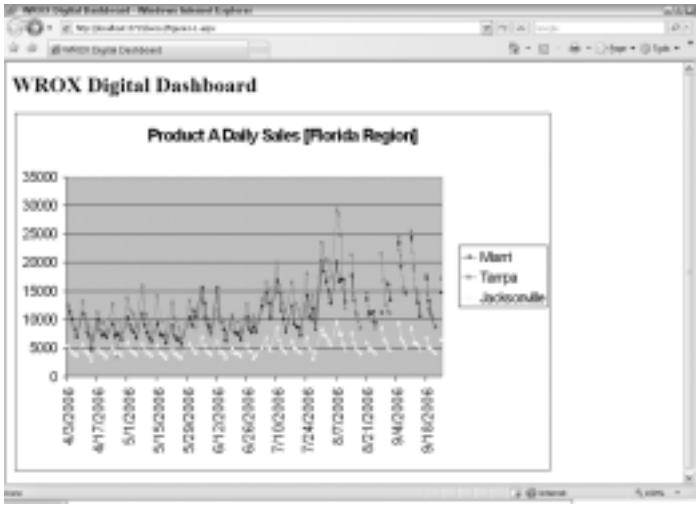

图 1-1

第二组也完成了初始数据转换，但是为了演示他们的提案，他们花时间拿出了一个既漂亮又直观的布局，希望项目管理层会喜欢。他们决定用 Adobe Photoshop 来表现自定义图形，并决定用 System.Drawing 命名空间创建自定义图形组件。他们密切关注站点的色彩方案，并努力使一切融合在一起。这使得他们没有时间来处理自动数据加载。他们呈现的示例如图 1-2 所示。

图 1-2

您认为在实际情况下，这两个项目中的哪一个会赢得管理层的首肯？

不要误会。表示层不是 Web 项目中唯一重要的组件。在上面的示例中，如果指定第二组来完成这个项目，项目开始以后不久，就会发现他们只能使站点看上去漂亮，而对于收集和分析数据却没有真正的价值，管理层对此是不会高兴的。仅使站点看上去漂亮无疑是不够的。不过本例主要是为了说明界面设计的重要性，提醒大家不要忽视它。

1.3 提高员工的期望

在如今快速发展的技术市场上，程序员被越来越频繁地要求成为编程方面的“ 多才多艺者 ”。这只是意味着客户和管理层希望程序员能处理 Web 项目的各个方面，从需求收集到应用程序开发，再到用户测试和文档化的协调。是的，甚至还有图形处理和设计。有多少工作是您希望公司不要都让您来做的呢？程序员已经发现完全不可能说‘ 我不管数据库，那是 DBA 的事。”随着时间的推移，您又会发现很难开口说“ 我不做图，那是设计师的事。”

即使这种情况还不是真的，也肯定有其他理由让您成为关于.NET 应用程序的更好的 Web 设计师。理由之一是，它可以使您从同行和同事中脱颖而出。假设您在一个做大量.NET Web 应用程序的编程工作室工作，其中大部分程序员精通.NET。您的能力差不多处于中间水平。您不是小组中最优秀的，当然也不是最差的。如何使自己与众不同呢？您当然可以多花时间成为小组中的最佳.NET 开发人员。但是除此之外还能做什么呢？

Phillip Van Hooser(*Willie's Way: 6 Secrets for Wooing,Wowing,and Winning Customers and Their Loyalty* 一书的作者，领导能力和客户服务方面的专家)提出了一个可以让您鹤立鸡群的简单而直接的方式。您不必比别人出色太多，只要稍微多做一点即可。他解释道，哪怕只比别人多做了一点点，人们都会注意到您而且会记住您。

那么，如何才能做得比小组中的同行好一点点呢？如果您精通 Web 设计与布局，如果您会用 Adobe Photoshop 等图形处理软件创建自定义图形，您就可以脱颖而出了。如果您创建了美学上给人深刻印象的应用程序，而其他人创建的应用程序乏味且缺乏灵气，那么人们自然会记住您。

1.4　为 Web 站点创建一致外观：aesthNETics 要点

本书将尝试概述部分基本工具，主要是在 Visual Studio 2005 中可用的基本工具，.NET 开发人员可用这些工具来使站点的外观与它们幕后的编码同样精彩。如果说存在严格意义上的 Web 开发规则，那就是：使站点外观一致。打开一个站点，一个页面是这种样式，而另一个页面又是完全不同的样式，将令人非常讨厌。这并不是说不同页面的布局不能有变化，而是说“让用户感觉每个页面是统一的”应成为开发人员的目标。的确，有时候必须拉长标题来适应大型的数据表。或者根据用户在站点中的位置，您会在页面的某些部分放置一些边栏链接，以及链接其他页面的链接，以便适应一个有很多内容页面的大站点。但是，当用户导航时链接的位置向左或向右移动 5~10 个像素，或者一个页面用蓝底模式，而另一个使用褐色阴影，将会使站点看起来比较邋遢。

当然，创建外观一致的站点的第一步是实实在在地确定您想要的外观。事实上，如果外观(可以轻松地应用到所有页面的外观)只是一个空白页面以及上面的文本和各种控件，那么它并不真正符合优秀 aesthNETics 的标准。虽然精确定义优秀 aesthNETics 的标准可能比较难，但是可以用“哇”因素来测试自己是否在正确的轨道上。如果向客户展示您的项目，客户说“哇”，那么您就可以知道自己可能符合 aesthNETics 标准了。如果客户没有这么说，或是看上去不大感兴趣，则您可能要重新思考自己的方式了。由于不同的客户有不同的“哇”标准，所以这种测试也是一个挑战。找出这些标准是一个 aesthNETics 开发人员的职责所在。您应当知道潜在客户是谁，他们会如何访问您提供的内容。您应当对 Web 站点的设计与布局有一个基本的了解。为了标新立异，还应对图形和色彩的应用有一个基本的了解，而且要熟悉帮助创建这两者的工具。您应在开发过程的早期就挑选颜色和创建图形，这样才好决定站点中的其他必需组件。关于如何做这件事的详细解释以及有哪些可用工具，请参见第 2 章。

一旦决定了站点的外观，就需要确定一种在站点中实现这些外观的方式。大多数应用程序都有模板工具。例如，您可以轻易找到可以应用的皮肤，Windows Media Player 等应用程序甚至可以自己创建皮肤。有些 Web 设计器应用程序也包括了皮肤。不过历史上.NET 并没有真正明确地用模板制作页面的方式。

这一点在.NET 2.0 Framework 发布时发生了巨大的变化。Master Page 现在是 Framework 的一部分，而且应当成为每个.NET 开发人员的设计工具与技巧的工具库。Master Page 允许开发人员为站点的设计创建结构化模板。这意味着您可以说“我希望标题在这个位置，页脚在那个位置，导航放这里，内容放那里”，然后将它保存为项目模板(找不到更好的说法了)。然后其他页面就可以继承 Master Page，它们的内容可以放到在 Master Page 中创建的内容区里。第 7 章将详细介绍模板的创建。目前，只需要知道 Master Page 能创建可导入

到 Web 页面中的项目模板就行了。

层叠样式表(Cascading Style Sheet，CSS)已经诞生十多年了，它是创建站点一致性的另一种很好的方法。使用 CSS，不仅能规定站点中 HTML 元素的外观，而且能规定 XML、XHTML、SVG、XUL，以及 SGML 的大多数派生元素的外观。例如，您想让整个站点的背景为灰色，就可以用 CSS 来实现。在新版 Visual Studio 2005 中，引入了一些帮助您在应用程序中创建精确样式表的工具。此外，还有几个将它们应用到站点的很酷的技巧，这将在第 4 章介绍。

然而，使用 CSS 的一个缺点是不能有效地用它创建 ASP.NET 元素的一致外观。这意味着单用样式表时不能说“我希望放在页面上的每个.NET 标签都是粗体红字，Arial 12 磅字体。”您只能创建一个格式化为这样的 CSS 类，然后在标签控件的“CssClass”属性中引用。但是，当您在该页面上放入另一个标签时会发生什么呢？如果没有在该标签的 CssClass 属性中对 CSS 类进行类似的引用，它就不会被应用；该标签只会使用标签的默认设置。然而，有了.NET 2.0 Framework 后，就能修改标签的默认设置。您可以告诉应用程序，对于放在任何页面上的每个标签，它应当应用某个特定的 CSS 类。更好的是，您可以说对于每个 GridView 控件，标题将是蓝色的，行样式将是灰色的，而交替行样式将是白色的。当您在页面上放下一个 GridView 时，不需要做其他任何事情，它就会被格式化。而且，您可以告诉应用程序，在一些情况下它应使所有的 GridView 都显示为上面描述的样子，而在其他情况下，它应使它们完全不同(基于配置文件、浏览器规范等)。您也可以基于控件 ID 让一些 GridView 看起来是这样，另一些看起来是那样。这些都是通过使用皮肤和主题来实现的，具体内容将在第 8 章和第 9 章介绍。

最后，开发人员需要确保客户可以轻松地访问他们的站点，而且用来导航站点的控件必须一致而直观。以前，开发人员不得不通过使用 JavaScript 或第三方控件来创建 Windows 样式的导航控件。某些勇敢者甚至试图为他们的.NET 项目编写自己的导航控件。然而，在.NET 2.0 Framework 中引入了站点导航控件，用来帮助开发人员在站点上创建一致且易用的导航元素。您希望采用在 Internet 上看到的导航控件(模仿 Windows 应用程序的控件种类)吗？现在它们已经包含在这个控件中了。您希望创建一个“breadcrumb”组件来显示用户在站点中的位置吗？现在这也是包含在这些控件中的默认行为的一部分了。开发人员现在能创建一个可被整个站点的导航控件所使用的 SiteMapDataSource 文件。更新该文件时，所有引用它的页面也会被更新。站点导航与.NET 2.0 Framework 的结合由来已久，第 6 章将介绍如何使用这些功能。

1.5 前提条件

无论是在.NET 上还是在任何其他开发平台上，学习优秀的 Web 设计原理并没有太多的前提条件。事实上，学习本书，至少表示了一种思考设计问题的意愿。不过从技术上来说，本书将对您的能力水平有一些要求。

首先，您应当相当熟悉 HTML 代码。如果下面的代码片段对您来说完全是陌生的，那么您可能难以理解本书所介绍的内容：

```
    <strong>Hello world!</strong>
    <br><br>
    <p>This is my first paragraph for my first page!</p>
```

显然，如果您熟悉 HTML、CSS 和优秀的 Web 设计的基本概念，就能看懂相关章节。类似地，如果您有开发.NET Web 应用程序(使用任意版本)的经验，就比较容易理解很多新增的.NET 概念。不过，没有什么概念是热诚的读者不能掌握的。

另一个相当重要的要求是至少粗略地了解.NET 知识。该知识是来自.NET 1.0 还是 1.1 与本书的讨论倒确实没有关系。不过，如果是完完全全的初学者，那么对于本书提出的一些编程概念的理解可能会有困难。本书不会详细解释什么是命名空间、什么是对象或者什么是页面类。在相关的时候，自然会详细介绍这些功能是如何与手边要介绍的主题进行交互的。然而，不要指望本书会给出.NET Framework 的全面细目分类。市面上有的是更全面地解释 Framework 的图书；本书专门介绍.NET 2.0 Framework 的一些令人激动的新功能。为了从本书中获得最大的益处，读者至少应熟悉面向对象编码，而且应有一定程度的.NET 经验。同样地，如果缺少这样的经验，虽然仍能从本书获得一些知识，但是肯定会有一些妨碍(只有您自己能指出有哪些妨碍)。

不过，为了贯彻这一思想——熟悉 Visual Studio 2005 确实有助于理解后面的章节提出的概念，大多数代码演示将用 Visual Studio 2005 执行。当引入 Visual Studio 的新功能时，当然会详细解释这个功能。然而，如果该概念不是 Visual Studio 2005 的新概念，而且不一定会妨碍正在进行的讨论，就不会为从来没有见过此接口的读者进行充分解释。

应当注意，对于 Visual Studio 2005 中的所有演示，您可以在 Visual Studio Web Developer 2005 Express Edition 中重新创建。事实上，作者所描述的大多数内容都是通过这个免费开发工具完成的。显然，如果您有购买完整版 Visual Studio 2005 IDE 所需的必要资金，当然应该购买。它有更多的功能，也更强大。不过如果您只是刚开始学习，或者只是一个业余爱好者，则用这个免费版本也可以完成本书的学习。

不过，学习本书最大的要求是对自己开发的 Web 站点在美学上更令人愉悦真正感兴趣。学习的愿望是不应低估的，如果您有这个愿望，就能利用本书的概念。这里提出的概念应当是每个 Web 开发人员构建应用程序的基础，而不仅仅适用于.NET 开发人员。本书会展示.NET 特有的优点，但是概念是通用的。如果您正在开发 Web 应用程序，应能从本书中学到一些有用的东西。

1.6　小结

aesthNETics 的精髓概念其实并不是新概念。Web 开发人员几年来一直在努力使他们的站点看起来漂亮，同时又要强大而有用。市面上有数千本讨论基础 Web 设计概念的图书。如果您在 Google 上搜索“ Web design concepts ”，将返回超过 100 000 000 条结果(字面上)。市面上有数百个应用程序声称它们能将普通人转变成专业的 Web 设计师，此外还有一些讲授 Web 设计基础的初级班。简而言之，我们根本不缺乏关于 Web 设计概念的资源。

缺乏的似乎是该信息的实际利用。很多开发人员(同样，这里也不仅是指.NET 开发人员)没有花时间使他们开发的站点更加美观。这些人在这里丢一个按钮控件，在那里丢几个文本框，甚至可能还在使用 10 年前为公司内网开发的色彩模式，然后就夸耀自己有了一个站点。

aesthNETics 所努力推行的是用.NET 2.0 Framework 的强大工具使站点在美学上更令人愉悦。本书重点关注的是.NET 开发团队。但是很多概念是通用的，因此其他开发人员也可能在本书的内容中发现一些有用的信息。本书主要面向.NET 开发人员，其中介绍的大多数工具是.NET 2.0 和 Visual Studio 2005 特有的。其他平台可能有类似的功能，但它们不是本书讨论的真正重点。本书的重点是使.NET 开发人员更加优秀。

所有程序员都有一些变得更优秀的理由，客户和管理层的期望就是最重要的理由之一。每个人都希望如今的开发人员是全能的。经典招聘广告的要求可能包括.NET 经验、数据库经验、nUnit 测试经验、项目管理经验，甚至还有图形图像处理经验。为了跟得上形势，开发人员需要不断地学习。他们需要拓展视野，挑战自我以学习本不属于自己擅长的领域的知识。对于认真的开发人员来说，这是学习 aesthNETics 的最佳理由。这样的挑战应当是学习新概念的实际吸引力。优秀开发人员与伟大开发人员之间的区别往往只在于不断学习的真正意愿。正如 Steve McConnell 在 *Code Complete 2nd Edition* 中所说的，“ 高级程序员的特征基本上与天才无关，而与亲自开发的承诺有关。”

那么，现在坐下来开始学习本书，力争使自己成为一名更优秀的.NET 开发人员吧。