“WPF...”入门

发布日期： 2006-12-26 | 更新日期： 2006-12-26

Laurence Moroney

Microsoft Corporation

适用于：

- “WPF/E”（代号）
- Microsoft Visual Studio 2005

摘要：本白皮书提供了深入“WPF/E”的高层次概述，并介绍了“WPF/E”与下一代 Web 应用程序开发堆栈中其他组件的协作情况。

目录

什么是“WPF/E”？

“WPF/E”是一种新 Web 呈现技术的代号，创建该技术的目的是使其能够在各种平台上运行。该技术支持创建丰富的、具有绚丽视觉效果的交互式体验，并且可以随处实现：无论是在浏览器内、在多个设备上还是在桌面操作系统（如 Apple Macintosh）中。Microsoft .NET Framework 3.0（Windows 编程基础结构）中的呈现技术 XAML（可扩展应用程序标记语言）遵循 WPF (Windows Presentation Foundation)，它是“WPF/E”呈现功能的基础。

本白皮书将逐步引导您了解“WPF/E”的基本情况，以及如何使用 Microsoft 的众多工具（包括 Microsoft Expression Graphic Designer、Microsoft Visual Studio 2005 和 XAML）来构建华丽的图形站点。首先，让我们了解一下有关“WPF/E”发展历程的背景信息，以及它在开发领域所处的位置。

返回页首

Web 开发的演变：转向 Web.Next

CERN 的 Tim Berners-Lee 发明现代 Web 时的初衷是将其作为允许在基于网络的系统上存储和链接静态文档的系统。之后的数年间，随着创新的发展和成熟，“活动”文档自然而然地成为了现代 Web 发展的新阶段，这些文档在收到访问请求时即会生成，文档中包含特定于时间或用户的信息。CGI 之类的技术成为了这一阶段的实现基础。随着时间的推移，在 Web 上生成文档的功能变得极为重要，技术上的发展也历经 CGI、Java、ASP，到达 ASP.NET 阶段。

在开发人员采用服务器开发模式并使用 Visual Studio 系列产品中的同类最佳工具快速开发高质量 Web 应用程序时所能拥有的能力方面，ASP.NET 树立了一个里程碑。

事实证明，用户体验是 Web 应用程序中的一大障碍，在这方面，技术上的限制使 Web 应用程序无法提供与使用本地数据的客户端应用程序同样丰富的用户体验。

XMLHttpRequest 对象（2000 年由 Microsoft 作为 Internet Explorer 5 的一部分发布）成为了异步 JavaScript 和 XML (AJAX) 技术的基础，该技术使 Web 应用程序能够对用户输入做出更加动态的响应，因为采用该技术时只会刷新网页的一小部分，并不需要重新加载所有内容。基于 AJAX 构建的创新型解决方案（如 Windows Live Local 映射）使 Web 应用程序更进一步，已经能够提供客户端式的用户体验。

“WPF/E”是应用程序开发人员和设计人员可以向其客户呈现的潜在用户体验丰富性的下一个发展阶段。它通过允许设计人员展现其创造力并以能够直接对 Web 产生影响的格式保存其工作来实现此目的。在过去，设计人员会使用提供了丰富输出功能的工具来设计网站和用户体验，但在实现能力上开发人员会受到 Web 平台的限制。在“WPF/E”模型中，设计人员可以构建其想要的用户体验，并将其表示为 XAML。开发人员随后可以使用“WPF/E”

运行时直接将该 XAML 并入到网页中。因此，两者可以比以往任何时候都更加紧密地合作，从而提供丰富的客户端用户体验。

由于 XAML 属于 XML，因此它是基于文本的，也就能够为这些丰富内容提供与防火墙兼容的、易于检查的说明。尽管可以使用其他技术（如 Java 小程序、ActiveX 和 Flash）来部署比 DHTML/CSS/JavaScript 更丰富的内容，但它们都会向浏览器发送二进制内容，这种内容难以进行安全性审核，更不用说还有更新上的困难，因为进行任何更改后都必须重新安装整个应用程序，而这并不是最友好的用户体验，并可能导致页面停滞。使用“WPF/E”时，如果需要对丰富内容进行更改，服务器端会生成新的 XAML 文件；下次用户浏览到该页面时，将会下载该 XAML 并更新体验，而不需要进行任何重新安装。

“WPF/E”的核心是浏览器增强模块，其作用是呈现 XAML 并在浏览器表面上绘制所生成的图形。它的下载体积较小（不到 2 MB），可以在用户点击包含“WPF/E”内容的站点时进行安装。该模块会向 JavaScript 开发人员公开 XAML 页面的底层框架，以便能够实现在页面级与内容进行交互，开发人员于是就可以进行自己的工作，例如：编写事件处理程序或使用 JavaScript 代码来处理 XAML 页面内容。

不过，理论方面的探讨已经够多的了！我们还是动动手，来看一看我们的第一个“WPF/E”项目。

返回页首

构建简单的“WPF/E”应用程序

我们先来看一看 Microsoft Expression Graphic Designer，该程序用于创建一个供“WPF/E”使用的非常简单的图形。图 1 显示的是创建过程中的产品。

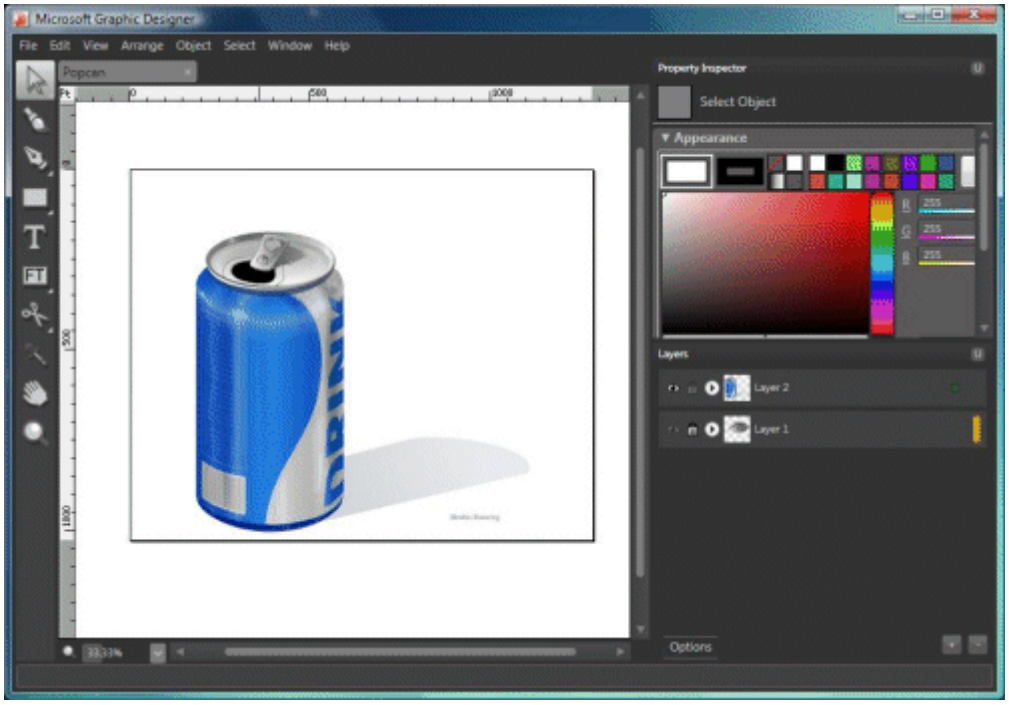

图 1. Expression Graphic Designer 工具

Expression Graphic Designer 是 Microsoft 开发的一个新工具，它将基于矢量和基于像素的图形设计工具的优点集于一身，设计人员可以利用该工具探索新的创意目标。可以使用该工具并入来自其他应用程序的图形以及将设计元素导出到各种软件工具（其中包括 XAML for WPF 和“WPF/E”）中。

返回页首

准备图形设计文件以供“WPF/E”使用

使用 Expression Graphic Designer 打开 Popcan.xpr 文件。可以在 Program Files\Microsoft Expression\Design Beta 1\Samples 目录中找到该文件。该文件就是图 1 中所使用的图形文件。

打开该文件后，您会发现如果按像素度量其尺寸相当大；使用垂直和水平标尺可以确定这一点，其实际尺寸为 1280 × 1024 像素。图 2 显示的是表示图像宽度的水平标尺。

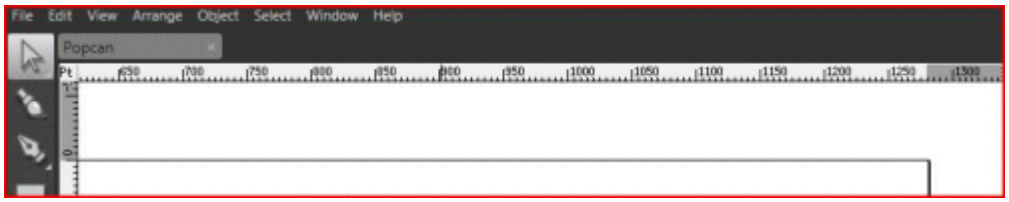

图 2. Expression Graphic Designer 中的水平标尺

也可以在“Document Size”（文档大小）对话框（在“File”（文件）中选择“Document Size”（文档大小））确定这一点，如图 3 中所示。

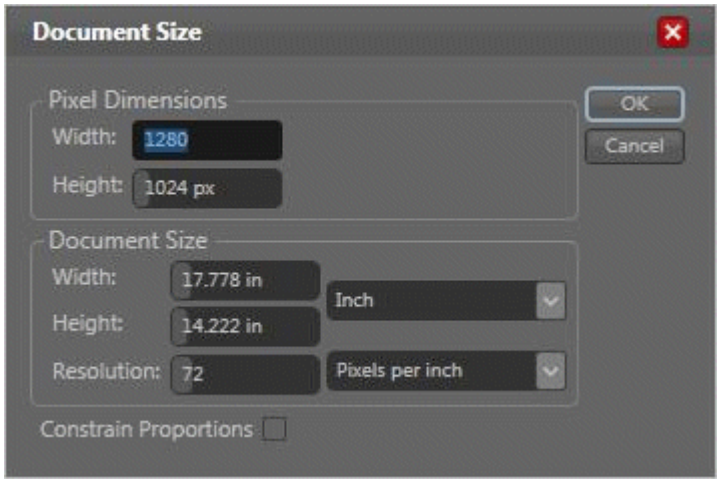

图 3.“Document Size”（文档大小）对话框

使用“Document Size”（文档大小）对话框将图像尺寸更改为 300 × 150 像素。更改方法是：在“Document Size”（文档大小）对话框的“Width”（宽度）文本框中键入 300，在其“Height”（高度）文本框中键入 150，如图 4 中所示。

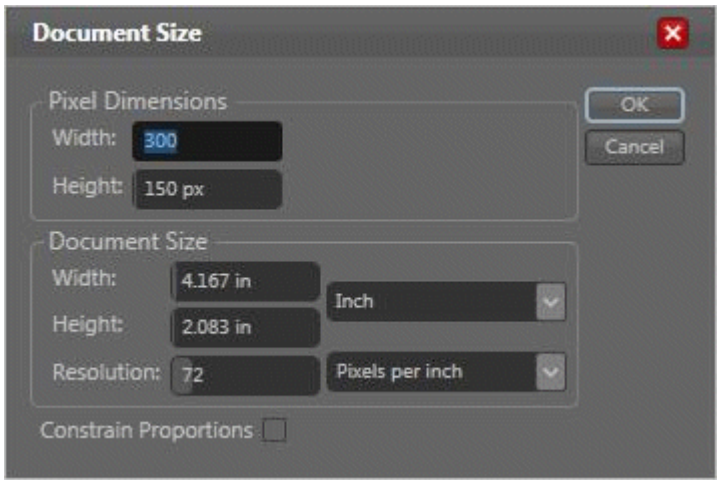

图 4. 将尺寸设置为 300 × 150 像素

单击该对话框中的"OK"（确定）时，将会调整图像的大小。此时，易拉罐的外观过度失真，显得又短又粗，如图 5 中所示。

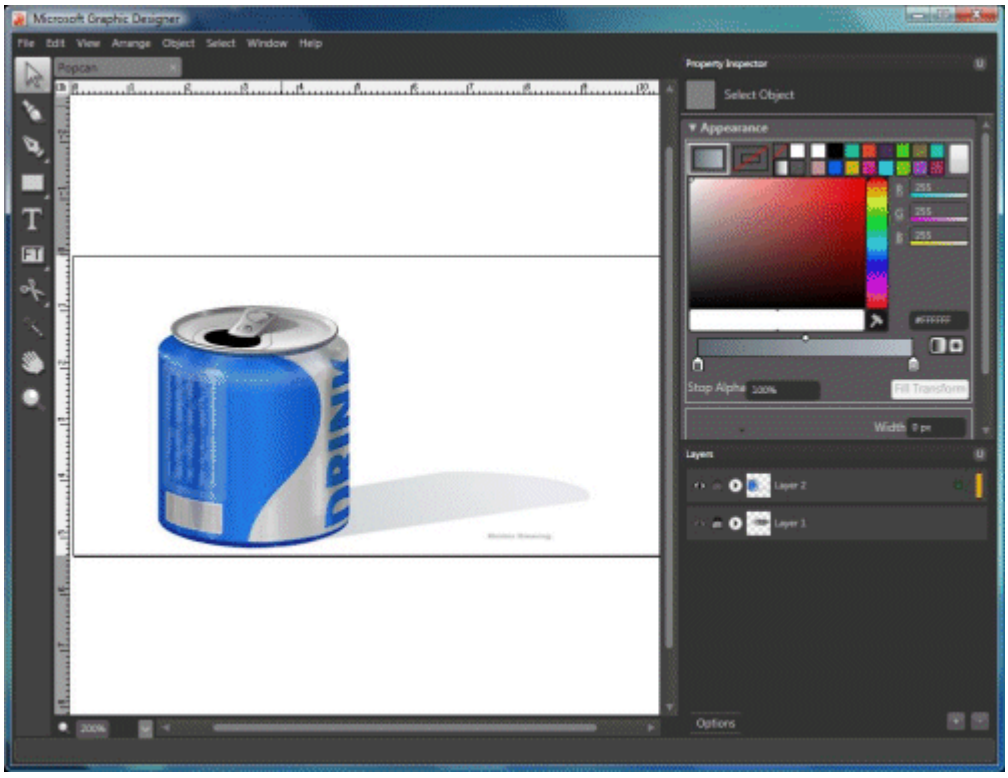

图 5. 大小调整为 300 × 150 像素的文档

由于绘图是基于矢量的绘图，因此可以不失真地调整大小。处于设计图面中时，按 CTRL+A 选择绘图中的所有元素。将出现一个绿色轮廓线，并显示选定的元素，如图 6 中所示。

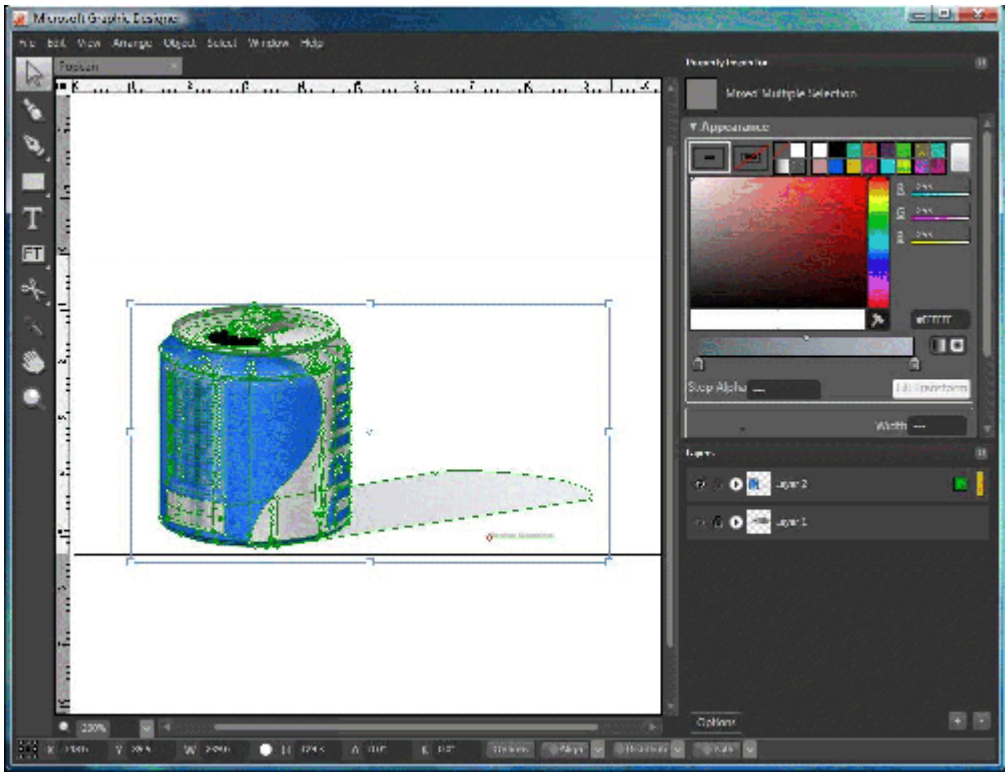

图 6. 选定的所有图像元素

此时可以在边框内四处拖动图像，并拖动它的各个角来调整图像大小，直到长宽比满意为止。图 7 显示的是易拉罐经过大小调整后看上去更加真实，并且放置在了文档的左上角。

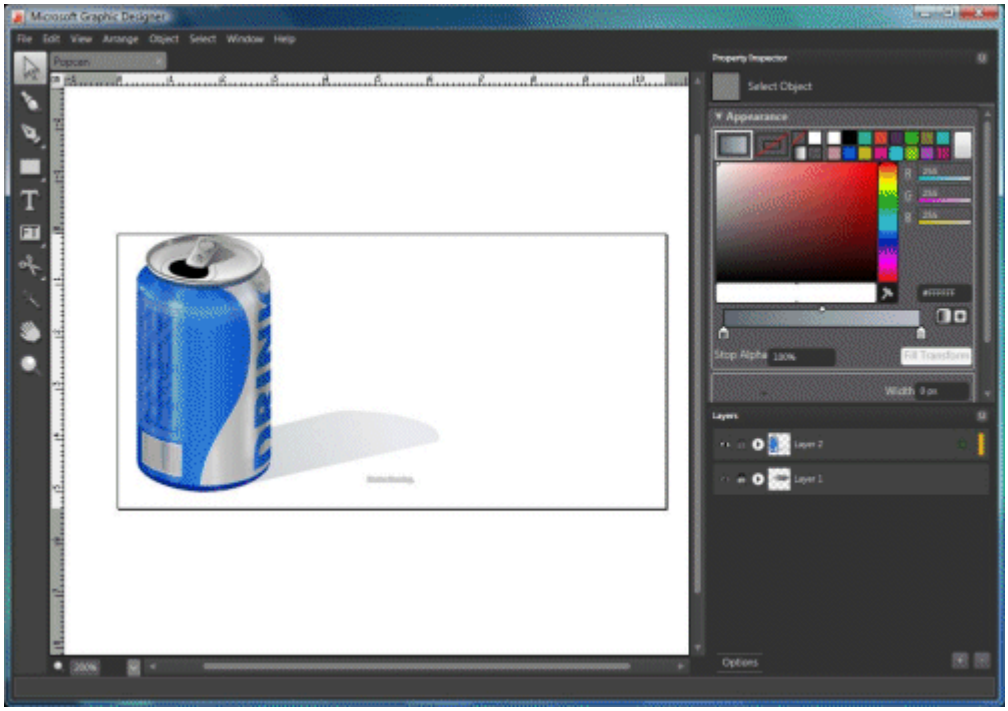

图 7. 调整易拉罐大小以得到更满意的长宽比

此时便可将该图像导出为“WPF/E”XAML 图像。

操作方法如下：依次单击“File”（文件）、“Export”（导出）、“XAML”；或使用 CTRL+ALT+X。该操作会调出“Common Save”（普通保存）对话框，可以使用它来指定想要放置 XAML 文件的位置。调用 Popcan.xaml 文件并选择“Save”（保存）。将出现“Xaml Export”（Xaml 导出）对话框。可以通过该对话框对导出文件做一些进一步指定，其中包括指定想要导出兼容“WPF/E”的 XAML，如图 8 中所示。

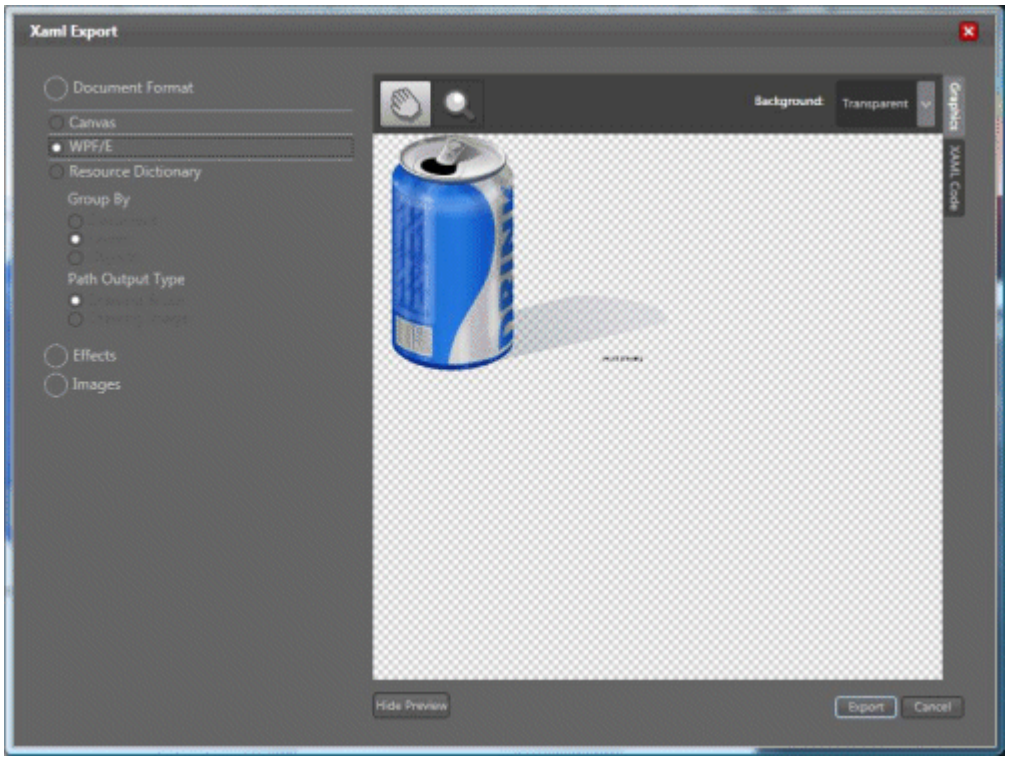

图 8. 导出“WPF/E”XAML

屏幕的右侧有一个“Preview”（预览）窗格，可在其中通过缩放和平移来检查图像，以确保图像外观正常，然后再进行后续操作。右侧的选项卡用于在检查图像和检查表示图像的 XAML 代码之间进行切换。如果是第一次接触 XAML，最好看一看“XAML Code”（XAML 代码）选项卡，以便将代码与代码应用到图像后的结果联系起来。右侧的下拉选项用于指定导出属性，其中包括应如何处理光栅化图像，即应该将它们矢量化，还是直接输出到特定目录。目前可以保持默认设置不变，但请确保选择了“Document Format”（文档格式）下的“WPF/E”，如图 8 中所示。

一切就绪后，请单击“Export”（导出）按钮，“WPF/E”XAML 代码将写入磁盘。稍后，将在 Visual Studio 2005 Web 项目中使用该代码。在下一步中，您将了解如何设置 Web 项目，以及如何准备该项目以供 XAML 使用。

↑返回页首

使用 Visual Studio 2005 构建“WPF/E”项目

“WPF/E”SDK 中包括用于 Visual Studio 2005 的模板，可以通过它构建“WPF/E”项目。为能在 Visual Studio 2005 中使用该模板，首先需要将下列加载项下载到 IDE。

首先，需要获得支持 Web 应用程序项目所需的更新，可以从 Microsoft 下载和安装该更新。

成功安装该更新后，就可以安装 Visual Studio 2005 Web 应用程序项目加载项本身了。单击下列链接来下载和安装 WebApplicationProjectSetup.msi。“WPF/E”SDK 下载中包括一个压缩文件，名为“WPF/E”JSApplication.zip。在 \Program Files\Microsoft Visual Studio 8\Common7\IDE\ProjectTemplates\CSharp 目录中创建一个“WPF/E”目录，然后将该压缩文件复制到该目录中。然后，在确定开发环境已关闭的情况下，通过命令提示符运行 devenv.exe/setup。

Visual Studio 此时已做好了创建“WPF/E”项目的准备。创建步骤如下：启动 Visual Studio IDE，依次单击“文件”、“新建项目”。您会在可用项目类型中看到“WPF/E”，并会看到用于创建新“WPF/E”应用程序的“WPF/E”JavaScript 应用程序模板，如图 9 中所示。

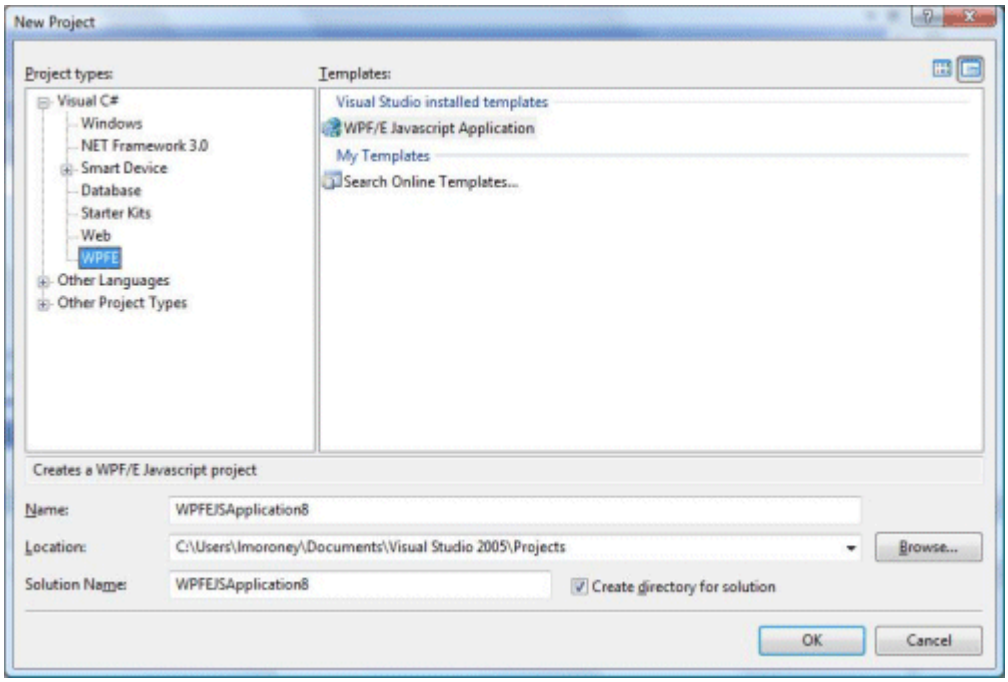

图 9. 使用 Visual Studio 模板

该操作会创建一个新项目，其中包含单个 HTML 页面和一个表示简单按钮的 XAML 文件。最好浏览一遍 HTML，以了解它如何使用 JavaScript 来设置“WPF/E”控件（稍后有关于此内容的详细说明），以及了解 XAML 如何将可以在 JavaScript 中捕获和处理的事件进行公开。

返回页首

准备 Visual Studio 2005 项目以供“WPF/E”使用

尽管可以使用上一部分中介绍的模板创建应用程序，但最好也了解一下“WPF/E”的工作方式和提供方式，对于其提供方式，您可以了解到将“WPF/E”模块部署到浏览器时需要对现有网站进行的配置竟是如此简单。在本部分中，您将了解如何手动将网站配置为“WPF/E”站点。

编写本白皮书时，我使用的是 Visual Studio 2005 和“Orcas”CTP Preview。您不需要使用“Orcas”，但如果已安装了它，则会给您键入 XAML 代码带来稍许方便，因为它具有 IntelliSense，而且可以在“Cider”XAML 设计器中预览完成的 XAML。不过，一定要注意 Cider 是专为 WPF（而不是“WPF/E”）而设计，因此此时不应使用它来开发面向“WPF/E”的 XAML 代码。

因此，作为第一个步骤，请启动 Visual Studio 2005，然后使用“文件”下的“新建网站”对话框创建新网站，如图 10 中所示。

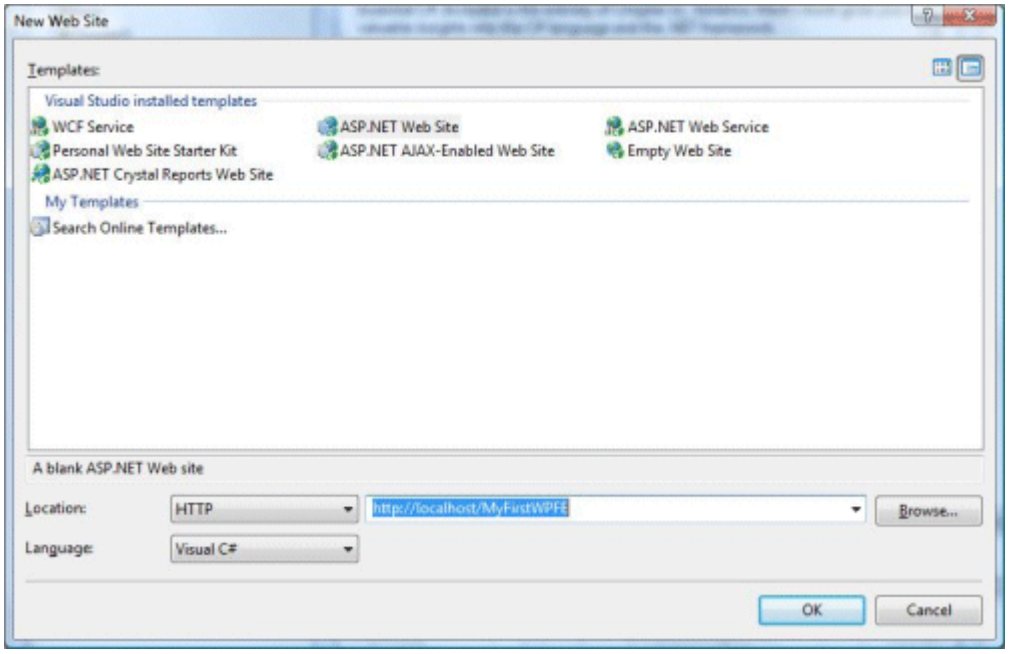

图 10. 在 Visual Studio 2005 中创建新网站

该操作会创建一个新解决方案，其中包含一个简单的 Default.aspx ASP.NET 页面，执行该解决方案时，会生成将在浏览器中呈现的 HTML。此步骤为您的第一个“WPF/E”页面奠定了基础。

接下来，需要导入用作插件的部署文件。操作方法是：先右键单击“解决方案资源管理器”中的项目 (http://localhost/MyFirstWPFE) 并选择“新建文件夹”，以在网站上创建一个 Bin 目录。将新文件夹重命名为“Install”，然后将“WPF/E”下载中的下列文件添加到这个新目录中：install.msi、MozillaControl1712.exe、WPFE.dmg、xcpctrl.cab 和 xcpctrl.xpi。（可以从 Windows 资源管理器拖动它们，然后将它们直接放置在 Visual Studio 2005 的“解决方案资源管理器”中的 Bin 目录内。）也可以从使用该 Visual Studio 2005 模板开发的任何项目（如上一部分中的项目）中获取上述文件。

接下来，应将 aghost.js 文件添加到您的网站中。同样可以通过从 Windows 资源管理器拖动该文件，然后将其放置在解决方案中来完成此操作。最后，在您的解决方案中创建另一个名为 XAML 的新文件夹，然后将先前创建的 Popcan.xaml 文件复制到该文件夹中。

↑返回页首

编辑网页以呈现“WPF/E”内容

现在，您将了解如何编辑网页以使其处理嵌入的“WPF/E”内容。这是非常简单的，因为只需要添加单个 JavaScript 引用和一个对 aghost.js 中所定义的 JavaScript 对象的调用。JavaScript 库和“WPF/E”安装程序会完成其余的工作。

首先，编辑页面，使其看起来就像列表 1 中的样子。此处所做的全部工作就是将“无标题页面”改为现在的标题、添加对 aghost.js 的 JavaScript 引用，以及向页体中添加一个新的 Div，其中包含对 agHost 对象的调用。在列表 1 中突出显示了这些工作。

列表 1. default.aspx HTML 代码

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"

Inherits="_Default" %>



<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">



<html  >

<head id="Head1" runat="server">

    <title>My first "WPF/E" Page!</title>

   <script type="text/javascript" src="agHost.js "></script>

</head>

<body>

    <form id="form1" runat="server">



    <div id="AgControl1Host">

    <script type="text/javascript">

    new agHost("AgControl1Host", "AgControl1", "400",

               "400", "#00000000", null, "xaml/Popcan.xaml",
```

```
                "True", "30", null);

        </script>

    </div>

    </form>

</body>

</html>
```

此刻，对“新的 AgHost”的 JavaScript 调用可能看起来有点晦涩，但是不要担心：稍后，您将对此进行深入了解。在下面各部分中，您将首先看到在 Internet Explorer 和 Mozilla Firefox 中运行此应用程序的情景。此后，您将深入了解控制“WPF/E”的 JavaScript 对象，以及如何进行一些编辑来向图形中添加动画，以使其变得生动。现在即可运行应用程序并让 ASP.NET 生成包含“WPF/E”内容的页面。可在 Visual Studio 2005 中按 F5 来启动该应用程序。首次运行应用程序时，浏览器会检测是否安装了“WPF/E”。如果未安装，则在页面启动时，将会打开一个对话框，询问您是否要安装“WPF/E”模块。附录 I 针对在 Windows 上运行的 Internet Explorer 和 Mozilla Firefox 详细介绍了相应的安装体验。

返回页首

了解 JavaScript

以下是使用“WPF/E”将此 XAML 嵌入页面的 JavaScript 代码。

```
new agHost("AgControl1Host", "AgControl1", "400", "400", "#00000000",

null, "xaml/Popcan.xaml",

          "True", "30", null);
```

此代码会创建一个新的“WPF/E”插件实例，其间使用了可用于创建该实例的参数列表。这些参数（依次）为：

hostElementID：这是将在其中托管插件控件的 HTML 元素的名称。因此，如果在包含该插件的页面上有一个 ，请为该 命名，并在此处使用该名称。

controlID：这是插件自身的 ID。

height：这是控件的所需高度（以像素为单位）。

width：这是控件的所需宽度（以像素为单位）。

backgroundColor：这是控件的所需背景色。

sourceElement：这是包含控件 XAML 的页面元素的名称。这是一种为 <script> 元素中所含控件配置 XAML 的方法（其中 XAML 在页面上）。若使用此方法，请将 <script> 元素的 ID 置于此参数中，而后控件将会从该处选取它。

sourceURL：这是外部 XAML 文件的位置。

isWindowless：这是一个布尔参数。如果要使“WPF/E”控件成为无窗口的，请将它设置为 True。这意味着在 HTML 页面中，“WPF/E”控件将与 HTML 嵌在一起，例如，如果将“WPF/E”控件设置为透明的，则 HTML 将显示在它的“后面”。如果它不是无窗口的（换句话说，如果将此参数设置为 False），则“WPF/E”内容将显示在页面的一个截然不同的区域中，而 HTML 标记将环绕在它的周围。

maxFrameRate：这是一个指定最大帧频的数字，“WPF/E”将按该帧频呈现动画内容。

loadHandler：这是在加载控件时要触发的页面上的一个 <script> 元素的名称。

errorHandler：这是当控件上出现错误时要触发的页面上的一个 <script> 元素的名称。

使用这些参数创建新的 agHost 控件实例时，将会对 agHost.js 进行调用（因此，在 HTML 中需有一个对此文件的脚本引用），这会生成包容控件的 <object> 标记。因此，在先前的情况下，如果使用的是 Internet Explorer，则会生成以下 <object> 标记。

```
<object id="AgCotnrol1" height="400" width="400"

   Codebase="install/xcpctrl.cab"

   classid="CLSID:32C73088-76AE-40F7-AC40-81F62CB2C1DA">

   <param name="Source" value="xaml/Popcan.xaml" />

   <param name="MaxFrameRate" value="30" />

   <param name="BackgroundColor" value="#00000000" />

   <param name="WindowlessMode" value="True" />

</object>
```

如果愿意，始终都可以在页面中直接使用此代码，但使用该 JavaScript 对象要繁琐得多，然而不用在 HTML 页面中过多注重 <object> 标记的实现细节。

返回页首

编辑 XAML 以添加文本

您可以直接从 Visual Studio 2005 内编辑 XAML，向其中添加某些内容。在本例中，将向 XAML 添加一个 TextBlock 控件，以包含一些诸如“大家好”之类的基本文本。

在 Visual Studio 2005 中打开 XAML 文件。如果已安装了“Orcas”CTP，则可以在“Cider”编辑器中预览 XAML，如图 11 所示。请注意，不能使用该工具箱向“WPF/E”XAML 添加控件，因为“Cider”设计器只适用于 WPF 应用程序。否则，只需将 XML 编辑器打开即可。

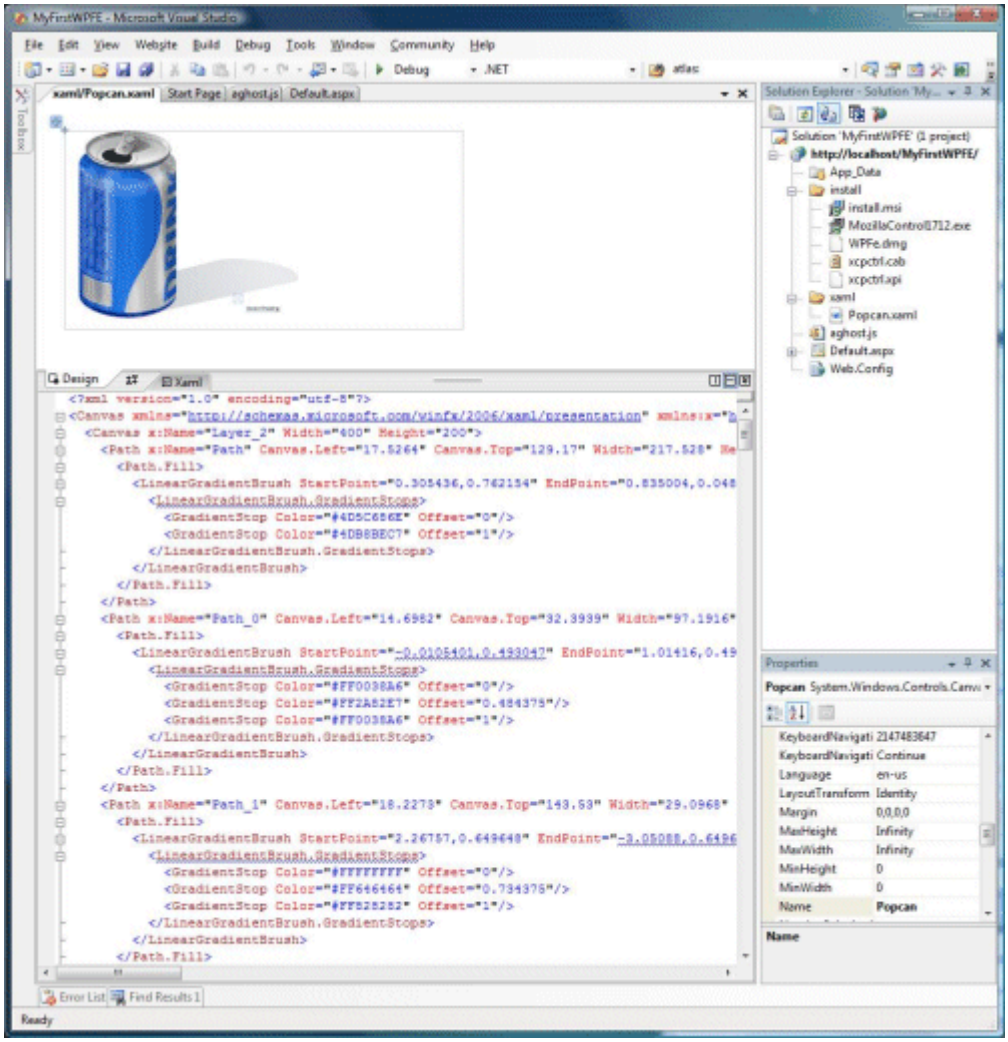

图 11. 使用 Visual Studio 2005 编辑 XAML

要向 XAML 添加文本块，请滚动到底部并在结束 </Canvas> 标记的紧前面添加以下代码。

```
<Canvas x:Name="Layer_3">

 <TextBlock x:Name="MyLink" Canvas.Left="150" Canvas.Top="0"

     FontFamily="Verdana" FontSize="18"

     Width="250" TextWrapping="Wrap">

         Drink some of this yummy WPFE soda!

         Much better than a Hello, World App, right?

  </TextBlock>
```

```
</Canvas>
```

可在图 12 中看到此代码在网页上的效果。

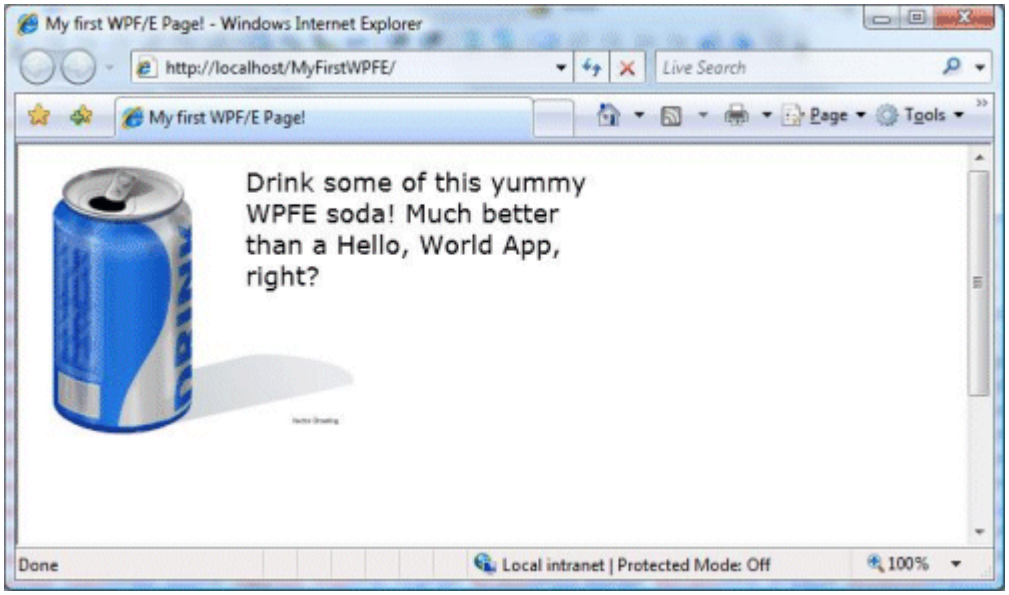

图 12. 包含文本块的 XAML

返回页首

编辑 XAML 以实现简单动画

XAML 指定了动画情节。在本例中，将向“WPF/E”页面添加一个简单动画，它会使文本在屏幕的顶部与底部之间来回弹跳。为此，需添加一个在加载 Canvas 时触发的 EventTrigger。此事件触发器可以包含一项操作。指定的操作将开始包含“DoubleAnimation”的情节。这种类型的动画用于操控开始于“From”而结束于“To”的整个范围内的数字。可以将其设置为永远重复和自动颠倒自身。因此，您可以指定使“弹跳”文本的顶部从 0 跳动到指定的量值并在动画完成时自动颠倒它，而后一直重复下去。下面是为了达到此效果而设计的情节。

```
<Storyboard Storyboard.TargetName="MyLink"

          Storyboard.TargetProperty="(Canvas.Top)" >

  <DoubleAnimation AutoReverse="True" RepeatBehavior="Forever"

                BeginTime="0" Duration="0:0:1"

                From="0" To="90">

  </DoubleAnimation>

</Storyboard>
```

下面是完整的 Canvas 规范，其中包含文本以及用于触发动画的触发器。

```
<Canvas x:Name="Layer_3">
  <Canvas.Triggers>
    <EventTrigger RoutedEvent="Canvas.Loaded">
      <EventTrigger.Actions>
        <BeginStoryboard>
          <Storyboard Storyboard.TargetName="MyLink"
                      Storyboard.TargetProperty="(Canvas.Top)" >
            <DoubleAnimation AutoReverse="True" RepeatBehavior="Forever"
                             BeginTime="0" Duration="0:0:1"
                             From="0" To="90">
            </DoubleAnimation>
          </Storyboard>
        </BeginStoryboard>
      </EventTrigger.Actions>
    </EventTrigger>
  </Canvas.Triggers>
  <TextBlock x:Name="MyLink" Canvas.Left="150" Canvas.Top="0"
             FontFamily="Verdana" FontSize="18" Width="250"
             TextWrapping="Wrap">
              Drink some of this yummy "WPF/E" soda!
              Much better than a Hello, World App, right?
  </TextBlock>
</Canvas>
```

现在，当您查看工作中的网站时，该文本会在右侧上下"弹跳"。

返回页首

编辑 XAML 以实现简单交互

“WPF/E”允许 JavaScript 译码器与 XAML 交互。在本例中，您将看到如何处理用户与 XAML 文件交互时所引发的事件。上个例子中的“弹跳”文本将变为一个超链接，随后可用它链接到其他页面。

为实现这个目的，需在 TextBlock 声明中指定事件处理程序。您将要捕获鼠标点击，因此需使用 MouseLeftButtonDown 事件处理程序指定将要处理点击的 JavaScript 函数。该文本块的样子如下，其中含有针对一个称为 hyperlink_MouseLeftButtonDown 的 JavaScript 处理程序的声明。

```
<TextBlock MouseLeftButtonDown="javascript:hyperlink_MouseLeftButtonDown"
   x:Name="MyLink" Canvas.Left="150" Canvas.Top="0"
   FontFamily="Verdana" FontSize="18"
   Width="250" TextWrapping="Wrap">Channel 9 is cool!</TextBlock>
```

现在即可在宿主页面上（或在页面引用的外部库中）使用 JavaScript 处理此事件，如下所示。

```
   <script language="javascript">
     function hyperlink_MouseLeftButtonDown(sender, args) {
         window.open("http://channel9.msdn.com");
      }
   </script>
```

现在，当运行页面时，将会显示一个移动的动画超链接，单击该超链接可浏览到 MSDN 第 9 频道站点！

返回页首

向混合体中添加媒体

“WPF/E”给 Web 开发人员带来了最佳类型的 Windows 媒体。使用 XAML 中的 <MediaElement> 标记很容易就能向“WPF/E”应用程序添加媒体内容，如下所示。

```
<Canvas x:Name="Layer_4">
   <MediaElement x:Name="VideoLayer" Source="Butterfly.wmv"
              Canvas.Top="0" Canvas.Left="130"
              Height="200" Width="200">
   </MediaElement>
```

```
</Canvas>
```

请注意，您的根网站中需有 Butterfly.wmv 文件，此代码才能奏效。此 WMV 文件是 Windows Vista 附带的“示例视频”之一，可在您的“文档”文件夹中找到。如果您没有此文件，可用任意的 WMV 文件代替它。

MediaElement 公开了可用于开始、停止、暂停或搜寻视频的方法。有关这方面的详细信息，请查看“WPF/E”文档和示例。

可在图 13 中看到带有嵌入视频的“WPF/E”应用程序。

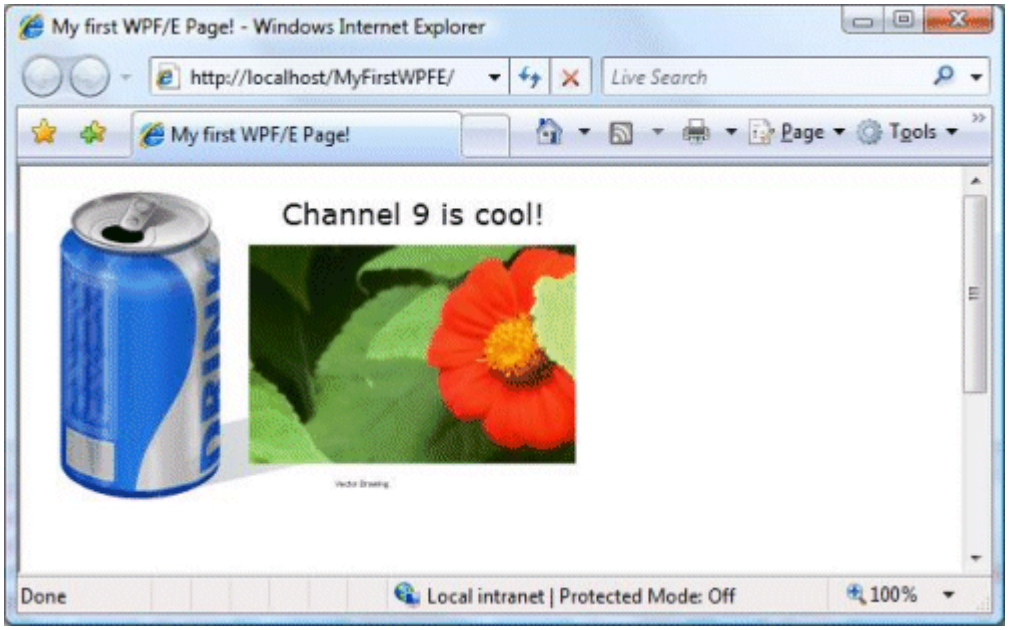

图 13. 带有嵌入视频的“WPF/E”

返回页首

结束语

在本白皮书中，向您提供了深入“WPF/E”的高层次概述，并介绍了“WPF/E”与下一代 Web 应用程序开发堆栈中其他组件的协作情况。您看到了 XAML 作为一个凝聚体是如何将设计人员的规范与开发人员的工具和面向用户的交付结合在一起的。您大体了解了 Expression Graphic Designer，并学习了如何用它来为网页定义图形以及如何将这些图形导出到“WPF/E”XAML。随后，您将此 XAML 导入到一个 Visual Studio 2005 Web 项目中，并且看到为了从设计器呈现“WPF/E”XAML，应如何对其进行配置以便将 XAML 运行库部署到 Internet Explorer 和 Firefox。最后，您深入了解了如何操控 XAML 以添加动画、交互和媒体。

您在本文中所了解的内容只触及到“WPF/E”可能功能的皮毛。在这项技术中有诸多功能，您现在就可以使用这项技术开始构建下一个 Web。这会是一个有趣的旅程，快来体验吧！

返回页首

附录 I：安装体验

为了查看含有“WPF/E”内容的网页，需要使用“WPF/E”增强您的浏览器。为此，您既可以使用某种对象标记方法提示用户进行安装，也可以检测是否使用“WPF/E”增强了浏览器，并将用户引到相应的下载页面以获取该模块：

- 对于 PC：http://go.microsoft.com/fwlink/?LinkID=77792&clcid=0x409
- 对于 Macintosh：http://go.microsoft.com/fwlink/?LinkID=77793&clcid=0x409

就我们的应用程序而言，我们将使用一个对象标记来提示用户安装相应的“WPF/E”模块（如果它尚未安装）。这样，在首次运行应用程序时，浏览器将检测是否安装了“WPF/E”。如果此时未安装，则在页面启动时，将会打开一个对话框，询问您是否要安装“WPF/E”。以下各部分分别详细介绍了使用 Microsoft Internet Explorer 和 Mozilla Firefox 按此工作流程操作时的用户体验。

在 Internet Explorer 中运行页面

将会出现图 14 中所示的“Internet Explorer 安全警告”对话框。如果选择“安装”，则会按以下各部分详述的那样下载并安装插件。

图 14.“WPF/E”安装时的浏览器安全警告

选择“安装”，然后将会启动安装程序向导，如图 15 所示。

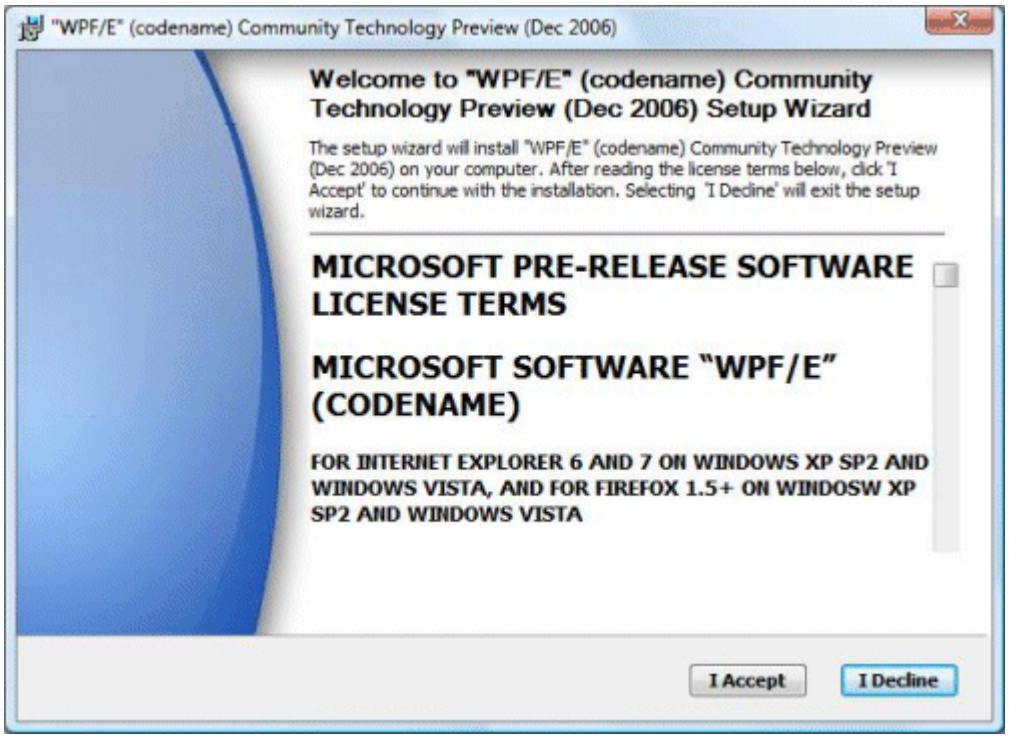

图 15. 启动"WPF/E"安装程序

认真阅读最终用户许可协议，如果您愿意继续，请选择"I Accept"（我接受）。随后便会开始进行 WPFE 安装，如图 16 所示。

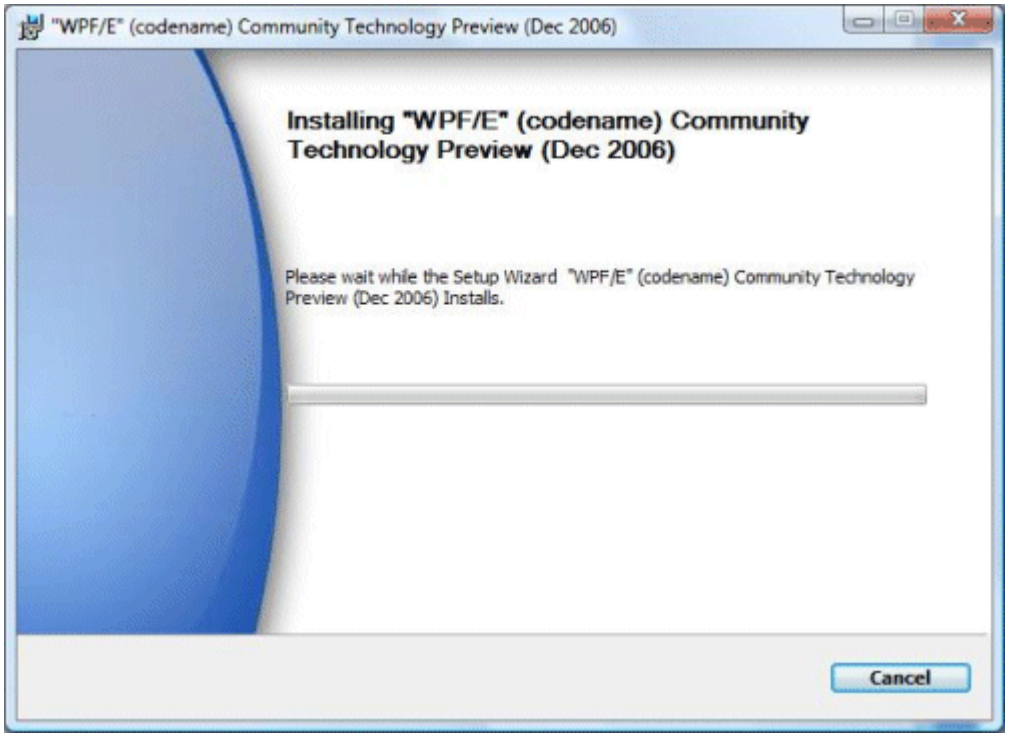

图 16. 安装"WPF/E"

当安装成功完成时，您会看到"Installation Complete"（安装完成）对话框，如图 17 所示。

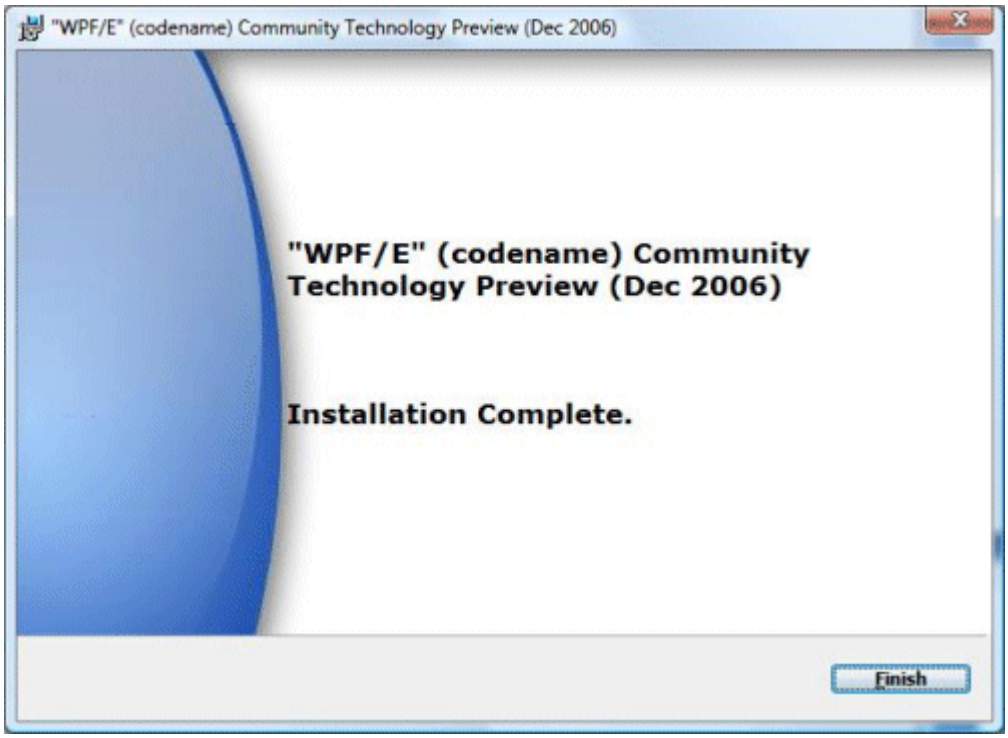

图 17. 安装 WPF/E

按“Finish”（完成），然后即可开始使用了。实际上，XAML 将会立即呈现，如图 18 所示。

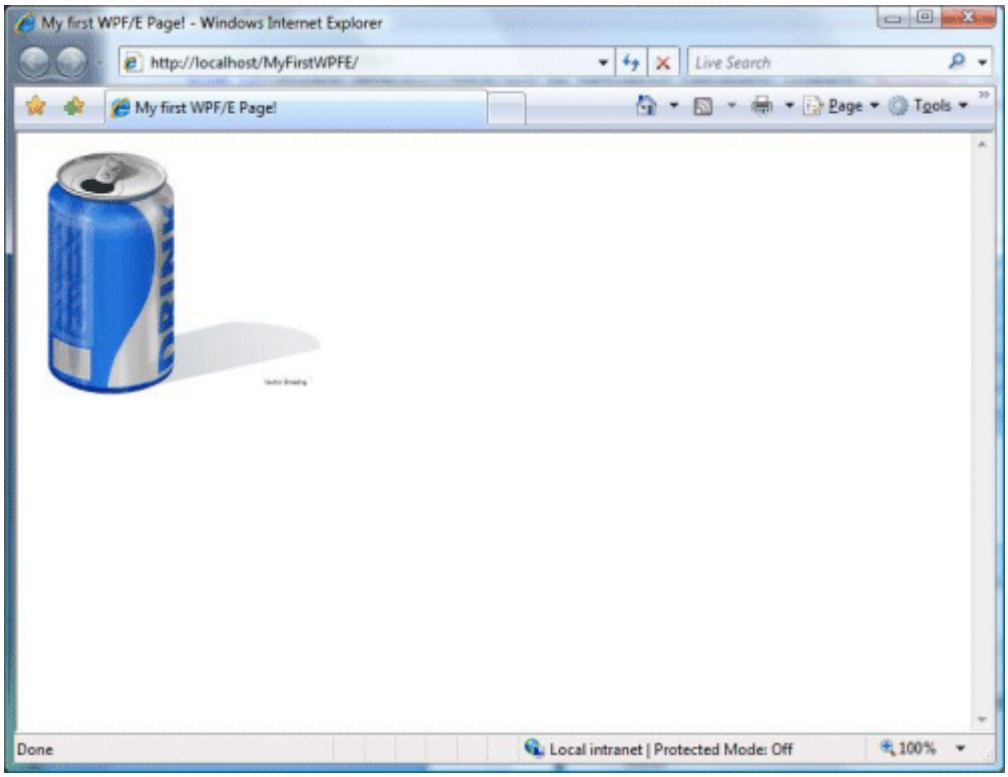

图 18. 在浏览器中呈现的 XAML

在下一部分，您将看到在 Mozilla Firefox 浏览器上安装该插件的情景。

在 Mozilla Firefox 中运行页面

在上一部分，您看到了“WPF/E”是如何增强 Internet Explorer 的。但“WPF/E”是一项多浏览器、多平台技术。在本部分，您将看到它在运行于 Microsoft Windows Vista 上的 Mozilla Firefox 中是如何工作的。

如果未安装“WPF/E”，则会在运行 Firefox 时看到类似于图 19 中的内容。

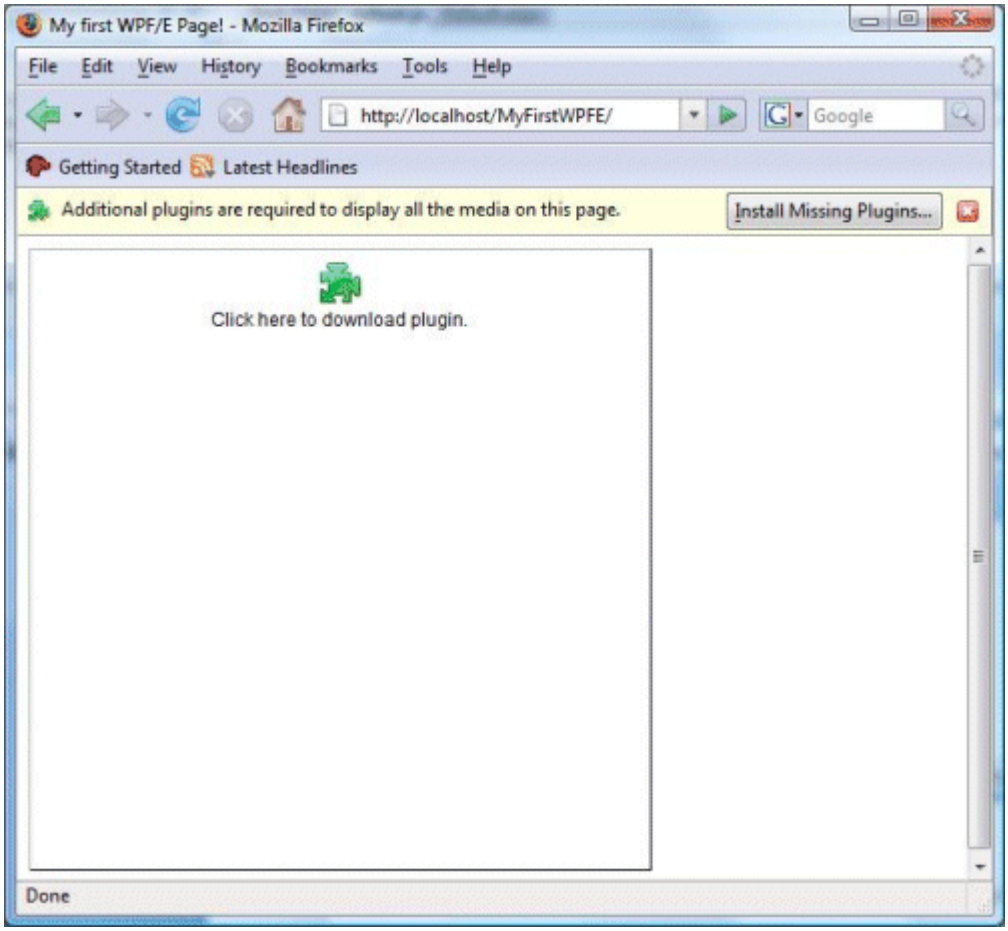

图 19. 在 Firefox 上运行“WPF/E”

单击屏幕右上方的“安装缺少的插件”。Firefox 将搜索与“WPF/E”类型 (application/xcp-plugin) 匹配的已知插件，但不会找到任何插件。这是因为“WPF/E”还不是已发行的产品，因而该插件尚未发布到“Firefox 插件查找器服务目录”（一旦发行了该产品，即会将其发布至此）。请参见图 20。

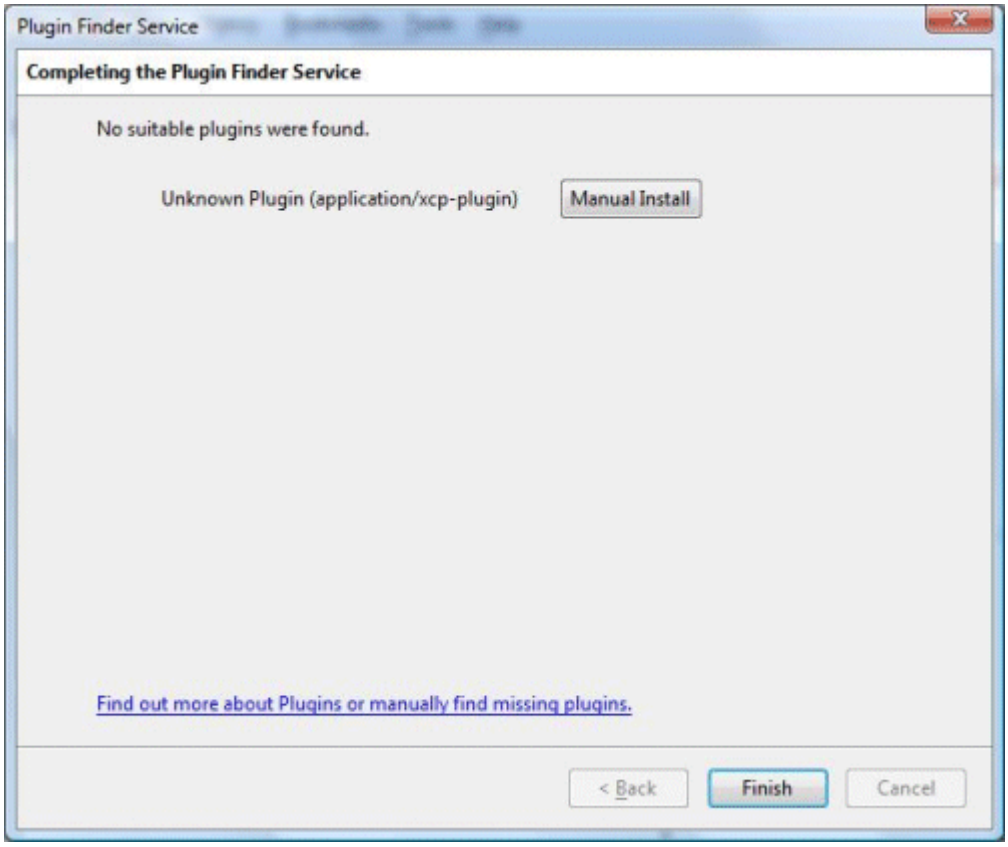

图 20. Firefox 插件查找器服务

但是，可以从“插件查找器服务”对话框执行手动安装。这会将您引领到一个 URL，其中包含用以安装“WPF/E”运行库的可启动 MSI 文件。自此以后的安装体验与 Internet Explorer 的安装体验完全相同，如上一部分所述。

完成后，请执行您的页面；XAML 将会由“WPF/E”在 Mozilla Firefox 中进行呈现，如图 21 所示。

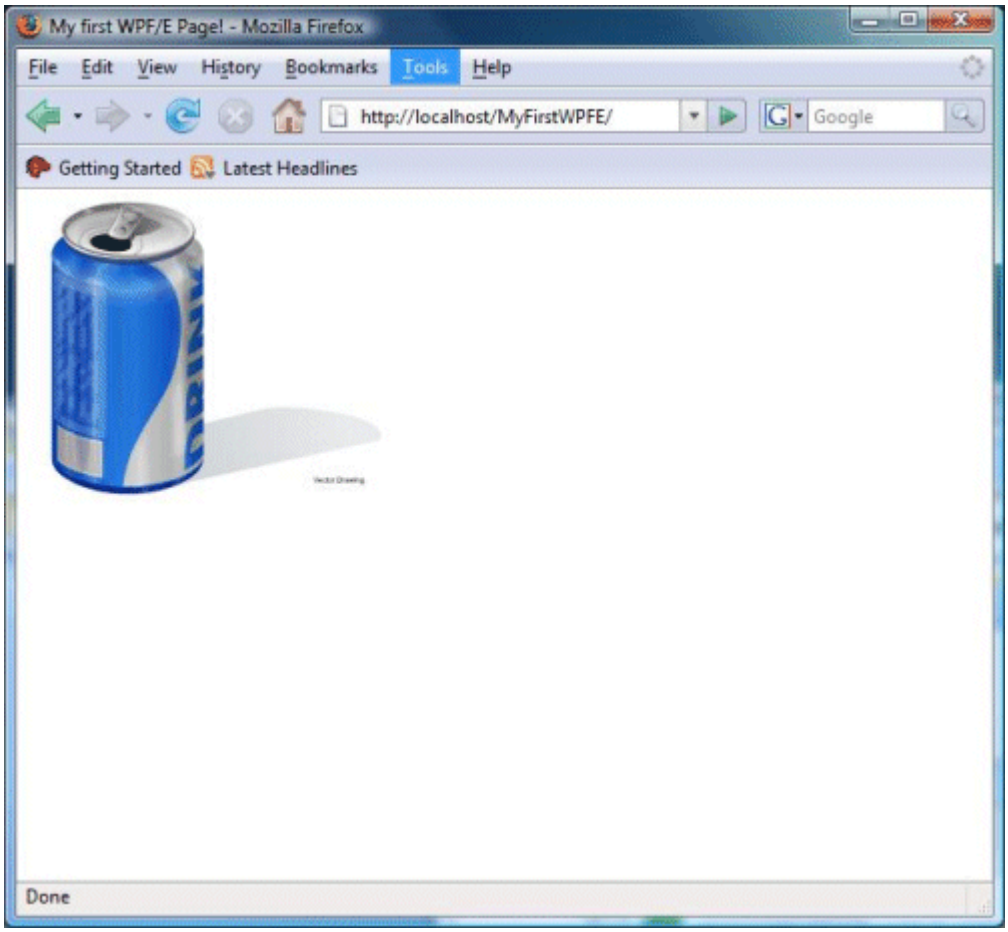

图 21. 在 Mozilla Firefox 中呈现 XAML

↑返回页首

附录 II：配置服务器 MIME 类型

配置 IIS 6.x

为交付“WPF/E”应用程序所需的 MIME 类型配置 IIS 非常简单。如果您得到的是空白页，而不是已呈现的 XAML，则可能是您尚未在 IIS 中将 XAML 配置为可识别的 XML 类型。为此，请调出网站的“属性”对话框，然后选择“HTTP 头”选项卡，如图 22 所示。

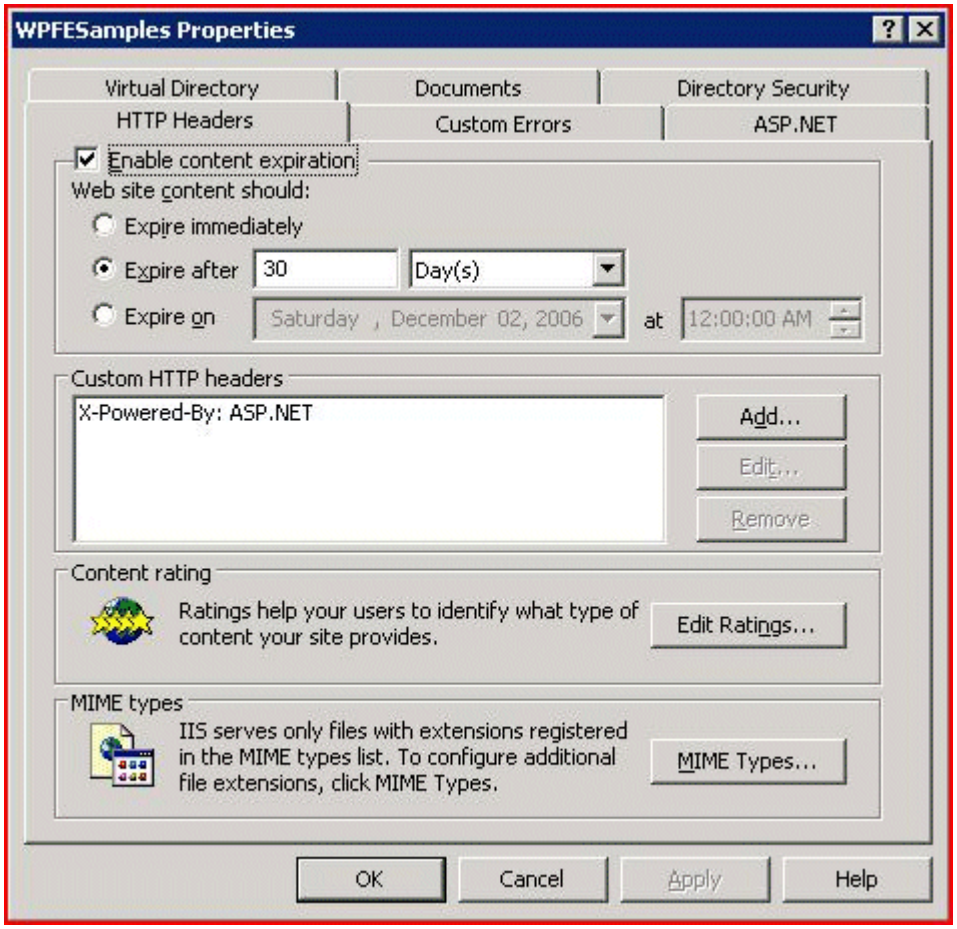

图 22. IIS 6.x 配置

在此对话框的底部有一个按钮，利用它可以指定这些类型。请注意，在此列表上，带有扩展名的仅限于 IIS 的服务器文件已注册，因此，如果 .xaml 未注册，将无法为这些文件提供服务，并且不会使 XAML 融入到“WPF/E”控件中。图 23 显示了应如何对此进行配置。如有必要，请使用此对话框上的“新建”按钮来添加它。

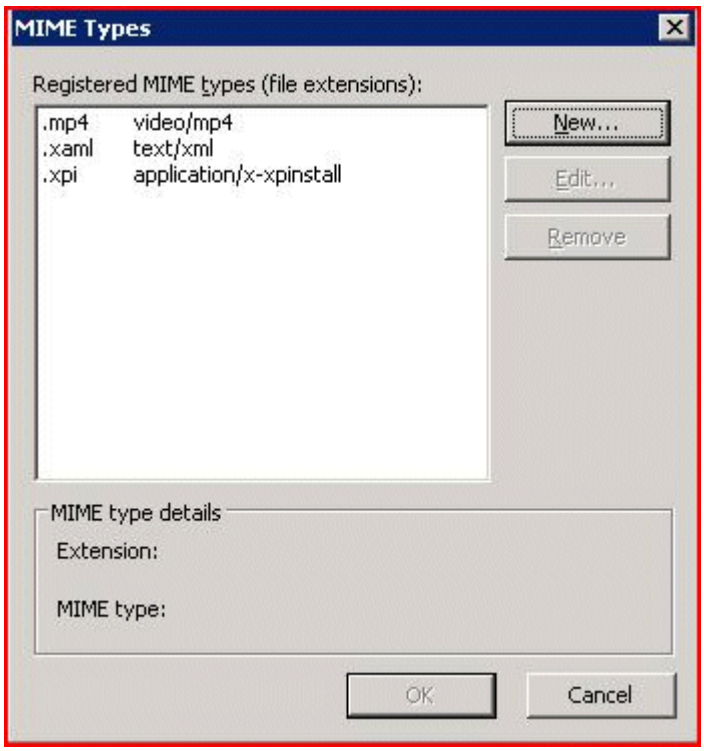

图 23. 在 IIS 中配置 MIME 类型

请注意，虽然图 23 中未显示，但如果您希望您的应用程序可以在 Macintosh 上安装，则应使用 application/octet-stream 作为 .DMG 文件扩展名的 MIME 类型。

配置 IIS 7.x

要在 Windows Vista 中配置 IIS 7.x，应首先在 Windows 的“开始”菜单上右键单击“计算机”，然后选择“管理”。即会启动“计算机管理”控制台，可在其中选择 IIS，如图 24 所示。

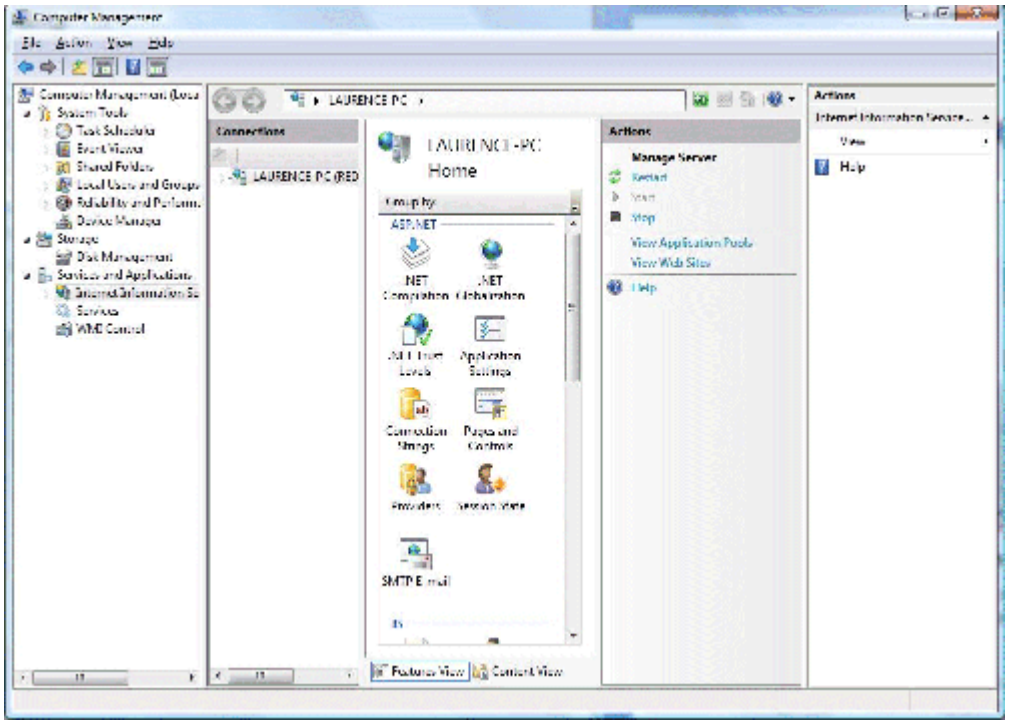

图 24. Windows Vista 中的“计算机管理”控制台

向下滚动项列表，直到您看到 MIME 类型小程序为止。启动此程序以配置 MIME 类型，如图 25 所示。

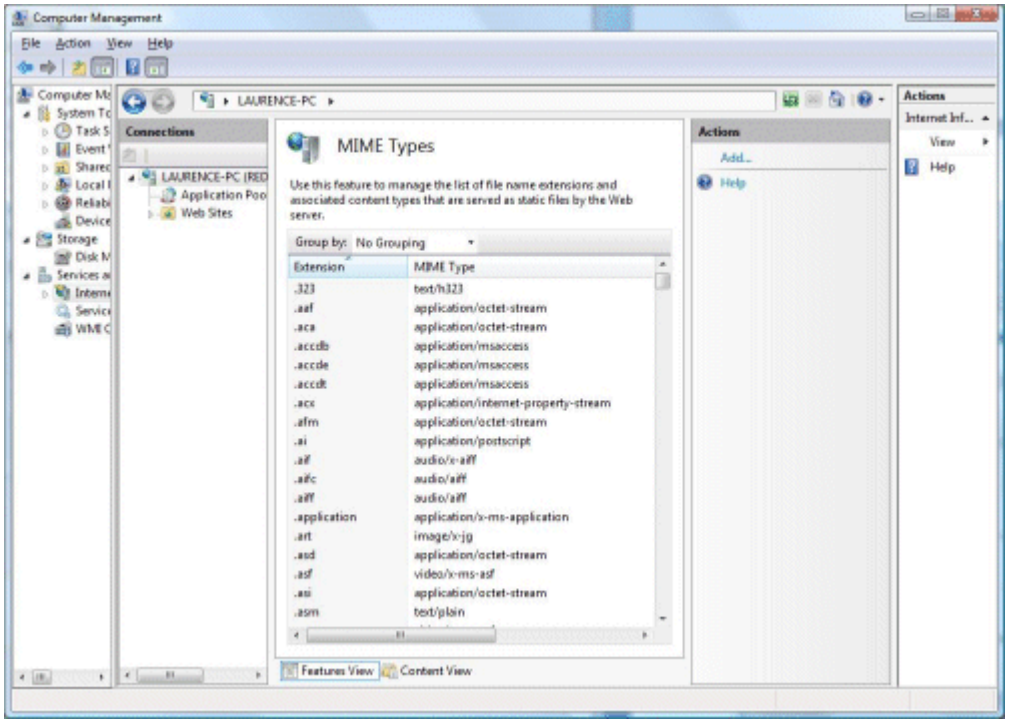

图 25. IIS 7 MIME 类型配置

可以使用“操作”窗格添加所需的 MIME 类型。