ArcGIS Engine 开发实例讲解

此过程说明适合那些使用.NET 建立和部署应用的开发者，它描述了使用 ArcGIS 控件建立和部署应用的方法和步骤。

你可以在下面的目录下找到相应的样例程序：
<安装目录>\DeveloperKit\Samples\Developer_Guide_Scenarios\ ArcGIS_Engine\Building_an_ArcGIS_Control_Application\Map_Viewer

注：*ArcGIS 样例程序不包含在 ArcGIS Engine 开发工具包“典型”安装方式中。如果你没有安装它们，则可以重新运行开发工具包安装向导，选择“定制”或“修改”方式，并选择软件开发包下的样例项进行安装。*

一、项目描述

利用视窗控件建立应用程序的目标是演示并使你熟悉在微软 Visual Studio .NET API 中使用标准 ArcGIS 控件开发和部署 GIS 应用所需的步聚。本节中使用了 Visual Studio .NET 开发环境中的 MapControl、 PageLayoutControl、TOCControl 和 ToolbarControl 等视窗控件。COM、Java 和 C++程序员应该参考如下章节：*利用 ActiveX 建立应用程序*、*利用可视化 JavaBeans 建立应用程序*、*建立命令行方式的 Java 应用*和*建立命令行方式的 C++应用*。

本节演示了创建查看 ArcMap 和 ArcGIS 桌面应用图形文档的 GIS 应用程序的步骤。此节包含了以下技术：

- 在微软 Visual Studio .NET 中加载和嵌入 ArcGIS 控件。
- 向 PageLayoutControl 和 MapControl 中加载图形文档。
- 设置 ToolbarControl 和 TOCControl 的绑定控件。
- 处理窗口缩放。
- 向 ToolbarControl 添加 ArcGIS Engine 命令和工具。
- 创建弹出式菜单
- 在 TOCControl 中管理标签编辑
- 在 MapControl 中绘制图形。
- 为 MapControl、PageLayoutControl 和 ToolbarControl 创建定制工具。
- 用户化 ToolbarControl。
    - 在 Windows 操作系统中部署应用。

二、概述

本方案使用微软 Visual Studio .NET 开发环境加以实现，并使用了 ESRI interop 程序集(Interop Assemblies)，它服务于被放置在.NET 窗体上的、位于.NET 窗体控件(.NET Windows Controls)中的 ArcGIS 控件，这些程序集在托管的.NET 代码和非托管的 COM 代码之间起了桥梁作用。对 COM ArcGIS 控件(COM ArcGIS Controls)成员的引用都要经过 Interop 程序集，然后到达实际的 COM 对象。同样，也从 COM 对象经过 Interop 程序集到达.NET 应用程序。每个 ArcGIS Engine 控件具有方法、属性与事件，它们能够被控件嵌入的容器(如，.NET 窗体)访问。每个控件对象及其功能可以与其他 ESRI ArcObjects 和自定义控件组合使用，创建用户化的客户应用程序。

此方案是使用了 C#和 Visual Basic .NET 两种语言创建，但以下技术实现集中倾向于 C#方案。许多开发者可能会感觉用 Visual Basic .NET 更舒服，那是因为他们已经比较熟悉 Visual Basic 6.0 代码，然而，对于 Java 和 C++程序员来说，他们将会觉得对 C

#程序语言的语法更熟悉。无论你使用哪种开发环境，对于使用 ArcGIS 控件的好坏既依赖于你的编程环境技术，也依赖于你所掌握的 ArcObjects 技术。

在本方案中，使用 ToolbarControl、TOCControl、PageLayoutControl 和 MapControl 来为应用程序提供用户界面。这些 ArcGIS 控件与其他 ArcObjects 和 ArcGIS Engine 命令被开发者一起使用，用来创建一个 GIS 视窗应用。

三、设计

此方案在设计时，首先强调了 ArcGIS 控件如何互相之间进行交互，其次，向开发者解释说明了 ArcGIS 控件对象模型的一部分。

每个.NET ArcGIS Engine 控件包含有一套能够被嵌入其内的窗口即时访问的属性页。这些属性些为控件属性和方法的选择提供了捷径，并且允许开发者不写任何代码即可创建一个应用程序。本方案并没有使用属性页，而是采用写代码的方式建立应用程序。关于属性页的更进一步的信息，请参考 *ArcGIS 开发帮助(ArcGIS Developer Help)*。

四、条件需求

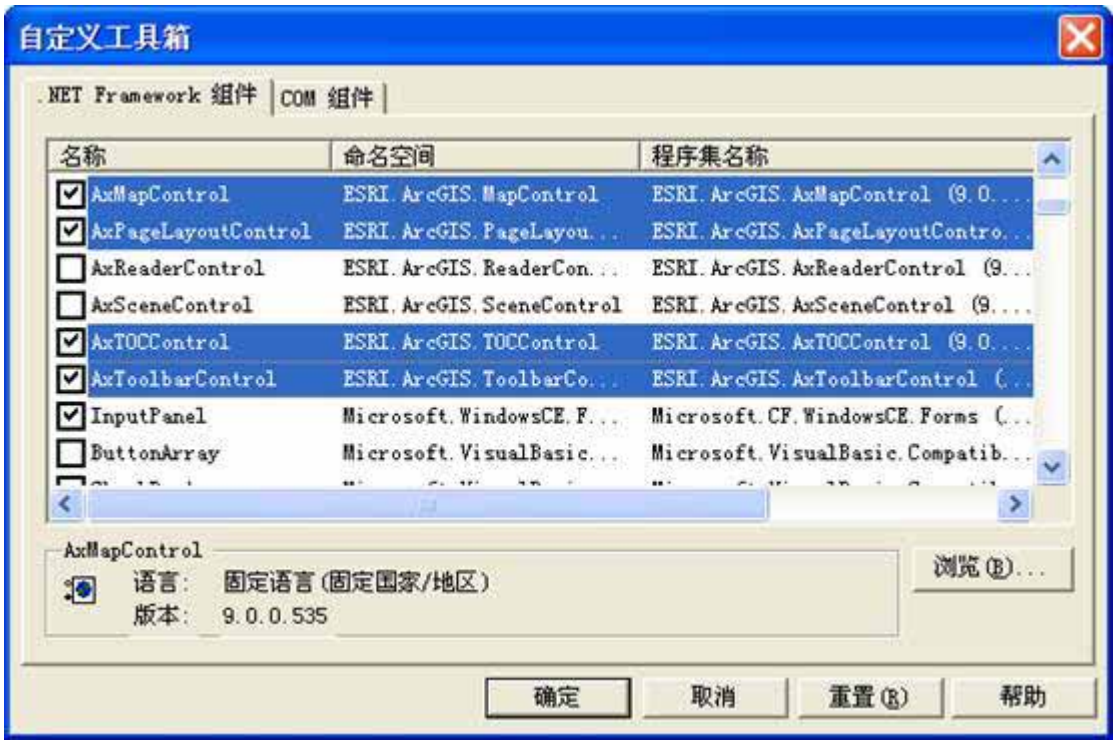
要顺利地完成以下方案，你需要以下条件（对于部署的需求将在后续的部署章节涉及到）：

- 安装具有授权文件的 ArcGIS Engine 开发工具包(Developer Kit)，使之能够用于开发。
- 安装有微软 Visual Studio .NET 2003 开发环境和微软.NET Framework 1.1 及其相应协议。
- 熟悉微软 Windows 操作系统和 Microsoft Visual Studio .NET 的工作知识，会用 C#或 Visual Basic .NET 编程语言。当然，此方案中提供了一些如何在 Microsoft Visual Studio .NET 中使用 ArcGIS 控件的信息，但它不能替代对开发环境的培训。
- 不需要对 ESRI 其它软件有足够的经验，但如果以前对 ArcObjects 有所接触并对 ArcGIS 应用(如，ArcCatalog，ArcMap)有一个基本了解，则对于开发更有利。
- 访问来自本方案的样例数据和代码，它位于：

  <安装目录>\DeveloperKit\Samples\Developer_Guide_Scenarios\ ArcGIS_Engine\Building_an_ArcGIS_Control_Application\Map_Viewer

  本方案中使用到的控件和库如下：

- AxMapControl
- AxPageLayoutControl
- ESRI.ArcGIS.Carto
- ESRI.ArcGIS.Display
- ESRI.ArcGIS.Geometry
- esriMapControl
- esriPageLayoutControl
- AxTOCControl
- AxToolbarControl
- ESRI.ArcGIS.System
- ESRI.ArcGIS.SystemUI
- ESRI.ArcGIS.Utility
- esriTOCControl
- esriToolbarControl



五、实现

下面的实现过程中提供了你成功完成方案所需所有代码。假设你对于开发环境已经有了一定的知识，所以下面没有逐步地详细介绍如何用 Microsoft Visual Studio .NET 开发应用。

(一) 加载 ArcGIS 控件

在你为应用程序编写代码之前，应该先将应用程序将用到的 ArcGIS 控件和其他 ArcGIS Engine 库引用装载到开发环境之中。

1. 启动 Visual Studio .NET，并从新建项目对话框中创建一个新的 Visual C# “ Windows 应用程序 ” 项目。

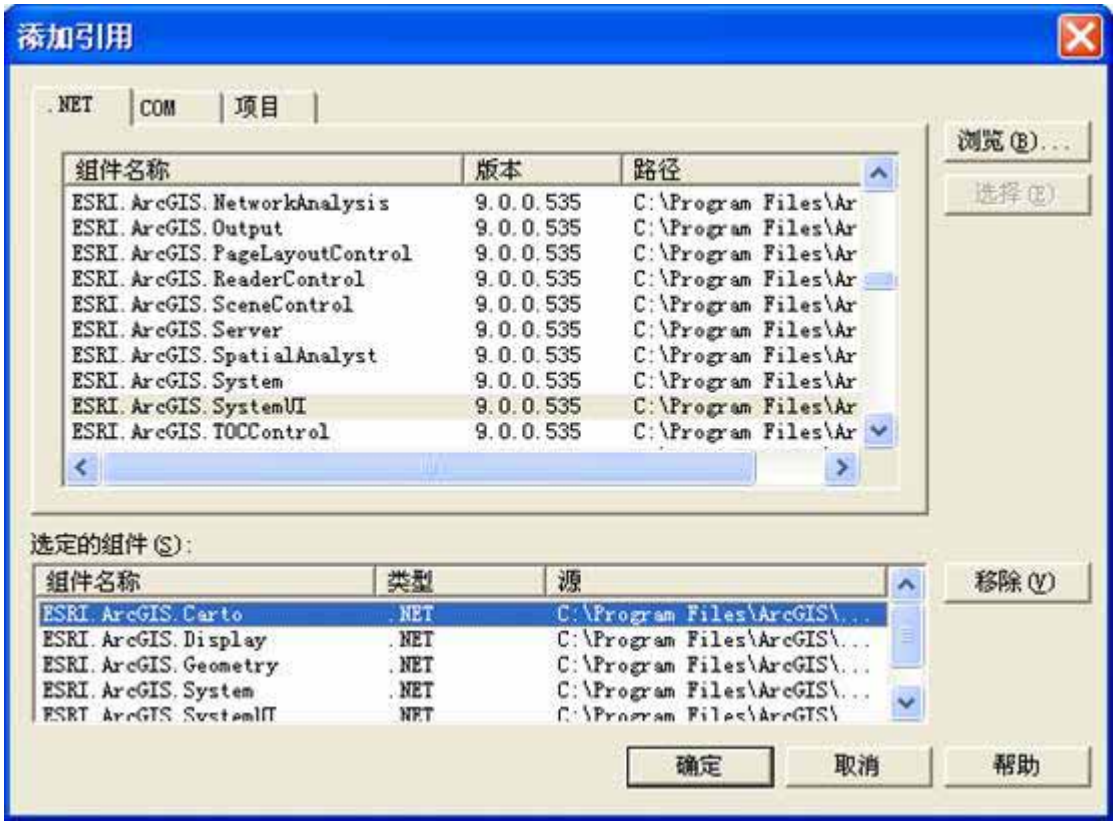
2. 将项目命名为 “ Controls ”，并选择位置存取该项目。

3. 在 “ 工具箱 ” 的 “ Windows 窗体 ” 标签栏中单击右键，然后从上下文菜单中选择 “ 添加/移除项(I)… ”。

4. 在“ 自定义工具箱 ”中选择“ .NET Framework 组件 ”，并复选“ AxMapControl ”，“ AxPageLayoutControl ”，“ AxTOCControl ” 和 “ AxToolbarControl ”，单击**确定**按钮。这样所选择的控件将显示在**工具箱**的 **Windows 窗体**标签栏中。

5. 单击**项目**菜单，并选择 “ 添加引用(R)… ”。

6. 在**添加引用**对话框中，双击 “ ESRI.ArcGIS.Carto ”，“ ESRI.ArcGIS.Display ”，“ ESRI.ArcGIS.Geometry ”，“ ESRI.ArcGIS.System ”，“ ESRI.ArcGIS.SystemUI ”，“ ESRI.ArcGIS.Utility ”。单击**确定**。



注：对于 ESRI .NET 程序集，将通过具体实例来说明，并使用.NET 框架提供的 COM 传送服务从你的 C#项目中调用 ESRI 对象库中的实体对象。

(二) 在容器中嵌入 ArcGIS 控件

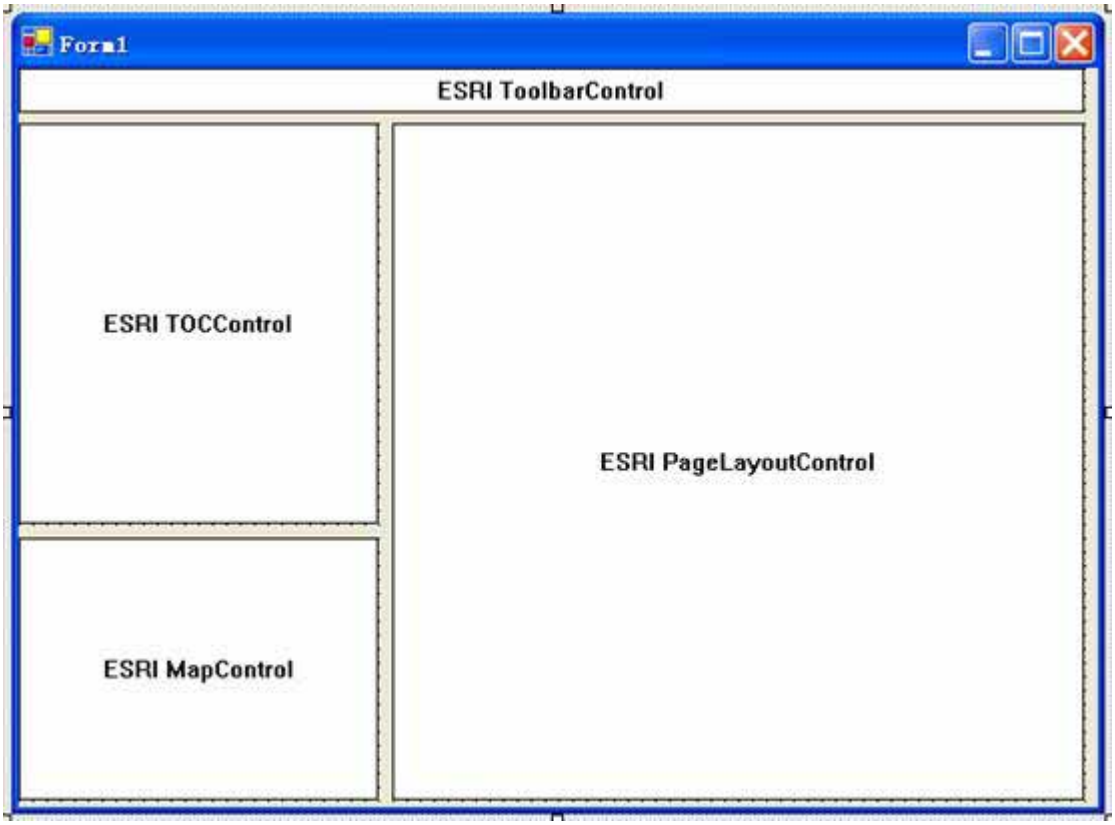
在你能够访问每个控件的事件、属性和方法之前，需要将控件嵌入到.NET 容器中。一旦将控件嵌入窗体内，它们将图形化应用程序的用户界面。

1. 在设计模式下打开.NET 窗体。

2. 双击工具箱 Windows 标签栏中的 AxMapControl 控件，将 MapControl 加入到窗体上。

3. 再将 AxPageLayoutControl、AxTOCControl 和 AxToolbarControl 如上添加到窗体中。

4. 重新调整窗体上各个控件的大小和位置，调整结果如下所示。



5. 在窗体上双击显示窗体代码窗口，在代码窗口的顶部增加“ using ”命令：

```
using System;
using System.Windows.Forms;
// ArcGIS Engine 引用
using ESRI.ArcGIS.SystemUI;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.ToolbarControl;
using ESRI.ArcGIS.TOCControl;
```

注：需注意 C#是区分大小写的。当你键入“ESRI.”时，智能敏感的自动完成功能将允许你通过按 Tab 键完成下一节。

(三) 加载 Map 文档到 MapControl 与 PageLayoutControl

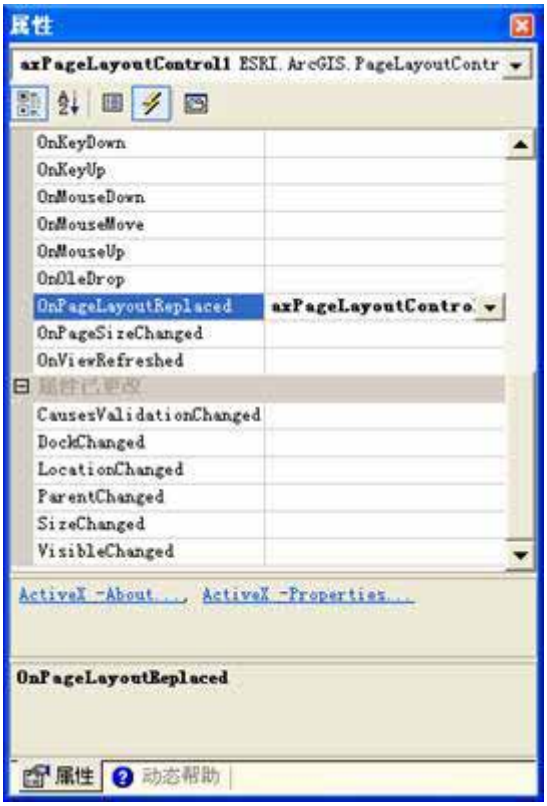
单独的数据层或者使用 ArcMap、ArcGIS 桌面应用程序产生的图形文档，能够被加载到 MapControl 和 PageLayoutControl 中。你可以加载样例图形文档，或者加

载你自己的图形文档。后面你将增加一个浏览图形文档的对话框。

1. 选择 Form_Load 事件，并输入下列代码（如果你使用你自己的图形文档，要替换为正确的文件名）：

```
// 使用相对路径向 PageLayoutControl 加载一个图形文档

string filename = @"..\..\..\..\..\..\..\Data\\ArcGIS_Engine_Developer_Guide\\gulf of st. lawrence.mxd";

if ( axPageLayoutControl1.CheckMxFile(filename) )

{

    axPageLayoutControl1.LoadMxFile(filename, "");

}
```

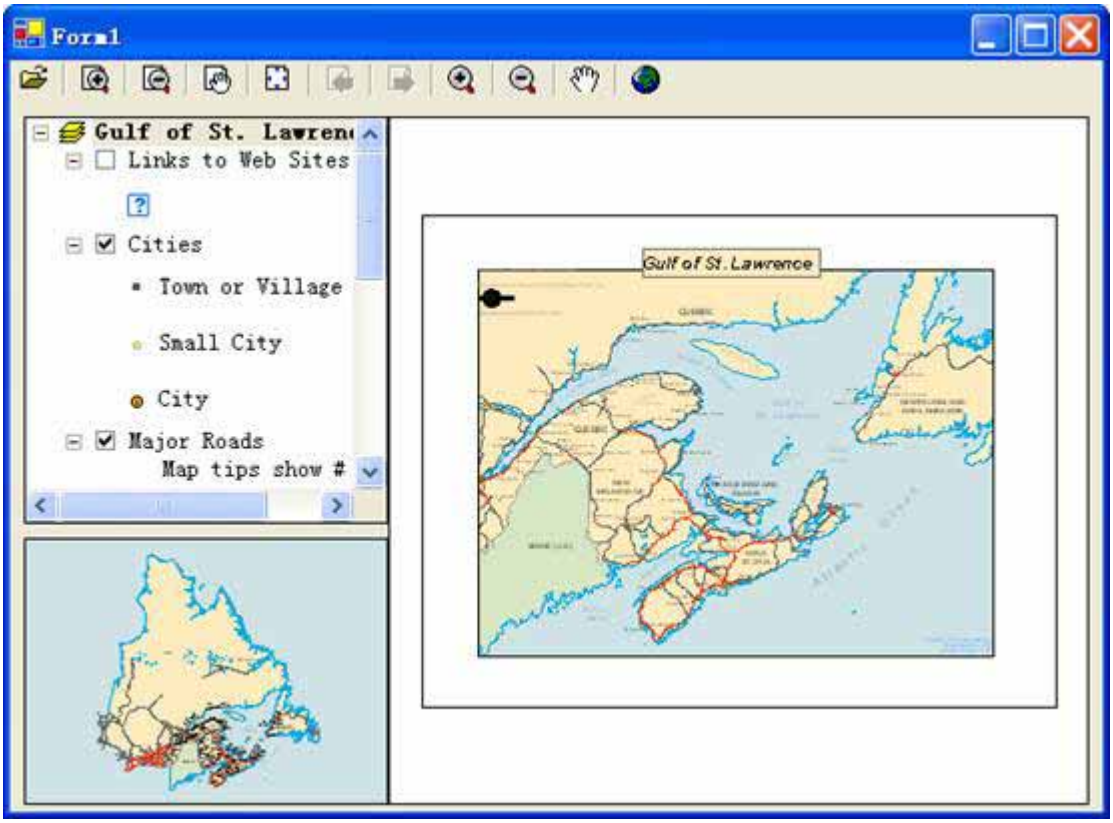
2. 在设计模式显示窗体并从属性窗选择 axPageLayoutControl1 控件，显示 axPageLayoutControl 事件。在 OnPageLayoutReplaced 事件上双击向代码窗口添加该事件的处理函数。



3. 在 axPageLayoutControl1_OnPageLayoutReplaced 事件中键入以下向 MapControl 加载样例图形文档的代码。当文档被装载入 PageLayoutControl 时 OnPageLayoutReplaced 事件将会被触发。

```
private void axPageLayoutControl1_OnPageLayoutReplaced(object sender, ESRI.ArcGIS.PageLayoutControl.IPageLayoutControlEvents_OnPageLayoutReplaced
```

```
Event e)
        {
            // 加载同样的文档到 MapControl
            axMapControl1.LoadMxFile(axPageLayoutControl1.DocumentFilename,
null, null);
            // 设置 MapControl 显示范围至数据的全局范围
            axMapControl1.Extent = axMapControl1.FullExtent;
        }
```



(四) 设置 ToolbarControl 与 TOCControl 控件的绑定控件

对于此应用程序，TOCControl 和 ToolbarControl 控件将与 PageLayoutControl 相互协作，而不是 MapControl。为此 PageLayoutControl 必须设置为绑定控件。TOCControl 使用绑定的 ActiveView 显示图形、图层和符号。而位于 ToolbarControl 上的任何命令、工具或菜单项会受绑定控件的显示影响。

1. 在 Form_Load 事件中的加载文档代码的后面键入以下红色部分内容：

```
private void Form1_Load(object sender, System.EventArgs e)
        {
            // 使用相对路径向 PageLayoutControl 加载一个图形文档
```

```
                string filename =
@"..\..\..\..\..\..\..\Data\\ArcGIS_Engine_Developer_Guide\\gulf of st.
lawrence.mxd";

            if ( axPageLayoutControl1.CheckMxFile(filename) )

            {

                axPageLayoutControl1.LoadMxFile(filename, "");

            }


            // 设置绑定控件

            axTOCControl1.SetBuddyControl(axPageLayoutControl1);

            axToolbarControl1.SetBuddyControl(axPageLayoutControl1);

        }
```

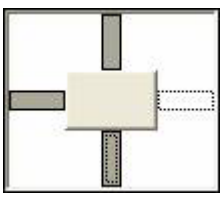
2. 生成并运行应用程序。图形文档被加载到 PageLayoutControl，并且 TOCControl 列出了图形文档中的数据图层。使用 TOCControl 通过复选和取消复选框控制图层的可见性。默认地，图形文档的焦点图(focus map)被装入 MapControl 控件。在这种当前情况下，ToolbarControl 控件显示是空的，因为没有为它添加任何命令。试着缩放窗体，你会注意到控件不会改变尺寸。

(五) 处理窗口缩放

当窗口在运行时进行缩放时，PageLayoutControl 和 MapControl 不会自动改变自身的尺寸。要改变控件的尺寸以便它们总是与匹配窗口的范围，你必须将控件锚定在窗口上。如果 PageLayoutControl 或 MapControl 包含大量的数据，在窗口缩放期间重绘这些数据显得相当重要。为了提高执行效率，你可以禁止数据重绘直到缩放操作完成后再重绘之。在缩放时，可以用一个可伸缩的位图来替代重绘数据。

1. 在设计模式显示窗体并从属性窗口中选择 axPageLayoutControl1。单击 Anchor 属性，将 axPageLayoutControl1 锚定在窗体的顶、左、底和右部。

2. 锚定 axMapControl 控件到窗体的顶、左和底部。



3. 在 Form_Load 事件的开头增加以下代码：

```
        // 当缩放时禁止重绘
```

```
    this.SetStyle(ControlStyles.EnableNotifyMessage, true);
```

4. 向类增加以下常量：

```
public class Form1 : System.Windows.Forms.Form
{
    // ……
    private const int WM_ENTERSIZEMOVE = 0x231;
   private const int WM_EXITSIZEMOVE = 0x232;
    // ……
}
```

5. 向重载的 OnNotifyMessage 方法中增加下列代码：

```
protected override void OnNotifyMessage(Message m)
{
    base.OnNotifyMessage (m);

    // 以下为手工添加的代码
    if ( m.Msg == WM_ENTERSIZEMOVE)
    {
        axMapControl1.SuppressResizeDrawing(true, 0);
        axPageLayoutControl1.SuppressResizeDrawing(true, 0);
    }
    else if ( m.Msg == WM_EXITSIZEMOVE)
    {
        axMapControl1.SuppressResizeDrawing(false, 0);
        axPageLayoutControl1.SuppressResizeDrawing(false, 0);
    }
}
```

6. 生成并运行应用程序，试着缩放窗口。

注：禁止缩放时重画方法是通过检查发送到窗体的 Windows 消息工作的。当窗口开发缩放时，Windows 发送 WM_ENTERSIZEMOVE 窗口消息。此时，我们禁止在 MapControl 和 PageLayoutControl 上绘制图形，而是使用“stretchy bitmap”绘制。当 Windows 发送 WM_EXITSIZEMOVE 消息时，窗体结束缩放，这时我们全部重绘新的范围。

(六) 向 ToolbarControl 增加命令

ArcGIS Engine 提供了 120 多个命令和工具，它们与 MapControl、PageLayoutControl 和 ToolbarControl 直接相互协作。这些命令和工具为你提供了大量的经常使用的地图导航、图形管理、地物选择等方面的 GIS 功能。现在将在你的应用程序中增加这些命令和工具的一部分。

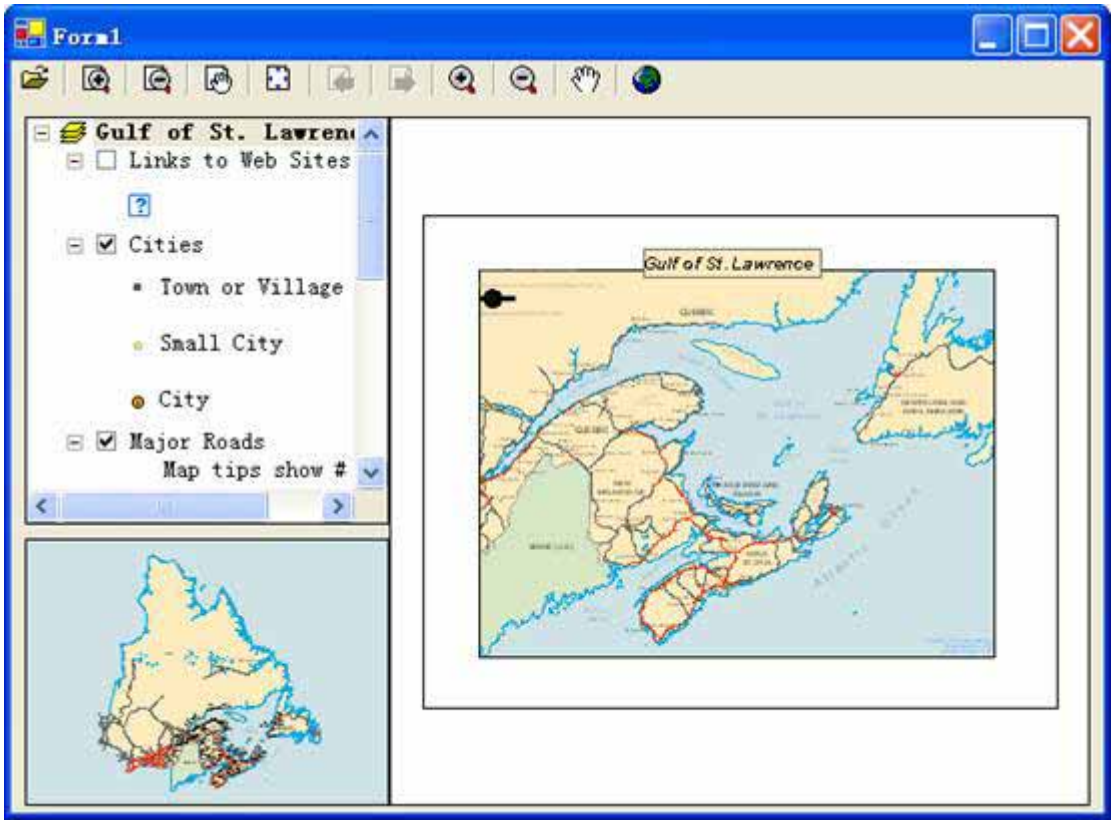
1. 在 Form_Load 事件中的加载文档代码之前添加如下代码。

```
// 增加打开档命令
string progID;
progID = "esriControlToolsGeneric.ControlsOpenDocCommand";
axToolbarControl1.AddItem(progID, -1, -1, false, 0,
    esriCommandStyles.esriCommandStyleIconOnly);
// 增加 PageLayout 导航命令
progID = "esriControlToolsPageLayout.ControlsPageZoomInTool";
axToolbarControl1.AddItem(progID, -1, -1, true, 0,
    esriCommandStyles.esriCommandStyleIconOnly);
progID = "esriControlToolsPageLayout.ControlsPageZoomOutTool";
axToolbarControl1.AddItem(progID, -1, -1, true, 0,
    esriCommandStyles.esriCommandStyleIconOnly);
progID = "esriControlToolsPageLayout.ControlsPagePanTool";
axToolbarControl1.AddItem(progID, -1, -1, true, 0,
    esriCommandStyles.esriCommandStyleIconOnly);
progID = "esriControlToolsPageLayout.ControlsPageZoomWholePageCommand";
axToolbarControl1.AddItem(progID, -1, -1, true, 0,
```

```
                esriCommandStyles.esriCommandStyleIconOnly);

                                        progID                                  =
"esriControlToolsPageLayout.ControlsPageZoomPageToLastExtentBackCommand";

            axToolbarControl1.AddItem(progID, -1, -1, true, 0,

                esriCommandStyles.esriCommandStyleIconOnly);

  progID                                                                        =
"esriControlToolsPageLayout.ControlsPageZoomPageToLastExtentForwardCommand"
;

 axToolbarControl1.AddItem(progID, -1, -1, true, 0,

                esriCommandStyles.esriCommandStyleIconOnly);

            // 增加地图导航命令

            progID = "esriControlToolsMapNavigation.ControlsMapZoomInTool";

            axToolbarControl1.AddItem(progID, -1, -1, true, 0,

                esriCommandStyles.esriCommandStyleIconOnly);

            progID = "esriControlToolsMapNavigation.ControlsMapZoomOutTool";

            axToolbarControl1.AddItem(progID, -1, -1, true, 0,

                esriCommandStyles.esriCommandStyleIconOnly);

            progID = "esriControlToolsMapNavigation.ControlsMapPanTool";

            axToolbarControl1.AddItem(progID, -1, -1, true, 0,

                esriCommandStyles.esriCommandStyleIconOnly);

                                        progID                                  =
"esriControlToolsMapNavigation.ControlsMapFullExtentCommand";

            axToolbarControl1.AddItem(progID, -1, -1, true, 0,

             esriCommandStyles.esriCommandStyleIconOnly);

        // 使用相对路径向 PageLayoutControl 加载一个图形文档

        // ……
```

2. 生成并运行应用程序。现在 ToolbarControl 包含了 ArcGIS Engine 命令和工具，你可以使用它们导航加载到 PageLayoutControl 中的图形文档。使用页面布局命令对当前的页面布局进行导航控制，要对存在于数据框架中的数据进行导航则使用地图命令。

利用找开文档命令可以浏览并加载其他的图形文档。



(七) 给 PageLayoutControl 添加弹出式菜单

与给跟绑定控件协作的 ToolbarControl 增加 ArcGIS Engine 命令一样，按照前面的步骤，你也可以从 ArcGIS Engine 命令创建弹出式菜单。下面将向你的应用程序中增加与 PageLayoutControl 协作的弹出式菜单。当在 PageLayoutControl 可视区域点击鼠标右键的时候，弹出式菜单将显示。

1. 向类中添加如下的成员变量（红色部分）：

```
public class Form1 : System.Windows.Forms.Form

{

    private ESRI.ArcGIS.MapControl.AxMapControl axMapControl1;

    private ESRI.ArcGIS.PageLayoutControl.AxPageLayoutControl
axPageLayoutControl1;

    private ESRI.ArcGIS.TOCControl.AxTOCControl axTOCControl1;

    private ESRI.ArcGIS.ToolbarControl.AxToolbarControl
axToolbarControl1;


    private IToolbarMenu m_ToolbarMenu = new ToolbarMenuClass(); // 弹出
式菜单
```

```
        // ……
```

2. 在 Form_Load 事件中向 ToolbarControl 增加命令代码的后面加载文档代码的前面增加如下代码。

```
private void Form1_Load(object sender, System.EventArgs e)

{

 // 前面是增加地图导航的代码……

// 共享 ToolbarControl 的命令池

m_ToolbarMenu.CommandPool = axToolbarControl1.CommandPool;

 // 向 ToolbarMenu 增加命令

 progID = "esriControlToolsPageLayout.ControlsPageZoomInFixedCommand";

m_ToolbarMenu.AddItem(progID, -1, -1, false,

 esriCommandStyles.esriCommandStyleIconAndText);

progID = "esriControlToolsPageLayout.ControlsPageZoomOutFixedCommand";

m_ToolbarMenu.AddItem(progID, -1, -1, false,

                  esriCommandStyles.esriCommandStyleIconAndText);
progID = "esriControlToolsPageLayout.ControlsPageZoomWholePageCommand";

m_ToolbarMenu.AddItem(progID, -1, -1, false,

                esriCommandStyles.esriCommandStyleIconAndText);

progID="esriControlToolsPageLayout.ControlsPageZoomPageToLastExtentBackComm
and";

m_ToolbarMenu.AddItem(progID, -1, -1, true,

                esriCommandStyles.esriCommandStyleIconAndText);

progID="esriControlToolsPageLayout.ControlsPageZoomPageToLastExtentForwardC
ommand";

m_ToolbarMenu.AddItem(progID, -1, -1, false,

                esriCommandStyles.esriCommandStyleIconAndText);



            // 设置与 PageLayoutControl 挂接
```

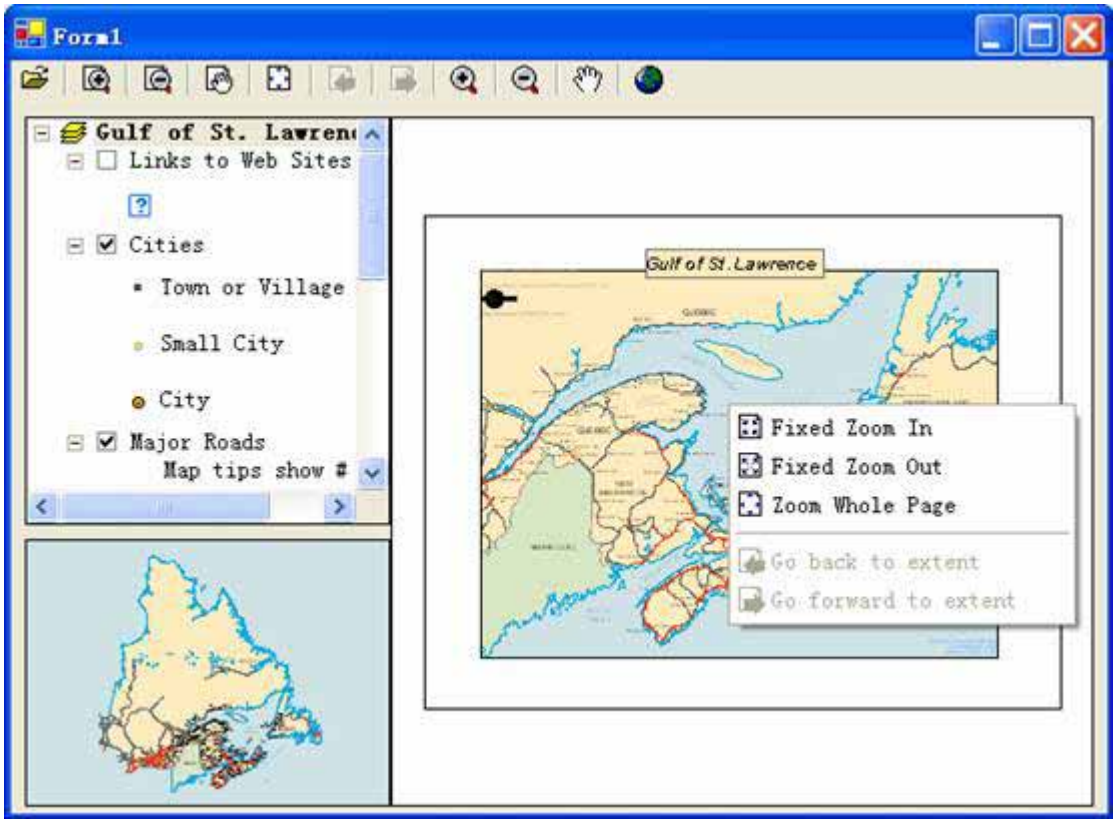
```
m_ToolbarMenu.SetHook(axPageLayoutControl1);


// 后面是加载图形文档的代码……

// ……
```

3. 在设计模式显示窗体并从属性窗口中选择 axPageLayoutControl1，显示 axPageLayoutControl 事件。双击 onMouseDown 事件，向代码窗口中增加事件处理代码。

4. 在 axPageLayoutControl1_onMouseDown 事件中增加如下代码：

```
private void axPageLayoutControl1_onMouseDown(object sender, ESRI.ArcGIS.PageLayoutControl.IPageLayoutControlEvents_onMouseDownEvent e)
{
    // 弹出 ToolbarMenu
    if ( e.button == 2)
    {
        m_ToolbarMenu.PopupMenu(e.x, e.y, axPageLayoutControl1.hWnd);
    }
}
```

5. 生成并运行应用程序。在 PageLayoutControl 的显示区域单击右键以显示弹出菜单，并为页面布局导航。



(八) 在 TOCControl 中控制标签编辑

TOCControl 默认允许用户自动地切换图层的可见性并改变显示在目录表中的名称。你可以增加代码防止用户在编辑名称时输入空的字符串。

1. 在 Form_Load 事件的开始增加下列代码。

```
private void Form1_Load(object sender, System.EventArgs e)
{
    // 当缩放时禁止重绘
    this.SetStyle(ControlStyles.EnableNotifyMessage, true);

    // 设置标签编辑为手动方式
                    axTOCControl1.LabelEdit                    =
esriTOCControlEdit.esriTOCControlManual;

    // 后面是加载文档代码
    // ……
```

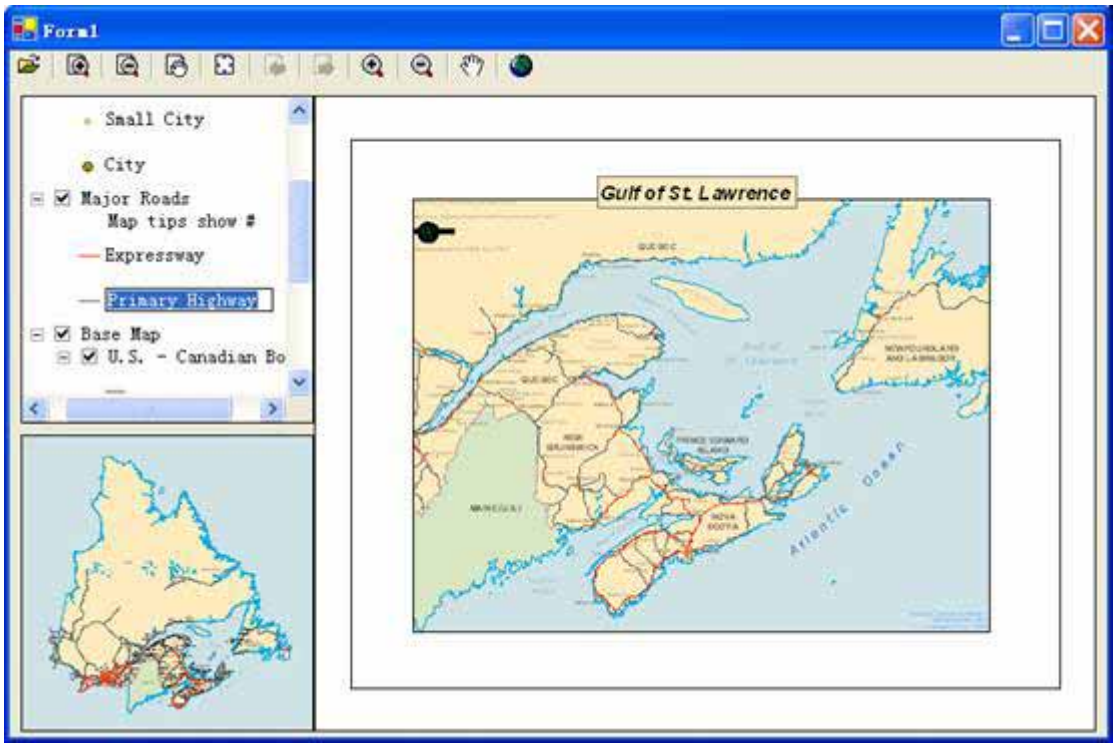
2. 在设计模式显示窗体并从属性窗口选择 AxTOCControl1 控件，显示

AxTOCControl 事件。双击 OnEndLabelEdit 向代码窗口添加事件处理函数。

3. 在 axTOCControl1_OnEndLabelEdit 事件中添加以下代码：

```
private void axTOCControl1_OnEndLabelEdit(object sender,
ESRI.ArcGIS.TOCControl.ITOCControlEvents_OnEndLabelEditEvent e)
{
    // 禁止在编辑标签时键入空字串
    string newLabel = e.newLabel;
    if ( newLabel.Trim() == "" )
    {
        e.canEdit = false;
    }
}
```

4. 生成并生成应用程序。编辑 TOCControl 控件的地图、图层、标题或图例类的标签，在其上点击一次，然后再点一次调用标签编辑。试着用空字串替代标签。在编辑期间，你可以随时使用键盘上的 ESC 键取消编辑。



(九) 在 MapControl 上绘制图形

你可以将 MapControl 作为缩略图窗体使用，并在其上绘制显示 PageLayoutControl 内的焦点地图的当前范围。当你浏览 PageLayoutControl 数据框

架内的数据时，你将看到缩略图窗口也进行了更新。

注：使用地图导航工具导航焦点图（活动图）将改变 PageLayoutControl 中焦点地图的范围并引起 MapControl 更新。使用页面布局工具导航页面布局将改变页面布局的范围（不是 PageLayoutControl 中的焦点图的范围），而 MapControl 将不更新。

1. 向类中增加下列成员变量：

```
public class Form1 : System.Windows.Forms.Form

{

    private ESRI.ArcGIS.MapControl.AxMapControl axMapControl1;

        private      ESRI.ArcGIS.PageLayoutControl.AxPageLayoutControl
axPageLayoutControl1;

    private ESRI.ArcGIS.TOCControl.AxTOCControl axTOCControl1;

            private         ESRI.ArcGIS.ToolbarControl.AxToolbarControl
axToolbarControl1;


    private IToolbarMenu m_ToolbarMenu = new ToolbarMenuClass(); // 弹出
式菜单


    private IEnvelope m_Envelope;  // MapControl 绘制的范围

    private Object m_FillSymbol;   // 在 MapControl 上绘制范围使用的符号

    private ITransformEvents_VisibleBoundsUpdatedEventHandler

            visBoundsUpdatedE;          // PageLayoutControl 的焦点图
事件
```

注：声明的变量 visBoundsUpdatedE 是一个托管。托管是一个类，它能够拥有对指定方法的引用，并使它链接到一个特定的事件。在事件和方法之间的链接过程有时在.NET 中被称作 wiring。

2. 创建一个叫 CreateOverviewSymbol 的新函数。这个函数是创建你将在 MapControl 中使用的符号的地方，此符号是用来描述 PageLayoutControl 焦点地图数据范围的。函数中增加的代码如下：

```
private void CreateOverviewSymbol()

 {
```

```
        // 获取 IRGBColor 接口
        IRgbColor color = new RgbColor();
        // 设置颜色属性
        color.RGB = 255;

        // 获取 ILine 符号接口
        ILineSymbol outline = new SimpleLineSymbol();
        // 设置线符号属性
        outline.Width = 1.5;
        outline.Color = color;

        // 获取 IFillSymbol 接口
        ISimpleFillSymbol simpleFillSymbol = new SimpleFillSymbolClass();
        // 设置填充符号属性
        simpleFillSymbol.Outline = outline;
        simpleFillSymbol.Style = esriSimpleFillStyle.esriSFSHollow;
        m_FillSymbol = simpleFillSymbol;
    }
```

3. 从 Form_Load 事件在 TOCControl 标签编辑代码之前调用 CreateOverviewSymbol 函数。

```
    private void Form1_Load(object sender, System.EventArgs e)
    {
        // 当缩放时禁止重绘
        this.SetStyle(ControlStyles.EnableNotifyMessage, true);

        // 创建 MapControl 使用的符号
        CreateOverviewSymbol();
```

```
            // 下面是标签编辑处理代码

            // ……

        }
```

4. 增加下列 OnVisibleBoundsUpdated 函数。此函数将与地图范围改变时触发的事件相连接，并用来设置新的地图可见边界范围框。通过刷新 MapControl，你强制它重绘其上显示的图形。

```
        private void OnVisibleBoundsUpdated(IDisplayTransformation sender,
bool sizeChanged)

        {

            // 设置新的可见范围

            m_Envelope = sender.VisibleBounds;



            // 改变 MapControl 的前景状态

            axMapControl1.ActiveView.PartialRefresh(

                esriViewDrawPhase,esriViewForeground, null, null);

        }
```

5. PageLayoutControl 默认的事件接口是 IPageLayoutControlEvents。这些事件不告诉我们数据边框内的地图范围。为此你需要使用 PageLayoutControl 的焦点地图的 ItransformEvents 接口。在 PageLayoutControl_OnPageLayoutReplaced 事件处理中的加载文档代码前面增加以下代码。

```
    private void axPageLayoutControl1_OnPageLayoutReplaced(object sender,
ESRI.ArcGIS.PageLayoutControl.IPageLayoutControlEvents_OnPageLayoutReplaced
Event e)

        {

            // 获取 PageLayoutControl 中焦点地图的 IActiveView 对象

            IActiveView activeView = (IActiveView)

                axPageLayoutControl1.ActiveView.FocusMap;
```

```
            // 捕捉 PageLayoutControl 的焦点图的 ITransformEvents 事件
                                    visBoundsUpdatedE                    =
new          ITransformEvents_VisibleBoundsUpdatedEventHandler(OnVisibleB
oundsUpdated);
            ((ITransformEvents_Event)activeView.ScreenDisplay
                        .DisplayTransformation).VisibleBoundsUpdated       +=
visBoundsUpdatedE;


            // 获取焦点图的范围
           m_Envelope = activeView.Extent;


            // 后面是加载地图文档的代码
            // ……
```

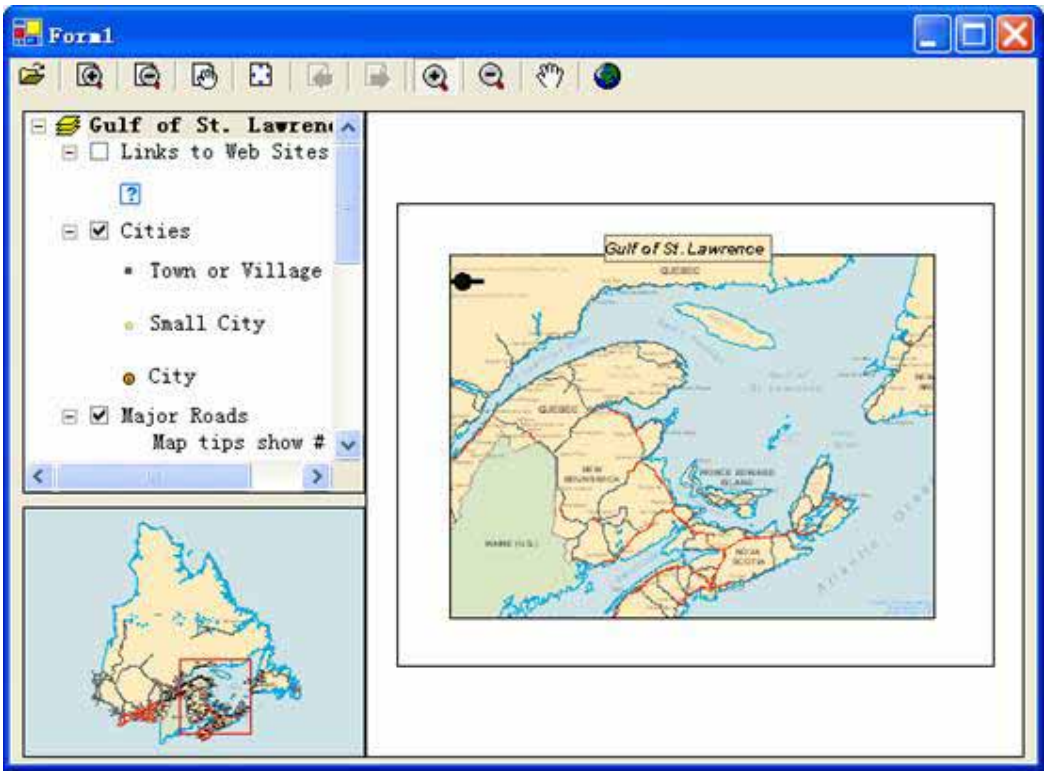
6. 在设计模式下显示窗体并从属性窗中选择 axMapControl1，显示 axMapControl 事件。双击 OnAfterDraw 向代码窗口中增加事件处理。

7. 向 axMapControl1_OnAfterDraw 事件处理中增加以下代码，使用前面创建的符号绘制 MapControl 显示边框。

```
    private      void      axMapControl1_OnAfterDraw(object      sender,
ESRI.ArcGIS.MapControl.IMapControlEvents2_OnAfterDrawEvent e)
     {
            if ( m_Envelope == null)
            {
                return;
            }

            // 如果前景状态被重绘
                        esriViewDrawPhase          viewDrawPhase          =
(esriViewDrawPhase)e.viewDrawPhase;
            if ( viewDrawPhase == esriViewDrawPhase.esriViewForeground )
```

```
        {
            IGeometry geometry = m_Envelope;
            axMapControl1.DrawShape(geometry, ref m_FillSymbol);
        }
    }
```

生成并运行应用程序。使用你先前已经加好的地图导航工具改变 PageLayoutControl 中焦点地图的范围。新的范围被绘制在 MapControl 上。



(十) 创建自定义工具

创建协同 MapControl 和 PageLayoutControl 一起运作的自定义命令和工具，与你或许已经做过的创建 ESRI ArcMap 应用程序命令非常相似。你将创建一个在 PageLayoutControl 鼠标点击位置添加包含当天日期的文本元素的自定义工具。不管用何种方法，创建同 MapControl 和 ToolbarControl 协作的命令与同 PageLayoutControl 一样。

这个自定义工具的相关代码与其他本方案源代码一样很有用。如果你想要直接使用自定义命令，而不自己创建它，请直接到第 24 步。

1. 从**新建项目**对话框创建一个新的 Visual C# “ 类库 ” 项目。
2. 将项目命名为 “ Commands ”，并选择保存位置存贮之。
3. 单击**项目**菜单并选择 “ 添加引用(R)… ”。
4. 在添加引用对话框中，复选“ ESRI.ArcGIS.Carto ”,“ ESRI.ArcGIS.Display ”，

“ESRI.ArcGIS.Geometry”，“ESRI.ArcGIS.System”，“ESRI.ArcGIS.SystemUI”，“ESRI.ArcGIS..Utility”和“ESRI.ArcGIS.ControlCommands”。

5. 在项目中增加一个类，名字叫 AddDateTool。

6. 点击**项目**菜单并选择**添加现有项**，浏览样例源码目录并找到 date.bmp 文件将其加入到你的项目。

7. 在**解决方案资源管理器**中点击 date.bmp 在属性窗口显示其属性。改变**生成操作**属性为**嵌入的资源**。这张位图将被用来作为命令按钮的外观。

8. 改变 AddDateTool 的命名空间的名称为 CSharpDotNETCommands。

```
namespace CSharpDotNETCommands
{
……
```

*注：要在 Visual Basi .NET 中改变命名空间的名称，则在**解决方案资源管理器**的项目上点击右键并选择**属性**，在项目属性页中选择常规并改变**根命名空间**后，按**确定**。*

9. 在 AddDateTool 类代码窗口的顶部增加以下引用。

```
using System;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.SystemUI;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.ControlCommands;
using ESRI.ArcGIS.Utility.BaseClasses;
using System.Runtime.InteropServices;
```

10. 指定 AddDateTool 类继承自 ESRI BaseTool 抽象类，并增加密封（sealed）类修饰。

```
public sealed class AddDateTool : BaseTool
{
```

……

注：抽象类是不能被实例化的类，通常仅包含部分实现代码，或者不包含任何实现代码。它们与接口密切相关；但与接口有明显的区别，也就是说，一个类可能实现任意数量的接口，但它仅能够从一个抽象类中继承。继承了 ESRI BaseTool 抽象类，你便可以比直接实现 esriSystemUI ICommand 和 ITool 接口更快速、简便地创建命令和工具。

密封类修饰说明一个类不能被继承。此类的设计是为了限制其他类从该类继承。

11. 向 AddDateTool 类的构造函数中增加下列代码：

```
public sealed class AddDateTool : BaseTool
{
    public AddDateTool()
    {
        // 获取程序集中的资源数组
        string[] res = GetType().Assembly.GetManifestResourceNames();

        // 设置工具属性
        base.m_bitmap = new System.Drawing.Bitmap(
            GetType().Assembly.GetManifestResourceStream(res[0]));
        base.m_caption = "添加日期";
        base.m_category = "CustomCommands";
        base.m_message = "在页面布局中增加一个日期元素";
        base.m_name = "CustomCommands_Add Date";
        base.m_toolTip = "添加日期";
    }
}
```

注:类构造函数是一个当类创建时被调用的方法。它可以用来初始化类成员变量。构造函数名与类名相同；与其他方法不同的是它没有返回类型。

程序中只个别地替换实现了位图、标题、目录、名称、消息和提示方法，你可以设置从这此方法返回的值，且依赖于 BaseTool 类为这此方法提供的实现。其它

的成员保留 BaseTool 类返回的默认值。

12. 向 AddDateTool 类增加下列成员变量。

```
public sealed class AddDateTool : BaseTool
{
    // HookHelper 对象处理通过 OnCreate 事件的回调
    private IHookHelper m_HookHelper = new HookHelperClass();
……
```

13. 在**类视图**窗口中，定位到 BaseCommand 类的 OnCreate 方法，右键点击之显示上下文菜单。选择**增加**，然后重载并增加该方法至代码窗口。

14. 在重载的 OnCreate 方法中增加以下代码。

```
public override void OnCreate(object hook)
{
    m_HookHelper.Hook = hook;
}
```

注 :要在 Visual Basic .NET 中重载属性和方法,从代码窗口顶部的“Class Name”组合框中选择“Overrides”,从“Method Name”组合框中选择属性或方法。

15. 在类视图中定位到BaseCommand类的Enabled属性并在其上点击右键显示上下文菜单。选择**添加**，然后点**重写**增加该属性至代码窗口。

16. 增加以下代码，重写 BaseTool 类的默认 Enabled 值。

```
public override bool Enabled
{
    get
    {
        // 设置使能属性
        if ( m_HookHelper.ActiveView != null )
        {
            return true;
```

```
            }
            else
            {
                return false;
            }
        }
    }
```

注：ICommand_OnCreate 事件向命令工作的应用程序传送一个句柄或回调。在这种情况下，它可以是 MapControl，PageLayoutControl 或 ToolbarControl。除向 OnCreate 事件增加代码外，你可以使用 HookHelper 判断传向命令的回调类型。命令或工具需要知道如何处理传送的回调，所以必须对 ArcGIS Control 传送的类型作检查。HookHelper 用来控件回调并返回 ActiveView 忽略的回调类型（MapControl、PageLayoutControl 和 ToolbarControl 都是这样）。

17. 在类视图中定位到 BaseTool 基类的 onMouseDown 方法，并在其上点击右键显示上下文菜单。选择**添加**，然后重载并增加该属性至代码窗口。

18. 增加下列代码，重载 BaseTool 类实现的默认 onMouseDown 函数。

```
public override void onMouseDown(int Button, int Shift, int X, int Y)
{
    // TODO:  添加 AddDateTool.onMouseDown 实现
    base.onMouseDown (Button, Shift, X, Y);

    // 获取活动视图
    IActiveView activeView = m_HookHelper.ActiveView;

    // 创建新的文本元素
    ITextElement textElement = new TextElementClass();
    // 创建文本符号
    ITextSymbol textSymbol = new TextSymbolClass();
    textSymbol.Size = 25;
```

```
            // 设置文本元素属性
            textElement.Symbol = textSymbol;
            textElement.Text = DateTime.Now.ToShortDateString();

            // 对 IElementQI
            IElement element = (IElement) textElement;
            // 创建页点
            IPoint point = new PointClass();
                                        point                                   =
activeView.ScreenDisplay.DisplayTransformation.ToMapPoint(X, Y);

            // 设置元素图形
            element.Geometry = point;

            // 增加元素到图形绘制容器
            activeView.GraphicsContainer.AddElement(element, 0);
            // 刷新图形
            activeView.PartialRefresh(esriViewDrawPhase.esriViewGraphics,
                null, null);
        }
```

19. ArcGIS Engine 期望自定义命令是一个 COM 类；因此，你必须指定你所创建的.NET 类也成为一个 COM 类，它是通过创建一个 COM 可调用包装(callable wrapper)实现的。在解决方案资源管理器窗口中，在 Commands 项目上右击鼠标键并从上下文菜单中选择**属性**。

20. 在项目属性页对话框中选择**配置属性**；并点击**生成**。在右面的面板中，改变为“为 Com Interop 注册”为 True，点**确定**。

注：设置“为 Com Interop 注册”属性为 True 会调用程序集注册工具(Regasm.exe)。这将增加客户端期望找到的类信息。

如果“为 Com Interop 注册”属性设为 False，则使项目不要是一个 C#类库类型。

21. 在 AddDateTool 类的代码编写窗口的 AddDateTool 类声明的开始位置增加下列代码，指定 COM 需要的属性。

```
[ClassInterface(ClassInterfaceType.None)]

[Guid("D880184E-AC81-47E5-B363-781F4DC4528F")]
```

*注：新的 GUID 可能通过 Visual Studio .NET 中的 GuidGen.exe 实用工具生成，或者从工具菜单中选择**创建 GUID**。GUID 应该像上面的格式并不包含大括号(curly braces)。*

22. 向 AddDateTool 类成员变量的后面增加下列代码。此代码定义了一些函数，这些函数使用目录实用工具向 ESRI 控件命令（ESRI Control Commands）组件目录注册和取消注册 AddDateTool 类。

```
// 在“ESRI Controls Commands”组件目录注册

#region Component Category Registration

[ComRegisterFunction()]

[ComVisible(false)]

static void RegisterFunction(String sKey)

{

    string fullKey = sKey.Remove(0, 18) + @"\nImplemented Categories";

    Microsoft.Win32.RegistryKey regKey =

          Microsoft.Win32.Registry.ClassesRoot.OpenSubKey(fullKey,
true);

    if (regKey != null)

    {

        regKey.CreateSubKey("{B284D891-22EE-4F12-A0A9-B1DDED9197F4}
");

    }

}



[ComUnregisterFunction()]
```

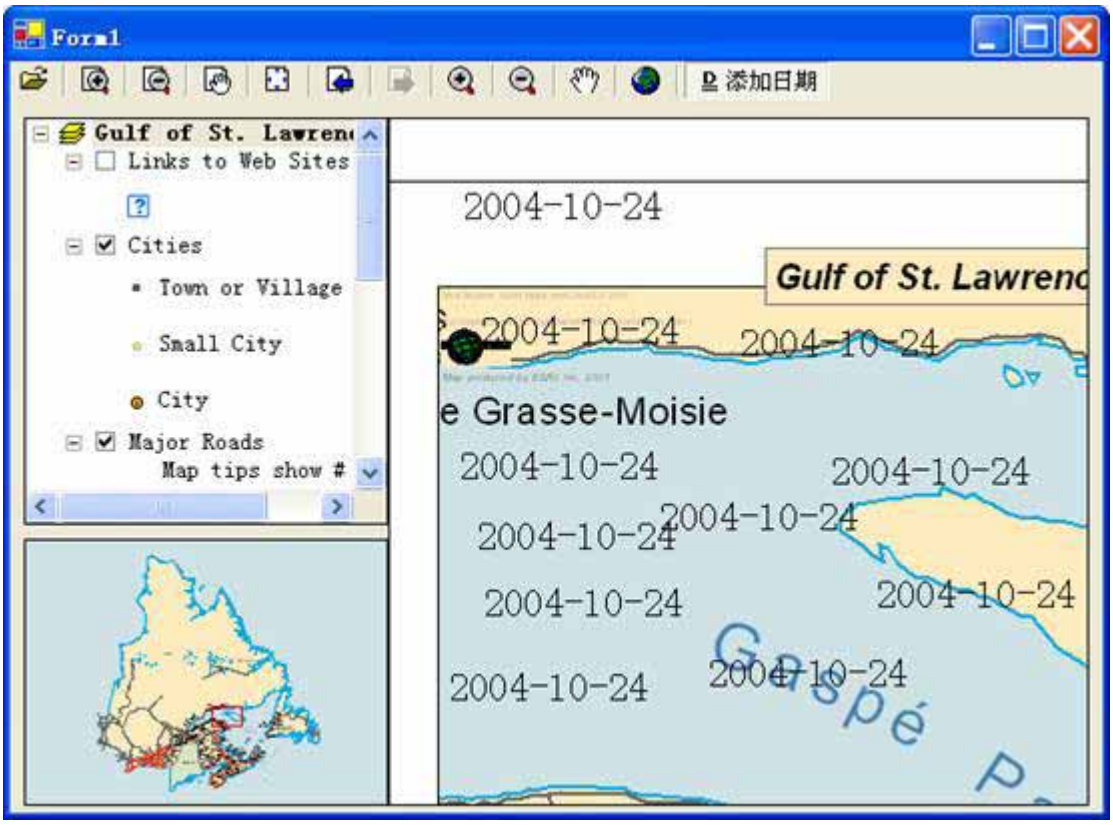
```
        [ComVisible(false)]
        static void UnregisterFunction(String sKey)
        {
            string fullKey = sKey.Remove(0, 18) + @"\Implemented Categories";
            Microsoft.Win32.RegistryKey regKey =
                  Microsoft.Win32.Registry.ClassesRoot.OpenSubKey(fullKey,
true);
            if (regKey != null)
            {
                regKey.DeleteSubKey("{B284D891-22EE-4F12-A0A9-B1DDED9197F4}
");
            }
        }
        #endregion
```

23. 生成工程。

24. 在方案开始创建的 Visual Studio .NET “Windows 应用程序”项目中，增加地图导航命令代码的后面增加以下代码。

```
        private void Form1_Load(object sender, System.EventArgs e)
        {
            // 前面是命令导航代码……
            // 添加自定义日期工具
            progID = "CSharpDotNETCommands.AddDateTool";
            axToolbarControl1.AddItem(progID, -1, -1, true, 0,
                esriCommandStyles.esriCommandStyleIconAndText);
          // 后面是 ToolbarMenu 相关代码……
       }
```

25. 生成并运行应用程序，使用添加日期工具向 PageLayoutControl 上增加一个包含当天日期的文本元素。



(十一) 自定义 ToolbarControl

同在 Form_Load 事件中向 ToolbarControl 控件增加 ArcGIS Engine 命令和工具一样，你也可以使用定制对话框和自定义 ToolbarControl 的方式添加命令和工具。要实现它，就要将 ToolbarControl 置为定制模式并显示定制对话框。

1. 向类中增加下列成员变量：

……

```
private ITransformEvents_VisibleBoundsUpdatedEventHandler
    visBoundsUpdatedE;          // PageLayoutControl 的焦点图事件


private ICustomizeDialog m_CustomizeDialog = new
    CustomizeDialogClass(); // CurtomizeDialog 被 ToolbarControl 使用
private ICustomizeDialogEvents_OnStartDialogEventHandler
    startDialogE; // CustomizeDialog 启动事件
private ICustomizeDialogEvents_OnCloseDialogEventHandler
    closeDialogE; // CustomizeDialog 关闭事件
```

……

注：Visual Studio .NET 提供了当程序集对 COM interop 开放时执行的函数在系统中被注册和取消注册的功能。这就允许你在定制对话框可能找到的组件目录中注册你自己的类。

2. 创建一个叫 CreateCustomizeDialog 的新函数，这个函数是你通过增加如下代码创建自定义对话框的地方。

```
private void CreateCustomizeDialog()

{

// 设置自定义对话框事件

startDialogE = new

ICustomizeDialogEvents_OnStartDialogEventHandler(OnStartDialog);

((ICustomizeDialogEvents_Event)m_CustomizeDialog).OnStartDialog+=
startDialogE;

closeDialogE = new

ICustomizeDialogEvents_OnCloseDialogEventHandler(OnCloseDialog);

 ((ICustomizeDialogEvents_Event)m_CustomizeDialog).OnCloseDialog +=

                closeDialogE;

// 设置标题

m_CustomizeDialog.DialogTitle = "自定义 ToolbarControl 项目";

// 显示“从文件添加”按钮

m_CustomizeDialog.ShowAddFromFile = true;

// 设置将增加新项目的 ToolbarControl

m_CustomizeDialog.SetDoubleClickDestination(axToolbarControl1);

}
```

注：设置 ComVisible 属性为 false 确保此方法不能被 COM 客户端直接调用。当程序集通过 COM 注册时，它不影响被调用的方法。

3. 在 Form_Load 事件中调用 CreateOverviewSymbol 子过程以前调用 CreateCustomizeDialog 函数。

```
            private void Form1_Load(object sender, System.EventArgs e)

            {
```

```
            // 当缩放时禁止重绘
            this.SetStyle(ControlStyles.EnableNotifyMessage, true);

            // 为 ToolbarControl 创建自定义对话框
        CreateCustomizeDialog();
    ......
    }
```

4. 在窗体上增加一个名叫“ chkCustomize ”的复选框，并将标题命名为“ 定制 ”。

5. 在设计模式显示窗体并从属性窗口选择 chkCustomize 控件，显示 chkCustomize 事件。在 CheckedChanged 事件上双击向代码窗口增加相应的事件处理。

6. 向 chkCustomize_CheckedChanged 事件中增加下列代码。

```
        private void chkCustomize_CheckedChanged(object sender, System.EventArgs e)
        {
            // 显示或隐藏自定义对话框
            if (chkCustomize.Checked == false )
            {
                m_CustomizeDialog.CloseDialog();
                axToolbarControl1.Customize = false;
            }
            else
            {
                m_CustomizeDialog.StartDialog(axToolbarControl1.hWnd);
                axToolbarControl1.Customize = true;
            }
    }
```

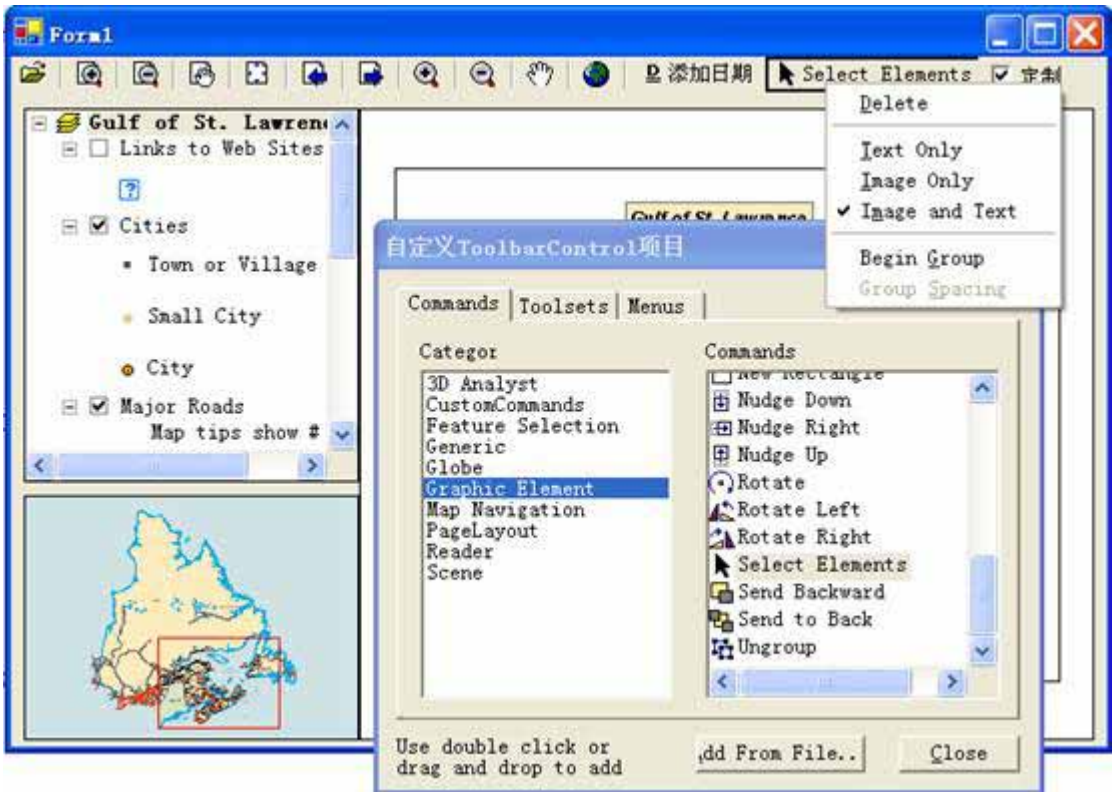
7. 增加下以下 OnStartDialog 和 OnCloseDialog 事件处理函数。这些函数将与自

定义对话框打开或关闭时触发的事件紧密连接。

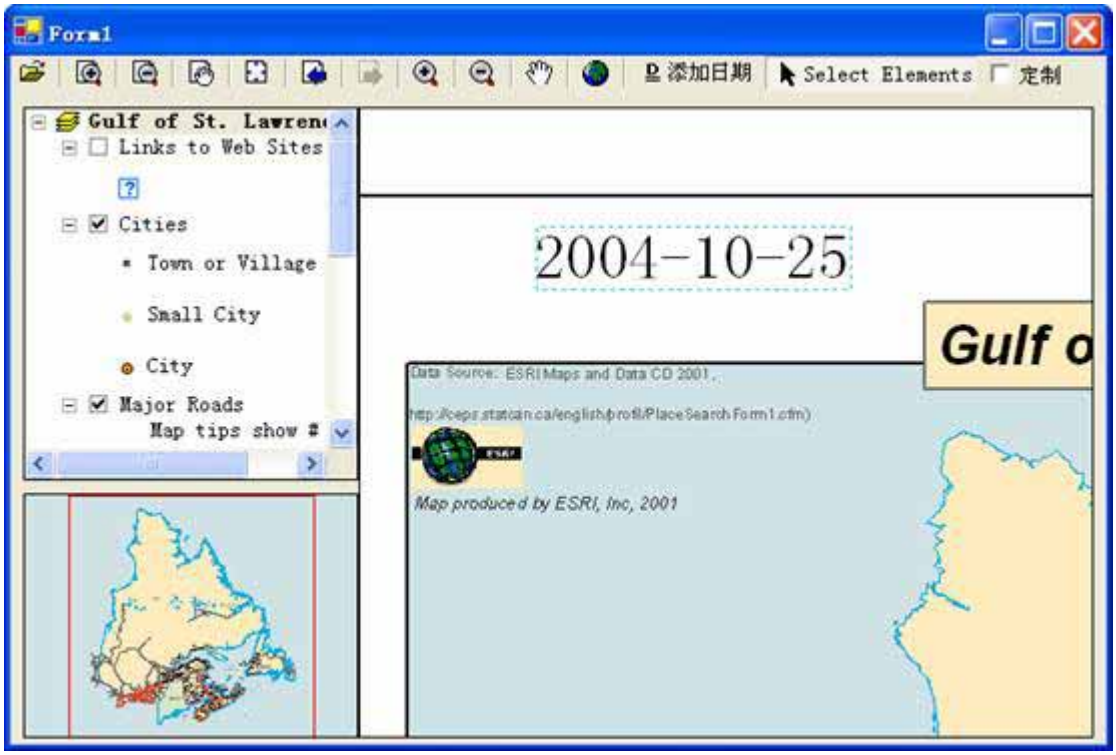
```
        private void OnStartDialog()
        {
            axToolbarControl1.Customize = true;
        }

        private void OnCloseDialog()
        {
            axToolbarControl1.Customize = false;
            chkCustomize.Checked = false;
    }
```

8. 生成并运行应用程序，选择定制复选框使 ToolbarControl 进入自定义模式，并打开自定义对话框。

9. 在自定义 ToolbarControl 项目对话框中的左边目录（Categories）列表中选择“ Graphic Element ”项，然后在右边的命令（Commands）列表中“ Select Elements ”项上双击将其加入到 ToolbarControl 工具栏中。右键点击 ToolbarControl 上的任何一个项目，你可以调整它的显示样式和组合特性。



10. 结束定制应用。使用选择工具移动包含日期的文本元素。



六、部署

要将应用程序成功地部署到另一台机器上，必须为应用程序配置协议。首先，它必须检查产品协议是否有效，其次，它必须初始化协议。如果协议配置不正确有，应用程序将不能运行。

注：当采用 ESRI ArcObjects 开发独立运行的程序时，应用程序负责检查并配置协议选项。它通过实现 CoClass AoInitialize 和 IAoInitialize 接口来支持协议配置。应用程序运行时，在任何 ESRI ArcObject 功能被访问之前协议初始化必须先被执行。如果初始化失败将导致应用程序错误。

1. 向类中增加下列成员变量。

```
public class Form1 : System.Windows.Forms.Form

    {

        private ESRI.ArcGIS.MapControl.AxMapControl axMapControl1;

            private     ESRI.ArcGIS.PageLayoutControl.AxPageLayoutControl
axPageLayoutControl1;

        private ESRI.ArcGIS.TOCControl.AxTOCControl axTOCControl1;

                private        ESRI.ArcGIS.ToolbarControl.AxToolbarControl
axToolbarControl1;


        // 应用初始化对象
```

```
        private IAoInitialize m_AoInitialize = new AoInitializeClass();


// 后面是弹出菜单变量声明代码
.....
```

2. 在 Form_Load 事件的最开始位置增加下列代码。

```
        private void Form1_Load(object sender, System.EventArgs e)
        {
            // 创建新的 AoInitialize 对象
            if ( m_AoInitialize == null)
            {
                System.Windows.Forms.MessageBox.Show(
                   "初始化失败，程序不能运行！");
                this.Close();
            }

            // 判断产品是否有效
            esriLicenseStatus licenseStatus = (esriLicenseStatus)
                m_AoInitialize.IsProductCodeAvailable(
                esriLicenseProductCode.esriLicenseProductCodeEngine);
            if (licenseStatus == esriLicenseStatus.esriLicenseAvailable )
            {
                licenseStatus = (esriLicenseStatus)
                    m_AoInitialize.Initialize(esriLicenseProductCode.esr
iLicenseProductCodeEngine);
                            if (licenseStatus != 
esriLicenseStatus.esriLicenseCheckedOut )
                {
```

```
                System.Windows.Forms.MessageBox.Show(
                    "初始化失败，应用程序不能运行！");
                this.Close();
            }
        }
        else
        {
            System.Windows.Forms.MessageBox.Show(
                "ArcGIS Engine 产品无效，此程序不能运行！");
            this.Close();
        }

        // 当缩放时禁止重绘
    this.SetStyle(ControlStyles.EnableNotifyMessage, true);
// 后面是创建自定义对话框的代码……
……
}
```

3. 在设计模式显示窗体并在属性窗口选择 Form1，显示窗体事件。在 Closing 事件上双击向代码窗口增加事件处理代码。

4. 在 Form_Closing 事件中增加以下代码：

```
        private void Form1_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            // 释放 COM 对象并关闭 AoInitialize 对象
            ESRI.ArcGIS.Utility.COMSupport.AOUninitialize.Shutdown();
            m_AoInitialize.Shutdown();
}
```

5. 在 Release 模式下生成项目和解决方案。

要将应用程序成功地部署到用户机器上：

- 要将应用程序的可执行文件和包含自定义命令的动态链接库 DLL 发布到用户机器上。程序集注册工具（RegAsm.exe）必须被用来向注册表增加关于自定义类的信息。

- 用户机器上需要安装有 ArcGIS Engine 运行时库和标准 ArcGIS Engine 协议。

- 客户机上需要安装 Microsoft .NET Framework 1.1。

七、附加资源

下列资源可以帮助你理解和应用在本方案中在在的概念和技术。

- 在 ArcGIS Engine 开发工具包中包含了其他可用的文档：ArcGIS 开发帮助，组件帮助，对象模型图表和适合于初学者的样例程序。

- ArcGIS 开发在线——一个 Web 站点，提供了最新的 ArcGIS 开发信息，包括程序样例和技术文档。请访问 http://arcgisdeveloperonline.esri.com

- ESRI 在线讨论组——Web 站点，从其他 ArcGIS 开发者提供无偿援助。请访问 http://support.esri.com 并点击用户论坛页签。

- 微软 Visual Studio .NET 开发环境中的文档。