MapGIS 10.2 二维插件式二次开发手册

武汉中地数码科技有限公司

中国·武汉

2017 年 3 月

目　录

第1章 基础应用

1.1 欢迎屏

1.1.1 示例功能概述

实现功能：启动数据中心应用程序之前将会先显示欢迎屏；

效果预览如下图所示：

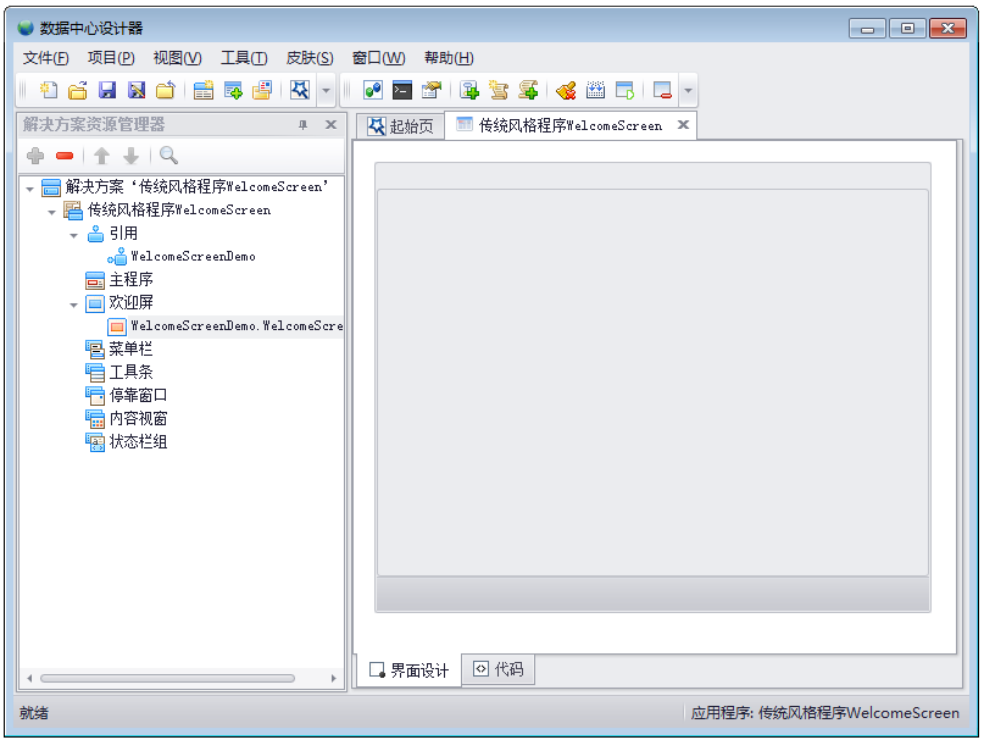

图 1.1 在数据中心设计器中加载欢迎屏

1.1.2 实现思路及关键代码

步骤一：打开 VS 2010 新建一个 MapGIS 项目，命名为“WelcomeScreenDemo”；在该 MapGIS 项目中添加一个 WelcomeScreen 插件项，命名为 WelcomScreen1，在属性窗口中设置属性 ScreenImage，指定需要在欢迎屏幕上显示的图片，图片类型为“.ICO|.BMP|.PNG”，并设置 Result 属性值为“Yes”或者“No”，设置完成后点击确定即可；

步骤二：此时的欢迎屏是一张简单的界面图片，可以定义一个对话框并在 Show()方法添加代码实现对话框显示图片的功能。在这里我们通过一个定义好的对话框显示图片为例说明该功能的用法。

在 MapGIS 项目中添加引用，引用文件为“MapGIS.PlugUtility.dll”，打开 WelcomeScreen1.cs 文件，添加 MapGIS.PlugUtility.dll 引用代码：

程序代码 1-1 添加引用关键代码

```
using MapGIS.PlugUtility;
```

然后找到 Show 方法添加如下代码：

程序代码 1-2 显示欢迎屏关键代码

```
public void Show()
{
    //定义一个欢迎屏显示窗体
    ScreenForm sf = new ScreenForm(this.ScreenImage);
    sf.Show();
}
```

1.2 地图及基本操作

1.2.1 示例功能概述

实现功能：用户可以点击菜单栏下的打开地图，可以实现弹框选择地图文档，并打开该地图文档，自动显示该文档第一个地图，同时还提供了地图的基本操作，如放大，缩小等。

效果预览如下图所示：

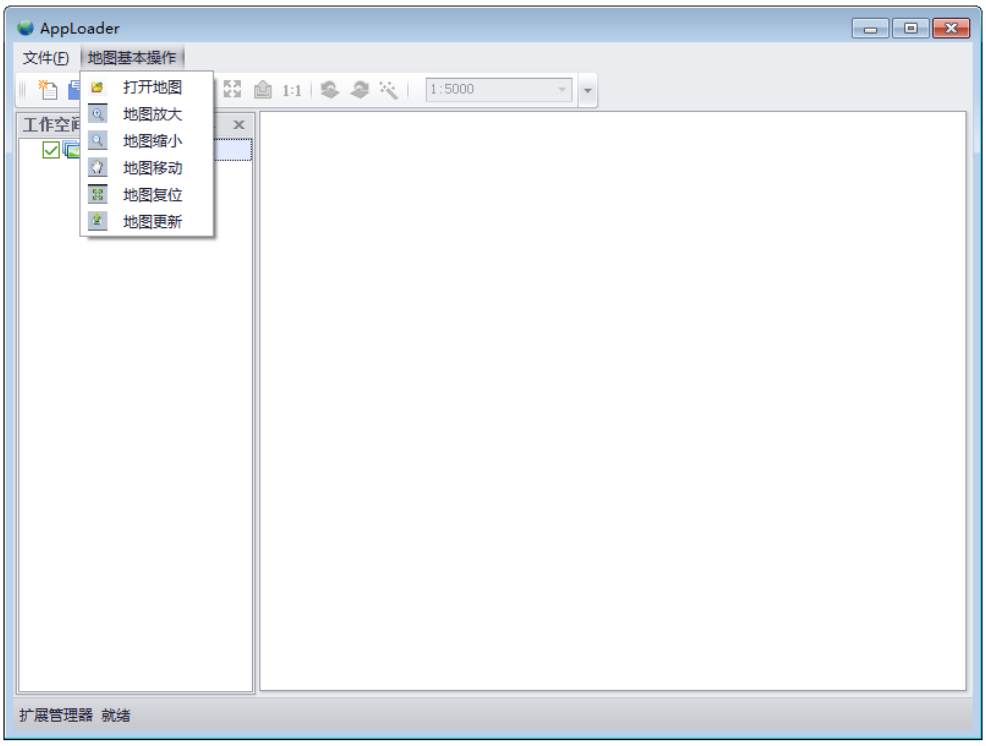

图 1.2 初始化界面示意图

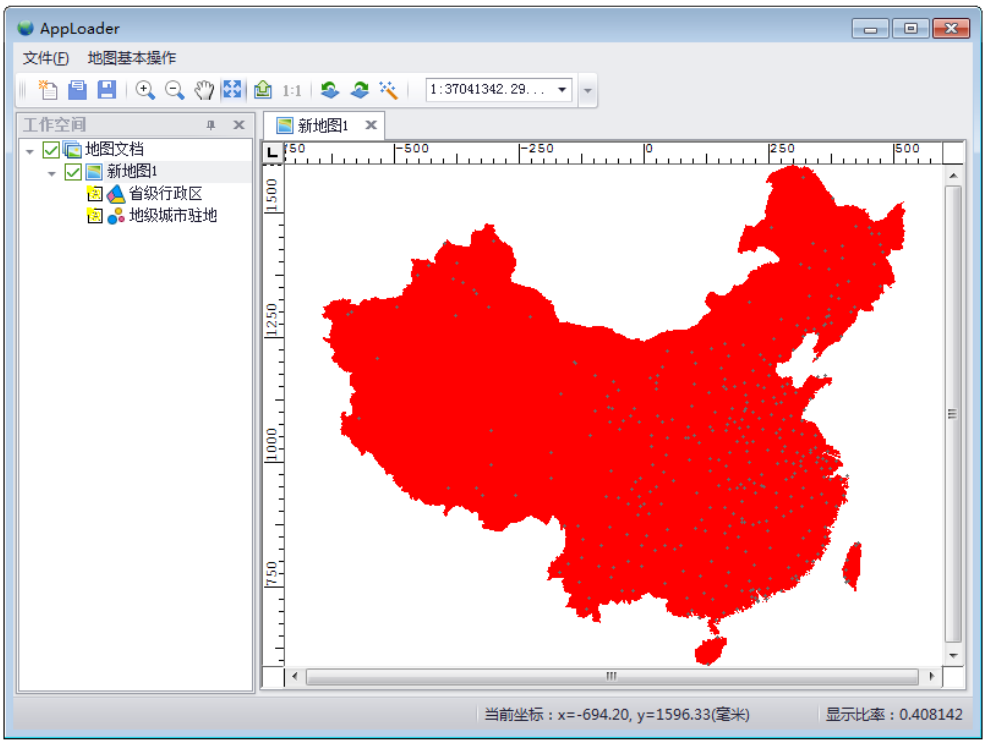

图 1.3 点击打开地图后示意图

1.2.2 实现思路及关键代码

插件项：MenuBar（菜单栏）、Command（命令按钮）。

步骤一：打开 VS 2010 新建一个 MapGIS 项目，命名为“MapBaseOperate”；在该 MapGIS 插件项目中创建一个 MenuBar 菜单栏，将 Name 属性值设为“MapViewMBar”，Caption 属性值设为“地图基本操作”；再添加 6 个 Command 插件项（命名为 OpenMapCmd、MapZoominCmd、MapZoomoutCmd、MapMoveCmd、MapRestoreCmd、MapRefreshCmd）实现的功能分别为打开地图、地图放大、地图缩小、地图移动、地图复位、地图更新；

步骤二：将 Command 插件项绑定到菜单栏中：在 MapViewMBar.cs 初始化函数中添加如下代码：

程序代码 1-3 MapViewMBar()关键代码

```
public MapViewMBar()
  {
      items = new IItem[6];
      items[0] = new Item();
      items[0].Key = "MapBaseOperate.OpenMapCmd"; //添加打开地图
      items[0].ShowLargeImage = false; //是否显示大图标
      items[0].Group = false; //是否分组
```

```
            items[1] = new Item();
            items[1].Key = "MapBaseOperate.MapZoominCmd";//添加地图放大
            items[1].ShowLargeImage = false;
            items[1].Group = false;

            items[2] = new Item();
            items[2].Key = "MapBaseOperate.MapZoomoutCmd"; //添加地图缩小
            items[2].ShowLargeImage = false;
            items[2].Group = false;

            items[3] = new Item();
            items[3].Key = "MapBaseOperate.MapMoveCmd"; //添加地图移动
            items[3].ShowLargeImage = false;
            items[3].Group = false;

            items[4] = new Item();
            items[4].Key = "MapBaseOperate.MapRestoreCmd"; //添加地图复位
            items[4].ShowLargeImage = false;
            items[4].Group = false;

            items[5] = new Item();
            items[5].Key = "MapBaseOperate.MapRefreshCmd"; //添加地图更新
            items[5].ShowLargeImage = false;
            items[5].Group = false;
        }
```

步骤二：打开地图文档：在 OpenMapCmd.cs 文件中定义一个应用框架对象全局变量（IApplication）hk，在 OnCreate 方法中添加如下代码：

程序代码 1-4 OnCreate 关键代码

```
        public void OnCreate(IApplication hook)
        {
            hk = hook;
        }
```

打开地图文档并显示地图，则可利用已有的地图视图控件显示，调用框架对象 hk 的工作空间引擎对象的 FireMenuItemClickEvent 方法；在 OnClick 方法体内实现打开地图文档的功能，关键代码如下：

程序代码 1-5 打开地图文档关键代码

```
        public void OnClick()
        {
```

```
                //变量定义
                Map map = null;
                string docuname = null;
                Document doc = null;

                //选择打开的地图文档
                OpenFileDialog ofd = new OpenFileDialog();
                ofd.Multiselect = true;
                ofd.Filter = ".mapx(地图 XML 文档)|*.mapx|.map(地图 XML 文档)|*.map";
                if (ofd.ShowDialog() == DialogResult.OK)
                {
                    docuname = ofd.FileName;
                }

                //打开指定的地图文档
                doc = this.hk.Document;
                doc.Open(docuname);
                Maps maps = doc.GetMaps();
                if (maps.Count > 0)
                {
                    map = maps.GetMap(0);
                    map.get_Layer(0).State = LayerState.Active;
                }

                //打开自带的地图视图
hk.WorkSpaceEngine.FireMenuItemClickEvent("MapGIS.WorkSpace.Style.PreviewMap", map);
            }
```

步骤三：地图基本操作功能，以地图放大为例：在 MapZoominCmd.cs 文件中定义一个应用框架对象全局变量（IApplication）hk，在 OnCreate 方法中添加代码将框架对象赋值给 hk，代码参考步骤二；在 OnClick 方法实现地图放大功能，关键代码如下：

程序代码 1-6 地图放大功能关键代码

```
        public void OnClick()
        {
            //判断是否有地图视图存在
            if ((hk.ActiveContentsView as IMapContentsView) == null)
                return;
            //获取当前处于激活状态的地图视图控件
```

```
                    MapControl    mapctr = (hk.ActiveContentsView as
IMapContentsView).MapControl;
                    //放大地图
                    mapctr.ZoomIn();
                }
```

注意：地图缩小、地图移动、地图复位和地图更新功能和地图放大功能类似，唯一不同的就是各个功能实现的方法不同，分别调用地图视图 MapControl 对象的 ZoomOut()、Move()、 Restore()和 Refresh()方法。

步骤四： 在 MapGIS.AppLoader.exe 上面加载完 MapGIS.WorkSpace.Plugin.dll，MapBaseOperate.dll 这个 2 个库后，就可以实现该功能。此加载的所有库的路径应该在 VS 程序自己设定的输出目录下库的路径，如果没有设置，则在 LayerFeatureEdit\bin\Debug 中。

1.3 图层要素编辑

1.3.1 示例功能及概述

实现功能：自动打开指定地图文档，并在地图视图中显示地图文档中第一个地图，还有一个名为图层要素编辑的菜单栏可以修改当前地图中处于编辑状态第一个图层的几何参数。

效果预览如下图所示：

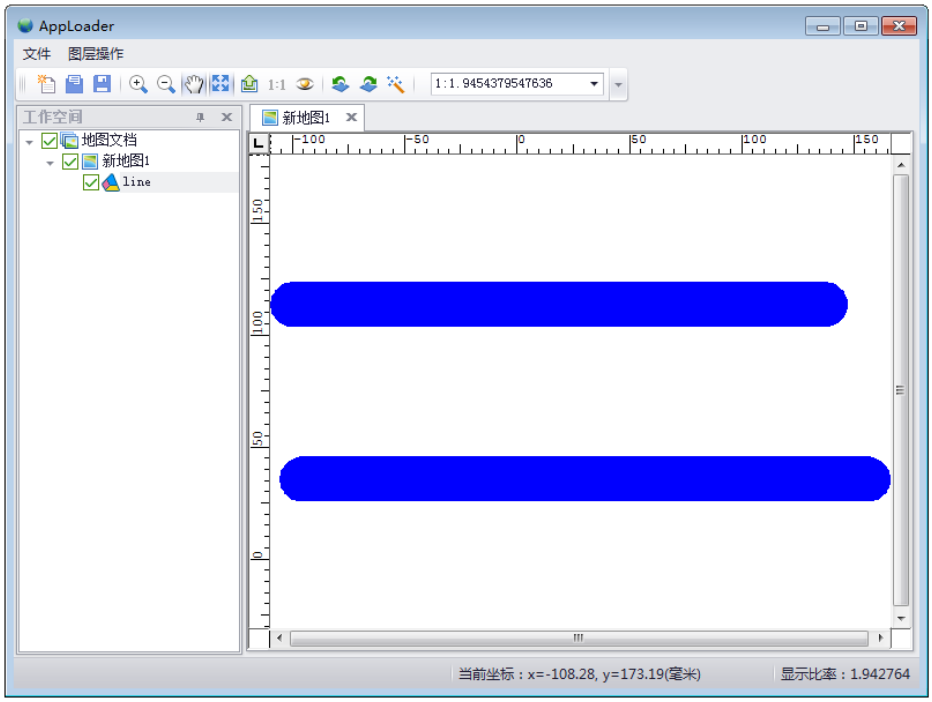

图 1.4 line 类的示意图

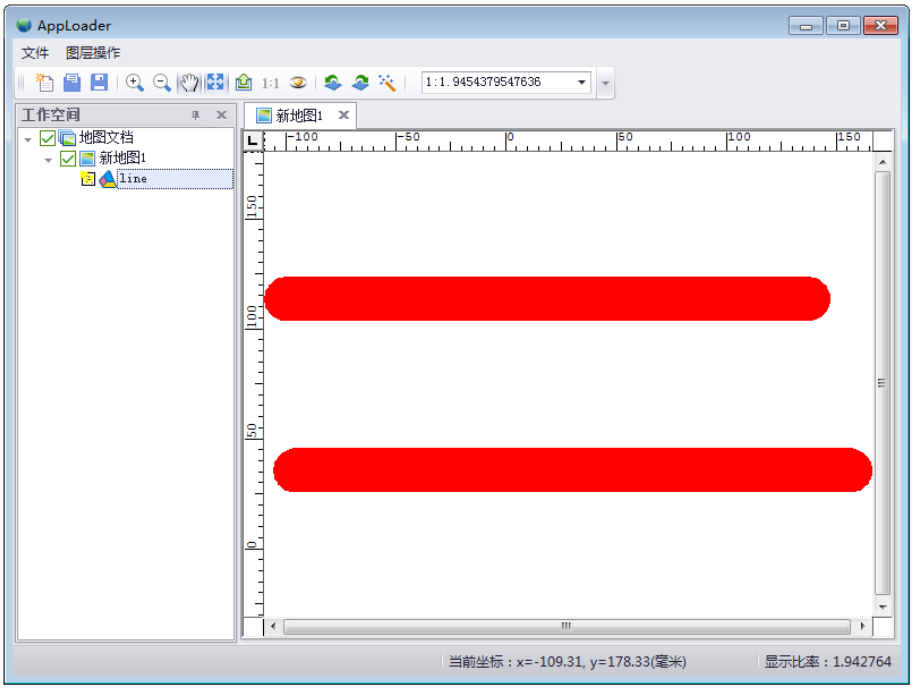

图 1.5 修改 line 线颜色为红色示意图

1.3.2 实现思路及关键代码

插件项：MenuBar（菜单栏）、Command（命令按钮）。

步骤一：打开 VS 2010 新建一个 MapGIS 项目，命名为“LayerFeatureEdit”；在该 MapGIS 插件项目中创建一个 MenuBar 菜单栏，将 Name 属性值设为“LayerMenuBar”，Caption 属性值设为“图层操作”；再添加一个 Command 插件项（命名为 ModifyLayerCmd）实现修改图层参数的功能；具体添加代码请参考 1.2 小节的内容；

步骤二：在 ModifyLayerCmd.cs 文件中实现修改图层参数功能，步骤如下所述。

(1) 获取当前激活地图视图插件对象（IMapContentsView）：调用框架对象 hk 的 ActiveContentsView 属性；

(2) 获取当前激活地图（Map）map：调用 IMapContentsView 的 MapControl 属性获取当前地图视图控件对象（MapControl），再调用 MapControl 的 ActiveMap 属性获取当前激活地图；

(3) 获取图层对应的矢量类（IVectorCls）vecCls：调用 map 对象的 GetEditLayer 方法获取目标图层对象（MapLayer），再调用 MapLayer 的 GetData 方法获取目标图层对应的矢量类；

(4) 修改图层关联的要素实体的图形参数：调用 vecCls 对象的 UpdateInfo 方法；

(5) 在 ModifyLayerCmd.cs 文件中的 OnClick 方法中实现该功能，关键代码如下：

程序代码 1-7 修改图层参数功能的关键代码

```
public void OnClick()
```

```
            {
                //获取当前激活视图的 MapControl 属性
                IMapContentsView mapContView = null;
                mapContView = hk.ActiveContentsView as IMapContentsView;
                if (mapContView == null) return;

                //地图视图控件
                MapControl mapctr = mapContView.MapControl;
                Map map = mapctr.ActiveMap;

                MessageBox.Show("此操作将修改第一个地图的第一个处于激活状态的图层所有要素为红色!!!  ");

                List<MapLayer> maplayer = map.GetEditLayer(EditLayerType.All, SelectLayerControl.Editable);

                if (maplayer == null || maplayer.Count == 0) return;

                //获取激活地图的处于编辑状态的第一个图层对象
                IVectorCls vecCls = (IVectorCls)maplayer[0].GetData();
                if (vecCls == null)
                {
                    MessageBox.Show("当前图层不是简单要素类图层，修改失败!!!  ");
                    return;
                }

                //根据图层几何类型来设置几何信息对象
                LinInfo lineinfo = new LinInfo(); //线
                PntInfo pntinfo = new PntInfo();    //点
                RegInfo reginfo = new RegInfo();    //区

                //图形信息设置
                int[] clr = new int[3];
                clr[0] = 6;
                lineinfo.OutClr = clr;
                pntinfo.OutClr = clr;
                reginfo.FillClr = 6;
```

```
            GeomType gtype = (vecCls as SFeatureCls).GeomType;

            //获取类中所有对象的 OID
            RecordSet rcdset = null;
            QueryDef querydef = new QueryDef();
            rcdset = vecCls.Select(querydef);

            if (rcdset == null) return;

            //更新对象的图形参数信息
            rcdset.MoveFirst();
            for (int i = 1; i <= rcdset.Count; i++)
            {
                if (gtype == GeomType.Lin)
                    vecCls.UpdateInfo(rcdset.CurrentID, lineinfo);
                if (gtype == GeomType.Pnt)
                    vecCls.UpdateInfo(rcdset.CurrentID, pntinfo);
                if (gtype == GeomType.Reg)
                    vecCls.UpdateInfo(rcdset.CurrentID, reginfo);

                rcdset.MoveNext();
            }
            mapctr.Restore();
        }
```

步骤三：在 MapGIS.AppLoader.exe 上面加载完 MapGIS.WorkSpace.Plugin.dll，LayerFeatureEdit.dll 这个 2 个库后，就可以实现该功能。此加载的所有库的路径应该在 VS 程序自己设定的输出目录下库的路径，如没有设置，则是在 LayerFeatureEdit\bin\Debug 中。

第2章 进阶应用

2.1 自定义地图视图

2.1.1 示例功能概述

实现功能：打开地图文档，并显示该地图文档里的第一个地图。

效果预览如下图所示：

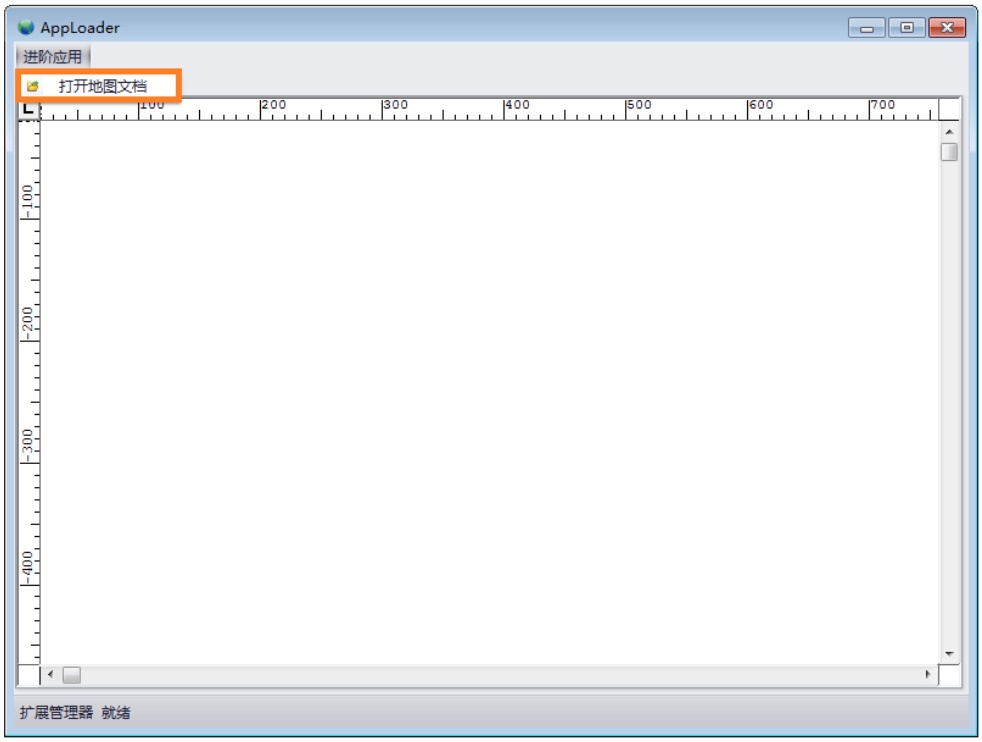

图 2.1 开始界面

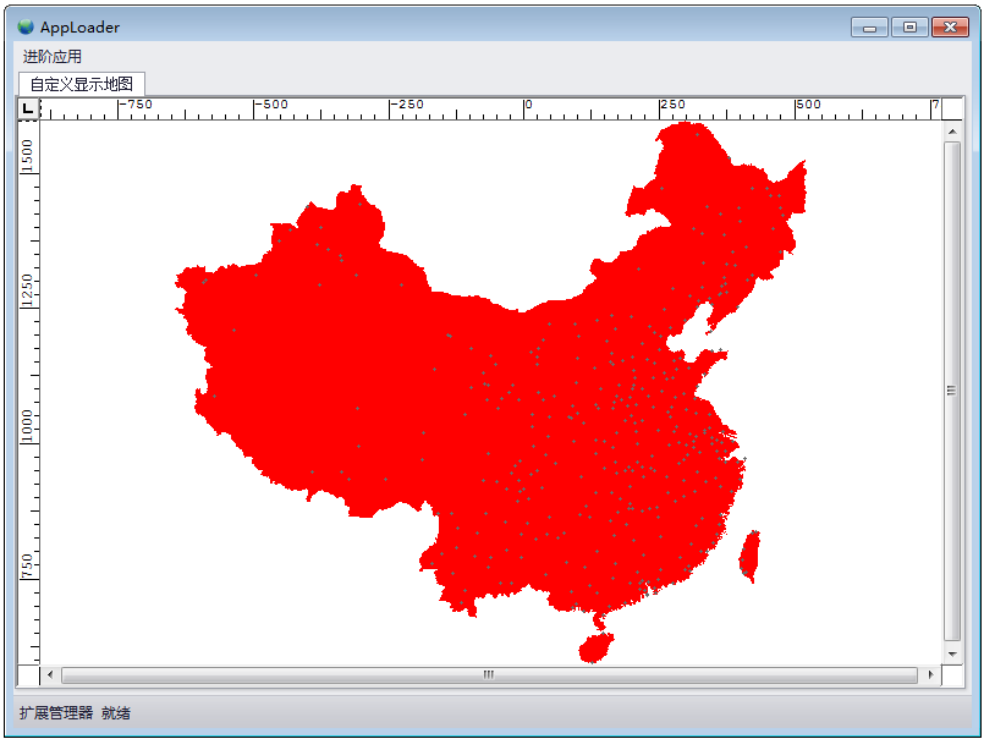

图 2.2 打开地图文档示意图

2.1.2 实现思路及关键代码

插件项：MenuBar（菜单栏）、Command（命令按钮）、DockWindow（停靠窗口）、MapContentsView（地图视图）。

步骤一：打开 VS 2010 新建一个 MapGIS 项目，命名为“CustomMapViewDemo”，在该 MapGIS 插件项目中创建一个 MenuBar 菜单栏，将 Name 属性值设为“MapOperationMB”，Caption 属性值设为“进阶应用”；添加一个 Command 插件项（命名为“OpenDocCmd”）打开地图文档视图功能；添加一个 MapContentsView 插件项（命名为“MapContentsView”）用于承载地图视图控件；在 VS 2010 中自定义一个用户控件，命名为“UserControl1”，在控件窗体中拖入一个地图视图 MapControl 控件，设置其 Modifiers 属性为 public，Dock 属性为 Fill；

步骤二：将 OpenDocCmd 命令按钮插件项添加到 MapOperationMB 菜单项中，具体实现代码请参考 1.2 小节的内容；

步骤三：将 UserControl1 自定义用户控件绑定到 MapContentsView 地图视图插件项中，具体实现步骤如下。

(1) 在 MapContentsView.cs 文件中定义一个自定义用户控件（UserControl1 对象）userControl 的全局变量；

(2) 修改 MapContentsView.cs 文件中的 ObjecthWnd 属性代码如下所示：

程序代码 2-1 修改 ObjecthWnd 属性代码

```
public Control ObjecthWnd
 {
```

```
            get { return userControl; }
        }
```

(3) 修改 MapControl 属性代码如下所示之后就完成了绑定功能。

程序代码 2-2 修改 MapControl 属性代码

```
    public MapGIS.GISControl.MapControl MapControl
    {
        get { return userControl.mapControl1; }
    }
```

步骤四：在 OpenDocCmd.cs 文件中实现打开地图文档，并显示该地图文档里的第一个地图功能，具体步骤如下。

(1) 在 OpenDocCmd.cs 文件中定义一个应用框架（IApplication 对象）hk 全局变量，定义一个地图视图控件（MapControl 对象）mapCtrl 全局变量，获取插件容器中的地图视图插件 MapContentsView 对象，然后根据地图视图插件 MapContentsView 对象来获取对应的 MapContorl 地图视图控件对象，在 OnCreate 方法中的关键代码如下：

程序代码 2-3 获取 MapControl 控件关键代码

```
public void OnCreate(IApplication hook)
        {
            hk = hook;

            //内容视图控件
            IContentsView cv = null;
            //获取指定的内容视图控件
hk.PluginContainer.ContentsViews.TryGetValue("CustomMapViewDemo.MapContentsView", out
cv);
            //获取地图视图插件项
            IMapContentsView mapview = cv as IMapContentsView;

            //获取该视图的 MapControl 属性
            mapCtrl = mapview.MapControl;
        }
```

(2) 打开地图文档，将地图文档中的第一个地图显示到 MapControl 中，在 OnClick 方法中关键代码如下：

程序代码 2-4 显示地图关键代码

```
    public void OnClick()
        {
            Map map = null;
            string docName = null;
```

```
            if (mapCtrl == null) return;
            Document doc = this.hk.Document;

            //选择打开地图文档
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Multiselect = true;
            ofd.Filter = "(地图文档 mapx)|*.mapx|(地图 map)|*.map";

            if (ofd.ShowDialog() == DialogResult.OK)
            {
                //获取打开的地图文档的名称
                docName = ofd.FileName;
            }
            if (doc == null) return;

            doc.Open(docName);
            Maps maps = doc.GetMaps();
            if (maps.Count > 0)
            {
                //获取当前第一个地图
                map = maps.GetMap(0);
                //设置地图的第一个图层为激活状态
                map.get_Layer(0).State = LayerState.Active;
                mapCtrl.ActiveMap = map;
                mapCtrl.Restore();
            }
        }
```

步骤五：在 MapGIS.AppLoader.exe 上面加载完 MapGIS.WorkSpace.Plugin.dll 此库后，用户单击“打开地图文档”，打开地图文档，并显示该地图文档里的第一个地图。

2.2 交互式几何查询

2.2.1 示例功能概述

实现功能：用户在默认地图窗口中拉框查询，闪烁查询结果，属性列表中显示当前被查询中的第一个激活图层中的记录属性。

效果预览如下图所示：

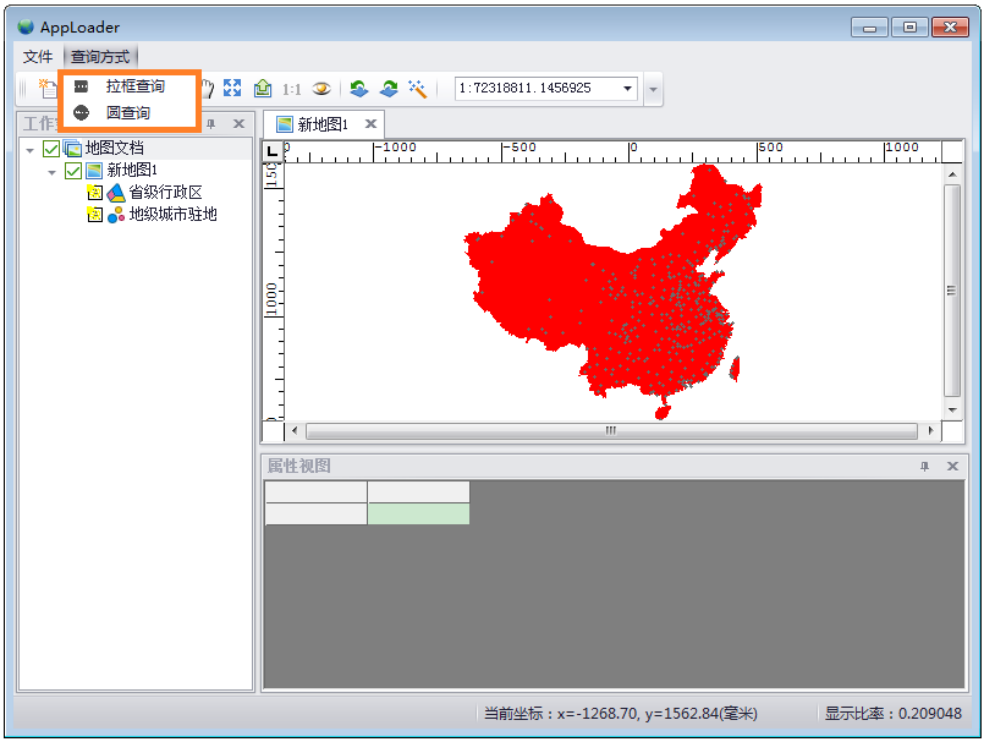

图 2.3 打开一幅地图界面示意图

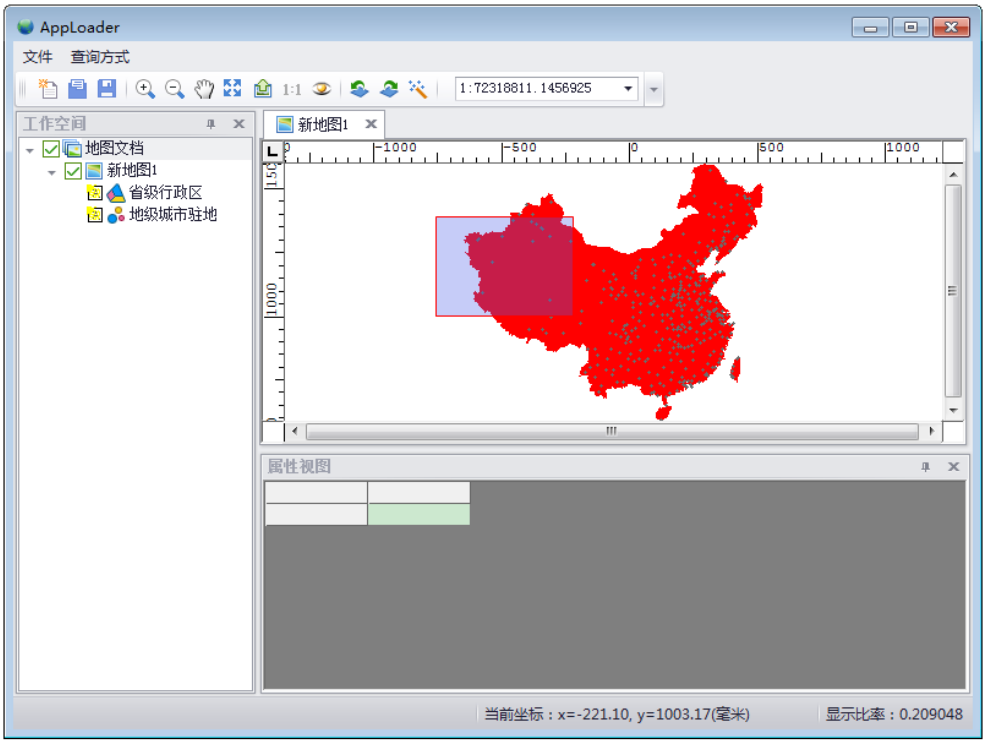

图 2.4 拉框查询示意图

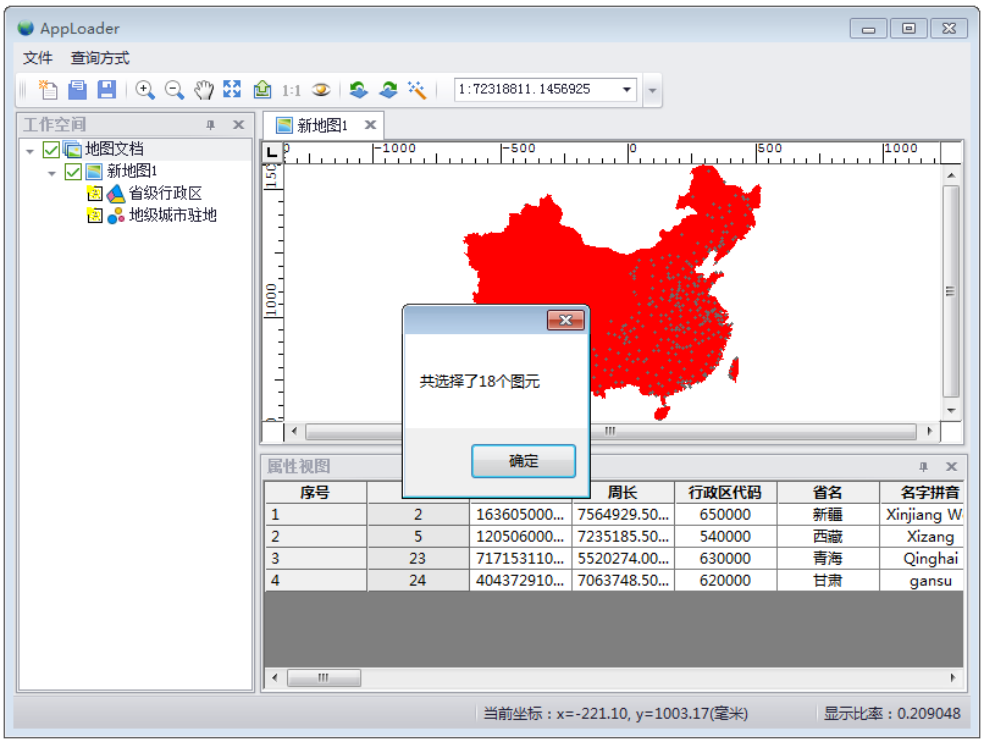

图 2.5 查询省级行政区结果图

2.2.2 实现思路及关键代码

插件项：MenuBar（菜单栏）、Command（命令按钮）、DockWindow（停靠窗口）。

步骤一：打开 VS 2010 新建一个 MapGIS 项目，命名为“InterGeomQuery”，在该 MapGIS 插件项目中创建一个 MenuBar 菜单栏，将 Name 属性值设为“SetBasToolMBar”，Caption 属性值设为“查询方式”；添加 2 个 Command 插件项（命名为“QueryByRectCmd”，“QueryByCirCmd”）分别实现矩形查询和圆查询的功能；添加一个 DockWindow 插件项（命名为“DockWindow”）用于承载属性视图控件；在 VS 2010 中自定义一个用户控件，命名为“AttUserControl”，在控件窗体中拖入一个属性视图 AttControl 控件，设置其 Modifiers 属性为 public，Dock 属性为 Fill；

步骤二：将 QueryByRectCmd 和 QueryByCirCmd 两个命令按钮插件项添加到 SetBasToolMBar 菜单项中，具体实现代码请参考 1.2 小节的内容；

步骤三：将 AttUserControl 自定义用户控件绑定到 DockWindow 停靠窗口插件项中，具体实现步骤如下。

(1) 在 DockWindow.cs 文件中自定义一个属性视图控件（AttUserControl 对象）attCtrl 全局变量；

(2) 修改 ChildHWND 属性字段代码如下所示，即可实现绑定功能：

程序代码 2-5 修改 ChildHWND 属性字段代码

```
public Control ChildHWND
 {
```

```
            get { return attCtrl; }
        }
```

步骤四：创建自定义类，命名为“SelectToolClass”，该类继承并重写 GISBasTool 类，在 SelectToolClass 类定义一个选择工具控件（SelectTool 对象 ）selTool 全局变量，注册 selTool 的选择事件，并在事件中完成查询功能，关键代码如下：

程序代码 2-6 SelectToolClass 类中的全局变量

```
        //地图视图控件
        MapControl mapCtrl;
        //交互工具：选择工具控件
        SelectTool selTool;
        //选择数据时的数据类型过滤
        SelectDataType dataType;
        //地理类对象基类接口
        IBasCls basClass = null;
        //结果集对象
        RecordSet rcdSet = null;
        //属性视图控件
        AttControl attCtrl = null;
        //查询选择方式：圆选择、多边形选择、矩形选择
        SelectType selType;
```

程序代码 2-7 SelectToolClass 类的构造函数代码

```
    /// <summary>
        /// 实现拉框选择查询
        /// </summary>
        /// <param name="control">地图视图控件</param>
        /// <param name="dataType">选择数据时的数据类型过滤</param>
        /// <param name="attctr">属性视图控件</param>
        /// <param name="seltype">查询选择方式：矩形选择 </param>
        public SelectToolClass(MapControl control, SelectDataType dataType, AttControl
attctr,SelectType seltype)
            : base()
        {

            this.mapCtrl = control;
            this.dataType = dataType;
            this.attCtrl = attctr;
            this.selType = seltype;
```

```
                    //查询选择项
                    SelectOption selOpt = new SelectOption();
                    selOpt.DataType = dataType;    //选择数据时的类型过滤类型
                    selOpt.SelMode = SelectMode.Multiply; //选择模式
                    selOpt.UnMode = UnionMode.Copy;    //结果数据合并模式
                    selOpt.LayerCtrl = SelectLayerControl.Editable; //选择数据时的图层过滤类
型


                    //创建圆交互工具
                    selTool = new SelectTool(control, selType, selOpt,
SpaQueryMode.MBRIntersect, control.Transformation);
                    //注册选择事件
                    selTool.Selected += new SelectTool.SelectHandler(selTool_Selected);
                    。。。。。。
                }
```

程序代码 2-8 SelectTool 的 Selected 事件响应关键代码

```
        void selTool_Selected(object sender, SelectEventArgs e)
            {
                if (e.SelSet != null)
                {
                    this.mapCtrl.FlashSelectSet();
                    int objCount = getSelectSetCount(e.SelSet);
                    MessageBox.Show("共选择了" + objCount + "个图元");
                }
            }
```

程序代码 2-9 获取选择集中的个数中的关键代码

```
            /// <summary>
            ///  获取选择集中的个数
            /// </summary>
            /// <param name="set">选择集对象</param>
            /// <returns>选择集中的个数</returns>
            public int getSelectSetCount(SelectSet set)
             {
                int count = 0;
                //记录符合选择项的 ids
                ObjectIDs oids = new ObjectIDs();
                ObjectID oid = new ObjectID();
```

```
            if (set != null)
            {
                //获取选择集列表
                List<SelectSetItem> lst = set.Get();

                foreach (SelectSetItem item in lst)
                {
                    count += item.IDList.Count;
                }
                if (lst == null || lst.Count == 0) return count;

                //获取图层信息
                MapLayer maplayer = lst[0].Layer;
                //获取图层对应的要素类的信息
                basClass = maplayer.GetData();

                //获取处于编辑状态第一个图层的要素 ID 列表
                List<long> idArr = lst[0].IDList;

                for (int i = 0; i < idArr.Count; i++)
                {
                    oid.Int64Val = idArr[i];
                    oids.Append(oid);
                }
                rcdSet = new RecordSet(basClass);
                //添加结果集
                rcdSet.AddSet(oids);
            }
            attCtrl.SetXCls((IVectorCls)basClass, rcdSet);
            return count;
        }
```

步骤四：实现查询功能，以矩形查询为例进行说明，在 QueryByRectCmd 插件中的 OnCreate 方法中获取当前地图视图控件 MapControl 对象，以及获取 DockWindow 插件所绑定的属性视图控件 AttControl 对象，具体代码如下所示：

程序代码 2-10 OnCreate 方法关键代码

```
    public void OnCreate(IApplication hook)
```

```
        {
            hk = hook;

            //获取 DockWindow 插件
hk.PluginContainer.DockWindows.TryGetValue("InterGeomQuery.DockWindow", out dw);
            //通过停靠窗口来获取用户控件中的属性控件对象
            attCtrl = (dw.ChildHWND as AttUserControl).attControl1;

            //获取当前激活视图的 MapControl 属性
            IMapContentsView mapConView = null;
            mapConView = hk.ActiveContentsView as IMapContentsView;
            if (mapConView == null) return;

            mapCtrl = mapConView.MapControl;
            mapCtrl.Restore();
        }
```

步骤五：在 QueryByRectCmd 插件中的 OnClick 方法中实现矩形查询，具体代码如下所示：

程序代码 2-11 OnClick 方法中的关键代码

```
        public void OnClick()
        {
            //设置为拉框查询
            SelectType seltype = SelectType.Rectangle;

            IMapContentsView mapConView = null;
            mapConView = hk.ActiveContentsView as IMapContentsView;
            if (mapConView == null) return;
            //获取当前激活视图的 MapControl 属性
            mapCtrl = (hk.ActiveContentsView as IMapContentsView).MapControl;

            //创建拉框选择类对象
            SelectToolClass basTool = new SelectToolClass(mapCtrl,
SelectDataType.Anyone, attCtrl, seltype);
            mapCtrl.SetBasTool(basTool);

            if (attCtrl == null) return;
            this.attCtrl.AttLButtonDown += new
AttControl.AttLBtDownEventHandler(attctrl_AttLButtonDown);
```

```
        }
```

步骤六：在 MapGIS.AppLoader.exe 上面加载完 MapGIS.WorkSpace.Plugin.dll，InterGeomQuery.dll 这个 2 个库后，就可以实现查询功能。

2.3 插件间通讯

2.3.1 示例功能概述

实现功能：插件间通讯，主要是调用目标插件的资源，如方法等；本示例功能为通过插件容器获取 MapGIS.WorkSpace.Plugin 插件中的 CmdOpen 命令按钮插件来打开一个地图文档。

效果预览如下图所示：

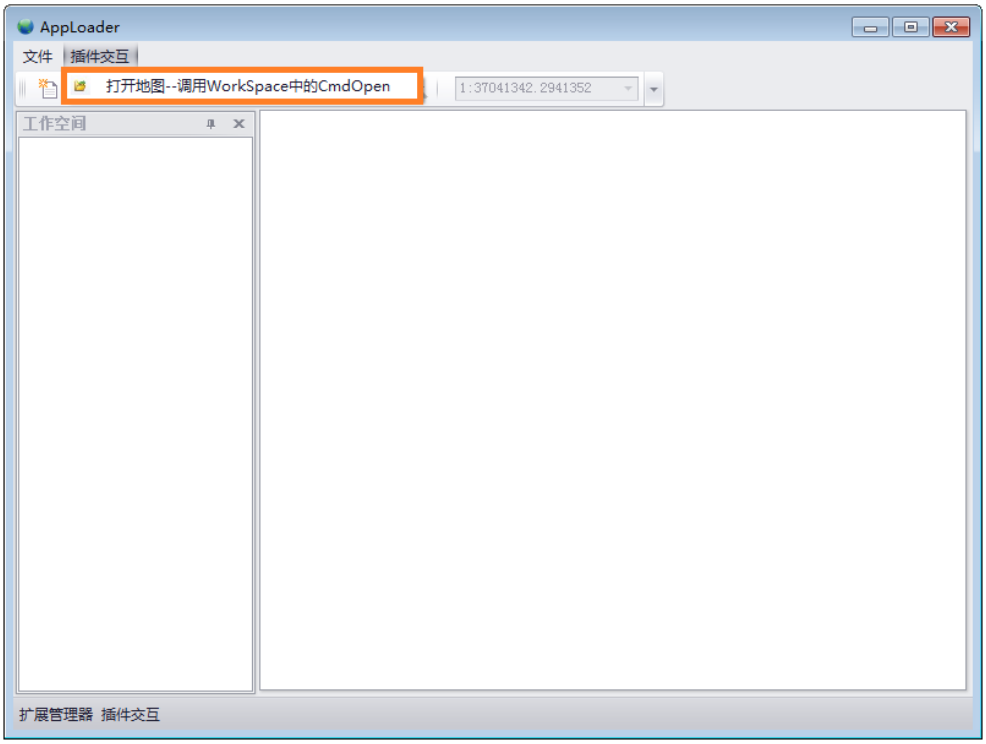

图 2.6 插件间通讯示意图

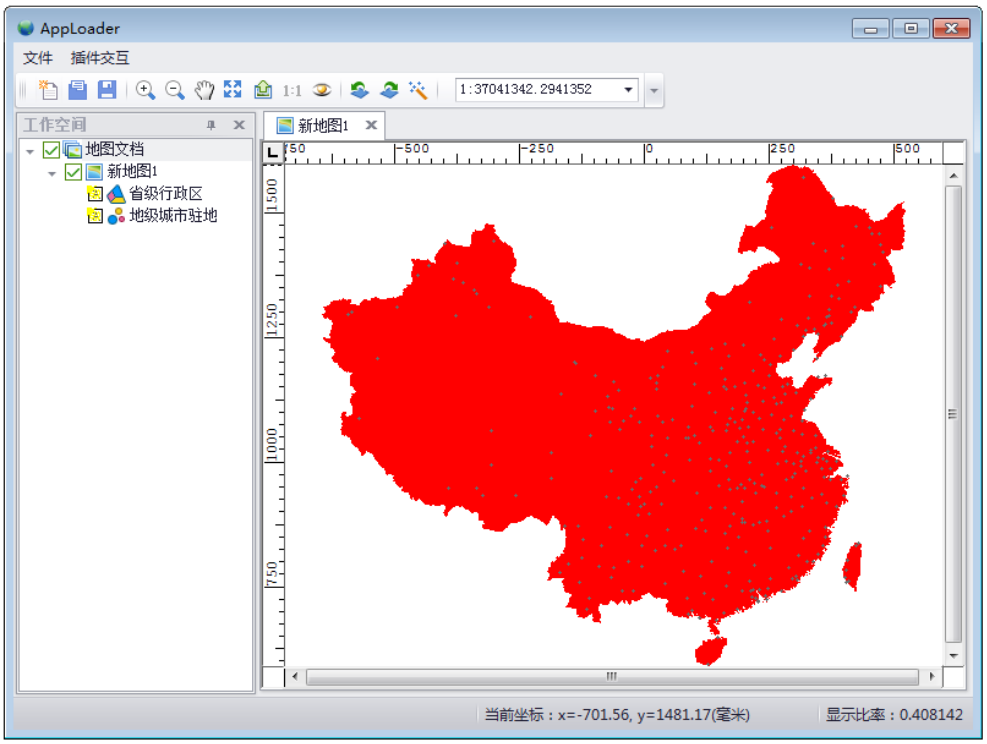

图 2.7 点击命令按钮示意图

2.3.2 实现思路及关键代码

插件项：MenuBar（菜单栏）、Command（命令按钮）。

步骤一：打开 VS 2010 新建一个 MapGIS 项目，命名为“PluginInteraction”，在该 MapGIS 插件项目中创建一个 MenuBar 菜单栏，将 Name 属性值设为“PluginMenuBar”，Caption 属性值设为“插件交互”；添加 1 个 Command 插件项（命名为“OpenCmd”）实现打开地图文档的功能；

步骤二：将 QueryByRectCmd 命令按钮插件项添加到 PluginMenuBar 菜单项中，具体实现代码请参考 1.2 小节的内容；

步骤三：定义一个命令按钮插件（ICommand 对象）cmd，通过插件容器获取 MapGIS.WorkSpace.Plugin 插件中的 CmdOpen 命令按钮插件，将该插件传给 cmd 对象，调用 cmd 插件的 OnClick 方法打开地图文档，关键代码如下所示：

程序代码 2-12 OnClick 方法实现打开地图文档代码

```
        public void OnClick()
            {
                //命名按钮插件对象
                ICommand cmd = null;
                //通过插件容器获取 MapGIS.WorkSpace.Plugin 插件中的 CmdOpen 命令按钮插件
hk.PluginContainer.Commands.TryGetValue("MapGIS.WorkSpace.Plugin.CmdOpen", out cmd);
```

```
                    //获取成功
                    if (cmd != null)
                    {
                        //打开地图文档
                        cmd.OnClick();

                        //获取打开的地图文档对象
                        Document doc = hk.Document;
                        //获取第一个地图
                        Map map = doc.GetMaps().GetMap(0);
                        if (map == null) return;

                        //在地图视图上显示第一个地图
hk.WorkSpaceEngine.FireMenuItemClickEvent("MapGIS.WorkSpace.Style.PreviewMap", map);
                }
```

步骤三：在 MapGIS.AppLoader.exe 上面加载完 MapGIS.WorkSpace.Plugin.dll，CustomMapViewDemo.dll 和 MapGIS.MapEditor.Plugin.dll 这个 2 个库后，就可以实现查询功能。

2.4 自定义工作空间

2.4.1 示例功能概述

实现功能：自定义工作空间的开发，涉及停靠窗口（IDockWindw）插件的开发，以及工作空间目录树中地图节点的显示，目录树右键菜单的维护等；主要借助平台提供的 MapGIS.UI.Controls 程序集里的 MapWorkSpaceTree 类实现目录树的构建。

效果预览如下图所示：

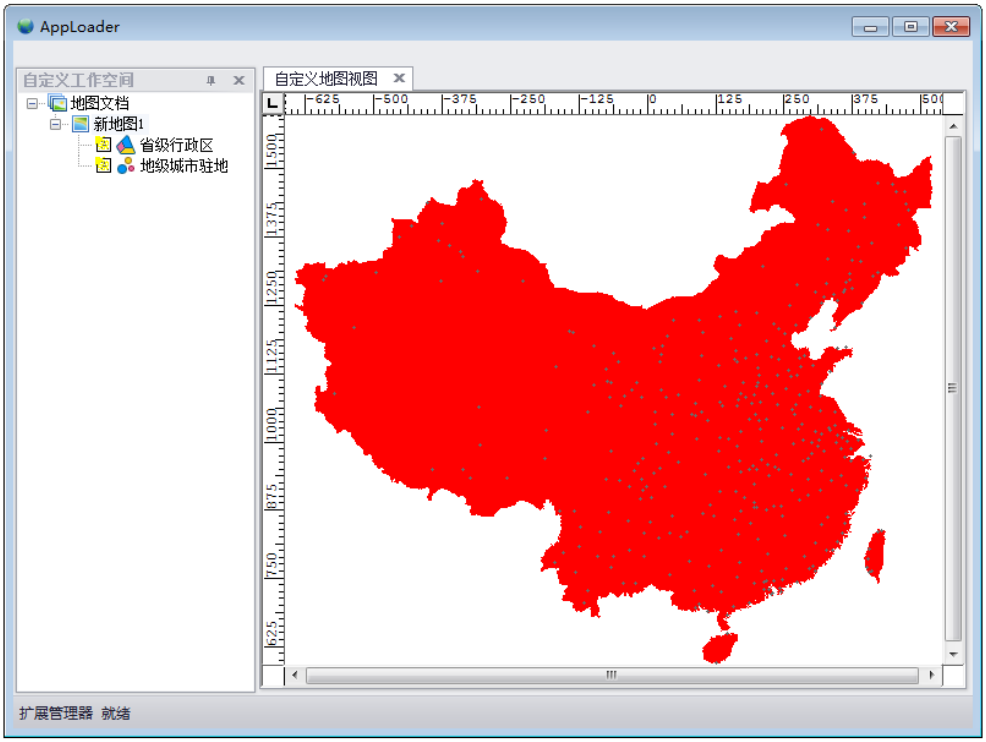

图 2.8 自定义工作空间示意图

2.4.2 实现思路及关键代码

步骤一：打开 VS 2010 新建一个 MapGIS 项目，命名为“CustomWorkspaceTree”，在该 MapGIS 插件项目中创建一个 IMapContentsView 地图视图插件，将 Name 属性值设为“MapView”，Caption 属性值设为“自定义地图视图”；添加一个 IDockWindow 插件项将 Name 属性值设为“WorkspaceTree”，Caption 属性值设为“自定义工作空间”；添加 2 个自定义用户控件，设置 Name 属性值分别为“MapViewCtrl”、“TreeView”；在“MapViewCtrl”自定义控件中拖入一个 MapControl 地图视图控件，设置 MapControl 地图视图控件 Name 属性值为“mapCtrl”；在“TreeView”自定义控件中拖入一个 MapWorkSpackTree 工作空间树控件，设置 MapWorkSpackTree 工作空间树控件的 Name 属性值为“m_tree”；

步骤二：将“MapViewCtrl”自定义控件绑定到“MapView”地图视图插件中，具体实现步骤如下：

(1) 在 MapView.cs 文件中定义一个自定义用户控件（MapViewCtrl 对象）mapViewCtrl 的全局变量；

(2) 修改 MapView.cs 文件中的 ObjecthWnd 属性代码如下所示：

程序代码 2-13 修改 ObjecthWnd 属性代码

```
public Control ObjecthWnd
 {
      get { return mapViewCtrl; }
 }
```

(3) 修改 MapView.cs 文件中的 MapControl 属性代码如下所示之后就完成了绑定功能。

程序代码 2-14 修改 MapControl 属性代码

```
public MapGIS.GISControl.MapControl MapControl
        {
            get { return mapViewCtrl. mapCtrl; }
        }
```

步骤三：将“TreeView”自定义用户控件绑定到“WorkspaceTree”停靠窗口插件项中，具体实现步骤如下。

(1) 在 DockWindow.cs 文件中定义一个自定义控件（TreeView 对象）tree 全局变量；

(2) 修改 ChildHWND 属性字段代码如下所示，即可实现绑定功能：

程序代码 2-15 修改 ChildHWND 属性字段代码

```
public Control ChildHWND
        {
            get { return tree; }
        }
```

步骤四：在 TreeView.cs 文件中实现自定义工作空间树的相应功能，实现自定义工作空间具体步骤为：

(1) 定义一个工作空间树控件（MapWorkSpackTree 对象）m_tree，添加该树的右键菜单事件处理，关键代码如下：

程序代码 2-16 TreeView.cs 文件中的全局变量

```
#region 变量定义
//工作空间树
MapWorkSpackTree m_tree = null;
//框架对象
IApplication app = null;
//地图视图控件
MapControl mapCtrl = null;
//地图集合对象
Maps maps = null;
#endregion
```

程序代码 2-17 TreeView 自定义控件加载事件代码

```
private void TreeView_Load(object sender, EventArgs e)
        {
            try
            {
                //获取框架对象
                app = WorkspaceTree.hk;
                //加载树
                m_tree = new MapWorkSpackTree();
```

```
                m_tree.Dock = DockStyle.Fill;
                //添加到用户控件
                this.Controls.Add(m_tree);

                //注册节点右键菜单单击事件
                m_tree.MenuItemOnClickEvent += new
MapGIS.WorkSpaceEngine.MenuItemOnClickHandler(m_tree_MenuItemOnClickEvent);
                。。。。。。
                m_tree.Document.Title = "地图文档";
            }
            catch(Exception ex){
                MessageBox.Show(ex.ToString());
            }
        }
```

(2) 实现目录树上的地图节点右键菜单预览地图的功能，在右键菜单单击事件中实现关键代码如下所示：

程序代码 2-18 m_tree_MenuItemOnClickEvent 关键代码

```
    void m_tree_MenuItemOnClickEvent(string typeName, MapGIS.GeoMap.DocumentItem
item)
        {
            try
            {
                #region 预览地图
                if (item is Map)
                {
                    Map map = item as Map;
                    //内容视图插件
                    IContentsView icv = null;
                    //获取地图视图
app.PluginContainer.ContentsViews.TryGetValue(map.Handle.ToString() + "$MapView", out icv);
                    if (icv == null)
                    {
                        //创建自定义地图视图插件
                        icv =
app.PluginContainer.CreateContentsView("CustomWorkspaceTree.MapView",
map.Handle.ToString() + "$MapView");
                        if (icv != null && icv is IMapContentsView)
                        {
```

```
                                    IMapContentsView mv = icv as IMapContentsView;
                                    //获取地图视图控件
                                    mapCtrl = mv.MapControl;
                                    //激活地图
                                    mv.MapControl.ActiveMap = map;
                                    //设置地图显示控件
                                    m_tree.WorkSpace.SetMapControl(map, mv.MapControl);
                                    //获取地图显示范围
                                    Rect rect = map.GetViewRange();
                                    if (rect != null && (rect.XMax - rect.XMin).CompareTo(0) >
0 && (rect.YMax - rect.YMin).CompareTo(0) > 0)
                                        mv.MapControl.Transformation.SetDispRect(rect);
                                    mv.MapControl.Refresh();
                                    app.StateManager.OnStateChanged(this, new
StateEventArgs());
                                }
                            }
                            else
                            {
                                //如果自定义地图视图插件已经打开
                                IMapContentsView mv = icv as IMapContentsView;
                                if (mv != null)
                                {
                                    //激活地图视图
                                    app.PluginContainer.ActiveContentsView(mv);
                                    Rect rt = map.GetEntireRange();
                                    Rect rt1 = map.GetViewRange();
                                    。。。。。。
                                    mv.MapControl.Refresh();
                                }
                            }
                        }
                        #endregion
                    }
                    catch(Exception ex){
                        MessageBox.Show(ex.ToString());
                    }
```

```
        }
```

步骤五：在地图右键菜单中添加一个自定义菜单，具体步骤为：

(1) 在 TreeView.cs 文件中定义一个自定义菜单类 CustomGloableMenu，该类继承并实现 ISingleMenuItem 接口，具体代码如下：

程序代码 2-19 CustomGloableMenu 类中代码

```
    #region 单选右键菜单项类
    class CustomGloableMenu : ISingleMenuItem
    {
        //新建地图文档

        #region ISingleMenuItem 成员
        public bool BeginGroup
        {
            get { return false; }
        }
        public Bitmap Bitmap
        {
            get { return null; }
        }
        public string Caption
        {
            get { return "自定义菜单"; }
        }
        public bool Checked
        {
            get { return false; }
        }
        public bool Enabled
        {
            get { return true; }
        }
        public bool Visible
        {
            get { return true; }
        }
        public void OnClick(DocumentItem item)
        {
```

```
                MessageBox.Show("自定义菜单演示！");
            }
            public void OnCreate(MapGIS.WorkSpaceEngine.IWorkSpace ws)
            {
            }
            #endregion
        }
        #endregion
```

(2) 调用 m_tree 类的 WorkSpace 属性获取工作空间 WorkSpace 对象，再调用 WorkSpace 对象的 GetMenuExtand 方法获取右键菜单项，将返回值传给菜单扩展（IMenuExtander 对象）ime；定义一个 CustomGloableMenu 类对象，并初始化，调用 ime 对象的 AddItem 方法将 CustomGloableMenu 类添加到右键菜单栏中，在 Document_OpenedDocument 打开地图文档事件中实现该功能，具体代码如下：

程序代码 2-20 自定义菜单实现具体代码

```
        void Document_OpenedDocument(object sender, EventArgs e)
        {
            //为地图节点添加右键菜单项
            IMenuExtander ime =
this.m_tree.WorkSpace.GetMenuExtand(typeof(MapGIS.GeoMap.Map));
            //自定义菜单对象
            CustomGloableMenu menuItem = new CustomGloableMenu();
            menuItem.OnCreate(this.m_tree.WorkSpace);
            //将自定义菜单添加到右键菜单项
            ime.AddItem(menuItem);
        }
```

效果如下图所示：

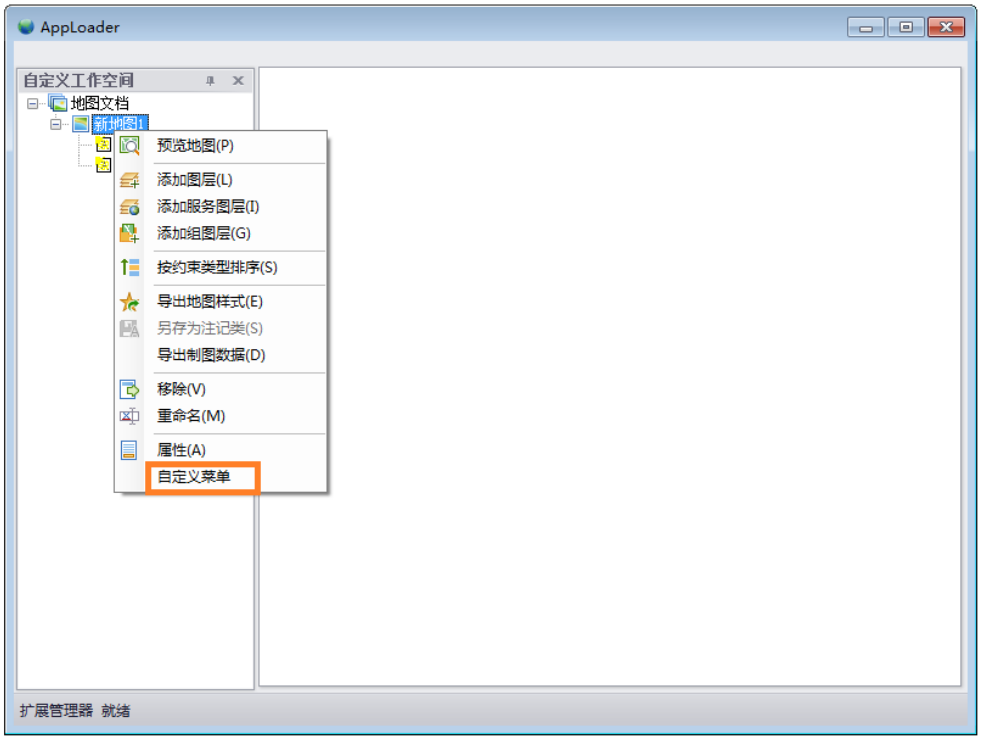

图 2.9 自定义菜单效果图

步骤五：在 MapGIS.AppLoader.exe 上面加载完 CustomWorkspaceTree.dll 这个库后，就可以实现自定义工作空间功能。