# MAPGIS7-IMS 行业版
# 程序设计高级篇

# 目录

目录 ..... - 2 -
示例站点搭建详解 ..... - 3 -
1 GIS常用术语与定义 ..... - 3 -
2 开发前的准备工作 ..... - 5 -
3 搭建开发环境 ..... - 5 -
4 示例站点简介 ..... - 5 -
5 示例开发与实现(.NET版) ..... - 6 -
5.1 站点搭建过程 ..... - 6 -
5.2 站点具体功能实现 ..... - 17 -
5.2.1 基本显示功能 ..... - 17 -
5.2.1.1 地图显示 ..... - 17 -
5.2.1.2 基本操作 ..... - 23 -
5.2.2 放大镜功能 ..... - 27 -
5.2.3 鹰眼显示功能 ..... - 29 -
5.2.4 地图测量功能 ..... - 32 -
5.2.5 系统设置功能 ..... - 34 -
5.2.6 地图搜索功能 ..... - 37 -
5.2.7 图层控制功能 ..... - 41 -
5.2.8 地图查询功能 ..... - 47 -
5.2.9 地图编辑功能 ..... - 69 -
5.2.10 空间分析功能 ..... - 79 -
5.2.11 统计分析功能 ..... - 94 -
5.2.12 坐标转换功能 ..... - 100 -
5.2.13 面积周长计算功能 ..... - 103 -
6 示例开发与实现(JAVA版) ..... - 108 -
6.1 站点搭建过程 ..... - 108 -
6.2 站点具体功能实现 ..... - 120 -
6.2.1 基本显示功能 ..... - 120 -
6.2.1.1 地图显示 ..... - 120 -
6.2.1.2 基本操作 ..... - 129 -
6.2.2 放大镜功能 ..... - 134 -
6.2.3 鹰眼显示功能 ..... - 136 -
6.2.4 地图测量功能 ..... - 137 -
6.2.5 系统设置功能 ..... - 138 -
6.2.6 地图搜索功能 ..... - 142 -
6.2.7 图层控制功能 ..... - 147 -
6.2.8 地图查询功能 ..... - 154 -
6.2.9 地图编辑功能 ..... - 174 -
6.2.10 空间分析功能 ..... - 182 -
6.2.11 统计分析功能 ..... - 197 -
6.2.12 坐标转换功能 ..... - 205 -
6.2.13 面积周长计算功能 ..... - 209 -

# 示例站点搭建详解

# 1 GIS 常用术语与定义

**地理数据：**

指表征地理圈或地理环境固有要素或物质的数量、质量、分布特征、联系和规律的数字、文字、图像和图形等的总称。

**空间数据：**

是指以地球表面空间位置为参照的自然、社会和人文经济景观数据，可以是图形、图像、文字、表格和数字等。

**要素：**

要素是对现实世界中现象的抽象，也是真实世界中的地理对象在地图上的表示；要素具有几何和属性。

**要素 ID：**

要素的唯一标识。

**缓冲区：**

所谓缓冲区就是地理空间目标的一种影响范围或服务范围。

**缓冲区分析：**

从数学的角度看，缓冲区分析的基本思想是给定一个空间对象或集合和半径，确定它们的邻域。邻近度（Proximity）描述了地理空间中两个地物距离相近的程度，其确定是空间分析的一个重要手段。交通沿线或河流沿线的地物有其独特的重要性，公共设施（商场、邮局、银行、医院、车站、学校等）的服务半径，大型水库建设引起的搬迁，铁路、公路以及航运河道对其所穿过区域经济发展的重要性等，均是一个邻近度问题。缓冲区分析是解决邻近度问题的空间分析工具之一 。

图 8-1-1 为点对象、线对象、面对象及对象集合的缓冲区示例。



图 8-1-1(a) 点对象缓冲区

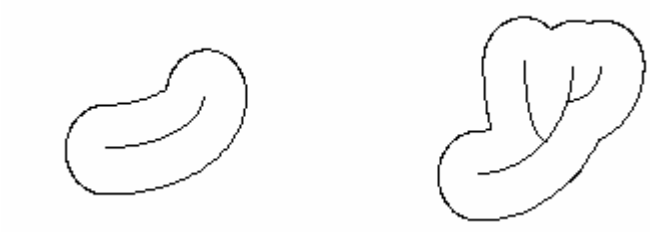


图 8-1-1(b) 线对象缓冲区



图 8-1-1(c) 多边形的缓冲区

**叠加分析：**

叠加分析是地理信息系统最常用的提取空间隐含信息的手段之一。该方法源于传统的透明材料叠加，即将来自不同的数据源的图纸绘于透明纸上，在透光桌上将其叠放在一起，然后用笔勾出感兴趣的部分，提取出感兴趣的信息。地理信息系统的叠加分析是将有关主题层组成的数据层面，进行叠加产生一个新数据层面的操作，其结果综合了原来两层或多层要素所具有的属性。

**网络分析：**

对地理网络（如交通网络）、城市基础设施网络（如各种网线、电力线、电话线、供排水管线等）进行地理分析和模型化，是地理信息系统中网络分析功能的主要目的。

网络数据模型是真实世界中网络系统（如交通网、通信网、自来水管网等）的抽象表示。构成网络的最基本元素是线性实体以及这些实体的连接交汇点。前者常被称为网线或链，后者一般称为节点。节点中又有下面几种特殊的类型。

- 障碍（Barrier），禁止网络中链上流动的点。
- 拐点（Turn），出现在网络链中的分割结点上，状态属性有阻力，如拐弯的时间和限制（如在 8：00 到 18：00 不允许左拐）。
- 中心（Center），是接受或分配资源的位置，如水库、商业中心、电站等，其状态属性包括资源容量（如总量），阻力限额（中心到链的最大距离或时间限制）。
- 站点（Stop），在路径选择中资源增减的节点，如库房、车站等，其状态属性有资源需求，如产品数量。

**路径分析：**

路径分析是网络分析中的一种，其核心是对最佳路径和最短路径的求解。例如救护车需要了解从医院到病人家里走哪条路最快；旅客往往要在众多航线中找到费用最小的中转方案，这些都是最佳路径求解的例子。

**拓扑分析：**

拓扑一词来自于希腊文，意思是“形状的研究”。拓扑学是几何学的一个分支，它研究在拓扑变换下能够保持不变的几何属性——拓扑属性。为了得到一些拓扑的感性认识，假设欧氏平面是一张高质量无边界的橡皮，该橡皮能够伸长和缩短到任何理想的程度。想象一下基于这张橡皮所绘制的图形，允许这张纸伸长但是不能撕破或者重叠，这样原来图形的一些属性将保留，而有些属性将会失去。例如，在橡皮表面有一个多边形，多边形内部有一个点。无论对橡皮进行压缩或拉伸，点依然存在于多边形内部，点和多边形之间的空间位置关系不改变，而多边形的面积则会发生变化。前者则是空间的拓扑属性，后者则不是拓扑属性。

拓扑分析就是研究空间的拓扑属性，即对图形在拓扑变化下不变的性质进行相关的分析。

# 2 开发前的准备工作

**开发工具：**

Microsoft Visual Studio 2005 / Eclipse +My Eclipse

**操作系统**

Windows XP / Windows 2000 / Windows 2003

**浏览器：**

IE 5.5 + 或更高

**环境支持：**

MAPGIS7-IMS 行业版

Microsoft SQL Server 2000

.Net FrameWork2.0

IIS5.0 或更高 / JDK1.5、TOMCAT 5.5 或更高

**测试数据：**

地图数据 world.HDF 等（示例中将自带）

# 3 搭建开发环境

搭建开发环境过程参见 MAPGIS7-IMS 行业版的安装与配置篇中的行业版服务器配置部分。

# 4 示例站点简介

MAPGIS7-IMS 行业版提供的示例站点 EntComm（.NET）和 JEntComm（JAVA），集成了栅格地图服务和矢量地图服务，是 MAPGIS7_IMS 的综合示例站点。

EntComm（或 JEntComm）示例站点主要包括以下两部分的功能：

1. MAPGIS7-IMS 行业版地图基本功能
   - 基本显示：支持放大、缩小、移动、复位、刷新等功能。
   - 放大镜：方便用户清晰查看地图。
   - 鹰眼：主图左下角的区域，提供地图的鸟瞰图。
   - 测量：测量地图指定区域的距离和面积。
   - 系统设置：提供对地图的显示设置和闪烁设置功能。
2. MAPGIS7-IMS 行业版地图服务功能
   - 地图搜索：通过输入关键字搜索相关信息，并提供信息定位显示功能。
   - 图层控制：实现图层的显示控制和图层状态控制功能。
   - 地图查询：分为基本查询和条件查询，根据用户选择的不同的查询方式，以对应的方式在地图绘制的查询范围内，交互查找相关信息，并对找到的信息进行定位闪烁。
   - 地图编辑：实现地图点、线、面的添加、修改和删除功能。
   - 空间分析：提供叠加分析、缓冲区分析、路径分析、拓扑分析和裁剪功能。
   - 统计分析：提供区域统计分析和属性分析功能。

- 坐标转换：根据输入转换的坐标值，选择目标坐标类型实现坐标的转换。
- 面积周长计算：在输入框中输入坐标值计算区域面积和周长。

EntComm（或 JEntComm）站点提供全面的功能，既可满足大众用户的需求，也可满足专业用户复杂的地图编辑与空间分析的需求，适用面更广，应用性更强。

# 5 示例开发与实现(.NET 版)

## 5.1 站点搭建过程

在 MAPGIS7-IMS 行业版的.NET 开发模式下提供了三种程序设计的方法：

第一种是自动搭建，即使用"MAPGISIMS 快速开发版配置向导"工具，自动搭建一个自定义基本功能的.NET 站点，具体操作流程参见 MAPGIS7-IMS 行业版程序设计初级篇；

第二种是控件式搭建，该方法是将动态组件做成控件的形式，在开发时把控件导入到 VS2005 中，直接拖放相关控件就能完成站点搭建，具体操作流程参见 MAPGIS7-IMS 行业版程序设计中级篇；

第三种是手动搭建，即根据示例站点，在 VS2005 中手动搭建站点。

由上知，前面两种方式的搭建方法简单、易操作，但有一定的局限性，而手动搭建方式可以任意自建框架，设置页面布局和样式，灵活性更大。

这里主要介绍手动搭建站点的方法，以下是运用 Visual Studio 2005 工具手动搭建示例站点的具体介绍。

EntComm 实例在 Visual Studio 2005 环境下的具体开发过程如下（具体代码见安装目录下站点 EntComm 的代码）：

1. 在 Microsoft Visual Studio 2005 开发环境中创建“EntComm”工程。具体步骤如下：

（1）新建网站：文件—>新建—>网站；

（2）选择网站的存储路径，并键入网站名- EntComm，选择 Visual C#语言；

（3）点击“确定”，网站创建成功。

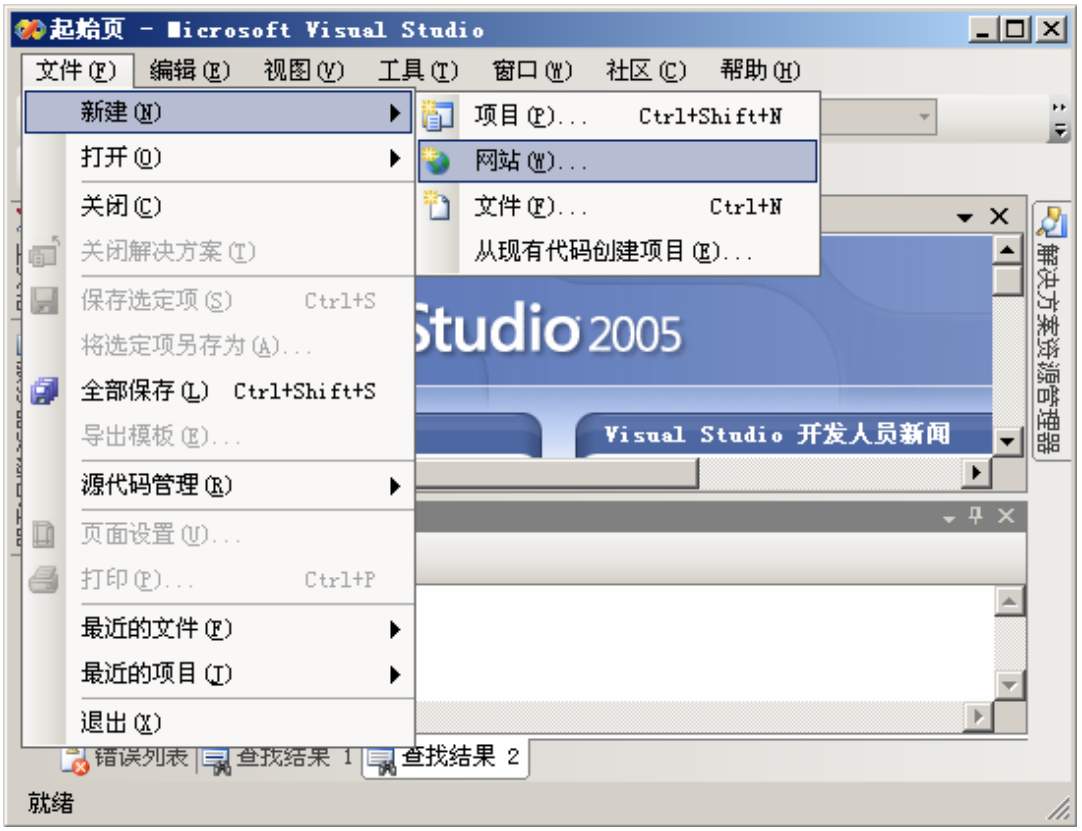



图 5-1-1 新建网站（一）

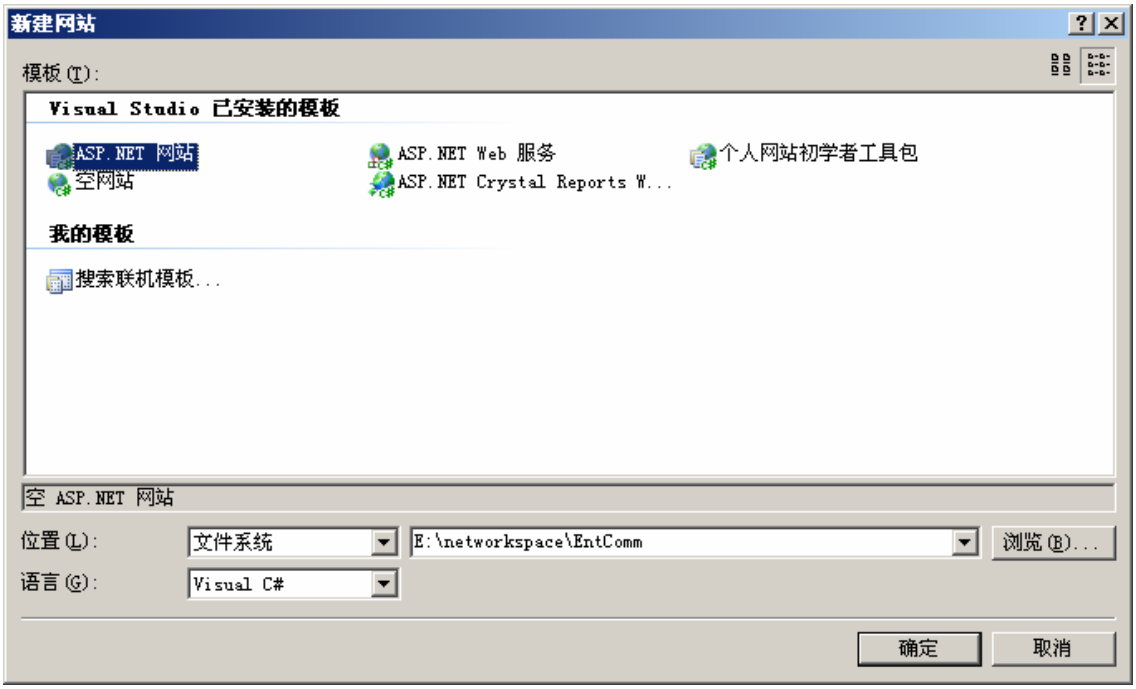



图 5-1-2 新建网站（二）

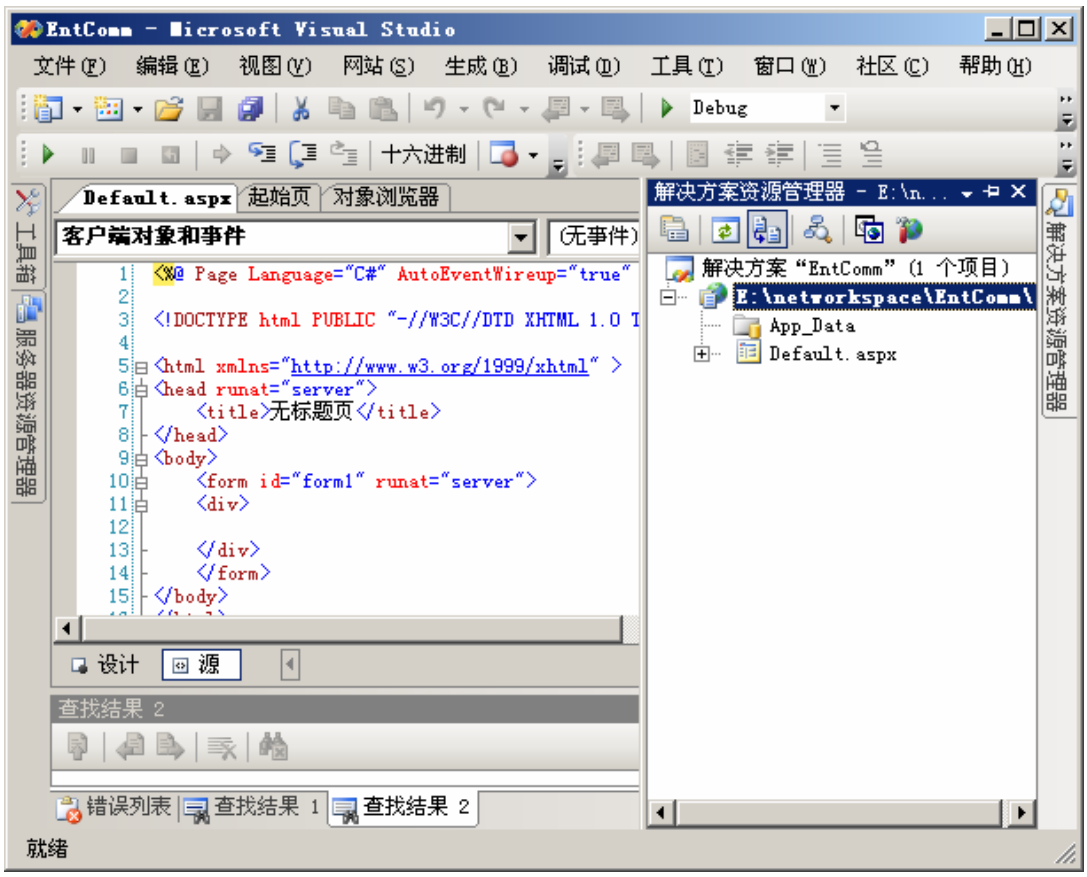



图 5-1-3 网站创建成功

2. 拷贝示例站点中的“icons”“images”、“css”文件夹到工程下面，然后刷新工程。“icons”与“images”用于存放站点使用的图标，“css”用于存放站点的样式表文件。具体如图：

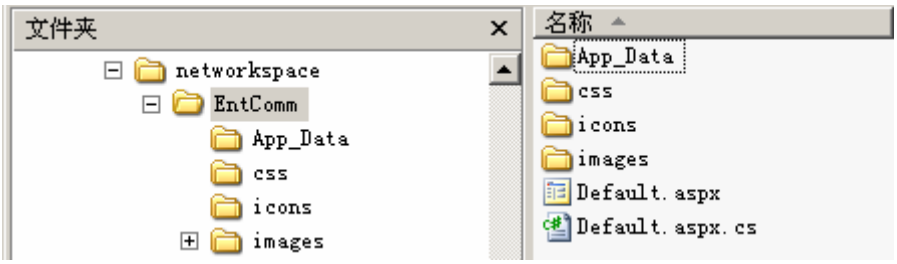



图 5-1-4 拷贝 css 等文件夹

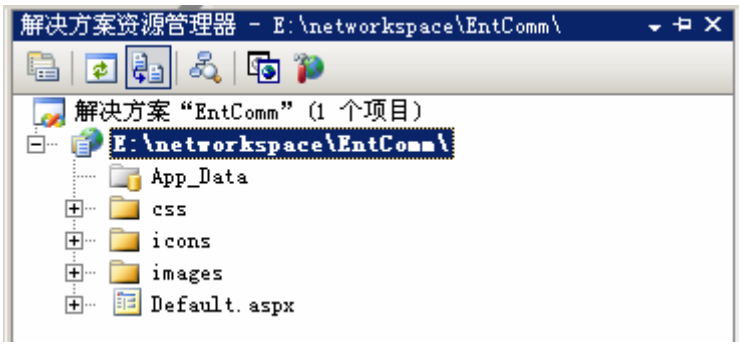



图 5-1-5 刷新工程

3. 在工程里面添加“jslib”文件夹，拷贝示例站点目录下的 jslib 文件夹中的脚本文件到新建的“jslib”文件夹中，然后刷新工程。具体如图：

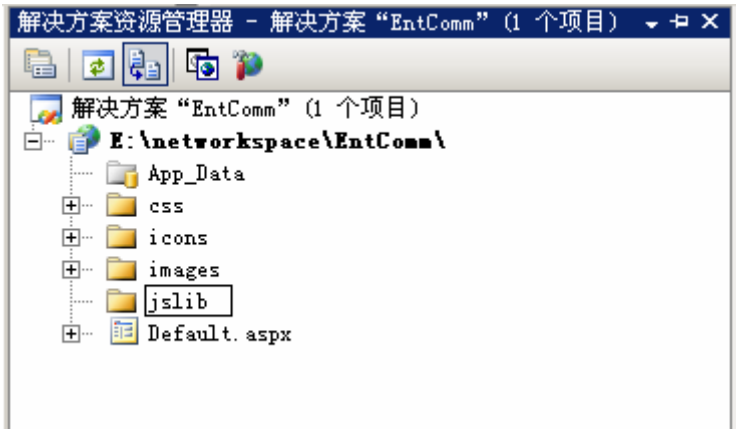



图 5-1-6 新建“jslib”文件夹

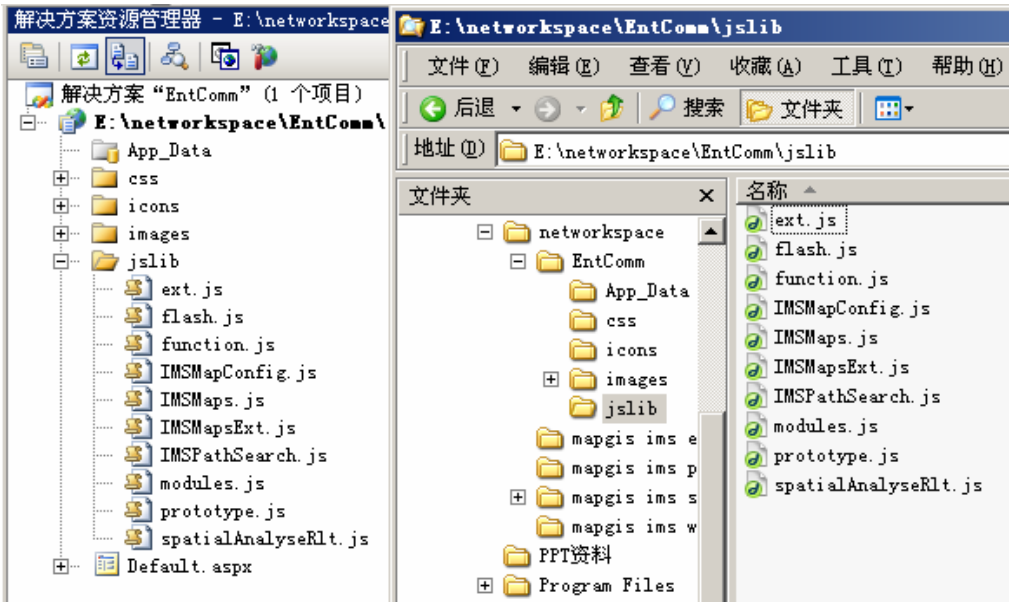



图 5-1-7 拷贝脚本文件并刷新

4. 右键工程—>添加 ASP．NET 文件夹—> “bin”文件夹，将示例站点 bin 目录下的组件拷贝至新建工程的 bin 目录下，然后刷新工程。具体如图：

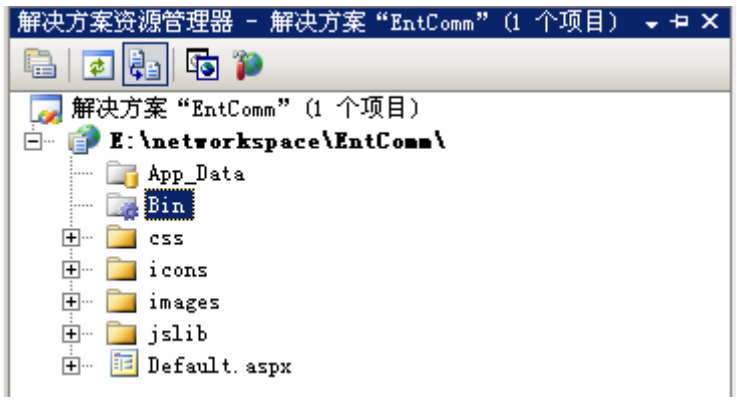



图 5-1-8 新建“Bin”文件夹

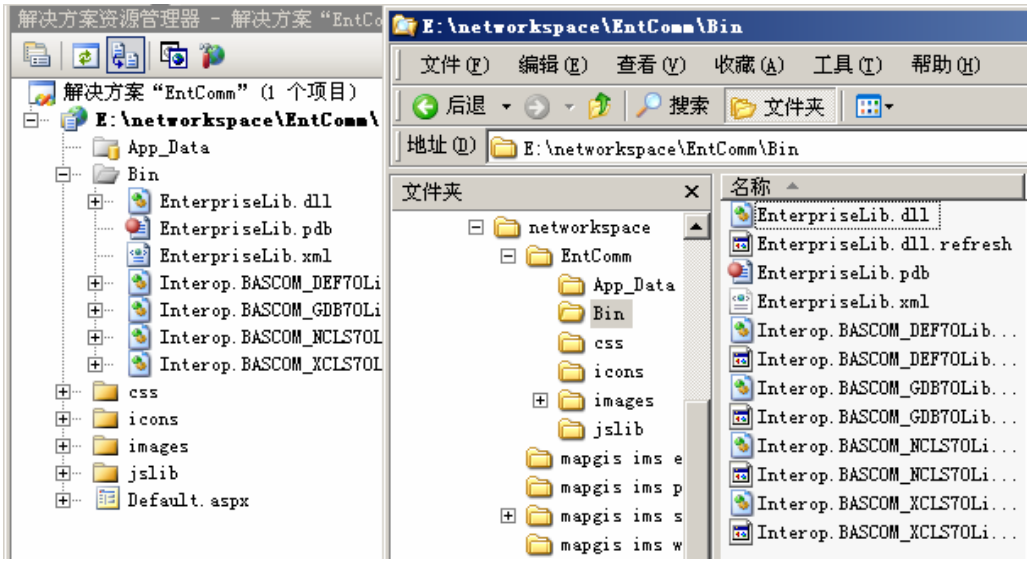



图 5-1-9 拷贝组件文件并刷新

5. 右键工程—>添加 ASP.NET 文件夹—> “App_Code”文件夹,将示例站点 App_Code 目录下的文件拷贝至新建工程的 bin 目录下，然后刷新工程。具体如图：

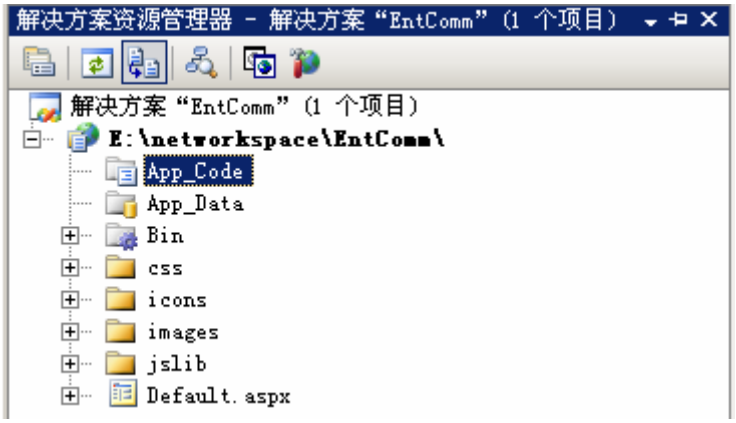



图 5-1-10 新建“App_Code”文件夹

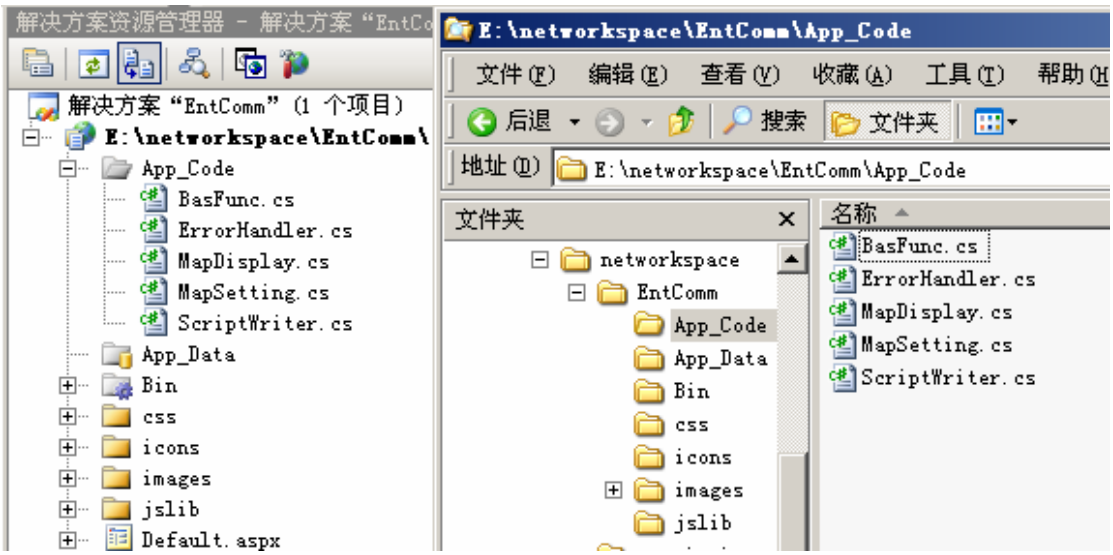



图 5-1-11 拷贝 cs 文件并刷新

6. 在该工程下面新建一个名为“mapimage”的文件夹，具体如图：

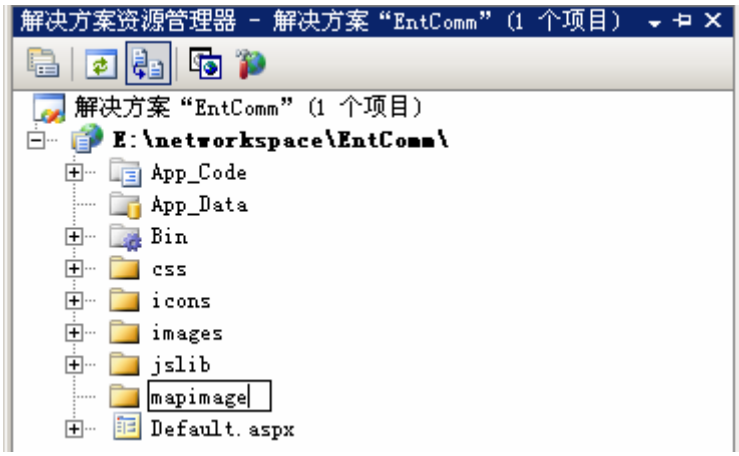



图 5-1-12 新建“mapimage”文件夹

7. 在解决方案资源管理器中，新建的“Web.config”文件并添加相关代码，步骤如下：

（1） 右键工程—〉“添加新项”，然后选择“web 配置文件”，在名称列填写“Web.config”，最后单击“添加”即可。若工程中已存在该文件，此步骤可省略。

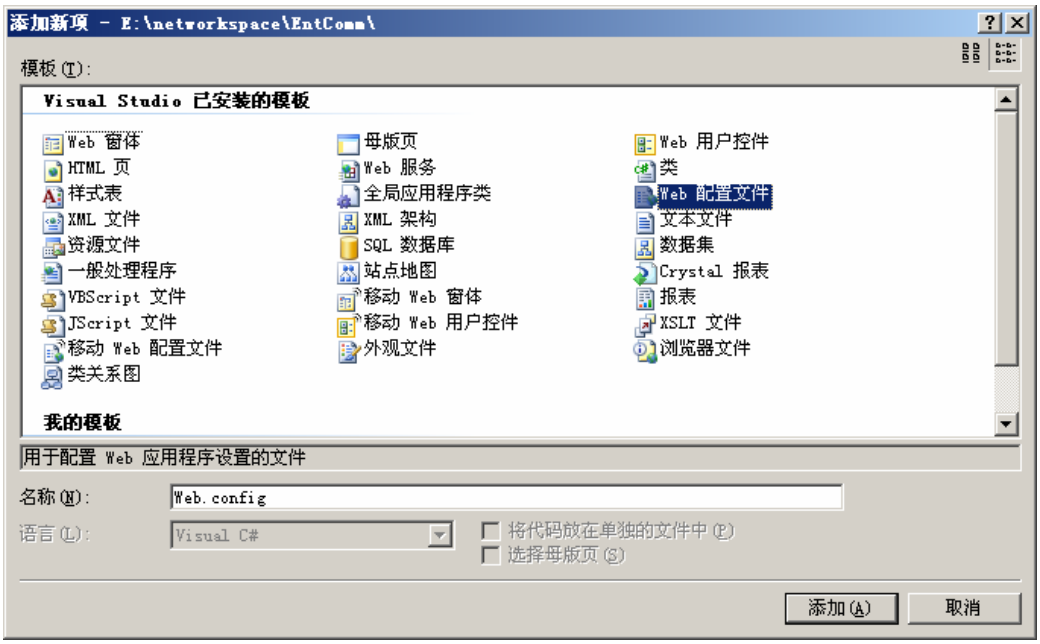



图 5-1-13 新建“Web.config”文件

（2） 在新建的“Web.config”文件里面添加如下代码（即可用下述代码覆盖新建的 web 配置文件中的〈/appSettings〉代码段）：

```
<appSettings>
        <add key="CityConStr" value="data source=127.0.0.1;initial
catalog=CityPosInfo;user id=sa;password=123"/>
        <!--地图类型:0,栅格;1,矢量;2,栅矢结合-->
        <add key="MapType" value="2"></add>
```

```
        <!--栅格流数据GIS服务器-->
        <add key="GisService" value="127.0.0.1"></add>
        <add key="GisPort" value="5142"></add>
        <add key="HDF" value="WhMap.HDF,car.HDF"></add>
        <add key="HDFBus" value="WhBus.HDF"></add>
        <!--网络分析矢量数据-->
        <add key="DBSVR" value="world"/>
        <add key="FSetName" value="ds"/>
        <add key="backFlag" value="0"/>
        <add key="netName" value="武汉道路网"/>
        <!--矢量数据-->
        <add key="needGisMgr" value="false"></add>
        <add key="ip" value="127.0.0.1"></add>
        <add key="port" value="5141"></add>
        <add key="mapName" value=" wh.Map "></add>
        <add key="writeLog" value="false"></add>
        <add key="logPath" value="c:\tmp"></add>
</appSettings>
```

参数解析：

- CityConStr: 数据库连接参数。
- MapType: 地图数据类型。

栅格流数据：

- GisService: 栅格服务gis服务器IP地址。
- GisPort: 栅格服务gis服务器端口。
- HDF: 栅格地图数据文件名。
- HDFBus: 公交换乘的地图数据文件名。

网络分析：

- DBSVR: 空间数据库名。
- FSetName: 要素数据集名。
- backFlag: 结果返回标志。
- netName: 网络类名。

矢量数据：

- needGisMgr: 负载均衡标记符。
- ip: 矢量服务gis服务器IP地址。
- port: 矢量服务gis服务器端口。
- mapName: 矢量地图文档名称。
- writeLog: 日志标记符。
- logPath: 日志写入路径。

8. 在工程中创建“Global.asax”文件，并添加相关代码，步骤如下：

（1） 在解决方案资源管理器中，右键工程—>“添加新项”—> “全局应用程序类”， 在名称处填写“Global.asax”，最后单击“添加” 即可。若工程中已存在该文件，此步骤可省略。

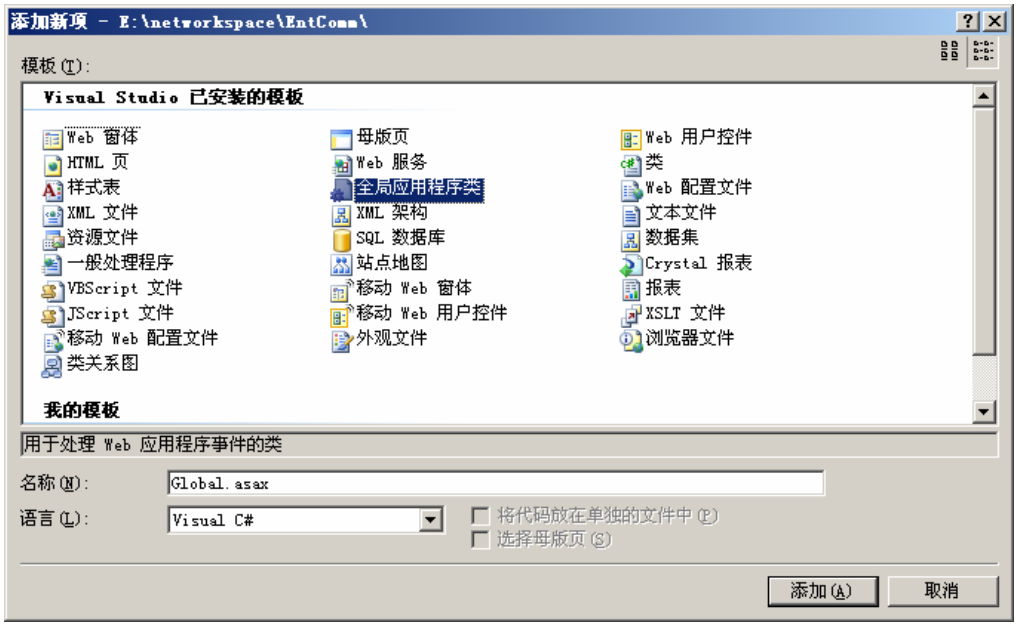



图 5-1-14 新建“Global.asax”文件

（2） 在新建的“ Global.asax”文件中添加如下代码，覆盖相应代码段：

```
void Application_Start(object sender, EventArgs e)
    {
        Application["bt"] =
System.IO.File.ReadAllBytes(Server.MapPath("~/images/mapgis.jpg"));
        Application["errorImg"] =
System.IO.File.ReadAllBytes(Server.MapPath("~/images/wait.gif"));

        string AbsFileName = Server.MapPath("mapimage");
        try
        {
            MAPGISIMS.EnterpriseLib.CMapView.InitWebServer(AbsFileName);
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }
void Session_Start(object sender, EventArgs e)
    {
        try
        {
            MAPGISIMS.EnterpriseLib.CMapViewPort viewport = new
MAPGISIMS.EnterpriseLib.CMapViewPort(System.Configuration.ConfigurationManager.AppSettin
gs["GisService"].ToString(),
Convert.ToInt32(System.Configuration.ConfigurationManager.AppSettings["GisPort"]), 512);
```

```
            Session["viewport1"] = viewport;
        }
        catch
        {
        }
        finally
        {
        }
    }
```

也可以把示例站点中的“Global.asax”文件直接拷贝到新建的工程下，不需要做上述操作。

9. 创建“GetIMG.ashx”一般处理程序页面（该页面的主要功能是取图），在页面中添加引用并更改相关代码。也可以把示例站点中的“GetIMG.ashx”文件直接拷贝到新建的工程下，不需要做如下 4 个步骤操作。

（1） 在解决方案资源管理器中，右键工程—〉“添加新项”—〉“一般处理程序”页面，在名称处填写“GetIMG.ashx”，最后单击“添加”即可。

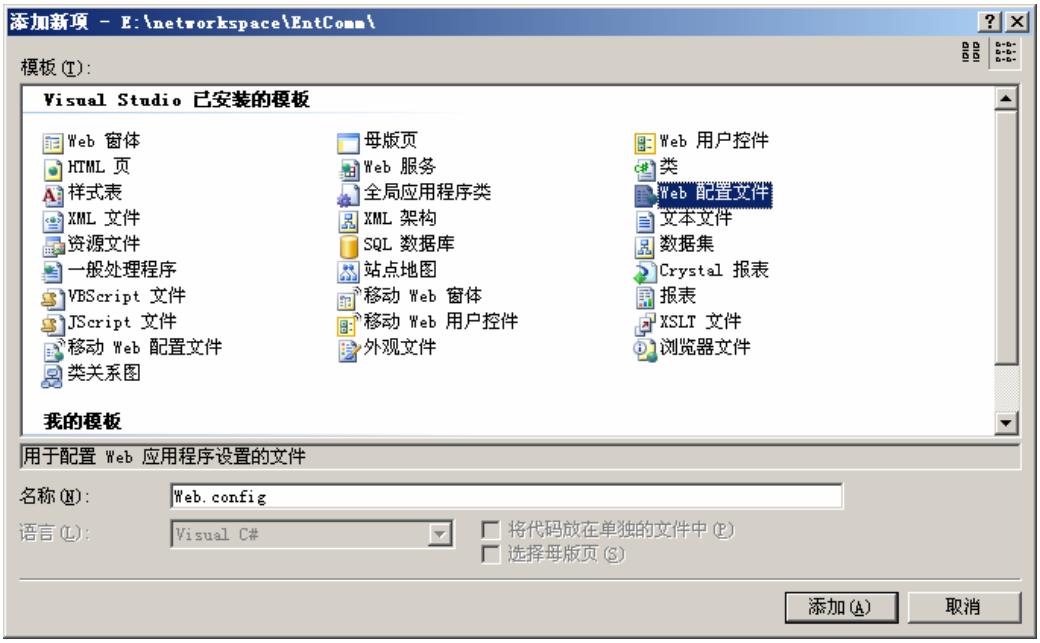



图 5-1-15 新建“GetIMG.ashx”页面

（2） 在页面中添加引用：

```
using System;
using System.Web;
using System.Web.SessionState;
using MAPGISIMS.EnterpriseLib;
using System.Configuration;
```

(3) 添加 IRequiresSessionState 标记接口的一个继承，如果不添加 IRequiresSessionState 标记接口将不能使用 session 对象。 具体方法如下：将 public class GetIMG: IHttpHandler 修改为 public class GetIMG : IHttpHandler, IRequiresSessionState。

（3） 最后添加如下代码，把新建文件的原来代码覆盖：

```
public void ProcessRequest(HttpContext context)
    {
        int rowNo = Convert.ToInt32(context.Request.QueryString["a"]);
        int lineNo = Convert.ToInt32(context.Request.QueryString["b"]);
        int level = Convert.ToInt32(context.Request.QueryString["c"]);
        String hdfName = "";
        int layerNo = Convert.ToInt32(context.Request.QueryString["layerNo"]);

        try
        {
            CMapViewPort viewport = new CMapViewPort();
            if (context.Session["viewport1"] == null)
            {
                viewport = new
CMapViewPort(ConfigurationManager.AppSettings["GisService"].ToString(),
Convert.ToInt32(ConfigurationManager.AppSettings["GisPort"]), 512);
                context.Session["viewport1"] = viewport;
            }
            else
                viewport = (CMapViewPort)context.Session["viewport1"];
            viewport.SetIndexOffset(0);
            if (layerNo == 0)
                hdfName = ConfigurationManager.AppSettings["HDF"].Split(',')[0];
            else
                hdfName = ConfigurationManager.AppSettings["HDF"].Split(',')[1];
                context.Response.BinaryWrite(viewport.GetMapImage(hdfName, 0, rowNo,
lineNo, level));
        }
        catch (Exception e)
        {
            context.Response.BinaryWrite((byte[])context.Application["bt"]);
        }

    }
```

10. 参照步骤 7 创建 GetIMG.ashx 的步骤，创建 GetBusIMG.ashx、GetMapBound.ashx 和 Maps.ashx 页面。
11. 拷贝示例站点根目录下的“VectorPage”文件夹、“default.aspx”、“default.aspx.cs”、“MapFrame.aspx”、“MapFrame.aspx.cs”和“menu.htm”等到新建的工程目录下，刷新工程。

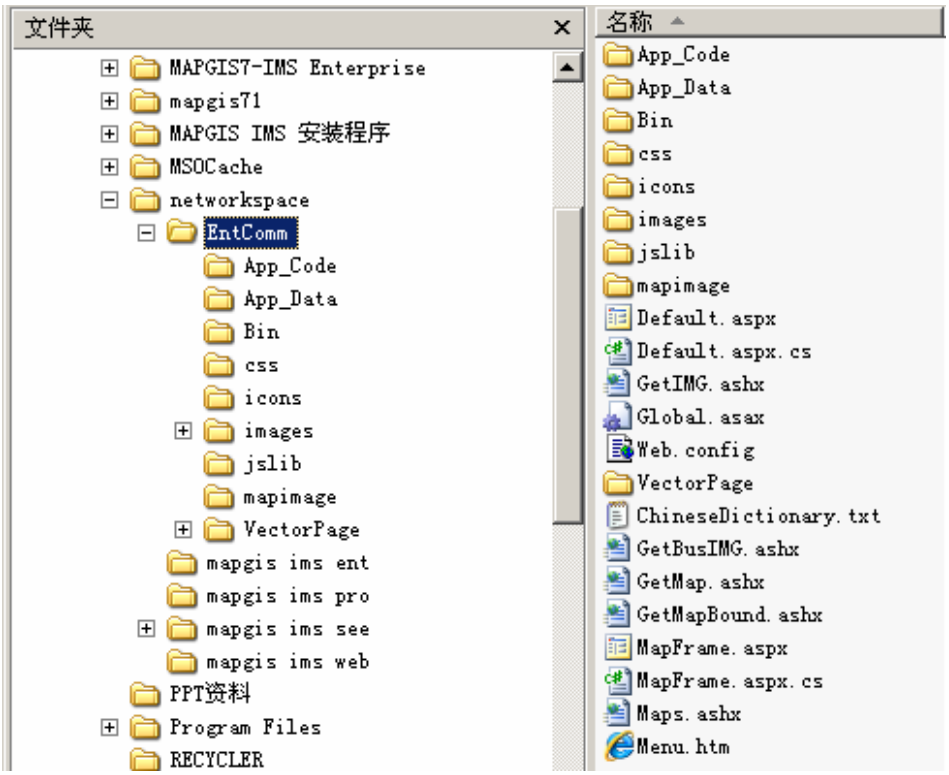



图 5-1-16 拷贝“VectorPage”文件夹和其他文件

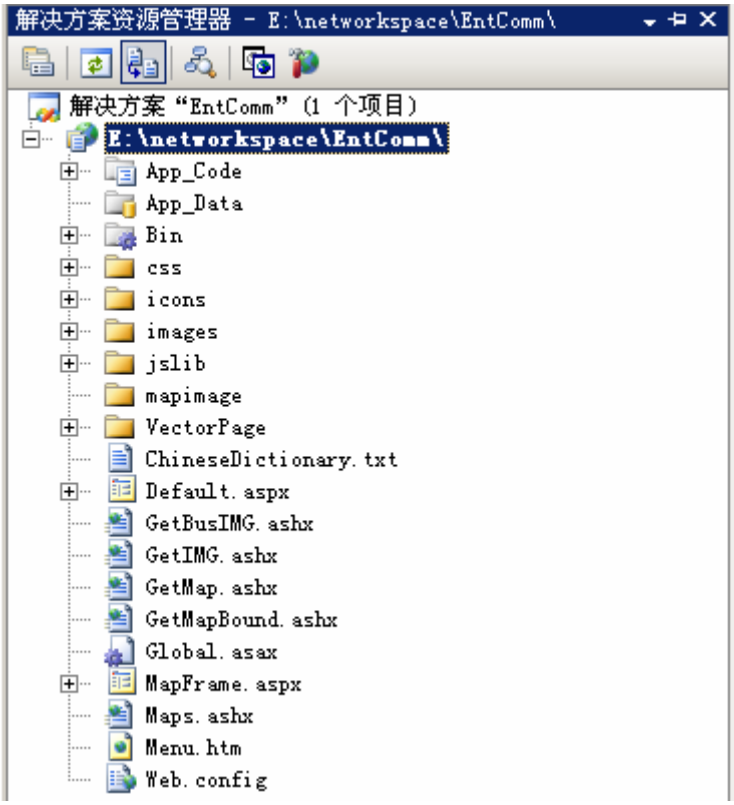



图 5-1-17 刷新工程

**注：**若想搭建新的框架创建一个不同风格的站点，前 10 个步骤不变，只须第 11 步时有所更改，即拷贝示例站点根目录下的其他文件，新建一个名为 MapFrame 的框架页面，并对新建框架添加相应功能代码。

# 5.2 站点具体功能实现

## 5.2.1 基本显示功能

### 5.2.1.1地图显示

地图显示功能是 MAPGIS7-IMS 行业版中最基本的功能，实现客户端向 GIS 服务器取图并把地图图像返回到客户端显示的过程。

地图显示效果如图：

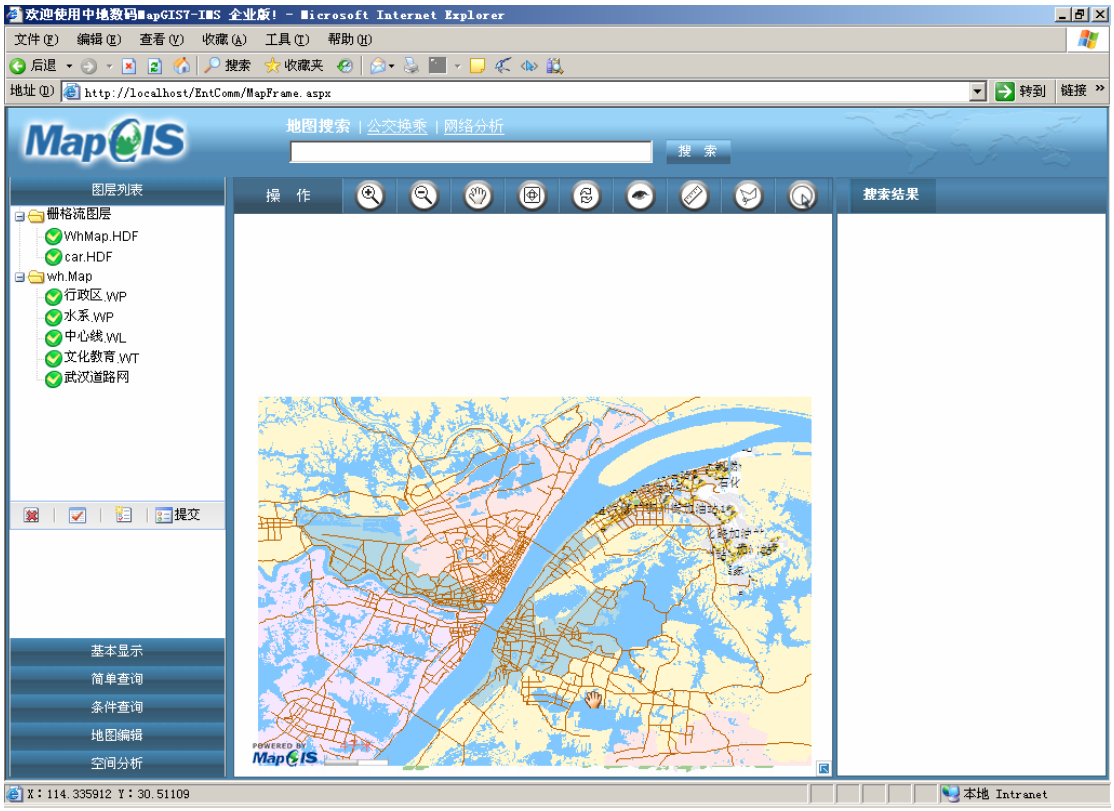



图 5-2-1-1 地图显示效果图

该功能的实现流程如下：

1. 在 default 表现页面创建表单 form1，客户端选择要显示的地图文档，通过处理页面调用 loadMap(mapname)脚本函数，并转到 MapFrame 框架页面。

➢ loadMap(mapname)脚本代码：

```
function loadMap(mapname)
{
   document.getElementById("mapName").value=mapname;
   document.getElementById("form1").submit();
}
```

该脚本函数位于 default.aspx 页面内。

2. 在 MapFrame.aspx 页面中添加 prototype.js、function.js、IMSMaps.js、IMSMapsExt.js、

IMSPathSearch.js 和 IMSMapConfig.js 的引用。

➢ **添加引用代码：**

```
<script type="text/javascript" src="jslib/prototype.js"></script>
<script type="text/javascript" src="jslib/function.js"></script>
<script type="text/javascript" src="jslib/IMSMaps.js"></script>
<script type="text/javascript" src="jslib/IMSMapsExt.js"></script>
<script type="text/javascript" src="jslib/IMSPathSearch.js"></script>
<script type="text/javascript" src="jslib/IMSMapConfig.js"></script>
```

3. 在 MapFrame.aspx 的 body 标记中调用函数 resize()和 pageload()。resize()获取地图容器的高宽，pageload()对 rsFrame 和 changeFrame 子框架进行初始化。这些脚本位于用户脚本库 function.js 中，具体代码参见示例站点的该脚本库。
4. 重要一步：在脚本函数库“function.js”中添加窗口事件监听，在该监听中调用初始化地图的脚本函数 initMAP()，在 initMAP()脚本函数中调用了 loadMap()脚本函数。

➢ **窗口事件监听代码：**

```
Event.observe(window,"load",function(){initMap();});
```

➢ **initMAP()部分代码：**

```
//初始化地图
function initMap(strlatlon,zoom){
    resize();
    loadMap(1,IMSMapConfig);
……
}
```

initMAP 脚本函数的省略部分用于添加右键菜单，用户可根据自己的需求进行适当修改。

➢ **添加 loadMap(mapType,mapConfig)，部分代码为：**

```
function loadMap(mapType,mapConfig)
{
    map = new IMSMaps("zdims",mapType,mapConfig);
    var sc = new IMSScale();
    sc.setUnits([103133.845,1000],["米","千米"]);
    map.sco = sc;
    map.addControl(sc,false);
    var l = new IMSLogo();
    map.addControl(l,false);
    ……
}
```

loadMap 脚本函数位于用户脚本库 function.js 中。它首先声明了 IMSMap 对象进行地图的初始化，然后可以在此添加 logo、鹰眼、放大镜、操作状态等各种控件。用户可根据自己的需要进行添加。其完整代码参见示例站点中 function.js 脚本库。

5. 在调用 loadMap(1,IMSMapConfig)函数中，声明了 IMSMaps 的对象，这是非常关键的一个地图类，位于脚本函数库 IMSMaps.js 中。IMSMaps 内含默认值。具体的配置在 IMSMapConfig 类中。

- IMSMapConfi.js 代码：

```
//地图配置类
function IMSMapConfig(){
    this.xMinMap = 114.125602229914;                    //整图的最小逻辑坐标
    this.yMinMap = 30.4539323507469;
    this.xMaxMap = 114.500788705197;                    //整图的最大逻辑坐标
    this.yMaxMap = 30.8291188260299;
    this.levelNum = 6;                                  //栅格地图级数
    this.statusBarFlag=true;                            //是否在状态栏显示当前鼠标的坐标
    this.gridImageUrl = ["GetIMG.ashx","GetIMG.ashx"];  //栅格图请求页面
    this.vectImageUrl = ["GetMap.ashx"];
    this.getVectCoorPage = "GetMapBound.ashx";
    this.zoomRate = 2;
}
//公交换乘配置文件
function IMSMapConfig1(){
    this.xMinMap = 20090.85;                            //整图的最小逻辑坐标
    this.yMinMap = 21884.4;
    this.xMaxMap = 61537.35;                            //整图的最大逻辑坐标
    this.yMaxMap = 63330.5;
    this.levelNum = 4;                                  //栅格地图级数
    this.statusBarFlag=true;                            //是否在状态栏显示当前鼠标的坐标
    this.gridImageUrl = ["GetBusIMG.ashx"];
        }
```

6. 在 IMSMaps.js 中通过 Init()实现栅格和矢量地图数据的判断功能。分为栅格和矢量地图数据两种方式来处理：

（1） 栅格数据：流程图如下：

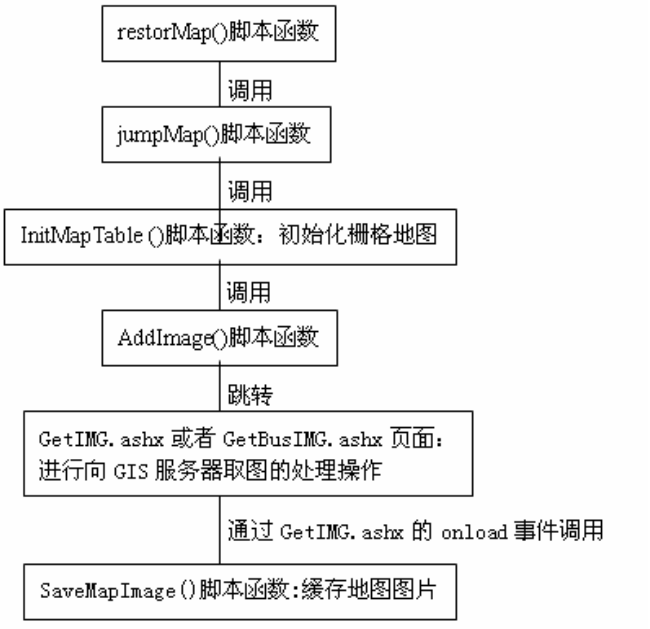



图 5-2-1-2 .NET 版的栅格数据流程图

➢ **服务器端 GetIMG.ashx 去栅格地图缓冲流数据页面代码：**

```
public void ProcessRequest(HttpContext context)
    {
        int rowNo = Convert.ToInt32(context.Request.QueryString["a"]);
        int lineNo = Convert.ToInt32(context.Request.QueryString["b"]);
        int level = Convert.ToInt32(context.Request.QueryString["c"]);
        String hdfName = "";
        int layerNo = Convert.ToInt32(context.Request.QueryString["layerNo"]);

        try
        {
            CMapViewPort viewport = new CMapViewPort();
            if (context.Session["viewport1"] == null)
            {
                viewport = new
CMapViewPort(ConfigurationManager.AppSettings["GisService"].ToString(),
Convert.ToInt32(ConfigurationManager.AppSettings["GisPort"]), 512);
                context.Session["viewport1"] = viewport;
            }
            else
                viewport = (CMapViewPort)context.Session["viewport1"];
            viewport.SetIndexOffset(0);
            if (layerNo == 0)
                hdfName = ConfigurationManager.AppSettings["HDF"].Split(',')[0];
            else
                hdfName = ConfigurationManager.AppSettings["HDF"].Split(',')[1];
                context.Response.BinaryWrite(viewport.GetMapImage(hdfName, 0, rowNo,
lineNo, level));
        }
        catch (Exception e)
        {
            context.Response.BinaryWrite((byte[])context.Application["bt"]);
        }
    }
```

该页面主要为获取客户端参数并调用 MAPGISIMS.EnterpriseLib 组件中的 **CMapViewPort** 类来从 GIS 服务器取图以及输出图片。

（2） 矢量数据：通过 GetVectBound 脚本函数转到 GetMapBound.ashx 页面处理获取矢量图坐标范围，然后再通过调用的 GetVectorMap()脚本函数中 r GetVectorMapByCood()脚本函数，转到 GetMap.ashx 页面,向 GIS 服务器进行取图处理操作。

➢ **GetMapBound. ashx 页面代码：**

```
public void ProcessRequest (HttpContext context) {
        try
        {
```

```
            int width = Convert.ToInt32(context.Request["w"]);
            int height = Convert.ToInt32(context.Request["h"]);
            bool isreload = Convert.ToBoolean(context.Request["reload"]);
            MapDisplay.CheckViewport();
            StringBuilder sb = new StringBuilder();
            CMapView viewport = (CMapView)HttpContext.Current.Session["viewport"];
            viewport.ImageWidth = width;
            viewport.ImageHeight = height;
            if (context.Session["restoreFlg"] == null || isreload)
            {
                viewport.Restore();
                context.Session["restoreFlg"] = true;
            }
            OpCRect rc = viewport.MapDisplayBound;
            sb.Append(rc.XMin.ToString() + ",");
            sb.Append(rc.YMin.ToString() + ",");
            sb.Append(rc.XMax.ToString() + ",");
            sb.Append(rc.YMax.ToString() + ",");
            rc = viewport.MapOriginalBound;
            sb.Append(rc.XMin.ToString() + ",");
            sb.Append(rc.YMin.ToString() + ",");
            sb.Append(rc.XMax.ToString() + ",");
            sb.Append(rc.YMax.ToString());
            context.Response.Write(sb.ToString());
        }
        catch
        {
            context.Response.Write("0,0,1,1,0,0,1,1");
        }
    }
```

该页面主要接收客户端参数并调用 MAPGISIMS.EnterpriseLib 组件中的 **CMapView 类**中的方法来得到当前地图显示范围和原始地图显示范围，并分别进行设置，最后向客户端输出图片逻辑范围。

- **GetMap.ashx 页面代码：**

```
public void ProcessRequest (HttpContext context) {
        try
        {
            byte[] bt;
            double xmin = Convert.ToDouble(context.Request.QueryString["LogicMinX"]);
            double ymin = Convert.ToDouble(context.Request.QueryString["LogicMinY"]);
            double xmax = Convert.ToDouble(context.Request.QueryString["LogicMaxX"]);
            double ymax = Convert.ToDouble(context.Request.QueryString["LogicMaxY"]);
            int width = Convert.ToInt32(context.Request.QueryString["ImageWidth"]);
            int height = Convert.ToInt32(context.Request.QueryString["ImageHeight"]);
```

```
            int layerNo = Convert.ToInt32(context.Request.QueryString["layerNo"]);
            bool isEagle = Convert.ToBoolean(context.Request.QueryString["IsEagle"]);
            MapDisplay.CheckViewport();
            CMapView viewport = (CMapView)HttpContext.Current.Session["viewport"];
            OpCRect rc = new OpCRect();
            rc.XMin = xmin;
            rc.YMin = ymin;
            rc.XMax = xmax;
            rc.YMax = ymax;
            viewport.ImageWidth = width;
            viewport.ImageHeight = height;
            viewport.ImageFormat = ImageFormatEnum.gif;
            string imgFileName =
viewport.CreateImageName(HttpContext.Current.Session.SessionID, viewport.ImageFormat);
            viewport.MapDisplayBound = rc;
            if (isEagle)
            {
                bt = viewport.GetBinaryImage(imgFileName);
                if (HttpContext.Current.Session["viewportRect"] != null)
                {
                    viewport.ImageWidth =
Convert.ToInt32(HttpContext.Current.Session["viewportRectWidth"]);
                    viewport.ImageHeight =
Convert.ToInt32(HttpContext.Current.Session["viewportRectHeight"]);
                    viewport.MapDisplayBound =
(OpCRect)HttpContext.Current.Session["viewportRect"];
                }
            }
            else
            {
                bt = viewport.GetBinaryImage(imgFileName);
                HttpContext.Current.Session["viewportRectWidth"] = width;
                HttpContext.Current.Session["viewportRectHeight"] = height;
                HttpContext.Current.Session["viewportRect"] = rc;
            }
            HttpContext.Current.Session["viewport"] = viewport;
            context.Response.BinaryWrite(bt);
        }
        catch
        {
            ErrorHandler.ResponseErrorMessageImage();
        }
    }
```

该页面主要为获取客户端参数并调用 MAPGISIMS.EnterpriseLib 组件中的

CMapView 类中的方法来从 GIS 服务器取图以及输出图片。

## 5.2.1.2基本操作

基本操作实现地图的放大、缩小、移动、更新、复位等操作。该操作实现流程如下：

1. 通过地图初始化 loadMap 脚本函数（位于 function.js 中）加载地图操作状态类。

➢ **加载地图操作状态类代码：**

```
//操作状态
var os = new IMSOperStatus();
map.addControl(os,false);
```

IMSOperStatus()位于 IMSMapsExt.js 脚本函数库中，是地图操作状态类。

2. 在框架页面的客户端工具栏（位于 MapFrame.htm、menu.htm 页面中）或初始化地图脚本函数右键菜单部分添加基本操作的功能按钮代码，通过触发 onclick 事件的 setCurOper 函数更改当前地图操作状态。

MapFrame.aspx 页面客户端工具栏的基本操作按钮效果如图：




图 5-2-1-3 基本操作按钮效果图

➢ **在 MapFrame.aspx 中 id 为 cmdGroup0 的 div 层中添加如下代码：**

```
<img class="btncursor" onclick="top.map.setCurOper(1001)" alt="放大"
src="images/zoomIn_1.gif" align="middle" border="0"><br>
    放大<br>
<img class="btncursor" onclick="top.map.setCurOper(1002)" alt="缩小"
src="images/zoomOut_1.gif" align="middle" border="0"><br>
    缩小<br>
<img class="btncursor" onclick="top.map.setCurOper(1003)" alt="移动"
src="images/pan.gif" align="middle" border="0"><br>
    移动<br>
<img class="btncursor" onclick="top.map.setCurOper(1004)" alt="复位"
src="images/restore_1.gif"  align="middle" border="0"><br>
    复位<br>
<img class="btncursor" onclick="top.onDisplayToolClick('reflash')" alt="刷新"
src="images/ref.gif" align="middle" border="0"><br>
    刷新<br>
```

setCurOper 函数就是工具栏按钮被点击时触发的函数，它是一个很关键的脚本函数，位于脚本库 IMSMaps.js 内。具有一个输入参数 operType，代表按钮触发事件的代码，如：

- 1001（ IMSOperType.ZoomIn）—放大；
- 1002（IMSOperType.ZoomOut）—缩小；
- 1003（IMSOperType.Drag）—移动；
- 1004（IMSOperType.Restore）—复位。

➢ **setCurOper 函数基本操作部分代码：**

```
//设定当前操作状态
```

```
this.setCurOper=function(operType)
{
    this.operType = parseInt(operType);
    this.clearOper();
    switch(this.operType)
    {
        case 1001:
            this.activeTool = "ZoomIn";
            this.drawAction = "Rect";
        break;
        case 1002:
            this.activeTool = "ZoomOut";
            this.drawAction = "Rect";
        break;
        case 1003:
            this.activeTool = "Pan";
            this.drawAction = "Pan";
        break;
        case 1004:
            this.restoreMap();
        break;
        case 1005://清除
            if(this.vmlObj!=null)
            {
                this.vmlObj.dispose();
                this.vmlObj = null;
            }
        break;
        case 1006://刷新
           this.refresh();
    }
};
```

3. 在地图上做点击或拉框操作时在客户端产生相应的参数，通过鼠标事件传到服务器端页面（GetIMG.ashx 或 GetMap.ashx）处理，处理完成后生成新的地图，并将新地图返回客户端显示。

例如放大操作：

通过鼠标弹起事件 OnMapMouseUp 调用 ChargeRect()函数，在该函数中根据传递参数调用 ScaleZoomOutOrIn()函数，然后再调用 jumpMap()进行处理，最后转到服务器端页面（GetIMG.ashx 或 GetMap.ashx）处理并返回新图。

- **ChargeRect 脚本函数代码：**

```
this.ChargeRect = function(lminx,lminy,lmaxx,lmaxy)
    {
        switch(this.activeTool)
```

```
        {
            case "ZoomIn":
                 this.ScaleZoomOutOrIn(true);
                break;
            case "ZoomOut":
                 this.ScaleZoomOutOrIn(false);
                break;
            case "Pan":
                 if(this.hasVectorMap)
                 {
                     if(this.mapType!=1)
                        this.GetVectorMap();
                     else
                     {
                        this.jumpMap(0,0,0);
                        break;
                     }
                 };
                this.AdjustControl();
                break;
         default :
              break;
        }
    };
```

➢ **ScaleZoomOutOrIn 脚本函数代码：**

```
//放大缩小地图
this.ScaleZoomOutOrIn=function(flag)
   {
       if(this.mapType==1)
       {
           this.jumpMap(0,0,0);
           return;
       }
       if(!flag&&this.inf[2]*1<=0)
        return;
       if(flag&&this.inf[2]*1>=this.levelNum-1)
        return;
       var x=(this.mdLogicX+this.muLogicX)/2;
       var y=(this.mdLogicY+this.muLogicY)/2;
       this.centerCoor = [x,y];
       if(flag)
       {
           this.inf[2]=parseInt(this.inf[2])+1;
```

```
        }
        else
        {
            this.inf[2]-=1;
        }
      this.jumpMap(x,y,(this.inf[2]+1));

    };
```

注：OnMapMouseUp()、ChargeRect()、ScaleZoomOutOrIn()、jumpMap()都位于IMSMaps.js脚本库中。jumpMap()等脚本函数参见示例站点脚本库源码。

4. 以放大功能为例，客户端具体操作如图：




图 5-2-1-4 步骤 1-点击放大按钮

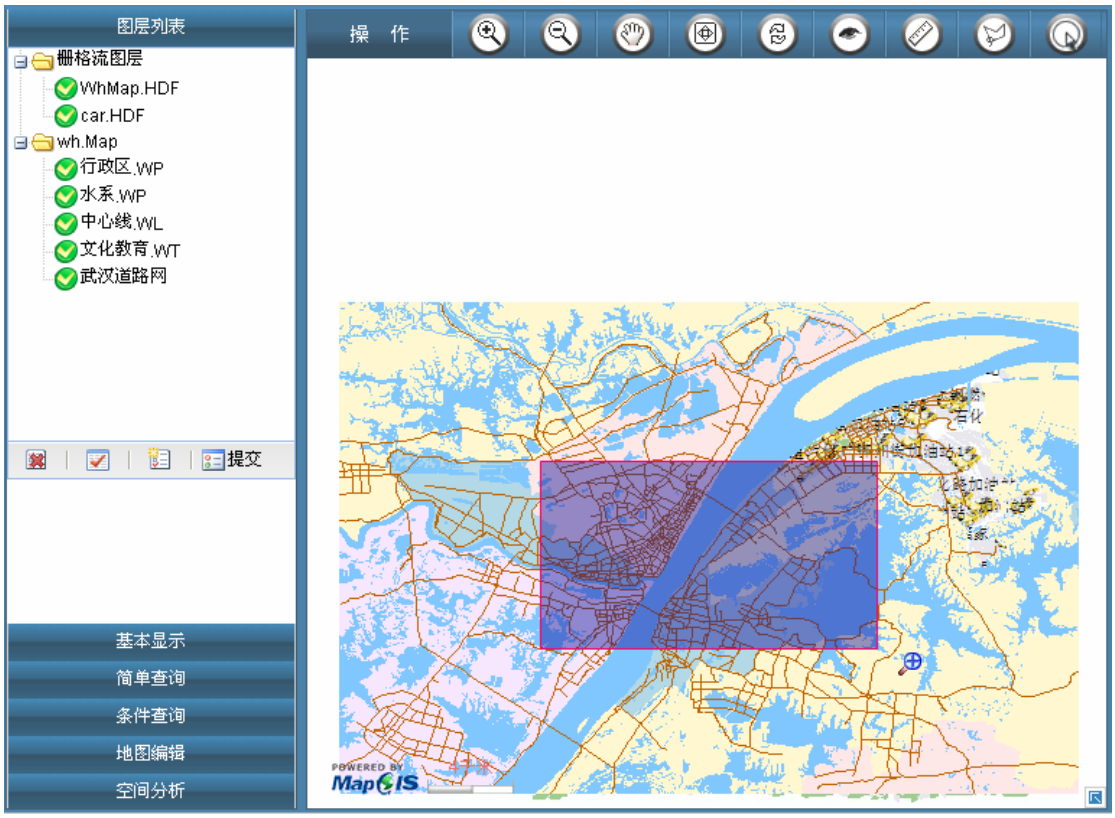



图 5-2-1-5 步骤 2-在地图上点击拉框

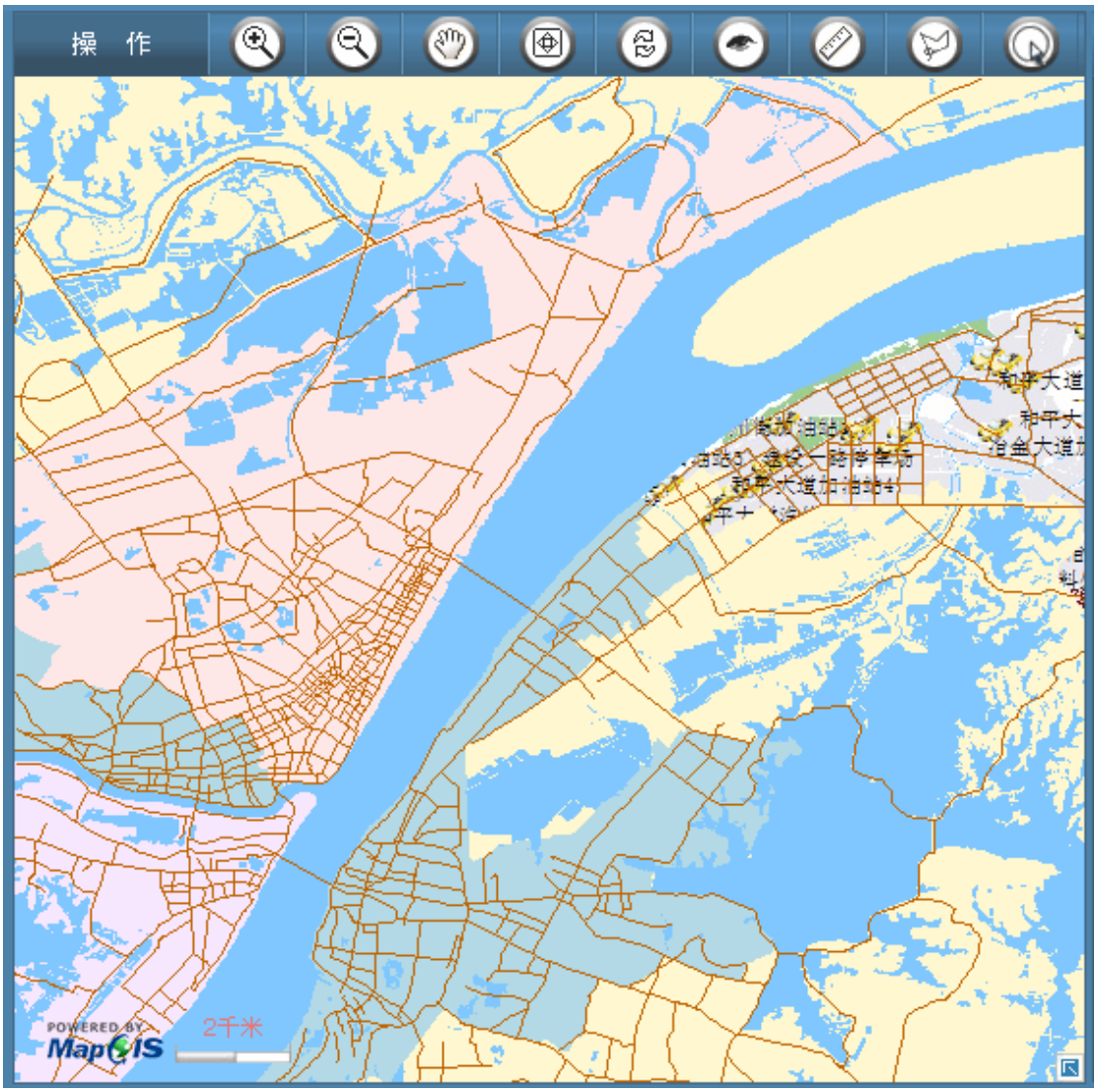



图 5-2-1-6 步骤 3-服务器端处理后放大结果显示

## 5.2.2 放大镜功能

放大镜功能是将鼠标指针所在点周围一定区域的图像按一定比例进行放大显示，以方便用户更加清楚的查看地图的内容，放大结果显示于鼠标右下方，放大镜显示的图像中心点是以鼠标为中心点。该功能是基于地图基本功能进行搭建的，实现流程如下：

1. 在地图图像显示功能的 Init 脚本函数（位于 IMSMaps.js 中）中通过调用 LoadGridMagnifier()和 LoadVectMagnifier()脚本函数加载放大镜。

➢ **LoadGridMagnifier()代码（用于在栅格地图中加载放大镜）：**

```
this.LoadGridMagnifier=function()//加载放大镜
{
    var zoomDiv = this.GetMapMagnifier();
       for(var i=0;i<this.gridImageUrl.length;i++)
       {
           zoomDiv.innerHTML+="<table id='zoomTab"+i+"' cellSpacing='0' cellPadding='0'
border='0' style='position:absolute;z-index:"+i+"'>"  +"<tr><td id='td"+i+"20'></td><td
```

```
id='td"+i+"21'></td><td id='td"+i+"22'></td></tr>"  +"<tr><td id='td"+i+"10'></td><td
id='td"+i+"11'></td><td id='td"+i+"12'></td></tr>"  +"<tr><td id='td"+i+"00'></td><td
id='td"+i+"01'></td><td id='td"+i+"02'></td></tr></table>";
        };
};
```

- **LoadVectMagnifier()代码（用于在矢量地图中加载放大镜）：**

```
this.LoadVectMagnifier=function()//加载放大镜
{
    var zoomDiv = this.GetMapMagnifier();
    for(var i=0;i<this.vectImageUrl.length;i++)
    {
        var vectDiv = this.vectLay[i].cloneNode(true);
        vectDiv.id="vectZoom"+i;
        vectDiv.style.position="absolute";
            zoomDiv.appendChild(vectDiv);
    };
};
```

2. 通过在框架页面的客户端工具栏（MapFrame.aspx 页面）添加放大镜的功能按钮代码，用 onclick 事件的 setCurOper 函数更改当前地图操作状态。

- **在 MapFrame.aspx 中 id 为 cmdGroup0 的 div 层中添加如下代码：**

```
<img src="images/buttonx/button11.gif" class="btnToolsmenu"
     onmouseover="this.src='images/buttonx/button11_02.gif'"
     onmouseout="this.src='images/buttonx/button11.gif'"
     title="放大镜" onclick="map.setCurOper(3003)"/>
```

setCurOper 函数就是工具栏按钮被点击时触发的函数，位于脚本库 IMSMaps.js 内。具有一个输入参数 operType，代表按钮触发事件的代码。放大镜功能的按钮触发事件的操作码为 3003（IMSOperType.OpenMagnifier）。

- **setCurOper 函数放大镜部分代码：**

```
case 3003:
            this.activeTool = "OpenMagnifier";
            this.drawAction = "OpenMagnifier";
            this.ShowMapMagnifier();
            this.addEventListener("mousemove",thisObj.MoveMapMagnifier,false);
 break;
```

在 setCurOper 函数中添加鼠标事件监听，通过该监听调用 ShowMapMagnifier()来显示放大镜，代码参见示例站点脚本库 IMSMaps.js 中的源码。

3. 在客户端改变地图操作状态为放大镜状态并在地图移动，通过鼠标监听事件把产生的参数传递到相应的函数处理即可实现放大镜功能。其客户端具体操作如图：




图 5-2-2-1 步骤 1-选择主图操作状态为放大镜




图 5-2-2-2 步骤 2-移动放大镜查看地图

## 5.2.3 鹰眼显示功能

地图鹰眼，俗称地图的鸟瞰图或缩略图。在电子地图中鹰眼的功能非常强大，通过鹰眼可以知道地图的当前位置；也可以在鹰眼上点击、拖动或移动到想要查看的位置。鹰眼的可视范围总是比主图的可视范围大，鹰眼中心框的可视范围就相当于主图的可视范围，主图的地理信息总是比鹰眼地图的地理信息详细。鹰眼地图的可视范围广阔，可以看到当前主图周边概况。该功能是基于地图基本显示功能进行搭建的，鹰眼的实现流程如下：

1. 在地图初始化 loadMap 脚本函数（位于 function.js 中）中加载鹰眼类。

➢ **加载鹰眼类代码：**

```
//鹰眼
var eo = new IMSEagleMap();
map.addControl(eo,false);
```

鹰眼类 IMSEagleMap 中的方法实现了鹰眼功能，该函数位于 IMSMaps.js 函数库中。

2. 通过在框架页面的客户端工具栏(MapFrame.aspx 页面)添加鹰眼的功能按钮代码，用其 onclick 事件的 EagleDisplay 脚本函数来控制鹰眼的显示状态。

➢ **在 MapFrame.aspx 中 id 为 cmdGroup0 的 div 层中添加如下代码：**

```
<img src="images/buttonx/button6.gif" class="btnToolsmenu"
     onmouseover="this.src='images/buttonx/button6_02.gif'"
     onmouseout="this.src='images/buttonx/button6.gif'"
```

```
title="鹰眼" onclick="EagleDisplay()"/>
```

- **EagleDisplay()代码：**

```
function EagleDisplay(){
    map.eagleEye.changeViewStatus();
}
```

通过 changeViewStatus 函数改变鹰眼的显示状态，该函数位于 function.js 中。

3. 在客户端单击鹰眼按钮触发 onclick 事件，显示鹰眼；并通过鼠标操作鹰眼中的地图，将产生的参数传递到相应函数实现鹰眼功能。其客户端具体操作如下：

（1）用左键点击鹰眼中心红色边框按住左键不放并拖动到你想要移动的位置松开左键即可实现鹰眼移动；鹰眼移动的同时将调用 IMSMaps 类中的 pntLocation 脚本函数以实现主图的同步移动。如下图：




图 5-2-3-1 步骤 1-拖动鹰眼中心框




图 5-2-3-2 步骤 2-鹰眼移动后的状态

（2）选择主图的移动操作状态，按住鼠标左键不放并拖动主图，松开左键的同时将调用鹰眼类 IMSEagleMap 中的 eagleJumpingMap 函数实现鹰眼的同步移动。如下图：




图 5-2-3-3 步骤 1-点击右键选择移动操作主图




图 5-2-3-4 步骤 2-主图移动鹰眼同步移动后的图示

注：IMSMaps 类中的 pntLocation 脚本函数与 IMSEagleMap 类中的 eagleJumpingMap 脚本函数都位于核心脚本库中，代码参见示例站点脚本库 IMSMaps.js 中的源码。

## 5.2.4 地图测量功能

地图测量功能实现包括距离测量、面积测量等操作。该功能是基于地图基本功能进行搭建的，该功能的实现流程如下：

1. 在地图初始化 loadMap 脚本函数（位于 function.js 中）中加载测量类。

➢ **加载测量类代码：**

```
var mo = new IMSMeasure();
mo.setDistanceUnits([103133.845,1000],["米","千米"]);
mo.setAreaUnits([10636589908.557,1000000],["平方米","平方千米"]);
```

其中 setDistanceUnits(valueArr,unitArr)和 setAreaUnits(valueArr,unitArr)各有两个参数：

- valueArr 为值数组，valueArr[0]记录当前坐标系下的比例度量，例如经纬度的地图 1 度代表的长度为 103133.845 米，那么 valueArr[0]值为 103133.845。如果地图为大地坐标系那么 valueArr[0]值为 1。valueArr[1]记录比例尺的比例，如果是米到千米的转换那么 valueArr[1]值为 1000。
- unitArr 为单位数组，是 valueArr 中值之间的转换度量的汉字表示，如“米”到“千米”或者“平方米”到“平方千米”。

2. 通过在框架页面（MapFrame.aspx、menu.htm 页面）的客户端触发工具栏，添加测量的功能按钮代码，用 onclick 事件的 setCurOper 函数更改当前地图操作状态。

➢ **在 MapFrame.aspx 中 id 为 cmdGroup0 的 div 层中添加如下代码：**

```
<img src="images/buttonx/button7.gif" class="btnToolsmenu"
     onmouseover="this.src='images/buttonx/button7_02.gif'"
```

```
        onmouseout="this.src='images/buttonx/button7.gif'"
        title="距离测量" onclick="map.setCurOper(3001)"/>
<img src="images/buttonx/button8.gif" class="btnToolsmenu"
        onmouseover="this.src='images/buttonx/button8_02.gif'"
        onmouseout="this.src='images/buttonx/button8.gif'"
        title="面积测量" onclick="map.setCurOper(3002)"/>
```

setCurOper 函数就是工具栏按钮被点击时触发的函数，位于脚本库 IMSMaps.js 内。带有一个输入参数 operType，代表按钮触发事件的代码：

- 3001（IMSOperType.OpenMagnifier）：代表距离测量。
- 3002（IMSOperType.OpenMagnifier）：代表面积测量。

➢ **setCurOper 函数测量部分代码：**

```
case 3001:
        if(this.mo==null){alert(this.gStr[1]);return;};
        this.mo.setMeasureMode("MeasureDistance","Polyline");
        break;
case 3002:
        if(this.mo==null){alert(this.gStr[1]);return;};
        this.mo.setMeasureMode("MeasureArea","Area");
        break;
```

在 setCurOper 函数中调用 setMeasureMode 函数（位于 IMSMaps.js 中），根据调用的参数不同，分别实现距离测量和面积测量功能。setMeasureMode()脚本函数的代码参见示例站点脚本库 IMSMaps.js 的源码。

3. 在客户端选择地图操作为距离测量或者面积测量，并在地图画线或画区指定操作的范围，通过鼠标监听事件把产生的参数传递到 setMeasureMode 函数处理，实现测量功能。以距离测量为例，客户端具体操作如图：




图 5-2-4-1 步骤 1-选择主图操作状态为测距




图 5-2-4-2 步骤 2-点击移动测距

## 5.2.5 系统设置功能

系统设置功能主要实现显示设置和闪烁设置。显示设置包括还原显示、显示标尺、显示坐标点、跟随显示、缓冲区显示、显示动态标注和图像质量的控制；闪烁设置主要实现定位闪烁的次数、间隔和绘图颜色三方面的控制。系统设置为用户提供灵活的、便捷的交互式控制面板，用户可以根据自己的需求和喜好进行定制。该功能的实现流程如下：

1. 在客户端添加系统设置的功能按钮代码，用 onclick 事件的 onSelectToolClick(位于 function.js 中）脚本函数进行处理。

munu.htm 页面客户端的系统设置功能按钮如图：




图 5-2-5-1 系统设置功能按钮

- **在 munu.htm 中添加如下代码：**

```
<img class="btncursor" onclick="top.onSelectToolClick('set',null)" alt="显示设置"
src="images/displayConfig.gif" align="middle" border="0"><br>
      显示设置<br>
<img class="btncursor" onclick="top.onSelectToolClick('flashParaSet',null)" alt="闪烁设置
" src="images/flashConfig.gif" align="middle" border="0"><br>
      闪烁设置<br>
```

- **onSelectToolClick()系统设置部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
        case "set":
            map.setCurOper(1003);
            var url = "VectorPage/ParamConfig.aspx?rnd="+Math.random();
            renderSelectResult(url);
        break;
        case "flashParaSet":
            map.setCurOper(1003);
            var url = "VectorPage/FlashParaSet.aspx?rnd="+Math.random();
            renderSelectResult(url);
        break;
```

```
    }
}
```

2. 在 onSelectToolClick()中根据不同的参数，调用 setCurOper 函数进行不同的操作状态设置，再转到不同页面进行处理。

（1）显示设置：通过 VectorPage 文件夹下的 ParamConfig.aspx 页面进行处理。即先在客户端进行显示设置，然后由后台代码进行处理。

➢ **ParamConfig.aspx.cs 处理页面代码：**

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            if (Session["viewport"] == null)
            {
                Response.Write("<script language=javascript>alert('地图文档装载失败或请求服务超时！');</script>");
                return;
            }
            viewport = (CMapView)Session["viewport"];
            viewport.GetImageDisplayParam(out originalFlag, out showScale, out showCoordinate, out followScale, out bufferShow, out dynamicNote, out imageQuality);
            this.origin.Checked=originalFlag;
            this.scale.Checked=showScale;
            this.Coordinate.Checked=showCoordinate;
            this.follow.Checked=followScale;
            this.buff.Checked=bufferShow;
            this.dynamic.Checked=dynamicNote;
            this.quality.SelectedIndex = imageQuality.GetHashCode();
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        originalFlag = this.origin.Checked;
        showScale = this.scale.Checked;
        showCoordinate = this.Coordinate.Checked;
        followScale = this.follow.Checked;
        bufferShow = this.buff.Checked;
        dynamicNote = this.dynamic.Checked;
        imageQuality=ImageQuality.middle;
        switch(this.quality.SelectedIndex)
        {
            case 0:
                imageQuality=ImageQuality.low;
                break;
```

```
            case 1:
                imageQuality=ImageQuality.middle;
                break;
            case 2:
                imageQuality=ImageQuality.high;
                break;
        }
        if (Session["viewport"] == null)
        {
            Response.Write("<script language=javascript>alert('地图文档装载失败或请求服务超时！');</script>");
            return;
        }
        viewport = (CMapView)Session["viewport"];
        viewport.SetImageDisplayParam(originalFlag, showScale, showCoordinate, followScale, bufferShow,dynamicNote, imageQuality);
        Response.Write("<script language=javascript>top.map.setCurOper(top.IMSOperType.Refresh);</script>");
        }
```

对显示设置的后台处理，先是通过 GIS 服务端 CMapView 类库的 GetImageDisplayParam 方法从 Session 中获取当前相关参数，然后再调用该类库的 SetImageDisplayParam 方法把从客户端设置的控制参数存到 GIS 服务器中，由此实现显示设置功能。

（2）闪烁设置：通过 VectorPage 文件夹下的 FlashParaSet.aspx 页面进行处理。即先在客户端进行闪烁设置，然后由脚本函数进行处理。

- **表单提交处理函数 SubmitColor()代码：**

```
function SubmitColor()
{
    var times = document.getElementById("flashTimes").value;
    var interval = document.getElementById("flashInterval").value;
    var color = document.getElementById("flashColor").value;

    if(times==""||interval==""||color=="")
    {
        alert("参数不可为空")
        return;
    }

    if(times!="")
    {
        var reg = /^[0-9]*$/
         if(!reg.test(times))
         {
```

```
                alert("闪烁次数需要输入整型数据！")
                return;
            }
        }
        if(interval!="")
        {
            var reg = /^\d+(\.\d+)?$/
             if(!reg.test(interval))
             {
                alert("闪烁间隔需要输入整型或浮点型数据！")
                return;
             }
        }
        top.flashTimes = times;
        top.flashInterval = interval;
        top.flashColor = color;
        alert("闪烁参数设置成功！");
    }
```

## 5.2.6 地图搜索功能

地图搜索主要为关键字搜索，能够将搜索到的分类信息以不同的图标进行展示。这些信息一般存于 SQL 数据库。该功能是基于地图基本功能进行搭建的，实现流程如下：

1. 在 MapFrame.aspx 页面中 id 为 hmenuTitle 的 div 层添加地图搜索的链接。

➢ **地图搜索链接的代码：**

```
<a href="javascript:void(0);" class='a1' onclick="funcChange(1,this);return false;地图搜索</a>
```

➢ **funcChange()的代码：**

```
//改变标题栏中的功能模块
function funcChange(type,obj){
    var aarr=obj.parentNode.getElementsByTagName("a");
    for(var i=0;i<aarr.length;i++){
        aarr[i].className='';
    }
    obj.className='a1';
    obj.blur();
    type=parseInt(type);
 for(var i=0;i<hdivs.length;i++){
     Element.hide(hdivs[i]);
}
    funChageClean();
    $("zdims").innerHTML="";
```

```
switch (type){
     case 1:{
          Element.show("lsForm");
          loadMap(1,IMSMapConfig);
          break;}
     case 2:{
          Element.show("bsForm");
          loadMap(0,IMSMapConfig1);
            break;}
     case 3:{
          Element.show("nsForm");
          loadMap(0,IMSMapConfig);
          break;}
     default:{
          Element.show("lsForm");}
}
   }
```

这是切换标题栏功能的脚本函数，位于脚本库 function.js 中。

在更改标题栏模块的地图搜索功能中调用了 loadMap(1,IMSMapConfig1)，最后用 GetMap.ashx 页面向 GIS 服务器取图并进行处理。loadMap 函数位于脚本函数库 function.js 中，其参数 mapType 为 1 表示请求取矢量图，mapConfig 为 IMSMapConfig 表示是基本地图配置类文件（详细配置位于脚本函数库 IMSMapConfig.js 中）。其后的取图显示流程见地图显示流程，loadMap 脚本函数代码见地图显示功能实现流程 4。

注：开始选择地图文档进入到主框架页面时，显示矢量地图，即可进行地图关键字搜索；如果当前不是初始默认的地图搜索状态，可以点击标题栏的“地图搜索”链接进入。

2. 在 MapFrame.aspx 页面中 id 为 lsForm 的 div 层添加关键字搜索的文本框与按钮。

➢ **关键字搜索的文本框与按钮的代码为：**

```
<div id="lsForm" class="hmenudiv">
     <table>
          <tr>
               <td><input class="inp1" name="keyword" id="keyword" /></td>
               <td>
                    <select id="searchtype" name="searchtype" style="display: none;">
                        <option value="-1">所有类型
                    </select>
               <td>
               <td><input name="lsFormSub" type="button" value="搜  索" class="btnfmt
               onclick="SearchByWords(document.getElementById('keyword').value)" /></td>
           </tr>
     </table>
</div>
```

SearchByWords 脚本函数就是地图关键字搜索时客户端入口函数，位于脚本库 IMSPathSearch.js 中。

3. 在地图关键字搜索按钮的 onclick 事件中调用 SearchByWords 脚本函数，然后转到 Maps.ashx 页面处理。

➢ **SearchByWords()代码：**

```
function SearchByWords(){
 var key = $("keyword").value;
    if(key.trim() == "")    {
        alert("请输入关键字！");
        return;
    }
    var dataurl =  "Maps.ashx?";
    var data = "_method=QueryByWords"+"&key="+escape(key.trim());
    AJAXRequest(dataurl,true,"POST",data,cbQuery);
     loadBar(1);
}
```

4. 在服务器端处理页面（Maps.ashx）中添加相关代码，实现查询数据库并将信息返回到客户端功能。

➢ **在 ProcessRequest 函数中添加如下代码：**

```
if (HttpContext.Current == null)
        {
            throw new NullReferenceException("The context is not set correct in
ProcessRequest().");
        }
        string strMethod = context.Request["_method"];
        string key = "";
        switch (strMethod)
        {
             case "QueryByWords"://模糊查询
                key = Regex.Unescape(context.Request["key"]);
                QueryByWords(key);
                break;
        }
```

➢ **添加 QueryByWords 函数（主要用来向数据库查询信息），代码为：**

```
public void QueryByWords(string key)
    {
        string path = HttpContext.Current.Server.MapPath(".") + @"\ChineseDictionary.txt";
        ChineseParse cp = new ChineseParse(path);
        ChineseWordUnit[] arryWords = cp.ParseChinese(key);
        GISDataQuery dq = new GISDataQuery(DatabaseType.SQLServer);//定义为SQL数据库
        dq.Open(System.Configuration.ConfigurationManager.AppSettings["CityConStr"]);
        string strCmd = "SELECT PlaceName as N,X,Y,Layer as L from c027 WHERE ";
        for (int i = 0; i < arryWords.Length; i++)
        {
```

```
            if (arryWords[i].word != null && arryWords[i].word.Trim() != "")
                strCmd += " PlaceName like '%" + arryWords[i].word + "%' AND";
        }
        string strQuery = strCmd.Substring(0, strCmd.Length - 3);
        string strResultXml = dq.GetDataByCondition(strQuery).GetXml();
        XmlDocument doc = new XmlDocument();
        XmlDeclaration declare = doc.CreateXmlDeclaration("1.0", "GB2312", "yes");
        doc.AppendChild(declare);
        doc.LoadXml(strResultXml);
        dq.Close();
        HttpContext.Current.Response.ContentType = "text/xml";
        HttpContext.Current.Response.Write(doc.InnerXml);
    }
```

在此查询中 GISDataQuery 类属于组件 MAPGISIMS.EnterpriseLib，它主要用来进行数据查询。

5. 在 MapFrame.aspx 页面的输入框中输入关键字，点击搜索按钮——>调用 SearchByWords 函数——>调用 AJAXRequest 函数，将关键字等相关参数以 ajax 异步请求的方式提交到服务器端页面（Maps.ashx），服务器端页面处理完成后，将结果返回由调用 AJAXRequest 时设定好的回调函数进行处理显示。

其客户端具体操作如图：




图 5-2-6-1 步骤 1-输入查询关键字




图 5-2-6-2 步骤 2-查询结果显示

## 5.2.7 图层控制功能

图层控制主要实现图层显示和图层状态控制功能，配合各种操作使用，以满足用户的需求。

图层列表及状态显示效果如下图：

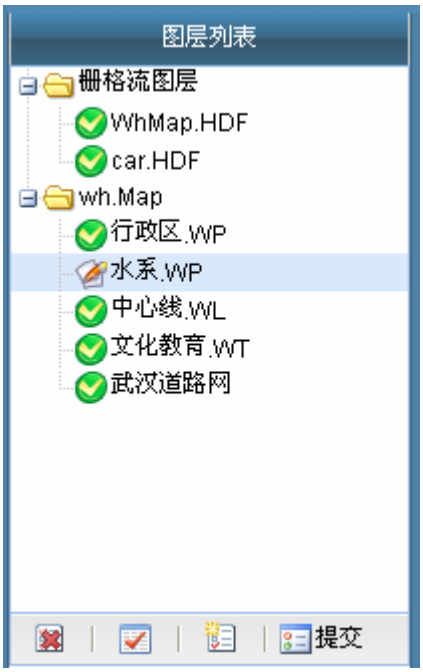



图 5-2-7-1 图层列表及状态显示效果

矢量图层显示功能实现流程：

1. 在 menu.htm 中添加显示图层列表的客户端代码。

➢ **图层列表的客户端代码：**

```
<tr  id="tr1" onclick="onMenuClick(this)">
    <td class="case_lr" width="100%" colspan=3>图层列表</td>
</tr>
<tr>
    <td width="4" bgcolor="#eaeaea"></td>
    <td  valign="top">
        <div id="layerTree" oncontextmenu="return false" style="height:400px"></div>
    </td>
    <td width="4" bgcolor="#eaeaea" height="30"></td>
 </tr>
```

2. 添加对脚本库 modules.js、ext.js 和样式表文件 tree.css 的引用。

➢ **引用的代码：**

```
<link type="text/css" href="css/tree.css" rel="stylesheet" />
<script type="text/javascript" src="jslib/ext.js"></script>
<script type="text/javascript" src="jslib/modules.js"></script>
```

3. modules.js 是图层列表树的脚本库，通过客户端 div 层为 layerTree 的 id，把列表树和相关按钮加载到客户端中，并转到 Maps.ashx 服务器页面处理。

➢ **modules.js 部分代码：**

```
//全局变量
var urls = {
    'showLayerTree' : 'Maps.ashx?_method=GetMapLayerList',
    'setLayerStatus': 'Maps.ashx'
}
var mapUrlCfg = new top.IMSMapConfig();
var tree;
/*图层树*/
function MLayerTree(){
    var tree;
    var ctx;
    var selectedNode;
    refreshTree = function(){
        tree.body.mask('加载图层中..', 'x-mask-loading');
        tree.root.reload();
        tree.root.collapse(true, true);
        setTimeout(function(){
            tree.body.unmask();
            tree.root.expand(true, true);
        }, 1000);
    }
……
tree = new Ext.tree.TreePanel({
        root: new Ext.tree.AsyncTreeNode({
            expanded: true,
            text: 'tree',
            allowDrag:false,
            allowDrop:false
        }),
        renderTo:'layerTree',
        id: 'layerTree2',
        height : 300,
        iconCls: 'nav',
        animate: true,
        //title: '图层列表',
        loader: new Ext.tree.TreeLoader({
            requestMethod :"GET",
            dataUrl: urls['showLayerTree'],
            listeners:{
                "load":function(o, node, response){
                if(response.responseText.indexOf("showCtx")==-1)
```

```
                tree.getBottomToolbar().hide();
            }
        }
    }),
……
```

➢ **Maps.ashx 中图层列表部分代码：**

```
string strMethod = context.Request["_method"];
        string key = "";
        switch (strMethod)
        {
          case "GetMapLayerList"://获取图层列表
                MapSetting ms = new MapSetting();
                context.Response.Write(ms.GetLayerListByJSON());
                break;
         }
```

4. 在 Maps.ashx 中创建 MapSetting 类对象，调用 GetLayerListByJSON 方法进行处理，以 JSON 的方式返回图层列表属性信息。MapSetting 类位于 App_Code 文件夹下的 MapSetting.cs 中，是图层显示与控制的关键类。

➢ **GetLayerListByJSON()代码：**

```
public string GetLayerListByJSON()
        {
            StringBuilder sb = new StringBuilder();
            string mapType = ConfigurationManager.AppSettings["MapType"];
            sb.Append("[");
            if (mapType == "0")
                sb.Append(this.GetGridLayer());
            else
            {
                if (mapType == "2")
                    sb.Append(this.GetGridLayer() + ",");
                sb.Append(@"{text:'" + MapSetting.GetMapNameConfig() + "',children:
["");
                ScriptWriter sw = new ScriptWriter();
                sw.SetClassAttribute("id", "text", "leaf", "cls", "qtip", "listeners");

                OpCLayerInformation[] layers = GetLayerList();
                for (int i = 0; i < layers.Length; i++)
                {
                    string[] param = new string[6];
                    param[0] = "'layer" + layers[i].LayerIndex + "'";
                    param[1] = "'" + layers[i].LayerName + "'";
                    param[2] = "true";
```

```
                    param[3] = "'" + GetLayerStatus(layers[i]) + "'";
                    param[4] = "'<span>右键单击选择图层状态<br>左键单击进入编辑状态<br>
左键拖动调整图层顺序</span>'";
                    param[5] = "{click:setEditable,contextmenu:showCtx}";
                    sw.AddClassData(param);
                }
                sb.Append(sw.GetJSONObjStr() + "]}");
            }
            sb.Append("]");
            return sb.ToString();
    }
```

图层状态控制功能是在图层显示的基础上实现的，其流程为：

1. 在图层列表中通过右键单击图层或左键单击图层列表下方的按钮改变图层的状态，功能函数在脚本库 modules.js 中。

改变图层状态控制按钮呈现效果如下图：

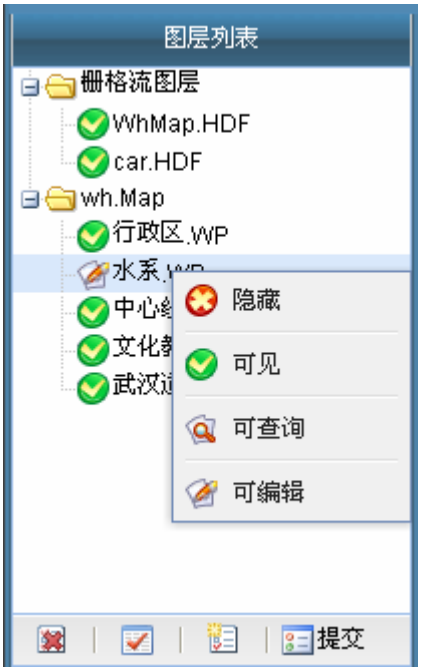



图 5-2-7-2 右键与列表下方的状态控制按钮

➢ **图层列表下方按钮的代码：**

```
bbar: [{
            icon: 'icons/hideAll.gif',
            minWidth: 35,
            cls: 'x-btn-icon',
            tooltip: '<span>隐藏全部图层</span>',
            handler: hideAll
        }, '-', {
            icon: 'icons/showAll.gif',
            minWidth: 35,
```

```
            cls: 'x-btn-icon',
            tooltip: '<span>显示全部图层</span>',
            handler: showAll
        }, '-', {
            icon: 'icons/activeAll.gif',
            minWidth: 35,
            cls: 'x-btn-icon',
            tooltip: '<span>查询全部图层</span>',
            handler: activeAll
        }, '-',{
            text: '提交',
            icon: 'icons/list-items.gif',
            cls: 'x-btn-text-icon',
            tooltip: '<span>更新图层状态</span>',
            handler: submitLayerStatus
        }]
```

2. 点击提交按钮，触发 submitLayerStatus()函数提交更改的图层状态信息，通过“setLayerStatus”的 url 标记传入参数 params(_method=UpdateMap)转到 Maps.ashx 服务器页面处理。提交成功后，通过 setCurOper 函数（位于脚本库 IMSMaps.js 中）设置地图操作状态为刷新状态，在地图容器中刷新地图。

➢ **submitLayerStatus()代码：**

```
submitLayerStatus = function(){
        var root = tree.getRootNode();
        var folder = root.childNodes;
        var nodes = folder[folder.length-1].childNodes;
        var params = '_method=UpdateMap';
        for (var i = 0; i < nodes.length; i++) {
            var leafEl = nodes[i].getUI().getEl();
            var leaves = leafEl.getElementsByTagName("div");
            var leaf = Ext.get(leaves[0]);
            params += "&" + nodes[i].id + "=" + getLayerIcon(leaf);
        }
        tree.body.mask('图层状态提交中...', 'x-mask-loading');
        Ext.Ajax.request({
            url: urls['setLayerStatus'],
            success: function(response){
                top.map.setCurOper(top.IMSOperType.Refresh);
                tree.body.unmask();
            },
            failure: function(response){
                alert("提交图层出错，请确定GIS服务器是否存在！");
                tree.body.unmask();
            },
```

```
            params: params
        });
    }
```

➢ **Maps.ashx 中更新地图文档部分代码：**

```
string strMethod = context.Request["_method"];
string key = "";
switch (strMethod)
        {
          case "UpdateMap"://更新地图文档
                ms = new MapSetting();
                ms.UpdateMap(context.Request.Form.ToString());
                break;
          }
```

3. 在 Maps.ashx 中创建 MapSetting 类对象，调用 UpdateMap ()方法进行处理，更新地图文档信息。MapSetting 类位于 App_Code 文件夹下的 MapSetting.cs 中，是图层显示与控制的关键类。

➢ **UpdateMap ()代码：**

```
    /// <summary>
    /// 更新地图文档
    /// </summary>
    /// <param name="idxList">新的图层序号</param>
        public void UpdateMap(string requestString)
        {
            string[] layerArr = requestString.Split('&');
            int layerNum = layerArr.Length-1;
            OpCUpdateLayerLst maplayer = new OpCUpdateLayerLst();
            maplayer.layerNum = layerNum;
            maplayer.layerInfo = new OpCLayerInformation[layerNum];
            for (int i = 0; i < layerNum; i++)
            {
                int index = Convert.ToInt32(layerArr[i + 1].Split('=')[0].Substring(5));
                OpCLayerInformation pMapLayer =
_viewport.MapObj.getLayerInformation(index);
                pMapLayer.DisplayIndex = i;
                SetLayerStatus(ref pMapLayer, layerArr[i + 1].Split('=')[1]);
                maplayer.layerInfo[index] = pMapLayer;
            }
            _viewport.UpdateLayerLst(maplayer);
            HttpContext.Current.Session["viewport"] = _viewport;
        }
```

4. 图层状态控制的客户端具体操作如图：




图 5-2-7-3 步骤 1-选择查询全部图层按钮并单击

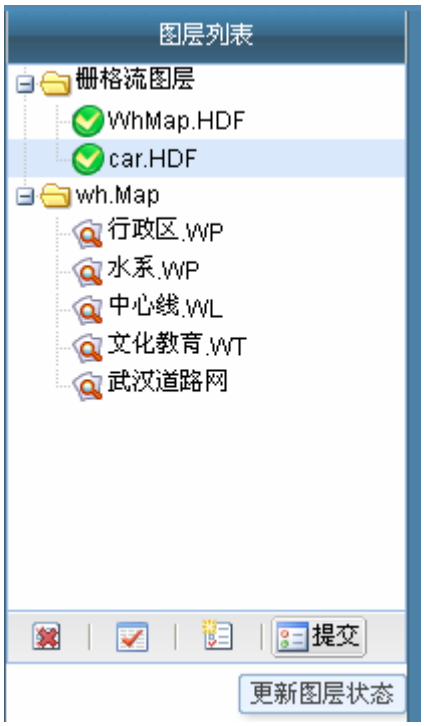



图 5-2-7-4 步骤 2-选择更新图层状态按钮并提交

## 5.2.8 地图查询功能

地图查询功能分为简单查询和条件查询。简单查询即基本查询，包括点击查询、拉框查询、圆查询、线查询和多边形查询五种；条件查询包括属性条件查询、点击条件查询、拉框条件查询、画圆条件查询、画线条件查询和多边形条件查询六种。这些功能都是在矢量地图的地图显示基础上实现的，所以对地图显示功能的透彻理解是做该开发的前提。地图查询功

能还涉及到闪烁和编辑功能，即查询到的点、线、区要素的定位显示和编辑。

地图查询功能实现流程为：首先通过客户端工具栏触发客户端脚本 Function.js，在 Function.js 中将各种查询和处理页面关联，最后调用后台页面 SelectAttribute.aspx 处理并生成查询结果。

具体实现流程为：

1. 在地图初始化 loadMap 脚本函数（位于 function.js 中）中加载图形搜索类和操作状态类。若这一步在地图显示中已经加载，则可跳过。

➢ **加载图形搜索类和操作状态类的代码：**

```
//图形搜索
pro = new IMSSearch(map);
map.addControl(pro,false);
//操作状态
var os = new IMSOperStatus();
map.addControl(os,false);
```

2. 在客户端工具栏（Menu.htm、MapFrame.aspx 页面）处添加地图查询功能按钮，进行事件编写及绑定。

Menu.htm 页面中地图查询的功能按钮呈现效果如下图：




图 5-2-8-1 简单查询和条件查询的按钮效果

➢ **地图查询功能按钮的代码：**

```
<img class="btncursor"
  onclick="top.onSelectToolClick('SelectByHitting','Point','SelectAttribute.aspx')"
            alt="点击查询" src="images/selectByHitting.gif" align="middle"border="0"><br>
            点击查询<br>
          <br>
<img class="btncursor"
  onclick="top.onSelectToolClick('SelectByRect','Rect','SelectAttribute.aspx')"
```

```
                alt="拉框查询" src="images/selectByRect.gif" align="middle"border="0"><br>
                   拉框查询<br>
                  <br>
  <img class="btncursor"
    onclick="top.onSelectToolClick('SelectByCircle','Circle','SelectAttribute.aspx')"
                alt="圆查询" src="images/selectByCircle.gif" align="middle"border="0"><br>
                   圆查询<br>
                   <br>
  <img class="btncursor"
    onclick="top.onSelectToolClick('SelectByPolyline','Polyline','SelectAttribute.aspx')"
                 alt="线查询" src="images/selectByPolyline.gif" align="middle"border="0"><br>
                 线查询<br>
                 <br>
 <img class="btncursor"
onclick="top.onSelectToolClick('SelectByPolygon','Polygon','SelectAttribute.aspx')"
 alt="多边形查询" src="images/selectByPolygon.gif"align="middle" border="0"><br>
              多边形查询<br>
              <br>
 <img class="btncursor" onclick="top.onDisplayToolClick('Condition',null)" alt="属性条件查询"
              src="images/selectByCondition.gif" align="middle" border="0"><br>
              属性条件查询<br>
              <br>
  <img class="btncursor"
onclick="top.onSelectToolClick('SelectByHittingAndAtt','Point','SelectByCondition.aspx')"
alt="点击条件查询" src="images/selectByHittingAndCondition.gif"align="middle" border="0"><br>
              点击条件查询<br>
              <br>
  <img class="btncursor"
onclick="top.onSelectToolClick('SelectByRectAndAtt','Rect','SelectByCondition.aspx')"
alt="拉框条件查询" src="images/selectByRectAndCondition.gif"align="middle" border="0"><br>
              拉框条件查询<br>
              <br>
 <img class="btncursor"
onclick="top.onSelectToolClick('SelectByCircleAndAtt','Circle','SelectByCondition.aspx')"
alt="画圆条件查询" src="images/selectByCircleAndCondition.gif"align="middle" border="0"><br>
              画圆条件查询<br>
              <br>
<img class="btncursor"
onclick="top.onSelectToolClick('SelectByPolylineAndAtt','Polyline','SelectByCondition.aspx')
" alt="画线条件查询" src="images/selectByPolylineAndConditon.gif"align="middle"
border="0"><br>
              画线条件查询<br>
              <br>
<img class="btncursor"
```

```
onclick="top.onSelectToolClick('SelectByPolygonAndAtt','Polygon','SelectByCondition.aspx')"
alt="多边形条件查询" src="images/selectByPolygonAndCondition.gif"align="middle"
border="0"><br>
            多边形条件查询<br>
            <br>
```

3. 在地图查询各功能按钮的 onclick 事件中调用了 onSelectToolClick 和 onDisplayToolClick 脚本函数进行处理，这些函数位于脚本库 function.js 中，这两个函数用户可以自定义编写，为不同的查询（点击，拉框，画圆，属性条件等）编写不同的自定义的处理方法。

➢ **onSelectToolClick()地图查询部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
        case "SelectByHitting":
        case "SelectByHittingAndAtt":
            map.setCurOper(2001);
            pro.setSelectByHitting(bufferSet(3),submitUrl);
        break;
        case "SelectByPolyline":
        case "SelectByPolylineAndAtt":
            map.setCurOper(2005);
            userActive = "SelectByPolygon";
            pro.setSelectByLine(bufferSet(3),submitUrl);
        break;
        case "SelectByRect":
        case "SelectByRectAndAtt":
            map.setCurOper(2002);
            pro.setSelectByRect(submitUrl);
        break;
        case "SelectByCircle":
        case "SelectByCircleAndAtt":
            map.setCurOper(2003);
            pro.setSelectByCircle(submitUrl);
        break;
        case "SelectByPolygon":
        case "SelectByPolygonAndAtt":
            map.setCurOper(2004);
            pro.setSelectByPolygon(submitUrl);
        break;
        default:
```

```
            var url = "VectorPage/"+TargetPage+"?rnd="+Math.random();
            renderSelectResult(url);
        break;
    }
}
```

上面的函数就是将各种查询和处理方法相关联。activeTool 参数：代表查询的类型； drawAction 参数：代表客户端鼠标的操作类型； TargetPage 参数：代表处理该查询的页面。

- **onDisplayToolClick()属性条件查询部分代码：**

```
function onDisplayToolClick(con)
{
    var type = con
    switch(type)
    {
        case "Condition":
            userActive = "SelectByCondition";
            subParam="";
            var url = "VectorPage/SelectByCondition.aspx?rnd="+Math.random();
            renderSelectResult(url);
        break;
    }
}
```

上面的函数就是将属性条件查询和处理方法相关联。con 参数值为 Condition，代表属性查询的类型。该函数将转到 VectorPage 文件夹下的服务器后台页面（SelectByCondition.aspx）中，对属性条件查询设置具体的查询条件，再将条件参数等传到服务器后台页面（SelectAttribute.aspx）中进行处理。

SelectByCondition.aspx 页面中的条件查询的条件编辑面板如下图：




图 5-2-8-1 条件查询的条件编辑面板

注：以下第 4、5 步骤为非属性条件查询的具体流程。

4. 在 onSelectToolClick 函数中首先根据参数调用 setCurOper 函数，设置地图操作状态为相应类型。在 setCurOper 函数中调用 setQueryMode 脚本函数，根据 drawAction 参数（客户端鼠标在地图上的操作类型）添加事件监听并调用相应的脚本处理函数，再处理函数中获取到的相应参数 param（为 submitUrl 函数的参数）。

- **setCurOper()地图查询部分代码：**

```
this.setCurOper=function(operType)
    {
        this.operType = parseInt(operType);
        this.clearOper();
        switch(this.operType)
        {
            case 2001:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByHitting","Point");
            break;
            case 2002:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByRect","Rect");
            break;
            case 2003:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByCircle","Circle");
            break;
            case 2004:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByPolygon","Area");
            break;
            case 2005:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByLine","Polyline");
            break;
      }
    };
```

- **setQueryMode()代码：**

```
this.setQueryMode = function(activeTool,drawAction)
{
     thisObj.activeTool = activeTool;
     thisObj.drawAction = drawAction;
     switch(drawAction){
        case "Point":
         if(this.listenerPoint){
            thisObj.addEventListener("mouseup",thisObj.so.selectByHitting , false);
```

```
            this.listenerPoint = false;
          }
          break ;
        case "Circle" :
          if(this.listenerCircle){
            thisObj.addEventListener("mouseup",thisObj.so.selectByCircle, false);
            this.listenerCircle = false;
           }
           break ;
        case "Rect" :
          if(this.listenerRect){
            thisObj.addEventListener("mouseup",thisObj.so.selectByRect, false);
            this.listenerRect = false;
           }
           break ;
        case "Area" :
          if(this.listenerPolygon){
            thisObj.addEventListener("dblclick",thisObj.so.selectByPolygon, false);
              this.listenerPolygon = false;
            }
            break ;
        case "Polyline":
          if(this.listenerPolyline){
            thisObj.addEventListener("dblclick",thisObj.so.selectByLine, false);
            this.listenerPolyline = false;
           }
           break;
        default:
            break ;
    }
};
```

- **以点击查询为例，setSelectByHitting()代码：**

```
this.setSelectByHitting = function(ioffset,dealFun)
{
   this.offset[0] = ioffset;
   this.dealFunArr[0] = dealFun;
    };
```

- **以点击查询为例，selectByHitting()代码为：**

```
this.selectByHitting= function()
{
    if(thisObj.activeTool!="SelectByHitting"){
        thisObj.removeEventListener("mouseup",thisObj.so.selectByHitting , false) ;
        thisObj.so.listenerPoint =true;
        return ;
```

```
        }
        var dx = thisObj.so.offset[0];
        var X = Math.max(thisObj.mdLogicX,thisObj.muLogicX);
        var Y = Math.max(thisObj.mdLogicY,thisObj.muLogicY);
        //用于计算网格编号
        var startX=X-dx;
        var startY=Y-dx;
        var endX=X+dx;
        var endY=Y+dx;
        var
param="centerX="+X+"&centerY="+Y+"&radius="+dx+"&GridCodes="+thisObj.getGridCodes(startX,sta
rtY,endX,endY)+"&lev="+(thisObj.inf[2]+1)+"&sType=SelectByHitting";
        if(thisObj.so.dealFunArr[0]!=null)
            thisObj.so.dealFunArr[0](param);
        };
```

通过鼠标单击事件获取点击的 x、y 坐标等参数值，调用 dealFunArr[0]，即 submitUrl 函数，把参数组 param 设置为 submitUrl 函数的参数。

5. 然后再分别调用自定义函数。各函数中的 submitUrl 参数是一个对象，是对各种查询的处理方法。由 submitUrl()处理将查询参数传到相应的服务器页面处理，并通过 renderSelectResult 函数将结果设置到相应位置显示。

➢ **submitUrl()代码：**

```
function submitUrl(param)
{
    var url = "VectorPage/"+userParam+"?" + param+"&op="+userActive;
    subParam = param;
    loadBar(1);
    renderSelectResult(url);
}
```

该脚本函数是各查询的处理方法，这里的 param 是在上面所介绍的核心脚本库 IMSMaps.js 中生成的。各种查询相应的名值对参数如下：

点击查询：centerX="+X+"&centerY="+Y+"&radius="+dx

拉框查询：startX="+startX+"&startY="+startY+"&endX="+endX+"&endY="+endY

画圆查询：centerX="+centerX+"&centerY="+centerY+"&radius="+radius

线查询：　　XArray="+Xarray+"&YArray="+Yarray

多边形查询：XArray="+Xarray+"&YArray="+Yarray

用户根据名值对参数，在自己的处理页面中即可取得相应的值。

如点击查询后台处理页面获取客户端的所点击的点的坐标和查询半径：

```
xmin = Convert.ToDouble(Request.QueryString["centerX"]);
ymin = Convert.ToDouble(Request.QueryString["centerY"]);
distance = Convert.ToDouble(Request.QueryString["radius"]);
```

➢ **renderSelectResult()代码：**

```
function renderSelectResult(url){
        //设置进度条为繁忙状态
```

```
var obj = document.getElementById("rsFrame");
if(obj==null)
{
    document.getElementById("rsPanel").innerHTML='<iframe id="rsFrame" style="BORDER: 0px; WIDTH: 100%; HEIGHT: 100%" src="" frameBorder="0" scrolling="auto" onload="loadBar(0)"></iframe>';
    obj = document.getElementById("rsFrame");
}
obj.src = url;
switchdiv(1);
}
```

该函数将处理结果显示到一个叫做 rsFrame 的 iframe 中，用户可以根据自己的需要修改。

注：如果是条件查询，则与属性条件查询一样，先转到 VectorPage 文件夹下的 SelectByCondition.aspx 页面设置具体的查询条件，然后再将条件参数与坐标值等参数传到服务器后台页面 SelectAttribute.aspx 中处理。

6. 服务器后台页面 SelectAttribute.aspx，用于处理各种查询。

- **SelectAttribute 页面查询处理代码：**

```
if (firstSelect != 1)
{
    switch (op)
    {
        case "SelectByHitting":
            点击查询
        case "SelectByRect":
            矩形查询
        case "SelectByCircle":
            画圆查询
        case "SelectByPolygon":
            多边形查询
        case "SelectByPolyline":
            线查询
        case "SelectByCondition":
            条件查询
            break;
        case "SelectByHittingAndAtt":
            点击条件查询
            break;
        case "SelectByRectAndAtt":
            矩形条件查询
            break;
        case "SelectByCircleAndAtt":
            圆条件查询
            break;
        case "SelectByPolygonAndAtt":
            多边形条件查询
            break;
        default:
            break;
    }
}
else
{
    ShowAttView();
}
```

这个页面根据查询类型（userActive）来接收不同的客户端参数，分别进行各项查询的处理。

➢ **以点击查询为例，具体代码为：**

```
//点击点的坐标
 xmin = Convert.ToDouble(Request.QueryString["centerX"]);
 ymin = Convert.ToDouble(Request.QueryString["centerY"]);
//误差半径
 distance = Convert.ToDouble(Request.QueryString["radius"]);
 p2d = new OpCPoint2D(xmin, ymin);
 OpCPointSelectClass pointSelectObj = new OpCPointSelectClass();
 pointSelectObj.Dot = p2d;
 pointSelectObj.Distance = distance;
 pointSelectObj.LayerIndex = CheckLayer();
 try
    {
        viewport.SelectByHitting(pointSelectObj);
        //保存上次查询对象
        page = new OpCGetOnePage();
        QueryTypeEnum querytype = QueryTypeEnum.SelectByHitting;
          page.PntSelect = pointSelectObj;
          page.SelectType = querytype;
          Session["page"] = page;
          Session["viewport"] = viewport;
          ShowAttView();
       }
       catch
       {
          Response.Write("<script language=javascript>alert('没有查询到任何实体!
')</script>");
          return;
        }
      break;
```

在点击查询处理中，首先接收客户端传过来的参数，其中 centerX 是点击的 x 坐标，centerY 是点击的 y 坐标，radius 是点击查询的查询半径；然后调用 GIS 服务器类库中的 SelectByHitting()方法来查询。

7. 以点击查询为例，设置所有图层为可查询状态后，其客户端具体操作如图：




图 5-2-8-3 步骤 1-选择点击查询按钮并单击

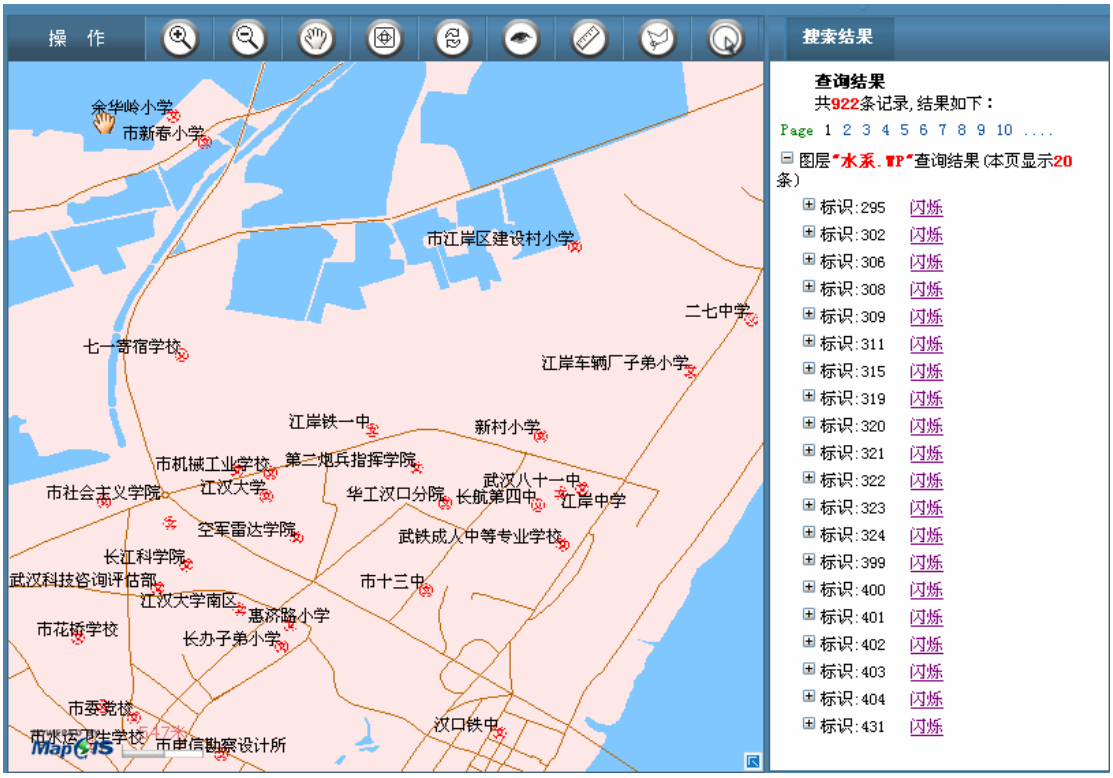



图 5-2-8-4 步骤 2-在地图上点击后显示查询结果

以上步骤实现了地图查询功能，下面介绍如何实现闪烁和编辑查询结果。闪烁和编辑功能涉及到的客户端操作较多，需要调用核心脚本库的方法，实现起来比较复杂。其实现流程为：

1. 首先在 SelectAttribute.aspx 页面客户端添加相应的代码。

➢ **闪烁和编辑功能客户端代码：**

```
<div onmouseover="this.style.background='#abcdf5'"
onmouseout="this.style.background='white'">
    <img alt="" src="../images/plus.gif" style="cursor:pointer" onclick="expend(this)" />
标识:<%= featureID%> <span style="TEXT-DECORATION:underline;cursor:pointer;"></span>
     <% if (layerSelectRlt[i].LayerIndex == editlayer){%>
    <span style="TEXT-DECORATION: underline;cursor:pointer; color:Purple" onclick =
"edit(<%=i %>,<%=j %>,'<%=layerName %>')">编辑</span><%} %>
    <span style="TEXT-DECORATION: underline;cursor:pointer; color:Purple" onclick =
"flash(<%=i %>,<%=j %>)">闪烁</span>
</div>
```

在上面的代码中，脚本函数 edit()进行编辑功能处理，flash()进行定位闪烁功能处理。这两个脚本函数都位于 SelectAttribute.aspx 页面中。

2. **定位闪烁功能：**主要通过脚本函数 flash()处理，将相关参数传到 VectorPage 文件夹下的 SelectGeometry.aspx 页面进行处理，与 GIS 服务器端交互完成闪烁功能。

➢ **flash()代码：**

```
function flash(layerIndex,countIndex)
{
```

```
        var arrayIndex=0;
        if(myFlashObjPoint!=null&&myFlashObjPoint.length>0)
        {
            for(arrayIndex=0;arrayIndex<myFlashObjPoint.length;arrayIndex++)
            {
                if(myFlashObjPoint[arrayIndex]!=null)
                {
                    myFlashObjPoint[arrayIndex].dispose();
                }
            }
         }
        if(myFlashObjLine!=null&&myFlashObjLine.length>0)
        {
            for(arrayIndex=0;arrayIndex<myFlashObjLine.length;arrayIndex++)
            {
                if(myFlashObjLine[arrayIndex]!=null)
                {
                    myFlashObjLine[arrayIndex].dispose();
                }
            }
        }
        if(myFlashObjPolygon!=null&&myFlashObjPolygon.length>0)
        {
            for(arrayIndex=0;arrayIndex<myFlashObjPolygon.length;arrayIndex++)
            {
                if(myFlashObjPolygon[arrayIndex]!=null)
                {
                    myFlashObjPolygon[arrayIndex].dispose();
                    delete myFlashObjPolygon[arrayIndex];
                }
            }
         }
       var url =
"VectorPage/SelectGeometry.aspx?layerIndex="+layerIndex+"&countIndex="+countIndex+"&qryT
ype=sel&rnd="+Math.random();
       var pntStr = top.AJAXRequest(url,false,"GET");
       var xmin = 0;
       var ymin = 0;
       var xmax = 0;
       var ymax = 0;
       if(pntStr.indexOf("(")!=-1)
       {
            alert("获取几何点信息失败！请重新查询");
       }
```

```
        else
        {
            var geoArr = pntStr.split('*'); //分割点线区
            //点
            if (geoArr.length>0&&geoArr[0] != "")
             {
                var pntArr = geoArr[0].split(';'); //分割各个点
                for (var i = 0; i < pntArr.length; i++)
                {
                    if(pntArr[i]!="")
                    {
                        pntDotArr =pntArr[i].split(','); //分割各个点的X,Y坐标
                        if(pntDotArr.length!=2){
                            return;
                        }
                        pntScreen = mapObj.logicToScreen(pntDotArr[0],pntDotArr[1]);
                        idIndex=idNum;
                        intervalId[idIndex] =
setTimeout("checkImgLoaded("+i+","+"'point',"+idIndex+")", 1000);
                        idNum++;
                    }
                }
                mapObj.pntLocation(pntDotArr[0],pntDotArr[1]);
            }
            //线
            if (geoArr.length>1&&geoArr[1] != "")
            {
                var lineArr = geoArr[1].split(';');//分割线
                for (var i = 0; i < lineArr.length; i++)
                {
                    if(lineArr[i]!="")
                    {
                        lineDotArr = lineArr[i].split(',');//分割各条线上的点坐标
                        idIndex=idNum;
                        intervalId[idIndex] =
setTimeout("checkImgLoaded("+i+","+"'line',"+idIndex+")", 1000);
                        idNum++;
                    }
                }
                mapObj.pntLocation(lineDotArr[0],lineDotArr[1]);
            }
            //区
            if (geoArr.length>2&&geoArr[2] != "")
            {
```

```
                var regArr = geoArr[2].split(';');//分割区中的复杂线
                for (var i = 0; i < regArr.length; i++)
                {
                    if(regArr[i]!="")
                    {
                        var regMultiArr=regArr[i].split(':');//分割复杂线中的所有弧段
                        for(var j=0;j<regMultiArr.length;j++)
                        {
                            regDotArr[j] = regMultiArr[j].split(',');//分割弧段上的各个点
坐标
                            idIndex=idNum;
                            intervalId[idIndex] =
setTimeout("checkImgLoaded("+j+","+"'reg',"+idIndex+")", 1000);
                            idNum++;
                        }

                    }
                }
                mapObj.pntLocation(regDotArr[0][0],regDotArr[0][1]);
            }
        }
    }
```

以上函数将参数传到 SelectGeometry.aspx 页面处理后，返回要素坐标的字符串，然后取出其闪烁要素的坐标，再调用 checkImgLoaded 函数对要素进行闪烁。具体步骤如下：

（1）根据查询结果的索引查询该要素的坐标（分别为 x 坐标数组和 y 坐标数组）。如代码所示，通过 SelectGeometry.aspx 页面来查询该要素的坐标。其中 layerIndex 即为该要素的索引。

- **以点要素的闪烁为例，在 SelectGeometry.aspx 中部分处理代码：**

```
//----------------------获取点查询结果中所有点数   start-------------------------//
        pntNum = featureGeometry.PointNumber;
        OpCPoint2D[] point = featureGeometry.Points;
        for (int i = 0; i < pntNum; i++)
        {
            sb.Append(point[i].X.ToString()).Append(",");
            sb.Append(point[i].Y.ToString()).Append(";");
        }
        if (sb.Length != 0)
        {
            sb.Remove(sb.Length - 1, 1);
        }
        sb.Append("*");
//----------------------获取点查询结果中所有点数   end-------------------------//
```

（2）调用同步的 AJAX 请求返回查询结果。AJAXRequest 方法的三个参数的意义分别为 url 指定处理页面，false 指定不用异步请求，get 是提交的方式。

（3）处理返回的坐标 pntStr。返回的坐标是一个字符串，通过 split 方法将之分割，为下一步的闪烁做准备。

（4）最后调用 checkImgLoaded 函数对要素进行闪烁。

➢ **以点要素的闪烁为例，checkImgLoaded()中部分代码：**

```
function checkImgLoaded(j,drawType,idindex)
    {
        var flashTimes,flashInterval,flashColor;
        var n=0;
        if(mapObj.loader.isBusy==false)
        {
            flashTimes = top.flashTimes
            flashInterval = top.flashInterval
            switch(drawType)
            {
                case "point":
                    var pntDiv = top.document.getElementById("pntFlashDiv");
                    if(pntDiv==null)
                    {
                        pntDiv = top.document.createElement("div");
                        pntDiv.id = "pntFlashDiv";
                        pntDiv.style.position = "absolute";
                        pntDiv.style.visibility = "hidden";
                        mapObj.addControl(pntDiv,true);
                    }
                    if(pntDotArr.length==2)
                    {
                        times = top.flashTimes*2;
                        clearTimeout(pntFlashTime);
                        DrawPnt();
                        FlashPnt();
                    }
                    break;
                default:
                    break;
            }
            clearTimeout(intervalId[idindex]);
        }
    }
```

3. **编辑功能：**主要通过脚本函数 edit()处理，将相关参数传到 VectorPage 文件夹下的 AttEdit.aspx 页面进行处理，与 GIS 服务器端交互完成编辑功能。

➢ **edit()代码：**

```
function edit(selLayerIndex,countIndex,layer)
 {
        var url =
"VectorPage/AttEdit.aspx?layer="+layer+"&selLayerIndex="+selLayerIndex+"&countIndex="+co
untIndex+"&rnd="+Math.random();
        top.document.getElementById("rsFrame").src = url;
 }
```

通过点击编辑按钮，调用 edit()转到 AttEdit.aspx 页面。该编辑页面中包括：移动，删除，更新和缓冲区分析等功能。以点编辑为例来说明如何实现。

AttEdit.aspx 页面编辑面板的效果图如下：

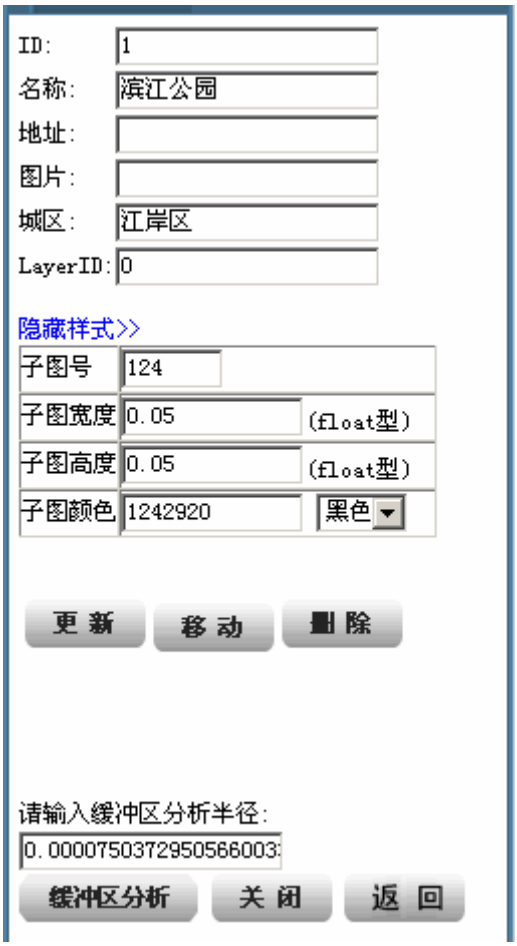



图 5-2-8-5 编辑面板

（1）**点的移动功能：**在右侧子框架的 AttEdit.aspx 编辑面板上点击移动，则在地图上显示这个要素，此时可以在地图上拖动图标了，移动图标到需要的位置后再点击更新，就将新的位置保存到服务器端了。下面主要介绍点要素移动的几个函数。

➢ **移动按钮客户端代码：**

```
<td height="20px" ><img ID="btn_Move" alt="移动"  src="../images/btn_move.gif"
style="margin:0;padding:0;cursor:pointer" onclick="onMoveFeature(<%=selLayerIndex%>,
<%=countIndex%>)"/>
</td>
```

通过触发点击事件中的 onMoveFeature 函数进行处理。

➢ **onMoveFeature()代码：**

```
function onMoveFeature(layerIndex,countIndex)
    {
       var url =
"VectorPage/SelectGeometry.aspx?layerIndex="+layerIndex+"&countIndex="+countIndex+"&qryT
ype=sel&rnd="+Math.random();
       var pntStr = top.AJAXRequest(url,false,"GET");
       var xmin = 0;
       var ymin = 0;
       var xmax = 0;
       var ymax = 0;
       if(pntStr.indexOf("(")!=-1)
       {
            alert("获取几何点信息失败！请重新查询");
       }
       else
       {
           var geoArr = pntStr.split('*'); //分割点线区
           //点
           if (geoArr.length>0&&geoArr[0] != "")
            {
                var pntArr = geoArr[0].split(';'); //分割各个点
                for (var i = 0; i < pntArr.length; i++)
                {
                    if(pntArr[i]!="")
                    {
                        pntDotArr =pntArr[i].split(','); //分割各个点的X,Y坐标
                        idIndex=idNum;
                        intervalId[idIndex] =
setInterval("checkImgLoaded("+i+","+"'point',"+idIndex+")", 1000);
                        idNum++;
                    }
                }
                mapObj.pntLocation(pntDotArr[0],pntDotArr[1]);
            }
    }
       return false;
}
```

上述代码是点移动功能显示要素的函数，该功能显示要素的原理与闪烁功能中的闪烁显示类似，不再重复说明。最后，checkImgLoaded 函数显示要素时，还调用了 addEventEditDiv 脚本函数进行处理。

➢ **addEventEditDiv()代码：**

```
function addEventEditDiv()
```

```
{
    //if(lisFlg) return;
    mapObj.drawAction = "editPoints";
    mapObj.activeTool="";
    EventObj.addListener(pntDiv.firstChild, "mousedown",editDivDown, false);
    EventObj.addListener(pntDiv.firstChild, "mousemove",editDivMove, false);
    EventObj.addListener(pntDiv.firstChild, "mouseup",editDivUp, false);
    EventObj.addListener(pntDiv.firstChild, "mouseout",editdivOut, false);
    lisFlg = true;
    }
```

addEventEditDiv 是向主图上注册 mouseDown,mouseMove,mouseUp 三个方法，这里用到的 EventObj.addListener(pntDiv, "mousedown",editDivDown, false)方法,是用来监听 pntDiv 上的 mousedown 事件，用 editDivDown 方法进行处理，false 表示阻止事件冒泡。另两个方法与第一个类似。

- **editDivDown()代码：**

```
function editDivDown(Evnt)
{
    if(mapObj.drawAction!="editPoints"){
        EventObj.removeListener(pntDiv.firstChild, "mousedown", editDivDown, false);
    }
    var e=Evnt?Evnt:window.event;
 var mX = Evnt?e.clientX:e.pageX;
     var mY = Evnt?e.clientY:e.pageY;
    mouseDownX=mX;
    mouseDownY=mY;
    xyLogic = [mapObj.xLogic,mapObj.yLogic];
    mouseDown = true;
    }
```

editDivDown 是处理 pntDiv 上的 mousedown 事件。函数内部首先判断当前 drawAction 是否为 editPoints。若不是 editPoints 则移除 pntDiv 上的 mousedown 事件的监听。否则保存当前的坐标。

- **editDivMove()代码：**

```
function editDivMove(Evnt)
{
    if(mapObj.drawAction!="editPoints"){
        EventObj.removeListener(pntDiv.firstChild, "mousemove", editDivMove, false);
    }
    if(mouseDown==true)
    {
        var e=Evnt?Evnt:window.event;
     var mX = Evnt?e.clientX:e.pageX;
         var mY = Evnt?e.clientY:e.pageY;
```

```
        pntDivTop =pntDiv.style.top==""?0:parseInt(pntDiv.style.top);
        pntDivLeft =pntDiv.style.left==""?0:parseInt(pntDiv.style.left);
        pntDiv.style.top= pntDivTop+(mY-mouseDownY)+'px';
        pntDiv.style.left= pntDivLeft+(mX-mouseDownX)+'px';
        mouseDownX = mX;
        mouseDownY = mY;
    }
    }
```

editDivMove 是处理 pntDiv 上的 mousemove 事件。函数内部首先判断当前 drawAction 是否为 editPoints。若不是 editPoints 则移除 pntDiv 上的 mousemove 事件的监听。接着改变 pntDiv 的位置，即：更改 pntDiv 的 left 和 top 值。这里需要注意的是要将当前的坐标赋值给 mouseDownX，这样才能有移动的效果。

➢ **editDivUp()代码：**

```
function editDivUp(Evnt)
{
    if(mapObj.drawAction!="editPoints"){
        EventObj.removeListener(pntDiv.firstChild, "mouseup", editDivUp, false);
    }
    mouseDown = false;
    var offX = mapObj.xLogic - xyLogic[0];
    var offY = xyLogic[1] -mapObj.yLogic;
    var offXY = [offX,offY];
    form1.featureOffset.value = offXY;
    }
```

editDivUp 是处理 pntDiv 上的 mouseup 事件。函数内部首先判断当前 drawAction 是否为 editPoints。若不是 editPoints 则移除 pntDiv 上的 mouseup 事件的监听。接着记录鼠标弹起时的坐标值，为更新要素的位置做准备。

至此，移动要素位置的功能就完成了，可以点击更新按钮更新要素的位置。

（2）**更新要素功能**：在右侧子框架的 AttEdit.aspx 编辑面板上修改要素属性值，通过点击客户端的更新按钮，触发事件 Btn_Update_Click 验证参数类型的合法性并调用 UpdateFeature()来更新要素几何信息，调用 UpdateAttribute()和 UpdateStyleInfo()来更新要素属性值。下面以修改子图高度为例，主要介绍更新要素的代码和函数。

➢ **子图高度的客户端代码：**

```
<tr id="tr_height">
        <td><asp:Label ID="Label_height" runat='server'>子图高度</asp:Label></td>
        <td style="width: 156px"><input id="patternHeight" name="para" runat="server"
type="text" value="20.0" style="width:85px" />(float型)</td>
</tr>
```

➢ **更新按钮的客户端代码：**

```
<td height="20px"><asp:ImageButton ID="Btn_Update" runat="server"  AlternateText="更新"
```

```
OnClick="Btn_Update_Click"  ImageUrl="~/images/btn_update.gif"/>
```

➢ **Btn_Update_Click()后台处理代码：**

```
protected void Btn_Update_Click(object sender, EventArgs e)
    {
        #region 验证参数类型的合法性
        略
        #endregion
        #region 更新要素几何信息
        if (Session["viewport"] == null)
        {
            Response.Write("地图文档装载失败或请求服务超时");
            Response.End();
        }
        viewport = (CMapView)Session["viewport"];
        UpdateFeature();
        #endregion
        //更新要素属性值
        UpdateAttribute(fldVal);
        if (Request.Form["showPattern"]!=null&&Request.Form["showPattern"]=="true")
        {
            UpdateStyleInfo();
        }
    }
```

以上代码先验证参数类型的合法性，然后通过 UpdateFeature()更新要素几何信息，最后通过 UpdateAttribute()和 UpdateStyleInfo()更新要素属性值。在更新子图高度属性时，主要是通过 UpdateStyleInfo()来更新其样式信息。

➢ **UpdateStyleInfo()后台处理代码：**

```
    /// <summary>
    /// 更新样式信息
    /// </summary>
    private void UpdateStyleInfo()
    {
        OpCUpdateFeatureStyleInfo updatePatter = new OpCUpdateFeatureStyleInfo();
        updatePatter.LayerIndex = layerIndex;
        updatePatter.FeatureId = featureID;
        updatePatter.PatternIndex = Convert.ToInt32(this.patternIndex.Value);
        updatePatter.MarginWidth = Convert.ToSingle(this.patternWidth.Value);
        updatePatter.MarginHeight = Convert.ToSingle(this.patternHeight.Value);
        updatePatter.Color = Convert.ToInt32(this.color.Value);
        int geoFlag = Convert.ToInt32(this.geoFlag.Value);
        try
        {
            updatePatter.FeatureType = geoFlag;
```

```
            viewport.UpdateFeatureStyleInfo(updatePatter);
        }
        catch (Exception ex)
        {
            Response.Write("<script language=javascript>alert('" + ex.Message.ToString() +
"')</script>");
        }
    }
```

这是更新子图高度功能的后台代码。更新要素属性值的过程比较简单，通过服务器端控件保存要素的属性值，在后台代码中获取属性值，更新即可。

（3）**缓冲区分析功能**：在右侧子框架的 AttEdit.aspx 编辑面板上输入缓冲半径，通过点击客户端的缓冲分析按钮，触发事件 Btn_BufferAnalyse_Submit_Click 在后台页面进行处理。

缓冲区分析的编辑面板如下图：




图 5-2-8-6 缓冲区分析编辑面板

- **缓冲区分析客户端代码：**

```
请输入缓冲区分析半径:<br/>
<asp:TextBox ID="tbBuffer" runat="server"></asp:TextBox><br />
 <asp:ImageButton ID="Btn_BufferAnalyse_Submit" runat="server"
OnClick="Btn_BufferAnalyse_Submit_Click" ImageUrl="../images/btn_bufanalyse.gif" />
 <img id="Button1" type="button" alt="关闭" onclick="Close()"
src="../images/btn_close.gif" style=" cursor:hand;" />
 <img  id="Btn_Back"  src="../images/btn_back.jpg"  alt=" 返 回 "  onclick="BackToQuery()"
style=" cursor:hand;" />
```

- **Btn_BufferAnalyse_Submit_Click()后台处理代码：**

```
protected void Btn_BufferAnalyse_Submit_Click(object sender, EventArgs e)
    {
        if (selRlt == null)
        {
            Response.Write("<script language=javascript>alert('没有查询到任何实体!
')</script>");
```

```
            return;
        }
        //获取该图层查到的记录总数
        int countIndex = Convert.ToInt32(Request.QueryString["countIndex"]);
        //获取buffer分析半径
        double radius = 0.0;
        try
        {
            radius = Convert.ToDouble(tbBuffer.Text);
        }
        catch
        {
            Response.Write("<script language=javascript>alert('请确认输入参数的合法性!
')</script>");
            return;
        }
        if (radius < 0)
        {
            Response.Write("<script language=javascript>alert('输入参数不可为负值!
')</script>");
            return;
        }
        //取指定图层查询到的实体号
        long[] featureIDs = new long[1];
        featureIDs[0] = layerSelectRlt[selLayerIndex].FeatureIDs[countIndex];
        //声明缓冲区分析对象、返回值描述对象
        OpCSpatialBuffer buffer = new OpCSpatialBuffer();
        Random rand = new Random();
        //缓冲区分析后生成文件的名称
        buffer.FeatureClassName = "buffer" + Convert.ToString(rand.NextDouble() * 1000);
        //要进行缓冲区分析的要素实体号
        buffer.FeatureIDS = featureIDs;
        //要进行缓冲区分析的要素个数
        buffer.FeatureNumber = Convert.ToInt16(featureIDs.Length);
        //缓冲区分析半径
        buffer.Radius = radius;
        //要进行缓冲区分析的图层索引号
        buffer.LayerIndex = layerSelectRlt[selLayerIndex].LayerIndex;
        //调用缓冲区分析函数，并通过返回值验证分析是否成功
        OpCSpatialOperResult SpatialOperRlt = null;
        try
        {
            SpatialOperRlt = viewport.SpatialAnalysisBuffer(buffer);
        }
```

```
            catch
            {
                Response.Write("<script language=javascript>alert('缓冲区分析失败!
');window.close();</script>");
                return;
            }
            Random rnd=new Random();
            string url = "AnalyRlt.aspx?operation=4&operFlg=1&fileName=" +
viewport.ZipFileName+"&rnd="+rnd.Next();
            Response.Write("<script language=javascript>window.open(\"" + url +
"\",'_self','width=200,height=200')</script>");
        }
```

以上代码首先将缓冲半径保存在服务器端控件<asp:TextBox>中，然后在后台页面中调用 GIS 服务端组件，根据缓冲半径生成结果，最后转到 AnalyRlt.aspx 页面显示分析结果。

地图查询中各查询具体操作、要素的闪烁与编辑功能的客户端操作请参见行业版的操作手册。

## 5.2.9 地图编辑功能

地图编辑主要实现在地图文档中动态地添加点、添加线、添加区和修改要素功能，并实时更新地图文档。地图文档的编辑功能是常见的应用之一。

具体实现流程如下：

1. 在地图初始化 loadMap 脚本函数（位于 function.js 中）中加载图形搜索类和操作状态类。若这一步在地图显示中已经加载，则可跳过。

➢ **加载图形搜索类和操作状态类的代码：**

```
//图形搜索类
pro = new IMSSearch(map);
map.addControl(pro,false);
//操作状态
var os = new IMSOperStatus();
map.addControl(os,false);
```

2. 在客户端（MapFrame.aspx、menu.htm 页面）添加地图编辑的功能按钮，进行事件编写及绑定。

   在 menu.htm 页面客户端的地图编辑功能按钮效果图如下：

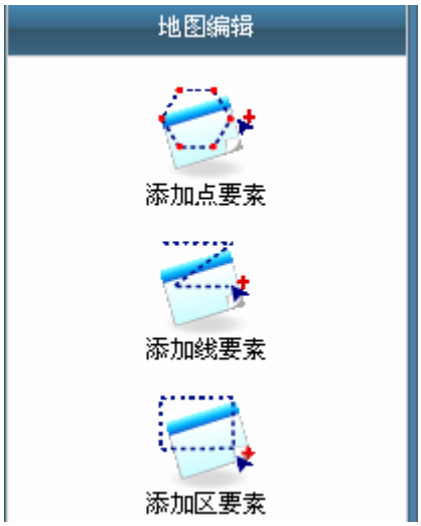



图 5-2-9-1 地图编辑功能按钮

➢ **地图编辑功能按钮的代码：**

```
<img class="btncursor" onclick="top.onSelectToolClick('0','Point','AddFeature.aspx')"
            alt="添加点要素" src="images/appendPnt.gif" align="middle"border="0"><br>
            添加点要素<br>
            <br>
<img class="btncursor" onclick="top.onSelectToolClick('1','Polyline','AddFeature.aspx')"
            alt="添加线要素" src="images/appendLin.gif" align="middle"border="0"><br>
            添加线要素<br>
            <br>
<img class="btncursor" onclick="top.onSelectToolClick('2','Polygon','AddFeature.aspx')"
            alt="添加区要素" src="images/appendReg.gif" align="middle"border="0"><br>
            添加区要素<br>
            <br>
```

3. 在地图编辑各功能按钮的 onclick 事件调用了 onSelectToolClick 函数进行处理，该函数位于脚本库 function.js 中。

➢ **onSelectToolClick()地图编辑部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
        case "0":                          //点编辑
            map.setCurOper(2001);
            pro.setSelectByHitting(1,submitUrl);
        break;
        case "1":                          //线编辑
            map.setCurOper(2005);
            pro.setSelectByLine(0,submitUrl);
```

```
            break;
            case "2":                                    //区编辑
                map.setCurOper(2004);
                pro.setSelectByPolygon(submitUrl);
            break;
            default:
                var url = "VectorPage/"+TargetPage+"?rnd="+Math.random();
                renderSelectResult(url);
            break;
        }
}
```

上面的函数就是将各种编辑和处理方法相关联。activeTool 参数：代表编辑的类型； drawAction 参数：代表客户端鼠标的操作类型； TargetPage 参数：代表处理该编辑的服务器页面。

4. 与地图查询类似，在 onSelectToolClick 函数中首先根据参数调用 setCurOper 函数设置地图操作状态为相应类型，在 setCurOper 函数中调用 setQueryMode 函数，根据 drawAction 参数（客户端鼠标在地图上的操作类型）添加事件监听并调用相应的脚本处理函数，在处理函数中获取相应的参数 param（为 submitUrl 函数的参数）。

- **setCurOper()地图编辑部分代码：**

```
this.setCurOper=function(operType)
    {
        this.operType = parseInt(operType);
        this.clearOper();
        switch(this.operType)
        {
            case 2001:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByHitting","Point");
            break;
            case 2004:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByPolygon","Area");
            break;
            case 2005:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByLine","Polyline");
            break;
      }
    };
```

- **setQueryMode()部分代码：**

```
this.setQueryMode = function(activeTool,drawAction)
```

```
{
     thisObj.activeTool = activeTool;
     thisObj.drawAction = drawAction;
     switch(drawAction){
        case "Point":
         if(this.listenerPoint){
            thisObj.addEventListener("mouseup",thisObj.so.selectByHitting , false);
             this.listenerPoint = false;
          }
          break ;
        case "Area" :
          if(this.listenerPolygon){
            thisObj.addEventListener("dblclick",thisObj.so.selectByPolygon, false);
              this.listenerPolygon = false;
            }
            break ;
        case "Polyline":
          if(this.listenerPolyline){
            thisObj.addEventListener("dblclick",thisObj.so.selectByLine, false);
            this.listenerPolyline = false;
           }
           break;
        default:
            break ;
     }
};
```

- **以添加点要素为例，setSelectByHitting()代码：**

```
this.setSelectByHitting = function(ioffset,dealFun)
{
   this.offset[0] = ioffset;
   this.dealFunArr[0] = dealFun;
    };
```

- **以添加点要素为例，selectByHitting()代码为：**

```
this.selectByHitting= function()
{
    if(thisObj.activeTool!="SelectByHitting"){
        thisObj.removeEventListener("mouseup",thisObj.so.selectByHitting , false) ;
        thisObj.so.listenerPoint =true;
        return ;
    }
    var dx = thisObj.so.offset[0];
    var X = Math.max(thisObj.mdLogicX,thisObj.muLogicX);
```

```
        var Y = Math.max(thisObj.mdLogicY,thisObj.muLogicY);
        //用于计算网格编号
        var startX=X-dx;
        var startY=Y-dx;
        var endX=X+dx;
        var endY=Y+dx;
        var
param="centerX="+X+"&centerY="+Y+"&radius="+dx+"&GridCodes="+thisObj.getGridCodes(startX,sta
rtY,endX,endY)+"&lev="+(thisObj.inf[2]+1)+"&sType=SelectByHitting";
        if(thisObj.so.dealFunArr[0]!=null)
            thisObj.so.dealFunArr[0](param);
        };
```

通过鼠标事件获取点击的 x、y 坐标值等参数值，调用 dealFunArr[0]，即 submitUrl 函数，把参数组 param 设置为 submitUrl 函数的参数。

5. 然后分别调用自定义函数处理。各函数中的 submitUrl 参数是一个对象，它就是对各种编辑的处理方法。由 submitUrl()处理将编辑参数传到相应的服务器页面处理，并通过 renderSelectResult 函数将结果返回。

- **submitUrl()代码：**

```
function submitUrl(param)
{
    var url = "VectorPage/"+userParam+"?" + param+"&op="+userActive;
    subParam = param;
    loadBar(1);
    renderSelectResult(url);
}
```

该函数是各种编辑的处理函数，其中 userParam 的值为从客户端传过来的 AddFeature.aspx 页面，param 为 selectByHitting()获取到的坐标值等参数，是在核心脚本库 IMSMaps.js 中生成的。各种编辑相应的名值对参数如下：

添加点：centerX="+X+"&centerY="+Y+"&radius="+dx

添加线：XArray="+Xarray+"&YArray="+Yarray

添加区：XArray="+Xarray+"&YArray="+Yarray

- **renderSelectResult()代码：**

```
function renderSelectResult(url){
        //设置进度条为繁忙状态
    var obj = document.getElementById("rsFrame");
    if(obj==null)
    {
        document.getElementById("rsPanel").innerHTML='<iframe id="rsFrame" style="BORDER:
0px; WIDTH: 100%; HEIGHT: 100%" src="" frameBorder="0" scrolling="auto"
onload="loadBar(0)"></iframe>';
        obj = document.getElementById("rsFrame");
    }
```

```
obj.src = url;
switchdiv(1);
}
```

该函数用于将处理结果显示到一个叫做 rsFrame 的 iframe 中，用户可以根据自己的需要修改。

6. 服务器后台页面 AddFeature.aspx，用于处理各种编辑。

AddFeature.aspx 页面客户端的编辑面板效果如下图：

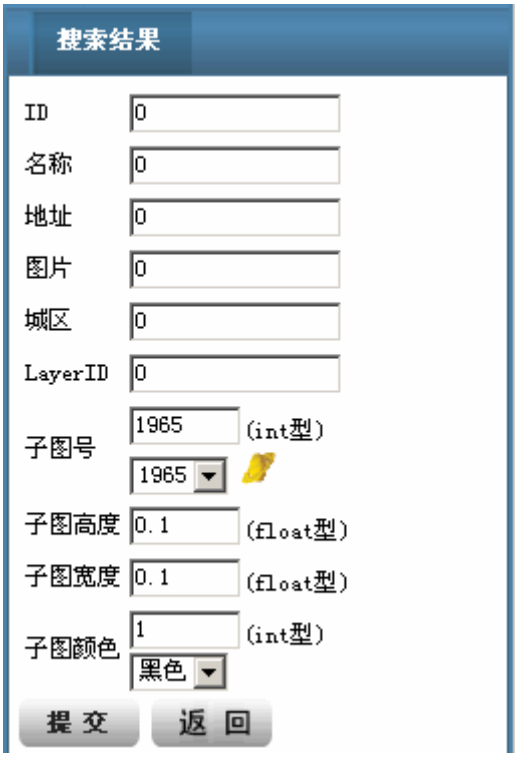



图 5-2-9-2 地图编辑面板

➢ **以添加点要素为例，AddFeature.aspx 页面的客户端代码：**

```
<tr>
  <td><asp:Label ID="Label1" runat='server'>子图号</asp:Label></td>
  <td><input id="text1" name="para" type="text" value="1965" style="width:50px" />(int型)<br/>
    <select id="dotImgNum" name="dotImgNum" onchange="document.getElementById('dotImg').src='../images/imgLib/'+this.value+'.bmp';document.getElementById('text1').value=this.value">
      <option value="1965" selected="selected">1965</option>
      <option value="1966">1966</option>
      <option value="1967">1967</option>
      <option value="1969">1969</option>
      <option value="1970">1970</option>
      <option value="1971">1971</option>
      <option value="1972">1972</option>
      <option value="1973">1973</option>
      <option value="1974">1974</option>
```

```
                <option value="1975">1975</option>
        </select>
        <img alt="子图" width="20" height="20" id="dotImg" src="../images/imgLib/1965.bmp" />
       </td>
    </tr>
    <tr>
       <td><asp:Label ID="Label2" runat='server'>子图高度</asp:Label></td>
       <td><input id="text2" name="para" type="text" value="0.1" style="width:50px"
/>(float型)</td>
    </tr>
    <tr>
       <td><asp:Label ID="Label3" runat='server'>子图宽度</asp:Label></td>
       <td><input id="text3" name="para" type="text" value="0.1" style="width:50px"
/>(float型)</td>
    </tr>
    <tr>
       <td><asp:Label ID="Label4" runat='server'>子图颜色</asp:Label></td>
       <td><input id="text4" name="para" type="text" value="1" style="width:50px" />(int
型)<br/>
           <select id="doImgColor" name="doImgColor"
onchange="document.getElementById('text4').value=this.value">
                <option value="1" selected="selected" style="color:Black;">黑色
</option>
                <option value="175"  style="color:Red;">红色</option>
                <option value="4"  style="color:Yellow;">黄色</option>
                <option value="7"  style="color:Green;">绿色</option>
                <option value="5"  style="color:Blue;">蓝色</option>
                <option value="3" style="color:Purple;">紫色</option>
           </select>
       </td>
     </tr>
```

这是添加线的客户端代码，这个编辑面板位于右侧 ID 为 rsFrame 的子框架里。

- **AddFeature.aspx 后台页面编辑处理代码：**

```
protected void Btn_Insert_Click(object sender, EventArgs e)
{
    string[] parameters = Request.Form["fieldValue"].Trim().Split(',');
     string[] otherPara = Request.Form["para"].Trim().Split(',');
```

```
验证参数类型的合法性
bool AppendFlg = false;
switch (op)
{
    case "0":  //点工作区
        添加点
    case "1":  //线工作区
        添加线
    case "2": //区工作区
        添加区
    default:
        break;
}
if (AppendFlg == true)
{
    Response.Write("<script language=javascript>alert('添加要素操作成功!
');top.map.setCurOper(top.IMSOperType.Refresh);");
    Response.Write("top.document.getElementById('rsFrame').src
='';top.switchdiv(0);");
    //Response.Write("top.myViewStateObj.refresh(top.mapFrame.mapObj);");
    Response.Write("</script>");
}
else
{
    Response.Write("<script language=javascript>alert('添加要素操作失败!
')</script>");
    return;
}
}
```

这个页面接收客户端参数，验证客户端所填属性的参数类型合法性，然后根据编辑类型（op）来进行各项编辑的处理。

- **添加点要素的服务器端处理代码：**

```
#region 添加点
        //接收AppendElimDialog客户端传来的鼠标点击点的逻辑坐标
        OpCPoint2D dotLog = new OpCPoint2D();
        //为添加点函数输入参数赋值
        OpCAppendAPoint thePoint = new OpCAppendAPoint();
        //点要添加到的图层索引号
        thePoint.LayerIndex = selectlayerIndex;
        //要添加的点属性字段长度
        thePoint.PropertyNumber = layerAttStruct.FieldNumber;
        //要添加的点属性字段信息
        thePoint.PropertyValues = parameters;
        //要添加的点的逻辑坐标
        dotLog.X = Convert.ToDouble(Request.QueryString["centerX"]);
        dotLog.Y = Convert.ToDouble(Request.QueryString["centerY"]);
        thePoint.ThePoint = dotLog;
```

```
                try
                {
                    thePoint.PicType = Convert.ToInt32(otherPara[0]);
                    thePoint.PicHeight = Convert.ToSingle(otherPara[1]);
                    thePoint.PicWidth = Convert.ToSingle(otherPara[2]);
                    thePoint.PicColor = Convert.ToInt32(otherPara[3]);
                }
                catch
                {
                    Response.Write("<script language=javascript>alert('传入参数不正确，请检查！')</script>");
                    return;
                }
                //添加点函数
                try
                {
                    AppendFlg = viewport.AppendOnePoint(thePoint);
                }
                catch (Exception ex)
                {
                    string str = ex.Message;
                    AppendFlg = false;
                }
                break;
            #endregion
```

以上代码先接收点要素的属性信息和坐标值等参数，然后调用 GIS 服务端组件把点要素添加到地图文档中，并返回结果。

7. 以添加点要素为例，客户端具体操作如图：

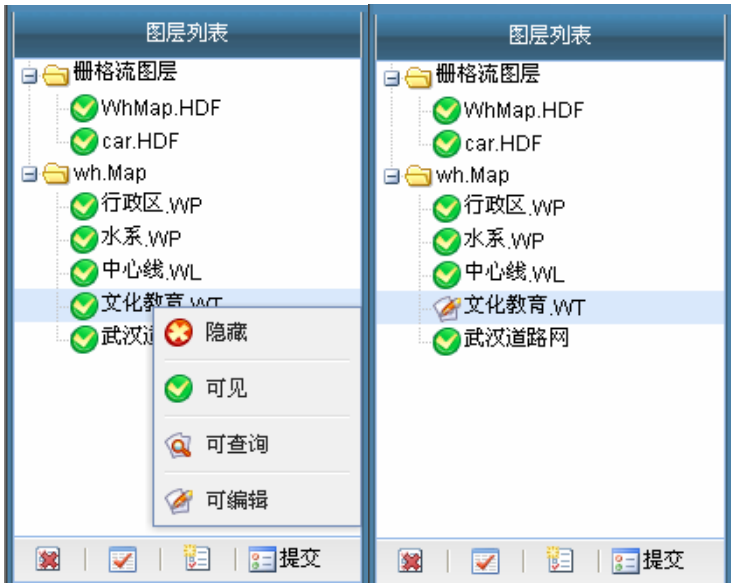



图 5-2-9-3 步骤 1-设置点文件为编辑状态并提交

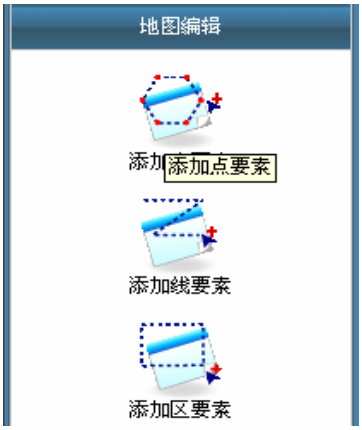



图 5-2-9-4 步骤 2-选择添加点要素按钮并单击

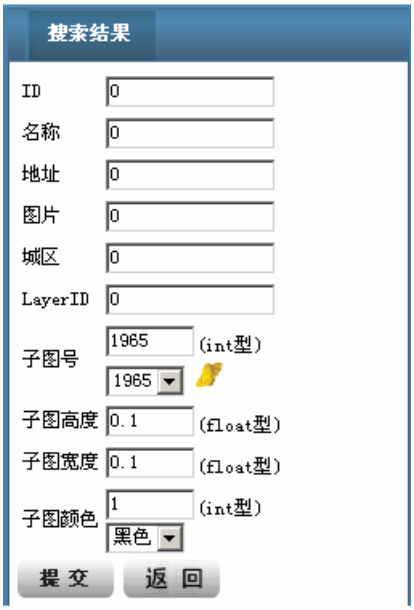



图 5-2-9-5 步骤 3-在地图上单击，右侧显示编辑面板

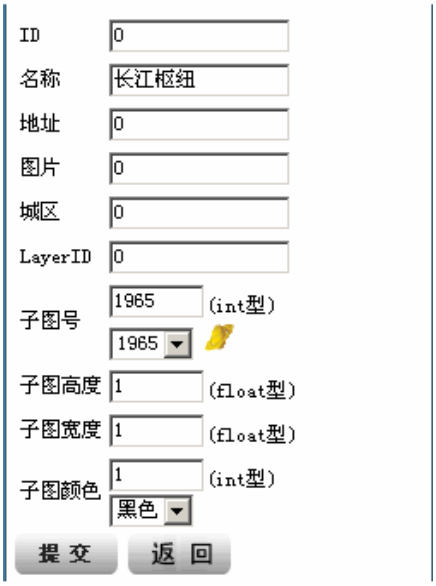



图 5-2-9-6 步骤 4-为添加的点要素设置属性




图 5-2-9-7 步骤 5-单击提交按钮，在地图上显示添加的点要素

## 5.2.10 空间分析功能

地图文档中实现对各要素进行路径分析、拓扑分析、各种叠加分析和裁剪分析等功能。客户端空间分析各功能按钮显示效果如下图：




图 5-2-10-1 空间分析各功能按钮

下面分别介绍空间分析的各个功能实现流程。

1. **叠加分析**，实现对要素类或简单要素类中两个图层的相交、相减、合并及对称差几种叠加分析功能。具体实现流程为：

（1）在客户端（MapFrame.aspx、menu.htm 页面）添加叠加分析的功能按钮，进行事件编写及绑定。

➢ **叠加分析功能按钮的代码：**

```
<img class="btncursor" onclick="top.onSelectToolClick('OverLayAnalyse',null)" alt="叠加分析" src="images/overlay.gif" align="middle" border="0"><br>
 叠加分析<br>
```

（2）在叠加分析功能按钮的 onclick 事件中调用 onSelectToolClick 函数进行处理，该函数位于脚本库 function.js 中。

➢ **onSelectToolClick()叠加分析部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
        case "OverLayAnalyse":
            var url = "VectorPage/OverlayAnalyse.aspx?rnd="+Math.random();
            renderSelectResult(url);
        break;
   }
}
```

该函数根据 userActive 参数的值是否等于 OverLayAnalyse，判断是否进行叠加分析，如果 userActive 参数的值=OverLayAnalyse，则转到 VectorPage 文件夹下的 OverlayAnalyse.aspx 页面进行叠加分析处理，返回分析结果。用 renderSelectResult 函数将叠加分析处理结果显示到一个叫做 rsFrame 的 iframe 中，用户可以根据自己的需要修改。

（3）服务器后台页面 OverlayAnalyse.aspx，用于处理叠加分析功能中的各种叠加功能。

➢ **OverlayAnalyse.aspx 页面的客户端代码：**

```
<form id="form1" runat="server">
    <div id="gdbNameDiv">
        数据源:<asp:DropDownList ID="gdbSourceList" runat="server" Width="112px" OnSelectedIndexChanged="gdbSource_SelectedIndexChanged" AutoPostBack="True">
        </asp:DropDownList><br />
        登陆名:<asp:TextBox ID="tb_gdbSvrUser" runat="server" Width="106px"></asp:TextBox><br />
        密  码:<asp:TextBox ID="tb_gdbSvrPwds" TextMode="Password" runat="server" Width="106px"></asp:TextBox><br />
        <asp:Button ID="btn_connect" runat="server" Text="连接数据库" OnClick="btn_connect_Click" />
        <p/>
```

```
        要素类型：<asp:DropDownList ID="layerType" runat="server" Width="112px">
            <asp:ListItem Value="3">要素类</asp:ListItem>
            <asp:ListItem Value="30">简单要素类</asp:ListItem>
        </asp:DropDownList><br />
        GDB 名:<asp:DropDownList ID="gdbNameList" runat="server" Width="112px"
OnSelectedIndexChanged="gdbNameList_SelectedIndexChanged" AutoPostBack="True">
        </asp:DropDownList>
        <br />
    </div>
    <br />
    <div>
         被叠加图层列表<br />
        <asp:ListBox ID="layerNames1" runat="server" Rows="8"
Width="80%"></asp:ListBox><br />
        <br />
         叠加图层列表<br />
        <asp:ListBox ID="layerNames2" runat="server" Rows="8"
Width="80%"></asp:ListBox><br />
        <br />
        叠加类型:<asp:DropDownList ID="overlayType" runat="server" Width="112px">
            <asp:ListItem Value="0">合并</asp:ListItem>
            <asp:ListItem Value="1">相交</asp:ListItem>
            <asp:ListItem Value="6">对称差</asp:ListItem>
            <asp:ListItem Value="2">相减</asp:ListItem>
        </asp:DropDownList><span onclick="showTip()" title="查看操作提示" style="
color:Green; cursor:help;">  提示</span><br />
        平差半径:<asp:TextBox ID="radius" runat="server"
Width="106px">0.0001</asp:TextBox><br />
        结果图名:<asp:TextBox ID="resultName" runat="server"
Width="106px">resultLayer0</asp:TextBox> </div><br />
        <asp:ImageButton ID="Btn_Submit" runat="server" OnClick="Btn_Submit_Click"
ImageUrl="../images/btn_analyze.gif" />
          <img alt="取消" style=" cursor:pointer;"
src="../images/btn_cancel.gif" onclick="Close()" />
        <br />
        <asp:TextBox ID="errorTextbox" runat="server"></asp:TextBox>
        <input id="fileName" type="hidden" name="fileName" runat="server" value="-1"/>
     <input id="flg" type="hidden" name="flg" runat="server" value="-1"/>
    </form>
```

在客户端连接叠加分析的数据源，选择图层和叠加类型等属性，点击分析按钮，触发事件 Btn_Submit_Click 在后台进行处理。

- **OverlayAnalyse.aspx 后台页面叠加分析部分代码：**

```
protected void Btn_Submit_Click(object sender, EventArgs e)
```

```
    {
        string layer1Name = this.layerNames1.SelectedValue;
        string layer2Name = this.layerNames2.SelectedValue;
        gdbName = this.gdbNameList.SelectedValue;
        if (gdbName == "")
        {
            Response.Write("<script language=javascript>alert('请选择数据库
')</script>");
            return;
        }
        if(layer1Name==""||layer2Name=="")
        {
            Response.Write("<script language=javascript>alert('图层名为空')</script>");
            return;
        }
        string overlayType = this.overlayType.SelectedValue;
        double r;
        try
        {
            r = Convert.ToDouble(this.radius.Text);
        }
        catch
        {
            this.errorTextbox.Visible=true;
            this.errorTextbox.Text = "结点平差半径不能为空值，且为double型";
            return;
        }
        string overlayResultName = this.resultName.Text;
        //声明叠加分析对象
        OpCSpatialOverlay overlay = new OpCSpatialOverlay();
        OpCSpatialOperResult SpatialOperRlt = null;
        //为叠加分析的两个图层赋值
        overlay.LayerIndex1 = overlay.LayerIndex2 = 0;
        overlay.GDBDataSourceName1 = this.gdbSourceList.SelectedValue;
        overlay.GDBName1 = gdbName;//gdbName.Contains(".HDF") ? gdbName : gdbName + ".HDF";
        overlay.GDBDataSourceName2 = this.gdbSourceList.SelectedValue;
        overlay.GDBName2 = gdbName;
        overlay.LayerType = (LayerTypeEnum)Convert.ToInt32(Request["layerType"]);
        overlay.FeatureIDSLst2 = "";
        overlay.FeatureIDSLst1 = layer1Name + "," + layer2Name;
        //叠加分析的类型，如并、交、差、减等
        overlay.OverlayType = Convert.ToInt16(overlayType);
        //叠加分析后生成的文件名
        Random rnd = new Random();
```

```
            overlayResultName += overlayResultName + rnd.Next(10000).ToString();
            overlay.OverlayGDBName = overlayResultName;
            overlay.userName1 = this.tb_gdbSvrUser.Text;
            if(Session["GdbSvrPwds"] != null)
                overlay.password1 = (string)Session["GdbSvrPwds"];
            overlay.userName2 = this.tb_gdbSvrUser.Text;
            if (Session["GdbSvrPwds"] != null)
                overlay.password2 = (string)Session["GdbSvrPwds"];
            overlay.SetNearDis(r);
            //叠加分析的返回结果对象
            try
            {
                SpatialOperRlt = viewport.SpatialAnalysisOverlay(overlay);
            }
            catch
            {
                flg.Value = "-1";
                Response.Write("<script language=javascript>alert('叠加分析失败!
')</script>");
                return;
            }
            //通过结果对象取叠加分析最后文件所在URL地址
            fileName.Value = viewport.ZipFileName;
            flg.Value = "1";
        }
```

通过服务器端控件取得叠加分析的图层，并取得叠加类型和平差半径。最后在后台页面进行叠加分析，生成供下载的结果。

2. **拓扑分析**，分析两个图层要素在拓扑变化下其不变的性质。

（1）在客户端（MapFrame.aspx、menu.htm 页面）添加拓扑分析的功能按钮，进行事件编写及绑定。

- **拓扑分析功能按钮的代码：**

```
<img class="btncursor" onclick="top.onSelectToolClick('TopoAnalyse',null)" alt="拓扑分析"
src="images/editable.gif" align="middle" border="0"><br>
    拓扑分析<br>
```

（2）在拓扑分析功能按钮的 onclick 事件调用 onSelectToolClick 函数进行处理，该函数位于脚本库 function.js 中

- **onSelectToolClick()拓扑分析部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
```

```
    {
      case "TopoAnalyse":
            var url = "VectorPage/TopoAnalyse.aspx?rnd="+Math.random();
            renderSelectResult(url);
        break;
  }
}
```

上面的函数就是将拓扑分析和设置拓扑分析的页面 TopoAnalyse.aspx 关联。即转到 TopoAnalyse.aspx 页面进行拓扑分析设置，用 renderSelectResult()将拓扑分析处理结果设置显示到一个叫做 rsFrame 的 iframe 中。

（3）在 TopoAnalyse.aspx 页面中，在客户端选择两个要进行分析的要素，然后通过分析按钮触发点击事件的 startAnalyse()在后台进行处理。

TopoAnalyse.aspx 页面客户端的拓扑分析设置面板如下图：

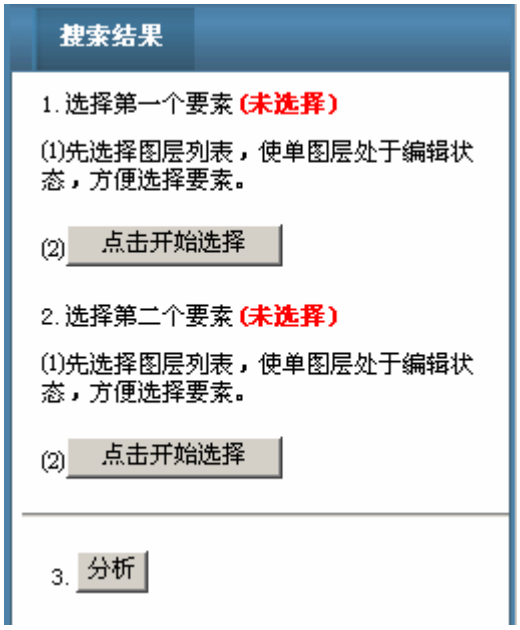



图 5-2-10-2 拓扑分析设置面板

- **TopoAnalyse.aspx 页面客户端代码：**

```
<form id="topoForm" runat="server">
    <div id="f1">
        <div>1.选择第一个要素<span id="Label1" style="color:Red;font-weight:bolder">(未选择)</span></div>
        <div>
        (1)先选择图层列表，使单图层处于编辑状态，方便选择要素。<br/><br/>
        (2)<input type="button" value="点击开始选择"  onclick="beginSelect(1)" />
        </div>
    </div>
    <p />
    <div id="f2">
        <div >2.选择第二个要素<span id="Label2" style="color:Red;font-weight:bolder">(未选择)</span></div>
```

```
        <div>
        (1)先选择图层列表，使单图层处于编辑状态，方便选择要素。<br/><br/>
        (2)<input type="button" value="点击开始选择"  onclick="beginSelect(2)" />
        </div>
    </div>
    <hr />
    <div style="padding-left:10px">
        3.<asp:Button ID="submit" runat="server" Text="分析" onclick="submit_Click" />
    </div>
     <div>
    <asp:Label ID="msg" runat="server" Text="" Font-Bold="true" ></asp:Label>
    </div>
    </form>
```

- **submit_Click ()后台处理代码：**

```
protected void submit_Click(object sender, EventArgs e)
    {
        CMapView viewport = (CMapView)Session["viewport"];
        OpCSpatialDiffer sd = (OpCSpatialDiffer)Session["Differ"];
        if (sd == null)
        {
            msg.Text = "";
            return;
        }
        TopoTypeEnum urlt = viewport.SpatialDiffer(sd);
        string result = "";
        switch (urlt)
        {
            case TopoTypeEnum.Adjacent:
                result = "相邻(" + urlt.ToString() + ")";
                break;
            case TopoTypeEnum.Disjoin:
                result = "相离(" + urlt.ToString() + ")";
                break;
            case TopoTypeEnum.Include:
                result = "包含(" + urlt.ToString() + ")";
                break;
            case TopoTypeEnum.Intersect:
                result = "相交(" + urlt.ToString() + ")";
                break;
            case TopoTypeEnum.None:
                result = "未知类型(" + urlt.ToString() + ")";
                break;
            default:
```

```
                result = "未知类型(" + urlt.ToString() + ")";
                break;
        }
        msg.Text = "两要素的拓扑关系为：<font color=red>" + result+"</font>";
        Session.Remove("Differ");
        }
```

在该页面中，调用 GIS 服务器组件中的 spatialDiffer 方法进行拓扑分析，并返回分析结果。

3. **裁剪分析**：实现圆裁剪和多边形裁剪功能。具体实现流程如下：

（1）在客户端（MapFrame.aspx、menu.htm 页面）添加裁剪分析的功能按钮，进行事件编写及绑定。

➢ **裁剪分析功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('ClipByCircle','Circle','ClipByCircleDialog.aspx')" alt="
圆裁剪" src="images/circlip.gif" align="middle" border="0"><br>
      圆裁剪<br>
      <br>
<img class="btncursor"
onclick="top.onSelectToolClick('ClipByPolygon','Polygon','ClipByPolygonDialog.aspx')"
alt="多边形裁剪" src="images/polyclip.gif" align="middle" border="0"><br>
      多边形裁剪<br>
      <br>
```

（2）在裁剪分析功能按钮的 onclick 事件调用 onSelectToolClick 函数进行处理，该函数位于脚本库 function.js 中。

➢ **onSelectToolClick()裁剪分析部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
    case "ClipByCircle":
        map.setCurOper(2003);
        pro.setSelectByCircle(submitUrl);
    break;
    case "ClipByPolygon":
        map.setCurOper(2004);
        pro.setSelectByPolygon(submitUrl);
        break;
   }
}
```

上面的函数就是将各种裁剪分析和设置裁剪分析参数的页面关联。activeTool 参

数：代表裁剪的类型； drawAction 参数：代表客户端鼠标的操作类型； TargetPage 参数：代表设置裁剪分析参数页面。

（3） 在 onSelectToolClick 函数中首先根据参数调用 setCurOper 函数设置地图操作状态为相应类型，再在 setCurOper 函数中调用 setQueryMode 函数，根据 drawAction 参数（客户端鼠标在地图上的操作类型）添加事件监听并调用相应的脚本处理函数，在处理函数中获取相应的参数 param（为 submitUrl 函数的参数）。

- **setCurOper()裁剪分析部分代码：**

```
this.setCurOper=function(operType)
    {
        this.operType = parseInt(operType);
        this.clearOper();
        switch(this.operType)
        {
            case 2003:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByCircle","Circle");
            break;
            case 2004:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByPolygon","Area");
            break;
          }
    };
```

- **setQueryMode()部分代码：**

```
this.setQueryMode = function(activeTool,drawAction)
{
     thisObj.activeTool = activeTool;
     thisObj.drawAction = drawAction;
     switch(drawAction){
        case "Circle" :
          if(this.listenerCircle){
            thisObj.addEventListener("mouseup",thisObj.so.selectByCircle, false);
            this.listenerCircle = false;
          }
          break ;
        case "Area" :
          if(this.listenerPolygon){
            thisObj.addEventListener("dblclick",thisObj.so.selectByPolygon, false);
              this.listenerPolygon = false;
            }
            break ;
```

```
            default:
                break ;
        }
    };
```

➢ **以圆裁剪为例，setSelectByCircle()代码：**

```
    this.setSelectByCircle = function(dealFun)
    {
        this.dealFunArr[2] = dealFun;
    };
```

➢ **以圆裁减为例，selectByCircle()代码为：**

```
this.selectByCircle= function()
{
    if(thisObj.activeTool!="SelectByCircle"){
        thisObj.removeEventListener("mouseup",thisObj.so.selectByCircle , false) ;
        thisObj.so.listenerCircle =true;
        return ;
    }
    var centerX=thisObj.mdLogicX ;
    var centerY=thisObj.mdLogicY ;
    var upX = thisObj.muLogicX ;
    var upY = thisObj.muLogicY ;
    var w = Math.abs(centerX - upX);
    var h = Math.abs(centerY - upY);
    var radius = Math.sqrt(w*w + h*h);
    var startX=centerX-radius;
    var startY=centerY-radius;
    var endX=centerX+radius;
    var endY=centerY+radius;
    var
param="centerX="+centerX+"&centerY="+centerY+"&radius="+radius+"&GridCodes="+thisObj.getGrid
Codes(startX,startY,endX,endY)+"&lev="+(thisObj.inf[2]+1)+"&sType=SelectByCircle";
    if(thisObj.so.dealFunArr[2]!=null)
        thisObj.so.dealFunArr[2](param);
    };
```

通过鼠标事件获取点击的 x、y 坐标值等参数值，调用 dealFunArr[2]，即 submitUrl 函数，把参数组 param 设置为 submitUrl 函数的参数。

（4）自定义函数中的 submitUrl 参数是一个对象，它就是对各种裁剪分析的处理方法。由 submitUrl()处理将裁剪参数传到相应的设置裁剪分析参数的页面，设置好参数后再转到后台页面 ClipByCircle.aspx 或 ClipByPolygon.aspx 进行处理，生成裁剪结果。由 renderSelectResult 函数将结果设置显示到一个叫做 rsFrame 的 iframe 中，用户可以根据自己的需要修改。

➢ **submitUrl()代码：**

```
function submitUrl(param)
{
    var url = "VectorPage/"+userParam+"?" + param+"&op="+userActive;
    subParam = param;
    loadBar(1);
    renderSelectResult(url);
}
```

该函数中 userParam 的值是从客户端传过来的设置裁剪分析参数的页面，即 ClipByCircleDialog.aspx 或 ClipByPolygonDialog.aspx 页面;param 为鼠标事件获取的坐标值等参数，是在核心脚本库 IMSMaps.js 中生成的。各种裁剪相应的名值对参数如下：

圆裁剪：centerX="+centerX+"&centerY="+centerY+"&radius="+radius

多边形裁剪：XArray="+Xarray+"&YArray="+Yarray

（5）服务器页面 ClipByCircleDialog.aspx 或 ClipByPolygonDialog.aspx，分别用于设置圆裁剪和多边形裁剪的裁剪容差半径。

裁剪设置面板如下图：




图 5-2-10-3 裁剪设置面板

➢ **以圆裁剪为例，ClipByCircleDialog.aspx 页面设置参数代码：**

```
裁剪容差半径：<input type="text" id="nearDis" value="0.0001"/> <br /><br />
    <img type="button" id="btn_clip" value="圆裁剪" onclick="clip()"
src="../images/btn_submit.gif" />
    <img id="close" type="button" value="关闭" onclick="Close()"
src="../images/btn_close.gif" />
```

根据值为nearDis的id，在核心脚本库中根据当前的像素对应的逻辑长度生成裁剪容差半径的范围。

➢ **clip()代码：**

```
function clip()
        {
            var dis = document.getElementById("nearDis").value;
            var lx = document.getElementById("lx").value;
            var ly = document.getElementById("ly").value;
            var r = document.getElementById("r").value;
            document.getElementById("clipCharge").src =
"ClipByCircle.aspx?nearDis="+dis+"&LogicX="+lx+"&LogicY="+ly+"&Radius="+r+"&rnd="+Math.r
andom();
        }
```

通过点击事件，clip()将参数传到服务器后台页面 ClipByCircle.aspx 中，对圆裁剪进

行处理。

（6）服务器后台页面 ClipByCircle.aspx 或 ClipByPolygon.aspx，分别用于处理圆裁剪和多边形裁剪。

➢ **以圆裁剪为例，ClipByCircle.aspx 页面圆裁剪部分代码：**

```
viewport = (CMapView)Session["viewport"];
  OpCSpatialOperResult spatialOperRlt = null;
  //接收客户端传来的圆心、半径及裁剪半径
  OpCPoint2D dotCenter = new OpCPoint2D();
  dotCenter.X = xmin;
  dotCenter.Y = ymin;
  circleClip.Center = dotCenter;
  circleClip.Radius = radius;
  circleClip.LayerIndexLst = layerStr;
  circleClip.LayerIndex = Convert.ToInt64(layerStr);
  circleClip.SetNearDis(dis);
  //裁剪类型
  Random rand = new Random();
  circleClip.ClipType = 1;
  circleClip.LayerType = ms.GetLayerType(Convert.ToInt32(layerStr));
  circleClip.FeatureClassName = "CircleClip" + Convert.ToString(rand.NextDouble() *
1000);
  try
  {
      spatialOperRlt = viewport.SpatialAnalysisCircleClip(circleClip);
  }
  catch
  {
      Response.Write("<script language=javascript>alert('圆裁剪失败!
');window.close();</script>");
      flg.Value = "-1";
      return;
  }
  fileName.Value = viewport.ZipFileName;
  flg.Value = "1";
```

在 ClipByCircle.aspx 后台页面中根据客户端生成的圆心，半径和裁剪容差半径生成裁剪的结果，然后通过前台脚本将结果传到 AnalyRlt.aspx 页面进行显示。

4. **路径分析**，通过在地图上设置路径结点和障碍点，实现最佳的路线的分析功能。具体实现流程如下：

（1）在客户端（MapFrame.aspx、menu.htm 页面）添加路径分析的功能按钮，进行事件编写及绑定。

➢ **路径分析功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('NetAnalyse','MultiPoints','NetAnalyseDialog.aspx')"
alt="路径分析" src="images/pathAnalyze.gif" align="middle" border="0"><br>
```

```
路径分析<br>
```

（2）路径分析功能按钮的 onclick 事件触发 onSelectToolClick 脚本函数进行处理，该函数位于脚本库 function.js 中。在 onSelectToolClick()中，通过 renderSelectResult 脚本函数将分析结果显示到一个叫做 rsFrame 的 iframe 中，用户可以根据自己的需要修改。

- **onSelectToolClick()路径分析部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
      case "NetAnalyse":
            var url = "VectorPage/"+userParam;
            renderSelectResult(url);
      break;
   }
}
```

上面的函数将路径分析功能和设置路径分析参数的页面 NetAnalyseDialog.aspx 关联。参数 TargetPage 代表设置路径分析参数页面。

（3）服务器页面 NetAnalyseDialog.aspx，用于设置路径分析参数。

NetAnalyseDialog.aspx 页面客户端的路径分析设置面板如下图：




图 5-2-10-4 路径分析设置面板

- **NetAnalyseDialog.aspx 页面设置参数的客户端代码：**

```
<div>
   <input title="点击选择进入路径结点设置状态" type="radio" name="radio1"
onclick="changePntImg('access')" />设置路径结点
    <input title="点击选择进入路径障碍点设置状态" type="radio" name="radio1"
onclick="changePntImg('barrier')" />设置障碍点
    <img type="button" value="提 交" onclick="submitPoints()"
src="../images/btn_submit.gif"/>
    <img type="button" value="闪 烁" onclick="flashPath()"
src="../images/btn_flash.png"/>
    <img id="close" type="button" value="关闭" onclick="Close()"
src="../images/btn_close.gif" />
    </p>
</div>
```

- **设置结点的脚本代码：**

```
function changePntImg(pntStyle)
    {
        if(!listenerFlag)
        {
            mapObj.drawAction = "MultiPoints";
            mapObj.activeTool = "MultiPoints";
            mapObj.addEventListener("mouseup", getPntCoor, false);
            listenerFlag = true;
        }
        mapObj.drawAction = "MultiPoints";
        mapObj.activeTool = "MultiPoints";
        pntType = pntStyle;
    }
   function getPntCoor()
    {
        if(mapObj.drawAction!="MultiPoints"){
            mapObj.removeEventListener("mouseup", getPntCoor, false);
            listenerFlag = false;
            return;
        }
        pointImg = (pntType=="access")?"images/star.gif":"images/delete.gif";
        var coorPnt = mapObj.getClickCoor();
        var screenPnt = mapObj.logicToScreen(coorPnt[0],coorPnt[1]);
        var wx=screenPnt[0]-6+"px";
        var wy=screenPnt[1]-6+"px";
        AnalysePntDiv.innerHTML += "<div
style='visibility:visible;position:absolute;left:" + wx + ";top:" + wy +"'><img src
="+pointImg+"></div>";
        AnalysePntDiv.style.visibility = "visible";
        if(pntType=="access")
        accessPntArr += coorPnt[0]+","+coorPnt[1]+",";
        else if(pntType=="barrier")
        barPntArr += coorPnt[0]+","+coorPnt[1]+",";
    }
```

changePntImg 用于向地图注册一个 mouseup 事件的监听，使用 getPntCoor 方法处理。getPntCoor 主要用于获取在地图上设置的路径节点或障碍点的坐标，并在地图上表示出来设置的路径节点或障碍点。

- **提交点坐标串 submitPoints()代码**

```
function submitPoints()
    {
        var accessPntNum=0;
        var barPntNum=0;
```

```
        if(accessPntArr.length>0)
            accessPntNum = (accessPntArr.split(',').length -1)/2;
        if(barPntArr.length>0)
            barPntNum = (barPntArr.split(',').length - 1)/2;
        if(accessPntNum<2)
        {
            alert("请添加路径点,至少两个");
            AnalysePntDiv.innerHTML="";
            return;
        }
        var pntArr =
(accessPntArr+barPntArr).substring(0,(accessPntArr+barPntArr).length-1).split(',');
        AnalysePntDiv.innerHTML="";
        accessPntArr=new Array();
        barPntArr=new Array();
        //下面一行需要修改
        var dxy = new Array();
        dxy[0] = mapObj.logicXlength*50/mapObj.pObj.offsetWidth;
        dxy[1] = mapObj.logicYlength*50/mapObj.pObj.offsetHeight;
        var url = "NetAnalyse.aspx?accessPntNum="+accessPntNum+"&barPntNum="+barPntNum;
        url += "&Points=" + pntArr + "&BufX=" + dxy[0] + "&BufY=" + dxy[1] + "&op=NetAnalyse";
        url += "&rnd=" + Math.random();
    top.document.getElementById("chargeFrame").contentWindow.location.href = url;

    }
```

通过该函数提交点坐标串到网络分析的后台页面 NetAnalyse.aspx 中处理。

（4）网络分析的后台页面 NetAnalyse.aspx，对网络分析进行处理。

- **NetAnalyse.aspx 后台页面部分代码**

```
        strBuilder = new StringBuilder(1024);
        //TODO:获取网文件图层
        MapSetting ms = new MapSetting();
        int layer = ms.GetNetLayerIndex();
        string pointsOfPolyline = Request.QueryString["Points"];
        if (Session["viewport"] == null)
        {
            Response.Write("<script language=javascript>alert('地图文档装载失败或请求服
务超时')</script>");
            return;
        }
        int accessPntNum = Convert.ToInt32( Request.QueryString["accessPntNum"]);
        int barPntNum = Convert.ToInt32(Request.QueryString["barPntNum"]);
        viewport = (CMapView)Session["viewport"];
        double radius = Convert.ToDouble(Request.QueryString["BufX"]);
        string[] pointsCoor = pointsOfPolyline.Split(',');
```

```
            OpCPoint2D[] points2D = new OpCPoint2D[accessPntNum];
            int barPntIndex = 0;
            for (int i = 0; i < accessPntNum; i++)
            {
                points2D[i].X = Convert.ToDouble(pointsCoor[2 * i]);
                points2D[i].Y = Convert.ToDouble(pointsCoor[2 * i + 1]);
                barPntIndex = i+1;
            }
            OpCPoint2D[] barpoints = new OpCPoint2D[barPntNum];
            for (int i = 0; i < barPntNum; i++, barPntIndex++)
            {
                barpoints[i].X = Convert.ToDouble(pointsCoor[2 * barPntIndex]);
                barpoints[i].Y = Convert.ToDouble(pointsCoor[2 * barPntIndex + 1]);
            }
            OpCBasicNetAnalysis net = new OpCBasicNetAnalysis();
            OpCNetAnalysisResult netRlt = null;
            net.NodeNumber = points2D.Length;
            net.barNum = barPntNum;
            net.barPoints = barpoints;
            net.Nodes = points2D;
            net.LayerIndex = layer;
            net.Radius = radius;
            try
            {
                netRlt = viewport.NetAnalysis(net);
            }
            catch(Exception ex)
            {
                Response.Write("<script language=javascript>alert('路径分析失败,请检查网文件
');</script>");
                return;
            }
```

该代码主要是根据路径节点和障碍点，调用服务器端组件，生成网络分析的结果。

5. **缓冲区分析功能：**实现的具体流程见地图查询功能小节的地图编辑最后部分，在这里不再重复介绍。

注：空间分析中各分析功能的客户端具体操作参见行业版的操作手册。

## 5.2.11 统计分析功能

统计分析功能实现以图表形式展示要素属性统计分析的结果。该功能的大致流程为：先在工具栏中触发统计功能，此时会提示在地图上拉框选择统计的要素，拉框完毕后跳转到 DrawVectorChart.aspx 页面，设置统计图的各个参数，设置完毕后再提交到本页，生成统计图。

统计分析效果如下图：

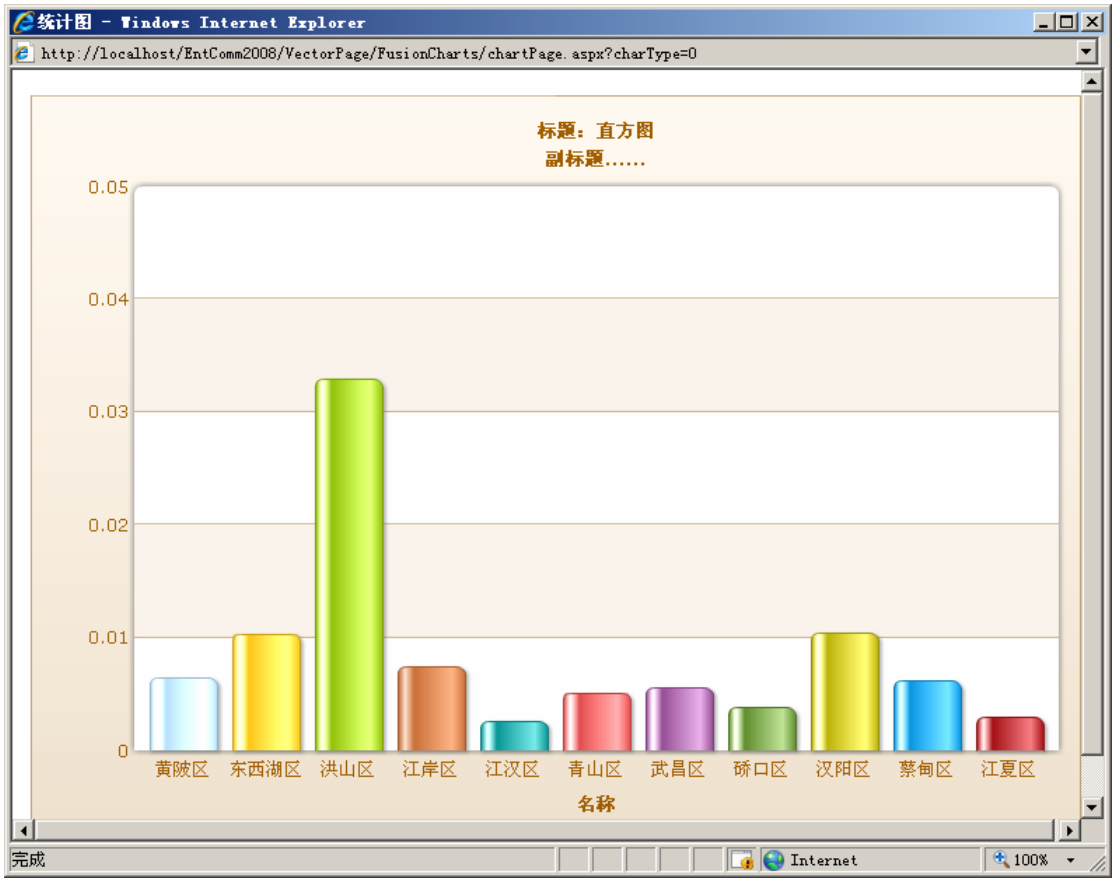



图 5-2-11-1 统计分析直方图

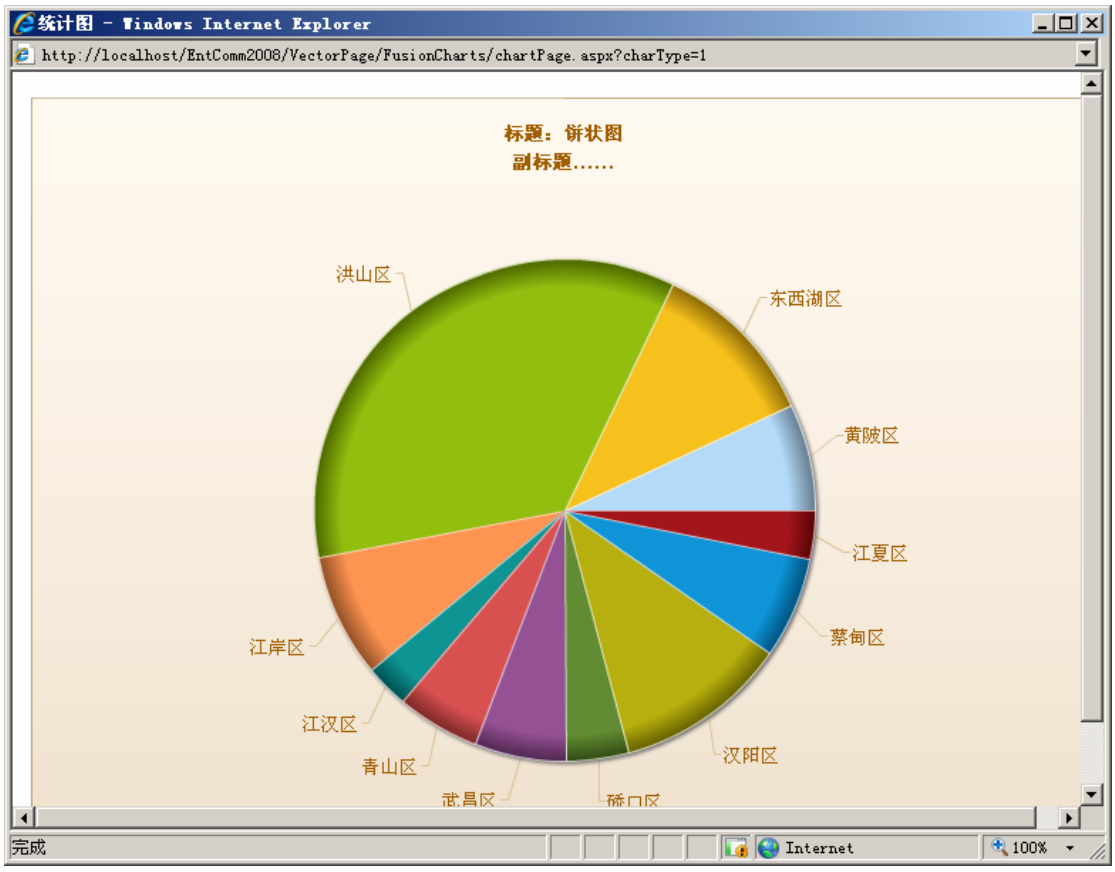



图 5-2-11-1 统计分析饼状图

具体实现流程为：

1. 在客户端 menu.htm 页面添加统计分析的功能按钮，进行事件编写及绑定。

menu.htm 页面客户端统计分析各功能按钮显示效果如下图：




图 5-2-11-3 统计分析各功能按钮

➢ **统计分析功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('Area','Rect','DrawVectorChart.aspx')" alt="区域统计分析"
src="images/chart.gif" align="middle" border="0"><br>
    区域统计分析<br>
    <br>
<img class="btncursor" onclick="top.onDisplayToolClick('queryChart')" alt="属性统计分析"
src="images/chart.gif" align="middle" border="0"><br>
    属性统计分析<br>
    <br>
```

2. **区域统计分析功能：** onclick 事件触发 onSelectToolClick 脚本函数进行处理，该函数位于脚本库 function.js 中。通过一系列的脚本函数处理后将客户端画区的所有参数传到后台页面 DrawVectorChart.aspx 中进行查询，再经过客户端设置统计图的统计字段，显示字段，统计图形状等属性，最后由后台处理生成统计图，并在 chartPage.aspx 页面显示。

➢ **onSelectToolClick()区域统计分析部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
      case "Area":
        map.setCurOper(2002);
        pro.setSelectByRect(submitUrl);
        alert("请在地图上拉框选择统计区域");
        break;
   }
}
```

该函数根据 userActive 参数的值为 Area，判断进行区域统计分析。其参数 TargetPage 的值为 DrawVectorChart.aspx，将区域统计分析功能和服务器后台页面

DrawVectorChart.aspx 关联。

➢ **DrawVectorChart.aspx 页面客户端代码：**

```
<form id="form1" runat="server">
    <div>
        X轴字段:<asp:DropDownList ID="DropDownList1" runat="server">
        </asp:DropDownList><br />
        Y轴字段:<asp:DropDownList ID="DropDownList2" runat="server">
        </asp:DropDownList><br />
        <br />
        统计图形状:<br />
        <asp:RadioButtonList ID="imageShape" runat="server" Width="99px">
            <asp:ListItem Value="0" Selected="True">直方图</asp:ListItem>
            <asp:ListItem Value="1">饼图</asp:ListItem>
        </asp:RadioButtonList><br >
         <asp:imagebutton ID="Btn_Submit" runat="server" OnClick="Btn_Submit_Click" Text="
提交" ImageUrl="../images/btn_stat.gif" />
        <img id="Button1" type="button" value="关闭" runat="server" onclick="Close()"
src="../images/btn_close.gif"/>
         <br />
         <br /><br />
         <asp:Panel ID="PIndex" runat="server" Height="50px" Width="125px">
        </asp:Panel>
    </div>
</form>
```

在客户端设置统计图表的 X 轴和 Y 轴字段，以及统计图表的形状类型。其设置面板效果如下图：

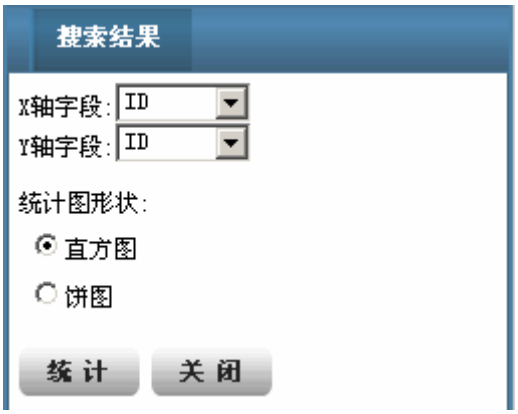



图 5-2-11-4 统计分析客户端设置面板

➢ **DrawVectorChart.aspx 后台页面部分代码：**

```
protected void Btn_Submit_Click(object sender, EventArgs e)
    {
        //创建作图类
        string type = this.imageShape.SelectedValue;
```

```
        if (type == "")
        {
            Response.Write("<script language=javascript>alert('" + "请选择统计图形状" +
"');</script>");
            return;
        }
        int fieldXIndex = this.DropDownList1.SelectedIndex;
        int fieldYIndex = this.DropDownList2.SelectedIndex;
        StringBuilder resSb = new StringBuilder();
        try
        {
            #region 取查询结果
            selRlt = viewport.LastSelectResult;
            layerSelectRlt = selRlt.LayerSelectResults;
            ArrayList[] PropertyValues = layerSelectRlt[0].PropertyValues;
            int selectResultCount = PropertyValues.Length;
            for (int k = 0; k < selectResultCount; k++)
            {
                ArrayList fldValCollection = PropertyValues[k];
                resSb.Append(fldValCollection[fieldXIndex]);
                resSb.Append("," + fldValCollection[fieldYIndex] + "$");
            }
            #endregion
            if (resSb.Length != 0 && resSb[resSb.Length - 1] != '*')
            {
                resSb.Remove(resSb.Length - 1, 1);
            }
            Session["resSb"] = resSb;
            HyperLink[] hlArr = new HyperLink[1];
            hlArr[0] = new HyperLink();
            hlArr[0].NavigateUrl = "javascript:ShowChart(" + type + ")";
            hlArr[0].Text = "点击查看统计结果";
            this.PIndex.Controls.Add(hlArr[0]);
        }
        catch (Exception ex)
        {
            Response.Write("<script language=javascript>alert('" + ex.Message +
"');</script>");
        }
    }
```

点击“统计”按钮，根据客户端设置统计图属性，生成统计图表。可以生成直方图和饼状图两种统计图表。

- **ShowChart()脚本代码：**

```
function ShowChart(type)
{
    var features =
"height=600,width=800,left=10,top=10,resizable=yes,status=no,toolbar=no,menubar=no,locat
ion=no,scrollbars=yes";
    window.open('FusionCharts/chartPage.aspx?charType='+type,'snowy',features);
}
```

在客户端单击“点击查看统计结果”后触发 ShowChart()脚本，转到 FusionCharts 文件夹下的 chartPage.aspx 页面进行处理，把生成的统计图表显示到客户端。

3. **属性统计分析功能：** onclick 事件触发 onDisplayToolClick 脚本函数进行处理，该函数位于脚本库 function.js 中。通过脚本函数转到服务器后台页面 AttributeCon.aspx，根据该页面的客户端设置条件进行查询，然后转到 DrawFlashChart.aspx 页面设置统计图的统计字段，显示字段，统计图形状等属性，最后由后台处理并生成统计图，在 chartPage.aspx 页面中显示。

➢ **onDisplayToolClick()属性统计分析部分代码：**

```
function onDisplayToolClick(con)
{
    var type = con
    switch(type)
    {
        case "queryChart":
            map.setCurOper(1003);
            var url = "VectorPage/AttributeCon.aspx?rnd="+Math.random();
            renderSelectResult(url);
        break;
    }
}
```

该函数根据 type 参数的值为 queryChart，判断该进行属性统计分析，将属性统计分析功能和服务器后台页面 AttributeCon.aspx 关联。

➢ **AttributeCon.aspx 页面客户端代码**

```
<img type="button" name="Btn_ConditionGenerator" value="查询" id="Btn_ConditionGenerator"
onclick="HandIn()" src="../images/btn_submit.gif"/>  
<img name="Button1" type="button" id="Button1" value="关闭
 "onclick="CloseConditionGenerator()" src="../images/btn_close.gif" />
```

该页面与属性条件查询页面类似，客户端为设置查询条件面板，在后台进行条件查询处理，同时通过点击“查询”按钮触发脚本 HandIn（），转到服务器后台页面 DrawFlashChart.aspx 中处理。

➢ **HandIn()脚本代码**

```
function HandIn()
    {
        var CondTmp = document.getElementsByTagName('textarea')[0].value;
```

```
    var url="";
    if(CondTmp == "")
{
     alert("查询条件不可为空！")
     return;
}
url = "VectorPage/DrawFlashChart.aspx?&condition="+CondTmp;
top.document.getElementById("rsFrame").src =url;
}
```

DrawFlashChart.aspx 页面与 DrawVectorChart.aspx 页面功能一样，其对统计图表的客户端属性设置、后台处理以及结果显示等流程大致相同，在这里不再详细介绍。想了解具体流程，参考区域属性统计的后半部分并参见站点中的详细代码。

## 5.2.12 坐标转换功能

坐标转换功能通过在客户端输入待转换的坐标值和设置相关属性，实现转换成目标坐标的功能。

坐标转换功能的效果如下图：

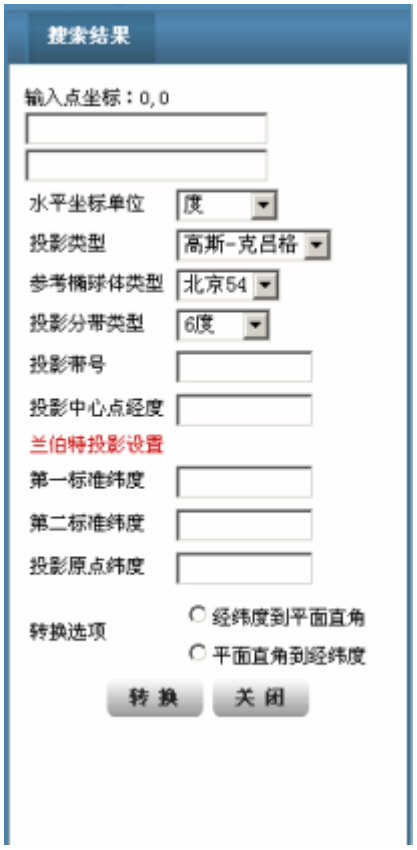



图 5-2-12-1 坐标转换效果

该功能具体实现流程如下：

1. 在客户端 menu.htm 页面添加坐标转换的功能按钮，进行事件编写及绑定。

➢ **坐标转换功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('Projection','Projection','Projection.aspx')"
     alt="坐标转换" src="images/faceStyle.gif" align="middle" border="0"><br>
     坐标转换<br>
     <br>
```

2. 坐标转换功能按钮的 onclick 事件触发 onSelectToolClick 脚本函数进行处理，该函数位于脚本库 function.js 中。

➢ **onSelectToolClick()坐标转换部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
        default:
            var url = "VectorPage/"+TargetPage+"?rnd="+Math.random();
            renderSelectResult(url);
        break;
    }
}
```

该函数的 TargetPage 参数值为 Projection.aspx，表示将转到服务器后台页面 Projection.aspx 中进行坐标转换处理。

3. 服务器后台页面 Projection.aspx，用于处理坐标转换功能操作。

➢ **Projection.aspx 页面客户端部分代码：**

```
<form id="form1" runat="server">
  <div id="inputDotInfo" runat="server"></div>
        x:<asp:TextBox ID="tb_x" runat="server"></asp:TextBox><br />
        y:<asp:TextBox ID="tb_y" runat="server"></asp:TextBox><br />
        <div>
          <table>
             <tr>
               <td style="width: 87px">投影分带类型</td>
               <td>
                        <asp:DropDownList ID="ddl_BZoneType" runat="server">
                            <asp:ListItem Value="0">6度</asp:ListItem>
                            <asp:ListItem Value="1">3度</asp:ListItem>
                            <asp:ListItem Value="2">1.5度</asp:ListItem>
                        </asp:DropDownList></td>
            </tr>
            ……
            <tr>
```

```
                <td style="width: 87px">投影原点纬度</td>
                 <td>
                     <asp:TextBox ID="TextBox3" runat="server"
Width="80px"></asp:TextBox></td>
            </tr>
        </table>
          <div style="text-align:center">
                <asp:imagebutton ID="btn_project" runat="server" Text="转 换"
OnClick="btn_project_Click" ImageUrl="~/images/btn_project.gif" />
                <img id="Button1" type="button" value="关闭" runat="server"
onclick="Close()" src="../images/btn_close.gif"/>
           </div>
  </div>
  <br />
    <div id="proRltDiv" runat="server"></div>
 </form>
```

上述代码用于在客户端输入坐标值，并设置需要转换的相关属性。其坐标转换的设置面板如下图：

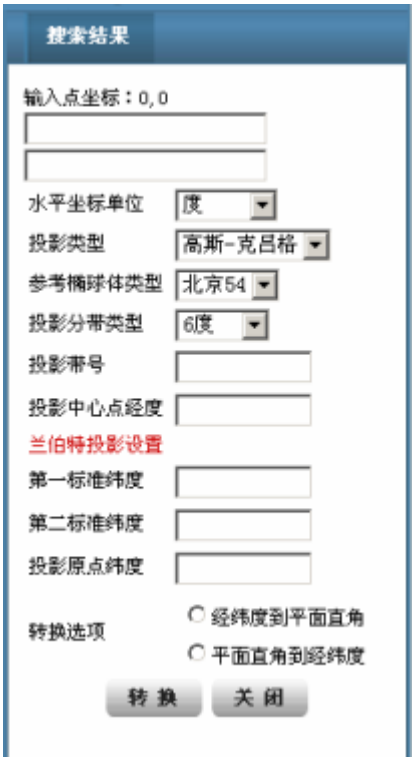



图 5-2-12-2 坐标转换设置面板

- **Projection.aspx 页面后台处理代码：**

```
protected void btn_project_Click(object sender, ImageClickEventArgs e)
    {
        if (Session["viewport"] == null)
        {
            Response.Write("地图文档装载失败或请求服务超时");
```

```
            return;
        }
        if (this.tb_DLon.Text.Equals("") || this.tb_IZone.Text.Equals(""))
        {
            this.proRltDiv.InnerHtml = "\"投影带号\",\"投影中心点经度\"参数不能为空.";
            return;
        }
        viewport = (CMapView)Session["viewport"];
        OpCPrjTran input = new OpCPrjTran();
        input.IPointNum = 1;
        input.ThePoint = new OpCPoint2D[1] { new
OpCPoint2D(Convert.ToDouble(this.tb_x.Text), Convert.ToDouble(this.tb_y.Text)) };
        input.ScaleDenominator = 1;
        input.BZoneType = Convert.ToInt16(this.ddl_BZoneType.SelectedItem.Value);
        input.DAngUnit = Convert.ToInt16(this.ddl_DAngUnit.SelectedItem.Value);
        input.DLon = Convert.ToDouble(this.tb_DLon.Text);//1170000;
        input.IProjTypeId = Convert.ToInt16(this.ddl_IProjTypeId.SelectedItem.Value);
        input.ISpheroid = Convert.ToInt16(this.ddl_ISpheroid.SelectedItem.Value);
        input.IType = Convert.ToInt16(this.ddl_IType.SelectedItem.Value);
        input.IZone = Convert.ToInt16(this.tb_IZone.Text);
        input.Unit = 2;
        input.ConverseFlag = false;
        OpCPrjTranRlt output = new OpCPrjTranRlt();

        try
        {
            output = viewport.PrjTran(input);
        }
        catch
        {
            this.proRltDiv.InnerHtml = "转换失败!";
            return;
        }
        OpCPoint2D[] pnts = output.PrjRltPoint;
        this.proRltDiv.InnerHtml = "转换结果：" + pnts[0].X.ToString() + "," +
pnts[0].Y.ToString();
    }
```

通过在客户端点击“转换”按钮，触发事件 btn_project_Click，调用 GIS 服务器端组件实现坐标转换功能。

## 5.2.13 面积周长计算功能

面积周长计算功能实现在客户端输入坐标值字符串，即可计算出该坐标字符串所形成区域的面积和周长。

面积周长计算功能的效果如下图：

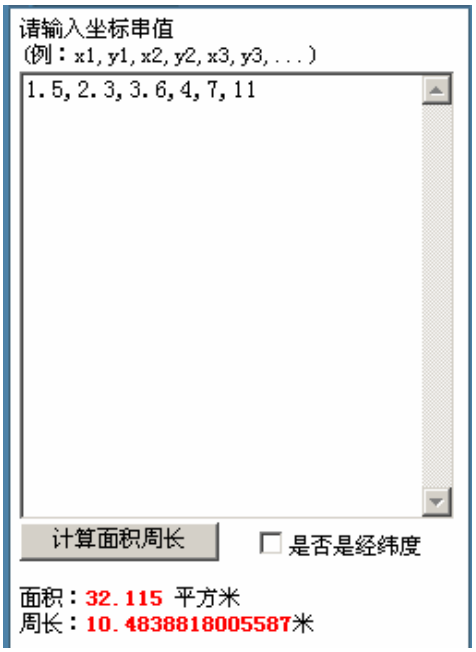



图 5-2-13-1 面积周长计算效果

该功能的具体实现流程如下：

1. 在客户端 menu.htm 页面添加面积周长计算的功能按钮，进行事件编写及绑定。

➢ **面积周长计算功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('CaculateArea','CaculateArea','CaculateArea.aspx')"
    alt="计算面积" src="images/searchable.gif" align="middle"   border="0"><br>
计算面积<br>
```

2. 面积周长计算功能按钮的 onclick 事件触发 onSelectToolClick 脚本函数进行处理，该函数位于脚本库 function.js 中。

➢ **onSelectToolClick()面积周长计算部分代码：**

```
function onSelectToolClick(activeTool,drawAction,TargetPage)
{
    userParam = TargetPage;
    userActive = activeTool;
    switch(userActive)
    {
        default:
            var url = "VectorPage/"+TargetPage+"?rnd="+Math.random();
            renderSelectResult(url);
        break;
    }
}
```

该函数的 TargetPage 参数值为 CaculateArea.aspx，表示将转到服务器后台页面 CaculateArea.aspx 中进行面积周长计算处理。

3. 服务器后台页面 CaculateArea.aspx，用于处理面积周长计算功能操作。

➢ **CaculateArea.aspx 页面客户端代码：**

```
<form id="form1" runat="server">
    <div>
   请输入坐标串值<br/>
   (例：x1,y1,x2,y2,x3,y3,...)
        <asp:TextBox ID="TextBox1" runat="server" TextMode="MultiLine" Width="230px"
Height="237px"></asp:TextBox><br />
        <asp:Button ID="Button1" runat="server" Text="计算面积周长"
OnClick="Button1_Click" />
         
        <asp:CheckBox ID="CheckBox1" runat="server" Text="是否是经纬度" /><br />
        <br />
        面积：<asp:Label ID="Label1" runat="server" Width="163px" Font-Bold="true"
ForeColor="Red"></asp:Label>
        平方米<br />
        周长：<asp:Label ID="Label2" runat="server" Width="174px" Font-Bold="true"
ForeColor="Red"></asp:Label>米</div>
    </form>
```

上述代码用于在客户端输入坐标值字符串和设置相关属性。其面积周长计算的设置面板如下图：

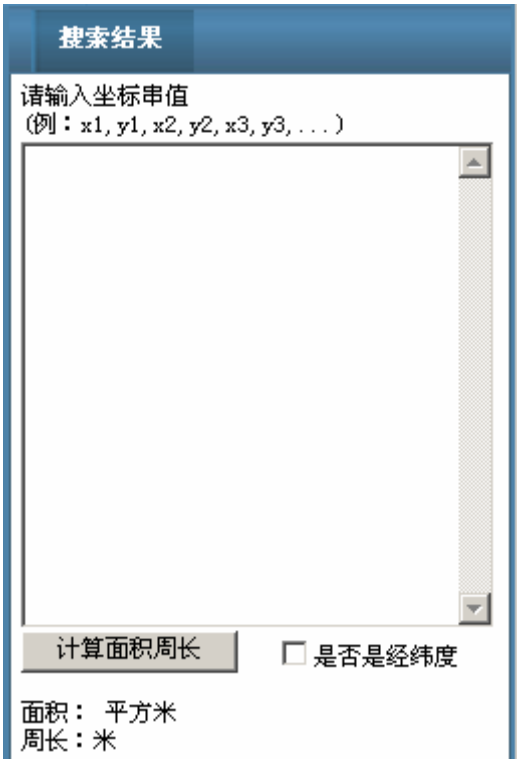



图 5-2-13-2 面积周长计算设置面板

➢ **CaculateArea.aspx 页面后台处理代码：**

```
    private double CaculateTheDistance(double[] xArr, double[] yArr)//计算机周长
    {
```

```
    double total = 0;
    int num = xArr.Length;
    for (int i = 0; i < num - 1; i++)
    {
        double dx = Math.Abs(xArr[i + 1] - xArr[i]);
        double dy = Math.Abs(yArr[i + 1] - yArr[i]);
        double dis = Math.Sqrt(Math.Pow(dx, 2) + Math.Pow(dy, 2));
        total += dis;
    }
    return total;
}
  private double CalculateArea2(double[] xArr, double[] yArr)//计算面积
{
    int len = xArr.Length;
    double t1 = 0;
    int p0 = 0;
    int p1 = 1;

    for (int i = 1; i < len; i++)
    {
        t1 += (yArr[p1] + yArr[p0]) * (xArr[p1] - xArr[p0]);
        p0++;
        p1++;
    }
    double area = t1 / 2.0;
    return Math.Abs(area);
}
protected void Button1_Click(object sender, EventArgs e)
{
    string[] coorArr;
    double[] xArr;
    double[] yArr;
    double area,len;
    string coordirateStr = this.TextBox1.Text;
    bool isJwd = this.CheckBox1.Checked;

    try
    {
        coorArr = coordirateStr.Split(',');
        if (coorArr.Length < 6)
            throw new Exception();
        xArr = new double[coorArr.Length/2];
        yArr = new double[coorArr.Length/2];
        int count = 0;
```

```
for (int i = 0; i < xArr.Length; i++)
{
    xArr[i] = Convert.ToDouble(coorArr[count++]);
    yArr[i] = Convert.ToDouble(coorArr[count++]);
}
if (isJwd)
{
    if (Session["viewport"] == null)
    {
        Response.Write("地图文档装载失败或请求服务超时");
        return;
    }
    CMapView viewport = (CMapView)Session["viewport"];
    OpCPrjTran input = new OpCPrjTran();

    int c = xArr.Length;
    input.IPointNum = (short)c;
    OpCPoint2D[] pnts = new OpCPoint2D[c];
    for (int i = 0; i < c; i++)
    {
        pnts[i] = new OpCPoint2D(xArr[i], yArr[i]);
    }
    input.ThePoint = pnts;
    input.ScaleDenominator = 1;
    input.BZoneType = 1;
    input.DAngUnit = 4;
    input.DLon = 1140000;
    input.IProjTypeId = 5;
    input.ISpheroid = 1;
    input.IType = 3;
    input.IZone = 38;
    OpCPrjTranRlt output;
    try
    {
        output = viewport.PrjTran(input);
    }
    catch
    {
        return;
    }
    for(int j=0;j<output.PrjPointNum;j++)
    {
        xArr[j] = output.PrjRltPoint[j].X;
        yArr[j] = output.PrjRltPoint[j].Y;
```

```
                }
            }
            area = CalculateArea2(xArr, yArr);
            len = CaculateTheDistance(xArr, yArr);
        }
        catch
        {
            this.Label1.ForeColor = System.Drawing.Color.Red;
            this.Label1.Text = "请确保您输入的坐标串正确!";
            return;
        }
        this.Label1.Text = area.ToString();
        this.Label2.Text = len.ToString();
    }
```

通过在客户端点击“计算面积周长”按钮，触发点击事件 Button1_Click 实现计算坐标值串形成区域的面积和周长功能。

# 6 示例开发与实现(JAVA 版)

## 6.1 站点搭建过程

以下是运用 Eclipse 平台手动搭建 JAVA 示例站点的具体介绍。

JEntComm 实例在 Eclipse 环境下的具体开发过程如下（具体代码见安装目录下站点 JEntComm 的代码）：

1. 在 Eclipse 开发环境中创建“JEntComm”工程。具体步骤如下：

（1） 新建 WEB 项目，单击文件-〉新建-〉项目，如图 6-1-1；

（2） 在新建项目窗口中选择 MyEclipse->J2EE Project-> Web Project 项，单击“下一步”，如图 6-1-2；

（3） 在 New J2EE Web Project 窗口中设置站点名称、源路径、站点目录和虚拟目录项，输入站点名为“JEntComm”后单击“完成”，如图 6-1-3；

（4） 站点创建成功，如图 6-1-4。

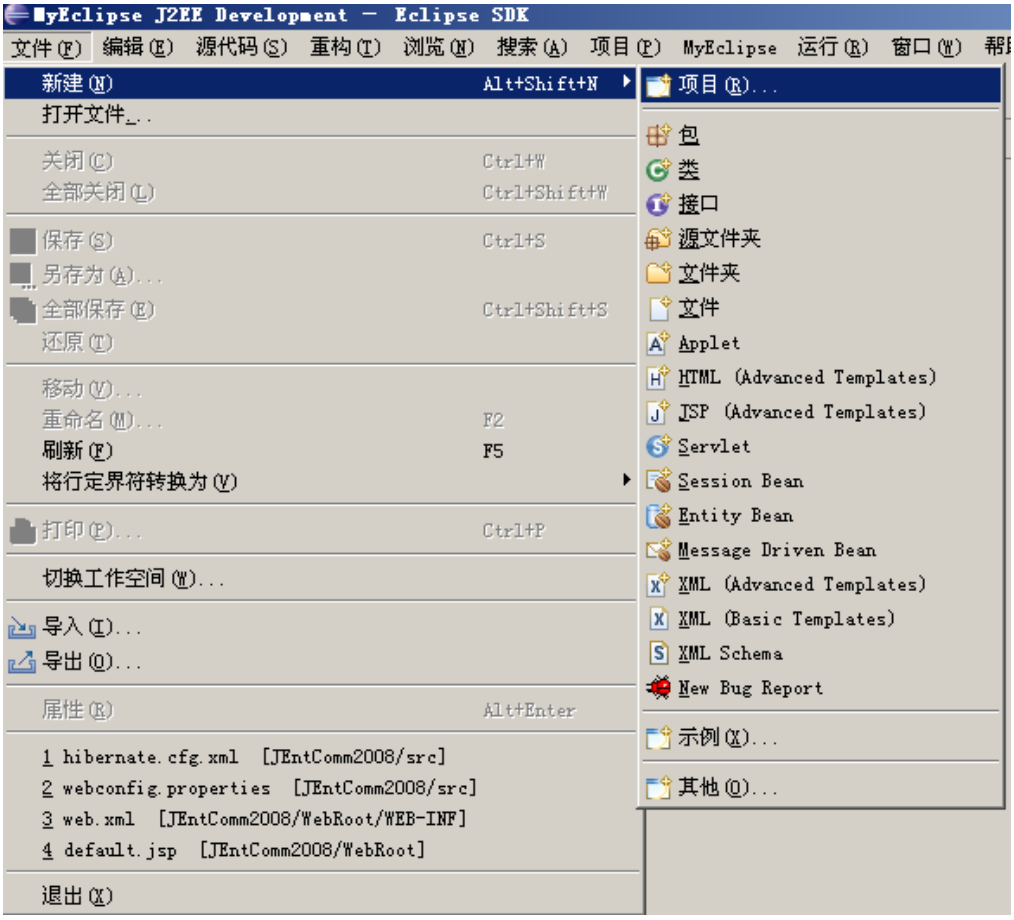



图 6-1-1 新建网站（一）

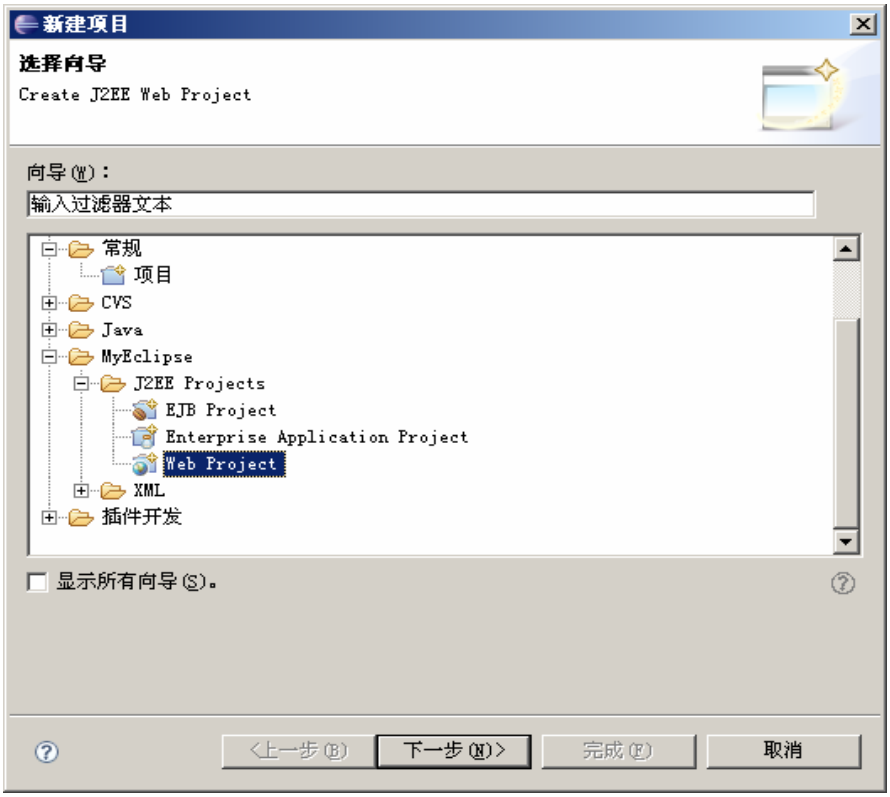



图 6-1-2 新建网站（二）

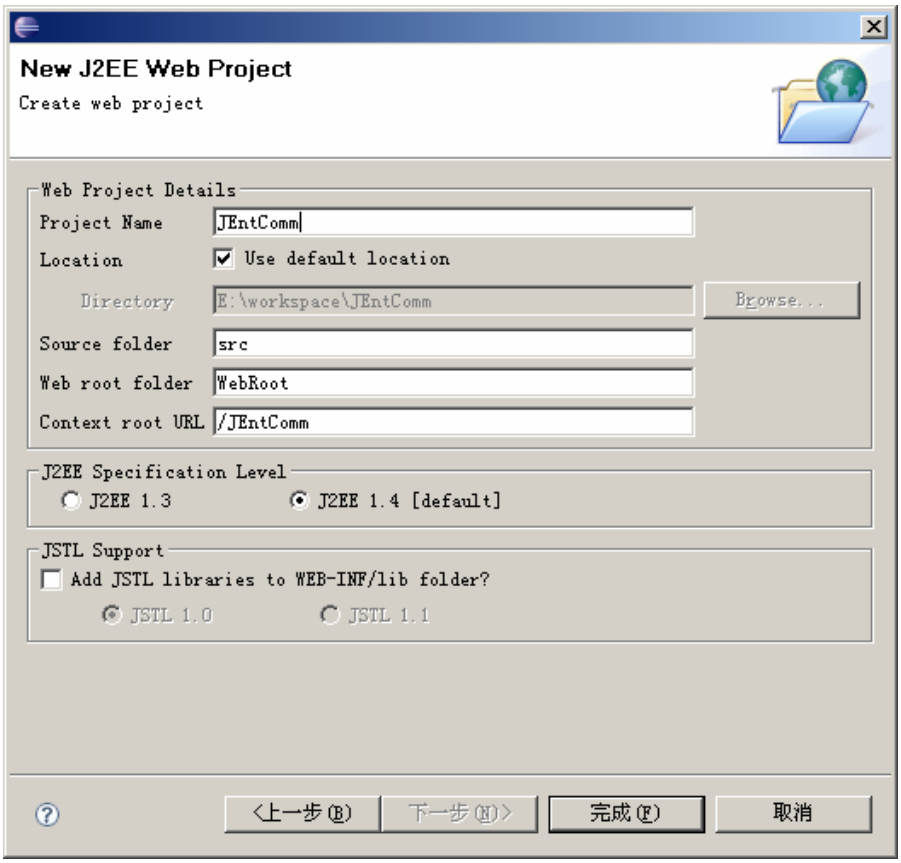



图 6-1-3 新建网站（三）

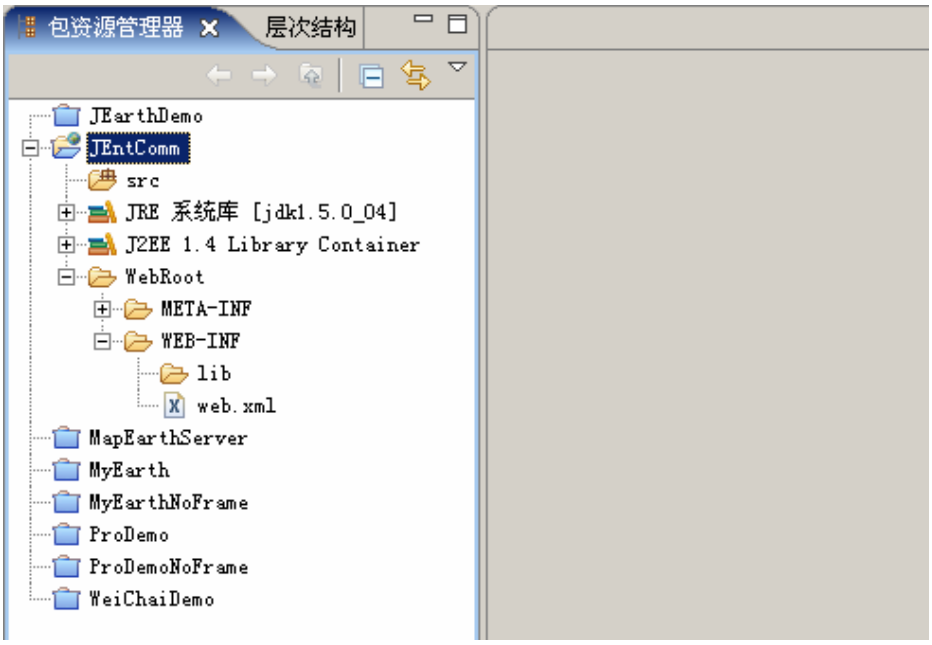



图 6-1-4 网站创建成功

2. 拷贝示例站点中的“icons”“images”、“css”文件夹到工程 WebRoot 下面，然后刷新工程。“icons”与“images”用于存放站点用到的图标，“css”用于存放站点的样式表文件。具体如图：

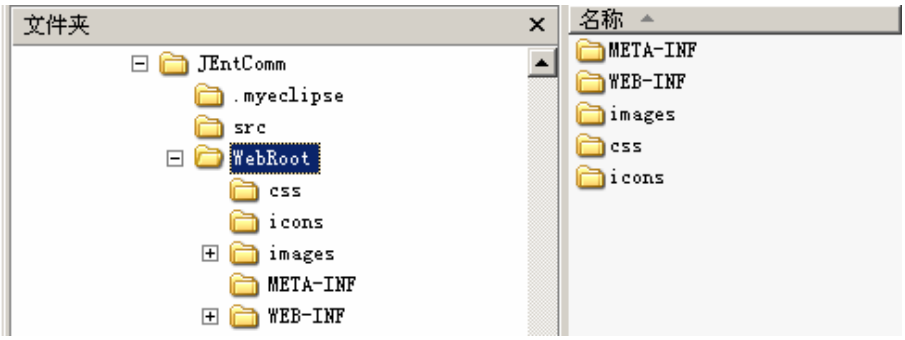



图 6-1-5 拷贝 css 等文件夹

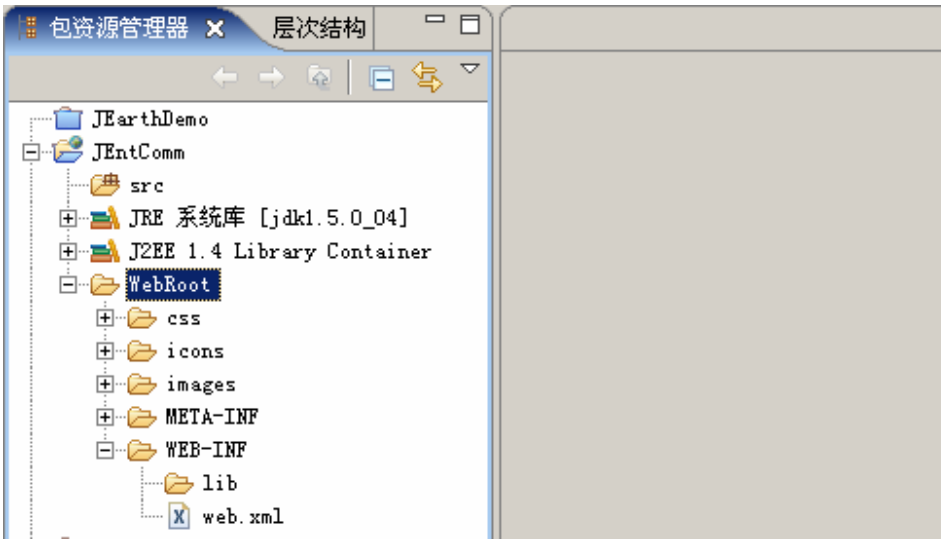



图 6-1-.6 刷新工程

3. 在工程 WebRoot 下面添加“jslib”文件夹，拷贝示例站点 WebRoot 目录下的 jslib 文件夹中的脚本文件到新建的“jslib”文件夹中，然后刷新工程。具体如图：

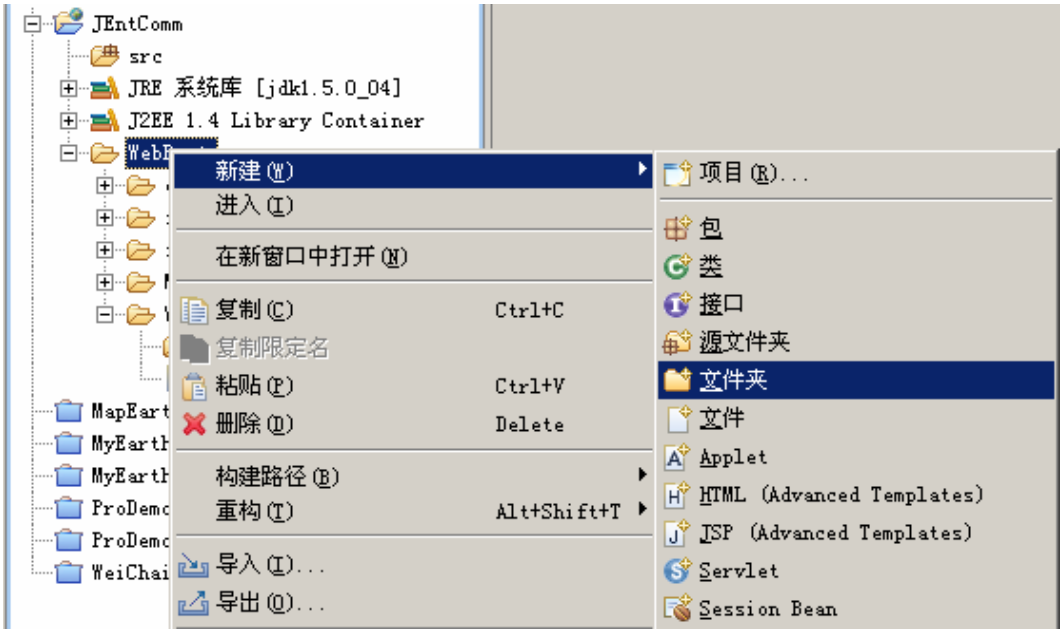



图 6-1-7 jslib 文件夹（一）




图 6-1-8 新建 jslib 文件夹（二）

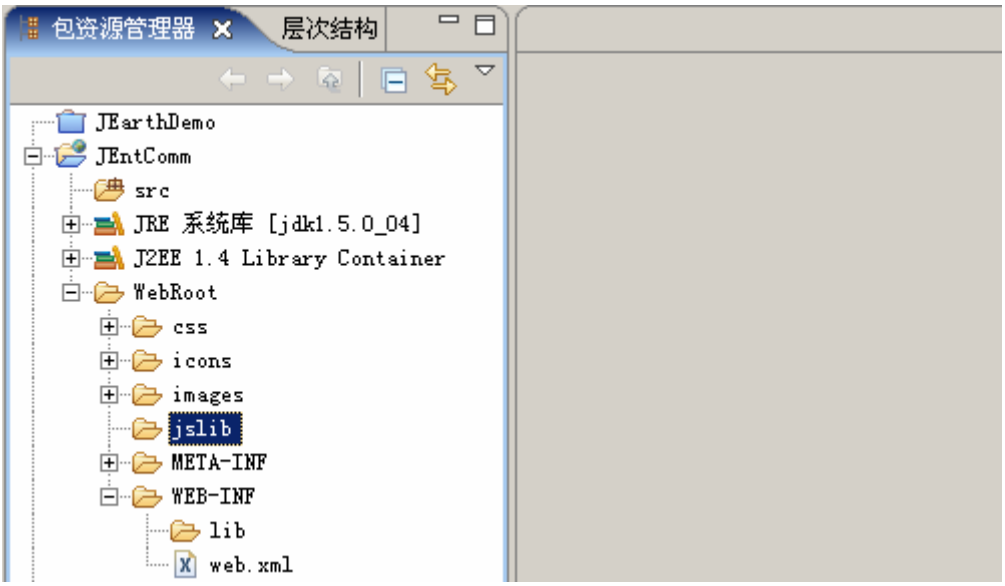



图 6-1-9 新建 jslib 文件夹（三）

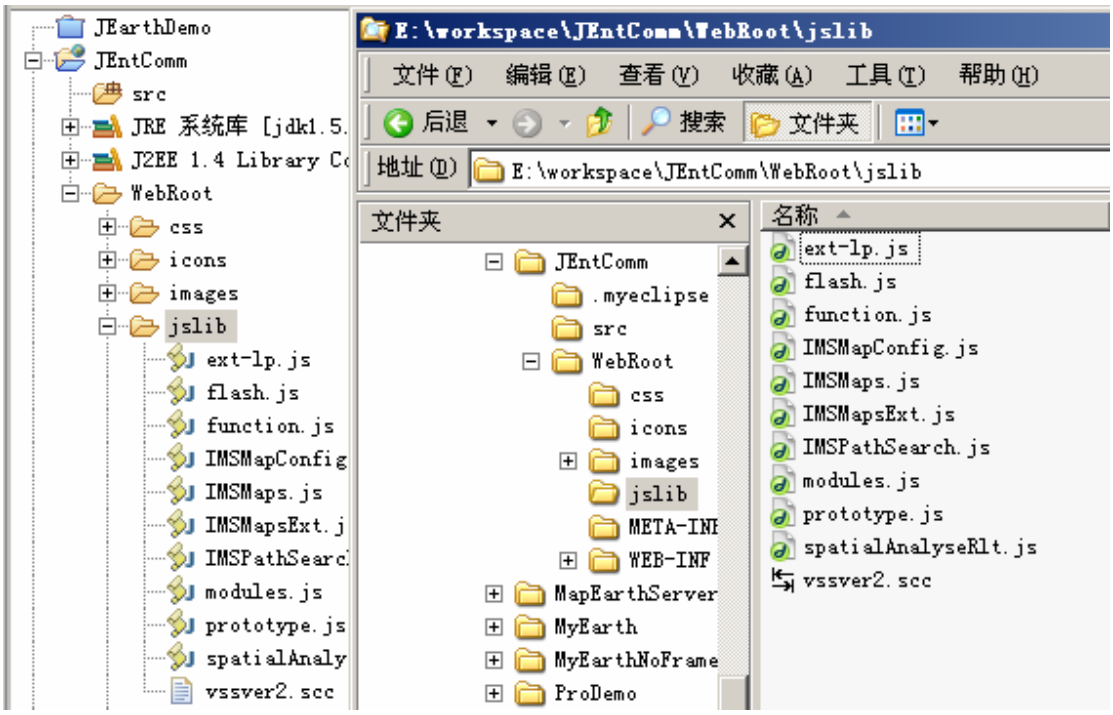



图 6-1-10 拷贝 js 文件并刷新

注：jslib 文件夹下的 js 文件是站点的客户端脚本库。

4. 将示例站点 WebRoot 下 WEB-INF 目录里的 lib 文件夹中所有类库拷贝到新建工程相应的目录下，然后刷新工程。具体如图:

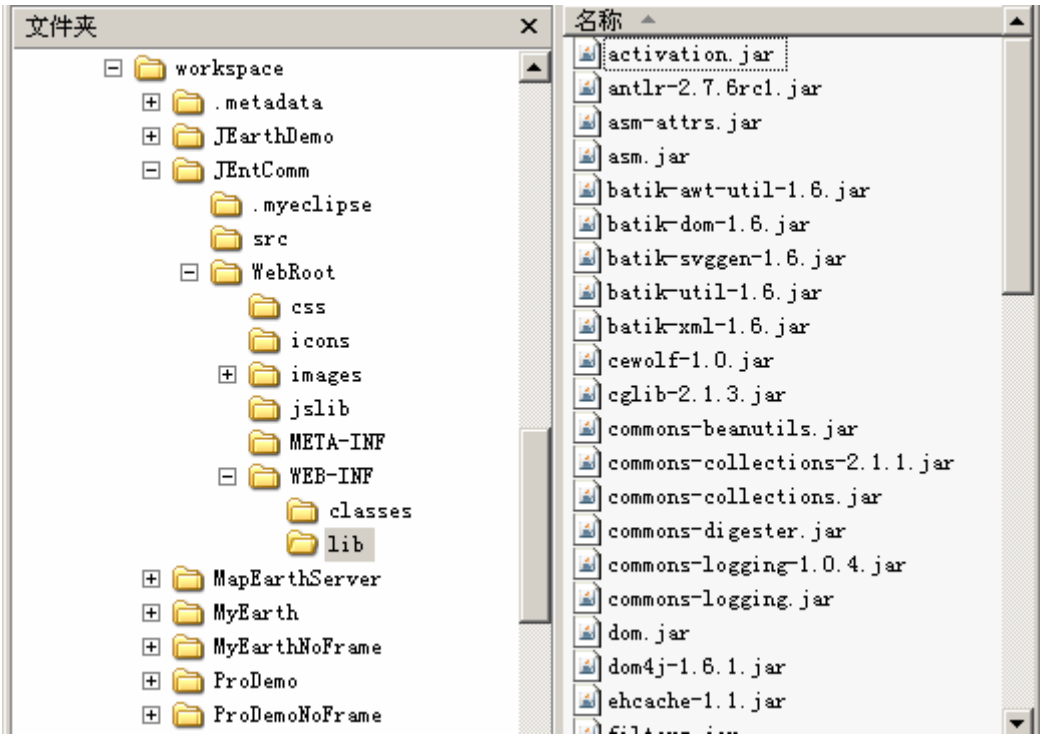



图 6-1-11 拷贝 lib 下 jar 文件

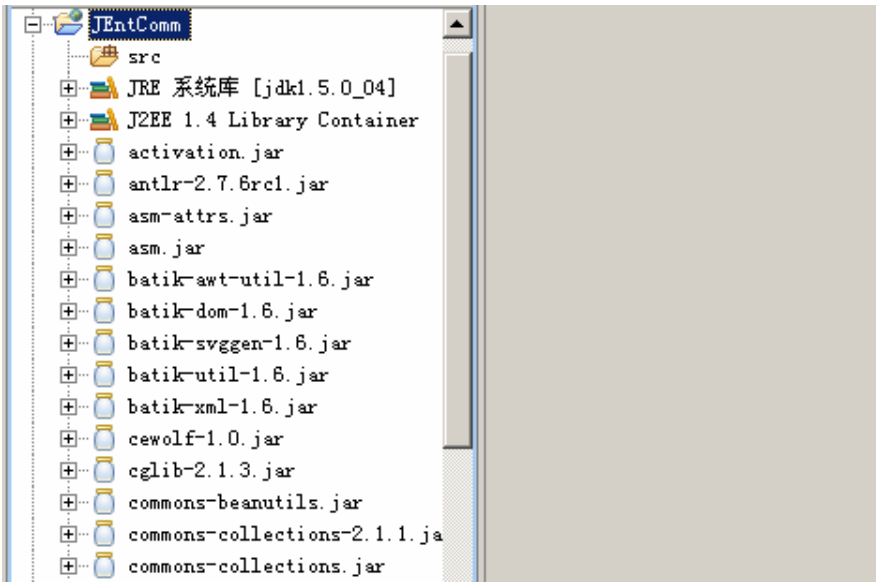



图 6-1-12 刷新工程

注：lib 文件夹下的 jar 包是站点中用到的所有类库，其中 jmapgis-enterprise.jar 是 GIS 服务器类库。

5. 将站点源目录 src 下的所有源文件拷贝到新建工程的源文件夹 src 中，然后刷新工程。接着根据站点实际情况修改其中的 hibernate.cfg.xml 文件设置数据库连接相关参数，修改 webconfig.properties 文件配置 GIS 服务器。具体如图：

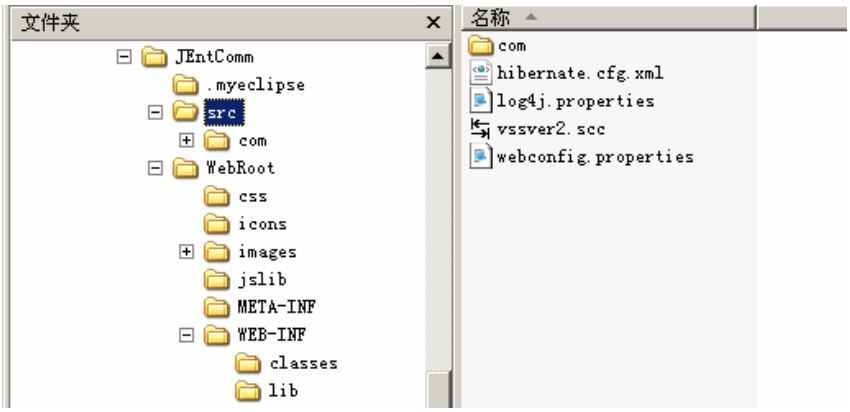



图 6-1-13 拷贝 src 下的源文件

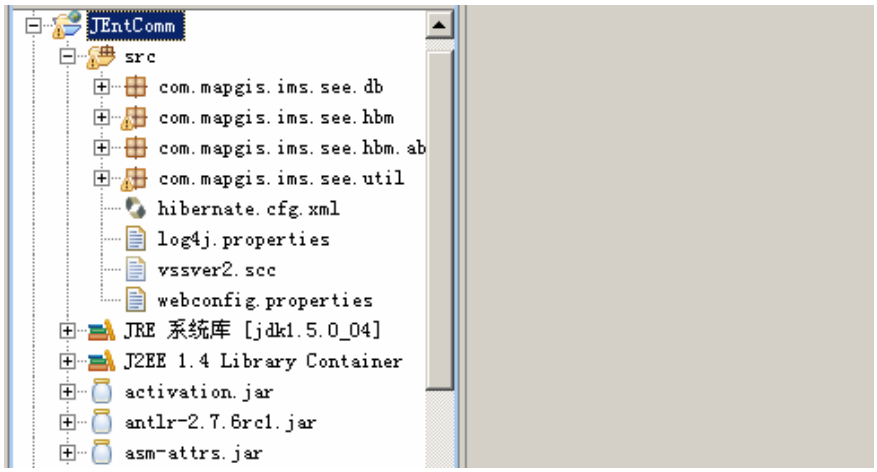



图 6-1-14 刷新工程

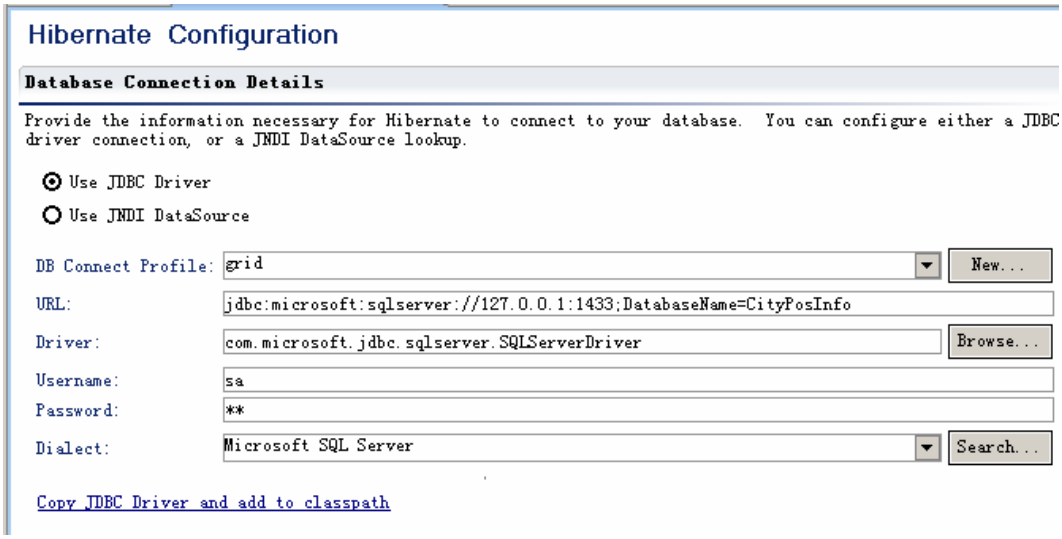



图 6-1-15 修改 hibernate.cfg.xml 文件

注：该站点是采用 hibernate 方式实现与数据库的交互操作。在这一步骤中可新添一个 hibernate 文件并进行相应配置；也可按照上述方法直接拷贝 hibernate 文件，然后在 hibernate.cfg.xml 文件中修改连接文件、数据库 IP、名称、用户名和密码等配置。

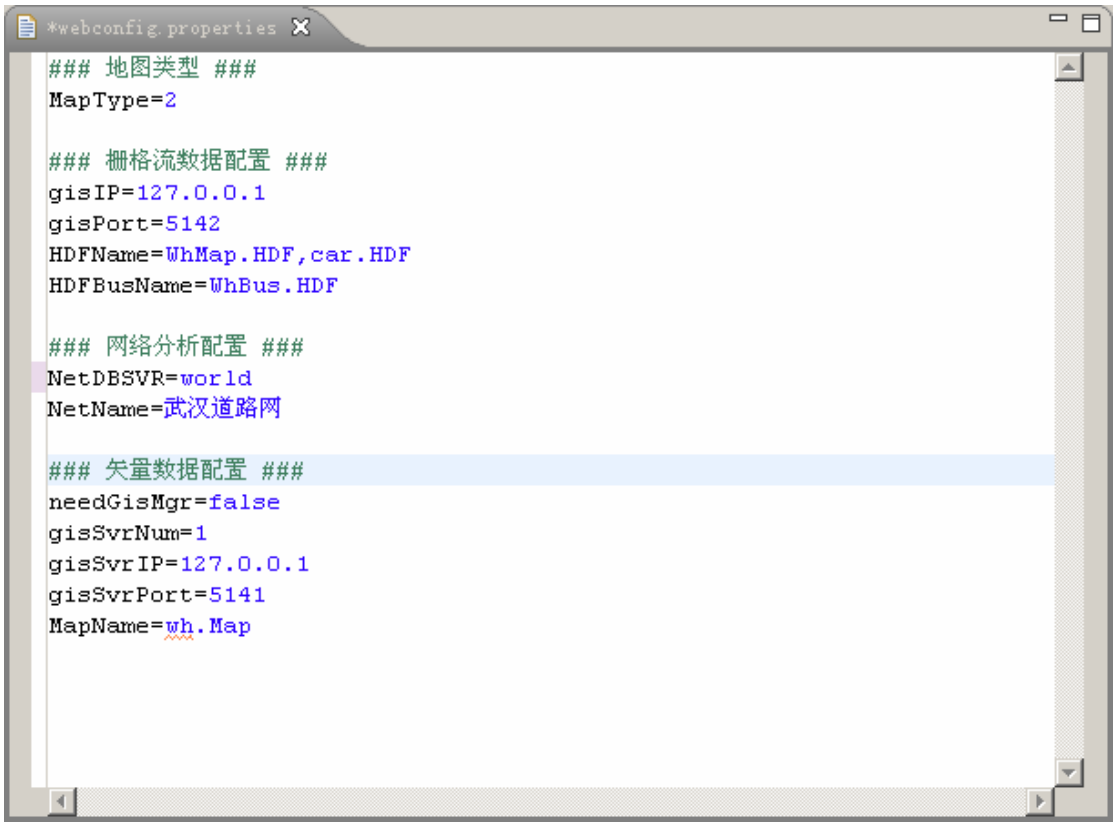



图 6-1-16 修改 webconfig.properties 配置文件

注：这是一个GIS服务器配置文件，具体参数如下所述。

地图类型：

- MapType: 数值代表地图的类型，0为栅格图，1为矢量图。

栅格流数据配置：

- gisIP: 栅格gis服务器IP地址。
- gisPort: 栅格gis服务器端口。

- HDFName：　基本搜索所用地图数据文件名。
- HDFBusName：　公交换乘所用的地图数据文件名。

网络分析配置：

- NetDBSVR：　表示网络分析所用数据文件的名称。
- NetName：　指的是网络分析要素数据集的名称。

矢量数据配置：

- needGisMgr：　GIS服务器负载均衡标记，值为ture时表示启用负载均衡，默认为false。
- gisSverNum：　数值代表GIS服务器的个数，值默认为1。
- gisSverIP：　矢量gis服务器IP地址。
- gisSverPort：　矢量gis服务器端口。
- MapName：　矢量地图文当名称。

6. 在新建工程 WebRoot 目录下新建一个名为“mapimage”的文件夹，该文件夹为站点进行相关操作产生的地图数据压缩包的存放目录。具体如图：




图 6-1-17 新建 mapimage 文件夹（一）

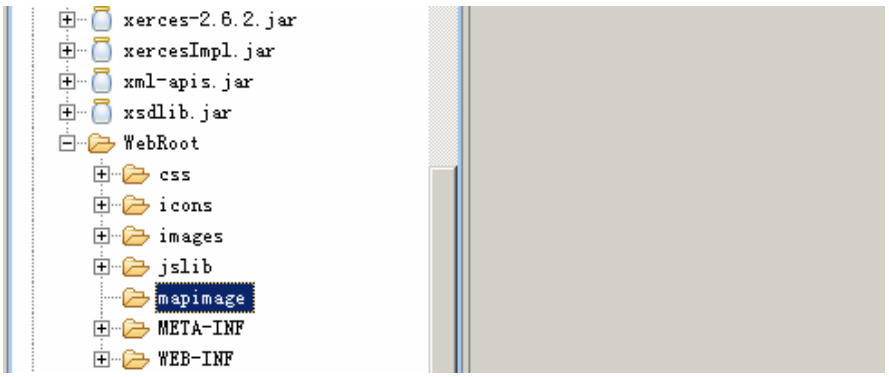



图 6-1-18 新建 mapimage 文件夹（二）

7. 将示例站点 WebRoot 中的所有.htm 文件和.jsp 文件直接拷贝到新建的工程的 WebRoot 目录下，然后刷新工程。具体如图：

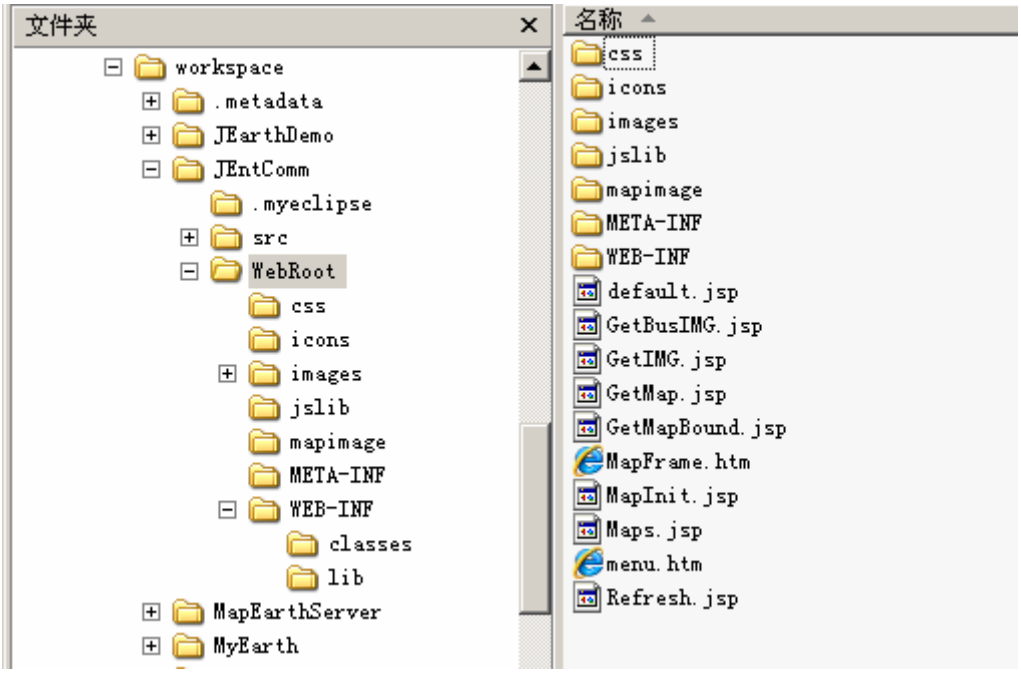



图 6-1-19 拷贝.htm 文件和.jsp 文件




图 6-1-20 刷新工程

注：MapFrame.htm 为主框架页面；GetIMG.jsp 和 GetBusIMG.jsp 文件为栅格流服务中取图的后台处理页面；GetMapBound.jsp 和 GetMap.jsp 文件为矢量服务中取图的后台处理页面；Maps.jsp 文件为各种查询的后台处理页面。这些都是站点中非常关键的文件。

8. 将示例站点 WebRoot 中的 VectorPage、ChartFiles、FusionCharts 三个文件夹及其下的所有文件、ChineseDictionary.txt 文本文件直接拷贝到新建的工程的 WebRoot 目录下，然后刷新工程。具体如图：

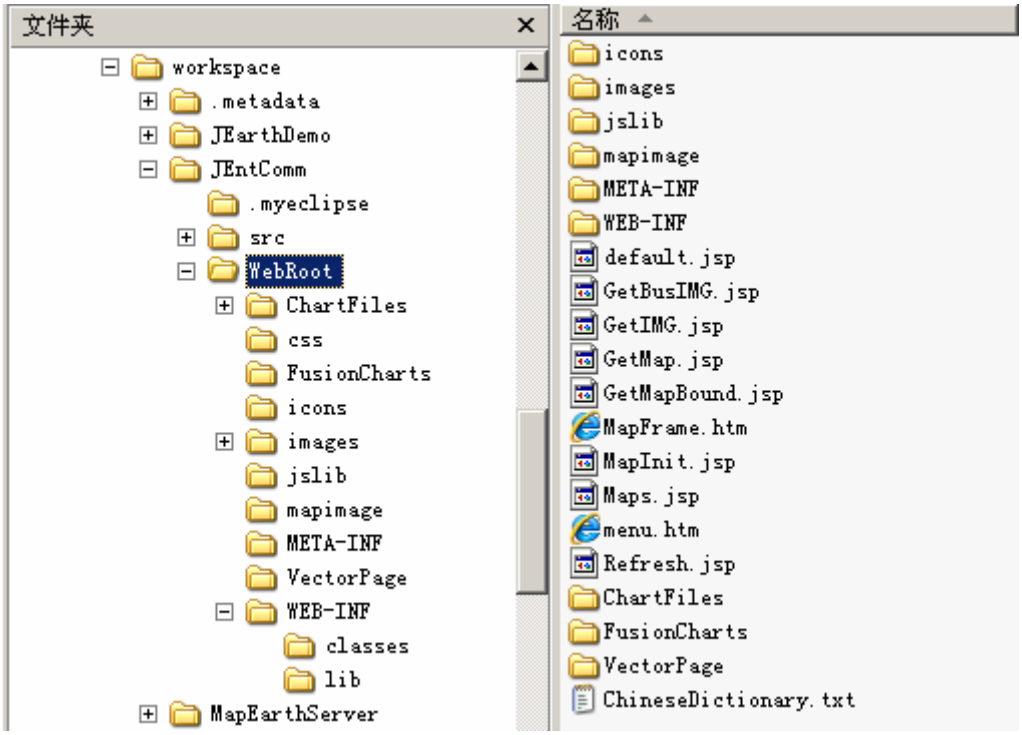



图 6-1-21 拷贝相关文件夹和文本文件

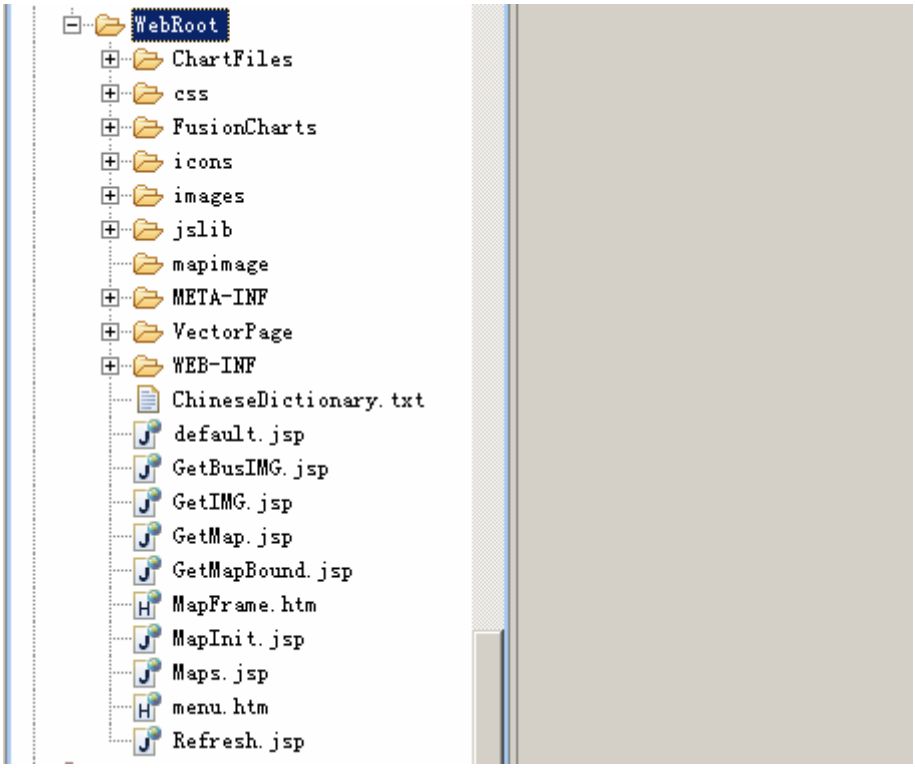



图 6-1-22 刷新工程

9. 在 Eclipse 环境中打开 WebRoot 目录下 WEB-INF 中的 web.xml 配置文件，添加相关代码，对网站起始页、字符编码等进行配置。该站点中，将 default.jsp 页面配置为起始页，字符编码方式为 UTF-8。具体如图：




图 6-1-23 打开 web.xml 配置文件

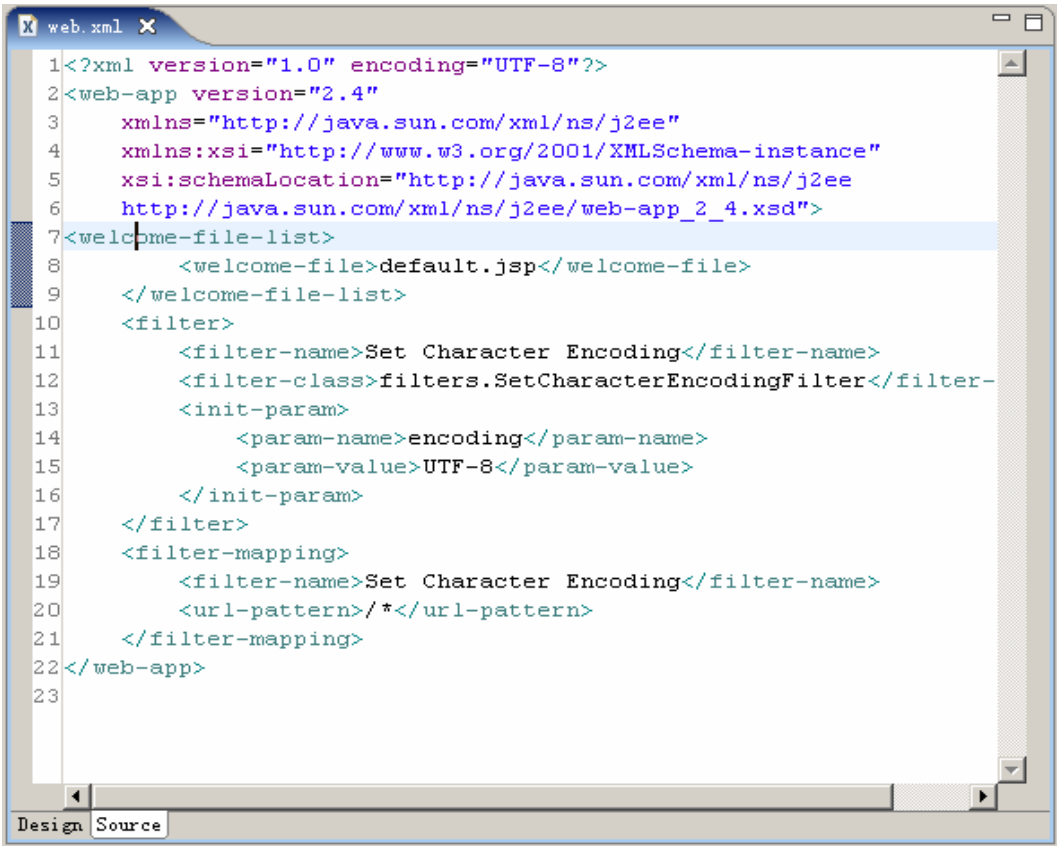



图 6-1-24 添加相关代码

注：若想搭建新的框架创建一个不同风格的站点，步骤与上面基本相同，只在第 7 步有所更改。即新建一个名为 MapFrame 的框架页面，并对新建框架添加相应功能代码，然后拷贝示例站点 WebRoot 目录下的其他.jsp 和.htm 文件。

# 6.2 站点具体功能实现

## 6.2.1 基本显示功能

### 6.2.1.1地图显示

地图显示功能是 MAPGIS7-IMS 行业版中最基本的功能，实现客户端向 GIS 服务器取图并把地图图像返回到客户端显示的过程。

矢量地图显示效果如图：

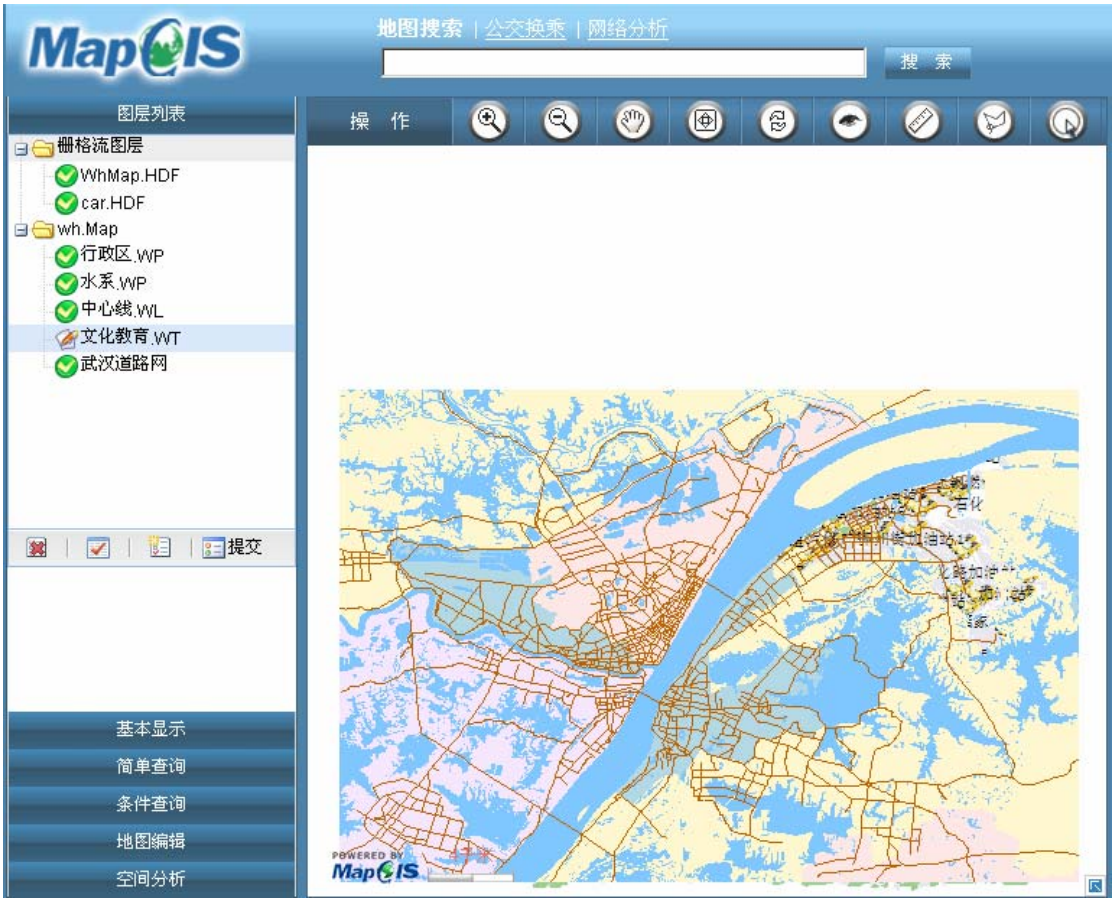



图 6-2-1-1 矢量地图显示效果图

该功能的实现流程如下：

1. 首先 default.jsp 页面从配置文件中获取地图文档名，再通过 onload 事件调用 AddMapList()得到地图文档名称列表并显示到客户端。
   Default.jsp 起始页面效果如下图：

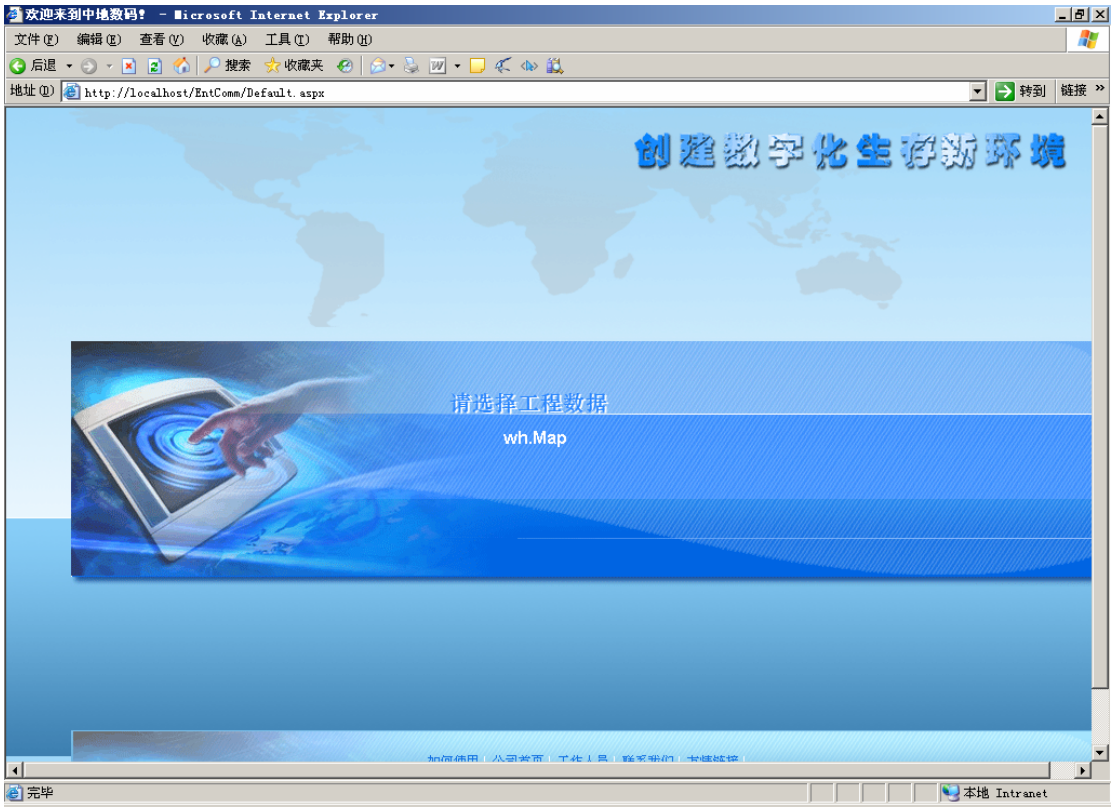



图 6-2-1-2 站点起始页面

➢ AddMapList()代码：

```
function AddMapList(){
   //得到地图文档名称列表
 var str="";
 var num = "<%=gisSvrMapNum%>";
   var gisSvrMapStr = "<%=gisSvrMap%>";
   var gisSvrMap = gisSvrMapStr.split(";");
   var i;

   for(i=0;i<num;i++){
    str +="<a
href=javascript:Transfer('"+gisSvrMap[i]+"')>"+gisSvrMap[i]+"</a><
BR>";
   }
 //把地图文档列表在客户端显示出来
 document.getElementById("link").innerHTML = str;
}
```

➢ Transfer()代码：

```
function Transfer(mapName){
 document.location.href =
"MapInit.jsp?mapName="+mapName+"&rnd="+Math.random();
}
```

在 AddMapList 函数中首先获取从配置文件得到地图文档名称，然后通过 Transfer 函数转到 MapInit.jsp 页面加载地图，同时把地图文档列表显示到 id 为 link 的 div 层中。AddMapList 与 Transfer 脚本函数都位于 default.jsp 页面中。

2. 通过 MapInit.jsp 页面接收从客户端传过来的地图文档名称参数，再调用服务器类库中的方法加载地图，最后转到主框架页面 MapFrame.htm 中。

➢ **MapInit.jsp 页面代码：**

```
<%@ page
import="com.mapgis.ims.pro.layer.*,com.mapgis.ims.pro.function.Map
ViewPort"%>
<%
 //获取地图文档名称
 String mapName = request.getParameter("mapName");
 MapViewPort viewport = null;
 CMapDisplay mapDisplay = new CMapDisplay();
 session.removeAttribute("restoreFlg");
 session.setAttribute("mapName",mapName);
 //加载地图
 try{
     viewport = mapDisplay.LoadMapDisplay(mapName);
 }catch(Exception ex){
     out.println("<script language='javascript'>alert('加载地图失败，
请确认是否有相应的地图文档。');</script>");
     return;
 }
 session.setAttribute("viewport",viewport);

 //跳转到主页面
 response.sendRedirect("MapFrame.htm");
%>
```

在该处理页面中，首先导入两个服务器端的类库，即 com.mapgis.ims.pro.layer 与 com.mapgis.ims.pro.function.MapViewPort，然后用 mapDisplay 对象调用 LoadMapDisplay 方法来加载地图。导入的两个类库位于 jmapgis-enterprise.jar 中。

3. 在 MapFrame.htm 框架页面中添加 prototype.js、function.js、IMSMaps.js、IMSMapsExt.js、IMSPathSearch.js 和 IMSMapConfig.js 的引用。

➢ **添加引用代码：**

```
<script type="text/javascript" src="jslib/prototype.js"></script>
<script type="text/javascript" src="jslib/function.js"></script>
<script type="text/javascript" src="jslib/IMSMaps.js"></script>
<script type="text/javascript" src="jslib/IMSMapsExt.js"></script>
<scripttype="text/javascript" src="jslib/IMSPathSearch.js"></script>
<script type="text/javascript" src="jslib/IMSMapConfig.js"></scrip>
```

4. 在 MapFrame.htm 的 body 标记中通过 onresize 事件触发 resize 脚本，onload 事件触发 pageload 脚本。resize()函数从客户端获取地图容器的高宽并对其进行初始化，pageload()对主框架中的子框架进行初始化。它们都位于 function. js 脚本库中。

5. 重要一步：在脚本函数库“function.js”中添加窗口事件监听，在该监听中调用初始化地图的脚本函数 initMAP()。

➢ **窗口事件监听代码：**

```
Event.observe(window, "load", function () {
 initMap();
});
```

➢ **initMap()部分代码：**

```
//初始化地图
function initMap(strlatlon, zoom) {
 resize();
 loadMap(1,IMSMapConfig);
  ……
}
```

initMap 脚本函数的省略部分用于添加右键菜单，用户可根据自己的需求进行适当修改。

➢ **loadMap()部分代码：**

```
function loadMap(mapType,mapConfig)
{
    map = new IMSMaps("zdims",mapType,mapConfig);
    //测距
    var sc = new IMSScale();
    sc.setUnits([103133.845,1000],["米","千米"]);
    map.sco = sc;
    map.addControl(sc,false);

    var l = new IMSLogo();
    map.addControl(l,false);
    ……
}
```

loadMap 脚本函数首先声明了 IMSMap 对象进行地图的初始化，然后可在该函数中添加 logo、鹰眼、放大镜、操作状态等各种控件。用户可根据自己的需要进行添加。

loadMap 脚本函数的参数 mapType 为地图类型，mapConfig 为地图配置类，根据需求获取不同的地图配置。该配置类位于脚本函数库 IMSMapConfig.js 中。站点默认打开矢量图，所以在 initMAP 脚本函数种调用 loadMap(1,IMSMapConfig)。

➢ **IMSMapConfig()配置代码：**

```
//地图配置类
function IMSMapConfig(){
    this.xMinMap = 114.125602229914;                    //整图的最小逻辑坐标
```

```
        this.yMinMap = 30.4539323507469;
        this.xMaxMap = 114.500788705197;                    //整图的最大逻辑坐标
        this.yMaxMap = 30.8291188260299;

        this.levelNum = 6;                                  //栅格地图级数
        this.statusBarFlag = true;                          //是否在状态栏显示当前鼠标
的坐标
        this.gridImageUrl = ["GetIMG.jsp","GetIMG.jsp"];    //栅格图请求页面
        this.vectImageUrl = ["GetMap.jsp"];
        this.getVectCoorPage = "GetMapBound.jsp";           //地图范围获取页面
    }
    //公交换乘配置文件
    function IMSMapConfig1(){
        this.xMinMap = 20090.85;                            //整图的最小逻辑坐标
        this.yMinMap = 21884.4;
        this.xMaxMap = 61537.35;                            //整图的最大逻辑坐标
        this.yMaxMap = 63330.5;
        this.levelNum = 4;                                  //栅格地图级数
        this.statusBarFlag=true;                            //是否在状态栏显示当前鼠标
的坐标
        this.gridImageUrl = ["GetBusIMG.jsp"];
    }
```

6. 在 Init()脚本函数中判断地图数据类型，可分为栅格和矢量地图数据两种方式处理：

（1） 栅格数据：流程图如下：

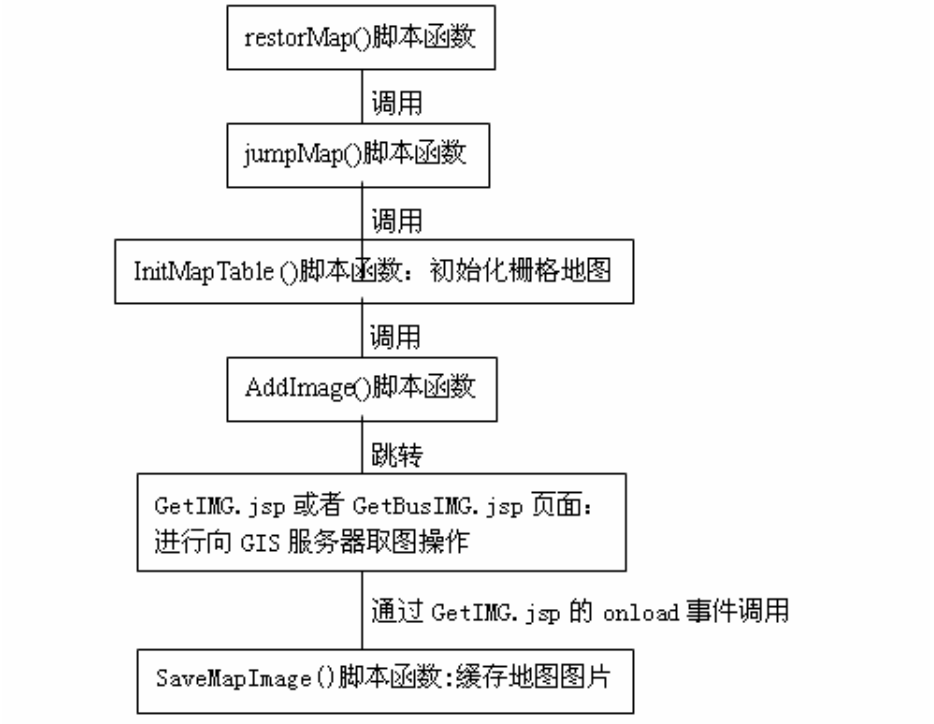



图 6-2-1-3 JAVA 版的栅格数据流程图

注：.NET 与 JAVA 开发使用的是同一套脚本库。InitMapTable()与 AddImage()脚本函数代码见.NET 示例开发与实现中地图显示部分的第六步，或者查看示例站点中的脚本库源码。

➢ GetIMG.jsp 页面代码：

```
<%@ page import="com.mapgis.ims.see.com.*" import="java.io.*"
pageEncoding="UTF-8" isThreadSafe="false"%><%
String hdfName[] = null;                    //HDF地图名称
byte[] by_mapImage = null;                  //地图图片二进制流
CMapViewPort imgviewport = null;
   int rowNo =
Integer.parseInt(request.getParameter("a").toString());
int lineNo =
Integer.parseInt(request.getParameter("b").toString());
int level = Integer.parseInt(request.getParameter("c").toString());
//加载地图参数
int layerNo = Integer.parseInt(request
              .getParameter("layerNo").toString());
ServletConfig myconfig = getServletConfig();
String gisIP = myconfig.getInitParameter("gisSvrIP");
int gisPort = Integer.parseInt(myconfig
              .getInitParameter("gisSvrPort"));
//获取地图图层名称
hdfName = myconfig.getInitParameter("HDFName").split(",");
//地图对象初始化
Object flag = session.getAttribute("imgviewport");
try{
    if(flag == null){
        imgviewport = new CMapViewPort(gisIP, gisPort);
        session.setAttribute("imgviewport",imgviewport);
    }else{
        imgviewport =
(CMapViewPort)session.getAttribute("imgviewport");
    }
}
catch(Exception ex){
    System.out.println("地图对象加载失败。");
    return;
}
try{
    //设置地图的偏移量
    imgviewport.SetIndexOffset(0);
    //获取地图
```

```
    by_mapImage =
imgviewport.GetMapImage(hdfName[layerNo],0,rowNo,lineNo,level);
}catch(Exception ex){
    System.out.println(ex.getMessage());
    FileInputStream in = new
FileInputStream(application.getRealPath("images/mapgis.jpg"));
        byte[] date = new byte[in.available()];
        in.read(date);
    by_mapImage = date;
}
   try{
    response.reset();
    response.getOutputStream().write(by_mapImage);
    response.getOutputStream().flush();
    response.getOutputStream().close();
}catch(IllegalStateException e)
{
    e.printStackTrace();
}
%>
```

该页面主要实现从 GIS 服务器取图和输出图片的功能,是通过调用服务器端类库中 CMapViewPort 类的 GetMapImage 方法实现的。

（2）矢量数据：通过 GetVectBound 脚本函数转到 GetMapBound. jsp 页面处理获取矢量图坐标范围，然后通过 Init()函数调用的 GetVectorMap 脚本函数中再调用 GetVectorMapByCood()，转到 GetMap.jsp 页面进行向 GIS 服务器取图的操作。

注： GetVectBound 与 GetVectorMapByCood 脚本函数代码见.NET 示例开发与实现中地图显示部分的第六步，或者查看示例站点中的脚本库源码。

- GetMapBound. jsp 页面代码：

```
<%@ page import="com.mapgis.ims.pro.layer.*"
import="com.mapgis.ims.pro.function.*"%>
<%
//当前地图范围
String curLogicMinX = "";
String curLogicMinY = "";
String curLogicMaxX = "";
String curLogicMaxY = "";
//原始地图范围
String orgLogicMinX = "";
String orgLogicMinY = "";
String orgLogicMaxX = "";
String orgLogicMaxY = "";
MapViewPort viewport = null;
```

```
CRect rc = null;
int width;
int height;
try{
    //设置地图高宽
    width = Integer.parseInt(request.getParameter("w"));
    height = Integer.parseInt(request.getParameter("h"));
    boolean isreload =
Boolean.parseBoolean(request.getParameter("reload"));
    String mapName =
String.valueOf(session.getAttribute("mapName"));
    CMapDisplay mapDisplay = new CMapDisplay();
    //加载地图
    if (session.getAttribute("viewport") == null){
        try{
            viewport = mapDisplay.LoadMapDisplay(mapName);
            session.setAttribute("viewport",viewport);
        }catch(Exception ex){
            System.out.println(ex.getMessage());
            out.println("0,0,0,0,0,0,0,0");
        }
    } else{
        viewport = (MapViewPort) session.getAttribute("viewport");
        viewport.setImageWidth(width);
        viewport.setImageHeight(height);
    }

    if (session.getAttribute("restoreFlg") == null || isreload){
          viewport.Restore();
          session.setAttribute("restoreFlg",true);
       }
       //得到当前地图显示范围
       rc = viewport.getMapDisplayBound();
    //设置当前地图显示范围
    curLogicMinX = String.valueOf(rc.getXMin());
    curLogicMinY = String.valueOf(rc.getYMin());
    curLogicMaxX = String.valueOf(rc.getXMax());
    curLogicMaxY = String.valueOf(rc.getYMax());
    //得到原始地图显示范围
    rc = viewport.getMapOriginalBound();
    orgLogicMinX = String.valueOf(rc.getXMin());
    orgLogicMinY = String.valueOf(rc.getYMin());
    orgLogicMaxX = String.valueOf(rc.getXMax());
    orgLogicMaxY = String.valueOf(rc.getYMax());
```

```
    //向客户端输出图片逻辑范围，有相应ajax函数获取
out.println(curLogicMinX.toString() + "," + curLogicMinY.toString()
+ "," + curLogicMaxX.toString() + ","+ curLogicMaxY.toString() + ","
+ orgLogicMinX.toString() + "," + orgLogicMinY.toString()+ "," +
orgLogicMaxX.toString() + ","+ orgLogicMaxY.toString());
 }catch(Exception ex){
    System.out.println(ex.getMessage());
    out.println("0,0,0,0,0,0,0,0");
 }
%>
```

该页面从客户端接收参数，加载地图后设置地图的高宽，然后调用服务器端类库的 getMapDisplayBound() 方法获得并设置当前地图显示范围，调用 getMapOriginalBound()方法获得并设置原始地图显示范围，最后向客户端输出图片逻辑范围。

- **GetMap.jsp 页面部分代码：**

```
 if(!errorFlag){
    try{
 //地图显示逻辑范围
    xmin = Double.parseDouble(request.getParameter("LogicMinX"));
    ymin = Double.parseDouble(request.getParameter("LogicMinY"));
    xmax = Double.parseDouble(request.getParameter("LogicMaxX"));
    ymax = Double.parseDouble(request.getParameter("LogicMaxY"));
 //地图高宽
    w = Integer.parseInt(request.getParameter("ImageWidth"));
    h = Integer.parseInt(request.getParameter("ImageHeight"));
    IsEagle =
Boolean.parseBoolean(String.valueOf(request.getParameter("IsEagle"
)));
    }catch(Exception e){
           errorFlag = true;
           message = "参数获取失败";
    }
 }
   if(!errorFlag){
    viewport.setImageWidth(w);
    viewport.setImageHeight(h);

    CRect rc = new CRect();
    rc.setXMin(xmin);
    rc.setYMin(ymin);
    rc.setXMax(xmax);
    rc.setYMax(ymax);
    //设置地图显示逻辑范围
```

```
    viewport.setMapDisplayBound(rc);
            try {
        //设置是否跟随比例显示
        viewport.setFollowScale("true");

viewport.setImageFolder(application.getRealPath("/mapimage"));
        //设置出图方式：true：直接生成二进制流；false：先在本地生成图片
        viewport.setBuildFileMode(true);
        //设置生成文件类型:2生成.png文件,1生成.gif文件
        viewport.setImageFormat(Byte.valueOf("1").byteValue());
        //获取地图图片
        bufferArray = viewport.GetMapImage();
    } catch (Exception e) {
        errorFlag = true;
        message = "加载地图失败";
    }
   //负载均衡标志位
    boolean needGisMgr =
Boolean.parseBoolean(webconfig.getInitParameter("needGisMgr"));
    //设置负载均衡标志位
    MapViewPort.setNeedGisMgr(needGisMgr);
……
try{
    response.reset();
    response.getOutputStream().write(bufferArray);
    response.getOutputStream().flush();
    response.getOutputStream().close();
}catch(IllegalStateException e){
    e.printStackTrace();
}
```

该页面主要接收地图的逻辑范围与高宽参数，再调用服务器端类库中 CMapView 类的方法进行各种参数设置，最后调用 GetMapImage()方法从 GIS 服务器取图，以二进制流形式返回输出图片。

## 6.2.1.2基本操作

基本操作实现地图的放大、缩小、移动、更新、复位等操作。该功能实现流程如下：

1. 通过地图初始化 loadMap 函数（位于 function.js 中）加载地图操作状态类。

➢ **加载地图操作状态类代码：**

```
//操作状态
   var os = new IMSOperStatus();
   map.addControl(os,false);
```

IMSOperStatus()位于 IMSMapsExt.js 脚本函数库中，是地图操作状态类。

2. 在框架页面的客户端工具栏（位于 MapFrame.htm、menu.htm 页面中）或初始化地图脚本函数右键菜单部分，并添加基本操作的功能按钮代码，通过触发 onclick 事件的 setCurOper 函数更改当前地图操作状态。

   menu.htm 页面中基本操作的功能按钮效果如下图：




图 6-2-1-4 基本操作的功能按钮

➢ **在 MapFrame.htm 页面 id 为 cmdGroup0 的 div 层中添加如下代码：**

```
<img src="images/buttonx/button1.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button1_02.gif'"
    onmouseout="this.src='images/buttonx/button1.gif'"
    title="放大" onclick="map.setCurOper(1001)"/>
 <img src="images/buttonx/button2.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button2_02.gif'"
    onmouseout="this.src='images/buttonx/button2.gif'"
    title="缩小" onclick="map.setCurOper(1002)"/>
<img src="images/buttonx/button3.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button3_02.gif'"
    onmouseout="this.src='images/buttonx/button3.gif'"
    title="移动" onclick="map.setCurOper(1003)"/>
<img src="images/buttonx/button4.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button4_02.gif'"
    onmouseout="this.src='images/buttonx/button4.gif'"
    title="复位" onclick="map.setCurOper(1004)"/>
 <img src="images/buttonx/button5.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button5_02.gif'"
```

```
onmouseout="this.src='images/buttonx/button5.gif'"
title="刷新" onclick="onDisplayToolClick('reflash')"/>
```

setCurOper 函数就是工具栏按钮被点击时触发的函数，它是一个很关键的脚本函数，位于脚本库 IMSMaps.js 内。具有一个输入参数 operType，代表按钮触发事件的代码，如：

- 1001（ IMSOperType.ZoomIn）—放大；
- 1002（IMSOperType.ZoomOut）—缩小；
- 1003（IMSOperType.Drag）—移动；
- 1004（IMSOperType.Restore）—复位。

➢ **setCurOper 函数基本操作部分代码：**

```
//设定当前操作状态
this.setCurOper=function(operType)
{
    this.operType = parseInt(operType);
    this.clearOper();
    switch(this.operType)
    {
        case 1001:
            this.activeTool = "ZoomIn";
            this.drawAction = "Rect";
        break;
        case 1002:
            this.activeTool = "ZoomOut";
            this.drawAction = "Rect";
        break;
        case 1003:
            this.activeTool = "Pan";
            this.drawAction = "Pan";
        break;
        case 1004:
            this.restoreMap();
        break;
        case 1005://清除
            if(this.vmlObj!=null)
            {
                this.vmlObj.dispose();
                this.vmlObj = null;
            }
        break;
        case 1006://刷新
              this.refresh();
       ……
      default:
            this.activeTool = "Pan";
```

```
            this.drawAction = "Pan";
        break;
      }
   };
```

3. 在地图上做点击或拉框操作时，客户端会产生相应的参数，通过鼠标事件传到服务器端页面（GetIMG.jsp 或 GetMap.jsp）处理，处理完成后生成新的地图，并将新地图返回客户端显示。

例如放大操作：

通过鼠标弹起事件 OnMapMouseUp 调用 ChargeRect()脚本函数，在该函数中根据传递参数调用 ScaleZoomOutOrIn()函数，然后在 ScaleZoomOutOrIn 脚本中调用 jumpMap()进行处理，最后转到服务器端页面（GetIMG.jsp 或 GetMap.jsp）处理并返回新图。

- **ChargeRect()脚本函数代码：**

```
this.ChargeRect = function(lminx,lminy,lmaxx,lmaxy)
   {
       switch(this.activeTool)
      {
         case "ZoomIn":
             this.ScaleZoomOutOrIn(true);
            break;
         case "ZoomOut":
             this.ScaleZoomOutOrIn(false);
            break;
         case "Pan":
             if(this.hasVectorMap)
             {
                if(this.mapType!=1)
                   this.GetVectorMap();
                else
                {
                   this.jumpMap(0,0,0);
                   break;
                }
             };
            this.AdjustControl();
            break;
       default :
           break;
      }
 };
```

- **ScaleZoomOutOrIn()脚本函数代码：**

```
//放大缩小地图
 this.ScaleZoomOutOrIn=function(flag)
   {
```

```
    if(this.mapType==1)
    {
        this.jumpMap(0,0,0);
        return;
    }
    if(!flag&&this.inf[2]*1<=0)
     return;
    if(flag&&this.inf[2]*1>=this.levelNum-1)
     return;
    var x=(this.mdLogicX+this.muLogicX)/2;
    var y=(this.mdLogicY+this.muLogicY)/2;
    this.centerCoor = [x,y];
    if(flag)
    {
        this.inf[2]=parseInt(this.inf[2])+1;
    }
    else
    {
        this.inf[2]-=1;
    }
  this.jumpMap(x,y,(this.inf[2]+1));
   };
```

注：OnMapMouseUp()、ChargeRect()、ScaleZoomOutOrIn()、jumpMap()都位于IMSMaps.js 脚本库中。jumpMap()等脚本函数参见示例站点脚本库源码。

4. 以放大功能为例，客户端具体操作如图：




图 6-2-1-5 步骤 1-点击放大按钮

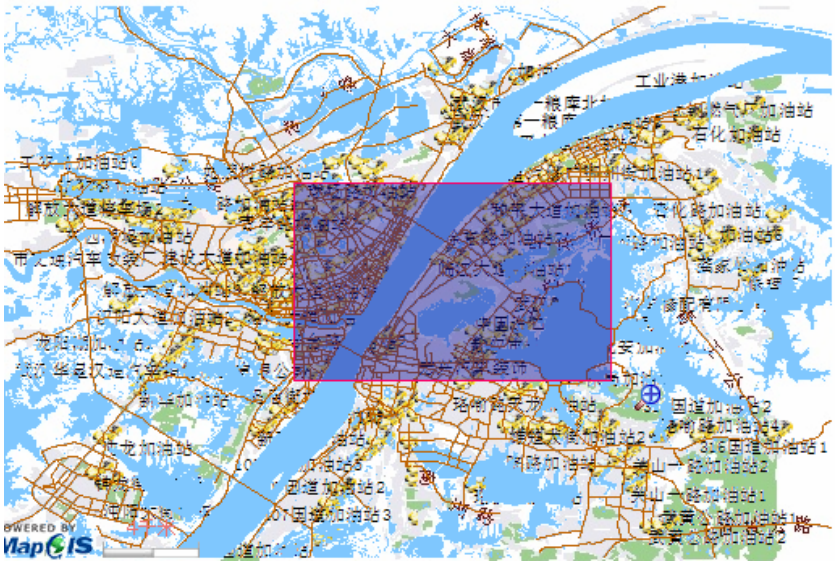


图 6-2-1-6 步骤 2-在地图上拉框




图 6-2-1-7 步骤 3-服务器端处理后放大结果显示

## 6.2.2 放大镜功能

放大镜功能是将鼠标指针所在点周围一定区域内的图像按一定比例进行放大显示，以方便用户更加清楚的查看地图的内容，放大结果显示于鼠标右下方，放大镜显示的图像中心点是鼠标的中心点。该功能是基于地图基本功能进行搭建的，实现流程如下：

1. 在地图图像显示功能的 InitMap()脚本函数（位于 IMSMaps.js 里）中通过调用 LoadGridMagnifier()或 LoadVectMagnifier()函数来加载放大镜。

➢ LoadGridMagnifier()代码：

```
this.LoadGridMagnifier=function()//加载放大镜
{
    var zoomDiv = this.GetMapMagnifier();
       for(var i=0;i<this.gridImageUrl.length;i++)
       {
          zoomDiv.innerHTML+="<table id='zoomTab"+i+"'
cellSpacing='0' cellPadding='0' border='0'
style='position:absolute;z-index:"+i+"'>"
                  +"<tr><td id='td"+i+"20'></td><td
id='td"+i+"21'></td><td id='td"+i+"22'></td></tr>"
                  +"<tr><td id='td"+i+"10'></td><td
id='td"+i+"11'></td><td id='td"+i+"12'></td></tr>"
                  +"<tr><td id='td"+i+"00'></td><td
id='td"+i+"01'></td><td id='td"+i+"02'></td></tr></table>";
       };
};
```

该函数是地图类型为栅格图时被调用的，即在栅格地图中加载放大镜。

➢ LoadVectMagnifier()代码：

```
this.LoadVectMagnifier=function()//加载放大镜
{
    var zoomDiv = this.GetMapMagnifier();
    for(var i=0;i<this.vectImageUrl.length;i++)
    {
        var vectDiv = this.vectLay[i].cloneNode(true);
        vectDiv.id="vectZoom"+i;
        vectDiv.style.position="absolute";
           zoomDiv.appendChild(vectDiv);
    };
};
```

该函数是当地图类型为矢量图时被调用的，即在矢量地图中加载放大镜。

2. 通过在框架页面（MapFrame.htm）的客户端工具栏中添加放大镜的功能按钮代码，触发 onclick 事件的 setCurOper 函数更改当前地图操作状态。

➢ 在 MapFrame.htm 中 id 为 cmdGroup0 的 div 层中添加如下代码：

```
<img src="images/buttonx/button11.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button11_02.gif'"
    onmouseout="this.src='images/buttonx/button11.gif'"
    title="放大镜" onclick="map.setCurOper(3003)"/>
```

setCurOper 函数就是工具栏按钮被点击时触发的函数，位于脚本库 IMSMaps.js 内。具有一个输入参数 operType，代表按钮触发事件的代码。该功能中的按钮触发事件的操作码为 3003（IMSOperType.OpenMagnifier），即代表放大镜。

➢ **setCurOper 函数放大镜部分代码：**

```
case 3003:
        this.activeTool = "OpenMagnifier";
        this.drawAction = "OpenMagnifier";
        this.ShowMapMagnifier();
        this.addEventListener("mousemove",thisObj.MoveMapMagnifier,false);
 break;
```

在 setCurOper 函数中添加鼠标事件监听，通过该监听调用 ShowMapMagnifier()函数来实现放大镜功能，代码参见示例站点脚本库 IMSMaps.js 中的源码。

3. 在客户端选择地图操作状态为放大镜状态，并在地图上移动，通过鼠标监听事件把产生的参数传递到相应的函数进行处理，即可实现放大镜功能。其客户端具体操作见.NET 开发中的相应部分或者行业版的操作手册。

## 6.2.3 鹰眼显示功能

地图鹰眼，俗称地图的鸟瞰图或缩略图。在电子地图中鹰眼的功能非常强大，通过鹰眼可以知道地图的当前位置；也可以在鹰眼上点击、拖动或移动到想要查看的位置。鹰眼的可视范围总是比主图的可视范围大，鹰眼中心框的可视范围就相当于主图的可视范围，主图的地理信息总是比鹰眼地图的地理信息详细。鹰眼地图的可视范围广阔，可以看到当前主图周边概况。该功能是基于地图基本显示功能进行搭建的，鹰眼的实现流程如下：

1. 首先地图初始化 loadMap()脚本函数（位于 function.js 中）中加载鹰眼类。

➢ **加载鹰眼类代码：**

```
//鹰眼
var eo = new IMSEagleMap();
map.addControl(eo,false);
```

鹰眼类 IMSEagleMap 中的方法实现了鹰眼功能，该函数位于 IMSMaps.js 函数库中。

2. 通过在框架页面（MapFrame.htm）的客户端工具栏添加鹰眼的功能按钮代码，触发 onclick 事件的 EagleDisplay()函数来改变鹰眼的显示状态。

➢ **在 MapFrame.htm 中 id 为 cmdGroup0 的 div 层中添加如下代码：**

```
<img src="images/buttonx/button6.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button6_02.gif'"
    onmouseout="this.src='images/buttonx/button6.gif'"
    title="鹰眼" onclick="EagleDisplay()"/>
```

➢ **EagleDisplay()脚本函数代码：**

当 flag 值为 true：

```
function EagleDisplay(){
   map.eo.setViewStatus(flag)
   flag = !flag ;
 }
```

当 flag 值为 false：

```
function EagleDisplay() {
```

```
if(thisObj.eagleEye==null||typeof(thisObj.eagleEye)=="undefined")
{
    var eo = new IMSEagleMap();
    map.addControl(eo,false);
}
thisObj.eagleEye.setViewStatus(flag);
flag = !flag;
}
```

利用 setViewStatus()脚本函数改变鹰眼的显示状态，该函数位于 function.js 中。

3. 在客户端单击该功能按钮触发 onclick 事件，显示鹰眼，并通过鼠标的操作，将产生的参数传递到相应函数实现鹰眼功能。具体操作如下：

（1）用左键点击鹰眼中心红色边框按住左键不放，并拖动到你想要移动的位置，松开左键，即可实现鹰眼移动；鹰眼移动的同时将调用 IMSMaps 类中的 pntLocation 脚本函数以实现主图的同步移动。具体操作图见.NET 开发中的相应部分。

（2）选择主图的移动操作状态，按住鼠标左键不放并拖动主图，松开左键，同时将调用鹰眼类 IMSEagleMap 中的 eagleJumpingMap 脚本函数实现鹰眼的同步移动。具体操作图见.NET 开发中的相应部分。

注：IMSMaps 类中的 pntLocation()与 IMSEagleMap 类中的 eagleJumpingMap()都位于核心脚本库中，代码参见示例站点脚本库 IMSMaps.js 中的源码。

## 6.2.4 地图测量功能

地图测量功能实现包括距离测量、面积测量等操作。该功能是基于地图基本功能进行搭建的，该功能的实现流程如下：

1. 在地图初始化 loadMap 脚本函数（位于 function.js 中）中加载测量类。

➢ **加载测量类代码：**

```
var mo = new IMSMeasure();
mo.setDistanceUnits([103133.845,1000],["米","千米"]);
mo.setAreaUnits([10636589908.557,1000000],["平方米","平方千米"]);
map.addControl(mo,false);
```

其中 setDistanceUnits(valueArr,unitArr)和 setAreaUnits(valueArr,unitArr)各有两个参数：

- valueArr 参数:值数组。valueArr[0]记录当前坐标系下的比例度量，例如经纬度的地图 1 度代表的长度为 103133.845 米，那么 valueArr[0]值为 103133.845。如果地图为大地坐标系那么 valueArr[0]值为 1。valueArr[1]记录比例尺的比例，如果是米到千米的转换那么 valueArr[1]值为 1000。
- unitArr 参数:单位数组。是 valueArr 中值之间的转换度量的汉字表示，如“米”到“千米”或则“平方米”到“平方千米”。

2. 通过在框架页面（MapFrame.htm、menu.htm 页面）的客户端工具栏添加测量的功能按钮代码，触发 onclick 事件的 setCurOper 函数更改当前地图操作状态。

➢ **在 MapFrame.htm 中 id 为 cmdGroup0 的 div 层中添加如下代码：**

```
<img src="images/buttonx/button7.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button7_02.gif'"
    onmouseout="this.src='images/buttonx/button7.gif'"
    title="距离测量" onclick="map.setCurOper(3001)"/>
<img src="images/buttonx/button8.gif" class="btnToolsmenu"
    onmouseover="this.src='images/buttonx/button8_02.gif'"
    onmouseout="this.src='images/buttonx/button8.gif'"
    title="面积测量" onclick="map.setCurOper(3002)"/>
```

setCurOper 函数就是工具栏按钮被点击时触发的函数，位于脚本库 IMSMaps.js 内。具有一个输入参数 operType，代表按钮触发事件的代码，如在该功能中：

- 3001（IMSOperType.OpenMagnifier）代表距离测量；
- 3002（IMSOperType.OpenMagnifier）代表面积测量

➢ **setCurOper 函数测量部分代码：**

```
case 3001:
      if(this.mo==null){alert(this.gStr[1]);return;};
      this.mo.setMeasureMode("MeasureDistance","Polyline");
break;
case 3002:
      if(this.mo==null){alert(this.gStr[1]);return;};
      his.mo.setMeasureMode("MeasureArea","Area");
break;
```

在 setCurOper 函数中调用 setMeasureMode 函数（位于 IMSMaps.js 中），根据调用时的不同参数分别实现距离测量和面积测量功能。setMeasureMode()脚本函数的代码参见示例站点脚本库 IMSMaps.js 的源码。

3. 在客户端选择地图操作状态为距离测量或者面积测量状态，并在地图上做相应的画线或画区操作，通过鼠标监听事件把产生的参数传递到 setMeasureMode（）函数处理，实现测量功能。其客户端具体操作图见.NET 开发中的相应部分或者行业版的操作手册。

## 6.2.5 系统设置功能

系统设置功能主要实现显示设置和闪烁设置。显示设置包括还原显示、显示标尺、显示坐标点、跟随显示、缓冲区显示、显示动态标注和图像质量的控制；闪烁设置主要实现定位闪烁的次数、间隔和绘图颜色三方面的控制。系统设置为用户提供灵活的、便捷的交互式控制面板，用户可以根据自己的需求和喜好进行定制。该功能的实现流程如下：

1. 在客户端添加系统设置的功能按钮代码，通过触发 onclick 事件的 onSelectToolClick（位于 function.js 中）函数进行处理。

➢ **在 munu.htm 中添加如下代码：**

```
<img class="btncursor" onclick="top.onSelectToolClick('set',null)"
alt="显示设置" src="images/displayConfig.gif" align="middle"
border="0"><br>
```

```
    显示设置<br>
    <br>
<img class="btncursor"
onclick="top.onSelectToolClick('flashParaSet',null)" alt="闪烁设置"
src="images/flashConfig.gif" align="middle" border="0"><br>
    闪烁设置<br>
    <br>
```

➢ **onSelectToolClick()系统设置部分代码：**

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
 userParam = TargetPage;
 userActive = activeTool;
 switch (userActive) {
    case "set":
          map.setCurOper(1003);
          var url = "VectorPage/PageInfo.jsp?op=MapSetting&rnd=" +
Math.random();
          renderSelectResult(url);
     break;
    case "flashParaSet":
          map.setCurOper(1003);
          var url = "VectorPage/PageInfo.jsp?op=FlashSetting&rnd=" +
Math.random();
          renderSelectResult(url);
     break;
   default:
     break;
     }
}
```

2. 在 onSelectToolClick()中根据参数不同，调用 setCurOper 进行不同的操作状态设置，然后转到 VectorPage 文件夹下的 PageInfo.jsp 页面进行处理。

（1）**显示设置**：在 PageInfo.jsp 页面中，在显示设置的面板上设置好属性，点击提交按钮触发 onclick 事件的 SetMap() 脚本函数，该函数将参数传到 PageSetting.jsp 后台页面处理。

显示设置的面板如下图：

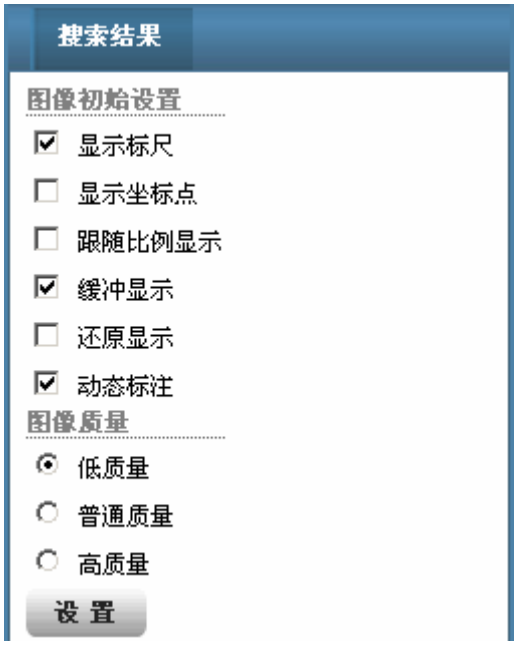



图 6-2-5-1 显示设置面板

- 表单提交处理函数 SetMap()代码：

```
function SetMap()
    {
        state="";
        for(var j=0;j<6;j++)
        {
        state += stateArr[j]+",";
        }
        state += stateArr[j];

 top.chargeFrame.location.href='PageSetting.jsp?op=mapSetting&mapS
tate='+state;
    }
```

- PageSetting.jsp 页面处理代码：

```
<%
    String operate = request.getParameter("op");
    if(operate.equals("mapSetting")){
……
    MapViewPort viewport = (MapViewPort)
session.getAttribute("viewport");
    String[] mapState;
    try{
        mapState =
request.getParameter("mapState").toString().split(",");
    }catch(Exception e){
        out.println("<script language='javascript'>alert('获取地图设
```

```
置信息失败。')</script>");
        return;
    }
    viewport.setImageDisplayParam(mapState);

    session.setAttribute("showScale",mapState[0]);
    session.setAttribute("showCoordinate",mapState[1]);
    session.setAttribute("followScale",mapState[2]);
    session.setAttribute("bufferShow",mapState[3]);
    session.setAttribute("originalFlag",mapState[4]);
    session.setAttribute("dynamicMark",mapState[5]);
    session.setAttribute("imageQuality",mapState[6]);
   %>
        <script type="text/javascript">
            top.onDisplayToolClick('reflash');
        </script>
<%
}else{
    out.println("<script language='javascript'>alert('请求错误，请重试。');</script>");
}
%>
```

对显示设置处理，先是接收显示设置参数 mapState，然后再调用 GIS 服务端 CMapView 类库的 SetImageDisplayParam 方法把从客户端设置的控制参数存到 GIS 服务器中，由此实现了显示设置功能。

（2）**闪烁设置**：在 PageInfo.jsp 页面中，在闪烁设置的面板上设置好属性，点击提交按钮，触发 onclick 事件的 SetFlash() 脚本函数进行处理。

闪烁设置的面板如下图：

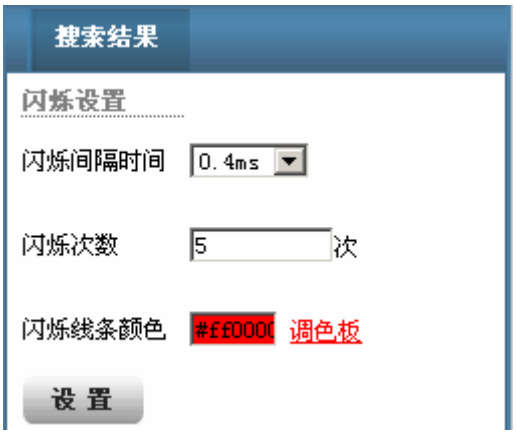



图 6-2-5-2 闪烁设置面板

➢ **表单提交处理函数 SetFlash()代码：**

```
function SetFlash()
{
```

```
    var times = document.getElementById("times").value;
    var interval = document.getElementById("flashInterval").value;
    var color = document.getElementById("flashColor").value;

    if(times==""||interval==""||color=="")
    {
        alert("参数不可为空")
        return;
    }

    if(times!="")
    {
        var reg = /^[0-9]*$/
        if(!reg.test(times))
        {
            alert("闪烁次数需要输入整型数据！")
            return;
        }
    }
    if(interval!="")
    {
        var reg = /^\d+(\.\d+)?$/
        if(!reg.test(interval))
        {
            alert("闪烁间隔需要输入整型或浮点型数据！")
            return;
        }
    }
    top.flashTimes = times;
    top.flashInterval = interval;
    top.flashColor = color;
    alert("闪烁参数设置成功！");
}
```

## 6.2.6 地图搜索功能

地图搜索主要为关键字搜索，能够将搜索到的分类信息以不同的图标进行展示。这些信息一般存于 SQL 数据库。该功能是基于地图基本功能进行搭建的，实现流程如下：

1. 在 MapFrame.htm 页面中 id 为 hmenuTitle 的 div 层中添加地图搜索的链接。

- **地图搜索链接的代码：**

```
<a href="javascript:void(0);" class='a1'
onclick="funcChange(1,this);return false;">地图搜索</a>
```

- **funcChange()的代码：**

```
function funcChange(type,obj){
    var aarr=obj.parentNode.getElementsByTagName("a");
    for(var i=0;i<aarr.length;i++){
        aarr[i].className='';
    }
    obj.className='a1';
    obj.blur();
    type=parseInt(type);
 for(var i=0;i<hdivs.length;i++){
     Element.hide(hdivs[i]);
     }
    $("zdims").innerHTML="";
 switch (type){
     case 1:{
         Element.show("lsForm");
         loadMap(0,IMSMapConfig);
         break;}
     case 2:{
         Element.show("bsForm");
         loadMap(0,IMSMapConfig1);
           break;}
     case 3:{
         Element.show("nsForm");
         loadMap(0,IMSMapConfig);
         break;}
     default:{
         Element.show("lsForm");}
 }
}
```

这是切换标题栏功能的脚本函数，位于脚本库 function.js 中。

在更改标题栏模块的地图搜索功能中调用了 loadMap(1,IMSMapConfig1)，最后用 GetMap.jsp 页面向 GIS 服务器取图并进行处理。loadMap 函数位于脚本函数库 function.js 中，其参数 mapType 值为 1 表示请求取矢量图，mapConfig 值为 IMSMapConfig 表示是基本地图配置类文件（详细配置位于脚本函数库 IMSMapConfig.js 中）。其后的取图显示流程见地图图像显示流程，loadMap 函数代码见地图图像显示功能实现流程 5。

注：开始选择地图文档进入到主框架页面时，显示矢量地图，即可进行地图关键字搜索；如果当前不是初始默认的地图搜索状态，可以点击标题栏的“地图搜索”链接进入。

2. 在 MapFrame.htm 页面中 id 为 lsForm 的 div 层中添加关键字搜索的文本框与按钮。

- **关键字搜索的文本框与按钮的代码为：**

```
<div id="lsForm" class="hmenudiv">
  <table>
    <tr>
```

```
      <td><input class="inp1" name="keyword" id="keyword" /></td>
      <td>
          <select id="searchtype" name="searchtype" style="display:
none;">
              <option value="-1">所有类型
          </select>
      </td>
      <td>
          <input name="lsFormSub" type="button" value="搜  索"
class="btnfmt"
 onclick="SearchByWords(document.getElementById('keyword').value)"
/>
     </td>
     <td id="showORhideType" style="font-size: 12px; color: blue;
cursor: pointer" onclick="showORhideType()">
        <u>显示搜索类型选项</u>
     </td>
   </tr>
  </table>
</div>
```

SearchByWords()脚本函数就是地图关键字搜索时客户端入口函数，位于脚本库IMSPathSearch.js 中。

3. 在地图关键字搜索按钮的 onclick 事件中调用 SearchByWords()脚本函数，然后转到 Maps.jsp 页面处理。

➢ SearchByWords()代码：

```
function SearchByWords(key){
   if(key.trim() == "")    {
      alert("请输入关键字！");
      return;
   }
   var type = document.getElementById("searchtype").value;
   if(!flag_searchType)
    type = "-1";

   var url = document.getElementById("rsFrame").src;
 if (map.isIE) {
    if (url != "VectorPage/Result.htm") {
       document.getElementById("rsFrame").src =
"VectorPage/Result.htm";
    }
 } else {
    var u = url.split("/");
    if (u[u.length - 1] != "Result.htm") {
```

```
            document.getElementById("rsFrame").src =
    "VectorPage/Result.htm";
        }
    }
        var dataurl =  "Maps.jsp?";
        var data =
    "_method=QueryByWords"+"&key="+encodeURI(key.trim())+"&type="+enco
    deURI(type);
        AJAXRequest(dataurl,true,"POST",data,cbQuery);
        loadBar(1);
    }
```

在该脚本函数中设置好查询结果的显示位置是 ID 值为 rsFrame 的子框架，处理结果显示的页面为 Result.htm，然后通过 AJAX 方式转到 Maps.jsp 后台页面处理，返回结果到相应位置。

4. 在服务器端处理页面 Maps.jsp 中添加相关代码，实现查询数据库并将信息返回到客户端功能。

➢ **服务器后台页面 Maps.jsp 信息查询部分代码：**

```
<jsp:useBean id="dbQuery"
class="com.mapgis.ims.see.util.CGISDataQuery" scope="session">
</jsp:useBean>
String res = "";                  //查询结果字符串
ChineseParse cpObj = null;          //中文分词对象
String strPath = "";             //中文字典路径
int iMapType = 2;                  //图层类型标识符(0:栅格流图层;1:矢量图
层;2:栅格流和矢量图层)
CBusCom dbCar = null;
String changerlt_xmlStr = "";
CBusData dbCarQuery = null;
String strMethod = request.getParameter("_method");
CMapConfig webconfig = new CMapConfig();
CMapLayerSetting mapLayer = null;
MapViewPort viewport = null;

if ( strMethod == "" ){
    out.println( "The context is not set correct in
ProcessRequest()." );
}
else{
  int iMethod = StrToInt(strMethod);

  switch(iMethod)
       {
      case 3://模糊查询
```

```
            String key =String.valueOf(request.getParameter("key"));
            String type = String.valueOf(request.getParameter("type"));
                //中文分词
            strPath = application.getRealPath("ChineseDictionary.txt");
                cpObj = new ChineseParse(strPath);

                ChineseWordUnit[] t_arryWords = cpObj.ParseChinese(key);
                int t_iArrLen = t_arryWords.length;
                String[] strQueryArr = new String[t_iArrLen];

                for (int i = 0; i < t_iArrLen; i++) {
                   if (t_arryWords[i] != null) {
                       strQueryArr[i] = t_arryWords[i].getWord();
                   } else {
                       break;
                   }
               }
               //关键字查询
               dbQuery.setSelectLayer(type);
               res = dbQuery.GetDataByCondition(strQueryArr);
               response.setContentType("text/html;charset=gb2312");
               out.clearBuffer();
               out.println(res);
                  break;
    }
}
```

在 Maps.jsp 页面中，用 javaBean 来实现查询操作。通过脚本函数 StrToInt()将查询类型转换为操作符，然后进行查询处理，主要调用 CGISDataQuery 类库的 GetDataByCondition()方法实现关键字查询功能。

➢ StrToInt()脚本函数部分代码：

```
public int StrToInt(String strMethod)
{
 int iMethod = 0;
 if(strMethod == null)
    return iMethod;
if(strMethod.equals("QueryByWords") == true)
 {
    iMethod = 3;
    return iMethod;
  }
  ……
  }
```

在该脚本函数中将查询类型为 QueryByWords 转换成操作符 3，便于判断查询类型。

5. 在 MapFrame.htm 页面的输入框中输入关键字，点击搜索按钮调用 SearchByWords 函数，然后 SearchByWords 函数再调用 AJAXRequest 函数将关键字等相关参数以 ajax 异步请求的方式提交到服务器端页面(Maps.jsp)，服务器端页面处理完成后，将结果返回由调用 AJAXRequest 时设定好的方式进行显示。其客户端具体操作如图：




图 6-2-6-1 步骤 1 输入查询关键字




图 6-2-6-1 步骤 2 查询结果显示

## 6.2.7 图层控制功能

图层控制主要实现图层显示和图层状态控制功能，配合各种操作使用，以满足用户的需求。

图层列表及状态显示效果如下图：

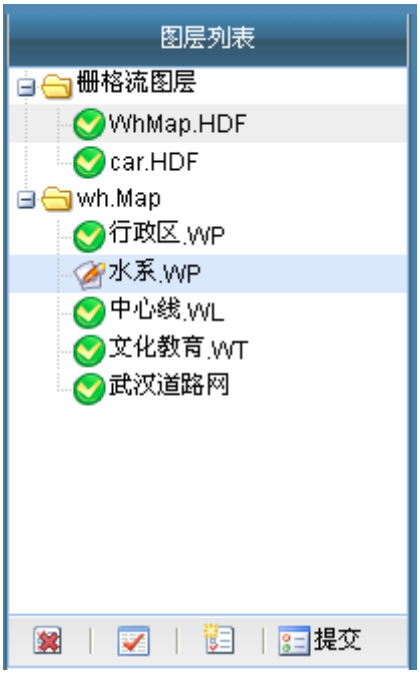



图 6-2-7-1 图层列表及状态显示效果

矢量图层显示功能的实现流程：

1. 在 menu.htm 中添加显示图层列表的客户端代码。

➢ **图层列表的客户端代码：**

```
<tr onclick="onMenuClick(this)">
      <td width="4" bgcolor="#eaeaea">
      </td>
      <td class="case_lr" width="98%">
           图层控制
      </td>
      <td width="4" bgcolor="#eaeaea">
      </td>
</tr>
<tr>
      <td width="4" bgcolor="#eaeaea">
      </td>
      <td valign="top">
         <div id="layerTree" oncontextmenu="return false"
style="height:300px"></div>
       </td>
       <td width="4" bgcolor="#eaeaea" height="30px">
       </td>
</tr>
```

2. 在 menu.htm 中添加对脚本库 modules.js 和样式表文件 tree.css 的引用。

➢ **引用的代码：**

```
<link type="text/css" href="css/tree.css" rel="stylesheet" />
<script type="text/javascript" src="jslib/ext-lp.js"></script>
```

```
<script type="text/javascript" src="jslib/modules.js"></script>
```

3. modules.js 是图层列表树的脚本库，通过客户端 div 层为 layerTree 的 id，把列表树和相关按钮加载到客户端中，并转到 Maps.jsp 服务器页面处理。

➢ **modules. js 部分代码：**

```
//全局变量
var urls = {
    'showLayerTree' : 'Maps.jsp?_method=GetMapLayerList',
    'setLayerStatus': 'Maps.jsp?_method=UpdateMap'
}
var mapUrlCfg = new top.IMSMapConfig();
var tree;
/* 图层树 */
Ext.namespace("MLayerTree");
function MLayerTree(){
    var tree;
    var ctx;
    var selectedNode;
    refreshTree = function(){
        tree.body.mask('加载图层中...', 'x-mask-loading');
        tree.root.reload();
        tree.root.collapse(true, true);
        setTimeout(function(){
            tree.body.unmask();
            tree.root.expand(true, true);
        }, 1000);
    }
……
  tree = new Ext.tree.TreePanel({
        root: new Ext.tree.AsyncTreeNode({
            expanded: true,
            text: 'tree',
            allowDrag:false,
            allowDrop:false
        }),
        renderTo:'layerTree',
        id: 'layerTree2',
        height : 300,
        iconCls: 'nav',
        animate: true,
        //title: '图层列表',
        loader: new Ext.tree.TreeLoader({
            requestMethod :"GET",
            dataUrl: urls['showLayerTree'],
```

```
            listeners:{
                "load":function(o, node, response){
                if(response.responseText.indexOf("showCtx")==-1)
                    tree.getBottomToolbar().hide();
                }
            }
        }),
    ……
    }
```

- Maps.jsp 中图层列表部分代码：

```
int iMethod = StrToInt(strMethod);
  switch(iMethod)
  {
     case 1://获取图层信息列表
        mapLayer = new CMapLayerSetting();
            //获取图层类型
        iMapType =
Integer.parseInt(webconfig.getInitParameter("MapType").toString())
;
       CLayerInformation[] layerInfo = null;
       String HDFName =
String.valueOf(webconfig.getInitParameter("HDFName"));
     if(iMapType != 0){
       viewport =(MapViewPort)session.getAttribute("viewport");
       if(viewport == null){
          out.println("<script type='language'>alert('工程加载失败。
')</script>");
          return;
        }
       try{
           layerInfo =viewport.getMapObj().getLayerInformationArr();
           }catch(Exception ex){
                out.println("<script type='language'>alert('获取图层
信息失败。')</script>");
                return;
                }
      }
     //获取图层列表
     out.clearBuffer();
     res =mapLayer.getLayerListByJSON(HDFName,layerInfo,iMapType);
     out.println(res);
     break;
}
```

在 Maps.jsp 页面中，将从客户端传过来的操作类型 GetMapLayerList 转换为操作符 1，然后获取图层列表，即调用 getLayerListByJSON()方法获取图层列表并返回结果。

图层状态控制功能是在图层显示的基础上实现的，其流程为：

1. 在图层列表中通过右键单击或左键单击图层列表下方的按钮改变图层的状态，功能函数在脚本库 modules.js 中。

右键和图层列表下方的图层状态控制按钮呈现效果如下图：

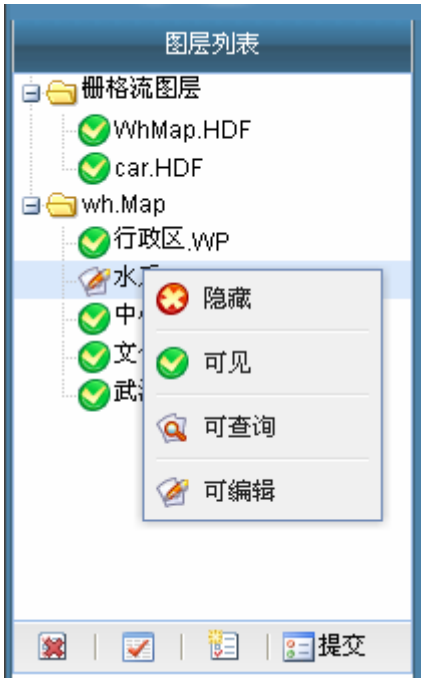



图 6-2-7-2 右键与列表下方的状态控制按钮

➢ **图层列表下方按钮的代码：**

```
bbar: [{
            icon: 'icons/hideAll.gif',
            minWidth: 35,
            cls: 'x-btn-icon',
            tooltip: '<span>隐藏全部图层</span>',
            handler: hideAll
        }, '-', {
            icon: 'icons/showAll.gif',
            minWidth: 35,
            cls: 'x-btn-icon',
            tooltip: '<span>显示全部图层</span>',
            handler: showAll
        }, '-', {
            icon: 'icons/activeAll.gif',
            minWidth: 35,
            cls: 'x-btn-icon',
            tooltip: '<span>查询全部图层</span>',
            handler: activeAll
```

```
        }, '-',{
            text: '提交',
            icon: 'icons/list-items.gif',
            cls: 'x-btn-text-icon',
            tooltip: '<span>更新图层状态</span>',
            handler: submitLayerStatus
        }]
```

2. 点击提交按钮，触发事件的 submitLayerStatus 脚本提交更改的图层状态信息。即通过“setLayerStatus” url 标记传入参数 params(_method=UpdateMap)转到 Maps.jsp 服务器页面处理，并在地图容器中刷新地图。

➢ submitLayerStatus()代码：

```
submitLayerStatus = function(){
        var root = tree.getRootNode();
        var folder = root.childNodes;
        var nodes = folder[folder.length-1].childNodes;
        var params = '_method=UpdateMap';
        for (var i = 0; i < nodes.length; i++) {
            var leafEl = nodes[i].getUI().getEl();
            var leaves = leafEl.getElementsByTagName("div");
            var leaf = Ext.get(leaves[0]);
            params += "&" + nodes[i].id + "=" + getLayerIcon(leaf);
        }
        tree.body.mask('图层状态提交中...', 'x-mask-loading');
        Ext.Ajax.request({
            url: urls['setLayerStatus'],
            success: function(response){
                //top.map.setCurOper(top.IMSOperType.Refresh);
                tree.body.unmask();
            },
            params: params
        });
        }
```

➢ Maps.jsp 中更新地图文档部分代码：

```
case 2://更新图层状态
    mapLayer = new CMapLayerSetting();
    viewport = (MapViewPort)session.getAttribute("viewport");
    if (viewport == null) {
        out.println("<script type='language'>alert('工程加载失败。
')</script>");
        return;
        }
```

```
    //获取图层个数
    int iLayernum = viewport.getMapObj().getLayerNumber();
    String[] strLayerStatus = new String[iLayernum];
    int selectcount = 0;   //图层可查状态的个数
    String strSelectIndex = "";
    for(int i=0; i<iLayernum; i++){
         strLayerStatus[i] = request.getParameter("layer"+i);
if(strLayerStatus[i].equalsIgnoreCase("selectable")||strLayerStat
us[i].equalsIgnoreCase("editable")){
                strSelectIndex += String.valueOf(i) + ",";
                selectcount++;
            }
        }
   //保存可编辑图层序号
   int editableLayer
=Arrays.asList(strLayerStatus).indexOf("editable");
   if(editableLayer >= 0){
session.setAttribute("editLayerIndex",editableLayer);
            }else{
             session.setAttribute("editLayerIndex",null);
            }
   mapLayer.updateMap(strLayerStatus,viewport);
  //单图层或者多图层条件查询判断
 if (selectcount > 1){
    session.setAttribute("singleLayerQuery",
String.valueOf("false"));
    session.setAttribute("NoselectalbeIndex","false");
session.setAttribute("selectLayerIndex",strSelectIndex);
          }else if(selectcount == 1){
session.setAttribute("singleLayerQuery",String.valueOf("true"));
            int selectStatusIndex =
Arrays.asList(strLayerStatus).indexOf("selectable");
            if(selectStatusIndex < 0)
                selectStatusIndex =
Arrays.asList(strLayerStatus).indexOf("editable");

session.setAttribute("selectLayerIndex",selectStatusIndex);
            session.setAttribute("NoselectalbeIndex","false");
          }else{
            session.setAttribute("NoselectalbeIndex","true");
          }
   break;
```

将从客户端传过来的操作类型 UpdateMap 转换为操作符 2，然后进行更新图层状态处理，即调用 CMapLayerSetting 类库中的 updateMap()方法更新图层状态，并在地

图容器中刷新地图。

3. 以修改地图图层状态为例，客户端具体操作图见.NET 开发中的相应部分或者行业版操作手册。

## 6.2.8 地图查询功能

地图查询功能分为简单查询和条件查询。简单查询即基本查询，包括点击查询、拉框查询、圆查询、线查询和多边形查询五种；条件查询包括属性条件查询、拉框条件查询、画圆条件查询和多边形条件查询四种。这些功能都是在矢量地图服务的地图显示基础上实现的，所以对地图显示功能的透彻理解是做该开发的前提。地图查询功能还涉及到闪烁和编辑功能，即查询到的点、线、区要素的定位显示和编辑。

地图查询功能实现流程大致为：首先通过客户端工具栏触发客户端脚本 Function.js，在客户端脚本 Function.js 中将各种查询和处理页面关联，最后调用后台页面 SelectAttribute.jsp 处理后生成查询结果。

地图查询的结果如下图：

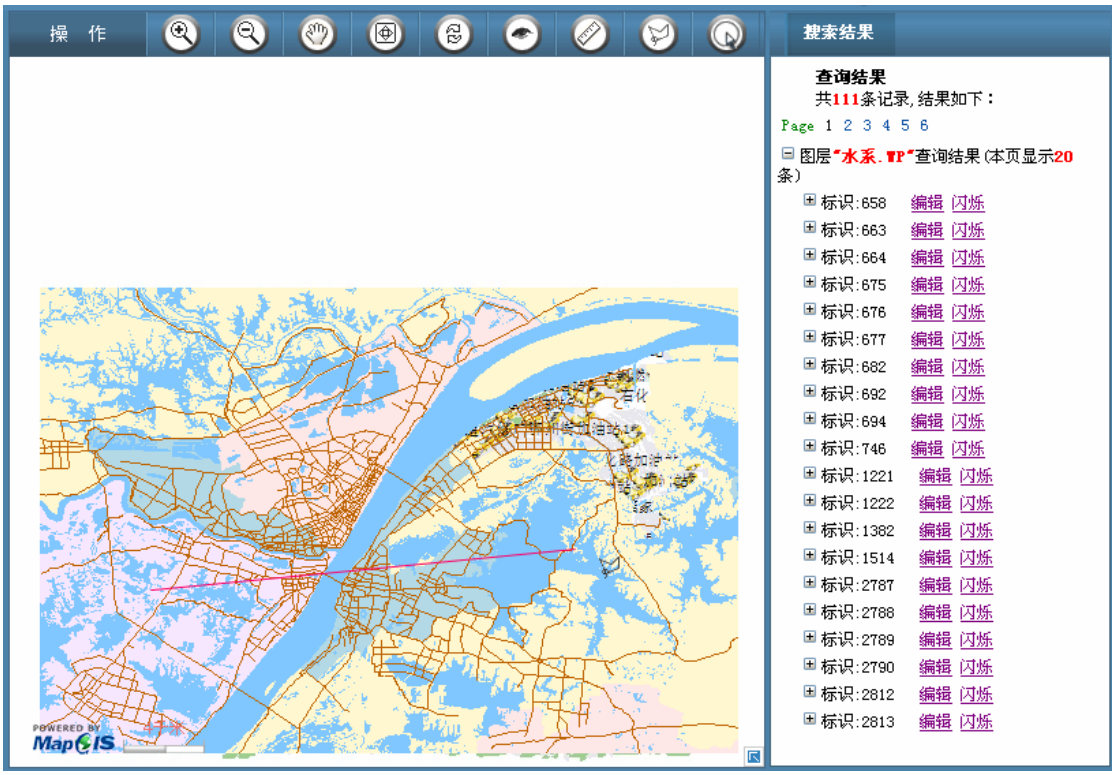



图 6-2-8-1 地图查询结果

具体实现流程为：

1. 在地图初始化 loadMap 脚本函数（位于 function.js 中）中加载图形搜索类和操作状态类对象。若这一步在地图显示中已经加载，则可跳过。

➢ **加载图形搜索类和操作状态类的代码：**

```
//图形搜索类
pro = new IMSSearch(map);
map.addControl(pro,false);
//操作状态
```

```
var os = new IMSOperStatus();
map.addControl(os,false);
```

2. 在客户端工具栏（Menu.htm、MapFrame.htm 页面）处添加地图查询功能按钮，进行事件编写及绑定。

   Menu.htm 页面中地图查询的功能按钮呈现效果如下图：




图 6-2-8-2 简单查询和条件查询的按钮效果

- **地图查询功能按钮的代码：**

```
 <img class="btncursor"
onclick="top.onSelectToolClick('SelectByHitting','Point','Search.j
sp')" alt="点击查询" src="images/selectByHitting.gif" align="middle"
     border="0"><br>
      点击查询<br>
<img class="btncursor"
onclick="top.onSelectToolClick('SelectByPolyline','Polyline','Sear
ch.jsp')" alt="线查询" src="images/selectByPolyline.gif"
align="middle" border="0"><br>
     线查询<br>
<img class="btncursor"
onclick="top.onSelectToolClick('SelectByRect','Rect','Search.jsp')
" alt="拉框查询" src="images/selectByRect.gif" align="middle"
     border="0"><br>
     拉框查询<br>
```

```
<img class="btncursor"
onclick="top.onSelectToolClick('SelectByCircle','Circle','Search.j
sp')" alt="圆查询" src="images/selectByCircle.gif" align="middle"
     border="0"><br>
     圆查询<br>
<img class="btncursor"
onclick="top.onSelectToolClick('SelectByPolygon','Polygon','Search
.jsp')" alt="多边形查询" src="images/selectByPolygon.gif"
     align="middle" border="0"><br>
     多边形查询<br>
<img class="btncursor"
onclick="top.onDisplayToolClick('Condition',null)" alt="属性条件查询
" src="images/selectByCondition.gif" align="middle"
     border="0"><br>
     属性条件查询<br>
<img class="btncursor"
onclick="top.onSelectToolClick('SelectByRectAndAtt','Rect','Condit
ionGenerator.jsp')" alt="拉框条件查询"
src="images/selectByPolylineAndConditon.gif" align="middle"
border="0"><br>
     拉框条件查询<br>
<img class="btncursor"
onclick="top.onSelectToolClick('SelectByCircleAndAtt','Circle','Co
nditionGenerator.jsp')" alt="画圆条件查询"
src="images/selectByCircleAndCondition.gif" align="middle"
border="0"><br>
     画圆条件查询<br>
<img class="btncursor"
onclick="top.onSelectToolClick('SelectByPolygonAndAtt','Polygon','
ConditionGenerator.jsp')" alt="多边形条件查询"
src="images/selectByPolygonAndCondition.gif" align="middle"
border="0"><br>
     多边形条件查询<br>
```

3. 在地图查询各功能按钮的 onclick 事件调用了 onSelectToolClick 和 onDisplayToolClick 脚本函数进行处理，这些函数位于脚本库 function.js 中，这两个函数用户可以自定义编写，为不同的查询（点击，拉框，画圆、属性条件等）编写不同的自定义的处理方法。

- **onSelectToolClick()地图查询部分代码：**

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
    userParam = TargetPage;
    userActive = activeTool;
    switch (userActive) {
      case "SelectByHitting":
```

```
  case "SelectByHittingAndAtt":
    map.setCurOper(2001);
    pro.setSelectByHitting(map.logicXlength / 400, submitUrl);
    break;
  case "SelectByPolyline":
    map.setCurOper(2005);
    pro.setSelectByLine(map.logicXlength / 400, submitUrl);
    break;
  case "SelectByRect":
  case "SelectByRectAndAtt"://拉框条件查询
    map.setCurOper(2002);
    pro.setSelectByRect(submitUrl);
    break;
  case "SelectByCircle":
  case "SelectByCircleAndAtt"://画圆条件查询
    map.setCurOper(2003);
    pro.setSelectByCircle(submitUrl);
    break;
  case "SelectByPolygon":
  case "SelectByPolygonAndAtt"://多边形条件查询
    map.setCurOper(2004);
    pro.setSelectByPolygon(submitUrl);
          break;
    }
}
```

上面的函数就是将各种查询和处理方法关联。activeTool 参数：代表查询的类型；drawAction 参数：代表客户端鼠标的操作类型； TargetPage 参数：代表处理该查询的服务器页面。

➢ **onDisplayToolClick()属性条件查询部分代码：**

```
function onDisplayToolClick(con) {
 var type = con;
 switch (type) {
  case "Condition"://条件查询
    map.activeTool = type;
    subParam = "";
    var url = "VectorPage/ConditionGenerator.jsp?op=" + type + "&rnd="
+ Math.random();
    renderSelectResult(url);
    break;
    }
}
```

上面的函数就是将属性条件查询和处理方法关联。con 参数的值为 Condition，代表属性查询的类型。该函数将转到 VectorPage 文件夹下的服务器后台页面

ConditionGenerator.jsp 中，对属性条件查询设置具体的查询条件，然后再将条件参数等传到服务器后台页面 Search.jsp 中处理。

ConditionGenerator.jsp 页面条件查询的条件编辑面板如下图：

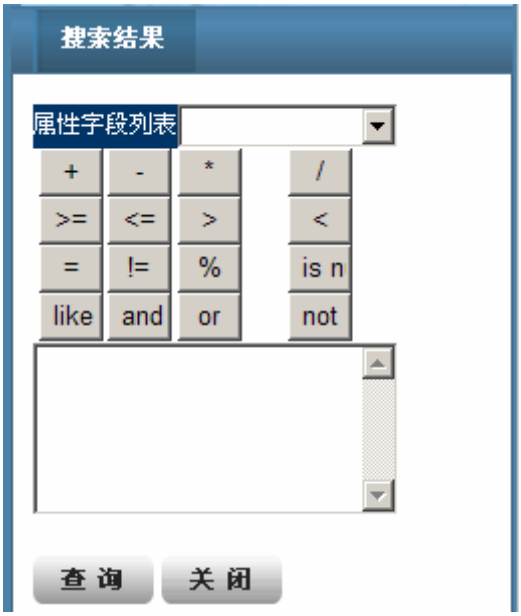



图 6-2-8-3 条件查询的条件编辑面板

注：简单查询将转到 Search.jsp 页面处理，条件查询则转到 ConditionGenerator.jsp 页面设置条件后再转到 Search.jsp 页面处理。以下第 4、5 步骤为非属性条件查询的具体流程。

4. 在 onSelectToolClick 函数中首先根据参数调用 setCurOper 函数设置地图操作状态为相应类型，再在 setCurOper 函数中调用 setQueryMode 函数，根据 drawAction 参数（客户端鼠标在地图上的操作类型）添加事件监听并调用相应的脚本处理函数，在处理函数中获取相应的参数 param（为 submitUrl 函数的参数）。

- **setCurOper()地图查询部分代码：**

```
//设定当前操作状态
    this.setCurOper=function(operType)
    {
        this.operType = parseInt(operType);
        this.clearOper();
        switch(this.operType)
        {
            case 2001:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByHitting","Point");
            break;
            case 2002:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByRect","Rect");
            break;
            case 2003:
```

```
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByCircle","Circle");
            break;
            case 2004:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByPolygon","Area");
            break;
            case 2005:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByLine","Polyline");
            break;
            default:
                this.activeTool = "Pan";
                this.drawAction = "Pan";
            break;
        }
    };
```

- setQueryMode()代码：

```
this.setQueryMode = function(activeTool,drawAction)
    {
        thisObj.activeTool = activeTool;
        thisObj.drawAction = drawAction;
        switch(drawAction){
            case "Point":
                if(this.listenerPoint){
thisObj.addEventListener("mouseup",thisObj.so.selectByHitting ,
false);
                    this.listenerPoint = false;
                }
                break ;
            case "Circle" :
                if(this.listenerCircle){
thisObj.addEventListener("mouseup",thisObj.so.selectByCircle,
false);
                    this.listenerCircle = false;
                }
                break ;
            case "Rect" :
                if(this.listenerRect){
        thisObj.addEventListener("mouseup",thisObj.so.selectByRect,
false);
                    this.listenerRect = false;
                }
```

```
                break ;
            case "Area" :
                if(this.listenerPolygon){
 thisObj.addEventListener("dblclick",thisObj.so.selectByPolygon,
false);
                    this.listenerPolygon = false;
                }
                break ;
            case "Polyline":
                if(this.listenerPolyline){
 thisObj.addEventListener("dblclick",thisObj.so.selectByLine,
false);
                    this.listenerPolyline = false;
                }
                break;
            default:
                break ;
        }
      };
```

- 以点击查询为例，setSelectByHitting()代码：

```
this.setSelectByHitting = function(ioffset,dealFun)
   {
      this.offset[0] = ioffset;
      this.dealFunArr[0] = dealFun;
   };
```

- 以点击查询为例，selectByHitting()代码为：

```
  this.selectByHitting= function()
   {
       if(thisObj.activeTool!="SelectByHitting"){

thisObj.removeEventListener("mouseup",thisObj.so.selectByHitting ,
false) ;
          thisObj.so.listenerPoint =true;
          return ;
       }
       var dx = thisObj.so.offset[0];
       var X = Math.max(thisObj.mdLogicX,thisObj.muLogicX);
       var Y = Math.max(thisObj.mdLogicY,thisObj.muLogicY);
       //用于计算网格编号
       var startX=X-dx;
       var startY=Y-dx;
       var endX=X+dx;
```

```
        var endY=Y+dx;
        var
param="centerX="+X+"&centerY="+Y+"&radius="+dx+"&GridCodes="+thisO
bj.getGridCodes(startX,startY,endX,endY)+"&lev="+(thisObj.inf[2]+1
)+"&sType=SelectByHitting";
        if(thisObj.so.dealFunArr[0]!=null)
            thisObj.so.dealFunArr[0](param);
        };
```

通过鼠标事件获取所点击的 x、y 坐标值等参数值，调用 dealFunArr[0]，即 submitUrl 函数，把参数组 param 设置为 submitUrl 函数的参数。

5. 然后分别调用自定义函数处理。各函数中的 submitUrl 参数是一个对象，它就是对各种查询的处理方法。由 submitUrl()处理将查询参数传到相应的服务器页面处理，并通过 renderSelectResult 函数将结果返回到指定的 ID 为 rsFrame 的子框架显示查询结果。

➢ **submitUrl()代码：**

```
function submitUrl(param) {
//记录组合条件查询的范围
subParam = param;
var url = "VectorPage/" + userParam + "?" + param + "&op=" + userActive;
if (userActive != "SelectByRectAndAtt" || userActive !=
"SelectByCircleAndAtt" || userActive != "SelectByPolygonAndAtt") {
    renderSelectResult(url);
    }
}
```

该函数是各查询的处理方法，这里的 param 是在上面所介绍的核心脚本库 IMSMaps.js 中生成的。各种查询相应的名值对参数如下：

点击查询：centerX="+X+"&centerY="+Y+"&radius="+dx

拉框查询：startX="+startX+"&startY="+startY+"&endX="+endX+"&endY="+endY

画圆查询：centerX="+centerX+"&centerY="+centerY+"&radius="+radius

线查询：XArray="+Xarray+"&YArray="+Yarray

多边形查询：XArray="+Xarray+"&YArray="+Yarray

用户根据名值对参数获取各查询的处理方法，并在自己的处理页面中获取相应的值。

➢ **renderSelectResult()代码：**

```
function renderSelectResult(url) {
var obj = document.getElementById("rsFrame");
switchdiv(1);
obj.src = "";
obj.src = url;
var locPntDiv = document.getElementById("locPntDiv");
if (locPntDiv != null) {
    locPntDiv.innerHTML = "";
    locPntInfoArr = new Array();
```

```
    }
}
```

该函数用于将处理结果显示到一个叫做 rsFrame 的 iframe 中，用户可以根据自己的需要修改。

6. 服务器后台页面 Search.jsp，用于处理各种查询。这个页面根据不同的查询类型来接收不同的客户端参数，分别进行各项查询的处理，并将查询结果传到客户端。下面以点击查询为例进行分析。

➢ **点击查询处理部分代码为：**

```
//判断是否有可查图层
if (firstSelect == null) {
  switch (op) {
      case 1://点击查询
        try {
             xmin = Double.parseDouble(request
                         .getParameter("centerX"));
             ymin = Double.parseDouble(request
                         .getParameter("centerY"));
                  // 由脚本库设置，3个屏幕像素
             HitRadius = Double.parseDouble(request.getParameter(
                         "radius").toString());
              } catch (Exception e) {
                  out.println("<script
language='javascript'>alert('获取点查询信息失败。');</script>");
                  return;
              }
    CPointSelectClass pointSelectObj = new CPointSelectClass();
    pointSelectObj.setDistance(HitRadius);
    CPoint2D dotLogMin = new CPoint2D();
    dotLogMin.setPointX(xmin);
    dotLogMin.setPointY(ymin);
    pointSelectObj.setDot(dotLogMin);
    pointSelectObj.setMapID(viewport.getMapID());
    pointSelectObj.setLayerIndex(layerQueryIndex);
// 调用点击查询函数,输入参数为上面赋值的点击查询对象
    flag = viewport.SelectByHitting(pointSelectObj);
    if (flag < 0) {
        out.println("<script language='javascript'>alert('查询失败。
');</script>");
        return;
        }
    getpage.setPntSelect(pointSelectObj);
    getpage.setQueryType(CQueryType.SelectByHitting);
    session.setAttribute("page", getpage);
```

```
 break;
    }
}
// 获取查询结果显示页面标识
        long currentPage = 1;
        String pageIndexstr;
        try {
            pageIndexstr = request.getParameter("pageIndex");
            getpage = (CGetOnePage) session.getAttribute("page");
        } catch (Exception e) {
            out.println("<script language='javascript'>alert('获取分
页信息失败。');</script>");
            return;
        }
        // 第一次显示查询结果
        if (pageIndexstr != null) {
            currentPage = Integer.parseInt(pageIndexstr);
            try {
                viewport.GetOnePageResult((int) currentPage,
getpage);
            } catch (Exception e) {
            selectResultObj = null;
            out.println("<script language='javascript'>alert('对不
起!服务器忙,请重新查询。');top.mapFrame.loaderObj.setFree()</script>");
                return;
            }
        }
        selectResultObj = viewport.getLastSelectResult();
        if (selectResultObj == null) {
            out.println("<script language='javascript'>alert('获取查
询结果失败，请重新操作。');</script>");
            return;
        }
        long totalCount = selectResultObj.getTotalFCount();
        long PageCount = 0;
        // 将查询结果的页面信息传入客户端
        if (totalCount % 30 == 0)
            PageCount = totalCount / 30;
        else
            PageCount = totalCount / 30 + 1;
        // 查询时不生成图片
        layerSelectRlt = null;
        layerSelectRlt = selectResultObj.getlayerSelInfo();
        session.setAttribute("selectResultObj", selectResultObj);
```

在点击查询处理中，首先接收客户端传过来的参数，其中 centerX 是点击的 x 坐标，centerY 是点击的 y 坐标，radius 是点击查询的查询半径；然后调用 GIS 服务器类库中的 SelectByHitting()方法来查询；最后将调用 getLastSelectResult()方法取出查询结果，调用 getlayerSelInfo()方法分图层取出结果。

以上只是部分代码，完整代码参见示例站点中的 Search.jsp 页面源码。

7. 以点击查询为例，客户端具体操作图见.NET 开发中的相应部分或者行业版操作手册。

以上步骤实现了地图查询功能，下面介绍如何实现闪烁和编辑查询结果。闪烁和编辑功能涉及到的客户端操作较多，需要调用核心脚本库的方法，实现起来比较复杂。其实现流程为：

1. 首先在 Search.jsp 页面客户端添加相应的代码。

➢ **闪烁和编辑功能客户端代码：**

```
<span style="TEXT-DECORATION: underline; cursor:
pointer;"onclick="edit(<%=j%>,<%=i%>)">编辑</span><%}} %>
<span style="TEXT-DECORATION: underline; cursor: pointer;"
onclick="jumpToFlash(<%=j%>,<%=i%>,<%=layerSelectRlt[i].getLayerI
ndex()%>)">闪烁</span>
```

在上面的代码中，脚本函数 edit()进行编辑功能处理，jumpToFlash ()进行定位闪烁功能处理。这两个脚本函数都位于 Search.jsp 页面中。

2. **定位闪烁功能**：主要通过脚本函数 jumpToFlash ()处理，将相关参数传到 VectorPage 文件夹下的 Flash.jsp 页面进行处理，与 GIS 服务器端交互完成定位闪烁功能。

➢ **jumpToFlash ()脚本函数代码：**

```
function jumpToFlash(j,i,rlayerIndex){
    var
url="VectorPage/Flash.jsp?rIndex="+j+"&sellayerIndex="+i+"&rlayerI
ndex="+rlayerIndex+"&action='Flash'" + "&rnd="+Math.random();
    top.document.getElementById("chargeFrame").src = url;
}
```

➢ **flash()脚本函数代码：**

```
function flash()
   {
      var arrayIndex=0;
      if(myFlashObjPoint!=null&&myFlashObjPoint.length>0)
      {
for(arrayIndex=0;arrayIndex<myFlashObjPoint.length;arrayIndex++)
         {
            if(myFlashObjPoint[arrayIndex]!=null)
            {
               myFlashObjPoint[arrayIndex].dispose();
```

```
                }
            }
         }
        if(myFlashObjLine!=null&&myFlashObjLine.length>0)
        {

for(arrayIndex=0;arrayIndex<myFlashObjLine.length;arrayIndex++)
            {
                if(myFlashObjLine[arrayIndex]!=null)
                {
                    myFlashObjLine[arrayIndex].dispose();
                }
            }
        }
        if(myFlashObjPolygon!=null&&myFlashObjPolygon.length>0)
        {
for(arrayIndex=0;arrayIndex<myFlashObjPolygon.length;arrayIndex++)
            {
                if(myFlashObjPolygon[arrayIndex]!=null)
                {
                    myFlashObjPolygon[arrayIndex].dispose();
                    delete myFlashObjPolygon[arrayIndex];
                }
            }
         }
       if(top.map.isIE)
            url =
"VectorPage/JumpMid.jsp?selLayerIndex="+sellayerIndex+"&countIndex
="+rIndex+"&layerIndex="+reallayerIndex+"&rnd="+Math.random();
       else
            url =
"JumpMid.jsp?selLayerIndex="+sellayerIndex+"&countIndex="+rIndex+"
&layerIndex="+reallayerIndex+"&rnd="+Math.random();
       var pntStr = top.AJAXRequest(url,false,"GET");
       var xmin = 0;
       var ymin = 0;
       var xmax = 0;
       var ymax = 0;

       if(pntStr.indexOf("(")!=-1){
           eval(pntStr);
       }else {
          var geoArr = pntStr.split('*'); //分割点线区
          //点
```

```
            if (geoArr.length>0&&geoArr[0] != "")
             {
                var pntArr = geoArr[0].split(';'); //分割各个点
                for (var i = 0; i < pntArr.length; i++)
                {
                   if(pntArr[i]!="")
                   {
                    pntDotArr =pntArr[i].split(','); //分割各个点的X,Y坐标
                        if(pntDotArr.length!=2){
                           return;
                        }
                       //中心点坐标
                        if(i==0)
                        {
                        mapObj.pntLocation(pntDotArr[0],pntDotArr[1]);
                         }
                        pntScreen =
mapObj.logicToScreen(pntDotArr[0],pntDotArr[1]);
                        idIndex=idNum;
                        intervalId[idIndex] =
setTimeout("checkImgLoaded("+i+","+"'point',"+idIndex+")", 100);
                        idNum++;
                    }
                }
                return;
             }
             //线
            ……
             //区
            ……
             }
        }
    }
```

注：以上只是 flash()脚本函数的部分，完整代码参见示例站点 Flash.jsp 页面的源码。

在 Flash.jsp 页面中，通过 body 标记中的 onload 事件触发 flash()脚本函数。在 flash()脚本函数中，首先通过 JumpMid.jsp 页面中处理获取查询结果中的点集合，然后分割闪烁要素（点、线、区），将得到要素的坐标值字符串转换成窗口坐标字符串，最后通过循环用 checkImgLoaded()脚本函数完成闪烁功能。

以点要素的闪烁为例，其具体步骤是：

（1）在 JumpMid.jsp 页面中，根据查询结果的索引获取点查询结果的几何信息，取得闪烁要素的坐标（分别为 x 坐标数组和 y 坐标数组）集合。

- **在 JumpMid.jsp 中部分处理代码：**

```
int pntNum = -1, RegNum = -1, LinNum = -1;
```

```
int selLayerIndex;
long featureID;
long LayerIndex;
int rIndex;
try{
    //获取查询结果所在的图层索引号,
    selLayerIndex =
Integer.parseInt(request.getParameter("selLayerIndex"));
    featureID = -1;
    LayerIndex =
Long.parseLong(request.getParameter("layerIndex"));
    rIndex = Integer.parseInt(request.getParameter("countIndex"));
}catch(Exception e){
    out.println("alert('获取闪烁要素信息失败。')");
    return;
}
//查询到的所有图层中的记录集合
CLayerSelectResult[] layerSelectRlt;
layerSelectRlt = selectResultObj.getlayerSelInfo();
featureID = layerSelectRlt[selLayerIndex].getFeatureIDs()[rIndex];

session.setAttribute("featureID",String.valueOf(featureID));

//获取查询结果的几何信息
CFeatureGeometry featureGeometry = null;
CGetFeatureInformation featureToGet = new CGetFeatureInformation();
featureToGet.setFeatureID(featureID);
featureToGet.setLayerIndex(LayerIndex);
featureToGet.setInformationType((byte) (0));
featureToGet.setMapID(viewport.getMapID());
try{
    featureGeometry = viewport.GetFeatureGeometry(featureToGet);
}catch(Exception e){
    out.println("alert('获取几何信息失败。')");
    return;
}
if (featureGeometry == null)
{
    throw new Exception("获取结果失败！请重新查询！");
}
StringBuffer sb = new StringBuffer(1024);
//---------------获取点查询结果中所有点数   start-----------------//
pntNum = featureGeometry.getPointNumber();
CPoint2D[] point = featureGeometry.getPoints();
```

```
for (int i = 0; i < pntNum; i++)
{
    sb.append(String.valueOf(point[i].getPointX()) + ",");
    sb.append(String.valueOf(point[i].getPointY()) + ",");
}
if (sb.length() != 0)
{
    sb.delete(sb.length() - 1, sb.length());
}
sb.append("*");

//---------------获取点查询结果中所有点数   end--------------------//
```

注：以上只是部分处理代码，完整代码参见示例站点 JumpMid.jsp 页面的源码。

如代码所示，通过 JumpMid.jsp 页面来获取点查询结果中所有点数。其中 layerIndex 即为该要素的索引。

（2）调用同步的 AJAX 请求返回查询结果。AJAXRequest 方法的三个参数的意义分别为 url 指定处理页面，false 指定不用异步请求，get 是提交的方式。

（3）处理返回的坐标 pntStr。返回的坐标是一个字符串，通过 split()方法将之分割，为下一步的闪烁做准备。

（4）最后调用 checkImgLoaded 函数对点要素进行闪烁。

➢ **checkImgLoaded()中部分代码：**

```
function checkImgLoaded(j,drawType,idindex)
   {
      var flashTimes,flashInterval,flashColor;
      var n=0;
      top.flashTimes = top.flashTimes;
      flashInterval = top.flashInterval;
      switch(drawType)
      {
         case "point":
            pntDiv = top.document.getElementById("pntFlashDiv");
            if(pntDiv==null)
            {
               pntDiv = top.document.createElement("div");
               pntDiv.id = "pntFlashDiv";
               pntDiv.style.position = "absolute";
               mapObj.addControl(pntDiv,true);
            }
            if(action == 'MoveFlash'){
               DrawPnt(pntDotArr[0],pntDotArr[1]);
               alert("请在地图上使用鼠标移动点几何");
               addEventEditDiv();
            }else if(pntDotArr.length==2){
```

```
                times = top.flashTimes*2;
                DrawPnt();
                FlashPnt();
                mapObj.pntLocation(pntDotArr[0],pntDotArr[1]);
            }
            break;
        default:
            break;
    }
    clearInterval(intervalId[idindex]);
}
```

注：以上只是 checkImgLoaded ()脚本函数的部分，完整代码参见示例站点 Flash.jsp 页面的源码。

3. **编辑功能**：主要通过脚本函数 edit()处理，将相关参数传到 VectorPage 文件夹下的 AttEdit.jsp 页面进行处理，与 GIS 服务器端交互完成编辑功能。

➢ **edit()代码：**

```
function edit(rIndex,sellayerIndex){
    getArgs();
    var url =
"VectorPage/AttEdit.jsp?sellayerIndex="+sellayerIndex+"&rIndex="+r
Index+"&rnd="+Math.random();
    top.document.getElementById("rsFrame").src = url;
}
```

通过点击编辑按钮，调用 edit()转到 AttEdit.jsp 页面。该编辑页面中包括：移动，删除，更新和缓冲区分析等功能。以点编辑为例来说明如何实现：

（1）**点的移动功能**：在右侧子框架的 AttEdit.jsp 编辑面板上点击移动，则在地图上显示这个要素，此时可以在地图上拖动图标，移动图标到需要的位置后再点击更新，就将新的位置保存到服务器端。下面主要介绍点要素移动的几个函数。

➢ **移动按钮客户端代码：**

```
<img id="Btn_Move" src="../images/btn_move.gif"
          onclick="MoveFeature()" />
```

通过触发点击事件中的 MoveFeature()脚本函数进行处理。

➢ **MoveFeature()代码：**

```
function MoveFeature()
       {
          var
url="VectorPage/Flash.jsp?rIndex="+rIndex+"&sellayerIndex="+selLay
erIndex+"&rlayerIndex="+realLayerIndex+"&action='MoveFlash'";
 top.document.getElementById("chargeFrame").src=url;
}
```

上述代码是点移动功能显示要素的函数，该功能在地图上显示要素的过程和代码与闪烁功能中的闪烁显示类似，不再重复说明。最后在 Flash.jsp 页面中调用

checkImgLoaded 函数显示要素时，flash()脚本函数调用了 addEventEditDiv()进行处理。

- **addEventEditDiv()代码：**

```
function addEventEditDiv()
 {
    mapObj.drawAction = "editPoints";
    mapObj.activeTool="";
    EventObj.addListener(pntDiv, "mousedown",editDivDown,
false);
    EventObj.addListener(pntDiv, "mousemove",editDivMove,
false);
    EventObj.addListener(pntDiv, "mouseup",editDivUp, false);
   }
```

addEventEditDiv()脚本函数是向主图上注册 mouseDown,mouseMove,mouseUp 三个方法。这里用的 EventObj.addListener(pntDiv, "mousedown",editDivDown, false)方法，意思是监听 pntDiv 上的 mousedown 事件，用 editDivDown 方法处理，false 是阻止事件冒泡。另两个方法与第一个类似。

注：editDivDown、editDivMove、editDivUp 三个鼠标监听处理方法代码见.NET 开发篇地图查询中的点移动功能部分，也可参见示例站点中 Flash.jsp 页面的相应源码。

至此，移动要素位置的功能就完成了，此时就可以点击更新按钮更新要素的位置。

（2）**更新要素功能**：在右侧子框架的 AttEdit.jsp 编辑面板上修改要素属性值，通过点击客户端的更新按钮，触发 UpdateFeature()脚本函数，通过该函数将从客户端获取的相关参数传到 UpdateFeature.jsp 页面，然后在 UpdateFeature.jsp 页面中分别进行要素属性信息更新、样式信息更新以及几何空间信息更新处理。下面以修改子图高度为例，主要介绍更新要素的样式：

- **子图高度的客户端代码：**

```
<tr>
  <td><label ID="Label_height">子图高度</label></td>
  <td>
    <input id="patternHeight" name="para" type="text"
value="20.0"style="width: 38px"
onchange="Validate('float','patternHeight','子图高度')" /> (float型)
  </td>
 </tr>
```

- **更新按钮的客户端代码：**

```
<img id="Btn_Update" src="../images/btn_update.gif"
onclick="UpdateFeature()" />
```

通过更新按钮触发点击事件的 UpdateFeature()脚本函数。

- **UpdateFeature()部分代码：**

```
function UpdateFeature()
```

```
    {
        var offset = getFeatureOffset();
        var rIndex = '<%=rIndex%>'
        var fldNum = '<%=valueLength%>';
        var selLayerIndex = '<%=selLayerIndex%>'
        tab =  document.getElementById("AttributeFrame");
        inputs = tab.getElementsByTagName("input");
        var str = "";
        for(var i=0;i<fldNum;i++)
        {
            str +=
"&fldValue"+i+"="+eval("document.all.tb"+i).value;
        }
        //样式属性
        var patternIndex =
document.getElementById("form1").patternIndex.value;
        var MarginWidth =
document.getElementById("patternWidth").value;
        var MarginHeight =
document.getElementById("patternHeight").value;
        var Color = document.getElementById("color").value;
        var geoFlag = "<%=patter.getFeatureType()%>";
        var showPattern =
document.getElementById("showPatternInput").value;
        var patternStr = "";
        patternStr =
"&showPattern="+showPattern+"&PatternIndex="+patternIndex+"&Margin
Width="+MarginWidth+"&MarginHeight="+MarginHeight+"&Color="+Color+
"&geoFlag="+geoFlag;
        var url =
"VectorPage/UpdateFeature.jsp?fldNum="+fldNum+"&selLayerIndex="+se
lLayerIndex+"&rIndex="+rIndex+"&offset="+offset+encodeURI(str)+pat
ternStr+"&rnd="+Math.random();
        top.document.getElementById("chargeFrame").src = url;
        }
```

在 UpdateFeature()脚本函数中，获取要素属性编辑面板中设置的样式属性值，然后将这些值传到服务器后台页面 UpdateFeature.jsp 中处理。

- **UpdateFeature.jsp 页面中样式更新处理代码：**

```
//****更新样式信息****//
int PatternIndex = 1;     //子图号
float MarginWidth = 20;   //子图宽度
float MarginHeight = 20;  //子图高度
int Color = 1;           //子图颜色
```

```
int geoFlag;            //子图类型
CUpdateFeatureStyleInfo updatePatter = new
CUpdateFeatureStyleInfo();
boolean showPattern =
Boolean.parseBoolean(request.getParameter("showPattern"));
if(showPattern){
    try {
        PatternIndex =
Integer.parseInt(String.valueOf(request.getParameter("PatternIndex
")));
        MarginWidth =
Float.parseFloat(String.valueOf(request.getParameter("MarginWidth"
)));
        MarginHeight =
Float.parseFloat(String.valueOf(request.getParameter("MarginHeight
")));
        Color =
Integer.parseInt(String.valueOf(request.getParameter("Color")));
        geoFlag =
Integer.parseInt(String.valueOf(request.getParameter("geoFlag")));
    } catch (Exception ex) {
        out.println("<script language='javascript'>alert('获取更新样
式参数失败。');</script>");
        return;
    }
    updatePatter.setLayerIndex(LayerIndex);
    updatePatter.setFeatureID(featureID);
    updatePatter.setPatternIndex(PatternIndex);
    updatePatter.setMarginWidth(MarginWidth);
    updatePatter.setMarginHeight(MarginHeight);
    updatePatter.setColor(Color);
    try {
        updatePatter.setFeatureType(geoFlag);
        viewport.updateFeatureStyleInfo(updatePatter);
    } catch (Exception ex) {
        out
        .println("<script language=javascript>alert('更新样式属性失败!
')</script>");
    }
}
```

这是更新样式信息的后台代码。更新要素样式属性值的过程比较简单，显示接收样式属性参数，然后通过服务器端方法设置要素的样式属性值，再通过updateFeatureStyleInfo()方法更新即可。

（3）**缓冲区分析功能**：在右侧子框架的 AttEdit.jsp 编辑面板上输入缓冲半径，通过点击客户端的缓冲分析按钮，触发 BufAnalyse()脚本函数，然后转到后台页面 BufAnalyse.jsp 中进行处理。

- **缓冲区分析客户端代码：**

```
<tr>
    <td>
        <input id='t1' width='20' value='<%=lengthOfOnePixel* 100000
/ 10000%>' onchange="Validate('double','t1','缓冲区半径')" />
    </td>
</tr>
<tr>
    <td>
        <img id="btn_bufAnalyse" src='../images/btn_bufanalyse.gif'
onclick='BufAnalyse()' />
        <img id="Btn_close" src="../images/btn_close.gif"
onclick="Close()" />
    </td>
</tr>
```

通过缓冲区分析按钮触发点击事件的 BufAnalyse()脚本函数。

- **BufAnalyse()脚本函数代码：**

```
//缓冲区分析
    function BufAnalyse()
    {
        var Obj =  document.getElementById("t1");
        if(Obj.value == "")
            {
            alert("请输入缓冲半径（double类型）。");
            return;
        }
        var radius= document.getElementById('t1').value
        var rIndex = '<%=rIndex%>'
        var selLayerIndex = '<%=selLayerIndex%>'
        var url =
"VectorPage/BufAnalyse.jsp?radius="+radius+"&selLayerIndex="+selLa
yerIndex
                    +"&rIndex="+rIndex+"&rnd="+Math.random();
        //top.mapFrame.loaderObj.setBusy();
        top.document.getElementById('rsFrame').src = url
        }
```

BufAnalyse()脚本函数位于 AttEdit.jsp 页面中，该函数将缓冲半径等相关参数传到服务器后台页面 BufAnalyse.jsp 中处理。

- **BufAnalyse.jsp 页面的部分处理代码：**

```
    //声明缓冲区分析对象、返回值描述对象
    CSpatialBuffer buffer = new CSpatialBuffer();
    Random rand = new Random();
    //缓冲区分析后生成文件的名称
    buffer.setFeatureClassName("buffer" +
String.valueOf(rand.nextDouble() * 1000));
    //要进行缓冲区分析的要素实体号
    buffer.setFeatureIDS(featureIDs);
    //要进行缓冲区分析的要素个数
    buffer.setFeatureNumber((short)1);
    //缓冲区分析半径
    buffer.setRadius(radius);
    //要进行缓冲区分析的图层索引号
    buffer.setLayerIndex(LayerIndex);
    //调用缓冲区分析函数，并通过返回值验证分析是否成功
    buffer.setMapID(viewport.getMapID());
    CSpatialOperResult SpatialOperRlt = null;
    try{
        SpatialOperRlt = viewport.SpatialAnalysisBuffer(buffer);
    }catch(Exception e){
        out.println("<script language='javascript'>alert('缓冲区分析失
败。');</script>");
     return;
    }

    request.setAttribute("operation","4");
    request.setAttribute("fileName",viewport.getZipFileName());

request.getRequestDispatcher("Download.jsp").forward(request,respo
nse);
```

注：这是 BufAnalyse.jsp 页面的部分代码，完整代码参见示例站点中 BufAnalyse.jsp 页面的源码。

以上代码将给参数设置到 buffer 对象中，然后调用 GIS 服务端类库的 SpatialAnalysisBuffer()方法进行缓冲区分析，最后将分析结果返回到 Download.jsp 页面中。

地图查询中各查询具体操作、要素的闪烁与编辑功能的客户端操作请参见行业版的操作手册。

## 6.2.9 地图编辑功能

地图编辑功能主要实现在地图文档中动态地添加点、线、区和修改要素，并实时更新地图文档。地图文档的编辑功能是常见的应用之一。

具体实现流程为：

1. 在地图初始化 loadMap 脚本函数（位于 function.js 中）中加载图形搜索类和操作状

态类。若这一步在地图显示中已经加载，则可跳过。

- **加载图形搜索类和操作状态类的代码：**

```
//图形搜索类
pro = new IMSSearch(map);
map.addControl(pro,false);
//操作状态
var os = new IMSOperStatus();
map.addControl(os,false);
```

2. 在客户端（MapFrame.htm、menu.htm 页面）添加地图编辑的功能按钮，进行事件编写与绑定。

   在 menu.htm 页面客户端的地图编辑功能按钮效果图如下：

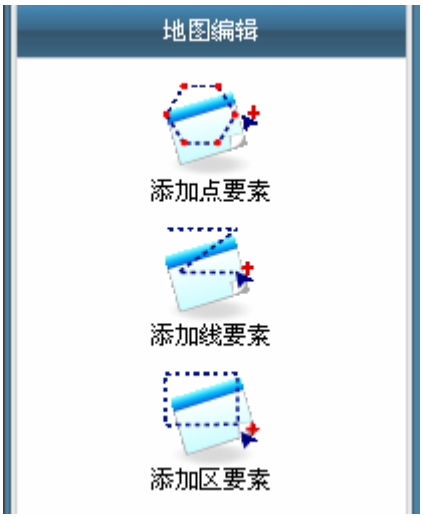



图 6-2-9-1 地图编辑功能按钮

- **地图编辑功能按钮的代码：**

```
 <img class="btncursor"
onclick="top.onSelectToolClick('0','Point','AddFeature.jsp')"
alt="添加点要素" src="images/appendPnt.gif" align="middle"
border="0"><br>
     添加点要素<br>
 <img class="btncursor"
onclick="top.onSelectToolClick('1','Polyline','AddFeature.jsp')"
alt="添加线要素" src="images/appendLin.gif" align="middle"
border="0"><br>
       添加线要素<br>
<img class="btncursor"
onclick="top.onSelectToolClick('2','Polygon','AddFeature.jsp')"
alt="添加区要素" src="images/appendReg.gif" align="middle"
border="0"><br>
       添加区要素<br>
```

3. 在地图编辑各功能按钮的 onclick 事件调用 onSelectToolClick 函数进行处理，该函

数位于脚本库 function.js 中。

- **onSelectToolClick()地图编辑部分代码：**

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
userParam = TargetPage;
userActive = activeTool;
switch (userActive) {
  case "0":                          //点编辑
    map.setCurOper(2001);
    pro.setSelectByHitting(1, submitUrl);
    break;
  case "1":                          //线编辑
    map.setCurOper(2005);
    pro.setSelectByLine(0, submitUrl);
    break;
  case "2":                          //区编辑
    map.setCurOper(2004);
    pro.setSelectByPolygon(submitUrl);
    break;
  default:
    break;
}
   }
```

上面的函数就是将各种编辑和处理方法关联。activeTool 参数：代表编辑的类型；drawAction 参数：代表客户端鼠标的操作类型； TargetPage 参数：代表处理该编辑的服务器页面。

4. 在 onSelectToolClick 函数中首先根据参数调用 setCurOper 函数设置地图操作状态为相应类型，在 setCurOper 函数中调用 setQueryMode 函数，根据 drawAction 参数（客户端鼠标在地图上的操作类型）添加事件监听并调用相应的脚本处理函数，在处理函数中获取相应的参数 param（为 submitUrl 函数的参数）。

- **setCurOper()地图编辑部分代码：**

```
//设定当前操作状态
 this.setCurOper=function(operType)
 {
    this.operType = parseInt(operType);
    this.clearOper();
    switch(this.operType)
    {
        case 2001:
            if(this.so==null){alert(this.gStr[0]);return;};
            this.so.setQueryMode("SelectByHitting","Point");
        break;
        case 2004:
            if(this.so==null){alert(this.gStr[0]);return;};
```

```
            this.so.setQueryMode("SelectByPolygon","Area");
          break;
          case 2005:
            if(this.so==null){alert(this.gStr[0]);return;};
            this.so.setQueryMode("SelectByLine","Polyline");
          break;
       }
   };
```

- setQueryMode()部分代码：

```
 this.setQueryMode = function(activeTool,drawAction)
   {
        thisObj.activeTool = activeTool;
        thisObj.drawAction = drawAction;
        switch(drawAction){
              case "Point":
                 if(this.listenerPoint){
 thisObj.addEventListener("mouseup",thisObj.so.selectByHitting ,
false);
                    this.listenerPoint = false;
                 }
                 break ;
              case "Area" :
                 if(this.listenerPolygon){
 thisObj.addEventListener("dblclick",thisObj.so.selectByPolygon,
false);
                    this.listenerPolygon = false;
                 }
                 break ;
              case "Polyline":
                 if(this.listenerPolyline){
thisObj.addEventListener("dblclick",thisObj.so.selectByLine,
false);
                    this.listenerPolyline = false;
                 }
        }
   };
```

- 以添加点要素为例，setSelectByHitting()代码：

```
  this.setSelectByHitting = function(ioffset,dealFun)
   {
      this.offset[0] = ioffset;
      this.dealFunArr[0] = dealFun;
   };
```

- **以添加点要素为例，selectByHitting()代码为：**

```
this.selectByHitting= function()
 {
    if(thisObj.activeTool!="SelectByHitting"){

thisObj.removeEventListener("mouseup",thisObj.so.selectByHitting ,
false) ;
        thisObj.so.listenerPoint =true;
        return ;
    }
    var dx = thisObj.so.offset[0];
    var X = Math.max(thisObj.mdLogicX,thisObj.muLogicX);
    var Y = Math.max(thisObj.mdLogicY,thisObj.muLogicY);
    //用于计算网格编号
    var startX=X-dx;
    var startY=Y-dx;
    var endX=X+dx;
    var endY=Y+dx;
    var
param="centerX="+X+"&centerY="+Y+"&radius="+dx+"&GridCodes="+thisO
bj.getGridCodes(startX,startY,endX,endY)+"&lev="+(thisObj.inf[2]+1
)+"&sType=SelectByHitting";
    if(thisObj.so.dealFunArr[0]!=null)
        thisObj.so.dealFunArr[0](param);
 };
```

通过鼠标事件获取所点击的 x、y 坐标值等参数值，调用 dealFunArr[0]，即 submitUrl 函数，把参数组 param 设置为 submitUrl 函数的参数。

5. 然后分别调用自定义函数处理。各函数中的 submitUrl 参数是一个对象，它就是对各种编辑的处理方法。由 submitUrl()处理将编辑参数传到相应的服务器页面处理，并通过 renderSelectResult 函数将结果返回。

- **submitUrl()代码：**

```
function submitUrl(param) {
 subParam = param;
 var url = "VectorPage/" + userParam + "?" + param + "&op=" + userActive;
 if (userActive != "SelectByRectAndAtt" || userActive !=
"SelectByCircleAndAtt" || userActive != "SelectByPolygonAndAtt") {
    renderSelectResult(url);
    }
}
```

该函数是各种编辑的处理函数，其中 userParam 的值为从客户端传过来的 AddFeature.jsp 页面，param 为 selectByHitting()获取到的坐标值等参数，是在核心脚本库 IMSMaps.js 中生成的。各种编辑相应的名值对参数如下：

添加点：centerX="+X+"&centerY="+Y+"&radius="+dx

添加线：XArray="+Xarray+"&YArray="+Yarray

添加区：XArray="+Xarray+"&YArray="+Yarray

- renderSelectResult()代码：

```
function renderSelectResult(url) {
 var obj = document.getElementById("rsFrame");
 switchdiv(1);
 obj.src = "";
 obj.src = url;
 var locPntDiv = document.getElementById("locPntDiv");
 if (locPntDiv != null) {
     locPntDiv.innerHTML = "";
     locPntInfoArr = new Array();
     }
}
```

该函数用于将处理结果显示到一个叫做 rsFrame 的 iframe 中。

6. AddFeature.jsp 页面，用于在客户端处理各种编辑。其大致流程为：首先，在该页面的客户端，根据编辑类型设置不同的编辑面板，放置于右侧 ID 为 rsFrame 的子框架里；然后通过提交按钮触发点击事件的 AddFeature()脚本函数，在该函数中将各编辑在客户端设置的参数传到 AddFeatureRlt.jsp 后台页面处理。

   AddFeature.jsp 页面客户端的编辑面板效果如下图：

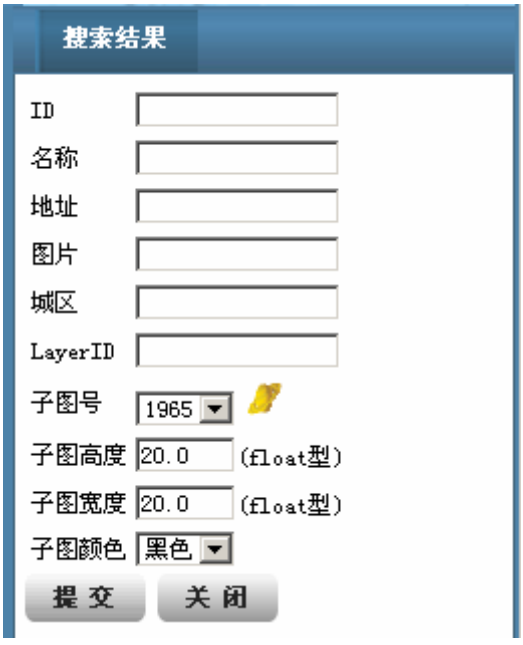



图 6-2-9-2 地图编辑面板

- AddFeature()脚本函数代码：

```
function AddFeature()
{
    var fldNum ='<%=fldNum%>';
```

```
    var op = '<%=op%>';
    var fieldvalue;
    tab =  document.getElementById("AttributeFrame");
    inputs = tab.getElementsByTagName("input");

    var num = parseFloat(fldNum);

    var str = "";
    var i;
    for(var i=0;i<fldNum;i++)
    {
        str += "&fldValue"+i+"="+eval("document.all.tb"+i).value;
    }
    if(<%=op%> == 0)
    {
        str +=
"&picType="+eval("document.all.fn0").value+"&picHeight="+eval("doc
ument.all.fn1").value+"&picWidth="+eval("document.all.fn2").value+
"&picColor="+eval("document.all.fn3").value;
    }
    else if(<%=op%> == 1)
    {
        str +=
"&lineType="+eval("document.all.fn0").value+"&penWidth="+eval("doc
ument.all.fn1").value+"&linColor="+eval("document.all.fn2").value+
"&lineRateX="+eval("document.all.fn3").value+"&lineRateY="+eval("d
ocument.all.fn4").value;
    }
    else if(<%=op%> == 2)
    {
        str +=
"&regionPic="+eval("document.all.fn0").value+"&regionColor="+eval(
"document.all.fn1").value+"&picHeight="+eval("document.all.fn2").v
alue+"&picWidth="+eval("document.all.fn3").value+"&picColor="+eval
("document.all.fn4").value;
    }
    var url =
"VectorPage/AddFeatureRlt.jsp?fldNum="+fldNum+"&op="+op+encodeURI(
str)+"&rnd="+Math.random();

    top.document.getElementById("chargeFrame").src = url;
}
```

7. 服务器后台页面 AddFeatureRlt.jsp，用于进行各种编辑的后台处理。大致流程为：

根据编辑类型（op）来进行各项编辑的处理，即接收客户端参数后验证客户端所填属性的参数类型合法性，设置参数对象，然后调用服务器类库的方法实现与 GIS 服务器交互完成编辑功能。

- **以添加点要素为例，其后台的处理代码：**

```
case 0:// 点工作区
        int picType;
        float picHeight;
        float picWidth;
        int picColor;
        try{
 picType = Integer.parseInt(request.getParameter("picType"));
 picHeight = Float.parseFloat(request.getParameter("picHeight"));
 picWidth = Float.parseFloat(request.getParameter("picWidth"));
 picColor = Integer.parseInt(request.getParameter("picColor"));
        }catch(Exception e){
            out.println("<script language='javascript'>alert('获取添加点要素信息失败。');</script>");
            return;
        }
 if(picType <= 0||picHeight <= 0||picWidth <= 0||picColor <= 0){
    out.println("<script language='javascript'>alert('请设置属性值为正数。');</script>");
            return;
        }

  CPoint2D dotLog = (CPoint2D) session.getAttribute("dotLogic");
        // 为添加点函数输入参数赋值
        CAppendAPoint thePoint = new CAppendAPoint();
        // 点要添加到的图层索引号
        thePoint.setLayerIndex(layerIndex);
        thePoint.setMapID(MapID);
        // 要添加的点属性字段长度
        thePoint.setPropertyNumber((short)fieldNum);
        // 要添加的点属性字段信息
        thePoint.setPropertyValues(fieldVal);
        // 要添加的点的逻辑坐标
        thePoint.setThePoint(dotLog);
        thePoint.setPicType(picType);
        thePoint.setPicColor(picColor);
        thePoint.setPicHeight(picHeight);
        thePoint.setPicWidth(picWidth);
        // 添加点函数
        try{
```

```
            viewport.AppendOnePoint(thePoint);
        }catch(Exception e){
            out.println("<script language='javascript'>alert('添加点要素失败。');</script>");
            return;
        }
            break;
```

以上代码先接收点要素的属性信息和坐标值等参数，验证后设置参数对象 thePoint，最后调用 AppendOnePoint()方法把点要素添加到地图文档中，并返回结果提示。

8. 以添加点要素为例，客户端具体操作见.NET 开发中的相应部分或者行业版操作手册。

## 6.2.10 空间分析功能

实现在地图文档中对点线区要素进行路径分析、拓扑分析、各种叠加分析与裁剪分析功能等。

客户端空间分析各功能按钮显示效果如下图：

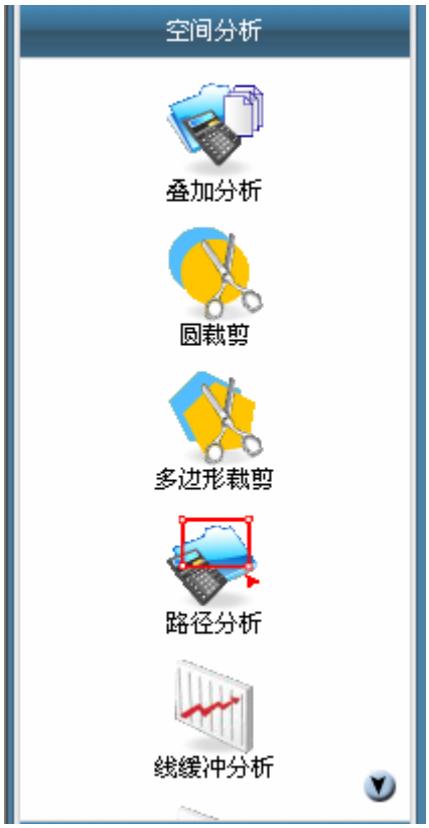



图 6-2-10-1 空间分析各功能按钮

下面分别介绍空间分析的各个功能实现流程。

1. **叠加分析**，实现对要素类或简单要素类中两个图层的相交、相减、合并及对称差几种叠加分析功能。具体实现流程为：

（1）在客户端（MapFrame.htm、menu.htm 页面）添加叠加分析的功能按钮，进行事件

编写及绑定。

- **叠加分析功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('OverLayAnalyse',null)" alt="叠加分析
" rc="images/overlay.gif" align="middle" border="0"><br>
    叠加分析<br>
```

（2）在叠加分析功能按钮的 onclick 事件调用 onSelectToolClick 函数进行处理，该函数位于脚本库 function.js 中。

- **onSelectToolClick()叠加分析部分代码：**

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
  userParam = TargetPage;
  userActive = activeTool;
  switch (userActive) {
    case "OverLayAnalyse":
      var url = "VectorPage/OverlayAnalyse.jsp?rnd=" + Math.random();
      renderSelectResult(url);
    break;
  }
}
```

该函数根据 userActive 参数值为 OverLayAnalyse，即进行叠加分析。转到 VectorPage 文件夹下的 OverlayAnalyse.jsp 页面进行叠加分析设置，用 renderSelectResult()将叠加分析处理结果显示到一个叫做 rsFrame 的 iframe 中，用户可以根据自己的需要修改。

（3）OverlayAnalyse.jsp 页面，用于进行叠加分析功能中的客户端参数设置，然后通过提交按钮触发 Btn_Submit_Click()脚本函数，在该函数中将参数传到服务器后台页面 OverlayAnalyseRlt.jsp 中处理。

- **Btn_Submit_Click()脚本函数代码：**

```
function Btn_Submit_Click()
   {
      var tab0 = document.getElementById("Mapgis_OverlayAnalyse1");
      var select1 = document.getElementById("layerList1");
      var select2 = document.getElementById("layerList2");
      var layerName0 = select1.value;
      var layerName1 = select2.value;
    layerName0=layerName0.replace(/\r\n/g, "");
    layerName1=layerName1.replace(/\r\n/g, "");
      var obj = document.getElementById("type");
       var type = obj.options[obj.selectedIndex].value;
      var radius = document.getElementById("noderadius").value;
      var user = document.getElementById("user").value;
      var password = document.getElementById("password").value;
      var resultLayer
=document.getElementById("resultLayer").value+Math.random();
```

```
if(layerName0==""||layerName1==""||radius==""||resultLayer=="")
       {
        alert("请设置参数。");
        return;
       }
       var regex =
"^(([0-9]+\\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\\.[0-9]+)|([0-9
]*[1-9][0-9]*))$";
        if(radius.match(regex) == null)
        {
        //alert("请设置结点平差半径为正浮点数。");
        //return;
        }
       var url =
"VectorPage/OverlayAnalyseRlt.jsp?&layerName0="+encodeURI(layerNam
e0)+"&layerName1="+encodeURI(layerName1)+"&radius="+radius+"&op="+
type+"&resultLayer="+resultLayer;
url+="&GDBName="+GDBName+"&GDBDataSourceName="+GDBDataSourceName+"
&user="+user+"&password="+password;
    url += "&rnd=" + Math.random();
    top.document.getElementById("rsFrame").src = url;
       }
```

在客户端连接叠加分析的数据源、选择图层和叠加类型等属性后，点击分析按钮，触发点击事件 Btn_Submit_Click 脚本，将参数传到后台页面处理。

（4）服务器后台页面 OverlayAnalyseRlt.jsp，用于进行叠加分析功能的后台处理。

- OverlayAnalyseRlt. jsp 页面部分处理代码：

```
  try {
    FileName = request.getParameter("resultLayer");
    GDBName = request.getParameter("GDBName");
    GDBDataSourceName = request.getParameter("GDBDataSourceName");
    layerName1 = request.getParameter("layerName0");
    layerName2 = request.getParameter("layerName1");
    operation = request.getParameter("op");
    radius = Double.parseDouble(request.getParameter("radius"));
    user = request.getParameter("user");
    password = request.getParameter("password");
} catch (Exception e) {
    out.println("<script language='javascript'>alert('获取叠加分析信
息失败。');</script>");
    return;
}
//声明叠加分析对象
CSpatialOverlay overlay = new CSpatialOverlay();
CSpatialOperResult SpatialOperRlt = null;
```

```
//为叠加分析的两个图层赋值
overlay.setLayerIndex1(-1);
overlay.setLayerIndex2(-1);
overlay.setGDBDataSourceName1(GDBDataSourceName);
overlay.setGDBName1(GDBName);
overlay.setFeatureIDSLst1(layerName1 + "," + layerName2);
overlay.setUserName1(user);
overlay.setPwd1(password);
overlay.setGDBDataSourceName2("");
overlay.setGDBName2("");
overlay.setFeatureIDSLst2("");
overlay.setUserName2("");
overlay.setPwd2("");
//叠加分析的类型，如并、差、交等
overlay.setOverlayType(Short.parseShort(operation));
//叠加分析后生成的文件名
overlay.setFeatureClassName(FileName);
overlay.setMapID(viewport.getMapID());
overlay.setRadius(radius);
overlay.setFlag(1);
try {
    SpatialOperRlt = viewport.SpatialAnalysisOverlay(overlay);
} catch (Exception e) {
    out.println("<script language='javascript'>alert('叠加分析失败。
');</script>");
    return;
}
try {
    //得到叠加分析后结果对象
    viewport.GetSpatialAnalysisResult(SpatialOperRlt);
} catch (Exception e) {
    out.println("<script language='javascript'>alert('从服务器获取叠
加分析结果失败。');</script>");
    return;
}
//通过结果对象取叠加分析最后文件所在URL地址
request.setAttribute("operation", "1");
request.setAttribute("fileName", viewport.getZipFileName());
request.getRequestDispatcher("Download.jsp").forward(request,
        response);
```

在该后台处理中，先接收各个参数，获取叠加分析的信息，然后设置对象 overlay，最后调用服务器类库的 SpatialAnalysisOverlay()方法进行叠加分析，最后调用 GetSpatialAnalysisResult()方法得到叠加分析后的结果对象，返回结果文件 URL 地址。

2. **拓扑分析**，分析两个图层要素在拓扑变化下其不变的性质。

（1）在客户端（MapFrame.htm、menu.htm 页面）添加拓扑分析的功能按钮，进行事件编写及绑定。

- **拓扑分析功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('SpatialDiffer','Point','TopoAnalys
e.jsp')" alt="拓扑分析" src="images/editable.gif" align="middle"
border="0"><br>
    拓扑分析<br>
```

（2）在拓扑分析功能按钮的 onclick 事件调用了 onSelectToolClick 函数进行处理，该函数位于脚本库 function.js 中

- **onSelectToolClick()拓扑分析部分代码：**

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
 userParam = TargetPage;
 userActive = activeTool;
 switch (userActive) {
   case "SpatialDiffer":
     var url = "VectorPage/TopoAnalyse.jsp?rnd=" + Math.random();
     renderSelectResult(url);
     break;
    }
}
```

上面的函数就是将拓扑分析和设置拓扑分析的页面 TopoAnalyse.jsp 关联。即转到 TopoAnalyse.jsp 页面进行拓扑分析设置，用 renderSelectResult()将拓扑分析处理结果显示到一个叫做 rsFrame 的 iframe 中。

（3）在 TopoAnalyse.jsp 页面中，在客户端选择两个要进行分析的要素，然后通过分析按钮触发点击事件的 startAnalyse()脚本函数处理。

TopoAnalyse.jsp 页面的拓扑分析设置面板如下图：

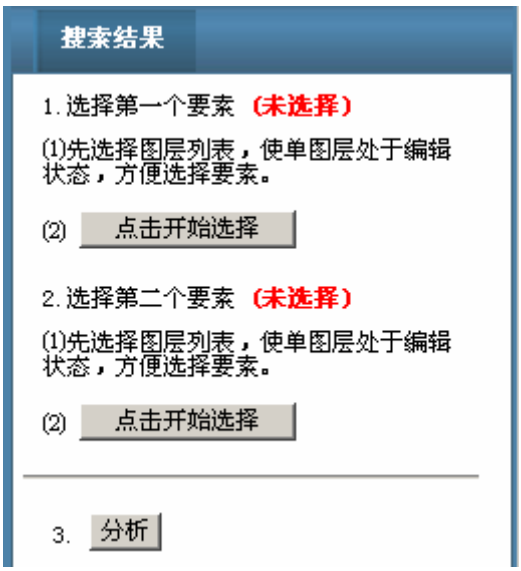



图 6-2-10-2 拓扑分析设置面板

- **TopoAnalyse.jsp 页面客户端代码：**

```
<form id="topoForm">
```

```
  <div id="f1">
    <div>
        1.选择第一个要素<span id="Label1" style="color: Red;
font-weight: bolder">(未选择)</span>
    </div>
   <div>
      ⑴先选择图层列表，使单图层处于编辑状态，方便选择要素。<br/><br/>
      ⑵<input type="button" value="点击开始选择"
onclick="beginSelect(1)" />
    </div>
   </div>
   <p />
   <div id="f2">
    <div>
        2.选择第二个要素<span id="Label2" style="color: Red;
font-weight: bolder">(未选择)</span>
    </div>
   <div>
        ⑴先选择图层列表，使单图层处于编辑状态，方便选择要素。<br/><br/>
        ⑵<input type="button" value="点击开始选择"
onclick="beginSelect(2)" />
   </div>
   </div>
   <hr/>
   <div style="padding-left: 10px">
      3.<input type="button" ID="submit" value="分析"
onclick="startAnalyse()" />
    </div>
    <div>
        <span ID="msg" style="color: Red; font-weight:
bolder"></span>
     </div>
    </form>
```

➢ startAnalyse()脚本函数代码：

```
function startAnalyse(){
  if(fIDArr.length < 2 || layerIndexArr.length < 2){
         alert("请选择两个要素进行拓扑分析");
         return;
    }
  var featureID1 = fIDArr[0];
  var featureID2 = fIDArr[1];
  var layerIndex1 = layerIndexArr[0];
  var layerIndex2 = layerIndexArr[1];
```

```
   var url = "VectorPage/SpatialDiffer.jsp?featureID1=" +  featureID1
+ "&featureID2=" + featureID2 + "&layerIndex1=" + layerIndex1 +
"&layerIndex2=" + layerIndex2;
   top.loadBar(0);
   var str=top.AJAXRequest(url,false,"GET");
   if(isNaN(str)){
          eval(str);
          return;
      }
   document.getElementById("msg").innerHTML = "两要素的拓扑关系为:
"+topoTypes[parseInt(str)];
   document.getElementById("Label1").style.color="red";
   document.getElementById("Label2").style.color="red";
   document.getElementById("Label1").innerHTML="(未选择)";
   document.getElementById("Label2").innerHTML="(未选择)";
   fIDArr = [];
   layerIndexArr = [];
}
```

通过该脚本函数，将参数以 AJAX 传到 SpatialDiffer.jsp 后台页面进行拓扑分析处理，返回分析结果。

（4）服务器后台页面 SpatialDiffer.jsp，用于拓扑分析处理，实现拓扑分析功能。

- **SpatialDiffer.jsp 后台页面处理代码：**

```
MapViewPort viewport =
(MapViewPort)session.getAttribute("viewport");
 long layerIndex1 =
Long.parseLong(request.getParameter("layerIndex1"));
 long layerIndex2 =
Long.parseLong(request.getParameter("layerIndex2"));
 long featureID1 =
Long.parseLong(request.getParameter("featureID1"));
 long featureID2 =
Long.parseLong(request.getParameter("featureID2"));
 //获取第一个要素的空间几何信息
 CGetFeatureInformation factor1 = new CGetFeatureInformation();
 factor1.setFeatureID(featureID1);
 factor1.setInformationType((byte)1);
 factor1.setLayerIndex(layerIndex1);
 factor1.setMapID(viewport.getMapID());
 CFeatureGeometry feature1 = viewport.GetFeatureGeometry(factor1);
 //获取第二个要素的空间几何信息
 CGetFeatureInformation factor2 = new CGetFeatureInformation();
 factor2.setFeatureID(featureID2);
 factor2.setInformationType((byte)1);
```

```
factor2.setLayerIndex(layerIndex2);
factor2.setMapID(viewport.getMapID());
CFeatureGeometry feature2 = viewport.GetFeatureGeometry(factor2);
//进行拓扑分析
CSpatialDiffer differ = new CSpatialDiffer();
differ.setFeature1(feature1);
differ.setFeature2(feature2);
try{
    TopoType topo = viewport.spatialDiffer(differ);
    out.print(topo.getType());
}catch(Exception e){
    out.print("拓扑分析失败，请重试");
}
```

在该页面中，先接收传过来的各参数，获取两个要素的空间几何信息，然后调用服务器类库的 spatialDiffer 方法进行拓扑分析，并返回拓扑分析结果。

**3. 裁剪分析**，实现圆裁剪和多边形裁剪功能。具体实现流程如下：

（1）在客户端（MapFrame.jsp、menu.htm 页面）添加裁剪分析的功能按钮，进行事件编写及绑定。

➢ **裁剪分析功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('ClipByCircle','Circle','Clip.jsp')
" alt="圆裁剪" src="images/circlip.gif" align="middle"
border="0"><br>
    圆裁剪<br>
<img class="btncursor"
onclick="top.onSelectToolClick('ClipByPolygon','Polygon','Clip.jsp
')" alt="多边形裁剪" src="images/polyclip.gif" align="middle"
border="0"><br>
    多边形裁剪<br>
```

（2）在裁剪分析功能按钮的 onclick 事件调用 onSelectToolClick 函数进行处理，该函数位于脚本库 function.js 中。

➢ **onSelectToolClick()裁剪分析部分代码：**

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
  userParam = TargetPage;
  userActive = activeTool;
  switch (userActive) {
   case "ClipByCircle":
    map.setCurOper(2003);
    pro.setSelectByCircle(submitUrl);
    break;
  case "ClipByPolygon":
    map.setCurOper(2004);
```

```
        pro.setSelectByPolygon(submitUrl);
        break;
    }
}
```

上面的函数就是将各种裁剪分析和设置裁剪分析参数的页面关联。activeTool 参数：代表裁剪的类型； drawAction 参数：代表客户端鼠标的操作类型； TargetPage 参数：代表设置裁剪分析参数页面。

（3）在 onSelectToolClick 函数中首先根据参数调用 setCurOper 函数设置地图操作状态为相应类型，在 setCurOper 函数中调用 setQueryMode 函数，根据 drawAction 参数（客户端鼠标在地图上的操作类型）添加事件监听并调用相应的脚本处理函数，在处理函数中获取相应的参数 param（为 submitUrl 函数的参数）。

- setCurOper()裁剪分析部分代码：

```
this.setCurOper=function(operType)
    {
        this.operType = parseInt(operType);
        this.clearOper();
        switch(this.operType)
        {
            case 2003:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByCircle","Circle");
            break;
            case 2004:
                if(this.so==null){alert(this.gStr[0]);return;};
                this.so.setQueryMode("SelectByPolygon","Area");
            break;
        }
    };
```

- setQueryMode()部分代码：

```
this.setQueryMode = function(activeTool,drawAction)
    {
        thisObj.activeTool = activeTool;
        thisObj.drawAction = drawAction;
        switch(drawAction){
              case "Circle" :
                  if(this.listenerCircle){
thisObj.addEventListener("mouseup",thisObj.so.selectByCircle,
false);
                      this.listenerCircle = false;
                  }
                  break ;
              case "Area" :
```

```
                if(this.listenerPolygon){
 thisObj.addEventListener("dblclick",thisObj.so.selectByPolygon,
false);
                    this.listenerPolygon = false;
                }
                break ;
        }
       };
```

➢ 以圆裁剪为例，setSelectByCircle()代码：

```
 this.setSelectByCircle = function(dealFun)
   {
       this.dealFunArr[2] = dealFun;
   };
```

➢ 以圆裁减为例，selectByCircle()代码为：

```
this.selectByCircle= function()
   {
      if(thisObj.activeTool!="SelectByCircle"){
thisObj.removeEventListener("mouseup",thisObj.so.selectByCircle ,
false) ;
         thisObj.so.listenerCircle =true;
         return ;
      }
      var centerX=thisObj.mdLogicX ;
      var centerY=thisObj.mdLogicY ;
      var upX = thisObj.muLogicX ;
      var upY = thisObj.muLogicY ;
      var w = Math.abs(centerX - upX);
    var h = Math.abs(centerY - upY);
      var radius = Math.sqrt(w*w + h*h);
      var startX=centerX-radius;
      var startY=centerY-radius;
      var endX=centerX+radius;
      var endY=centerY+radius;
      var
param="centerX="+centerX+"&centerY="+centerY+"&radius="+radius+"&G
ridCodes="+thisObj.getGridCodes(startX,startY,endX,endY)+"&lev="+(
thisObj.inf[2]+1)+"&sType=SelectByCircle";
      if(thisObj.so.dealFunArr[2]!=null)
         thisObj.so.dealFunArr[2](param);
   };
```

通过鼠标事件获取所点击的 x、y 坐标值组等参数值，调用 dealFunArr[2]，即 submitUrl 函数，把参数组 param 设置为 submitUrl 函数的参数。

（4）自定义函数中的 submitUrl 参数是一个对象，它就是对各种裁剪分析的处理方法。由 submitUrl()处理将裁剪参数传到后台页面 Clip.jsp 中处理，生成裁剪结果传到 Download.jsp 页面，最后通过 renderSelectResult 函数将裁剪分析结果设置显示到一个叫做 rsFrame 的 iframe 中。

➢ **submitUrl()代码：**

```
function submitUrl(param) {
 subParam = param;
 var url = "VectorPage/" + userParam + "?" + param + "&op=" + userActive;
 if (userActive != "SelectByRectAndAtt" || userActive !=
"SelectByCircleAndAtt" || userActive != "SelectByPolygonAndAtt") {
    renderSelectResult(url);
    }
}
```

该函数中 userParam 的值为从客户端传过来的裁剪分析的后台处理页面，即 Clip.jsp 页面；param 为鼠标事件获取到的坐标值等参数，是在核心脚本库 IMSMaps.js 中生成的。各种裁剪相应的名值对参数如下：

圆裁剪：centerX="+centerX+"&centerY="+centerY+"&radius="+radius

多边形裁剪：XArray="+Xarray+"&YArray="+Yarray

（5）服务器后台页面 Clip.jsp，根据裁剪类型用于处理圆裁剪和多边形裁剪。

➢ **以圆裁剪为例，Clip.jsp 页面圆裁剪部分代码：**

```
if(op.equalsIgnoreCase("ClipByCircle"))
{
       //声明圆裁剪对象、返回值对象
    CSpatialCircleClip circleClip = new CSpatialCircleClip();
       //要素类查询
       double xCir;
       double yCir;
       double radius;
       try{
        //接收客户端传来的圆心、半径及裁剪半径
        xCir =
Double.parseDouble((String)request.getParameter("centerX"));
        yCir =
Double.parseDouble((String)request.getParameter("centerY"));
        radius =
Double.parseDouble(request.getParameter("radius"));
       }catch(Exception e){
        out.println("<script language='javascript'>alert('获取裁剪信
息失败。');</script>");
        return;
    }
    CPoint2D dotCenter = new CPoint2D();
```

```
    dotCenter.setPointX(xCir);
    dotCenter.setPointY(yCir);
    circleClip.setCenter(dotCenter);
    circleClip.setRadius(radius);
    circleClip.setMapID(MapID);
    circleClip.setLayerIndex(Integer.parseInt(layerIndex));
    circleClip.setLayerIndexLst(layerIndex);
       //裁剪类型
    circleClip.setClipType((short)1);

 circleClip.setDisRadius((viewport.getMapOriginalBound().getXMax()
- viewport.getMapOriginalBound().getXMin())/10000);
    Random rand = new Random();
    circleClip.setFeatureClassName("CircleClip" +
String.valueOf(rand.nextDouble()*1000));
    try{
       SpatialOperRlt =
viewport.SpatialAnalysisCircleClip(circleClip);
    }catch(Exception e){
       out.println("<script language='javascript'>alert('裁剪失败。
');</script>");
       return;
    }

    request.setAttribute("operation","2");
 }
 request.setAttribute("fileName",fileName);
 request.getRequestDispatcher("Download.jsp").forward(request,resp
onse);
```

在 Clip.jsp 后台页面中根据客户端生成的圆心，半径和裁剪容差半径生成裁剪的结果，然后传到 Download. jsp 页面进行显示。

4. **路径分析**，通过在地图上设置路径结点和障碍点，实现最佳的路线的分析功能。具体实现流程如下：

（1）在客户端（MapFrame.htm、menu.htm 页面）添加路径分析的功能按钮，进行事件编写及绑定。

- **路径分析功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('NetAnalyse','MultiPoints','PathSet
ting.jsp')" alt="路径分析" src="images/pathAnalyze.gif"
align="middle" border="0"><br>
     路径分析<br>
```

（2）路径分析功能按钮的 onclick 事件触发 onSelectToolClick 脚本函数进行处理，该函

数位于脚本库 function.js 中。

- onSelectToolClick()路径分析部分代码：

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
 userParam = TargetPage;
 userActive = activeTool;
 switch (userActive) {
   case "NetAnalyse":
     var url = "VectorPage/PathSetting.jsp";
     renderSelectResult(url);
     break;
     }
}
```

上面的函数将路径分析功能和设置路径分析参数的页面 PathSetting. jsp 关联，然后用 renderSelectResult()脚本函数将路径分析处理结果设置显示到一个叫做 rsFrame 的 iframe 中。

（3）PathSetting.jsp 页面，用于设置路径分析参数。通过该页面的路径分析提交按钮触发点击事件的 submitPoints()脚本函数，用该函数提交点坐标串，然后转到 NetAnalyse.jsp 后台页面处理。

PathSetting.jsp 页面路径分析设置面板如下图：




图 6-2-10-3 路径分析设置面板

- PathSetting. jsp 页面设置参数的客户端代码：

```
 <table>
    <tr><td><input type="radio" name="radio1" checked="checked"
onfocus="top.changePntImg('access')">设置路径节点<p></td>
    </tr>
    <tr><td><input type="radio" name="radio1"
onfocus="top.changePntImg('barrier')">设置路径障碍点<br><p></td>
    </tr>
    <tr><td>
      <img id="btn_path" src="../images/btn_path.gif" onclick =
"top.submitPoints('NetAnalyse')">
      <img id="Bnt_flash" src="../images/btn_flash.png"
onclick="top.flashPath()">
       <img id="Bnt_close" src="../images/btn_close.gif"
```

```
onclick="top.Close()">
      <input type="button" value="清除节点"
onclick="top.cleanPoint()"/>
   </td>
 </tr>
 </table>
```

➢ **设置结点的脚本代码：**

```
function changePntImg(pntStyle) {
if (listenerFlag == false) {
    map.drawAction = "MultiPoints";
    map.addEventListener("mouseup", getPntCoor, false);
    listenerFlag = true;
}
pntType = pntStyle;
}

function getPntCoor() {
if (map.drawAction != "MultiPoints") {
    ZDEvent.removeListener(AnalysePntDiv, "mouseup", changePntImg,
false);
    listenerFlag = false;
    return;
}
pointImg = (pntType == "access") ? "images/lableBar.gif" :
"images/delete.gif";
var coorPnt = map.getClickCoor();
var screenPnt = map.logicToScreen(coorPnt[0], coorPnt[1]);
var wx = screenPnt[0] - 6 + "px";
var wy = screenPnt[1] - 6 + "px";
AnalysePntDiv.innerHTML += "<div style='position:absolute;left:" +
wx + ";top:" + wy + "'><img src =" + pointImg + "></div>";
if (pntType == "access") {
    accessPntArr += coorPnt[0] + "," + coorPnt[1] + ",";
} else {
    if (pntType == "barrier") {
       //alert("bar");
    barPntArr += coorPnt[0] + "," + coorPnt[1] + ",";
       }
    }
}
```

changePntImg 用于向地图注册一个 mouseup 事件的监听，使用 getPntCoor 方法处理。getPntCoor 主要用于获取在地图上设置的路径节点或障碍点的坐标，并在地图上表示出来设置的路径节点或障碍点。

➢ **提交点坐标串 submitPoints()代码**

```
function submitPoints(op) {
 var accessPntNum = 0;
 var barPntNum = 0;
 if (accessPntArr.length > 0) {
     accessPntNum = (accessPntArr.split(",").length - 1) / 2;
 }
 if (barPntArr.length > 0) {
     barPntNum = (barPntArr.split(",").length - 1) / 2;
 }
 if (accessPntNum < 2) {

 alert("\u8bf7\u6dfb\u52a0\u8def\u5f84\u70b9,\u81f3\u5c11\u4e24\u4
e2a");
     return;
 }
 var pntArr = (accessPntArr + barPntArr).substring(0, (accessPntArr
+ barPntArr).length - 1).split(",");
//下面一行需要修改
 var dxy = new Array();
 dxy[0] = map.logicXlength * 50 / map.pObj.offsetWidth;
 dxy[1] = map.logicYlength * 50 / map.pObj.offsetHeight;
 var url = "VectorPage/NetAnalyse.jsp?accessPntNum=" + accessPntNum
+ "&barPntNum=" + barPntNum;
 url += "&Points=" + pntArr + "&BufX=" + dxy[0] + "&BufY=" + dxy[1]
+ "&op=" + op;
 url += "&rnd=" + Math.random();
 document.getElementById("chargeFrame").src = url;
    }
```

通过该函数提交点坐标串到网络分析的后台页面 NetAnalyse.jsp 中处理。

注：changePntImg、getPntCoor、submitPoints 三个脚本函数都位于用户脚本库 function.js 中。

（4）路径分析的后台页面 NetAnalyse.jsp，对路径分析进行处理。

➢ **NetAnalyse.jsp 后台页面部分代码**

```
String PointOfPolyline = request.getParameter("Points");
    radius = Double.parseDouble(request.getParameter("BufX"));
    accessPntNum =
Integer.parseInt(request.getParameter("accessPntNum"));
    barPntNum =
Integer.parseInt(request.getParameter("barPntNum"));
   //得到客户端点击点个数
    String[] pointsCoor = PointOfPolyline.split(",");
    //申请相应空间来存放由客户端传过来的点坐标集，此时已是逻辑坐标
```

```
    CPoint2D[] accessPntArr = new CPoint2D[accessPntNum];
    for (int i = 0; i < accessPntNum; i++) {
        accessPntArr[i] = new CPoint2D();
        accessPntArr[i].setPointX(Double.parseDouble(pointsCoor[2 *
i]));
        accessPntArr[i].setPointY(Double.parseDouble(pointsCoor[2 *
i + 1]));
    }
    CPoint2D[] barPntArr = new CPoint2D[barPntNum];

    for (int i = 0; i < barPntNum; i++) {
        barPntArr[i] = new CPoint2D();
        barPntArr[i].setPointX(Double.parseDouble(pointsCoor[2 *
(i+accessPntNum)]));
        barPntArr[i].setPointY(Double.parseDouble(pointsCoor[2 *
(i+accessPntNum) + 1]));
    }

    CBasicNetAnalysis net = new CBasicNetAnalysis();
    CNetAnalysisResult netRlt = new CNetAnalysisResult();

    net.setAccessNodeNum(accessPntNum);
    net.setAccessNodes(accessPntArr);
    net.setLayerIndex(layerIndex);
    net.setRadius(radius);
    net.setObstacleNodeNum(barPntNum);
    net.setObstacleNodes(barPntArr);
    try{
        netRlt = viewport.NetAnalysis(net);
    }catch(Exception e){
        out.println("<script language='javascript'>alert('路径分析失
败。');</script>");
        return;
            }
```

该代码主要是根据路径节点和障碍点，设置参数对象 net，然后调用服务器类库的 NetAnalysis()方法，生成网络分析的结果。

5. **缓冲区分析功能：**实现的具体流程见地图查询功能小节的地图编辑最后部分，在这里不再重复介绍。

注：空间分析中各分析功能的客户端具体操作参见行业版的操作手册。

## 6.2.11 统计分析功能

统计分析功能实现以图表形式展示要素属性统计分析的结果。该功能的大致流程为：先

在工具栏中触发统计功能，此时会提示在地图上拉框选择统计的要素，拉框完毕后跳转到 DrawVectorChart.aspx 页面，设置统计图的各个参数，设置完毕后再提交到本页，生成统计图。

统计分析效果如下图：

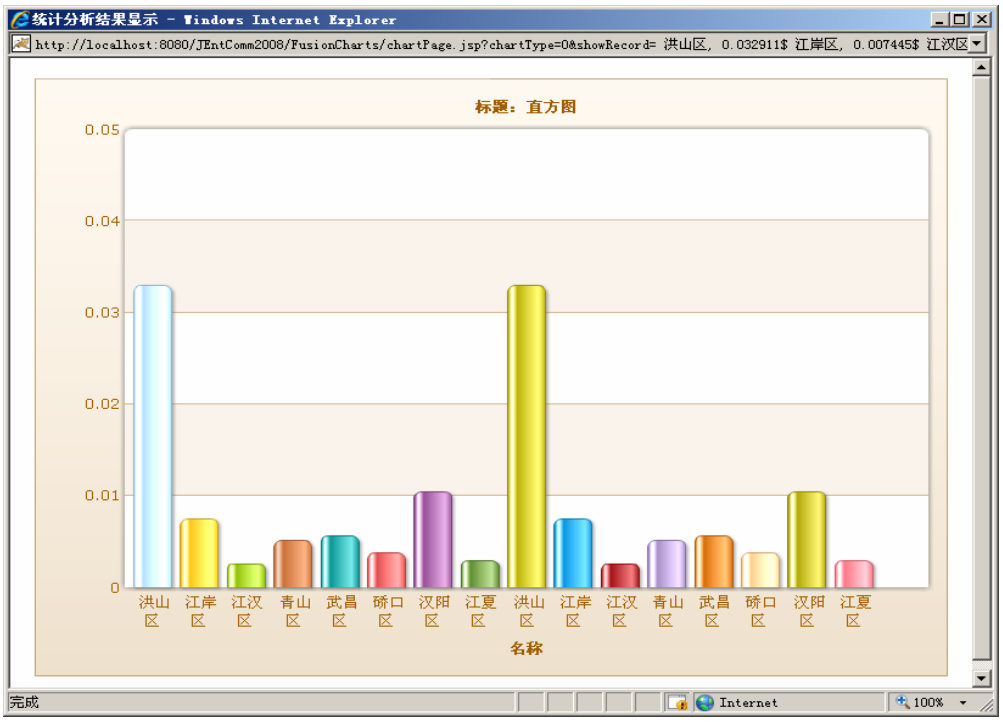



图 6-2-11-1 统计分析直方图

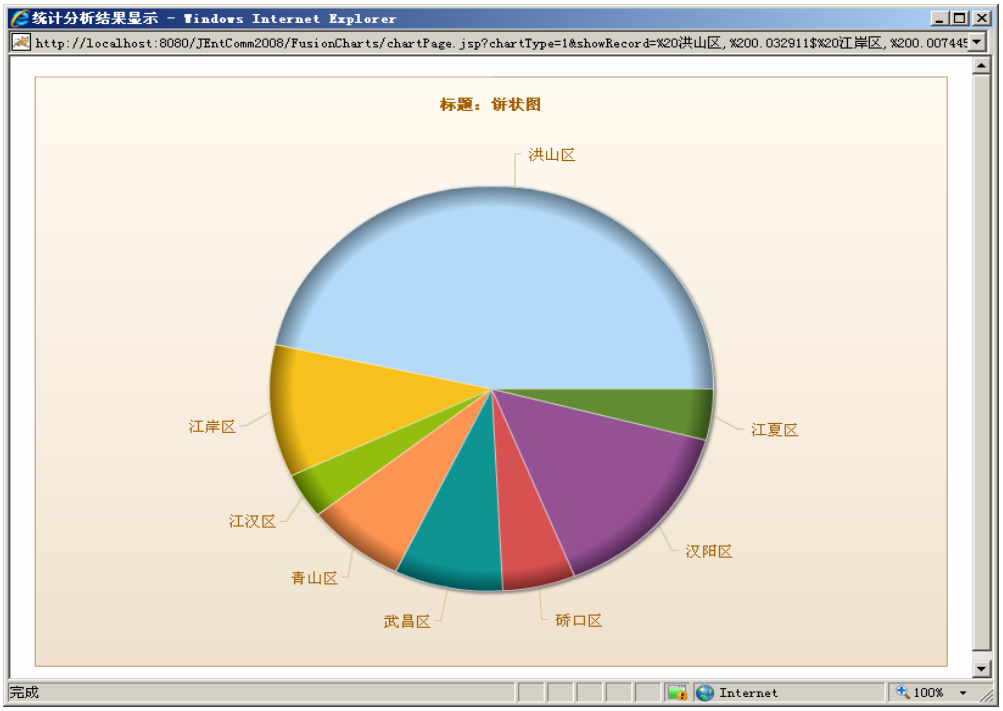



图 6-2-11-1 统计分析饼状图

具体实现流程为：

1. 在客户端 menu.htm 页面添加统计分析的功能按钮，进行事件编写及绑定。

   menu.htm 页面客户端统计分析各功能按钮显示效果如下图：




图 6-2-11-3 统计分析各功能按钮

➢ **统计分析功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('Area','Rect','Search.jsp')" alt="区域统计分析" src="images/chart.gif" align="middle" border="0"><br>
      区域统计分析<br>
<img class="btncursor"
onclick="top.onDisplayToolClick('queryChart')" alt="属性统计分析"
src="images/chart.gif" align="middle" border="0"><br>
      属性统计分析<br>
```

2. **区域统计分析功能**：按钮的 onclick 事件触发 onSelectToolClick 脚本函数进行处理，然后转到后台页面 Search.jsp 中进行区域查询统计，再转到 AttStat.jsp 页面设置显示样式，最后由 chartPage.jsp 页面处理并显示统计图表。具体流程及函数代码如下：

（1）用 onSelectToolClick 脚本函数进行处理，然后通过一系列的脚本函数处理后将客户端画区的所有参数传到后台页面 Search.jsp 中。

➢ **onSelectToolClick()区域统计分析部分代码：**

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
 userParam = TargetPage;
 userActive = activeTool;
 switch (userActive) {
   case "Area":
     map.setCurOper(2002);
     pro.setSelectByRect(submitUrl);
 alert("\u8bf7\u5728\u5730\u56fe\u4e0a\u62c9\u6846\u9009\u62e9\u7edf\u8ba1\u533a\u57df");
     break;
     }
}
```

该函数根据 userActive 参数的值为 Area，即进行区域统计分析。其参数 TargetPage 的值为 Search.jsp，将区域统计分析功能和服务器后台页面 Search.jsp 关联。该函数位于脚本库 function.js 中。

（2）服务器后台页面 Search.jsp，用于实现区域统计分析中的区域查询统计。

➢ **Search. jsp 页面区域统计分析部分代码：**

```
case 11://区域统计分析
 try {
    xmin = Double.parseDouble(request.getParameter("startX"));
    ymin = Double.parseDouble(request.getParameter("startY"));
    xmax = Double.parseDouble(request.getParameter("endX"));
    ymax = Double.parseDouble(request.getParameter("endY"));
     } catch (Exception e) {
       out.println("<script language='javascript'>alert('获取统计分
析信息失败。');</script>");
           return;
       }
    rc = new CRect();
    rc.setXMin(xmin);
    rc.setYMin(ymin);
    rc.setXMax(xmax);
    rc.setYMax(ymax);
    rectSelectObj = new CRectangleSelectClass();
    rectSelectObj.setMapID(viewport.getMapID());
    rectSelectObj.setRectangleArea(rc);
    try {
         rectSelectObj.setLayerIndex(layerQueryIndex);
        } catch (Exception e) {
           out.println("<script language='javascript'>alert('请先提
交图层。');</script>");
           return;
           }
    flag = viewport.SelectByRect(rectSelectObj);
    if (flag < 0) {
       out.println("<script language='javascript'>alert('查询失败。
');</script>");
       return;
       }
    selectResultObj = viewport.getLastSelectResult();
    // 查询时不生成图片
    session.setAttribute("selectResultObj", selectResultObj);
    out.println("<script
language='javascript'>top.document.getElementById('rsFrame').src =
'VectorPage/AttStat.jsp?rnd=" + Math.random() + "'</script>");
    break;
```

该页面接收从客户端传过来的所画区域坐标值组等参数，然后对该区域进行查询统计后，将结果传到 AttStat.jsp 页面中处理。

（3）AttStat.jsp 页面，用于设置统计图表类型等属性，然后通过脚本函数将统计结果转到 chartPage.jsp 页面处理以图表样式显示。

AttStat.jsp 页面设置面板效果如下图：

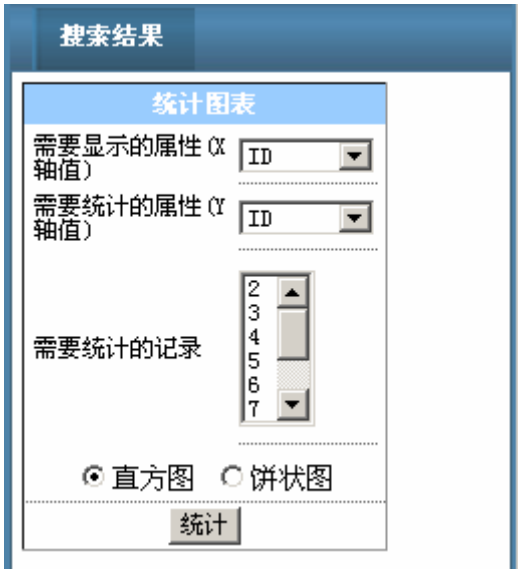



图 6-2-11-4 统计分析设置面板

➢ AttStat.jsp 页面客户端代码：

```
<tr>
  <td>需要显示的属性(X轴值)</td>
  <td style='BORDER-BOTTOM: #666666 1px dotted;HEIGHT: 21px'>
      <SELECT ID='nameIndex' name='nameIndex'>
          <%String[] fieldNames = layerSelectRlt.getFieldNames();
                for(short k = 0; k < colslen; k++){%>
          <OPTION VALUE="<%=k%>"><%=fieldNames[k]%></OPTION><%}%>
      </SELECT>
  </td>
</tr>
<tr>
  <td>需要统计的属性(Y轴值)</td>
  <td style='BORDER-BOTTOM: #666666 1px dotted'>
      <SELECT ID='valueIndex' name='valueIndex'>
          <%for(short k = 0; k < colslen; k++){%>
          <OPTION VALUE="<%= k%>"><%=fieldNames[k]%></OPTION>
          <%}%>
      </SELECT>
  </td>
</tr>
<tr>
   <td >需要统计的记录</td>
  <td style='BORDER-BOTTOM: #666666 1px dotted'>
      <SELECT ID='recordes' name = 'recordes'size='6'
multiple='multiple'>
 <%ArrayList[] propertyValues = layerSelectRlt.getPropertyValues();
      for(int i = 0; i< rowslen; i++){%>
          <OPTION VALUE="<%= propertyValues[i]%>">
```

```
            <%= propertyValues[i].get(0).toString()%>
            </OPTION>
           <%}%>
        </SELECT>
    </td>
 </tr>
……
<tr>
    <td align='center' style='BORDER-BOTTOM: #666666 1px dotted'>
<font style='FONT-SIZE: 14px; FONT-STYLE: normal; FONT-FAMILY: 宋体
  '> <input type='radio' id='Radio_Bar' value='bar' name='type'
  checked>直方图
 <input type='radio' id='Radio_Pie' value='pie' name='type'>饼状图
</font>
  </td>
</tr>
<tr>
    <td align='center'><input type='Button' ID='HandIn' value='统计
' onclick=showChart()></td>
</tr>
```

在客户端设置统计图表的 X 轴和 Y 轴字段以及统计图表的形状类型（直方图和饼状图），通过统计按钮触发点击事件的 showChart()脚本函数。

- showChart()脚本函数代码：

```
function showChart(){
    //得到需要统计的记录集合
    var obj = document.getElementById("recordes");
    var itemsOptions = obj.options;
    var itemsLength = itemsOptions.length;
    if(itemsLength == 0){
       alert("请选择需要统计的记录");
       return;
    }
    var tmpArray = new Array();
    valueIndex = document.getElementById("valueIndex").value;
    nameIndex = document.getElementById("nameIndex").value;
    for(var i = 0; i < itemsLength; i++){
       if(itemsOptions[i].selected == true){
           records[i] = obj.options[i].value;
           tmpArray = records[i].substring(1,records[i].length -
1).split(',');
           strResChart += tmpArray[nameIndex];
           strResChart += "," + tmpArray[valueIndex] + "$";
       }
```

```
    }
     //统计类型
     var isRadio_Bar = document.getElementById("Radio_Bar").checked;
     var isRadio_Pie = document.getElementById("Radio_Pie").checked;
     if(!isRadio_Bar)
         chartType = 1;
     else
         chartType = 0;
     document.getElementById("chartDiv").innerHTML = "<p><a
href='javascript:openResult()'>统计分析结果显示</a>";
       }
```

➢ openResult()脚本函数代码：

```
function openResult(){
    var fieldX =
document.getElementById("nameIndex").children[nameIndex].innerText
;
    var fieldY =
document.getElementById("valueIndex").children[valueIndex].innerTe
xt;
        var features =
"height=600,width=800,left=10,top=10,resizable=yes,status=no,toolb
ar=no,menubar=no,location=no,scrollbars=yes";
        var url =
"../FusionCharts/chartPage.jsp?chartType="+chartType+"&showRecord=
"+strResChart+"&fieldX="+fieldX+"&fieldY="+fieldY;
    window.open(url,'snowy',features);
    strResChart = "";
   }
```

通过 showChart()与 openResult()脚本函数，将统计结果传到 chartPage.jsp 页面，生成统计图表显示到客户端。chartPage.jsp 页面详细代码参见示例站点 FusionCharts 文件夹下的 chartPage.jsp 页面源码。

3. **属性统计分析功能**：其按钮的 onclick 事件触发 onDisplayToolClick 脚本函数进行处理，接着转到 ConditionGenerator.jsp 页面中进行查询条件设置，然后将设置的查询条件等参数传到服务器后台页面 Search.jsp 中进行属性条件查询统计，再转到 AttStat.jsp 页面设置显示样式，最后由 chartPage.jsp 页面处理并显示统计图表。具体流程及函数代码如下：

（1）用 onDisplayToolClick 脚本函数进行处理，然后通过一系列的脚本函数处理得到客户端画区的相关参数，再转到 ConditionGenerator.jsp 中设置属性查询条件。该函数位于脚本库 function.js 中。

➢ onDisplayToolClick()属性统计分析部分代码：

```
function onDisplayToolClick(con) {
 var type = con;
```

```
switch (type) {
 case "queryChart":
       map.setCurOper(1003);
       map.activeTool = type;
       subParam = "";
       var url = "VectorPage/ConditionGenerator.jsp?op=" + type +
"&rnd=" + Math.random();
       renderSelectResult(url);
       break;
    }
}
```

该函数根据 type 参数的值为 queryChart，即进行属性统计分析，将属性统计分析功能和设置属性条件的页面 ConditionGenerator.jsp 关联。

（2）在 ConditionGenerator.jsp 页面中设置属性查询条件，然后通过点击按钮触发事件的 conditionGenerator()脚本函数进行处理。

- **conditionGenerator()脚本函数代码**

```
//生成条件查询语句
   function conditionGenerator()
   {
     var url  = '<%=url%>';
     var op = '<%=op%>'
     var conditionStr
=document.getElementById('tb_condition').value;
     if(conditionStr=="")
     {
         alert("查询条件不能为空");
         return;
     }
     url =
"VectorPage/Search.jsp?"+top.subParam+"&op="+op+"&condition="+enco
deURI(conditionStr);
     top.document.getElementById("rsFrame").src = url;
       }
```

通过该脚本函数，将查询条件等参数传到服务器后台页面 Search.jsp 中处理。

（3）服务器后台页面 Search.jsp，用于实现属性统计分析中的属性条件查询统计。

- **Search. jsp 页面属性统计分析部分代码：**

```
case 12://属性统计分析
    CAttSelectStru statisticalString = new CAttSelectStru();
    statisticalString.setAttCondition(condition);
    statisticalString.setMapID(viewport.getMapID());
    statisticalString.setLayerIndex(layerQueryIndex);
    flag = viewport.SelectByCondition(statisticalString);
```

```
        if (flag < 0) {
            out.println("<script language='javascript'>alert('查询失败。
');</script>");
            return;
            }
        selectResultObj = viewport.getLastSelectResult();
        session.setAttribute("selectResultObj", selectResultObj);
        out.println("<script
language='javascript'>top.document.getElementById('rsFrame').src =
'VectorPage/AttStat.jsp?rnd="+ Math.random() + "'</script>");
        break;
```

该页面接收传过来的查询条件等参数，然后根据条件进行属性查询统计后，将结果传到 AttStat.jsp 页面中处理。

属性统计分析和区域统计分析在 AttStat.jsp 页面中进行同样的处理，其后的结果显示流程也完全相同，在这里不再详细介绍。想了解具体流程与代码，请参考区域统计的后半部分并参见站点中的详细代码。

## 6.2.12 坐标转换功能

坐标转换功能通过在客户端输入待转换的坐标值并设置相关属性，实现转换成目标坐标的功能。

坐标转换功能的效果如下图：




图 6-2-12-1 坐标转换效果

具体实现流程：

1. 在客户端 menu.htm 页面添加坐标转换的功能按钮，进行事件编写及绑定。

➢ **坐标转换功能按钮的代码：**

```
<img class="btncursor"
```

```
onclick="top.onDisplayToolClick('Projection',null)" alt="坐标转换"
alt="坐标转换" src="images/faceStyle.gif" align="middle"
border="0"><br>
    坐标转换<br>
```

2. 坐标转换功能按钮的 onclick 事件触发 onDisplayToolClick 脚本函数进行处理，该函数位于脚本库 function.js 中。

➢ **onDisplayToolClick()坐标转换部分代码：**

```
function onDisplayToolClick(con) {
 var type = con;
 switch (type) {
  case "Projection":
     var url = "VectorPage/Projection.jsp?rnd=" + Math.random();
     renderSelectResult(url);
     break; }
}
```

该函数的 type 参数值为 Projection，进行坐标转换处理，即将转到 Projection.jsp 页面中进行坐标转换的参数设置。

3. Projection.jsp 页面，用于在客户端设置坐标转换的各参数。在该页面中，先输入坐标值，设置需要转换的相关属性，然后通过转换按钮触发点击事件的 projectClick() 脚本函数进行处理。

Projection.jsp 页面坐标转换的设置面板如下图：

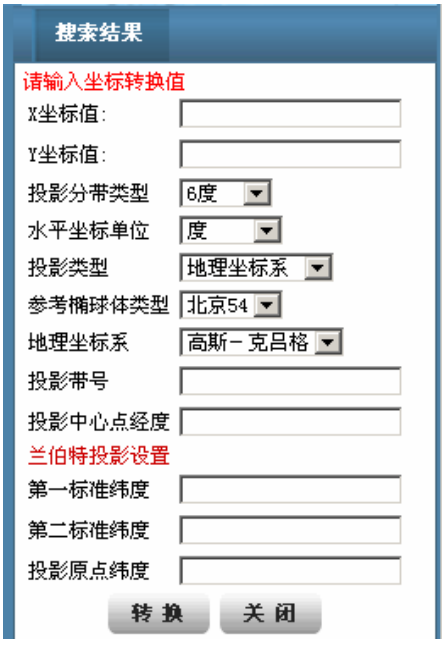



图 6-2-12-2 坐标转换设置面板

➢ **projectClick()脚本函数代码：**

```
//点击投影坐标转换
   function projectClick()
   {
```

```
    if (myform.tb_IZone.value=="" || myform.tb_DLon.value=="")
       {
          this.proRltDiv.innerHTML = "\"投影带号\",\"投影中心点经度\"
参数不能为空.";
          return;
       }
       var xmin = myform.tb_x.value;
       var ymin = myform.tb_y.value;
       var BZoneType =
myform.ddl_BZoneType.options[myform.ddl_BZoneType.selectedIndex].v
alue;
       var DAngUnit =
myform.ddl_DAngUnit.options[myform.ddl_DAngUnit.selectedIndex].val
ue;
       var DLon = myform.tb_DLon.value;
       var IProjTypeId =
myform.ddl_IProjTypeId.options[myform.ddl_IProjTypeId.selectedInde
x].value;
       var ISpheroid =
myform.ddl_ISpheroid.options[myform.ddl_ISpheroid.selectedIndex].v
alue;
       var IType =
myform.ddl_IType.options[myform.ddl_IType.selectedIndex].value;
       var IZone = myform.tb_IZone.value;
       var url =
"VectorPage/ProjectTransform.jsp?xmin="+xmin+"&ymin="+ymin;
       url +=
"&BZoneType="+BZoneType+"&DAngUnit="+DAngUnit+"&DLon="+DLon+"&IPro
jTypeId="+IProjTypeId;
       url +=
"&ISpheroid="+ISpheroid+"&IType="+IType+"&IZone="+IZone;
       //alert(url)
       top.document.getElementById("chargeFrame").src = url +
"&rnd="+Math.random();
       }
```

上述代码用于从客户端获取须转换的坐标以及各个参数，然后将这些参数传到 ProjectTransform.jsp 后台页面处理。

4. 服务器后台页面 ProjectTransform.jsp，用于处理坐标转换操作。

➢ **ProjectTransform. jsp 页面部分处理代码：**

```
double xmin =
Double.parseDouble(String.valueOf(request.getParameter("xmin")));
double ymin =
Double.parseDouble(String.valueOf(request.getParameter("ymin")));
```

```
short BZoneType =
Short.parseShort(String.valueOf(request.getParameter("BZoneType"))
);
//投影长度单位
short DAngUnit =
Short.parseShort(String.valueOf(request.getParameter("DAngUnit")))
;
 //投影中心点经度
 double DLon =
Double.parseDouble(String.valueOf(request.getParameter("DLon")));
 short IProjTypeId =
Short.parseShort(String.valueOf(request.getParameter("IProjTypeId"
)));
 //参考椭球体参数类型
 short ISpheroid =
Short.parseShort(String.valueOf(request.getParameter("ISpheroid"))
);
 short IType =
Short.parseShort(String.valueOf(request.getParameter("IType")));
 short IZone =
Short.parseShort(String.valueOf(request.getParameter("IZone")));
 CPrjTran paramProject = new CPrjTran();
    paramProject.setIPointNum((short)1);
    CPoint2D[] thePoint = new CPoint2D[1];
    thePoint[0] = new CPoint2D();
    thePoint[0].setPointX(xmin);
    thePoint[0].setPointY(ymin);
    paramProject.setThePoint(thePoint);
    paramProject.setBZoneType(BZoneType);
    paramProject.setDAngUnit(DAngUnit);
    paramProject.setDLon(DLon);
    paramProject.setIProjTypeId(IProjTypeId);
    paramProject.setISpheroid(ISpheroid);
    paramProject.setIType(IType);
    paramProject.setIZone(IZone);
    //坐标投影结果
    CPrjTranRlt prjRlt = viewport.TransPrj(paramProject);
    CPoint2D[] pnts = prjRlt.getPrjRltPoint();
       out.println("<script type='text/javascript'>alert( '点投影转换
结  果  :  '  +  '"+pnts[0].getPointX()  +"  '+','+'  "+
pnts[0].getPointY()+"')</script>");
```

该页面首先接受传过来的各参数，设置参数对象 paramProject，然后调用服务器类库的 TransPrj()方法实现坐标转换功能。ProjectTransform.jsp 页面的完整代码参见示例站点中的源码。

# 6.2.13 面积周长计算功能

面积周长计算功能实现在客户端输入坐标值字符串，即可计算出该坐标字符串所形成区域的面积和周长。

面积周长计算功能的效果如下图：

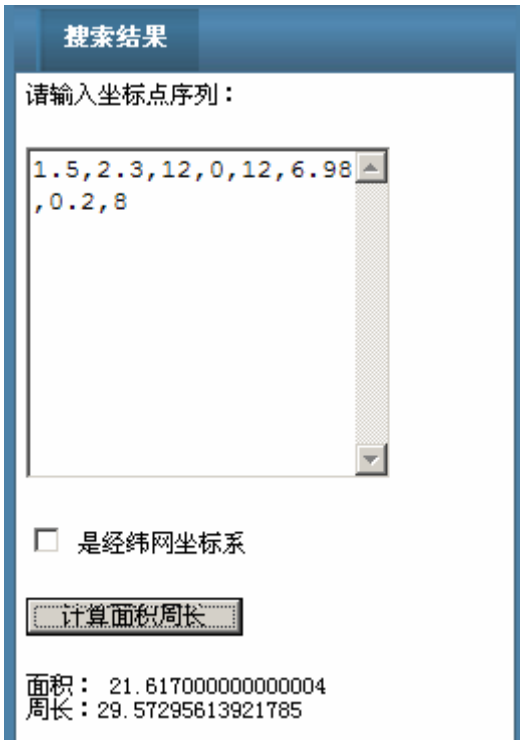



图 6-2-13-1 面积周长计算效果

具体实现流程如下：

1. 在客户端 menu.htm 页面添加面积周长计算的功能按钮，进行事件编写及绑定。

➢ **面积周长计算功能按钮的代码：**

```
<img class="btncursor"
onclick="top.onSelectToolClick('calculateArea',null)"  alt="计算面积" src="images/searchable.gif" align="middle" border="0"><br>
    计算面积<br>
```

2. 面积周长计算功能按钮的 onclick 事件触发 onSelectToolClick 脚本函数进行处理，该函数位于脚本库 function.js 中。

➢ **onSelectToolClick()面积周长计算部分代码：**

```
function onSelectToolClick(activeTool, drawAction, TargetPage) {
 userParam = TargetPage;
 userActive = activeTool;
 switch (userActive) {
   case "calculateArea":
     var url = "VectorPage/CalculateInput.html?rnd=" + Math.random();
     renderSelectResult(url);
     break;
     }
```

```
}
```

该函数将转到 CalculateInput.html 页面中进行面积周长计算的坐标值输入及相关属性设置。

3. CalculateInput.html 页面，用于实现客户端的坐标值输入和属性设置操作，然后通过计算面积周长按钮触发点击事件的 calculate()脚本函数，由该函数进行处理。

CalculateInput.html 页面中面积周长计算的设置面板如下图：




图 6-2-13-2 面积周长计算设置面板

➢ calculate()脚本函数代码：

```
function calculate(){
          var pntArr = document.getElementById("pntArr").innerText;
          var isLatLong =
document.getElementById("isLatLong").checked;
          var url =
"VectorPage/Calculate.jsp?pntArr="+pntArr+"&isLatLong="+isLatLong;
          var rlt = top.AJAXRequest(url,false,"GET");
          var info = rlt.split(",");
          if(isNaN(info[0])){
             alert(info);
             return;
          }
          document.getElementById("calculateRlt").innerHTML = "面积:
"+info[0]+"<br/>周长: "+info[1];
        }
```

该脚本函数从客户端获取相关参数，然后以 AJAX 方式转到服务器后台页面 Calculate.jsp 中进行处理，并返回结果。

4. Calculate.jsp 后台页面，实现面积周长计算功能。

➢ Calculate. jsp 页面部分处理代码：

```
try{
boolean isLatLong =
Boolean.parseBoolean(request.getParameter("isLatLong"));
String[] pntArrStr = request.getParameter("pntArr").split(",");
List<CPoint2D> dotList = new ArrayList<CPoint2D>();
for(int i = 0,n = pntArrStr.length/2;i < n;i++){
    double x = Double.parseDouble(pntArrStr[2*i]);
    double y = Double.parseDouble(pntArrStr[2*i+1]);
    CPoint2D dot = new CPoint2D(x,y);
    dotList.add(dot);
}
MapViewPort viewport =
(MapViewPort)session.getAttribute("viewport");

if(isLatLong){
    convertToDescartes(dotList,viewport);
}
double area = calculateArea(dotList);
double perimeter = calculatePerimeter(dotList);
out.print(area + "," + perimeter);
}catch(Exception e){
out.print("面积计算有误，请重试；每个数值间用逗号隔开");
   }
```

该页面首先接收传过来的参数，然后调用相关方法实现计算坐标值串形成区域的面积和周长功能。Calculate.jsp 页面的完整代码参见示例站点中的源码。