

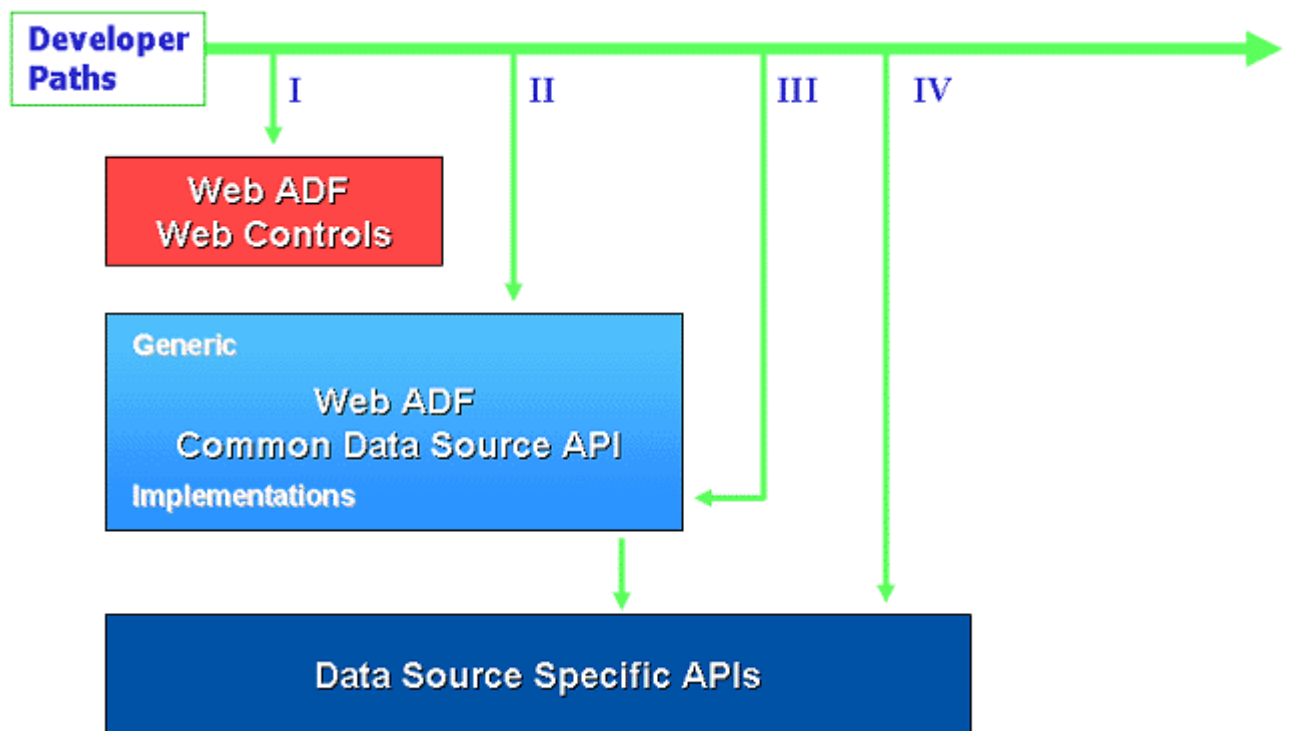
ArcGIS Server 开发系列（一）--编程框架总览

作者: Flyingis

文章仅用于学习与交流，禁止用于任何商业目的<http://flyingis.cnblogs.com/>

ArcGIS Server是一个发布企业级GIS应用程序的综合平台，如果想对ArcGIS Server基础有更多的了解，可以先看看《ArcGIS Server 体系结构》、ESRI(中国)BBS论坛和ESRI网站。这里及接下来的一系列文章将主要介绍ArcGIS Server开发相关的内容（如不加说明，后文中Server专指ArcGIS Server），由浅入深，充分利用Server的资源来构建我们的企业级应用，同时也可以了解Server的优势及不足，在实际应用中扬长避短。

进行 Server 开发之前，我们首先需要了解 Server 开发的整体架构。下图（截取自官方文档）可以看到，Server 提供了一系列的应用开发途径，从 1 到 4 难度逐步增加，当然功能和可定制性也越来越强，我们可以充分利用 Server 提供的这些资源来构建我们的应用。



Web应用开发

Server 为 Web 应用开发提供了一系列的开发方式，在 visual studio 2005 中可以直接使用 Web Mapping Application template 建立应用，它包含了 Web ADF 框架，是学习 Server 开发的入口。对照上面的示意图，可以这样来理解，如果想使用各种 Web 控件加上少量代码来构建 Web 应用，可以直接使用第一种途径，这样的应用比较死板，让人感觉有点想搭积木，程序员可能没有任何成就感，好处是开发迅速，适合入门学习或做个简单的 Server 演示。如果想

在 Common API 基础上，使用各种通用的 functionality，通过各种 data source 类型来定制 Web ADF，可以使用第二种开发途径。如果想使用 Common API，并且需要通过 Common API 调用 specific API 来定制 Web ADF，可以通过第三种途径来开发。

Web Service开发

Web Service 是什么不用再介绍，ArcGIS 提供了两种类型的 Web Service 创建方法: GIS Web Service 和 Application Web Service。

GIS Web Service 提供了一种将 ArcGIS Server Object (Local data source) 发布为 ArcGIS Server Web Service (Internet data source) 的 ESRI 标准，GIS Web Service 不用于开发，通常它们是用来发布信息和提供资源，ArcMap 就可以直接使用 GIS Web Service 的资源而不用进行任何开发，另外 Web ADF 控件和 Common API 也可以使用 GIS Web Service 资源。因为 GIS Web Service 基于标准 web service，它可以作为传统 web service 来使用，ArcGIS Server 提供了 SOAP API 进行相关的开发，以后的学习中会使用到。

Application Web Service 是基于标准 web service 建立的应用，使用一种 ESRI 的 data source 进行开发。因为 web service 没有用户接口，上图中途径一不适合这样的应用，途径四是最适合的，因为 web service 可充分利用 ESRI 提供的各种 data source specific API 的所有 functionality。

桌面客户端应用开发

ESRI 在桌面客户端应用中提供了两种实现 GIS 功能和服务的方法: ArcGIS Engine 和 data source specific developer APIs，上图中只有第四种途径可用于桌面客户端应用。

ArcEngine 不论是和本地 GIS 资源交互，还是使用远程数据资源，都拥有丰富的控件和众多 API，但它在使用和分发之前必须进行授权注册，而任何 data source specific developer APIs 都可以在桌面客户端应用中编程使用。

移动应用开发

移动应用和 Web 应用、web services 比较类似，但它们是某一个特定的客户端环境而设计，如 PocketPC。Mobile ADF 是为移动应用定制的开发框架，通常使用途径四来进行开发，这样可以充分利用 ArcGIS Server Web services 和 SOAP API 提供的功能。

在以后的实际开发过程中，我们会逐步了解上述各种开发方式，并不断深入。

ArcGIS Server 开发系列（二）--Web ADF 编程

作者: [Flyingis](#)

Web ADF全名是Web Application Develop Framework，是ArcGIS Server专用的开发框架，9.2 版本的ArcIMS也开始提供ADF的封装。完全使用Web控件编程几乎不能解决我们实际项目中的问题，因此我们从《[ArcGIS Server 开发系列（一）--编程框架总览](#)》中提到的途径二开始，即web controls + common datasource apis。

目标:

根据查询语句实现图层信息查询

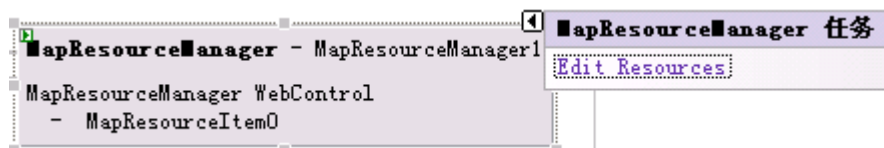
准备工作:

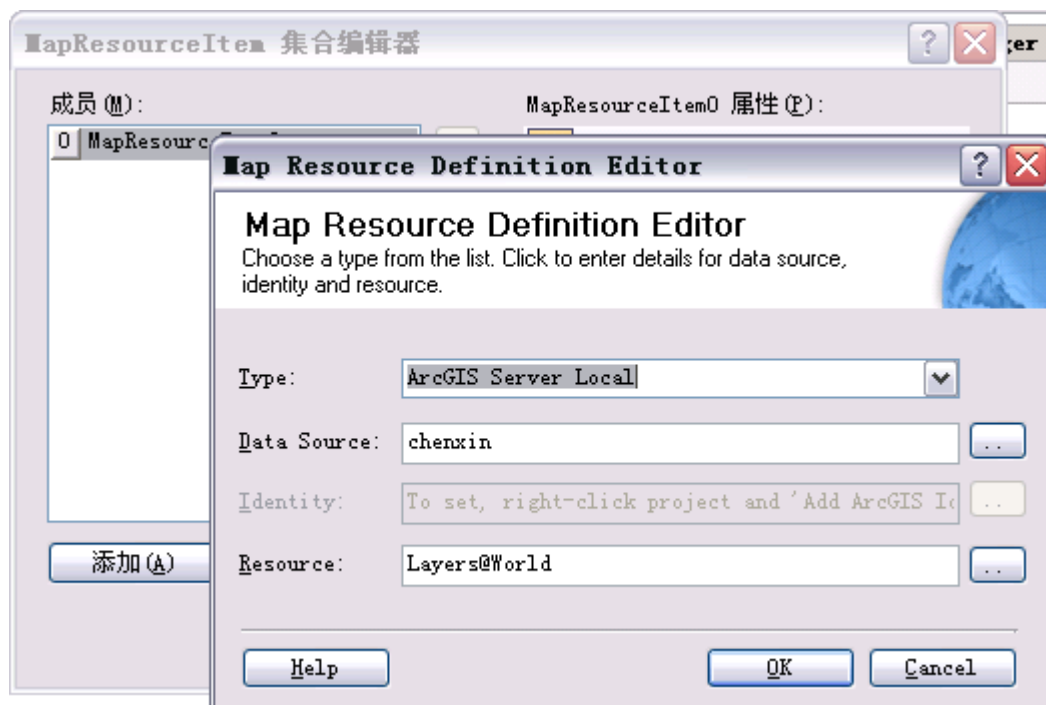
1.Windows XP sp2 中/英文版

2.Visual Studio 2005 中/英文版

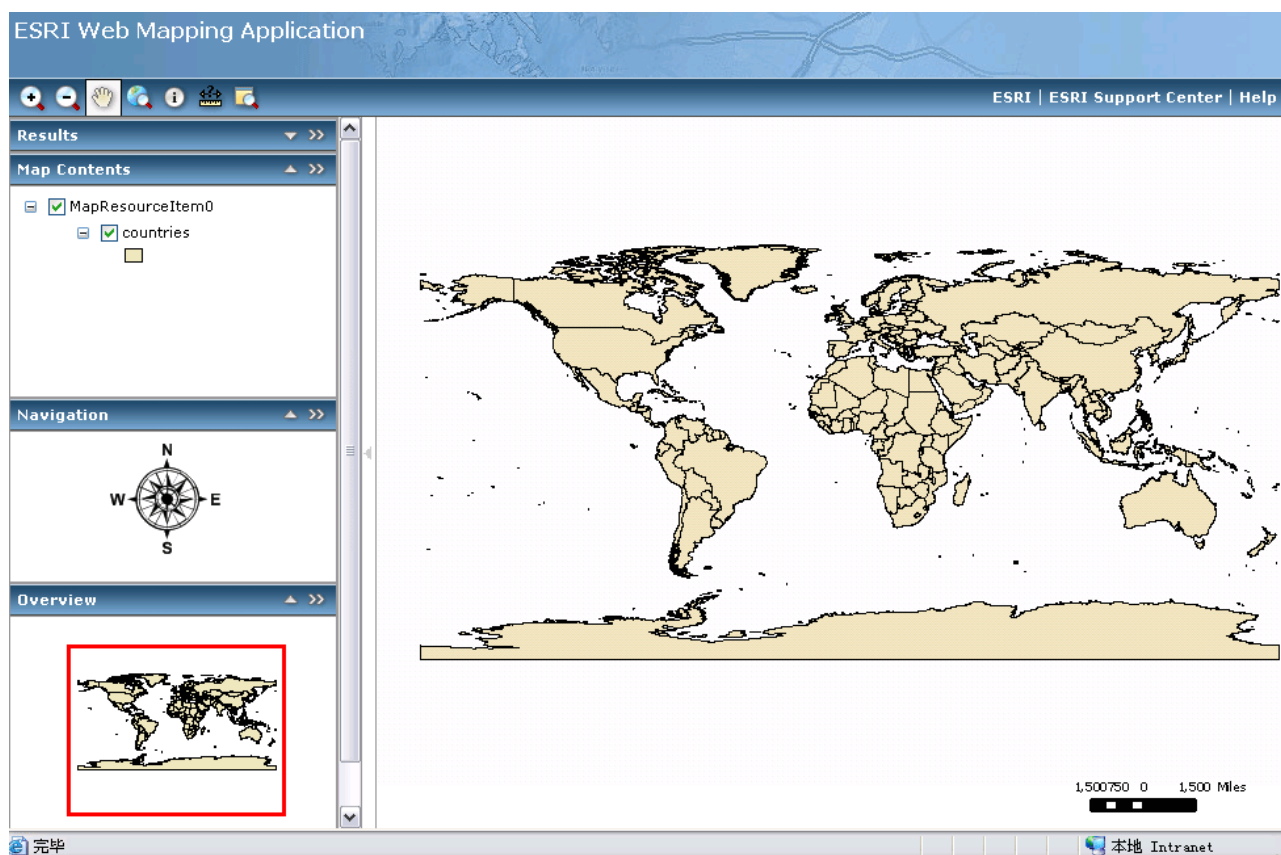
3.ArcGIS Server 9.2(我打上了 sp2 补丁)，创建一个 ArcGIS Server 服务，Map Server 属性中更改 Pooling，使用池化连接方式，Progresses 选择“In a seperate process for each instance(high isolation)”，我的服务为“World”

4.利用 vs2005 创建一个模板 server 应用--Web Mapping Application，并更改 MapResourceManager 属性



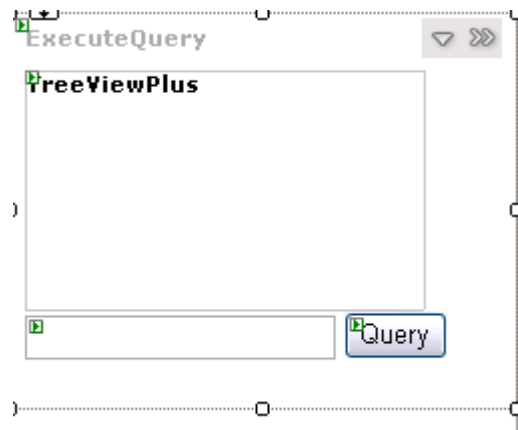


这样我们可以测试程序看是否能够正常运行，启动调试看到如下结果：



代码实现：

我们要完成的工作是对图层属性信息的查询，首先在页面左侧添加一个新的 panel，左侧的 Tasks、Results 等都是放置在 panel 之中，在这个新的 panel 中分别添加 TreeViewPlus、TextBox、Button 三个控件，TextBox 输入查询条件，Button 确定查询， TreeViewPlus 显示结果，在设计试图中浏览效果如下：



双击“Query”Button，编写点击事件。

```
protected void cmdQuery_Click(object sender, EventArgs e)
{
    Query(Map1.Extent);
}
```

Query 方法实现了对图层属性信息的查询。

```
protected void Query(ESRI.ArcGIS.ADF.Web.Geometry.Geometry geometry)
{
    IEnumerable func_enum = null;
    //获取当前 map1 控件中所有的 functionality
    func_enum = Map1.GetFunctionalities();

    System.Data.DataTable datatable;
    //对所有的 functionality 进行遍历
    foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gisfunction
```

```

ality in func_enum)
{
    |     ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource = null;
    |
    |     //得到该 functionality 的 resource
    |
    |     gisresource = gisfunctionality.Resource;
    |
    |
    |     //判断该 resource 是否支持 IQueryFunctionality
    |
    |     bool supported = false;
    |
    |     supported = gisresource.SupportsFunctionality(typeof(ESRI.ArcGIS.ADF.
Web.DataSources.IQueryFunctionality));
    |
    |
    |     if (supported)
    |
    |     {
    |
    |         ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality qfunc;
    |
    |         qfunc = (ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality)gis
resource.CreateFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFu
nctionality), null);
    |
    |
    |         string[] lids;
    |
    |         string[] lnames;
    |
    |         //获得图层的 layerId 和 layerName, GetQueryableLayers 的重载方法可以
指定图层类型
    |
    |         qfunc.GetQueryableLayers(null, out lids, out lnames);
    |
    |
    |         ESRI.ArcGIS.ADF.Web.SpatialFilter spatialfilter = new ESRI.ArcGIS.A
DF.Web.SpatialFilter();
    |
    |         //设置过滤器的过滤条件, txtQuery 就是 panel 中 text box 的 ID
    |
    |         spatialfilter.ReturnADFGeometries = false;
    |
    |         spatialfilter.MaxRecords = 1000;
    |
    |         spatialfilter.WhereClause = txtQuery.Text;

```

```

|         spatialfilter.Geometry = geometry;
|
|         //对指定的图层进行查询，查询的结果保存为 DataTable
|
|         datatable = qfunc.Query(null, lids[0], spatialfilter);
|
|         TreeViewPlus1.Nodes.Clear();
|
|         if (datatable != null)
|         {
|             System.Data.DataSet ds = new System.Data.DataSet();
|             ds.Tables.Add(datatable);
|
|             TreeViewPlus1.ShowClearAllButton = false;
|
|             //将结果绑定到 TreeViewPlus 控件上
|
|             TreeViewPlus1.BindToDataSet(ds);
|
|             TreeViewPlus1.Nodes[0].Expanded = true;
|         }
|     }
| }
| }
| }
| }

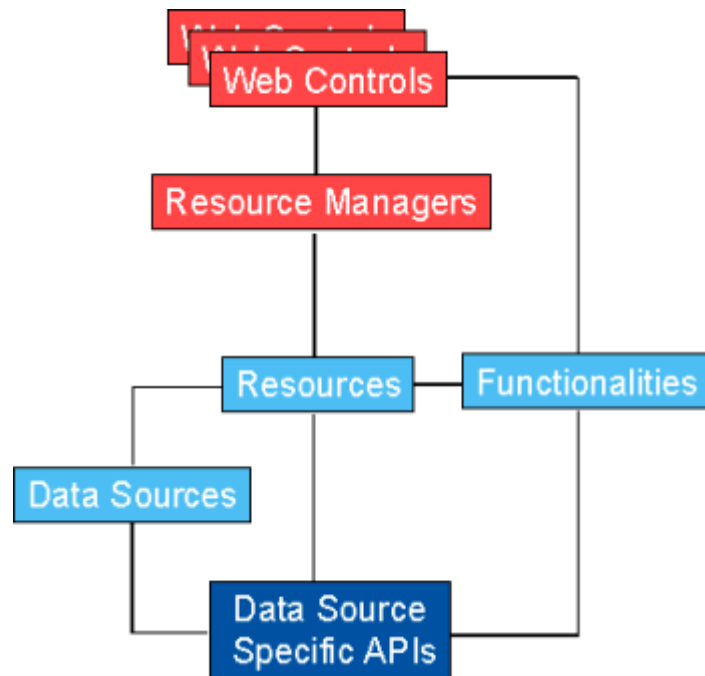
```

运行程序，text box中输入“CNTRY_NAME LIKE 'A%’”，查询结果：



这样这个小程序就完成了，在server中实现了对图层属性信息的查询，现在我们需要进行更多的考虑：

第一，考虑web控件、 functionality、datasource、resource、resourcemanager之间的关系，代码中多次对它们的关系进行了描述，用文档中的一幅图可以概括整个框架，这篇文章对此进行了阐述--《ArcGIS Server .Net Web ADF体系结构》。



第二，程序中使用的mxd仅仅包含一个world图层，因此对图层进行遍历的过程中只需获得第一图层的ID，如果指定图层名，可以在Inames得到所需的图层进行查询。

第三，程序实现的查询方式相当简单，在 text box 中输入原始的 sql 来完成 where 语句，只仅仅是一个测试，一般我们会屏蔽掉 sql 相关的关键字，提供多个 text box 或 dropdownlist 供用户选择进行字段查询。

第四，如果查询信息量不大，返回结果数量不多，完全可以考虑 ajax 来完成页面请求，不论是速度还是用户体验都会得到提高。

第五，通过程序可以看到，查询结果保存在一个 datatable 中，这极大的方便了 server 查询和.net 的无缝连接，直接将结果和 TreeViewPlus 绑定，换成其他的 visual studio 控件也没问题。

只要整个框架思路清晰了，查询功能的实现其实很简单，这也是途径 2 开发方式的一个例子，可以看出 ArcGIS Server ADF 给开发提供了很多便利，和 ArcIMS 相比开发难度相当，多看看文档和 ADF OMD 图可以让我们对 ArcGIS Server ADF 开发有更多的了解，记住，这只是 server 开发的起步。

ArcGIS Server 开发系列（三）--漫游 Graphics data sources

作者: Flyingis

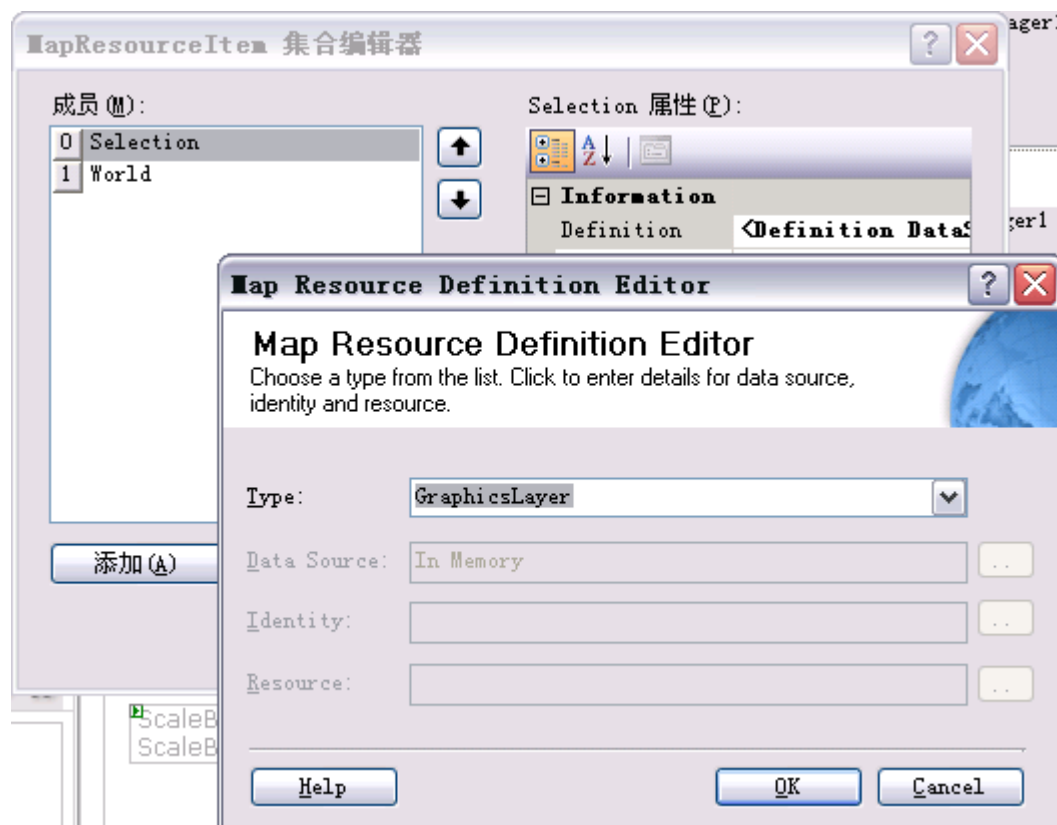
和ArcGIS Server Local、ArcGIS Server Internet一样，GraphicsLayer是ArcGIS Server MapResource的一种，提供functionality给web controls使用。本文将在《ArcGIS Server 开发系列（二）--Web ADF 编程》示例基础上，增加查询结果高亮显示的功能，因为高亮显示的结果并不是图层本身所具备的，因此只需将高亮显示的图片存为graphics即可。

目标:

查询结果的高亮显示

准备工作:

- 1.以《ArcGIS Server 开发系列（二）--Web ADF 编程》示例配置和代码为基础。
- 2.MapResourceManager属性中增加一个名为Selection的MapResource，并将它移动到编号为 0 的位置，即显示在所有MapResource最上面。



可以看到GraphicsLayer的datasource是在内存中的，也就是说是为了临时显示或存储使用的，这样速度比较快。Selection一定要放在World上面，否则就被World图层覆盖掉了。

代码实现：

在 UI 界面上，增加一个 command，用来清除 graphics。



双击“Select”生成事件响应方法：

```
protected void cmdSelect_Click(object sender, EventArgs e)
{
    SelectFeatures();
}
```

代码的核心就在 `SelectFeature()` 里，它分为两个步骤，第一步对图层进行属性查询，第二步对查询结果进行高亮显示。首先是图层的属性查询：

```
int resource_index = 1;
string targetlayername = "countries";
System.Data.DataTable datatable = null;
//直接获取 MapResourceName 为 world 的 MapFunctionality，它的编号为 1

ESRI.ArcGIS.ADF.Web.DataSources.IMapFunctionality mf = (ESRI.ArcGIS.ADF.Web.DataSources.IMapFunctionality)Map1.GetFunctionality(resource_index);
//先得到 functionality，再获取 resource
ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource = mf.Resource;

bool supported = gisresource.SupportsFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality));
if (supported)
{
    ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality qfunc;
```

```

|   qfunc = (ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality)gisresource
|   .CreateFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionali
|   ty), null);
|
|   string[] lids;
|   string[] Inames;
|   qfunc.GetQueryableLayers(null, out lids, out Inames);
|
|   ESRI.ArcGIS.ADF.Web.SpatialFilter spatialfilter = new ESRI.ArcGIS.ADF.Web.
SpatialFilter();
|   spatialfilter.ReturnADFGeometries = false;
|   spatialfilter.MaxRecords = 1000;
|   spatialfilter.WhereClause = txtQuery.Text;
|
|   datatable = qfunc.Query(null, lids[0], spatialfilter);
| }

```

这段代码和《ArcGIS Server 开发系列（二）--Web ADF 编程》示例中的代码相比，没有太多改动的地方，用到了 ESRI.ArcGIS.ADF.Web.DataSources.IMapFunctionality，它继承于 ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality 接口。每一个 web control 可能拥有多个 functionality，而 functionality 是各种 resource 展现出来的，因此可以通过 web controls--functionalities--resources 这条路线来获得当前的资源，那么如何让查询结果高亮显示呢？

```

//重新获得 Map1 控件所有的 functionality
IEnumerable gfc = Map1.GetFunctionalities();
ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource gResource = null;
foreach (IGISFunctionality gfunc in gfc)
{
|   //找到名为"Selection"的 MapResource
|   if (gfunc.Resource.Name == "Selection")

```

```

    {
        //down cast 到 ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource
        gResource = (ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource)
gfunc.Resource;
    }
}

if (gResource == null)
    return;

ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer glayer = null;

foreach (System.Data.DataTable dt in gResource.Graphics.Tables)
{
    if (dt is ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer)
    {
        glayer = (ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer)d
t;
        break;
    }
}

if (glayer == null)
{
    glayer = new ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer(
);
    gResource.Graphics.Tables.Add(glayer);
}

```

```

//清除已有数据

glayer.Clear();

DataRowCollection drs = datatable.Rows;

int shpind = -1;
for (int i = 0; i < datatable.Columns.Count; i++)
{
    if (datatable.Columns[i].DataType == typeof(ESRI.ArcGIS.ADF.Web.Geometry.Geometry))
    {
        //找到 Geometry 字段的序号
        shpind = i;
        break;
    }
}

try
{
    foreach (DataRow dr in drs)
    {
        ESRI.ArcGIS.ADF.Web.Geometry.Geometry geom = (ESRI.ArcGIS.ADF.Web.Geometry.Geometry)dr[shpind];

        //创建一个 GraphicElement
        ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement ge = null;
        ge = new ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement(geom,
System.Drawing.Color.Yellow);

        ge.Symbol.Transparency = 50.0;
    }
}

```

```

|         //将 GraphicElement 添加到 ElementGraphicsLayer 中
|         glayer.Add(ge);
|     }
| }
| }

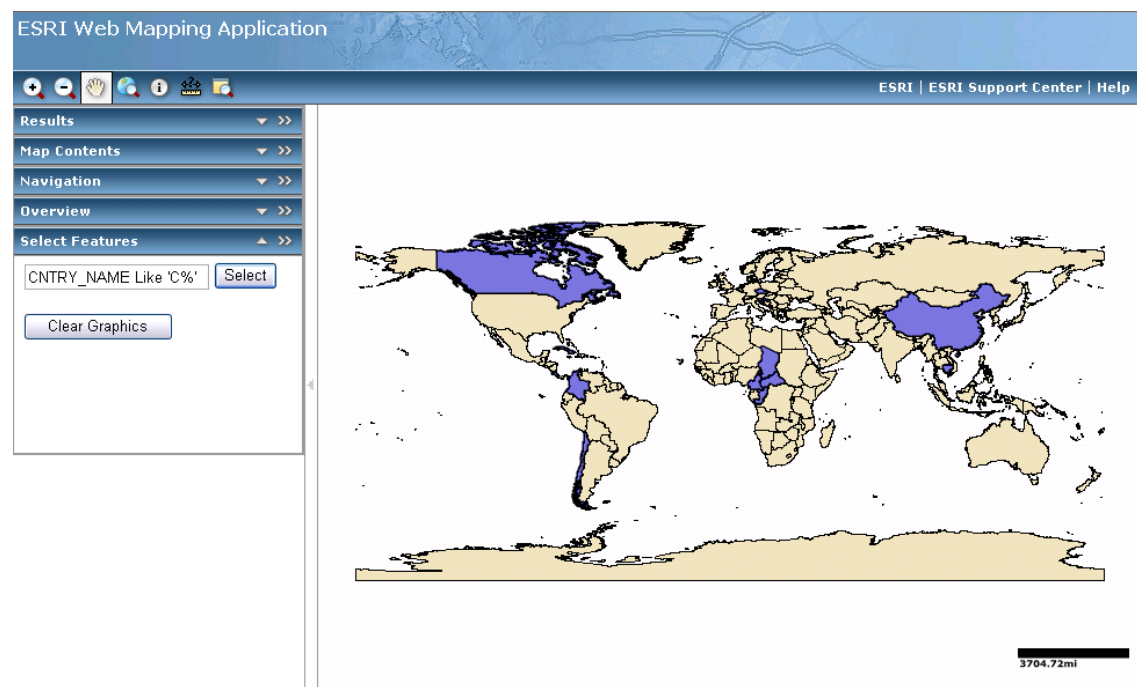
catch (InvalidCastException ice)
{
|     throw new Exception("No geometry available in datatable");
| }

if (Map1.ImageBlendingMode == ImageBlendingMode.WebTier)
{ Map1.Refresh(); }

else if (Map1.ImageBlendingMode == ImageBlendingMode.Browser)
{
|     //只刷新 Graphics Resource
|     Map1.RefreshResource(gResource.Name);
| }

```

这次我们没有将搜索到的结果绑定到控件上，只要得到高亮显示的结果，测试一下程序，看看能得到什么样的效果。



搜索出国家名称以"C"开头的国家，最典型的“中国”、“加拿大”已经找到了，这样我们就实现了高亮显示的功能。同样，我们进行开发后的小结，能想到些什么呢？还是按照 CH 风格来进行总结：

第一，GraphicsLayer 有两个子类，ElementGraphicLayer 和 FeatureGraphicLayer，因为程序中只需要暂时显示查询的结果，因此将查询要素存为 ElementGraphicLayer 就可以，想想在什么情况下使用 FeatureGraphicLayer。

第二，ElementGraphicLayer 继承于 System.Data.DataTable，gResource.graphics 属于 System.Data.DataSet 类型，这样使得我们在开发过程中，可以将 GraphicElement 添加到 ElementGraphicLayer，然后将 ElementGraphicLayer 添加到 gResource.graphics，通过这种途径来向 GraphicsLayer 的 mapresource 中添加数据，这种机制方便了我们能够像操纵 datatable 和 dataset 一样来控制 mapresource 中的数据，既和 .Net 无缝整合，也在一定程度上降低了 Server 开发难度，例如代码中 glayer.Clear() 调用了 datatable 的 clear() 方法，还有后面 GraphicElement 的创建。

第三，Map1.ImageBlendingMode 决定了地图的刷新是刷新整个页面，还是仅刷新当前 mapresource，这样的设计在 web 开发中尽可能的较少了网络数据传输量。

Graphics data sources 是学习 ArcGIS Server data sources 的基础，下面一篇，将介绍 ArcGIS Server data sources 的开发，之前网上已经有朋友要求加快写作进度了，不过日常工作中的琐事实实在比较多，写代码、文章经常会被打断，写的太差又对不住大家花的时间，所以只能尽量以最快的速度写好每一篇博客，大家的支持就是我的动力:)这篇到此为止，写完收工，回家过周末～～

ArcGIS Server 开发系列（四）--ArcGIS Server data sources 开发

作者: [Flyingis](#)

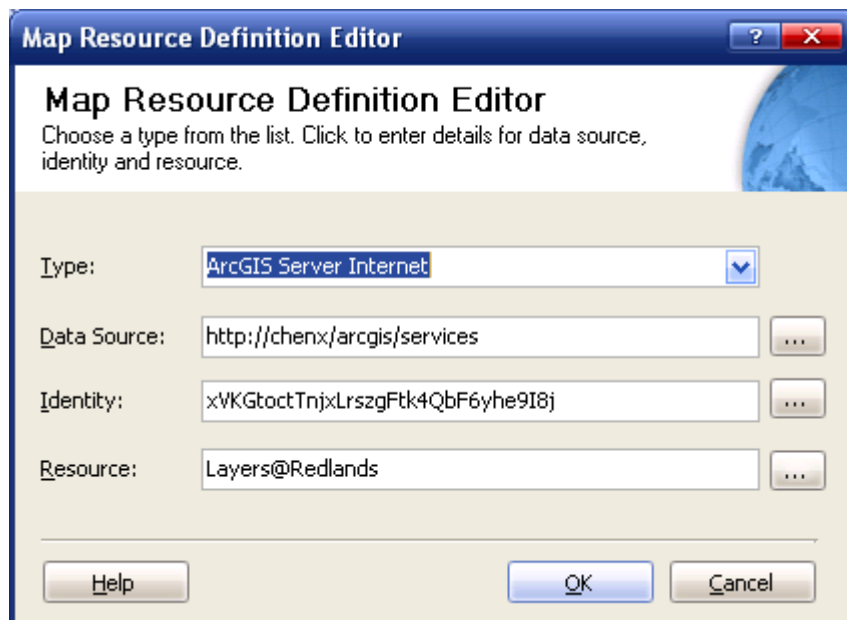
ArcGIS Server 开发系列前几篇文章搭建了一个基本的 webgis 开发框架，包括模板应用程序搭建、属性查询、查询结果高亮显示，在 arcims 中，不论是使用 9.2 之前的 javaconnector、.net_link、htmlviewer、还是 9.2 里的 adf，这些功能都可以轻松实现，从软件成本上来看，ArcGIS Server 企业版要比 ArcIMS 高出不少，如何体现 ArcGIS Server 价值呢？我们就从这一篇开始挖掘 ArcGIS Server 的价值，从 ArcGIS Server data sources 启航.....

目标：

对点要素进行缓冲区分析

准备工作：

- 1.了解 ArcGIS Server 中的 ValueObject 和 ComObject。
- 2.AO 接口中缓冲区分析的编程方法。
- 3.重新回顾第一篇中 ArcGIS Server 开发的四种方法。
- 4.利用 vs2005 创建一个模板 server 应用--Web Mapping Application，并更改 MapResourceManager 属性。



注意这里使用的 map resource 类型是 ArcGIS Server Internet，在界面上增加一个新的 panel，里面包括两个 textbox 和一个 comand，textbox 对应的分别是缓冲区中心点的 x、y 坐标，以该点为中心，一定半径做圆形缓冲区。



思路:

缓冲区分析需要在 AO 接口中实现, 输入的点应该是一个 COM 对象, 而在页面中输入 xy 坐标点是一个 SOAP API valueobject, valueobject 可以用于 adf web controls, 但不能用于 AO 接口, 因此需要进行 valueobject 到 comobject 的转换, 在调用 ITopologicalOperator 接口的 Buffer 方法后会得到缓冲区分析的结果, 即一个 polygon, 同理, 该 polygon 是一个 comobject, 需要逆转换为 valueobject 才能显示到网页上。

代码实现:

新建一个类 XYBuffer, 缓冲功能的在其 buffer 方法中实现。首先设置 textbox 中输入点的渲染方式, 该作为一个 ESRI.ArcGIS.ADF.ArcGIServer.PointN 对象, 渲染方法如下:

```
ESRI.ArcGIS.ADF.ArcGIServer.PointN pt = new ESRI.ArcGIS.ADF.ArcGIServer.  
PointN();  
  
pt.X = x;  
pt.Y = y;  
  
// 设置点的颜色  
  
ESRI.ArcGIS.ADF.ArcGIServer.RgbColor rgb = new ESRI.ArcGIS.ADF.ArcGISSe  
rver.RgbColor();  
  
rgb.Red = 0;  
rgb.Blue = 0;  
rgb.Green = 20;  
  
// 设置点的符号  
  
ESRI.ArcGIS.ADF.ArcGIServer.SimpleMarkerSymbol sms = new ESRI.ArcGIS.A  
DF.ArcGIServer.SimpleMarkerSymbol();  
  
sms.Style = ESRI.ArcGIS.ADF.ArcGIServer.esriSimpleMarkerStyle.esriSMSCircl  
e;
```

```

sms.Color = rgb;

sms.Size = 20;


ESRI.ArcGIS.ADF.ArcGISServer.MarkerElement marker = new ESRI.ArcGIS.ADF.
ArcGISServer.MarkerElement();

marker.Symbol = sms;

marker.Point = pt;

```

然后用 arcgis server local 方式建立到 datasource 的连接，这点非常重要，主要是为了在这种连接状态下进行 valueobject 和 comobject 之间的转换。建立连接的用户应属于 ArcGIS Server 管理组。

```

ESRI.ArcGIS.ADF.Identity identity = new ESRI.ArcGIS.ADF.Identity("user", "pas
sword", "localhost");

ESRI.ArcGIS.ADF.Connection.AGS.AGSServerConnection agsconnection;

agsconnection = new ESRI.ArcGIS.ADF.Connection.AGS.AGSServerConnection("
localhost", identity);

agsconnection.Connect();

ESRI.ArcGIS.Server.IServerObjectManager som = agsconnection.ServerObjectM
anager;

ESRI.ArcGIS.Server.IServerContext serverContext = som.CreateServerContext
("Redlands", "MapServer");

```

这样我们就可以在 arcgis server local 连接方式下进行对象转换：

```

// 定义 COM 对象的点

ESRI.ArcGIS.Geometry.IPoint ipnt;

// 进行 valueobject 到 comobject 之间的转换

ipnt = (ESRI.ArcGIS.Geometry.IPoint)ESRI.ArcGIS.ADF.Web.DataSources.ArcGI
SServer.Converter.ValueObjectToComObject(pt, serverContext);

```

下面是 AO 中缓冲区分析的代码，熟悉 AO 编程的对下面代码应该很了解了：

```

ESRI.ArcGIS.Geometry.ITopologicalOperator topop = (ESRI.ArcGIS.Geometry.I
TopologicalOperator)ipnt;

double bufferDistance = map.Extent.Width / 6;

```

```
ESRI.ArcGIS.Geometry.IPolygon bufferPolygon;  
bufferPolygon = (ESRI.ArcGIS.Geometry.IPolygon)topop.Buffer(bufferDistance);
```

bufferPolygon 就是缓冲区分析的结果，但它还不是我们最后想要的，因为

ESRI.ArcGIS.Geometry.IPolygon 无法在 adf web control 中显示，还需要做一次转换：

```
// 定义 valueobject 的点  
ESRI.ArcGIS.ADF.ArcGISServer.PolygonN buffer_polyn;  
// 进行 comobject 到 valueobject 之间的转换  
buffer_polyn = (ESRI.ArcGIS.ADF.ArcGISServer.PolygonN)ESRI.ArcGIS.ADF.Web.  
DataSources.ArcGISServer.Converter.ComObjectToValueObject(bufferPolygo  
n, serverContext, typeof(ESRI.ArcGIS.ADF.ArcGISServer.PolygonN));
```

buffer_polyn 就是我们最后想要的结果，定义一种渲染方式：

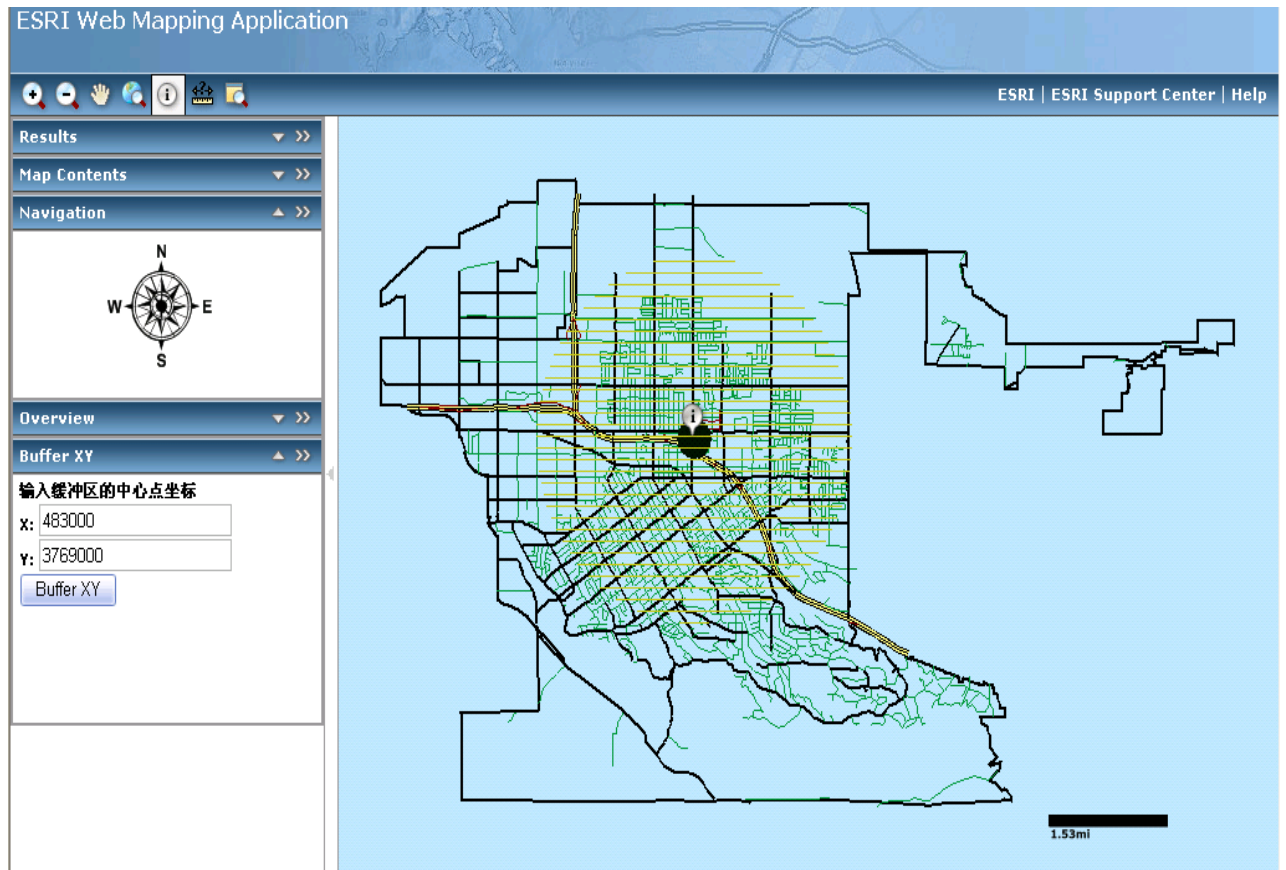
```
ESRI.ArcGIS.ADF.ArcGISServer.RgbColor rgb1 = new ESRI.ArcGIS.ADF.ArcGISS  
erver.RgbColor();  
rgb1.Red = 200;  
rgb1.Green = 200;  
rgb1.Blue = 20;  
// 设置区的填充色  
ESRI.ArcGIS.ADF.ArcGISServer.SimpleFillSymbol sfs1 = new ESRI.ArcGIS.ADF.  
ArcGISServer.SimpleFillSymbol();  
sfs1.Style = ESRI.ArcGIS.ADF.ArcGISServer.esriSimpleFillStyle.esriSFShorizont  
al;  
sfs1.Color = rgb1;  
  
ESRI.ArcGIS.ADF.ArcGISServer.PolygonElement polyelement1 = new ESRI.ArcG  
IS.ADF.ArcGISServer.PolygonElement();  
polyelement1.Symbol = sfs1;  
polyelement1.Polygon = buffer_polyn;
```

最后将 marker 和 polyelement1 添加到

ESRI.ArcGIS.ADF.ArcGISServer.GraphicElement 对象数组中，传给当前

MapFunctionality 的 CustomGraphics 属性，刷新 map 控件，看看显示的效果图，缓冲区

显示的样式为一系列水平的平行线：[A](#)



以上在 ArcGIS Server 中实现了缓冲区分析功能，虽然 arcims 也能实现（arcims 分析功能也仅限于此），但是两者的本质有天然的差别，前者是基于 AO，因此类推可以将 AO 中的分析功能全部引入 server 中，实现更多更复杂的分析功能。

最后，我们还有哪些需要考虑的呢？

1. ESRI.ArcGIS.ADF.ArcGISServer 命名空间中我们用到了 PointN 类，但同样可以找到 PointB 类，对于其他几何类型也是如此，如 PolylineN 和 PolylineB，它们之间有什么差别呢？
2. 整个开发过程我们用到了 Data Source Specific API，重新回想一下第一篇中提到的途径三和途径四两种开发方式。
3. 例子中我们仅仅是将缓冲区显示出来，如果需要用缓冲区做进一步的分析，如一条街道向两侧拓展 3 米，有哪些房屋或建筑需要拆除或改建呢？这时需要做进一步的相交分析，同样可以调用 AO 接口实现，最后将结果转换为 valueobject 显示出来，这样可以在 server 中实现决策分析的功能模块。当然我们可以通过这个例子做更多更深入的延伸。
4. 如何将显示的结果清除掉？

ArcGIS Server 开发系列（五）--自定义 Toolbar 工具

作者：Flyingis

前面的开发系列均是使用 server 开发模板程序 Web Mapping Application，工具条上的基本工具是已经在模板中定制好的，在实际项目应用中，我们需要的工具远远不仅如此，如何在工具条中增加新的自定义工具是开发系列（五）所要描述的，其中使用 ASP.Net 2.0 Callback framework 进行异步刷新地图是重点。

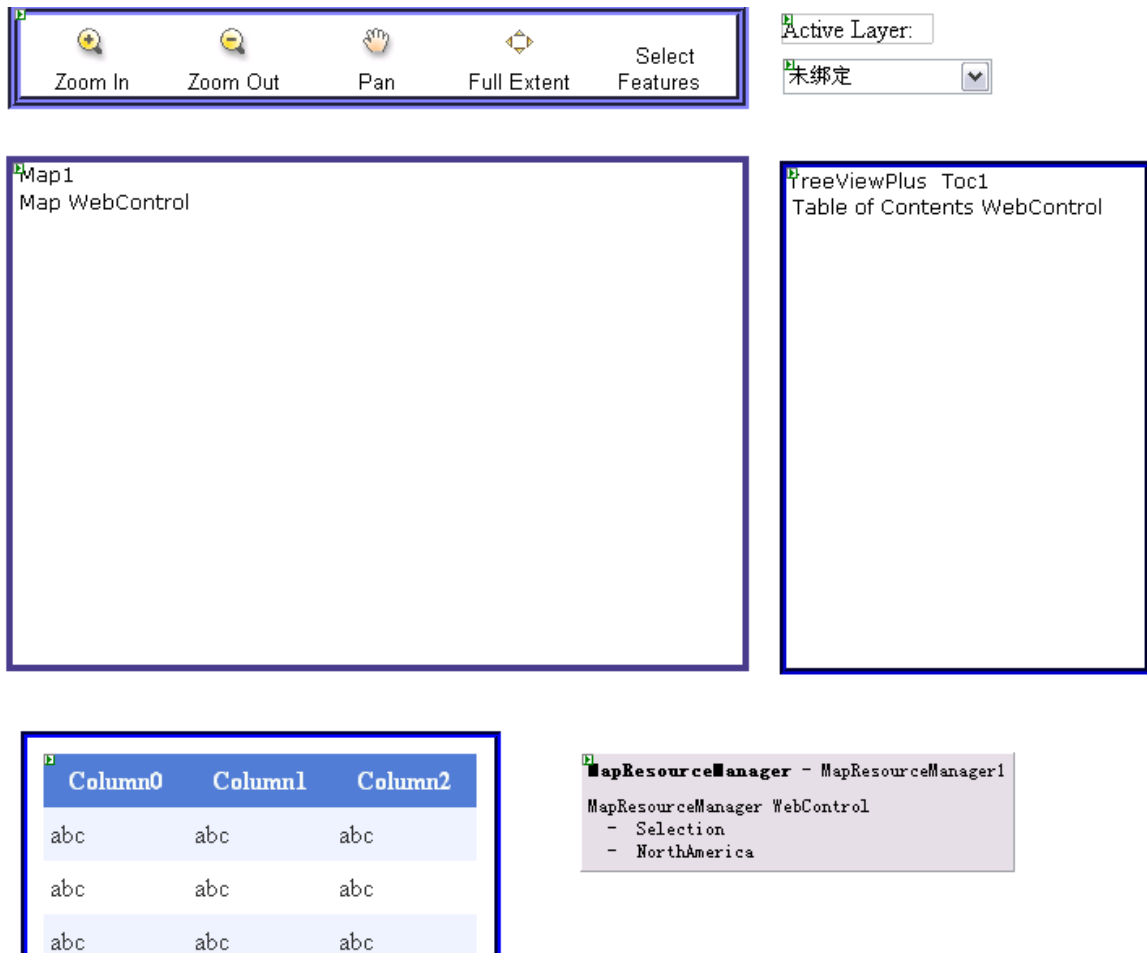
目标：

自定义工具按钮进行矩选查询，高亮显示所选择的地图要素，页面下方的 Gridview 显示所选择要素的属性信息。

准备工作：

- 1.了解 ESRI.ArcGIS.Server.WebControls.IMapServerToolAction 接口
- 2.了解 ASP.Net 2.0 Callback framework
- 3.新建一个网站，在 ArcGIS Web Controls 控件中拖动如下控件：Toolbar、Map、Toc、MapResourceManager，以及常用控件 Label、DropDownList、GridView。
- 4.设置控件属性，Toolbar、Toc 的 BuddyControls 均为 Map1，Toolbar 的 BuddyControlType 为 Map，Map 控件的 MapResourceManager 为 MapResourceManager1。
- 5.更改 MapResourceManager 属性，添加两个 Resource: Selection 和 NorthAmerica，类型分别是 Graphics Layer 和 ArcGIS Server Internet。

最后视图效果：



思路：

现在重新想想我们要做什么，首先要自定义一个工具按钮，使用该工具后在地图上进行矩形选择，对选择的要素高亮显示，同时 **gridview** 显示出这些要素的属性信息。整个过程看似容易，实际上需要在客户端和服务端之间来回切换，异步调用，这里用到了 **ASP.Net Callback framework**，其实 **Server** 中很多地图操作都基于 **asp.net callback**，或是实现了 **ICallbackEventHandler** 接口，理解了这一段程序开发有利于深入了解 **Server** 地图刷新、**Task** 等组件的工作机制。

首先在 **Toolbar** 上新增一个按钮 **Select Features**，上图其实已经加入了，加入的方法是，选择 **Toolbar** 控件属性 **ToolbarItems**，添加一个 **Tool**，设置以下值：

Text: Select Feature

ClientAction: DragRectangle

Name: SelectTool

ServerActionAssembly: App_Code

ServerActionClass: SelectFeatures

OK，搞定！

除了上述属性外，还可以设置该按钮各种状态下的图片显示、ToolTip 等等，这里就省了，纵观这些属性，可以看出既有 js 脚本的交互（已经封装了，通过 ToolEventArgs 传入），也有服务器端功能的实现，这时我们需新建一个类 SelectFeatures，并实现

IMapServerToolAction 接口，类中实现 IMapServerToolAction 的方法 ServerAction。

```
public class SelectFeatures : IMapServerToolAction {  
    public void ServerAction(ToolEventArgs args)  
}
```

代码实现：

1. 获取矩形框的屏幕坐标

要查询矩选的地图信息，首先应知道矩形的坐标，在服务器端如何获取呢？

```
Map mapctrl = null;  
mapctrl = (Map)args.Control;  
  
// 获取下拉框中的数据，在后面实现  
string targetlayername = (string)mapctrl.Page.Session["TargetLayer"];  
  
RectangleEventArgs rectargs = null;  
  
// 强制类型转换为 RectangleEventArgs  
rectargs = (RectangleEventArgs)args;  
  
// 获取矩形选择框的屏幕坐标  
System.Drawing.Rectangle rect = rectargs.ScreenExtent;  
  
ESRI.ArcGIS.ADF.Web.Geometry.Point minpnt = ESRI.ArcGIS.ADF.Web.Geometry  
.Point.ToMapPoint(rect.Left, rect.Bottom, mapctrl.Extent, (int)mapctrl.Width.Value  
e, (int)mapctrl.Height.Value);  
  
ESRI.ArcGIS.ADF.Web.Geometry.Point maxpnt = ESRI.ArcGIS.ADF.Web.Geometry  
.Point.ToMapPoint(rect.Right, rect.Top, mapctrl.Extent, (int)mapctrl.Width.Value  
, (int)mapctrl.Height.Value);
```

```
ESRI.ArcGIS.ADF.Web.Geometry.Envelope mappoly = null;  
  
// minpnt、maxpnt 分别是左下、右上坐标点  
  
mappoly = new ESRI.ArcGIS.ADF.Web.Geometry.Envelope(minpnt, maxpnt);
```

所有的信息都是通过 args 获取，它是一个

ESRI.ArcGIS.ADF.Web.UI.WebControls.ToolEventArgs 对象，包含了客户端 Map 控件和当前客户端工具的信息，RectangleEventArgs 是它的子类，强制性转换后得到矩选的矩形坐标，最后得到一个 Envelope，用于 spatialfilter.Geometry 属性。

2. 查询所选择的要素并对Graphics Layer进行渲染实现高亮

这部分内容完全可以参考《[ArcGIS Server 开发系列（三）--漫游 Graphics data sources](#)》，只需要注释掉WhereClause属性赋值，再增加一行代码：

```
ESRI.ArcGIS.ADF.Web.SpatialFilter spatialfilter = new ESRI.ArcGIS.ADF.Web.SpatialFilter();  
  
spatialfilter.ReturnADFGeometries = false;  
spatialfilter.MaxRecords = 1000;  
  
//spatialfilter.WhereClause = txtQuery.Text;  
spatialfilter.Geometry = mappoly;
```

3. 异步刷新 Gridview 显示地图要素的属性

```
GridView gdview = (GridView)mapctrl.Page.FindControl("GridView1");  
  
object[] oa = new object[1];  
string showtable = "'visible'";  
  
// datatable 为矩选时所选择的地图要素，绑定到 gridview  
gdview.DataSource = datatable;  
gdview.DataBind();
```



```

string returnstring = null;

using (System.IO.StringWriter sw = new System.IO.StringWriter())
{
    | HtmlTextWriter htw = new HtmlTextWriter(sw);
    | gdview.RenderControl(htw);
    | htw.Flush();
    | returnstring = sw.ToString();
    | }

    // innercontent 相当于 innerhtml

    CallbackResult cr = new CallbackResult("div", "griddiv", "innercontent", returnstring);

    // 通过回调将信息从服务器端传输到客户端
    mapctrl.CallbackResults.Add(cr);

    if (datatable.Rows.Count > 1)
    {
        | showtable = "visible";
        | }
        else
        {
            | showtable = "hidden";
            | }

        string sa = "var griddiv = document.getElementById('griddiv');";
        sa += "griddiv.style.visibility = " + showtable + ";";
        oa[0] = sa;

```

```
CallbackResult cr1 = new CallbackResult(null, null, "javascript", oa);  
mapctrl.CallbackResults.Add(cr1);
```

这段代码最关键的类是 `CallbackResult`，它简化了 web adf framework 中客户端回调的处理，不用再创建自己的客户端和服务端逻辑，使用 `CallbackResult` 就可以将信息传回客户端，更新客户端页面的内容、图片或执行 js 脚本。关于 `CallbackResult` 构造方法第三个参数，下面 js 代码写的很详细：

```
if (action=="content") {  
    o = document.getElementById(actions[1]);  
    if (o != null)  
    {  
        o.outerHTML=actions[3];  
    }  
}  
  
else if (action=="innercontent") {  
    o = document.getElementById(actions[1]);  
    if (o != null)  
    {  
        o.innerHTML=actions[3];  
    }  
}  
  
    else if (action=="image")  
    {  
        o = document.images[actions[1]];  
        if (o != null)  
        {  
            o.src = actions[3];  
        }  
        else alert (actions[1] + " was null");  
    }
```

```

else if (action == "javascript") {
    eval(actions[3]);
}

```

4. 填充 DropDownList

DropDownList 显示的是 ArcGIS Server Internet 地图数据源所包含的图层名称，选择哪个图层，矩选时就对哪个图层进行查询，DropDownList 的填充在 Page_PreRender 过程中。

```

if (!IsPostBack)
{
    ESRI.ArcGIS.ADF.Web.DataSources.IMapFunctionality mf = (ESRI.ArcGIS.ADF.Web.DataSources.IMapFunctionality)Map1.GetFunctionality(1);

    ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource = mf.Resource;
    bool supported = gisresource.SupportsFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality));

    if (supported)
    {
        ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality qfunc = (ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality)gisresource.CreateFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality), null);

        string[] lids;
        string[] lnames;
        qfunc.GetQueryableLayers(null, out lids, out lnames);
        for (int i = 0; i < lnames.Length; i++)
        {
            LayerDropDownList1.Items.Add(lnames[i]);
        }
    }
}

```

```

|     Session["TargetLayer"] = LayerDropDownList1.Items[0].Value;
| }
| }
| }

```

5. 实现 ICallbackEventHandler 接口

Default.aspx.cs 的 _Default 实现 ICallbackEventHandler 接口，在类中实现 RaiseCallbackEvent 和 GetCallbackResult 两个方法，做 ASP.Net 2.0 的对 ICallbackEventHandler 应该是再熟悉不过了:)但下来框显示图层为什么要用到 callback?

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Session["TargetLayer"] = "";
    }
    LayerDropDownList1.Attributes.Add("onchange", "ChangeLayer()");
    sADFCallBackFunctionInvocation = Page.ClientScript.GetCallbackEventReference(this, "message", "processCallbackResult", "context", "postBackError", true);
}

public void ChangeDropDownListServer(string ea)
{
    char[] parser_char = { ',' };
    string[] messages = ea.Split(parser_char);
    string dll1 = messages[1];
    Session["TargetLayer"] = dll1;
}

ICallbackEventHandler 成员

```


原因就在这里，改变 Session ["TargetLayer"]的值，SelectFeatures 需要知道是对哪个图层进行查询的，从而对在那个图层选择要素进行高亮及属性显示，这里 callback 仅仅是在做了变量值的处理。最后在页面之间加入 js 脚本 ChangeLayer()。


```
<script type="text/javascript" language="javascript">
var context;


function ChangeLayer()
{
    var message;
    var ddl1value = document.getElementById('LayerDropDownList1').value;
    message = 'ddl1';
    message += ',' + ddl1value;
    <%=sADFCallBackFunctionInvocation%>
}
</script>
```


运行程序：

http://localhost/serverapp07/Default.aspx

 Zoom In

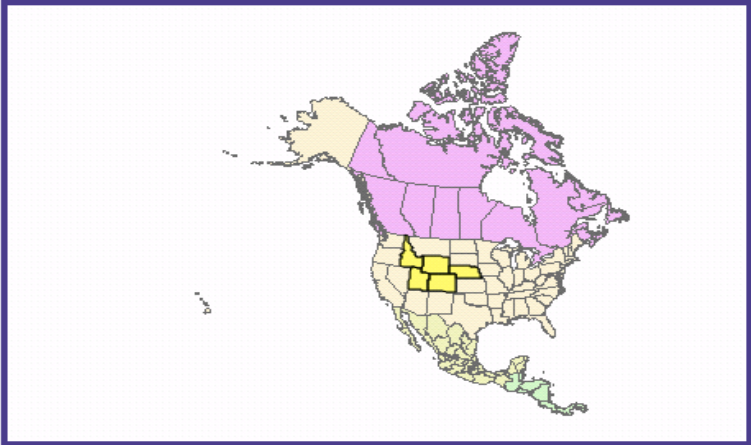
 Zoom Out

 Pan

 Full Extent

Select Features

Active Layer:
United States ▼



☒ Selection

☒ NorthAmerica

- ☒ Canada
- ☒ Mexico
- ☒ United States
- ☒ Central America

其中黄色区域就是Select Features按钮矩选的要素，下方gridview显示了查询到的属性结果。

程序中有两个地方用到了异步刷新，一个是ASP.Net 2.0 原有接口 ICallbackEventHandler，另一个是Web ADF framework的CallbackResult类，最初认为简单异步刷新用自己写的XMLHttpRequest请求更为简单，如上例中对 session存储值的改变，不用ICallbackEventHandler，但是在server地图互操作的过程中，ICallbackEventHandler给我们提供了更多的便利。

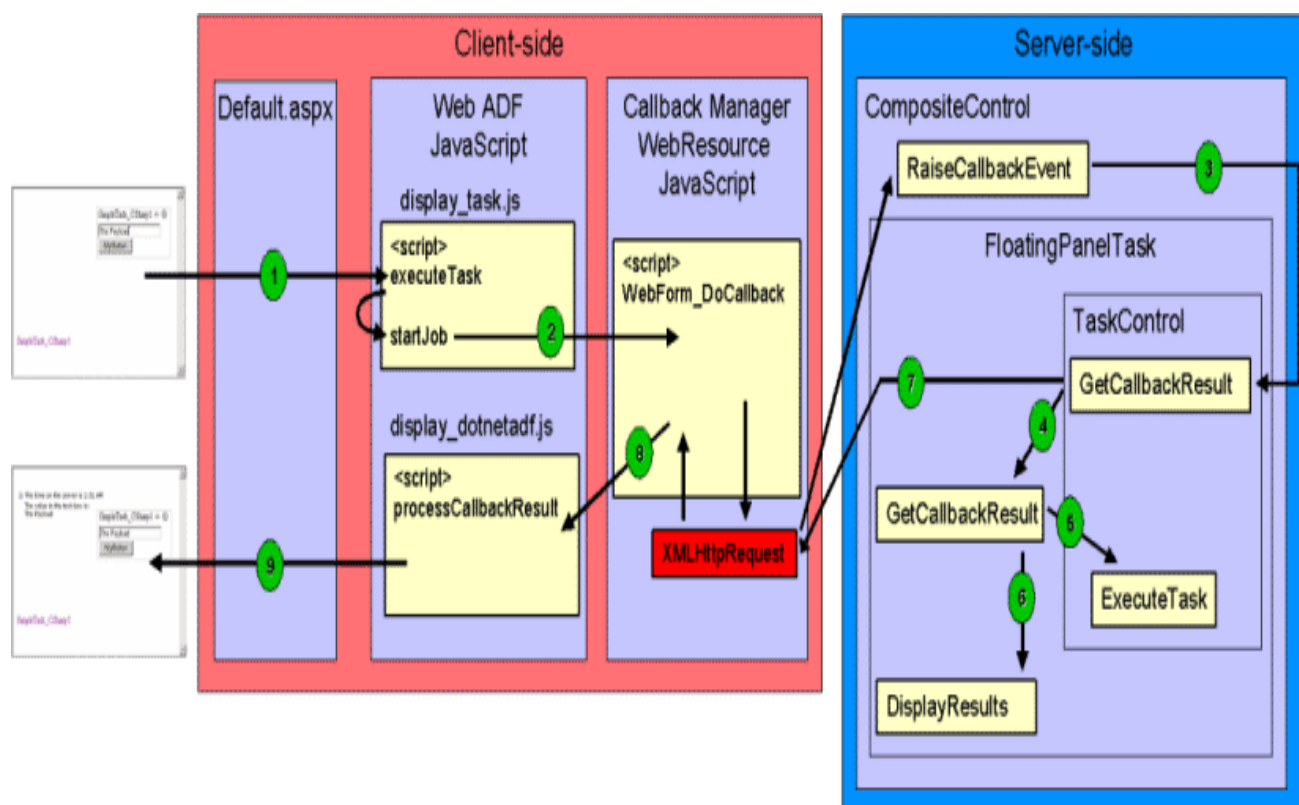
继续思考：

- 1.本例实现了根据地图查询属性，反过来根据属性查询几何图形怎么实现呢？其实前面《[ArcGIS Server 开发系列（三）--漫游 Graphics data sources](#)》已经讲到了，只不过需要将条件查询的信息，更改为在gridview或其他地方选择的属性信息，然后高亮显示相应的几何要素。
- 2.这种几何要素图形和属性信息的关联可以应用于各种不同的业务需求中，如图形和属性的同步删除、位置定位、类似结果查询等等。
- 3.如何改进或提升这种图形和属性的异步刷新带来的用户体验？

ArcGIS Server 开发系列（六）--自定义 Tasks

Tasks 由一组相互关联的组件、动作组成，并可以展现最后的结果，如 QueryAttributeTask 执行空间数据属性查询，结果返回到 Results 控件中，开发中我们既可以使用 .Net Web ADF 已有的 Tasks 控件，也可以自定义 Tasks，构建一些通用的 GIS 功能。使用已有的 Tasks 控件比较容易，所有的 Tasks 均有 TaskManager 管理，编辑 TaskManager 可以找到当前应用中所有的 Tasks，因此，一个基本页面中只需包含 TaskManager、TaskResults、及一个或多个 Task 控件就可以实现 Task 功能。而实际开发中，我们更多需要的是自定义各种 Task，包括功能、外观，现在我们就一起来设计一个 custom task。

动工之前，先来一图——Task Runtime Workflow，描述了整个 Task 的工作机制。



目标：

自定义一个功能，对指定的图层进行属性查询，返回查询的地图信息，将该功能封装成一个 Task，添加到 TaskManager 中。

准备工作：

1. 熟悉 Web ADF Task 已有控件的使用
2. 了解 Task Runtime Workflow

3.了解 ESRI.ArcGIS.ADF.Web.UI.WebControls.ITask 接口

4.了解 ASP.Net 2.0 Callback framework

5.更改 MapResourceManager 属性，添加一个 Resource: MapResourceItem0，类型是 ArcGIS Server Internet

开发分为两部分，一是功能实现，二是封装 **Task**，发布整个 **Web** 应用。

首先，创建一个 ClassLibrary 模板的工程 DefQuery，生成的 DefQuery 类实现抽象类 FloatingPanelTask，后者继承了 FloatingPanel，实现了 ITask 接口，因此 DefQuery 必须实现 FloatingPanelTask 中的抽象方法。编码过程需要根据 Task 的运行流程进行。

1.事件触发 Task

事件触发有多种方式，例如点击 button、textbox 回车等等，事件触发后需要调用 display_task.js 里的 executeTask 函数，通常我们这样写：

```
executeTask('args=' + document.getElementById('TaskManager1_DefQuery1_sellayer').value + ':' + document.getElementById('TaskManager1_DefQuery1_textBox').value, "WebForm_DoCallback('TaskManager1$DefQuery1',argument,processCallbackResult,context,postBackError,true)");
```

里面包含两部分，一是传入的参数，一是 WebForm_DoCallback，后者可以去看 ASP.Net2.0 异步调用相关资料。executeTask 函数在 display_task.js 中，它完成两件事情，分配一个 job id 来跟踪异步请求，和初始化一个回调并触发 TaskResults 控件中的指示器（指示 Task 程序正在运行中）。

2.调用 startJob 函数

startJob 提供了一系列的参数，通过键值对方式传送用户请求：

```
EventArg=startTaskActivityIndicator&taskJobID=1&args=0: TYPE='St'
```

然后通过参数_callbackArg 传入 RaiseCallbackEvent 方法。

3.服务器端处理

客户端的请求传入服务器端在 GetCallbackResult 方法中进行处理，这里分为两个步骤，一是 startTaskActivityIndicator，它会在 TaskResults 控件中指示 Task 任务正在执行，二是 executeTask，即处理真正的核心业务：

```
public override string GetCallbackResult()  
{  
    NameValueCollection keyValColl = CallbackUtility.ParseStringIntoNameValueCollection(_call
```



```

backArg);

    if (keyValColl["EventArg"] == "executeTask")
    {
        string sInArgs = keyValColl["args"];

        string delim = ":";

        char[] delchar = delim.ToCharArray();

        string[] args = sInArgs.Split(delchar);

        object[] inputs = new object[2];

        inputs[0] = args[0];

        inputs[1] = args[1];

        Input = inputs;
    }

    else if (keyValColl["EventArg"] == "startTaskActivityIndicator")
    {
    }

    else if (keyValColl["EventArg"] == "hidden")
    {
    }

    return base.GetCallbackResult();
}

```

参数为"executeTask"时，会执行 ExecuteTask 重写方法，里面可以处理业务逻辑，将结果保存为一种 Results，可以是 SimpleTaskResult、DataSet 或 TaskResultNode。

4.DisplayResults 显示结果

Task 处理完之后，最终会交给 DisplayResults 将结果显示出来，看看 DisplayResults 代码：

```

protected virtual void DisplayResults(string jobID, object taskInput, object taskResults)
{
    ITaskResultsContainer resultsContainer = null;

    for (int i = 0; i < TaskResultsContainers.Count; i++)
    {

```

```

|         resultsContainer = Utility.FindControl(TaskResultsContainers[i].Name, Page) as ITaskRes
ultsContainer;

|         if (resultsContainer != null)
|         {
|             resultsContainer.DisplayResults(this, jobId, taskInput, taskResults);
|             CallbackResults.CopyFrom(resultsContainer.CallbackResults);
|         }
|     }
| }
| }
| }

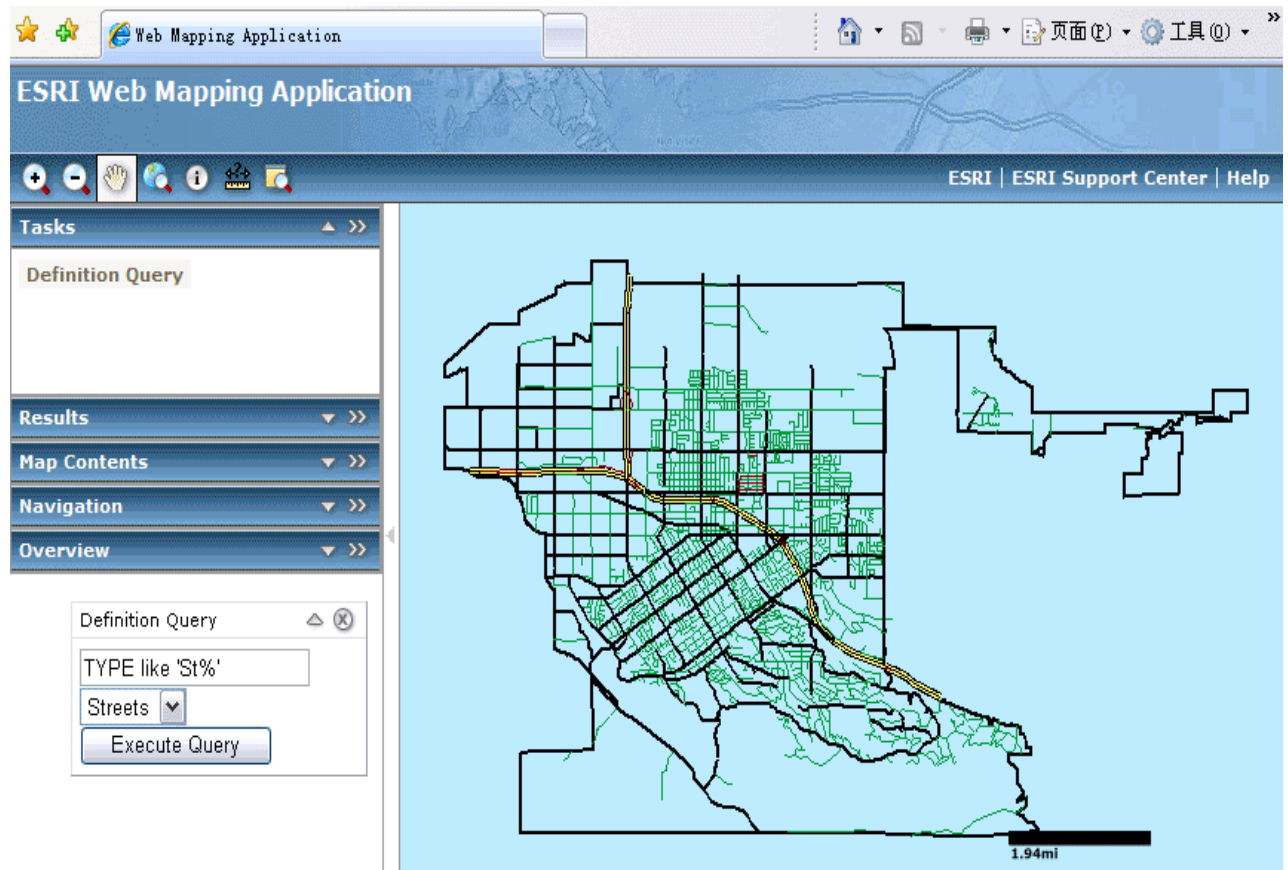
```

5.processCallbackResult

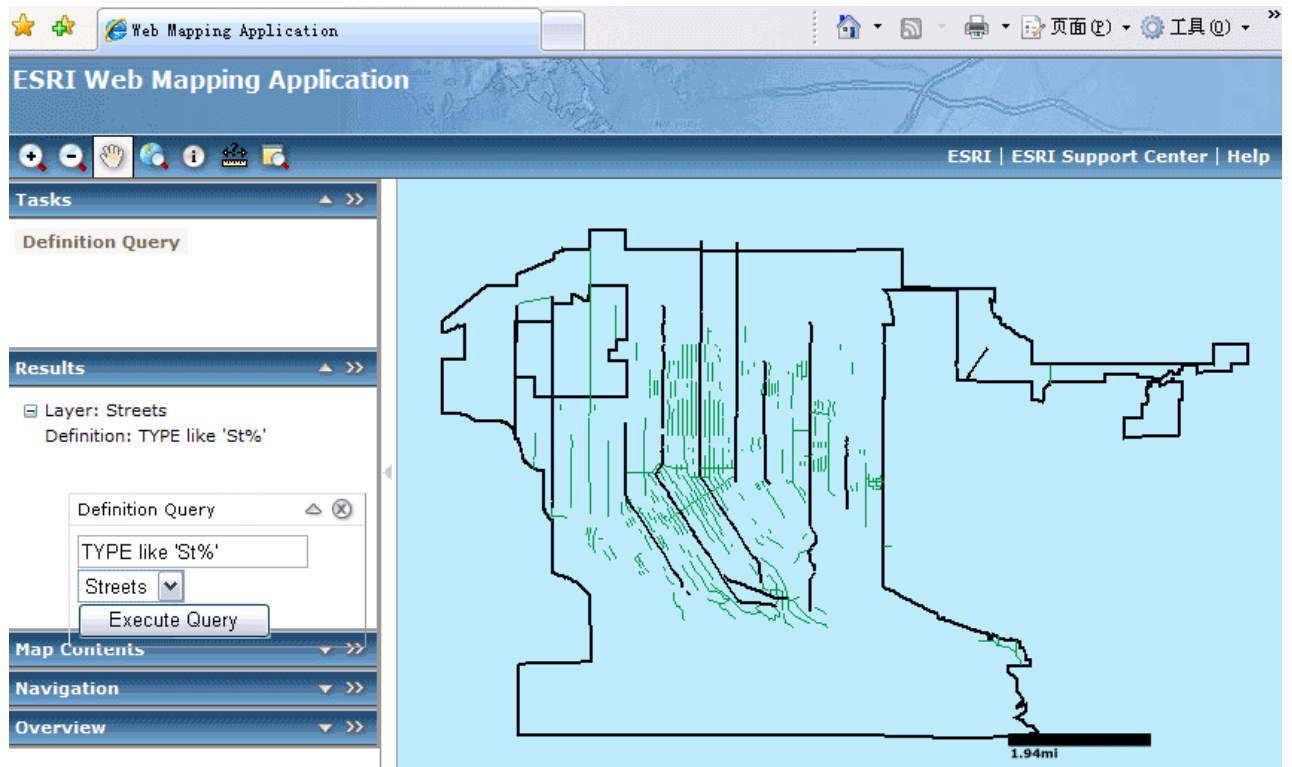
最后所有回调的结果均返回到回调响应函数——processCallbackResult，它专门处理客户端 Web ADF 控件的回调响应，并触发一系列后续动作，更新客户端显示，这样整个 Task 就执行完成了。

Build 整个工程，生成 DefQuery 的 DLL 文件，用模板新建一个 ArcGIS Server 应用，展开工具栏右键空白处，选择之前生成的 dll，此时就可以像 QueryAttributeTask 等原有的 Task 一样来使用 DefQuery 了，添加到 TaskManager 中，发布整个 Web 应用，这样自定义的 Task 就完成了，可以看看效果，下图做了一个简单的查询 Task，根据用户写的属性查询条件，在地图上返回所查询的地图。

Task 查询之前



Task 查询之后



进一步完善应用：

- 1.以上自定义 Task 的 UI 基于 FloatingPanel，我们可以更改它的现实样式，比如透明度、颜色、布局等等。
- 2.可以将显示出来的结果存为一个 Graphic 图层，保留原始图层。
- 3.根据属性查空间信息，考虑查询到结果时，同时将 Map 定位到结果所在的窗口范围。