ArcGIS API for Windows Phone 开发实例

diligentpig

2011/5/7

ArcGIS API for Windows Phone 开发实例(0):为什么选择 Windows Phone？

为什么要了解 Mobile GIS？

GIS 技术固然有自己独特的理论知识，但 GIS 的实际应用离不开 IT 技术，GIS 应用的发展离不开主流 IT 技术的发展。上面这句话我们可以看做一条公理，公理是不需要证明的。GIS 技术的应用，从 C/S 到 B/S，从 SOAP 到 REST，从 SOA 到云，都印证了这条公理。

从台式机到笔记本，从上网本到平板电脑和遍地开花的智能手机，移动不仅是 IT 技术的发展趋势之一，也是所有硬件的发展趋势。

综上所述，作为 GISser 的你，没有理由不了解 Mobile GIS 技术。

ArcGIS 移动产品线

为什么要学习 ArcGIS API for Windows Phone？从两个角度来回答：ArcGIS 和 Windows Phone。ArcGIS 目前有 5 种移动产品，所有产品的介绍，帮助，下载详见这里。

ArcPad。这是一个开箱即用的软件产品,基于 Windows Mobile 平台，最新版本是 ArcPad 10（支持 Windows Mobile 6.5）。特点是数据采集精度高（可到分米级），支持并推荐采用完全离线作业方式，包含丰富的 GIS 功能，比如图形（有捕捉功能）/属性/符号编辑，图层管理等，界面类似 ArcMap，适合 GIS 专业人员使用，可通过界面或脚本语言进行定制。由于推出早（历经 5.X，6.X，7.0,7.1,8.0.10

几个版本），功能实用而丰富，收到广大用户青睐，目前全球用户超过 100,000。

ArcGIS Mobile。是目前 ArcGIS 应用最广的移动产品之一，基于 Windows Mobile 平台，最新版本是 ArcGIS Mobile 10.0（支持 Windows Mobile 6.5）。特点是拥有自己的离线缓存格式，可完全离线使用，也可与 ArcGIS Server 随时进行各种粒度的缓存同步；基于任务模式，由工作流驱动，非 GIS 人员可以很快上手；可进行离线数据编辑，属性/空间查询，外业人员协作等任务；具有丰富的定制功能，并提供功能全面的 SDK 可进行二次开发。软件历经 9.2,9.3,9.3.1,10.0 几个版本，在国内拥有广大的用户群体。

ArcGIS for iOS。包括一个开箱即用的应用程序（ArcGIS for iOS Application，可在 App Store 中免费下载）和提供二次开发功能的 ArcGIS API for iOS。基于苹果公司的 iOS 系统，可在 iPhone，iPad，iPod Touch 产品上运行。最初于 2010 年 4 月发布，目前应用程序和 API 版本是 1.8。

ArcGIS for Windows Phone。包括一个开箱即用的应用程序（ArcGIS for Windows Phone Application，可在 MarketPlace 中免费下载）和提供二次开发功能的 ArcGIS API for Windows Phone。基于微软公司的 Windows Phone 系统，可在基于该系统的手机上运行。最初于 2010 年 9 月发布，目前应用程序和 API 版本是 2.2 beta。

ArcGIS for Android。基于 Google 公司的 Android 系统。目前 ArcGIS API for Android 处于 public beta 阶段，可在 ArcGIS Beta Community 中免费申请试用。正式版推出后，应该也会有开箱即用的应用程序，照惯例可通过 Android Market 免费下载。最初与 2010 年 10 月内测，今年 2 月底开始公测。

以上三个产品（ArcGIS for iOS，ArcGIS for Windows Phone，ArcGIS for Android）是 ArcGIS 新一代的移动产品，大部分功能都是基于 ArcGIS Server 所发布的地图服务来使用（需要网络环境支持），包括地图操作，GraphicsLayer/FeatureLayer 支持，各种 Task（Identity/Query/Find/GeoProcessing 等）的使用等。可以看出，这三种移动产品与 ArcGIS 客户端 API（ArcGIS API for Javascript/Flex/Silverlight）所提供的功能基本一致，因此它们的概念和开发方式与三种客户端 API 无异。大家可能比较关心这三种 API 的离线使用方式，目前来说，理论上可以实现离线使用，但需要自己进行开发定制，可参考 iOS 中自定义图层的例子。以后的版本中肯定会加入离线模式，但目前无法给出具体的时间表。

为什么选择 Windows Phone？

现在最流行的手机操作系统有三种 iOS，Android 和 Windows Phone。关于这三种操作系统究竟谁好谁坏，仁者见仁，每个人都有自己的答案。如果你是苹果的忠实粉丝，那么你可能对 Android 或 Windows Phone 系统不屑一顾，没关系，Android 和 Windows Phone 用户也是这么想的。简单介绍一下。

iOS。苹果公司的操作系统，用于其所有移动设备之上（iPhone/iPad/iPod Touch）。操作体验极好，界面华丽（容易吸引 mm 和领导），应用程序丰富；软硬件环境统一，用户群体相对固定，忠实度高；开发使用 Objective C 语言，难度相对较大。

Android。Google 公司的手机操作系统。07 年底推出，占有率迅速上升，目前是市场占有率最高的智能手机系统。系统本身基于 Linux，开源（软件版本多，

定制版本多）；集成 google 各种产品，包括 gmail，gtalk，latitude 等；应用软件丰富，用户群体广泛，以 google 的忠实用户为代表；开发基于 Java 语言。

Windows Phone。微软公司 2010 年 10 月推出的新一代移动操作系统，用以取代即将被淘汰的 Windows Mobile。不同于封闭的 iOS 和稍显混乱的 Android，它的硬件环境统一，操作体验好；MarketPlace 在不到半年的时间里，应用程序已达 10,000 个（我写文章这会是 9643 个）；目前的生产厂商有 HTC，Dell，三星，LG，还有即将加入的 Nokia。

你可能有很多理由不选择微软，但不管你喜欢不喜欢，它就在那里，不离不弃。这里给出选择 Windows Phone 的几个理由，供参考。

1、微软的云+端战略。Windows Azure 是微软自己的云平台，至于它的优劣，可以自己搜索。但有一点是肯定的，这是微软不惜重金打造的战略平台，在未来数年内会主导微软其他产品的发展方向，而微软也会不遗余力的推广它。端是指客户端，包括电脑，电视和移动端的手机，也就是 Windows Phone 了，所以其推广和宣传力度可知。

2、Windows Phone 上有两种开发架构，Silverlight 和 XNA。Silverlight 和 WPF 是微软下一代的开发技术，而 XNA 是微软 XBOX 平台上的游戏开发技术。也就是说，不论是 Silverlight 的程序还是 XNA 的程序，基本上拿来就可以在 Windows Phone 上运行，效果不打折扣，这得益于微软统一的平台策略。给我们带来的直接好处就是，对于熟悉 Silverlight 或 WPF 的同学来说，Windows Phone 的开发门槛几乎为零，而这两者所能够实现的功能，大家也是有目共睹，毋庸置疑。

3、Windows Phone 的娱乐性。XNA 开发出的游戏得到了全球资深游戏玩家

的一直认可，因为主流游戏平台就是 PC+XBOX+PlayStation。如果你看过去年微软 TechEd 上 Windows Phone 的游戏演示（点这里），那你一定不会再留恋 iOS 上的极品飞车了。

讲座内容

本系列文章内容以去年微软 TechEd 上的演示 demo 为例，从零开始，教你一步步完成这个 Windows Phone 应用程序实例。如果你学习过 ArcGIS API for Silverlight，那么很好，你会在这里学到有关 Windows Phone 的开发知识；如果没学过，也没关系，本系列也会再次讲解 API 中所有的相关概念和内容。

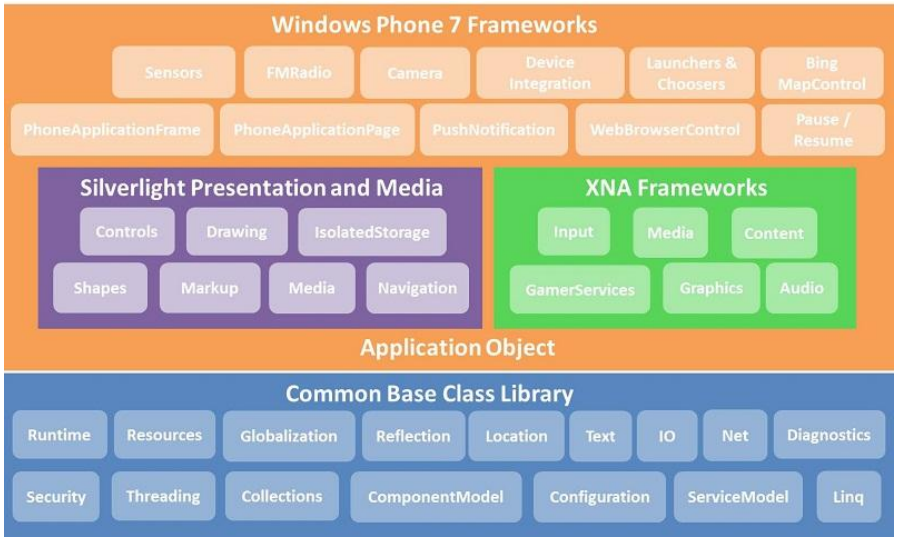
每篇文章内只提供相关代码和说明，希望在学习的过程中大家多思考为什么，而不是简单追求拷贝粘贴看效果。好了，该说的也都说了，如果你选择继续信任并使用 Windows Phone，请继续关注；如果你对 ArcGIS API for iOS 感兴趣，请看 barry.z 的系列教程，如果你对 ArcGIS API for Android 感兴趣，请看牛魔王的系列教程。

ArcGIS API for Windows Phone 开发实例(1):准备工作

本文有三部分内容。注：以后的文章中以 WP 作为 Windows Phone 的简写。

一、 Windows Phone 开发需要哪些知识？

先来看一下 WP 的整体结构：



最底下是通用的基础类库，可以当做是.NET Framework 中保留了核心功能的简化版 CLR；最上层是 WP 特有的类库，提供给开发者与手机相关的功能，比如说屏幕控制，摄像头控制，获取位置等等。

中间这层是 WP 为我们提供的两套开发框架：Silverlight 或 XNA。也就是说，你可以利用 Silverlight 技术或者 XNA 技术来开发运行在 WP 上的应用程序。大致地，在 WP 上，Silverlight 适合编写大多数应用程序，XNA 适合编写游戏程序。XNA 是微软提供的一套游戏开发环境，它有多强大，看看 XBOX 有多成功就知道了，感兴趣的同学可以深入研究。Silverlight 是微软抢占 RIA 市场的有力武器之一，也是以后微软.NET Framework 中主打开发技术之一——WPF——的子集。

作为.NET 开发人员，如果还不熟悉的话，有必要赶紧熟悉一下。关于它的介绍很多，可以参考我之前写的这篇，或者自己搜索。

要强调的一点是，你在别处写的 Silverlight 或 XNA 程序基本不需要或只需要进行少量的更改（比如屏幕尺寸调整或使用 WP 的硬件功能），便可顺利运行在 WP 上，这是微软统一的开发平台给我们带来的好处。

这次，我们只关心 Silverlight，它就是开发 WP 程序所需的知识。

二、ArcGIS API for Silverlight/WPF 和 ArcGIS API for Windows Phone 的异同？

在比较 Esri 的这几个客户端 API 之前，我们需要看一看 Windows Phone 上使用的 Silverlight 技术和普通的 Silverlight 技术有什么区别。目前是 WP 的第一个版本，也就是 Windows Phone 7，它所使用的 Silverlight 开发技术是基于 3 版本的，在 WP 开发中你基本可以使用 Silverlight 3 的所有特性，还包括 WP 特有的触控功能。具体可参考这两篇文章：Features Supported in Silverlight for Windows Phone 和 Differences Between Silverlight and Silverlight for Windows Phone。需要强调的一点是，Silverlight 4 的新特性在 WP 上基本无法使用，对此我们只能表示遗憾。但可以期待在以后的 WP 版本中，Silverlight 版本能够与其他平台统一起来。

那么 Esri 的这三个 API（最新的 2.2beta 版本中已将 Silverlight 和 WPF 两个 API 分开）就比较好理解了。ArcGIS API for Silverlight，ArcGIS API for WPF，ArcGIS API for Windows Phone 三者的功能，架构，包括源代码完全一致，只是编译的目标平台不同。对于 Esri 的这三个 API，你只需熟悉其中之一，就可以在其他两个

平台上进行开发。

所以，如果你没有学过 ArcGIS API for Silverlight 的话也没有关系，会了 Windows Phone API，你也就学会了 Silverlight API。

三、我们要完成的工作

在上一篇文章中，我们已经了解到，最终我们需要完成这么一个 WP 上的应用程序：连锁超市的老板出差在外，在机场候机的过程中想查看一下自己的几家店面最近的营业情况。传统的统计报表只能通过电子邮件发送给老板，他需要不停的查看自己的邮箱；而里面的内容既不直观也不易懂。将报表数据发布成地图服务，这样即可在任何时间任何地点访问业务数据，也可通过地图的方式充分挖掘数据的潜在价值。老板通过 windows phone7 上的地图应用，对自己店面近来营业情况一目了然，利用应用所提供的分析功能，即使在千里之外也能随时对自己的生意轻松经营。

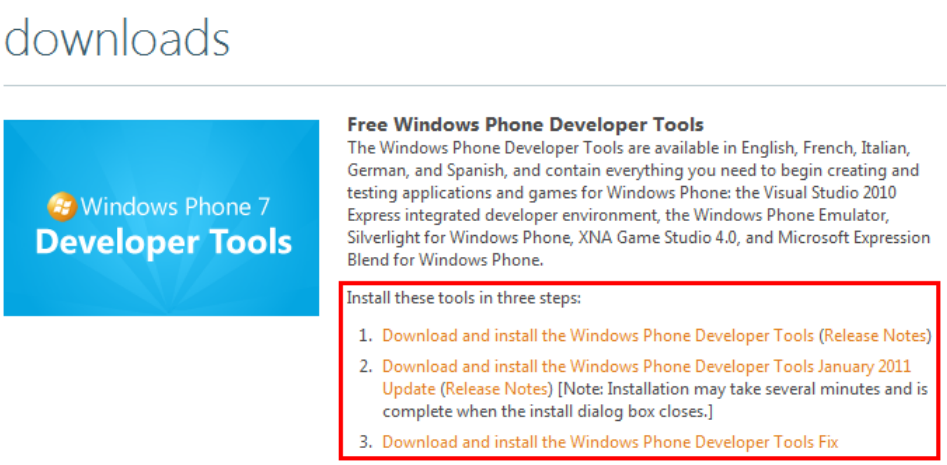
好了，要了解的预备知识已经梳理完毕。有人可能说，都看了这么多了，还没进入动手阶段？我是希望大家每时每刻都知道我们在做什么，为什么要这么做，知其然也知其所以然。就跟听别人的 PPT 一样，对于我们不熟悉的内容，如果前面两页他交代不清楚，可能整个 PPT 就是在浪费我们的时间。凡事皆如此。

ArcGIS API for Windows Phone 开发实例(2):HelloMap

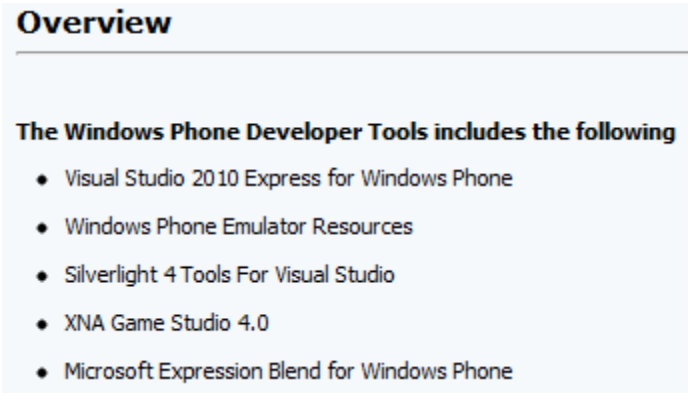
本文内容：WP 开发环境的搭建和第一个程序 HelloMap。所有参考资料都会给出相应链接，供仔细的朋友进一步学习，后同。

一、开发环境搭建

1、安装微软的 Windows Phone Developer Tools。微软的开发离不开 Visual Studio,Silverlight 开发又需要 Silverlight Tools for Visual Studio,WP 的开发需要 Windows Phone Developer Tools...看起来有点复杂，**其实很简单，只需到 APP HUB 网站，按照 1、2、3 的说明依次安装即可。**如图：

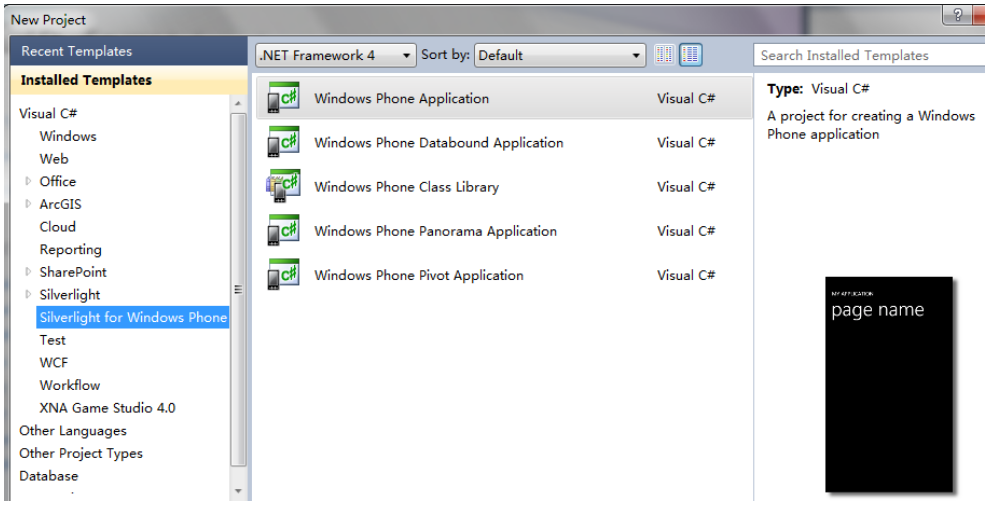


其中最主要的是 1 中提到的 Windows Phone Developer Tools，它会下载一个 3.2M 大小的 vm_web.exe 安装程序，这其实是一个安装工具，不管你的机器是否安装了 VS 2010（不支持以前的版本），是否安装了 Silverlight Tools，它都会检测你机器开发 WP 程序所需的所有工具，并提示你进行安装（比如没有 VS 的话会为你安装 VS 2010 Express，已有的就不再重复提示了），有点类似 1 click install，非常方便。如果你的机器上什么都没有，那么它会依次提示你安装以下几个工具：



另外 APP HUB 网页上第二步中提到的 Windows Phone Developer Tools January 2011 Update 修复了最初版本中的 bug，提供了一个新的模拟器，以及一些新的特性，比如拷贝和粘贴。第三步中提到的 Windows Phone Developer Tools Fix 修正了不能给真机上部署超过 64M 大小的 XAP 文件的 bug。所以，2、3 属于选装，但强烈推荐安装。

安装完成之后，打开 VS 2010，新建工程中应该能够看到以下画面：

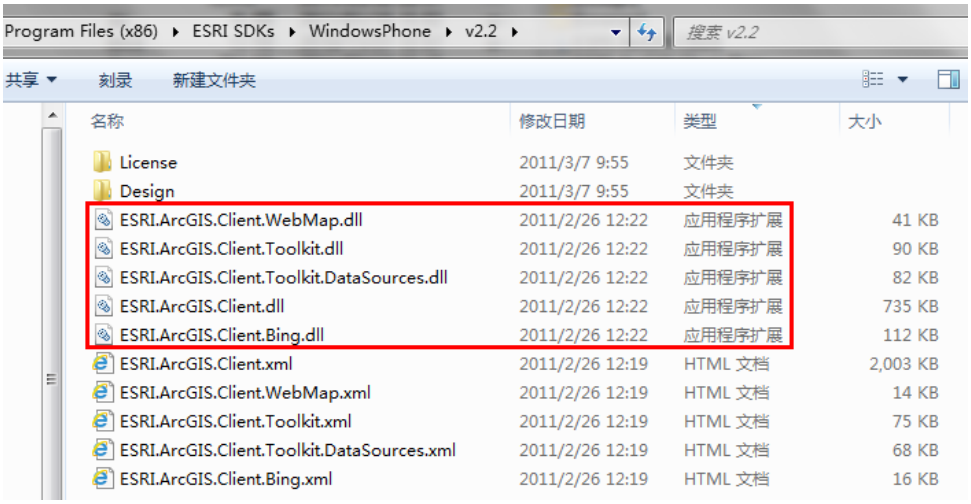


至此，我们已经可以用 Silverlight 开发 WP 的应用程序了。

另外，与 Silverlight Toolkit 类似，微软还提供了 Silverlight for Windows Phone Toolkit，地址都在这里。它们都是封装好的，开源的一些控件或效果，用来补充自带控件的不足，为你的程序增加效果。

2、安装 Esri 的 ArcGIS API for Windows Phone。ArcGIS API for Windows Phone

与 ArcGIS for Silverlight/WPF 一样，是 Esri 免费提供给我们的开发包，里面包含了几个 dll 文件（包含各种类，函数）供我们引用。目前最新版本是 2.2 beta，可利用自己的 Esri Global Account（可免费申请）登陆 Beta Community 后进行下载。如果你不想参与 beta community，也可以在这里按照提示下载 2.1 版本的 API。2.2 beta 版本的 API 提供了 exe 安装程序（2.1 版本没有），安装好后，你可以在安装目录中找到以下内容：



- ESRI.ArcGIS.Client.dll。这是 API 中的核心类库，包括最常用的 Map 控件，各种图层类，Graphics，Geometry，Symbols 等，同时还包括了各种 Task：Identity，Find，Query，Geoprocessing 等；
- ESRI.ArcGIS.Client.Bing.dll。如果你需要在程序中调用 Bing Maps 的底图或使用它的 Geocoding 或 Routing 服务，则需要这个 dll；
- ESRI.ArcGIS.Client.Toolkit.dll。类似 Silverlight Toolkit，是 Esri 提供的已经封装好的，开源的，一些常用控件，比如缩略图，书签，导航条，InfoWindow 等，可以在这里下载它们的源码；
- ESRI.ArcGIS.Client.Toolkit.DataSources.dll。提供对一些通用数据源的加载支持，

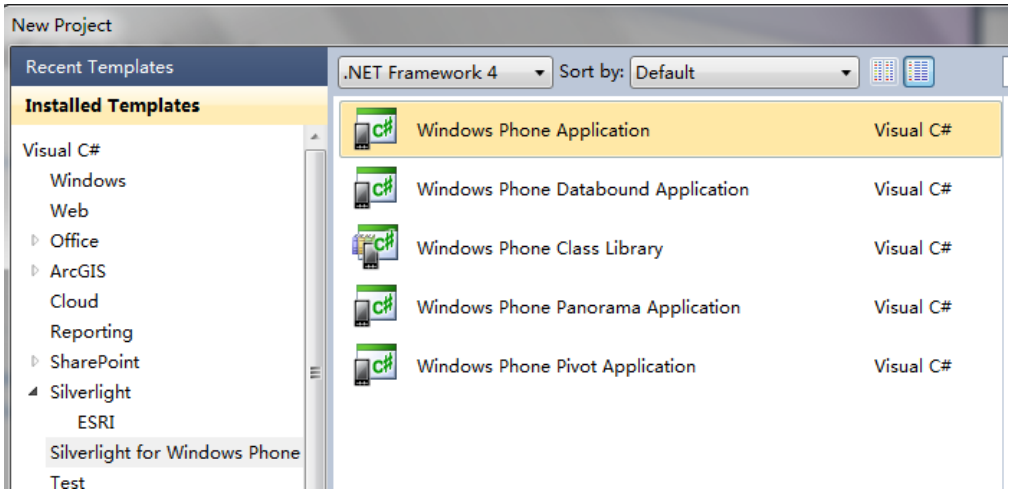
比如 OpenStreetMap，WMS 服务，KML 等，可以在这里下载它们的源码；

- ESRI.ArcGIS.Client.WebMap.dll。可用于加载 ArcGIS Online 的在线地图。

好了，下面我们就可以开始我们的第一个应用程序 HelloMap 了。

二、 HelloMap

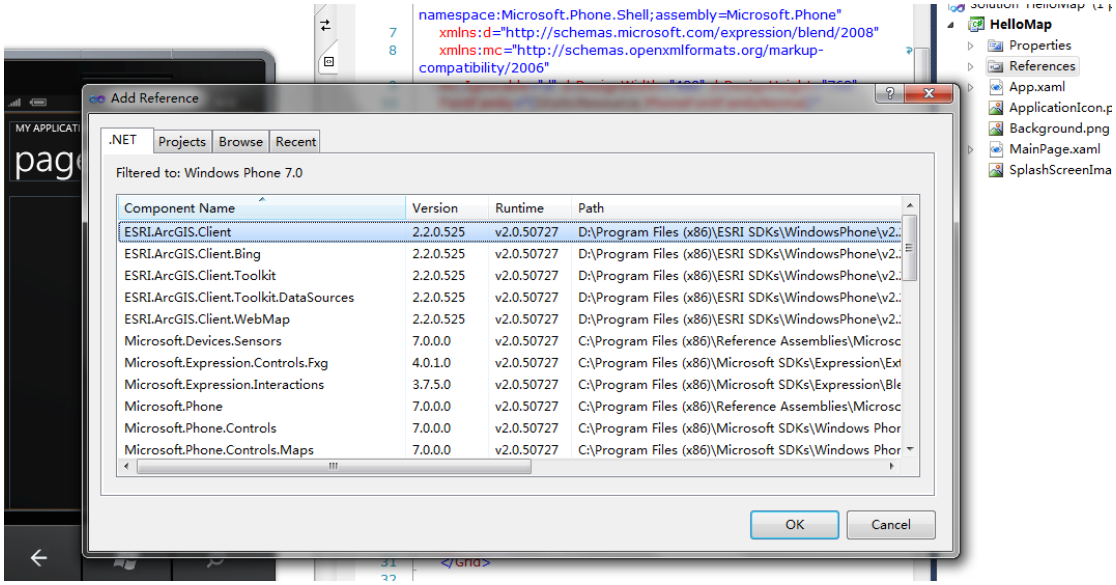
打开 VS 2010，New Project，左侧选择 Silverlight for Windows Phone，右侧 Windows Phone Application,然后 OK：



可以看到新建好的工程中包含几个文件，我们对其中几个 xaml 类型文件做一下解释。Silverlight 开发中，文件主要分为两种类型，以.xaml 结尾的文件和以.xmal.cs（或.xaml.vb）结尾的文件，前者是一种扩展的 xml 语言，负责页面的布局、样式，后者是普通的代码文件，主要负责程序的逻辑功能。界面与功能很清楚地分离开来。App.xaml（App.xaml.cs）是程序的入口，与应用程序相关的全局事件，比如进入或退出应用程序都在这里；MainPage.xaml(MainPage.xaml.cs)是自动创建的主页面，程序默认会启动这个页面。注：你也可以在 WMAppManifest.xml 文件中来显示指定程序的第一个启动页面。

因为我们要使用 API 中的 Map 控件，所以先添加对 ESRI.ArcGIS.Client.dll 的

引用（如果是 2.1 版本，需要手动 Browse 到这个 dll）：



然后在 MainPage.xaml 中为这个 dll 引入一个 namespace(类似代码文件中的 using)：

```
<phone:PhoneApplicationPage
    x:Class="HelloMap.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:esri="clr-namespace:ESRI.ArcGIS.Client;assembly=ESRI.ArcGIS.Client"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
```

接下来添加我们用以显示地图的 Map 控件。在 MainPage.xaml 中找到名为 ContentPanel 的 Grid 部分，在其中添加我们的 Map 控件，并在加入一个底图图层：

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <esri:Map x:Name="MyMap">
```

```
                <esri:ArcGISTiledMapServiceLayer ID="MyLayer"

Url="http://services.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapSe
rver" />
            </esri:Map>
        </Grid>
```

按 F5，编译运行（注意将 Targets 选择为 Windows Phone Emulator 而不是 Windows Phone Device），会启动 Windows Phone 模拟器，就完成了我们的 HelloMap 程序。

解释一下。Grid 是 Silverlight 里最常用的布局控件之一（另外还有 StackPanel 和 Canvas），类似于 Html 里的 Table，可自行搜索学习或参考这里；x:Name 相当于该元素的一个 id，便于在.cs 的代码页面中直接引用。

Map 控件是 ArcGIS API for Windows Phone 中最基本的控件，所有能够看见的

与地图有关的元素都将呈现在这个控件以内。但它本身是空的，所以我们需要给其中添加一个图层，用以显示上面的地图。我们这里使用的图层类型是 ArcGISTiledMapServiceLayer，它专门并且只能用于加载 ArcGIS Server 发布的经过切片的缓存地图服务，与之相对应的还有 ArcGISDynamicMapServiceLayer，专门并且只能用于加载 ArcGIS Server 发布的动态地图服务，以及 ArcGISImageServiceLayer，用于加载 ArcGIS Server 发布的影像服务。这里的 Url 是一个地图服务的 REST 方式的节点，可通过 ArcGIS Server 的 Service Directory 查看。比如我们可以查看通过 http://services.arcgisonline.com/arcgis/rest/services 这个地址查看 ArcGIS Online 上发布的所有服务。在这里你并不需要掌握有关 ArcGIS Server 的任何知识，只需要知道所有的资源，包括地图，查询等功能都是服务器端 ArcGIS Server 通过服务形式暴露给你的，而我们只需通过一个 Url 去引用它即可。当然如果你对 ArcGIS Server 感兴趣的话，可以在这里找到资料。

好了，这就是我们 HelloMap 的所有内容。最后给出几个相关链接，也是我们学习过程中最常用到的资料，如能掌握，则可事半功倍。

关于 Map 控件和图层的详细解释：

http://help.arcgis.com/en/arcgismobile/10.0/apis/WindowsPhone/help/011v/011v00000005000000.htm

ArcGIS API for Windows Phone 的在线例子（可操作，有源码）：

http://help.arcgis.com/en/arcgismobile/10.0/apis/WindowsPhone/samples/start.htm

关于 WP7 的知识，推荐 Windows Phone 7 in 7 Minutes!系列视频教程，全面、快速、易懂

关于 ArcGIS Server 的知识：

http://help.arcgis.com/en/arcgisserver/10.0/help/arcgis_server_dotnet_help/index.html

ArcGIS API for Windows Phone 开发实例(3):在地图上显示超市数据

本文内容：GraphicsLayer，FeatureLayer，自定义地图符号。

我们的应用场景是对超市数据进行展示，查询和分析，因此第一步当然是要将超市显示在地图之上。如所有的 GIS 软件一样，要将数据加载到地图里面，需要图层的概念，这里我们也需要合适的图层，将超市数据加载进来。

先看一下 ArcGIS API for Windows Phone/Silverlight/WPF 中的主要图层类型。

```
Layer
  |--TiledMapServiceLayer
  |        |--ArcGISTiledMapServiceLayer
  |--DynamicLayer
  |        |--DynamicMapServiceLayer
  |                  |--ArcGISDynamicMapServiceLayer
  |                  |--ArcGISImageServiceLayer
  |                  |--GPResultImageLayer
  |--GraphicsLayer
  |        |--FeatureLayer
  |--ElementLayer
```

Layer 最基本的基类，常用的几种图层类型我已经用红色标记出来了。前两种 ArcGISTiledMapServiceLayer 和 ArcGISDynamicMapServiceLayer 分别用于加载 ArcGIS Server 发布的缓存地图服务和动态地图服务，这个在上一节中已经说过。再来看一下中间的 GraphicsLayer 和 FeatureLayer，可以说这是 API 中最常用的两种图层，大部分的程序功能都要基于它们来完成。

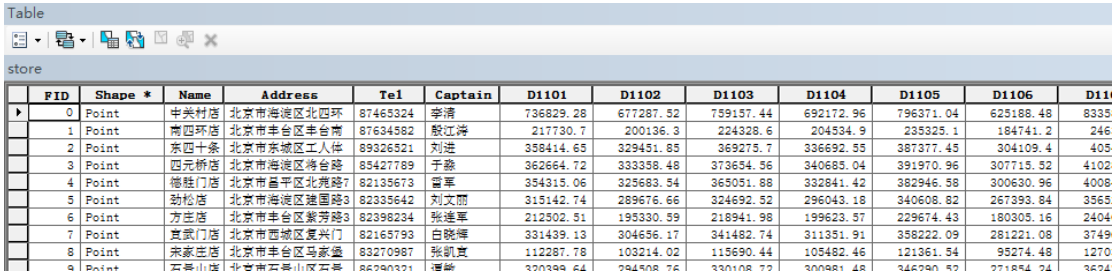
GraphicsLayer 是保留在内存中的一种图层(与 ArcMap、ArcGIS Engine、ArcGIS Server ADF 程序中的相应概念类似)，顾名思义是很多 Graphic 的集合，而所有与用户交互的内容通常都用 Graphic 来显示。比如多边形查询中用户画出的多边形，

属性/空间查询结果中的所有要素等内容，都是 Graphic。可以说，除了地图本身，基本上看到的所有与地理位置有关的东西都可以用 Graphic 来表示。Graphic 对象有 3 个重要的属性：Geometry，Symbol 和 Attributes。Geometry 代表了一个 Graphic 的几何形状（可疑是点、线、面任意一种）或地理位置，而 Symbol 则表示 Graphic 的呈现样子，比如颜色、效果，同时有了这两个属性（缺一不可），Graphic 就可以显示到地图上了。而 Attributes 是键值对集合，可在里面存储任意类型的对象，比如一个要素的属性信息。

FeatureLayer 继承自 GraphicsLayer，它与后者的区别是，GraphicsLayer 中的 Graphic 都是人为创建出来的，而 FeatureLayer 中的 Graphic 都是从 ArcGIS Server 发布的服务中读取出来的，因此 FeatureLayer 比 GraphicsLayer 多了一个 URL 属性。这个 URL 通常指向一个 ArcGIS Server 发布的，MapService 或 FeatureService 的子图层（对应一个 FeatureClass）。FeatureLayer 有了这个 URL 后，就可以读取出该服务对应子图层里的所有要素内容，因此 FeatureLayer 里 Graphic 的 Geometry 属性会自动被 FeatureClass 的 Shape 字段填充，而 Graphic 的 Attributes 字段则会根据要求，被 FeatureClass 中的属性信息所填充。如果发布服务的服务器是 ArcGIS Server 10 版本，则 Graphic 的 Symbol 属性会自动被服务的 DrawingInfo 信息填充。另外，FeatureLayer 是客户端 API 中对 FeatureService 的唯一载体，这是它另一个非常重要的作用（也是主要作用），以后的讲座中会提到。

ArcGIS API for Windows Phone/Silverlight/WPF/Javascript/Flex/iOS/Android 都是基于 ArcGIS Server 所发布的 REST 服务，所以，在开始之前，我们需要将超市数据用 ArcGIS Server 发布成 MapService。这个过程不属于本文要讨论的内容，不

熟悉的同学可自己查阅相关资料。超市图层的属性表如下，前面是每个店面的基本信息，后面是该月每天的营业额信息：



Table

store

FID	Shape *	Name	Address	Tel	Captain	D1101	D1102	D1103	D1104	D1105	D1106	D11
0	Point	中关村店	北京市海淀区北四环	87465324	李清	736829.28	677287.52	759157.44	692172.96	796371.04	625188.48	8335
1	Point	南四环店	北京市丰台区丰台南	87634582	殷江涛	217730.7	200136.3	224328.6	204534.9	235325.1	184741.2	246
2	Point	东四十条	北京市东城区工人体	89326521	刘进	358414.65	329451.85	369275.7	336692.55	387377.45	304109.4	405
3	Point	四元桥店	北京市海淀区海台路	85427789	于淼	362664.72	333358.48	373654.56	340685.04	391970.96	307715.52	4102
4	Point	德胜门店	北京市昌平区北苑路7	82135673	雷军	354315.06	325683.54	365051.88	332841.42	382946.58	300630.96	4008
5	Point	劲松店	北京市海淀区建国路3	82335642	刘文丽	315142.74	289676.66	324692.52	296043.18	340608.82	267393.84	3565
6	Point	方庄店	北京市丰台区紫芳路3	82398234	张连军	212502.51	195330.59	218941.98	199623.57	229674.43	180305.16	2404
7	Point	宣武门店	北京市西城区复兴门	82165793	白晓辉	331439.13	304656.17	341482.74	311351.91	358222.09	281221.08	3749
8	Point	宋家庄店	北京市丰台区马家堡	83270987	张凯宜	112287.78	103214.02	115690.44	105482.46	121361.54	95274.48	1270
9	Point	石景山店	北京市石景山区五景	86290321	谭敏	320399.64	294508.76	330108.72	300981.48	346290.52	271854.24	3624

超市的 ShapeFile 在这，点我下载。在我本机上发布出来的地址是：http://localhost/ArcGIS/rest/services/supermarket/MapServer/0。

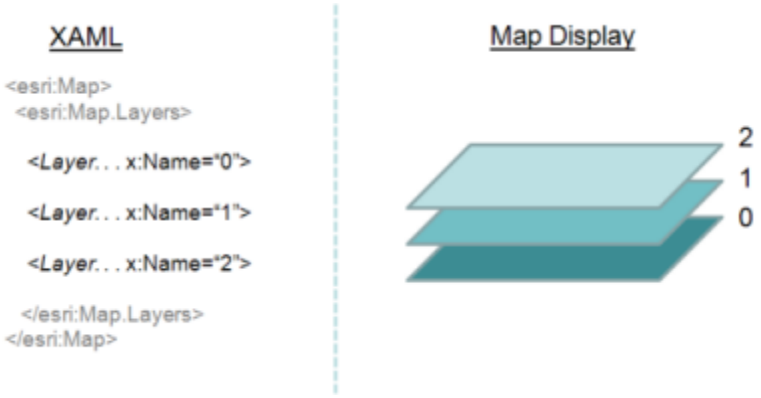
现在打开上一节我们完成的 HelloMap 程序，来继续改造之。底图服务我们依旧使用 ArcGIS Online 提供的 StreetMap 数据 http://services.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer，然后我们在 Map 控件中加入一个 FeatureLayer，其 URL 属性指向发布在本机的超市地图服务。由于我们的数据范围仅限于北京市附近，因此我们可以利用 Map 的 Extent 属性指定地图的初始范围，代码现在看起来是这样的：

```
<esri:Map x:Name="map1" Extent="12926244,4840437,12982108,4878585">
            <esri:Map.Layers>
                <esri:ArcGISTiledMapServiceLayer ID="BaseMap"
Url="http://services.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/M
apServer" />
                <esri:FeatureLayer ID="BusinessLayer"

Url="http://localhost/arcgis/rest/services/supermarket/MapServer/0"
                    OutFields="*" />
            </esri:Map.Layers>
        </esri:Map>
```

其中 FeatureLayer 的 OutFields 属性代表从服务器端返回的属性字段，这里“*”表示返回所有属性字段，也就是我们的 Graphic 对象的 Attributes 里自动填充了超市的所有属性值。

需要注意，在 xaml 代码中，最靠上的图层在显示时会出现在 Map 控件的最底端，如图：



Silverlight 中，可以将常用的任何资源保存到 Resource 集合中，以便重复利用。因此我们将两个服务地址保存到 App.xaml 文件的 ResourceDictionary 中（可在所有页面中直接引用）：

```
<Application
    x:Class="esridemo.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"

xmlns:esri="clr-namespace:ESRI.ArcGIS.Client;assembly=ESRI.ArcGIS.Client"
    xmlns:sys="clr-namespace:System;assembly=mscorlib"

xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone">

    <!--Application Resources-->
    <Application.Resources>
        <sys:String
x:Key="BaseMap">http://services.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer</sys:String>
        <sys:String
x:Key="BusinessLayer">http://localhost/arcgis/rest/services/supermarket/MapServer/0</sys:String>
    </Application.Resources>
```

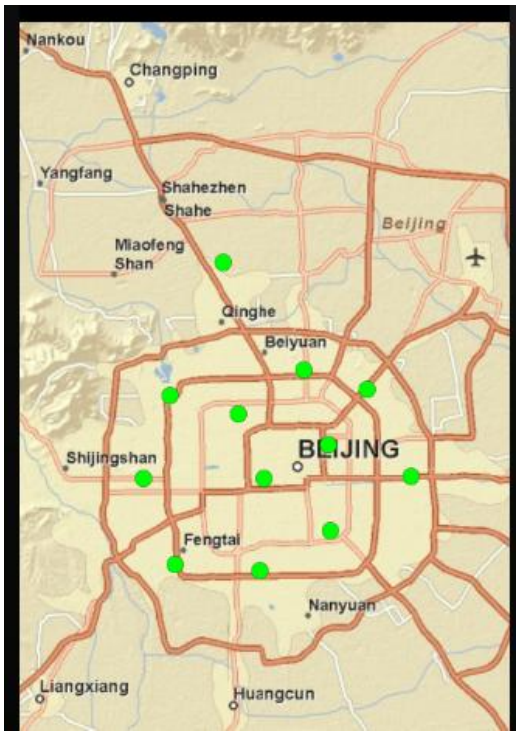
```
    <Application.ApplicationLifetimeObjects>
        <!--Required object that handles lifetime events for the application-->
        <shell:PhoneApplicationService
            Launching="Application_Launching" Closing="Application_Closing"
            Activated="Application_Activated"
Deactivated="Application_Deactivated"/>
    </Application.ApplicationLifetimeObjects>

</Application>
```

将 MainPage.xaml 中的代码改写成：

```
<esri:Map x:Name="map1" Extent="12926244,4840437,12982108,4878585">
            <esri:Map.Layers>
                <esri:ArcGISTiledMapServiceLayer ID="BaseMap"
Url="{StaticResource BaseMap}" />
                <esri:FeatureLayer ID="BusinessLayer"
                    Url="{StaticResource BusinessLayer}"
                    OutFields="*" />
            </esri:Map.Layers>
        </esri:Map>
```

运行一下



可以看到，超市已经显示在地图上面了。但超市的符号依然是 ArcMap 配置地图文档时的默认值，这里我们来利用 FeatureLayer 的 FeatureSymbol 属性，改

变一下超市的地图符号。

在 App.xaml 文件的资源中，添加一个 MarkerSymbol 类型的超市符号：

```
<esriSymbols:MarkerSymbol x:Key="SuperMarket">
            <esriSymbols:MarkerSymbol.ControlTemplate>
                <ControlTemplate>
                    <Canvas>
                        <VisualStateManager.VisualStateGroups>
                            <VisualStateGroup x:Name="SelectionStates">
                                <VisualState x:Name="Selected">
                                    <Storyboard>
                                        <Storyboard.Children>
                                            <Storyboard>
                                                <ObjectAnimationUsingKeyFrames
BeginTime="00:00:00" Storyboard.TargetName="ellipse"
Storyboard.TargetProperty="(UIElement.Visibility)">
                                                    <DiscreteObjectKeyFrame
KeyTime="00:00:00">

<DiscreteObjectKeyFrame.Value>

<Visibility>Collapsed</Visibility>

</DiscreteObjectKeyFrame.Value>
                                                    </DiscreteObjectKeyFrame>
                                                    <DiscreteObjectKeyFrame
KeyTime="00:00:00.01">

<DiscreteObjectKeyFrame.Value>

<Visibility>Visible</Visibility>

</DiscreteObjectKeyFrame.Value>
                                                    </DiscreteObjectKeyFrame>

</ObjectAnimationUsingKeyFrames>
                                            </Storyboard>
                                            <Storyboard
RepeatBehavior="ForEver">
                                                <DoubleAnimation BeginTime="0"
Storyboard.TargetName="ellipse"
```

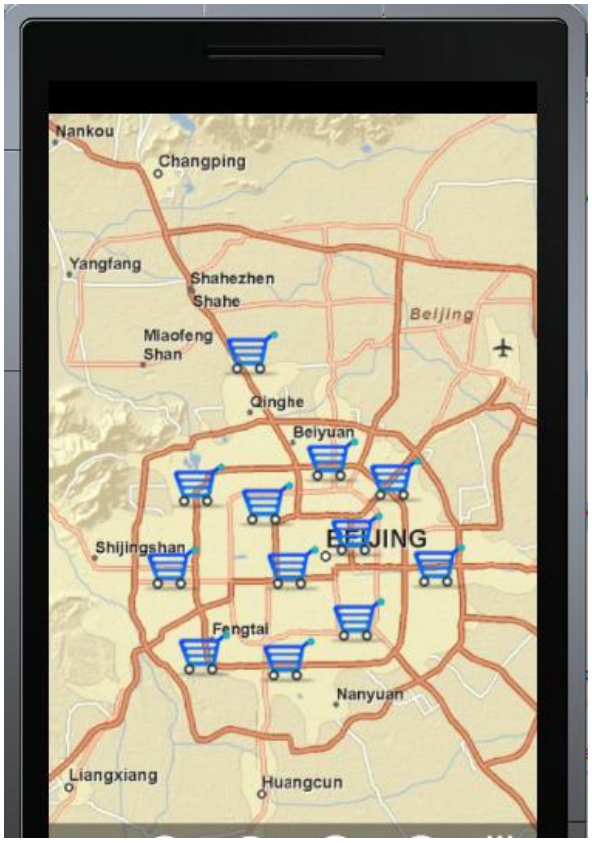
```
Storyboard.TargetProperty="(UIElement.RenderTransform).(ScaleTransform.Scal
eX)" From="1" To="10" Duration="00:00:01" />
                                                <DoubleAnimation BeginTime="0"
Storyboard.TargetName="ellipse"
Storyboard.TargetProperty="(UIElement.RenderTransform).(ScaleTransform.Scal
eY)" From="1" To="10" Duration="00:00:01" />
                                                <DoubleAnimation BeginTime="0"
Storyboard.TargetName="ellipse"
Storyboard.TargetProperty="(UIElement.Opacity)" From="1" To="0"
Duration="00:00:01" />
                                            </Storyboard>
                                        </Storyboard.Children>
                                    </Storyboard>
                                </VisualState>
                                <VisualState x:Name="Unselected" />
                            </VisualStateGroup>
                        </VisualStateManager.VisualStateGroups>
                        <Ellipse Height="20" Width="20" Canvas.Left="-10"
Canvas.Top="-10" RenderTransformOrigin="0.5,0.5" x:Name="ellipse"
IsHitTestVisible="False" Visibility="Collapsed">
                            <Ellipse.RenderTransform>
                                <ScaleTransform />
                            </Ellipse.RenderTransform>
                            <Ellipse.Fill>
                                <RadialGradientBrush>
                                    <GradientStop Color="#00FF0000" />
                                    <GradientStop Color="#FFFF0000"
Offset="0.25" />
                                    <GradientStop Color="#00FF0000"
Offset="0.5" />
                                    <GradientStop Color="#FFFF0000"
Offset="0.75" />
                                    <GradientStop Color="#00FF0000" Offset="1"
/>
                                </RadialGradientBrush>
                            </Ellipse.Fill>
                        </Ellipse>
                        <Image Height="60" Width="60" Canvas.Left="-30"
Canvas.Top="-30" Source="Images/SuperMarket.png" x:Name="ellipse1" />
                    </Canvas>
                </ControlTemplate>
            </esriSymbols:MarkerSymbol.ControlTemplate>
```

```
        </esriSymbols:MarkerSymbol>
```

这个超市符号看起来很复杂，这里我们只需要知道这是一个继承自 MarkerSymbol 类型的自定义符号就行了，与最常用的点符号 SimpleMarkerSymbol 属于并列关系。上面的那么多内容只是自定义了它的 ControlTemplate，结果是当选中了这个超市后，符号会有向外发散的效果。

如果是 ArcGIS Server 10 发布的服务，本身会带有 DrawingInfo 的内容，因此 FeatureLayer 的 Renderer 必须设置为 Null 后，FeatureSymbol 属性才会起作用。最后我们的代码看起来是这样：

```
<esri:Map x:Name="map1" Extent="12926244,4840437,12982108,4878585">
            <esri:Map.Layers>
                <esri:ArcGISTiledMapServiceLayer ID="BaseMap"
Url="{StaticResource BaseMap}" />
                <esri:FeatureLayer ID="BusinessLayer"
                    Url="{StaticResource BusinessLayer}"
                    OutFields="*"
                    FeatureSymbol="{StaticResource SuperMarket}"
                    Renderer="{x:Null}"/>
            </esri:Map.Layers>
        </esri:Map>
```



好了，已经完成了我们的准备工作。下节中，我们来完成点击查询（Identify）的功能。

相关链接：

超市信息的 shapefile 下载：

http://blog.newnaw.com/wp-content/uploads/2011/03/supermarket_shp.zip

Map 控件中添加图层：

http://help.arcgis.com/en/webapi/silverlight/help/0166/016600000019000000.htm

GraphicsLayer 概述：

http://help.arcgis.com/en/webapi/silverlight/help/0166/01660000001q000000.htm

FeatureLayer 概述：

http://help.arcgis.com/en/webapi/silverlight/help/0166/016600000015000000.htm

ArcGIS API for Windows Phone/Silverlight/WPF 中自定义符号：

http://help.arcgis.com/en/webapi/silverlight/samples/SymbolGalleryWeb/start.htm

Silverlight 中的 StaticResource：

http://msdn.microsoft.com/en-us/library/cc189045(v=vs.95).aspx

ArcGIS API for Windows Phone 开发实例(4):点击查看超市信息

本文内容：Silverlight 的自定义 UserControl，NavigationService（页面导航），WP 的 Application Bar，ArcGIS API 的 ElementLayer。

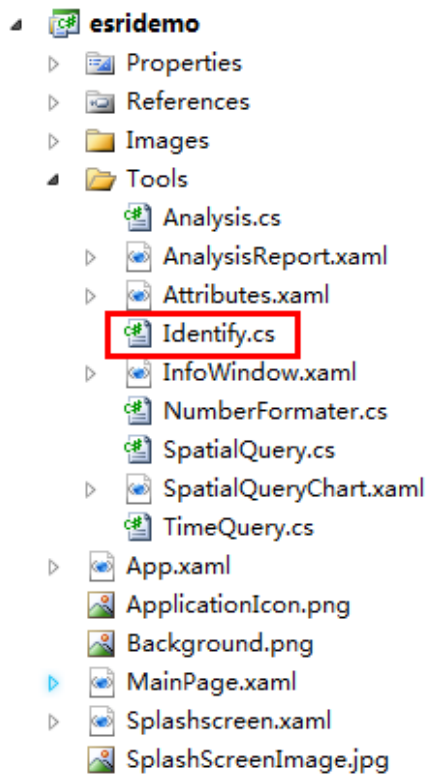
上一节中，已经完成了程序的准备工作，利用 FeatureLayer 来显示超市位置，接下来的几篇文章中我们就来依次实现程序的四个功能点：

- 点击查看某个超市的详细信息；
- 按空间范围查看某几个店面的销售总额；
- 按时间的方式动态查看每个店面的营业情况；
- 对某个店面的近期营业情况做出具体分析。

第一个功能在 GIS 中叫做 Identify。这个功能的作用是，当使用者激活该功能后，点击地图上的某个超市图标，会弹出一个类似气泡的小窗口（InfoWindow）显示超市名称，点击这个气泡后，程序可导航到另一个页面来显示该超市的详细信息。

由于很多功能都是利用单击地图的方式来实现，为了不让这些功能混淆，我将程序中的四个功能分别做成不同的工具（封装成不同的类），点击某个工具后就激活它，此时地图或其中的控件就响应该工具的功能。

我们在工程中新建一个文件夹 Tools，放置与功能点有关的类；并在其中新建一个 Identify.cs 的文件，用来实现我们的第一个功能点。




```
public class Identify
    {
        private bool _isActivated;
        private ElementLayer _elementLayer; //InfoWindow layer
        public Map _map1 { get; set; }
        public GraphicsLayer GLayer { get; set; } //supermarket layer
        public bool IsActivated
        {
            get { return _isActivated; }
            set
            {
                if (_isActivated != value)
                {
                    _isActivated = value;
                    if (value)
                    {
                        if(_elementLayer ==null)
                            _ elementLayer = new ElementLayer();
                        if (!_map1.Layers.Contains(_elementLayer))
                            _map1.Layers.Add(_elementLayer);
                        GLayer.MouseLeftButtonDown +=
GLayer_MouseLeftButtonDown;
                    }
                    else
```

```
                {
                    if (_map1.Layers.Contains(_elementLayer))
                        _map1.Layers.Remove(_elementLayer);
                    GLayer.MouseLeftButtonDown -=
GLayer_MouseLeftButtonDown;
                    if (GLayer.SelectedGraphics.Count() > 0)
                        GLayer.SelectedGraphics.ToList()[0].UnSelect();
                    _ elementLayer.Children.Clear();
                }
            }
        }
    }

    public Identify(Map map,GraphicsLayer glayer)
    {
        _map1 = map;
        GLayer = glayer;
    }
}
```

在这个类（Identify 工具）的构造函数中，有两个参数：一个 Map 控件和一个 GraphicsLayer，分别是我们程序中的 Map 控件和超市图层 FeatureLayer（继承自 GraphicsLayer），因此我们可以在 Identify 工具中控制它们的行为。

IsActived 属性控制着这个工具的状态，如果 IsActived=ture，则该工具处于激活状态，此时地图和超市图层响应 Identify 工具规定的行为，如果 IsActived=false，则代表程序不使用该工具，做一些清理工作。其他工具也是如此，可以用代码来控制某一时刻只能有一个工具处于激活状态。

另外我们还可以看到，Identify 这个类中还有一个 ElementLayer 类型的_Elayer 属性。ElementLayer 是 ArcGIS API for Windows Phone/Silverlight/WPF 中的一种图层类型，主要用来承载 Silverlight 中的原生对象（UIElement），比较关键的一点是，ElementLayer 中的元素会随着地图范围的变化而变化（缩放/平移），而不用自己去处理这些 UIElement 的地理坐标。所以在这里我们就用 ElementLayer 来放

置我们的气泡（InfoWindow）。

来看一下 Identify 的主要功能：

```
void GLayer_MouseLeftButtonDown(object sender, GraphicMouseButtonEventArgs
e)
        {
            if (GLayer.SelectedGraphics.Count() > 0)
                GLayer.SelectedGraphics.ToList()[0].UnSelect();

            Graphic g = e.Graphic;
            _ELayer.Children.Clear();//remove other infowindow
            InfoWindow infoWindow = new InfoWindow(_ELayer, e.Graphic);
            ESRI.ArcGIS.Client.ElementLayer.SetEnvelope(infoWindow, new
Envelope((e.Graphic.Geometry as MapPoint), (e.Graphic.Geometry as MapPoint)));
            _ELayer.Children.Add(infoWindow);

            e.Graphic.Select();
        }
```

GraphicsLayer（超市图层）的 MouseButtonDown 事件可以保证，只有当点击到某个 Graphic（超市）后，才会触发此事件，而点击空白地方是不会触发此事件的。我们在这里并需要做任何查询操作，就可以通过被点击 Graphic 的 Attributes 属性获得该超市的所有信息，因为所有超市的属性信息（还记得 OutFields 属性的“*”吗）在 FeatureLayer 初始化的时候，已经被传送到了客户端（手机上）。点击某个超市后，屏幕上会出现一个气泡（InfoWindow 类）显示超市店名。

这个类是我们自定义的一个 UserControl，可以很方便的给其中加入额外的功能，比如点击了这个 InfoWindow 后，将程序导航到超市详细信息的页面。关于这个 InfoWindow 类详细代码以及在线例子，可以在这里查看。此外 2.1 以后版本的 API 中也提供了 InfoWindow 的 ToolKit，有兴趣的同学可以在这里查看。

Windows Phone 程序中，由于屏幕尺寸原因，如果需要另外显示比较多的内容，则需要将程序导航到一个新的页面中，比如这里我们需要显示超市的详细信息。

所以在 InfoWindow 本身的单击事件中，用下面的代码将程序导航到显示超市详细信息的页面（Attributes.xaml）。

```
private void Canvas_MouseLeftButtonDown(object sender,
System.Windows.Input.MouseButtonEventArgs e)
        {
            //add clicked graphic to app level, so attributes.xaml could access
it
            App app = Application.Current as App;
            if (app.AppParameters.ContainsKey("IdentifyGraphic"))
                app.AppParameters["IdentifyGraphic"] = Graphic;
            else
                app.AppParameters.Add("IdentifyGraphic", Graphic);
            (app.AppParameters["MainPage"] as
PhoneApplicationPage).NavigationService.Navigate(new
Uri("/Tools/Attributes.xaml", UriKind.Relative));
```

```
        }
```

其中重要的类是 NavigationService，每个 Windows Phone 的页面（PhoneApplicationPage）都有这样一个属性。在多页面的 Silverlight 程序中也使用此类来导航。虽然 Navigate 方法可以在页面之间传递参数（类似 asp.net），但仅限于 string 类型，因此我们的超市属性信息集合不好处理。我在 App.xaml.cs 文件里的 Application 类中，定义一个全局的 Dictionary，用于存储我们需要用到的全局变量。

```
//used to store variables which need to pass from one page to another
        public Dictionary<string, object> AppParameters;
```

先将被点击的 Graphic 存入程序的 Silverlight 的全局变量 AppParameters 中，然后在 Attributes.xaml 页面就能取到。Attributes.xaml 中用 ListBox 控件来显示超市信息，由于 Windows Phone 基于 Silverlight 3，而 Graphic 的属性信息 Attributes 是 Dictionary 集合，dictionary binding 在 Silverlight 4 中才支持，所以这里我们依然需要使用到 DictionaryConverter 来在绑定过程中进行转换。Attributes.xaml 页面：

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <ListBox x:Name="listbox" Margin="0,0,-12,0"
ItemsSource="{Binding}">
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <StackPanel Margin="0,0,0,17" Width="432">
                            <Canvas>
                                <Rectangle Fill="#FF83919D" Canvas.Left="0"
Canvas.Top="0" Height="31" Width="800"/>
                                <TextBlock Text="店名"
                            TextWrapping="Wrap" Margin="12,-4,4,5" FontSize="26"
Foreground="White"/>
                            </Canvas>
                            <TextBlock Text="{Binding Path=Attributes,
```

```
Converter={StaticResource MyDictionaryConverter},
ConverterParameter=Name}"
                            TextWrapping="Wrap" FontSize="30"
Margin="12,28,12,3" Foreground="#FF188DC6"/>
                            <Canvas>
                                <Rectangle Fill="#FF83919D" Canvas.Left="0"
Canvas.Top="0" Height="31" Width="800"/>
                                <TextBlock Text="地址"
                        TextWrapping="Wrap" Margin="12,-4,4,5" FontSize="26"
Foreground="White"/>
                            </Canvas>
                            <TextBlock Text="{Binding Path=Attributes,
Converter={StaticResource MyDictionaryConverter},
ConverterParameter=Address}"
                            TextWrapping="Wrap" FontSize="30"
Margin="12,28,12,3" Foreground="#FF188DC6"/>
                            。。。。。。。。。。。。
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>
        </Grid>
```

Attributes.xaml.cs 文件：

```
public partial class Attributes : PhoneApplicationPage
    {
        private Graphic _graphic;
        public Attributes()
        {
            InitializeComponent();

            _graphic = (Application.Current as
App).AppParameters["IdentifyGraphic"] as Graphic;
            GraphicCollection gc = new GraphicCollection();
            gc.Add(_graphic);
            listbox.ItemsSource = gc;
        }
    }
```

完成了整个 Identify 功能后，在 MainPage.xaml 页面中，我们在 Application Bar 中添加四个按钮，点击某个按钮后，就激活相应的工具。比如在点击了 Identify

功能按钮后，我们需要将 Identify 工具的 IsActived 属性设为 true，将其他工具的 IsActived 属性设为 false。最后别忘了，在超市图层 FeatureLayer 初始化时(Update 事件)，实例化我们的四个工具：

```
public partial class MainPage：PhoneApplicationPage
    {
        private Identify _OpIdentify;
        private SpatialQuery _OpSpatialQuery;
        private TimeQuery _OpTimeQuery;
        private Analysis _OpAnalysis;
        private FeatureLayer _FLayer;

        // Constructor
        public MainPage()
        {
            InitializeComponent();
            _FLayer = map1.Layers["BusinessLayer"] as FeatureLayer;
            _FLayer.UpdateCompleted += (s, a) =>
            {
                //Add MainPage to app level, so other pages can navigates from it.
                App app = Application.Current as App;
                if (!app.AppParameters.ContainsKey("MainPage"))
                    app.AppParameters.Add("MainPage", this);
                _OpIdentify = new Identify(map1, _FLayer);
                _OpSpatialQuery = new SpatialQuery(map1, BusyIndicator);
                _OpTimeQuery = new TimeQuery(map1);
                _OpAnalysis = new Analysis(map1,this,BusyIndicator);
            };
        }
    }
```

下一节中，我们来实现按空间范围查看某几个店面的销售总额的功能，也就是 SpatialQuery 这个类。

参考资料：

Windows Phone 中的 Application Bar 和 Application Menu：

http://msdn.microsoft.com/en-us/library/ff431801(v=VS.92).aspx

Windows Phone 中的页面和导航框架：

http://msdn.microsoft.com/en-us/library/ff402536(v=vs.92).aspx

Silverlight 中自定义 UserControl：

http://www.cnblogs.com/Terrylee/archive/2008/03/08/Silverlight2-step-by-step-part10-using-user-controls.html

ArcGIS API for Windows Phone/Silverlight/WPF 中的 ElmentLayer：

http://help.arcgis.com/en/webapi/silverlight/samples/start.htm#ElementLayer

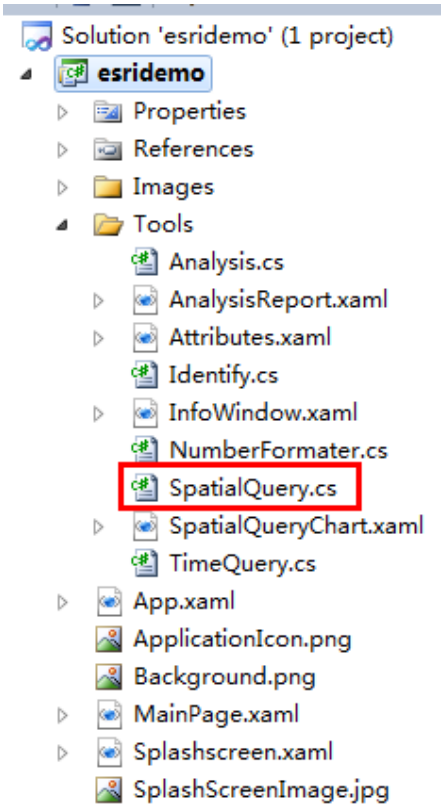
ArcGIS API for Windows Phone 开发实例(5):对超市信息进行空间查询

本文内容：ArcGIS API 中 Task 的概念，QueryTask 的使用，以及 Draw 对象。

空间查询 GIS 中一个非常常用的功能：在地图上画出任意多边形，从自己感兴趣的事物中筛选出与所画多边形有指定空间关系（通常是相交）的要素来，进一步查看。在本次开发实例中，第二个功能就是空间查询。用手势在地图上画一个范围，筛选出落入该范围的超市店面，从而进一步查看它们的营业额统计信息。

ArcGIS API 中，给我们提供了许多 Task 类，来完成一些常见的 GIS 功能，比如属性/控件查询，几何对象的拓扑处理，特定工作流的地理任务等。它们都是已经封装好的 Task 类，使用起来都遵循 3 个步骤的原则：1、为某个 Task 设置所需的相应参数；2、通过 Task 对象向服务器发送处理请求；3、接受服务器端返回的结果。所有的计算和处理工作都是由 ArcGIS Server 发布的 REST 服务来完成，是典型的客户端请求，服务器端相应的流程。

QueryTask 是 ArcGIS API 提供的诸多 Task 之一，它接受 Query 类型的参数。该参数有几个常用的属性，比如 Where 属性和 Geometry 属性，通过对这两个属性的设置，我们就可以完成最常见的属性查询和空间查询功能。依然将空间查询这个功能封装成一个工具，在主界面中进行调用。



这里为了清晰起见，我省去与空间查询功能无关的代码（所有程序代码会在教程完结后提供下载）。要使用 QueryTask 的功能，我们按照前面说三个步骤来做。首先设置好查询参数 Query，然后通过 QueryTask 对象提交查询请求：

```
void _draw_DrawComplete(object sender, DrawEventArgs e)
        {
            Polygon polygon = null;
            if (_usingFreeHand) //geometry is freehand polyline
            {
                Polyline polyline = e.Geometry as Polyline;
                ESRI.ArcGIS.Client.Geometry.PointCollection pc = polyline.Paths[0];
                pc.Add(pc[0]);
                polygon = new Polygon()
                {
                    SpatialReference = map1.SpatialReference,
                };
                polygon.Rings.Add(pc);
            }
            else //geometry is polygon
            {
                polygon = e.Geometry as Polygon;
```

```
            }

            _GLayer.Graphics.Clear();
            Graphic g = new Graphic()
            {
                Geometry = polygon,
                Symbol = new SimpleFillSymbol()
                {
                    Fill=new SolidColorBrush(Color.FromArgb(33,255,0,0)),
                    BorderBrush=new SolidColorBrush(Colors.Red),
                    BorderThickness=2
                }
            };
            _GLayer.Graphics.Add(g); //display the geometry created by Draw object

            QueryTask queryTask = new
QueryTask(App.Current.Resources["BusinessLayer"] as string);
            Query query = new Query();
            //102100 to 4326
            ESRI.ArcGIS.Client.Projection.WebMercator wm = new
ESRI.ArcGIS.Client.Projection.WebMercator();
            query.Geometry = wm.ToGeographic(polygon);
            query.OutFields.AddRange(new string[] { "*" });//return all attributes
fields
            query.SpatialRelationship = SpatialRelationship.esriSpatialRelIntersects;
            queryTask.ExecuteCompleted += new
System.EventHandler<QueryEventArgs>(queryTask_ExecuteCompleted);
            queryTask.Failed += (s, a) =>
            {
                MessageBox.Show("查询失败" + a.Error.Message);
            };
            _busyIndicator.Visibility = Visibility.Visible;
            queryTask.ExecuteAsync(query);
        }
```

代码中，我们首先利用超市图层的服务地址，初始化了一个 QueryTask 对象。对于 Query 参数，这里设置了 Geometry 属性，作为空间查询的图形；对 OutFields 参数的设置表示在查询结果中返回所有属性字段；指定空间关系为与 Geometry 相交。然后通过 ExecuteAsync 方法将查询请求提交到服务器端。注意到在设置

Query 的 Geometry 属性之前，我们对 Polygon 对象做了空间参考的转换，将其从 102100 坐标系（WGS 1984 Web Mercator Auxiliary Sphere，这是地图控件的坐标系）转换到了 4326 坐标系（WGS 1984，这是超市图层的坐标系），如果 Query 中 Geometry 的坐标系不正确，查询结果往往会不可预料。

发出请求后，我们就可以在设置好的 queryTask_ExecuteCompleted 方法中取得查询结果了。

```
void queryTask_ExecuteCompleted(object sender, QueryEventArgs e)
        {
            _busyIndicator.Visibility = Visibility.Collapsed;
            if (e.FeatureSet.Features.Count == 0)
//typeof(e.FeatureSet.Features)==GraphicCollection
            {
                MessageBox.Show("未选中任何店面");
            }
            else
            {
                string str = "是否查看图表详情？";
                if (MessageBox.Show(str,
                "查询结果", MessageBoxButton.OKCancel) == MessageBoxResult.OK)
                {
                    //add selected graphics to app level, so spatialquerychart.xaml
could retrieve them
                    App app = Application.Current as App;
                    if (app.AppParameters.ContainsKey("QueryedGraphics"))
                        app.AppParameters["QueryedGraphics"] =
e.FeatureSet.Features;
                    else
                        app.AppParameters.Add("QueryedGraphics",
e.FeatureSet.Features);

                    (app.AppParameters["MainPage"] as
PhoneApplicationPage).NavigationService.Navigate(new
Uri("/Tools/SpatialQueryChart.xaml", UriKind.Relative));
                }
                _GLayer.ClearGraphics();
            }
```

```
        }
```

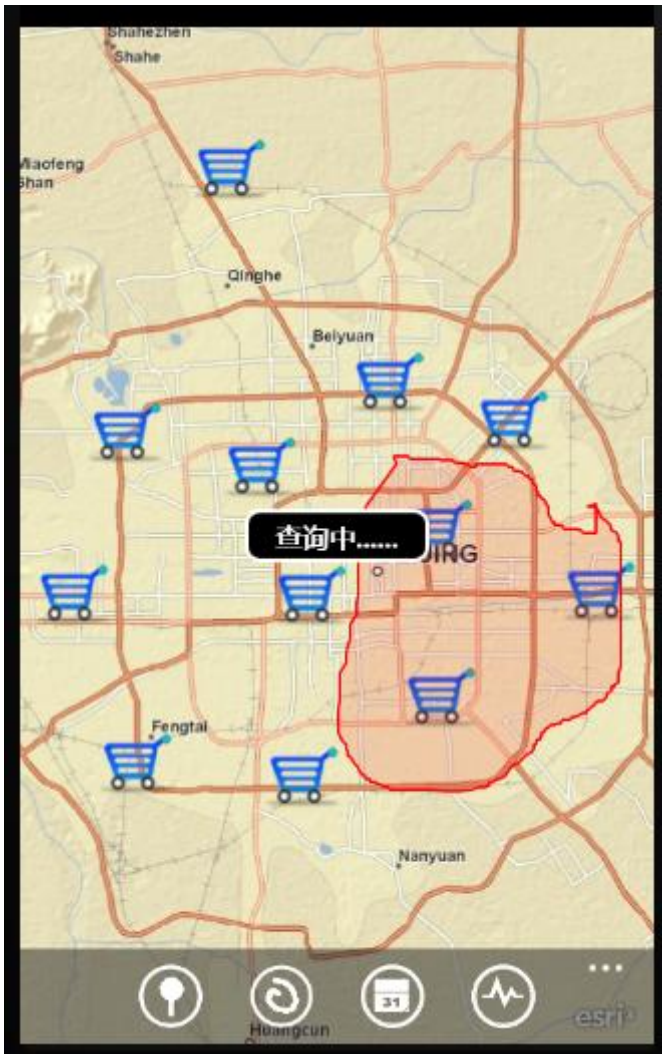
可以看到，查询结果（落入所选范围内的超市）以 Graphic 的形式存储在事件参数中。Graphic 的 Geometry 就是超市的图形信息，而 Attributes 属性中是我们想要的所有营业信息。如果查询结果不为空，我们将其存入全局变量中，以便在另一个页面中用图表的形式来显示。这里我们使用 VisiFire 控件来显示图表，它是一套可用于 WP 上的 Silverlight 插件，此处不进行过多讨论，有兴趣同学可自己搜索。

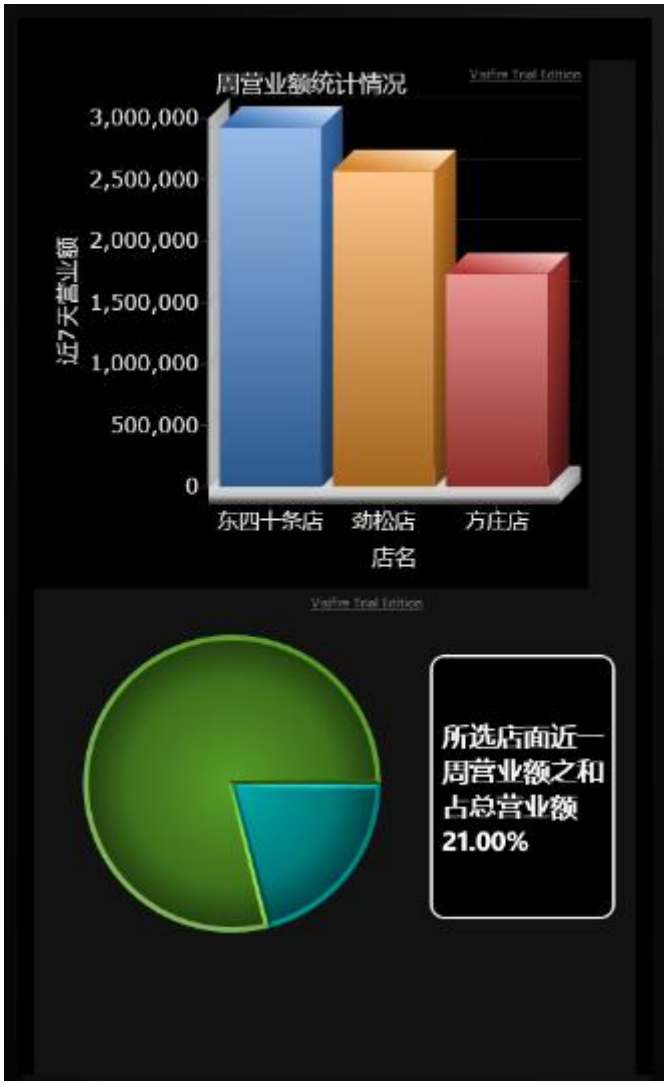
整个查询的过程在 QueryTask 的帮助下变得非常简单。细心的朋友可能会有疑问，我们这个空间查询的图形是如何得到的？在前面的代码中可以看到，发起查询请求的代码是写在_draw_DrawComplete 这个事件中的。_draw 是我们提前定义好的一个 Draw 对象：private Draw _draw;。

为了方便与用户的交互，ArcGIS API 中提供了 Draw 这个类，可以利用鼠标或手势来交互地画出 Point，Polyline，Polygon，Freehand（Polyline）等几何对象，在 2.2 版本的 API 中，Draw 还新增了 arrow，triangle，circle，ellipses 几个原生图形。

使用 Draw 这个对象也比较简单，初始化，设定好要画的几何图形类型，然后将其 IsEnabled 属性设为 True，就进入了交互状态，绘制完毕后，就会触发

DrawComplete 事件，在事件的参数中就可以得到画出的 Geometry 结果，得到结果之后，就可以利用 Graphic 将这个看不见摸不着的 Geometry 显示出来了，这样就达到了交互的目的。我们空间查询的几何对象就是利用 Draw 得到的。





参考资料：

ArcGIS API 中各种 Task 介绍：

http://bbs.esrichina-bj.cn/ESRI/thread-45302-1-1.html

Draw 对象的使用：

http://help.arcgis.com/en/arcgismobile/10.0/apis/WindowsPhone/help/011v/011v00000019000000.htm

QueryTask 的使用：

http://help.arcgis.com/en/arcgismobile/10.0/apis/WindowsPhone/help/011v/011v00000016000000.htm

VisiFire：

http://www.visifire.com/

ArcGIS API for Windows Phone 开发实例(6):对超市信息进行时间查询

本文内容：时态数据的概念，ArcGIS API 中符号的运用

本文来完成按时间对超市营业额信息查看的功能。拖动屏幕上方的滑块，当前日期会随着变化，而地图上显示的内容则是当前日期（某一天）每个店面的营业额状况：符号大小直观的表示了每个店面营业额的大体情况，当地图放大到一定程度时，显示出具体的数字，代表该店面当天营业额的多少。选中“连续查看”后，则可以以动画的形式对一段时间内的营业额进行连续播放。

ArcGIS 10 中强调了时态数据的概念。可以通过矢量或影像属性表中一个（某

一个时刻）或两个字段（某一时间段），来表示该行数据发生或持续的时间。时态数据通常分为两类，一类是随着时间的变化，要素的图形或位置发生了变化（shape 字段的变化），比如随着时间的变化，台风中心位置（点）进行了移动，或者火灾中的过火面积（面）发生了改变；另一类是随着时间的变化，要素的属性值发生了变化（非 shape 字段的变化），比如本例中每个超市的营业额。ArcGIS 中，不同时刻或时段的数据用表中不同的行来存储，为了减少数据冗余，通常也采取多表关联的形式来管理时态数据。

本例中为了数据存储简单起见，将每天的营业额按列存储，也就是每天的营业额存储在了每个超市属性的不同字段中。这样我们就需要自己通过代码来完成时间展示的功能。当然也可以用 ArcGIS 提供的 Transpose Fields 工具，轻松将数据转换为 ArcGIS 能识别的分行存储的时态格式，从而利用 API 中 Map/FeatureLayer 的 TimeExtent 属性来完成我们的功能。

先来看一下我们的 TimeQuery 工具初始化完成的工作：

```
        private bool _isActivated;
        DispatcherTimer _dTimer;//use to control autoplay
        private Border _Border;//stackpanel used to hold the slider and textblock
        private CheckBox _chkboxAutoPlay;//auto paly animation?
        private Slider _slider;//time slider
        private TextBlock _textblock;//date textblock
        private FeatureLayer _FLayer;//business layer
        private GraphicsLayer _GLayerSymbol;//do nothing with Business
FeatureLayer, using another graphicslayer to display symobls
        private GraphicsLayer _GLayerText;//becuase the binding didn't work well in
wp7 api, so using another textsymbol in a graphicslayer
        //instead of a custom markersymbol with a text in it(using binding).
        private double _salesmin, _salesmax;//use to determine symbol size
        private enum SymbolSize
        {
            small=20,
```

```
            middle=40,
            large=60
        }
        public Map map1 { get; set; }
        public bool IsActivated
        {
            get { return _isActivated; }
            set
            {
                if (_isActivated != value)
                {
                    _isActivated = value;
                    if (value)
                    {
                        SlidePanel(Visibility.Visible, TimeSpan.FromMilliseconds(300));
                        _FLayer.Visible = false;
                        map1.Layers.Add(_GLayerSymbol);
                        map1.Layers.Add(_GLayerText);
                        InitSymbols();
                        _chkboxAutoPlay.Visibility = Visibility.Visible;
                        _slider.ValueChanged += _slider_ValueChanged;
                        map1.ExtentChanged += map1_ExtentChanged;
                    }
                    else
                    {
                        SlidePanel(Visibility.Collapsed,
TimeSpan.FromMilliseconds(300));
                        _FLayer.Visible = true;
                        map1.Layers.Remove(_GLayerText);
                        map1.Layers.Remove(_GLayerSymbol);
                        UnInitSymbols();
                        _chkboxAutoPlay.Visibility = Visibility.Collapsed;
                        map1.ExtentChanged -= map1_ExtentChanged;

                        _slider.ValueChanged -= _slider_ValueChanged;
                        _chkboxAutoPlay.IsChecked = false;
                        _dTimer.Stop();
                    }
                }
            }
        }
```

```
        /// <summary>
        /// display or hide the time panel with a slide animation
        /// </summary>
        /// <param name="visible"></param>
        /// <param name="duration"></param>
        private void SlidePanel(Visibility visible, TimeSpan duration)
        {
            DoubleAnimation da = new DoubleAnimation();
            da.Duration = duration;
            if (visible == Visibility.Visible)
            {
                //_Border.Visibility = Visibility.Visible;
                da.From = 0;
                da.To = 100;
            }
            else
            {
                //_Border.Visibility = Visibility.Collapsed;
                da.From = 100;
                da.To = 0;
            }
            Storyboard sb = new Storyboard();
            Storyboard.SetTarget(da, _Border);
            Storyboard.SetTargetProperty(da, new PropertyPath("Height"));
            sb.Children.Add(da);
            sb.Begin();
        }

        private void InitSymbols()
        {
            foreach (Graphic g in _FLayer.Graphics)
            {
                //determin symbol size
                double size = -1;
                if ((double)g.Attributes["D1101"] <= (_salesmax + _salesmin)/2 * 1 /
3)
                    size = (double)SymbolSize.small;
                else if ((double)g.Attributes["D1101"] > (_salesmax + _salesmin) / 2 *
1 / 3 && (double)g.Attributes["D1101"] <= (_salesmax + _salesmin) / 2 * 2 / 3)
                    size = (double)SymbolSize.middle;
                else
                    size = (double)SymbolSize.large;
```

```
                //change Businesslayer symbol to custom symbols and display in
_GLayerSymbol for each store
                Graphic gsymbol = new Graphic()
                {
                    Geometry = g.Geometry,
                    Symbol = new PictureMarkerSymbol()
                    {
                        Source = new BitmapImage(new Uri("../Images/Dollar.png",
UriKind.Relative)),
                    }
                };

                (gsymbol.Symbol as PictureMarkerSymbol).Height = (gsymbol.Symbol
as PictureMarkerSymbol).Width = size;
                (gsymbol.Symbol as PictureMarkerSymbol).OffsetX = (gsymbol.Symbol
as PictureMarkerSymbol).OffsetY = size / 2;
                g.Attributes.Add("symbol", gsymbol);

                //add a graphic with textsymbol, which display sales, to _GLayerText for
each store
                Graphic gtext = new Graphic()
                {
                    Geometry = g.Geometry,
                    Symbol = new TextSymbol()
                    {
                        Foreground = new
SolidColorBrush(Color.FromArgb(225,255,255,255)),
                        Text =
double.Parse(g.Attributes["D1101"].ToString()).ToString("###,###"),
                        OffsetX = -size / 2+15,
                        OffsetY = -size / 2+15,
                        FontSize=30
                    }
                };
                gtext.Symbol.ControlTemplate =
(App.Current.Resources["LegendTextSymbol"] as TextSymbol).ControlTemplate;
                g.Attributes.Add("text", gtext);

                _GLayerText.Graphics.Add(gtext);
                _GLayerSymbol.Graphics.Add(gsymbol);
            }
```

```
            _GLayerText.Visible = false;
        }
```

在这个工具类的构造函数中，添加了一些控件，包括一个控制时间的 slider，一个自动播放的 checkbox，一个显示当前日期的 textblock。另外找出了所有店面所有时间范围内营业额的最大值和最小值，这样有助于对我们的营业额进行“归一化”，从而决定地图符号的大小。

```
        public TimeQuery(Map map)
        {
            map1 = map;
            _GLayerText = new GraphicsLayer();
            _GLayerSymbol= new GraphicsLayer();
            _FLayer = map1.Layers["BusinessLayer"] as FeatureLayer;
            _slider = new Slider()
            {
                Height=84,
                Width=360,
                Minimum=1,
                Maximum=30,
                SmallChange=1,
                LargeChange=5,
                Margin=new Thickness(0,0,0,-30)
            };

            _textblock = new TextBlock()
            {
                Text = "时间:11月1日",
                Foreground=new SolidColorBrush(Colors.White),
                FontSize=20,
                FontWeight=FontWeights.Bold,
                Margin=new Thickness(20,0,0,0),
            };
            StackPanel sp = new StackPanel()
            {
                HorizontalAlignment=HorizontalAlignment.Center,
                VerticalAlignment=VerticalAlignment.Center,
            };
```

```
            sp.Children.Add(_slider);

            //auto play box
            _chkboxAutoPlay = new CheckBox()
            {
                Content = "连续查看",
                FontSize = 20,
                FontWeight=FontWeights.Bold,
                Foreground = new SolidColorBrush(Colors.White),
                Visibility = Visibility.Collapsed,
                Margin = new Thickness(90, -22, 0, 0),
            };

            _dTimer = new DispatcherTimer()
            {
                Interval = TimeSpan.FromMilliseconds(500)
            };
            _dTimer.Tick += (sender, args) =>
            {
                if (_slider.Value == 30)
                    _slider.Value = 1;
                else
                    _slider.Value += 1;
            };

            _chkboxAutoPlay.Click += (s, a) =>
            {
                CheckBox chkbox = s as CheckBox;
                if (chkbox.IsChecked == true)
                {
                    _dTimer.Start();
                }
                else
                {
                    _dTimer.Stop();
                }
            };
            StackPanel sp1 = new StackPanel()
            {
                Orientation = Orientation.Horizontal
            };
            sp1.Children.Add(_textblock);
```

```
            sp1.Children.Add(_chkboxAutoPlay);
            sp.Children.Add(sp1);

            _Border = new Border()
            {
                Background=new SolidColorBrush(Color.FromArgb(225,0,0,0)),
                Width=400,
                Height=0,
                CornerRadius= new CornerRadius(10),
                BorderBrush=new SolidColorBrush(Colors.White),
                BorderThickness=new Thickness(5),
                //Visibility=Visibility.Collapsed,
                VerticalAlignment=VerticalAlignment.Top,
            };
            _Border.Child = sp;

            (map1.Parent as Grid).Children.Add(_Border);


            _salesmin = _salesmax = 0;
            //find day minsales and maxsales in all graphics and all days
            for (int i = 1; i <= 30; i++)
            {
                string strDay = string.Empty;
                if (i < 10)
                    strDay = "0" + i.ToString();
                else
                    strDay = i.ToString();
                double fieldmax = (from graphic in _FLayer.Graphics
                                   select graphic).Max(a =>
(double)a.Attributes["D11" + strDay]);
                _salesmax = _salesmax > fieldmax ? _salesmax : fieldmax;
                double fieldmin = (from graphic in _FLayer.Graphics
                                   select graphic).Min(a => (double)a.Attributes["D11"
+ strDay]);
                _salesmin = _salesmin < fieldmin ? _salesmin : fieldmin;
            }
        }
```

当控制时间的 slider 发生变化时，我们就根据当前日期的营业额，计算出每个超市店面符号的大小，从而使用新的符号来显示。

```
        void _slider_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            int day = (int)e.NewValue;
            //change text
            _textblock.Text = string.Format("时ºiÀ间?:11月?{0}日¨?", day.ToString());

            ChangeSymbolAndText(day);
        }

        /// <summary>
        /// change symbols and text according to current day
        /// </summary>
        /// <param name="day"></param>
        private void ChangeSymbolAndText(int day)
        {
            string strDay = string.Empty;
            if (day < 10)
                strDay = "D11"+"0" + day.ToString();
            else
                strDay = "D11" + day.ToString();

            foreach (Graphic g in _FLayer.Graphics)
            {
                //determin symbol size
                double size = -1;
                if ((double)g.Attributes[strDay] <= (_salesmax - _salesmin) * 1 / 3)
                    size = (double)SymbolSize.small;
                else if ((double)g.Attributes[strDay] > (_salesmax - _salesmin) * 1 / 3
&& (double)g.Attributes[strDay] <= (_salesmax - _salesmin) * 2 / 3)
                    size = (double)SymbolSize.middle;
                else
                    size = (double)SymbolSize.large;

                PictureMarkerSymbol symbol = (g.Attributes["symbol"] as
Graphic).Symbol as PictureMarkerSymbol;
                symbol.Height = symbol.Width = size;
                symbol.OffsetX = symbol.OffsetY = size / 2;

                TextSymbol text = (g.Attributes["text"] as Graphic).Symbol as
TextSymbol;
                text.Text =
```

```
double.Parse(g.Attributes[strDay].ToString()).ToString("###,###");
                text.OffsetX = -size / 2 + 10;
                text.OffsetY = -size / 2 + 10;
                //text.FontSize = size == (double)SymbolSize.small ? size : size - 10;
            }
        }
```

最后，还要控制营业额数字的显示范围。当比例尺较小时，超市图标分布较为密集，具体营业额数据不予显示；当比例尺较大时，才显示每个超市的具体营业额数字。

```
        void map1_ExtentChanged(object sender, ExtentEventArgs e)
        {
            if (map1.Resolution > 50.2185141425366)//38
                _GLayerText.Visible = false;
            else
                _GLayerText.Visible = true;
        }
```

至此，我们的第三个功能也完成了。在这里请各位朋友思考一下，如果使用 ArcGIS 格式的时态数据，我们按时间查看超市营业额的功能该怎么做？相较于本文中的方法，各自的优缺点是什么？

参考资料：

ArcGIS 中的时态数据：

http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/What_is_temporal_data/005z00000001000000/

ArcGIS API for Windows Phone 开发实例(7):利用 Geoprocessing 分析超市的营业状况

本文内容：Geoprocessing

前几节中，我们利用 GIS 这个工具，对连锁超市店面的营业状况做了充分的展示。这一节中，我们利用 GIS 特有的 Geoprocessing 功能，结合超市的营业数据统计，对超市的营业状况做一个比较合理的判断，从而有助于经营者及时调整策略，适应市场的变化。

我们的大致分析思路是这样的：对于每个超市来说，会有固定的消费群体，一般通常是距离该超市一定范围内的常驻居民，这部分占绝大多数（一次性路过的消费者人数较少，可以忽略不计）；首先我们利用 GIS 这个工具，设法得到每个超市周围一定范围内的常驻人口数，做为该超市的潜在消费群体；然后根据过去一定时间内，这个品牌所有超市的营业额统计，计算出所有超市店面潜在消费群体每人每天的大约消费额度是多少，做为该品牌超市的一个平均竞争力（每人/天购买力是多少）；这样就能利用这个平均竞争力数值和某个超市潜在消费人口数，预估出该超市一定时期内大约的营业额收入。如果实际营业额与这个预估相差不大，我们认为该超市营业状况正常；如果营业额统计明显小于这个预估值，则有可能是该超市营业范围内新增了竞争对手的店面，或竞争对手采取了促销措施等手段分流了购买力，从而提醒经营者及时对营业策略做出调整，避免进一步的损失。

超市店面的营业额统计我们有了，下一步就需要获得每个超市潜在的消费人群数量。我们分两步来完成这项工作：第一步计算出距离某个超市店面一定时间

车程的地理范围，作为该超市潜在消费群体的居住区域；第二部查询出该区域内的常驻人口数。

GIS 正是用于处理所有与地理分布相关任务的一个有力工具，Geoprocessing 则是它的核心所在。上面的两个步骤我们就可以利用 Geoprocessing 轻松完成。ArcGIS Server 中，将具有一定逻辑相关的一组任务封装成一个处理流程，发布成 Geoprocessing Service（简称 GP 服务），通过 Task 的形式供客户端调用。客户端只需提供相关的参数后，就可直接得到处理结果，而不需要理会服务器端复杂的处理流程。ArcGIS Online 为我们提供了两个现成的 Geoprocessing Service：CreateDriveTimePolygons 和 PopulationSummary。前者用来获得距某点一定时间的车程范围，后者用来获得某范围的常驻人口，刚好符合我们的需求。因此只需调用这两个 GP 服务，即可得到我们想要的数据，从而完成相关计算工作。

注：CreateDriveTimePolygons 适用于美国境内，在此仅作说明之用；由于数据原因，程序内所使用的北京市内的对应 GP 服务无法公开。PopulationSummary 适用于全世界范围，但结果精度不做保证。关于两个服务的使用限制，请查看相关页面。

我们依然创建一个工具类，取名 Analysis，该类包含两个 GeoProcessing Task（Geoprocessor）：_gpDriveTime 和_gpPopulation，分别用于获取某超市的营业范围和该范围内的常驻人口数。

```
        private Geoprocessor _gpDriveTime = new
Geoprocessor(App.Current.Resources["GPDriveTime"] as string);
        private Geoprocessor _gpPopulation = new
Geoprocessor(App.Current.Resources["GPPopulation"] as string);
        _gpDriveTime.ExecuteCompleted += gpDriveTime_ExecuteCompleted;
                        _gpDriveTime.Failed += (s, a) =>
```

```
                {
                    MessageBox.Show("ServiceArea"+a.Error.Message + "\n"
+ a.UserState);
                };
                _gpPopulation.ExecuteCompleted +=
gpPopulation_ExecuteCompleted;
                _gpPopulation.Failed += (s, a) =>
                {
                    ChangeMapOpacity(-1);
                    MessageBox.Show("Population"+a.Error.Message + "\n" +
a.UserState);
                };
```

GP Task 是 ArcGIS API for Windows Phone 提供给我们用来调用 Geoprocessing Service 的类，与 Query Task，Identify Task 等相同，使用分为三个步骤：准备参数，提交请求，处理结果。我们首先来看一下 CreateDriveTimePolygons 这个 GP 服务所需的参数：

ArcGIS Services Directory

Home > Network > ESRI_DriveTime_US (GPServer) > CreateDriveTimePolygons

Task: CreateDriveTimePolygons

Display Name: CreateDriveTimePolygons

Category:

Help URL: http://sampleserver1c.arcgisonline.com/arcgisoutput/Network_ESRI_DriveTi

Execution Type: esriExecutionTypeSynchronous

Parameters:

Parameter: Input_Location
- **Data Type:** GPFeatureRecordSetLayer
- **Display Name:** Input Location
- **Direction:** esriGPParameterDirectionInput
- **Default Value:**
  - **Geometry Type:** esriGeometryPoint
  - **Spatial Reference:** 4326
  - **Fields:**
    - FID *(Type: esriFieldTypeOID, Alias: FID)*
    - Shape *(Type: esriFieldTypeGeometry, Alias: Shape)*
    - Id *(Type: esriFieldTypeInteger, Alias: Id)*
- **Parameter Type:** esriGPParameterTypeRequired
- **Category:**

Parameter: Drive_Times
- **Data Type:** GPString
- **Display Name:** Drive Times
- **Direction:** esriGPParameterDirectionInput
- **Default Value:** 5 10 15
- **Parameter Type:** esriGPParameterTypeOptional
- **Category:**

总共有三个参数。Direction 为 Input 的参数，即输入参数，有两个：Input_Location 和 Drive_Times。因此我们只需准备好这两个参数，即可调用该服务。最后结果中，会包含 Direction 为 Output 的参数，即就是我们的结果了（esriGeometryPolygon）。

```
void _FLayer_MouseLeftButtonDown(object sender, GraphicMouseButtonEventArgs e)
        {
            //……
            //prepare parameters for GP Service
```

```
            GraphicCollection gc = new GraphicCollection();
            gc.Add(e.Graphic);
            CalculateServiceArea(gc);
            //……
        }
        /// <summary>
        /// find the service area of a given supermarket
        /// </summary>
        /// <param name="gc">the graphic collection contains the
supermarket</param>
        private void CalculateServiceArea(GraphicCollection gc)
        {
            ESRI.ArcGIS.Client.Projection.WebMercator wm = new
ESRI.ArcGIS.Client.Projection.WebMercator();
            GraphicCollection graphicCollection = new GraphicCollection();//using new
gc to remain point(121000) stay on map
            foreach (Graphic g in gc)
            {
                Graphic graphic = new Graphic()
                {
                    Geometry=wm.ToGeographic(g.Geometry)
                };
                graphicCollection.Add(graphic);
            };

            FeatureSet fs = new FeatureSet(graphicCollection);
            List<GPParameter> parameters = new List<GPParameter>();
            parameters.Add(new GPFeatureRecordSetLayer("Input_Facilities", fs));
            parameters.Add(new GPString("Drive_Time_Values", "20 40"));
            //execute the GP Task
            _gpDriveTime.ExecuteAsync(parameters);
        }

void gpDriveTime_ExecuteCompleted(object sender, GPExecuteCompleteEventArgs e)
        {
            ESRI.ArcGIS.Client.Projection.WebMercator wm = new
ESRI.ArcGIS.Client.Projection.WebMercator();
            foreach (GPParameter parameter in e.Results.OutParameters)
            {
                if (parameter is GPFeatureRecordSetLayer)
                {
                    GPFeatureRecordSetLayer gpLayer = parameter as
```

```
GPFeatureRecordSetLayer;
                    //40 min
                    gpLayer.FeatureSet.Features[0].Symbol = new SimpleFillSymbol()
                    {
                        Fill=new SolidColorBrush(Color.FromArgb(119,153,255,153)),
                        BorderBrush=new
SolidColorBrush(Color.FromArgb(255,153,255,153)),
                        BorderThickness=2
                    };
                    gpLayer.FeatureSet.Features[0].Geometry =
wm.FromGeographic(gpLayer.FeatureSet.Features[0].Geometry);
                    _GLayer.Graphics.Add(gpLayer.FeatureSet.Features[0]);
                    //20 min
                    gpLayer.FeatureSet.Features[1].Symbol = new SimpleFillSymbol()
                    {
                        Fill = new SolidColorBrush(Color.FromArgb(119, 153, 153,
255)),
                        BorderBrush = new SolidColorBrush(Color.FromArgb(255, 153,
153, 255)),
                        BorderThickness = 2
                    };
                    gpLayer.FeatureSet.Features[1].Geometry =
wm.FromGeographic(gpLayer.FeatureSet.Features[1].Geometry);
                    _GLayer.Graphics.Add(gpLayer.FeatureSet.Features[1]);

                    map1.ZoomTo(gpLayer.FeatureSet.Features[0].Geometry);

                    //……
                    (_BusyIndicator.Child as TextBlock).Text = "查询服务区内人口
数……";
                    //继续查询该范围人口数
                    //……
                }
            }
        }
```

可以看到，我们在 parameters 集合中添加了 GP 服务所要求的所有输入参数，然后执行；而结果中，我们也取得了 GP 服务中的输出参数。这里说明一点，GP 服务的参数 GPParameter 比较灵活，新建参数时的名称，类型一定要保持与 Service Directory 服务列表中一致，否则可能导致调用失败。另外由于 GP 服务的空间参考（4326）与我们的底图（102100）不一致，所以做了坐标转换。

此外我们还注意到一点，CreateDriveTimePolygons 服务的执行方式是“esriExecutionTypeSynchronous”即同步调用。这种方式一般用于执行过程时间较短的 GP 服务，缺点是客户端需要在返回结果之前等待。具体可参考这里。

获得了超市的服务范围之后，需要用第二个 GP 服务，PopulationSummary 来获得该范围内的人口总数；之后就可按照前面的分析思路，来计算出该超市的营业额预期，与实际营业状况比较后，可得出相应的结论。

参考资料：

Geoprocessing 概念：

http://help.arcgis.com/en/arcgisdesktop/10.0/help/002s/002s00000001000000.htm

ArcGIS Server 中的 Geoprocessing Service：

http://help.arcgis.com/en/arcgisserver/10.0/help/arcgis_server_dotnet_help/index.html#//009300000028000000.htm

ArcGIS API for Windows Phone 中调用 Geoprocessing Service：

http://help.arcgis.com/en/arcgismobile/10.0/apis/windowsphone/help/011v/011v0000001m000000.htm

至此，《ArcGIS API for Windows Phone 开发实例》已经结束，希望大家能通过这几篇文章对 ArcGIS API for Windows Phone 有所了解。目前 API 的版本是 2.2

beta，这仅仅是一个开始，随着以后大家对移动 GIS 的关注越来越多，相信 esri 也会在 API 中逐步推出更多的功能来满足我们更多的应用需求。

在前不久召开的 MIX11 大会上，微软展示了新的 Windows Phone Mongo，该版本的 SDK 中不仅 WP 的性能有了很大提升，而且有着全面支持真正的多任务，Silverlight 4，内嵌全功能 IE9 浏览器，支持 socket 通信，允许同时使用 Silverlight 和 XNA 编程模型（SL 中就会有原生 3D 功能支持）等令人兴奋的新特性。伴随着市场占有率不断提升的这一事实，有着真正硬实力的 Windows Phone 有理由让我们相信不仅对于开发者，同样对于用户而言，它能给我们带来更多机会。