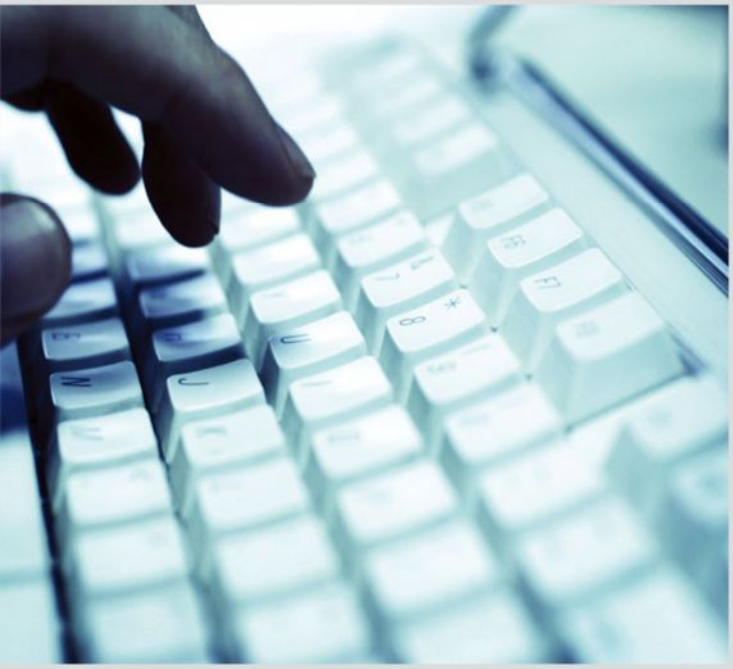
ArcGIS ESRI

(version 1.0)



ArcGIS API for Flex 教程

Technical Marketing Department

2008–11–24

ESRI China (Beijing)
Geographic Thinking

版权声明

本文档版权为 ESRI 中国（北京）有限公司所有。未经本公司书面许可，任何单位和个人不得以任何形式摘抄、复制本文档的部分或全部，并以任何形式传播。

本文档版本依照以下原则进行（version x.yz）：

x: 文档结构变化，设计框架升级

y: 具体内容更新，章节内容修正

z: 文字格式调整，奇数为包含对上一版修改记录或批注的稿子，偶数为对上一版修改稿的定稿

版本修订记录

版本	提交日期	负责人	描述
0.10	2008 年 11 月 13 日	张宝才	初级篇初稿
0.20	2008 年 11 月 14 日	张宝才	中级篇初稿
0.30	2008 年 11 月 17 日	张宝才	高级篇初稿
0.32	2008 年 11 月 19 日	张宝才	格式整理
0.33	2008 年 11 月 20 日	陈欣	内容修改，格式整理
0.34	2008 年 11 月 21 日	张宝才	内容修改，格式整理，代码用暗紫色标识
0.90	2008 年 11 月 24 日	陈欣	格式调整，提交 0.90 版本
1.00	2008 年 11 月 25 日	王昊	提交 1.00 版本

目录

一 初级篇 4

1.1 环境搭建 4

1.1.1 概述 4

1.1.2 相关资源 4

1.1.3 创建基本程序 5

1.1.4 参考资料及其他 6

1.2 地图的创建 6

1.2.1 概述 6

1.2.2 创建地图 7

1.2.3 事件（Event） 8

1.3 绘制与符号化 9

1.3.1 概述 9

1.3.2 绘图控件的使用 9

1.3.3 符号化 9

1.3.4 地图的浏览 10

1.4 TASK 之 QUERY & FIND & IDENTIFY 10

1.4.1 概述 10

1.4.2 在 mxml 文档中嵌入 ActionScript 11

1.4.3 QueryTask 11

1.4.4 FindTask 13

1.4.5 IdentifyTask 15

1.4.6 InfoWindow 17

二 中级篇 18

2.1 地理定位（LOCATOR） 18

2.1.1 概述 18

2.1.2 地理编码（GeoCode） 18

2.1.3 逆地理编码（Reverse Geocode） 20

2.2 地理处理（GEOPROCESSING） 21

2.3 几何服务（GEOMETRY SERVICE） 22

2.4 打印地图（PRINT） 23

2.5 GEORSS 的读取 24

2.5.1 GeoRSS 概述 24

2.5.2 GeoRSS 读取 24

三 高级篇 26

3.1 WEBSERVICE 的使用 26

3.2 ESRI TILEMAP 四叉树索引研究 28

3.3 自定义控件的开发 31

3.4 与 GOOGLE MAP 的融合 32

一 初级篇

1.1 环境搭建

1.1.1 概述

ArcGIS API for Flex 是 ESRI 2008 年新推出的 WebGIS 客户端开发包，用于富互联网应用程序 RIA（Rich Internet Applications）的开发，为 ArcGIS Server 9.3 提供了一套全新的开发方式，其优点是运行速度快，为用户提供优秀的用户体验。

使用 ArcGIS API for Flex 可以实现如下效果：

1）显示地图数据并与之交互

2）在服务器上执行空间处理模型并显示结果

3）基于 ArcGIS Online 的地图服务显示自己的业务数据

4）根据属性或者位置查找数据并显示结果

5）查找地址并显示结果

6）创新结果的显示方式

7）创建 Mushups 应用

1.1.2 相关资源

ArcGIS API for Flex 线上帮助系统的网址：

http://resources.esri.com/arcgisserver/apis/flex/index.cfm?fa=samples

ArcGIS API for Flex 下载地址：

http://resources.esri.com/arcgisserver/apis/flex/index.cfm?fa=home

开发 ArcGIS API for Flex 的程序需要 flex 环境的支持。

Flex SDK 3 下载地址：

http://opensource.adobe.com/wiki/display/flexsdk/download?build=3.0.1.2012&pkgtype=1

为了开发方便，最好使用 Flex Builder 开发环境，从 Adobe 官方网站下载试

用版：

http://download.macromedia.com/pub/flex/flex_builder/FB3_win.exe

1.1.3 创建基本程序

1）打开 Flex Builder，创建一个 Flex 工程（名称 Demo）。

2）右键单击工程名，选择属性，在属性对话框中选择 Flex Build Path 如图 1-1 ，选择 Libaray Path 选项卡，单击 Add SWC 把下载的 ArcGIS API for Flex 添加进去，环境就配置好了。

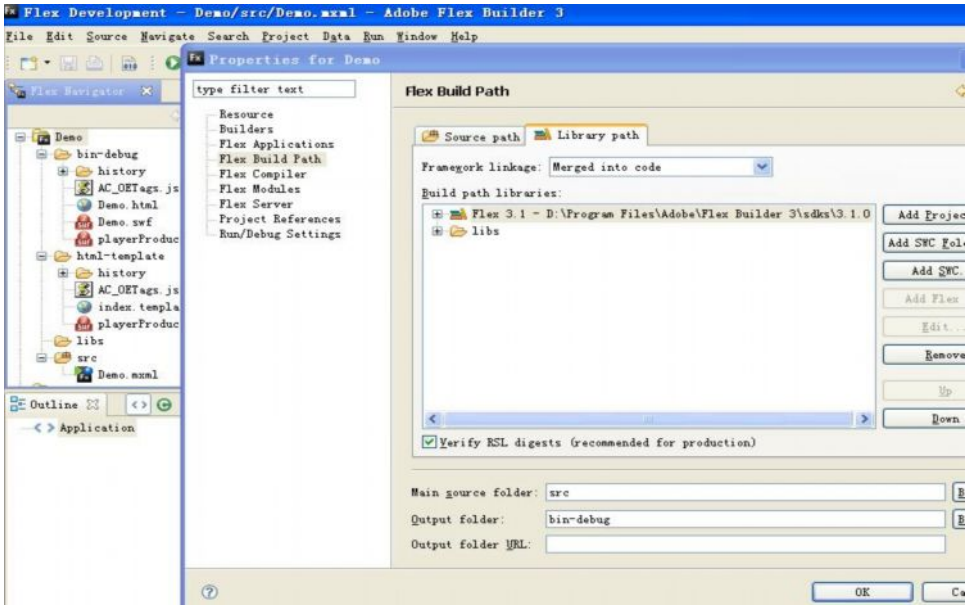


图 1-1

在 Demo.mxml 文件中输入下面这代码：

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:esri="http://www.esri.com/2008/ags"
    pageTitle="First ArcGIS API for Flex program"
    styleName="plain">
    <esri:Map crosshairVisible="true">
        <esri:ArcGISTiledMapServiceLayer
url="http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer
" />
    </esri:Map>
</mx:Application>
```

这是一个最基础的ArcGIS API for Flex应用程序，编译运行即可。

完整的例子可以在/samples/1.1 下面找到。

1.1.4 参考资料及其他

文章部分内容参考了以下文章：

http://www.gisempire.com/blog/user1/1/200888102647.html

关于配置环境的详细信息还可以参考下面的视频：

http://resources.esri.com/help/9.3/arcgisserver/apis/flex/video/ArcGISFlexAPI_beta_demo_skin.swf

1.2 地图的创建

1.2.1 概述

地图（map）是 ArcGIS API for Flex 中最重要的对象之一，map 负责对 layer 进行组织，在逻辑上 map 可以包含任意多个 layer。

在 ArcGIS API for Flex 中，主要包括以下几种 layer：

ArcGISDynamicMapServiceLayer ：用户通过 ArcGIS Server REST API 以动态地图服务的形式对数据进行访问（Allows you to work with a dynamic map service resource exposed by the ArcGIS Server REST API）。

ArcGISImageServiceLayer：用户通过 ArcGIS Server REST API 以影像服务的形式对数据进行访问（Allows you to work with an image service resource exposed by the ArcGIS Server REST API）。

ArcGISTiledMapServiceLayer ：用户通过 ArcGIS Server REST API 以影像服务的形式对数据进行访问（Allows you to work with a cached map service resource exposed by the ArcGIS Server REST API）。

ArcIMSMapServiceLayer ：用户可以访问 ArcIMS image service 提供的数据(Allows you to work with an ArcIMS image service)。

GraphicsLayer：用于显示用户在客户端绘制的图形要素的图层，该图层可以包含一个或多个图形要素（ A layer that contains one or more Graphic features）。

1.2.2 创建地图

使用 ArcGIS API for Flex 创建一个地图，在 mxml 文件中直接使用<esri:Map>标签即可，如下：

```
<esri:Map width="100%" height="50%" id="EsriMap" resize="EsriMapResize(event);" extentChange="ESRIMapExtentChange(event);" mouseMove="OnDrawMouseMove(event)" />
```

<esri:Map>标签定义一个 map 对象，其中 width，height，id 为 map 的属性，resize="EsriMapResize(event);"extentChange="ESRIMapExtentChange(event);" mouseMove="OnDrawMouseMove(event)" 为 map 的消息以及消息响应函数。

关于 map 的属性和消息的详细列表请参考以下地址：

http://resources.esri.com/help/9.3/arcgisserver/apis/flex/apiref/index.html

map 创建好之后，使用 layer 把需要访问的数据加载到 map 上,在<esri:Map>标签下面创建 <esri:ArcGISTiledMapServiceLayer > 子标签。<esri:ArcGISTiledMapServiceLayer >标签定义一个 layer 对象，如下：

```
<esri:ArcGISTiledMapServiceLayer url="http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer" />。
```

其最重要的属性就是 url，url 定义了访问数据的地址，在上面的代码中 url 指向一个 arcgisonline 提供的 TiledMap 服务。

有关 arcgisonline 提供的服务信息请查看下面地址：

http://server.arcgisonline.com/ArcGIS/rest/services/

控制 map 的初始显示范围，在<esri:Map>标签下创建<esri:extent>标签即可。使用<esri:extent>标签来定义一个范围对象，例如设定 map 显示的范围为北京地区：

```
<esri:extent>
        <esri:Extent xmin="114" ymin="38" xmax="116" ymax="42"/>
</esri:extent>
```

注意标签的大小写，标签是大小写敏感的。

在这个例子中，我们将添加上面提到的 3 种类型的 layer 到 map 中，并控制其初始显示范围为北京地区，如下面代码：

```
<esri:Map >
    <esri:ArcGISTiledMapServiceLayer visible="{mapSwitch.selectedIndex == 0}" url="http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer
```

```
" />
    <esri:ArcGISDynamicMapServiceLayer visible="{mapSwitch.selectedIndex == 1}"
    url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Popul
ation_World/MapServer"/>
    <esri:ArcIMSMapServiceLayer visible="{mapSwitch.selectedIndex == 2}"
            serviceHost="http://www.geographynetwork.com"
             serviceName="ESRI_World" />
    <esri:extent>
            <esri:Extent xmin="114" ymin="38" xmax="116" ymax="42"/>
    </esri:extent>
</esri:Map>
```

1.2.3 事件（Event）

地图创建之后，如果要得到当前鼠标单击位置的坐标，则需要处理 Map 的事件，通过响应事件来获取鼠标单击位置的坐标。在 ActionScript 中的事件和 JAVA 的事件类似，是一种典型的观察者模式。

在 ArcGIS API for Flex 中提供的事件主要有：DrawEvent、 ExtentEvent、 FindEvent、GeometryServiceEvent、GeoprocessorEvent、GraphicEvent 、 IdentifyEvent 、 LayerEvent 、 LocatorEvent 、MapEvent 、MapImageEvent NavigationEvent、PanEvent 、QueryEvent 、ZoomEvent 。

关于事件的详细描述可以参考下面的地址：

http://resources.esri.com/help/9.3/arcgisserver/apis/flex/apiref/index.html

下面的例子实现一个简单的事件响应：当双击 map 的时候，弹出对话提示正在双击 map。

修改<esri:Map>标签，响应 doubleClick 事件。

```
<esri:Map    doubleClick="Alert.show('doubleClick Map')">
```

为了能够使用Alert，在 <mx:Application>下面插入下面代码：

```
<mx:Script>
        <![CDATA[
            import mx.controls.Alert;
        ]]>
 </mx:Script>
```

关于mxml文件中嵌入ActionScript脚本部分将在后面详细讲解。

完整的例子可以在/samples/1.2 下面找到。

1.3 绘制与符号化

1.3.1 概述

ArcGIS API for Flex 在客户端提供了强大的绘制功能，丰富的符号支持，可以通过简单的调用来完成复杂的绘制工作，达到很好的交互效果，提供完美的用户体验。

1.3.2 绘图控件的使用

完成客户端的绘制功能，需要在 map 上定义一个 GraphicsLayer，客户端绘制的结果都会添加到 GraphicsLayer 上。

使用<esri:GraphicsLayer> 标签定义一个 GraphicsLayer 对象。

```
<esri:GraphicsLayer id="myGraphicsLayer"/>
```

id 唯一标识一个 GraphicsLayer。

GraphicsLayer 创建后，进行绘制工作只要使用 ArcGIS API for Flex 提供的 draw 工具即可。 ArcGIS API for Flex 已经对绘制进行了很好的封装，通过简单的调用即可完成复杂的绘制工作，类似于 AE 中的 drawToolbar。

首先，定义一个 draw 控件，使用<esri:Draw>标签来完成创建工作，如下面的代码：

```
<esri:Draw id="drawToolbar" map="{myMap}" graphicsLayer="{myGraphicsLayer}" />
```

id 唯一标识一个 Draw 对象，map 属性绑定到需要绘制的地图上，graphicsLayer 属性绑定到需要绘制到的 graphicsLayer 上。

Draw 对象创建后，调用 Draw 的 activate 方法来激活所需要的工具进行交互的绘制，例如绘制多边形，只需要调用 activate(Draw.POLYGON)函数即可，其中 POLYGON 为 Draw 的静态成员。

在绘制完图形后，释放掉绘制工具，调用 deactivate()函数即可。

1.3.3 符号化

在 map 中对绘制的图形进行符号化，设置 Draw 对象的 markerSymbol，

lineSymbol，fillSymbol 属性即可。其中 markerSymbol 为点的符号，lineSymbol 为线的符号，fillSymbol 为填充符号。

设置 Draw 对象的 markerSymbol，lineSymbol，fillSymbol 属性之前，需要创建 markerSymbol，lineSymbol，fillSymbol 对象。以 markerSymbol 为例，使用 <esri:SimpleMarkerSymbol> 标签定义一个 markerSymbol，如下面代码：

```
<esri:SimpleMarkerSymbol id="sms" style="square" color="0xFF0000" size="11"/>
```

id 唯一标识一个 markerSymbol，style 属性为 markerSymbol 的风格，color 属性为 markerSymbol 的颜色，size 属性为 markerSymbol 的大小。

lineSymbol 和 fillSymbol 的定义与 markerSymbol 类似。

1.3.4 地图的浏览

与 Draw 控件类似，在 ArcGIS API for Flex 中提供了地图浏览的工具，通过简单的调用，就能够实现放大、缩小、平移、复位等地图浏览操作。

使用<esri:Navigation>标签定义地图浏览控件：

```
<esri:Navigation id="navToolbar" map="{myMap}"/>
```

map 属性绑定需要进行浏览操作的地图。

Navigation 对象创建后，使用 Navigation 的 activate 方法来激活需要的工具进行浏览。例如放大操作：

```
activate(Navigation.ZOOM_IN);
```

释放浏览工具，调用 Navigation 对象的 deactivate()函数即可。

完整的例子可以在 samples/1.3 下面找到。

1.4 Task 之 Query & Find & Identify

1.4.1 概述

在 GIS 系统中，Query 、Find 、 Identify 都是最常用，最基本的功能，也是使用频率最高的功能之一。在 ArcGIS API for Flex 中提供了 QueryTask，FindTask，IdentifyTask 来完成 Query 、Find 、 Identify 的功能。

这些 Task 共同的特点是，在客户端组织好参数信息，执行，在服务器端返回

结果，对结果在客户端处理。

1.4.2 在 mxml 文档中嵌入 ActionScript

在使用 QueryTask，FindTask，IdentifyTask 之前，首先介绍一下如何在 mxml 文档中嵌入 ActionScript.。通过在 mxml 文档中嵌入 ActionScript 脚本我们可以更加灵活的使用 ArcGIS API for Flex。

关于 ActionScript 的语法可以参考 ActionScript 的相关书籍。

要在 mxml 文档中嵌入 ActionScript，需要使用<mx:Script>标签：

```
<mx:Script>
    <![CDATA[
    ]]>
</mx:Script>
```

ActionScript 是一种类 java 语言，它本身有一个 AVM，把 ActionScript 编译成 java 的代码，然后再通过 JVM 转换成字节码执行。

1.4.3 QueryTask

在 ArcGIS API for Flex 中进行查询操作，需要定义一个查询任务面板。

使用<esri:QueryTask>标签来定义一个查询任务面板。

```
<esri:QueryTask id="queryTask"
url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/5">
    <esri:Query id="query"
        text="{qText.text}"
        returnGeometry="true"
        spatialRelationship="esriSpatialRelEnvelopeIntersects">
        <esri:outFields>
            <mx:String>MED_AGE</mx:String>
            <mx:String>POP2007</mx:String>
        </esri:outFields>
    </esri:Query>
</esri:QueryTask>
```

id 唯一标识这个查询任务，url 指定查询服务的地址。

<esri:Query>定义一个查询，text 属性指定查询的内容，<esri:OutFields>子标签定义 Query 查询的结果返回哪些字段的内容。

QueryTask 定义好之后，在界面上定义一个文本输入框和一个查询按钮来调用这个 QueryTask：

```
<mx:Panel title="Query a layer (search for a state)"
layout="horizontal" backgroundColor="0xB2BFC6" borderStyle="solid">
```

```
    <mx:TextInput width="100%" id="qText" enter="doQuery()"text="California"/>
    <mx:Button label="Do Query" click="doQuery()"/>
</mx:Panel>
```

文本输入框用来输入查询的内容，button 用来执行查询的动作。

下面实现查询功能：

首先，使用 import 指令引入我们需要的命名空间，和 java 基本一样。

```
<mx:Script>
    <![CDATA[
        import com.esri.ags.Graphic;
        import com.esri.ags.tasks.FeatureSet;
        import com.esri.ags.tasks.Query;
        import mx.controls.Alert;
        import mx.rpc.AsyncResponder;
    ]]>
</mx:Script>
```

其次，定义 doQuery()函数（注意 ActionScript 代码要放到<mx:Script>标签中）：

```
private function doQuery() : void
{
    queryTask.execute( query, new AsyncResponder( onResult, onFault ));
}
```

在 doQuery()函数中直接调用了 queryTask 的 execute 方法，这是一个异步调用。成功响应 onResult 函数，失败则响应 onFault 函数。最后查询得到的结果将在 onResult 函数中处理。

下面是 onResult 函数的定义以及实现：

```
private function onResult( featureSet : FeatureSet, token : Object = null ) : void
{
 var displayFieldName : String = featureSet.displayFieldName;
    for each ( var myGraphic : Graphic in featureSet.features )
    {
      // ToolTip
      myGraphic.toolTip = "The 2007 population of "
      + myGraphic.attributes[displayFieldName] + " was "
      + myNumberFormatter.format(myGraphic.attributes.POP2007)
      + "\nMedian Age: " + myGraphic.attributes.MED_AGE + ".";
     // show on map
     myGraphicsLayer.add( myGraphic );
     }
}
```

查询结果返回一个 FeatureSet，在 onResult 函数中遍历这个 FeatureSet，然后把每个 feature 绘制到 GraphicLayer 上，并显示 ToolTip。

如果查询失败，则在 onFault 函数中对失败进行处理，如下代码：

```
private    function onFault( info : Object, token : Object = null ) : void
{
      Alert.show( info.toString() );
}
```

弹个对话框出来告诉用户失败的详细信息。

完整例子可以在 samples/1.4/3 目录下找到。

1.4.4 FindTask

在 ArcGIS API for Flex 中执行 Find 任务，需要使用 FindTask。

使用<esri:FindTask >标签定义一个 FindTask 对象，如下面的代码：

```
<esri:FindTask id="findTask" executeComplete="executeCompleteHandler(event)"
url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_StatesCitiesRivers_USA/MapServer/find"/>
```

id 唯一标识 FindTask，executeComplete 事件指定 Find 任务完成后调用的函数，url 指定提供 Find 服务的地址。

使用 FindTask，还需要一个 FindParameters 对象来指定查找的参数。

```
<esri:FindParameters id="myFindParams" returnGeometry="true" contains="true"
searchText="{fText.text}" layerIds="[2]" searchFields="['STATE_ABBR','STATE_NAME']"
/>
```

id 唯一标识 FindParameters，searchText 指定查找的字符串，searchFields 指定在哪些字段查找。

定义好 FindTask 和 FindParameters 之后，在界面上定义一个文本输入框和一个 button 来完成 Find 功能的调用工作。

```
<mx:HBox width="100%" height="40" backgroundColor="0xDDDDFF" paddingTop="10"
          horizontalAlign="center">
    <mx:Text text="Search for names of States"/>
    <mx:TextInput maxWidth="400" id="fText" enter="doFind()" text="NEW"/>
    <mx:Button label="Find" click="doFind()"/>
</mx:HBox>
```

文本输入框用来输入查找的内容，button 用来执行查询的动作。

实现 Find 功能：

首先，使用 import 指令引入需要的命名空间：

```
import com.esri.ags.events.FindEvent;
```

其次，定义 doFind()函数：

```
Private function doFind():void
 {
        findTask.execute( myFindParams );
  }
```

在 doFind()函数中 FindTask 对象直接调用了 execute 方法，并把 FindParameters 对象做为传入参数。

下面实现 executeCompleteHandler()函数，这个函数在定义 FindTask 的时候被指定为响应 executeComplete 事件的函数，即当 Find 任务完成的时候会响应 executeCompleteHandler()函数，Find 得到的结果将在这个函数里进行处理。

```
private function executeCompleteHandler( event : FindEvent ) : void
{
      myGraphicsLayer.clear();
      var graphic : Graphic;
      resultSummary.text = "Found " + event.findResults.length + " results.";
      for (var i : Number = 0; i < event.findResults.length; i++)
      {
            graphic = event.findResults[i].feature;
            graphic.toolTip = event.findResults[i].foundFieldName + ": " +
              event.findResults[i].value;
            myGraphicsLayer.add(graphic);
       }
}
```

在 executeCompleteHandler 函数中遍历 Find 的结果，给每一个 graphic 添加一个 toolTip，显示该结果对应的字段名和字段值。

同时把查询到的结果显示到 DataGrid 中：

```
<mx:DataGrid
      dataProvider="{findTask.executeLastResult}"
      scroll="true" width="100%" height="40%">
      <mx:columns>
            <mx:DataGridColumn dataField="layerId" headerText="Layer ID" width="70"/>
            <mx:DataGridColumn dataField="layerName" headerText="Layer Name"/>
            <mx:DataGridColumn dataField="foundFieldName"
                                    headerText="Found Field Name"/>
            <mx:DataGridColumn dataField="value" headerText="Found Field Value"/>
        </mx:columns>
</mx:DataGrid>
```

只要把 DataGrid 的 dataProvider 绑定到 findTask.executeLastResult 就可以了。

完整例子可以在 samples/1.4/4 目录下找到。

1.4.5 IdentifyTask

Identify 是 GIS 中比较常用的工具之一，在 ArcGIS API for Flex 中，使用 IdentifyTask 来实现 Identify 的功能。

使用<esri:IdentifyTask>标签来定义一个 Identify Task 对象，如下面代码：

```
<esri:IdentifyTask id="identifyTask"
        identifyComplete="identifyCompleteHandler(event)"
         url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_States
         CitiesRivers_USA/MapServer"/>
```

id 唯一标识 IdentifyTask ，identifyComplete 事件绑定 identifyCompleteHandler()函数，url 指定 IdentifyTask 服务的地址。

执行 Identify，需要定义一个 IdentifyParameters 对象。下面使用 ActionScript 来定义一个 IdentifyParameters 对象，使用<IdentifyParameters>标签也可以完成同样的工作。

```
var identifyParams : IdentifyParameters = new IdentifyParameters();
            identifyParams.returnGeometry = true;
            identifyParams.tolerance = 3;
            identifyParams.width = 600;
            identifyParams.height = 550;
            identifyParams.geometry = geometry;
identifyParams.layerOption = IdentifyParameters.LAYER_OPTION_ALL;
            identifyParams.mapExtent = map.extent;
            identifyTask.execute( identifyParams );
```

其中 tolerance 是容差半径，geometry 是用来做 identify 的几何。

下面在界面上使用鼠标交互来完成 Identify：

首先，定义一个 draw 控件 ：

```
<esri:Draw id="drawToolbar" map="{map}"
graphicsLayer="{myGraphicsLayer}" drawEnd="drawEndHandler(event)">
```

drawEnd 事件绑定到 drawEndHandler(event)函数，这个事件会在绘制完成后触发。

使用 ActionScript 脚本实现 drawEndHandler 和 identifyCompleteHandler 函数：

首先，引入命名空间：

```
import com.esri.ags.Graphic;
import com.esri.ags.events.DrawEvent;
import com.esri.ags.events.IdentifyEvent;
import com.esri.ags.geometry.Geometry;
import com.esri.ags.symbol.Symbol;
import com.esri.ags.tasks.IdentifyParameters;
import com.esri.ags.tasks.IdentifyResult;
```

```
import com.esri.ags.toolbars.Draw;
```

其次，定义并实现 drawEndHandler()函数：

```
private function drawEndHandler(event:DrawEvent):void
{
    var geometry : Geometry = event.geometry;
    var identifyParams : IdentifyParameters = new IdentifyParameters();
    identifyParams.returnGeometry = true;
    identifyParams.tolerance = 3;
    identifyParams.width = 600;
    identifyParams.height = 550;
    identifyParams.geometry = geometry;
    identifyParams.layerOption = IdentifyParameters.LAYER_OPTION_ALL;
    identifyParams.mapExtent = map.extent;
    identifyTask.execute( identifyParams );
}
```

在 drawEndHandler() 函数中定义了一个 identifyParams，每次调用 drawEndHandler 的时候，identifyTask 都会把 identifyParams 作为传入参数调用 execute 方法去执行。执行完成后会响应 identifyCompleteHandler()函数。

定义并实现 identifyCompleteHandler()函数：

```
private function identifyCompleteHandler(event:IdentifyEvent):void
{
    for each (var result:IdentifyResult in event.identifyResults)
    {
    myGraphicsLayer.add(result.feature);
        switch (result.feature.geometry.type)
        {
            case Geometry.MAPPOINT:
            {
                var mp:MapPoint = result.feature.geometry as MapPoint;
                var txt :Text=new Text();
                txt.text="Point";
                myMap.infoWindow.content=txt;
                myMap.infoWindow.show(mp);
                break;
             }
            case Geometry.POLYLINE:
            {
                var lin:Polyline =    result.feature.geometry as Polyline;
                var txt :Text=new Text();
                txt.text="Line";
                myMap.infoWindow.content=txt;
                myMap.infoWindow.show(lin.extent.center);
```

```
                    break;
                }
                case Geometry.POLYGON:
                {
                    var pgn:Polygon=    result.feature.geometry as Polygon;
                    var txt :Text=new Text();
                    txt.text="polygon";
                    myMap.infoWindow.content=txt;
                    myMap.infoWindow.show(pgn.extent.center);
                    break;
                  }
            }
        }
}
```

在 identifyCompleteHandler()函数中，遍历 identifyResults，将 identifyResults 添加到 GraphicsLayer 上。

完整例子可以在 samples/1.4/4 目录下找到。

1.4.6 InfoWindow

InfoWindow 是 ArcGIS API for Flex 中提供的类似于标注的窗口，可以用来显示用户自定义的信息，可以是文本、图片，也可以是复杂的自定义组件。

使用 InfoWindow，只要设置 map 的 infoWindow 属性即可。如下面代码：

```
var canvas:Canvas = new Canvas();
 var txtTem :Text = new Text();
 txtTem.Text= "aa";
canvas.addChild(txtTem);
 myMap.infoWindow.content = canvas;
var mapPnt2:MapPoint = new MapPoint(114.1547298,30.5127677);
myMap.infoWindow.show(mapPnt2);
```

content 设置 infoWindow 的内容，show 方法设置 infoWindow 的显示位置。

关于 InfoWindow 的使用已经结合在上一小节中（IdentifyTask），例子以及代码请参考上例。

二 中级篇

2.1 地理定位（Locator）

2.1.1 概述

地理定位（Locator）即地理编码（Geocode）又称地址匹配（address-matching），是指建立地理位置坐标与给定地址一致性的过程。也是指在地图上找到并标明每条地址所对应的位置。地理编码是 GIS 中比较重要的一个功能。

2.1.2 地理编码（GeoCode）

在 ArcGIS API for Flex 中使用地理编码和执行查询任务类似，首先使用<esri:Locator>标签定义一个 Locator 对象：

```
<esri:Locator
id="locateTask"
url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/ESRI_Geocode_NA/GeocodeServer"/>
```

id 唯一标识 Locator,url 指向提供 Locator 服务的地址。

Locator 定义之后，在界面上定义一组文本输入框和一个执行按钮来调用这个 Locator：

```
<mx:Panel title="Find an address" top="5" horizontalCenter="0">
        <mx:Form>
            <mx:FormItem label="Street">
                <mx:TextInput width="100%" id="address" text="380 New York St"/>
            </mx:FormItem>
            <mx:FormItem label="City">
                <mx:TextInput width="100%" id="city" text="Redlands"/>
            </mx:FormItem>
            <mx:FormItem label="Zip Code or postal code">
                <mx:TextInput width="100" id="zip" text="92373"/>
            </mx:FormItem>
            <mx:FormItem label="State/Province">
                <mx:TextInput width="100" id="state" text="CA"/>
            </mx:FormItem>
            <mx:FormItem label="Country">
                <mx:ComboBox id="country" selectedIndex="1">
```

```
                    <mx:ArrayCollection>
                        <mx:String>Canada</mx:String>
                        <mx:String>USA</mx:String>
                    </mx:ArrayCollection>
                </mx:ComboBox>
            </mx:FormItem>
            <mx:FormItem>
                <mx:Button label="Find Address" click="doQuery()"/>
            </mx:FormItem>
        </mx:Form>
        <mx:Text id="myInfo" width="100%" color="0x00FF00" textAlign="center" />
    </mx:Panel>
```

文本输入框用来输入地址的详细信息，button 用来执行查询的动作。

实现地理编码的功能：

首先，定义一个 Object 来存储地址的详细信息，包括 Address，City，State，Zip，Country 等信息。

然后，定义一个 Array 来存储输出字段的名称。

最后，把 myAddress 和 myOutFields 作为输入参数调用 locateTask 对象的 addressToLocations 方法。

具体代码请参考下面的代码：

```
private function doGeoCode() : void
{
    var myAddress:Object = {
            Address: address.text,
            City: city.text,
            State: state.text,
            Zip: zip.text,
            Country: country.text
            };
    var myOutFields:Array = ["Loc_name"];
    locateTask.addressToLocations(myAddress, myOutFields, new AsyncResponder(onResult,
onFault));
 function onResult( candidates : Array, token : Object = null ) : void
 {
      if (candidates.length > 0)
       {
            var addressCandidate : AddressCandidate = candidates[0];
            var myGraphic : Graphic = new Graphic();
            myGraphic.geometry = addressCandidate.location;
            myGraphic.symbol = mySymbol;
```

```
                    myGraphic.toolTip = addressCandidate.address.toString();
                    myGraphic.id = "graphic";
                    myGraphicsLayer.add( myGraphic );
                    myMap.centerAt( addressCandidate.location );
                    myInfo.htmlText = "<b>Found:</b><br/>"
                     +addressCandidate.address.toString();
            }
            else
            {
                    myInfo.htmlText = "<b><font color='#FF0000'>Found
nothing :(</b></font>";
                    Alert.show("Sorry, couldn't find a location for this address"
                                    + "\nAddress: " + address.text
                                    + "\nCity: " + city.text
                                    + "\nZIP Code: " + zip.text
                                    + "\nState: " + state.text
                                    + "\nCountry: " + country.text);
                }
        }
        function onFault( info : Object, token : Object = null ) : void
            {
                myInfo.htmlText = "<b>Failure</b>" + info.toString();
                        Alert.show("Failure: \n" + info.toString());
             }
}
```

其中，addressToLocations 的定义为：

```
public function addressToLocations(address:Object, outFields:Array = null, responder:IResponder
= null):void
```

关于该函数的详细信息请参考下面地址：

http://resources.esri.com/help/9.3/arcgisserver/apis/flex/apiref/index.html

该方法是一个异步调用的方法，成功则调用 onResult（）函数，失败则调用 onFault 函数。这个过程和查询任务的调用过程相同。在 onResult 函数中我们处理返回的结果，地图中心平移到返回的地址，并增加一个点来标识。在 onFault 中我们处理失败，将通过对话框返回错误信息。

2.1.3 逆地理编码（Reverse Geocode）

逆地理编码就是把地理坐标解析成对应的地址。Locator 也提供的进行逆地理编码的函数，即 locationToAddress 方法。该方法的调用方式和

addressToLocations 方法相同，只是参数略有不同。locationToAddress 方法的定义为：

locationToAddress(location:MapPoint, distance:Number = 0, responder:IResponder = null)

关于该函数的详细信息请参考下面地址：

http://resources.esri.com/help/9.3/arcgisserver/apis/flex/apiref/index.html

完整例子可以在 samples/2.1 下面找到。

2.2 地理处理（Geoprocessing）

在 ArcGIS API for Flex 中可以使用 GP 服务来进行地理处理。在 ArcGISOnline 上的 gp 服务有 CreateDriveTimePolygons 和 Viewshed，下面以 CreateDriveTimePolygons 服务为例来实现在 ArcGIS API for Flex 中调用 GP 服务。

首先，使用<esri:Geoprocessor >标签定义一个 gp 服务，url 指向提供 gp 服务的地址。

```
<esri:Geoprocessor id="gp" url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Network/ESRI_DriveTime_US/GPServer/CreateDriveTimePolygons" />
```

定义 GP 对象后，再定义一个 GP 的参数对象来传递参数，如下面代码：

```
var params:Object = {
                "Input_Location" : featureSet,
                "Drive_Times" : driveTimes
                    };
```

其中参数 Input_Location 和 Drive_Times 与发布的 GP 服务相关，不同的 GP 服务有不同的参数。

关于 GP 参数设置的详细信息请参考下面地址：

http://resources.esri.com/help/9.3/arcgisserver/apis/rest/index.html?gpexecute.html

定义好 GP 对象和 GP 参数对象之后，下面实现在地图上单击，然后计算 DriveTimes，并把得到的结果绘制在地图上的功能。

其中 GP 调用的代码如下：

```
var featureSet:FeatureSet = new FeatureSet([graphic]);
var params:Object = {
        "Input_Location" : featureSet,
        "Drive_Times" : driveTimes
        };
gp.execute(params, new AsyncResponder( onResult, onFault ));
function onResult(
        gpResult : ExecuteResult,
```

```
        token : Object = null
        ) : void
{
          var pv : ParameterValue = gpResult.parameterValues[0];
          var fs : FeatureSet = pv.value as FeatureSet;
          graphicsLayer.graphicProvider = fs.features;
   }
   function onFault( info : Object, token : Object = null ) : void
   {
              Alert.show( info.toString() );
    }
```

定义好 GP 之后，直接调用 GP 的 execute 的方法即可，成功响应 onResult，失败响应 onFault。

关于 GP 调用的详细信息请参考下面地址：

http://resources.esri.com/help/9.3/arcgisserver/apis/flex/apiref/index.html

完整例子可以在 samples/2.2 目录下找到。

2.3 几何服务（Geometry Service）

Geometry Service 顾名思义，就是提供针对几何层级的服务，例如 Project， Simplify ， Buffer，Areas And Lengths ， Lengths 等。

详细信息可以参考下面地址：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Geometry/GeometryServer

上面的地址发布了 5 个 Geometry Service：Project， Simplify ， Buffer，Areas And Lengths ， Lengths。

下面以 Buffer 为例，来实现在 ArcGIS API for Flex 中调用 Geometry service。

首先，使用<esri:GeometryService>标签定义一个 GeometryService 对象。

```
<esri:GeometryService id="myGeometryService"
url="http://sampleserver2.arcgisonline.com/ArcGIS/rest/services/Geometry/GeometryServer"/>
```

id 唯一标识 Geometry Service，url 指定提供 Geometry Service 的地址。

和 Identify 工具类似，进行 Buffer 操作首先要创建一个需要做 buffer 的几何，然后定义一个 BufferParameters ，执行 Buffer 操作，最后将 buffer 的结果绘制到 GraphicsLayer 上。

下面介绍一下 BufferParameters 参数的意义：distances 为 buffer 半径，features

为需要做 buffer 的要素集合，unit 为单位，bufferSpatialReference 为 buffer 操作时的空间参照系。

Buffer 调用的代码如下：

```
var bufferParameters : BufferParameters = new BufferParameters();
 bufferParameters.features = [point];
bufferParameters.distances = [3000];
bufferParameters.unit = BufferParameters.UNIT_METER;
bufferParameters.bufferSpatialReference=newSpatialReference( 602113);
 myGeometryService.addEventListener
 (GeometryServiceEvent.BUFFER_COMPLETE, bufferCompleteHandler);
      myGeometryService.buffer( bufferParameters );
```

关键的部分是 bufferParameters.features 参数设置的 features 一定要有空间参考系，不然 Buffer 不成功。

关于 Geometry Service 的详细信息请参考下面地址：

http://resources.esri.com/help/9.3/arcgisserver/apis/flex/apiref/index.html

http://resources.esri.com/help/9.3/arcgisserver/apis/flex/help/index.html

其余的绘制操作，将结果绘制到 GraphicsLayer 上的操作请参考 1.3 节绘制与符号化部分。

完整例子可以在 samples/2.3 目录下找到。

2.4 打印地图（Print）

在 Flex 中打印地图全部依赖于 Flex 的打印功能。

在 Flex 中打印，使用 mx.printing.FlexPrintJob 的功能即可。因为 flex 的打印功能实现比较简单，所以就不做详细讲解。主要的内容就是定义一个 FlexPrintJob 对象，然后调用 FlexPrintJob 对象的 send（）方法把打印指令发送给打印机。

代码如下：

```
private function doPrint(myFlexPrintJobScaleType:String):void
{
    var myPrintJob:FlexPrintJob = new FlexPrintJob();
    if (myPrintJob.start())
    {
        try
        {
```

```
                myMap.zoomSliderVisible = false;
                myPrintJob.addObject(myPanel, myFlexPrintJobScaleType);
                myMap.zoomSliderVisible = true;
            }
            catch (e:Error)
            {
                Alert.show( e.toString() );
            }
            myPrintJob.send();
        }
    }
```

详细示例可以在 samples/2.4 中找到。

2.5 GeoRSS 的读取

2.5.1 GeoRSS 概述

GeoRSS 是指给 RSS feed 添加位置信息或者地理标签到的一种协议，它标准化了表达地理信息的方式，简捷并高效地满足了给 Web 内容添加位置信息的需求。时空信息在 RSS 里以 XML 格式指定，和 RSS 0.9、RSS 1.0、RSS 2.0、Atom 甚至其他基于 XML 格式的文件协作，实现在网络上编码时空信息。

GeoRSS 提供了一种地理位置搜索与聚合的方案，并且可以用于地理分析、地理搜索、地理聚合。例如在指定地点 10 公里范围内，所有可能受地震影响的地物的信息，在自己出行道路中出现交通事故的位置点等等。只要 RSS 包含了地理位置信息，就可以将应用进行扩展。

2.5.2 GeoRSS 读取

在 ArcGIS API for Flex 中虽然没有直接对 GeoRSS 进行支持，但在 ArcGIS API for Flex 的 samples 里提供了对 GeoRSS 读取的库，在 arcgis_api_for_flex_1_0\ArcGIS_Flex\samples\src\com\esri\ags\samples 目录下面，以 as 文件的形式存在，主要包括 3 个 as 文件：Namespaces.as，GeoRSSProvider.as，GeoRSSUtil.as 。

关于 GeoRSS 读取可以详细剖析这三个文件，下面主要介绍如何使用这个库

来读取 GeoRSS。

首先，使用<samples:GeoRSSProvider>标签定义一个 GeoRSSProvider 对象，如下面代码：

```
<samples:GeoRSSProvider id="georss" error="errorHandler(event)"/>
```

id 唯一标识 GeoRSSProvider。

然后，定义一个 ComboBox 和一个 Button。

```
<mx:ControlBar>
        <mx:Spacer width="100%"/>
        <mx:ComboBox id="cb" dataProvider="{arr}" labelField="url"/>
        <mx:Button label="Load" click="load_clickHandler()"/>
      <mx:Spacer width="100%"/>
</mx:ControlBar>
```

ComboBox 用来存储 GeoRSS 提供者的地址，Button 用来加载 GeoRSS。

其中 ComboBox 的属性 dataProvider 绑定 arr。arr 的定义如下：

```
private var arr : Array = [
{url:"feed://earthquake.usgs.gov/eqcenter/catalogs/7day-M5.xml",
                        label:"Earthquake",
                        georssFunction:usgs},
                {url:"feed://www.gdacs.org/XML/RSS.xml",
                        label:"Image",
                        georssFunction:gdacs},
{url:"http://ws.geonames.org/rssToGeoRSS?feedUrl=http://feeds.reuters.com/reuters/topNews?format=xml",
                        label:"News",
                        georssFunction:reuters},
{url:"feed://api.flickr.com/services/feeds/geo/?tags=ESRI&amp;lang=en-us&amp;format=rss_200",
                        label:"UFO",
                        georssFunction:flickr}
            ];
```

其中 georssFunction 为进行解释 url 提供的 GeoRSS 的回调函数。每个地址提供的 GeoRSS 的结构都是不同的，所以要写回调函数自己去解析。

关于 GeoRSS 的更多详细信息请参考下面的地址：

http://georss.org/

http://www.georss.cn/

Button 的 click 事件响应 load_clickHandler()函数：

```
private function load_clickHandler() : void
```

```
{
        map.infoWindow.content = m_image;
        map.infoWindow.label = cb.selectedItem.label;
        georss.url = cb.selectedItem.url;
        georss.georssFunction = cb.selectedItem.georssFunction;
 }
```

load_clickHandler()函数实现很简单，只是把 combox 中选中的 GeoRSS 的 url 和回调函数的名字赋值给 georss 对象。

其中回调函数的定义如下：

```
private function rsoe(
                    arrcol : ArrayCollection,
                    x : XML
                ) : void
                {
                    const geometry : Geometry = GeoRSSUtil.toGeometry(x);
                    const graphic : Graphic = new Graphic( geometry, sms1);
                    graphic.toolTip = x.description;
                    arrcol.addItem( graphic );
                }
```

arrcol : ArrayCollection 为返回参数，带回解析后的 graphic。

x : XML 为传入参数，传入需要解析的 xml。

不同来源的 GeoRSS 回调函数实现是不同的，但参数必须保持一致。

最后定义一个:GraphicsLayer 对象，把 georss.result 绑定到 GraphicsLayer 的 graphicProvider 属性上。

```
<esri:GraphicsLayer graphicProvider="{georss.result}"/>
```

完整例子可以在 samples/2.5 里找到。

三 高级篇

3.1 WebService 的使用

Flex 本身对 webservices 有着良好的支持，我们可以调用互联网上的各种 webservices 来结合 ESRI 的 map 做出自己想要的东西。

下面以在地图上显示天气预报为例，介绍 Flex 中调用 webservices。

首先，找到提供天气预报的 webservices：

http://www.webxml.com.cn/WebServices/WeatherWebService.asmx?

然后，使用 <mx:WebService>标签定义一个 WebService 对象：

```
<mx:WebService id="weatherWS"
wsdl="http://www.webxml.com.cn/WebServices/WeatherWebService.asmx?WSDL"
showBusyCursor="true"/>
```

id 唯一标识 webservice，wsdl 指向提供 webservice 的地址。

在 Application 创建完成的时候调用这个 webservice，得到结果后直接显示到 map 上 。

在<mx:Application>标签的 creationComplete 事件绑定 Init()方法：

```
creationComplete="Init()"
```

使用 ActionScript 脚本实现 Init()方法：

```
private function Init():void
{
    weatherWS.addEventListener(ResultEvent.RESULT, WSGetWeatherResult);
    weatherWS.getWeatherbyCityName("武汉");
 }
```

在 Init()方法中首先添加 WebService 调用后 ResultEvent 的 Listener，结果返回后响应 WSGetWeatherResult 方法。然后调用 WebService 的 getWeatherbyCityName 方法去取天气预报的数据。

getWeatherbyCityName 方法是下面 webservice 提供的方法：

http://www.webxml.com.cn/WebServices/WeatherWebService.asmx?

下面实现 WSGetWeatherResult 方法，用来读取得到的天气预报数据并进行处理。

```
private function WSGetWeatherResult(event:ResultEvent):void
{
    weatherWS.removeEventListener(ResultEvent.RESULT,WSGetWeatherResult);
    var arrC:ArrayCollection =event.result as ArrayCollection;
    if(arrC.length > 0)
     {
          var str:String = arrC.getItemAt(0).toString();
          var str2:String = arrC.getItemAt(1).toString();
          var vbox :VBox = new VBox();
          var vbox2 :VBox = new VBox();
          var hbox :HBox = new HBox();
          var canvas:Canvas = new Canvas();
          var path:String = "assets\\weather\\";
          var img1 :Image = new Image;
```

```
            var index1 :int = new int(arrC.getItemAt(8).toLocaleString());
            img1.load(picArray[index1]);
             hbox.addChild(img1);
             var img2 :Image = new Image;
             var index2 :int = new int(arrC.getItemAt(9).toLocaleString());
             img2.load(picArray[index2]);
             hbox.addChild(img2);
             var txtTem :Text = new Text();
             txtTem.text =    arrC.getItemAt(5).toString();
             var txtWea :Text = new Text();
             txtWea.text =    arrC.getItemAt(6).toString();
             var txtWind :Text = new Text();
             txtWind.text =    arrC.getItemAt(7).toString();
             vbox.addChild(txtTem);
             vbox.addChild(txtWea);
             vbox.addChild(txtWind);
             vbox2.addChild(hbox);
             vbox2.addChild(vbox);
             canvas.addChild(vbox2);
             myMap.infoWindow.content = canvas;
             var mapPnt2:MapPoint = new MapPoint(114.31,30.52);
             myMap.infoWindow.show(mapPnt2);
        }
}
```

在上面函数中把得到的天气预报数据进行解析，把对应的天气图片，气温等信息分类整理，使用 infoWindow 显示出来。

完整例子可以在 samples/3.1 下面找到。

3.2 ESRI Tilemap 四叉树索引研究

注：这里仅限于讨论下面地址提供的 tilemap 服务：

http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer。

因为 ESRI 提供了 cache map 功能 ，用户可以自己定义图片的大小和切割方式。

ESRI Tilemap 四叉树索引和 google map 的四叉树索引是有区别的，区别在于 google map 在第一次分幅的时候分为 4 片，但 esri tilemap 只分为 2 片，相当于在第三和第四象限没有图。只有（0,0）(0,1)的时候才有图，这就造成了 google map

和 ESRI Tilemap 融合的时候比较复杂，不能使用相同的分幅方法。

在下面的例子里，实现了根据显示级别（nzoom），以及经纬度取 ArcGISOnline 服务器上对应的图片。图片是 512*512 的。这个算法是用 ActionScript 来实现的，也可以用其他语言去实现。

算法的核心思想是通过经纬度算出其在 nzoom level 上所对应的图片的索引，然后拼成 url 去服务器端取数据。

url 的拼接规则为：

"http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer/tile/"+nZoom+"/"+yIndex+"/"+xIndex.

其中 nZoom 为显示级别，15>nZoom>=0，yIndex 为当前经纬度对应的图片的 Y 方向索引, xIndex 为当前经纬度对应的图片的 X 方向索引。

算法流程：

1 将 X 轴下移 90 个单位，保证第一次分幅只能分成 2 幅。

2 使用四叉树编码的方式对当前(X,Y)坐标解析。

3 对四叉树编码进行解析，得到当前坐标对应图片的索引值。

算法实现为：

```
private function GetTileXY(nZoom:int,mp:Point):Point
    {
        var wx:Number;
        var wy:Number;
        var cx:Number;
        var cy:Number;
        var xArray:Array = new Array();
        var yArray:Array = new Array();
        cx =   0;
        cy = -90;
        wx = wy = 180;
        var i:int = 0;
        var x:int = 0;
        var y:int = 0;
        for (i = 0; i <= nZoom; i++)
        {
            if (mp.x >= cx)
            {
                if (mp.y >= cy)
                {
```

```
                    xArray.push(1);
                    yArray.push(0);
                    cx += wx/2;
                    cy += wy/2;
                }
                else
                {
                    xArray.push(1);
                    yArray.push(1);
                    cx += wx/2;
                    cy -= wy/2;
                }
            }
            else
            {
                if (mp.y < cy)
                {
                    xArray.push(0);
                    yArray.push(1);
                    cx -= wx/2;
                    cy -= wy/2;
                }
                else
                {
                    xArray.push(0);
                    yArray.push(0);
                    cx -= wx/2;
                    cy += wy/2;
                }
            }
            wx = wx/2;
            wy = wy/2;
        }
        for(i = nZoom;i >=0;i--)
        {
            x = x+xArray[i]*Math.pow(2,nZoom-i);
            y = y+yArray[i]*Math.pow(2,nZoom-i);
        }
        var pnt :Point = new Point(x,y);
        return pnt;
    }
```

完整例子可以在 samples/3.2 目录下找到。

3.3 自定义控件的开发

ArcGIS API for Flex 本身只提供了 Navigation 和 ScaleBar 两个控件，如果需要添加自己的控件，则需要自己实现。下面以实现一个能在地图上动态显示当前鼠标经纬度的控件为例子，介绍如何实现自定义控件。

1 定义控件类，继承 UIComponent ，实现 IMapAware 接口：

```
public class ESRIStatusBar extends UIComponent implements IMapAware
{
    private var m_map:Map;
    private var m_stateLabel:TextField;
    public function ESRIStatusBar()
    {
     m_stateLabel = new TextField();
     m_stateLabel.width = 152;
    }
}
```

在 ESRIStatusBar 类里定义一个 TextField 来显示坐标，并在 ESRIStatusBar 类的构造函数里初始化。

2 重载 createChildren 函数：

```
override protected function createChildren() : void
{
     super.createChildren();
     var pnt:Point    = new Point;
     if(m_map.loaded)
      {
          var mapPnt:MapPoint = new MapPoint(m_map.extent.xmax,m_map.extent.ymin);
          pnt = m_map.toScreen(mapPnt);
          m_stateLabel.x = pnt.x-150;
          m_stateLabel.y = pnt.y-16;
     }
     addChild(m_stateLabel);
     return;
}
```

createChildren 函数用来把 stateLabel 添加到 DisplayList 中，用来在地图上显示。

3 实现 public function set map(map:Map) : void 函数：

```
public function set map(map:Map) : void
{
```

```
    m_map = map;
    m_map.addEventListener(MouseEvent.MOUSE_MOVE,mouseMoveHandler);
    m_map.addEventListener(ExtentEvent.EXTENT_CHANGE, extentChangeHandler);
  }
```

set map(map:Map)用来把当前地图对象传递给 ESRIStatusBar 控件，使控件能够响应 map 色事件，操纵 map 上的对象。

4 添加自己的功能代码;

首先，添加对 map 的 MOUSE_MOVE 事件的响应

```
m_map.addEventListener(MouseEvent.MOUSE_MOVE,mouseMoveHandler);
```

其次，在 mouseMoveHandler 中动态取得鼠标位置转换成经纬度坐标显示在 TextField 上。

```
private function mouseMoveHandler(event:MouseEvent):void
    {
        if(m_map)
        {
            if(m_map.loaded)
            {
                var mapPoint : MapPoint = m_map.toMapFromStage(event.stageX,
                event.stageY);
                  m_stateLabel.text =    "x="+mapPoint.x.toString()+" ,
                y="+mapPoint.y.toString();
              }
          }
     }
```

最后，将这个 ESRIStatusBar 加入到 map 上:

```
var statusBar:ESRIStatusBar = new ESRIStatusBar();
    IMapAware(statusBar).map = EsriMap;
    EsriMap.addChild(statusBar);
```

完整例子可以在 samples/3.3 下面找到。

3.4 与 Google Map 的融合

（注意，该例子使用的是 google map flex api v1.6 现在 google 网站上提供下载的 api 是 v1.7a，使用 google map flex api v1.7a 将无法显示地图和控件）。

本程序的目的就是使用 Flex + google map 的基础数据+ESRI 强大的分析功能给用户提供一个良好用户体验的 GIS 大众应用。

该程序的思路如下：

首先是绘制一个点，然后调用 ArcGIS API for Flex 提供的 Geometry Service 做一个缓冲区（buffer），再通过调用 ArcGIS API for Flex 提供的 GP 服务根据得到的 buffer 去查询在 buffer 内的人口数，最后通过 google map 的 InfoWindow 显示出来。

其中 ArcGIS API for Flex 中 Geometry service，GP 的调用前面已经讲过，下面主要讲一下 gmap 数据和 ESRI 数据的转换，主要逻辑在 GMapAddOverlay()函数中，如下面代码：

```
private function GMapAddOverlay(geometry:com.esri.ags.geometry.Geometry ):void
{
    switch( geometry.type )
    {
        case Geometry.MAPPOINT:
        {
            var pnt :MapPoint = geometry as MapPoint ;
            var markerA:Marker = new Marker(new LatLng(pnt.y, pnt.x),new
MarkerOptions({strokeStyle: new StrokeStyle({color: 0x987654}),fillStyle: new FillStyle({color:
0x223344, alpha: 0.8}),radius: 12,hasShadow: true}));
                gMap.addOverlay(markerA);
            }
            break;
        case Geometry.POLYLINE :
                var ii:int = 0;
                var jj:int = 0;
                var geometryPolyline: com.esri.ags.geometry.Polyline =
 geometry as com.esri.ags.geometry.Polyline;
                var pnt1Array:Array = new Array();
                for(ii = 0; ii <geometryPolyline.paths.length;ii++)
                {
                    for(jj = 0; jj <geometryPolyline.paths[ii].length; jj++)
                    {               pnt1Array.push(new
LatLng(geometryPolyline.getPoint(ii,jj).y,geometryPolyline.getPoint(ii,jj).x));
                    }               }
                var polyline:com.google.maps.overlays.Polyline = new
com.google.maps.overlays.Polyline(pnt1Array, new PolylineOptions({strokeStyle: new
StrokeStyle({color: 0xFF0000,thickness: 4,alpha: 0.7})}));
                gMap.addOverlay(polyline);
                break;
            case Geometry.POLYGON :
                    var i:int = 0;
                    var j:int = 0;
                    var geometryPolygon : com.esri.ags.geometry.Polygon = geometry as
```

```
com.esri.ags.geometry.Polygon;
                        var pntArray:Array = new Array();
                        for(i = 0; i <geometryPolygon.rings.length;i++)
                         {
                    for(j = 0; j <geometryPolygon.rings[i].length; j++)
                                {
                                pntArray.push(new
LatLng(geometryPolygon.getPoint(i,j).y,geometryPolygon.getPoint(i,j).x));
                                }
                          }
                        var polygon:com.google.maps.overlays.Polygon = new
com.google.maps.overlays.Polygon(pntArray,new PolygonOptions({ strokeStyle: new
StrokeStyle({color: 0x223355,thickness: 4,alpha: 0.7}), fillStyle: new FillStyle({color:
0x223355,alpha: 0.7})}));
                        gMap.addOverlay(polygon);
                        break;
            }
}
```

ArcGIS API for Flex 和 google map 中几何类型的对应关系如下：

ArcGIS API for Flex	google map
MapPoint	Marker
Polyline	Polyline
Polygon	Polygon

在 GMapAddOverlay()函数中对上面 3 种几何类型进行转换，将 ArcGIS API for Flex 中的几何数据转换成 google map 中的几何数据并调用 addOverlay 函数添加到地图上。

完整例子可以在 samples/3.4 目录下找到。