ArcGIS RIA 开发实践

ArcGIS Server 9.3 | ArcGIS API for Flex 1.2 | Flex SDK 3.2

牛魔王

Email: warrenwyf@gmail.com

2009-5-26

目录

ArcGIS RIA 开发实践【背景篇】 4

I RIA 介绍 4

一 RIA 相比传统 Web 应用的优势 4

二 RIA 的局限性 6

三 目前主流 RIA 技术及特点 7

1 Adobe Flex/Flash 7

2 Microsoft Silverlight 7

3 Sun JavaFX 7

4 Curl 8

II ArcGIS Server 的 REST 接口 9

一 REST 的概念 9

二 通过 REST 的接口使用 ArcGIS 功能 10

ArcGIS RIA 开发实践【Flex 篇】 14

I、 ArcGIS Flex API 基础 14

一 Flex 的历史和现状 14

二 ArcGIS Flex API 概述 15

三 一些基本概念 16

1 MapServiceLayer 16

2 GraphicLayer 与 Graphic 16

3 Geometry 17

4 Symbol.....17

四 地图功能组件.....20

1 Map 组件.....20

2 Navigation 工具.....24

3 Draw 工具.....27

4 InfoWindow 窗口.....29

五 常用功能.....32

1 QueryTask.....32

2 FindTask.....34

3 IdentifyTask.....36

4 GeometryService.....38

5 Geoprocessor.....40

6 RouteTask.....43

7 Locator.....45

六 **ArcGIS Server 中配置安全策略文件**.....48

II、 ArcGIS Flex API 高级.....49

一 控制 Map 与 Layer.....49

1 Map 中地图服务图层的控制.....49

2 地图服务图层中子图层的控制.....51

3 子图层中内容的控制.....53

4 动态投影.....55

二　业务信息的表达 57

1　分类渲染 57

2　专题图 61

3　实时的数据变化 63

4　符号扩展 65

三　与其它系统的整合 68

1　与 JavaScript 交互 68

2　HTTPService 与 WebService 70

3　RemoteObject 与 BlazeDS 72

4　通过 BlazeDS 调用 AO 76

III、　综合场景演示 80

1　某系统演示 80

2　其它一些案例 83

ArcGIS RIA 开发实践【背景篇】

I RIA 介绍

一 RIA 相比传统 Web 应用的优势

RIA 作为一个互联网领域越来越火的术语，究其根本在于它相对传统 Web 应用的诸多优势。RIA 作为“富互联网应用”的代名词（Rich Internet Application），从名字上就已经可以看出它最大的特点：Rich，这个 Rich 是多方面的，包括丰富的表现力、富有交互、内容丰富、基于富客户端引擎等等。同时，RIA 是 Internet 的，它在功能上希望和传统的 Web 应用一样通用、标准，因此，这也是为什么眼下很多 2D/3D 效果俱佳、操作人性化的网络游戏不能称为 RIA 的原因。

传统 Web 应用的很多不足都来源于 HTML 的静态性，虽然由于 JavaScript 脚本的存在可以在浏览器中实现一些客户端交互和效果；同时，Web2.0 浪潮推广的 AJAX 技术也使浏览器和服务器的通讯更及时、页面表现也更友好。但是，传统 Web 的根基使这些进步都是局限的，在声音、视频、Socket 通讯、本地存储这些方面它都捉襟见肘。

RIA 则在某种程度上颠覆了传统 Web 应用的理念和设计。为了能提供客户端更优秀的表现和交互能力，RIA 一般会在客户端添加一个引擎，这个引擎和 Java 虚拟机的作用很相似， RIA 则会在这个引擎中运行所有编译过的脚本及相关的资源。通过这个引擎的中介作用，RIA 就把传统 Web 应用中很多在服务器上执行的功能（比如输出页面）转移到了客户端。一方面减轻了服务器的压力，另一方面也提高了客户端的处理和交互能力。

通常，RIA 一般会提供比较出色的表现能力，这也是 RIA 之所以被称为“Rich”的最重要原因。传统的 Web 页面即使通过 JavaScript 脚本的帮助也很难达到 RIA 能做到的页面效果和功能，特别是 GIS 领域，在用户有复杂交互、地图无缝平滑缩放等需求的时候，RIA 的这个优势尤其突出。由于 RIA 在表现方面的特色，用户在 RIA 与传统 Web 站点之间能感到明显的体验上的差距，这是 RIA 受欢迎的重要因素之一。

另外值得一提的是 RIA 的标准性。传统 Web 应用在面向不同的的操作系统和浏览器的时候，呈现出的页面可能千差万别；RIA 则不同，由于操作系统或浏览器中都会有一个客户端引擎来运行我们的 RIA，因此在各种支持的操作系统和浏览器中都没有差别，也就是说，你设计好的 RIA 在各种环境下表现地都会很标准。

总的来说，RIA 相比传统 Web 应用的优势主要集中在以下几点：

1. 表现力丰富
2. 网络效率高
3. 交互能力强
4. 面向操作系统和浏览器透明
5. 沙箱环境提供更可靠的安全性
6. 易于与现有系统集成

二 RIA 的局限性

当然，RIA 也并不是没有缺陷，它也有它的局限性。RIA 的局限性主要在两方面：其一，是客户端引擎的下载和安装；其二，是目前难于做 SEO。

事实上 RIA 的概念和若干年前的 Java Applet 很相似，但是 Java Applet 已经基本消亡，如今的 RIA 却突然火爆，这和 RIA 的缺陷在目前得到某种程度的补偿有关系。

首先，客户端引擎的下载和安装。比如 Flex，由于 Flash 在互联网的应用极其普遍，其赖以生存的 Flash Player 在各种系统的占有率在 97%左右；Silverlight 则估计会随着下一版本操作系统的推广成为将来 Windows 用户的标准配置，因此，客户端引擎对它们来说都不是很大的障碍。

而对于搜索引擎优化，一方面，只有面向公众的网站对这方面要求比较高，其它业务型的企业级应用则并不很关心，即使关心也可以把一些无需被搜索的内容放到 RIA 中（比如地图），而在 Web 页面中保留需要被搜索的内容；另一方面，Google 等搜索引擎也正在和 Adobe 等公司进行深入合作，将 RIA 中内容的索引提上了日程，可以看到，RIA 的迅猛发展使搜索引擎也开始关注这块重要领地了。

三 目前主流 RIA 技术及特点

1 Adobe Flex/Flash

Flex 无疑是目前最为流行的 RIA 技术，这主要得益于 Flash 的风行。从 Web 出现不久的时候，网页制作工具就有“三剑客”的称谓，其中一个就是 Flash。

Flash 是为动画美工准备的工具，理论上讲用 Flash 也可以做出 Flex 能做的东西；但是为了拉拢众多的程序员参与 RIA 开发，在 Flash 的基础上出现了 Flex，Flex 工具最终出厂的产品也是 Flash（.swf），但是整个开发流程则完全是程序开发模式的。

Flex 的开发平台有 Adobe 的 Flex Builder（基于 Eclipse，$300），开源的 FlashDevelop 等。

2 Microsoft Silverlight

Silverlight 是 Microsoft 在 RIA 领域的重大动作，很多人将其视为 Flex 的主要竞争对手，它吸取了很多 Flex 的优点，同时又和 Visual Studio 等工具进行了完美的结合，因此对于习惯 Microsoft 产品线的开发人员来说，Silverlight 的开发环境和理念都是比较容易接受的。

Silverlight 目前的版本是 Silverlight 3 Beta。

3 Sun JavaFX

JavaFX 是 Sun 公司推出的 RIA 平台，2008 年底发布了第一个正式版本，目的和 Silverlight 一样——争夺目前被 Flex/Flash 牢牢占据的 RIA 市场。但目前为止，JavaFX

还没有表现出可以与 Flex/Flash 甚至 Silverlight 一较高下的态势。另外，Sun 被 Oracle 收购以后，这些产品线将何去何从也是一个未知数。

4 Curl

Curl 是一个试图在企业级 RIA 领域与 Flex/Flash 一较高下的平台，它是 Curl 公司的主打产品。Curl 在部署的时候需要一个许可，对于商业应用来说，Curl 要求的商业许可相当昂贵。另外，Curl 的运行环境目前还没有普及，对于没有把握让用户先耐心把 Curl RTE（运行环境）下载并安装上的应用，使用 Curl 一定要谨慎。

II ArcGIS Server 的 REST 接口

ArcGIS 在 RIA 领域目前有 Flex 和 Silverlight 两种 API ,这两种 API 的工作基础都是 ArcGIS Server 的 REST 接口(REST SDK)。因此 ,下面我们对 REST 和 ArcGIS Server 的 REST 接口做些介绍。

一 REST 的概念

REST（Representational State Transfer）是 Roy Fielding 博士在 2000 年他的博士论文中提出来的一种软件架构风格。REST 本身并不涉及任何新的技术，它基于 HTTP 协议，比起 SOAP 和 XML-RPC 来说它更加地简洁、高效，越来越多的大型网站正在使用 REST 风格来设计和实现。

REST 最突出的特点就是用 URI 来描述互联网上所有的资源，Roy Fielding 博士通过观察互联网的运作方式对其进行了抽象，他认为：设计良好的网络应用表现为一系列的虚拟“网页”，或者说这些虚拟网页就是资源状态的表现（Representational）；用户选择这些链接导致下一个虚拟的“网页”传输到用户端展现给使用的人，而这正代表了资源状态的转变（State Transfer）。

REST 主要有以下的特点：

- 资源通过 URI 来指定和操作
- 对资源的操作包括获取、创建、修改和删除资源，这些操作正好对应 HTTP 协议提供的 GET、POST、PUT 和 DELETE 方法
- 连接是无状态性的
- 能够利用 Cache 机制来提高性能

二 通过 REST 的接口使用 ArcGIS 功能

ArcGIS Server 原生地支持 REST ,也就是说在安装完 ArcGIS Server 后 REST 接口已经存在。

使用 REST 接口最重要的就是 REST 服务目录 ,它是将 ArcGIS Server 提供的 REST 资源通过 HTML 表现出来的 URL。Java 版本的在浏览器中访问 http://<host>:8399/argis/rest ;.Net 版本访问 http://<host>/arcgis/rest 就可以看到当前所有的服务及其这些服务超链接到的资源和操作。

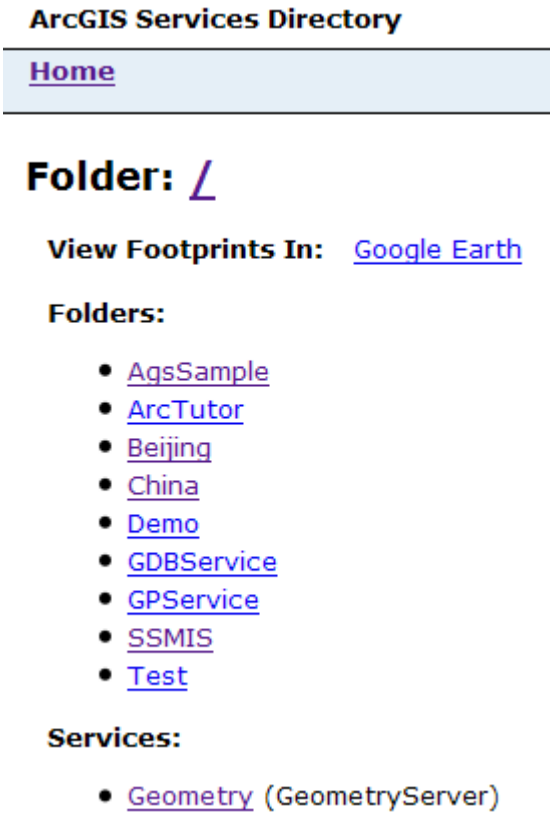


图 1 ArcGIS Server REST 服务目录

在这里顺便提一下使用 REST 接口的一个特别需要注意的地方，或许你有时新发布了一个服务 ,然后随即进入 REST 服务目录查看 ,但是却看不到这个服务。这是由于 REST

的缓存机制造成的，你需要去清空一下旧的缓存，方法就是进入 REST 管理界面 http://<host>[:8399]/arcgis/rest/admin,然后使用 ArcGIS Server 管理员帐号登录:

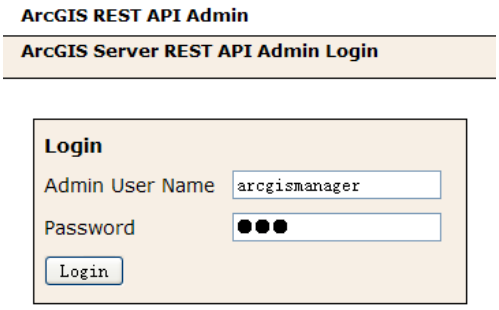


图 2　　ArcGIS Server REST 管理界面登录

登录后，在“Clear Cache Options”链接中可以对旧的 REST 缓存进行清除：

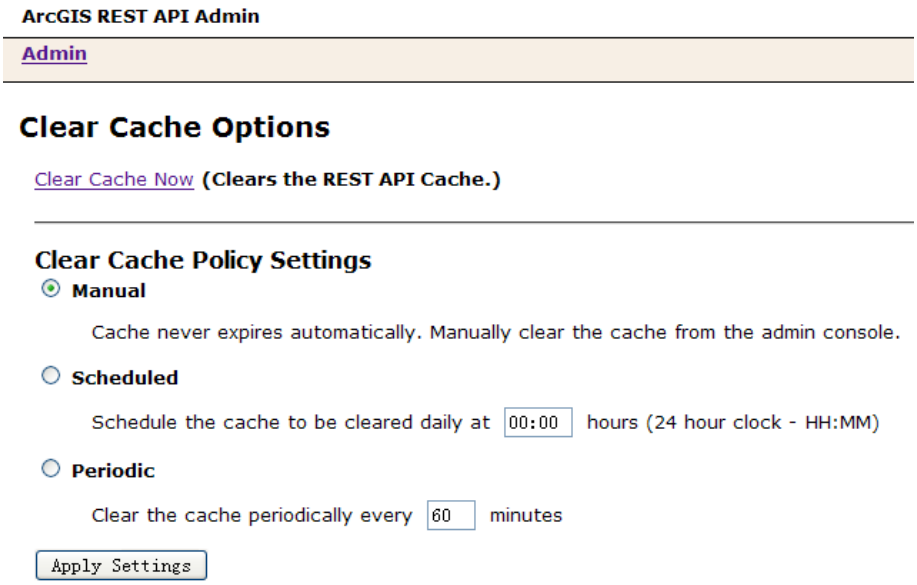


图 3　　清除 ArcGIS Server REST 缓存

下面让我们看一下我在 ArcGIS Server for Java 服务器的“AgsSample”目录下发布的一个名为“World”的世界地图服务，在 REST 服务目录中，通过链接最终进入这个服务，这个链接应该是类似这样的格式（注意 URL 中的“rest”）：http://localhost:8399/arcgis/rest/services/AgsSample/World/MapServer。下面我们可以看到关于这个服务的信息和相关的操作：



图 4　　ArcGIS Server 地图服务的 REST 信息

下面就让我们点击“Export Map”链接去使用地图服务的“export”操作。你会发现，这个操作的 URL 是这样的：

http://localhost:8399/arcgis/rest/services/AgsSample/World/MapServer/export?bbox=-198.000000454545,-99.0000004545455,198.000009545455,99.0000095454545。在浏览器中访问这个 URL 的结果如下：

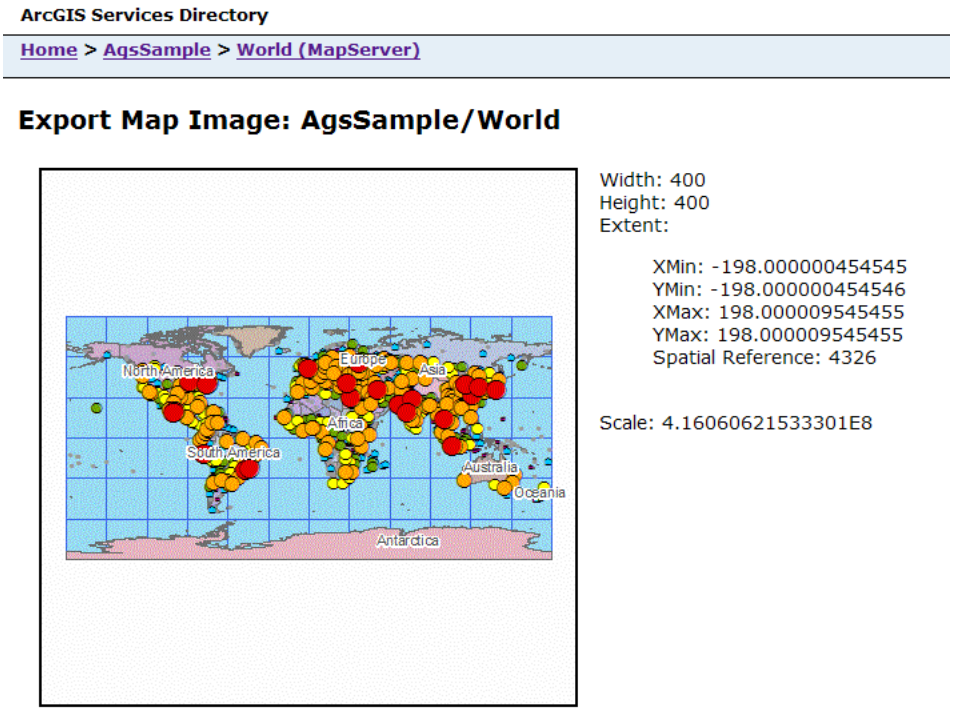


图 5　　ArcGIS Server 地图服务 REST 接口中的 export 操作

从这里你也可以总结出来：在 ArcGIS Server REST 接口中，请求服务的 URL 一般应该是这样的：http://<host>[:<port>]/arcgis/rest/services/<服务>/<服务类型>/<操作>?<参数>。关于 REST 接口支持的资源类型和具体操作参数，你可以查阅 ArcGIS Server 的 REST SDK 文档，在这里就不赘述了。

ArcGIS RIA 开发实践【Flex 篇】

I、 ArcGIS Flex API 基础

一 Flex 的历史和现状

Flex 的前身是 Flash，Flash 是极为流行的互联网矢量动画解决方案，目前据统计 97%的浏览器都安装有 Flash Player 插件用以播放 swf 动画，其中未安装的 3%还有很多是因为某些场合的安全限制导致的，可以说 Flash 是极为普及的 RIA 平台。

但是由于 Flash 是一个动画制作软件，其中有很多比如时间轴、影片剪辑等程序员不容易理解的概念，因此 Macromedia 公司推出了 Flex。Flex 抛弃了许多动画设计术语，转而使用程序员喜欢的方式开发 RIA 应用，并且 Flex 能编译生成可以在 Flash Player 中运行的 swf 文件，这无疑吸引了大量程序员，并且实现了和 Flash 平台的无缝拼接，从而利用 Flash 平台多年积累的大量素材、美工和设计者。

Adobe 公司在 2005 年收购了 Macromedia 公司，并在第一时间将 Flex/Flash 冠以 Adobe 的商标推向市场，可见其对 RIA 市场和 Flex/Flash 的重视。

目前，可以说 Adobe Flex/Flash 是最流行且最成熟的 RIA 平台。

二 ArcGIS Flex API 概述

ArcGIS Flex API 是我自己使用的 ArcGIS API for Flex 的简称，在本文中将一直使用这个名称。

ArcGIS Flex API 是 ArcGIS 在 RIA 领域的第一个产品，因此选择了最为成熟的 Flex/Flash 平台。使用 ArcGIS Flex API 可以开发运行于浏览器中的 Web 应用或者运行于桌面的 AIR 应用 ,它基于 ArcGIS Server 的 REST 接口 ,所有的功能都可以在 REST SDK 中找到影子。

使用 ArcGIS Flex API 开发的 Flex 应用可以非常便捷地使用地图功能和 REST 接口提供的 GIS 查询及分析功能；同时，ArcGIS Flex API 专注于 GIS 功能的实现而不重复地创造组件，因此可以让你的业务和 GIS 方便地结合起来。

具体关于 ArcGIS Flex API 可以实现哪些功能，我们在下面的章节中将进行详细的讲解。

三 一些基本概念

首先，让我们先来了解一些 ArcGIS Flex API 中的基本概念。事实上你会发现这里的概念和桌面或 ADF 开发都有相通之处，不过为了强调一些重点，顺便介绍些 ArcGIS Flex API 中的特殊之处，这里还是先对这些内容做点介绍。

1 MapServiceLayer

MapServiceLayer 对应的是 ArcGIS 发布的地图服务，它们是一对一的关系，每个地图服务在 ArcGIS Flex API 中都会以一个图层的形式出现。不同种类的地图服务将会对应不同的 MapServiceLayer ,ArcGIS Flex API 中主要有以下几种 MapServiceLayer：

ArcGISDynamicMapServiceLayer	ArcGIS Server 发布的动态地图服务
ArcGISTiledMapServiceLayer	ArcGIS Server 发布的切片地图服务
ArcGISImageServiceLayer	ArcGIS Image Server 发布的影像服务
ArcIMSMapServiceLayer	ArcIMS 发布的地图服务

2 GraphicLayer 与 Graphic

GraphicLayer 并不对应到服务器端的某个地图服务，它完全是客户端的图层。因此，GraphicLayer 在客户端数据表达方面有非常重要的作用，它可以根据各种情况动态地在客户端显示一些符号化的几何对象——Graphic。

ArcGIS Flex API 中的 Graphic 是一个继承于 UIComponent 的类，因此它的表现和 Canvas、Button 等普通的 Flex 组件一样，在地图中 Graphic 就是一个可视化的、可响应鼠标事件的客户端要素。

Graphic 有 3 个很重要的属性 :geometry、symbol 和 attributes。其中 geometry

属性定义的是一个几何对象，它是 Graphic 的基础，因为 Graphic 要表达的就是这个几何对象；symbol 属性则定义了这个几何对象通过什么符号表现出来；attributes 则可以存放一系列与空间信息无关的属性数据。从上面可以发现，Graphic 可以被理解成是一个在客户端符号化的要素。关于 Graphic 的 geometry 和 symbol 属性所对应的 Geometry 类和 Symbol 类，接下来在这里做一点更详细的说明。

3 Geometry

Geometry 是 GIS 中很基本的概念，ArcGIS Flex API 中定义了如下几何对象：

MapPoint	点
Multipoint	多点
Polyline	多段线
Polygon	多边形
Extent	边界范围

我们可以发现 ArcGIS Flex API 主要定义了点、线、面这些简单对象，事实上这和 REST SDK 中的几何对象是一一对应的，当 ArcGIS Flex API 和服务器交互的时候，这些几何对象都会被转化成字符串形式的 JSON 格式发送到服务器。

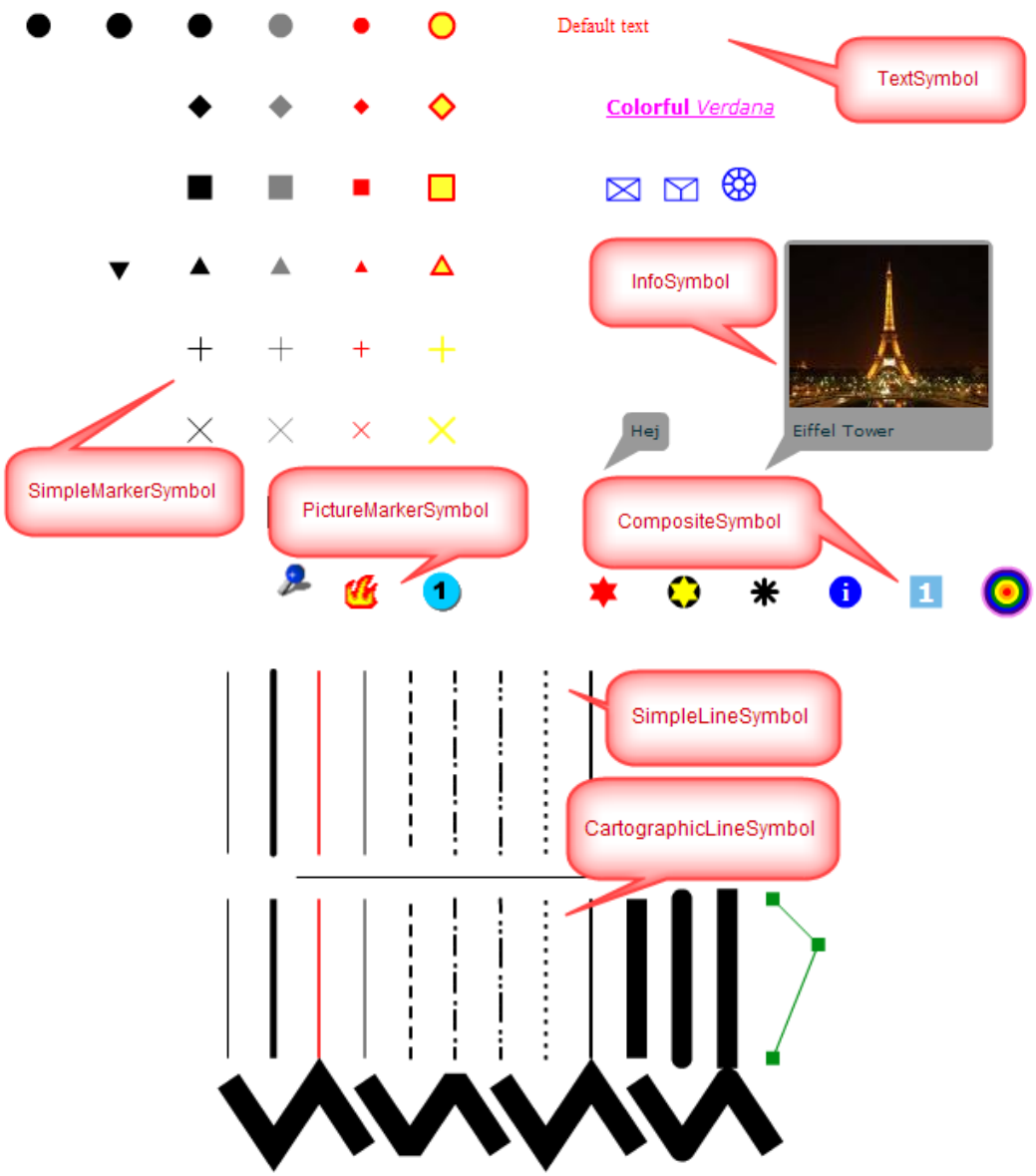
4 Symbol

Symbol 定义了 Geometry 使用什么符号表现出来。在 ArcGIS Flex API 中，一般使用 MarkerSymbol 定义点的符号、LineSymbol 定义线的符号、FillSymbol 定义填充的符号，另外还有 TextSymbol、InfoSymbol 等其它符号帮助进行其它信息的辅助表达。下面列出的是 ArcGIS Flex API 中提供的常用的 Symbol：

SimpleMarkerSymbol	简单点符号
SimpleLineSymbol	简单线符号
SimpleFillSymbol	简单填充符号

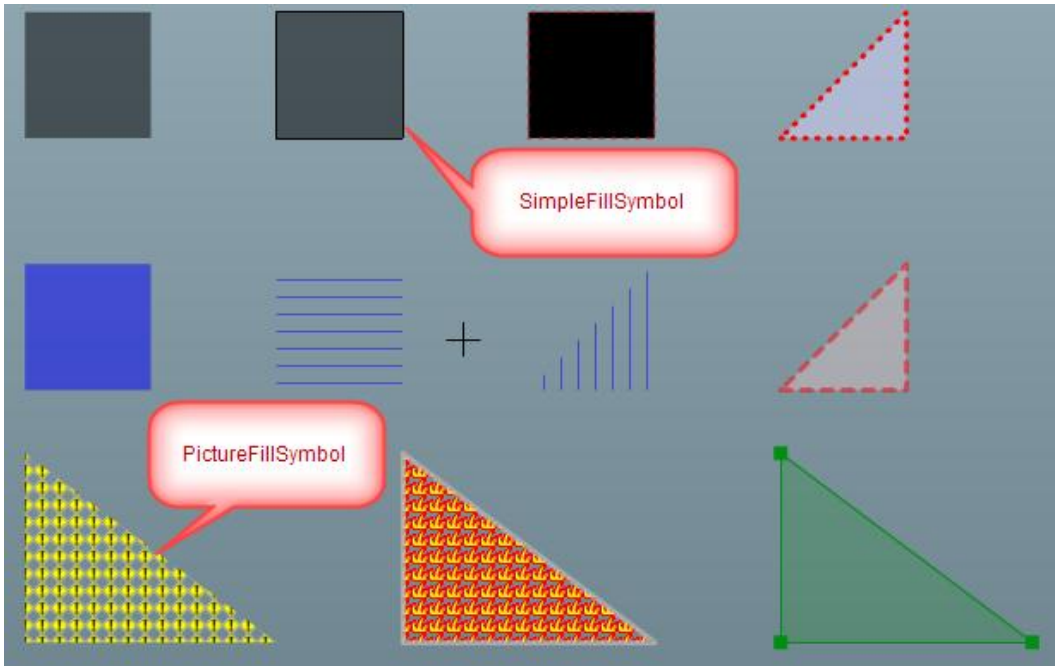
CartographicLineSymbol	制图线符号，可定义端点、折点样式
PictureMarkerSymbol	图片[1]点符号
PictureFillSymbol	图片填充符号
TextSymbol	文字符号
InfoSymbol	信息提示符号
CompositeSymbol	复合符号[2]

下面是上面这些符号的预览：



[1] 支持 JPG、GIF、PNG、SWF、SVG 格式

[2] 上面若干种符号的组合


SimpleFillSymbol
PictureFillSymbol

四 地图功能组件

在这小节中，我们主要介绍 ArcGIS Flex API 中对于一般地图功能的实现和使用。在这里让我们暂时先抛开“GIS”，去熟悉一下“地图”，毕竟地图是 GIS 可视化和功能交互的基石。

1 Map 组件

Map 是 ArcGIS Flex API 中唯一的可视化组件，它可以直接被放置到 Flex 应用中进行布局、展现。

Map 可以被看作是一个地图， ArcGIS Flex API 中所有的操作和功能都会在 Map 中进行，因此 Map 是一个特别重要的对象。如果在应用中有别的业务组件或模块需要和地图交互（比如把业务数据标到图上），那么只要它们可以获得 Map 对象的引用就可以了（比如一种做法就是在它们的构造函数中设置），所有和 GIS 相关的功能都可以通过 Map 来实现。

Map 中包含各种 Layer，这在 GIS 中非常容易理解——地图中需要加入若干个图层。在 ArcGIS Flex API 中，Map 中的 Layer 有两种，一种是对应某个地图服务的 MapServiceLayer、一种是用以绘制自定义要素的 GraphicLayer，这里我们先看一下如何在 Map 中加载某个地图服务（MapServiceLayer）及其效果。

```
<esri:Map id="map">
    <esri:ArcGISTiledMapServiceLayer
        url="http://localhost:8399/arcgis/rest/services/China/map/MapServer"/>
</esri:Map>
```



图 6　　Map 和 MapServiceLayer

默认的 Map 中还有几个其它的元素，包括控制地图缩放的 zoomSlider、地图比例尺 scaleBar 以及 ESRI 的 logo。如果你不需要这些元素，都可以通过 Map 中对应的属性对它们进行隐藏，下面我们看一下代码和隐藏后的效果：

```
<esri:Map id="map" zoomSliderVisible="false" scaleBarVisible="false" logoVisible="false">
      <esri:ArcGISTiledMapServiceLayer
         url="http://localhost:8399/arcgis/rest/services/China/map/MapServer"/>
</esri:Map>
```



图 7　　隐藏不需要的元素后的 Map

需要顺便提一下的是，ESRI 的 logo 能被隐藏掉的前提条件是你使用的是本地的 ArcGIS Server 服务。另外，zoomSlider 和 scaleBar 除了可以被隐藏掉，还可以通过定制样式来改变外观，或许你仅仅是讨厌默认的外观但却需要这个功能。

以下的 CSS 代码将 Map 的 zoomSlider 移到了右边，并且对它按钮的样式、滑块位置等一系列外观做了定制，接着我们就可以看到新的 zoomSlider 的样子了：

```
<mx:Style>
.mapStyle{ navigation-style-name: navigationStyle;}
.navigationStyle{
    top: 6;
    right: 6;
    navigation-minus-button-style-name: navigationMinusButton;
    navigation-plus-button-style-name: navigationPlusButton;
    navigation-slider-style-name: navigationSlider;
}
.navigationMinusButton {
    disabledSkin: Embed(source="assets/zoomout.png");
    downSkin: Embed(source="assets/zoomout.png");
    overSkin: Embed(source="assets/zoomout.png");
    upSkin: Embed(source="assets/zoomout.png");
}
.navigationPlusButton{
    disabledSkin: Embed(source="assets/zoomin.png");
    downSkin: Embed(source="assets/zoomin.png");
    overSkin: Embed(source="assets/zoomin.png");
    upSkin: Embed(source="assets/zoomin.png");
}
.navigationSlider{
    dataTipPlacement: "left";
    tickColor: #FFFFFF;
    tickLength: 10;
    tickOffset: 13;
    tickThickness: 3;
}
</mx:Style>

<esri:Map id="map" styleName="mapStyle"></esri:Map>
```



图 8　　改变 zoomSlider 样式后的 Map

另外，Map 还有一些默认隐藏的外观设置，比如地图四周的导航按钮、地图中心的十字符号，包括拉框缩放的方框样式等都可以进行设置，下面是修改了一些样式的 Map：



图 9　　更改一些隐藏外观样式后的 Map

从上面我们可以看到，Map 有很灵活的可配置性，它的各种元素的外观也很容易定制；当然，如果实在有特殊的需要，Map 也可以通过隐藏所有元素来提供给你一个非常干净的地图组件，你则可以在此基础上加上自己的内容。

在有了一个令自己满意的 Map 的基础上，你就可以使用 ArcGIS Flex API 中很多激动人心的功能了——同时你会慢慢发现，所有的功能都是紧紧围绕着这个 Map 的。

2 Navigation 工具

当准备好一个满意的 Map 的时候，我们已经可以在地图中看到我们在 ArcGIS Server 中发布的服务了。在这个时候，我们可能马上想到的是如何去操作地图，比如缩放、漫游、前进、后退等等地图导航功能。在这个时候你或许已经发现，当前地图默认的操作是漫游（Pan）；同时，你滚动鼠标滚轮的时候地图会进行缩放；如果按住 shift 键并在地图上拖动时会出现一个矩形框，松开鼠标后地图会缩放到框中的范围。似乎 Map 已经可以进行地图导航了。

然而，当你想要切换地图到拉框缩小状态或者别的什么操作，或许就感到有点无所适从了。这时，我们友好的 Navigation 工具就正式登场了。

Navigation 提供了对 Map 进行导航的诸多功能，通过它可以实现漫游、拉框放大、拉框缩小等类似 ADF 中 Tool 的功能和前一视图、后一视图、全图等类似 ADF 中 Command 的功能。但是，它并不是一个工具条，而是提供了一些功能，你可以自定义工具条和一些按钮然后绑定到 Navigation 的功能上。

首先让我们看一下 Navigation 中漫游、拉框放大、拉框缩小等功能和工具条按钮的绑定。这些类似 Tool 的功能有个特点就是同时只有一个按钮需要被激活，它指示了当前 Map 的操作状态，因此我们可以借助 Flex 中的 ToggleButtonBar 作为这些功能的宿主。下面的代码定义了一个 ToggleButtonBar 并绑定了 Navigation 的上面几个功能：

```
<mx:Script>
    <![CDATA[
        private function itemClickHandler(event:ItemClickEvent):void
        {
            switch (event.index)
            {
                case 0:{ nav.activate(Navigation.PAN , true);break; }
                case 1: {nav.activate(Navigation.ZOOM_IN , true); break; }
                case 2: {nav.activate(Navigation.ZOOM_OUT , true); break; }
            }
        }
    ]]>
</mx:Script>

<esri:Navigation id="nav" map="{map}"/>

<esri:Map id="map">
    <esri:ArcGISTiledMapServiceLayer
    url="http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_Imagery_World_2D/MapS
    erver"/>
</esri:Map>

<mx:HBox horizontalAlign="center" width="100%">
    <mx:ToggleButtonBar id="tbb" itemClick="itemClickHandler(event)">
        <mx:dataProvider>
            <mx:Array>
                <mx:String>漫游</mx:String>
                <mx:String>拉框放大</mx:String>
                <mx:String>拉框缩小</mx:String>
            </mx:Array>
        </mx:dataProvider>
    </mx:ToggleButtonBar>
</mx:HBox>
```

当我们点击 ToggleButtonBar 中某个按钮时 ,itemClickHandler 这个监听被触发，于是 Navigation 就会根据当前 ToggleButtonBar 所选中的按钮来设置 Map 的操作状态——调用 Navigation 的 activate 方法。而 Navigation 之所以可以和 Map 联动，是因为 Navigation 有个 map 属性会存放 Map 的引用。

下面是添加了 ToggleButtonBar 的效果：

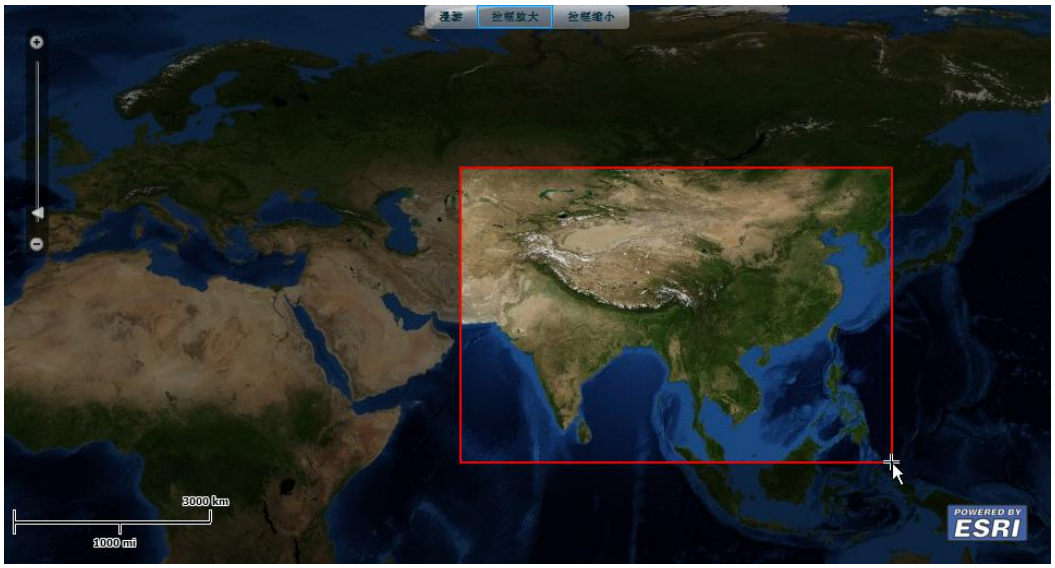


图 10　　绑定了漫游、拉框放大、拉框缩小等功能的 ToggleButtonBar

下面再让我们看一下 Navigation 中前一视图、后一视图、全图等功能和工具条按钮的绑定。这些类似 Command 的按钮每个都会实现单一的功能，不过，需要注意的是，前一视图、后一视图的按钮并不是一直可用，比如如果用户从来没有点击过前一视图按钮，那么后一视图按钮肯定是不可用的对吧？这些细节 Navigation 也已经为你考虑到了，下面我们看一下如何定义这些按钮并绑定到 Navigation 的功能上：

```
<mx:Button label="前一视图"
    click="nav.zoomToPrevExtent()"
    enabled="{!nav.isFirstExtent}"/>
<mx:Button label="后一视图"
    click="nav.zoomToNextExtent()"
    enabled="{!nav.isLastExtent}"/>
<mx:Button label="全图"
    click="nav.zoomToFullExtent()"/>
```

从这里可以看到，Navigation 不仅可以改变 Map 的操作状态，还有直接操作 Map 的方法，通过对这些方法的调用，Navigation 就可以是 Map 根据你的需要切换视图。下面是添加了这几个功能按钮的效果：

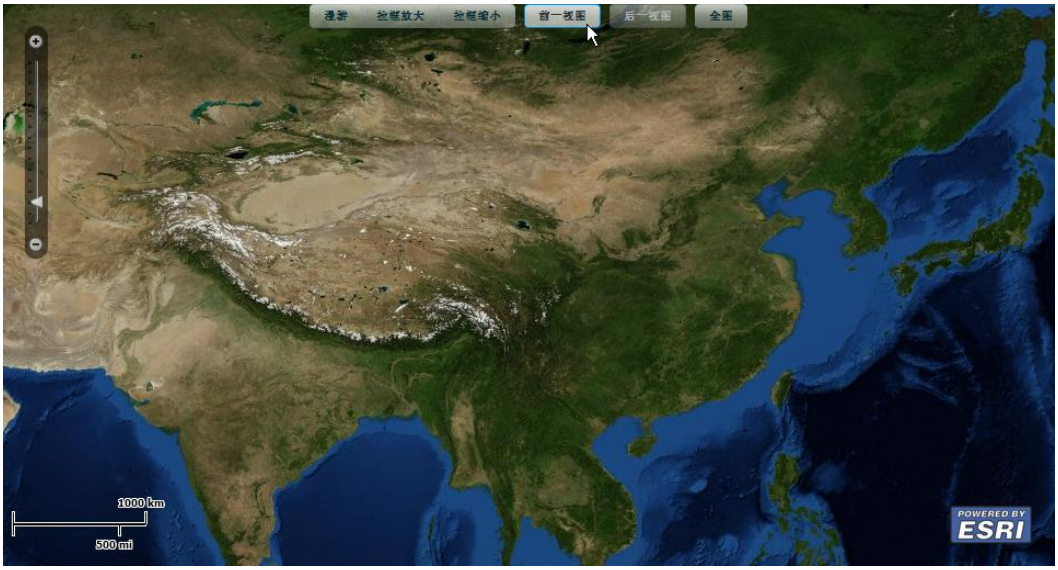


图 11　　绑定了前一视图、后一视图、全图等功能的按钮

到现在为止，我们已经知道怎么样使用 Map 来展现服务器端的地图服务，然后使用 Navigation 去导航地图，所以对地图的浏览已经基本可以说无所不能了。下面，或许你已经开始更多地考虑用户怎么和地图进行更多的交互了，比如客户端绘制。

3　Draw 工具

Draw 是 ArcGIS Flex API 另外一个和 Navigation 性质类似的工具。Draw 工具提供了在客户端绘制各种几何对象的功能，类似 Navigation 的使用那样，我们用一个 ToggleButtonBar 去绑定 Draw 工具的功能：

```
<mx:Script>
    <![CDATA[
        private function itemClickHandler(event:ItemClickEvent):void
        {
            switch (event.index)
            {
                case 0:{draw.activate(Draw.MAPPOINT);break;}
                case 1:{draw.activate(Draw.MULTIPOINT);break;}
                case 2:{draw.activate(Draw.LINE);break;}
                case 3:{draw.activate(Draw.POLYLINE);break;}
                case 4:{draw.activate(Draw.FREEHAND_POLYLINE);break;}
```

```
                    case 5:{draw.activate(Draw.POLYGON);break;}
                    case 6:{draw.activate(Draw.FREEHAND_POLYGON);break;}
                    case 7:{draw.activate(Draw.EXTENT);break;}
                }
            }
    ]]>
</mx:Script>

<esri:Draw id="draw" map="{map}" graphicsLayer="{drawGraphicsLayer}"
markerSymbol="{sms}" lineSymbol="{sls}" fillSymbol="{sfs}" />

<esri:Map id="map">
    <esri:ArcGISTiledMapServiceLayer
    url="http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_Imagery_World_2D/MapS
    erver" />
    <esri:GraphicsLayer id="drawGraphicsLayer"/>
</esri:Map>

<mx:HBox width="100%" horizontalAlign="center">
    <mx:ToggleButtonBar id="tbb" itemClick="itemClickHandler(event)">
        <mx:dataProvider>
            <mx:Array>
                <mx:String>点</mx:String>
                <mx:String>多点</mx:String>
                <mx:String>直线</mx:String>
                <mx:String>多线</mx:String>
                <mx:String>手绘多线</mx:String>
                <mx:String>多边形</mx:String>
                <mx:String>手绘多边形</mx:String>
                <mx:String>矩形</mx:String>
            </mx:Array>
        </mx:dataProvider>
    </mx:ToggleButtonBar>
</mx:HBox>

<esri:SimpleMarkerSymbol id="sms" style="{SimpleMarkerSymbol.STYLE_DIAMOND}"
color="0xFFFF00" size="12"/>
<esri:SimpleLineSymbol id="sls" style="{SimpleLineSymbol.STYLE_DASHDOT}"
color="0x00FF00" width="2"/>
<esri:SimpleFillSymbol id="sfs" style="{SimpleFillSymbol.STYLE_DIAGONAL_CROSS}">
    <esri:outline>
        <esri:SimpleLineSymbol color="0xFF0000" width="2"/>
    </esri:outline>
</esri:SimpleFillSymbol>
```

同样，当我们点击 ToggleButtonBar 中某个按钮时，Draw 工具会根据当前 ToggleButtonBar 所选中的按钮来激活某个绘制功能。可选的几何对象类型包括点、多点、直线、多线、手绘多线、多边形、手绘多边形、矩形。同时，对这些绘制对象的样式我们也可以进行设置，总的来说，我们可以通过 ArcGIS Flex API 绘制各种不同类型、样式可设置的几何对象。

下面是使用 Draw 工具绘制一些几何对象的效果：

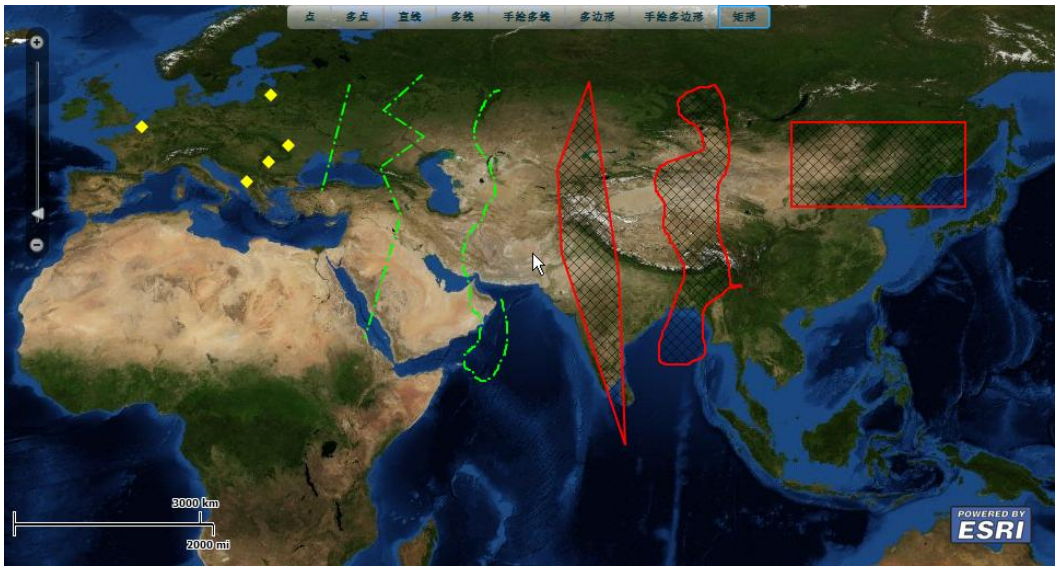


图 12　　Draw 工具可以绘制的几何对象

当我们可以在客户端绘制各种几何对象以后，加上 ArcGIS Server 后台提供的诸多功能，很快我们就可以实现很多业务逻辑了。

4　InfoWindow 窗口

InfoWindow 窗口主要提供在 Map 上弹出信息提示的功能，比如当我点击一个要素的时候，可能同时需要查询这个要素的属性值并在地图上显示出来，InfoWindow 窗口主要就是实现这样的功能。

当定义一个 Map 的时候，Map 会生成一个自带的 InfoWindow 并放置在自己的 infoWindow 属性中，因此 InfoWindow 窗口并不需要你去定义。你只需要知道 InfoWindow 应该弹出的位置（地图上的某个坐标点），那么在这个地方会弹出一个提示窗口，并且不会随着地图移动、缩放而移动位置或消失。

下面我们在地图上弹出一个 InfoWindow，并在其中放上一个图片：

```
var image:Image = new Image();
image.source = "http://www.esrichina-bj.cn/images/in_flashRig.jpg";

map.infoWindow.content = image;
map.infoWindow.show(new MapPoint(116, 40));
```

InfoWindow 使用的重点在于其 content 属性，它是 InfoWindow 窗口中的信息内容，它对应一个 DisplayObject 的对象，言下之意任何 Flex 中的可视组件都可以放置于其中。在这里我在其中放置了一张图片，当然，你可以使用任何按钮、文本、图表、视频等元素及其组合。

下面是在北京弹出一个 InfoWindow 的效果：



图 13　　弹出的 InfoWindow 窗口

从上面的内容，我们知道了如何使用一个 Map 来获得和显示 ArcGIS Server 发布

的服务 ,然后使用 Navigation 工具来进行导航 ,用 Draw 工具在客户端绘制几何对象，用 InfoWindow 来显示提示信息。对于地图操作的浏览、交互我们都已经涉及到了。下面，我们将会在地图的基础上使用更多的 ArcGIS Server 的 GIS 功能。

五 常用功能

通过前面的内容，我们已经知道 ArcGIS Flex API 怎么实现并使用一个“地图”。当然，对于 GIS 应用来说，仅仅实现地图功能是不够的，我们需要融入各自独特的业务逻辑和空间分析功能。那么，接着我们就来看一下 ArcGIS Flex API 中一些 GIS 的常用功能。

1 QueryTask

QueryTask 是一个进行空间和属性查询的功能类，它可以在某个地图服务的某个子图层内进行查询，顺便需要提一下的是，QueryTask 进行查询的地图服务并不必须加载到 Map 中进行显示。

QueryTask 的执行需要两个先决条件：一个是需要查询的图层 URL、一个是进行查询的过滤条件，下面让我们通过代码看一下这两个条件应该怎样设置：

```
<mx:Script>
    <![CDATA[
        private function doQuery():void
        {
            var query:Query = new Query();
            query.geometry = map.extent;
             query.spatialRelationship = Query.SPATIAL_REL_WITHIN;
            query.where = "Name like '%"+keyword.text+"%'";
             query.outFields = ["Name", "Address"];
             query.returnGeometry = true;
             queryTask.execute(query);
        }

        private function onResult(event:QueryEvent):void
        {
             var featureSet:FeatureSet = event.featureSet;
            resultLayer.clear();
            for each (var g:Graphic in featureSet.features)
```

```
                {
                    resultLayer.add(g);
                }
            }
        ]]>
</mx:Script>

<esri:QueryTask id="queryTask"
    url="http://localhost:8399/arcgis/rest/services/Beijing/beijing/MapServer/0"
 executeComplete="onResult(event)" />
```

QueryTask 有个 url 属性指向需要查询的子图层的 REST URL（索引从顶部图层为 0 开始）；当需要进行查询的时候我们需要调用 QueryTask 的 execute 方法，execute 方法的第一个参数就是对查询进行设置的 Query 对象，在这个 Query 对象的属性中我们可以设置过滤条件和返回内容。

在 Query 对象的属性中，geometry 和 spatialRelationship 属性用于定义空间查询的条件、where 属性用于定义属性查询的条件。在上面的代码中，我们设置的查询条件是：在当前地图范围内，所有“Name”字段中包含关键字字符串的对象。

当 execute 方法被调用，服务器收到请求并进行查询后，服务器会将结果发给 ArcGIS Flex API，因此 QueryTask 对象的 executeComplete 监听器将会被触发，接下来我们可以在监听的 onResult 方法中对结果进行处理。

从 onResult 方法的 QueryEvent 参数中我们可以获得查询返回的所有结果集 featureSet，featureSet 的 features 属性中存放了所有查询结果的 Graphic 对象。实际上 ArcGIS Server 查询到结果并输出响应给 ArcGIS Flex API 的时候，ArcGIS Flex API 就自动将这些结果构造成了一个 FeatureSet 对象，在 FeatureSet 对象中不仅包含了和查询结果相对应的 Graphic 对象，还包含比如几何对象类型、字段别名列表等相关信息。

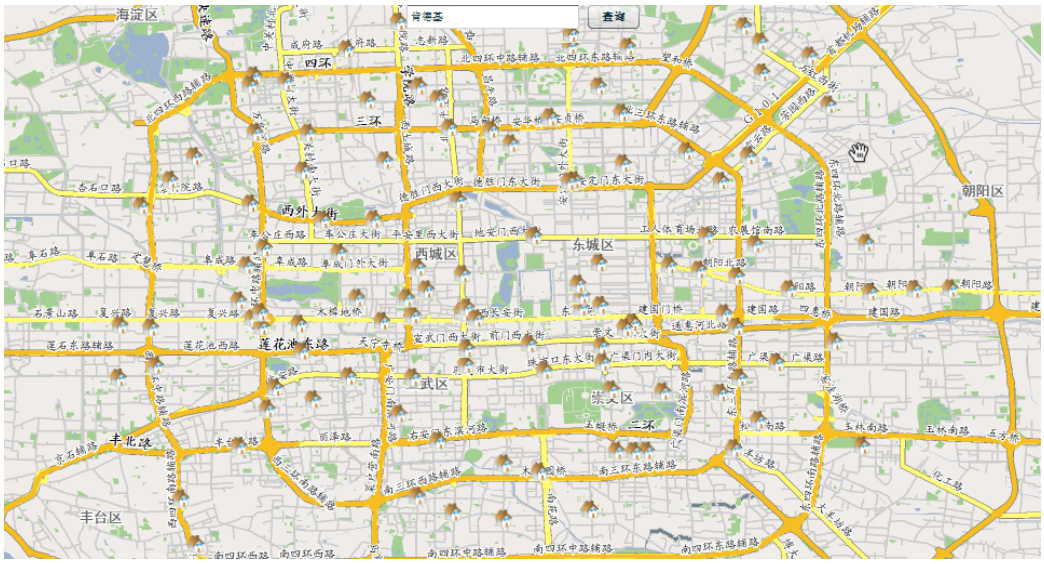


图 14　　QueryTask 的使用

2 FindTask

FindTask 是在某个地图服务中进行属性查询的功能类。FindTask 与 QueryTask 的使用很类似，当然，QueryTask 在 execute 的时候需要给一个 Query 对象作为参数，FindTask 则是给一个 FindParameters 对象作为参数。另外，FindTask 的 url 属性需要指向所查询的地图服务的 REST URL，而不像 QueryTask 需要指定子图层的 URL。

下面我们看一段 FindTask 使用的代码：

```
<mx:Script>
    <![CDATA[
     private function doFind():void
     {
            var params:FindParameters = new FindParameters();
            params.contains = true;
            params.layerIds = [0,1,2];
            params.returnGeometry = true;
            params.searchFields = ["Name", "Address"];
            params.searchText = keyword.text;

            findTask.execute(params);
     }

        private function onResult(event:FindEvent):void
```

```
        {
            var results:Array = event.findResults;
            var resultCount:int = results.length;
            resultLayer.clear();
            for (var i:Number=0; i<resultCount; i++)
            {
                var g:Graphic = results[i].feature;
                resultLayer.add(g);
            }
        }
    ]]>
</mx:Script>

<esri:FindTask id="findTask"
    url="http://localhost:8399/arcgis/rest/services/Beijing/beijing/MapServer"
    executeComplete="onResult(event)" />
```

FindParameters 对象有一些很重要的属性设置了 FindTask 执行的过滤条件，contains 属性设置是否进行模糊查询、layerIds 属性设置了需要查询的子图层、searchFields 属性设置了需要查询的字段、searchText 属性设置了需要查询的字符串。通过以上的设置，我们就可以从一个地图服务的若干个图层的若干个字段中找到我们需要查询的属性信息了。

同样，在查询完成以后，FindTask 的 executeComplete 监听器被触发，onResult 方法被调用，从它的 FindEvent 事件中可以获得所有查询到的结果，这个结果是一个 Array 类型的集合，这个集合的每个对象都有一个 feature 属性，它就是该结果对应 Graphic 对象，我们可以直接把这些 Graphic 对象标注到地图上去：

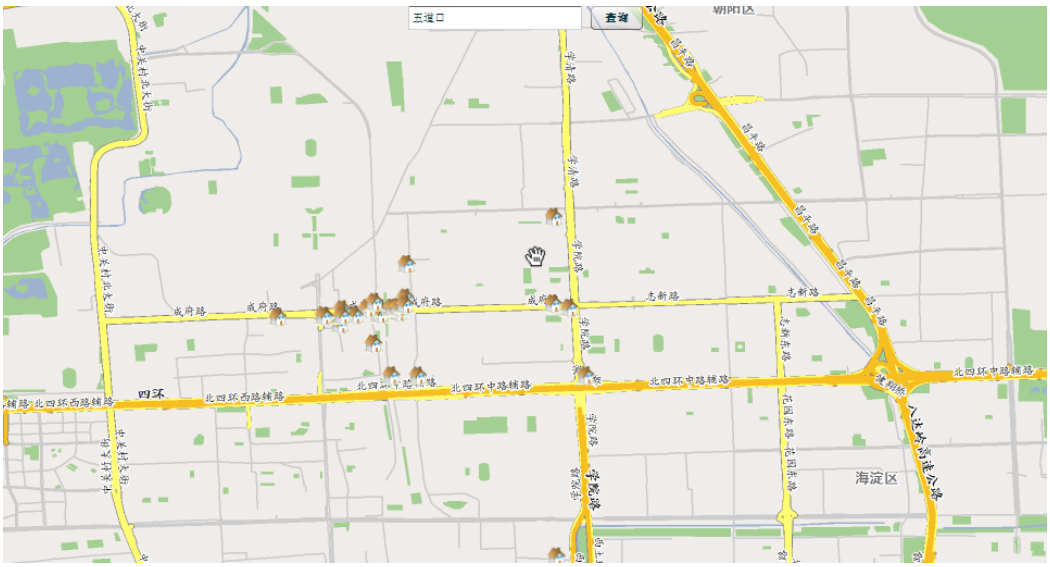


图 15　FindTask 的使用

3 IdentifyTask

IdentifyTask 是一个在地图服务中识别要素的功能类。当用户在客户端使用 Draw 工具绘制了一个几何对象以后，这个几何对象就可以作为 IdentifyTask 的参数发送到服务器进行识别，满足条件的要素将会被输出到 ArcGIS Flex API 中，同样，这些要素都可以作为 Graphic 被添加到地图上。

下面我们看一段实现 IdentifyTask 功能的代码：

```
<mx:Script>
    <![CDATA[
        private function onDrawEnd(event:DrawEvent):void
        {
            var identifyParams:IdentifyParameters = new IdentifyParameters();
            identifyParams.returnGeometry = true;
            identifyParams.width = map.width;
            identifyParams.height = map.height;
            identifyParams.mapExtent = map.extent;
            identifyParams.geometry = event.graphic.geometry;
            identifyParams.tolerance = 3;
            identifyParams.layerOption = IdentifyParameters.LAYER_OPTION_VISIBLE;

            identifyTask.execute(identifyParams);
        }
```

```
        private function onIdentifyComplete(event:IdentifyEvent):void
        {
              var results:Array = event.identifyResults;
            for each (var result:IdentifyResult in results)
            {
                  resultLayer.add(result.feature);
            }
        }
    ]]>
</mx:Script>

<esri:IdentifyTask id="identifyTask"
    url="http://localhost:8399/arcgis/rest/services/China/map/MapServer"
    identifyComplete="onIdentifyComplete(event)" />
```

与 QueryTask 和 FindTask 相仿 ,IdentifyTask 在调用 execute 方法的时候需要提供的参数是一个 IdentifyParameters 对象。这个对象最重要的属性是 geometry、layerOption 和 tolerance，geometry 属性设置的是用于要素识别的几何对象、layerOption 属性设置的要素识别的选项（最顶层还是所有可见图层等）、tolerance 属性设置了识别的误差范围（基于屏幕像素）。通过它们的组合，IdentifyTask 就可以把满足条件的要素查询出来，并返回给 ArcGIS Flex API 进行处理。

对于 identifyComplete 监听器的处理，因为和 QueryTask 和 FindTask 的 executeComplete 监听器类似，这里就不赘述了。下面是分别用点、线、面进行要素识别的结果：

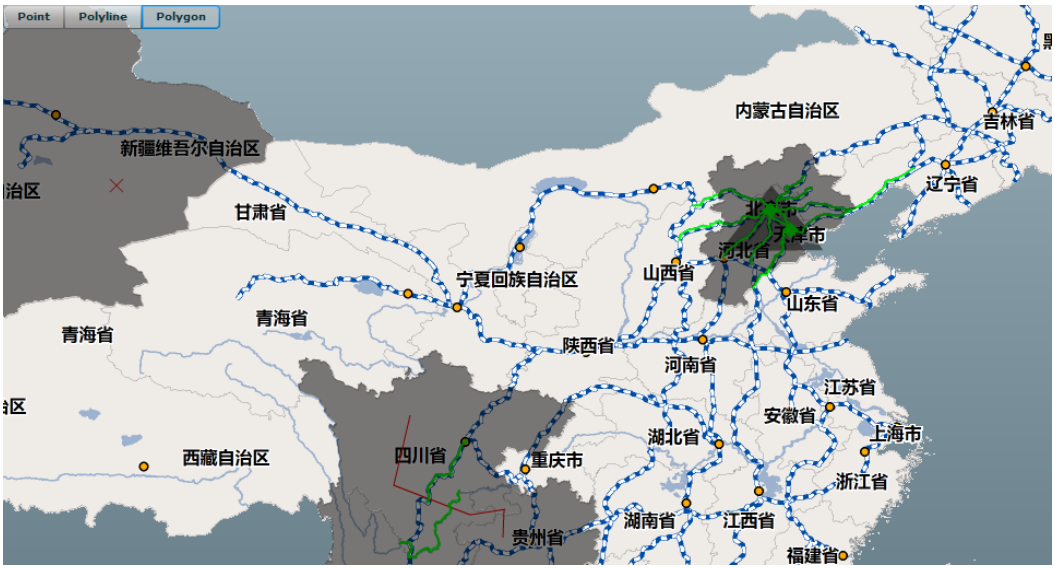


图 16　IdentifyTask 的使用

4 GeometryService

每个 ArcGIS Server 都可以发布一个几何服务（Geometry Service）用以处理几何对象，比如投影转换、简化对象、缓冲、量测、判断空间关系、计算标注点等功能。ArcGIS Flex API 中对应地定义了一个 GeometryService 功能类来使用几何服务提供的功能。

由于 GeometryService 提供的功能比较多，因此它实现了很多监听器供不同的功能使用，比如投影转换有 projectComplete 监听、判断空间关系有 relationComplete 监听等等。在使用的时候，分别针对不同的功能，调用不同的 project 方法、relation 方法等（与上面的监听对应）。当然，GeometryService 的功能并不仅限于上面提到的两个方法，下表给出了 Geometry 的主要功能：

project	投影转换
simplify	简化几何对象
buffer	缓冲
areasAndLengths	量测多边形的面积与周长
lengths	量测多段线的长度

relation	判断空间关系
labelPoints	计算标注点

由于 GeometryService 不同功能的使用有类似性，我们这里就挑一个功能进行了解，如下的代码调用了 Geometry 的 relation 方法，实现了判断几何对象空间关系的功能：

```
<mx:Script>
 <![CDATA[
      private var graphics1:Array;
      private var graphics2:Array;

      private function onQueryCity(event:QueryEvent):void
      {
            graphics1 = new Array();
            var featureSet:FeatureSet = event.featureSet;
            var count:int = featureSet.features.length;
            for (var i:int=0; i<count; i++)
            {
                  var g:Graphic = featureSet.features[i];
                  graphics1.push(g);
            }
      }

      private function onQueryProvice(event:QueryEvent):void
      {
            var featureSet:FeatureSet = event.featureSet;
            var g:Graphic = featureSet.features[0];
            graphics2 = [g];

            geometryService.relation(graphics1, graphics2,
                        GeometryService.SPATIAL_REL_INTERSECTION);
      }

      private function onRelationComplete(event:GeometryServiceEvent):void
      {
            var relations:Array = event.relations;
            resultLayer.clear();
            for (var i:int=0; i<relations.length; i++)
            {
                  var g:Graphic = relations[i].graphic1;
                  resultLayer.add(g);
```

```
            }
        }
]]>
</mx:Script>

<esri:GeometryService id="geometryService"
 url="http://localhost:8399/arcgis/rest/services/Geometry/GeometryServer"
 relationComplete="onRelationComplete(event)" />
```

以上的 GeometryService 调用了 relation 方法，判断了两个 Graphic 对象集合之间，取出满足“INTERSECTION”条件的所有对象。下图是实际运行的结果，实现的功能是“得到江苏省内所有地级市”。



图 17　GeometryService 的使用

5 Geoprocessor

GP 服务（Geoprocessing Service）可以使用户能够使用高级的模型功能，ArcGIS Flex API 中的 Geoprocessor 功能类对应的就是 GP 服务的使用。

Geoprocessor 功能的调用有两种方式：execute 方法和 submitJob 方法，其中 execute 对应同步调用（synchronous）、submitJob 对应异步调用（asynchronous）

[1]，当你知道一个 GP 服务是同步还是异步以后，你就知道应该使用 execute 还是 submitJob 方法来使用这个 GP 服务了。

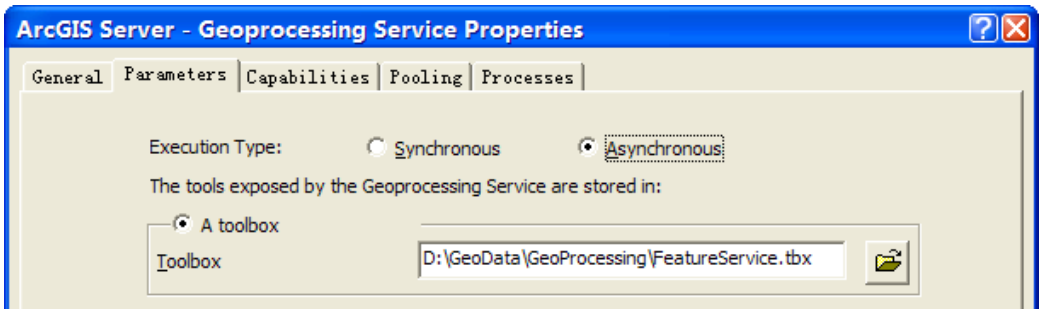


图 18　发布 GP 服务时选择同步或异步调用

关于怎么准备数据、建立模型、发布 GP 服务，这里就不详细讲述了，下面我们先通过一个“漂流瓶之旅”的例子来看一下 Geoprocessor 怎样调用 ArcGIS Server 发布的 GP 服务。这个例子调用的是 ArcGIS Online 上的一个 GP 服务，服务的地址是 http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_Currents_World/GPServer/MessageInABottle，你可以在浏览器中打开这个地址，查看关于这个服务的详细内容，在这里我们可以看到，这个服务是同步的，需要输入两个条件参数 :Input_Point(GPFeatureRecordSetLayer 类型)和 Days(GPDouble 类型)，输出的结果是 Output（GPFeatureRecordSetLayer 类型)。

下面就我们看一下调用这个 GP 服务的代码：

```
<mx:Script>
    <![CDATA[
        private function startGeoProcess(event:MapMouseEvent):void
        {
            this.cursorManager.setBusyCursor();

            var mapPoint:Geometry = event.mapPoint;
            var startGraphic:Graphic = new Graphic(mapPoint, sms);
            resultLayer.add(startGraphic);
```

[1] 同步调用时用户必须等待调用结束后方可继续操作，异步则不需要

```
            var featureSet:FeatureSet = new FeatureSet();
            featureSet.features = [ { geometry:mapPoint } ];

            var parms:Object = new Object();
            parms.Input_Point = featureSet;
            parms.Days = Number(numberOfDays.text);
            geoprocessTask.execute(parms);
        }

        private function onExecuteComplete(event:GeoprocessorEvent):void
        {
            for each (var g:Graphic in
                event.executeResult.parameterValues[0].value.features)
            {
                g.symbol = sls;
                resultLayer.add(g);
            }

            this.cursorManager.removeBusyCursor();
        }
    ]]>
</mx:Script>

<esri:Geoprocessor id="geoprocessTask"
 url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_Currents_
World/GPServer/MessageInABottle"
 executeComplete="onExecuteComplete(event)" />
```

这个 GP 服务可以计算海洋中洋流的活动，假设我们在地图上点击了一下，表示在这个地方投了一个漂流瓶，调用这个 GP 服务可以计算若干天内这个漂流瓶的漂流轨迹。

在调用这个 GP 服务之前，首先我们需要获得两个参数：Input_Point 和 Days。Input_Point 是一个 FeatureSet 对象，"[{ geometry:mapPoint }]" 代表的是 Feature 对象的集合，当然，在这里只存放了一个 Feature 对象，这个 Feature 对象的几何对象是鼠标点击处的 mapPoint；Days 则很简单，直接获得文本框中的文字后转成数值即可。当已经获得这两个参数，并把这两个参数放到一个 Object 对象中做为 Geoprocessor 执行参数使用后，服务器上的 GP 服务就可以开始处理这个计算了。

当 GP 服务计算完成并返回结果时，onExecuteComplete 监听将会被触发，我们可以把计算结果绘制到一个 GraphicLayer 上。

下面是这个 Geoprocessor 执行的效果：

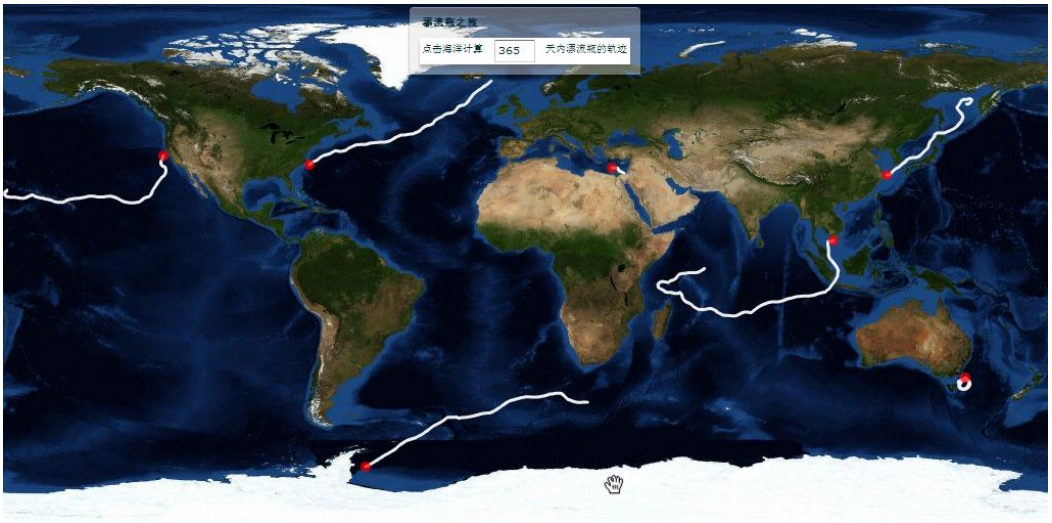


图 19　Geoprocessor 的使用

6 RouteTask

RouteTask 对应的是路径分析功能，它需要 ArcGIS Server 9.3.1 的支持，因此我们这里使用 Sample 中使用的在线网络分析服务：

http://tasks.arcgisonline.com/ArcGIS/rest/services/NetworkAnalysis/ESRI_Route_NA/NAServer/Route。

从这个服务的 REST 目录我们可以看到 ,网络分析服务的主要操作是 Solve Route，在 ArcGIS Flex API 中对应调用的就是 RouteTask 的 solve 方法，下面让我们从代码看一下 RouteTask 的使用：

```
<mx:Script>
    <![CDATA[
        [Bindable]private var stops:FeatureSet = new FeatureSet([]);
        [Bindable]private var barriers:FeatureSet = new FeatureSet([]);
        [Bindable] private var lastRoute:Graphic;
```

```
        private function mapClickHandler(event:MapMouseEvent):void
        {
            if (selectedBtn == addStopsBtn)
            {
                var stop:Graphic = new Graphic(event.mapPoint, stopSymbol);
                inputsLayer.add(stop);
                stops.features.push(stop);
            }
            else
            {
                var barrier:Graphic = new Graphic(event.mapPoint, barrierSymbol);
                inputsLayer.add(barrier);
                barriers.features.push(barrier);
            }

            if (stops.features.length > 1)
            {
                routeTask.solve(routeParams);
            }
        }

        private function solveCompleteHandler(event:RouteEvent):void
        {
            var routeResult:RouteResult = event.routeSolveResult.routeResults[0];
            lastRoute = routeResult.route;
        }
    ]]>
</mx:Script>

<esri:RouteTask id="routeTask"
    concurrency="last" requestTimeout="30" showBusyCursor="true"
    solveComplete="solveCompleteHandler(event)"
    url="http://tasks.arcgisonline.com/ArcGIS/rest/services/NetworkAnalysis/ESRI_Route_NA/NAServer/Route"/>
<esri:RouteParameters id="routeParams" stops="{stops}" barriers="{barriers}"/>

<esri:Map mapClick="mapClickHandler(event)">
    <esri:GraphicsLayer graphicProvider="{lastRoute}" symbol="{routeSymbol}"/>
    <esri:GraphicsLayer id="inputsLayer"/>
</esri:Map>
```

在这个例子中，每次在地图上点击会添加一个停靠点或路障点，每次点击后都会调

用 RouteTask 进行计算，计算的条件就是前面在地图上点击添加的停靠点及路障点；当然，由于设置了 concurrency 属性，因此只有最后一次调用可以返回结果。在这里顺便说一下 concurrency 属性 其实前面的那些功能类也都有这个属性 ,它是 BaseTask 基类的一个属性，它指示的是该功能是否允许多次调用，你可以选择多次调用、单次调用或者取消前面的调用仅执行最后一次调用。

下面让我们看一下路径分析的结果：

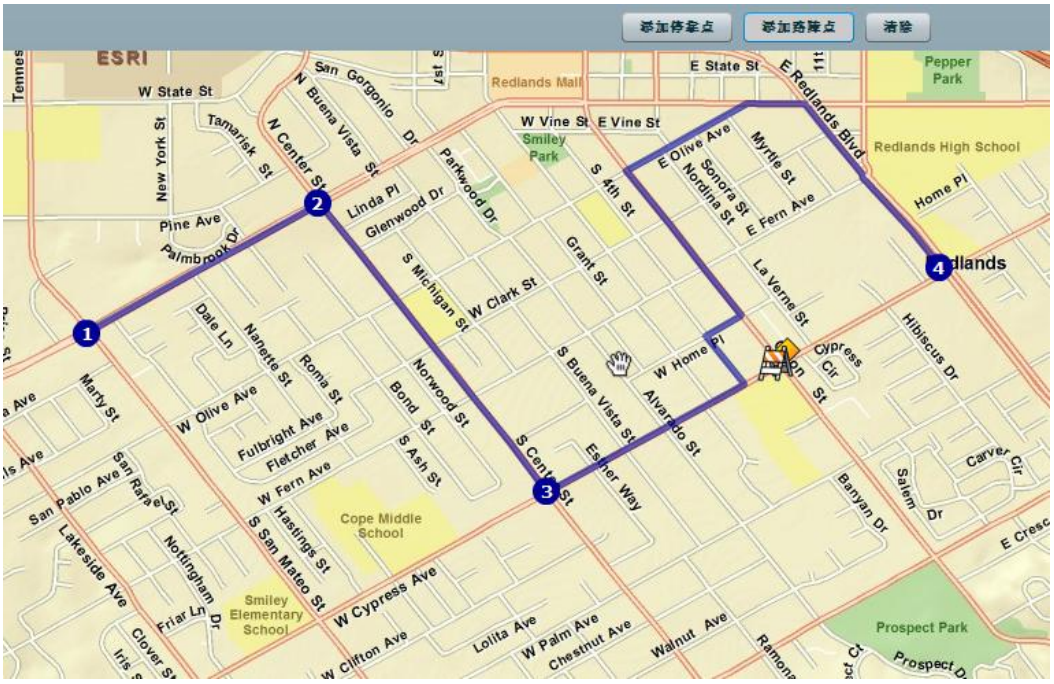


图 20　RouteTask 的使用

7 Locator

Locator 对应使用的是 ArcGIS Server 发布的地理编码服务(Geocode Service)，它可以从地址找到坐标或从坐标找到地址。比如下面的 URL 对应的就是一个 Geocode Service 的 REST 接口：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/ESRI_Geo

code_NA/GeocodeServer ,从浏览器打开这个链接我们可以看到关于这个服务相关信息。

从服务的描述中可以看到，Geocode Service 包含两个操作：Find address candidates 和 Reverse geocode ,前者从地址找到坐标 ,后者则可以从坐标找到地址，它们在 ArcGIS Flex API 中分别对应了 Locator 功能类的 addressToLocations 方法和 locationToAddress 方法。

下面我们使用 Locator 的 locationToAddress 方法从地图的点击点找到相应的地址信息：

```
<mx:Script>
    <![CDATA[
        private function onMapClick(event:MapMouseEvent):void
        {
            locator.locationToAddress(event.mapPoint, 100);
        }

        private function onLocationToAddressComplete(event:LocatorEvent):void
        {
            var candidate:AddressCandidate = event.addressCandidate;

            if (candidate != null && candidate.address)
            {
                var g:Graphic = new Graphic(candidate.location, sms, candidate.address);
                    graphicLayer.clear();
                graphicLayer.add(g);

                var text:TextArea = new TextArea();
                text.htmlText = "<b>地址:</b>\n"
                    + candidate.address.Address.toString() + "\n"
                    + candidate.address.City.toString() + "," +
candidate.address.State.toString();
                map.infoWindow.content = text;
                map.infoWindow.show(g.geometry as MapPoint);
            }
        }
    ]]>
```

```
</mx:Script>

<esri:Locator
    id="locator"
    url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/ESRI_Geocode_NA/GeocodeServer"
    concurrency="last"
    locationToAddressComplete="onLocationToAddressComplete(event)"/>
```

Locator 的方法调用很简单，经过前面这么多功能的使用，我们基本上也是熟稔于胸了。从上面我们可以看到，调用 locationToAddress 方法需要两个参数，一个是坐标点、另一个是距离（单位：米），当 Geocode Service 处理完并发回结果时，我们就可以把这个结果标到地图上了。

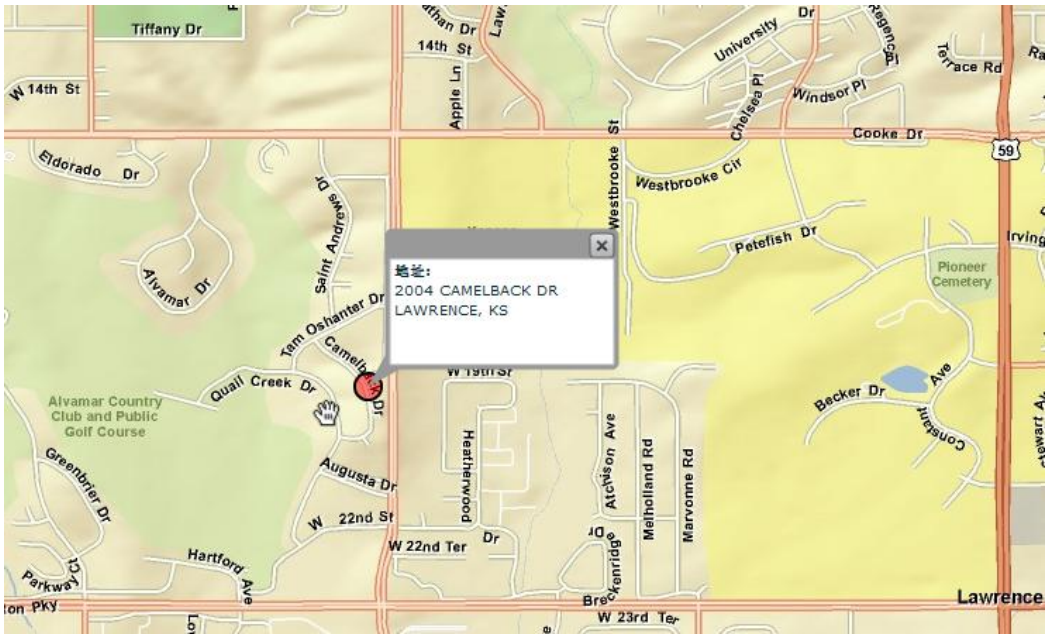


图 21　Locator 的使用

六 ArcGIS Server 中配置安全策略文件

Flash Player 对访问外部资源有比较严格的限制，因此如果你需要访问跨域的 ArcGIS Server 服务(比如本地非本机、不同域等),一定要注意 Flash 的安全策略问题。当你在使用 ArcGIS Flex API 的时候出现了安全沙箱错误（注意，这个错误在调试状态下不会出现，因为 IDE 已经忽略了安全策略）；或者运行的时候出不来地图，但是在浏览器中查看 REST 服务目录是正常的，那么这个时候就需要注意下你是不是进行了跨域访问。

配置服务器允许被 Flash 进行跨域访问一般需要在远程的 Web 服务器根目录下放置一个安全策略文件。当我们开发一个 ArcGIS Flex API 应用，同时它需要访问跨域的 ArcGIS Server 的时候，一定要在 ArcGIS Server 的 REST Web 根目录下放置如下名为“crossdomain.xml”的安全策略文件：

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="all"/>
    <allow-access-from domain="*"/>
</cross-domain-policy>
```

以上的安全策略是允许所有的连接，这个安全策略文件需要被放到特定的位置，比如 ArcGIS Server for .Net 应该是 IIS 的根目录（如：C:\Inetpub\wwwroot）；ArcGIS Server for Java 应该是 web_output 目录（如：C:\Program Files\ArcGIS\java\web_output）。

II、 ArcGIS Flex API 高级

一 控制 Map 与 Layer

1 Map 中地图服务图层的控制

Map 中图层的控制是很多人首先关心的问题，因为根据业务来切换图层的状态是经常遇到的一种需求。

首先让我们看一下对图层可见性的控制，Layer 本身就有一个 visible 属性，这个属性指示的就是当前图层是否在 Map 中显示。不管是 MapServiceLayer 还是 GraphicLayer，如果设置了 visible 为 false，那么在这个图层是不会被显示的。

除了控制图层是否显示，我们还可以控制图层的透明度，这是由 Layer 的 alpha 属性来控制的。这样我们可以将多个图层叠加复合显示，这时候会给用户一种复合的多信息量的地图：

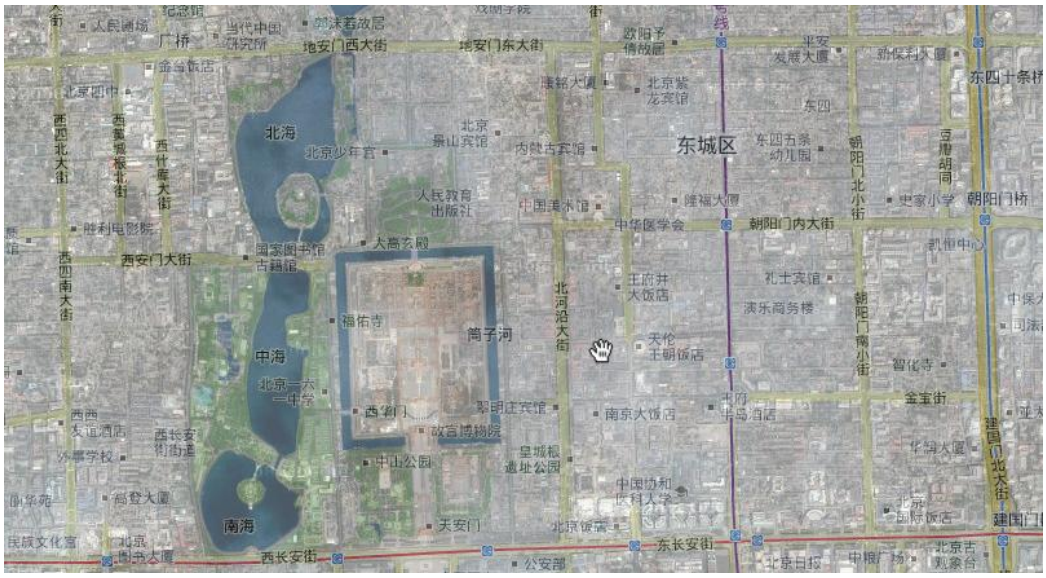


图 22 多图层透明复合效果

另外，图层的叠放顺序也是一个经常遇到的问题，在 ArcGIS Flex API 中，后加入

的地图被放置在最上层，同时，最底层的索引（index）为 0。



图 23　Map 中 Layer 的存放顺序

Map 中有一个 layers 属性存放了地图中所有 Layer 的集合，另外还有 layerIds 属性存放了对应的所有 Layer 的 id 的集合，每当添加一个 Layer 到 Map 中的时候，ArcGIS Flex API 会给这个图层一个 id 并存放到 layerIds 中保存。通过调用 Map 的 reorderLayer 方法可以给这些图层重新排序。

下面我们在 Map 中添加 3 个图层，并在点击地图的时候把最上层的图层移到底层，从而实现图层叠置顺序的改变：

```
<mx:Script>
 <![CDATA[
      private function init():void
      {
            var layer:Layer;
            layer = new ArcGISTiledMapServiceLayer("http://server.arcgisonline.com
/ArcGIS/rest/services/ESRI_Imagery_World_2D/MapServer");
            map.addLayer(layer);
            layer = new ArcGISTiledMapServiceLayer("http://server.arcgisonline.com
/ArcGIS/rest/services/NPS_Physical_World_2D/MapServer");
            map.addLayer(layer);
            layer = new ArcGISTiledMapServiceLayer("http://server.arcgisonline.com
/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer");
            map.addLayer(layer);
      }
      private function onMapClick():void
      {
            map.reorderLayer(map.layerIds[map.layers.length-1], 0);
      }
 ]]>
</mx:Script>
```

```
<esri:Map id="map" click="onMapClick()"/>
```

2 地图服务图层中子图层的控制

不仅 Map 中所有 Layer 是可控制的，一个 Layer 中的内容也是可控制的。当然，GraphicLayer 本身就是客户端的图层，对它的控制自然不在话下；这里我们主要讲的是对 MapServiceLayer 的控制，更确切地说是对 ArcGISDynamicMapServiceLayer 的控制——毕竟 TiledMapService 已经生成了图片，再进行内容上的控制是不可能的。

大家都知道知己知彼百战不殆是至理名言，要想对动态地图服务中的内容进行控制，当然首先需要知道服务中有些什么内容。比如我想控制某个服务只显示其中的某些图层，那么首先必须知道总共有哪些图层才可以。

ArcGISDynamicMapServiceLayer 有一个 layerInfos 属性，这个属性是一个数组，它存放了这个地图服务中所有子图层的相关信息。比如我这里有一个世界地图的服务：

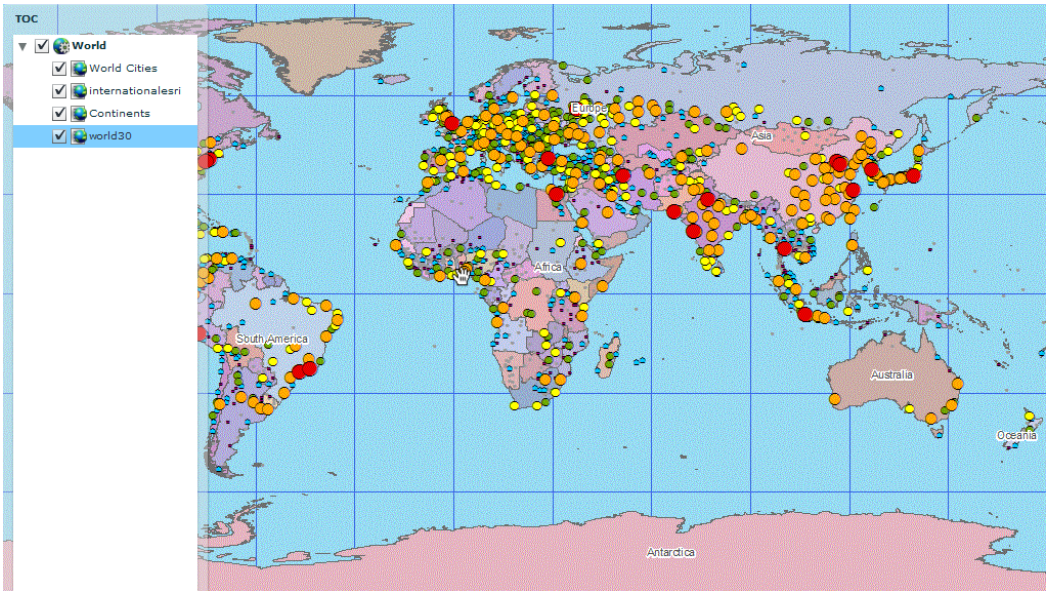


图 24　世界地图服务

在浏览器中打开这个服务的 REST 地址可以看到以下的信息：



图 25　通过 REST 接口查看地图服务的子图层

在 ArcGIS Flex API 项目中调试你可以看到 layerInfos 就是一个长度为 4 的数组，数组中存放了跟上面看到的列表类似的信息：

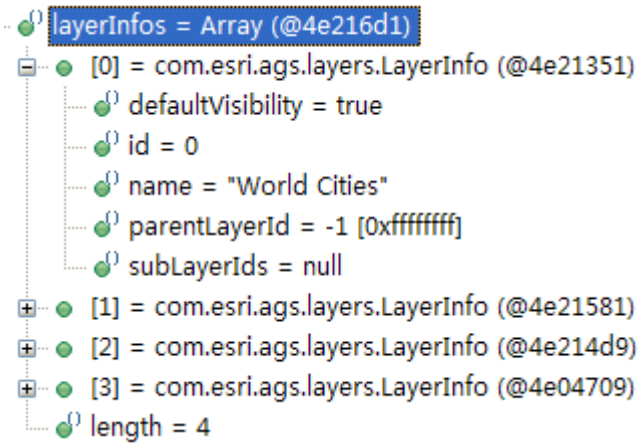


图 26　layerInfos 的值

值得注意的是，这里子图层的 id 虽然也是从 0 开始，但是和 Map 中的地图服务图层索引正好相反，最上层的子图层为 0。

另外，这里还不得不用一些篇幅来特别说明一下关于地图服务加载的话题。当我们把一个 ArcGISDynamicMapServiceLayer 或 ArcGISTiledMapServiceLayer 添加到 Map 中的时候，初始状态只不过是在 Map 中添加了一个自定义的 UIComponent；然后，这些地图服务图层会根据 url 属性的设置向服务器发送请求，获取关于这个地图服务的描述，这个过程称之为“加载”；当加载完成以后，这个 Layer 在 Map 中就算是定位了，然后它会根据 Map 的操作来计算自身需要从 ArcGIS Server 服务器获取什么

数据；随即获取数据然后再显示出来。以上最关键的过程是“加载”的完成，只有在加载完成以后，Layer 的 load 监听被触发，这个时候地图服务图层的 layerInfos 属性才会有值，这个图层才算真正可以使用。

当我们获得 layerInfos 以后，我们就可以控制服务中的子图层了。控制通过 ArcGISDynamicMapServiceLayer 的 visibleLayers 属性实现，visibleLayers 对应一个数组，这个数组存放了需要显示的所有子图层的 id。比如上面的世界地图服务，如果我设置 visibleLayers 值为[0, 2, 3]，那么 id 为 1 的国界图层就会被隐藏掉，地图会变成如下的效果：

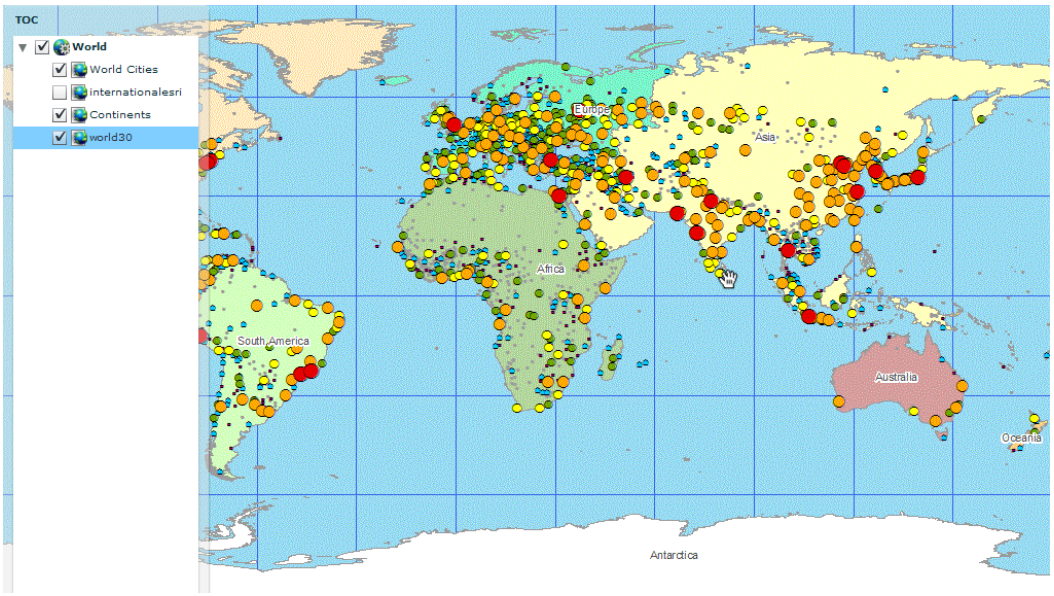


图 27　设置 visibleLayers 属性后的世界地图服务

3　子图层中内容的控制

上面我们了解了 Map 中地图服务图层的控制和地图服务图层中子图层的控制，这个小节里我们再来看一下如何控制某个子图层中的内容。

ArcGISDynamicMapServiceLayer 有一个 layerDefinitions 属性，这个属性对应

的是一个和 layerInfos 长度相同的数组——你也可以联想到，这个数组中的对象和子图层是一一对应的。

layerDefinitions 数组中的对象是字符串，索引从 0 开始，每个字符串对应一个子图层。这个字符串是 SQL 语句中条件子句的 WHERE 部分，它可以使满足条件的要素才被显示。

下面的 MXML 给上面使用过的世界地图服务图层设置了一个 layerDefinitions 值，它给 id 为 1(可以看到它是 layerDefinitions 数组中的第 2 个元素)的国界子图层做了一个过滤，让它只显示中国的国界：

```
<esri:ArcGISDynamicMapServiceLayer
 url="http://localhost:8399/arcgis/rest/services/AgsSample/World/MapServer">
    <esri:layerDefinitions>
      <mx:Array>
            <mx:String></mx:String>
            <mx:String><![CDATA[CNTRY_NAME='China']]></mx:String>
            <mx:String></mx:String>
            <mx:String></mx:String>
      </mx:Array>
    </esri:layerDefinitions>
</esri:ArcGISDynamicMapServiceLayer>
```

其中“<![CDATA[”和“]]>”是 XML 解析器的忽略标签，标签中的“ CNTRY_NAME='China' ”就是对图层 1 设置的过滤条件。

下面是按照上面的代码设置了 layerDefinitions 属性值后的世界地图服务图层效果。

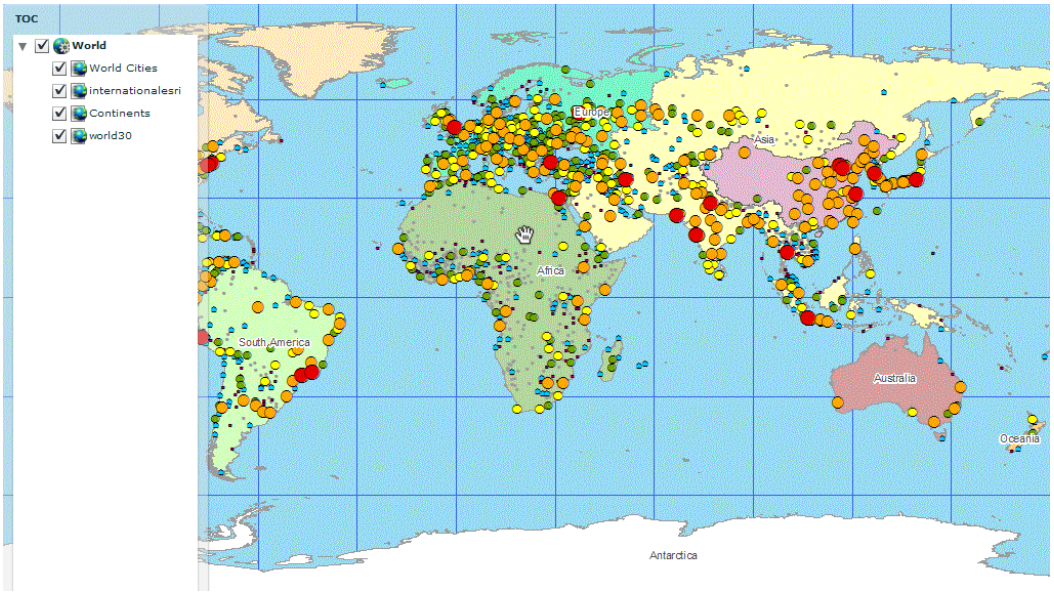


图 28　设置 layerDefinitions 属性后的世界地图服务

在这里你可以想象一下，你是不是可以根据你的业务来随意设定在地图上需要表现出来的要素了呢？

4　动态投影

对于动态地图服务（Dynamic Map Service），你可以根据需要设置不同的空间参考，从而实现动态的投影。

对 Map 的投影设置可以通过给 Map 指定不同空间参考的 extent 属性来实现，下面的代码给 Map 设置了 Well-Known ID 为 102113 的空间参考，该空间参考采用 WGS84 地理坐标和 Web Mercator 投影：

```
<esri:Map>
    <esri:extent>
        <esri:Extent spatialReference="{new SpatialReference(102113)}"
         xmin="-20037508.342787" xmax="20037508.342787"
         ymin="-20037508.342787" ymax="20037508.342787"/>
    </esri:extent>
    <esri:ArcGISDynamicMapServiceLayer
     url="http://localhost:8399/arcgis/rest/services/AgsSample/World/MapServer"/>
```

```
</esri:Map>
```

这个 102113 空间参考在 ArcGIS Server 中已经有定义，事实上 ArcGIS Flex API 只是保存了这个空间参考的 id，本身并不作任何的计算。另外，设置的 Well-Known ID 必须是 ArcGIS Server 中已有的定义，具体可用的空间参考列表可以在 REST SDK 文档中找到。

我们可以看到，通过上面的设置，Map 中呈现的地图跟原来 4326 空间参考下的地图明显不同：

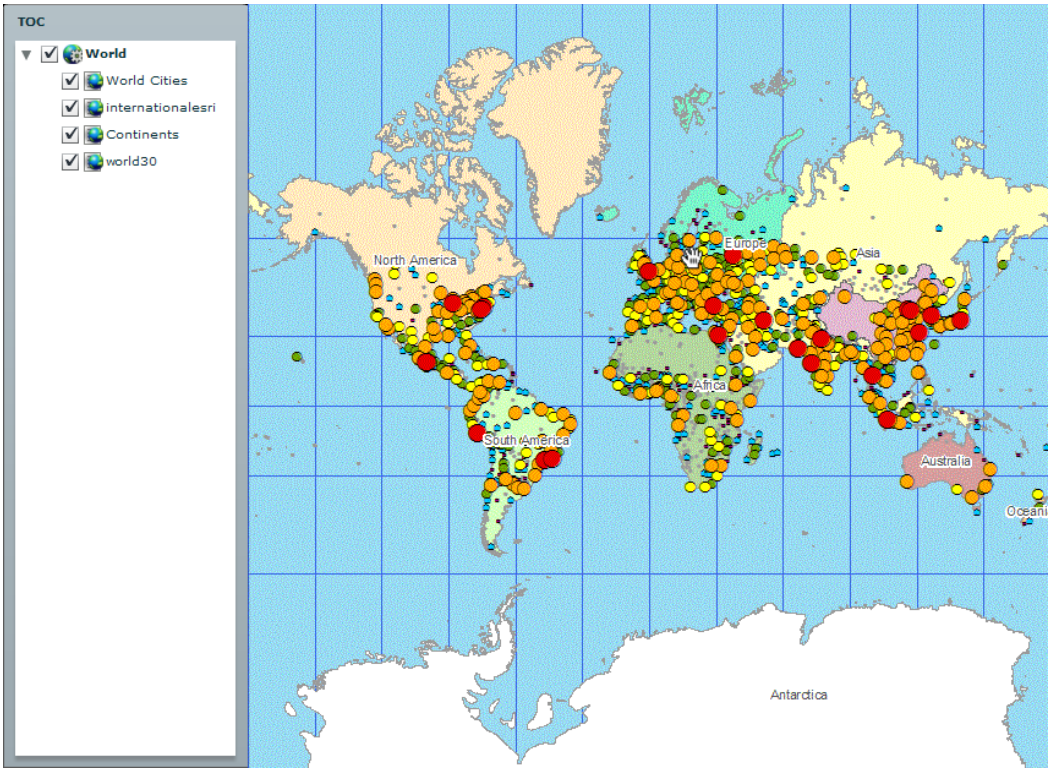


图 29　设置投影为 Web Mercator 的世界地图服务

二 业务信息的表达

1 分类渲染

地图服务图层的分类渲染是在 MXD 文档中定义的，因此没有太多可讨论的，这里主要讨论的是 GraphicLayer 的分类渲染。当你从服务器获得一些数据想标注到 GraphicLayer 上，同时希望这些 Graphic 根据不同的属性值来使用不同的符号标注的时候，这个问题就出现了。

当然，比较笨拙的办法就是每绘制一个 Graphic 之前先判断一下它的属性，然后根据判断的情况来给 Graphic 赋一个不同的符号值——这样的写法或许会让你觉得代码有点混乱，因此，我们这里要讲的是另外一种比较好维护，代码更加优雅的方式。

GraphicLayer 有个属性是 symbolFunction，这个属性对应的是一个函数类型的对象，在这里我们可以定义一个函数来对符号进行判断。

下面的代码实现了这样的功能：从美国 1999 年州人口统计数据中，根据人口的分布区间，使用不同的颜色进行表示。

```
<mx:Script>
 <![CDATA[
      private function getSymbol(g:Graphic):Symbol
      {
            var symbol:SimpleFillSymbol = new SimpleFillSymbol();

            var pop:Number = g.attributes["POP1999"];
            if( pop<1000000 )
            {
                  symbol.color = 0x00FF00;
            }
            else if( pop>=1000000 && pop < 5000000 )
            {
                  symbol.color = 0x55FF00;
            }
```

```
            else if( pop>=5000000 && pop < 10000000 )
            {
                symbol.color = 0xAAFF00;
            }
            else if( pop>=5000000 && pop < 10000000 )
            {
                symbol.color = 0xFFFF00;
            }
            else if( pop>=10000000 && pop < 15000000 )
            {
                symbol.color = 0xFF9900;
            }
            else if( pop>=15000000 && pop < 20000000 )
            {
                symbol.color = 0xFF4400;
            }
            else
            {
                symbol.color = 0xFF0000;
            }

            return symbol;
        }
    ]]>
</mx:Script>

<esri:Map id="map">
    <esri:ArcGISTiledMapServiceLayer
    url="http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_Imagery_World_2D/MapS
erver"/>
    <esri:GraphicsLayer id="graphicLayer" symbolFunction="getSymbol"/>
</esri:Map>
```

GraphicLayer 的 symbolFunction 属性对应的函数的参数是 Graphic 对象，其返回值为 Symbol 对象，这就把 GraphicLayer 上的 Graphic 和它应该对应的 Symbol 联系了起来。在上面的代码中我们给每个 Graphic 都新建了一个 Symbol 对象，实际应用时应该减少 Symbol 的数量，比如我有 n 个区间需要分类渲染，那么可以预先定义 n 个 Symbol，然后在 SymbolFunction 中根据不同的条件选择已经定义好的 Symbol。这也是在 Flex 开发中需要遵循的一个原则：能重用的对象不要重新生成。

下面是实现 GraphicLayer 分类渲染的效果：

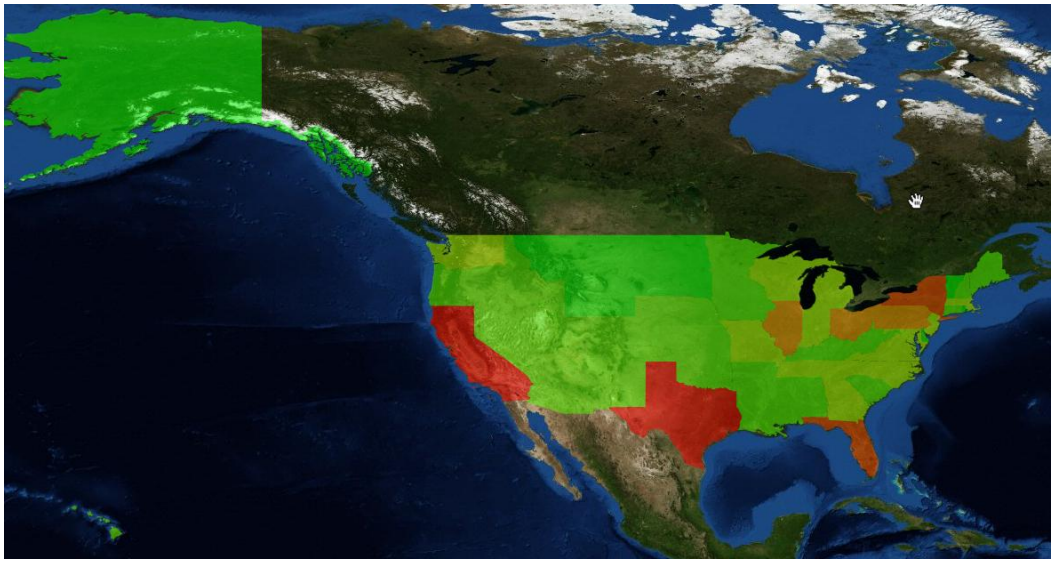


图 30　GraphicLayer 的分类渲染

在 ArcGIS Flex API 1.2 版本中新添加了一个功能类 Renderer，它可以更方便地实现地图的客户端渲染，主要有这样两种 Renderer：ClassBreaksRenderer 和 UniqueValueRenderer，看这个名字是不是感到很亲切？下面让我们看一下用 ClassBreaksRenderer 来实现的分类渲染：

```
<mx:Script>
    <![CDATA[
        private function initApp():void
        {
            // 添加 100 随机的 Graphic，注意它的 ranking 属性
            for (var i:int; i < 100; i++)
            {
                var x:Number = Math.random() * 360 - 180;
                var y:Number = Math.random() * 180 - 90;
                var attributes:Object = { "ranking": Math.random() };
                var g:Graphic = new Graphic(new MapPoint(x, y), null, attributes);
                graphicsLayer.add(g);
            }
        }
    ]]>
</mx:Script>

<esri:SimpleMarkerSymbol id="smallSym" size="6" color="0xFF0000" alpha="0.7"/>
```

```
<esri:SimpleMarkerSymbol id="medSym" size="10" color="0xFF0000" alpha="0.7"/>
<esri:SimpleMarkerSymbol id="largeSym" size="16" color="0xFF0000" alpha="0.7"/>

<esri:Map>
    <esri:ArcGISTiledMapServiceLayer
    url="http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer"/>
    <esri:GraphicsLayer id="graphicsLayer">
        <esri:renderer>
            <esri:ClassBreaksRenderer attribute="ranking">
                <esri:ClassBreakInfo maxValue="0.33" symbol="{smallSym}"/>
                <esri:ClassBreakInfo minValue="0.33" maxValue="0.67"
symbol="{medSym}"/>
                <esri:ClassBreakInfo minValue="0.67" symbol="{largeSym}"/>
            </esri:ClassBreaksRenderer>
        </esri:renderer>
    </esri:GraphicsLayer>
</esri:Map>
```

上面的代码在地图上随机添加了 100 个 Graphic，每个 Graphic 随机指定一个“ranking”属性值，GraphicLayer 的 ClassBreaksRenderer 根据这个“ranking”属性值的分布来决定使用大、中、小三种符号中的哪种符号进行渲染：

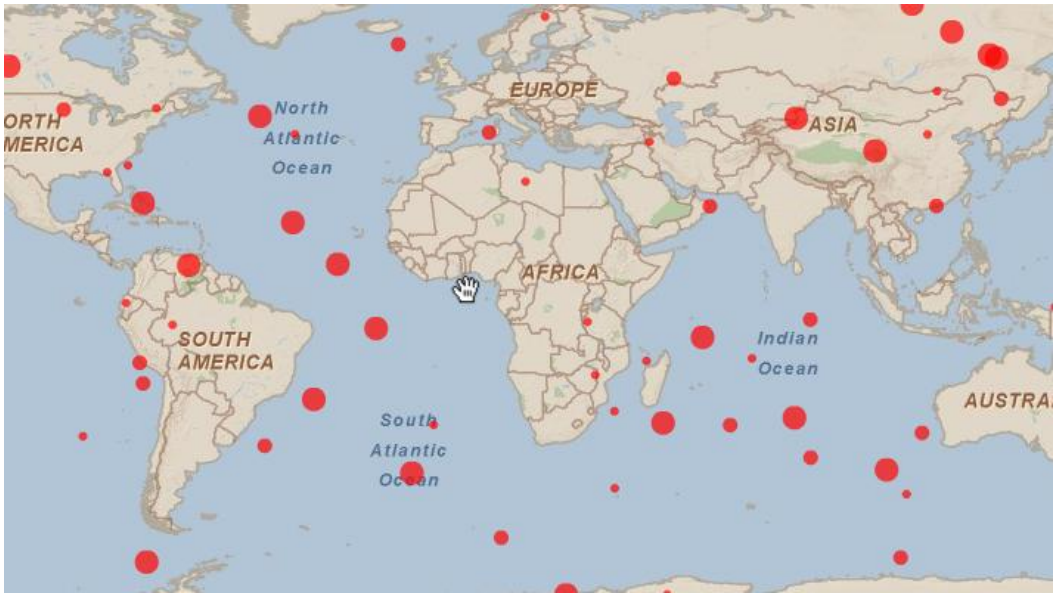


图 31　使用 ClassBreaksRenderer 实现的分类渲染

2 专题图

这一小节让我们去看一下如何使用专题图在地图上表现丰富的业务数据。

专题图主要通过 InfoSymbol 来实现，它可以显示和 InfoWindow 类似的符号。在 InfoSymbol 中放置比如饼图、柱状图等组件后，其对应的 Graphic 就可以被绘制成一个信息非常丰富的符号，同时，符号内的数据和 Graphic 的 attributes 是相关联的。

下面首先看一下使用饼图的专题图效果：

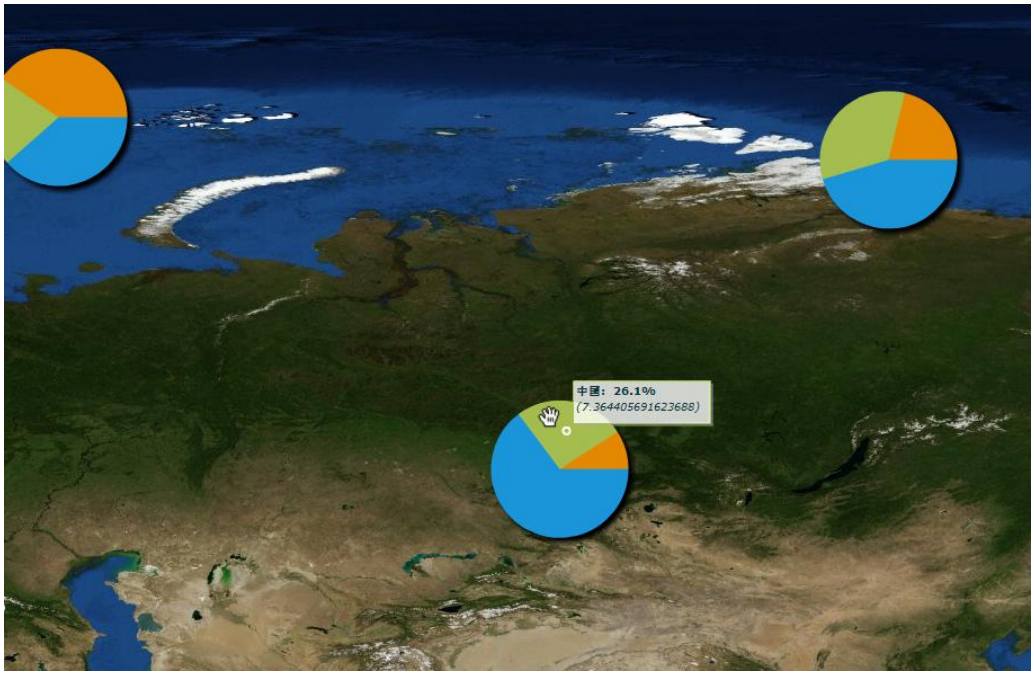


图 32　InfoSymbol 实现的专题图效果

以下是对应的源代码：

```
<mx:Script>
 <![CDATA[
      private function init():void
      {
            var mp:MapPoint = new MapPoint( x, y );
            var g:Graphic = new Graphic(mp);
            g.attributes = new Object();

            var thematic:ArrayCollection = new ArrayCollection( [
```

```
                { Country: "美国", Gold: goldA },
                { Country: "中国", Gold: goldC },
                { Country: "俄罗斯", Gold: goldR } ]);

            g.attributes.thematic = thematic;
            this.thematicLayer.add(g);
        }
]]>
</mx:Script>

<esri:InfoSymbol id="infoSymbol" infoPlacement="center">
 <esri:infoRenderer>
        <mx:Component>
            <mx:PieChart id="chart"
                showDataTips="true" width="150" height="150"
                dataProvider="{data.thematic}">
                <mx:series>
                    <mx:PieSeries field="Gold" displayName="资源占有率"
                        nameField="Country"/>
                </mx:series>
            </mx:PieChart>
        </mx:Component>
 </esri:infoRenderer>
</esri:InfoSymbol>

<esri:Map>
    <esri:ArcGISTiledMapServiceLayer
    url="http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_Imagery_World_2D/MapS
    erver"/>
    <esri:GraphicsLayer id="thematicLayer" symbol="{infoSymbol}" />
</esri:Map>
```

在这个例子里 ,我们在程序初始化的时候生成了一些带有统计信息的 Graphic 添加到地图上 ,这些 Graphic 对象的 attributes 属性中都带有一个 thematic 集合来保存各个统计的对象，每个统计的对象包含两个字段：Country 表示国家，Gold 表示资源占有率，下面我们会在 InfoSymbol 的定义中再次看到这两个字段。

当定义好这些 Graphic 对象以后，我们就可以把它们添加到设置了 InfoSymbol 符号的 GraphicLayer 上了。在 InfoSymbol 的定义中，我们可以看到，在这个

InfoSymbol 中添加了一个饼图组件 PieChart，这个饼图的 dataProvider 属性绑定的是{data.thematic}，它代表的其实就是 Graphic 对象的 attributes 属性的 thematic 对象。你可以简单地这样认为：InfoSymbol 中的 data 代表的就是其对应的 Graphic 对象的 attributes 属性。

既然在 InfoSymbol 中可以获得 Graphic 的属性信息，那么根据 Graphic 的属性信息来绘制不同的专题图就是水到渠成的事情了，上面的代码就可以做出图 32 的效果来。

3 实时的数据变化

Flex 应用可以和服务器建立 Socket 连接，从而真正实现从服务器向客户端“推送”数据，这使构建真正的实时监控平台等应用成为可能——当然，如果你采用定时从服务器刷新数据这种方式来构建“准”实时的应用，那么从 ArcGIS Flex API 的角度来说也没什么区别。在这个小节中我们主要来看一下当刷新数据的时候我们应该怎么改变原先 Graphic 对象的位置，以及在构建实时的应用时需要注意的一些特殊的细节。

当我们接收到数据，从而需要更新某个对象的位置或属性的时候，首先需要从一个 GraphicLayer 上获得那个特定的 Graphic（如果获取不到则新建）。由于所有 Graphic 都是 GraphicLayer 的子对象，因此，我们可以通过 GraphicLayer 的 getChildAt 或 getChildByName 方法来获取一个 Graphic。

更改这个 Graphic 的位置其实就是更新这个 Graphic 的 geometry 属性，把这个属性更新以后，这个 Graphic 会自动重绘到新的位置去。

一般的实时监控应用可能会有比较多的 Graphic，因此下面我们就几个比较重要的

细节做点说明。

首先是资源嵌入的问题，如果你使用了 PictureMarkerSymbol 之类的符号，它将会使用一个图片资源进行渲染，一旦你没把这个图片资源嵌入到 SWF 中去，每次使用这个 Symbol 的时候应用都会从服务器动态获取这个图片再标注到地图上，因此这里会额外存在一个加载的过程，而且这个图片绘制的效果会比较差。因此，最好把使用到的资源图片等都嵌入到当前应用中来，这样可以美化绘制的效果，同时提升绘制的效率。

还有一个是 Graphic 的 autoMoveToTop 属性，这是一个默认为 true 的属性，这种情况当鼠标移动到这个 Graphic 对象上的时候，这个 Graphic 会自动被移动到所有 Graphic 对象的最顶端，当 Graphic 数量比较多、并且 Graphic 相互叠置比较多的时候，这会消耗比较多的资源进行计算。因此在这种情况下，最好把这个选项设为 false。

另外，对于浏览器中的 Flash Player 大概可以承受的 Graphic 的数量级，最好也心里有数。我之前做过一些相关的测试，如果 Graphic 之间没有相互叠置，那么大概在 5000 个静态的（不刷新）Graphic 之内用户的操作体验还是可以的——事实上如果 Graphic 太多了在屏幕上也很难看清楚。如果业务数据量实在非常大，而且想用 Graphic 来表示，那么最好还是考虑对数据做些处理，比如使用热点图来表示数据分布等方法。

下面是我实现的一个实时监控 Demo 的效果，这个 Demo 中监控了 1000 个不同的对象，服务器每 0.5 秒给客户端推送一次数据，每次推送的数据改变所有这 1000 个对象的位置。在这样的压力下，这个 Demo 的交互操作基本上没有延迟：

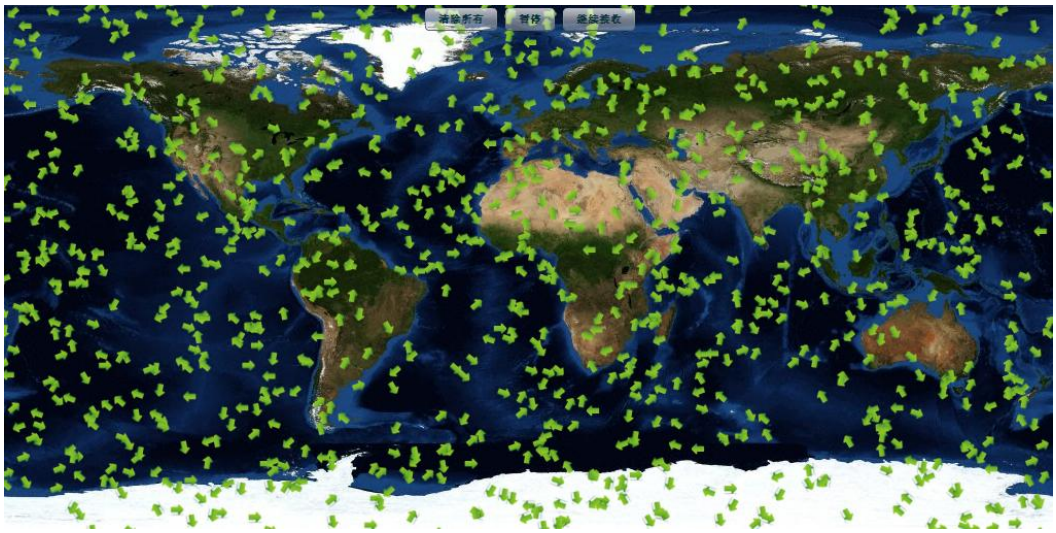


图 33　实时监控 Demo

4　符号扩展

在这个小节中，我们来看一下如何对符号进行扩展。虽然 ArcGIS Flex API 已经提供了非常多的符号供使用，但是如果你和我一样对符号有更多的需求，比如希望实现渐变效果、使用贝塞尔曲线来绘制符号等等，那么，这小节的内容对你将会有一些帮助。

首先，让我们从一个点符号入手。一个点符号可以继承 MarkerSymbol 来扩展，其中最关键的在于重写 MarkerSymbol 的 draw 方法，这个方法负责绘制这个符号，下面让我们看一下这个 draw 方法的原型：

```
override public function draw(sprite:Sprite, geometry:Geometry, attributes:Object,
map:Map):void
```

这个方法带 4 个参数，其中 sprite 是 Flash 中一个轻量级的容器 Sprite 对象，用于在其上进行绘图；geometry 是使用这个符号的几何对象；attributes 是 Graphic 的属性值；map 是 Map 组件的引用。

绘制符号需要做两件事情：首先需要给 sprite 定位、定尺寸；然后再在 sprite 中绘制图形。下面的代码使用渐变填充绘制了一个圆形符号：

```
var half:Number = size/2;

//定位、定尺寸
sprite.x = toScreenX(map, mp.x);
sprite.y = toScreenY(map, mp.y);
sprite.width = size;
sprite.height = size;

//绘制图形
sprite.graphics.beginGradientFill(GradientType.RADIAL, colors, alphas, ratios);
sprite.graphics.drawCircle(0, 0, half);
sprite.graphics.endFill();
```

下面是这个扩展符号的效果：

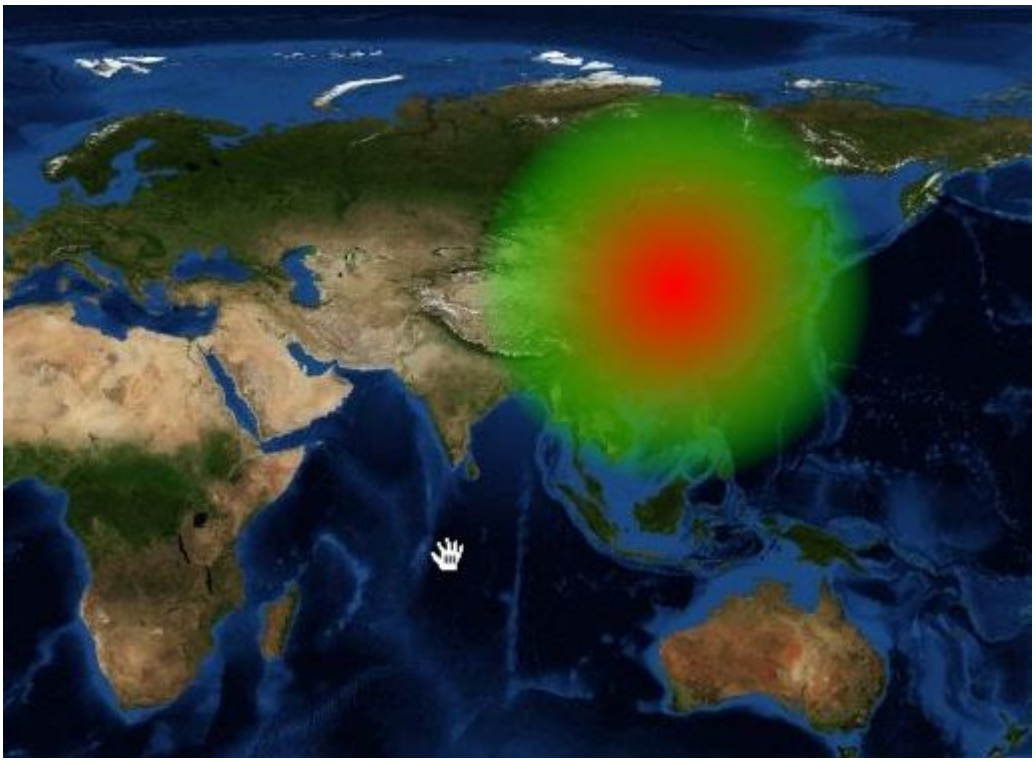


图 34　渐变填充的扩展符号

下面让我们再来看线符号的扩展，和点符号类似，线符号需要继承 LineSymbol。实现的方法和上面类似，主要是重写 draw 方法，下面是我实现的一个绘制曲线行军箭

头符号的效果：

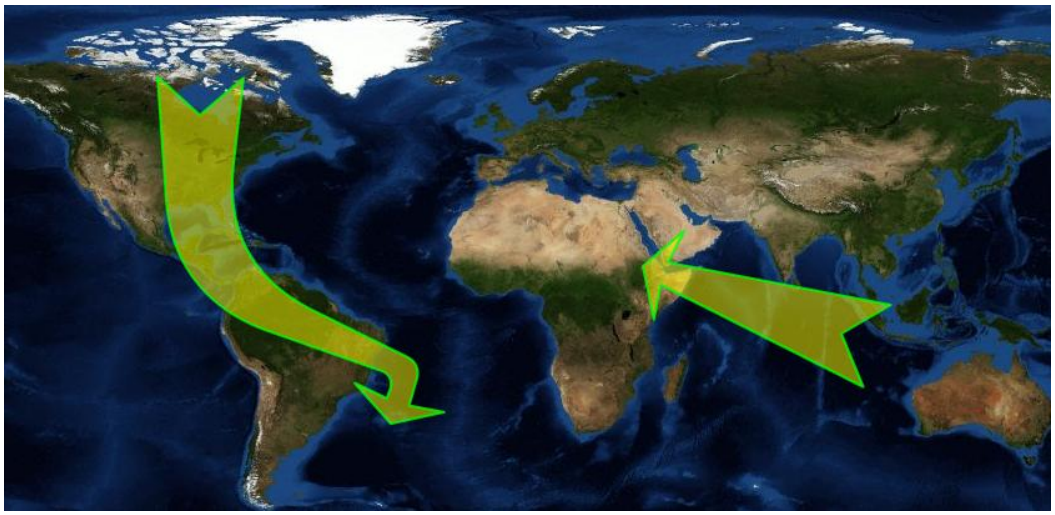


图 35　扩展的行军箭头符号

三 与其它系统的整合

1 与 JavaScript 交互

Flex 可以触发页面上的 JavaScript 函数；同时，页面上的 JavaScript 也可以触发 Flex 中的函数。因此，Flex 可以和 JavaScript 双向交互，这使 ArcGIS Flex API 应用可以以非常独立的方式嵌入到某个页面。

在我做的某个项目中，原先客户已经有自己的 MIS 系统，然后希望在其中加入 GIS 的功能。由于原来的 MIS 系统已经开发结束并具备了完整的功能，查询信息都能输出到页面的表格中，因此，给这个客户设计的解决方案就是采用 ArcGIS Flex API，将 GIS 功能封装到 Flex 中，数据交互则通过 JavaScript 进行。

我们从其中一个功能可以大致了解这种方案的实现和优点：刷新报警点。用户原先的报警点是输出到当前页面的表格中的，我们改写了 JavaScript，使其在刷新报警点的时候，调用另外一个函数把这个表格中的所有数据发送到 Flex 中去，然后由一个 GraphicLayer 来负责绘制。

下面是刷新报警点的 JavaScript 函数：

```
function addPoints()
{
    var map = document.getElementById("Map");
    map.clearHotPoints();

    var table = document.getElementById("DataGrid");
    var tBody = table.tBodies[0];
    var rows = tBody.rows;
    for(var i=1; i<rows.length; i++)
    {
        var row = rows[i];
        var attributes = new Array();
        for (var j=0; j<row.cells.length; j++)
```

```
            {
                attributes[j] = row.cells[j].firstChild.nodeValue;
            }
            map.addHotPoint(attributes);
        }
    }
```

在这个函数中出现了两个 Flex 中的方法，一个是在函数调用开始清空原先数据的 clearHotPoints 方法 ;另一个是添加报警点的 addHotPoint 方法。这两个方法都在 Flex 应用中如下定义：

```
private function init():void
{
 //供 JavaScript 调用
 ExternalInterface.addCallback("clearHotPoints", clearHotPoints);
 ExternalInterface.addCallback("addHotPoint", addHotPoint);
}

private function clearHotPoints():void
{
 this.hotPointLayer.clear();
}

private function addHotPoint(attributes:Array):void
{
 var pt:MapPoint = new MapPoint(attributes[0], attributes[1]);
 var g:Graphic = new Graphic(pt);
 g.attributes = attributes;
 this.hotPointLayer.add(g);
}
```

反过来，当用户在 Flex 中浏览到新的地图范围时，需要刷新数据以获得新的报警点，这是就需要从 Flex 调用 JavaScript 函数：

```
private function refresh():void
{
 //调用 JavaScript 函数
 ExternalInterface.call("refreshHotPoints", xmin, xmax, ymin, ymax);
}
```

这个 refreshHotPoints 就是页面中负责刷新报警点的 JavaScript 函数:

```
function refreshHotPoints(xmin, xmax, ymin, ymax)
```

```
{
    //原 MIS 系统的查询
}
```

2 HTTPService 与 WebService

Flex 通过 HTTPService 类从 Web 服务器获取数据，通过 WebService 类从 Web Service 获取数据。虽然这两个类与 ArcGIS Flex API 没有关系，但是，它们可以让你的 Flex 应用和原先可能已经存在的 Web 应用进行整合，融入除了 ArcGIS Server 提供的空间数据之外还可能存在的其它业务数据。

HTTPService 通过 GET/POST 方式访问某个 URL 来获取数据，数据的类型没有任何的要求，你只需要在 Flex 应用中监听响应事件，对响应的数据做相应的处理即可。

下面的代码 POST 了一个点坐标到 Web 服务器保存，并对保存结果进行了处理：

```
<mx:Script>
 <![CDATA[
      private function addPoint():void
      {
            var mapPoint:MapPoint = graphic.geometry as MapPoint;
            var params:Object = new Object();
            params.x = mapPoint.x;
            params.y = mapPoint.y;

            this.httpAddPoint.send(params);
      }

      private function onAddPointResult(event:ResultEvent):void
      {
            var str:String = event.result as String;
            var json:Object = JSON.decode(str);
            //对 JSON 数据的处理
      }
 ]]>
</mx:Script>
```

```
<mx:HTTPService id="httpAddPoint" url="{ROOT_URL+'/point.php?action=add'}"
method="POST" showBusyCursor="true" result="onAddPointResult(event);" />
```

HTTPService 的 url 属性是 GET/POST 请求的地址，它可以是任何 Web 服务器的 URL。在这里我使用了一个 PHP 应用，它的 point.php 页面中实现了添加一个点的 x、y 坐标到数据库的功能，当在 Flex 应用中使用 HTTPService 的 send 方法发送数据到这个页面上时，PHP 应用会保存这个点到后台数据库，同时返回一个 JSON 格式的保存结果。Flex 应用监听到这个 ResultEvent 以后就可以根据结果来进行不同的处理了。

对于 HTTPService 来说，如果是从原有的 Web 应用上构建 Flex 应用，或许你还需要做些扩展，因为原先的 Web 应用可能输出的是 HTML 页面，但是 Flex 应用只需要 XML、JSON 等格式的数据；但是对于原有的 Web Service 则不存在这个问题，下面让我们看一下 Flex 中 WebService 类的使用：

```
<mx:WebService id="ws"
 wsdl="http://livecycledata.org/services/ProductWS?wsdl"
 useProxy="false" showBusyCursor="true"/>

<mx:DataGrid dataProvider="{ws.getProducts.lastResult}" >
    <mx:columns>
        <mx:DataGridColumn dataField="productId" headerText="编号"/>
        <mx:DataGridColumn dataField="name" headerText="名称"/>
        <mx:DataGridColumn dataField="price" headerText="价格"/>
    </mx:columns>
</mx:DataGrid>

<mx:Button label="获取产品信息" click="ws.getProducts()"/>
```

这里我们使用了一种不同的方式来处理服务器返回的结果——数据绑定。虽然监听 ResultEvent 事件提供了很大的灵活性，但是对于很简单的业务逻辑（比如从服务器取数据填充到表格），我们也可以直接把 HTTPService 或者 WebService 的 lastResult 绑定到某个组件上（比如 DataGrid）。上面的这个示例中，当我们点击按钮调用 Web

Service 的 getProducts 操作后，服务器返回的最新数据就会自动绑定到 DataGrid 显示给用户——这样的代码能让逻辑简单的业务显得十分优雅。

通过 HTTPService 和 WebService，我们可以很容易地在原有的 Web 应用基础上把功能移植或者添加到 Flex 应用中来，从而大大减少开发量。

3 RemoteObject 与 BlazeDS

RemoteObject 功能类可以通过 AMF（Action Message Format）数据格式来访问远程应用服务器上的类(Java、.Net、PHP 等)。AMF 是一种压缩的二进制数据格式，专门用于 ActionScript 与服务器端通讯，其通讯效率比较高。总的来说，RemoteObject 可以使你专注于 Flex 客户端开发和服务器端功能开发，而隐藏掉服务器提供 Web 功能的细节。

在这个小节里我们主要针对 Java 来做详细的说明，因为 Adobe 公司针对 Java 发布了一个很好的开放源代码的服务组件 BlazeDS，我们的 Flex 应用通过 BlazeDS 的帮助就可以无缝地调用远程服务器上的 Java 方法提供的功能。

关于如何配置 Flex 应用使其支持 J2EE，如何在其中加入 BlazeDS 支持，如何进行开发调试，在这里就不详细叙述了。我们这里主要了解 Flex 应用如何调用服务器端的 Java 方法及其效果。

下面是一个从数据库获得污染节点并计算污染影响范围的例子。首先，我在服务器端编写了一个类 Wind 用以获得污染节点并通过风力参数来计算其影响范围：

```
package wuyf;

public class Wind
{
```

```
private Connection conn = null;

public Connection getConn()
{
	if (conn==null)
	{
		try
		{
			Class.forName("org.postgresql.Driver");
			String url = "jdbc:postgresql://localhost:5432/sde" ;
			conn = DriverManager.getConnection(url, "sde" , "sss" );
			conn.setAutoCommit(false);
		}
		catch(Exception e)
		{
			System.err.print(e);
		}
	}

	return conn;
}

public ArrayList<HashMap<String, String>> getEffectZones(double xMin, double yMin,
double xMax, double yMax)
{
	ArrayList<HashMap<String, String>> result = new ArrayList<HashMap<String,
String>>();

	if ( this.getConn()==null )
		return result;

	try
	{
		String sql = "select *.....from sde.wind";

		Statement st = this.getConn().createStatement();
		st.setFetchSize(0);
		ResultSet rs = st.executeQuery(sql);
		while (rs.next())
		{
			HashMap<String, String> map = new HashMap<String, String>();
			map.put("shape", rs.getString("ST_AsGeoJson"));
			map.put("velocity", rs.getString("velocity"));
			map.put("direction", rs.getString("direction"));
```

```
                result.add(map);
            }
            rs.close();
            st.close();
        }
        catch(Exception e)
        {
            System.err.print(e);
        }

        return result;
    }
}
```

要想这个类可以在 Flex 应用中通过 RemoteObject 可以访问到，我们需要在“remoting-config.xml”文件中添加这个类的定义：

```
<destination id="Wind">
 <properties>
    <source>wuyf.Wind</source>
    <scope>application</scope>
 </properties>
</destination>
```

下面我们就想在 Flex 应用中调用 Wind 类的 getEffectZones 方法来得到污染影响范围，下面让我们来看一下 Flex 中相应的代码：

```
<mx:Script>
 <![CDATA[
    private function getEffectZones():void
    {
        var extent:Extent = map.extent;

        var args:Array = new Array();
        args.push(extent.xmin);
        args.push(extent.ymin);
        args.push(extent.xmax);
        args.push(extent.ymax);

        var opt:AbstractOperation = this.roWind.getOperation("getEffectZones");
        opt.addEventListener(ResultEvent.RESULT, onGetEffectZonesResult);
        opt.arguments = args;
        opt.send();
```

```
        }

        private function onGetEffectZonesResult(event:ResultEvent):void
        {
            var effectZones:ArrayCollection = event.result as ArrayCollection;
            for each (var effectZone:Object in effectZones)
            {
                try
                {
                    var velocity:int = effectZone.velocity;
                    var direction:int = effectZone.direction;
                    var strShape:String = effectZone.shape;

                    //根据结果生成 Graphic

                    windLayer.add(g);
                }
                catch(err:Error)
                {
                    trace(err);
                }
            }
        }
    ]]>
    </mx:Script>

    <mx:RemoteObject id="roWind" destination="Wind" />
```

这里 RemoteObject 的 destination 属性对应的是在“remoting-config.xml”中配置的 Java 类的 id，这时，你就可以近似把这个名为 roWind 的 RemoteObject 对象当成 Wind 这个 Java 类了。通过 roWind.getOperation("getEffectZones")你就可以“获得”Wind 类的 getEffectZones 方法(实际上是一个 Flex 中的 AbstractOperation 对象），这个方法可以是带参数的。最后，通过调用 AbstractOperation 对象的 send 方法，你就相当于调用了服务器端 Wind 类的 getEffectZones 方法，接下来的事情就是去监听相应的 ResultEvent 事件从而去处理服务器端 Java 类返回的结果了。

下面是通过 RemoteObject 调用该 Java 类并返回的结果：

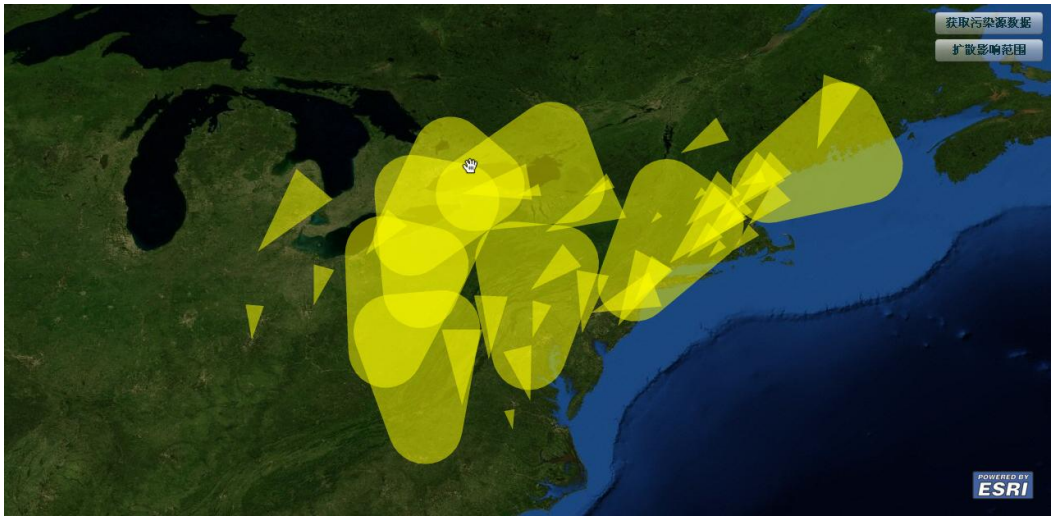


图 36　RemoteObject 的使用

4　通过 BlazeDS 调用 AO

Flex 应用是客户端应用，本身肯定是不能调用 AO 的，所谓调 AO 也就是将 Flex 端的数据发送到服务器上使用 AO 进行处理。我经常被问的一个问题就是：到底能不能用 ArcGIS Flex API 进行数据编辑，以前我一般回答不推荐使用 Flex 进行编辑。如果光从技术角度讲，我也曾经使用过 WFS-T、GP 服务、ADF 等方法实现了编辑，所以实在要问我应该怎么在 Flex 中做数据编辑，我一般会说，那就自己做个 Web 应用去接收 Flex 发来的数据再解析、保存吧。

现在有了 BlazeDS，上面的过程就可以被简化了。从上一小节我们已经知道如何使用 BlazeDS 来调用服务器端的 Java 方法，那么顺理成章，我们接着在这些方法中再调用 AO[1]就可以实现保存编辑等功能了。

比如我们在 ArcGIS Flex API 中使用 Draw 工具绘制了一个 MapPoint，随即要把

[1] 注意在服务器应用中添加 arcojects.jar 包的支持

这个点保存到服务中，那么首先我们需要把这个 MapPoint 对象作为一个参数通过

BlazeDS 的 AMF 发送到服务器，在服务器端我们实现了这样一个 Java 类：

```
public class AGSServer
{
	private IServerContext serverContext = null;

	public IServerContext getServerContext()
	{
		if ( serverContext==null )
		{
			try
			{
				ServerInitializer initializer = new ServerInitializer();
				initializer.initializeServer("*", "wuyf", "pwd");
				ServerConnection conn = new ServerConnection();
				conn.connect("localhost");
				IServerObjectManager som = conn.getServerObjectManager();
				serverContext = som.createServerContext("Demo/demo", "MapServer");
			}
			catch (Exception ex)
			{
				ex.printStackTrace();
			}
		}

		return serverContext;
	}

	public boolean addPoint(ASObject asObj)
	{
		boolean result = false;

		try
		{
			IServerContext serverContext = this.getServerContext();
			IServerObject so = serverContext.getServerObject();
			MapServer mapServer = (MapServer)so;
			IFeatureLayer feaLayer =
				(IFeatureLayer)mapServer.getLayer(mapServer.getMapName(0), 0);
			IFeatureClass feaClass = feaLayer.getFeatureClass();
			IFeature fea = feaClass.createFeature();
			IPoint pt = AGSFlexToAOUtil.MapPointToAO(asObj, serverContext);
```

```
                fea.setShapeByRef(pt);
                fea.store();
                mapServer.refreshServerObjects();

                result = true;
            }
            catch (Exception ex)
            {
                ex.printStackTrace();
            }

            return result;
        }
}
```

上面的代码主要功能就是在我本机的“Demo/demo”这个地图服务中使用 AO 接口保存点要素。其中 addPoint 方法是保存点要素的主要方法，其参数 asObj 对应的是 Flex 中的 MapPoint，asObj 是一个 ASObject 对象，它是一个继承自 HashMap 的类，所有 Flex 中的结构都会被映射成一系列的键值对应，然后发送到服务器。我们根据自己的需要，通过对 ASObject 对象的解析就可以得到相应的 AO 对象了。

这里我还新建了一个 AGSFlexToAOUtil 类专门用于转化 ASObject 对象到 AO 对象，MapPointToAO 方法就是负责转化 Flex 中的 MapPoint 到 AO 的 IPoint 的，这个方法的代码如下：

```
static public IPoint MapPointToAO(ASObject asObj, IServerContext scxt) throws Exception
{
    Point pt = (Point)scxt.createObject(Point.getClsid());
    double x = Double.valueOf(asObj.get("x").toString());
    double y = Double.valueOf(asObj.get("y").toString());
    pt.putCoords(x, y);
    return pt;
}
```

类似的，我还分别构造了 PolylineToAO、PolygonToAO 方法用以转化 Flex 传来的 Polyline、Polygon 对象为 AO 对象。下面是在 Flex 中对地图服务图层进行编辑并

保存后的效果：



图 37　通过调用 AO 保存客户端编辑

III、 综合场景演示

1 某系统演示

下面是使用 ArcGIS Flex API 开发的一个输电输气管道监控管理系统的 Demo，让我们试着从这个 Demo 去直观地了解在面对具体的业务时，ArcGIS Flex API 可以发挥什么样的作用。

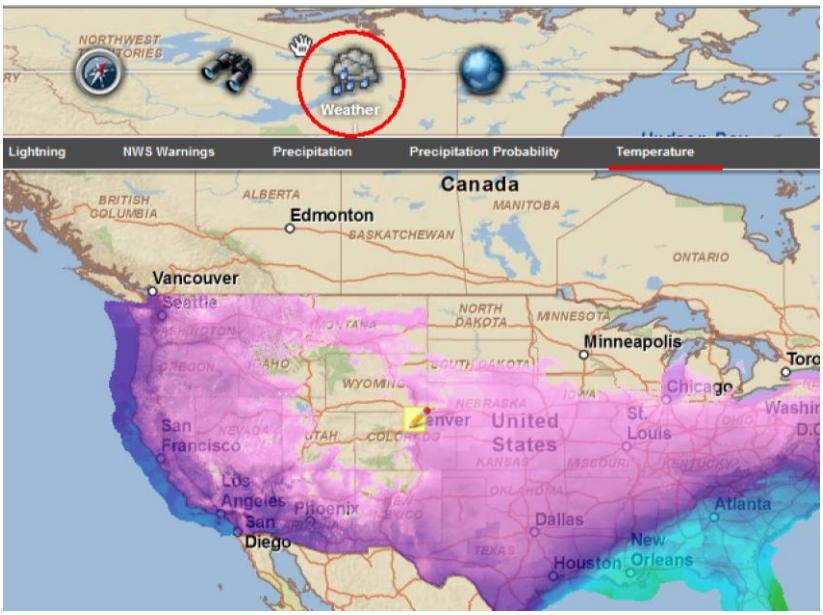


图 38　查看温度分布

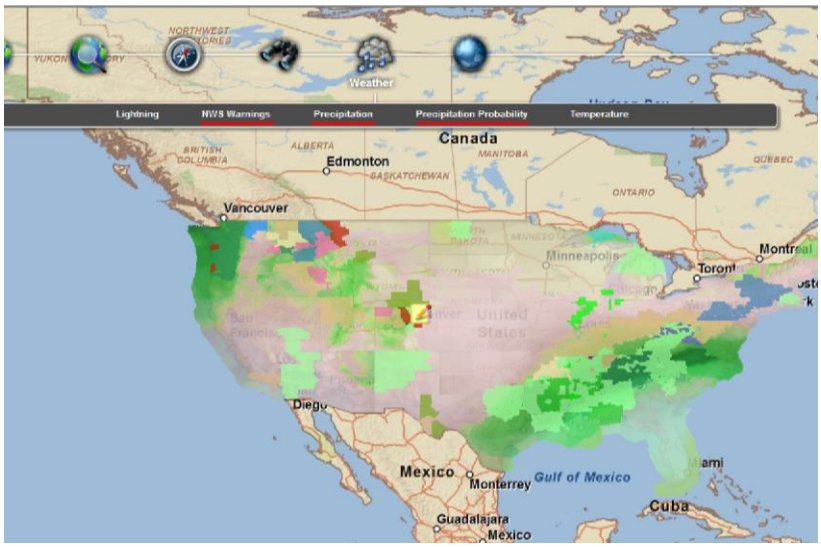


图 39　降水、美国国家气象局警报

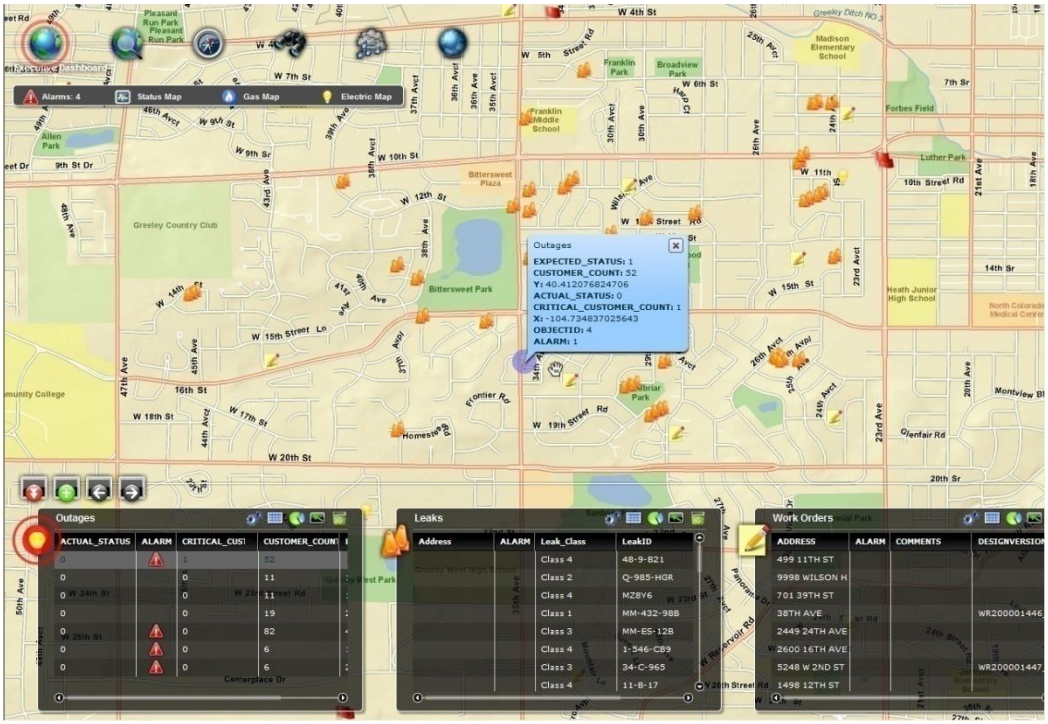


图 40　查看输气管道的泄漏情况



图 41　查看输气泄漏的统计情况

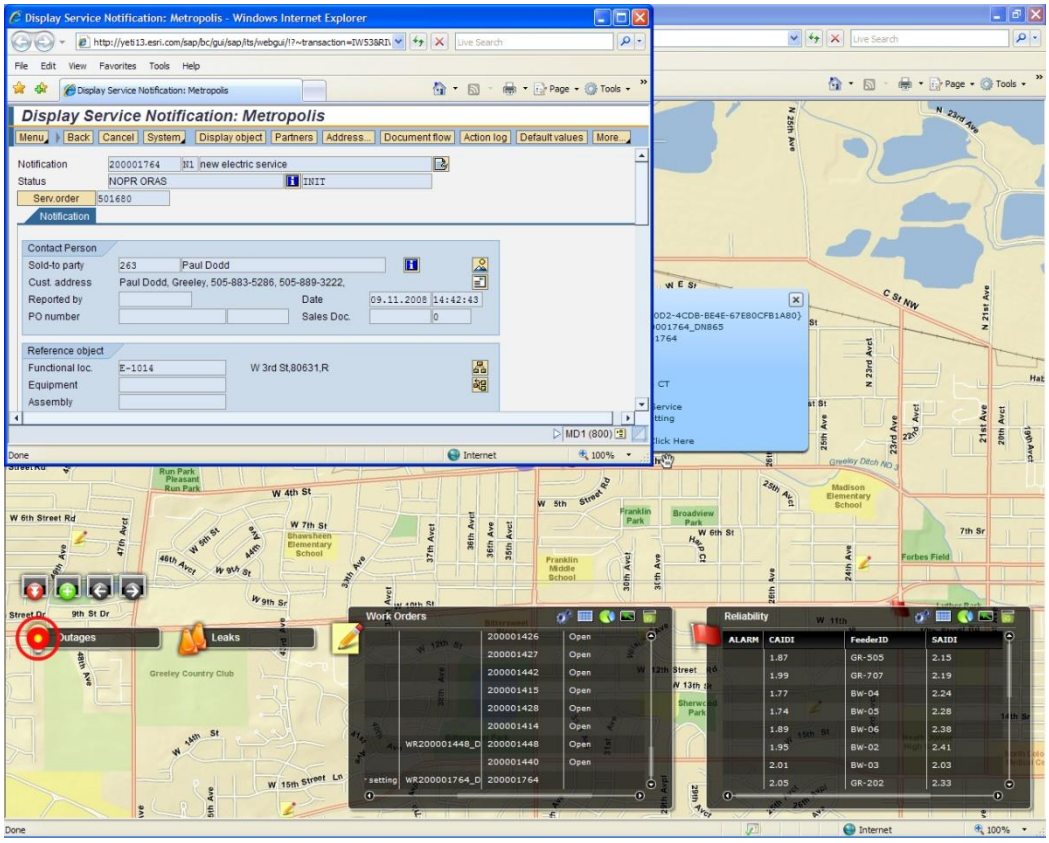
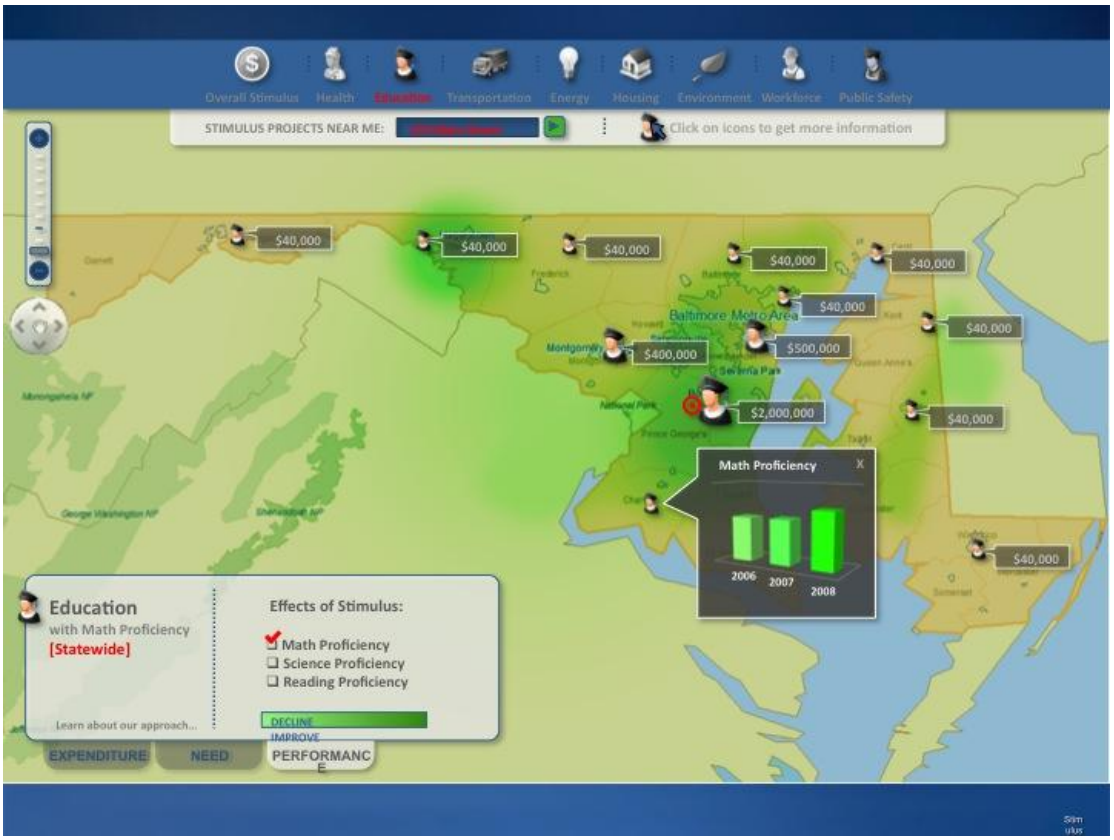


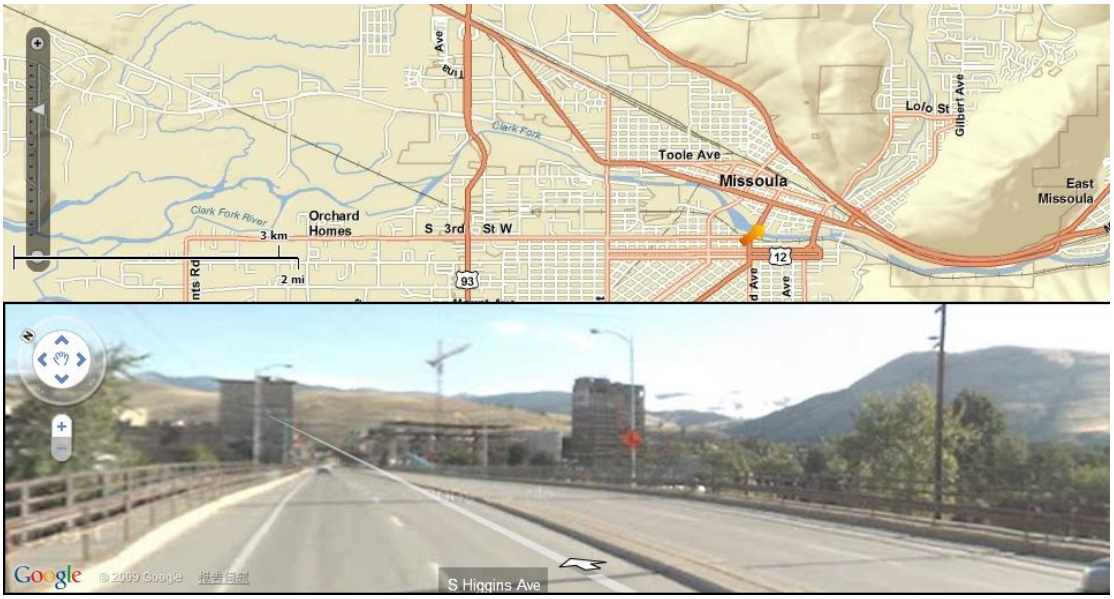
图 42　查看调度指令并可在新窗口中查看外部 MIS 系统

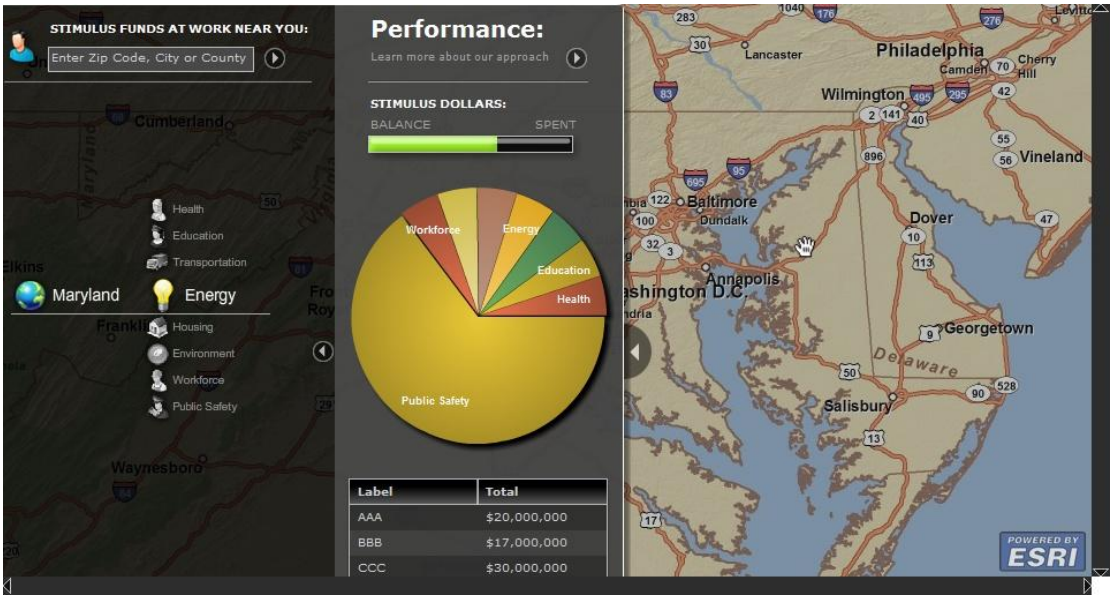


图 43　查看实时的道路交通情况

2 其它一些案例


Overall Stimulus
Health
Education
Transportation
Energy
Housing
Environment
Workforce
Public Safety
STIMULUS PROJECTS NEAR ME:
Click on icons to get more information
$40,000
$400,000
$500,000
$2,000,000
Baltimore Metro Area
Math Proficiency
2006
2007
2008
Education
with Math Proficiency
[Statewide]
Learn about our approach...
Effects of Stimulus:
Math Proficiency
Science Proficiency
Reading Proficiency
DECLINE
IMPROVE
EXPENDITURE
NEED
PERFORMANCE


Toole Ave
Missoula
East Missoula
Clark Fork
Clark Fork River
Orchard Homes
S 3rd St W
3 km
2 mi
93
12
Lolo St
Gilbert Ave
Google
© 2009 Google
S Higgins Ave


STIMULUS FUNDS AT WORK NEAR YOU:
Enter Zip Code, City or County
Cumberland
Health
Education
Transportation
Maryland
Energy
Housing
Environment
Workforce
Public Safety
Waynesboro
Performance:
Learn more about our approach
STIMULUS DOLLARS:
BALANCE
SPENT
Workforce
Energy
Education
Health
Public Safety
Label
Total
AAA
$20,000,000
BBB
$17,000,000
CCC
$30,000,000
Lancaster
Philadelphia
Camden
Cherry Hill
Wilmington
Vineland
Baltimore
Dundalk
Dover
Annapolis
Georgetown
Delaware
Salisbury
POWERED BY ESRI