



ArcGIS Server 9.3 REST 基础教程

**THE BEST OF REST Using REST
in ArcGIS Server 9.3**

ESRI 中国(北京)有限公司

关于本书

本书不是一本表述性状态转移应用程序接口（**REST API**）的完整参考书，相反，本书只是展示**REST API**的一些基础示例和最佳实务。本书内容反映了**ESRI**公司相关**REST**项目团队的最新工作，没有该团队工作人员的奉献精神 and 艰苦工作，本书不可能成型。

书中的提示，指南，编码样本包括了来自用户和各领域专家的工作。

在附录**A**的资源中可以找到更多的帮助信息及指南。

本书的翻译工作由中科院地理所郭兆成完成，校对工作由**ESRI**中国(北京)有限公司产品技术部汪维莉完成。

目 录

引言	1
第一章: REST——轻松创建 Web 服务	3
1.1 Web 服务和面向服务架构	3
1.2 Web 服务类型	4
1.2.1 基于 SOAP 的 Web 服务	4
1.2.2 REST 风格的 Web 服务	5
1.3 REST 类型 Web 服务的优势	5
1.4 ESRI 推进对 REST 的支持	8
第二章: ArcGIS REST API	11
2.1 支持的服务类型	11
2.1.1 地图服务	12
2.1.2 地理编码服务	13
2.1.3 地理处理 (Geoprocessing) 服务	13
2.1.4 几何服务	14
2.1.5 图像服务	15
2.1.6 其他服务	15
2.2 创建 GIS 资源	15
2.3 资源发布为服务	19
2.3.1 使用 ArcCatalog 发布服务	19
2.3.2 利用 ArcGIS Server 管理器发布服务	22
2.4 浏览服务	25
2.4.1 浏览服务器内容	25
2.4.2 查看服务空间范围	28
2.4.3 测试 REST 服务	29
2.4.4 获取开发信息	32
2.4.5 如何在开发中使用服务目录的示例	32
2.4.6 支持的输出格式	33
2.5 管理服务	35

2.5.1 启动, 停止, 暂停服务	35
2.5.2 管理服务缓存	37
2.5.3 启用和禁用服务目录	39
第三章: REST 应用	41
3.1 易于使用	41
3.2 不编程使用 REST API	43
3.2.1 ArcGIS Server JavaScript 地图浏览器	43
3.2.2 ArcGIS Explorer	44
3.2.3 ArcMap	45
3.2.4 微软虚拟地球	46
3.2.5 谷歌地图	46
3.2.6 谷歌地球	47
3.2.7 利用 Web 浏览器	47
3.3 基于浏览器端编程使用 REST API	48
3.3.1 JavaScript	48
3.3.2 利用 Flex	57
3.3.3 利用 Silverlight	63
3.4 通过服务器端和桌面编程使用 REST API	66
3.4.1 利用 Python	67
3.4.2 利用 ASP.Net	71
3.4.3 利用 Java	72
第四章: 优化方法	76
4.1 保证 REST 服务安全	76
4.2 改进性能	77
4.2.1 缓存	77
4.2.2 压缩	78
4.2.3 图像格式	78
4.2.4 响应格式	80
词汇表	82
附录 A: REST 资源	92

引言

本书《ArcGIS Server 9.3 REST基础教程》是专门面向2008年首次发布的最新ArcGIS软件而设计，书中内容着重针对表述性状态转移应用编程接口（REST API）进行论述和展开。

《ArcGIS Server 9.3 REST基础教程》一书是ESRI公司软件和产品开发工程师利用业余时间协同努力撰写的成果，书中介绍了他们对这些正在成为通用的API的第一手知识。ESRI中国感谢他们为开发者和最终用户做出的努力工作，本书通过活灵活现的实用示例使REST有关知识和所有功能跃然纸上，读者和用户能够一目了然。

ESRI应用开发服务部和专业服务组的高级GIS开发工程师Pinde Fu撰写了本书的第一章，他首先介绍了REST的功能和应用前景，通过一个网址就可以使GIS拥有真正的用户友好界面。在第一章中，撰写者还介绍了REST的历史渊源，论述了REST能够将GIS应用普及到从技术专家到GIS初学者的巨大潜力。

产品工程师Sarah Osborne和开发工程师Keyur Shah，共同奠定了第2章“ArcGIS REST API”的基础。Sarah和Keyur站在GIS开发者应用的角度，提供了在各种计算机语言使用REST的具体例子，其中着重介绍了如何通过利用JavaScript API来使用REST的内容。本章接下来的部分中，Pinde和高级软件工程师Al Pascual详细介绍了如何使用URL创建和发布REST服务。

产品工程师Jeremy Bartley，在繁忙的日常事务中抽出时间撰写了第三章“All you need is a URL”的基础内容。本章内容介绍了如何通过编程或不通过编程的两种方式利用REST的具体步骤。Pinde和Al再次撰写了本章中的一些细节内容，其中Pinde负责撰写使用简单网页或者编程语言来利用REST服务的步骤向导。实际上，Pinde在本书撰写过程中承担了很多整理工作，将本有关的众多复杂内容灵巧地编排在一起，成为一本完整的REST手册。

通过提供在ArcGIS Server 9.3利用REST的实用小技巧 and 最优实务，Al在第4章中很好地总结了如何才能最有效地利用这一技术。

在ArcGIS Server 9.3版中，JavaScript和REST API极大地推进了Mashing 概念的发展，以及整合ArcGIS内容和其他GIS内容到网络中。通过引入JavaScript和REST API以及

新兴的“Mashup”功能，ArcGIS Server 9.3旨在推进互联网上大量的地理信息数据和工具的共享，使之真正成为全民型的地理信息系统。

感谢Pinde, Al, Sarah, Keyur和Jeremy等人的努力，帮助本书尽可能快地完成和分发给大家。开发小组还提供了书中的示例编程，读者可以使用这些示例编程了解REST的使用，其中Jeremy负责Python编程示例，Silverlight负责C#编程示例，Pinde负责Java编程。通过这种团队的合作，我们开创“REST风格”的新模式，概述了ArcGIS Server最有潜力的REST应用。我们希望本书能够抛砖引玉，不仅吸引读者涉足REST技术，希望读者能够更加深入的使用REST。我们深信，利用REST这个支点，世界就在你的指尖。

本书附录词汇表同样汇集了更多开拓进取人们的辛勤工作，这些工作使读者能够更容易和深入地理解GIS。我们要特别感谢ESRI出版社编辑Mike Schwartz和Candace Hogan在上一本书《Implementing ArcGIS Server Systems: Configuration Basics and Best Practices》中对GIS一些名词的严格定义。词汇表中有关本书主题的词汇大多都被包括进本书附录的词汇表中。

本书词汇表还包括ESRI出版的，Tasha Wade和Shelly Sommer编辑的《A to Z GIS: An Illustrated Dictionary of Geographic Information Systems》一书中有关的名词定义。Pinde严格审查了本书词汇表中专业词汇的确切定义。最后，词汇表中有几个词条引自微软出版社2002出版的第五版微软电脑字典。

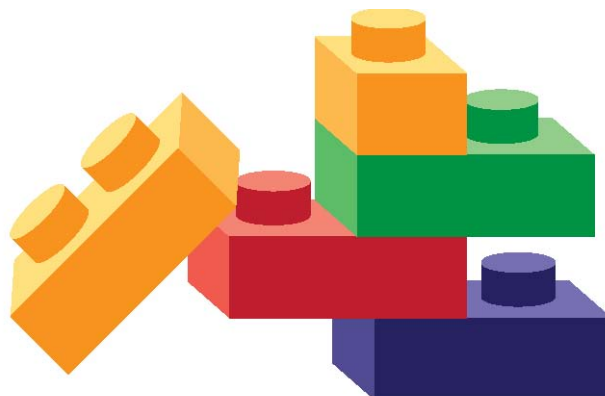
第一章：REST——轻松创建 Web 服务

REST(表述性状态转移)风格的Web服务已成为越来越流行的Web信息系统创建方法。虽然该术语的初始含义还没有标准的解释,但就其本质而言,REST很简单,只要使用网址,就可以很容易地创建、发布和使用“REST风格”的网络服务。

相比REST的前一个标准SOAP(简单对象访问协议),即基于SOAP标准的Web服务,REST风格的网络服务是轻量级的,使用非常简便、灵活。基于REST的这些优势,ArcGIS Server 9.3引入REST作为新型强大的功能,使用户能够没有任何困难地发布和使用Web服务。REST将成为各层次开发者利用ArcGIS Server创建自定义应用的通用方法。

1.1 Web 服务和面向服务架构

孩子们利用想象力使用积木来搭建自己的城堡,信息系统的发展史也是同样的道理。只不过信息技术人员将这些积木称为“组件”,计算机发明后,组件技术就一直不断得到发展。在20世纪60年代,早期的程序编译阶段,构建信息系统通常需要收集常用的函数或子程序组件。在20世纪90年代,大多数开发者采用了面向对象编程(OOP)的理念,对象(封装模块)就是被用来作为信息系统的构建模块。最近的十几年中,开发人员开始采用一种新的玩具模块,新的方式来使用这些模块。

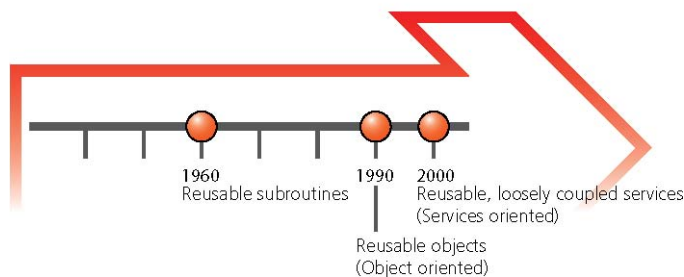


基于新的模块,Web服务,就必须使用一种新的系统规划——面向服务的架构(SOA)。

Web服务是互联网上基于标准互联网协议(超文本传输协议,

HTTP)使用的软件组件,可以实现组件最新的和改进的请求过程。Web服务可独立于平台和语言实现分布在Internet/Intranet上的应用程序或组件的无缝互操作。通过在ArcGIS Server 9.3中引入REST,我们可以通过网络轻松直观地实现GIS的无缝互操作。

从本质上讲,在简单的浏览器中使用REST服务可以实现所有的请求都如使用URL一样简单。在本章接下来的部分中,我们将讨论REST的所有强大功能及其灵活性。



对比传统的方法，Web服务有很多优势：

- 用户不必在本地机器安装执行服务所需的软件。例如，用户可以不安装地理信息系统软件，就可以进行 GIS 分析。
- Web 服务特别适合于复杂环境中、数据快速变化的操作；一个单一的具有中央副本的数据比多个用户计算机上的数据更容易维护。
- 软件授权和知识产权问题更容易得到解决。
- 系统客户端只需在特定的时间点，即可通过 Web 服务接受数据。

基于这些优势，Web服务已成为面向服务架构的理想实现方式。本质上讲，SOA只是一种计算方法，其所有的功能都是独立的，通过友好界面，松散耦合的服务可以按照特定序列被调用。通过提供整合基于位置独立的应用和平台异构的服务功能，Web服务和SOA正在成为流行的信息技术。

1.2 Web 服务类型

正如玩具积木有不同的形状和大小，Web服务也有不同的类型。总的来说，Web服务可分为以下三种主要类型：

- 基于 SOAP 的 Web 服务；
- REST 风格的 Web 服务；
- 其他方式的网络服务。

其他方式Web服务混合了上述两种类型的特点：一般使用XML发送请求，得到响应，而不是被嵌入到SOAP中。

1.2.1 基于 SOAP 的 Web 服务

SOAP是基于计算机网络扩展标记语言（XML）的协议。

1998年，SOAP开始被作为一个跨互联网形式的分布式组件对象模型（DCOM）或公共对象请求代理体系（CORBA）。在2003年，SOAP协议成为万维网联盟（W3C）的标准。最开始W3C将该术语称为“简单对象访问协议”，在2007年的1.2版的标准中改为当前的术语名词。SOAP经常结合使用Web服务描述语言（WSDL）来提供互联网上的网络服务，WSDL是一种基于XML的描述Web服务，以及如何访问Web服务的语言。

基于SOAP/WSDL的网络服务的优势在于：服务格式定义严格。其中的每一种方法，都需要输入参数、请求参数的类型和返回结果的类型，这些都需要在WSDL中严格定义。开发环境，如.NET或Java都提供基于SOAP的工具包，可以自动生成本地类，进而确保服务使用者保持与Web服务的交互。

但是，基于SOAP的Web服务是完全预先定义的，不容易被很多的开发者使用。在某些情况下，例如考虑了SOAP堆栈层上更多额外的Web服务（WS-*），SOAP协议就变的很复杂了。

但“REST风格”的Web服务是动态，易于使用和灵活的，并且不需要在客户端或者服务器端进行很多的工作。

1.2.2 REST 风格的 Web 服务

单纯就REST术语的出现而言，REST是Roy Fielding在其2000年的论文中首次提出的一种软件架构。具体地说，REST用来定义一个Web服务应用程序编程接口（API），通过HTTP来进行资源管理，例如CRUD（创建、读取、更新和删除）。

Roy Fielding指出，虽然REST架构专为大规模超媒体分发，但它并不是一种“专用”架构。目前，最具REST风格的Web服务可以简单为HTTP“Get”——即URL网址，同时也是最简单的利用Web服务请求的提出方式。

REST中，CRUD意味着创建/读取/更新/删除地图服务的子组件，也就是一个层。一个功能齐全的REST地图服务可以让使用者建立一个层，实现读取层，更新层，或删除层。

目前，ArcGIS Server REST API只允许用户读取层（查询层，并查看该层地图）。查询可以基于浏览器或在多种编程语言中实现，例如.NET,Java,JavaScript, Ruby, Python等。REST类型的Web服务遵循以下这些基本原则：

- 设定地址资源：可以使用一个网址访问任何资源。
- 通用接口：标准的 Put,Get,Post 和 Delete 操作都需要得到请求才能完成。
- 无状态消息：每个请求都是独立的，每个请求都必须包括自己的参数，来完成操作。
- 表达：请求结果的渲染可以通过多个形式实现，例如地图和影像，XML 和 JSON。

“利用REST，输入网址即可实现一切！”

1.3 REST 类型 Web 服务的优势

早期，人们认为SOAP协议将成为访问Web服务的最终方式。SOAP功能强大并且比较全面。但是，SOAP比较复杂且不如REST使用简单。

表 1.1 基于 REST 风格的 Web 服务和 SOAP 的 Web 服务比较

REST风格		SOAP
针对市场	从小规模到大规模IT系统建设，大市场	重大IT系统建设，大公司
程序员	地理学者与专业开发者	专业开发者

性能	90%	100%
成本	10%	100%
时间	10%	100%
风险	风险低	风险高

2002年，亚马逊公司发布了其电子商务的网络服务，由于意识到了“REST与SOAP各自的优缺点”（表1.1），亚马逊为其网络服务提供了SOAP和REST的接口。两年后，这个平台吸引了逾5万开发者的关注，但其中大多数开发者倾向于使用REST方法。事实上，百分之八十的请求是以亚马逊的REST Web服务为基础的，只有百分之二十是基于SOAP Web服务。这就表明，开发者更喜欢使用简单的REST。

REST类型的Web服务吸引力在哪里？就是在于其使用极其简单。REST类型的Web服务要求很少的编码工作量，能够减少很多不必要的工作。

例如，假设用户需要从ArcGIS Server地图服务器生成空间范围在185.33度以西，59.53度以东，74.08度以北，15.20度以南地区，JPG格式的800x500像素的地图，就可以访问：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/。

ArcGIS Server提供基于SOAP和REST接口，用户可以任意选择其一。使用基于SOAP接口，用户首先需要从WSDL中产生一系列的SOAP工具包：

（[http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/? WSDL](http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/?WSDL)）。随之，用户还需要研究SOAP接口的使用方法。最后，用户还需要利用某种编程语言编写代码，例如以下代码（以C#代码为例）：

```
MapService_MapServer mapservice = new MapService_MapServer();
mapservice.Url = "http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/";
MapServerInfo mapinfo = mapservice.GetServerInfo(mapservice.GetDefaultMapName());
MapDescription mapdesc = mapinfo.DefaultMapDescription;
ImageType imgtype = new ImageType();
imgtype.ImageFormat = esriImageFormat.esriImagePNG;
imgtype.ImageReturnType = esriImageReturnType.esriImageReturnURL;
ImageDisplay imgdisp = new ImageDisplay();
```

```
imgdisp.ImageHeight = 800;

imgdisp.ImageWidth = 500;

imgdisp.ImageDPI = 96;

ImageDescription imgdesc = new ImageDescription();

imgdesc.ImageDisplay = imgdisp;

imgdesc.ImageType = imgtype;

MapService_MapServer.EnvelopeN env = new MapService_MapServer.EnvelopeN();

env.XMin = -185.33;

env.XMax = -59.53;

env.YMin = 15.20;

env.YMax = 74.08;

mapdesc.MapArea.Extent = env;

MapImage mapimg = mapservice.ExportMapImage(mapdesc, imgdesc);
```

可以看到，使用SOAP的Web服务是非常不方便的。使用基于浏览器端编程语言的SOAP 类型Web服务，如JavaScript，通常比使用REST类型的Web服务更加困难。

如果使用REST类型的Web服务，完成上述任务将变的非常容易。用户不需要SOAP工具包。用户所有需要做的工作只是建立一个如下所示的网址：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/export?bbox=-185.33%2C15.20%2C-59.53%2C74.08&size=800%2C500&format=jpg&dpi=96&f=image

这个链接的响应就是返回用户想要的地图图片。重要的是，用户可以在各种计算机编程语言如.NET,Java,JavaScript,Flex代码中嵌入这个网址来生成所需要的影像地图，而且只需在Web浏览器中运行该段代码，就能看到所需要的地图。多么简单！

REST技术具有很多优点：

- 无需引入 SOAP 消息传输层，轻量级和高效率的 HTTP 格式可直接被应用。
- 灵活性和易用性。
 - ✧ 无需建立庞大的 XML 消息。只需建立一个网址字符串来访问 REST 服务。
 - ✧ 可以轻易地在任何编程语言中实现，尤其是在 JavaScript 中。使用 SOAP 与 JavaScript 的 Web 服务非常繁琐，但使用 REST 与 JavaScript 的 Web 服务就非常简单。

✧ REST 有利于促进 JavaScript 和 XML(AJAX)异步 Mashup 和 Web 2.0 应用的进一步发展。Mashup 能够聚合来自不同网站的资源结果。因为容易被调用、使用方便，REST 服务就成为实现 Mashup 的最优方法。

- 可以不使用任何编程语言就能访问服务，而只要使用 Web 浏览器或 ArcGIS Explorer。
- 更好的性能和缓存支持——REST 类型 Web 服务可以利用高速缓存控制，从而减少对带宽的需求。使用 REST 可以改善响应时间和改进用户体验。
- 可扩展性和无状态性——每个请求都是独立的。一旦被调用，服务器不保留任何会话，这样就可以更具响应性。通过减少事件后通讯状态的维护工作，提高了服务器的可扩展性。
- 易于被索引和发现——REST 网址能够被如谷歌，雅虎或 MSN 的搜索引擎索引，这使得它们更容易地被发现。不通过单独的资源挖掘机制就可以发现网址，例如 UDDI（通用描述，发现和综合机制）。

REST类型的Web服务简单、有效、直观且用户界面友好。在许多情况下，简单有效地使用REST技术优于使用复杂的基于SOAP的Web服务。使用REST对许多方面都非常有利：

- 对于服务提供者：利用 REST 可以降低创建服务的成本，降低托管和维护支持服务的费用。
- 对于服务使用者：REST 网络服务可以降低学习难度，减少构建地理信息系统应用所需要的时间和投入。
- 对于管理人员：REST 有许多非常理想的体系结构特性，可伸缩性、高性能、可靠性和可扩展性。这些特点很好地与现代商业环境相协调，这就要求技术解决方案必需与业务本身具有同样的适应性和敏捷性。

1.4 ESRI 推进对 REST 的支持

ESRI认为Web服务、面向服务的架构以及Mashup风格的应用程序开发是信息技术最重要的发展趋势。ESRI已经率先着手推动这一战略，进一步创建、发布、发现和使用地理信息系统网络服务。

尽管自早期的ArcGIS Server 版本发布开始，ESRI就一直支持基于SOAP的Web服务，但在ArcGIS Server 9.3中引入了创新的和强大的基于REST的Web服务架构。面向ArcGIS Server的REST API并不意味着取代SOAP API，而是对SOAP API的增强和改进。

ArcGIS Server REST API具有以下特点：

- 简单地通过一个网址可以访问所有资源。包括地图服务，单个地图层，以至所有地理数据和地理处理服务。

- 技术强大。支持各种业务，从执行地理处理请求，地理编码到几何运算，都可以访问金字塔和创建 KML。
- 基于 REST 的 Web 服务可以使用缓存服务，来更快地获取结果，更加容易和更加快速地获取地图。REST 利用高速缓存控制，来识别给定网址的内容是否经过修改，因此，用户就可以清楚知道是否有必要继续下载还是简单地使用已经下载的内容。
- 在 ArcGIS Server 中发布一个新的服务只是利用 REST API 自动地生成 HTML 网页。该网页不仅是“书签式”的浏览页面，而是可以被搜索引擎“索引”的。这意味着用户可以很容易从浏览器使用 Web 服务，或同样容易地从谷歌，雅虎，微软 Live 搜索或 ArcGIS Explorer 这种搜索引擎中发现 REST 类型的 Web 服务。
- REST API 支持的结果表达格式包括 HTML, Help, Image, JSON 和 KMZ 等所有流行的 Web 2.0 编程格式。
- 一切只是一个网址，Web 服务可以很容易地被一些语言和开发环境访问。支持的编程语言有 Java 和 .NET, Python, Ruby, Yahoo! Pipes, Perl 和 PHP 等。JavaScript 同样也是很容易被支持。使用各种编程语言实现 REST 的具体例子详见第 3 章。
- REST Web 服务还可以不通过任何编程就能访问。REST 可在各种 Mashup 环境下访问，包括 ArcGIS Explorer，谷歌地球，谷歌地图，微软虚拟地球。具体实例见第 3 章内容。
- REST 将 ArcGIS Server 先进的分析功能封装为一个简单的网址，并把这些功能扩展延伸到互联网，大大普及了 ArcGIS Server 的使用。
- REST 服务同其他服务一样安全，包括令牌。

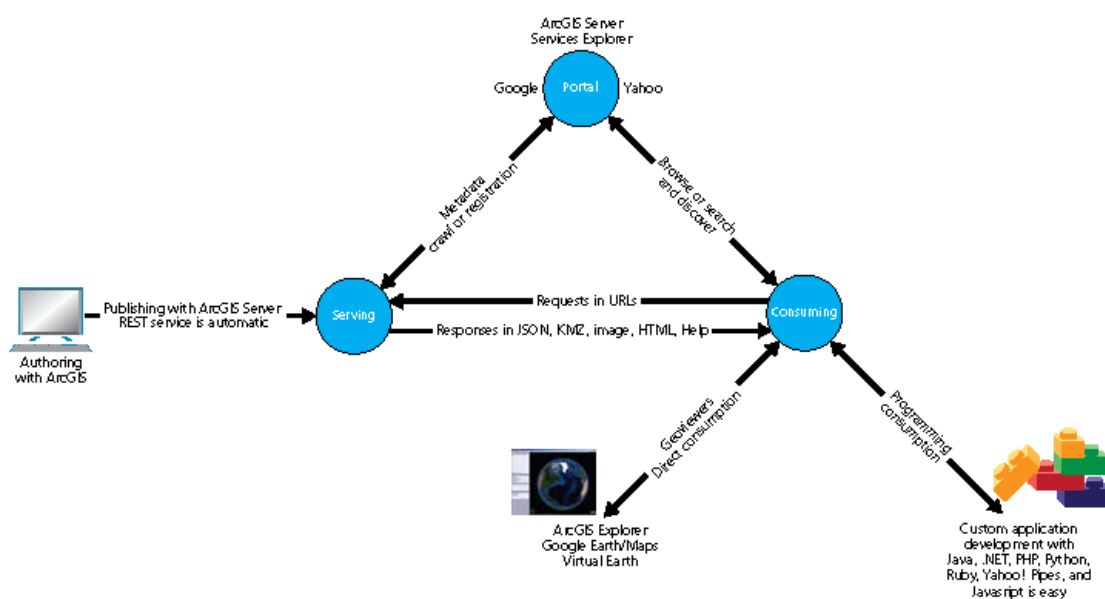


图 1.1 ArcGIS Server支持REST Web服务的体系

REST简单易用。对于那些已经使用过其他比较复杂类型服务的开发人员来说，使用

REST的简单程度令人耳目一新。REST的强大，就在于其接口很容易实现多种地理处理服务。

ArcGIS Server向用户提供了创建，发布和提供REST Web服务的方式（图1.1），同时自动生成服务元数据。服务元数据可以在ArcGIS Server服务浏览器中自动浏览。此外，服务元数据利用简单的HTML格式，这样就可以很容易地利用公共搜索引擎，例如谷歌，雅虎或微软Live搜索找到索引。用户可以直接或通过搜索引擎连接到REST服务。另外，用户还可以通过使用地理浏览器，不经过编程使用REST服务或通过各种不同的编程语言建立其自定义应用。

使用REST操作就会发现REST可以使用户有更多的时间关注操作业务本身，而不是在其使用技术方面。同时，各级开发人员可以从REST网络服务的简单易用中受益匪浅，激发开发者更多的聪明才智和创造力。

第二章: ArcGIS REST API

ArcGIS Server代表性状态转移应用程序编程接口, 提供了ArcGIS Server提供的地理信息系统服务的一个简单、开放的界面。ArcGIS Server发布的GIS服务都可以通过各级端点或统一资源定位器 (URL) 访问针对REST API开放的所有资源和操作。

ArcGIS Server允许用户创建, 发布和使用REST服务。本章的重点旨在创建和发布REST服务。下一章中将提供具体例子来说明如何基于不同的客户端使用REST API。

本章包括以下几个部分的内容:

1、 服务支持的类型——如果已经决定使用 REST 服务, 用户需要做的第一件事通常是决定哪些 ArcGIS Server 服务是自己所需要的。用户是否需要地图服务、地理编码服务、地理处理服务、几何服务、图片服务、地球模型服务 (Globe 服务) 或网络分析服务? 本章第一节内容着重说明 ArcGIS Server 支持的服务类型, 以及这些服务类型支持的操作。

2、 创建 GIS 资源——确定了需要创建的服务类型, 用户就要判断哪些地理信息系统资源需要创建。不同类型的服务需要不同类型的地理信息系统资源, 而这些资源分别可以在不同的 ArcGIS 模块中创建。本章第 2 节介绍了用户需要创建哪些类型的地理信息系统资源, 以及如何利用 ArcGIS 的模块创建这些 GIS 资源。

3、 发布资源与服务——创建资源以后, 用户就可以通过使用 REST API 将它们发布为 Web 服务, 如此一来这些服务就可以被 Internet 或 Intranet 客户端应用程序远程调用。本章第 3 节说明如何在 ArcCatalog 和 ArcGIS Server 管理器中发布服务。

4、 浏览服务——服务创建完成后, 用户可以在 ArcGIS Server 的服务浏览器中浏览服务和目录。服务目录格式是基于 ArcGIS Server REST API 的 HTML。通过服务浏览器, 用户可以浏览服务器内容, 查看可用的 GIS Web 服务, 获取对开发的有用信息, 甚至可以测试用户自定义的服务。本章第 4 节详细介绍了这方面的所有功能。

5、 管理服务——创建的服务需要进行管理, 用户可以停止, 启动或暂停这些服务, 更新缓存, 并且启用/禁用服务浏览器。ArcGIS Server 允许用户通过 ArcCatalog, ArcGIS Server 管理器, 或者通过 REST API 管理控制台来实现这些管理工作。

2.1 支持的服务类型

ArcGIS Server支持多种类型的REST地理信息系统服务, 包括地图制图、地理编码、地理数据、几何运算、地理处理、地球模型、图片和网络分析服务。每种类型的服务都具有其特定的功能。要确定需要创建哪种类型的地理信息系统服务, 首先需要了解ArcGIS Server

的服务能够实现哪些功能。表2.1简要概述了用户使用ArcGIS Server能够创建的地理信息系统服务类型及其功能。

表 2.1 ArcGIS Server 支持的服务类型及其功能

服务类型	功能
地图服务	最常用的 ArcGIS 服务。包括许多功能，提供对地图和图层内容的访问。主要进行制图、地图浏览或图层查询。支持地理处理（需要分析层）及网络分析（需要网络分析层）。
地理编码服务	主要进行地理编码。地理编码是按照坐标指定位置匹配到描述该位置的属性地址的过程，这些地址属性通常出现在参考材料中。也可以进行反地理编码，虽然现在很多商业的地理编码服务，但一些组织可能找不到适合他们的服务：地址信息不够更新，地址格式不一样或希望人们通过要素的本地名字来发现地址（例如，“大剧场，大体育馆等”）。所有这些情况需要特定的地理编码解决方案，这样就需要花费一定时间来构建适合自己需要的地理编码服务。
地理数据服务	使用 ArcGIS Server，通过本地网络或互联网可以访问地理数据库。能够执行地理数据库复制操作，使用数据提取进行拷贝，执行地理数据库查询。该服务非常适合远程访问地理数据库。例如，公司可以安装 ArcSDE 地理数据库来管理洛杉矶和纽约办公室的数据。地理数据服务创建后，每个办公室可以发布其 ArcSDE 地理数据库到互联网。地理数据库可以被用来创建 ArcSDE 地理数据库的备份，还可以周期同步的在互联网上改变地理数据库。
几何服务	执行几何计算，例如缓冲区、单一化、计算面积和长度、投影。
地理处理服务	表现为一系列已发布的操作和分析地理信息的工具集。每个工具执行一个或多个操作，例如投影转换，增加属性表的列，创建要素缓冲区。工具接受输入（要素集，表，以及属性值），执行输入数据操作，生成表达在地图或者进一步需要处理的输出。工具可以被同步或异步执行。
Globe 服务	给网络用户提供 3D 地图浏览。
图像服务	通过 Web 服务提供只读访问镶嵌影像或栅格数据集。
网络分析服务	执行网络分析功能，例如最优路径分析、最近设施查找、计算服务面积。

2.1.1 地图服务

地图服务提供访问地图和图层的功能。地图服务可以分为缓存地图服务或动态地图服务。通过高速缓存预先创建金字塔而不是动态渲染地图的地图服务，被称为缓存地图服务。动态地图服务则是在每一次请求提供地图时都需要服务器渲染一次地图，使用金字塔缓存的地图服务可以显著提高地图传输的速度，而动态地图服务则具有更高的灵活性。

REST API利用地图服务资源实现地图服务。这些资源只对发布的地图文档中默认的数据框架起作用。资源提供了地图，图层的基本信息，包括地图缓存、空间参考、坐标原点和内容、地图单位以及标注文字。同时这些资源也提供了一些相关的数据服务，如服务描述、服务作者和关键字。如果是缓存地图，额外的信息包括缓存金字塔框架，缓存金字塔细节，金字塔大小等。

地图服务资源支持以下多种操作：

- 地图输出——用于从动态地图服务中输出地图图片。地图可从原始的数据源转换为不同投影的显示结果。地图图片生成之后，地图服务就无法改变已有层的要素渲染，不能添加动态图层或改变图层的绘制顺序。
- 点击查看——基于用户鼠标在地图上的点击返回一个或多个图层的要素属性信息。
- 关键字查找——基于关键字返回一个或多个图层的要素属性信息。
- 条件查询——基于查询标准返回一个要素子集。
- 生成 KML——创建封装在 KMZ 文件中的 KML 文档。该文档包含一个利用指定的属性和参数链接到 KML 服务端点的网络。如果没有通过令牌服务限制，这一操作就是有效的服务。

地图服务不具有编辑功能。用户只能读取要素和属性内容。

2.1.2 地理编码服务

地理编码是分配位置的过程，通过地址对位置描述的要素建立坐标点与地址的一致性。描述地址有许多形式，从通用的门牌号码、街道名称的地址格式，到其它可行的位置描述信息，如邮政区划代码或人口普查编号。地址包括任意可区别地方位置的信息。

REST API的地理编码服务资源是一个地理编码（定位）服务。资源提供与地理编码相关的基本信息，如服务描述、地址属性、空间参考和位置属性。

地理编码服务资源支持两种操作：

- 查找候选地址——基于地址信息和位置返回候选列表。
- 反地理编码——返回的所有属性涉及反地理编码地址以及它的确切位置信息。

2.1.3 地理处理（Geoprocessing）服务

地理处理是企业地理信息系统业务的一个基本组成部分。地理处理提供所有地理信息系统用户所必需的数据分析、数据管理和数据转换工具。

地理处理服务表现为一系列已发布的操作和分析地理信息的工具集。每个工具执行一项或多项操作，例如地图投影变换、新增表的属性列或建立要素周围的缓冲区。工具接受输入

（如要素集，表和属性值），执行输入数据操作，并生成输出到地图或进一步加工的软件客户端。工具可以同步或异步执行。

使用地理处理服务可以实现：

- 列出可用的工具及其输入/输出参数。
- 同步执行一项任务。
- 异步提交工作的任务。
- 获取工作信息，包括工作状态。
- 使用地图服务显示结果。
- 检索由客户端作进一步处理的结果。

许多地理信息系统的使用涉及到重复工作，这就需要创建一个自动化工作流的框架。地理处理服务通过综合一系列按顺序的操作模型来满足这个需要，并将模型输出为一个工具。REST API地理处理服务资源提供了基本的信息与服务，如服务说明、提供的任务、执行类型和结果的地图服务器名称。地理处理服务资源的操作返回任务顺利完成后的结果。这些操作包括：

- 执行任务：执行类型为同步使用。当一个任务是同步执行，使用者必须等待执行的结果。
- 提交工作：执行类型为异步时使用。当工作是异步提交，用户可以进行其他的工作，并同时等待任务完成的通知。

2.1.4 几何服务

几何服务包含复杂的和经常使用的几何运算等实用方法。ArcGIS Server Web只提供一个名为“Geometry”的几何服务。请注意，如果需要几何服务的输入和输出，总是将其封装为一个数组。

使用几何服务可以实现：

- 缓冲区、投影和单一的几何运算。
- 计算面积和长度。

REST API几何服务资源主要是处理和算法资源，可以支持有关的几何运算。几何服务资源具有以下操作：

- 投影——返回几何体投影的一个数组。
- 简化——返回了一系列的简化几何体。
- 缓冲区——返回对输入几何体指定距离的一系列多边形。可用的一个选项是对每一个距离缓冲图形的合并。
- 计算面积和长度——计算指定输入的每个多边形面积和周长。

- 长度——计算指定输入的每条线的长度。

2.1.5 图像服务

图像服务提供只读访问镶嵌的图片或栅格数据集功能。REST API图像服务资源描绘ArcGIS Server发布的图像服务。资源提供图像服务的基本的信息，如服务描述、名称、说明、范围、像素大小与波段数。

使用图像服务，用户可以实现：

- 获取图像服务信息，包括其空间参考、范围、像素大小、像素类型、波段数量和波段统计特征。
- 生成图像。

图像服务资源只支持一个操作即输出图像，它会传回图像资源。

2.1.6 其他服务

REST API还提供了其他类型的服务。服务信息可以检索，但这些服务没有具体的操作与之相对应：

- 网络服务——描述网络分析服务。资源提供有关服务的信息，如服务描述和各种网络层，如网络分析中的路线、最接近的设施和服务面积。
- 地理数据服务——表达地理数据服务和提供服务描述等信息，工作空间类型，默认工作目录，版本和复制品。
- 地球模型服务（Globe Service）——表达地球模型服务，提供服务说明和发布的Globe 文档中各图层等信息。

2.2 创建 GIS 资源

当确定需要创建的服务类型后，用户就可以决定需要哪些地理信息系统资源。每种类型的地理信息服务需求相应的地理信息系统的资源。地理信息系统资源不是来自ArcGIS Server，而是通过使用ArcGIS桌面软件模块来创建。表2.2总结了每种类型GIS服务需要的GIS资源以及相对应的ArcGIS桌面应用程序。

表 2.2 服务类型和相关资源

服务类型	需要的GIS资源	ArcGIS Desktop 模块
地图服务	地图文档(.mxd, .pmf)	ArcMap
地理编码服务	地址定位(.loc, .mxs, SDE batch locator)	ArcCatalog

地理数据服务	数据库连接文件 (.sde) 或personal geodatabase或file geodatabase或地图文档参考数据	ArcCatalog
几何服务	不需要GIS资源	N/A
地理处理服务	带有工具图层的地图文档或工具层 (.tbx)	ArcMap或ArcCatalog 的 ArcToolbox和 ModelBuilder
Globe服务	Globe 文档(.3dd, .pmf)	ArcGlobe
图像服务	影像数据集或引用栅格数据集的图层文件或编译的图像服务定义(.ISCDDef)	ArcCatalog, ArcMap, 或 ArcGIS Image Server
网络分析服务	包含网络分析层的地图文档	ArcMap

创建地理信息系统资源，需要牢记的一点是：这些资源将发布为一种服务，而且这些服务可能同时被很多人访问。为了保持服务最佳性能，需要在创建资源时进行一些特别的考虑：

- 存储数据，必要的 ArcGIS Server 组件可以访问该数据。
- 设置某些 ArcGIS Server 的帐户权限，可以有权限访问该数据。
- 调整非缓存地图服务的性能，例如，使用尺度效应、简单符号和简化标注等方法。
- 针对缓存地图服务的每个尺度选择适当的尺寸和提高地图表现力。
- 选择输出目录或将数据嵌入到地理数据服务。
- 为地理处理服务选择一个适当的输出坐标系统。

如果需要更加详细的资料，请参考ArcGIS帮助文件，如“发布服务资源准备”，“ArcGIS Server地图发布准备”，“地理处理空间参考”，“地理处理服务小技巧”和“配置地理数据服务建议”等。

下面的一系列快照图（图2.1～图2.4）展示了如何使用ArcGIS模块来创建GIS服务所需要的资源。

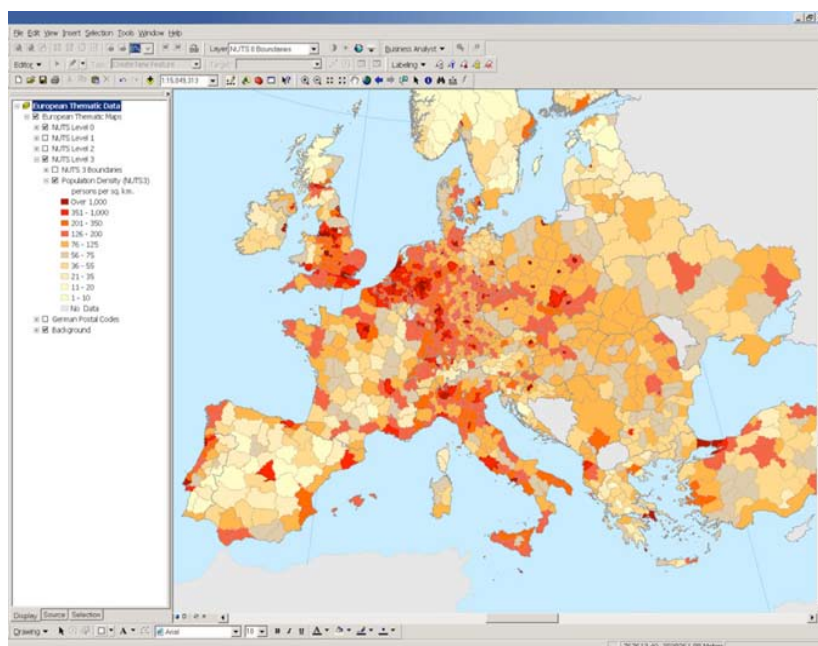


图 2.1 在ArcMap中创建地图文档以生成地图服务

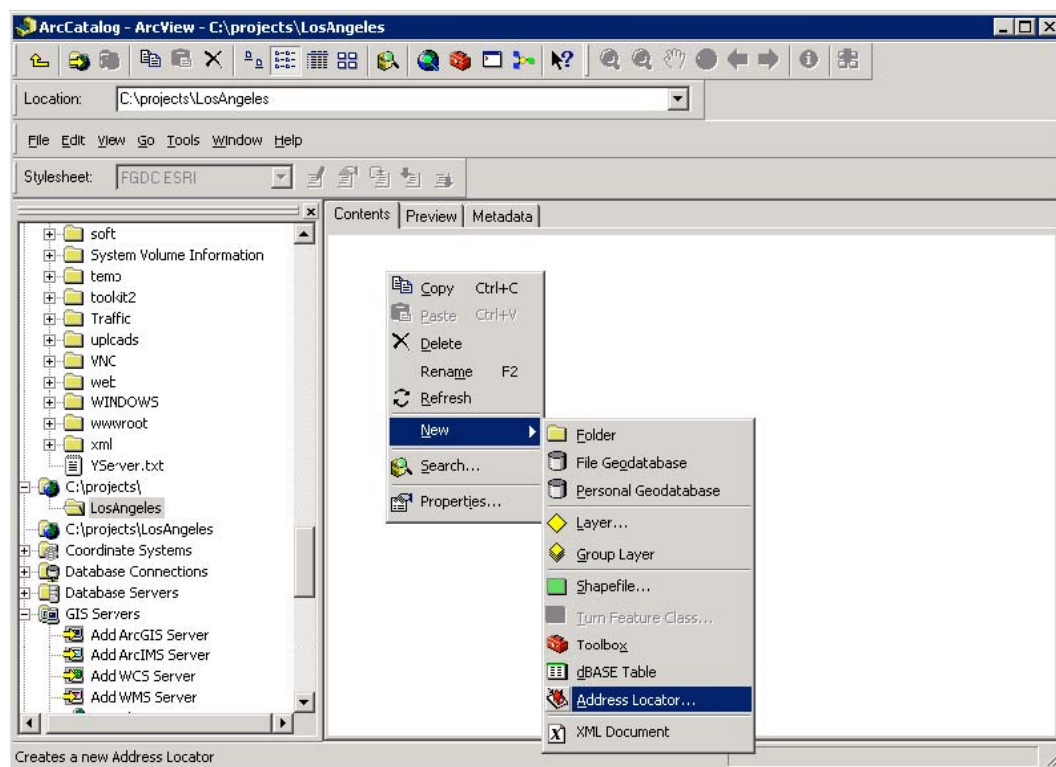


图2.2 在ArcCatalog中创建Address Locator资源以生成地理编码服务

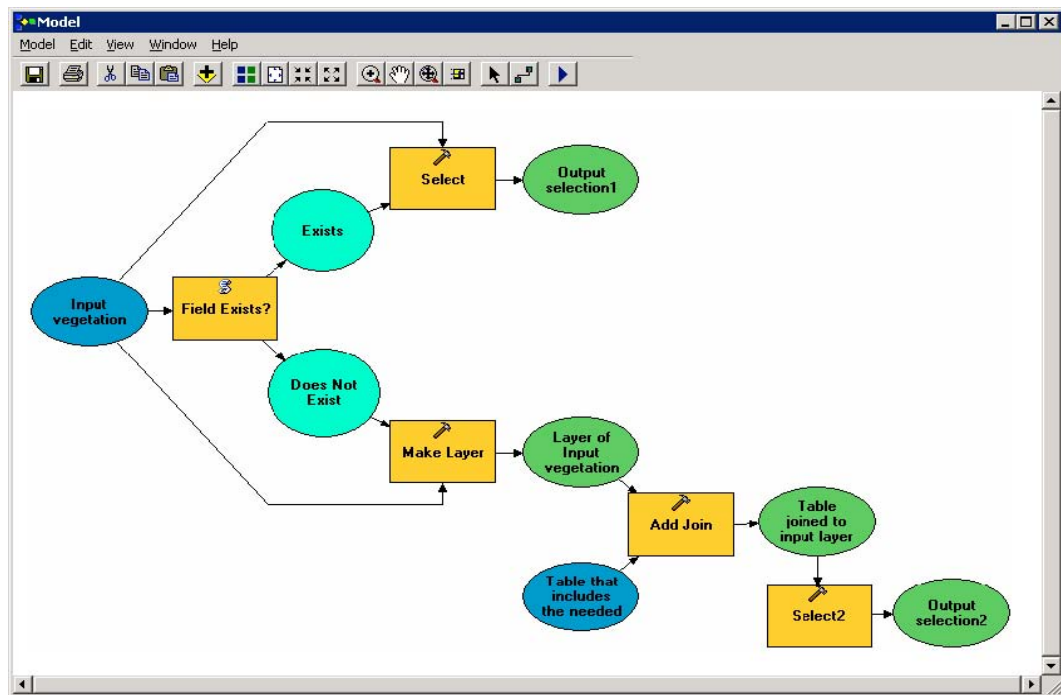


图2.3 在ArcGIS桌面ModelBuilder中创建模型以生成地理处理服务

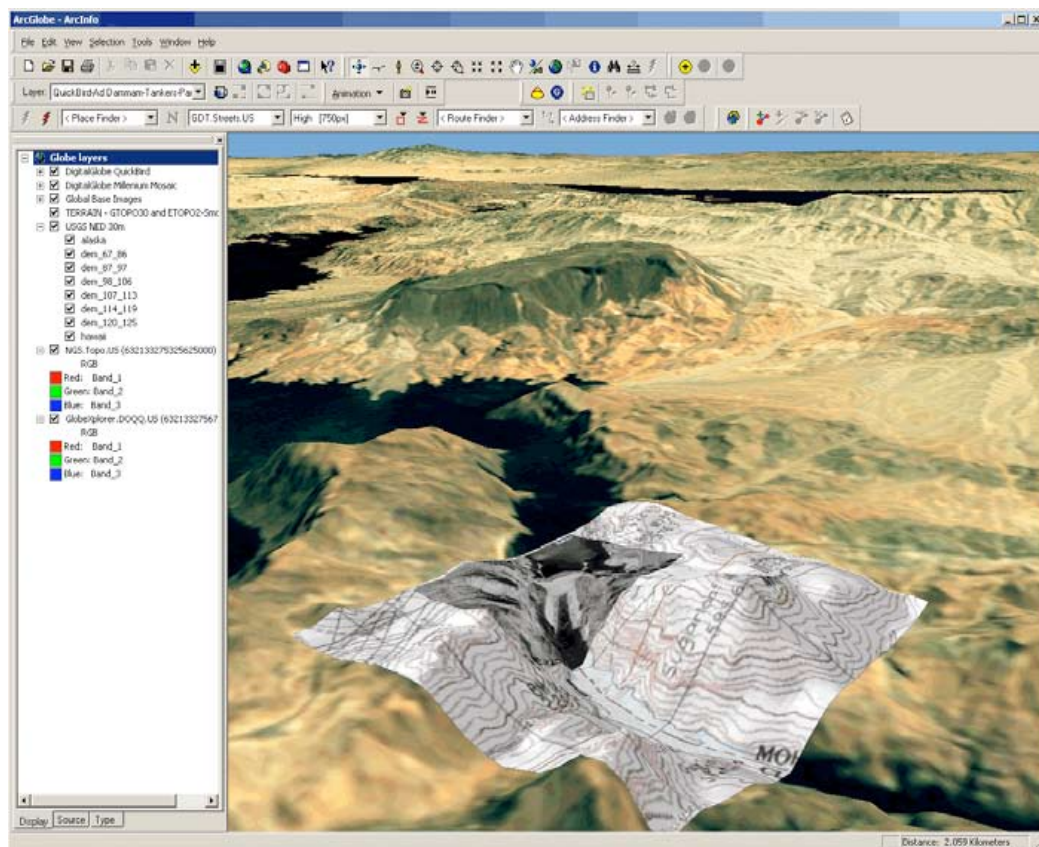


图2.4 在ArcGlobe中创建global文档生成global服务

2.3 资源发布为服务

GIS资源创建完成后，用户就可以将它们发布为Web服务，使他们能够被互联网或局域网的客户端应用程序远程调用。有两种方法可以发布GIS资源为Web服务：其一是使用“Publish GIS Resource”或“添加新的服务”功能。其二是使用ArcCatalog或ArcGIS Server管理器。请注意：

- 几何服务不需要任何资源，并且只能通过“添加新的服务”选项进行发布。
- 要实现快速的性能，地图服务和地球模型服务通常使用缓存。缓存的好处包括：更快的性能和更好的质量。欲了解更多信息，请参阅 ArcGIS Server 帮助文件“缓存服务”。

2.3.1 使用 ArcCatalog 发布服务

利用ArcCatalog，通过“发布GIS资源”或“添加新服务”功能发布资源。

2.3.1.1 发布 GIS 资源

以下是利用ArcCatalog发布地理信息系统资源为Web服务的具体操作步骤：

- 1、利用目录树视图，浏览到要发布的资源。例如，定位到磁盘上的一个地图文档，如 LAMap.mxd（图 2.5）。
- 2、右键单击地图文档，点击“Publish to ArcGIS Server”。
- 3、选择想要发布资源的服务器，输入服务的名称，然后选择服务存储的文件夹。单击下一步。
- 4、选择资源要启用的功能，然后单击下一步。
- 5、修改服务，单击完成后，服务可能要做出必要的调整。注意，选择某些类型的功能，将导致建立多种服务。

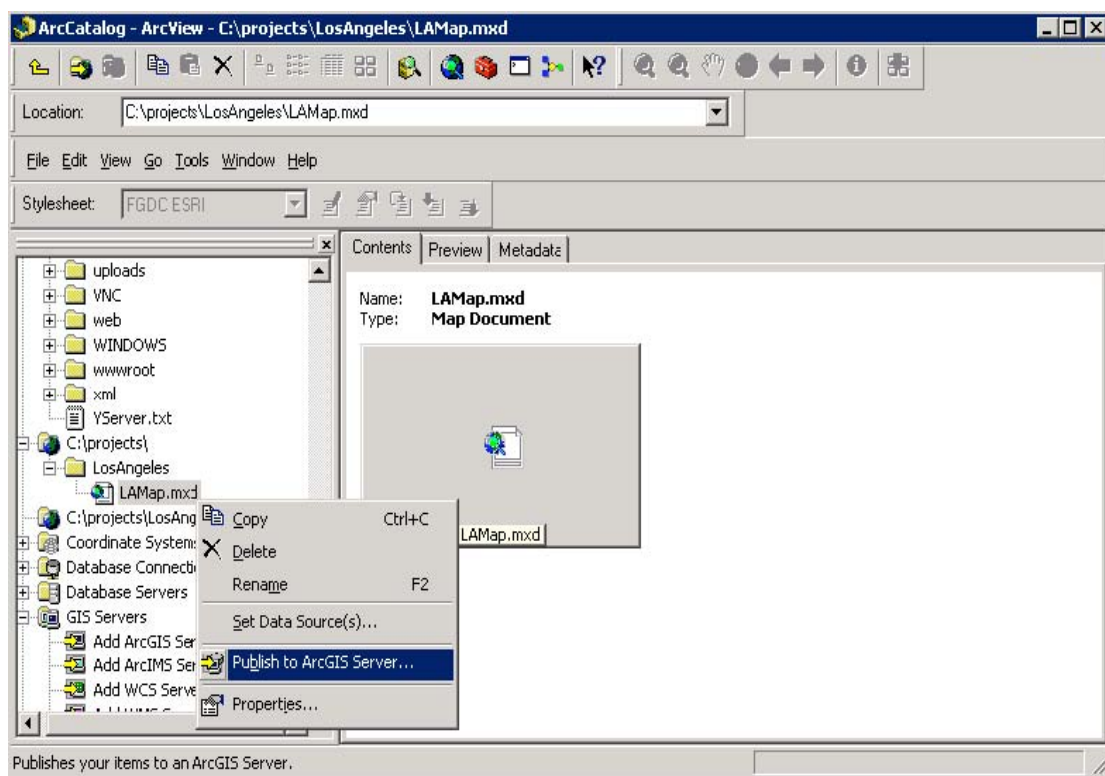


图 2.5 利用 ArcCatalog 发布地图文档为地图服务

2.3.1.2 添加新服务

以下是利用 ArcCatalog 增加新服务的具体操作步骤：

- 1、在要添加服务的 GIS 服务器上创建管理连接。
- 2、右键单击新增服务的服务器或文件夹，然后点击“Add New Service”（图 2.6）。
- 3、输入服务的名称。该名称就是用户看见以及用来识别服务。
- 4、点击 Type 下拉箭头，选择服务类型（图 2.7），然后单击下一步。
- 5、按照指示向导设置要创建的服务类型的初始化参数：
 - a. 如果创建一个地图服务，浏览地图文件，并选择显示的数据框架。指定输出的目录和缓存目录。
 - b. 如果创建一个地理编码服务，浏览地理处理位置，并设置批处理的大小。单击下一步。
 - c. 如果创建一个地理数据服务，选择创建的服务是直接来自 Geodatabase 还是包含 Geodatabase 中图层的地图文档。
 - d. 如果创建一个几何服务，不需要指定选择地理信息系统资源，因此，向导第二面板跳过。
 - e. 如果创建一个地理处理服务，首先选择工作是否将同步（适用于短期业务）或异步（适用于长事务，结果存储到服务器以备检索）进行。然后，浏览选择工具箱或要发布的地图文档。

- f. 如果建立一个地球模型服务，浏览要使用的地球文档，然后指定预先创建的地球模型高速缓存的位置和虚拟目录。
 - g. 如果建立一个图像服务，浏览栅格数据集，指引到栅格数据集的层文件，或要发布的编译图像服务定义文件。选择一个输出目录。
 - h. 如果建立一个网络分析服务，浏览选择包含网络分析层的地图文档。
- 6、如果创建的服务类型有可供选择的功能，将弹出新的网页，可以选择要启用的功能，并设置其属性。在该网页上，用户也可以选择服务是否可以被网络访问或者允许被网络远程操作。
- 7、点击 **Pooled** 或 **Not Pooled**，随意改变最大使用量和等候时间。单击下一步。
- 8、设置进程隔离级别和循环参数。单击下一步。
- 9、单击 **Yes** 来启动服务器对象，然后单击完成。
- 10、确认服务是否工作正常。在 **ArcCatalog** 中可以预览或显示服务的属性，以确保服务的正确配置。如果因为某些原因，服务没有出现预期的效果，可以查看日志文件排查错误。

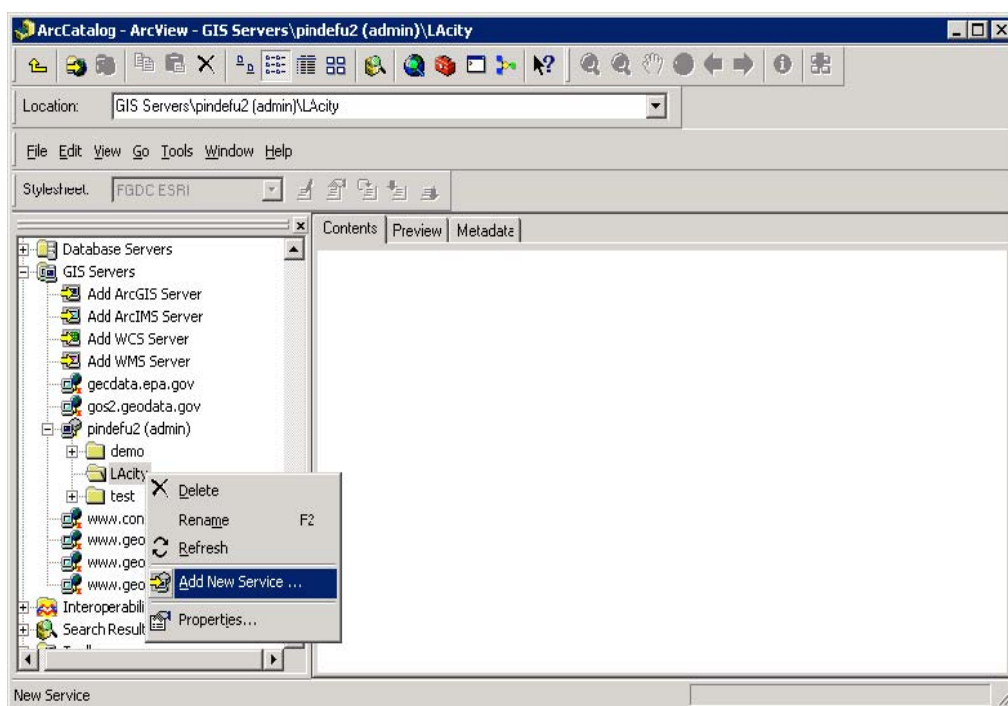


图2.6 在ArcCatalog中添加新服务

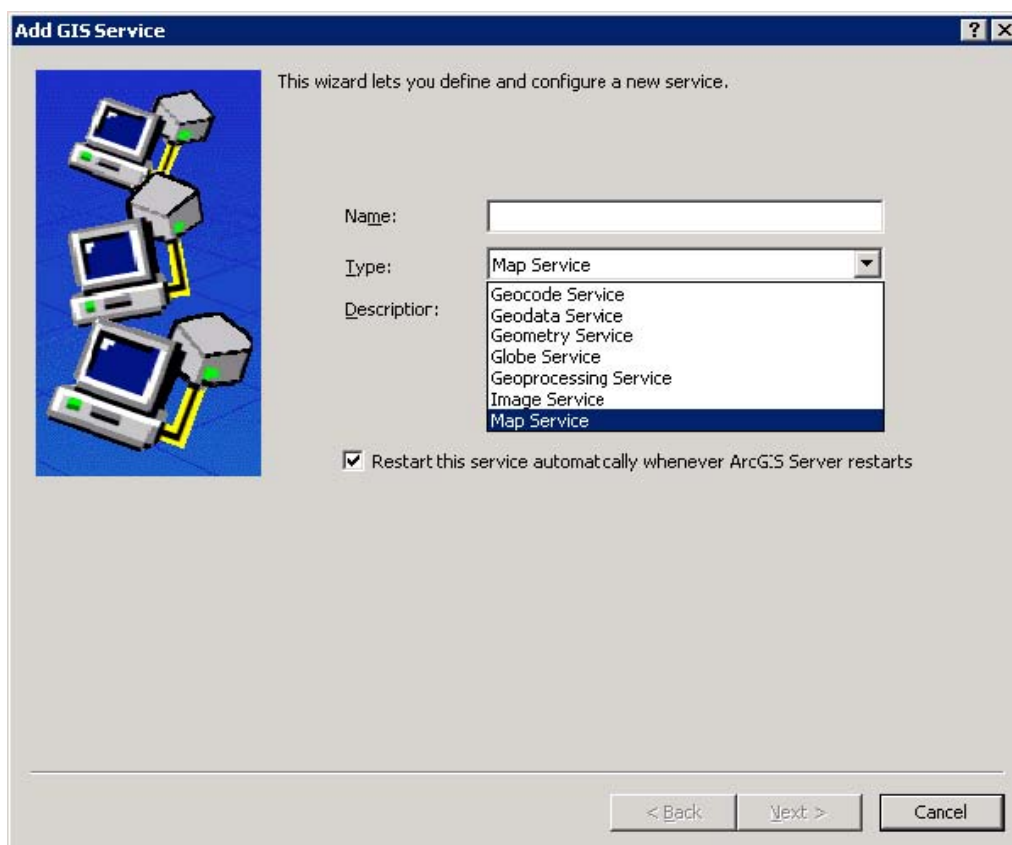


图2.7 选择在ArcCatalog创建的服务类型

2.3.2 利用 ArcGIS Server 管理器发布服务

利用ArcGIS Server管理器，用户可以通过“发布GIS资源”或“Add New Service”两个功能将资源发布为服务。

2.3.2.1 发布地理信息系统资源

以下是利用ArcGIS Server管理器发布GIS资源的具体操作步骤（图2.8）：

- 1、单击服务器管理器中的 **Services** 选项卡。
- 2、点击“Publish GIS Resource”。
- 3、点击 **Resource** 下拉列表选择要发布的资源或直接输入资源路径。
- 4、可更改服务的默认名称。
- 5、选择需要发布服务的文件夹。可以指定现有的文件夹或创建一个新文件夹。单击下一步。
- 6、选择要启用的功能。根据资源的不同类型及其所包含的信息，会发现可用功能会有所不同。单击下一步。

- 7、确认即将创建服务的有关信息。如果要通过 Web 调用服务，需特别注意该服务的网址。
- 8、如果想改变任何属性，可以点击“Previous”返回上一步。点击“Finish”发布资源。

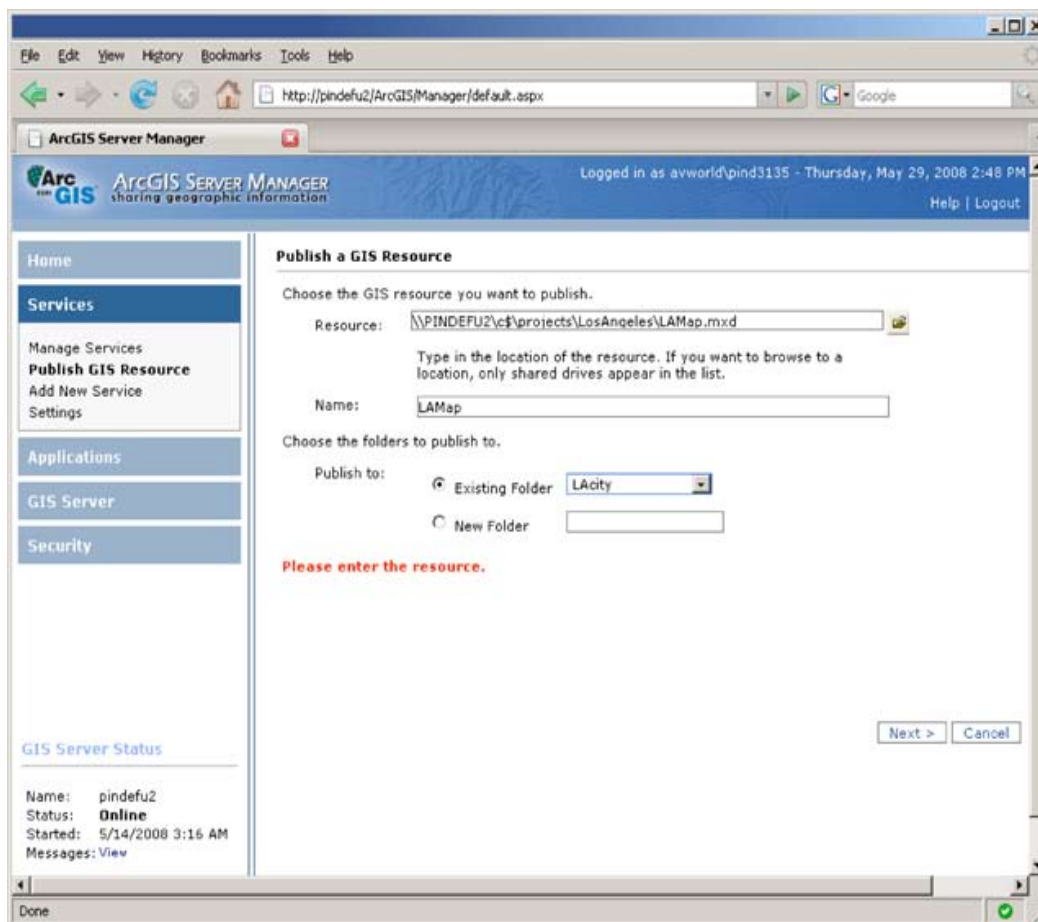


图2.8利用ArcGIS Server管理器发布地图服务

2.3.2.2 添加新服务

以下是通过ArcGIS Server管理器的添加新服务功能发布GIS资源的具体操作步骤（图 2.9）：

- 1、在服务器管理器中，点击“Services”标签。
- 2、单击“Add New Services”选项。
- 3、输入服务的名称。这样用户就能发现和使用确定被描述的服务。
- 4、点击“Type”下拉箭头，单击服务的类型，然后单击下一步。
- 5、按照下列服务类型设置向导设置初始化参数：
 - a. 如果创建一个地图服务，浏览地图文件，并选择显示的数据框。指定输出目录和缓存目录。

- b. 如果创建一个地理编码服务，浏览地理处理的位置，并设置批处理的大小。单击下一步。
 - c. 如果创建一个地理数据服务，直接从地理数据库或从地图文件选择是否要创建的服务，或从地理数据库的图层中选择。
 - d. 如果创建一个几何服务，不需要指定选择地理信息系统资源，因此，向导跳过。
 - e. 如果创建一个地理处理服务，首先选择工作是否将同步（适用于短期业务处理）或异步（适用于长期工作，结果存储到服务器以备检索）进行。然后，浏览想要发布的任何工具箱或地图文件。
 - f. 如果建立一个地球模型服务，浏览要使用的地球模型文件，然后指定预先创建的地球模型高速缓存的位置和虚拟目录。
 - g. 如果建立一个图像服务，浏览栅格数据集，指引到栅格数据集的层文件，或要发布的编译图像服务定义文件。选择一个输出目录。
 - h. 如果建立一个网络分析服务，浏览包含网络分析层的地图文件。
- 6、如果选择创建的服务类型，还有可供选择的功能，将弹出新的网页，可以选择要启用的功能，并设置其属性。在该网页上，用户也可以选择服务是否可以被网络访问或者允许被网络操作。
- 7、点击 **Pooled** 或 **Not Pooled** 随意改变最大使用量和等候时间。单击下一步。
- 8、设置进程隔离级别和循环参数。单击下一步。
- 9、回顾即将创建的服务的有关信息，然后单击完成。
- 10、确认服务是否工作正常。如果该服务已启动和正常工作，你应该能够看到服务器管理器中服务标签的缩略图，点击加号（+）按钮，则依次出现到下一个服务的名称。如果出于某些原因，发布的服务不是如预期般的工作，可以查看日志文件中的错误。

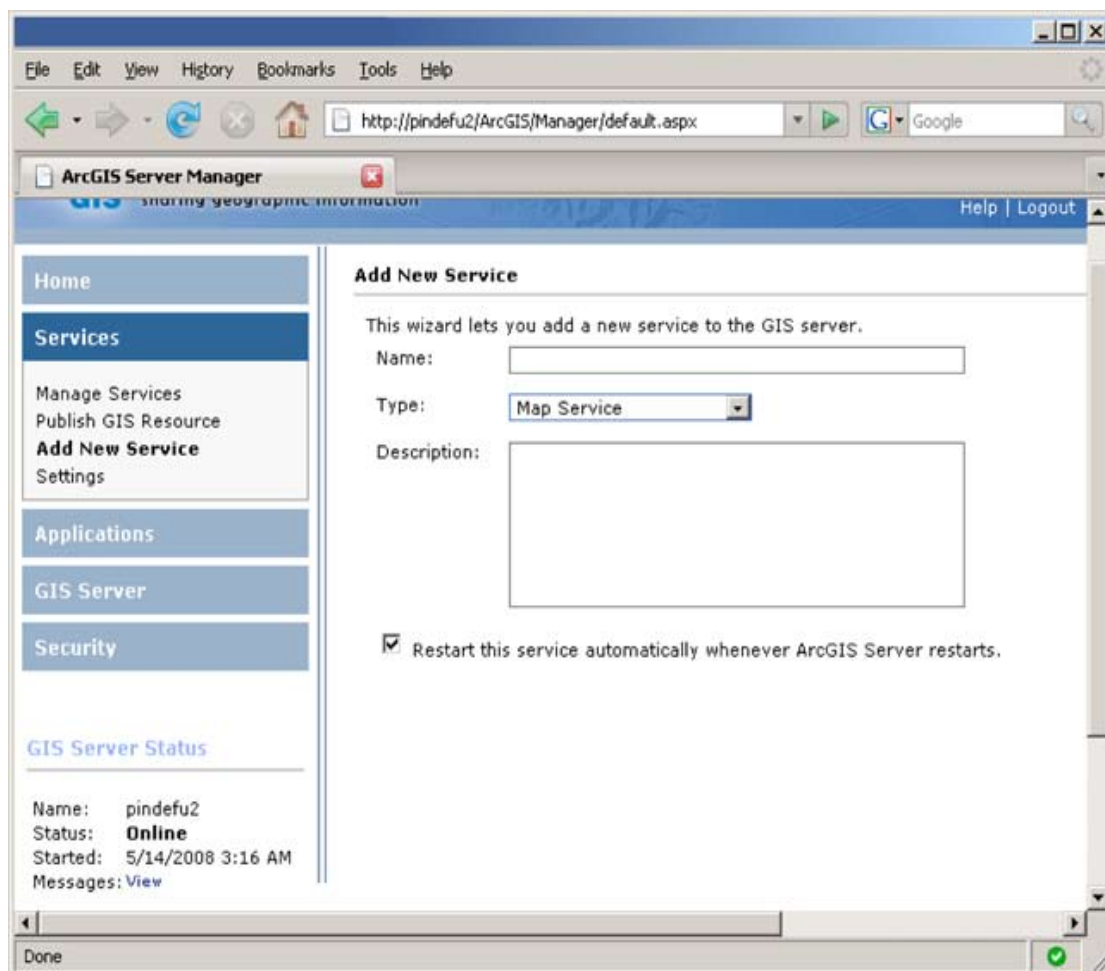


图2.9利用ArcGIS Server管理器添加新的地图服务

2.4 浏览服务

ArcGIS Server 9.3引入了服务浏览器。每个ArcGIS Server服务器都整合了一个服务浏览器，用来查看服务目录。服务目录是通过ArcGIS Server的REST API的HTML格式实现。

通过URL，REST使发现工作和查找所需的信息更加容易。利用服务浏览器，用户可以浏览服务器内容，查看可用的地理信息系统网络服务，也可获取开发过程中的有用信息。

2.4.1 浏览服务器内容

使用服务资源管理器打开ArcGIS Server上的目录，例如，打开 <http://sampleserver1.arcgisonline.com/ArcGIS/rest/services>，首先会看到主页（图2.10），其中列出了所有服务目录以及含有更多服务的文件夹。除了服务名称，还可以看到服务类型，如地图服务或地理编码服务等。

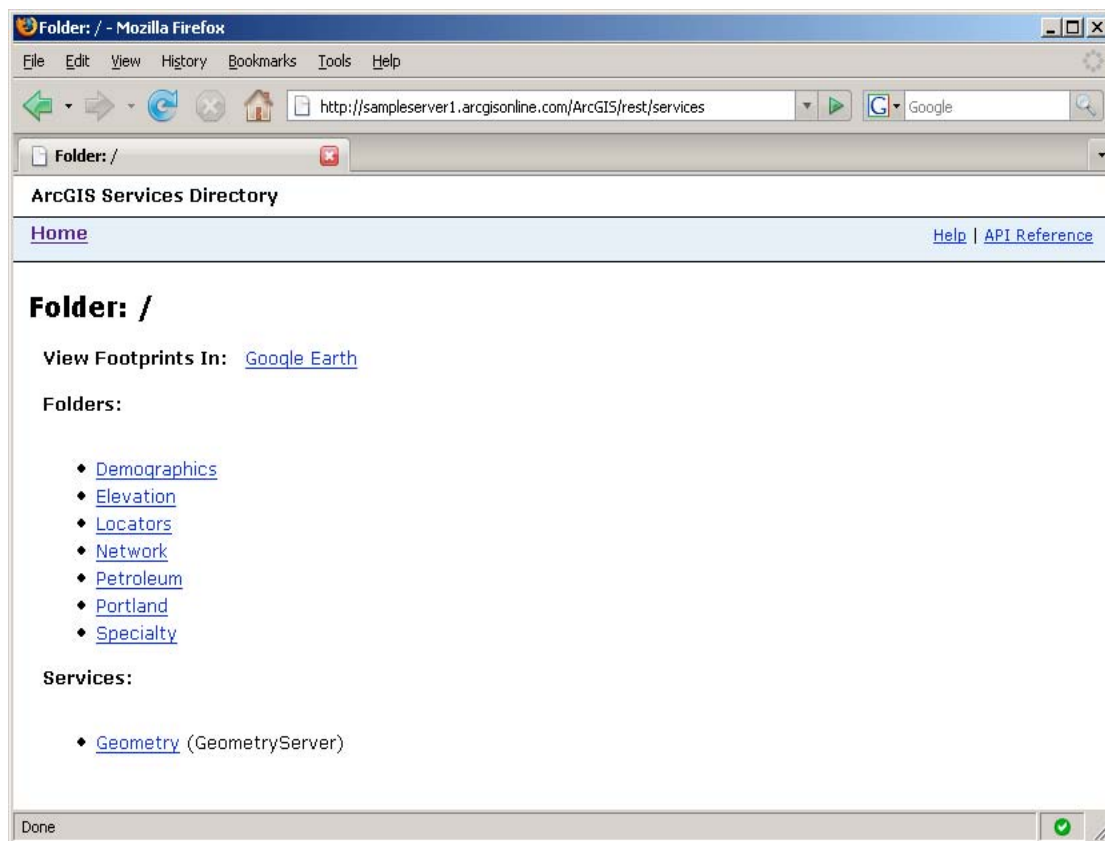


图2.10 服务浏览器显示ArcGIS服务器根目录的内容

注：服务器管理员决定文件夹结构。用户无法通过服务目录控制修改文件夹的结构。

点击服务名称，可以获得更多的信息。信息因服务类型而异。如果单击地图服务（地图服务器），用户将看到的信息包括如图层名称，文档信息以及支持的程序接口。

如果继续点击链接，你就可以了解服务中每个图层的信息。通过这种方式，服务目录可以展示服务的大量元数据（图2.11）。

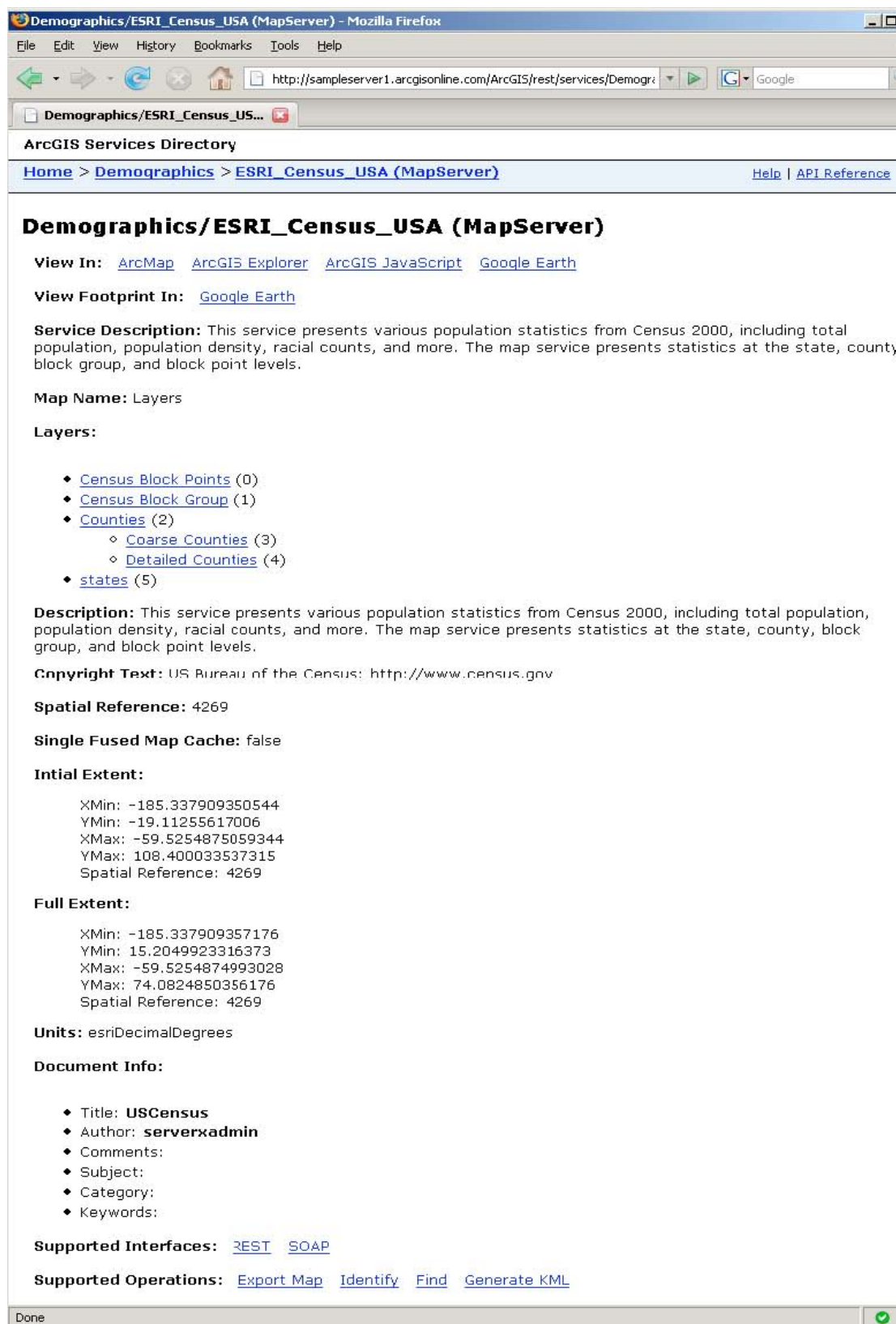


图2.11 ArcGIS 服务目录显示地图服务的元数据

2.4.2 查看服务空间范围

地理空间范围代表服务覆盖的实际自然面积。从本质上讲，它是一种显示地理范围的服务。

如果利用谷歌地球（图2.12）查看服务空间范围，打开一个KMZ文件，显示的KML“地标”来指示每个可用的服务

<http://sampleserver1.arcgisonline.com/ArcGIS/rest/services?f=kmz>。前提是必须安装谷歌地球才能看到这个结果。单击一下地标就会在弹出的对话框中显示更多信息服务。



图2.12 ArcGIS 服务目录提供察看服务空间范围的链接

如果选择在服务器的根目录查看服务足迹，可以从根目录一级和所有文件夹层次看到服务空间范围。另外，也可以只在文件夹层次查看服务空间范围。

用户也可以使用服务空间范围显示服务器上的服务。使用服务目录，发现服务空间范围的URL（图2.13）。然后，如果共享网址，其他用户就能够看到服务器上可用的最新服务。请注意，如果服务使用了基于令牌的认证安全机制，服务空间范围链接就不再可用。

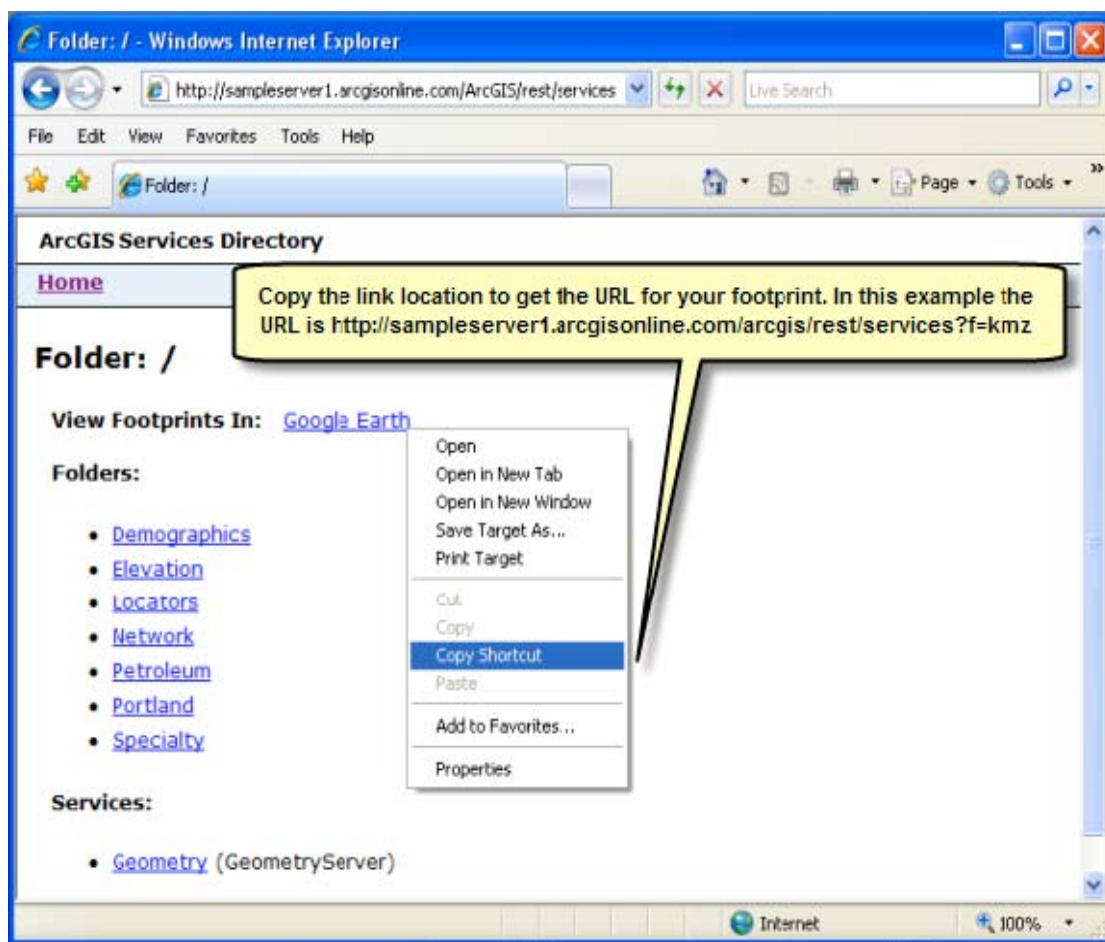


图2.13 如何获得服务空间范围的URL

2.4.3 测试 REST 服务

利用ArcGIS Server服务浏览器，用户可以浏览服务的目录，检查每一个服务的元数据，并测试REST Web服务。例如，地图服务让你查看地图和生成KML。

2.4.3.1 查看地图

发布地图服务之后，就可以通过利用服务目录应用几种不同格式查看其内容(图2.14)。当浏览到一个地图服务的主页，就会看到“View In”提供了不同应用程序中查看的选项。这些选项包括：

- **ArcMap:** 该网址提供了一个图层文件（.lyr）来参考服务。用户可将图层添加到任何的 ArcMap 文件中。利用图层文件的优势是其包含链接信息，不必使用手动连接服务器。
- **ArcGIS Explorer:** 这个网址提供了一个 ArcGIS Explorer 地图文件（.nmf）参考服务。使用此选项，用户可以在 ArcGIS Explorer 中浏览 3D 服务。

- **ArcGIS JavaScript:** 这个网址提供了 Web 浏览器中地图的一个简单预览。预览使用了 ArcGIS JavaScript API。这个选项的优势是，不需要任何插件或从客户端下载。
- **谷歌地球:** 本网址将地图的内容作为网络连接的 KML (.kmz)，适用于在谷歌地球中应用。KML 使用底层叠加来显示栅格化的地图。如果地图缓存在一个支持的坐标系统中，KML 区域就被使用。如果服务使用基于令牌环的认证安全机制，服务就不再可用。
- **微软虚拟地球:** 本网址提供微软虚拟地球的一个预览图。此链接只能适用于已缓存，且投影为 Web 麦卡托投影的服务（102113）。
- **谷歌地图:** 本网址提供谷歌地图的预览图。此链接只能适用于已缓存，且投影为 Web 莫卡托投影的服务（102113）。

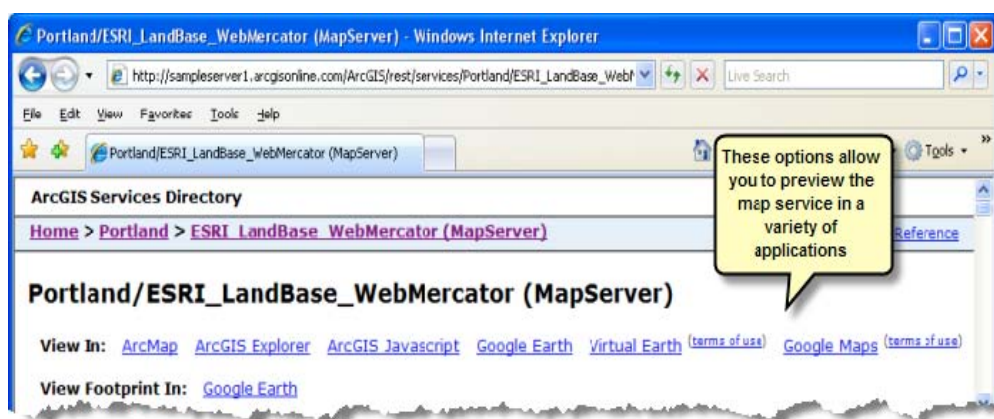


图2.14 ESRI_LandBase_WebMercator服务可以被多个程序预览

2.4.3.2 生成 KML

所有具有KML网络连接的地图和图像服务，都可以选择在上文提到的谷歌地球中查看。如果需要一个网络链接，例如拥有属性的矢量要素类型，或其他默认的类型，用户都可以使用服务目录创建自己的KML网络连接。生成自己的KML网络连接，浏览要查看的地图服务页面。从Support Operations清单中，点击“Generate KML”链接（图2.15）。

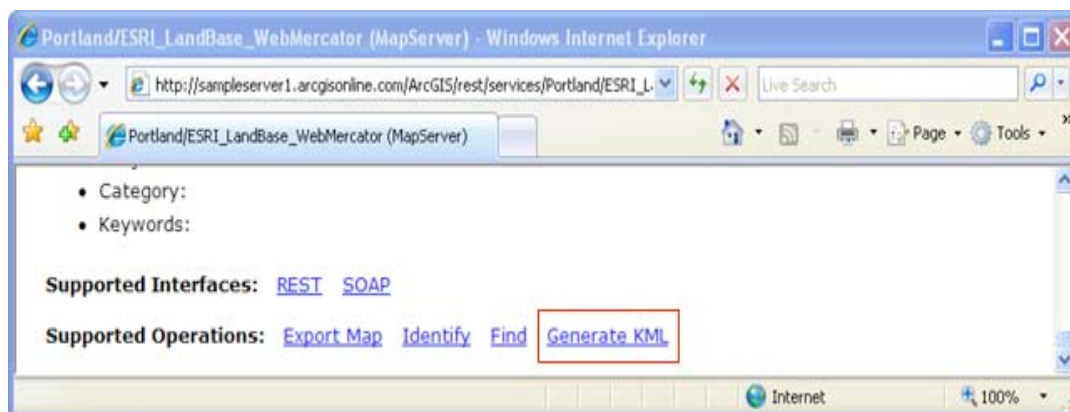


图2.15产生kML网络连接的地图服务

强调一点：如果服务器管理员已经停用了KML服务，则生成KML的链接将无法使用。此外，如果使用令牌为基础的验证安全机制，这个链接也不会提供服务。

在这个网页上，用户可以设置KML网络连接的一些基本的功能，包括名称、图层及图层绘制顺序。如果想查看矢量要素，则选择第三个选项，“矢量层作为矢量和栅格层作为图像”（图2.16）。

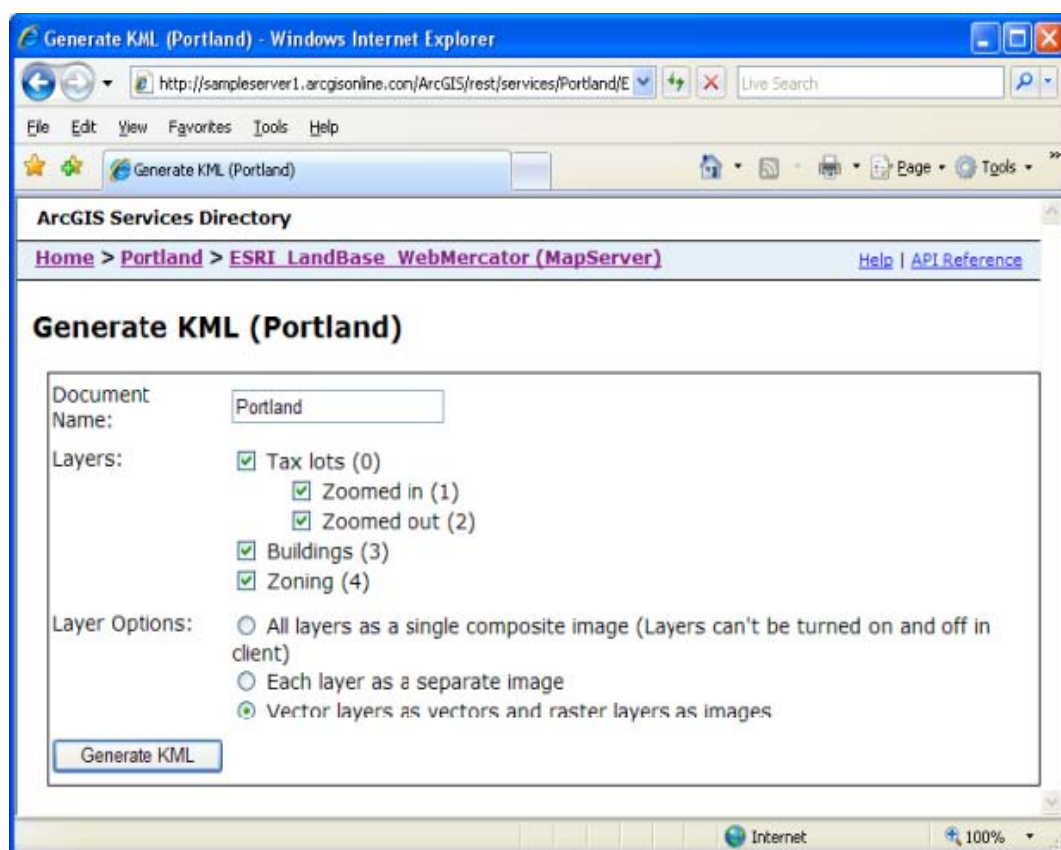


图2.16 生成可以选择层绘制选项的kML页

请注意，服务器管理员可以对服务目录中设置的一些选项进行限制或禁用。例如，管理员可以停止服务器返回矢量要素的功能，或管理员可以设定服务器可返回KML大小的限制。

一旦生成网络链接，就可以分发到其他需要该链接的用户。利用分布式网络链接而不是静态KML的优势是，网络链接指向地图服务的KML。引用动态的KML方式可以确保用户永远看到最新的功能。

2.4.4 获取开发信息

当需要应用JavaScript进行开发时，服务目录可以帮助获得有用的信息。ArcGIS Server 包括的JavaScript API是基于REST，每一个ArcGIS Server通过REST的端点或网址提供信息。每个端点返回服务器或其服务的信息片段。

例如，使用ArcGIS Server JavaScript API，开发者就可以编写代码，在Web浏览器中显示地图。代码需要地图服务的REST端点，看起来就像这样：

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

怎么知道如何建造这个端点？如果用户熟悉ArcGIS Server，就可以从内存中构建端点。但更有可能，用户使用服务目录来发现端点。利用服务目录，开发者就可以浏览服务器的内容，一直到地图服务。然后，开发者就可以复制浏览器网址，并粘贴到其代码中。

2.4.5 如何在开发中使用服务目录的示例

添加缓存地图到用户应用的程序代码看起来应该这样：

```
myTiledMapServiceLayer = new esri.layers.ArcGISTiledMapServiceLayer  
("http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/Portland_ESRI_LandBase_AGO  
/MapServer ");  
  
myMap.addLayer(myTiledMapServiceLayer);
```

注意地图服务的REST端点网址：

<http://sampleserver1.arcgisonline.com/ArcGIS/rest/services>。

如果不知道这个网址，按照下面的步骤使用服务目录就可以帮助你发现该网址：

- 1、打开一个 Web 浏览器
<http://sampleserver1.arcgisonline.com/ArcGIS/rest/services>。记住这个网址格式——<http://<服务器名称>/<实例名称>/rest/服务>——这就是如何打开某一特定 GIS 服务器的服务目录。
- 2、查找需要的服务网址，然后单击看到服务的主页。
- 3、复制浏览器网址，并将其粘贴到代码中（图 2.17）。



图2.17 可以拷贝的REST端点URL

有时候，开发者需要访问地图中的单个层。例如，建立层查询的代码如下：

```
function init() {  
  //build query  
  myQueryTask = new esri.tasks.QueryTask  
  ("http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/Portland_ESRI_LandBase_AGO  
  /MapServer/1");  
  ...  
}
```

上述代码中利用的网址中看起来很像地图网址，利用索引指出哪个图层可以访问（上述网页例子中的索引为“1”）。

索引是地图目录中图层的位置。既然可能不会立即知道该索引，就可以使用目录找到它。

究其详细的步骤，可以回到上述第3步。在复制网址之前，仔细浏览页面上有有关于“Portland”的地图服务。将会看到一个可点击的图层列表。点击任何层就会生成一种新的网址层索引，进而就可以粘贴到代码中使用。

2.4.6 支持的输出格式

REST API支持很多种格式的响应。使用查询参数f，用户就可以指定返回响应的格式。有效的格式包括以f为参数的所有资源和操作。第3章中，我们将详细解释每种格式的使用。

- 如果 f=html：除非另有说明，就是默认格式。响应格式就是 HTML 网页格式。每个资源的 HTML 网页集就是所谓的目录。

使用以下网址，可以查看ArcGIS Online根目录的服务目录的网页：

<http://server.arcgisonline.com/arcgis/rest/services?f=html>

请注意，如果“HTML”是默认值，就不需要在网址中包括这个参数。也就是说，上述网址是等同于以下网址：

<http://server.arcgisonline.com/arcgis/rest/services>

- 如果 **f=json**：则 REST API 的响应就是一个 JSON 对象。这种格式主要用于 JavaScript API。

要检索 JSON 对象的信息，可以使用以下网址：

<http://server.arcgisonline.com/arcgis/rest/services?f=json>

也可以在下列网址中引用一个回调函数：

<http://server.arcgisonline.com/arcgis/rest/services?f=json&callback=myMethod>

如果要使 JSON 对象更具可读性，就可以使用 **pjson** 或 **pretty** 标记。不必将影响表达的参数包括到应用中。而只使用这些参数用于调试目的：

<http://server.arcgisonline.com/arcgis/rest/services?f=pjson>

<http://server.arcgisonline.com/arcgis/rest/services?f=json&pretty=true>

- 如果 **f=image**：则响应格式就是一个影像流，没有任何其他信息。

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_StateCityHighway_USA/MapServer/export?bbox=-117,35.79,-122,42.38&f=image

- 如果 **f=help**：则响应的就是一个上下文帮助文件。下面的网址打开页面上的帮助，提供有关地图服务资源的信息：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer?f=help

- 如果 **f=lyr**：则响应就在 ArcMap 中生成 layer 文件。当然前提是必须安装 ArcMap。

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=lyr

- 如果 **f=nmf**：则响应的就是在 ArcGIS Explorer 中生成的 layer 文件。

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=nmf

- 如果 **f=jsapi**：则响应的是使用 ArcGIS JavaScript API 在 Web 浏览器中显示地图服务的网页。

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=jsapi

- 如果 **f=ve**：响应的就是使用虚拟地球浏览地图服务的 Web 网页。请注意，此格式只支持缓存地图和具有 Web 莫卡托投影的服务（102113）：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=ve

- 如果 **f=gmaps**：响应的是使用谷歌地球浏览地图服务的 Web 网页。此格式只支持缓存地图和具有 Web 莫卡托投影的服务（102113）。

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=gmaps

- 如果 **f=kmz**：则响应的是封装在 KMZ 文件中的 KML 文件。可能是一个操作的快照。前提是必须安装谷歌地球。

在下面的例子中，是一个 KML 地图服务空间范围的请求：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Specialty/ESRI_StateCityHighway_USA/MapServer?f=kmz

KML 区域或底层叠加文件：安装谷歌地球后，封装在 KMZ 文件中的 KML 区域或底层叠加文件就可以被请求。

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer/kml/mapImage.kmz

2.5 管理服务

正确地管理服务非常重要。如果要启动，停止或暂停服务，可以通过 ArcCatalog 或 ArcGIS Server 管理器来实现。如果更新服务缓存或启用/禁用服务浏览器，则必须通过 REST API 的管理控制台来实现。

2.5.1 启动，停止，暂停服务

服务只有启动以后客户端才能访问。当停止服务，服务器立即删除所有服务。当暂停服务，服务器拒绝任何新的客户端的服务请求。但是现有的客户端可以完成其使用的服务。

用户可以通过 ArcGIS Server 管理器或 ArcCatalog 实现这些管理任务。

2.5.1.1 使用管理器启动，停止或暂停服务

在ArcGIS Server管理器中，单击服务选项卡上显示的服务清单，选择启动、停止、暂停，以及重新启动命令（图2.18）。当点击其中的一个命令，它将适用于所有清单中的服务。

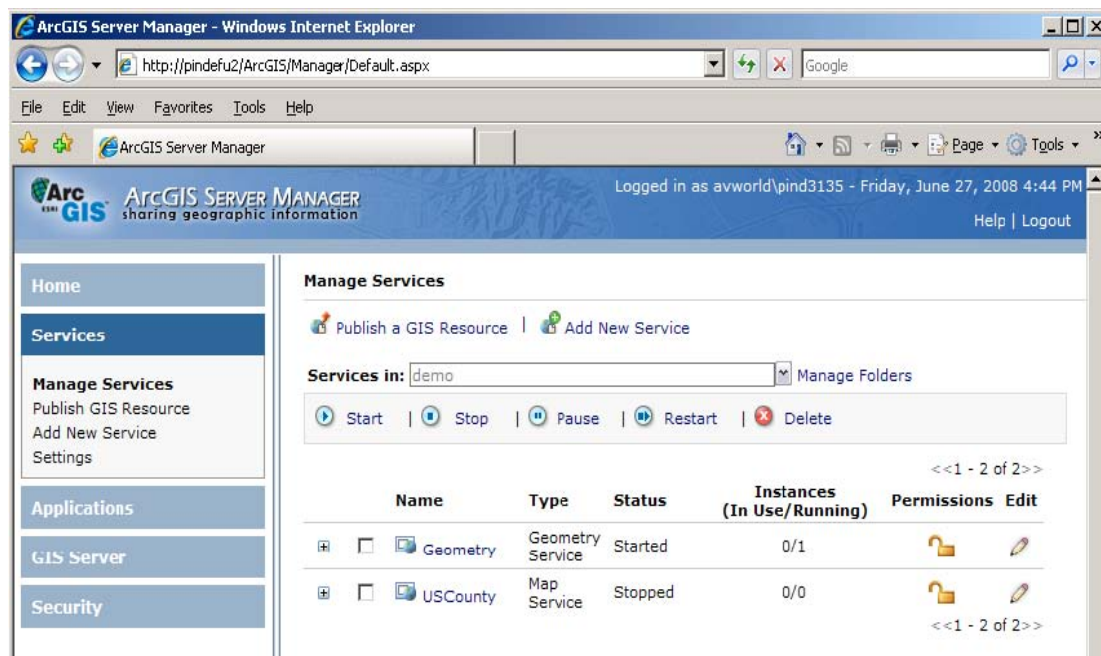


图2.18 利用ArcGIS Server管理器管理服务

2.5.1.2 利用 ArcCatalog 启动，停止或暂停服务

在ArcCatalog中，生成服务器的管理连接，并浏览需要管理的服务。

右键单击该服务以显示菜单，选择启动，停止，暂停，或重新启动服务（图2.19）。另外，也可以利用ArcGIS Server的服务管理工具，同样可以实现这些功能。

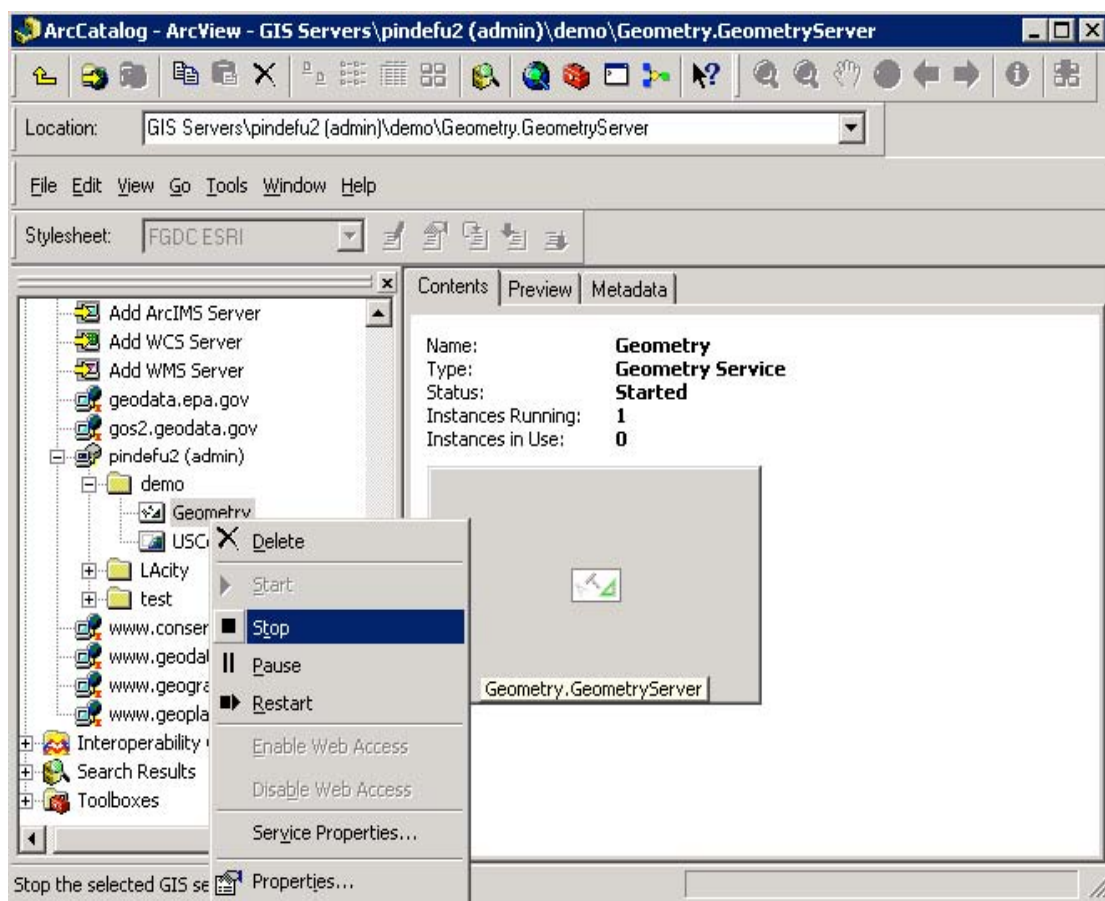


图2.19 利用ArcCatalog 启动，停止，暂停和重启服务

2.5.2 管理服务缓存

ArcGIS Server REST API缓存包括目录，服务，地图，模型等。缓存内容通过REST API就能显著地改善性能。至关重要的是，如果我们添加，删除或更新服务，这些缓存也会更新到REST API新版本。

ArcGIS Server提供一个管理控制台来管理REST服务缓存。按照默认安装，管理控制台可通过以下网址使用：

Java Server: <http://<host>:8399/arcgis/rest/admin>

.NET Server: <http://<host>/arcgis/rest/admin>

REST “管理员” 具有安全保护机制，所以，只有ArcGIS Server管理员（agsadmin）组权限的管理员用户才可用。使用管理控制台，必须使用管理员帐户的用户名和密码登录(图2.20)。

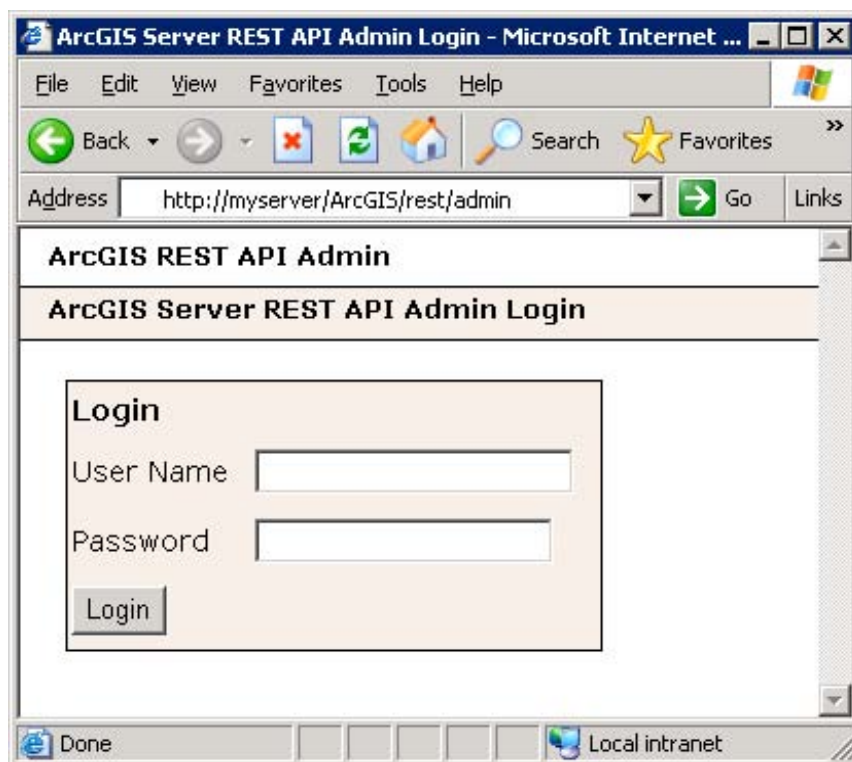


图2.20 输入域名/用户名和密码来访问admin

登录后，就可以看到“清除缓存选项”的页面，通过点击清理缓存按钮或其他选项就可以清理页面。（图2.21）。



图 2.21 使用管理控制台清除缓存

点击“Clear Cache Now”将能释放整个REST API缓存。用户可以使用此选项立即明确地清除缓存。此外，用户还可以配置如下的缓存交换规则：

- 手动——在手动模式下，缓存是不会自动清除。用户必须自己手动使用“清空缓存”选项，如上所述。
- 预定——在预定的模式下，按照每天固定时间点，高速缓存将被清除。
- 定期——在定期模式中，用户设定的一定的间隔，高速缓存将按照规定的时间间隔被清除。

请注意，用户不仅可以通过运行管理控制台来设定这些选项，还可以通过部署REST API时的配置文件来设置这些选项。在Java编程实现环境中，缓存策略设置可以通过配置REST config.properties文件中的config.cache属性来实现，而REST config.properties文件存储在WEB-INF/classes/resources文件夹中。在.NET编程实现环境中，缓存策略设置可通过配置REST Web应用程序根文件夹中的rest.config文件的缓存内容来实现。

2.5.3 启用和禁用服务目录

服务目录提供ArcGIS Server中已发布服务的一个简单的HTML视图。在有些情况下，如果不想让用户看到服务器和服务的内容。在这种情况下，我们就需要禁用服务目录。

默认安装情况下，服务目录是启用的（图2.22）。管理员可以将其停用，并重新启用。一旦停用，用户将无法查看服务目录中的服务。一旦服务目录被管理员禁用，用户访问将会得到返回的错误信息。



图 2.22 利用REST API admin可以启动和关闭服务目录

请注意，用户不仅可以通过运行管理控制台来设定这些选项，还可以通过部署REST API时的配置文件来设置这些选项。在Java实现环境中，缓存策略设置可以通过配置REST config.properties文件中的config.cache属性来实现，REST config.properties文件存储在WEB-INF/classes/resources文件夹中。在.NET实现环境中，缓存策略设置可通过配置REST

Web应用程序根文件夹中的rest.config文件的缓存内容来实现。

第三章: REST 应用

我们在第2章已经介绍了如何创建REST Web服务,本章将论述使用REST Web服务的方法,包括如何通过浏览器端编程、服务器端编程和台式机编程,或者甚至不通过编程来使用REST API的方法。

阅读本章中使用REST API的各种编程语言,需要具备基本的编程语言基础知识。但本章并不强调基础的编程语言知识,而是介绍如何快速地使用REST Web服务。

3.1 易于使用

从根本上说, REST API就是由网址组成的。使用REST API的过程基本分为四个步骤(图3.1):

- (1) 构建请求网址。
- (2) 发送请求到ArcGIS Server。
- (3) 接收服务器的响应。
- (4) 解析和服务器响应。

REST API易于学习和应用。这种优势对基于浏览器的编程方面特别重要,这就是REST API能应用于常用的浏览器端编程语言,例如JavaScript, Flex和Silverlight的原因。服务器端编程和传统的台式机编程采用了SOAP API和工具。但是,服务器端和桌面编程使用REST API也是非常简单,有很大的发展潜力。

注: ArcGIS Server JavaScript API和ArcGIS Server Flex API集成了REST API,可以自动地执行上述四个步骤。如果使用JavaScript API或者 Flex API,就不需要自己完成这四个步骤,但阅读本节仍然有助于了解一些背景知识。

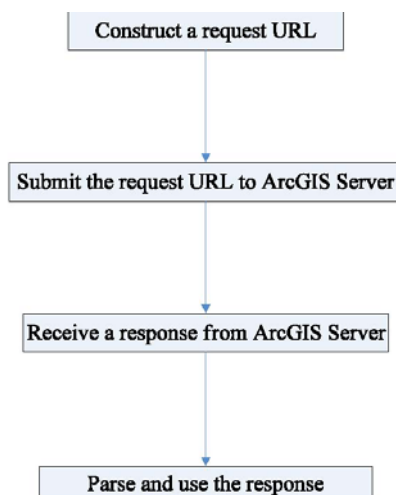


图 3.1使用ArcGIS Server REST API步骤

1、构建请求 URL。

- A、确定节点：每个 GIS 服务都有一个节点，正如在第 2 章 2.4.4 节“获取开发信息”所述。例如，ArcGIS Server 上 Demographics 文件夹下名为 ESRI_Census_USA 的一个地图服务 sampleserver1.arcgisonline.com 的节点为
http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer。
- B、确定操作：不同地理信息系统服务支持不同的操作。不同的操作会返回不同的结果。地图服务可以地图输出，点击查看，查找和生成 KML。输出地图可以生成地图，同时可以点击查看是否给出地图服务图层的属性表。
- C、确定参数：不同的操作需要不同的参数。例如，如果请求地图图片，需要提供地图范围的四周角点坐标参数，也就是地图覆盖范围。
- D、确定输出格式：REST API 支持很多输出格式，例如 JSON,KMZ, 图片和 HTML。

基于这一考虑，我们需要构建的 URL 就是：`http://{ArcGIS Server name}/ArcGIS/rest/services/{folder name}/{service name}/{service type}/{operation}?{parameter1}={some values}& parameter2={some values}`

- 2、提交 URL 请求到 ArcGIS Server Sending，可以不通过编程发送 URL 请求。例如，只需在网页浏览器的地址栏中输入网址，如 Internet Explorer 或 Firefox。每种编程语言都有不同的提出请求方式。详细资料可参考本章后续内容。
- 3、接收 ArcGIS Server 的响应，ArcGIS Server 处理请求并返回响应到客户端。对于一个同步的工作，客户端一直等待到收到服务器的响应。对于异步工作，服务器发送一份工作编号来定期跟踪客户端的工作状态。
- 4、解析和使用响应，ArcGIS Server REST Web 服务的响应可以是多种格式，例如 JSON,KMZ, 图片和 HTML。客户端可判断响应是成功还是失败。如果失败了，客户端可以判断错误消息。如果响应是成功的，客户端可以解析响应所需的信息（例如，输出地图图像，几何特征和属性等），并恰当地利用这些信息（例如，显示在地图的影像，高亮显示结果要素等）。

请记住，不需要将ArcGIS Server在本地计算机上安装，就可以应用ArcGIS REST API编程。你只需要有一个或多个通过网址访问的ArcGIS Server提供的服务。

ESRI公司提供了一个服务器范例，你可以用来练习使用REST API。服务网址是：
<http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/>。

3.2 不编程使用 REST API

因为无需编程就可以使用REST API，所以并非只有程序员才能使用REST API。使用REST API，所有需要做的就只是建立一个请求网址。只要知道如何建立请求的网址，就可以在一些Web浏览器和地理浏览器中查看结果。我们已在第2章2.4.6节，“支持的输出格式”部分，列出了支持的这些格式。例如，我们可以通过REST API使用下面的地图服务：

- ArcGIS Server: <http://sampleserver1.arcgisonline.com/arcgis/rest/services>。
- Portland 地图服务。
- 名为 ESRI_LandBase_WebMercator 的地图服务。
- 这个地图服务的节点是
http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer。

在这一节中，我们将展示如何利用现有的地理浏览器使用REST API。这些地理浏览器可以通过REST API与ArcGIS Server进行通讯。

3.2.1 ArcGIS Server JavaScript 地图浏览器

对于ArcGIS Server JavaScript地图浏览器而言，输出格式的“F”参数应该是“jsapi”。因此，我们构建的网址如下：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=jsapi

点击网址链接，你就可以看到使用ArcGIS JavaScript API的Web浏览器中的地图服务。你可以与地图浏览器交互（图3.2）。地图浏览器通过REST API与ArcGIS Server通讯。

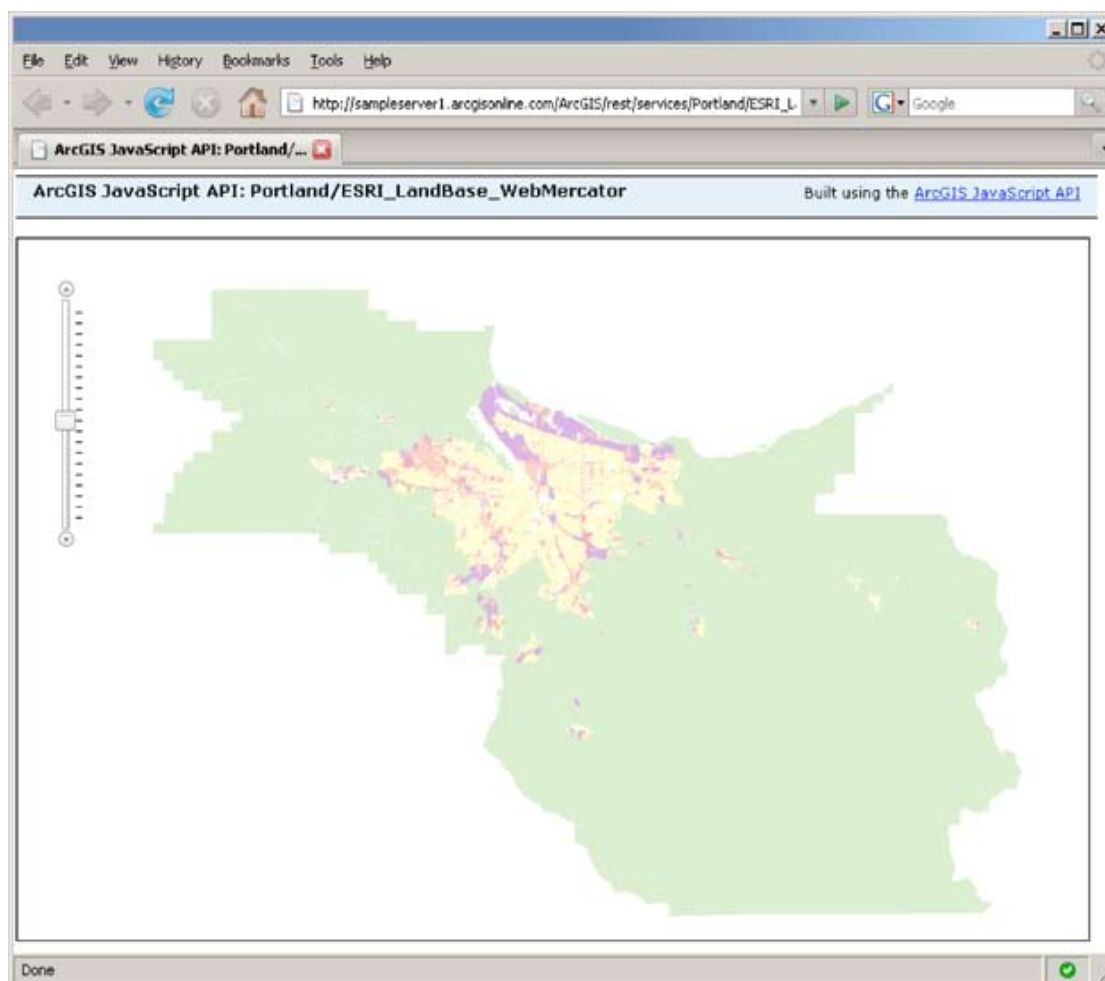


图3.2 在ArcGIS Server JavaScript地图浏览器中查看地图服务

3.2.2 ArcGIS Explorer

利用ArcGIS Explorer，输出格式的参数f=nmf。或者，您可以使用f=kmz。我们构建的网址就应该是

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=nmf

如果电脑已安装ArcGIS Explorer，那么the.nmf文件默认情况下直接展示在ArcGIS Explorer中。只需点击前面的网址链接，就可以在ArcGIS Explorer中看到地图（图3.3）。

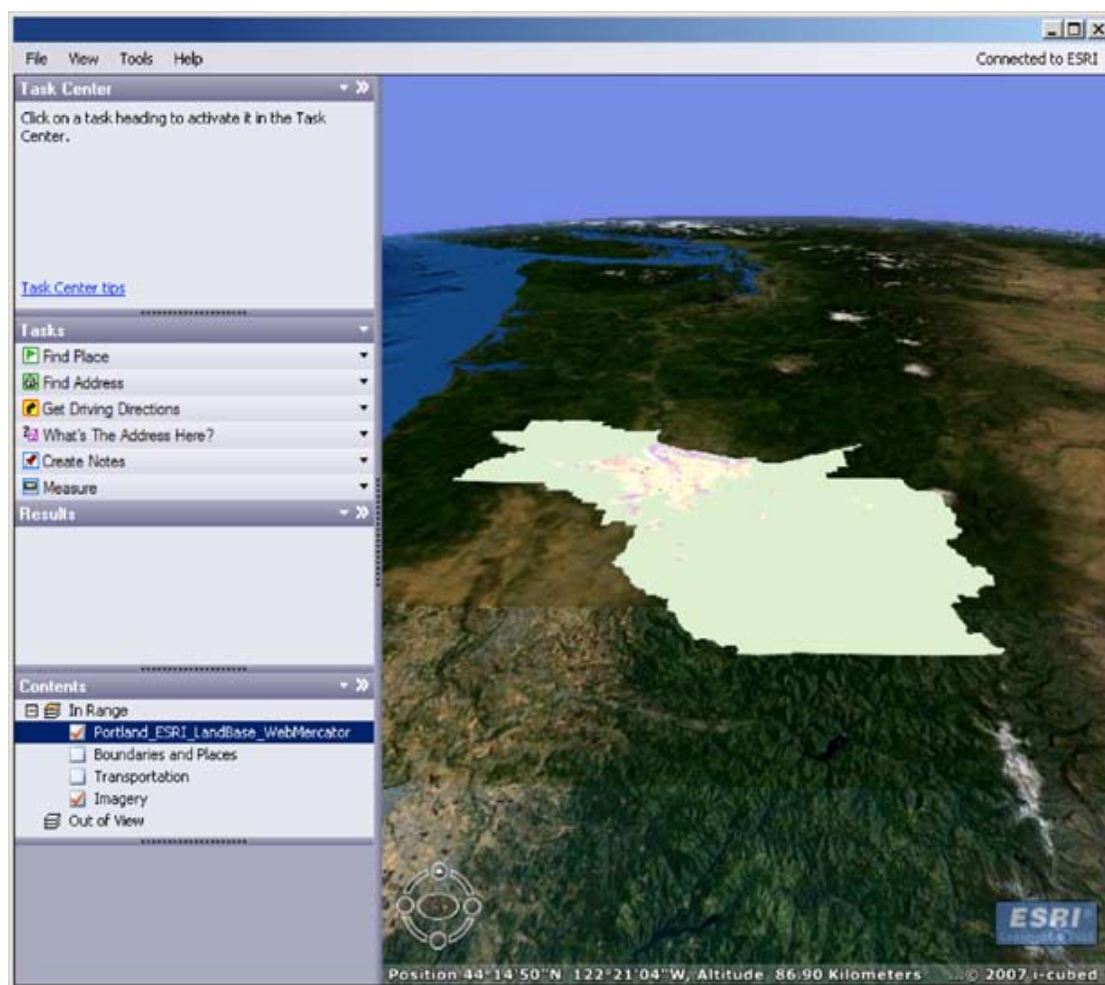


图3.3 在ArcGIS Explorer中查看地图服务

3.2.3 ArcMap

对于ArcMap，输出格式参数 `f=lyr`。此外，我们还需要指定ArcMap的版本参数。例如，对于ArcMap9.2版。我们构建的网址是

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=lyr&v=9.2

如果电脑已安装有ArcMap，默认情况下.lyr文件类型注册到ArcMap打开。只需点击前面的网址链接，就可以在ArcMap视图中看到地图（图3.4）。

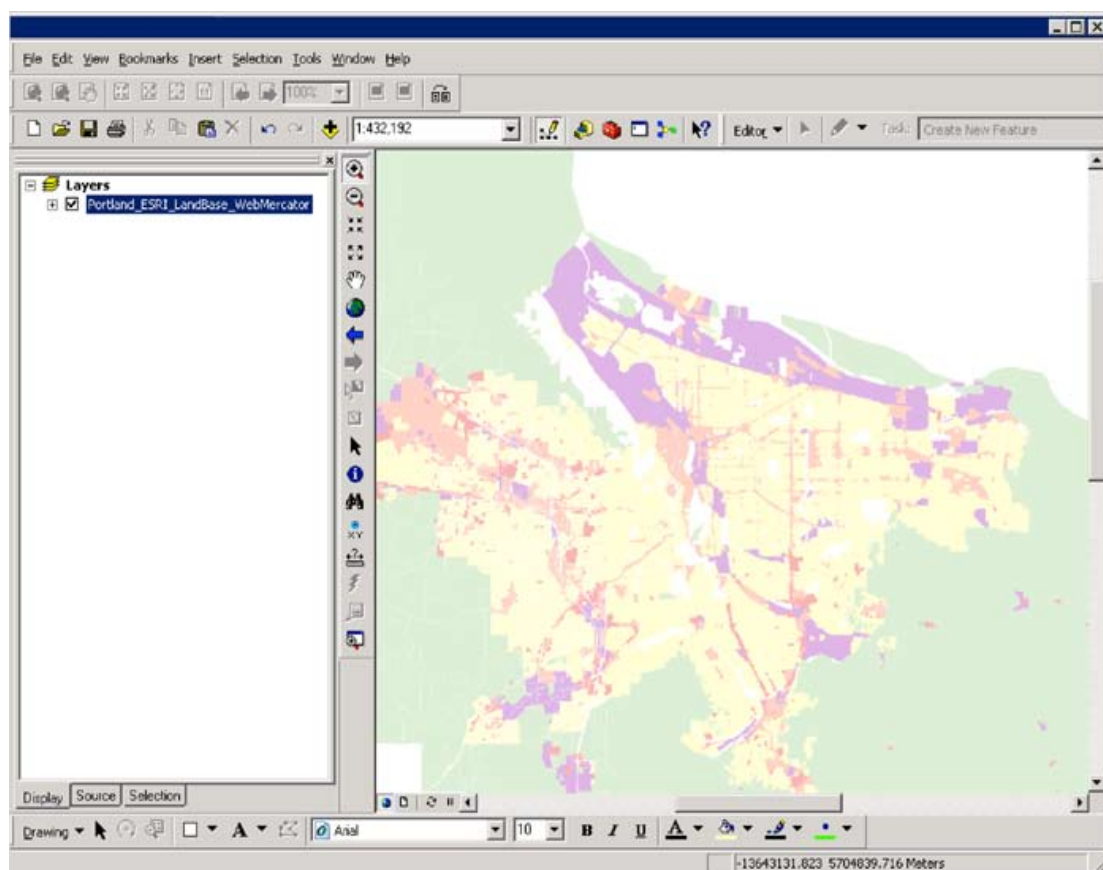


图3.4 在ArcMap中查看地图服务

3.2.4 微软虚拟地球

如果不通过编程在微软虚拟地球中使用REST API，则地图服务必须是缓存地图，而且其投影必须是Web莫卡托投影（102113）。我们创建的示例地图服务“ESRI_LandBase_WebMercator”就符合上述要求。

对于虚拟地球，输出格式的参数f=ve。我们构建的网址是

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=ve

对于二维虚拟地球，你只需点击前面的网址链接，就可以与地图服务交互。但如果要浏览三维地图，需要额外安装三维插件。

3.2.5 谷歌地图

不通过编程在谷歌地图中使用REST API，地图服务必须是缓存地图，并且其投影必须是Web莫卡托投影（102113）。我们创建的示例地图服务“ESRI_LandBase_WebMercator”就符合这个要求。

对于谷歌地图，输出格式参数**f=gmaps**，或者可以使用**f=kml**。我们构建的网址是：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer?f=gmaps

只要点击上述网址链接，你就可以在Web浏览器看到谷歌地图上的地图服务。

3.2.6 谷歌地球

ArcGIS Server REST API可以生成一个.kmz文件，作为谷歌地球的底面叠加层。构建的网址可以如下：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer/kml/mapImage.kmz

如果计算机已安装谷歌地球，KMZ就注册到谷歌地球打开，只需点击前面的网址链接。如果没有安装，则需要安装谷歌地球，并点击打开文件，选择上述的网址。网址请求被提交，然后才可以使用所产生的交互式的谷歌地球地图。

3.2.7 利用 Web 浏览器

通过REST API，ArcGIS JavaScript API就可以在地图浏览器，虚拟地球，谷歌地图等地理浏览器中查看地图服务。但即使没有任何地理浏览器，也可以在Web浏览器中使用地图服务。

为了做到这一点，需要指定下列参数：

- 操作：输出
- 地图边界框（取值格式：西，南，东，北）：**bbox=-13652886,5709630,-13646905,5713428**
- 输出图像尺寸：尺寸= **600×400**

现在，我们把上述信息按照以下格式构建网址（第3.1节已介绍）：

<http://{ArcGIS Server name}/ArcGIS/rest/services/{folder name}/{service name}/{service type}/{operation}?{parameter1}={some values}& parameter2={some values}>

我们构建的网址是：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Portland/ESRI_LandBase_WebMercator/MapServer/export?bbox=-13652886,5709630,-13646905,5713428&size=600,400&f=image

只要点击链接，或者在Internet Explorer，Firefox或其他Web浏览器的地址栏中复制/粘贴。浏览器提交请求，得到ArcGIS Server的响应，我们就可以看到显示结果（图3.5）。

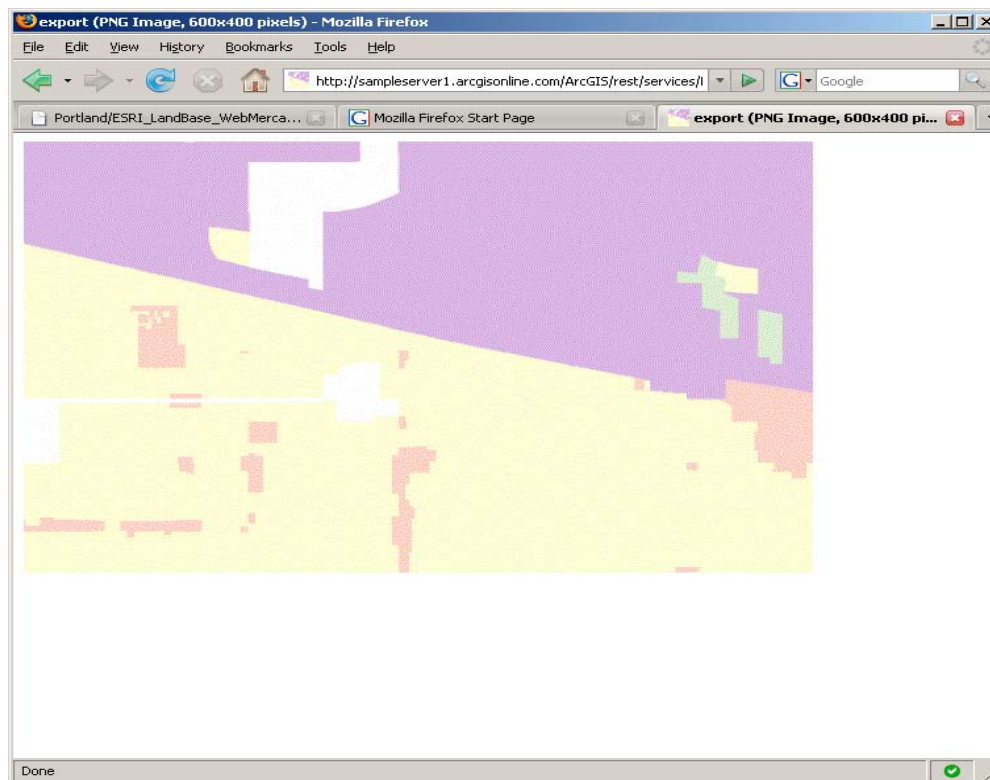


图3.5 f=image直接在Web浏览器中浏览地图

3.3 基于浏览器端编程使用 REST API

ArcGIS Server提供ArcGIS Server JavaScript和ArcGIS Server Flex APIs，来构建REST的请求网址和解析请求的响应，从而能够更容易地完成工作。此外，还可以处理用户交互，如鼠标点击事件，地图和图形显示等。我们建议你在应用程序中使用ArcGIS Server JavaScript APIs或者ArcGIS Server Flex APIs。

3.3.1 JavaScript

JavaScript是最常用的编程语言之一。在JavaScript中使用REST API ArcGIS Server JavaScript API是一个很好的选择。

3.3.1.1 使用 ArcGIS Server JavaScript API

ArcGIS Server JavaScript API是建立在ArcGIS Server REST API基础上。JavaScript API是将地理信息系统的地图和任务嵌入到Web应用程序的一种轻量级方式。该API可以免费使用和部署到应用程序。

利用JavaScript API，可以实现如下工作：

- 使用 ArcGIS 地图服务交互式显示用户数据的地图。
- 基于某些鼠标事件绘制图形或弹出窗口提供信息。
- 使用地理处理服务，执行服务器上的一个地理信息系统模型，并显示模型结果。
- 在 ArcGIS Online 基础地图上显示用户的数据。
- 在用户的地理信息系统数据中搜索要素或属性，显示结果。
- 使用地理编码服务，搜索地址，并显示结果。

关于JavaScript API的详细帮助，可以访问ESRI公司资源中心：

<http://resources.esri.com/arcgisserver/index.cfm?fa=JSAPIs>。本书中，我们只是利用一个简单的例子来介绍如何通过JavaScript API使用REST技术。

这个例子中，你可以输入一个地址，则显示结果就匹配到该地址的点位。实际上是利用REST API调用地理编码服务：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Find Address</title>
<style type="text/css">
@import "http://serverapi.arcgisonline.com/jsapi/arcgis/1/js/dojo/dijit/themes/tundra/tundra.css";
</style>
<script type="text/javascript" src="http://serverapi.arcgisonline.com/jsapi/arcgis/?v=1"></script>
<script type="text/javascript">
dojo.require("esri.map");
dojo.require("esri.tasks.locator");
var map, locator;
function init() {
map = new esri.Map("map", { extent: new esri.geometry.Extent(-125, 28, -62, 45, new esri.
SpatialReference({wkid:4326})) });
var tiledMapServiceLayer = new
esri.layers.ArcGISTiledMapServiceLayer("http://server.arcgisonline.com/
ArcGIS/rest/services/ESRI_StreetMap_World_2D/MapServer");
map.addLayer(tiledMapServiceLayer);
locator = new esri.tasks.Locator("http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/
ESRI_Geocode_USA/GeocodeServer");
dojo.connect(locator, "onAddressToLocationsComplete", showResults);
}
function locate() {
map.graphics.clear();
var add = dojo.byId("address").value.split(",");
```



```

var address = {
  Address : add[0],
  City: add[1],
  State: add[2],
  Zip: add[3]
}; locator.addressToLocations(address,["Loc_name"]);
}
function showResults(candidates) {
  var candidate;
  var symbol = new esri.symbol.SimpleMarkerSymbol();
  var infoTemplate = new esri.InfoTemplate("Location", "Address: ${address}<br />Score: ${score}<br />Source
  locator: ${locatorName}");
  symbol.setStyle(esri.symbol.SimpleMarkerSymbol.STYLE_DIAMOND);
  symbol.setColor(new dojo.Color([255,0,0,0.75]));
  var points = new esri.geometry.Multipoint(map.spatialReference);
  for (var i=0, il=candidates.length; i<il; i++) {
    candidate = candidates[i];
    if (candidate.score > 70) {
      var attributes = { address: candidate.address, score:candidate.score, locatorName:candidate.attributes.
      Loc_name };
      var graphic = new esri.Graphic(candidate.location, symbol, attributes, infoTemplate);
      map.graphics.add(graphic);
      map.graphics.add(new esri.Graphic(candidate.location, new
      esri.symbol.TextSymbol(attributes.address).
      setOffset(0, 8)));
      points.addPoint(candidate.location);
    }
  }
  map.setExtent(points.getExtent().expand(3));
}
dojo.addOnLoad(init);
</script>
</head>
<body class="tundra">
  Address : <input type="text" id="address" size="60" value="380 New York St, Redlands, CA, 92373" />
  <i>(Address, City, State, Zip)</i>
  <input type="button" value="Locate" onclick="locate()" /><br />
  <br />
  <div id="map" style="width:1200px; height:600px; border:1px solid #000;"></div>
</body>

```

在ArcGIS JavaScript API中，用户可以使用Locator类进行地理编码。Locator类构建

器需要ArcGIS Server的地理编码服务的网址。本实例使用ESRI示例服务器的ESRI_Geocode_USA服务。用户可以使用服务目录发现网址，并应用到自定义的服务中。当用户点击Locate按钮，Locate功能就被调用。该功能将用户输入的地址文本解析为一个四项矩阵。四项矩阵（街道地址，城市，州和邮政编码）对应locator中定义的地址属性。

在服务目录中查询地理编码服务就可以发现这些地址属性。例如，网页显示本实例中的四个地址属性。

当addressToLocations方法被调用后，JavaScript API构建一个REST请求网址如下所示：

```
http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/ESRI_Geocode_USA/GeocodeServer/findAddressCandidates?f=json&outFields=Loc_name&Address=380%20New%20York%20St&City=%20Redlands&State=%20CA&Zip=%2092373&callback=dojo.io.script.jsonp_dojoloScript2._jsonpCallback
```

网址中的值是网址编码。解释为：

- 服务节点：
http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/ESRI_Geocode_USA/GeocodeServer
- 操作：findAddressCandidates
- 输出格式：f=json
- 输出属性：outFields = Loc_name
- 地址：地址= 380 New York St
- 城市：城市=雷德兰兹
- 国家：州=加利福尼亚
- 邮编：邮政编码= 92373

JavaScript API构建REST请求网址并传送到服务器。当locator完成匹配后，它实际上将JSON返回以下响应：

```
dojo.io.script.jsonp_dojoloScript2._jsonpCallback({"candidates":[{"address":"380 NEW YORK ST, REDLANDS, CA, 92373","location":{"x":-117.195681386,"y":34.0575170970001},"score":100,"attributes":{"Loc_name":"Street_Address"}}, {"address":"458 NEW YORK ST, REDLANDS, CA, 92373","location":{"x":-117.19569688,"y":34.0585149200002},"score":60,"attributes":{"Loc_name":"Street_Address"}}, {"address":"298 NEW YORK ST, REDLANDS, CA, 92373","location":{"x":-117.19565025,"y":34.0555536900001},"score":60,"attributes":{"Loc_name":"Street_Address"}}]}
```

```

t_
Address"}}, {"address": "474 NEW YORK ST, REDLANDS, CA,
92373", "location": {"x": -117.19569396, "y": 34.058597
1300001}, "score": 60, "attributes": {"Loc_name": "Street_Address"}}, {"address": "500 NEW YORK ST,
REDLANDS,
CA,
92373", "location": {"x": -117.19572503, "y": 34.0592554100002}, "score": 59, "attributes": {"Loc_name": "Stree
t_
Address"}}, {"address": "198 NEW YORK ST, REDLANDS, CA,
92373", "location": {"x": -117.19562531, "y": 34.053680
8300001}, "score": 58, "attributes": {"Loc_name": "Street_Address"}}, {"address": "98 NEW YORK ST,
REDLANDS,
CA, 92373", "location": {"x": -117.19560231, "y": 34.05197162}, "score": 55, "attributes": {"Loc_name": "Street_
Address"}}, {"address": "NEW YORK
ST", "location": {"x": -117.19572988, "y": 34.0595619300001}, "score": 49, "attributes": {"
Loc_name": "Street_Address"}}, {"address": "92373", "location": {"x": -117.186535, "y": 34.0385580000001}, "
score": 100, "at
tributes": {"Loc_name": "Zipcode"}}, {"address": "REDLANDS,
CA", "location": {"x": -117.171316, "y": 34.0502740000001}, "s
core": 100, "attributes": {"Loc_name": "CityState"}}}}];

```

JavaScript API使用事件监听器来监测服务器的响应。当收到请求的响应，就调用 `showResults` 功能，添加得分超过80的候选地址到地图中。候选地址由一系列的 `AddressCandidate` 对象传递给监听的回调函数。

每个候选地址包括地理位置点，这样就可以将其绘制到地图中作为新的图形（图3.6）。

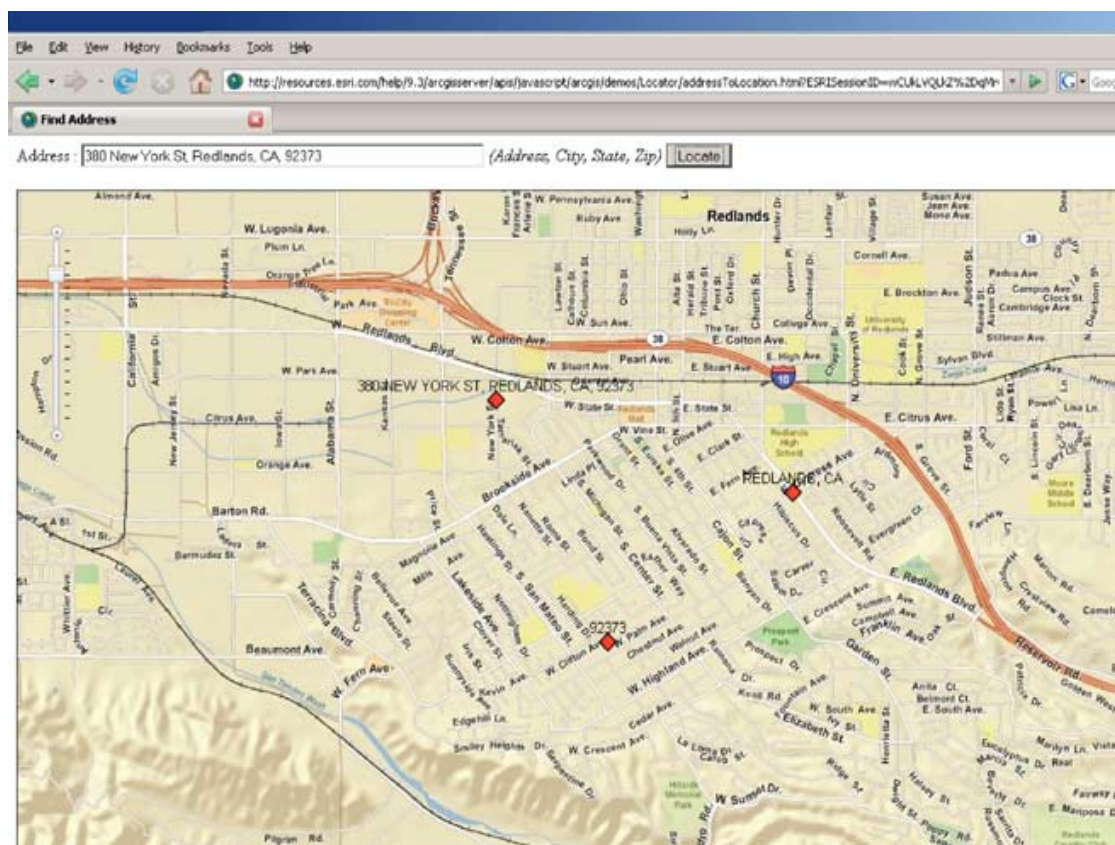


图 3.6 使用ArcGIS Server JavaScript API进行地理编码

3.3.1.2 直接应用 JavaScript

正如我们以前所述，ArcGIS Server JavaScript API封装了REST API和其它很多操作。例如，它处理用户的互动，并显示互动的结果。我们推荐用户优先使用JavaScript API。

使用JavaScript API通常要求，用户的Web浏览器能够访问本地服务器上的JavaScript API库。JavaScript API提供了大量的功能。

本节将演示如何直接在JavaScript中使用REST API。下面的例子中，对输入的地址进行编码，以表格的形式显示结果。

我们使用与第3.3.1.1节中例子相同的服务，就是 http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/ESRI_Geocode_USA/GeocodeServer。点击链接就会看到需要输入的地址字段，支持的操作，以及候选地址列，可返回ArcGIS Services目录。例如，地理编码一个地址：

- 服务端点：
http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/ESRI_Geocode_USA/GeocodeServer
- 操作是 findAddressCandidates
- 输入地址有四列：地址，市，州和邮政编码（邮编）

现在，我们需要写一个JavaScript程序，使用第3.3.1.1节中介绍的格式构建一个网址：

```
<html>
<head>
<title>Geocode using ArcGIS Server REST API</title>
<script language="javascript">
var geocodeService="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Locators/
ESRI_Geocode_USA/GeocodeServer";
var geocodeOperation="findAddressCandidates";
var outputFormat="f=json";
function callGeocode() {
var geocodeFrm=document.forms["geocodeForm"];
var street=geocodeFrm.street.value;
var city=geocodeFrm.city.value;
var state=geocodeFrm.state.value;
var zip=geocodeFrm.zip.value;
// Here we build the REST request URL
var RequestURL=geocodeService+"/"+geocodeOperation+"?" +outputFormat+"&"
+ "Address=" + street + "&"
+ "city=" + city + "&"
+ "state=" + state + "&"
+ "zip=" + zip ;
geocodeFrm.restURL.value=RequestURL;
var xmlhttp=GetXmlHttpRequest();
xmlhttp.onreadystatechange= function() {
if (xmlhttp.readyState==4) {
if (xmlhttp.status==200) {
// Here we process the response
var responseText=xmlhttp.responseText;
var result=eval('(' + responseText + ')');
geocodeFrm.loc_name.value=result.candidates[0].address;
geocodeFrm.x.value=result.candidates[0].location.x;
geocodeFrm.y.value=result.candidates[0].location.y;
geocodeFrm.score.value=result.candidates[0].score;
} else {
alert("An error has occurred");
}
}
}
```

```

}
// This program runs fine in Internet Explorer, but not in Firefox.
// To work around the cross-domain security control in Firefox,
// you can write a local proxy program, and submit the request via this proxy
// xmlhttp.open("GET","proxy.aspx?" + RequestURL,true);
xmlhttp.open("GET",RequestURL,true); //submit the URL request
xmlhttp.send(null);
}
function GetXmlHttpRequest() {
var xmlhttp=null;
try {
// Firefox, Opera 8.0+, Safari
xmlhttp=new XMLHttpRequest();
} catch (e) {
// Internet Explorer
try {
xmlhttp=new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
}
if (xmlhttp==null) {
alert ("Your browser does not support AJAX!");
return;
}
return xmlhttp;
}
</script>
</head>
<body>
<h3>Geocode using ArcGIS Server REST API in JavaScript</h3>
<form id="geocodeForm">
<table>
<tr><td colspan="2"><b>Input Address</b></td></tr>
<tr><td>Street Address:</td><td><input id="street" size="20" value="380 New York St" /></td></tr>
<tr><td>City:</td><td><input id="city" size="20" value="Redlands" /></td></tr>
<tr><td>State:</td><td><input id="state" size="20" value="California" /></td></tr>
<tr><td>Zip:</td><td><input id="zip" size="20" value="92373" /></td></tr>
<tr><td colspan="2">
<input id="btn" type="button" value="Submit" onclick="callGeocode()" /></td></tr>
<tr><td colspan="2"><b>The REST Request URL is:</b><br>
<textarea id="restURL" rows="4" cols="80" /><br>
</td></tr>

```

```

<b>Output -- Best match result:</b><br>

</td></tr>
<tr><td>Address:</td><td><input id="loc_name" size="60" /></td></tr>
<tr><td>Longitude:</td><td><input id="x" size="40" /></td></tr>
<tr><td>Latitude:</td><td><input id="y" size="40" /></td></tr>
<tr><td>Score:</td><td><input id="score" size="40" /></td></tr>
</table>
</form>
</body>

</html>

```

JavaScript通常使用XMLHttpRequest或XMLHTTP发送请求和接收响应。你可以复制上述代码，在文本编辑器保存为HTML文件。如果在Internet Explorer中运行HTML，点击提交按钮。将会看到JavaScript构建的REST API，以及从ArcGIS Server地理编码服务返回的匹配地址（图3.7）。

注意：该段程序在Internet Explorer中能够很好地运行，但不能在Firefox中运行。这是因为Firefox浏览器拥有强大的跨域安全控制，可以阻止JavaScript提交URL请求到一个不同网域的服务器。为解决此问题，需要用户编写一个本地代理程序，通过代理提交请求。

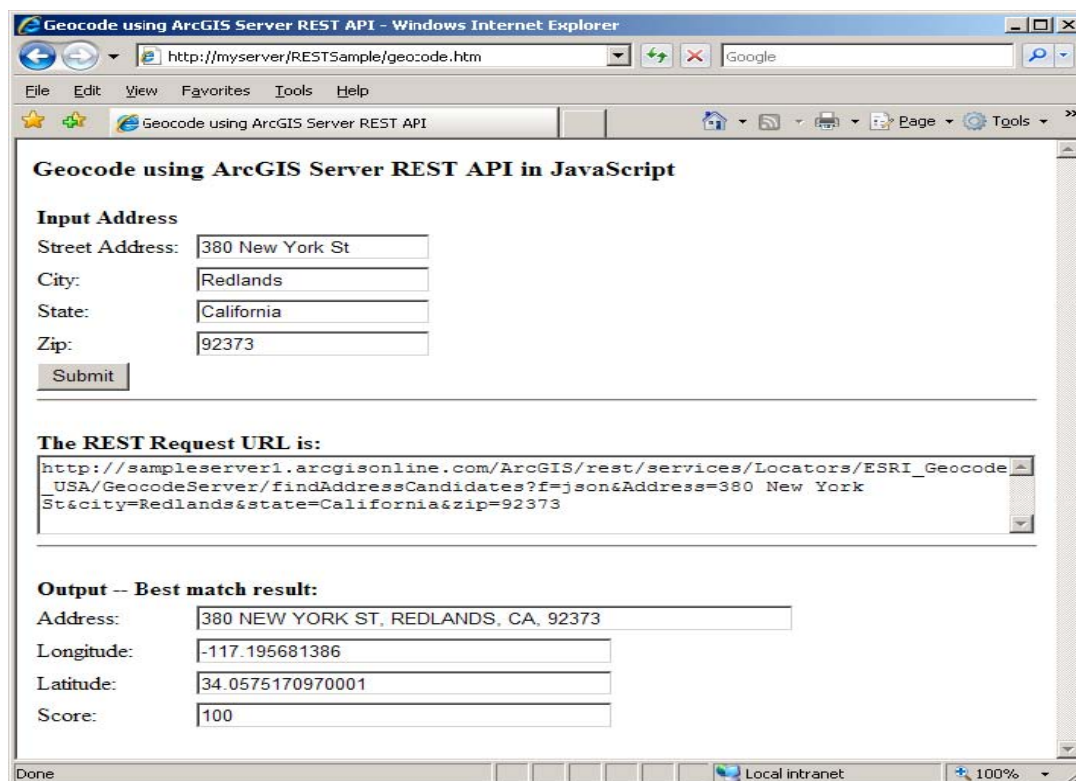


图3.7 通过REST使用ArcGIS Server Locator Service 的Html网页示例

3.3.2 利用 Flex

Flex是Adobe Systems针对跨平台开发和部署而发布的技术集，应用程序可运行在Adobe Flash，Web浏览器或Adobe AIR及其他桌面应用程序。

应用Flex开发的程序可以运行在大多数的已经安装Adobe Flash Player的 Web浏览器中。

ArcGIS Server提供Flex API。与ArcGIS Server JavaScript API类似，ArcGIS Server提供的Flex API不仅封装了REST API，也可以处理用户互动，如鼠标点击事件、地图和图形显示等。我们建议你优先使用ArcGIS Server Flex API。

3.3.2.1 使用 ArcGIS Server Flex API

ArcGIS Server Flex API允许你将ArcGIS Server地图和任务应用到Web应用程序。

ArcGIS Server Flex API提供的功能与ArcGIS Server JavaScript API类似。例如，利用ArcGIS Server Flex API可以实现：

- 互动显示用户数据的地图。
- 执行服务器上的一个地理信息系统模型，并显示模型结果。
- 在 ArcGIS Online 基础地图上显示用户数据。
- 查找用户地理信息系统数据的要素或属性，显示查找结果。
- 搜索地址，并显示搜索结果。
- 创建 Mashups（整合来自多个网站源的信息）。

详细的ArcGIS Server Flex API帮助，请访问ESRI资源中心

<http://resources.esri.com/arcgisserver/apis/flex>。

在这里，我们只是展示一个简单的例子，它可以让你使用地图服务，执行查询州的人口任务，并显示查询结果：

```
<?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="Query Task"
>
<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;
private function doQuery() : void
{
    queryTask.execute( queryMap, new AsyncResponder( onResult, onFault ));
    function onResult( featureSet : FeatureSet, token : Object = null ) : void
    {
        for each ( var myGraphic : Graphic in featureSet.features )
        {
            // ToolTip
            myGraphic.toolTip = "The 2007 population of "
            + myGraphic.attributes.STATE_NAME + " was "
            + myNumberFormatter.format(myGraphic.attributes.POP2007)
            + "\nMedian Age:" + myGraphic.attributes.MED_AGE + ".";
            // show on map
            myGraphicsLayer.add( myGraphic );
        }
    }
    function onFault( info : Object, token : Object = null ) : void
    {
        Alert.show( info.toString() );
    }
}
]]>
</mx:Script>
<mx:NumberFormatter id="myNumberFormatter"
useThousandsSeparator="true"/>
<!-- Layer with US States -->
<esri:QueryTask id="queryTask" url="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/
Demographics/ESRI_Census_USA/MapServer/5">
<esri:Query id="queryMap"
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>
<mx:Panel title="Query a layer (search for a state)" layout="horizontal">
<mx:TextInput width="100%" id="qText" enter="doQuery()"
text="California"/>

```

```
<mx:Button label="Do Query" click="doQuery()"/>
</mx:Panel>
<esri:Map>
<esri:extent>

<esri:Extent xmin="-170" ymin="15" xmax="-65" ymax="75"/>
</esri:extent>
<esri:ArcGISTiledMapServiceLayer
url="http://server.arcgisonline.com/ArcGIS/rest/services/NPS_Physical_World_2D/MapServer" />
<esri:GraphicsLayer id="myGraphicsLayer"/>
</esri:Map>

</mx:Application>
```

运行该段代码，需要将源代码复制和粘贴到Flex开发环境，例如，Adobe Flex Builder。你还需要从<http://resources.esri.com/arcgisserver/apis/flex>下载ESRI公司的Flex API库。经过源代码编译，就可以运行这段代码。

指定一个州的名称，Flex 示例将建立一个如下所示的REST请求网址：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/5/query?text=California&f=json&returnGeometry=true&outFields=MED%5FAGE%2CPOP2007

该网址意味着查询地图服务“ESRI_Census_USA”的第五个层，返回加利福尼亚州的坐标边界和两个属性栏：MED_AGE和POP2007。ArcGIS Server Flex API然后发出请求到ArcGIS Server地图服务，将返回州的几何形状和人口数量。

Flex然后利用返回的几何形状，高亮显示查询到的州，并利用工具提示显示该州的人口数量。如果将鼠标移至突出显示的州，就可以通过工具提示显示该州的人口数（图3.8）

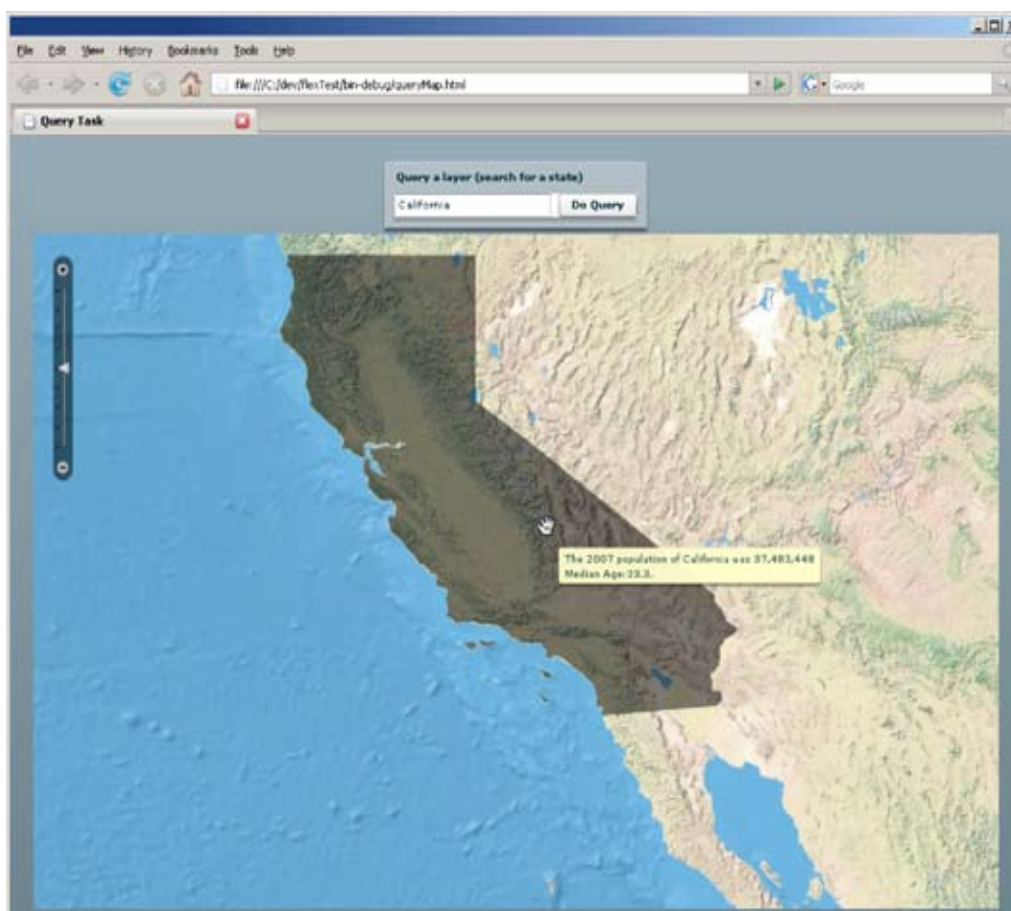


图3.8 通过ArcGIS Server Flex API使用REST API

3.3.2.2 直接使用 Flex

正如我们以前所述，ArcGIS Server Flex API封装了REST API和其它很多工作。例如，它处理用户的互动，并显示结果。我们鼓励你使用ArcGIS Server Flex API。此外，该ArcGIS Server Flex API是免费下载和使用。如果你选择使用Flex，ArcGIS Server Flex API是首选。

但是，如果你出于某种原因，选择直接通过Flex使用REST API，本节将指导你如何做到这一点。下面的例子是在地图服务中查询层，然后将结果按照表格的形式显示。我们将使用与第3.3.2.1相同的层和相同的服务：

http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/5。

点击链接查看可由ArcGIS Services Directory返回的支持操作类型，只要输入的参数，属性列，就可以看到ArcGIS服务目录。查询层：

- 服务节点：
http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/5
- 操作：查询
- 输入参数：
 - ✧ 查询条件= certain_condition
 - ✧ 返回几何体：否
 - ✧ 输出格式：f=json
 - ✧ 输出列AGE_5_17, AGE_18_21, AGE_22_29, AGE_30_39, AGE_40_49, AGE_50_64, AGE_65_UP

现在我们编写Flex程序，使用我们在第3.1节中提出的方式构建URL：

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical">
<mx:Script>
<![CDATA[
import mx.controls.Alert;
import mx.collections.ArrayCollection;
import mx.rpc.events.ResultEvent;
import com.adobe.serialization.json.JSON;
private var s_endPoint:String = "http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/
Demographics/ESRI_Census_USA/MapServer/5";
private var s_operation:String = "query";
private var s_outFields:String =
"AGE_5_17,AGE_18_21,AGE_22_29,AGE_30_39,AGE_40_49,AGE_50_64,AG
E_65_UP";
private var s_format:String="json";

private var s_returnGeometry:String="false";
private function doQuery() : void
{
// construct the URL
var s_requestURL:String=s_endPoint + "/" + s_operation + "?" +
"where=" + escape("state_name =" + stateName.text + "") +
"&outFields=" + s_outFields +
"&returnGeometry=" + s_returnGeometry +
"&f=" + s_format;
// send the request
myHttpService.url=s_requestURL;
myHttpService.send();
}
```

```

private function displayResult(event:ResultEvent) : void
{
// receive the response
var rawData:String = "[" + String(event.result) + "]";
// process the response
var arr:Array = (JSON.decode(rawData) as Array);
// display the response
grid.dataProvider = arr[0].features[0].attributes;
}
]]>
</mx:Script>
<mx:NumberFormatter id="myNumberFormatter"
useThousandsSeparator="true"/>
<mx:Panel title="Query a layer (search for a state)" layout="horizontal">
<mx:TextInput width="100%" id="stateName" enter="doQuery()" text="California"/>
<mx:Button label="Do Query" click="doQuery()"/>
</mx:Panel>
<mx:Label text="Query Result:" fontWeight="bold" fontSize="12"/>
<mx:DataGrid id="grid">
<mx:columns>
<mx:DataGridColumn headerText="STATE" dataField="STATE_NAME"/>
<mx:DataGridColumn headerText="AGE 5-17" dataField="AGE_5_17"/>
<mx:DataGridColumn headerText="AGE 18-21" dataField="AGE_18_21"/>
<mx:DataGridColumn headerText="AGE 22-29" dataField="AGE_22_29"/>
<mx:DataGridColumn headerText="AGE 30-39" dataField="AGE_30_39"/>
<mx:DataGridColumn headerText="AGE 40-49" dataField="AGE_40_49"/>
<mx:DataGridColumn headerText="AGE 50-64" dataField="AGE_50_64"/>
<mx:DataGridColumn headerText="AGE 65-UP" dataField="AGE_65_UP"/>
</mx:columns>
</mx:DataGrid>
<mx:HTTPService id="myHttpService" result="displayResult(event)"/>
</mx:Application>

```

Flex通常使用HTTPService发送请求和接收响应。你可以复制上述代码到Flex开发环境，如Flex Builder，编译并运行该代码。代码一旦运行，请点击“Do Query”按钮。该代码构建一个REST请求地址，传送，并将显示结果在网格中（图3.9）。

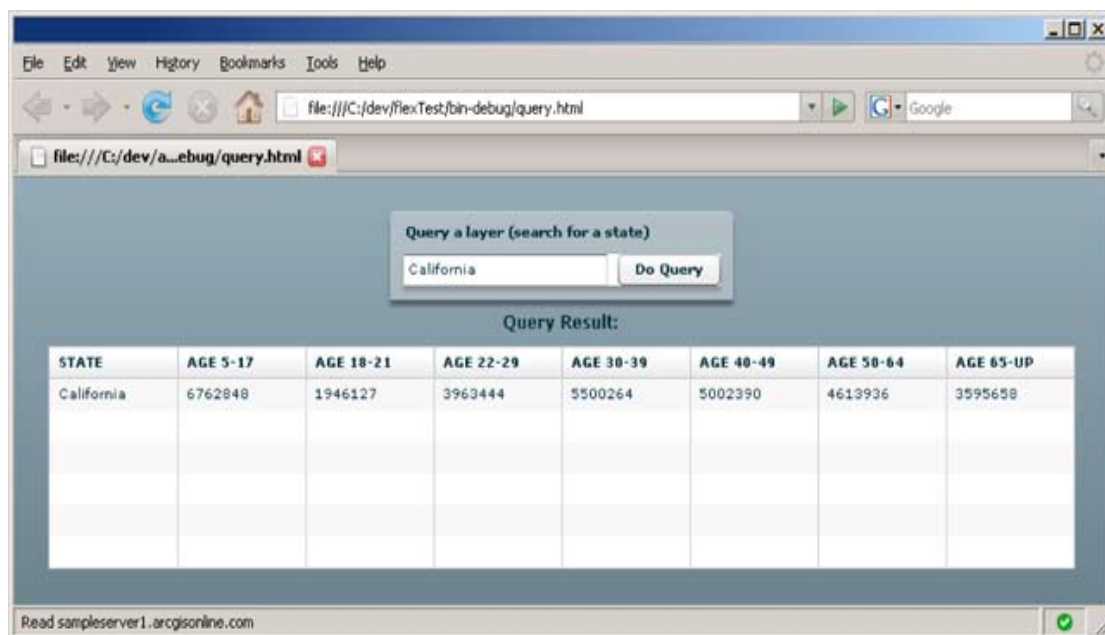


图 3.9 直接在Flex 中使用REST API

3.3.3 利用 Silverlight

Silverlight是HTTP支持的一个浏览器插件。虽然浏览器的安全机制使之无法访问跨域服务，但Silverlight和AJAX代码也足够强大，能够用于REST。Silverlight是一个新的Web技术，可以在多种平台上运行。它能够创造丰富，逼真和交互式的应用程序，可以运行在任何地方，多种浏览器，设备和各种桌面操作系统。与WPF（Windows Presentation Foundation）标准相符，Windows编程基础设施Microsoft.NET框架3.0，XAML（可扩展应用标记语言），是Silverlight表达功能的基础。

你可以添加UserControl到ASP.NET页或HTML网页。完成编译后，它将生成一个XAP文件：

```
<asp:Silverlight ID="Xaml1" runat="server"
Source="~/ClientBin/SilverlightMap.xap"
Version="2.0" Width="100%" Height="100%" />
The UserControl code:
namespace SilverlightMap
{
public partial class Page : UserControl
{
// -180.0,-90.0,180.0,90.0
public double xmin = -180.0;
public double ymin = -90.0;
```

```
public double xmax = 180.0;
public double ymax = 90.0;
Point lastMousePos = new Point();
double _zoom = 1;
bool mouseButtonPressed = false;
bool mouselsDragging = false;
Point dragOffset;
Point currentPosition;
public Page()
{
    InitializeComponent();
    this.MouseMove += delegate(object sender, MouseEventArgs e)
    {
        if (mouseButtonPressed)
        {
            mouselsDragging = true;
        }
        this.lastMousePos = e.GetPosition(this.msi);
    };
    this.MouseLeftButtonDown += delegate(object sender, MouseButtonEventArgs e)
    {
        mouseButtonPressed = true;
        mouselsDragging = false;
        dragOffset = e.GetPosition(this);
        currentPosition = msi.ViewportOrigin;
    };
    this.msi.MouseLeave += delegate(object sender, MouseEventArgs e)
    {
        mouselsDragging = false;
    };
    this.MouseLeftButtonUp += delegate(object sender, MouseButtonEventArgs e)
    {
        mouseButtonPressed = false;
        if (mouselsDragging == false)
        {
            bool shiftDown = (Keyboard.Modifiers & ModifierKeys.Shift) == ModifierKeys.Shift;
            ZoomFactor = 2.0;
            if (shiftDown) ZoomFactor = 0.5;
            Zoom(ZoomFactor, this.lastMousePos);
        }
        mouselsDragging = false;
    };
    this.MouseMove += delegate(object sender, MouseEventArgs e)
```



```

{
if (mouseIsDragging)
{
Point newOrigin = new Point();
newOrigin.X = currentPosition.X - (((e.GetPosition(msi).X - dragOffset.X) / msi.ActualWidth) * msi.
ViewportWidth);
newOrigin.Y = currentPosition.Y - (((e.GetPosition(msi).Y - dragOffset.Y) / msi.ActualHeight) * msi.
ViewportWidth);
msi.ViewportOrigin = newOrigin;
}
};
new MouseWheelHelper(this).Moved += delegate(object sender, MouseEventArgs e)
{
e.Handled = true;
if (e.Delta > 0)
ZoomFactor = 1.2;
else
ZoomFactor = .80;
Zoom(ZoomFactor, this.lastMousePos);
};
}

////////////////////////////////////
public double ZoomFactor
{
get { return _zoom; }
set { _zoom = value; }
}
public void Zoom(double zoom, Point pointToZoom)
{
Point logicalPoint = this.msi.ElementToLogicalPoint(pointToZoom);
this.msi.ZoomAboutLogicalPoint(zoom, logicalPoint.X, logicalPoint.Y);
}
public void GetImage()
{
//Uri getImage = new Uri("http://apascual/target93/ESRI.ArcGIS.ADF.Web.UI.WebControls.
MapHandler.ashx?MapID=Map1&ManagerID=MapResourceManager1&PageID=/target93/Default.
aspx&Operation=DrawExtent&ResourceID=webTierBlendingOfAllResources&Extent=-179.296875,-
70.3125,178.59375,107.578125&Width=1018&Height=506");
string sHost = Application.Current.Host.Source.OriginalString.Substring(0, Application.Current.Host.
Source.OriginalString.IndexOf("ClientBin"));
string sFirstExtend = "http://maps.arcgisonline.com/rest/ESRI_StreetMap_World_2D/MapServer/Layers
/export?bbox=-180.0,-90.0,180.0,90.0&f=json";
string sAll = sHost + "Proxy.aspx?SilverlightUrl=" + sFirstExtend;

```

```

Uri getImage = new Uri(sAll);
System.Net.WebClient client = new System.Net.WebClient();
//client.DownloadStringCompleted += new System.Net.DownloadStringCompletedEventHandler
(client_DownloadStringCompleted);
//client.DownloadStringAsync(getImage);
client.OpenReadCompleted += new System.Net.OpenReadCompletedEventHandler
(client_OpenReadCompleted);
client.OpenReadAsync(getImage);
//"/apascual/target93/ESRI.ArcGIS.ADF.Web.UI.WebControls.MapHandler.ash
x?MapID=Map1&ManagerID=MapResourceManager1&PageID=/target93/Default.
aspx&Operation=DrawExtent&ResourceID=webTierBlendingOfAllResources&Extent=-179.296875,-
70.3125,178.59375,107.578125&Width=1018&Height=506&t=1205950042138"

}
void client_OpenReadCompleted(object sender, System.Net.OpenReadCompletedEventArgs e)
{
    if (e.Error == null)
    {
        Image image = new Image();
        BitmapImage bitmapImage = new BitmapImage();
        bitmapImage.SetSource(e.Result);
        image.Source = bitmapImage;
        image.Stretch = Stretch.Fill;
        image.HorizontalAlignment = HorizontalAlignment.Stretch;
        //MultiScaleImage multiImage = this.Map1.Children[0];
        //multiImage.
        //this.Map1.Children.Add(image);
    }
    else
    {
        string s = e.Error.Message;
    }
}
}
}
}
}

```

3.4 通过服务器端和桌面编程使用 REST API

到目前为止，服务器端和桌面编程基本上都是使用SOAP Web服务。但是，REST风格的网络服务，由于其易于使用，已在服务器端和桌面开发中越来越受欢迎。服务器端和桌面编程的编程语言可以是Python,ASP.Net和Java。

3.4.1 利用 Python

当使用动态数据，Python语言非常适用于REST查询层，并找到所需的信息。下面的例子将查询所有信息层，并在控制台屏幕显示其所有属性：启动它，创建一个test.py文件包括下面的代码，然后从控制台窗口调用“python test.py”。

```
import simplejson, urllib, sys, string
baseUrl =
"http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/3
/query?text=all&geometry=&geometryType=esriGeometryEnvelope&inSR=&spatialRel=esriSpatialRelIntersects
&where=&returnGeometry=true&outSR=&outFields=&f=json"
def loadJSON(url):
request = simplejson.load(urllib.urlopen(url))
return request
results = loadJSON(baseUrl)['results']
for result in results:
print result['id'], result['name']
for attribute in result['attributes']:
print attribute, result['attributes'][attribute]
print result['geometry']['x'], result['geometry']['y']

print
```

你还可以使用以下代码在HTML中找回信息来获得图像：

```
import urllib, sys, string
baseUrl =
"http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/export?bbox=-185.337909357176%2C15.2049923316373%2C-59.5254874993028%2C74.082485035617
6&bboxSR=&layers=&layerdefs=&size=&imageSR=&format=png&transparent=false&dpi=&f=html"
request = urllib.urlopen(baseUrl).read()

print request
```

接下来，从ArcGIS Online下载扩展到HTML页面，包括影像和扩展属性。你就可以剪切和粘贴以下代码，执行代码将返回下列HTML文件：

```
import urllib, sys, string
baseUrl =
"http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/
export?bbox=-185.337909357176%2C15.2049923316373%2C-59.5254874993028%2C74.0824850356176&bboxSR=
```

```
&layers=&layerdefs=&size=&imageSR=&format=png&transparent=false&dpi=&f=html"
request = urllib.urlopen(baseUrl).read()

print request
```

返回如下HTML文件:

```
<html>
<head>
<title>Export Map Image: Layers</title>
<link href='/ArcGIS/rest/ESRI.ArcGIS.Rest.css' rel='stylesheet' type='text/css'>
</head>
<body>
<table width="100%" id="userTable">
<tr>
<td id="titlecell">ArcGIS Services Directory</td>
</tr>
</table>
<table id="navTable" width="100%">
<tbody>
<tr valign="top">
<td id="breadcrumbs">
<a href="/ArcGIS/rest/services">Home</a> >
<a href="/ArcGIS/rest/services/Demographics">Demographics</a>
> <a href="/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer">ESRI_Census_USA
(MapServer)</
a> </td>
<td align="right" id="help">
<a href="http://resources.esri.com/93beta/arcgisserver/apis/rest/servicesexplorer.html"
target="_blank">Help</
a> | <a href="?f=help" target="_blank">API Reference</a></td>
</tr>
</tbody>
</table>
<div class='restHeader'>
<h2>Export Map Image: Layers</h2>
</div>
<div class='restBody'>
<table cellpadding="5" border="0">
<tr valign="top">
<td>

</td>
```

```

<td>
Width: 400<br/>
Height: 400<br/>
Extent:
<ul>
XMin: -186.187993281927<br/>
YMin: -19.11255617006<br/>
XMax: -58.6754035745517<br/>
YMax: 108.400033537315<br/>
Spatial Reference: 4269<br/>
<br/>
</ul><br/>
Scale: 133972136.331439<br/>
</td>
</tr>
</table>
<form>
<table style="border:1px solid #000000;">
</tr>
<tr valign="top">
<td>
Bounding Box:</td>
<td><input type="text" name="bbox" value="-185.337909350544,-19.11255617006,-
59.5254875059344,108.400033537315" size="75"/></td>
</tr>
<tr valign="top">
<td>
Bounding Box Spatial Reference (WKID):</td>
<td><input type="text" name="bboxSR" value=""/></td>
</tr>
<tr valign="top">
<td>
Layers:</td>
<td><input type="text" name="layers" value=""/></td>
</tr>
<tr valign="top">
<td>
Layer Definitions:</td>
<td><input type="text" name="layerdefs" value="" size="75"/></td>
</tr>
<tr valign="top">
<td>
Image Size:</td>

```

```

<td><input type="text" name="size" value=""/></td>

</tr>
<tr valign="top">
<td>
Image Spatial Reference (WKID):</td>
<td><input type="text" name="imageSR" value=""/></td>
</tr>
<tr>
<td>Image Format</td>
<td>
<select name="format">
<option value="png" selected='true'>png</option>
<option value="png8">png8</option>
<option value="png24">png24</option>
<option value="jpg">jpg</option>
<option value="pdf">pdf</option>
<option value="bmp">bmp</option>
<option value="gif">gif</option>
<option value="svg">svg</option>
</select>
</td>
</tr>
<tr>
<td>Background Transparent</td>
<td>
<input type="radio" name="transparent" value="true"/>Yes    
<input type="radio" name="transparent" value="false" checked='true'>No</td>
</tr>
<tr valign="top">
<td>
DPI:</td>
<td><input type="text" name="dpi" value=""/></td>
</tr>
<tr>
<td>Format: </td>
<td>
<select name="f">
<option value="html" selected='true'>HTML</option>
<option value="image">Image</option>
<option value="kmz">KMZ</option>
<option value="pjson">JSON</option>
</select>
</td>
</tr>

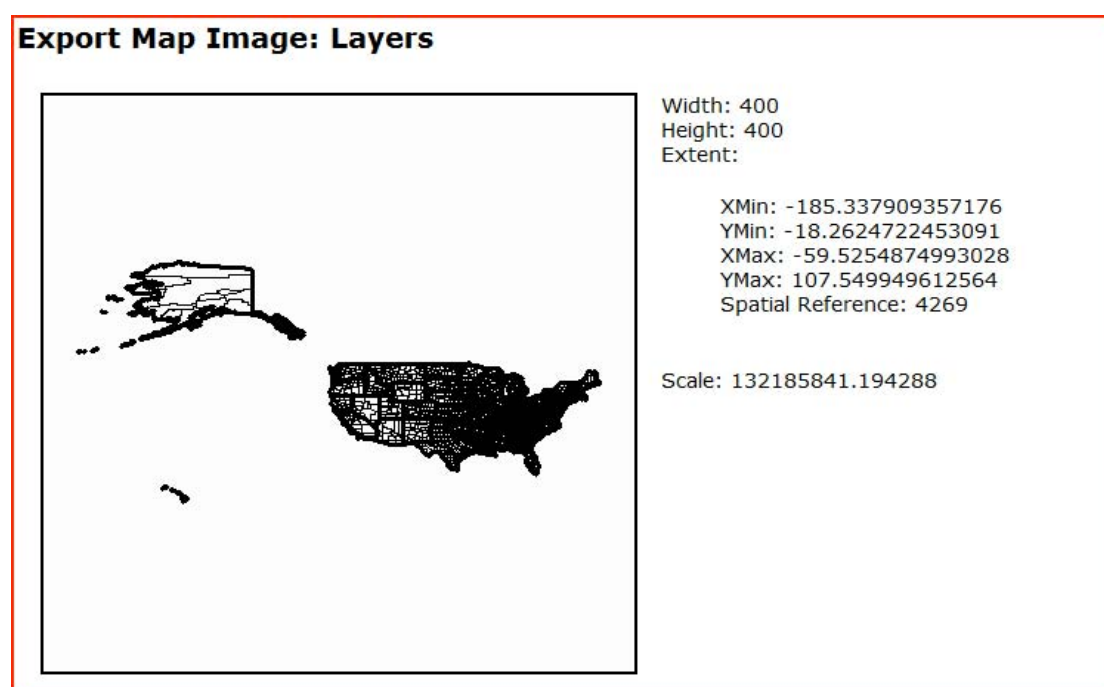
```

```

<tr><td colspan="2" align="left"><input type="submit" value="Export Map Image" /></td></tr>
</table>
</form>
<br/>
</div>
</body>
</html>

```

把上述代码保存为HTML，并在浏览器中打开，将会出现如下所示的HTML网页：



3.4.2 利用 ASP.Net

以下是使用通过ASP.NET/C#代码来使用REST API下载地图影像的示例：

```

protected void Page_Load(object sender, EventArgs e)
{
    string sURL =
    "http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_Imagery_World_2D/MapServer/
    export?bbox=-179.999996642358%2C-131.23152464566%2C179.999996642358%2C131.2315246456
    6&bboxSR=&I
    ayers=&layerdefs=&size=&imageSR=&format=png&transparent=false&dpi=&f=image"
    String sRes = GetRequest(sURL);
    Response.Clear();
    Response.ClearHeaders();
    Response.ClearContent();
    Response.ContentType = "image/png";
    Response.Expires = 0;
}

```



```
Response.Cache.SetCacheability(HttpCacheability.NoCache);
Response.Cache.SetNoServerCaching();
Response.Cache.SetNoStore();
Response.Cache.SetMaxAge(System.TimeSpan.Zero);
MemoryStream msimage = new MemoryStream(imagedata);
msimage.WriteTo(Response.OutputStream);

msimage.Flush();
Response.End();
}
private string GetRequest(string sUrl)
{
    string sRes = "";
    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(sUrl);
    HttpWebResponse response = (HttpWebResponse)request.GetResponse();
    Stream receiveStream = response.GetResponseStream();
    StreamReader readStream = new StreamReader(receiveStream, Encoding.UTF8);
    sRes = readStream.ReadToEnd();
    response.Close();
    return sRes;
}
```

3.4.3 利用 Java

利用Java构建REST请求网址和发送URL请求是很容易的。大多数情况下，我们要求接收JSON格式的响应。有许多免费的Java示例程序或Java程序库，可以解析JSON字符串变成有意义的对象。在这一节中，我们将演示如何查询地图服务的一层，并将结果按照表格的形式显示。功能是与第3.3.2.2节所述的相同，但实现方法是不同的。第3.3.2.2节使用浏览器端的Flex编程，而本节使用服务器端的Java编程。为了更好地说明调用REST API的原则，我们按照Java Server Pages(JSP)的格式编写示例代码。

因此，这个文件包括服务器端和浏览器端代码。浏览器端代码提供结果展示。但在服务器端构建REST请求网址，发送请求，并解析请求。

我们将使用与第3.3.2.2节相同的层和相同的服务，就是
http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/5。

点击链接查看由ArcGIS Services Directory返回的支持操作类型，需要输入的参数，属性列，可以返回ArcGIS服务目录。查询层：

- 服务端点:
http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/ESRI_Census_USA/MapServer/5
- 操作: 查询
- 输入参数:
 - ✧ 查询标准: where = certain_condition。
 - ✧ 返回几何体: 否。
 - ✧ 输出格式: f=json。
 - ✧ 输出列: AGE_5_17, AGE_18_21, AGE_22_29, AGE_30_39, AGE_40_49, AGE_50_64, AGE_65_UP。

现在, 我们需要写JSP程序, 并使用3.1节中提出的规则构建URL地址:

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="org.json.*" %>
<%@ page import="java.io.*" %>
<%@ page import="java.net.URL" %>
<%@ page import="java.net.URLEncoder" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset="UTF-8">
<title>Query US State</title>
</head>
<body>
<%
String stateName = (String)request.getParameter("stateName");
%>
<h3>Query ArcGIS Server for the population information of a State</h3>
<form>
<input type="text" name="stateName" size="20" value='<%=stateName==null?
"California":stateName%>'>
<input type="submit" value="Submit">
</form>
<%if (stateName!=null) {
String s_endPoint="http://sampleserver1.arcgisonline.com/ArcGIS/rest/services/Demographics/
ESRI_Census_USA/MapServer/5";
String s_operation="query";
String
s_outFields="AGE_5_17,AGE_18_21,AGE_22_29,AGE_30_39,AGE_40_49,AGE_50_64,AGE_65_UP"
```

```

;
String s_format="json";
String s_returnGeometry="false";
// construct the URL
String s_requestURL =s_endPoint + "/" + s_operation + "?" +
"where=" + URLEncoder.encode("state_name = " + stateName + "','UTF-8") +
"&outFields=" + s_outFields +
"&returnGeometry=" + s_returnGeometry +
"&f=" + s_format;
// send the request and receive the response
HTTPUtil mHttpUtil=new HTTPUtil();

String s_Result=mHttpUtil.getResponse(s_requestURL);

// parse the response
JSONArray results = new JSONArray("[ " + s_Result + " ]");
JSONArray features=results.getJSONObject(0).getJSONArray("features");
JSONObject fields = new JSONArray("[ " + features.getJSONObject(0).getString("attributes") + " ]").
getJSONObject(0);
%>
<hr>
<h3>Result</h3>
<TABLE border="1">
<TR>
<TD>AGE_5_17</TD><TD>AGE_18_21</TD><TD>AGE_22_29</TD><TD>AGE_30_39</TD>
<TD>AGE_40_49</TD><TD>AGE_50_64</TD><TD>AGE_65_UP</TD>
</TR>
<TR>
<TD><%=fields.getString("AGE_5_17")%></TD>
<TD><%=fields.getString("AGE_18_21")%></TD>
<TD><%=fields.getString("AGE_22_29")%></TD>
<TD><%=fields.getString("AGE_30_39")%></TD>
<TD><%=fields.getString("AGE_40_49")%></TD>
<TD><%=fields.getString("AGE_50_64")%></TD>
<TD><%=fields.getString("AGE_65_UP")%></TD>
</TR>
</TABLE>
<%
}
%>
</body>
</html>
<%!
public class HTTPUtil
{

```

```
public HTTPUtil()
{
}

public String getResponse(String urlString) throws IOException
{
    String str = "";
    try
    {
        URL url = new URL(urlString);
        BufferedReader in = new BufferedReader(new InputStreamReader(new BufferedInputStream(url.
        openStream())));
        StringBuffer sb = new StringBuffer();

        String line;
        while((line = in.readLine()) != null)
        {
            sb.append(line);
        }
        in.close();
        str = sb.toString();
    }
    catch (IOException e)
    {
        e.printStackTrace();
        throw e;
    }
    return str;
}

%>
```

注意，为了解析JSON格式的响应，这个示例中包括了“org.json.*”库，该库可以从<http://www.json.org/java/json.zip>下载。你可以复制上述示例程序到Java开发环境，如NetBeans, JBuilder或者Eclipse，编译并运行。一旦运行，点击提交按钮。则JSP程序建立一个REST请求网址，发送请求，并在一个表中显示结果。

第四章：优化方法

4.1 保证 REST 服务安全

REST API是无状态的，而且可以被许多种类的资源各自调用，必须在单个服务器端设置安全机制。REST技术可用于互联网的上任何人，因此最重要的是要保护你的数据，只有通过验证的使用者才可以检索数据。ArcGIS网络管理器中基于角色的安全机制可以让你更容易地确保Web服务和网络应用以及所有提供服务的安全（图4.1）。

REST API安全机制与ArcGIS Server中安全规则相互配合。通过ArcGIS管理器产生令牌或者登录到服务，可以使所有服务具备安全机制。欲了解更多的ArcGIS Server 9.3的有关安全信息，请访问：<http://webhelp.esri.com/arcgisserver/9.3/dotNet/index.htm>。



图 4.1 ArcGIS Server交互设置REST安全

ArcGIS管理器将产生的令牌分发给在ArcGIS Server上申请REST服务的应用程序。服务许可被存储在服务器对象管理器（SOM）。

服务器可以发布任何具有安全保障或无安全保障的REST服务。基于角色的安全性是基于.NET（ASP.NET）的主动服务器网页安全框架。这可让你随时随地将用户存储到现有的供应商——主动目录（Windows用户）和微软SQL（结构化查询语言）服务器。使用ASP.NET成员提供的框架可以很容易地将安全性扩展到其他数据库。当然，这仅仅是通过建立其安全和配置ArcGIS Server来使用创建一个新的服务供应商。

4.2 改进性能

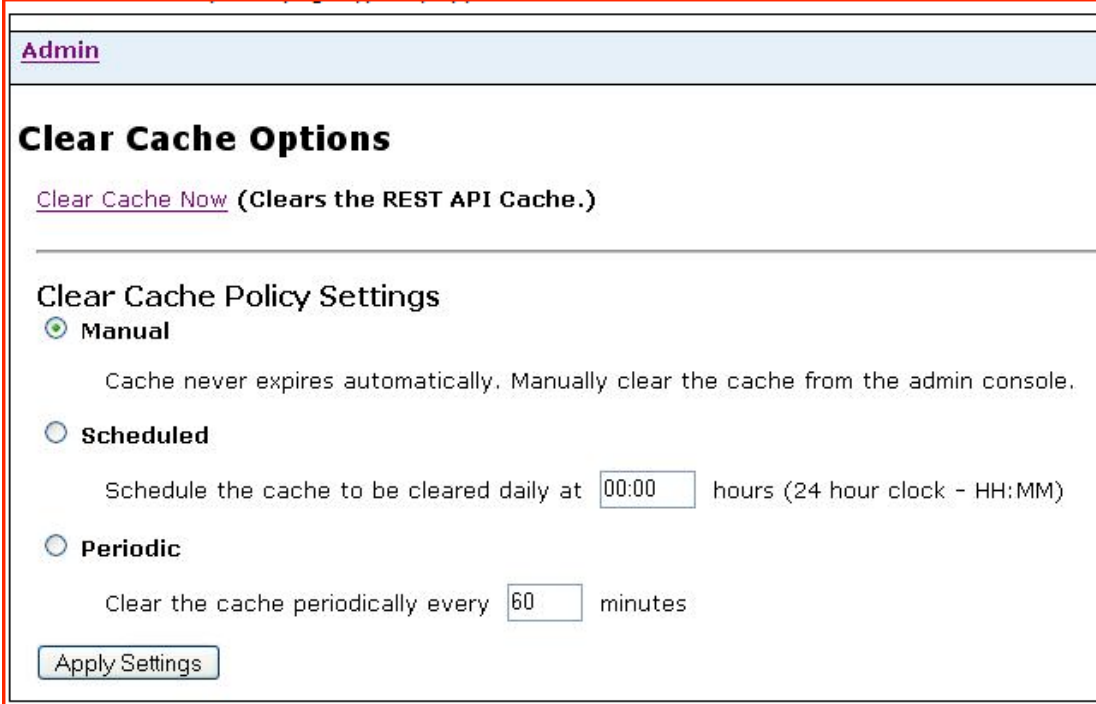
REST服务有一个配置文件（`rest.config`），通过该文件你可以改变所有你想要改变的默认值。请求结果的类型和复杂程度都会影响服务的性能。对大多数的用户来说，速度是至关重要的。这就是利用REST带来的好处。但是，修改其他方面因素也可以达到理想的速度效果。

ArcGIS REST目录下可以发现配置文件。通过配置文件可以设定如何缓存金字塔和地图，是否压缩数据以便于传输和获得何种类型的图片或结果。配置这些ArcGIS/REST的目录属性可以显著改善服务性能。

4.2.1 缓存

缓存是改善REST地图性能的关键技术。实时创建地图是最费力和最耗时的方式，所以从一个仓库或地图高速缓存中找回地图将大大提高性能。最好的方法就是建立一个缓存地图。如果存储空间不受限制，高速缓存是最佳的选择方案。

REST API管理器让你通过服务控制缓存的产生（图4.2）。你可以决定是否将其手动或自动清除。手动清除缓存的方式更好，因为清除时可以控制，从而可以建立一个较大的缓存。



The screenshot displays the 'Admin' section of the REST API Manager. Under the heading 'Clear Cache Options', there is a link 'Clear Cache Now (Clears the REST API Cache.)'. Below this, the 'Clear Cache Policy Settings' section shows three radio button options: 'Manual' (selected), 'Scheduled', and 'Periodic'. The 'Manual' option is accompanied by the text 'Cache never expires automatically. Manually clear the cache from the admin console.' The 'Scheduled' option is accompanied by the text 'Schedule the cache to be cleared daily at 00:00 hours (24 hour clock - HH:MM)'. The 'Periodic' option is accompanied by the text 'Clear the cache periodically every 60 minutes'. At the bottom of the settings section is an 'Apply Settings' button.

图4.2 REST admin管理缓存删除方式

4.2.2 压缩

压缩能够加快数据的下载速度，从而改进性能。REST技术在rest.config文件中设置压缩方式。默认情况下，REST设置为压缩模式。

4.2.3 图像格式

REST API可让你选择GIS服务器使用的金字塔格式。默认情况下，图像格式是PNG，一种轻便的网络图形（图4.3）。然而，你也可以选择PNG8或PNG24，这取决于你想要8位或24位的PNG、压缩的JPEG格式JPG、Joint Photographic Experts Group（图4.4）、便携式文档格式PDF或位图格式BMP（图4.5）；图形交换格式GIF或可伸缩矢量图形SVG。

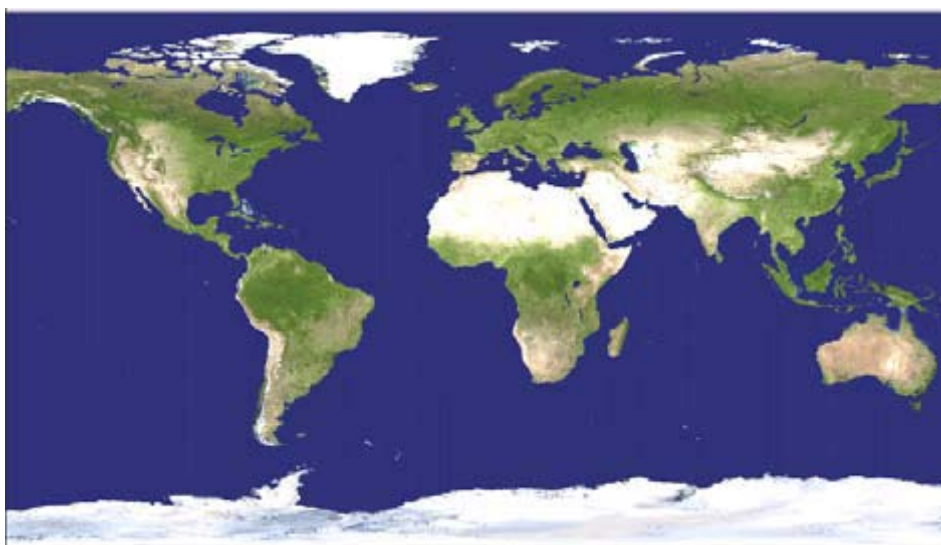


图 4.3 PNG格式

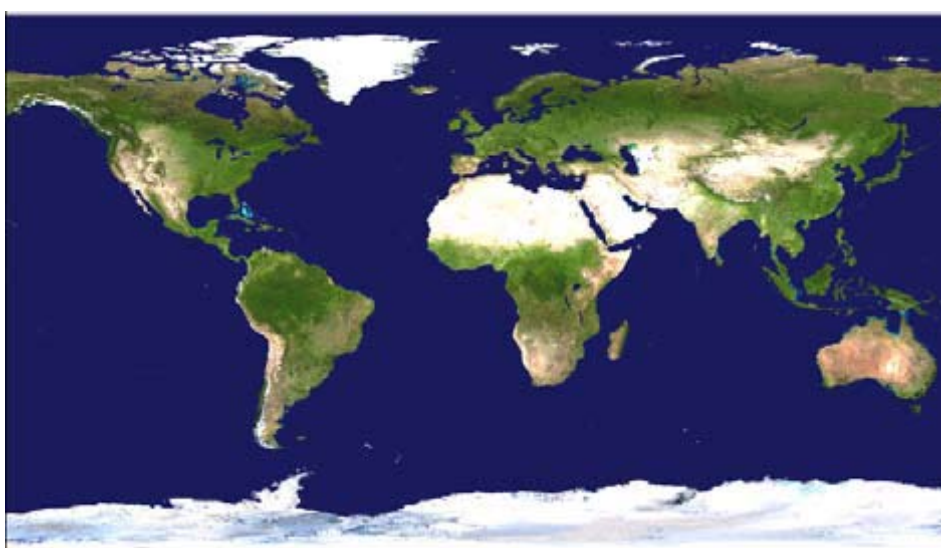


图4.4 JPG格式

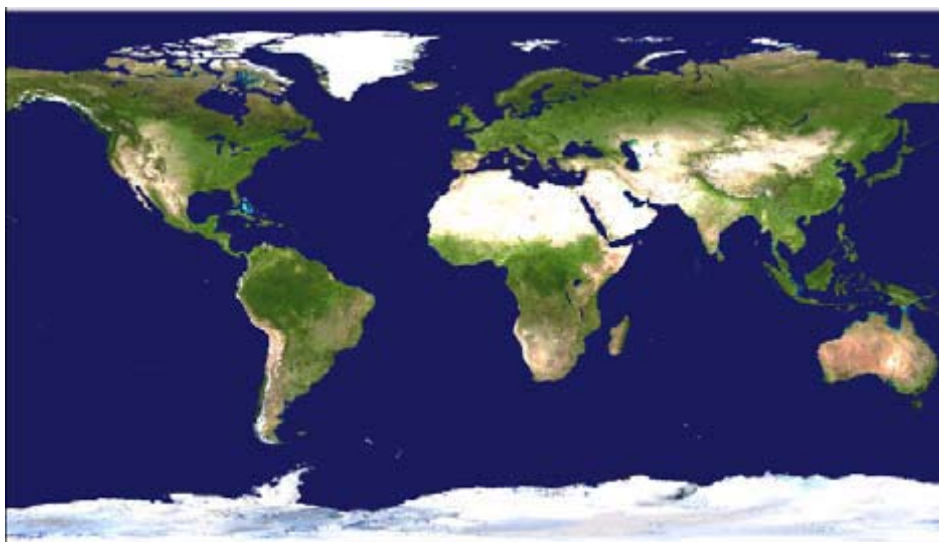


图4.5 BMP格式

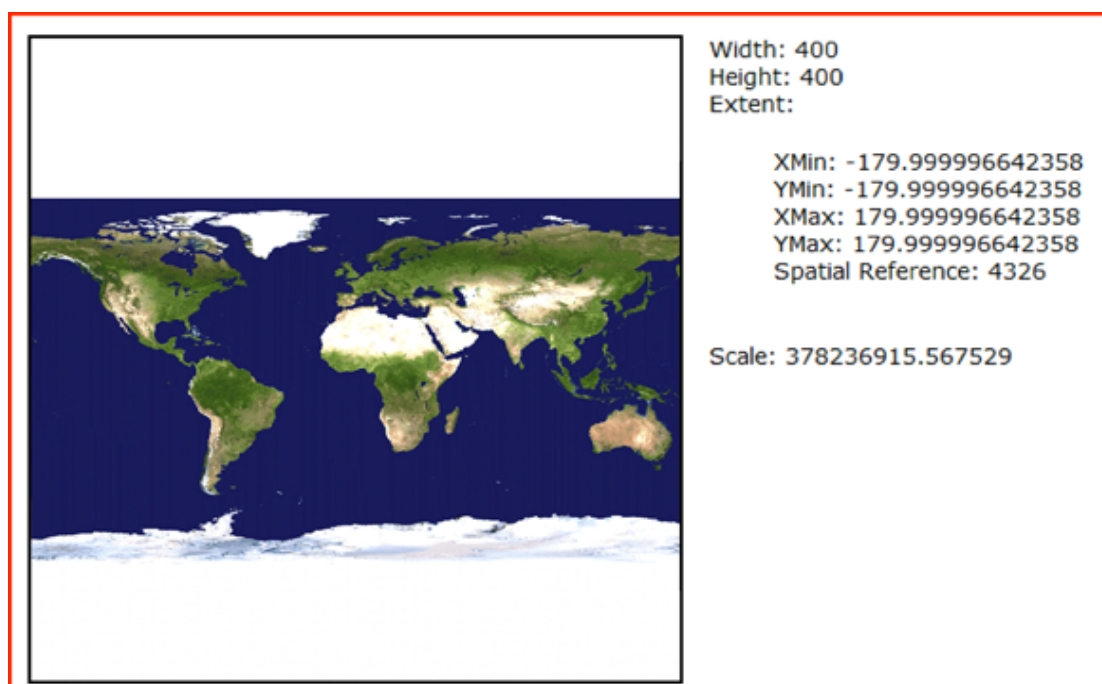


图 4.6 默认图像大小为400 x 400像素

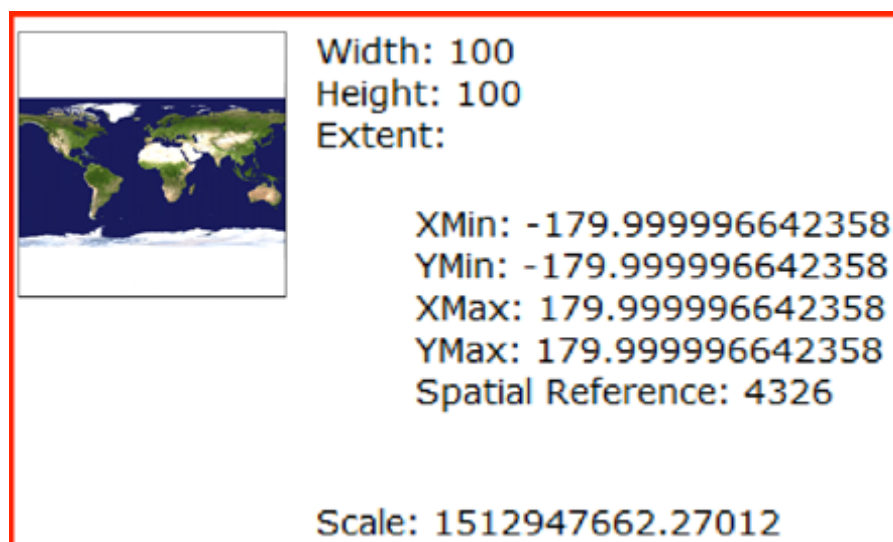


图4.7 图像大小变为100 x 100像素

可以针对栅格地图如影像使用JPG格式；针对矢量为基础的地图，如街道地图使用JPG或PNG；针对覆盖服务使用PNG8。当然，对于报告文件，PDF格式应该是最优选择方案。默认情况下，所有的浏览器都支持JPG、PNG和PDF格式。其他格式，如BMP，以及GIF和SVG虽然不太常用，但也支持。

图像的大小也将发挥重要作用（图4.6）。图像大小不仅影响到ArcGIS Server所需的处理时间，而且还需要带宽以把图像返回到客户端。如果图像更小（图4.7）将加快处理速度和减少带宽的大小，进而提高性能。REST API可让你选择图像的尺寸。默认情况下，图像是按照400×400的像素大小生成。

图像分辨率，每英寸的像素点数（dpi），也是影响REST API性能的一个因素。降低分辨率能使图像生成速度更快，下载到客户端也就使用较少的字节，花费较少的带宽。默认情况下，图像分辨率为96。这个分辨率适合大多数应用的目的，因此是整个互联网中最常用的图像分辨率。

4.2.4 响应格式

REST服务应对每个请求能够返回四种不同的响应格式：HTML格式，JSON，图片或KMZ（压缩版本的KML，或锁孔标记语言）。默认情况下，响应格式是HTML和嵌入图像的HTML。HTML格式是全球最大规模的响应格式，却也是最慢的响应格式，但所有的浏览器，REST服务的通用客户端都能默认显示HTML。虽然HTML是默认格式，但考虑到其他因素时，也可以选择使用其他格式。

响应速度往往也是用户关注的一个主要因素。图像格式的响应（图4.8），和HTML格式一样能够提供图像，但图像格式的响应速度更快，从REST服务到客户端的字节流没有包

括来自其他服务的任何元数据。如果您不需要数据，图像格式是最佳响应格式，因为它的数据流直接发送到客户端。许多用户喜欢JSON格式，该格式很容易解析，并且只占用较小的带宽。但如果需要JSON提供更详细的信息（图4.9），就需要一个额外的步骤来获得图像。

请注意，如果按照HTML和图片格式返回图像，JSON和KMZ返回元数据。客户端必须提出额外的请求来下载图片。

四个响应格式中，图片和JSON具有最好的综合效果。图片提供了最快的响应速度，而JSON不仅返回元数据（图4.9），而且提供链接来下载图片。KMZ也返回一个链接，但使用JSON比KMZ更加简单，节省带宽，这也使它更快地被接收。



图4.8 图像格式响应提供即时图像

```
{"href":"http://server.arcgisonline.com/arcgisoutput/_ags_map4e244e7d57784f968d92a7e227dc05d9.png", "width":400, "height":400, "extent":  
{"xmin":-179.999996642358, "ymin":-179.999996642358,  
"xmax":179.999996642358, "ymax":179.999996642358,  
"spatialReference":{"wkid":4326}}, "scale":378236915.567529}
```

图 4.9 JSON格式响应不包括图像

词汇表

管理控制台：用来管理系统的控制面板。ArcGIS Server提供了一个REST API管理控制台，主要用来管理REST API的缓存刷新策略。REST API使用服务器端缓存来提高性能。利用管理控制台删除或更新服务器上的REST服务，更新缓存。

API：应用编程接口。API是应用程序开发者使用用来开发或定制软件程序的一套接口、方法、协议和工具。通过提供预先写好的程序块，测试，文档代码整合到新的程序，API使编程更为容易。API可被任何编程语言所利用。详见REST API。

ArcCatalog：ESRI公司的应用软件，主要用来定位，浏览和管理空间数据。ArcCatalog也可以发布服务和执行管理任务。ArcCatalog中所有数据管理和服务管理任务都是通过简单易用使用界面和向导步骤进行。利用ArcCatalog，用户可以选择数据集，然后拖放到ArcMap进行查询和分析。

ArcGIS桌面：ESRI公司的软件，提供了丰富的分析框架，包括大量即用型工具进行数据可视化、和集成。利用ArcGIS桌面，可以建立过程模型、脚本和工作流程；执行高级分析模型和自动化业务流程；在专业地图上展示成果。ArcGIS桌面的目的是让用户发现数据中的模式、关系和趋势，这些是不容易在数据库，电子表格，或统计软件包被发现。ArcGIS桌面还可以使用和分发许多种格式的数据。

ArcGIS Explorer：ESRI公司的软件，提供了一个全面的方式来查看和共享各种地理信息，包括图像、地形图、自然要素、渲染地貌、历史地图，街道地图等等。访问全世界的数据；融合本地数据与ArcGIS Server,ArcIMS及其他GIS服务器的数据；利用各种任务进行地理信息系统的分析，包括可见性，建模，缓冲区，和邻近搜寻分析等。

ArcGIS Server 9.3：ESRI公司配备给终端用户的地理信息系统旗舰产品，提供空间数据管理、化和空间分析应用和服务功能。

ArcGIS Server管理器：ESRI公司发布的基于Web的地理信息系统服务。可以执行服务的管理任务以及地理信息系统服务器本身的管理。它可以生成高级的Web制图应用。

架构风格：应用程序或软件包的核心设计方式，软件或硬件组成功能单元的方式。

ArcMap：基础的ArcGIS应用程序，用来展示、查询、编辑、创建和分析数据。提供数据可视化、查询的工具，并创建专业的地图。ArcMap还具有分析、图表、报告功能，以及一套完整的创建和编辑地理数据的编辑工具。

ArcSDE：ESRI公司的软件扩展，用来整合空间数据库引擎（SDE）客户机/服务器技术的ArcInfo软件。用户可以通过定义层来访问ArcSDE。ArcSDE由两个单独的服务器组成——面向Coverage的SDE和面向DBMS的SDE。面向Coverage的SDE以数据文件为基础的数据来源作为SDE层。这些文件格式包括coverages, ArcView GIS的shapefiles, ArcStorm层和地图库层。面向DBMS的SDE将SDE层存储到一个关系数据库中。

ArcToolbox: ESRI提供了大量的通用地理处理工具集合。数以百计工具按照方便的树状视图组织。这些工具可以执行地理处理操作，如投影转换（改变输入数据投影格式），叠加处理、缓冲区分析和空间统计。

AJAX: JavaScript和XML异步。用于创建交互式Web应用程序的一组相互关联的Web开发技术。其主要特点是通过交换少量服务器不公开的数据来增加Web的反应能力和交互性，使整个网页从服务器提取数据时不用重复下载。AJAX技术通常使用REST风格的Web服务。

ASP.NET: 微软.NET框架下的一套技术，用于建立Web应用和基于XML的Web服务。

异步: 非同步。也就是说，不会一起或在同一时间发生。异步调用是客户端应用程序独立于任何时间机制的调用。异步通常意味着客户端（例如JavaScript或Flex程序）发送请求到服务器（例如，一个REST Web服务），但客户端并不停止等待服务器的响应。一定时间间隔后，客户端可以继续执行工作，当回调函数启动后，客户端接收服务器的响应。

BMP: 位图。一种图像文件格式，其中一个或多个位（计算机的最小信息单位）代表屏幕上的一个像素（像元）。每个像素的位数决定位图可以表达的色块。

可书签性 (bookmarkable): 浏览器能够被“书签”，或为后续用户标示参考。

缓冲区: 按照一定距离或时间单位覆盖地图要素的区。缓冲区有助于进行近邻分析。

缓存: 存储信息到计算机目录的过程，无论是地图金字塔、查询结果、或地理处理结果。ArcGIS Server 9.2和9.3允许应用程序将数据缓存到网络服务器上。这样，服务器可以避免重复处理数据。服务器就最大限度地减少通过网络返回请求数据的时间和所需带宽。

缓存控制头: HTTP响应头中的指令，在给定的URL中告诉Web浏览器内容的时间戳和浏览器内容更新的时间长度。因此，浏览器可以决定是否有必要继续下载或简单地使用已经下载的内容。

Call (调用): 从客户端到服务器的请求。

子组件: 层次项目类型的子选项，其中父组件比子组件更接近根目录。目录就是说明父组件和子组件层次关系的一个很好例子。

客户端: 客户机/服务器模型中的一个应用程序、电脑或设备，生成对服务器的请求来获取信息。客户端使用请求的服务，而不需要“知道”其他程序或服务本身的有关任何工作。

客户端系统: 互联网上程序直接运行在客户端电脑上而不是服务器计算机上的系统。

配置文件: 包含机器可读的硬件或软件操作规范的文件，或包含其他文件或者对某一具体用户信息的文件，如用户的登录名。

CORBA技术: 通用对象请求代理体系结构。对象管理组于1992年开发的规范，大量程序（对象）可以同其他程序中的对象进行通讯，即使这两个程序是在不同平台运行和不同语言编写的。CORBA是针对面向对象的工作环境而设计。

CRUD: 创建、读取、更新和删除的首字母缩写，有时是创建、返回、更新和注销的缩写。这被认为是允许查看、搜索、改变信息的一个完整应用程序必备的四个基本功能。

DCOM技术: 分布式组件对象模型。扩展COM来支持网络上不同计算机对象之间的通讯。ArcGIS Server就是建立在COM组件基础上，DCOM用来远程访问这些组件。

DOJO: JavaScript写成的开源动态HTML（DHTML）工具包。DOJO旨在解决DHTML长期存在的问题，适用于大规模动态Web应用开发。

动态地图服务: 根据请求动态生成地图的一种地图服务。缓存地图服务的地图是预先生成的。

节点: 标志着线段或间隔结束的两个点或值。地理概念上，节点就是端点。

范围: 由数据源坐标界定的矩形边界（xmin, ymin, xmax, ymax）。见地图范围。

ETag: 实体标签。Web服务器返回的一个HTTP响应头，用来确定给定网页上内容的变化。

Flex: Adobe Flex是Adobe公司发布用于跨平台开发和部署的一系列技术集合，扩展了基于Flash平台互联网应用。

空间范围: ArcGIS服务的空间范围。

格式: 文档中数据的排列，这样文档就可以被某应用程序读写。文件格式可以是文本文件格式或图像文件格式。

地理编码: 转换街道地址到空间数据，以便在地图上显示地理信息系统操作，通常是从一个街道分段数据层参照到地址信息。

地理数据库: ArcGIS使用的地理数据集。可以包括不同类型的地理数据，如要素类、属性表、栅格数据、网络数据、拓扑结构以及其他数据。

地理数据服务: 一个地理信息系统服务，仅提供没有处理的地理数据。

几何服务: 一个地理信息系统服务，执行复杂的和常用的几何操作，如计算长度和面积，以及投影和简化几何形状。功能包括：坡度分析、最小距离以及最优路径。详见投影几何和单一化几何形状。

地理处理服务: 一个地理信息系统服务，用户能够在服务器上运行地理处理模型。地理处理服务，可以执行多种多样的空间分析操作，如地理要素叠加、覆盖选择和分析，拓扑处理，数据转换等等。

地理浏览器: 可以显示一系列标准地理信息系统网络服务的地图浏览器——例如，ArcGIS Explorer和Google Earth。

get: 客户端通过HTTP发送请求到服务器的一种方法。本质上，所有参数都在请求的网址内。

GIF: 图形交换格式。GIF支持静止图像和简单动画的存储。

Globe服务: 地理信息系统服务的一种类型，提供位置或覆盖区的三维表达。

底面叠加: 谷歌地球的一种影像，导入并覆盖到地形上代替默认的影像。

HTML: 超文本标记语言。用于创建发布在因特网上网页的一种标记语言。HTML是一个系统标签，在一个文件中定义了文本、图形、声音和视频的功能，现在是万维网联盟推出的一个互联网标准。

HTTP: 超文本传输协议。该协议由万维网联合会维护，旨在管理互联网上服务器和客户端交换HTML文件的通讯。

图像: 场景的表达或者描述，通常由一个摄像机或扫描辐射计获得。常见的例子包括遥感数据（例如，卫星数据）、扫描数据和照片。图像存储为二进制代码或整数值的一个栅格数据，这些值就代表了电磁波谱中的反射光的强度、热或其他各种数值。

索引: 通过搜索引擎如谷歌，雅虎，或MSN（微软网络）能够被“检索”，或编入索引的能力。通过搜索引擎可以非常容易地发现经过索引的REST服务目录。

Java: 类似于C++的一种面向对象的编程语言。由于其自动管理内存，Java比C++更小，更方便，受人欢迎且易于使用。Java设计考虑了十分强大的安全和程序中立性，也就是说，它可以运行在任何平台上。

JavaScript: 与Java相关不大的一种脚本语言。虽然与Java 性能有很大的差距，但JavaScript代码通常更简单容易。JavaScript通常使用REST API，但在ArcGIS Server 9.3中，实际是JavaScript API在调用REST API。

JPEG: 联合图像专家组。1、常用在互联网上的失真图像压缩格式。失真压缩提供了高压缩比（例如10:1到100:1），但不保留所有的信息数据。在地理信息系统中，有损压缩是用来压缩栅格数据作为背景图片，但不适合用于数据分析或产生其他数据产品的图像。2、图形存储为JPEG格式的文件。

JPG: 压缩的JPEG文件格式。

JSAPI: JavaScript应用程序接口的缩写。新的ArcGIS JavaScript API是整合ArcGIS Server和ArcGIS Online及其他网络地图平台如Google Maps或Microsoft Virtual Earth地图服务的简单方法。ArcGIS JavaScript API降低了学习难度和分布式可扩展网络地理信息系统的开发时间。

JSON: JavaScript对象注释。轻量级的电脑数据交换格式，JSON是一个易于阅读，基于文本格式来表达简单的数据结构和关联数组（对象）。它是REST返回客户端的两个主要格式，另外一个就是HTML。JSON不需要解析为对象结构就可用于JavaScript。

KML: Keyhole 标记语言。一种基于XML的文件格式，用来在谷歌地球上显示三维空间数据。KML是已被批准的OGC标准。

KMZ: 压缩的KML文件。当一个KMZ文件（.kmz）解压缩后，单个doc.kml文件通常包含重叠的图像和KML引用的图标。

层: 1、电子地图中地理数据的视觉表达。从理论上说，层是地理实体在某一特定地区的一个切片，大致相当于地图上的一个图例。例如，在一个道路图上，道路、国家公园、行政区划界线和河流就可能被视为不同的层。2、ArcGIS软件中，特指一个数据来源，如shapefile,coverage,geodatabase feature class或raster，定义了数据如何在地图上表达。层可以存储在地图文件（.mxd）或单独保存为层文件（.lyr）。

地图范围: 地图上地理区域的边界，通常是一个矩形。动态地图中，通过缩放和漫游，地图范围可以改变。当选择高速缓存的比例尺水平时，如果需要详细的地图，那么就要求更多的金字塔来覆盖地图范围，缓存生成需要的时间就越长。如果每次把缓存比例尺放大两倍，则金字塔需要覆盖4倍的地图范围。例如，一个1:500的正方形地图的金字塔是1:1000地图的4倍，1:250地图的金字塔是1:1000地图金字塔的16倍之多。

地图服务器: 创建和管理地图服务的服务器。

地图服务: 一种生成地图的Web服务类型。

Mashup: 将多个Web数据源数据整合成一个单一的综合应用程序。Mashup可以通过多种方式和格式实现。利用JavaScript创建Mashup可能是最简单的一种方法。Mashup在地理信息系统产业中已经越来越流行，ArcGIS Server 9.3提供了一个JavaScript API来推进地理空间产业的Mashup潮流。

元数据: 描述数据的信息，即数据内容、质量、状况、来源和数据的其他特征或信息。空间数据元数据可以描述和记录主题事件；如何、何时、何地、谁来收集数据；数据可用性和分布信息、数据投影、比例尺、分辨率、精度和基于一些标准的可靠性。

模型: 1、对现实的抽象，用来表达对象、过程或事件。2、表现一种现象或预测结果的一套规则和程序。3、现实的数据表达，如矢量数据模型。

ModelBuilder: ArcGIS Desktop的图形建模工具。它允许用户不使用编程就可以设计和执行地理处理模型。模型可以在ArcGIS Desktop中运行，也可以作为地理处理服务。

镶嵌图: 镶嵌组成。镶嵌图是两个或两个以上的栅格数据集合并组成的，例如，通过合并一些单个图片或相邻地区照片创建一个图像。镶嵌图还可以是邻近地区相同的空间范围和比例尺，且边界已经匹配和解散的地图。

鼠标事件: 发生在软件组成部分的鼠标行为。该事件通常会触发某些事件处理程序的功能。

默认空间参考: 地理信息系统服务的默认空间参考。

.NET: 构建、托管、部署和使用网络服务的一套微软技术集。

网络: 点和线的相互关联，表达从一个位置到另一个位置的可能路线。地理网络由边缘要素，连接要素，以及它们之间的相互连接组成。例如，城市的街道层就是线相互关联的一种网络

类型。

网络分析服务：解决网络问题的服务，例如通行能力、流速或者使用网络连接的容量。

面向对象编程（OOP）：一种编程范式，其中程序被看作是一个离散对象的集合，这些离散对象包括与其他对象交互的子程序和数据结构。面向对象的设计是一种模块化的方法，来创建一个软件产品或计算机系统，其中模块（对象）可以很容易地进行调整，以满足新的需求。

解析：将一个序列的字母和数字部分分解，尤其是测试他们是否符合一套语法规则。

PDF格式：便携式文件格式。电子文本文件使用的格式。

Perl：实用报表提取语言。基于C和一些UNIX工具的一种解释语言。Perl具有强大的字符串处理功能，可以从文本文件中提取信息。Perl可以组装字符串并将其发送到shell（软件，通常是一个单独的程序，提供用户和操作系统的直接通讯）作为一个命令。Perl常常被用于系统管理任务。见URL字符串。

PHP：超文本预处理器，**Hypertext preprocessor**的标准缩写。该开源脚本语言使用HTML文件来执行服务器端的交互功能。PHP可以运行在所有主要的操作系统上，主要用于在Linux和UNIX Web服务器或有附加软件的Windows服务器上。PHP初始版本主要面向个人主页，而更高版本主要面向PHP超文本预处理，或简单PHP。PHP的语法非常简单，与Perl非常类似。它也可以被视为一种技术。

像素：图片元素的缩写。这是图片或栅格地图的最小数据单位，通常是正方形或长方形。像素通常与单元是同义词。

PJSON：简洁版JSON。添加到编码中的一个标记，使JSON格式的响应更易于阅读。

地标：位置的标志，可以标在地图和一个文件夹，或在谷歌地球上。

PNG：便携式网络图形。最常见的一种图像格式。

Polyline：由一个或多个路径定义的ArcGIS形状，其中一条路径就是一系列的连接片段。如果polyline有一个以上的路径（多折线），路径可分支或间断。Polyline要素是数字地图要素，可以表达一个具有长度但没有面积的位置。

Pooled（池化）：服务的属性可以池化或者非池化（Unpooled）。池化的服务可以在多个应用程序之间会话中共享。当应用程序会话返回一个池化服务实例到服务器，就可以被其他应用程序的会话使用。因此，池化服务只适用于无状态操作。

简洁标记（Pretty flag）：编程的格式代码，使其打印时更易于阅读和理解。在ArcGIS Server REST API URL中，如果标记f=pjson，则结果生成简洁JSON。

投影几何：变换地球经度和纬度的刻度线到平面系统的数学方法。不同的投影有不同的坐标系。地理或三维空间及投影系统，或二维系统之间的连接，就是通过地图投影完成的。

协议：一种编程标准或公约，制定了计算终端设备之间的连接，通信和数据传输规则。

查询：从数据库中选择要素和属性表的请求。

Python：一种面向对象的高级编程解释语言，可在多种平台上运行。

栅格：一种空间数据模型，将空间定义为大小相等的单元按行和列排列的数组，组成一个或多个波段。每个单元包含一个属性值和位置坐标。栅格数据模型将现实世界表面划分为规则的网格。栅格模型有利于存储不断变化的数据，例如航空照片，卫星图像，表面化学物质浓度，或地形高程表面。

表达：说明数据的一种方法，以便数据可以被观察和理解。在制图学中，表达是通过描述符号化或者关联真实要素的方式来描述真实世界要素的相似性。表达用来提供信息的格式是可见的，可储存和可传输的，最重要的是能在地图上表达。

资源：本质上讲，资源就是地理信息系统服务。资源，可以返回很多格式的数据，包括字符串，地图和驻留对象（即在两个进程之间保留的数据）。每个服务就是一种具有独特网址的资源。

REST：表述性状态转移。Roy Fielding在其2000年的论文中提出的概念。REST是一个直观的架构风格，通过网址提交到Web服务的请求来创建地图图像。REST允许程序在不同计算机上独立于操作系统或平台进行通讯，通过发送一个HTTP请求到网址，并获取某种格式的返回数据，支持的格式包括XML或嵌入网址的XML。REST中，数据在调用之后仍保持原始状态。REST认为，网络已经拥有Web服务所必需的一切，不需要增加额外的规范协议，例如SOAP。利用REST，任何资源都可以作为URI提供（即代表）和可以使用在HTTP中定义的一个简单操作操纵，“get”读，“put”创建，“post”更新，“delete”删除。从本质上说，REST中所有的请求就是一个简单的网址。

REST API：REST应用程序编程接口。REST API的目的是让人一目了然，可以很容易地应用在多种编程语言中，使用通过浏览器与服务进行通讯，例如.NET, Java, JavaScript, Ruby, Python, Perl等等。REST API利用网址可以访问任何资源。

REST风格：一个描述按照REST架构风格的系统和网络服务的形容词。也就是说，使用服务仅仅需要来自客户端的一个网址。

反地理编码：从地图上的点发现街道地址的过程。

根级别：服务目录的起始文件夹。

可扩展性：指系统随着其规模和复杂性的增加而没有显示出更大的负面影响。

面向服务的架构（SOA）：创建和使用松散耦合，分布式服务的计算机系统架构风格，通过公认的标准进行通讯和交互。

服务目录：HTML格式的ArcGIS Server REST API列表，用户可以浏览服务器内容和获取用来开发应用程序的信息。

服务浏览器：ArcGIS Server 9.3中引入的软件，利用该软件用户可以看到服务器的服务目录。你也可以浏览服务器内容，查看服务器可用的Web服务。使用服务浏览器也可以浏览服务的元数据。

会话状态：一种服务器端技术，其中一个Web应用程序维护同一个客户端和同一个Web应用程序的一系列请求信息。会话是一个程序运行的周期。在多数互动程序中，会话代表程序接收数据输入和处理信息的时间。

Silverlight：一个跨浏览器，跨平台和跨设备的微软插件，用来将.NET为基础的应用程序传输到Web。

单一化几何形状：清除线条或多边形中不必要的细节而使之更普遍地被利用，进行浏览或分析。

SOAP：最初只是简单对象访问协议的缩写，但目前该词的这个含义已经被弃用。SOAP是分布式环境中基于XML语言的协议，用来进行信息交换。通过使用万维网HTTP和XML的通讯机制，SOAP协议允许不同计算机上独立于系统和平台的程序进行交互。SOAP协议主要面向网络服务，现在是一个W3C规范。

SOAP协议栈：功能的一种分层，处理收集和输送系列化的XML编码数据。

空间参考：用于存储空间数据集的坐标系统。

SQL：结构化查询语言。在关系数据库中检索和操纵数据的语法。SQL已经成为大多数关系数据库管理系统的一个行业标准查询语言。

状态：程序中对象包括当前数据。

无状态：程序中不保留调用之间的交换。无状态对象或应用程序被调用后不存储参数或值，始终是在其原始状态。

图像流：图像被实时传输，而不是首先被储存为一个本地文件。图像流允许大型多媒体文件在下载至客户端计算机之前被浏览。当客户端（本地计算机）收到图像后，数据不会被压缩，并使用软件显示，旨在迅速地解释和显示数据。

样式表：提供风格和布局信息的一个文件或表格，如一个XML或HTML文件的页白，字体，对齐和标记内容。样式表通常使用简化的XML和HTML文件设计，这样一个样式表可以适用于很多文件。转换样式表也能包含代码来转换XML文件结构，并将其内容写入到另一个文件。

SVG技术：可扩展矢量图形。一种基于XML的2D图形描述语言。当打印出来或者在不同显示器上显示时，SVG的图像保持其外观不变。

同步：编程时一系列行动或事件的发生必须按照指定的顺序进行。例如，一个程序启动另外的程序，并等待程序完成后才开始运行。

TCP/IP协议：传输控制协议/Internet协议。TCP/IP协议是最常见的互联网协议。TCP是一个在IP层次以上的通信协议，这是一套非专有通信协议或规则，允许电脑在网络上传送及接收数据。

金字塔：空间数据的内在子集（例如，一个栅格）转变到便于管理的矩形集，或像素的行列数。通常用来处理或分析大量消耗计算机内存的海量光栅数据。

令牌：类似短消息或小型包的一个特殊信号（一个传输单位代表数据和包含识别号码、来源和目的地以及误差控制数据的头），用来管理局域网络中的消息传递。

工具：行为执行之前需要与图形用户界面（GUI）或工具栏交互的命令。例如，放大或漫游工具要求用户使用鼠标点击或绘制一个四边形到数字地图上，这样地图才能绘制到一个更大比例尺上或移动地图的中心。

合并缓冲：（V）实现两个或两个以上的多边形空间数据的拓扑叠加，确保所有要素都在输入数据的空间范围内。也就是说，两个数据集的所有要素都被保留并形成一个新的多边形数据。

URI：统一资源标识符。联系特定对象的一种独特的字符串或数值。

网址（URL）：统一资源定位器。网站地址的标准格式。一个网址格式可能如下格式：
<http://www.un.int/ireland>, www.esri.com。第一部分表明使用什么协议（如http或FTP），而第二部分指定IP地址或主机名称（包括域名）。来表明其在网络上的位置。第三部分可有可无，用来指明特定路径的文档或资源（<http://www.esri.com/products.html>）。

URL字符串：字母和数字的序列片段或数字字母的混合片段，代表网址的任何一部分。

VE：编程中使用的微软虚拟地球缩写。

矢量：基于坐标的一种数据模型，可以表达地理要素的点，线和多边形。每个点具有一对坐标，而线和多边形要素则是通过一系列的顺序顶点来表达。每个矢量要素都有属性，而栅格数据模型是将属性赋值到网格单元上。

Web莫卡托：基于莫卡托投影的一种常用地图投影。

Web服务：可在万维网上被其他应用程序使用的软件组件。Web服务不依赖于任何特定的操作系统或编程语言，可被众多类型的应用程序访问。

Web服务描述语言（WSDL）：描述基于SOAP的Web服务的方法，类型和连接点的XML格式。

向导：交互式的用户界面，可以按部就班地帮助用户完成某项任务。向导往往被设计为一系列的对话框，用户填写必要的详细信息就可以完成特定任务。向导通常用于简化漫长的、困难的和复杂的任务。

万维网联盟（W3C）：开发标准的组织，旨在促进万维网和Web技术的互操作性，例如浏

览器，编程语言和设备等。其成员来自世界各地，该组织主导开发了如XML，SOAP，HTML和其他许多基于Web的协议标准。

XML：可扩展标记语言。W3C开发的文本格式的标准通用标记语言，旨在推动计算机应用程序之间的数据交换。XML是建立标准信息格式一套规则，使用定制标签来共享不同程序间的数据和格式。

附录 A: REST 资源

通过ESRI公司的网上资源中心可以获取有关在ArcGIS Server 9.3中应用REST API的最新帮助。ArcGIS Server REST API资源中心的网址为：

<http://resources.esri.com/help/9.3/arcgisserver/apis/rest/default.htm>。

资源中心的帮助内容包括REST概述，入门和服务浏览器的有关知识，资源和操作，输出格式，版本以及管理。资源中心介绍了每项资源的信息及用法，并提供编码样本下载。

ESRI资源中心还提供利用REST 服务的JavaScript API帮助信息，网址为：

<http://resources.esri.com/arcgisserver/index.cfm?fa=jsapis>。

同样在如下网址：<http://resources.esri.com/arcgisserver/apis/flex>，可以获取利用REST服务的Flex API的帮助。