

ArcGIS Server开发



从入门到精通

● 何正国 杜娟 编著

源于实践 成就行录

ArcGIS Server

上海软件行业协会 秘书长 杨根兴

江苏省软件行业协会 副会长 徐雷

鼎力推荐

- ▶ 完整的综合案例帮助读者进行项目开发
- ▶ **14**个实例贯穿ArcGIS Server开发的方方面面
- ▶ **250**分钟的视频讲解和全部源程序（见光盘）
- ▶ 疑难解答和实战技巧帮助读者提高工作效率

 人民邮电出版社
POSTS & TELECOM PRESS



开发快速精通最佳流程



何正国

毕业于武汉大学遥感信息工程学院，系统分析师。基于J2EE的Web GIS平台软件“GeoSurf”获测绘科技进步一等奖，发表论文多篇，成功开发多个地理信息系统项目。参与开发的“广州市规划局统一信息交换平台”获GIS特别成就奖。目前主要从事地理信息系统的设计和开发工作。

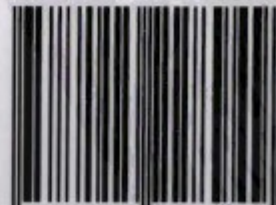
分类建议：计算机 / 地理信息系统 / ArcGIS Server

人民邮电出版社网址：www.ptpress.com.cn

封面设计：任文杰



ISBN 978-7-115-22894-9



9 787115 228949 >

ISBN 978-7-115-22894-9

定价：55.00 元(附光盘)

ArcGIS Server开发 从入门到精通

◎ 何正国 杜娟 编著



人民邮电出版社
北京

图书在版编目 (C I P) 数据

ArcGIS Server开发从入门到精通 / 何正国, 杜娟编
著. — 北京: 人民邮电出版社, 2010. 6
ISBN 978-7-115-22894-9

I. ①A… II. ①何… ②杜… III. ①地理信息系统—
应用软件, ArcGIS—软件开发 IV. ①P208

中国版本图书馆CIP数据核字(2010)第079633号

内 容 提 要

本书以 ArcGIS Server 开发为主线, 全面阐述了 ArcGIS Server 的基础知识及体系结构, 按照 ArcGIS Server 开发的难易程度, 由易到难、由浅入深, 系统地介绍了 ArcGIS Server 的开发, 以及 ArcGIS Server 的其他开发模式。全书共分为 6 篇, 第一篇 ArcGIS Server 基础篇, 包括第 1 章~第 3 章, 主要介绍了 ArcGIS Server 的安装与配置、ArcGIS Server 的体系结构以及 ArcGIS Server 与 ArcObjects 的关系; 第二篇 ArcGIS Server 的开发基础篇, 包括第 4 章~第 6 章, 讲述了准备开发数据、.NET 开发知识和 ArcGIS Server .NET 提供的开发控件等; 第三篇 ArcGIS Server 开发提高篇, 包括第 7 章~第 12 章, 本篇引用大量实例讲解了查询检索、缓冲区、装题图、符号化和空间数据编辑等内容; 第四篇 ArcGIS Server 高级功能开发篇, 包括第 13 章~第 16 章, 主要讲解了性能优化及 ArcGIS Server 的其他开发模式, 如基于 Java 的开发等内容; 第五篇通过讲述一个大型的基于 ArcGIS Server 的开发项目, 把前面所学的知识贯穿起来, 使读者具备开发项目的实力; 最后一篇(第六篇)讲述了 ArcGIS Server 开发中常见疑难问题及其解决办法, 目的是帮助读者找到解决问题的捷径, 掌握设计技巧, 提高开发效率。

本书的内容覆盖面广, 深入浅出, 通俗易懂, 可操作性强, 适用于政府、企业相关部门的 GIS 研究与开发人员, 也适用于高等院校地理学、地理信息系统、房地产、环境科学、资源与城乡规划管理、区域经济学等专业学生参考与学习。本书还适合作为各种 GIS 培训学员的学习教材与参考书。

ArcGIS Server 开发从入门到精通

◆ 编 著 何正国 杜 娟
责任编辑 张 涛

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京铭成印刷有限公司印刷

◆ 开本: 787×1092 1/16

印张: 21.5

字数: 565 千字

印数: 1—3 000 册

2010 年 6 月第 1 版

2010 年 6 月北京第 1 次印刷

ISBN 978-7-115-22894-9

定价: 55.00 元(附光盘)

读者服务热线: (010)67132692 印装质量热线: (010)67129223

反盗版热线: (010)67171154

前言

地理信息系统最早应用在资源环境管理中,随着社会的发展,目前它已经广泛用于资源环境方面,如森林、矿产、水利及农牧业的管理;自然资源方面,如农业、地质矿藏、水资源等的调查;自然灾害方面,如水灾、虫灾、地震灾害的监测、预报、评估;环境保护方面,如水土流失、荒漠化治理;城市建设方面,如城市规划、土地利用规划、房产管理及交通规划等。

GIS 发展趋势

随着 Web Service 技术的发展,基于 Web 的应用在信息系统占据越来越重要的位置,GIS 应用的深入使得它与各应用领域业务的融合越来越紧密,而 SOA(面向服务架构)正好是 GIS 与业务系统的粘合剂。利用 SOA、Web Service 等架构实现地理空间信息访问接口以提供服务发布,更重要的是这种服务还能让客户把其他系统提供的服务聚合起来一起使用,这就是所谓的“数据共享与功能共享”,这种全新的地理信息共享模式必然是未来 GIS 的发展趋势。国际巨头 ESRI 率先吹响进军 SOA 的号角,这标志着整个 GIS 界开始进入 SOA 时代。而 ESRI 基于 SOA 的产品就是 ArcGIS Server。本书正是一本讲述 ArcGIS Server 开发的书,掌握了 ArcGIS Server 的开发技术,就拥有了开发 GIS 大门的钥匙。

本书内容

本书是基于 ESRI 公司的 ArcGIS Server 9.3 开发进行讲解的,通过阅读本书,读者可以掌握 ArcGIS Server 开发的基础知识、常用 GIS 功能的开发、GIS 高级功能的开发、开发.NET 的应用程序及程序发布、ArcGIS Server 性能优化以及常见的错误及解决方法。全书共分为 6 篇,第一篇 ArcGIS Server 基础篇,包括第 1 章~第 3 章,主要介绍了 ArcGIS Server 的安装与配置、ArcGIS Server 的体系结构以及 ArcGIS Server 与 ArcObjects 的关系;第二篇 ArcGIS Server 的开发基础篇,包括第 4 章~第 6 章,讲述了准备开发数据、.NET 开发知识和 ArcGIS Server .NET 提供的开发控件;第三篇 ArcGIS Server 开发提高篇,包括第 7 章~第 12 章,本篇引用大量实例讲解了查询检索、缓冲区、装题图、符号化和空间数据编辑等内容;第四篇 ArcGIS Server 高级功能开发篇,包括第 13 章~第 16 章,主要讲解了性能优化及 ArcGIS Server 的其他开发模式,如基于 Java 的开发等内容;第五篇通过讲述一个大型的基于 ArcGIS Server .NET 的开发项目,把前面所学的知识贯穿起来,使读者具备开发项目的实力;最后一篇(第六篇)讲述了 ArcGIS Server 开发的常见疑难问题及其解决办法,目的是帮助读者找到解决问题的捷径,掌握设计技巧,提高开发效率。

本书的特点

实用是本书的最大特点。本书的作者有 10 余年的 GIS 从业经历,一直从事 GIS 的开发,本书基于作者多年的 GIS 开发经验,对 GIS 开发过程中基本的、常用的功能,如查询、检索、定位、基本操作、专题图、符号化、自定义工具条和命令行等进行详细的讲述,读者在掌握这些基本知识后就可以进行 GIS 项目的开发,本书对高级开发的知识也有涉猎,起到抛砖引玉的作用,

希望能对读者 GIS 开发工作有很好的帮助。

- ArcGIS 资深技术专家执笔。作者深入理解了 ArcGIS Server 内涵、精髓，结合自己丰富的培训经验，并结合大量的一线工程实践经验，仔细研究读者学习规律，潜心编写而成。
- 软件版本采用当前最为流行的 ArcGIS Server 版本。在知识点讲解过程中穿插了新功能的讲述与应用。
- 知识全面、系统，科学安排内容的层次架构；由浅入深，循序渐进，适合读者的学习规律。
- 理论与实践应用紧密结合。基础理论知识穿插在知识点的讲述中，言简意赅、目标明确，其目的是使读者知其然，亦知其所以然，达到学以致用目的。
- 知识点+针对每个知识点的小实例+综合实例的讲述方式，可以使读者快速地学习、掌握 ArcGIS Server 软件操作及应用该知识点解决工程实践中的问题。综合实例部分，深入细致地剖析工程应用的流程、细节、难点以及技巧，可以起到融会贯通的作用。
- 常见问题解答与技巧集萃。针对读者学习过程中容易遇到的问题，作者把实践过程中的经验与技巧进行了总结，力求在最大程度上贴近和满足读者工作实践的需要。

本书附带所有实例操作的视频光盘。

本书由何正国、杜娟主编，参与编写的还有郝旭宁、李建鹏、赵伟茗、刘钦、于志伟、张永岗、周世宾、姚志伟、曹文平、张应迁、张洪才、邱洪钢、张青莲、陆绍强、汪海波。

读者对象

本书适用于政府、企业相关部门的 GIS 研究与开发人员，也适用于高等院校地理学、地理信息系统、房地产、环境科学、资源与城乡规划管理、区域经济学等专业学生参考与学习。本书还适合作为各种 GIS 培训学员的学习教材与参考书。

本书在编写过程中，参考了 <http://bbs.esrichina-bj.cn> 网站的一些资料，在此表示感谢！由于编者水平有限，时间仓促，不足之处，敬请原谅。

联系邮箱为 zhangtao@ptpress.com.cn。

编 者

目 录

第一篇 ArcGIS Server基础

第1章 ArcGIS Server概述----- 2

- 1.1 ArcGIS Server简介 ----- 2
- 1.2 ArcGIS Server架构 ----- 3
- 1.3 ArcGIS Server的功能 ----- 4
- 1.4 ArcGIS Server的安装
与配置 ----- 5
 - 1.4.1 安装准备 ----- 5
 - 1.4.2 安装 ----- 6
 - 1.4.3 安装与配置说明 ----- 8
- 1.5 小结 ----- 9

第2章 ArcGIS与ArcGIS Engine ----- 10

- 2.1 ArcGIS软件体系结构 ----- 10
- 2.2 组件对象模型 ----- 12
- 2.3 ArcObjects简介 ----- 13
 - 2.3.1 ArcObject的组织划分 --- 13
 - 2.3.2 ArcObject的开发 ----- 14
- 2.4 ArcGIS Engine ----- 17
 - 2.4.1 ArcGIS Enigne构成 ----- 17
 - 2.4.2 ArcGIS Engine功能 ----- 19
 - 2.4.3 ArcGIS Engine开发
环境 ----- 19
 - 2.4.4 ArcGIS Engine与
ArcGIS Server ----- 24
 - 2.4.5 ArcGIS Engine如何
调用ArcGIS Server ----- 24
- 2.5 小结 ----- 26

第3章 空间数据管理 ----- 27

- 3.1 空间数据库模型
Geodatabase ----- 27
 - 3.1.1 Geodatabase概念 ----- 27
 - 3.1.2 Geodatabase模型 ----- 28
- 3.2 访问空间数据库 ----- 30

- 3.2.1 打开数据库工作空间 --- 30
- 3.2.2 通过NAME对象方式 --- 31
- 3.2.3 获得工作空间
实际元素 ----- 32

3.3 矢量数据 ----- 33

- 3.3.1 文件数据导入
Geodatabase ----- 34
- 3.3.2 从Geodatabase复制特征
数据集到个人数据库 --- 36
- 3.3.3 编辑Geodatabase中的
数据 ----- 37
- 3.3.4 空间数据拓扑检查 ----- 39

3.4 栅格数据 ----- 41

- 3.4.1 打开栅格工作空间 ----- 41
- 3.4.2 获得栅格数据集 ----- 42
- 3.4.3 获得栅格目录 ----- 43
- 3.4.4 栅格数据上载 ----- 44
- 3.4.5 栅格数据拼接 ----- 46

3.5 小结 ----- 47

第二篇 ArcGIS Server 的开发基础

第4章 ArcGIS Server地图服务发 布（准备开发的数据） -- 50

- 4.1 制作地图文档 ----- 50
 - 4.1.1 获取空间数据 ----- 51
 - 4.1.2 使用ArcMap编辑地图
文档 ----- 51
- 4.2 用户权限设置 ----- 56
- 4.3 在ArcCatalog中发布
Map Service ----- 57
- 4.4 在ArcGIS Server Manager
中发布Map Service ----- 61
- 4.5 在ArcGIS Server Manager
中发布OGC地图服务 ----- 63
- 4.6 小结 ----- 65

第5章 ArcGIS Server开发
基础ASP.NET ----- 66

5.1	ASP.NET简介-----	66
5.1.1	Web开发技术-----	66
5.1.2	ASP.NET特点-----	67
5.1.3	ASP.NET 2.0-----	69
5.1.4	ASP.NET 3.5-----	70
5.1.5	Visual Studio 2008 与ArcGIS Server 9.3 开发环境-----	70
5.2	JavaScript 和Ajax技术---	73
5.2.1	JavaScript本质-----	73
5.2.2	JavaScript基本函数-----	73
5.2.3	理解Ajax-----	74
5.2.4	在客户端回调中 使用Ajax-----	75
5.3	ASP.NET Ajax-----	76
5.3.1	ASP.NET Ajax简介-----	76
5.3.2	服务器回调-----	77
5.3.3	ASP.NET Ajax服务器 控件-----	79
5.3.4	深入客户端库-----	83
5.3.5	控件扩展器-----	85
5.4	ArcGIS Server Web ADF 中的Ajax-----	86
5.4.1	.NET ADF中Ajax的 调用过程-----	86
5.4.2	Web ADF Ajax调用的 示例详解-----	87
5.5	小结-----	92

第6章 ArcGIS Server控件介绍 - 93

6.1	资源管理控件-----	93
6.1.1	MapResourceManager 控件-----	93
6.1.2	GeoprocessingResource Manager控件-----	97
6.1.3	GeocodeResource Manager控件-----	99
6.2	地图显示及其相关 控件-----	100
6.2.1	Map控件-----	100
6.2.2	MapTips控件-----	102
6.2.3	Maginifier控件-----	103

6.2.4	OverviewMap控件-----	104
6.2.5	Toolbar控件-----	105
6.2.6	Toc控件-----	107
6.2.7	ScaleBar控件-----	108
6.2.8	Navigation控件-----	108
6.2.9	ZoomLevel控件-----	109
6.2.10	MapCopyrightText 控件-----	110
6.3	TaskManger与Task控件-	111
6.3.1	TaskManager控件-----	111
6.3.2	EditorTask控件-----	112
6.3.3	SearchAttributesTask 控件-----	115
6.3.4	QueryAttributesTask 控件-----	117
6.3.5	GeoprocessingTask 控件-----	117
6.3.6	FindAddressTask 控件-----	118
6.3.7	FindPlaceTask控件-----	119
6.3.8	PrintTask控件-----	120
6.3.9	TaskResults控件-----	121
6.4	其他控件-----	122
6.4.1	FloatingPanel控件-----	122
6.4.2	ContextMenu控件-----	123
6.4.3	DocExtender控件-----	125
6.4.4	HoverExpandExtender 控件-----	126
6.4.5	ColorPicker控件-----	128
6.5	小结-----	129

第三篇 ArcGIS Server
开发提高

第7章 ArcGIS Server开发
概述 ----- 132

7.1	地图的基本操作-----	132
7.1.1	地图显示-----	133
7.1.2	地图缩放与漫游-----	135
7.2	查询定位-----	137
7.2.1	根据坐标定位-----	137
7.2.2	根据属性值定位-----	139

7.3	缓冲区分析	142
7.4	自定义Tool	147
7.5	自定义Command	151
7.6	ArcGIS Server Task	154
7.6.1	Task的工作流程	154
7.6.2	应用ArcGIS Server Task	155
7.6.3	自定义Task	157
7.7	小结	161

第8章 ArcGIS Server基于模板开发 162

8.1	配置模板开发中资源配置	162
8.2	图形选择	165
8.2.1	单击选择	165
8.2.2	矩形选择	165
8.2.3	折线选择	166
8.2.4	多边形选择	167
8.2.5	画圆选择	168
8.3	控制地图图层的显示	169
8.4	TOC图层移动	171
8.5	动态添加图层	175
8.6	地图导出	182
8.7	小结	184

第9章 ArcGIS Server专题图开发 185

9.1	柱状图	185
9.2	饼状图	188
9.3	分级专题图	190
9.3.1	Graduated colors	191
9.3.2	Graduated sysmbols	193
9.3.3	Dot desity (点密度专题图)	196
9.4	分类专题图	198
9.4.1	根据某个字段的惟一值	198
9.4.2	根据多个字段的值	201
9.5	自定义专题图	202
9.6	小结	208

第10章 ArcGIS Server 符号化 209

10.1	使用符号库中的符号进行符号化	209
10.2	使用TureType进行符号化	213
10.3	自画符号	215
10.3.1	MarkerSymbol符号	215
10.3.2	LineSymbol符号	215
10.3.3	FillSymbol符号	216
10.3.4	使用图片进行符号化	216
10.4	小结	219

第11章 ArcGIS Server数据在线编辑 220

11.1	EditorTask的数据编辑	220
11.1.1	点的编辑	220
11.1.2	线的编辑	221
11.1.3	面的编辑	222
11.2	编辑功能的定制	223
11.3	编辑功能的扩展	227
11.4	编辑属性数据	230
11.5	小结	231

第12章 ArcGIS Server Web 应用程序部署 232

12.1	应用程序部署环境	232
12.2	IIS中部署步骤	232
12.3	部署中的常见问题及解决方案	235
12.4	小结	238

第四篇 ArcGIS Server 高级功能开发

第13章 ArcGIS Server ADF For .NET高级功能 240

13.1	一般服务器对象扩展	240
------	-----------	-----

13.2	Geodata的签入 签出-----	244
13.3	空间查询-----	250
13.4	Geoprocessing缓冲-----	252
13.5	最短路径分析-----	255
13.6	小结-----	261

第14章 ArcGIS Server 9.3 开发模式----- 262

14.1	.NET Web ADF开发----	262
14.2	Java Web ADF 开发-----	264
14.3	SOAP API应用-----	267
14.4	REST API-----	271
14.5	Mobile ADF-----	274
14.6	ArcGIS JavaScript API----	279
14.7	JavaScript Extension for Virtual Earth-----	281
14.8	JavaScript Extension for Google Maps API-----	282
14.9	小结-----	284

第15章 ArcGIS Server For Java----- 285

15.1	ArcGIS Server Java 开发基础-----	285
15.2	ArcGIS Server Java ADF 开发初步——自定义 工具-----	289
15.3	ArcGIS Server Java自定义 Task-----	292
15.4	小结-----	298

第16章 ArcGIS Server性能 优化----- 299

16.1	确定应用系统的瓶颈--	299
16.2	硬件和网络优化-----	302
16.3	数据-----	304
16.3.1	数据量-----	304

16.3.2	数据组织-----	305
16.3.3	数据渲染-----	306
16.4	服务-----	306
16.4.1	服务组织-----	307
16.4.2	服务设置-----	308
16.5	应用系统配置-----	309
16.6	小结-----	309

第五篇 ArcGIS Server 综合案例

第17章 某市土地开发中心地理 信息管理平台----- 312

17.1	需求概述-----	312
17.1.1	明确需求-----	312
17.1.2	设计实现框架-----	315
17.2	数据库设计-----	316
17.3	业务逻辑设计-----	318
17.4	界面设计-----	321
17.5	综合实现-----	322
17.6	小结-----	328

第六篇 常见疑难解答 与设计技巧

第18章 ArcGIS Server常见 问题及其处理----- 330

18.1	防火墙问题-----	330
18.2	ArcCatalog中服务不能 预览-----	331
18.3	安装错误-----	332
18.4	能否使用new关键字 创建对象-----	333
18.5	ArcGIS Server连接 方式-----	334
18.6	Map控件的 ImageBlending Mode 属性-----	334

第一篇

ArcGIS Server

基础

- ▶ 第 1 章 ArcGIS Server 概述
- ▶ 第 2 章 ArcGIS 与 ArcGIS Engine
- ▶ 第 3 章 空间数据管理

这篇主要讲述 ArcGIS Server 的基础知识，包括 ArcGIS Server 的体系结构、ArcGIS Server 能够实现的功能 ArcGIS Server 的核心——ArcObjects 等以及 ArcGIS Server 的数据管理，让读者对 ArcGIS Server 有一个全面的了解。

第 1 章

ArcGIS Server概述

1.1 ArcGIS Server 简介

ArcGIS Server 是一个基于 Web 的企业级 GIS 解决方案，从 ArcGIS 9.0 版本开始 ESRI 产品家族才有 ArcGIS Server。ArcGIS Server 是一套用于开发基于网络的企业级服务器端程序的组件集，服务器端包括 Web Service、Web 应用程序和 EJB 等，它为创建和管理基于服务器的 GIS 应用提供一个高效的平台。它充分利用了 ArcGIS 产品的核心组件库 ArcObjects（简称 AO），并基于工业标准提供 GIS 服务。ArcGIS Server 将两项功能强大的技术——地理信息系统（GIS）和网络技术（Web）结合在一起：GIS 擅长于空间相关的查询、定位、分析和处理，网络技术则提供全球互连，促进信息共享。这两项技术协同合作，构成了 ArcGIS Server 的主旋律。

ArcGIS Server 是一个用于构建集中管理、支持多用户的企业级 GIS 应用的平台软件。ArcGIS Server 产品包括两个部分：一个是 GIS Server，它是一个提供 GIS 服务的服务器端软件产品，包括一系列核心 AO 库和运行这些 AO 组件的环境。另外一个 ADF（Application Developer Framework），即应用程序开发框架，它有 Java 和 .NET 两种组件包，是用来开发和部署基于 GIS Server 的 Web 应用程序的产品，包括组件对象以及与 Web GIS 相关的 Web 控件、通用的 Web 模板和开发帮助；它还有一个 Web 应用程序的 Runtime，专门用于发布和部署使用 ADF 开发的 Web 程序，如 ASP.NET 等。

GIS Server 是一套 GIS 服务器组件，专门用来管理和发布地图服务，安装在 GIS 服务器上。ADF 是供开发人员使用的开发组件集（开发包），安装在开发人员的机器上。Web 应用程序、Web 服务和桌面端程序，都可以使用 ADF。ADF Runtime 是专门用于部署开发人员的 GIS Web 程序和 GIS Web Service 的运行环境，安装在 Web 服务器上。GIS 服务器、Web 服务器和开发人员的计算机可以是同一台机器，也可以是不同的机器。

ArcGIS Server 作为服务器端的 GIS 软件，与传统的桌面端 GIS 软件和基于 B/S 的 WebGIS 软件有许多不同。与以往的 WebGIS 产品相比，它不仅具备发布地图服务的功能，而且还具有在线编辑和强大的分析能力，这对于 WebGIS 的发展是非常有意义的。ArcGIS Server 又是基于 Web 的，它不仅可以为局域网用户提供 GIS 服务，还可以为广大的互联网用户提供 GIS 服务，

功能与桌面 GIS 软件基本相当。由于 ArcGIS Server 基于强大的核心组件库 AO 构建，并且以主流的网络技术作为通信手段，所以它具有许多令人惊喜的优点和特征。表现如下：

- 集中式管理带来成本的降低。企业级的 GIS 系统由于系统体系庞大，用户数量众多，在管理上往往会造成相当大的开销。使用 ArcGIS Server 无论是从数据的维护和管理上还是从应用系统的更新升级上来说，都只需要在服务器端进行集中的处理，而无须在每一个终端用户上进行大量的维护工作，这不但极大地节约了投入的时间成本和人力资源，而且还有利于空间数据的一致性。
- 瘦客户端也可以享受企业级的 GIS 服务。过去高级 GIS 功能一般都是在桌面端 GIS 软件中实现，使用 ArcGIS Server 产品后，在浏览器中也可以实现 GIS 的高级操作——在服务器上部署好用 ArcGIS Server 搭建的 GIS 服务，客户端通过网络浏览器（IE、Netscapes、Firefox 等）调用服务器端的服务即可实现，通常只要求服务器的配置比较高而对客户端的硬件配置没有要求。
- 安全的连接和访问机制。安全性是企业级 GIS 服务器的一个非常重要的指标。采用 ArcGIS Server 构建的企业级 GIS 系统是一个高度安全的系统，它只能被 GIS 服务器管理员授权的用户访问。
- ArcGIS Server 具备了在线数据编辑功能。用户在野外作业时可以通过手持移动设备直接对服务器端的数据库进行维护和更新，大大减少了将测绘资料带回室内后再上传更新的重复工作量，为野外测绘和勘测提供了极大的便利。
- 支持大量的并发用户访问，具有负载均衡能力。ArcGIS Server 采用分布式组件技术，可以将大量的并发用户访问均衡地分配到多台服务器上，可大幅地减少响应时间、提高用户的并发访问数量。
- 支持多种客户端。ArcGIS Server 是基于 AO 组件的一个服务器产品，它可以和许多客户端一起工作。它支持基于 C/S 模式的桌面应用程序、B/S 的浏览器和嵌入式设备。而且在开发 Web 应用时，支持 Java、.Net、JavaScript、Flex、Sliverlight、SOAP、REST 等。这样用户就可以根据需要选用合适的技术来定制自己的 WebGIS 解决方案，使得用户在实现自己的应用上具有很大的灵活性。

1.2 ArcGIS Server 架构

ArcGIS Server 是一个分布式系统，它由几个部分组成，可以分别部署在不同的机器上，它们分别在进程的管理、启动和关闭以及对象运行的服务器的负载均衡方面起着专门的作用。ArcGIS Server 的体系结构，如图 1.1 所示。

客户端表现层包括了基于浏览器/服务器（B/S）结构的浏览器和基于客户端/服务器（C/S）结构的桌面应用程序。该层主要是调用 ArcGIS Server 开发的功能和访问 ArcGIS Server 发布的数据公共接口。Web 浏览器是用户通过浏览器来连接到 Web 层的用户接口，而桌面程序是通过 HTTP 协议连接到运行在 Web 服务器上的 Web Service 或直接通过内部局域网连接到 GIS Server 上来获得相应服务对象的代理。

应用层可分为 Web 应用层和 GIS Server 层。Web 应用层主要负责用户通过 Web 浏览器和 Web Services 发送的请求，并根据用户请求从 GIS Server 中获得相应的结果返回给客户端显示。GIS Server 层是由服务器对象管理器（SOM）管理一个或者多个服务器对象容器（SOC）。

其中 SOM（Server Object Manger）是一个 Windows/UNIX 服务，它管理一组分布在一个或多个 SOC 上的服务器对象（Server Object），而 Server Object 的运行是在 ArcSOC.EXE 进程中，

同时它还对 SOC 机器的负载平衡进行动态调节。实际上用户通过局域网或者互联网连接到 ArcGIS Server 服务器时,用户连接的对象是 SOM, SOM 得到用户的请求后根据负载平衡的原则去自动调用相应的 SOC 机器里的服务器对象。

SOC (Server Object Container) 是 SOM 启动的一个进程,一个或多个服务器对象。服务器对象运行在 SOC 机器上。每一个容器可运行多个容器进程,而每一个容器进程中可有一个或多个服务器对象。容器进程是由 SOM 来控制启动或者停止的。

服务器管理员可以使用 ArcCatalog 管理 GIS Server。可以注册新的服务器机器、添加和删除服务器对象、启动和停止服务器对象。

数据层主要负责为 SOC 提供相应的空间数据。可以通过 ArcSDE 来访问的关系型数据或者是基于文件类型来存储的数据 (Personal Geodatabase、ShapeFile 等)。

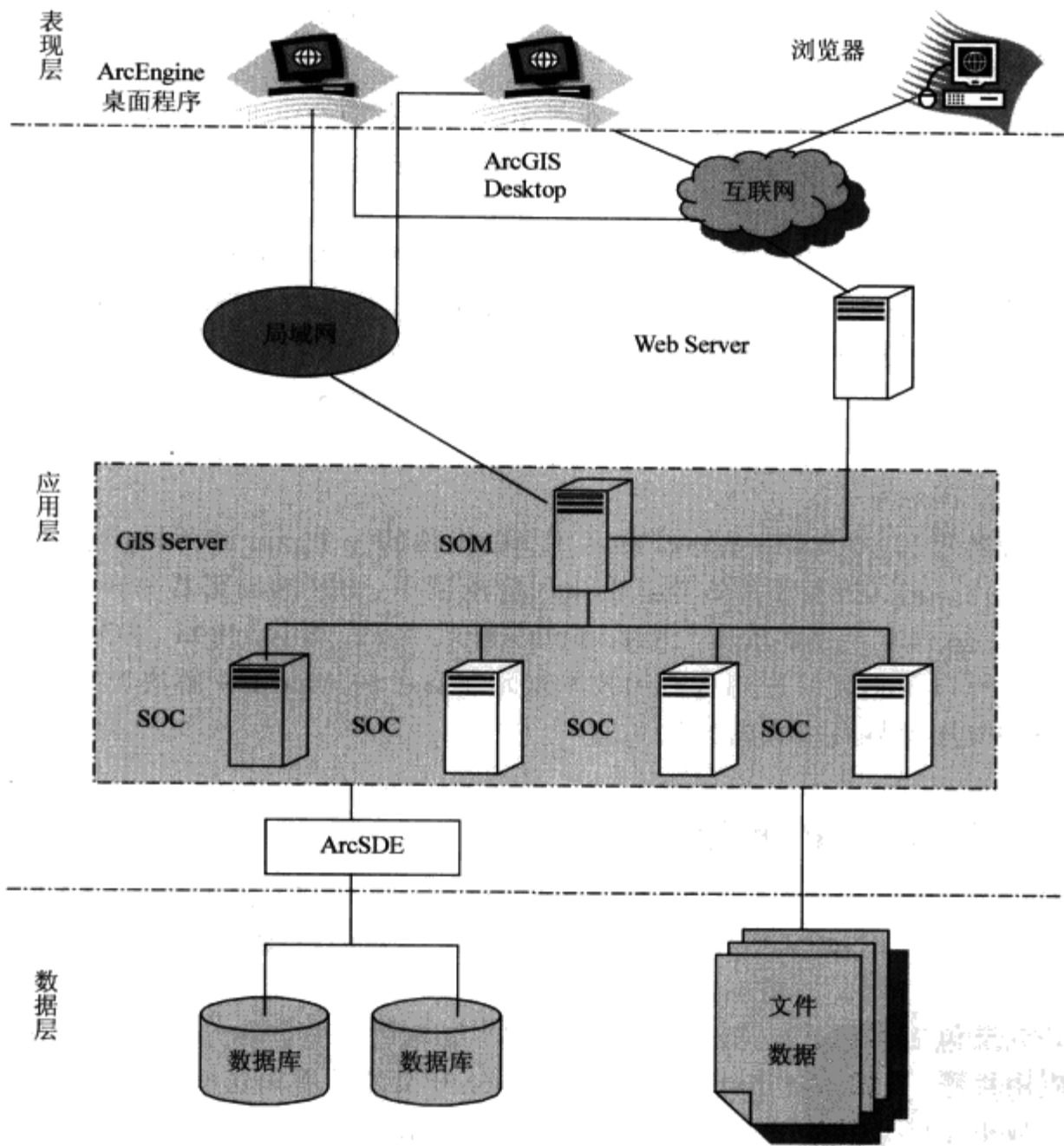


图 1.1 ArcGIS Server 体系结构

1.3 ArcGIS Server 的功能

ArcGIS Server 作为服务器端产品,为 WebGIS 提供的功能能够与传统的桌面端 GIS 相媲美。那么用 ArcGIS Server 开发的 WebGIS 应用具体能够实现哪些功能呢?下面列举通用 WebGIS 的功能:

- 关闭或打开某一图层;

- 放大图层的范围;
- 缩小图层的范围;
- 全图显示;
- 漫游地图;
- 矢量数据和影像数据无缝叠加显示;
- 查询空间要素的属性信息;
- 根据属性值查询空间要素;
- 量测长度或面积;
- 以矩形、圆形、任意多边形、折线、点来选择空间对象;
- 计算缓冲区;
- 下载地图图片;
- 打印地图图片。

除了这些通用的 GIS 功能外, ArcGIS 还提供了以下功能:

- 通过 Internet 编辑在线编辑空间要素, 包括其空间位置和属性信息;
- 动态渲染空间要素;
- 动态管理地图中的图层, 可以添加和删除图层;
- 高级 GIS 分析功能, 如最短路径分析等;
- 提供 GIS 服务, 不仅提供符合工业标准的数据服务还可以提供 GIS 功能;
- 可以动态显示实时空间数据。

1.4 ArcGIS Server 的安装与配置

1.4.1 安装准备

(1) 预备工作。安装 ArcGIS Server 之前一定要了解机器的配置, 建议机器配置如下。

处理器要求: 最小 Pentium 或更高。

内存要求: 最小 512MB, 推荐 1GB 或更高。

硬盘空间要求: NTFS 360MB FAT 597MB。

(2) 系统软件要求。

- 操作系统:

Windows 2000 Professional, Service Packs 1-4 are optional。

Windows XP Professional, Service Pack 1 is optional。

Windows 2000 Server。

Windows Server 2003。

- 浏览器:

Internet Explorer 6.0 或更高版本。

- Microsoft DirectX 要求:

DirectX 可以自动升级或是从微软网站下载。

(3) 附加的系统要求。

- Server Object Container .NET feature。

要求.NET Framework 2.0，否则不支持安装.NET 子集（.NET 应用）。

- Software Developer Kit .NET feature。
要求.NET SDK 2.0，否则不支持.NET 开发。
- 卸载 ArcGIS 9.0 以前的任何版本。
- ArcGIS Authorization file（license file）。

1.4.2 安装

（1）ArcGIS Server 安装启动界面如图 1.2 所示。选择要安装的类型，本例选择 for the Microsoft .NET Framework。单击“Next”按钮进行安装，完成安装后的界面如图 1.3 所示。

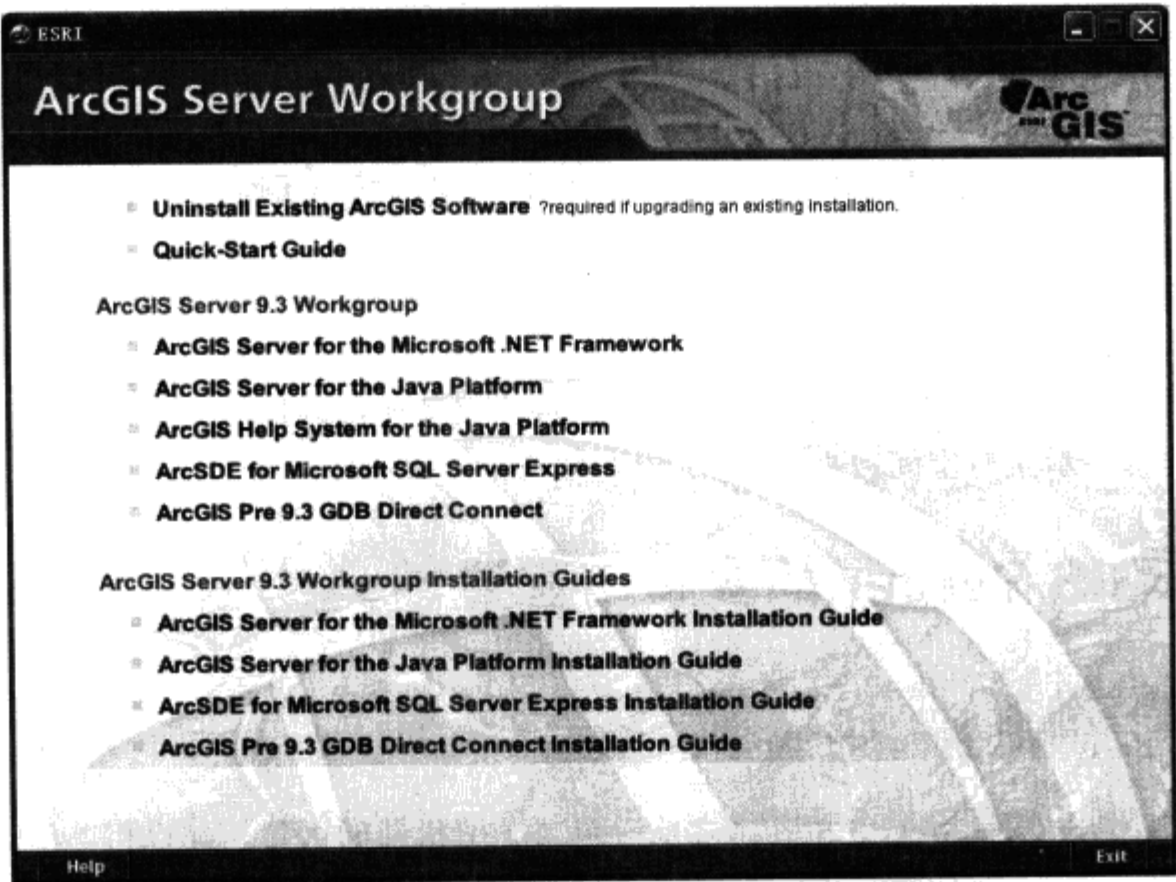


图 1.2 安装启动界面

（2）安装完成后，按要求一步步进行 ArcGIS Server 相关配置，如图 1.4～图 1.11 所示。

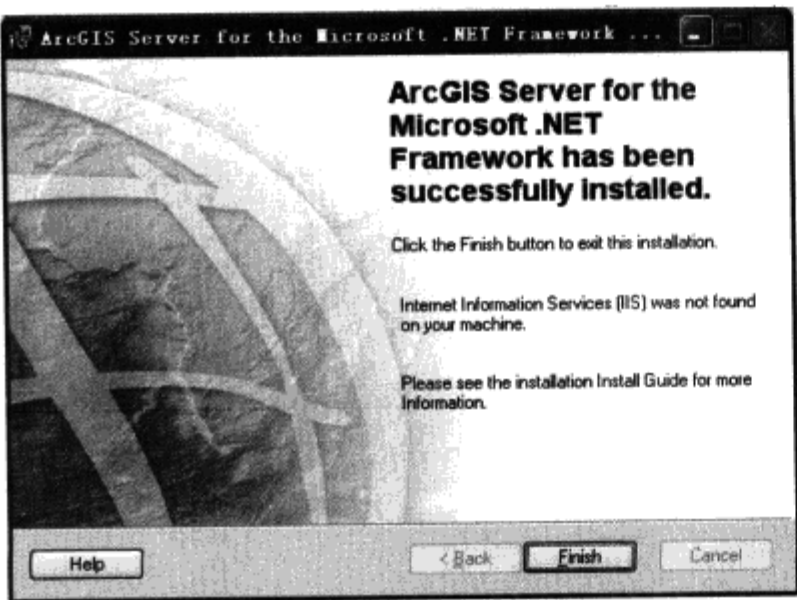


图 1.3 安装结束

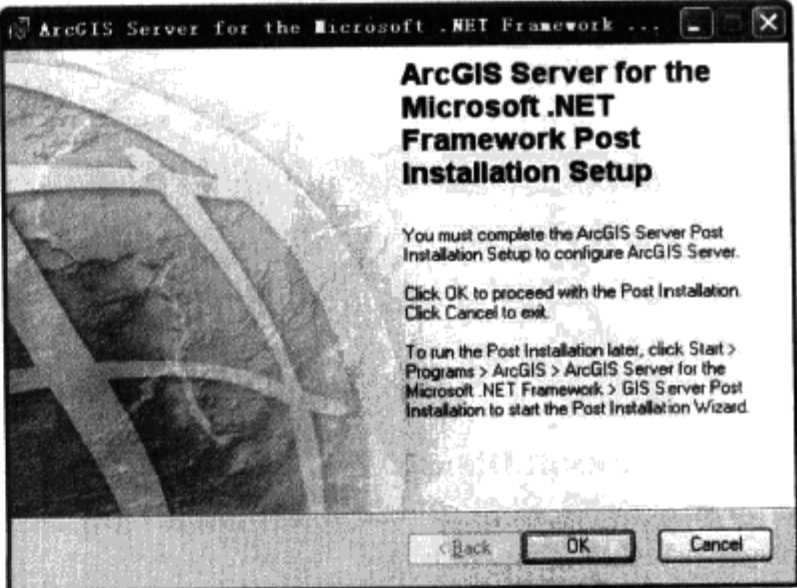


图 1.4 安装配置

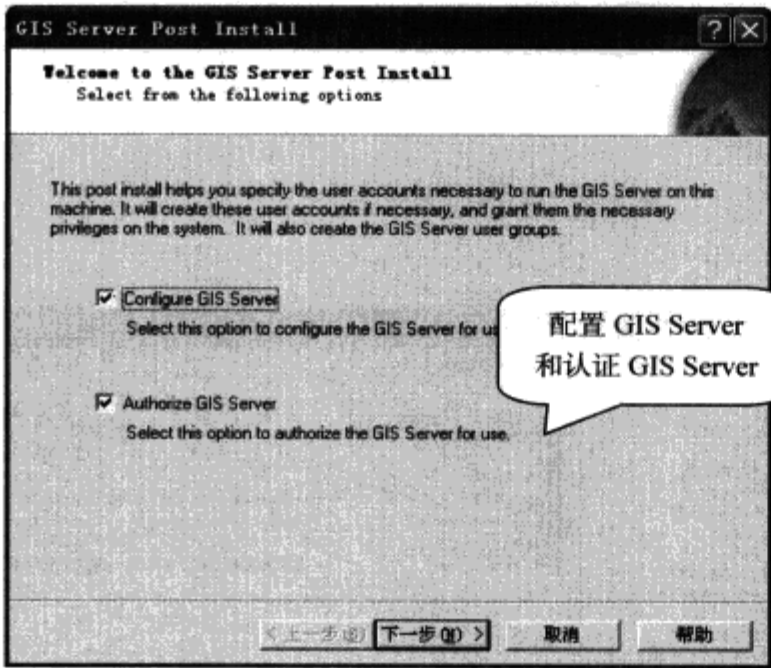


图 1.5 配置和认证 GIS Server

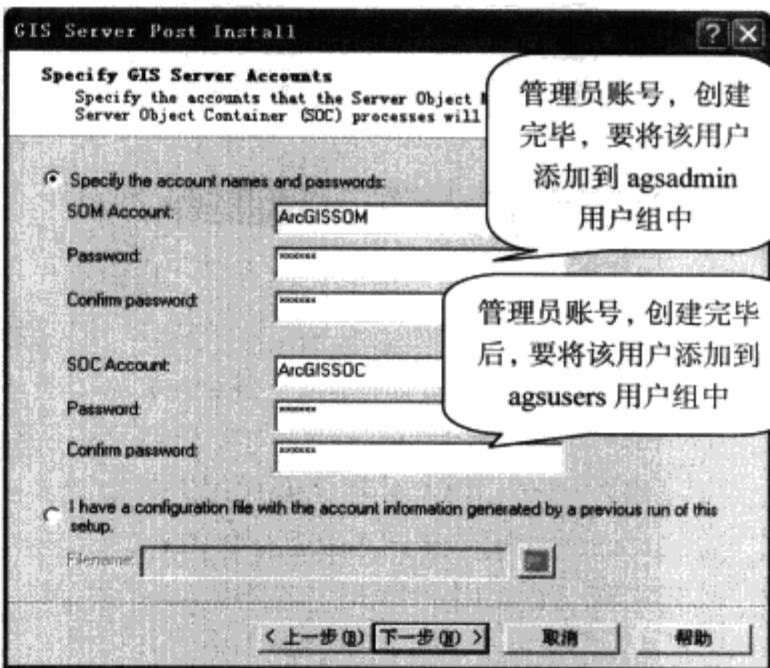


图 1.6 添加 SOM、SOC 账号

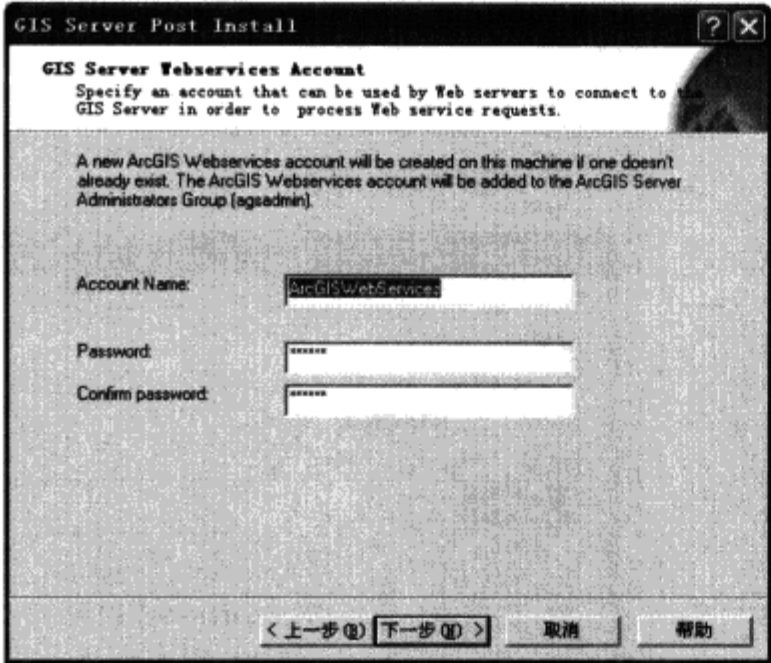


图 1.7 添加 ArcGIS Web Services 账号

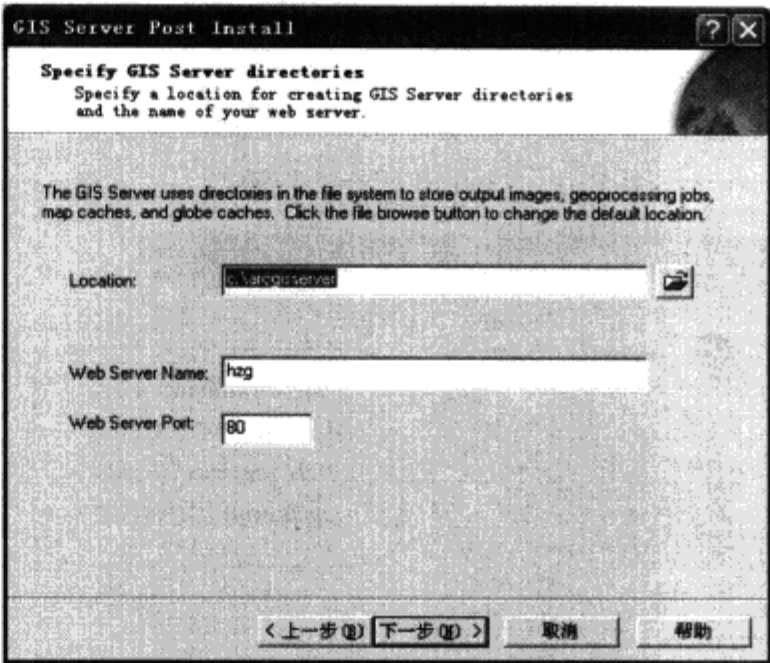


图 1.8 配置路径、服务器及端口号

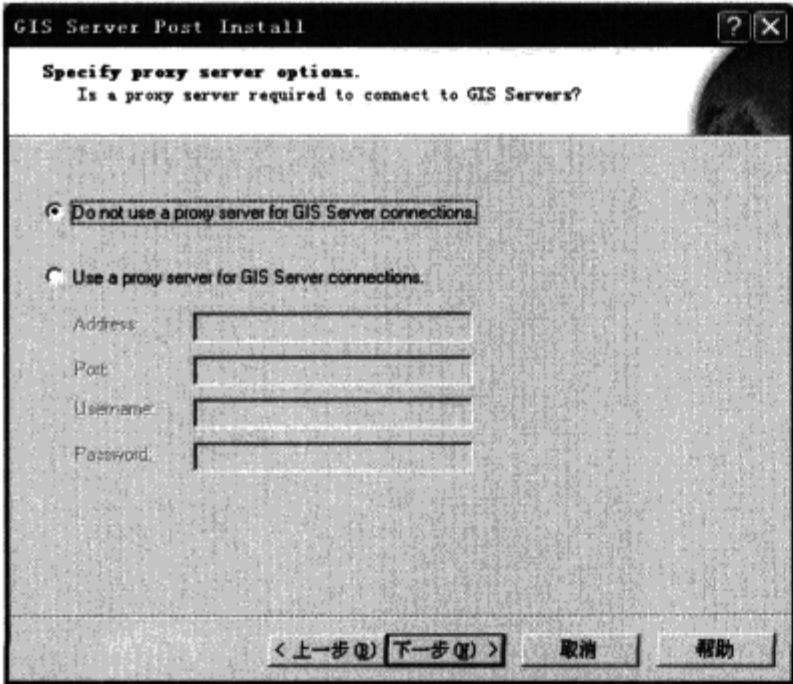


图 1.9 配置是否使用代理服务器

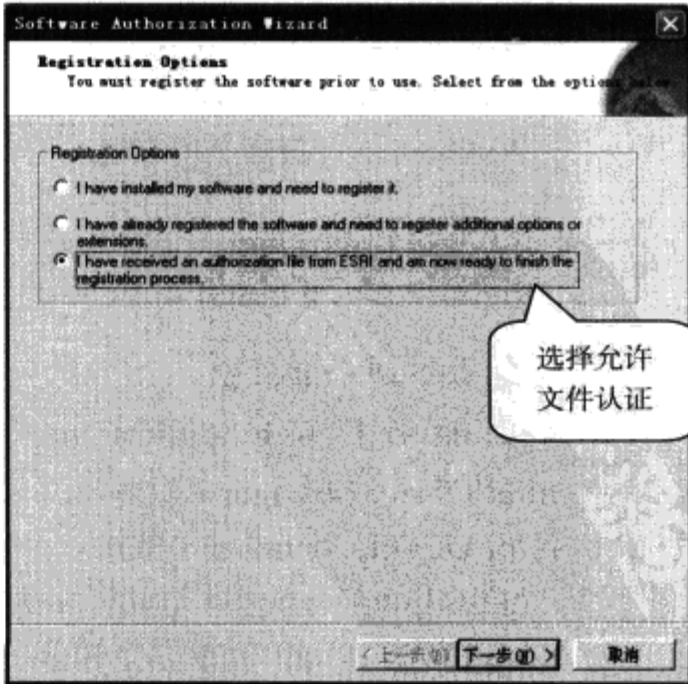


图 1.10 注册许可权限项

(3) 单击“完成”按钮完成配置。选择“添加→高级→快速查找”按钮将创建的用户 (ArcGISSOM、ArcGISSOC、ArcGIS Web Services) 添加到相关的用户组中，如图 1.12~图 1.14 所示。

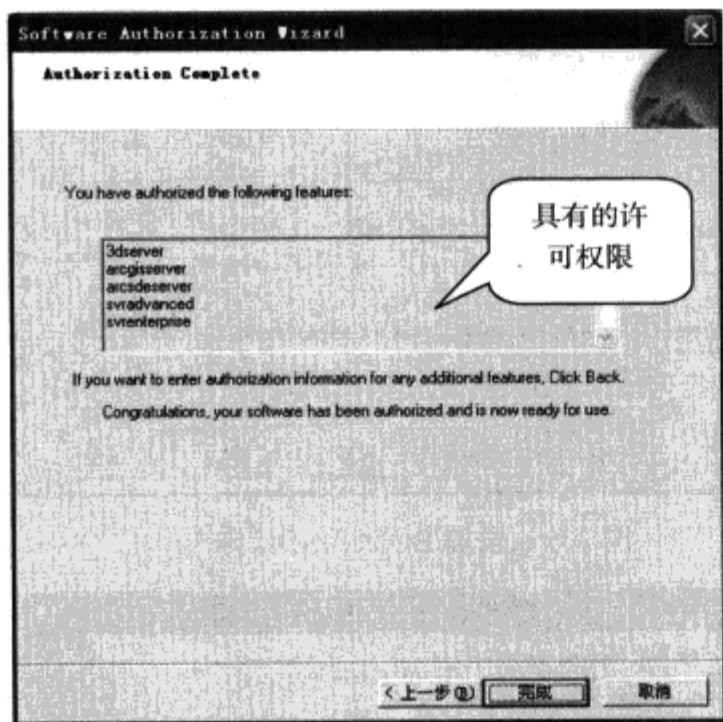


图 1.11 该许可具有的权限

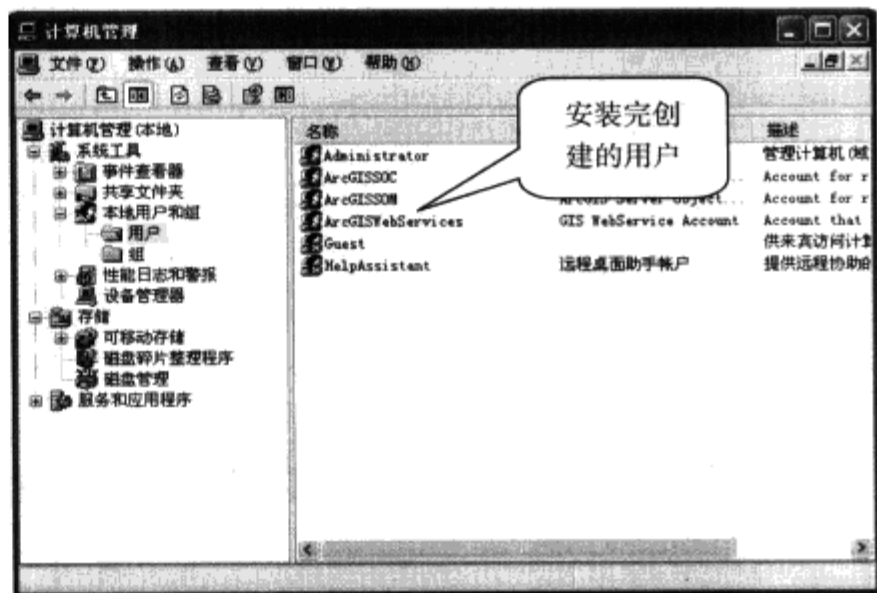


图 1.12 用户列表



图 1.13 ArcGIS Server 管理员用户组



图 1.14 ArcGIS Server 用户组

1.4.3 安装与配置说明

- 安装包括以下两个部分：
- (1) GIS Server 和 Web Applications。
 - Server Object Manager (SOM)；
 - Server Object Container (SOC)。
 - (2) Application Developer Framework。
- GIS Server Post Install 包括以下两步：
- (1) Configure GIS Server。

- 制定 SOM、SOC 账户：ArcGISSOM、ArcGISSOC；
- 制定 ArcGIS Manager 账户，将被加入 ArcGIS Server Administrators Group (agsadmin) 和 ArcGIS Server User Group (agsusers) 用户组 agsusers；
- 制定 ArcGIS Server 的工作目录。

(2) Authorize GIS Server。

安装时应注意的问题如下：

(1) 如果在 Windows XP +SP2 环境下应注意关掉“文件夹”选项中的“简单文件共享”。“简单文件共享”会启用特别的 Windows 认证方式，该认证方式与 ArcGIS Server 的不兼容。

(2) 关掉杀毒软件的防火墙。

(3) 关闭 Windows 自带的防火墙。

1.5 小结

本章主要对 ArcGIS Server 的体系结构、功能、安装进行了介绍，阅读完本章，读者应对 ArcGIS Server 的结构有大致地了解，对 ArcGIS Server 能够实现哪些功能有基本的认识，能够正确地安装和配置 ArcGIS Server。

第 2 章

ArcGIS与ArcGIS Engine

2.1 ArcGIS 软件体系结构

ArcGIS 是 ESRI 用现代主流技术针对 GIS 应用开发的一系列产品，它是一个完整的地理信息平台。包括桌面端 GIS、浏览器端 GIS、服务器端 GIS、移动 GIS、空间数据引擎以及二次开发组件。ArcGIS 系列产品，其体系结构如图 2.1 所示。

1. 桌面 GIS

ArcGIS 的桌面客户端产品包括 ArcInfo、ArcEditor、ArcView、ArcReader 和基于 ArcGIS Engine 组件二次开发的产品。桌面 GIS 是用户创建、编辑、设计和使用地理信息的主要应用程序。产品功能从 ArcReader、ArcView、ArcEditor 到 ArcInfo 依次增强。ArcInfo、ArcEditor、ArcView 3 个客户端都有统一风格的桌面应用：ArcCatalog、ArcMap 和 ArcToolbox。其中：

- **ArcCatalog** 是一个集成化的空间数据管理工具。用于空间数据浏览、Geodatabase 结构定义、空间数据导入导出、网络模型生成、对象关系和规则的定义、元数据的定义和编辑修改等。ArcCatalog 支持大量的数据格式，包括：ESRI shapefiles、geodatabases、ArcSDE layers、INFO tables、images、grids、TINs、CAD 文件、动态分段事件以及其他 ESRI 数据类型和文件等。
- **ArcMap** 是集空间数据显示、版本数据编辑、冲突数据检查、查询检索、统计、报表生成、空间分析和高级制图等众多功能于一体的桌面应用平台软件。ArcMap 提供类似 CAD 的空间数据编辑工具，全面支持空间数据的可视化交互操作。ArcMap 提供了所见即所得的符号编辑器，令用户可以随心所欲地生成任意复杂的点线面符号。ArcMap 拥有强大的空间数据直接读取能力，多种格式的空间数据无须进行转换或利用中间交换格式即可动态地直接读取。ArcMap 支持的空间数据格式包括：ArcInfo coverages、ESRI shapefiles、ArcSDE layers，DXF 和 DWG、DGN，大量的图像格式，GRID、TIN，等等。ArcMap 还支持空间数据的动态投影（on-the-fly projection）。

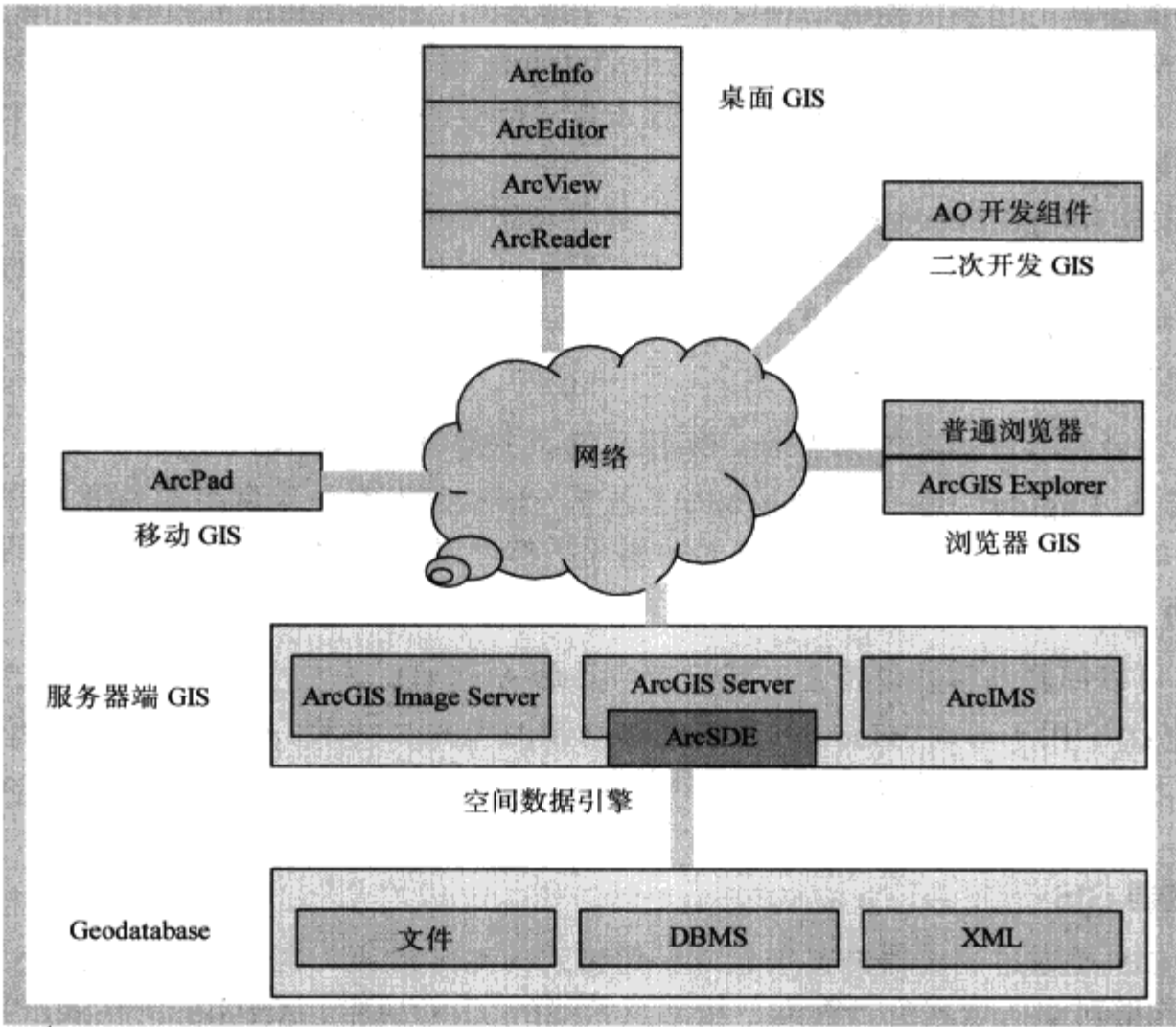


图 2.1 ArcGIS 软件体系结构图

- **ArcToolbox** 是用于空间数据格式转换、叠加处理、空间分析、坐标转换等的集成化“工具箱”。ArcToolbox 以树形结构方式组织了 140 多个不同的空间数据处理工具，并且都以菜单驱动的方式提供，这为用户以一种确定的、轻松的方式去完成复杂的工作提供了极大的方便。

ArcReader 是免费的地图和全球三维可视化浏览器。ArcReader 应用程序已经包含在基于 Intel 的微软 Windows, Sun Solaris 和 Linux 平台的 ArcGIS 桌面安装程序中。ArcReader 帮助用户以多种方式部署 GIS，并提供了开放的访问 GIS 数据的方式，可以在高质量的专业地图中展现信息。ArcReader 的使用者也可以交互地使用和打印地图，浏览和分析数据。

ArcObjects ArcGIS 开发的核心是 ArcObjects 软件组件库。ArcObjects 是跨平台 GIS 软件组件的集合。这套共享的 ArcObjects 库为 ArcGIS Desktop、ArcGIS Engine、ArcGIS Server 提供共同开发组件。它提供了模块、可伸缩、跨平台的结合和通用的 API，如 C++、.NET 和 Java。

2. 服务器 GIS

ArcGIS 服务器的 GIS 软件，可以在服务器集中管理 GIS 数据，并提供应用服务。产品包括 ArcGIS Server、ArcSDE、ArcIMS、ArcGIS Image Server 等软件。

- **ArcGIS Server** 功能强大的基于服务器产品，用于构建集中管理的、支持多用户、具备高级 GIS 功能的产品。ArcGIS Server 包含了 ArcSDE 空间数据管理技术。
- **ArcSDE** 空间数据引擎，用于对海量空间数据及其属性数据进行管理和驱动。ArcSDE 支持目前市面上流行的 DBMS，如 Oracle、SQL Server、DB2、Informix 等。ArcSDE 除了支持关系数据理论外，还引入其独特的异步缓冲和协同操作机制，使得空间数据的响

应速度提高。它不仅提供一种空间数据存储方式，还是真正的空间数据引擎。

- **ArcIMS** WebGIS 产品，基于 Internet/Intranet 发布动态地图和应用。ArcIMS 提供用户在服务器端可选的影像方式或矢量流方式进行空间数据发布，客户用标准的浏览器即可方便地浏览空间数据。
- **ArcGIS Image Server** 基于网络的、提供动态影像处理服务的服务器端软件。它可以根据用户要求完成海量影像数据的快速访问和可视化。在大量并发用户使用的环境下，无须借助关系数据库，能够快速显示海量的影像数据。

3. 二次开发 GIS

ArcGIS 为开发人员提供可编程的 GIS 开发包。主要包括 ArcGIS Engine 产品。

- **ArcGIS Engine** 是一个 ArcObjects 之上，用于开发客户化桌面端 GIS 的产品，它拥有强大的功能，在 ArcMap 能实现的功能几乎都可以用 ArcGIS Engine 开发实现。

4. 移动 GIS

ArcGIS 可以部署到移动设备上，其代表产品是 ArcPAD。

- **ArcPAD** 用于移动 Windows 设备的移动制图和 GIS 技术。ArcPAD 为野外作业用户通过手持和移动设备提供数据访问、制图、GIS 和 GPS 的综合应用。通过 ArcPAD 可以实现快速、便捷的数据采集，大大提高了野外作业数据的可用性和有效性。

5. 浏览器 GIS

浏览器 GIS 是通过浏览器浏览矢量或影像格式的 GIS 数据，与传统的桌面 GIS 数据的浏览工具一样，能够对空间数据进行查询、检索及其他的 GIS 功能。ArcGIS 的代表产品是 ArcGIS Explorer。

- **ArcGIS Explorer** 是一个免费的虚拟地球浏览器，它提供自由、快速的二维和三维地理信息浏览，同时还提供空间数据的查询和分析功能。ArcGIS Explorer 通过继承 ArcGIS Server 完整的 GIS 性能（包括空间处理和三维服务），达到整合丰富的 GIS 数据集和服务空间处理应用的目的。ArcGIS Explorer 支持 OGC WMS 以及 Google KML 数据。

2.2 组件对象模型

组件对象模型（Component Object Model, COM）是微软 1993 年提出的元件式软件开发平台。它不仅定义了组件程序进行交互的标准，而且提供组件程序运行所需环境的 API，并提供类似客户对组件的查询、注册以及反注册等一系列服务。在 COM 结构中，对象的使用者通常称为客户。一般来说 COM 库由操作系统来实现，客户不必关心其实现的细节，如我们经常看到的 ActiveX、DirectX、OLEDB 都是基于 COM 技术的，主要应用于 Microsoft Windows 操作系统平台上。通常，COM 是以 Win32 动态链接库（DLL）或可执行文件（EXE）的形式发布。

在 COM 中接口就是一切，对于客户来说，一个组件就是一个接口集。COM 接口是一个包含一个函数指针数组的内存结构。组件本身只不过是接口的实现细节。接口的优点是，保护系统免受外界变化的影响，客户可以用同样的方式处理不同的组件。接口具有二进制标准，因此一个接口必须具有一定的结构，是关于如何建立组件以及如何建立应用程序的一个规范，说明如何动态更新组件。

对象是 COM 的基本要素之一，和 C++ 中的对象不同的是其封装特性是真正意思上的封装，对于对象使用者而言是不可见的。此外，COM 对象的可重用性表现在 COM 对象的包容和聚合，

一个对象可以完全使用另外一个对象的所有功能，而 C++ 对象的可重用性表现在 C++ 类的继承性。

接口是指组件对象的接口，它是包含了一组函数的数据结构，通过这组数据结构，客户代码可以调用组件对象的功能，组件对象间的访问都是通过接口来进行的。这个接口就是 IUnknown。IUnknown 接口的定义包含在 Win32 SDK 中的 unknown.h 的头文件中，引用如下：

```

///IUnknown 的定义
interface IUnknown
{
virtual HRESULT __stdcall QueryInterface(const IID& iid,void **ppv)=0;
virtual ULONG __stdcall AddRef()=0;
virtual ULONG __stdcall Release()=0;
};

```

所有的 COM 都要继承 IUnknown。可以用 IUnknown 的接口指针来查询该组件的其他接口，并且每个接口的 vtbl 中前 3 个函数都是 QueryInterface、AddRef 和 Release。这使得所有的 COM 接口都可以被当成 IUnknown 接口来处理。由于所有的接口都支持 QueryInterface，因此组件的任何一个接口都可以被客户用来获取它所支持的其他接口。

在用 QueryInterface 将组件抽象成由多个相互独立的接口构成的集合后，还需要管理组件的生命期。这一点是通过对接口的引用计数实现的，客户并不能直接控制组件的生命期。当使用完一个接口而要用组件的另一个接口时，是不能将该组件释放的。对组件的释放可以在客户使用完所有的组件后由组件自己完成。IUnknown 的另外两个成员函数 AddRef 和 Release 的作用就是给客户提供一种让它指示何时处理完一个接口的手段。AddRef 和 Release 实现的是一种名为引用技术的内存管理技术。当客户从组件获得一个接口时，此引用计数值将增 1。当客户使用完某个接口时，组件的引用计数值将减 1，当引用计数值为 0 时，组件可以将自己从内存中删除。AddRef 和 Release 可以增加和减少这一计数值。

在使用 COM 库函数之前，必须调用 CoInitialize 函数对 COM 库进行初始化。客户在获取某个组件接口指针之前，必须先将相应的 DLL 装载到其进程空间并创建此组件。CreateInstance 可以建立一个组件的实例并为客户返回一个 IUnknown 接口指针。之后，可以装载 DLL 并调用其中的函数。此功能可由 COM 库函数 CoCreateInstance 来实现。CoCreateInstance 创建组件的过程是：传给它一个 CLSID，然后它创建相应的组件，并返回指向所请求的接口的指针。事实上，常用类工厂来创建组件，类工厂就是一个带有能够创建其他组件接口的组件。客户先创建类工厂本身，然后再用一个接口（如 IClassFactory）来创建所需的组件，并用 DLLRegisterServer 在 Windows 中注册这个组件。

ArcGIS 9.3 中，其核心功能都被封装在 21 个 COM 组件对象库中，其中基本的有图形库 esriGeometry、显示库 esriDisplay、制图库 esriCarto 等，扩展的对象库有三维分析库 esri3DAnalyst、网络分析库 esriNetworkAnalyst 等。

2.3 ArcObjects 简介

2.3.1 ArcObject 的组织划分

ArcObjects 的对象模型分为基本模块和扩展模块。基本模块主要用于图形的显示、基本操作

(如放大、缩小、漫游和全图显示)、查询检索等。扩展模块包括空间分析、三维分析和图形网络发布等模块，如图 2.2 所示。

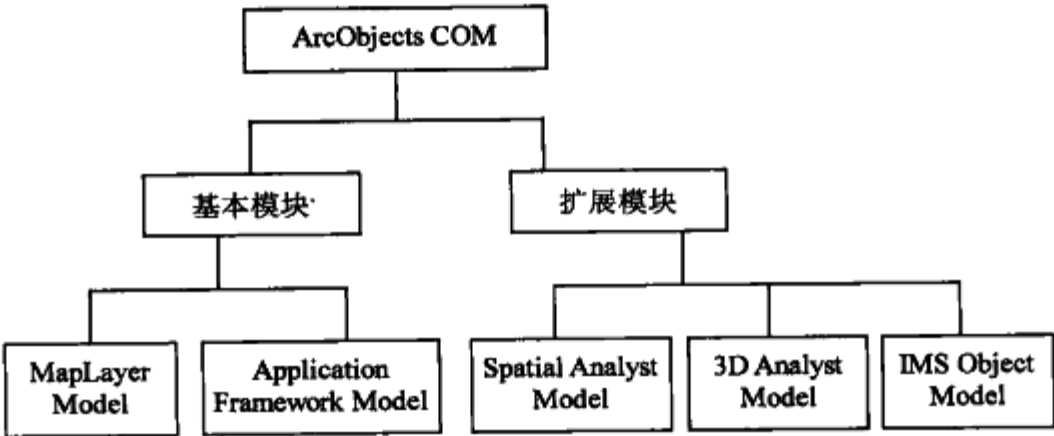


图 2.2 ArcObjects 组件结构图

2.3.2 ArcObject 的开发

ArcObject 可选的开发方式可以分为两种：一种是在 ArcMap 应用框架基础上进行定制开发，另一种是脱离 ArcMap 应用框架去开发独立的应用程序。通常情况下，开发一些功能比较少的定制应用时，开发环境可以选择 ArcMap 本身自带的 VBA，而开发一些复杂的应用时，选择脱离 ArcMap 环境，开发独立的应用程序。

1. 如何在 ArcMap 的 VBA 环境中编程

ArcMap 是 ArcGIS 家族的成员之一，它内置了一种集成编程环境——VBA (Visaul Basic for Apllications)。通过 VBA 编程，用户不但可以扩展 ArcMap 的菜单、工具栏等，而且可以完成大多数用户的特定需求。

ArcMap 中 VBA 编程方法有两种：一种是写 VBA 宏，另一种是创建 UIControl 并在其事件中写入实现用户需求的代码。下面列出两种方法的一般步骤。

方法一：写 VBA 宏（直接在 VBA 编辑器中编辑函数和过程）。

(1) 如图 2.3 所示，单击菜单栏中的“Tools”命令，选择“Macros”→“Visual Basic Editor”项，直接启动 ArcMap 的 VBA 编辑器；或者选择“Macros”→“Macros”项，进入如图 2.4 所示的“Macro”对话框，在“Macro Name”文本框中输入要创建的宏的名称，并单击“Create”按钮，启动 VBA 编辑器。

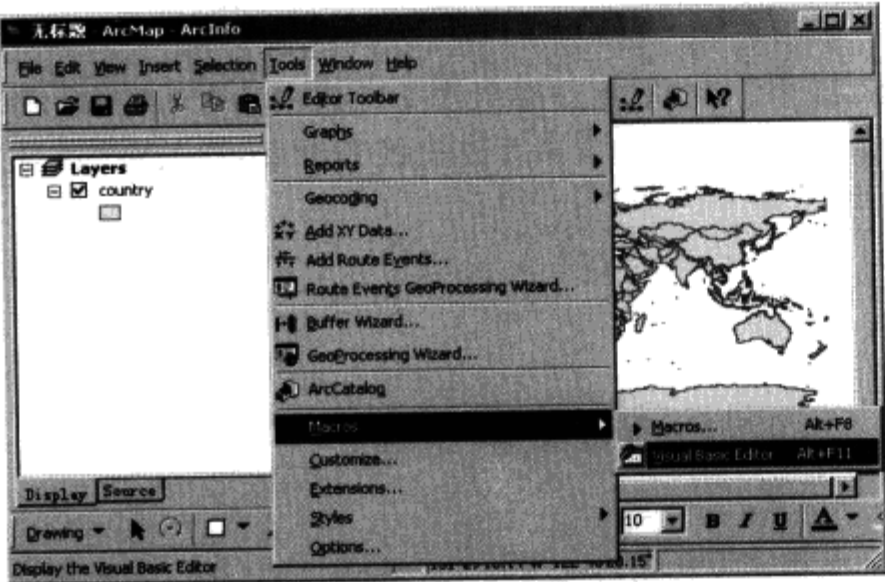


图 2.3 启动 Macro 对话框/启动 VBA 编辑器

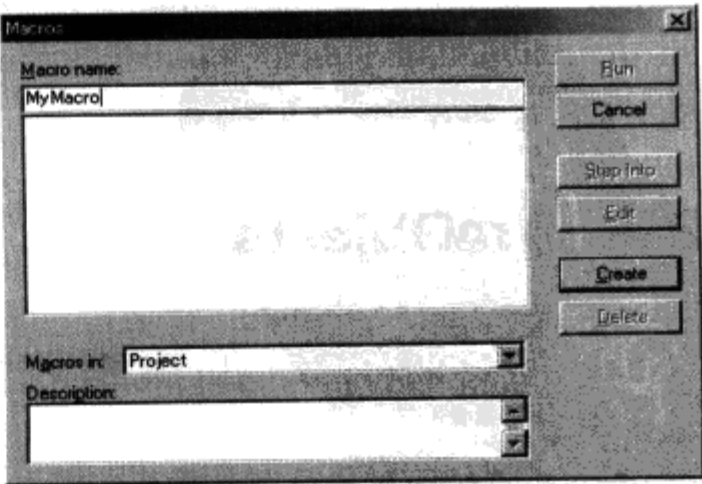


图 2.4 “Macro”对话框

(2) 在图 2.5 所示的窗口中, 用户可以根据实际需要选择在 Normal 节点或者 Project 节点的 ThisDocument、Forms、Modules 中编写宏 (函数或过程)。在 Normal 节点下所写的宏系统自动保存, 除非用户删除, 否则它将始终存在并在任何项目中都有效; 而在 Project 节点下所写的宏随项目保存 (如不保存项目, 则宏也将不被保存), 并只在项目中有效。



图 2.5 VBA 编辑器 (VBE)

(3) 运行 VBA 宏。

在 VBA 编辑器中写好 VBA 代码后, 有两种方式运行: 第一, 单击 VBA 编辑器工具栏中的 (运行) 按钮, 可立即运行写好的代码; 第二, 退出 VBA 编辑器, 重新启动 Macro 对话框 (如图 2.4 所示), 选择要运行的 VBA 宏名称, 单击 “Run” 按钮即可运行相应的 VBA 宏。

方法二: 创建 UIControl (交互式 VBA 编程)。

(1) 用鼠标右击任何工具栏, 在弹出的菜单中选择 “Customize” 菜单项, 如图 2.6 所示, 进入如图 2.7 所示的 “Customize” 对话框。

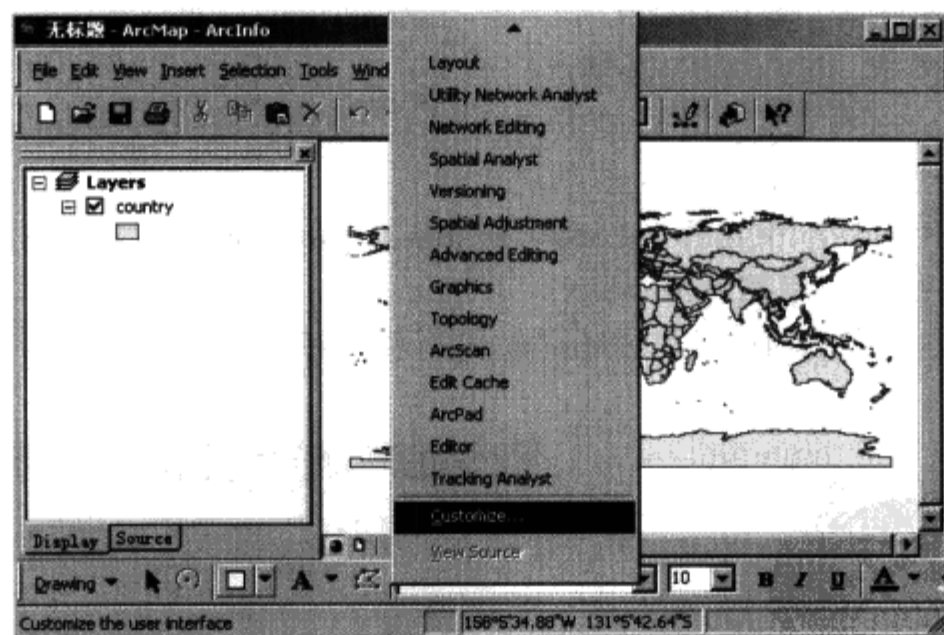


图 2.6 启动 “Customize” 对话框

(2) 在 “Customize” 对话框的 “Commands” 标签页, 选中 “UIControls” 项后单击 “New UIControl” 按钮, 进入如图 2.8 所示的 “New UIControl” 对话框。

(3) 在 “New UIControl” 对话框中, 用户可根据需要选择 UIControl 的类型。

- UIButtonControl: 创建 Button。
- UIToolControl: 创建与 Map 交互的 Tool。

- **UIEditBoxControl**: 创建 EditBox。
- **UIComboBoxControl**: 创建 ComboBox。

最后单击“Create”按钮只创建 UIControl，或者单击“Create and Edit”按钮创建 UIControl 并进入 VBA 编辑器。与方法一不同，此时应在 UIControl 的事件中进行 VBA 编程。

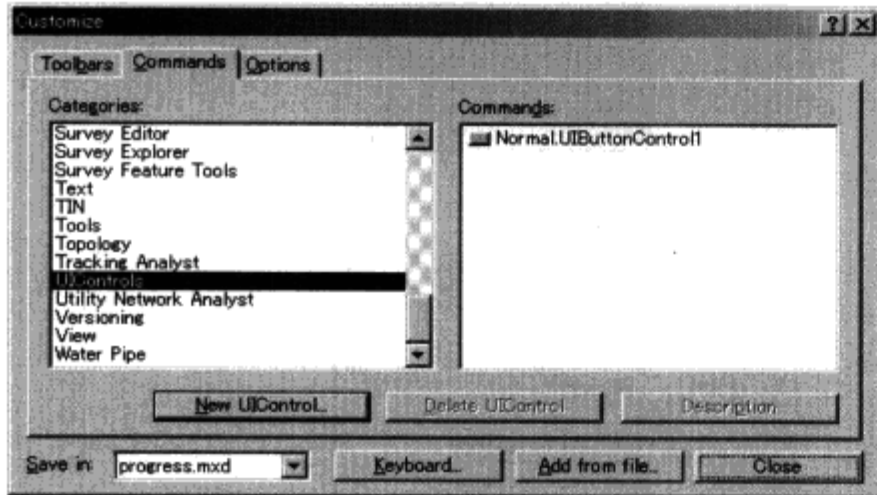


图 2.7 “Customize” 对话框

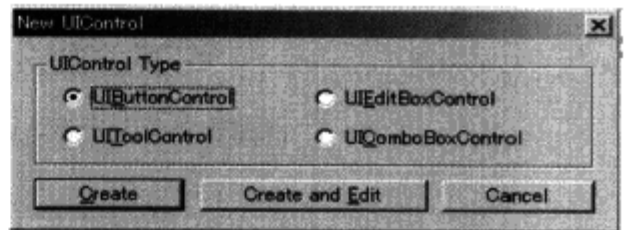


图 2.8 “New UIControl”对话框

(4) 创建 UIControl 后, 在如图 2.7 所示的“Customize”对话框中选择 UIControls 并将其拖放到任意工具栏上, 用户便可像使用系统中已有的 Control 一样使用所创建的 UIControl。

2. 如何在 Visual Basic 环境中利用 ArcObjects 控件开发 EXE 或 ActiveX DLL

在某些情况下，对于特殊的应用，用户需要脱离 ArcGIS 环境而在 Visual Basic 开发环境开发外部独立的应用程序，这个外部独立的应用程序有两种形式：EXE 或 ActiveX DLL。下面介绍在 VB 环境下，利用 ArcObjects 对象库实现开发独立的 EXE 程序，其他的开发语言与此类似。

在 VB 环境利用 ArcObjects 组件开发 EXE 的步骤如下。

(1) 启动 VB 开发环境, 在如图 2.9 所示的“New Project”对话框中选择“Standard EXE”项, 并单击“打开”按钮, 进入 VBE 环境。

(2) 引用 ArcObjects 对象库。首先单击“Project”菜单中的“References”项，如图 2.10 所示，进入对象库引用对话框，如图 2.11 所示。

(3) 在对象库引用对话框中选择“ESRI ArcMap Object Library”和“ESRI Object Library”两项，并单击“OK”按钮，返回 VBE 环境。

(4) 单击“Project”菜单项中的“Components”项, 打开“Components”对话框, 如图 2.12 所示。

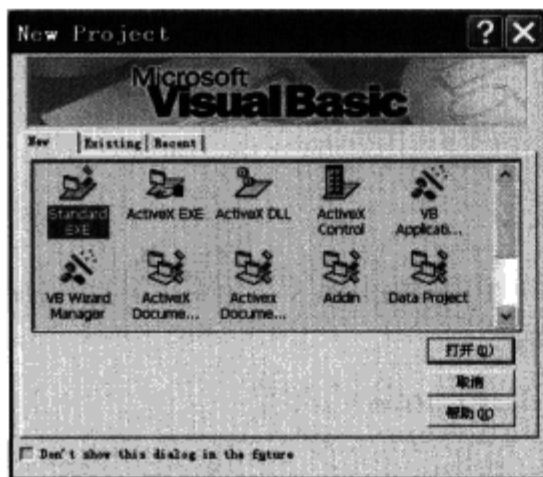


图 2.9 “New Project” 对话框

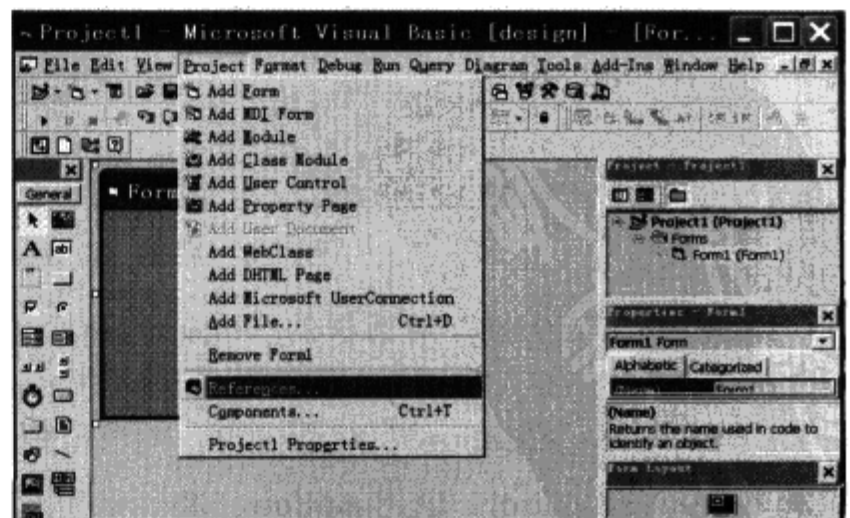


图 2.10 启动对象库引用对话框

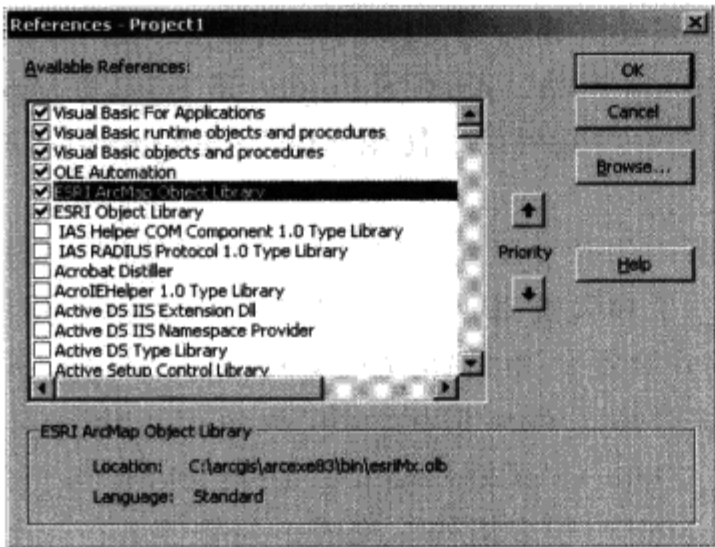


图 2.11 对象库引用对话框

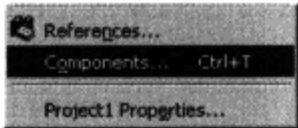


图 2.12 打开 Components 对话框

(5) 在“Components”对话框中，切换到 Controls 标签页，并选中“ESRI MapControl”项，单击“应用”按钮或“确定”按钮，如图 2.13 所示。

(6) 如图 2.14 所示，加载 MapControl 控件之后，在 VBE 的控件面板中出现了 MapControl 控件图标，用户便可以像在 Form 中添加 Button 一样在 Form 中添加 MapControl 控件，并利用它开发 EXE。

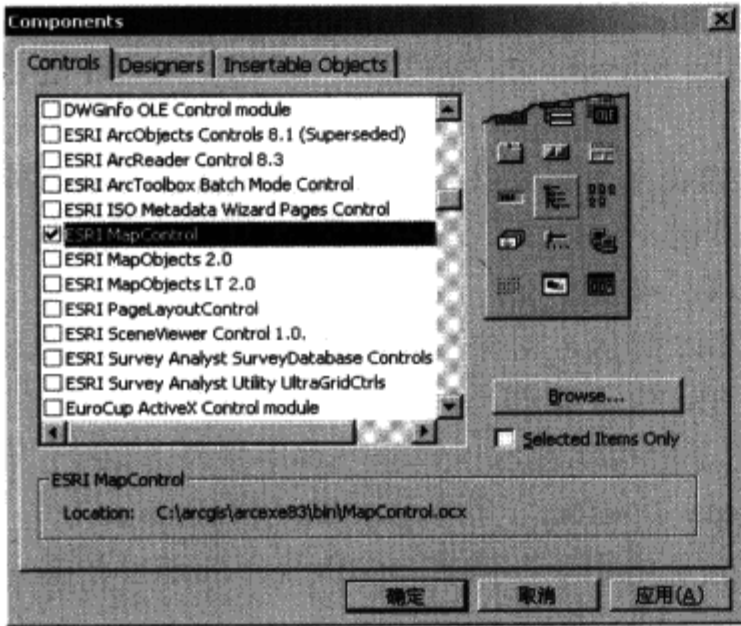


图 2.13 “Components”对话框

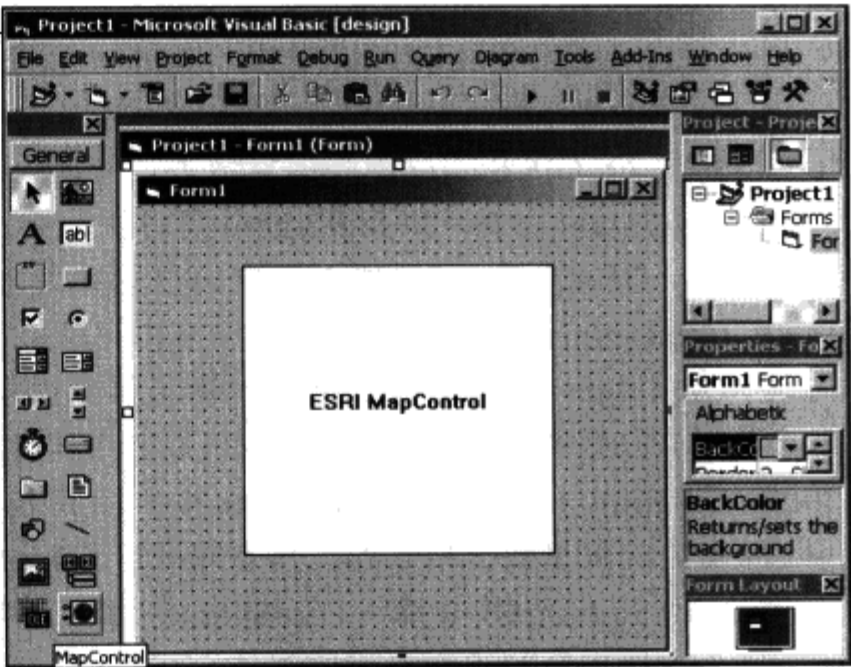


图 2.14 添加 MapControl 控件

2.4 ArcGIS Engine

2.4.1 ArcGIS Engine 构成

ArcObjects 是所有 ArcGIS 平台的 API，整个 ArcGIS 都是基于其上构建的。ArcGIS Engine 是将 ArcObjects 部分功能封装，提供给用户的一个单独的二次开发包。ArcGIS Engine 是一个产品，而 ArcObjects 不是一个产品，它是基础，ArcGIS Engine 是 ArcObjects 的简化版本，并且可以打包发布，ArcObjects 是不能脱离 ArcMap 平台的。在 2.3 节中已经讲过，利用 ArcObjects 开发应用有 3 种方法：一种是 VBA，一种是开发能够嵌入 ArcGIS Desktop 的 DLL 文件，另外一种

就是使用 MapControl、LayoutControl 控件开发具有独立界面的 GIS 的应用程序。这 3 种开发方式都要求客户端必须安装一定级别的 ArcGIS Desktop 产品。ArcGIS Engine 是 ESRI 在 ArcGIS 9 版本才开始推出的新产品，它是一套完备的嵌入式 GIS 组件库，使用 ArcGIS Engine 开发的应用程序，可以脱离 ArcGIS Desktop 而运行。

ArcGIS Engine 由两个产品组成：构建软件所用的开发工具包以及使已完成的应用程序能够运行的可再发布的 Runtime。ArcGIS Engine 开发工具包是基于组件的软件开发产品，可用于构建自定义 GIS 和制图软件。它并非一个终端用户产品，而是针对开发人员的工具包，适于为 Windows、UNIX 或 Linux 用户构建基础制图和综合动态 GIS 应用软件。ArcGIS Engine Runtime 是一个使终端用户软件能够运行的核心 ArcGIS Objects 组件产品，并且被安装在每一台运行 ArcGIS Engine 应用程序的计算机上。ArcGIS Engine 由很多组件库构成，列举如下：

- **System 库** System 是 ArcGIS 系统最底层的一个库，该库包含了揭示 ArcGIS 中其他组件库所使用的组件，这些组件非常底层，如数组、集合等。
- **SystemUI 库** SystemUI 库主要定义了 ArcGIS 系统中所使用的用户界面组件对象，如 ICommand、ITool 等接口。
- **Geometry 库** Geometry 库包含了核心几何对象，如点、线、多边形及几何类型和定义等。除了几何对象之外，还包含空间参考对象，如几何坐标系、投影系统、坐标转换等。
- **Display 库** Display 库包含了支持向输出装置绘制符号体系的组件，它包括 Display 对象、Color 对象、ColorRamp 对象、Symbol 对象等。
- **Server 库** Server 库包含了用于获取到 ArcGIS Server 连接的对象。ArcGIS Server 连接通过 ArcGIS Server 库中的对象进行管理。
- **Geodatabase 库** Geodatabase 是一种采用标准关系数据库技术来表现地理信息的数据模型。Geodatabase 支持在标准的数据库管理系统（DBMS）表中存储和管理地理信息。Geodatabase 库包含了所有与数据访问相关定义的类型。
- **GISClient 库** GISClient 库包含了操作远程 GIS 的服务对象（ArcIMS 或 ArcGIS Server 都可以提供这些服务对象），可以通过本地 GISClient 来调用远程的 ArcGIS Server 服务。
- **DataSourcesFile 库** DataSourcesFile 库包含了使用地理数据库应用程序接口（API）所支持的矢量数据格式的工作空间工厂（WorkspaceFactory）和工作空间（Workspace）。
- **DataSourcesGDB 库** DataSourcesGDB 库包含了适用于存储在 RDBMS 中的地理数据库所支持矢量和栅格格式的工作空间工厂和工作空间。
- **GeoDatabaseDistributed 库** GeoDatabaseDistributed 库包含了支持分布式地理数据库的检出（checkout）/检入（checkin）操作所必需的对象。
- **DataSourcesOleDB 库** DataSourcesOleDB 库提供了操作“基于对象连接和嵌入数据库”（OLE_DB-based）的数据源的工作空间。
- **DataSourcesRaster 库** DataSourcesRaster 库包含了适用于基于文件的、栅格数据格式的工作空间工厂和工作空间。
- **Carto 库** Carto 库包含了用于显示数据的对象，如指北针、图例、比例尺等。
- **Location 库** Location 库包含了与位置数据操作相关的对象。位置数据可以是路径事件，或地理编码的位置。
- **NetworkAnalysis 库** NetworkAnalysis 支持应用网络的创建和分析。
- **Controls 库** Controls 库包含了用于软件开发的控制器，包含通过控制器来使用的命令和工具。

- **GeoAnalyst 库** GeoAnalyst 库包含了核心空间分析功能，这些功能在空间分析（SpatialAnalyst）模块和 3D 分析扩展模块中都会用到。
- **3Danalyst 库** 3Danalyst 库包含了用于进行数据 3D 分析以及支持 3D 数据显示的对象。该库中有一控件（SceneControl）可供开发人员使用，需要说明的是使用该库需要一个 3D 分析扩展授权。
- **GlobeCore 库** GlobeCore 库包含用于进行球体数据分析以及支持球体数据显示的对象。该库中有一控件（GlobeControl）可供开发人员使用，使用该库也需要一个 3D 分析扩展授权。
- **SpatialAnalyst 库** SpatialAnalyst 库包含了用于进行栅格与矢量数据空间分析的对象。操作该库中的对象需要一个空间分析授权。

2.4.2 ArcGIS Engine 功能

如上所述，ArcGIS Engine 是将 ArcObjects 部分功能封装，提供给用户的一个单独的二次开发包。ArcGIS Engine 的功能十分强大，那么用 ArcGIS Engine 来进行二次开发，具体能够实现哪些功能呢？下面一一列举。

- 显示多个图层组成的地图，能够打开或关闭某一图层；
 - 漫游和缩放地图；
 - 查找地图中的要素；
 - 用要素的某一字段显示标注；
 - 显示航片和遥感影像的栅格数据；
 - 绘制几何要素；
 - 绘制描述性的文字；
 - 沿线或者用多边形、圆等要素选择；
 - 根据一定距离选择要素；
 - 通过 SQL 表达式查询要素；
 - 渲染要素；
 - 动态显示实时数据或时间系列数据；
 - 地图定位；
 - 创建和更新地理要素和属性；
 - 读取所有支持的 ESRI 数据格式包 Geodatabase；
 - 地图制作（创建和编辑 MXD）；
 - 地理编码；
 - 空间数据版本管理、冲突检测；
 - 空间分析；
 - 3D 显示。
- 通过这些基本功能还可以组合出更高级的功能，在此不再列举了。

2.4.3 ArcGIS Engine 开发环境

ArcGIS Engine 开发包提供 4 种开发方式：COM、.NET、Java、C++。使用 COM 方式编程，

通常可以选用的开发工具有 Visual Studio 6.0 (VB、VC++)、Delphi；采用.NET 方式开发可以选用的开发工具有 Visual Studio .NET (VB.NET、C#、VC++)；采用 C++方式进行开发，常用的开发工具有 Visual Studio 6.0 、Borland C++、C++等；利用 Java 进行开发，常见的开发工具有 JBuilder、Eclipse、JDK 等。本小节将介绍利用常见的开发工具 Visual Basic、Visual C++、Visual Studio .NET 等搭建 ArcGIS Engine 的开发环境，有关如何搭建 Visual Basic 与 ArcGIS Engine 的开发环境参见 2.3.2 小节，这里不再赘述。

1. 配置 ArcGIS Engine 在 VC++6.0 中的开发环境

(1) 添加新工程。

启动 VisualC++6.0 的开发环境，在“New”对话框中选择“Projects”中的“MFC AppWizard(exe)”项，在“Project name”和“Location”文本框中输入项目名称和项目存放的路径，如图 2.15 所示。

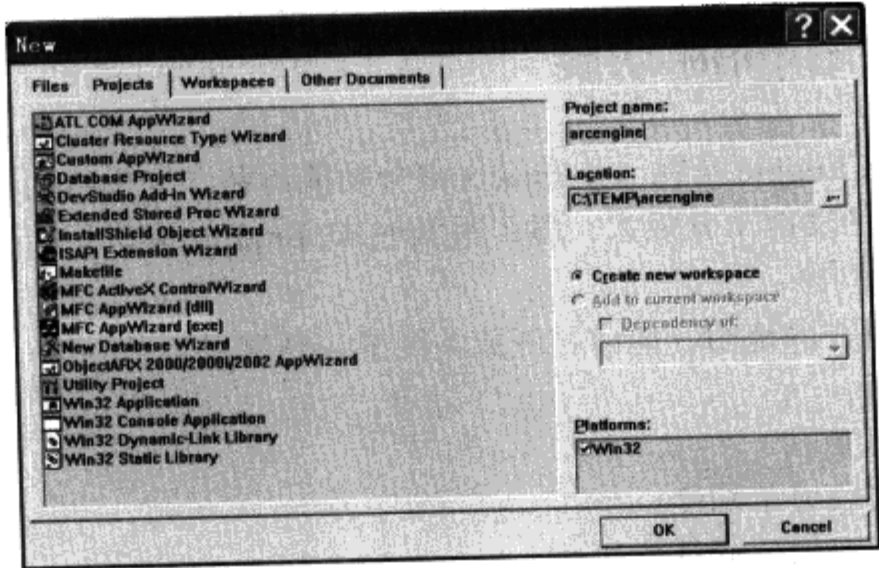


图 2.15 “New”对话框

(2) 配置库路径。

单击“Project”菜单中的“Settings”菜单项，修改项目属性。在“Project Settings”对话框中选择“Debug C/C++”标签页，在“Category”下拉列表中选择“Preprocessor”，在“Additional include directories”文本框中，添加头文件和库的路径，比如：C:\Program Files\ArcGIS\Include\CPPAPI；C:\Program Files\ArcGIS\com，如图 2.16 所示。

(3) 添加 Symbol。

在“Debug C/C++”标签页的“Preprocessor definitions”文本框中，在所有 SYMBOL 的末尾，添加“，ESRI_WINDOWS”，如图 2.17 所示。

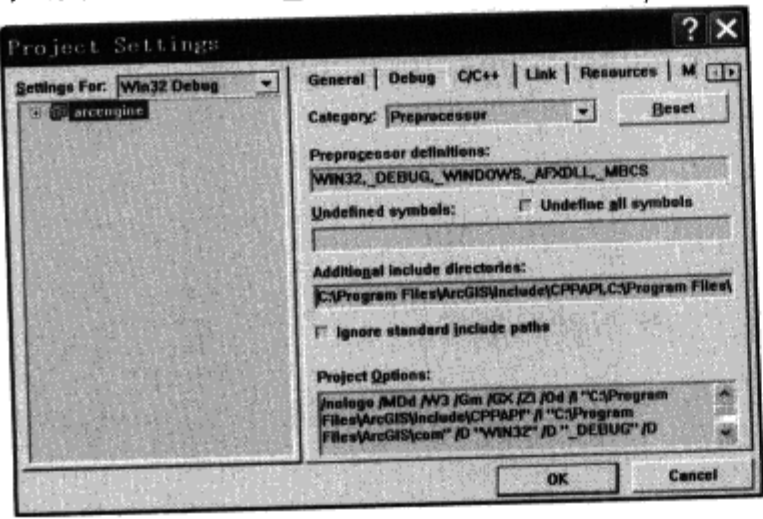


图 2.16 “Project Settings”对话框 1

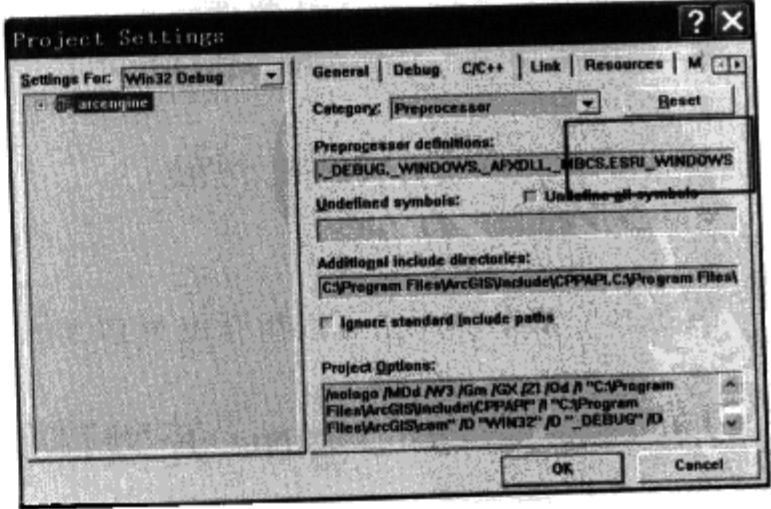


图 2.17 “Project Settings”对话框 2

(4) 引入头文件。

在需要使用 ArcObjects 组件的类文件中，需要引入头文件定义 ArcSDK.h，通常可以在 Stdafx.h 中引入，比如：`#include <ArcSDK.h>`，如图 2.18 所示。

(5) 使用控件。

如果使用 MapControl、TocControl、SceneControl、ToolbarControl 等控件，有两种方式。

● 添加控件到项目中。

如果需要在项目中生成相应的类，可以选择“Project”菜单中的“Add to Project”菜单中的“Components and Controls”选项，在弹出的对话框中添加相应控件类，如图 2.19 所示。

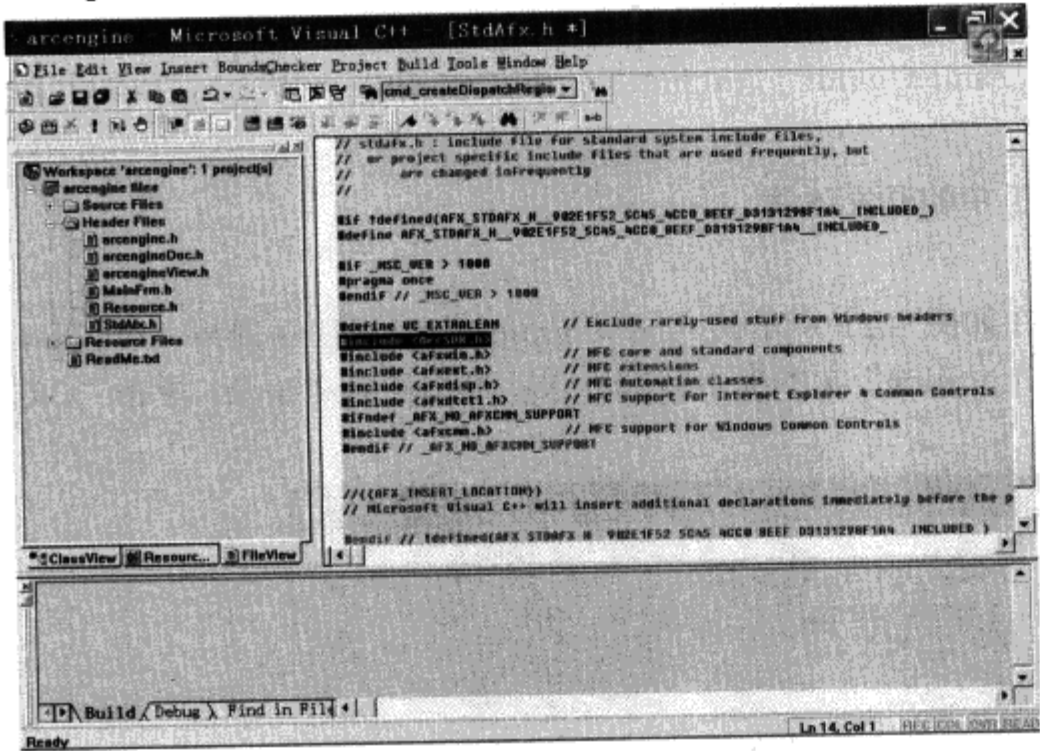


图 2.18 引入头文件

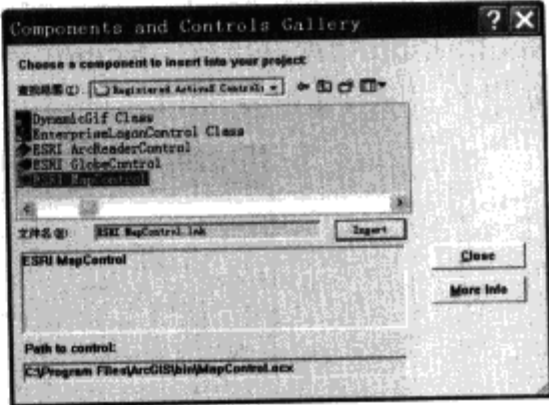


图 2.19 添加控件

如果不需要在项目中生成相应的类，可以在对话框资源或者 FormView 中的右键菜单中选择“Insert ActiveX Controls”选项，如图 2.20 所示。

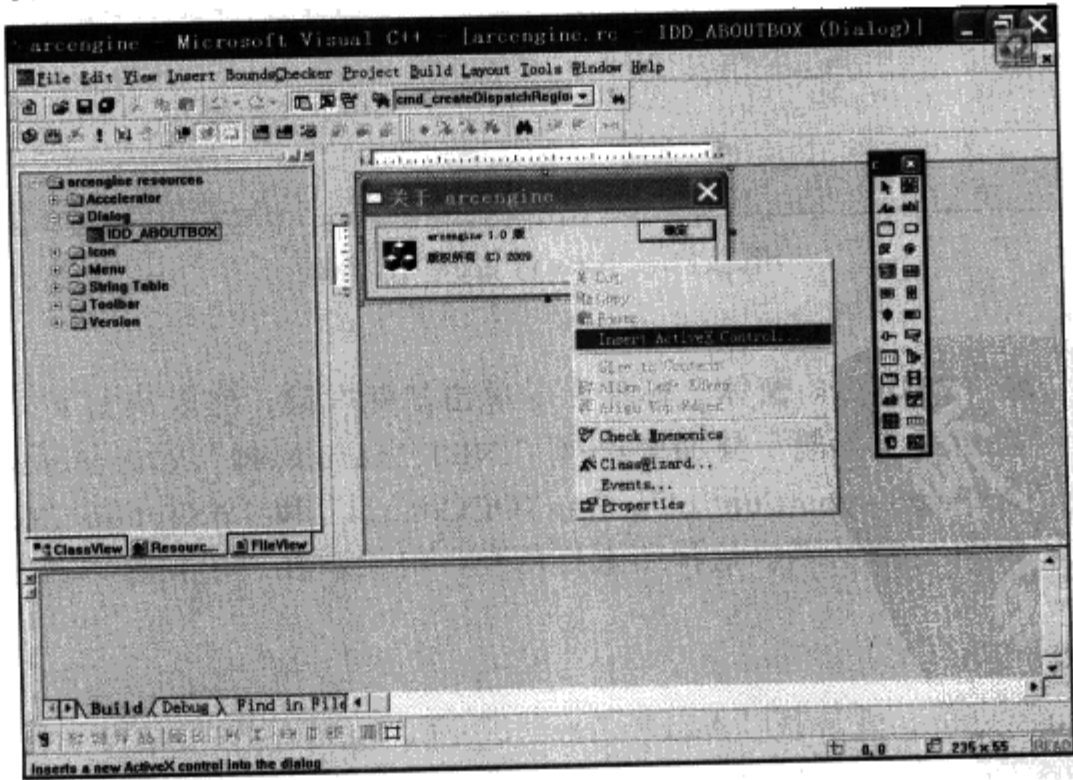


图 2.20 不需要生成相应控件类

- 引入 OCX 文件。

通常可以在 StdAfx.h 中引入 OCX 文件，例如：

```
#import<C:\Program Files\ArcGIS\bin\mapcontrol.ocx> raw_interfaces_only
raw_native_types no_namespace named_guids exclude("OLE_COLOR", "OLE_HANDLE",
"ICursorPtr", "VARTYPE")
```

```
#import<C:\Program Files\ArcGIS\bin\PageLayoutControl.ocx> raw_interfaces_only
raw_native_types no_namespace named_guids exclude("OLE_COLOR", "OLE_HANDLE",
"ICursorPtr", "VARTYPE")
```

```
#import<C:\Program Files\ArcGIS\bin\TOCControl.ocx> raw_interfaces_only
raw_native_types no_namespace named_guids exclude("OLE_COLOR", "OLE_HANDLE",
"ICursorPtr", "VARTYPE")
```

2. 配置 ArcGIS Engine 在 .NET 中的开发环境

(1) 添加新项目。

启动 Visual .NET 开发环境，在如图 2.21 所示的“New Project”对话框中选择“Windows Application”项，并在“Name”和“Location”文本框中分别输入项目文件名和路径，单击“OK”按钮，进入 C# 开发环境。

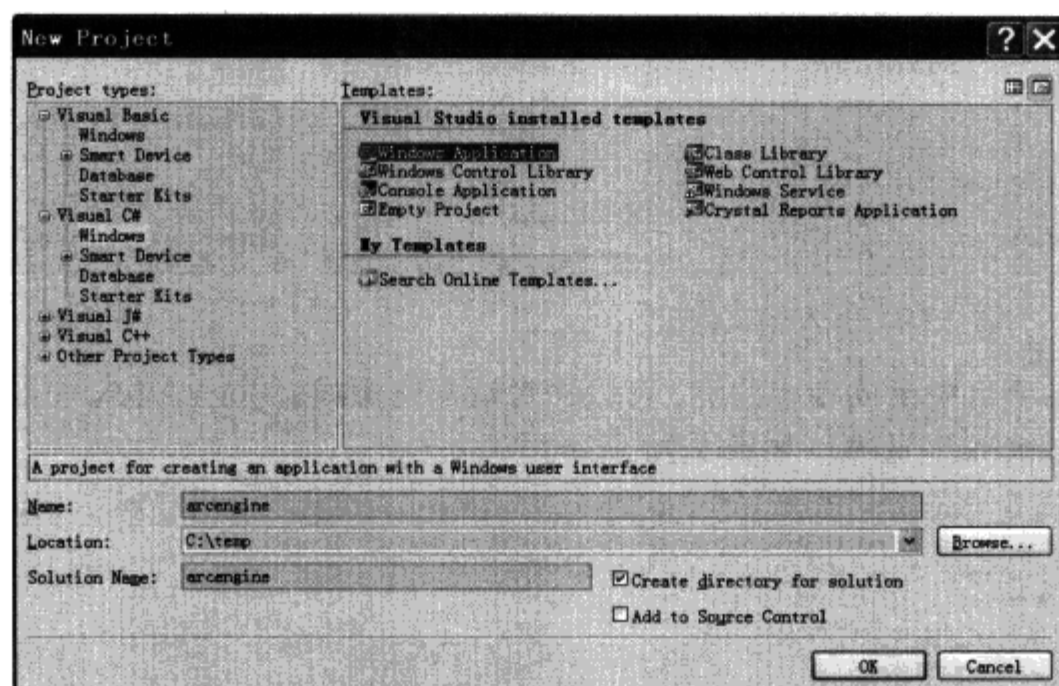


图 2.21 “New Project”对话框

(2) 添加控件。

在“工具箱”的“Windows 窗体”标签栏中单击鼠标右键，然后从上下文菜单中选择“添加/移除”项。在“自定义工具箱”界面中选择“.NET Framework Components”标签，并选择“AxMapControl”、“AxPageLayoutControl”、“AxTOCControl”和“AxToolBarControl”等复选框，单击“OK”按钮，如图 2.22 所示。这样所选择的控件将显示在“工具箱”的“Windows 窗体”标签栏中。

(3) 添加应用。

选择“Projects”菜单，并选择“Add Reference”项。在打开的“Add Reference”对话框中，双击“ESRI.ArcGIS.Carto”、“ESRI.ArcGIS.Display”、“ESRI.ArcGIS.Geometry”、“ESRI.ArcGIS.System”、“ESRI.ArcGIS.SystemUI”、“ESRI.ArcGIS.Utility”。单击“OK”按钮，如图 2.23 所示。

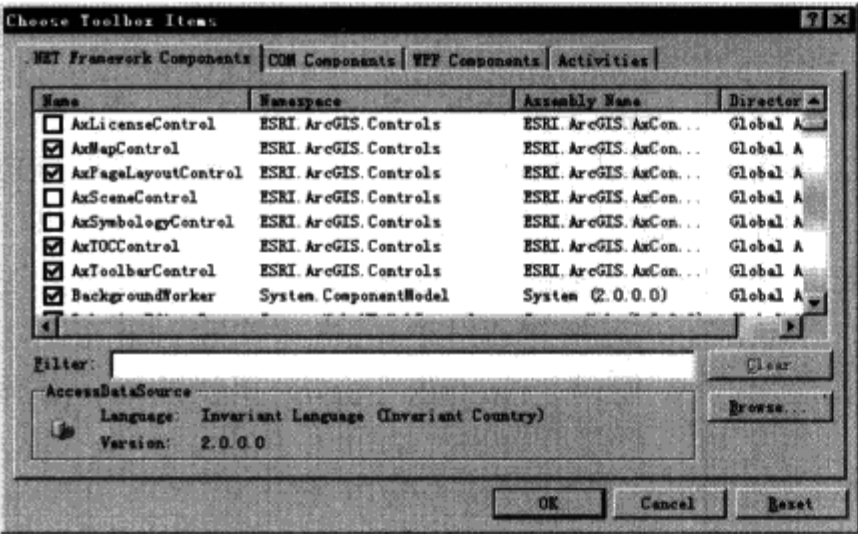


图 2.22 “自定义工具箱”界面

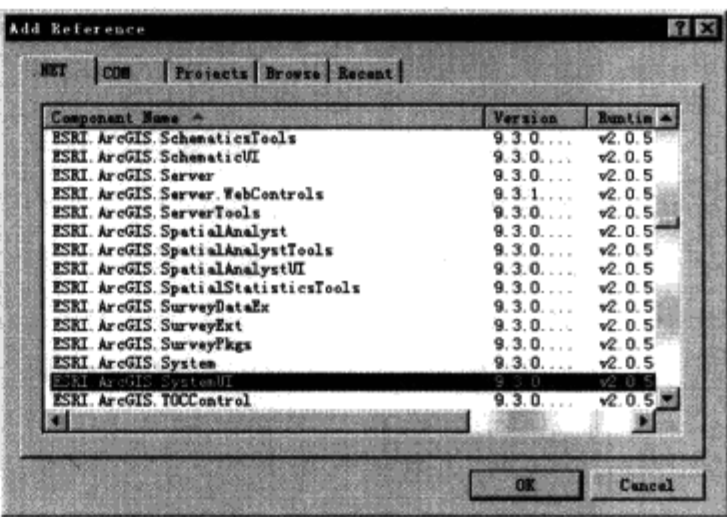


图 2.23 “Add Reference”对话框

(4) 在窗体上添加具体的控件。

双击工具箱“Windows 窗体”标签栏中的 AxMapControl 控件, 将 MapControl 添加到窗体上。再将 AxPageLayoutControl、AxTOCControl 和 AxToolbarControl 也添加到窗体中, 如图 2.24 所示。

(5) 在代码中加入引用。

在窗体上双击显示窗体代码窗口, 在代码窗口的顶部增加“using”命令。

```
using System;
using System.Windows.Forms;
// ArcGIS Engine 引用
using ESRI.ArcGIS.SystemUI;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.ToolbarControl;
using ESRI.ArcGIS.TOCControl;
```

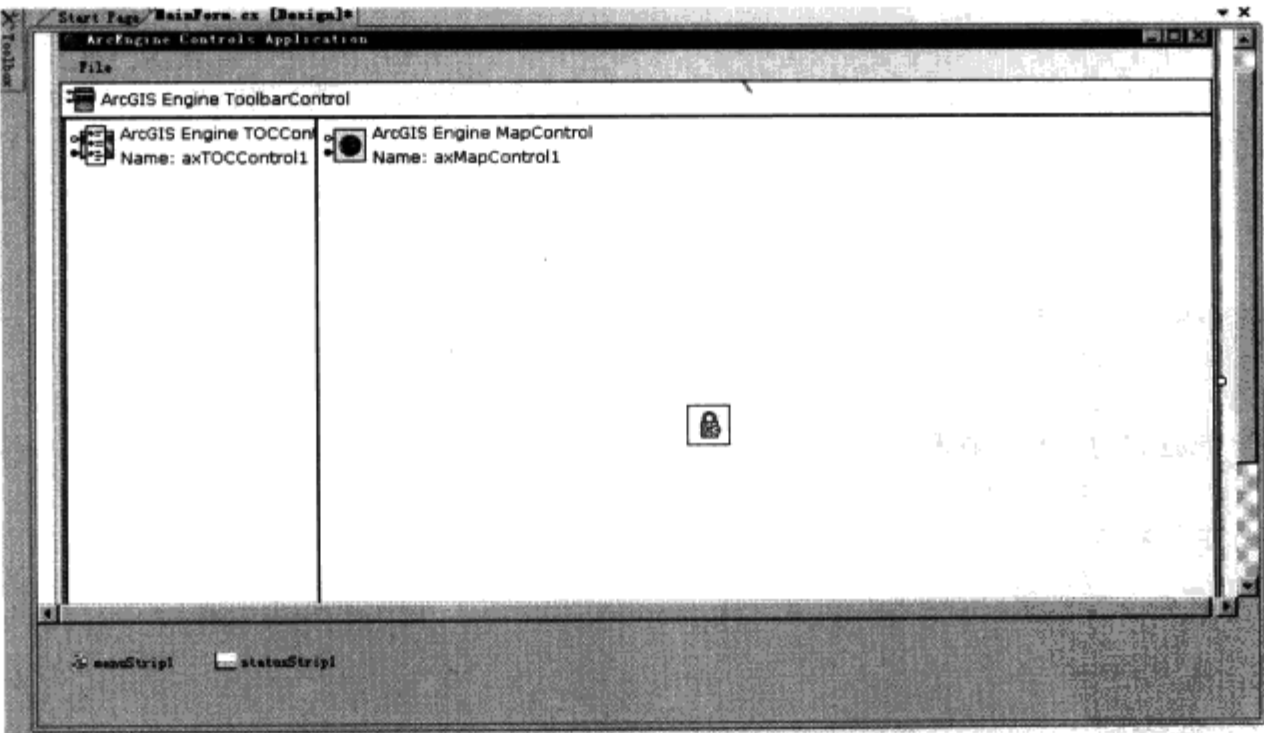


图 2.24 控件布局

2.4.4 ArcGIS Engine 与 ArcGIS Server

ArcGIS Engine 和 ArcGIS Server 的交叉点是它们都是由 ArcObjects 构建，它们底层的 API 相同。因而利用 ArcGIS Engine 开发 COM 组件的方式开发某些特定应用的组件，然后可以供 ArcGIS Server 来调用，或者通过类的方式来集成（注意通过 ArcGIS Engine 提供组件的实现和逻辑应该分开，即 ArcGIS Engine 开发组件不能带有界面，而只能通过 Engine 来实现封装特定功能的组件，因为 IE 和开发 COM 组件的界面无法融合），同时，ArcGIS Engine 提供的组件必须提供符合 ArcGIS Server 要求的统一接口。如果部署在一台服务器上，也可以通过 Web Services 的方式进行集成。另外，ArcGIS Server 可以作为 ArcGIS Engine 的数据源供其调用，下一小节会详述。

虽然它们都是基于 ArcObjects 创建，但是它们针对的开发用户是不同的。开发 Web 应用程序，可以只采用 ArcGIS Server，不需要 ArcGIS Engine；开发桌面应用，一般是采用 ArcGIS Engine。由此可见，采用何种开发架构模式取决于用户需求和具体的应用，两者实现的方式有一定的差别。

2.4.5 ArcGIS Engine 如何调用 ArcGIS Server

ArcGIS Server 的用户可以有很多种，从最普通的 IE 浏览器到大型的 ArcGIS 桌面应用程序都可以。这两种客户端都很强大，能够完成从数据浏览到简单的地图编辑等工作。但其中最强大的仍然是 ArcGIS Desktop 或者使用 ArcGIS Engine 开发的胖客户端。因为这种类型的客户端不但可以跨网络完成瘦客户端能完成的所有功能，更进一步充分施展了 ArcGIS Server 的在线编辑功能，使得 Server 这个平台的功能被利用得更加充分。

ArcGIS Engine 中调用 ArcGIS Server 首先需要连接远程的 Server 地图服务，并获取它所有的 Map Server，然后获取相应的图片，并在界面上显示该图片。

从本质上来讲，ArcGIS Server 通过互联网发布地图服务时实质上是在发送地图服务的当前图片，所以我们只需要准确地获得地图服务此时的图片以及图片的状态信息就可以在 PictureBox 中相应地显示地图服务中的内容了。DrawMap (ref IMapDescription pMapDescription、ref IServerContext pServerContext) 函数就是用来把 ArcGIS Server 的图片转换为普通的图片显示。

主要代码示例 (C#) 如下所示：

```
private void btnConnect_Click(object sender, System.EventArgs e)
{
    this.Cursor = Cursors.WaitCursor;
    try
    {
        //如果连接时用户正在和另外一个 Server 地图服务交互，释放这个 Server Objects
        if (m_pServerContext != null)
            m_pServerContext.ReleaseContext();
        // 连接到 ArcGIS Server
        conn = new ESRI.ArcGIS.Server.WebControls.ServerConnection
            (txtServer.Text, true);
        m_pSOM = conn.ServerObjectManager;
        // 得到 Map Server 的地图服务
        cboMapServer.DataSource = conn.GetServerObjectNames("MapServer");
    }
}
```

```

        cboMapServer.Enabled = true;
        this.Cursor = Cursors.Default;
    }
    catch(Exception exception)
    {
        this.Cursor = Cursors.Default;
        MessageBox.Show(exception.Message, "An error has occurred");
    }
}

//画 ArcGIS Server 图片
private void drawMap(ref IMapDescription pMapDescriptoin, ref
IServerContext pServerContext)
{
    IMapServer pMapServer = pServerContext.ServerObject as IMapServer;

    //生成一个新的 ImageDescription 对象并接受之前的 ImageDisplay 和
//ImageType。这时我们就有了一个大小为 400*500，设备分辨率为 150 的 JPG 文件了
    IImageType it = pServerContext.CreateObject("esriCarto.
ImageType") as IImageType;
    it.Format = esriImageFormat.esriImageJPG;
    it.ReturnType = esriImageReturnType.esriImageReturnMimeData;

    idisp = pServerContext.CreateObject("esriCarto.ImageDisplay")
as IImageDisplay;
    idisp.Height = 400;
    idisp.Width = 500;
    idisp.DeviceResolution = 150;

    IImageDescription pID = pServerContext.CreateObject("esriCarto.
ImageDescription") as IImageDescription;
    pID.Display = idisp;
    pID.Type = it;

    IImageResult pMI = pMapServer.ExportMapImage(pMapDescriptoin,
pID) as IImageResult;
    System.IO.Stream pStream = new System.IO.MemoryStream
((byte[])pMI.MimeData);
    pImage = Image.FromStream(pStream);

    pictureBox1.Image = pImage;
    pictureBox1.Refresh();

    return;
}

```

程序运行结果如图 2.25 所示。

因为 ArcGIS Engine 开发的客户端和瘦客户端最大的区别在于其底层是基于 ArcObjects 构建的,所以客户端不仅能够浏览数据还可以进一步地处理远程客户端上的 ArcGIS Server 地图服务。就像我们在上面所看到的那样,通过连接我们已经成功地获得了 Map Server,那么只要顺着这个

思路进一步地开发下去，就可以获得日常开发中更为常用的地图、图层以及更多的信息。最终，ArcGIS Server 变成了 Engine 客户端强大的远程服务器和数据库。大规模的数据虽然存储在互联网的另外一端，但是在我们的客户端上仍然可以顺畅地进行数据的分析和编辑等高级的地理信息操作。

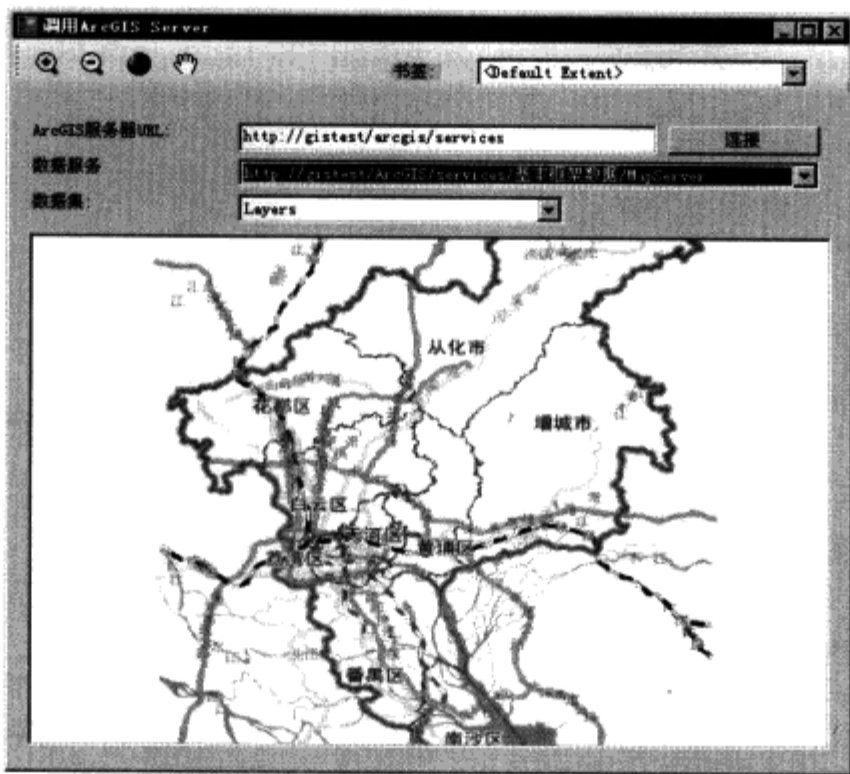


图 2.25 ArcGIS Engine 中调用 ArcGIS Server

2.5 小结

本章讲述了 ArcGIS 软件体系结构，并介绍了所有的 ArcGIS 产品的核心库 ArcObjects，ArcGIS 9 的两个新产品 ArcGIS Engine、ArcGIS Server 都是基于 ArcObjects 组件构建的。ArcGIS Engine 一般是用于局域网环境下开发胖客户端的产品，其开发的是定制的桌面 GIS 产品；而 ArcGIS Server 一般是用于广域网环境下开发瘦客户端的产品，用于开发 WebGIS 应用。阅读完本章，读者应该对 ArcGIS 软件的体系结构有比较清楚的了解，熟悉 ArcGIS Engine 的组成和功能，能够正确地构建 ArcGIS Engine 的开发环境，对于 ArcGIS Engine 与 ArcGIS Server 之间的关系与区别以及如何利用 ArcGIS Engine 调用 ArcGIS Server 的数据服务有基本的认识。

第 3 章

空间数据管理

数据是 GIS 系统的“血液”，数据的好坏直接决定了 GIS 应用系统的成败。本章先简单介绍 ArcSDE 的 Geodatabase 模型概念，分析模型中相关的对象，并解释各对象的功能；然后介绍基于 ArcObjects，通过 ArcSDE 访问空间数据库的例子，相应地给出了上下载栅格数据和矢量数据的实例代码。

3.1 空间数据库模型 Geodatabase

地理空间数据库系统实现的具体方法主要采用的是对现有关系数据库平台的扩展，即内嵌对象-关系模型。它提供对空间数据的支持，基于现有主流数据库系统，对其进行对象-关系模型特征扩展来实现，以 ESRI 公司的 Geodatabase 地理空间数据模型最为典型。

3.1.1 Geodatabase 概念

Geodatabase 是一种采用标准关系数据库系统技术来表现地理信息的数据模型，它支持在标准的数据库管理系统（DBMS）表中存储和管理地理信息。

Geodatabase 支持多种 DBMS 结构和多用户访问，且大小可伸缩。从基于 Microsoft Jet Engine 的小型单用户数据库，到工作组、部门和企业级的多用户数据库，Geodatabase 都支持。目前有两种 Geodatabase 结构：个人 Geodatabase 和多用户 Geodatabase（Multiuser Geodatabase）。

个人 Geodatabase 使用 Microsoft Jet Engine 数据库文件结构，将 GIS 数据存储在小数据库库中，更像基于文件的工作空间，数据库存储量最大为 2GB。个人 Geodatabase 使用微软的 Access 来存储属性表。

对于小型的 GIS 项目和工作组来说，个人 Geodatabase 是非常理想的工具。通常，GIS 用户采用多用户 Geodatabase 来存储和并发访问数据。个人 Geodatabase 支持常用户编辑，不支持版本管理。

多用户 Geodatabase 通过 ArcSDE 支持多种数据库平台，包括 IBM DB2、Informix、Oracle（有或没有 Oracle Spatial 都可以）和 SQL Server。多用户 Geodatabase 使用范围很广，主要用户为工作组、部门和企业。

Geodatabase 是 ArcInfo 8 引入的一个新的基于关系数据库的地理空间数据模型和一体化管理的空间数据。即采用面向对象技术将现实空间地理要素（地物）抽象为由若干对象类组成的数据管理模型，对象类之间建立关联，对矢量、栅格、三维表面、网络、地址等，进行统一的描述。用户可以在已有的空间数据模型之上，建立符合应用需求的扩展模型。在 Geodatabase 模型中，地理空间要素的表达较之以前的模型更接近于现实事物对象的认识和表达方式。Geodatabase 中引入了地理空间要素的行为、规则和关系（分为基本的和特殊的两种行为规则）。

在 Geodatabase 数据库中，Geodatabase 是最高层次的地理数据单元，所有的地理数据由一个或多个 Geodatabase 组成（如图 3.1 所示）。一个 Geodatabase 由多个抽象数据集组成。数据集通过继承得到 4 个可创建对象，即 Tin（Triangulated irregular network）数据集、栅格数据集、要素数据集和表。其中 Tin 数据集、栅格数据集和要素数据集又由抽象地理数据集派生而来，它们可归纳为地理空间数据。表对象相当于地理属性数据；Tin 数据集是指一套带有 z 值的不规则三角网，用它来精确表示表面；栅格数据集是存储有不同光谱或分类值的多光谱带的一个简单数据集或复合数据集；要素数据集是具有相同空间参考系（spatial reference）的要素集合，它由几何网络和对象组成。几何网络同时也称 r 拓扑，被绑定在一个包含有完整拓扑的要素类的图层中。要素类还可派生为点、线、面和注记 4 种要素。数据集的另一类地理数据（属性数据）表通过继承可以得到属性关系类和对象类。属性关系类是一张存储要素与要素之间或对象与对象之间关联的表。对象类则是用于关联行为的表。对象类向下可泛化为要素类，并为要素类制定了相应的约束机制，即规则；对象类同时又与几何网络组合成要素数据集，而且对象又继承自表，这就把地理空间数据和属性数据联系在一起了。

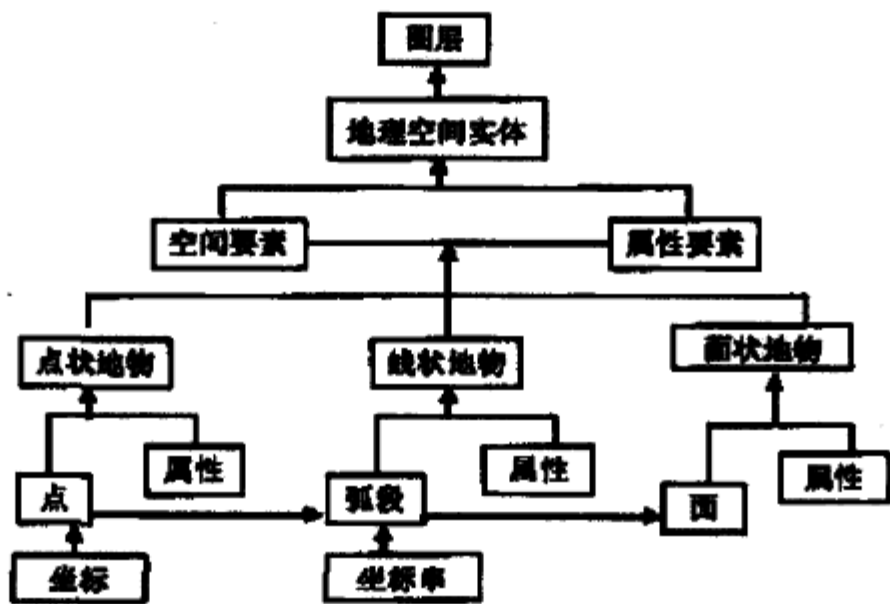


图 3.1 地理空间数据模型 Geodatabase

3.1.2 Geodatabase 模型

Geodatabase 模型的体系结构由要素数据集（feature dataset）、对象类（object class）、要素类（feature class）、关系类（relationship class）、域值（domain）、栅格数据集（格网数据集）（raster dataset）、点要素类（point feature）、线要素类（line feature）、面要素类（polygon feature）、特征

数据集 (feature dataset)、规则 (rules) 等要素组成。

Geodatabase 数据模型的结构体系可以通过 ArcCatalog 视图形式表达, 如图 3.2 所示。

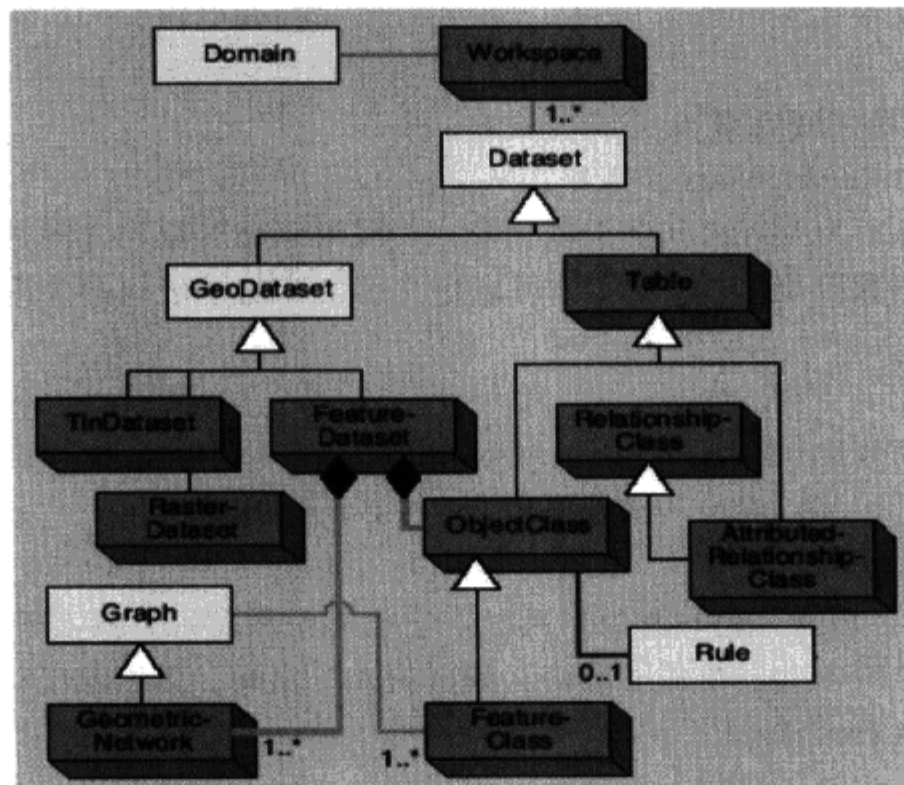


图 3.2 Geodatabase 空间数据模型

下面对 Geodatabase 数据模型中的各对象进行简单的介绍:

(1) 域 (Domain)。

对象属性的有效值集合。可以是文本型的, 也可以是数值型的。

(2) 工作空间 (Workspace)。

表示一个含有数据集的数据库或者数据源, 数据集中可以包括表、要素类以及关系表。

(3) 数据集 (Dataset)。

数据集包含地理空间几何数据和关系表。地理空间几何数据包含栅格数据和矢量数据。

(4) 要素数据集 (Feature Dataset)。

要素数据集是具有相同空间参考系的要素类集合。将不同的要素类放到一个要素数据集下可能有很多原因, 一般而言, 主要有以下 3 种情况。

- 专题归类表示: 当不同的要素类属于同一范畴。例如, 就全国范围内某种比例尺的水系数据而言, 其点、线、面类型的要素类可组织为同一个要素数据集。
- 创建几何网络 (GeometricNetwork): 在同一几何网络中充当连接点和边的各种要素类, 应组织到同一要素数据集中。例如, 在配电网络中, 有各种开关、变压器、电缆等, 它们分别对应点或线类型的要素类, 在配电网络建模时, 我们要将其全部考虑到配电网络对应的几何网络模型中去。
- 考虑平面拓扑 (planar topology): 共享公共几何特征的要素类, 如用地、水系、行政区界等。当移动其中的一个要素时, 其公共的部分也要求一起移动, 并保持这种公共关系不变。

(5) 不规则三角形网络数据集 (TIN Dataset)。

是从表面上采样高程点数据生成的不规则三角形。TIN 可以用于模拟地球表面, 同时也可用于连续性的环境因子的分布研究, 如碳元素的分布。

(6) 栅格数据集 (Raster Dataset)。

可以表现为影响地图、表面、表现某个环境因子采样数据的 Grid，或者是普通的实物照片。有些栅格数据具有多个波段。

(7) 属性表 (Table)。

与空间要素相关联亦或完全无关的关系表。

(8) 关系类 (RelationshipClass)。

定义了对象类、要素类两者之间的关联信息。关联可以是对象类之间的，也可以是要素类之间的或者是要素类和对象类之间的。例如，可以定义房主和房子之间的关系，房子和地块之间的关系等。

(9) 规则 (Rule)。

通过关联类和连通规则，属性验证用以增强数据的完整性。例如，规定不同管径的水管连接，则必须有一个合适的转接头；规定一块地可以有 1~3 个权利人等。

(10) 对象类 (ObjectClass)。

在 Geodatabase 中，对象类是一种特殊的类，它没有空间特征，其实例为可关联某种特定行为的表记录，如某块地的主人。在“某地块”和“主人”之间，可以定义某种关系。

(11) 要素类 (FeatureClass)。

地理要素类是具有相同属性集、相同行为和规则的空间对象的集合，如河流、道路、植被、用地、电缆等。

(12) 几何网络 (Geometric Network)。

用于模拟线形系统，如道路交通网络。支持丰富的网络跟踪和分析功能。

3.2 访问空间数据库

3.2.1 打开数据库工作空间

WorkspaceFactory 对象负责 Workspace 对象的创建分配，并允许客户端通过连接属性设置特定的工作空间 (Workspace，一个工作空间表示一个含有数据集的数据库或者数据源，数据集中可以包括表、要素类以及关系表)。

WorkspaceFactory 有一个连接池，能够维护当前应用程序激活的工作空间，连接属性可以用对象 PropertySet 及保存的连接文件来设置。

WorkspaceFactory 所支持的方法包括：浏览和管理文件数据工作空间及管理远程数据库空间的连接文件。在 Geodatabase 中，工作空间分为以下几个类型。

- EsriFileSystemWorkspace：文件空间，如 shp 工作空间、ArcInfo 工作空间等；
- esriLocalDatabaseWorkspace：本地个人数据库，存储在 Microsoft Access 中；
- esriRemotedatabaseWorkspace：远程地理数据库，存储在大型商业 RDBMS 中，如 Oracle、DB2、SQL Server、Infomix，并通过空间数据库引擎访问数据。

1. 远程数据方式

访问远程数据库主要通过 WorkspaceFactory 的 Open 和 OpenFromFile 两种方法。Open 方法通过输入连接空间数据库的属性来创建连接，通过 ArcSDE 访问空间数据库，一般的连接属性有：

用户名、密码、数据库名、机器名、服务名以及版本信息。数据库名属性是可选的，但在 ArcSDE 管理多个数据时必须要求。版本参数是在多版本数据库中指定当前激活的版本。如果没有版本信息，连接将返回一个默认版本。连接创建成功后，可以通过工作空间对象对在其内的数据集、表等进行操作。

下面是通过 ArcSDE 访问 Oracle 空间数据库的实例代码。

```
public IWorkspace IWorkspaceFactory_Open_Example(String server, String instance,
String user, String password, String database, String version)
{
    ESRI.ArcGIS.esriSystem.IPropertySet propertySet = new ESRI.ArcGIS.
esriSystem.PropertySetClass();
    propertySet.SetProperty("SERVER", server);           //服务器名
    propertySet.SetProperty("INSTANCE", instance);       //实例名
    propertySet.SetProperty("DATABASE", database);       //数据库名
    propertySet.SetProperty("USER", user);               //用户名
    propertySet.SetProperty("PASSWORD", password);      //密码
    propertySet.SetProperty("VERSION", version);         //版本
    IWorkspaceFactory workspaceFactory = new ESRI.ArcGIS.DataSourcesGDB.
SdeWorkspaceFactoryClass();
    return workspaceFactory.Open(propertySet, 0);
}
```

2. 个人数据库方式

打开个人数据库，基本与远程访问数据库相同，通过 WorkspaceFactory 对象所提供的 OpenFromFile 方法打开工作空间。在创建连接时，没有复杂的参数设置，只要将本地的*.mdb 文件全路径作为参数传给 OpenFreomFile 即可。

```
//e.g., nameOfFile = "E:\\data\\english\\FileGDB\\canada\\canada.gdb"
public IWorkspace IWorkspaceFactory_OpenFromFile_Example_FileGDB(string
nameOfFile)
{
    IWorkspaceFactory workspaceFactory = new ESRI.ArcGIS.DataSourcesGDB.
FileGDBWorkspaceFactoryClass();
    return workspaceFactory.OpenFromFile(nameOfFile, 0);
}
```

3.2.2 通过 NAME 对象方式

一个 NAME 对象支持 Open 方法，允许客户端通过相应的 NAME 对象获得一个真正对象的实例（如数据集、工作空间）。NAME 对象起着绑定对象的作用。

在 Geodatabase 中具有管理 NAME 对象的接口方法，这些方法可以在浏览客户端显示工作空间中的数据集或实例化某个给定的数据集。NAME 对象还带有描述对象命名的属性信息，客户端可以利用这些信息显示额外的信息。NAME 对象提供了访问元数据，改变具体对象许可的接口。在某种意义上，当需要获得对象的更多属性信息或调用对象的额外方法时，NAME 对象是实际对象的一个轻量级代理。

NAME 对象能够用来定义一个数据集，例如，通过空间处理操作，数据集被创建。NAME

对象具有很多类型、工作空间、表、要素类、特征数据集、栅格（Raster）以及关系类。

工作空间对象是任何在工作空间中的数据集 NAME 对象的重要组成部分。通过接口 `IWorkspaceName` 可以访问工作空间名称对象。

为了创建一个新的工作空间名称对象，必须通过全路径或者路径参数来设置 `WorkspaceFactoryProgID`，下面是通过 NAME 对象访问个人数据库的程序片段。

```
public IWorkspace AddAccessDBByName()
{
    IWorkspaceName pWorkspaceName = new WorkspaceNameClass();
    pWorkspaceName.WorkspaceFactoryProgID = "esriDataSourcesGDB.
AccessWorkspaceFactory";
    pWorkspaceName.PathName = @"D:\test\Ao\data\sh\MapData.mdb";
    IName n = pWorkspaceName as IName;
    IFeatureWorkspace Workspace = n.Open();

    return Workspace;
}
```

所有的工作空间工厂都在 `esriDataSources` 中，在上面的代码中，NAME 对象能够指向一个已经存在的或者创建的工作空间对象。如果工作空间对象已经存在，可以通过 `IName: Open` 接口有效地打开，这个程序与通过 `IWorkspaceFactory` 的 `Open` 或者 `OpenFromFile` 打开工作空间相当。如果工作空间对象不存在，可用 `IWorkspaceFactory: create` 创建。

在某些环境下，可能有工作空间对象，但需要工作空间名称对象。示例代码如下：

```
public IWorkspace CreateAccessWorkspace()
{
    // 实例化一个 Access 数据，并创建一个连接

    IWorkspaceFactory workspaceFactory = new AccessWorkspaceFactoryClass();
    IWorkspaceName workspaceName = workspaceFactory.Create("C:\\temp\\",
        "Sample.mdb", null, 0);

    // Cast the workspace name object to the IName interface and open the workspace.
    IName name = (IName)workspaceName;
    IWorkspace workspace = (IWorkspace)name.Open();
    return workspace;
}
```

3.2.3 获得工作空间实际元素

在 Geodatabase 中，一个工作空间对象即可认为是一个数据库。在一个工作空间中可以包含多个数据集、要素类、表等。根据上述方法打开工作空间数据库后，返回一个工作空间对象。

工作空间对象通过两种方法获得当前工作空间中所包含的数据集：`get_DatasetNames` 和 `get_Datasets`。其函数的基本的形式如下：

```
public IEnumDatasetName get_DatasetNames(
    esriDatasetType DatasetType
```

```

    );
    public IEnumDataset get_Datasets(
        esriDatasetType DatasetType
    );

```

由上可以看出，两种方法基本相似。第一种方法所获得的是 Name 对象的枚举对象，而第二种方法所获得的是实际的数据集对象。第一个参数 `esriDatasetType` 为输入参数，可以用来过滤感兴趣的数据集对象。第二个参数为输出参数，返回工作空间中所指定类型的枚举集。如果参数 `esriDatasetType` 设置为 `esriDTAny`，将返回工作空间中的所有数据集或者数据集名称，但是只有顶层的数据集才能被返回。当一个工作空间既包含独立的要素类又包含数据集中的要素类，通过调用 `get_datasets` 或者 `get_datasetnames` 设置数据集类型为 `esriDTFeatureClass` 时，返回的只是独立要素类。

下面是通过 `Get_Datasets` 方法获得的工作空间中所有数据集的示例代码。

```

public void IWorkspace_DatasetNames_Example(IWorkspace workspace)
{
    //这个函数列出工作空间下的数据表名

    //所有的名字都打在屏幕
    IEnumDatasetName enumDatasetName = workspace.get_DatasetNames
(esriDatasetType.esriDTFeatureDataset);
    IDatasetName datasetName = enumDatasetName.Next();
    while (datasetName != null)
    {
        Console.WriteLine(datasetName.Name);
        datasetName = enumDatasetName.Next();
    }
}

public void DisplayDatasetName(IWorkspace workspace)
{
    // 得到工作空间下的表集合
    IEnumDataset enumDataset = workspace.get_Datasets(esriDatasetType.
esriDTAny);
    // 列出表的名字
    IDataset dataset = null;
    while ((dataset = enumDataset.Next()) != null)
    {
        Console.WriteLine(dataset.Name);
    }
}

```

3.3 矢量数据

矢量数据是一种最常见的图形数据结构，地理实体用一系列 x 、 y 坐标来确定它们的位置，即通过记录坐标的方式，尽可能地将点、线、面等地理实体表现得精确无误。该数据结构常用于描述线状分布的地理要素，如河流、道路、等值线等。

任何点、线、面实体都可以用直角坐标点 x 、 y 来表示，这里， x 、 y 可以对应于地面坐标经度和纬度，也可以对应于数字化时所建立的平面坐标 x 和 y 。点可以被表示成一组坐标 (x,y) ，对于线和面，则均被表示为多组 $(x_1,y_1; x_2,y_2; \cdots; x_n, y_n)$ ，由于表示面的多边形由首尾相连的线所组成，所以其起、止点的坐标相同。这些由 x 、 y 坐标表示的点都是由光滑的曲线间隔采样得到的，同样的一条曲线，取的点越多，则以后恢复时就越接近原来的曲线，失真越少，但数据量将增加；反之，取的点太少，恢复时就可能成为折线，出现失真的情况。这种用 x 、 y 坐标表示的矢量编码方法文件结构简单，易于实现以多边形为单位的运算和显示。矢量数据结构如图 3.3 和图 3.4 所示。

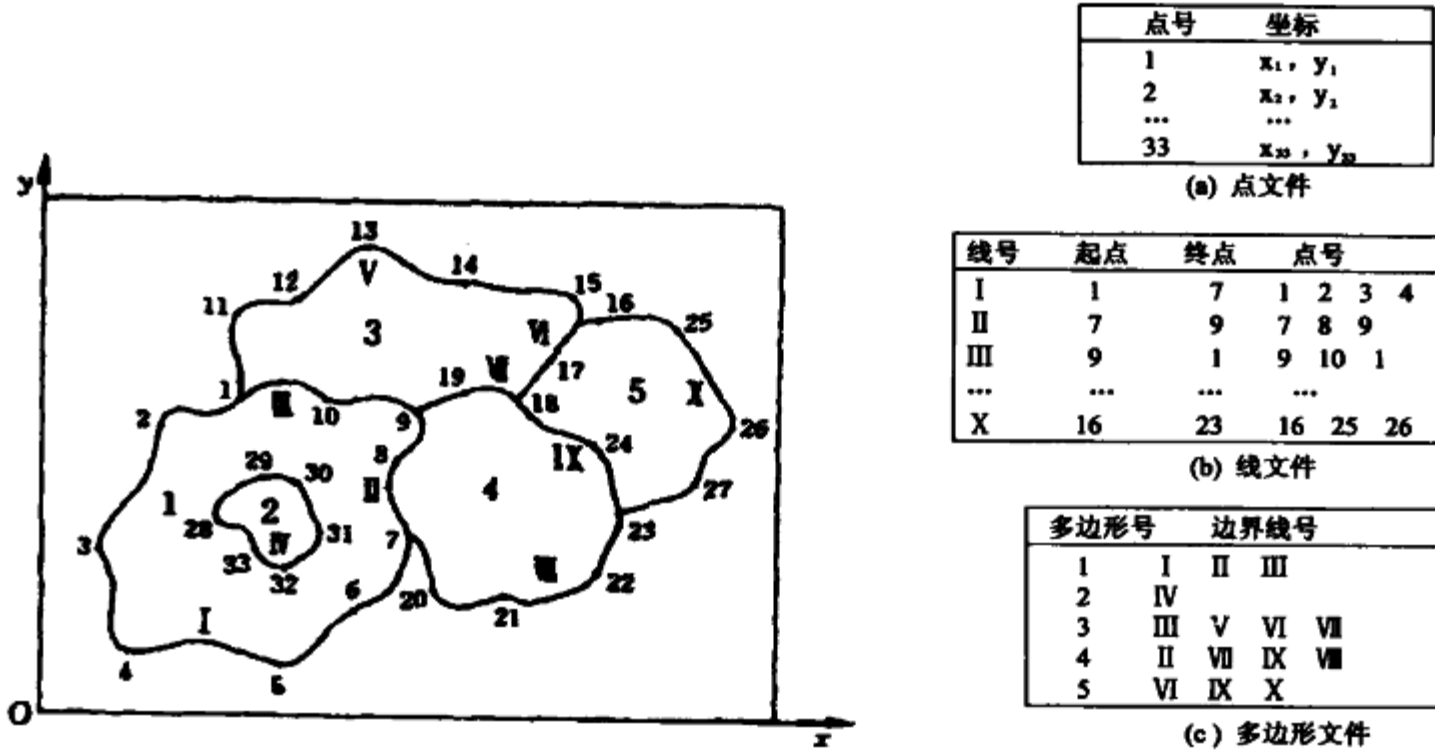


图 3.3 多边形矢量编码

图 3.4 树状索引的文件结构

3.3.1 文件数据导入 Geodatabase

矢量数据文件入库是数据管理模块必需的功能，ArcObjects 也提供了相应的接口 IFeatureDataConverter。该接口的功能非常强大，无论是 Geodatabase 中的要素类还是本地系统中的要素类都得到了很好的支持。IFeatureDataConverter 能够将已有的表、要素类、特征数据集复制或转换到另一个地址或数据中。但此接口对 ArcInfo Convergence 的转换尚不支持。IFeatureDataConverter 对大数据的上载非常有效。一些对象或接口的 IFeatureDataConverter 非常有用，可实现下列功能：

- 用 IFieldChecher 检查字段名；
- 用 IEnumInvalidObject 检查转换过程中的不合法数据；
- 用 IFeatureProgress 保存最后用户的信息。

接口 IFeatureDataConverter 提供了两种不同格式的数据集与数据集，要素类与要素类之间的转换方法。该接口支持数据文件导入 Geodatabase，具体步骤如下：

- (1) 打开 Geodatabase 工作空间；
- (2) 打开本地文件数据的工作空间获得数据集或要素类；
- (3) 在 Geodatabase 指定相应的数据名或要素类名；

(4) 调用 IFeatureDataConverter 转换函数。

下面的代码是调用 IFeatureDataConverter 所提供的方法, 实现 SHP 数据文件导入 Geodatabase, 创建相应的要素类, 并录入数据, 示例程序如下:

```
//例如: nameOfSourceFeatureClass = "ctgFeatureshp.shp"
//      nameOfTargetFeatureClass = "ctgFeature"
public void IFeatureDataConverter_ConvertFeatureClass_Example(IWorkspace
sourceWorkspace, IWorkspace targetWorkspace, string nameOfSourceFeatureClass, string
nameOfTargetFeatureClass)
{
    //创建源工作空间
    IDataset sourceWorkspaceDataset = (IDataset)sourceWorkspace;
    IWorkspaceName sourceWorkspaceName = (IWorkspaceName)sourceWorkspaceDataset.
FullName;
    //创建源数据集
    IFeatureClassName sourceFeatureClassName = new FeatureClassNameClass();
    IDatasetName sourceDatasetName = (IDatasetName)sourceFeatureClassName;
    sourceDatasetName.WorkspaceName = sourceWorkspaceName;
    sourceDatasetName.Name = nameOfSourceFeatureClass;
    //创建目标工作空间
    IDataset targetWorkspaceDataset = (IDataset)targetWorkspace;
    IWorkspaceName targetWorkspaceName = (IWorkspaceName)targetWorkspaceDataset.
FullName;
    //创建目标数据集
    IFeatureClassName targetFeatureClassName = new FeatureClassNameClass();
    IDatasetName targetDatasetName = (IDatasetName)targetFeatureClassName;
    targetDatasetName.WorkspaceName = targetWorkspaceName;
    targetDatasetName.Name = nameOfTargetFeatureClass;
    //源数据集的字段集
    ESRI.ArcGIS.esriSystem.IName sourceName = (ESRI.ArcGIS.esriSystem.IName)
sourceFeatureClassName;
    IFeatureClass sourceFeatureClass = (IFeatureClass)sourceName.Open();
    //验证字段
    IFieldChecker fieldChecker = new FieldCheckerClass();
    IFields targetFeatureClassFields;
    IFields sourceFeatureClassFields = sourceFeatureClass.Fields;
    IEnumFieldError enumFieldError;
    // 设置验证的对象
    fieldChecker.InputWorkspace = sourceWorkspace;
    fieldChecker.ValidateWorkspace = targetWorkspace;
    fieldChecker.Validate(sourceFeatureClassFields, out enumFieldError, out
targetFeatureClassFields);
    //找到空间对象字段
    IField geometryField;
    for (int i = 0; i < targetFeatureClassFields.FieldCount; i++)
    {
        if (targetFeatureClassFields.get_Field(i).Type == esriFieldType.
esriFieldTypeGeometry)
        {
            geometryField = targetFeatureClassFields.get_Field(i);
```

```

        // 得到空间字段的定义
        IGeometryDef geometryDef = geometryField.GeometryDef;
        //得到空间字段的索引
        IGeometryDefEdit targetFCGeoDefEdit = (IGeometryDefEdit)geometryDef;
        targetFCGeoDefEdit.GridCount_2 = 1;
        targetFCGeoDefEdit.set_GridSize(0, 0); //为导入的数据设置合法的索引大小
        targetFCGeoDefEdit.SpatialReference_2 = geometryField.GeometryDef.
SpatialReference;
        // 开始导入
        IQueryFilter queryFilter = new QueryFilterClass();
        queryFilter.WhereClause = "";
        // 导入所有的输入对象
        IFeatureDataConverter fctofc = new FeatureDataConverterClass();
        IEnumInvalidObject enumErrors = fctofc.ConvertFeatureClass
(sourceFeatureClassName, queryFilter, null, targetFeatureClassName, geometryDef,
targetFeatureClassFields, "", 1000, 0);
        break;
    }
}
}

```

当然，数据入库还有其它方法。例如，用户可以先在 Geodatabase 中创建数据集或要素类，然后再加入数据。由于篇幅限制，在此不作详细讲述。从空间数据库中转出数据文件时，其实也是文件数据导入的过程，也可以利用 IFeatureDataConverter 的接口来实现。

3.3.2 从 Geodatabase 复制特征数据集到个人数据库

熟悉 ArcCatalog 的用户都知道，ArcCatalog 提供了非常方便的数据操作接口，如空间数据集的复制功能，使用相当方便。该功能不仅可以在 SDE 数据库与 SDE 数据库之间进行数据的复制，而且也可以在 SDE 数据库与个人数据库之间进行数据的复制。此功能可以通过 IFeatureDataConverter: ConvertFeatureDataset 来实现。

下面的例子就是在一个新创建的数据库中从 Geodatabase 复制一特征数据集。

```

//IFeatureDataConverter ConvertFeatureDataset Example
//e.g.,元数据集 nameOfSourceFeatureDataset = "USA"
//目标数据集 nameOfTargetFeatureDataset = "USA"
public void IFeatureDataConverter_ConvertFeatureDataset_Example(IWorkspace
sourceWorkspace, IWorkspace targetWorkspace, string nameOfSourceFeatureDataset,
string nameOfTargetFeatureDataset)
{
    //创建源工作空间
    IDataset sourceWorkspaceDataset = (IDataset)sourceWorkspace;
    IWorkspaceName sourceWorkspaceName = (IWorkspaceName)sourceWorkspaceDataset.
FullName;
    //创建源数据集
    IFeatureDatasetName sourceFeatureDatasetName = new FeatureDatasetName
Class();
    IDatasetName sourceDatasetName = (IDatasetName)sourceFeatureDatasetName;
    sourceDatasetName.WorkspaceName = sourceWorkspaceName;
}

```

```

        sourceDatasetName.Name = nameOfSourceFeatureDataset;
        //创建目标工作空间
        IDataset targetWorkspaceDataset = (IDataset)targetWorkspace;
        IWorkspaceName targetWorkspaceName = (IWorkspaceName)targetWorkspaceDataset.
FullName;
        //创建目标数据集
        IFeatureDatasetName targetFeatureDatasetName = new FeatureDatasetName
Class();
        IDatasetName targetDatasetName = (IDatasetName)targetFeatureDatasetName;
        targetDatasetName.WorkspaceName = targetWorkspaceName;
        targetDatasetName.Name = nameOfTargetFeatureDataset;
        //开始复制
        IFeatureDataConverter featureDataConverter = new FeatureDataConverter
Class();
        featureDataConverter.ConvertFeatureDataset(sourceFeatureDatasetName,
targetFeatureDatasetName, null, "", 1000, 0);
        Console.WriteLine("Conversion Complete");
    }

```

3.3.3 编辑 Geodatabase 中的数据

应用程序用事务来管理数据库中数据的直接更新,并且负责在事务边界处取消所有的缓冲对象。ITransactions 是一个可选的接口,它允许应用程序明确地控制数据库的事务,但不支持内嵌事务。在一个编辑会话中进行更新操作,应用程序不必用事务。根据编辑会话的上下文环境,事务将会在需要的时候自动开启或者停止。

应用程序在使用 ArcObjects 中访问空间数据库的对象进行 DDL 操作时,事务能够确保数据字典表的完整性,并在操作结束后提交事务。

在一个事务中,应用程序不必激活 DDL 操作,但必须限制 DML 操作。应用程序能够利用 Geodatabase 编辑会话来管理长事务。当 Geodatabase 中对象进行更新时,即 IWorkspaceEdit 能够使应用程序开始或停止,编辑会话和长事务一致。当前应用程序的改变会立即被编辑会话发现,但其他应用程序所作的修改直到编辑会话保存或者取消时才发现。如果已经用 ESRI 编辑工具开始编辑工作空间,就不能用这个接口控制编辑。

调用 StartEditing 方法可以开始一个编辑会话,参数 withUndoRedo 能够使得工作空间支持 Undo/Redo (如 SHP 文件)。

方法 startEditOperation 开始一个编辑操作,编辑操作可以认为是一个短事务嵌套相应的长事务编辑会话。如果执行编辑操作失败,应用程序必须调用 AbortEditOperation 方法放弃所做的编辑。调用 StopEditOperation 方法标志成功完成编辑,成功的编辑可以认为被压入 Undo 栈。

方法 UndoEditOperation 能够将编辑会话回滚到执行编辑最顶层 Undo 栈的状态。取消编辑将从 Undo 栈中弹出,并压入 Redo 栈中。RedoEditOperation 能够回滚到执行编辑的顶层 Redo 栈的会话状态。从 Redo 栈中弹出 Redo 操作,并把它压入 Undo 栈中。如果开始新的编辑操作时清除 Redo 栈中的内容。

方法 StopEditing 用来结束编辑会话。参数 sveEdits 控制所做的编辑是否被保存或取消。一个多版本的数据库可以支持多个编辑会话在同一个数据库版本上进行操作。在这种情况下,如果数据库的状态与开始的版本不一致,调用 StopEditing 将返回错误代码 FDO_E_VERSION_

REDEFINED。应用程序可以调用 `IVersionEdit: Reconcile` 方法协调版本间冲突，协调后再调用 `StopEditing`。

Geodatabase 保证编辑会话中的行对象具有“唯一实例”。如果在应用程序中对象已经被实例化，那么数据访问时，检索 Object ID 并在内存中返回对象实例。当在更新复杂的对象模型时，这能保证应用程序正确。例如，基于关系的或网络特征的模型更新几何对象将影响整个数据的几何拓扑关系。

下面的代码是关于简单编辑会话的示例。注意如果用户在进行编辑并选择 Undo 操作时，将没有快速的保存出现，因为在编辑会话中会对所有编辑过的对象进行处理。在编辑会话外，Geodatabase 的数据访问 APIS (`IRow: Store`, `ITable: Update`, `ITable: Insert`) 将会失去作用。要素类也要求编辑会话来确保唯一实例。

```
//IWorkspaceEdit Example
//e.g., nameOfFeatureClass = "States"
//on ArcSDE 用完整的用户名
public void IWorkspaceEdit_Example(IWorkspace workspace, string
nameOfFeatureClass)
{
    IFeatureWorkspace featureWorkspace = (IFeatureWorkspace)workspace;
    IFeatureClass featureClass = featureWorkspace.OpenFeatureClass
(nameOfFeatureClass);
    IWorkspaceEdit workspaceEdit = (IWorkspaceEdit)workspace; //开始拥有回滚的编辑
    workspaceEdit.StartEditing(true);
    workspaceEdit.StartEditOperation();
    IFeature feature = featureClass.GetFeature(1);
    feature.Delete();
    workspaceEdit.StopEditOperation();
    Console.WriteLine("你要回滚你刚才的操作么? Y or N");
    string response = Console.ReadLine();
    if(response.ToUpper() == "Y")
    {
        workspaceEdit.UndoEditOperation();
    }
    bool hasEdits = false;
    workspaceEdit.HasEdits(ref hasEdits);
    if (hasEdits)
    {
        Console.WriteLine("是否保存编辑? Y or N");
        response = Console.ReadLine();
        if (response.ToUpper() == "Y")
        {
            workspaceEdit.StopEditing(true);
        }
        else
        {
            workspaceEdit.StopEditing(false);
        }
    }
}
```


3.3.4 空间数据拓扑检查

拓扑是同一数据集中的简单要素类的集合，这些简单要素类参与给定的拓扑关系。在同一个拓扑类中可以有多要素类。一个特征数据集可以有多个拓扑关系，但是一个要素类必须属于一个拓扑关系。每个拓扑类相应地有一个拓扑图，这个拓扑图是参与拓扑关系的要素类几何对象的平面表示。

当新的要素被创建、编辑、删除时，拓扑类将创建或修改脏数据区域（Dirty Area，包括要素的外接矩形区域）。脏数据区域是特殊的要素类型，在这个区域下表示不能识别的拓扑状态。被脏数据区域覆盖的要素能够进行编辑或查询，但是不能保证它们的拓扑关系正确。为了发现拓扑类中的隐含要素，脏数据必须通过验证。

拓扑对象是不可以创建的，拓扑对象必须通过调用 `ITopologyContainer: CreateTopology` 接口创建。

方法 `AddClass` 能够在拓扑关系中增加要素类，并指定相应的权重和等级。非简单要素类，如标记，带有度量值以及几何网络关系的要素类，不能加入；对象类、表以及注册为版本的要素类也不能加入。在一个要素类加入已经存在的关系类后，整个或部分改变拓扑状态，并创建与要素范围相一致的脏数据区域。已经在 SDE 中注册为版本的拓扑类不能调用 `AddClass` 方法，但是未注册的能够调用。

- 拓扑关系支持 `IFeatureClassContainer` 接口，这个接口能够返回参与拓扑关系的要素类。
- `Cache` 返回拓扑图的一个引用。拓扑图能够和单独拓扑（如边、节点）一同起作用。
- `ClusterTolerance` 返回所创建的拓扑类的容差值。这个容差值不能被改变，如果想修改这个值，必须删除拓扑并重建。
- `DirtyArea` 方法返回脏数据区域的多边形。`DirtyArea` 方法需要有 `IPolygon` 对象作为输入参数，这个多边形对象必须与拓扑范围一致或是这个范围的子集。如果在给定的区域里没有脏数据区域，一个空的多边形对象将会返回。这个返回的多边形对象可能是多边形。

每个拓扑类都有一个最大错误数，可以通过 `MaximumGenerateErrorCount` 设定其值，默认情况下为 -1，表示为不限制最大数目。`ValidateTopology` 函数验证给定范围内的拓扑类的脏数据。`ValidateTopology` 将根据设定的拓扑关系验证整个区域，并返回验证区域的外接矩形。`ValidateTopology` 能够在会话外起作用，但是如果在 Geodatabase 中拓扑类注册为版本后，必须在一个编辑会话中才可以。

接口 `ITopologyRuleContainer` 为拓扑关系提供了增加、删除和返回拓扑关系的接口，这个接口还提供了控制拓扑错误和异常表示的接口。函数 `CanAddRule` 返回 `Bool` 值，验证是否这个拓扑关系已经存在。当某个拓扑关系已经存在，或目前所拥有的关系能够等价所添加的关系，已经指定的拓扑规则已被定义为子类型时，这个函数将会返回 `False`。`AddRule` 方法可以在拓扑关系中增加新的规则，这个新增加的规则将导致该拓扑范围内的脏数据区域重新创建，并改变拓扑状态。

`ITopologyErrorFeature` 接口提供只读的拓扑错误要素，平台错误的要素表示违反拓扑规则，这些错误的要素在进行拓扑验证时被发现。错误的要素不能直接编辑，因此当通过接口调用 `IFeature` 时，调用 `IFeature:: Value` 或者 `IFeature:: Store` 将会失败。唯一能够修改的拓扑错误要素是通过 `ITopologyRuleContainer:: PromoteToRuleException` 标志异常。相反地异常可以作为一个参数传给 `ITopologyRuleContainer:: DemoteFromRuleException`。`OriginID` 和 `DestinationID` 即在源要素类和目标要素类中有错误的要素 ID。一般情况下，只返回源要素类中有拓扑错误的要

素 ID, 目标要素的 ID 为 0 值。根据 esriTRTNnGaps 规则产生的拓扑错误, 无论 OriginID 和 DestinationID 都返回 0 值。另外, 在源要素类和目标要素类中定义的拓扑规则一般为目标要素类返回 0 值。常用的拓扑规则有:

- EsriTRTAreaNooverlap;
- EsriTRTAreaNooverlapArea;
- EsriTRTLineNoOverlap;
- EsriTRTLineNoIntersection;
- EsriTRTLineNoOverlapLine;
- EsriTRTLineNoIntersectOrInteriorTouch。

拓扑错误要素 ErrorID 对于拓扑关系中所有的特征要素类来说是不唯一的, 但是对具体某一几何要素类型的要素类是唯一的。面状要素类的 ErrorID 可能与点状要素类中的 ErrorID 相同, 但是在面状要素类中, 其值是唯一的。结合 ErrorID 和几何类型, 可以在错误要素类中确定唯一值。

```
public void esriGeoDatabase__IErrorFeatureContainer_ErrorFeatures(ITopology
topology, IWorkspaceEdit workspaceEdit)
{
    IGeoDataset geoDataset = (IGeoDataset)topology;
    IFeatureClassContainer featureclassContainer = (IFeatureClassContainer)
topology;
    ITopologyRuleContainer topologyruleContainer = (ITopologyRuleContainer)
topology;
    // 得到规则 LineNoOverlap Rule
    IEnumRule enumRule = topologyruleContainer.Rules;
    enumRule.Reset();
    ITopologyRule topologyRule = (ITopologyRule)enumRule.Next();
    while (topologyRule.TopologyRuleType != esriTopologyRuleType.
esriTRTLineNoOverlap)
    {
        topologyRule = (ITopologyRule)enumRule.Next();
    }
    IErrorFeatureContainer errorfeatureContainer = (IErrorFeatureContainer)
topology;
    IEnumTopologyErrorFeature enumerrorfeatureContainer;
    ITopologyErrorFeature topologyerrorFeature;
    string strOIDs;
    //得到错误的几何对象
    enumerrorfeatureContainer = errorfeatureContainer.get_ErrorFeatures
(geoDataset.SpatialReference, topologyRule, geoDataset.Extent, true, false);
    // 得到 OID
    topologyerrorFeature = enumerrorfeatureContainer.Next();
    strOIDs = topologyerrorFeature.DestinationOID.ToString();
    do
    {
        topologyerrorFeature = enumerrorfeatureContainer.Next();
        strOIDs = strOIDs + ", " + topologyerrorFeature.DestinationOID;
    } while (topologyerrorFeature != null);
    // 打开目标图层
    IFeatureClass featureClass = featureclassContainer.get_ClassByID
```

```
(topologyRule.DestinationClassID);
    ITable table = (ITable)featureclassContainer;
    //查询
    IQueryFilter queryFilter = new QueryFilterClass();
    queryFilter.WhereClause = "OBJECTID in (" + strOIDs + ")";
    IFeatureCursor featureCursor = featureClass.Update(queryFilter, false);
    // 删除错误对象
    workspaceEdit.StartEditing(true);
    workspaceEdit.StartEditOperation();
    table.DeleteSearchedRows(queryFilter);
    workspaceEdit.StopEditOperation();
    workspaceEdit.StopEditing(true);
}
```

3.4 栅格数据

栅格数据结构由像元阵列构成，每个像元用网格单元的行和列来确定它的位置，常用于表示地质、气候、土地利用或地形等面状要素。任何面状的对象，如土地利用、土壤类型、地势起伏、环境污染等，都可以用栅格数据来表示。栅格数据（如卫星图片、扫描图和航片等）能够表达各种地表类型，如高程、植被等。在 ArcObjects 中，栅格子模块主要包括访问和处理栅格的对象，如 Rasters、RasterDatasets 和 Raster Catalogs。这些对象不仅可以访问基于文件的栅格数据，也可以访问存储在 Geodatabase 中的栅格数据，如图 3.5 所示。

ArcGIS 所支持的栅格数据有：GRID、TIFF、ERDAS IMANE、JPEG 等，还支持企业数据库和本地数据库。不管是何种数据源，表达栅格数据有两种方式：一种是栅格数据集（Raster Dataset），另一种是栅格列表（Raster Catalog）。栅格数据集方式表示一个已经存在的数据集，而不管这个数据是以某种格式存储在硬盘上的栅格文件或在数据库中的数据。栅格列表方式是 Geodatabase 中特殊的要素类，是将多个数据集的集合作为同一实体来管理。一个数据集中可以有一个或多个波段，可以包含有金字塔、统计信息或调色板，用这些参数可以实现数据的快速调度与显示。

3.4.1 打开栅格工作空间

就栅格数据而言，基于文件的工作空间，基于 Access 的个人数据库，基于企业数据库的数据库工作空间都是工作空间。在访问栅格数据时，必须打开一个工作空间。工作空间是不能被创建的，必须通过工作空间工厂来初始化，初始化时采用 RasterWorkspaceFactory，访问 Access 工作空间采用 AccessWorkspaceFactory，访问数据库工作空间用 SdeWorkspaceFactory。

下面的代码表示打开一个给定文件夹的栅格工作空间。

```
IRasterWorkspace OpenRasterWorkspace(string filePath)
{
    IWorkspaceFactory wsFactory = new RasterWorkspaceFactoryClass();
    IRasterWorkspace rasterWS = (IRasterWorkspace)wsFactory.OpenFromFile
```

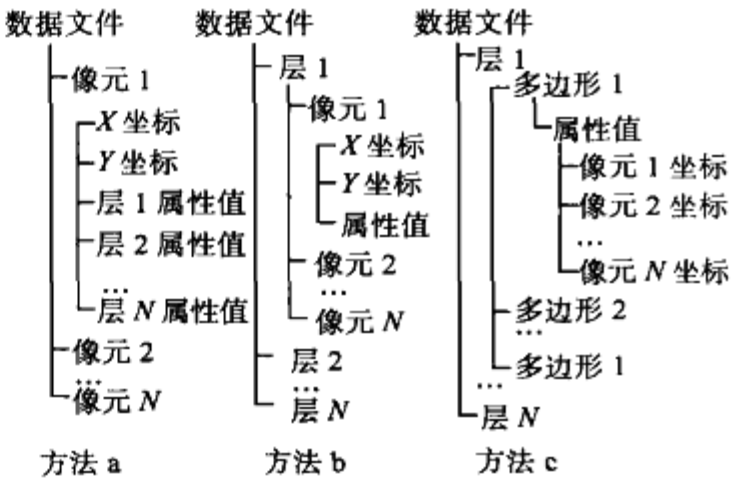


图 3.5 栅格数据集方式

```
(filePath, 0);  
    return rasterWS;  
}
```

下面的代码表示打开一个个人数据库栅格工作空间：

```
IRasterWorkspaceEx OpenFileGDBWorkspace(string filePath)  
{  
    IWorkspaceFactory wsFactory = new FileGDBWorkspaceFactoryClass();  
    IRasterWorkspaceEx rasterWS = (IRasterWorkspaceEx)wsFactory.OpenFromFile(  
        filePath, 0);  
    return rasterWS;  
}
```

下面是通过 SDE 打开数据库栅格工作空间的示例程序：

```
IRasterWorkspaceEx OpenSDEWorkspace(string sServer, String sInstance, String sDB,  
    String sUser, String sPasswd, String sVersion)  
{  
    IWorkspaceFactory wsFactory = new SdeWorkspaceFactoryClass();  
    IPropertySet sdeConnection = new PropertySetClass();  
    sdeConnection.SetProperty("Server", sServer);  
    sdeConnection.SetProperty("Instance", sInstance);  
    sdeConnection.SetProperty("Database", sDB);  
    sdeConnection.SetProperty("User", sUser);  
    sdeConnection.SetProperty("Password", sPasswd);  
    sdeConnection.SetProperty("Version", sVersion);  
    IRasterWorkspaceEx ws = (IRasterWorkspaceEx)wsFactory.Open(sdeConnection, 0);  
    return ws;  
}
```

3.4.2 获得栅格数据集

获得栅格数据集主要有两种途径：一种是通过接口调用已有的数据集，一种是创建新的栅格数据集。上述 3 个工作空间对象都提供了访问栅格数据和栅格列表。接口 `IRasterWorkspace` 和 `IRasterWorkspace2` 能够打开和创建基于文件的栅格数据集。下面的代码就是打开*.img 文件的片段程序。

```
public IRasterDataset OpenRasterDataset(string path, string name)  
{  
    //this example opens a raster from a raster format  
    try  
    {  
        IWorkspaceFactory wsFactory = new RasterWorkspaceFactoryClass();  
        IRasterWorkspace ws = (IRasterWorkspace)wsFactory.OpenFromFile(path, 0);  
        IRasterDataset rasterDataset = ws.OpenRasterDataset(name);  
        return rasterDataset;  
    }  
    catch (Exception ex)
```



```

    {
        System.Diagnostics.Debug.WriteLine(ex.Message);
        return null;
    }
}

```

另外，还可以创建一个新的基于文件的栅格数据集。但是除其他必须设置的参数外，数据集的原点和宽高必须设定。栅格数据集在创建时将会设定维数以及默认的像素值，这个像素值可以通过像素块进行修改。

接口 `IRasterWorkspaceEx` 支持个人数据库和企业数据库的栅格数据集访问。下面的代码表示打开数据中的栅格数据集。

```

static IRasterDataset OpenGDBRasterDataset(IRasterWorkspaceEx rasterWorkspaceEx,
string datasetName)
{
    //Open a raster dataset in a geodatabase including PGDB, FGDB, and ArcSDE.
    return rasterWorkspaceEx.OpenRasterDataset(datasetName);
}

```

在 Geodatabase 中创建的栅格数据集是没有维数的，通常只是数据集中某些属于信息的占位符，如波段数、像素值、栅格列属性、几何列属性。一旦空的栅格数据集创建成功，栅格的像素值可以从别的数据集中通过拼接（mosaic）添加进来。下面的代码是在 Geodatabase 中创建栅格数据集的例子。

```

static void Mosaic(IRasterCatalog rasterCatalog)
{
    //Mosaics all rasters in the raster catalog to an output raster dataset
    IMosaicRaster mosaicRaster = new MosaicRasterClass();
    mosaicRaster.RasterCatalog = rasterCatalog;

    //Set mosaicking options, you may not need to set these for your data
    mosaicRaster.MosaicColormapMode = rstMosaicColormapMode.MM_MATCH;
    mosaicRaster.MosaicOperatorType = rstMosaicOperatorType.MT_LAST;

    //Open output workspace
    IWorkspaceFactory workspaceFactory = new RasterWorkspaceFactoryClass();
    IWorkspace workspace = workspaceFactory.OpenFromFile(outputFolder, 0);

    //Save out to a target raster dataset
    //It can be saved to IMG,TIFF,GRID,BMP,GIF,JPEG2000,JPEG,Geodatabase, ect.
    ISaveAs saveas = (ISaveAs)mosaicRaster;
    saveas.SaveAs(outputName, workspace, "IMAGINE Image");
}

```

3.4.3 获得栅格目录

栅格列表是在 ArcGIS 9.0 中新增加的数据类型，这种数据类型对空间数据库有效。栅格列表将多个数据集集合作为同一实体来管理，是一种特殊的要素类。该要素类中，栅格列存储栅格数据集，集合列存储栅格数据集的边框，栅格列中存储数据集的像素值。当然，在这个表中也

可以增加别的字段，如元数据等。

在栅格列中的值叫做 `RasterValue`，一个 `RasterValue` 包含一个 `RasterDataset` 和一个 `RasterStorageDef`。栅格列中还存储了作用于所有像素的空间参考信息，该空间参考是通过接口 `IRasterDef` 来定义的。几何列的属性是通过接口 `IGeometryDef` 来定义的。存在几何列表中的栅格数据集边框是由 `Geodatabase` 自动管理并被检索的。建议栅格列的空间参考与几何列的空间应相一致。

另外，接口 `IRasterDef` 也可以定义如何管理栅格列表。一个托管的栅格列存储栅格像素值，而一个非托管的栅格列仅仅存储的是栅格数据集的路径，同时这些数据集是基于文件方式的。在企业数据库中的栅格列是托管类型的，而在个人数据库中的栅格列是非托管类型的。对象 `RasterStorageDef` 定义如何在企业数据库存储像素值。用户可以定义块的大小、原点值，也可以定义压缩类型以及构建金字塔采用的重采样方式。

作为要素类的一个子类，`RasterCatalog` 是由行组成的。每行中的要素都是栅格列的条目，一种要素类型。当访问或更新栅格列中的数据集时，其操作与要素类相同。通过标准的 `FeatureCursor` 枚举栅格列中的数据集，并且可以通过插入光标或更新光标进行相应的插入或删除操作。

接口 `IRasterWorkspaceEx` 支持不管是个人数据库还是企业数据库的栅格列表访问。

3.4.4 栅格数据上载

1. 栅格列表方式

要想创建一个栅格列表首先要创建一个新的空栅格列表，然后记录，将栅格像素值插入栅格字段。下面的代码就是栅格数据上载到栅格列表中的一种方法。

```
void AddRasterToRasterCatalog(IRasterDataset rasterDataset, IRasterCatalog
rasterCatalog)
{
    IFeatureClass fClass = (IFeatureClass)rasterCatalog;
    IFeature feature = fClass.CreateFeature();
    //Create a raster value.
    IRasterValue rasterValue = new RasterValueClass();
    rasterValue.RasterDataset = rasterDataset;

    feature.set_Value(rasterCatalog.NameFieldIndex, rasterValue);
    feature.Store();
}
```

2. 栅格数据集方式

`ArcObjects` 的函数是基于 `ArcSDE C API` 实现的。这个函数主要是为了将栅格数据导入数据库中。用户可以基于 `ArcObjects` 开发自定义的应用程序来实现特殊需要的栅格数据的上载。`ArcObjects` 提供了许多额外的接口函数，这些接口或函数在 `ArcGIS` 中是不支持的。例如，使用 `IRaster` 接口用户可以上载部分数据到数据库中，也可以在上载前进行预处理。

接口 `IRasterSDEConnection` 定义连接信息，包括输出的工作空间名、输入栅格名、输出栅格名以及掩码文件。接口 `IRasterSDEStorage` 用来设置栅格数据的存储参数，如空间索引、块大小、压缩类型、配置关键字以及是否建立金字塔等信息。接口 `IRasterServerOperation` 主要操作服务器 `SDE` 会话，如创建、删除、拼接、更新、构建金字塔以及计算统计信息等。下面的函数是在

给定栅格文件路径下下载到 Geodatabase 的例子。

```

IWorkspaceName pWorkname = new ESRI.ArcGIS.Geodatabase.WorkspaceNameClass();
pWorkname.ConnectionProperties = pProp;
//pWorkname.WorkspaceFactoryProgID = "esriDataSourcesGDB.SdeWorkspace-
//Factory.1";

IRasterSdeConnection pSdeCon = new RasterSdeLoaderClass();
//pSdeCon.SdeWorkspaceName = pWorkname;

pSdeCon.ServerName = "192.168.9.177";
pSdeCon.Instance = "port:5151";
pSdeCon.UserName = "sde";
pSdeCon.Password = "sde";
pSdeCon.Database = "orcl";
pSdeCon.SdeRasterName = "S20080622130000";
pSdeCon.InputRasterName = @"d:\test\20080622130000.img";
IRasterSdeStorage pSDEStorage = (IRasterSdeStorage)pSdeCon;
IWorkspaceFactory pWKSF = new RasterWorkspaceFactoryClass();
IWorkspace pWorkspace = pWKSF.OpenFromFile(@"d:\test", 0);
IRasterWorkspace pRasterWS = (IRasterWorkspace)pWorkspace;

IGeoDataset pGeoDs = (IGeoDataset)pRasterWS.OpenRasterDataset ("200806
22130000.img");

// Set spatialreference
pSDEStorage.SpatialReference = pGeoDs.SpatialReference;
// Set compression
pSDEStorage.CompressionType = esriRasterSdeCompressionTypeEnum.esriRa-
sterSdeCompressionTypeUncompressed;
//Set tilesize
pSDEStorage.TileHeight = 128;
pSDEStorage.TileWidth = 128;
// Pyramids option
pSDEStorage.PyramidOption = esriRasterSdePyramidOptEnum.esriRaster-
SdePyramidBuildWithFirstLevel;
pSDEStorage.PyramidResampleType = rstResamplingTypes.RSP_ Bilinear-
Interpolation;
//pSDEStorage.CompressionType = esriRasterSdeCompressionTypeJPEG2000;

// Start loading
IRasterSdeServerOperation pSDEOp = (IRasterSdeServerOperation)pSdeCon;

pSDEOp.Create();
pSDEOp.Update();
pSDEOp.ComputeStatistics();

//释放内存
pSdeCon = null;
pSDEStorage = null;
pSDEOp = null;

```

```
pWKSF = null;
pRasterWS = null;
```

3.4.5 栅格数据拼接

影像拼接是将输入的多个图像根据坐标范围拼接成一个图像。影像拼接通俗地讲也可以认为是合并或添加。基本的思想是：首先，将一个图像通过 ArcSDE 导入到数据库；其次，一系列的影像图在原图的基础上进行拼接，形成一个新的图像。当影像进行拼接的时候，在两个图像重叠区域，会用新图像去覆盖原来图像的区域。

ArcSDE 对图像拼接的要求很高，具体的要求如下：

- 运行图必须具有相同的像素深度；
- 影像图必须具有相同的波段数；
- 在数据库中存在的影像图必须具有空间参考；
- 影像图必须具有相同的格子大小；
- 影像图必须进行精确像素纠正。即相邻影像图间各行像元要对齐，否则无法入库。

一般情况下，进行拼接的数据为经过校正后的数据。在数据库中图像拼接的时候，检测数据库中原有图像是否构建金字塔结构，如果是将先删除金字塔结构，然后进行拼接。因此，一般建议先导入图像，导入完成后，再构建金字塔，这样可以减轻服务器的压力，减少数据读写硬盘的次数，提高整体的性能。

下面的代码就是实现一栅格文件入库后与数据库中已经存在的数据集进行拼接，具体实现如下：

```
//利用 geoprocessing mosaic tool class to 把输入的图片拼接为目标图层中
//geodatabase (file geodatabase or ArcSDE) 栅格数据
Mosaic mosaic = new Mosaic();
mosaic.inputs = @"c:\tempdata\air\air1.tif;c:\tempdata\air\air2.img";
mosaic.target = @"c:\tempdata\new.gdb\AIRPHOTOS";

//设置参数

//设置背景值
mosaic.background_value = 0;

//设置 NoData 值
mosaic.nodata_value = 0;

//设置拼接模式
mosaic.mosaic_type = "LAST";

//设置颜色匹配模式
mosaic.colormap = "MATCH";

//设置容差
mosaic.mosaicking_tolerance = 0.45;
Geoprocessor geoprocessor = new Geoprocessor();
```

```
//执行拼接  
object outRaster = geoprocessor.Execute(mosaic, null);
```

3.5 小结

通过本章的学习，读者将掌握如下内容：

- 基本了解 Geodatabase 模型的概念以及明白其模型中的主要对象的功能。
- 掌握打开远程数据库或者个人数据库的数据库工作空间的方法，并懂得通过 Name 对象方式和通过实体对象方式访问空间数据库中要素的方法。
- 了解矢量文件导入数据库的方法，了解对数据库中数据库编辑的事务流程，并掌握对数据库中的数据进行拓扑查错的方法。
- 掌握栅格数据在空间数据库中的存储模式以及如何获得栅格数据，并给出两种上载数据的方式，掌握栅格数据进行拼接的方法。

第二篇

ArcGIS Server

的开发基础

- ▶ 第 4 章 ArcGIS Server 地图服务发布
(准备开发的数据)
- ▶ 第 5 章 ArcGIS Server 开发基础 ASP.NET
- ▶ 第 6 章 ArcGIS Server 控件介绍

本篇主要对 ArcGIS Server 的开发基础：数据准备、ASP.NET 的相关知识以及 ArcGIS Server 9.3 提供的控件进行一一介绍，通过这些开发基础的学习，读者可进行一些简单功能的开发。

第4章

ArcGIS Server地图服务发布 (准备开发的数据)

ArcGIS Server 9.3 可以支持多种服务类型，用户通过 ArcGIS Server 发布的这些服务可以享受 GIS 功能。ArcGIS Server 9.3 支持以下的服务类型。

- **Map Service** 是 ArcGIS Server 9.3 用户使用最多的一种服务，它支持发布二维矢量数据和栅格数据，支持建模及 OGC 的 WMS 和 KML 服务，也支持空间数据的在线编辑。
- **Geocode Service** 即地址编码服务，它可以把文本描述的地址转化为一个地理坐标。
- **Geoprocessing Service** 是基于 Web 的地理处理工具。客户端提交处理请求，服务器执行空间分析和建模，然后把处理结果返回客户端展现。
- **Geodata Service** 提供访问 Geodatabase 数据库空间数据的功能，支持空间数据在线查询、检索和更新。
- **Globe Service** 是 ArcGIS Server 提供的 3D 服务。利用 ArcGlobe 中创建的 3DD 文档，然后通过 ArcGIS Server 发布即可。用户可以通过 ESRI 的免费产品 ArcExplorer 来浏览三维空间数据服务。
- **Geometry Service** 是空间对象服务，它支持投影变换，创建缓冲区以及确定空间对象的拓扑关系。
- **Image Services** 可以直接发布栅格数据，发布栅格数据的性能比 Map Service 更好，客户端可以改变影像的压缩率，支持 OGC 的 WMS、WCS、REST、KML 等标准。

4.1 制作地图文档

要制作 ArcGIS Server 的服务，首先必须准备空间数据，下面就讲述如何获取空间数据，以及如何制作发布服务所需要的地图文档。

4.1.1 获取空间数据

在搜索界面输入“国家基础地理信息系统”，如图 4.1 所示。经搜索可以定位“国家基础地理系统”网站。国家基础地理信息系统是中国最大的全国地理信息存储、数据管理、地图生产和数据应用系统之一。是国家测绘局的专业系统，是国家空间数据基础设施的重要组成部分。进入“国家基础地理信息系统”界面，可以发现该网站提供地理信息数据下载，如图 4.2 所示。单击下载服务，进入下载页面，如图 4.3 所示。主要可供下载的空间数据有全国 1:400 万矢量数据，包括国界、省界、地市级以上城市、三级以上河流、主要公路和主要铁路等数据。这些矢量数据是经纬度坐标 e00 文件格式的数据。本次实验主要下载 1:400 万国界和省界、1:400 万一级以上河流、1:400 万主要公路、1:400 万主要铁路、1:400 万首都和省级行政中心，填写完用户信息后就可以完成 1:400 万空间数据的下载。

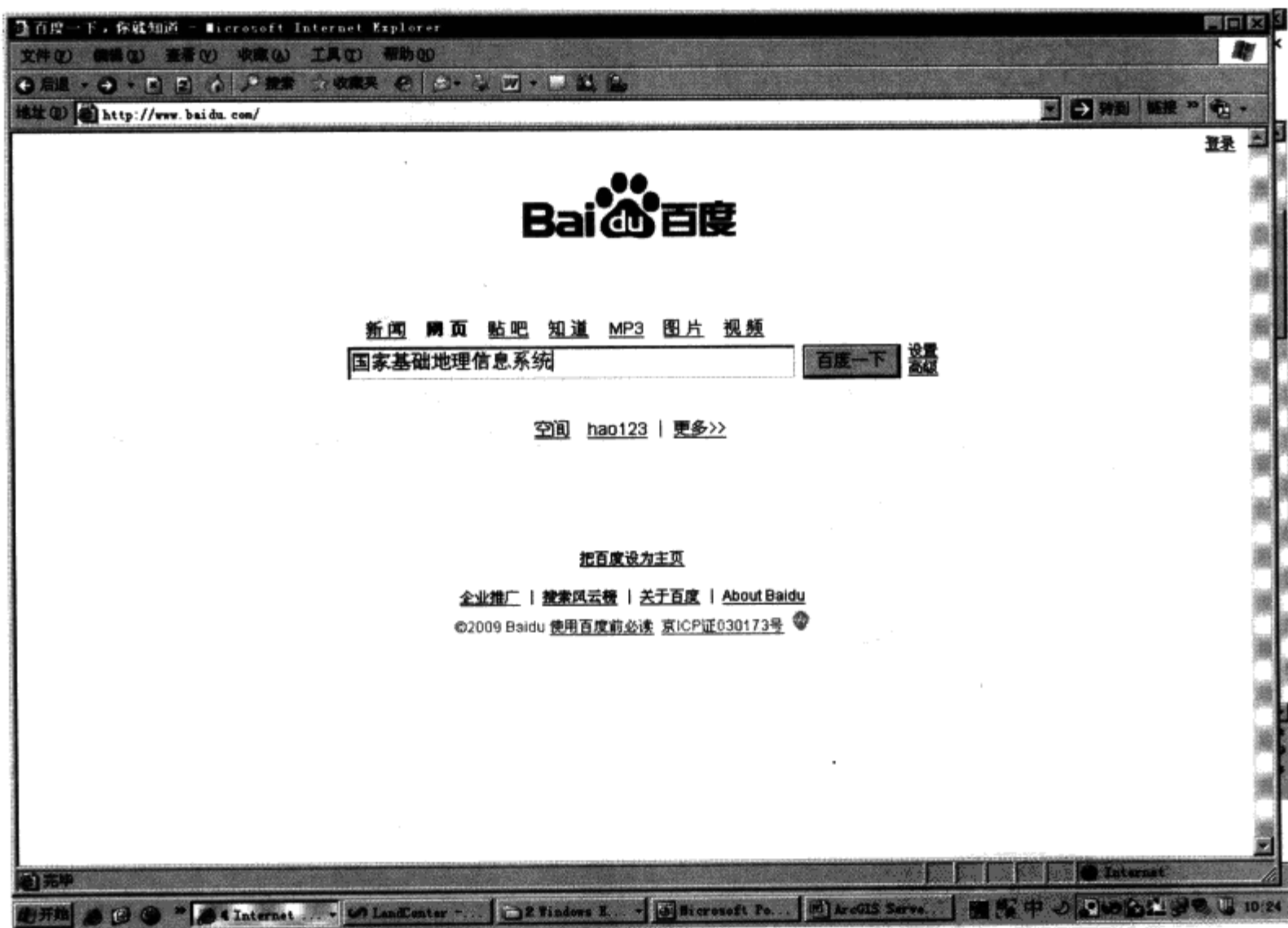


图 4.1 搜索空间数据下载网址

4.1.2 使用 ArcMap 编辑地图文档

在 4.1.1 节中下载的数据是 e00 格式的数据，经过转换处理为编程通用的 Shape 格式，下面讲述如何编辑地图文档。

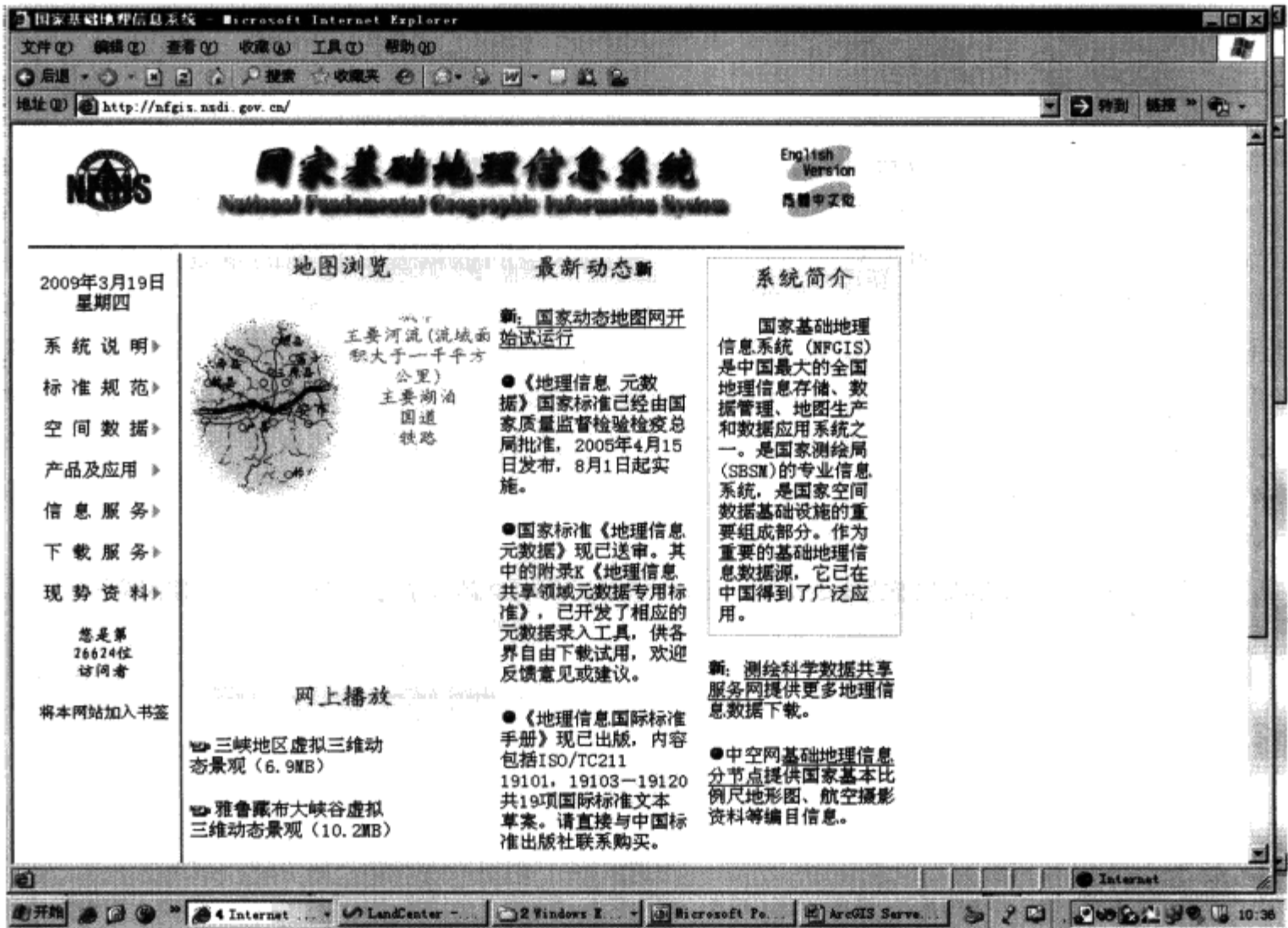


图 4.2 “国家基础地理信息系统”网站

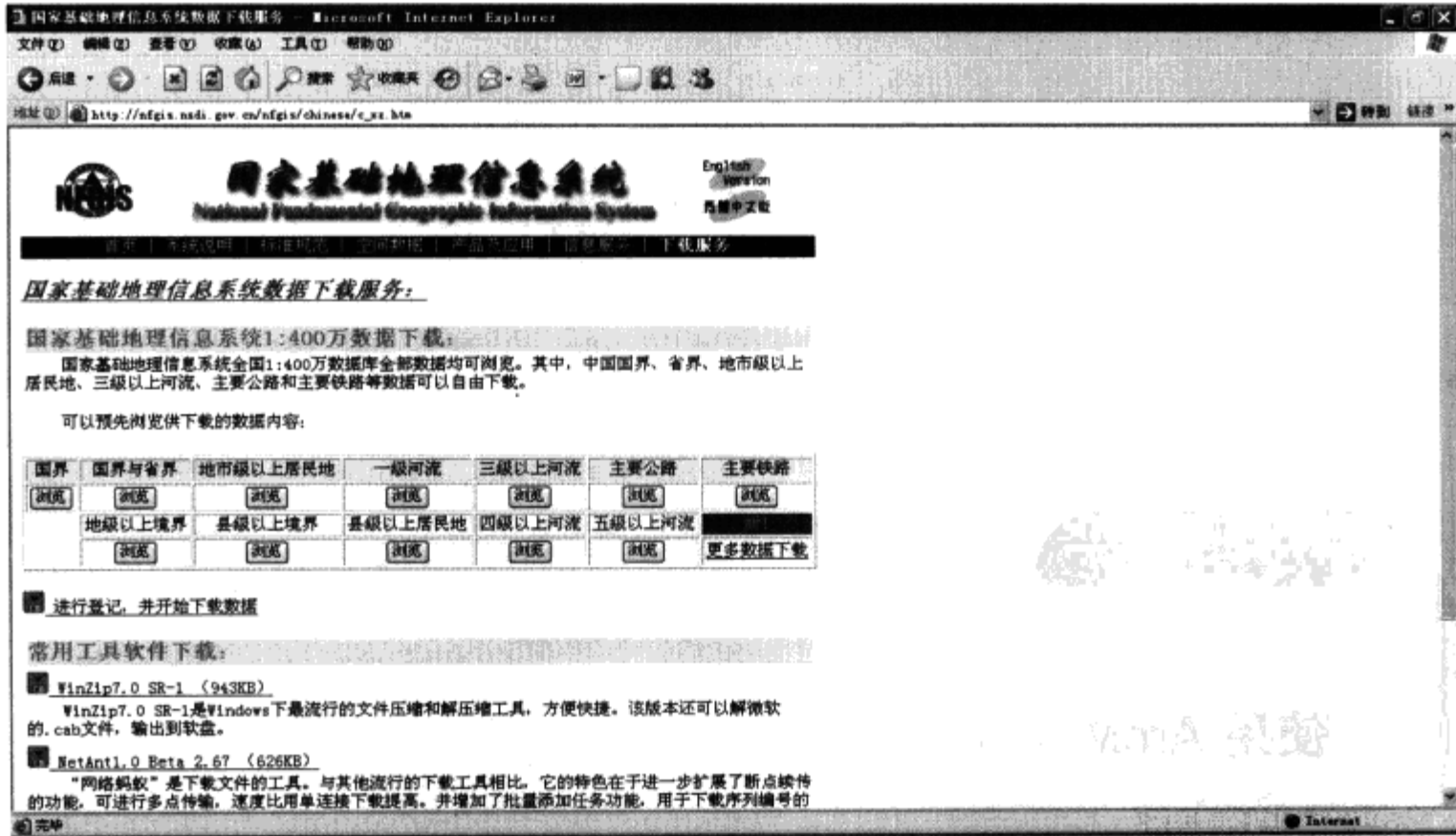


图 4.3 矢量空间数据下载页面

(1) 运行 ArcMap，创建一个空的地图文档，如图 4.4 所示。

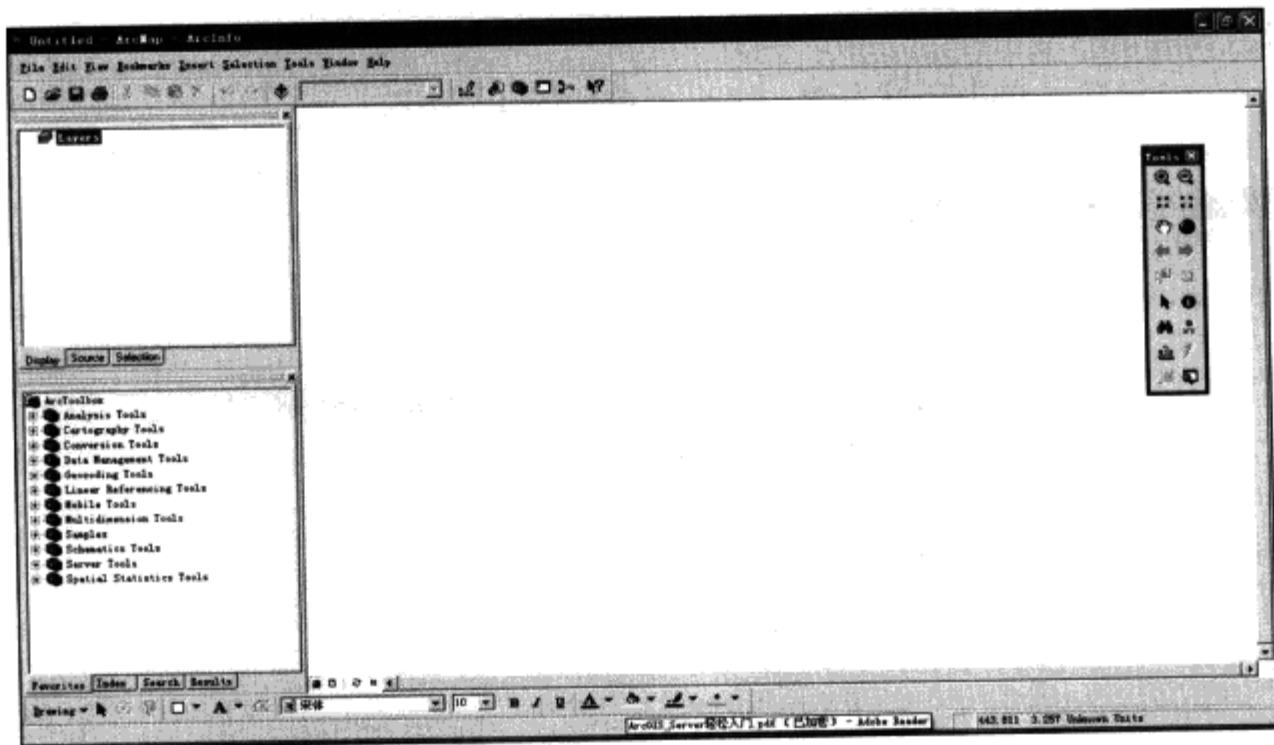


图 4.4 ArcMap 运行界面

(2) 单击“Add Data”按钮，系统弹出“Add Data”对话框，如图 4.5 所示。

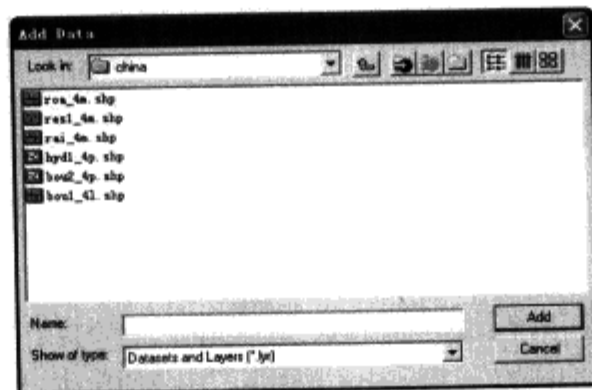


图 4.5 “Add Data”对话框

(3) 将 4.1.1 节中下载的 1:400 万全国的矢量数据全部选中并添加到 ArcMap 中，如图 4.6 所示。

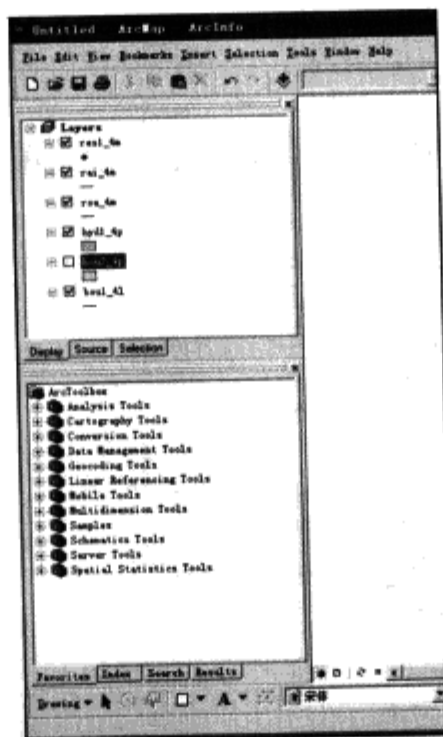


图 4.6 在 ArcMap 中加载数据

(4) 设置国界的符号。

选中 bou1_4l 链接，单击鼠标右键，弹出菜单如图 4.7 所示。

单击“Properties”菜单，系统弹出“Layer Properties”对话框，如图 4.8 所示。

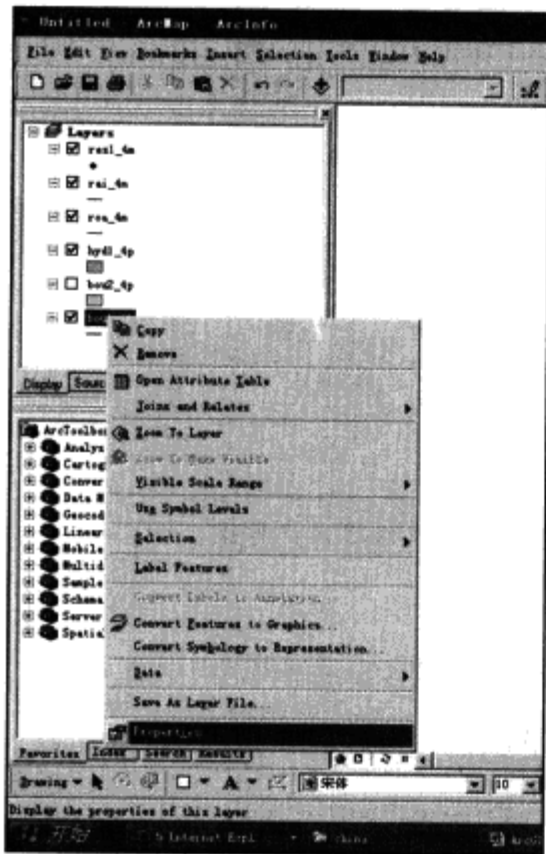


图 4.7 设置国界图层属性

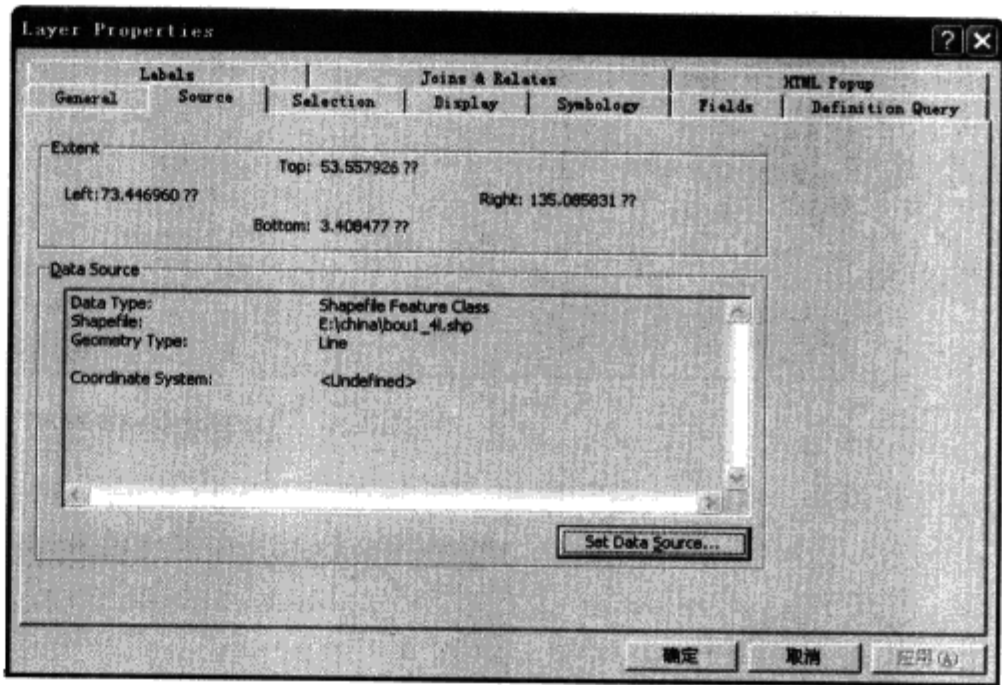


图 4.8 “Layer Properties”对话框

设置图层的别名以及符号，如图 4.9 所示。

(5) 设置省界符号。设置步骤与设置国界相似，不同的是首先在“Symbology”标签页的“Show”列表框中选中“Categories”选项中的“Unique values”，然后在“Value Field”中选择“NAME”字段，在“Color Ramp”颜色列表中选择自己喜欢的一种颜色变化范围，然后单击“Add All Values”按钮，如图 4.10 所示。

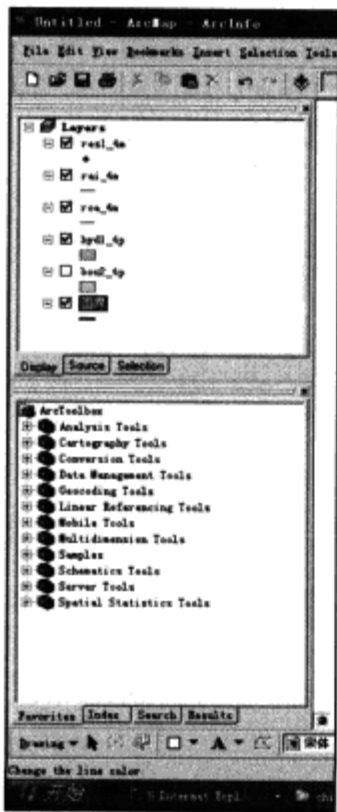


图 4.9 设置国界别名和符号

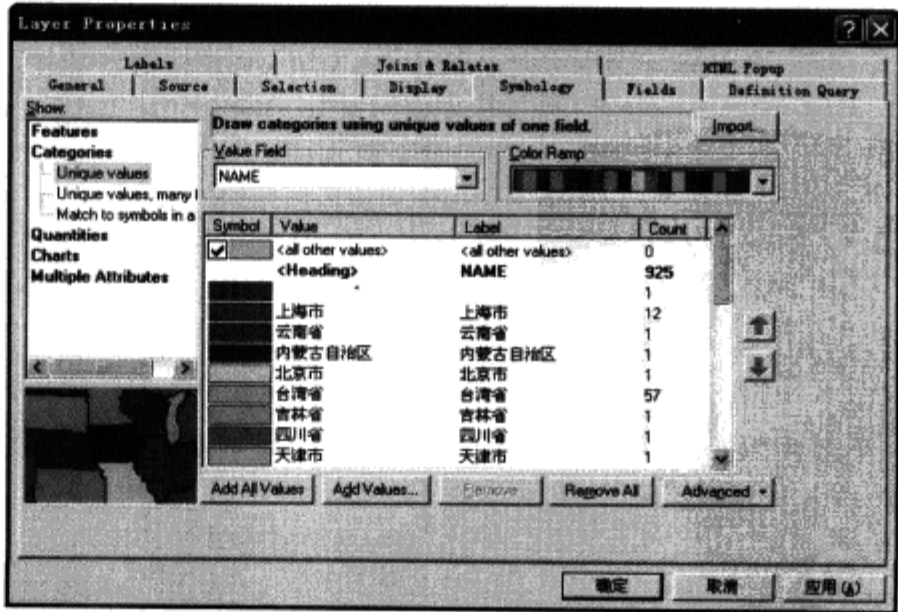


图 4.10 设置省界符号

在“Labels”标签页中，选中“Label features in this layer”复选框，在“Label Field”下拉列表框中选择“NAME”字段，如图 4.11 所示。

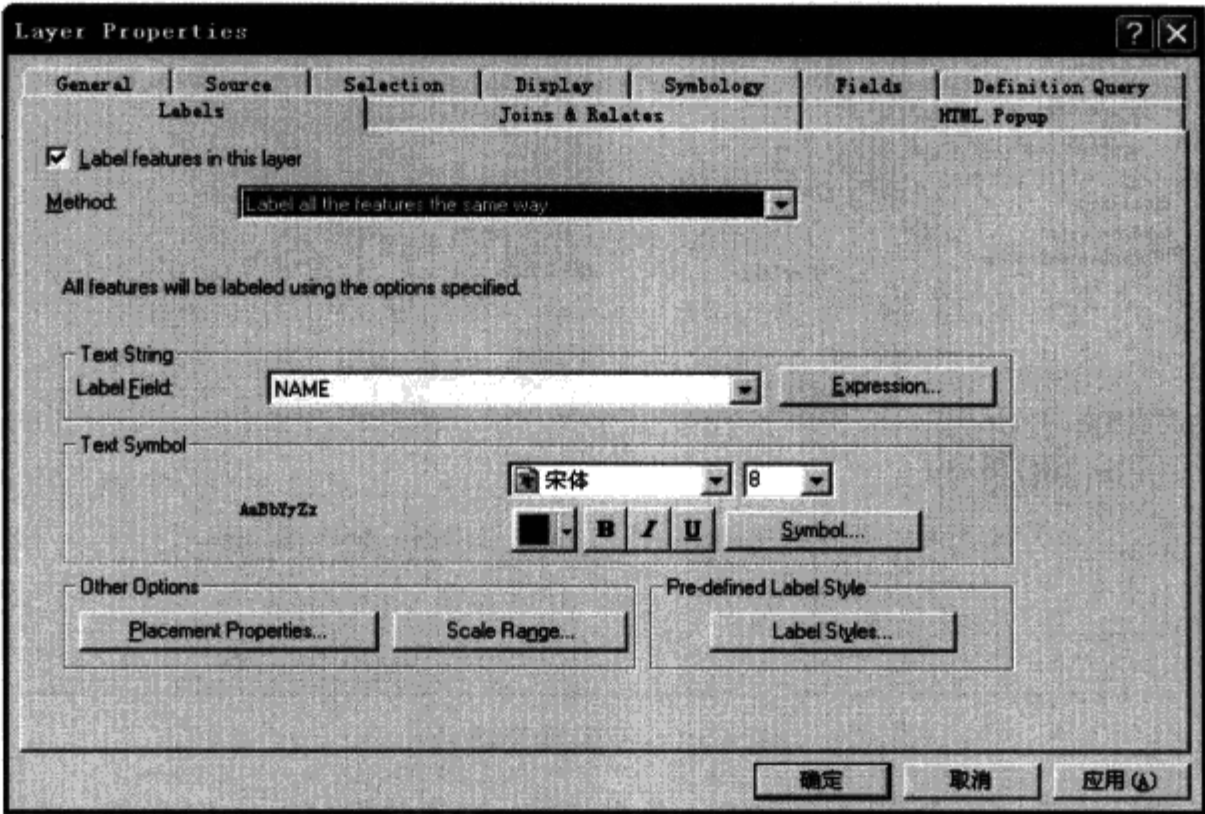


图 4.11 设置省界 Label 字段

（6）设置主要铁路。设置方法与前面类似，不同的是在设置符号的时候，双击符号按钮，选择铁路的符号，如图 4.12 所示。

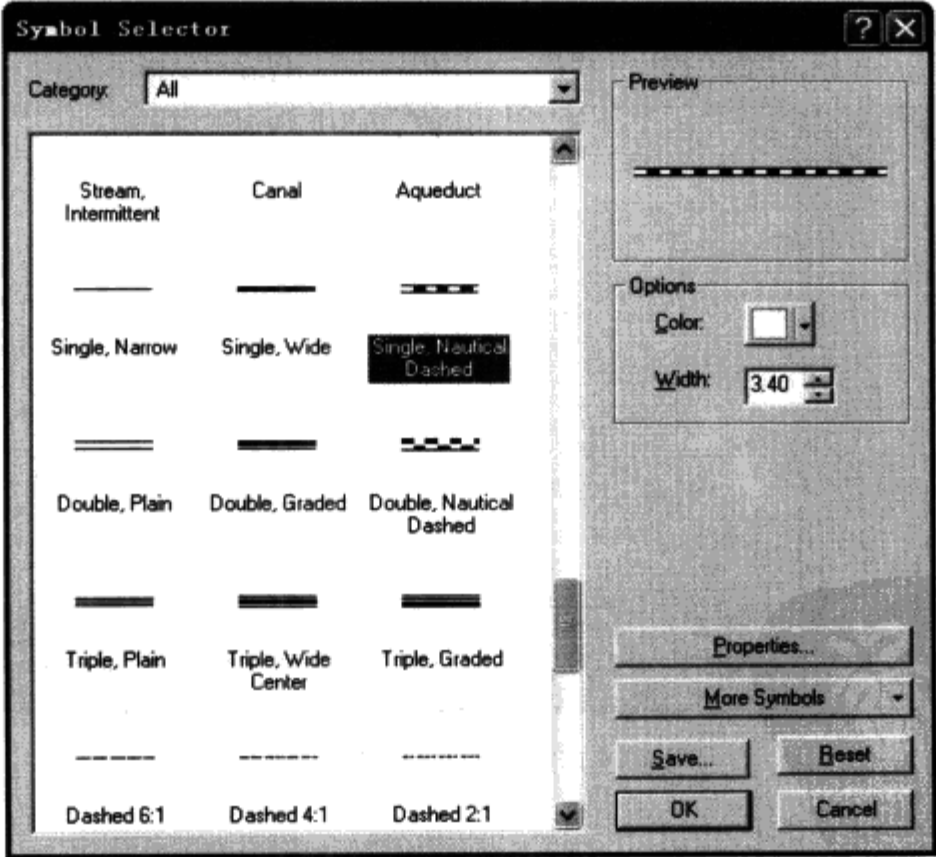


图 4.12 设置铁路符号

- （7）设置主要公路，与上述步骤相似。
- （8）设置主要城市。设置主要城市符号如图 4.13 所示，在此不再赘述。

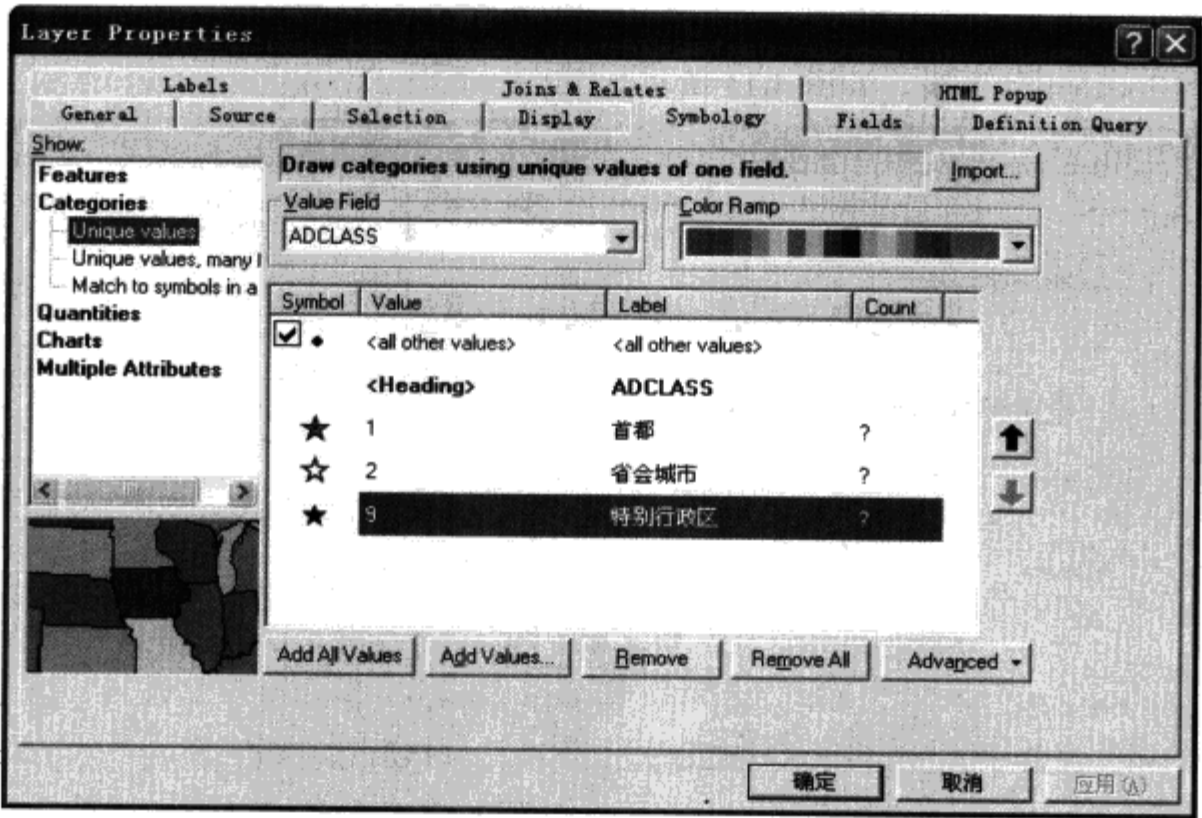


图 4.13 设置主要符号

(9) 在所有的符号和别名设置完成后，单击“File”菜单的“Save”选项，系统弹出“另存为”对话框，选择文件保存的路径，并在“文件名”文本框输入文件名，在“保存类型”下拉列表框选择“ArcMap Document”，然后单击“保存”按钮，如图 4.14 所示。

至此，待发布的地图文档制作完成。

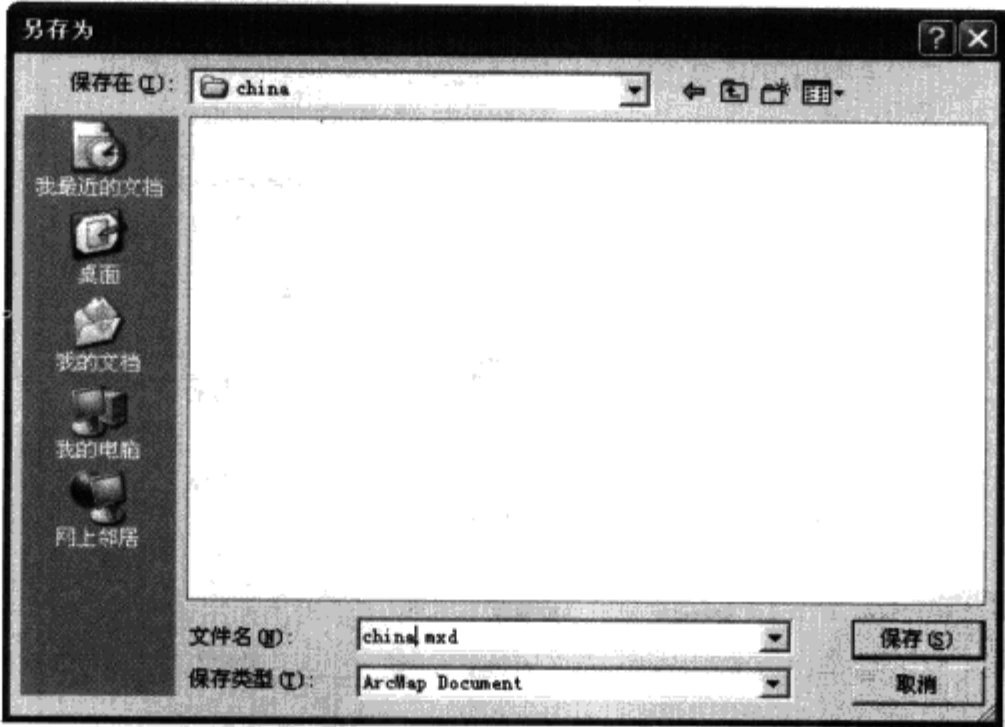


图 4.14 保存地图文档对话框

4.2 用户权限设置

ArcGIS Server 安装完成后，创建了两个组 agsadmin 和 agsusers，agsadmin (ArcGIS Server Administrator) 是 ArcGIS Server 的管理员组；agsusers (ArcGIS Server Users) 是 ArcGIS Server 的用户组。在 ArcGIS Server 9.3 版本之前，用户必须具有 Web 服务器的管理员权限才能登录到

管理器。在 9.3 版本中用户已不再必须具备管理员身份，只需要是 SOM 机器 agsadmin 组的一个成员就可以登录到管理器。

agsadmin 用户可以浏览、创建、停止、启动和删除服务。非系统管理员可以利用地理信息系统服务器，修改 SOC 机器、日志文件属性、服务器目录等内容。

如果用户需要利用管理器创建或编辑 Web 应用程序，则依然需要管理员身份才能登录到 ArcGIS 9.3。

用户在进行 ArcGIS Server 的管理和应用时，可以根据需要设置用户的权限。下面详细讲述如何设置 agsadmin 账户。

(1) 设置 agsadmin 用户，选中“我的电脑”，单击鼠标右键，系统弹出如图 4.15 所示的菜单。

(2) 选中“管理”项，系统弹出“计算机管理”对话框，选中“本地用户和组”，如图 4.16 所示。

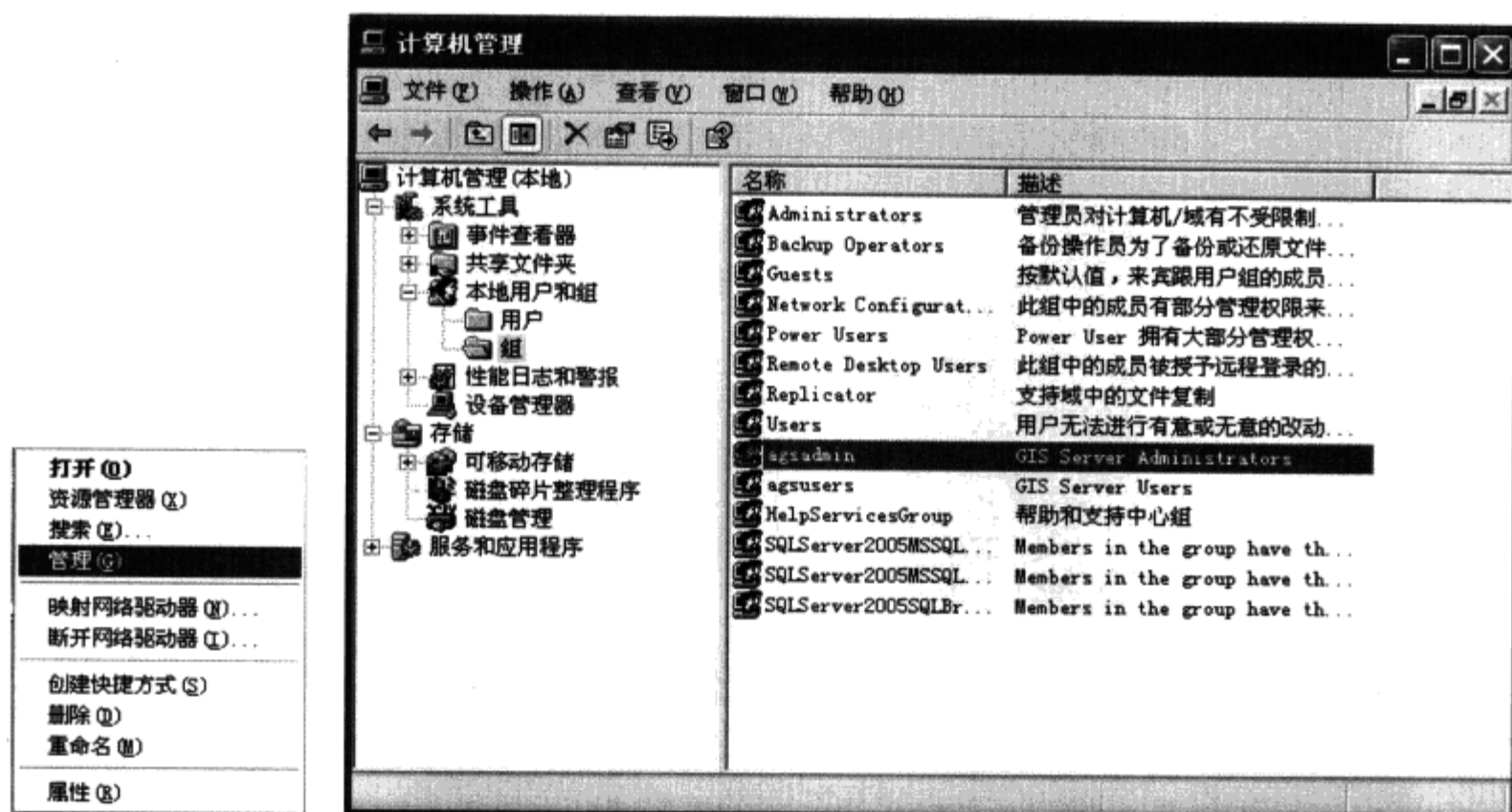


图 4.15 计算机管理菜单

图 4.16 “计算机管理”对话框

(3) 双击 agsadmin 用户，在弹出的“agsadmin 属性”对话框中设置 ArcGIS Server 管理员账户，如图 4.17 所示。

设置 agsusers 的步骤与设置 agsadmin 账户相同，在此不再赘述。

4.3 在 ArcCatalog 中发布 Map Service

前面几节讲述了如何准备 ArcGIS Server 服务发布的数据和设置 ArcGIS Server 的用户权限，本节主要讲述如何在 ArcCatalog 发布 ArcGIS Server 中最常见的服务 Map Service。步骤如下。

(1) 以 agsadmin 组的用户身份登录操作系统。

(2) 启动 ArcCatalog，如图 4.18 所示。

(3) 在 ArcCatalog 目录树中，展开 GIS Servers，双击“Add GIS Server”项，系统弹出“Add ArcGIS Server”对话框，选中“Manage GIS Services”选项，如图 4.19 所示。



图 4.17 设置 agsadmin 账户

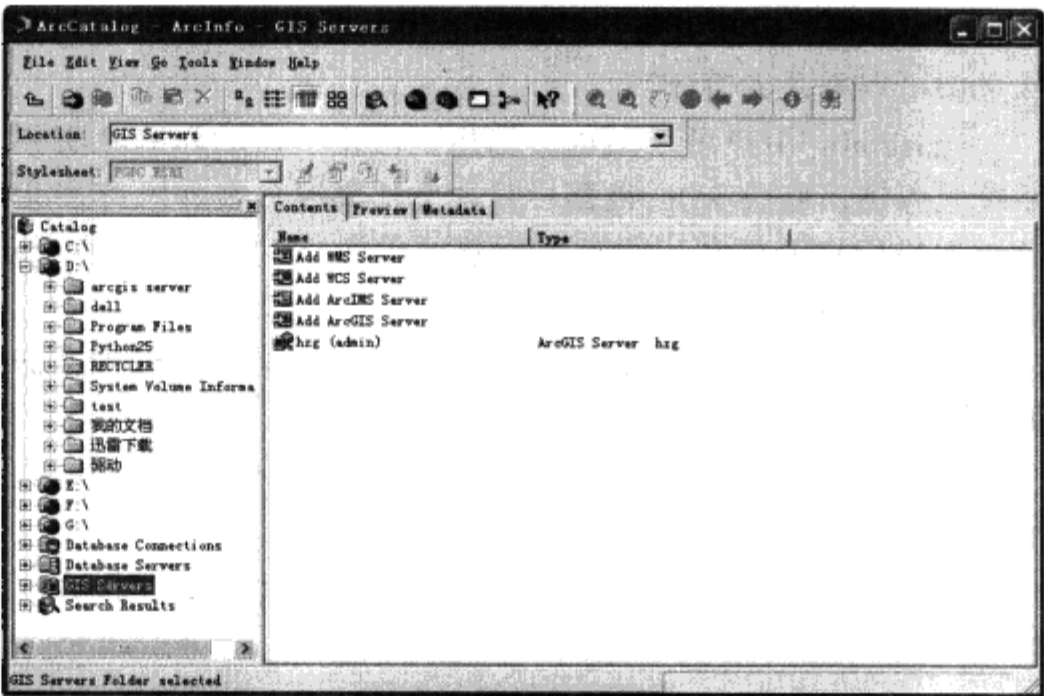


图 4.18 ArcCatalog 界面

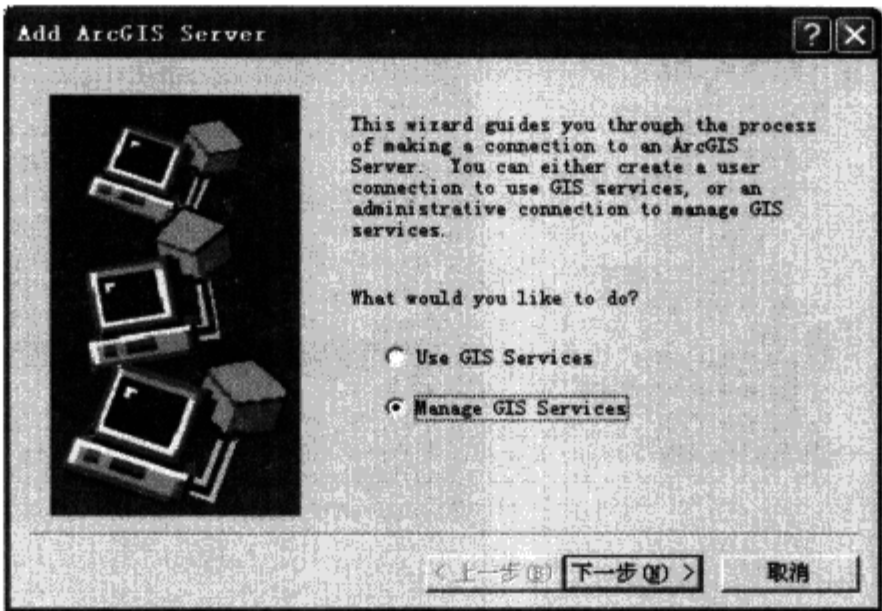


图 4.19 “Add ArcGIS Server” 对话框

(4) 单击“下一步”按钮，系统弹出“General”对话框，如图 4.20 所示。

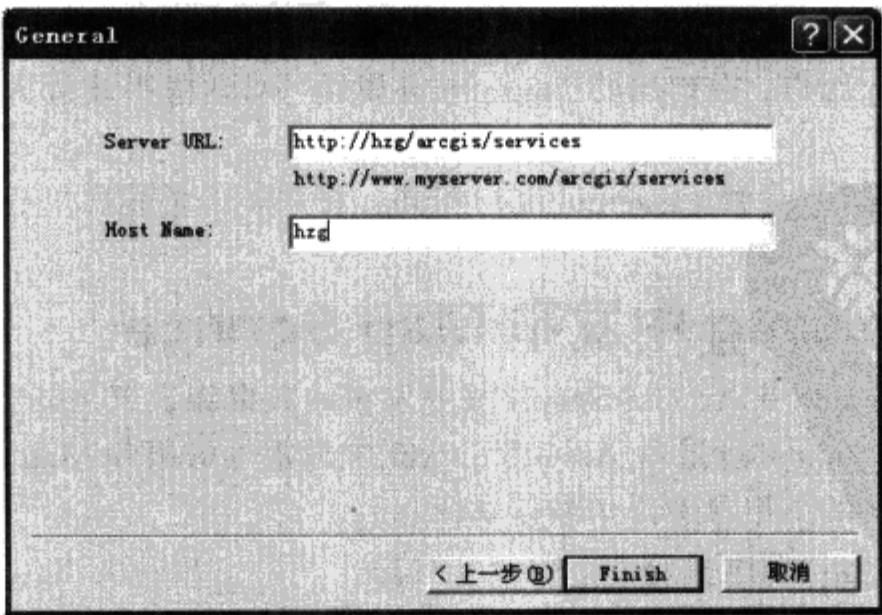


图 4.20 “General” 对话框

(5) 在“Server URL”和“Host Name”文本框中输入本机的 ArcGIS Server 的 URL 和主机名字，单击“Finish”按钮。

(6) 在 ArcCatalog 目录树中，定位在 4.1 节中制作的地图文档，如图 4.21 所示。

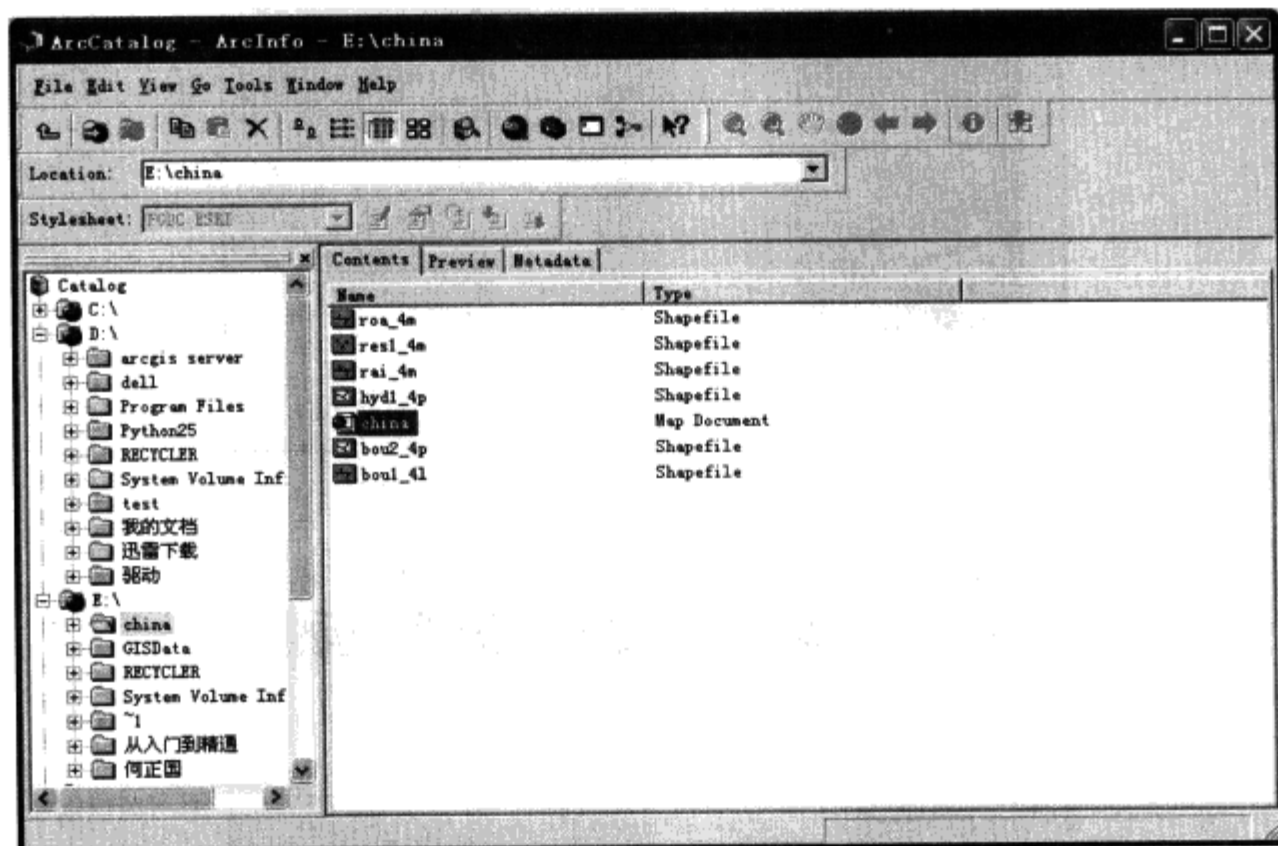


图 4.21 定位 china.mxd 文档

(7) 选中 china.mxd 文档，单击鼠标右键，系统弹出如图 4.22 所示的菜单。

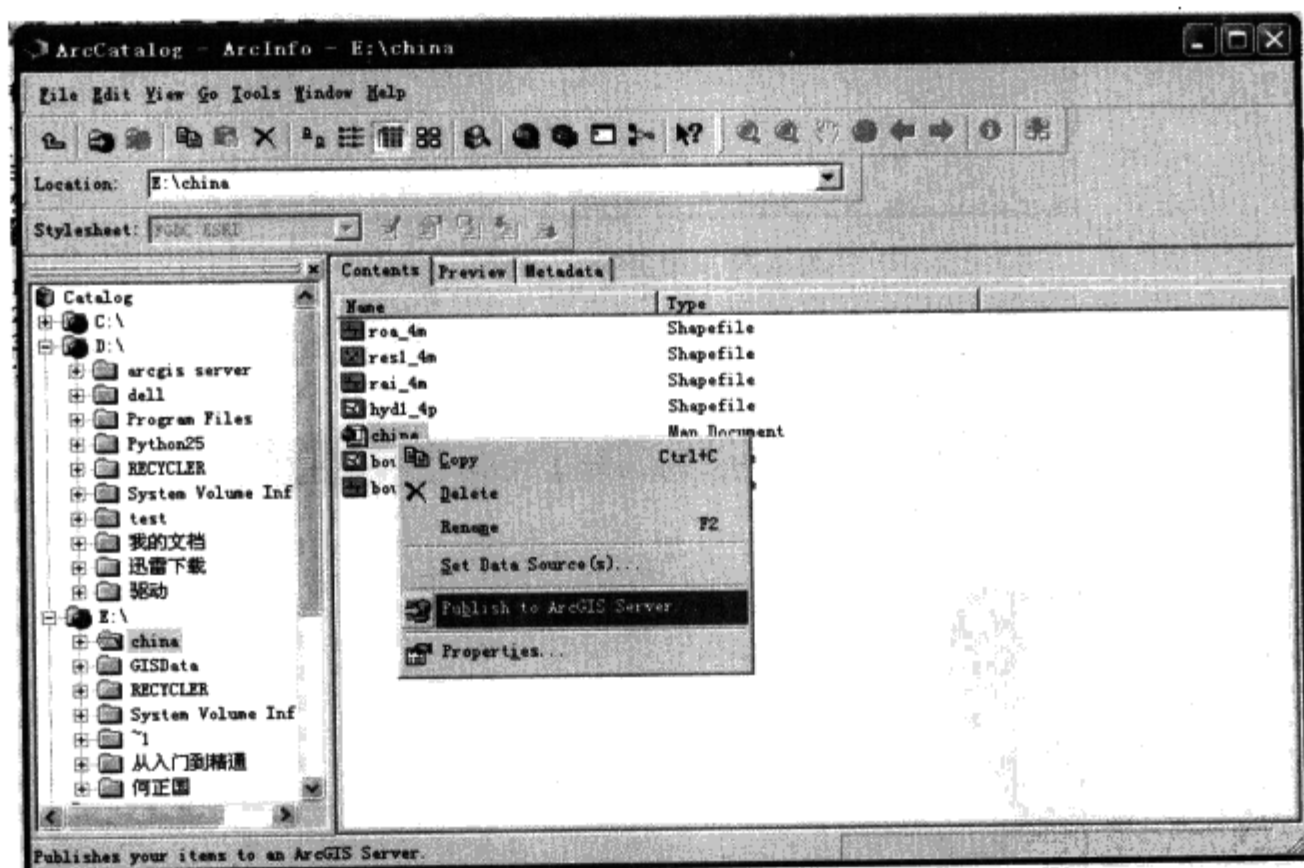


图 4.22 选择“Publish to ArcGIS Server”选项

(8) 选择“Publish to ArcGIS Server”项，系统弹出“Publish to ArcGIS Server”对话框，如图 4.23 所示。

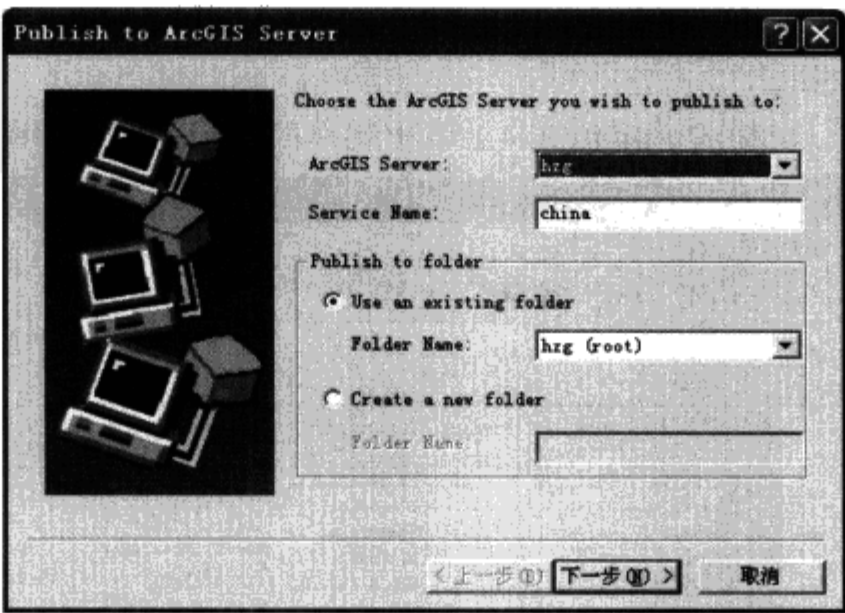


图 4.23 “Publish to ArcGIS Server” 对话框

(9) 单击“下一步”按钮，系统弹出发布类型对话框，如图 4.24 所示。

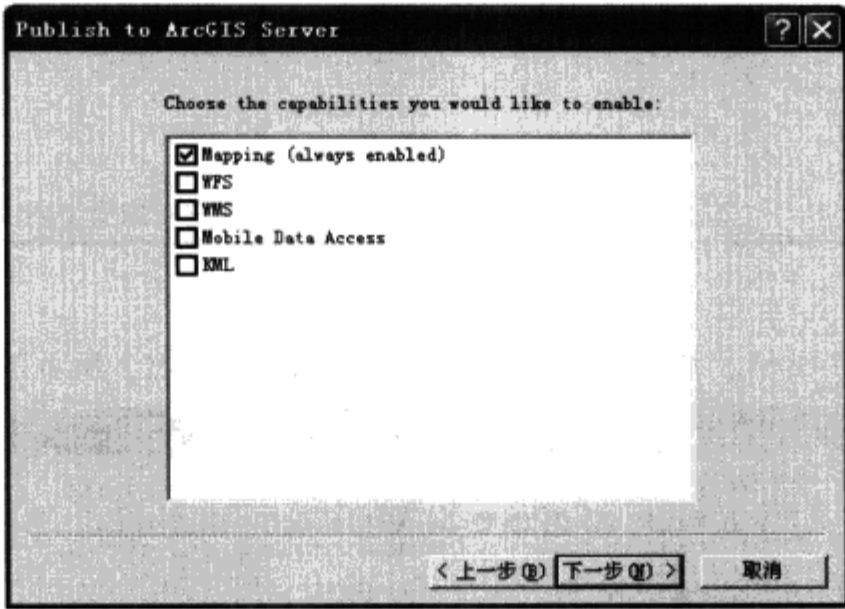


图 4.24 服务类型选择

(10) 在发布的服务类型中选择“Mapping (always enabled)”选项，单击“下一步”按钮，系统弹出发布服务结束对话框，如图 4.25 所示。

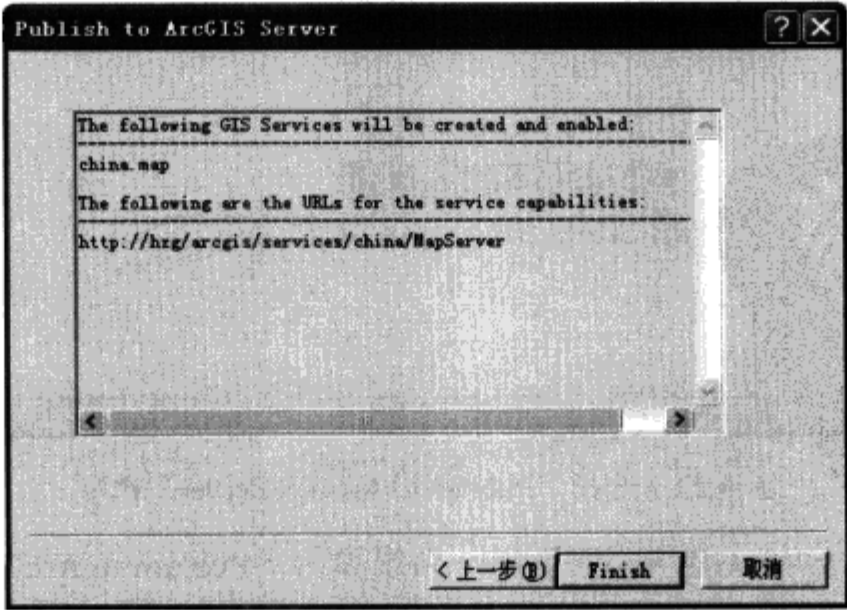


图 4.25 发布服务结束

（11）单击“Finish”按钮，完成服务的发布。在 ArcCatalog 中查看服务是否发布成功，成功则可以查看地图，反之则不能，如图 4.26 所示。

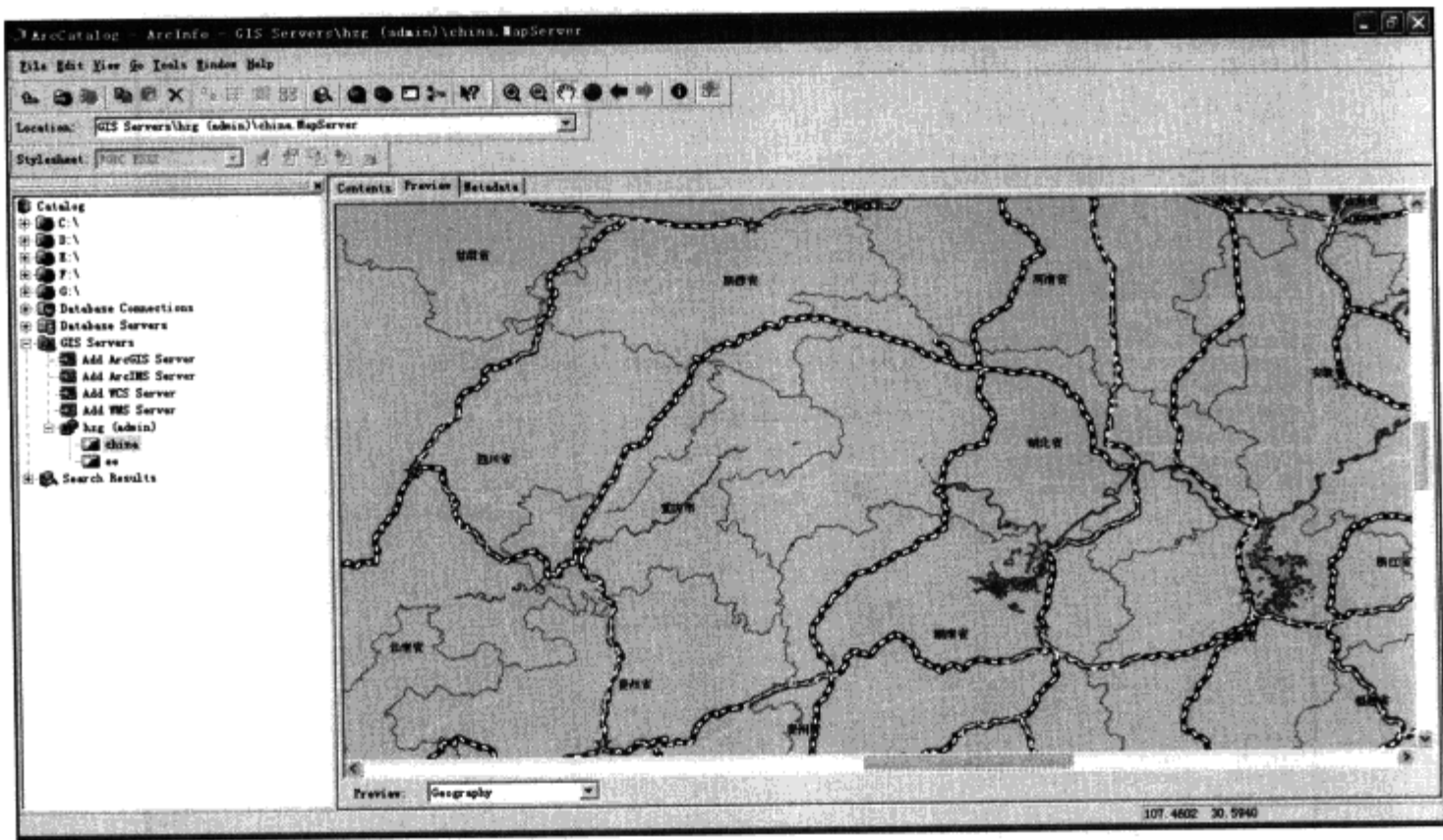


图 4.26 发布服务成功示意图

4.4 在 ArcGIS Server Manager 中发布 Map Service

除了在 ArcCatalog 中可以发布地图服务外，在 ArcGIS Server Manager 中也可以发布地图服务。发布步骤如下。

- （1）首先共享地图文档目录。
- （2）启动 ArcGIS Server Manager 登录界面，如图 4.27 所示。

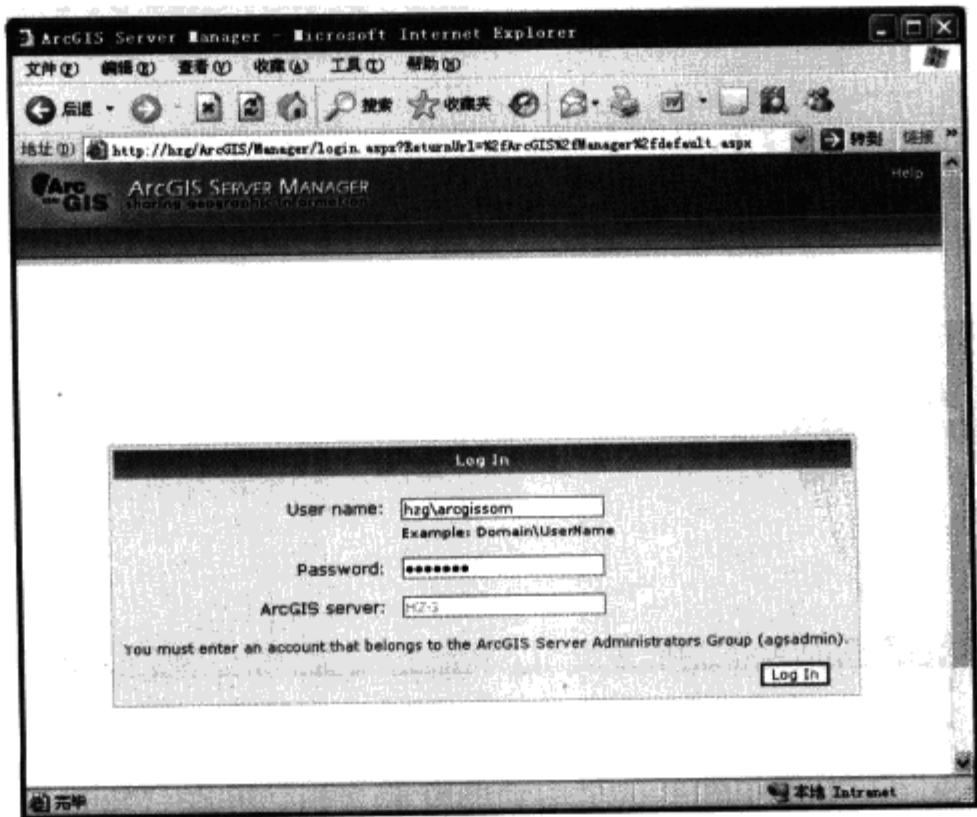


图 4.27 “ArcGIS Server Manager” 登录界面

(3) 在“User name”和“Password”对话框中分别输入 agsadmin 用户组中的用户名和密码，单击“Log In”按钮，系统进入“ArcGIS Server Manager”管理界面，如图 4.28 所示。

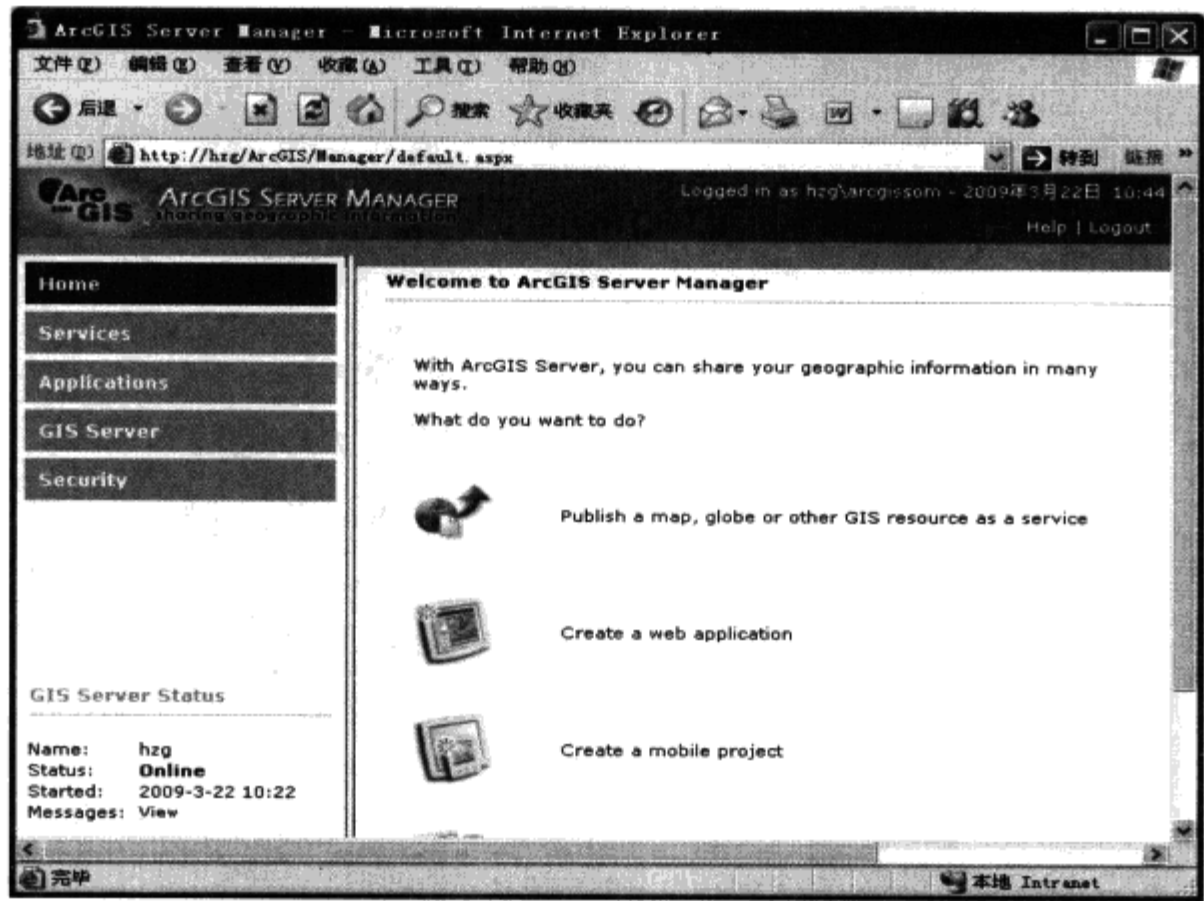


图 4.28 “ArcGIS Server Manager” 管理界面

(4) 单击“Publish a map, globe or other GIS resource as a service”选项，系统界面如图 4.29 所示。

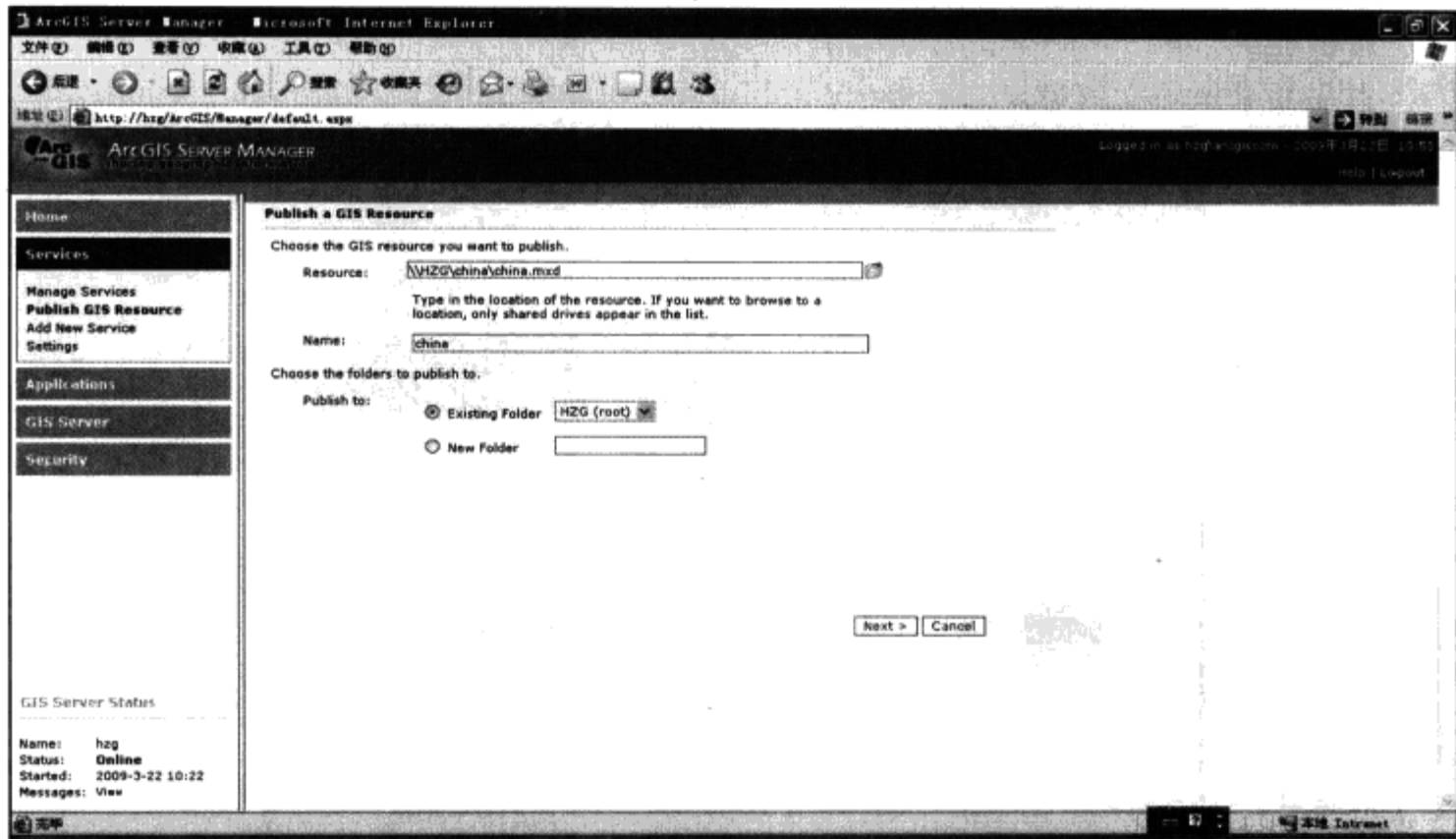


图 4.29 选择地图资源目录

(5) 在“Resource”文本框中选择共享的地图资源文档，并在“Name”文本框中输入服务的描述信息，然后单击“Next”按钮，系统进入服务类型选择界面，如图 4.30 所示。

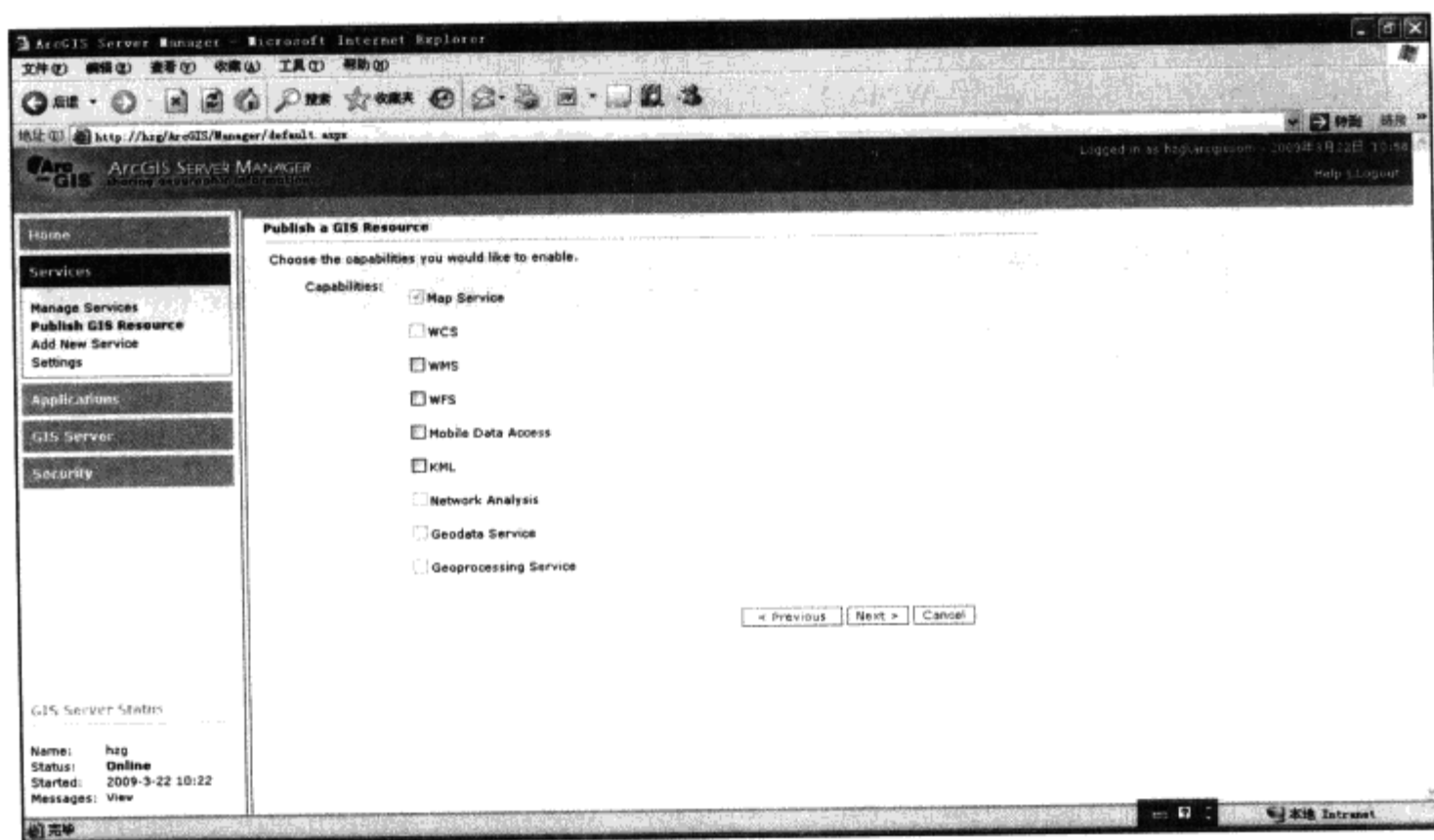


图 4.30 服务类型选择界面

(6) 选择“Map Service”选项，单击“Next”按钮进入服务创建界面，然后单击“Finish”按钮，结束服务的创建，如图 4.31 所示。

服务发布完成后，可以在 ArcCatalog 或 ArcMap 中查看刚才发布的地图服务。

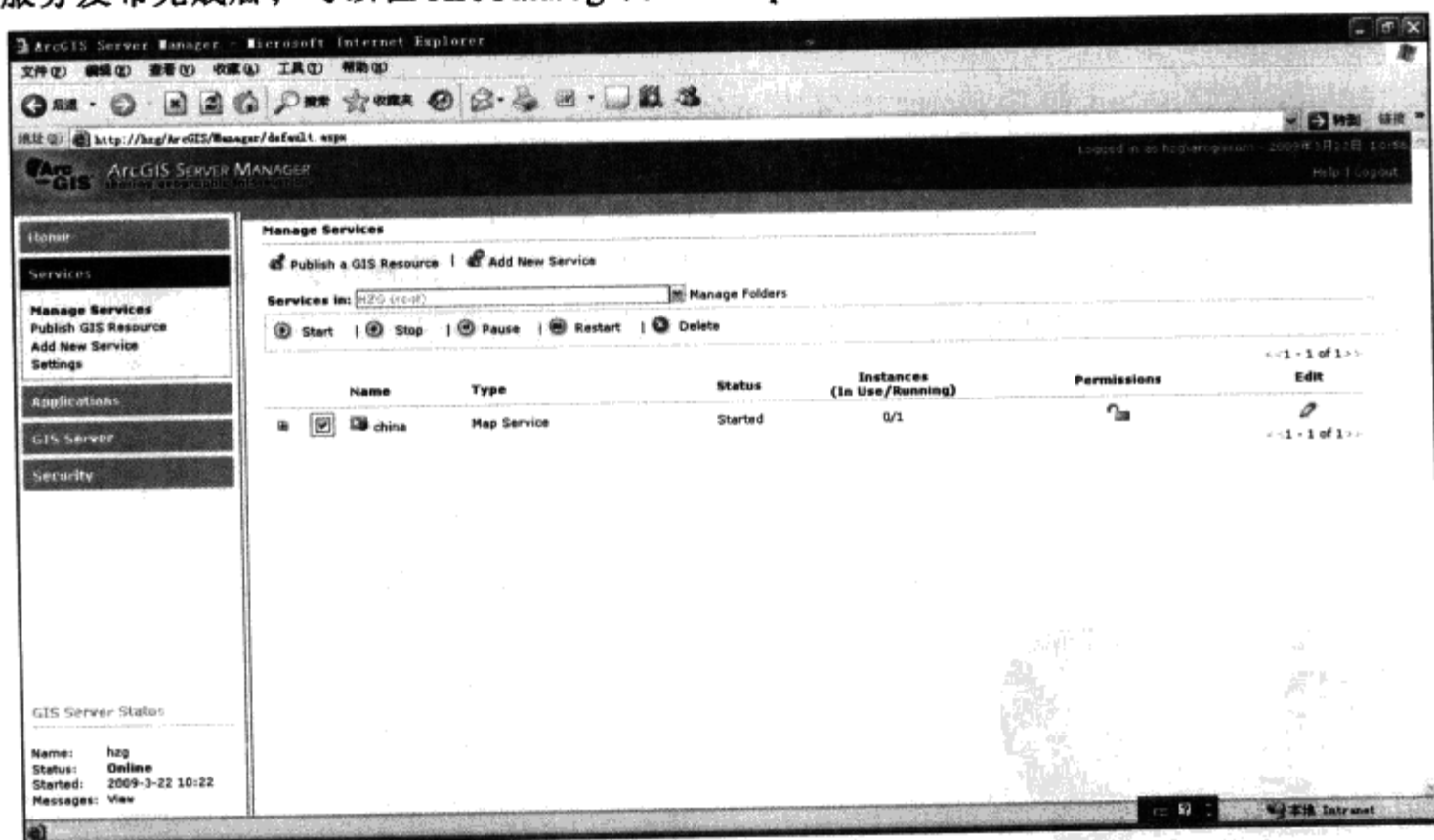


图 4.31 服务发布完成界面

4.5 在 ArcGIS Server Manager 中发布 OGC 地图服务

在 ArcGIS Server Manager 中，除了发布 ArcGIS Server Manager 最常用的 Map Service 外，还可以发布 OGC 的地图服务，下面介绍在 ArcGIS Server Manager 中发布 OGC 的地图服务。

(1) WMS 是地图描述服务。用户可以通过该服务中提供的图层集合生成地图。
在上面发布服务的时候，选中 WMS 复选框，如图 4.32 所示。

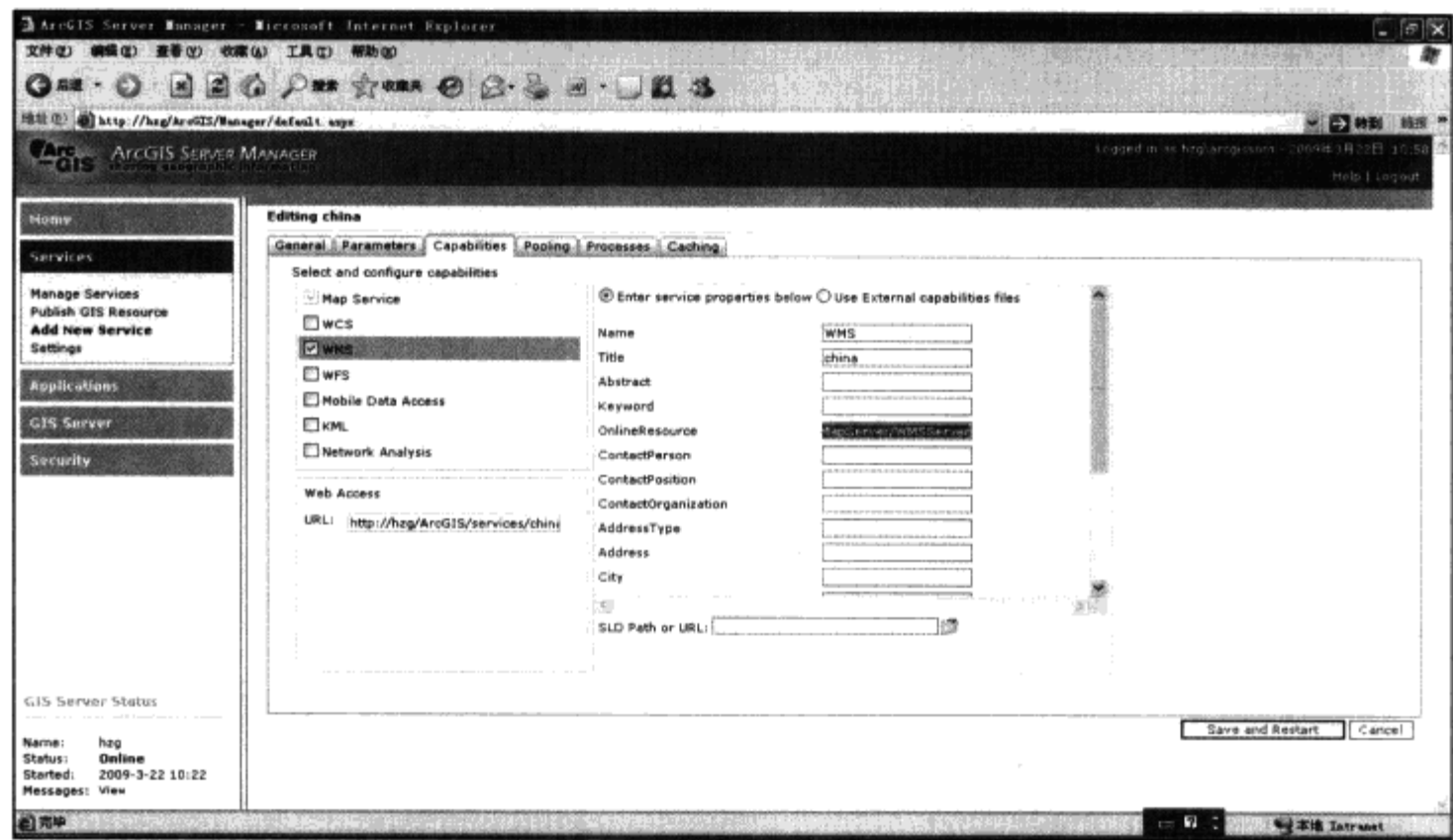


图 4.32 选中“WMS”复选框

ArcGIS Server 中的 WMS Services 有以下 3 个特点：

- 支持 SLD。
- 在 ArcGIS Desktop 和 Explorer 中支持 WMS 1.3.0。
- 在 Web ADF 中支持 WMS 1.3.0。

可以在 ArcCatalog 中查看发布的 WMS 服务，如图 4.33 所示。

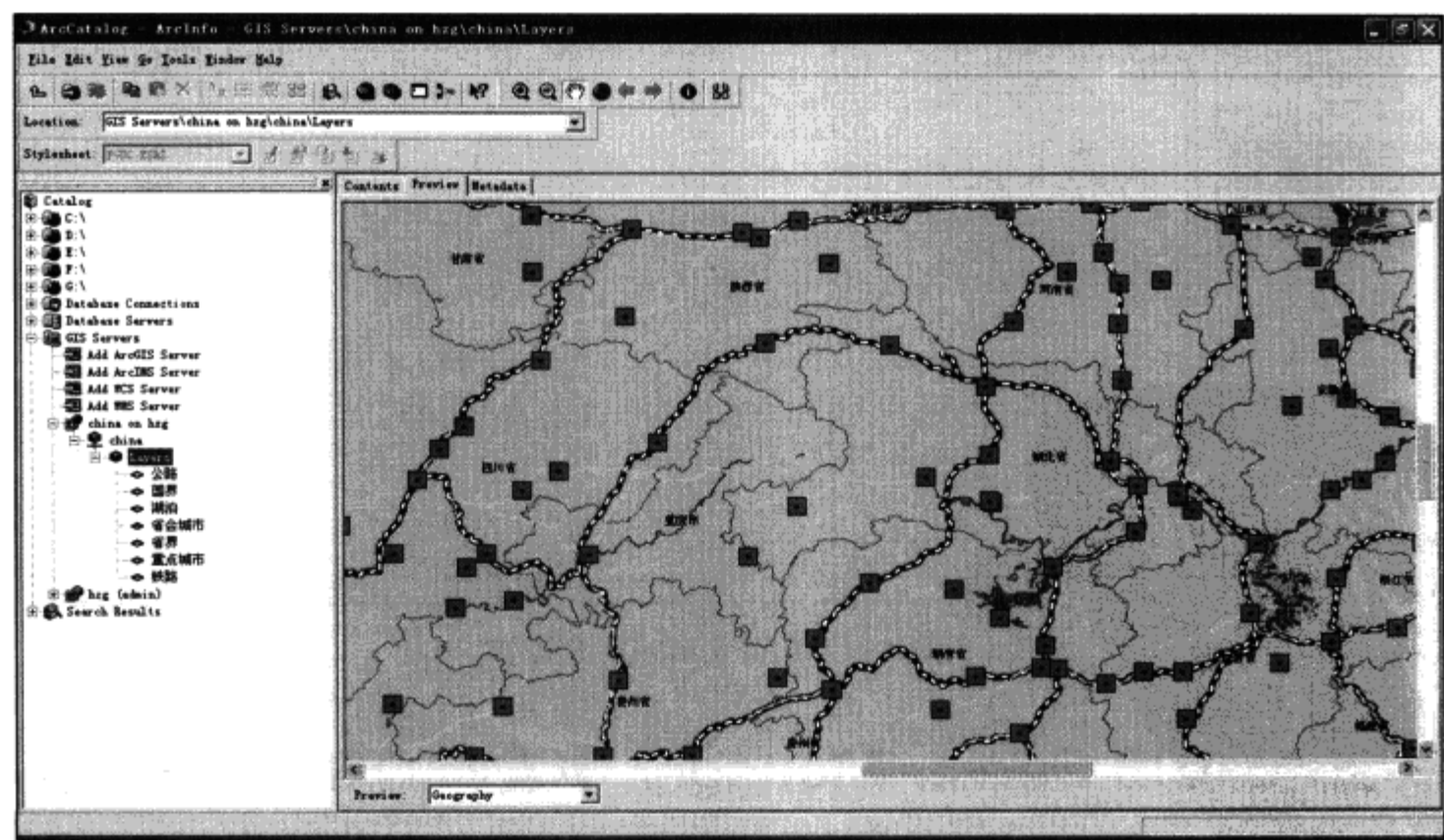


图 4.33 查看发布的 WMS 服务

(2) WFS 是地图要素服务。按照要素类型组织的要素集合, 相当于 ArcGIS 的 Feature Classes, 客户端可以更新数据。发布的步骤与发布 WMS 服务相同, 在发布 WFS 服务时, 选中 WFS 复选框即可。

ArcGIS Server 9.3 中的 WFS 服务有以下特点。

- 采用 WFS 1.1 标准。
- 可以基于地图文档或者 Geodatabase 发布 WFS 服务。
- 对于版本和非版本的 Geodatabase 同样有效。

(3) WCS 是发布 “Converage” (栅格数据) 给用户端的服务, 客户端可以获取数据并直接使用数据进行处理分析工作。

ArcGIS Server 中 WCS 服务有以下特点:

- 将栅格数据发布为 OGC WCS Services。
- 可以基于如下几种服务类型发布栅格数据:
 - Map Services;
 - Geodata Services;
 - Image Services。
- 支持 WCS 1.0.0 和 1.1.0 标准并满足 1.1.1 特性。
- 输出数据包括 GeoTIFF、NITF、HDF、JPEG、JPEG2000、PNG 等数据格式。

(4) KML 是 Google 数据格式的一种服务, 现被纳入 OGC 标准。

ArcGIS Server 中 KML 服务有以下特点:

- 能够将地图输出为 KML 文档格式。
- 服务浏览页面包含 KML 链接, 可直接用 Google earth 或 ArcExplorer 打开。

4.6 小结

本章主要讲述了如何制作地图文档以及如何在 ArcGIS Server 中发布地图服务: 可以用 ArcCatalog 和 ArcGIS Server Manager 工具来发布地图服务。还介绍了在 ArcGIS Server 中发布 OGC 的地图服务。阅读完本章, 读者应该掌握如何利用 ArcGIS 的相关软件来制作地图文档, 并根据需要发布地图服务。

第 5 章

ArcGIS Server开发基础ASP.NET

5.1 ASP.NET 简介

5.1.1 Web 开发技术

第一代 Web 应用程序比较难于开发和管理，在性能和可扩展性方面表现得差强。总体上来讲，早期的 Web 开发技术分为以下两种。

- Command Gateway Interface，俗称 CGI，由服务器调用独立的应用程序。这个开发方式存在的主要问题是占用大量的服务器资源：每一个用户请求需要运行一个独立的应用程序。
- 脚本程序，在服务器端放置一些脚本。脚本文件由浏览器解析，最终以 HTML 呈现在用户面前。这种方式存在的问题是，浏览器不但要解析脚本，还要执行嵌入式代码。与已经编译的程序相比，运行效率不高。

ASP.NET 并不是这两种 Web 开发技术的简单演变，也不是一组让你在服务器上触发应用程序或运行组件的“钩子”，而是一个运行已编译的代码，并由 .NET 的运行管理库（NET Framework）管理的完整的 .NET 应用程序。ASP.NET 能够使用 .NET Framework 的所有功能，与普通运行在 .NET Framework 环境下的应用程序（EXE）一样。其实，ASP.NET 将桌面开发工具和技术拓展到了 Web 开发活动中。

ASP.NET 提供一个更加现代的 Web 开发环境，但以前构建 Web 应用程序的传统观念在 ASP.NET 还能适用。每一个 Web 应用程序都由网页组成，仍然可以使用 JavaScript 和 HTML。但 ASP.NET 与传统的 ASP、JSP 和 PHP 等脚本语言是有很多不同的，主要区别如下：

- ASP.NET 是面向对象的开发方法，是基于事件和控件的架构，可以进行代码封装和复用。
- ASP.NET 支持多种开发语言，如 C#、Basic、J#等开发语言。
- ASP.NET 的性能高，不必每次都编译其页面和组件，而是根据需要才进行编译。

ASP.NET 还包括动态的数据绑定、绘图和更好的安全性，这些都是以前的 Web 开发技术所不能比拟的。

5.1.2 ASP.NET 特点

ASP.NET 的特点主要表现在以下几个方面。

(1) ASP.NET 需要与 .NET Framework 集成应用。

.NET Framework 是由各个功能模块集合在一起的，它将各种各样的类按照顺序划分到对应的命名空间。这种类的使用与 .NET 的其他开发方式（独立的 Windows 应用程序）是相同的，这样用户开发 .NET 的 Web 应用程序与开发 .NET 的 EXE 是一样的。

(2) ASP.NET 是编译执行而不是解释执行。

传统 ASP 程序性能不高的一个原因是浏览器都是通过解释脚本代码来执行的，当用户浏览网页时，位于 Web 服务器上的脚本语言需要逐行解释脚本代码并翻译为机器代码，这个过程比较耗时间，影响了应用程序的执行效率。

ASP.NET 应用程序实际上要经历两个阶段的编译。首先，编写的 C#（或其他语言）被编译成 MSIL 的中间语言（简称 IL），这一步是 .NET 能够做到的与语言无关的关键。从本质上来说，所有的 .NET 开发语言（C#、Basic、J#）都会被编译为 IL 代码，当页面被第一次请求的时候，编译过程会自动进行。

.NET 编译的第二个阶段在这个页面的实际执行时开始，此时，中间代码被编译成本机机器代码。该阶段的编译叫即时编译（just-in-time，简称 jit），其他类型的 .NET 应用程序也会有这样的编译过程，编译过程如图 5.1 所示。

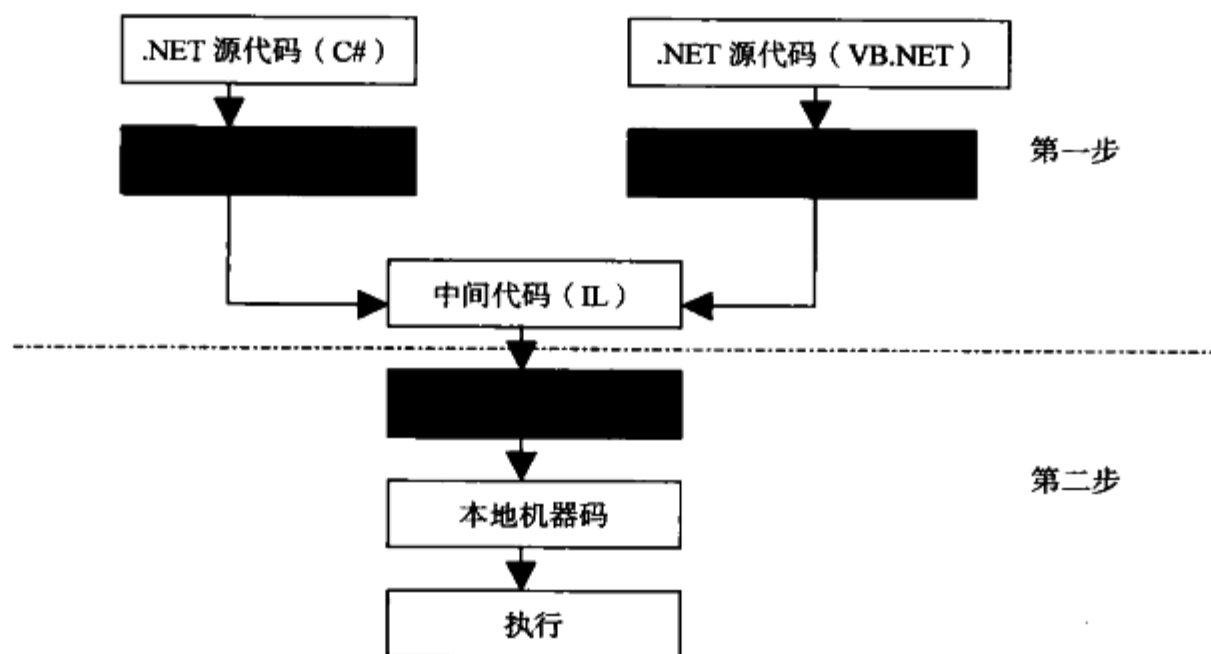


图 5.1 ASP.NET 的编译过程

需要说明的是，并非每一次用户请求网页都会执行第一步的编译过程，网页第一步的编译只是在第一次用户请求时执行，并把编译的中间代码缓存在本机 C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files 的目录下。这样基于 ASP.NET 的应用程序第一运行的时候比较慢，以后就很快了，因为本机已经缓存中间代码。

(3) ASP.NET 支持多种开发语言。

不论采用 C# 还是 VB.NET，或是其他的开发语言，最终应用程序都会被编译成中间语言。

中间语言是所有托管应用程序的基础，是能被托管应用程序识别的语言。下面举例如下：
C#的代码如下：

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

namespace helloworld
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

用 VB.NET 编写的程序如下：

```
Partial Public Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
        Console.WriteLine("Hello World")

    End Sub

End Class
```

经查看，它们编译的中间程序是相同的（使用 ildasm.exe 可查看 .NET 编译的中间程序）：

```
.method family hidebysig instance void Page_Load(object sender,
                                                    class [mscorlib]System.EventArgs e)
cil managed
{
    // Code size      13 (0xd)
    .maxstack 8
    IL_0000: nop
    IL_0001: ldstr      "Hello World!"
    IL_0006: call       void [mscorlib]System.Console::Write(string)
```

```

IL_000b:  nop
IL_000c:  ret
} // end of method _Default::Page_Load

```

这样无论使用 .NET 的哪种开发语言，最终编译的中间语言都是一样的，转化为本机的机器码也是相同的。

(4) ASP.NET 应用程序运行在公共运行语言运行库 (CLR) 内。

.NET Framework 称为托管代码，ASP.NET 是运行在 CLR 内。CLR 有以下特点。

- 自动管理内存和垃圾回收。当应用程序实例化某个对象时，CLR 会自动分配内存空间，当关闭应用程序时，系统开发人员不必清空内存，CLR 会自动回收不再使用的内存。
- 类型安全。当编译程序时，.NET 会在程序中加入描述信息，保证类型不会变成其他的类。
- 结构化的错误处理。.NET 提供结构异常处理，它允许开发人员逻辑、准确地来组织错误处理。
- 多线程。CLR 提供线程池，可以自动地进行多线程的管理。

(5) ASP.NET 是面向对象的。

ASP.NET 中体现面向对象的例子就是服务器控件，这些控件都是封装好的对象，开发人员可以通过编程来操作这些控件。控件在页面生成后，自动生成 HTML 脚本代码，开发人员可以不去理会这些底层的 HTML 代码，只需要控制这些控件对象就可以了。

(6) ASP.NET 与设备和浏览器无关。

对于 Web 开发人员来说，不同公司的浏览器和同一公司不同版本的浏览器，HTML 和 JavaScript 等脚本代码支持有一些区别。对于 ASP.NET 来说，开发人员不必去关心客户浏览器，因为 ASP.NET 控件会自动根据客户浏览器的情况来生成相应的 HTML 代码来适应客户浏览器。

(7) ASP.NET 发布和配置容易。

.NET Framework 提供了相同的核心类库，ASP.NET 应用程序发布的时候，只需要将相应的文件复制到虚拟目录下，当配置用户安全时，只需要编辑 web.config 文件即可，非常方便。

5.1.3 ASP.NET 2.0

微软发布的 ASP.NET 1.0 很快被接受，并成为采用微软技术开发 Web 应用程序的标准。与 ASP.NET 1.0 相比，ASP.NET 2.0 有一些新特性。

- 新的服务器控件。ASP.NET 2.0 引入了许多新的服务器控件，它包含了 40 多个新控件。
- 母板页。它可以重复使用。例如，可以使用母板来确保应用程序都有相同的页眉、页脚或导航控件。这样可以提供应用程序的可维护性，并避免不必要的代码重复。
- 主题和外观。允许开发人员为 Web 控件定义标准的外观。一旦这些样式信息定义完成，开发人员可以将其应用到 Web 页面上保持外观一致。
- 登录管制。新的登录控件提供以认证和授权为基础的访问。
- 数据源控件。数据源控件允许页面与标记中的数据源进行声明性的交互，而不必手工编写等效的存取代码。
- Web 部件。它在一个页面将彼此独立的面板上不同信息组织在一起。
- 配置文件。这个特性允许开发人员不用编写代码就能在数据库中存储用户的特定信息。

5.1.4 ASP.NET 3.5

ASP.NET 2.0 之后，微软以 .NET Framework 3.0 发布了一系列的新技术，最著名的有 WPF (Windows Presentation Foundation, Windows 表现层基础)，但是没有发布 ASP.NET 3.0。ASP.NET 的下一个版本是 ASP.NET 3.5，其新特性如下。

- **LINQ** LINQ (Languages Integrated Query, 语言集成查询) 是一组基于 C# 和 VB.NET 语言的扩展。它允许编写 C# 或 VB.NET 代码以查询数据相同的方式操作内存数据。LINQ 定义了大约 40 个操作符，如 select、from、in、where 等，使用这些操作符可以编写查询语句。

下面是一个简单的 LINQ 查询例子，查询一个 int 数组中小于 5 的数字，并按照大小顺序排列。

```
class Program
{
    static void Main(string[] args)
    {
        int[] arr = new int[] { 8, 5, 89, 3, 56, 4, 1, 58, 56, 2, 98 };
        var m = from n in arr where n < 5 orderby n select n;
        foreach (var n in m)
        {
            Console.WriteLine(n);
        }
        Console.ReadLine();
    }
}
```

LINQ 查询可以方便地对内存中的信息进行查询而不仅仅是外部数据源，为开发人员提供了极大的便利。

- **ASP.NET Ajax** Ajax 是一项客户端快捷编程技术，它使得页面不必触发一次完整的回调就可以调用服务器方法并更新自身的内容。通常 Ajax 页面通过客户端脚本触发一次幕后的异步请求。服务器端接受客户请求后，执行某些代码，返回页面所需的数据，然后客户端获得新数据后再刷新页面。

ASP.NET Ajax 是一项多层的技术，为给普通的 ASP.NET Web 页面增加 Ajax 特性提供了一系列的方法。在底层，可以借助 ASP.NET 编写更强的 JavaScript；在高层上，开发人员可以选用服务器端组件不必编写代码，如 UpdatePanel 等控件。

- **green bits 和 red bits** ASP.NET 3.5 并没有包含完整新版本的 ASP.NET，它只是在 ASP.NET 2.0 的基础上增强了一些新特性。ASP.NET 3.5 没有变化的那一部分叫做 red bits，变化的部分叫做 green bits。需要说明的是，CLR、ASP.NET 引擎以及 ASP.NET 2.0 的所有类库都没有改变，发生变化的是 LINQ 和 ASP.NET Ajax 等方面。

5.1.5 Visual Studio 2008 与 ArcGIS Server 9.3 开发环境

在机器上安装 Visual Studio 2008 和 ArcGIS Server 9.3 后，就可以进行 ArcGIS Server 的开发了。ArcGIS Server 有两种开发模式：定制基于模板开发和自定义开发。

(1) 启动 Visual Studio 2008 开发环境，如图 5.2 所示。

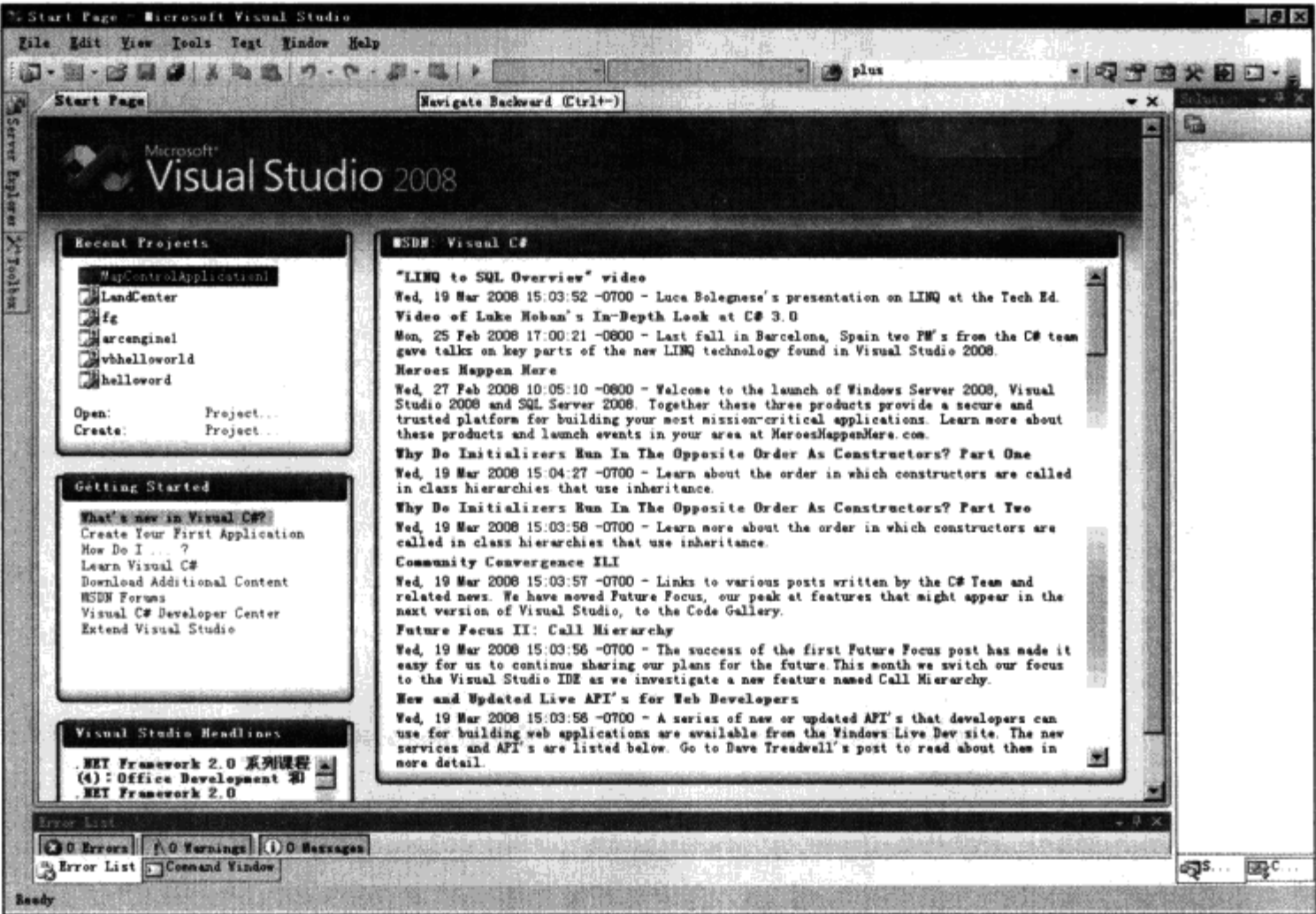


图 5.2 Visual Studio 2008 界面

(2) 单击“File”→“New”→“Web Site”菜单，系统弹出“New Web Site”对话框，如图 5.3 所示。

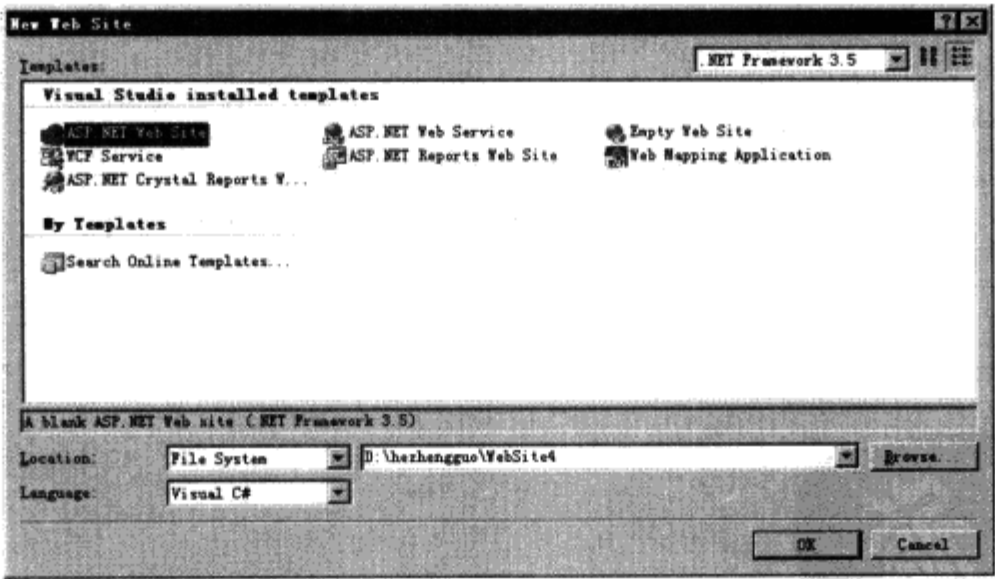


图 5.3 “New Web Site”对话框

(3) 在图 5.3 中，选择“ASP.NET Web Site”选项，单击“OK”按钮进行自定义开发，如图 5.4 所示。

(4) 在图 5.4 中，选择“Web Mapping Application”选项，单击“OK”按钮进行自定义开发，如图 5.5 所示。

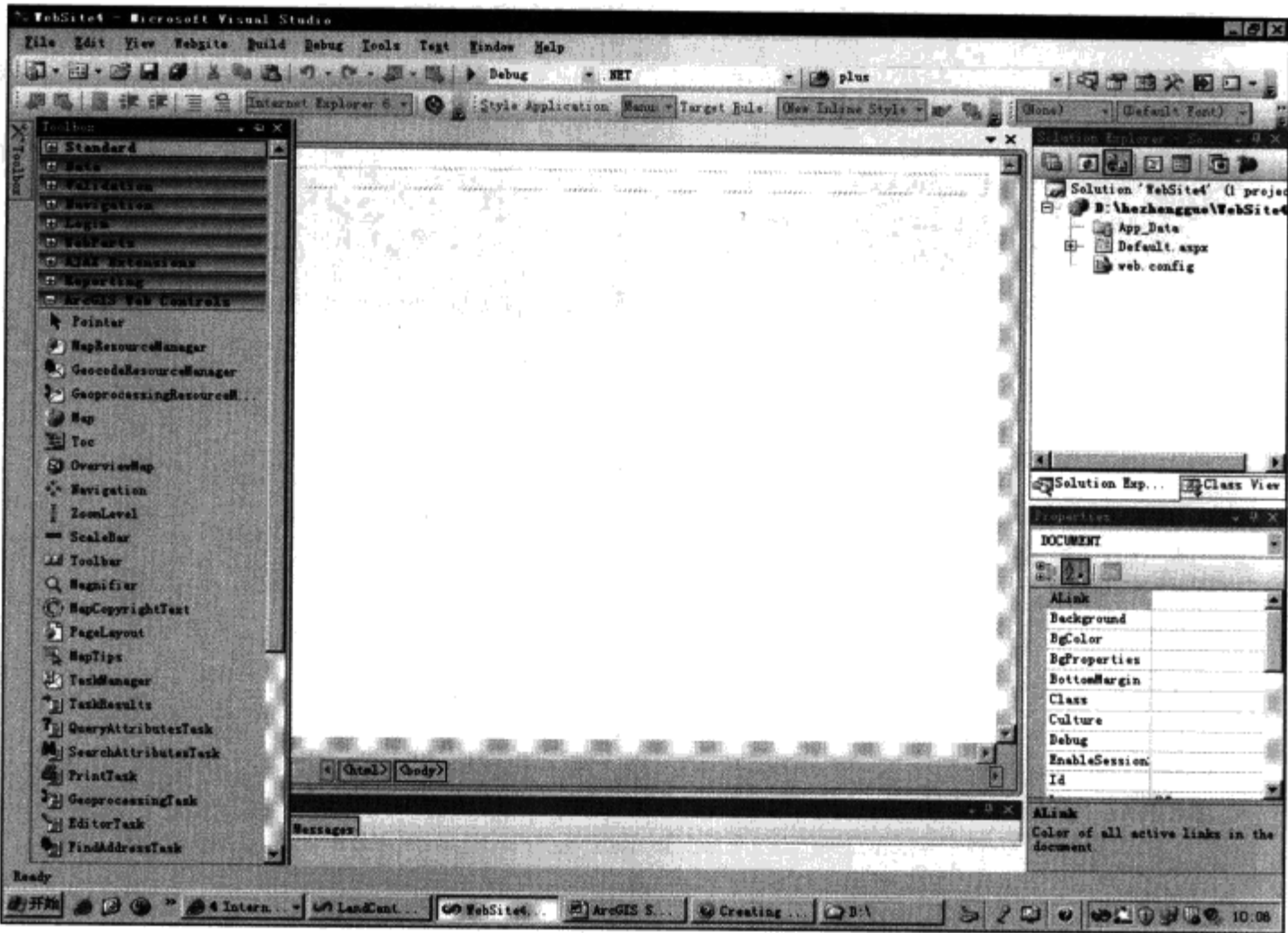


图 5.4 自定义开发界面

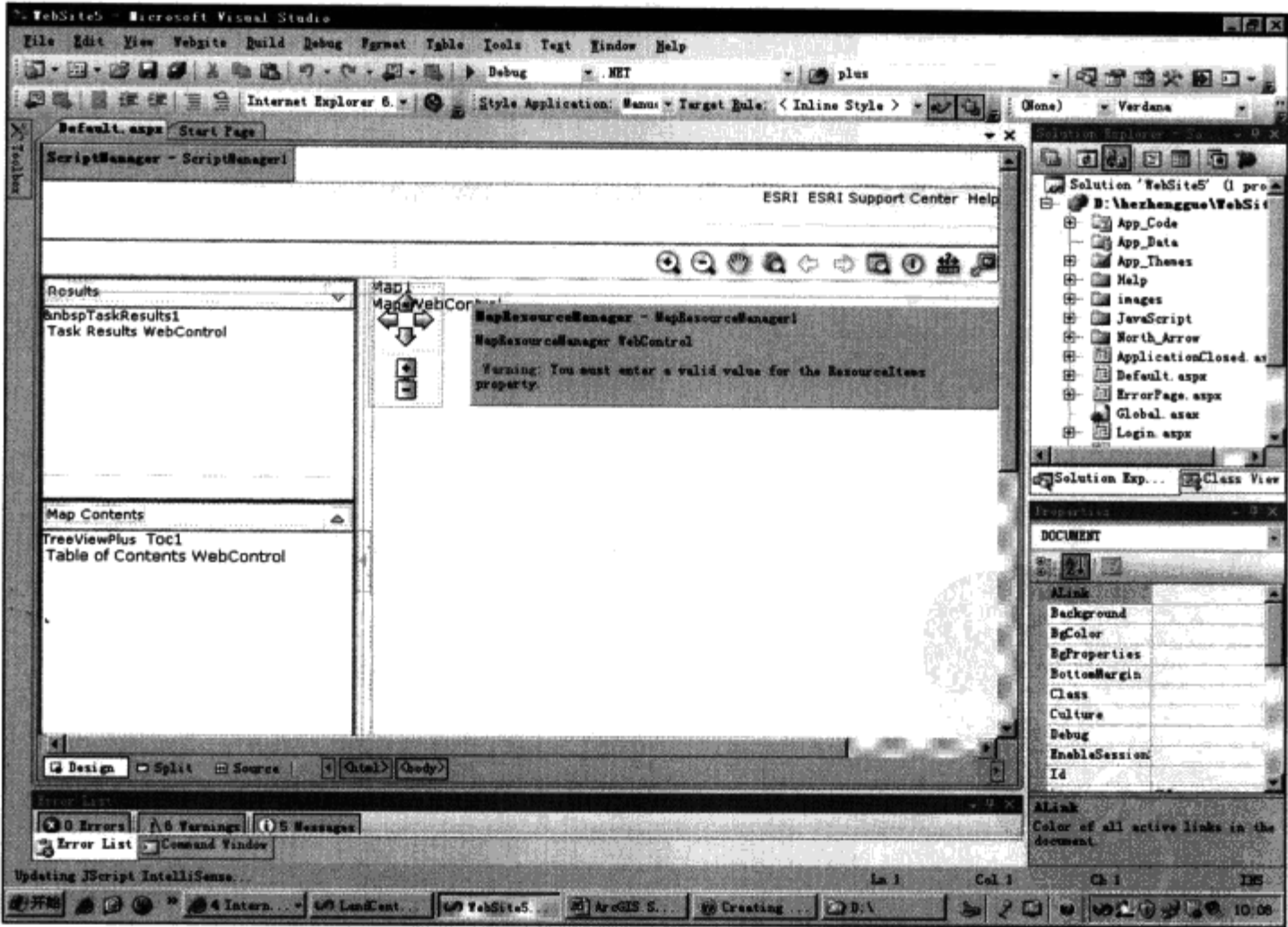


图 5.5 定制模板开发

5.2 JavaScript 和 Ajax 技术

由于 ArcGIS Server 的开发是基于 Web 的开发, Web 页面的表现效果多是由 JavaScript 和 Ajax 来控制, 有必要对 JavaScript 和 Ajax 进行简单的介绍。

5.2.1 JavaScript 本质

JavaScript 是一种广泛使用的 Web 开发语言, 它内置在 Web 浏览器和其他应用程序中, 用户友好、易于使用。使用 JavaScript 能实现的功能如下。

- 验证表单字段。在表单把内容提交给服务器之前, 验证表单中的输入。
- 设置和检索 Web cookie。将用户名、账户或一些习惯性设置的信息永久保存在一个安全的环境中。
- 动态更新页面元素。可以根据用户需要动态地改变某些元素的某些特征, 如放大字体等。
- 隐藏或显示页面元素。
- 在页面上移动页面元素。
- 捕捉用户事件。可以是页面的鼠标或者键盘事件, 并响应相应的事件。
- 与服务器交互。在不离开本页的情况下, 可以与服务器进行连接, 并更新页面。

5.2.2 JavaScript 基本函数

DOM 是文档对象模型 (Document Object Model) 的缩写, 它是供 HTML 和 XML 文件使用的一组 API, 建立网页与脚本或者程序语言之间沟通的桥梁。DOM 中 HTML 文档的各个节点被视为各种类型的节点, 如表 5.1 所示。

表 5.1 HTML 文档常用的节点

接 口	nodeType 值	说 明
Element	1	元素节点
Text	3	文本节点
Document	9	Document
Comment	8	注释
DocumentFragment	11	Document 片段
Attribute	2	节点属性

DOM 数的根节点是一个 Document 对象, 操作 Document 的常用函数, 如表 5.2 所示。

表 5.2 常用 Document 函数

函数方法	说 明
createAttribute	用指定的名字创建新的 Attribute 节点
createComment	用指定的字符串创建新的注释
createElement	用指定的标记名创建新的 Element 节点
createTextNode	用指定的文本创建新的 TextNode 节点
getElementById	返回文档中具有指定 ID 属性的 Element 节点
getElementsByTagName	返回文档中具有指定标记名的所有 Element 节点

在 DOM 模型中，操作 Element 的属性动态地改变网页，常用函数如表 5.3 所示。

表 5.3 常用 Element 函数

函数方法	说 明
getAttributeNode	以 Attribute 节点的形式返回指定属性的值
getElementsByTabName	返回一个 Node 数据，包含指定标记名的所有 Element 节点的子孙节点，其顺序为文档中出现的顺序
hasAttribute	如果该元素具有指定文字的的属性，则返回 True，反之则返回 False
removeAttribute	从元素中删除指定的属性
removeAttibuteNode	从元素的列表中删除指定属性的 Attribute 节点
setAttribute	把指定的属性设置为指定的字符串值，如果该属性不存在则增加一个新属性值
setAttributeNode	把指定的 Attribute 节点添加到该元素的属性列表中

5.2.3 理解 Ajax

传统的 Web 应用程序采用同步交互过程，这种情况下，用户首先向 Web 服务器发送一个请求，然后 Web 服务器根据用户请求的内容，执行相应的任务，并向用户返回结果，如图 5.6 所示。

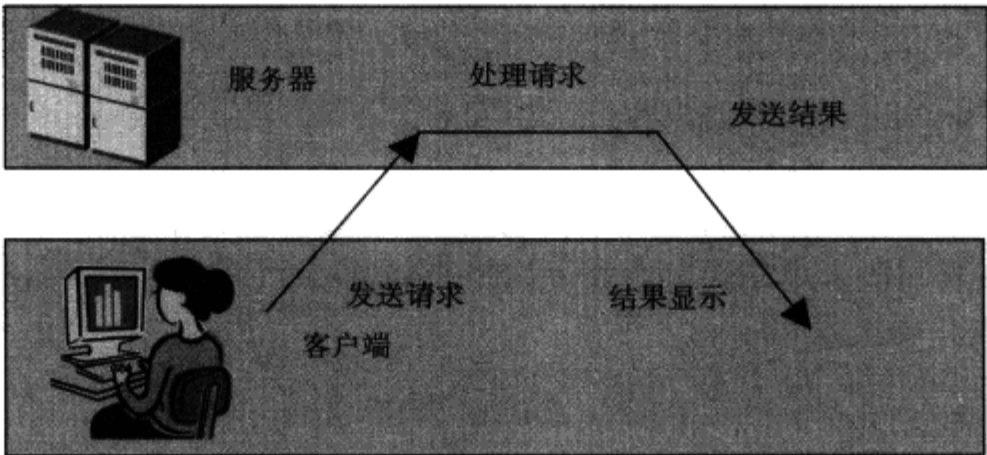


图 5.6 传统 Web 请求过程

这样在服务器处理用户请求的时候，用户只能等待，此时浏览器显示的页面是空白的。此外，当用户在某些时候仅需要改变页面中某部分的数据时，也必须刷新整个页面。

与传统的 Web 应用不同，Ajax 采用异步交互的方式。它在用户与服务器之间引入中间媒介（Ajax 引擎），负责转发用户界面和服务端之间的交互。Ajax 引擎允许用户和应用系统之间的交互以异步的方式进行，独立于用户与 Web 服务器之间的交互。这样数据编辑、页面导航不再需要刷新整个页面。用户请求过程如图 5.7 所示。

XMLHttpRequest 对象是 Ajax 的奠基石。XMLHttpRequest 让用户能够以文本的形式向服务器异步发送请求并以文本的形式获得结果。

（1）发送请求。

使用 XMLHttpRequest 可以用两种方法 open 和 send 来发送请求，如：

xmlRequest.open ("GET", "http://192.168.1.1")。

（2）处理响应。

发送请求后，如何处理请求后的响应呢？只需要在 onreadystatechange 属性附加一个事件处理程序即可，如：

xmlRequest.onreadystatechange = updatePage。

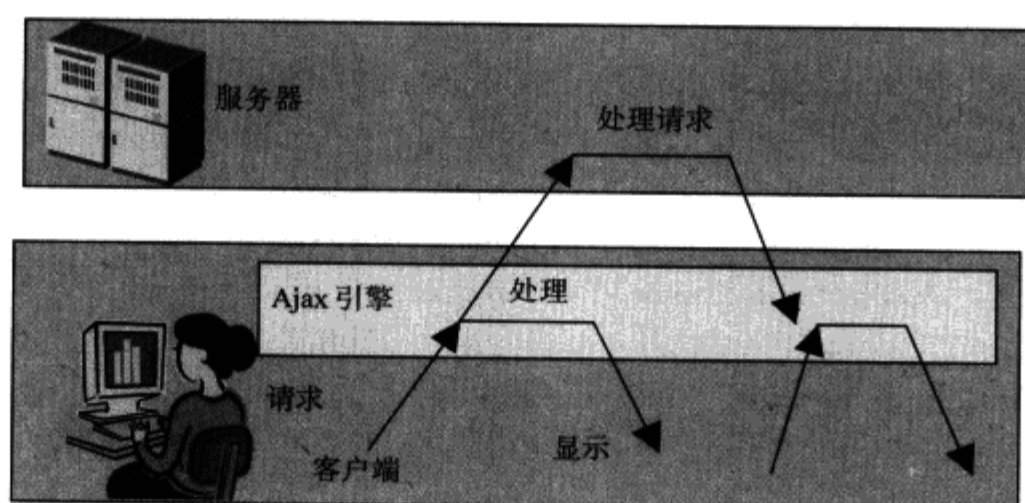


图 5.7 Ajax 异步请求过程

5.2.4 在客户端回调中使用 Ajax

客户端回调让开发人员能够刷新页面的数据部分而不需要刷新整个页面，甚至不需要使用 XMLHttpRequest 对象的脚本代码。在 .NET 环境下实现回调，只需要实现 ICallbackEventHandler 接口的类，ICallbackEventHandler 接口定义了两个方法：RaiseCallbackEvent() 和 GetCallbackResult()。RaiseCallbackEvent() 以字符串参数的形式得到浏览器事件，它首先被触发，接着 GetCallbackResult() 触发并把结果返回给页面。

```
<%@ page language="C#" CodeFile="Default.aspx.cs" Inherits="Default_aspx" %>
<html>
<head>
<title>Server Time</title>
<script language="javascript">

function GetServerTime()
{
    var message = '';
    var context = '';
    <%=sCallBackFunctionInvocation%>
}
function ShowServerTime(timeMessage, context) {
    alert('现在服务器上的时间是:\n' + timeMessage); }
</script> </head> <body>
<form id="MainForm" runat="server">
<input type="button" value="得到服务器端时间" onclick="GetServerTime();" /></form>
</body>
</html>
服务器端:
using System;
using System.Web.UI;
public partial class Default_aspx : Page, ICallbackEventHandler
{
    //一定要实现 IcallbackEventHandler 接口
    public string sCallBackFunctionInvocation;
    void Page_Load(object sender, System.EventArgs e)
```



```
{
    sCallbackFunctionInvocation = Page.ClientScript.GetCallbackEventReference
(this, "message", "ShowServerTime", "context");
}

public void RaiseCallbackEvent(string eventArgument)
{
}

public string GetCallbackResult()
{
    return DateTime.Now.ToString();
}
}
```

5.3 ASP.NET Ajax

如果只用纯粹的 JavaScript,开发人员必须弥补 ASP.NET 服务器端抽象和局限的 HTML DOM 之间的鸿沟,没有 Visual Studio 的智能提示和调式工具,编写正确的代码和处理错误语法都将很困难,而 ASP.NET Ajax 能够很方便地构建 Ajax 风格的页面。

5.3.1 ASP.NET Ajax 简介

ASP.NET Ajax 由两部分构成:客户端部分和服务端部分。客户端部分是一组 JavaScript 库,它建立用于开发 ASP.NET Ajax 页面的基础;服务器端部分包括那些客户端脚本的控件和组件。

1. 客户端的 ASP.NET Ajax:脚本库

ASP.NET Ajax 的客户端部分依赖于小组 JavaScript 文件。部署 ASP.NET Ajax 的方法有两种:构建 ASP.NET 3.5 的应用程序和创建的非 ASP.NET 的应用程序的 HTML 页面添加 Microsoft Ajax Library 的引用。基于 ASP.NET 3.5 的应用程序, Ajax 的 JavaScript 被嵌入 System.Web.Extensions.dll 程序集里面,在应用程序需要的时候再调用;在 HTML 页面添加 Ajax 库的引用,是 Ajax 库已封装好异步刷新的 JavaScript 的程序。

下载 Microsoft Ajax Library,就会发现 ASP.NET Ajax 其实使用了 3 个核心的 JavaScript 文件——Microsoft Ajax.js、Microsoft Ajax WebForms.js 和 Microsoft Ajax Timers.js。

在 ASP.NET 3.5 应用程序的 web.config 文件中,可以发现一个映射,把对 ScriptResource.axd 的请求链接到 System.Handlers.ScriptResourceHandler 类。

2. 服务器的 ASP.NET Ajax:ScriptManager

如果开发人员需要在每个 ASP.NET Ajax 的页面都输入指向脚本的 URL 的话,会让他们觉得非常麻烦。在 ASP.NET 3.5 中,开发人员可以省去这样的麻烦,ASP.NET 3.5 提供了一个 ScriptManager 的控件。

ScriptManager 是服务器端 ASP.NET Ajax 模型的中枢,它在页面运行的时候是不可见的。但是,它执行一个主要的任务:呈现到 ASP.NET Ajax JavaScript 库的链接。要在页面上使用异步刷新的技术需要在页面上添加一个 ScriptManager,需要从工具箱的 Ajax Extensions 页把它拖

到页面上。ScriptManager 在 aspx 文件中的声明如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

5.3.2 服务器回调

在 ASP.NET Ajax 中回调总是通过一个单独的服务器端方法来实现。

1. ASP.NET Ajax 中的 Web 服务

通过 ASP.NET Ajax 执行服务器回调时，客户端的 JavaScript 代码调用一个服务端的 Web 服务（Web 服务是一个或多个服务器方法的集合，它可以由远程客户调用）。客户端在 HTTP 上发送一个消息请求调用 Web 服务，这和执行一个 Web 页面回调的过程相似，除了请求的主体外是要传递给方法的参数。然后 ASP.NET 创建一个 Web 服务对象，运行对应 Web 方法里的代码，返回结果并释放 Web 服务对象。请求和响应消息的格式可以变化。传统上，称为 SOAP——一个基于 XML 的标准，但在 ASP.NET Ajax 里，它们是基于更轻型的文本的，被称为 JSON（JavaScript Object Notation，JavaScript 对象表示法），这主要是出于浏览器兼容的原因。

虽然 ASP.NET Ajax 回调机制使用 Web 服务，但这是一个特殊的实现。如果熟悉 Web 服务，就会发现 ASP.NET Ajax 强加了一些其他的限制。首先，Web 页面不能调用非 ASP.NET Ajax Web 对象，这是因为它们不支持 ASP.NET Ajax 所使用的简单 JSON 模型。其次，Web 页面不能调用其他域（在其他 Web 服务器上）的 Web 服务，因为大多数情况下浏览器都禁止这类跨域使用 XMLHttpRequest 对象，以防止跨站点的脚本攻击。

虽然 ASP.NET Ajax 页面以一种特殊的方式使用 Web 服务，但定义 Web 服务的方式都是一样的。和其他所有的 ASP.NET Web 服务一样，ASP.NET Ajax 使用的 Web 服务由两部分组成：一个 asmx 文件，它作为 Web 服务的端点，还有一个包含真正 C# 代码的 CS 文件。开发人员需要把这些文件添加到包含将要使用 Web 服务的 ASP.NET Ajax 页面的网站。

在 Visual Studio 2008 中创建 Web 服务的步骤如下：

（1）运行 Visual Studio 2008 后，选择“New”菜单的“Web Site”选项，系统弹出“New Web Site”对话框，如图 5.8 所示。

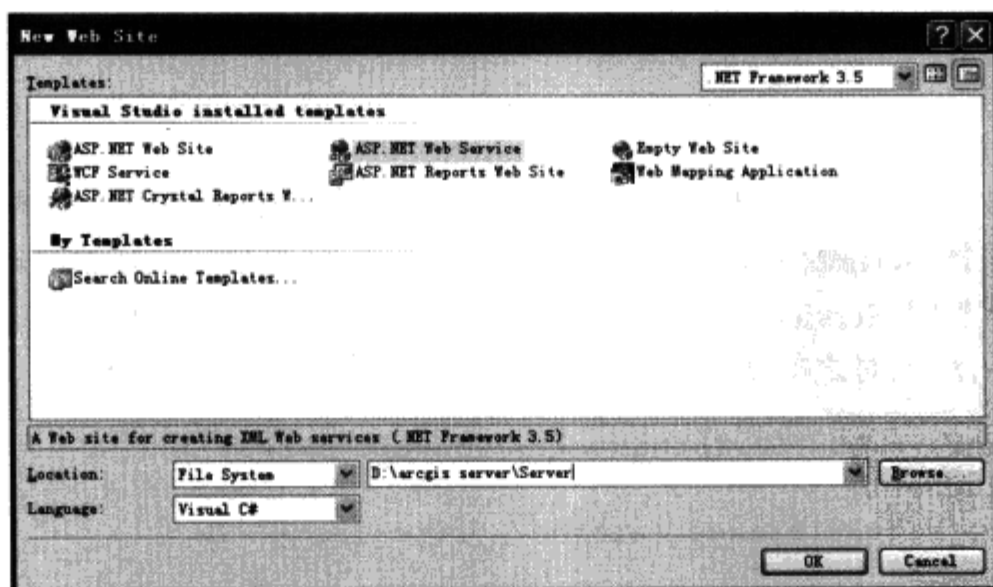


图 5.8 “New Web Site”对话框

（2）选择“ASP.NET Web Service”选项，在 Location 文本框输入对应的路径，单击“OK”

按钮，就创建了一个 Web 服务，如图 5.9 所示。

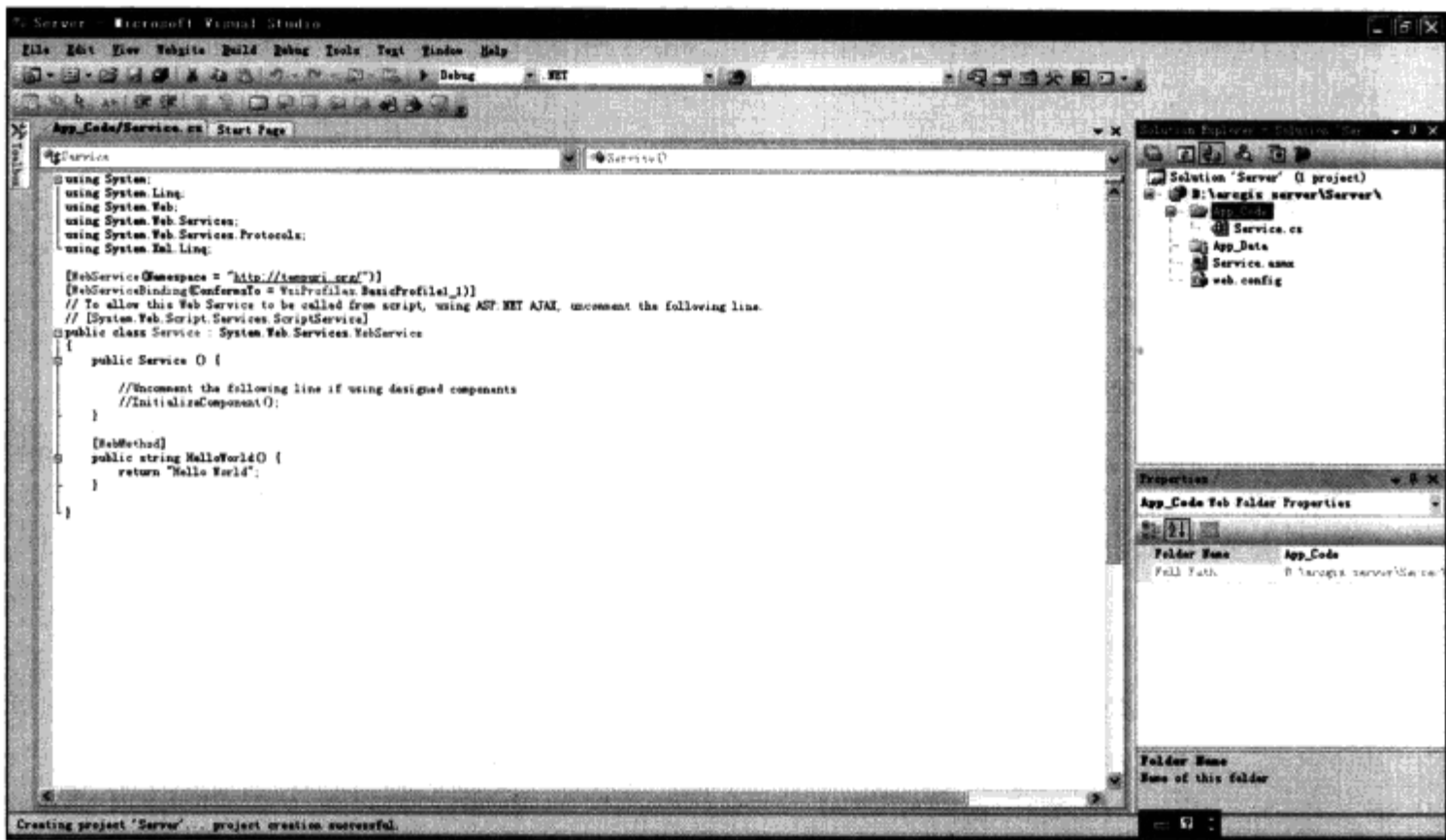


图 5.9 创建一个 Web 服务

asmx 文件将被放到 Web 应用程序目录，而对应的 CS 文件放到 App_Code 文件夹中，这样它们能够被自动编译。

(3) 创建 Web 方法。

创建好 Web 服务后，就为在 Web 服务中编写代码准备好了条件。每个 Web 服务都有一个或多个用 WebMethod 特性标注的方法。WebMethod 特性使方法能够被远程调用。如果添加的方法没有标记 WebMethod 特性，服务器端的代码还是可以使用它，但在客户端的 JavaScript 就不能直接调用它了。创建 HelloWorld 方法的程序如下：

```
[WebMethod]
public string HelloWorld() {
    return "Hello World";
}
```

(4) 调用 Web 服务。

创建了 Web 服务之后，下一步是配置页面让页面知道 Web 服务中可供调用的方法。这时，需要给页面添加一个 ScriptManager 对象，然后，在标签页给 ScriptManager 对象添加一个 <Services> 节。这个节用 ServiceReference 元素列出了页面使用的所有服务以及它们的位置。下面的代码演示了如何添加对之前演示的 TerritoriesService.asmx 文件的应用：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <Services>
        <asp:ServiceReference Path="~/Service.asmx" />
    </Services>
</asp:ScriptManager>
```

ScriptManager 生成一个 JavaScript 代理，开发人员可以用它实现调用。对于目前这个示例，

可以实现调用。用如下的 JavaScript 代码可以实现对 Web 服务的 Web 方法调用。

```
<script language = "javascript" type = "text/javascript">
    //最后一个参数为回调函数名称
    Service.Hello(updatepage);
    //其中 result 即为返回的结果
    function updatepage(result)
    {
        alert(result);
    }
</script>
```

客户端的 Web 服务调用是异步的，因此开发人员除了需要提供原始的 Web 方法参数外，还要提供另外一个参数，它定义了接收到结果时要调用的函数。

2. Web 服务代理

为了让客户端 JavaScript 代码能够调用 Web 服务，web.config 文件需要进行另外一些配置，但是如果使用 Visual Studio 2008 开发 JavaScript 应用程序，这些设置自动添加到 web.config 文件中。

在 web.config 文件中，asmx 被映射到一个名为 ScriptHandlerFactory 的 HTTP 处理程序：

```
<httpHandlers>
    <remove verb="*" path="*.asmx"/>
    <add verb="*" path="*.asmx" validate="false" type="System.Web.Script.
Services.ScriptHandlerFactory, System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
    <add verb="*" path="*_AppService.axd" validate="false" type="System.Web.
Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
    <add verb="GET,HEAD" path="ScriptResource.axd" type="System.Web.Handlers.
ScriptResourceHandler, System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" validate="false"/>
</httpHandlers>
<httpModules>
```

ScriptHandlerFactory 除了支持普通 Web 服务调用外，还增加对 JSON 请求的支持。

3. ASP.NET Ajax 应用程序服务

能够在 ASP.NET Ajax 里创建和调用自定义 Web 服务当然是一项很有价值的技术。开发人员可以调用服务器端数据库的其他数据，触发服务器端任务，获得用户特定的会话信息。在能够使用这些服务之前，需要在 web.config 文件中的 <system.web.extensions> 节开启它们。

5.3.3 ASP.NET Ajax 服务器控件

ASP.NET Ajax 的 Web 服务特性为客户端代码访问服务器提供了一个重要的窗口。但是，这要求开发人员承担大多数的困难任务。开发人员必须小心创建正确的 Web 方法，在合适的时机调用它们，适当地更新页面，除了 JavaScript 其他什么也不能用，这个过程非常复杂。不过，Visual Studio 2008 提供一些 ASP.NET Ajax 控件，省去了开发人员的麻烦。

1. 使用 UpdatePanel 的局部呈现

UpdatePanel 是一个方便的控件，它让开发人员能够利用服务器端逻辑处理普通页面并让它以无刷新的 Ajax 风格刷新自己。其核心思想就是把一个 Web 页面分解为多个独立的区域，其中每个区域都包含在一个不可见的 UpdatePanel 里。当 UpdatePanel 中发生本来要触发一次回发的事件时，UpdatePanel 截取该事件代之以执行一次异步回调。下面是发生过程的一个实例。

aspx 文件示例程序：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="
_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            <asp:ScriptManager ID="ScriptManager1" runat="server">

            </asp:ScriptManager>
            <asp:UpdatePanel ID="UpdatePanel1" runat="server">
                <ContentTemplate>
                    <asp:ListBox ID="ListBox1" runat="server"></asp:ListBox>
                    <asp:Button ID="Button1" runat="server" Text=" 添加 " onclick=
"Button1_Click" />
                </ContentTemplate>
            </asp:UpdatePanel>
            <script language="javascript" type="text/javascript">
                //alert(Service.HelloWorld(OnRequestComplete,onerror));
            </script>
        </div>
    </form>
</body>
</html>
```

C#示例程序：

```
public partial class _Default : System.Web.UI.Page
{

    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
```



```

        int i = ListBox1.Items.Count;
        i++;
        ListBox1.Items.Add(i.ToString());
    }
}

```

该段程序能够自动实现 ListBox 的异步刷新。

2. 使用 Timer 的定时刷新

UpdatePanel 需要进行某一操作后, 页面才能刷新。使用 Timer 时无须用户操作而定时刷新。Timer 控件的使用非常简单, 只需要把它加入到页面上, 然后把它的 Interval 设置为刷新的时间, Timer 控件将会在一次固定的时间内刷新。

Timer 控件会引发服务器端的 Tick 事件, 可以在事件中处理更新。但是, Tick 事件不是强制刷新的, 因为定时器触发时会执行完整的页面生命周期。换句话说, 就是可以响应其他页面和控件事件。Timer 控件适合只需要定期刷新页面的部分内容, 同时刷新不是特别明显, 没有闪烁, 用户不易觉察到。

aspx 文件示例程序:

```

<asp:Timer ID="Timer1" runat="server" ontick="Timer1_Tick" Interval="3000">
</asp:Timer>

<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    </ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID = "Timer1" EventName = "Tick" />
    </Triggers>
</asp:UpdatePanel>

```

C#示例程序:

```

protected void Timer1_Tick(object sender, EventArgs e)
{
    int tickCount = 0;
    if (ViewState["TickCount"] != null)
    {
        tickCount = (int)ViewState["TickCount"];
    }
    tickCount++;
    ViewState["TickCount"] = tickCount;
    TextBox1.Text = tickCount.ToString();
    if (tickCount > 20) Timer1.Enabled = false; //关闭触发
}

```

3. 使用 UpdateProgress 的长时间更新

ASP.NET Ajax 还包含一个 UpdateProgress 控件, 它和 UpdatePanel 控件一起使用, 可以显示

页面更新的进度。一般比较费时的更新使用 UpdateProgress 控件，在异步请求开始的时候显示某些内容，在异步请求结束的时候这些内容自动消失。

aspx 文件示例代码：

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">
    <title>UpdateProgress</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <div>

            <asp:UpdatePanel ID="UpdatePanel1" runat="server">
                <ContentTemplate>
                    <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="
自动刷新" /><br />
                    <asp:Label ID="LabelTime" runat="server" Text="Label"></asp:Label>

                </ContentTemplate>
            </asp:UpdatePanel> <br />

            <asp:UpdateProgress ID="UpdateProgress1" runat="server"
                AssociatedUpdatePanelID="UpdatePanel1" DisplayAfter="5">
                <ProgressTemplate>
                <div style="font-size:xx-large">
                    更新到...
                </div>
                </ProgressTemplate>
            </asp:UpdateProgress>

        </div>

    </form>
</body>
</html>
```

C#示例程序：

```
protected void Button1_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(TimeSpan.FromSeconds(20));
    LabelTime.Text = DateTime.Now.ToLongTimeString();
}
```

当回调完成后，UpdateProgress 自动消失而 UpdatePanel 被更新。UpdateProgress 控件还支持取消操作，用户单击取消异步调用后，UpdateProgress 的内容将自动消失。

5.3.4 深入客户端库

UpdatePanel、Timer、UpdateProgress 等控件都封装好了 Ajax 框架，并没有很深入了解客户端模型。本小节简单介绍一下 Web 客户端模型以及自定义控件的开发。

1. 理解客户端模型

ASP.NET Ajax 的基础框架是一些客户端的 JavaScript 库。ASP.NET Ajax 的架构相当完整，这里只是简单地介绍一下。

ASP.NET Ajax 包括服务器端和客户端两部分，UpdatePanel、UpdateProgress 等控件都属于服务器端控件。客户端则提供了一整套的 Client-Script Library，让开发人员可以轻易地通过简化且扩充功能后的 JavaScript 来调用服务器端所撰写的 Web 服务，并且分析该 Web 服务所返回的 XML 数据。这组 Library，还提供了让开发人员可以撰写具有面向对象的 JavaScript 的能力，并且可以让开发人员轻易撰写出具有绚丽互动效果的前端页面，并与 ASP.NET 后端代码完美结合，如图 5.10 所示。

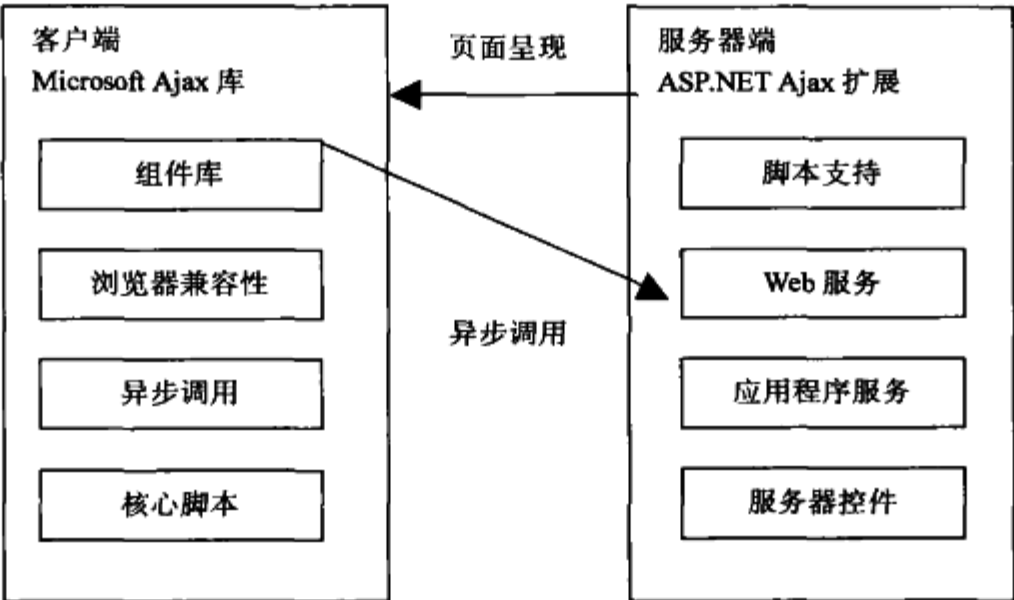


图 5.10 ASP.NET Ajax 基础架构

2. JavaScript 的面向对象编程

JavaScript 并非一门真正面向对象的语言，但经常被描述为基于对象的语言，因为它提供创建的对象。

在 JavaScript 中创建对象的方式有以下两种。

- **闭包** 封装了类的函数。页面运行的时候不能运行闭包函数，而是运行它的嵌入函数，这些函数是类的方法。示例程序如下：

```
<script language = "javascript" type = "text/javascript">
function Student(id,name)
{
    var _id = id ;
    var _strName = name;
    this.set_Id = function(id)
    {
        _id = id;
    }
}
```

```
    }
    this.get_Id = function()
    {
        return _id;
    }
    this.set_Name(name)
    {
        _strName = name;
    }
    this.get_Name = function()
    {
        return _strName;
    }
}
var stu = new Student(1,"johnhee");
</script>
```

- **原型** 利用 JavaScript 对象都拥有的 prototype 属性，它暴露对象的公共接口，如果对象增加可调用的方法，那么必须给 prototype 属性赋予新的属性。
-

```
<script language="javascript" type="text/javascript">
    Student = function(id,name)
    {
        this._id = id;
        this._strName = name;
    }
    Student.prototype.set_Id = function(id)
    {
        this._id = id;
    }
    Student.prototype.get_Id = function()
    {
        return this._id;
    }
    Student.prototype.set_Name = function(name)
    {
        this._strName = name;
    }
    Student.prototype.get_Name = function()
    {
        return this._strName;
    }
    var stu = new Student(1,"johnhee");
    alert(stu.get_Name());
</script>
```

本质上，闭包为对象创建专门的成员，每个闭包创建一个新对象。但使用原型时，对象只创建和配置一次，然后复制对象。所以原型在某些浏览器上性能会得到提高。

JavaScript 还可以注册类和命名空间，例如，`Type.registerNamespace("test");` 注册一个 test 的命名空间，使用 `registerClass` 方法可以注册对象类，在此不再详细介绍了。

3. Web-Page 框架

ASP.NET Ajax 客户端使用多层设计。在底层是一组允许 JavaScript 增强以及一组对核心 JavaScript 基本数据类的扩展。ASP.NET Ajax 还有一个核心客户端类和客户端页面模型。

(1) Application 类。

Web-Page 模型的入口是 Sys.Application 类，在浏览器中加载 ASP.NET Ajax 中页面之后，ScriptManager 会创建一个 Sys.Application 实例，并帮助它完成客户端的工作。

Application 对象引发两个主要的事件。load 事件在页面第一次被处理以及触发异步调用后触发，unload 事情在用户离开当前页面浏览其他新页面时发生。

```
function pageLoad()  
{  
    alert("load");  
}  
function pageUnload()  
{  
    alert("unload");  
}
```

Application 类还提供一个 init 事件，它在所有脚本都已经被加载但对象还没有创建时发生。init 事件只在页面第一次处理时发生一次，异步回调后不发生。ASP.NET Ajax 组件响应 init 事件创建客户端组件。

(2) PageRequestManager 类。

如果页面支持异步刷新，并在服务器端放置一个或多个 UpdatePanel 控件，PageRequestManager 就会被创建。

PageRequestManager 引发一组事件（如表 5.4 所示），开发人员可以在客户端 JavaScript 代码中响应它们。

表 5.4 PageRequestManager 引发事件

事 件	描 述
initializeRequest	在异步调用开发发生前发生。此事件可以通过传递到事件处理程序的 Sys.WebForms.InitialixeRequestEventArgs 对象的 Cancel 属性取消回调
beginRequest	在异步请求之前调用，发生在 initializeRequest 事件之后
pageLoading	在异步调用请求接收到之后页面被更新之前发生
pageLoaded	在异步调用请求接收之后页面更新之后发生
endRequest	在异步响应被处理后或在处理响应过程发生错误时发生

5.3.5 控件扩展器

ASP.NET 没有包括任何控件扩展器，但是 ASP.NET Ajax 控件工具包括了。ASP.NET Ajax 包括大量的控件和控件扩展器，而且这些控件都是开源的，免费提供源代码。要获取 ASP.NET Ajax 工具包，可访问 <http://ajax.asp.net/ajaxtoolkit> 网站，找到相关的下载地址。在下载页面，可以根据开发人员使用的 ASP.NET 版本找到相应的下载选项，下载解压缩后，包含一个名为 ControlToolkit.dll 的程序集及用于支持本地化版本的小程序。ControlToolkit.dll 中为开发人员提供扩展的控件，并有引用程序示例，有关扩展控件的使用，本文不再详述。

5.4 ArcGIS Server Web ADF 中的 Ajax

Web ADF 是 ArcGIS Server 专门用于 Web 应用程序开发的框架,有 .NET 和 Java 两种开发环境。ADF 为基于 Web 开发 GIS 提供了丰富的控件和组件, ArcGIS Server .NET Web ADF 在 ASP.NET 2.0/3.5 实现了 ADF Callback 框架,实现了对包含有 ADF 组件页面的异步刷新。本节主要讨论 Web ADF 在 .NET 中的回调过程。

5.4.1 .NET ADF 中 Ajax 的调用过程

ASP.NET 2.0/3.5 回调事件中,服务器处理完成并返回浏览器,JavaScript 解析返回值并利用浏览器内嵌技术动态更新页面。为支持 ADF 的异步调用,ADF Callback 框架提供了客户端 JavaScript 库,JavaScript 函数库包括众多 JS 文件。ADF 组件预先在服务器封装 JavaScript 函数库,应用程序运行时,客户端初始化加载页面,JavaScript 代码嵌入浏览器,一旦用户触发操作,JavaScript 代码将生成 XMLHttpRequest 对象与服务器进行异步通信,响应返回时 JavaScript 代码解析字符串并调用方法动态更新网页。

.NET ADF 提供一个比较统一的框架,把底层的一些函数进行了包装,提供统一的入口和模式。.NET ADF 提供了很多控件,这些控件都不是简单几条 JavaScript 语句就能够完成的,如果这些语句都需要开发人员自己来编写的话,开发人员会觉得非常复杂,使用 .NET ADF 的控件就可以免去开发人员在这方面的烦恼,开发人员只需要按照相应的格式把刷新请求返回客户端的相应函数,ADF 提供的各种控件刷新 JavaScripts 就能帮开发人员完成相应的刷新任务。

开发基于 Web ADF 的应用程序,如果要实现异步调用,可以选用 Web ADF 中提供的工具条等组件,也可以选择 ASP.NET 控件如 UpdatePanel,或者两种机制相结合。这两种机制实现异步调用是一样的,都统一于 ASP.NET Ajax 机制。

客户端页面加载初始化后,浏览器一旦触发回调事件,脚本函数负责处理回调信息及相关的上下文,并调用 ASP.NET 2.0/3.5 中 WebForm_DoCallback 函数将回调请求返回服务器。其实质就是利用 Microsoft.XMLHTTP 组件创建一个新的 HttpRequest 对象,并将客户端请求异步发送回服务器,其格式如下:

```
WebForm_DoCallback('Map1',argument,processCallbackResult,context,postBackError,false)
```

客户端回调请求到达后,服务器初始化并调用继承 ICallbackEventHandler 接口的 ADF 组件或者 Page 的 RaiseCallbackEvent() 方法来获取回调请求字符串,然后定义 RaiseCallbackEvent() 方法处理回调请求,并将不同组件处理结果分别存入 CallbackResults 对象,处理结果由 GetCallbackResult() 返回。

ADF Callback 框架中提供 processCallbackResult 方法处理返回客户端的回调响应,在回调响应返回客户端后,processCallbackResult 函数根据事件参数解析客户端回调响应字符串并调用其他 ADF JavaScript 函数库的其他方法解析 DOM 元素并更新页面。

.NET ADF 应用程序中的 Ajax 调用过程如图 5.11 所示共 4 步,有些调用细节或许不同,如自定义的 Tool,但基本步骤就是 4 步。在 .NET ADF 应用程序中开发人员一般不用编写处理服务器返回结果 processCallbackResult 的函数,ADF 框架程序已经编写好该程序。当然,如果开发人员需要有独特的响应方式,可以编写自己的处理函数,并在处理函数中调用 processCallbackResult 即可。

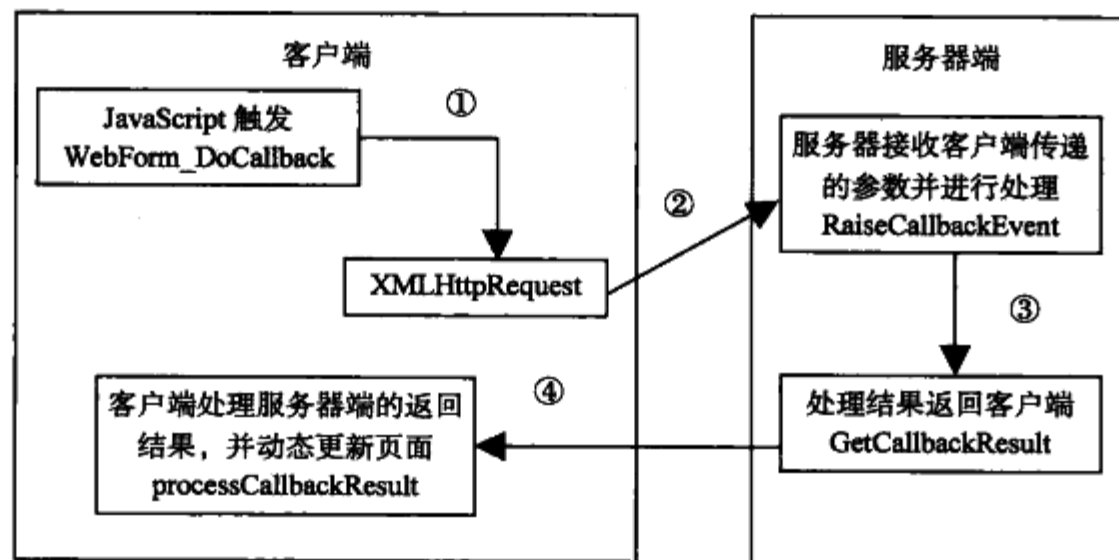


图 5.11 .NET ADF 中 Ajax 调用过程

5.4.2 Web ADF Ajax 调用的示例详解

本示例主要通过客户端输入中国的省会名称，在页面上实现异步刷新。系统实现的步骤如下。

- (1) 启动 Visual Studio 2008，建立一个“ASP.NET Web Site”应用，名称为“china”。
- (2) 在 Default 页面拖入 MapResourceManager 控件，如图 5.12 所示。
- (3) 单击“Edit Resource”，系统弹出“MapResourceItem Collection Editor”对话框，如图 5.13 所示。

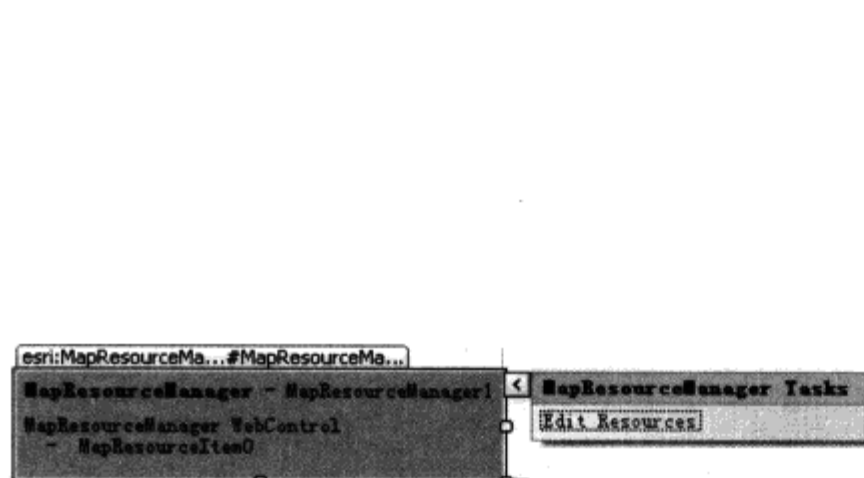


图 5.12 MapResourceManager 空间

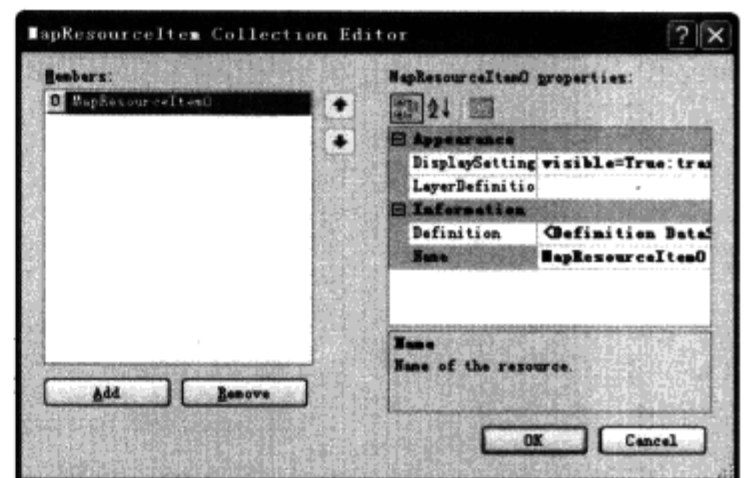


图 5.13 “MapResourceItem Collection Editor”对话框

- (4) 在相应的选项中输入在第 4 章发布的地图服务，单击“OK”按钮，完成地图资源的配置。
- (5) 在 Default 页面上拖入 Map 控件，并在 Map 控件的“MapResourceManager”选项中选择第 4 步配置的地图资源服务，如图 5.14 所示。
- (6) 在 Default 页面上拖入 Toolbar 控件，并在 Toolbar 控件的“BuddyControls”属性项选择第 5 步中拖入的地图对象，如图 5.15 所示。
- (7) 单击 Toolbar 的“ToolBarItems”属性，系统会弹出“ToolBarCollectionEditorForm”对话框，如图 5.16 所示。
- (8) 把“Map Navigation”下面的各项，全部添加到“Current Toolbar Contents”列表框中，单击“OK”按钮，完成 Toolbar 控件的设置。
- (9) 在 Default 页面中，输入客户端控件录入省份名称的 HTML 代码和客户端响应的

JavaScript 代码。

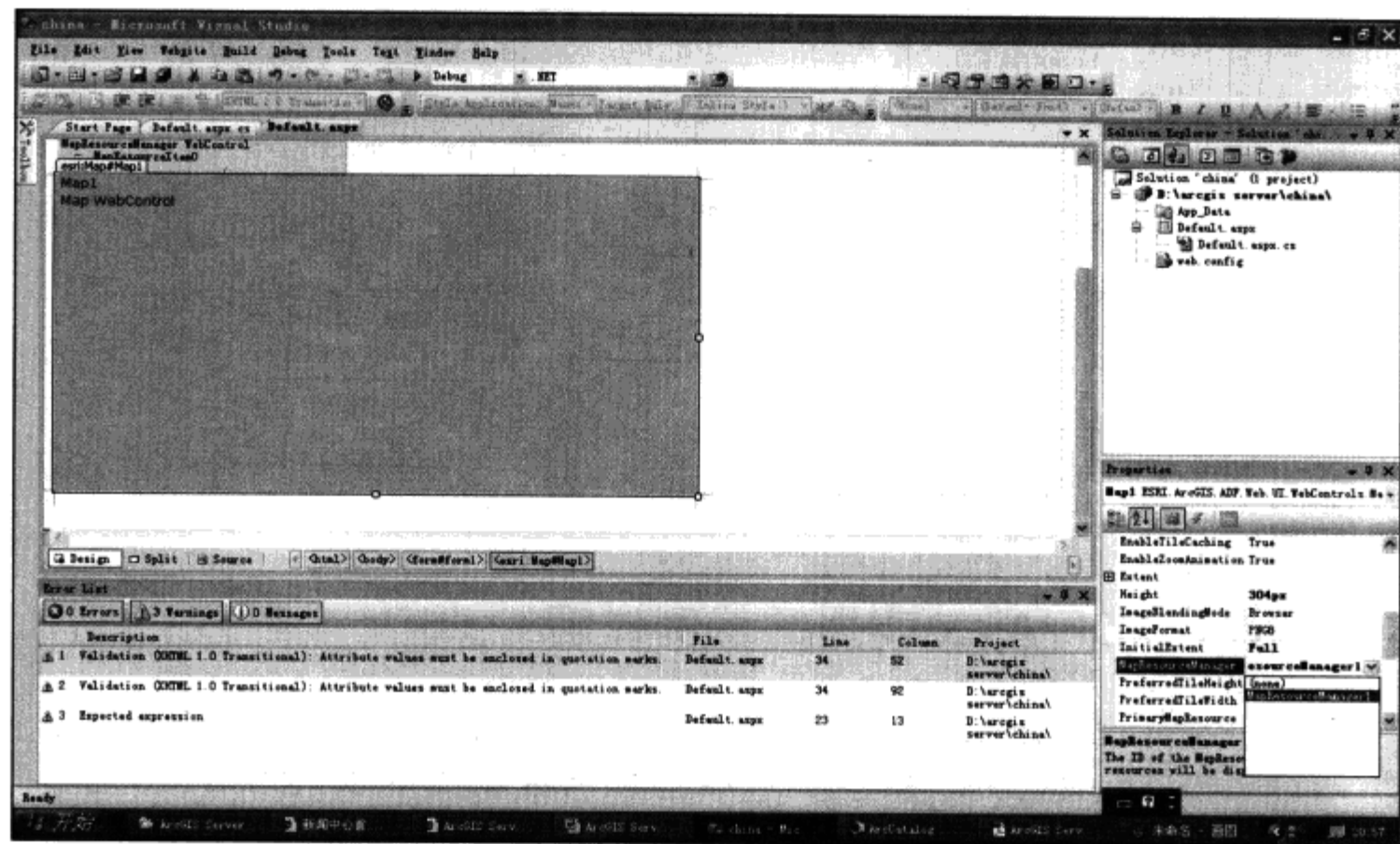


图 5.14 配置 Map 的地图资源

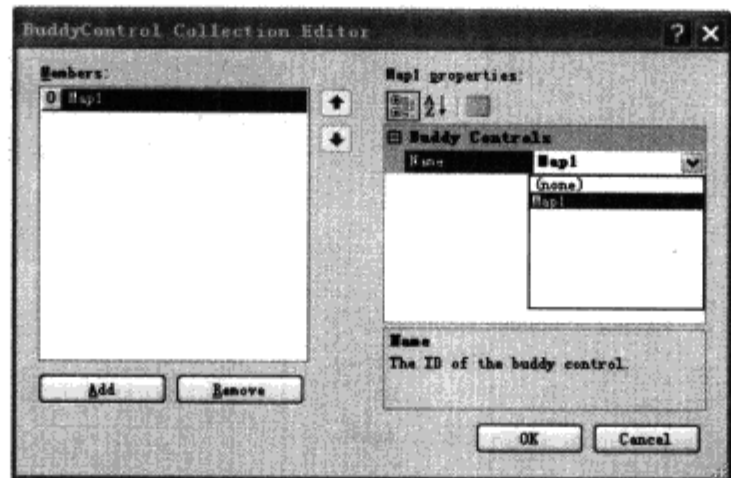


图 5.15 设置 Toolbar 的 BuddyControls 属性



图 5.16 设置 Toolbar 的 ToolbarItems 属性

```

        <td>请输入省会名称</td>
        <td><input id = 'provice' type=text /></td>
        <td><input id ='OkBut' type=button value ="定位"  onclick="javascript:
SearchPostion();" /></td>
    </tr></table>
    <script language="javascript" type ="text/javascript">
        function SearchPostion()
        {
            var strName = document.getElementById('provice').value;

            if(strName=='')
            {
                alert('请输入定位的省会名称!');
                return ;
            }
            var context = "Map1";
            var message;
            message = strName;
            <%=sADFCallBackFunctionInvocation%>
        }
    </script>

```

(10) 在服务器端代码中, 编写服务器端的响应代码, 如下所示:

```

public partial class _Default : System.Web.UI.Page, ICallbackEventHandler
{
    public string sADFCallBackFunctionInvocation;
    protected void Page_Load(object sender, EventArgs e)
    {
        sADFCallBackFunctionInvocation = Page.ClientScript.
GetCallbackEventReference(this, "message", "processCallbackResult", "context",
"postBackError", true);
    }

    public void RaiseCallbackEvent(string eventargs)
    {
        System.Collections.IEnumerable func_enum = null;
        //获取当前 Map1 控件中所有的 functionality
        func_enum = Map1.GetFunctionalities();

        System.Data.DataTable datatable;
        //对所有的 functionality 进行遍历
        foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gisfunctionality
in func_enum)
        {
            ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource = null;
            //得到该 functionality 的 resource
            gisresource = gisfunctionality.Resource;
            //判断该 resource 是否支持 IQueryFunctionality
            bool supported = false;
            supported =
gisresource.SupportsFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQuer
yFunctionality));
            if (supported)

```

```
{
    ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality qfunc;
    qfunc = (ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality)
gisresource.CreateFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunc
ionality), null);

    string[] lids;
    string[] lnames;
    //获得图层的 layerId 和 layerName, GetQueryableLayers 的重载方法可以指定
//图层类型
qfunc.GetQueryableLayers(null, out lids, out lnames);

    int selindex = -1;
    for (int i = 0; i < lids.Length; i++)
    {
        if (lnames[i] == "省界")
        {
            //找到省界土层的序号
            selindex = i;
            break;
        }
    }
    //设置过滤器的过滤条件
    ESRI.ArcGIS.ADF.Web.SpatialFilter spatialfilter = new ESRI.ArcGIS.
ADF.Web.SpatialFilter();
    spatialfilter.ReturnADFGeometries = true;
    spatialfilter.MaxRecords = 100;
    spatialfilter.WhereClause = "name='"+eventargs+"'";

    //对指定的图层进行查询，查询的结果保存为
    datatable = qfunc.Query(null, lids[selindex], spatialfilter);
    if (datatable != null)
    {
        if (datatable.Rows.Count > 0)
        {
            for (int j = 0; j < datatable.Columns.Count; j++)
            {
                if (datatable.Columns[j].DataType == typeof(ESRI.ArcGIS.
ADF.Web.Geometry.Geometry))
                {
                    ESRI.ArcGIS.ADF.Web.Geometry.Geometry geom = (ESRI.
ArcGIS.ADF.Web.Geometry.Geometry)datatable.Rows[0][j];
                    Map1.Zoom(geom);
                    sADFCallbackFunctionInvocation = Map1.
CallbackResults.ToString();
                    return;
                }
            }
        }
    }
}
}
```



```

public string GetCallbackResult()
{
    return sADFCallBackFunctionInvocation;
}
}

```

(11) 系统运行结果如图 5.17 所示。

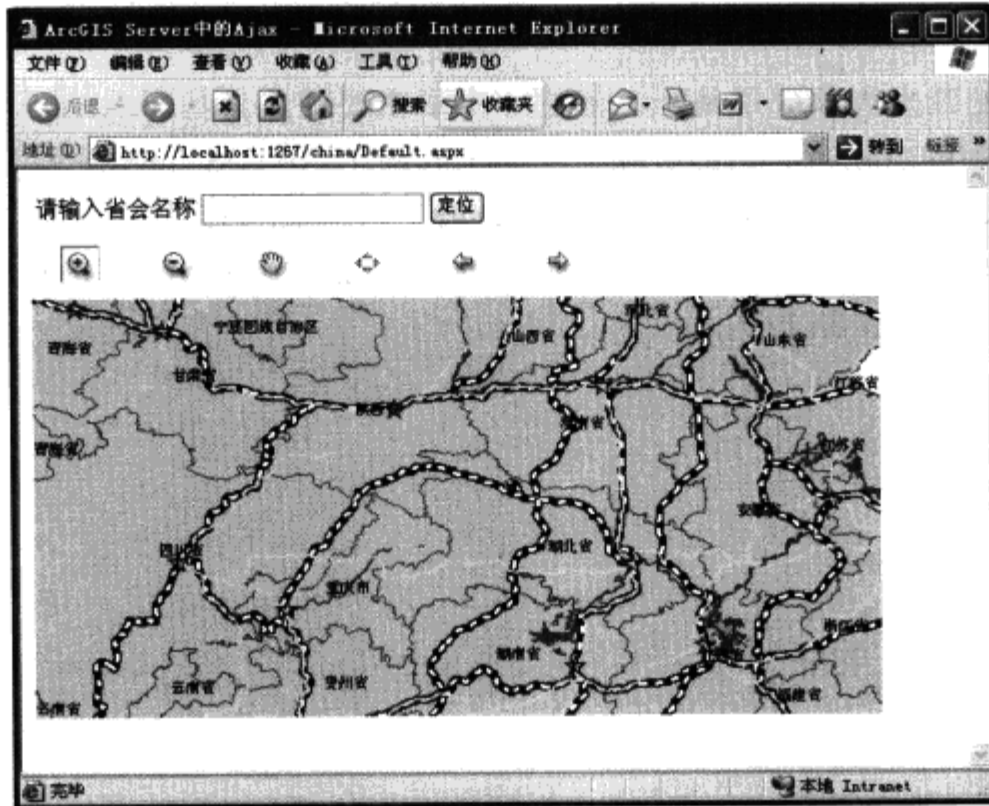


图 5.17 系统启动界面

(12) 在“请输入省会名称”文本框中输入北京市，单击“定位”按钮，系统定位如图 5.18 所示。

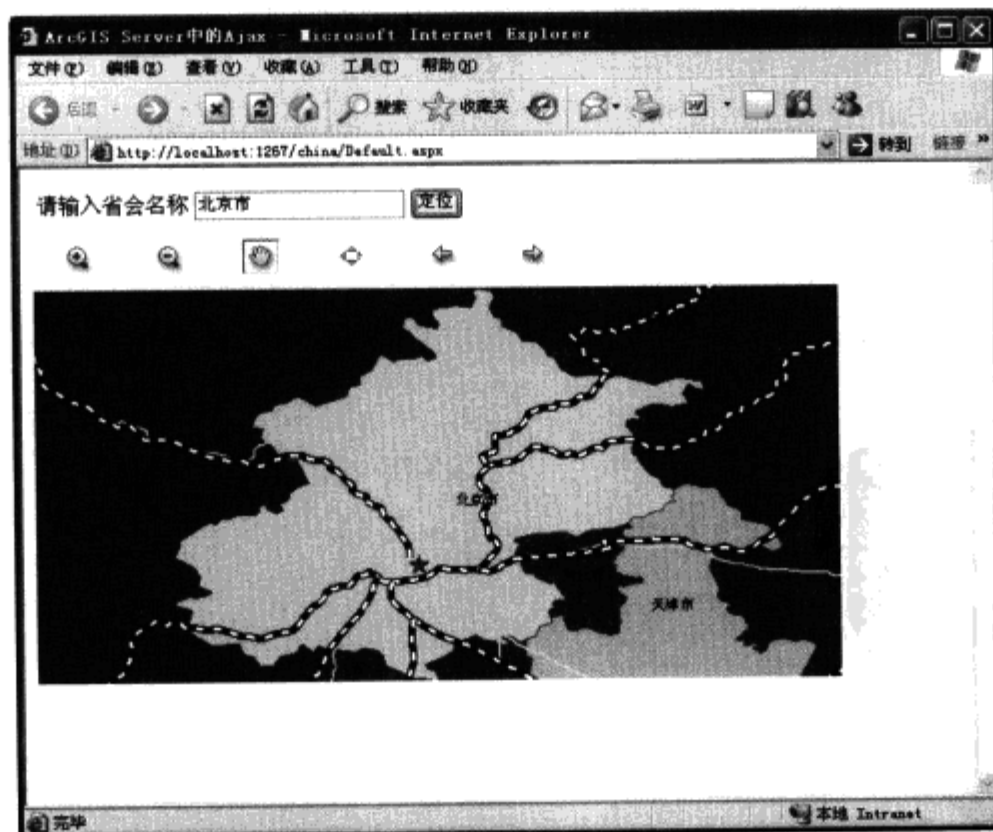


图 5.18 北京市地图定位

本示例是根据省会名称来定位，程序中的 Ajax 调用过程与 5.4.1 小节中介绍的完全一样，客

户端的 JavaScript:<%=sADFCallbackFunctionInvocation%>, 其作用就是 WebForm_DoCallback 触发异步调用。当用户输入省会名称, 单击“定位”按钮, 就会触发 XMLHttpRequest, 服务器接收客户端传递过来的省会名称, 用 RaiseCallbackEvent 函数查找该对象并定位, 最后把服务器端处理结果返回客户端, 地图就会刷新到相应的省会位置。

5.5 小结

ArcGIS Server 的开发大部分是基于 Web 的。本章主要讲述了基于 .NET Web 开发的基础, 重点介绍了 ASP.NET 中的 Ajax 和 JavaScript, 还简单介绍了 ArcGIS Server Web ADF 在 ASP.NET 中 Ajax 调用过程。读完本章, 读者应该熟悉 ASP.NET 的 Ajax 的调用, 并对 ArcGIS Server Web ADF 中的 Ajax 有大致地了解。有关 ArcGIS Server Web ADF 中的 Ajax, 后面的章节会有更详细的介绍, 读者需要了解其基本过程。

第6章

ArcGIS Server控件介绍

为了方便用户进行自己特定的 GIS 应用开发, ArcGIS Server 把通用的 GIS 功能进行了封装, 提供了大量的控件。下面将一一介绍这些控件。

6.1 资源管理控件

资源管理控件主要管理本应用的一些服务: 地图服务、地理处理服务和地理编码服务, 主要包括 MapResourceManager、GeoprocessingResourceManager 和 GeocodeeResourceManager 等控件。

6.1.1 MapResourceManager 控件

MapResourceManager 地图资源管理控件通常是在开发 ArcGIS Server Web ADF 应用时第一个使用的控件, 也是最重要的控件之一。在软件设计的时候开发人员可以在网页上添加、删除 MapResourceManager 控件, 也可以设置 MapResourceManager 的属性。MapResourceManager 控件可以同时管理多资源, 并且可以设置图层的顺序以及图像输出的透明度。很多控件可以使用同一个地图资源管理控件去访问不同的资源, 因此, 对资源的改变, 也能在资源管理控件上反映出来。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.MapResourceManager。
- **JavaScript libraries** display_common.js, display_dotnetadf.js。

MapResourceManager 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点, 在设计模式下打开一个页面, 选择工具箱并且展开 ArcGIS 控件标签, 选中 MapResourceManager 控件, 然后将它拖到 Web 页面中, 如图 6.1 所示。

(2) 添加资源。

选中刚才添加的控件, 单击右上角的三角按钮出现 MapResourceManager 任务标签, 单击“Edit

Resources” 菜单，弹出资源编辑界面，如图 6.2 所示。

单击“Add”按钮，系统自动添加一个 MapResourceItem，MapResourceItem 包含一系列在 Web 应用程序的控件（如 Map、Toc）中如何显示地图资源的属性，如图 6.3 所示。

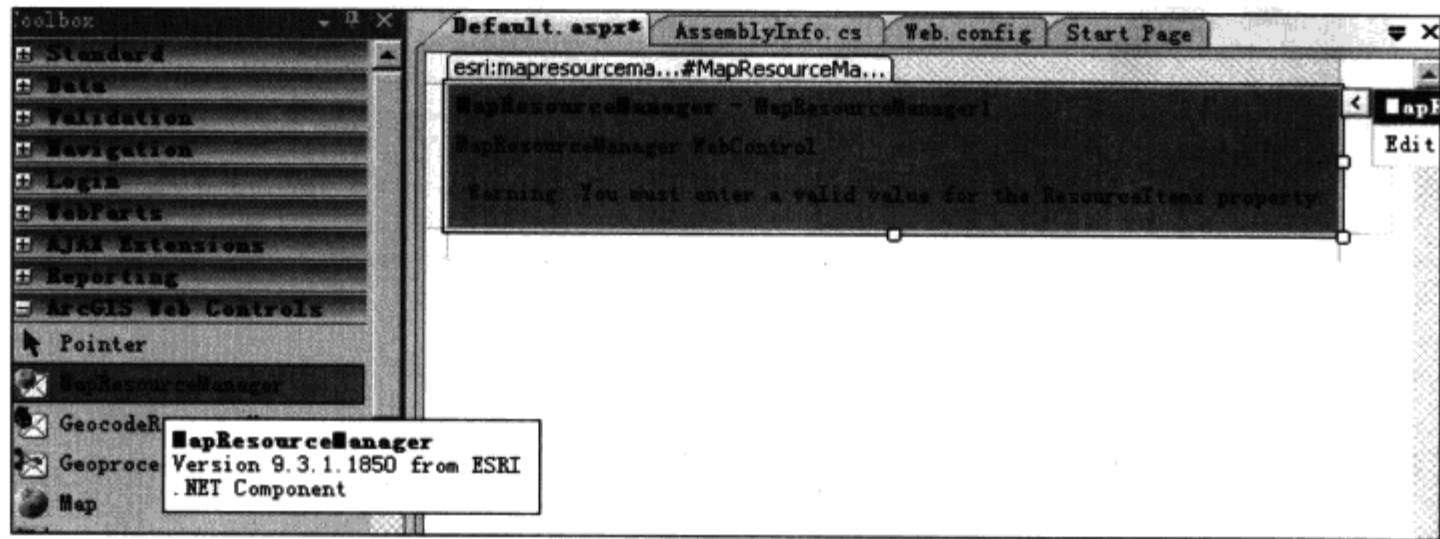


图 6.1 添加 MapResourceManager 控件

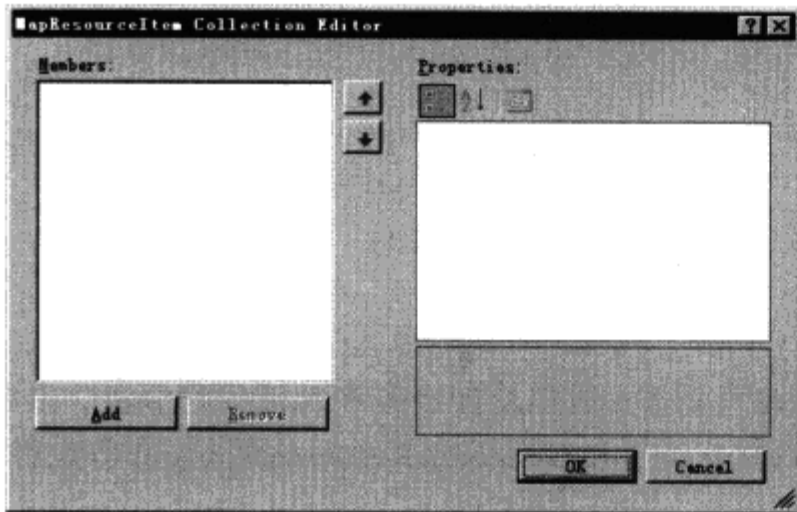


图 6.2 编辑地图资源对话框

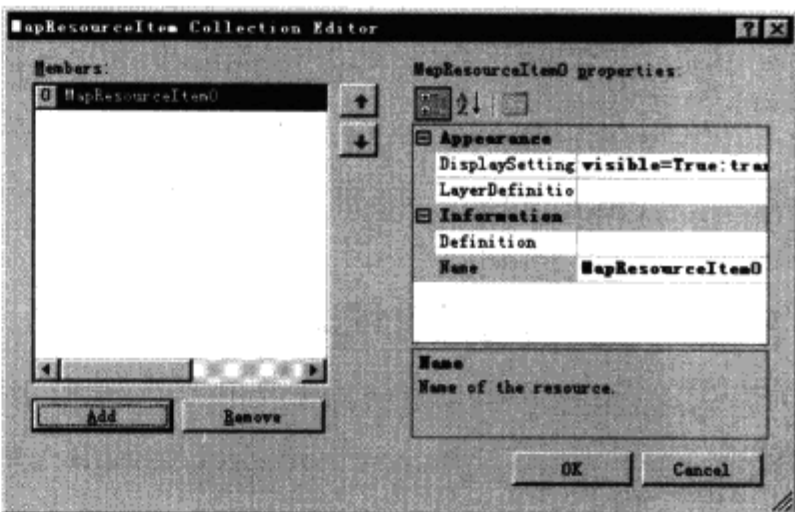


图 6.3 MapResourceItem 属性

MapResourceItem 的属性包括。

- **Name** 惟一标识地图资源的名字，该名字会出现在控件 Toc 中。开发人员可以根据自己的喜好来命名其名称，但在同一个 MapResourceManager 中其必须是惟一的。
- **DisplaySetting** 定义地图如何显示，如图 6.4 所示。

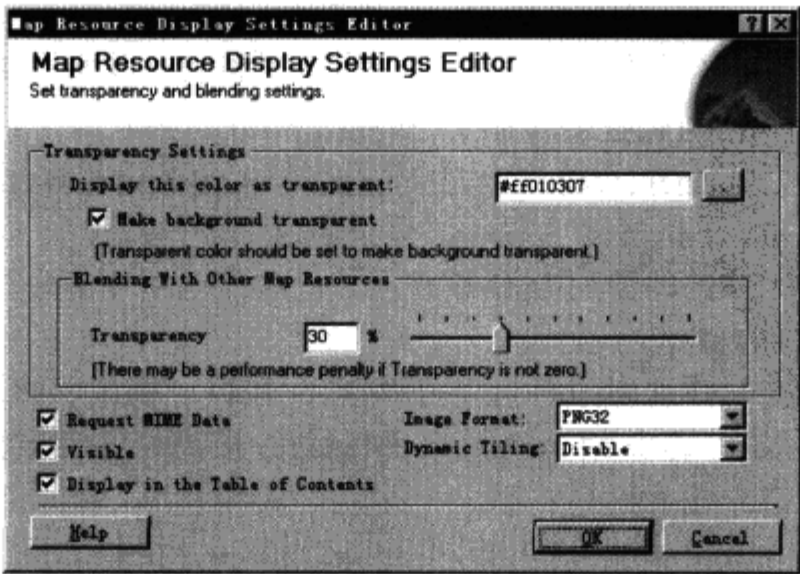


图 6.4 地图显示设置对话框

其中：

- Transparency Settings: 设置透明度, Transparency 值 0 为不透明; 100 则为完全透明。
- Request MIME Data: 数据源中的地图图片访问方式。如果数据源支持通过 MIME 格式发送地图图片, 那么图片就可以存储在 Web ADF 应用程序的内存中; 反之, 数据源只能在共享的输出目录下生成图片供 Web ADF 应用程序访问。
- Image Format: 生成图片的格式, 需要在图片质量与网络传输的数据量间进行平衡。
- Visible: 决定图片是否生成。如果 Visible 是 True 的话, 应用程序运行时生成该地图图片, 在 Toc 控件中该地图资源是 checked, 反之则不生成。
- Display in the Table of Contents: 地图资源是否在 Toc 控件中出现。
- Dynamic Tiling: 动态生成瓦片。
- **Layer Definition** 层的定义, 可以对层的属性进行设置, 如高亮显示的符号, 字段的别名等, 如图 6.5 所示。

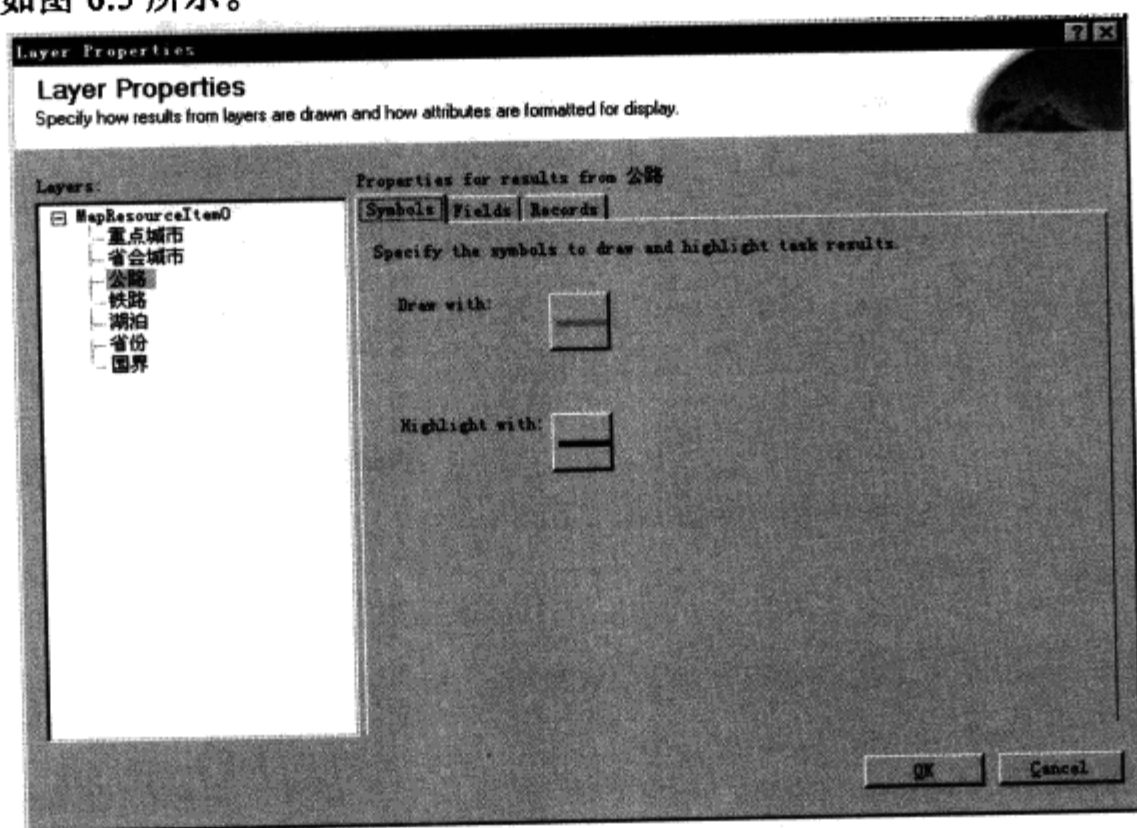


图 6.5 定义地图资源中的层

- **Definition** 定义连接方式。空间数据源的连接方式有 ArcGIS Server Internet、ArcGIS Server Local、ArcIMS、ArcWeb Services、GraphicsLayer、OGC (WMS) Service 等, 如图 6.6 所示。

ArcGIS Server Internet 表示 Web 应用服务器与空间数据服务可以是局域网也可以位于广域网, 该连接方式空间数据服务以 Web Service 的方式提供。定义数据源输入一个 URL 地址即可, 如图 6.7 所示。

ArcGIS Server Local 表示空间数据服务与 Web 应用服务器处于同一局域网内, 这种连接方式使用简单, 在“Data Source”文本框内输入 GIS Server 的名字, 并在“Resource”列表框选择好空间数据的数据资源。使用这种连接方式时, 一般需要添加“ArcGIS Identify”, 添加的方法如图 6.8 所示。

GraphicsLayer 也是常用的一种连接方式, 通常用于对特殊的图层或元素进行渲染, 如选中的高亮显示。它一般存储在 Web ADF 应用程序的内存中, 在应用程序设计的时候是不可见的,

需要开发人员编写代码来控制其表现形式。应注意，其图层在 Toc 控件中可见，也可以不见。

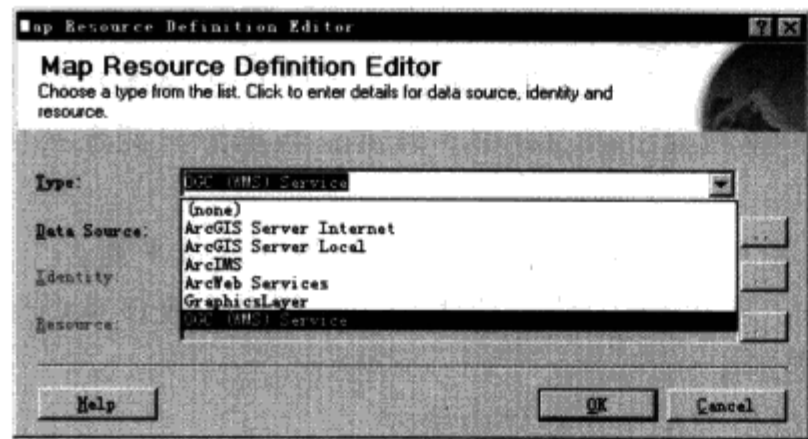


图 6.6 空间数据源定义

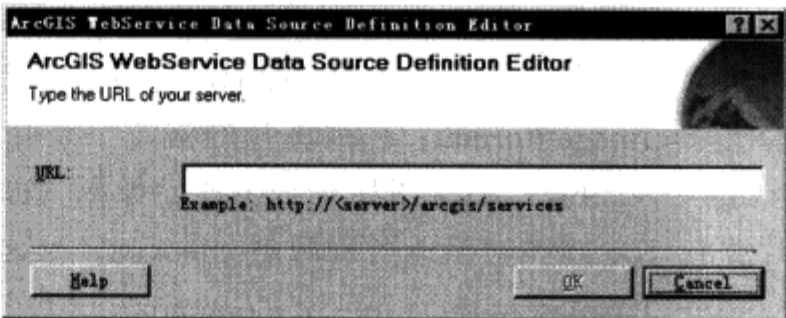


图 6.7 输入 ArcGIS Server 数据源的 URL

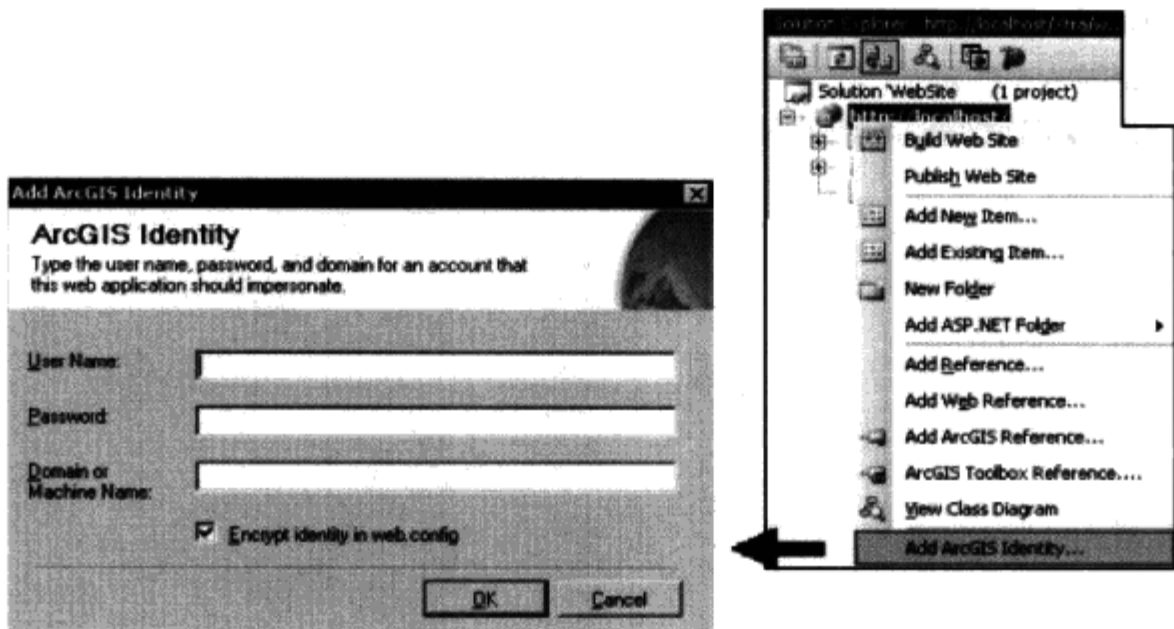


图 6.8 添加 ArcGIS Identify

有关 ArcIMS、ArcWeb Services、OGC (WMS) Service 等连接方式与上面的类似，只是连接的数据源不同，在此不再赘述。

前面介绍的都是用 MapResourceManager 控件在应用程序设计的时候来配置数据源，能否用程序来动态创建数据的连接呢？答案是可以，请看下面的示例程序：

```
static public MapResourceItem createMapResource(string name, string type, bool
showInToc, bool isvisible, string hostname, string resname)
{
    MapResourceItem item = new MapResourceItem();
    item.Name = name;
    item.Definition = new GISResourceItemDefinition();
    item.DisplaySettings = new ESRI.ArcGIS.ADF.Web.DisplaySettings();
    if (type == "0")//创建 GraphicsLayer
    {
        item.Definition.DataSourceShared = true;
        item.Definition.DataSourceDefinition = "In Memory";
        item.Definition.DataSourceType = "GraphicsLayer";
        item.DisplaySettings.DisplayInTableOfContents = showInToc;
        item.DisplaySettings.Visible = isvisible;
    }
}
```

```

else
{
    item.Definition.DataSourceShared = true;
    item.Definition.DataSourceDefinition = hostname;
    item.Definition.DataSourceType = "ArcGIS Server Local";
    item.Definition.ResourceDefinition = resname;
    item.DisplaySettings.DisplayInTableOfContents = showInToc;
    item.DisplaySettings.Visible = isVisible;
}
MapResourceManager man1 = new MapResourceManager();
man1.ResourceItems.Insert(0, item);
man1.CreateResource(item);
item.InitializeResource();
return item;
}

```

6.1.2 GeoprocessingResourceManager 控件

GeoprocessingResourceManager 是管理空间数据处理服务的资源管理器。空间数据处理服务一般是由 ArcToolbox 中的 ModelBuilder 工具建立好空间数据的处理模型，并把空间数据处理模型发布成空间数据处理服务，GeoprocessingResourceManager 控件就可以配置 Web ADF 应用程序访问的空间数据处理服务了。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.GeoprocessingResourceManager。
 - **JavaScript libraries** display_common.js, display_dotnetadf.js。
- GeoprocessingResourceManager 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 GeoprocessingResourceManager 控件，然后将它拖曳到 Web 页面中，如图 6.9 所示。

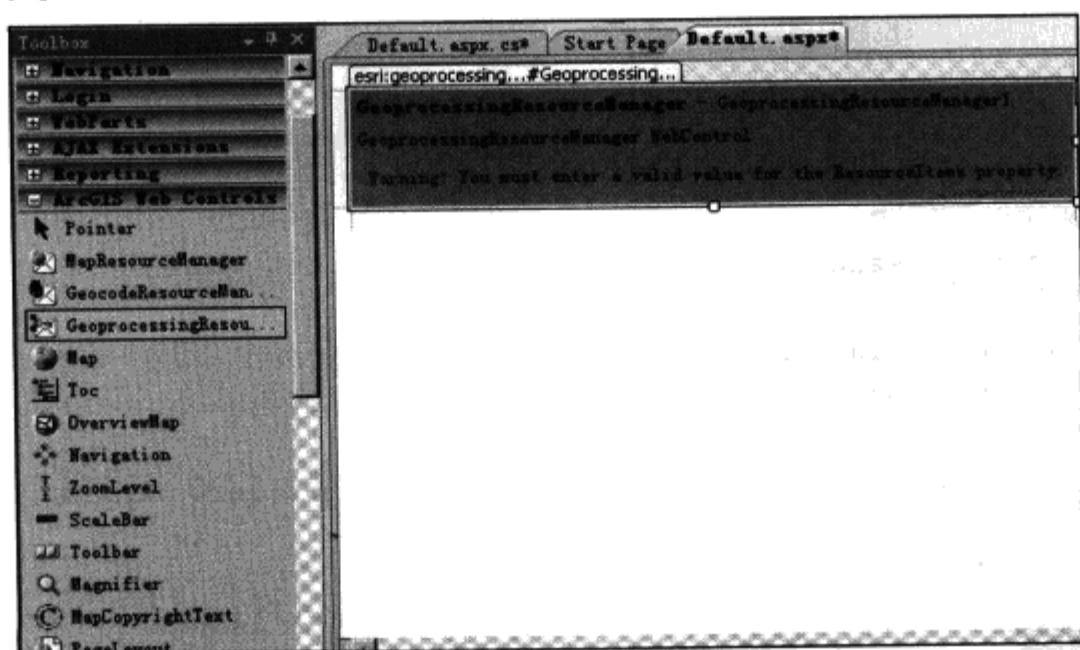


图 6.9 添加 GeoprocessingResourceManager 控件

(2) 添加控件处理资源。

选中刚才添加的控件,单击右上角的三角按钮出现 GeoprocessingResourceManager 任务标签,单击“Edit Resources”菜单,弹出资源编辑界面,如图 6.10 所示。

- **Name** 空间资源处理的名字,开发人员根据实际情况来命名。
- **Definition** 定义空间数据资源的连接。单击“OK”按钮,系统弹出空间数据处理对话框,如图 6.11 所示。

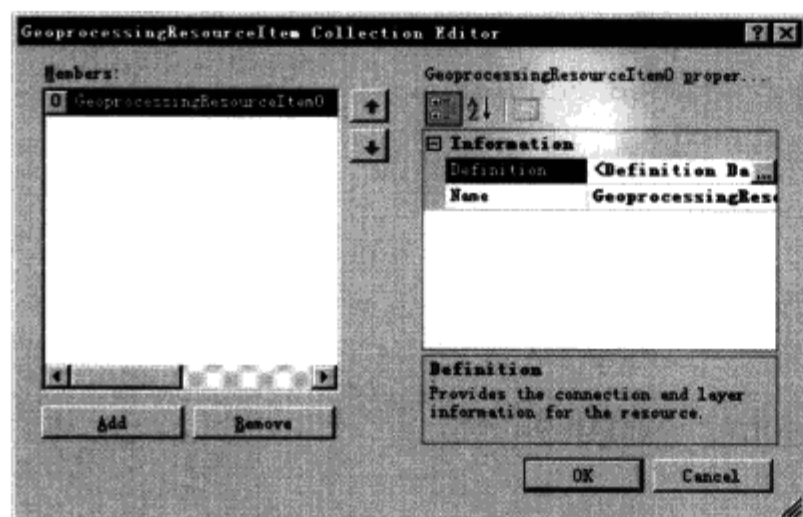


图 6.10 GeoprocessingResourceItem 属性对话框

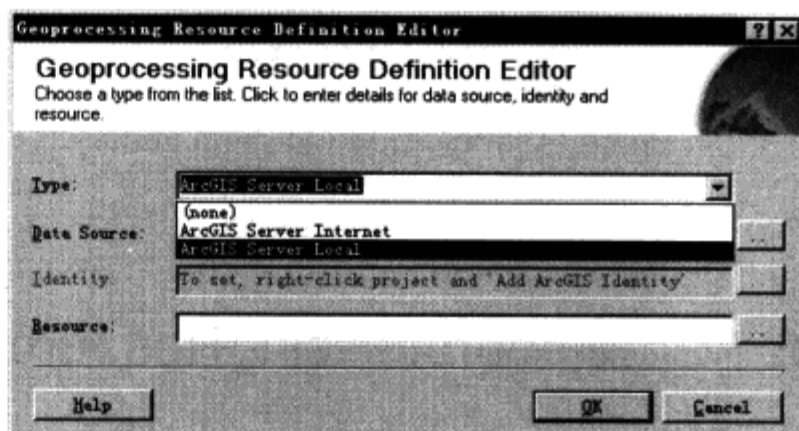


图 6.11 空间处理服务定义对话框

与 MapResourceManager 不相同的是,空间处理服务的连接方式只有 ArcGIS Server Internet 和 ArcGIS Server Local 两种方式,这两种连接方式的配置与 MapResourceManager 的配置方式相似,此处不再赘述。

GeoprocessingResourceManager 也可以动态绑定,示例程序如下:

```
GeoprocessingResourceManager    geoprocessingResourceManager    =    new
GeoprocessingResourceManager();

GeoprocessingResourceItem        geoprocessingResourceItem        =    new
GeoprocessingResourceItem();
geoprocessingResourceItem.Name = "GeoprocessingResourceItem";

GISResourceItemDefinition definition = new GISResourceItemDefinition();

definition.DataSourceDefinition = "http://localhost/arcgis/services";
definition.DataSourceType = "ArcGIS Server Internet";
definition.ResourceDefinition = "BufferTools";

geoprocessingResourceItem.Definition = definition;

geoprocessingResourceManager.ResourceItems.Add(geoprocessingResourceItem);
geoprocessingResourceManager.CreateResource(geoprocessingResourceItem);

IGISResource gisResource = geoprocessingResourceManager.GetResource(0);

if (!gisResource.Initialized)
    geoprocessingResourceItem.InitializeResource();
```

6.1.3 GeocodeResourceManager 控件

GeocodeResourceManager 是地理编码资源管理控件，开发人员在开发 Web ADF 应用程序的时候可以在网页上添加、删除、修改地理编码控件的属性。FindAddressTask 就是利用 GeocodeResourceManager 提供的地理编码资源服务的控件。另外，开发人员在开发应用程序的时候可以利用通用 API: IGeocodeResource 和 IGeocodeFunctionality 接口来访问地理编码服务。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.GeocodeResourceManager。
 - **JavaScript libraries** display_common.js, display_dotnetadf.js。
- GeocodeResourceManager 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 GeocodeResourceManager 控件，然后将它拖到 Web 页面中，如图 6.12 所示。

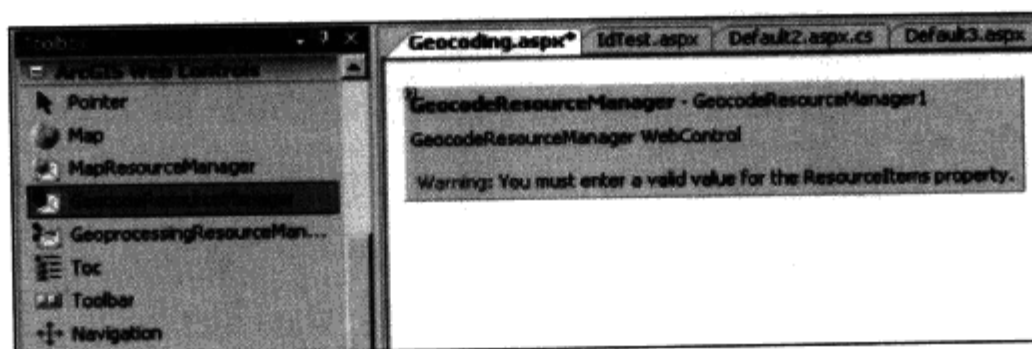


图 6.12 在页面上添加 GeocodeResourceManager 控件

(2) 添加资源。

选中刚才添加的控件，单击右上角的三角按钮出现 GeocodeResourceManager 任务标签，单击“Edit Resources”菜单，单击“Add”按钮，弹出资源编辑界面，如图 6.13 所示。

- **MinCandidateScore** 是介于 0~100 的整数，地理编码服务查找返回结果的个数。默认值为 10。
- **MinMatchScore** 是介于 0~100 的整数，根据地址匹配最小的相似程度。一般认为相似程度在 80~99 的匹配是一个好的地址匹配。
- **ShowAllCandidates** 当其值为 True 时，显示所有的返回结果；反之显示匹配程度大于 MinMatchScore 的结果，默认是 False。
- **Name** 地理编码服务的名字，开发人员根据实际情况来命名。
- **Definition** 定义连接地理编码服务的方式。单击右边的按钮，系统弹出地理编码资源定义对话框，如图 6.14 所示。

地理编码资源的连接方式有：ArcGIS Server Internet、ArcGIS Server Local、ArcIMS、ArcWeb Services 等方式，其配置与 MapResourceManager 中的相似。

GeocodeResourceManager 同样可以动态绑定，示例代码如下：

```
GeocodeResourceManager geocodeResourceManager = new GeocodeResourceManager();

GeocodeResourceItem geocodeResourceItem = new GeocodeResourceItem();
```

```
geocodeResourceItem.Name = "GeocodeResourceItem";

GISResourceItemDefinition definition = new GISResourceItemDefinition();

definition.DataSourceDefinition = "localhost";
definition.DataSourceType = "ArcGIS Server Local";
definition.ResourceDefinition = "PortlandGeocode";

geocodeResourceItem.Definition = definition;

geocodeResourceManager.ResourceItems.Add(geocodeResourceItem);
geocodeResourceManager.CreateResource(geocodeResourceItem);

IGISResource gisResource = geocodeResourceManager.GetResource(0);

if (!gisResource.Initialized)
    geocodeResourceItem.InitializeResource();
```

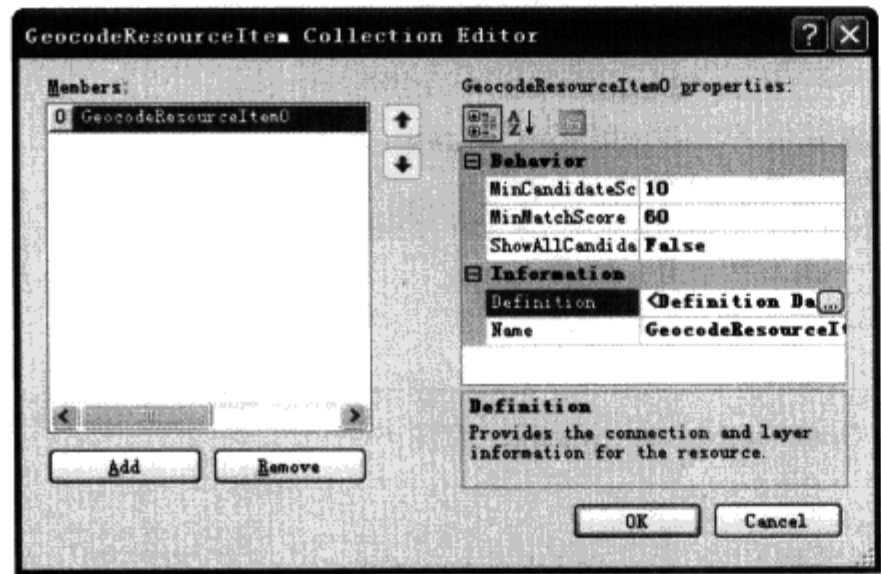


图 6.13 “GeocodeResourceItem 集合编辑”对话框

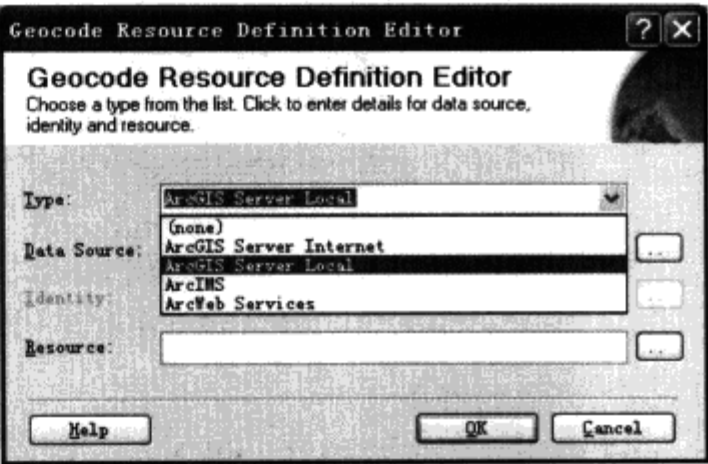


图 6.14 地理编码资源定义对话框

6.2 地图显示及其相关控件

下面要介绍的控件要么是显示地图、要么是控制地图的显示，也就是说都是与地图显示相关的一些控件。

6.2.1 Map 控件

Map 是用来显示一个或多个不同类型数据资源的地图图形控件。Map 控件必须与 MapResourceManager 控件关联并用来显示 MapResourceManager 中定义的地图资源。前面已经讲过 MapResourceManager 控件可以定义地图服务的透明图等，可以把每个地图服务当成一个单独的逻辑层来表示。Map 控件可以把多个不同类型的地图服务融合为一个地图，并且有异步刷新功能，如在放大、漫游地图时不用刷新整个页面就可以实现地图图片的更新。还有很多其他的控件与 Map 控件关联可以控制地图的显示，后面的章节会进行详细的讲解。

- Assembly ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。

- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.Map。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_map.js, isplay_vector.js 。

Map 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 Map 控件，然后将它拖到 Web 页面中，如图 6.15 所示。

(2) 配置地图服务资源。

在 Map 控件需要现实的地图资源时，先要配置 MapResourceManager 的地图服务(参见 6.1.1 小节)。配置好 MapResourceManager 地图服务后，设置 Map 的“MapResourceManager”属性，如图 6.16 所示。

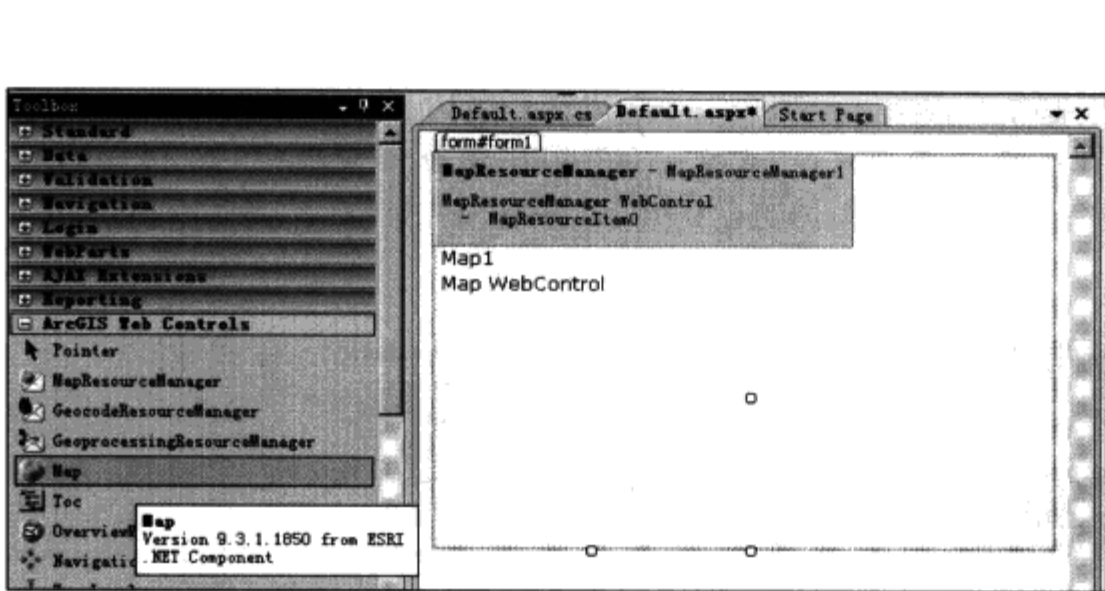


图 6.15 添加 Map 控件



图 6.16 Map 控件属性设置

需要说明的是，如果是显示多个地图服务，Map 控件在融合多个服务时以 PrimaryMapResource 设置的服务为基础去融合其他的地图服务，注意 PrimaryMapResource 不能为 Graphics Layer。

Map 控件中默认的一些操作见表 6.1 所示。

表 6.1 键盘鼠标快捷操作地图

Keyboard/Mouse Shortcut	Map Action
SHIFT、left mouse click、hold、drag box	ZoomIn (放大)
ALT、left mouse click、hold、drag box	ZoomOut (缩小)
Left mouse click、hold、move	Pan (漫游)
Num Lock disabled、use keypad	Pan (漫游)
Arrow keys、Home、End、Page Up、Page Down	Pan (漫游)
CTRL、left mouse click	Identify (属性查询)
Mouse Wheel Up	ZoomIn (放大)
Mouse Wheel Down	ZoomOut (缩小)

当图的内容发生改变，或者图层的可见性发生改变时，就需要用代码来刷新地图。注意，如果地图的比例尺没有改变，这个地图不需要完全刷新。在默认情况下，Map 控件利用浏览器来

融合不同地图服务，刷新地图只需要刷新发生改变的地图服务即可，对于其他没有发生改变的地图服务就不用刷新。示例代码如下：

```
if (map.ImageBlendingMode == ImageBlendingMode.WebTier)
{
    map.Refresh();
}
else if (map.ImageBlendingMode == ImageBlendingMode.Browser)
{
    map.RefreshResource(MyMapResource.Name);
}
```

地图融合模式（ImageBlendingMode），该属性只在针对多个地图服务的时候采用，它有两种方式：WebTier 和 Browser。Browser 是默认的方式，地图服务的图片通过 URL 或者 MIME 数据流的方式来访问，浏览器分别画不同的地图服务，这样客户端就可以分别刷新不同的地图服务，而不用全部刷新，如图 6.17 所示。

WebTier 的地图融合方式则不同，它是通过 Web ADF 应用服务器分别访问不同的 GIS Server，并把它们的图片进行融合，然后客户端浏览器访问一个融合后的图片，其访问机制如图 6.18 所示。

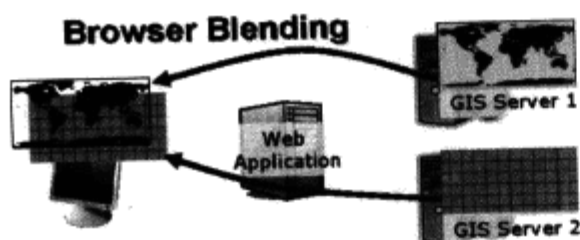


图 6.17 Browser 刷新机制

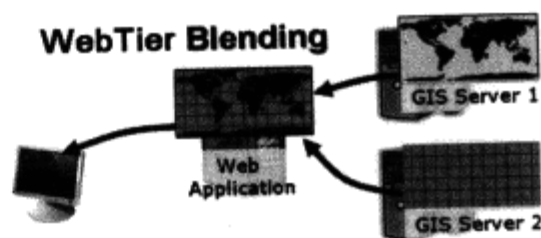


图 6.18 WebTier 融合机制

6.2.2 MapTips 控件

MapTips 可以显示与之相关联的 Map 控件图层中鼠标光标停留位置元素的属性。MapTips 控件通过 Map 属性与单个的 Map 控件关联，通过 Layer 属性设定显示哪个图层的元素的属性，并且可以设定属性的显示样式。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.MapTips。
 - **JavaScript libraries** display_common.js, display_dotnetadf.js, display_maptips.js。
- MapTips 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 MapTips 控件，然后将它拖到 Web 页面中，如图 6.19 所示。

(2) 配置 MapTips 属性。

MapTips 有两个比较重要的属性：Layer 和 Map。Layer 配置 MapTips 要显示的图层，配置方法如图 6.20 所示。

选择要实现的 Map 控件，如图 6.21 所示。

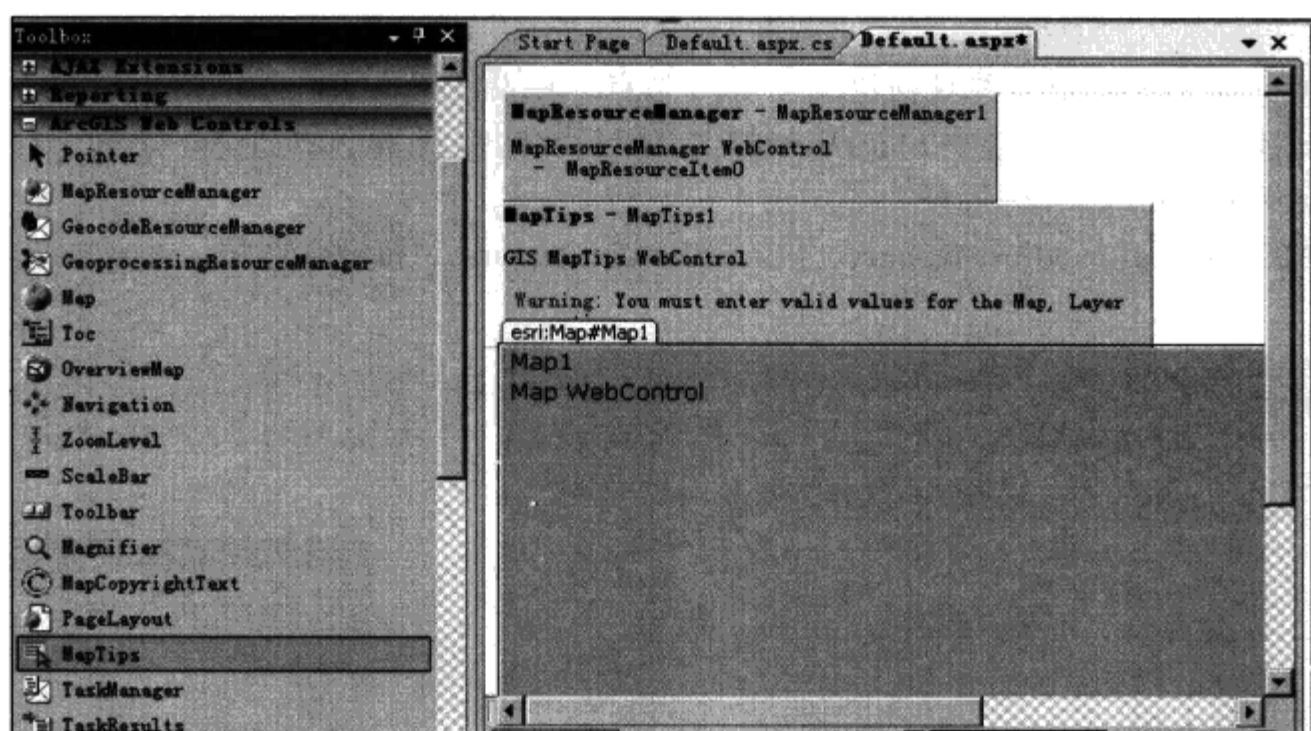


图 6.19 添加 MapTips 控件

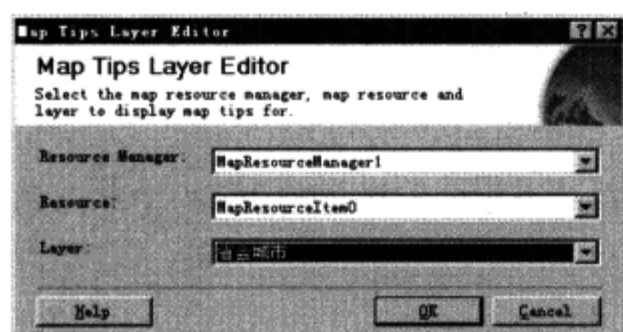


图 6.20 MapTips 的 Layer 属性编辑对话框



图 6.21 配置 MapTips 的 Map 属性

需要说明的是，在使用 MapTips 的过程中，如果 MapTips 被 map image 遮挡，可以通过修改 PageContent div 的 z-index 为 0 来改正遮挡；如果想修改 MapTips 的弹出信息，读者有兴趣的话可以去研究 display_maptips.js 文件，该文件的保存路径为：aspnet_client\ESRI\WebADF\JavaScript。

6.2.3 Maginifier 控件

Magnifier 控件就是放大镜，它允许用户放大某一小块的位置。用户拖动 Maginifier 窗口，Magnifier 就放大该窗口所对应的 Map 控件上的地图，同时还可以设置放大的倍数。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.Magnifier。

Magnifier 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 Magnifier 控件，然后将它拖到 Web 页面中，如图 6.22 所示。

(2) 定义 Magnifier 控件属性。

设置 Map、MapResourceManager、MagnifierMapResource (待放大图层) 的属性，如图 6.23 所示。

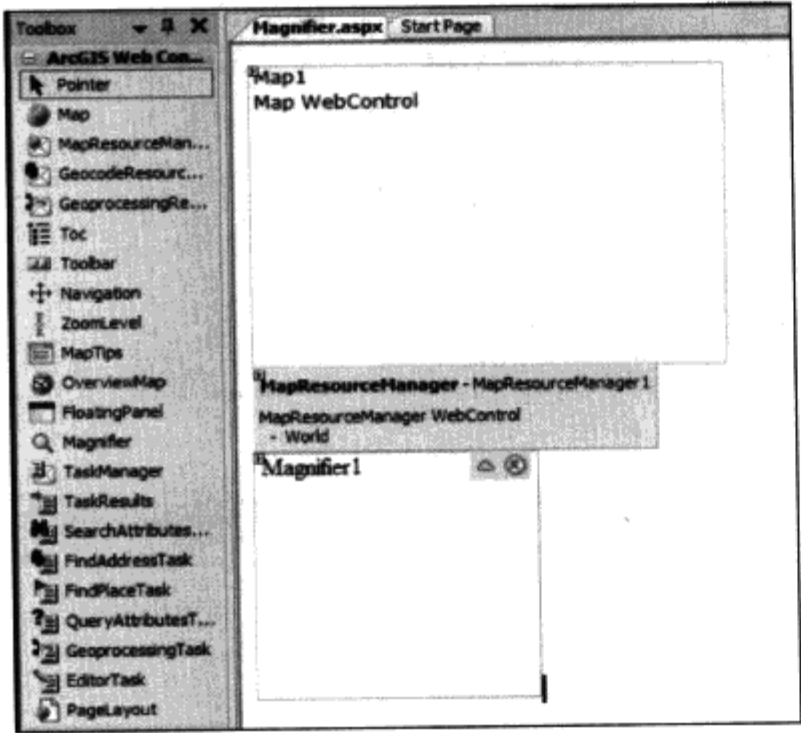


图 6.22 添加 Magnifier 控件

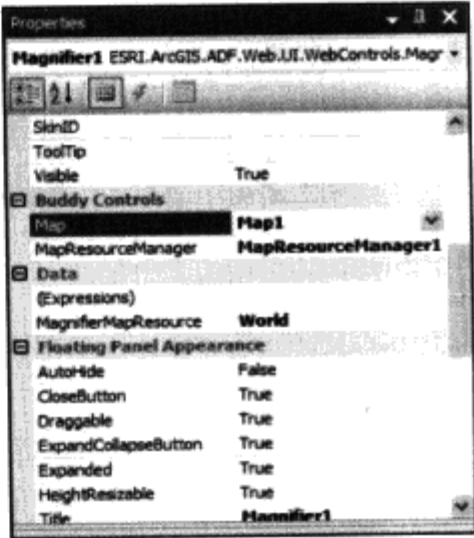


图 6.23 设置 Magnifier 控件属性

6.2.4 OverviewMap 控件

OverviewMap 缩略图也称鹰眼，把当前的地图范围画在一个小的矩形范围内，当主地图的范围发生改变时，缩略图不需要整个页面的刷新而能够自动更新。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.OverviewMap。
 - **JavaScript libraries** display_common.js, display_dotnetadf.js, display_overview.js。
- OverviewMap 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 OverviewMap 控件，然后将它拖到 Web 页面中，如图 6.24 所示。

(2) 设置 OverviewMap 控件属性。

设置 OverviewMap 控件的 Map、MapResourceManager、OverviewMapResource 等相关属性，如图 6.25 所示。

需要对 OverviewMap 控件的 StaticMode 属性说明一下：如果该属性为 true，则控件的灵活性比较弱，但可以优化性能，该控件的范围就是整个地图的范围，当主地图发生改变时，OverviewMap 标明当前地图所在的位置。若为 False，则可以提供更多的用户体验，比如当 Map 的范围变化时，该控件的范围也会随之变化，上面的 aoi 框大小也会随之重画。

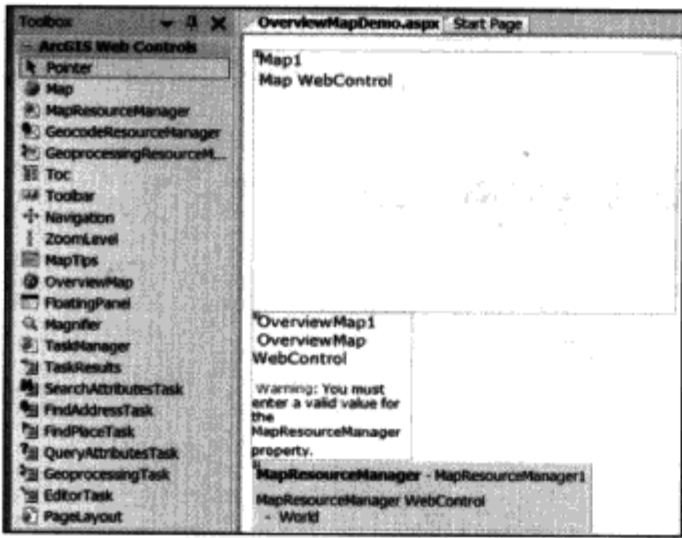


图 6.24 添加 OverviewMap 控件



图 6.25 设置 OverviewMap 属性界面

6.2.5 Toolbar 控件

Toolbar 控件可以是工具和命令行的集合，这些工具或命令被分成不同种类，执行相似的操作。在添加工具的时候，可以添加 Web ADF 已经封装好的一些操作，如放大、缩小、漫游等，这个工具不需要添加任何代码；开发人员也可以添加自定义的工具或命令，这在后面的章节中会详细讲解，本小节主要讲述一些常用工具的添加。一般来说，需要进行图形交互的采用工具，反之则采用命令。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.Toolbar。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_toolbar.js。

Toolbar 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并且展开 ArcGIS 控件标签，选中 Toolbar 控件，然后将它拖到 Web 页面中，如图 6.26 所示。

(2) 定义 Toolbar 控件。

设置 Toolbar 控件的 BuddyControl 属性，如图 6.27 所示。

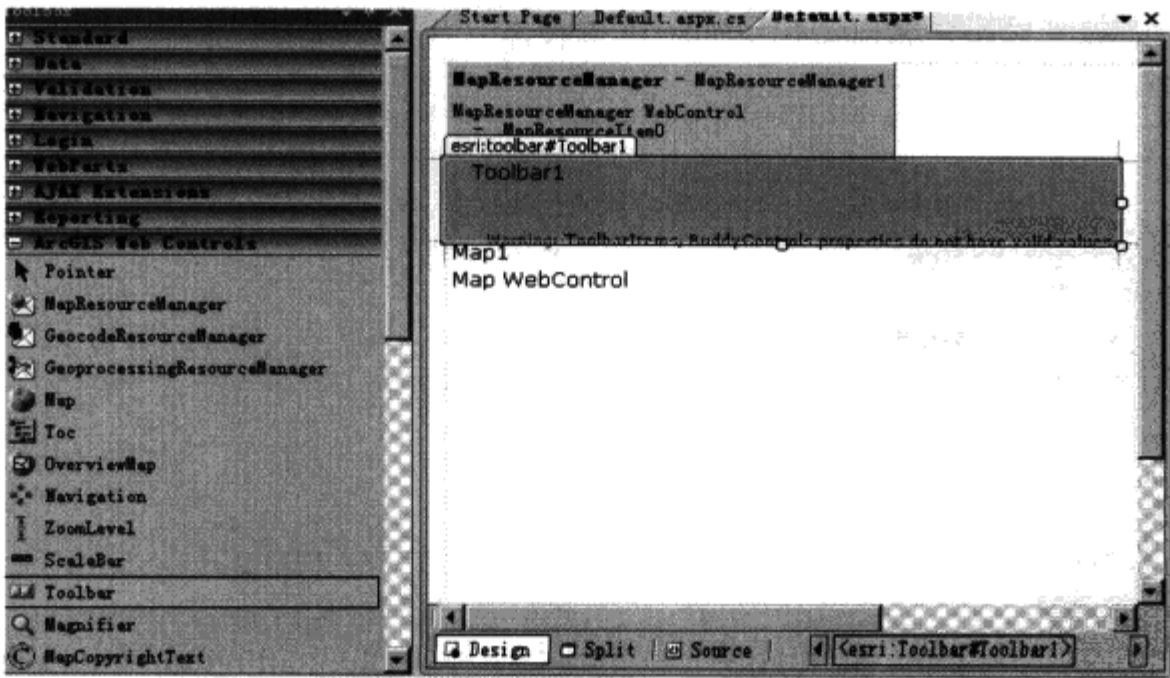


图 6.26 添加 Toolbar 控件界面

编辑 Toolbar 控件的 ToolbarItems 属性，如图 6.28 所示。可以添加 Map Navigation 下面系统已经封装好的 Tool 工具，选中具体的工具名，单击“Add”按钮即可。右边的列表框还可以进行工具的属性设置。

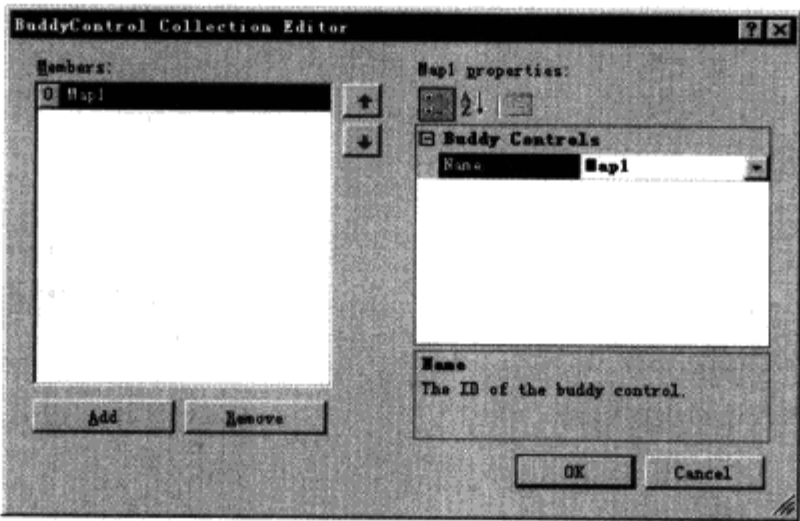


图 6.27 编辑 Toolbar 的 BuddyControl 属性

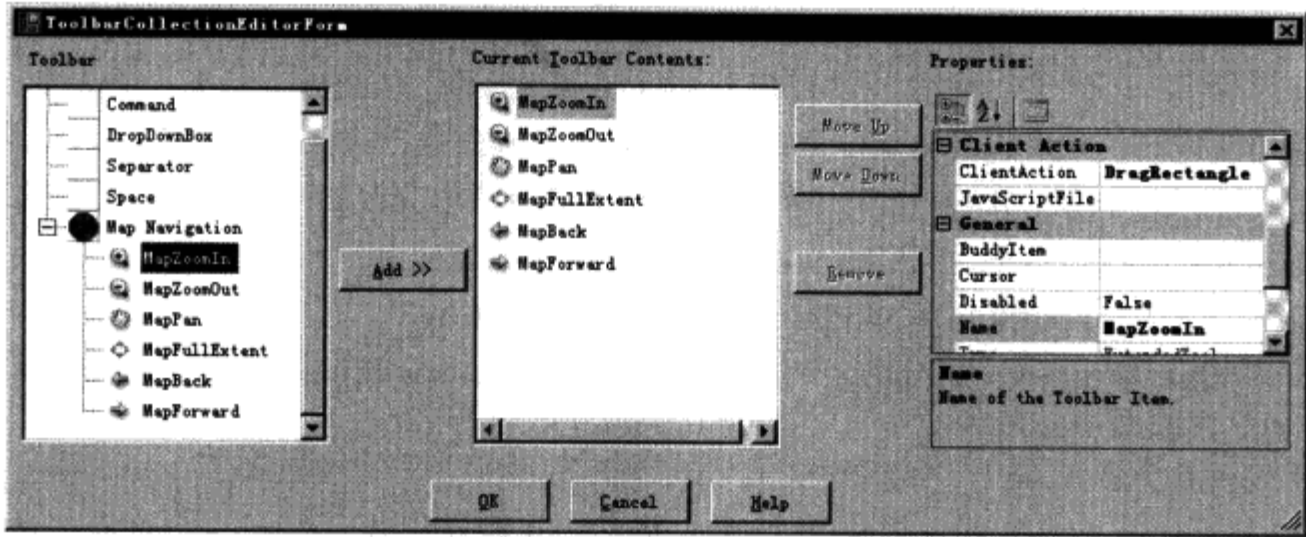


图 6.28 添加封装好的工具

所有的 Toolbar Item 响应被定义为一个名叫 ServerAction 的接口，根据 Item 类型的不同而不同。ServerAction 方法处理 Command 和 DropDownBox 有同样的类型 ToolbarItemInfo，对于 Tool 则是 ToolEventArgs，下面是如何应用它们的示例程序：

```
protected void Page_Load(object sender, EventArgs e)
{
    public class CustomTool : IMapServerToolAction {
        public void ServerAction(ToolEventArgs args)
        {
            Map map = (Map)args.Control;
            PointEventArgs pea = (PointEventArgs)args;
        }
    }

    public class CustomCommand : IMapServerCommandAction {
        public void ServerAction(ToolbarItemInfo info) {
            Map map = (Map)info.BuddyControls[0];
        }
    }
}
```

```

public class CustomDropDownBox : IMapServerDropDownBoxAction {
    public void ServerAction(ToolBarItemInfo info) {
        DropDownList dropDownBox =
        (DropDownList)info.ToolBar.ToolBarItems.Find("DropDownBoxId");
        string dropDownBoxValue = dropDownBox.SelectedValue;
    }
}

```

ToolBar 有很多属性可以控制 ToolBar 的显示风格,读者在使用 ToolBar 的过程可以根据需要进行相应的设置。例如,ToolBarStyle 有 3 种风格可供选择:ImageAndText、TextOnly、ImageOnly,用户可以根据应用程序的需求来设置。

6.2.6 Toc 控件

Toc 控件全称 Table of Contents, Toc 控件列出 Map 控件中所有的图层。Toc 控件与 Map 控件相关联,打开或关闭 Toc 控件的图层可以控制 Map 控件图层可见或不可见。Toc 控件从 TreeViewPlus 控件继承而来,可以用代码来控制 TreeViewPlus 中的节点。Toc 控件是异步刷新的,更新 Toc 控件不需要刷新整个页面。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.Toc。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_toc.js, display_treeviewplus.js。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点,在设计模式下打开一个页面,选择工具箱并展开 ArcGIS 控件标签,选中 Toc 控件,然后将它拖到 Web 页面中,如图 6.29 所示。

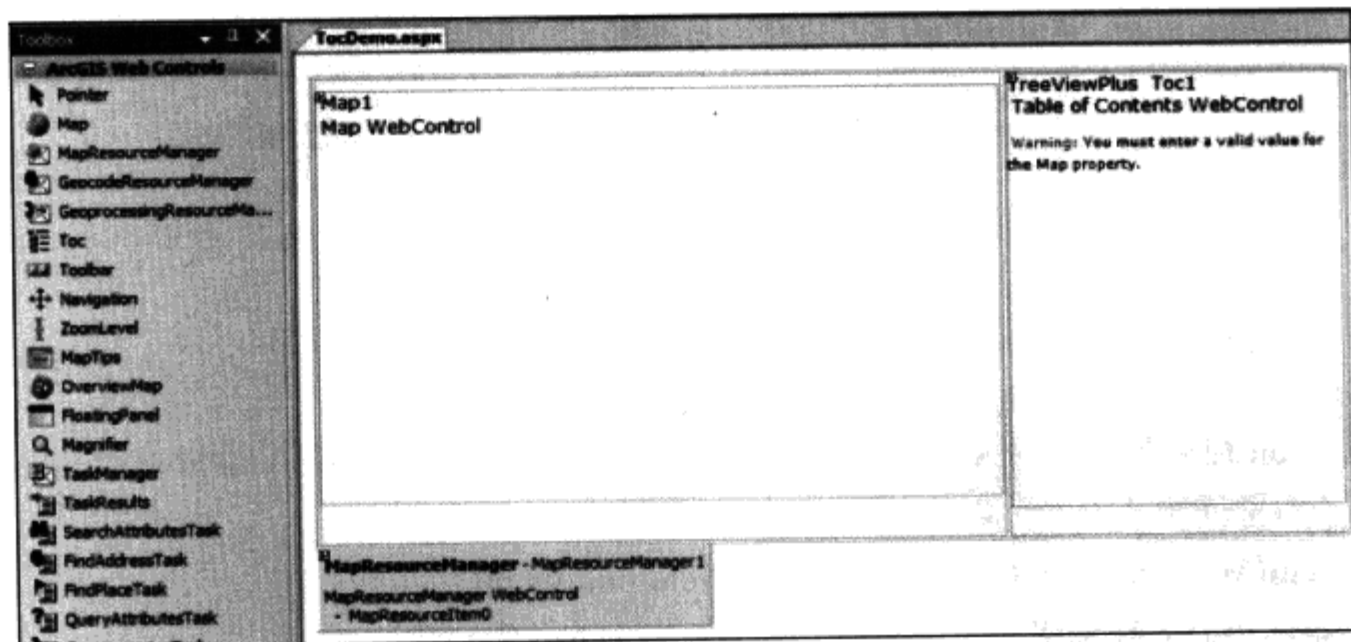


图 6.29 添加 Toc 控件

(2) 定义 Toc 控件。

定义 Toc 控件的 BuddyControl 属性,如图 6.30 所示。

6.2.7 ScaleBar 控件

ScaleBar 控件为 Map 提供一个简单的比例尺。ScaleBar 控件的一些属性如颜色、字体、高度和比例尺的单位都可以通过属性来设置。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.ScaleBar。
- **JavaScript libraries** display_common.js, display_dotnetadf.js。

ScaleBar 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 ScaleBar 控件，然后将它拖到 Web 页面中，如图 6.31 所示。

(2) 设置 ScaleBar 控件属性。

设置 ScaleBar 控件的 Map、BarUnits 等属性，如图 6.32 所示。

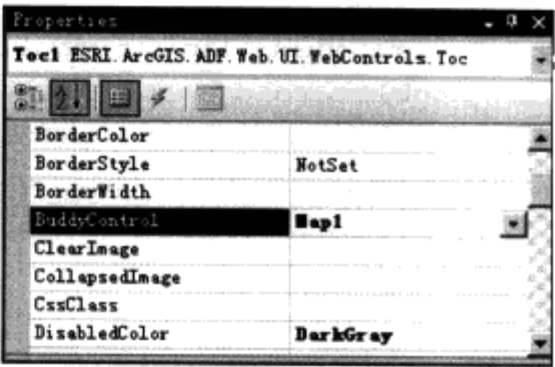


图 6.30 设置 Toc 控件属性

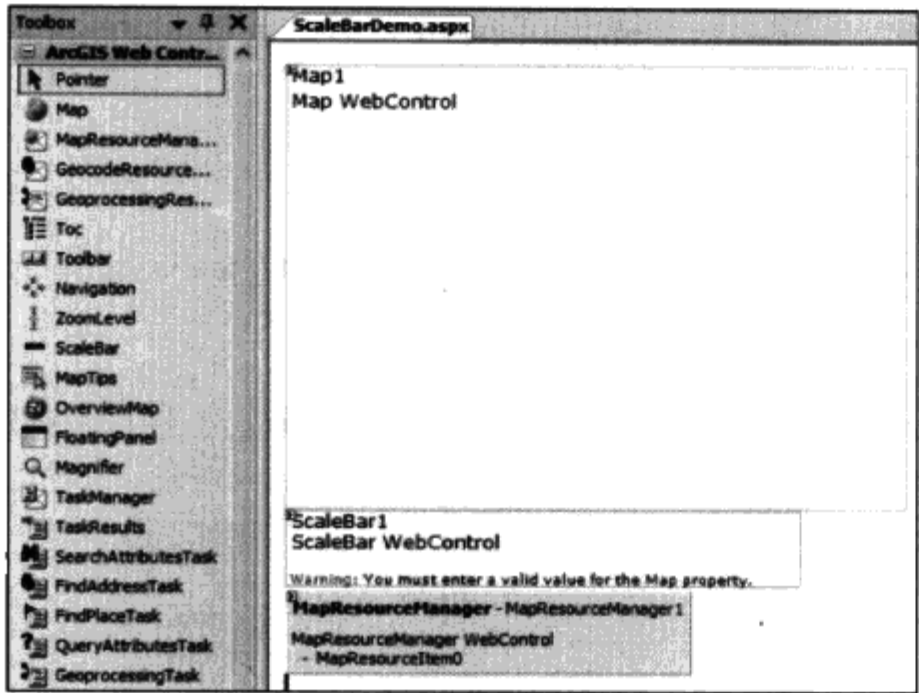


图 6.31 添加 ScaleBar 控件

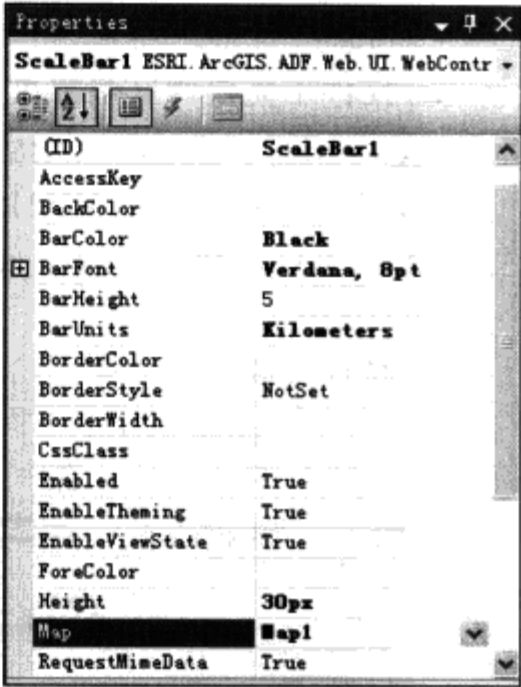


图 6.32 设置 ScaleBar 属性

6.2.8 Navigation 控件

Navigation 控件与单个的 Map 控件相关联，它提供通用的地图移动、缩放功能。Navigation 控件可以被渲染成图片，也可以是 TrueType 字符。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.Navigation。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_navigation.js。

Navigation 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展

开 ArcGIS 控件标签，选中 Navigation 控件，然后将它拖到 Web 页面中，如图 6.33 所示。

(2) 设置 Navigation 控件的属性。

设置 Navigation 控件的 Map、DisplayCharacter 等属性，如图 6.34 所示。DisplayCharacter 表示系统提供多种显示字符，单击下拉列表，开发人员可根据需要选择字符，如图 6.35 所示。

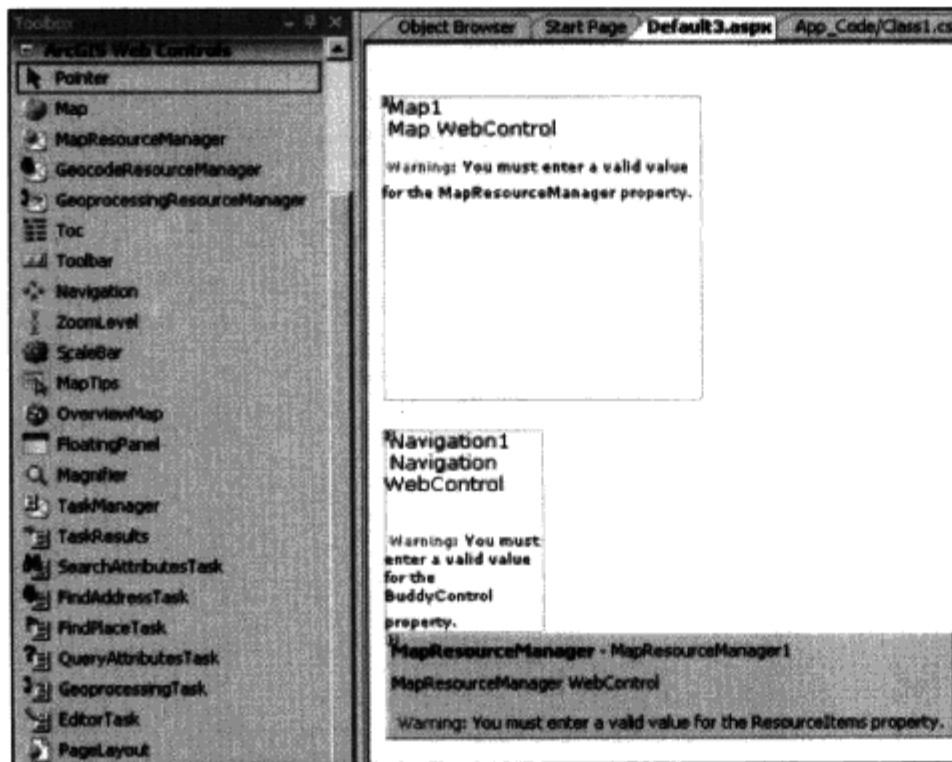


图 6.33 添加 Navigation 控件

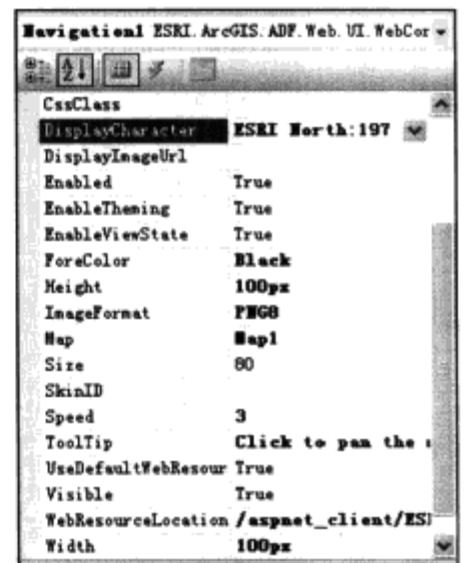


图 6.34 设置 Navigation 属性

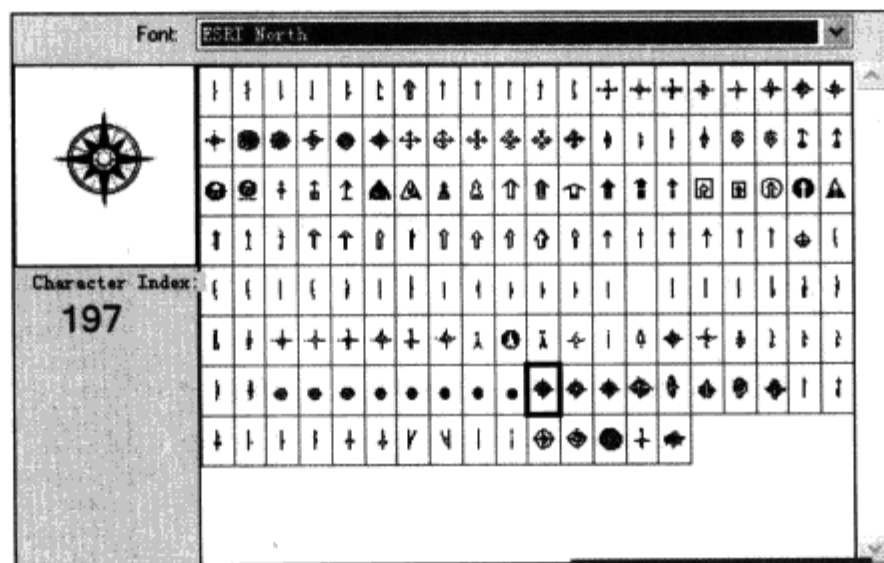


图 6.35 DisplayCharacter 属性字符设置

6.2.9 ZoomLevel 控件

ZoomLevel 控件能够让用户快速定位到某一比例尺。它与 Map 控件相关联，开发人员可以将其放在页面上的任何地方。ZoomLevel 控件只针对建立了 Cache 的地图服务，如果是实时产生图片的服务则 ZoomLevel 控件没有任何作用。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.ZoomLevel。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_navigation.js。

ZoomLevel 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 ZoomLevel 控件，然后将它拖到 Web 页面中，如图 6.36 所示。

(2) 设置 ZoomLevel 控件的属性。

设置 ZoomLevel 控件的 Map 等属性，如图 6.37 所示。

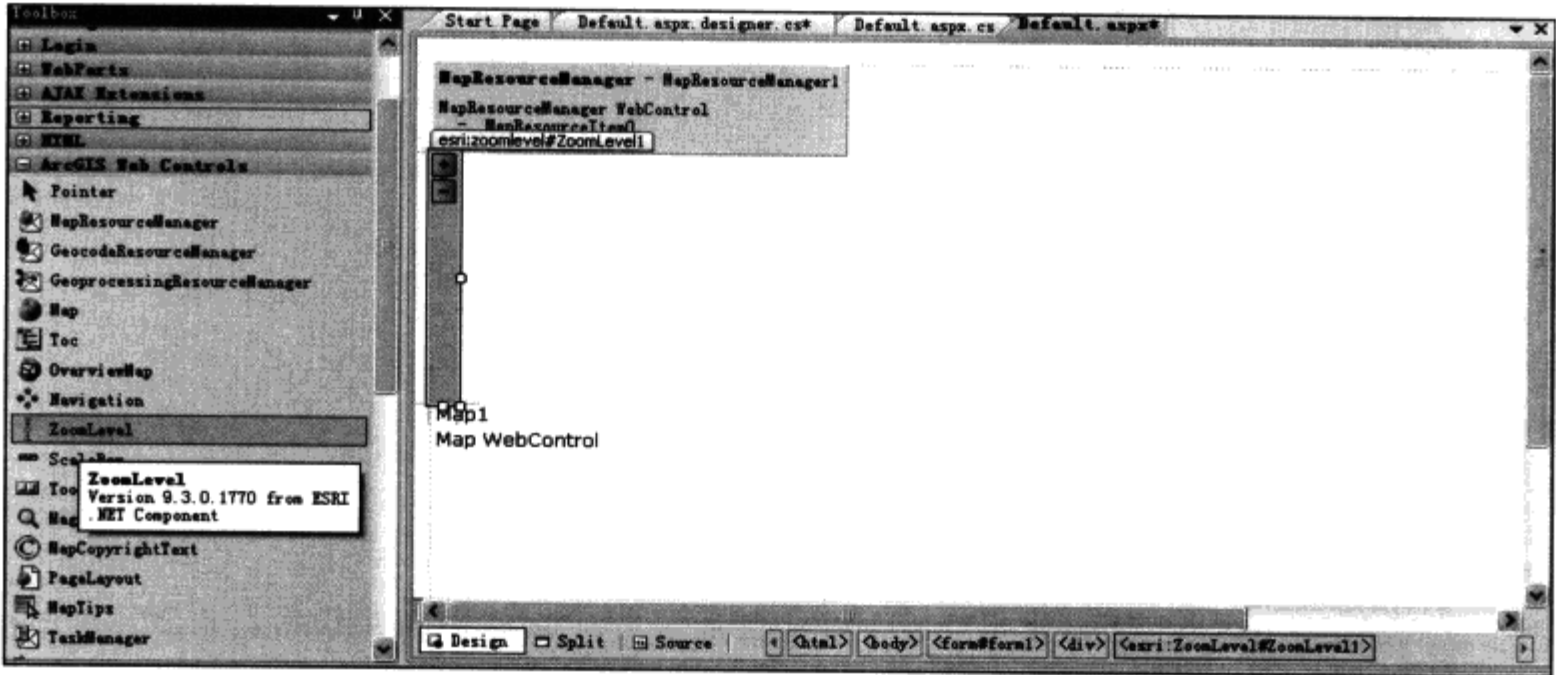


图 6.36 添加 ZoomLevel 控件

6.2.10 MapCopyrightText 控件

MapCopyrightText 控件用来显示 Map 控件的版权。在运行的时候，MapCopyrightText 控件以文本的方式出现，单击后显示版权信息的对话框。

在 ArcMap 中可以设置图层的版权信息，在图层的属性对话框中，在“General”页面的“Credits”文本框中可以输入版权信息，单击 MapCopyrightText 控件可以显示图层的版权信息。

- **Assembly** ESRI.ArcGIS.ADF.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.UI.WebControls.MapCopyrightText。
- MapCopyrightText 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 MapCopyrightText 控件，然后将它拖到 Web 页面中，如图 6.38 所示。

(2) 设置控件属性。

在 MapCopyrightText 控件的属性对话框中，设置 CopyrightText、Map、Text 等属性，如图 6.39 所示。

CopyrightText 控件的运行结果如图 6.40 所示。

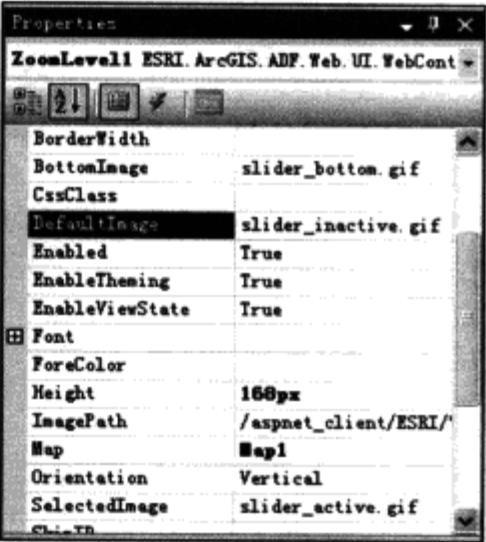


图 6.37 设置 ZoomLevel 属性

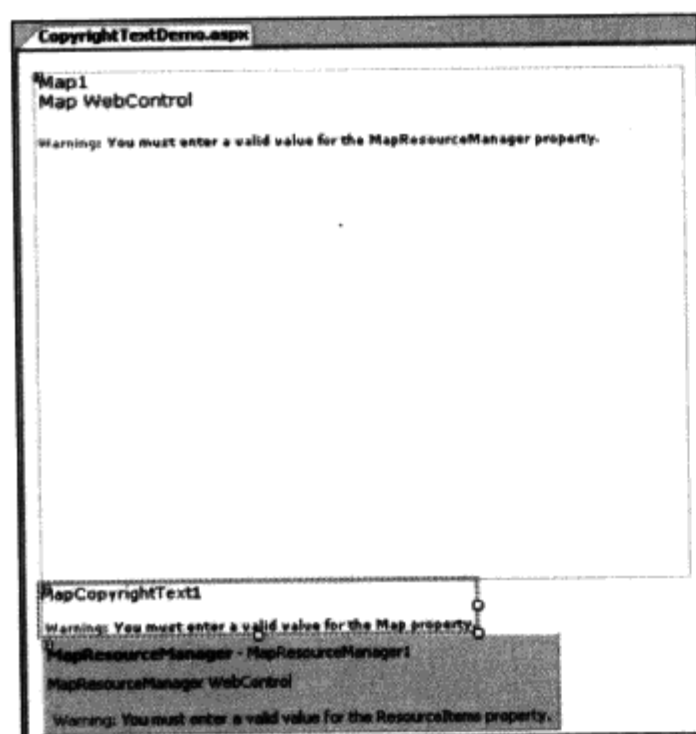


图 6.38 添加控件



图 6.39 设置属性界面

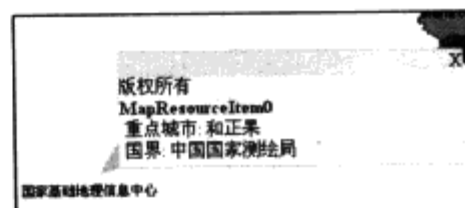


图 6.40 CopyrightText 的运行结果

6.3 TaskManger 与 Task 控件

在 ArcGIS Server 中，所有的 Task 均由 TaskManager 来管理。本节将介绍 ArcGIS Server 提供的各种 Task 和 TaskManager 控件及 Task 结果的显示。

6.3.1 TaskManager 控件

TaskManager 控件在 Web ADF 应用程序中管理各种 Tasks。TaskManager 控件产生树形 XML 格式的数据，这种数据能够被 ASP.NET 的 Menu、TreeView 等控件导航。在运行的时候，Menu、TreeView 控件可以用来控制 Task 的显示。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.TaskManager。
 - **JavaScript libraries** display_common.js, display_dotnetadf.js。
- TaskManager 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 TaskManager 控件，然后将它拖到 Web 页面中，如图 6.41 所示。

(2) 添加 Menu 控件到页面，如图 6.42 所示。

(3) 设置 TaskManager 控件的 BuddyControl 属性，如图 6.43 所示。

(4) 添加一个 Task 到 TaskManager 中。

在 ArcGIS Web Controls 工具箱中选择一个 Task 或者自定义的 Task 控件，并把它拖到 TaskManager 中。在一般情况下要把 Task 控件设置为

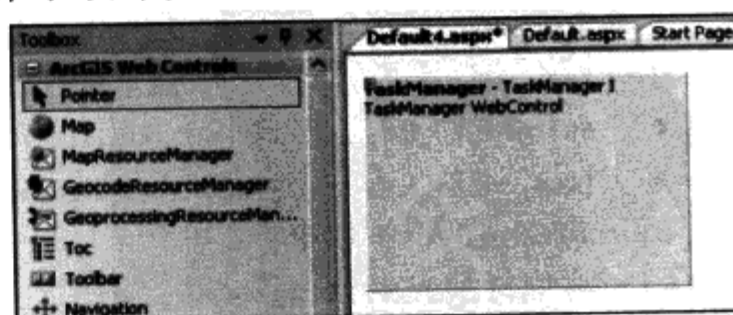


图 6.41 添加 TaskManager 控件

False，定义一个菜单项作为菜单的 Task 项，系统的配置如图 6.44 所示。
Task 的运行界面如图 6.45 所示。

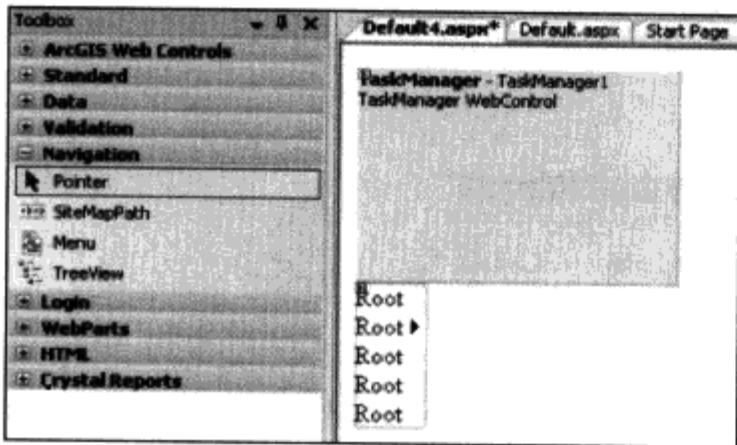


图 6.42 添加 Menu 控件

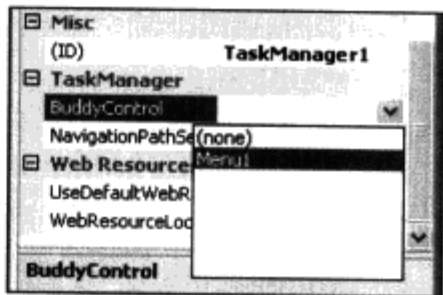


图 6.43 设置 TaskManager 属性界面

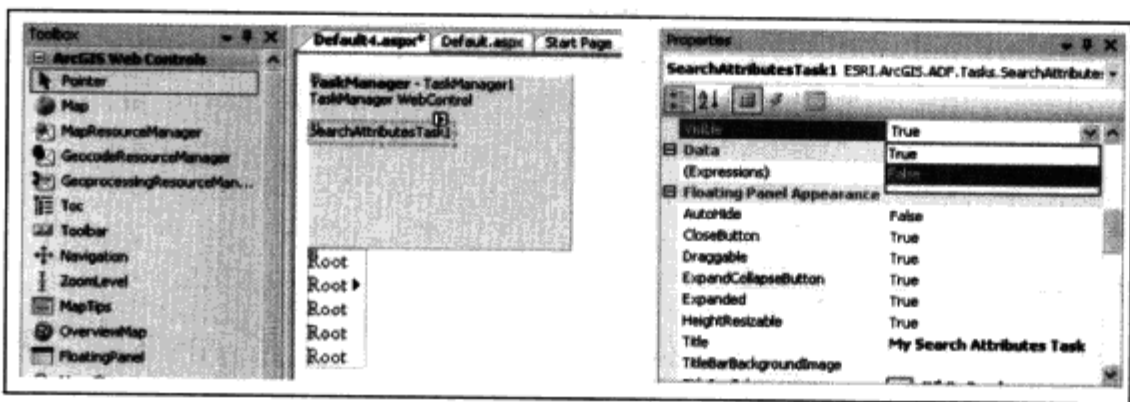


图 6.44 添加 Task 设置 Task 属性



图 6.45 Task 的运行界面

6.3.2 EditorTask 控件

EditorTask 控件为 ArcGIS Server 地图服务提供一套基于 Web 的编辑工具，可以修改、添加、删除几何元素或修改其属性。

使用 EditorTask 控件需要满足以下条件：

- 地图服务必须是 ArcGIS Server Local 类型；
- 地图服务可以是 pooled 或者 non-pooled；
- 地图服务必须是基于 MXD 文档，不支持 MSD 服务；
- 在 non-pooled 的服务中，有没有注册版本均可编辑；pooled 的服务只能编辑没有注册版本的数据；
- EditorTask 中不支持 Shapefile 和 personal geodatabase 的数据编辑；
- 待编辑的数据图层最好有唯一的名字；
- **Assembly** ESRI.ArcGIS.ADF.ArcGISSEditor.dll；
- **Class** ESRI.ArcGIS.ADF.ArcGISSEditor.EditorTask。

EditorTask 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，在使用 EditorTask 的页面中至少需要 MapResourceManager 和 Map，如图 6.46 所示。

(2) 设置 EditorTask 控件属性。

可以选择 EditorTask 的属性框进行属性设置，也可以选中 EditorTask 控件，单击“>”按钮进行必要的属性设置，如图 6.47 所示。

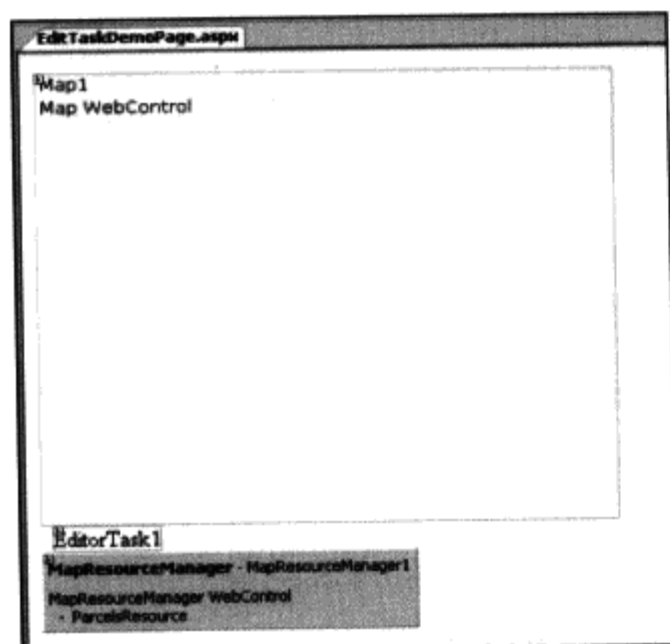


图 6.46 添加控件

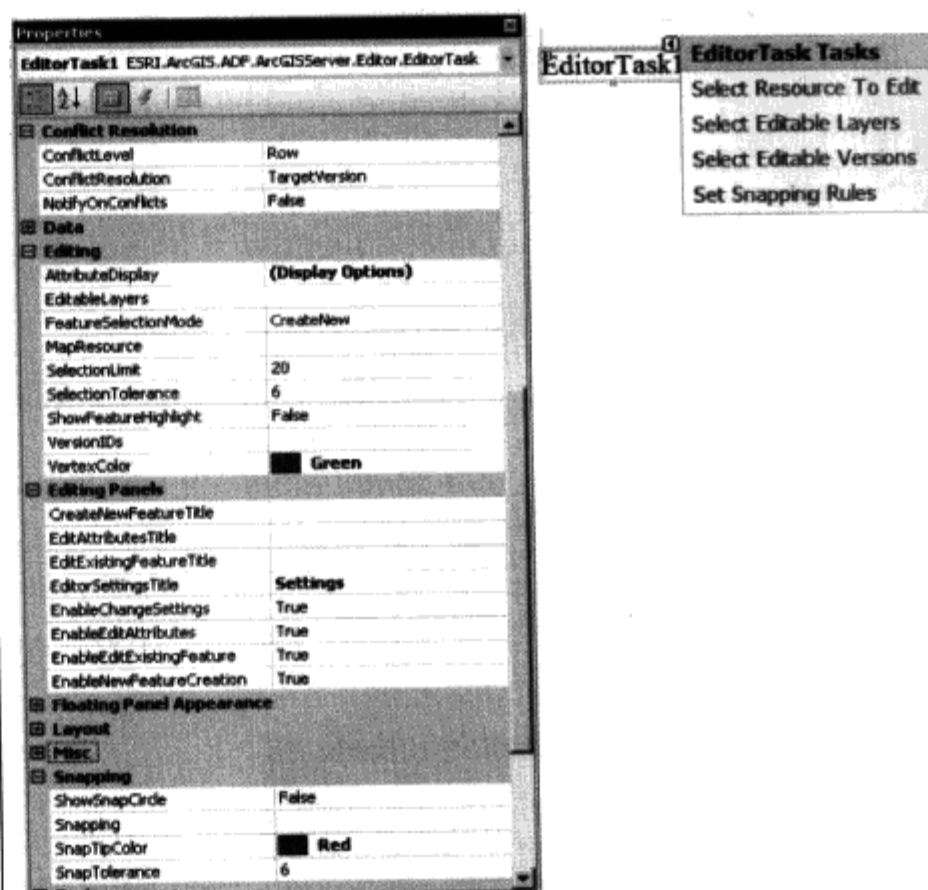


图 6.47 设置 EditorTask 属性

- ① 设置编辑的数据源，如图 6.48 所示。
- ② 设置可编辑的图层，如图 6.49 所示。

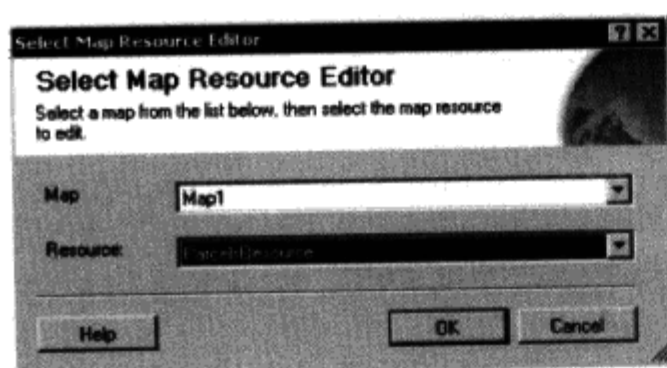


图 6.48 设置数据源

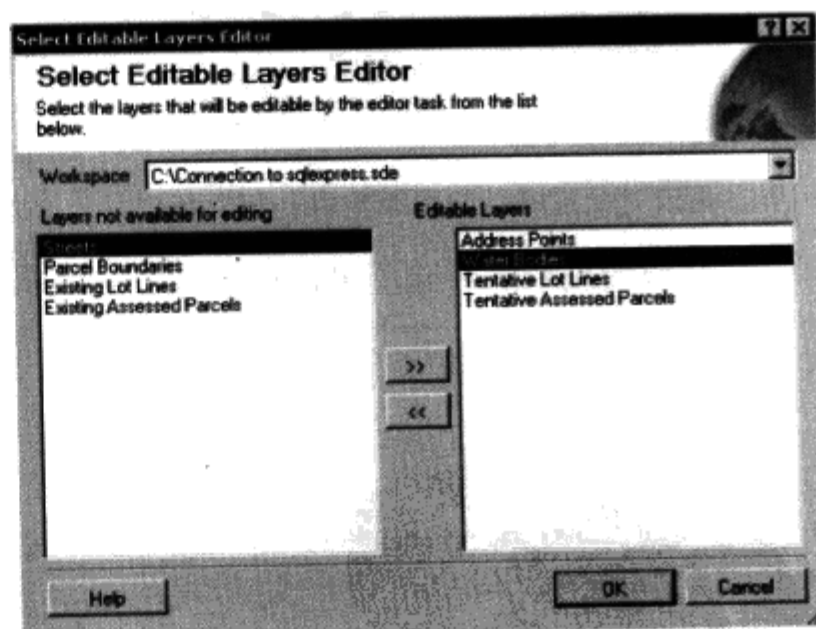


图 6.49 设置可编辑的图层

- ③ 设置编辑的版本，如图 6.50 所示。
 - ④ 设置捕捉，如图 6.51 所示。
- (3) 设置属性的可编辑性。

用户可以利用 AttributeDisplay 来设置图层的哪些属性是可以用来编辑，哪些属性是不能编辑的，如图 6.52 所示。

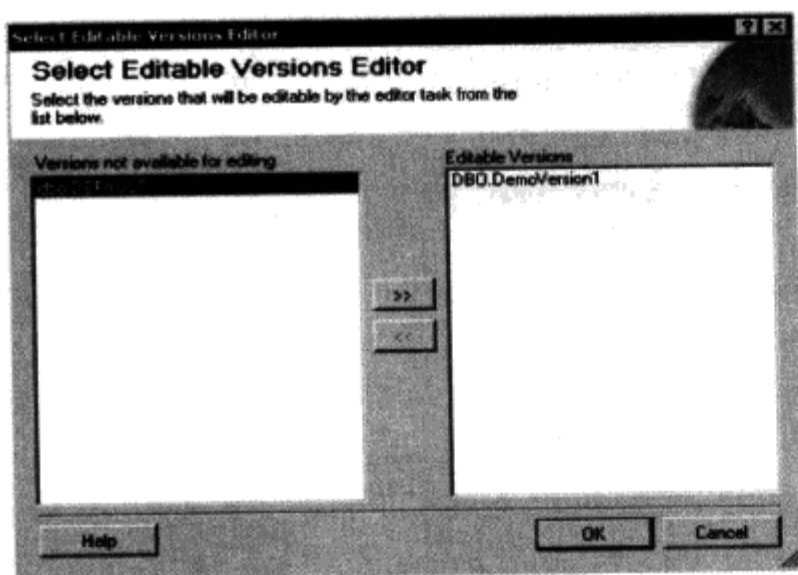


图 6.50 设置编辑图层的版本

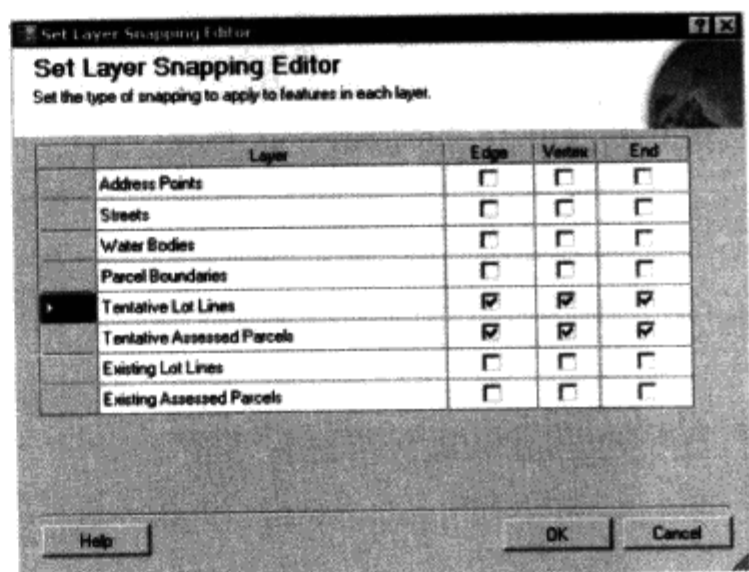


图 6.51 设置编辑捕捉

(4) 添加 ArcGIS Identity。

由于地图服务采用 ArcGIS Server Local 的方式，所以必须添加 ArcGIS Identity。选中项目，单击鼠标右键，系统弹出菜单如图 6.53 所示，选中“Add ArcGIS Identity”选项，系统就会弹出“ArcGIS Identify”对话框。

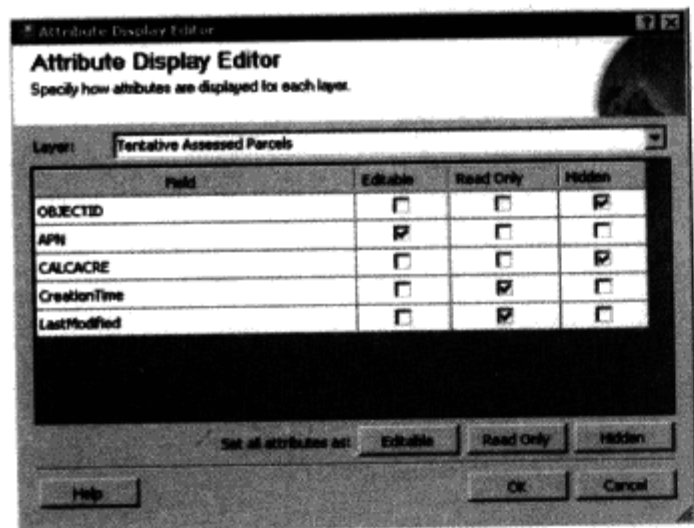


图 6.52 设置属性的可编辑性

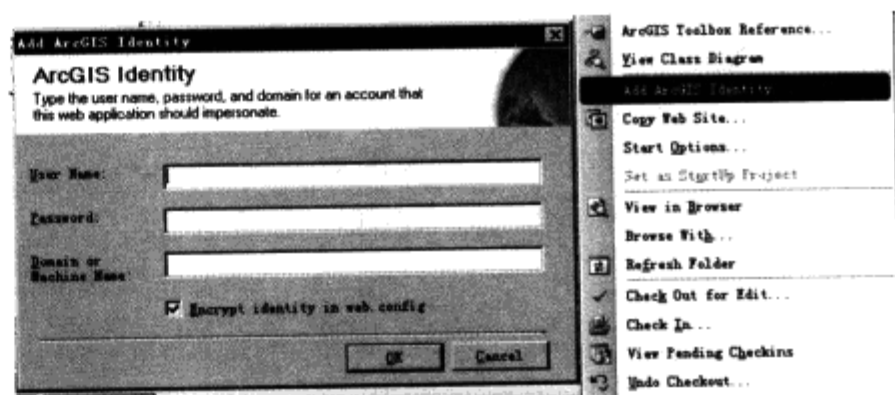


图 6.53 添加 ArcGIS Identity

图层字段的编辑属性除了可以设置外，还可以通过编程来控制，示例代码如下：

```
private void FilterAttributes()  
{  
    AttributeDisplayInfo adi = new AttributeDisplayInfo(5, AttributeDisplayMode.  
Hidden);  
    adi.Overrides.Add(new AttributeDisplayOverride("STATE_NAME", AttributeDisplayMode.  
ReadOnly));  
    adi.Overrides.Add(new AttributeDisplayOverride("STATE_FIPS", AttributeDisplayMode.  
ReadOnly));  
    adi.Overrides.Add(new AttributeDisplayOverride("SUB_REGION", AttributeDisplayMode.  
ReadOnly));  
    adi.Overrides.Add(new AttributeDisplayOverride("POP1990", AttributeDisplayMode.  
ReadOnly));  
    adi.Overrides.Add(new AttributeDisplayOverride("POP1999", AttributeDisplayMode.  
Editable));  
}
```

```

        adi.Overrides.Add(new AttributeDisplayOverride("AVG_SALE87", AttributeDisplayMode.
Editable));
        EditorTask1.AttributeDisplay.AttributeDisplayInfos.Add(adi);
    }

```

6.3.3 SearchAttributesTask 控件

SearchAttributesTask 控件提供用户输入字段的属性值来查找几何元素,待查找的图层只能是 MapResourceManager 中的图层,查找结果由 ADO.NET DataSet 返回到 TaskResults 中。

- **Assembly** ESRI.ArcGIS.ADF.Tasks.dll。
- **Class** ESRI.ArcGIS.ADF.Tasks.SearchAttributesTask。
- **JavaScript libraries** display_common.js,display_dotnetadf.js,display_task.js, display_floatingpanel.js。

SearchAttributesTask 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点,在设计模式下打开一个页面,选择工具箱并展开 ArcGIS 控件标签,在使用 SearchAttributesTask 的页面中至少需要 MapResourceManager、Map、TaskResults,如图 6.54 所示。

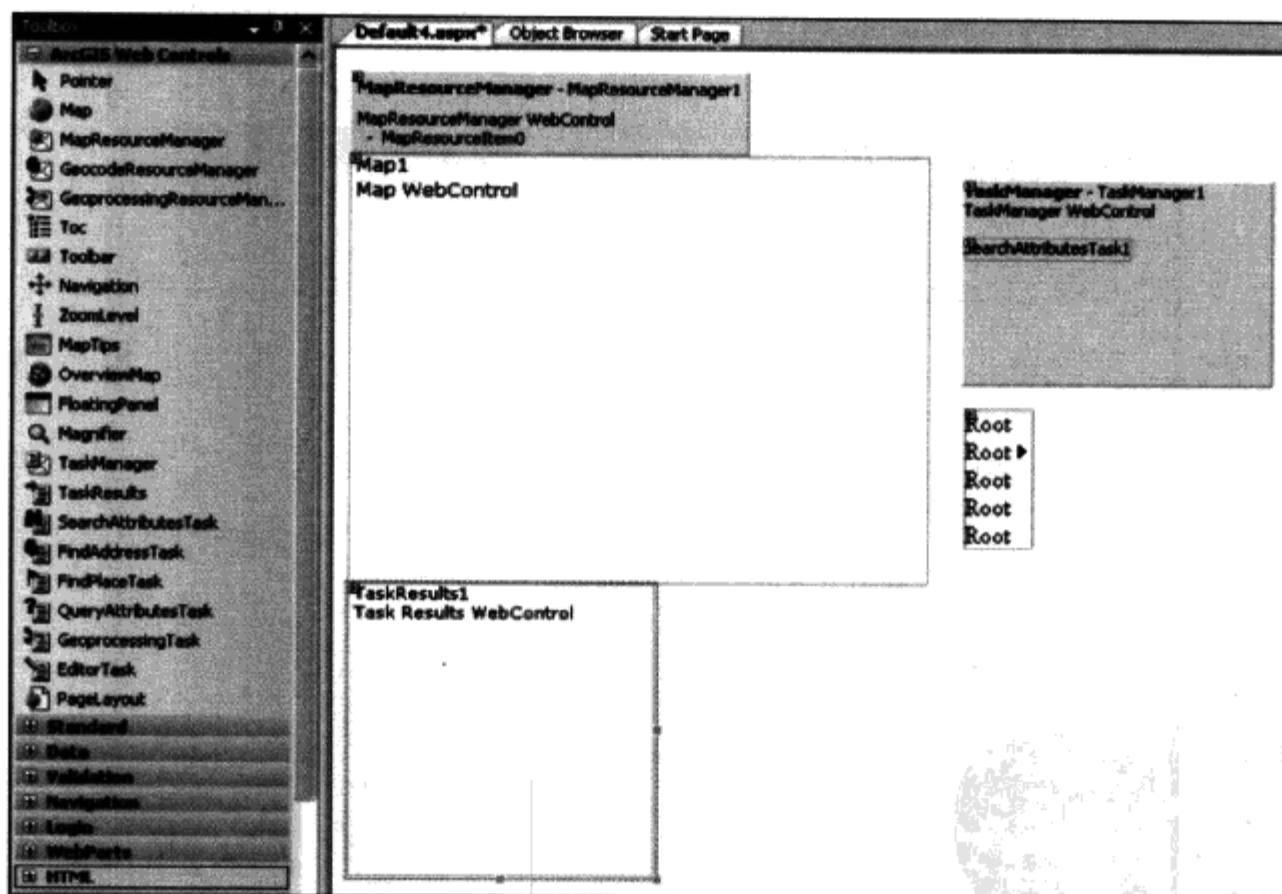


图 6.54 添加控件

(2) 设置控件属性。

① 设置 SearchAttributesTask 结果显示控件,如图 6.55 所示。

② 设置查询字段,如图 6.56 所示。注意设置查询字段的时候,可以设置一个字段,也可以设置多个字段,还可以设置多个图层。

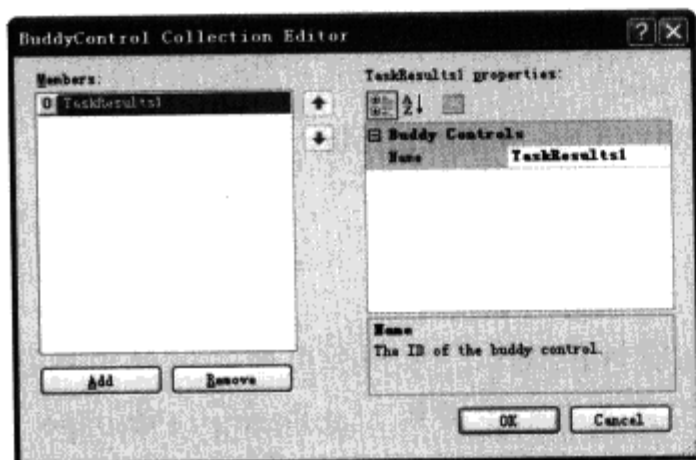


图 6.55 设置结果显示控件

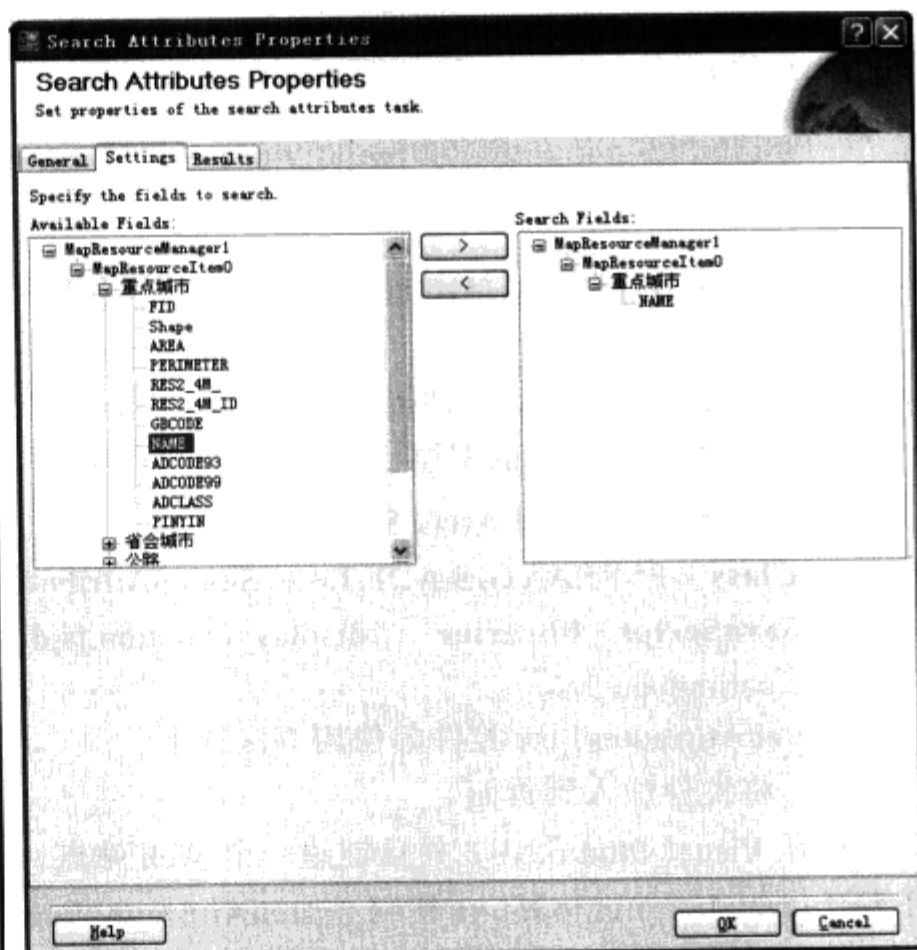


图 6.56 设置查询属性

运行结果如图 6.57 所示。

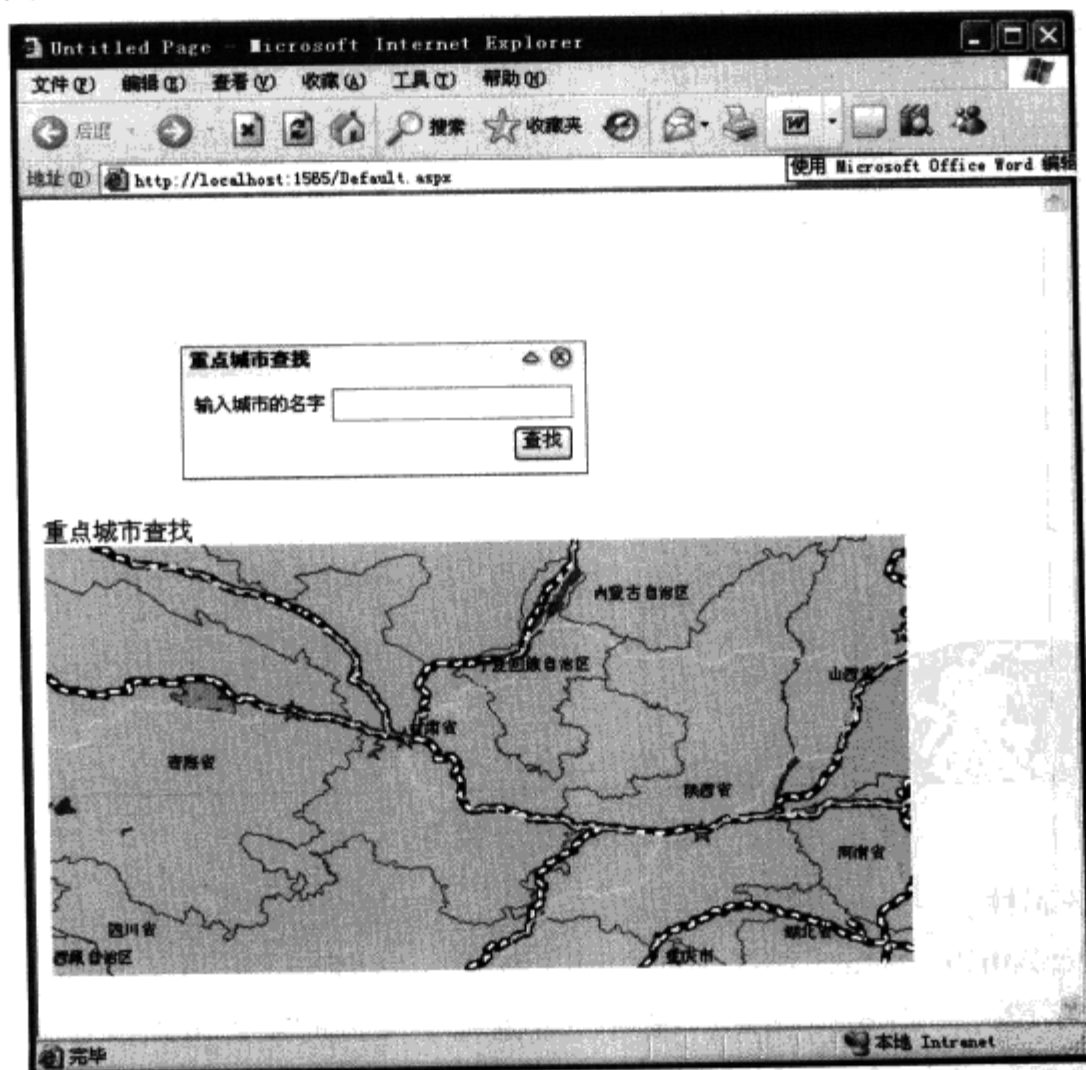


图 6.57 系统运行界面

6.3.4 QueryAttributesTask 控件

QueryAttributesTask 控件能够让用户确切定义一个字段的值，该控件提升了 SearchAttributesTask 控件的功能，SearchAttributesTask 找到的时候使用关系操作符是 like，而 QueryAttributesTask 字段可以是“=”、“>”、“like”等操作符，查询的结果与 SearchAttributesTask 相似。

- **Assembly** ESRI.ArcGIS.ADF.Tasks.dll。
- **Class** ESRI.ArcGIS.ADF.Tasks.QueryAttributesTask。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_task.js, display_floatingpanel.js。

QueryAttributesTask 控件的使用与 SearchAttributesTask 相似，在此不再重复讲解。唯一不同的是设置查询字段有些许区别，如图 6.58 所示。

在设置查询字段的时候，可以自定义查询字段。单击“Add”菜单，系统弹出查询字段条件编辑对话框，开发人员可以根据实际情况来设置查询条件。

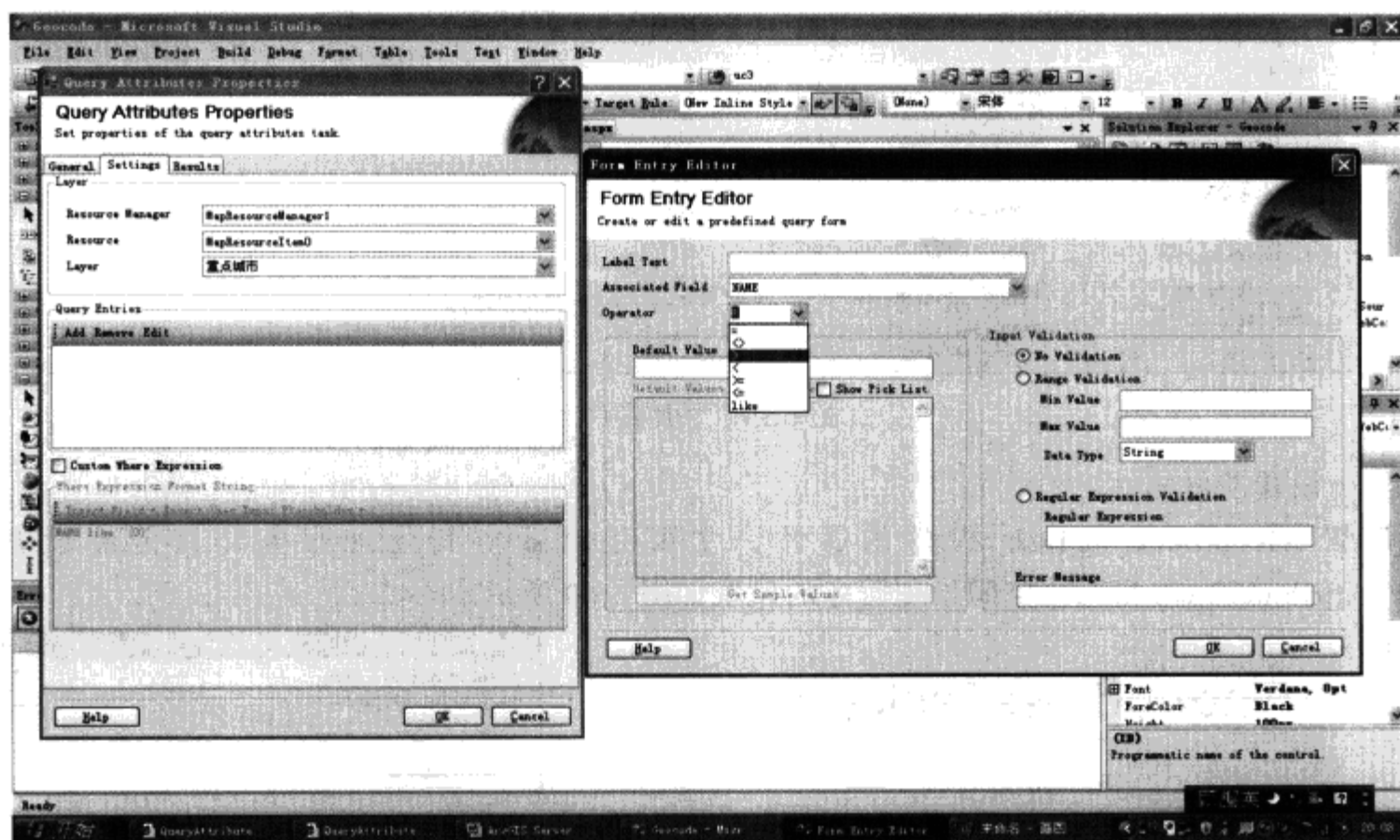


图 6.58 编辑查询条件

6.3.5 GeoprocessingTask 控件

GeoprocessingTask 控件利用 GeoprocessingResourceManager 控件数据处理服务来执行空间处理服务。GeoprocessingTask 控件根据用户的输入参数来执行空间处理，并把处理结果显示在 TaskResults 中。

- **Assembly** ESRI.ArcGIS.ADF.Tasks.dll。
- **Class** ESRI.ArcGIS.ADF.Tasks.GeoprocessingTask。

- **JavaScript libraries** display_common.js,display_dotnetadf.js,display_task.js, display_floatingpanel.js。

GeoprocessingTask 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签。在使用 GeoprocessingTask 的页面中至少需要 MapResourceManager、GeoprocessingResourceManager、Map、TaskResults，如图 6.59 所示。

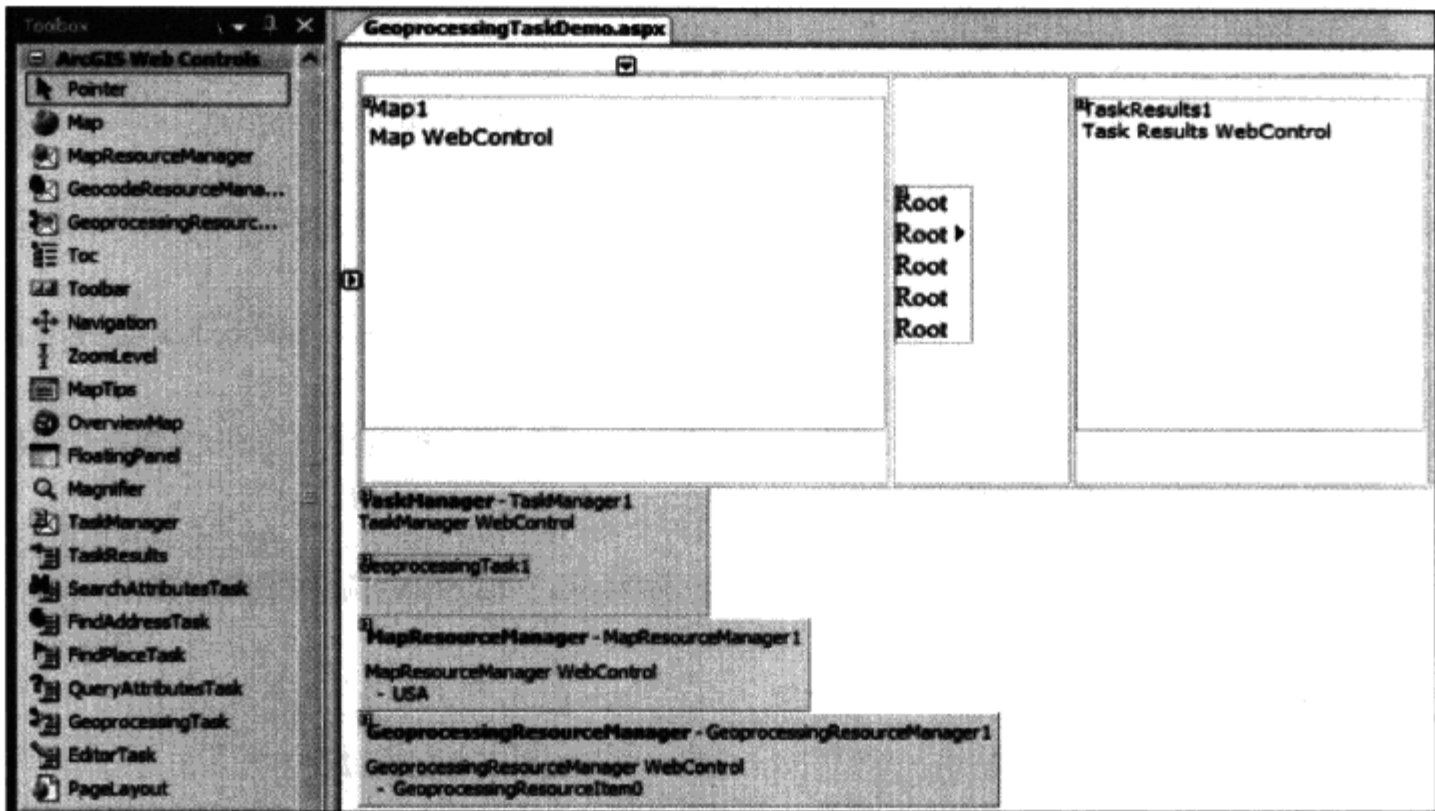


图 6.59 添加控件

(2) 设置控件属性。

设置 GeoprocessingTask 控件的属性，主要是设置空间处理服务，其他的设置与 SearchAttributesTask 相同。空间处理服务的设置如图 6.60 所示。

6.3.6 FindAddressTask 控件

FindAddressTask 控件利用 GeocodeResourceManager 中地理编码来执行地理编码操作，地理编码是把一个地址描述转变为地图的 x, y 坐标。FindAddressTask 控件把用户输入的地址描述匹配到地址库中，结果以 ADO.NET 的 DataSet 显示在 TaskResults 中。

- **Assembly** ESRI.ArcGIS.ADF.Tasks.dll。
- **Class** ESRI.ArcGIS.ADF.Tasks.FindAddressTask。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_task.js, display_floatingpanel.js。

FindAddressTask 控件的使用方法如下。

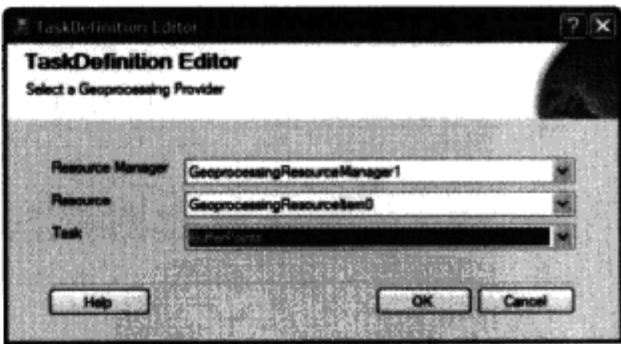


图 6.60 设置空间处理服务

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点, 在设计模式下打开一个页面, 选择工具箱并展开 ArcGIS 控件标签。在使用 FindAddressTask 的页面中至少需要 MapResourceManager、GeocodeResourceManager、Map、TaskResults, 如图 6.61 所示。

(2) 设置控件属性。

选中 FindAddressTask 控件, 在其属性编辑框设置其属性, 如图 6.62 所示。

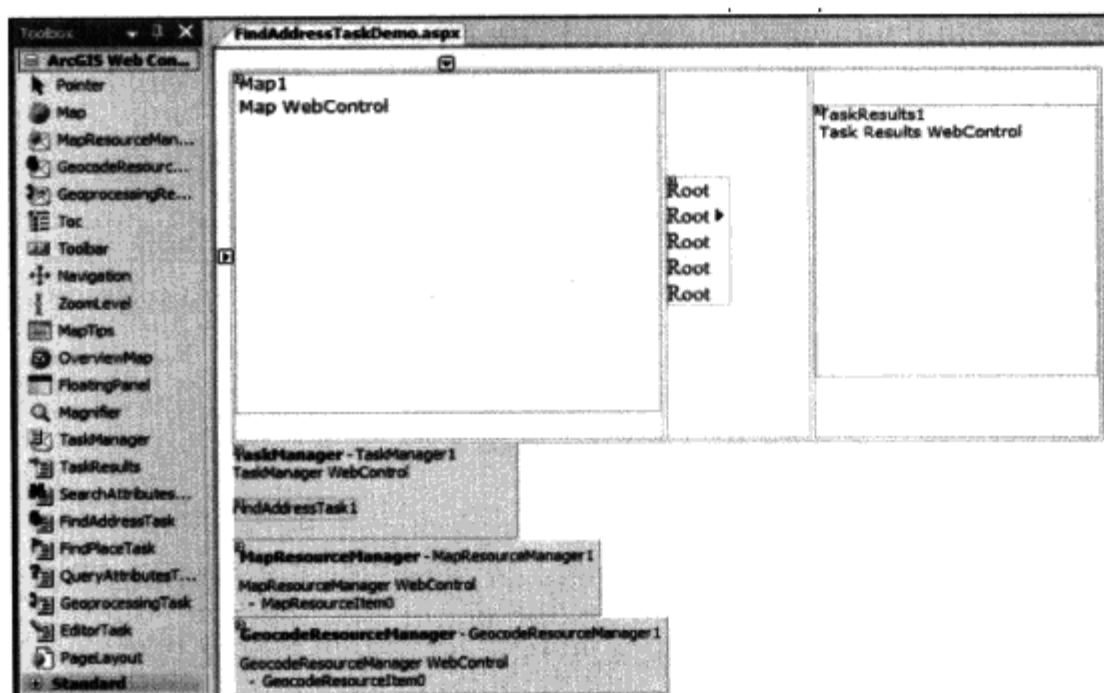


图 6.61 添加控件

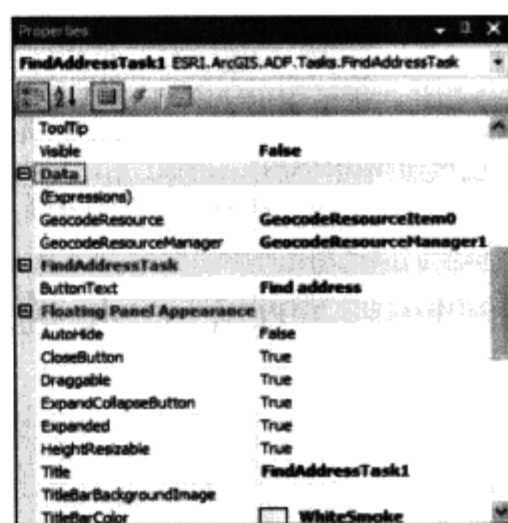
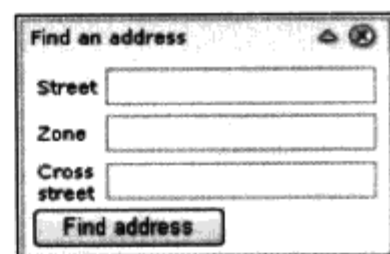


图 6.62 设置控件属性

FindAddressTask 控件运行界面如图 6.63 所示。



6.3.7 FindPlaceTask 控件

FindPlaceTask 控件利用提供地名搜索 ArcWeb Services。

FindPlaceTask 控件包括一个文本框用来输入地名, 结果以 ADO.NET 的 DataSet 显示在 TaskResults 中。在使用 FindPlaceTask 时必须使用 ArcWeb Services 的账号。

- **Assembly** ESRI.ArcGIS.ADF.Tasks.dll。
- **Class** ESRI.ArcGIS.ADF.Tasks.FindPlaceTask。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_task.js, display_floatingpanel.js。

FindPlaceTask 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点, 在设计模式下打开一个页面, 选择工具箱并展开 ArcGIS 控件标签。在使用 FindPlaceTask 的页面中至少需要 MapResourceManager、Map、TaskResults, 如图 6.64 所示。

(2) 设置控件属性。

设置 ArcWeb Source, 输入用户名、密码, 单击“Service”下拉列表框, 选择一个 ArcWeb Service, 如图 6.65 所示。

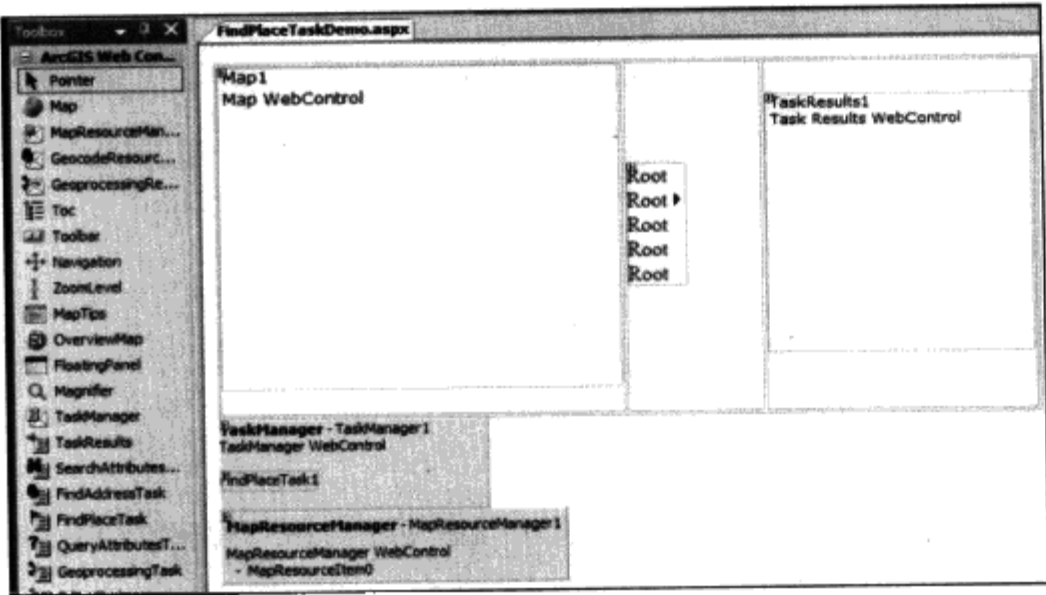


图 6.64 添加控件

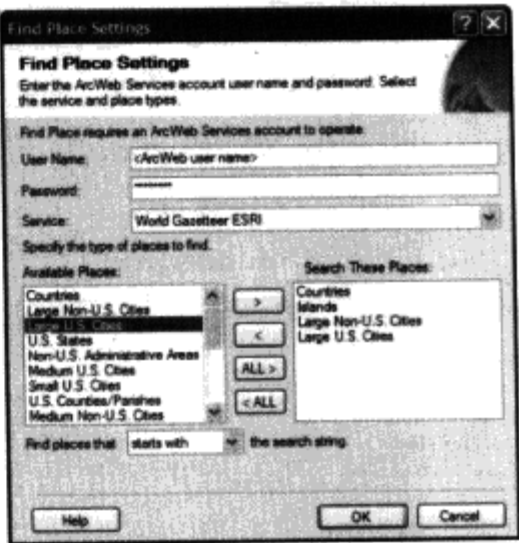


图 6.65 设置 FindPlaceTask 属性界面

FindPlaceTask 控件的运行界面如图 6.66 所示。

6.3.8 PrintTask 控件



图 6.66 FindPlaceTask 控件运行界面

PrintTask 控件顾名思义就是打印地图的控件。
PrintTask 控件把当前 Map 控件显示的图片嵌入到一个打印网页中去,可以设置打印文档的字头、图片大小、图例等。

- **Assembly** ESRI.ArcGIS.ADF.Tasks.dll。
- **Class** ESRI.ArcGIS.ADF.Tasks.PrintTask。
- **JavaScript libraries** display_common.js,display_dotnetadf.js,display_task.js,display_floatingpanel.js。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点,在设计模式下打开一个页面,选择工具箱并展开 ArcGIS 控件标签。在使用 PrintTask 的页面中至少需要 MapResourceManager、Map、TaskResults,如图 6.67 所示。

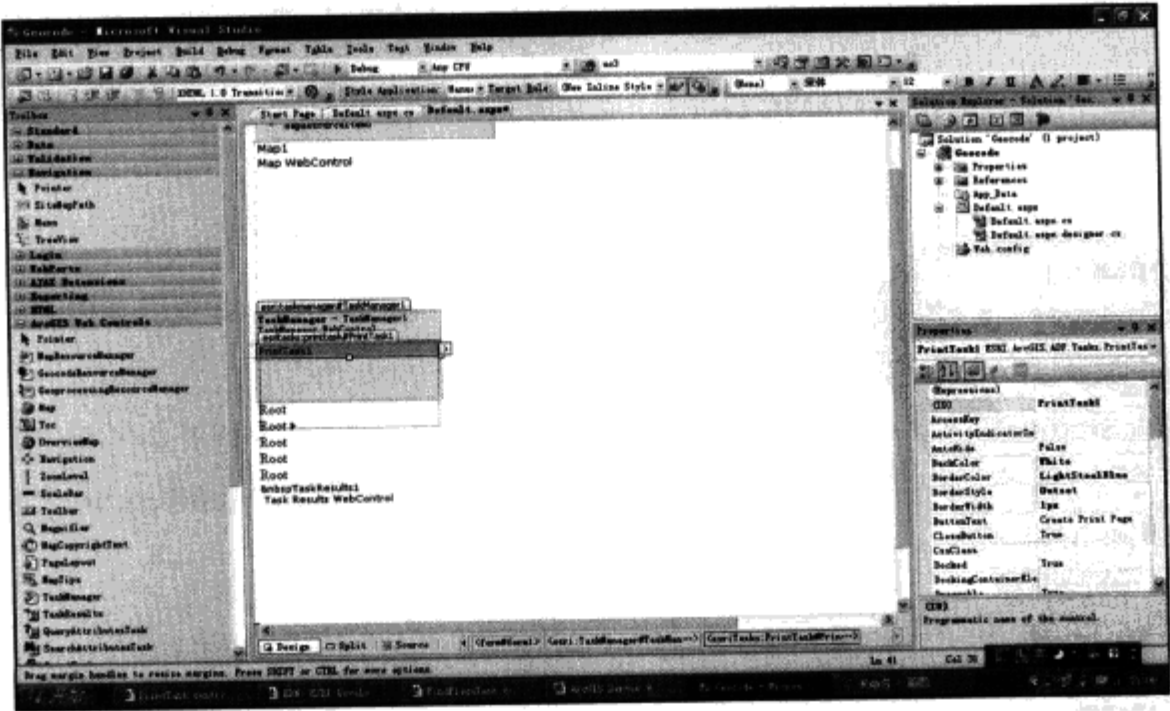


图 6.67 添加控件

(2) 设置控件属性。

- ① 设置要打印的图例，如图 6.68 所示。
- ② 设置图片的大小 (WidthSettings)，如图 6.69 所示。
- ③ 设置是否打印查询结果，如图 6.70 所示。



图 6.68 选择打印图例

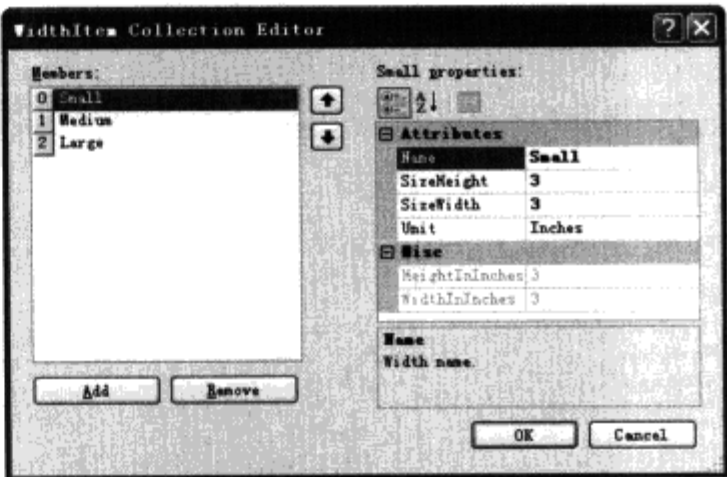


图 6.69 设置图片大小

PrintTask 运行界面如图 6.71 所示。

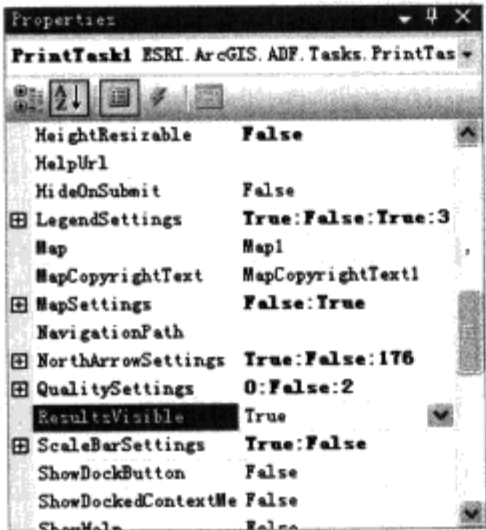


图 6.70 设置是否打印查询结果

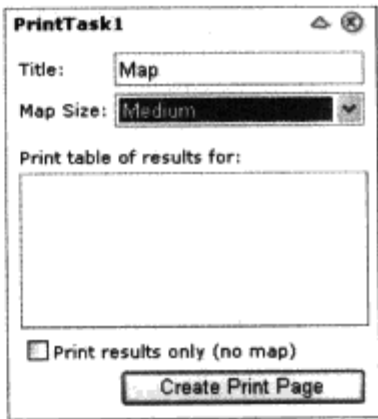


图 6.71 PrintTask 运行时界面

单击“Create Print Page”按钮，系统会弹出打印页面，如图 6.72 所示。

6.3.9 TaskResults 控件

TaskResults 控件用来显示其他 Task 操作结果，它以 ADO.NET 的 DataSet 形成存储，以 TreeView 形式显示。TaskResults 控件提供异步刷新的放大、平移以及高亮显示选中的对象，也可以在结果集删除选择的对象。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.TaskResult。
- **JavaScript libraries** display_common.js, display_dotnetadf.js, display_taskresults.js, display_treeviewplus.js, display_contextmenu.js。

TaskResults 一般是与其他的 Task 一起使用，方法前面的章节已经介绍过，在此不再重述。

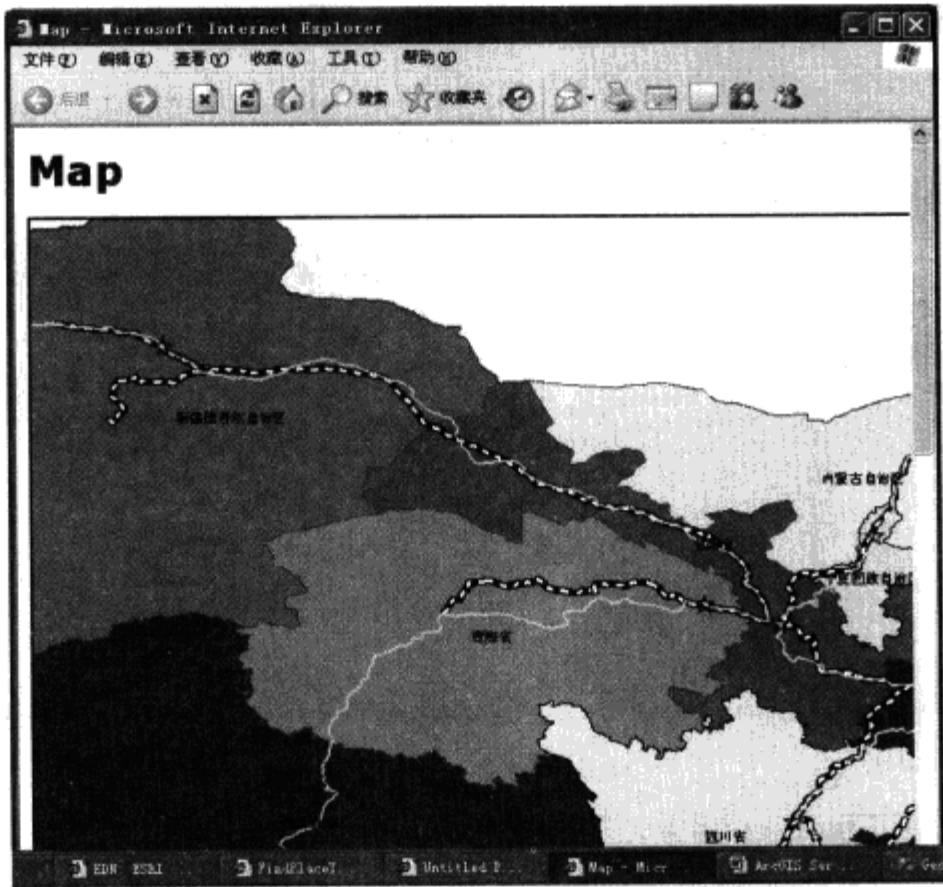


图 6.72 打印页面

6.4 其他控件

下面的控件与地图显示没有直接的关系，然而在开发 ArcGIS Server 应用程序的过程中，我们也经常会用到。

6.4.1 FloatingPanel 控件

FloatingPanel 控件允许用户拖动，它的面板可以自动浮动在页面上。这个控件可以用于其他的控件，包括 ASP.NET 提供的标准控件。Web ADF 的很多控件都是基于 FloatingPanel 的，如 Navigation、Magnifier 和各种 Task 控件。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.FloatingPanel。
 - **JavaScript libraries** display_common.js, display_dotnetadf.js, display_floatingpanel.js。
- FloatingPanel 控件的使用方法如下。

(1) 打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，编辑一个 UserControl，如图 6.73 所示。



图 6.73 编辑 UserControl

(2) 在页面上加入 UserControl 和 FloatingPanel，并把 UserControl 嵌入到 FloatingPanel 中，脚本程序如下：

```
<%-- 按地名定位 --%>
<esri:FloatingPanel ID="FloatingPanel1" runat="server" BackColor="White"
    BorderColor="Gray" BorderStyle="Solid" BorderWidth="1px"
    Font-Names="宋体"
    Font-Size="9pt" ForeColor="Black" Height="40px" Title="输
```

入定位名进行定位"

```

TitleBarColor="WhiteSmoke" TitleBarHeight="20px" TitleBar
SeparatorLine="False"
Transparency="35" Width="250px" Visible="False">
<uc3:PlaceLocation ID="PlaceLocation1" runat="server" />
</esri:FloatingPanel>

```

(3) 用 JavaScript 来控制 FloatingPanel 的显示，示例代码如下：

```

function setLocationShow()
{
    //alert('Show PlaceLocation');
    var locdialog = $find("FloatingPanell");
    if (locdialog)
    {
        locdialog.show();
        locdialog._moveTo(350,150);
    }
}

```

也可以像下面这样控制 FloatingPanel 的显示。

```

<input id="MyHTMLButton" type="button" value="HTML Button"
onclick="javascript: toggleFloatingPanelVisibility('FloatingPanell1')" />

```

(4) 在 FloatingPanel 属性列表框中设置其显示属性，如图 6.74 所示。

(5) FloatingPanel 运行界面如图 6.75 所示。



图 6.74 设置控件属性



图 6.75 FloatingPanel 运行界面

6.4.2 ContextMenu 控件

ContextMenu 控件允许用户与 Web ADF 应用程序在运行时进行交互。ContextMenu 控件可以包含一个或几个菜单项，每个菜单项可以执行客户端或服务器端代码。

- **Assembly** ESRI.ArcGIS.ADF.Web.UI.WebControls.dll。
- **Class** ESRI.ArcGIS.ADF.Web.UI.WebControls.ContextMenu。
- **JavaScript libraries** display_common.js, display_dotnetadf.js。

ContextMenu 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 ContextMenu 控件，然后将它拖到 Web 页面中，如图 6.76 所示。

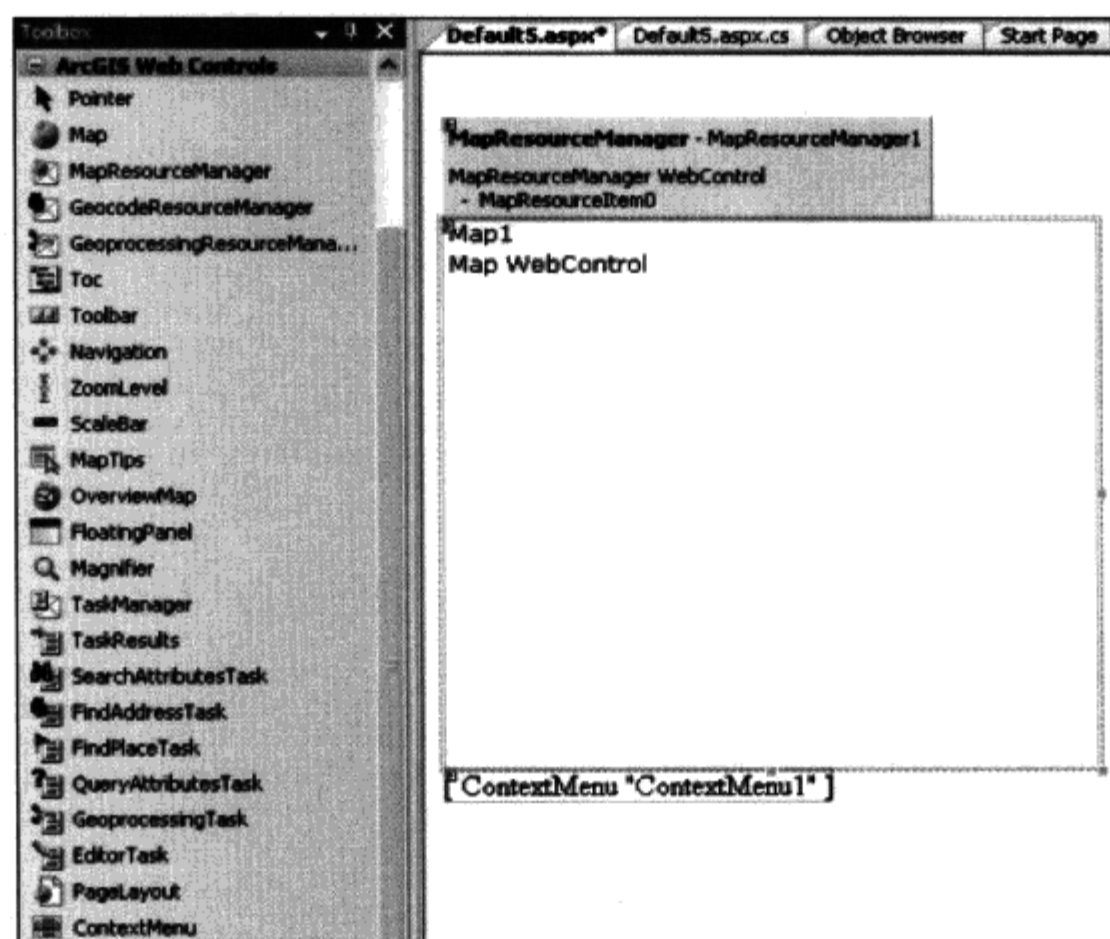


图 6.76 添加控件

(2) 用代码为控件添加菜单项，示例程序如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        ESRI.ArcGIS.ADF.Web.UI.WebControls.ContextMenuItem
contextMenuItem =
        new ESRI.ArcGIS.ADF.Web.UI.WebControls.ContextMenuItem();

        contextMenuItem.Text = "Zoom In";

        ContextMenu1.Items.Add(contextMenuItem);

        string showContextMenu = string.Format("esriShowContextMenu(event,
'{0}','{1}','{2}')";return false;", ContextMenu1.ClientID, Map1.UniqueID, "");

        Map1.Attributes.Add("oncontextmenu", showContextMenu);
    }
}
```

(3) 添加菜单单击响应事件，如图 6.77 所示。

(4) 完成菜单响应程序：

```
protected void ContextMenu1_ItemClicked(object sender, ESRI.ArcGIS.ADF.Web.UI.
WebControls.ContextMenuItemEventArgs args)
{
    switch (args.Item.Text)
    {
        case "Zoom In":
        {
            Map1.Zoom(2);
            ContextMenu1.CallbackResults.CopyFrom(Map1.CallbackResults);
            break;
        }
    }
}
```

菜单运行界面如图 6.78 所示。

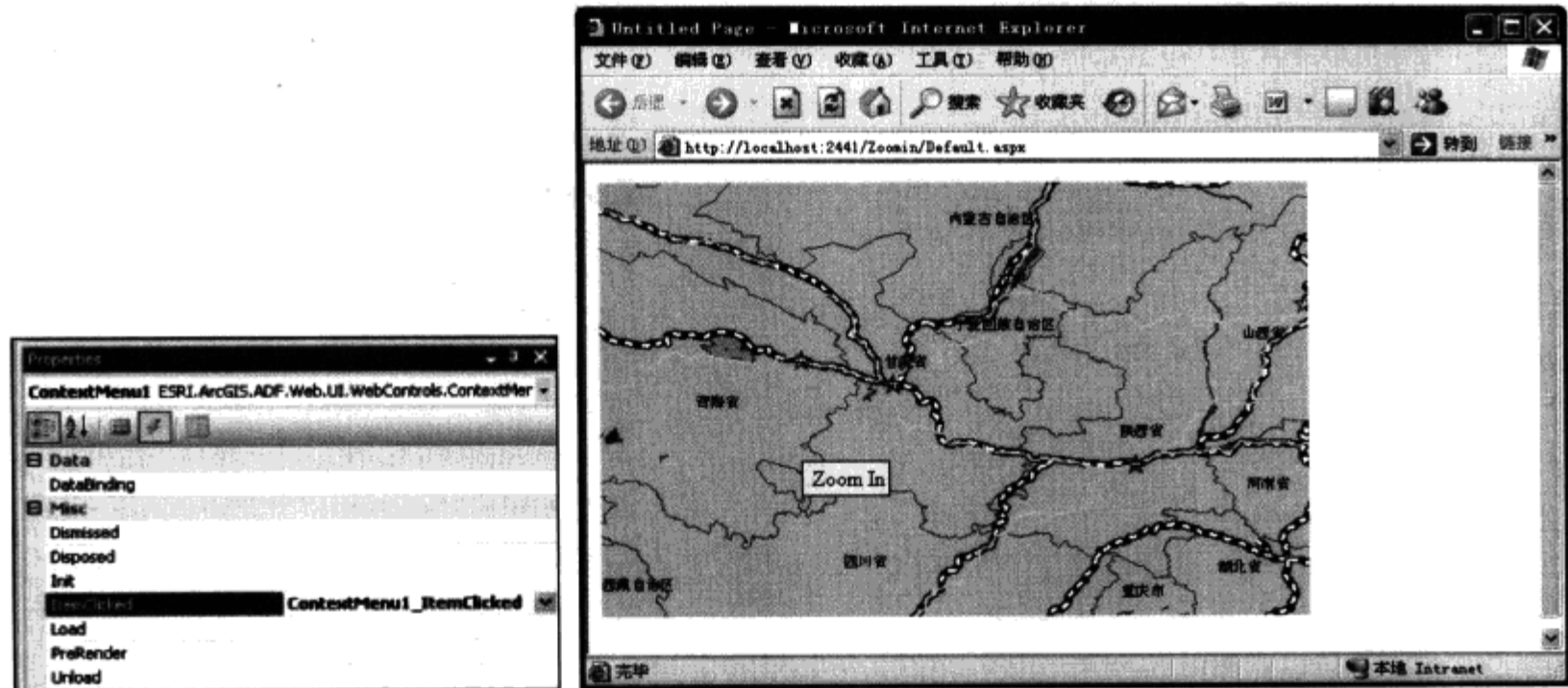


图 6.77 添加事件

图 6.78 菜单运行界面

6.4.3 DocExtender 控件

DocExtender 控件使 Web 控件停泊在 Map 控件上面。DocExtender 控件继承于 ASP.NET Ajax Control Toolkit 库中的 ExtenderControlBase。

- **Assembly** ESRI.ArcGIS.ADF.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.UI.WebControls. DocExtender。
- DocExtender 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签。在使用 DocExtender 的页面中至少需要 MapResourceManager、Map、Label。Label 主要用来泊在 Map 控件上，如图 6.79 所示。

(2) 设置控件属性。

设置 DockControlID 停泊控件的 ID、TargetControlID (目标控件的 ID), Alignment (停泊的位置), 如图 6.80 所示。

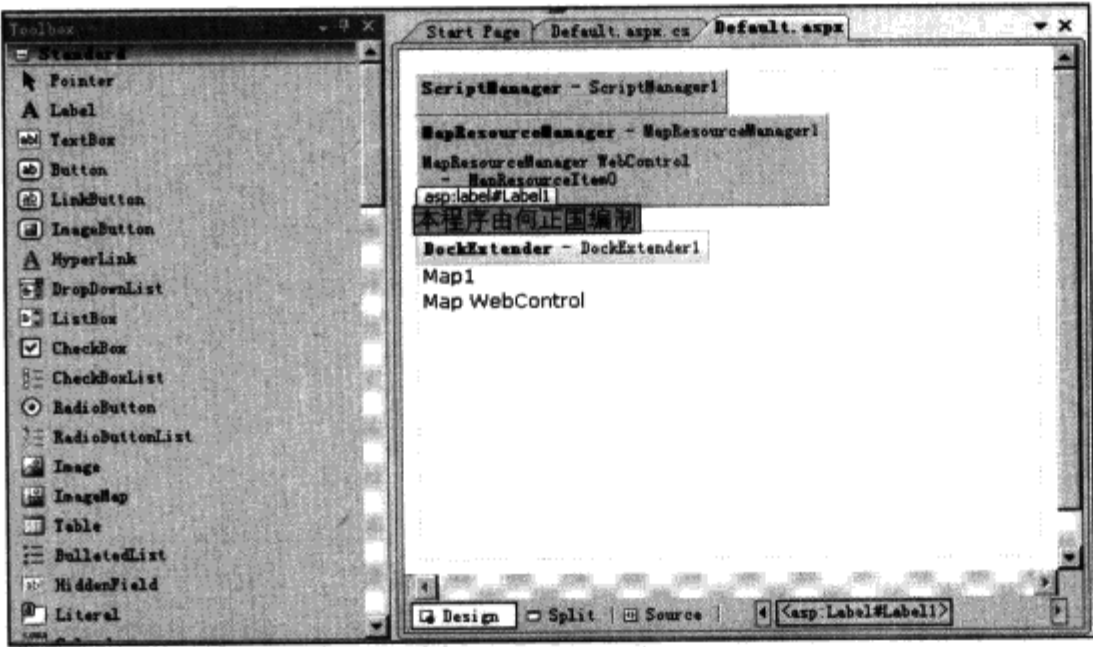


图 6.79 添加控件

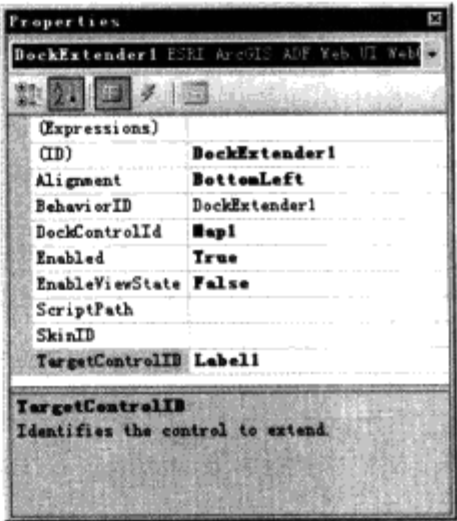


图 6.80 设置属性

运行后 Label 控件停泊在 Map 控件的左下方, 如图 6.81 所示。



图 6.81 Label 控件停泊在 Map 控件的左下方

6.4.4 HoverExpandExtender 控件

HoverExpandExtender 控件通过两种行为提高 Web 目标控件的可用性。第一, 当鼠标光标停留在控件上时, 使目标控件的透明性发生改变; 第二, 单击小图片可以折叠或展开控件。HoverExpandExtender 控件继承于 ASP.NET Ajax Control Toolkit 库中的 ExtenderControlBase。

- Assembly ESRI.ArcGIS.ADF.UI.WebControls.dll。

● **Class** ESRI.ArcGIS.ADF.UI.WebControls.HoverExpandExtender。

HoverExpandExtender 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签。在使用 HoverExpandExtender 的页面中至少需要 MapResourceManager、Map、OvervierMap。OvervierMap 主要用来做透明度的改变，如图 6.82 所示。

(2) 设置控件属性，如图 6.83 所示。

① 通过 TargetControlID 设置目标控件的 ID。

② 设置鼠标光标在目标控件的上方的行为。属性 MouseOutOpacity 和 MouseOverOpacity 是在 0.1~1.0 的浮点数。

③ 设置控件展开的行为。PinnedImageUrl、UnpinnedImageUrl、ThumbnailImageUrl 设置图片的路径。

④ 设置 Pinned 属性。

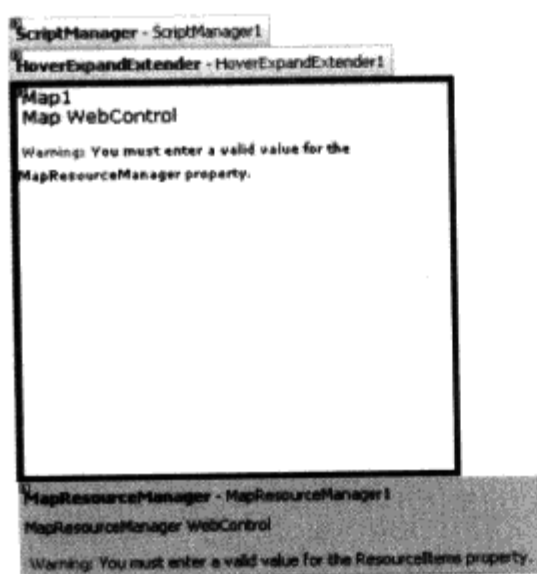


图 6.82 添加控件



图 6.83 设置控件属性

运行结果如图 6.84 和图 6.85 所示。

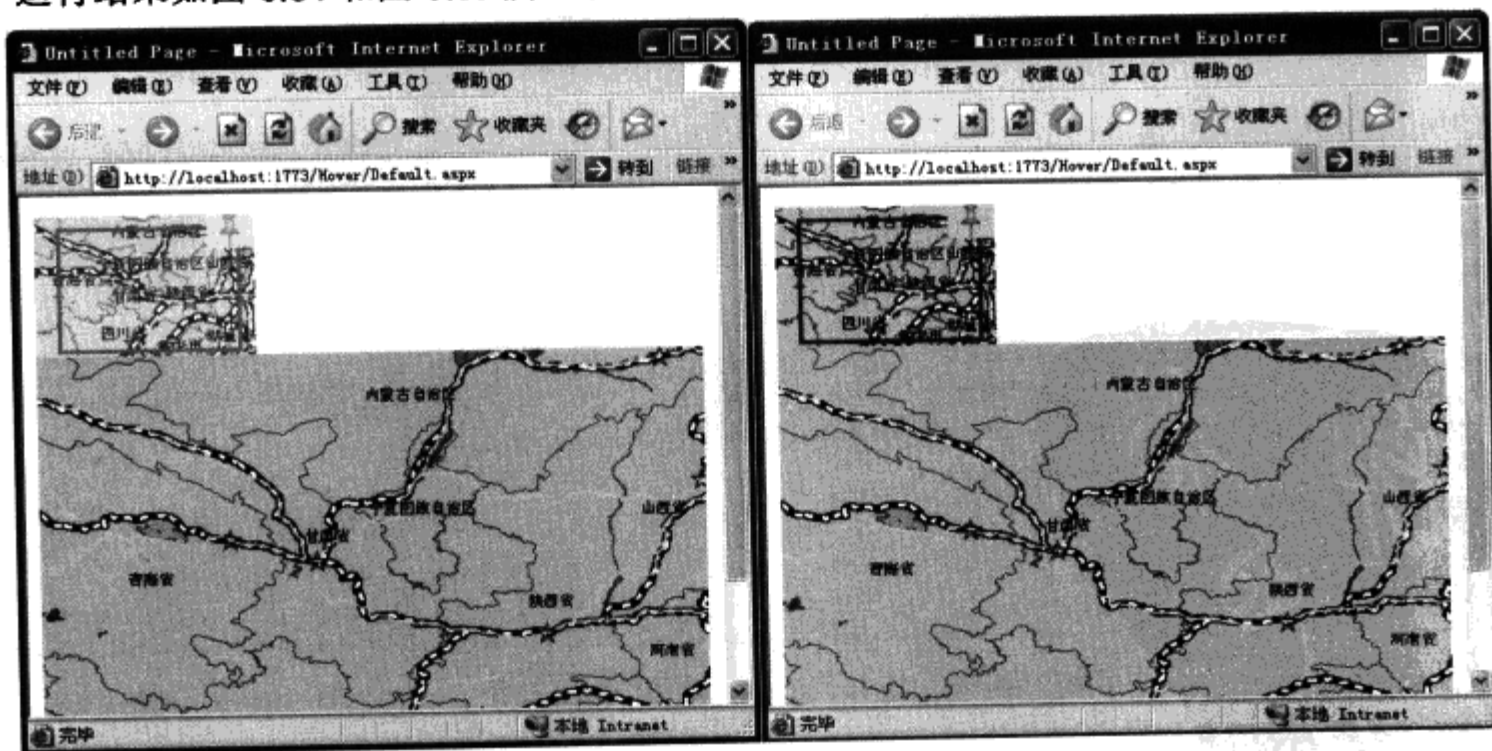


图 6.84 没有泊在与泊在缩略图对比图

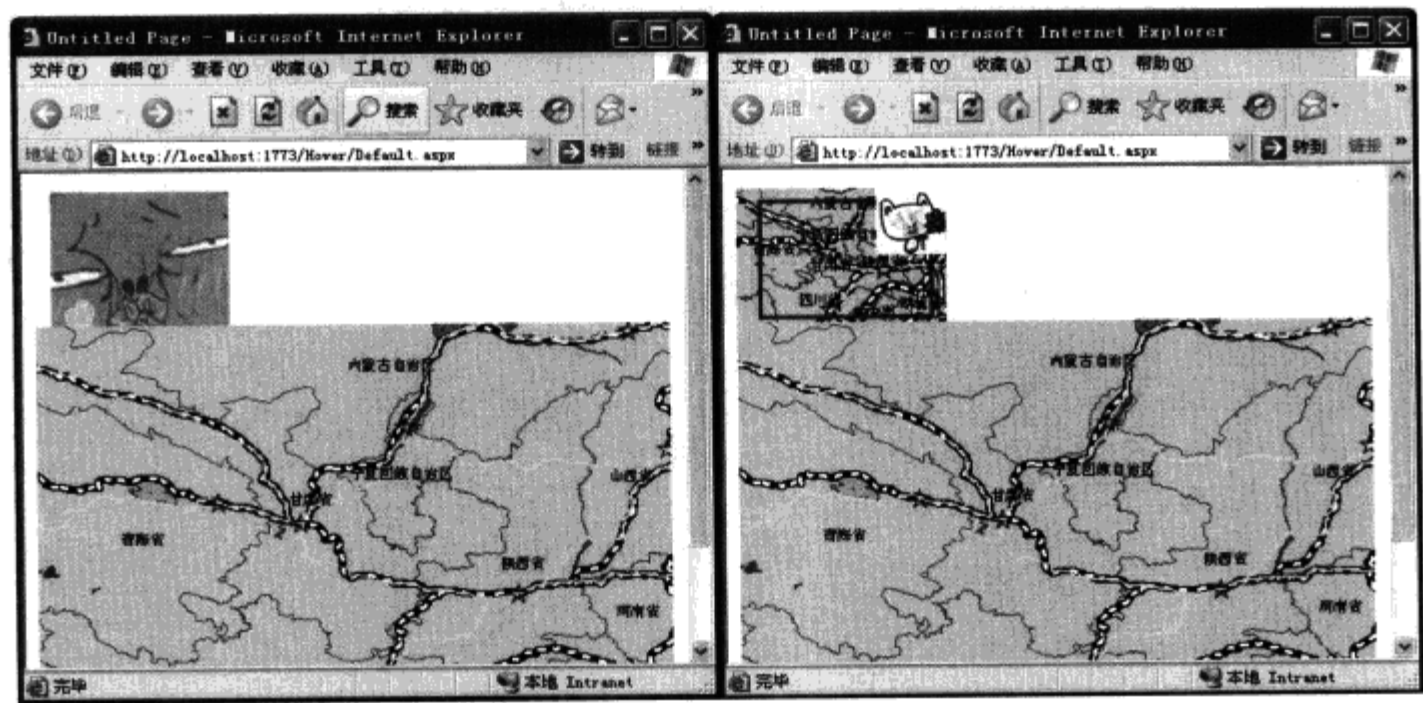


图 6.85 单击 Pin 图像对比

6.4.5 ColorPicker 控件

ColorPicker 控件是颜色选择器，用户想修改控件的颜色或者图形的颜色时可以使用。颜色选择器有很多，但 ColorPicker 可以与 Map 等控件相关联实现异步刷新。

- **Assembly** ESRI.ArcGIS.ADF.UI.WebControls.dll。
 - **Class** ESRI.ArcGIS.ADF.UI.WebControls.ColorPicker。
- ColorPicker 控件的使用方法如下。

(1) 将控件加入到页面。

打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 ColorPicker 控件，然后将它拖到 Web 页面中，如图 6.86 所示。

(2) 添加颜色选择事件，如图 6.87 所示。

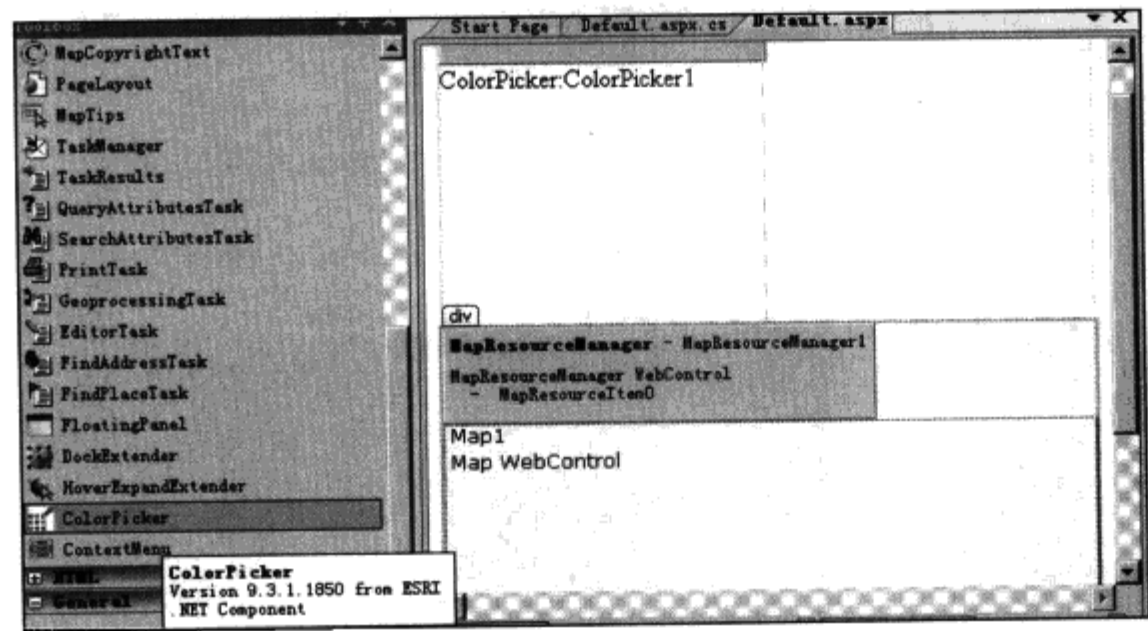


图 6.86 添加控件

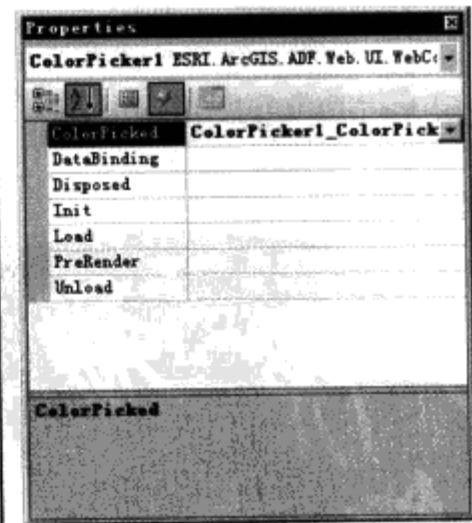


图 6.87 添加事件

(3) 获得选择的颜色的程序：

```
protected void ColorPicker1_ColorPicked(object sender, System.Drawing.Color .
```

```
chosenColor)
{
    Map1.BackColor = ColorPicker1.ChosenColor;
    // Map1.Zoom(2);
    ColorPicker1.CallbackResults.CopyFrom(Map1.CallbackResults);
}
```

运行结果如图 6.88 所示。

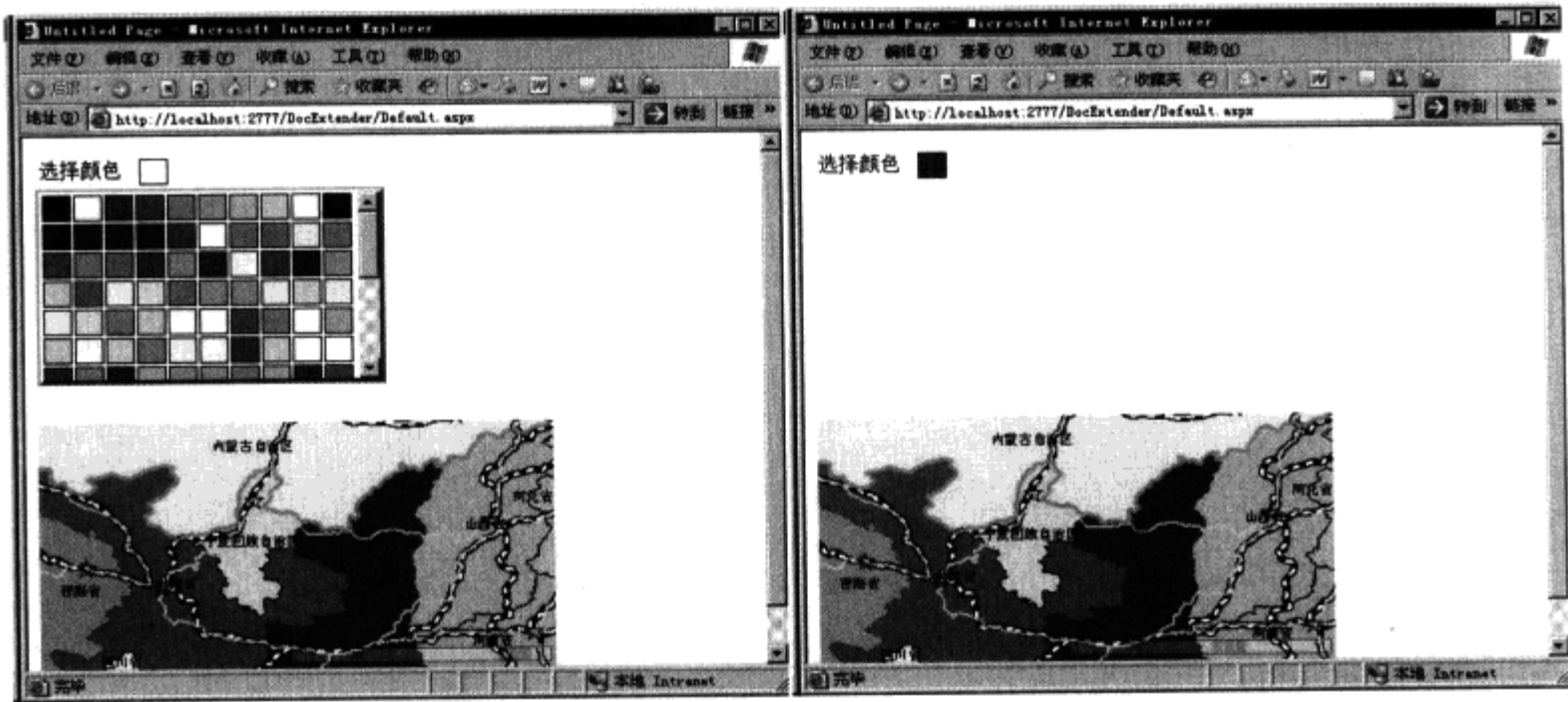


图 6.88 选择颜色后改变 Map 控件的对比

6.5 小结

本章主要介绍了 ArcGIS Server 提供的各种控件,读者通过对本章的阅读,应对 ArcGIS Server 提供的控件、控件的用途和控件的基本使用方法有了全面的了解,并建立起 ArcGIS Server 编程的基本概念。

第三篇

ArcGIS Server

开发提高

- ▶ 第 7 章 ArcGIS Server 开发概述
- ▶ 第 8 章 ArcGIS Server 基于模板开发
- ▶ 第 9 章 ArcGIS Server 专题图开发
- ▶ 第 10 章 ArcGIS Server 符号化
- ▶ 第 11 章 ArcGIS Server 数据在线编辑
- ▶ 第 12 章 ArcGIS Server Web 应用程序部署

掌握了 ArcGIS Server 的开发基本知识后，本部分主要介绍在日常 GIS 应用程序开发过程中经常遇到的 GIS 功能开发，如查询检索、缓冲区、装题图、符号化、空间数据编辑等，这些功能对读者的实际开发非常有帮助。

第 7 章

ArcGIS Server开发概述

前面的章节已经介绍过 ArcGIS Server 基于.NET 开发的一些小例子，但有些操作说明不够详细，本章将详细讲述 ArcGIS Server 基于.NET 的基本功能。

Web ADF 编程通用步骤如下。

- (1) 从 Web Controls 开始。
- (2) 访问 Resource Manager。
- (3) 找到待访问的 Resource。
- (4) 决定 Resource 支持哪个 Functionality。
- (5) 执行 Functionality。

开发流程图如图 7.1 所示。

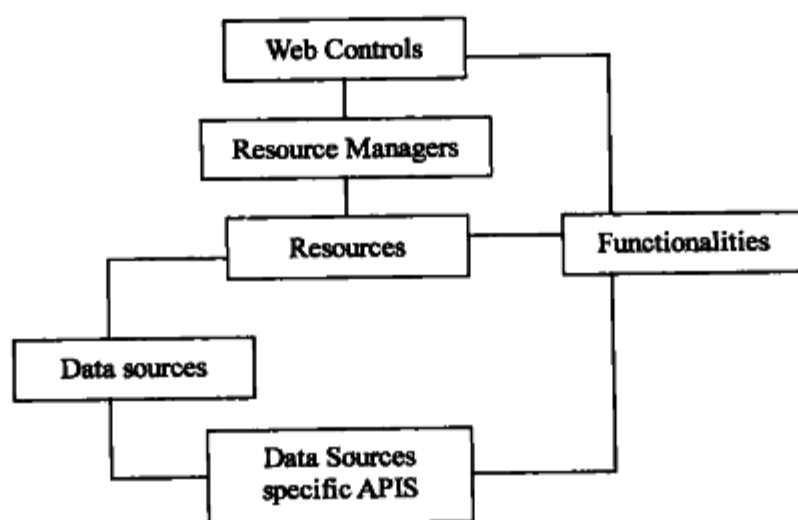


图 7.1 Web ADF 开发流程图

7.1 地图的基本操作

几乎所有的 GIS 应用程序都提供地图的基本操作如大、缩小、平移和全图显示，这些功能是 GIS 应用最基本的，也是必备的。事实上，这些功能的开发几乎不用开发人员编写任何代码，

只需要进行相关的配置就可以了。由此可见，于 ArcGIS Server 的开发是比较容易的。

7.1.1 地图显示

地图显示是 GIS 应用系统最基本的功能之一，基于 ArcGIS Server 的地图显示其实是显示 ArcGIS Server 发布的地图服务，既可以显示一个地图服务也可以显示多个地图服务，当然叠加多个地图服务，必须统一多个地图服务的坐标系，否则在 ArcGIS Server 中不能正确地显示。

地图显示需要的控件有地图资源管理控件 MapResourceManager、地图显示控件 Map，并需要一个地图显示的地图服务。下面列出地图显示的详细步骤。

(1) 启动 Microsoft Visual Studio 2008，系统运行后如图 7.2 所示。

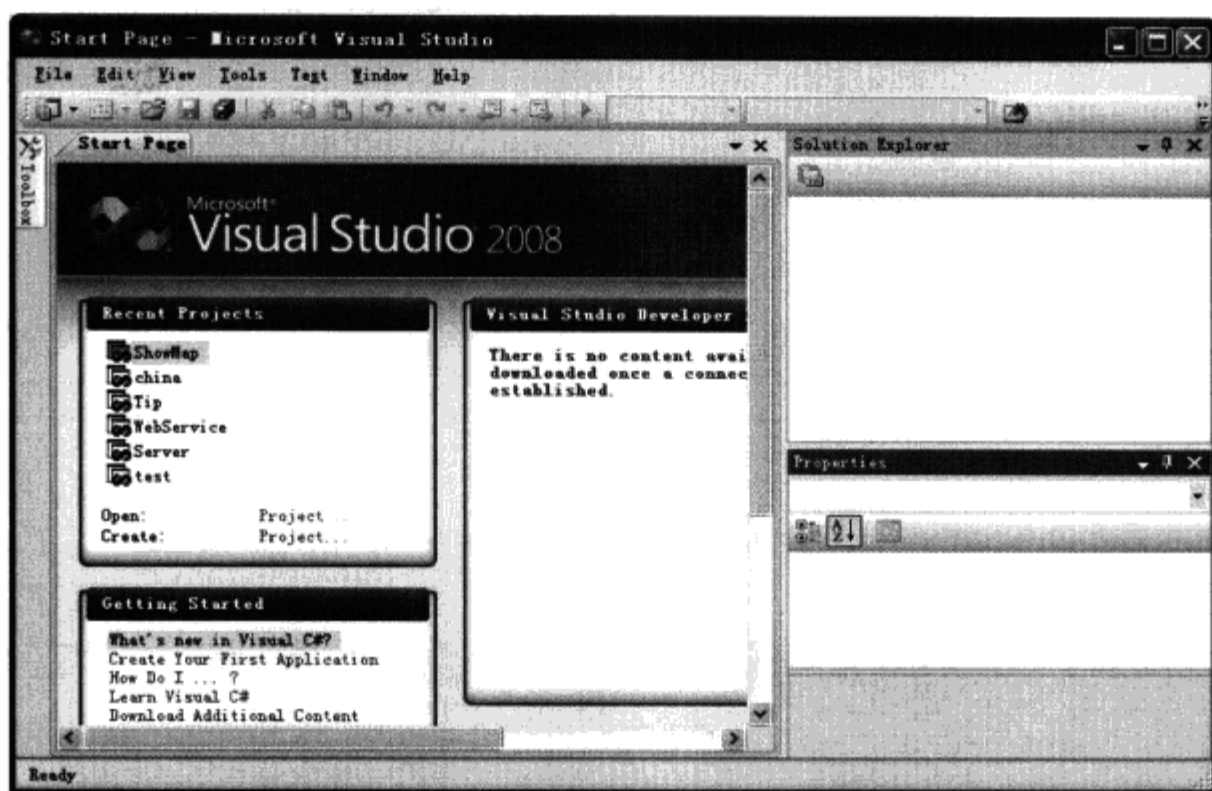


图 7.2 Microsoft Visual Studio 2008 运行界面

(2) 单击“File”→“New”→“Project”菜单，系统弹出“New Project”对话框，如图 7.3 所示。

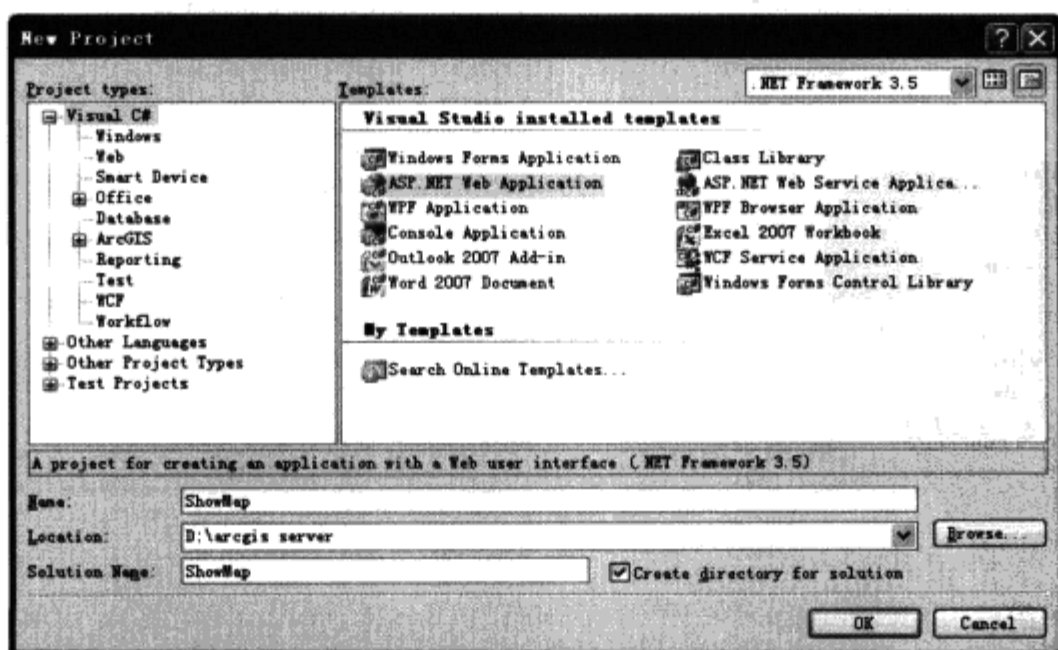


图 7.3 “New Project”对话框

(3) 在“Name”文本框中输入项目的名字，在“Location”文本框输入项目保存的路径，单击“OK”按钮，系统创建完毕项目，如图 7.4 所示。

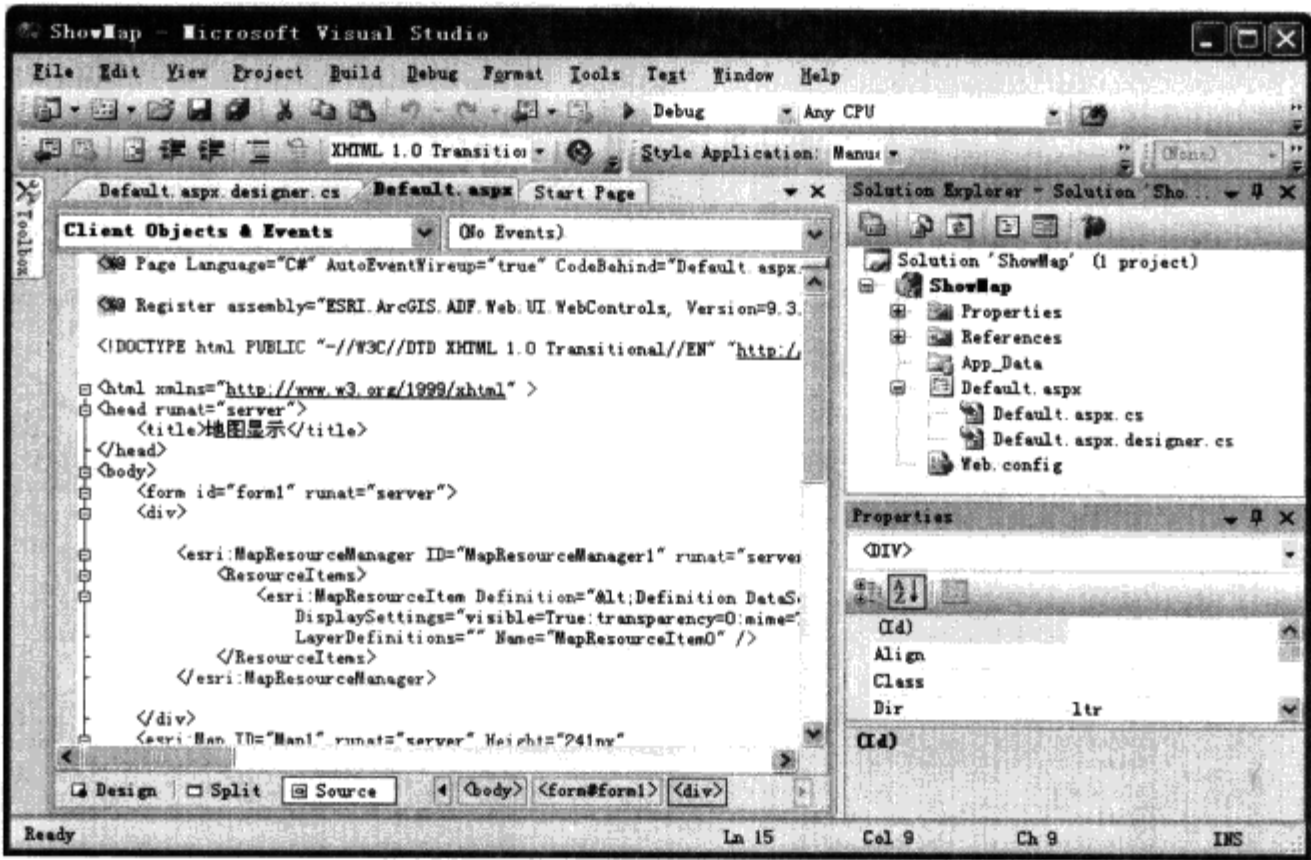


图 7.4 已创建项目

(4) 在“Toolbox”的“ArcGIS Web Controls”选项中把 MapResourceManager、Map 控件拖到 Default 页面上，如图 7.5 所示。

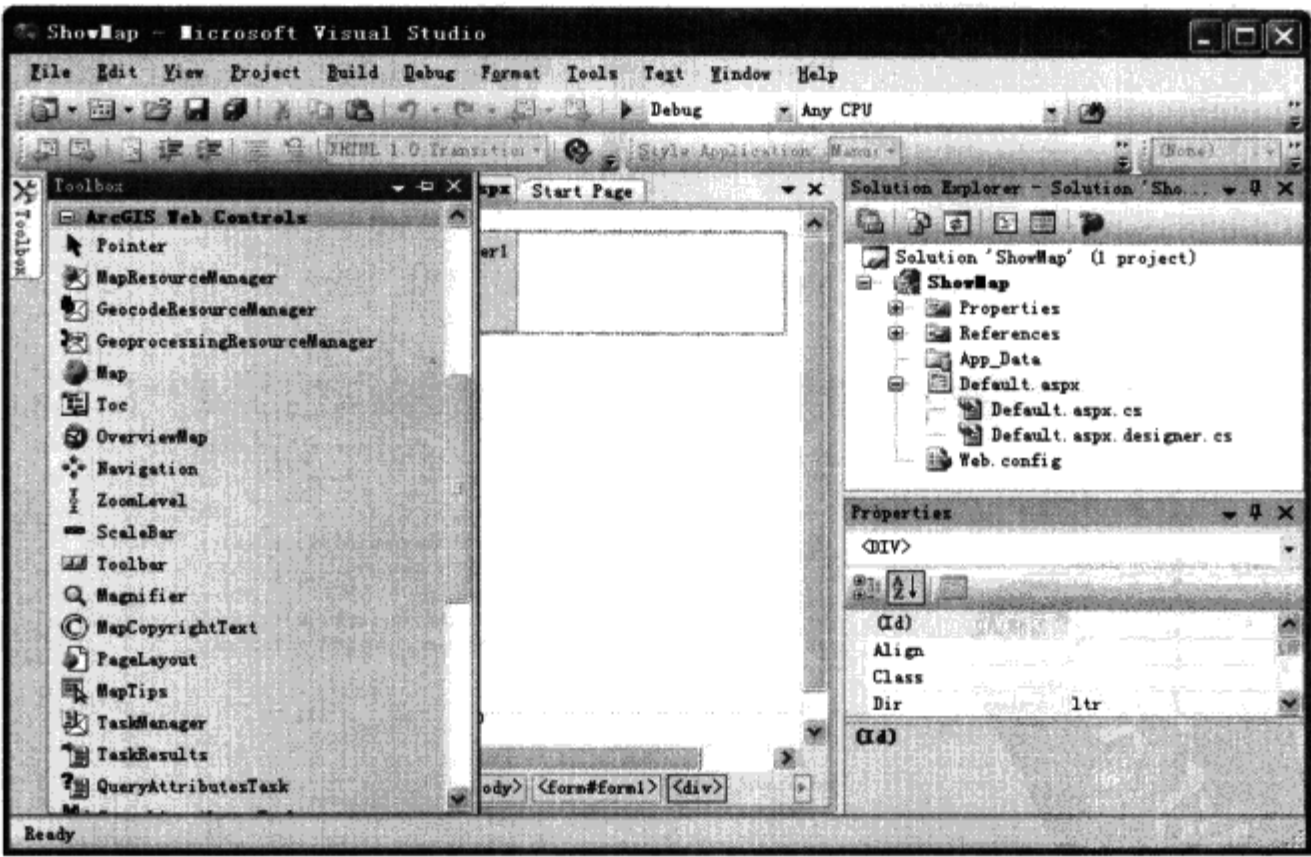


图 7.5 “Toolbox” 菜单

(5) 设置 MapResourceManager 控件的资源属性 ResourceItems，配置好地图资源，如图 7.6 所示。

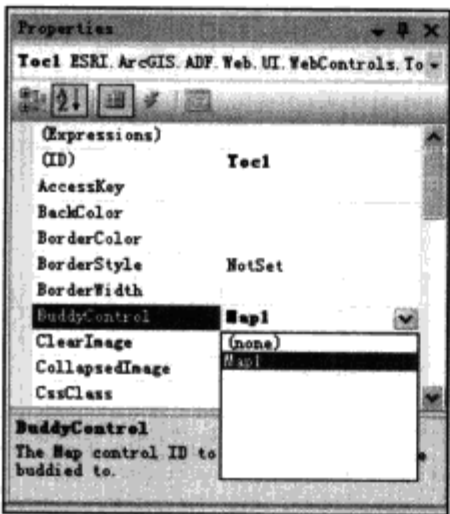


图 7.9 设置 Toc 控件的 BuddyControl 属性

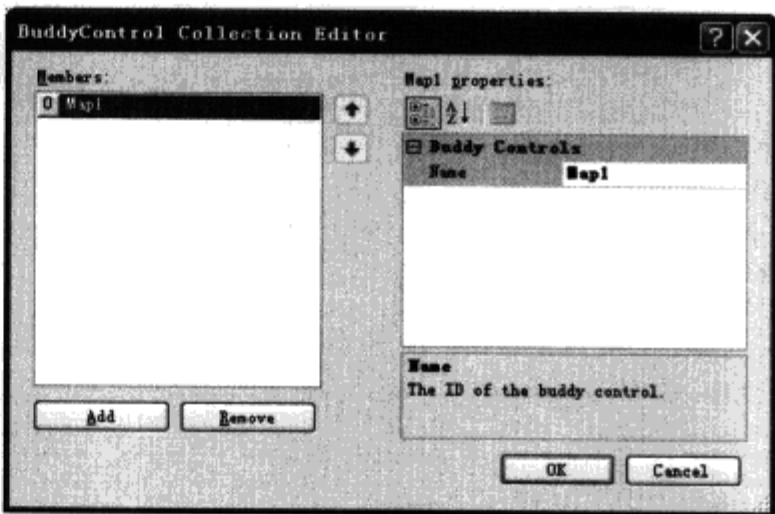


图 7.10 设置 Toolbar 控件的 BuddyControl 属性

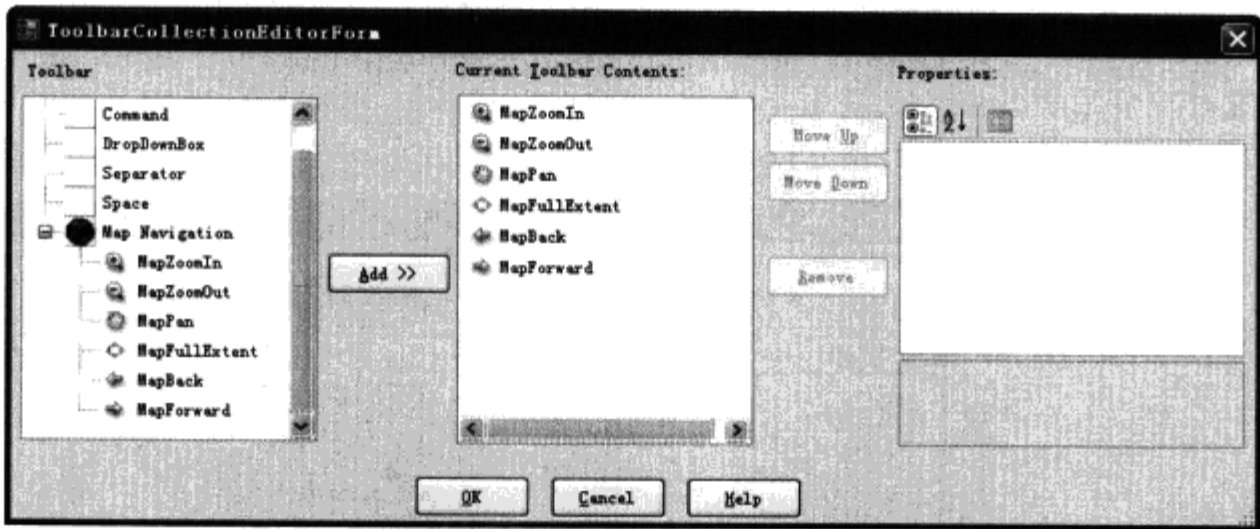


图 7.11 设置 Toolbar 的 ToolbarItems 属性

这样一个简单的 WebGIS 就做成了，开发人员不用编写任何代码，只需要进行一些配置就可以了。系统的运行界面如图 7.12 所示。其功能实现了地图放大、地图缩小、平移、全图显示、前一屏、后一屏以及图层的打开和关闭。

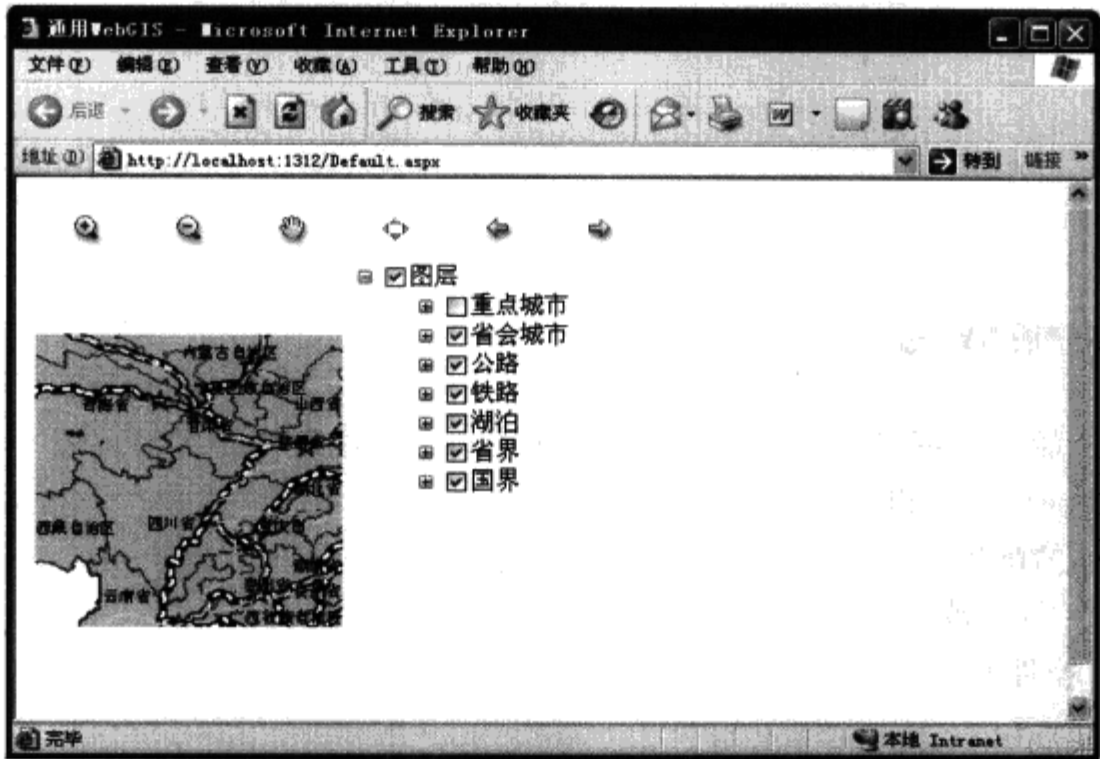


图 7.12 系统运行界面

7.2 查询定位

有了地图的显示、缩放以及平移,人们对地理空间的认识比较形象了,然而仅仅有这些功能,还是不够的,如地图的范围较大时,人们通常很难快速找到自己感兴趣的地点、位置等,此时就需要对地图进行快速定位。

7.2.1 根据坐标定位

根据坐标定位就是根据用户输入的某一个 X、Y 坐标,系统利用异步刷新实现坐标的快速定位。从实现快速定位的例子中可以进一步体会 ArcGIS Server 的 Ajax 的调用过程。有关 ArcGIS Server 控件的配置与 7.1 节中相似(也可在配书光盘中观看),本节不再赘述。坐标快速定位步骤如下:

(1) 布置好页面各控件以及各控件的位置,如图 7.13 所示。

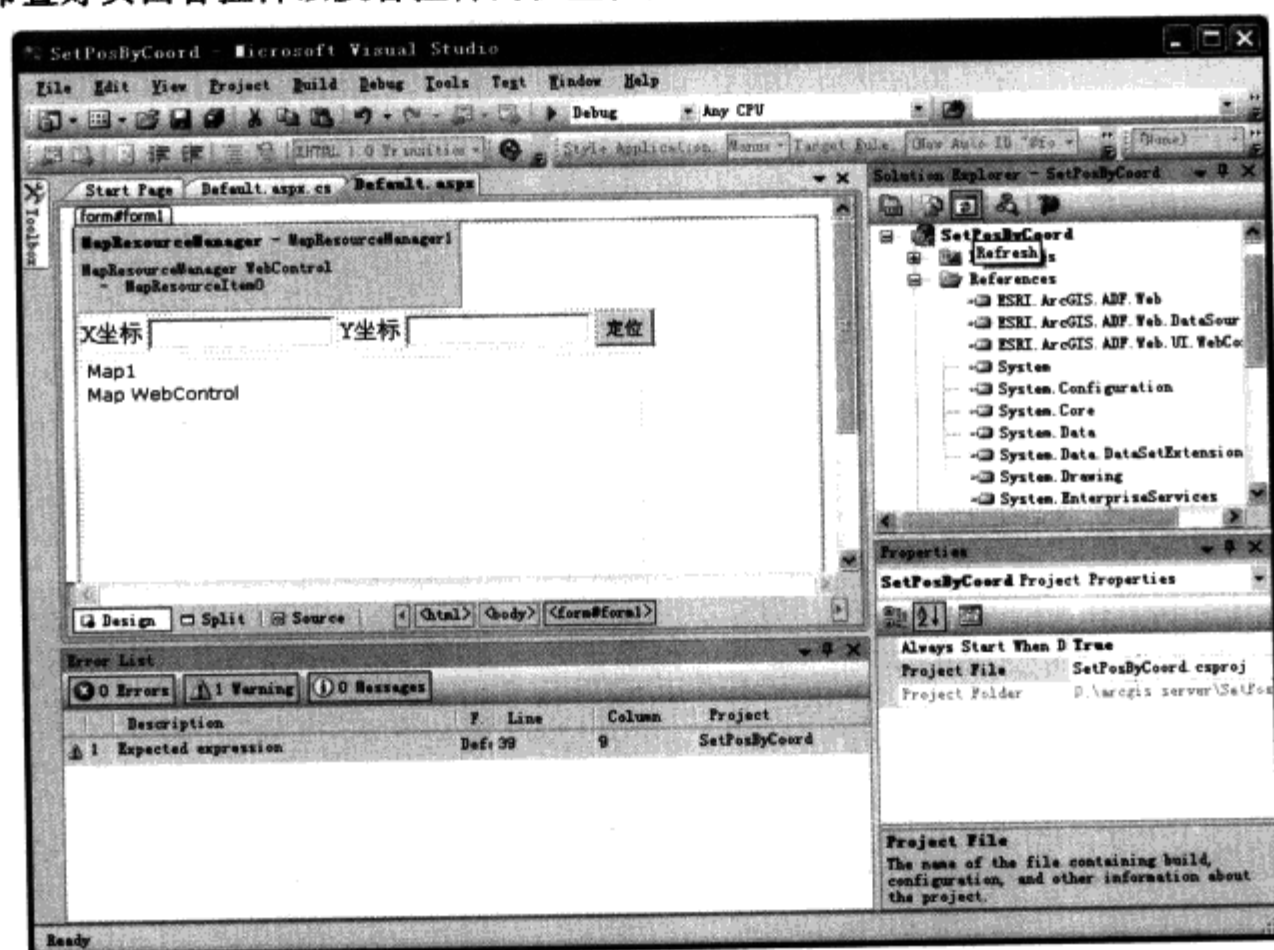


图 7.13 页面控件布局图

(2) 在 Default 页面的 Page_Load 事件中注册异步刷新信息:

```
m_ADFCallbackFunctionString = Page.ClientScript.GetCallbackEventReference(this,
"message",
"processCallbackResult", "context", "postBackError", true);
```

(3) 编写客户端向服务器传递消息,促使服务发生异步调用的 JavaScript 函数:

```
<script language="javascript" type="text/javascript">
function GetCoord()
{
    var x = document.getElementById("TxtX").value;
    if(x=='')
```

```

    {
        alert('请输入 X 坐标! ');
        return;
    }
    if(isNaN(x))
    {
        alert('请输入数字! ');
        document.getElementById("TxtX").focus();
        return;
    }
    var y = document.getElementById("TxtY").value;
    if(y=='')
    {
        alert('请输入 Y 坐标! ');
        return;
    }
    if(isNaN(y))
    {
        alert('请输入数字! ');
        document.getElementById("TxtY").focus();
        return;
    }
    var message = 'X='+x+"&Y="+y;      //传递消息
    var context = 'Map1';
    <%=m_ADFCallbackFunctionString%> //调用异步处理
}
</script>
<td><input id="OkBut" type="button" value="定位" onclick="GetCoord()"/></td>

```

(4) 当用户单击“定位”按钮时，客户端触发服务端异步处理程序 RaiseCallbackEvent:

```

public void RaiseCallbackEvent(string eventArgs)
{
    System.Collections.Specialized.NameValueCollection nameValueCollection =
    ESRI.ArcGIS.ADF.Web.UI.WebControls.CallbackUtility.ParseStringIntoNameValueCollection
    (eventArgs);
    double x = Convert.ToDouble(nameValueCollection["X"]);
    double y = Convert.ToDouble(nameValueCollection["Y"]);
    ESRI.ArcGIS.ADF.Web.Geometry.Point adfCenterPoint = new ESRI.ArcGIS.ADF.
    Web.Geometry.Point(x, y);
    Map1.CenterAt(adfCenterPoint);
    Map1.Zoom(20);
    m_ADFCallbackFunctionString = Map1.CallbackResults.ToString();
}

```

(5) 服务器经过 RaiseCallbackEvent 函数处理客户端传递的消息后，通过 GetCallbackResult 函数把处理结果返回客户端刷新。

(6) 客户端根据服务处理的处理结果，通过 JavaScript 函数 processCallbackResult (此函数 ArcGIS Server 已经封装) 来响应服务端的处理结果，如图 7.14 所示。

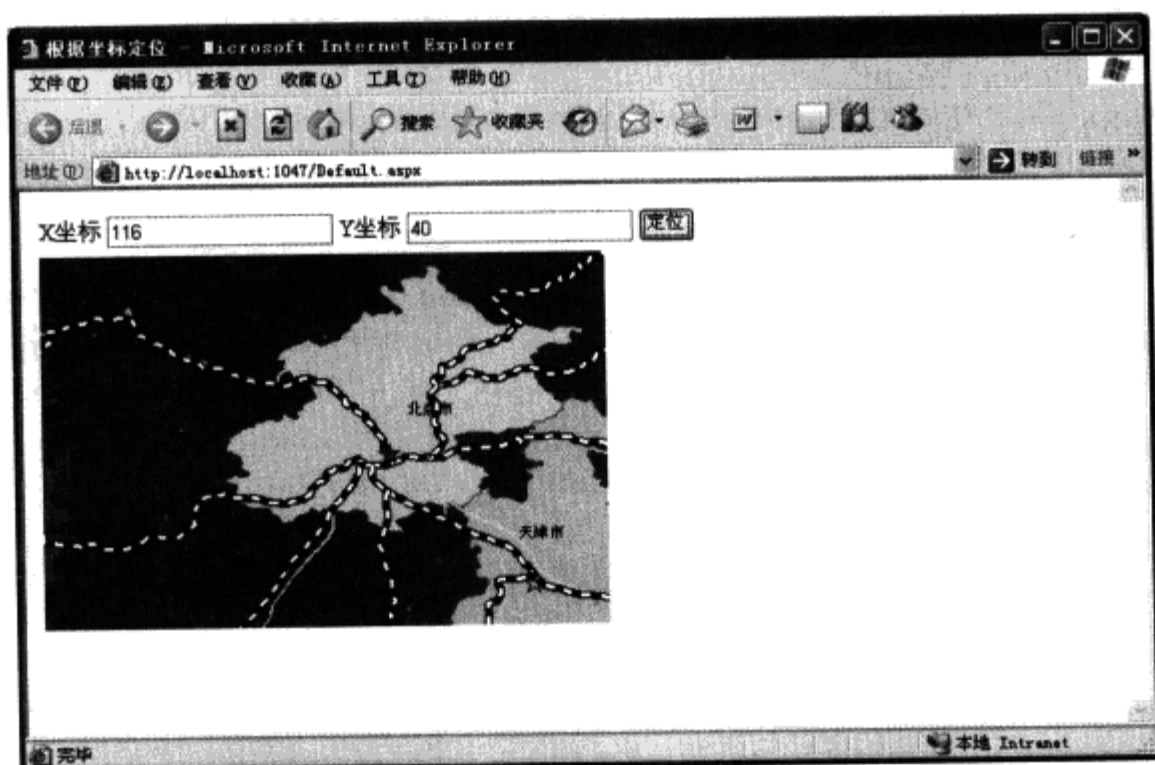


图 7.14 运行结果图

通过上述根据坐标值来实现地图快速定位的简单例子（详细程序见本书附带的光盘），可以让读者亲身体验 ArcGIS Server 异步刷新的处理过程，弄懂了这个过程，对于进行 ArcGIS Server 的编程会起到事半功倍的效果。

7.2.2 根据属性值定位

人们通常能够记住的坐标实在是有限，然而却很容易记住空间对象的某一属性值。本小节将介绍如何根据空间图层的某一属性值来定位空间对象。

根据属性值来定位空间对象与根据坐标来定位空间对象的异步刷新过程一样，都是通过单击“定位”按钮，客户端应用程序触发服务端异步刷新，服务器接收到客户端传递的参数，启动 RaiseCallbackEvent 函数来处理异步刷新，并把处理结果通过 GetCallbackResult 函数返回客户端，客户端通过 ArcGIS Server 封装的 processCallbackResult 来实现客户端的刷新。不同之处在于服务器处理客户端传递参数的函数不同，根据属性定位的处理程序如下。

```
public void RaiseCallbackEvent(string eventargs)
{
    System.Collections.IEnumerable func_enum = null;
    //获取当前 map1 控件中所有的 functionality
    func_enum = Map1.GetFunctionalities();

    System.Data.DataTable datatable;
    //对所有的 functionality 进行遍历
    foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gisfunctionality
        in func_enum)
    {
        ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource = null;
        //得到该 functionality 的 resource
        gisresource = gisfunctionality.Resource;
        //判断该 resource 是否支持 IQueryFunctionality
    }
}
```

```

        bool supported = false;
        supported =
            gisresource.SupportsFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQuery
yFunctionality));
        if (supported)
        {
            ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality qfunc;
            qfunc = (ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality)
gisresource.CreateFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunct
ionality), null);

            string[] lids;
            string[] lnames;
            //获得图层的 layerId 和 layerName, GetQueryableLayers 的重载方法可以
            //指定图层类型
            qfunc.GetQueryableLayers(null, out lids, out lnames);

            int selindex = -1;
            for (int i = 0; i < lids.Length; i++)
            {
                if (lnames[i] == "省会城市")
                {
                    //找到省界图层的序号
                    selindex = i;
                    break;
                }
            }

            if (selindex > -1)
            {
                //设置过滤器的过滤条件
                ESRI.ArcGIS.ADF.Web.SpatialFilter spatialfilter = new
ESRI.ArcGIS.ADF.Web.SpatialFilter();
                spatialfilter.ReturnADFGeometries = true;
                spatialfilter.MaxRecords = 100;
                spatialfilter.WhereClause = "name like '%" + eventargs + "%'";

                //对指定的图层进行查询, 查询的结果保存为
                datatable = qfunc.Query(null, lids[selindex], spatialfilter);
                if (datatable != null)
                {
                    if (datatable.Rows.Count > 0)
                    {
                        for (int j = 0; j < datatable.Columns.Count; j++)
                        {
                            if (datatable.Columns[j].DataType == typeof(ESRI.
ArcGIS.ADF.Web.Geometry.Geometry))
                            {
                                ESRI.ArcGIS.ADF.Web.Geometry.Point pnt =
(ESRI.ArcGIS.ADF.Web.Geometry.Point)datatable.Rows[0][j];
                                Map1.CenterAt(pnt);

```

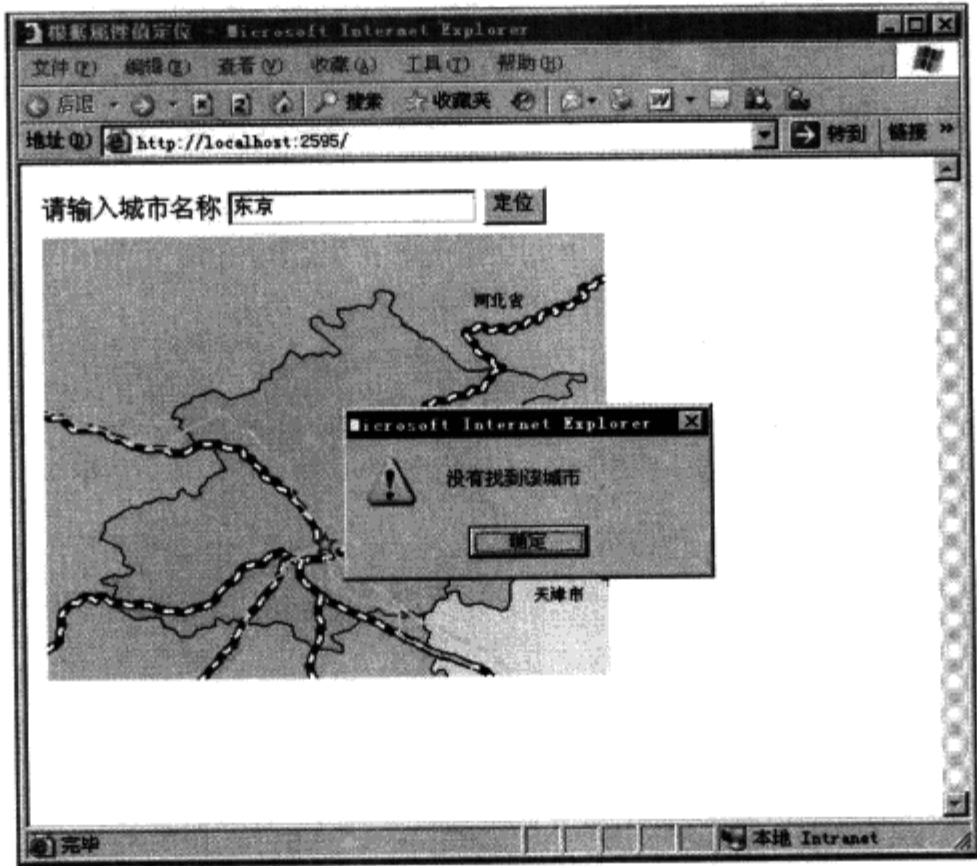



图 7.16 系统提示没有找到

7.3 缓冲区分析

ArcGIS Server Web ADF 的开发没有提供缓冲区的算法，不过对于属性 ArcObjects (AO) 开发人员都知道，在 ArcObjects 中实现缓冲区非常简单，只需要调用 ITopologicalOperator 接口即可。

这里有必要对 ArcGIS Server 的几何对象进行解释。在 ArcObjects 中，几何对象都是 COM 对象，而在 ADF Web Controls 中的几何对象都是 ValueObject，这些几何对象使用不同的开发方式时必须要进行转换。如在 ArcGIS Server 中调用 ArcObjects 接口，就必须把 ADF Web Controls 的 ValueObject 转换为 COM 的 ComObject 使用，ArcGIS Server 提供 Converter 来处理这些不同类型几何对象的转换。

谈到 ArcGIS Server 中的 AO 编程，有必要对 ArcGIS Server 的两种通用连接方式进行解释：ArcGIS Server Internet 和 ArcGIS Server Local。ArcGIS Server Internet 方式连接 GIS Server 时，就相当于连接标准的 Web Service，而对于 Web Service 的用户而言，只有其暴露的方法才能使用，没有暴露的方法则不能使用。因而通过 ArcGIS Server Internet 方式连接得到某个服务的时候所获得的功能是有限的。ArcGIS Server Local 通过局域网连接 GIS Server，这时可以使用后台功能强大的 AO 组件，也就是说能远程调用 ArcGIS Server 提供的 AO 组件，可以实现强大 GIS 功能的开发。

在 ArcGIS Server 进行远程调用 AO 时候，必须创建一个服务器上下文 (ServerContext)，只有拥有服务器上下文，服务器对象和所有相关的其他对象才是活动的，可以调用，一旦释放了服务器上下文，就不能再使用服务器对象。服务器上下文可以看做一个进程 (ArcSOC.EXE)，作为访问 Server Object 和 ArcObjects 的入口。在 ArcGIS Server 中使用 Server Context 的一般步骤如下。

- (1) 建立与服务器的连接。
- (2) 获得服务器对象。

(3) 使用服务器对象。

(4) 释放服务器对象。

ArcGIS Server 中远程调用 ArcObjects 示意图如图 7.17 所示。

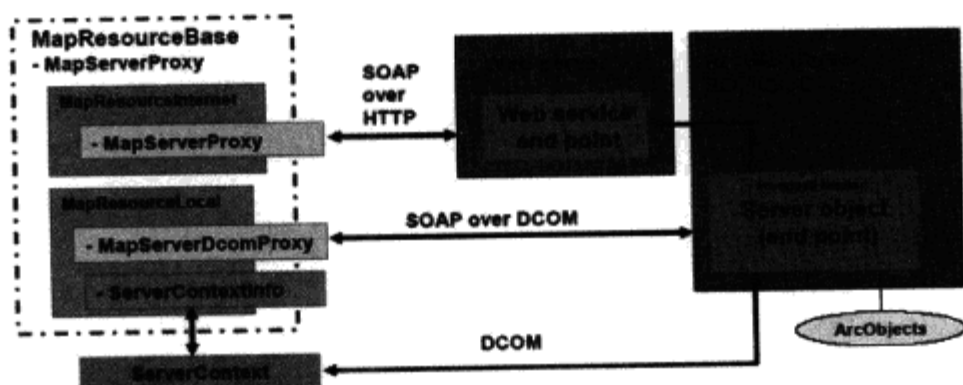


图 7.17 ArcGIS Server 远程调用 ArcObjects

进行缓冲区分析后, 就该考虑如何表现缓冲区了。在 ADF 开发过程中, 我们经常需要实现一些功能, 如选择、地图标注等, 一般都采用服务器创建或使用 Graphics Layers。ADF 通过 MapResourceManage 添加数据源时除了可以添加 ArcGIS Server、ArcIMS、OGC 等地图数据源外, 还支持 Graphics 数据源。ADF 中的 Graphics 数据源可以支持两种图层, 分别为 ElementGraphicsLayer 和 FeatureGraphicsLayer, 两者都是 System.Data.DataTable 类型。ElementGraphicsLayer 图层一般用来显示基本的图形元素, 一个 ElementGraphicsLayer 可以同时存放不同的几何类型(点、线、面等)元素。一个典型的应用为图层显示被选择的要素 (Feature)。FeatureGraphicsLayer 图层用来模拟要素图层并且存储要素的属性信息。因而客户端可以根据要素的不同属性来渲染不同的要素。一个 FeatureGraphicsLayer 图层只能存放一种几何类型的元素, 而且 FeatureGraphicsLayer 图层还支持查询。

Graphics 图层中的元素都存储在内存中, MapResourceManager 只能添加空的 Graphics 数据源, 图层 (GraphicsLayer) 以及图元素 (Graphic Element) 的操作都必须通过编程来实现, 实现步骤如下。

(1) 创建两个资源连接, 一个是连接地图服务, 另外一个为内存图像, 用来画缓冲图形, 如图 7.18 所示。

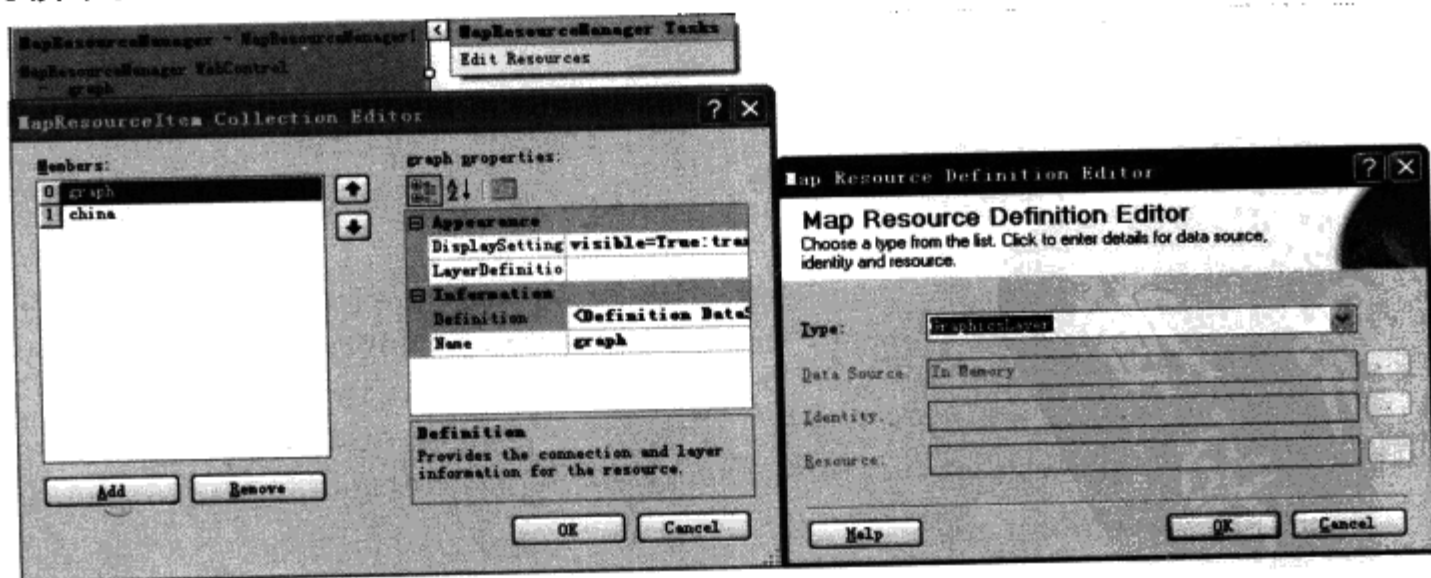


图 7.18 配置资源连接

(2) 在 ArcGIS Server 中实现对 ArcObjects 的访问, 利用 ArcObject 来实现缓冲算法, 核心

程序如下：

```
//找到该点
ESRI.ArcGIS.ADF.Web.Geometry.Geometry pnt = (ESRI.ArcGIS.ADF.Web.Geometry.Geometry)
datatable.Rows[0][j];

//对该点进行缓冲分析
//1. 连接服务器
ESRI.ArcGIS.ADF.Connection.AGS.AGSServerConnection connection = new ESRI.ArcGIS.
ADF.Connection.AGS.AGSServerConnection();
connection.Host = "localhost";
connection.Connect();

//2. 获得服务器对象
ESRI.ArcGIS.Server.IServerObjectManager pSOM = connection.ServerObjectManager;
ESRI.ArcGIS.Server.IServerContext pSC = pSOM.CreateServerContext("china", "Map
Server");//服务名和类型
ESRI.ArcGIS.Carto.IMapServer mapserver = pSC.ServerObject as ESRI.ArcGIS.Carto.
IMapServer;

//3. 使用服务器对象
ESRI.ArcGIS.Carto.IMapServerObjects pMSO = (ESRI.ArcGIS.Carto.IMapServerObjects)
mapserver;
ESRI.ArcGIS.Geometry.IGeometry compnt = (ESRI.ArcGIS.Geometry.IGeometry)ESRI.ArcGIS.
ADF.Web.DataSources.ArcGISServer.Converter.ToIGeometry(pnt,
pSC);//ValueObjectToComObject(pnt, pSC);
ESRI.ArcGIS.Geometry.SpatialReferenceEnvironment sre = new
ESRI.ArcGIS.Geometry.SpatialReferenceEnvironment();

ESRI.ArcGIS.Geometry.ISpatialReference pSR = sre.CreateGeographicCoordinateSystem
(4326);
compnt.SpatialReference = pSR;
ESRI.ArcGIS.Geometry.ITopologicalOperator pTOPO = (ESRI.ArcGIS.Geometry.ITopologicalO
perator)compnt;

pTOPO.Simplify();
double bufdis = Map1.Extent.Width / 2;
ESRI.ArcGIS.Geometry.IPolygon bufPoly = (ESRI.ArcGIS.Geometry.IPolygon)pTOPO.
Buffer(1);
ESRI.ArcGIS.ADF.ArcGISServer.PolygonN valuePoly = (ESRI.ArcGIS.ADF.ArcGISServer.
PolygonN)ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.Converter.ComObjectToValueOb
ject(bufPoly, pSC, typeof(ESRI.ArcGIS.ADF.ArcGISServer.PolygonN));
ESRI.ArcGIS.ADF.Web.Geometry.Polygon adfpoly = (ESRI.ArcGIS.ADF.Web.Geometry.
Polygon)ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.Converter.ToAdfPolygon(valueP
oly);

ESRI.ArcGIS.ADF.Web.Geometry.Geometry geom = adfpoly;
ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement ge = new ESRI.ArcGIS.ADF.Web.
```

```

Display.Graphics.GraphicElement(geom, System.Drawing.Color.BlueViolet);
Map1.Zoom(adfpoly);

ge.Symbol.Transparency = 70;

System.Collections.IEnumerable gfc = Map1.GetFunctionalities();
ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource gmap = null;

foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gfunc in gfc)
{
    if (gfunc.Resource.Name == "graph")
    {
        gmap = (ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource)gfunc.Resource;
    }
}

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

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

glayer.Clear();//清除以后数据
glayer.Add(ge);

//4. 释放服务器对象
pSC.ReleaseContext();

```

(3) 实现不同数据源几何对象之间的转换，核心程序如下：

```

ESRI.ArcGIS.Geometry.IGeometry compnt = (ESRI.ArcGIS.Geometry.IGeometry)ESRI.
ArcGIS.ADF.Web.DataSources.ArcGISServer.Converter.ToIGeometry(pnt, pSC);//实现 ADF 到
COM 对象的转换

ESRI.ArcGIS.ADF.ArcGISServer.PolygonN valuePoly = (ESRI.ArcGIS.ADF.ArcGISServer.
PolygonN)ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.Converter.ComObjectToValueOb
ject(bufPoly, pSC, typeof(ESRI.ArcGIS.ADF.ArcGISServer.PolygonN)); //实现 COM 到 Server
对象的转换

ESRI.ArcGIS.ADF.Web.Geometry.Polygon adfpoly = (ESRI.ArcGIS.ADF.Web.Geometry.

```



```
Polygon)ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.Converter.ToAdfPolygon(valuePoly); //实现 Server 到 ADF 对象的转换
```

(4) 实现缓冲图形的添加与刷新, 核心程序如下:

```
ESRI.ArcGIS.ADF.Web.Geometry.Geometry geom = adfpoly;
ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement ge = new ESRI.ArcGIS.ADF.Web.
Display.Graphics.GraphicElement(geom, System.Drawing.Color.BlueViolet);
Map1.Zoom(adfpoly);

ge.Symbol.Transparency = 70;

System.Collections.IEnumerable gfc = Map1.GetFunctionalities();
ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource gmap = null;

foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gfunc in gfc)
{
    if (gfunc.Resource.Name == "graph")
    {
        gmap = (ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource)gfunc.Resource;
    }
}

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

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

glayer.Clear();//清除以后数据
glayer.Add(ge);
//刷新

if (Map1.ImageBlendingMode == ESRI.ArcGIS.ADF.Web.UI.WebControls.ImageBlendingMode.
WebTier)
    Map1.Refresh();
else if (Map1.ImageBlendingMode == ESRI.ArcGIS.ADF.Web.UI.WebControls.ImageBlendingMode.
Browser)
```

```
Map1.RefreshResource(gmap.Name);
```

应用程序运行界面如图 7.19 所示。

通过对一个城市缓冲区的分析，可以对一个城市的经济辐射范围进行分析，从而对该城市的影响力进行判定。在这里需要指出的是，在进行缓冲区分析的时候，如果服务没有制定空间坐标参考，在求空间对象时必须指定几何坐标参考，否则在求缓冲区的时候会出错。详细的源代码见本书附带光盘。

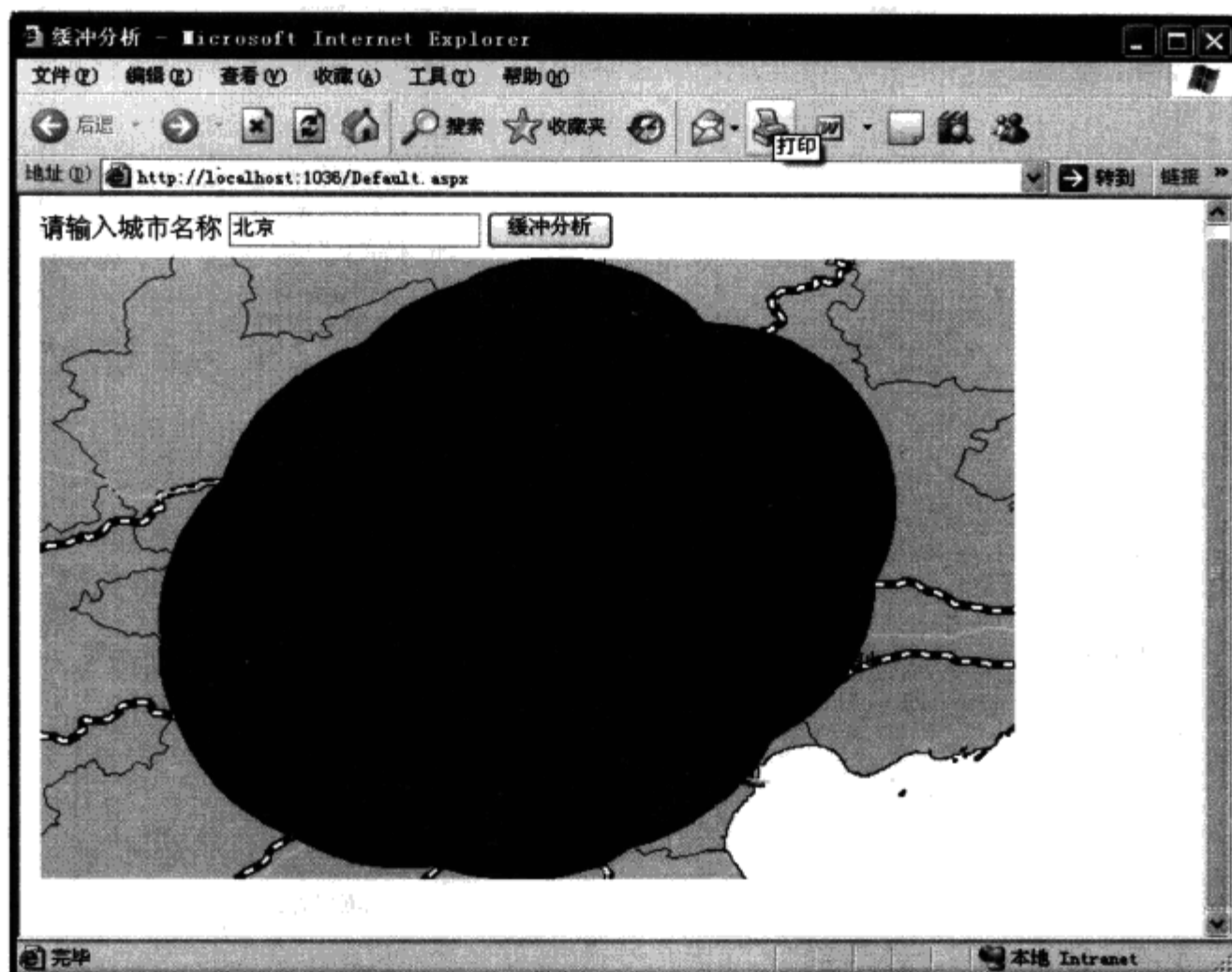


图 7.19 北京的缓冲区

7.4 自定义 Tool

在开发 GIS 应用程序的时候，经常需要处理人机交互。例如，在地图上单击一点，找到该点对应图层的几何对象。这是一种非常简单也非常通用的 GIS 功能，用户在地图上单击一点，可以确定该点的屏幕坐标，经过转换屏幕坐标为图面坐标，就可以在相应图层找到与该点相交的几何对象。

在 ArcGIS Server ADF 开发中，这种涉及空间交互的过程通常采用自定义 Tool 来实现，实现的步骤如下。

- (1) 在页面上放置好 MapResourceManager、Map、Toc、Toolbar 等相关控件，如图 7.20 所示。
- (2) 有关各控件的配置与前面的相同，添加自定义 Tool，如图 7.21 所示。

在添加自定义 Tool 时，需要注意几个重要的属性项：ServerActionAssembly 为服务器实现类存放的地方；ServerActionClass 为服务器实现类的名称；EnablePostBack 为 False 时表示只刷新地图，为 True 时则刷新整个页面。

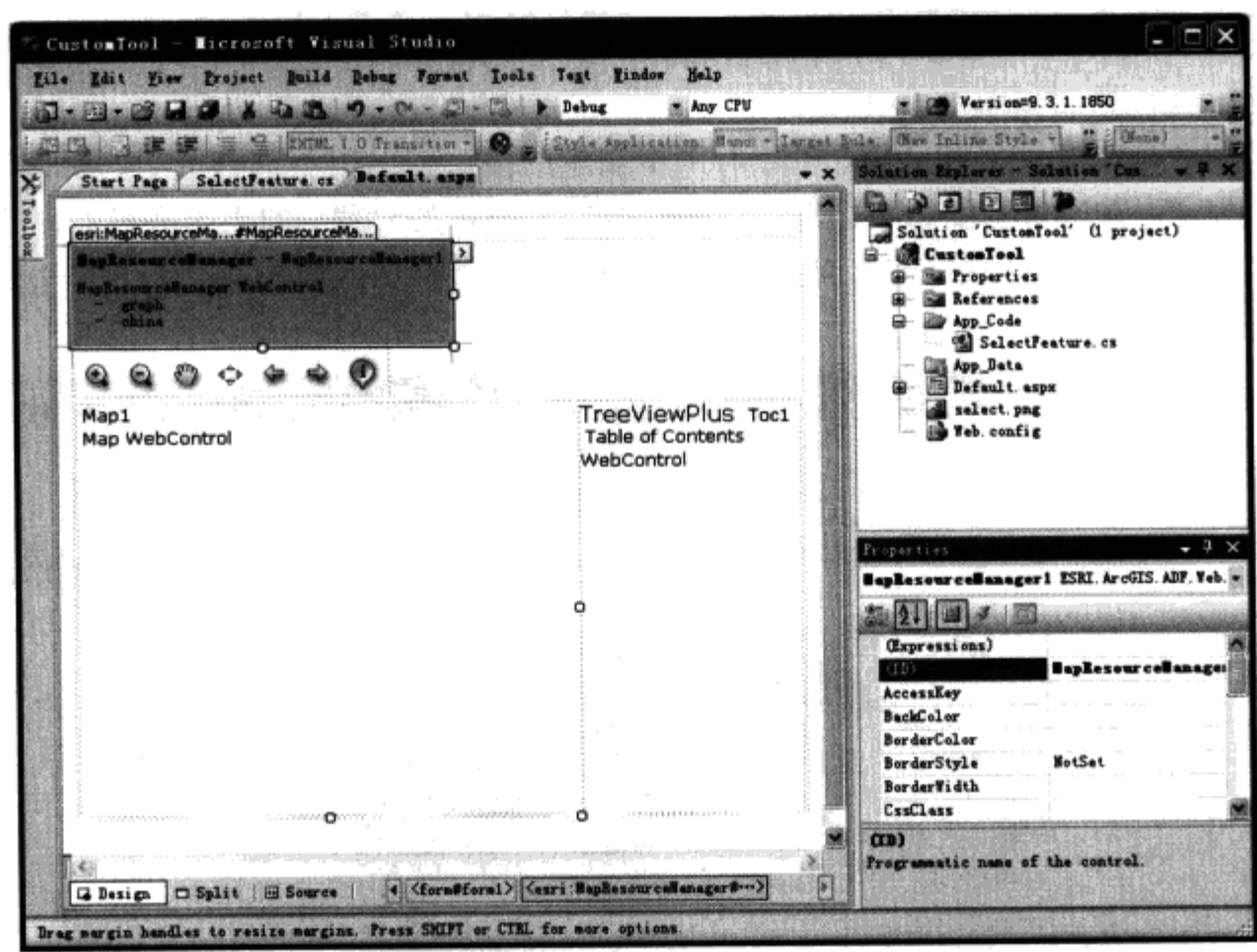


图 7.20 定义配置界面

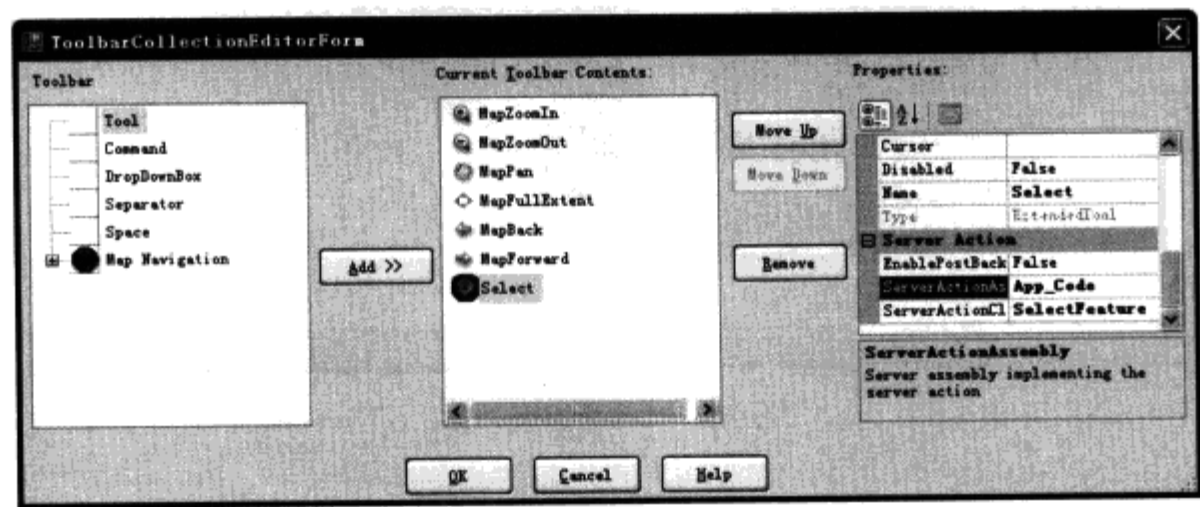


图 7.21 添加自定义 Tool

(3) 添加完自定义控件后，开始编写服务器的实现类。在实现服务器类的时候，要继承 IMapServerToolAction，客户端的 Click 事件通过封装好的 JS 代码来触发服务器端的 Server Action 方法，参数通过 ToolEventArgs 来传递，核心程序如下。

```
public void ServerAction(ToolEventArgs args)
{
    Map mapctrl = null;
    mapctrl = (Map)args.Control;
    string strName = "";
    PointEventArgs pointargs = null;
    pointargs = (PointEventArgs)args;

    System.Drawing.Point pt = pointargs.ScreenPoint;
```

```

//转换为地图上的点
ESRI.ArcGIS.ADF.Web.Geometry.Point point = ESRI.ArcGIS.ADF.Web.Geometry.
Point.ToMapPoint(pt.X, pt.Y, mapctrl.Extent, (int)mapctrl.Width.Value, (int)mapctrl.
Height.Value);

//查找图层
System.Collections.IEnumerable func_enum = null;
func_enum = mapctrl.GetFunctionalities();
ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource graphResource = null;

System.Data.DataTable datatable;
foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gisfunction in
func_enum)
{
    ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource = null;
    gisresource = gisfunction.Resource;
    if (gisfunction.Resource.Name == "graph")
    {
        graphResource = (ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource)
gisfunction.Resource;//找到内存图像资源
    }
    bool supported = false;
    //地理资源是否支持查询
    supported = gisresource.SupportsFunctionality(typeof(ESRI.ArcGIS.ADF.
Web.DataSources.IQueryFunctionality));
    if (supported)
    {
        ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality qfunc;
        qfunc = (ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality) gisre
source.CreateFunctionality(typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctiona
lity), null);

        string[] lids;
        string[] lnames;

        qfunc.GetQueryableLayers(null, out lids, out lnames);
        if (lids == null) continue;//如果是遥感影像就不支持查询
        int layerindex = -1;
        for (int i = 0; i < lnames.Length; i++)
        {
            if (lnames[i] == "省界")
            {
                layerindex = i;
                break;
            }
        }

        if (layerindex > -1)
        {
            //找到该图层

```

```

        ESRI.ArcGIS.ADF.Web.SpatialFilter sfilter = new SpatialFilter();
        sfilter.ReturnADFGeometries = true;
        sfilter.MaxRecords = 100;
        sfilter.Geometry = point;

        datatable = qfunc.Query(null, lids[layerindex], sfilter);

        if (datatable != null)
        {
            if (datatable.Rows.Count > 0)
            {
                //选中查找的对象，并进行定位
                for (int j = 0; j < datatable.Columns.Count; j++)
                {
                    strName = datatable.Rows[0]["name"].ToString();
                    if (datatable.Columns[j].DataType == typeof(ESRI.
ArcGIS.ADF.Web.Geometry.Geometry))
                    {
                        //找到该集合对象
                        ESRI.ArcGIS.ADF.Web.Geometry.Geometry geom = (ESRI.
ArcGIS.ADF.Web.Geometry.Geometry)datatable.Rows[0][j];

                        ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer glayer = null;
                        foreach (System.Data.DataTable dt in graphResource.
Graphics.Tables)
                        {
                            if (dt is ESRI.ArcGIS.ADF.Web.Display.Graphics.
ElementGraphicsLayer)
                            {
                                glayer = (ESRI.ArcGIS.ADF.Web.Display.
Graphics.ElementGraphicsLayer)dt;
                                break;
                            }
                        }
                        if (glayer == null)
                        {
                            glayer = new ESRI.ArcGIS.ADF.Web.Display.
Graphics.ElementGraphicsLayer();
                            graphResource.Graphics.Tables.Add(glayer);
                        }
                        glayer.Clear();
                        ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement ge =
null;

                        ge = new ESRI.ArcGIS.ADF.Web.Display.Graphics.
GraphicElement(geom, System.Drawing.Color.Red);
                        ge.Symbol.Transparency = 50;
                        glayer.Add(ge);
                    }
                }
            }
        }
    }

```



```

    }
    }
    }
}

if (mapctrl.ImageBlendingMode == ImageBlendingMode.WebTier)
{
    mapctrl.Refresh();
}
else if (mapctrl.ImageBlendingMode == ImageBlendingMode.Browser)
{
    mapctrl.RefreshResource(graphResource.Name);
}

object[] oa = new object[1];
string sa = " alert('" + strName + "')";
oa[0] = sa;
CallbackResult crl = new CallbackResult(null, null, "javascript", oa);
mapctrl.CallbackResults.Add(crl);

```

(4) 系统运行结果如图 7.22 所示。详细代码参见本书附带光盘。

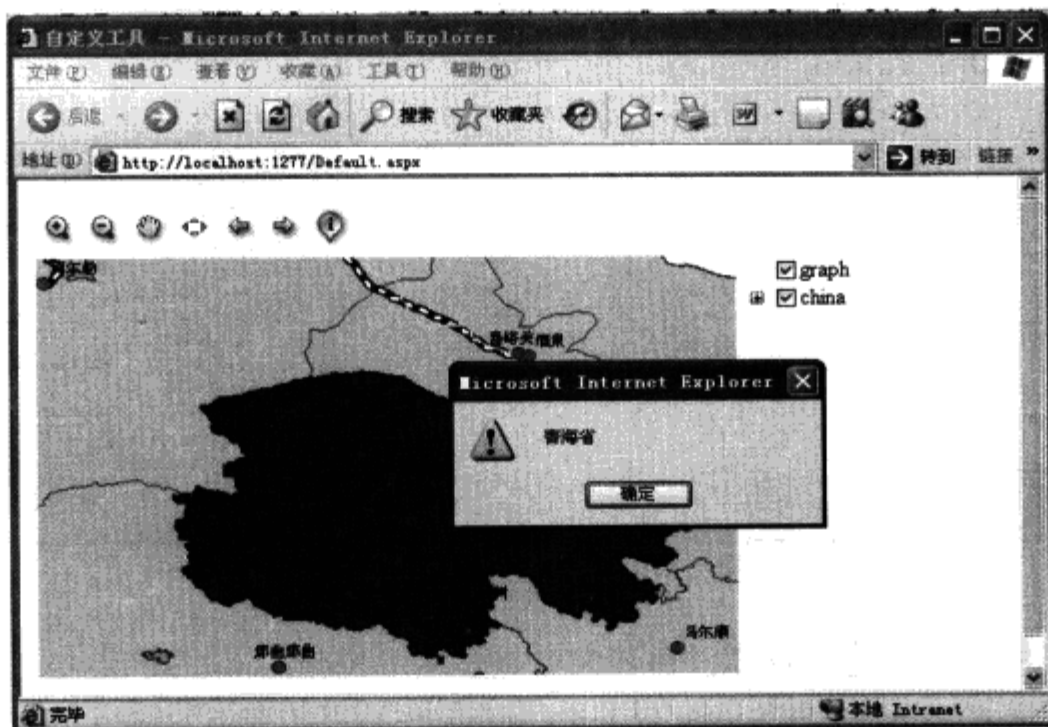


图 7.22 自定义 Tool 运行效果

应用程序运行时，不仅高亮显示选中的省空间对象，还提示省份名称。

7.5 自定义 Command

7.4 节已介绍当涉及空间交互时，通常采用自定义的 Tool 来实现。如果交互不是空间的呢？如客户端需要向服务器传递属性值。在 ArcGIS Server Web ADF 的开发中，有很多方式能够实现客户端与服务端端的交互，如 7.2 节所述就实现了客户端与服务端端的属性交互，除了采用 7.2 节中的方式，也可以采用自定义 Command。自定义 Command 顾名思义就是自定义命令行，用过 DOS 命令行的用户都知道命令行是可以携带参数的。笔者认为用自定义 Command 来实现这类功

能比较麻烦，要实现这类功能还是采用 7.2 节中的方式比较方便。

本节要介绍的自定义 Command，并非是带参数的命令行，而是不带参数的自定义 Command 的用法，如 Toolbar 的“全屏”，就是一个典型的 Command。客户端不需要向服务器传递任何参数，只需要发出一个请求，触发服务器的处理函数进行处理，并把处理结果返回给客户端供其刷新。

本节要讲述的例子，是根据 7.4 节介绍的功能来实现的。在 7.4 节中，利用自定义的 Tool 实现了客户端与服务器端的空间交互，在地图上单击一点，选择地图中某一图层的几何对象并高亮显示。但是，如果清除高亮显示时，可用自定义 Command 来实现，实现步骤如下。

(1) 在页面上放置好 MapResourceManager、Map、Toc、Toolbar 等相关控件，如图 7.23 所示。

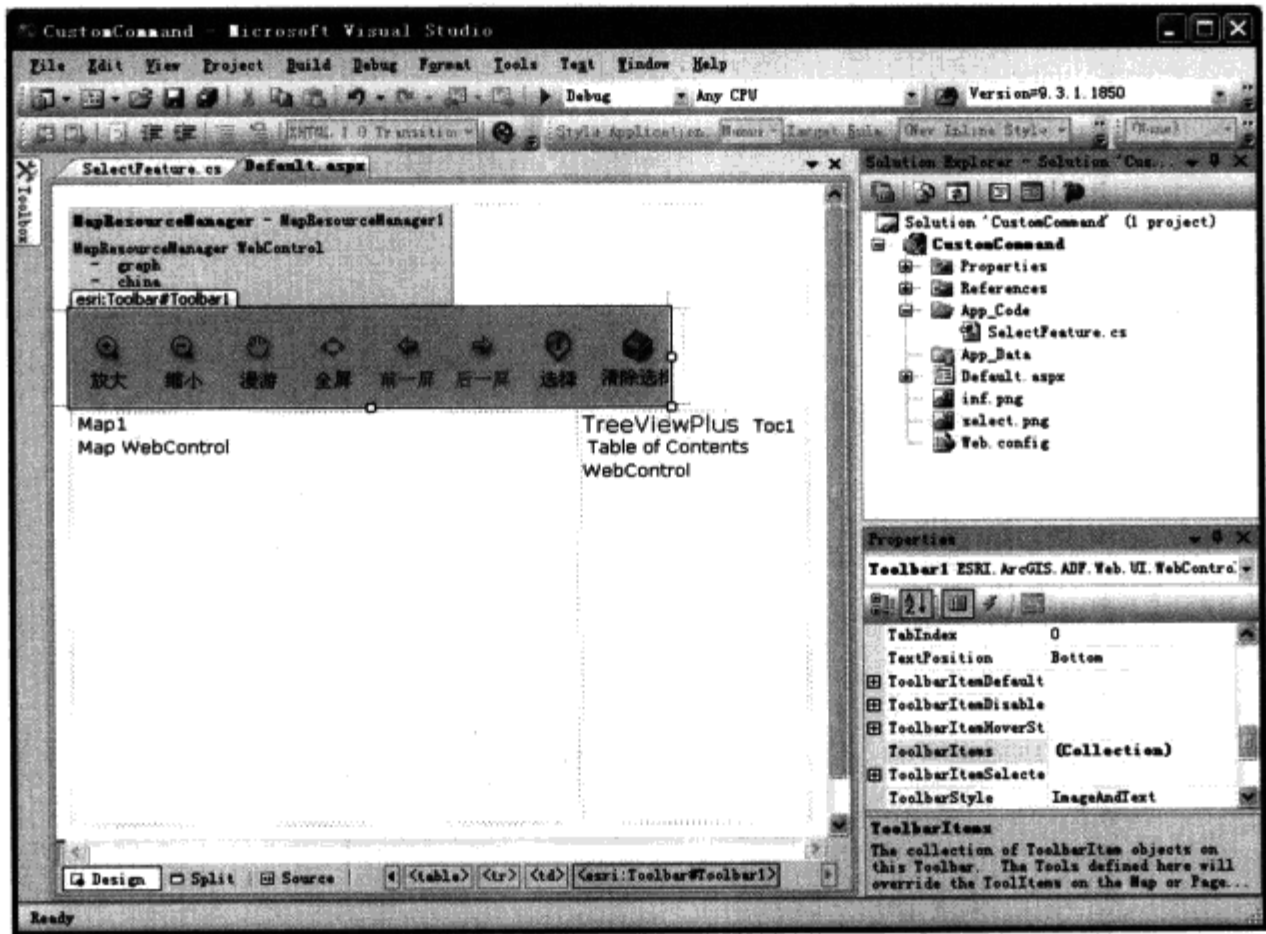


图 7.23 配置界面

将各控件的位置和大小进行布局，并且配置好各相关控件。

(2) 添加自定义 Commad。在 ServerActionAssembly 中填写实现类的路径，在 ServerActionClass 属性项输入实现类的类名称，如图 7.24 所示。

(3) 完成自定义控件的添加后，开始编写服务器的实现类。在实现服务器类的时候，要继承 IMapServerCommandAction，客户端的 Click 事件通过封装好的 JS 代码来触发服务器端的 ServerAction 方法，参数通过 ToolbarItemInfo 来传递，核心程序如下。

```
//相应 Command
public void ServerAction(ToolbarItemInfo iteminfo)
{
    Map mapctrl = null;
    mapctrl = (Map)iteminfo.BuddyControls[0]; //得到地图控件
```

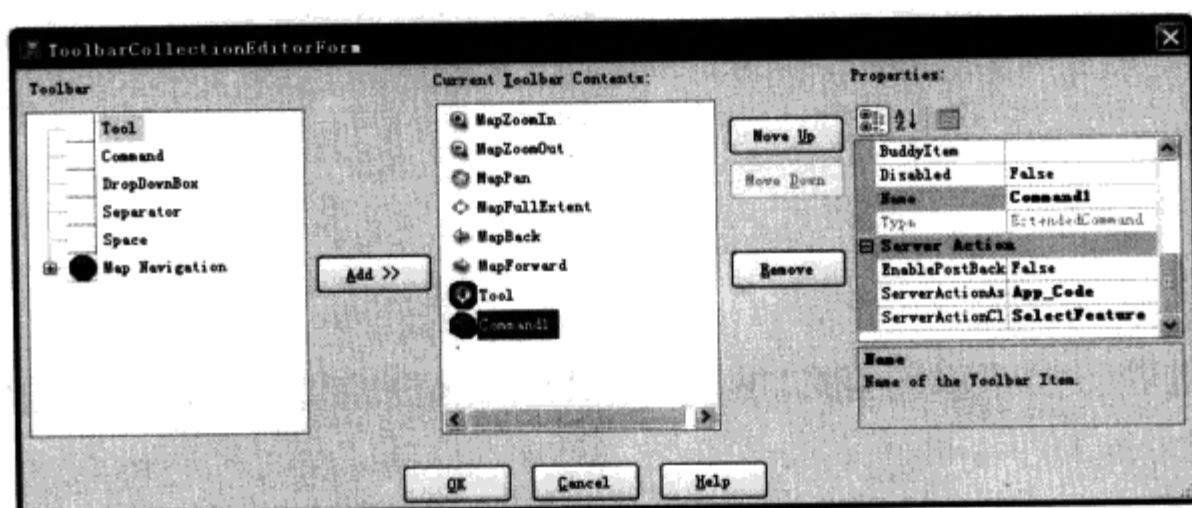


图 7.24 添加自定义 Command

```

ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapFunctionality gmf = null;
foreach (ESRI.ArcGIS.ADF.Web.DataSources.IMapFunctionality imf in mapctrl.
GetFunctionalities())
{
    if (imf.Resource.Name == "graph")//找到内存图像资源
    {
        gmf = (ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapFunctionality)imf;
        break;
    }
}

if (gmf == null)
{
    return;
}

ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer egl = null;
foreach (System.Data.DataTable dt in gmf.GraphicsDataSet.Tables)
{
    if (dt is ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer)
    {
        egl = (ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer)dt;
        break;
    }
}

if (egl != null)
{
    egl.Clear();//清除选中的图形
}

//刷新地图
if (mapctrl.ImageBlendingMode == ImageBlendingMode.WebTier)
{
    mapctrl.Refresh();
}

```

```
else if (mapctrl.ImageBlendingMode == ImageBlendingMode.Browser)
{
    mapctrl.RefreshResource(gmf.Resource.Name);
}
}
```

(4) 系统运行结果如图 7.25 所示。详细代码参见本书附带光盘。

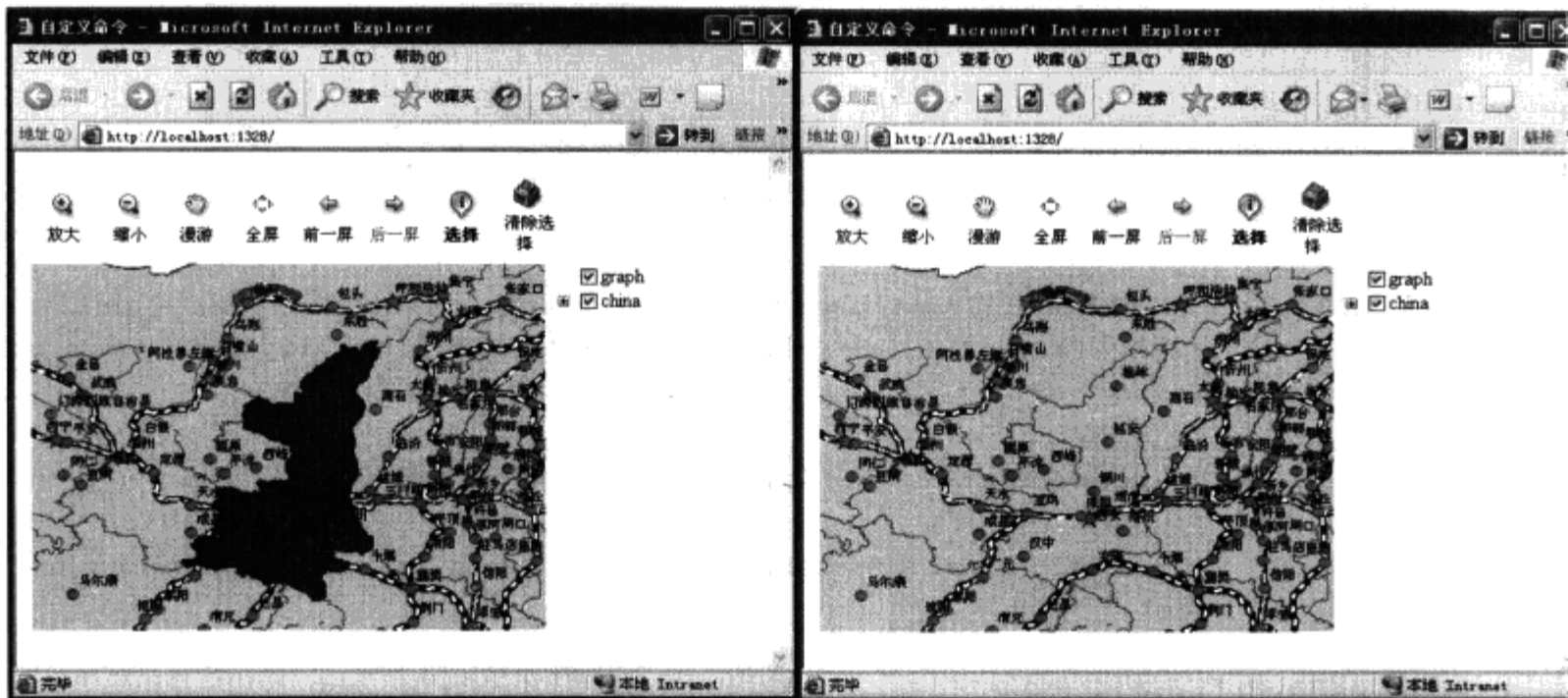


图 7.25 系统运行界面

单击选择“选择”按钮后，系统选中一个几何对象并高亮显示，单击“清除选择”按钮，系统清除刚才选中的几何对象。

7.6 ArcGIS Server Task

ArcGIS ServerTask 是由一组相关功能封装的集合。可以定制客户端的输入，如可以定义输入地理位置、坐标以及地址的相关信息，也可以自定义客户端的相关输入界面。ArcGIS Server 提供一些现成的 Task，可以直接供开发人员使用。这些 Task 包括：SearchAttributeTask（搜索属性）、QueryAttributeTask（查询属性）、FindAddressTask（查找地址）、FindPlaceTask（查找定位）、EditorTask（编辑）、PrintTask（打印）、GeoprocessingTask（地理处理）等。这些 Task 都封装好了人机交互界面，有时候并不能完全满足需要，也可以自定义 Task。

所有的 Task 都有 TaskManager 管理，编辑 TaskManager 可以找到当前应用中所有的 Task，Task 执行的结果通常由 TaskResults 来展示。因此，一个基本页面中包含 TaskManager、TaskResults、Task 控件就可以实现 Task 功能。本节将先介绍 Task 的工作流程，然后讲述如何应用 Task，最后介绍如何实现自定义的 Task。

7.6.1 Task 的工作流程

ArcGIS Server 中的 Task 消息流程如图 7.26 所示。

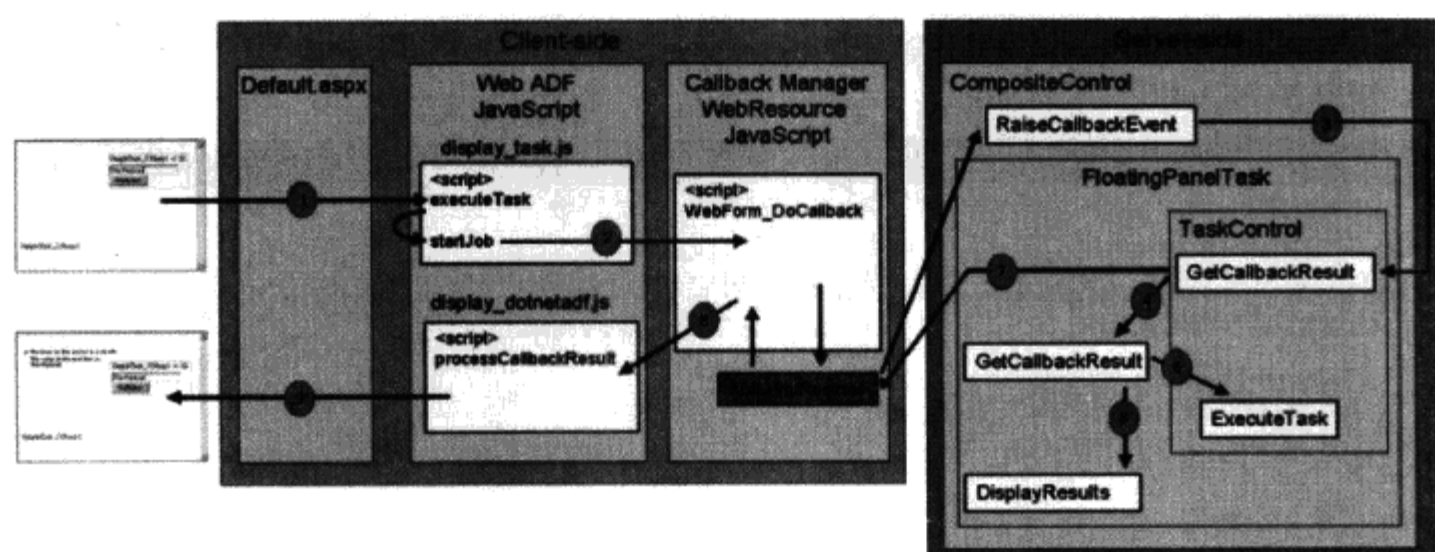


图 7.26 Task 消息流程图

Task 中消息传递步骤如下。

(1) 客户端执行触发事件, 发出异步请求。如常用的 button 的 click 事件, 用户在单击 button 的 click 时, 执行 Display_Task.js 文件的 executeTask() 方法, 该方法会调用这个 js 文件中的 StartJob() 方法: 启动一个 job 分配一个 jobid 号, 来跟踪这个异步请求

(2) startJob() 方法, 提供一系列的参数, 通过键值方式传送用户请求, 然后通过参数 _callbackArg 传入 RaiseCallbackEvent 方法, RaiseCallbackEvent 方法就是 Microsoft 回调的主体。也就是说, 通过 startJob 方法, 把 ArcGIS Server 中的 JavaScript 和微软的 JavaScript 融合在一起。

(3) 服务器端处理回调函数。回调方法会自动调用 FloatingPanelTask 中的 GetCallBackResult(), 这样就从客户端进入到服务器端。

(4) GetCallBackResult() 方法会根据客户端传入的相应参数做出相应的响应。

(5) exectueTask() 就是 FloatPanelTask 的主体方法, 其中主要封装了业务处理逻辑。在此方法中把处理的结果保存成 Results, 可以是 Result、dataset 和 TaskResultNode。

(6) executeTask() 方法处理完之后, 就会调用 DisplayResults() 方法, 该方法显示处理的结果。

(7) 在第 (3) 步中会把操作的结果返回给客户端, 即调用 base.GetCallResult() 方法。也就是把服务器的处理结果返回给客户端。

(8) 通过 processCallbackResult() 函数处理服务器端的返回结果。

(9) 显示处理结果。

整个处理过程是一个经典的异步刷新过程, 首先由客户端发出异步请求, 然后服务器接收到客户端传入的参数并执行相关的处理, 最后把处理结果返回给客户端显示。

7.6.2 应用 ArcGIS Server Task

ArcGIS Server 提供的 Task 应用起来相对简单, 除了使用普通的地图浏览那些控件外, 还需要一些与使用 Task 相关的控件: Menu、TaskManager、Task、Taskresult。Menu 控件主要用来与 TaskManager 相关联, 控制 Task 的显示; TaskManager 是 Task 的管理器; Taskresult 用来显示 Task 执行的结果。下面介绍 SearchAttributesTask 的应用实例。

(1) 在界面上布局 MapResourceManager、Map、TOC、Toolbar、Menu、TaskManager、Task、

Taskresult 各控件。有关 MapResourceManager、Map、TOC、Toolbar 控件的配置参见前面几小节的内容。控件的布局如图 7.27 所示。

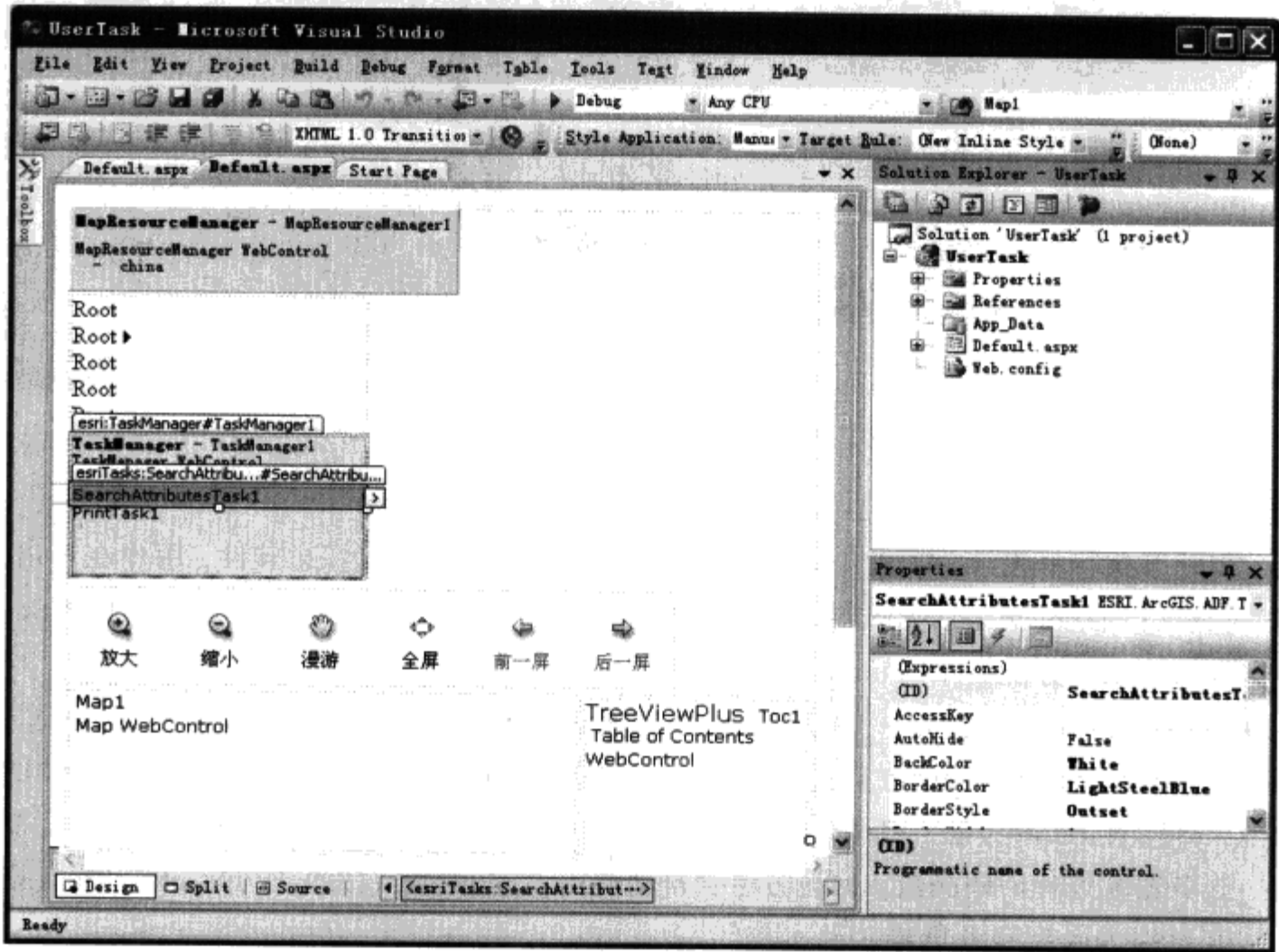


图 7.27 系统配置界面

(2) 配置 TaskManager 的 BuddyControl 属性。选择菜单名字，这样就能用菜单来配置 TaskManager 控制 Task 的显示，如图 7.28 所示。

(3) 配置 SearchAttributesTask 查询结果的显示控件，如图 7.29 所示。

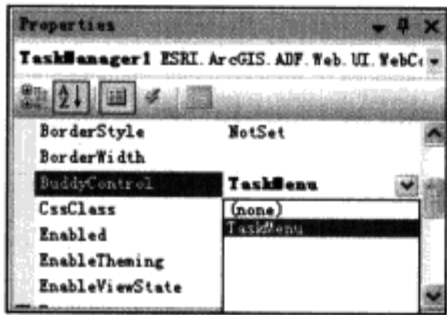


图 7.28 配置 BuddControl 属性

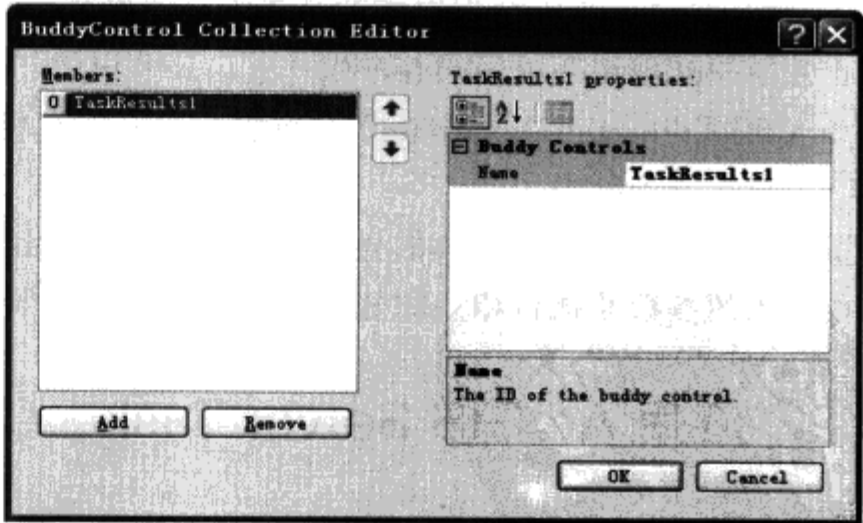


图 7.29 配置 Task 的结果显示控件

(4) 配置 SearchAttributesTask 的查询图层和查询字段，如图 7.30 所示。可将 SearchAttributes Task 的有关查询提示信息改为中文，只需要在相应的属性项输入提供的中文信息即可。如在 LabelText 属性项中输入“请输入相应的省份名称”。

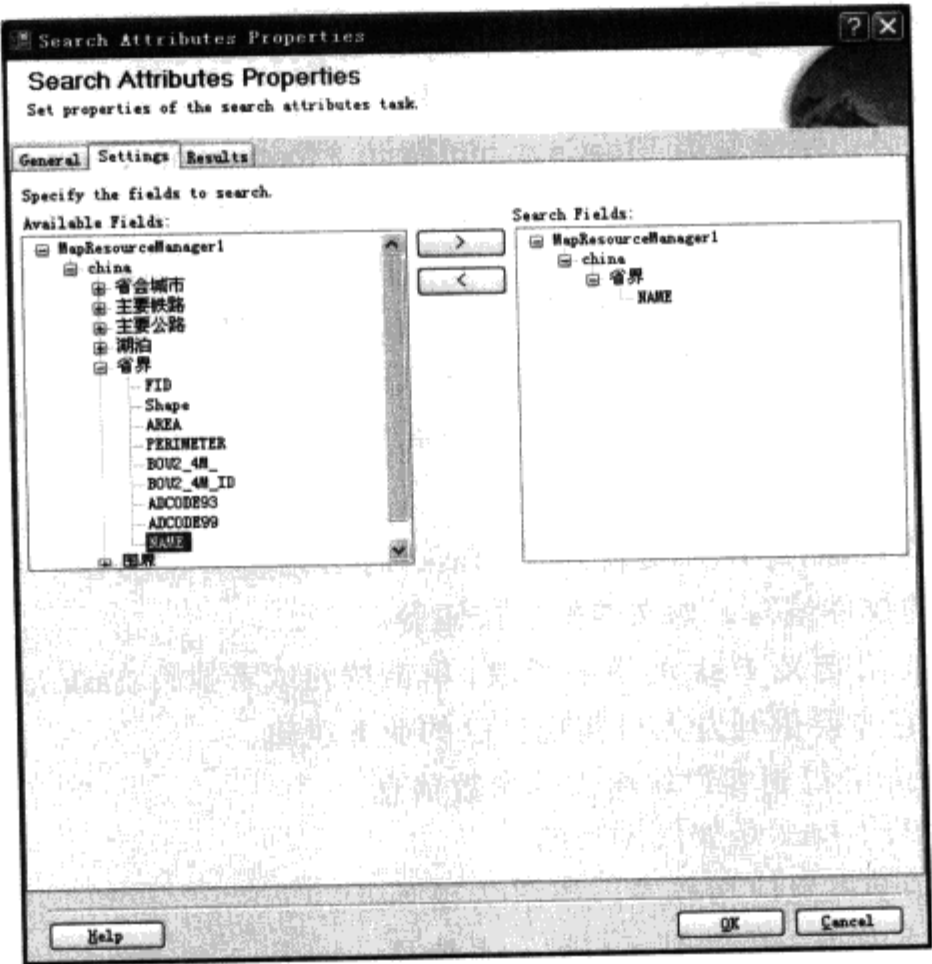


图 7.30 配置 SearchAttributesTask 的查询图层和查询字段

(5) 配置完成后，运行应用程序，如图 7.31 所示。

ArcGIS Server 提供的其他 Task 与 SearchAttributesTask 的使用步骤差不多，也不需要编写任何代码，就能轻松实现一个 GIS 的常用功能。

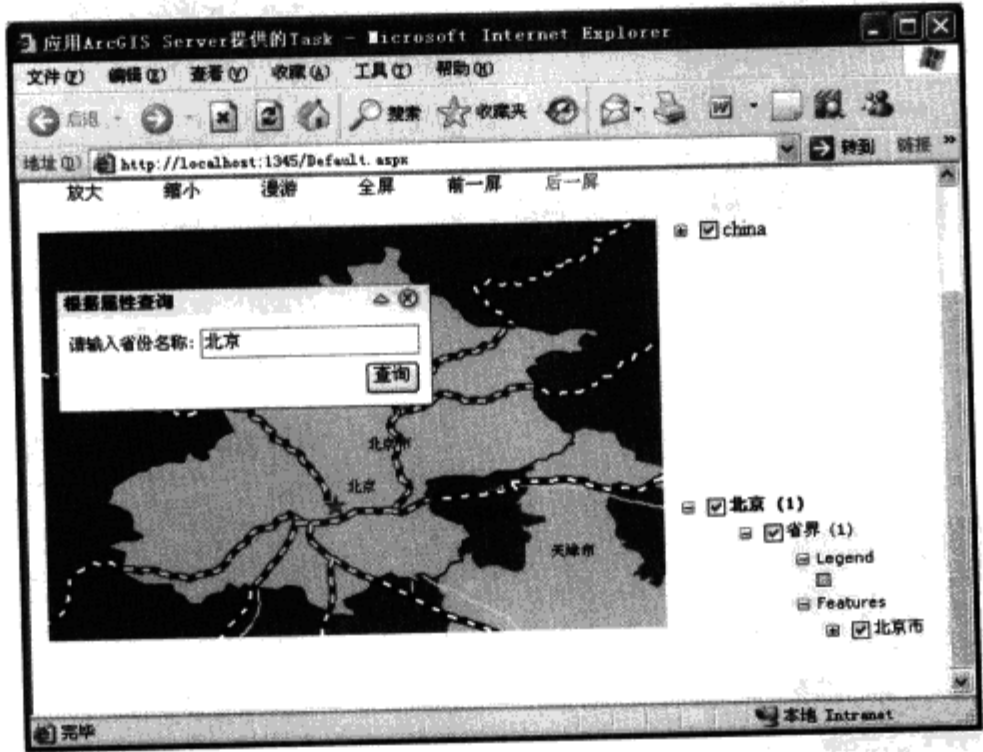


图 7.31 SearchAttributeTask 运行界面

7.6.3 自定义 Task

ArcGIS Server 软件提供的多种 Task，功能强大而且应用简单，只需要进行简单的配置就可

以了。然而，Task 的输出格式或者是方式，有时候与我们的应用习惯并不相符，因此在很多情况下都需要编写自己的 Task——也就是自定义 Task。

实现自定义 Task 必须继承基类 FloatingPanelTask。FloatingPanelTask 可以让开发人员在浏览器界面拖动其中的内容，可以设置让用户进行大小的调整、控件内容隐藏以及关闭控件，可以通过设置接口属性来调整控件的位置。自定义 Task 与自定义 Tool 一样，只要按照一定的规则实现这些接口或抽象类的方法即可。其他的如客户端与服务器端的交互，Web ADF 的框架都已经帮我们封装好了，开发人员只需要实现业务逻辑就可以了。

实现自定义的 Task，有几个关键的方法比较重要，开发人员在开发时经常会使用到这些方法。

- CreateChildControls()是在开发自定义 Task 时必须复写的方法，它的主要功能是实现 FloatPanel 上的控件编写，如文本框、按钮等。
- ExecuteTask()是自定义 Task 的核心函数，单击按钮或者执行 Task 最后操作的动作都会调用这个方法。它主要供开发人员实现自己的业务逻辑。
- GetCallbackResult()得到客户端输入的参数信息。

下面介绍一个简单的 Task 是如何编写的。

(1) 先配置一个地图浏览应用程序，如图 7.32 所示。

(2) 选中项目解决方法，单击鼠标右键，系统显示如图 7.33 所示的菜单。选择“Add”→“New Project”菜单项。

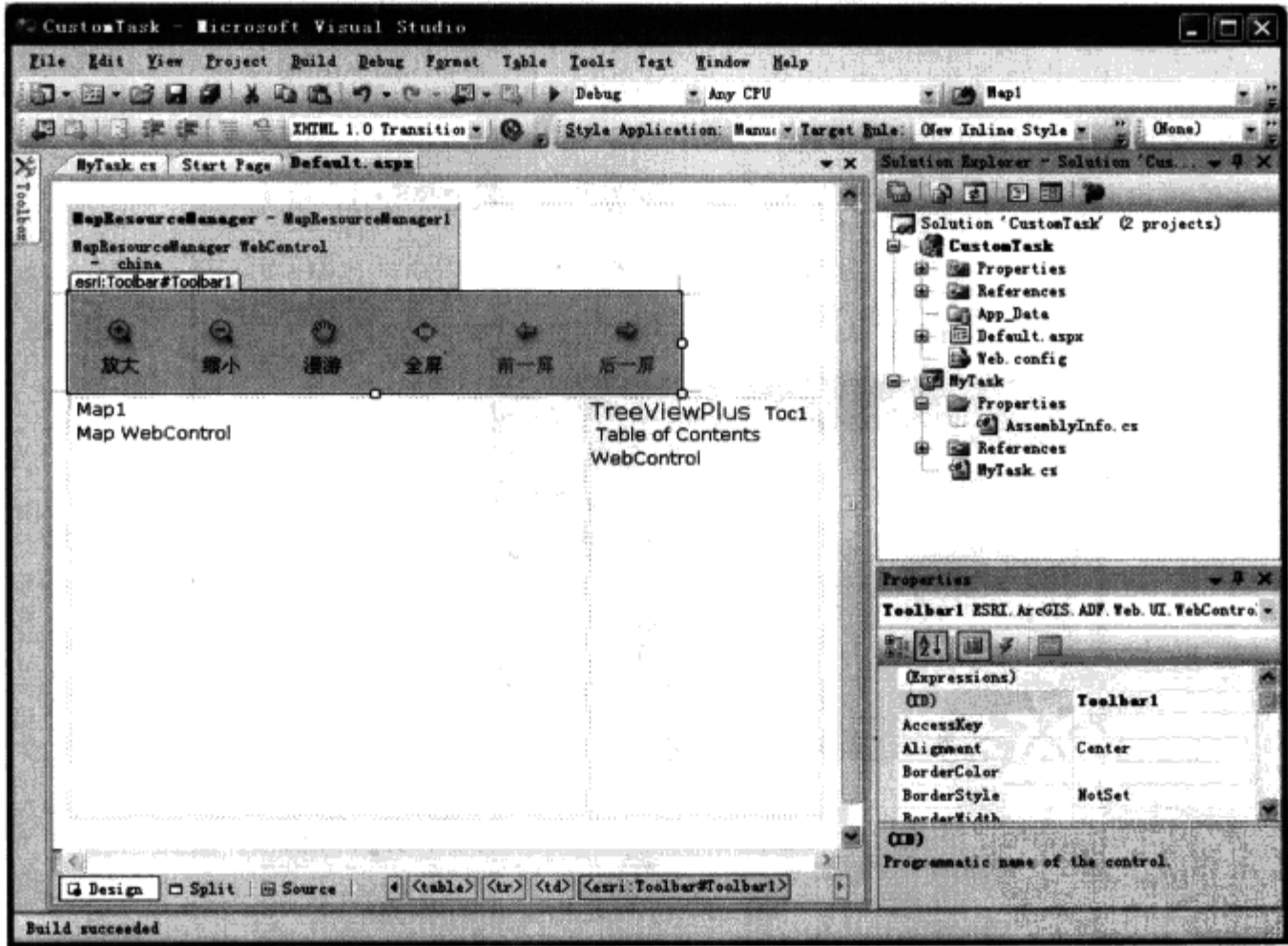


图 7.32 配置一个地图浏览项目

(3) 系统弹出“Add New Project”对话框，在“Templates”中选择“Class Library”选项，在“Name”文本框中输入项目的名字，如图 7.34 所示。

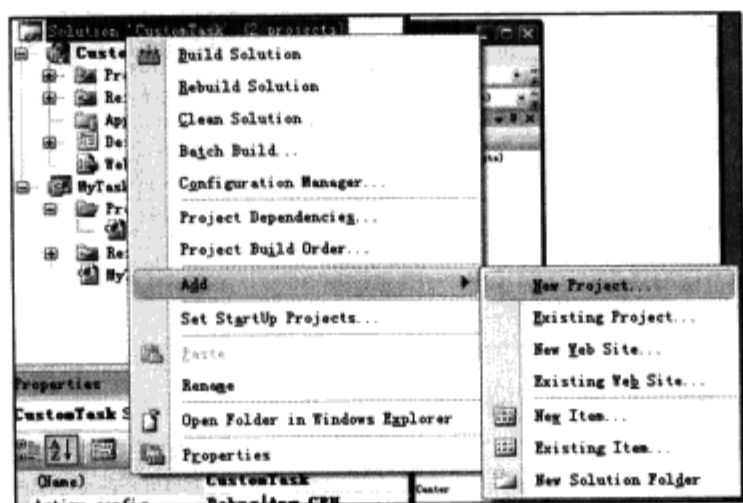


图 7.33 添加新项目菜单

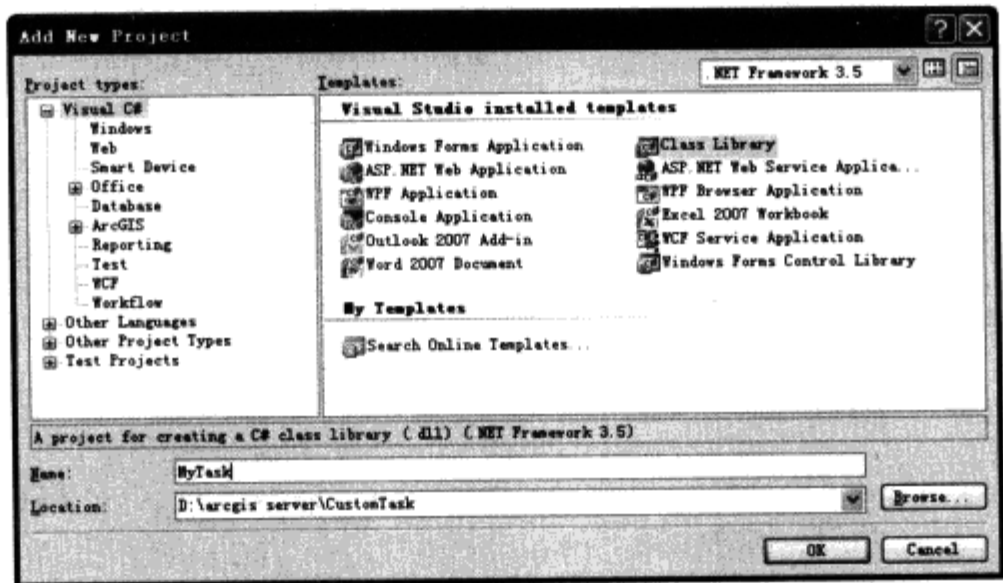


图 7.34 “Add New Project” 对话框

(4) 编写 MyTask 的实现程序：

```
namespace UserTask
{
    public class MyTask:ESRI.ArcGIS.ADF.Web.UI.WebControls.FloatingPanelTask
    {
        //Panel 上面的控件
        private TextBox txtbox1 = null;
        private HtmlInputButton button = null;

        //添加加载页面上的子控件
        protected override void CreateChildControls()
        {
            //清除 floatPanel 上的所有控件，然后重新加载
            Controls.Clear();

            //调用基类创建控件的方法
            base.CreateChildControls();

            //创建 TextBox 控件
            txtbox1 = new TextBox();
            txtbox1.ID = "textbox1";
```

```

        //创建 button
        button = new HtmlInputButton();
        button.ID = "OkBut";
        button.Value = "确定";

        //把控件添加到 floatpanel 中
        Controls.Add(txtbox1);
        Controls.Add(button);

        //声明各种 JavaScript 的代码
        string getArgumentJS = string.Format("'txtboxvalue=' + document.
getElementById('{0}').value", txtbox1.ClientID);
        string onClick = string.Format("executeTask({0}, \"{1}\");", getArgument
JS, this.CallbackFunctionString);
        string onKeyDown = string.Format("if(event.keyCode==13){ {0}return
false; }", onClick);
        button.Attributes.Add("onclick", onClick);
        txtbox1.Attributes.Add("onkeydown", onKeyDown);
    }

    //得到输入的值
    public override string GetCallbackResult()
    {
        NameValueCollection kevalcoll = CallbackUtility.ParseStringIntoName
ValueCollection(this._callbackArg);
        this.input = kevalcoll["txtboxvalue"];
        return base.GetCallbackResult();
    }

    //执行主业务逻辑区
    public override void ExecuteTask()
    {
        this.Results = null;
        if (this.input == null) return;

        string txtvalue = this.input as string;
        //声明显示的结果
        string heading = string.Format("The time on the server is {0}",
DateTime.Now.ToShortDateString());
        string detail = string.Format("The value in the Textbox is:{0}",
txtvalue);
        //把结果保存成一个 SimpleTaskResult 对象
        SimpleTaskResult str = new SimpleTaskResult(heading, detail);
        this.Results = str;
        // throw new NotImplementedException();
    }

    public override System.Collections.Generic.List<GISResourceItemDependency>
GetGISResourceItemDependencies()
    {
        System.Collections.Generic.List<GISResourceItemDependency> list = new
System.Collections.Generic.List<GISResourceItemDependency>();
    }

```



```

        return list;
        //throw new NotImplementedException();
    }
}
}

```

代码实现的关键点为。

- protected override void CreateChildControls() 在 Task 实例化时被调用，用来创建用户界面和控件的客户端事件来触发时执行的 JavaScript 脚本代码。
- string onClick = string.Format("executeTask({0},{1})", getArgumentJS, this.CallbackFunctionString); button.Attributes.Add("onclick", onClick); 就是设置 button 的 onclick 事件上注册 javascript 方法。调用的就是 ESRI JavaScript 脚本中的 executeTask() 方法，传入的参数是自定义的参数字符串和微软的 callback 字符串。
- public override string GetCallbackResult() 方法，就是在事件触发后执行的方法，即 JavaScript 中执行 executeTask() 方法后在服务器首先调用的方法。该方法主要用来得到客户端传递的参数，并把得到的值放在 input 属性中。
- public override void ExecuteTask() 执行完 GetCallbackResult() 方法后，系统就执行 ExecuteTask() 方法，该方法主要根据传入的参数进行处理，并把处理的结果返回。

(5) 在代码编译完成后，系统会在 Toolbox 标签页添加一个 MyTask 的控件，如图 7.35 所示。

(6) 把 MyTask 拖到页面上，同时拖入 TaskResult 控件，并把 MyTask 的结果控件进行配置，如图 7.36 所示。

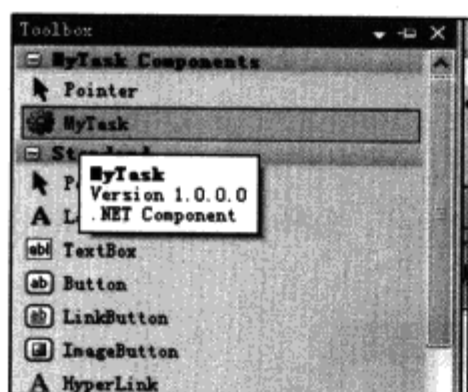


图 7.35 生成的 MyTask 控件

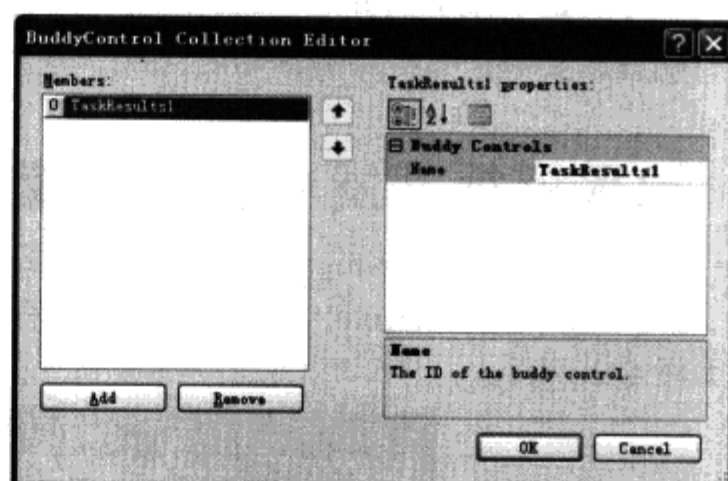


图 7.36 配置 MyTask 的结果显示控件

(7) 系统运行结果。

以上步骤就是一个简单的自定义 Task 实现过程，更复杂的 Task 的步骤与此类似，只是界面和业务逻辑更加复杂，读者在理解 Task 的工作流程与实现步骤后可以自己实现更高级的 Task。

7.7 小结

本章由浅入深地讲述了 ArcGIS Server .NET 的 ADF 开发过程。首先介绍了开发人员如何利用 ArcGIS Server 提供的控件不用编写任何代码就可以开发出一个简单的 WebGIS 应用，然后讲述了一些高级功能的开发，如 ArcGIS Server 调用 ArcObjects 的功能实现缓冲区，自定义 Tool、Command、Task，等等。读者掌握这些基本功能的开发，可实现一个常用的 WebGIS 应用程序的开发，并对 ArcGIS Server 开发流程，消息传递机制有更深入地了解。

第 8 章

ArcGIS Server 基于模板开发

第 7 章讲述了基于控件自定义的开发方法，在开发过程中发现，要把 ArcGIS Server 提供的各控件布置得合理，并不是一件很简单的事情。其实 ArcGIS Server 提供另外一种开发模式：基于模板的开发。基于模板的开发，已经构建了一个基本的 WebGIS 开发框架，开发人员只需要进行简单的配置就可以完成一个 WebGIS 应用程序的开发。

8.1 配置模板开发中资源配置

ArcGIS Server 基于模板的开发非常简单方便，开发步骤如下。

(1) 运行 Microsoft Visual Studio 2008 应用程序，单击菜单“File”，选择“New”→“Web Site”选项，如图 8.1 所示。

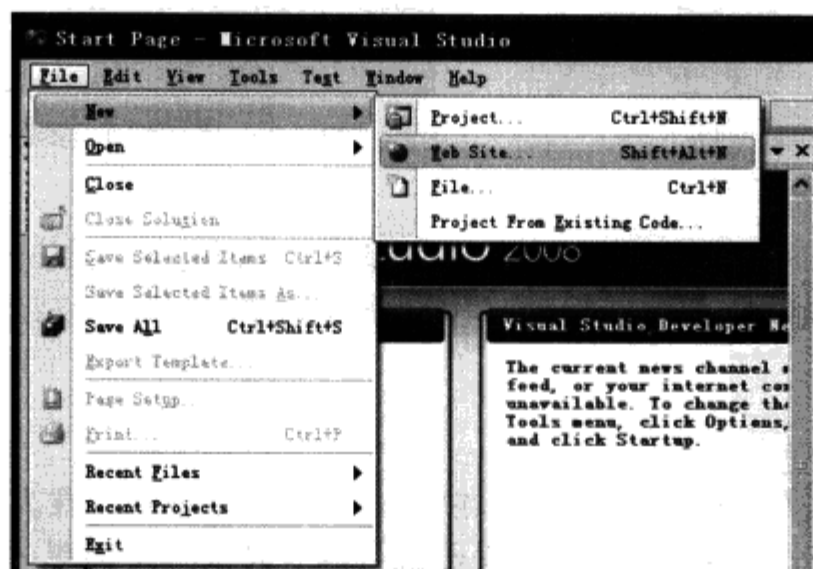


图 8.1 新建 Web Site

(2) 系统弹出“New Web Site”对话框，在“Templates”中选择“Web Mapping Application”选项，在“Location”文本框中输入应用程序的目录，如图 8.2 所示。

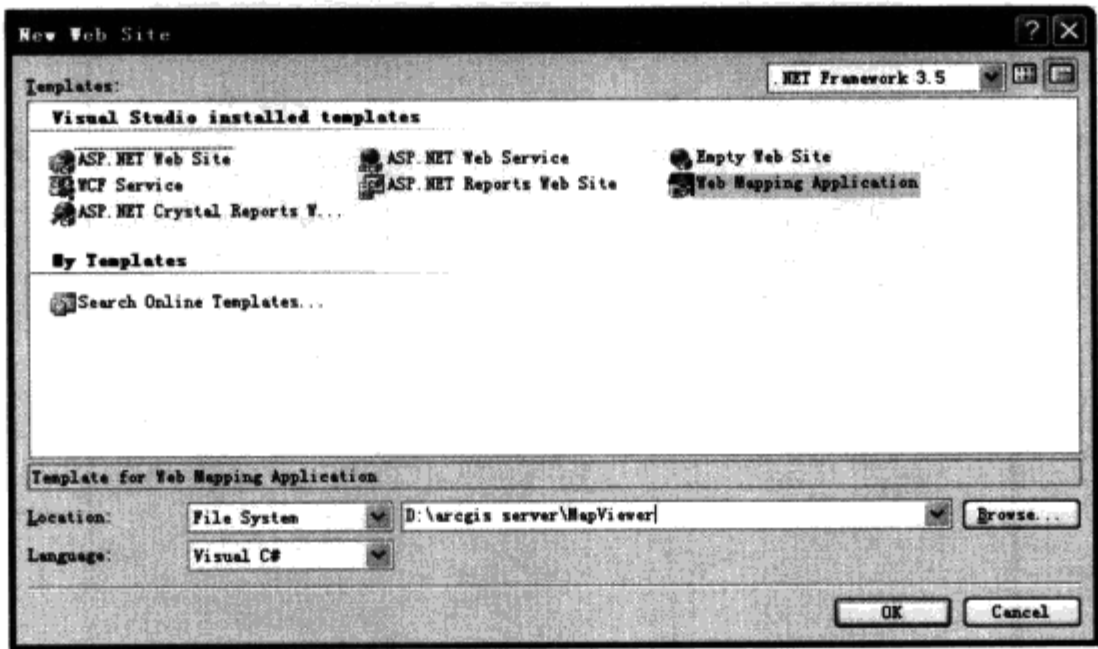


图 8.2 “New Web Site” 对话框

(3) 单击“OK”按钮，系统将生成一个 WebGIS 应用程序的框架程序，生成的程序各控件已经基本配置好，只需对 MapResourceManager1 控件进行配置后，一个 WebGIS 应用程序就开发完成了，如图 8.3 所示。

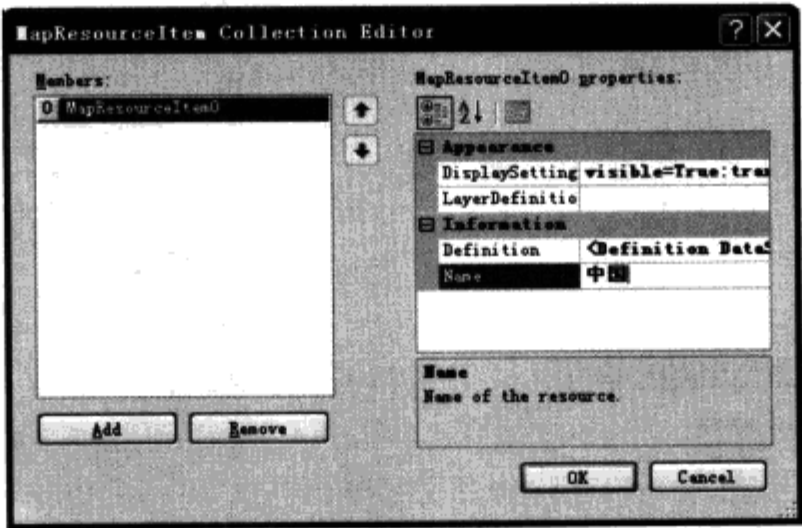


图 8.3 配置 MapResourceItem 地图资源

(4) 运行应用程序，一个界面布局合理的 WebGIS 应用网站生成了，如图 8.4 所示。其中 GIS 的基本功能包括：放大、缩小、漫游、全图显示、前一屏、后一屏；可以量测面积和长度；缩略图（鹰眼）、放大镜；空间要素属性查询；图层控制；比例尺显示以及地图上下左右移动功能。如果地图服务建立了 cache，还有一个可进行拖动的显示地图比例尺的控件。

(5) 由于该程序很多都是英文的，开发可以根据自己的习惯和需求改写这些提示。由于模板只提供空间到属性的查询，没有提供从属性到空间的查询，可以在 TaskManager1 中添加 SearchAttributesTask，并配置 SearchAttributesTask1 查询图层字段以及结果显示等属性，系统运行界面如图 8.5 所示。

(6) 应用程序的默认风格是“Blue_Bridge”，模板应用程序还提供其他的显示风格，如图 8.6 所示。开发人员可以在 web.config 文件根据自己的喜好修改显示风格。可在 web.config 文件中设置显示风格的脚本如<pages theme="Blue_Bridge">，开发人员可以自行设置。

(7) 模板应用程序提供 3 个 JavaScript 文件：display_measure.js、MapIdentify.js、WebMapApp.js。开发人员可以根据自己的需求来修改这些 JavaScript 文件，使之更符合我们的习惯。

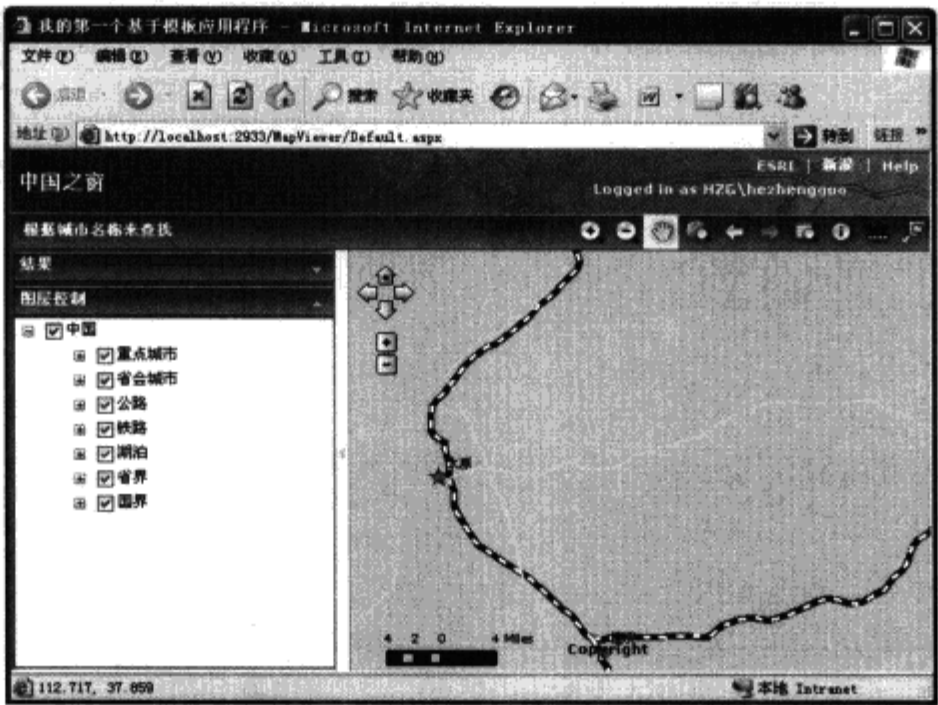


图 8.4 基于模板生成的应用程序

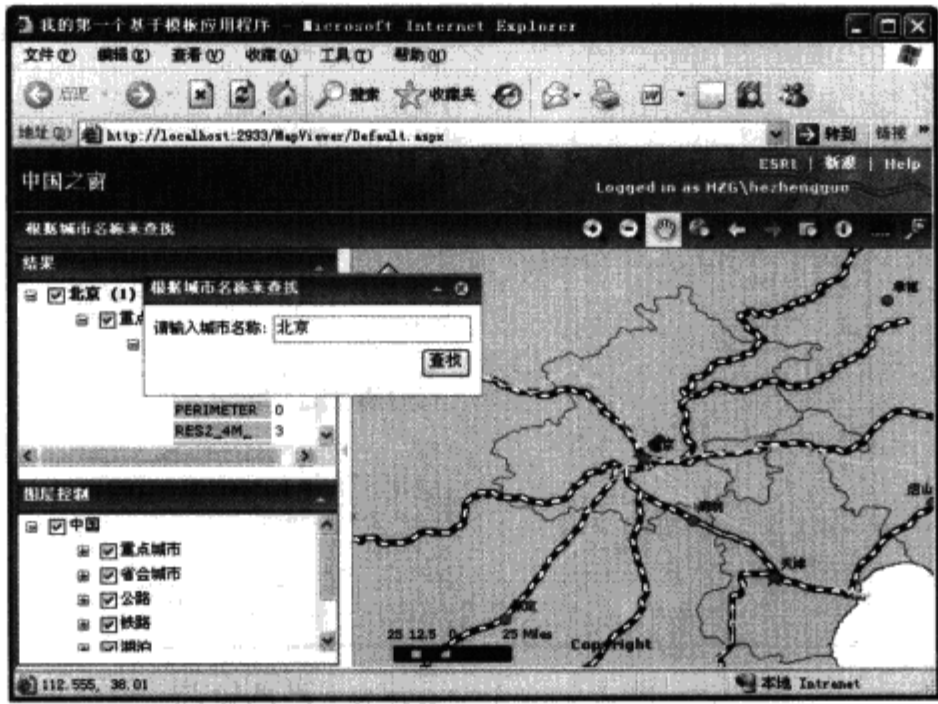


图 8.5 修改模板生应用程序的提示和添加 Task

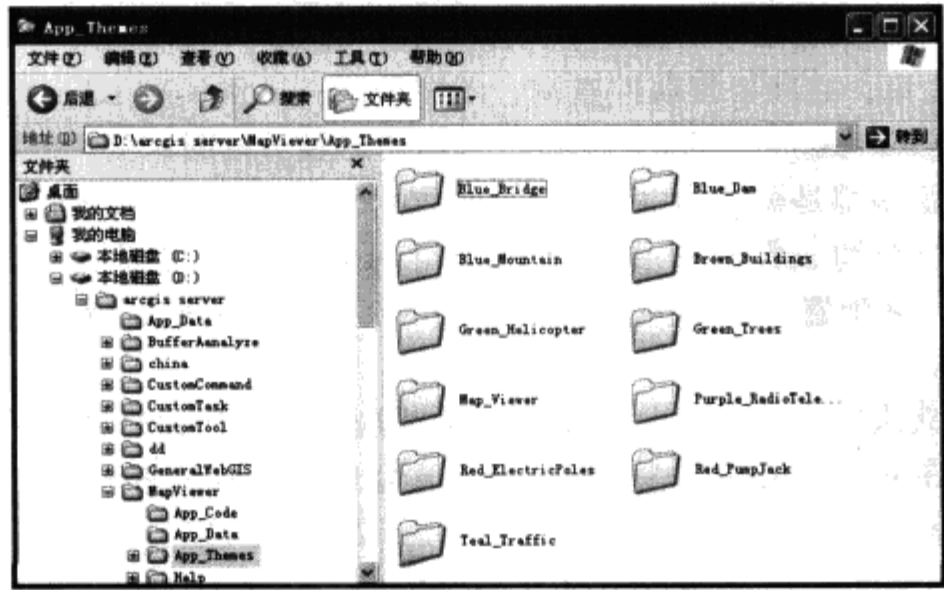


图 8.6 应用程序的显示风格

8.2 图形选择

在开发 GIS 应用程序的过程中,经常需要进行图形选择,其方式多种多样:单击选择、矩形选择、折线选择、多边形选择、画圆选择等。下面分别讲述在自定义 Tool 下不同的选择方式,有关自定义 Tool 在第 7 章已经介绍,本章主要介绍如何将屏幕上的对象转换为 ADF 上的对象。

8.2.1 单击选择

进行单击选择的时候,需要把屏幕上的点对象转换成 ADF 上的几何点对象,转换程序如下:

```
//把屏幕上的点转换为 Web ADF 上的点对象
private ESRI.ArcGIS.ADF.Web.Geometry.Geometry ConvertToADFGeometry(Point
EventArgs args, Map adfmap)
{
    System.Drawing.Point screenpoint = args.ScreenPoint;
    ESRI.ArcGIS.ADF.Web.Geometry.Point adfPoint = ESRI.ArcGIS.ADF.Web.Geometry.
Point.ToMapPoint(screenpoint.X, screenpoint.Y, adfmap.GetTransformationParams
(ESRI.ArcGIS.ADF.Web.Geometry.TransformationDirection.ToMap));
    return adfPoint;
}
```

在自定义控件中选择“ClientAction→Point”选项,如图 8.7 所示。



图 8.7 设置自定义控件的客户端操作(点转换)

8.2.2 矩形选择

矩形选择也需要把屏幕上的矩形转换成为 ADF 的矩形对象,也就是说把屏幕矩形的点转化为 ADF 的点,然后再构成 ADF 的矩形对象,转换程序如下:

```
//把屏幕上的点转换为 Web ADF 上的矩形对象
private ESRI.ArcGIS.ADF.Web.Geometry.Geometry ConvertToADFGeometry(Rectangle
EventArgs args, Map adfmap)
{
    System.Drawing.Rectangle rect = args.ScreenExtent;
    ESRI.ArcGIS.ADF.Web.Geometry.Point point1 = ESRI.ArcGIS.ADF.Web.Geometry.
Point.ToMapPoint(rect.Left, rect.Bottom, adfmap.GetTransformationParams(ESRI.ArcGIS.
ADF.Web.Geometry.TransformationDirection.ToMap));
    ESRI.ArcGIS.ADF.Web.Geometry.Point point2 = ESRI.ArcGIS.ADF.Web.Geometry.
```



```

Point.ToMapPoint(rect.Right, rect.Top, adfmap.GetTransformationParams(ESRI.ArcGIS.
ADF.Web.Geometry.TransformationDirection.ToMap));
    ESRI.ArcGIS.ADF.Web.Geometry.Envelope pnve = new ESRI.ArcGIS.ADF.Web.
Geometry.Envelope(point1, point2);
    return pnve;
}

```

在自定义控件中选择“ClientAction→DragRectangle”选项，如图 8.8 所示。

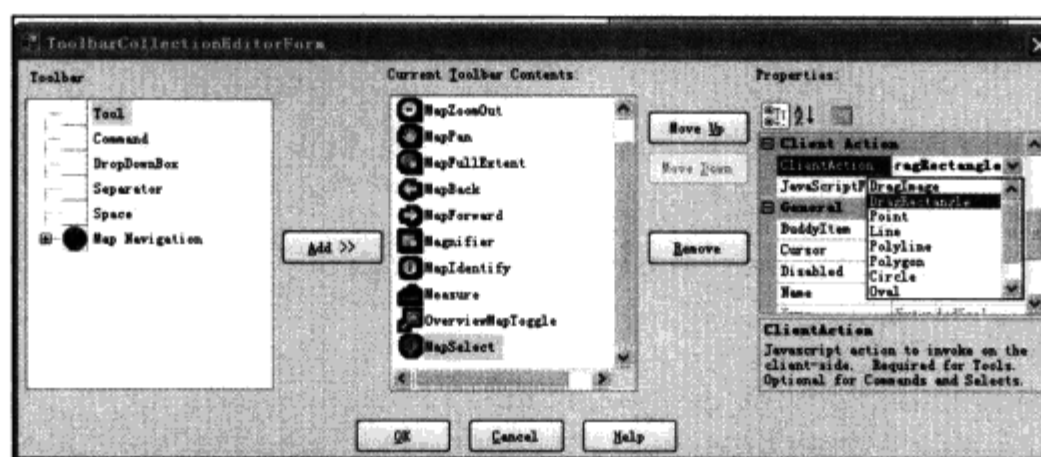


图 8.8 设置自定义控件的客户端操作（矩形转换）

8.2.3 折线选择

折线的转换方式与矩形相似，转换程序如下：

```

//把屏幕上的点转换为 Web ADF 上的折线对象
private ESRI.ArcGIS.ADF.Web.Geometry.Geometry ConvertToADFGeometry(Polyline
EventArgs args, Map adfmap)
{
    //ESRI.ArcGIS.ADF.Web.Geometry.Point adfPoint1 = ESRI.ArcGIS.ADF.Web.
Geometry.Point.ToMapPoint(args.BeginPoint.X, args.BeginPoint.Y, adfmap. Get
TransformationParams(ESRI.ArcGIS.ADF.Web.Geometry.TransformationDirection.ToMap));
    ESRI.ArcGIS.ADF.Web.Geometry.Path path = new ESRI.ArcGIS.ADF.Web.Geometry.
Path();
    for (int i = 0; i < args.Vectors.Length; i++)
    {
        ESRI.ArcGIS.ADF.Web.Geometry.Point adfPoint = ESRI.ArcGIS.ADF.Web.
Geometry.Point.ToMapPoint(args.Vectors[i].X, args.Vectors[i].Y, adfmap.Get
TransformationParams(ESRI.ArcGIS.ADF.Web.Geometry.TransformationDirection.ToMap));
        path.Points.Add(adfPoint);
    }

    ESRI.ArcGIS.ADF.Web.Geometry.Polyline polyline = new ESRI.ArcGIS.ADF.Web.
Geometry.Polyline();
    polyline.Paths.Add(path);

    return polyline;
}

```

自定义 Tool 客户端的设置如图 8.9 所示。

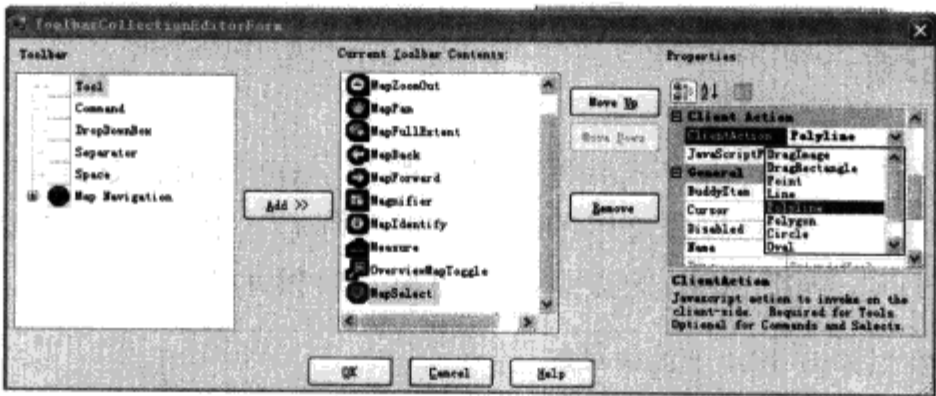


图 8.9 设置自定义控件的客户端操作（折线转换）

8.2.4 多边形选择

多边形的转换方式与折线的相似，转换程序如下：

```
//把屏幕上的点转换为 Web ADF 上的多边形对象
private ESRI.ArcGIS.ADF.Web.Geometry.Geometry ConvertToADFGeometry(Polygon
EventArgs args, Map adfmap)
{
    //ESRI.ArcGIS.ADF.Web.Geometry.Point adfPoint1 = ESRI.ArcGIS.ADF.Web.
    Geometry.Point.ToMapPoint(args.BeginPoint.X, args.BeginPoint.Y, adfmap.Get
    TransformationParams(ESRI.ArcGIS.ADF.Web.Geometry.TransformationDirection.ToMap));
    ESRI.ArcGIS.ADF.Web.Geometry.Ring ring = new ESRI.ArcGIS.ADF.Web.Geometry.
    Ring();
    for (int i = 0; i < args.Vectors.Length; i++)
    {
        ESRI.ArcGIS.ADF.Web.Geometry.Point adfPoint = ESRI.ArcGIS.ADF.Web.
        Geometry.Point.ToMapPoint(args.Vectors[i].X, args.Vectors[i].Y, adfmap.Get
        TransformationParams(ESRI.ArcGIS.ADF.Web.Geometry.TransformationDirection.ToMap));
        ring.Points.Add(adfPoint);
    }

    ESRI.ArcGIS.ADF.Web.Geometry.Polygon polygon = new ESRI.ArcGIS.ADF.
    Web.Geometry.Polygon();
    polygon.Rings.Add(ring);

    return polygon;
}
```

自定义 Tool 客户端的设置如图 8.10 所示。

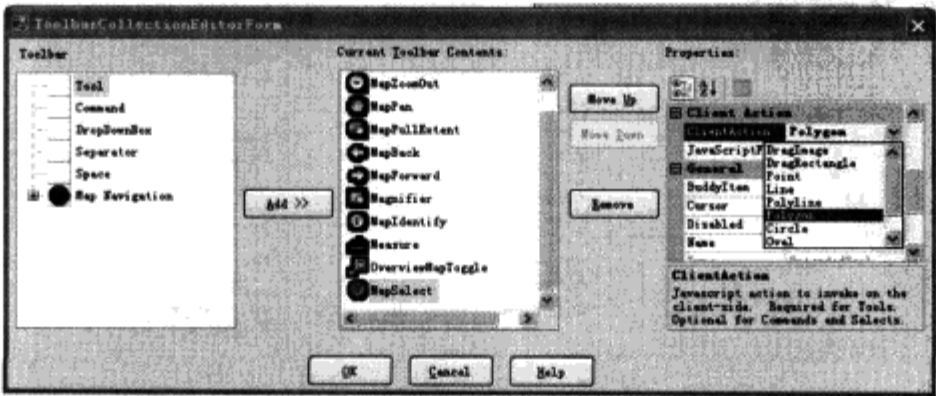


图 8.10 设置自定义控件的客户端操作（多边形转换）

8.2.5 画圆选择

圆的转换与上面所提到的转换方式有些区别，因为 ADF 没有圆这样的对象，所以只能把圆转换为多边形对象。在把圆转换为多边形的时候，根据圆心、半径，并在一定的角度取点后，把这些点转换为 ADF 的点，再把 ADF 的点组成多边形，转换示例程序如下：

```
//把屏幕上的点转换为 Web ADF 上的圆对象
private ESRI.ArcGIS.ADF.Web.Geometry.Geometry ConvertToADFGeometry( Circle
EventArgs args, Map adfmap)
{
    ESRI.ArcGIS.ADF.Web.Geometry.PointCollection pc = new ESRI.ArcGIS.ADF.
Web.Geometry.PointCollection();
    double degress;
    double rad = args.Radius;
    for (int i = 0; i < 360; i++)
    {
        degress = i * (Math.PI / 180);
        double x = args.CenterPoint.X + Math.Cos(degress) * rad;
        double y = args.CenterPoint.Y + Math.Sin(degress) * rad;
        ESRI.ArcGIS.ADF.Web.Geometry.Point pt = ESRI.ArcGIS.ADF.Web.Geometry.
Point.ToMapPoint((int)Math.Round(x), (int)Math.Round(y), adfmap.GetTransformation
Params(ESRI.ArcGIS.ADF.Web.Geometry.TransformationDirection.ToMap));
        pc.Add(pt);
    }

    ESRI.ArcGIS.ADF.Web.Geometry.Ring ring = new ESRI.ArcGIS.ADF.Web.Geometry.
Ring();
    ring.Points = pc;
    ESRI.ArcGIS.ADF.Web.Geometry.Polygon polygon = new ESRI.ArcGIS.ADF.Web.
Geometry.Polygon();

    polygon.Rings.Add(ring);

    return polygon;
}
```

自定义 Tool 客户端的设置如图 8.11 所示。

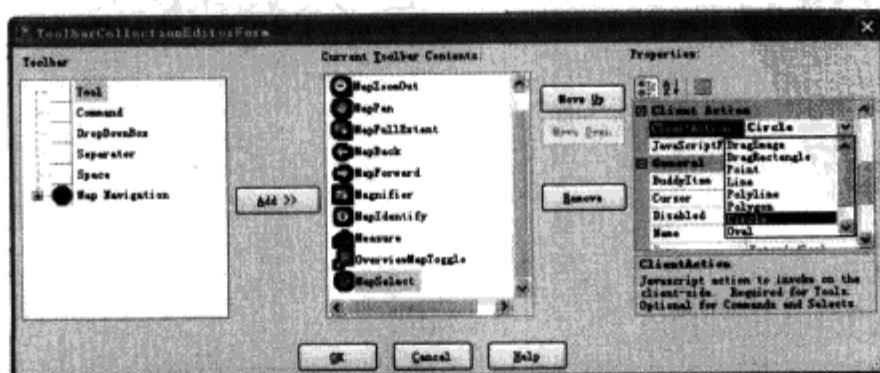


图 8.11 设置自定义控件的客户端操作（圆的转换）

8.3 控制地图图层的显示

ArcGIS 提供的 TOC 控件，无论是桌面的 TOC 控件，还是 Web 的，都存在一种缺点，就是在操作逻辑图层的时候父对象的状态可以影响地图的显示，但是不能影响子对象的 check 状态。既然 TOC 控件本身不提供这种功能，那么就需要我们开发这种功能。由于没有办法表示子对象部分被选中，因此只能提供全部选中，或者全部不选中的父与子的同步。

TOC 控件提供 NodeChecked 事件，该事件在图层打开或关闭时发生。实现父与子图层联动的关键程序如下：

```
protected void Toc1_NodeChecked(object sender, TreeViewPlusNodeEventArgs args)
{
    //判断该层是否有子节点
    TreeViewPlusNode layernode = args.Node;
    TravelChild(layernode, layernode.Checked);
    if (Map1.ImageBlendingMode == ESRI.ArcGIS.ADF.Web.UI.WebControls. Image
BlendingMode.Browser)
    {
        Map1.RefreshResource("MapResourceItem0");
    }
    else
    {
        Map1.Refresh();
    }
    Toc1.BuddyControl = "Map1";
    Toc1.Refresh();
}
```

浏览图层的递归函数程序：

```
private void TravelChild(TreeViewPlusNode node, bool bState)
{
    //该节点有子节点
    TreeViewPlusNode hChildNode;

    for (int i = 0; i < node.Nodes.Count; i++)
    {
        hChildNode = node.Nodes[i]; //找到该节点的第一个子节点

        hChildNode.Checked = bState;

        SynchroLayerState(hChildNode.Text, bState);
        if (hChildNode.Nodes.Count > 0)
        {
            //递归调用
            TravelChild(hChildNode, bState);
        }
    }
}
```

关闭或打开图层函数程序：

```
private void SynchroLayerState(string strLayerName, bool bState)
{
    System.Collections.IEnumerable func_enum = null;
    func_enum = Map1.GetFunctionalities();

    string[] layerIDs = null;
    string[] layerNames = null;
    foreach (ESRI.ArcGIS.ADF.Web.DataSources.IMapFunctionality mapfunction in
func_enum)
    {
        ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource =
mapfunction.Resource;
        bool supported = false;
        //地理资源是否支持查询
        supported = gisresource.SupportsFunctionality(typeof(ESRI.ArcGIS.ADF.
Web.DataSources.IQueryFunctionality));
        if (supported)
        {
            ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality
queryFunctionality = (ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality)gisresource.
CreateFunctionality
                (typeof(ESRI.ArcGIS.ADF.Web.DataSources.IQueryFunctionality), null);

            queryFunctionality.GetQueryableLayers(null, out layerIDs, out layerNames);
            //得到地图的图层

            for (int index = 0; index < layerIDs.Length; index++)
            {
                if (strLayerName == layerNames[index])
                {
                    mapfunction.SetLayerVisibility(layerIDs[index], bState);
                    return;
                }
            }
        }
    }
}
```

关闭或打开图层也可以在 ArcGIS Server 中用 ArcObjects 的方法来实现，读者有兴趣的话可以去试试。程序的运行效果图如图 8.12 所示。

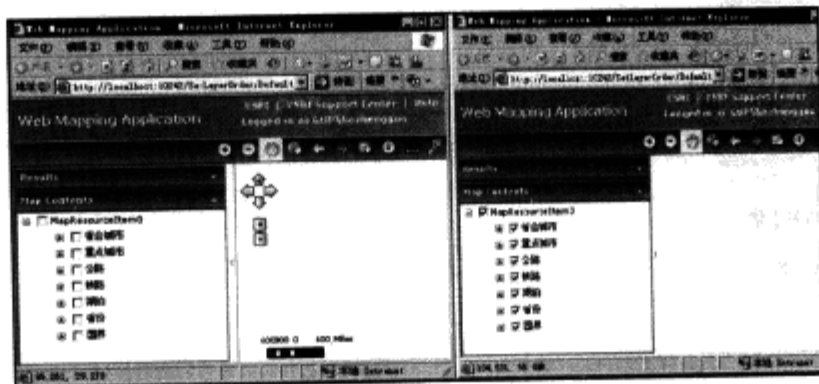


图 8.12 逻辑图层与子图层状态一致

8.4 TOC 图层移动

利用 ArcGIS Server 发布地图服务的时候, MXD 文件已经配置好了图层的顺序。很多时候, 不同的用户感兴趣的空间数据不同, 由于不同的图层数据之间有压盖, 所以用户往往需要调整 TOC 控件的图层顺序。使用 ArcObjects 开发的人都知道: MAP 对象中 MoveLayer 方法可以调整图层的顺序。这样在 ArcGIS Server 中就可以通过调用 ArcObjects 的功能来实现图层顺序的控制。实现步骤如下:

(1) 在 TOC 控件的图层上实现右键菜单。需要在页面上添加 ESRI 的 ContextMenu 控件, ContextMenu 菜单不能手工配置, 只能用程序来构建, 程序如下:

```
private void CreateContextMenu()
{
    if (this.ContextMenu1.Items.Count == 0)
    {
        ESRI.ArcGIS.ADF.Web.UI.WebControls.ContextMenuItem newItem = null;
        newItem = new ContextMenuItem("", "上移图层", null);
        ContextMenu1.Items.Add(newItem);
        newItem = new ContextMenuItem("", "下移图层", null);
        ContextMenu1.Items.Add(newItem);
    }

    for (int i = 0; i < Toc1.Nodes[0].Nodes.Count; i++)
    {
        ESRI.ArcGIS.ADF.Web.UI.WebControls.TreeViewPlusNode node = Toc1.Nodes
[0].Nodes[i];
        node.ClickBehavior = ESRI.ArcGIS.ADF.Web.UI.WebControls.TreeViewPlus Node
ClickBehavior.None;
        string showContextMenu = string.Format("esriShowContextMenu(event, '{0}',
'{1}','{2}')";this.style.backgroundColor='{3}';return false;", ContextMenu1.
ClientID, UniqueID, node.Value, System.Drawing.ColorTranslator.ToHtml(Toc1.Selected
Color));
        string onMouseOver = "this.style.cursor='arrow'";
        if (!node.TextCellAttributes.ContainsKey("oncontextmenu"))
            node.TextCellAttributes.Add("oncontextmenu", showContextMenu);
        if (!node.TextCellAttributes.ContainsKey("onmouseover"))
            node.TextCellAttributes.Add("onmouseover", onMouseOver);
        node.ToolTip = "单击右键弹出菜单";
    }
}
```

完成构建菜单的代码后, 可以在 Page_PreRenderComplete 方法中调用。

(2) 要使用 ArcObjects, 需要得到服务器上下文, 程序如下:

```
private ESRI.ArcGIS.Server.IServerContext getServerContext()
{
    ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.MapFunctionality mf = (ESRI.
ArcGIS.ADF.Web.DataSources.ArcGISServer.MapFunctionality)this.Map1.GetFunctionality(0);
```

```

        if (mf.MapResource is ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.
MapResourceLocal)
        {
            ESRI.ArcGIS.ADF.ArcGISServer.MapDescription mapDescription = mf.Map
Description;
            MapResourceLocal mapResourceLocal = mf.MapResource as MapResourceLocal;
            IServerContext sc = mapResourceLocal.ServerContextInfo.ServerContext;
            return sc;
        }
        return null;
    }
}

```

(3) 由于创建了“上移图层”和“下移图层”两个右键菜单，所以可以实现图层的上移或下移。上移图层程序如下：

```

public void MoveLayerUp(string layerName)
{
    IServerContext serverContext = getServerContext();
    IMapServer mapServer = serverContext.ServerObject as IMapServer;
    IMapServerObjects mapServerObj = (IMapServerObjects)mapServer;
    IMap map = mapServerObj.get_Map(mapServer.DefaultMapName);

    IGISFunctionality iFunctionality = Map1.GetFunctionality(0);
    MapResourceLocal mrl = iFunctionality.Resource as MapResourceLocal;
    mrl.ApplyMapDescriptionToServer();

    int gotcha = 0;
    string temp = "";
    int i, parent = 0;
    for (i = 0; i < Toc1.Nodes[0].Nodes.Count; i++)
    {
        if (Toc1.Nodes[0].Nodes[i].Text == layerName)
        {
            parent = i;
            temp = Toc1.Nodes[0].Nodes[i].Text;
            break;
        }
    }

    for (i = 0; i < Toc1.Nodes[0].Nodes.Count; i++)
    {
        if (mapServerObj.get_Layer(mapServer.DefaultMapName, i).Name == temp)
        {
            gotcha = i;
            break;
        }
    }

    if (parent > 0)
    {
        int lid = gotcha;

```

```

        int index = parent - 1;
        //移动图层
        map.MoveLayer(mapServerObj.get_Layer(mapServer.DefaultMapName, lid),
            index);
        mrl.RefreshServerObjects();//刷新对象

        Toc1.Refresh();
        Map1.CallbackResults.CopyFrom(Toc1.CallbackResults);
        Map1.Refresh();
    }
}

```

下移图层程序如下：

```

public void MoveLayerDown(string layerName)
{
    IServerContext serverContext = getServerContext();
    IMapServer mapServer = serverContext.ServerObject as IMapServer;
    IMapServerObjects mapServerObj = (IMapServerObjects)mapServer;
    IMap map = mapServerObj.get_Map(mapServer.DefaultMapName);

    IGISFunctionality iFunctionality = Map1.GetFunctionality(0);
    MapResourceLocal mrl = iFunctionality.Resource as MapResourceLocal;
    mrl.ApplyMapDescriptionToServer();

    int gotcha = 0;
    string temp = "";
    int i, parent = 0;
    for (i = 0; i < Toc1.Nodes[0].Nodes.Count; i++)
    {
        if (Toc1.Nodes[0].Nodes[i].Text == layerName)
        {
            parent = i;
            temp = Toc1.Nodes[0].Nodes[i].Text;
            break;
        }
    }

    for (i = 0; i < Toc1.Nodes[0].Nodes.Count; i++)
    {
        if (mapServerObj.get_Layer(mapServer.DefaultMapName, i).Name == temp)
        {
            gotcha = i;
            break;
        }
    }

    if (parent < Toc1.Nodes[0].Nodes.Count)
    {
        int lid = gotcha;
        int index = parent + 1;
    }
}

```

```

        //移动图层
        map.MoveLayer(mapServerObj.get_Layer(mapServer.DefaultMapName, lid),
index);

        mrl.RefreshServerObjects();//刷新

        Toc1.Refresh();
        Map1.CallbackResults.CopyFrom(Toc1.CallbackResults);
        Map1.Refresh();
    }
}

```

(4) 响应菜单的单击事件 ItemClicked。

```

protected void ContextMenu1_ItemClicked(object sender, ContextMenuItemEventArgs
args)
{
    ESRI.ArcGIS.ADF.Web.UI.WebControls.TreeViewPlusNode node =Toc1.Nodes. Find
ByValue(args.Context);

    if (node != null)
    {
        ESRI.ArcGIS.ADF.Web.UI.WebControls.MapResourceItem mapResourceItem =
null;

        int index = -1;
        switch (args.Item.Text)
        {
            case "上移图层":
                MoveLayerUp(node.Text);
                Map1.Refresh();
                ContextMenu1.CallbackResults.CopyFrom(Map1.CallbackResults);
                break;
            case "下移图层":
                MoveLayerDown(node.Text);
                Map1.Refresh();
                ContextMenu1.CallbackResults.CopyFrom(Map1.CallbackResults);
                break;
        }
    }
    Toc1.BuddyControl = "Map1";
    Toc1.Refresh();
    //setupContextMenu();
}

```

这样，就完成了 TOC 中右键菜单实现图层顺序的控制，效果如图 8.13 所示。

读者也可以参考本例来实现图层的其他功能，如 ArcMap 中右键可以实现查看图层属性等。8.4 节和 8.5 节的例子生成的一个程序，请参见本书附带光盘。


```

JavaScript.Append("var IpFile = GetTopFile(); \n");
JavaScript.Append("if(IpFile == null || IpFile.value == null ||
IpFile.value.length ==0)\n");
JavaScript.Append("{\n");
JavaScript.Append("alert('Please select a file to add.');

```

```

JavaScript.Append("BtnAdd.disabled = false;\n");
JavaScript.Append("}\n");
JavaScript.Append("}\n");
JavaScript.Append("Item.appendChild(A);\n");
JavaScript.Append("Item.onmouseover = function()\n");
JavaScript.Append("{\n");
JavaScript.Append("Item.bgColor = Item.style.backgroundColor;\n");
JavaScript.Append("Item.fColor = Item.style.color;\n");
JavaScript.Append("Item.style.backgroundColor = '#C6780B';\n");
JavaScript.Append("Item.style.color = '#ffffff';\n");
JavaScript.Append("Item.style.fontWeight = 'bold';\n");
JavaScript.Append("}\n");
JavaScript.Append("Item.onmouseout = function()\n");
JavaScript.Append("{\n");
JavaScript.Append("Item.style.backgroundColor = Item.bgColor;\n");
JavaScript.Append("Item.style.color = Item.fColor;\n");
JavaScript.Append("Item.style.fontWeight = 'normal';\n");
JavaScript.Append("}\n");
JavaScript.Append("return Item;\n");
JavaScript.Append("}\n");
JavaScript.Append("function Clear()\n");
JavaScript.Append("{\n");
JavaScript.Append("DivListBox.innerHTML = '';\n");
JavaScript.Append("DivFiles.innerHTML = '';\n");
JavaScript.Append("DivFiles.appendChild(CreateFile());\n");
JavaScript.Append("BtnAdd.disabled = false;\n");
JavaScript.Append("}\n");
JavaScript.Append("function GetTopFile()\n");
JavaScript.Append("{\n");
JavaScript.Append("var Inputs = DivFiles.getElementsByTagName('input'); \n");
JavaScript.Append("var IpFile = null;\n");
JavaScript.Append("for(var n = 0; n < Inputs.length && Inputs[n].type ==
'file'; ++n)\n");
JavaScript.Append("{\n");
JavaScript.Append("IpFile = Inputs[n];\n");
JavaScript.Append("break;\n");
JavaScript.Append("}\n");
JavaScript.Append("return IpFile;\n");
JavaScript.Append("}\n");
JavaScript.Append("function GetTotalFiles()\n");
JavaScript.Append("{\n");
JavaScript.Append("var Inputs = DivFiles.getElementsByTagName('input'); \n");
JavaScript.Append("var Counter = 0;\n");
JavaScript.Append("for(var n = 0; n < Inputs.length && Inputs[n].type ==
'file'; ++n)\n");
JavaScript.Append("Counter++;\n");
JavaScript.Append("return Counter;\n");
JavaScript.Append("}\n");
JavaScript.Append("function GetTotalItems()\n");
JavaScript.Append("{\n");

```

```

JavaScript.Append("var Items = DivListBox.getElementsByTagName('div');
\n");

JavaScript.Append("return Items.length;\n");
JavaScript.Append("}\n");
JavaScript.Append("function DisableTop()\n");
JavaScript.Append("{\n");
JavaScript.Append("if(GetTotalItems() == 0)\n");
JavaScript.Append("{\n");
JavaScript.Append("alert('Please browse at least one file to upload.');

```

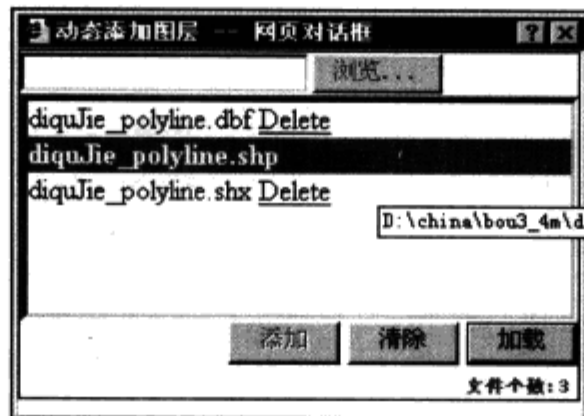


图 8.14 加载本地文件

服务器端上传文件程序：

```

HttpFileCollection oHttpFileCollection = e.PostedFiles;
HttpPostedFile oHttpPostedFile = null;
string strFileName="";
if (e.HasFiles)
{
    for (int n = 0; n < e.Count; n++)
    {
        oHttpPostedFile = oHttpFileCollection[n];
        int nPos1 = oHttpPostedFile.FileName.LastIndexOf('\\');
        int nPos2 = oHttpPostedFile.FileName.LastIndexOf('.');
        strFileName = oHttpPostedFile.FileName.Substring(nPos1 + 1, nPos2
- nPos1-1);

        if (oHttpPostedFile.ContentLength <= 0)
            continue;
        else
            oHttpPostedFile.SaveAs(Server.MapPath("Files") + "\\" + System.
IO.Path.GetFileName(oHttpPostedFile.FileName));
    }
}

```

```
Response.Write("<script language='javascript'>window.returnValue =  
'" + strFileName + "';window.close();</script>");
```

(2) 完成文件上传后, 利用 ArcObjects 功能来实现将上传文件加载到数据服务中。
首先需要获得服务器上下文:

```
//获得服务器上下文
private ESRI.ArcGIS.Server.IServerContext GetServerContext()
{
    string serverName = "localhost"; //服务器机器名称
    string mapServiceName = "china"; //空间数据服名称
    ESRI.ArcGIS.Server.IServerObjectManager serverObjectManager;

    // 获得 SOM, 并放入 Session 变量中
    if (Session["SOM"] == null)
    {
        // 用 ADF connection 库
        ESRI.ArcGIS.ADF.Connection.AGS.AGSServerConnection agsServerConnection =
            new ESRI.ArcGIS.ADF.Connection.AGS.AGSServerConnection();
        agsServerConnection.Host = serverName;
        agsServerConnection.Connect();

        serverObjectManager = agsServerConnection.ServerObjectManager;
        Session["SOM"] = serverObjectManager;
    }
    else
    {
        serverObjectManager = Session["SOM"] as ESRI.ArcGIS.Server.IServer
ObjectManager;
    }
    //根据服务名来创建上下文
    ESRI.ArcGIS.Server.IServerContext serverContext =
        serverObjectManager.CreateServerContext(mapServiceName, "MapServer");

    return serverContext;
}
```

加载 shp 文件:

```
//添加 Shp 图层
private void AddLayer(ESRI.ArcGIS.Server.IServerContext serverContext, string
fileName)
{
    // 得到地图服务下的 ArcObjects map 对象
    ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
serverContext.ServerObject;
    ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.ArcGIS.
Carto.IMapServerObjects2)mapServer;
    string mapName = mapServer.DefaultMapName;
    ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);
```

```

string filePath = Server.MapPath("Files");//得到上传文件的路径

//根据 shapefile 文件路径, 建立 ArcObjects FeatureWorkspace
ESRI.ArcGIS.Geodatabase.IWorkspaceFactory workspaceFactory =

(ESRI.ArcGIS.Geodatabase.IWorkspaceFactory)serverContext.CreateObject("esriDataSou
rcesFile.ShapefileWorkspaceFactory");
    ESRI.ArcGIS.Geodatabase.IWorkspace workspace = workspaceFactory.OpenFrom
File(filePath, 0);
    ESRI.ArcGIS.Geodatabase.IFeatureWorkspace featureWorkspace = (ESRI.ArcGIS.
Geodatabase.IFeatureWorkspace)workspace;

//把 shapefile 变成 GeoFeatureLayer
    ESRI.ArcGIS.Carto.IFeatureLayer aoFeatureLayer = (ESRI.ArcGIS.Carto.
IFeatureLayer)serverContext.CreateObject("esriCarto.FeatureLayer");
//也可以用 guid 来创建 FeatureLayer
//IFeatureLayer layer = (IFeatureLayer)in_mapcontext.CreateObject
("{E663A651-8AAD-11D0-BEC7-00805F7C4268}");
    aoFeatureLayer.FeatureClass = featureWorkspace.OpenFeatureClass(fileName);
    aoFeatureLayer.Name = fileName;
    ESRI.ArcGIS.Carto.IGeoFeatureLayer geoFeatureLayer = (ESRI.ArcGIS.Carto.
IGeoFeatureLayer)aoFeatureLayer;

// 创建渲染的颜色
    ESRI.ArcGIS.Display.IRgbColor rgbColor = (ESRI.ArcGIS.Display.IRgbColor)
serverContext.CreateObject("esriDisplay.RgbColor");
    rgbColor.Red = 0;
    rgbColor.Green = 0;
    rgbColor.Blue = 210;

//根据图层的类型来渲染图层
    ESRI.ArcGIS.Carto.ISimpleRenderer aoSimpleRenderer = (ESRI.ArcGIS.Carto.
ISimple Renderer)geoFeatureLayer.Renderer;
    ESRI.ArcGIS.Geometry.esriGeometryType geometryType = aoFeatureLayer.
FeatureClass.ShapeType;
    if (geometryType == ESRI.ArcGIS.Geometry.esriGeometryType.esriGeometry
Point)
    {
        ESRI.ArcGIS.Display.ISimpleMarkerSymbol simpleMarkerSymbol =
            (ESRI.ArcGIS.Display.ISimpleMarkerSymbol)aoSimpleRenderer.Symbol;
        simpleMarkerSymbol.Color = (ESRI.ArcGIS.Display.IColor)rgbColor;
    }
    else if (geometryType == ESRI.ArcGIS.Geometry.esriGeometryType.esri
GeometryPolyline)
    {
        ESRI.ArcGIS.Display.ISimpleLineSymbol simpleLineSymbol = (ESRI.ArcGIS.
Display.ISimpleLineSymbol)aoSimpleRenderer.Symbol;
        simpleLineSymbol.Color = (ESRI.ArcGIS.Display.IColor)rgbColor;
    }

```



```

else if (geometryType == ESRI.ArcGIS.Geometry.esriGeometryType.esri
GeometryPolygon)
{
    ESRI.ArcGIS.Display.ISimpleFillSymbol simpleFillSymbol = (ESRI.ArcGIS.
Display.ISimpleFillSymbol)aoSimpleRenderer.Symbol;
    simpleFillSymbol.Color = (ESRI.ArcGIS.Display.IColor)rgbColor;
}
else
{
    throw new System.Exception("No renderer or symbol selected. Shape type
undetermined.");
}

//把图层添加到 ArcObjects Map 对象
aoMap.AddLayer(aoFeatureLayer);
mapServerObjects.RefreshServerObjects();
//serverContext.ReleaseContext();
}

```

刷新客户端:

```

string strFileName = eventArgument;
//处理动态添加图层
IServerContext pSC= GetServerContext();
AddLayer(pSC, strFileName+".shp");
pSC.ReleaseContext();
//刷新客户端
Toc1.Refresh();
Map1.CallbackResults.CopyFrom(Toc1.CallbackResults);
\
Map1.RefreshResource("MapResourceItem0");

m_callbackInvocation = Map1.CallbackResults.ToString();

```

(3) 单击图 8.14 中的“加载”按钮, 将 Shape 文件加载到地图上, 如图 8.15 所示。



图 8.15 动态添加图层

动态添加图层功能很有用, 可以让 WebGIS 拥有桌面 GIS 的功能, 但是文件上传和调用服务

器端 ArcObjects 实现文件加载以及异步刷新, 却比较麻烦。例如, MapInfo 格式实现起来就很困难。

8.6 地图导出

一般的 WebGIS 应用程序提供将地图输出为图片的功能, 该功能在 ArcGIS Server 中也能实现类似的功能, 本节将介绍如何实现地图的导出, 步骤如下:

(1) 编写一个 UserControl, 里面放置一个 asp:Image 控件, 控件的脚本程序如下:

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="OutImage.ascx.cs"
Inherits="OutImage" %>
<table>
<tr>
<td>
<asp:UpdatePanel ID="UpdatePanell" runat="server">
<ContentTemplate>

<asp:Image ID="MapImage" runat="server" Height="385px" Width="485px" BorderStyle=
"Solid" BorderWidth="1px" />
</ContentTemplate>
</asp:UpdatePanel>
</td>
</tr>
</table>
```

(2) 在页面上添加 FloatingPanel 控件和前面生成的 UserControl 控件, 在应用程序运行的时候, 先隐藏 FloatingPanel 控件, 脚本程序如下。

```
<esri:FloatingPanel ID="FloatingPanell" runat="server" BackColor="White"
BorderColor="Gray" BorderStyle="Solid" BorderWidth="1px"
Font-Names="Verdana"
Font-Size="8pt" ForeColor="Black" Height="405px" Title=
"导出地图"
TitleBarColor="WhiteSmoke" TitleBarHeight="20px" TitleBar
SeparatorLine="False"
Transparency="35" Width="500px" Visible="False"
onshown="FloatingPanell_Shown">
<uc5:OutImage ID="OutImage1" runat="server" />
</esri:FloatingPanel>
```

(3) 在 NavigateUrl 属性中编写 JavaScript 程序, 该程序主要用来显示 FloatingPanel 控件, 示例程序如下:

```
<script language="javascript" type="text/javascript">
function showImage()
{
var dia = $find("FloatingPanell");
if(dia)
{
dia.show();
dia._moveTo(300,100);
}
```

```

    }
}
</script>

```

(4) 在 ArcGIS Server 中, 把地图输出为图片, 示例代码如下。

```

//把地图导出为 JPG 图片格式
public void CreateMapImage(ESRI.ArcGIS.Server.IServerContext serverContext,
    ESRI.ArcGIS.Carto.IMapDescription aoMapDescription)
{
    ESRI.ArcGIS.Carto.IImageType imageType;
    ESRI.ArcGIS.Carto.IImageDescription imageDescription;
    ESRI.ArcGIS.Carto.IImageDisplay imageDisplay;

    imageType = serverContext.CreateObject("esriCarto.ImageType") as ESRI.
    ArcGIS.Carto.IImageType;
    imageDescription = serverContext.CreateObject("esriCarto.ImageDescription")
    as ESRI.ArcGIS.Carto.ImageDescription;
    imageDisplay = serverContext.CreateObject("esriCarto.ImageDisplay") as ESRI.
    ArcGIS.Carto.ImageDisplay;

    //设置
    imageType.Format = ESRI.ArcGIS.Carto.esriImageFormat.esriImageJPG;
    imageType.ReturnType = ESRI.ArcGIS.Carto.esriImageReturnType.esriImage
    ReturnURL;

    // 设置高度、宽度、比例尺
    imageDisplay.Height = (int)MapImage.Height.Value;
    imageDisplay.Width = (int)MapImage.Width.Value;
    imageDisplay.DeviceResolution = 86;

    imageDescription.Display = imageDisplay;
    imageDescription.Type = imageType;

    ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
    serverContext.ServerObject;

    //生成图片并显示
    ESRI.ArcGIS.Carto.IImageResult mapImage = mapServer.ExportMapImage(aoMap
    Description, imageDescription);
    MapImage.ImageUrl = mapImage.URL;
    MapImage.Visible = true;
}

```

系统运行效果如图 8.16 所示。

也可以对图片上水印、加版权等, 可利用 ArcObjects 在图层中添加符号或注记来实现上述功能, 有兴趣的读者可以自己动手试验一下。源代码请参见本书附带光盘。

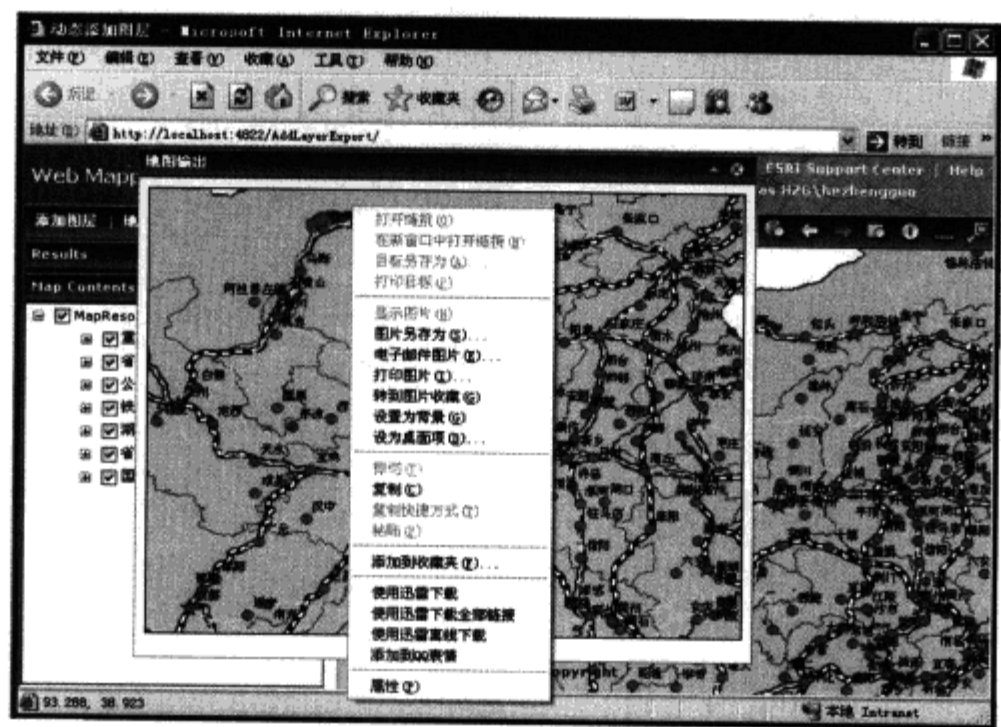


图 8.16 地图导出为图片

8.7 小结

本章讲述的例子都是基于 ArcGIS Server 模板开发的。图形选择、图层打开关闭、图层顺序调整、动态添加图层、地图导出等功能都是我们在开发过程中经常需要实现的功能，本章给出了最基本的实现方法。还有其他的方法，读者可以试着自己去实现。

第9章

ArcGIS Server专题图开发

专题图制图作为空间地理目标专题数据或属性数据的最有效的可视化工具，不仅能将 GIS 中的专题信息进行渲染和直观地可视化分析，而且能对地理信息进行加工，发掘隐藏在数据中的模式以及发展趋势，为人们获取某种或某些特定的信息提供一种有效的手段。它是使用各种图形样式（颜色或填充模式）图形化显示地图基础信息的一类地图。ArcGIS Server 中提供的除了自定义专题图 GraphicsLayer 外，还可以使用 ArcObjects 中的专题图：柱状图、饼状图、分级专题图、分类专题图等，下面将介绍 ArcGIS Server 中专题图的开发。

9.1 柱状图

柱状图一般是用柱状的图形来形象地表示某种统计值的多少。由于它很清楚地反映出某种指标的多少，因此在 GIS 的专题图中非常常见，下面以 ArcGIS Server 中柱状图的实现为例进行介绍。

ArcGIS Server 中把柱状图和饼状图都归类到图表专题图中。实现柱状图的步骤如下：

（1）首先需要找到画柱状化图层所对应的图层：

```
// 得到地图服务下的 ArcObjects map 对象
ESRI.ArcGIS.Server.IServerContext pServerContext = GetServerContext();

ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
pServerContext.ServerObject;
ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.ArcGIS.
Carto.IMapServerObjects2)mapServer;
string mapName = mapServer.DefaultMapName;
ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);

ESRI.ArcGIS.Carto.ILayer pLayer = aoMap.get_Layer(layerID);//得到图层
ESRI.ArcGIS.Carto.IGeoFeatureLayer pGeoLayer = pLayer as IGeoFeatureLayer;
```

(2) 设置柱状图的属性字段:

```
//设置专题图元素的属性名称列表
    IChartRenderer pChartRender = pServerContext.CreateObject("esriCarto.
ChartRenderer") as IChartRenderer;
    IRendererFields pRenderFields = pChartRender as IRendererFields;
    foreach (string var in fields)
    {
        pRenderFields.AddField(var, var);
    }
```

(3) 实例化图表对象:

```
//实例化图表对象并取得元素指定属性的最大值
    IBarChartSymbol pBarChartSymbol = pServerContext.CreateObject("esri
Display.BarChartSymbol") as IBarChartSymbol;
    IChartSymbol pChartSymbol = pBarChartSymbol as IChartSymbol;
    pChartSymbol.MaxValue = GetStaMaxMin(fields, pGeoLayer)[0];
    pBarChartSymbol.Width = 8;
    IMarkerSymbol pMarkerSymbol = pBarChartSymbol as IMarkerSymbol;
    pMarkerSymbol.Size = 50;
```

(4) 设置柱状图的填充效果:

```
//设置柱状图每列填充效果
    ISymbolArray pSymbolArray = pBarChartSymbol as ISymbolArray;
    Random ranColor = new Random();
    for (int i = 0; i < fields.Length; i++)
    {
        IFillSymbol pFillSymbol = pServerContext.CreateObject("esriDisplay.
SimpleFillSymbol") as IFillSymbol;
        pFillSymbol.Color = GetRGB(ranColor.Next(255), ranColor.Next(255),
ranColor.Next(255), pServerContext);
        pSymbolArray.AddSymbol((ISymbol)pFillSymbol);
    }
```

(5) 应用柱状专题到指定图层:

```
//设置地图图层背景
    ESRI.ArcGIS.Display.ISimpleFillSymbol pFSymbol = pServerContext.Create
Object("esriDisplay.SimpleFillSymbol") as ESRI.ArcGIS.Display.SimpleFillSymbol;
    pFSymbol.Color = GetRGB(239, 228, 249, pServerContext);
    pChartRender.BaseSymbol = pFSymbol as ISymbol;

    //应用柱状专题到指定图层
    pChartRender.ChartSymbol = pBarChartSymbol as IChartSymbol;
    pChartRender.Label = "Test";
    pChartRender.UseOverposter = false;
    pChartRender.CreateLegend();
    pGeoLayer.Renderer = pChartRender as IFeatureRenderer;
```

(6) 刷新地图:

```
//刷新地图显示图表及图例
mapServerObjects.RefreshServerObjects();
Map1.RefreshResource("MapResourceItem0");
Toc1.BuddyControl = "Map1";
Toc1.Refresh();
Map1.Refresh();
pServerContext.ReleaseContext();
```

应用程序初始界面如图 9.1 所示。然后，我们把每个省的面积用柱状图来表示，如图 9.2 所示。通过柱状图，用户可以清楚地看出哪个省的面积大，哪个省的面积小。

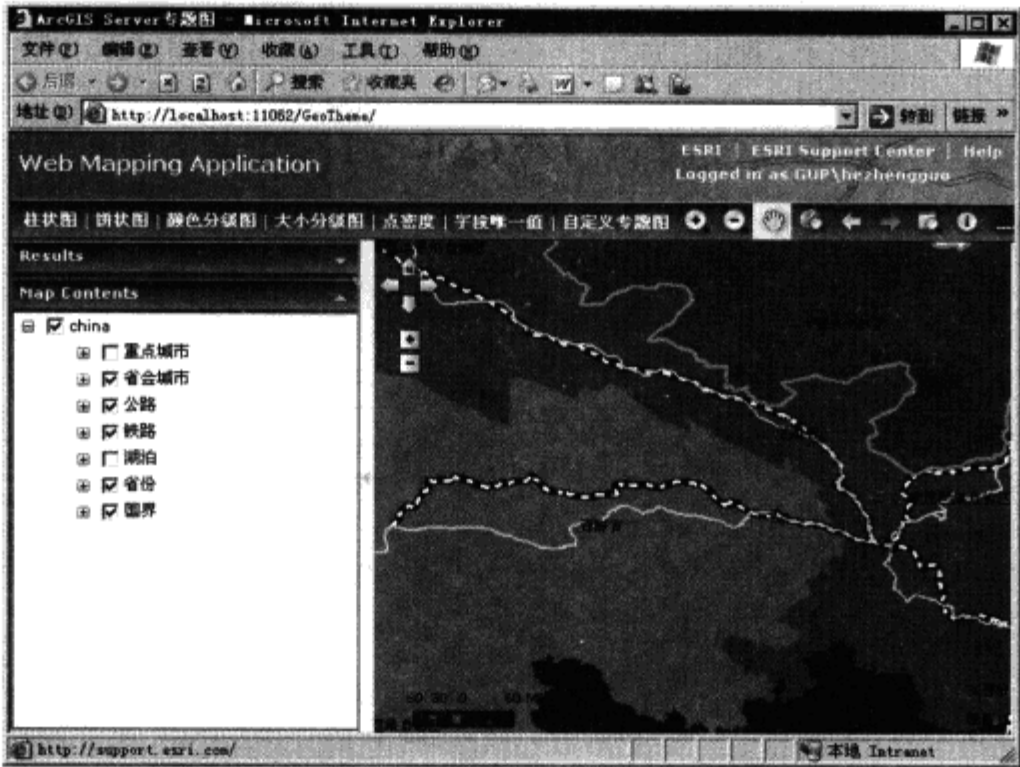


图 9.1 应用程序初始界面

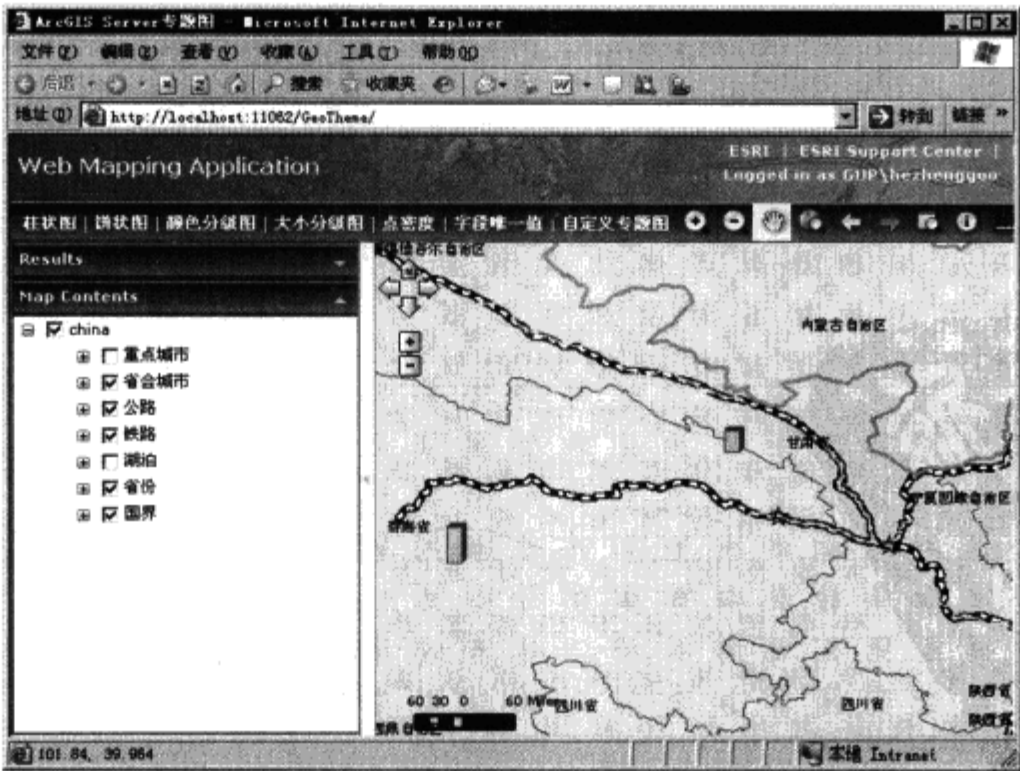


图 9.2 各省面积柱状图

9.2 饼状图

饼状图与柱状图相似，只是表现形式不同而已：一个是柱状图形，一个是饼状图型，它们都是用来表示某种指标的多少或大小。饼状图的实现步骤如下：

(1) 首先需要找到要饼状化图层所对应的图层：

```
// 得到地图服务下的 ArcObjects map 对象
ESRI.ArcGIS.Server.IServerContext pServerContext = GetServerContext();

ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
pServerContext.ServerObject;
ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.ArcGIS.
Carto.IMapServerObjects2)mapServer;
string mapName = mapServer.DefaultMapName;
ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);

ESRI.ArcGIS.Carto.ILayer pLayer = aoMap.get_Layer(layerID); //得到图层
ESRI.ArcGIS.Carto.IGeoFeatureLayer pGeoLayer = pLayer as IGeoFeatureLayer;
```

(2) 设置饼状图的属性字段：

```
//设置专题图元素的属性名称列表
IChartRenderer pChartRender = pServerContext.CreateObject("esriCarto.
ChartRenderer") as IChartRenderer;
IRendererFields pRenderFields = pChartRender as IRendererFields;
foreach (string var in fields)
{
    pRenderFields.AddField(var, var);
}
```

(3) 实例化图表对象：

```
ESRI.ArcGIS.Display.IPieChartSymbol pPieSym = pServerContext.CreateObject("esri
Display.PieChartSymbol") as ESRI.ArcGIS.Display.IPieChartSymbol;
ESRI.ArcGIS.Display.IChartSymbol pCharSym = pPieSym as ESRI.ArcGIS.Display.
IChartSymbol;
pPieSym.Clockwise = true;
pPieSym.UseOutline = true;

pCharSym.MaxValue = GetStaMaxMin(fields, pGeoLayer)[0];
```

(4) 设置饼状图的填充效果：

```
//设置饼图外围线
ISimpleLineSymbol pOutLine = pServerContext.CreateObject("esriDisplay.
SimpleLineSymbol") as ISimpleLineSymbol;
pOutLine.Color = GetRGB(255, 0, 128, pServerContext);
pOutLine.Style = ESRI.ArcGIS.Display.esriSimpleLineStyle.esriSLSSolid;
pOutLine.Width = 1;

pPieSym.Outline = pOutLine;
```

```

IMarkerSymbol pMarkSym = pPieSym as IMarkerSymbol;
pMarkSym.Size = 5;

//设置饼状图填充效果
ESRI.ArcGIS.Display.ISymbolArray pSymArr = pPieSym as ISymbolArray;
ISimpleFillSymbol pSimFillSym = pServerContext.CreateObject("esriDisplay.
SimpleFillSymbol") as ESRI.ArcGIS.Display.SimpleFillSymbol;
pSimFillSym.Color = GetRGB(128, 128, 128, pServerContext);
pSimFillSym.Outline = pOutline;

Random randColor = new Random();
for (int i = 0; i < fields.Length; i++)
{
    IFillSymbol pFillSym = pServerContext.CreateObject("esriDisplay.Simple
FillSymbol") as IFillSymbol;
    pFillSym.Color = GetRGB(randColor.Next(255), randColor.Next(255),
randColor.Next(255), pServerContext);
    pSymArr.AddSymbol((ISymbol)pFillSym);
}

//设置地图图层背景
pSimFillSym = pServerContext.CreateObject("esriDisplay.SimpleFillSymbol")
as ESRI.ArcGIS.Display.SimpleFillSymbol;
pSimFillSym.Color = GetRGB(255, 128, 255, pServerContext);
pCharRenderer.BaseSymbol = pSimFillSym as ISymbol;

//设置饼状图表属性
IPieChartRenderer pPieChartRenderer = pCharRenderer as IPieChartRenderer;
pPieChartRenderer.MinValue = 0.1;
pPieChartRenderer.MinSize = 1;
pPieChartRenderer.FlanneryCompensation = false;
pPieChartRenderer.ProportionalBySum = true;
pPieChartRenderer.ProportionalField = fields[0];
pCharRenderer.ChartSymbol = pPieSym as IChartSymbol;
pCharRenderer.Label = "面积";

```

(5) 应用饼状专题到指定图层:

```

//应用饼状专题到指定图层
pCharRenderer.UseOverposter = false;
pCharRenderer.CreateLegend();
pGeoLayer.Renderer = pCharRenderer as IFeatureRenderer;

```

(6) 刷新地图:

```

//刷新地图显示图表及图例
mapServerObjects.RefreshServerObjects();
Map1.RefreshResource("MapResourceItem0");
Toc1.BuddyControl = "Map1";
Toc1.Refresh();

```

```
Map1.Refresh();
pServerContext.ReleaseContext();
```

应用程序初始界面如图 9.3 所示。然后，我们把每个省的面积用饼状图来表示如图 9.4 所示。通过饼状图，用户可以清楚地看出各省的面积对比。

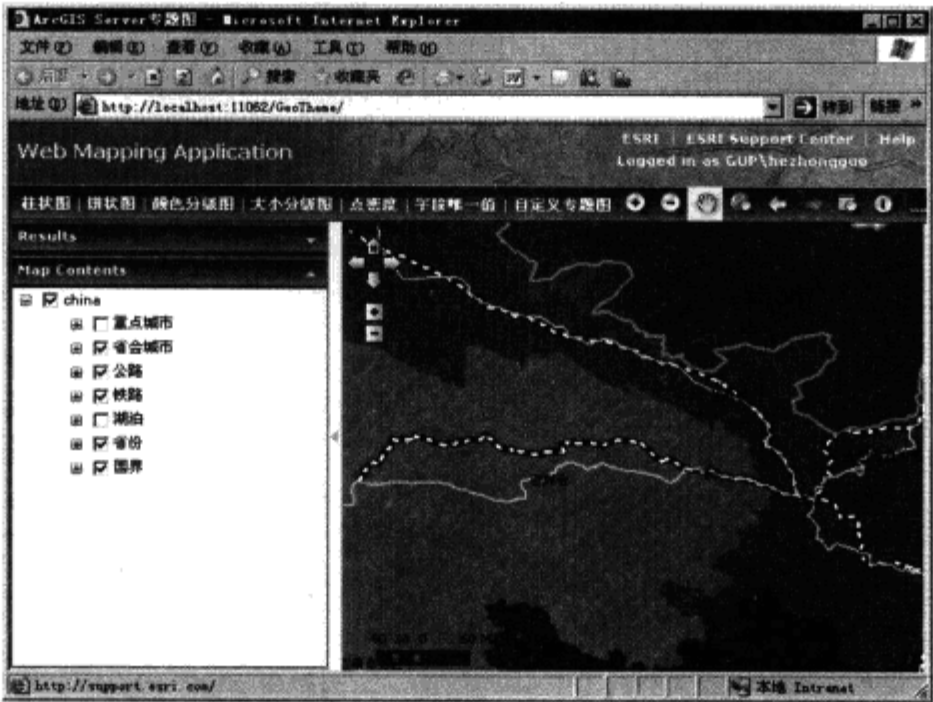


图 9.3 应用程序初始界面

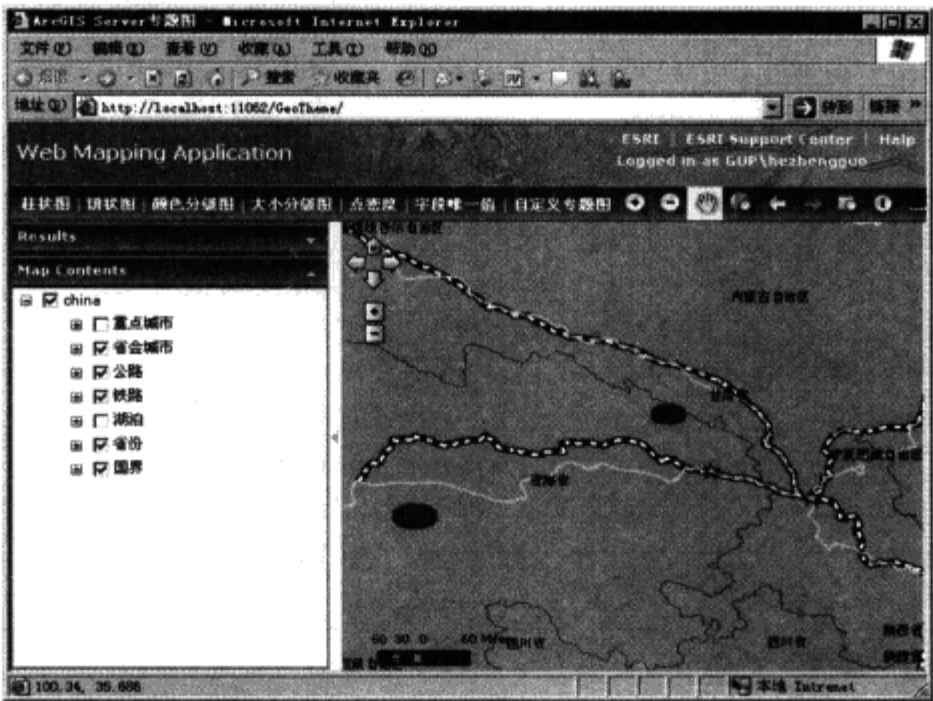


图 9.4 各省面积饼状图

采用饼状图或柱状图来表示专题图，用户可根据自己的喜好而定。

9.3 分级专题图

人们通常用级别来表示不同程度，如世卫组织发布猪流感警告级别有 1 级、2 级、3 级、4 级、5 级。同样地，我们也可以将空间几何对象的某个属性值进行分级以便来区分几何对象，并以不同的符号来表示。这就是所谓的分类分级专题图。

9.3.1 Graduated colors

Graduated colors 就是以不同的颜色来表示某一属性值在某一范围内的专题图。实现步骤如下:

(1) 首先需要找到要颜色分级化图层所对应的图层:

```
// 得到地图服务下的 ArcObjects map 对象
ESRI.ArcGIS.Server.IServerContext pServerContext = GetServerContext();

ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
pServerContext.ServerObject;
ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.ArcGIS.
Carto.IMapServerObjects2)mapServer;
string mapName = mapServer.DefaultMapName;
ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);

ESRI.ArcGIS.Carto.ILayer pLayer = aoMap.get_Layer(layerID); //得到图层
ESRI.ArcGIS.Carto.IGeoFeatureLayer pGeoLayer = pLayer as IGeoFeatureLayer;
```

(2) 设置分级的属性字段:

```
ESRI.ArcGIS.Carto.IClassBreaksRenderer pClassBreaksRenderer = pServerContext.
CreateObject("esriCarto.ClassBreaksRenderer") as IClassBreaksRenderer;
ESRI.ArcGIS.esriSystem.QuantileClass pQuanClass = new ESRI.ArcGIS.
esriSystem.QuantileClass();

ITable pTable = pGeoLayer as ITable;
long iCount = pTable.RowCount(null);
string strFields="";
int iFieldIndex = -1;
for (int i = 0; i < fields.Length; i++)
{
    iFieldIndex = pTable.Fields.FindField(fields[i]);
    if (iFieldIndex > -1)
    {
        strFields +=fields[i]+",";
    }
}

if (iFieldIndex < 0) return;
strFields = strFields.Substring(0,strFields.Length-1);
ITableSort pTableSort = pServerContext.CreateObject("esriGeodatabase.
TableSort") as ITableSort;
pTableSort.Fields = strFields;
pTableSort.set_Ascending(strFields, true);
pTableSort.QueryFilter = null;
pTableSort.Table = pTable;
pTableSort.Sort(null);
```

(3) 设置分级的起始颜色和分级数目:

```

IRgbColor pColorFrom = (IRgbColor)GetRGB(255, 0, 0, pServerContext);
IRgbColor pColorTo = (IRgbColor)GetRGB(255, 0, 255, pServerContext);

//设置起始颜色和分级
IAlgorithmicColorRamp pColorRamp = pServerContext.CreateObject ("esri
Display.AlgorithmicColorRamp") as IAlgorithmicColorRamp;
pColorRamp.FromColor = pColorFrom;
pColorRamp.ToColor = pColorTo;
pColorRamp.Size = 10;

bool bResult = false;
pColorRamp.CreateRamp(out bResult);

```

(4) 设置不同级别的值和符号:

```

pClassBreaksRenderer.Field = fields[0];
pClassBreaksRenderer.BreakCount = 10;
IDataSampling pDataSample = pClassBreaksRenderer as IDataSampling;
pDataSample.MaxSampleSize = 10000;
IClassBreaksUIProperties pUIProperties = pClassBreaksRenderer as IClass
BreaksUIProperties;
pUIProperties.ColorRamp = "自定义";
double dMax = GetStaMaxMin(fields,pGeoLayer)[0];
ISimpleFillSymbol pSimFillSym = pServerContext.CreateObject("esriDisplay.
SimpleFillSymbol") as ESRI.ArcGIS.Display.SimpleFillSymbol;
ISimpleLineSymbol pOutLine = pServerContext.CreateObject("esriDisplay.
SimpleLineSymbol") as ISimpleLineSymbol;
pOutLine.Color = GetRGB(255, 0, 128, pServerContext);
pOutLine.Style = ESRI.ArcGIS.Display.esriSimpleLineStyle.esriSLSSolid;
pOutLine.Width = 1;
pSimFillSym.Outline = pOutLine;

IEnumColors pEnumColor = pColorRamp.Colors as IEnumColors;
pEnumColor.Reset();
for (int j = 0; j < 10; j++)
{
    pSimFillSym.Color = pEnumColor.Next();
    pClassBreaksRenderer.set_Break(j, (j + 1) * dMax / 10);
    pUIProperties.set_LowBreak(j, (j + 1) * dMax / 10);
    pClassBreaksRenderer.set_Label(j, "第" + (j + 1).ToString() + "级");
    pClassBreaksRenderer.set_Symbol(j, pSimFillSym as ISymbol);
}

```

(5) 应用分级专题到指定图层:

```

pGeoLayer.Renderer = pClassBreaksRenderer as IFeatureRenderer;

```

(6) 刷新地图:

```

//刷新地图显示图表及图例

```

```

mapServerObjects.RefreshServerObjects();
Map1.RefreshResource("MapResourceItem0");
Toc1.BuddyControl = "Map1";
Toc1.Refresh();
Map1.Refresh();
pServerContext.ReleaseContext();

```

颜色分级前后的运行界面如图 9.5 和图 9.6 所示。

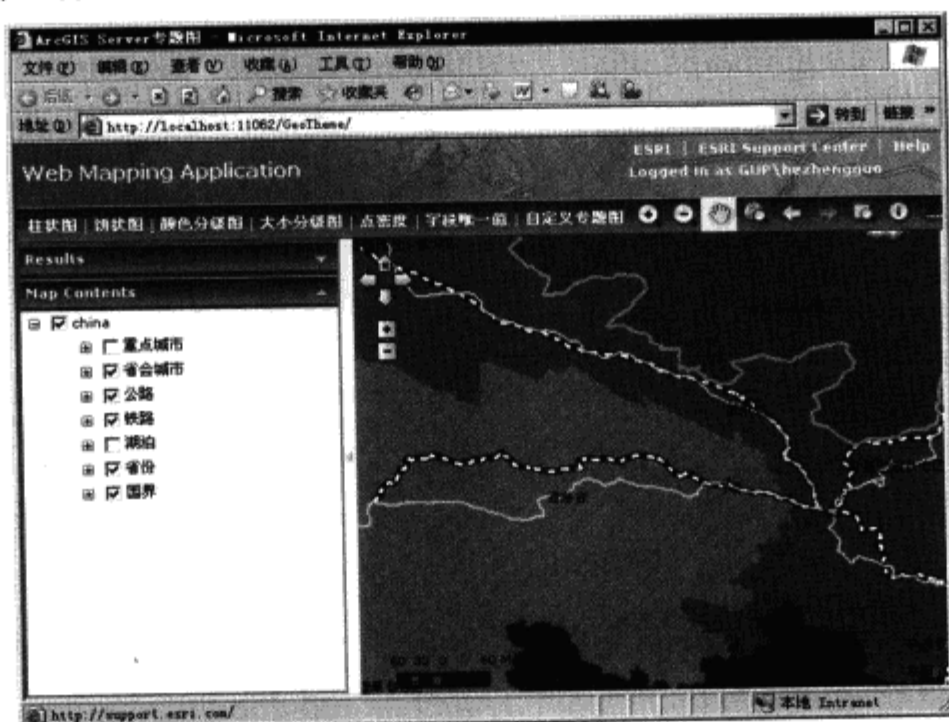


图 9.5 初始界面

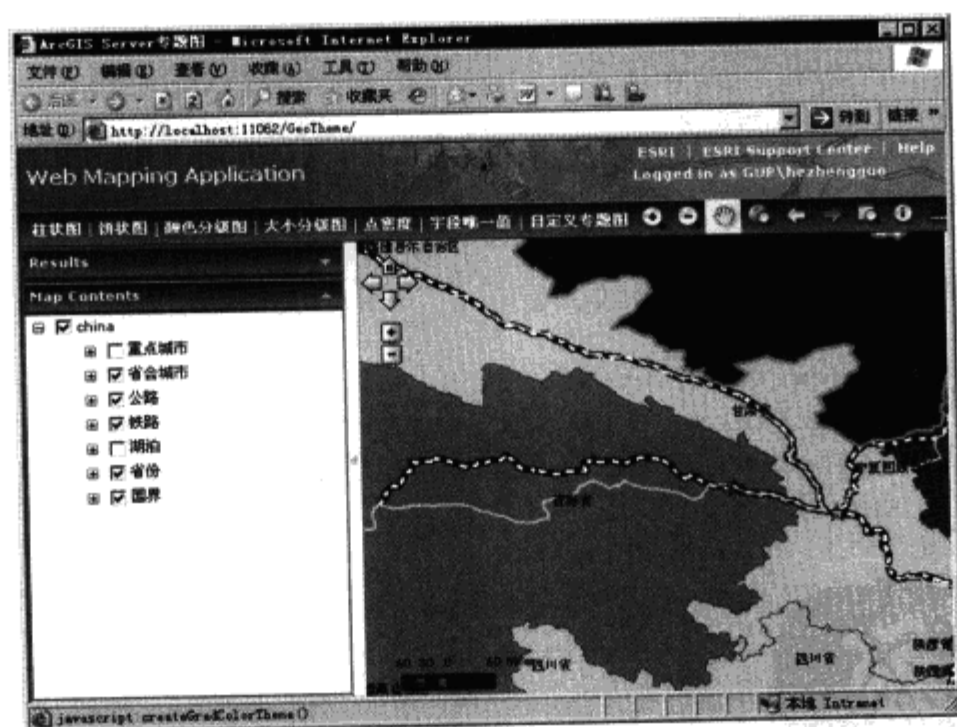


图 9.6 颜色分级界面

9.3.2 Graduated symbols

Graduated symbols 与 Graduated colors 相似, Graduated colors 是以不同的颜色来表示不同的级别, 而 Graduated symbols 是以符号的大小来表示不同的级别。实现步骤基本相同。

(1) 首先需要找到要符号大小化图层所对应的图层:

```
// 得到地图服务下的 ArcObjects map 对象
ESRI.ArcGIS.Server.IServerContext pServerContext = GetServerContext();

ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
pServerContext.ServerObject;
ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.ArcGIS.
Carto.IMapServerObjects2)mapServer;
string mapName = mapServer.DefaultMapName;
ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);

ESRI.ArcGIS.Carto.ILayer pLayer = aoMap.get_Layer(layerID); //得到图层
ESRI.ArcGIS.Carto.IGeoFeatureLayer pGeoLayer = pLayer as IGeoFeatureLayer;
```

(2) 设置分级的属性字段:

```
ESRI.ArcGIS.Carto.IClassBreaksRenderer pClassBreaksRenderer = pServerContext.
CreateObject("esriCarto.ClassBreaksRenderer") as IClassBreaksRenderer;
ESRI.ArcGIS.esriSystem.QuantileClass pQuanClass = new ESRI.ArcGIS.
esriSystem.QuantileClass();

ITable pTable = pGeoLayer as ITable;
long iCount = pTable.RowCount(null);
string strFields="";
int iFieldIndex = -1;
for (int i = 0; i < fileds.Length; i++)
{
    iFieldIndex = pTable.Fields.FindField(fileds[i]);
    if (iFieldIndex > -1)
    {
        strFields +=fileds[i]+",";
    }
}

if (iFieldIndex < 0) return;
strFields = strFields.Substring(0,strFields.Length-1);
ITableSort pTableSort = pServerContext.CreateObject("esriGeodatabase.
TableSort") as ITableSort;
pTableSort.Fields = strFields;
pTableSort.set_Ascending(strFields, true);
pTableSort.QueryFilter = null;
pTableSort.Table = pTable;
pTableSort.Sort(null);
```

(3) 设置分级的字段和分级数目:

```
IRgbColor pColor = (IRgbColor)GetRGB(255, 128, 128, pServerContext);

pClassBreaksRenderer.Field = fileds[0];
pClassBreaksRenderer.BreakCount = 10;
IDataSampling pDataSample = pClassBreaksRenderer as IDataSampling;
pDataSample.MaxSampleSize = 10000;
```

```

IClassBreaksUIProperties pUIProperties = pClassBreaksRenderer as IClass
BreaksUIProperties;
pUIProperties.ColorRamp = "自定义";

```

(4) 设置不同级别的值和符号大小:

```

ISimpleMarkerSymbol pMarkerSym = pServerContext.CreateObject("esriDisplay.
SimpleMarkerSymbol") as ESRI.ArcGIS.Display.ISimpleMarkerSymbol;
pMarkerSym.OutlineColor = GetRGB(255, 0, 0, pServerContext);
pMarkerSym.Style = ESRI.ArcGIS.Display.esriSimpleMarkerStyle.esriSMS
Circle;

pMarkerSym.Outline = true;
pMarkerSym.Color = GetRGB(0, 128, 128, pServerContext);
for (int j = 0; j < 10; j++)
{
    pMarkerSym.Size = 2 + j * 10;
    pClassBreaksRenderer.set_Break(j, (j + 1) * dMax / 10);
    pUIProperties.set_LowBreak(j, (j + 1) * dMax / 10);
    pClassBreaksRenderer.set_Label(j, "第" + (j + 1).ToString() + "级");
    pClassBreaksRenderer.set_Symbol(j, pMarkerSym as ISymbol);
}

```

(5) 应用分级专题到指定图层:

```
pGeoLayer.Renderer = pClassBreaksRenderer as IFeatureRenderer;
```

(6) 刷新地图:

//刷新地图显示图表及图例

```

mapServerObjects.RefreshServerObjects();
Map1.RefreshResource("MapResourceItem0");
Toc1.BuddyControl = "Map1";
Toc1.Refresh();
Map1.Refresh();
pServerContext.ReleaseContext();

```

符号分级前后的运行界面如图 9.7 和图 9.8 所示。

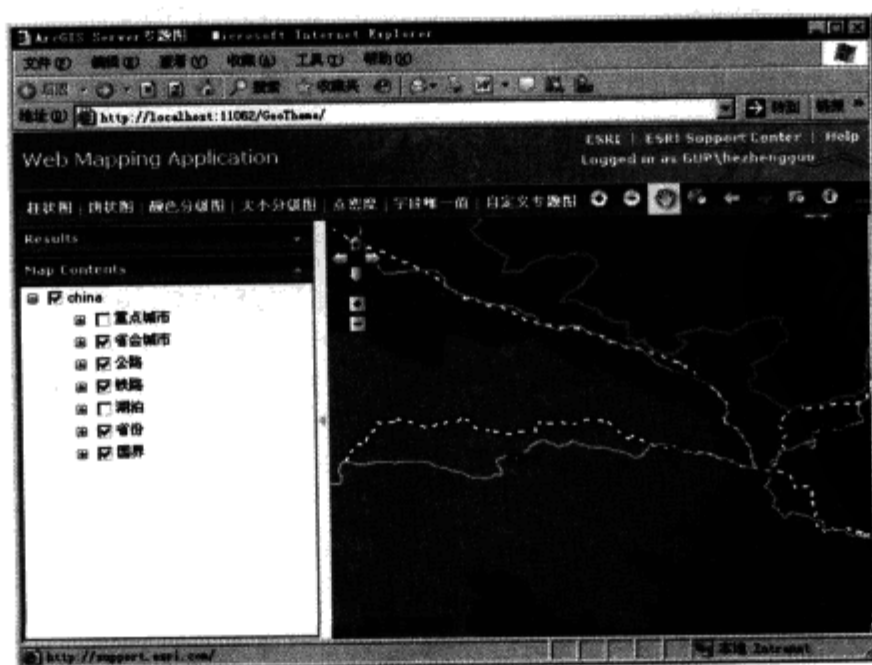


图 9.7 初始界面

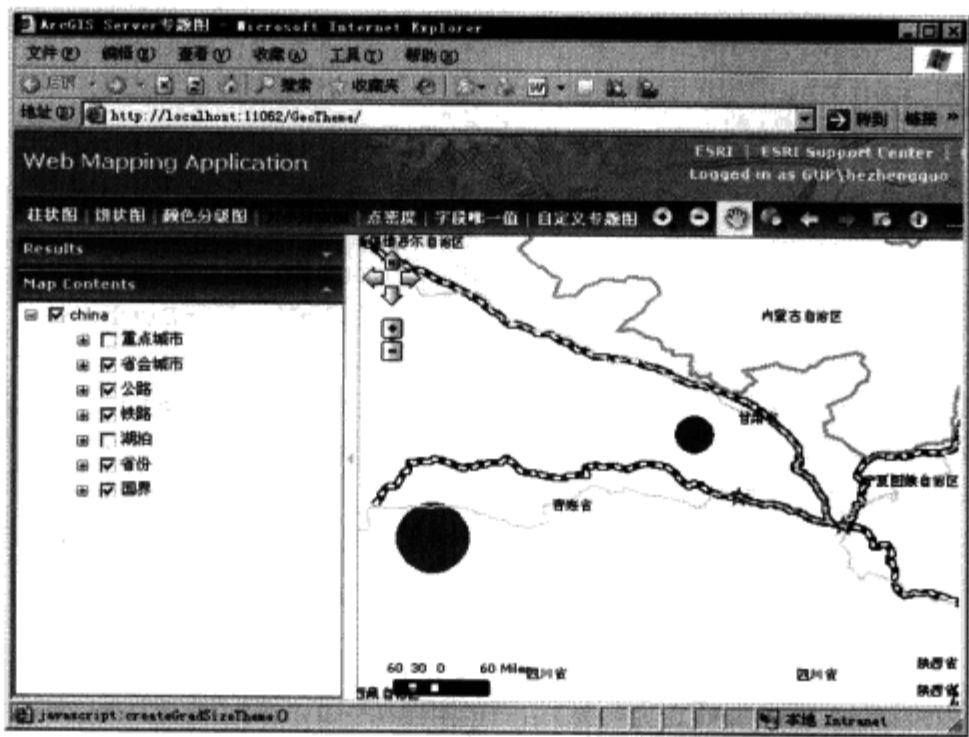


图 9.8 符号大小分级界面

9.3.3 Dot desity (点密度专题图)

Dot desity 就是所谓的点密度专题图。用点密度的大小来表示几何对象某一属性的大小。实现步骤如下。

(1) 首先需要找到要点密度化图层的所对应的图层:

```
// 得到地图服务下的 ArcObjects map 对象
ESRI.ArcGIS.Server.IServerContext pServerContext = GetServerContext();

ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
pServerContext.ServerObject;
ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.ArcGIS.
Carto.IMapServerObjects2)mapServer;
string mapName = mapServer.DefaultMapName;
ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);

ESRI.ArcGIS.Carto.ILayer pLayer = aoMap.get_Layer(layerID); //得到图层
ESRI.ArcGIS.Carto.IGeoFeatureLayer pGeoLayer = pLayer as IGeoFeatureLayer;
```

(2) 实例化点密度渲染工具:

```
ESRI.ArcGIS.Carto.IDotDensityRenderer pDotDensityRenderer = pServerContext.Create
Object("esriCarto.DotDensityRenderer") as IDotDensityRenderer;
IRendererFields pRenderFields = pDotDensityRenderer as IRendererFields;

for (int i = 0; i < fileds.Length; i++)
{
    pRenderFields.AddField(fileds[i], fileds[i]);
}

IDotDensityFillSymbol pDotDensityFillSym = pServerContext.CreateObject
```



```

("esriDisplay.DotDensityFillSymbol") as IDotDensityFillSymbol;
    pDotDensityFillSym.DotSize = 2;
    pDotDensityFillSym.Color = GetRGB(200, 200, 200, pServerContext);

    ISymbolArray pSymArray = pDotDensityFillSym as ISymbolArray;
    ISimpleMarkerSymbol pMarkerSym = pServerContext.CreateObject("esri Display.
SimpleMarkerSymbol") as ESRI.ArcGIS.Display.ISimpleMarkerSymbol;
    pMarkerSym.OutlineColor = GetRGB(255, 0, 0, pServerContext);
    pMarkerSym.Style = ESRI.ArcGIS.Display.esriSimpleMarkerStyle.esriSMSCircle;
    pMarkerSym.Outline = true;
    pMarkerSym.Color = GetRGB(0, 128, 128, pServerContext);

    pSymArray.AddSymbol((ISymbol)pMarkerSym);
    pDotDensityRenderer.DotDensitySymbol = pDotDensityFillSym;
    pDotDensityRenderer.DotValue = 1; //设置点的个数
    pDotDensityRenderer.CreateLegend(); //创建图例

```

(3) 应用点密度专题到指定图层:

```
pGeoLayer.Renderer = pDotDensityRenderer as IFeatureRenderer;
```

(4) 刷新地图:

//刷新地图显示图表及图例

```

mapServerObjects.RefreshServerObjects();
Map1.RefreshResource("MapResourceItem0");
Toc1.BuddyControl = "Map1";
Toc1.Refresh();
Map1.Refresh();
pServerContext.ReleaseContext();

```

点密度前后的运行界面如图 9.9 和图 9.10 所示。

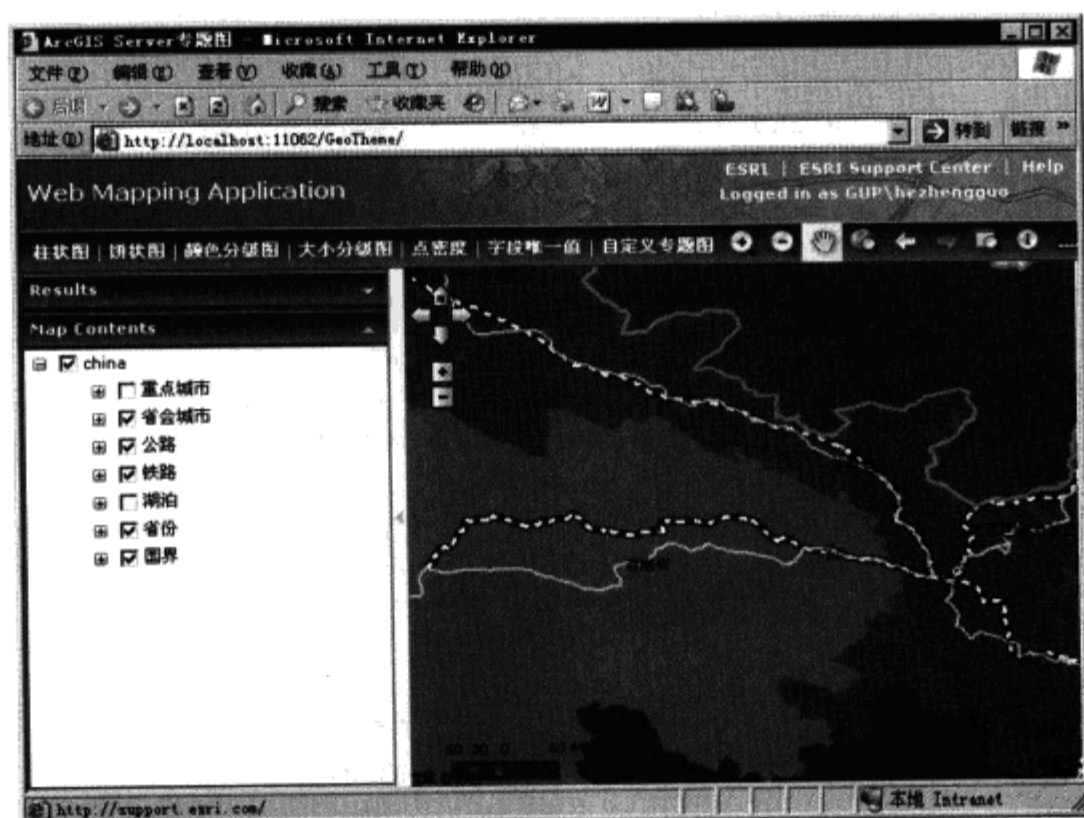


图 9.9 初始界面

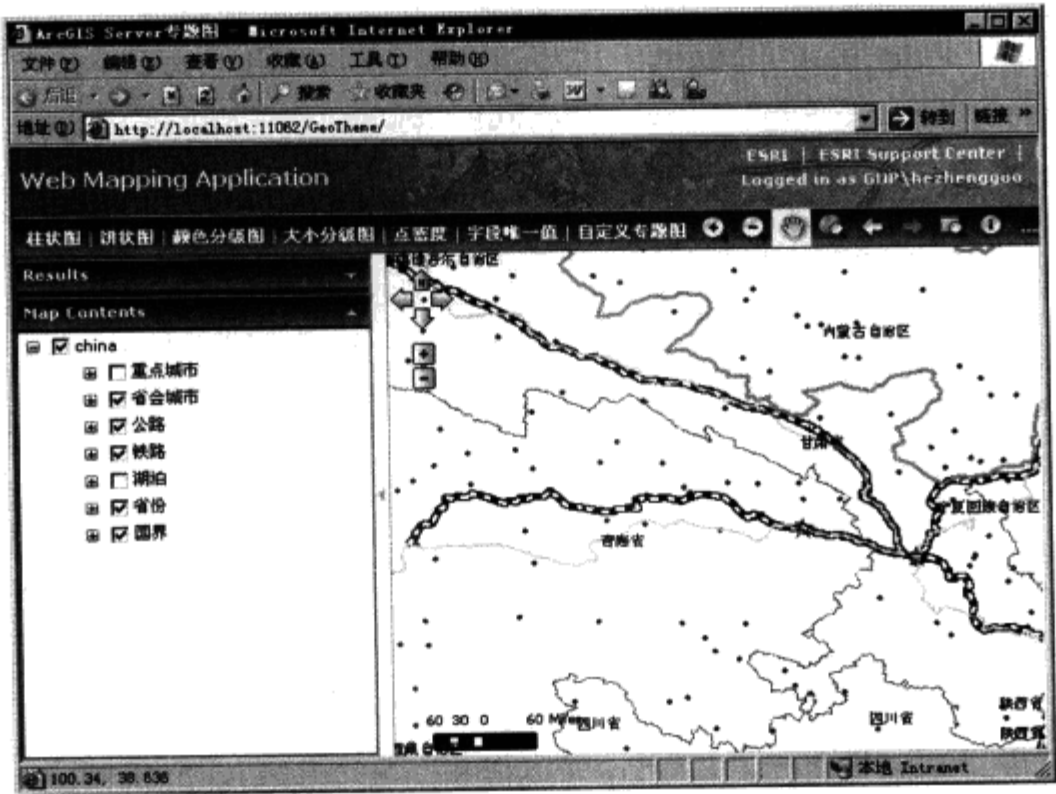


图 9.10 点密图界面

9.4 分类专题图

古人云：“物以类聚，人以群分”。就是说同类的东西常聚在一起，志同道合的人相聚成群，反之就分开。我们可以根据字段的属性来分类几何对象。

9.4.1 根据某个字段的唯一值

根据字段属性的唯一值来设置几何对象的符号，实现步骤如下。

(1) 找到唯一化图层所对应的图层：

```
// 得到地图服务下的 ArcObjects map 对象
ESRI.ArcGIS.Server.IServerContext pServerContext = GetServerContext();

ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
pServerContext.ServerObject;
ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.ArcGIS.
Carto.IMapServerObjects2)mapServer;
string mapName = mapServer.DefaultMapName;
ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);

ESRI.ArcGIS.Carto.ILayer pLayer = aoMap.get_Layer(layerID); //得到图层
ESRI.ArcGIS.Carto.IGeoFeatureLayer pGeoLayer = pLayer as IGeoFeatureLayer;
```

(2) 实例化唯一值渲染工具：

```
ESRI.ArcGIS.Carto.IUniqueValueRenderer pUniqueRender = pServerContext.Create
Object("esriCarto.UniqueValueRenderer") as IUniqueValueRenderer;

ISimpleFillSymbol pSimFillSym = pServerContext.CreateObject("esriDisplay.
```

```

SimpleFillSymbol") as ESRI.ArcGIS.Display.SimpleFillSymbol;
    ISimpleLineSymbol pOutline = pServerContext.CreateObject("esriDisplay.
SimpleLineSymbol") as ISimpleLineSymbol;
    pOutline.Color = GetRGB(255, 0, 128, pServerContext);
    pOutline.Style = ESRI.ArcGIS.Display.esriSimpleLineStyle.esriSLSSolid;
    pOutline.Width = 1;
    pSimFillSym.Outline = pOutline;

    pUniqueRender.FieldCount = fileds.Length;
    for (int i = 0; i < fileds.Length; i++)
    {
        pUniqueRender.set_Field(i, fileds[i]);
    }

    pUniqueRender.DefaultSymbol = pSimFillSym as ISymbol;
    pUniqueRender.UseDefaultSymbol = true;

```

(3) 得到字段的唯一属性值:

//得到单个字段的唯一值

```

private Hashtable GetUniqueValue(ITable pTable, string colName)
{
    Hashtable hTable = new Hashtable();
    ICursor pCursor = pTable.Search(null, false);
    IRow pRow = pCursor.NextRow();
    while (pRow != null)
    {
        int index = pTable.FindField(colName);
        if (index > -1)
        {
            object ob = pRow.get_Value(index);
            if (ob != null)
            {
                string str = ob.ToString();
                if (!hTable.Contains(str))
                {
                    hTable.Add(str, string.Empty);
                }
            }
            pRow = pCursor.NextRow();
        }
    }

    return hTable;
}

```

(4) 设置不同值的符号:

```

foreach (DictionaryEntry de in hashtable)
{
    int n = 0;

```

```

        ISimpleFillSymbol pFillSym = pServerContext.CreateObject("esriDisplay.
SimpleFillSymbol") as ESRI.ArcGIS.Display.SimpleFillSymbol;
        ISimpleLineSymbol pOutLine1 = pServerContext.CreateObject("esri Display.
SimpleLineSymbol") as ISimpleLineSymbol;
        //pOutLine1.Color = GetRGB(ran.Next(255), ran.Next(255), ran.Next(255),
pServerContext);
        pOutLine1.Style = ESRI.ArcGIS.Display.esriSimpleLineStyle.esriSLSSolid;
        pOutLine1.Width = 1;
        pFillSym.Outline = pOutLine1;
        pFillSym.Color = GetRGB(ran.Next(255), ran.Next(255), ran.Next(255),
pServerContext);
        //pUniqueRender.set_Value(n,de.Key.ToString());
        //pUniqueRender.set_Symbol(de.Key.ToString(), pFillSym as ISymbol);
        pUniqueRender.AddValue(de.Key.ToString(), de.Key.ToString(), pFillSym
as ISymbol);
        n++;
    }

```

(5) 应用唯一值专题到指定图层:

```
pGeoLayer.Renderer = pDotDensityRenderer as IFeatureRenderer;
```

(6) 刷新地图:

//刷新地图显示图表及图例

```

mapServerObjects.RefreshServerObjects();
Map1.RefreshResource("MapResourceItem0");
Toc1.BuddyControl = "Map1";
Toc1.Refresh();
Map1.Refresh();
pServerContext.ReleaseContext();

```

唯一值化前后的运行界面如图 9.11 和图 9.12 所示。

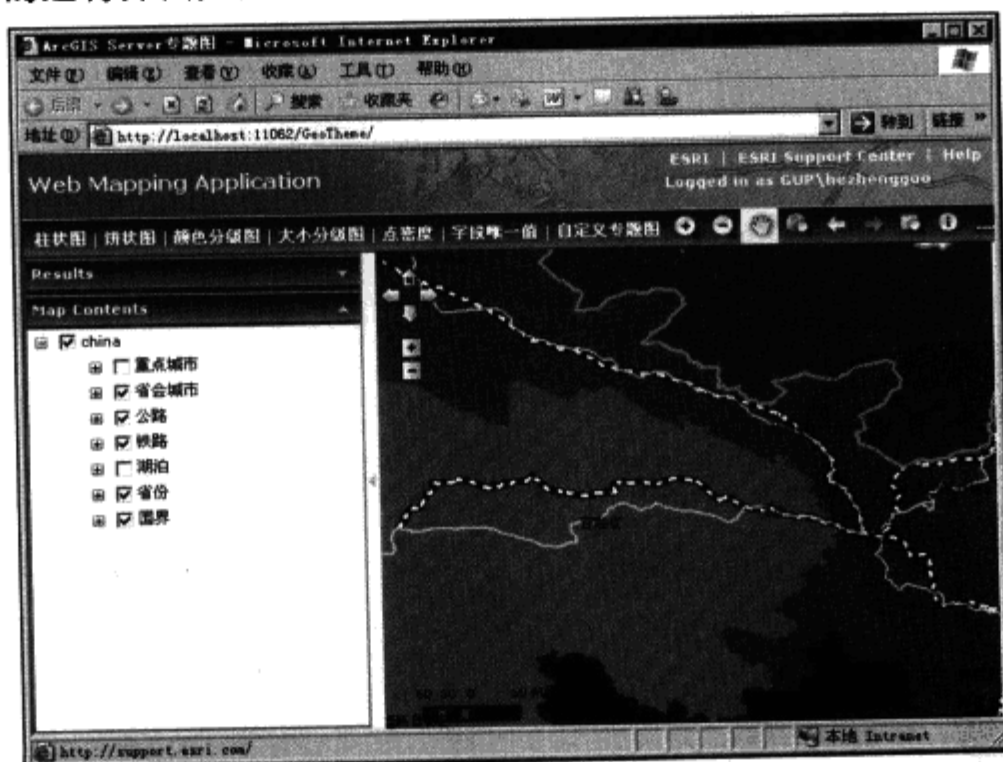


图 9.11 初始界面

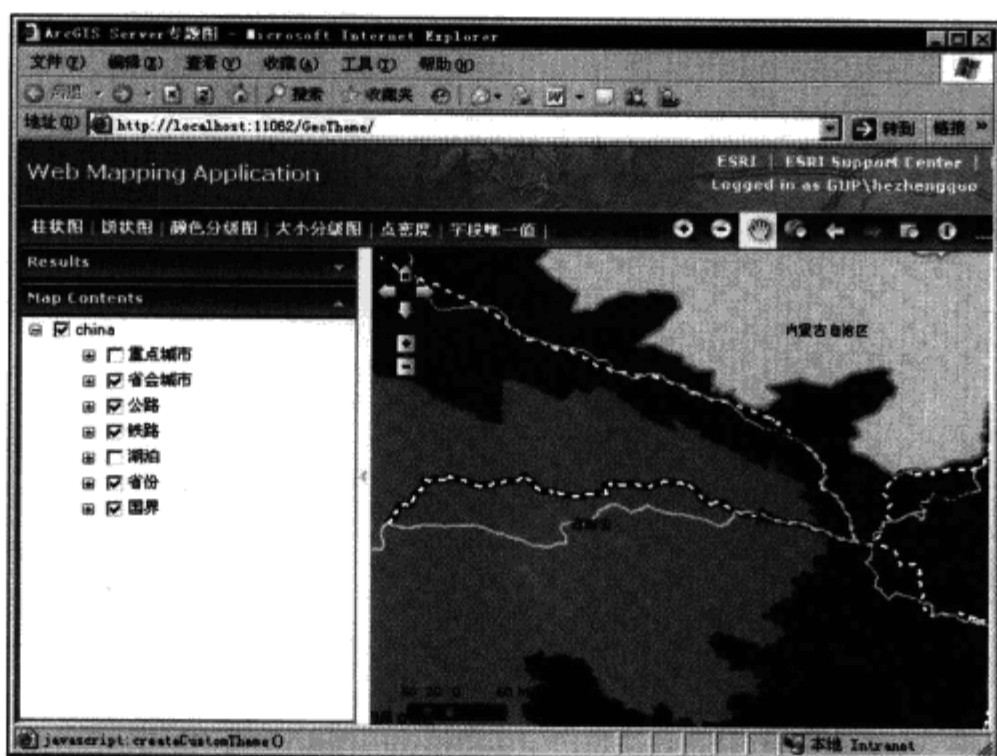


图 9.12 唯一值化后的界面

9.4.2 根据多个字段的值

多个字段属性唯一值与一个字段属性的唯一值实现步骤一样，唯一不同的是在第 3 步，取字段唯一值的时候，多个字段需要取多个组合后的唯一值。程序如下所示：

```
//得到多个字段的唯一值
private Hashtable GetUniqueValue(ITable pTable, string[] colName)
{
    Hashtable hTable = new Hashtable();
    ICursor pCursor = pTable.Search(null, false);
    IRow pRow = pCursor.NextRow();
    while (pRow != null)
    {
        //int index = pTable.FindField(colName);
        int[] index = new int[colName.Length];
        object[] ob = new object[colName.Length];
        for (int i = 0; i < colName.Length; i++)
        {
            index[i] = pTable.FindField(colName[i]);
            if (index[i] > -1)
            {
                ob[i] = pRow.get_Value(index[i]);
            }
        }

        string str = "";
        for(int j = 0;j<colName.Length;j++)
        {
            if (ob[j] != null)
            {

```

```

        str += ob[j].ToString();
    }
}

if (!hTable.Contains(str))
{
    hTable.Add(str, string.Empty);
}
pRow = pCursor.NextRow();
}

return hTable;
}

```

9.5 自定义专题图

8.3 节介绍过 GraphicsLayer，本节要介绍的自定义专题图就是要在 GraphicsLayer 中根据自己定义的颜色来画不同的元素。本例要实现的自定义专题的功能是列出省份的名称和颜色，用户可以自定义颜色，最后根据该自定义颜色来填充这些省份。实现的步骤如下：

(1) 定义 UserControl，用来列举省份名称和颜色。UserControl 的脚本程序如下：

```

<%@ Control Language="C#" AutoEventWireup="true" CodeFile="UserTheme.ascx.cs"
Inherits="UserTheme" %>
<script language="javascript" type="text/javascript">
function drawTheme()
{
    var
tableobj=document.getElementById("fpUserTheme_UserThemel_colorTable");//.rows.length);
    if(tableobj!=null)
    {

        var result="";
        for(var i=0;i<tableobj.rows.length;i++)
        {
            result+=tableobj.rows[i].cells[0].innerText+"=";
            result+=tableobj.rows[i].cells[1].style.backgroundColor+"&"
        }
        var message = result.substr(0,result.length-1);
        var context="Map1";
        <%=m_callbackInvocation%>
    }
}
</script>
<asp:UpdatePanel ID ="UpdatePanel" runat ="server" UpdateMode="Conditional">
<ContentTemplate>
<table>
<tr>
<td>
<div id="ColorList" style="width: 233px; text-align: center;">

```



```

        <asp:Table ID="colorTable" Width="100%" runat="server"
            oninit="colorTable_Init" >
        </asp:Table>
    </div>
</td>
<td>
    <input id = "RedLine" type="button" value="确定" onclick="drawTheme()" />
</td>
</tr>
</table>
</ContentTemplate>
    <Triggers>
        <asp:AsyncPostBackTrigger ControlID="colorTable" EventName="Init" />
    </Triggers>
</asp:UpdatePanel>

```

(2) 把用户控件封装在 FloatingPanel 中，脚本程序如下：

```

esri:FloatingPanel ID="fpUserTheme" runat="server" BackColor="White"
    BorderColor="Gray" BorderStyle="Solid" BorderWidth="1px"
Font-Names="Verdana"
    Font-Size="8pt" ForeColor="Black" Height="250px" Title="
自定义专题图"
    TitleBarColor="WhiteSmoke" TitleBarHeight="20px" TitleBar
SeparatorLine="False"
    Transparency="35" Width="250px" Visible="False" >
    <uc3:UserTheme ID="UserTheme1" runat="server" />
</esri:FloatingPanel>

```

(3) 封装 UserControl 后，就可以用 JavaScript 来控制 FloatingPanel 的显示：

```

<script language="javascript" type="text/javascript">
    function createCustomTheme()
    {
        var locdialog = $find("fpUserTheme");
        if(locdialog)
        {
            locdialog.show();
            locdialog._moveTo(350,100);
        }
    }
</script>

```

(4) 初始化 UserControl 的显示，列举省份和省份的颜色：

```

protected void colorTable_Init(object sender, EventArgs e)
{
    string strLayerName = "省份";

    this.colorTable.Rows.Clear();
    IEnumerable func_enum = null;
    Map mapctrl = this.Page.FindControl("Map1") as Map;
    func_enum = mapctrl.GetFunctionalities();
}

```

```

        System.Data.DataTable datatable;
        foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gisfunction in
func_enum)
        {
            ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource = null;
            gisresource = gisfunction.Resource;
            bool supported = false;

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

                string[] lids;
                string[] lnames;

                qfunc.GetQueryableLayers(null, out lids, out lnames);
                if (lids == null) continue;
                int selindex = -1;
                for (int i = 0; i < lnames.Length; i++)
                {
                    if (lnames[i] == strLayerName)
                    {
                        selindex = i;
                        break;
                    }
                }

                if (selindex > -1)
                {
                    ESRI.ArcGIS.ADF.Web.QueryFilter qfilter = new QueryFilter();
                    qfilter.SubFields.Add("name");
                    qfilter.ReturnADFGeometries = false;
                    qfilter.MaxRecords = 2000;

                    datatable = qfunc.Query(null, lids[selindex], qfilter);
                    if (lids == null) continue;
                    if (datatable != null)
                    {
                        TableRow tr = new TableRow();
                        TableCell tc = new TableCell();
                        Random rand = new Random();
                        Color cc = new Color();
                        int i = 0;
                        string temp = string.Empty;

                        Hashtable htable = GetUniqueValue(datatable, "NAME");

                        foreach (DictionaryEntry de in htable)

```

```

        {
            tr = new TableRow();
            tc = new TableCell();
            tc.Attributes.Add("style", "text-align:left");
            tc.Text      =      de.Key.ToString().Trim();//ddr["LANDUS
ECODE"].ToString().Trim();

            tr.Cells.Add(tc);
            tc = new TableCell();
            cc  = Color.FromArgb(rand.Next(255), rand.Next(255),
rand.Next(255));

            tc.ID = colorTable.ID + "_r" + i.ToString() + "_c1";
            tc.BackColor = cc;

            tc.Attributes.Add("style", "cursor:hand;width:50%;");
            tc.Attributes.Add("onclick", @"
var arr =showModalDialog( 'selcolor.htm','','font-family:宋体;font-size:12;
dialogWidth:375px; dialogHeight:245px:center=yes;status:no;help:no' );
if(arr != null)
    document.getElementById('fpUserTheme_UserThemel_" + tc.ID + @").style.
backgroundColor=arr;

");
            tr.Cells.Add(tc);
            colorTable.Rows.Add(tr);
            i++;
        }
        colorTable.Attributes.Add("type", "usetype");
    }
}
}
}

```

初始化用户界面如图 9.13 所示。从初始化代码可以看出单击颜色列弹出颜色选择对话框，如图 9.14 所示。

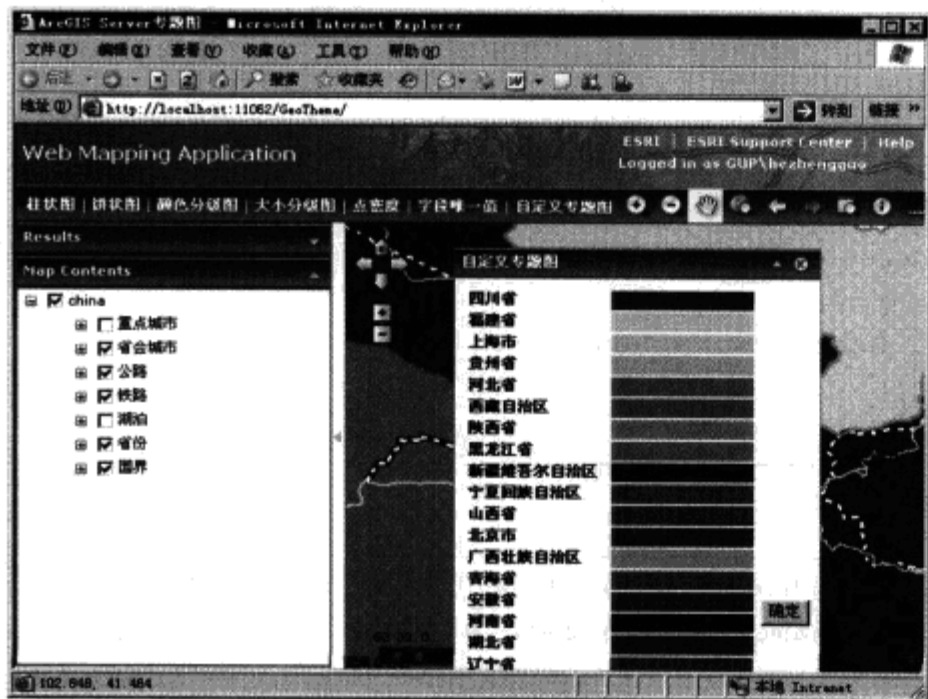


图 9.13 列举省份和颜色

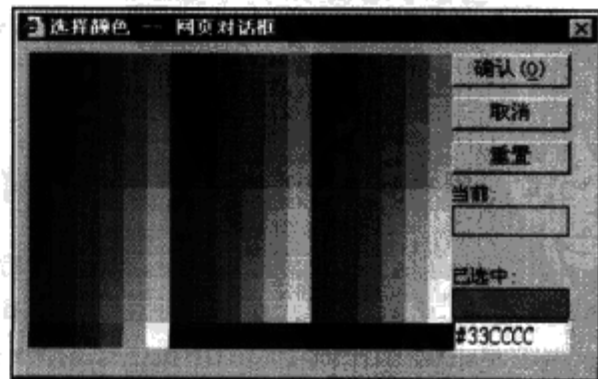


图 9.14 颜色编辑器

(5) 创建新的 GraphicsLayer:

```

ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource
GetGraphicsResource(MapResourceManager mapResourceManager)
{
    MapResourceItem resourceItem = mapResourceManager.ResourceItems.Find(__
ResourceName__);
    if (resourceItem == null)
    {
        #region 创建新的 ElementGraphicsLayer
        resourceItem = new ESRI.ArcGIS.ADF.Web.UI.WebControls.MapResource
Item();
        resourceItem.Definition = new ESRI.ArcGIS.ADF.Web.UI.WebControls.
GISResourceItemDefinition("<Definition DataSourceDefinition=\"In Memory\" Data
SourceType=\"GraphicsLayer\" Identity=\"\" ResourceDefinition=\"\" DataSourceShared
=\"True\"></Definition>");
        resourceItem.Name = __ResourceName__;
        resourceItem.DisplaySettings = new ESRI.ArcGIS.ADF.Web.Display Settings();
        resourceItem.DisplaySettings.Visible = true;
        resourceItem.DisplaySettings.DisplayInTableOfContents = false;
        resourceItem.DisplaySettings.EnableDynamicTiling = false;
        resourceItem.DisplaySettings.ImageDescriptor.ImageFormat = ESRI.ArcGIS.
ADF.Web.ImageFormat.PNG32;
        mapResourceManager.ResourceItems.Insert(0, resourceItem);
        resourceItem.DisplaySettings.ImageDescriptor.TransparentBackground =
true;

        resourceItem.DisplaySettings.DisplayInTableOfContents = false;
        ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource resource =
resourceItem.CreateResource() as ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource;
        resource.Graphics.DataSetName = __ResourceName__ + ".DataSet";
        #endregion
        #region 初始化 resource and functionalities
        if (mapResourceManager.Initialized)
            resourceItem.InitializeResource();
        #endregion
        return resource;
    }
    else
    {
        resourceItem.InitializeResource();
        return resourceItem.Resource
as ESRI.ArcGIS.ADF.Web.DataSources. Graphics.MapResource;
    }
}

```

(6) 在 GraphicsLayer 中添加几何对象:

```

public ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement AddGeometry (Map Resource
Manager mapResourceManager, string layerName, ESRI.ArcGIS.ADF.Web. Geometry.Geometry
geom, System.Drawing.Color color)
{

```

```

    ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer layer = Get
    Layer(mapResourceManager, layerName);
    if (layer == null) //No layer. Create one
    {
        layer = CreateLayer(mapResourceManager, layerName);
    }
    ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement elm = new ESRI.ArcGIS.
    ADF.Web.Display.Graphics.GraphicElement(geom, color);

    ESRI.ArcGIS.ADF.Web.Display.Symbol.SimpleFillSymbol simpleFillSymbol =
    new ESRI.ArcGIS.ADF.Web.Display.Symbol.SimpleFillSymbol();
    simpleFillSymbol.Color = color;
    simpleFillSymbol.Transparency = 0;
    ESRI.ArcGIS.ADF.Web.Display.Symbol.Symbol sym = simpleFillSymbol as
    ESRI.ArcGIS.ADF.Web.Display.Symbol.Symbol;
    elm.Symbol = sym;
    System.Data.DataRow row = layer.Add(elm);
    return elm;
}

```

(7) 刷新地图:

```

public static void RefreshMap(Map map)
{
    if (map.ImageBlendingMode == ImageBlendingMode.WebTier)
        map.Refresh();
    else
        map.RefreshResource(__ResourceName__);
}

```

系统的运行界面如图 9.15 所示。



图 9.15 自定义专题图

9.6 小结

本章主要讲述了 ArcGIS Server 中专题图的实现，基本涵盖了所有的专题图类型。读者掌握本章所讲述的专题图的实现方法后，可以在此基础上实现各种专题图，本章的详细代码参见附带光盘。由于专题图是以不同的符号来表示的，有关符号化的用法第 10 章将详细讲述。

第 10 章

ArcGIS Server 符号化

地图可视化的主要目的就是将地形地貌的知识用图形表示出来，是一种空间认知行为，它借助可视化软件将其他形式的数据用可见的图形表示，并针对同一数据集，采用多种图形显示。由于地图符号是地图的语言，是表达地理事物的基本手段，在现实应用中，无论是在地图制图系统中还是在 GIS 中，地图数据符号化显示都是极其重要的。专题图也是符号化中的一种，在第 9 章中已经讲述。

ArcGIS 将其符号分为点状符号（MarkerSymbol）、线状符号（LineSymbol）、面状符号（FillSymbol）等类型。ArcGIS Server 也一样，也支持上述的符号类型。ArcGIS Server 不仅可以通过 ADF 提供的对象来自定义点、线、面，还可以通过 ArcObjects 来调用符号库中的符号。

10.1 使用符号库中的符号进行符号化

ArcGIS 自带有丰富的符号库，在 ArcGIS Server 中可以使用 ArcObjects 来调用符号库中的符号对图层进行符号化。

调用 ArcGIS 符号库中符号的步骤如下。

（1）首先要知道符号库的符号分类，才能在对图层进行符号化，挑选合适的符号进行符号化。程序如下所示：

```
//初始化符号类表
ESRI.ArcGIS.Server.IServerContext pServerContext = GetServerContext();
ESRI.ArcGIS.Display.IStyleGallery pStyleGallery = pServerContext.
CreateObject("esriDisplay.ServerStyleGallery") as ESRI.ArcGIS.Display. ServerStyle
Gallery;
ESRI.ArcGIS.Display.IStyleGalleryStorage pStyleGalleryStorage = pStyle
Gallery as ESRI.ArcGIS.Display.IStyleGalleryStorage; //管理 IStyleGallery 对象
string strServerPath = pStyleGalleryStorage.DefaultStylePath;//得到
Style 文件的默认目录
string strStyleFileName = strServerPath + "ESRI.ServerStyle";//Style
```

文件名称

定目标文件

```
pStyleGalleryStorage.TargetFile = strStyleFileName; //此处指
```

```
ESRI.ArcGIS.Display.IStyleGalleryClass pStyleGalleryClass;
for (int i = 0; i < pStyleGallery.ClassCount; i++)
{
    pStyleGalleryClass = pStyleGallery.get_Class(i);
    string strName = pStyleGalleryClass.Name;
    this.ClassName.Items.Add(strName);
}
```

(2) 选择符号库的种类后，就需要知道该类别下有哪些符号：

```
ESRI.ArcGIS.Display.IEnumStyleGalleryItem pEnumStyleGalleryItem = pStyleGallery.
get_Items(strClassName, pStyleGalleryStorage.TargetFile, "Default");
```

```
pEnumStyleGalleryItem.Reset();
```

```
ESRI.ArcGIS.Display.IStyleGalleryItem pStyleItem = pEnumStyleGallery
Item.Next();
```

```
int j = 0;
```

```
while (pStyleItem != null)
```

```
{
```

```
    SymbolName.Items.Add(pStyleItem.Name);
```

```
    if (j == 0)
```

```
    {
```

```
        //初始化图片
```

```
        ESRI.ArcGIS.Display.ISymbol sym = pStyleItem.Item as ESRI.
ArcGIS.Display.ISymbol;
```

```
        Bitmap bmp = StyleGalleryItemToBmp(32, 32, pStyleGalleryClass,
pStyleItem);
```

```
//SymbolToBitmap(pMSymbol as ISymbol, 50, 40);
```

```
        string strPath= Server.MapPath("images") + "\\temp.png";
```

```
        bmp.Save(strPath, System.Drawing.Imaging.ImageFormat.Png);
```

```
        Image1.ImageUrl = "~/images/temp.png";
```

```
        Image1.DataBind();
```

```
    }
```

```
    pStyleItem = pEnumStyleGalleryItem.Next();
```

```
    j++;
```

```
}
```

```
pServerContext.ReleaseContext();
```

```
}
```

把符号库中的符号转化为图片的程序如下：

```
public System.Drawing.Bitmap SymbolToBitmap(ESRI.ArcGIS.Display.ISymbol pSymbol,
int iwidth, int iheight)
```

```
{
```

```

//根据高宽创建图像
Bitmap bmp = new Bitmap(iwidth, iheight);
Graphics gImage = Graphics.FromImage(bmp);
gImage.Clear(Color.White);
double dpi = gImage.DpiX;

IEnvelope pEnvelope = new EnvelopeClass();
pEnvelope.PutCoords(0, 0, (double)bmp.Width, (double)bmp.Height);

tagRECT deviceRect;
deviceRect.left = 0;
deviceRect.right = bmp.Width;
deviceRect.top = 0;
deviceRect.bottom = bmp.Height;

IDisplayTransformation pDisplayTransformation = new DisplayTransformation
Class();

pDisplayTransformation.VisibleBounds = pEnvelope;
pDisplayTransformation.Bounds = pEnvelope;
pDisplayTransformation.set_DeviceFrame(ref deviceRect);
pDisplayTransformation.Resolution = dpi;

IGeometry pGeo = CreateSymShape(pSymbol, pEnvelope);
ESRI.ArcGIS.Geometry.SpatialReferenceEnvironment sre = new ESRI.ArcGIS.
Geometry.SpatialReferenceEnvironment();
ESRI.ArcGIS.Geometry.ISpatialReference pSR = sre.CreateGeographic
CoordinateSystem(4326);
pDisplayTransformation.SpatialReference = pSR;
//pGeo.SpatialReference = pSR;

System.IntPtr hdc = new IntPtr();
hdc = gImage.GetHdc();

//将符号的形状绘制到图像中
pSymbol.SetupDC((int)hdc, pDisplayTransformation);
pSymbol.Draw(pGeo);
pSymbol.ResetDC();
gImage.ReleaseHdc(hdc);
gImage.Dispose();

return bmp;
}

public ESRI.ArcGIS.Geometry.IGeometry CreateSymShape(ISymbol pSymbol, IEnvelope
pEnvelope)
{
    // 根据传入的符号以及外包矩形区域返回对应的几何空间实体(点、线、面)
    //判断是否为“点”符号
    ESRI.ArcGIS.Display.IMarkerSymbol IMarkerSym;
    IMarkerSym = pSymbol as IMarkerSymbol;
    if (IMarkerSym != null)
    {

```

```
        //如为“点”符号则返回 IEnvelope 的中心点
        IArea pArea;
        pArea = pEnvelope as IArea;
        return pArea.Centroid as IGeometry;
    }
    else
    {
        //判断是否为“线”符号
        ESRI.ArcGIS.Display.ILineSymbol IlineSym;
        ESRI.ArcGIS.Display.ITextSymbol ITextSym;
        IlineSym = pSymbol as ILineSymbol;
        ITextSym = pSymbol as ITextSymbol;
        if (IlineSym != null || ITextSym != null)
        {
            //返回 45 度的对角线
            ESRI.ArcGIS.Geometry.IPolyline IpLine;
            IpLine = new PolylineClass();
            IpLine.FromPoint = pEnvelope.LowerLeft;
            IpLine.ToPoint = pEnvelope.UpperRight;
            return IpLine as IGeometry;
        }
        else
        {
            //直接返回一个 IEnvelope 矩形区域
            return pEnvelope as IGeometry;
        }
    }
}
```

(3) 选择符号名称后，就可以获取符号：

```
if (pStyleItem.Name == strSymbolName)
{
    //初始化图片
    ESRI.ArcGIS.Display.ISymbol sym = pStyleItem.Item as ESRI.
ArcGIS.Display.ISymbol;
}
```

(4) 对选择的图层进行符号化：

```
if (pStyleItem.Name == strSymbolName)
{
    //初始化图片
    ESRI.ArcGIS.Display.ISymbol sym = pStyleItem.Item as ESRI.ArcGIS.
Display.ISymbol;

    ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.Imap
Server)pServerContext.ServerObject;
    ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.
ArcGIS.Carto.IMapServerObjects2)mapServer;
    string mapName = mapServer.DefaultMapName;
```

```

ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);

ESRI.ArcGIS.Carto.ILayer pLayer = aoMap.get_Layer(LayerName.
SelectedIndex); //得到图层
ESRI.ArcGIS.Carto.IGeoFeatureLayer pGeoLayer = pLayer as ESRI.
ArcGIS.Carto.IGeoFeatureLayer;
ESRI.ArcGIS.Carto.ISimpleRenderer pSimRenderer = pServerContext.
CreateObject("esriCarto.SimpleRenderer") as ESRI.ArcGIS.Carto.ISimpleRenderer;
pSimRenderer.Symbol = sym;
pGeoLayer.Renderer = pSimRenderer as ESRI.ArcGIS.Carto.Ifeature
Renderer;
}

```

(5) 刷新地图:

//刷新地图显示图表及图例

```

mapServerObjects.RefreshServerObjects();
Map Map1 = Page.FindControl("Map1") as Map;
Toc Toc1 = Page.FindControl("Toc1") as Toc;
Map1.RefreshResource("MapResourceItem0");
Toc1.BuddyControl = "Map1";
//Toc1.Refresh();
Map1.Refresh();
m_callbackInvocation = Map1.CallbackResults.ToString();

```

系统的运行界面如图 10.1 所示。

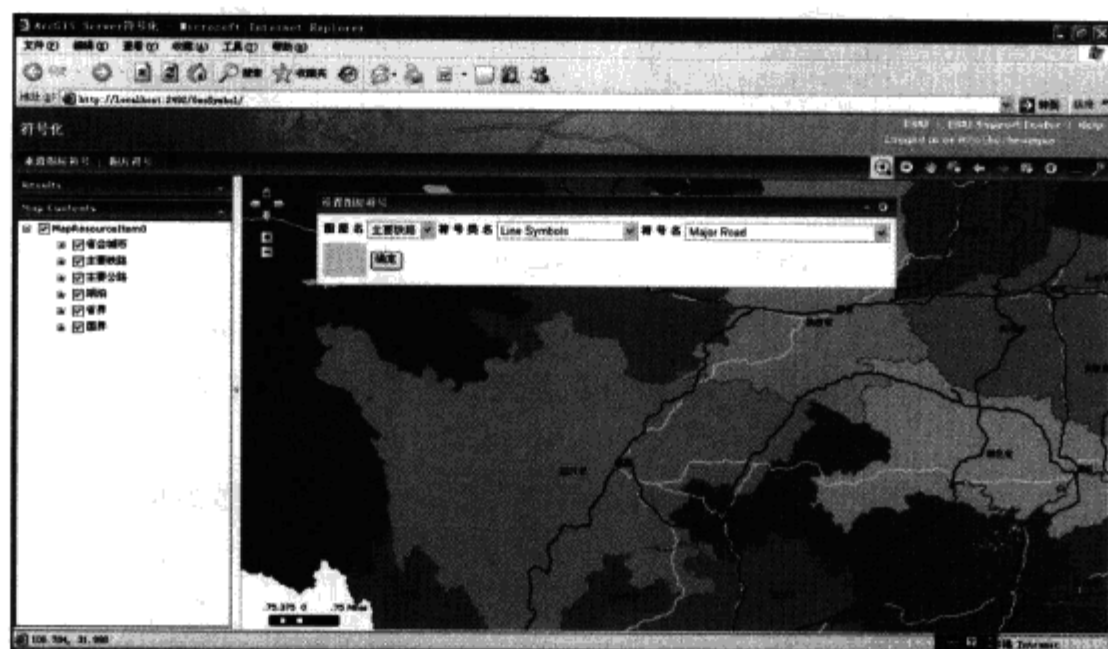


图 10.1 利用 ArcGIS Server 符号库中的符号来符号化图层

10.2 使用 TrueType 进行符号化

很多 GIS 软件都能识别 TrueType 的点符号，ArcGIS 系列软件当然也不例外，也能加载 TrueType 的点符号。利用其他的工具制作好 TrueType 字体后，用 ArcMap 把 TrueType 字体制作成 Style 文件。制作 Style 文件的步骤如下。

- (1) 把制作的 TrueType 文件拷贝到 \WINDOWS\Fonts 目录下。
- (2) 单击 ArcMap 的 “Tools” → “Styles” → “Style Manager” 菜单，如图 10.2 所示。

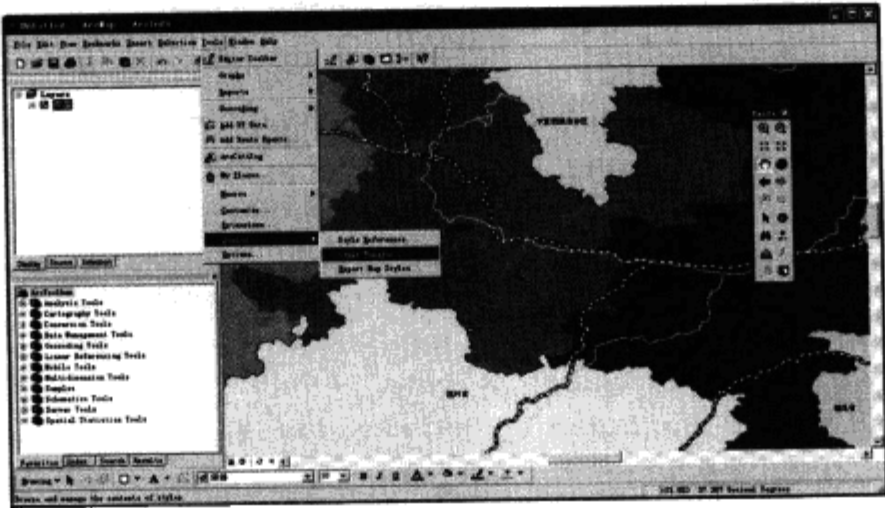


图 10.2 Style Manager 菜单

(3) 系统弹出“Style Manager”对话框，选中“Marker Symbols”项，在右边的列表框中单击鼠标右键，选择“New”→“Marker Symbol”选项，如图 10.3 所示。

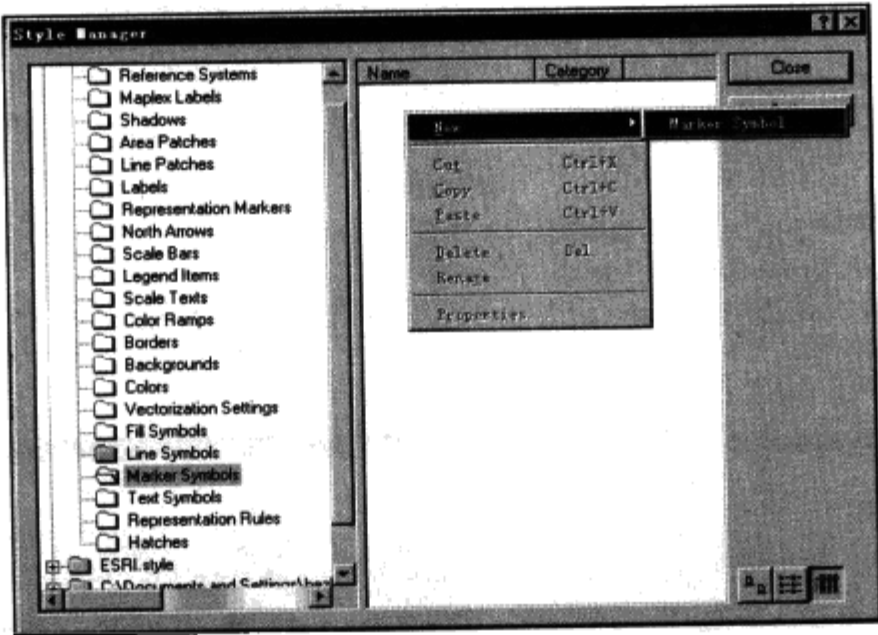


图 10.3 “Style Manager”对话框

(4) 系统弹出符号编辑框，在“Type”下拉列表框中选中“Character Marker Symbo”选项，“Font”下拉列表框中选中自己编辑的 TrueType 字段，在符号列表中选中对应的符号，如图 10.4 所示。

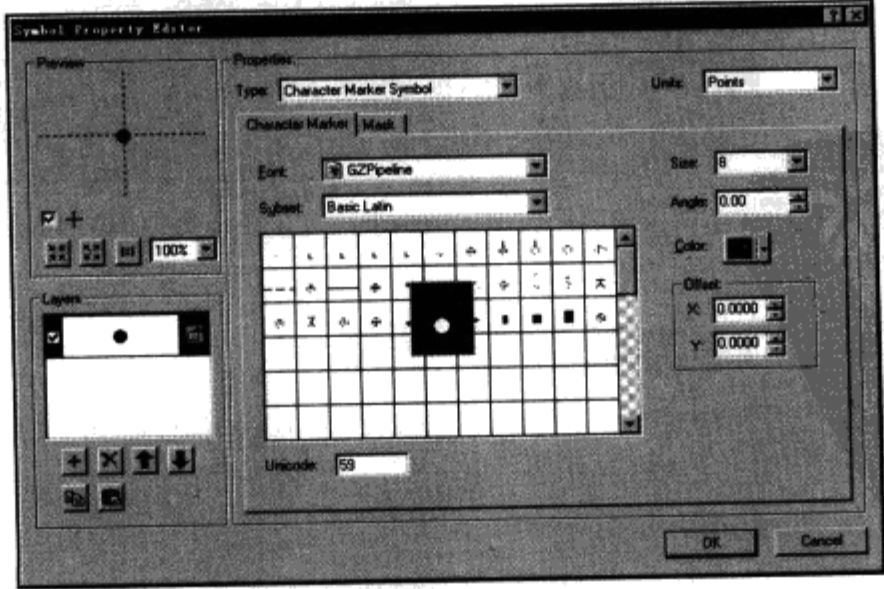


图 10.4 编辑符号

(5) 编辑完所有的符号后，即生成了对应的 Style 文件。有了相应的 Style 文件，调用方法与代码与 10.1 节相同，只需把对应的 Style 文件替换成自己的 Style 文件就可以了。

10.3 自画符号

ArcGIS Server ADF 提供一套符号体系，供用户开发自定义的符号。ArcGIS Server 常用的符号体系如图 10.5 所示。

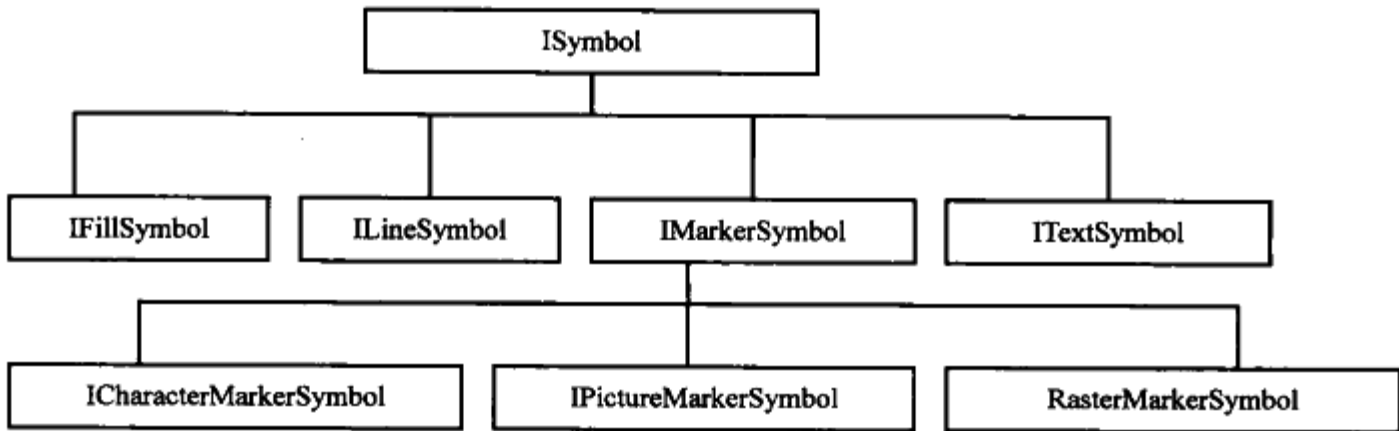


图 10.5 常用符号结构体系图

10.3.1 MarkerSymbol 符号

MarkerSymbol 是指点符号，点符号的属性有：填充颜色、边框线颜色、点符号类型、符号宽度和透明度。

创建一个简单的 MarkSymbol 示例程序如下：

```
static public Symbol CreateSimpleMarkerSymbol(Color color, double transparency,
MarkerSymbolType type,Color outcolor)
{
    SimpleMarkerSymbol simpleMarkerSymbol = new SimpleMarkerSymbol();
    simpleMarkerSymbol.Color = color;
    simpleMarkerSymbol.Transparency = transparency;
    simpleMarkerSymbol.Width = 12;
    simpleMarkerSymbol.Type = type;
    simpleMarkerSymbol.OutlineColor = outcolor;
    return simpleMarkerSymbol;
}
```

这样，开发人员就可以根据自己的需要来创建合适的符号。

10.3.2 LineSymbol 符号

LineSymbol 是指线符号，线符号的属性有：颜色、透明度、线型和线宽。创建一个简单的线符号示例程序如下：

```
static public Symbol CreateSimpleLineSymbol(Color color, double transparency)
{
    SimpleLineSymbol simpleLineSymbol = new SimpleLineSymbol();
```

```
        simpleLineSymbol.Color = color;
        simpleLineSymbol.Transparency = transparency;
        simpleLineSymbol.Width = 5;
        simpleLineSymbol.Type = LineType.Solid;
        return simpleLineSymbol;
    }
}
```

10.3.3 FillSymbol 符号

FillSymbol 是指填充符号，也就是面对象的填充符号。它有边框线颜色、边框节点类型、连接点类型、边框行宽度、填充颜色、填充类型和透明度等属性。创建一个简单的填充符号示例程序如下：

```
static public Symbol CreateSimpleFillSymbol(Color color, double transparency)
{
    SimpleFillSymbol simpleFillSymbol = new SimpleFillSymbol();
    simpleFillSymbol.Color = color;
    simpleFillSymbol.Transparency = transparency;
    simpleFillSymbol.FillType = PolygonFillType.Solid;
    return simpleFillSymbol;
}
```

10.3.4 使用图片进行符号化

图片符号，就是用图片表示某一类特殊的元素，一般用于点符号，也可用于面的填充图案。用来表示点的示例程序如下：

```
public static ESRI.ArcGIS.ADF.Web.Display.Symbol.RasterMarkerSymbol LoadRasSym
(string filename, int size, System.Drawing.Imaging.ImageFormat format)
{
    System.Drawing.Image sysmImage = System.Drawing.Image.FromFile(filename);
    ESRI.ArcGIS.ADF.Web.Display.Symbol.RasterMarkerSymbol rms = new ESRI.
ArcGIS.ADF.Web.Display.Symbol.RasterMarkerSymbol();
    ESRI.ArcGIS.ADF.Web.CartoImage cimg = new ESRI.ArcGIS.ADF.Web. Carto
Image(filename);
    rms.Image = cimg;
    rms.Width = size;
    return rms;
}
```

ArcGIS Server 中符号很多，常用的就是点、线、面、图片。这些符号表现多与 GraphicsLayer 相关，下面对 GraphicsLayer 进行简单的介绍。

GraphicsLayer 有两种，ElementGraphicsLayer 和 FeatureGraphicsLayer，从类关系图可以看出。

Inheritance Hierarchy

```

System.Object
  System.ComponentModel.MarshalByValueComponent
    System.Data.DataTable
      ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicsLayer
        ESRI.ArcGIS.ADF.Web.Display.Graphics.FeatureGraphicsLayer
          ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer
  
```

ElementGraphicsLayer 和 FeatureGraphicsLayer 是两个相似类型的东西，都是 GraphicsLayer，都显示在地图数据的最上层，当对数据进行查询，缓冲区分析需要添加临时数据时使用。

ElementGraphicsLayer.Add 添加的是 graphicElement，就是说在 ElementGraphicsLayer 保存的是 graphicElement 集合，每个 graphicElement 都有各自的样式和显示对象。这个 graphicElement 可以是点、线、面，并且每种都有自己的显示样式。可以由 rows 通过转换得到 graphicElement。graphicElement 有一个几何体属性和一个显示样式属性。

FeatureGraphicsLayer.add 添加的是 Geometry，就是说 FeatureGraphicsLayer 保存的是 Geometry 的集合，FeatureGraphicsLayer 只有一种样式对象，而且只能保存一种几何类型，要么是点，要么是线，要么是面。通过得到 rows 可以得到几何体，FeatureGraphicsLayer 有一个 render 属性，通过这个属性可以进行显示样式的设置。

下面给出操作 ElementGraphicsLayer 的示例代码。

如得到 GraphicsResource 的示例程序：

```

ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource    GetGraphicsResource(Map
ResourceManager mapResourceManager)
{
    MapResourceItem resourceItem = mapResourceManager.ResourceItems.Find
(__ResourceName__);
    if (resourceItem == null)
    {
        #region 创建新的 ElementGraphicsLayer
        resourceItem = new ESRI.ArcGIS.ADF.Web.UI.WebControls.MapResource
Item();

        resourceItem.Definition = new ESRI.ArcGIS.ADF.Web.UI.WebControls.
GISResourceItemDefinition("<Definition DataSourceDefinition=\"In Memory\" Data
SourceType=\"GraphicsLayer\" Identity=\"\" ResourceDefinition=\"\" DataSourceShared
=\"True\"></Definition>");
        resourceItem.Name = __ResourceName__;
        resourceItem.DisplaySettings = new ESRI.ArcGIS.ADF.Web.Display
Settings();

        resourceItem.DisplaySettings.Visible = true;
        resourceItem.DisplaySettings.DisplayInTableOfContents = false;
        resourceItem.DisplaySettings.EnableDynamicTiling = false;
        resourceItem.DisplaySettings.ImageDescriptor.ImageFormat = ESRI.ArcGIS.
ADF.Web.ImageFormat.PNG32;
        mapResourceManager.ResourceItems.Insert(0, resourceItem);
        resourceItem.DisplaySettings.ImageDescriptor.TransparentBackground =
true;

        resourceItem.DisplaySettings.DisplayInTableOfContents = false;
        ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource resource =
resourceItem.CreateResource() as ESRI.ArcGIS.ADF.Web.DataSources.Graphics.
MapResource;
  
```

```

        resource.Graphics.DataSetName = __ResourceName__ + ".DataSet";
    #endregion
    #region 初始化 resource and functionalities
    if (mapResourceManager.Initialized)
        resourceItem.InitializeResource();
    #endregion
    return resource;
}
else
{
    resourceItem.InitializeResource();
    return resourceItem.Resource as ESRI.ArcGIS.ADF.Web.DataSources.
Graphics.MapResource;
}
}

```

创建新的 ElementGraphicsLayer 示例程序如下：

```

ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer
CreateLayer(MapResourceManager mapResourceManager, string layerName)
{
    ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource resource = Get
GraphicsResource(mapResourceManager);
    ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer layer = new
ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer(layerName);
    resource.Graphics.Tables.Add(layer);
    return layer;
}

```

向 ElementGraphicsLayer 中添加几何对象，可以以自定义的符号来表示该对象：

```

public ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement AddGeometry(Map
ResourceManager mapResourceManager, string layerName, ESRI.ArcGIS.ADF.Web.Geometry.
Geometry geom, System.Drawing.Color color)
{
    ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer layer = Get
Layer(mapResourceManager, layerName);
    if (layer == null) //No layer. Create one
    {
        layer = CreateLayer(mapResourceManager, layerName);
    }
    ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement elm = new ESRI.
ArcGIS.ADF.Web.Display.Graphics.GraphicElement(geom, color);

    ESRI.ArcGIS.ADF.Web.Display.Symbol.SimpleFillSymbol simpleFillSymbol =
new ESRI.ArcGIS.ADF.Web.Display.Symbol.SimpleFillSymbol();
    simpleFillSymbol.Color = color;
    simpleFillSymbol.Transparency = 0;
    ESRI.ArcGIS.ADF.Web.Display.Symbol.Symbol sym = simpleFillSymbol as
ESRI.ArcGIS.ADF.Web.Display.Symbol.Symbol;
    elm.Symbol = sym;
    System.Data.DataRow row = layer.Add(elm);
}

```

```
        return elm;
    }
```

刷新 GraphicsLayer 的示例程序如下：

```
public static void RefreshMap(Map map)
{
    if (map.ImageBlendingMode == ImageBlendingMode.WebTier)
        map.Refresh();
    else
        map.RefreshResource(__ResourceName__);
}
```

有了 GraphicsLayer，开发人员就可以定义各种图形了。

10.4 小结

本章主要讲述 ArcGIS Server 中常用的符号，以及一些简单的用法。只要弄清符号的基本用法，就可以开发出符合自己需求的符号。回忆第 9 章的内容，专题图其实也是符号的一种，是比较特殊的符号而已。学习完第 9 章和第 10 章后，读者就可以对 ArcGIS Server 应用得心应手了。

第 11 章

ArcGIS Server 数据在线编辑

前面在介绍控件的时候，介绍过 ArcGIS Server 的编辑功能。ArcGIS Server 中的服务有池化和非池化两种类型，对于编辑功能而言，池化和非池化类型服务都可以进行编辑，区别在于：

- 非池化服务中的数据可以是注册版本或非注册版本的，如果注册了版本，可以进行 redo、undo 编辑操作，并可以进行冲突处理。非注册版本的数据，不能进行 redo、undo 操作，可以对编辑内容选择一次保存和不保存。
- 池化服务中的数据只能是非注册版本。这种情况下，编辑功能非常简单，编辑的内容一旦保存，就无法进行 redo、undo 操作。

11.1 EditorTask 的数据编辑

EditorTask 是 ArcGIS Server 已经封装好针对 ArcSDE 空间数据的编辑。下面将详细讲述 EditorTask 中的点、线和面的编辑。

11.1.1 点的编辑

点的编辑应首先准备 ArcSDE 的点数据，然后在 EditorTask 工具中配置编辑的相关设置，步骤如下。

(1) 打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 MapResourceManager、Map 和 EditorTask 等控件，然后将它拖到 Web 页面中，如图 11.1 所示。

(2) 配置编辑点数据的数据源，如图 11.2 所示。

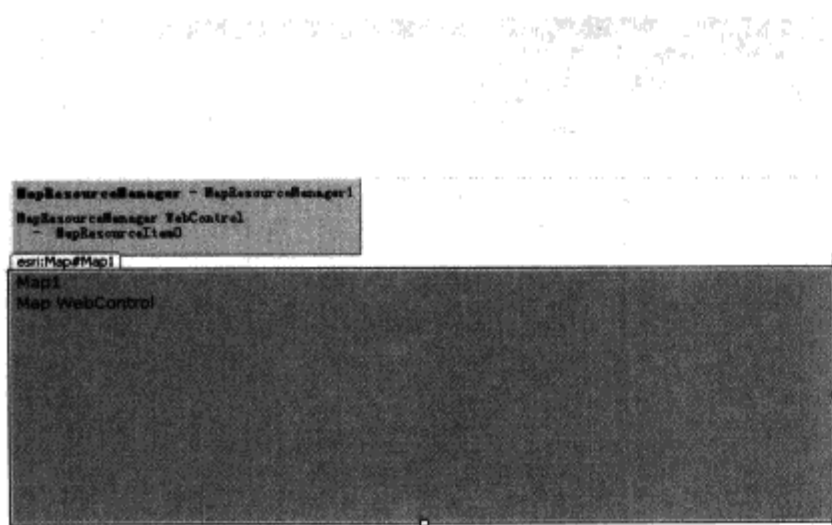


图 11.1 将控件添加到页面

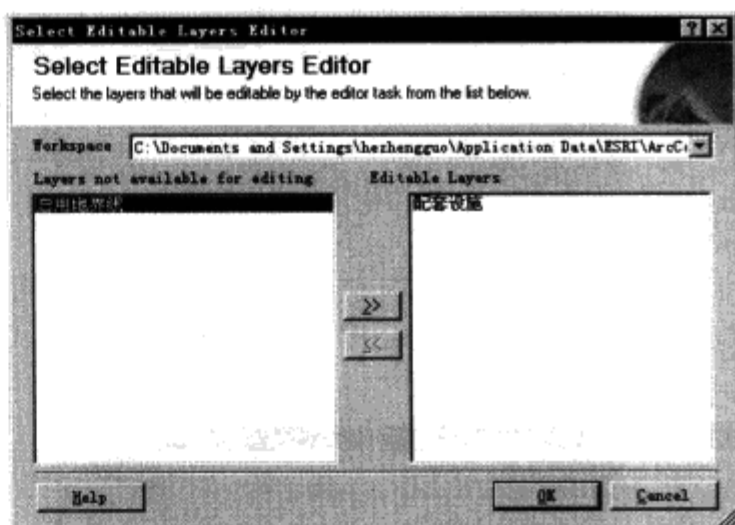


图 11.2 配置编辑的点数据图层界面

(3) 运行并进行点编辑，运行界面如图 11.3 所示。

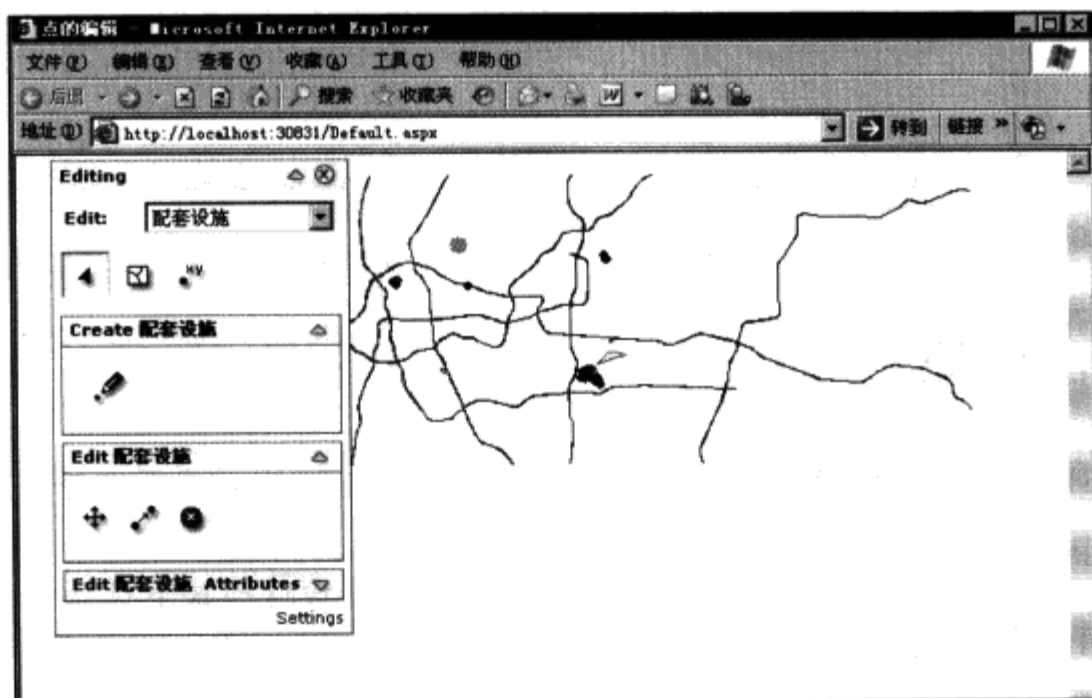


图 11.3 点编辑运行界面

点编辑提供的功能有：选中、清除选中、添加 XY 坐标、添加点、移动点、拷贝点、删除点以及点要素的属性编辑。

11.1.2 线的编辑

线的编辑应首先准备 ArcSDE 的线数据，然后在 EditorTask 工具中配置好编辑的相关设置，步骤如下。

(1) 打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 MapResourceManager、Map 和 EditorTask 等控件，然后将它拖到 Web 页面中，与点对象的编辑相似。

(2) 配置编辑线数据的数据源，如图 11.4 所示。

(3) 运行并进行线编辑，运行界面如图 11.5 所示。

线编辑提供的功能有：选择、清除选择、显示线的节点、添加 XY 坐标、移动线、拷贝线、切割线、合并线、删除线，添加节点、移动节点、删除节点以及线要素的属性编辑。

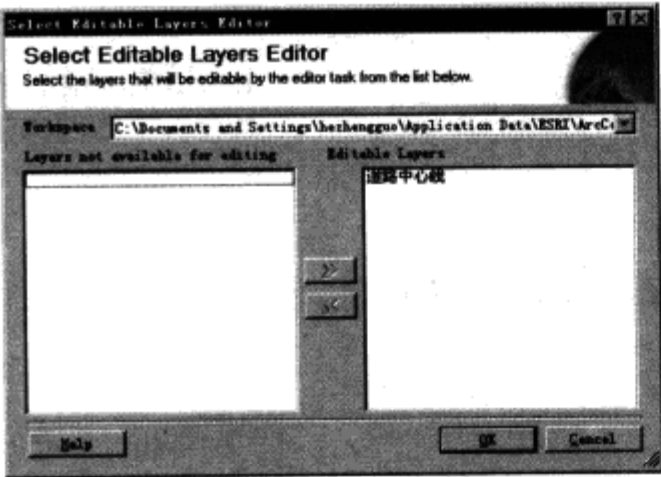


图 11.4 配置线数据源

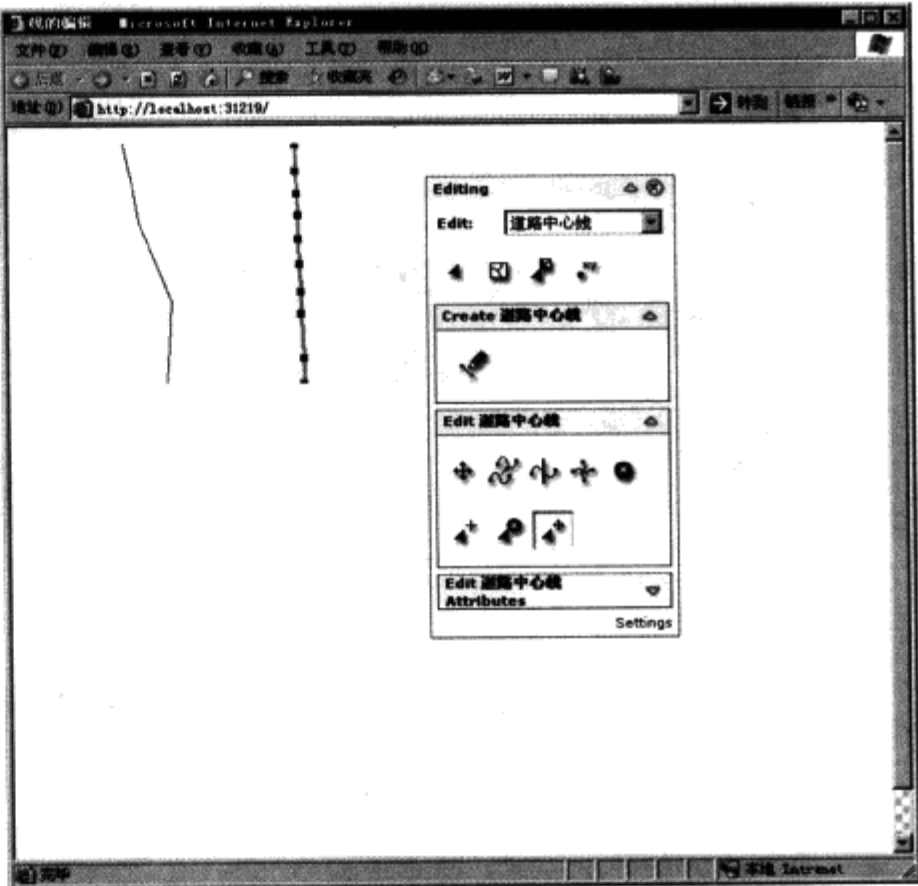


图 11.5 线编辑的运行界面

11.1.3 面的编辑

面的编辑应首先准备 ArcSDE 的面数据，然后在 EditorTask 工具中配置好编辑的相关设置，步骤如下。

- (1) 打开 Visual Studio.NET 并且创建一个 Web 站点，在设计模式下打开一个页面，选择工具箱并展开 ArcGIS 控件标签，选中 MapResourceManager、Map 和 EditorTask 等控件，然后将它拖到 Web 页面中，与点对象的编辑相似。
- (2) 配置编辑面数据的数据源，如图 11.6 所示。
- (3) 运行并进行面编辑，运行界面如图 11.7 所示。

面编辑提供的功能有：选择、清除选择、显示面的节点、添加 XY 坐标、移动面、拷贝面、切割面、合并面、删除面，添加节点、移动节点、删除节点以及面要素的属性编辑。

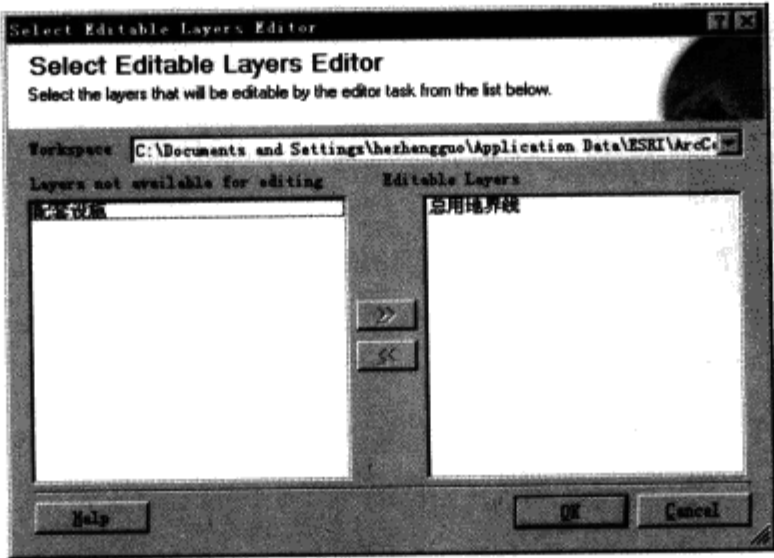


图 11.6 配置数据面数据

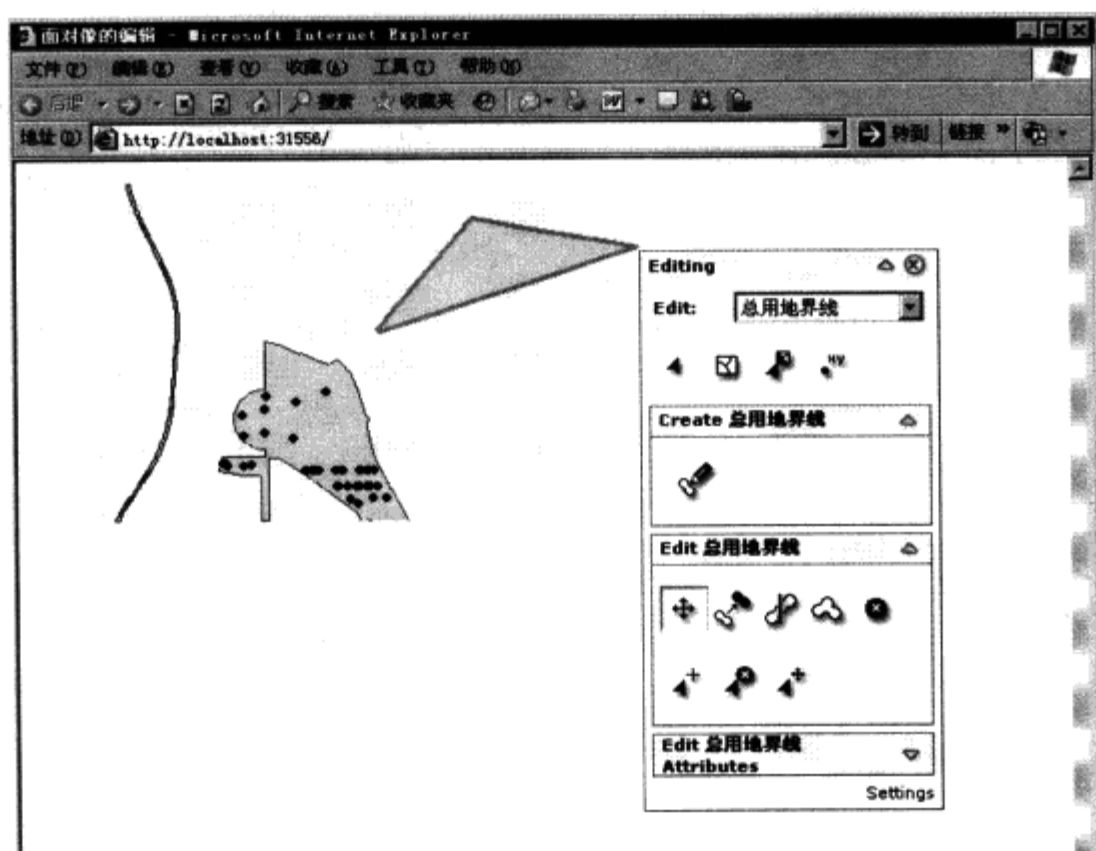


图 11.7 面编辑的运行界面

11.2 编辑功能的定制

11.1 节介绍了 EditorTask 的数据编辑，细心的读者可能会发现点、线、面编辑工具的面板大同小异。本节主要介绍根据特定应用的需求来定制编辑。

首先介绍编辑面板的功能分区如图 11.8 所示，让读者对 EditorTask 提供的编辑面板有一个全面的了解。

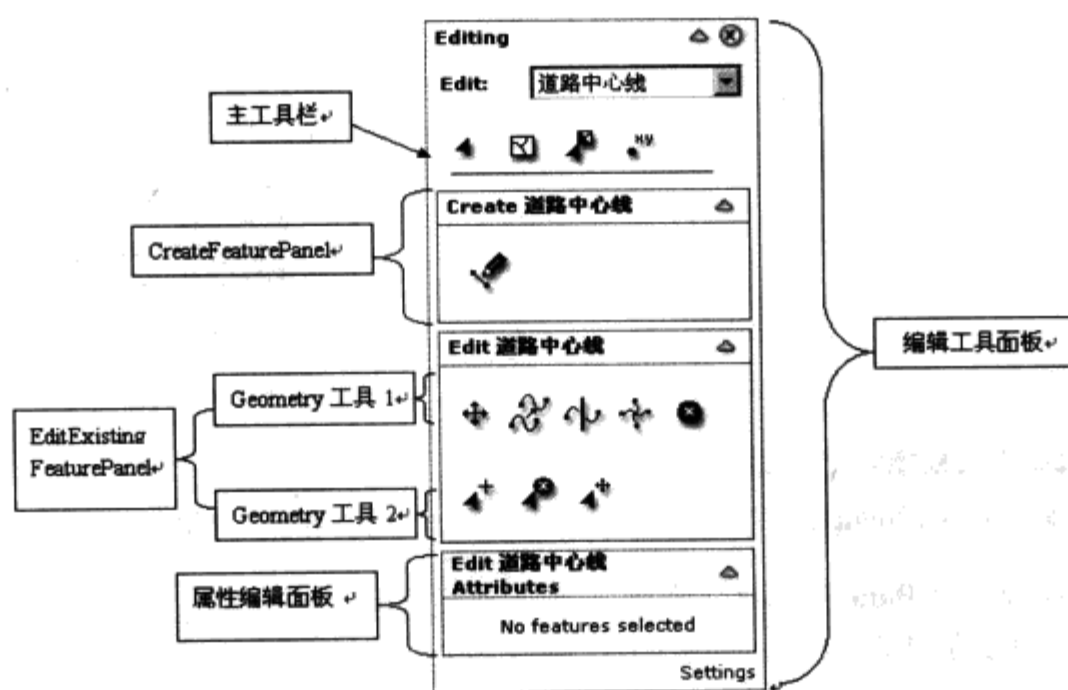


图 11.8 编辑面板的功能分区图

编辑功能定制要使用 `ESRI.ArcGIS.ADF.ArcGISSEditor.Editor` 和 `ESRI.ArcGIS.ADF.ArcGISSEditor.Tools` 命名空间。这两个命名空间提供组成编辑任务的各个控件和类，这些控件和类是实现编辑功能定制的前提。从图 11.8 中可以看出，编辑工具面板有很多功能分区，

我们可以在这些功能分区中定制自己的工具、工具栏、控制面板。

下面我们首先来讲述在主工具栏添加自己的功能按钮。首先，添加 ToolsCreated 事件，如图 11.9 所示。在 EditorTask1_ToolsCreated 事件中编写程序如下：

```
if (e.Parent == this.EditorTask1.Editor)
{
    //在主工具栏中添加
    Toolbar toolbar = e.Toolbars[0];
    // 创建工具
    EditorTool tool = new EditorTool("MyTool1", Map1.ClientID, true,
    ToolGeometry.All, 1);
    tool.ClientAction = "Point";
    tool.DefaultImage = "~/images/verticecount.png";
    tool.HoverImage = "~/images/verticecount_ON.png";
    tool.SelectedImage = "~/images/verticecount_HOVER.png";
    tool.ServerActionAssembly = "App_Code";
    tool.ServerActionClass = "";
    tool.ToolTip = "IdentifyAllTool";
    //添加工具
    toolbar.ToolbarItems.Add(tool);
    toolbar.Width = new System.Web.UI.WebControls.Unit(toolbar.Width.
    Value + 35, UnitType.Pixel);
}
```

程序运行界面如图 11.10 所示。



图 11.9 添加 ToolsCreated 事件



图 11.10 在主工具栏中添加自己的工具

在 EditExistingFeaturePanel 中 Geometry 工具 1 和 Geometry 工具 2 也可以添加自定义的工具，方法与在主工具栏中添加工具类似，只是代码略有不同，示例程序如下：

```
if (e.Parent == this.EditorTask1.Editor.ExistingFeatureEditor)
{
    //创建工具，并把这个工具添加到编辑要素 panel 中的一个工具栏中
    EditorTool clip = new EditorTool("Clip", Map1.ClientID, false,
    ToolGeometry.All, 1);
```

```

clip.ClientAction = "Point";
clip.DefaultImage = "~/images/verticecount.png";
clip.SelectedImage = "~/images/verticecoun_ONt.png";
clip.HoverImage = "~/images/verticecount_HOVER.png";
clip.ToolTip = "节点个数";
clip.Text = "节点个数";
clip.ServerActionAssembly = "ESRI.ArcGIS.ADF.ArcGISSEditor.Editor";
clip.ServerActionClass = "ESRI.ArcGIS.ADF.ArcGISSEditor.Tools.
AddVertex";

double oldWidth = e.Toolbars[0].Width.Value;
e.Toolbars[0].Width = new System.Web.UI.WebControls.Unit(oldWidth + 35,
UnitType.Pixel);
e.Toolbars[0].ToolBarItems.Add(clip); //添加到第一个工具栏上
}

```

程序运行界面如图 11.11 所示。



图 11.11 EditExistingFeaturePanel 中添加工具

在 EditExistingFeaturePanel 中 Geometry 工具 2 添加工具,只需要把 e.Toolbars[0].ToolBarItems.Add(clip)改为 e.Toolbars[1].ToolBarItems.Add(clip)即可。

除了添加工具,还可以添加自己的工具栏,方法与上面的类似,示例程序如下:

```

if(e.Parent == this.EditorTask1.Editor.ExistingFeatureEditor)
{
//创建工具栏
EditorToolbar toolbar = new EditorToolbar();
toolbar.ID = "MyToolbar";
toolbar.BuddyControlType = BuddyControlType.Map;
toolbar.BuddyControls.Add(new BuddyControl("Map1"));
toolbar.ToolbarStyle = ToolbarStyle.ImageOnly;
toolbar.Alignment = Alignment.Left;
//创建工具
EditorTool tool = new EditorTool("MyTool2", Map1.ClientID, true,
ToolGeometry.All, 1);
tool.ClientAction = "Point";
tool.DefaultImage = "~/images/clip.png";
}

```

```

tool.HoverImage = "~/images/clip_HOVER.png";
tool.SelectedImage = "~/images/clip_ON.png";
//tool.ServerActionAssembly = "App_Code";
//tool.ServerActionClass = "CustomToolLibrary.IdentifyAllTool";
tool.ToolTip = "IdentifyAllTool";
toolbar.ToolbarItems.Add(tool);

e.Toolbars.Add(toolbar);
}

```

程序运行的界面如图 11.12 所示。

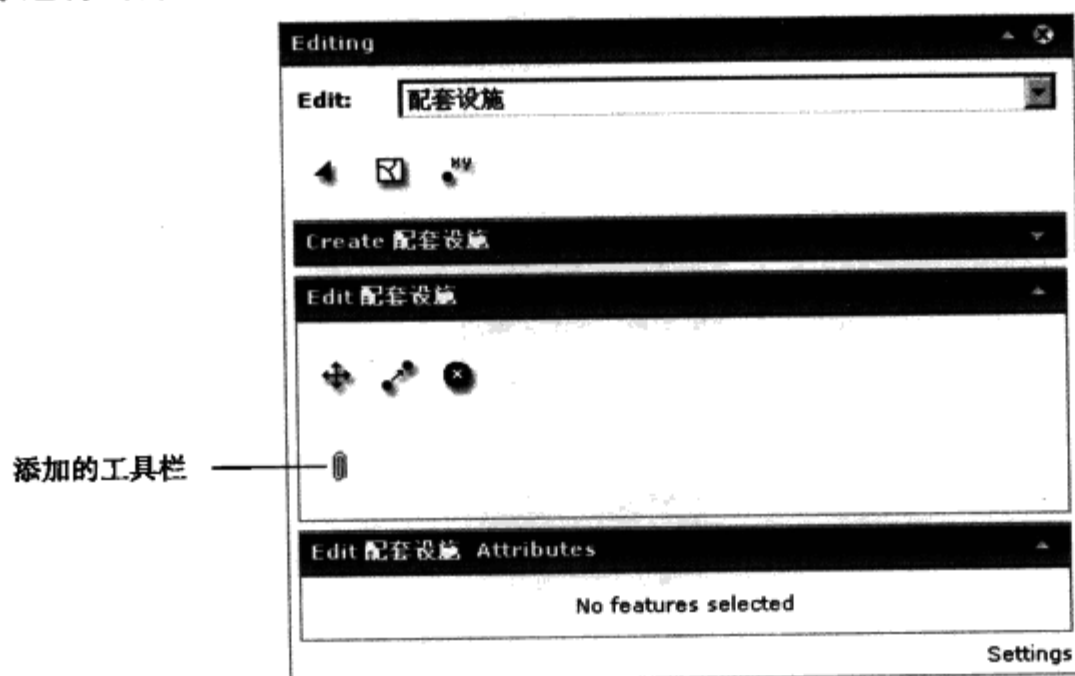


图 11.12 添加工具栏

上面讲述了工具栏和工具的自定义，下面来说说如何添加自定义的 Panel 到编辑工具面板中。首先，定义一个 Panel，该类继承 EditorPanel，示例程序如下：

```

public class MyEditPanel : ESRI.ArcGIS.ADF.ArcGISSEditor.Editor.EditorPanel
{
    public MyEditPanel(string title, ESRI.ArcGIS.ADF.ArcGISSEditor.Editor.
EditorTask task, string ID) : base(title, task, ID) { }
    protected override void CreateChildControls()
    {
        base.CreateChildControls();
        Table myTable = new Table();
        myTable.ID = "table1";
        TableRow tr = new TableRow();
        TableCell tc = new TableCell();
        tc.Text = "名字";
        tr.Cells.Add(tc);
        tc = new TableCell();
        TextBox textbox = new TextBox();
        textbox.ID = "TextBox1";
        tc.Controls.Add(textbox);
        tr.Cells.Add(tc);
    }
}

```



```
        tc = new TableCell();
        Button but = new Button();
        but.ID="But_OK";
        but.Text="查找";
        tc.Controls.Add(but);
        tr.Cells.Add(tc);
        myTable.Rows.Add(tr);
        this.Controls.Add(myTable);
    }
}
```

下一步，添加事件 EditorPanelsCreated，如图 11.13 所示。

```
protected void EditorTask1_EditorPanelsCreated(object sender, ESRI.ArcGIS.ADF.
ArcGISSEditor.Editor.EditorPanelsCreatedEventArgs e)
{
    MyEditPanel editorpanel = new MyEditPanel("添加编辑 Panel", EditorTask1,
"111");
    e.EditorPanels.Add(editorpanel);
}
```

程序运行界面如图 11.14 所示。

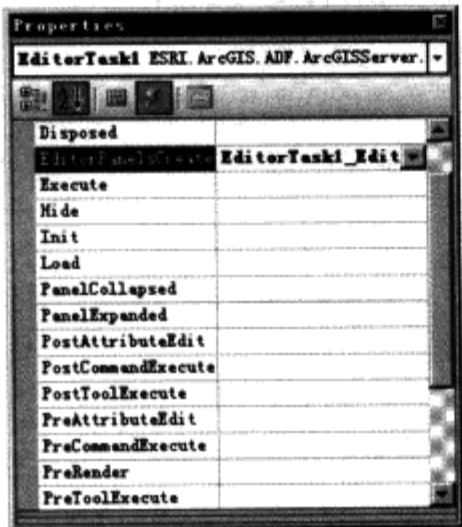


图 11.13 添加事件



图 11.14 添加自定义 Panel

11.3 编辑功能的扩展

在上一节中，介绍了如何编辑定制的编辑界面，本节将介绍如何实现具体的功能。以实现对选择要素进行裁减为例，来讲述如何实现编辑功能的扩展。

首先，来看看添加编辑工具的程序：

```
//创建 Toolbar
if (e.Parent == this.EditorTask1.Editor.ExistingFeatureEditor)
{
    //创建工具，并把这个工具添加到编辑要素 panel 中的一个工具栏中
    EditorTool clip = new EditorTool("Clip", Map1.ClientID, false,
ToolGeometry.Line|ToolGeometry.Polygon, 1);
    clip.ClientAction = ESRI.ArcGIS.ADF.Web.UI.WebControls.Constants.
```

```

HTML_DRAG_RECTANGLE;
        clip.DefaultImage = "~/images/clip.png";
        clip.SelectedImage = "~/images/clip_ON.png";
        clip.HoverImage = "~/images/clip_OVER.png";
        clip.ToolTip = "矩形裁减";
        clip.Text = "矩形裁减";
        clip.ServerActionAssembly = "App_Code";
        clip.ServerActionClass = "ClipFeatures";

        double oldWidth = e.Toolbars[0].Width.Value;
        e.Toolbars[0].Width = new System.Web.UI.WebControls.Unit(oldWidth + 35,
UnitType.Pixel);
        e.Toolbars[0].ToolBarItems.Add(clip);    //添加到第一个工具栏中
    }

```

读者可能会对代码 `clip.ServerActionAssembly = "App_Code"; clip.ServerActionClass = "ClipFeatures"` 比较熟悉, 没错, 这与前面 7.4 节中的自定义工具相同。其实类 `ESRI.ArcGIS.ADF.ArcGISSEditor.Tools.EditorTool` 继承于类 `ESRI.ArcGIS.ADF.Web.UI.WebControls.Tool`, 所以它们的实现几乎是相同的。略有不同的是, `Tool` 实现类 `ESRI.ArcGIS.ADF.Web.UI.WebControls.Tools.IMapServerToolAction`, `EditorTool` 类需要实现 `ESRI.ArcGIS.ADF.ArcGISSEditor.Tools.EditorServerToolAction`, 在编写编辑代码的时候, ArcGIS Server 已经封装好了很多功能, 如 `StartEditOperation()` 开始编辑, 用户不必重新编辑等。

在编辑任务上扩展编辑功能, 与第 7 章中的自定义工具和自定义命令的实现类似, 同时 ArcGIS Server 为开发人员封装了编辑环境, 开发人员只需要编写少量的代码就可以实现编辑功能的扩展。如裁剪类的实现程序如下:

```

public class ClipFeatures : ESRI.ArcGIS.ADF.ArcGISSEditor.Tools.Editor
ServerToolAction
{
    private System.Collections.Generic.List<int> IDsList = new System.Collections.
Generic.List<int>();

    public System.Collections.Generic.List<int> Features
    {
        get { return IDsList; }
    }

    protected override bool Init(ESRI.ArcGIS.ADF.ArcGISSEditor.Editor
editor)
    {
        IDsList.Clear();
        return base.Init(editor);
    }

    public ClipFeatures()
    {
        //
        // TODO: Add constructor logic here
    }
}

```

```

    //
}

protected override void EditorServerAction()
{
    //得到用户在界面画出的多边形,该对象类型为 ArcObjects 类型
    ESRI.ArcGIS.Geometry.IEnvelope agsComEnve = this.Geometry as ESRI.ArcGIS.
Geometry.IEnvelope;
    //得到编辑地图描述
    ESRI.ArcGIS.ADF.ArcGISServer.MapDescription agsSoapMapDesc = Editor.Map
Functionality.MapDescription;
    //得到地图选择要素的 ID 集合
    int[] OIDS = LayerDescription.SelectionFeatures;

    if (OIDS != null && OIDS.Length > 0)
    {
        ESRI.ArcGIS.Geometry.ITopologicalOperator agsComTop = null;
        ESRI.ArcGIS.Geodatabase.IFeature agsComFeatrue = null;

        try
        {
            //开始编辑
            this.StartEditOperation();
            //得到选择的 ArcObjects 要素集合
            ESRI.ArcGIS.Geodatabase.IFeatureCursor agsComFC = this.Feature
Layer.FeatureClass.GetFeatures(OIDS, false);

            agsComFeatrue = agsComFC.NextFeature();
            while (agsComFeatrue != null)
            {
                agsComTop = (ESRI.ArcGIS.Geometry.ITopologicalOperator)agsCom
Featrue;

                agsComTop.Clip(agsComEnve); //进行裁剪
                //得到裁剪结果
                ESRI.ArcGIS.Geometry.IGeometry agsComResult = (ESRI.ArcGIS.
Geometry.IGeometry)agsComTop;
                if (!agsComResult.IsEmpty)
                {
                    //保存结果
                    agsComFeatrue.Shape = agsComResult;
                    agsComFeatrue.Store();
                    IDsList.Add(agsComFeatrue.OID);
                }

                agsComFeatrue = agsComFC.NextFeature();
            }
            //结束编辑
            this.StopEditOperation();
        }
        catch (Exception ex)
        {

```

```
        //编辑异常处理
        AbortEditOperation(ex);
    }
}
```

程序运行效果如图 11.15 和图 11.16 所示。

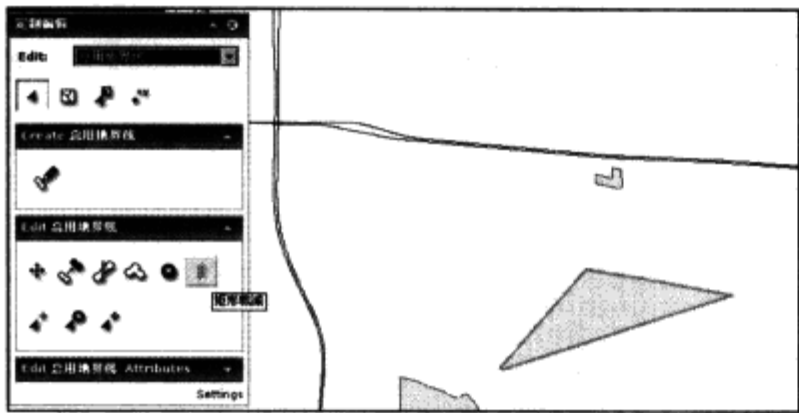


图 11.15 添加裁剪按钮

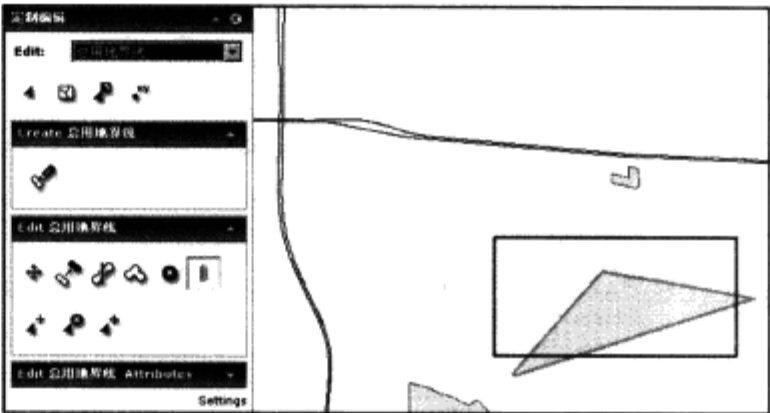


图 11.16 进行矩形裁剪

11.4 编辑属性数据

EditorTask 工具提供对空间要素的编辑，如图 11.17 所示。



图 11.17 属性编辑界面

用户直接在属性编辑面板上修改相关的属性即可完成对空间要素的编辑。然而，当用户不希望对空间对象几何位置进行修改，只需要对几何要素的某些字段进行修改时，如果还用 EditorTask 来对属性进行编辑，使用起来相对复杂。

其实，空间要素的属性在 Geodatabase 中就是关系表，既然是关系表我们就可以通过 SQL 来修改表字段的值，这与我们常用的数据编程没有什么两样，示例程序如下：

```
public DataSet GetDataset(string strSQL, out string strMessage)
{
    OracleConnection con = new OracleConnection(strConnection);
    strMessage = FlagSuccess;
```

```

OracleCommand cmd = new OracleCommand();
cmd.Connection = con;
DataSet ds = new DataSet();
OracleDataAdapter ad = new OracleDataAdapter();
cmd.CommandText = strSQL;
ad.SelectCommand = cmd;

try
{
    con.Open();
    ad.Fill(ds, "tablename");
    con.Close();
}
catch (OracleException e)
{
    con.Close();
    strMessage = e.Message;
}
catch (Exception e)
{
    con.Close();
    strMessage = e.Message;
}
finally
{
}
return ds;
}

```

编写 SQL 语句后，调用该方法就能修改空间要素的属性值：

```

strSQL = "Update gzldc_storeland t Set t.storeproj_code='" + txtProjCode.Text +
"',t.storeproj_name='" + txtProjName.Text + "',t.yearly='" + txtYear.Text + " Where
t.replyname = '" + temp[i] + "'";
GetDataset(strSQL,out strMessage);

```

11.5 小结

数据在线编辑，在某些时候确实非常有用，如用户野外现场调查时可直接把现场调查的数据修改到空间数据库中，不必像传统的那样把现场调查数据带回去进行处理再传回空间数据库。笔者认为在页面上的编辑不是太方便，鼠标的操作也不太灵活，在编辑空间数据时网页不适合复杂空间数据的编辑，只对一些简单的空间数据编辑可行，而复杂的空间数据编辑使用 ArcObjects 效率更高。

第 12 章

ArcGIS Server Web应用程序部署

开发一个 ArcGIS Server 的应用程序后，就需要考虑如何把编写好的应用程序发布给用户。ArcGIS Server 应用程序有两种开发环境：DOT.NET 和 Java，本章将介绍如何在.NET 环境下部署 ArcGIS Server 的应用程序。

12.1 应用程序部署环境

ArcGIS Server 产品包括两个部分：

- GIS Server，它是一个提供 GIS 服务的服务器产品，包括一系列核心 ArcObjects 库和管理这些 ArcObjects 组件的可伸缩的运行环境。
- ADF，即应用程序开发架构，它有 Java 和.NET 两种开发组件集，是用来开发和部署 GIS Server 的 Web 应用程序的产品，包括组件对象、Web 控件、Web 模板和开发帮助。它还有一个 Web 应用程序的 runtime，专门用于发布和部署使用 ADF 开发的 Web 程序，如 ASP.NET 等。

因此 ArcGIS Server 应用程序的部署环境必须有 Web 应用服务，如 IIS、TOMCAT、WebLogic 等，还需要安装 ADF 的 runtime。

12.2 IIS 中部署步骤

在 Visual Studio 中开发网站后，可以通过使用 Visual Studio 中的工具将 ArcGIS Server 应用程序复制到目标服务器或者预编译站点，并将输出文件复制到指定服务器。

“复制网站”工具类似 FTP 工具，可打开目标服务器上的文件夹，然后在当前网站和目标网站之间上传或下载文件。“复制网站”工具支持同步功能，该功能同时检查两个网站中的文件并自动确保两个网站都有最新版本的文件。

复制网站的步骤如下。

- (1) 在 Visual Studio 的项目文件中，用鼠标选中待发布的网站，单击鼠标右键，选择“Copy

Web Site”菜单，如图 12.1 所示。

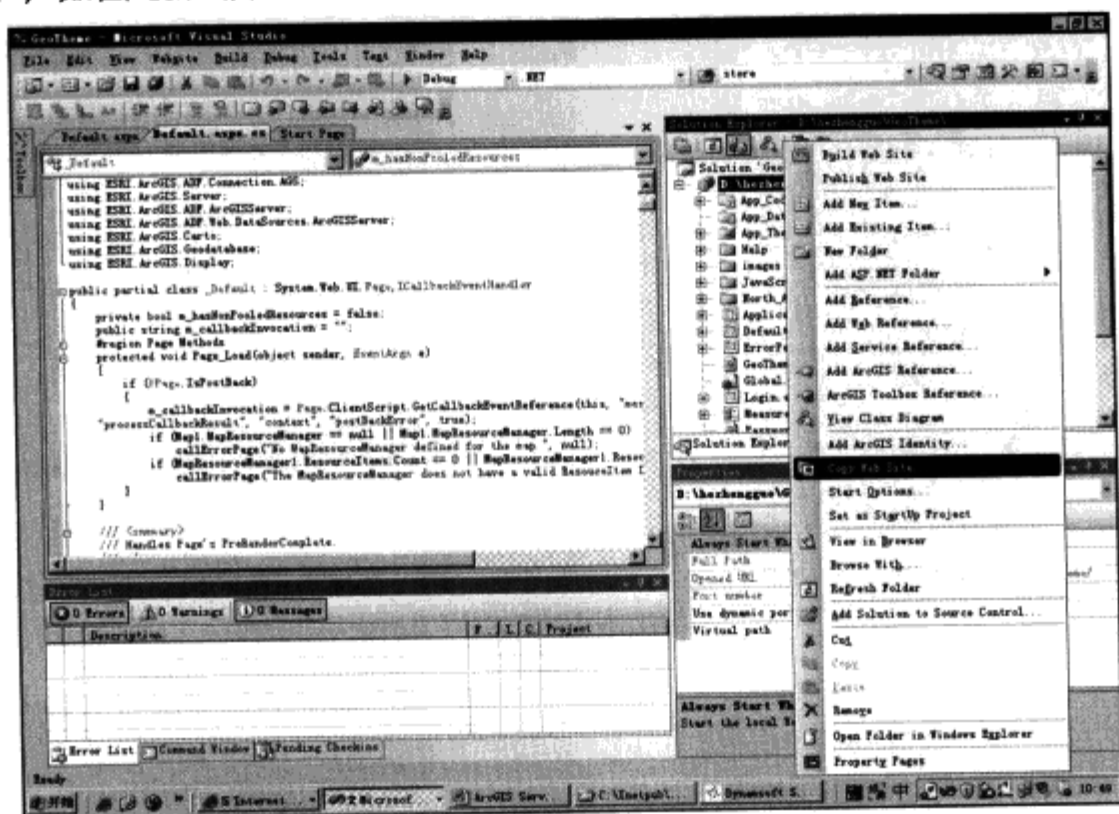


图 12.1 复制网站

(2) 系统弹出“Copy Web Site”页面，单击“Connect”按钮，系统弹出“Open Web Site”对话框，选择“File System”和目标目录，如图 12.2 所示。

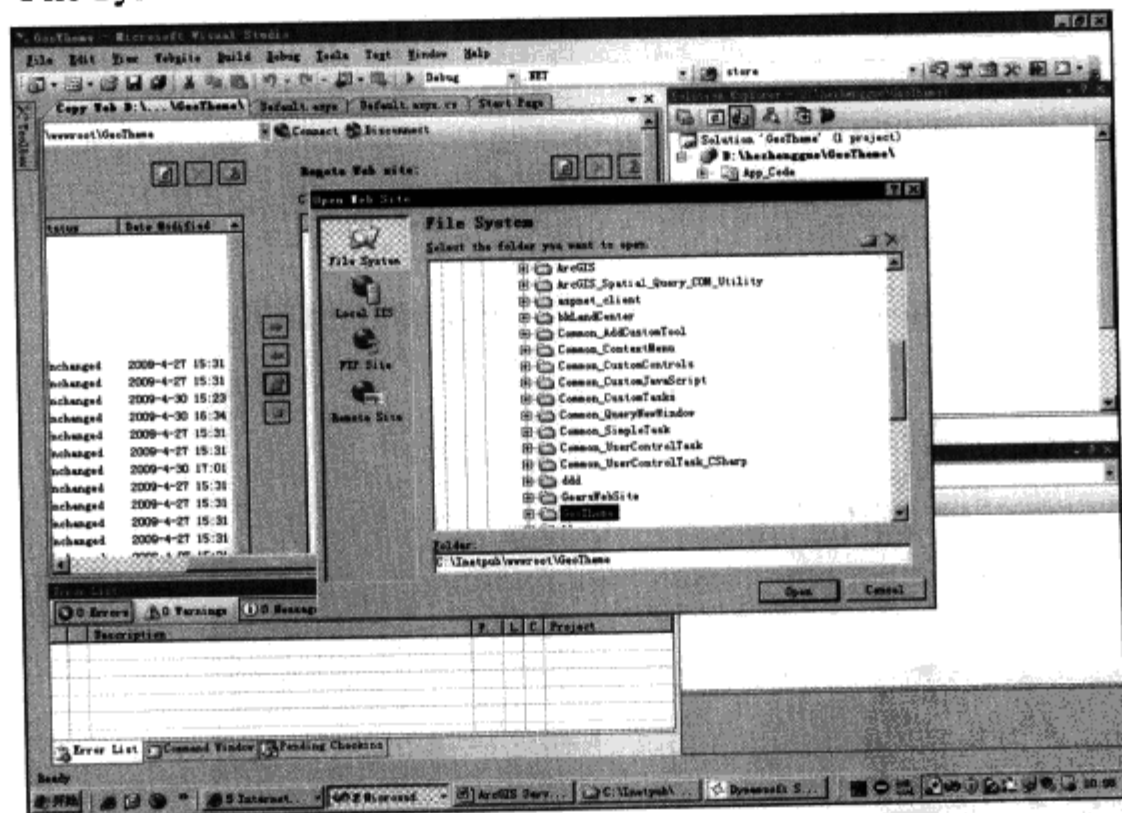


图 12.2 选择复制的目录

(3) 选中当前网站的文件，单击“同步”按钮，系统会自动更新当前网站与目标网站的文件，使两个网站的文件保持最新版本，如图 12.3 所示。

(4) 在 IIS 中打开刚才发布的目录，单击右键“属性”菜单，系统弹出“GeoTheme 属性”对话框，如图 12.4 所示。

(5) 单击“创建”按钮，单击“确定”按钮，这样一个基于 ArcGIS Server 的应用程序就发

布成功了。

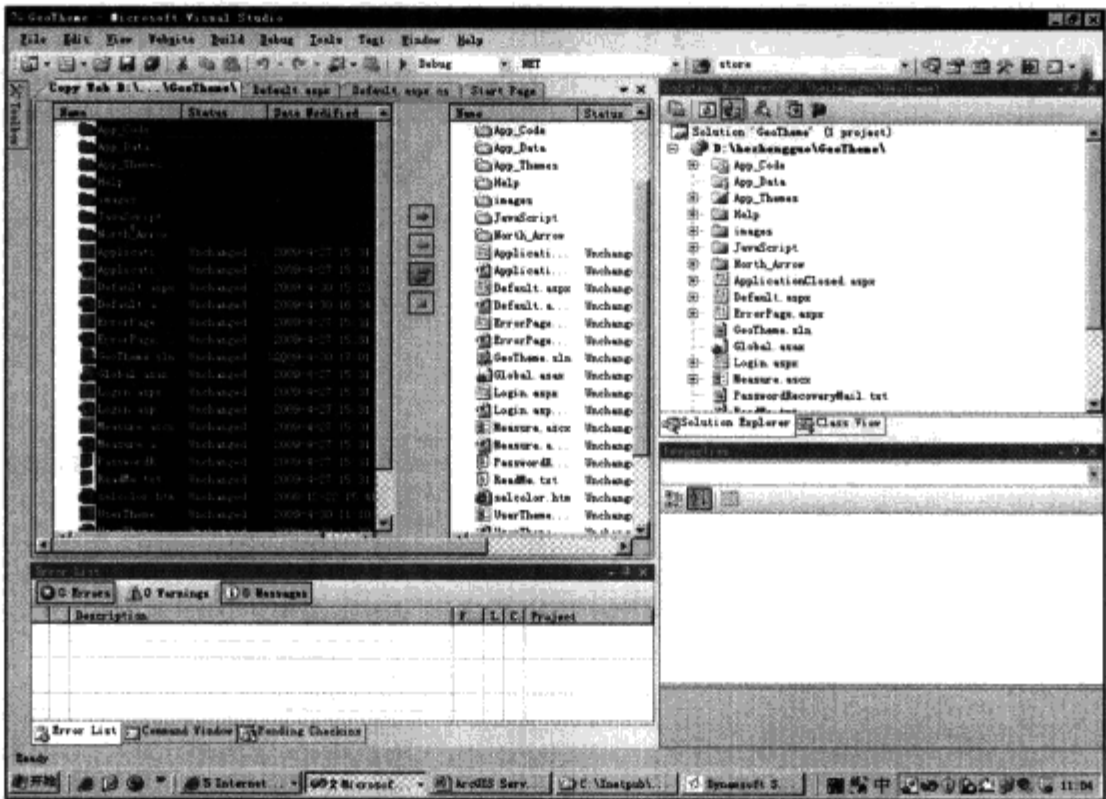


图 12.3 同步文件

使用“复制网站”工具，站点是按原样复制的，因此，如果任何文件包含编译错误，将直到运行引发错误的网页时才会发现错误，而且是直接源代码发布，没有任何加密措施。

除了用“复制网站”工具外，还可以用“发布网站”工具。“发布网站”使用“Publish Web Site”菜单工具预编译网站的内容，包括网页（.aspx）和代码（*.cs），然后将输出文件复制到指定的目录或服务器位置，可以作为预编译网站并从文件中去掉源代码，从而保留页面文件和一编译程序集的存根文件。在用户请求页面时，ASP.NET 用预编译的程序集执行请求。在 Web 应用程序项目中，将所有类文件编译到单个程序集中。

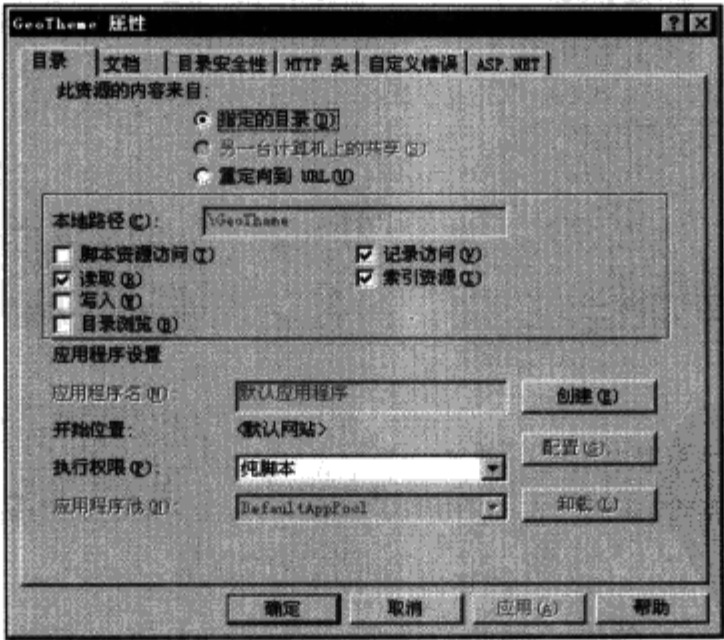


图 12.4 “GeoTheme 属性”对话框

发布网站的步骤如下：

(1)在 Visual Studio 的项目文件中，用鼠标选中待发布的网站，单击鼠标右键，选择“Publish Web Site”菜单，如图 12.5 所示。

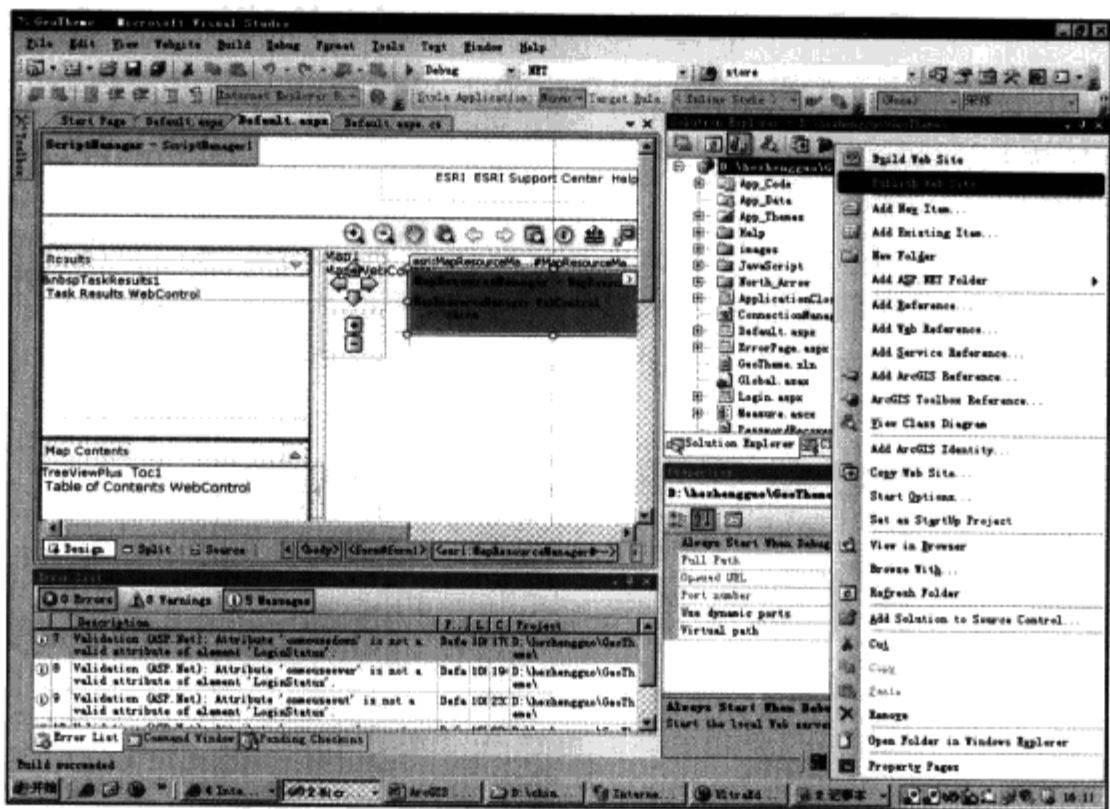


图 12.5 发布网站

(2) 系统弹出“Publish Web Site”页面，在“Target Location”文本框输入或者选择路径，单击“OK”按钮，系统就会把网站发布到相应的目录下，如图 12.6 所示。

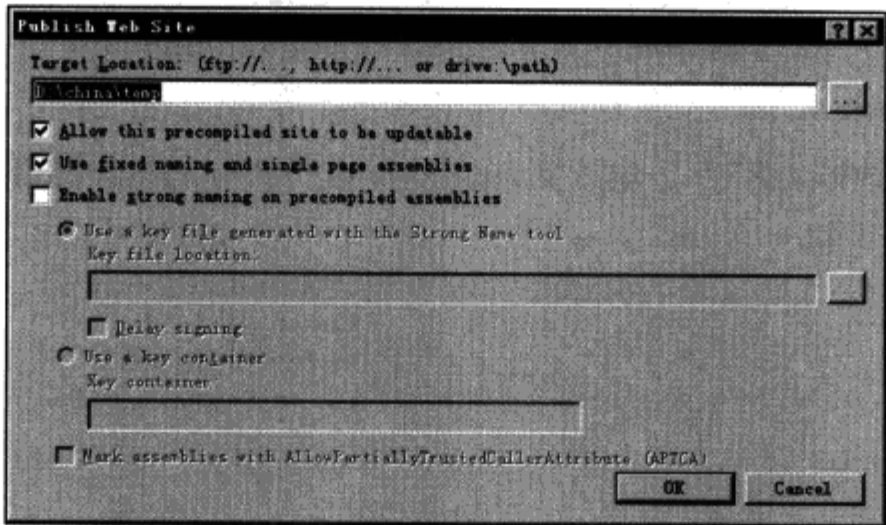


图 12.6 发布到相应的目录

(3) 有了应用程序的发布文件，剩下的工作就是在 IIS 中配置一个虚拟目录，在此就不再详述了。

使用“发布网站”工具，如果站点更新时需要重新编译该站点。因此，在改动频繁的站点不太适合这种发布方式。另外，它无法将已编译的站点部署到远程服务器上，只能复制到本地计算机或局域网的其他计算机上。

12.3 部署中的常见问题及解决方案

在发布网站的时候，经常碰到一些问题导致网站不能成功发布，网站部署实施过程中常见问题及解决方案总结如下。

(1) 在 ASP.NET 的默认安装过程中，ASP.NET 将在辅助进程中运行 Web 应用程序代码。默认情况下，此进程使用名为 ASPNET 账户（全名 aspnet_wp）的本地账户。ASP.NET 账户仅属于

该计算机上的用户组。因此，ASP.NET 账户具有与用户组关联的所有权限，并且可以访问用户组对其具有访问限的任何资源。ASP.NET 向 ASP.NET 账户授予对以下文件夹的完全访问权限。

- 临时 ASP.NET 文件。
- %windir%\temp。

如果发布的网站存在访问权限问题，开发人员应考虑 ASP.NET 用户的权限是否正确。

(2) 检查发布网站的配置版本是否正确，如图 12.7 所示。如图示 ASP.NET 1.1 版本，应选用 1.1 版本，2.0 要选择 2.0 版本，注意 Visual Studio 2008 是基于 ASP.NET 3.5 的，但在发布网站的时候还是用 2.0。

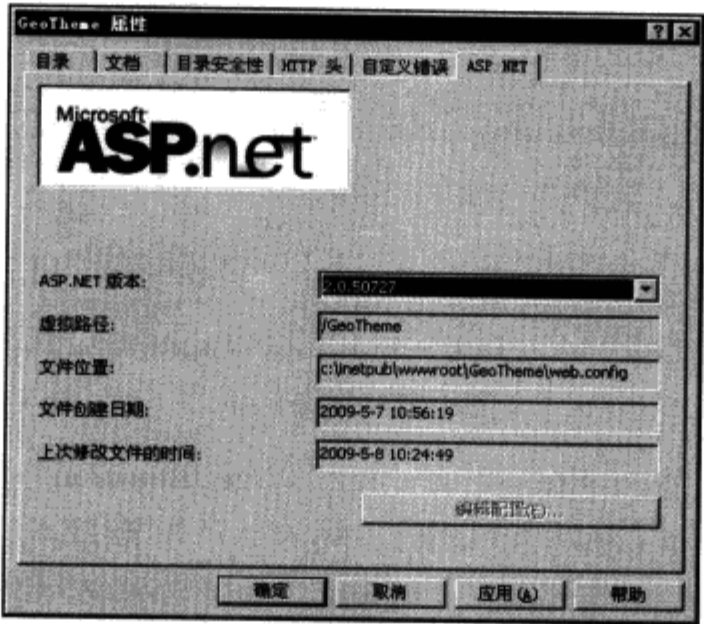


图 12.7 配置 ASP.NET 版本

(3) 应用程序的“执行权限”，应该选择“纯脚本”，“应用程序池”文本框中选择默认的“DefaultAppPool”，如图 12.8 所示。

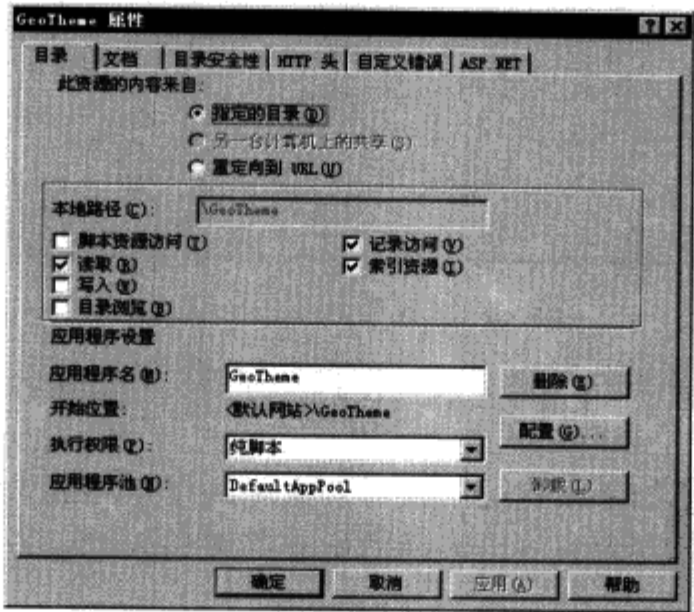


图 12.8 配置执行权限和应用程序池

(4) ASP.NET 身份验证模式。

ASP.NET 身份验证模式包括：Windows、Forms（窗体）、Passport（护照）和 None（无），如图 12.9 所示。

- Windows 身份验证 ASP.NET 依赖于 IIS 对用户进行验证，并创建一个 Windows 访问令牌来表示通过验证的标识。IIS 提供以下几种身份验证机制：基本身份验证、简要身份验证、

- 集成 Windows 身份验证、证书身份验证、匿名身份验证。
- 护照身份验证 ASP.NET 使用 Microsoft Passport 的集中式身份验证服务，ASP.NET 为 Microsoft Password 软件开发包（SDK）所提供的功能提供了一个方便的包装（Wrapper）。此 SDK 必须安装在 Web 服务器上。
 - 窗体身份验证 使用客户端重定向功能，将通过身份验证的用户转化到特定的用户登录窗体，要求用户输入其凭据信息（用户名和密码）。这些凭据信息被验证后，系统生成一个身份验证票证（ticket）并将其返回客户端。身份验证票证可在用户的会话期间维护用户的身份标识信息，以及用户所属列表（可选）。
 - None 不希望对用户进行验证，或采用自定义的身份验证协议。
- 在部署 ASP.NET 网站时，要清楚本网站采用的身份验证方式并进行正确地配置。

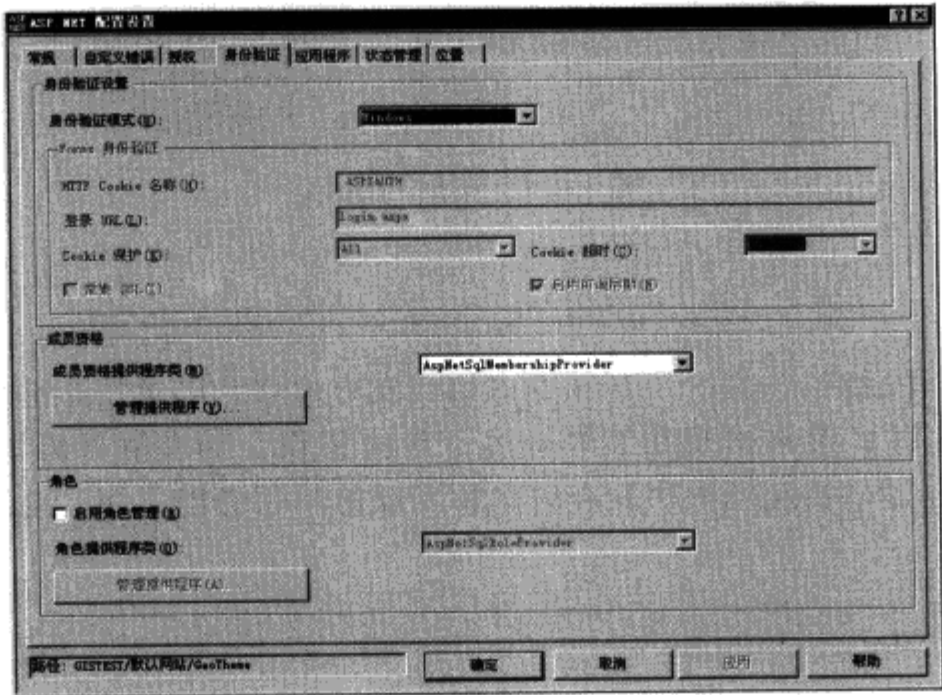


图 12.9 ASP.NET 身份验证模式

(5) MapResourceManager 采用 ArcGIS Server Local 连接方式。

在 Visual Studio 2008 编译环境下，应用程序运行正常，并能发布到 IIS 中，其他的配置都正常，但是访问却不成功，系统出现的错误信息如图 12.10 所示。

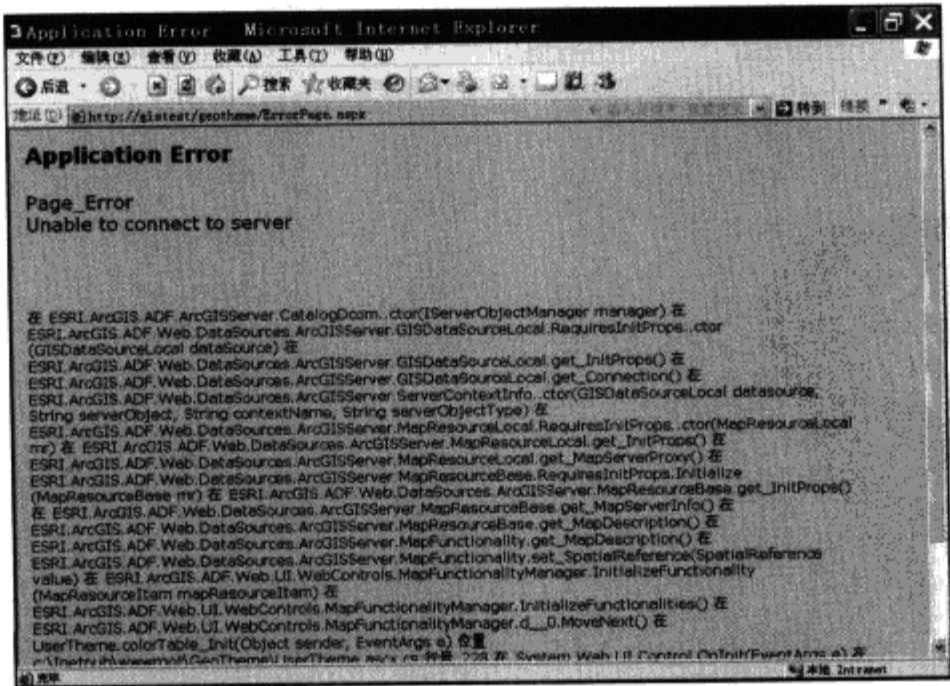


图 12.10 ArcGIS Server Local 连接服务错误

解决方法是，添加一个 Identity，然后重新发布，应用程序就发布成功了。添加 Identity 的方法是选中待发布的网站，单击鼠标右键，选择“Add ArcGIS Identity”菜单，系统弹出“Add ArcGIS Identity”对话框，在相应的文本框中输入对应的值，单击“OK”按钮，完成 Identity 的添加，如图 12.11 所示。

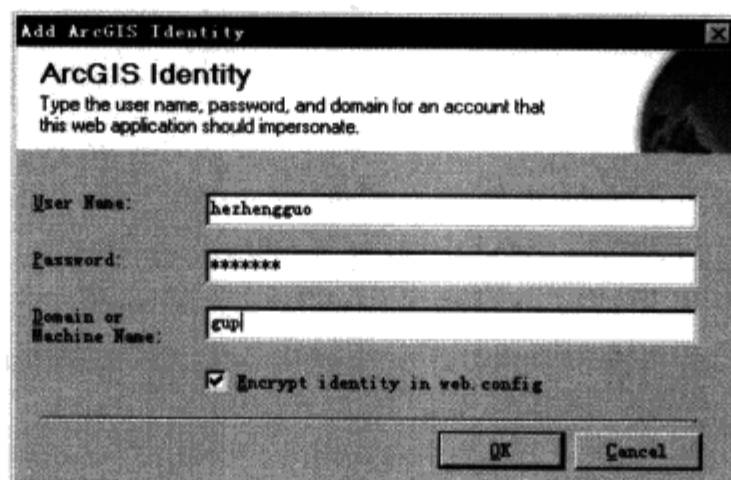


图 12.11 添加 Identity

12.4 小结

本章主要讲述编译后的 ArcGIS Server 程序如何发布部署，以及发布部署应该注意的一些问题。在部署应用程序时，常常出现在编译环境下没有错误，而在发布部署的时候出现各种错误，本章将 IIS 中 ASP.NET 应用程序发布时经常出现的问题进行了简单总结，希望对读者能够有帮助。

第四篇

ArcGIS Server

高级功能开发

- ▶ 第 13 章 ArcGIS Server ADF For .NET 高级功能
- ▶ 第 14 章 ArcGIS Server 9.3 开发模式
- ▶ 第 15 章 ArcGIS Server For Java
- ▶ 第 16 章 ArcGIS Server 性能优化

本部分主要讲述 ArcGIS Server 的高级功能。ArcGIS 是一个平台软件，提供的功能非常强大，而且不同的产品（工具）之间是可以互操作的，甚至还可以调用其他公司的 GIS 服务。

第 13 章

ArcGIS Server ADF For .NET高级功能

前面的章节介绍了 ArcGIS Server ADF For .NET 的通用功能开发, 这些功能是开发人员在开发项目中经常遇到的。本章主要介绍 ADF For .NET 的高级功能开发, 有些功能需要调用 ArcObjects 来实现。

13.1 一般服务器对象扩展

前面的章节我们已经介绍过 ArcGIS Server Web For .NET 应用程序中可以调用 ArcObjects 的函数来扩展 ArcGIS Server Web 应用程序的功能。本节要介绍的服务器对象扩展是 ArcGIS Server Web 应用程序调用基于 ArcObjects 编写的 Class Library。

用 ASP 开发的应用程序可以调用其他程序语言开发的 ActiveX; C#开发的.NET 应用程序可以调用 C#编写的 COM 组件。服务器对象的扩展其实就是调用 ArcObjects 开发的 COM 组件。下面介绍一般服务器对象的扩展方法。

实现步骤如下。

(1) 首先编写基于 ArcObjects 的 COM 程序。用 C#开发建立一个 Class Library 项目, 如图 13.1 所示。

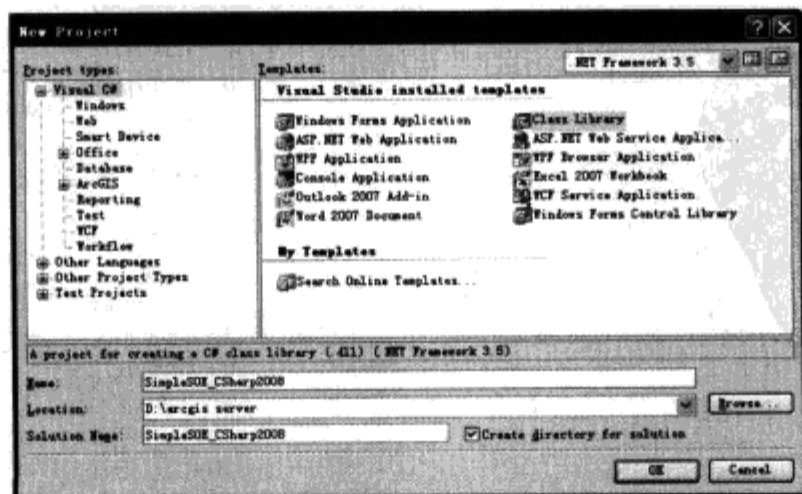


图 13.1 建立 Class Library 项目

(2) 完成基于 ArcObjects 的功能开发, 本例主要是用 ArcObjects 来实现一个点的 Buffer。实例程序如下:

```
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Server;

[GuidAttribute("18bc771b-59eb-4fb8-ae67-264dee546fcf")]
public interface IUtilSOE_CSharp
{
    //定义接口
    IGeometry ReturnCircle(IPoint inpoint, double radius);
}

namespace SimpleSOE_CSharp
{
    [AutomationProxy(true), ClassInterface(ClassInterfaceType.None), GuidAttribute("a9ed9e17-d2b2-461c-9da3-2915af9a3f00")]
    public class UtilSOE_CSharp : ServicedComponent, IUtilSOE_CSharp, IServerObjectExtension
    {
        private IServerObjectHelper m_SOH;
        //实现接口
        public IGeometry ReturnCircle(IPoint inpoint, double radius)
        {
            ESRI.ArcGIS.Geometry.ITopologicalOperator topop = (ESRI.ArcGIS.Geometry.ITopologicalOperator)inpoint;
            ESRI.ArcGIS.Geometry.IGeometry circle = topop.Buffer(radius);
            return circle;
        }

        #region IServerObjectExtension Members

        public void Init(IServerObjectHelper pSOH)
        {
            m_SOH = pSOH;
        }

        public void Shutdown()
        {
            m_SOH = null;
        }

        #endregion
    }
}
```

(3) 编写上面的项目，生成 DLL。注意该 DLL 必须注册后才能使用，它所使用的注册方法与 Regsvr32 不同，注册 C# 编写的 COM 用 RegAsm，命令程序如下：

```
regasm SimpleSOE_CSharp.dll /codebase
```

(4) 发布地图服务和服务器对象扩展，如图 13.2 所示。

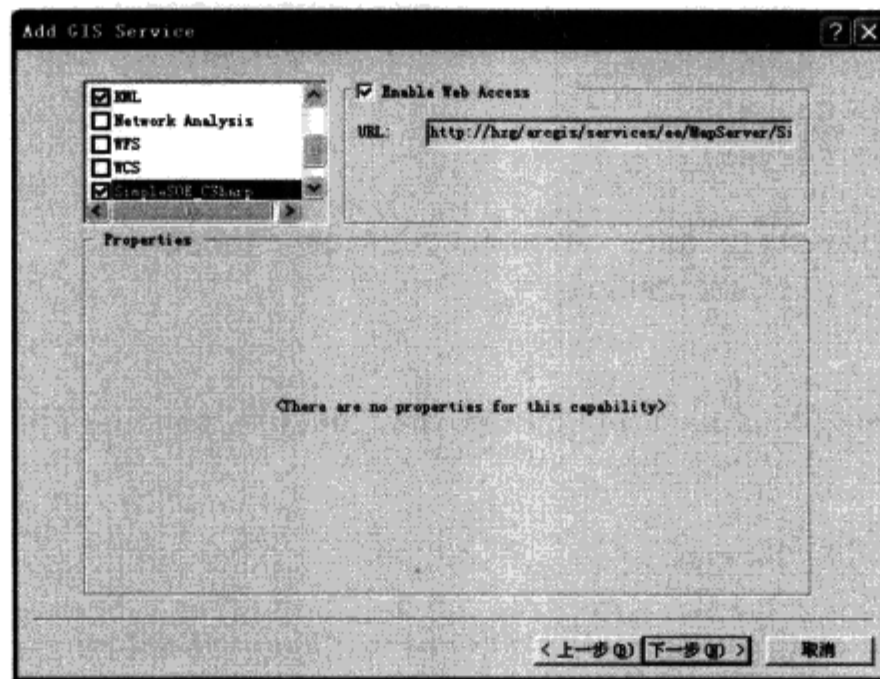


图 13.2 发布地图服务和服务器扩展对象界面

(5) 编写 ArcGIS Server Web 应用程序来调用服务器扩展对象提供的方法：

```
public void ServerAction(ToolEventArgs args)
{
    ESRI.ArcGIS.ADF.Web.UI.WebControls.Map mapctrl = (ESRI.ArcGIS.ADF.Web.
    UI.WebControls.Map) args.Control;

    PointEventArgs pntargs = (PointEventArgs) args;
    System.Drawing.Point mypnt = pntargs.ScreenPoint;

    ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.MapFunctionality mf = (ESRI.
    ArcGIS.ADF.Web.DataSources.ArcGISServer.MapFunctionality) mapctrl.GetFunctionality(
    mapctrl.MapResourceManagerInstance.ResourceItems.Count - 1);
    ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.MapResourceLocal mr = (ESRI.
    ArcGIS.ADF.Web.DataSources.ArcGISServer.MapResourceLocal) mf.MapResource;
    IServerContext sc = mr.ServerContextInfo.ServerContext;
    IMapServer ms = mr.MapServer;

    IImageDisplay imageDisp = (ImageDisplay) sc.CreateObject("esriCarto.
    ImageDisplay");
    imageDisp.Height = (int) mapctrl.Height.Value;
    imageDisp.Width = (int) mapctrl.Width.Value;
    imageDisp.DeviceResolution = mf.DisplaySettings.ImageDescriptor.Dpi;

    ESRI.ArcGIS.Carto.MapDescription md = (ESRI.ArcGIS.Carto.MapDescription)
    ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.Converter.ValueObjectToComObject(mf.M
    apDescription, sc);
    IPoint pnt = ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.Converter.
```

```

ToMapPoint(sc, ms, md, imageDisp, mypnt.X, mypnt.Y);

    IServerObjectExtensionManager soexm = (IServerObjectExtensionManager)ms;
    //查找服务器扩展对象
    IServerObjectExtension soext = soexm.FindExtensionByTypeName("SimpleSOE_
CSharp");
    IUtilSOE_CSharp utilsoe = (IUtilSOE_CSharp)soext;

    double radius = mapctrl.Extent.Width * 0.1;
    //调用服务器方法提供的接口, 实现对服务器的调用
    IPolygon circle = (IPolygon) utilsoe.ReturnCircle(pnt, radius);
    circle.Densify(0.1, 0.01);

    IEnumerable gfc = mapctrl.GetFunctionalities();
    ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource gResource = null;
    foreach (IGISFunctionality gfunc in gfc)
    {
        if (gfunc.Resource is ESRI.ArcGIS.ADF.Web.DataSources.Graphics.
MapResource)
        {
            gResource = (ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource)
gfunc.Resource;
            break;
        }
    }

    if (gResource == null)
        return;

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

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

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

    ESRI.ArcGIS.ADF.Web.Geometry.Point[] adf_parray = ESRI.ArcGIS.ADF.Web.

```



```

DataSources.ArcGISServer.Converter.FromIPointCollection((ESRI.ArcGIS.Geometry.IPointCollection)circle);
    ESRI.ArcGIS.ADF.Web.Geometry.PointCollection adf_pc = new ESRI.ArcGIS.ADF.Web.Geometry.PointCollection();
    for (int i = 0; i < adf_parray.Length - 1; i++)
    {
        adf_pc.Add(adf_parray[i]);
    }
    ESRI.ArcGIS.ADF.Web.Geometry.Ring adf_ring = new ESRI.ArcGIS.ADF.Web.Geometry.Ring();
    adf_ring.Points = adf_pc;
    ESRI.ArcGIS.ADF.Web.Geometry.RingCollection adf_rcol = new ESRI.ArcGIS.ADF.Web.Geometry.RingCollection();
    adf_rcol.Add(adf_ring);
    ESRI.ArcGIS.ADF.Web.Geometry.Polygon adf_pg = new ESRI.ArcGIS.ADF.Web.Geometry.Polygon();
    adf_pg.Rings = adf_rcol;

    ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement ge = new ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement(adf_pg, System.Drawing.Color.Yellow);
    ge.Symbol.Transparency = 70.0;
    glayer.Add(ge);

    if (mapctrl.ImageBlendingMode == ImageBlendingMode.WebTier)
    { mapctrl.Refresh(); }
    else if (mapctrl.ImageBlendingMode == ImageBlendingMode.Browser)
    { mapctrl.RefreshResource(gResource.Name); }
}

```

程序运行结果如图 13.3 所示。

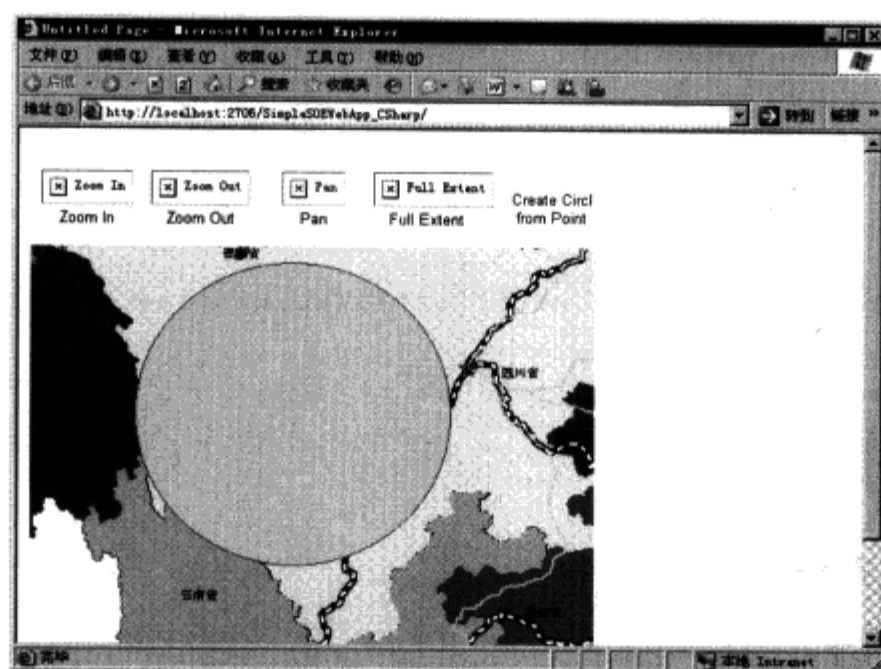


图 13.3 服务器对象扩展运行效果

13.2 Geodata 的签入签出

在第 12 章中讲过, ArcGIS Server 提供在线编辑功能, 在线编辑只适合那些功能简单的编辑,

对于复杂的数据编辑，ArcGIS Server 在线编辑无论是编辑效率还是操作方便性方面均不能让人满意。对于复杂的数据编辑可以采用另外一种方式：先将需要编辑的数据下载到 Personal Geodatabase 中，再将下载的数据用 ArcMap 或其他的工具进行编辑，通常这些编辑工具的效率比较高，把最后编辑的数据签入数据库完成对数据的修改。这种功能对于那些需要野外调查的数据修改非常有用，也可以是数据的离线编辑。

下面就来详细介绍 ArcGIS Server 的 Geodata 的签出和签入。

(1) 首先需要发布一个 Geodata Service，如图 13.4 所示。

Add New Service - General

This wizard lets you add a new service to the GIS server.

Name:

Type:

Description:

☒ Restart this service automatically whenever ArcGIS Server restarts.

图 13.4 发布 Geodata 服务

(2) 配置服务的物理和虚拟路径，如图 13.5 所示。

Add New Service - Parameters

Create a Geodata Service using: ☐ A geodatabase ☒ A map

Map Document:

Type in the location of the resource. If you want to browse to a location, only shared drives appear in the list.

Data Frame:

Output Directory:

Virtual Output Directory:

图 13.5 配置相关路径

(3) 地图发布完成后，编写相关代码，完成对当前视窗范围内数据的签出，示例程序如下：

```
MapFunctionality mf = (MapFunctionality) Map1.GetFunctionality(0);

// 得到 server context and server object
MapResourceLocal mr_local = (MapResourceLocal)mf.MapResource;
IServerContext ctx = mr_local.ServerContextInfo.ServerContext;

IMapServer ms = (IMapServer)ctx.ServerObject;
IMapServerObjects mso = (IMapServerObjects)ms;

// 得到数据框架的名字
//让地图与服务器对象关联
MapInformation mi = (MapInformation)mr_local.MapInformation;
String dataFrame = mi.DataFrame;
IMap map = mso.get_Map(dataFrame);
```

```

        //创建一个描述签出数据的 GP Replica Descripton 对象
        IGPReplicaDescription gpRepDesc = (IGPReplicaDescription)ctx.CreateObject(
("esriGeodatabase.GPReplicaDescription");
        gpRepDesc.ModelType = ESRI.ArcGIS.Geodatabase.esriReplicaModelType.esri
ModelTypeFullGeodatabase;
        gpRepDesc.SingleGeneration = true;
        gpRepDesc.TransferRelatedObjects = true;

        // 把当前显示范围内的数据签出

        gpRepDesc.QueryGeometry = ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.
Converter.ToIGeometry(Map1.Extent, ctx);
        gpRepDesc.SpatialRelation = ESRI.ArcGIS.Geodatabase.esriSpatialRelEnum.
esriSpatialRelIntersects;

        // 创建签出数据的数据集
        IGPReplicaDatasets resds = (IGPReplicaDatasets) ctx.CreateObject
("esriGeodatabase.GPReplicaDatasets");

        // 得到地图中的层的集合
        ESRI.ArcGIS.esriSystem.UID ltype = (ESRI.ArcGIS.esriSystem.UID) ctx.
CreateObject("esriSystem.UID");
        ltype.Value = "{6CA416B1-E160-11D2-9F4E-00C04F6BC78E}";

        IEnumLayer el = map.get_Layers(ltype, true);
        el.Reset();

        ILayer l = null;
        IFeatureLayer fl = null;
        IDataset ds = null;

        while ((l = el.Next()) != null)
        {
            if (l is IFeatureLayer)
            {
                fl = (IFeatureLayer)l;
                ds = (IDataset)fl.FeatureClass;

                // 创建 feature class 的描述
                IGPReplicaDataset rd = ctx.CreateObject("esriGeodatabase.
GPReplicaDataset") as IGPReplicaDataset;
                rd.Name = ds.Name;
                rd.DatasetType = ESRI.ArcGIS.Geodatabase.esriDatasetType.esriDT
FeatureClass;
                rd.RowsType = ESRI.ArcGIS.Geodatabase.esriRowsType.esriRowsType
Filter;
                rd.UseGeometry = true;

                //加入签出数据集中
                resds.Add(rd);
            }
        }

```

```

    }

    // 设置复制描述
    gpRepDesc.ReplicaDatasets = resds;

    ESRI.ArcGIS.GeoDatabaseDistributed.IGeoDataServer gs = CreateGeoDataServer
    InMapsContext(mf);

    // 设置被签出数据的格式
    ESRI.ArcGIS.GeoDatabaseDistributed.IGDSEExportOptions exo;
    exo = (ESRI.ArcGIS.GeoDatabaseDistributed.IGDSEExportOptions)ctx.Create
    Object("esriGeoDatabaseDistributed.GDSEExportOptions");
    exo.ExportFormat = ESRI.ArcGIS.GeoDatabaseDistributed.esriGDSEExportFormat.
    esriGDSEExportFormatPersonalGdb;

    // 签出数据, 并转到签出数据的 URL

    ESRI.ArcGIS.GeoDatabaseDistributed.IGDSData gsdata = null;
    try
    {
        gsdata = gs.CreateReplica(gs.DefaultWorkingVersion, Session.SessionID,
        gpRepDesc, null, exo, ESRI.ArcGIS.GeoDatabaseDistributed.esriGDSTransportType.
        esriGDSTransportTypeUrl);
        Response.Redirect(gsdata.URL, false);
    }
    catch (Exception ex)
    {
        this.lblInfo.Text = ex.Message;
    }
}

```

(4) 在 ArcMap 中编辑签出的数据, 并把编辑后发生变化的数据导出, 如图 13.6 所示。

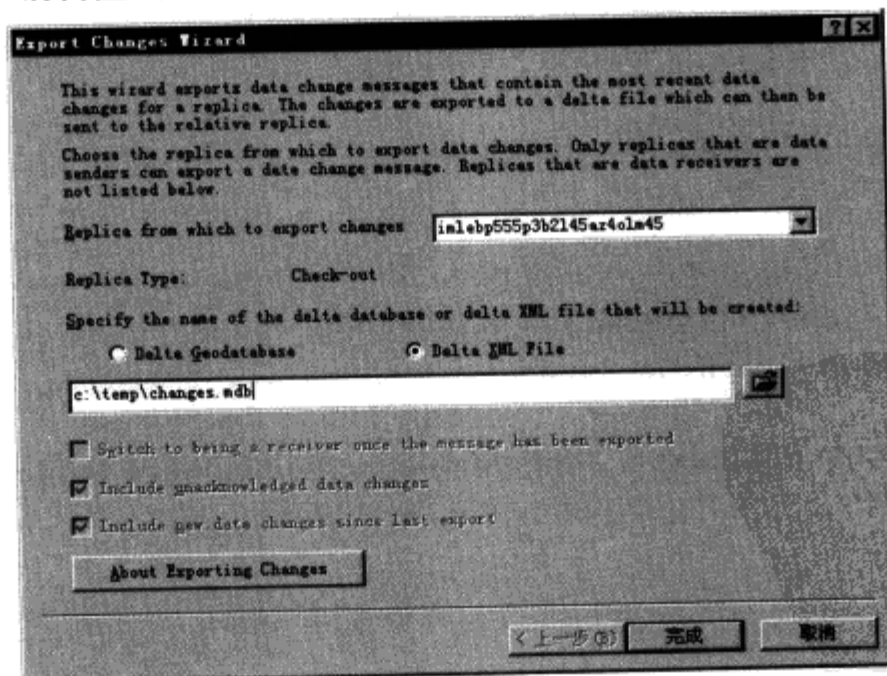


图 13.6 导出数据

(5) 将编辑后的数据签入, 实现原始空间数据的修改, 如图 13.7 所示。

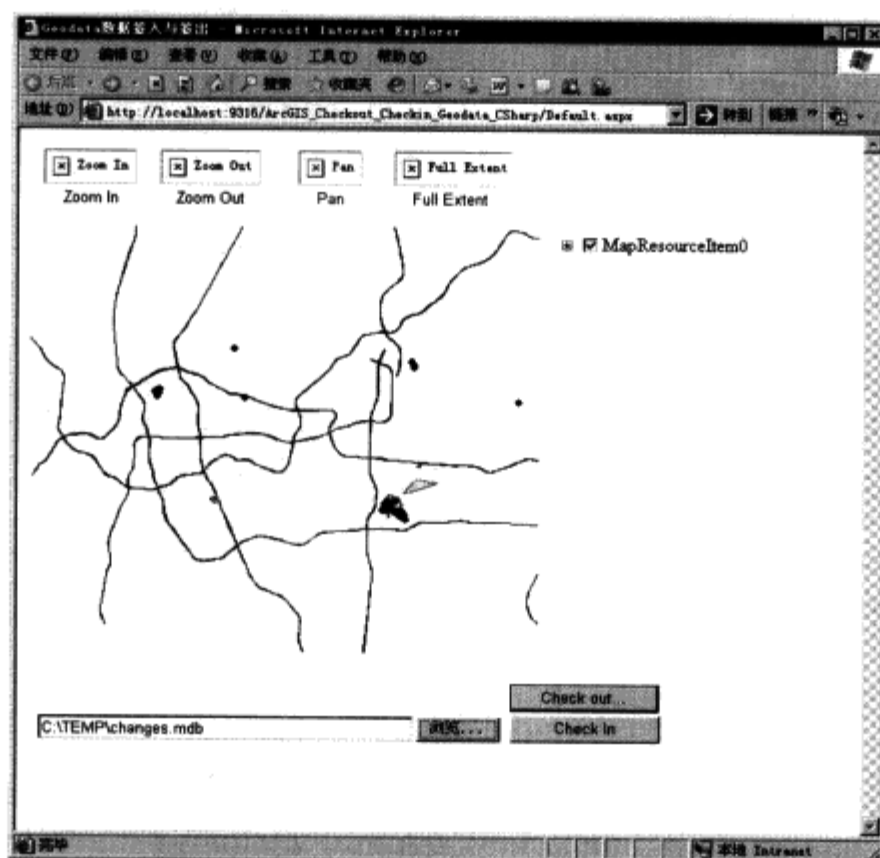


图 13.7 签入编辑后发生改变的数据

签入的实例程序如下：

```
HttpPostedFile postedFile = inFile.PostedFile;

//没有输入文件
if (postedFile == null)
    return;

//文件为空退出
int size = postedFile.ContentLength;
if (size <= 0)
{
    this.lblInfo.Text = "File has no data.";
    return;
}

// 上传文件必须为 Personal GeoDatabase
if (System.IO.Path.GetExtension(postedFile.FileName) != ".mdb")
{
    this.lblInfo.Text = "Error: can only use delta personal geodatabases.";
    return;
}
else
    this.lblInfo.Text = "";

//把文件保存到临时目录
string fn = System.IO.Path.GetTempFileName();
string fn2 = System.IO.Path.ChangeExtension(fn, ".mdb");
postedFile.SaveAs(fn2);
```

```

//创建 geodata server
IEnumerable ienum = Map1.GetFunctionalities();

MapFunctionality mf = null;

foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gisfunc in ienum)
{
    if (gisfunc is MapFunctionality)
    {
        mf = (MapFunctionality)gisfunc;
        break;
    }
}
MapResourceLocal mr = (MapResourceLocal)mf.MapResource;
IServerContext ctx = mr.ServerContextInfo.ServerContext;

try
{
    //实例化数据
    System.IO.FileStream f = new System.IO.FileStream(fn2, System.IO.
FileMode.Open);
    byte[] bytes = new byte[f.Length];
    f.Read(bytes, 0, (int)f.Length);

    ESRI.ArcGIS.GeoDatabaseDistributed.IGDSData data = new ESRI.ArcGIS.
GeoDatabaseDistributed.GDSDataClass();
    data.TransportType = ESRI.ArcGIS.GeoDatabaseDistributed.esriGDSTransport
Type.esriGDSTransportTypeEmbedded;
    data.set_EmbeddedData(ref bytes);

    ESRI.ArcGIS.GeoDatabaseDistributed.IGeoDataServer gs = CreateGeoData
ServerInMapsContext(mf);

    //签入变化后的数据
    gs.ImportReplicaDataChanges(ESRI.ArcGIS.GeoDatabaseDistributed.esriGDSReplicaImport
Source.esriGDSReplicaImportSourceDeltaPersonalGDB,
ESRI.ArcGIS.Geodatabase.esriReplicaReconcilePolicyType.esriReplicaResolveConflicts
InFavorOfImportedChanges, false,
    data);
}

catch (Exception exception)
{
    this.lblInfo.Text = "Check In Dialog - Error executing check in request:
" + exception.Message;
}

```

通过空间数据的签出 (Checkout) 和签入 (Checkin) 可以实现数据的离线编辑。签出的数

据往往只是部分数据，而且是在 C/S 架构下进行编辑，这比在线编辑的效率更高，也非常适合需要野外调查的数据编辑。

13.3 空间查询

在前面的章节中，经常讲到 ArcGIS Server 的空间查询，不过那些空间查询都是 ArcGIS Server 本身的 API 来实现的。本节要讲述的是另外一种实现空间查询的方法，即基于 ArcObjects 的函数封装空间查询的业务逻辑，然后在 ArcGIS Server Web For .NET 中调用空间查询的业务逻辑，最终实现空间查询。

下面就来介绍基于 ArcObjects 封装一个 DLL 供 ArcGIS Server 调用的方法，该方法与 13.1 节中方法的实现步骤基本类似，这里就不再详细介绍步骤。

(1) 基于 ArcObjects 封装一个 DLL，该 DLL 利用 ArcObjects 来实现空间查询，示例程序如下：

```
public IVegResults sumVegetationType(ref IFeatureClass pVegClass, ref IPoint
pPoint, ref double dDistance, ref string sSummaryFld)
{
    // 生成点的 Buffer
    ITopologicalOperator pTopoOp = pPoint as ITopologicalOperator;
    IGeometry pGeom = pTopoOp.Buffer(dDistance);

    //空间查询
    ISpatialFilter pSFilter = new SpatialFilter();
    pSFilter.Geometry = pGeom;
    pSFilter.SpatialRel = esriSpatialRelEnum.esriSpatialRelIntersects;
    pSFilter.GeometryField = pVegClass.ShapeFieldName;

    IFeatureCursor pFCursor = pVegClass.Search(pSFilter, true);

    pTopoOp = pGeom as ITopologicalOperator;
    int lPrim = pVegClass.FindField(sSummaryFld);

    System.Collections.Specialized.ListDictionary dict = new System.Collections.
Specialized.ListDictionary();

    // 生成符号
    ISimpleFillSymbol pSFS = new FillS();
    IGraphicElements pGraphics = new GraphicElements();

    IFeature pFeature;
    while ((pFeature = pFCursor.NextFeature()) != null)
    {
        // create the graphic
        IFillShapeElement pFE = new PolygonElement() as IFillShapeElement;
        IElement pElement = pFE as IElement;

        // 裁剪几何要素
        IGeometry pNewGeom = pTopoOp.Intersect(pFeature.Shape, esriGeometryDimension.
esriGeometry2Dimension);
```



```

pElement.Geometry = pNewGeom;
pFE.Symbol = pSFS;
IGraphicElement ge = pFE as IGraphicElement;
pGraphics.Add(ge);

// add to dictionary
IArea pArea = pNewGeom as IArea;
string sType = pFeature.get_Value(1Prim) as string;
if(dict.Contains(sType))
    dict[sType] = (double)dict[sType] + pArea.Area;
else
    dict[sType] = pArea.Area;
}

```

(2) 编译该项目生成 DLL，并注册该 COM，命令如下：

```
regasm VegCOMCSharp.dll /tlb:VegCOMCSharp.tlb /codebase
```

(3) 在 ArcGIS Server Web 应用程序中调用该 COM，实例程序如下：

```

ESRI.ArcGIS.ADF.Web.UI.WebControls.Map mapctrl = (ESRI.ArcGIS.ADF.Web.UI.Web
Controls.Map) args.Control;

// 从 Map 控件得到 MapFunctionality
MapFunctionality mapfunc = (MapFunctionality) mapctrl.GetFunctionality
(mapctrl.MapResourceManagerInstance.ResourceItems.Count - 1);
MapResourceLocal mapres = (MapResourceLocal) mapfunc.MapResource;
IServerContext sc = mapres.ServerContextInfo.ServerContext;
IMapServer map = mapres.MapServer;

IMapServerObjects mapobj = (IMapServerObjects) map;
IMap fgmap = mapobj.get_Map(map.DefaultMapName);
IFeatureLayer fl = (IFeatureLayer) fgmap.get_Layer(0);
IFeatureClass fc = fl.FeatureClass;

PointEventArgs pargs = (PointEventArgs) args;
ESRI.ArcGIS.ADF.Web.Geometry.Point inpt = ESRI.ArcGIS.ADF.Web.Geometry.
Point.ToMapPoint(pargs.ScreenPoint, mapctrl.Extent, (int)mapctrl.Width.Value, (int)
mapctrl.Height.Value);
IPoint pt = (IPoint)sc.CreateObject("esriGeometry.Point");
pt.X = inpt.X;
pt.Y = inpt.Y;

string tbxvalue = (string) mapctrl.Page.Session["TextBox1Value"];

double distance = 0;

if (!Double.TryParse(tbxvalue, out distance))
{
    distance = 10000;
}

string fldName = "Name";
//找到 COM 并调用其接口
IVegUtils vegutils = (IVegUtils) sc.CreateObject("VegCOMCSharp.VegUtils");

```

```
IVegResults vegresults = vegutils.sumVegetationType(ref fc, ref pt, ref distance, ref fldName);
```

如果仅仅是实现简单的空间查询，ArcGIS Server ADF 本身提供的函数足够使用，而且相对简单；利用 ArcObjects 封装 COM 是复杂的空间查询业务逻辑，可扩展性更强，但是其使用相对复杂。

13.4 Geoprocessing 缓冲

获取空间对象的缓冲区，在 GIS 应用程序的开发过程中经常遇到，13.1 节中介绍利用 ArcObjects 封装 COM 来扩展 Server Objects，就可以很容易得到几何要素的缓冲区。本节将介绍另一种方法，即发布一个缓冲地理处理服务来获取缓冲区。

下面将详细介绍如何利用 Geoprocessing 来处理缓冲区。

(1) 利用 ArcGIS 的 ModelBulider 工具建立缓冲区地理处理模型，如图 13.8 所示。

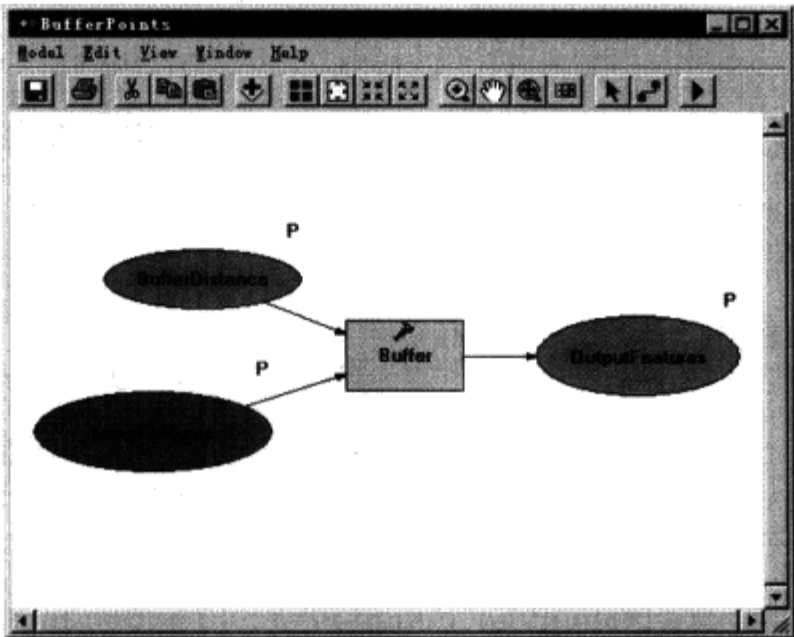


图 13.8 建立缓冲区的处理模型

(2) 发布地理处理服务 BufferPoints，如图 13.9 所示。

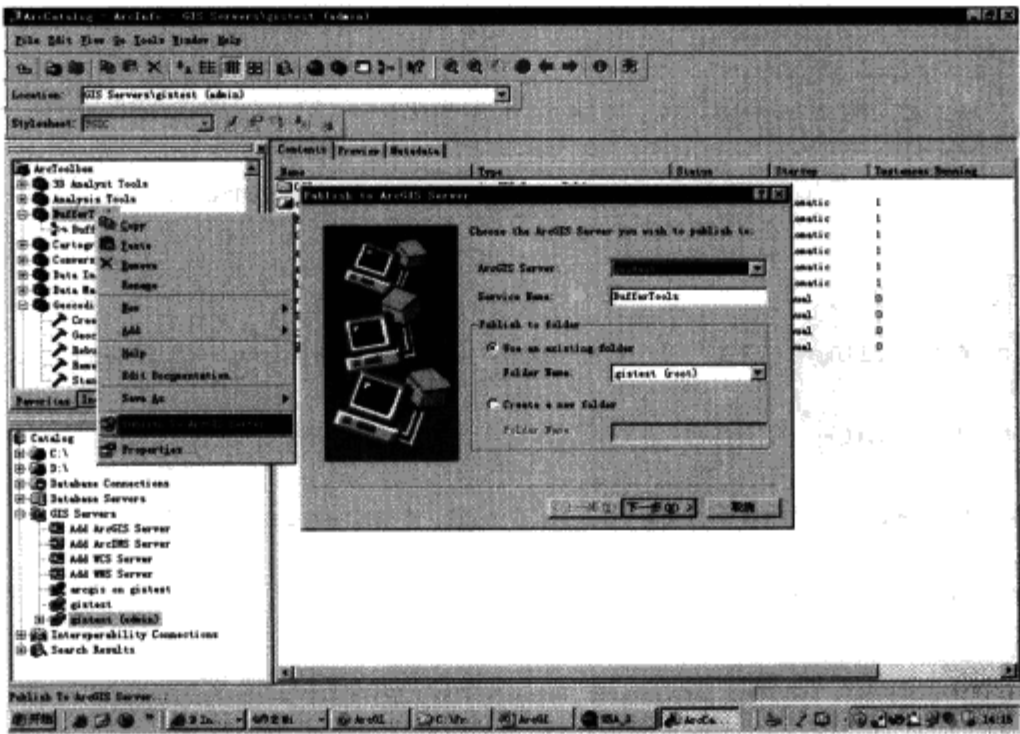


图 13.9 发布缓冲区空间处理服务

(3) 调用地理处理服务, 处理缓冲分析, 示例程序如下:

```

// 初始化 GP resource and functionality
ESRI.ArcGIS.ADF.Web.UI.WebControls.GeoprocessingResourceItem
geoprocessingResourceItem =
    GeoprocessingResourceManager1.ResourceItems.Find(m_geoprocessing
ResourceName);

if (!geoprocessingResourceItem.Resource.Initialized)
    geoprocessingResourceItem.InitializeResource();

// 得到 GP resource and its functionality 的参考信息
ESRI.ArcGIS.ADF.Web.DataSources.IGeoprocessingResource geoprocessing
Resource =
    (ESRI.ArcGIS.ADF.Web.DataSources.IGeoprocessingResource) geoprocessing ResourceItem.
Resource;

    ESRI.ArcGIS.ADF.Web.DataSources.IGeoprocessingFunctionality
commonGeoprocessingFunctionality =
    (ESRI.ArcGIS.ADF.Web.DataSources.IGeoprocessingFunctionality)
    geoprocessingResource.CreateFunctionality(typeof
    (ESRI.ArcGIS.ADF.Web.DataSources.IGeoprocessingFunctionality),
    null);

    ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.GeoprocessingFunctionality
    agsGeoprocessingFunctionality = commonGeoprocessingFunctionality as
    ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.Geoprocessing
Functionality;

//初始化 geoprocessing functionality
if (!agsGeoprocessingFunctionality.Initialized)
    agsGeoprocessingFunctionality.Initialize();

// 设置 geoprocessing (GP) task 的名字
string GPTaskName = "BufferPoints";

// 得到 Geoprocessing 的参数
ESRI.ArcGIS.ADF.Web.DataSources.GPToolInfo adfGPToolInfo =
    agsGeoprocessingFunctionality.GetTask(GPTaskName);
ESRI.ArcGIS.ADF.Web.DataSources.GPParameterInfo[] adfGPParameterInfo
Array =
    adfGPToolInfo.ParameterInfo;

//得到第一个图层
ESRI.ArcGIS.ADF.Web.DataSources.GPFeatureGraphicsLayer adfGPFeature
GraphicsLayer =
    (ESRI.ArcGIS.ADF.Web.DataSources.GPFeatureGraphicsLayer)adfGP ParameterInfoArray[0].
Value;

    ESRI.ArcGIS.ADF.Web.Display.Graphics.FeatureGraphicsLayer featureGraphics
Layer =

```

```

        adfGPFeatureGraphicsLayer.Layer;

//得到包含用户点的图层
ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource adfGraphicsMap
Resource =
    Map1.GetFunctionality(m_graphicsResourceName).Resource as
    ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource;

    ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer pointElement
GraphicsLayer =
    adfGraphicsMapResource.Graphics.Tables[m_pointGraphicsLayerName] as
    ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer;

// 没有退出
if ((pointElementGraphicsLayer == null) || (pointElementGraphicsLayer.
Rows.Count < 1))
{
    ESRI.ArcGIS.ADF.Web.UI.WebControls.CallbackResult
noPointsAlertCallbackResult =
    ESRI.ArcGIS.ADF.Web.UI.WebControls.CallbackResult.CreateJava
Script(
        "alert('No points to buffer')");
    m_callbackResultCollection.Add(noPointsAlertCallbackResult);
    m_callbackResultCollection.Add(HideActivityIndicators());
    return;
}

foreach (System.Data.DataRow dataRow in pointElementGraphicsLayer.Rows)
{
    ESRI.ArcGIS.ADF.Web.Geometry.Point adfPoint =
        pointElementGraphicsLayer.GeometryFromRow(dataRow) as
        ESRI.ArcGIS.ADF.Web.Geometry.Point;
    featureGraphicsLayer.Add(adfPoint);
}

// 得到单位
ESRI.ArcGIS.ADF.Web.DataSources.GPLinearUnit adfGPLinearUnit =
    new ESRI.ArcGIS.ADF.Web.DataSources.GPLinearUnit();
ESRI.ArcGIS.ADF.Web.DataSources.Units adfUnits =
    (ESRI.ArcGIS.ADF.Web.DataSources.Units)System.Enum.Parse(typeof(
    ESRI.ArcGIS.ADF.Web.DataSources.Units), units, true);

adfGPLinearUnit.Units = adfUnits;
adfGPLinearUnit.Value = bufferDistance;

// 设置输入参数开始地理处理任务
ESRI.ArcGIS.ADF.Web.DataSources.GPValue[] adfGPValueArray =
    new ESRI.ArcGIS.ADF.Web.DataSources.GPValue[2];
adfGPValueArray[0] = adfGPFeatureGraphicsLayer;

```

```

        adfGPValueArray[1] = adfGPLinearUnit;
        string jobID = agsGeoprocessingFunctionality.SubmitJob(GPTaskName,
adfGPValueArray);

        // 得到处理结果
        string outputTaskParameters = null;
        ESRI.ArcGIS.ADF.Web.DataSources.GPParameterInfo agsGPParameterInfo;
        for (int i = 0; i < adfGPParamterInfoArray.Length; i++)
        {
            agsGPParameterInfo = adfGPParamterInfoArray[i];

            if (agsGPParameterInfo.Direction ==
                ESRI.ArcGIS.ADF.Web.DataSources.GPParameterDirection.Output)
            {
                outputTaskParameters += agsGPParameterInfo.Name;
                if (i != adfGPParamterInfoArray.Length - 1)
                    outputTaskParameters += ";";
            }
        }

        string jsCheckGeoprocessingJob = string.Format("window.setTimeout(
'executeCallback(" +
            "\"checkGeoprocessingJob\"",
            "\"JobID={0}&TaskName={1}&TaskOutputParameters={2}\"',1000);",
            jobID, GPTaskName, outputTaskParameters);
        ESRI.ArcGIS.ADF.Web.UI.WebControls.CallbackResult
checkGeoprocessingJobCallbackResult =

ESRI.ArcGIS.ADF.Web.UI.WebControls.CallbackResult.CreateJavaScript(jsCheckGeoproc
ssingJob);

        m_callbackResultCollection.Add(checkGeoprocessingJobCallbackResult);

```

通过 ModelBuilder 可以建立各种地理处理服务，包括很多复杂的地理处理服务。这些地理处理模型如果使用 ArcGIS Server ADF 或者 ArcObjects 开发会非常困难，但是通过 ModelBuilder 来建立服务就比较简单。通过 ArcGIS Server 发布这些地理处理模型能与 ArcGIS Server ADF 很好结合，这些地理处理模型也能很方便地供广大的 Web 用户访问。

13.5 最短路径分析

近年来由于 GIS 在大型网状设施（如城市中的道路）中的应用，使得网络分析功能需求迅速增长，通用的网络分析包括路径分析、资源分配、连通分析、流分析，等等。网络分析中最基本的是最短路径分析。最短路径分析就是在制定网络中两节点间找一条阻碍强度最小的路径。根据阻碍强度的不同定义，最短路径不仅指一般地理意义上的最短距离，可以引申到其他的度量，如时间、费用、线路等。

在 ArcGIS 中实现最短路径，不需要用实现最短路径的具体算法，如经典的 dijkstra 算法等，ArcGIS 已经为我们封装好了具体的算法，用户只需要准备好数据就可以。

在一个非拓扑数据中是无法进行最短路径分析的。在 ArcGIS Server 中要进行几何网络分析，首先要对 FeatureClass 进行几何网络的建立。在 ArcCatalog 中单击右键，选择新建 (New) → 几何网络 (Geometric Network)。有了网络拓扑数据，就可以进行最短路径分析了。注意，在 ArcGIS Server 发布网络地图服务时，进行网络分析必须先发布 Network Analysis，如图 13.10 所示。

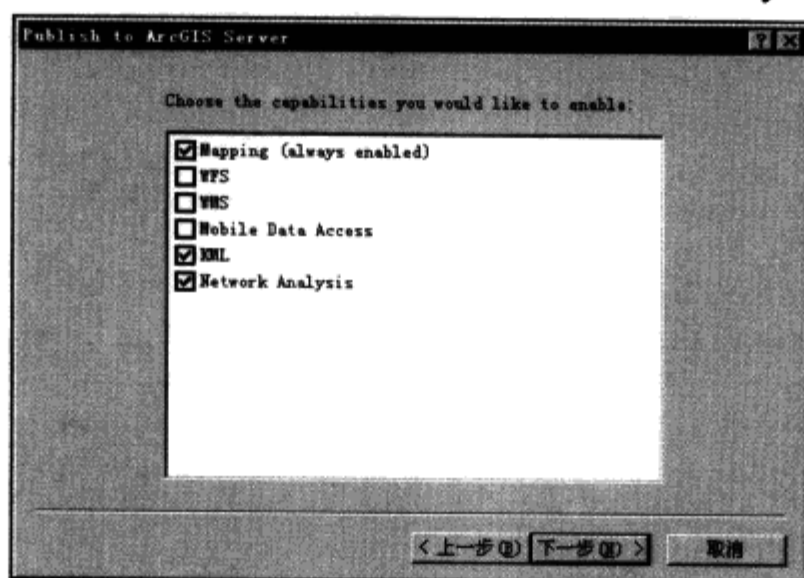


图 13.10 发布网络分析服务

下面给出最短路径分析的示例程序，供读者参考：

```
//根据起止点查找最短路径
private void FindPath(string name1, string name2)
{

    //ags 的服务器名
    string SERVER_NAME = "HZG";

    //ags 里发布的 Map Service 名
    string ROUTE_SERVICE_NAME = "SanFrancisco";

    //创建 NAServerProxy
    NAServerProxy naServerProxy = NAServerProxy.Create(SERVER_NAME, ROUTE_
SERVICE_NAME, null);

    if (naServerProxy == null)
    {
        naServerProxy.Dispose();
        throw (new System.Exception("错误"));
    }
    else
    {

        //获取网络层的名称
        string[] nLayers = naServerProxy.GetNALayerNames(esriNAServerLayer
Type.esriNAServerRouteLayer);
        NAServerSolverParams solverParams = naServerProxy.GetSolverParameters
(nLayers[0]) as NAServerSolverParams;
```



```

//路由分析参数
NAServerRouteParams routeParams = solverParams as NAServerRouteParams;

//不返回地图
routeParams.ReturnMap = false;

//返回 RouteGeometries
routeParams.ReturnRouteGeometries = true;
routeParams.ReturnStops = true;
routeParams.ReturnDirections = true;

//设置起点 PropertySet 参数
PointN point = QueryPoint(name1);
PropertySet propSet = new PropertySet();
PropertySetProperty[] propSetProperty_new = new PropertySetProperty[2];
propSet.PropertyArray = propSetProperty_new;

PropertySetProperty propSetProperty = new PropertySetProperty();
propSetProperty.Key = "Shape";
propSetProperty.Value = point;

PropertySetProperty propSetProperty2 = new PropertySetProperty();
propSetProperty2.Key = "Name";
propSetProperty2.Value = name1;

propSet.PropertyArray[0] = propSetProperty;
propSet.PropertyArray[1] = propSetProperty2;

//设置终点 PropertySet 参数
PointN point2 = QueryPoint(name2);
PropertySet propSet2 = new PropertySet();
PropertySetProperty[] propSetProperty_new2 = new PropertySetProperty[2];
propSet2.PropertyArray = propSetProperty_new2;

PropertySetProperty propSetProperty3 = new PropertySetProperty();
propSetProperty3.Key = "Shape";
propSetProperty3.Value = point2;

PropertySetProperty propSetProperty4 = new PropertySetProperty();
propSetProperty4.Key = "Name";
propSetProperty4.Value = name2;

propSet2.PropertyArray[0] = propSetProperty3;
propSet2.PropertyArray[1] = propSetProperty4;

//设置 Stops 参数
PropertySet[] propSets = new PropertySet[2];
propSets[0] = propSet;

```

```
propSets[1] = propSet2;
NAServerPropertySets StopsPropSets = new NAServerPropertySets();
StopsPropSets.PropertySets = propSets;
routeParams.Stops = StopsPropSets;
NAServerSolverResults solverResults;

try
{
    //进行分析
    solverResults = naServerProxy.Solve(solverParams);
    NAServerRouteResults RouteSolverResults = solverResults as NAServer
RouteResults;

    //显示分析结果
    ShowResults(solverResults);
    //释放 naServerProxy
    naServerProxy.Dispose();

}
catch (Exception e)
{
    //释放 naServerProxy
    naServerProxy.Dispose();
}

smapstring = Map1.CallbackResults.ToString();
}
```

显示路径分析结果的示例程序:

```
public void ShowResults(NAServerSolverResults solverResults)
{
    NAServerRouteResults RouteSolverResults = solverResults as NAServerRoute
Results;

    //开始点终点路径显示
    AddRoutesAndStops(RouteSolverResults);
    //路径区域全屏显示
    PolylineN polylineN = RouteSolverResults.RouteGeometries[0] as PolylineN;
    EnvelopeN envelopeN = polylineN.Extent as EnvelopeN;
    double width = envelopeN.XMax - envelopeN.XMin;
    double height = envelopeN.YMax - envelopeN.YMin;
    double fivePercent;

    if (width > height)
    {
        fivePercent = width * .05;
    }
    else
    {
        fivePercent = height * .05;
    }
}
```

```

        envelopeN.XMin = envelopeN.XMin - fivePercent;
        envelopeN.YMin = envelopeN.YMin - fivePercent;
        envelopeN.XMax = envelopeN.XMax + fivePercent;
        envelopeN.YMax = envelopeN.YMax + fivePercent;

        Map1.Extent = ESRI.ArcGIS.ADF.Web.DataSources.ArcGISSEServer.Converter.
ToAdfEnvelope(envelopeN);
        Map1.Refresh();
    }

    private void AddRoutesAndStops(NAServerRouteResults rResult)
    {
        //分析结果路径
        Polyline[] lines = rResult.RouteGeometries;
        RecordSet stops = rResult.Stops;
        //获取 Buffer 的 MapFunctionality
        ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapFunctionality mapFunct =
(ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapFunctionality)Map1.GetFunctionality("
pathLayer");
        //获取 Buffer 的 MapResource
        ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource gResource = (ESRI.
ArcGIS.ADF.Web.DataSources.Graphics.MapResource)mapFunct.Resource;

        //把 buffer 结果范围进行显示
        ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer glayer = null;
        //查找 ElementGraphicsLayer 在 Buffer 中
        foreach (System.Data.DataTable dt in gResource.Graphics.Tables)
        {
            if (dt is ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer)
            {
                glayer = (ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphics
Layer)dt;
                break;
            }
        }
        //如果 Buffer 中没有 ElementGraphicsLayer 就新增加一个 ElementGraphicsLayer
        if (glayer == null)
        {
            glayer = new ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphics
Layer();
            gResource.Graphics.Tables.Add(glayer);
        }
        //清除 ElementGraphicsLayer 中的内容
        glayer.Clear();
        ESRI.ArcGIS.ADF.Web.Geometry.Geometry geom = (ESRI.ArcGIS.ADF.Web.
Geometry.Geometry)ESRI.ArcGIS.ADF.Web.DataSources.
ArcGISSEServer.Converter.ToAdfPolyline((PolylineN)lines[0]);
        //设置点显示
        ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement ge = new ESRI.ArcGIS.
ADF.Web.Display.Graphics.GraphicElement(geom, System.Drawing.Color.Red);
        //设置透明度
        ESRI.ArcGIS.ADF.Web.Display.Symbol.SimpleLineSymbol sls = new ESRI.ArcGIS.
ADF.Web.Display.Symbol.SimpleLineSymbol();
        sls.Width = 5;
    }

```

```

sls.Color = System.Drawing.Color.Blue;
sls.Type = ESRI.ArcGIS.ADF.Web.Display.Symbol.LineType.Dash;
sls.Transparency = 50;
ge.Symbol = sls;
// ge.Symbol.Transparency = 50;
//添加到 Buffer 中进行显示
glayer.Add(ge);

Record[] stopRecords = stops.Records;
int stopCount = stopRecords.Length;
for (int iStop = 0; iStop < stopCount; iStop++)
{
    ESRI.ArcGIS.ADF.Web.Geometry.Geometry geom2 = (ESRI.ArcGIS.ADF.Web.
    Geometry.Geometry)ESRI.ArcGIS.ADF.Web.DataSources.ArcGISServer.Converter.
    ToAdfPoint(stopRecords[iStop].Values[1] as PointN);

    //设置点显示
    ESRI.ArcGIS.ADF.Web.Display.Graphics.GraphicElement ge2 = new ESRI.
    ArcGIS.ADF.Web.Display.Graphics.GraphicElement(geom2, System.Drawing.Color.Red);
    //设置透明度

    ge2.Symbol.Transparency = 50;

    //添加到 Buffer 中进行显示
    glayer.Add(ge2);
}
}

```

程序运行结果如图 13.11 所示。

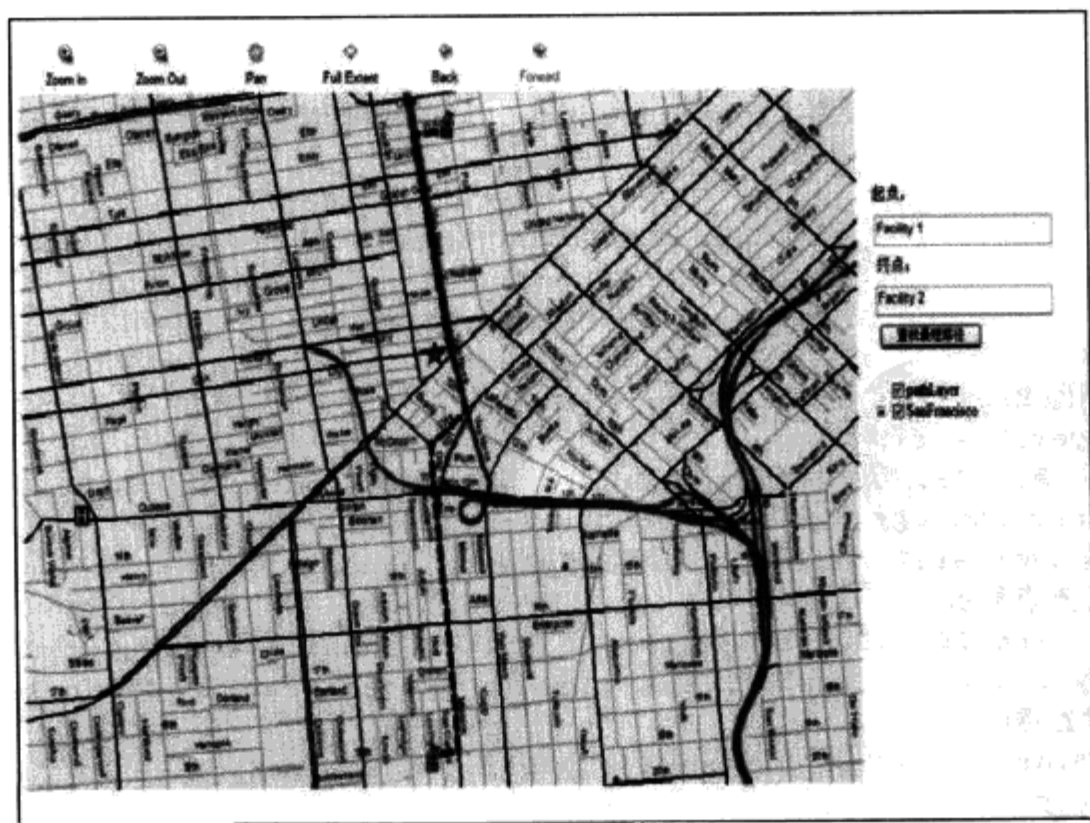


图 13.11 最短路径分析结果图

13.6 小结

本章主要讲述在 ArcGIS Server 中的一些高级应用，所谓高级就是稍微有些复杂。更多的是利用 ArcObjects 的 API 函数封装 COM 动态链接库供 ArcGIS Server 调用，如果是简单的应用，大可不必采用调用 COM 的方式，利用 ArcGIS Server ADF 来实现，简单得多。但是利用 ArcObjects 和 ModelBuilder 可以实现复杂的功能，效率更高，读者应根据自己的实际情况进行处理。

第 14 章

ArcGIS Server 9.3开发模式

ArcGIS Server 作为一个 GIS 平台软件，提供一系列的 GIS 解决方案。当然，它的二次开发模型也非常多，本章主要讲述 ArcGIS Server 9.3 的开发模式。

14.1 .NET Web ADF 开发

本书介绍的 ArcGIS Server 多以 .NET Web ADF 为例。使用 ArcGIS Server .NET 来创建应用有 3 种方式。

- 使用 ArcGIS Server Manager 来创建一个 Web 应用。
- 使用 Web Mapping Application 模板创建一个 Web 应用。
- 使用 ADF 来创建应用。

对于一个开发项目来说，一般采用第二种或第三种方式来进行开发。第一种开发方式适用于了解 ArcGIS Server 的 .NET ADF 开发应用。在弄懂 ADF 的体系结构后，就可以采用后两种方式进行开发。

首先 .NET ADF 的数据源可以是 ArcGIS Server 本身的，也可以是 ArcIMS 的。

.NET ADF 为开发人员提供控件和 API。控件提供用户界面，而 Command API 和 Specific API 的类用来具体完成 GIS 的功能。有关控件的详细内容请参见第 6 章。

(1) Resource、Resource Manager、WebControls、Functionality 的关系，如图 14.1 所示。

控件和数据源之间的关系是通过一些 Resource Manager 控件来维护管理的。Resource Manager 决定哪些数据源可以使用 Resources，以及这些 Resources 如何被控件所使用。一旦一个数据源被 Resource Manager 管理后，就表现为 Resources。控件通过 Resources 到达数据源。Resources 可以把数据源以多种形式表现出来，比如可以提供一幅地图展现在 Map 控件中，可以把数据源以图层列表的方式展现在 Toc 控件中。不同的 Resources 拥有不同的功能，也就是 Functionality。从控件的角度来讲，不同的控件可以通过不同的方式来使用相同的数据源，比如一个 Resource 可以为 Map 控件提供地图，也可以为 Toc 控件提供图层列表，这就是 Resources 的不同的

Functionality。从数据源的角度来讲，不同的 Resources 会展现一些通用功能，也提供不同的 Functionality，比如展现地图、查询地图等。

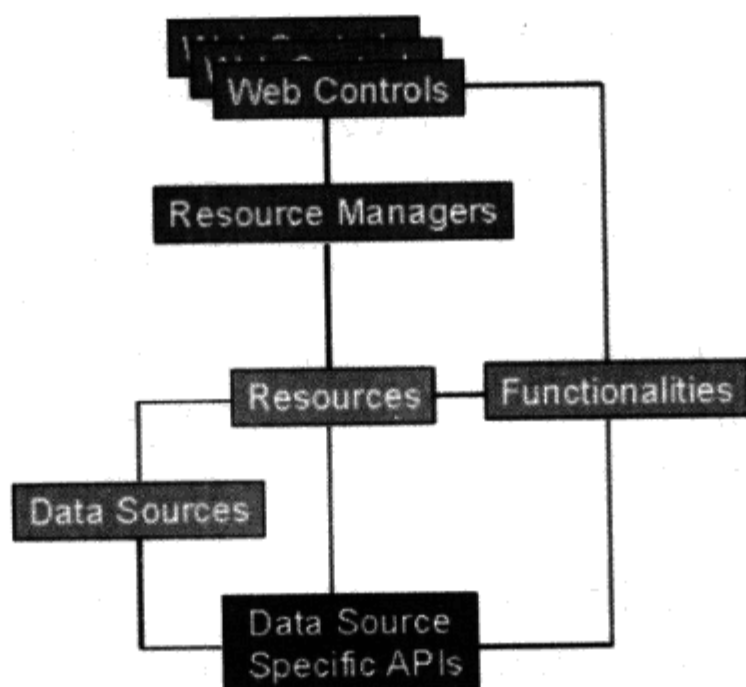


图 14.1 控件与数据源关系图

(2) Common APIS 和 Specific APIS 的关系。

上面介绍了 Resources 可以展现不同功能的能力，但是具体如何展现取决于数据源本身能够提供什么样的功能。有些能力是所有数据源所共有的，也就是说 Resources 可以表现出一些所有的数据源都可以提供的能力，比如提供地图服务——这是最基本的能力。因此，ADF 将实现这类基本功能所需要的类归为 Common APIS。有些功能是某些数据所特有的，比如网络分析服务，那些没有建立拓扑结构的空数据是无法提供此类服务的。这些就被称为 Specific APIS。Specific APIS 包括 ArcIMS API、ArcWeb API、OGC\WMS、ArcGIS Server SOAP API、ArcGIS Server ArcObjects API，等等。

(3) Common APIS 的基本结构，ADF 中如何使用这些 Common APIS。

在开发过程首先接触到的就是 ADF 提供的这些 Common APIS，它的结构比较明朗，不像 Specific APIS 那样多样。其基本接口如下。

- IGISDataSource——定义了数据源的连接。
- IGISResource——定义了 DataSource 提供的信息类型。
- IGISFunctionality——定义了 Resource 如何被使用。

这 3 个接口是不同的数据源可以展现一些基本功能的基本接口，也就是说不同的数据源要实现基本的功能必须要实现这 3 个基本的类，才能在控件上展现这些基本的能力。各种数据源都用相应的类来实现这 3 个接口。

下面来看看这 3 个接口之间的关系。

一个 IGISDataSource 会包含一系列的 IGISResource。IGISDataSource 是一个相对比较大的概念，主要包含了 ArcGIS Server Local、ArcGIS Server Internet、ArcIMS、WMS、ArcWeb 等。而 ArcGIS Server Local 包含 MapResource、GeocodeResource、GeoprocessingResource 等。

IGISResource 包含一系列的 IGISFunctionality。Functionality 主要有两种。MapFunctionality 用来展现 Resource 的地图能力，比如输出地图图片，改变地图范围，设置图层的可见性等。QueryFunctionality 主要展现 Resource 的数据空间和属性的查询能力。

ArcGIS Server Local 数据源的相应实现类如下。

- IGISDataSource——GISDataSourceLocal。
- IGISResource——MapResourceLocal、GeocodeResourceLocal。
- IGISFunctionality——MapFunctionality、QueryFunctionality。

这就是 Web ADF 的优势所在，它可以使得各种不同的数据源都展现为 Resource，使得它们可以以相同的方式使用。对于控件而言，每个 Resource 就是一个图层，而不管数据源是什么。

就地图缩小的功能来说，一个 MapControl 中有两个 DataSource，一个是 ArcGIS Server Local，一个是 ArcIMS。当地图的范围重新设定之后，控件都通过 Resources 提供的 MapFunctionality 来给每个 Resources 重新设定范围，每个 Resources 输出这个新的地图，而 MapControl 负责把这些输出图片显示在同一界面上。对于控件而言，每个 Resource 就是一个图层，从 Resource 可以访问 DataSource 本身。

14.2 Java Web ADF 开发

Java Web ADF 与 NET Web ADF 基本上相似。Java Web API 的体系结构如图 14.2 所示。

Java Web ADF 可以分为两部分：Common API（括一些 Web Controls）、Resource Specific APIs，有关 Common API 与 Specific 的解释参见 14.1 节。

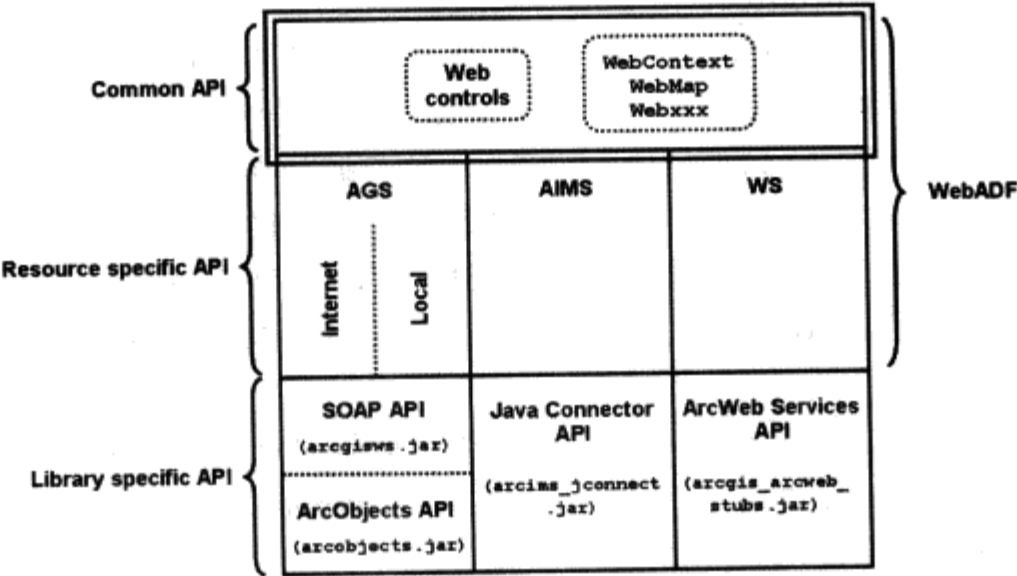


图 14.2 Java Web ADF 体系结构

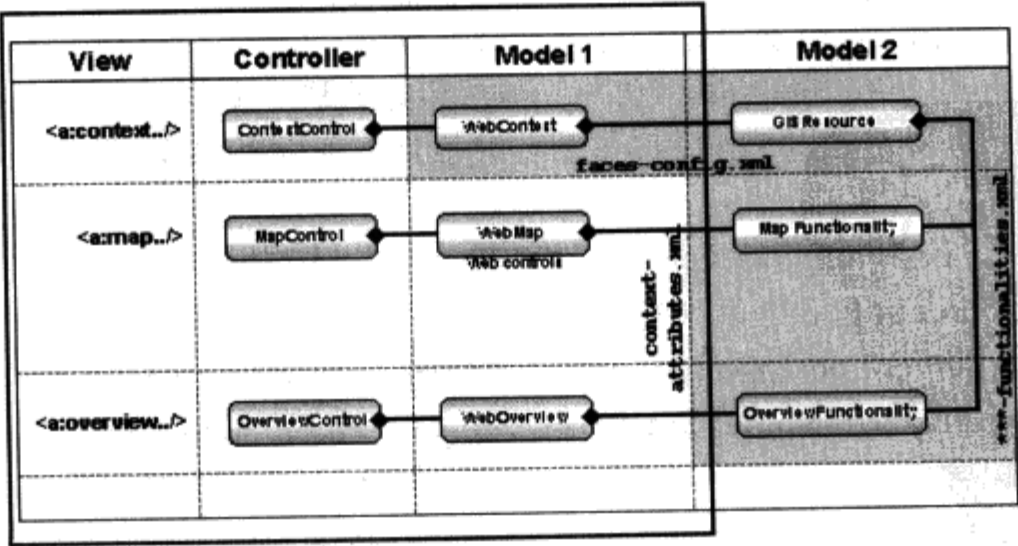


图 14.3 Java Web MVC 架构

ArcGIS Server Java Web ADF 的 MVC 架构如图 14.3 所示。Web ADF 架构中, Controls 是第一层, 它在 MVC 架构包含 View/Controller 层, 并同时存在于 View 和 Controls 层, 因为它不仅和客户端交互并且向客户端输出数据, 在 Request 生命周期响应应用流程。Controls 的 JavaDoc 可以在包 `com.esri.adf.web.faces.componet` 中找到。Controls 底下的层是 MVC 架构的第一层 Model 层, 是一些和 Web Controls 直接工作的数据对象。这些对象通常和业务对象在其他层相连接但不是必须的, 该层的控制对象是在 WebContext 中, 它不仅控制对数据资源的连接并且协调这个层其他对象之间的状态; 所有其他对象都在 Context 中注册为属性, 作为 Context 的属性并执行一个接口, 这一层的其他对象将被通知应用状态的改变和 Web ADF 提供的 Request 生命周期。Model 层包含: GISResource 和 GISFunctionality, GISResource 是 Web ADF 用来显示和分析的数据来源, GISFunctionality 是对于特定的 GISResource 用来查询和提供地图服务。Java Web ADF 的 Common API 如图 14.4 所示。

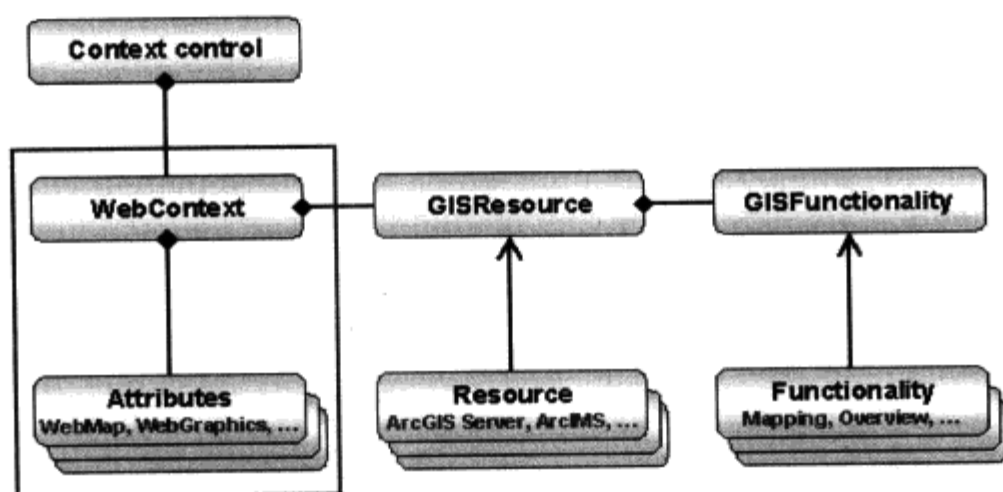


图 14.4 ArcGIS Server Java 开发架构

WebContext 是一个 Model Tier1 组件, Context 也扮演 Model Tier1 组件的管理者。它连接 Data Sources 和 Controls。如果想使协调起作用, 所有的 Model Tier1 组件必须在 Context 中注册为被管理属性。实例脚本如下:

```
<managed-property> <property-name>attributes</property-name>
<map-entries>
<map-entry>
<key>map</key>
<value>#{map}</value>
</map-entry> </map-entries> </managed-property>
```

Context Control 中唯一例外的信息是应用的地理资源 (Resources)。这将是应用中所要使用的数据的每一个数据源的入口。另外, 每一个资源在语法上使用 JSF-EL 来关联一个包含合适连接信息的 managed bean。下面的程序片段显示出一个 managed bean, ags; 在这种情况下, 是一个 ArcGIS Server 连接。

```
<managed-property>
<property-name>resources</property-name>
<list-entries>
value>#{ags1}</value>
</list-entries>
</managed-property>
```

每一种和 Context 相关联的 Resource 都需要一个 managed bean 来处理与特定 Data Source 的交互。在这个例子中，只有一个 Resource，而这个 Resource 使用 ArcGIS Server，示例脚本如下：

```
<managed-bean>
<managed-bean-name>ags1</managed-bean-name>
<managed-bean-class>com.esri.adf.web.ags.data.AGSMapResource</managed-bean-class>
<managed-bean-scope>none</managed-bean-scope>
<managed-property>
<property-name>user</property-name>
<value>#{agsUser1}</value>
</managed-property>
<managed-property>
<property-name>serverObjectName</property-name>
<value>usa</value>
</managed-property>
<managed-property>
<property-name>hosts</property-name>
<list-entries>
<value>YourServer</value>
</list-entries>
</managed-property>
<managed-property>
<property-name>functionalities</property-name>
<map-entries>
<map-entry>
<key>map</key>
<value>#{agsMap}</value>
</map-entry>
<map-entry>
<key>overview</key>
<value>#{agsOverview}</value>
</map-entry>
</map-entries>
</managed-property>
</managed-bean>
```

Web ADF 提供 5 种不同的 datasource 的 functionalities: ags-functionalities.xml、aims-functionalities.xml、aws-functionalities.xml、ejb-functionalities.xml、wms-functionalities.xml，每一种只有一种合适的 functionalities。

Web ADF 为暴露 functionalities 的 managed bean 使用一个一致的命名习惯。ArcGIS Server 的 functionalities 在 JavaDoc 中可以找到，每一个 functionality 的 managed bean 名称是 source 的名字，都是小写，functionality 名称大小写混合。例如，ArcGIS Server map functionality 类命名为 AGSMapFunctionality，而 managed bean 命名为 agsMap。如果想在地图上显示一个 resource，一定要声明这个 resource 的 functionality。反过来也是这样。如果想使用一个 resource 但是并不使用它所有的 functionality，唯一的方法是在 faces-config.xml 文件上，对于这个 resource，省略不需要的 functionalities。

14.3 SOAP API 应用

ArcGIS Server SOAP API 基于 XML 和 SOAP 标准协议来访问 ArcGIS Server 提供的服务。服务器对象和服务对象扩展定义了 SOAP 元素和属性，通常称为架构。它们可以处理 SOAP 的请求和应答。由于其具有这种能力，开发人员可以通过 SOAP 来访问服务器对象而不必通过 Web Service。也就是说，可以通过 C/S 方式来访问 ArcGIS Server 提供的服务而不必一定采用 B/S 方式。ArcGIS Server Web Service 通过 Web Service endpoint 来暴露这种能力，但是 SOAP 的请求和应答都是服务器对象来处理。

ArcGIS SOAP API 被设计为无状态 ArcGIS Server 的服务，因此利用服务器对象实现的无状态 ArcObjects 接口方法映射为 SOAP 代理服务器上的方法。例如，MapServer 代理服务器和 IMapServer ArcObjects COM 接口都暴露了 ExportMapImage 方法。虽然方法是相同的，但是协议和种类却不同。调用 ExportMapImage 方法产生新的地图图片，但没有改变服务器对象的状态。

1. 服务器连接

ArcGIS Server Service 可以通过局域网或互联网来连接。局域网连接是指客户端通过利用 ArcObjects 来连接 Server Object Manager (SOM)；互联网连接是指客户端通过利用裸对象（开发环境 .NET、Java）来连接 Web Service endpoint。

- 互联网连接经常使用 SOAP API 并且定义一个连接 Web Service endpoint 的 URL。它可以被广域网和局域网的客户端调用。
- 局域网连接通常使用 ArcObjects API。这种连接是使用 SOM 机器的名字和用户提供的验证信息来定义，它只能被局域网的客户端使用。

服务器连接如图 14.5 所示。

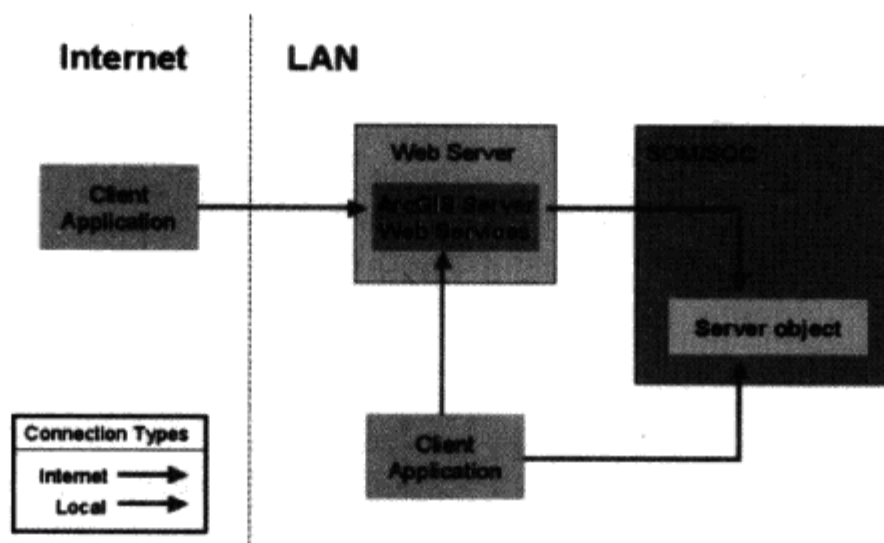


图 14.5 SOAP API 连接

2. Web Service 标准

ArcGIS Server SOAP API 依赖两个标准：Web Service 描述语言，即 WSDL（基于 XML 的）和简单对象协议（SOAP）。WSDL 是一种用来描述 Web 服务和说明如何与 Web 服务通信的 XML 语言。客户端可以通过这些服务访问点对包含面向文档信息或面向过程调用的服务进行访问（类似远程过程调用）。WSDL 首先对访问的操作和访问时使用的请求/响应消息进行抽象描述，然后将其绑定到具体的传输协议和消息格式上以最终定义具体部署的服务访问点。具体部署的服务访问点通过组合成为抽象的 Web 服务。简单对象访问协议（SOAP）是一种轻量级的、简单的、

基于 XML 的协议，它被设计成在 Web 上交换结构化的、固化的信息。SOAP 可以和现存的许多 Internet 协议和格式结合使用，包括超文本传输协议（HTTP）、简单邮件传输协议（SMTP）、多用途网际邮件扩展协议（MIME）。它还支持从消息系统到远程过程调用（RPC）等大量的应用程序。

3. ArcGIS Server SOAP 实现

SOAP 支持服务器对象和服务对象扩展，因此在生命周期的某些点上，ArcObjects 必须被使用。两个基本的 ArcObjects 接口被用来调用 SOAP API: IServiceCatalogAdmin 和 IRequestHandler 接口。两个接口的版本同时更新，如 IServiceCatalogAdmin2 和 IRequestHandler2。IServiceCatalogAdmin 接口用来获得服务器对象、服务器对象扩展、ServiceCatalog 的 WSDL；IRequestHandler 接口用来处理 SOAP 请求和应答。访问 ArcGIS Server Service 主要要实现这两个接口。

可以通过 SOAP API 来访问的 ArcGIS Server 服务如下表。

通过 SOAP API 来访问的 ArcGIS 服务

Service Type	ArcObjects Server Object 或 Extension Type
Map Service	MapServer
Geocode Service	GeocodeServer
Geodata Service	GeoDataServer
Geoprocessing Service	GPServer
Globe Service	GlobeServer
Network Analysis	NAServer

利用 ArcObjects API 和 IServiceCatalogAdmin 接口访问 ServiceCatalog 对象，可以得到每一种服务类型的 WSDL。调用 GetCatalogDescriptionDocument()方法返回 ServiceCatalog WSDL，而 GetDescriptionDocument()则返回 MapServer、GeocodeServer 的 WSDL。IServiceCatalogAdmin 接口包含于 ESRI.ArcGIS.Server.DLL 中，示例程序如下：

```
ESRI.ArcGIS.ADF.Connection.AGS.AGSServerConnection gisconnection =
    new ESRI.ArcGIS.ADF.Connection.AGS.AGSServerConnection();
gisconnection.Host = servername;
gisconnection.Connect();

IServerObjectManager serverManager = gisconnection.ServerObjectManager;
IServerContext serverContext = serverManager.CreateServerContext(servicename,
servertype);

IServiceCatalogAdmin2 isc = (IServiceCatalogAdmin2)serverContext.CreateObject
("esriServer.ServiceCatalog");

UTF8Encoding utf8 = new UTF8Encoding();

// Catalog WSDL
byte[] bitscatalog = isc.GetCatalogDescriptionDocument("Catalog", "http://
localhost");
string catalog_wsdl = utf8.GetString(bitscatalog);
```



```
// Service WSDL
byte[] bitsservice = isc.GetDescriptionDocument(servicename, servertime, "http://localhost");
string service_wsdl = utf8.GetString(bitsservice);
```

4. SOAP 请求与应答

所有访问 ArcGIS Server 服务的请求都被服务器对象通过 IRequestHandler 接口来处理。因此，基于 ArcObjects API 的客户端可以利用 IRequestHandler 接口提交 SOAP 请求，并返回一个 SOAP 应答。下面的示例代码定义 HandleStringRequest 方法来处理 SOAP 的请求字符串并产生 SOAP 字符串响应。

```
IRequestHandler irh = (IRequestHandler)serverContext.ServerObject;

string soap_request = "<?xml version='1.0' encoding='utf-8' ?>";
soap_request += "<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/' xmlns:tns='http://www.esri.com/schemas/ArcGIS/9.2'>";
soap_request += "<soap:Body>";
soap_request += "<tns:GetDefaultMapName>";
soap_request += "</tns:GetDefaultMapName>";
soap_request += "</soap:Body>";
soap_request += "</soap:Envelope>";

string soap_response = irh.HandleStringRequest("map,query,data", soap_request);
```

创建和解析 SOAP 字符串能被智能处理。利用原始 WSDL 可以决定如何产生 SOAP 请求并且处理 SOAP 响应。SOAP 的请求与应答如图 14.6 所示。

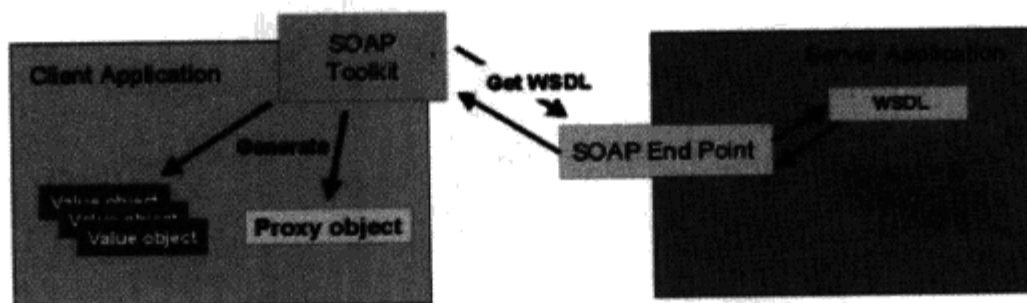


图 14.6 SOAP 的请求与应答

下面介绍一个使用 SOAP API 的例子。

(1) 在客户端项目解决方案中，选中 References 后单击鼠标右键，选择“Add Web Reference”选项，如图 14.7 所示。

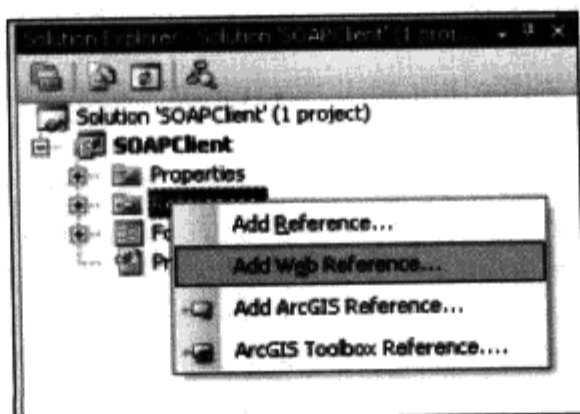


图 14.7 “Add Web Reference”选项

(2) 在弹出的“Add Web Reference”对话框中，在 URL 下拉列表框中输入 Web Service endpoint 的 WSDL 的 URL。例如：<http://localhost/arcgis/services/NorthAmerica/MapServer?wsdl>，如图 14.8 所示。

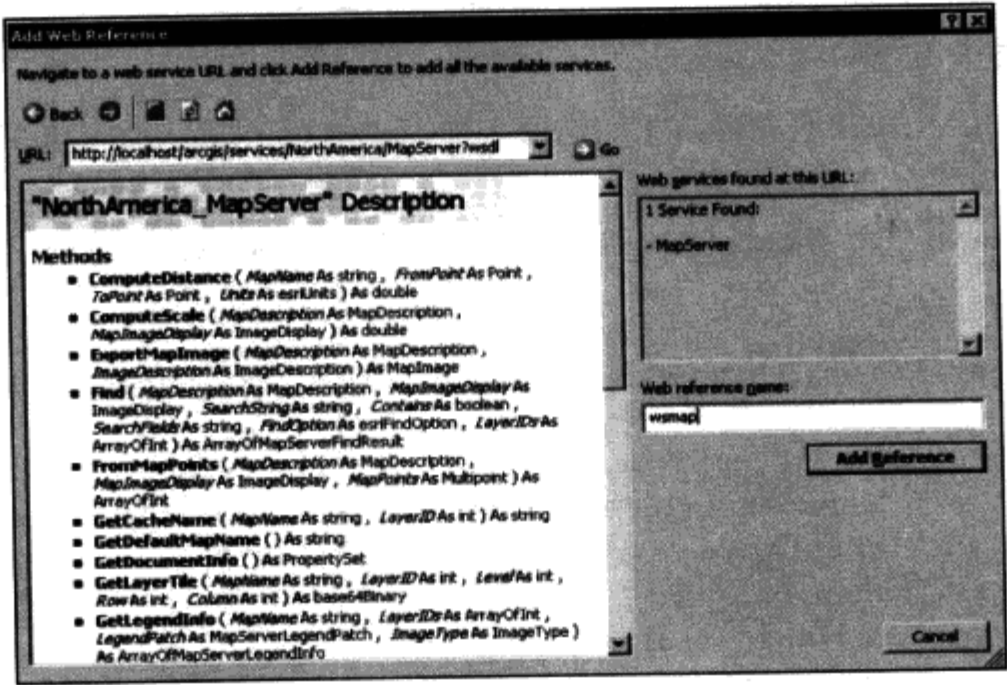


图 14.8 输入 URL

(3) 编辑客户端界面，如图 14.9 所示。

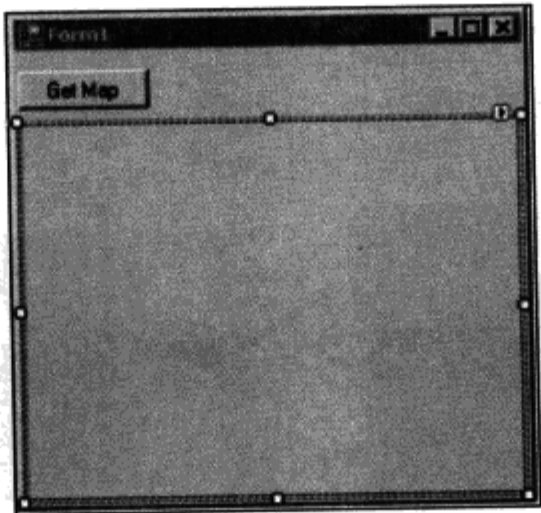


图 14.9 编辑客户端界面

“Gel Map” 按钮用来获得 ArcGIS Server Service，绘画框用来显示地图。
示例程序如下：

```
wsmapi.NorthAmerica_MapServer mapserver = new wsmapi.NorthAmerica_MapServer();
mapserver.Url = "http://localhost/arcgis/services/NorthAmerica/MapServer";
wsmapi.MapServerInfo mapinfo = mapserver.GetServerInfo(mapserver.GetDefault
MapName());
wsmapi.MapDescription mapdesc = mapinfo.DefaultMapDescription;
wsmapi.ImageType imgtype = new wsmapi.ImageType();
imgtype.ImageFormat = wsmapi.esriImageFormat.esriImageJPG;
imgtype.ImageReturnType = wsmapi.esriImageReturnType.esriImageReturnURL;
wsmapi.ImageDisplay imgdisp = new wsmapi.ImageDisplay();
imgdisp.ImageHeight = pictureBox1.Height;
imgdisp.ImageWidth = pictureBox1.Width;
```

```

imgdisp.ImageDPI = 96;
wsmap.ImageDescription imgdesc = new wsmap.ImageDescription();
imgdesc.ImageDisplay = imgdisp;
imgdesc.ImageType = imgtype;
wsmap.MapImage mapimg = mapserver.ExportMapImage(mapdesc, imgdesc);
System.Net.HttpWebRequest webreq =
    (System.Net.HttpWebRequest)System.Net.WebRequest.Create(mapimg.ImageURL);
System.Net.HttpWebResponse webresp =
    (System.Net.HttpWebResponse)webreq.GetResponse();
System.Drawing.Image img = System.Drawing.Image.FromStream(webresp.GetResponse
Stream());
pictureBox1.Image = img;

```

程序运行效果如图 14.10 所示。



图 14.10 运行界面

14.4 REST API

说到 ArcGIS Server REST API，需要先介绍一下 REST。REST 是表述性状态转移（REpresentational State Transfer）的简称，它本身是一个抽象的概念，是一种为了实现互联网的超媒体分布式系统的行动指南。REST 与 URL（统一资源定位）是一种超媒体系统，它可以把网络上所有的资源进行唯一的定位，不管是图片、文档还是视频文件，也不管是 txt 文件格式、xml 文件格式还是其他文本文件格式。它利用支持 HTTP 的 TCP/IP 协议来确定互联网上的资源，所以，可以说成是 REST IS URL。

ArcGIS Server 9.3 提供全新的 REST API，实现一些 ArcGIS Server 的通用功能和服务，便于 ArcGIS Server 客户端开发，如 JavaScript API。安装 ArcGIS Server 9.3 后，可以访问 REST。例如：

- Java:http://<host>:8339/arcgis/test;
- .NET:http://<host>arcgis/rest。

总之，可以通过 `http://<host><instance>/services/<folder>` 方式来访问这些服务的根目录，其中“instance”默认为“arcgis/rest”，里面列举了当前服务器所有已启动的 Server 服务，并且可以将这些列表导入 Google Earth，格式为 KML。REST 访问地址中，folder 和 Server name 是对大小写敏感的，folder 前面部分则没有影响。

单击根目录中的一个服务，页面会展示该服务的所有基本信息，包括地图图层、图层编号、

空间参考、部分 Cache 信息、初始化地图显示范围、地图全图显示范围、地图单位、MXD 文档基本信息等，如图 14.11 所示。

同时，还可以选择将该服务放在哪一种客户端（平台）上显示。ArcGIS JavaScript 链接地址为：<http://gistest/ArcGIS/rest/services/china/MapServer?f=jsapi>。这为 ArcGIS GIS Server JavaScript API 访问 REST API 提供了一种途径，其他的客户端都可以用“f”这个参数来制定。

REST 中所描述的服务，包括资源和操作两种类型，资源是指描述该服务的一些属性信息；操作指的是基于该服务能够实现的功能，如导出地图、查询、搜索、生成 KML。ArcGIS Server 9.3 中资源的服务组织结构如图 14.12 所示。

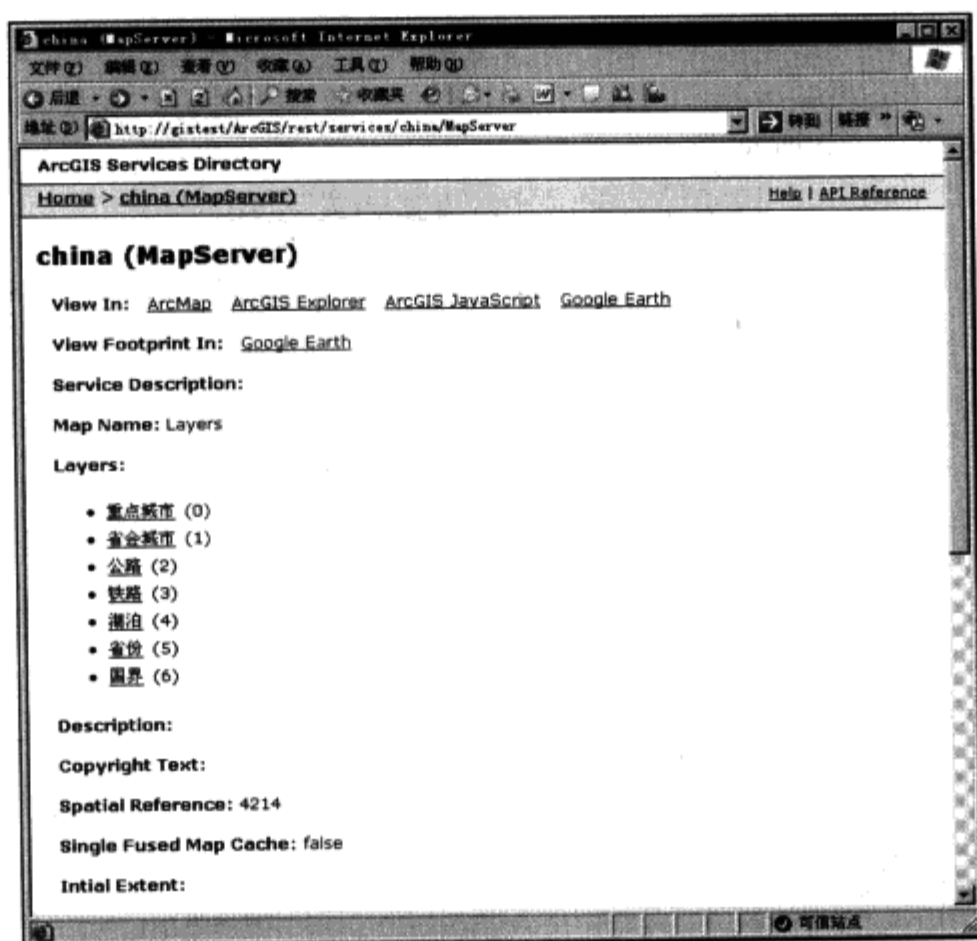


图 14.11 REST 服务基本信息界面

每一种资源都由统一的 URL 来标识，资源通过连接被互联网关联在一起。如 Map Service 下的各种操作功能。资源有多种表达方式，如希望地图服务返回图片，可以在 URL 制定参数“f=image”，返回在 ArcMap 中浏览的图层文件；制定“f=lyr”，返回在 JavaScript API 客户端浏览的网页，制定“f=jsapi”等，这些都取决于应用的要求。这种框架粗看有点 ArcXML 的感觉，实际上它体现了更多的语义。如何在 JavaScript API 中调用这些资源呢？如查询：<http://gistest/ArcGIS/rest/services/china/MapServer/find?searchText=%E5%8C%97%E4%BA%AC&contains=true&searchFields=name&sr=&layers=1&returnGeometry=true>。

由上述的例子可以看出，ArcGIS Server REST API 始终遵循了 REST 的 5 个基本原则：

- 为所有“事物”定义 ID；
- 将所有事物链接在一起；
- 使用标准方法；
- 资源多重表述；
- 无状态通信。

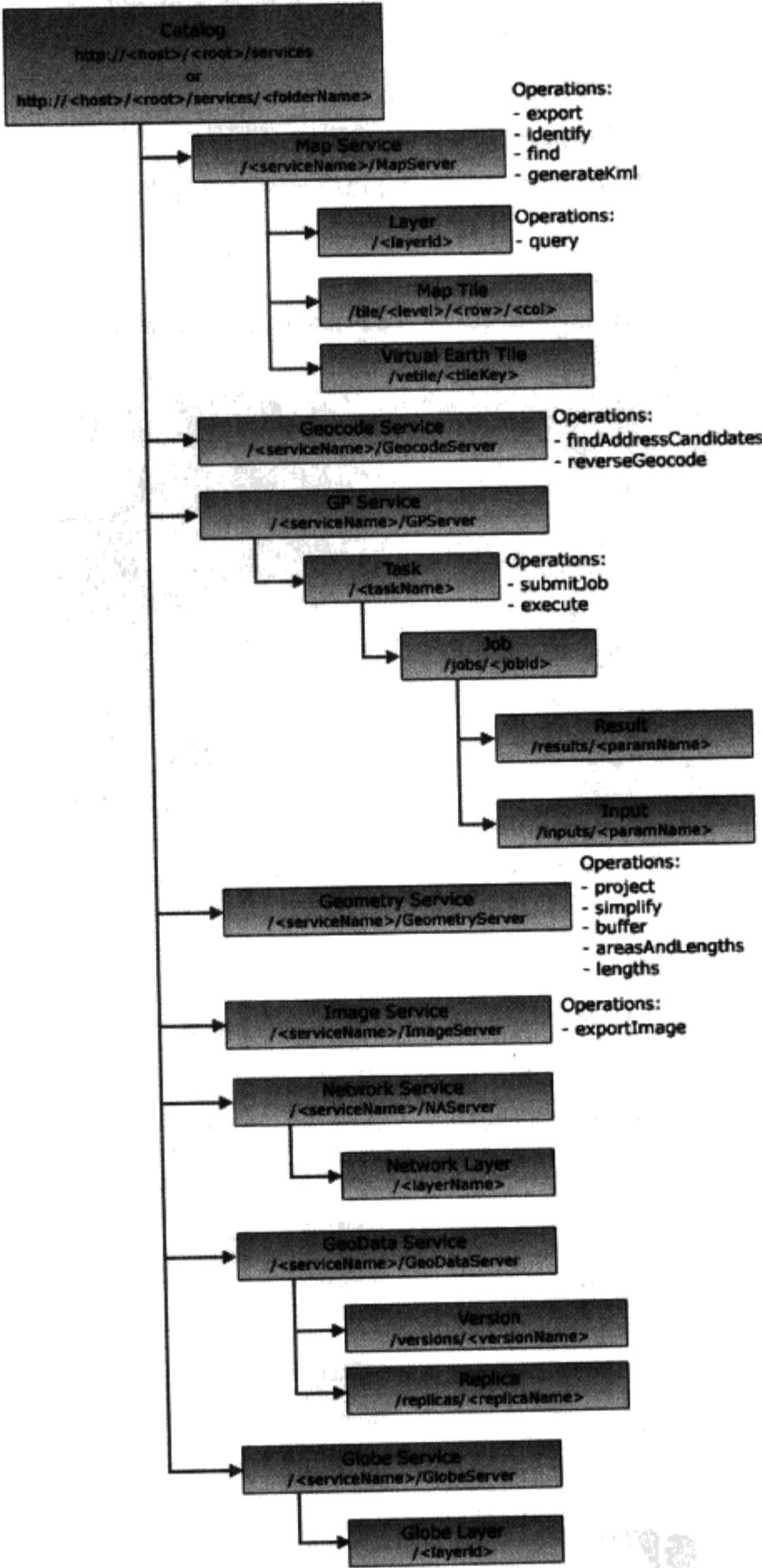


图 14.12 ArcGIS Server 服务组织结构

综合分析 ArcGIS Server REST API 有以下优点：

(1) ArcGIS Server REST 将 GIS 基础和核心功能全部进行了封装，并以服务的方式提供给客户端，如常见的地图展示、图层信息访问、空间几何查询、高级分析功能（网络分析、地理统计、空间分析统计、水文分析、地址编码、逻辑网络、坐标转换等）。无论是 JavaScript、Flex

还是 Silverlight 都无须关注 GIS 功能的实现，只关注人机交互和用户 UI 设计。

(2) 有利于采用快速原型的开发方式。

没有任何后台逻辑之前，表现层就可以开始设计，FlexViewer 无疑是最好的说明。

(3) 合理分配负载，减轻服务器压力。

REST 负责将需要的数据传回客户端，工作量较大的渲染工作放在客户端进行，有效地减轻了服务器的压力，使得用户体验更佳，视觉效果更好。利用 Flex 开发不仅有友好的 UI，而且网站运行效率很高，如图 14.13 所示。

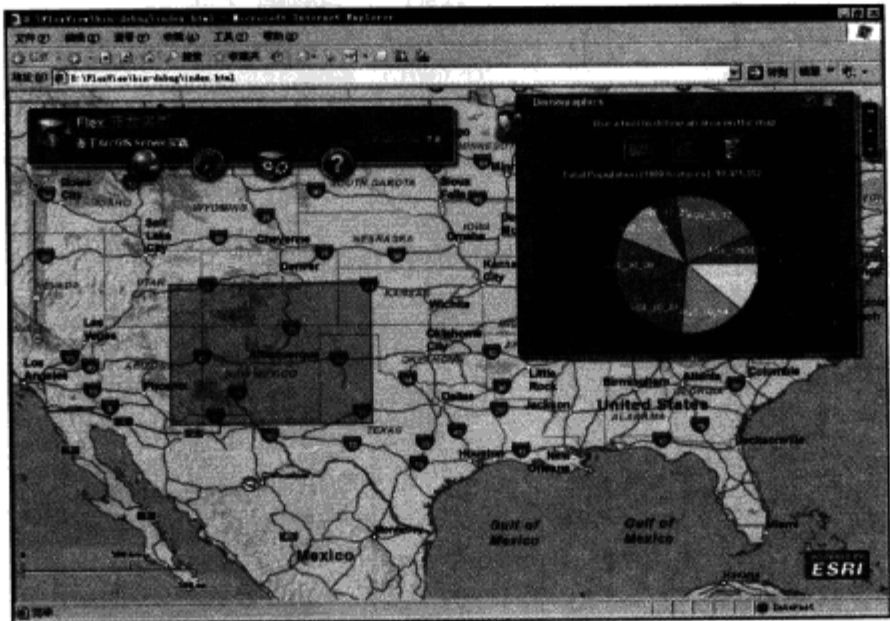


图 14.13 Flex+ArcGIS Server REST API 开发的 WebGIS

14.5 Mobile ADF

ArcGIS Server Mobile ADF 是基于 .NET 的可以用来开发手机上 GIS 的软件开发框架——可以用来开发专业的 GIS Mobile 应用。Mobile GIS 应用可以用来显示、捕捉以及更新图形信息，可以在手持设备上调用 ArcGIS Server 发布的服务。ArcGIS Server 发布的地图服务，不仅能供桌面端工具、浏览器访问，也能供一些手持设备访问，如图 14.14 所示。

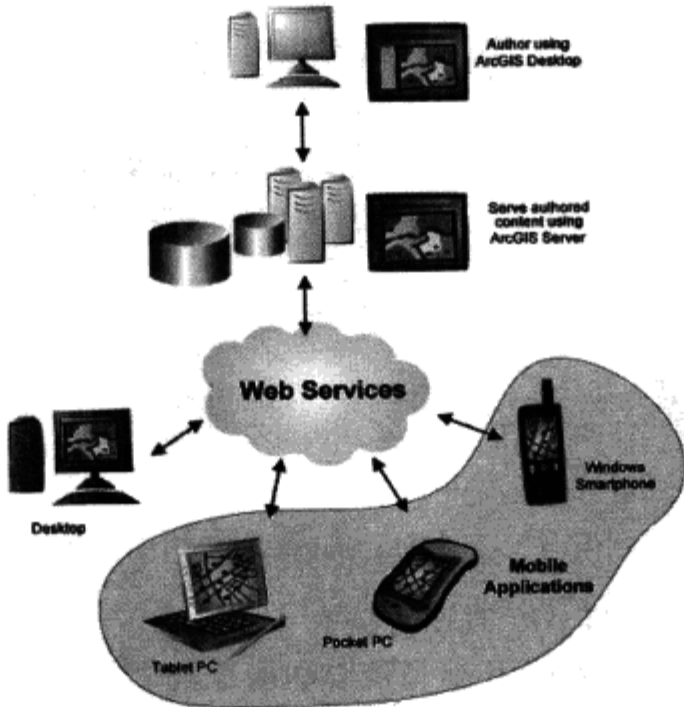


图 14.14 ArcGIS Server 服务的访问方式

传统的空间信息经常被应用到纸质地图,如地图集,还被应用到一些桌面 GIS 或者 CAD 应用系统。但是这些系统还是不够方便(无法随着人的移动而移动,特别是在野外作业时),因此有人想到将 GIS 应用搬到手持设备上,这样就形成了 Mobile GIS。

利用 ArcGIS Server 发布一个 Mobile 应用需要经过以下一些步骤。

- (1) 用 ArcMap 设计编制移动应用的地图。
- (2) 把移动应用地图发布成地图服务。
- (3) 用 Visual Studio .NET 创建基于 ArcGIS Server Mobile SDK 的应用程序。
- (4) 把应用和相关数据部署到移动设备上。

有关设计编制地图与我们通常的地图编制没有什么区别,本节对地图编制就不再重述了。发布移动的数据服务时候,记住必须选择“Mobile Data Access”选项,如图 14.15 所示。

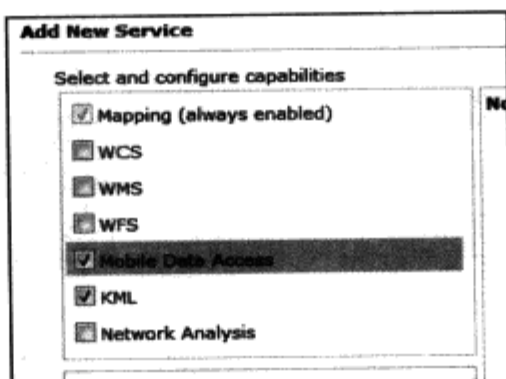


图 14.15 发布移动地图服务

Mobile ADF 的开发环境如下:

- Smart Device;
- Visual Studio .NET 2008;
- ArcGIS Server Mobile ADF。

需要说明的是,Mobile ADF 使用的应用程序框架是 .NET Compact Framework,它是 .NET Framework 的子集,有些方法或类在 .NET Framework 能用,但在小型设备上不一定能用。

下面给出在 PocketPC 环境下,创建和发布 ArcGIS Mobile 应用的实例。该类主要显示 MapControl 来显示 ArcGIS Server Map Service,并提供通用的地图浏览工具如放大、漫游和空间要素属性查询等功能。

- (1) 在 Visual Studio.NET 创建一个 Smart Device 项目,开发语言选择 C#,如图 14.16 所示。

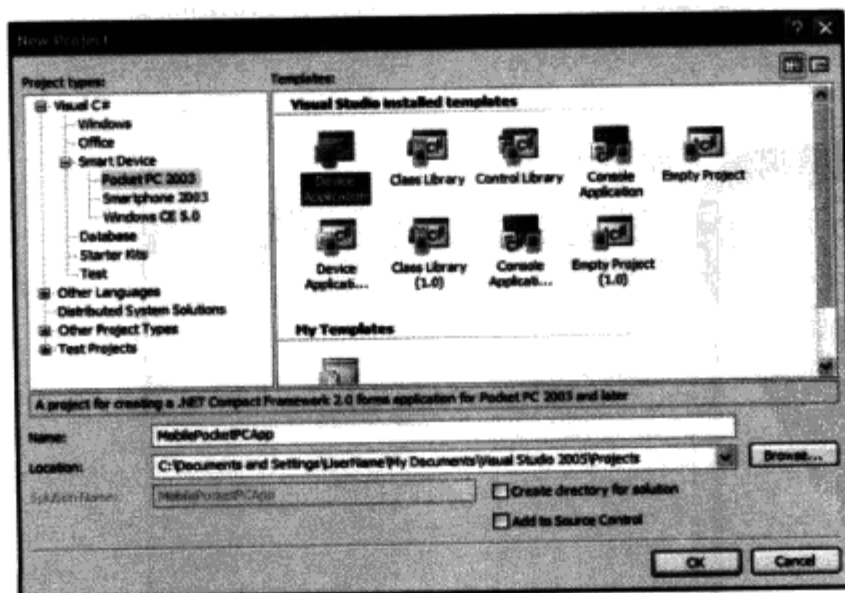


图 14.16 添加新项目

(2) 打开 ArcGIS Mobile Controls 工具箱，把相关的控件拖到窗口上，如图 14.17 所示。

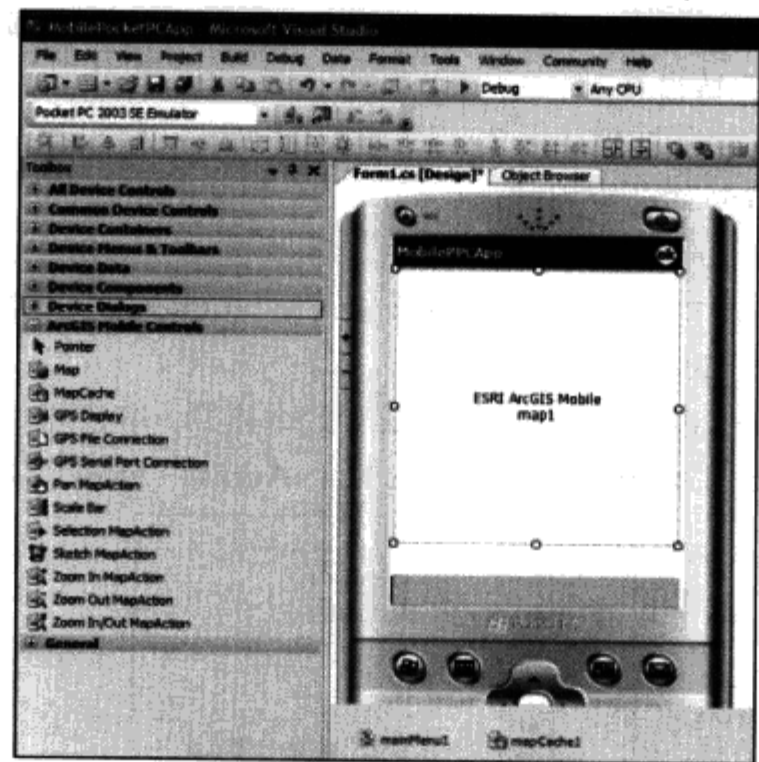


图 14.17 设计界面

- (3) 配置相关控件的属性，如 Map 控件的 MapCache 属性。
- (4) 在 From1_Load 事件编写代码来打开地图，示例程序如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    if (!mapCache1.IsValid)
    {
        MessageBox.Show("Map Cache is not valid!");
        return;
    }
    try
    {
        mapCache1.Open();
    }
    catch
    {
        MessageBox.Show("Cannot open map cache");
    }
}
```

(5) 配置 PocketPC 模拟器，如图 14.18 所示。

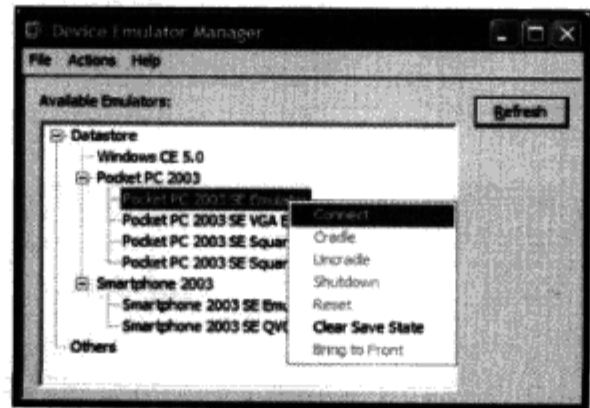


图 14.18 配置 PocketPC

(6) 添加鼠标事件。将 Mobile 的 Pan、Zoom in 和 Zoom out 组件添加到项目中，如图 14.19 所示。



图 14.19 添加鼠标事件

示例代码如下：

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    map1.CurrentMapAction = panMapAction1;
}
private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    map1.CurrentMapAction = zoomInMapAction1;
}
private void radioButton3_CheckedChanged(object sender, EventArgs e)
{
    map1.CurrentMapAction = zoomOutMapAction1;
}
```

(7) 编写空间要素属性查询，示例程序如下：

```
if (map1.CurrentMapAction != null)
    return;

Cursor.Current = Cursors.WaitCursor;
MapMouseEventArgs me = e as MapMouseEventArgs;
Envelope qEnv = new Envelope(me.MapCoordinate, me.MapCoordinate);

int mapTolerance = map1.ToMap(3);
qEnv.Resize(mapTolerance, mapTolerance);

QueryFilter qFilter = new QueryFilter(qEnv, EsriGeometricRelationship.Intersect);

string txtResult = "Identify Results: ";
int intFields;
```

```

foreach (MapLayer lyr in map1.MapLayers)
{
    FeatureLayer featLayerDT = lyr.Layer as FeatureLayer;

    if (featLayerDT == null)
        continue;

    txtResult += "\r\nLayer " + featLayerDT.Name;

    using (FeatureDataReader featReader = featLayerDT.GetDataReader(qFilter))
    {
        intFields = featReader.FieldCount;

        while (featReader.Read())
        {
            for (int i = 0; i < intFields; i++)
                txtResult += "\r\n" + featReader.GetName(i) + ": " + featReader.
GetValue(i).ToString();
        }
    }
}
Cursor.Current = Cursors.Default;
MessageBox.Show(txtResult);

```

(8) 编译并运行系统，运行结果如图 14.20 所示。

上面详细介绍了如何开发一个 ArcGIS Server Mobile ADF 应用，接下来需要把应用部署到移动设备上。ArcGIS Server Mobile 部署如图 14.21 所示。

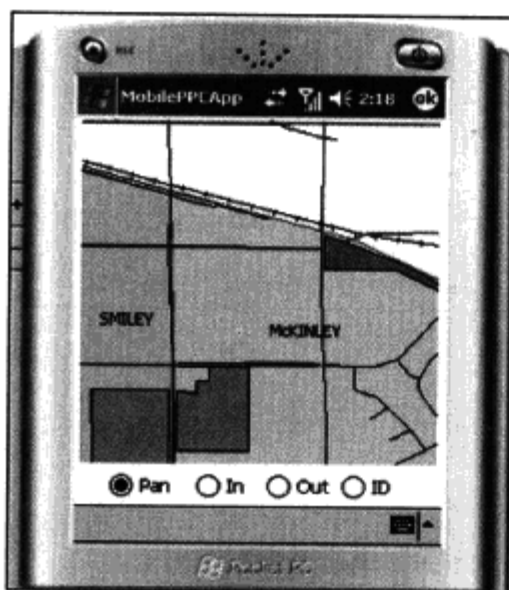


图 14.20 Mobile 应用运行效果

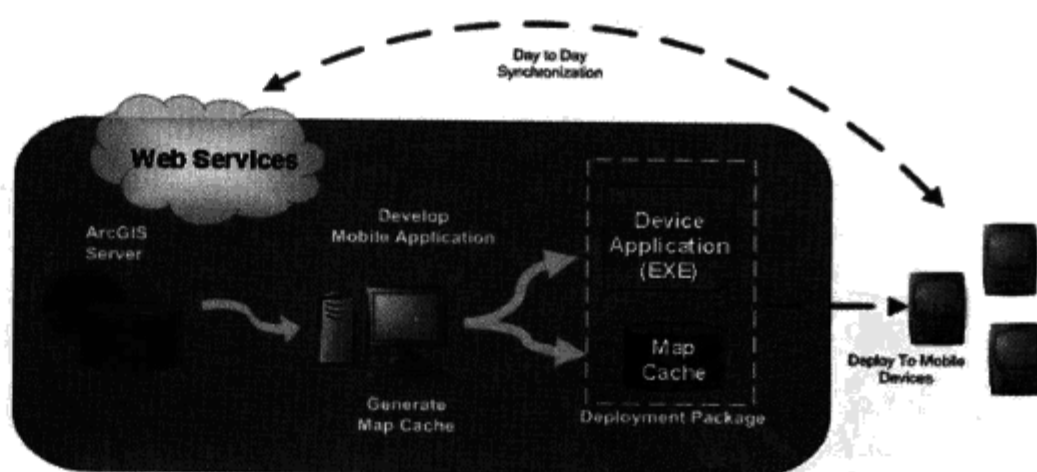


图 14.21 ArcGIS Server Mobile ADF 部署图

从上图可以看出，ArcGIS Server Mobile 应用程序部署的时候，可以把移动设备当做一个特定的应用程序。不仅要把应用程序部署到移动设备，还需要把数据部署到移动设备上。数据和应用程序的部署可以采用复制和打包的形式进行部署。移动设备进行数据编辑处理后，可以利用无线网络更新后台的数据库，或者通过同步软件与服务器进行同步更新后台数据库。

14.6 ArcGIS JavaScript API

在 14.4 节介绍 REST 时，已提及 ArcGIS Server 9.3 以全新的 REST 方式实现了一些 ArcGIS Server 通用功能和服务，便于 ArcGIS Server 客户端开发，如 JavaScript。

JavaScript API 是 ArcGIS Server 9.3 新增的一套 API 框架，为创建 WebGIS 应用提供了轻量级的解决方案，在客户端可以轻松地利用 JavaScript API 来调用 ArcGIS Server 所提供的服务，实现地图应用和地理处理功能。所有的操作都是在客户端用脚本调用服务端的接口完成的，不需要任何服务端的代码。

使用过 ArcIMS 的用户一定记得曾经用过的 ArcIMS BlueViewer 示例框架。里面的所有操作都基于客户端实现，放大、缩小、平移都是在 JavaScript 脚本中完成的，将结果返回服务器得到我们所需的图片。相同的是，JavaScript API 也是纯粹的客户端操作，但是它基于更炫的 Dojo 库，并用面向对象的方式重新封装了功能。目前 ArcGIS Server JavaScript API 已经封装好的通用功能有：

- 以用户数据和服务器端服务结合显示交互性的地图；
- 执行一个 GIS 空间分析模型并显示结果；
- 在 ArcGIS Server 服务的基础上添加新的数据；
- 空间查询和属性查询；
- 地址搜索并定位。

ArcGIS Server JavaScript API 包含的资源组件有：

- **Maps** 支持 ArcGIS Server 上地图的显示，可以在客户端按需要制定投影参考系，这样就可以轻松实现与各种标准地图服务的叠加。
- **Graphics** 绘图，通过鼠标、键盘等操作实现属性信息的展示或地理图操作。
- **Tasks** 和 ADF Tasks 类似，提供 Querying、Finding Addresses、Finding attributes、Identifying Features、Geoprocessing 等丰富的功能。

基于 Dojo 和其他库进行扩展，如利用 Dojo、Google Map API、Virtual Earth API，等等。ArcGIS Server 选择 Dojo 作为 JavaScript API 作为基础库有如下几点好处：

- Dojo 宽松的使用许可，基于标准的开发式应用。
- Dojo 功能全面，事件设计模式、矢量图形支持、显示效果、widgets、Ajax、JSON 等都是 ArcGIS Server 客户端开发所必需的，并且 Dojo 利于扩展，方便 Mashup 应用，还有大量的参考资料。
- IBM、SUN、AOL 等公司的支持，可轻松与大型系统集成。

基于 Dojo 的 JavaScript API 天生具有 Dojo 的跨浏览器特性和强大的前端制图功能，由于 Dojo 的开发性和扩展性，JavaScript API 将具有令人期待的发展前景。下面让我们来看一个基于 JavaScript API 的应用示例。

(1) 使用 Dojo 提供的样式表：

```
<style type="text/css">
  @import
"http://serverapi.arcgisonline.com/jsapi/arcgis/1.1/js/dojo/dijit/themes/tundra/tundra.css";
</style>
```

(2) 引用脚本文件:

```
<script type="text/javascript" src="http://serverapi.arcgisonline.com/jsapi/arcgis?v=1.1"></script>
```

其实,在部署 JavaScript API 时,针对不同类型的服务器代码(Java、.NET、PHP)会有不同的 Default 页面。以.NET 为例,Default.ashx 引用 3 个文件:

```
context.Response.WriteFile(context.Server.MapPath("js\"esri\"esri.js"));
context.Response.WriteFile(context.Server.MapPath("js\"dojo\"dojo\"dojo.xd.js"));
context.Response.WriteFile(context.Server.MapPath("js\"esri\"jsapi.js"));
```

(3) 对页面中添加的地图元素与功能进行定制,同样在脚本中完成,示例程序如下:

```
<script type="text/javascript">
    dojo.require("esri.map"); //引入所需要的包
    var map;
    function Init() {
        map = new esri.Map("map"); //在指定的 div 创建地图
        var layer = new esri.layers.ArcGISDynamicMapServiceLayer("http://gistest/
ArcGIS/rest/services/china/MapServer");//获得 REST API 所提供的 cache 地图服务。
        map.addLayer(layer);
    }
    dojo.addOnLoad(Init);
</script>
```

(4) 页面的 Body 区,脚本程序如下:

```
<body class="tundra">
<center>
<table style="width:99%">
<tr>
<td>
<table id="navTable" width="100%">
<tbody>
<tr valign="top">
<td id="breadcrumbs">
ArcGIS JavaScript API: china
</td>
<td align="right" id="help">
Built using the <a href="http://resources.esri.com/arcgisserver/apis/javascript
/arcgis">ArcGIS JavaScript API</a>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</table>
<br />
<div id="map" style="position:relative;width:800px; height:600px;border:1px
solid #000;"></div>
```



```
</center>
</body>
```

(5) GIS 通用功能。

Esri.map 类包含地图放大、缩小、平移等基本功能的实现,可以将这些操作和业务应用绑定。快捷方式如下:利用键盘方向键上下左右完成地图上下左右的固定单位移动,利用鼠标滚轮实现地图放大与缩小,“Shift+单击”为居中,“Shift+双击”为居中放大,“+”放大一级,“-”缩小一级。

系统的运行界面如图 14.22 所示。

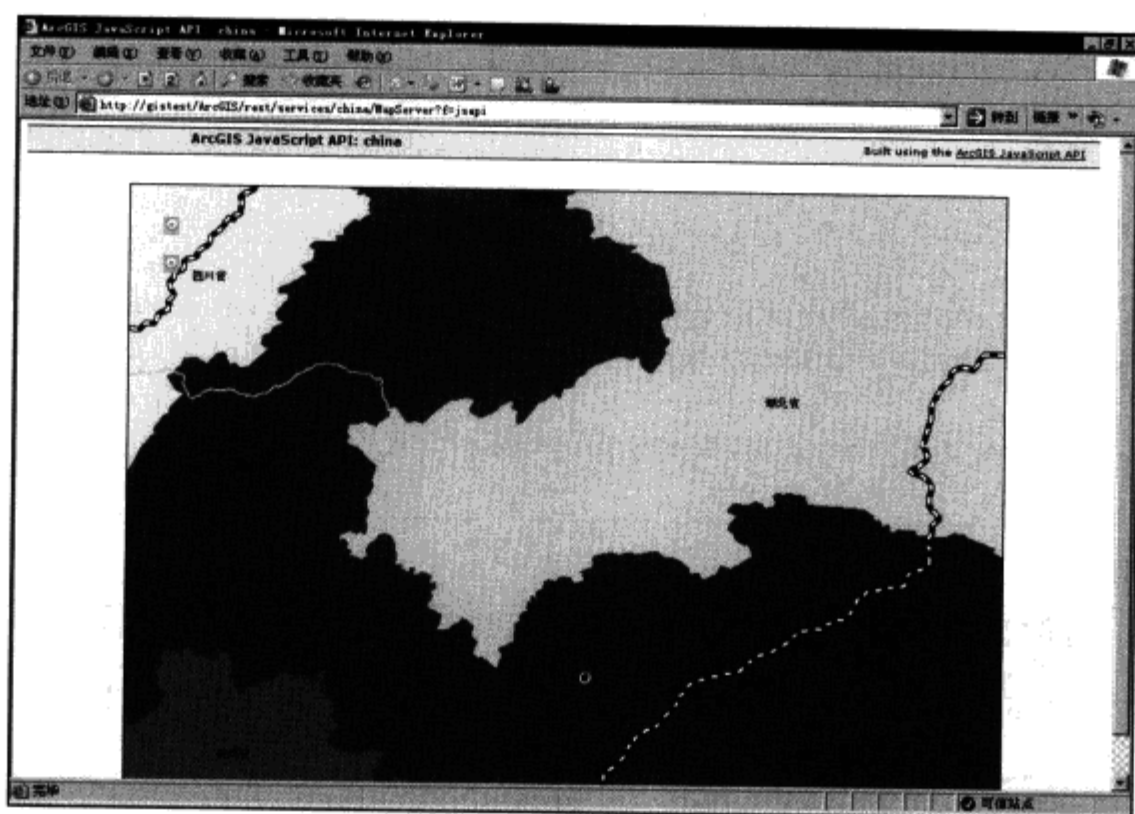


图 14.22 基于 ArcGIS Server JavaScript API 开发的 WebGIS

14.7 JavaScript Extension for Virtual Earth

前面讲过, ArcGIS Server 的地图服务可与 Virtual Earth 的服务进行叠加。Virtual Earth 是 Microsoft 公司的一款虚拟地球产品,在此不作详细的介绍。

下面介绍如何利用 ArcGIS Server JavaScript API 叠加 Virtual Earth。

(1) 引用 Virtual Earth 和 ArcGIS Server JavaScript API 的脚本:

```
<script src="http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=6.2" type="text/javascript"></script>
<script src="http://serverapi.arcgisonline.com/jsapi/ve/?v=1.3" type="text/javascript"></script>
```

(2) 打开 Virtual Earth 的地图, 脚本程序如下:

```
function OnPageLoad() {
    var centerat = new VELatLong(38.23, -85.68); //地图中心点
    map = new VEMap('mymap');
    map.LoadMap(centerat, 11, VEMapStyle.Road, false); //打开地图
}
```

(3) 页面的 Body 区，脚本程序如下：

```
<body onload="OnPageLoad()" >
  <form action="" >
    <div id='mymap' style="position:relative; width: 750px; height: 500px;">
  </div>
    <input type="button" id="addMapButton" value="Add ArcGIS Service" style=
"width: 200px;" onclick="AddMap()" />
  </form>
</body>
```

(4) 叠加 ArcGIS Server 的地图服务，脚本程序如下：

```
function AddMap() {
  agisve_dyn_service = new ITNexus.Libraries.WebADF.JavaScript.VE.
ArcGISDynamicService();

  agisve_dyn_service.CreateLayer("http://sampleserver1.arcgisonline.com/ArcGIS/rest/
services/Louisville/LOJIC_PublicSafety_Louisville/MapServer","LOJIC_PublicSafety_L
ouisville");

  agisve_dyn_service.addDynamicService(map,1,true);
}
```

系统运行效果如图 14.23 所示。

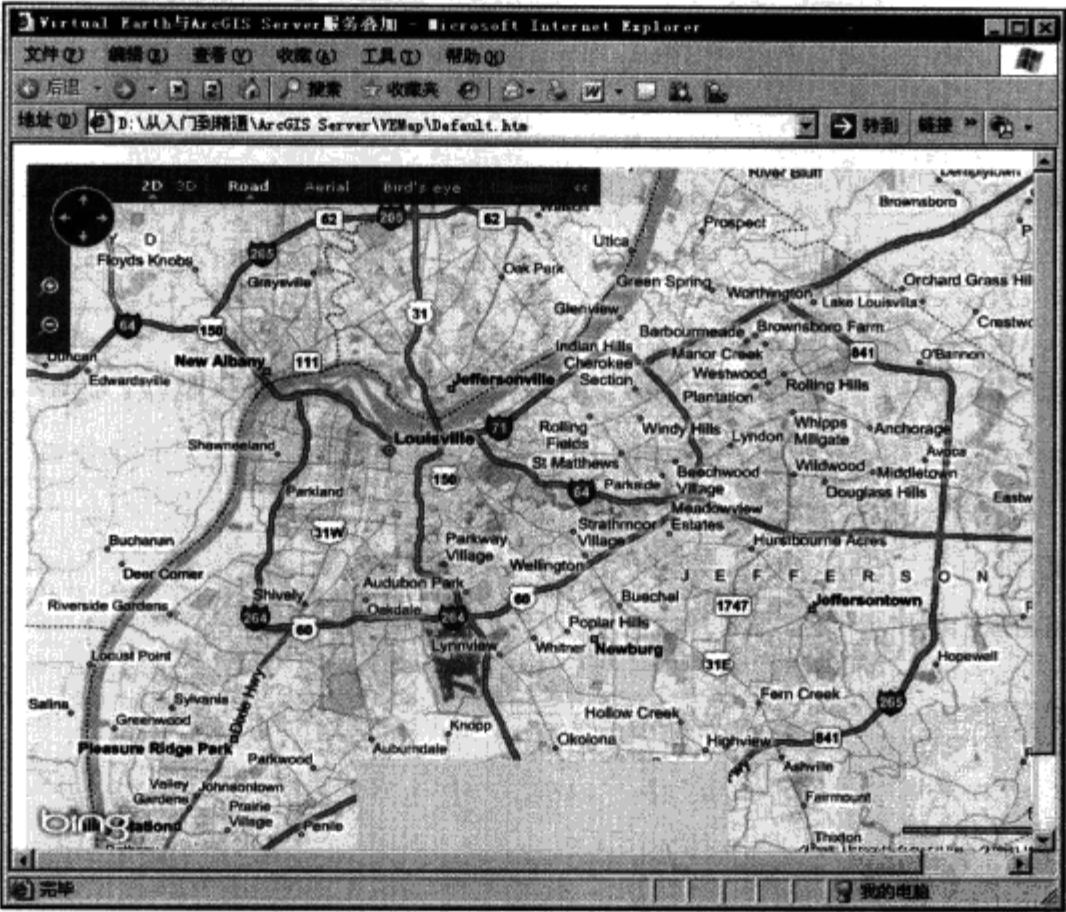


图 14.23 Virtual Earth 与 ArcGIS Server 地图服务叠加

14.8 JavaScript Extension for Google Maps API

Google Maps 是 Google 公司一款在线地图产品，它的在线地图也可以与 ArcGIS Server 的地图服务进行叠加，开发步骤与 14.7 节中描述的差不多。

(1) 引用 Google Maps 和 ArcGIS Server JavaScript API 的脚本。

```
<script
src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAUTPMg VlwRf2kxdi
70rbKHxTlkeZTPJs6szozlbsxSkFnXAV7iRSEHuAl_Iz2OGx114DKHYvlVNkcNQ"
type="text/javascript"></script>
<script src="http://serverapi.arcgisonline.com/jsapi/ve/?v=1.3" type="text/
javascript"></script>
```

(2) 打开 Google Maps 的地图，示例程序如下：

```
function initialize() {
    if (GBrowserIsCompatible()) {

        _map = new GMap2(document.getElementById("map_canvas"));

        document.getElementById("GCS").innerHTML = '<div style="cursor:pointer;
background: url(http://www.gcs-research.net/GmapsGeoMarcDemo/GCS.png);width:89px;
height:50px; overflow:hidden;" onClick="window.open(\'http://www.gcs-research.
com\');"></div>';
        var timage = new GControlPosition(G_ANCHOR_BOTTOM_LEFT, new GSize(4,36));
        timage.apply(document.getElementById("GCS"));
        _map.getContainer().appendChild(document.getElementById("GCS"));

        addMapType("Satelite", "G_SATELLITE_MAP");
        addMapType("Map", "G_NORMAL_MAP");

        var center = new GLatLng(46.585914, -112.018365);
        _map.setCenter(center, 17, G_NORMAL_MAP);

        //Sink Event Listeners
        GEvent.addListener(_map, "click", function(overlay,point){mapClick (overlay,
point)}));

        _map.addControl(new GLargeMapControl());
        _map.addControl(new GMapTypeControl());

        _map.removeMapType(G_NORMAL_MAP);
        _map.removeMapType(G_SATELLITE_MAP);
        _map.removeMapType(G_HYBRID_MAP);

    }
}
```

(3) 页面的 Body 区，脚本程序如下：

```
<body onload="initialize()" onunload="GUnload()">
    Click on the map to identify and buffer parcels...
    <div id="map_canvas" style="width: 1000px; height: 500px"></div>
    <div id="GCS" style="display:block"></div>
```

本网站是 Google Maps 与 ArcGIS Server 集成，ArcGIS Server 做地理分析，Google Map 显示分析结果

</body>

(4) 利用 ArcGIS Server 进行查询，示例程序如下：

```
function InvokeASHX(cmd,param)
{
    InitXmlHttp();
    xmlhttp.onreadystatechange= XMLHttpRequestCompleted;
    xmlhttp.open("GET", "ParcelDemoHandler.ashx?cmd=" + cmd + "&param=" + param,
true);
    xmlhttp.send(null);
}
```

系统运行如图 14.24 所示。

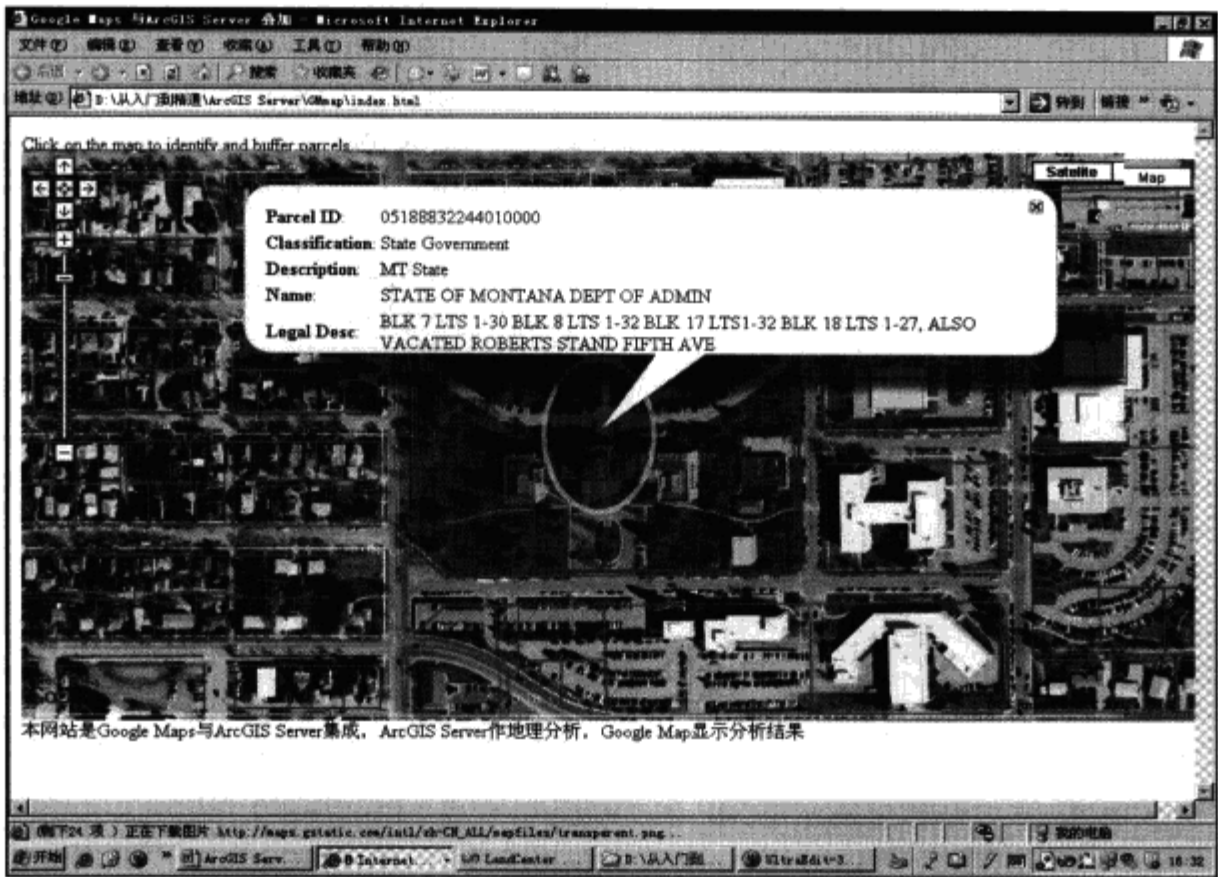


图 14.24 Google Map 与 ArcGIS Server 叠加

14.9 小结

本书重点介绍了 ArcGIS Server 的.NET Web ADF 开发方式，而本章则介绍 ArcGIS Server 的其他一些开发方式，如 Java、SOAP、REST。重点讲述了 JavaScript API 的开发及其与当前流行的在线地图应用集成。ArcGIS Server 提供众多的开发模式，为用户应用程序的开发提供了更多的选择。每种开发方式有其优点也有其不足的地方，用户在选择开发模式的时候应根据项目的实际情况以及开发人员对开发语言的熟悉程度来选择恰当的开发方式。

第 15 章

ArcGIS Server For Java

本书主要讲述 ArcGIS Server 基于 .NET 的 ADF 开发，但 ArcGIS Server 不仅提供基于 .NET 的 ADF 开发，还提供基于 Java 的 ADF 开发，本章主要讲述基于 Java 的 ADF 开发。

15.1 ArcGIS Server Java 开发基础

C# 是微软用来与 Java 抗衡的武器，因为二者在很大程度上有着惊人的相似，尽管如此，两者不同的地方也很多。下面简单地将 C# 和 Java 进行比较。

相同点。

- 二者都编译成跨平台、跨语言的代码，并且代码只能在一个受控制的环境中运行。
- 自动回收内存，并且消除了指针。
- 都不需要头文件，所有的代码都被“包（package）”限制在某个范围内。
- 所有的类都从对象派生出来，并且必须使用 New 关键字分配内存。
- 用对象加锁的方式来支持多线程，都具有接口的概念。

二者区别。

- 在 C# 中，是以 Main 方法定位入口的，Java 中公共类的入口是 main。
- 在 Java 中，一个文件中只能有一个 public 类，而且这个类的名称必须与文件名一样，C# 则没有此限制。
- 在 Java 中，包的名字同时也是实际存在的实体，它决定了放置 Java 文件的目录结构；C# 中，物理的包和逻辑的名称之间是完全分离的。

下面我们来看一个简单的 Servlet 的配置文件：

```
<servlet>
  <servlet-name>Faces Servlet</servlet-name>
  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
```

```

    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.jsf</url-pattern>
    </servlet-mapping>

```

该 Servlet 映射表示，将所有对后缀名为 jsf 的请求，都交由 javax.faces.webapp.FacesServlet 来处理。在访问某个 JSF 页面的时候，映射到某个目录即可。这个目录在 web.xml 进行了设置，比如下面的配置就可以把所有的对 faces 的请求，让 Faces Servlet 来处理：

```

<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
</servlet-mapping>

```

所以，如果在根目录下放了一个 hello.jsp 页面，这个 jsp 页面使用了 JSF 框架，那么在访问这个 jsp 页面时用 http://servername/face/hello.jsp 就可以了，容器会把这样的请求提交给 Faces Servlet 来处理。

了解 Servlet 是如何处理 JSF 请求后，那么接下来我们来看看，让 JSF 程序运行需要什么必要条件：一个必需的 Servlet 容器，通常的选择是 Tomcat，价格便宜量又足。当然如果有条件的话，IBM 的 J2EE 容器 Websphere 和 Bea 公司的 Weblogic 也是不错的选择。除了一个好的容器，还需要一个 JSF 的实现，多采用 Sun 公司的 Reference Implementation。也可以选择功能更加强大的 Myfaces。ESRI 的 ArcGIS Server ADF 遵循标准的 JSF，所以 ADF 可以运行在 Sun 的 JSF 实现，或者 Myfaces 上。

由于 ArcGIS Server 的 Java ADF 开发会用到 JSF 的知识，因此下面结合 ArcGIS Server 的开发，介绍有关 JSF 的知识。

Struts 和 JSF/Tapestry 都属于表现层框架，这两种分属不同性质的框架，后者是一种事件驱动型的组件模型，而 Struts 只是单纯的 MVC 模式框架。

事件是指从客户端页面（浏览器）由用户操作触发的事件，Struts 使用 Action 来接收浏览器表单提交的事件，这里使用了 Command 模式，每个继承 Action 的子类都必须实现一种方法：execute。在 Struts 中，一个表单（Form）对应一个 Action 类（或 DispatchAction）。换句话说，在 Struts 中一个表单只能对应一个事件，Struts 事件方式称为 Application event，Application event 和 component event 相比是一种粗粒度的事件。

Struts 重要的表单对象 ActionForm 是一种对象，它代表了一种应用，这个对象中至少包含几个字段，这些字段是 JSP 页面表单中的 input 字段。因为一个表单对应一个事件，所以，当我们需要将事件粒度细化到表单中这些字段时，也就是说，一个字段对应一个事件时，单纯使用 Struts 就不太可能，当然通过结合使用 JavaScript 也是可以实现的。

而这种情况使用 JSF 就可以方便实现：

```

<h:inputText id="userId" value="#{login.userId}">
    <f:valueChangeListener type="logindemo.UserLoginChanged" />
</h:inputText>

```

#{login.userId} 表示从名为 login 的 JavaBean 的 getUserId 获得的结果，这个功能使用 struts 也可以实现，name="login" property="userId"。关键是第二行，这里表示如果 userId 的值改变并

且确定提交后，将触发调用类 UserLoginChanged 的 processValueChanged(...)方法。

JSF 可以为组件提供两种事件：Value Changed 和 Action。相当于 Struts 中表单提交 Action 机制，它的 JSF 写法如下：

```
<h:commandButton id="login" commandName="login">
    <f:actionListener type=" logindemo.LoginActionListener" />
</h:commandButton>
```

从代码可以看出，这两种事件是通过 Listener 观察者模式贴在具体组件字段上的，而 Struts 类事件是原始的一种表单提交 Submit 触发机制。如果说前者比较语言化（编程语言习惯做法类似 Swing 编程），则后者属于 Web 化，因为它来自 Html 表单。

Struts 和 JSF 都是一种框架，JSF 必须有两种包：JSF 核心包、JSTL 包（标签库）。此外，JSF 还将使用 Apache 项目的一些 commons 包，这些 Apache 包只要部署在服务器中即可。

JSF 包下载地址为 <http://java.sun.com/j2ee/javaxserverfaces/download.html>，选择其中的 Reference Implementation。

JSTL 包在 http://jakarta.apache.org/site/downloads/downloads_taglibs-standard.cgi 中下载。

所以，从 JSF 的驱动包组成来看，其开源基因也占据很大的比重，JSF 是一个 Sun 工业标准和开源之间的混血儿。

上述两个地址下载的 jar 合并在一起就是 JSF 所需要的全部驱动包了。在 ArcGIS Server 项目下的 lib 目录中有相关的库文件如下：commons-beanutils.jar、commons-collections.jar、commons-digester.jar、commons-lang-2.0.jar、commons-logging.jar、jsf-api.jar、jsf-impl.jar、jstl.jar、standard.jar 等。与 Struts 的驱动包一样，这些驱动包必须位于 Web 项目的 Web-INF/lib 中，必须在 Web.xml 中有如下配置：

```
<web-app>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.faces</url-pattern>
    </servlet-mapping>
</web-app>
```

这里和 Struts 的 Web.xml 配置相似。正如 Struts 的 struts-config.xml 一样，JSF 也有类似的 faces-config.xml 配置文件：

```
<faces-config>
    <navigation-rule>
        <from-view-id>/index.jsp</from-view-id>
        <navigation-case>
            <from-outcome>login</from-outcome>
            <to-view-id>/welcome.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
    <managed-bean>
```

```
<managed-bean-name>user</managed-bean-name>
<managed-bean-class>com.corejsf.UserBean</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
</managed-bean>
</faces-config>
```

上面介绍了客户端架构，ArcGIS Server Java Web ADF 的处理流程，如图 15.1 所示。

通过“a:map”标签来定义一个 Map 组件，在 ADF 中，其对应的 JSF 组件为 MapControl。MapControl 中包含 WebMap、MapRenderer、XslUrl 等属性，WebMap 负责维护地图的功能，MapRenderer 负责将其输出为 XML，最后再通过 XslUrl 指定的 XSL 文件将 XML 解释为 DHTML 输出。

打开“/src/xsl/”目录，会发现下面有很多“*.xsl”文件，其中的“map.xsl”就是 MapControl 渲染时默认使用的 XSL 文件。当用户向服务器发送请求的时候，JSF 会首先创建或者还原 MapControl 视图对象，然后再使用其 MapRenderer 按照 XSL 文件渲染成 DHTML 响应输出给浏览器。

无论是.NET 还是 Java，Ajax 在 ArcGIS Server 的开发中都扮演关键角色，Ajax 在 Java ADF 的生命周期如图 15.2 所示。

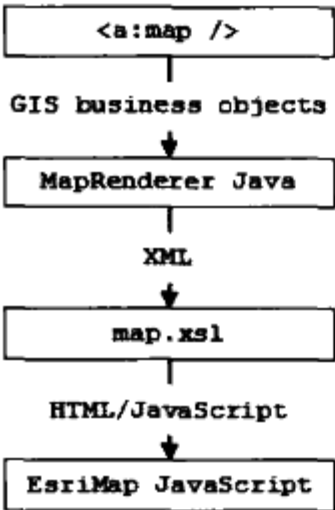


图 15.1 ADF 组件处理流程

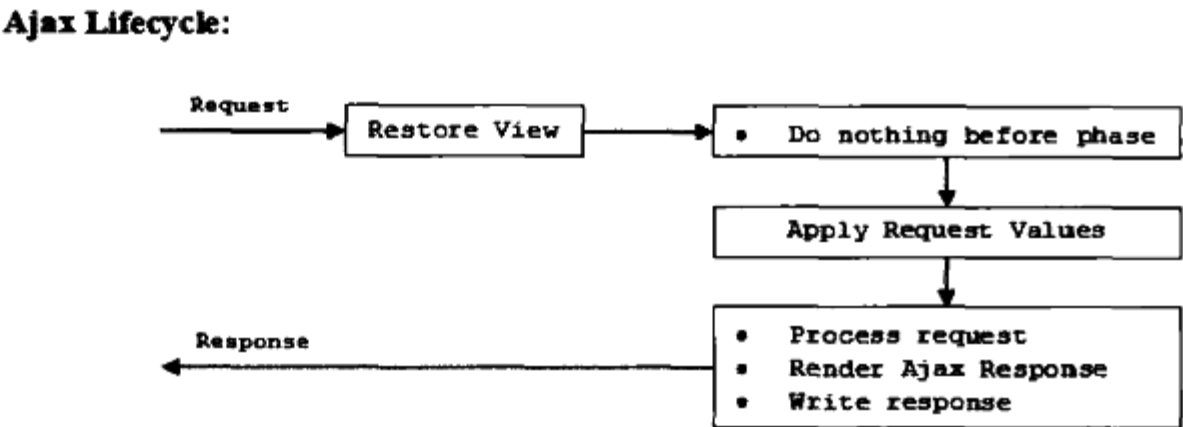


Figure showing JSF lifecycle with Ajax.

图 15.2 JSF 中 Ajax 的生命周期

Ajax 生命周期在服务器端是单独执行的。浏览器发出一个携带一定参数的请求，服务器端收到客户端的请求后，进行相应的处理，并把处理结果写成 XML 后返回客户端。这种响应可以由任何类型的 JSF PhaseListeners 或者 Servlet 来解析或者就是静态的 XML 文件。

下面给出一个简单的 Servlet 来处理客户端的请求：

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws java.io.IOException {

    String action = req.getParameter("action");
    String item = req.getParameter("item");

    if ((action != null)&&(item != null)) {

        // Add or remove items from the Cart
        if ("add".equals(action)) {
            cart.addItem(item);
```

```

    } else if ("remove".equals(action)) {
        cart.removeItem(item);
    }
}

// Serialize the Cart's state to XML
String cartXml = cart.toXml();

// Write XML to response.
res.setContentType("application/xml");
res.getWriter().write(cartXml);
}

```

如果使用 JSF 框架，一般不会再通过 Servlet 来实现服务器端的响应，而使用 JSF 中的 `PhaseListener` 来进行处理。`PhaseListener` 是一个特殊的接口，在请求开始处理，请求处理结束时可以插入自己的代码，这个接口最重要的两个方法（处理结束和处理开始两个方法）如下：

```

void afterPhase(javax.faces.event.PhaseEvent event)
    Handle a notification that the processing for a particular phase has just been
    completed. void beforePhase(javax.faces.event.PhaseEvent event)
    Handle a notification that the processing for a particular phase of the request
    processing lifecycle is about to begin.

```

开发人员可以根据需要加入自己的代码，对用户的请求进行处理。如可以在 `afterPhase` 方法中，插入下面的代码对数据进行处理：

```

FacesContext facesContext = phaseEvent.getFacesContext();
ExternalContext externalContext = facesContext.getExternalContext();
Map paramMap = externalContext.getRequestParameterMap();

```

有了这些基本知识，下面我们就可以开始进行 ArcGIS Server 的 Java Web ADF 开发了。

15.2 ArcGIS Server Java ADF 开发初步——自定义工具

在 ArcGIS Server 生成的 Web App 中，自定义工具可以分为两类。

- 命令 (Command) 不与用户进行交互。
- 工具 (Tool)：通过界面与用户进行交互。

下面来看看工具的定义，程序如下：

```

<a:toolbar id= "Toolbar" mapid = "map" orientation = "VERTICAL">
<a:toolid= "pan"
toolTip="漫游"
defaultImage="images/tasks/maptools/pan.png"
hoverImage = "images/tasks/maptools/panU.png"
selectedImage = "images/tasks/maptools/panD.png"
clientAction="EsriMapContinuousPan"
serverAction="com.esri.adf.web.faces.evnet.PanToolAction"
clientPostBack="true"/>
</a:toolbar>

```

Tool 定义时两个最重要的属性就是 `clientAction` 和 `serverAction`。`clientAction` 定义的是在浏

览器端执行的操作，比如这里的“EsriMapContinuousPan”代表浏览器端执行的是连续漫游操作——当然，如果需要在地图中画个多边形之类的操作也可以使用其他的定义，可用的操作如表 15.1 所示。

表 15.1 浏览器端支持的操作列表

EsriEditingLine	编辑直线	EsriMapLine	画直线
EsriEditingPoint	编辑点	EsriMapMouseWheel	滚轮滚动
EsriEditingPolygon	编辑多边形	EsriMapOval	画椭圆
EsriEditingPolyline	编辑多线	EsriMapPan	漫游
EsriMapCircle	画圆	EsriMapPoint	画点
EsriMapContinuousPan	连续漫游	EsriMapPolygon	画多边形
EsriMapImage	添加图片	EsriMapPolyline	画多线
EsriMapKeyNavigation	键盘导航	EsriMapRectangle	画矩形

当用户在浏览器执行操作后，ADF 会把这个操作相关的信息发送到服务器，然后 Tool 的 serverAction 属性中定义的这个类就会起作用。比如上面的这个 Tool，服务器接收到请求后，会通过 PanToolAction 的定义对请求的参数进行处理（比如向右漫游 100 个像素），经过一系列计算后它更新服务器端 WebMap 等对象的状态，所有工作完成后再调用 WebContext 的 refresh 方法进行刷新，随后向浏览器输出响应信息。

serverAction 指向的类需要实现 MapToolAction 接口，继承的接口如下：

```
public interface com.esri.adf.web.faces.event.MapToolAction{
    void execute(MapEvent event);
}
```

前面已经介绍 Tool 与 Command 的区别在于它与地图进行交互后再执行命令，而 Command 则不需要交互直接执行命令。最简单的 Tool 和 Command 的对比是放大是一个 Tool，而全图显示是一个 Command。

写一个工具，实现 MapToolAction 接口，实现接口的类程序如下：

```
public class MyTool implements MapToolAction {

    public void execute(MapEvent event) {

        WebContext ctx = event.getWebContext();
        WebGeometry screenGeom = event.getWebGeometry();
        WebGeometry mapGeom = screen.toMapGeometry(ctx.getWebMap());
    }

}
```

通过 MapEvent 的 getWebGeometry 方法可以得到客户端所画的多边形，此时得到的是屏幕坐标，然后再通过 toMapGeometry 方法转换为地图坐标。

ClientActionArgs 是一个基类，其所有的子类对应于每一个客户端的操作。每个基类需要实现它的抽象方法。最重要的两个方法是 init()和 getWebGeometry()，init 从 request 中得到所有的参数，getWebGeometry 根据不同的参数，构建不同的 geometry 返回。ADF 会根据不同的客户端操作，创建不同的 ClientActionArgs 类。ClientActionArgs 有一个静态方法如下：

```
getClientActionArgs(java.lang.String clientAction, java.util.Map requestParameters,
java.lang.String controlId)
```

ADF 就是用这个静态方法，根据不同的 clientAction 得到 ClientActionArgs 的。

由此可以看出实现自定义工具其实很简单，例如要实现在客户端画一个矩形，并查看矩形内包含多少几何对象，客户端的脚本代码如下：

```
<a:tool id="countFeature" defaultImage="images/selection.gif" hoverImage=
"images/selectionU.gif" selectedImage="images/selectionD.gif" clientAction=
"EsriMapRectangle" serverAction="com.cj.ucdemo.CountFeatureTool" clientPostBack=
"true"/>
```

服务器端实现矩形查找几何对象的个数，程序如下：

```
package com.hzg.demo;

import java.rmi.RemoteException;
import com.esri.adf.web.ags.ADFAGSException;
import com.esri.adf.web.ags.data.AGSMapResource;
import com.esri.adf.web.data.WebContext;
import com.esri.adf.web.data.geometry.WebExtent;
import com.esri.adf.web.faces.event.MapEvent;
import com.esri.adf.web.faces.event.MapToolAction;
import com.esri.arcgisws.EnvelopeN;
import com.esri.arcgisws.EsriSearchOrder;
import com.esri.arcgisws.EsriSpatialRelEnum;
import com.esri.arcgisws.MapServerPort;
import com.esri.arcgisws.SpatialFilter;

public class CountFeatureTool implements MapToolAction {

    WebContext context=null;
    int countedFeatures;

    private void countFeatures(WebExtent extent){

        //得到 MapServer

        AGSMapResource agsMap = ((AGSMapResource)context.getResources().
get("ags1"));

        MapServerPort mapServer = agsMap.getMapServer();
        //构建一个矩形
        EnvelopeN env = new EnvelopeN(extent.getMinX(),
extent.getMinY(), extent.getMaxX(), extent.getMaxY(),
        null, null, null, null, null);

        //已构建矩形来过滤图层
        SpatialFilter spatialFilter = new SpatialFilter();
        spatialFilter.setSpatialRel(EsriSpatialRelEnum.esriSpatialRel
Intersects);

        spatialFilter.setWhereClause("");
```

```

        spatialFilter.setSearchOrder(EsriSearchOrder.esriSearchOrderSpatial);

        spatialFilter.setSpatialRelDescription("");
        spatialFilter.setGeometryFieldName("");
        //Set the envelope as the geometry
        spatialFilter.setFilterGeometry(env);

        //MapServer::queryFeatureCount() executes on the server and can throw a Remote
        Exception

        try{
            //得到矩形中几何对象的个数
            int layerId = 0;
            this.countedFeatures =
        mapServer.queryFeatureCount(mapServer.getDefaultMapName(), layerId, spatial
        Filter);

            System.out.println("你选择了 "+countedFeatures+" 要素");

        }catch(RemoteException rme){
            //Rethrow this as ADFAGSException so that it can participate in the exception
            framework
            throw new ADFAGSException("Could not execute MapServer::queryFeatureCount()",
            rme);
        }
    }

    public void execute(MapEvent arg0) throws Exception {
        // TODO Auto-generated method stub
        try{
            this.context=arg0.getWebContext();
            WebExtent ex=(WebExtent)arg0.getWebGeometry();
            ex=(WebExtent)ex.toMapGeometry(arg0.getWebContext().getWebMap
        ());

            this.countFeatures(ex);
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
}

```

15.3 ArcGIS Server Java 自定义 Task

Task 是实现业务逻辑的重要部分。通俗地说，Task 可以让输入的一些参数，通过 Command 和 Tool 的组合，帮助用户完成特定的“任务”并返回结果，这就是所谓的工作流程。

由此可见，Task 的生命周期始于客户端输入一些参数，如图 15.3 所示。

图 15.3 所示是一个简单的 Task 请求面板，在面板上有一些参数（一般以文本框、组合框等形式出现）和一些 Command 或 Tool。

在 IDE 中新建一个 Task 最简单的方法就是使用菜单“File”→“New”→“Task”，然后在“Name”文本框中输入 Task 的名字，如图 15.4 所示。

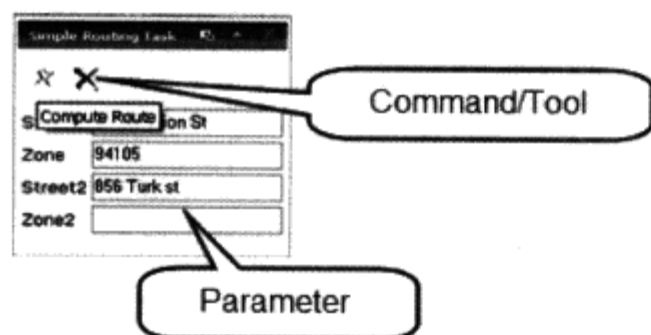


图 15.3 Task 客户端输入

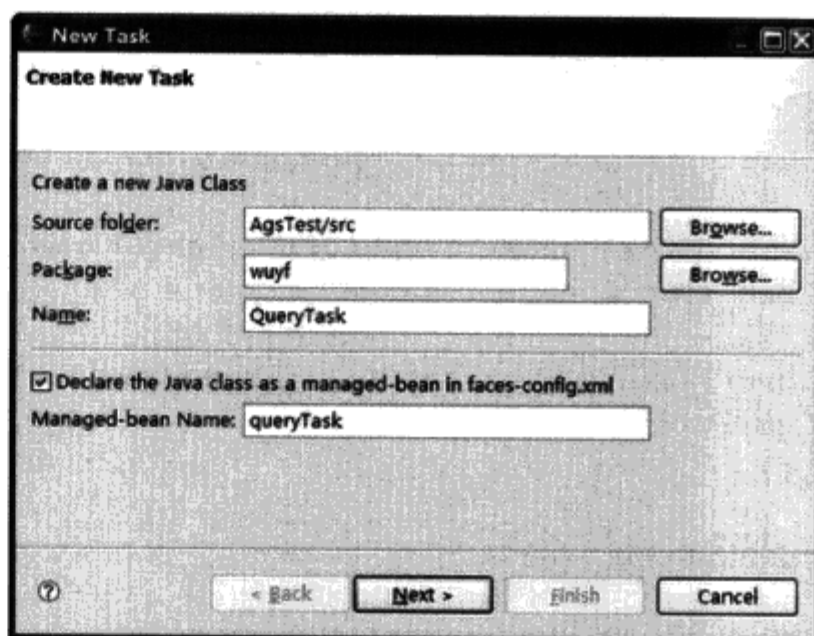


图 15.4 创建新的 Task

单击“Next”按钮，系统弹出“New Task”属性设置对话框，可以在这里设置 Task 的名字等各项参数，如图 15.5 所示。

单击“Finish”按钮后，系统完成自定义 Task 的创建，在 Java 源文件目录下多了两个 Java 文件：QueryTask.java、QueryTaskTaskInfo.java。Task 的面板如图 15.6 所示。

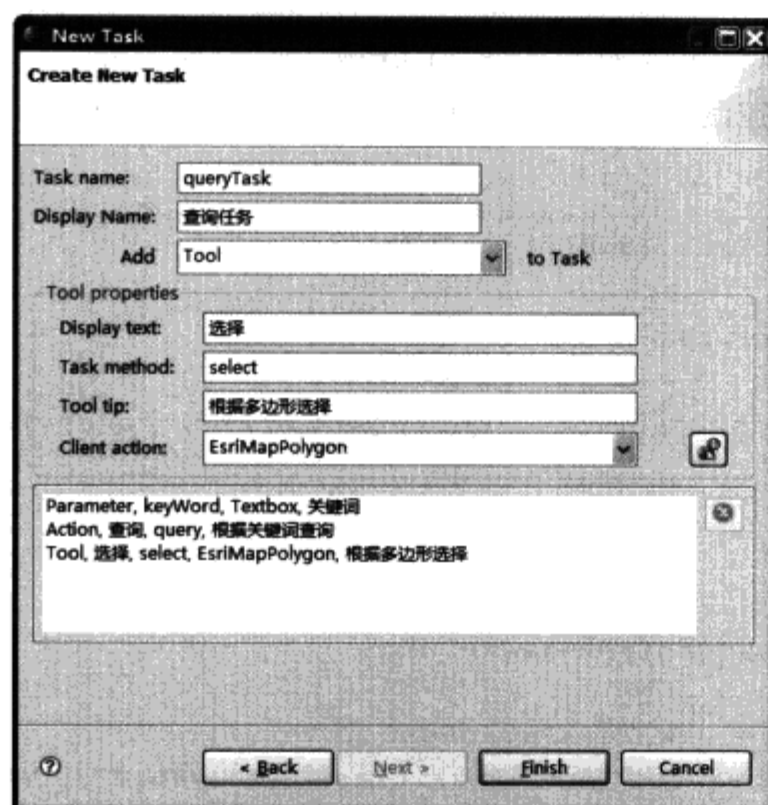


图 15.5 设置 Task 参数

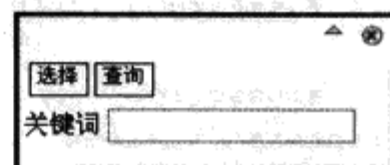


图 15.6 Task 面板

生成两个 Java 文件后，需要在这两个文件中实现功能。如实现上例中根据属性查找的示例程序：

```
public void query(TaskEvent event)
```

```
{
WebContext webContext = event.getWebContext();

WebQuery webQuery = (WebQuery) webContext.getAttribute("query");
TextCriteria textCriteria = new TextCriteria();
textCriteria.setMaxRecordCount(100);
textCriteria.setSearchText(this.keyWord);

List listQueryResult = webQuery.query(textCriteria, webQuery.getQueryLayers());

webContext.getWebGraphics().clearGraphics();
for (int i = 0; i <>
{
QueryResult queryResult = listQueryResult.get(i);
queryResult.highlight();
}

webContext.refresh();
}
```

从 TaskEvent 中可以直接获得 WebContext 对象；从 WebContext 开始可以执行一系列的操作，其中用到了 keyWord 这个属性，keyWord 对应的就是用户在 Task 面板的文本框中输入的关键词；最后将查询结果在地图上通过 Graphic 高亮显示，并通过 WebContext 的 refresh 方法输出到浏览器。

这里涉及 Parameter 和 Command 的使用；至于 Tool，唯一的区别就是 WebContext 是从 MapEvent 中获得，这里就不详细介绍了。

Task 的功能类中还有一个属性，IDE 默认生成的名字是 TaskInfo，它对应另外一个 Java 类（如前面的 QueryTaskTaskInfo）。这个类主要控制 Task 的渲染。下面来看看 QueryTaskTaskInfo.Java 的源程序：

```
public class QueryTaskTaskInfo extends SimpleTaskInfo
{
public TaskDescriptor getTaskDescriptor()
{
TaskDescriptor descriptor = new TaskDescriptor(QueryTask.class, "queryTask", "
查询任务");
return descriptor;
}

public TaskParamDescriptorModel[] getParamDescriptors()
{
TaskParamDescriptor[] descriptors = new TaskParamDescriptor[1];
descriptors[0] = new TaskParamDescriptor(QueryTask.class, "keyWord", "关键词",
"getKeyWord", "setKeyWord");
return descriptors;
}

public TaskActionDescriptorModel[] getActionDescriptors()
{
}
```

```

TaskActionDescriptor[] descriptors = new TaskActionDescriptor[1];
descriptors[0] = new TaskActionDescriptor(QueryTask.class, "query", "查询");
return descriptors;
}

public TaskToolDescriptorModel[] getToolDescriptors()
{
    TaskToolDescriptor[] descriptors = new TaskToolDescriptor[1];
    descriptors[0] = new TaskToolDescriptor(QueryTask.class, "select", "选择",
"EsriMapPolygon");
    return descriptors;
}
}

```

从上面的源代码可以看出，TaskInfo 类中包含 4 个主要的属性。

- TaskDescriptor 对象：Task 的外观，如标题等。
- TaskParamDescriptor[] 数组：描述参数的外观。
- TaskActionDescriptor[] 数组：描述 Command 的外观。
- TaskToolDescriptor[] 数组：描述 Tool 的外观。

可以通过修改这些参数的值来更改外观，如下面程序所示：

```

descriptors[0].setRendererType(TaskToolDescriptor.IMAGE_RENDERER_TYPE);
descriptors[0].setDefaultImage("images/tasks/maptools/polygon.gif");
descriptors[0].setHoverImage("images/tasks/maptools/polygonU.gif");
descriptors[0].setSelectedImage("images/tasks/maptools/polygonD.gif");
descriptors[0].setDisabledImage("images/tasks/maptools/polygonX.gif");

```

下面给出一个实现地图居中的完整示例程序：

MyTask.Java 源程序：

```

package com.example;

import java.util.LinkedHashMap;
import java.util.Map;

import com.esri.adf.web.data.WebContext;
import com.esri.adf.web.data.WebMap;
import com.esri.adf.web.data.geometry.WebExtent;
import com.esri.adf.web.data.geometry.WebGeometry;
import com.esri.adf.web.data.geometry.WebPoint;
import com.esri.adf.web.faces.event.MapEvent;
import com.esri.adf.web.faces.event.TaskEvent;
import com.esri.adf.web.tasks.MapToolsTask;

public class MyTask extends MapToolsTask{

    double zoomFactor = 0.8;

    //super.zoomIn;

```

```
public void zoomin(MapEvent event){

    super.zoomIn(event);
}

public double getZoomFactor() {
    return zoomFactor;
}

public void setZoomFactor(double zoomFactor) {
    this.zoomFactor = zoomFactor;
}

public void zoom(TaskEvent event){
    /***Exercise 6 step 3.a: get WebContext and WebMap
    WebContext webContext = event.getWebContext();
    WebMap webMap = webContext.getWebMap();

    /***Exercise 6 step 3.a: get current extent
    WebExtent webExtent = webMap.getCurrentExtent();

    /***Exercise 6 step 3.a: expand extent by factor
    webExtent.expand(zoomFactor);

    /***Exercise 6 step 3.a: set new extent
    webMap.setCurrentExtent(webExtent);

    /***Exercise 6 step 3.a: refresh context
    webContext.refresh();
}

public Map getZoomFactors() {
    Map factors = new LinkedHashMap();
    for(int i = 1; i<=10;i++) {
        Double factor = new Double((double)(i * 2)/10);
        factors.put(factor, factor);
    }
    return factors;
}

public void centerAt(MapEvent event){
    /***Exercise 6 step 4.a: get WebContext and WebMap
    WebContext webContext = event.getWebContext();
    WebMap webMap = webContext.getWebMap();

    /***Exercise 6 step 4.a: get screen geom (in pixels)
    WebGeometry screenGeom = event.getWebGeometry();

    /***Exercise 6 step 4.a: convert to map geom
    WebPoint pnt = (WebPoint)screenGeom.toMapGeometry(webMap);
```

```

        /***Exercise 6 step 4.a: center extent around user click
        webMap.centerAt(pnt, 1.0);

        /***Exercise 6 step 4.a: refresh context
        webContext.refresh();
    }

}

```

MyTaskTaskInfo.java 源代码:

```

package com.example;

import com.esri.adf.web.data.tasks.SimpleTaskInfo;
import com.esri.adf.web.data.tasks.TaskActionDescriptor;
import com.esri.adf.web.data.tasks.TaskActionDescriptorModel;
import com.esri.adf.web.data.tasks.TaskDescriptor;
import com.esri.adf.web.data.tasks.TaskParamDescriptor;
import com.esri.adf.web.data.tasks.TaskParamDescriptorModel;
import com.esri.adf.web.data.tasks.TaskToolDescriptor;
import com.esri.adf.web.data.tasks.TaskToolDescriptorModel;
import com.esri.adf.web.faces.event.ClientActions;
import com.esri.adf.web.tasks.MapToolsTaskInfo;
import com.exercise06.NavigationTask;

public class MyTaskTaskInfo extends SimpleTaskInfo {

    public TaskToolDescriptorModel[] getToolDescriptors() {

        /***Exercise 6 step 4.b: create TaskToolDescriptor array
        TaskToolDescriptor[] toolDescArray = new TaskToolDescriptor[2];

        /***Exercise 6 step 4.b:populate the TaskToolDescriptor array

        TaskToolDescriptor toolDesc=newTaskToolDescriptor(NavigationTask.class, "centerAt",
"点图为中心", ClientActions.MAP_POINT);

        toolDescArray[0] = toolDesc;

        TaskToolDescriptor toolDesc1 = new TaskToolDescriptor(NavigationTask.class,
"zoomin", "放大", ClientActions.MAP_RECTANGLE);

        toolDescArray[1] = toolDesc1;

        /***Exercise 6 step 4.b: return the TaskToolDescriptor array
        return toolDescArray;

    }

    public TaskActionDescriptorModel[] getActionDescriptors(){

```

```
        TaskActionDescriptorModel[] actionDescArray=new TaskActionDescriptorModel[1];
        TaskActionDescriptorModel actionDesc = new TaskActionDescriptor(NavigationTask.
class, "zoom", "缩放");

        actionDescArray[0] = actionDesc;
        return actionDescArray;
    }

    public TaskParamDescriptorModel[] getParamDescriptors() {

        // TODO Auto-generated method stub

        TaskParamDescriptorModel[] td=new TaskParamDescriptorModel[1];

        TaskParamDescriptorModel paraDesc=new TaskParamDescriptor(NavigationTask.
class, "zoomFactor", "缩放比例");

        td[0]=paraDesc;

        return td;
    }

    public TaskDescriptor getTaskDescriptor() {

        TaskDescriptor td=new TaskDescriptor(NavigationTask.class);

        td.setDisplayName("我的任务");

        return td;
    }

}
```

15.4 小结

本章只是简单地讲述了 ArcGIS Server 的 Java Web ADF 编程方法,不太适合初学者,可供有一定开发经验的开发人员参考。

第 16 章

ArcGIS Server性能优化

ArcGIS Server 作为一个功能强大的 GIS 平台软件，其性能涉及方方面面，如何在现有条件下使 ArcGIS Server 的性能达到最优化呢？本章将介绍有关 ArcGIS Server 的性能调整。

16.1 确定应用系统的瓶颈

网站的性能一直是影响浏览量的重要因素之一。让网站始终保持良好的性能是一个复杂的问题。一个典型的 Web 访问通常受到路由的影响，同时还受到防火墙、Web 服务器和用户浏览的影响，其中任何一个环节都有可能延长访问响应时间。

基于 ArcGIS Server 开发的网站不仅具有一般网站的特征，而且其性能还与 ArcGIS Server 性能、空间数据服务器等相关。下面给出了访问一个 ArcGIS Server 网站的访问顺序，如图 16.1 所示。

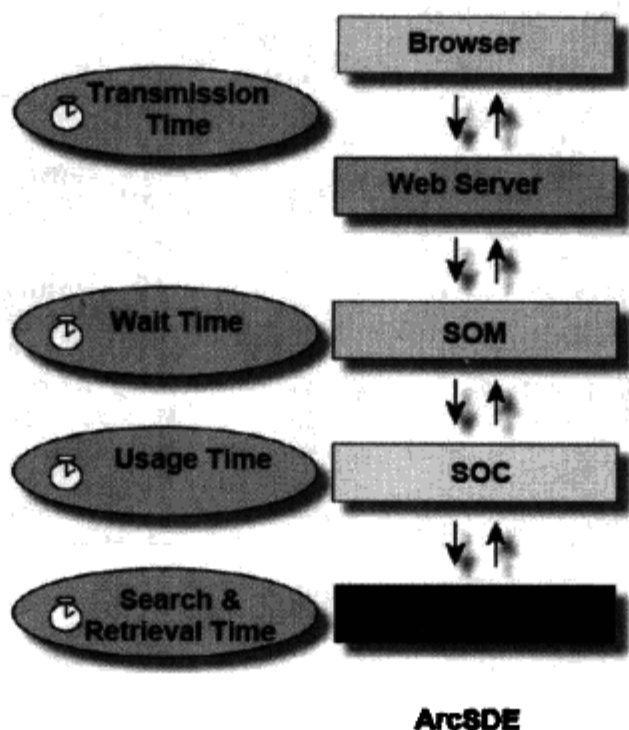


图 16.1 ArcGIS Server 网站访问顺序图

如图中所示，访问 ArcGIS Server 网站需要经过 5 个步骤，每个步骤或步骤之间都会花掉一些时间。应用程序的性能瓶颈有可能存在于每个步骤或某个步骤中。要调整网站的性能，必须先确定网站的瓶颈所在。监测一个网站的性能，有很多工具可供选用。例如，Loadrunner、WinRunner，等等。如果是基于.NET 开发的网站，也可以直接用 Visual Studio 提供的工具直接测试。直接在测试菜单中添加网站进行性能测试，如图 16.2 和图 16.3 所示。

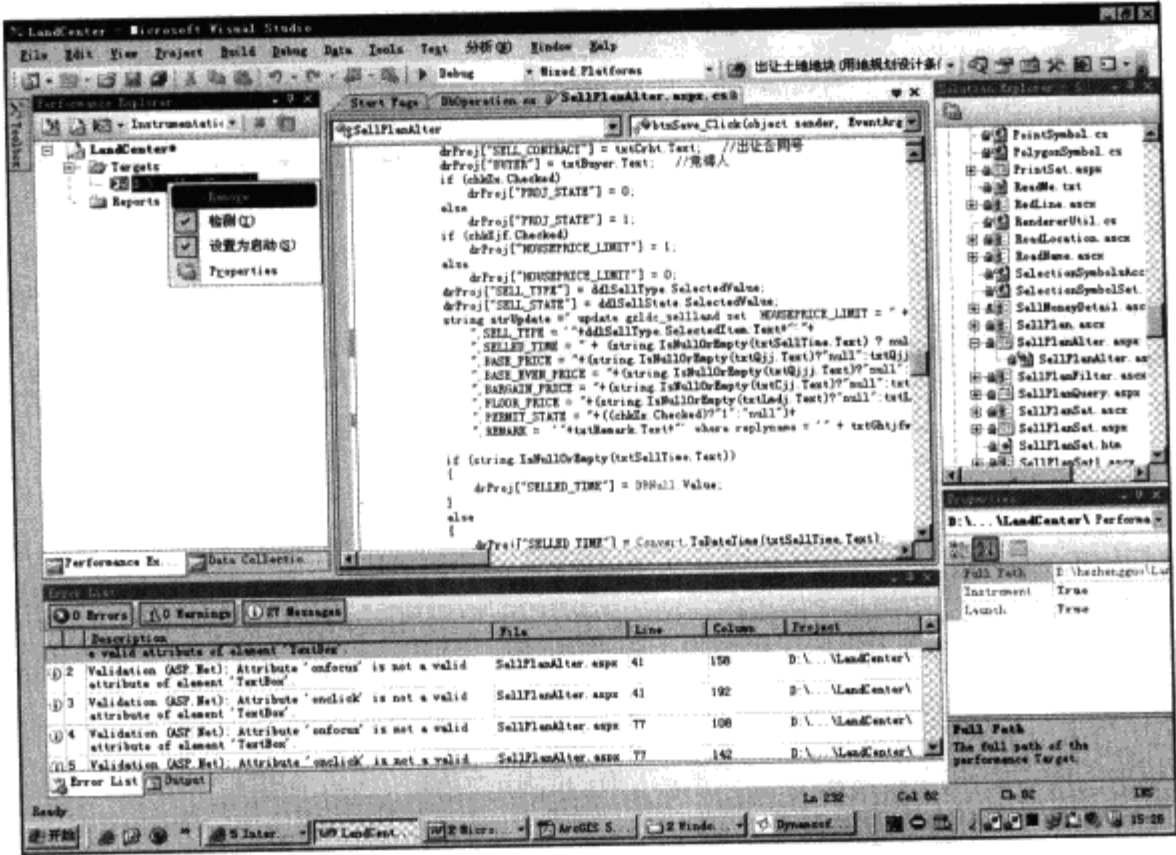


图 16.2 配置性能测试

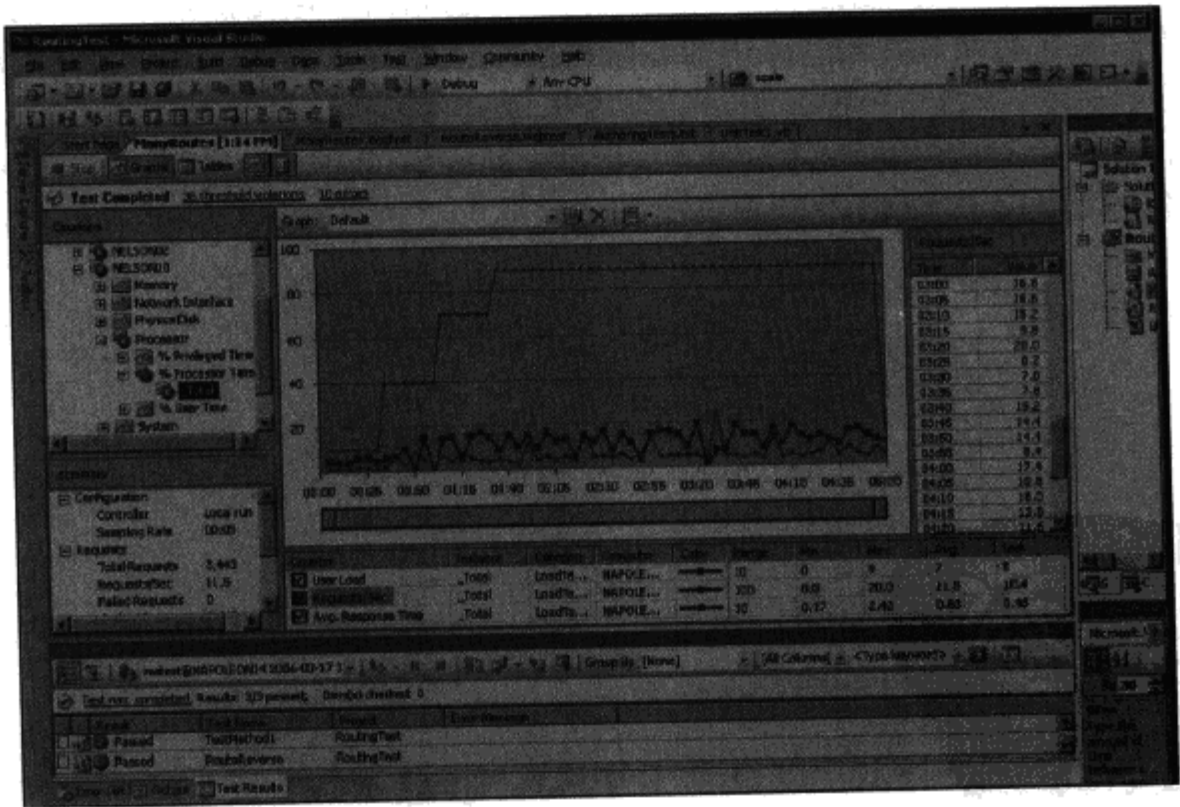


图 16.3 性能测试结果

通过网站性能测试反馈的结果，可以确定影响网站响应速度的几个方面。如果 Web 应用服务器、数据服务器部署在一台机器上，则通用指标如表 16.1 所示。

表 16.1 通用指标

指 标	说 明
ProcessorTime	服务器 CPU 占用率，一般平均达到 70%时，服务就接近饱和
MemoryAvailableMbyte	可用内存数，如果测试时发现内存有变化情况也要注意，如果是内存泄露则比较严重
PhysicsdiskTime	物理磁盘读写时间情况

如果 Web 服务器与数据服务器分开部署，则应该分别检测它们的指标，如表 16.2～表 16.4 所示。

表 16.2 Web 服务器指标

指 标	说 明
Requests PerSecond(Avg Rps)	平均每秒钟响应次数=总请求时间/秒数
Avg time to last byte perterstion(mstes)	平均每秒业务脚本的迭代次数
Successful Rounds	成功的请求
Failed Requests	失败的请求
Successful Hits	成功的点击次数
Failed Hits	失败的点击次数
Hits Per Second	每秒点击次数
Successful Hits Per Second	每秒成功的点击次数
Failed Hits Per Second	每秒失败的点击次数
Attempted Connections	尝试链接数

表 16.3 数据库服务器指标

指 标	说 明
User 0 Connections	用户连接数，也就是数据库的连接数量
Number of deadlocks	数据库死锁
Butter Cache hit	数据库 Cache 的命中情况

表 16.4 稳定网站性能指标

性 能 项	资 源	评 价
CPU 占用率	70%	好
	85%	坏
	90%+	很差
磁盘 I/O	<30%	好
	<40%	坏
	<50%+	很差
网络	<30%带宽	好
运行队列	<2*CPU 数量	好
内存	没有页交换	好
	每个 CPU 每秒 10 个页交换	坏
	更多的页交换	很差

通过表 16.4 的指标，可以测试自己应用的网站各项指标，以确定网站的性能瓶颈所在。由于 ArcGIS Server 处理和连接的是空间数据，其网站性能不仅与硬件和网络相关，还与空间数据

的组织、服务的配置等相关。下面分别进行介绍。

16.2 硬件和网络优化

如果系统的应用经费足够的话，可以购买更好的硬件和网络设备，这将大大地提高系统网站的响应速度。如果在服务器和网络带宽等设备既定的情况下，也可通过一些设置来优化硬件和网络。本节以客户端操作系统 Windows XP 为例进行介绍。

1. 硬件优化

(1) 内存性能优化。

Windows XP 中有几个选项可以优化内存性能，它们位于注册表下面：HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager\Memory Management。

- 禁用内存页面调度 (Paging Executive)。把 DisablePagingExecutive 的值从 0 改为 1。不允许操作系统把数据写入硬盘而是一直让数据保留在内存中，从而提升系统性能。
- 提升系统缓存。把 LargeSystemCache 键值从 0 改为 1，Windows XP 会将 4MB 之外的系统内存全部分配到文件系统缓存中，这意味着 XP 的内核能够在内存中运行，将大大提高系统运行速度。

这两项设置如图 16.4 所示。

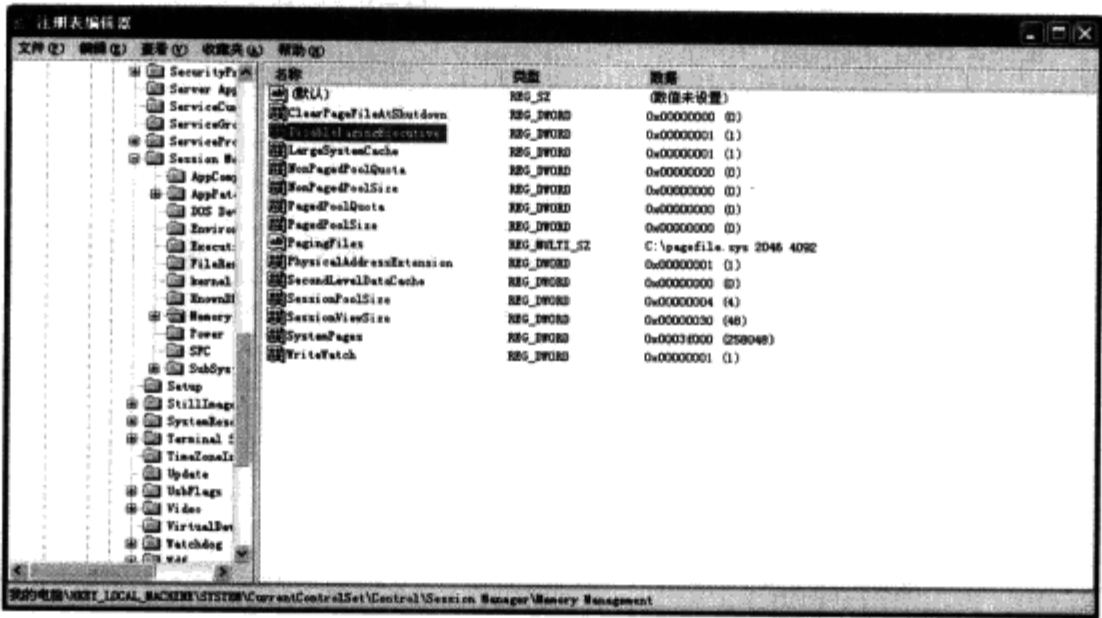


图 16.4 优化内存

(2) 打开 DMA。在“计算机管理”窗口选择“IDE ATA/ATAPI 控制器”选项，进行主要 IDE/次要 IDE 通道的属性设置，将所有的转送模式都设为使用 DMA (如果可用的话)，系统就会自动打开 DMA 资源 (在 BIOS 里也应先设为资源 DMA)，如图 16.5 所示。

(3) 在 Windows XP 中关闭光驱自启动功能。单击“我的电脑”，右键单击 CD-ROM 驱动器，然后单击“属性”，在弹出的“DVD 驱动器属性”对话框中选择“自动播放”标签，对各类文件均选择“不执行操作”选项，如图 16.6 所示。

(4) 设置 CPU。Windows XP 无法自动检测处理器的二级缓存容量，打开注册表：

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager\Memory Management\，选择“SecondLevelDataCache”，根据自己所用的处理器设置即可。例如 Pentium III Coppermine/Pentium IV Willamette 是“256”，Athlon XP 是“384”，PentiumIV Northwood 是

“512”，如图 16.7 所示。



图 16.5 设置 DMA

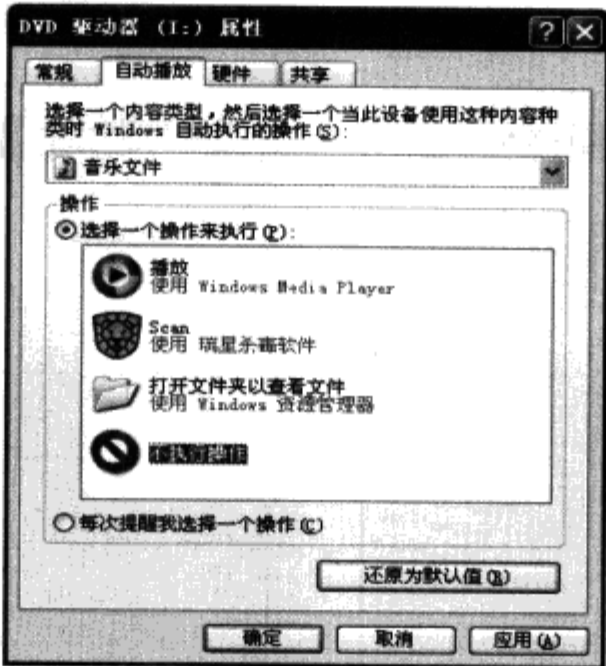


图 16.6 关闭自动播放

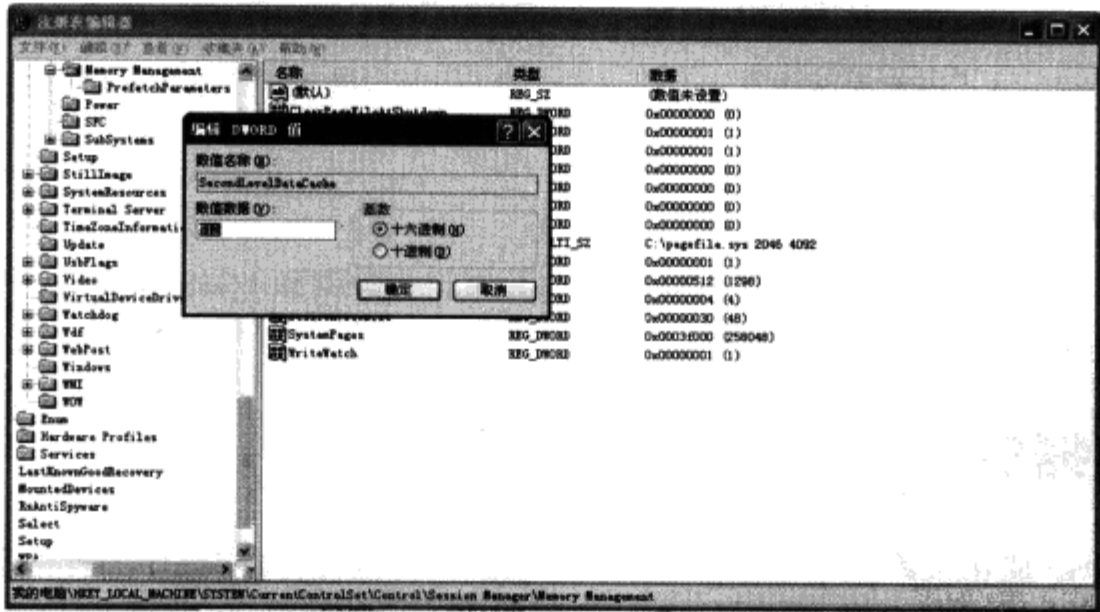


图 16.7 设置二级缓存

2. 网络优化

(1) 加速共享。通常情况下，当 Windows XP 连接到其他计算机时，会检查对方机上所有预定的任务，而且还会等待 30s，这不利于网络的快速访问。可以在注册表中禁止检查对方机器。

打开注册表 HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Current Version\Explorer\RemoteComputer\NameSpace。其中有一个 {D6277990-4C6A-11CF-8D87-00AA0060F5BF} 键，只需将它删除，重新启动计算机后，Windows 不需再检查预定任务，使得速度明显提高。

(2) 解决 Windows XP 执行时停顿的问题。

双击“网上邻居”，右键选择“属性”→“本地连接”，右键选择“属性”，选择“Internet 协议 (TCP/IP)”选项，单击“属性”按钮，在弹出的“Internet 协议 (TCP/IP) 属性”对话框中选中“使用下面的 IP 地址”项，这里选择的 IP 地址为 192.168.0.1，子网掩码为 255.255.255.0，这样以后开机就不会停顿了，如图 16.8 所示。

(3) 在默认情况下，Windows XP 会保留一块网卡 20% 的带宽。方法是，以管理员权限登录，选择“开始”→“运行”菜单，键入 gpedit.msc，出现“组策略”窗口，展开“管理模板”中的“网络”项，展开“QoS 数据包计划程序”，在右边窗格中右键单击“限制可保留带宽”项，在“属性”中的“设置”标签中有“限制可保留带宽”项，选择“已禁用”选项，单击“确定”按钮即可，如图 16.9 所示。

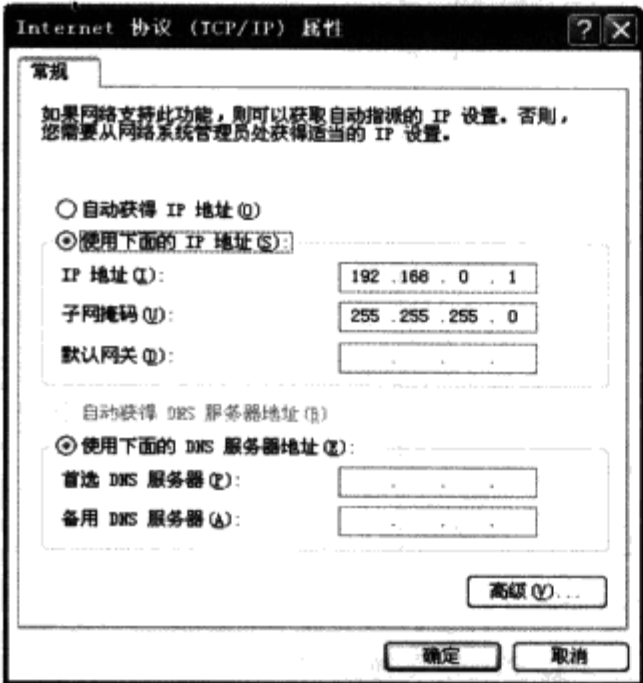


图 16.8 设置本地连接

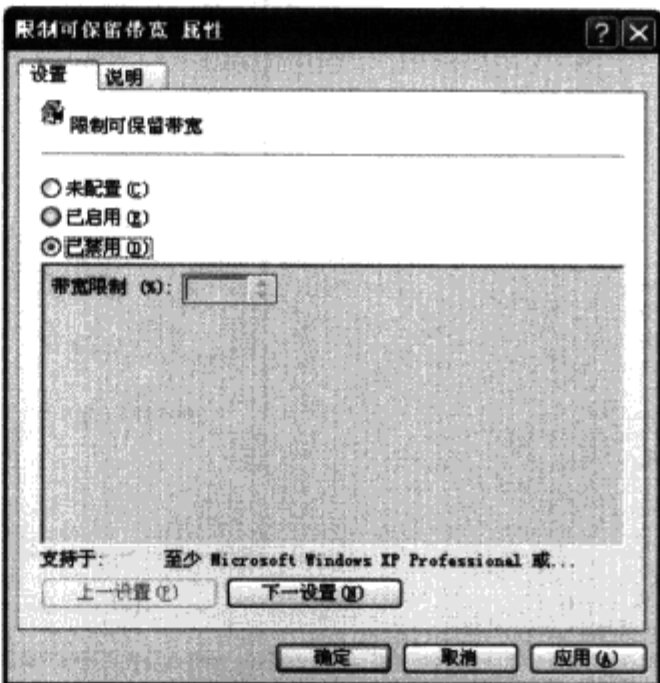


图 16.9 禁用保留带宽

16.3 数据

ArcGIS Server 最终的目的是把空间数据展示给用户，所以 ArcGIS Server 应用程序的效率与空间数据息息相关。

16.3.1 数据量

ArcGIS Server 应用大多是基于 B/S 架构的，基于浏览器的应用程序的性能瓶颈大多在网络传输，传输的数据量越大，需要的网络带宽也越大，反之亦然。在网络带宽一定的情况下，需要传输的数据量越小速度越快。

关于减少网络传输的数据量，用户肯定会想到我所发布的数据都是必须的，怎么可能减少数据量呢？其实则不然。例如，基础地形图，一般都有多个比例尺：1:500、1:2000、1:10000 等，1:500 的基础地形图当然比 1:2000 的地形图数据量要大得多，可以在发布地图服务时设置不同图

层的显示比例，当用户访问的比例尺是 1:500 时就显示 1:500 的基础地形图，当用户访问的比例尺为 1:2000 时，不再显示 1:500 的地形图而是显示 1:2000 的地形图，依次类推，这样就大大减少了网络传输的数据量，提高了应用程序的性能。如果没有多比例尺的空间数据也没有关系，ArcGIS Server 提供很多地图综合工具，我们可以采用大比例尺综合成小比例尺的工具，如图 16.10 所示。

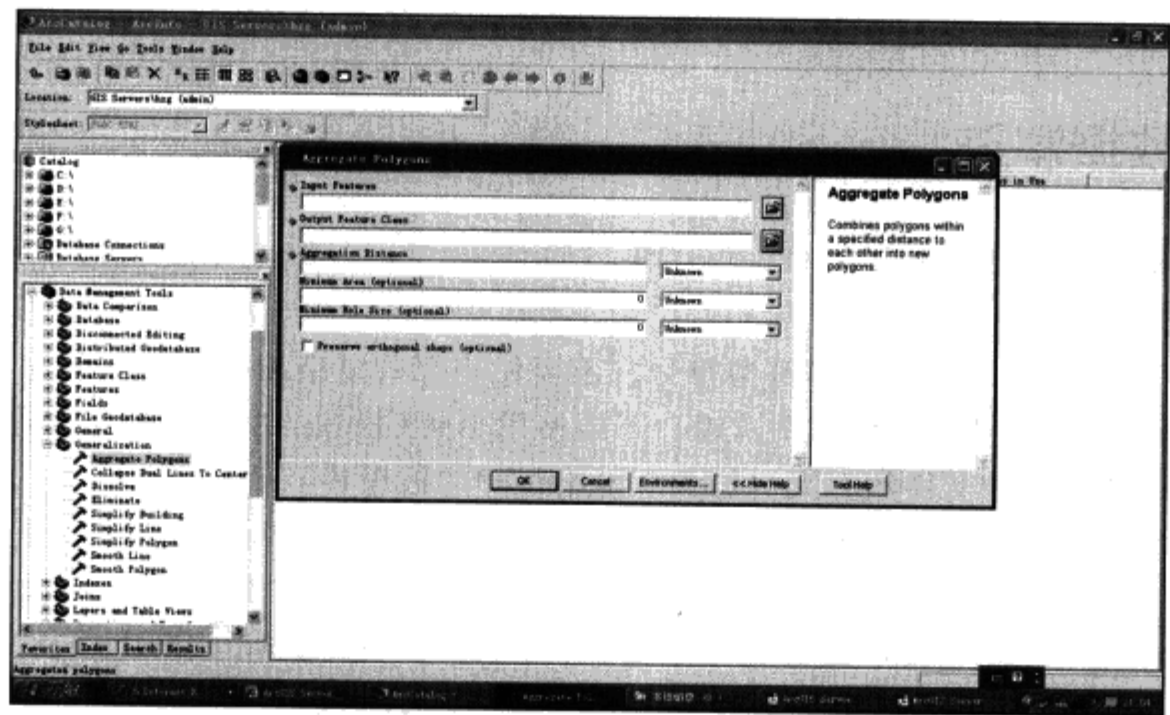


图 16.10 地图综合界面

除了地图综合可以减少空间数据量外，还有另外一种方式可以减少网络传输的数据量。空间数据往往都带有很多属性，有些属性对于特定的用户而言是没有任何意义的，因而以字段的不可见减少数据的传输量，如图 16.11 所示。

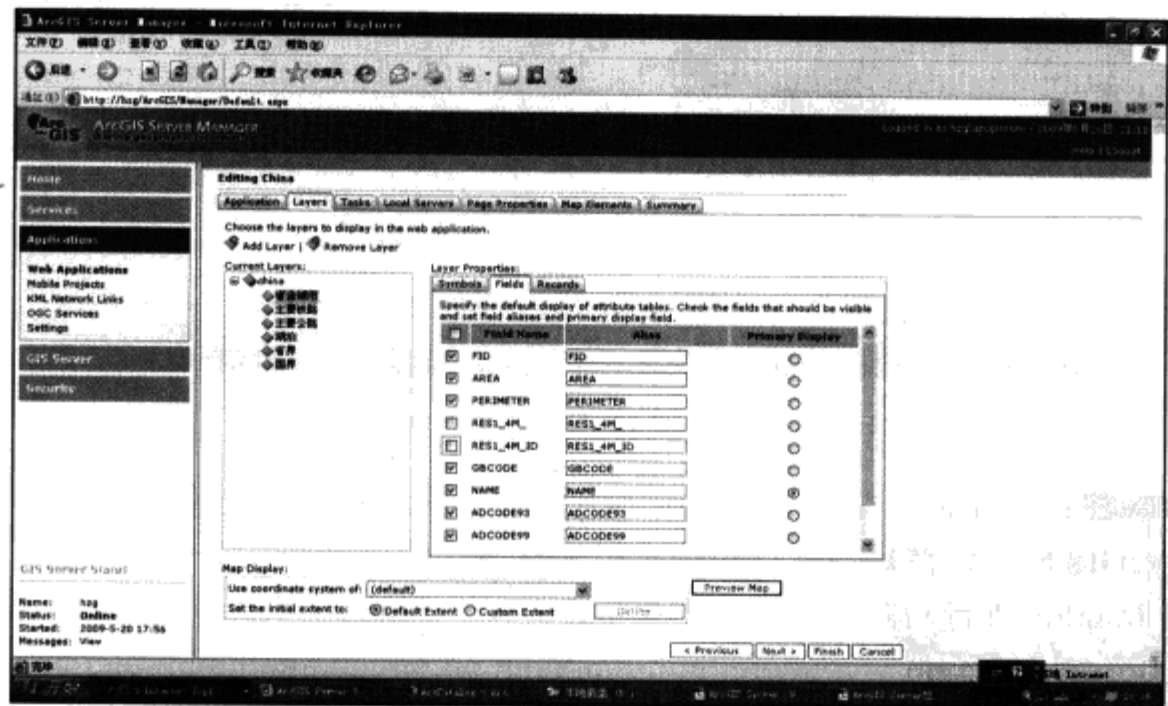


图 16.11 设置字段可见与否

16.3.2 数据组织

空间数据的组织主要是基于 ArcSDE 来访问。提高空间数据访问效率有以下方式：

- 对经常要进行搜索的属性字段建立索引。
- 空间索引大小的调整会影响空间数据访问的效率。空间数据大小的调整要依照空间数据每个单元的大小而定。一般建立两级索引就可满足要求,第二级索引是第一级的 4~5 倍,第一级索引是要素层中大多空间要素的大小。
- 尽量减少叠加图层和服务的个数。如果发布的空间数据服务越多, ArcGIS Server 在处理数据叠加时所需要的时间就越长,系统的效率也就越低。

16.3.3 数据渲染

空间数据的渲染无非就是设定点、线、面、注记的符号。为了提高动态图层的访问效率,需要遵循以下原则。

- **点要素:** 尽量使用简单或字符型符号,不要使用那些复杂的符号,如晕轮、掩模或形状复杂的符号。在点符号使用图片时,不要使用 BMP 格式的文件,尽量使用 EMF 格式,因为 BMP 格式的文件数据大,加大了网络传输的数据量。
- **线和多变性要素:** 使用 ESRI_Optimized 符号,外廓线用简单的线,不要使用制图线。填充时不要使用 BMP 格式文件,而是使用 EMF 格式。使用 ESRI_Optimized 的符号,是因为这些符号由 ESRI 公司进行优化,显示效率大大提高了,如图 16.12 所示。

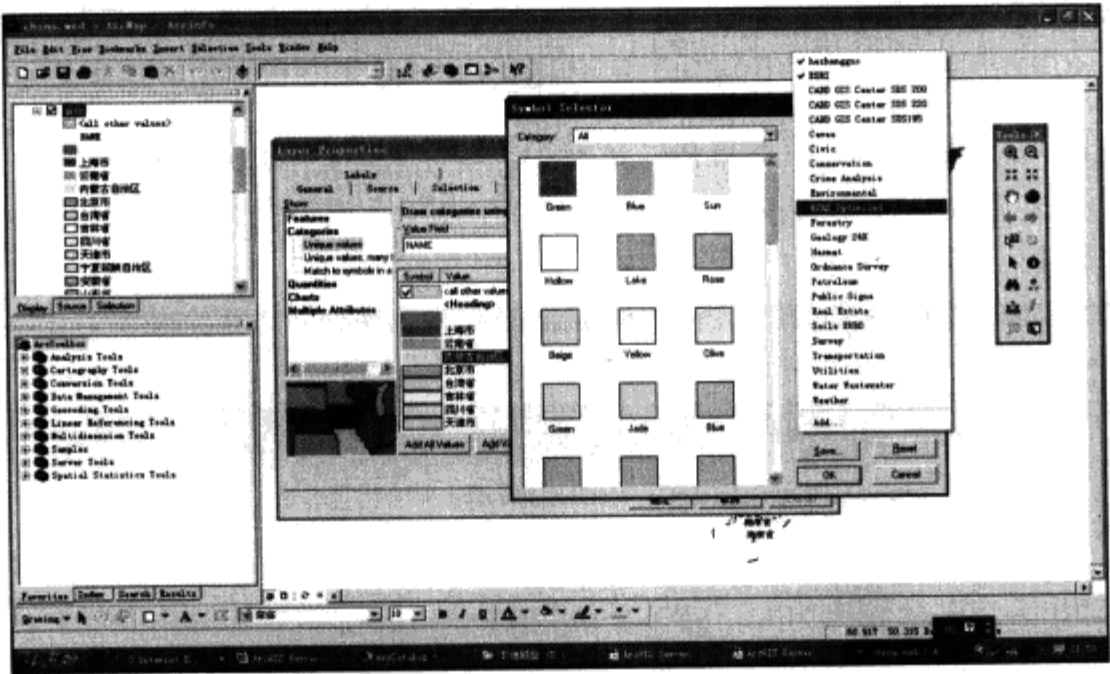


图 16.12 设置 ESRI_Optiomized 符号

- **文本和标注:** 用注记代替标注 (Labe), 这样也许会加大网络传输的数据量, 然而 Label 需要 ArcGIS Server 在每一次访问的时候计算文本的位置和样式——消耗的资源更多; 避免使用 Maplex 进行动态标注, 理由与 Label 一样; 避免使用特殊效果, 如填充模式、晕轮、callouts、背景等。

16.4 服务

除了有很多工具可以监测网站的性能外, ArcGIS Server 也提供工具来测试 ArcGIS Server 提供空间数据服务的能力。

如 ArcGIS Server 的统计参数, 如图 16.13 所示。它提供 3 个参数的统计: 服务创建时间、

服务等待时间、服务使用时间。如果服务创建时间过长,可以考虑事先把服务对象装载到内存中,进行池化处理;如果服务等待时间过长或有超时的情况,可以查看最大等待时间设置或增加服务对象个数以减少等待时间;如果服务使用时间太长,需要检查应用程序代码是否正确,或者客户端操作过于复杂。

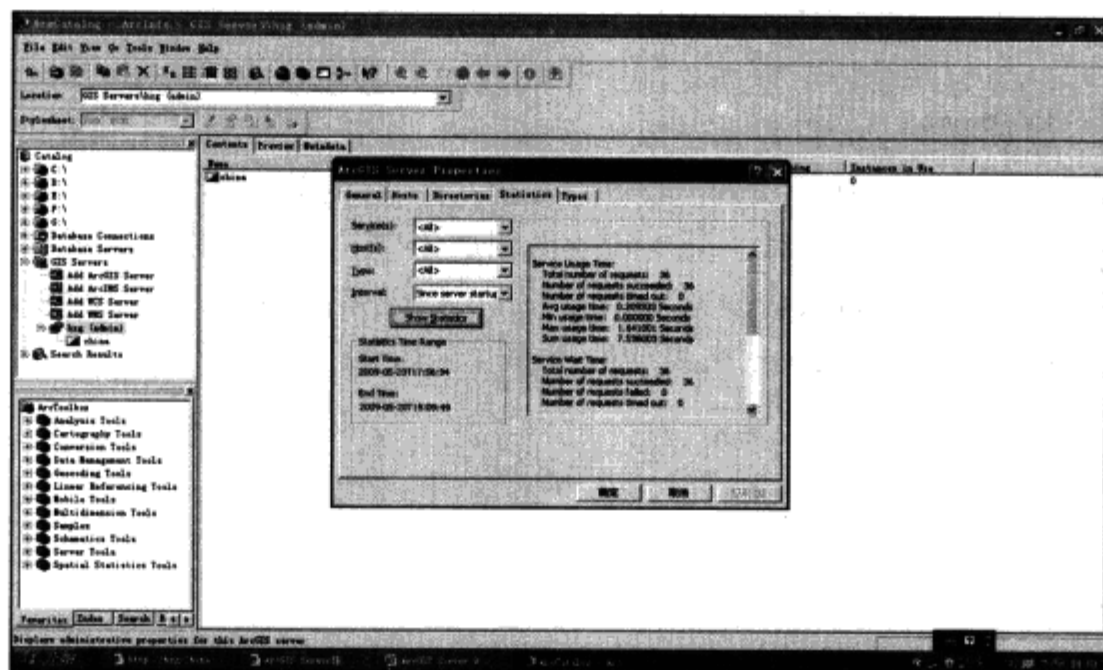


图 16.13 ArcGIS Server 统计参数

除了使用 ArcGIS Server 的统计参数, ArcGIS Server 的日志也给我们提供了 ArcGIS Server 性能方面的警告或错误,如图 16.14 所示。

通过查看 ArcGIS Server 日志,可以看到 ArcGIS Server 性能方面的问题,根据日志的提示进行相应的处理,也可以提高系统的性能。

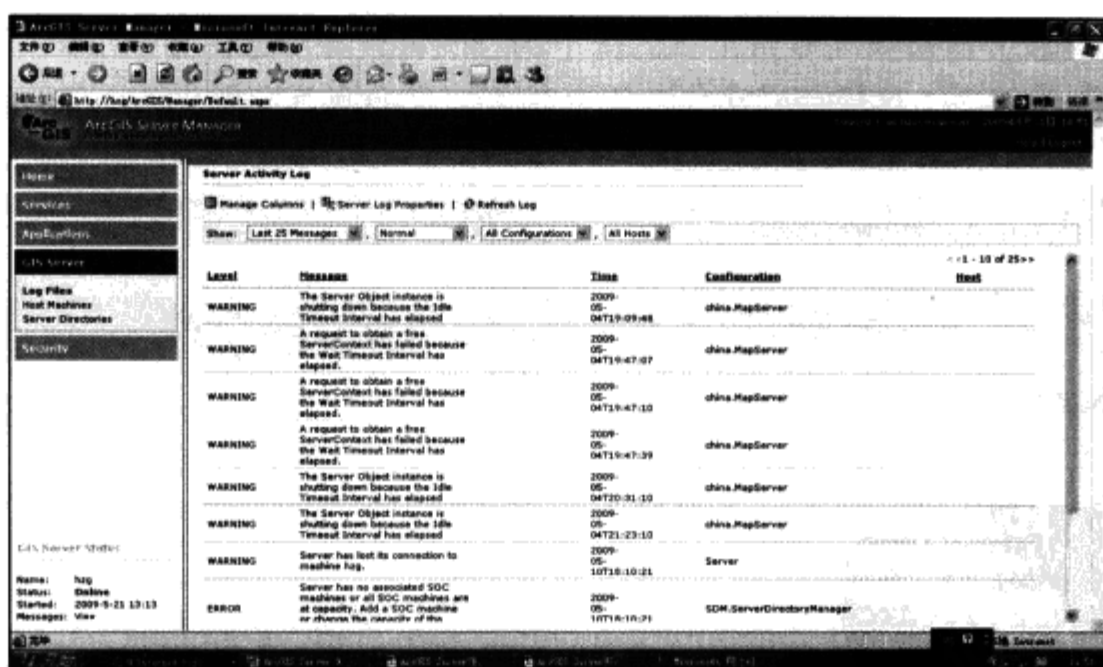


图 16.14 ArcGIS Server 日志

16.4.1 服务组织

ArcGIS Server 的地图发布可以采用 Cache 技术,该技术大大提升了 ArcGIS Server 访问空间数据的能力。其原理是将图层切片保存在服务器上,客户端请求时直接访问 Cache 好的图片,这

样 ArcGIS Server 不用适时地生成图片，其性能自然得到大幅提升。

说到 ArcGIS Server 的 Cache，就必须谈谈 ArcGIS Server 的 Cache 组织问题。我们在发布地图服务的时候，经常面对的情况是有些基础数据几乎不变或者很长时间才变化一次，而有些与业务相关的空间数据经常发生改变，或者说一些空间数据是定期更新的而另外一部分空间数据则是不定期更新的。针对这种情况，处理方式如下：

（1）对于那些经常更新或者不定期更新的空间数据采用 ArcGIS Server 直接访问空间数据，这类的空间数据一般数据量不大，数据层数也不多。

（2）对于那些长时间才更新一次的空间数据或定期更新的数据则采用创建 Cache 来缓存地图，在创建 Cache 的时候，并非为每一个空间数据图层创建一个缓存服务，而是根据空间数据的相关性分名别类，为相关的空间数据创建一个服务。这样做是因为服务太多创建服务需要的时间更长，维护也更麻烦，ArcGIS Server 在叠加地图的时候所占用的资源更大。

16.4.2 服务设置

ArcGIS Server 的服务模式有池化和非池化两种。如果是非池化的服务，服务中的数据可以是注册了版本或者没有注册版本的，如果是注册了版本的数据，所有的编辑操作能进行 redo、undo 操作，并且有冲突解决的方案。如果是没有注册版本的数据，就不能进行 redo、undo 操作，但是可以对编辑内容选择一次保存和不保存。如果是池化的服务，那么服务中的数据就只能是而非版本的。这种情况下的编辑功能是最简单的，编辑的内容一旦完成就进行保存，没有办法进行 redo、undo 操作。比如对一个多边形进行节点移动操作，一旦操作完成就不能恢复。

池化服务还有另外一个特点，它与线程池和数据连接池概念有些类似，即客户端请求使用服务器对象，使用完毕后并不释放服务器对象，而是把这些对象放到池中，当下一个客户到来时，服务器端不用创建服务器对象而是直接取出池中的空闲服务器对象供客户端调用。这些服务器对象对于用户来说是共享的，这样与非池化的服务相比池化的服务性能更高。有关池化的设置如图 16.15 所示。

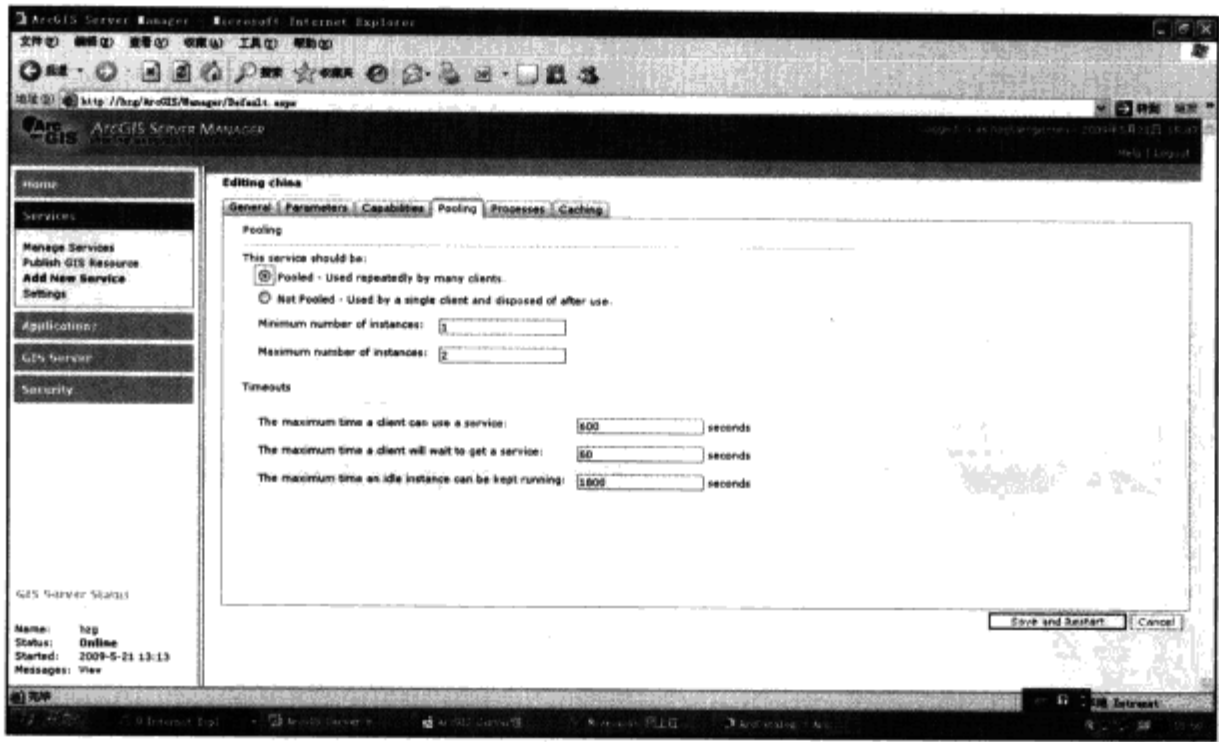


图 16.15 设置服务模式

16.5 应用系统配置

为了提高 ArcGIS Server 应用程序的性能,而且基于 ArcGIS Server 多是基于 B/S 架构的程序,用户访问量非常大,这样有必要实现应用系统的负载均衡。

目前采用的网络负载均衡转发技术主要有 3 种:网络地址转换、直接路由和 IP 隧道技术。采用不同的方法将客户端发送的包转发到目的服务器上,并确保目的服务器的返回包可以顺利到达客户端。通过 ArcGIS Server 的体系结构,可以发现对基于 ArcGIS Server 的企业级应用是采用负载均衡技术,主要表现在 Web 服务器、ArcGIS Server (SOM+SOC)、ArcSDE (主要从 Oracle 的角度考虑)。

(1) Web 服务器的负载主要考虑通过对 Web 服务器的网络载荷进行监听,配置多余的服务器环境。在正常运行时,通过负载均衡软件对用户的请求进行分配,保持网络载荷的平衡;在有服务器发生故障时,只向正常工作的服务器发送信息。目前很多 Web 服务软件都有这项技术,如 IIS 中有 ISAPI 过滤器;WebLogic 也可以很好地实现负载均衡。

(2) ArcGIS Server 的负载均衡由一个镜像的 AGS 的 SOM 配置提供。一个 SOM 可以管理多个 SOC,并能自动地把客户端的请求分配到空闲的 SOC 机器上。在 SOM 中添加 SOC,如图 16.16 所示。

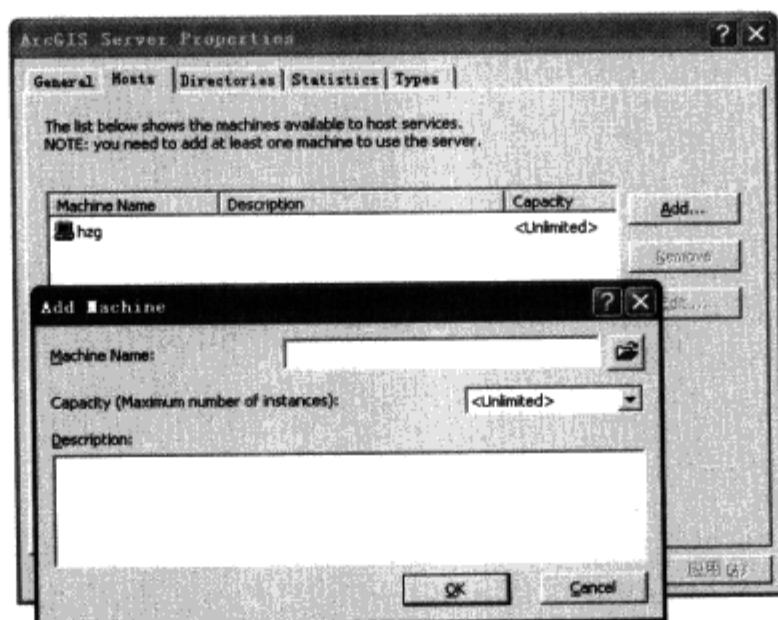


图 16.16 添加 SOC 机器

(3) ArcSDE 的负载策略集中于数据库的负载策略。对于 ArcSDE For Oracle 来说,解决 Oracle 的负载均衡和集群技术比较关键。

集群系统通过连接一台或多台计算机,使得它们对客户端好像一台机器。数据存放在一个共享的磁盘中。使用集群的主要目的就是提供系统的容错功能(其中一台机器出现问题并不影响使用,但应用程序的内存不被传递),通过添加或减少节点进行水平伸缩。集群主要是从系统的容错能力以及可扩展性角度描述了系统的设计策略。同时系统良好的可扩展性也会增加系统处理网络负载的能力。

16.6 小结

在发布应用网站的时候,经常发现开始网站的响应速度并不慢,随着用户数量的增多,系统会变得越来越慢。本章主要讲述了基于 ArcGIS Server 的网站优化,以及服务端的数据、服务、应用系统配置等。通过修改操作系统的一些设置可提高客户端的硬件和网络性能,从而达到提高访问 ArcGIS Server 应用程序的力能。

第五篇

ArcGIS Server

综合案例

► 第 17 章 某市土地开发中心地理信息管理平台

本部分主要讲述基于 ArcGIS Server .NET Web ADF 的开发实践, 通过一个具体案例来说明在开发过程如何使用 ArcGIS Server .NET Web ADF, 这能为读者在实际开发项目中提供一定的参考意义。

第 17 章

某市土地开发中心地理信息管理平台

本章主要讲述基于 ArcGIS Server .NET Web ADF 的土地开发中心的业务办公平台，该平台不仅要处理大量的地理空间信息，还要处理日常办公用文件的传送，是一个综合 WebGIS 与办公自动化的业务办公系统。

17.1 需求概述

一个城市的地图储备管理部门一般是土地储备（开发）中心，这个部门本身没有多少空间数据，要建立土地储备管理信息系统，需要规划国土部门提供大量的数据支持。下面简单介绍一下土地储备中心的业务流程：土地储备中心储备土地需要到规划部门进行红线选址，确定选址后，需要征地的地块由征地部门进行征地，同时资金计划部门制定该地块的投资计划，该地块征地完成后，由工程部门进行土地整理，并办理该地块规划设计条件、用地预审和环评等相关手续，最后由土地出让部门进行土地拍卖，这样就完成了土地储备的完整流程。

17.1.1 明确需求

土地开发中心下辖有 5 个部门：前期研发部、征地部、业务指导部、计划财务部、土地出让部。整个地图开发中心的业务流程大致如图 17.1 所示。

具体各部门的需求如下。

本系统的用户主要划分为 5 类：中心领导、前期研发部、征地部、业务指导部、出让部和系统管理员。下面我们从各类用户的角度来分析他们对系统的需求，其中中心领导的需求可参见其他 4 类用户。

1. 前期研发部

前期研发部主要负责经营性土地红线储备，具体需求如表 17.1 所示。

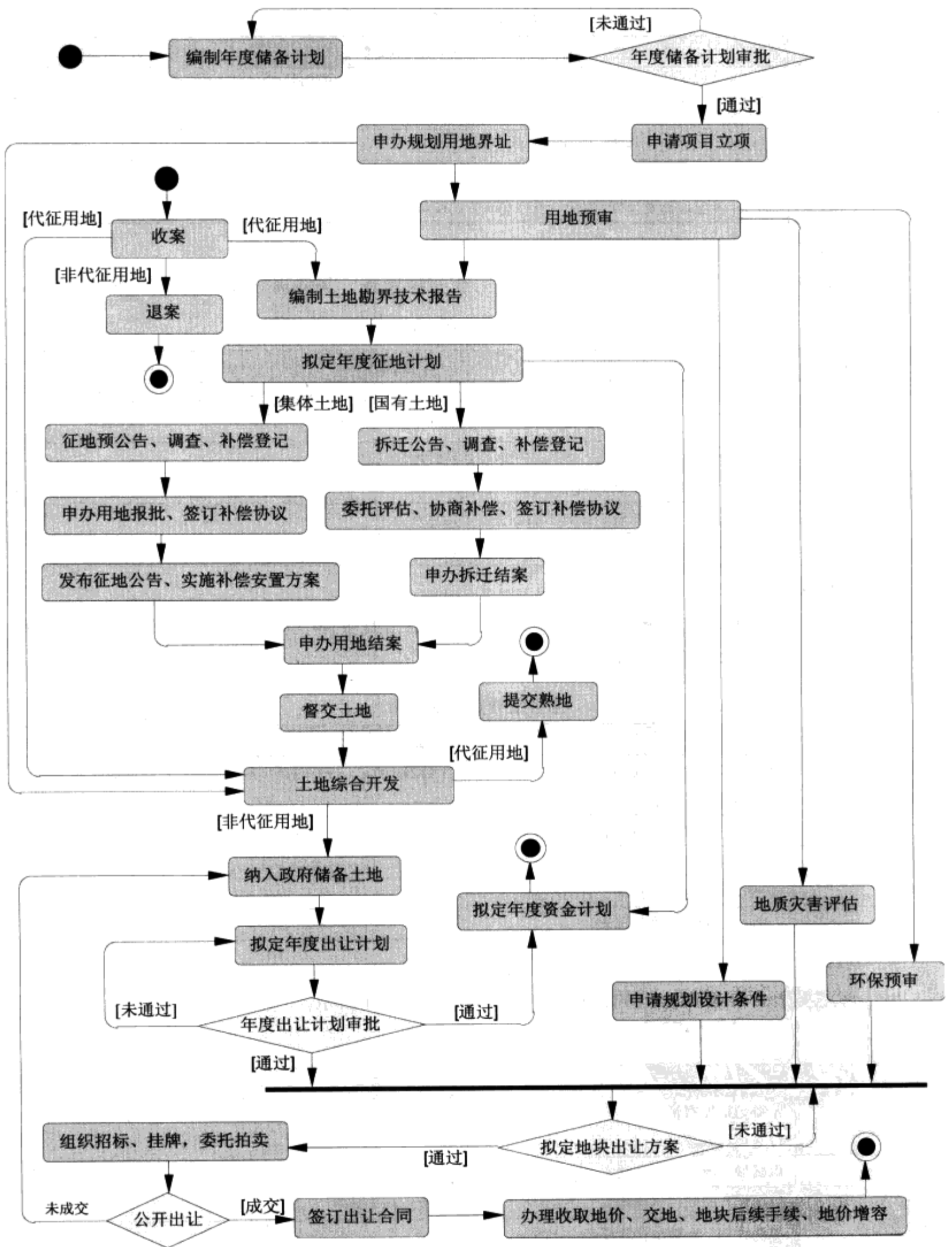


图 17.1 地图开发中心业务流程图

表 17.1 前期研发部功能需求

功能名称	功能描述
查询	基础地形图查询、土地利用规划查询、城市总体规划查询、用地红线查询、储备地块查询
统计	以用地性质、地块性质、区域、时间等统计地块信息
专题图	以用地性质、地块性质、区域、时间等制作专题图
编辑	新建红线储备地块（基本信息、储备规划、年度计划）
输出	输出年度土地储备计划（红线）、红线储备进度状况表

2. 征地部

征地部主要负责经营性土地征地和拆迁，完成实物储备，具体需求如表 17.2 所示。

表 17.2 征地部功能需求

功能名称	功能描述
查询	基础地形图查询、土地利用规划查询、城市总体规划查询、用地红线查询、储备地块查询、红线储备进度查询
统计	以区域、时间、进度状态、地块或用地性质统计地块信息
专题图	以区域、时间、进度状态制作专题图
编辑	实物储备计划、年度资金计划、储备进度等信息录入
输出	输出年度实物储备计划，实物储备进度状况表

3. 业务指导部

业务指导部主要负责经营性土地的综合开发，包括市政道路、公建配套、大型绿化等工程，将生地变为熟地，具体需求如表 17.3 所示。

表 17.3 业务指导部功能需求

功能名称	功能描述
查询	基础地形图查询、土地利用规划查询、城市总体规划查询、控制性详细规划查询、地块征地或拆迁情况和进度查询、资金计划查询等
统计	以区域、时间、进度状态等统计土地综合开发信息
专题图	以区域、时间、进度状态制作专题图
编辑	工程设计、报建、招标、合同等信息录入
输出	土地综合开发进度状况表

4. 出让部

出让部主要负责经营性土地的出让，具体需求如表 17.4 所示。

表 17.4 出让部功能需求

功能名称	功能描述
查询	基础地形图查询、土地利用规划查询、城市总体规划查询、用地红线查询、储备地块查询、实物储备进度查询
统计	以区域、时间、进度状态、地块或用地性质统计地块出让信息
专题图	以区域、时间、出让状态、是否限价房等属性要素制作专题图
编辑	出让计划、土地出让状况、出让明细信息
输出	输出年度出让计划、出让进度状况表、出让明细统计表

5. 系统管理员

系统管理员主要负责用户权限的管理和数据的管理，具体需求如表 17.5 所示。

表 17.5 系统管理员功能需求

功能名称	功能描述
用户管理	增删改用户
角色管理	增删改角色
权限管理	增删改权限
角色配置	为用户配置角色
权限配置	为角色配置权限
数据迁移建库	将已有数据迁移到本系统的数据库
数据导入导出	由其他数据源定期导入更新数据，或定期导出本系统数据

17.1.2 设计实现框架

由于土地储备单位的业务性质，与规划国土部门有很多的业务往来，这样就决定了土地储备系统不仅要访问规划国土部门提供的空间数据服务，还需要与他们的业务办公系统进行通信，使业务数据能够无缝地在系统之间进行传递。为了满足系统的这些要求，土地储备系统采用 B/S 架构，与规划国土部门的业务系统交互采用标准 Web Service 接口。对于国土规划空间数据的共享，为了保护数据提供单位的数据投资和数据使用的安全性，数据提供单位提供标准的空间数据服务而不是采用文件共享或数据直接访问的方式，这样系统就必须采用分布式架构，如图 17.2 所示。

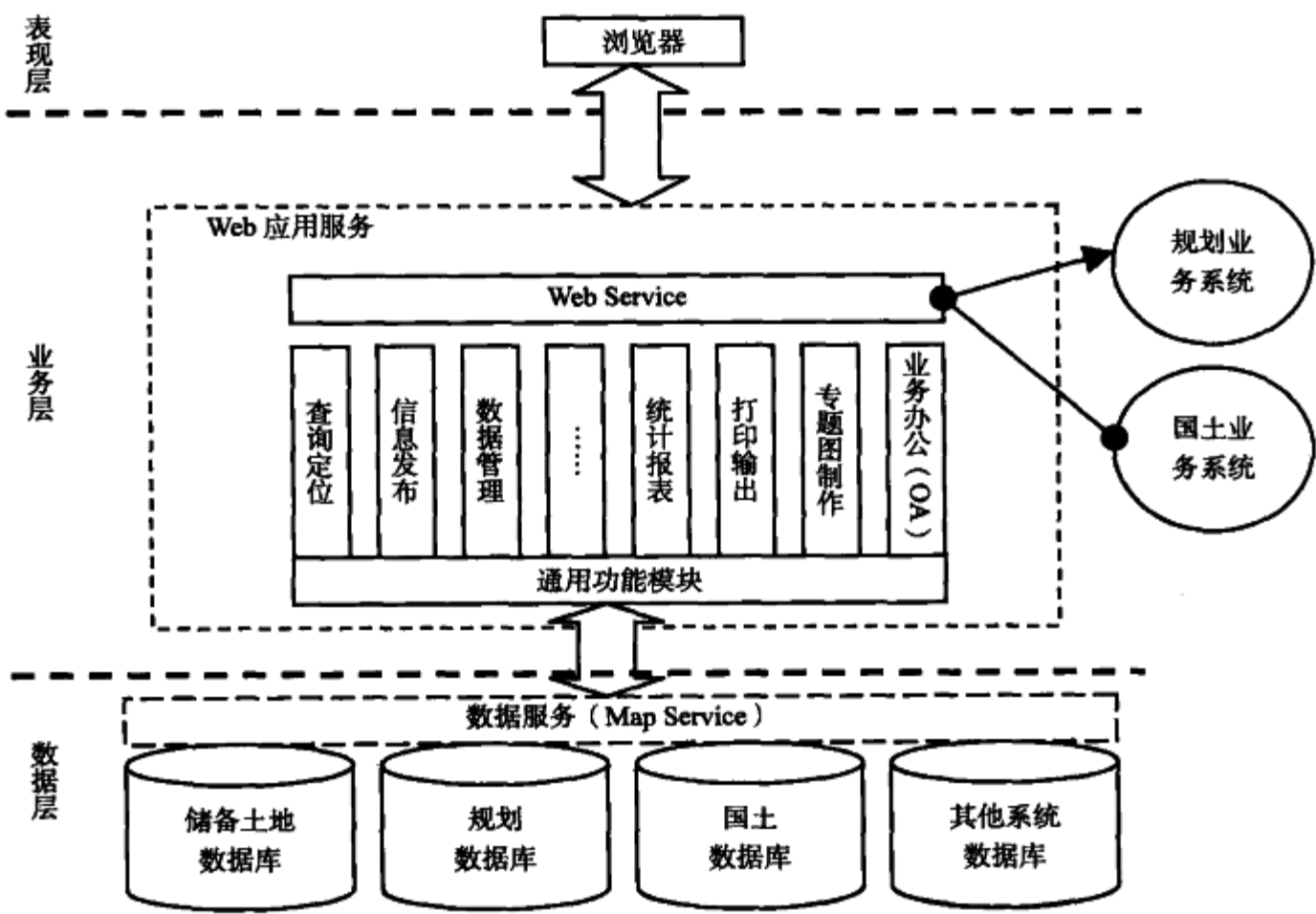


图 17.2 系统体系结构

17.2 数据库设计

土地开发数据库建设是本系统实施的一项核心工作内容。土地开发数据库既能给土地开发业务办理提供基础依据，同时也应随着土地开发业务办理过程的进行实现数据库的动态维护。

土地开发数据库总由 5 个部分组成，包括：基础地理空间数据库、城市规划空间数据库、国土空间数据库、土地储备开发地理信息数据库、土地储备开发业务信息数据库。前三者是基础数据，可以通过共享应用市规划局和市国土局已有的信息资源，大大缩短本系统的建设周期和系统的建设成本。后两者则作为本项目的核心工作内容，随土地开发业务办理流程建立完成。土地开发数据库组成结构如图 17.3 所示。

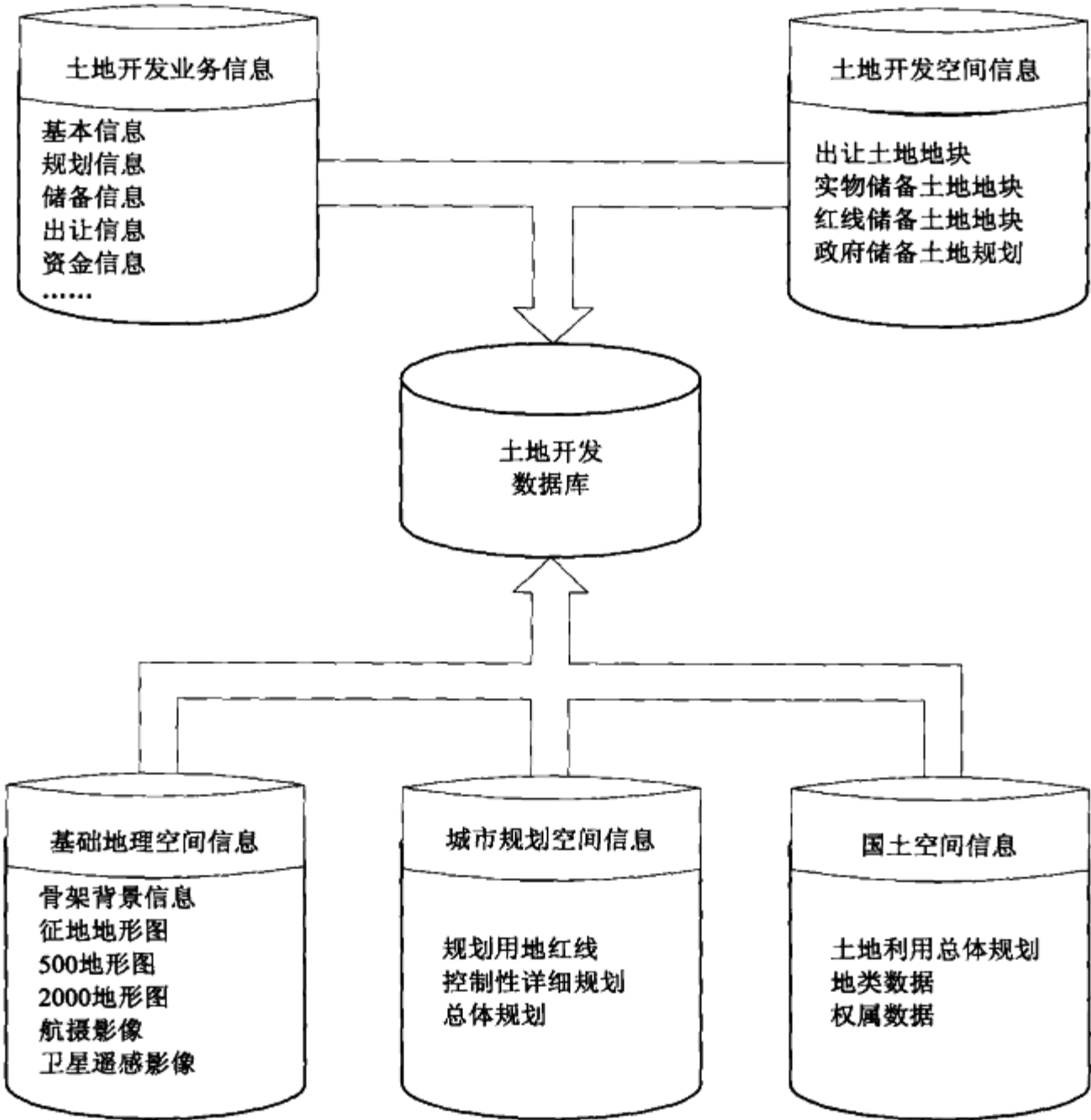


图 17.3 土地开发数据库组成结构图

土地储备开发数据是本系统的核心业务数据，包括土地储备开发地块的地理信息（GIS 数据）和业务信息。业务信息包括储备地块的规划信息（用地面积、建筑面积、用地性质等）、储备地块的储备计划和进度、储备地块的出让计划、实施进度和出让金给付情况、储备地块的资金计划和资金使用情况等。储备地块的 GIS 数据来自市规划局的用地数据和市测绘所的土地勘测定界，属性数据来自土地开发中心的业务工作，主要来自前期研发部、征地部和出让部。

针对土地开发中心的业务部门和业务流程,设计以下 3 个空间数据层来对应年度的土地储备计划、征地计划和出让计划,分别为。

(1) 红线储备土地地块。

取得市规划局用地界址批复和用地预审意见的地块,图形数据由土地开发中心向市规划局申请用地界址批复后系统自动录入。

(2) 实物储备土地地块。

已完成征地或拆迁结案和用地结案手续的权属界土地,图形数据来源于市测绘所编制的《土地勘测定界技术报告书》附图电子文件,入库流程为制定土地勘测定界报告附图电子文件标准→市测绘所提交标准土地勘测定界报告附图电子文件→系统入库。

(3) 出让土地地块。

已完成实物储备并取得规划设计条件的地块,图形数据由土地开发中心向市规划局申请规划设计条件批复后系统自动录入。

土地储备开发空间地理数据构成和入库流程如图 17.4 和表 17.6 所示。

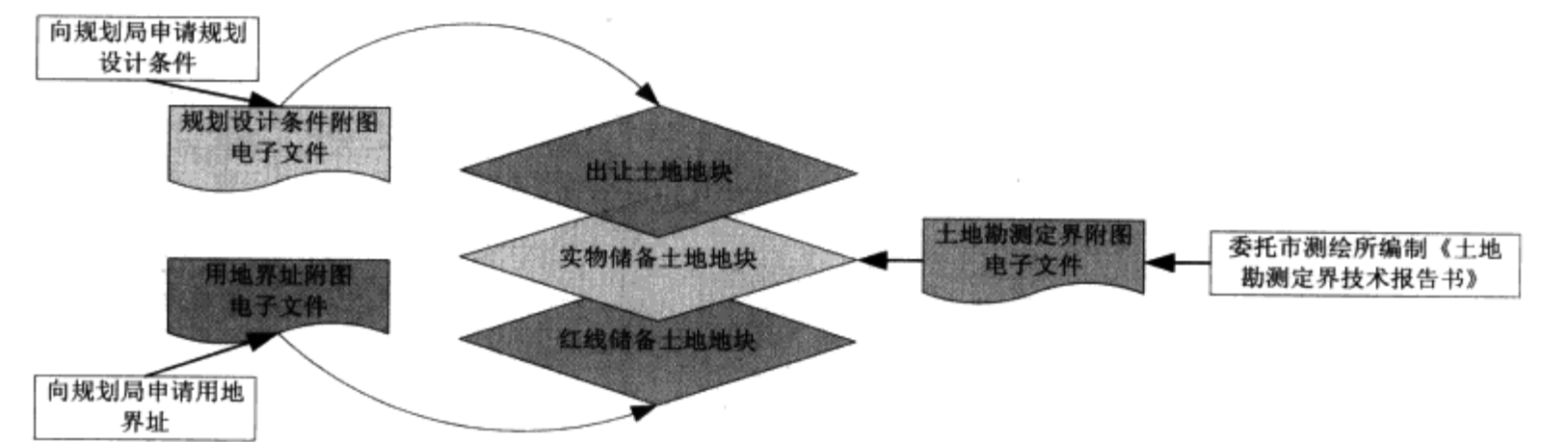


图 17.4 土地储备开发空间地理数据入库流程

表 17.6 土地储备开发 GIS 数据

数据类别	数据名称	要素名称	数据源	说明
土地储备中心 GIS 数据	出让土地	出让土地地块	向市规划局申请用地界址附图	初始图形数据由市规划局用地红线批复录入
	实物储备土地	实物储备土地地块	向市测绘所申请土地勘测定界报告附图	制定土地勘测定界报告附图电子文件标准→市测绘所提交标准土地勘测定界报告附图电子文件→系统入库
	红线储备土地	红线储备土地地块	向市规划局申请规划设计条件附图	初始图形数据由市规划局用地红线批复录入
	政府储备土地 规划	烂尾地块	政府储备土地规划编制成果	数据录入、更新、维护由市规划局负责
		可储备地块图则		
		储备意向		
		可储备单元		

土地开发业务数据库内容包括：土地开发业务案件基本信息、规划信息、储备信息、征地信息、出让信息、资金信息等。

17.3 业务逻辑设计

土地开发中心的业务逻辑设计，设计各部门的实际用例。

1. 前期研发部

前期研发部用例设计如图 17.5 所示。

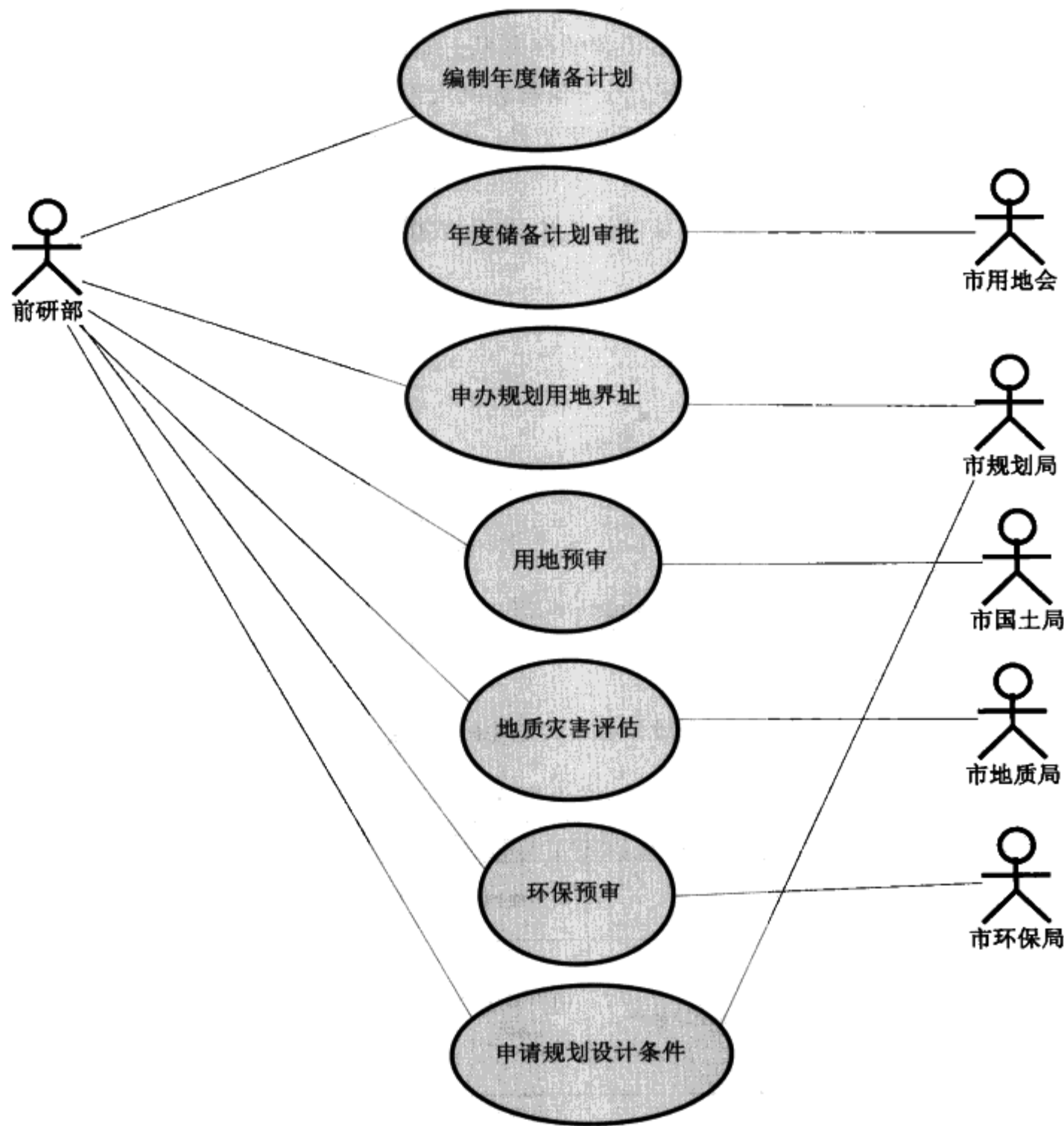


图 17.5 前期研发部用例设计

2. 征地部

征地部用例设计如图 17.6 所示。

3. 业务指导部

业务指导部用例设计如图 17.7 所示。

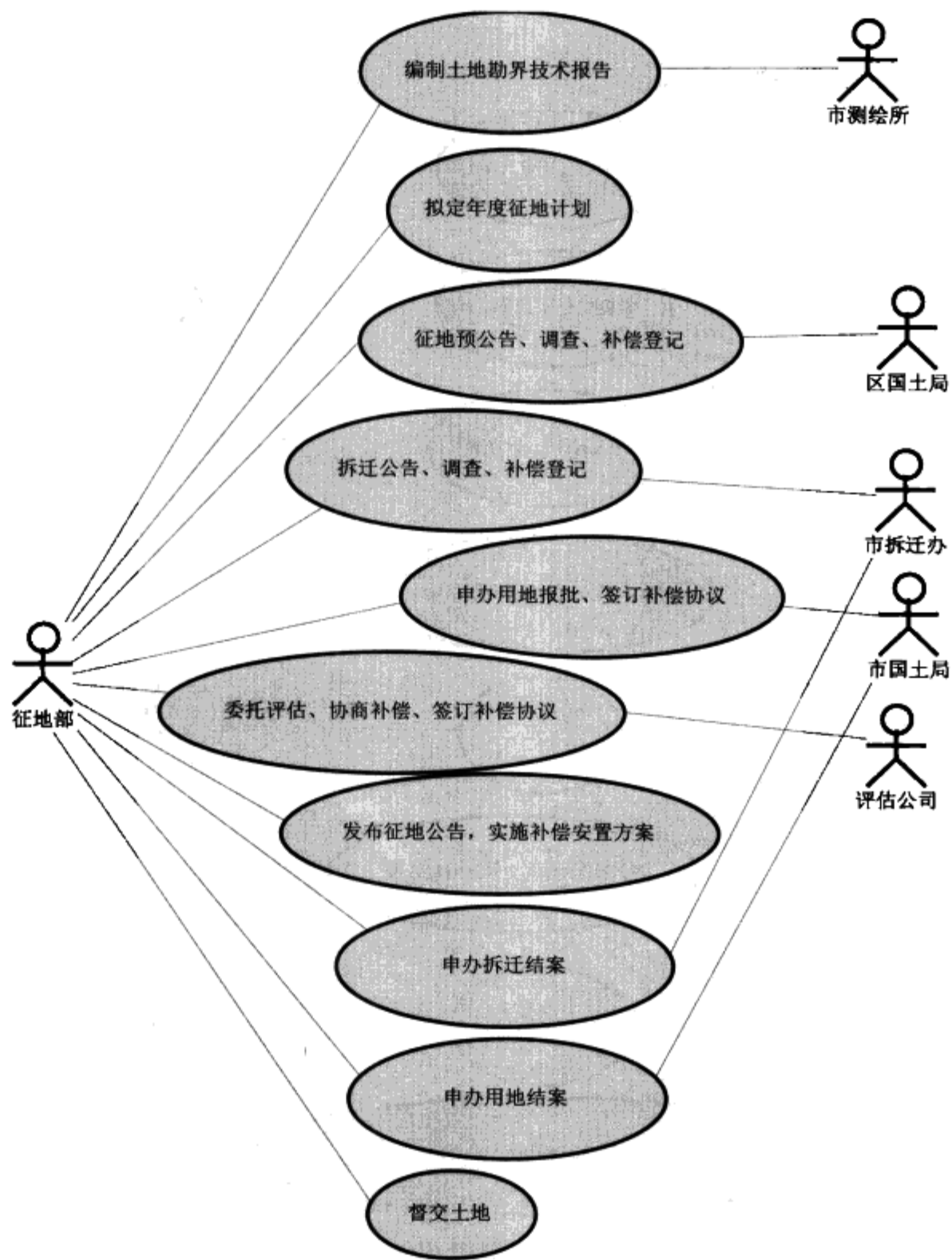


图 17.6 征地部用例设计

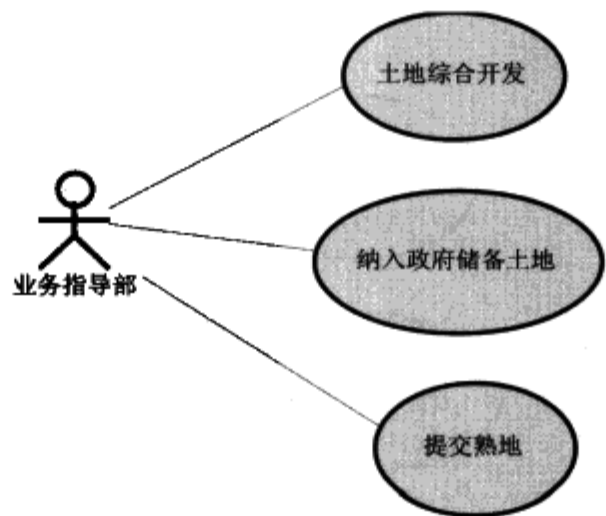


图 17.7 业务指导部用例设计

4. 出让部

出让部用例设计如图 17.8 所示。

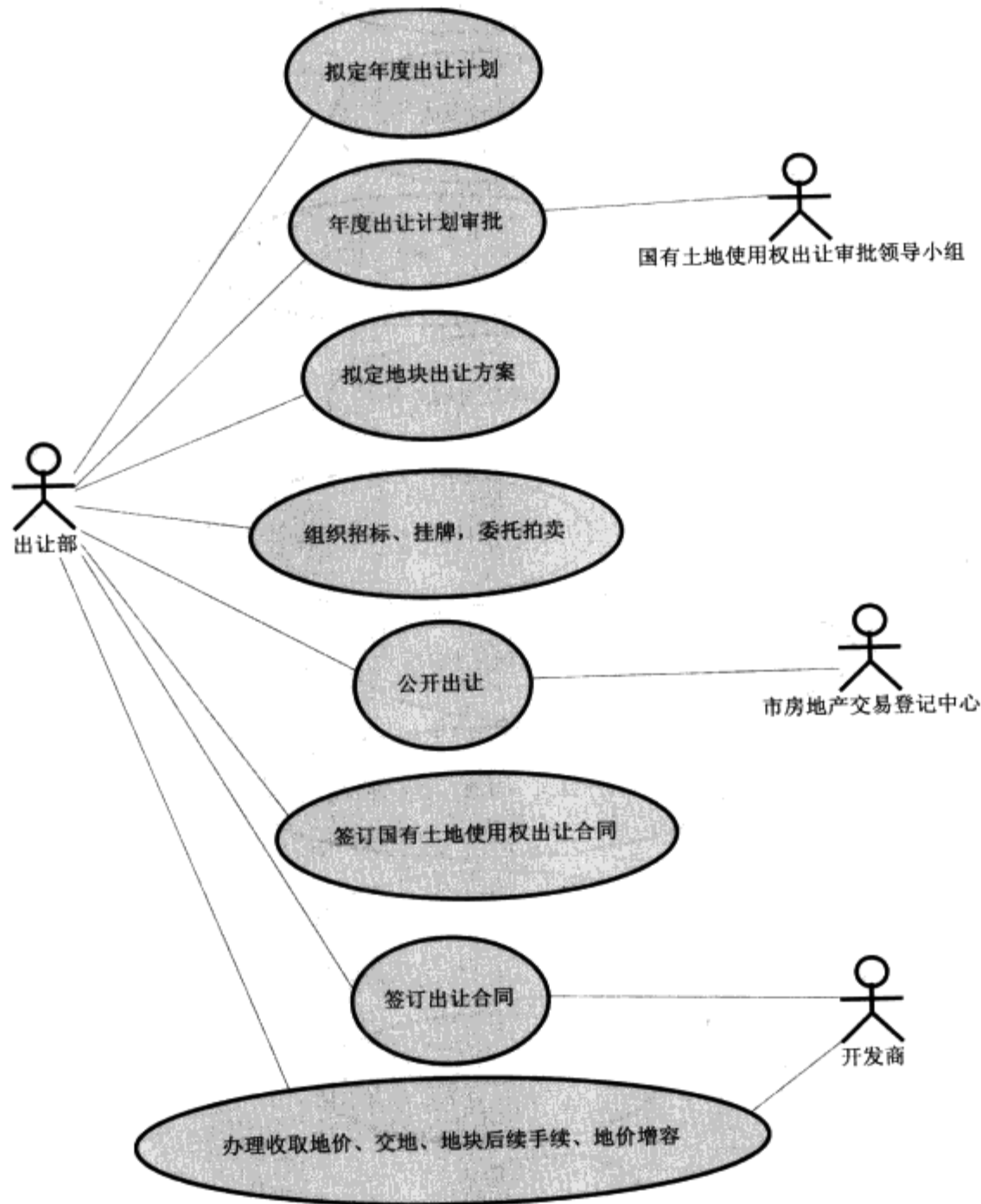


图 17.8 出让部用例设计

5. 计财部

计财部设计用例如图 17.9 所示。

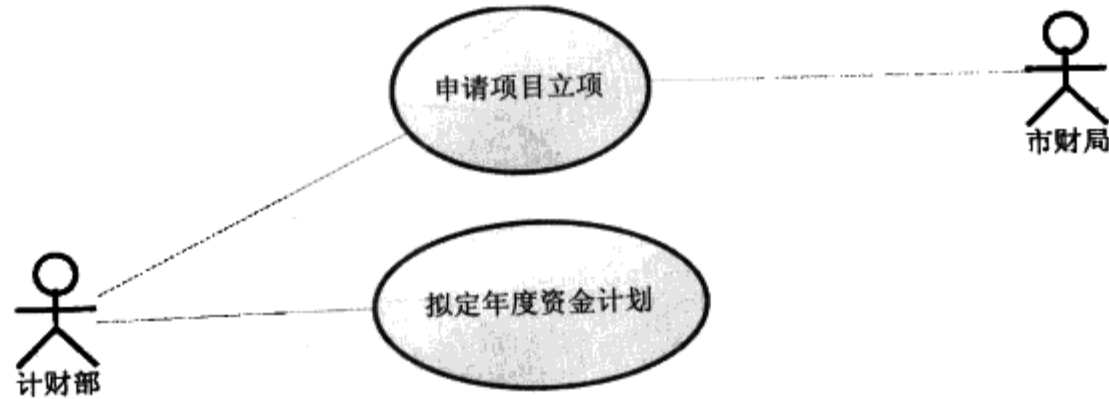


图 17.9 计财部设计用例

17.4 界面设计

由于采用 B/S 架构，整个应用系统采用 Web 方式来实现。因而包括如下各项设置。

(1) img 控件。

- alt: 所有展示类图片都要具有能简要描述图片内容的文字说明。

(2) Input 控件。

- maxlength: 所有 INPUT 控件都需要制定 maxlength 属性，默认值为数据库中对应的字段的长度。
- readonly: 所有不可更改的信息都要使用 readonly 属性。

(3) Form 控件。

- action: 所有 Form 都要指定 action，如果提交给本身就指定 action=""。
- method: 执行不可逆动作使用 POST，可逆动作使用 GET。
- onsubmit: 所有 form 都要指定提交前需要的检查程序。所有 form 都要有对应的 resetbutton。

(4) button 控件。

- onclick: form 中用于提交的 button 不允许使用此方法，所有数据检查通过 form 的 onsubmit 激活。

(5) title 属性。

所有页面都要具有和本页标题相同的 title。

(6) 控件的命名。

采用控件类型缩写（小写）+英文单词（第一个字母大写）的方法。

开发中控件涉及以下几类。

- button: btn。
- form: frm。
- select: sel。
- textarea: txt。
- input: ipt。
- UserControl: uc。

(7) 语言设置。

所有中文页面都要加上如下语句：

```
<meta http-equiv="Content-Language" content="zh-cn">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

(8) 控件属性赋值。

所有控件的属性值都要使用双引号或者单引号括起来。

进行软件设计的一个基本原则是，在用户使用 B/S 架构的网页时感觉就像操作 C/S 桌面程序一样，尽量把业务相关的工作做成 UserControl，这样可使重用性非常高，如图 17.10、图 17.11 和图 17.12 所示。



图 17.10 录入界面设计

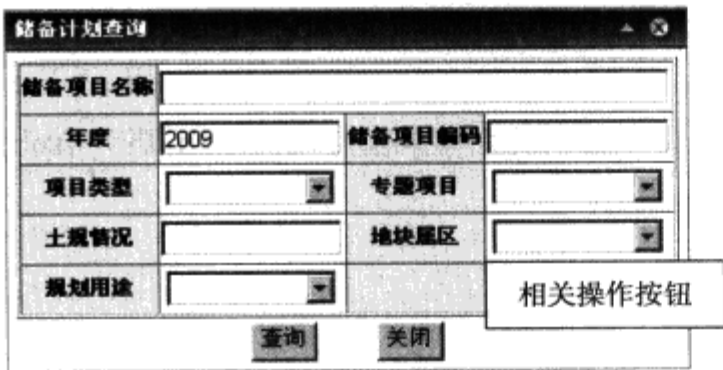


图 17.11 录入查询界面设计



图 17.12 项目列表

17.5 综合实现

本系统与其他系统的交互采用 Web Service 来进行，后台属性数据库采用 Oracle 数据库，数据库的连接与操作采用 ADO.NET。

访问 ADO.NET 中的数据源是由托管提供程序所控制。虽然托管提供程序与 OLEDB 有两处重大的不同，但是两者是极为类似的。第一，托管提供程序在 .NET 环境下运行，通过 `DataReader` 和 `DataTable.NET` 类来检索和展示数据。第二，它们的体系结构都比较简单，都是为了适应 .NET 而进行了优化。

此时，ADO.NET 分为两种不同类型的托管提供程序：一种用于 `SQLServer 7.0` 或更高版本，另一种适用于所有已安装的 OLEDB 提供程序。虽然应用在两种托管提供程序中的类是不同的，但它们却都遵循相似的命名方式。除前缀之外，其他名称都是相同的。前一种情况前缀为 `SQL`，后一种则是 `ADO`。

需要利用 `SQL` 类来访问表，因为 `SQL` 类会跳过由 OLEDB 提供程序呈现的中间层而直接进入数据库服务器内部 API。`ADO` 类是位于 OLEDB 提供程序顶端的 .NET 接口，利用 `COMInterop` 桥来进行工作。

ADO 访问数据的示例程序如下：

```
using System;
using System.Collections.Generic;
using System.Linq;
```



```
using System.Text;
using System.Data;
using System.Data.OracleClient;
namespace UserControlTask
{
    //
    public class DbOperation
    {
        private const string strConnection = "user id=GZLDC3;data source=gsonllg.up.
gz;password=ldc3gz";
        private const string FlagSuccess = "success";

        //读取数据表
        public DataSet GetDataset(string strSQL, out string strMessage)
        {
            OracleConnection con = new OracleConnection(strConnection);
            strMessage = FlagSuccess;

            OracleCommand cmd = new OracleCommand();
            cmd.Connection = con;
            DataSet ds = new DataSet();
            OracleDataAdapter ad = new OracleDataAdapter();
            cmd.CommandText = strSQL;
            ad.SelectCommand = cmd;

            try
            {
                con.Open();
                ad.Fill(ds, "tablename");
                con.Close();
            }
            catch (OracleException e)
            {
                con.Close();
                strMessage = e.Message;
            }
            catch (Exception e)
            {
                con.Close();
                strMessage = e.Message;
            }
            finally
            {
            }
            return ds;
        }

        //修改数据表
        public bool UpdateDataset(string strSQL, DataSet ds)
        {

```

```
OracleConnection con = new OracleConnection(strConnection);
OracleCommand cmd = new OracleCommand();
cmd.Connection = con;
OracleDataAdapter ad = new OracleDataAdapter();
cmd.CommandText = strSQL;
ad.SelectCommand = cmd;
OracleCommandBuilder cb = new OracleCommandBuilder(ad);

DataSet dataset = new DataSet();
try
{
    con.Open();
    ad.Fill(dataset, "tablename");
    dataset = ds;
    ad.Update(dataset, "tablename");
    dataset.AcceptChanges();
    con.Close();
    return true;
}
catch (OracleException e)
{
    con.Close();
}
catch (Exception e)
{
    con.Close();
}
finally { }
return false;
}
```

//执行 SQL 语句

```
public bool ExecuteSQL(string strSQL)
{
    OracleConnection con = new OracleConnection(strConnection);
    OracleCommand cmd = new OracleCommand();

    cmd.Connection = con;
    cmd.CommandText = strSQL;
    try
    {
        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
        return true;
    }
    catch (OracleException e)
    {
        con.Close();
    }
}
```

```

    }
    catch (Exception e)
    {
        con.Close();
    }
    finally { }
    return false;
}
}
}

```

Web GIS 的功能主要由 ArcGIS Server .NET Web ADF 来实现, 地名定位的示例程序如下:

```

public void RaiseCallbackEvent(string eventArgument)
{
    Map mapctrl = this.Page.FindControl("Map1") as Map;
    MapResourceManager mrm = this.Page.FindControl("MapResourceManager1") as
MapResourceManager;
    System.Collections.Specialized.NameValueCollection nameValueCollection =
    ESRI.ArcGIS.ADF.Web.UI.WebControls.CallbackUtility.ParseStringIntoNameValueCol
lection(eventArgument);

    try
    {
        string strName = nameValueCollection["layername"];
        string id = nameValueCollection["id"];

        IEnumerable func_enum = null;
        func_enum = mapctrl.GetFunctionalities();
        System.Data.DataTable datatable;
        foreach (ESRI.ArcGIS.ADF.Web.DataSources.IGISFunctionality gisfunction
in func_enum)
        {
            ESRI.ArcGIS.ADF.Web.DataSources.IGISResource gisresource = null;
            gisresource = gisfunction.Resource;
            bool supported = false;

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

                string[] lids;
                string[] lnames;

                qfunc.GetQueryableLayers(null, out lids, out lnames);
                if (lids == null) continue;
            }
        }
    }
}

```

```

        int selindex = -1;
        for (int i = 0; i < lnames.Length; i++)
        {
            if (lnames[i] == strName)
            {
                selindex = i;
                break;
            }
        }
        if (selindex > -1)
        {
            ESRI.ArcGIS.ADF.Web.SpatialFilter sfilter = new Spatial
Filter();

            sfilter.ReturnADFGeometries = true;
            sfilter.MaxRecords = 10;
            sfilter.WhereClause = " objectid = " + id;
            datatable = qfunc.Query(null, lids[selindex], sfilter);

            if (datatable != null)
            {
                if (datatable.Rows.Count > 0)
                {
                    for (int j = 0; j < datatable.Columns.Count; j++)
                    {
                        if (datatable.Columns[j].DataType == typeof
(ESRI.ArcGIS.ADF.Web.Geometry.Geometry))
                        {
                            ESRI.ArcGIS.ADF.Web.Geometry.Geometry geom
= (ESRI.ArcGIS.ADF.Web.Geometry.Geometry)datatable.Rows[0][j];

                            ESRI.ArcGIS.ADF.Web.Geometry.Point pt = (ESRI.
ArcGIS.ADF.Web.Geometry.Point)geom;

                            double x = pt.X;
                            double y = pt.Y;
                            double dis = 200;
                            ESRI.ArcGIS.ADF.Web.Geometry.Envelope enve
= new Envelope(x - dis, y - dis, x + dis, y + dis);
                            mapctrl.Zoom(enve);

                            break;
                        }
                    }
                }
            }
            else
            {
                throw new Exception("没有找到该地名点!");
            }
        }
    }
}

```

```
    }  
  
    }  
    m_callbackInvocation = mapctrl.CallbackResults.ToString();  
}  
catch (Exception e)  
{  
    ESRI.ArcGIS.ADF.Web.UI.WebControls.CallbackResult  
errorCallbackResult  
        = Utility.CreateMessageCallbackResult(e);  
    mapctrl.CallbackResults.Add(errorCallbackResult);  
    m_callbackInvocation = mapctrl.CallbackResults.ToString();  
}  
}
```

系统的运行界面如图 17.13 所示；编辑专题图颜色窗口如图 17.14 所示；地名定位及办理征
地进度界面如图 17.15 和图 17.16 所示。

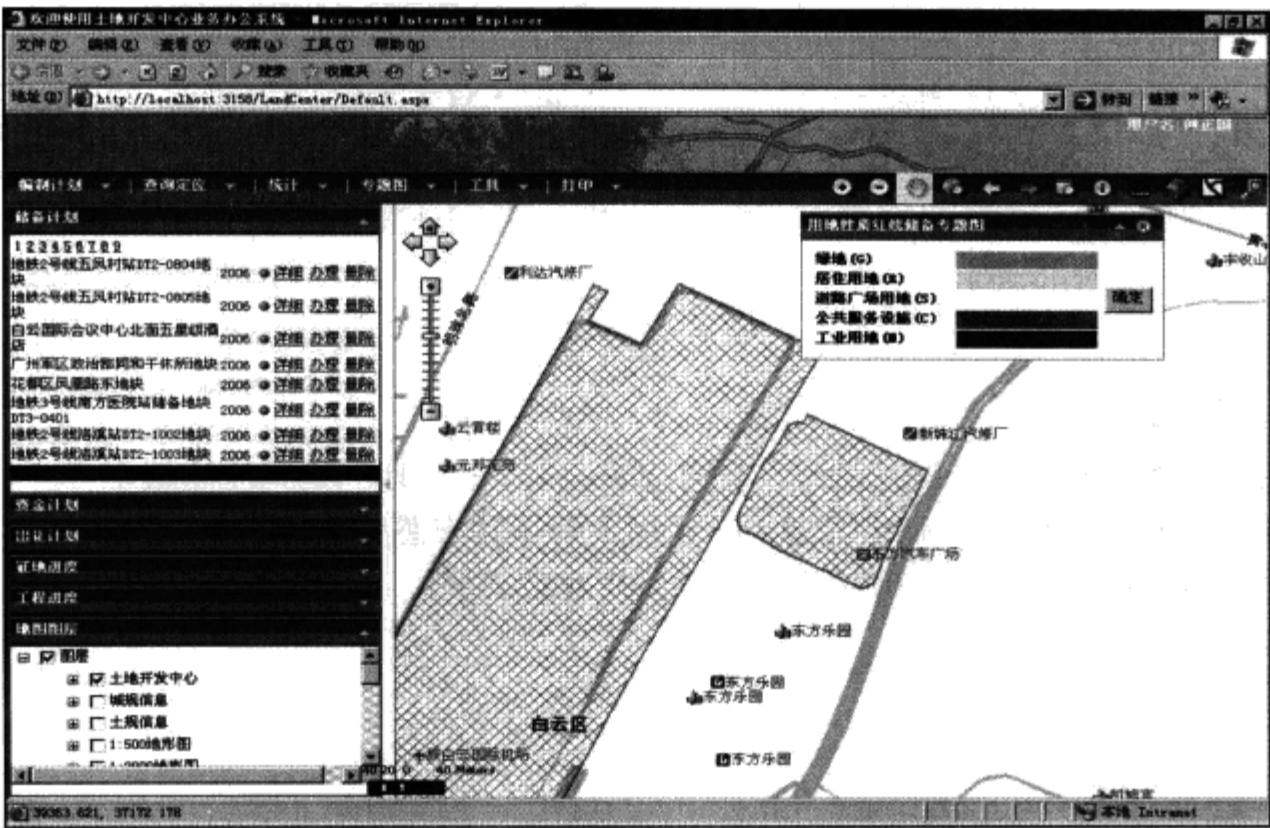


图 17.13 系统运行主界面

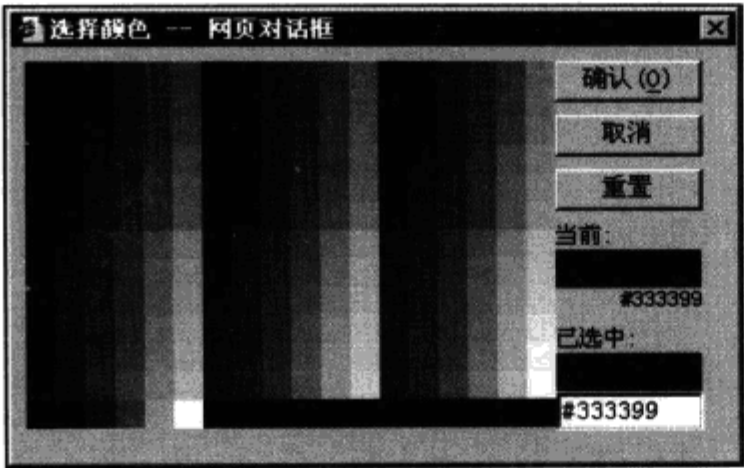


图 17.14 编辑专题图的颜色

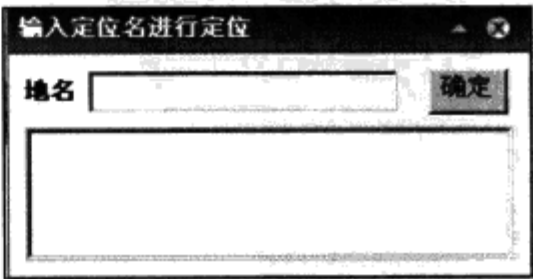


图 17.15 地名定位

办理征地进度

计划信息

储备项目名称

新客站项目

征地状态

进行中

土地面积
(平方米)

3340000

须拆迁的建
筑面积(平方
米)

20000000

征地进度年度

2009

已征地面
积(平方米)

0

待收地面
积(平方米)

2994361.4670

年度拟完成储
备量(平方米)

2994361.4670

年度计划安排
资金(万元)

1000

年度实际安
排资金(万元)

用地案件类型

储备用地

经手人

周理

现状、存在问
题及解决措施

编辑土地所有证号

计划工作内容

编号	勘界报告编号	权属单位	联系人	功能
1	11160101301	大洲村		删除
2	11160101301	屏山一村		删除
3	11160101301	胜石村		删除
4	11160101301	石壁三村		删除
5	11160101301	石壁四村		删除

征地地块信息

编号	储备项 目编码	新客站项目
所属勘界 报告编号 *	权属单位 *	
权属性质	用地案件类型	储备用地
地块面积 (平方米) *	须拆迁建筑面 积(平方米)	
拆迁补偿费 (万元)	联系人	
完成征收时间		

征地交地明细

清空

保存

关闭

17.16 办理征地进度

17.6 小结

本章主要讲述了基于 ArcGIS Server .NET Web ADF 的土地开发中心系统设计与开发。在开发过程中，设计了大量的 Web 弹出式窗口，让用户使用起来感觉就像运行 C/S 架构的 EXE 应用程序一样，而客户端不用安装任何应用程序，使用非常方便。

第六篇

常见疑难解答与 设计技巧

► 第 18 章 ArcGIS Server 常见问题及其处理

本部分主要讲述在使用 ArcGIS Server .NET Web ADF 时一些常见问题及其处理方法，这对读者在日常使用过程中有帮助。

第 18 章

ArcGIS Server常见问题及其处理

本章将对 ArcGIS Server 在使用过程中经常出现的问题进行总结。

18.1 防火墙问题

当系统升级为 Windows XP SP2 时，将安装 Windows 防火墙。其默认的设置使得 Windows 阻止所有与 ArcGIS Server 的连接。解决这一问题，需要在 Windows 防火墙设置中打开 80 端口、135 端口、ArcSOM.exe 和 ArcSOC.exe。

配置防火墙的步骤如下。

(1) 单击菜单→开始→控制面板→Windows 防火墙，默认情况下，防火墙是启用的，如图 18.1 所示。

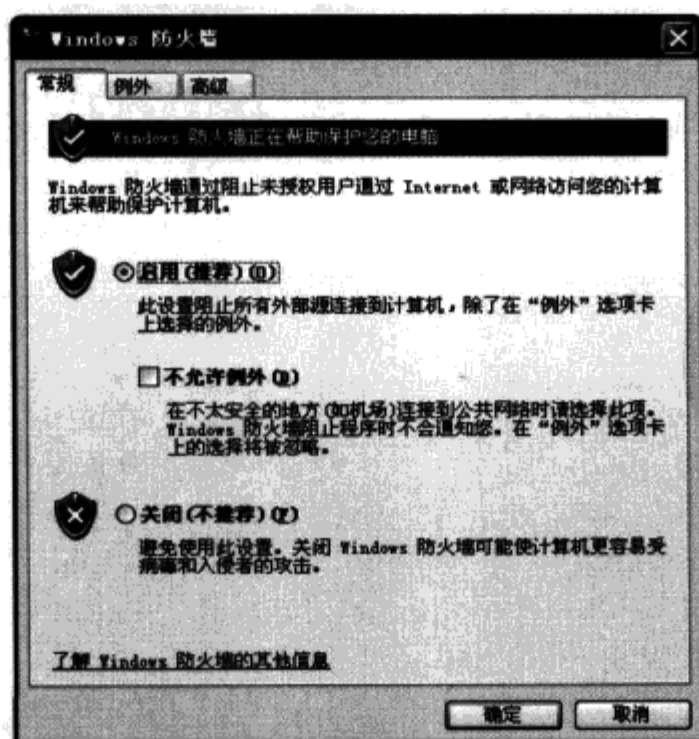


图 18.1 Windows 防火墙

(2) 选中“例外”标签卡, 单击“添加端口”按钮, 添加 80 和 135 端口, 如图 18.2 所示。

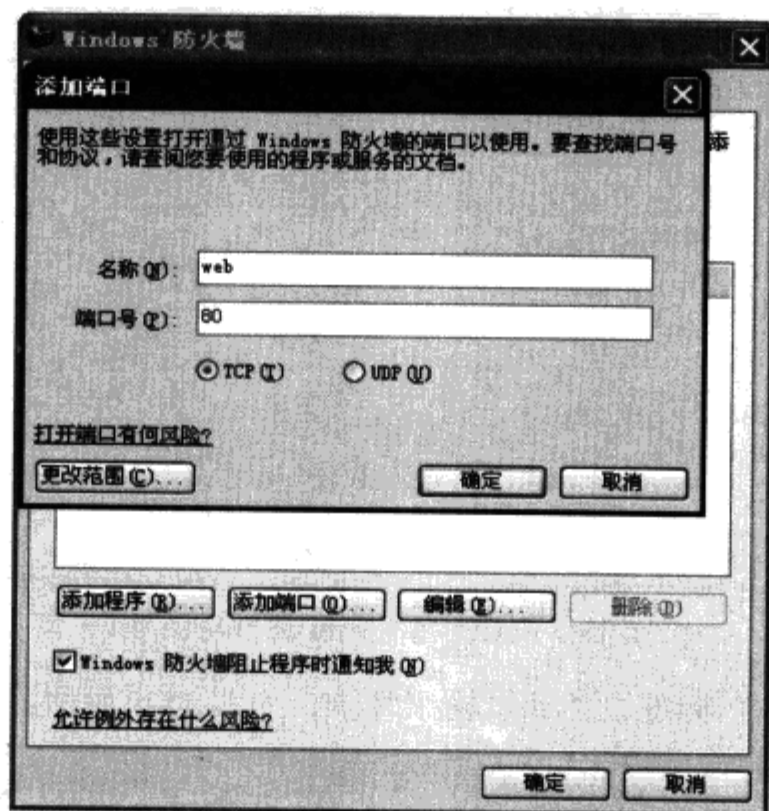


图 18.2 添加端口界面

(3) 单击“添加程序”按钮, 添加 ArcSOC.exe、ArcSOM.exe, 如图 18.3 所示。



图 18.3 添加程序界面

(4) 在单击“确定”按钮之前, 确保上面要添加的选项被选中, 这样就完成了 Windows 防火墙的设置。

18.2 ArcCatalog 中服务不能预览

在 ArcCatalog 中, 当成功地创建 Server Object 之后, 服务不能被预览。ArcCatalog 显示错误信息如下: “你的选择不能在当前视图中显示”。

导致这种错误可能的原因是: 服务器属性中被指定的输出值和 Http 位置可能不正确, 解决方法是停止所有创建的服务。

(1) 在 ArcCatalog 中, 右击已经添加的 ArcGIS Server, 并选择 Server 属性, 如图 18.4 所示。

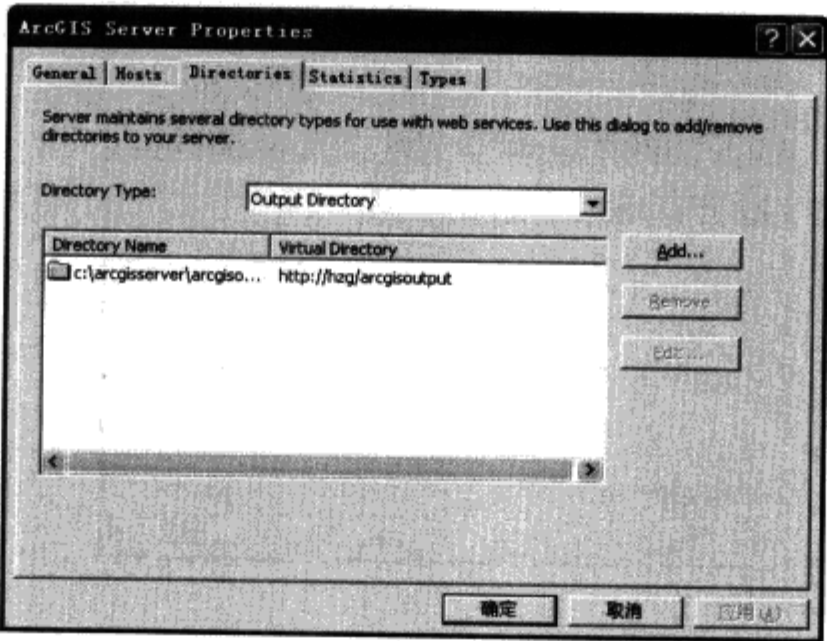


图 18.4 ArcGIS Server 属性对话框

(2) 选择 “Directories” 标签，并编辑输出目录，如图 18.5 所示。

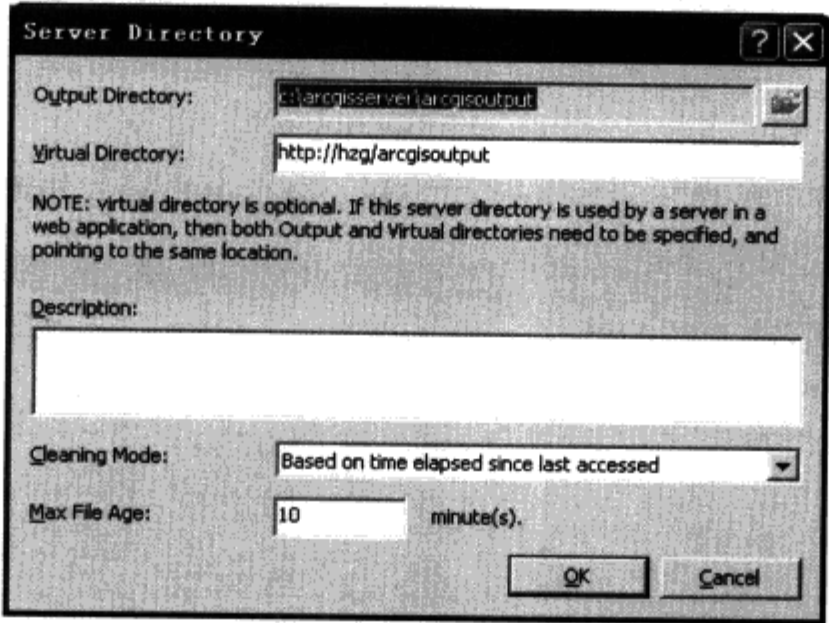


图 18.5 编辑输出目录

(3) 输出目录字段应该指向一个有效的目录，比如 C:\arcgisservice\arcgisoutput。如果虚拟目录已经被指定，那么看上去应和下面的格式一样，Http://<servername>/arcgisoutput。

(4) 如果有错误请更改；然后重新创建新的服务来确认更改是否有效。

18.3 安装错误

当安装 ArcGIS Server 时，如果出现如下错误。

"Error 1935: An error occurred during the installation of assembly component {303994BA-6487-47AE-AF1D-7AF6088EEBDB}. HRESULT: -2147024894"，那么这样的错误信息将发生在试图安装 Microsoft XML Parser 4.0 时，也可能发生在更新它的时候。总之经常会因为其他产品的安装而影响系统文件的破坏。

解决办法是。

(1) 尽可能停止一些后台程序，例如 adware 或 spyware，或者停止 Adwatch，在机器上删除所有的 adware。

- (2) 尽可能停止一些 Windows 服务。
- (3) 检查 Internet Explorer 6.0 的安装。检验 Microsoft Service 安装包和安装更新。
- (4) 在安装 ESRI 软件产品前请先安装 Microsoft XML 4.0 SP2, 这有可能下载其相关联的部分。

18.4 能否使用 new 关键字创建对象

在 ArcGIS Server 9.3 中, ADF 有很多类, 这些类可以使用 new 关键字来创建, 但是在涉及服务器端的 COM 组件时不能使用 new 关键字, 需要使用 ServerContext 的 CreateObject 来实现。例如, 创建新的 ElementGraphicsLayer 类时, 由于其为 ADF 本身固有的类, 所以可以用 new 关键字来创建:

```
ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer CreateLayer(Map
ResourceManager mapResourceManager, string layerName)
{
    ESRI.ArcGIS.ADF.Web.DataSources.Graphics.MapResource resource = Get
GraphicsResource(mapResourceManager);
    ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer layer = new
ESRI.ArcGIS.ADF.Web.Display.Graphics.ElementGraphicsLayer(layerName);
    resource.Graphics.Tables.Add(layer);
    return layer;
}
```

在创建 ChartRenderer 图表渲染时, 由于用到 ServerObject 对象, 因此不能用 new 来创建新对象, 只能用 CreateObject 创建, 示例程序如下:

```
ESRI.ArcGIS.Server.IServerContext pServerContext = GetServerContext();

ESRI.ArcGIS.Carto.IMapServer mapServer = (ESRI.ArcGIS.Carto.IMapServer)
pServerContext.ServerObject;
ESRI.ArcGIS.Carto.IMapServerObjects2 mapServerObjects = (ESRI.ArcGIS.
Carto.IMapServerObjects2)mapServer;
string mapName = mapServer.DefaultMapName;
ESRI.ArcGIS.Carto.IMap aoMap = mapServerObjects.get_Map(mapName);

ESRI.ArcGIS.Carto.ILayer pLayer = aoMap.get_Layer(nLayerID); //得到图层
ESRI.ArcGIS.Carto.IGeoFeatureLayer pGeoLayer = pLayer as IGeoFeatureLayer;

//设置专题图的属性列表
ESRI.ArcGIS.Carto.IChartRenderer pCharRenderer = pServerContext.
CreateObject("esriCarto.ChartRenderer") as IChartRenderer;
ESRI.ArcGIS.Carto.IRendererFields pRenderFields = pCharRenderer as
IRendererFields;
foreach (string var in fields)
{
    pRenderFields.AddField(var, var);
}

//实例化图表对象并取得元素指定属性的最大值
ESRI.ArcGIS.Display.IPieChartSymbol pPieSym = pServerContext.CreateObject
```

```
("esriDisplay.PieChartSymbol") as ESRI.ArcGIS.Display.IPieChartSymbol;
    ESRI.ArcGIS.Display.IChartSymbol pCharSym = pPieSym as ESRI.ArcGIS.Display.
IChartSymbol;
    pPieSym.Clockwise = true;
    pPieSym.UseOutline = true;

    pCharSym.MaxValue = GetStaMaxMin(fields, pGeoLayer)[0];

    //设置饼图外围线
    ISimpleLineSymbol pOutLine = pServerContext.CreateObject("esriDisplay.
SimpleLineSymbol") as ISimpleLineSymbol;
    pOutLine.Color = GetRGB(255, 0, 128, pServerContext);
    pOutLine.Style = ESRI.ArcGIS.Display.esriSimpleLineStyle.esriSLSSolid;
    pOutLine.Width = 1;
```

18.5 ArcGIS Server 连接方式

ArcGIS Server 两种常见连接方式：ArcGIS Server Internet 和 Arc Server Local，这两种连接方式有什么区别呢？ArcGIS Server Internet 方式连接 GIS Server 时，相当于连接标准的 Web Service。使用过 Web Service 的用户都知道，Web Service 有两类 Function：WebMethod 和一般的 Function，只有那些带有“WebMethod”的方式才能供外界用户调用，因此通过 ArcGIS Server Internet 方式连接时所得到的功能是有限的。

ArcGIS Server Local 方式连接 GIS Server 时，相当于局域网连接，可以远程调用 ArcGIS Server 提供的 ArcObjects 组件。如空间数据编辑就只能采取 ArcGIS Server Local 的连接方式。

18.6 Map 控件的 ImageBlendingMode 属性

Map 控件的 ImageBlendingMode 属性有两种：WebTier 和 Browser，如图 18.6 所示。

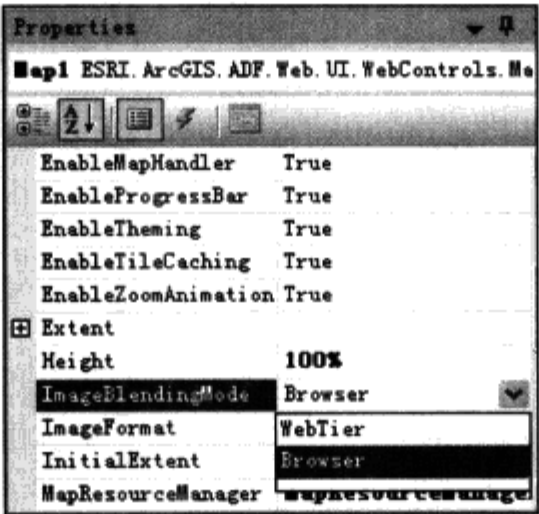


图 18.6 ImageBlendingMode 属性

WebTier 和 Browser 有什么区别呢？如果选择 Browse 的话，当 Map 控件显示的是多个地图服务时，服务器把多个地图服务的图片传递到客户端再进行融合，于是融合由客户端完成。WebTier 则是把多个地图服务的图片放在 Web Server 融合之后传递到客户端，这样会加大 Web Server 的负载，用户可以根据自身的硬件配置和负载均衡来正确配置，以达到提高应用系统效率的目的。

[G e n e r a l I n f o r m a t i o n]

书名= A r c G I S S e r v e r 开发从入门到精通 _ 1 2 5 8 6 8 6 9

S S 号= 1 2 0 9 7 1 7 4