

---

# 开源 **WebGIS**：地图发布与地图服务

发布 **0.2**

卜坤、王卷乐

**2020 年 02 月 20 日**



# Contents

<b>1 MapServer 的介绍与安装、配置</b>	<b>3</b>
1.1 MapServer 简介 . . . . .	3
1.1.1 MapServer 的发展历史 . . . . .	4
1.1.2 MapServer 的特点 . . . . .	4
1.1.3 MapServer 支持的数据格式 . . . . .	6
1.1.4 MapServer 的应用架构 . . . . .	6
1.1.5 MapServer 的两种模式 . . . . .	8
1.1.6 MapServer 和 GeoServer 的对比 . . . . .	10
1.2 Web 基础概念 . . . . .	10
1.2.1 HTTP . . . . .	11
1.2.2 URL . . . . .	13
1.2.3 HTML . . . . .	14
1.3 MapServer 运行的 Web 技术环境 . . . . .	15
1.3.1 Apache Web 服务器 . . . . .	16
1.3.2 公共网关接口（CGI）的概念与技术 . . . . .	17
1.3.3 FastCGI 介绍 . . . . .	17
1.4 安装与配置 MapServer . . . . .	18
1.4.1 安装与配置 MapServer 的基本要求 . . . . .	18
1.4.2 Debian / Ubuntu 系统下安装 MapServer . . . . .	20
1.4.3 Debian / Ubuntu 中配置 Apache . . . . .	20
1.4.4 Windows 下面安装与配置 MapServer . . . . .	21
1.4.5 编译安装 MapServer 的说明 . . . . .	22
1.5 使用 MapServer 发布遥感影像 . . . . .	23
1.5.1 在 Mapfile 中添加遥感影像 . . . . .	23
1.5.2 使用命令行工具 shp2img 生成结果 . . . . .	26
1.6 Mapfile 介绍 . . . . .	27
1.6.1 Mapfile 的组成部分与配置步骤 . . . . .	27
1.6.2 Mapfile 概述 . . . . .	28
1.6.3 Mapfile 的地图对象 . . . . .	28
1.6.4 使用 Mapfile 的注意事项 . . . . .	30
1.7 Mapfile 的辅助工具 . . . . .	31

1.7.1	Mapfile 的编辑器 . . . . .	31
1.7.2	shp2img 命令行工具 . . . . .	31
1.7.3	mappyfile 工具 . . . . .	31
<b>2</b>	<b>使用 MapServer 发布地图 (1)</b>	<b>33</b>
2.1	使用 MapServer 发布 DEM 栅格数据 . . . . .	33
2.1.1	栅格数据介绍 . . . . .	33
2.1.2	直接发布栅格数据 . . . . .	33
2.1.3	使用自动拉伸对数据进行预处理 . . . . .	35
2.1.4	根据属性值对栅格数据进行分类展示 . . . . .	36
2.1.5	在 MapServer 中控制地图范围 . . . . .	38
2.1.6	MapServer 查看的不同模式 . . . . .	39
2.2	通过 Web 查看地图 . . . . .	40
2.2.1	通过 HTTP 请求返回地图 . . . . .	41
2.2.2	通过 Web 浏览器查看结果 . . . . .	42
2.2.3	实时查看 Web 服务请求 . . . . .	42
2.2.4	Web 中的图层设计与交互元素 . . . . .	43
2.3	地图中内嵌比例尺与图例 . . . . .	44
2.3.1	地图比例尺的概念 . . . . .	44
2.3.2	在 MapServer 中创建地图比例尺 . . . . .	44
2.3.3	地图图例的概念 . . . . .	46
2.3.4	在 MapServer 中创建图例 . . . . .	47
2.3.5	legend 命令 . . . . .	48
2.4	定义投影和范围 . . . . .	48
2.4.1	地图投影的概念及常用投影 . . . . .	49
2.4.2	使用 MapServer 发布 EPSG:4326 投影世界地图 . . . . .	50
2.4.3	在 Mapfile 中定义投影 . . . . .	51
<b>3</b>	<b>使用 MapServer 发布地图 (2)</b>	<b>55</b>
3.1	使用 MapServer 发布矢量数据 . . . . .	55
3.1.1	图层的概念 . . . . .	56
3.1.2	在 Mapfile 中定义矢量图层 . . . . .	56
3.1.3	使用两个图层的静态地图 . . . . .	58
3.1.4	在 Mapfile 中定义多个矢量图层 . . . . .	59
3.1.5	MapServer 图层定义顺序 . . . . .	60
3.1.6	数据的选择 . . . . .	60
3.2	根据属性进行分类显示 . . . . .	61
3.2.1	MapServer 中分类展示的几种方法 . . . . .	62
3.2.2	使用字符串比较 . . . . .	62
3.2.3	使用正则表达式的要素分类 . . . . .	64

3.2.4	使用列表表达式进行分类 . . . . .	66
3.3	基于逻辑表达式的分类显示 . . . . .	67
3.3.1	逻辑表达式的数据类型 . . . . .	67
3.3.2	实例 . . . . .	67
3.3.3	MapServer 中支持的操作符 . . . . .	69
3.4	在地图中定义与使用符号与样式 . . . . .	69
3.4.1	MapServer 中定义符号 . . . . .	70
3.4.2	点状符号的定义 . . . . .	70
3.4.3	线状符号与样式的定义 . . . . .	74
3.4.4	面状符号定义 . . . . .	77
3.4.5	在 MapServer 中使用自定义符号 . . . . .	79
3.5	对地图进行文字标注 . . . . .	80
3.5.1	字体的定义 . . . . .	80
3.5.2	标注功能说明 . . . . .	80
3.5.3	地图标注实例 . . . . .	84
3.5.4	标注的自动优化 . . . . .	86
3.5.5	文字标注的其它问题 . . . . .	87
3.6	MapServer 输出格式 . . . . .	87
3.6.1	在 Mapfile 中定义输出格式 . . . . .	88
3.6.2	对比查看不同的输出结果 . . . . .	91
3.6.3	MapServer 输出的其他格式 . . . . .	91
<b>4</b>	<b>创建交互式地图应用</b>	<b>93</b>
4.1	MapServer 交互操作：CGI 变量及用户接口 . . . . .	93
4.1.1	MapServer CGI 程序处理流程 . . . . .	94
4.1.2	使用 MapServer CGI 程序的步骤 . . . . .	95
4.1.3	CGI 变量与模板文件 . . . . .	95
4.1.4	MapServer 初始化文件 . . . . .	96
4.1.5	配置 MapServer 临时文件夹位置 . . . . .	96
4.2	创建可交互地图：响应鼠标点击 . . . . .	97
4.2.1	修改 Mapfile 定义 WEB 支持 . . . . .	97
4.2.2	创建初始化模板文件 . . . . .	98
4.2.3	MapServer 的模板文件定义 . . . . .	99
4.2.4	运行查看结果 . . . . .	100
4.3	MapServer 地图的缩放控制 . . . . .	102
4.3.1	控制比例尺优化地图内容 . . . . .	102
4.3.2	在 Mapfile 中定义比例尺可见内容 . . . . .	103
4.3.3	控制 MapServer 的放大、缩小 . . . . .	104
4.4	MapServer GGI 图层开关控制 . . . . .	105
4.4.1	控制图层的方法 . . . . .	105

4.4.2	HTML 模板文件 . . . . .	105
4.4.3	地图控制说明 . . . . .	107
4.5	使用 MapServer 参照地图 . . . . .	107
4.5.1	使用参照地图 . . . . .	108
4.5.2	HTML 页面的定义 . . . . .	109
4.5.3	实例说明 . . . . .	109
4.6	使用独立比例尺与图例对象 . . . . .	110
4.6.1	比例尺控件的使用 . . . . .	111
4.6.2	图例控件的使用 . . . . .	112
4.7	在 MapServer 中进行属性查询 . . . . .	113
4.7.1	MapServer 中查询功能 . . . . .	114
4.7.2	MapServer 案例 . . . . .	114
4.7.3	单图层检索模式 . . . . .	116
4.7.4	多图层检索模式 . . . . .	118
4.7.5	QUERYMAP 说明 . . . . .	118
<b>5</b>	<b>Web 地图服务 (WMS) 的概念与应用</b>	<b>121</b>
5.1	使用 MapServer 发布 WMS . . . . .	121
5.1.1	MapServer 对 WMS 的支持 . . . . .	121
5.1.2	使用 MapServer 发布 WMS . . . . .	122
5.1.3	访问 GetCapabilities 接口 . . . . .	123
5.1.4	功能元数据文档分析说明 . . . . .	124
5.1.5	使用 OWSLib 模块访问 GetCapabilities 接口 . . . . .	126
5.2	WMS GetMap 访问方法 . . . . .	127
5.2.1	MapServer 中配置支持 GetMap . . . . .	127
5.2.2	GetMap 请求的参数 . . . . .	128
5.2.3	对 HTTP 方式的 GetMap 请求响应 . . . . .	131
5.2.4	使用 OWSLib 访问 GetMap . . . . .	132
5.3	WMS 的 GetFeatureInfo 请求 . . . . .	133
5.3.1	GetFeatureInfo 接口说明 . . . . .	133
5.3.2	配置 MapServer . . . . .	134
5.3.3	请求 GetFeatureInfo . . . . .	135
5.3.4	OWSLib 访问 . . . . .	137
5.4	分布式地图服务：使用 WMS 作为数据源 . . . . .	138
5.4.1	将 MapServer 用作 WMS 服务器端 . . . . .	138
5.4.2	MapServer 作为 WMS 客户端的配置 . . . . .	139
5.4.3	效果与说明 . . . . .	140
<b>6</b>	<b>MapServer 中的样式定制与实现</b>	<b>143</b>
6.1	SLD 标准介绍 . . . . .	143

6.1.1	样式层描述器介绍 . . . . .	143
6.1.2	SLD 技术细节 . . . . .	144
6.1.3	设计与使用 SLD 的方法 . . . . .	144
6.2	SLD 访问接口 . . . . .	145
6.2.1	DescribeLayer 请求 . . . . .	145
6.2.2	GetStyles 请求 . . . . .	146
6.2.3	GetLegendGraphic 请求 . . . . .	147
6.2.4	客户端的访问方式 . . . . .	149
6.3	在 WMS 中使用内联的样式层描述器：使用 SLD_BODY 参数 . . . . .	149
6.3.1	请求示例 . . . . .	149
6.3.2	获取图例 . . . . .	151
6.4	在 WMS 中使用样式层描述器：使用服务器端 SLD 文件 . . . . .	152
6.4.1	WMS 对 SLD 的支持 . . . . .	152
6.4.2	SLD 中线状效果 . . . . .	153
6.4.3	SLD 参数使用说明 . . . . .	154
6.5	用命名样式绘制地图 . . . . .	155
6.5.1	MapServer 对命名样式（Named Styles）的支持 . . . . .	155
6.5.2	使用命名样式 . . . . .	157
6.5.3	多图层与多样式 . . . . .	158
7	<b>Web 要素服务（WFS）的概念与应用</b>	<b>161</b>
7.1	Web 要素服务（WFS）的基本概念 . . . . .	161
7.1.1	WFS 的功能 . . . . .	162
7.1.2	WFS 的基本要求与接口 . . . . .	162
7.1.3	事务型 WFS 的概念 . . . . .	163
7.2	在 MapServer 中配置使用 WFS . . . . .	163
7.2.1	配置 MapServer 以支持 WFS . . . . .	164
7.2.2	请求 GetCapabilities 接口 . . . . .	165
7.2.3	请求 DescribeFeatureType 接口 . . . . .	167
7.2.4	使用 OWSLib 访问 . . . . .	168
7.3	WFS 的 GetFeature 请求与响应 . . . . .	169
7.3.1	支持的操作 WFS GetFeature . . . . .	169
7.3.2	使用参数限制数目 . . . . .	170
7.3.3	根据 ID 请求单个要素 . . . . .	170
7.3.4	根据空间范围进行检索 . . . . .	171
7.4	WFS 中使用 Filter . . . . .	172
7.4.1	Filter Encoding 标准规范 . . . . .	172
7.4.2	Filter 的三种操作符 . . . . .	172
7.4.3	MapServer 对 Filter Encoding 的支持 . . . . .	175
7.5	将 WFS 图层添加到 MapServer 地图文件 . . . . .	176

7.5.1	MapServer 用作 WFS 服务器的说明	176
7.5.2	在 Mapfile 中进行客户端定义	176
7.5.3	通过浏览器查看	177
<b>8</b>	<b>Web 覆盖服务 (WCS) 的概念与应用</b>	<b>179</b>
8.1	Web 覆盖层服务	179
8.1.1	WCS 基本概念和特征	179
8.1.2	WCS 在数据共享与分发中的作用	180
8.1.3	WCS 的主要接口	180
8.2	在 MapServer 中定义与使用 WCS	181
8.2.1	Mapfile 定义	181
8.2.2	GetCapabilities 操作	182
8.2.3	DescribeCoverage 请求	183
8.2.4	使用 OWSLib 访问 WCS	184
8.3	使用 GetCoverage 接口获取数据	185
8.3.1	GetCoverage 使用方法	185
8.3.2	WCS 输出格式	186
8.4	使用 GetCoverage 接口获取数据的更多参数与选项	187
8.4.1	空间投影	188
8.4.2	设置图像的返回尺寸	189
8.4.3	图像波段组合	189
8.4.4	空间分辨率（像元大小）	190
<b>9</b>	<b>地图瓦片技术与应用：使用 MapProxy</b>	<b>191</b>
9.1	地图瓦片的基本概念	191
9.1.1	瓦片地图的技术原理	192
9.1.2	瓦片地图的特点	193
9.1.3	瓦片地图的应用	193
9.1.4	瓦片地图使用的注意事项	194
9.1.5	矢量瓦片：下一代瓦片地图	194
9.2	地图瓦片的相关标准	195
9.2.1	创建与提供瓦片地图服务的策略	195
9.2.2	OSGeo WMS-C 标准	196
9.2.3	OSGeo TMS 标准	196
9.2.4	OGC WMTS 标准	196
9.2.5	不同标准的比较	197
9.3	MapProxy 安装与使用	197
9.3.1	MapProxy 介绍	198
9.3.2	安装 MapProxy	198
9.3.3	配置	199

9.3.4	启动 . . . . .	200
9.3.5	配置格式 . . . . .	200
9.3.6	瓦片地图预生成方法 . . . . .	202
9.4	配置 MapProxy 提供地图服务 . . . . .	203
9.4.1	配置布局 . . . . .	203
9.4.2	配置示例 . . . . .	204
9.4.3	更多图层的配置 . . . . .	207
9.5	MapProxy 实例教程：进阶 . . . . .	207
9.5.1	合并多个层 . . . . .	207
9.5.2	覆盖范围 . . . . .	208
9.5.3	定义分辨率 . . . . .	208
9.5.4	定义网格 . . . . .	209

## 10 WebGIS 前端类库 Leaflet 的使用 211

10.1	什么是 Web 制图 API? . . . . .	211
10.1.1	Ajax 技术 . . . . .	212
10.1.2	选择 Web 制图 API . . . . .	212
10.1.3	WebGIS 前端类库介绍 . . . . .	213
10.1.4	Google 地图和 Bing 地图 API . . . . .	214
10.1.5	其他 Web 制图 API . . . . .	215
10.2	LeafletJS 介绍 . . . . .	215
10.2.1	JavaScript 语言介绍 . . . . .	215
10.2.2	Leaflet 简介 . . . . .	215
10.3	LeafletJS 的基本用法 . . . . .	216
10.3.1	在 HTML 页面添加地图对象 . . . . .	216
10.3.2	Leaflet 中的几何要素用法 . . . . .	217
10.3.3	使用弹窗 (popups) 展示信息 . . . . .	217
10.3.4	事件处理 . . . . .	218
10.4	在 LeafletJS 中使用 WMS . . . . .	218
10.4.1	使用 Leaflet 调用 MapServer . . . . .	218
10.4.2	访问 GetFeatureInfo . . . . .	219
10.4.3	基础底图与专题层 . . . . .	219
10.4.4	添加切片地图 . . . . .	220
10.5	在 Leaflet 中调用 WFS . . . . .	221
10.5.1	Leaflet 中调用 WFS . . . . .	221
10.5.2	LeafletJS 加载 WFS 自定义样式 . . . . .	224
10.6	访问 MapProxy 的不同服务形式 . . . . .	225
10.6.1	WMS . . . . .	225
10.6.2	WMS-C . . . . .	225
10.6.3	TMS 访问 . . . . .	225

10.6.4 WMTS 访问 . . . . .	226
--------------------------	-----

《开源 WebGIS：地图发布与地图服务——基于 MapServer 与 Leaflet 的实践》

作者：

**卜坤** 中国科学院东北地理与农业生态研究所，高级工程师

**王卷乐** 中国科学院地理科学与资源研究所，研究员；世界数据系统-可再生资源与环境数据中心主任

### 授权声明

本书内容会在网站 <http://webgis.cn/> 发布并更新，包括在线版本、PDF 电子书版本。

本电子书及在线内容依据 CC-BY（知识共享-署名）协议进行授权：只要在使用时署名，那么使用者可以对本创作进行转载、节选、混编、二次创作以及商业目的使用。



PDF 文件下载：

- 版本：v0.2
- 发布日期：2020-2-20
- 下载地址：《开源 WebGIS：地图发布与地图服务》.

### 内容简介

本书介绍开源 WebGIS 的基本原理与技术方法，通过原理与实例结合是本书特点。

WebGIS 是非常综合的，涉及的技术庞杂，想在一本书中说明是很难的。但是作为教程，更加重要的可能是对技术框架的梳理，对不同技术之间关系的说明。在这些方面，凭借着多年实践经验，还是可以试着说一说的。

本章节要求使用 JavaScript 和 Leaflet API 进行一些编程。不需要了解有关 Leaflet 的任何信息，但是需要具备以下条件：

- 在编写计算机程序或脚本方面有足够的正式经验，就可以轻松地识别和使用基本结构，如变量、循环、决策结构、错误处理、对象等；
- 足够的 JavaScript 使用经验，可以在一段 JavaScript 代码中看到上述结构时，便可以轻松识别出这些结构；
- 具有足够的 HTML 和 CSS 经验，可以轻松查看和解释页面标记的基本元素，例如标题，正文，脚本标签等。。

技术上会涉及以下方面：

- MapServer

最初是由美国明尼苏达大学和美国太空总署（NASA）开发的一个开源的 WebGIS 软件。MapServer 是基于胖服务器/瘦客户端模式开发的 WebGIS 平台，读取地理数据，并利用 AGG 等

绘图库渲染生成 Jpeg/PNG/GIF 格式的图片后再传回客户端浏览器。MapServer 支持 Windows、UNIX、Linux 等多种平台。MapServer 支持 OGC 发布的 WMS/WFS 等若干标准规范，支持分布和互操作。MapsServer 本身是由 C 语言编写的程序，提供了两种开发模式，一种是基于 CGI 的，另一种是 MapScript 方式。MapScript 支持的包括语言：PHP，Perl，Python，Java，Tcl，C# 等。

- **MapProxy**

MapProxy 是 WMS 网页地图服务与切片服务提供者的瑞士军刀。它对已有地图服务的数据服务进行缓存、加速和转换，服务支持 OGC 标准的桌面与网页客户端。MapProxy 开发与部署方式灵活，易于同 Apache/Nginx 等环境集成。MapProxy 也能作为一个独立服务使用，这是对于新用户而言最容易的方式。

- **LeafletJS**

是一个为建设移动设备友好的互动地图，而开发的现代的、开源的 JavaScript 库。它是由 Vladimir Agafonkin 带领一个专业贡献者团队开发，虽然代码仅有 33 KB，但它具有开发人员开发在线地图的大部分功能。Leaflet 设计坚持简便、高性能和可用性好的思想，在所有主要桌面和移动平台能高效运作，在现代浏览器上会利用 HTML5 和 CSS3 的优势，同时也支持旧的浏览器访问。支持插件扩展，有一个友好、易于使用的 API 文档和一个简单的、可读的源代码。

### 前言

“人人共享”使命的核心是认为知识是一种公共利益，教育中的开放原则将为每个人提供巨大的机会。虽然我们社区的成员来自不同的背景，但我们都寻求消除数字鸿沟，赋予所有人充分的公民权利，并为建立开放知识做出贡献，造福于整个社会和子孙后代。

开源 WebGIS 教程网站是两位作者在合作过程中基于实际工作经验进行总结提炼而产生的。开源 WebGIS 对于 GIS 技术与数据共享工作都有很多意义，也有很多技巧值得分享。

# 第1章 MapServer 的介绍与安装、配置

WebGIS 的实现方法之一是利用 Common Gateway Interface (CGI) 技术。在服务器端通过 CGI 应用程序连接 WebServer 和 GIS 空间数据库，客户端仅须使用浏览器就可以对空间数据进行查询分析。著名的开源 GIS 服务器端工具 MapServer 就是使用了 CGI 技术。

MapServer 是本书中使用的主要的开源 Web 制图工具。MapServer 主站点位于 <https://www.mapserver.org/>。

相比与由商业企业提供的众多 WebGIS 解决方案，MapServer 是开源项目。这就意味着可以免费使用 MapServer，并具有自行修改、复制以及再分发的权利。需要指出的是 MapServer 作为开源项目，在不断的发展中；但这些年功能已经比较完善，更新较小。

目前主流的 Linux 操作系统如 Debian、Ubuntu 等带的 MapServer 版本都在 7.0 以上。本书介绍的内容是以 MapServer 7.0 以后版本为基础的。较新的版本往往会有新的功能，也会废弃一些不合理的设计。Redhat Linux、Centos 中没有 MapServer，需要编译安装。

这一章会说明 MapServer 的安装、配置，来了解基本的用法以及相关的工具，如何生成地图图片，以及如何通过 Web 访问。建立 MapServer 之后，需要告诉它所有资源的位置并确保所有的权限都设置正确。本书中会尽量保持应用程序的简化，方便说明，也便于在有错误出现时容易调试。

MapServer 的核心程序用来根据地理空间数据生成地图，配置使用 CGI 的功能，给予了其 WebGIS 的功能。MapServer 支持 OGC 协会的若干标准，支持分布和互操作。不论是使用 MapServer CGI 模块或服务端脚本语言设置 MapServer MapScript，都可以实现服务端与客户浏览器的动态交互。

## 1.1 MapServer 简介

MapServer 是一套基于胖服务器端/瘦客户端模式的地图发布系统。MapServer 的核心部分是用 C 语言编写的一个开源地理空间数据渲染引擎，以及一些地图操作模块，它的许多功能都依赖一些开源或免费的库。利用 GEOS、OGR/GDAL 对多种矢量和栅格数据的支持，通过 PROJ 共享库实时进行投影变换。MapServer 遵循 OGC 系列规范，可以集成开源数据库 PostgreSQL 和基于其上的 PostGIS，并对地理空间数据进行存储和 SQL 查询操作，基于 Leaflet、OpenLayers 等一系列客户端 JavaScript API 来支持对地理空间数据的传输与表达，并且遵守 OGC 制定的 WMS、WFS、WCS、WMC、SLD、GML 和 Filter Encoding 等一系列规范。

对不同数据格式的支持很重要。在实施项目时，或者全部进行格式转换，统一起来；或者尽量保持原来格式不变，这两种方式都是可行的策略。无需转换即可即时访问多种数据格式的能力使 MapServer 成为那些不能（或不会）批量转换为特定格式的人的选择之一。

无论是为网页准备静态地图图像还是发布交互式网站，MapServer 都能胜任。交互式地图允许用户放大特定区域并打开或关闭地图图层。可让用户选择地图上的要素并将其链接到其他数据的应用程序。可能性是无止境的，但最终产品通常是指向，单击和查看的过程。用户标识要查看的区域或要查看的层，然后使用请求的内容更新网页。如果需要静态地图，则可以保存地图以备后用。

MapServer 可以用作通用网关接口（CGI）应用程序，或使用通用 Web 编程语言（如 PHP，Perl，Python 和 Java）编写脚本，也可以通过命令行来运行。无论使用 CGI 版本还是自己编写脚本，MapServer 的运行时配置文件都可以控制显示哪些图层以及如何绘制它们。通过编辑配置文件，可以轻松将制图数据添加到现有应用程序。

### 1.1.1 MapServer 的发展历史

MapServer 起源于 20 世纪 90 年代中期美国明尼苏达大学（University of Minnesota, UMN）、NASA 及明尼苏达州自然资源部（Minnesota Department of Natural Resources, MNDNR）的一个合作项目——ForNet，以及之后的 TerrSIP 项目。政府的支持在 MapServer 前期的发展中起了很大的作用。MapServer 基于麻省理工学院模式的许可（MIT-style License）作为开源软件发布，用于在 Web 上发布空间数据和交互式地图应用，可支持所有主流的操作系统，包括 Windows、Linux、Mac OS X 等。它的定位不是实现全功能的 GIS，而是提供满足大多数 Web 应用的核心 GIS 功能。

MapServer 在发展壮大中，并不是孤立的，而是得到了许多开源社区和开源爱好者的支持。2005 年 11 月，提出建立 MapServer 基金会，基金会本着“促进专业的开源网络制图开发环境和社区”。即使最初集中于网络制图的项目，但希望能够给其他开源地理信息的项目提供资助”的宗旨，来促进 MapServer 的专业化发展，进行促进了整个开源网络制图技术的发展。随着开源地理信息系统软件的进一步发展以及开源网络制图环境的进一步优化，2006 年 2 月 MapServer 基金会计划放弃，而是正式以开源地理空间基金会 (OSGeo) 的机构出现，Autodesk 公司将 MapGuide 作为开放源代码加入了该基金会，进一步促进了 MapServer 的发展。

### 1.1.2 MapServer 的特点

人们决定使用 MapServer 的原因有很多。一种是使其他人，尤其是通过 Internet 广泛访问其制图信息的能力。许多 GIS 和地图分析人员需要为其支持或工作的产品创建自定义地图产品。MapServer 使用户无需安装特定工具或获得地图分析师的帮助即可创建地图。这反过来减轻了专业人员的压力。

其他人选择 MapServer 是因为它是可用于具有多种数据格式的解决方案的少数解决方案之一。通过使用 GDAL / OGR 之类的库，MapServer 可以访问各种数据格式而无需进行数据转换。

在最基本的形式中，MapServer 就是 Web 服务器上的一个的 CGI 程序。当一个请求发给 MapServer 之后，它会使用请求的 URL 中传递的 Mapfile 及相应参数，创建一个请求的地图的图像，并且可以返回图例，比例尺，参考地图及 CGI 传递的变量值。它具有以下特点：

- (1) 支持显示和查询数以百计的栅格，矢量和数据库格式，并支持分块的（tiled）矢量和栅格数据。支持的矢量格式：ESRI shapefiles, PostGIS, ESRI 的 ArcSDE 等（通过 OGR 库实现）；支持的栅格格式：TIFF/GeoTIFF, EPPL7 等（通过 GDAL 库实现）。对 Shapefile 文件，可以建立四元树空间索引。
- (2) 能够运行在多种不同的系统上（Windows, Linux, Mac OS X 等），支持许多开源应用环境，并有一些基于 MapServer 开发的应用套件。提供两种工作方式，CGI 方式（适用于 CGI、Ajax、Flex 开发人员）和 MapScript 方式（适用于 PHP、Java、C# 开发人员），以原生 CGI 方式效率最高；对流行的脚本语言和开发环境（PHP, Python, Perl, Ruby, Java, .NET）提供支持。
- (3) 高质量绘制模型。支持微软和 Apple 公司共同研制的字型标准 TrueType。地图元素（如比例尺、图例，参照图等）自动控制。图像比例自动调节。特征标注（包含标注冲突解决）。支持动态的投影变换（on-the-fly）。
- (4) Mapfile 是 MapServer 的核心，它将各种地图要素组织成具有层次关系的对象系统；并定义数据来源，使用的数据格式，用户交互以及对 OGC 协议的支持。利用逻辑或正则表达式生成专题地图。完全支持定制的 template 的输出。可以通过 URLs 动态的对 MapServer 进行配置。
- (5) 对 OGC 若干网络服务规范的支持，包括 WMS (client/server)，非事务 WFS (client/server)，WCS (server only)，WMC，SLD，GML 和 Filter Encoding。

MapServer 及其支持工具可用于许多硬件和操作系统。此外，可以通过多种编程语言界面访问 MapServer 功能，从而可以将 MapServer 功能集成到自定义程序中。MapServer 可以在其他 Web 制图服务器可能无法运行的自定义环境中使用。

由于 MapServer 是开源的，因此开发人员可以改进，修复和自定义 MapServer 背后的实际代码，并将其移植到新的操作系统或平台上。实际上，如果需要一项新功能，则可以聘请开发人员来添加它，社区中的每个人都可以从这项工作中受益。

MapServer 主要是一个查看和制图应用程序；用户通过 Web 浏览器或其他 Internet 数据共享协议访问地图。这允许使用 OGC 规范以可视方式共享地图信息并与其他应用程序实时共享数据。MapServer 可以通过读取各种格式并使用通用协议提供对另一台服务器或应用程序的访问来执行伪数据转换。MapServer 不是分析工具，但它可以使用不同的制图技术显示地图信息以可视化结果。

### 1.1.3 MapServer 支持的数据格式

MapServer 支持多种格式。有些是 MapServer 程序内部支持的，而另一些则通过 GDAL / OGR 库进行访问。通过库进行的访问会在 MapServer 和数据源本身之间增加额外的通信，在某些情况下可能会导致性能下降。

MapServer 根据以数字格式存储的空间信息创建地图图像。它可以处理矢量和栅格数据。MapServer 本机读取两种栅格格式：GeoTIFF 和 EPPL7，但可以通过 GDAL 包读取支持的格式（包括 Windows 位图，GIF 和 JPEG）。更值得关注的是，MapServer 还支持网络数据资源（如 WMS、WFS）作为图层，即从其他 WMS 服务器获取地图图片来构成地图对象的一个图层。

通常，MapServer 本身支持的格式应比使用 GDAL / OGR 的格式运行得更快。例如，MapServer 使用的最基本格式是 ESRI shapefile 或 GeoTiff 图像。OGR 支持美国人口普查 TIGER 文件格式。加载 TIGER 或 Shapefile 之间的性能差异可能很大。但是，使用 GDAL / OGR 可能不是问题。进一步的研究表明，数据格式通常是瓶颈。如果文件中的数据以难以访问的方式构造或需要多种解释级别，则它将影响地图绘制速度。

获得最佳性能的一般经验法则是使用 ESRI shapefile 格式或 GeoTiff 图像格式。由于 gdal\_translate 和 ogr2ogr 可以写入这些格式，因此可以使用这些工具来转换大多数源数据。如果通过网络访问数据，则将数据存储在 PostGIS 数据库中可能是更好的选择。由于 PostGIS 直接在服务器上处理对数据的查询，因此可以只将期望的结果会通过网络发送回去。而基于文件的数据，在 MapServer 决定需要哪些数据之前，必须传递更多的数据。PostGIS 数据库中的服务器端处理可以显着提高 MapServer 应用程序的性能。

在本书的介绍中会使用 Shapefile 与 SpatiLite 两种矢量数据格式；以及 GeoTIFF 栅格数据格式。使用其他数据格式在具体的用法上有一些差异，需要进一步阅读相关技术文档。

### 1.1.4 MapServer 的应用架构

一个完整的 WebGIS 应用至少包括三个层次的体系结构：

- 应用层：客户端浏览器；
- 中间层：MapServer CGI 模块 / MapScript 脚本环境，Web 服务器；
- 存储层：GIS 空间数据库。

一个简单的 MapServer 包含以下几部分，如 图 1.1 所示：

Mapfile 是 MapServer 应用的结构化的文本配置文件，所有的数据资源与字体资源都要在 Mapfile 中声明。它定义了地图的范围，用来告诉 MapServer 数据的位置以及在输出的路径；它还定义了地图图层，包括数据源，投影和符号。Mapfile 的扩展名必须为 .map，否则 MapServer 识别不了。Mapfile 文件将各种地图要素组织成具有层次关系的对象系统，用户界面和对 OGC 协议的支持也在 Mapfile 中定义。

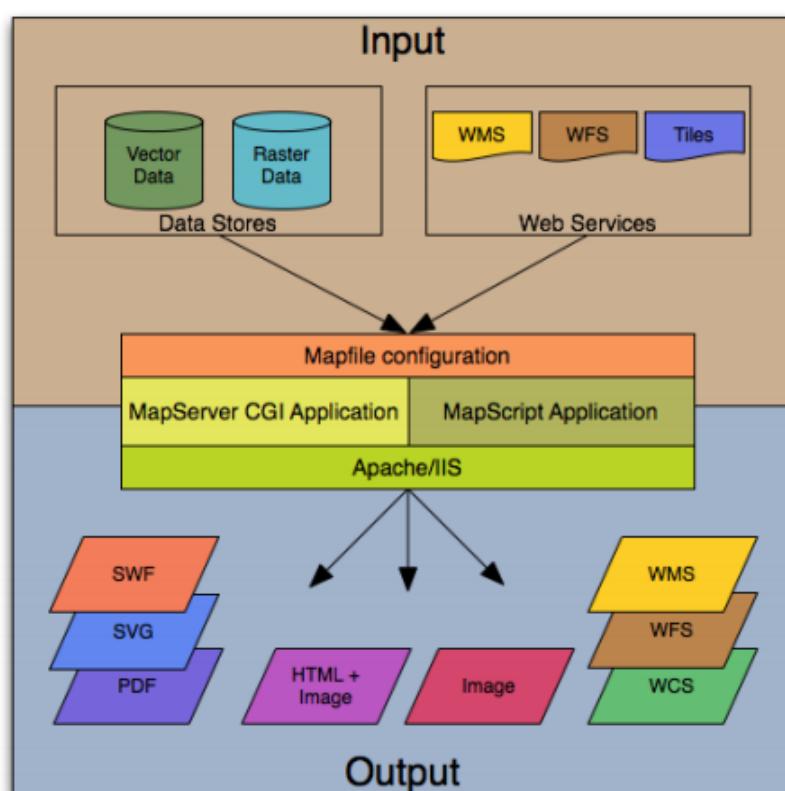


Fig. 1.1: The basic architecture of MapServer applications.

图 1.1: MapServer 架构

HTML 页面是可选的，在 MapServer 中一般作为模板，用以包装数据以进行展示。HTML 页面是用户和 MapServer 之间的接口。它们通常位于 Web 根目录。在最简单的形式中，MapServer 可以被调用作为一个静态的地图图像到 HTML 页面上；为了使地图能够交互，图像被放置在页面的一个 HTML 表单上。

MapServer CGI 程序是二进制的可执行文件。可以接收请求并返回图像，数据等。它位于 web 服务器的 cgi-bin 或者 scripts 目录下。Web 服务器的用户必须有这些目录的执行权限。出于安全的考虑，它们不能位于 WEB 的根目录下。在 Unix/Linux 操作系统下这个程序一般为 mapserv，在 Windows 操作系统下则带有 .exe 扩展名。

WEB/HTTP 服务器的作用是当用户的浏览器发起请求时提供 HTML 页面。例如 Apache 或者 Microsoft 的 IIS，MapServer 需要使用它们来提供服务，耦合得较紧，与 MapServer 运行在同一台机器上。这种方式可以视为传统的三层结构。

MapServer 通常在 Web 服务器应用程序后面运行。Web 服务器接收对地图的请求，并将其传递给 MapServer 进行创建。MapServer 生成请求的地图图像并将其交给 Web 服务器，然后将其发送回用户。显示了用户如何与 Web 服务器进行交互，然后 Web 服务器向 MapServer 程序发出请求。

每当用户向 MapServer 提出新地图请求时，都会发生此绘制过程（也称为渲染），例如，当用户放大地图以进行仔细查看时。当用户手动请求重绘时（例如，一个数据层的内容发生更改，并且用户希望查看更改时），也会发生此过程。

### 1.1.5 MapServer 的两种模式

MapServer 可以在两种不同的模式下运行：CGI 和 MapScript。在 CGI 模式下，MapServer 在 Web 服务器环境中作为 CGI 脚本运行。这很容易设置并产生快速，直接的应用程序。在 MapScript 模式下，可以从 PHP，Perl，Python，Java，Tcl 或 C# 访问 MapServer API。MapScript 接口允许灵活，功能丰富的应用程序，仍然可以利用 MapServer 的模板工具。在服务器端可以使用任一模块，编写 WebGIS 程序。不过它们使用相同的配置文件 Mapfile。

#### CGI 模式

使用 CGI 模块编程就是利用 CGI 变量和它的替换变量，而 CGI 模块是编译好的程序 (mapserv.exe 或 mapserv)；使用 CGI 模块难以实现较为复杂的动态交互，但是这一情况随着现代 WebGIS 服务的产生在改变着。利用其他服务端脚本语言和 MapScript 可以更容易实现复杂的 WebGIS 功能。

CGI 是进行 MapServer 应用开发最基本的方法。在 CGI 模式下，MapServer 利用 AGG 图形库将空间数据转化成地理图形再传回到用户的浏览器。用户不需要安装任何特定的浏览器程序就可以通过浏览器对其进行操作。

用户可以通过类似下面的 URL 直接向 MapServer 服务器发出请求，MapServer 根据请求的参数创建合适的地图图片，并将结果返回给用户的浏览器：

如 <http://webgis.cn/123>

URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfa1.map
2 layer=world-country
3 mode=map
```

这里只需要打开链接查看，不必深究 URL 的含义，相关说明会在后面展开。

## MapScript 模式

MapScript 是一个 API（应用程序接口），允许应用程序使用多种语言来调用 MapServer 的功能，然后以希望的方式使用结果。MapScript 将开发者从基于 WEB 用户界面的局限性中释放出来并提供了访问其他工具（如数据库管理系统）的途径，这样非常强大的应用可以被构造。这样的应用程序可以作为工具套件的一部分提供地图绘制和空间查询能力。

在 MapServer 的官方网站中，现在支持 MapScript 的语言包括 PHP 和 Python。PHP 对 MapScript 的支持更好一些。关于 Python 的 MapScrpit，MapServer 中集成的只支持 Python 2.7 版本；另外有独立的 PyPI 维护的代码 (<https://pypi.org/project/mapscript/>) 可以支持 Python 3，但目前只支持 Windows 下面 Visual Studio 编译，对于 Linux 的支持在计划之中。

肖恩 · 吉利斯（Sean Gillies）曾经是 MapScript 维护人员（直到 2006 年），他有几篇关于 MapScript 的博客文章有力地说明了应该使用 Mapfile 而不是 Mapscript。尤其是最近 WebGIS 出现的一些设计理念与方法，可以使用更加直观与灵活的方式来实现一些功能。因此，本书中介绍的内容不会涉及到 MapScript。

MapScript 绑定有几种不同的语言，这要归功于 SWIG 为 C 语言创建了包装代码。

SWIG 是由 David Beazley 开发的，后来，他建立了最初基于哪个 Mapfile 的 PLY。PLY 是针对 Python 的 Lex 和 Yacc 解析工具的一个实现——MapServer 本身用于在 C 语言中解析 Mapfile 的工具。

### 1.1.6 MapServer 和 GeoServer 的对比

MapServer 与 GeoServer 都是得到广泛应用的开源 WebGIS 工具，这两个都是通过网络来发布地图，也常常会被人拿来比较。通过比较，也可以进一步了解 MapServer 的特点。MapServer 同时符合 OGC 的 WMS 和非事务性 WFS；GeoServer 是用 Java 编写的，基于 servlet 并使用 struts 框架。

很难区分 MapServer 与 GeoServer 哪个更好，两者都很出色，并且都有优秀的文档和用户群。GeoServer 实现了在线编辑空间数据，生成专题地图。地图发布是用 XML 文件。MapServer 擅长于生成专题地图，比以前更成熟了，地图发布是通过文本配置文件。MapServer 在功能方面弱于 GeoServer，它不是一个功能齐全的 GIS，没有提供集成的 DBMS（数据库管理系统）工具，分析能力有限，而且没有地理配准工具。

如果只是发布地图而不允许修改的话可以使用 MapServer，MapServer 对 WMS（Web Map service）的支持更为高效，维护起来更简单容易些。选择 GeoServer 是因为想要更好的功能，Geoserver 更擅长于结合 WFS（Web Feature service）规范的属性查询，如在线编辑和数据库的支持像 PostgreSQL 或 Oracle 空间数据库；GeoServer 的另一个优势就是有一个免费的客户端软件 UDIG。

除了功能上的比较，在开发中进行技术选择时尤其要注意，MapServer 专注于地图服务功能，可以作为项目的组件（甚至是核心组件），但是其他的功能大部分需要由开发人员实现；而 GeoServer 则是已经比较完善的套件，部署安装后基本上就可以作为产品来使用了。

## 1.2 Web 基础概念

Web 应用遵循客户端/服务器（C/S）的架构模式。其中，客户端被称为 Web 客户端，服务器被称为 Web 服务器。一个典型的 Web 应用系统通常是三层体系架构，即数据层、逻辑层和表现层，如 图 1.2 所示。很多情况下，Web 客户端是 Web 浏览器。因此，Web 应用架构也被称为浏览器/服务器（B/S）架构。

Web 应用的基本工作流程如下：

- 用户使用 Web 客户端（往往是 Web 浏览器），通过在地址栏上输入网络资源的 URL（Uniform Resource Locator; 统一资源定位符）地址，或点击网页上的 URL 链接，向 Web 服务器发出请求；
- Web 服务器收到请求后，搜索服务器上存储的文件或脚本，并将文件或执行脚本得到的结果返回给 Web 客户端；
- Web 客户端收到响应结果，并把返回的结果渲染和显示出来。

HTTP、URL 和 HTML 是 Web 技术的三大基石。这些协议由 Tim Berners-Lee 提出，现在已成为国际标准，由 W3C(世界万维网联盟) 管理与维护，以下分别对其进行简要介绍。

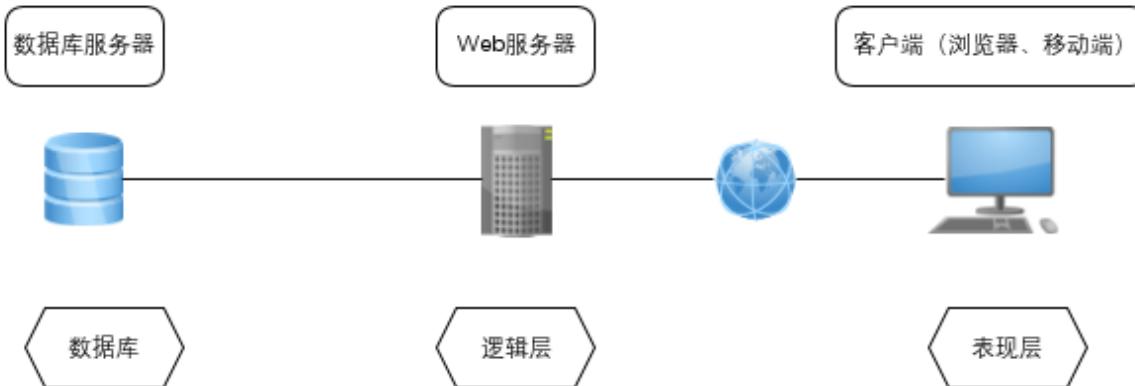


图 1.2：基本 WEB 应用的三层架构

### 1.2.1 HTTP

大多数 Web 应用中涉及两个或多个组成部分，因此需要定制一套各组成部分都应该遵循的规范。HTTP 作为一种协议，则是定义了一套 Web 服务器与客户端进行请求和应答时所应遵守的规范。例如，HTTP 请求和应答信息都包含消息头和消息体，根据 HTTP 协议，Web 服务器知道把哪些信息放在消息头中、哪些放在消息体中，Web 客户端也知道应当到消息头中和消息体中读取哪些信息。

HTTP 是一个属于应用层的面向对象的协议，基于 TCP/IP 通信协议来传递数据（HTML 文件，图片文件，查询结果等），由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于 1990 年提出，经过多年的使用与发展，得到不断地完善和扩展。

HTTP 协议的主要特点包括：

- 简单：键入 URL 或点击链接即可执行。
- 无记忆状态：服务器对客户端做出响应后，它们之间的连接会被立即撤消，服务器不会保留客户端相关信息，以减轻服务器的负载。
- 灵活：可以传输的内容类型丰富，包括参数、图片、PDF、音频和视频等。

HTTP 的状态码

HTTP 状态码（英语：HTTP Status Code）是用来表示网页服务器超文本传输协议响应状态的 3 位数字代码。它由 RFC 2616 规范定义的，并得到 RFC 2518、RFC 2817、RFC 2295、RFC 2774 与 RFC 4918 等规范扩展。所有状态码的第一个数字代表了响应的五种状态之一。所示的消息短语是典型的，但是可以提供任何可读取的替代方案。除非另有说明，状态码是 HTTP / 1.1 标准（RFC 7231）的一部分。

常见的 HTTP 状态码，如 表 1.1：

表 1.1: 状态码列表

状态码	功能
200 OK	客户端请求成功
400 Bad Request	客户端请求有语法错误，不能被服务器所理解
401 Unauthorized	请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用
403 Forbidden	服务器收到请求，但是拒绝提供服务
404 Not Found	请求资源不存在，eg：输入了错误的 URL 没有查询到正确的地址
500 Internal Server Error	服务器发生不可预期的错误
503 Server Unavailable	服务器当前不能处理客户端的请求，一段时间后可能恢复正常

HTTP 共定义了八种请求方式，分别为 GET、POST、HEAD、PUT、DELETE、TRACE、OPTIONS 和 CONNECT，其中 GET 和 POST 最为常用。HTTP 消息头中包含缓存控制信息、内容类型和状态码等。

以请求 <http://webgis.cn> 网站为例：

请求头：

```
GET / HTTP/1.1
Host: webgis.cn
```

响应头：

```
HTTP/1.1 200 OK
Date: Wed, 06 Nov 2019 00:57:18 GMT
Server: Apache/2.4.25 (Debian)
Last-Modified: Sun, 03 Nov 2019 09:14:41 GMT
ETag: "4bdd-5966da115de5d-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 6631
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

响应头中主要包含了服务器的信息。GET 与 POST 是最常用的 HTTP 请求方式。

GET 提交时请求的数据会附在 URL 之后（就是把数据放置在 HTTP 协议头中），以？分割 URL 和传输数据，多个参数用 & 连接；如果数据是英文字母/数字，原样发送，如果是空格，转换为 +，如果是中文或其他字符，则直接把字符串用 BASE64 方式进行编码。

在 GET 请求中，浏览器会对 URL 的长度有所限制，下面是几种常见浏览器的 URL 长度限制:(单位: 字节)

表 1.2: 浏览器的 url 长度限制

浏览器	长度限制:(单位: 字节)
IE	2803
Firefox	65536
Chrome	8182
Safari	80000
Opera	190000

POST 提交把提交的数据放置在是 HTTP 包的包体中。GET 提交的数据会在地址栏中显示出来，而 POST 提交地址栏不会改变，所以 POST 的提交方式要比 GET 的安全性高。POST 提交由于不是通过 URL 传值，理论上数据不受限。但实际上 POST 所能传递的数据量大小取决于服务器的设置和内存大小。

HTTPS (Secure Hypertext Transfer Protocol; 安全超文本传输协议) 是建立在安全套接层之上的安全版 HTTP。在普通 HTTP 连接中，服务器和客户端间传输的数据可以被拦截；而 HTTPS 通过加密避免数据被窃听，通常用来传输敏感数据，如用户的个人信息、登录密码和信用卡信息。采用 HTTPS 协议的网站需要在服务器上安装认证机构所颁发的证书文件。

## 1.2.2 URL

HTTP 使用统一资源标识符 (Uniform Resource Identifiers, URI) 来传输数据和建立连接。URL 是一种特殊类型的 URI，包含了用于查找某个资源的足够的信息 URL, 全称是 UniformResourceLocator, 中文叫统一资源定位符，是互联网上用来标识某一处资源的地址。简单来讲，URL 就是 Web 地址，俗称“网址”。每个网页都有一个全球唯一的 URL 标识。就像现实生活中街道地址用来定位家庭住址一样，网址是用来定位万维网上成千上亿的网页的。没有 URL，Web 客户端就无法找到 Web 服务器上的资源；没有 URL，互联网上的资源也不能相互连接形成万维网。

HTTP 支持两个请求方法：GET 和 POST 传统的 MapServer 请求，或基于 OGC 标准的请求都定义用来调用操作的 HTTP GET。用于 HTTP GET 请求的在线资源 URL 事实上仅仅是一个 URL 前缀，为了建立一个有效的操作请求，在其后还添加了另外的参数。URL 前缀被定义为一个不透明的字符串，它包括协议、主机名、端口号（可选）、路径、和一个问号？，还可以包括一个或几个用于具体服务器的参数并以 & 结束。

URL 的基本格式是，如 表 1.3

```
Protocol: //hostname[port]/filepathname?query_string
```

表 1.3: URL 的字段代表的意义

URL 的字段	意义
Protocol(协议)	指客户端与服务器间的传输协议，常用的有 HTTP、HTTPS、FTP 和 MMS 等协议。
hostname (主 机 名) 或 IP 地址	是指存放资源的服务器。
Port(端口号)	省略时系统将使用默认端口。例如，HTTP 默认端口为 80；HTTPS 默认端口为 443。
path(文件路径)	表示 Web 服务器上资源存放的目录和文件名（这些目录和文件名可以是虚拟的，即不是真正的文件路径）。
query_string(查询字 符串)	可选，用于向 Web 服务器发送 HTTP 请求参数。

URL 规范保留了一些特定的字符并赋予它们必要的意义，如 表 1.4：

表 1.4: 特定的字符

字符	意义
?	查询语句开始的分隔符。
&	查询语句参数之间的分隔符。
=	参数名字和参数值之间的分隔符。
/	格式参数值中 MIME 类型子类型之间的分隔符。
:	SRS 参数值中命名空间和标识之间的分隔符。
,	清类型参数中单个值的分隔符（例如 GETMAP 请求中的 BBOX, LAYERS 和 STYLES）。

### 1.2.3 HTML

HTML 称为超文本标记语言英文全称是 Hypertext Marked Language，是一种标识性的语言。它包括一系列标签。通过这些标签可以将网络上的文档格式统一，使分散的 Internet 资源连接为一个逻辑整体。HTML 文本是由 HTML 命令组成的描述性文本，HTML 命令可以说明文字，图形、动画、声音、表格、链接等。

HTML 是用于创建网页的主要语言，目前大多数网页源代码都使用这种格式。与 Word 文档相似，HTML 也包含内容、布局和格式等信息。当 Web 页面被加载到 Web 浏览器时，Web 浏览器解释 HTML 代码，按照其中指定的格式显示网页内容。作为一种标记语言，HTML 是由

一套标签所标识的纯文本文件，如 head、body、table、center 和 fonts 等标签。此外，HTML 的外观和布局信息可由 CSS(cascading style sheets，层叠样式表) 来定义。CSS 可以直接包含在 HTML 文件中，或者存放在一个单独的文本文件中被 HTML 引用。

发展到今天，HTML 标准将逐渐更替为 HTML5。HTML5 是包括 HTML、CSS 和 JavaScript 在内的一套技术组合，它希望能够在不依赖 Adobe Flash 和 Microsoft Silverlight 等插件的情况下实现丰富的互联网应用。HTML5 添加了许多新的语法特征，其中包括 <video>、<audio> 和 <canvas> 元素；同时，集成了可缩放矢量图形 (scalable vector graphics, SVG) 内容。这些元素是为了更容易地在网页中添加和处理多媒体和图片内容而添加的。它还增加了其他元素，强化了 Web 网页的表现性能，追加了本地数据库等能力，提高了缓存和离线运行的能力。

超级文本标记语言文档制作不是很复杂，但功能强大，支持不同数据格式的文件嵌入，其主要特点如下：

- 简易性：超级文本标记语言版本升级采用超集方式，从而更加灵活方便。
- 可扩展性：超级文本标记语言的广泛应用带来了加强功能，增加标识符等要求，超级文本标记语言采取子类元素的方式，为系统扩展带来保证。
- 平台无关性：超级文本标记语言可以使用在广泛的平台上
- 通用性：另外，HTML 是网络的通用语言，一种简单、通用的全置标记语言。它允许网页制作人建立文本与图片相结合的复杂页面，这些页面可以被网上任何其他人浏览到，无论使用的是什么类型的电脑或浏览器。

在 Firefox 中打开 URL <http://webgis.cn>，在页面空白处右键，在弹出的上下文菜单中点击“查看源代码”就可以看到网页后面的 HTML 内容。

### 1.3 MapServer 运行的 Web 技术环境

与完整的软件套件相比，MapServer 运行环境的搭建可能比较麻烦一些，但是其灵活性也是得到许多人青睐的原因。本节对 MapServer 运行涉及到的其他技术做一下说明，了解在安装、配置中会遇到的一些问题。

MapServer 本身是生成地图的程序，但是提供了 CGI 接口，可以通过 Web 访问来调用相关功能。所以很多时候为了运行 MapServer，需要安装 Web 服务器（如 Apache 2，Nginx，或微软的 IIS），以及让应用程序（MapServer）与 Web 服务器（Apache）通信的工具（如 FastCGI）。从技术上讲，MapServer 被认为是基于 HTTP 的无状态进程，无状态意味着它处理一个请求然后会停止运行。

MapServer 的最简单形式是在 Web 服务器上作为可执行的 CGI 应用程序运行。MapServer CGI 可执行文件充当地图数据文件和请求地图的 Web 服务器程序之间的中间人。这些请求以 CGI 参数的形式从 Web 服务器传递到 MapServer CGI；MapServer CGI 程序访问一个地图文件，从数据源中获取信息；MapServer 应用程序根据信息进行处理，创建的地图并反馈到 Web 服务器；最终反馈到用户的 Web 浏览器。

MapServer 的运行取决于 Web 服务器，MapServer 应用程序开发人员需要意识到 Web 服务器的开发可能影响其应用程序的更改。

### 1.3.1 Apache Web 服务器

在本书配套网站中使用 Apache Web 服务器为 MapServer 提供基本的 Web 服务能力。Apache Web 服务器的成功之处主要在于它的源代码开放、有一支开放的开发队伍、支持跨平台的应用（可以运行在几乎所有的 Unix、Windows、Linux 系统平台上）以及它的可移植性等方面。

Apache HTTP Server（简称 Apache）是 Apache 软件基金会的一个开放源码的网页服务器，可以在大多数计算机操作系统中运行，由于其多平台和安全性被广泛使用，是最流行的 Web 服务器端软件之一。Apache 的诞生极富有戏剧性。当 NCSAWWW 服务器项目停顿后，那些使用 NCSAWWW 服务器的人们开始交换他们用于该服务器的补丁程序，他们也很快认识到成立管理这些补丁程序的论坛是必要的。就这样，诞生了 Apache Group，后来这个团体在 NCSA 的基础上创建了 Apache。它快速、可靠并且可通过简单的 API 扩展将 Perl/Python 等解释器编译到服务器中。Apache 取自“a patchy server”的读音，意思是充满补丁的服务器，因为它是自由软件，所以不断有人来为它开发新的功能、新的特性、修改原来的缺陷。Apache 的特点是简单、速度快、性能稳定，并可做代理服务器来使用。

原本 Apache 只用于小型或试验 Internet 网络，后来逐步扩充到各种 Unix 系统中，尤其对 Linux 的支持相当完美。Apache 有多种产品，可以支持 SSL 技术，支持多个虚拟主机。Apache 是以进程为基础的结构，进程要比线程消耗更多的系统开支，不太适合于多处理器环境，因此，在一个 Apache Web 站点扩容时，通常是增加服务器或扩充群集节点而不是增加处理器。

在过去的几年里，Apache 在百万级别的网站中其份额一直在稳步下降，至 2019 年 12 月份已降至 33.04%。虽然目前仍占有最大的份额，但其利润率正在迅速萎缩，其主导地位可能会受到 Nginx 在一年内的挑战。尽管包含这些损失，但运行 Apache 软件的面向 Web 的计算机的数量确实在继续增长。如果准备选择 Web 服务器，毫无疑问 Apache 是最佳选择之一。

Apacheweb 服务器软件拥有以下特性：

1. 支持各主要版本的 HTTP 通信协议，如广泛应用的 HTTP/1.1 及最新的 HTTP/2；
2. 拥有简单而强有力的基于文件的配置方法；
3. 支持通用网关接口；
4. 支持基于 IP 和基于域名的虚拟主机；
5. 支持多种方式的 HTTP 认证；
6. 集成代理服务器模块；
7. 支持实时监视服务器状态和定制服务器日志；
8. 支持服务器端包含指令 (SSI) 与安全 Socket 层 (SSL)；

9. 提供用户会话过程的跟踪；
10. 支持 FastCGI。

### 1.3.2 公共网关接口（CGI）的概念与技术

公共网关接口（Common Gateway Interface，CGI）是 Web 服务器运行时外部程序的规范，按 CGI 编写的程序可以扩展服务器功能，遵循 CGI 通信规则的其他软件叫 CGI 程序或 CGI 脚本。CGI 程序是在网站上实现动态页面的最简单和常用的方法，使外部程序与 Web 服务器之间交互成为可能。在交互过程中，CGI 程序处理从 Web 服务器获得的数据（如对表单数据处理，查询数据库等），然后将处理结果返回给 Web 服务器。

CGI 分为标准 CGI 和间接 CGI 两种。标准 CGI 使用命令行参数或环境变量表示服务器的详细请求，服务器与浏览器通信采用标准输入输出方式。间接 CGI 又称缓冲 CGI，在 CGI 程序和 CGI 接口之间插入一个缓冲程序，缓冲程序与 CGI 接口间用标准输入输出进行通信。

CGI 程序可以使用任何语言编写，只要遵循 CGI 通信规则，几乎所有服务器都支持 CGI。可以使用编译语言：C、C++ 等；脚本语言：Perl, Python, Bourne shell, Java 等。MapServer CGI 模块是用 C 语言编写的 CGI 程序。

CGI 由于其简单性而成为最受欢迎的工具。编辑基于文本的运行时配置文件，创建网页，然后将其设置为由 Web 服务器提供，在这个过程无需编程。

早期的 CGI 程序运行在独立的进程中，并对每个 Web 请求创建一个进程，这种方法非常容易实现，但效率很差，难以扩展。面对大量请求，进程的大量创建和消亡使操作系统性能大大下降，此外由于地址空间无法共享，也限制了资源重用。

### 1.3.3 FastCGI 介绍

FastCGI 是一个编程接口，它可以加速 Web 服务器调用应用程序的 Web 接口，即公共网关接口（common gateway interface，CGI）。FastCGI 是独立于语言的，它是由 Open Market 公司开发的，版权归其所有。FastCGI 标准规范可以免费使用，并作为一个开放的标准提供，它提供了一种跨平台和任何 Web 服务器使用的单一非专有方法。

FastCGI 在一个进程中管理多个 CGI 请求，为每个请求保存许多程序指令。在没有 FastCGI 的情况下，每个请求服务的用户实例都会导致 Web 服务器打开一个新进程，该进程获得控制权，执行服务，然后关闭。使用 FastCGI，一个进程的开销在当前处理的所有请求之间共享。与使用 FastCGI 的 CGI 不同，进程独立于 Web 服务器运行，将其隔离，从而提供更高的安全性。

FastCGI 是 Web 服务器的插件。它只需要对现有服务器应用程序（如 Perl 或 Tcl 脚本和 C 和 C++ 程序）进行少量更改，以获得性能好处。根据一个 FastCGI 实现者的说法，使用 FastCGI 来处理用户访问网站和使用特定应用程序的请求可以快 3 到 30 倍。

## 1.4 安装与配置 MapServer

本节将介绍 MapServer 的安装与配置，后面章节再说明应用它们开发 WebGIS 应用的方法。由于 MapServer 的功能由外部类支持，所以会介绍一下每一个类的主要的功能，及需要哪个类来建立基本的 MapServer 环境。

本书的测试是在 Debian 系统下进行的。第一次安装 MapServer 是具有挑战性的，尤其是对于 Linux 环境不熟悉的人来讲。在各个库之间有许多依赖性，编译安装时候的顺序是非常重要的。现在的 Linux 发行版已经能够足够好地处理好依赖关系，不必再考虑太多。但是了解底层的一些细节还是有很多的好处。

### 1.4.1 安装与配置 MapServer 的基本要求

MapServer 并非是完整的软件套件，想要发挥其完整或大部分功能，需要其他的组件进行配合。MapServer 是使用 C 语言编写的开源软件，它本身又依赖一些开源的或免费的库，有时候还依赖于某些类库的特定版本。尤其是编译安装 MapServer 时，更需要了解这些类库。这些库有些是必须的，有些则是可选的。

在 Debian / Ubuntu 中，相关的库会根据依赖关系自动安装。在 Debian 中使用 apt 工具查看依赖关系，如下：

```
② ~ apt depends cgi-mapserver
cgi-mapserver
  Depends: mapserver-bin
  Suggests: mapserver-doc
  Suggests: libmapscript-perl
  Suggests: python-mapscript
  Suggests: ruby-mapscript
② ~ apt depends mapserver-bin
mapserver-bin
  Depends: libc6 (>= 2.4)
  Depends: libcairo2 (>= 1.2.4)
  Depends: libcurl3-gnutls (>= 7.16.2)
  Depends: libfcgi0ldbl
  Depends: libfreetype6 (>= 2.2.1)
  Depends: libfribidi0 (>= 0.19.2)
  Depends: libgdal20 (>= 1.10.1-0~)
  Depends: libgeos-c1v5 (>= 3.4.2)
  Depends: libgif7 (>= 5.1)
  Depends: libglib2.0-0 (>= 2.12.0)
  Depends: libharfbuzz0b (>= 0.6.0)
  Depends: libjpeg62-turbo (>= 1.3.1)
  Depends: libmapserver2 (>= 7.0.0)
```

(下页继续)

(续上页)

```

Depends: libpng16-16 (>= 1.6.2-1)
Depends: libpq5
Depends: libproj13 (>= 4.8.0)
Depends: libprotobuf-c1 (>= 1.0.0)
Depends: librsvg2-2 (>= 2.14.4)
Depends: libxml2 (>= 2.6.27)
Depends: zlib1g (>= 1:1.1.4)
Suggests: cgi-mapserver
Suggests: mapserver-doc
Suggests: shapelib
Suggests: libmapscript-perl
Suggests: python-mapscript
Suggests: ruby-mapscript

```

- **Shapelib** 提供了读、写和更新“ESRI Shapefile”格式数据的能力，并能修改相应的属性文件；
- **FreeType**（或 FreeType 2）是一个渲染字体的库，能够渲染大部分的矢量和 bitmap 字体格式，设计思想是小、高效、高度定制化，并且在不损失性能和功能的前提下尽可能轻量；
- **Proj.4** 是一个地理投影库，提供多种投影的定义和使用接口；
- **GDAL/OGR**, **GDAL** (Geospatial Data Abstraction Library) 是一个在 X/MIT 许可协议下的开源栅格空间数据转换库。它利用抽象数据模型来表达所支持的各种文件格式。它还有一系列命令行工具来进行数据转换和处理。**OGR** 是 **GDAL** 项目的一个分支，功能与 **GDAL** 类似，只不过它提供对矢量数据的支持。有很多著名的 GIS 类产品都使用了 **GDAL/OGR** 库，包括 ESRI 的 ArcGIS, Google Earth 和跨平台的 GRASS GIS 系统。使用 **GDAL/OGR** 库来读入 raster (GDAL) and vector (OGR) 格式的数据。
- **AGG/GD Library**: MapServer 过去使用 GD graphics library 来渲染生成 PNG、JPEG、GIF、WEBP、XPM、BMP 格式的图像，现在则是通过 AGG。这些图像通常通过 MapServer CGI 在浏览器上浏览。为了生成相应格式的图像，通常需要相关类库的支持，如 JPEG library、PNG library。
- **Regex** 是一个正则表达式库，为 MapServer 提供正则表达式支持。
- **EPPL7**: 用来读取 EPPL7 数据库和旧版本的 Erdas LAN/GIS 文件。它是作为 MapServer 的内置库。

通过了解这些类库，能够减少要安装的功能，而简化安装过程。尽管只是安装 MapServer 的一个基本版本，但是它仍然有能力创建一个强大的应用。在对 MapServer 熟悉后，可以添加想要的其它功能。在 Debian 发行版中集成的软件包，会尽量包含大多数常用的功能。

### 1.4.2 Debian / Ubuntu 系统下安装 MapServer

Debian / Ubuntu 是本书推荐使用的 Linux 发布版本，其良好设计的包管理工具真是让人非常愉悦。安装 MapServer 只需要运行下面的安装命令（需要有管理员权限）：

在 Debian 10 (Debian Buster) 与在 Ubuntu 18.04 中：

```
apt install -y apache2 libapache2-mod-fcgid cgi-mapserver mapserver-bin
```

以上安装命令安装了 Apache、FastCGI 程序、MapServer CGI 程序，MapServer 工具等。在安装这些软件包时，如果缺少相应的依赖会自动安装。

这里安装了 EastCGI 程序 libapache2-mod-fcgid。mod\_fcgid 是 mod\_cgi 或 mod\_cgid 的高性能替代品，它启动足够数量的 cgi 程序实例来处理并发请求，并且这些程序仍在运行以处理进一步的传入请求。它被 PHP 开发人员所青睐，例如，作为运行 mod\_PHP\_in-process 的首选替代方案，提供非常相似的性能。

在安装完成后，可以在终端输入命令 mapserv-v 来查看安装是否成功，并进一步查看安装完成的 MapServer 所支持的功能（类库）。在 Debian 10 中，返回结果如下：

```
$ mapserv -v
MapServer version 7.0.4 OUTPUT=PNG OUTPUT=JPEG OUTPUT=KML SUPPORTS=PROJ
    SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=CAIRO SUPPORTS=SVG_SYMBOLS
    SUPPORTS=RSVG SUPPORTS=ICONV SUPPORTS=FRIBIDI SUPPORTS=WMS_SERVER
    SUPPORTS=WMS_CLIENT SUPPORTS=WFS_SERVER SUPPORTS=WFS_CLIENT
    SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVER SUPPORTS=FASTCGI
    SUPPORTS=THREADS SUPPORTS=GEOS INPUT=JPEG INPUT=POSTGIS
    INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE
```

### 1.4.3 Debian / Ubuntu 中配置 Apache

在本机安装完成 Apache 后，大部分常用的发行版会自动启动 Web 服务，可以访问 <http://localhost> 来查看是否成功。有时候需要输入 apachectl start 来启动它，如在 Windows 下的 Linux 系统中。

为了使 Apache 与 MapServer 能够合作运行，还需要进行一些配置。在过去的时候有许多工作需要修改 Apache 的配置文件，现在则可以通过命令来完成：

```
a2enmod authnz_fcg
a2enmod cgi
service apache2 restart
```

Apache 的配置可以通过文件 httpd.conf 进行。有兴趣的读者可以进一步了解。

在本书介绍的 MapServer 应用中，都会在 URL 中声明 Mapfile 的路径，这样做是为了教程中说明方便起见。在实际部署的生产环境中，可以用 Apache 别名（变量）指定 Mapfile 的位置（而不是向外部公开完整的 Mapfile 路径）。

在 Apache 的 httpd.conf 中设置别名（在本例中为 MY\_MAPFILE）：

```
SetEnv MapFile1 "/gdata/mfa8.map"
```

则在调用的时候可以隐藏掉具体的 Mapfile 路径，使用别名来替代：

```
http://webgis.cn/cgi-bin/mapserv?map=MapFile1&mode=...
```

配置 Apache 以使用 MapServer 不会太复杂。如果将 mapserv / mapserv.exe 程序放入 Web 服务器的默认 cgi-bin 位置，则可能没有太多事情要做才能使程序运行。

还需要其他设置才能访问 MapServer 生成的地图。运行 MapServer 时，它需要一个临时区域来创建地图，图例，比例尺等的图像。Web 也必须可以访问此临时文件夹，以便用户可以查看 MapServer 创建的图像。

有其他 Linux 中套件可以选择，但现在已经不再推荐这种方法。如果想尝试一下，可从 <http://maptools.org/fgs/> 获得一个名为“免费开源软件 GIS 套件 (FGS)”的项目，但是这个项目比较老。该项目为运行 FGS 应用程序所需的所有必需的二进制文件，库和其他支持文件提供了一种打包机制。初始版本为 Apache，PHP 和 MapServer 安装了最小环境。

#### 1.4.4 Windows 下面安装与配置 MapServer

在 Windows 下面安装 MapServer，也需要安装 Apache2，CGI，以及 MapServer 程序，这些也都有 Windows 下面的二进制软件包，但是这样一步一步安装是比较麻烦的，很容易出现问题。希望使用 Microsoft Windows 运行 MapServer 的 MapServer 应用程序设计人员可以下载标准 Windows 程序包，其中包括入门所需的一切。这些软件包将所有必需的 MapServer 程序压缩到一个文件中以供下载。

[MapTools.org](http://MapTools.org) 网站提供了最易于使用的软件包，强烈建议没有经验的、Windows 的 MapServer 应用程序开发人员应用。该程序包称为 MapServer For Windows（简称 MS4W）。

MS4W 由 DM Solutions Group（加拿大渥太华）合并而成。该公司参与 MapServer 及其相关技术的开发，实施和支持。他们提供了此 Windows 软件包，以帮助其用户快速启动并运行。

该软件包是一个单个 ZIP 文件，其中包含使用 MapServer 入门所需的所有内容。包括 Apache Web 服务器、GDAL、Proj4、MapServer CGI、MapScript 以及其他必要的库和实用工具。并且附带如 Geomoose、Openlayers 等。

其实安装 MS4W 非常简单，选择任一个盘符，解压 MS4W.zip 文件到根目录下（如 c:\ms4w\），可以将包移动到文件系统上的其他位置，但这需要更改各种 Web 服务器设置。然后执行 c:\ms4w\apache-install.bat，将安装并启动 Apache Web Server 服务，默认情

况下该服务的端口号为 80，如果与其他 Web 服务（如 IIS）有冲突，可以通过修改 c:\ms4w\Apache\conf\httpd.conf 文件修改其端口（如 8080）。安装好 MapServer 后，就可以通过（<http://localhost:8080/>）直接进入 MapServer 的主页面。

这意味着 Apache 已启动并正在运行。至此，已经做好了开发 MapServer 应用程序的准备。MS4W 软件包包含所需要的所有库，程序和配置文件。

MS4W 包含的主要内容：

- Apache version 2.4.37
- PHP version 5.6.40
- MapServer 7.0.7 CGI and MapScript
- GDAL 2.2.4 and bindings
- MapCache 1.6.1
- mapserver utilities
- gdal/ogr utilities
- proj.4 utilities
- shp2tile utility
- shapelib utilities
- shpdiff utility
- avce00 utilities
- spatialite utilities
- unixutils
- OWTChart 1.2.0
- H264 Streaming Module for Apache

### 1.4.5 编译安装 MapServer 的说明

开源项目都必须提供对源代码的访问。在 MapServer 的情况下，允许修改 MapServer 代码以满足自己的特定需求。即使不打算修改源代码，仍可能希望直接从源代码进行编译。一些 MapServer 用户发现他们需要编译自己的版本以包括（或排除）某些功能。例如，MapServer 的二进制发行版中通常包含各种数据格式。通常，格式要比给定应用程序所需的格式多，也可以删除对这些格式的支持。如果默认情况下无法使用某种格式，则通常可以在编译过程中添加该格式。

如果发行版自带的 MapServer 环境无法满足要求，还可以通过编译的方式来安装软件。编译安装需要对 Autoconf 及 Make 这两个 Build 工具比较熟悉，并应该大体了解如何生成 Apache 的配置文件。Autoconf 工具会通过自动配置每个 Build 来减少安装的冗长的过程。要注意的是，可能其它的安装已经安装了想要安装的类库。在这种情况下，只要版本没有问题，不必再重新编译安装，而只要使用正确的安装路径。还要注意的是，有的时候尽管版本没有问题，但是还是不能被 MapServer 使用。这时就需要重新安装适配的版本。

另外还要注意类库是必需的还是可选的。大多数类库是可选的，并且不需要在开始学习 MapServer 时介绍。它们提供了对商业数据库信息的访问，还提供了对 WMS（Web Mapping Service）的支持，及除了 PNG、JPEG 及 GIF 的其它的输出格式。

## 1.5 使用 MapServer 发布遥感影像

一般称由卫星或其他传感器设备获取的可视化产品为“影像”，而不是“图像”。“图像”概念的内涵要更广一点，“影像”是“图像”的一种。影像与形象共同组成了图像世界。“影像”主要指由摄影、电视、电影、数码成像、电脑绘画、网络影像所形成的复制性影像世界，当然也是包括遥感手段。

“图片”是指由图形、图像等构成的平面媒体。注意此处的定义，“图像”并非是“图片”的一种。图片首先是一种媒体（或者说介质），是实际存在的（可以为电子的）；另外，它还是平面的。“图片”是“图像”的一种表现形式。

三波段真彩色影像文件可能是最简单的空间数据，其带有颜色信息，在进行地图发布的时候不需要进行额外的定义。

### 1.5.1 在 Mapfile 中添加遥感影像

MapServer 通过 Mapfile（地图文件）来配置要发布的地图。Mapfile 是使用 MapServer 的核心，它将各种地图要素组织成具有层次关系的对象系统，声明生成地图的大小、格式，源数据的路径、名称、格式，以及种种决定页面显示的地图的外观和行为的控制选项。基于同一地理数据，制图应用使用不同的 Mapfile 能够呈现不同特点的地图，对应着用户的不同行为。

---

**注解：**要理解 MapServer 使用文本文件进行配置的设计，需要进一步了解 Linux 中“一切皆文件”的设计哲学，以及广泛应用使用文本文件作为软件参数配置背后的实践思想。

---

地图文件的定义由关键值对组成。含有空格的关键字值必须用引号包含起来，对所有的字符串括起来都是很好的习惯，单括号和双括号都可以。

下面使用遥感影像发布作为示例。用到的数据是已经做好的全球地形渲染图。这个数据是由 NASA 基于卫星观测数据合成的，于 2002 年发布。效果可以通过 <http://webgis.cn/834> 打开。这个 URL 会返回 PNG 文件，在浏览器中可以直接查看。

数字地图有一个同样重要的因素：源数据的质量。由于地图基于某种类型的源数据，因此一开始就必须具有高质量的信息。错误的数据会导致错误的地图和分析。

下面是效果（图 1.3）：

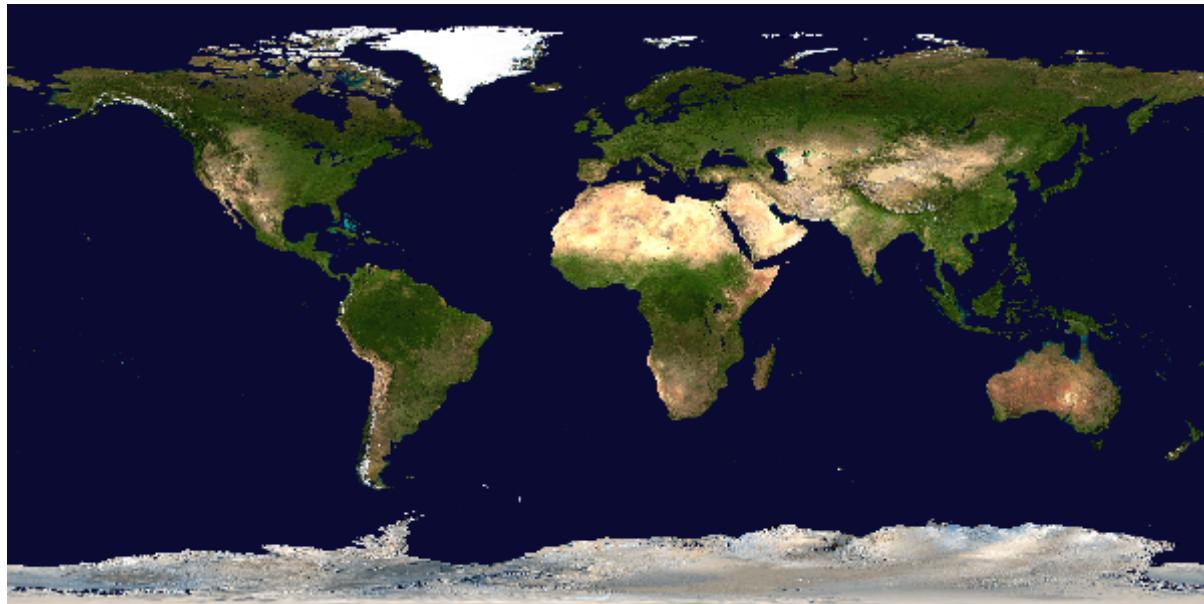


图 1.3：使用 MapServer 发布的遥感影像

具体的 Mapfile 可以通过 <http://webgis.cn/mf3t.map> 获取，内容如下，一共 13 行，非常简单。如果想自己进行实验或测试，可以使用任意的文本编辑器将下面的代码输入其中，并保存成以 .map 为后缀的纯文本文件，如 mf9a.map。文件的保存位置不重要，只要 MapServer 能够有权限读取即可。

```
1 MAP
2   EXTENT -180 -90 180 90
3   SIZE 600 300
4   IMAGECOLOR 255 255 255
5   IMAGETYPE PNG
6   SHAPEPATH "/gdata"
7   LAYER
8     NAME "topo"
9     DATA "land_shallow_topo_8192.tif"
10    STATUS OFF
11    TYPE RASTER
12  END
13 END
```

在 Mapfile 中，MAP 对象中定义了地图基本要素，包括大小和背景颜色，以及怎样在网络页面中显示地图。LAYER 对象负责画什么以及怎样画。

## MAP 对象说明

第 2 至 6 行建立了基本的地图图像参数。关键字 NAME 定义创建图像的基本名称。MapServer 每次调用时，它会通过连接系统时间（即自 1970 年 1 月 1 日 00:00:00 的秒数）和进程 ID 创建唯一的标识符；这种独特的标识符被附加到基本名称上，形成的文件名；后面会有文件扩展名，来说明文件的类型。在某些情况下，地图服务器的基本名称后插入另一个字符串，区分从地图图像本身的参考地图图像或图例图像。

EXTENT 声明初始地图范围，由西南和东北角的坐标确定。

关键字 SIZE 指定地图图像的像素尺寸（宽 × 高）。

IMAGECOLOR 设置的背景色为白色的图像（回想一下，颜色的选择在地图服务器通过指定三个整数的 RGB 分量值 0 和 255 之间，白色为 255, 255, 255）。IMAGECOLOR 的参数是三个十六进制的数字，分别代表了 R, G, B，即红色波段、绿色波段和蓝色波段的值，范围都是 0-255，例如想显示绿色 IMAGECOLOR 0 255 0。

图像类型（IMAGETYPE）设置为 PNG。在 Mapfile 中也可以使用 JPEG，或 GIF 图像。

SHAPEPATH 指定了数据所在的基础路径，是存放 Shapefile 或 GeoTiff 等数据的路径。在第 6 行中 /gdata 被指定为 MapServer 将寻找空间数据集的根目录。SHAPEPATH 下可以有其他子目录。

## LAYER 对象说明

从第 7 行开始了 LAYER（图层）对象，第 8 行至第 12 行是缩进的，表示都是在 LAYER 对象内部定义的。要注意缩进只是方便阅读，并不是必须的，Mapfile 并不使用缩进定义对象的层次。

LAYER 对象使用单个数据集并包含了一系列的元素，使用特定的投影（本书后面部分将包括投影的内容）可以把这些内容在特定的尺度上描绘出来。图层对象以关键字 LAYER 开始，以关键字 END 结束。

关键字 NAME 指定了图层的名字。这提供了图层与网络页面之间的联系。通过在 CGI 的表单中嵌入图层的名字，它可以交互式的指定显示哪个图层。

关键字 DATA 识别出该图层要显示的数据的基本名字（例如，忽略扩展名.shp）。与关键字 DATA 相关的值为之前 SHAPEPATH 的相对路径。Mapfile 中指定数据的路径也可以使用绝对路径，这个时候不必考虑 SHAPEPATH 的设置；如果是相对路径，则要注意 Mapfile 的存放位置与数据路径的关系。

图层的 STATUS 可以有以下三个值：OFF，ON 或者 DEFAULT。如果一个层的 STATUS 值为 ON，缺省情况下该层将会被显示；如果是 OFF，缺省情况下该层不会显示。如果 STATUS 的值为 OFF 或者 ON，可以通过 WEB 表单适当的响应切换到相反的值；然而如果 STATUS 的值为 DEFAULT，将会永久的显示，不会被关闭。

每一层都有相关的类型，类型由关键字 TYPE 指定。层的类型决定了 MapServer 与该层空间数据之间的相互作用如何。与关键字 TYPE 有关的值解释如 表 1.5：

表 1.5: 关键字 TYPE 的取值与说明

类型	说明
POINT	点图层用来显示单个的空间数据点。例如，城市的位置以及其它感兴趣的点。点图层中的每个要素用单个的符号进行显示，符号有特定的大小和颜色。
LINE	线图层用来显示一系列相连的点。例如，道路或者河流。连续的点用一条线联系起来，线有着特定的尺寸和颜色。
POLYGON	多边形图层用来显示一些列的点，这些点组成闭合的图形。与线不同的是，多边形第一个点和最后一个点必须是同一点。由于多边形包含了一定的区域，它有内部填充颜色。
RASTER	栅格图层渲染的是有地理参考的地图，并能够给地图制造商提供嵌入矢量数据的能力，嵌入的矢量数据含有真实世界中的地理范围。这可能是航空或者卫星影像，或者是用不同颜色表示的高程图。事实上，任何地理学上的分布信息都可以成标记地图并可以渲染为栅格图层。
QUERY	查询层把地图影像上的鼠标点击行为和指定的数据集联系起来。查询层不是绘制的，但是其属性可以被查询。例如，它可以减少绘制地图细节的工作量，但是基于空间数据库的属性仍然能够向用户提供查询数据的能力。

对于 TYPE，可以设置：

- ellipse: x 和 y 方向的半径值定义椭圆。
- hatch: 整个（多边形）形状中生成阴影线。
- pixmap: 用户提供的图像将用作符号。
- svg: 可缩放矢量图形 (SVG) 符号。需要 libsvg / libsvg-cairo 库（或者 librsvg 库）。
- truetype: 要在 MAP FONTSET 中定义的 TrueType 字体。
- vector: 矢量绘图用于定义符号的形状。

## 1.5.2 使用命令行工具 shp2img 生成结果

MapServer 命令 shp2img 获取 Mapfile 中的设置并生成地图图像。这里显示了命令的语法以及如何使用 global.map 文件运行该命令的示例。

配置好 Mapfile 后，可以通过 Web 来查看结果；但也可以不用。可以使用命令 shp2img 将其渲染成为图片，很多时候都可以使用这个最基本命令来尝试生成结果。这个命令可以作为检查 Mapfile 是否正确的一个基本工具。根据 Mapfile 创建地图图像默认情况下输出为 PNG，但可以指定 Mapfile 中定义的其他输出格式。只需提供 Mapfile 的路径和输出图像的名称，然后运

行的结果会生成图像文件。如果 **Mapfile** 有错误，程序会给出提示信息。下面命令会在目录下生成一个名叫 `out.png` 的图片文件。

```
shp2img -m mf3t.map -o out.png
```

虽然 **Mapfile** 是任何 **MapServer** 应用程序的中心部分，但是生成的地图图像通常是最终用户所追求的。除了地图，比例尺，图例和参考地图也可以是 **MapServer** 应用程序输出结果的一部分。这些以与地图图像相似的方式处理。

## 1.6 Mapfile 介绍

当请求到达 **MapServer** 应用程序时，该请求必须指定要使用的 **Mapfile**，然后 **MapServer** 根据 **Mapfile** 中的设置创建地图。这使 **Mapfile** 成为任何 **MapServer** 应用程序的关键部分。本书中对 **Mapfile** 语法与关键词进行必要的阐述，但并不能全面说明使用方法。**MapServer** 网站上的 **Mapfile** 参考文档是了解详细功能的必要参考。

**MapServer** 程序需要知道要绘制哪些地图图层，如何绘制它们以及源数据位于何处。**Mapfile** 是一个文本配置文件，其中列出了用于绘制地图以及与地图交互的设置。它包括有关要绘制哪些数据层，地图的地理重点在哪里，正在使用哪种投影系统以及要使用哪种输出图像格式的信息，并且它设置了图例和比例尺的绘制方式。

可能看起来静态配置文件的功能会有限，但是 **MapServer** 的设计使得使用 **Mapfile** 可以生成十分强大的应用程序。

### 1.6.1 Mapfile 的组成部分与配置步骤

配置一个完整的 **Mapfile** 文件需要 **GIS** 数据源，字体文件，符号文件。**GIS** 数据源可以分布在不同的计算机上；字体文件用来定义输出地图中使用的字体类型。地图中使用的点、线、面符号可以在 **Mapfile** 文件 **Symbol** 对象中定义；不过为了实现符号的重用，通常在 **Mapfile** 外部定义独立的符号文件，无须在不同的 **Mapfile** 文件中定义相同的符号。

**MapServer** 可以使用大量数据源来创建地图，本身就支持了最常见的数据格式。可选的数据访问附加组件可打开对多种矢量和栅格格式（**GDAL** 和 **OGR** 库支持的格式）的访问。这些可以是 **GIS** 数据文件，数据库连接，甚至可以使用 **OGR** 的虚拟空间数据格式功能（如以逗号分隔的 **CSV** 文本文件）。**MapServer** 还可以使用 **OGC Web** 规范通过 Internet 访问和共享数据。可以从也使用 **OGC** 规范的远程服务器请求地图层。有关 **OGC** 和 **OGC Web** 服务的目的和目标的更多信息，请参见 [节 10](#)。

在规划好 **WebGIS** 网站之后，配置 **Mapfile** 文件步骤如下：

1. 准备 **Mapfile** 文件需要的资源，包括 **GIS** 数据，字体文件，符号文件。

2. 按照 WebGIS 网站的需求，遵循 Mapfile 的语法，使用文本编辑软件或其他工具来编写或生成 Mapfile。
3. 使用调试软件或辅助工具测试、查看 Mapfile 配置是否正确、合适。

---

**注解：**从目前来看，软件生成的 Mapfile 只能作为辅助，大部分情况需要对生成的结果进行修改。

---

### 1.6.2 Mapfile 概述

Mapfile 是纯文本文件，以 .map 为扩展名，可以使用各种文字编辑器编写。包含嵌入的空格的字符串必须加引号，单引号或双引号都可以的，但必须成对使用。

注释可以使用 # 符号声明单行注释，在地图文件中进行文字说明。MapServer 将忽略在同一行上书写的符号之后的所有内容。

Mapfile 的语法是相对简单的。Mapfile 的关键字包括对象名（图中的所有对象）、对象属性（keyword），对象结束标记 END。一般来说，MapServer 的关键字和参数值不区分大小写。然而在本书中，关键字是始终是大写，参数值是小写。这种做法是为阅读方便起见，并非是必需的。

---

**注解：**应该注意与空间数据集交互，属性值仍然是区分大小写。例如，底层数据库中的属性名称可能是大小写敏感的，因此表达式包含属性的引用也将是区分大小写。

---

缩进行是可选的，但建议使用，以使地图文件更具可读性。通过缩进对象的开头和不缩进对象的末尾，以分层方式处理此结构。

### 1.6.3 Mapfile 的地图对象

Mapfile 是使用分层或嵌套对象结构来定义地图，每个对象都有不同类型的设置。Mapfile 由对象（Object）或节（Section）组成。每个对象都由一个关键字开头，使用 END 表示结束。这些对象包括比例尺、图例、地图配色、地图名称以及地图图层等等。对象可以在其中嵌套其他子对象（如果适用），例如，图层定义将包括定义如何绘制图层或如何为图层创建标注的对象。

Mapfile 定义了构成地图的各要素以及它们之间的关系，并告诉 MapServer 从哪里加载数据并如何表现这些数据。Mapfile 文件将各种地图要素组织成具有层次关系的对象系统。数据来源，使用的数据格式，用户交互和对 OGC 协议的支持也在 Mapfile 中定义 [图 1.4](#)。

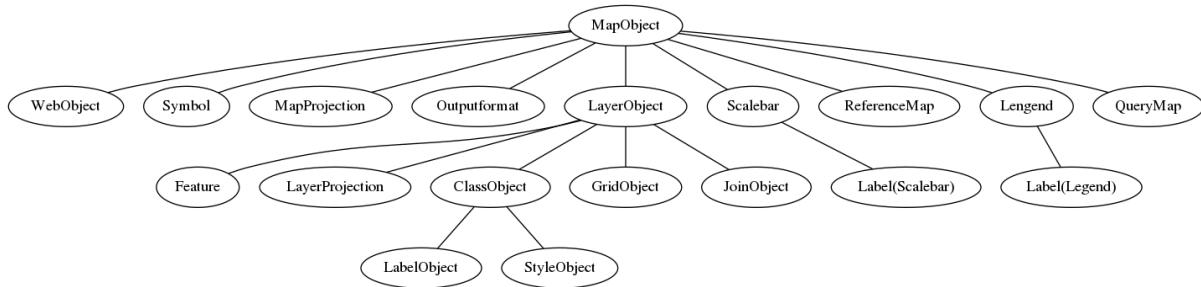


图 1.4: Mapfile 中的对象及其层次关系

## Map 对象

Mapfile 以关键字 MAP 作为地图对象的开始，以关键字 END 作为地图对象定义的结束标志，整个 Mapfile 就代表一个地图对象。在 MAP 和 END 关键之间是以关键字/值对的方式描述地图对象的属性。

MAP 对象有许多属性，例如 节 1.5 介绍过的地图图片格式 (IMAGETYPE)、地图大小 (SIZE)、地图数据的路径 (SHAPEPATH)、地图范围 (EXTENT) 以及地图使用的符号集 (SYMBOLSET) 和字体集 (FONTSET) 等。在旧版本的 MapServer 中，默认有最多 100 个图层的限制；但在新版本中不必考虑这个问题了。

## 图层对象

每个图层也有一些常用的属性，如图层名称 (NAME)、数据集 (DATA)、类型 (TYPE) 以及是否显示 (STATUS) 等。

每个图层有一个或多个 Class 对象，用来实现对图层中要素进行分类表现。通常图层对象中还有一个关键字 CLASSITEM (属性)，用来指定一个属性字段，以便可以根据该字段的值将图层分成若干个类别 (Class) 或主题 (Theme)。如果在图层对象中指定了关键字 CLASSITEM，则每 Class 对象中就需要一个关键字 EXPRESSION 来指示该类别代表的属性值。Class 对象通常有一个 Name 属性。

每个 Class 对象都用一个或多个 Style 对象来定义该类要素的显示样式，如符号类型 (SYMBOL)、填充的颜色 (COLOR 或 OUTLINECOLOR) 和符号的大小 (SIZE) 等。

## 其他对象

Mapfile 中大部分对象或与地图要素相对应，或是 GIS 中应用较普遍的概念；如 Legend，Scalebar，Layer 等。这些对象的作用是较为明显，易懂的。在这里解释四个特别对象的意义：Web Object，Outputformat Object，Query Object，Reference Map Object。

- Web Object 定义如何处理 WEB 接口。如：属性 IMAGEPATH 定义存储临时文件和图像的路径，TEMPLATE 定义使用的交互界面文件。

- **Outputformate Object** 定义输出图像的格式。MapServer 将 GIS 数据格式文件（如 shapefile）转换成 MIME（Multipurpose Internet Mail Extensions）协议支持的图像格式，如 gif，png，jpeg 等。
- **QueryMap Object** 定义了查询结果的生成机制。如：属性 **STYLE**（不是对象）设置选中的特征如何显示。
- **Reference Map Object** 定义参照地图如何创立。MapServer 支持三种参照地图类型。参照地图最普遍的用法是做为 map 的观察窗口。

在 Mapfile 中，图层（Layer）是数据和样式的综合体；而其中的数据是空间数据和属性数据的联合；样式是通过 CLASS 和 STYLE 对象来实现的。

#### 1.6.4 使用 Mapfile 的注意事项

Mapfile 由层次结构的对象组成。在层次结构的顶部是地图对象（即 Mapfile 的本身）。地图对象包含简单的和结构化的项目。简单的项目由关键字值对组成，结构性项目包含其他项目，每一个都可以是简单的或结构性的。

Mapfile 行中的每一个都包含了一些简单的 Mapfile 的对象。这些关键字指定一个值，使地图作为一个整体才有意义。例如，关键字名称首先设置地图的名称。此关键字在地图的层次上使用来指定的字符串，用于标识所有在 Mapfile 的生成输出文件 MapServer 的过程中生产的地图。但是要注意，在 MapServer 中可以在不同的对象中使用相同名字的关键字，它的作用取决于在哪里使用。在关键字 EXTENT 设置了整个地图的范围，所以必须被定义在地图层次；而 NAME 则可用于在较低的层次，如 LAYER 对象中。

---

**注解：**如果刚开始使用 MapServer，在 Mapfile 的不同级别中使用相同的关键字（如 NAME 或 COLOR）可能会造成混淆。当熟悉了 mapfile 的概念，这将不存在问题；事实上，用户会很感激开发者选择在不同的层面使用相同的关键字定义相对相似的概念。

---

## 1.7 Mapfile 的辅助工具

Mapfile 是一种纯文本文件。有一些工具可以作为辅助来更好地编写，或检查其中的问题。

### 1.7.1 Mapfile 的编辑器

Mapfile 可以使用常用的文本编辑器进行编辑。但最好选择能显示行号的编辑器。本书在介绍过程中说明代码时会说明行数，从而方便对照。目前常用的代码编辑器都支持行号显示，如 VSCode, Sublime 等。由于应用 Mapfile 的使用范围较小，所以没有太多的代码高亮工具。在 MapServer 官网上，有 VIM 与 Emacs 的代码高亮说明。较新的编辑器中则有 Sublime 可以支持高亮显示。

### 1.7.2 shp2img 命令行工具

shp2img 是 MapServer 自带的命令行工具。shp2img 实用程序还有其他几个选项，允许选择不同的参数，以覆盖在地图文件中定义的内容。在 [节 1.5](#) 使用了 -m 声明地图文件，-o 声明结果的名称。

另外，-i 参数可以更改输出的图层格式，如 Jpeg , PNG; -l 参数用来控制输出的图层；-e 参数来修改命令，并指定地图文件中的范围；-s 参数则用来声明输出结果的大小。还可以添加调试的信息，通过 -all\_debug n 设置地图与所有图层的调试级别；通过 -map\_debug n 设置地图的调试级别；通过 -layer\_debug layer\_name n 设置某个图层的调试级别。

shp2img 配合使用各种参数，不需要更改地图文件就可以得到不同场景下的结果，在辅助修改 Mapfile 中非常方便。

### 1.7.3 mappyfile 工具

mappyfile 是用 Python 写的解析 Mapfile 的工具，可以使用 Python 编程处理 Mapfile，也有在终端可以直接运行的命令行工具。这个工具是使用 Python 来处理 Mapfile，对于实际工作中是非常有用的。mappyfile 开发路线图包括允许对较新版本的 MapServer 进行验证的计划，例如验证 MapServer 7.0 或 7.2 的 Mapfile。不太可能考虑 7.0 之前的版本。

在 mappyfile 的设计中将 Mapfile 视为 DSL (a domain specific language)。在 MapServer 的发展与改进中，Mapfile 作为 DSL 的概念已经实现了几次。由 AllanDoyle 编写的 Python Mapfile builder 借用了 XML 技术；GISLite 的设计中使用 XLSX 文件进行地图样式定义也是如此。

与 shp2img 命令行工具不同，这个工具并不尝试生成地图渲染结果，只是对 Mapfile 本身进行语法分析，所以运行的时候甚至不必有 MapServer 运行环境支持。mappyfile 将 Mapfile 作为输入并将其解析为 Abstract syntax tree (AST) 使用 lark 一个 python 解析库。然后，mappyfile 可以将 AST 转换为字典结构，其中包含字典的键和值，以及列表。此结构可以直接编辑，或者可以通过

过解析更多的 Mapfile 文本并插入字典结构来添加新对象。mappyfile 还包括格式化工具，用于将字典结构导出为 Mapfile，并使用关键字格式和缩进。

mappyfile 只处理文本，所以不能像使用 MapScript 那样通过层检索功能或连接到数据库。mappyfile 的方法是构建一个 Mapfile，然后使用 mapserv 命令来处理这些需求。mappyfile 是使用 MapScript 的一种可能的替代方法，可以实现以下功能：

1. 从同一源 Mapfile 轻松生成开发、切分和生成 Mapfile；
2. 从单个 Mapfile 为不同的数据集创建 Mapfile；
3. 从 Python 中创建、操作和测试 Mapfile。

mappyfile 命令行包含两个应用程序，validate 命令与 format 命令。命令行界面（CLI）允许将 mappyfile 轻松集成到 Travis 和 Appveyor 等持续集成（CI）平台中，并自动验证和格式化 Mapfile。

validate 命令可用于检查 Mapfile 中使用的关键词与值是否有效，它的参数与 validate() 函数相同。

format 命令可用于对 Mapfile 应用一致的格式、缩进的空格数，以及清除注释。它具有与 save() 函数相同的参数。

使用默认格式设置将 valid.map 格式化为新的 valid\_formatted.map 文件，使用以下语法：

```
mappyfile format valid.map valid_formatted.map
```

format 命令还有许多选项可以控制结果输出，可以通过 help 命令查看，如下所示。

```
mappyfile format --help
```

在本书的配套网站中，也是使用这个工具将 Mapfile 进行最后的处理，进行验证与格式化，并最终发布出来。

## 第2章 使用 MapServer 发布地图（1）

上一章中介绍了 MapServer 地图制图涉及的基本概念，可以帮助理解 MapServer 的基本运行与使用，没对 MapServer 的语法和用法进行广泛的讨论。这一章会通过 MapServer 中发布栅格数据与遥感影像，以及如何使用分类，图例等制图功能，进一步介绍 MapServer 的功能。

### 2.1 使用 MapServer 发布 DEM 栅格数据

遥感影像是一种栅格数据，在 GIS 应用中，栅格数据的概念更广一些。除了支持影像数据，MapServer 也能显示栅格数据。

#### 2.1.1 栅格数据介绍

栅格数据就是将空间分割成有规律的网格，每一个网格称为一个单元，并在各单元上赋予相应的属性值来表示实体的一种数据形式。每一个单元（像素）的位置由它的行列号定义，所表示的实体在栅格行列位置隐含位置中，数据组织中的每个数据表示地物或现象的非几何属性或指向其属性的。栅格结构是用有限的网格逼近某个图形，因此用栅格数据表示的地表是不连续的，是近似离散的数据。对于固定范围的地区，越小的栅格则会越精确，相应地也会导致数据量呈几何增幅地变大。

从文件格式与概念上来说，栅格数据包括影像数据；但是影像数据大多指的是遥感影像、航空相片。通过使用 GDAL 库，MapServer 支持多种栅格格式的输入输出。然而 4.x 版本以下的栅格输入只限定于单图层，灰度或索引图。当前 MapServer 版本支持 RGB 和多光谱（多图层）影像。

#### 2.1.2 直接发布栅格数据

在本示例中，数据是全球 DEM 数据。这个数据是由 NASA 提供的全球 1KM 分辨率的 DEM 数据，文件格式是 Tiff 文件，是单波段文件，不是三波段，但这个单波段文件也并非是灰度文件。

图片根据存储的不同，可以分为 RGB 彩色文件，灰度文件，索引文件等。

看一下示例 <http://webgis.cn/413>（图 2.1）。

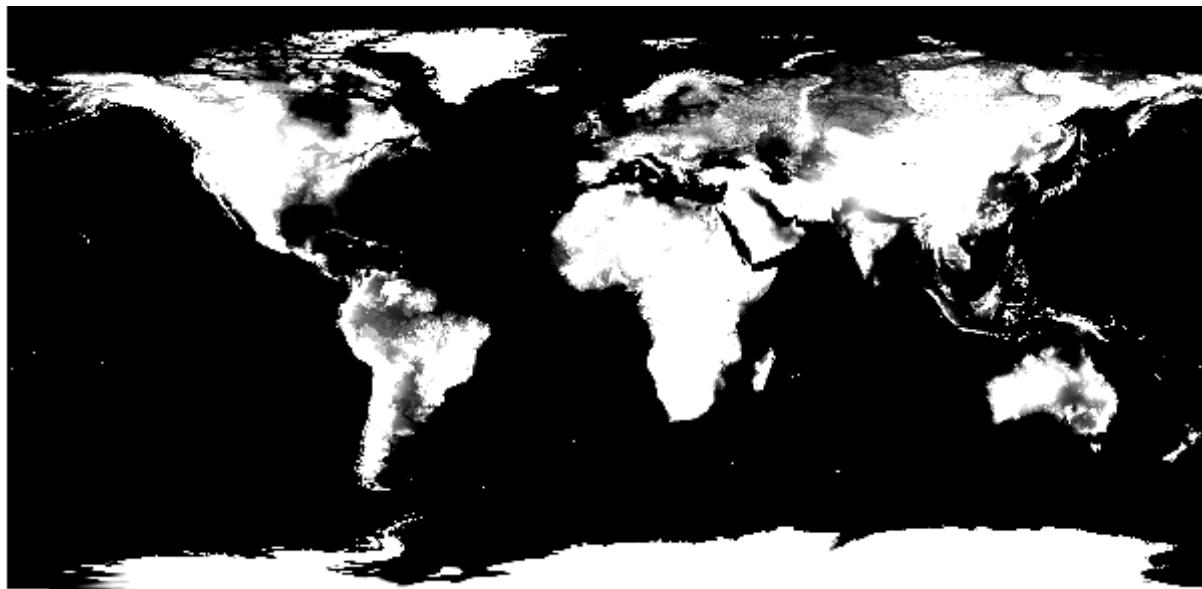


图 2.1：原始 DEM 数据直接制图

其 Mapfile 为 <http://webgis.cn/mfn8.map>，内容如下：

```
1 MAP
2     EXTENT -180 -90 180 90
3     SIZE 600 300
4     SHAPEPATH "/gdata"
5     PROJECTION
6         "init=epsg:3857"
7     END
8     LAYER
9         PROJECTION
10            "init=epsg:3857"
11        END
12        NAME "nasa-dem"
13        DATA "raster/nasa-dem.tif"
14        STATUS OFF
15        TYPE RASTER
16    END
17 END
```

看起来 Mapfile 与第一个例子差不多，只不过使用的数据源进行了修改。上面将 DEM 数据进行直接展示，没有什么处理，问题在于上面的结果除了海洋部分，陆地部分显示的基本都是白色，没有区分开不同的高度。在这幅图上想说明不同海拔高低的地理知识是比较难的。

原因在于 MapServer 在制图时，生成的结果是 256 色的，并且默认数据源也是 256 色的。DEM 数据是地球的高程，大部分地区是高于 255m 的，对于这些部分，MapServer 会以值 255 来展示。对于 8 位的图像，可用的值的数目为  $2^8 = 256$  个，值以 0 开始，能表示的最大值

为 255。

数字高程模型 (Digital Elevation Model), 简称 DEM , 是通过有限的地形高程数据实现对地面地形的数字化模拟 (即地形表面形态的数字化表达), 它是用一组有序数值阵列形式表示地面高程的一种实体地面模型, 是数字地形模型 (Digital Terrain Model, 简称 DTM ) 的一个分支, 其它各种地形特征值均可由此派生。

一般认为, DTM 是描述包括高程在内的各种地貌因子, 如坡度、坡向、坡度变化率等因子在内的线性和非线性组合的空间分布, 其中 DEM 是零阶单纯的单项数字地貌模型, 其他如坡度、坡向及坡度变化率等地貌特性可在 DEM 的基础上派生。

目前常用的全球公开 DEM 数据, 还有 SRTM DEM , 与 ASTER DEM 。

### 2.1.3 使用自动拉伸对数据进行预处理

对于数据值超出阈值范围问题, 要想尽可能使用其信息, 最常用的方法是“归一化”, 将数据的值域按比例 (或其他算法) 缩放到可用的值域。MapServer 可以自动对原始数据的值的范围进行拉伸, 拉伸到 0-255 之间。

示例如: <http://webgis.cn/561> , (图 2.2)。



图 2.2: 自动拉伸的 DEM 数据

其 Mapfile 为 <http://webgis.cn/mfq8.map> , 内容如下:

```

1 MAP
2   EXTENT -180 -90 180 90
3   SIZE 600 300
4   SHAPEPATH "/gdata"

```

(下页继续)

(续上页)

```

5   PROJECTION
6     "init=epsg:3857"
7   END
8
9   LAYER
10    PROJECTION
11      "init=epsg:3857"
12    END
13    NAME "nasa-dem"
14    DATA "raster/nasa-dem.tif"
15    STATUS OFF
16    TYPE RASTER
17    PROCESSING "SCALE=AUTO"
18  END
END

```

现在看一下效果，整体是暗色调的。黑色的地方原本为海洋，现在看起来面积变得更大。白色是海拔高的地方，现在看起来一处在青藏高原，另一处是南美洲的安第斯山脉。这些大概是从图中可以读出的信息。

自 MapServer 4.4 起，已添加对非 8bit 栅格输入分类的支持，即值在 0-255 范围之外的栅格数据。缺省情况下认为栅格数据值域的范围为 0-255，但有一些警告和选项来提供显式控制。

上面进行自动拉伸需要用到 PROCESSING 指令。PROCESSING 指令可用于覆盖要预分类的值的范围，或覆盖要分类的该范围内的值数（也称为 **bucket**）。SCALE=min, max, PROCESSING 指令控制范围。SCALE\_BUCKETS PROCESSING 指令控制 bucket 数量。在某些情况下，通过选择有限的输入值范围和减少的缩放值（bucket），渲染速度可以大大加快。

自动拉伸是将数据（图像）处理交给 MapServer。MapServer 主要用于地图制图与发布，并不擅长进行数据处理。一般而言未经颜色处理的遥感影像整体像素值偏低，有时使用自动拉伸可以快速进行效果查看。在大多数情况下，建议在数据发布之前首先将数据处理好。

#### 2.1.4 根据属性值对栅格数据进行分类展示

从 图 2.2 中能够看到不同地势的高低差异，但是并不算好的展示方式。可以根据像元的范围进行分类，每种类别使用一种颜色以获得更好的结果。

在 MapServer 中对栅格数据进行分类是通过对所有预期的输入值进行预分类，并使用分类结果表在渲染时查找每个像素来完成的。这是因为根据一系列类定义来评估像素值对于典型渲染图像中数十万像素来说成本提高。

对于简单的 8 位输入，只需对 256 个输入值进行预分类。但对于非 8bit 输入，需要对更多的值进行分类。对于 16 位整数输入，所有 65536 个可能的输入值都是预先分类的。对于浮点和其他输入数据类型，最多可以根据输入值的最大预期范围对 65536 个值进行预分类。

示例如：<http://webgis.cn/713>，如图 2.3。

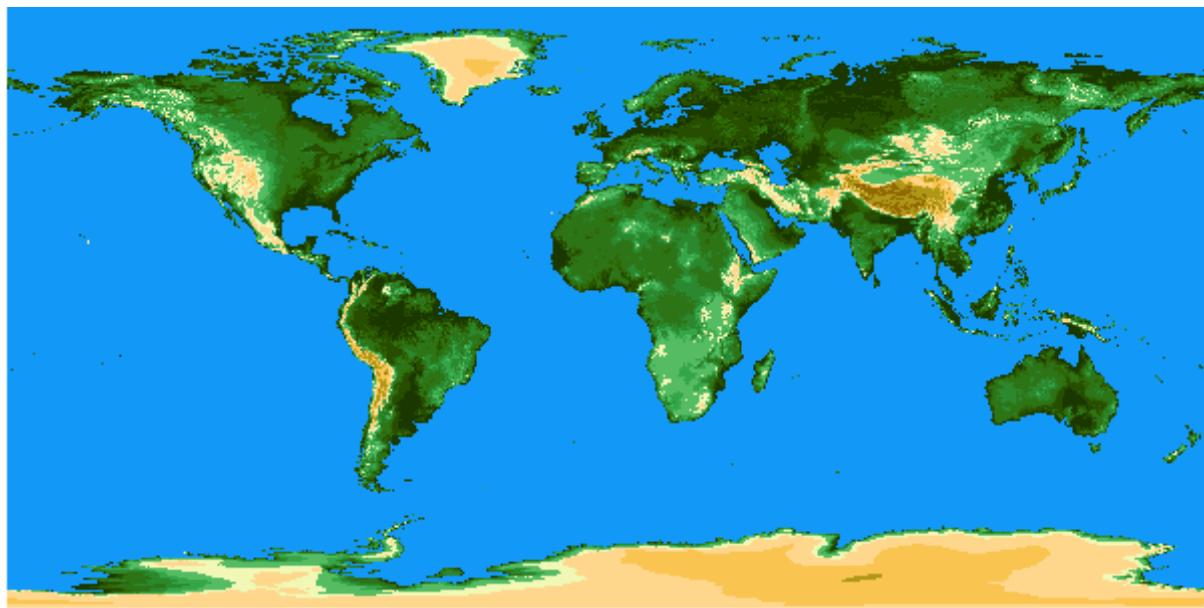


图 2.3：使用颜色分级的 DEM 地图

用到的 Mapfile 为 <http://webgis.cn/mfu7.map>，主要的代码为：

```

17      CLASS
18          NAME "g0"
19          EXPRESSION ( [pixel] <= 0 )
20          STYLE
21              COLOR 18 153 248
22          END
23      END
24      CLASS
25          NAME "r01"
26          EXPRESSION ( ( [pixel] > 0 ) AND ( [pixel] < 100 ) )
27          STYLE
28              COLOR 26 55 0
29          END
30      END

```

分类使用关键词 CLASS，对于栅格数据，其值固定为 [pixel]，表示像元值。然后使用 EXPRESSION 表达式来说明每一个类别中像元值所在的范围。

必须得注意，分成几类，各个类别的范围，以及每种类别的颜色，都会产生不同的结果，从而也会导致对数据理解的偏差。

## 参数说明

看一下这个 MapFile 文件中的新参数。

### IMAGETYPE

PNG 是一种 24 位或真颜色版本的 PNG 格式。也可以试着将这个值改为 PNG8。无论使用何种格式注意生成图像的时间。当在真彩色和索引色之间选择时，需要考虑生成图像的时间。

在新添加的 LAYER 对象中，DATA 参数指向一幅 GeoTIFF 图像。正如支持多格式矢量数据，MapServer 也支持多种栅格文件格式。这种支持是通过使用 GDAL 库来获取的。

### PROCESSING

LAYER 对象参数是在 MapServer4.x 中添加的。PROCESSING "BANDS=1,2,3" 关键字有很多值，但是这里将用它来指定多光谱数据中哪些波段显示。这个字符串类型的值会传给 GDAL 库。

### OFFSITE

这个参数告诉 MapServer 哪个像素值需要作为背景（或无效值）去渲染，这个问题也是在栅格数据制中经常会碰到的。有一些栅格数据中会有 NODATA 值的定义，但也有一些会用某个特定值，如 -1, -99, -9999 等，这些特定值都是认为在这个栅格数据中不会发生的值。可以用图像处理或图像操作程序来获取这个像素值（如开源图像处理软件 GIMP）。

## 2.1.5 在 MapServer 中控制地图范围

在 Mapfile 中，需要设置好正确的范围才能显示数据。前面的例子中显示的是全球范围，设置的值为：EXTENT -180 -90 180 90。比如中国范围（ToDo: 修改为非洲）的数据，可以通过链接 <http://webgis.cn/324> 查看，效果如 图 2.4：

这个 URL 请求与前面类似，不再说明。使用的 Mapfile 为 <http://webgis.cn/mfs1.map>，主要是范围的设置：

3

```
EXTENT 73 3 136 54
```

在上面展示的是地图能够展示的范围。另外，还可以使用参数来指定返回的范围大小（如图 2.5），可打开 <http://webgis.cn/392> 查看。

URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[\?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mf3t.map
2 layer=topo
3 mode=map
```

(下页继续)

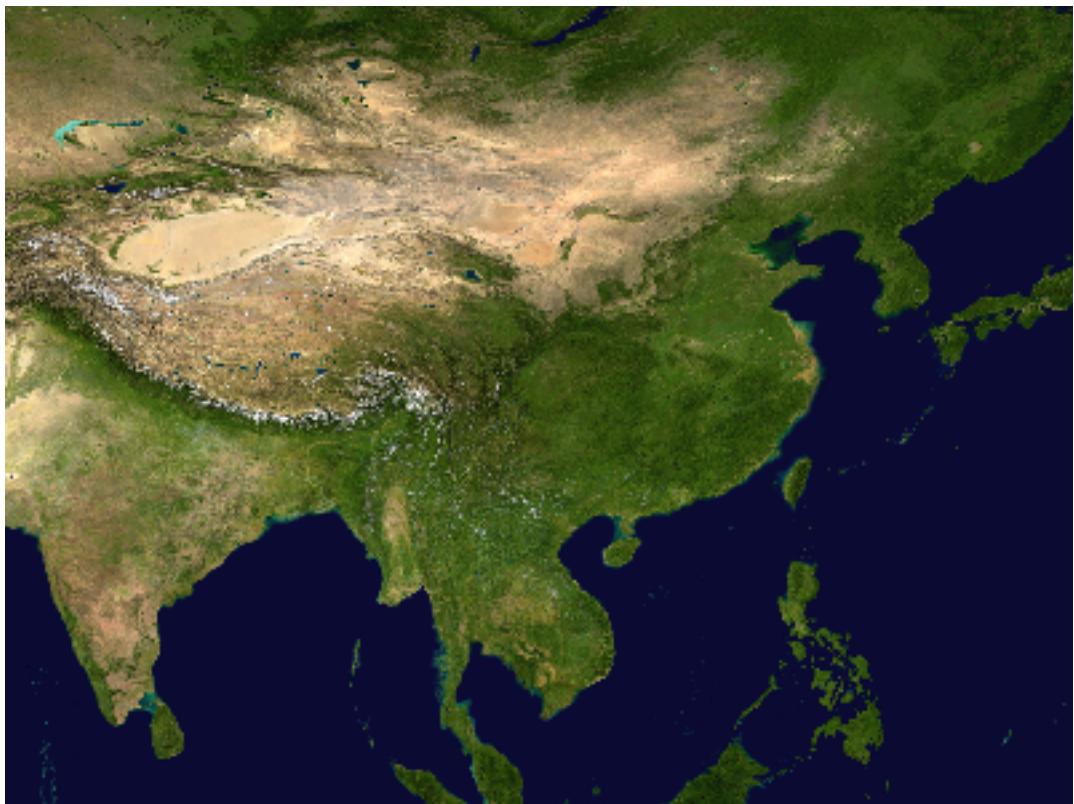


图 2.4：显示中国范围地图

(续上页)

```
4 imgext=73+3+136+54
5 imgxy=300.0+150.0
```

使用参数来请求控制地图范围一般不会手工构建，在后面的章节中，会使用动态交互的方式，将相关的参数提交到 MapServer CGI 程序，实现这样的功能。所以在这里就不介绍具体的参数意义了。

### 2.1.6 MapServer 查看的不同模式

注意上面的 mode=map，这是 MapServer 支持的查看的模式。MapServer 的 mode 的 CGI 变量可以使用 map 以外的其他值。除了 map 模式之外，还有 browse 模式。例如，如果使用 map=browse，MapServer 的将在服务器上的临时目录中生成图片。使用 browse 模式需要指定 template 参数。一般需要先定义一个模板文件，没有模板文件的情况下 browse 模式无法正常工作，这个在后面会说明（[节 4.2](#)）。而在 MapServer 6.x 中，template 参数允许使用 OpenLayers 关键词，可以使用 OpenLayers 工具来查看 MapServer 的结果。

打开网址 <http://webgis.cn/893>，会使用 OpenLayers 的缺省配置来查看地图。注意，要调用 MapServer 网站的 OpenLayers 类库，所以初次会比较慢一点。

URL 解析如下：

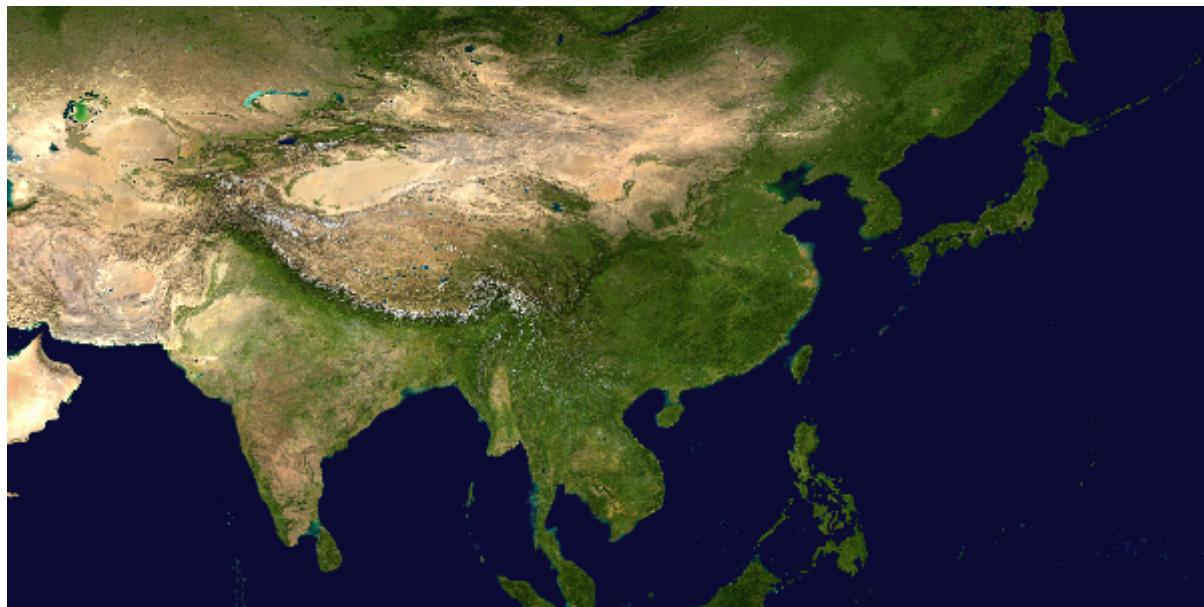


图 2.5：使用参数请求西伯利亚地区地图

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]  
0 http://webgis.cn/cgi-bin/mapserv  
1 map=/owg/mfa1.map  
2 layer=states  
3 mode=browse  
4 template=OpenLayers
```

关于 OpenLayers 模式，可查看：

<https://mapserver.org/cgi/openlayers.html>

可以尝试修改 Mapfile 中的关键字的值并查看结果。这些实验会帮助了解这些关键字。

## 2.2 通过 Web 查看地图

到目前为止，虽然通过浏览器来访问和查看地图，但是并没有使用 Web 页面。

由于各种原因，构建和使用 Web 地图与使用桌面 GIS 不同：

1. 在 Web 地图中，查看到的任何信息都必须“通过网络”从服务器发送到浏览器，从而引入延迟。
2. 在 Web 地图中，用户可能正在从多个不同的服务器中提取信息。因此，Map 性能受到正在使用的所有服务器的可用性和速度的限制。
3. 在 Web 地图中，性能可能会受到同时使用服务器的其他用户的影响。

4. 在 Web 地图中，受限于客户端应用程序支持的显示技术，客户端应用程序可能是一个基本的 Web 浏览器。

这些情况有时会让人很难适应。例如，如果过去只使用 ArcMap 或 QGIS 等桌面软件，可能不习惯考虑宽带速度的影响或与其他人共享机器的情形。

### 2.2.1 通过 HTTP 请求返回地图

这里首先通过对 URL 进行请求，在浏览器导航栏中输入 <http://webgis.cn/834>（图 图 1.3），会跳转到一个更加完整的 URL。对 URL 进行分解，如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mf3t.map
2 layer=topo
3 mode=map
```

如果在学习过程中自行部署服务器，记住替换主机名或者 IP 地址；例如使用本地主机名 localhost，或者直接使用 IP 地址，如 127.0.0.1。

这个 URL 可以被分解成三个部分：

第一部分，`http://webgis.cn/cgi-bin/mapserv?`，调用 MapServer 的 CGI 程序。在不同的系统上，可能是 `mapserv` 或 `mapserv.exe`，也有可能是其他名称。如果直接调用打开上面的网址，会得到这个消息：

```
No query information to decode. QUERY_STRING is set, but empty.
```

要得到正确的信息，需要添加额外的参数，也就是接下来的三个部分。查询字符串包含 CGI 参数（变量和它们的值），与被符号（&）分隔的每个参数。

现在来看看查询字符串，第一个参数与值，如 表 2.1

表 2.1: MapServer 请求的基本参数

参数	赋值	说明
map	/owg/mf3t.map	这告诉 MapServer 的 CGI 的程序需要处理解析的地图文件。当然这里的路径是服务器上的路径。
layer	topo	告诉 mapserv 使用 topo 图层，名称来自于在 Mapfile 中的图层命名对象。
mode	map	告诉 mapserv 做什么以模式输出 Mapfile。在这个示例中，它告诉 mapserv 直接将图像发送到 Web 浏览器，无需在服务器上创建临时文件。

## 2.2.2 通过 Web 浏览器查看结果

刚才的地图是由 WebGIS 服务器在服务器端后台生成，然后传送到客户端的；MapServer 可以创建一个图像并保存到一个当地目录，然后在 Web 页面中加载。

要通过 Web 页面查看地图，必须建立最基本的 HTML 文件，并将地图作为 `<img> </img>` 对象放到 HTML 文件中。

这个页面可以是远程服务器中的，也可以是在本地计算机中创建图片对象。用到远程 MapServer 发布的地图结果。在网页中声明图片的代码是：

```

```

上面代码是 HTML 中声明图片的基本用法，不过 `src` 的参数并不是常见的 Jpeg, PNG, GIF 等图片格式，而是 MapServer CGI 程序，以及一些参数。这里只需要知道 MapServer CGI 程序会根据这些参数来返回一幅图片，因此上面的 HTML 图片声明其实没有什么特别。

完整的 HTML 代码如下，这个是最基本的符合 HTML 5 规范的 Web 页面：

```
1  <!DOCTYPE html>
2  <html lang="zh-cmn-Hans">
3  <head>
4      <meta charset="UTF-8">
5      <title> 最简单的 Web 地图</title>
6  </head>
7  <body>
8      
9  </body>
10 </html>
```

## 2.2.3 实时查看 Web 服务请求

在 Web 开发调试中，经常会需要查看 Web 请求的一些细节。现代的 Web 浏览器一般都提供了查看的方法。

下面是一个简单的说明，当浏览一个网站时，可以通过浏览器查看“后台”发出的 Web 请求。这些说明适用于 Mozilla Firefox 提供的工具。Chrome 和其他浏览器有类似的工具，通常以“开发人员工具”或“Web 工具”的名字命名，应该不难找到。

1. 打开 Mozilla Firefox（如果它还没有运行），然后从主菜单中选择 Open menu-> Web Developer->Network。这将打开浏览器底部的“开发人员工具”窗口。在窗口的顶部，就可以看到不同的选项卡“Inspector”、“Console”等。现在，“Network”选项卡处于活动状态，用于监视网络流量。除了“Network”之外，“Console”选项卡对本节也很重要，因为如果 Javascript 代码有问题，它将显示 Javascript 和其他错误消息。

2. 确保在选项卡下面的过滤器菜单中突出显示“ALL”（将包含所有、HTML、CSS、JavaScript 等项）。
3. 浏览地图，这里使用的 URL 为 <http://webgis.cn/mapfile-raster.html>，可以换成其它的 URL。
4. 注意 Web 请求的发送。将会看到很多对地图的请求。
5. 将鼠标悬停在请求上以查看其完整 URL。如果响应是图像，则将鼠标悬停在 URL 前面的小缩略图上会显示返回图像的较大版本。
6. 当选择请求列表中的某个条目时，窗口的右侧部分将显示有关请求和响应的许多附加技术信息。

这种开发工具在以后的章节中很有价值，因为可以开发自己的 Web 地图，主要用于排除地图出现而浏览器没有发送预期响应的情况。

#### 2.2.4 Web 中的图层设计与交互元素

到目前为止，对于 Web 地图绘制者来说，最大的挑战是了解地图中显示的数据量，以及如何以亚秒级的速度获取在网络用户屏幕上绘制的所有信息。许多使用过桌面 GIS 软件包的人习惯于在地图上添加几十（甚至几百）层，并根据需要打开或关闭它们。强大的桌面计算机可能能够处理此类地图的绘制；但是，如果尝试将地图直接移动到 Web 上，性能将非常缓慢。服务器需要宝贵的时间来遍历所有这些层、检索数据、绘制数据并将图像发送回客户端。

为了解决这个问题，大多数 Web 地图将层分解成处理方式非常不同的组。其唯一目的是提供地理环境的图层被组合在一起，并作为一个平铺的基础地图带入地图。相反，主题层（地图的焦点层）作为一个或多个单独的 Web 服务引入，并放置在基础地图的顶部。用户还可以决定包含一组交互式元素，如弹出窗口、图表、分析工具等。

网络地图通常配备有交互式元素，可以帮助了解地图中的图层。这些可以是单击某个功能时出现的信息弹出窗口，在页面的单独部分中绘制的图表和图形，调整地图中显示的数据时间片的滑块，等等。有些 web 地图允许实时编辑地理信息系统数据，或向服务器提交地理处理作业并查看屏幕上绘制的响应。

这些元素使地图生动起来。制作一个有效的网络地图的关键是要包含对用户最有用的交互元素，而不是让他们无法选择或者使任务过于复杂。即使是一点点的改进，比如在弹出窗口中包含用户友好的字段别名，也可以大大提高地图的友好性和实用性。

交互元素是 web 地图中最需要自定义编程的部分。可自由添加的交互性数量也许与自己知道如何进行的 JavaScript 编程数量密切相关。诸如 OpenLayers 和 Leaflet 之类的开放式 Web 制图 APIs 提供了一些方便的方法来执行某些最常见的事情，例如打开弹出窗口。

## 2.3 地图中内嵌比例尺与图例

地图三要素包括比例尺、方向与图例（或图例与注记）。现代地图尤其是 WebGIS 中，地图的方向基本都是上北下南，一般可以用经纬网标识出来。

在 MapServer 中地图的比例尺与图例都属于地图中的附加部分，相对比较简单，放到一起来说明。比例尺提供了一种关联地图特征与真实世界特征的方式，图例展示了不同的地图符号表征的意义。在地图制图中，这两个部分都可以嵌入到地图中，作为地图的一部分；嵌入的图例是数字时代之前的人工制品，当时制图时会受到纸张大小与材质等各方面的限制，有时候还会在地图中添加与地形无关的装饰。但是作为互联网中的地图，它们很多时候是与地图本身分开的。在 WebGIS 中，使用外部的图例是首选。MapServer 支持这两种情况。

### 2.3.1 地图比例尺的概念

地图比例尺是地图上的线段长度与实地相应线段经水平投影的长度之比。它表示地图图形的缩小程度，又称缩尺，由比例尺图示和文本组成，描述比例尺中线段表示的距离。向地图文件添加比例尺设置非常简单。一般地，地图比例尺越大误差越小，图上测量精度越高。

严格讲，只有在表示小范围的大比例尺地图上，由于不考虑地球的曲率，全图比例尺才是基本一致的。通常绘注在地图上的比例尺称为主比例尺。在地图上，只有某些线或点符合主比例尺。比例尺与地图内容的详细程度和精度有关。一般讲，大比例尺地图，内容详细，几何精度高，可用于图上测量。小比例尺地图，内容概括性强，不宜于进行图上测量。

### 2.3.2 在 MapServer 中创建地图比例尺

在 Mapfile 中，比例尺对象开始于关键字 SCALEBAR，终止于关键字 END，并且包含控制外观和位置的关键字。

这里使用的 Mapfile 为 <http://webgis.cn/mfa7.map>，

对于本示例，比例尺在 8 至 25 行中定义的。比例尺里面，为了指定与比例尺相关的字体大小和颜色，需要定义一个 LABEL 对象（9 - 13 行）。

```

8  SCALEBAR
9    LABEL
10      COLOR 0 0 0
11      ANTIALIAS TRUE
12      SIZE SMALL
13  END
14  POSITION LR
15  INTERVALS 5
16  STATUS EMBED

```

(下页继续)

(续上页)

```

17      SIZE 144 5
18      STYLE 0
19      UNITS MILES
20      BACKGROUNDCOLOR 255 0 0
21      IMAGECOLOR 255 255 255
22      COLOR 128 128 128
23      OUTLINECOLOR 0 0 255
24      TRANSPARENT OFF
25      END

```

关键字 STATUS 可以采取三个值：ON，OFF，或 EMBED。在 STATUS 中，ON 指示一个单独的比例尺图像将被创建，OFF 状态表示没有图像被创建。如果喜欢在地图本身嵌入比例尺，而不是创建一个单独的图像，可以设置为 EMBED。

嵌入式比例尺可设为六个位置中的一个。该位置由关键字位置通过以下两个字母的代码设定值：UL，UC，UR，LL，LC，LR 等，这些值用来指定文本标注位置时具有相同的含义。

关键字 SIZE 被分配一对整数值表示标尺图像的一对宽度和高度的（像素）。关键字 INTERVAL 用分配的值指定比例尺中的间隔次数。关键字 STYLE 的值可以取 0 或 1。关键字 UNITS 可以设置的值为：feet、inches、kilometers、meters、miles、nauticalmiles。基于选择的值，MapServer 将计算比例尺间隔和标签的长度。

SCALEBAR 对象中有几种颜色相关的关键字，他们的名字并不总是意味着它们的特征。关键字 IMAGECOLOR 指定比例尺图像的背景颜色，其中包括格子本身和任何文字标签。（如果该关键字透明设置为 on，IMAGECOLOR 将是透明的。）关键字 BACKGROUNDCOLOR 的设置比例尺中格子的背景色，而不是文本标签。关键字 COLOR 用于设置比例尺的前景颜色。如果被指定，那么多单间隔的颜色和 BACKGROUNDCOLOR 将沿长度方向交替。最后，关键字 OUTLINECOLOR，指定格子周围的 1 个像素宽的边界（不包括任何标签文本），如 图 2.6。

这个地图可以通过 URL <http://webgis.cn/522> 访问，解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[\?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfa7.map
2 layer=topo
3 mode=map

```

SCALEBAR 设置的所有选项都列在 MapServer Mafile 参考文档中。添加到地图文件的第一行是 UNITS 关键字。DD 告诉 MapServerP 制图数据以十进制坐标/单位表示。此设置会影响整个地图文件，而不仅仅针对是比例尺的，这就是为什么它不在 SCALEBAR ... END 对象内的原因。

Mapfile 的 SCALEBAR 对象有多个选项。此示例中使用了一些（但不是全部）选项，都是一些常用的设置。STATUS EMBED 选项与前面图例示例中使用的选项类似，它将图形放在地图上，而不是要求创建单独的图形。单位组号指定刻度条上显示的测量单位。在这种情况下，刻

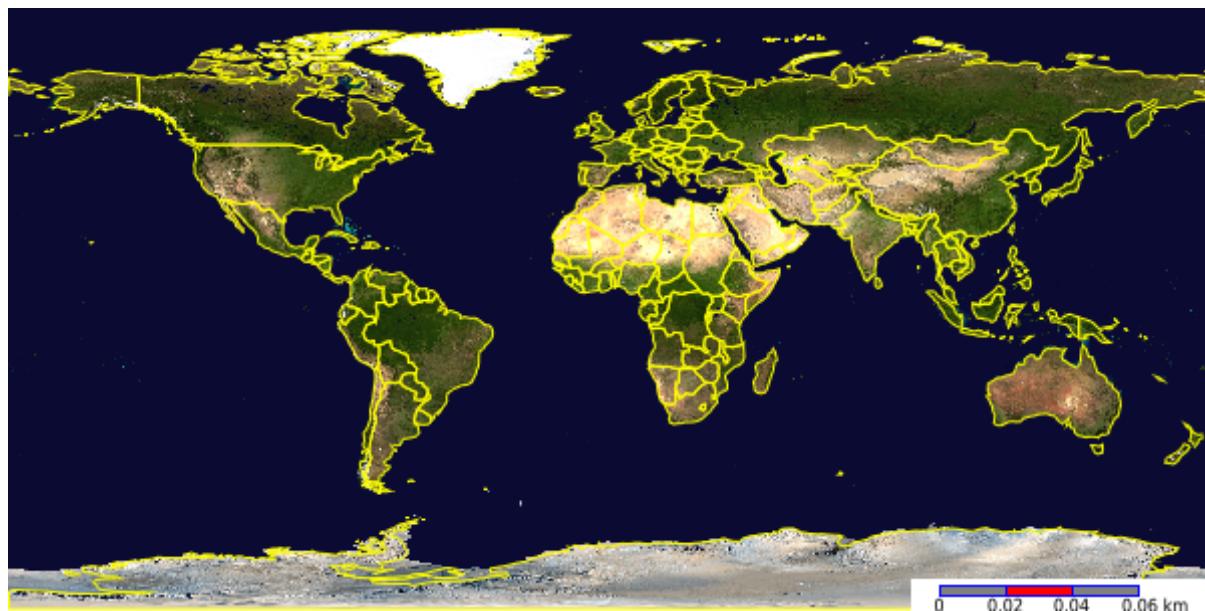


图 2.6：添加地图比例尺

度条的每个段都标有公里。这可以更改为英寸、英尺、米等。

地图单位，坐标系和地图投影的主题不在本章范围之内。当使用来自不同来源的数据时，或者想要为不同目的制作地图时，这一点很重要。

如果不想在地图中嵌入比例尺，请将 STATUS EMBED 更改为 STATUS ON，并使用比例尺命令行实用程序创建单独的图形图像。它与前例示例中的图例实用程序一样使用。下面的示例演示如何使用命令：

```
scalebar mfa7.map scalebar.png
```

### 2.3.3 地图图例的概念

在地图上表示地理环境各要素，比如山脉、河流、城市、铁路等所用的符号叫做图例，它是地图上各种符号和颜色所代表内容与指标的说明，集中于地图一角或一侧。图例是表达地图内容的基本形式和方法，是现代地图的语言，是读图和用读所借助的工具。地图符号一般包括各种大小、粗细、颜色不同的点、线、图形等。符号的设计要能表达地面景物的形状、大小和位置，而且还能反映出各种景物的质和量的特征，以及相互关系。因此图例常设计成与实地景物轮廓相似的几何图形。它有助于用户更方便的使用地图、理解地图内容。

### 2.3.4 在 MapServer 中创建图例

任何地图的一个重要部分是描述颜色和符号含义的图例。MapServer 可以在生成地图时实时创建图例，也可以作为单独的图像创建图例。

在 MapServer 中，要显示在图例中的符号需要命名类所使用的符号。回想一下，绘制一种分类时并不需要被命名，但是，为了在图例中显示出来，必须有一个名字。要向地图添加图例，必须添加 `LEGEND ... END` 对象到地图。请注意，这不是 `LAYER` 对象的一部分。它是整个 `MAP` 对象设置的一部分。

效果为 <http://webgis.cn/542>，如 图 2.7：

图例可以使用 TrueType 字体，并且以这种方式，图例中的 `LABEL` 对象以与在类中标签同样的方法工作。图像的图例的背景颜色用关键字 `BACKGROUNDCOLOR` 设置。

其 Mapfile 为 <http://webgis.cn/mfuk.map>。

```

9      LEGEND
10     STATUS EMBED
11     IMAGECOLOR 230 230 230
12     LABEL
13       TYPE TRUETYPE
14       FONT "arial"
15       COLOR 0 0 0
16       SIZE 10
17       ANTIALIAS TRUE
18     END
19   END

```

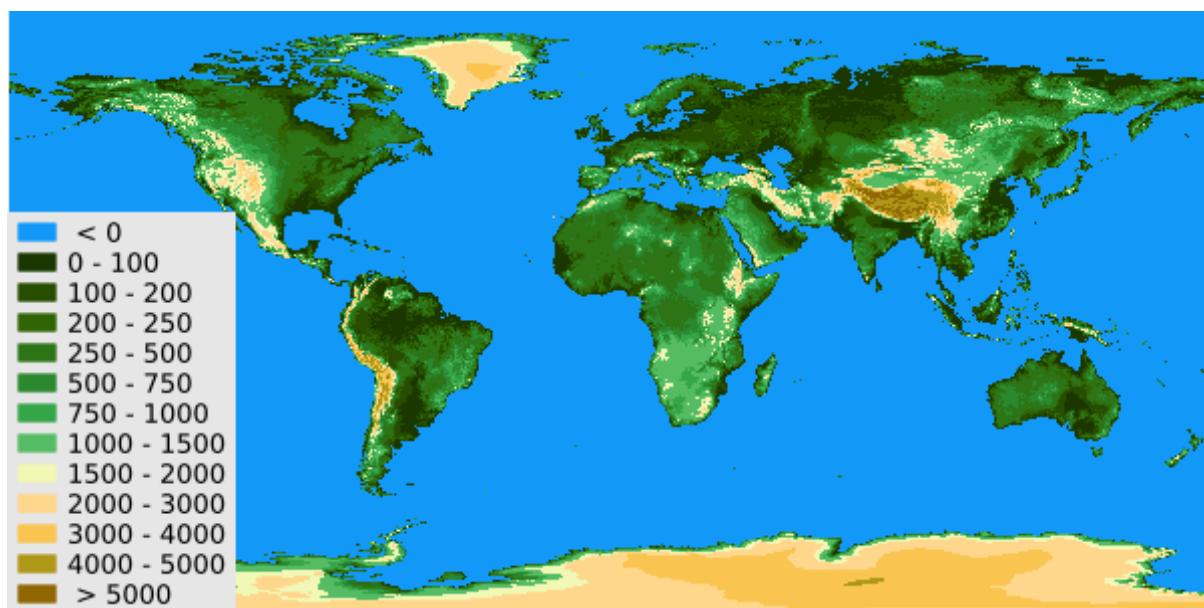


图 2.7：添加地图图例

在此应用中，图例定义在第 9 行至 19 行。关键字 STATUS 的值是 EMBED，因此地图服务器将产生一个内嵌的图例图像。如果状态为关闭，没有图像将被创建，如果状态设置嵌入，然后将图例嵌入在地图图像中。

STATUS 可以设置为 ON、OFF 或 EMBED。EMBED 选项是将图例直接添加到地图的方法。如果不使用 EMBED，则必须将图例创建为单独的图像。将图例设置为具有透明背景也是一个不错的选择。如果不希望它是透明的，则可以删除 TRANSPARENT TRUE 行。图例后面将绘制默认的白色背景。

绘制图例还需要一次更改。必须为每个类别设置 NAME 关键词。给定的名称将是该类符号旁边的图例中显示的文本。例如，将 NAME " < 0" 添加到第一类，将 NAME "0 - 100" 添加到第二类。如果不设置的话则类别不会出现在图例中。

POSITION 设置有几个基本选项，用于告诉 MapServer 使用相对位置的双字母代码将图例放置在哪里：右下角 (LR)、上中心 (UC) 等。

### 2.3.5 legend 命令

到目前为止，图例已嵌入到地图图像中，但也可以创建仅包含图例的单独图像文件。这是使用 legend 命令完成的：

```
legend mfuk.map legend.png
```

legend 命令创建一个图像，其中只有图例。它采用命令名称后两个参数：首先是 Mapfile 文件名，然后是要创建的输出图像文件名。这个命令根据地图文件的 LEGEND 对象中的设置创建图例。可以更改某些设置，但 POSITION 和 EMBED 选项实际上被忽略。这些选项用于 shp2img 或 mapserv Web 应用程序。

## 2.4 定义投影和范围

地图的投影也是一个很大的专题，是地图制图中必须要考虑的。地图投影的实质就是将地球椭球面上的地理坐标转化为平面直角坐标，用某种投影条件将投影球面上的地理坐标点一一投影到平面坐标系内。

MapServer 地图文件中的投影设置分为两个级别：源数据的投影，以及输出地图的投影。

## 2.4.1 地图投影的概念及常用投影

地球是一个赤道略宽两极略扁的不规则的梨形球体，其表面是一个不可展平的曲面，所以运用任何数学方法转换到平面都会产生误差和变形，为按照不同的需求缩小误差，就产生了各种投影方式。

由于投影的变形，地图上所表示的地物，如大陆、岛屿、海洋等的几何特性（长度、面积、角度、形状）也随之发生变形。每一幅地图都有不同程度的变形；在同一幅图上，不同地区的变形情况也不相同。地图上表示的范围越大，离投影标准经纬线或投影中心的距离越长，地图反映的变形也越大。因此，大范围的小比例尺地图只能供了解地表现象的分布概况使用，而不能用于精确的量测和计算。

尽管地图投影非常多，但在 Web 地图中常用的只有两种。一种是简单圆柱投影（Simple Cylindrical projection，又名 plate carrée projection，或 geographic projection），一种是 Web 墨卡托投影。在前端类库 Leaflet 中只支持这两种。

简单圆柱投影就是直接将经纬度作为坐标，映射到平面坐标上。这种投影非常简单，但也因为简单得到了广泛的应用。

前端类库 OpenLayers 能够处理大多数投影。如果不显式设置，地图将使用默认的网络墨卡托投影（EPSG:3857），例如地图 OpenStreetMap 项目和商业产品比如必应地图或者谷歌地图使用了这种投影。如果想要一张显示整个世界的地图，这个投影是个不错的选择。

要创建 WEB 地图可以使用任何坐标系。但是，如果准备将自己的地图切片叠加在 OpenStreetMap 或谷歌、微软、ERSRI 的瓦片地图上，则必须将 GIS 数据转换到 Web 墨卡托投影。该投影创建的目的仅仅是方便将整个世界镶嵌为一组正方形切片。GIS 纯粹主义者拒绝接受该投影方式，并预测这将无法得到大规模认可，但事实与他们的期待大相径庭。

Web 墨卡托投影在很长一段时间内并没有被 EPSG 的投影数据库所接纳。EPSG 认为它不能算作科学意义上的投影，所以只是给了一个 EPSG: 900913 的标号，这个标号游离在 EPSG 常规标号范围之外。因此对于一些老的软件或 API，可能使用了该代码或其他代码。直到 2008 年，EPSG 才恍然明白：不管椭球体还是球体，其实都是对地球的模拟，只是精确程度上的差别，没有本质上的不同。或者是不得不接受广泛的事实标准，EPSG 最终接纳了这个投影，定义投影坐标系 PROJCS 的名字为“Popular Visualization CRS / Mercator”，分配了 EPSG:3785 的代码，但在同一年没多久，又被弃用，重新分配了 EPSG:3857 的正式代码，使用至今。地理坐标系 GEOGCS 的名字为“Popular Visualization CRS”，SRID 为 EPSG:4055。这些标号已经进入“正常范围”。

另外要注意的是，即使使用 EPSG:3857，即 Web 墨卡托投影显示地图，也不能在该投影下进行量测功能，包括线的长度与多边形面积的量算。即使在中纬度地区，结果都存在很大的偏差。要执行量测功能，最好是将几何图形投影到本地坐标系统。

以下链接指向一个小型 Web 应用程序，该程序生动展示了 Web 墨卡托投影如何影响距离和面积计算。

```
http://links.esri.com/web_mercator_measurements
```

## 2.4.2 使用 MapServer 发布 EPSG:4326 投影世界地图

对前面 图 2.7 的示例进行一点修改，使用另外一种投影。效果见 <http://webgis.cn/453>，如图 2.8：

URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]  
0 http://webgis.cn/cgi-bin/mapserv  
1 map=/owg/mfa6.map  
2 layer=topo  
3 mode=map
```

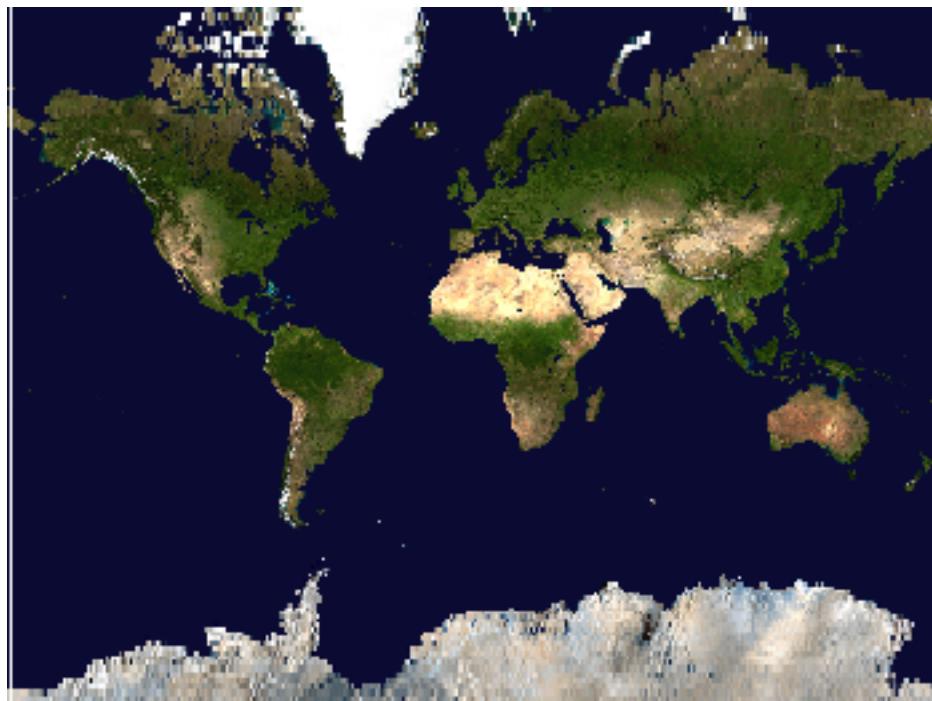


图 2.8：使用 EPSG:4326 投影的世界影像图

下面是本例中用到的 Mapfile 源文件 <http://webgis.cn/mfa6.map>：

```
1 MAP  
2   IMAGETYPE "PNG24"  
3   EXTENT -20042150 -7515806 20104978 7515807  
4   SIZE 350 260  
5   SHAPEPATH "/gdata"  
6   SYMBOLSET "../symbols/symbols35.sym"
```

(下页继续)

(续上页)

```

7  FONTSET ".../fonts/fonts.list"
8
9      PROJECTION
10         "init=epsg:3857"
11     END
12
13     LAYER
14         NAME "topo"
15         DATA "land_shallow_topo_8192.tif"
16         STATUS OFF
17         TYPE RASTER
18         PROJECTION
19             "init=epsg:4326"
20         END
21     END
22
23 END

```

这幅地图也是全球地图，但看起来不太一样。这幅地图使用了另外一种地图投影。

首先会注意到，Mapfile 是原来的 EXTENT 被替换成新的范围值，看起来不像是经度、纬度。此外，增加了一个投影对象 PROJECTION，并在 LAYER 中也添加了 PROJECTION 定义。

经纬度的范围是大部分人都了解了，但是投影之后的范围则会差异很大。看一下新的空间范围的参数。

```
EXTENT -20042150 -7515806 20104978 7515807
```

### 2.4.3 在 Mapfile 中定义投影

可以在 MAP 对象，以及 LAYER 对象中进行投影的定义。在 MAP 对象中可以定义投影（只能定义一次，但可以定义多个，定义多个表示可以有多种投影输出），这个定义的投影是地图的输出投影，MapServer 将会通过这个投影来渲染地图结果。在 LAYER 对象中定义的投影是输入投影，也就是图层对象对应的数据的投影。图层对象中投影的定义可以与地图对象中的不同，MapServer 会进行投影转换，重新投影到输出投影。如果 LAYER 对象中没有投影信息，MapServer 会假设输入投影与输出投影是一致的。

PROJECTION 对象通常用于地图对象和图层对象。它可以为地图指定一个投影，该投影称为输出投影，因为 MapServer 使用该投影绘制并输出地图。当为图层指定一个投影时，表示在读取该图数据时使用的投影，所以又称为输入投影。一般情况下，MapServer 假定所有图层的输入投影和地图的输出投影是相同的。

在 Mapfile 中，PROJECTION 对象并不是必需的。但是如果要支持 OGC 互操作规范（WMS/WFS/WCS）的话，还是需要在 Mapfile 中明确定义的（参见 [节 5.1](#)）。

## 在 MapServer 中使用 PROJ 方式定义投影

MapServer 传统的方式是通过 Proj.4 参数来定义，比如下面定义的美国大陆兰勃特方位等积 (Lambert Azimuthal Equal-Area) 投影，这是输出投影定义：

```
PROJECTION
    "proj=laea"
    "ellps=clrk66"
    "lat_0=45"
    "lon_0=-100"
END
```

代码中使用的关键词 proj , ellps , lat\_0 , lon\_0 都是 MapServer 后面使用的称为 PROJ.4 的投影库的关键字。还有其他选项（例如，基准，单位等），但这是最常见的。PROJ.4 不仅是 MapServer 使用的一组库。它还带有一些命令行实用程序。其中之一是一个名为 proj 的程序，该程序列出了可用的投影，椭圆体等。运行命令

```
> proj -l
```

返回可用地图投影的列表，包括 MapServer 的 proj = 设置中使用的缩写。该实用程序还可以通过键盘或文本文件交互投影坐标；非常适合批量投影坐标。

代码中的第一行 "proj=aea" 是最重要的。它指定投影的名称。缩写 aea 是 Albers Equal Area 投影的缩写。每个 PROJECTION ... END 对象都必须指定一个投影名称。还应该指定一个椭球，例如 ellps = WGS84。某些功能不需要使用 ellps 参数，但如果该参数没有，则某些功能将失败。如有疑问，请使用世界大地测量系统 WGS84。所有其他参数可以是可选的。

第二行 "ellps=WGS84" 指定用于投影的椭球。WGS 84 是地球形状的具体表示。根据要制图的区域，可能会使用或多或少的精确椭球体。共有 42 个。选择一种使用不一定是一件容易的事，除非找到专门针对自己所关注的国家或地区生成的一种。如果可以找到这样的投影，将获得更准确的投影结果。WGS 84 是适用于全球应用的通用椭圆形。

第三行 "lat\_0=10" 将原点指定为北纬 10°。如果未指定，则默认的原始纬度为 0°。在此纬度上，投影的北/南坐标将全部从 0 开始。

## 在 MapServer 中使用 EPSG 代码定义投影

另一种方法是使用 EPSG 代码。这些代码是欧洲石油调查组织 (EPSG) 所定义的标准投影代码（或空间参考标识符）。在 MapServer 中通过下面的 EPSG 代码定义，下面的 EPSG:2163 同样表示兰勃特方位等积投影，效果是一样的：

```
PROJECTION
    "init=epsg:2163"
```

(下页继续)

[\(续上页\)](#)

END

根据要使用的地图投影，可能需要更多或更少的设置。并非总是需要指定所有设置。有一些可用的快捷方式。一种常见的方法是使用 PROJ.4 随附的主空间参考系统，列表在 Debian 系统中，文件位置在 /usr/share/proj/epsg，包含预定义的投影，原点的纬度，椭球等。然后，可以使用单个数字引用一组更复杂的选项。这些代码称为 EPSG 代码。它们是由欧洲石油调查组 (EPSG) 开发和维护的。

显然，这两个选项中都使用了多个选项。两者都是单行，但为清楚起见，此处显示为多行。不必像示例中那样指定每一个，只需提供 EPSG 代码即可：

```
PROJECTION
  "init=epsg:26910"
END
```

在 MapServer PROJECTION 对象中使用 EPSG 代码时，文本 epsg 应该为小写。在其他上下文中使用时（例如，使用 MapServer 的 Open Geospatial Consortium Web Map Server 请求），文本必须为大写。例如，如果在图层的 METADATA 对象中具有 wms\_srs 属性，则它将为："wms\_srs" "EPSG:26910"。在 MapServer 中使用 EPSG 代码时，这是一个常见问题。

这将自动设置所有投影选项，以使用 NAD83 基准将地图投影到 UTM 区域 10N 中。下一部分将描述 UTM 区域和其他投影。

同样，"init=epsg:42101" 会将其设置为 Lambert Conformal Conic (LCC) 投影，并提供适合于加拿大地图绘制的特定选项。此定义被认为是自定义或非标准的。它是独立于 EPSG 组织而开发的，不一定会出现在默认的 EPSG 文件中。PROJ.4 附带有许多自定义定义，它们位于单独的文件中。如果要使用自定义投影，只需将其他文本定义行添加到 EPSG 文件中，即可立即使用它们。



## 第3章 使用 MapServer 发布地图（2）

这一章重点针对矢量数据说明地图发布，矢量数据结构更加复杂，也提供了更多制图的可能性。

在本章中，将探索 **Mapfile** 的一些使用关键细节，渲染不同格式的地图。它会更加的复杂并且能够证明 **MapServer** 作为渲染地图引擎的强大能力。将会看到如何使用层和类来更加有效的显示信息。

这一章探讨必须进行的审美判断和 **MapServer** 提供给的实现这些判断方法。将引入一些新的概念，更多的涉及颜色，形状和大小等设置方法，来了解如何制作美观的地图。

如何制作精美地图并非完全是技术问题。本书会介绍制图的一些技术，但并不保证会制作好精美的地图来。地图的制作是一个不断优化的过程。

这一节先对 **Mapfile** 的逻辑结构和 **MapServer** 的简单的处理进行一些介绍，以形成概括性的框架。在 **MapServer** 地图文件中还可以定义比例尺（Scalebar）、图例（Legend）和参考地图（Reference Map）等对象。

这一章的内容会涉及到：

### 3.1 使用 MapServer 发布矢量数据

这一节开始在 **MapServer** 中使用矢量数据。**GIS** 中的矢量数据与 **CAD** 软件，或 **Illustrator** 中的矢量数据类似，使用抽象的点、线、面来表达事物，除了有地理空间位置的差别，还有一个重要的方面就是 **GIS** 中的矢量数据是没有样式信息的。**CAD** 或 **Illustrator** 以绘制为主，所采用的矢量数据，需要使用宽度、颜色、样式等来进行区分与美化，在 **GIS** 中，则将这些工作交由地图制图阶段来完成，同样的数据，可以在不同的专题图中，不同的应用场景中使用不同的样式来绘制。

在 **MapServer** 中可以定义任意多的图层。旧版本的 **MapServer** 是有图层数目限制的，但在新版本中不必考虑这个问题。

### 3.1.1 图层的概念

图层的概念对所有地图数字化过程至关重要的，前面已经用到，在矢量数据中理解起来有一些不同，这里进一步进行说明。简单地说，一个图层是来自一个单一的空间数据集，所有都在同一比例尺的绘制。大多数（尽管不是全部）有效的地图是对现实世界的比例表示。地图在某个尺度下的数据可能会是  $1:1,000,000$ ，在地图上 1 英寸（或英里或米），在地面上等于 1,000,000 英寸（或英里或米）。这些各种各样的地图，所有的要素都呈现在同一范围内，以保持相对的尺寸和距离。

不成比例显示地图的一个很好的例子是地铁地图。地铁地图一般都是示意性的，没有相关联的比例尺；因为地铁站的更重要的是它们之间的序列，而不是位置精确。

每个空间要素有一个类型，这就决定了要素元素是如何被存储，检索和呈现。MapServer 能识别的矢量数据类型是 POINT、LINE 与 POLYGON。由于在结构和渲染需求的这些差异，一个图层只包含一种要素类型。

1. 点状类型拥有的唯一特征是它的位置，此位置可以在地图上表示，具有特定的大小和颜色的符号。
2. 线状要素（点组成的有序序列）由特定的宽度和颜色表示，也可以具备样式（例如虚线）。
3. 多边形（点的有序序列包围的区域），可以利用线状要素的所有样式，但因为它包围区域，多边形可以有一个填充颜色。

在数据类型方面，MapServer 使用的类型要少的多，很多数据类型在制图表达中是没法分辨的。在目前相关标准与软件中，除了基本的点、线、面，还有复合点、复合线、复合面，以及集合等类型。

图层的使用使得生成的地图更加灵活，它允许地图的组件以不同的方式进行组合，以满足不同的需要。在 MapServer 中，使用图层对象作为基本的绘制单元。

### 3.1.2 在 Mapfile 中定义矢量图层

MapServer 通过 LAYER 对象定义图层。此对象指定要呈现的数据集、类型、状态（即，该层是否应该显示）。也可以提供其它任选层的特性（如图层的名称，在该图层应该呈现的最大或最小的尺度，和标签信息）。

每一个 Mapfile 至少需要一个图层。Layer 对象告诉 MapServer 渲染什么，另一个结构性对象 CLASS，告诉 MapServer 在一层中如何来渲染要素的指令。

这个例子使用将矢量数据（Shapfile）作为图层发布。发布后的效果如 图 3.1：

可以不在 HTML 页面浏览它，仅仅输入这个网址 <http://webgis.cn/123>，这会返回一张图片。

网址解析如下：

图 3.1: MapServer 渲染的单图层矢量数据

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfa1.map
2 layer=world-country
3 mode=map
```

对应的 Mapfile <http://webgis.cn/mfa1.map> , 内容为:

```
1 MAP
2     IMAGETYPE "PNG"
3     EXTENT -180 -90 180 90
4     SIZE 600 300
5     SHAPEPATH ""
6     IMAGECOLOR 255 255 255
7     LAYER
8         NAME "world-country"
9         DATA "wcountry.shp"
10        STATUS OFF
11        TYPE POLYGON
12        CLASS
13            NAME "The Upper Great Lakes States"
```

(下页继续)

(续上页)

```

14      STYLE
15          COLOR 232 232 232
16          OUTLINECOLOR 32 32 32
17      END
18  END
19  END
20 END

```

该层开始关键字为 LAYER，结束关键字为 END。使用了 wcountry.shp 的空间数据，多边形层渲染了全球的行政区划数据。NAME 关键字指定层的名称。

STATUS 关键字决定该图层是否渲染，以及是否可以改变其状态。STATUS 默认图层总是渲染，同时可以改变状态。

在栅格数据中，使用 CLASS 进行分类；在矢量数据中，同样也是使用 CLASS 关键字，但在用法上差异很大。

12 至 18 行指定这一层唯一的 CLASS 对象参数。一个 CLASS 对象的由关键字 class 开始和关键字 END 终止（18 行）。在这里指定一个单一的默认类，它将包括 Shapefile 中的每一个要素。一个类的名称将出现在与地图相关的图例上。如果一个类没有名字，它依然会被渲染，但不会出现在图例中。样式对象关键字 COLOR 指定绘制的颜色；OUTLINECOLOR 指定绘制边界的颜色，用于概述多边形和某些标记符号的颜色。对线状要素没有影响，轮廓的宽度可以使用 WIDTH 来指定。如果未指定 WIDTH，则将默认绘制一个像素的轮廓。如果为 STYLE 定义了 SYMBOL，则将使用 OUTLINECOLOR 为该 SYMBOL 创建轮廓。如果没有为 STYLE 定义 SYMBOL，则多边形将获得轮廓。因为这层是一个多边形，它会被指定的颜色填充。注意，如果图层是线状图层，COLOR 值将指定线状要素颜色。

### 3.1.3 使用两个图层的静态地图

进一步来了解图层定义的顺序对绘图结果的影响。MapServer 是按照地图文件中定义的图层相反的顺序来绘制和显示各个图层，即最先定义的图层显示在地图的最下层。MapServer 会根据上覆图层的显示与否自动判断某一图层是否绘制，从而可以适当优化地图绘制的性能。

示例如 图 3.2，这个地图的 URL 为 <http://webgis.cn/723>。

URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfa2.map
2 layer=world-country
3 layer=country-line
4 mode=map

```



图 3.2：两图层静态地图

这个例子与前面的稍有不同，多边形的颜色进行了修改。

#### 3.1.4 在 Mapfile 中定义多个矢量图层

看一下这个例子使用的 Mapfile (<http://webgis.cn/mfa2.map>)，这个 Mapfile 定义了两个图层：

```

7 LAYER
8     NAME "world-country"
9     DATA "wcountry.shp"
10    STATUS OFF
11    TYPE POLYGON
12    CLASS
13        NAME "States"
14        STYLE
15            COLOR 232 232 232
16        END
17    END
18
19 LAYER
20     NAME "country-line"
21     DATA "wcountry.shp"
22     STATUS OFF
23     TYPE LINE
24     CLASS
25         NAME "State Boundary"
26         STYLE

```

(下页继续)

(续上页)

```
27      COLOR 132 132 32
28      END
29      END
30      END
```

查看这个 `Mapfile` 文件，就会发现在多边形图层后面（下面）添加了一个新的 `LAYER` 对象。在第一层仍然是一个多边形层，但样式中不再有 `OUTLINECOLOR` 属性，只定义了填充的颜色；第二个图层的 `TYPE` 是 `LINE`，颜色的值与在第一个例子中的 `OUTLINECOLOR` 的颜色不同，定义了多边形边界的样式。这样会生成与第一个示例略有差别的图像。

图层的类型不需要匹配 `Shapefile` 的类型，但是层必须符合 `Shapefile` 的要求。本案例中使用的 `Shapefile` 是多边形。然而，定义的层却为线状类型 (`LINE`)，因为渲染的仅仅是国家的边界。

### 3.1.5 MapServer 图层定义顺序

这里解释一下为什么要这样处理。如果继续添加态层的顶部上的层，轮廓最有可能被掩盖了这些层。为了能够看到国界线，添加其他层，必须从多边形层分离国界线层，并把它放在其他的其他层的顶部。如何定义/添加图层是有顺序的，按照“后来者居上”（后面定义的图层会显示在上方）记住即可。

`MapServer` 的渲染层是按照他们在 `Mapfile` 中指定的顺序，`Mapfile` 中的最后一层是地图渲染结果的顶层，在前面的所有层的上面来构建地图。这意味着，`Mapfile` 中早一些显示的细节（即，在地图图像中的较低层）可能会被遮住。当渲染多边形层时这是最重要的，因为多边形可以用一个指定的颜色填充。如果地图服务器渲染点层（即，例如，城市的位置），然后渲染状态一个多边形层，点状图层将不可见，除非指定多边形层的颜色是透明的。这个问题很容易补救，将多边形层定义在前面。也可以通过一系列的线（即一个型线层）渲染多边形。由于线状要素是二维的，没有要填充的区域，所以下面层的细节仍然是可见的。无填充颜色时，指定值为：`-1, -1, -1`，也可以达到透明度的效果。

更多图层的定义与使用与上面的说明类似，不多赘述。

### 3.1.6 数据的选择

在前面的例子中，图层由一组来自相同的数据集的要素组成，所有的要素使用同样的样式来呈现。然而，可能不希望绘制包含在一个给定的数据集的所有特征，或者也可能不希望以同样的方式绘制的所有特征。例如，如果有一个数据集包含所有公路，使用这样的数据集来渲染世界地图将需要相当长的时间，并且产生的地图过于杂乱以至于不是太有用的；同样，使用相同的符号来渲染乡村道路与高速公路产生的地图，用户无法选择出行的优先级别；而在现实世界上，这种区别是很明显的。很多时候要对数据按类别划分，然后进行分类展示。

然而，基于前面提到的两个原因：首先，可能不希望绘制每一个特征；第二，可能想用不

同的符号，颜色或尺寸呈现属性不同的某些特征。通常希望避免只使用默认的类，并转而选择依据特征对数据进行分类显示。

```
FILTERITEM [attribute]
FILTER [string]
```

Item to use **with** simple FILTER expressions. OGR **and** shapefiles only.

此参数允许在执行空间过滤的同时执行特定于数据的属性过滤，但在计算任何类表达式之前。字符串只是一个 MapServer 表达式：

```
FILTER ("[type]">'road' and [size]<2)
```

Native filters are supported through the NATIVE\_FILTER PROCESSING key:

```
PROCESSING 'NATIVE_FILTER=id=234'
```

```
LAYER
  NAME 'regexp-example'
  FILTERITEM 'placename'
  FILTER /hotel/
  ...
END
```

FILTER 是区分大小写的，如果有一个字符不同都会被过滤掉。

在属性 year 中匹配具有 2000 到 2099 之间值的记录：

```
FILTERITEM "year"
FILTER /^20[0-9][0-9] /
```

## 3.2 根据属性进行分类显示

在一个空间数据集的层中所呈现的所有要素必须是相同的几何类型。然而，不是每一个点，线，或一个数据组中的多边形需要具有相同的重要性，道路有不同的级别，应该在制图中进行区别以避免（可能是灾难性的）混乱。换句话说，要素分类根据特定标准，不同的级别与类型应该有不同的展示方法。

每一个图层都需要至少一个 CLASS 对象，不需要指定选择标准。如果他们没有指定，在数据集中的所有要素将默认被包含在类中。MapServer 使用类级别的关键字表达式指定选择条件。一个类的样式对象包含了类渲染的信息（如符号大小和颜色）。在一个图层可以定义多个类，每种类的要素可以用不同的风格来渲染。

根据属性分类需要了解数据的信息。如果想查看 Shapefile 或其他矢量数据的字段，可以有几种方法。Shapefile 的数据库记录存储在 DBF 文件。打开一个电子表格程序，如 Openoffice.org 的 CALC，或者在桌面如 GIS 软件 QGIS。在 Excel 打开相关的 DBF 文件或其他可以读取 DBF 文件的电子表格程序，可以看到要素属性值。如果数据有元数据（应该有），可以通过元数据文件的内容来了解。如果只想找出属性的名称，可以使用的实用程序 dbfinfo，shapelib 库的一部分。还可以使用 GDAL 工具 ogrinfo 显示矢量数据的基本属性信息；ogrinfo 提供地理信息，以及要素值。

### 3.2.1 MapServer 中分类展示的几种方法

在一个图层中的不同分类可以用多种方法来区分，但是一个分类的定义方法只能是一种。

分类展示需要使用表达式用于将属性值与某些逻辑检查相匹配。MapServer 可以使用两种类型的表达式：第一是使用某个字段 (CLASSITEM) 进行比较，第二种是使用逻辑表达式比较一个或多个属性。对于每一种类别指定使用的属性值，这是最简单的表达形式。表达式可以是比较这更加复杂的，允许一个常规的比较表达式或逻辑表达式。

字段比较又分成三种情况：

1. 字符串比较：将单个属性与字符串值进行比较。
2. 正则表达式：单个属性与正则表达式匹配。
3. 列表表达式：将字符串属性与多个可能值的列表进行比较

这一节介绍使用单个字段进行分类，下一节介绍使用表达式进行分类。

### 3.2.2 使用字符串比较

CLASS 允许基于属性区分要素，并基于类区分特征。前面的地图使用 CLASS，因为每一层都至少有一个类。最简单的，默认的 CLASS 包括数据集的每一个特征。在这种情况下，该层具有一个 CLASS 并且在这个类中的每一个特征以同样的方式呈现。

下面是一个有不同特征的例子。通过使用在数据中的非空间属性信息，可以创建这样的地图 <http://webgis.cn/483>，如图 3.3：

这里用到的数据为全球城市数据。这个数据中有一个字段 STATUS 来说明城市的类别，这个类别可以直接从字面意义看得出来。

Mapfile 的 URL 为 <http://webgis.cn/mfk3.map>，其中的关键代码如下：

```

20      CLASSITEM "STATUS"
21      CLASS
22          NAME "Provincial capital"
23          EXPRESSION "Provincial capital"

```

(下页继续)

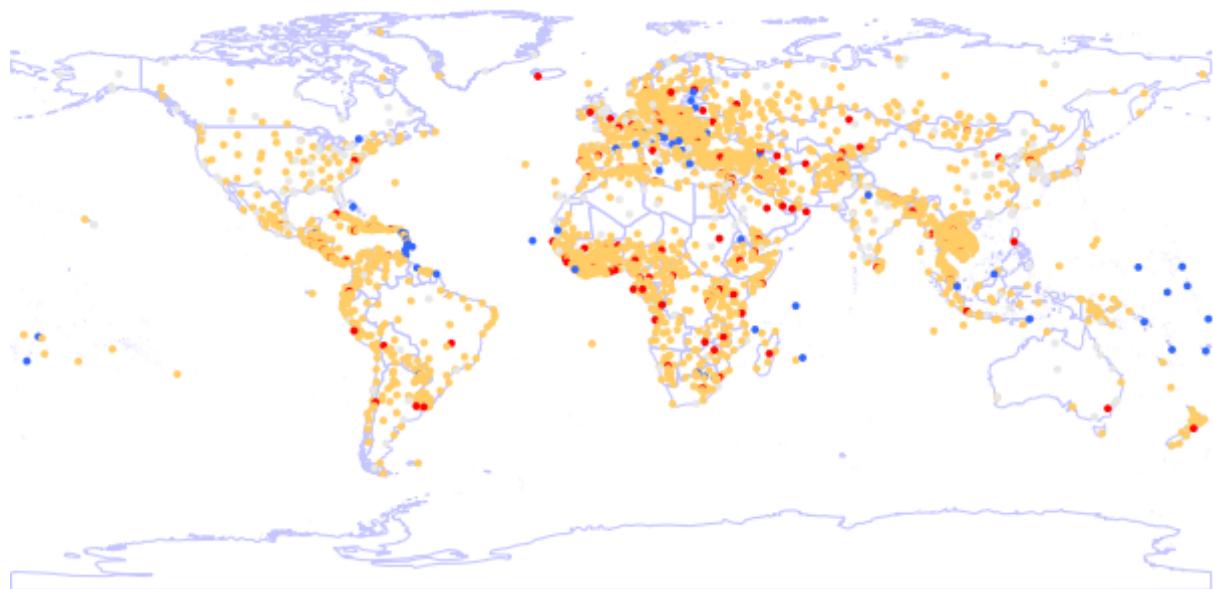


图 3.3：基于字符串比较分类展示图图

(续上页)

```

24      SYMBOL "circle"
25      SIZE 4
26      COLOR 255 204 102
27  END
28  CLASS
29      NAME "National capital"
30      EXPRESSION "National capital"
31      SYMBOL "circle"
32      SIZE 4
33      COLOR 51 102 255
34  END
35  CLASS
36      NAME "National and provincial capital"
37      EXPRESSION "National and provincial capital"
38      SYMBOL "circle"
39      SIZE 4
40      COLOR 255 204 102
41  END
42  CLASS
43      NAME "States"
44      SYMBOL "circle"
45      SIZE 4
46      COLOR 232 232 232
47  END

```

Mapfile 仍然只有两个图层，但多边形层被细分为几个类别。CLASSITEM 属性表达字符串

的值将与数据集的每一个功能。如果表达字符串匹配 CLASSITEM 相同的值，该要素将被包含在此类中。虽然快速且易于使用，这种方法不是很灵活，因为指定的字符串表达式必须通过 CLASSITEM 确定匹配的属性值。

在第 20 行 CLASSITEM 关键字是用来指定分类的对象使用什么属性。在这个例子中，属性是 STATUS。

如果表达式的值由一个引号的字符串组成，然后对于数据集中的每一个要素，MapServer 会比较引用的字符串值和由 CLASSITEM 指定的属性值。如第 23、30、37 行，如果要素的属性与给定的值相等，该要素将依照此类的设定进行展示。

使用表达式定义类的最简单的（和最快）方法来确定类包含的 MapServer 是使用字符串的比较。使用层级的关键字 CLASSITEM 识别的属性名称，将用于分类的功能。然后指定比较字符串使用类级别的关键字表达。

### 3.2.3 使用正则表达式的要素分类

MapServer 的正则表达式与字符串比较的工作方式类似，但允许更复杂的操作。它们比纯字符串比较慢，但可能仍然比逻辑表达式更快。对于字符串比较，当使用正则表达式时，如果项目未包含在 LAYER FILTER 或 CLASS EXPRESSION 中，则必须定义 FILTERITEM (LAYER FILTER) 或 CLASSITEM (CLASS EXPRESSION)。

正则表达式是一个复杂的话题，在这里无法对其详细说明，请参阅任何标准 Unix 文本或指南的正则表达式语法说明。

如果表达式的值由正则表达式组成，正斜杠分隔，然后对于每个数据集的要素，MapServer 会比较正则表达式和由 CLASSITEM 指定的属性值。如果找到一个匹配，该要素包含在类。

<http://webgis.cn/485>

在 Mapfile 正则表达式必须用正斜线 (/) 包括起来。关键字 EXPRESSION 所指定的正则表达式的与 CLASSITEM 属性相比，并如果找到一个匹配包含在类。该规范通过正则表达式匹配的标准是非常灵活的，但它仍然限制一个单个的属性的值相匹配。例如，下面的代码：

使用的 Mapfile 为 <http://webgis.cn/mfk5.map>。

```

21      CLASS
22          NAME "land"
23          EXPRESSION /1|3/
24          SYMBOL "circle"
25          SIZE 4
26          COLOR 255 204 102
27      END

```

这个例子使用 POP\_RANK 这个字段，表示人口数据的级别。



图 3.4：基于正则表达式分类展示图层

在这种情况下，有时为了书写 **Mapfile**，可能会对属性值进行规范约定来配合正则表达式的要求。在数据不满足的情况下可以对属性进行重新赋值。

正则表达式通常由具有特殊含义的字符和按其原因解释的字符组成。具有特殊含义的字符如表 3.1：

表 3.1: 特殊字符

字符	含义
.	将匹配单个字符。
[ ]	用于分组。例如，[A-Z] 将匹配字符 A, B, C, ..., X, Y, Z。
{, }	和 * 用于指定某些内容应匹配的频率。
^	匹配开头，\$ 匹配值的结尾。
\	反斜杠用于消除特殊含义。例如 \, \$ 将匹配美元符号。

MapServer 支持两种正则表达式运算符：

- ~ 大小写敏感的正则表达式
- ~\* 不区分大小写的正则表达式

注意正则表达式对大多数的 Unix 和类 Unix 操作系统来说是一个共同的特点。正则表达式（或正则表达式）是一串表示字符串模式的字符。与 DOS 和 Windows 的通配符 ? /\* 类似（但更广泛），正则表达式用于寻找匹配的字符串模式。MapServer 的正则表达式（在 Linux 下运行）

与 POSIX 兼容，所以其功能可移植到 Windows。正则表达式的语法说明超出了本书的范围，任何标准的 UNIX 用户指南或众多在线资源咨询其中任意一个都可以对这个问题有帮助。

POSIX 是可移植操作系统接口（Portable Operating System Interface）的首字母缩写。

POSIX 是基于 UNIX 的，这一标准意在期望获得源代码级的软件可移植性。换句话说，为一个 POSIX 兼容的操作系统编写的程序，应该可以在任何其它的 POSIX 操作系统（即使是来自另一个厂商）上编译执行。POSIX 标准定义了操作系统应该为应用程序提供的接口：系统调用集。POSIX 是由 IEEE（Institute of Electrical and Electronic Engineering）开发的，并由 ANSI（American National Standards Institute）和 ISO（International Standards Organisation）标准化。大多数的操作系统（包括 Windows NT）都倾向于开发它们的变体版本与 POSIX 兼容。

### 3.2.4 使用列表表达式进行分类

在 MapServer 6.4 以后的版本中支持列表表达式。列表表达式（参见 MS RFC 95：在表达式解析器中支持列表）是一种将字符串属性与多个可能值的列表进行比较的有效方法。这种实现方式与正则表达式或下一节的逻辑表达式有些重复，但是它们的性能显著提高，可读性也更好。要使用它们，请在 {} 之间用逗号分隔的值列表，而不添加引号或额外空格。这里没使用实际的案例，只给出一些示例代码。

```
LAYER
  NAME 'list-example'
  CLASSITEM 'roadtype'
  ...
  CLASS
    EXPRESSION {motorway,trunk}
    ...
  END
  CLASS
    EXPRESSION {primary road,secondary road}
    ...
  END
END
```

意义是很明显的，不作过多说明。

### 3.3 基于逻辑表达式的分类显示

通过使用包含一个或多个属性值的逻辑表达式 EXRESSION，能够实现更为复杂的分类。在这种情况下不需要指定 CLASSITEM（实际上，如果指定的话也会被忽略）。语法很简单：一个逻辑表达式由名括在方括号中的“属性”，一个“比较操作符”和“值”构成，这个逻辑表达式可以进一步与其他逻辑表达式进行布尔计算。

如果表达式的值是由一个逻辑表达式组成，然后对于每个数据集的记录（要素），MapServer 计算逻辑表达式。如果评价为 True，则该类别中包含相应的要素。

#### 3.3.1 逻辑表达式的数据类型

在逻辑表达式中使用了方括号 [] 来引用矢量数据的属性，需要注意属性类型的表达方式。属性类型由下列方式决定：

包含在引号或单引号字符中的属性认为是字符串：

```
"[string_attribute]" or '[string_attribute]'  
"string_literal" or 'string_literal'
```

没有使用引号则认为是数值：

```
[numeric_attribute]  
numeric_value
```

如果使用了反引号则认为是日期时间：

```
`[date_time_attribute]`  
`date_time_literal`
```

在这里看起来，更像是占位符。

#### 3.3.2 实例

效果如 <http://webgis.cn/484>。

使用的 Mapfile 为 <http://webgis.cn/mfk4.map>。

```
42      EXPRESSION ( '[STATUS]' ne 'Provincial capital' )
```

在这种情况下，会使用字符串比较。在第 42 行，关键字表达式告诉 MapServer，数据集的每一个要素，属性 STATUS 将会与字符串 Provincial capital 进行比较。如果它们是不相等的，要素是该类的成员，将被渲染。



图 3.5：按属性类型显示图层（基于表达式分类）

不同的语言中，如 C、Perl、MapServer 的使用不同的比较字符串以及比较数字，必须查看与使用差异。如果一个属性是字符串值，然后其引用必须用引号括起来，单引号与双引号都可以，但是必须匹配。考虑下面的代码：

```
EXPRESSION ( '[STATE_FIPS]' eq 'MN' )
```

这将包括一个特征仅当这个值属性 STATE\_FIPS 是等于字符串 MN 的。逻辑表达式可以使用连词和分离运算符 (AND, OR) 被结合起来。考虑下面的例子：

```
EXPRESSION (( [POPULATION] < 100000 ) and ( '[STATE_FIPS]' eq 'MN' ))
```

这将匹配人口小于 100000 同时 STATE\_FIPS 等于 MN 的要素。

注意，如果一个字符串值属性包含数字字符串（如 "123"），可能有混乱出现。如果数字比较字符串值的属性，永远不会有得到匹配（123 永远不会等于 "123"），并且也不会产生错误。可以在 QGIS 中查看数据，进一步了解数据类型。

还要注意，属性包含单引号（'）或双引号（“）字符可能让 MapServer 混淆。例如，表达式 ([NAME]='O'Doyle') 不会选择要素的属性 [NAME] 等于 O'Doyle 的，因为三个单引号，在 MapServer 中是一个无效的字符串值。因此，表达式将永远不会计算为 TRUE。为了解决这个问题，可以使用双引号，修改为这样的表达式 (" [NAME]" EQ "O'Doyle")。“其他”选项改变数据的设置和用双引号替换每一个出现的单引号。

### 3.3.3 MapServer 中支持的操作符

下面列出了 MapServer 中的操作符 表 3.2：

表 3.2: MapServer 中的操作符

Operator	Data Type
$!=$	Numeric
$=$	Numeric
$>$	Numeric
$<$	Numeric
$\geq$	Numeric
$\leq$	Numeric
and	Logical
or	Logical
eq	String
ge	String
gt	String
le	String
lt	String
ne	String

除了各种比较运算，MapServer 还支持长度函数，返回其字符串值的参数的长度（以字符为单位）。

## 3.4 在地图中定义与使用符号与样式

在前面的章节中，看到了如何使用类级别的关键字颜色的颜色特征。通过使用可扩展的符号 (SYMBOL)，可以在 CLASS 对象设置符号的样式与大小（以像素为单位），从而产生更丰富的地图表现形式。

符号定义不包含颜色信息，颜色设置在 STYLE 对象中。线条的端点、连接处样式也在 CLASS 对象的 STYLE 属性中定义，不在 SYMBOL 中定义。

Style 保存符号化和样式的参数。可以在 CLASS 或 LABEL 中应用多种样式。该对象出现在 MapServer 4.0 中，目的是将逻辑与外观分开，从而获得可通过 Mapfile 重用的命名样式。

### 3.4.1 MapServer 中定义符号

符号定义可以包含在主 Mapfile 中，或者更常见地包含在单独的文件中。使用 SYMBOLSET 关键字指定单独文件中的符号定义，作为 MAP 对象的一部分，推荐这样的设置，非常适合在多个 MapServer 应用程序中重复使用符号定义。MapServer 中有 3 种主要的符号类型：Markers，Lines 和 Shadsets。值 0 始终是特定类符号的退化情况：对于点符号 0 是单个像素，对于阴影（即，填充的多边形）符号 0 是实心填充，对于线符号 0 是单个像素宽线。

符号在地图级别定义，因此可以用于所有图层中的所有类别。符号以关键字 SYMBOL 开始，以关键字 END 结束。关键字的名称 NAME 是用来给某个符号分配一个名称；使用时，用这个名称来引用符号的定义。有几种类型的符号：矢量符号（VECTOR）、椭圆符号（ELLIPSE）、像素图（PIXMAP）、SVG 文件（SVG）、TRUETYPE 和 HATCH。

1. 一个矢量符号（VECTOR）由一系列描述符号的形状的点组成。定义时使用关键字 POINTS 开始，随后由坐标对，以终止关键字 END 定义。
2. 椭圆符号（ELLIPSE）与矢量符号使用相同的语法，但解释是不同的。关键字 POINT 中只包含一个数值对，这些数字被解释为椭圆的长轴和短轴的相对长度（在 x 和 y 方向）。如果两个坐标相等，则是一个圆。
3. 像素图符号（PIXMAP）使用 GIF 或 PNG 图像作为标志，SVG 文件（SVG）使用 SVG 格式的图像作为标志，这两种类型都使用包含该图像的文件被通过关键字 IMAGE 的值识别。
4. HATCH 关键词只用于多边形，在（多边形）形状中生成阴影线。

虽然所有的符号定义可以被放置在一个外部文件（由关键字 SYMBOLSET 的值在 MapServer 上确定），在本章中将它们放在 Mapfile 中让事情变得简单。

接下来分别以点、线、面三种类型的几何对象作为示例进行说明，但是要注意几何对象与符号类型并没有相关性，一般情况下同样的符号可以用于不同的几何对象。

### 3.4.2 点状符号的定义

在这里看一下不同点状符号的定义，如 图 3.6：

图 3.6 (a) 定义了一个圆。一个符号可以作为轮廓或填充多边形绘制。设置关键字 FILLED 为真，将导致符号被使用设置的颜色绘制。

- NAME [string] 定义符号的别名。这个名称用在 CLASS STYLE 对象中。
- TYPE ellipse 定义符号的类型。
- POINTS [x y] [x y] …END：表示构成 TYPE 向量符号或定义 TYPE 椭圆符号的 x 和 y 半径的点序列的开始。末尾用关键字 END 表示。X 和 Y 可以用十进制数给出的值。最大 x 和 y 值定义符号的边界框。符号的大小（实际高度）在 STYLE 中定义。x 值向右增加，y 值向下增加。由于坐标关键字点 POINTS 的坐标值是相等的，会画一个圆圈。



图 3.6: MapServer 中的点状符号

- FILLED [true|false] 默认值为 false , 针对类型为椭圆和矢量。如果为 true 则将视为多边形, 如果为 false 则将视为线条。

```

8   SYMBOL
9     NAME "circle"
10    TYPE ELLIPSE
11    FILLED TRUE
12    POINTS
13      4 4
14    END
15  END

```

定义了符号后, 必须应用到某种 CLASS 对象中, 符号使用时是作为 STYLE 的对象出现。

- SYMBOL 引用呈现要素符号的名称。
- COLOR [r] [g] [b] | [十六进制字符串] | [属性] , 用于绘制符号的颜色。
- SIZE [double | attribute] , 要使用的符号/模式的高度。默认值取决于符号的类型。
- WIDTH [double | attribute] 指的是绘制线条的粗细。默认值为 1.0。

```

71  CLASS
72    NAME "a1"
73    EXPRESSION "1"
74    STYLE
75      SYMBOL "circle"

```

(下页继续)

(续上页)

```

76      SIZE 8
77      COLOR 100 100 100
78      END
79      END

```

图 3.6 (b) 例子定义了正方形，这里与上个示例不同之处在于对 TYPE 符号定义不同，将其设定为矢量符号 (VECTOR)，并且 FILLED 给定之为 false，符号将不会有填充色。

- ANCHORPOINT [x] [y] : 用于指定旋转符号并将符号放在地图上时用作锚点的位置 (符号内)。默认值为 0.5 0.5 (对应于符号的中心) 示例与上一个相同，但在定义符号时进行了调整，将其设定为矢量符号 (VECTOR) 并给定 5 个坐标关键点，呈现出一个正方形。

```

16 SYMBOL
17   NAME "square"
18   TYPE VECTOR
19   POINTS
20     0 0
21     0 1
22     1 1
23     1 0
24     0 0
25   END
26   ANCHORPOINT 0.5 0.5
27 END

```

图 3.6 (c) 例子定义了五角星，这个例子虽然复杂，但结构很简单，只是在用坐标绘制五角星的顶点时需要更多的坐标。

```

28 SYMBOL
29   NAME "star-dots"
30   TYPE VECTOR
31   FILLED TRUE
32   POINTS
33     0 0.375
34     0.35 0.375
35     0.5 0
36     0.65 0.375
37     1 0.375
38     0.75 0.625
39     0.875 1
40     0.5 0.75
41     0.125 1
42     0.25 0.625

```

(下页继续)

(续上页)

```
43      END
44      END
```

图 3.6 (d) 例子并没有定义新符号，而是前面两种符号 ((b) 与 (c)) 的组合。通过这种组合，可以产生复杂的样式。

```
101     STYLE
102         SYMBOL "square"
103         SIZE 20
104         COLOR 100 100 100
105     END
106     STYLE
107         SYMBOL "star-dots"
108         SIZE 16
109         COLOR 200 10 10
110     END
```

图 3.6 (e) 的例子创建了两条线段。一方面可以使用样式的组合来实现，但 MapServer 提供了更简便的方法。这里通过在 POINTS [x y] [x y] ...END 中插入 -99 -99 来创建非连续路径。

```
45     SYMBOL
46         NAME "cross"
47         TYPE VECTOR
48         POINTS
49             0 0
50             1 1
51             -99 -99
52             1 0
53             0 1
54     END
55 END
```

图 3.6 (f) 例子使用了字体符号，使用了定义好的字体 "mapsym"，符号则通过字符（这里是 "&#53;"）来定义，产生了个飞机的符号。这个符号固然可以使用坐标绘制，但是比较麻烦。有很多字体都已经完成了这样的绘制工作，直接采用是比较方便的方案。像 ArcGIS 软件中就提供了很多符号的字体，但是在使用的时候要注意版权。

- CHARACTER [char]：用于引用特定 TrueType 字体字符的字符。需要弄清楚从键盘字符到字体字符的映射。
  - FONT [string]：要在 FONTSET 中定义的 TrueType 字体的名称。
- 在代码中首先要载入字体字义，然后定义样式：

```
7 FONTSET "./fonts/fonts.list"
```

```
56 SYMBOL
57   NAME "aircraft"
58   TYPE TRUETYPE
59   FONT "mapsym"
60   FILLED TRUE
61   ANTIALIAS TRUE
62   CHARACTER "&#53;"
63 END
```

### 3.4.3 线状符号与样式的定义

线状符号与样式的示例如图 3.7。

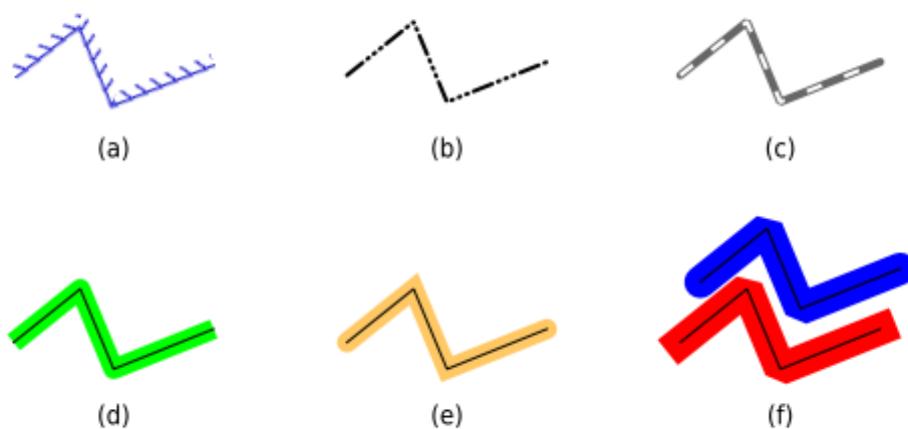


图 3.7：MapServer 中的线状符号

### 线的符号与样式

可以使用符号来绘制线状要素，符号会沿着线的方向延伸。注意第 30 行使用了角度旋转。

- ANGLE [double | attribute | AUTO]，以度为单位的角度，用于旋转符号（逆时针）。默认值为 0（无旋转）。

图 3.7 (a) 代码为：

```

8   SYMBOL
9     NAME "v-line"
10    TYPE VECTOR
11    FILLED FALSE
12    POINTS
13      0 0
14      5 10
15      10 0
16    END
17  END

```

```

27   STYLE
28     SYMBOL "v-line"
29     SIZE 8
30     ANGLE 60
31     COLOR 0 0 200
32   END

```

虚线图案或风格，可以设置关键字 PATTERN。与 PATTERN 相关联的值是一个整数序列。第一个整数指定绘制的像素的数目，下一个整数指定跳过的像素的数量，下面再指定绘制的像素的数目，如此反复。如图 3.7 (b) 代码为：

```

37   STYLE
38     COLOR 0 0 0
39     WIDTH 2
40     PATTERN
41       10 3
42       0 3
43       0 3
44       0 3
45     END
46   END

```

铁路的样式是非常常用的，使用两种样式的组合。如图 3.7 (c) 代码为：

```

51   STYLE
52     COLOR 102 102 102
53     WIDTH 4.0
54   END
55   STYLE
56     COLOR 255 255 255
57     WIDTH 2.0
58     LINECAP BUTT

```

(下页继续)

(续上页)

```

59      PATTERN
60          8 12
61      END
62  END
63  END

```

### 线的端点与连接点样式

线状要素的端点与连接点可以有不同的样式，产生不同的效果。下面的例子中都绘制了原始的线，用作比较。

- LINECAP 定义端点，可以使用的值包括 BUTT ROUND、SQUARE
- LINEJOIN 定义连接方式，可以使用的值包括 ROUND MITER BEVEL NONE

图 3.7 (d) 使用了 BUTT 端点，ROUND 连接方式，代码为：

```

67      STYLE
68          COLOR 0 255 0
69          WIDTH 10
70          LINECAP BUTT
71          LINEJOIN ROUND
72      END

```

如图 3.7 (e) 使用了 ROUND 端点，MITER 连接方式，代码为：

```

81      STYLE
82          COLOR 255 200 100
83          WIDTH 10
84          LINECAP ROUND
85          LINEJOIN MITER
86          LINEJOINMAXSIZE 3
87      END

```

如图 3.7 (f) 这个示例使用了两个样式，首先使用了 SQUARE 端点，BEVEL 连接方式；第二个样式没定义 LINECAP 样式，缺省为 ROUND，LINEJOIN 为 NONE，另外，这个样式还定义了 OFFSET，水平方向向右偏移了 10 个像素，垂直方向向上（值为负时）偏移了 30 个像素。代码为：

```

96      STYLE
97          COLOR 255 0 0
98          WIDTH 16
99          LINECAP SQUARE

```

(下页继续)

(续上页)

```

100      LINEJOIN BEVEL
101      END
102      STYLE
103          COLOR 0 0 255
104          WIDTH 16
105          OFFSET 10 -30
106          LINEJOIN NONE
107      END

```

### 3.4.4 面状符号定义

如图 3.8。

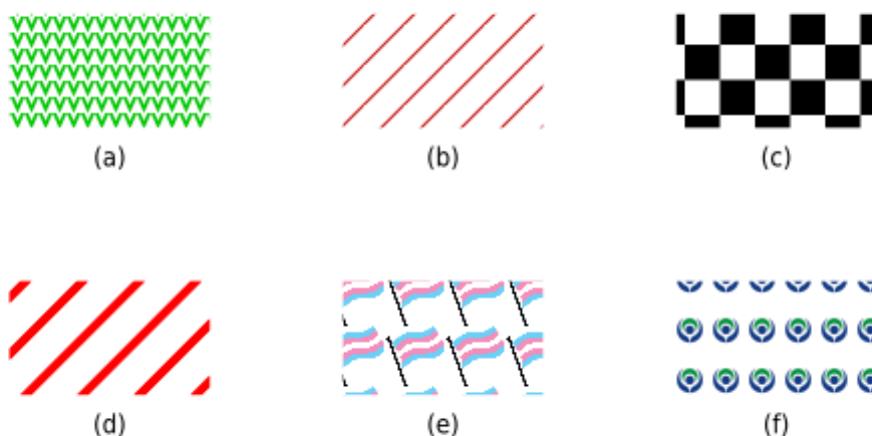


图 3.8: MapServer 中的面状符号

图 3.8 (a) 使用了前面定义的样式，只需要在 **STYLE** 对象中引用：

```

72      STYLE
73          SYMBOL "v-line"
74          SIZE 8
75          COLOR 0 200 0
76      END

```

图 3.8 (b) 定义了斜线绘制方式。

- TRANSPARENT [color index] : 为像素图符号设置输入图像的透明颜色，或确定是否所有阴影符号都应具有透明背景。对于阴影符号，可能需要使背景特征“透视”透明阴影

图案，从而创建更复杂的图。默认情况下，符号的背景与父图像相同（即颜色 0）。这是用户可配置的。

```
22 SYMBOL
23   NAME "diag45fill_thin"
24   TYPE VECTOR
25   TRANSPARENT 0
26   POINTS
27     0 6
28     6 0
29   END
30 END
```

图 3.8 (c) 定义了国际象棋棋盘的黑、白方格交替的样式：

```
31 SYMBOL
32   NAME "chess"
33   TYPE VECTOR
34   FILLED TRUE
35   POINTS
36     0 0
37     10 0
38     10 10
39     0 10
40     0 0
41     -99 -99
42     10 10
43     20 10
44     20 20
45     10 20
46     10 10
47   END
48 END
```

图 3.8 (d) 代码使用了 HATCH 类型，这是面状要素常用的填充方式。

```
49 SYMBOL
50   NAME "hatch-test"
51   TYPE HATCH
52 END
```

图 3.8 (e) 与 (f) 分别使用了 PIXAMP 与 SVG 两种类型，定义时只需要指定相应文件的路径：

```
82 SYMBOL
```

(下页继续)

(续上页)

```

83      NAME "img-type"
84      TYPE PIXMAP
85      IMAGE "flag-v.png"
86  END

```

```

58      SYMBOL
59          NAME "svg-type"
60          TYPE SVG
61          IMAGE "flag-g.svg"
62      END

```

### 3.4.5 在 MapServer 中使用自定义符号

有了前面的定义，在使用时就很简单了。这里使用了绘制全球城市的简单案例，查看 <http://webgis.cn/895>，如图 3.9。

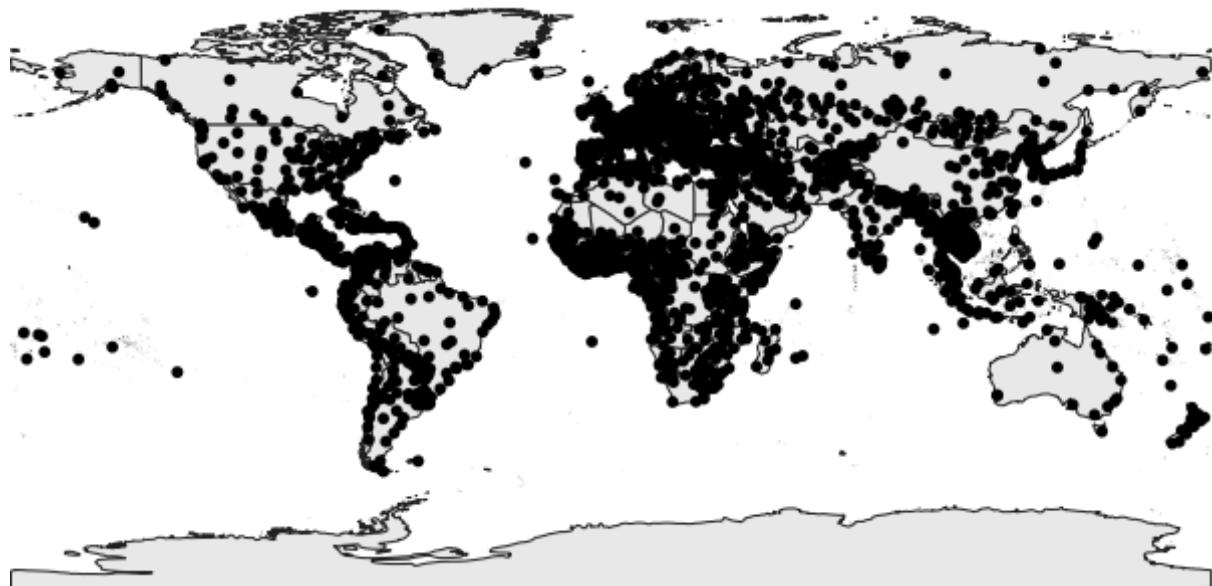


图 3.9：使用符号显示全球城市

使用的 Mapfile 为 <http://webgis.cn/mfy3.map>，内容为：

```

7      SYMBOL
8          NAME "circle"
9          TYPE ELLIPSE
10         FILLED TRUE
11         POINTS
12             10 10

```

(下页继续)

(续上页)

```
13      END  
14      END
```

## 3.5 对地图进行文字标注

MapServer 有一个非常灵活的标注引擎，它支持本地的点阵字体和 TrueType 字体。生成地图时，MapServer 会自动执行多项任务，它标注要素并防止相邻标注之间的冲突。

### 3.5.1 字体的定义

在文字标注中需要用到字体文件，按以下语法进行定义。

```
FONTSET ".../fonts/fonts.list"
```

在这里 FONTSET 指定了 TrueType 字体列表（或字体的完整路径集）的文件。在 Mapfile 中，通过使用这种方式将样式、字体声明为外部文件。该文件列出了每个可用字体。

fonts.list 文件的内容如下，在此文件的同一文件夹下面，则放置各种字体文件。注意 Mapfile 中字体的名称可以与实际的字体没有关系。

```
arial           Vera.ttf  
arial-bold     VeraBd.ttf  
arial-italic   VeraIt.ttf  
times          VeraSe.ttf  
times-bold     VeraSeBd.ttf  
vera_sans      Vera.ttf  
vera_sans-bold VeraBd.ttf  
vera_sans-italic VeraIt.ttf  
vera_serif    VeraSe.ttf  
simsun         simsun.ttc
```

### 3.5.2 标注功能说明

一个标注对象开始于关键字 LABEL，终止 END 关键字。它包含在一个 CLASS 对象，并且针对这一类别进行标注。标注具有多种特征，包括字体，颜色，方向，位置和大小。

标注对象也可以用在图例对象或比例尺对象，语法几乎是相同的。

## 点状要素的标注

实际的标注文本通常是在矢量数据的属性表中。使用该属性指定使用关键字 LABELITEM 。与 LABELITEM 相关的值必须是属性名。在案例说明中属性的名称是 text 。

进行文字标注需要在 Mapfile 中进行一些通用设置。与定义 CLASS 对象类似，需要事先为 LABEL 对象指定一个字段 (LABELITEM)。如：

```
21    CLASSITEM "idx"
22    LABELITEM "text"
```

案例的效果如 图 3.10 。

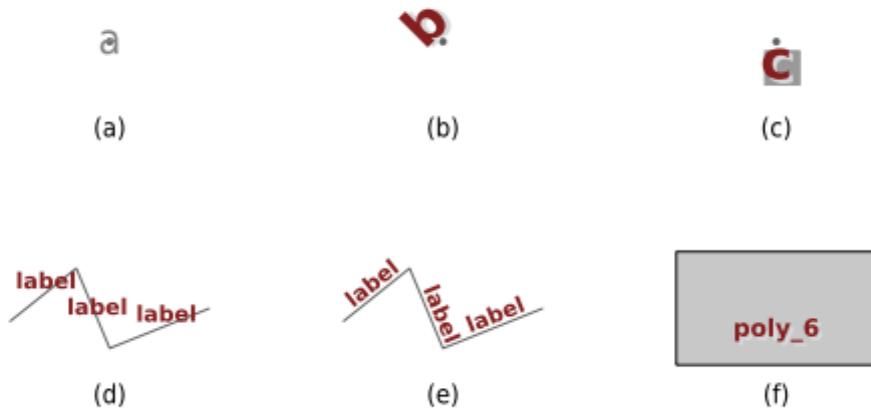


图 3.10：MapServer 中的不同标注方式

图 3.10 (a) 代码如下：

```
31    LABEL
32        TYPE BITMAP
33        COLOR 132 131 131
34        SIZE GIANT
35    END
```

TYPE 在 LABEL 对象，类型指定使用什么类型的字体。可以选择的 TrueType 或 BITMAP (点阵字体，也是缺省值)。第 1 个例子设置为 BITMAP 。

SIZE 设置字体大小，通常通过字号来确定。位图字体的大小可以设置为 TINY 、 SMALL 、 MEDIUM 或 GIANT 。使用数字来指定字体大小 (或像素) 必须是整数，当缩放生效时 (为 LAYER 指定了 SYMBOLSCALEDENOM )， SIZE 给出了在地图比例 1:SYMBOLSCALEDENOM 中使用的字体大小。另外也可以指定矢量数据的属性用于标注大小 (在 5.0 版中引入的)，属性需

要使用中括号来定义，如 [字段名称]。

图 3.10 (b) 展示了位置与方向不同的示例，代码如下：

```

45      LABEL
46          COLOR 132 31 31
47          SHADOWCOLOR 218 218 218
48          SHADOWSIZE 2 2
49          TYPE TRUETYPE
50          FONT "arial-bold"
51          SIZE 20
52          POSITION UC
53          ANGLE 45
54          PARTIALS FALSE
55          MINDISTANCE 300
56      END

```

TYPE 使用 TrueType 字体会产生更高质量的标签，需要更多的设置和一些字体文件，具体的字体则由 FONT 关键词指定。使用 TrueType 字体只能通过一个数值指定标注的大小，而不是像 TINY 或 LARGE 这样的文字说明。

文字标注可以添加阴影。SHADOWCOLOR 指定的标注文字的阴影颜色。SHADOWSIZE 指定阴影的大小，该值对应于在 X 和 Y 方向的像素数目。因此，第 48 行是指两个像素宽、两个像素高。

POSITION 定位标注文本的标注点。该值是垂直和水平位置的组合（垂直在前，水平在后）。垂直对齐有以下几种选择：C 为居中垂直对齐，U 为上，和 L 为下。对于水平对齐有以下选择：C 为居中，L 为左，R 为右。因此，要标注文本对齐要素中心，应使用值 CC（中心 - 中心）；如果在右上方，应使用 UR。使用 AUTO 值会让 MapServer 自动决定标注的最佳位置。

关键字 ANGLE 可以用来指定绘制标注时的角度。一个角度为正则按逆时针方向旋转，而为负按顺时针方向旋转。这里为设置为 45 度。

PARTIALS 告诉 MapServer 是否生成不全的标注文本。这里的默认值是不产生标注文本的碎片。该值可设置为 TRUE 或 FALSE。

MapServer 虽然决定在哪里绘制标注，但它不是“AI”，并不拥有任何审美意识。在能找到空间的情况下，MapServer 将进行尽可能多的标注。这可能导致的情况是一个单一的特征很大程度上（就像高速公路）可能有多个标注。如果标注相距足够远，这不会导致问题。然而，当它们靠得太近，需要绘制相同的功能的标注数量需要进行限制。要做到这一点，可以使用关键字时 MINDISTANCE 指定像素之间重复的标注的数量。另一方面，某些功能可能是如此之小（如大尺度的一个小湖），标注的大小与特征相比将相形见绌。关键字 MINFEATURESIZE 设置将被标记为最小特征尺寸（像素）。

图 3.10 (c) 为文字添加背景，代码如下：

```

67      LABEL
68          COLOR 132 31 31
69          SHADOWCOLOR 218 218 218
70          SHADOWSIZE 2 2
71          TYPE TRUETYPE
72          FONT "arial-bold"
73          SIZE 20
74          POSITION LC
75          PARTIALS FALSE
76          MINDISTANCE 300
77          BUFFER 4
78          STYLE
79              GEOMTRANSFORM LABELPOLY
80                  COLOR 153 153 153
81                  OFFSET 3 2
82          END
83      END

```

BUFFER 填充（像素）的标注。这是用来提高可读性，避免标注靠得太近。缓冲区 4 个像素的意思是没有标注将被绘制于彼此的四个象素之中。

在定义标注的背景时，使用了 STYLE 对象。在其中将关键词 GEOMTRANSFORM 定义为 LABELPOLY；其它如 COLOR，OFFSET 等意义如前。

### 线状要素的标注

图 3.10 (d) 针对线状要素进行标注，代码如下：

```

98      LABEL
99          REPEATDISTANCE 45
100         COLOR 132 31 31
101         TYPE TRUETYPE
102         FONT "arial-bold"
103         SIZE 8
104         POSITION UC
105         PARTIALS TRUE
106         MINFEATURESIZE 35
107     END

```

图 3.10 (e) 展示了沿线的方向标注的文字，代码如下：

```

116      LABEL
117          ANGLE AUTO

```

(下页继续)

(续上页)

```

118      REPEATDISTANCE 45
119      COLOR 132 31 31
120      TYPE TRUETYPE
121      FONT "arial-bold"
122      SIZE 8
123      POSITION UC
124      PARTIALS TRUE
125      END

```

对于道路和河流（或任何其他的线性要素），标注沿着他们所标注的要素是最好的。这里必须使用 TrueType 字体，因为它可以改变与位图字体创建的标注的方向。代码中第 118 行线状要素的标注的 ANGLE 关键词设置为 AUTO，那么 MapServer 将自动计算使标注沿直线方向标注。

### 面状要素标注

图 3.10 (d) 展示了多边形标注的示例，代码如下：

```

142      LABEL
143          COLOR 132 31 31
144          SHADOWCOLOR 218 218 218
145          SHADOWSIZE 2 2
146          TYPE TRUETYPE
147          FONT "arial-bold"
148          SIZE 9
149          POSITION LC
150          PARTIALS FALSE
151          MINDISTANCE 300
152          BUFFER 4
153      END

```

标注的位置以多边形的中心进行判断，这里设置为 LC，位于中心的左侧中间。

### 3.5.3 地图标注实例

在 节 1.7 说明了直接使用 Mapfile 进行文字标注。在 GIS 中更常用的是将数据存储与数据之中，基于数据进行标注。

下面是地图文件 (`mfa4.map`)，一个带标注的 Mapfile。该文件包含字体别名字体位置由 `FONTSET` 关键字指定。

关键代码如下：

```

9      NAME "states_poly"
10     DATA "wcountry.shp"
11     STATUS OFF
12     TYPE POLYGON
13     LABELITEM "NAME"

```

```

19     LABEL
20       COLOR 132 31 31
21       SHADOWCOLOR 218 218 218
22       SHADOWSIZE 2 2
23       TYPE TRUETYPE
24       FONT "arial-bold"
25       SIZE 8
26       ANTIALIAS TRUE
27       POSITION CC
28       PARTIALS FALSE
29       MINDISTANCE 300
30       BUFFER 4
31   END

```

与 Shapefile 相关的是数据库文件。Shapefile 中的要素在数据库中都有对应的记录。记录描述了要素相关的信息。数据库中的每一列都有名称。

图层对象中 LABELITEM 值设置为 NAME , 选择属性 NAME 作为标签文本的来源。每个标签都将被绘制成 COLOR 132 31 31 。

这个世界地图可以通过 <http://webgis.cn/932> 打开, 如 图 3.11。



图 3.11：进行标注的世界地图

URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfa4.map
2 layer=states_poly
3 layer=states_line
4 mode=map
```

### 3.5.4 标注的自动优化

为了避免地图看起来杂乱，MapServer 会根据数据的比例尺进行标注的优化调整。自 6.2 版以来，MapServer 能够为标签空间有问题的功能绘制标签线（通常在标签文本大于标记的多边形时）。此功能是通过添加的 MS RFC 81：带引线的偏移标签。此功能只针对多边形标注，如图 3.12。效果可通过 <http://webgis.cn/378> 查看：



图 3.12：标注的自动调整

URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfs2.map
2 layer=states_poly
3 layer=states_line
4 mode=map
```

Mapfile 只是修改了第 3 行：

3

EXTENT 73 0 136 55

如果将 EXTENT 行更改为较小的范围，则将显示更多标签，因为国家/地区的大小会更大。

### 3.5.5 文字标注的其它问题

在某些情况下，可能想在一层中只呈现标注，而不包含要素。这可以是有许多原因。可能要栅格图像显示（例如，空中照片或类似的东西），并且要避免在它的上面渲染标记多边形层，这会掩盖光栅层。另外，也可能会被要求多次呈现同一个层，而只需要绘制一次标签。只有标签被绘制的层被称为一个标注层。可以创建标注来分离一个多边形层。与注释的数据类型一起做。

关于文本标注还有一个技巧。在复杂背景（如遥感影像）进行标注时，为了突出标注的文本，可以使用字晕。通过定义 OUTLINECOLOR 的值，可以围绕文本绘制有 1 个像素宽的轮廓，这使得在杂乱的背景下标注更容易阅读。

关键字 WRAP 指定一个字符，会导致标注文本换到一个新行。在这种情况下，已经指定了一个空间。这样做是为了产生多行的标注，如果标注文本比较长会看到这种设置的实际效果。

ANTIALIAS 这将 Truetype 的抗锯齿开启或关闭。对于只针对 GD 渲染器，但是现在新版本的 MapServer 都是 AGG，所以基本可以认为这个关键词已经废弃。

有很多其他 LABEL 关键字尚未讨论，可以通过官方文档进一步了解。

## 3.6 MapServer 输出格式

MapServer 支持多种地图的输出格式，不仅仅是图片。图片格式也有许多选项。比如 Jpeg，不能支持透明。而 PNG 格式是目前最常用的，支持透明。对于矢量数据，是否支持透明是很重要的。

MapServer 可以生成不同图像格式的地图图像、图例和比例条。在本章中，PNG 格式显示在示例中，因为它是默认图像格式。MapServer 包可以具有不同的默认图像格式。许多 MapServer 包通常支持 PNG 格式，但在某些情况下，MapServer 可能无法生成 PNG 文件，或者默认情况下无法生成它们。

可以使用许多不同的选项编译 MapServer。术语“MapServer 包”是指正在使用的程序集，例如 shp2img。这些程序的功能取决于它们是如何编译的。某些包将启用比其他包更多的功能。

由于 MapServer 5.x 使用 AGG 或 GD 2.0.x 生成输出图像，它也支持 RGB（24 位或真彩色）输出。所以，在支持 8 位（索引色或灰度）PNG8，也可以使用 PNG（真彩色）输出。在此案例中使用 PNG 作为 IMAGETYPE。

注意：注意：当使用 RGB 和展示多光谱数据时会有显示的信息损失；当使用 RGB 输入时，使用 PNG 相比 PNG8 输出的时候性能会有显著下降。

MapServer 实际上是可以使用 GDAL 输出数据的，但那是另一个话题。

### 3.6.1 在 Mapfile 中定义输出格式

输出图像格式可以在地图文件中显式设置。这需要在 Mapfile 中添加 OUTPUTFORMAT，指定要请求的输出格式。

当使用 shp2img 命令与 -i 选项时，可以覆盖此设置。在使用缩放栏或图例等命令时，无法覆盖它，如前面在“创建第一个地图图像”中所述。

```
mapserv -v
```

查看终端显示内容中的 OUTPUT，此列表显示此 MapServer 能输出 PNG、JPEG 或 KML 文件格式。这些是 MapServer 内建支持的格式，除此之外，MapServer 还可以通过 GDAL/OGR 支持更多的文件格式。

总体而言，输出格式可以分为三类。一类是针对地图制图的，一类是输出矢量数据，一类是输出栅格数据（非地图）。这三种类别基本也对应着 MapServer Web 服务的三种标准 WMS、WFS 与 WCS。当然现在只碰到了地图制图这一种情况。

对于许多矢量制图应用程序，PNG 输出格式就足够了。在许多情况下，它将是默认输出格式。如果栅格图层的颜色超过 256 色，则使用默认 PNG 图像类型可能无法查看所有的信息。通常，只需增加 MapServer 在输出图像中创建的颜色深度。这可以通过请求 PNG 输出格式的 24 位版本来设置，例如：IMAGETYPE PNG24。

请注意：增加颜色深度还会增加图像的文件大小。在网络上发送图形时，这可能会导致性能问题。

地图文件可以具有零个、一个或多个 OUTPUTFORMAT 对象声明，定义支持的可用输出格式，包括 PNG、GIF、JPEG、GeoTIFF、SVG、PDF 和 KML 等格式。如果在 Mapfile 中找不到 OUTPUTFORMAT 节声明，则将进行隐式声明。只有编译支持的那些实际上才可用。GeoTIFF 依赖于使用 GDAL 支持构建，PDF 和 SVG 依赖于使用 Cairo 支持构建。

列举部分使用方法：

```
9    OUTPUTFORMAT
10       NAME "png"
11       DRIVER "GD/PNG"
12       MIMETYPE "image/png"
13       IMAGEMODE PC256
14       EXTENSION "png"
15   END
16   OUTPUTFORMAT
17       NAME "png24"
```

(下页继续)

(续上页)

```

18     DRIVER "GD/PNG"
19     MIMETYPE "image/png"
20     IMAGEMODE RGBA
21     EXTENSION "png"
22 END
23
24 OUTPUTFORMAT
25     NAME "GTiff"
26     DRIVER "GDAL/GTiff"
27     MIMETYPE "image/tiff"
28     IMAGEMODE RGB
29     EXTENSION "tif"
30 END
31
32 OUTPUTFORMAT
33     NAME "AGG"
34     DRIVER "AGG/PNG"
35     IMAGEMODE RGB
36 END
37
38 OUTPUTFORMAT
39     NAME "AGGA"
40     DRIVER "AGG/PNG"
41     IMAGEMODE RGBA
42 END
43
44 OUTPUTFORMAT
45     NAME "AGGJ"
46     DRIVER "AGG/JPEG"
47     IMAGEMODE RGB
48 END

```

参数含义：

**DRIVER [name]**：用于生成此输出格式的驱动程序的名称。如果驱动程序支持多种格式，则某些驱动程序名称包括格式的定义。对于 AGG，可能的驱动程序名称是“AGG / PNG”和“AGG / JPEG”。对于通过 OGR 的输出，附加 OGR 驱动程序名称，例如“OGR/Mapinfo File”。对于通过 GDAL 的输出，附加格式的 GDAL 短名称，例如“GDAL/GTiff”。

**EXTENSION [type]**：提供在创建此类文件时使用的扩展名。

**FORMATOPTION [option]**：提供驱动程序或格式特定选项。OUTPUTFORMAT 声明中可能存在零个或多个 FORMATOPTION 语句。

- **AGG / JPEG:** “QUALITY = n” 选项可用于设置生成的 jpeg 的质量（值从 0 到 100）。
- **AGG / PNG:** “COMPRESSION = n” 用于确定应用于 png 创建的 ZLIB 压缩。n 预计是 0 到 9 之间的整数值，0 表示不压缩（不推荐），1 表示最快压缩，9 表示最佳压缩。压缩级别需要付出代价（无论是在 cpu 处理还是文件大小方面，都选择最适合的设置）。默认值为

COMPRESSION = 6。

- AGG / PNG 支持从 24/32 位到 8 位的量化，以减少最终图像大小（从而节省带宽）。
- AGG / MIXED 是一个元驱动程序，可根据结果图像是否具有透明度自动选择适当的格式。它需要定义 TRANSPARENT\_FORMAT 和 OPAQUE\_FORMAT 选项并将其设置为另一种有效格式。TRANSPARENT\_FORMAT 和 OPAQUE\_FORMAT 的有效格式是由 AGG / PNG, AGG / JPEG 或 GDAL / xxxx 驱动程序处理的格式。默认情况下，“jpeg\_png” 格式（带有 “image / vnd.jpeg-png” mime 类型）预先配置为当它们是非不透明像素时使用 PNG 图像，而当它们只是不透明像素时使用 JPEG。类似地，“jpeg\_png8” 格式（具有 “image / vnd.jpeg-png8” mime 类型）被预先配置为当它们是非不透明像素时使用 PNG 8 位图像而当它们仅是不透明像素时使用 JPEG。
- AGG / “GAMMA = n” 用于指定要应用于多边形渲染的伽马校正。允许值为 [0.0,1.0]，默认值为 0.75。此值用于防止伪影出现在连续多边形的边界上。设置为 1.0 以禁用伽马校正。

**IMAGEMODE [PC256|RGB|RGBA|INT16|FLOAT32|FEATURE]** : 选择生成输出的成像模式。

对于像 Flash 这样的非栅格格式很重要。并非所有格式都支持所有组合。

1. PC256: 在调色板中生成最多 256 种颜色的伪彩色结果（传统 MapServer 模式）。
2. RGB: 以 24 位红/绿/蓝模式渲染。支持所有颜色，但不支持透明度。
3. RGBA: 以 32 位红/绿/蓝/ Alpha 模式渲染。支持所有颜色和基于 Alpha 的透明度。所有功能都是在最初透明的背景下呈现的。
4. BYTE: 渲染原始 8 位像素值（无演示）。目前仅适用于 RASTER 图层（通过 GDAL）和 WMS 图层。
5. INT16: 渲染原始 16 位带符号像素值（无显示）。目前仅适用于 RASTER 图层（通过 GDAL）和 WMS 图层。
6. FLOAT32: 渲染原始 32 位浮点像素值（无显示）。目前仅适用于 RASTER 图层（通过 GDAL）和 WMS 图层。
7. FEATURE: 输出是非图像结果，例如通过模板或 OGR 写入的功能。

**MIMETYPE [type]** : 选择生成输出的成像模式。对于像 Flash 这样的非栅格格式很重要。并非所有格式都支持所有组合。

**NAME [name]** : 要在地图文件的 IMAGETYPE 关键字中使用的名称，以选择此输出格式。此名称也用于描述允许的 Wxs 格式的元数据，并且可以使用（有时与 mimetype 一起）通过 OGC 请求中的关键字选择输出格式。

### 3.6.2 对比查看不同的输出结果

默认情况下，会使用支持透明的：

<http://webgis.cn/934>

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]  
0 http://webgis.cn/cgi-bin/mapserv  
1 map=/owg/mfa8.map  
2 layer=wcountry-line  
3 mode=map
```

或手工声明：

<http://webgis.cn/920>

```
0 http://webgis.cn/cgi-bin/mapserv  
1 map=/owg/mfa8.map  
2 layer=wcountry-line  
3 map.imagetype=png24  
4 mode=map
```

也可以生成其他格式的，这个是不透明的。注意用的是 png，PNG 文件并不总是透明的。

<http://webgis.cn/527>

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]  
0 http://webgis.cn/cgi-bin/mapserv  
1 map=/owg/mfa8.map  
2 layer=wcountry-line  
3 map.imagetype=png  
4 mode=map
```

在大多数情况下，支持透明总是好的。目前还未遇到过必须要使用不透明图片的情况。

### 3.6.3 MapServer 输出的其他格式

除了图片，也可以作为数据返回，如：

<http://webgis.cn/253>

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]  
0 http://webgis.cn/cgi-bin/mapserv  
1 map=/owg/mfa8.map  
2 layer=wcountry-line
```

(下页继续)

(续上页)

```
3 map.imagetype=gtiff  
4 mode=map
```

打开上面网址，会返回 .tif 格式的文件。由于浏览器不能直接显示 Tiff 格式的文件，所以会弹出对话框进行下载。下载后保存，可以使用 QGIS 打开查看。

结果是作为“图”返回的，而不是“数据”，但是这个图是有地理空间坐标的，可以作为“数据”来使用。

## 第4章 创建交互式地图应用

到目前为止，只能创建和查看静态地图。创建 Web 制图应用程序，通常的意义是创建能够由用户（应用程序的）改变的地图。也就是说，一个用户能够改变地图的内容（或者信息），如放大、缩小，以及控制某个图层是否显示等。要实现这样的功能，需要使用 MapServer HTML 模板。

本章介绍的应用程序将会是交互的并允许用户对地图影像进行缩放和漫游。并进一步介绍怎样生成比例尺及图例，以及怎样在输出的 HTML 中嵌入有效的信息（类似地图的尺寸和鼠标点击的坐标）。

另外，除了动态性，可变换比例尺的地图也会表现出不同的呈现细节的更多方法。这一节针对 CGI 的交互操作进行较多的说明。查询功能也是 MapServer 实现的重要功能只进行简单说明。

MapServer 通常通过 Web 服务器（例如 Apache 或 IIS）使用。任何支持 CGI 应用程序的 Web 服务器都可以使用 MapServer。Web 服务器传递与地图相关的请求，并将结果返回到浏览器。当本书提到“将 MapServer 与 Web 服务器集成”时，它意味着将 mapserv 可执行文件（在 Windows 上为 mapserv.exe）与 Web 服务器一起使用。

### 4.1 MapServer 交互操作：CGI 变量及用户接口

Mapfile 用于定义一个基于 CGI 的地图应用的对象。它不仅决定了地图的外观和感觉，也决定了 MapServer 在被 Web 调用时的如何表现。为了 MapServer 的正常运行，必须了解如何处理几十个 Mapfile 关键字，CGI 表单变量，替换字符串。这些比 MapServer 的地图制图要了解、使用的技术复杂地多。尝试建立一个真正的应用程序需要有更多的知识，但 Mapfile 的逻辑结构和 MapServer 的强大功能，将克服这一障碍。

在 CGI 环境中，用户通过网页浏览器可以连接到 Apache 网络服务器。在这种环境下，Apache 会调用 MapServer，可以从浏览器传递各种形式的变量。使用这些信息，MapServer 生成地图和网页，Apache 再把这些内容返还给浏览器。当然，使用 MapServer 生成地图仅仅需要浏览器的信息是不够的。事实上，CGI MapServer 网络应用有四个组分：Mapfile，HTML 初始化的形式，一个或多个模板文件，以及空间数据（库）。

#### 4.1.1 MapServer CGI 程序处理流程

基于 CGI 程序，MapServer 数据处理流程如 图 4.1：

1. 当浏览器通过 URL 告知 Web Server 调用 MapServer CGI 模块时，Web Server 就为 CGI 模块创建一个进程；CGI 模块运行并加载 URL 指定的 Mapfile 文件，
2. 根据 Mapfile 中指定的 data 路径读取相关数据。将读入的数据进行文件标注并进一步生成地图工作。该项工作调用了 TrueType 字体引擎配合图形渲染库（如 AGG）进行的。图形制作完成后，保存至客户端参数中指定的目录。至此完成制图工作。
3. 读取 Mapfile 文件中的模板文件（HTML 文件），并将模板文件中 CGI 变量的替换变量（template substitutions）替换成具体的值；处理完模板文件后，CGI 模块将模板文件处理结果（HTML 文件）返回给 Web Server，Web Server 在将 HTML 文件输出到用户浏览器上。这时模板文件（HTML 文件）就作为用户的交互界面。
4. 当请求者更改页面上的任何表单元素（例如，通过更改缩放方向或缩放值）并单击提交按钮时，MapServer 将从 Web 服务器接收具有这些新值的请求。
5. 然后循环再次开始。

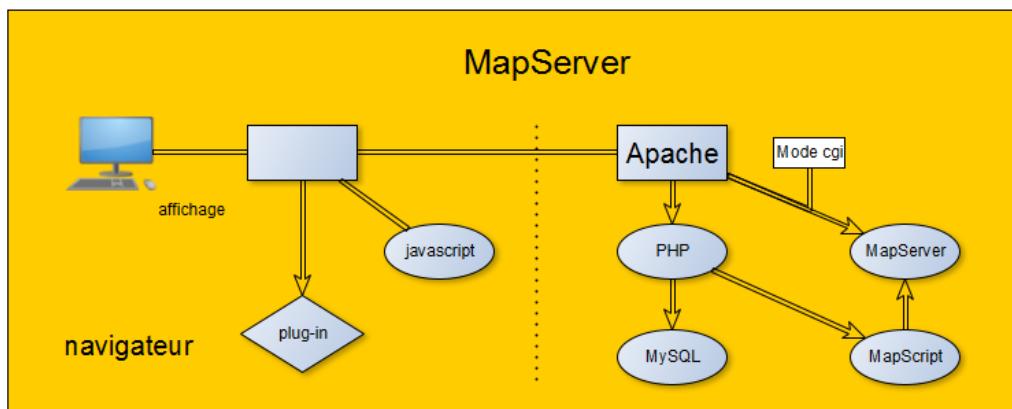


图 4.1: mapserver 应用图

### 4.1.2 使用 MapServer CGI 程序的步骤

了解了 CGI 程序处理流程，再看一下使用 MapServer CGI 程序设计 WebGIS 的基本步骤。前面章节已对地图制图部分进行了说明，在这里不再重复。

#### 1. 配置 Mapfile：

在 Web Object 的 TEMPLATE 属性中指明使用的用户交互界面。用户也可以通过 URL 指明使用的用户交互界面。

#### 2. 设计初始化界面：

考虑到访问 CGI 程序 URL 很长，而且对于普通用户并不知道访问 CGI 程序的 URL 格式；所以在初始交互界面中要嵌入指向 CGI 程序的链接（URL）。

#### 3. 设计用户交互及交互界面：

这里需要引出 MapServer CGI 模块的关键概念：CGI 变量和模板文件。MapServer CGI 变量可以看作是 CGI 模块 mapserv 的接口。在 URL 或 HTML 的表单中调用 CGI 变量就可以完成大多数动态交互，如图层选择，放大缩小等。

模板文件是 HTML 文件或 URL。在模板文件中含有 CGI 变量和它的替换变量，这里替换变量 (template substitutions) 与 CGI 变量相一一对应。

### 4.1.3 CGI 变量与模板文件

MapServer 提供了许多 CGI 变量用于 Web 制图，MapServer 的地图文件中几乎所有的关键字都可以定义为一个变量，但是用于制图的核心的 CGI 变量并不多。另外，用户也可以自定义变量，如用户可以定义一个名为 root 的变量用来代表应用程序的根目录。

所有 CGI 参数都可以在模板中替换引用，MapServer 特定参数以及用户定义的参数。原则上，参数由 MapServer 直接传递，不进行任何处理。此功能对于实现 MapServer 应用程序至关重要。

每个模板由传统的 HTML 标记，标记和特殊的 MapServer 替换字符串组成。例如，这些字符串用于指定 MapServer 创建的地图图像的路径，识别要渲染的图层，以及指定缩放级别和方向。

模板的占位字符串区分大小写。属性项替换必须与 DBase 文件中（对 Shapefile 而言）的项名称相同。ArcView 和 ArcInfo 通常会生成 dbase 文件，其项目名称全部为大写。当模板是 URL 时，应用适当的 URL 编码（如 ' '（空格）到 '+'）。一些占位字符串也可以以转义形式（如 URL 编码）提供。

模板只是 HTML 文件或 URL 字符串，其中包含每次处理模板时由 mapserv 程序替换的特殊字符。简单替换允许诸如活动层或空间范围之类的信息从用户传递到 mapserv 并再次传回。大多数情况下，新值被转储到将再次传递的表单变量中。下面给出了特殊字符和表单变量的列表。模板文件可以包含任何内容，包括 HTML、CSS、JavaScript 用其他 Web 对象。

在 HTML 文件中，属性值可以在引号 ("") 内。在引号内写入属性值允许设置值无法正常使用的特殊字符（如：]，=，" 和空格。要在属性值中使用单引号，只需使用两个引号 ("")。

#### 4.1.4 MapServer 初始化文件

MapServer 与所有的网络应用都相似，是基于无状态的协议。也就是在每次调用时，它仅知道浏览器传达给它的信息而不会知道上次被调用时的信息以及产生的响应。无状态协议排除了使用程序对最后一个问题之外的内容进行回答。然而，一些“聪明”的代码可以提供有状态的环境，从而使得网络应用程序有能力开展更复杂的工作。例如，通过隐藏形式的变量中存储的状态信息可以在页面跳转之间维持状态，在 URL 或者 cookie 中就可以存储这样的信息。但是有些方法需要对程序进行引导，所以它包含了第一次调用所需的信息。

在 MapServer 中，通过初始化文件来完成这项内容。在 CGI MapServer 应用程序中，初始化文件是传统的 HTML 表单，它把信息把硬编码转化为表单变量，来提供初始化的信息。几乎所有 MapServer 用到的值都可以在初始化文件中设置。

当 Apache 使用 HTML 的初始化表单第一次调用 mapserv 程序的时候，表单变量用来指定地图文件的名称（扩展名通常是 .map）。它随后找出文件的字体、符号、模板以及空间数据。Mapfile 同样指定了最后地图的大小、地理范围，以及地图是否是 GIF，JPEG 或者是 PNG 格式。读取完地图文件后，MapServer 会渲染一幅或多幅影像：地图本身，图例和比例尺，可能还会有参考地图。它会把这些影像存储在地图文件的指定位置中。很多信息都是 MapServer 的制图功能，但也有一些信息是 CGI 程序运行需要的。MapServer 用合适的值替代字符串并把修改的 HTML 返还给请求信息的浏览器。

为了展现结果，MapServer 需要设计地图格式并组织相关元素使之成为 Web 页面。程序本身并不会生成 HTML，相反，它会使用 HTML 模板并替换其中的字符串。字符串替代可以是参考文件、地图几何的细节信息、图层规范或者缩放因素等；它们也可以是 CGI 变量的常数，例如图像大小、地图文件的名称、地图范围等等。

#### 4.1.5 配置 MapServer 临时文件夹位置

IMAGEPATH '/data/tmp/'，本地绝对路径到 Web 访问的临时目录。运行 Web 服务器进程的用户应该能够写入到这个目录中。确保该路径包含在最后一个斜杠 (/)。（IMAGEPATH 可能是这个样子：/var/apache/htdocs/tmp/ 或 C:/Inetpub/wwwroot/tmp/）。

IMAGEURL '/tmp/'，这是怎么 IMAGEPATH 会出现相对 Web 服务器的根目录。如果必须输入完整的 URL，这将是 http://webgis.cn/ms\_tmp/。请确保路径包含在最后一个斜杠 (/)。

设置 MapServer 应用程序时，了解 ms\_tmp 文件夹的文件系统路径（不必将其称为 ms\_tmp；可以随意命名）与在线 URL 之间的区别非常重要。MapServer 需要知道可以在其上写入地图图像文件的位置。它还需要知道要发送回 Web 浏览器的 URL，以将地图图像加载到网页中。

通过将文件系统路径视为内部或专用文件夹（其中只有 Web 服务器可以创建文件，而 URL 作为公用文件夹，任何人都可以查看但不能查看文件的公共文件夹），可以通过两种不同的方式查找文件夹改变他们。

如果使用的是全新安装的 Apache，则需要在配置文件中手动设置临时位置和别名。有几种方法可以配置设置。本书会说明在 Debian Linux 上使用 Apache 2 的一种方法。可以在文件系统上的任何位置设置 MapServer 临时文件夹，将服务器中存储地图图片结果的文件夹设置为 IMAGEPATH "/owg/ms\_tmp/"，将 Web 访问的 URL 设置为 IMAGEURL "/ms\_tmp/"。

Mapfile 文件 <http://webgis.cn/mfc1.map> 中按如下声明：

```

10 WEB
11   TEMPLATE "example1-9.html"
12   IMAGEPATH "/owg/ms_tmp/"
13   IMAGEURL "/ms_tmp/"
14 END

```

在 Linux/Unix 操作系统中，必须正确设置 ms\_tmp 文件夹的权限，MapServer 才能保存地图图像。通过以下命令将 Web 服务器用户和组分配为文件夹的所有者：

```
# chown www-data.www-data /owg/ms_tmp/
```

这应该使 Web 服务器以及 MapServer 都可以访问它。在其他平台上，此过程将有所不同，具体取决于使用 Web 服务器的方式。

在 /owg/ms\_tmp/ 中创建自己的临时图像，这样，当 MapServer 设置为在 http://localhost/ms\_tmp 中查找时，它将能够找到它。在本地进行测试时，如果将图像 mymap.png 放在 /owg/ms\_tmp/ 中，则应该可以在 http://localhost/ms\_tmp/mymap.png 上看到它。

## 4.2 创建可交互地图：响应鼠标点击

本节中创建的绘图程序会比之前的更为复杂并能够生成简单的交互应用。

### 4.2.1 修改 Mapfile 定义 WEB 支持

要在 Web 中生成地图应用要样需要 Mapfile 的支持。需要进行下面的修改。

Mapfile 文件 <http://webgis.cn/mfc1.map> 包含以下几行：

```

10 WEB
11   TEMPLATE "tmpl-pan.html"
12   IMAGEPATH "/owg/ms_tmp/"

```

(下页继续)

(续上页)

```
13      IMAGEURL "/ms_tmp/"  
14      END
```

这是一个 Mapfile 使用 Web 功能支持的一个例子。网页对象由关键字 WEB 生成并由关键字 END 作为结束。

- 关键字 TEMPLATE 指定了 HTML 模板的名字，可以使用地图文件的相对路径或者是绝对路径。
- 关键字 IMAGEPATH 告诉 MAPSever 把生成的图像放在哪。这个路径是服务器上的本地路径。注意修改这个路径的权限。
- 关键字 IMAGEURL 指定了 URL 地址告诉浏览器到哪去查找地图。指定了 Web server 根目录的位置。这个是 URL 。

当 MapServer 被调用的时候，它读取地图文件并对地图进行渲染。它随后读取模板文件并把自身的信息插入到模板中，使用方括号括起来的替代字符串对其信息进行说明。

#### 4.2.2 创建初始化模板文件

为了展现 MapServer 生成的地图，需要把它嵌入到网络页面。通常的做法是生成一个模板，模板包括了显示图像所需的全部 HTML 标签，能够提供地图控制，并呈现 MapServer 生成的其它信息。但是，在完成这项工作前，还需要创建初始化文件。

初始化 HTML 文件有三个作用：

1. 当第一次被调用的时候，它会初始化 MapServer 应用程序。
2. 它会有效的设置地图的格式以及相关的信息。
3. 通过保存输入字段的参数可以保持状态。

注意：可能会认为，因为 MapServer 被初始化页面调用，它会知道 HTML 模板文件中使用的 HTML 模板。但是实际上并不是这样的。MapServer 被初始化页面调用，但是模板文件它并不知道它已被调用，它是无意识的。MapServer 可以假设在模板文件中插入的程序的名称是自己的名字，在大多数情况下也是这样的，但会有一些场景不希望这样的事情发生的时候。因此需要告诉 MapServer 表单下一次提交时，应调用什么程序模板文件。MapServer 的 CGI 变量 [program] 被解释为程序名称的替换字符串。

尽管初始化的任务可以由独立的网络页面完成，在许多情况下把初始值嵌入到模板文件并在随后的调用中更新这些值会更容易。本章会进行简化，将初始值嵌入到模板文件中。该文件将指定运行 CGI 程序的名称，使用 Mapfile 的名称，地图的原始范围，初始的缩放因子，首次调用显示的图层。

### 4.2.3 MapServer 的模板文件定义

模板文件由 HTML 文档指定。它设置了一些占位符，以获取 MapServer 创建的数据并将其嵌入到输出网页中。此页面中的 HTML 应该针对特定应用程序进行自定义。

#### 模板文件的基本要求

MapServer 的模板文件名后缀是有限制的，可用的后缀为：.gml，.html，.htm，.js，.kml，.svg，.tmpl，.wml，.xml。

MapServer 的模板文件在第一行必须包含特定的字符串 `<!-- MapServer Template -->`，以说明此文件为 Mapfile 所使用的模板文件。一般使用 HTML，JavaSript 或 XML 注释的形式给出。此行经 MapServer CGI 处理后不会传递到客户端。这个特定字符串不区分大小写。

如果没有这个字符串的话，在使用 Web 访问，或使用 GetFeatureInfo（带有 info\_format 集合）作为 text/html 请求时会提示：

```
Content-type: text/html isValidTemplate(): Web application error. Missing
magic string,
hello.html doesn't look like a MapServer template.
```

#### MapServer 模板文件中的表单

生成动态地图需要 HTML 模板中有一个表单用来与用户进行交互。

一个简单的表单如下：

```
17  <form name="mapserv" method="GET" action="/cgi-bin/mapserv"
18    class="form-horizontal">
19      <input type="hidden" name="map" value="[map]">
20      <input type="hidden" name="imgext" value="[mapext]">
21      <input type="hidden" name="imgxy" value="[center]">
22      <input type="hidden" name="zoom" value="0">
23      <input type="hidden" name="layer" value="topo">
24      <input type="hidden" name="layer" value="states_line">
25      <input type="hidden" name="mode" value="browse">
26      <input type="image" name="img" src=".../[img]"
27        width="600" height="300" border="1">
28  </form>
```

如果熟悉 HTML 表单和标签，则可能会注意到此页面上唯一可见的对象是标题，标题和图像。生成页面时，MapServer 会实时填充图像源（SRC）。

<form> 为表单标签，method 为响应方法，action 定义响应程序的链接。<input> 定义了多个输入内容。一个 MapServer HTML 模板实际上是一个有几个特定 MapServer 标记 (tag) 的 HTML 文件，MapServer 标记是以方括号 ([] ) 括起来的 MapServer CGI 变量，如 [map] 、[img] 和 [mapext] 等，它们分别代表地图文件、地图图片 URL 和地图的范围。通过 HTML 模板，可以大量简化 MapServer CGI 编程；现代的 Web 开发中的模板也都采用这种技术。

## 表单中的 CGI 变量

CGI 应用程序的运行方式类似于中介。MapServer CGI 应用程序接收对地图的请求，包括地理范围，要显示的图层等信息。然后，它们返回显示所请求地图内容的网页。如果了解如何手动形成适当的请求，则可以将一个 URL 组合在一起，以从 MapServer CGI 请求地图。发出初始请求后，所有设置和请求信息都存储在 URL 中，并在进一步请求时发送回 CGI。对于这样的应用，模板文件对于应用程序不是必不可少的，但是确实使它入门变得容易得多。

MapServer CGI 应用程序向查看器发送一个新的网页。该网页的 URL 包含多个变量。当用户请求更改地图时，它会将这些变量发送回 CGI 进行重新处理。初始化页面的目的是使整个过程开始，而不会强迫使用外观复杂的初始 URL。

MapServer 的 Mapfile 提供了一些变量。几个常用的 CGI 变量设计作为地图界面初始化的一个部分，但是实际上所有 Mapfile 参数都能够被定义为变量。

在这个文件中，关键是表单中的定义。注意 MapServer 理解这个文件中使用的每一个 CGI 变量的名称。这是 HTML 和 Mapfile 交流的方式，因此，拼写很重要，而不是大小写。

### 4.2.4 运行查看结果

当 MapServer CGI 程序处理应用时，它首先解析参数和地图文件，并产生必要的输出。这些输出需要写在 HTML 模板文件中，必须在 Mapfile 中定义，通过使用网络模板关键字（或者在一个单独的 HTML 初始化文件中）。CGI 程序在将它发送到 Web 浏览器之前，将用适当的值取代所有在 HTML 模板中的变量。如果直接在网页浏览器上查看 HTML 模板的内容（没有经过 CGI 程序处理），将不会有任何地图提供，反而会有空的图像甚至其它不明意义的字符。

#### 查看 Web 页面

在浏览器中输入 <http://webgis.cn/349> 并按下回车。在扫描和替代完所有的可替代字符串后，MapServer 把模板文件的内容（包括修改后的 09 行）传给 Apache 服务器，服务器再把内容传送给浏览器。图像从指定的 URL 中重新生成，浏览器接收解析结果并呈现出来。

MapServer 现在读取了之前创立的 <http://webgis.cn/mfc1.map> 地图文件。MapServer 检查地图文件中创建的要素是否在定义的范围内。如果在范围内，它就会渲染并生成地图。

在页面中查看源代码，内容如下：

```

16   <form name="mapserv" method="GET" action="/cgi-bin/mapserv" class=
17     ↪"form-horizontal">
18     <input type="hidden" name="map" value="/owg/mfc1.map">
19     <input type="hidden" name="imgext" value="-180.301003 -90.000000_
20     ↪180.301003 90.000000">
21     <input type="hidden" name="imgxy" value="300.0 150.0">
22     <input type="hidden" name="zoom" value="0">
23     <input type="hidden" name="layer" value="topo">
24     <input type="hidden" name="layer" value="states_line">
25     <input type="hidden" name="mode" value="browse">
26     <input type="image" name="img" src="..//ms_tmp/
      ↪mapv1915797771279137.png"
          width="600" height="300" border="1">
    </form>

```

在返回页面的源代码中会发现相应的变量被替换。

MapServer 会给图像命名，图像名称综合了地图文件名（例如，Hello\_World），系统生成的字符串编码，以及图像类型。

字符串 [img] 被当做替代字符串可以扩展为图像的 URL。MapServer 知道地图文件中 IMAGEURL 给出了基本的 URL，因此在第 26 行中它用生成的图片路径替代字符串 [img] 得到目前的结果。

本阶段常见的错误可能是：给之前创建的目录设置了不正确的权限，地图文件或模板文件中的错误，或者是忘记了渲染 PNG 图像的资源。注意的是 MapServer 会对指定的地理范围进行严格地检查。如果要素的位置为 0.0, 0.0 没有在地图定义的范围之内，那么该点就不会显示。

## 鼠标点击事件的请求和响应

在地图上用鼠标点击，这一次 Apache 调用可执行的 MapServer 并向它传递第 21 行的变量。它将单击位置设置为下一幅地图的中心，然后更新地图。待网页中的图像更新后，可以再次单击。

点击之后，会发现地图点击的位置变成了现在地图图片的中心。网络页面仍然与原来一样，但是图片变化了。如果一切配置和输入都是正确的，将会看到彩色的矩形，宽为 600 个像素高为 300 像素。

在打开的地图页面中，使用鼠标点击地图，浏览器会获得鼠标的活动位置以及状态，一旦获取到鼠标的点击事件发生就会响应，获取所在的点的地址，连接到标签内的 Url 地址，请求新的内容。将 URL 拆分。

观察 URL 每次的变化，这里给出一个示例 <http://webgis.cn/923>，URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]  
0 http://webgis.cn/cgi-bin/mapserv  
1 map=/owg/mfc1.map  
2 imgext=-77.658863+-140.267559+282.943144+39.732441  
3 imgxy=300.0+150.0  
4 zoom=0  
5 layer=topo  
6 layer=states_line  
7 mode=browse  
8 img.x=342  
9 img.y=119
```

新的 URL 会将 IMGEXT 的值替换掉换来的值，再发起请求。

## 4.3 MapServer 地图的缩放控制

电子地图应该可以进行互动，如放大、缩小。对于地图来讲，不同的比例尺下展示的细节需要不同。

### 4.3.1 控制比例尺优化地图内容

由于地图的主要目的毕竟是有效地呈现信息，除了如缩放，平移，选层等不同的互动功能，地图应该在视觉上有吸引力。

大量的空间信息可以被压缩到任何地图，但超过某个程度后信息是不可识别或读取的，因为地图太密集了。另一方面，非常稀疏的地图是不实用的，因为显示的信息密度低。作为地图和地图服务的开发或运维人员，目标是要在地图上提出足够的信息，这样观者既方便也不混淆。可以放大的地图会期望放大后看到更多的细节。

比例尺定义为图上距离与实际地面距离之比。在纸质地图，必须考虑特定比例尺中要到达的内容。然而，随着数字地图的出现，应该随着地图比例尺的不同而绘制不同的内容。在小比例尺地图中会忽略很多的要素，这些要素可能包括村庄，当地的道路，或小湖泊；但在大比例尺地图上忽略特征通常没有任何意义。除了包括或省略基于地图的比例尺的特征，还可以改变用于绘制特定特征的符号，改变的符号的大小和颜色，甚至可以省略或代替全层（用面状要素代替点状要素，如城市）。

### 4.3.2 在 Mapfile 中定义比例尺可见内容

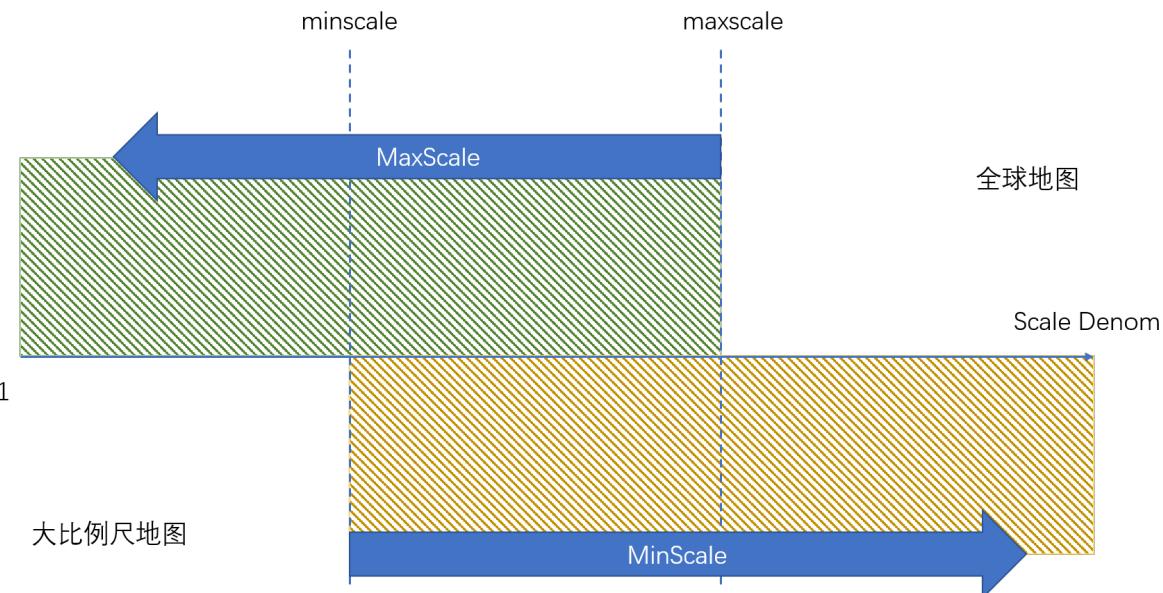


图 4.2: MapServer 比例尺视图关系图

图层水平的关键字 MAXSCALEDENOM 设置最大比例尺分母（最小比例尺），大于等于这个比例尺的地图上将会展示出此图层。同样地，关键字 LABELMAXSCALEDENOM 用来设置与比例尺相关的标注。关键词 MAXSCALEDENOM、MINSCALEDENOM、LABELMAXSCALEDENOM 与 LABELMINSCALEDENOM 被称为比例尺关键词。

MAXSCALEDENOM 与 LABELMAXSCALEDENOM 这两个参数的值不必相同的，并且可能不应该相同。考虑一下随着比例尺的减小，地图会发生什么。以公路要素及标注为例，超过一定比例尺之后，名字标注变得过大，展示出来的只是杂乱地图；然而，在这样的比例尺地图中依然要绘制公路要素，因为展示了如何连接不同的地方（这是比他们的名字更重要的）。

除了在图层对象中，比例尺关键词也可以用到 CLASS 对象中。实例即以 CLASS 对象进行说明。在 Mapfile 文件 <http://webgis.cn/mfca.map> 中，内容如下：

```

44 CLASS
45   NAME "National and provincial capital"
46   EXPRESSION "National and provincial capital"
47   MAXSCALEDENOM 1800
48   SYMBOL "circle"
49   SIZE 4
50   COLOR 255 0 0
51 END
52 CLASS
53   NAME "National capital"
54   EXPRESSION "National capital"
55   MAXSCALEDENOM 1000

```

(下页继续)

(续上页)

```

56 SYMBOL "circle"
57 SIZE 4
58 COLOR 255 204 102
59 END
60 CLASS
61   NAME "Provincial capital"
62   EXPRESSION "Provincial capital"
63   MAXSCALEDENOM 600
64   SYMBOL "circle"
65   SIZE 4
66   COLOR 51 102 255
67 END
68 CLASS
69   NAME "States"
70   MAXSCALEDENOM 400
71   SYMBOL "circle"
72   SIZE 4
73   COLOR 100 100 100
74 END

```

在命名为 National and provincial capital 的类别中，MAXSCALEDENOM 设置为 1800；在 National capital 的类别中，MAXSCALEDENOM 设置为 1000；其他的类别中按类似的方式定义。

### 4.3.3 控制 MapServer 的放大、缩小

现在的 HTML 模板。它比 节 4.2 应用程序或第一张地图的模板更复杂，因为它必须能够做得更多。因为更多的元素在屏幕上的显示，将需要更多的代码。此外，有几个替换字符串，会从浏览器来操控地图。

这里来看一下例子，<http://webgis.cn/983>。这个例子实现了地图缩放的效果，主要的不同在于表单。在“地图控制”的控件中有其他的一些选项，可以选择“放大”、“缩小”及相应的比率。

```

40 <select name="zoom" class="form-control">
41   <option value="4" [zoom_4_select]>放大 4 倍</option>
42   <option value="3" [zoom_3_select]>放大 3 倍</option>
43   <option value="2" [zoom_2_select]>放大 2 倍</option>
44   <option value="0" [zoom_0_select]>置中</option>
45   <option value="-2" [zoom_-2_select]>缩小 2 倍</option>
46   <option value="-3" [zoom_-3_select]>缩小 3 倍</option>
47   <option value="-4" [zoom_-4_select]>缩小 4 倍</option>
48 </select>

```

从第 49 行开始了 SELECT 控件，名称的 NAME 属性为 zoom，表示用来控制地图的缩放。在 OPTION 标签中的属性 value，正值表示放大、负值表示缩小，值为 0 时无缩放。注意 OPTION 标签中的类似 [zoom\_4\_select] 的属性，这个属性用来表示控件中当前选中的状态，通过这种方式可以这次的选项带到下一次中。

另外在这个页面中还添加的地图模式的选项，这样，用户可以在表单中选择不同的模式，得到不同的结果。而不必像编程人员一样需要通过繁琐的编码方式来构建不同的输出方式。

```
31 <select name="mode" class="form-control">
32   <option value="browse">Browse</option>
33   <option value="map">Map</option>
34 </select>
```

在 <http://webgis.cn/983> 中，打开后一张全球地图。在“地图控制”的选择“放大 2 倍”，然后点击“刷新”，或者在地图中用鼠标点击，随着地图的放大，可以看到更多级别（不同颜色）的城市展示出来了。

可以进一步选择不同的缩放方式及因子来了解 MapServer 中控制地图缩放的方式。

## 4.4 MapServer GGI 图层开关控制

传统的地形也是有层次的，但是不管多少层次，都绘制在一张硬拷贝的介质上。而电子地图则可以更方便地进行图层控制，可以选择显示或不显示某个/些图层，从而得到不同的结果。这个特点也是电子地图与传统地图的区别，同样也需要对图层进行仔细的组织。

### 4.4.1 控制图层的方法

MapServer 能够实现针对图层的控制，这样可以决定绘图时不同图层的打开与关闭。Mapfile 中定义不同图层的状态，这些状态决定了结果中地图包含有哪些图层。同时，通过 CGI 程序可以控制 Mapfile 的行为；在服务器端根据图层的状态，MapServer 程序会生成不同的地图结果。

这个示例中的 Mapfile 文件 <http://webgis.cn/mfu3.map> 定义了多个图层，这里没有引入新的概念，代码就不展示了。可以查看一下定义的每个图层的信息。

### 4.4.2 HTML 模板文件

在这个示例中，将图层控制的功能分成两个部分，一部分是通过 CHECKBOX 组来实现。通过选择想要显示的图层，可以简单、直观的选择要显示的图层。

```
26 <div class="form-group">
27   <label class="col-sm-2 control-label">选择图层:</label>
28   <div class="col-sm-3">
```

(下页继续)

(续上页)

```

29      <input type="checkbox" name="layer" value="states_line" [states_
30      ↪line_check]>
31          国家与地区界线
32      </div>
33      <div class="col-sm-3">
34          <input type="checkbox" name="layer" value="wriver" [wriver_check]>
35          河流
36      </div>
37      <div class="col-sm-3">
38          <input type="checkbox" name="layer" value="wroads" [wroads_check]>
39          道路
40      </div>
</div>

```

这个应用定义了三个复选框。每一个复选框中有一个图层的名称，如果一层要被渲染，那么与该层相关的复选框必须进行选中（复选框有一个值等于该层的名称）。

在每一行（第 29、33、37 行）中定义的属性如下：

- `type="checkbox"` 定义类型为复选框；
- `name="layer"` 定义名称为 `layer`；
- `value="the_val"` 定义图层的名称 `the_val`，由 `Mapfile` 中的图层名称确定。

`Mapfile` 的每一层都将有一个与它的名字相关的替换字符串，这是通过在图层名称上追加一些字符（`_check`）。这些字符串可以有两个值：`checked` 或 `" "`（空），就像在单选按钮的情况下。当 `MapServer` 扫描模板，它用适当的值取代了复选框替换字符串。

第二部分通过 `SELECT` 组来实现了地图背景的选择。地图的背景是相互覆盖的，所以同一时间只能显示一幅。但是 `Mapfile` 无法判断，在制图的时候可能会让两幅都显示，但结果只会显示后定义的一幅。通过前端的逻辑，通过 CGI 程序，可以选择不同的地图背景。

代码与前面类似，只是在判断是否选中处，在图层名称上追加的字符变成（`_select`）。

```

41 <div class="form-group">
42     <label class="col-sm-2 control-label">选择背景:</label>
43     <div class="col-sm-8">
44         <select name="layer" class="form-control">
45             <option value=" " [_select]>无背景</option>
46             <option value="land" [land_select]>
47                 全球遥感影像
48             </option>
49             <option value="topo" [topo_select]>
50                 全球地形

```

(下页继续)

(续上页)

```

51      </option>
52    </select>
53  </div>
54</div>
```

#### 4.4.3 地图控制说明

打开页面 <http://webgis.cn/726>，在页面中对控件进行选择，以查看不同的效果。

控制图层的相关信息都写到 HTML 模板中，所有的信息同样通过 CGI 程序传递给 MapServer，然后 MapServer 再根据不同的参数，来渲染结果 图 4.3：

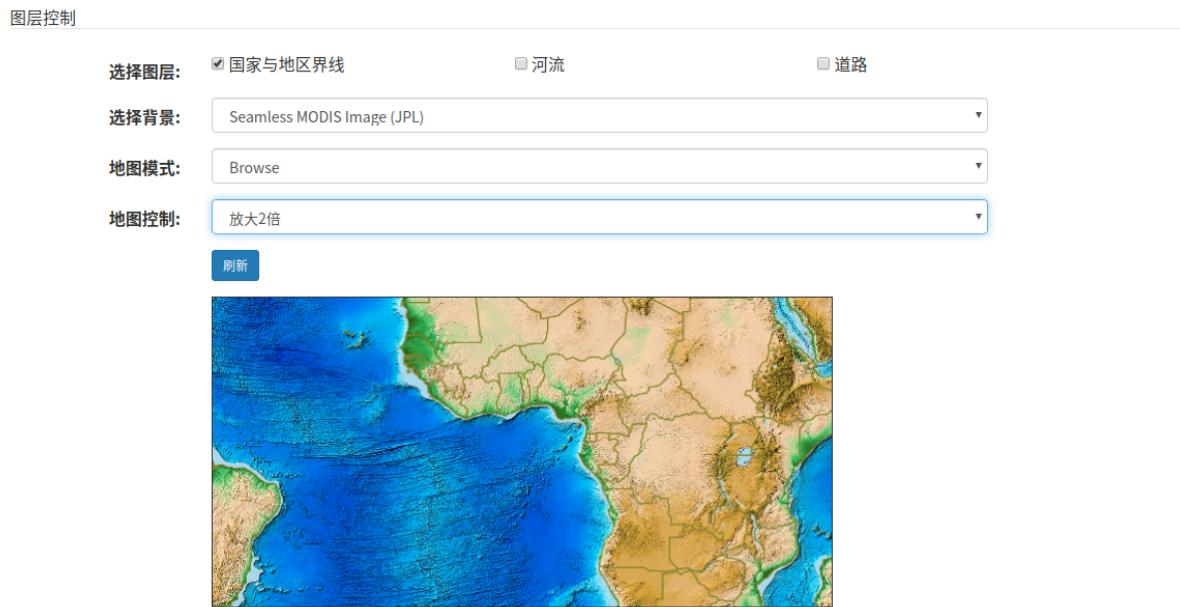


图 4.3：控制图层操作

虽然这个例子中在前端定义了“选择图层”与“背景图层”两组图层，“选择图层”还是由 3 个独立图层构成，但是渲染的结果只是一张图。

#### 4.5 使用 MapServer 参照地图

鸟瞰图是过去 WebGIS 常见的一项功能，在 MapServer 中，使用参照地图（Reference Map）这个概念。参照地图是与地图同步显示的图片，这个图片的内容是固定的，但是在参照地图中会标识出地图浏览的位置与范围。

MapServer 支持三种类型的参照地图。最常见的是显示交互式界面中的地图范围。

另外也可以请求参照地图作为查询的一部分：点查询将生成带有放置在查询点的标记的图像，基于地区的查询将描述感兴趣区域的范围；最后，基于特征的查询将显示所使用的选

征。这一部分与 MapServer 的查询功能有关，在这一节中不会涉及。

#### 4.5.1 使用参照地图

在大比例尺地图中快速移动到其他位置是困难的。通过允许使用参考图，MapServer 的解决了这个问题，它显示了当前视图，用矩形框或 Marker 同步示意地图位置与范围。参照地图可以是任何大小，一般情况下都会比地图小很多。

参照地图开始于关键字 REFERENCE，终止于关键字 END。一个参照地图需要一个图像作为地图。图像通过关键字 IMAGE 被确定，其值是参照地图的路径（Mapfile 的绝对的或相对的位置）。参照地图的格式原来要求为 GIF 格式，但现在已经没有这个限制。

以下是一个添加了参照地图的 Mapfile <http://webgis.cn/mfk2.map>，主要内容变动如下：

```

29  REFERENCE
30    IMAGE ".../images/worldref.png"
31    SIZE 300 150
32    EXTENT -180 -90 180 90
33    STATUS ON
34    MINBOXSIZE 30
35    MAXBOXSIZE 150
36    COLOR 255 255 0
37    OUTLINECOLOR 255 50 50
38    MARKERSIZE 30
39    MARKER "star"
40  END

```

此应用程序的参考图被定义在 29 行至 40 行。参照地图是 worldref.png。如果是要生成一个参照地图，那么 STATUS 值的关键字状态设置为 ON；如果不是，它设置为 OFF。

参照地图的大小由关键字 SIZE（以像素为单位）指定。原始的图片不必是这个大小，但是应保持比例一致，避免出现变形。这个尺寸与用户点击时的图像坐标相关。

EXTENT [minx] [miny] [maxx] [maxy] 设置参照地图的空间范围。这个一般情况下应与地图对象中同名关键词的设置一致。

COLOR 是指示框的填充色，设置为黄色。在简单的背景上可以设置为无填充颜色，通过将关键字 COLOR 中的至少一个的组成部分设置为 -1。

OUTLINECOLOR [r] [g] [b] | [十六进制字符串] 用于设置指示框的颜色。将任何组件设置为 -1 表示没有轮廓。这里设置为红色。

第 34、35 行定义了指示框的最小尺寸与最大尺寸。指示框是展示在参照地图上的矩形框，与地图的范围一致。尺寸大小的设置请参考定义的 SIZE，最大尺寸应比 SIZE 小。

第 38、39 行定义了 **Marker**，使用了预定义的符号 `star`。当地图放大到一定程度，使用矩形来示意范围大小以致于难以观察时，可以使用此处定义的符号来表示位置。

---

注解：MapServer 本身可以用于创建参照地图。命令行工具 `shp2img` 可以做到这一点。

---

## 4.5.2 HTML 页面的定义

与图层控制相同的是有放置地图的容器，不过需要多加一个控制容器来显示参照地图。参照地图的替代关键词为 `[ref]`。

```

91 <div class="form-group">
92   <label class="col-sm-2 control-label"> 参照地图: </label>
93   <div class="col-sm-8">
94     <input type="image" name="ref" src="../[ref]" border="1">
95   </div>
96 </div>
```

注意第 94 行定义的参照地图对象，不要定义其 `width` 与 `height` 属性。在进行位置与范围控制时，MapServer 需要计算地图与参照地图的大小、坐标，这些信息都是在 `Mapfile` 中定义的。如果在 `HTML` 中修改了参照地图的大小，则会导致传递的信息出现偏差。MapServer CGI 是基于 `HTTP` 的无状态协议传递信息，它无法感知前端网页对参照地图的修改。

## 4.5.3 实例说明

这里使用实例：<http://webgis.cn/362>，URL 解析为：

```

>>> [print(i,p) for i,p in enumerate(re.split('(?: )+',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfk2.map
2 layer=states_line
3 zoom=0
4 mode=browse
5 root=/mstuto
6 program=/cgi-bin/mapserv
7 map_web=template+tmpl-k2.html
```

打开链接后会显示出来参照地图，但是并没有指示框；当地图开始放大时，矩形框会出现（如 [图 4.4](#)），标识地图的位置与大小。通过指定参考地图中参考框的大小和位置，MapServer 可以以适当的范围和图像大小绘制地图。在参照地图中点击不同的位置，将导致地图的跳转。

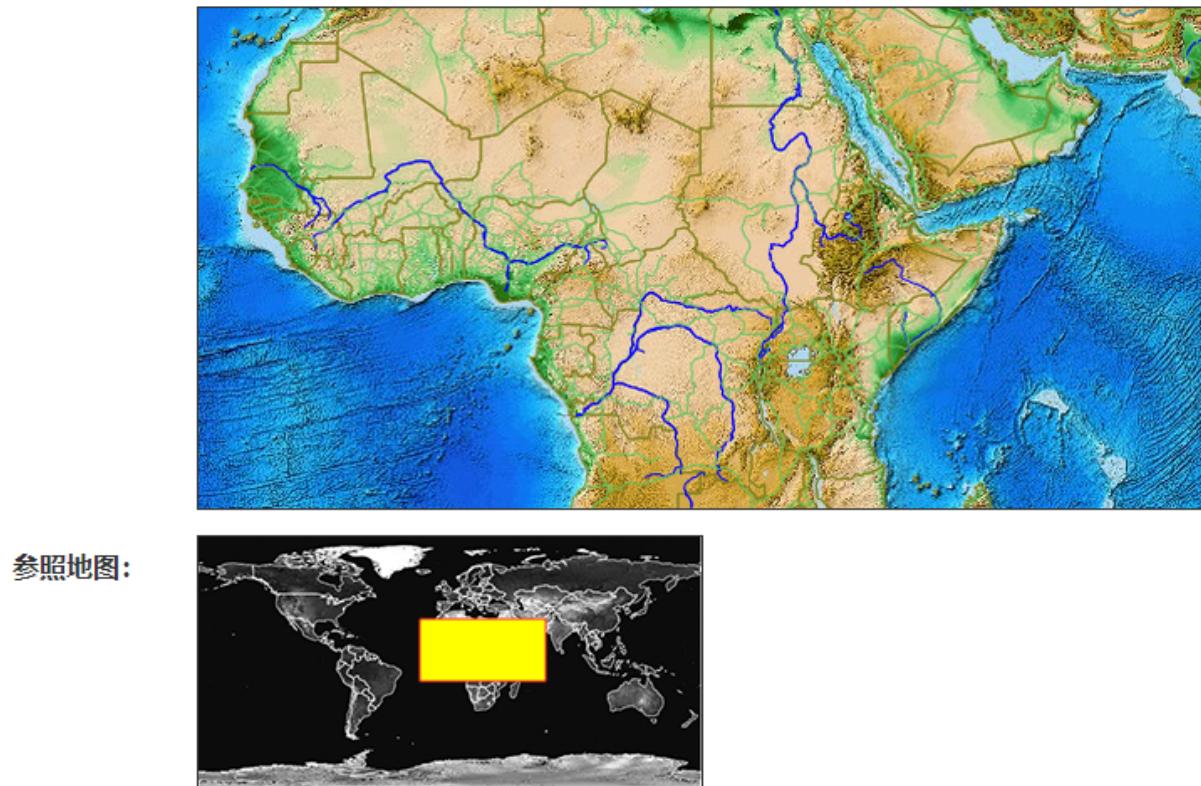


图 4.4：参照地图：矩形符号

继续放大地图，当指示框的尺寸小于设置的 MINBOXSIZE 值时，则指标框会消失，同时星形的点状指示符号出现（如 图 4.5）。点击参照地图同样会导致地图跳转。

## 4.6 使用独立比例尺与图例对象

MapServer 的比例尺与图例在 节 2.3 已经说明过，是作为地图的一部分嵌入在地图中的。在 WebGIS 中比例尺与图例可以与地图分开，这样创建应用程序更加方便。比例尺在地图应用交互性与可用性发展中起着如此重要的作用，应该始终将地图的比例尺（使用替换字符串 [scale]）在网页上显示出来。

比例尺与图例图像可以单独使用，或它们可以嵌入在地图图像中。比例尺与图例图片的大小根据地图的内容确定，不能在创建之前确定，所以独立使用的情况下应该避免在模板代码的 <img> 标记中指定这两个对象的高度与宽度。

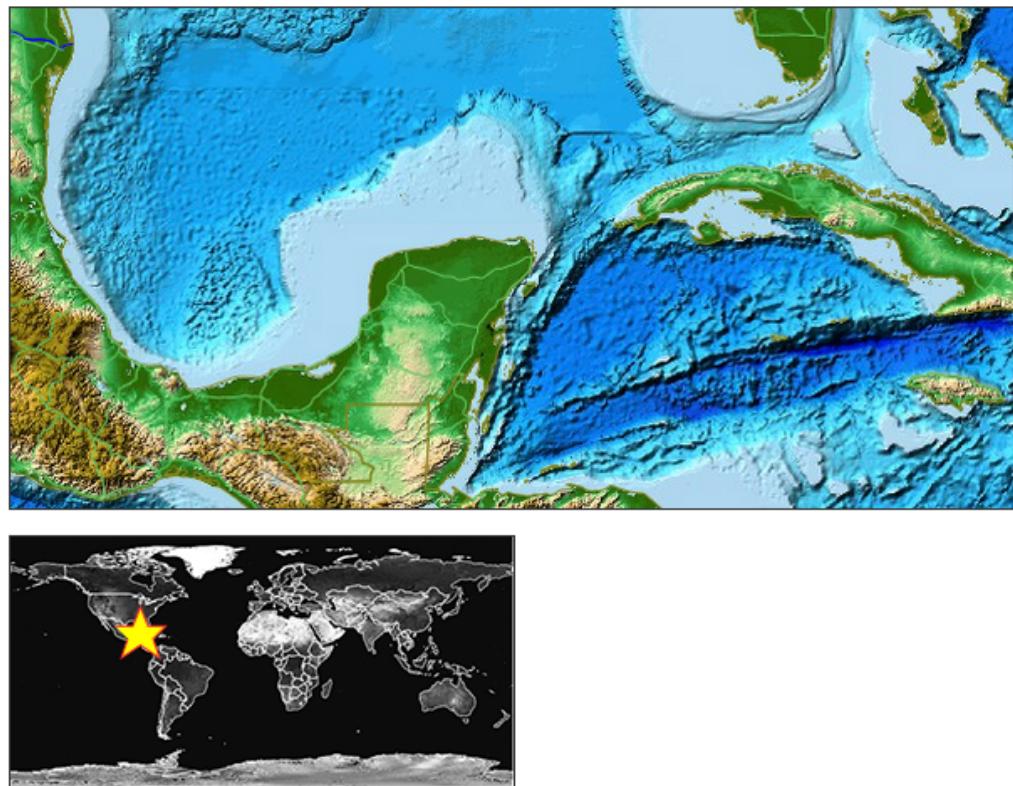


图 4.5：参照地图：点状符号

#### 4.6.1 比例尺控件的使用

主要实现的 Mapfile 文件 <http://webgis.cn/mfv9.map> 内容与前例区别如下：

21 STATUS ON

STATUS 是否要创建比例尺图像，默認為 OFF。嵌入的情况下设置为 EMBED，此处设置为 ON。

模板文件 <http://webgis.cn/tmpl-v9.html> 的主要代码为：

91 

请求网址为 <http://webgis.cn/249>，URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 root=%2Foswebgis
2 program=%2Fcgi-bin%2Fmapserv
3 map=%2Fowg%2Fmfv9.map
4 layer=states_line
5 zoom=1
```

(下页继续)

(续上页)

```
6 map_web=template+tmpl-v9.html
```

比例尺示例（图 4.6）见：

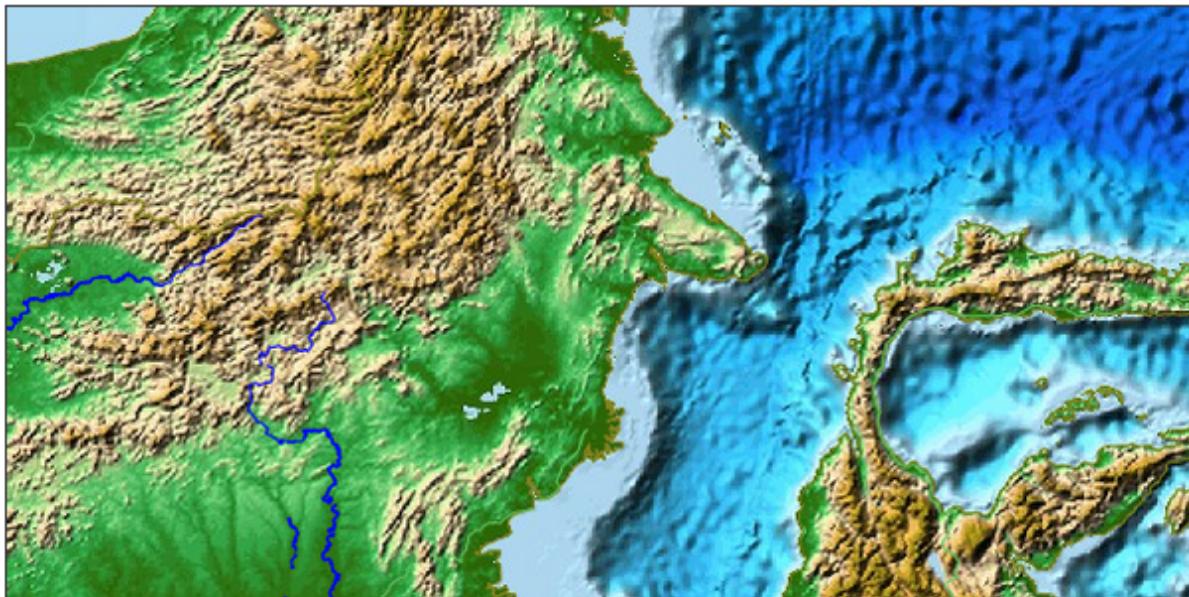


图 4.6：地图比例尺

#### 4.6.2 图例控件的使用

图例对象的使用与比例尺对象类似。

示例使用的 Mapfile 如下：

```
16 STATUS ON
```

STATUS [on | off | embed] 确定要创建图例图像，同样设置为 ON。

模板文件 <http://webgis.cn tmpl-x8.html> 的主要代码为：

```
88 
89 
```

示例（图 4.7）见：<http://webgis.cn/634>，URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split(' [&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 root=%2Foswebgis
```

(下页继续)

(续上页)

```

2 program=%2Fcgi-bin%2Fmapserv
3 map=%2Fowg%2Fmfx8.map
4 layer=states_line
5 zoom=1
6 map_web=template+tmpl-x8.html

```

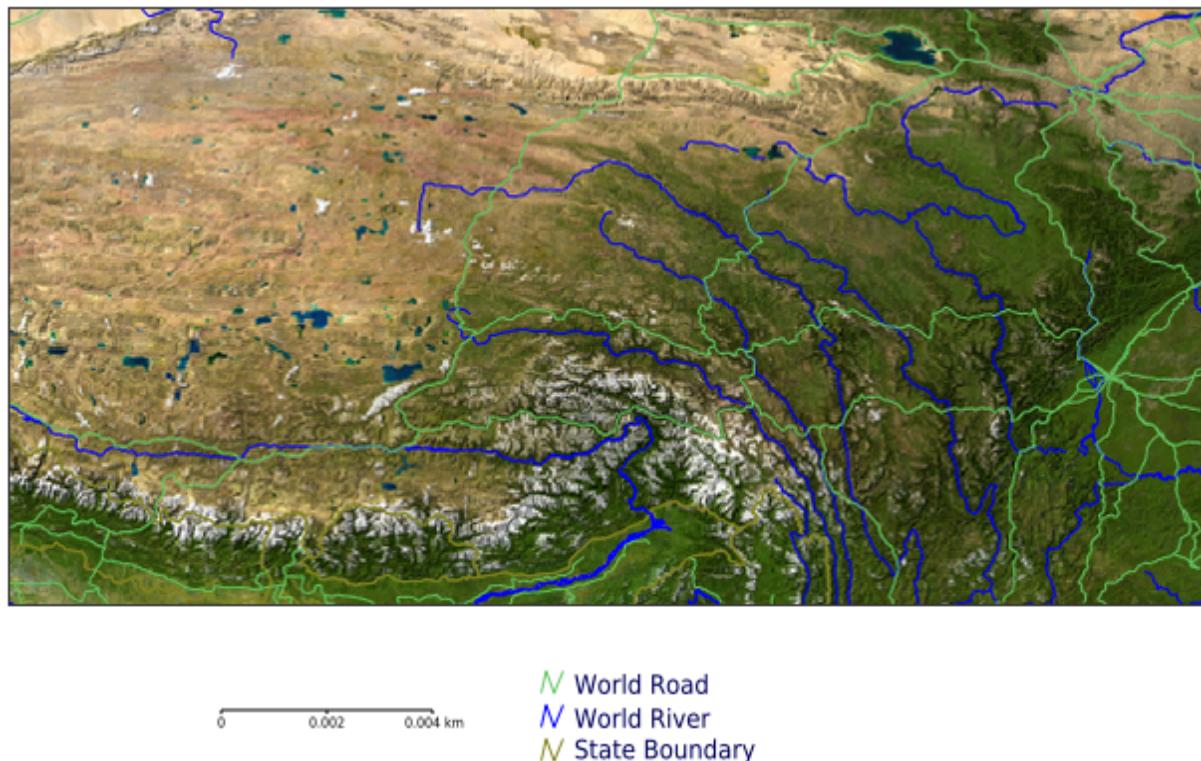


图 4.7：地图图例

## 4.7 在 MapServer 中进行属性查询

到现在为止，已经创建的地图都在 map 或 browse 模式下产生的。在这种模式下，当用户单击地图图像时，浏览器会将点击的位置、缩放大小和方向发送到服务器，后者再将它转发给 MapServer 程序。基于从浏览器的响应，MapServer 呈现出地图图像，扫描模板替换字符串，并将此信息发送回浏览器。所以，尽管 MapServer 可以访问底层的空间数据和属性数据，但该数据是对用户隐藏的。

MapServer 提供了强大的空间和属性查询功能。在 CGI 模式下，广泛使用 HTML 模板来构建查询和组织查询返回的信息，这通常涉及与地图文件、模板和 MapServerCGI 程序间复杂的交互。用户可以访问此信息使用的 MapServer 查询功能，它可以以表格的形式和以图形方式作为地图来显示结果。

在本节中，将探索 MapServer 的查询功能，并熟悉其 Mapfile 和模板的要求。要获得这些功能的知识，需要构建一个查询应用程序，要使用简单的 MapServer 的查询功能。MapServer 的查

询功能是一个复杂的话题，但是考虑到现代地图一般通过地图服务形式给出，这种传统的应用方式越来越少。因此只用一节的篇幅进行介绍。但为了尽可能说明功能，本节不再循序渐进，而是给出一个稍显复杂的案例。

### 4.7.1 MapServer 中查询功能

浏览模式是 MapServer 的默认模式 (`mode=browse`)，正因为如此，先前已经创建的地图都没有特别说明模式。但是 MapServer 有很多模式，其中有 18 种是不同的查询方式。在这些模式中，用户可以定义查询的选择标准，通过以下方法之一：

- 通过用鼠标选择地图影像的点或区域；
- 通过输入点或区域的坐标；
- 通过输入属性标准的表达式；
- 通过输入特征的索引号（例如，序列号）。

在查询模式中，浏览器将坐标或表达式和其他形式变量返回到 MapServer，而渲染地图图像，因为它会在浏览模式中。MapServer 搜索一个或多个图层并填充模板文件中的信息，该信息来自与特征或属性相匹配的查询参数。MapServer 可以执行空间查询（基于坐标）和属性的查询（根据属性值）。

当执行最简单的空间查询时，MapServer 注意到鼠标点击的位置，并选择基于点击点附近的匹配特征。但是，MapServer 不局限于点查询，可以提供空间范围（甚至是任意形状）来代替点，将返回指定范围（或形状）附近或之内的特征。在属性查询模式中，MapServer 查询能够匹配用户指定表达式的特征的属性表。MapServer 也支持分级查询，其中空间查询根据属性搜索返回的是结果集。

### 4.7.2 MapServer 案例

由于查询结果返回，MapServer 将它们插入到一个或多个 HTML 模板文件中，并将其转发到浏览器。MapServer 事先不知道任何查询的结果集可能会是什么，MapServer 必须构建其输出，以足够通用的方式处理这两种情况。为此，它通过指定查询模板的层次结构。

可以在地图级别、图层级别、类级别和连接级别定义查询模版。模板指定在哪个级别影响如何以及何时 MapServer 将使用它。在地图层面，一个查询模板用于呈现整个查询的摘要（例如，类似于匹配总数和与匹配图层的数目这样的信息）。查询模板在指定图层级别时，执行类似的功能，但在这种情况下，他们给特定图层的摘要信息。定义类级别模板用来呈现个别要素属性（例如，一个国家的位置，人口，国花）的结果集的元素。（或者，而不是指定多个类级别的模板，可以指定一个图层级别的模板，用来报告图层中的所有类级别的详细结果。）最后，如果在空间数据集和外部表之间定义一个一到多的连接，连接的每个匹配元素的结果将使用连接级别的模板来报告。

这里使用的示例为说明；展示一个地图中使用头部、底部模板的示例。使用的 Mapfile 为 <http://webgis.cn/mfpq.map>。示例请查看 <http://webgis.cn/592>。

地图模板文件为 <http://webgis.cn/tmpl-pq.html>，这里面定义了本案例不同的模式：

```

57 <div class="form-group">
58   <label class="col-sm-2 control-label">地图模式:</label>
59   <div class="col-sm-8">
60     <select name="mode" class="form-control">
61       <option value="browse" [browse_select]>
62         Browse
63       </option>
64       <option value="query" [query_select]>
65         Query
66       </option>
67       <option value="nquery" [nquery_select]>
68         Query Multiple Layers
69       </option>
70     </select>
71   </div>
72 </div>
```

MapServer 可以执行不同的查询模式。一些模式只会查询单个图层和返回单个结果——例如，QUERY 和 ITEMQUERY 模式。其他的，例如 NQUERY 和 ITEMNQUERY 模式，搜索所有可查询的图层和返回所有的结果（所有这些类型在名称中都包含 NQUERY）。这些查询可以通过设置可变 `qlayer` 方式（例如，查询图层）的值成为图层的名称可以被查询，以限定到单个图层——如果不这样做，MapServer 将搜索所有可查询图层。在 Mapfile 指定的顺序中搜索图层。此外，该特征查询模式使用一个选择图层（必须是多边形类型）指定可变 `qlayer` 方式。在选择图层上进行空间查询，并返回一个或多个多边形要素。随后，搜索其他可查询图层，选定的多边形内（或接近）被发现的任何要素将被返回。

在展示检索结果模板方面，Mapfile 首先在 MAP 对象中定义了 HEADER 关键词，与 FOOTER 关键词。

```

62 HEADER "pq-header.html"
63 FOOTER "pq-footer.html"
```

这两部分，分别用于输出检索结果的详情，与检索结果的地图。为了说明问题，本节在模板中进行了如此定义，但是实际上并无对应的关系。

查询模版是 HTML 一部分，必须被组合起来提供一个完整的网络页面。

### 4.7.3 单图层检索模式

在国家与区域图层中，定义如下：

```

101 LAYER
102     NAME "states_poly"
103     GROUP "states"
104     DATA "wcountry.shp"
105     STATUS ON
106     TYPE POLYGON
107     LABELITEM "NAME"
108     TOLERANCE 3
109     CLASS
110         NAME "the States"
111         TEMPLATE "c_query.html"
112         STYLE
113             COLOR 232 232 232
114             OUTLINECOLOR 132 132 32
115     END
116 END
117 END

```

为了执行 MapServer 的查询，Mapfile 中必须至少包含一个可查询图层。如果一个图层是活动的（即，其状态 ON 或 DEFAULT），并且包含一个模板（LAYER 级别或 CLASS 级别指定了关键字 TEMPLATE），那么该图层是可查询的。如果可查询图层存在，MapServer 为那些符合特定空间或属性标准的逐功能地搜索数据集。如果存在模板呈现出匹配特征，该特征被纳入结果集。一个 LAYER 级别模板的存在，保证每一个符合搜索条件的图层中的特征将被选中；另一方面，如果没有指定一个图层模板，没有模板的类别将不会返回任何结果。

c\_query.html 文件用来渲染结果，根据矢量数据的字段。

国家与区域图层的检索模板对应的内容如下：

```

14 <!-- MapServer Template -->
15 <h2> 图层检索结果：国家与地区 </h2>
16 <table class="table table-bordered">
17     <tr bgcolor="#CCCCCC">
18         <td bgcolor="#ffffff"> </td>
19         <th>NAME</th>
20         <th>STATE</th>
21     </tr>
22     <tr>
23         <td>[lrn]</td>
24         <td>[NAME]</td>

```

(下页继续)

(续上页)

```

25    <td> [FENAME ] </td>
26  </tr>
27 </table>
```

不管模板的级别, MapServer 采用了相同的替换字符串方法来填充查询模版, 因为它用来填充主要的应用模板。在查询模板的情况下, 替换字符集串包含属性列名称括在方括号中, 字符串代表 MapServer 返回的信息。

有时一个查询可能不会返回任何结果 (即, 无要素或属性的匹配查询参数)。在这种情况下, MapServer 在浏览器中显示未格式化的消息类似于以下内容:

```
msQueryByPoint(): Search returned no results. No matching record(s) found.
```

由于这个信息不是很美观, MapServer 提供了一种机制来代替格式化网页, 通过 WEB 对象的 EMPTY 关键字的方式告知用户空值。这个页面并非是模板, 类似于 404 页面。

打开示例的 URL , 标中“地家与地区”图层, 然后在地图中用鼠标点选某个国家, 查看检索结果的页面 (图 4.8 )。

## 查询结果页面

查询范围:	-180.301003 -90.000000 180.301003 90.000000
查询坐标:	463, 97 (Image coordinates) 98.428094, 31.605351 (Map coordinates)
结果统计:	查询图层数目=1 查询结果数目=1

## 图层检索结果: 国家与地区

	NAME	STATE
1	CHINA	Peoples Republic of China

## 查询地图示例



图 4.8: MapServer 检索结果

#### 4.7.4 多图层检索模式

上面地图的定义在单图层模式下工作的很好，但是如果选择“多图层检索”，在地图上点击两个或更多国家交界处。当检索的结果不仅仅 1 个时，会发现检索结果的表格也出现多个，与结果的数目一样。这样的结果没有问题，但是看起来并不美观。究其原因，就是在于定义查询模板时没有分开模板。

这里给出针对河流的案例，在河流图层中，按如下定义：

```

130 LAYER
131   HEADER "r_header.html"
132   FOOTER "r_footer.html"
133   NAME "wriver"
134   DATA "wriver.shp"
135   TEMPLATE "r_query.html"
136   STATUS ON
137   TYPE LINE
138   CLASS
139     NAME "World River"
140     STYLE
141       COLOR 0 0 255
142     END
143   END
144 END

```

这里与模板相关的文件定义了 3 个。对应地将国家与区域图层中定义的模板分成了 3 部分。这样在组合检索结果时，河流表格的头与底部只出现一次，从而较好地合并为 1 个表格文件。其中 r\_query.html 文件内容如下：

```

2 <!-- MapServer Template -->
3 <tr><td>[lrn]</td><td>[NAME]</td><td>[FENAME]</td></tr>

```

#### 4.7.5 QUERYMAP 说明

在最后对结果页面底部的查询地图进行一下说明。

QUERYMAP 是一个能展示查询结果，突出要素，符合选择标准的地图。QUERYMAP 在默认情况下不会产生，它们必须在 Mapfile 中指定 QUERYMAP 对象。如果不指定 Mapfile 中的 QUERYMAP，查询仍然可以进行，但只有表格结果将是可用的：

在 Mapfile 中按如下定义：

```

74 QUERYMAP
75   SIZE 200 150

```

(下页继续)

(续上页)

```
76 STATUS ON  
77 STYLE HILITE  
78 COLOR 255 0 0  
79 END
```

在 HTML 结果中，由于 QUERYMAP 是作为普通地图上的图像以相同的方式在模板中所引用的，它们允许使用平移和缩放控制。QUERYMAP 和普通地图图像的唯一明显的区别是查询地图图像中存在的突出特征。然而，从开发者的角度来看，MapServer 应用程序允许用户浏览查询地图图像比不能浏览更复杂。但是现代的 WebGIS 检索方式要比这种方式方便地多，所以不再深入说明。



# 第5章 Web 地图服务（WMS）的概念与应用

本章前面对 MapServer 的功能介绍皆为较传统的方式。目前 WebGIS 的形式可能更多地通过地图服务的形式，MapServer 目前也支持大部分的地图服务标准。这其中最基本的就是 WMS，一种地图发布服务。

地理信息系统专业人员多年来一直在使用 WMS 绘制自由／开源软件网络地图。WMS 没有使用最新最先进的技术，但它是一个功能性和广泛的规范。

## 5.1 使用 MapServer 发布 WMS

MapServer 较早提供了地图发布功能，是传统的方式；后来才响应 OGC 标准实现了基于 OGC 的服务标准。单独使用 MapServer 哪种方式都可以，但是为了与其他软件工具进行交互，需要使用 WMS 等标准，因此，只要其他产品或应用程序使用通用的标准或协议，它们便可以一起工作。

MapServer 同样通过修改 Mapfile 来支持 WMS。

### 5.1.1 MapServer 对 WMS 的支持

WMS 提供了一种通过 Web 发送地图图像的方法。MapServer 应用程序可以从具有 WMS 可用的远程源中请求自定义地图。对于 MapServer 启用此服务很简单。

需要修改地图文件的两个部分以支持 WMS。需要在地图文件的 WEB 部分进行一些更改；例如，在地图文件的 WEB 部分中，需要进行一些更改。需要在要提供的图层中进行其他更改。

从 WMS 请求地图时，必须提供一些必需的参数，大多数 GIS 软件提供了与 WMS 交互的用户友好界面，这样就不必自己构建 URL。在许多情况下，都可以在 Mapfile 中仅指定 SRS、图层和图像类型。其余请求由 MapServer 在发出实际请求时处理。例如，MapServer 知道地图的大小，并请求相同的图像大小；它还知道地图的范围，因此它会自动传递带有正确值的边界框参数。如果要手动构建 URL，则必须指定所有这些参数。

参数名称不区分大小写，但传递给参数的值区分大小写。根据接收请求的服务器，则需要精确匹配它们的值。例如，某些服务器允许 `request=getMap`，而其他服务器则要求 `request=getmap`。由于图层名称使用错误的大小写也是一个常见的错误。

### 5.1.2 使用 MapServer 发布 WMS

接下来介绍如何使用 MapServer 发布 WMS，以及如何使用。

MapServer 发布 WMS，是通过定义 WMS 的 Mapfile；Mapfile 定义了数据源，包含数据库的和本地的文件；还包含一些 WMS 图层的样式，包含配色、线条粗细、是否用相应的属性标识地物等等，这些都是前面章节说明的。除了地图的样式定义，要发布 Web 服务需要在 Mapfile 中补充 WEB 对象的定义。

看一下 Mapfile，URL 为 <http://webgis.cn/mfb1.map>，与 WMS 定义的关键代码如下：

```

7 WEB
8     METADATA
9         "wms_title" "WMS Demo Server"
10        "wms_onlineresource" "//webgis.cn/cgi-bin/mapserv?map=/owg/mfb1.
11        ↳map&""
12        "ows_onlineresource" "//webgis.cn/cgi-bin/mapserv?map=/owg/mfb1.
13        ↳map&""
14         "wms_srs" "EPSG:4269 EPSG:4326"
15         "wms_enable_request" "* !GetFeatureInfo"
16     END
17 END

```

注意 wms\_enable\_request 或 ows\_enable\_request，配置完成信息之后中，还要配置可供使用的接口。

1. "wms\_enable\_request" 声明了 WMS 的功能如何启用，
2. "wms\_onlineresource" 定义了 MapServer CGI（地图渲染程序）程序的位置，
3. "ows\_onlineresource" 与 "wms\_onlineresource" 作用相同。
4. "wms\_srs" 规定了 WMS 服务的坐标参考系

说明：以空格分隔的启用请求列表。默认为无。可以启用以下请求：GetCapabilities，GetMap，GetFeatureInfo 和 GetLegendGraphic。请求前面的！将禁用该请求，使用 \* 表示启用所有请求。

下面的代码仅启用 GetMap 和 GetFeatureInfo：

```
"wms_enable_request" "GetMap GetFeatureInfo"
```

下面的代码会开启除了 GetFeatureInfo 以外的其他请求：

```
"wms_enable_request" "* !GetFeatureInfo"
```

### 5.1.3 访问 GetCapabilities 接口

要了解 WMS 提供的功能，就需要访问此功能元数据。这是对 WMS 文档（也称为功能元数据）的引用。对于刚开始使用 WMS 的人来说，这可能没有太大意义，但这是共享连接信息的标准方法。

一旦 WMS 服务器启动并以对地图文件的先前修改运行，就可以对其进行测试。GetCapabilities 请求返回有关服务的元数据，可以在发出其他类型的请求时将其用作指南。当调用 GetCapabilities 时，通常还不太了解该服务；因此，不需要提供一堆参数。实际上是在问，“告诉这个服务包含什么，它能做什么。”

配置好 Mapfile 之后，可以按照 WMS 协议来访问相关信息。GetCapabilities 是获取服务元数据的接口，其相关参数如下 表 5.1：

表 5.1: GetCapabilities 请求 URL 的参数

参数	是否必选	说明
VERSION=version	可选	请求版本，URL 中可选的请求参数，当没有使用时就使用当前服务提供商提供的最高的 wms 版本服务。可用的值为 1.1.1，或 1.3.0
SERVICE=WMS	必选	服务类型
REQUEST=GetCapabilities	必选	请求名，请求中必须设置当前请求的操作，可以设置 request 的值为 Getcapabilities、GetMap、GetFeatureInfo
FORMAT=MIME_type	可选	服务元数据的输出格式
UPDATESEQUENCE=string	可选	用于控制缓冲存储的数字序列或字符串

GetCapabilities 请求的样例 <http://webgis.cn/621> 参数分解如下：

```
>>> [print(i,p) for i,p in enumerate(url.split('&'))]
0 http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb1.map
1 SERVICE=WMS
2 VERSION=1.3.0
3 REQUEST=GetCapabilities
```

在浏览器中访问地址后，返回的是一个 XML 文件。如果浏览器没有自动显示文档的内容，则可能需要将其保存到系统上的文件中，然后使用其他程序将其打开。文件中的内容很多，包括 Service、Capability 等部分，详细信息可以查看返回 XML 文件。

### 5.1.4 功能元数据文档分析说明

功能元数据的 XML 格式是具有非常特定结构的文本。WMS 信息在成对的标记标签之间编码，标记的文本片段之间用小于号和大于号 <> 包围。当看到带有前面的反斜杠/的标记时，它表示一条信息的结尾。例如，<Layer ....> 是某些图层信息的开始，而</Layer> 是该部分的结尾。请注意，在其他部分中有一些缩进和嵌入的文本。例如，<NAME> 在 <LAYER> 中。这意味着名称信息与该特定层部分有关。

```
<?xml version='1.0' encoding="UTF-8" standalone="no" ?>
<WMS_Capabilities version="1.3.0" xmlns="" ...>
<Service>
  <Name>WMS</Name>
  <Title>WMS Demo Server</Title>
  <OnlineResource xmlns:xlink="..." />
  <ContactInformation>
    </ContactInformation>
  <MaxWidth>4096</MaxWidth>
  <MaxHeight>4096</MaxHeight>
</Service>
...
</WMS_Capabilities>
```

这个文档主要包含两个节点 Service 和 Capability。Service 节点包含了有关服务器的一般特性数据。Capability 对于应用更加重要，如果想正确的使用 WMS 服务器，这个节点里面的内容就必须搞清楚了。可以从这个节点获得如下信息：

1. 服务器支持哪些方法，WMS 服务器必须支持 GetCapabilities 和 GetMap，此外还有许多扩展的方法。
2. 服务器发布了哪些图层，这个是 WMS 的重点，不然连最简单的 GetMap 调用都构造不出来。
3. 服务器支持哪些返回格式，WMS 返回的地图都是渲染好的图片，因此这里的格式基本上都是图片格式，如 image/png。
4. 功能元数据通常包含丰富的信息，可能包括维护服务人员的姓名和使用限制。最吸引人的部分描述了可用的投影，图层和图像格式等技术细节。

功能元数据在有效的 Web 地图服务中起着不可或缺的作用。它们包含的元数据（有关数据的数据）通过查看数据本身并不明显。这些文档允许工具搜索信息或目录服务的可用性，甚至可以按地理位置搜索。功能元数据描述了可用的信息以及如何访问它们。没有这些文件，OGC Web 服务的实施会受到阻碍。

需要知道的是每个小部分都包含有关该服务的一些信息。下面对文档中的关键内容进行一下说明。

在 LAYER 对象中可以看到有关 WMS 中每个层的信息，这在发出 GetMap 请求时非常有用，并且希望知道要为 LAYERS 参数指定的适当名称。NAME 元素为图层提供了一个唯一名称，需要在应用程序中引用该名称。图层名称可能非常隐讳，TITLE 元素可以帮助了解名称的含义。使用 MapServer 地图文件中图层的名称而非标题。

```

91 <Layer queryable="0" opaque="0" cascaded="0">
92   <Name>topo</Name>
93   <!-- WARNING: Mandatory metadata '..._title' was missing in this_
94   -->
95   <Title>topo</Title>
96   <CRS>EPSG:4326</CRS>
97   <EX_GeographicBoundingBox>
98     <westBoundLongitude>-180.022</westBoundLongitude>
99     <eastBoundLongitude>179.978</eastBoundLongitude>
100    <southBoundLatitude>-89.978</southBoundLatitude>
101    <northBoundLatitude>90.022</northBoundLatitude>
102  </EX_GeographicBoundingBox>
103  <BoundingBox CRS="EPSG:4326"
104    minx="-89.978" miny="-180.022" maxx="90.022" maxy="179.978
  " />
</Layer>
```

如果不熟悉 WMS 覆盖的地理区域，EX\_GeographicBoundingBox 节点非常有用。坐标表示服务定义的覆盖地理范围。

```

85 <EX_GeographicBoundingBox>
86   <westBoundLongitude>-180</westBoundLongitude>
87   <eastBoundLongitude>180</eastBoundLongitude>
88   <southBoundLatitude>-90</southBoundLatitude>
89   <northBoundLatitude>90</northBoundLatitude>
90 </EX_GeographicBoundingBox>
```

在 MapServer 中使用 WMS 图层时，投影尤为重要。远程 WMS 源可以提供各种投影形式的地图。设置图层时，需要确定要请求的投影。这应该与应用程序中使用的投影相同。显示了服务的标题以及可以选择的两个空间参考系统代码。该示例的 EPSG 代码为 4326，这是一个通常称为 WGS 84 的全局坐标参考系统。它被列为一个选项：<SRS>EPSG:4326</SRS>。注意这里也有一个空间范围，这个空间范围是根据此图层计算出来的，而并非是定义的。

```

102 <BoundingBox CRS="EPSG:4326"
103   minx="-89.978" miny="-180.022" maxx="90.022" maxy="179.978" />
```

### 5.1.5 使用 OWSLib 模块访问 GetCapabilities 接口

前面介绍了 Mapfile 中配置的 WMS\_ONLINERESOURCE 与 OWS\_ONLINERESOURCE，但是似乎有些多余。需要通过这些参数的地址访问 URL，然后才会返回这个已经预先知道的信息。

实际上，首先 WMS\_ONLINERESOURCE 与 OWS\_ONLINERESOURCE 只是字符串，即使写成无效的地址，也不会有任何问题。其次，Web 服务是供计算机或者网络来使用的，这些进一步的信息，可以供后面的操作继续使用。

下面使用 OWSLib 模块访问 GetCapabilities 接口看一下：

```
>>> from owslib.wms import WebMapService
>>> wms = WebMapService(
...     'http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map',
...     version='1.3.0',
... )
```

建立了连接后，可以查看相关的属性：

```
>>> wms.identification.type
'WMS'
>>> wms.identification.title
'WMS Demo Server'
>>> list(wms.contents)
['EX2_', 'modis', 'states', 'states_line']
```

以及可用的操作：

```
>>> [op.name for op in wms.operations]
['GetCapabilities', 'GetMap', 'GetFeatureInfo',
'DescribeLayer', 'GetLegendGraphic', 'GetStyles']
```

针对 GetCapabilities 操作，可以具体查看：

```
>>> wms.getOperationByName('GetCapabilities').methods
[{'type': 'Get',
 'url': 'http://webgis.cn/cgi-bin/mapserv?map=/owg/example_ogc.map&'},
 {'type': 'Post',
 'url': 'http://webgis.cn/cgi-bin/mapserv?map=/owg/example_ogc.map&'}]
>>> wms.getOperationByName('GetCapabilities').formatOptions
['text/xml']
>>> wms.version
'1.3.0'
```

这个会返回前面介绍过的 XML 的信息，内容很多不再展示。

```
>>> wms.getServiceXML()
```

## 5.2 WMS GetMap 访问方法

GetMap 接口的目的在于请求服务器生成一幅具有确定地理位置坐标范围的地图图像，可以认为是 WMS 的核心操作。按照 WMS 规范，这个操作需要明确地指定出操作本身遵循的 WMS 规范的版本号，以及请求的图层（Layers）、各图层使用的样式（Styles）、空间参照系（SRS）、边界矩形（BBox）、图片的格式（Format）和大小（Width、Height）等参数。

GetMap 允许客户端请求多个服务器来制作叠置的地图图层，从而处理具有相同参照系、大小、比例的像素几何体。这些图层可以在客户端按一定的顺序排列显示，而且可以选用透明像素技术使不同来源的地图信息根据要求来显示。Web 地图服务支持图片或图形格式显示地图视图。图片格式包括公共的影像格式如 GIF、PNG、JPEG 等，这些格式被大多数 Web 浏览器支持，而其他图形格式的显示可能还需要一些助手程序支持。图形格式包括 SVG 和 WebCGM 格式，在 WMS 中不常用。

另外，可选的 Transparent 参数用来指定地图的背景是否透明，默认值是 False。允许返回结果以透明的方式绘制的功能，这样就可以叠置不同请求的地图。每个 WMS 最好提供一种支持透明显示的图像格式，以便与其他地图图片进行叠置，如 PNG 格式。WMS 返回的图像所支持的格式，是在 Mapfile 中由 OUTPUTFORMAT 参数配置的。

### 5.2.1 MapServer 中配置支持 GetMap

要使用 GetMap，需要在 WMS 中开启 GetMap 选项。使用的 Mapfile 为 <http://webgis.cn/mfb2.map>。

```

17 WEB
18   TEMPLATE "tmpl_ogc.html"
19   IMAGEPATH "/owg/ms_tmp/"
20   IMAGEURL "/ms_tmp/"
21 METADATA
22   "wms_title" "WMS Demo Server"
23   "wms_version" "1.3.0"
24   "wms_onlineresource" "//webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.
25   ↳map&" 
26   "wms_srs" "EPSG:4326"
27   "wms_enable_request" "*"
28   "ows_enable_request" "*"
29   "wms_feature_info_mime_type" "text/html"
   "wms_format" "image/png"
```

(下页继续)

(续上页)

```
30    END  
31 END
```

如果不开启 wms\_enable\_request 的话，访问 URL 会显示错误信息：

```
<?xml version='1.0' encoding="UTF-8" standalone="no" ?>  
<ServiceExceptionReport version="1.3.0" xmlns="" ... >  
<ServiceException>  
msWMSDispatch(): WMS server error. WMS request not enabled. ...  
</ServiceException>  
</ServiceExceptionReport>
```

未定义投影的 MapServer 并没有语法错误，也可以正常使用；但是要支持 WMS，则必须设置好投影。如果没设置好，访问会出现：

```
<?xml version='1.0' encoding="UTF-8" standalone="no" ?>  
<ServiceExceptionReport version="1.3.0" xmlns="" ...>  
<ServiceException code="InvalidCRS">  
msWMSLoadGetMapParams(): WMS server error. Cannot set new  
CRS on a map that doesn't have any projection set. ...  
</ServiceException>  
</ServiceExceptionReport>
```

## 5.2.2 GetMap 请求的参数

如表 5.2

表 5.2: WMS 请求参数说明

WMS 例子	通用名	描述
ser- vice=WMS	服务类型告诉 服务器要发出 WMS 请求。	这是由 MapServer 自动完成的。
re- quest=getmap	请求地图	这是由 MapServer 自动完成的。
version=1.3.0	指定要使用的 WMS 的版本。	有些 WMS 只支持某些版本。
CRS=CRS: 84	投影或空间参 考系统 (SRS)	使用 EPSG 投影代码请求的地图投影。SRS 在未来的 WMS 版本中将被称为 CRS。
for- mat=image/ jpeg	返回地图图像 的格式	地图将以何种图像格式发回, 可以为 image/gif, image/jpg,image/svg+xml 等等。
lay- ers=Countries,road,water,cities	请求的图层名 称	服务器用来描述要绘制的层或层组的名称。作为 URL, 该参数称为 layers。地图文件使用 wms_name 参数指定图层。若存在 SLD 参数则是可选的
width="800"& height="400"以像 素为单位。	图层绘制样式 名称	如果服务器支持, 则可以以不同的方式绘制图层, 例如, 使用不同的数据集。将其保留为空白将选择默认样式。若存在 SLD 参数则是可选的
'bbox=- 120,28,- 60,58	地理范围或边 界框, 以 SRS 单 位表示	图片的宽度和高度, 以像素为单位。许多服务器将图片的大小限制为 1024x1024 或 2000x2000。宽度和高度的不同设置可能会引起返回图像的变形
		定义地图西南角和东北角的两对坐标。

另外还有一些其他可选参数, 如 TRANSPARENT 控制地图背景的透明性, BGCOLOR 以十六进制 RGB 颜色值表示的背景颜色, EXCEPTIONS 表示 WMS 通告异常的格式、TIME 与 ELEVATION 表示请求层的时间值与高程, 以及其它维度与可选的实验性参数。

注: 在 LAYERS 参数值和 STYLES 参数值之间存在一一对应关系。

下面是具体的请求 <http://webgis.cn/632>, 对 URL 进行解析:

```
>>> [print(i,p) for i,p in enumerate(url.split('&'))]
0 http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map
1 SERVICE=WMS
2 VERSION=1.3.0
3 REQUEST=GetMAP
```

(下页继续)

(续上页)

```
4 LAYERS=states
5 BBOX=-180,-90,180,90
6 CRS=CRS:84
7 INFO_FORMAT=text/html
8 format=image/png
9 width=600
10 height=300
11 styles=
```

返回的图像如 图 5.1 所示：



图 5.1：使用 GetMap 获取的地图

注意此处同 节 2.2 介绍的获取地图方式截然不同。同样是获取地图，这里使用了 WMS 标准，所用的参数与 MapServer 本身支持方法差异很大。当然关键并非是参数的差异；采用 WMS 的方式，对于任何服务都是一样的。

这是一个非常简单的示例，尽管看起来可能很复杂。一些 WMS 提供程序可以使用其他参数和选项，但这只是最基本的。如果需要其他大小的地图或其他图层，可以通过更改各个参数来查看效果。

此 WMS 的局限性在于，如果过快地发出请求，可能会带来错误。如果出现错误，请等待几秒钟，然后重试。

### 5.2.3 对 HTTP 方式的 GetMap 请求响应

一个有效的 GetMap 请求将返回一张由具有地理参照信息的图层组成的地图，这些图层有指定的样式、空间参照系统、边界矩形、大小、格式和透明度。一个无效的 GetMap 请求返回一个格式化的错误信息。在 HTTP 环境下，返回值的内容类型是一个 MIME 类型。

对有效的 GetMap 请求的响应必须是一张地图，它包含了所请求的具有地理参考坐标的信息层，使用的是期望的样式，并且采用了指定的空间参照系、范围框、大小和透明性。一个无效的 GetMap 请求必须产生一个按请求的 Exceptions 格式输出的错误信息（或在极端情况下回答一个网络协议错误）。

如果参数设置错误，将返回一个 XML 文件来描述错误信息。例如：

```
<ServiceExceptionReport version="1.1.1">
  <ServiceException code="">
    WIDTH and HEIGHT incorrectly specified
  </ServiceException>
</ServiceExceptionReport>
```

---

注解：WMS 1.1.1 与 WMS 1.3.0 对于坐标系统的请求参数是不一样的：

- WMS 1.1.1，使用 SRS=EPSG:4326
  - WMS 1.3.0，使用 CRS=CRS:84
- 

上面获取的是整个世界地图，通过修改参数，要获取中国部分，则可以使用下面参数：

URL 为 <http://webgis.cn/671>。

```
>>> [print(i,p) for i,p in enumerate(url.split('&'))]
0 http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map
1 SERVICE=WMS
2 VERSION=1.3.0
3 REQUEST=GetMAP
4 LAYERS=states
5 BBOX=73,3,136,54
6 CRS=CRS:84
7 INFO_FORMAT=text/html
8 format=image/png
9 width=200
10 height=150
11 styles=
```

效果如下 图 5.2:

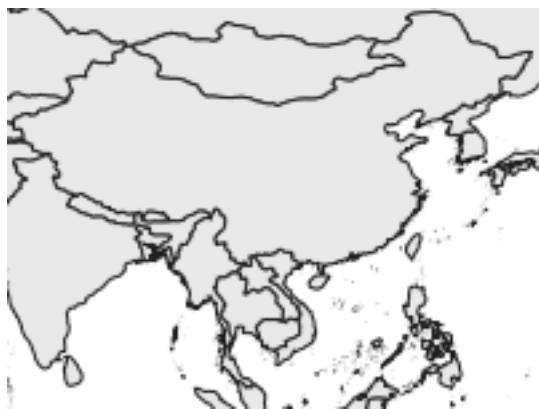


图 5.2：修改参数后的 GetMap

#### 5.2.4 使用 OWSLib 访问 GetMap

然后在 OWSLib 中继续看一下访问的方式与返回的结果。

```
>>> wms.getOperationByName('GetMap').methods
[{'type': 'Get',
 'url': 'http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map&'},
 {'type': 'Post',
 'url': 'http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map&'}]
```

此处获得的方法，在稍后的 getmap 函数中，会使用这些 URL 进行请求。

```
>>> wms.getOperationByName('GetMap').formatOptions
['image/png', 'image/jpeg', 'image/png; mode=8bit',
 'application/x-pdf', 'image/svg+xml', 'image/tiff',
 'application/vnd.google-earth.kml+xml',
 'application/vnd.google-earth.kmz']
```

```
>>> img = wms.getmap( layers=['states'],
 ...      crs = 'CRS:84',
 ...      bbox=(73,3,136,54),
 ...      size=(300, 250),
 ...      format='image/png',
 ...      transparent=True )
```

img 是图片对象，进一步将其保存成文件。

```
>>> out = open('xx_getmap.png', 'wb')
>>> out.write(img.read())
26553
>>> out.close()
```

## 5.3 WMS 的 GetFeatureInfo 请求

一个 WMS 可以提供可选的 GetFeatureInfo 操作，对提供这种操作的 WMS，称其地图为“可查询的”。GetFeatureInfo 操作返回地图上指定点附近要素的信息，操作要求客户端指定查询的图层（或图层列表）以及要素信息的返回格式。客户端可以通过增加一个位置参数（距图片左上角的偏移量）或该点附近需要查询信息的要素的数量等参数，到所请求地图的 URL 中，就可以请求地图上相应要素的信息了。

要实现前端的功能，实现用户点击像素查询要素的模式和元数据的交互操作，需要使用前端的类库与工具，在后面章节会说明一点。

### 5.3.1 GetFeatureInfo 接口说明

该操作向 WMS 的客户端程序提供了进一步查询特定空间实体信息的能力。这种操作往往是由客户程序在 WMS 服务器先前返回的地图上指定了一个空间实体，进而提交查询而形成。对于其它层，客户端不能发送 GetFeatureInfo 请求。

GetFeatureInfo 是服务发布者可以启用的可选方法，允许用户在特定位置查询 WMS 层的属性数据。这使得添加交互元素（如信息弹出窗口）成为可能，而无需寻求第二个 web 服务的帮助。

当发出 GetFeatureInfo 请求时，需要提供与 GetMap 请求相同的许多参数，并添加要查询的像素的屏幕坐标。

GetFeatureInfo 只支持对于属性 queryable="1"（真）的图层。如果一个 WMS 客户端向不支持该操作的图层发出请求，将返回一个格式化的服务异常信息。

由于 WMS 是无状态的，所以在 GetFeatureInfo 请求中还应包括原始的 GetMap 请求中所有的参数（除 Version 和 Request 外）。根据 GetMap 请求中的空间上下文信息（BBOX、SRS、Width 和 Height）和用户选择的位置，WMS 返回该位置更多的信息。

在 GetFeatureInfo 请求参数中还包括 QUERY\_LAYERS 参数，该参数列表指明获取要素信息的图层。该参数列表中的图层包含在 GetCapabilities 返回的 XML 文档中。

GetFeatureInfo 请求中可选的 Info\_Format 参数指明用什么格式返回要素的信息。该格式在 WMS 的 Capabilities XML 文档中定义为 MIME 类型。如 Info\_Format=application/vnd.ogc.gml 表示返回结果用 GML 描述。

### 5.3.2 配置 MapServer

Mapfile 中有模板 (TEMPLATE) 这个参数，而这个参数就负责配置 WMS GetFeatureInfo 操作的返回结果内容和格式。MapServer 的模板在前面介绍过了，使用此项功能来返回查询信息。

这里使用的 Mapfile 的 URL 为 <http://webgis.cn/mfb2.map>。

其中 WEB 部分为：

```

17 WEB
18   TEMPLATE "tmpl_ogc.html"
19   IMAGEPATH "/owg/ms_tmp/"
20   IMAGEURL "/ms_tmp/"
21 METADATA
22   "wms_title" "WMS Demo Server"
23   "wms_version" "1.3.0"
24   "wms_onlineresource" "//webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.
25   ↪map&" 
26   "wms_srs" "EPSG:4326"
27   "wms_enable_request" "*"
28   "ows_enable_request" "*"
29   "wms_feature_info_mime_type" "text/html"
30   "wms_format" "image/png"
31 END
END

```

图层部分的定义如下：

```

45 LAYER
46   NAME "states"
47   DATA "wcountry.shp"
48   STATUS OFF
49   TYPE POLYGON
50   TEMPLATE "tmpl_ogc.html"

```

这其中 WEB 对象和 LAYER 对象都发生了变化。WEB 对象增加了 "wms\_feature\_info\_mime\_type" "text/html"，从字面意思可以看出该参数主要配置 GetFeatureInfo 操作返回的结果格式类型，这里返回 HTML 格式；另外 LAYER 对象多出了 TEMPLATE 参数，这个参数配置返回的 HTML 内容，参数值是一个路径，指定模板文件所在位置。模板文件的内容如下：

```

1 <!-- MapServer Template -->
2 <!DOCTYPE html>

```

(下页继续)

(续上页)

```

3 <html lang="zh-cmn-Hans">
4 <head>
5   <meta charset="UTF-8">
6   <title>Mapserver template test</title>
7   <link rel="stylesheet" href="/tass/css/style.css"/>
8 </head>
9
10 <body>
11
12 <div class="container">
13   <table id="feature_info" class="table table-bordered">
14     <tr>
15       <td class="text-center"><label>gid :</label></td>
16       <td id="gid" class="text-center">
17         <div style="padding-left:13px;">[NAME]</div>
18       </td>
19     </tr>
20     <tr>
21       <td class="text-center"><label>name :</label></td>
22       <td class="text-center">
23         <input type="text"
24           readonly="readonly"
25           class="form-control min-input custom-form-control"
26           value="[POP]"/>
27       </td>
28     </tr>
29   </table>
30 </div>

```

模板文件在首行同样必须包含 `<!-- MapServer Template -->`，这会告诉 MapServer 填充需要填充的值（这是与普通 html 唯一的不同），需要填充的值是以中括号 [ ] 括起来的，内容是字段名，如数据库中的字段名有个 `id` 字段，要输出该字段就直接书写 `[id]` 即可，MapServer 会输出实际该字段的值取代该内容。

### 5.3.3 请求 GetFeatureInfo

GetFeatureInfo 进行请求时需要有不同的参数，在不同的 WMS 版本是不一样的。本书中以最新的 WMS 1.3.0 版本进行说明，在这个版本中 GetFeatureInfo 的位置参数使用使用 `I, J` 传递。

WMS 1.3.0 的请求参数如 表 5.3：

表 5.3: GetFeatureInfo 请求参数 (WMS 1.3.0)

请求参数	必选 (M) / 可选 (O)	描述
VERSION=version	M	请求版本。
REQUEST=GetFeatureInfo	M	请求名称。
map request part	M	地图请求参数的部分拷贝，这些参数产生了需要查询其信息的地图。
QUERY_LAYERS=layer	M	用逗号分隔的需要查询的一个或多个层的列表。
INFO_FORMAT=output	format	要素信息的返回格式 (MIME 类型)。
WIDTH	M	返回图片的宽度
HEIGHT	M	返回图片的高度
I=pixel_column	M	用像素表达的要素的 X 坐标 (以左上角坐标为 0)。
J=pixel_row	M	用像素表达的要素的 Y 坐标 (以左上角坐标为 0)。
FEATURE_COUNT=number	O	需要返回其信息的要素个数 (default=1)。
EXCEPTIONS=exception_format	O	WMS 报告异常信息采用的格式 (default=application/vnd.ogc.se_xml)。

复制下面的 URL 在浏览器中打开可以看到内容。

<http://webgis.cn/641>

对 URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(url.split('&'))]
0 http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map
1 SERVICE=WMS
2 VERSION=1.3.0
3 REQUEST=getfeatureinfo
4 QUERY_LAYERS=states
5 LAYERS=states
6 BBOX=-180,-90,180,90
7 CRS=CRS:84
8 INFO_FORMAT=text/html
9 WIDTH=600
10 HEIGHT=300
11 I=479
12 J=95
```

查询宽度为 600、高度为 300 的地图图片中坐标  $i=479$  和  $j=95$  处对应要素的属性：在这种情况下，返回的纯文本将响应坐标处的信息。根据结果，返回的是“中国”的信息，以 HTML

文档来描述对象，如下所示：

```
gid : RUSSIAN FEDERATION
name : 14553
```

从结果可以看出，MapServer 是将 TEMPLATE 配置参数的 html 文件替换相应的值后，然后返回。

将上面的参数 j 修改得小点 j=43 位置是位于中国的北边，打开 <http://webgis.cn/486>，可以看到返回的是俄罗斯的信息。

### 5.3.4 OWSLib 访问

同样使用 OWSLib 来看一下请求的方式。首先还是建立连接：

```
>>> from owslib.wms import WebMapService
>>> wms = WebMapService(
...     'http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map',
...     version='1.3.0',
... )
>>> wms.getOperationByName('GetFeatureInfo').methods
[
    {'type': 'Get',
     'url': 'http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map&'},
    {'type': 'Post',
     'url': 'http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map&'}]
```

发起请求，所需要的参数与前面类似：

```
>>> feainfo = wms.getfeatureinfo(request='GetFeatureInfo',
...     service = 'WMS',
...     version = '1.3.0',
...     query_layers=['states'],
...     layers = ['states'],
...     bbox=[-180,-90,180,90],
...     srs='CRS:84',
...     infor_format='text/html',
...     styles = '',
...     info_format = 'text/plain',
...     size=[600, 300],
...     xy = [480, 90],
...     format='image/png',
... )
```

返回的结果以纯文本返回，也可以以 HTML 方式返回。查看返回的结果：

```
>>> print( feainfo.read().decode())
GetFeatureInfo results:
Layer 'states'
  Feature 246:
```

## 5.4 分布式地图服务：使用 WMS 作为数据源

到目前为止，使用 WMS 获取地图看起来与原来类似，只不过通过 Web 浏览器中的 URL 发送 WMS 请求时调用的参数与原来不同。实际上，除了更强调标准规范，WMS 更重要的是提供了机器可解析的机制。

MapServer 可以分享（或服务）MapFile 中的图层到其他地图服务器，这种情况可以认为是 WMS（WFS）服务器。MapServer 应用程序也可以成为 WMS（WFS）客户端的作用。当 WMS 与自己的本地存储数据结合使用时，WMS 的功能将大大增强。远程源对地图上的其他图层没有任何关系，但可以添加到其中。MapServer 提取由请求产生的图像，并将其视为要覆盖的另一层。甚至可以有一个仅使用外部 WMS 层的应用程序。WMS 真正的威力在于将自己的信息与其他来源的信息相结合，可以产生比自己能够做的更加丰富的地图。

这样的应用体现了应用程序之间的互操作性；可以使用包括 MapServer 和任何其他 WMS 兼容的制图程序。这就是 WMS 的强大功能：应用程序独立性不与一个特定的软件包捆绑在一起。对于 MapServer 发布的 WMS 服务，任何其他支持 WMS 的程序都可以连接到它，如 GeoServer，ArcGIS Server。

### 5.4.1 将 MapServer 用作 WMS 服务器端

WMS 本质上担当着两个角色：作为使用者访问远程数据源，以及为其他人提供或共享数据。WMS 与共享信息有关。

服务器端的功能与使用在前面已经说明。WEB 部分为整个 MapServer 应用程序定义了一些设置。在这种情况下，需要向外部 WMS 客户端添加必要的定义 WMS 的关键词。

作为服务器端提供服务的 Mapfile 为 <http://webgis.cn/mfba.map>，对前面使用的 Mapfile 增加了道路与河流图层。

### 5.4.2 MapServer 作为 WMS 客户端的配置

访问远程数据源需要一些有关数据源的特定信息。细节可能包括要连接的服务器地址，可用的数据层或可以将数据发送的格式。一旦了解详细信息，就可以设置应用程序以确切地请求所想要的内容。MapServer 会将自己的信息添加到每个请求中，这样就可以在地图上得到一些符合逻辑的信息。

另外，作为客户端时，Mapfile 中不需要配置 WEB 对象；而在服务端则是必需的。作为客户端不配置 WEB 对象时，这样对外不能提供 WMS 服务，只表现为普通的 MapServer 程序；如果同时进行配置，则可作为服务器端继续对外提供服务。

在示例中，使用了同一服务器提供的 WMS，但是结构上是绝对不一样的。原来的数据资源都是由操作系统的文件管理系统提供了，示例中则是由 Web 服务提供的。这个 Web 资源可以换成其他服务，而更换本地的数据资源则要麻烦许多。

下面是本例中使用的 Mapfile (<http://webgis.cn/mfb4.map>)，调用了刚才 Mapfile 中的图层。

```

10 LAYER
11     NAME "basemap"
12     TYPE RASTER
13     OFFSITE 0 0 0
14     STATUS OFF
15     CONNECTIONTYPE WMS
16     CONNECTION "http://webgis.cn/cgi-bin/mapserv?map=/owg/mfba.map"
17     METADATA
18         "wms_srs" "EPSG:4326"
19         "wms_name" "wcountry,wriver,wroads"
20         "wms_server_version" "1.1.1"
21         "wms_format" "image/png"
22     END
23     PROJECTION
24         "init=epsg:4326"
25     END
26 END

```

连接设置必须设置为 CONNECTIONTYPE WMS，以告诉 MapServer 发出 WMS 请求。然后，将基本 URL 作为连接字符串，使之能够连接从另一台服务器上的数据。在 WMS 连接的情况下，这是一个 URL，格式为一个单一的字符串。如果服务器基于 MapServer，则可能还需要在 CONNECTION 字符串中指定 Mapfile 的路径。在这种情况下，基本网址将在此设置后以 & 号结尾；例如第 16 行所示。

在上面的资源中，WMS 需要使用 `http://` 开头，而不能用 `//`。

所有其余参数都放入该层的 METADATA 对象中。这些设置/值对由 MapServer 获取，并转换为适当的格式，以在后台请求地图图像。METADATA 标志着 WMS 层的元数据对象。MapServer

的使用此对象定义的参数，以及与上述连接参数，以形成有效的 WMS 请求。这个对象中包含下面的一些定义：

- “wms\_srs” “epsg:4326”，WMS 投影。有时 WMS 服务器支持多个投影。如果是这样的话，可能会想要求输出投影（EPSG: 2163）地图；
- “wms\_name” “wcountry,wriver,wroads”，要添加的 WMS 图层名称或列表，不局限于 1 个图层。所需要的所有图层都需要在 wms\_name 设置中进行添加，名称之间使用逗号分隔，这样实现了对多个图层的组合应用。
- “wms\_server\_version” “1.1.1”，WMS 版本的服务器的版本。可以查看更多支持的版本；
- “wms\_format” “image/png”，图像格式，预计接收来自 WMS 服务器。可以试试 image/jpg 或其他的允许取值。

Mapfile 中的设置与 URL 请求中使用的参数非常相似，如果将它们与前面的示例进行比较，则大多数设置都是不言自明的。最大的区别是需要设置的参数没有那么多，因为 MapServer 是会内部生成它们的（例如，宽度，高度，bbox）。MapServer 可以根据正在查看的当前地图，其范围，大小等来推断这些设置。

注意，在 WMS 层中不再使用任何 CLASS 对象。除非使用另一个称为样式层描述符（SLD）的 OGC 规范，否则不能更改 WMS 制图的绘制方式。如果 WMS 支持，则 SLD 允许请求 WMS 图层的特定样式。

### 5.4.3 效果与说明

发送请求 <http://webgis.cn/798> 以查看效果。URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(url.split('&'))]  
0 http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb4.map  
1 layers=basemap  
2 mode=map
```

效果如图 5.3：

在这里可以看到这个结果与前面的类似。这个地图是由三个图层构成的，但是通过 WMS，在 Mapfile 中组合成了一层，所以在参数调用中只使用了 layers=basemap。

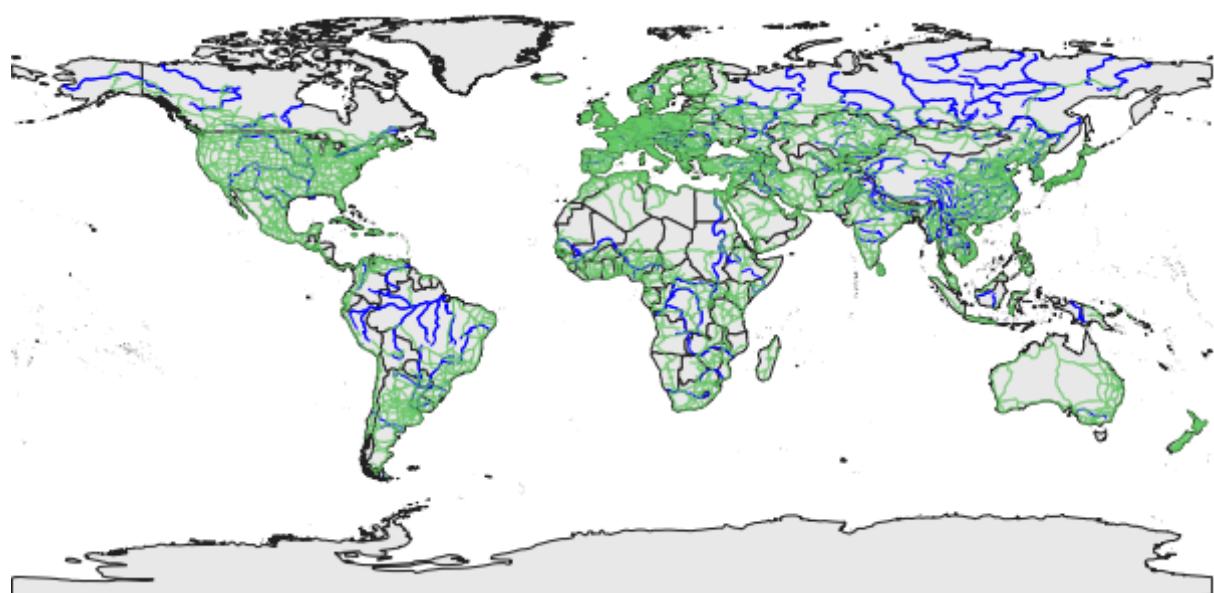


图 5.3：添加 WMS



# 第6章 MapServer 中的样式定制与实现

地图制图除了根据既有的样式进行制图，也发展出了进一步的需求。用户希望数据与样式进行分离，从而可以在客户端实现不同样式的自由定义，从而得到更加丰富的地图绘图成果。

在这方面，MapServer 开始提供了命名样式的方式，不同的样式有不同的名称，可以让用户进行选择，但这种方式依然有很大的限制。后来随着样式层描述器（Styled-Layer Descriptor, SLD）概念与标准的出现，也提供了更加灵活的方案，MapServer 实现了对 SLD 的支持。

本章主要对 SLD 的应用进行介绍，这是 OGC 标准；在最后一节对 MapServer 支持的命名样式进行一下说明。

## 6.1 SLD 标准介绍

WMS 的请求包括一些参数，但不能通过这些参数修改样式。样式的定义涉及很多方面，如果全部通过参数来传递会很复杂。在没有 SLD 之前，MapServer 使用一些已经在服务器上规定好的样式对地图进行可视化。而当使用了实现了 SLD 标准之后，它允许从客户端来对地图进行样式自定义、分级显示等操作，极大的扩展了地图可视化的灵活性。

SLD 是样式层描述器（Styled Layer Descriptor）的英文缩写，是 2005 年 OGC 提出的一个标准，这个标准在一定条件下允许 WMS 服务器对地图可视化的表现形式进行扩展。目前最新版本为 SLD 1.1.0，常用的开源软件如 QGIS 可以输出 SLD 的定义文件。

设计样式层描述器的目的是用来扩展 Web 地图服务器，以允许用户自己定义要素数据的符号表达。通常，在客户请求之前需要了解存储在远程主机上的数据信息（元数据）。这就要求在定义样式语言之外还要定义一些新的服务操作。

### 6.1.1 样式层描述器介绍

OGC 允许通过符合 SLD 标准的 XML 块来实现的来调整 WMS 层中使用的地图符号。SLD 描述了应该在 WMS 中应用的所有符号大小、颜色和标记。SLD 非常复杂，使用一系列的标准规范文档来定义它们应该如何操作。

SLD 是对 WMS 的扩展，使用由用户提供的样式信息绘制地图，增加了基本 WMS 不支持的操作：DescribeLayer、GetLegendGraphic、GetStyles。

### 6.1.2 SLD 技术细节

SLD 规范采用 XML 定义地图显示样式，通过自定义 SLD 来配置地图图层渲染的可视化风格，可以设置过滤器，自定义图例等。`rule` 是 SLD 最重要的一个元素，因为它允许根据给定的某个参数（使用过滤器）对数据集进行分类，所有与分类有关的重要参数都必须在 `rule` 元素中设置。

关于 SLD Schema，和 GML 类似，SLD 也通过一些 XML Schema 来定义 SLD 文档的结构，这些定义可以通过访问 <http://schemas.opengis.net/sld/> 得到。SLD 的 Schema 还依赖其它的 OGC 标准，其中包括：GML、Filter Encoding、Symbology Encoding。

使用 SLD 来进行地图样式自定义必须结合使用 Symbology Encoding 标准。Symbology Encoding 是标准也是基于 XML 模式定义的，标准允许自定义不同的符号样式来表达地图上不同的要素。SLD 文件使用这种语言，这样在地图渲染时地图服务可以解释由用户定义的样式。

一个 SLD 样式文档一般包含多个部分，而最重要的如下：

- `FeatureTypeStyle`：这一部分是整个样式文档的根节点，说明什么是它的样式，以及应用的特征类型。`FeatureTypeStyle` 包含一个或者多个 `Rule` 元素，`Rule` 元素允许有条件的映射。
- `RULE`（规则）：规则是根据属性条件和地图比例尺来对要素进行分组渲染，一般 `RULE` 中只允许渲染一种几何类型的要素，即点，线，面等其中的一种，但是可以和注记同时使用。
- `Symbolizer`（符号）：`Symbolizer` 指定数据应该如何符号可视化，在 1.0 的标准中包含五种类型的符号，分别是 `PointSymbolizer`（点符号）、`LineSymbolizer`（线符号）、`PolygonSymbolizer`（面符号）、`TextSymbolizer`（注记）、`RasterSymbolizer`（栅格）。

通过 SLD 可以设置的不仅仅是简单的颜色和宽度。点可以设置成常用的形状，如圆形，方形，星形，甚至可以是自定义的图片图形或文字来指定；线可以渲染成虚线或者点虚线等，多边形可以填充自定义的平铺图像，样式可以基于数据的属性给定，以至于要素能够被渲染成不同的风格。

### 6.1.3 设计与使用 SLD 的方法

使用 SLD 设计数据集样式有两种基本的方法，与 MapServer 制图的概念相同，只是方式不一样。最简单的方式是对某一数据集所有的要素用同一方式绘制。这种方式不需要了解基本数据要素的属性或要素类型，只需要知道描述这些样式的语言。这类需要在 SLD 文档中用 `FeatureTypeStyle` 描述。

更加复杂的需求是根据要素的属性，用不同的样式描述要素。例如，在一个道路数据集中，对高速公路用 3 像素宽的红线表示，对四车道的道路用 2 像素宽的黑线表示，对两车道的道路用 1 像素宽的黑线表示。

SLD 可以由服务发布者或客户端设计。设计 SLD 目的是用来扩展 Web 地图服务器，对人

或机器用户提供自定义样式规则的功能需要一个客户端和服务器都能理解的样式语言，以允许用户自己定义要素数据的符号表达。要制定一个真正有用的数据语言，在客户请求之前需要了解存储在远程主机上的数据信息（如 WMS 图层），需要知道描述要素类型的数据集的属性。可以通过调用 WMS 上的可选接口（如 `DescribeLayer`）操作来实现这一点。

一旦为 SLD 创建了 XML，就可以通过以下几种方法之一应用它。最常见的方法是将 SLD 放在 WEB 服务器上的某个地方，并在发出 GetMap 请求时引用该文件的 URL。GetMap 请求中有一个可选的 SLD 参数，可以将其用于此目的。另一种方法是使用 GetMap 请求的可选 `SLD_BODY` 参数，只需在请求的 URL 中直接提供相关的 XML 块。这显然会创建长而笨拙的 URL，并且需要很多特殊的字符编码或转义。

与 SLD 一起使用的 XML 可能会变得冗长，并且往往包含许多嵌套级别。因此，不太可能坐下来从头开始创作 SLD。相反，可能会从一个样本开始，并调整它以满足需要。或者，可以使用 QGIS 等软件作为辅助设置图层样式，然后将其保存为 SLD。

## 6.2 SLD 访问接口

通常，自定义 SLD 的方法只适用于矢量图层。有多种方法可以发送带有 WMS 请求的 SLD 文件，要求服务器以适合自己的方式设置地图样式。但是并不是所有的服务器（或客户机）都支持这一点，因为它的效率要低得多（例如，它防止服务器为其他用户缓存结果）。有一些接口可以来了解一些与样式相关的信息，这一节先对这些接口进行学习。

先定义一个 Mapfile 用做数据服务，具有缺省的简单样式。这个例子同样使用前面的 Mapfile <http://webgis.cn/mfb2.map>。具体不再说明。

### 6.2.1 `DescribeLayer` 请求

`DescribeLayer` 操作用来了解一个关于要素或要素类型的用户定义样式的需要信息。因为用户定义的样式要被应用到一个命名的层，所以需要一个机制来获取一个命名图层或覆盖层要素的信息。

一个 WMS 有可选的 `DescribeLayer` 请求。`DescribeLayer` 请求可以应用于多个图层：

<http://webgis.cn/681>

解析这个 URL 查看参数：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb2.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=DescribeLayer
```

(下页继续)

(续上页)

```

5 LAYERS=states
6 SLD_VERSION=1.1.0

```

其中：DescribeLayer 是 REQUEST 参数的一个新选项，LAYERS 参数用名称来指定一组命名的图层。对于每一个命名图层，应当指明是否基于要素数据，如果是，应当指明相应的 WFS/WCS 和要素类型。所以支持 DescribeLayer 是客户端使用自定义符号类型与服务器交互的唯一方式。

返回的结果为 XML 文件，内容如下：

```

1 <?xml version='1.0' encoding="UTF-8"?>
2 <DescribeLayerResponse xmlns="http://www.opengis.net/sld" ...
3   <Version>1.1.0</Version>
4   <LayerDescription>
5     <owsType>wfs</owsType>
6     <se:OnlineResource xlink:type="simple"/>
7     <TypeName>
8       <se:FeatureTypeName>states</se:FeatureTypeName>
9     </TypeName>
10    </LayerDescription>
11 </DescribeLayerResponse>

```

注意第 5 行，虽然 Mapfile 中定义的是 WMS，这里却返回 WFS。

## 6.2.2 GetStyles 请求

GetStyles 操作用来从一个支持 SLD 的 WMS 获取用户定义的样式。若访问成功，将返回一个 MIME 类型的 SLD 文档。在 SLD 文档中每个样式都用一个 UserStyle 元素表示，包括所有不能用命名样式（NameStyle）元素返回的样式。一个 UserStyle 元素提供关于一个图层内部组织更多的信息。

使用下面的请求查看是否生效。

<http://webgis.cn/231>

参数解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[\?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb2.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetStyles

```

(下页继续)

(续上页)

```
5 LAYERS=states
6 SLD_VERSION=1.1.0
```

结果如下以 XML 文件格式返回，其符合 SLD 1.10 版本的 XSD，这个文件的第 4 行开始定义 UserStyle，第 8 行至 16 行定义了样式的具体信息，内容如下：

```
8 <se:PolygonSymbolizer>
9   <se:Fill>
10    <se:SvgParameter name="fill">#e8e8e8</se:SvgParameter>
11   </se:Fill>
12   <se:Stroke>
13    <se:SvgParameter name="stroke">#202020</se:SvgParameter>
14    <se:SvgParameter name="stroke-width">1.00</se:SvgParameter>
15   </se:Stroke>
16 </se:PolygonSymbolizer>
```

### 6.2.3 GetLegendGraphic 请求

图例通常用来向用户说明地图中各种要素是如何表达的。因此，在地图显示的客户端使用 SLD 格式描述的样式生成图例非常重要。将 UserStyles 应用到 FeatureTypeStyles 和 Rules 结构中就可以方便地实现图例生成。因为 Rules 节点标示了每一个用来绘制地图不同类型的图形符号。根据 SLD 中的 UserStyle 定义，地图客户端可以为每一个图层生成一个图例。支持 SLD 的 WMS 的 GetLegendGraphic 操作用来引用图例图像，每一个图例图像使用一个独立的引用。

在 MapServer 6.2 之前，没有级联 GetLegendGraphic 请求。对于启用了 SLD 的 WMS，GetLegendGraphic 操作本身是可选的。除了 WMS 功能的 LegendURL 参考之外，它还提供了一种获取图例符号的通用机制。支持 GetLegendGraphic 调用的服务器可能会将 LegendURL 引用编码为 GetLegendGraphic，以实现接口一致性。可以将特定于供应商的参数添加到 GetLegendGraphic 请求中，并应用所有通常的 OGC 接口选项和规则。

在 GetLegendGraphic 请求中，参数 REQUEST、LAYER、FORMAT 是必须要有的。相关参数如 表 6.1：

表 6.1: 参数列表

参数类型	是否需要 (R 必须的, O 可选的)	功能
REQUEST	R	值必须为 GetLegendGraphic
LAYER	R	为其生成图例图形的层
FORMAT	R	这给出了用于返回图例图形的文件格式的 MIME 类型
STYLE	O	为其生成图例图形的图层样式如果不存在，则选择默认样式
FEATURE-TYPE	O	为其生成图例图形的特征、类型。如果图层只有一个要素类型，则不需要
RULE	O	样式规则，以生成图例图形
SCALE	O	在未为样式指定 RULE 的情况下，选择更合适的代表性图形
SLD	O	此参数指定对外部 SLD 文档的引用
SLD_BODY	O	此参数允许 SLD 文档直接包含在 HTTP-GET 请求中
WIDTH	O	以像素为单位的返回图形的宽度
HEIGHT	O	高度
EXCEPTIONS	O	返回异常的格式的 MIME 类型
LANGUAGE	O	允许设置样式标题和规则标题的标签语言

使用 <http://webgis.cn/914> 发起请求，URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[\?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb2.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetLegendGraphic
5 LAYER=states
6 SLD_VERSION=1.1.0
7 format=png
```

请求会返回图例图片，这个图片的大小会根据图例数量的多少而不同。

这个示例中比较简单（图 6.1）：

## □ States

图 6.1: GetLegendGraphic 请求返回的图例文件

### 6.2.4 客户端的访问方式

最后了解一下利用 SLD 请求 WMS 的三种方式：

1. 用 HTTP/GET 请求，参数中用 **SLD\_BODY** 参数包含一个 **SLDXML** 文档。第 1 种方法使用方便，但是可能存在由于过长的 URLs 而出现问题。
2. 用 HTTP/GET 请求，参数中引用一个远程的 SLD。这种方法的不足之处是客户端必须将 SLD 文档上传到 WMS 可以访问的一个 Web 地点。
3. 用 HTTP/POST 交互，用 XML 编码的 GetMap 请求，其中包含了一个 SLD。第 3 种方式比其他两种方式优越，但是 XML-POST 形式的 GetMap 请求方法目前缺乏支持。

在实际应用中，可以根据 **LAYERS** 参数是否出现，SLD 采用两种不同的方式。如果 **LAYERS** 没有出现，SLD 中的所有图层用所有定义的样式绘制，这与 XML-POST 请求相同。如果提供了 **LAYERS** 参数，只有指定的图层被绘制。这时 SLD 作为样式库，SLD 中定义的样式比地图服务器中定义的样式优先使用。

## 6.3 在 WMS 中使用内联的样式层描述器：使用 **SLD\_BODY** 参数

MapServer 在 4.2 版本中添加了 SLD 支持。本节描述了使用 MapServer 的 WMS GetMap 请求中利用样式化层描述符（SLD）支持的过程。SLD 支持服务器端的应用（能够读取 SLD 并将其与 GetMap 请求一起应用）和客户端的应用（包括将 SLD 请求发送到服务器并从 Mapfile 动态生成 SLD 文件）。

这一节首先看一下直接从客户端发送请求的方法。

### 6.3.1 请求示例

SLD 使用 XML 格式进行定义，先看一下 SLD 的实例。SLD 的格式要满足 XML 的语法要求，并且要满足 SLD 标准的约束：

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <StyledLayerDescriptor version="1.1.0"
3      xmlns="http://www.opengis.net/sld"
4      xmlns:se="http://www.opengis.net/se"
5      xmlns:ogc="http://www.opengis.net/ogc"
6      xmlns:xlink="http://www.w3.org/1999/xlink"
7      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

(下页继续)

(续上页)

```
8 xsi:schemaLocation="http://www.opengis.net/sld
9 http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd">
```

这个 SLD 文件的完整内容可通过 [http://webgis.cn/sld\\_world\\_lyr110.xml](http://webgis.cn/sld_world_lyr110.xml) 访问。在 SLD 中定义了面状的样式。MapServer 会根据 Mapfile 中定义，选择合适的定义进行渲染。其中面状样式定义如下：

```
17 <se:PolygonSymbolizer>
18   <se:Geometry>
19     <ogc:PropertyName>center-line2</ogc:PropertyName>
20   </se:Geometry>
21   <se:Fill>
22     <se:SvgParameter name="fill">#e8e8e8</se:SvgParameter>
23   </se:Fill>
24   <se:Stroke>
25     <se:SvgParameter name="stroke">#0000ff</se:SvgParameter>
26     <se:SvgParameter name="stroke-width">1.00</se:SvgParameter>
27   </se:Stroke>
28 </se:PolygonSymbolizer>
```

1. 一定要注意样式格式的正确性，否则容易报错；
2. 图层名称一定要在 UserLayer 里的 Name 节点中声明；
3. 样式的名称要在 Rule 中的 Name 节点进行定义；
4. 样式文本内容不要过多，URL 有字数上限；
5. 可以使用程序生成需要的样式文本，不必从头开始自行编辑。

作为 URL 的参数，在传递之前要将 SLD\_BODY 进行 URLEncode 转换。

在线转换可以使用：<https://www.urlencoder.org/>。或者用 Python 3 的 urllib 模块。

使用 SLD 信息发送请求的示例见 <http://webgis.cn/673>。该 SLD 文档内联在 GET 请求中，SLD\_body 的参数是 URLENCODE 转换后的内容。对其 URL 解析结果为：

```
>>>[print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb2.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetMAP
5 LAYERS=states
6 BBOX=73,3,136,54
7 CRS=CRS:84
```

(下页继续)

(续上页)

```

8 INFO_FORMAT=text/html
9 format=image/png
10 width=200
11 height=150
12 styles=
13 SLD_body=%3C%3Fxml%20version=%221.0%22%20encoding%3D%2 ...

```

上面第 13 行的 `SLD_body` 参数内容很多，书中未完全写出，完整的内容可以通过在线访问查看。

效果如图 6.2，这里使用了与原来不同的颜色，颜色的信息是在 XML 文件中定义的。

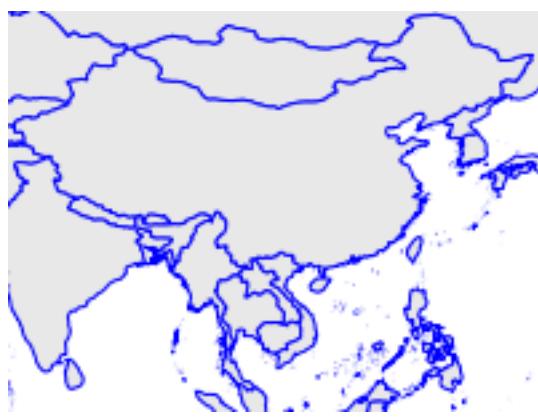


图 6.2: SLD 的样式

这种方式不需要客户发布 SLD 文档到 Web 上，特别适用于不支持 POST 请求方式的客户端。需要记得将 SLD 文件的内容进行 `UrlEncode` 编码。

### 6.3.2 获取图例

使用自定义的信息，访问 <http://webgis.cn/168> 可以得到图例文件。注意这个图例文件，是根据提交的 SLD 返回的图例内容。

URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb2.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetLegendGraphic
5 LAYER=states
6 SLD_VERSION=1.1.0

```

(下页继续)

(续上页)

```
7 format=png
8 SLD_body=%3C%3Fxml%20version=%221.0%22%20encoding%3D%2 ...
```

返回的图例如 图 6.3：



图 6.3：图例文件

## 6.4 在 WMS 中使用样式层描述器：使用服务器端 SLD 文件

这里来介绍使用 SLD 的第二种方法。如果 SLD 文件在服务器端需要使用 SLD 参数进行传递。

### 6.4.1 WMS 对 SLD 的支持

这种方法是客户端必须上载 SLD 文档到 WMS 可能访问的一个 Web 地点，如果有可用的资源可以使用这种方式。需要预先定义一个可访问的 [http://webgis.cn/sld\\_world\\_lyr110.xml](http://webgis.cn/sld_world_lyr110.xml) 资源，该文档中含前述的 `StyledLayerDescriptor` 元素。

访问 <http://webgis.cn/672>，这个是具有 SLD 参数的 GetMap 请求。在使用 SLD 传递样式的情况下，`styles` 参数是可选的。URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb2.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetMAP
5 LAYERS=states
6 BBOX=73,3,136,54
7 CRS=CRS:84
8 INFO_FORMAT=text/html
9 format=image/png
10 width=200
11 height=150
12 SLD=http://webgis.cn/sld_world_lyr110.xml
```

这儿使用的数据与定义的样式基本与上一节一样，但为了后面的说明，将边界进行了加粗：

```

24 <se:Stroke>
25   <se:SvgParameter name="stroke">#0000ff</se:SvgParameter>
26   <se:SvgParameter name="stroke-width">3</se:SvgParameter>
27 </se:Stroke>

```

结果见 图 6.4。



图 6.4：使用 SLD 的多边形样式效果

#### 6.4.2 SLD 中线状效果

再看一下线的效果。定义的样式如下：

```

40 <se:LineSymbolizer>
41   <se:Geometry>
42     <ogc:PropertyName>center-line</ogc:PropertyName>
43   </se:Geometry>
44   <se:Stroke>
45     <se:SvgParameter name="stroke">#ff0000</se:SvgParameter>
46   </se:Stroke>
47 </se:LineSymbolizer>

```

访问 <http://webgis.cn/513> 查看效果，URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[\?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb2.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetMAP
5 LAYERS=states_line
6 BBOX=73,3,136,54

```

(下页继续)

(续上页)

```

7 CRS=CRS:84
8 INFO_FORMAT=text/html
9 format=image/png
10 width=200
11 height=150
12 SLD=http://webgis.cn/sld_world_lyr110.xml

```

这里面指定了 LAYERS=states\_line，得到线状要素的渲染结果（图 6.5）：

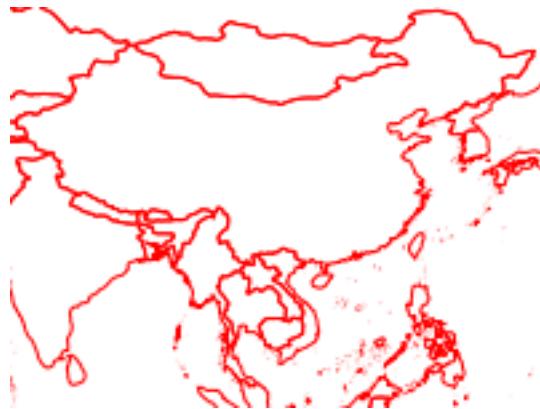


图 6.5：SLD 的线状样式

#### 6.4.3 SLD 参数使用说明

在有 SLD 参数的情况下，LAYERS 参数同样是可选的。

看一下没有 LAYERS 参数的示例 <http://webgis.cn/768>，URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb2.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetMAP
5 BBOX=73,3,136,54
6 CRS=CRS:84
7 INFO_FORMAT=text/html
8 format=image/png
9 width=200
10 height=150
11 SLD=http://webgis.cn/sld_world_lyr110.xml

```

由于未指定 layers 参数，意味着会使用 SLD 文件中定义的图层，图层的顺序按 SLD 中的前后定义。这幅图是由两个层图构成的，线状图层在上。由于线状图层的宽度为 1，面状图

层边界线的宽度为 3，所以线状图层未完全遮住面状图层的边界线，得到 图 6.6 的效果。



图 6.6：不使用 LAYERS 参数的效果

## 6.5 用命名样式绘制地图

更方便使用的 WMS 用“样式层”来描述一幅地图。可以认为一个样式层是由符号化的要素组成的一个透明层。一幅地图是由这些样式层按一定的顺序组成的，通常称这些样式层是“Z 值排序”。用户可以通过增加或删除样式层来定义复杂或简单的地图。一个样式层实际上是一个图层和一个样式的组合。

### 6.5.1 MapServer 对命名样式 (Named Styles) 的支持

MapServer 5.2 中引入了命名样式支持，使用命名样式，WMS 客户端可以使用 WMS 服务器预定义的样式呈现指定的图层。这样的功能简化了传递 SLD 的问题，同时也提供了用户定义样式的自由（尽管有时候有限）。

MapServer 5.2 引入了使用两个新的非强制关键字 CLASSGROUP (在层级别) 和 GROUP (在类级别) 将组分配给在层对象上定义的一系列类的可能性：

一个 WMS 本身可能不知道样式层如何进行有意义的组合，这一切都由客户端决定。WMS 用名称去标识样式和层，其他文档引用这种命名的层和样式，这是目前 WMS 定义样式层的方法。

通过命名样式可以在 WMS GetMap 请求中使用 styles 参数，或者通过 SLD 文档中的 <NamedStyle> 参数指定命名样式。

定义的 Mapfile 为 <http://webgis.cn/mfb5.map>，与命名样式有关的代码如下：

列表 6.1: mfb5.map

```

50 CLASSGROUP "grploy"
51 CLASS
52   NAME "Poly Style 1"
53   GROUP "psty1"
54   STYLE
55     COLOR 232 232 232
56     OUTLINECOLOR 32 32 32
57   END
58 END
59 CLASS
60   NAME "Poly Style 2"
61   GROUP "psty2"
62   STYLE
63     COLOR 204 204 153
64     OUTLINECOLOR 0 255 255
65   END
66 END
67 CLASS
68   NAME "Line Style"
69   GROUP "lsty"
70   STYLE
71     OUTLINECOLOR 255 0 0
72   END
73 END

```

这里有 1 个图层，针对这个图层定义了 3 种样式，并有相应的名称（通过 NAME 关键词定义）：

- psty1：定义了多边形的样式，与本书前面定义的一致；
- psty2：使用了另外的颜色进行多边形样式的定义；
- lsty：只定义了多边形的 OUTLINE 颜色，表现为线状的样式。

访问实例的 URL 为 <http://webgis.cn/661>，对这个 URL 进行解析：

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb5.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetMAP
5 LAYERS=wcountry

```

(下页继续)

(续上页)

```

6 BBOX=-180,-90,180,90
7 CRS=CRS:84
8 INFO_FORMAT=text/html
9 format=image/png
10 width=600
11 height=300
12 styles=psty1

```

这个图已经展示过了，全球的行政界限图，这里不再展示。区别在于请求时使用了 `styles=psty1`。

在呈现时，如果定义了类组，则只使用具有相同组名的类。基于这个概念，WMS/SLD 支持使用类组作为命名样式。每一组类都被视为等同于一个命名样式。

## 6.5.2 使用命名样式

命名样式的定义可以以 SLD 方式返回。使用 <http://webgis.cn/674> 通过 `GetStyles` 请求，内容就不展示了。URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb5.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetStyles
5 LAYERS=wcountry

```

看一下使用另外一种样式的 WMS 请求 <http://webgis.cn/548> 示例，URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb5.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetMAP
5 LAYERS=wcountry
6 BBOX=-180,-90,180,90
7 CRS=CRS:84
8 INFO_FORMAT=text/html
9 format=image/png
10 width=600
11 height=300
12 styles=psty2

```

效果如图 6.7：

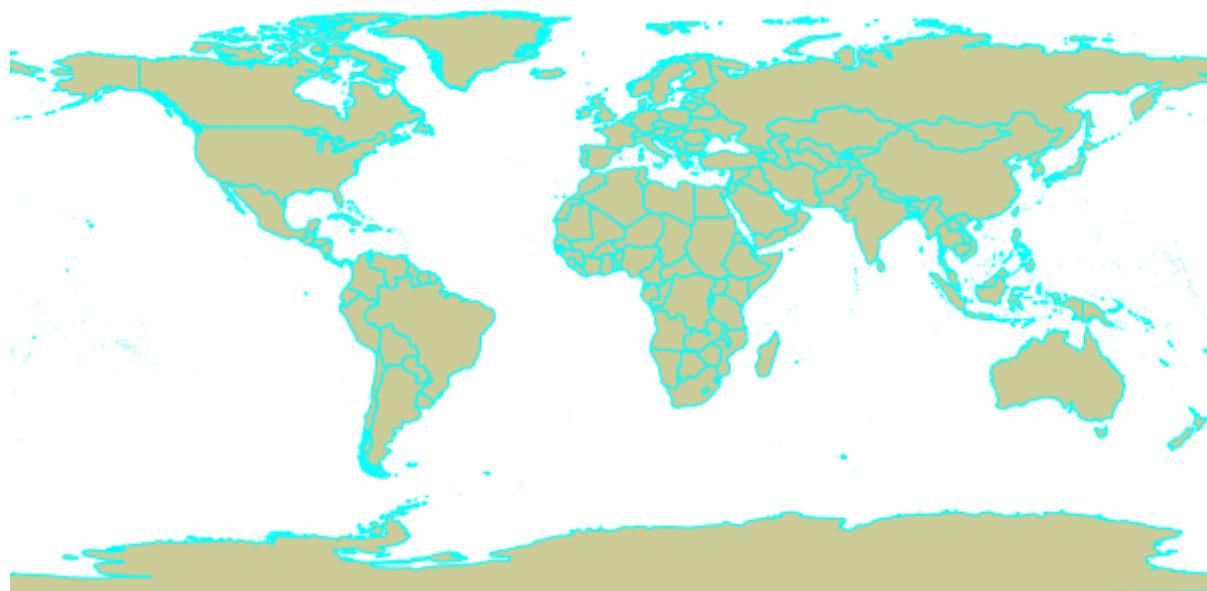


图 6.7：使用命名样式绘制地图

### 6.5.3 多图层与多样式

从概念上看，图层定义了一个要素流，而样式定义了这些要素如何符号化。在一个 WMS 过程中，一幅地图的绘制样式是根据 **LAYERS** 和 **STYLES** 请求参数指定的。图层及样式是按一定的顺序叠加的。一个图层可以用不同的样式绘制多次，以达到用户需要的效果。

<http://webgis.cn/346> 是将行政边界用 group1 和 group2 两种样式绘制了两次：URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[\?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfb5.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetMAP
5 LAYERS=wcountry,wcountry
6 BBOX=-180,-90,180,90
7 CRS=CRS:84
8 INFO_FORMAT=text/html
9 format=image/png
10 width=600
11 height=300
12 styles=psty2,1sty
```

效果如图 6.8，这个中的多边形内部颜色属于 **psty2** 中定义的，而边界颜色则由 **1sty** 定

义。



图 6.8：使用多个命名样式进行叠加

通过 <http://webgis.cn/985> 来请求图例，效果如 图 6.9：

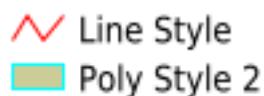


图 6.9：多个命名样式的图例



# 第7章 Web要素服务（WFS）的概念与应用

用户可以从 WMS 服务器方便的获得指定区域内的地图，但是只能获得渲染后的地图。有时候希望获得指定图层的地理坐标和属性等数据，更进一步地，当需要修改数据的时候，WMS 就不能满足需要了。为了规范通过 Web 服务发送矢量数据的过程，OGC 提供了另外一个标准 Web Feature Service（WFS）来应对以上需求。只要服务器与客户端都遵循同一规范，那么源数据可以是任意格式的 WFS。可以根据请求从 Web 服务中将数据以可用的格式发送到客户端。

WFS 规范同样在多个制图应用程序中实现，MapServer 只是其中之一。

WFS 的运行方式与 WMS 相似，但有很大的不同。使用 WMS，仅会返回一张地图图像。在 MapServer 中，可以像绘制其他任何图层一样绘制要素，因为它具有要使用的数据副本。WFS 提供了更大的灵活性，服务器发送回地理坐标数据，例如线，点或面要素，实际的要素数据将返回给客户端。

本章将介绍 WFS 及在 MapServer 中的发布、访问与应用。

## 7.1 Web 要素服务（WFS）的基本概念

WebGIS 服务器除了能返回一张地图图像之外，也可以返回绘制该地图图像所使用的真实地理数据。用户利用这些数据可以创建自己的地图与应用、数据格式转换以及底层的地理操作。WFS 是 Web Feature Service 的简写，也就是网络要素服务，提供地图数据要素一级数据的服务，也能接受客户端请求要素、编辑要素和删除要素等相关操作。WMS 是由服务器绘制地图，将地图图像发送给客户端；而 WFS 是服务器将矢量数据发送给客户端，地图由客户端绘制。

根据 WFS 支持的操作，可以分为基本 WFS（Basic WFS，BWFS）和事务型 WFS（Transaction WFS，TWFS）两类。BWFS 将实现 GetCapabilities、DescribeFeatureType 以及 GetFeature 操作，是一个只读类型的 WFS。

### 7.1.1 WFS 的功能

Web 要素服务（Web Feature Service, WFS）接口提供了对 OGC 简单要素的操作（例如，点、线和多边形），从而使服务器和客户端可以实现在要素层次上进行通信。

一个 WFS 请求由应用于一个或多个要素的查询描述或数据传输操作组成。WFS 的请求在客户端生成，然后被提交到 WFS 服务器；WFS 服务器执行该请求，以 GML 形式或其他格式返回要素集，然后由客户端处理该要素集。

WMS 用一张图片描述服务器处理请求的结果，而 WFS 实现在客户端支持动态查询和访问矢量要素数据和相关属性。这个功能开辟了基于数据的空间分析、建模和其他操作的可能性。

WFS 请求就像许多 GIS 和 RDBMS 包支持的操作，由许多数据变换和查询的操作组成，这些操作应用于 WFS，就可以在 Web 上实现空间数据仓库的功能。与 WFS 经常配合在一起的是 OGC 制定的过滤器编码（Filter Encoding），过滤器编码是用 XML 为查询谓词定义的一个标准编码。使用 XML 编码，可以在空间 Web 服务中定义从一个特定的区域内获取对象的一个查询操作；也可以限制在一个特定区域内具有某个非定间特性的对象实例上执行操作。

WMS 允许一个客户端叠加显示来自互联网上的多个 Web 地图服务器的地图图片。与此类似，Web 要素服务让一个客户获取来自多个 Web 要素服务器的以 GML 编码的地理空间数据。

### 7.1.2 WFS 的基本要求与接口

WFS 标准的基本需求如下：

1. 接口必须用 XML 定义。
2. 接口中的要素必须用 GML 描述，一个最小的 WFS 实现必须可以用 GML 描述要素。
3. 查询语言或谓词要用 XML 定义并符合空间 Web 目录接口中定义的 CQL 标准。
4. 存储地理要素的数据库对客户应用是不透明的，客户应用只能通过 WFS 接口访问数据。
5. 用 Xpath 子集描述对要素特性的引用。

与 WMS 类似，WFS 支持几个操作，这些操作包括：GetCapabilities，DescribeFeatureType，GetFeature，LockFeature（可选），Transaction（可选）。其中 GetFeature 操作用于获取要素数据。

WFS 必须可以描述其功能。它必须表明可以支持的要素类型以及每个要素类型所支持的操作。GetCapabilities 操作的结果是一份描述 WFS 功能的 XML 文档。

同时 WFS 必须可以描述它可以服务的任何要素类型的结构。DescribeFeatureType 操作的功能是向客户提供一个 WFS 所提供的所有要素类型的模式定义。DescribeFeatureType 请求的结果是关于该 WFS 所提供的一个或多个要素类型描述的 XML 文档。

### 7.1.3 事务型 WFS 的概念

WFS 规范同时定义了要素编辑的规则，这就为基于 Web 进行矢量数据编辑打开了大门。通过 WFS 服务对源数据库中的数据进行更改称为事务性 WFS 或 WFS-T。一旦启用了事务功能，WFS 客户端便可使用事务性 WFS 方法对地理数据库中的数据进行更改。MapServer 本身不支持 WFS-T，作为套件的一部分，有 TinyOWS 可供选择。在本书中对这部分的具体技术不多作介绍，仅在下面把概念说明一下。

TWFS 操作以 HTTP 作为分布式计算平台，支持对地理要素的插入 (INSERT)、更新 (UPDATE)、删除 (DELETE)、查询 (QUERY) 操作。

处理一个 WFS 请求协议的过程如下：

1. 客户请求来自 WFS 功能元数据。这份元数据包含 WFS 支持的所有操作的描述和可以服务的所有要素的列表。
2. 客户（可选）向 WFS 请求可以服务的一个或多个要素类型的定义（文档）。
3. 根据要素类型的定义，客户生成由该文档指定格式的请求。
4. 请求被提交到一个 Web 服务器。
5. Web 服务器调用 WFS 来接收并服务该请求。
6. 当 WFS 完成对请求的处理，它将生成一个状态报告并返回给客户，如果发生错误，就生成异常报告。

一个事务型的 WFS 将支持基本 WFS 的所有操作，另外还将实现事务操作。事务可选的 Transaction 操作和 LockFeature 操作。

简而言之，TWFS 对地理要素操作的主要功能包括创建、删除、更新、查询要素，也就是数据库的增、删、改、查（CRUD）。

## 7.2 在 MapServer 中配置使用 WFS

上一章说明了配置 MapServer 支持 WMS，这一章则说明如何让 MapServer 支持 WFS。目前，MapServer 支持 WFS 的主要版本包括：1.0.0、1.1.0、2.0.0，暂不支持 3.0.0。

通过 URL 调用 MapServer 提供的 WFS，WFS 默认会返回 GML 格式的结果，在浏览器中可看其调用结果。

## 7.2.1 配置 MapServer 以支持 WFS

MapServer 发布服务都是通过 Mapfile 来配置的，WFS 服务也不例外，和配置 WMS 类似，发布 WFS 要在 Mapfile 配置文件中的 METADATA 部分添加关于 WFS 的键值对。只有在图层满足如下条件的时候，MapServer 才会包含 WFS 的功能：

- 数据源是矢量数据，如 Shapefile、OGR 支持的矢量格式、PostGIS、SDE（ArcSDE）；
- 图层名称必须设置；
- 图层数据空间类型必须是 point、line、polygon 三者之一，也就是矢量数据。
- 关键词 wfs\_onlineresource 和 wfs\_enable\_request 必须设置。

这里使用的 Mapfile 为 <http://webgis.cn/mft2.map>。在发布 WMS 的 Mapfile 配置基础上，添加 WFS 配置。内容如下：

```

10 METADATA
11   "wfs_title" "world country"
12   "wfs_onlineresource" "http://webgis.cn/cgi-bin/mapserv?map=/owg/mft2.
13   ↳map"
14   "wfs_srs" "EPSG:4326"
15   "wfs_abstract" "This text describes your wfs service"
16   "wfs_enable_request" "*"
17   "wms_enable_request" "*"
18   "ows_keywordlist" "world"
19   "gml_include_items" "all"
END

```

相对于只配置 WMS 的 Mapfile，主要的改变在于 MAP 对象的 METADATA 子对象和 LAYER 对象的 METADATA 子对象，在 MAP 对象的 METADATA 中，增加的键值对包括：

- wfs\_title，必需，GetCapabilities 请求中作为服务响应的 XML 文档的 title 元素返回；
- wfs\_onlineresource，必需，提供 WFS 的服务器地址；
- wfs\_srs，数据的空间参考系（SRS，Spatial Reference System）；
- wfs\_abstract，GetCapabilities 请求中作为服务响应的 XML 文档的 Abstract 元素返回；
- wfs\_enable\_request，表示配置 MapServer 支持 WFS 的操作。
- gml\_featureid，图层对应的要素 ID 属性名，可选；

除了 Map 对象，LAYER 对象也需要设置 METADATA 节点。配置和 MAP 对象的 METADATA 有很多重复，如果不设置，就会继承 MAP 对象的 METADATA 的配置，如果设置就会使用 LAYER 的配置，这在一个 MAP 对象包含多个 LAYER 时非常有用。

## 7.2.2 请求 GetCapabilities 接口

GetCapabilities 接口会返回 WFS 的功能元数据。WFS 的 GetCapabilities 的调用格式与 WMS 差不多, 这里来具体看一下。

### 支持的操作 WFS GetCapability

测试 WFS 服务器比测试 WMS 服务器容易一点, 因为需要更少的参数。要从测试层获取数据, 必须将请求的类型更改为 &request=GetFeature , 并基于功能文档中该层的 FeatureType 名称添加 &typename=countries 。

GetCapabilities 操作主要是请求 MapServer WFS 服务支持的操作, 请求的 URL <http://webgis.cn/731> 及其携带参数如下:

URL 的解析如下:

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]  
0 http://webgis.cn/cgi-bin/mapserv  
1 map=/owg/mft2.map  
2 SERVICE=WFS  
3 VERSION=2.0.0  
4 REQUEST=GetCapabilities
```

参数的说明如下:

- REQUEST=GetCapabilities , 请求的操作名;
- SERVICE=WFS , 请求的服务类型;
- VERSION=2.0.0 , WFS 的版本;
- map=/owg/mft2.map , 相应的 Mapfile 配置文件。

结果以 XML 文件的形式显现, 与 WMS 功能文档格式类似。得到的响应 (部分) 如下:

该 Mapfile 所需的关键信息是 OnlineResource 和 FeatureType 的名称。

```
232 <FeatureTypeList>  
233   <FeatureType>  
234     <Name>ms:world-city</Name>  
235     <Title>worldcity</Title>  
236     <DefaultCRS>urn:ogc:def:crs:EPSG::4326</DefaultCRS>  
237     <OutputFormats>  
238       <Format>application/gml+xml; version=3.2</Format>  
239       <Format>text/xml; subtype=gml/3.2.1</Format>  
240       <Format>text/xml; subtype=gml/3.1.1</Format>
```

(下页继续)

(续上页)

```

241   <Format>text/xml; subtype=gml/2.1.2</Format>
242 </OutputFormats>
243 <ows:WGS84BoundingBox dimensions="2">
244   <ows:LowerCorner>-176.151565551758 -54.7919998168945</
245   ↪<ows:LowerCorner>
246     <ows:UpperCorner>179.221893310547 78.1999969482422</ows:UpperCorner>
      </ows:WGS84BoundingBox>
```

请注意，该文件充满了 MapServer 自动生成的许多条目，并不是 Mapfile 中 WFS METADATA 对象设置的内容。MapServer 甚至计算了 LatLongBoundingBox。它还基于图层的名称设置创建了 NAME``元素，如果需要其他设置，可以使用 ``wfs\_name 参数在 METADATA 部分中进行设置。

## 进一步查看

返回的数据是 XML 格式，由于数据太多，只贴出前两层。

```

<wfs:WFS_Capabilities xmlns:gml="http://www.opengis.net/gml/3.2" ...>
  <ows:ServiceIdentification>...</ows:ServiceIdentification>
  <ows:ServiceProvider>...</ows:ServiceProvider>
  <ows:OperationsMetadata>...</ows:OperationsMetadata>
  <FeatureTypeList>...</FeatureTypeList>
  <fes:Filter_Capabilities>...</fes:Filter_Capabilities>
</wfs:WFS_Capabilities>
```

会看到了这样几个节点：ServiceIdentification，ServiceProvider，OperationsMetadata，FeatureTypeList 和 Filter\_Capabilities。前三个节点是 OWS 定义的，后面两个是 WFS 追加的。这些节点从字面上就能看出他们的作用。

查看节点的 FeatureTypeList。

```

<FeatureTypeList>
  <FeatureType>
    <Name>ms:world-country</Name>
    <Title>worldcountry</Title>
    <DefaultCRS>urn:ogc:def:crs:EPSG::4326</DefaultCRS>
    <OutputFormats>
      <Format>application/gml+xml; version=3.2</Format>
      <Format>text/xml; subtype=gml/3.2.1</Format>
      <Format>text/xml; subtype=gml/3.1.1</Format>
      <Format>text/xml; subtype=gml/2.1.2</Format>
    </OutputFormats>
```

(下页继续)

(续上页)

```

<ows:WGS84BoundingBox dimensions="2">
    <ows:LowerCorner>-180 -90</ows:LowerCorner>
    <ows:UpperCorner>180 83.62741851</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
</FeatureTypeList>

```

很明显，这个节点包含服务器发布的 FeatureType（FeatureType 节点）。以及对这些 FeatureType 的操作（Operations 节点）。FeatureType 都有一个 Name 节点，可以用它的值来指定 FeatureType。DefaultSRS 节点说明 FeatureType 在使用哪个坐标系。OutputFormats 说明 FeatureType 用哪些格式返回数据。WGS84BoundingBox 就是这个 FeatureType 的范围的经纬度。

另外一个值得关注的就是 Filter\_Capabilities 节点。这个节点显示了服务器对 Filter 的支持情况。

在 WFS 标准里面，Capability 数据块的结构还要复杂得多。目前了解这些也就足够了。有了这些数据就知道该如何访问一个 WFS 服务器。

```

<ows:OperationsMetadata>
    + <ows:Operation name="GetCapabilities"></ows:Operation>
    + <ows:Operation name="DescribeFeatureType"></ows:Operation>
    + <ows:Operation name="GetFeature"></ows:Operation>
    + <ows:Operation name="GetPropertyValues"></ows:Operation>
    + <ows:Operation name="ListStoredQueries"></ows:Operation>
    + <ows:Operation name="DescribeStoredQueries"></ows:Operation>
    + <ows:Parameter name="version"></ows:Parameter>
    ...
</ows:OperationsMetadata>

```

### 7.2.3 请求 DescribeFeatureType 接口

在操作数据库的时候，有时需要了解数据库的结构，在使用 WFS 的时候也存在同样问题。有时候需要知道某个 FeatureType 有哪些属性以及分别是什么类型，这时就需要 DescribeFeatureType 方法。

一个典型的 DescribeFeatureType 调用可以打开 <http://webgis.cn/532> 查看，对 URL 进行解析：

```

>>> [print(i,p) for i,p in enumerate(url.split('&'))]
0 http://webgis.cn/cgi-bin/mapserv?map=/owg/mft2.map
1 service=WFS
2 VERSION=2.0.0

```

(下页继续)

(续上页)

```
3 request=DescribeFeatureType
4 TypeName=world-city
```

访问网址后，返回内容是 XML 文本（可能以 mapserv 程序方式返回），使用文本程序打开，返回结果是这样的：

```
19 <extension base="gml:AbstractFeatureType">
20   <sequence>
21     <element name="msGeometry" type="gml:GeometryPropertyType" ...>
22     <element name="CITY_NAME" minOccurs="0" type="string"/>
23     <element name="GMI_ADMIN" minOccurs="0" type="string"/>
24     <element name="ADMIN_NAME" minOccurs="0" type="string"/>
25     <element name="FIPS_CNTRY" minOccurs="0" type="string"/>
26     <element name="CNTRY_NAME" minOccurs="0" type="string"/>
27     <element name="STATUS" minOccurs="0" type="string"/>
28     <element name="POP_RANK" minOccurs="0" type="string"/>
29     <element name="POP_CLASS" minOccurs="0" type="string"/>
30     <element name="PORT_ID" minOccurs="0" type="string"/>
31     <element name="LABEL_FLAG" minOccurs="0" type="string"/>
32   </sequence>
33 </extension>
```

这个数据块携带 FeatureType 的结构信息，为以后的访问提供了支持。要注意一个叫 msGeometry 属性，它的类型是 gml:GeometryPropertyType，这就是 Feature 的几何数据的属性名，可以通过这个名称来获得 Feature 的地理数据。WFS 用这种方法来统一数据的访问形式，避免了需要特殊处理的情况。

#### 7.2.4 使用 OWSLib 访问

使用 OWSLib 访问，连接后使用 getcapabilities() 函数来获取功能元数据。MapServer 暂未支持 WFS 3.0。在 WFS 2.0 中，如下：

```
>>> wfs20 = WebFeatureService(
...     url='http://webgis.cn/cgi-bin/mapserv?map=/oswebgis/mft2.map',
...     version='2.0.0'
... )
>>> response = wfs20.getcapabilities()
```

返回的内容需要进行解码：

::

```
>>> reponse.read().decode()
```

上面返回的结果，即为 XML 的内容，这里就不多展示了。

## 7.3 WFS 的 GetFeature 请求与响应

可以通过 URL 手动请求 WFS 源。WMS 提供了一张地图图像；WFS 提供了可使用的原始数据。

该 URL 看起来与 WMS URL 相似，但是通常要简单得多。由于 WFS 会发送原始数据以供处理，因此无需向服务发送太多的请求参数，例如地图的大小或图像格式。

### 7.3.1 支持的操作 WFS GetFeature

GetFeature 方法是 WFS 主要操作，它的用途是获得要素，即矢量的数据。数据默认使用地理标记语言（Geography Markup Language，GML）返回。GML 可以同时包含图形与属性信息。

访问 <http://webgis.cn/742> 来发起 GetFeature 请求，URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mft2.map
2 service=WFS
3 VERSION=2.0.0
4 request=GetFeature
5 TypeName=world-city
```

上面的 GET 请求中相关参数通过 URL 传递给 MapServer，MapServer 会去查询响应的图层，并找到符合条件的要素，以 GML 格式返回。来看看 URL 中的参数分别代表什么：

- REQUEST=getfeature，请求的操作；
- SERVICE=WFS，请求的服务；
- TYPE\_NAME=world-city，表示图层名是 world-city 的要素；
- VERSION=2.0.0，WFS 的版本；
- map=/owg/mft2.map，Mapfile 配置文件的位置。

这个调用意味着返回 world-country 的所有数据，会得到一个很大的 XML 文档。如果用浏览器查看，可能会导致停止响应。

### 7.3.2 使用参数限制数目

为了能方便地查看结果，使用 `maxfeatures=2` 或者 `count=2` 来限制一下要素的数目：

<http://webgis.cn/741>

解析

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mft2.map
2 service=WFS
3 VERSION=2.0.0
4 request=GetFeature
5 TypeName=world-city
6 maxfeatures=2
```

结果同样是一个 XML 文档，当然内容要少得多。可以查看一下要素的坐标与属性：

```
24 <ms:msGeometry>
25   <gml:Point gml:id="world-city.NOR-BSK.1" srsName="urn: ...
26     <gml:pos>59.700001 10.200000</gml:pos>
27   </gml:Point>
28 </ms:msGeometry>
29 <ms:CITY_NAME>Drammen</ms:CITY_NAME>
30 <ms:GMI_ADMIN>NOR-BSK</ms:GMI_ADMIN>
31 <ms:ADMIN_NAME>Buskerud</ms:ADMIN_NAME>
32 <ms:FIPS_CNTRY>NO</ms:FIPS_CNTRY>
```

这里使用的数据为 `world-city`，点状要素的坐标比较简单，属性与原始矢量数据中一致，但这里没有全列出来。可以试一下请求 `world-contry` 要素集，看一下多边形要素返回的结果。

这里使用的 Mapfile 为 <http://webgis.cn/mft2.map>。

### 7.3.3 根据 ID 请求单个要素

除了限定数目，也可以直接指定返回哪个要素。要实现此项功能，需要在 Mapfile 中按如下配置：

```
38 "gml_include_items" "all"
39 "gml_featureid" "GMI_ADMIN"
```

"`gml_include_items`"，是可选字段，设置为 "`all`" 表示会返回所有的属性，"`gml_featureid`" 设置要素的 ID，这里 `GMI_ADMIN` 是矢量数据中的字段名称；作为

ID，在记录中应该是唯一的；但可以并非是索引字段，数据类型也没有限制。

作为比较，可以看一下设置 "gml\_featureid" "GMI\_ADMIN" 与否对于 GetFeature 返回结果中 gml:id 属性的影响。

访问 <http://webgis.cn/389> 来请求单个要素，URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mft2.map
2 service=WFS
3 VERSION=2.0.0
4 request=GetFeature
5 TypeName=world-city
6 FeatureId=world-city.CHN-BJN
```

这次返回的数据是单个的要素。在上述请求中，FeatureId=world-city.CHN-BJN 表示需要获取的要素的 ID 为 CHN-BJN。仔细查看上述请求返回的结果，同样可在其中找到图形数据与属性数据。

### 7.3.4 根据空间范围进行检索

除了根据属性进行查询，还可以针对空间范围进行检索。

来看另外一个调用 <http://webgis.cn/744>，这一次使用了 BBOX 参数作为空间范围的检索条件。URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfw1.map
2 service=WFS
3 VERSION=2.0.0
4 request=GetFeature
5 TypeName=world-country
6 BBOX=30,100,40,110,EPNG:4326
```

注意 BBOX 定义的形式，以下、左、上、右定义，并且需要声明坐标的空间参考。结果同样以 GML 格式返回。

## 7.4 WFS 中使用 Filter

在真实的应用中，除了上一节介绍的几种方法，还需要更强大的检索机制来获取数据。这样就需要选择的功能。

### 7.4.1 Filter Encoding 标准规范

WFS 请求 URL 中允许使用属性来选择/过滤要素，这里用到的技术是 OGC 的另一个标准 Filter Encoding（简称 FE），这个标准基于 XML 和 KVP（Extensible Markup Language）编码。

OGC Filter 定义了一些操作符，从而实现了查询要素集合的接口。其作用是构建一个表达式，应用程序可以将它转换为查询语言，检索要素集合后返回有效的要素子集。该子集也可能以某种特定方式排序。WFS Filter 结果同样是 XML/GML 文件。

FE 编码被认为是与系统无关的，因为使用当今众多的 XML 工具，XML 编码的投影、选择和排序子句可以很容易地被验证、解析，然后转换成检索或修改存储在某个持久对象存储中的资源所需的任何目标查询语言。例如，由投影、选择和排序子句组成的 XML 编码查询可以转换为 SQL 语句，以获取存储在基于 SQL 的关系数据库中的数据；同样，为了从 XML 文档中检索数据，同样的 XML 编码查询表达式也可以很容易地转换为 XQuery 表达式。

FE 标准中描述的投影、选择和排序子句的 XML 和 KVP 编码是一些 WEB 服务可以一起使用或单独使用的公共组件。任何需要从 WEB 可访问存储库查询对象的服务都可以使用本国际标准中描述的查询表达式的 XML 和 KVP 编码。例如，在 ISO 19142/OGC 09-025r2 中定义的 GetFeature 操作使用从 FE 标准定义中派生的元素来编码查询表达式。

还要注意 Filter Encoding 也有不同的版本，目前主要是 2.0，以及 2.0.2 修订版本。

### 7.4.2 Filter 的三种操作符

在 Filter 中分为空间操作（Spatial operators），比较操作（Comparison operators）和逻辑操作（Logical operators）三类。空间操作主要针对几何数据，比较操作主要针对一般的属性数据，逻辑操作被用来组合前两种操作的结果。

#### 空间操作符

空间操作符定义了几何图形之间的关系，是二元操作符，或一元操作符，假设两个对象为 A 与 B。关于空间操作符或空间关系的概念，可以了解一下“九交模型”的概念来进一步了解。操作符与意义如下：

表 7.1: 部分开源 GIS 项目列表

名称	含义
Equals	A 等于 B
Disjoint	A 与 B 不相交
Intersects	A 与 B 相交 (存在交集)
Touches	A 与 B 的边界相接 (但内部不交集)
Within	A 在 B 的内部
DWithin	A 在 B 指定的距离 d 之内
Overlaps	A 与 B 有重叠部分
Crosses	A 穿过 B
Contains	A 包含 B
Beyond	A 在 B 指定的距离 d 之外
BBOX	在 BBOX 定义的范围之内

例如，如下定义：

```
<BBOX>
<PropertyName>geom</PropertyName>
<Envelope srsName="EPSG:4326">
  <lowerCorner>-40 -30</lowerCorner>
  <upperCorner>35 45</upperCorner>
</Envelope>
</BBOX>
```

注意此处的定义，与 节 7.3.4 中的 BBOX 类似，但是定义方式完全不一样。

### 比较操作符

比较操作符定义了属性的操作方式，是二元操作符，或一元操作符，假设两个对象为 A 与 B，他们有：

表 7.2: 部分开源 GIS 项目列表

名称	含义
PropertyIsEqualTo	A 与 B 相等
PropertyIsNotEqualTo	A 与 B 不相等
PropertyIsLessThan	A 小于 B
PropertyIsGreaterThan	A 大于 B
PropertyIsLessThanOrEq	A 小于等于 B
PropertyIsGreaterThanOrEqualTo	A 大于等于 B
PropertyIsLike	A 与 B 利用通配符进行模糊匹配
PropertyIsNull	A 的属性为空
PropertyIsBetween	A 的属性在... 之间

下面给出一个不等于的例子。关键字 PropertyIsNotEqualTo , PropertyName 声明了矢量数据的字段名称，Literal 则定义了要进行比较的值。

```
<PropertyIsNotEqualTo>
  <PropertyName>field-name</PropertyName>
  <Literal>value1</Literal>
</PropertyIsNotEqualTo>
```

## 逻辑操作符

逻辑操作符定义了组合前面两种操作的方式。操作符有 And, Or , Not ，分别表示“与”，“或”，“取反”。

下面的代码将前面两个查询条件组合在一起，使用 AND 进行连接：

```
<Filter>
  <And>
    <PropertyIsNotEqualTo>
      <PropertyName>field-name</PropertyName>
      <Literal>value1</Literal>
    </PropertyIsNotEqualTo>
    <BBOX>
      <PropertyName>geom</PropertyName>
      <Envelope srsName="EPSG:4326">
        <lowerCorner>-40 -30</lowerCorner>
        <upperCorner>35 45</upperCorner>
      </Envelope>
    </BBOX>
  </And>
</Filter>
```

(下页继续)

(续上页)

```
</And>
</Filter>
```

### 7.4.3 MapServer 对 Filter Encoding 的支持

一个 WFS 必须可以支持获取要素实例的请求。GetFeature 操作用来从一个 Web 要素服务器上获取所需的要素，通常返回包含了请求结果的 XML 文档。另外，一个客户应能够指定需要哪些要素特性并且能够进行空间或非空间上的查询限定。

实例的 Mapfile 使用 <http://webgis.cn/mft2.map>。对于 FE，MapServer 提供了相对有限的支持。下面的例子描述了 PropertyIsEqualTo 的定义，用来选择 STATUS 属性为 National capital 的要素：

```
<Filter>
  <PropertyIsEqualTo>
    <PropertyName>STATUS</PropertyName>
    <Literal>National capital</Literal>
  </PropertyIsEqualTo>
</Filter>
```

访问示例通过 <http://webgis.cn/359> 查看。注意对 FE 的编码，与 [节 6.3.1 内联 SLD](#) 一样，也需要进行 URLEncode 转换。这样构造出来的 URL 一般都是比较长的。下面对 URL 解析中也没有完全打印出来：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mft2.map
2 service=WFS
3 VERSION=2.0.0
4 request=GetFeature
5 TypeName=world-city
6 Filter=%3CFilter%3E%3CPropertyIsEqualTo%3E%3CPropertyName%3 ...
```

上面查询同样返回 GML 格式的要素子集。可以检查所有的要素属性，都有 `<ms:STATUS>National capital</ms:STATUS>`。

```
28 <ms:CITY_NAME>Dalap-Uliga-Dorrit</ms:CITY_NAME>
29 <ms:GMI_ADMIN>MHL</ms:GMI_ADMIN>
30 <ms:ADMIN_NAME>Marshall Is.</ms:ADMIN_NAME>
31 <ms:FIPS_CNTRY>RM</ms:FIPS_CNTRY>
32 <ms:CNTRY_NAME>Marshall Is.</ms:CNTRY_NAME>
33 <ms:STATUS>National capital</ms:STATUS>
```

(下页继续)

(续上页)

```

34 <ms:POP_RANK>7</ms:POP_RANK>
35 <ms:POP_CLASS>Less than 50,000</ms:POP_CLASS>
36 <ms:PORT_ID>0</ms:PORT_ID>
37 <ms:LABEL_FLAG>0</ms:LABEL_FLAG>

```

## 7.5 将 WFS 图层添加到 MapServer 地图文件

本章前面一直在说明的都是配置 MapServer 以用作 WFS 服务器，这样就很容易使其他人可以访问自己的矢量数据，

与 WMS 的应用一样，MapServer 也可以用作 WFS 客户端，调用其他的 WFS 进行中转，重新对外提供服务。

### 7.5.1 MapServer 用作 WFS 服务器的说明

这里服务器的 Mapfile 使用前面的 <http://webgis.cn/mft2.map>。仅矢量数据层可用于 WFS，GML 支持矢量数据类型，而不是图像；如果需要提供图像，则应使用 WMS 或 WCS。

如果只想通过 WFS 提供数据，在服务器端可以不用 CLASS 对象来配置样式，因为不需要进行制图，仅将要素数据发送回客户端。如果需要的话可以设置图层来绘制地图，并提供 WFS 或 WMS 数据。

### 7.5.2 在 Mapfile 中进行客户端定义

将 mft2.map 中的 WFS 添加为 Mapfile 中的图层非常简单。与 WMS 中使用类似，请求的一些参数需要在 METADATA 对象中进行配置。

如果 WFS 源是更早的 MapServer 版本，则必须包括最后一个参数: wfs\_request\_method；否则，此参数是可选的。这对于其他软件发布的 WFS 也可能是必需的。

在 Mapfile 中将 WFS 源用作图层，Mapfile 文件为 <http://webgis.cn/mft6.map>，添加如下定义：

```

32 LAYER
33   NAME "wcity_wfs"
34   TYPE POINT
35   OFFSITE 0 0 0
36   STATUS ON
37   CONNECTIONTYPE WFS
38   CONNECTION "http://webgis.cn/cgi-bin/mapserv?map=/owg/mft2.map&"
39   METADATA
40     "wfs_version" "1.0.0"

```

(下页继续)

(续上页)

```

41      "wfs_srs" "EPSG:4326"
42      "wfs_typename" "world-city"
43      "wfs_request_method" "GET"
44      "wfs_connectiontimeout" "60"
45
46      END
47      PROJECTION
48          "init=epsg:4326"
49      END
50      CLASS
51          NAME "cities"
52          SYMBOL "circle"
53          SIZE 6
54          COLOR 255 255 0
55      END
END

```

用作 WFS 客户端时，在第 37 行声明 CONNECTIONTYPE WFS，并同时定义 CONNECTION 的值为 MapServer 的 WFS 源。

与 WFS 源连接的信息在 METADATA 对象中定义。图层的名称使用 wfs\_typename 进行声明。

注意第 49 - 54 行使用 CLASS 对象重新进行了样式定义。

另外在 mft6.map 中还定义了本地的图层 wcountry。

### 7.5.3 通过浏览器查看

打开 <http://webgis.cn/468> 查看，URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mft6.map
2 SERVICE=WMS
3 VERSION=1.3.0
4 REQUEST=GetMAP
5 LAYERS=wcountry,wcity_wfs
6 BBOX=73,3,136,54
7 CRS=CRS:84
8 INFO_FORMAT=text/html
9 format=image/png
10 width=200
11 height=150
12 styles=

```

这里请求的图层为使用 `wcity_wfs` 命名的远程数据与本地数据 `wcountry`。注意此处使用了 `BBOX` 参数限定数据的范围。如果通过 WFS 一次请求的数据过多，可能会导致失败。

结果如 图 7.1：

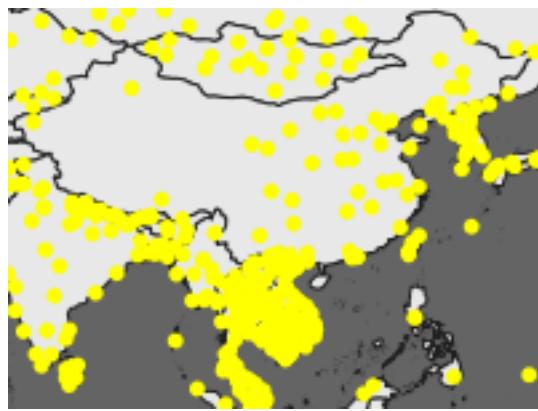


图 7.1：使用后的结果

为了限制返回要素的数目，还可以使用下面的代码：

```
"wfs_maxfeatures" "100"
```

限制了返回要素的最大数目。

# 第8章 Web 覆盖服务 (WCS) 的概念与应用

WFS 是在 Web 上发布矢量数据服务，而对于栅格数据服务，OGC 制定的则是 WCS (Web Coverage Services，网络覆盖服务)。通过 WCS 服务获取的栅格数据集被称为覆盖，这里的覆盖一般是栅格数据，但也包括 TIN 等其他数据格式。WCS 返回的数据可作为分析和建模操作的输入参数。这与 OGC WMS 服务形成鲜明对比，后者仅返回数据的图片。

前端库 Leaflet 与 OpenLayers 中都没有对 WCS 的内置支持。如果自己是服务/数据提供者，并且覆盖范围数据是二维的，那么可以考虑同时提供数据的 WMS。如果自己不是服务提供者，并且只有一个 WCS 端点可以使用，那么将不得不编写自己的代码来生成地图结果。类似的，如果自己有多维覆盖数据，希望提供二维数据，也得这样处理。

与 WMS、WFS 不同，目前 MapServer 只支持作为 WCS 服务器端，不能作为 WCS 客户端。栅格数据数据量较大，通过互联网分布式实时调用目前尚不现实。

## 8.1 Web 覆盖层服务

Web 覆盖服务 (WCS) 是开放地理空间联盟 (OGC) 开发和发布的一个标准。该服务基于客户机-服务器原理，允许在 Internet 环境中以“Coverage”的形式共享地理空间数据，这里的覆盖意指表示空间变化现象的地理信息。WCS 能够返回其时空域中任意指定点的数据，其形式易于输入到模型中使用，通常以 GeoTIFF、NITF 和 HDF 格式分发，并提供和使用下载服务的条款。

### 8.1.1 WCS 基本概念和特征

WCS 以“Coverage”的形式实现了栅格影像数据集的共享，与 WFS 类似。WCS 允许访问描述与地理位置相关的值和特性的地理覆盖层，而不是生成地图图片。也就是说，WMS 和 WCS 都提供基于栅格的信息的生成和分发，但是 WMS 返回的影像是一个由像素值组成的数组，而 WCS 返回一个要素的集合，它提供给客户端感兴趣的值，如类型、温度、降水量、反照率等。与 WMS 只实现了可视化不同，WCS 提供包括评价、操作和联合多个覆盖层来回答特定问题。如果要进行可视化显示，这些值可以作进一步处理。这种访问原始地理空间数据的功能对于客户端处理、多值覆盖和作为科学计算模型的输入是非常必要的。

狭义的说，可以认为 WCS 中发布的 Coverage 数据就是影像数据或者栅格数据。广义上说，Coverage 数据主要是指时空现象的数字表示，可以是多维的，例如一维的传感器时间序列信号，

二维的卫星影像，三维的 x/y/t 格式的“图像时间序列”，或者 x/y/z 格式的三维地理断层数据，四维的 x/y/z/t 格式的数据，数据中既包含 x、y、z 坐标信息，还包含时刻信息。常见的 Coverage 数据有：航拍图、遥感分类数据、多光谱反演数据、DEM 等，常用的数据格式有：GeoTIFF、GRIB2、JPEG2000、netCDF。

### 8.1.2 WCS 在数据共享与分发中的作用

地理科学数据共享一直都是一个难题。除了少数的一些组织机构，很难见到完全公开共享的数据集或数据库。尤其是电子数据使用协议的约束性不强，数据发布方对于数据的使用也有各方面的担心。WCS 作为一种技术提供了数据共享与发布的一种方式。

WMS 和 WCS 都提供生成和分发基于栅格的空间信息。WMS 返回由像素阵列组成的“影像”，因而只提供空间数据可视化和有限的查询。虽然 WCS 与 WMS 从服务器上返回的都是图像，但是它们之间一个重要的区别是 WCS 能够返回更多的详细信息，包括有价值的元数据和多种格式的数据。WCS 返回的数据可作为分析和建模操作的输入参数，这样就实现了在线按需获取必要的数据。此外，如果 WCS 可以返回一个多维格式，那么可以实现更为精确的查询。

### 8.1.3 WCS 的主要接口

WCS 提供 GetCapabilities、DescribeCoverageLayer 和 GetCoverage 三个基本接口，各接口能够实现的操作如下：

- GetCapabilities 请求服务级别元数据和数据的简要描述。
- DescribeCoverage 请求一个或多个覆盖的完整描述。
- GetCoverage 以某种格式请求覆盖数据。

#### GetCapabilities 接口

GetCapabilities 接口用来描述关于一个 WCS 服务本身的信息和所提供的数据集的信息。GetCapabilities 返回 XML 文档，该文档由服务元数据（Service-Metadata）、操作标记（OperationSignatures）、内容元数据（ContentMetadata）和附加功能（AdditionalCapabilities）4 部分组成，其中操作标记是一个 WSDL 服务描述，引入了许多模式用来定义请求的语法。内容元数据引入特定的 WCS 服务内容有关的覆盖层列表（CoverageLayerList）模式。

在覆盖层列表模式中定义了栅格覆盖层（GridCoverage）、TIN 覆盖层（TINCoverage）、多点覆盖层（MultiPointCoverage）、曲线段覆盖层（Segmented-CurveCoverage）和泰森多边形覆盖层（ThiessenPolygonCoverage）共 5 种不同的覆盖层图层（CoverageLayer），这 5 种不同的覆盖层图层都是从抽象覆盖层图层（AbstractCoverageLayer）继承而来。抽象覆盖层图层定义了所有覆盖层图层公共的特性，例如 LayerID、Title、SRS、BBOX 和 MetaDataURL 等。

## DescribeCoverageLayer 接口

`DescribeCoverageLayer` 操作允许客户端请求关于一个或多个覆盖层图层的完整描述。一个 WCS 的功能返回 XML 格式的文档，包含了对所有可用的覆盖层图层的详细描述，但是查询一个或少数覆盖层图层的信息有时非常必要，并且对于将 WCS 集成到只存储了图层名和图层摘要信息的目录服务非常有用。将来 WCS 可能只通过 `DescribeCoverageLayer` 操作来描述关于覆盖层图层的完整描述信息。

## GetCoverage

`GetCoverage` 操作允许从一个覆盖层图层获取一个或多个覆盖层的数据。在请求参数中通过 `RANGE` 参数列表指定需要获取的覆盖层。如果某个 `RANGE` 由一个复合变量组成，还需要通过 `RANGE:PARAM` 参数进行过滤。如 `radiance:band=1,5,3`（表示波段索引分别为 1、5、3 的 3 个波段）或 `count:age=<V18`（表示小于 18 岁年龄的人口分布）。对于栅格覆盖层还需要指定 `(X, Y, Z)` 方向栅格单元的精度，即像元大小。

一个有效的 `GetCoverage` 的返回值是从 WCS 请求后返回的一个栅格文件，该数据集具有指定的空间参照系、边界矩形、大小和格式。

## 8.2 在 MapServer 中定义与使用 WCS

在 MapServer 版本 xx 中，开始提供 WCS 功能。MapServer 支持的 WMS 版本包括 1.0.0, 1.1.1, 2.0.1，在 MapServer 6.0 后实现了对 WCS 2.0 规范的支持。现行的 WCS 版本包括 2.0 与 2.1；而不推荐的版本包括 0.7、1.0、1.1、2.0。所以在本书的介绍中仅介绍 2.0.1 版本。

### 8.2.1 Mapfile 定义

与配置 WMS 和 WFS 一样，MapServer 发布 WCS 同样需要修改对应的 Mapfile 文件，增加全局的 WEB 对象的 METADATA 子对象的键值对，但是 WFS 都是针对矢量数据，如 ESRI 的 Shapefile 格式，而 WCS 主要是针对栅格格式，所以 Mapfile 中图层 TYPE 需要设置为 RASTER，数据源也要使用几种支持的格式（如遥感影像，栅格数据，TIN 等）。

Mapfile 文件需要满足如下条件，对应的图层服务才会拥有 WCS 的能力：

1. 图层的数据源是栅格（raster）数据（如 GeoTIFF 或者 Erdas Imagine 格式）；
2. 必须设置图层名称，图层类型设置为 RASTER；
3. 与 WMS、WFS 一样，必须设置图层投影，可以使用设置在地图级别；
4. 必须设置 Web 元数据或层元数据 `wcs_enable_request`；

以上就是 Mapfile 需要满足的要求，满足这些条件，WCS 就可以发布成功。当然，WCS 的规范中还规定了很多参数，可以在 Mapfile 中定义。

下面通过一个例子来实现针对特定图层发布 WCS。依然只定义必须的参数，更多的参数可以在需要的时候再使用。

本节使用的 Mapfile 为 <http://webgis.cn/mfw8.map>。在 METADATA 部分定义如下：

```

15 METADATA
16   "wcs_onlineresource" "http://webgis.cn/cgi-bin/mapserv?map=/owg/mfw8.
17   ↪map"
18   "wcs_keywordlist" "wcs,test"
19   "wcs_enable_request" "*"
20 END

```

配置文件中的 EXTENT 参数可以省略，如果省略，MapServer 会自动计算出其范围；反之，指定这个参数，那么就会省去每次计算的时间。

## 8.2.2 GetCapabilities 操作

GetCapabilities 操作用于得到 WCS 服务器所支持的操作和服务（能力）的 XML 文档。一个典型的 WMS 服务的 GetCapabilities 请求使用地址 <http://webgis.cn/832>，URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfw8.map
2 SERVICE=WCS
3 REQUEST=GetCapabilities

```

由于许多 WCS 支持多个版本，事实上当前许多服务同时支持 WCS 1.0.0 与 1.1.x，甚至是 2.0.x。因此如果在请求地址中没有包含版本参数，那么得到的是可用的最新版本的信息。

通过查看此信息，便可得到服务支持的所有版本信息。例如上述地址返回的响应中包含如下描述所支持的版本信息：

查看输出的 XML 格式的结果，可以看到支持的 WCS 的版本有：2.0.1、1.1.1、1.0.0：

```

13 <ows:ServiceTypeVersion>2.0.1</ows:ServiceTypeVersion>
14 <ows:ServiceTypeVersion>1.1.1</ows:ServiceTypeVersion>
15 <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>

```

在返回的 GetCapabilites 文档中，OperationsMetadata 部分包含了服务支持的操作。GetCapabilites 文档中 Contents 部分包含了各个覆盖的 ID 以及其他一些信息，例如

所包含的波段数。下面是返回的 GetCapabilites 文档 Contents 部分中一个覆盖的信息：

```

136 <wcs:Contents>
137   <wcs:CoverageSummary>
138     <wcs:CoverageId>world-img</wcs:CoverageId>
139     <wcs:CoverageSubtype>RectifiedGridCoverage</wcs:CoverageSubtype>
140   </wcs:CoverageSummary>
141 </wcs:Contents>
```

得到覆盖的 ID 后，便可执行 DescribeCoverage 操作了。要注意的是，上面的 GetCapabilites 文档是使用 2.0.1 版本请求所返回的响应，覆盖 ID 使用的参数名称是 CoverageId。

而对于 1.1.1 的版本，使用的是 Identifier。本书中仅针对 WCS 2.0.1 进行说明。在构造 DescribeCoverage 操作与 GetCoverage 操作时，对于不同版本需要使用其对应的参数。

### 8.2.3 DescribeCoverage 请求

DescribeCoverage 操作允许客户端请求某个 WCS 服务的一个或多个栅格图层的全部描述信息。服务端会返回描述所请求的覆盖图层详细信息的 XML 文档。

DescribeCoverage 操作区分不同版本，主要请求参数（2.0.1 版本）如表 8.1 所示。

表 8.1: DescribeCoverage 操作的主要请求参数（2.0.1 版本）

请求参数	是否必需	描述
service=WCS	是	服务类型
request=DescribeCoverage	是	请求名称
version	是	请求的 WCS 服务的版本
COVERAGEID	是	指定所要请求的图层 COVERAGEID=id1, id2

通过 DescribeCoverage 接口来查看 Mapfile 中定义的名为 world-img 的图层的信息。请求地址为 <http://webgis.cn/239>，URL 解析如下：

```

>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfw8.map
2 SERVICE=WCS
3 VERSION=2.0.1
4 REQUEST=DescribeCoverage
5 COVERAGEID=world-img
```

返回的结果中列出了与图层相关的信息，如在特定空间参考下的地理空间范围：

```

5 <gml:Envelope srsName="http://www.opengis.net/def/crs/EPSG/0/4326"_
6   ↪axisLabels="y x" uomLabels="m m" srsDimension="2">
7     <gml:lowerCorner>-89.97802734375 -180.02197265625</gml:lowerCorner>
8     <gml:upperCorner>90.02197265625 179.97802734375</gml:upperCorner>
9   </gml:Envelope>

```

不同版本之间的参数有较大差别，返回的结果也不同，可以看一下。版本 1.0.0 的 URL 为 <http://webgis.cn/732>，版本 1.1.1 的 URL 为 <http://webgis.cn/638>。可以了解一下差别，具体就不再说明了。

#### 8.2.4 使用 OWSLib 访问 WCS

下面还是来看一下使用 OWSLib 访问的情况。

首先还是连接 WCS，定义为 `wcs_svr` 变量：

```

>>> from owslib.wcs import WebCoverageService
>>> wcs_svr = WebCoverageService('http://webgis.cn/cgi-bin/mapserv?map=/
↪owg/mfw8.map')

```

然后可以查看 `wcs_svr` 的一些属性与方法：

```

>>> wcs_svr.operations
[<owslib.ows.OperationsMetadata GetCapabilities at 0x7f58684ca6d8>,
 <owslib.ows.OperationsMetadata DescribeCoverage at 0x7f58684ca400>,
 <owslib.ows.OperationsMetadata GetCoverage at 0x7f58684cac50>]
>>> wcs_svr.version
'2.0.1'
>>> wcs_svr.items()
[('world-img', <owslib.coverage.wcs201.ContentMetadata object at
↪0x7f58684caa58>)]

```

目前没有 `wcs_svr` 对象与 `GetCapabilities` 对应的直接方法。可以通过 `getOperationByName('GetCapabilities')` 方法获取有限的支持。

```

>>> cap_method = wcs_svr.getOperationByName('GetCapabilities')

```

通过 `wcs_svr.items()` 方法列出 WCS 提供的名称后，就可以通过 `getDescribeCoverage()` 方法获得更多的信息。

```

>>> desc_cov = wcs_svr.getDescribeCoverage('world-img')
>>> type(desc_cov)
<class 'lxml.etree._Element'>

```

返回的对象为 XML 对象，可以通过 XML 的一些属性、方法来操作。

```
>>> desc_cov.prefix
'wcs'
>>> desc_cov.keys()
['{http://www.w3.org/2001/XMLSchema-instance}schemaLocation']
```

对于 WCS，在 Python 使用 OWSLib 来调用可能更加常用一些。这种方式可以更方便地用于地理信息相关的模型计算与数据处理。

### 8.3 使用 GetCoverage 接口获取数据

通过 GetCapabilities 和 DescribeCoverage 这两个操作，可以了解服务端允许哪些请求以及哪些数据是可以获取的，然后进行 GetCoverage 操作。该操作最终返回指定地理范围内指定域值内的某栅格数据。

使用 GetCoverage 有两方面要注意。一是不同版本之间请求的参数有较大差异，在构建请求 URL 前一定要弄清版本与所需要的参数；二是 WCS 的请求参数是由相关标准约束的，并不是简单的约定。

#### 8.3.1 GetCoverage 使用方法

在构造 GetCoverage 操作时，除了包含 service、request、version 与 identifiers 参数值之外，还需要有数据范围的参数。在 WCS 2.0.1 中，这个参数是通过 SUBSET 定义。format 参数用来声明输出的格式，这个参数是可选的。版本 2.0.1 的请求参数相对旧版本要简单一些。

GetCoverage 返回数据，也就是栅格数据。构造 URL 请求没法指定文件的名称与后缀，可以将文件下载后保存，并修改为相应格式的文件，如 GeoTiff 文件可添加 .tif 文件后缀。保存的文件具有地理空间信息。

与 1.1.0 版本不同，2.0.1 不再使用 boundingbox 参数，而是利用 SUBSET 参数分别传递水平与垂直方向的范围。数据范围的参数需要分别设置，不能写到一起。例如 SUBSET=x(70, 120),y(25, 47) 为错误写法，下面的链接表示请求世界地图数据的中国地区：

示例 <http://webgis.cn/237>，这个 GetCoverage 操作请求地址解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfw8.map
2 SERVICE=WCS
3 VERSION=2.0.1
4 REQUEST=GetCoverage
5 COVERAGEID=world-img
```

(下页继续)

(续上页)

```
6 SUBSET=x(71,132)
7 SUBSET=y(20,47)
```

服务器端接到请求后，根据客户端发送请求中的参数，以输出设置中 `format` 指定的格式返回符合条件的数据。这里没声明 `format` 参数，WCS 会根据 `Mapfile` 中的定义来返回可用的格式。返回的是 `GeoTiff` 格式的数据，直接返回数据会比较大，这个文件接近 10M。所以在前面的例子中选择了较小的范围。

### 8.3.2 WCS 输出格式

在 [节 3.6](#)，提到过 `MapServer` 不同格式的输出差异。就地图制图而言，这些输出格式的不同可能影响不大，但是对于 `WCS`，数据格式的不同影像着是否能够正确的返回源文件的数据。

`WCS` 输出格式是在 `Mapfile` 中配置的。当前输出格式列表如下：

图像：

- `JPEG - ( format=jpeg )`
- `GIF - ( format=gif )`
- `PNG - ( format/png )`
- `Tiff - ( format=tif )`
- `BMP - ( format=bmp )`

`WCS` 同样可以以图像格式返回，方便进行查看。

具有地理参考的格式：

- `GeoTiff - ( format=geotiff )`
- `GTopo30 - ( format=gtopo30 )`
- `ArcGrid - ( format=ArcGrid )`
- `GZipped ArcGrid - ( format=ArcGrid-GZIP )`

注意，在 `ArcGrid` 的情况下，`GetCoverage` 请求必须确保 `x` 和 `y` 分辨率相等，否则将抛出异常（`ArcGrid` 被设计为方形单元格）。

可以以 `Jpeg` 格式返回以方便查看，请求 <http://webgis.cn/327>，这个 `GetCoverage` 操作请求地址解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('[\?&]',url))]
```

(下页继续)

(续上页)

```
2 SERVICE=WCS
3 VERSION=2.0.1
4 REQUEST=GetCoverage
5 COVERAGEID=world-img
6 format=image/jpeg
7 SUBSET=x(71,132)
8 SUBSET=y(20,47)
```



图 8.1：使用 GetCoverage 得到的地图结果

这个文件可以另存一下，文件大小要小得多。如果使用 `format=image/png`，文件会大一些，可以访问 <http://webgis.cn/628> 看一下。

## 8.4 使用 GetCoverage 接口获取数据的更多参数与选项

这一节来看下获取数据的其他几种参考。GetCoverage 返回覆盖数据的子集，一般以栅格数据的方式返回。栅格数据有各种信息，包括地图投影、数据范围、波段组合、空间分辨率等，这些都可以在 GetCoverage 请求中声明。

为了方便查看，这一节返回数据的格式皆指定为 `format=image/png`。

### 8.4.1 空间投影

在获取时可以按指定的投影返回数据。如果要输出与源不一样的投影，需要配置 Mapfile 以支持。配置方法为在图层的 META 对象中定义 `wcs_srs`。如果要同时支持 WMS，可以配置 `ows_srs`。

本节使用的 Mapfile 为 <http://webgis.cn/mfs3.map>，声明的元数据信息如下：

```

25 METADATA
26   "ows_srs" "EPSG:4326 EPSG:3857 EPSG:27700 EPSG:4258"
27   "wcs_keywordlist" "land, shallow, topo, world"
28 END

```

请求 `GetCoverage` 时传递的参数包括 `SUBSETTINGCRS=crs`，`OUTPUTCRS=crs`。`SUBSETTINGCRS` 定义了所有子图像（包括输出图像）的空间参考，`OUTPUTCRS` 定义了输出图像的空间参考，下面的链接表示请求过程中的所有子图像使用空间参考为 `EPSG:4326`（注意各字符之间不能有空格）。

<http://webgis.cn/135>

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfs3.map
2 SERVICE=WCS
3 VERSION=2.0.1
4 REQUEST=GetCoverage
5 COVERAGEID=world-img
6 format=image/png
7 SUBSET=x(71,132)
8 SUBSET=y(20,47)
9 SUBSETTINGCRS=EPSG:4326

```

链接 <http://webgis.cn/761> 稍有不同，表示输出图像的空间参考为 `EPSG:3857`。

```

>>> [print(i,p) for i,p in enumerate(re.split('[?&]',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfs3.map
2 SERVICE=WCS
3 VERSION=2.0.1
4 REQUEST=GetCoverage
5 COVERAGEID=world-img
6 format=image/png
7 SUBSET=x(71,132)
8 SUBSET=y(20,47)
9 OUTPUTCRS=EPSG:3857

```

注意，在 URL 中声明投影时，除了用 EPSG 代码，也可以使用 OpenGIS 文档来定义，如：

```
OUTPUTCRS=http://www.opengis.net/def/crs/EPNG/0/3857
```

## 8.4.2 设置图像的返回尺寸

有时需要重新指定返回数据的范围。WCS2.0 分为核心标准与扩展，修改分辨率需要使用尺度扩展及控制交互操作的互操作扩展。

请求 GetCoverage 时传递的参数 SIZE=axis(value)、  
SCALESIZE=axis(size) [, axis(size)] 、 SCALEAXES=axis(value) [, axis(value)] 或 SCALEEXTENT=axis(min:max) [, axis(min:max)] 。注意这里“轴”的定义，不一定是 x 轴与 y 轴，具体由数据定义。

链接 <http://webgis.cn/816> 表示请求数据以 500\*300 的尺寸返回：

URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfs3.map
2 SERVICE=WCS
3 VERSION=2.0.1
4 REQUEST=GetCoverage
5 COVERAGEID=world-img
6 format=image/png
7 SCALESIZE=x(500),y(300)
```

## 8.4.3 图像波段组合

栅格数据一般都由多个波段构成，在请求数据是可以指定波段。传递参数 RANGESUBSET=band1 [, band2 [, …]] ，链接 <http://webgis.cn/293> 表示请求数据以 1、3、2 波段合成返回：

URL 解析如下：

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfs3.map
2 SERVICE=WCS
3 VERSION=2.0.1
4 REQUEST=GetCoverage
5 COVERAGEID=world-img
```

(下页继续)

(续上页)

```
6 format=image/png
7 SCALESIZE=x(500),y(300)
8 SUBSET=x(71,132)
9 SUBSET=y(20,47)
10 RANGESUBSET=1,3,2
```

#### 8.4.4 空间分辨率（像元大小）

传递参数 RESOLUTION=axis (value) , 设置各坐标轴方向上的空间分辨率, 与 SUBSET 类似, 空间分辨率的 X/Y 轴需要分别设置, 这里需要特别注意, 由于数据的空间范围是固定的, 一旦确定了空间分辨率, 则图像的尺寸也随之确定, 反之亦然, 因此 SUBSET , SIZE , SCALESIZE , RESOLUTION 等一些列参数不能同时使用。

链接 <http://webgis.cn/186> 表示的是请求数据以  $0.5 \times 0.5$  度 (这里的单位与投影有关系) 的栅格大小返回:

URL 解析如下:

```
>>> [print(i,p) for i,p in enumerate(re.split('(?:&)',url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfs3.map
2 SERVICE=WCS
3 VERSION=2.0.1
4 REQUEST=GetCoverage
5 COVERAGEID=world-img
6 format=image/png
7 SUBSET=x(71,132)
8 SUBSET=y(20,47)
9 RANGESUBSET=1,3,2
10 RESOLUTION=x(.5)
11 RESOLUTION=y(.5)
```

# 第9章 地图瓦片技术与应用：使用 MapProxy

**WFS**、**WCS** 面向的是相对比较专业的用户，他们所关心的是数据的详实程度、可信程度，访问速度并不是主要关心的指标。**WMS** 面向的用户则要广泛的多，很多都是公众用户，对于这些用户，网页的响应速度是评价用户交互好坏与否的重要一环。

如果要提高 Web 地图的访问速度，使用地图瓦片是非常有效的方法。**Map Tiles**，中译为地图瓦片、地图切片。在本书中认为“切片”的说法注重操作的过程，而“瓦片”更有状态的意义，所以采用“地图瓦片”这一说法。地图瓦片就是在多个比例尺下配置地图，然后提前把每个比例尺下的地图绘制为小块图片，保存在服务器上名为缓存的目录中。这样客户端在访问地图时，可以直接获取需要的小块图片拼接成整幅地图，而不是由服务器动态创建出一幅图片再送到客户端，从而极大提高了访问速度。

瓦片地图是相对较小的矩形（很多情况下是正方形）数据块（栅格化地图图像或原始矢量坐标），这些数据由服务器预先生成并存储在缓存目录中。当 Web 用户浏览地图时，服务器只需提供这些图块，而不是动态生成地图。

## 9.1 地图瓦片的基本概念

最初的 Web 地图，无论在地图中包含了多少个图层，也无论有多少访问请求，通常都是由服务器动态绘制。这也就是前面介绍的使用 WMS 的方式。人们希望在 WebGIS 中也能随意切换图层的可见性、重新排序图层、动态更改图层符号，以及其他所有桌面 GIS 应用程序能做的一切。但是这类地图中符号、标注与注记的选择非常有限而且难以应用。事实上，为了避免妨碍性能，WebGIS 出现后的很长一段时间内，制图者不得不使用最少的图层与简单符号来构造地图。许多情况下，在开发 WebGIS 应用时，甚至不需要专业制图人员的参与，而是由服务器管理员通过软件工具定义图层顺序和符号大小等。这种情况在开源 Web 服务（如 MapServer）与商业 Web 服务（如 ESRI 的 ArcGIS Server）中都存在。

### 9.1.1 瓦片地图的技术原理

这些早期动态绘制的 Web 地图在响应速度和并发性方面遇到了挑战。

MapServer 或其他地图服务器提供 Web 地图需要两个过程，首先是由 MapServer 根据 Mapfile 的设置将数据渲染成图片，然后再通过 Web 的方式提供。在绘制地图时，如果每次都这么请求，每次 MapServer 都要进行渲染过程，那么效率就太低了；而且服务器一次只能容纳有限数量的制图请求，并发请求太多会导致响应变慢甚至崩溃。于是就想到将请求的地图渲染成图片，存储在服务器，每次访问的时候直接越过 MapServer 直接读取现成的图片，这样就省去了渲染过程，减小服务器的开销。

但是这样存在一个问题，由于显示器大小有限，以及有户感兴趣的区域不一样，在用户只想浏览一小块地图的情况下，将整幅地图加载进来是完全没有必要的。

地图将渲染成的图片按照一定的行列规则分成多个小块存储在硬盘上，在调用的时候则根据索引号进行拼接，构成整个地图。这样做可以只加载用户可见的区域的瓦片，其余的不加载，这样也加快了访问速度。

如 [图 9.1](#) 显示了瓦片地图是如何由一个“金字塔”图像组成的，是一种多分辨率层次模型，从瓦片金字塔的底层到顶层，比例尺越来越小，分辨率越来越低，但表示的地理范围不变。这些图像覆盖了地图在不同尺度上的范围。瓦片贴图通常带有一个级别、行和列编号方案，可以跨缓存共享，以确保在覆盖两个瓦片集时瓦片边界匹配。通常都带有一个级别、行与列编号方案，以便将来自多个瓦片地图服务的瓦片放置到正确的位置。

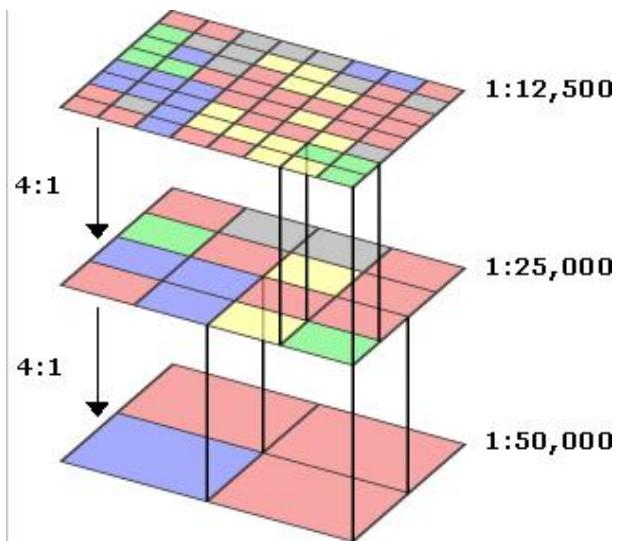


图 9.1：瓦片的 web 地图采用金字塔的形式，在金字塔中，地图以一系列渐进的比例级别绘制，最小（缩小）比例使用较少的瓦片。

### 9.1.2 瓦片地图的特点

制图师们在早期很大程度上由于网络地图的速度而放弃了制图美观的追求，现在他们也意识到了瓦片地图技术的潜力。地图中的层数不再会减慢服务器的速度：预先生成图块之后，不论是简单的地图还是精美的地图响应速度都一样（网络传输速度是有一点差别）。网络地图成为锻炼制图技术并使最有吸引力的地图成为可能的机会。这样就诞生了如今在 Google, Microsoft Bing, OpenStreetMap 和其他流行网站上常见的精美，快速且详细的“Web 2.0”底图。例如，如果使用 Firebug 等开发者工具来检查谷歌地图的话，可以看到其底图与专题图层都是以瓦片方式提供的。

进一步地，在 Google 地图的引领下，许多网站开始使用一种被称为 Asynchronous JavaScript 和 XML (AJAX) 的创新技术提供“预先准备好”的瓦片地图图像，这种技术消除了早期 Web 地图中任何导航操作后出现的普遍且烦人的闪烁现象。现在，浏览地图基本感受不到延迟。

瓦片地图具有如下特点：

1. 速度快，预先配色，显示效果佳，满足高并发访问，适合做底图。
2. 瓦片地图缓存非常高效，如果曾经查看过某一区域的瓦片，当再次查看该区域的瓦片时，浏览器可能使用之前缓存的相同的瓦片，而不用重新下载一次。
3. 瓦片地图可以渐进加载。中心区域的瓦片可能优先于边缘区域的瓦片加载，可以移动或定位某一点，即使当前地图边缘区域还未加载。
4. 简单易用。所以很容易在服务器、网络、桌面或移动设备上实现技术集成。
5. 需要额外占用磁盘空间，需要预先生成瓦片，无法自定义地图。

### 9.1.3 瓦片地图的应用

瓦片地图是唯一一种能够合理地为成千上万的用户同时提供复杂网络地图的模型。但是，它们限制用户更改图层顺序或符号的能力。人们开始着手解决这个问题，把通用的基础地图图层作为瓦片，然后用专题信息覆盖一个单独的图层。通用的基础图块可以在许多应用程序中重用。如果数据变化不太快或覆盖范围过大，主题层也可以做成瓦片。

如果希望基础底图能够快速查看，或者同时有几十个并发用户，这时应该为底图创建瓦片缓存。如果专题图层中地物要素的空间位置与属性信息不经常变换，那么也可以创建瓦片缓存。另外，瓦片地图避免了传输原始数据，对数据的保密起到了一定的作用。

创建瓦片底图需要大量丰富的数据、高端的地图制作软件和制图技能，以及潜在的大量时间和磁盘空间。正是由于存在这些挑战，通用的 Web 混搭常常使用他人创建的地图瓦片。选择使用公用瓦片地图，需要检查是否有“问题地图”情况存在。

如果想没有任何限制的自由与免费使用，那么 OpenStreetMap 或其衍生产品是一个选择。但可惜的是其中国国界及一些地区的标注不规范或者存在问题。而对于谷歌、微软或 ESRI 的瓦片地图，则根据地图的性质（商业或不以盈利为目的），以及多少人使用应用程序，确定是免费

还是付费使用。而其他一些公司，例如 CloudMade 和 MapBox，则以 OpenStreetMap 数据为基础提供他们自己版本的瓦片。

如果决定创建自己的基础底图，那么有经验的制图人员是必不可少的。另外还有一个关键问题，与桌面或传统地图制图不同，绝大多数 WEB 地图都是多比例尺地图。每个比例尺下，地图都应该有合适的符号、颜色以及相应的详细程度。仅仅为地图不同的比例尺创建注记就是一项令人望而生畏的任务。此外，如果在底图中包含卫星或航空影像，那么制图人员还需要另外再单独制作一组瓦片，因为需要不同的颜色与符号。

#### 9.1.4 瓦片地图使用的注意事项

使用瓦片地图，必须了解瓦片代表着瓦片创建时该时刻点的地图快照。当后端数据发生变化时，这些瓦片不会自动更新。为了更新地图，瓦片地图服务提供者必须定期创建新的瓦片。对于那些大规模的瓦片缓存，有时管理员只针对变换领域更新瓦片，而不是重建整个地图范围的所有瓦片。而这又需要记录哪些地方进行过编辑，或比较几个版本的数据。

另外，如果要混搭使用 OpenStreetMap、Google、Microsoft 或 Esri 中的瓦片地图（甚至尝试与它们相似），则必须将 GIS 数据投影成一个修改过的球形墨卡托投影，该投影是专为将世界拟合到一组方形瓦片上而创建的。也是目前 WebGIS 中最常用的地图投影 EPSG:3857。

在 EPSG:3857 中显示地图时，不要在此投影中执行测量，即使在中纬度，结果也会大大偏斜。在执行任何测量之前，最好确保将几何图形投影到更局部的坐标系中。大多数流行地图绘制工具（例如 Google Earth 和 ArcGIS.com 地图查看器）在使用它们的测量工具时都会在后台执行投影变换的操作；但是如果要构建自己的解决方案，则不会自动发生。

除了保证使用相同的投影之外，还需要确保比例尺一致。瓦片地图的比例尺序列与通常见到的地形图比例尺序列很不一样。我国地形图比例尺一般有 1:10000、1:50000 等。瓦片地图的比例尺是计算出来的。以 EPSG: 3857 投影为例，最小比例尺是将整个世界放置在  $2 \times 2$  的网格中，每个网格大小是 256 \* 256 像素，因此最小比例尺为 1:295829355.45；再放大一个级别，那么比例尺在原基础上乘以 2，结果为 1:147914677.73；以此类推。由于这些比例尺很不好记忆，也很不好用于交流，并且互联网地图用户也没必要关心这么精确的数字，因此技术人员（WebGIS 开发人员）通常用瓦片地图的级别来表示，如“14 级”，“15 级”，“20 级”等。对于消费用户，利益于瓦片技术，可以也鼠标或手势来快速定位感兴趣的区域。

#### 9.1.5 矢量瓦片：下一代瓦片地图

尽管在中讨论的栅格化瓦片集能够以相对快速的格式提供漂亮的地图，但它们更新起来可能会很麻烦，并且需要大量的计算资源。为了应对这些挑战，一种称为“矢量瓦片”的数据存储格式在过去几年中越来越流行。Mapbox 公司领导了矢量图块的开发工作，并根据知识共享许可证共享了 Vector tile specification，最新版本是 2016 年发布的 2.1 版本。矢量瓦片被设计成磁盘上的小瓦片，并采用了许多优化方法来减少存储地理数据和属性所需的字符数量。

矢量瓦片存储矢量数据块，而不是存储地图图像。矢量瓦片背后的思想是，将数据样式与数据坐标和属性分开更有效。客户机可以使用一组预定义的样式规则来绘制由服务器发送的原始向量坐标和属性数据的瓦片。这允许动态地重新建模数据，这是栅格化分片的另一个严重限制。想想看：如果想用栅格化瓦片改变用来绘制公园的绿色阴影，必须重建每个包含公园的瓦片。如果想对矢量瓦片执行相同的操作，只需在一个地方更新样式说明，瓦片本身就保持不变。其他显示操作（如旋转地图）也变得更容易用矢量瓦片实现。

## 9.2 地图瓦片的相关标准

标准的形成是一个过程。地图瓦片技术先于标准出现，形成不同的实践，这样关于地图瓦片，就出现了 WMS-C、TMS、WMTS 不同的技术与标准。

在地图瓦片的原理与技术出现之后，OSGeo 想出了两种方法来实现这个问题。一个是扩展 WMS 协议与额外的信息，以便符合客户端可以发送正确排队的请求。这被称为 WMS-C。另一种是 TMS，一种全新的协议，专为瓦片而设计。特别地，它使用整数索引用于瓦片，而不是要求客户端请求与特定网格对齐的边界框。

然后，OGC 又提出了原来的 WMS 规范，决定制作一个瓦片协议，结果是 WMTS。这就像 TMS 是在整数索引的瓦片，但它也支持一些相比 WMS 而言 TMS 缺乏的接口，如 Dimensions 和 GetFeature，而且它也比 TMS 复杂得多。

### 9.2.1 创建与提供瓦片地图服务的策略

预渲染的瓦片可以使用三种协议中的任何一种来提供，并且存在支持所有三种协议的瓦片服务器，一些在常规 WMS 服务器之前用作缓存代理（如 GeoServer 和 GeoWebCache，或 MapServer 与 MapCache）。

地图瓦片一般都采用简单的文件夹结构，以便提供服务。然而，由于瓦片数量非常多，因此它们的管理变得非常复杂。当前互联网的瓦片地图基本有如下两种方式组织：

第一种将瓦片图像以文件夹的结构放在服务器上，用户直接请求文件。

在这种方法中，只需要将单独的瓦片图像组织在代表比例级别、行和列的文件夹结构中。很多地图 API 通过访问包含代表级别、行和列结构的 URL 来访问瓦片。例如，当使用 Leaflet API 访问地图瓦片时，必须提供格式为 `http://{s}.somedomain.com/path/{z}/{x}/{y}.png` 的网址，其中 z 是缩放级别，x 和 y 分别是列和行。

第二种将瓦片以 Web 服务的方式提供访问。

在这种方法中，虽然 Web 服务仍然需要用户提供访问瓦片的具体缩放级别、行与列，但是其背后文件的组织形式是看不见的。该方法比直接使用文件夹的方式相对稍微复杂一些，因此同时也会带来延时。OGC 的 Web 地图瓦片服务（Web Map Tiled Service，WMTS）规范就是按照这种方式来提供瓦片服务的。

### 9.2.2 OSGeo WMS-C 标准

WMS-C 全称是 Web Mapping Service - Cached，2006 年在 FOSS4G 会议上提出讨论，目的在于提供一种预先缓存数据的方法，以提升地图请求的速度，自始至终该标准都没有写入 OGC 之中。WMS-C 通过 `bbox` 和 `resolutions` 决定请求的地图层级，为了更加直观的请求地图瓦片，一些软件做了一些改进，例如 WorldWind 在请求中使用 `level/x/y` 三个参数，直观明了。典型的基于 WMS-C 的实现是 TileCache。这个项目早已停止开发，不建议使用。

### 9.2.3 OSGeo TMS 标准

TMS (tile map Servcie) 定义了一些操作，这些操作允许用户访问瓦片地图，实现了一种瓦片地图服务。TMS 的算法很简单，就是把投影后的世界地图按照层级进行四叉树 (待验证) 切割，切割后的瓦片数量随层级呈金字塔型，数量和层级关系如下 表 9.1 所示：

表 9.1: TMS 的瓦片表达

级别	分片数目	瓦片总数目
0	1 tile covers whole world	1 tile
1	$2 \times 2$ tiles	4 tiles
2	$4 \times 4$ tiles	16 tiles
...	... tiles	... tiles
n	$2^n \times 2^n$ tiles	$2^{(2n)}$ tiles
...	... tiles	... tiles
12	4096 x 4096 tiles	16,777,216
16	Maximum zoom for OpenCycleMap (mostly)	$2^{32} = 4,294,967,296$ tiles
17	Maximum zoom for Osmarender layer	17,179,869,184 tiles
18	Maximum zoom for Mapnik layer	68,719,476,736 tiles

### 9.2.4 OGC WMTS 标准

WEB 地图瓦片服务 (WMTS，OpenGIS Web Map Title Service) 提供了一种采用预定义瓦片方法发布数字地图服务的标准化解决方案。WMTS 弥补了 WMS 不能提供瓦片地图的不足。WMS 针对提供可定制地图的服务，是一个动态数据或用户定制地图（需结合 SLD 标准）的理想解决办法。WMTS 牺牲了提供定制地图的灵活性，代之以通过提供静态数据（基础地图）来增强伸缩性，这些静态数据的范围框和比例尺被限定在各个瓦片内。这些固定的瓦片集使得对 WMTS 服务的实现可以使用一个仅简单返回已有文件的 Web 服务器即可，同时使得可以利用一些标准的诸如分布式缓存的网络机制实现伸缩性。

WMTS (OpenGIS® Web Map Tile Service) 当前版本是 1.0.0。可能是 OGC 首个支持 RESTful 访问的服务标准。WMTS 接口支持的三类资源如下：

1. 一个服务元数据（ServiceMetadata）资源（面向过程架构风格下对 GetCapabilities 操作的响应）（服务器方必须实现）。ServiceMetadata 资源描述指定服务器实现的能力和包含的信息。在面向过程的架构风格中该操作也支持客户端与服务器间的标准版本协商。
2. 瓦片资源（对面向过程架构风格下 GetTile 操作的响应）（服务器方必须实现）。瓦片资源表示一个图层的地图表达结果的一小块。
3. 要素信息（FeatureInfo）资源（对面向过程架构风格下 GetFeatureInfo 操作的响应）（服务器方可选择实现）。该资源提供了瓦片地图中某一特定像素位置处地物要素的信息，与 WMS 中 GetFeatureInfo 操作的行为相似，以文本形式通过提供比如专题属性名称及其取值的方式返回相关信息。

在一个 WMTS 服务中包括以下 3 个操作：

1. GetCapabilities（获取服务的元信息）
2. GetTile（获取瓦片）
3. GetFeatureInfo（可选，获取点选的要素信息）

可以看到这些操作和 WMS 的操作非常的类似。

### 9.2.5 不同标准的比较

不同瓦片标准之间的差异主要在于协议、瓦片组织方式。

瓦片技术开始出现的时候 OSGeo 更快地推出了解决方案，TMS 较多地得到了实践应用，但是 WMTS 在以后的发展中应该会得到更多的支持。WMS-C 和 TMS 是成熟的，但并非是 OGC 规范。WMS-C 功能有点弱，而 TMS 缺乏一些有用的功能。WMTS 拥有 OGC 的支持，并结合了 TMS 的瓦片特定设计和 WMS 的额外功能，WMTS 理论模型更好，最重要是对互操作性的支持，也就是 OpenGIS 的相关协议。

与 WMS 相比，WMS 非常灵活，可以提供定制地图服务。WMTS 相比 WMS，牺牲掉灵活性来换取性能，返回的都是基础地图，所有静态数据的范围框和比例尺统统被限定在各个瓦片中。然后这些瓦片集使得对 WMTS 的实现只需要一个仅仅返回简单已有文件的 WEB 服务器即可。

## 9.3 MapProxy 安装与使用

前面两节对地图切片的原理与标准进行了一些介绍，在本章的后面部分，会使用 MapProxy 作为实践工具介绍如何来使用。

这一节先说明安装 MapProxy 的方法，以及初步的用法。

### 9.3.1 MapProxy 介绍

在 Google Maps 发布后的一两年内，商业 GIS 软件开始提供构建地图瓦片的能力。对许多人来说，可以选择使用 ArcGIS Server，因为可以使用 ArcMap 中成熟的地图创作工具来创作地图；然而，成本可能会是一项限制。地图切片的技术方案有多种，关于地图切片的开源工具有许多，免费开源的 Mapnik 库也可以构建瓦片，但直到最近几年，TileMill 这样的项目才在 Mapnik 上封装了一个用户友好的 GUI。到目前，地图瓦片的软件工具已经很成熟，是当前各种 FOSS 软件中的一个基本工具。MapServer 有 MapCache 模块，GeoServer 有 GeoWebCache 模块来创建地图切片；TileMill 和 CartoCSS 标记语言能以更灵活的方式创建地图切片；还有 GeoWebCach，TileCache，TileStache，Tile-Server 及 MapProxy 等常用的小工具。其他还有一些工具了解的人少一些。

根据使用过 TileCache 与 GeoWebCache 的用户反馈，更推荐使用 MapProxy。本书介绍的就是 MapProxy，一个使用 Python 开发的工具。MapProxy 可以说是目前最容易使用的地图切片工具。具有如下特点：

1. 工具包很轻量，但是表现很出色；
2. 理解和配置相当简单；
3. 如果将缓存目录映射为 Web 目录，则可以使用 Nginx 绕过 Mapproxy 直接使用既有的切片。仅在缓存未命中情况下调用 Mapproxy；
4. 项目非常活跃，并且几乎没有 Bugs。
5. 对目前的瓦片标准都有支持。

### 9.3.2 安装 MapProxy

MapProxy 的官方网站为 <https://mapproxy.org/>，在网络上可以找到下载的地址。

MapProxy 是完全用 Python 写的，最简单的安装方法是使用 pip 命令自带的安装工具进行安装，命令如下：

```
# pip install mapproxy
```

安装之后可以使用命令进行测试，根据返回的情况来查看安装是否成功。

```
mapproxy-util --version
```

### 9.3.3 配置

MapProxy 是按照项目组织运行的，在开始运行之前，需要使用命令行工具 `mapproxy-util` 建立 MapProxy 项目。这个工具可以运行不同的命令，命令还有不同的选项可以使用。

最简单的情况下，可以使用命令创建实例配置文件，进行测试。命令如下：

```
mapproxy-util create -t base-config ./
```

运行后会生成 4 个文件：`full_example.yaml`，`full_seed_example.yaml`，`mapproxy.yaml`，`seed.yaml`。

这些基本的文件可以看成是模板，要发布自己的数据，可以针对配置文件进行修改。

然后根据 `mapproxy.yaml` 文件配置，下面介绍一下配置文件的组成部分。本节在测试过程中，只使用了 `mapproxy.yaml` 的配置。

- `mapproxy.yaml`：主要配置文件，配置服务的所有部分，比如那些服务需要启动，数据来自哪里，那些需要缓存

文件中包含的部分在下一节说明。

自动生成的示例代码 `mapproxy.yaml` 如下所示，在测试的时候可以使用：

```
services:
  demo:
  tms:
    use_grid_names: true
    # origin for /tiles service
    origin: 'nw'
  kml:
    use_grid_names: true
  wmts:
  wms:
    md:
      title: MapProxy WMS Proxy
      abstract: This is a minimal MapProxy example.
layers:
  - name: osm
    title: Omniscale OSM WMS - osm.omniscale.net
    sources: [osm_cache]
caches:
  osm_cache:
    grids: [webmercator]
    sources: [osm_wms]
sources:
```

(下页继续)

(续上页)

```

osm_wms:
  type: wms
  req:
    url: https://maps.omniscale.net/v2/demo/style.default/service?
    layers: osm
grids:
  webmercator:
    base: GLOBAL_WEBMERCATOR
globals:

```

### 9.3.4 启动

接下来使用命令来启动 MapProxy，测试是否成功。同样使用 mapproxy-util 工具，但是使用的命令并不一样。

```
mapproxy-util serve-develop ./mapproxy.yaml
```

如上面运行成功，说明 MapProxy 已经可以使用。但是可能还会有其他的问题会导致切片无法生成，需要进行检查。

要访问启动后的 MapProxy，在浏览器输入 <http://localhost:8080> 进行访问。

如果不使用参数，运行后是只能在本地访问的。设置其他服务器可以访问：

```
mapproxy-util serve-develop ./mapproxy.yaml -b 0.0.0.0:8080
```

### 9.3.5 配置格式

这一节继续介绍 MapProxy 配置，来熟悉 MapProxy 的语法与规则，以便为后面发布自己的数据奠定基础。

MapProxy 的配置使用 YAML 格式。YAML 是 JSON 的超集。这意味着每个有效的 JSON 也是有效的 YAML。MapProxy 不使用 YAML 的高级特性，所以甚至可以直接使用 JSON 格式的文件。YAML 使用更可读和用户友好的语法，鼓励大家使用它。

先来介绍下 YAML 的基本语法，如果熟悉的话可以直接跳到下一节。

YAML 配置由注释、字典、列表、字符串、数字和布尔值组成。# 字符后的所有内容是注释，将被忽略；任何数值，比如 12，-4，0 和 3.1415，会识别为数字；单引号或双引号内的任何字符串；如果字符串在 YAML 语法中没有其他含义，则可以省略引号。

例如：

```
'foo'  
foo  
'43' # with quotes, otherwise it would be numeric  
'[string, not a list]'  
A string with spaces and punctuation.
```

布尔值，表达正确或错误值：

```
yes  
true  
True  
no  
false  
False
```

列表是其他有效对象的集合。有两种格式。压缩形式使用方括号：

```
[1, 2, 3]  
[42, string, [another list with a string]]
```

块窗体要求每一个列表项都位于单独的行上，从 - (破折号和空白)：

```
- 1  
- 2  
- 3  
  
- 42  
- string  
- [another list]
```

字典将键映射到值。值本身可以是任何有效的对象。有两种格式。压缩形式使用大括号：

```
{foo: 3, bar: baz}
```

块形式需要单独行上的每个键值对：

```
foo: 3  
bar: baz
```

还可以嵌套字典。每个嵌套字典都需要缩进一个或多个空格。同一字典的所有键都需要缩进相同数量的空格。

```
baz:  
    ham: 2
```

(下页继续)

(续上页)

```
spam:  
  bam: True  
  inside_baz: 'yepp'
```

### 9.3.6 瓦片地图预生成方法

MapProxy 根据需要创建所有切片，这意味着，只缓存一次请求的瓦片。如果所有的瓦片都在进行时创建，可能会导致访问缓慢，产生不佳的用户体验。幸运的是，MapProxy 附带了一个命令行脚本 `mapproxy-seed`，用于预生成所有需要的瓦片地图。

- `mapproxy-seed` MapProxy 创建所有需要的图片，为了提高请求速度，这个工具可以缓存一个或多个多边形区域。

#### 配置

运行 `mapproxy-seed` 使用自己的配置文件，创建项目后默认文件名为 `seed.yaml`，称为种子文件。

和所有的 MapProxy 配置文件一样，它使用 YAML 语法进行定义。必须有 `seeds` 键。在这里，可以创建多个种子设定任务，定义应该播种的内容。可以指定用于种子设定的缓存列表 `caches`。缓存名称应与 MapProxy 配置中的名称匹配。如果在 MapProxy 配置中为一个缓存指定了多个网格，则可以选择这些缓存进行种子设定。它们还必须符合 MapProxy 配置中的缓存。此外，还可以限制应该播种的级别。如果只想在有限的区域播种，可以使用 `coverages` 选项进行控制。

在下面的示例中，配置 `mapproxy-seed` 为以前创建的缓存 `test_wms_cache` 从 6 级到 16 级。为了显示定义覆盖率的不同可能性，使用一个多边形文件来确定要缓存的区域。

```
seeds:  
  test_cache_seed:  
    caches: [test_wms_cache]  
    levels:  
      from: 6  
      to: 16  
    coverages: [germany]  
  
coverages:  
  germany:  
    polygons: ./GM.txt  
    polygons_srs: EPSG:900913
```

正如在 coverages 截面 polygons 选项指向文本文件。此文本文件包含已知文本 (WKT) 形式的多边形。第三个选项表示 mapproxy-seed 这个 srs WKT 多边形。

## 开始运行种子文件

现在该开始运行种子文件。mapproxy-seed 有几个选择。必须使用选项 -s 定义 seed.yaml 和 -f 对于 MapProxy 配置文件。也可以使用 --dry-run 选择查看 MapProxy 将做什么，而不向源发出任何实际请求。配置错误的种子文件可能需要几天或几周的时间才能发现，因此注意 --dry-run 打印出来的切片编号。

使用下面命令运行 mapproxy-seed：

```
mapproxy-seed -f mapproxy.yaml -s seed.yaml --dry-run
```

如果确定种子文件工作正常，移除 --dry-run 参数，即可真正生成地图瓦片

## 9.4 配置 MapProxy 提供地图服务

这里提供了一个简化案例，来逐步配置，最终实现瓦片地图服务。

### 9.4.1 配置布局

注意，根据 YAML 的文件格式，不能在文件中使用制表符，而只能用空格。这种强调的要求避免了一些无谓的误会。MapProxy 配置是一个字典，使用 YAML 格式，每个键配置 MapProxy 的不同方面。主要的有以下键：

- **globals**：在这里可以定义默认值，全局变量，在其他配置部分使用；
- **services**：这是激活和配置 MapProxy 能够提供的服务的地方，如 WMS 和 TMS；
- **sources**：定义 MapProxy 在哪里可以检索到新的数据；
- **caches**：可以在这里配置内部缓存；
- **layers**：配置 MapProxy 提供的图层，每一个图层都可以包含多个数据源 sources 和缓存 caches。
- **grids**：MapProxy 将所有缓存图像（分片）与网格对齐。在这里可以定义那个网格。

这其中，services 配置了对外提供的服务，服务由 layers 具体实施；layers 的数据源则由 sources 或 caches 提供。根据 YAML 的语法，这些指令的顺序并不重要，可以按照方便的方式进行组织。

sources 这个是 MapProxy 对外的入口，作为数据源输入。其支持 WMS、ArcGIS REST API、Tiles、Mapserver、Mapnik、Debug。不同的数据源有不同的选项。

## 9.4.2 配置示例

先来配置一下最简单的应用，只用到一个图层。

示例见：<http://webgis.cn:6001/>，其对应的配置文件为 <http://webgis.cn/tuto-a.yaml>。这个文件可以打开对照查看，在下面会展示其中的片段。

### 配置服务

一开始需要配置至少一个服务。要启用服务，必须将其名称作为键包含在 services 字典。例如：

```

25 services:
26   demo: null
27   kml:
28     use_grid_names: true
29   tms:
30     origin: nw
31     use_grid_names: true
32   wms:
33     md:
34       abstract: This is a minimal MapProxy example.
35       title: MapProxy WMS Proxy
36   wmts: null

```

每个服务都是一个 YAML 字典，以服务类型为键。字典的值可以是空的，但需要添加冒号，以便配置解析器知道它是字典类型。

服务可能接受更多的配置选项。例如，WMS 服务使用带有元数据的字典。这些数据在功能文档中使用。

下面是一个包含更多信息的示例，但并不是本节示例文件中的内容。

```

services:
  wms:
    md:
      title: MapProxy WMS Proxy
      abstract: This is the fantastic MapProxy.
      online_resource: http://mapproxy.org/
      contact:
        person: Your Name Here
        position: Technical Director
        organization:
        address: Fakestreet 123

```

(下页继续)

(续上页)

```

city: Somewhere
postcode: 12345
country: Germany
phone: +49(0)000-000000-0
fax: +49(0)000-000000-0
email: info@omniscale.de
access_constraints:
  This service is intended for private and
  evaluation use only. The data is licensed
  as Creative Commons Attribution-Share Alike 2.0
  (http://creativecommons.org/licenses/by-sa/2.0/)
fees: 'None'

```

`access_constraints` 演示如何在多行上编写字符串，只需按与第一行相同的方式缩进每一行。记住，YAML 不接受制表符，必须使用空格。

对于本教程，添加了另一个名为 `demo`。这是一个演示服务，列出了所有配置的 WMS 和 TMS 层。可以使用一个简单的 Openlayers 客户机测试每个层。所以配置文件应该如下所示：

```

services:
  demo:
    wms:
      [rest of WMS configuration]

```

## 添加源

接下来需要定义源中的数据。每个源都有一个名称和类型。添加一个 WMS 源：

```

37 sources:
38   maplet_WorldCountry:
39     image:
40       transparent_color: '#ffffffff'
41       transparent_color_tolerance: 0
42     req:
43       layers: lyr_WorldCountry
44       url: http://webgis.cn/813
45     type: wms

```

在这个例子中 `test_wms` 是源的名称，稍后需要此名称来引用它。大多数数据源需要更多的参数——有些是可选的，有些是必需的。类型 `wms` 需要 `req` 描述 WMS 请求的参数。至少需要定义一个 URL 和层名称，但是可以添加更多选项，如 `transparent` 或 `format`。

## 添加层

在定义了一个源之后，可以使用它创建 WMS 图层。

一个层需要一个标题，该标题将在功能文档和源中使用。对于这个层，要使用 test\_wms 数据来源：

```
16 layers:
17 - name: maplet_WorldCountry
18   sources:
19     - maplet_WorldCountry
20   title: maplet_WorldCountry
```

现在将 MapProxy 设置为级联 WMS。这意味着 MapProxy 只将请求重定向到在中定义的 WMS test\_wms 数据源。启动开发服务器

这是第一个配置，现在可以 start MapProxy ::

```
mapproxy-util serve-develop mapproxy.yaml
```

在这里，可以看到自己配置的服务的功能，并观察它的运行情况。

## 添加缓存

为了用 MapProxy 加速源代码，对于这个来源创建对应的缓存。

每个缓存都需要知道在哪里可以获取新数据，以及应该如何缓存。定义的 test\_wms 作为缓存的源。MapProxy 将图像拆分为小图块，这些图块将与网格对齐。它还以不同的分辨率缓存图像，如图像金字塔。可以详细定义这个图像金字塔，但是从默认的 MapProxy 网格定义开始。GLOBAL\_GEOGRAPHIC 定义一个覆盖整个世界的网格。它使用 epsg:4326 作为空间参考系统，并与 Openlayers 使用的默认网格和分辨率对齐。

缓存配置现在应该如下所示：

```
1 caches:
2   maplet_WorldCountry:
3     grids:
4       - webmercator
5     sources:
6       - maplet_WorldCountry
```

## 添加缓存层

现在可以使用定义的缓存作为层的源。当客户机请求层时，MapProxy 会在缓存中查找请求的数据，并且仅当它尚未缓存数据时，才会请求 test\_wms 数据源。

层配置现在应该如下所示：

```
layers:
  - name: test_wms_cache
    title: Cached Test Layer
    sources: [test_wms_cache]
```

可以 [download the configuration](#) .

### 9.4.3 更多图层的配置

要配置使用更多的图层，方法与之相同。需要注意名称不要重复了。实际进行编辑修改会发现尽管 YAML 宣称易于编辑，但是还是容易出错。

示例见：<http://webgis.cn:6002/> 。

文件为 <http://webgis.cn/tuto-b.yaml> 。

## 9.5 MapProxy 实例教程：进阶

这一节进一步来了解 MapProxy 的一些用法。

### 9.5.1 合并多个层

如果有两个 WMS，并且想要提供一个包含来自两个服务器的数据的单层，那么可以将它们组合到一个缓存中。MapProxy 将在磁盘上存储瓦片之前组合图像。应该从下到上定义源，并且除底部之外的所有源都必须是透明的。

合并图层，可以在数据源，也可以在图层。

示例见：<http://webgis.cn:6003/>

下面的代码是将 MapProxy 配置为在一个缓存和一个层中组合两个 WMS 的示例：

```
22 maplet_mulwold:
23   grids:
24     - webmercator
25   sources:
26     - maplet_WorldCountry
```

(下页继续)

(续上页)

```

27   - maplet_WorldRailway
28   - maplet_WorldRiver
29   - maplet_WorldRoad

```

## 9.5.2 覆盖范围

有时，不想在一个层中提供 WMS 的完整数据。使用 MapProxy，就可以定义数据可用或感兴趣的数据所在的区域。MapProxy 提供了三种限制可用数据区域的方法：边界框、多边形和 ogr 数据源。为了简单起见，只讨论边界框。有关其他方法的更多信息，请查看 the [coverages documentation](#)。为了用边界框限制区域，必须在数据源的覆盖选项中定义它。下面的列表将可请求区域限制为德国：

```

sources:
  test_wms:
    type: wms
    req:
      url: http://osm.omniscale.net/proxy/service?
      layers: osm
    coverage:
      bbox: [5.5, 47.4, 15.2, 54.8]
      bbox_srs: 'EPSG:4326'

```

显而易见，覆盖范围边界框的符号与网格选项中的符号类似。元数据块和元数据缓冲区

当有经验与 WMS 在切片客户，应该知道的问题，标签问题。MapProxy 可以通过调用两个方法来帮助解决这些问题。Meta Tiling 和 Meta Buffering。

有一个 chapter on WMS labeling issues 讨论这些选项。

## 9.5.3 定义分辨率

默认情况下，MapProxy 缓存两个图像金字塔的传统功能，默认缓存的分辨率为 20。每层金字塔之间的分辨率是原来的两倍。如果想改变这个，可以通过 [defining your own grid](#)。幸运的是，MapProxy 网格提供了从其他网格继承的能力。让网格继承以前使用的 GLOBAL\_GEODETIC 网格并添加五个固定分辨率。

网格配置应该如下所示：

```

grids:
  res_grid:
    base: GLOBAL_GEODETIC
    res: [1, 0.5, 0.25, 0.125, 0.0625]

```

一目了然，曾经 base 继承 GLOBAL\_GEODETIC 和 res 定义的首选解决方案。分辨率始终以 SRS 为单位，在这种情况下，以每个像素的度数为单位。可以使用 MapProxy scales util 在刻度和分辨率之间转换。

也可以定义一个用于计算分辨率的因子，而不是定义固定分辨率。此因子的默认值为 2，但可以将其设置为所需的每个值。只是改变 res 具有 res\_factor 在后面加上因素。

神奇的价值 res\_factor 是  $\text{SqRT2}$ ，2 的平方根。它将缓存的分辨率数量加倍，因此有 40 个而不是 20 个可用的分辨率。每一秒的分辨率都与两个分辨率的功率相同，因此不仅可以在具有免费 Zomming 的经典 WMS 客户机中使用此层，而且可以在基于平铺的客户机（如 Openlayers）中使用此层，后者只请求这些分辨率。看看 configuration examples for vector data for more information

#### 9.5.4 定义网格

在上一节中，看到了如何扩展网格以提供自定义的解决方案，但有时 GLOBAL\_GEODETIC 网格并不有用，因为它覆盖了整个世界，只需要其中的一部分。那么看看如何 define our own grid

对于这个例子，为德国定义了一个网格。需要一个与德国区域匹配的空间引用系统（SRS）和一个围绕德国的边界框（BBOX），以限制可请求的 AREA。使 bbox 稍微容易一点，把 bbox\_srs 网格配置的参数。所以可以定义 bbox 在 EPSG: 4326。

这个 grids 配置是一个字典，每个网格配置都由其名称标识。将其称之为网格 germany 其配置应如下所示：

```
grids:
  germany:
    srs: 'EPSG:25832'
    bbox: [6, 47.3, 15.1, 55]
    bbox_srs: 'EPSG:4326'
```

必须更换 GLOBAL\_GEODETIC 在缓存配置中，germany 网格。之后，MapProxy 将所有数据缓存到 utm32 中。

MapProxy 请求网格投影中的源。可以配置 the supported SRS for each WMS source 如果 srs 网格与数据源不同。



# 第 10 章 WebGIS 前端类库 Leaflet 的使用

到目前为止，已经能够创建几种不同类型的层，例如动态服务的 web 地图服务（WMS）和静态瓦片地图。也已经以各种方式预览了这些层，但可能会得出这样的结论：使用这些预览机制，它们不是非常有用或很难共享。现在，将学习如何编程一个结合了自己的层的 web 应用程序，为自己的 web 地图创建一个易于共享的“前端”。将学习什么是 Leaflet API，以及如何使用它来达到这个目的。

如果对 web 编程还不太熟悉，建议花点时间查看 HTML 和 JavaScript 的教程。不需要知道所有的内容，但确实需要足够熟悉代码和标记模式，以便在稍后的章节中查看示例时能够解释所看到的大部分内容。花费一些额外的时间和精力，应该能够看到 JavaScript 与已经知道的语言之间的关系。

在本章中，将前面学习的地图服务作为服务提供者，将 Leaflet 作为应用前端，形成现代通用的架构。

Leaflet 是的功能相对简单，与 OpenLayers 相比并没有包含太多功能，但是对于学习与使用大部分情况下足够了。如果需要更复杂的功能，可以选择使用 OpenLayers。

本章主要包含以下内容：

1. 了解常用的 Web 制图 API（专有和自由／开源软件），清楚 Leaflet 的特点；
2. 掌握 Web 开发中 Leaflet 的基本用法，包括如何定义，如果添加弹出窗口等；
3. 使用 Leaflet 调用 WMS、WMTS 等；
4. 使用 Leaflet 调用 WFS，并进行样式定义。

## 10.1 什么是 Web 制图 API？

API（应用程序编程接口）是可用于编写程序的框架。它提供了一组类和函数，可避免编写所有低级代码来执行特定操作。例如，Web Mapping API 通常包括用于地图和图层的类，因此不必编写用于显示交互式地图图像并在其上绘制新图层的所有低级代码。相反，可以只创建一个新的地图对象，创建一个新的图层对象，然后调用诸如 `layer.addTo(map)` 之类的方法。该 API 提取了任务的复杂性，可以轻松地将精力集中在应用程序的制图方面，而不必花时间在底层物流上。

专门为制作 web 地图而设计的 API 包括 OpenLayers、Leaflet、Google Maps API 和用于 JavaScript 的 ArcGIS API。后两者更为具体，因为它们是围绕特定的专有平台设计的。本节将介绍一些不同的 API 和应用程序开发方法，然后详细介绍如何使用 Leaflet API。

请注意，API 不是编程语言。相反，它是一种语言调用的组件。一些 API 支持与多种编程语言一起使用，而其他 API 则与一种特定的语言绑定在一起。例如，既有一种语言，又有一种名为 Java 的 API。Java 语言用于 Java API（和其他 API）。相比之下，.NET Framework 仅仅是一个 API。没有称为.NET 的语言。使用.NET Framework 的应用程序通常使用 C # 或 Visual Basic 语言进行编程。

### 10.1.1 Ajax 技术

Ajax 即 “Asynchronous Javascript And XML”（异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术。Ajax 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。通过在后台与服务器进行少量数据交换，Ajax 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。传统的网页（不使用 Ajax）如果需要更新内容，必须重新加载整个网页内容。

### 10.1.2 选择 Web 制图 API

当开始创建一个 web 地图时，将做出的最重要的选择之一是使用哪个 API。如果应用程序涉及到很多客户，那么这个决定会影响多年的职业活动和发展轨迹。如何选择最适合自己的需求和技能集的 API？

API 的选择通常与平台和编程语言的决定紧密相关。这两个因素会影响可以使用的 API。例如，如果要在 Android 平板电脑上运行自己的应用程序，则首先需要确定是否要构建功能完善的本机应用程序（换句话说，该应用程序可在 Google Play 中使用并可以访问设备硬件（例如相机）或仅在 Android 平板电脑上的网络浏览器中运行的设备。开发本机应用程序很可能意味着将使用 Java，而开发基于浏览器的应用程序则可以提供更大的灵活性，并且可以使用 JavaScript 和 HTML 来完成，也许使用设计为移动友好的 API（换句话说，它支持触摸手势，调整大小到设备宽度等等）。

从这个例子中可以看出，考虑开发人员熟悉的语言和平台很重要。如果自己的员工了解 Java 或 Objective C，那么在开发本地移动应用程序的选择就会增加。但是，HTML 和 JavaScript 的知识通常足以构建基于浏览器的应用程序。大多数用于 Web 制图的 FOSS API 都针对 HTML 和 JavaScript 方法。

### 10.1.3 WebGIS 前端类库介绍

#### WebGIS 前端类库介绍

##### OpenLayers

[OpenLayers](#) 是用于构建 Web 地图应用程序的成熟且功能丰富的 JavaScript API。尽管有些材料对于初学者来说很难理解，但它具有大量的文档和样本集合。

尽管 OpenLayers 对于初学者而言不像其他一些 API 那样容易使用，但它的在线帮助资源的积累以及对 FOSS 和商业来源的许多层类型的支持都比其他 API 更具优势。2014 年，OpenLayers 3 发布，这是重要的一步，因为它是对库的完全重写，不再与 OpenLayers 2 分支兼容。相比之下，当前版本的 OpenLayers 5 仍然向后兼容 OpenLayers 3 的先前版本。如果在本节课后对 JavaScript 和 Leaflet 感到满意，并决定在最终项目中使用 OpenLayers，欢迎继续进行，牢记其优势和劣势。

##### Leaflet

[Leaflet](#) 是一种较年轻的 FOSS Web 制图 API，旨在轻巧，适合移动设备且易于入门。在过去的几年中，它变得非常流行（这是在本书中进行讲授的一个原因），并且许多公司（例如 Mapbox）都将其用作自己的 API 基础。Leaflet 重点强调使用从 GeoJSON 等来源绘制的切片地图和客户端矢量图形（将在下一课中详细了解后者）。对于使用这些图层类型的基本地图，Leaflet 是一个绝佳的选择，已经吸引了许多 GIS 开发人员。

Leaflet 包含完整的 API 参考，但与 OpenLayers 相比，它只包含了少数完整的工作示例。对于初学者来说，超越这些例子可能是很棘手的；然而，API 的简单性很好地帮助了他们快速学习。

##### D3

[D3](#) 是一个 FOSS 数据可视化库，经常用于绘制图表，但也包含许多地图示例。它将数据元素绑定到页面的文档对象模型（DOM），允许有趣和灵活的数据动画和转换。尽管 D3 对于新手来说有着更陡峭的学习曲线，但它是一个很好的选择，可以用来编写带有交互式地图和图表的 web 应用程序。

## Polymaps

Polymaps 是简单的 FOSS 制图 API，主要用于将具有从 GeoJSON 和其他来源绘制的矢量特征的地图图块混搭。但是，开发人员示例还展示了 Polymaps [如何将光栅图像](#) 转换并覆盖到现有的图块集上。[k-means 聚类示例](#) 展示了 Polymaps API 的独特功能，可以动态地概括大量点。

## ModestMaps

ModestMaps 是用于显示瓦片地图的轻量级 FOSS API。就设计而言，它缺少其他 API 的许多功能。运行 JavaScript 代码需要将该代码传输到浏览器。如果知道只想显示一个地图，为什么要加载数百个函数？

### 10.1.4 Google 地图和 Bing 地图 API

Google Maps API 使开发人员有机会将自己的数据覆盖在谷歌地图的瓦片地图层之上。覆盖的数据通常通过 KML 文件提供，并显示为在客户端绘制的交互式矢量图形。开发人员可以重新设计这些图形的样式以使用自定义标记符号，还可以将这些图形绑定到弹出窗口或表格以在单击鼠标时显示其他信息。

也许 Google Maps API 的最大优势是它将 Google Maps 的外观和感觉带到了应用程序中。许多互联网用户都有使用谷歌地图的经验，当他们看到谷歌地图导航控件或地图样式时可能会感觉更舒服，即使这是嵌入到一个不熟悉的第三方应用程序中。可以说，Google Maps API 并不比上面描述的一些 FOSS API 更健壮或更易于使用；但是，它有完整的文档记录，并提供了一个大型的开发人员社区。

2018 年 7 月，谷歌地图（Google Maps）采用现收现付模式，所有客户每月可获得 200 美元的信用额度，并且必须支付超过此额度的服务使用费。根据他们在撰写本文时的定价计划，这将足以覆盖 100000 个静态地图视图或 28500 个动态地图视图。见 [this page](#) 关于谷歌定价和计划的最新信息。

微软的 Bing Maps 作为另一家大型商业地图提供商，必应地图为网络和移动应用程序提供了与谷歌范围相似的 API。必应地图为企业提供了免费使用层和基于数量的定价（参见详细信息 [here](#)）。与 Google 的一个区别是 Bing 地图 API 不太重视 KML 的使用，因为 Google 普及了 KML 格式，并且是创建 KML 文件的主要平台。

Google 和 Bing 地图 API 是 place finder 应用程序开发人员中的一个流行选择，这些应用程序显示 real estate listings , businesses , churches 但是，一些网站开始采用自由／开源软件替代方案。例如 Craigslist 在显示房地产搜索结果时采用了传单 + 开放式街道地图的方法。

### 10.1.5 其他 Web 制图 API

这些年来，出现了许多其他免费和专有的 API，它们的功能几乎与上面列出的相同。其中一些，例如 Mapbox.js 和 CARTO Maps API，与基于云的地图和位置服务相关联。请绕行阅读此 GIS Stack Exchange 帖子，其中描述了可用的 Web 制图 API。

## 10.2 LeafletJS 介绍

这一节先对 Leaflet 介绍一下，了解基本的技术背景。

### 10.2.1 JavaScript 语言介绍

JavaScript 是一种嵌入式的脚本开发语言，几乎所有的 Web 浏览器都支持 JavaScript。早期的 JavaScript 主要用来在 Web 客户机端处理一些基本的有效性验证，如今的 JavaScript 已经可以与浏览器窗口及其文档内容的各个部分进行交互。

大名鼎鼎的 Adobe 公司的 Flash 技术可谓家喻户晓，通过 Flash 技术可以实现各种复杂的交互效果，20 世纪初期各种漂亮的网站很多都是基于 Flash 开发的，也广泛应用到 WebGIS 中。但随着 HTML5 的兴起，Flash 技术的应用范围也越来越小。

基于浏览器的 JavaScript 由三个部分组成：核心 JavaScript (ECMAScript)、文档对象模型 (DOM) 和浏览器对象模型 (BOM)。

常用的文本编辑器都可以用来编写 JavaScript 代码，一些专业的文本编辑器如 UltraEdit、Editplus 以及 JavaScriptEditor 等，还提供了许多代码编辑和调试的工具。

从 Web 的发展历史中已经知道了一个 HTML 网页是静态的，随着时代的发展，已经不能满足于单纯的信息展示和阅读了，于是诞生了网页脚本语言，它是一种由浏览器执行的语言，最后流行开来并得到广泛应用的便是 JavaScript 语言了。

有了网页脚本语言支持，可以做出复杂的交互效果，比如下拉菜单，表单验证，甚至是动画效果。加上现在发展起来的 HTML5 和 CSS3 技术，甚至可以做出媲美桌面软件的效果，这也是发展的大趋势。

### 10.2.2 Leaflet 简介

Leaflet 是一个为建设移动设备友好的互动地图，而开发的现代的、开源的 JavaScript 库。它是由 Vladimir Agafonkin 带领一个专业贡献者团队开发，代码量很小，但具有开发在线地图的大部分功能。Leaflet 设计坚持简便、高性能和可用性好的思想，在所有主要桌面和移动平台能高效运作，在现代浏览器上会利用 HTML5 和 CSS3 的优势，同时也支持旧的浏览器访问。支持插件扩展，有一个友好、易于使用的 API 文档和一个简单的、可读的源代码。

## 10.3 LeafletJS 的基本用法

本书中会使用 Leaflet 作为前端的工具，但不会涉及到复杂的用法。在开始之前，还是通过一步一步的引导，快速从 Leaflet 基础开始，包括设置一个 Leaflet 地图，并配合一些标记、折线、弹窗和处理事件。

### 10.3.1 在 HTML 页面添加地图对象

这里使用了在线的库。也可以将类库下载到本地来测试，这样速度会快一些。在 head 区域引入 Leaflet CSS 文件

```
6   <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/
7     ↪leaflet.css"
8       integrity="sha512-xwE/
9     ↪Az9zrjB1phAcBb3F6JVqxf46+CDLwfLMHloNu6KEQCAWi6HcDUbeOfBIptF7tcCzusKFjFw2yuvEpDL9wQ==
10    ↪"
11      crossorigin="" />
```

在 CSS 引入后引入 Leaflet JavaScript 文件

```
9   <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"
10     integrity="sha512-gZIG9x3wUXg2hdXF6+rVkLF/
11     ↪0Vi9U8D2Ntg4Ga5I5BZpVkvx1JWbSQtXPSiUTtC0TjtG0mxa1AJPuV0CPthew=="
12     crossorigin=""></script>
```

在地图的位置设置一个带有确定 id 的 `<div>` 元素

```
19 <div id="mapid"></div>
```

确定地图区域有一个明确的高度，可以通过 CSS 来定义

```
12 <style>
13   #mapid {
14     height: 480px;
15   }
```

一个用漂亮的 Mapbox Streets 的地图图块创建的伦敦中心的地图，首先需要初始化地图，并且设置视图用选择的地理坐标和缩放级别。

```
21 var mymap = L.map('mapid').setView([51.505, -0.09], 13);
```

默认情况下，地图上的所有鼠标和触摸的交互控件都是激活的，可以执行，并且有缩放和属性控件。注意：`setView` 方法可以返回地图对象——大部分的 Leaflet 方法不返回明确的数值时就像这样，允许类似于 jQuery 的方便的方法链。

### 10.3.2 Leaflet 中的几何要素用法

除了图层之外，还可以简单的在地图上添加其他的元素，包括标记、折线、多边形、圆和弹窗。这些由 Leaflet 进行定义。

例如添加标记：

```

22 var marker = L.marker([51.5, -0.09]).addTo(mymap);
23 var circle = L.circle([51.508, -0.11], {
24   color: 'red',
25   fillColor: '#f03',
26   fillOpacity: 0.5,
27   radius: 500
28 }).addTo(mymap);
29 var polygon = L.polygon([
30   [51.509, -0.08],
31   [51.503, -0.06],
32   [51.51, -0.047]
33 ]).addTo(mymap);

```

添加一个圆也是类似方法（除了第二个参数为半径设置特定的单位），当创建对象时通过传递最后一个参数，可以控制它的外观。

添加多边形也是类似，比较简单易行。

### 10.3.3 使用弹窗（popups）展示信息

当想在地图上，通过运用 popups 在某个特定对象上附加一些信息。Leaflet 有一个简便的捷径。

```

34 marker.bindPopup("<b>Hello world!</b><br>I am a popup.");
35 circle.bindPopup("I am a circle.");
36 polygon.bindPopup("I am a polygon.");

```

尝试点击对象，绑定的 `popup()` 方法在标记 `marker` 上附带一个有特定 HTML 内容的弹窗 `popup`。

所以当点击对象时，这个弹窗 `popup` 会出现，并且 `openPopup` 方法（仅适用于标记 `markers`）就会立即打开这个附加的弹窗 `popup`。

### 10.3.4 事件处理

每当 Leaflet 上发生一些事情时，例如用户点击 marker 或者地图缩放改变时，相应的对象会发送一个事件，可以通过一个函数来监听事件，允许对用户的交互做出应答。

```

38     function onMapClick(e) {
39         L.popup()
40             .setLatLng(e.latlng)
41             .setContent("You clicked the map at " + e.latlng.toString())
42             .openOn(mymap);
43     }
44     mymap.on('click', onMapClick);

```

任一对象拥有自己的一组事件，监听器函数的第一个参数是一个事件对象——它包含有关事件发生时的有用信息。例如，地图点击事件对象有一个经纬度的属性，该属性是发生点击时的位置。例：通过 `popup` 而不是 `alert` 来改进事例

## 10.4 在 LeafletJS 中使用 WMS

如果没有这个服务，可能要自己做判断，就像文章开头所述的方法那样，或者是将数据库中的数据在客户端渲染成矢量地图，并为每个矢量要素绑定点击事件，这样做，在数据量比较小时没有太大劣势，但是一旦数据量增大，传输时间会增加，客户端渲染也需要时间，这样会影响体验。当然也可以简化矢量数据的数据量，但这会影响数据的精度。

### 10.4.1 使用 Leaflet 调用 MapServer

前端类库往往封闭了很多底层的接口，所以在使用的时候，很多时候只是接口的使用。

在前面使用 CGI 的程序中，使用了 GET 或 POST 来提交信息。每次浏览都是一个新的 URL 。页面也会整体重新生成一次。

这里重新使用 Leaflet 加载地图服务来看一下效果，感受一下使用 Ajax 的体验。

下面是一个实例，访问 URL 为 [http://webgis.cn/leaflet\\_using\\_wms\\_tms/wms-example1.html](http://webgis.cn/leaflet_using_wms_tms/wms-example1.html) 。部分代码如下：

```

21 <div id='map'></div>
22 <script type="text/javascript">
23     var map = L.map('map', {
24         center: [0, 0],
25         zoom: 1
26     });
27

```

(下页继续)

(续上页)

```

28 var wmsLayer = L.tileLayer.wms(
29   'http://webgis.cn/cgi-bin/mapserv?map=/owg/mfb2.map&',
30   {
31     layers: 'states',
32     format: 'image/png',
33     transparent: true,
34   }).addTo(map);
35 </script>

```

这里 `wms()` 函数调用的是地图服务。

注意其中的 `L.tileLayer.wms()` 函数。这其中用到了 WMS 服务，需要在后面进行说明。但是使用这种地图的访问方式，可以明显发现交互使用要方便很多。

#### 10.4.2 访问 GetFeatureInfo

不要被这个 URL 吓到，它很长，确实！这是请求 WMS 标准的 `GetFeatureInfo` 规定的参数，通常不会自己手动构造它，可以使用 OpenLayers 等工具构造它。调用后，MapServer 将以上的 `TEMPLATE` 配置的字段用实际内容（数据库中的内容）代替后，输出的结果如下：

LeafletJS 本身并不支持 `GetFeatureInfo`，但是有插件可以进行功能扩展。

要实现的效果就是当用户点击地图上某一块地物的时候，从服务器返回对应地物的属性信息，并以弹出框的形式显示。这里主要讲其后端的通信与处理过程。

#### 10.4.3 基础底图与专题层

基础底图为地图提供基本的底图环境。换言之，这通常不是人们看地图的主要原因，但没有它，地图将很难解释。在线使用的最常见的基础地图是行政区划，交通道路和遥感图像。

尽管一个基础底图可能包含许多子层（如道路、湖泊、建筑物等），但这些子层通常融合在一起，形成一组栅格化的平铺图像，并在网络地图中作为一个单层处理。这些瓦片地图通常由数千或数百万个预先绘制的图像组成，这些图像保存在服务器上，并在人们在地图周围移动时传递给 Web 浏览器。

在过去的几年中，服务器在某些应用中已经流行了将基础底图作为矢量坐标块，有时称为“矢量瓦片”。将 `BaseMap` 显示为矢量而不是栅格化地图，允许在可视化中具有更大的灵活性。例如，当标签保持右侧朝上时，可以旋转地图。如果在智能手机上查看 Google 地图应用程序，就可以看到一个矢量地图正在运行。

有时两个平铺层将一起工作，形成一个基础底图。例如，可能有一个带有航空图像的平铺层和一个带有矢量道路覆盖的第二个平铺层，该矢量道路覆盖被设计为位于图像的顶部。（在

Google Maps 中，当选中 Labels 项时会出现此项）。将这两个 tileset 分开会占用更少的磁盘空间，并且更容易更新图像。

专题层（也称为业务层或操作层）位于基础底图的顶部。他们是人们来访问地图的原因。如果放在基础底图上，它们可能不是每个人都感兴趣，但是当放在关注的 Web 地图上时，它们是感兴趣的主要层。如果地图名为“费城农贸市场”，那么农贸市场就是主题层。如果地图标题是“北美鸟类的迁徙模式”，那么迁徙模式就是主题层。

与基础底图一样，专题图层有时也显示为瓦片地图；但是，由于某些数据的性质迅速变化，这并不总是可能的。例如，如果需要显示警车的实时位置，则不能依赖预先绘制的平铺，必须使用其他方法绘制数据。有各种 Web 服务，例如 WMS，它们旨在以这种方式动态绘制地图。可以把这些用于主题层。另一个选项是查询服务器上的所有数据，并使用浏览器绘制这些数据。这种方法非常适合于诸如弹出窗口之类的交互元素。

专题层与基础地图层一起工作，形成一个有效的网络地图。有趣的是，主题层并不总是第一层。在 2009 年的一篇博客文章（不再在线）中，Esri 制图师 Charlie Frye 描述了一种“地图三明治”方法，其中一个主题层（可以平铺也可以不平铺）放置在两个平铺的基础地图层之间，这两个层提供地理背景。底层具有地形特征，顶层具有标志和边界。这是“面包”，中间的主题层是“肉”，ArcMap 和 ArcGIS 联机中的 Esri “浅灰色 Canvas” 底图使用了这种技术。自由／开源软件技术肯定也允许这种方法。

自己的地图可能包含多个主题层，允许用户打开和关闭。要实现这一点，可以使用具有多个子层的单个 Web 服务，也可以使用每个子层都包含单个层的多个 Web 服务。但是，为了保持应用程序的可用性和相对快速的性能，应该避免在 Web 地图中包含许多主题层。

#### 10.4.4 添加切片地图

WMS 请求每次都需要应用程序来处理数据并渲染成地图，所以前面的示例虽有所改进，但速度尚不理想。在这里修改为地图切片服务，调用方面与 WMS 完全一样，只是修改一下数据源：

```
27 var wmsLayer = L.tileLayer.wms(
28   "http://47.94.22.90:6001/service?", {
29     layers: "maplet_WorldCountry",
30     format: "image/png",
31     transparent: true,
32     attribution: 'Map &copy; <a href='http://www.osgeo.cn/map/'>
```

←

```
33   }
34 ).addTo(map);
```

## 10.5 在 Leaflet 中调用 WFS

使用 WFS 返回的数据不能直接查看，为了绘制，需要访问坐标与绘图的方法；但另一方面，可以使用成熟的类库来实现这样的任务。

虽然 WFS 的请求与响应的语法初看起来有些复杂，不过在实际使用过程中，并不需要手工来编写。GIS 软件中通常都支持查看与发布 WFS 服务。

正如上面所演示的，MapServer 可将图层发布为 WFS 服务，而且这是默认设置，并不需要用户额外的配置；其他开源 GIS 服务器软件，例如 MapServer 与 Degree 等也都支持创建 WFS 服务。在 MapServer 中需要进行相关的配置。

在商业软件领域，ESRI 的 ArcGIS Server 也可用于发布 WFS 服务。不过，ESRI 开发了其自身基于 REST 的“要素服务”，用于完成 WFS 相同功能。ESRI 的 ArcGIS API for JavaScript 及其编辑控件使用的是其自身的要素服务，而不是标准的 WFS。

对于大多数的 Web 地图 API，通常将 WFS 作为一图层。例如，在 OpenLayers 中，要加入一个 WFS 服务，与前面内容介绍的加入 KML 与 GeoJSON 数据一样，使用 OpenLayers 的 Layer.Vector 即可，只是协议不同。对于要素的样式与符号，与前面内容介绍的也完全一样。

客户端 GIS 软件一般都能查看 WFS 服务，不过 Leaflet 以及许多其他轻量级的开源 Web 地图 API 并不支持 WFS。

### 10.5.1 Leaflet 中调用 WFS

WFS 服务返回的是数据，而不是地图。返回数据意味着更多信息，用户可以对信息解析，来完成更复杂的任务；但同时也意味着绘图的任务交由客户端来完成，需要更多的编码工作。

#### 如何配置 WFS 服务

使用 Leaflet 绘制矢量数据。在 Web 浏览器或客户端中绘制矢量数据，主要介绍了 KML 与 GeoJSON 格式的数据。

Leaflet 可以使用 GeoJSON 数据进行绘制。要显示 WFS 服务返回的数据，使用 JSON 作为返回的数据类型，需要在 MapServer 中开启。

以下内容是一个简单的示范，对应的 Mapfile 为 <http://webgis.cn/mfw2.map>：

```

1 OUTPUTFORMAT
2   NAME "geojson"
3   DRIVER "OGR/GEOJSON"
4   MIMETYPE "application/json; subtype=geojson"
5   FORMATOPTION "STORAGE=stream"

```

(下页继续)

(续上页)

```

6 FORMATOPTION "FORM=SIMPLE"
7 END

```

或者可以开启更多的类型：

```

1 OUTPUTFORMAT
2   NAME "geojson"
3   DRIVER "OGR/GEOJSON"
4   MIMETYPE "application/json; subtype=geojson"
5   FORMATOPTION "STORAGE=stream"
6   FORMATOPTION "FORM=SIMPLE"
7 END
8 OUTPUTFORMAT
9   NAME "kmz"
10  DRIVER KMZ
11  MIMETYPE "application/vnd.google-earth.kmz"
12  IMAGEMODE RGB
13  EXTENSION "kmz"
14 END
15 OUTPUTFORMAT
16   NAME "png8"
17   DRIVER AGG/PNG8
18   MIMETYPE "image/png; mode=8bit"
19   IMAGEMODE RGB
20   EXTENSION "png"
21   FORMATOPTION "QUANTIZE_FORCE=on"
22   FORMATOPTION "QUANTIZE_COLORS=256"
23   FORMATOPTION "GAMMA=0.75"
24 END

```

并且在 METADATA 中声明：

```
"wfs_getfeature_formatlist" "geojson"
```

## 示例

这里使用的 Mapfile，通过 <http://webgis.cn/751> 访问查看。URL 解析如下。

```

>>> [print(i, x) for i, x in enumerate(re.split('[?&]', url))]
0 http://webgis.cn/cgi-bin/mapserv
1 map=/owg/mfw2.map
2 SERVICE=WFS

```

(下页继续)

(续上页)

```

3 VERSION=2.0.0
4 REQUEST=GetCapabilities

```

注意，由于数据较大，使用择舍器选择了中国的范围。但是一般网络情况下加载还是比较慢。在这里，定义了如下函数来访问 WFS 数据，并加到地图：

```

1  function loadWFS(layerName, epsg) {
2      var param = {
3          service: 'WFS',
4          version: '2.0.0',
5          request: 'GetFeature',
6          typeName: layerName,
7          outputFormat: 'geojson',
8          filter: '<Filter><PropertyIsEqualTo>
9              <PropertyName>NAME</PropertyName>
10             <Literal>CHINA</Literal>
11             </PropertyIsEqualTo></Filter>'
12      };
13      var rooturl = 'http://webgis.cn/cgi-bin/mapserv?map=/owg/mfw2.map';
14      var u = rooturl + L.Util.getParamString(param, rooturl);
15      $.ajax({
16          url: u,
17          dataType: 'json',
18          success: loadWfsHandler,
19      });
20      var layer;
21
22      function loadWfsHandler(data) {
23          console.log(data);
24          layer = L.geoJson(data, {}).addTo(map);
25      }
26  }

```

效果可通过 <http://webgis.cn/wfs-leaflet1.html> 查看，如图 图 10.1 所示：



图 10.1：使用 Leaflet 加载的 WFS

### 10.5.2 LeafletJS 加载 WFS 自定义样式

WMS 由服务器端完成地图渲染的任务，如果要对地图的样式进行定制，可以使用 SLD，将样式的定义文件传递给服务器，服务器再返回地图的结果。而 WFS 返回的数据则可以由客户端来完成样式的调整与渲染。

在这种情况下不需要对服务器端进行任何的修改，服务器只提供数据即可。

在客户端，可以定义匿名函数实现样式的定义。

```
1 style:function(feature) {
2     return {
3         stroke:true,
4         color: '#333333',
5         opacity: 1,
6         fillOpacity: 0.1,
7         fillColor: '#333333',
8         weight:1
9     }
10 },
```

效果可通过 <http://webgis.cn/wfs-leafleta2.html> 查看，如 图 10.2 所示：



图 10.2：在 Leaflet 中自定义样式

## 10.6 访问 MapProxy 的不同服务形式

MapProxy 同时以不同地图瓦片标准形式提供服务，包括 WMS-C、TMS、WMTS。

### 10.6.1 WMS

WMS 是最支持最好的。切片可以通过与 WMS 提供的接口访问。

### 10.6.2 WMS-C

MapProxy WMS 服务还支持 OSGeo 的 WMS 平铺客户端建议。

如果在 GetCapabilities 请求中添加 tiled=true，MapProxy 将向 WMS 功能文档添加有关内部瓦片结构的元数据。支持 WMS-C 的客户机可以使用此信息在确切的瓦片边界请求瓦片。MapProxy 可以返回这些请求所需的瓦片，性能与 TMS 服务相当。

当 tiled=true 添加到 GetMap 请求和对于以下请求，它将返回 WMS 服务异常如果请求的图像大小或格式不同，则不匹配确切的平铺边界。

### 10.6.3 TMS 访问

在默认的 Demo 中，有错误，导致 TMS 无法访问。

Demo 中为：

[http://121.42.29.253:9011/tms/1.0.0/maplet\\_WorldRailway/webmercator/1/0/1.png](http://121.42.29.253:9011/tms/1.0.0/maplet_WorldRailway/webmercator/1/0/1.png)

需要修改为：

[http://121.42.29.253:9011/tms/1.0.0/maplet\\_WorldRiver/EPSC3857/0/1/1.png](http://121.42.29.253:9011/tms/1.0.0/maplet_WorldRiver/EPSC3857/0/1/1.png)

关于在 Leaflet 中的调用，参见：

[https://gis.stackexchange.com/questions/212102/wrong-coordinates-of-mapproxy-tms-in-leaflet-0-7?  
noredirect=1](https://gis.stackexchange.com/questions/212102/wrong-coordinates-of-mapproxy-tms-in-leaflet-0-7?noredirect=1)

似乎应该都使用 EPSG 4326 投影才可以。

#### 10.6.4 WMTS 访问

使用实例。