

New Users

Information to help a New User get going with MapServer.

MapServer New Users

Revision: 1.0.0.00

Date: 2005-04-04

Author: David Fawcett

Contact: david.fawcett@moea.state.mn.us

Contents

- [1 MapServer Overview](#)
- [2 MapScript](#)
- [3 Anatomy of a MapServer Application](#)
- [4 Getting Started](#)
 - ◆ [4.1 Hardware Requirements](#)
 - ◆ [4.2 Software Requirements](#)
 - ◆ [4.3 Skills](#)
- [5 Build your first MapServer Application](#)
 - ◆ [5.1 Get MapServer Running](#)
 - ◆ [5.2 Get Demo Running](#)
- [6 Making the Site Your Own](#)
 - ◆ [6.1 Adding Data to Your Site](#)
 - ◆ [6.2 Projections](#)
- [7 Enhancing your site](#)
 - ◆ [7.1 Adding Query Capability](#)
 - ◆ [7.2 Interfaces](#)
 - ◆ [7.3 Data Optimization](#)
- [8 How do I get Help?](#)
 - ◆ [8.1 Documentation](#)
 - ◆ [8.2 List Archives](#)
 - ◆ [8.3 Ask for help](#)
 - ◆ [8.4 Examples](#)

1 MapServer Overview

In its most basic form, MapServer is a CGI program that sits inactive on your Web server. When a request is sent to MapServer, it uses information passed in the request URL and the Map File to create an image of the requested map. The request may also return images for legends, scale bars, reference maps, and values passed as CGI variables. Here is a conceptual [diagram](#) of a typical MapServer application.

MapServer can be greatly extended and customized. It can be built to support many different input data formats and output types. This is done at the time the MapServer binary is compiled. See the [MapServer Home Page](#) for a full list of current features. Many of the features that are not 'built-in', are enabled through the use of [OGR](#).

2 MapScript

MapScript provides a scripting interface for MapServer for the construction of Web and stand-alone applications. MapScript is used independently of CGI MapServer, it is a loadable module that adds

MapServer capability to your favorite scripting language. MapScript currently exists in Php, Perl, Python, Ruby, Tcl, Java, and C# flavors. (Describe how SWIG fits in).

This New User Guide will not explicitly discuss MapScript, check out: [MapScript HowTo's](#) , [SWIG MapScript Reference](#), [MapScript Download](#)

3 Anatomy of a MapServer Application

A simple MapServer application consists of:

- **Map File** - a structured text configuration file for your MapServer application. It defines the area of your map, tells the MapServer program where your data is and where to output images. It also defines your map layers, including their data source, projections, and symbology. It usually as a .map extension.
- **Geographic Data** - MapServer can utilize many geographic data source types. The default format is the ESRI shapefile. Many other data formats can be supported, this is discussed further below in [Adding data to your site](#).
- **HTML Pages** - the interface between the user and MapServer . They normally sit in Web root. In it's simplest form, MapServer can be called to place a static map image on a html page. To make the map interactive, the image is placed in an html form on a page.

CGI programs are 'stateless', every request they get is new and they don't remember anything about the last time that they were hit by your application. For this reason, every time your application sends a request to MapServer, it needs to pass context information (what layers are on, where you are on the map, application mode, etc.) in hidden form variables or URL variables.

A simple application may include two html pages:

- ◆ **Initialization File** - uses a form with hidden variables to send an initial query to the http server and MapServer. This form could be placed on another page or be replaced by passing the initialization information as variables in a URL.
- ◆ **Template File** - controls how the maps and legends output by MapServer will appear in the browser. By referencing MapServer CGI variables in the template html, you allow MapServer to populate them with values related to the current state of your application (e.g. map image name, reference image name, map extent, etc.) as it creates the html page for the browser to read. The template also determines how the user can interact with the MapServer application (browse, zoom, pan, query).
- **MapServer CGI** - The binary or executable file that receives requests and returns images, data, etc. It sits in the cgi-bin or scripts directory of the http server. The Web server user must have execute rights for the directory that it sits in, and for security reasons, it should not be in the web root.
- **HTTP Server** - serves up the html pages when hit by the user's browser. You need a working HTTP (Web) server, such as [Apache](#) or Microsoft Internet Information Server, on the machine on which you are installing MapServer.

4 Getting Started

The best strategy for learning how to build MapServer applications and understand how it all works is to start very simple. Even if your ultimate goal is to build a very complex, cutting edge application, make the first step a small one. Get the simple demo running and then morph it into your own application. Here is what you need to get started:

4.1 Hardware Requirements

MapServer runs on Linux, Windows, Mac OS X, Solaris, and more. To compile or install some of the required programs, you may need administrative rights to the machine. People commonly ask questions about minimum hardware specifications for MapServer applications, but the answers are really specific to the individual application. For development and learning purposes, a very minimal machine will work fine.

4.2 Software Requirements

You need a working and properly configured HTTP (Web) server, such as [Apache](#) or Microsoft Internet Information Server, on the machine on which you are installing MapServer. If you are on a Windows machine, and you don't have a HTTP server installed, you may want to check out [MS4W](#), which will install a preconfigured HTTP server, MapServer, and more. The [FGS Linux Installer](#) provides similar functionality for several Linux distributions.

You will also need a Web browser, and a text editor (vi, emacs, notepad, homesite) to modify your html and Map files.

4.3 Skills

In addition to learning how the different components of a MapServer application work together and learning Map File syntax, building a basic application requires some conceptual understanding and proficiency in several skill areas.

You need to be able to create or at least modify [HTML](#) pages and understand how HTML forms work. Since the primary purpose of a MapServer application is to create maps, you will also need to understand the basics of geographic data and likely, map projections. As your applications get more complex, skills in SQL, DHTML/Javascript, Java, databases, expressions, compiling, and scripting may be very useful.

5 Build your first MapServer Application

5.1 Get MapServer Running

Download the appropriate source code or pre-compiled binaries for your operating system from [OSGeo](#) or [MapTools](#). [FGS](#) from [maptools.org](#) provides a stand-alone environment that includes Apache http server, MapServer, and PHP MapScript.

Compilation instructions are available for [Unix/Linux](#) and [Windows](#), with additional [Build Notes for RedHat](#). [Refractions Research Inc](#) also maintains a set of [Linux RPMs](#). If you are on the Windows platform, compilation can be quite challenging, for your first application, it is recommended that you utilize pre-compiled Windows binaries.

Once MapServer is installed, there are two ways that you can test it. First, at the command line, type 'mapserv -v', you should see a message describing your MapServer install, something like:

```
MapServer version 4.8.0-beta2 OUTPUT=GIF OUTPUT=PNG
OUTPUT=JPEG OUTPUT=WBMP SUPPORTS=PROJ SUPPORTS=FREETYPE
SUPPORTS=WMS_SERVER INPUT=SHAPEFILE DEBUG=MSDEBUG
```

You can also send a HTTP request directly to the MapServer CGI program without passing any configuration variables (e.g. <http://your.domain.name/cgi-bin/ms4/mapserv.exe>). If you receive the message, 'No query information to decode. QUERY_STRING not set.', your installation is working.

5.2 Get Demo Running

Download the [MapServer Demo](#). UnZip it and follow the directions in ReadMe.txt. You will need to move the demo files to their appropriate locations on your HTTP server, and modify the Map File and html pages to reflect the paths and URLs of your server. Next, point your browser to init.html and hit the 'initialize button'. If you get errors, verify that you have correctly modified the demo files.

6 Making the Site Your Own

Now that you have a working MapServer demo, you can use the demo to display your own data. Add new LAYERs to your Map file that refer to your own geographic data layers. (You will probably want to delete the existing layers or set their status to OFF.)

Unless you are adding layers that fall within the same geographic area as the demo, modify [MAP EXTENT](#) to match the extent of your data. To determine the extent of your data, you can use [OGRInfo](#). If you have access to a GIS, you could use that as well. Map EXTENT needs to be in the units of your output projection.

If you add geographic data layers of different projections, you will need to modify your Map File to add a PROJECTION block to the [MAP Object](#) (output projection) and each of the [LAYER](#) (existing layer projection).

6.1 Adding Data to Your Site

MapServer supports several data input formats 'natively', and many more if it is compiled with the open source libraries [GDAL](#) and [OGR](#).

6.1.1 Vector Data

Vector data includes features made up of points, lines, and polygons. MapServer supports ESRI shapefiles by default, but it can be compiled to support spatially enabled databases such as [PostgreSQL-PostGIS](#), [Geography Markup Language \(GML\)](#), [MapInfo](#), delimited text files, and more formats with [OGR](#). See the [OGR HowTo](#) for more info.

See the [Vector Data Reference Guide](#) for examples on how to add different geographic data sources to your MapServer project.

6.1.2 Raster Data

Raster data is image or grid data. By default, MapServer supports Tiff/GeoTiff, and EPPL7. With [GDAL](#), it supports GRASS, Jpeg2000, ArcInfo Grids, and [more formats](#). If you do compile MapServer with GDAL, which includes tiff support, do not compile with native tiff support, as this will cause a conflict. More specific information can be found in the [Raster Data HowTo](#).

6.2 Projections

Because the earth is round and your monitor (or paper map) is flat, distortions will occur when you display geographic data in a two-dimensional image. Projections allow you to represent geographic data on a flat surface. In doing so, some of the original properties (e.g. area, direction, distance, scale or conformality) of the data will be distorted. Different projections excel at accurately portraying different properties. A good [primer](#) on map projections can be found at the University of Colorado.

With MapServer, if you keep all of your spatial data sets in the same projection (or unprojected Latitude and

Longitude), you do not need to include any projection info in your Map File. In building your first MapServer application, this simplification is recommended.

On-the-fly projection can be accomplished when MapServer is compiled with [Proj.4](#) support. Instructions on how to enable Proj.4 support on Windows can be found on the old [Wiki](#).

7 Enhancing your site

7.1 Adding Query Capability

There are two primary ways to query spatial data. Both methods return data through the use of templates and CGI variable replacement. A [QUERYMAP](#) can be used to map the results of the query.

To be queryable, each mapfile [LAYER](#) must have a [TEMPLATE](#) defined, or each [CLASS](#) within the [LAYER](#) must have a [TEMPLATE](#) defined. More information about the CGI variables used to define queries can be found in the [MapServer CGI Reference](#).

7.1.1 Attribute queries

The user selects features based on data associated with that feature. 'Show me all of the lakes where depth is greater than 100 feet', with 'depth' being a field in the shapefile .dbf or the spatial database. Attribute queries are accomplished by passing query definition information to MapServer in the URL (or form post). `Mode=itemquery` returns a single result, and `mode=itemnquery` returns multiple result sets.

The request must also include a [QLAYER](#), which identifies the layer to be queried, and a [QSTRING](#), which contains the query string. Optionally, [QITEM](#), can be used in conjunction with [QSTRING](#) to define the field to be queried. Attribute queries only apply within the [EXTENT](#) set in the map file.

7.1.2 Spatial queries

The user selects features based on a click on the map or a user-defined selection box. Again the request is passed through a URL or form post. By setting `mode=QUERY`, a user click will return the one closest feature. In `mode=NQUERY`, all features found by a map click or user-defined selection box are returned. Additional query options can be found in the [MapServer CGI Reference](#)

7.2 Interfaces

Once they have a basic MapServer application going, people often want to add additional functionality and a more sophisticated interface that isn't possible with a straight HTML template. One can use DHTML, Java, and Flash on the client side to improve the interface. Demos for [DHTML](#) and [Flash](#) interfaces are available for download. [JBox](#) is a Java applet for building MapServer interfaces.

7.3 Data Optimization

Data organization is at least as important as hardware configuration in optimizing a MapServer application for performance. MapServer is quite efficient at what it does, but by reducing the amount of processing that it needs to do at the time of a user request, you can greatly increase performance. Here are a few rules:

- **Index Your data** - By creating spatial indexes for your shapefiles using [shptree](#). Spatial indexes should also be created for spatially aware databases such as PostGIS and Oracle Spatial.
- **Tile Your Data** - Ideally, your data will be 'sliced up' into pieces about the size in which it will be displayed. There is unnecessary overhead to searching through a large shapefile or image of which

you are only going to display a small area. By breaking the data up into tiles and creating a tile index, MapServer only needs to open up and search the data files of interest. Shapefile data can be broken into smaller tiles and then a tileindex shapefile can be created using [tile4ms](#). A shapefile tileindex for raster files can be created using [gdaltindex](#).

- **Pre-Classify Your Data** - MapServer allows for the use of quite complex [EXPRESSIONs](#) to classify data. However, using logical and regular expressions is more resource intensive than string comparisons. To increase efficiency, you can divide your data into classes ahead of time, create a field to use as the CLASSITEM and populate it with a simple value that identifies the class, such as 1,2,3, or 4 for a four class data set. You can then do a simple string comparison for the class EXPRESSION.
- **Pre-Process Your Images** - Do resource intensive processing up front. See the [Raster HowTo](#)
- **Generalize for Overview** - create a more simple, generalized data layer to display at small scales, and then use scale-dependent layers utilizing [LAYER MINSCALE](#) and [LAYER MAXSCALE](#) to show more detailed data layers as the user zooms in. This same concept applies to images.

8 How do I get Help?

8.1 Documentation

MapServer documentation takes many forms

[References](#) -formal documentation for syntax, structure, etc.

[HowTo's](#) -step by step instructions

[Tutorials](#) - detailed tutorials

[FAQ's](#) - frequently asked questions

[Error Reference](#)

8.2 List Archives

[Archive](#) of MapServer Users list. Check here before posting a question to the listserv, your question has likely been asked before...

[Public, Searchable Archive of MapServer-Users](#) is available also.

8.3 Ask for help

8.3.1 Users Listserv

Register and post questions to the [MapServer Users](#) listserv. Questions to the list are usually answered quickly and often by the developers themselves. A few things to remember:

1. Search the archives for your answer first, people get tired of answering the same questions over and over.
2. Provide version and configuration information for your MapServer installation, and relevant snippets of your map and template files.
3. Always post your responses back to the whole list, as opposed to just the person who replied to your question.

8.3.2 IRC

mapserver users and developers can be found on Internet Relay Chat. The channel is #mapserver on irc.freenode.net. [More Information](#)

8.4 Examples

8.4.1 Gallery

[Examples](#) of MapServer sites

8.4.2 Tutorial

Perry Nacionales built a great [Tutorial](#) on how to build a MapServer application. You are invited to extend the collection of examples if you see cases that are missing.

8.4.3 Test Suite

[Demonstration](#) of some MapServer functionality

8.4.4 Books

[Web Mapping Illustrated](#) , a new book by Tyler Mitchell that describes well and provides real-world examples for the use of Web mapping concepts, Open Source GIS software, MapServer, Web services, and PostGIS.

[Mapping Hacks](#) , by Schuyler Erle, Rich Gibson, and Jo Walsh, creatively demonstrates digital mapping tools and concepts. MapServer only appears in a handful of the 100 hacks, but many more are useful for concepts and inspiration.

[Beginning MapServer: Opensource GIS Development](#) , by Bill Kropla, is a new book focusing on MapServer. So new, I haven't seen it yet. According to the publisher, it covers installation and configuration, basic MapServer topics and features, incorporation of dynamic data, advanced topics, MapScript, and the creation of an actual application.

8.4.5 Online Demo

Hopefully we will have this some day...