GeoMOOSE Documentation

Release 2.X

Dan "Ducky" Little

October 23, 2014

CONTENTS

Welcome to the GeoMOOSE website. For first time users, we hope this site is easy to use and understand. For returning users, we hope this is a pleasant new experience. This new format for the website has been chosen for its ease of maintenance, ease of portability, and compact delivery size.

CHAPTER

ONE

WHAT IS GEOMOOSE?

GeoMOOSE is a Web Client Javascript Framework for displaying distributed cartographic data. GeoMOOSE has a number of strengths including modularity, configurability, and delivers a number of core functionalities in its packages. GeoMOOSE is also very light weight for servers making it easy to handle a large number of users, with a large number of layers, and a large number of services without stressing a server.

The GeoMOOSE core is written using JavaScript and HTML. It is entirely possible to run GeoMOOSE with nothing more than a basic webserver (Nginx, Apache, IIS). But besides the basic client core, GeoMOOSE also comes prepackaged with a number of built in services written in PHP. These services add the ability to perform drill-down identify operations, selection operations, and search datasets. If you have existing scripts that perform similar functions, GeoMOOSE can be tuned to work with those services, no matter which language they were written.

Being an open source project GeoMOOSE is also built upon other open source projects:

- MapServer
- OpenLayers

CHAPTER

TWO

GEOMOOSE GALLERY

See working examples of real-world GeoMOOSE implementations. By viewing the GeoMOOSE Gallery.

CHAPTER

THREE

MORE INFORMATION ABOUT GEOMOOSE

3.1 GeoMOOSE License

3.1.1 License Summary

The GeoMOOSE license is an MIT based license.

3.1.2 License Text

3.2 Gallery

We are currently working on porting the gallery to the new website.

Please see the old gallery in the interim.

3.3 Mailing Lists

Please join the GeoMOOSE Mailing List The GeoMOOSE Mailing Lists are the best way to contact GeoMOOSE developers and other users for help.

Please follow this link for information on signing up for the mailing lists.

Developers regularly follow the mailing lists and try to assist users in as immediate fashion as possible.

3.4 Commercial Support

List of companies that offer commercial support as a service. If you would like to be added to this list please email the GeoMOOSE developer mailing list and we will get your company added.

Companies that offer commercial support for installing, configuration, custom development and hosting:

- Excensus, LLC - Contact Dan "Ducky" LIttle, dan.little ~at~ excensus.com
- SharedGeo
- Houston Engineering, Inc., Contact Brian Fischer, bfischer ~at~ houstoneng.com

3.5 GeoMOOSE Demo

This demo is the standard bearer for the vast majority of GeoMOOSE applications. It features all of the common features being used and gives the user a good feel for the functionalities delivered by GeoMOOSE.

View the Demo Here

3.6 Downloads

3.6.1 GeoMOOSE 2.X

GeoMOOSE 2.4RC1 is the latest in the GeoMOOSE series.

- Download GeoMOOSE 2.4RC1
- Download GeoMOOSE 2.4RC1 for MS4W
- Download GeoMOOSE 2.4RC1 for MS4W - Web Mercator Demo

GeoMOOSE 2.2 is the latest in the GeoMOOSE series.

- Download GeoMOOSE 2.2
- Download GeoMOOSE 2.2 for MS4W
- Download GeoMOOSE 2.2 for MS4W - Web Mercator Demo

3.6.2 GeoMOOSE 1.X

GeoMOOSE 1.6.1 is the latest in the GeoMOOSE 1.X series

- Download GeoMOOSE 1.6.1

- Download GeoMOOSE 1.6.1 for MS4W

Nightly Build Packages

These packages are built nightly from SVN. There are no promises these packages will work as they are built from trunk.

- Download GeoMOOSE Nightly Build
- Download GeoMOOSE Nightly Build for MS4W
- Download GeoMOOSE Nightly Build for MS4W - Web Mercator Demo

Other Packages

If you're having problems with the download links above or are looking for an older package check here

MS4W Installation Note

Prior to unzipping the folder, please ensure you have the MS4W package installed on configured. There are installation instructions located *here*.

Note: The MS4W package can also be downloaded from http://www.maptools.org.

If you are upgrading from a previous GeoMOOSE version, ensure you back up all files prior to extracting this zip archive.

3.6.3 SourceForge / SVN

GeoMOOSE SourceForge Page

GeoMOOSE maintains a project page on SourceForge. We also use SourceForge for our download host. On the sourceforge project page you can find the most current information on checking out the latest version of the code. The current branch is "moose2-trunk".

CHAPTER

FOUR

GEOMOOSE NEWS

4.1 3/25/2011 - GeoMOOSE 2.4RC1 Ready for Download

After over a year we have assembled the first release candidate for GeoMOOSE 2.4. This integrates a lot of minor enhancements and fixes that have been contributed to the code, please download and test!

4.2 2/18/2010 - GeoMOOSE 2.2 Officially Released

The offical GeoMOOSE 2.2 Release has arrived! This is the version is worth the upgrade from any 1.X series of GeoMOOSE installations. View the full release notes: *GeoMOOSE 2.2 Release*.

4.3 2/18/2010 - Web Mercator Demo Added

To show case the new capabilities of 2.2, including use of Web Mercator layers, we've put together a new demo. *GeoMOOSE Demo*

4.4 2/3/2010 - NEW WEBSITE!

The new website is much improved:

- We have live demos.
- A new Gallery!
- A new Trac ticketing system that allows any GeoMOOSE user to submit a ticket to GeoMOOSE developers.

4.5 2/3/2010 - GeoMOOSE 2.2 RC1 Released

This is THE real 2.2 update. This is the first RC, we will take another week for bugs to be filed in the new Trac system. View the *GeoMOOSE 2.2 Release* and check out all the new features!

4.6 10/9/2009 - GeoMOOSE 2.0.1 Released

Great update! Download and Install! View The *GeoMOOSE 2.0.1 Release*.

4.7 8/20/2009 - Updated Documentation

We've added some important new documentation. UN*X installation instructions (for packages starting with 2.0.1) and a comprehensive list of configuration options that can be set in the mapbook. Check 'em out!

4.8 7/21/2009 - GeoMOOSE 2.0 Released

GeoMOOSE 2.0 released! See downloads to get your copy. Also, we've rolled out this shiney new website!

CHAPTER

FIVE

MORE INFORMATION ABOUT GEOMOOSE

5.1 GeoMOOSE License

5.1.1 License Summary

The GeoMOOSE license is an MIT based license.

5.1.2 License Text

5.2 Gallery

We are currently working on porting the gallery to the new website.

Please see the old gallery in the interim.

5.3 Mailing Lists

Please join the GeoMOOSE Mailing List The GeoMOOSE Mailing Lists are the best way to contact GeoMOOSE developers and other users for help.

Please follow this link for information on signing up for the mailing lists.

Developers regularly follow the mailing lists and try to assist users in as immediate fashion as possible.

5.4 Commercial Support

List of companies that offer commercial support as a service. If you would like to be added to this list please email the GeoMOOSE developer mailing list and we will get your company added.

Companies that offer commercial support for installing, configuration, custom development and hosting:

- Excensus, LLC - Contact Dan "Ducky" LIttle, dan.little ~at~ excensus.com
- SharedGeo
- Houston Engineering, Inc., Contact Brian Fischer, bfischer ~at~ houstoneng.com

5.5 GeoMOOSE Demo

This demo is the standard bearer for the vast majority of GeoMOOSE applications. It features all of the common features being used and gives the user a good feel for the functionalities delivered by GeoMOOSE.

View the Demo Here

5.6 Downloads

5.6.1 GeoMOOSE 2.X

GeoMOOSE 2.4RC1 is the latest in the GeoMOOSE series.

- Download GeoMOOSE 2.4RC1
- Download GeoMOOSE 2.4RC1 for MS4W
- Download GeoMOOSE 2.4RC1 for MS4W - Web Mercator Demo

GeoMOOSE 2.2 is the latest in the GeoMOOSE series.

- Download GeoMOOSE 2.2
- Download GeoMOOSE 2.2 for MS4W
- Download GeoMOOSE 2.2 for MS4W - Web Mercator Demo

5.6.2 GeoMOOSE 1.X

GeoMOOSE 1.6.1 is the latest in the GeoMOOSE 1.X series

- Download GeoMOOSE 1.6.1

- Download GeoMOOSE 1.6.1 for MS4W

Nightly Build Packages

These packages are built nightly from SVN. There are no promises these packages will work as they are built from trunk.

- Download GeoMOOSE Nightly Build
- Download GeoMOOSE Nightly Build for MS4W
- Download GeoMOOSE Nightly Build for MS4W - Web Mercator Demo

Other Packages

If you're having problems with the download links above or are looking for an older package check here

MS4W Installation Note

Prior to unzipping the folder, please ensure you have the MS4W package installed on configured. There are installation instructions located *here*.

Note: The MS4W package can also be downloaded from http://www.maptools.org.

If you are upgrading from a previous GeoMOOSE version, ensure you back up all files prior to extracting this zip archive.

5.6.3 SourceForge / SVN

GeoMOOSE SourceForge Page

GeoMOOSE maintains a project page on SourceForge. We also use SourceForge for our download host. On the sourceforge project page you can find the most current information on checking out the latest version of the code. The current branch is "moose2-trunk".

CHAPTER

SIX

GEOMOOSE DOCUMENTATION

This is the new GeoMOOSE documentation repository. We are starting with the new GeoMOOSE 2 documentation and will work to integrate all of the tutorials.

6.1 References and Write-Ups

6.1.1 GeoMOOSE Roadmap

The development team for GeoMOOSE put together for a roadmap for the next few foreseeable versions of GeoMOOSE. This list is subject to change depending upon available funding and interest of contributors. Here's what has been targeted for each version:

GeoMOOSE 2.0 Final Release (targeted for June, 2009)

This will primarily consist of bug fixes identified from the beta release. Features in this release include; performance enhancements, new configuration files, integration with OpenLayers library, zoom navigation tools, measure tools, layer controls (tip, max/min scale, opacity and legends), identify service, itemquery service, buffer service, select service, mailing labels service, print service, popup service, feature report service, ability to define your own tabs and projects and feature editor service using PostGIS. A unified demo combining the old state and county demos. Update website and implement trac for bug reporting and feature requests

GeoMOOSE 2.2 Release (targeted for Fall of 2009)

- geocoder service using google maps api (contributed)
- query builder service (partially funded)
- more documentation
- support for tilecache layers
- saving user state for layers and location
- animation for WMS-T layers
- support for Google maps layers
- ability to add markers to the map
- document the GeoMOOSE api
- add/remove map services

- KML output
- wrapping text of long fields in the feature report

GeoMOOSE 2.4 Release (no set target date)

New print service based on Cario library (need to wait for MapServer 6.0 release)

GeoMOOSE 3.0 Release (no set target date)

Possible integration with Ruby on Rails framework (need to discuss this further)

6.1.2 Mapbook Reference Guide

The mapbook document is an XML file that is used as the configuration file for a GeoMOOSE application. It configures things such as map sources, layers, services and tools in the application. For an example of a mapbook, please refer to the GeoMOOSE demo and the mapbook.xml file in the conf folder.

This reference guide explains the structure, elements, and attributes of the Mapbook XML tags, and how they effect the user interface. GeoMOOSE version 2.0 looks to extend the capabilities and ease of configuration from GeoMOOSE 1.x series and therefore a new version of the mapbook format was required to support all of those features.

<mapbook>

All version 2.0 mapbooks should express themselves by using the 'version' tag. GeoMOOSE will check the version attribute at startup and if the minimum version requirement is not met, GeoMOOSE will alert the user with an error. This makes the version attribute required.

The <mapbook> tag contains 6 child elements. Each of these child elements can have their own child elements and are described below.

- <configuration> : The <configuration> tag is used to specify user interface startup settings.
- <map-source> : The <map-source> tag is used to specify a single or collection of layers in GeoMOOSE.
- <layer-controls> : The <layer-control> tag is used to link a layer control to a service. As of GM2 the only layer control available are popups.
- <service> : The <service> tag is used to define a service such as select, printing and identify.
- <catalog> : The <catalog> tag is the layers listing found in the information panel.
- <toolbar> : The <toolbar> tag is the tools that are displayed on the user interface toolbar.

<configuration>

The configuration tag is used to specify user interface startup settings and other application settings. The configuration tag has <param> children. <param> elements have a simple format, a "name" attribute which specifies the interface setting to change and then CDATA that specifie the values. Here is an example from the GeoMOOSE Demo:

```
<configuration>
        <param name="mapserver_url">/cgi-bin/mapserv.exe</param>
        <param name="mapfile_root">/ms4w/apps/geomoose2/maps</param>
        <param name="projection">EPSG:26915</param>
        <param name="zoomto['Jump To:']"><![CDATA[
```

```
        {
                'Dakota County' : [521238.614537864,4924218.86673578,473921.947801381,4974430.3688503
                'Parcel Data' : [497205.409367,4923984.423582,477595.805945,4941970.52988],
                'Full State of MN' : [189783.560000,4816309.330000,761653.524114,5472346.500000]
        }
        ]]></param>
        <param name="max_extent">189783.560000,4816309.330000,761653.524114,5472346.500000</param>
        <param name="initial_extent">497205.409367,4923984.423582,477595.805945,4941970.52988</param>
        <param name="layer_controls.up.on">false</param>
        <param name="layer_controls.down.on">false</param>
        <param name="layer_controls.legend.on">false</param>
</configuration>
```

<map-source>

The map-source tag is used to specify a single or collection of layers in GeoMOOSE. All map-sources have two required attributes: type – The type attribute determines what other children are required and tells GeoMOOSE what type of mapping service with which to communicate. Valid types are:

- **mapserver** – This type of layer is meant to communicate with the default mapserver as specified in config.js.
  - <file> - Layers of type 'mapserver' require a value <file> child specifying the location of the mapfile on disk. Mapfiles can be specified with relative paths in <file> tags.
- **wms** – This type of layer is meant to communicate with Web Mapping Service servers. This is the OGC standard for serving raster images over the web. Many sites deliver WMS services that can be consumed by a GeoMOOSE application. If the type is a WMS data source an optional attribute can also be specified to request the images as tiles. This attribute is tiled=true.
  - <url> - Layers of type 'wms' require a <url> child specifying the URL of the WMS.
  - *name* – This is the name used internally, by GeoMOOSE to refer to the layer. Unlike GeoMOOSE 1.0, the title of the layer displayed in the Catalog is not a one-to-one relationship with the entry in the mapbook. This allows greater flexibility in divorcing the display-order of layers in the map and display order in the catalog.
  - *queryable* – This is an optional attribute the tells GeoMoose whether a WMS map-source type is queryable. This attribute is only need for WMS map-source types. Valid options are queryable='true/false'.

All map-source types support two children:

- **<param> – The 'param' child will add or change parameters sent over the URL to a <map-source>. This can be necessary**
  - *name* – This is the name that will be used in the URL.
  - *value* – This is the value the name will be set to in the URL.
  - <layer> (required) – At least one layer child is required for each map-source. Even if a mapserver-type layer is used where all the layers are set to 'default' in the mapfile, it is necessary to create a layer child representing all of the layers. Layer elements have one required attribute, name. For a WMS map-source, this refers to a layer as specified in the GetCapabilities document. For a Mapserver map-source, this refers to the name of a layer in the Mapfile. If a Mapfile has all of the layers set to default or all of the layers should be on or off as a whole, then the 'name' should be set to 'all.' <layer>s also have a "visible" attribute which defaults to false. To set a layer to default visibility, set "visible" to true.

Examples (more can be found in the GeoMOOSE demo mapbook):

```
<map-source path="county" type="mapserver">
          <file>./countydata/feedlots.map</file>
          <layer name="all"/>
</map-source>

<map-source name="basemap" type="mapserver">
        <file>./basemap/basemap.map</file>
        <layer name="cities"/>
        <layer name="counties"/>
        <layer name="roads"/>
        <layer name="lakes"/>
        <layer name="streams"/>
        <layer name="ditches"/>
        <layer name="TLGAreaLandmarks"/>
        <layer name="TLGLandmarkPoints"/>
</map-source>

<map-source path="metro" type="wms">
        <url>http://www.datafinder.org:80/wmsconnector/com.esri.wms.Esrimap/MN_MetroGIS_DataFinder_WM
        <layer name="stream_net_l"/>
        <param name="TRANSPARENT" value="TRUE"/>
        <param name="FORMAT" value="image/png"/>
</map-source>
```

`map-source``s also support an ``opacity` attribute that is a number between 0 and 1 specifying how transparent the image will be displayed. For example `opacity=".5"` will display the image from the map-source as half-transparent.

- `buffer` attribute. Boolean, defaults to true. When set to "false" the image does not "buffer" around the map, it is trimmed to the exact size of the map display. Useful for MapServer scalebars.

<service>

The service element is the key to GeoMOOSE's interoperability. All functions are a type of service (identify, select, Birdseye view, etc.). Services have attributes to define some of it's behaviour: * *name* - This is the internal name of the service. * *display* - Sets whether the the service will be displayed while "launching." It is useful to set this to false if there is no text-entry required for the service. * *title* - This title will be displayed automatically in the input tab. * *target* - If set then this attribute will display the results in a different window.

Service have one <*url*> child. The <*url*> child specifys the URL for of the service for GeoMOOSE to call:

```
<url>php/idnetify.php</url>
```

Services each have <*step*> children. There are two types of steps:

- ***type=spatial* - This type of step allows the layer to. Spatial steps have additional inputs that are used to define additional in**

  - *name* - This attribute specifys the CGI/form variable name in which the representation of the shape will be stored.
  - *default* - Sets the default tool used for drawing the shape. Valid values: point, line, polygon.
  - *point* | *line* | *polygon* | *edit-polygon* - Turns on/off the various drawing tools. Valid values: true, false. The default behaviour for these are stored in Configuration.
  - ***format* - This specifys the format of the shape:**

    * =*WKT* - Use well-known text.

        * =*delim* - Use a text delimited format.
    - *jump-start* - When true then the service will attempt to start after the shape has been finished.
    - *reproject* - This can be used to specify a shape projection other than the default map projection.
- *type=input* - This step only allows for the standard inputs.

Steps can have many <*input*> children. <*input*> s are very similar to the ones used in HTML forms. All <*input*> s require a *name* attribute and a *type* attribute. The following documents the various *type* s of inputs:

- *type=extent* - Sends the current extent to the service.
- *type=hidden* - Hidden inputs are not displayed to the user but the value set in the *value* attribute is sent to the service.
- *type=select* - Creates a drop down box. The *title* attribute sets the label for the drop down. These have the same <*option*> children as the HTML <*select*> elements.
- *type=sketches* - Sends JSON describing all of the user sketches to a service.
- *type=user* - These are basic user input text fields. Their default value can be set using the *value* attribute. The *title* attribute sets the label for the text field.
- *type=visiblelayers* - Sends the list of visible layer paths.
- *type=ajax_select* - Creates a drop down box populated by a remote server script. These inputs types feature a <*url*> child which references the script to call. The script should return HTML-style <*option*> elements, a sample return may look like this:

```
<options>
        <option value="a">A</option>
        <option value="b">B</option>
</options>
```

- *type=precision* - Sends the precision of the map to the service.

<layer-controls>

<catalog>

The GeoMOOSE Catalog, or Catalog, is the layers listing found in the table of contents on the user interface. The catalog is represented by the <catalog> child inside of the mapbook. The catalog has two types of children:

- layer – The layer child creates an entry in the catalog for the user to turn on and off layers. The layer child has two required attributes:
- show-legend - When set to "true" the legend will be displayed by default. CONFIGURATION.layer_controls.legend.on or the "legend" must be set to "true" for this to function.
- src – This is attribute tells GeoMOOSE where to find the source for the layer. The source of the layer is the 'name' attribute specified in the <map-source> plus a '/' combined with the name attribute of the layer as specified in the <map-source>'s <layer> child. In the examples above, there are two complete paths:

```
county/all
metro/stream_net_1
```

Multiple paths can also be specified to turn on/off multiple layers with a single control. To do this, the paths need to be divided with colons. If both layers in the example above were to be turned on simultaneously the 'path' attribute would be specified as follows:

`county/all:metro/stream_net_1`

- title – This is the text that will be shown in the Catalog to represent the layers specified in the path attribute.

Layers also have an optional attribute:

- status – Layers can be turned on by default by setting 'status' to 'on'. This defaults to 'off'.
- group – The 'group' child creates folders in the catalog. Groups contain layer children and can also contain other groups. Groups have two attributes, both of which are optional:
- expand – A group can be started as 'open' where the contents of the group are listed. The 'expand' attribute can either be 'true' or 'false'. This defaults to 'false'.
- multiple – While most layers are stacked, there may be a set of layers that should not be stacked on top of each other. The most common example is background layers. In the case of background layers, only one should be selected at a time. To do this in GeoMOOSE, set the group's 'multiple' attribute to 'false'. This defaults to 'true'.
- metadata – When set to 'true', the metadata tag will be included as described below. This defaults to 'false'.

Layers May Optionally Have related Metadata Via a "metadata" tag: * <metadata>URL</metadata> - The URL refers to the location of the metadata for this catalog item.

<toolbar>

GeoMOOSE now has a better defined toolbar. Just as the catalog and layer definitions have become more separated, so too has the toolbar and service definitions. The toolbar is specified using the <toolbar> child. The <toolbar> child has no attributes and two kind of child, the `<tool>` child and the `<drawer>` child.

`<tool>` children define the tool in the toolbar and tells geomoose what to do when the tool is clicked. <Tool> children have a number of attributes:

- name – The name is the name for the tool.
- title - This is the title displayed in the tool bar if the text is shown and in the mouseover popup.
- selectable - This is a boolean attribute. Examples of "selectable" tools: Identify, Select, Zoom. Examples of tools that are not usually "selectable": Print, Search Parcels.
- **type – This attribute determines whether the tool calls an internal function or a service as defined in a 'service' child. The**
  - type = 'service' adds an additional required attribute, 'service' which is the name of the service specified in the <service> tag.
  - type = 'javascript' has no additional required attributes but the node's value should contain javascript to be executed when the tool is selected.
  - **type = 'internal' adds an additional required attribute, 'action' which is the type of action to be taken current va**
    - zoomin – start the zoomin tool
    - zoomout – start the zoom-out tool
    - pan – start the pan tool
    - previous – step back to the previous extent
    - next – step forward to the next extent if jumped back using the 'previous' tool.
    - measure – start the measure length tool.

* measurearea – start the measure area tool.
* fullextent – zoom to the predefined initial extent of the map.
* draw_polygon - start the polygon drawing tool for the sketch layer.
* draw_line - start the line drawing tool for the sketch layer.
* draw_point - start the point drawing tool for the sketch layer.
* draw_remove - start the tool for removing shapes from teh sketch layer.
* draw_edit - start the sketch layer shape editing tool
* draw_edit_attributes - start the edit attribute tool. This is the tool that allows the user to change the shape colors and opacity.

<drawer> children contain tools and create a small "drop down menu" in the toolbar to compress the space required for related tools. For example the drawing tools can easily be combined into a <drawer>:

```
<drawer>
        <tool name="draw_polygon" title="Draw Polygon" type="internal" action="draw_polygon"/>
        <tool name="draw_line" title="Draw Line" type="internal" action="draw_line"/>
        <tool name="draw_point" title="Draw Point" type="internal" action="draw_point"/>
        <tool name="draw_remove" title="Remove Drawing" type="internal" action="draw_remove"/>
        <tool name="draw_edit" title="Edit Drawing" type="internal" action="draw_edit"/>
</drawer>
```

Specifying Icons for a Tool in the Toolbar

Every tool in the toolbar is given a unique ID that allows it to be referenced in CSS. The icons are all specified as backgrounds in the htdocs/css/user_tools.css file. Editing that will show examples of how to specify the icons for a new tool. This change was made to support the use of CSS sprites to increase the load time of GeoMOOSE.

Here is an example featuring the "Print" tool:

```
#tool-print {
        width: auto;
        background-position: 2 2;
        background-image: url('../images/toolbar/printer.png');
}

#tool-print .ToolContent {
        display: -moz-inline-box;
        display: inline-block;
        width: auto;
}

#tool-print .ToolText {
        display: block;
        padding-top: 3px;
        padding-right: 3px;

        display: -moz-inline-box;
        display: inline-block;
}
```

6.1.3 GeoMOOSE Configuration Options

Introduction

GeoMOOSE has a plethora of configuration options. All of them can be changed in the mapbook and it is possible to highly customize the interface without writing a single line of code. This document attempts to keep up with the various settings options in order to give the user more control over their GeoMOOSE installation. Organization is a little rocky as it mirrors the internal structure and nothing that was ever designed to be human readable. Please bear with us as the documentation gets updated.

How to Change a Setting

Every Mapbook has a <configuration> section. In the <configuration> section there are multiple <param> fields. The <param> fields are how settings are changed from their defaults. For example, to add some HTML to the Links bar:

```
<configuration>
        ...
        <param name="links_bar_html"><![CDATA[This is some html, with a <a href="http://www.google.co
        ...
</configuration>
```

The Settings

links_bar_html

HTML Text. Default is ‘’. Changes the HTML of the “links bar” formally the “menu bar” in the interface.

waiting_html

HTML Text. Default is ‘Loading...’. Changes the HTML of the waiting message while services are running.

mapserver_url

String. MapServer URL used when calling type=”mapserver” map-source’s.

mapbook

String. Default is ‘php/getmapbook.php’. The URL to the mapbook, this is the only configuration setting that cannot be changed by the mapbook.

fractional_zoom

Boolean. Default is false. This allows the Map to use fractional scales like GeoMOOSE 1.X. This functionality, however, breaks tiled layers.

scales

Array of Floating Point Numbers. The scales are actually ground units per pixel resolution settings and not true scales. This array is what sets the "scales" shown on the map.

max_extent

Array of Floating Point Numbers. An array, in the format of [MinX, MinY, MaxX, MaxY] that sets the maximum extents of the
<param name="max_extent">[-180,-90,180,90]</param>

initial_extent

Array of Floating Point Numbers. This sets the intial view for the map.

projection

String. The projection of the map, defaults to UTM-15N (EPSG: 26915). GeoMOOSE is primarily developed in the state of Minnesota, USA, so many of our test datasets are in UTM-15N. The default Projection is this way for no other reason than that.

ground_units

String. Default is 'm'. Set to 'm', 'in', 'mi', 'dd', to reflect the projection being used.

coordinate_display.xy

Boolean. Default is true. Toggles the display of the ground unit X,Y in the footer of the map.

coordinate_display.latlon

Boolean. Default is true. Toggles the display of the latitude and longitude in the footer of the map.

coordinate_display.usng

Boolean. Default is true. Toggles the display of the United States National Grid Coordinates in the footer of the map.

tabs

Associative Array. The keys of the associative array are the titles of the tabs and the values is the DIV that represent the tab. This would be an example setting that is different than the default:

```
<param name="tabs"><![CDATA[
{
        'Layers' : 'layers-tab',
        'Information' : info-tab'
}]]></param>
```

default_tab

String. Defaults to 'Catalog'. References the title of the tab that should be shown by default.

catalog_name

String. Defaults to 'Catalog'. Reference the title of the tab that should be populated with the layers list.

catalog.toggle_controls

Boolean. Defaults to true. When set to true, clicking on the layer name allows the controls below the layer to be turned on or off. When set to false the visibility of the controls is set by catalog.show_controls.

catalog.show_controls

Boolean. Defaults to true. Sets whether or not the controls below the layer names should be visible.

show_service_settings_in

String. Defaults to 'Services'. Reference the title of the tab that should display the prompts for a service when it is called.

group_checkboxes

Boolean. Defaults to true. Toggles whether group checkboxes are displayed.

zoomto

Associative Array of Associative Arrays. This is the settings that populate the "jump to" drop downs. Best to explain this by example. If you have difficulties please email the mailing list:

```
<param name="zoomto['Jump To:']"><![CDATA[
        {
                'Dakota County' : [521238.614537864,4924218.86673578,473921.947801381,4974430.368850
                'Parcel Data' : [497205.409367,4923984.423582,477595.805945,4941970.52988],
                'Full State of MN' : [189783.560000,4816309.330000,761653.524114,5472346.500000]
        }
]]></param>
```

jumpto_scales

Associative Array. This associative array features keys that are shown in the "scale drop down" and the scale itself is the value.

startup_service

String. When set, the named service will be called on startup.

layer_controls.up.on

Boolean. Defaults to true. Toggles the layer control to move the layer up on the image stack.

layer_controls.down.on

Boolean. Defaults to true. Toggles the layer control to move the layer down on the image stack.

layer_controls.fade.on

Boolean. Defaults to true. Toggles the layer control to fade the layer out.

layer_controls.unfade.on

Boolean. Defaults to true. Toggles the layer control to unfade the layer out.

layer_controls.refresh.on

Boolean. Defaults to false. Toggles the layer control to allow the user to manually refresh the layer.

layer_controls.cycle.on

Boolean. Defaults to false. Toggles whether the layer control to refresh the layer every layer_controls.cycle.seconds seconds.

layer_controls.cycle.seconds

Floating Point Number. Defaults to 10. The number of seconds between layer refreshes when used with the cycle tool.

layer_controls.legend.on

Boolean. Defaults to true. Toggles whether to show the legend control by default.

layer_controls.metadata.on

Boolean. Defaults to false. Toggles whether or not to show the link to a layer's metadata.

reference_map.enabled

Boolean. Defaults to true. Toggle whether the refrence map should be added to the map.

reference_map.maximized

Boolean. Defaults to true. Toggle whether the reference map is expanded by default.

reference_map.width

Integer. Width of the reference map.

reference_map.height

Integer. Height of the reference map.

reference_map.minimum_ratio

Integer. Minimum ratio of display between the main map and the reference map.

reference_map.maximum_ratio

Integer. Maximum ratio of display between the main map and the reference map.

scale_line.enabled

Boolean. Defaults to true. Toggle whether the scale line is shown on the may.

scale_line.top_units

String. Defaults to 'ft'. Possible unit values are 'm', 'in', 'km', 'ft', 'mi'.

scale_line.bottom_units

String. Defaults to 'mi'. Possible unit values are 'm', 'in', 'km', 'ft', 'mi'.

scale_line.width

Integer. Defaults to 200. Defines the width of the scale line on the map.

drawing_tools.default_fill

Color. Defaults to 'green'. Sets the default fill for all sketches.

drawing_tools.default_stroke

Color. Defaults to 'red'. Sets the default stroke for all sketches.

drawing_tools.default_opacity

Floating Point Number between 0 and 1. Defaults to .8. Sets the opacity of the sketches.

messages.requirement_failure

String. Message displayed when a service requirement was not met.

messages.invalid_tool

String. Displayed when a drawing tool that is not defined is called. The string *%TOOL%* will be replaced with the name of the offending tool.

messages.service_config_error

String. Is displayed when there is an error with the service being called.

messages.service_not_found

String. Message displayed when a service is called but is not defined in the mapbook. The string *%SERVICE%* will be replaced with the name of the invalid service.

messages.invalid_response

String. Messaged displayed when a service does not return a valid answer from the server.

messages.mapbook_version

String. Error presented to the user if they try to load a mapbook that is not compliant with the current GeoMOOSE version. The string *%VERSION%* will be replaced with the mapbook's version.

messages.mapbook_required

String. The error shown when a required <param> setting is missing in the mapbook. The string *%FIELD%* will be replaced with the required field's name.

messages.mapbook_param_error

String. When a param cannot be properly set due to any sort of error, this error message will be displayed. *%FIELD%* will be replaced with the "name" attribute from the param.

popups.autosize

Boolean. Sets whether or not popups are autosized when added to the map.

6.1.4 Changing the Projection

GeoMOOSE 2 uses the Proj4js library to perform all of its projection work. This is much more robust than the older GeoMOOSE reprojection library but requires a much different setup to include a different projection within the application.

Many applications may use a well-known projection. Most of these projections are assigned an EPSG code.

Step 1: Determine your projection.

This may have already been done for you. Having a predetermined projection makes this step quick and easy. However, some projections are not found in the standard collection of projections and you will have to define your own. Proj4js uses the same projection parameter style as the Proj4 C-library. If you need to make a custom projection, create one using an EPSG code of greater than 100000. For example, I may have a projection called "Duck's Projection." If I were to use "Duck's Projection" I would need to define it using proj params (these are form a randomly select filter):

```
+proj=tmerc +lat_0=38 +lon_0=125 +k=1 +x_0=200000 +y_0=500000 +ellps=bessel +units=m +no_defs
```

Step 2: Create a Projection Javascript File.

You can create this file in the htdocs directory but it should be named "EPSGXXXXX.js" where "XXXXX" is the number for your projection. If this is a custom projection give it a number above 100000. In side of this file you should have:

```
Proj4js.defs["EPSG:XXXXX"] = "+proj=tmerc +lat_0=38 +lon_0=125 +k=1 +x_0=200000 +y_0=500000 +ellps=be

Again, substitute your own EPSG number and projection information above.
```

Step 3: Change the mapserver parameter for your projection.

In the <configuration> section of the mapbook book you will need to change/create the following line:

```
<param name="projection">EPSG:XXXXX</param>
```

Step 4: Include the Javascript File

Find the section in the <head> of geomoose.html that states, "Include your projection here" and place some code:

```
<script type="text/javascript" src="EPSGXXXXX.js"></script>
```

For more information on finding projections and projection definition strings visit http://www.spatialreference.org.

6.1.5 Settings.ini Reference Guide

The settings.ini file found in the conf folder is a text document used as the general configuration file for a GeoMOOSE application. It configures things such as projection, paths, mailing labels. For an example of a settings.ini file, please refer to the GeoMOOSE demo and the file in the conf folder. This reference guide explains the optional settings.

[defaults]

- mapbook – location of default mapbook

[map]

- projection – EPSG code of coordinate system for the map

[paths]

- root – root for the mapfiles
- temp – web accessible location of the temp files relative to the root of the domain

[identify]

This setting specifies which metadata parameters in the mapfiles are used for the identify service output templates.

- identify_header=identify/header.html
- identify_footer=identify/footer.html
- wms_header=identify/wms_header.html
- wms_record=identify/wms_record.html
- wms_footer=identify/wms_footer.html

[select]

- highlight_map – this setting specifies which mapfile should be used to symbolize the selected or highlight feature on the map

[itemquery]

This setting specifies which metadata parameters in the mapfiles are used for the itemquery service output templates.

- itemquery_header=itemquery/header.html
- itemquery_footer=itemquery/footer.html
- itemquery_miss=itemquery/miss.html

[mailing_labels]

These setting specify formatting options for the labels generated from the mailing labels service. Specifies how many rows/columns to print per page:

- label_rows=10
- label_columns=3

Specifies page layout information (only applies to PDF Format):

- label_origin_x=.25
- label_origin_y=.5
- label_width=2.5
- label_height=1

Specifies Font for the label output (only applies to PDF Format):

- label_font=Arial
- label_font_size=8

Specifies the settings for the actual lines of the label. The items in parentheses relate to the data source in the mailing labels service. Items must be enter with parentheses and case sensitive field name.

- label_lines=3
- label_line_1=%OWNER_NAME%
- label_line_2=%BLDG_NUM% %STREETNAME% %SUFFIX_DIR% %STREETTYPE%
- label_line_3=%CITY%, MN %ZIP%

If this is set to "true" then any blank lines in the labels will be "collapsed" in the PDF output.

- label_lines_collapse=true

6.1.6 Service Communication Protocol Reference

GeoMOOSE service's 'talk back' to GeoMOOSE using either pure HTML or by using the GeoMOOSE 2.0 Service XML format. If a service returns content of type text/html then GeoMOOSE will simply display such content in the 'Results' tab. This type of return has very limited potential as it does not provide a way to control functions in the GeoMOOSE interface. In GeoMOOSE 1.0 we addressed this by using an alternate GeoMOOSE namespace embeded in the HTML. While this was a very technical and generally 'neat' solution it was very difficult to explain and XML was quite under powered as a way to allow services to properly manipulate the user interface. GeoMOOSE 2.0 fixes this by using a new XML format that allows the services to specify Javascript to be run upon loading and HTML to display in the interface. This communication protocol is useful for developers writing their own services.

<results>

The document element is mainly ignored for now. But it has three children:

- <script>
- <popup>
- <html>

<script>

Contains a single CDATA section that contains Javascript to be executed upon loading the interface. Javascripts in the <script> tag SHOULD NOT call methods or depend on attributes in GeoMOOSE objects. GeoMOOSE changes all of the time and calling specific methods (like those in the Catalog) may change calling format or even names. There is a limited set of functions that will be supported and whose API will not change without significant notice to users and developers. Those are contained in the single "GeoMOOSE" object. Other examples using this protocol can be found in the layer query templates such as /maps/landrecords/parcels_popup.html in the GeoMOOSE demo.

<popup>

Popups contain a single CDATA section that contians HTML to be displayed in a popup. Popups have four attributes:

- x – The "east/west" position of the popup in ground units.
- y - The "north/south" potion of the popup in ground units.
- width – The pixel width of the popup.

- height – The pixel height of the popup.

Example:

```
<popup x="-93.129" y="42.111" w="200" h="200">
        This is a cool popup!
</popup>
```

<html>

Contains a single CDATA section that contains HTML to be displayed in the Results Tab.

Example:

```
Content-type: application/xml
<results>
        <script><![CDATA[
                GeoMOOSE.refreshLayers('/wms/metro/stream_net_1');
                alert('I also made an alert!');
        ]]></script>

        <html><![CDATA[
                I refreshed a layer!
        ]]></html>
</results>
```

6.1.7 GeoMOOSE Feature Editor

What is the GeoMOOSE Feature Editor?

The GeoMOOSE Feature Editor allows the user to create, modify, and delete the geometry and attributes of vector features stored in a PostGIS database. The feature layers can be points, lines, or polygons. You can have as many layers set up for editing as you like, each with a separate table.

Setting Up the Feature Editor

To setup a layer for editing in GeoMOOSE it requires configuring ahead of time. First, the layer needs to be stored in a PostGIS database. Other relational database systems may be supported in the future. Shapefiles are not editable as they cannot be properly shared for editing amongst multiple users. Second, the layer's table in PostGIS needs to have a primary key, a sequence for generating keys, and have an entry in the PostGIS geometry_columns table. Finally, the layer must be configured for the GeoMosse Feature Editor by creating an "ini" for the editor script, adding the proper service entries to the mapbook, and finally configuring a Mapserver identify record template.

Setting up the Layer's PostGIS Table for Editing

Teh layer's table requires a primary key in the table, a sequence that works as the iterator for that primary key, and a geometry column. You may add other columns as well. The SQL to create an example would be as follows:

To create the table, the CONSTRAINT .. PRIMARY KEY directive specifies the primary key (in this example, polygon_id) for the table:

```
CREATE TABLE test_polygons
(
  polygon_id integer NOT NULL,
  title character varying(100),
  owner character varying(100),
  wkb_geometry geometry,
  CONSTRAINT pk_test_polygons PRIMARY KEY (polygon_id)
)
```

To create the sequence:

```
CREATE SEQUENCE test_polygons_seq;
```

Now add the table to the PostGIS register of geometry_columns. This may vary depending on your dataset, but for the purposes of the example this is what was used:

```
insert into geometry_columns values ('', 'public', 'test_polygons', 'wkb_geometry', 2, -1, 'GEOMETRY'
```

Create a Mapfile for the Layer

There are two files you must create for each editable layer: a mapfile and a .ini file. It is highly recommended keeping these named the same as the PostGIS table to prevent confusion. In this example the mapfile name would be "test_polygons.map". To keep things organized you may wish to put the files for each layer into their own folder. For example, create a "test_polygons" directory and then place "test_polygons.map" and "identify.html" (see details below) in the "test_polygons" directory. A sample mapfile:

```
MAP
        SIZE 400 400
        EXTENT  427632.500000 4893613.330000 560300.922104 5015936.680000
        UNITS METERS
        SYMBOLSET '../symbols/symbol.sym'
        STATUS ON

        IMAGETYPE GIF

        TRANSPARENT TRUE

        LAYER
                NAME 'polygons'
                TYPE POLYGON
                STATUS ON

                CONNECTIONTYPE POSTGIS
                CONNECTION 'host=192.168.52.1 dbname=geomoose_test user=postgres password=postgr
                DATA 'wkb_geometry from (select polygon_id, title, owner, astext(wkb_geometry) a

                CLASS
                        STYLE
                                SYMBOL 'circle'
                                OUTLINECOLOR 255 255 0
                                SIZE 10
                        END
                END

                METADATA
                        'identify_record' 'identify.html'
```

```
                END
            END
    END
```

Adding the Layer to the Interface

The first step is to create map-source entry in the GeoMOOSE mapbook that points to the test_polygons mapfile:

```
<map-source name="poly_editor" type="mapserver">
        <file>/var/www/geomoose2/maps/test_polygon/test_polygon.map</file>
        <layer name="polygons"/>
</map-source>
```

Then add a Layer entry to the catalog:

```
<group title="Editing Layers" expand="true">
        <layer title="Polygons" src="poly_editor/polygons"/>
</group>
```

Setting up the Layer's .INI file

The .INI file is where all the vital information about the layer's table is stored for the Feature Editor. All of the .INI files are stored in GeoMOOSE's "conf/editor" folder. To continue with the example from earlier, create a "test_polygons.ini" file. Since each table has a separate host/dbname configuration it is possible for a single GeoMOOSE interface to edit layers on multiple different servers without the user knowing the difference.

Example INI file:

```
; Basic Connection Information
host=192.168.52.1
dbname=geomoose_test
username=postgres
password=postgres22

; Table Information
tablename=test_polygons                          ; Database table name
geometry_column=wkb_geometry                     ; Column name from the CREATE TABLE SQL
primary_key_column=polygon_id                    ; Primary Key Column from CREATE TABLE
primary_key_sequence=test_polygons_seq           ; Sequence name from CREATE SEQUENCE
srid=26915                                       ; SRID for the Dataset in the database

; GeoMOOSE Information
layerpath=poly_editor/polygons                   ; The full path in GeoMOOSE (same as catalog)
```

Setting up a Create/Add Service

This may be the most difficult part of setting up the service as it requires the most amount of knowledge on how GeoMOOSE services are configured. But all tables should follow roughly the same pattern. All of the columns in the layer's table are referred to as "feature:[COLUMN NAME]" so if the table as a column named "owner" it is referenced as "feature:owner" for the name to be used with the editor script.

The service definition starts simple with the header:: <service name="add_polygon">

And add a reference to the editor url:

```
<url>php/editor.php</url>
```

Now, we need a step to tell GeoMOOSE to draw a polygon:

```
<step type='spatial' name='feature:wkb_geometry' line='false' polygon='true' point='false' default='p
</step>
```

You can see, above, that the "name" references the name of the geometry column. We also turn off all of the tools except the polygon drawing tool by specifying "false" values for the "line" and "polygon" attributes. This would change depending on the dataset (as some layers have a geometry type of points and others lines). The "default" attribute tells GeoMOOSE to start this step with the "polygon" tool enabled instead of Navigation or other tools.

Then there is a required step to describe which operation we're performing and the attributes that will be edited:

```
<step type="input"> <!-- This tells GeoMOOSE there will be user input -->
        <input type="hidden" name="op" value="create"/>
        <!-- The "op" variable tells the editor which operation to perform: create, update, or delete
        <input type="hidden" name="table" value="test_polygons"/>
        <!-- The "table" refers to the name of the "ini" file.  -->
        <input type="user" name="feature:title" title="Title:"/>
        <input type="user" name="feature:owner" title="Owner:"/>
        <!-- These previous two lines refer to columns in the table. -->
</step>
```

Putting it all together:

```
<service name="add_polygon">
        <url>php/editor.php</url>
        <step type='spatial' name='feature:wkb_geometry' line='false' polygon='true' point='false' de
        </step>
        <step type="input">
                <input type="hidden" name="op" value="create"/>
                <input type="hidden" name="table" value="test_polygons"/>
                <input type="user" name="feature:title" title="Title:"/>
                <input type="user" name="feature:owner" title="Owner:"/>
        </step>
</service>
```

Finally, the service needs to be exposed on the toolbar with an "Add Polygon" tool:

```
<toolbar>
        ...
        <tool name="add_polygon" title="Add Polygon" type="service" service="add_polygon"/>
        ...
</toolbar>
```

It's usually nice to add some CSS to decorate up the Polygon Tool, so something like this might be prudent in htdocs/css/user_tools.css:

```
#tool-add_polygon {
        background-image: url('../images/toolbar/add.png');
        background-position: 2 2;
        background-repeat: no-repeat;
        padding-left: 2px;
}

#tool-add_polygon .ToolText {
        padding: 2px;
        display: block;
        width: auto;
```

```
}

#tool-add_polygon .ToolContent {
        width: auto;
}
```

Creating the Update Service

While it is nice to create features, it may be necessay to edit those features. Modifying requires setting up a similar service to creating.

The entry for the Modify Polygon service will look like this:

```
<service name="modify_polygon">
        <url>php/editor.php</url>
        <step type='spatial' name='feature:wkb_geometry' line='false' polygon='true' point='false' de
                <input type="hidden" name="op" value="update"/>
                <!-- This just changes the option to 'update' instead of 'create' -->
                <input type="hidden" name="table" value="test_polygons"/>
                <input type="hidden" name="feature:polygon_id"/>
                <input type="user" name="feature:title" title="Title:"/>
                <input type="user" name="feature:owner" title="Owner:"/>
        </step>
</service>
```

Creating the Delete Service

This is the easiest service to create as there are no user inputs or spatial steps to define:

```
<service name="delete_polygon">
        <url>php/editor.php</url>
        <step type='input'>
                <input type="hidden" name="op" value="delete"/>
                <input type="hidden" name="table" value="test_polygons"/>
                <input type="hidden" name="feature:polygon_id"/>
        </step>
</service>
```

Create the Identfy.html File

Now get the layer properly identifying in the interface.

identify.html:

```
<tr bgcolor="#AA55AA">
        <td colspan="2"><b>[title] ([polygon_id])</b></td>
</tr>
<tr>
        <td>Owner:</td><td>[owner]</td>
</tr>
<tr>
<td>
        <a href="javascript:GeoMOOSE.startService('modify_polygon', {'feature:polygon_id' : '[polygon
</td>
<td>
```

```
        <a href="javascript:if(confirm('Are you sure you want to delete this?')) { GeoMOOSE.startServ
</td>
</tr>
```

The key to the editing functionality is in the <a href="javascript...">> links. These links are calling the editing service definitions with the values "filled in" for the selected object from the identify result. Every different feature type (i.e table) requires changing this list of attributes to reflect the column names of the specific table.

6.1.8 GeoMOOSE Sprite Generator, Toolbar and Catalog Styles

To make GeoMOOSE more efficient all toolbar and layer controls have been combined into a single image. All of the tool images are changed by manipulating CSS settings.

The GeoMOOSE Sprite

A "sprite" is a combined image that contains many smaller "sub images." In older versions of GeoMOOSE every image was sto
To decrease this load we have created a method for creating a sprite and a CSS file that ties together the combined image and the differing tools.

Explaining the Sprite Layout

The sprite has two columns. The left column is the tool in its "normal" state and the right column contains the tool in its "selected" state. Each row represents a different tool.

Generating the Sprite

The sprite is generated with the "createSprite.py" script located in the "tools" directory. The "createSprite.py" script also generates the CSS. The CSS outputs to the standard output. The general operation for generating the sprite is as follows:

1. .cd tools
2. ./createSprite.py > ../htdocs/css/sprite.css
3. Open htdocs/images/all.png in GIMP
4. Save the PNG as a GIF for IE6 support. *If steps 3 and 4 are neglected IE users will see very strange sprites.*

To specify an image for a "selected" tool add an image in the images/toolbar directory with "-selected" included at the end of the name. For example, the popups tool has two images in the toolbar directory: "popups.png", "popups-selected.png". When createSprite.py runs it searches the directory for the "selected" images and places them in the right-column of the sprite as appropriate.

Toolbar Styling

The toolbar styling has two components:

1. The sprite imaging layout explained above.
2. .Tool and .selected.
   (a) .Tool – defines the basic CSS for laying out the tool.
   (b) .selected – defines additional information when the tool is selected. In the default installation this makes the toolbar tool background grey.

..note:: You will need to have Python installed on your computer to run this script. If you are running Windows, you probably need to get PIL library for Python. See http://www.pythonware.com/products/pil/

6.1.9 What happened to the Menubar?

For the most part it is gone. There is still a place for the menubar to exist, however, there is no longer an elaborate separate code base to make a javascript powered menu function. If you would like to do this, the ability is still there. We removed this functionality for a few reasons.

1. We were maintaining the code. The menubar.js and related configuration was maintained by the GeoMOOSE authors and like much of the Javascript code base, minor browser changes could effect the function and break it. Even worse, when it did break, there were very few people who noticed. This gave us an indication that the feature was under utilized.
2. A secondary configuration file was slow to load. To have two configuration files at startup was causing two Ajax requests to be made at startup. This noticibly slowed the loading time and would cause skews in how the interface rendered.

6.1.10 Using the Reference Map

The display in the reference map is done on a service-by-service basis. This document attempts to cover the implementation of various reference map tasks.

Reference Layers Display with the Main Map

This is the common GeoMOOSE reference map. Under this scenario a source is included in both the main map and the reference map. To do this simply add set the `reference` attribute to true in the `map-source`:

```
<map-source name="basemap" type="mapserver" reference="true">
        <file>./demo/statedata/basemap.map</file>
        <layer name="county_borders"/>
        <layer name="county_labels"/>
        <layer name="city_poly"/>
        <layer name="city_labels"/>
        <layer name="USGSGagingStations"/>
</map-source>
```

Now whenever one of these layers is made visible, it will also be made visible in the reference map. These layers would be controlled in catalog just as they would if they were not included in the reference map.

A Fixed Reference Map set of Layers

There are situations in which having an extremely dynamic reference map is wasteful. In this case, it's desirable to setup the reference map to have a fixed set of reference layers. To make a source display all of it's layers in the reference map by default the `reference` attribute must be set to true at both the source *and* layer level:

```
<map-source name="states" type="mapserver" reference="true">
        <file>./demo/statedata/statesUTM15N.map</file>
        <layer name="all" reference="true"/>
</map-source>
```

To prevent this layer from changing DO NOT include it in the catalog. If you would like to have this layer in the main map and fixed in the reference map you should create TWO `map-source` entries. One entry should be configured similar to above, for the reference map, and another should be configured with a different name and without any of the

`reference` attributes set to true. This second `map-source` will be used in the main map (and consequently in the catalog).

Mixing and Matching Layers

The two setups above involved only a single layer in their demonstrations. The Reference Map could have a mix of fixed and dynamic layers. The configurations above just show small examples of how the maps can be setup.

Starting up the Reference Map Collapsed

By default the reference map starts open, this can be changed with the "reference_map.maximized" parameter:

```
<param name="reference_map.maximized">false</param>
```

Getting Rid of the Reference Map

In the case that a reference map is undesirable:

```
<param name="reference_map.enabled">false</param>
```

6.1.11 GeoMOOSE User Extensions

GeoMOOSE User Extensions are the latest way to customize a GeoMOOSE installation. While it has been common practice to modify the root of the code to create various custom tools, that has left some installations stagnant and unable to smoothly upgrade or integrate with new GeoMOOSE releases when they have become available. To address these concerns, we have developed GeoMOOSE User Extensions. These extensions will never be clobbered by a GeoMOOSE upgrade. However, extensions written that access core library functions may break as the underlying library API's may change over time.

What is the purpose of User Extensions?

The purpose of user extensions is for the customizer of a site to, "go crazy." It allows for users to add any functionality they wish to the user interface without forcing their code to break away from the main code base.

But Duck, how do I make one?

Good Question. Here is a brief tutorial...

Step 1: Come up with an idea for a user extension.

Say you want a disclaimer to show at the beginning of the application. Let's just say that disclaimer is, "Hello, World!"

Step 2: Write the User Extension

User extensions are stored in the "extensions/" folder under "htdocs/". For this example we'll create a file called, "Disclaimer.js". In "Disclaimer.js" we need to add the basic outline of the class by inhereting from the base user extension class. The file should look like this:

```
DisclaimerExtension = new OpenLayers.Class(GeoMOOSE.UX.Extension, {
        CLASS_NAME : "DisclaimerExtension"
});
```

This sets up a blank extension that does nothing. Let's break down the code as to what's happening:

- *DisclaimerExtension* - This defines the Javascript name for your extension. You'll want to give your classes long descriptive names so they do not clobber other potential global variables. Those who are more advanced are probably asking "Why didn't you name space them?" and the answer is: "To make it easier on people who don't necessarily understand what 'namespacing' is."
- *new OpenLayers.Class* - This is the OpenLayers class constructor.
- *GeoMOOSE.UX.Extension* - The base GeoMOOSE User Extension class. By putting this class as the first parameter we're going to "inheret" all of the methods from the base class.
- *CLASS_NAME : 'DisclaimerExtension'* - This configures the class so it knows its own name.

Now we need to have the class actually do something. The initial function that is called when an extension is loaded is called "load." So, now we define the load method andh have it display our disclaimer using "alert."

```
DisclaimerExtension = new OpenLayers.Class(GeoMOOSE.UX.Extension, {
        load: function() {
                alert("Hello, World!");
        },

        CLASS_NAME: "DisclaimerExtension"
});
```

Finally, we need to tell GeoMOOSE to load the extension at startup by adding this at the end of the file:

```
GeoMOOSE.UX.register('DisclaimerExtension');
```

The completed "Disclaimer.js" should look like this:

```
DisclaimerExtension = new OpenLayers.Class(GeoMOOSE.UX.Extension, {
        load: function() {
                alert("Hello, World!");
        },

        CLASS_NAME: "DisclaimerExtension"
});

GeoMOOSE.UX.register('DisclaimerExtension');
```

Step 3: Add the Extension to geomoose.html

- Open "geomoose.html" in a text editor.
- Find "</head>"
- On the directly above "</head>" add:

```
<script type="text/javascript" src="extensions/Disclaimer.js"></script>
```

Step 4: Enjoy!

Now you should see "Hello, World!" when the application starts.

Other Examples

An example of an User Extension which updates the interface on a periodic basis can be found in the "extensions/" directory with GeoMOOSE 2.2, the file is called "ColorChanger.js".

6.1.12 UNIX Installation Guide

Introduction

Pre-Requisites

GeoMOOSE requires a pretty full feature MapServer build. The following options are used by the development team on UN*X based systems to build MapServer:

```
--with-php=[path to php includes]
--with-agg=[path to agg libs]
--with-gd=[path to gd libs]
--with-wfs
--with-wmsclient
--with-wfsclient
--with-postgis
--with-proj=[path to proj]
--with-geos=[path to geos-config]/geos-config
--with-freetype=[path to freetype]
--with-gdal=[path to gdal-config]/gdal-config
--with-ogr
```

MapServer's PHP/Mapscript must also be installed. Please see the MapServer website for additional details on how to install and compile MapServer. You will also find information about the MapServer mailing lists which are very helpful for trouble shooting and additional instructions.

Current development efforts are against MapServer versions 5.2 and 5.4.

VERY IMPORTANT NOTE: Some Linux distributions come pre-packaged with a version of MapServer. This version of MapServer can be extremely out-dated and can be missing critical functionality (AGG rendering for example).

Installation

If the Pre-req's have been met then the hard part is actually over. Extract the GeoMOOSE .tar.gz file to a conveinent directory:

```
tar -xzvf geomoose-[version].tar.gz
```

Enter the directory and run configure:

```
cd geomoose-[version]
./configure
```

There are a few important configuration options to keep in mind before moving on:

```
--prefix=DIR    Installation directory (defaults to /opt/geomoose2)
--with-temp-directory=DIR       Temporary directory for script output (/tmp/www)
--with-url-path=DIR     URL Path, the root web directory for the application (/geomoose2/)
--with-projection=SRS   SRS string defining the map's projection (default EPSG:26915)
--with-mapfile-root=PATH        Path to the mapfiles root (default PREFIX/maps)
--with-mapserver-url=URL        URL to mapserver (default /cgi-bin/mapserv)
--with-server-name=SERVERNAME   Self referential name of the server (defaults to localhost)
```

The most important of these is –prefix, –with-temp-directory, and –with-url-path as they define where GeoMOOSE will be installed, GeoMOOSE's critical temporary directory, and the URL from which GeoMOOSE will be served. The defaults have been tested to work but they are not appropriate for all installations.

Next we need to actually install the files:

```
make install
```

This will move all of the files to the appropriate directory and write an apache configuration file that matches your parameters a

1. **Add an "Include" directive to httpd.conf::** Include "/opt/geomoose2/geomoose2_httpd.conf"
2. **Add a symbolic link into a http.d directory. This varies greatly by system, this is just an example.::**
   ln -s /opt/geomoose2/geomoose2_httpd.conf /etc/apache/http.d/other/geomoose2_httpd.conf

6.1.13 MS4W Installation Guide

VERY IMPORTANT NOTE There is no direct upgrade path from GeoMOOSE 1.x to GeoMOOSE 2.2, but the mapbook and map files from your GeoMOOSE 1.x installation are very useful when configuring GeoMOOSE 2.2. If you are upgrading your GeoMOOSE installation and have made configuration changes or added your own data to the maps folder, you should backup those files before proceeding. Once you have your MS4W folders backed up, you should delete these files and folders before proceeding.

Introduction

GeoMOOSE is a client side framework that is developed for use with MapServer (http://mapserver.gis.umn.edu). While GeoMOOSE does not require MapServer to be installed when setting up a basic map viewer, many of the services included with GeoMOOSE (identify, query, search, etc.) do require MapServer. Most people will find it worthwhile to install MapServer and if you are going to install GeoMOOSE on a Windows-based computer, it is highly recommended that you use the MS4W package (MapServer for Windows). If you are installing GeoMOOSE on a Linux computer, see instructions at (http://geomoose.org/docs/install_unix.html).

In an effort to make it as simple as possible to get MapServer installed, we have packaged GeoMOOSE in an MapServer for Windows (MS4W) package. This document provides the steps necessary to download and install all the components required for GeoMOOSE.

IMPORTANT NOTE You need to have administrator level access to your computer to install and configure either Apache or IIS Server.

NOTE If you are new to web servers and developing web sites we STRONGLY recommend using Apache as your web server.

Step 1: MS4W Installation

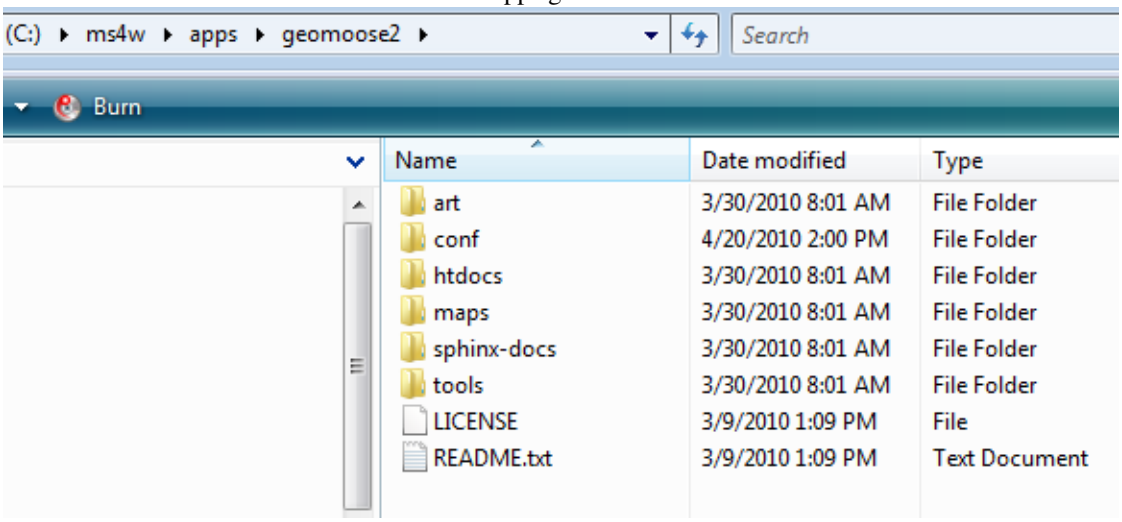
IMPORTANT NOTE There are compatibility issues with MS4W 3.0 beta 10 and GeoMOOSE, we recommend that you use MS4W 2.3.1. Developers are working to make GeoMOOSE compatible with the newest release of MS4W.

- The MS4W download will install Apache, MapServer and PHP. Download the MS4W package from the Maptools web site at http://www.maptools.org/ms4w.
- There is not a traditional "install" program for MS4W, rather you are going to unzip the contents to the root of your C: drive. Actually, MS4W does not have to be installed on the C drive, but it is good practice to have the MS4W directory at the root of the drive (not in "Program Files"). You should not have multiple MS4W folders.

- Once you have the MS4W package unzipped on your computer, you will need to decide which web server to use. The two most common choices are Apache and Microsoft IIS. Unless you need the integrated Windows authentication IIS provides or your organization requires the use of IIS, we recommend using the Apache web server that is included with MS4W.

Step 2: GeoMOOSE Installation

- Go to the GeoMOOSE download page (http://geomoose.org/info/download.html). There are two versions of GeoMOOSE for MS4W. "GeoMOOSE 2.2 for MS4W - Web Mercator Demo" is pre-configured with the setting necessary for Google, Yahoo, bing, and OSM base maps; if you want to use one of these base maps, your map will have to be in the web Mercator projection. If you are not going to use one of these base maps, we recommend using the other "GeoMOOSE 2.2 for MS4W" download.
- The GeoMOOSE installation is similar to MS4W in that there is not a traditional "install" program. The only step is to unzip the files to the C:/MS4W directory. When finished, the GeoMOOSE installation should be in C:/ms4w/apps/geomoose2



Step 3a: Configuring Apache web server

NOTE Skip this step if you going to install IIS server in Step 3b.

- Apache comes pre-configured with MS4W and as long as it is installed at the root of the drive (i.e. "C:/ms4w"), all that is needed to complete the process is installing Apache as a Windows service. From Windows Explorer (or a command prompt), double-click on `C:/ms4w/apache-install.bat`. A command window will appear for a couple seconds and then disappear.
- To test if the installation is working, from a browser on the computer you installed MS4W and GeoMOOSE, type in the address `http://localhost`. Your screen should display information about MS4W and the applications installed like the image below.

MS4W - MapServer 4 Windows - version 2.3.1

Introduction

Welcome to MS4W v2.3.1, the MapServer package for Windows. This package is intended to simplify your life if yc

- a novice MapServer user with little knowledge (or time) to compile MapServer from scratch.
- interested in one of the MS4W addon packages and looking to avoid configuration issues.
- any MapServer user who needs a quick and dirty installation on Windows.

In any of these cases, and probably more, MS4W can help you. However, a few notes about this package:

- This is not a true installer, it will not modify any file or registry setting outside of the MS4W directory.
- Since it is not a true installer, it will not install to any location. It ****** MUST ****** be installed at the root of one
- The MS4W addon packages must be extracted at the same root as the base installer, and files must be o

Features

MS4W contains default installations of Apache, PHP, MapServ, MapScript (CSharp, Java, PHP, Python) and some messing up the rest of the install. The base installer package comes pre-configured with the following software:

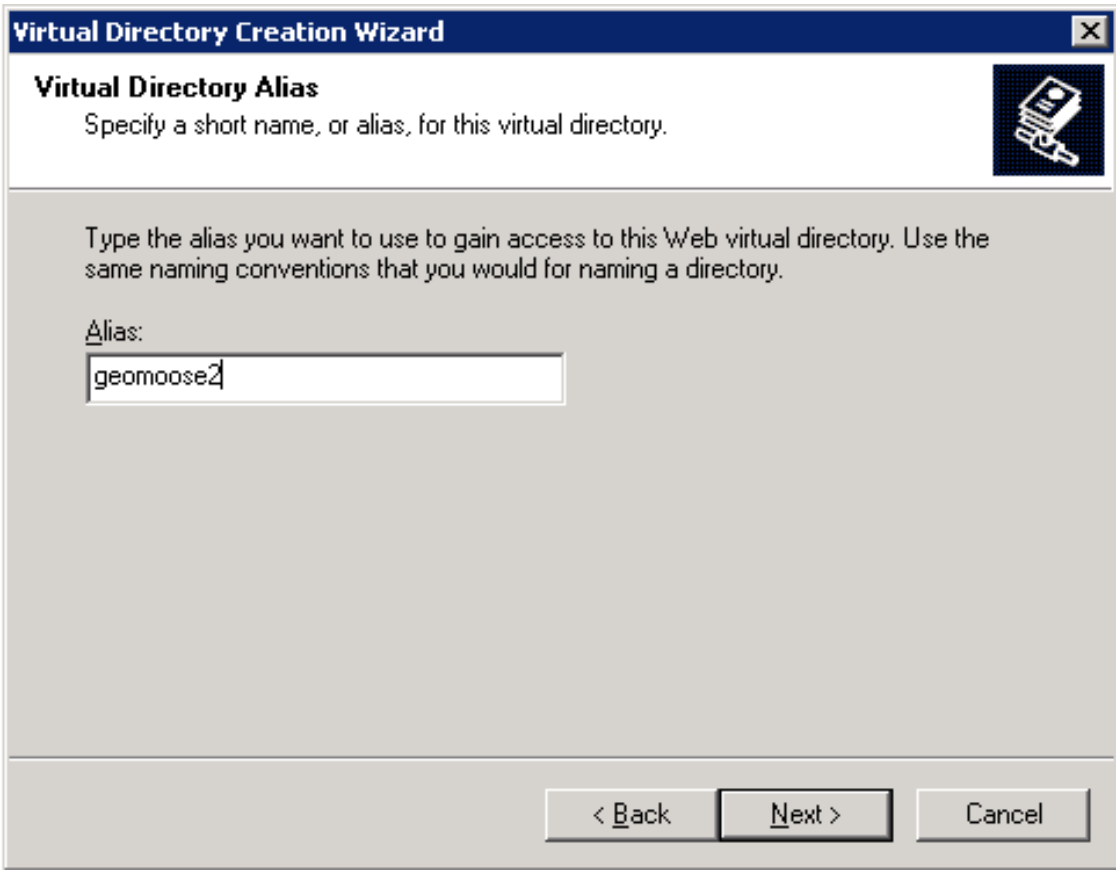
- Apache 2.2.10 (with OpenSSL 0.9.8i)
- PHP 5.2.6
- mapserv 5.2.1 at /cgi-bin/mapserv.exe
- mapscript 5.2.1 flavours:
  - php_mapscript.dll (phpinfo)
  - CSharp mapscript at *\ms4w\Apache\cgi-bin\mapscript\csharp*
  - Java mapscript at *\ms4w\Apache\cgi-bin\mapscript\java*
  - Python mapscript at *\ms4w\Apache\cgi-bin\mapscript\python*
- mapserv utils at *\ms4w\tools\mapserv* which include:
  - legend.exe MapServer utility documentation
  - msencrypt.exe
  - scalebar.exe
  - shp2img.exe
  - shptree.exe (execute **\ms4w\setenv.bat** before using mapserv utilities)
  - shptreevis.exe
  - sortshp.exe
  - tile4ms.exe

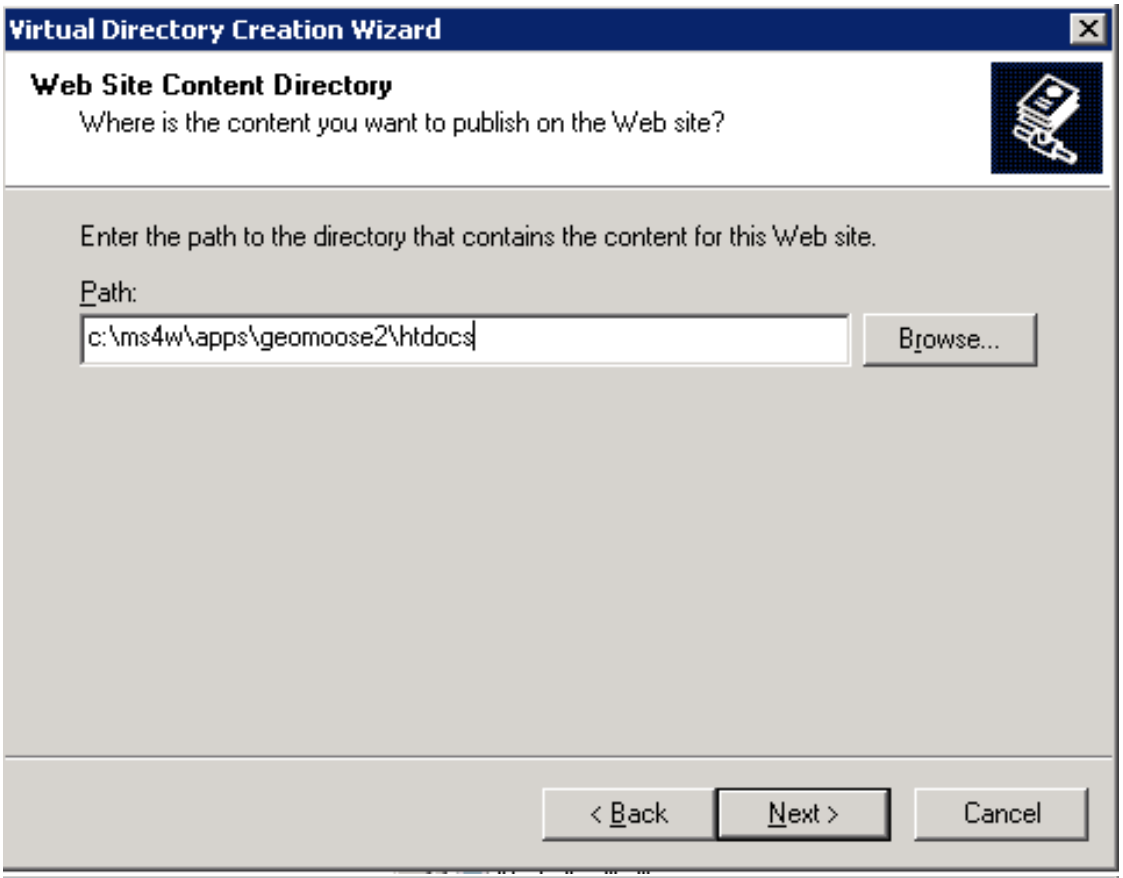
- At the bottom of this page is a list of applications installed which use MS4W. Click on the GeoMOOSE 2.2 link to test the installation.
- Further help and troubleshooting information is provided in the MS4W documentation (http://www.maptools.org/ms4w/index.phtml?page=README_INSTALL.html).

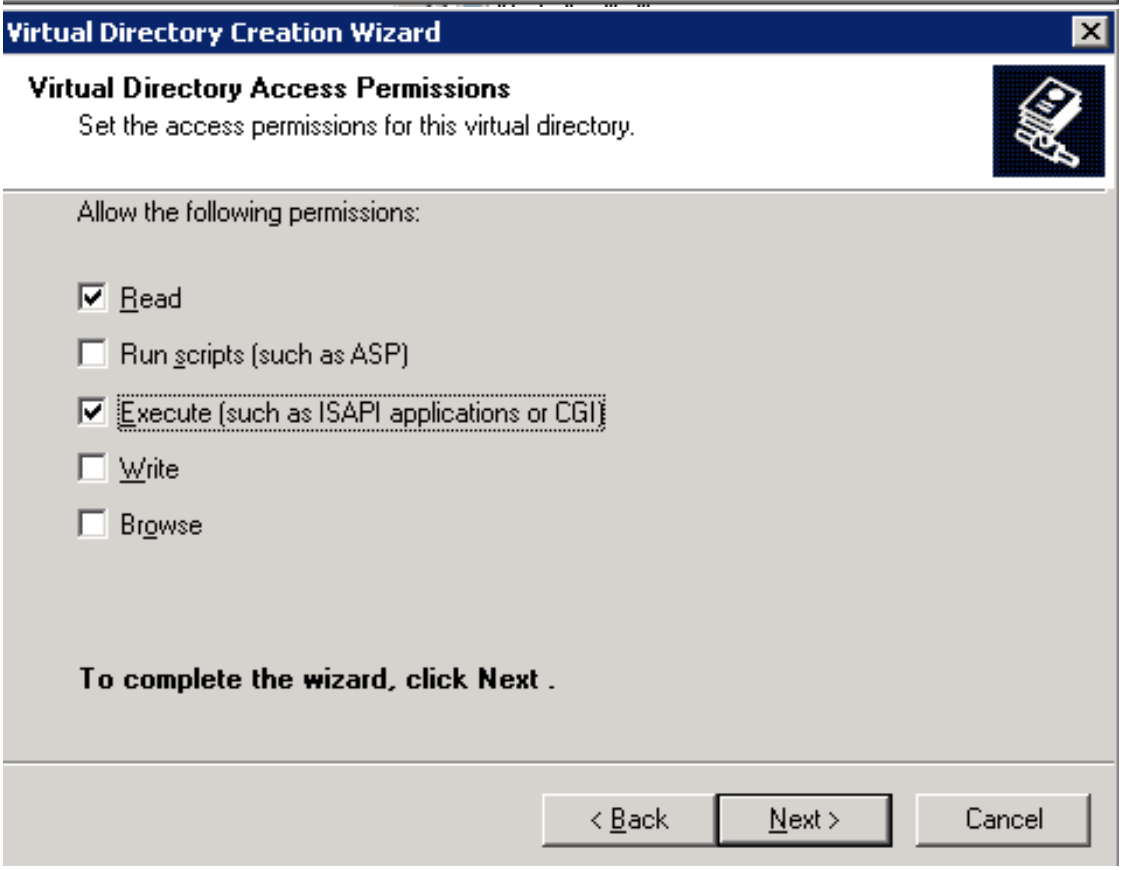
Step 3b: Configuring Microsoft IIS 6.0 web server

NOTE Skip this step if you installed Apache server in Step 3a. **NOTE** These instructions are for IIS 6.0 which is part of Windows Server 2003 and Windows XP Professional 64 bit. The steps are similar for other versions of IIS, however the interface can be very different and will not match screen shots provided here.

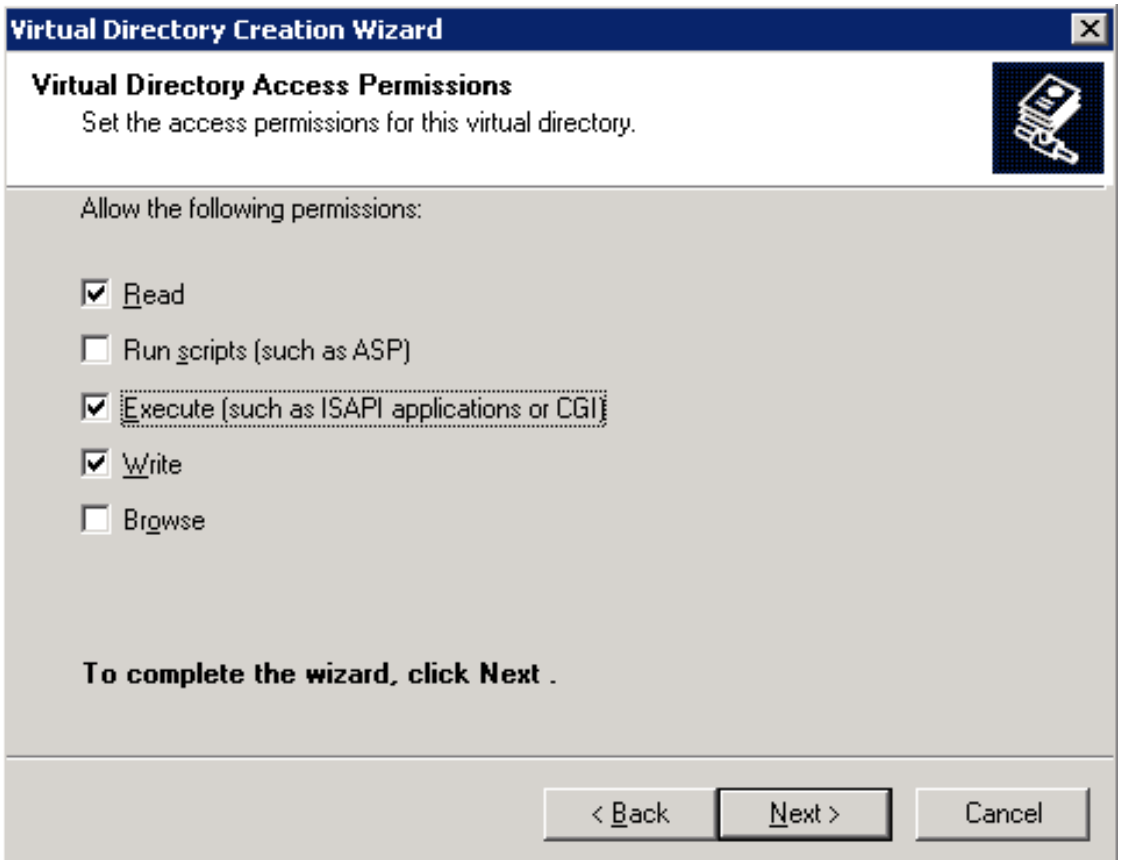
- Open the "IIS Manager".
- Create a virtual directory called "geomoose2" and point it to the path "C:/ms4w/apps/geomoose2/htdocs". See screen shots below for an example of adding a virtual directory.


Virtual Directory Creation Wizard
Virtual Directory Alias
Specify a short name, or alias, for this virtual directory.
Type the alias you want to use to gain access to this Web virtual directory. Use the same naming conventions that you would for naming a directory.
Alias:
geomoose2
< Back
Next >
Cancel


Virtual Directory Creation Wizard
Web Site Content Directory
Where is the content you want to publish on the Web site?
Enter the path to the directory that contains the content for this Web site.
Path:
c:\ms4w\apps\geomoose2\htdocs
Browse...
< Back
Next >
Cancel

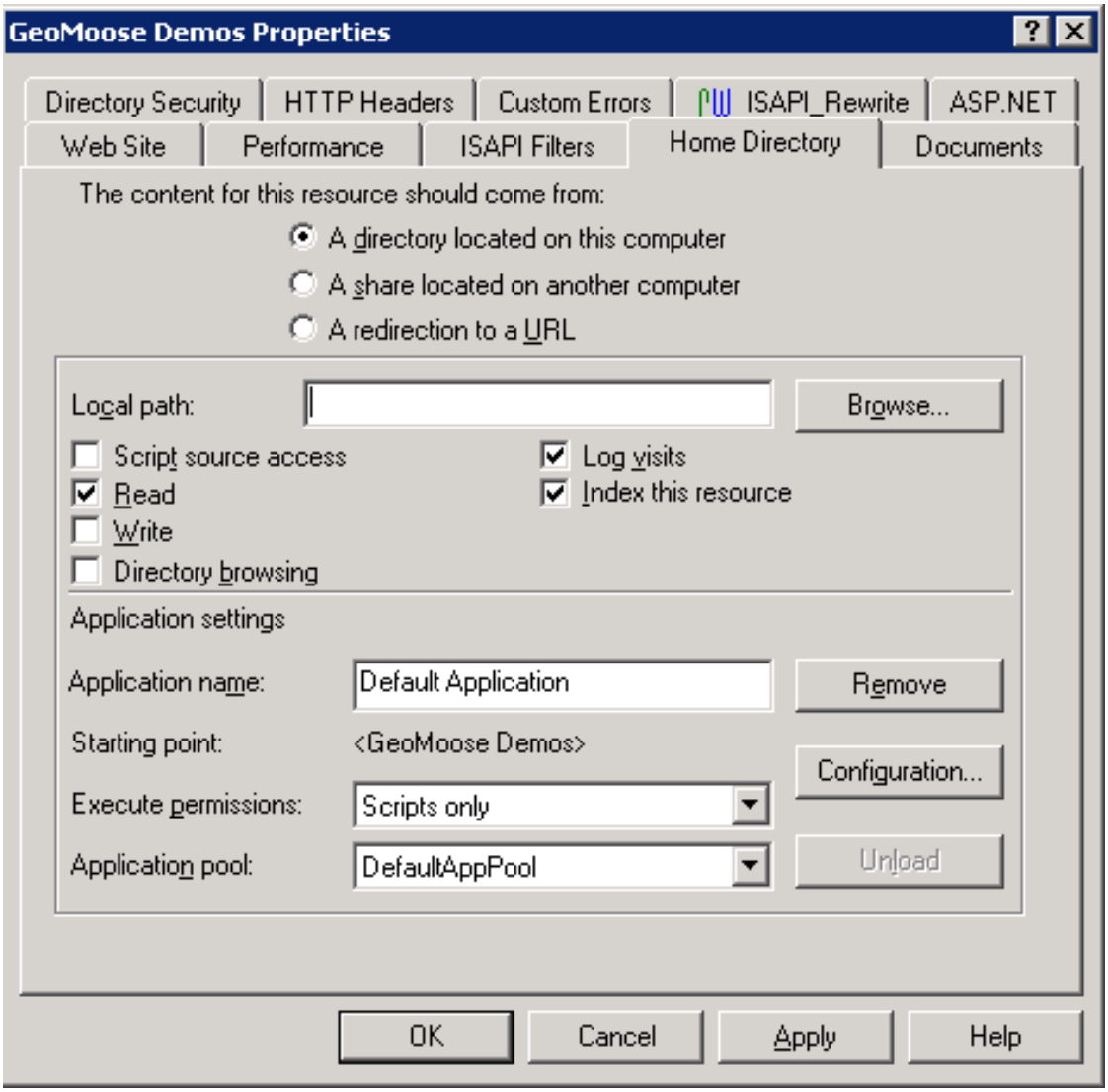

Virtual Directory Creation Wizard
Virtual Directory Access Permissions
Set the access permissions for this virtual directory.
Allow the following permissions:
Read
Run scripts (such as ASP)
Execute (such as ISAPI applications or CGI)
Write
Browse
To complete the wizard, click Next .
< Back
Next >
Cancel

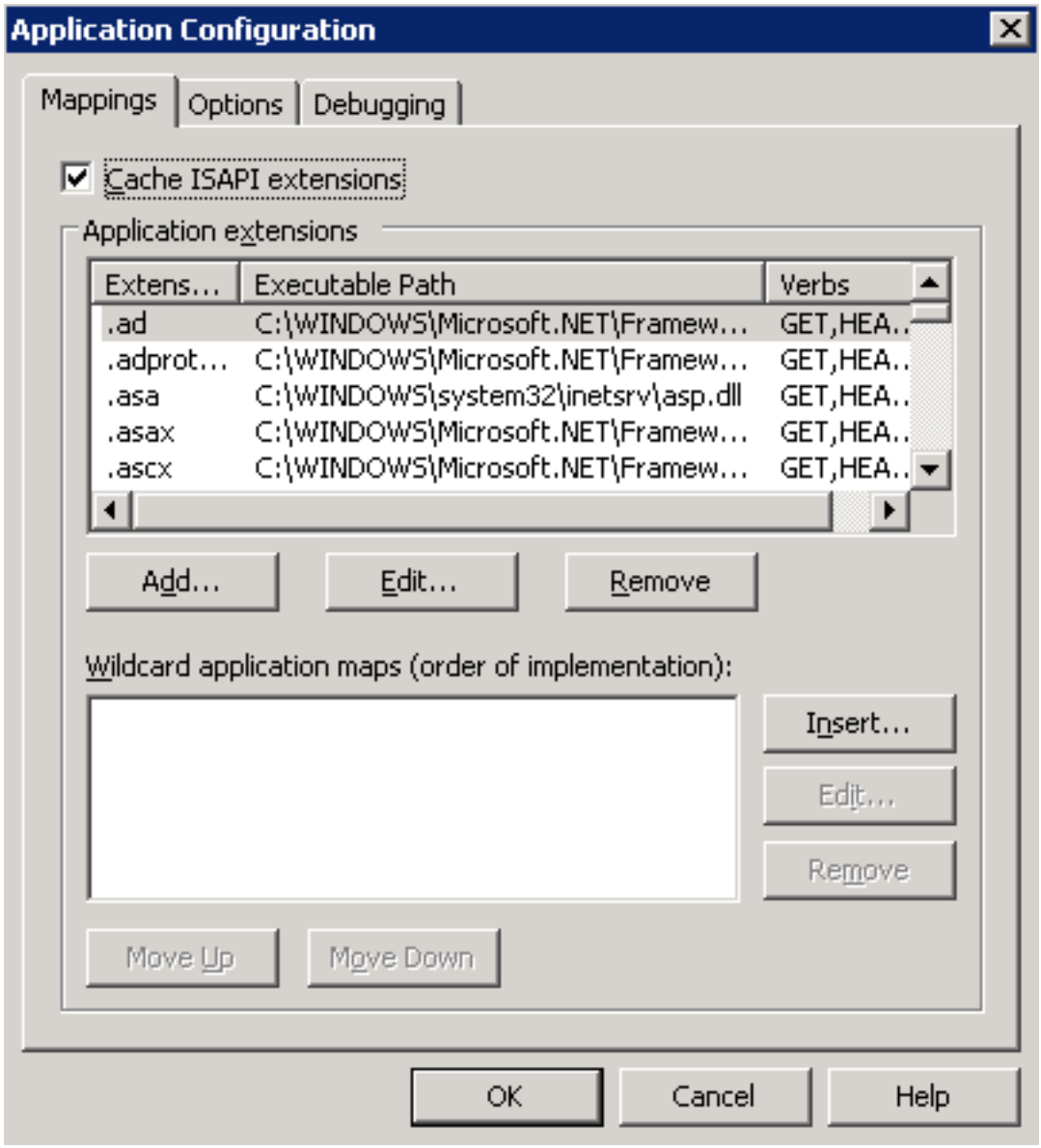
- At the same level as the “geomoose2” directory, create a virtual directory called “cgi-bin” and point it to the path “C:/ms4w/Apache/cgi-bin” using the same settings as the “geomoose2” directory.
- At the same level as the “geomoose2” directory, create a virtual directory called “ms_tmp” and point it to the path “C:/ms4w/tmp/ms_tmp” setting the permissions as shown below.

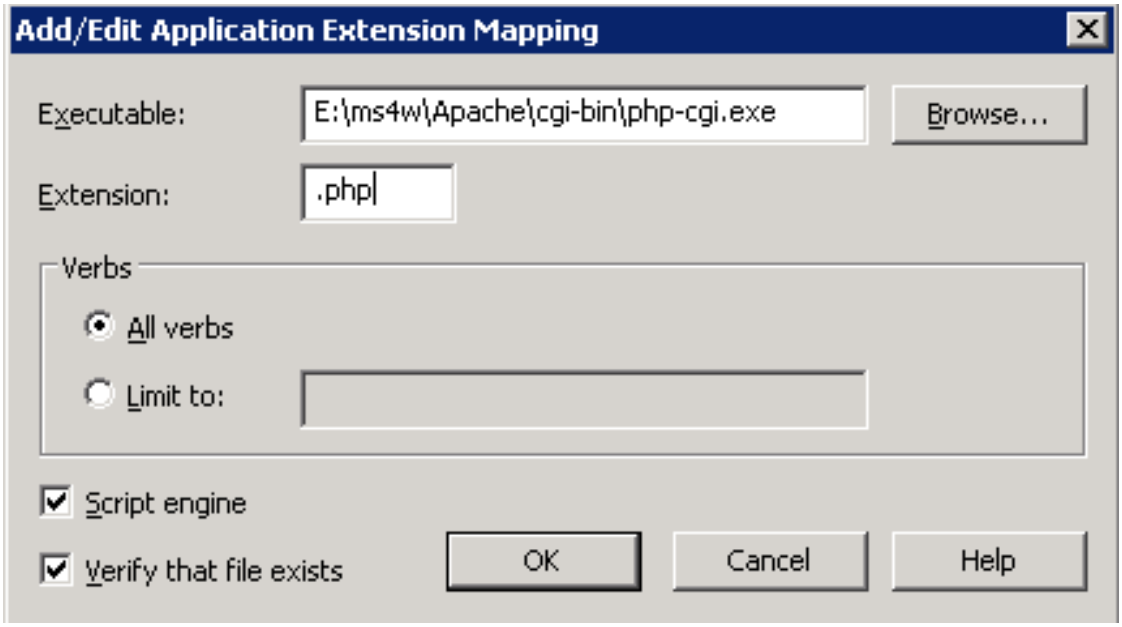


IMPORTANT NOTE There are many versions of PHP available, it is highly recommended that you use the version provided with MS4W as it includes all of the necessary components for MapServer to run properly.

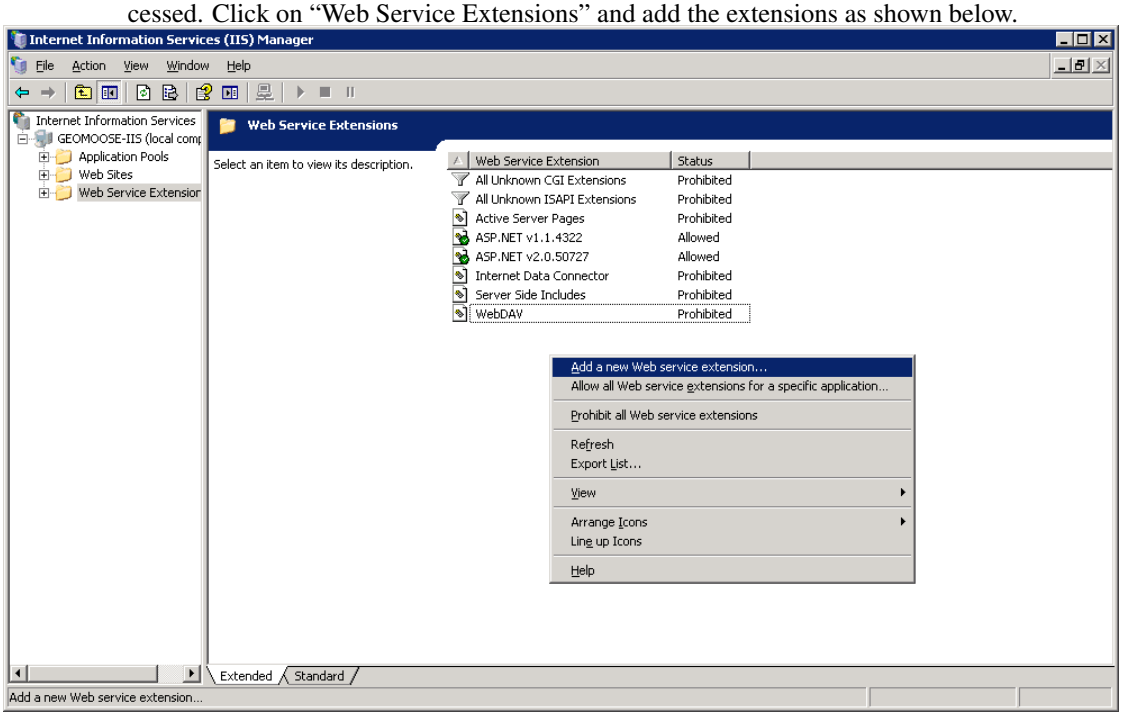
- Set the web service extensions to use PHP from "C:/ms4w/Apache/cgi-bin". This configuration can be done once at the web site level to provide support for all virtual directories (recommended), or at the directory level if some directories need to use different versions of PHP. If you configure PHP at the directory level, make sure to set PHP on all three GeoMOOSE directories.

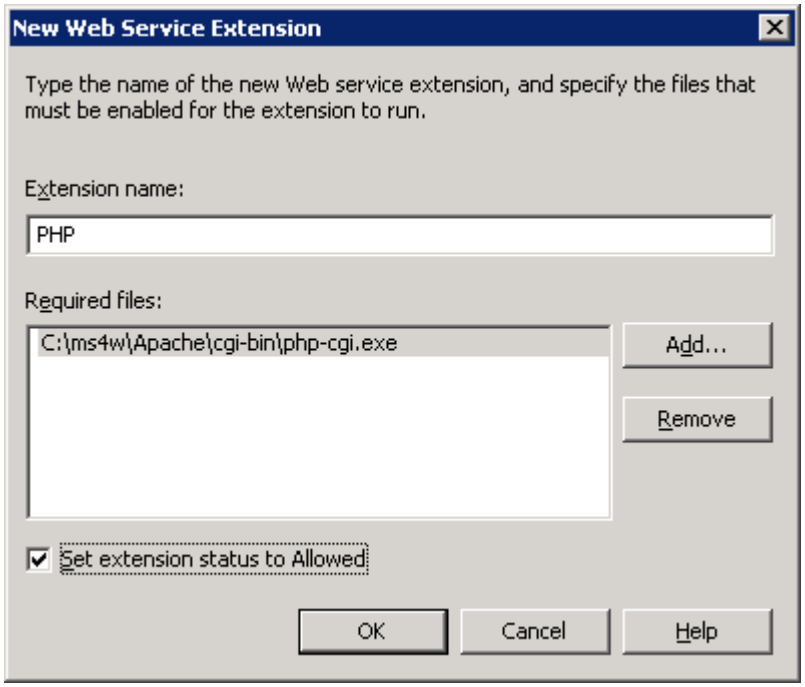

GeoMoose Demos Properties
Directory Security
HTTP Headers
Custom Errors
ISAPI_Rewrite
ASP.NET
Web Site
Performance
ISAPI Filters
Home Directory
Documents
The content for this resource should come from:
A directory located on this computer
A share located on another computer
A redirection to a URL
Local path:
Browse...
Script source access
Read
Write
Directory browsing
Log visits
Index this resource
Application settings
Application name:
Default Application
Remove
Starting point:
<GeoMoose Demos>
Configuration...
Execute permissions:
Scripts only
Application pool:
DefaultAppPool
Unload
OK
Cancel
Apply
Help

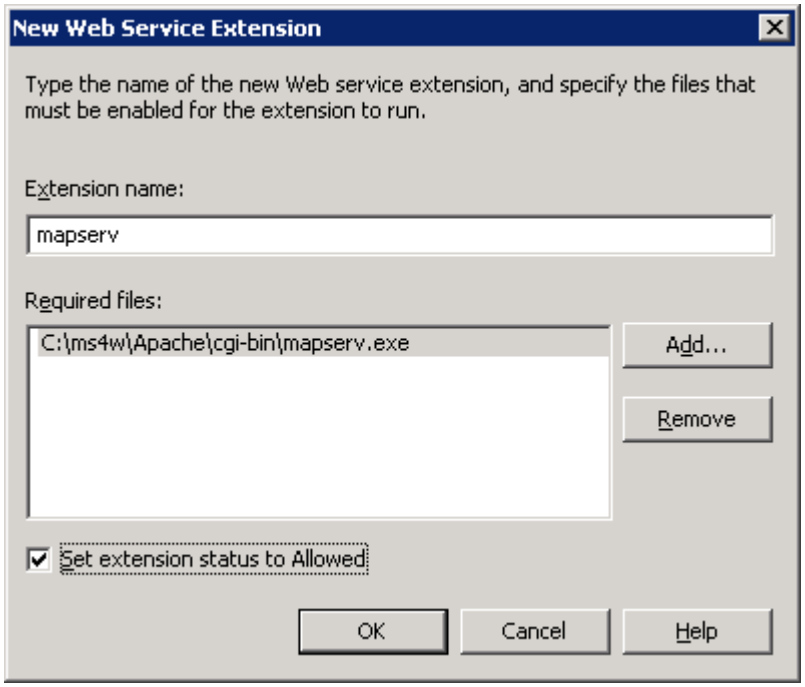

Application Configuration
Mappings
Options
Debugging
Cache ISAPI extensions
Application extensions
Extens...
Executable Path
Verbs
.ad
C:\WINDOWS\Microsoft.NET\Framew...
GET,HEA...
.adprot...
C:\WINDOWS\Microsoft.NET\Framew...
GET,HEA...
.asa
C:\WINDOWS\system32\inetsrv\asp.dll
GET,HEA...
.asax
C:\WINDOWS\Microsoft.NET\Framew...
GET,HEA...
.ascx
C:\WINDOWS\Microsoft.NET\Framew...
GET,HEA...
Add...
Edit...
Remove
Wildcard application maps (order of implementation):
Insert...
Edit...
Remove
Move Up
Move Down
OK
Cancel
Help



- IIS needs to be configures to allow two CGI applications (php-cgi.exe and mapserv.exe) to be accessed. Click on "Web Service Extensions" and add the extensions as shown below.







- From a text editor (i.e. WordPad) and open "C:ms4wApachecgi-binphp.ini". Search for "cgi.force_redirect" - this value must be set to zero. It will look like this `cgi.force_redirect = 0`
- To test if the installation is working, from a browser on the computer you installed MS4W and GeoMOOSE, type in the address `http://localhost/geomoose2/geomoose.html`.
- At the bottom of this page is a list of applications installed which use MS4W. Click on the GeoMOOSE 2.2 link to test the installation.

NOTE On some Windows-based computers, you will need to set an environment variable for the location of the prog4 library. This step is needed if you recieve the error "msProcessProjection(): Projection li-

brary error. No such file or directory". Instructions for adding the envronment variable can be found at `http://mapserver.org/errors.html`.

Common Issues

- Not having Administrator level access. This will prevent Apache from being installed as a service.
- Another web server is running on the computer. If IIS or Apache is already running on the computer on port 80, you will receive an error message when trying to start your new web server. There are many sources on the web which will provide instructions on running both IIS and Apache on the same server by changing the port used by one of the servers.
- Firewall blocking port 80. If you can access GeoMOOSE from the localhost but not other computers, check firewall settings on the computer hosting GeoMOOSE and other external firewalls.

Next Steps

Now that GeoMOOSE is installed, the next step is to start customizing it for your map. Some helpful documents to look at include:

- *Mapbook Reference Guide*
- *GeoMOOSE Configuration Options*
- *How To Add Your Own Layers*
- *How To Create Your Own Skin*

6.2 Service Documentation

6.2.1 General Query Service (query.php)

2.2+ ONLY

General Description

GeoMOOSE provides a general query service that allows the user to query multiple layers and fields through a single CGI query. Results can be optionally highlighted.

The Query service is intented to replace the itemquery.php service and eventually the identify service. Both ItemQuery and Identify may be dropped as of the official 2.0 release. Currently Shapefiles, OGR, Oracle Spatial, and PostGIS are the only supported layer types for the General Query Service. If you plan to search a layer other than you may consider modifying query.php to meet your needs or write another service entirely.

Important settings.ini Variables

- query_header - This will be prepended to the HTML results.
- query_footer - This will be append to the HTML results.
- query_miss - This is the message that will be displayed when no results are found.

As a compatibility step, the current version of query.php will default to using the "itemquery_..." versions of the above settings when it cannot find the proper "query_..." settings.

Parameters

The parameters for this service are unique because they can be variable based on the number of fields and layers that will be queried by the service.

- *mode* is a hidden input that should always be set to "search".
- *highlight* is a hidden input that is a boolean. "True" will mean the selected features will be highlighted, "false" means they will not be highlighted.
- *zoom_to_first* is a hidden input that is a boolean. When set to "true" GeoMOOSE will always zoom to the first feature found in the search, when set to "false" GeoMOOSE will not exhibit this behaviour.
- *layer[0..n]* is the Mapbook path of the layer to search. Only the 1st layer (layer0) is required. If no other layers are specified then it is assumed that the target of all of the following inputs are layer0.
- *value[0..n]* the value which should be search against. Only the 1st value (value0) is required. If no other values are specified then it is assumed that the initial value will be used to search all other fieldnames.
- *fieldname[0..n]* is the name of the field in the table or DBF to search against. This is required for every field.
- ***comparitor[0..n]* is the name of the comparison operator. Valid comparison operators include:**
  - "eq-str" - String equals match.
  - "like" - Case-sensitive contains operator. The equivalent SQL is "fieldname like '%value%"'.
  - "right-like" - Case-sensitive "begins with" operator. The equivalent SQL is "fieldname like 'value%"'
  - "left-like" - Case-sensistive "ends with" operator. The equivalent SQL is "fieldname like '%value"'
  - "like-icase" - Case-insensitive "like."
  - "right-like-icase" - Case-insensitive "right-like."
  - "left-like-icase" - Case-insensitive "left-like."
  - "eq" - Equals operator. This is used for numeric operations.
  - "gt" - Greater-than operator.
  - "lt" - Less-than operator.
  - "ge" - Greather-than or equal-to operator.
  - "le" - Less-than or equal-to operator.
- ***operator[1..n]* is the type of connecting operator between searching predicates. There is no "operation0" as it is the start o**
  - "and" - Require both this statement (n) and the previous one (n-1).
  - "or" - Require either this statement (n) or the previous one (n-1).
  - "nand" - Require this statement to be false (n) and the previous one to be true (n-1).
  - "nor" - Require this statement to be false (n) or the previous one to be true (n-1).
- *blanks[0..n]* is a hidden boolean tied to the value[0..n]. If set to "true" blank values will mean this predicate is ignored, when set to "false" then a blank value will be evaluated.
- *template[0..n]* specifies the template to use. This template is specified in the LAYER's METADATA tag. Only template0 is required. If no other templates are specified it will try to use the template named in template0 for all layers. For example, to mimic the itemquery functionality the "itemquery" template is used in the examples below. It is possible to use other templates, such as *identify_record*, and is even possible to provide this as a drop-down option to the user. This would allow a single search to have different result outputs.

Sample Mapbook Service Definitions

This is a basic example that emulates the old itemquery.php service:

```
<service name="search_parcels">
        <url>php/query.php</url>
        <step type="input">
                <input type="select" name="fieldname0" title="Search By:">
                        <option value="OWNER_NAME">Owner</option>
                        <option value="PIN">Parcel ID</option>
                </input>
                <input type="hidden" name="comparitor0" value="like-icase"/>
                <input type="user" name="value0" title=""/>
                <input type="hidden" name="layer0" value="parcels/parcels"/>
                <input type="hidden" name="template0" value="itemquery"/>

                <input type="hidden" name="highlight" value="true"/>
                <input type="hidden" name="mode" value="search"/>
        </step>
</service>
```

This is an example that allows the user to choose what comparitor is used to search:

```
<service name="search_parcels">
        <url>php/query.php</url>
        <step type="input">
                <input type="select" name="fieldname0" title="Search By:">
                        <option value="OWNER_NAME">Owner</option>
                        <option value="PIN">Parcel ID</option>
                </input>
                <input type="select" name="comparitor0" title="That: ">
                        <option value="like-icase">Contains</option>
                        <option value="right-like-icase">Begins With</option>
                        <option value="eq-str">Matches Exactly</option>
                </input>
                <input type="user" name="value0" title=""/>
                <input type="hidden" name="layer0" value="parcels/parcels"/>
                <input type="hidden" name="template0" value="itemquery"/>

                <input type="hidden" name="highlight" value="true"/>
                <input type="hidden" name="mode" value="search"/>
        </step>
</service>
```

Finally, this example shows searching multiple fields with selectable operators and comparitors:

```
<service name="search_parcels">
        <url>php/query.php</url>
        <step type="input">
                <input type="hidden" name="highlight" value="true"/>
                <input type="hidden" name="mode" value="search"/>

                <input type="hidden" name="layer0" value="parcels/parcels"/>
                <input type="hidden" name="template0" value="itemquery"/>

                <input type="select" name="fieldname0" title="Search By:">
                        <option value="OWNER_NAME">Owner</option>
                        <option value="PIN">Parcel ID</option>
                </input>
```

```
                <input type="select" name="comparitor0" title="That: ">
                        <option value="like-icase">Contains</option>
                        <option value="right-like-icase">Begins With</option>
                        <option value="eq-str">Matches Exactly</option>
                </input>
                <input type="user" name="value0" title=""/>

                <input type="hidden" name="fieldname1" value="FIN_SQ_FT"/>
                <input type="select" name="operator1">
                        <option value="or">OR</option>
                        <option value="and">AND</option>
                </input>
                <input type="select" name="comparitor1" title="Having Fin. Sq. Ft. ">
                        <option value="gt">Greater Than</option>
                        <option value="eq">Equal To</option>
                        <option value="lt">Less Than</option>
                </input>
                <input type="user" name="value1" title=""/>
        </step>
</service>
```

6.3 Extensions Documentation

6.3.1 Disclaimer

Description

Some users wish to put a disclaimer at the beginning of their application. Disclaimer.js provides this by prompting the user with an alert box when the application starts.

Adding to Application

To add Dynamic Select to your application add:

To add the Disclaimer to your application add the following lines:

```
<script type="text/javascript">
        window.DISCLAIMER_MESSAGE = "This is my disclaimer.";
</script>

<script type="text/javascript" src="extensions/Disclaimer.js"></script>
```

Setting *DISCLAIMER_MESSAGE* sets what disclaimer is show at startup. Without it the user will see "Hello, world."

6.3.2 Dynamic Select

Description

The default "Zoom To"s with GeoMOOSE are static and predefined in the Mapbook. They do not change based on updates of a layer, and can be tedious to type out even for layers with relatively few features. To address this, DynamicZoom provides an Ajax-capable feature look up that allows the user to select a dataset, then a feature from that dataset. After selecting the feature, the map zooms to the extent of that feature.

Adding to Application

To add Dynamic Select to your application add:

```
<script type="text/javascript" src="extensions/DynamicZoom.js"></script>
```

Configuration

Configuring Dynamic Select consists of three steps:

1. Create a template file.
2. Edit the layer's metadata.
3. Add the layer to DynamicZoomConfiguration

1. Create the template file

To create the template file, goto the direct where the mapfile exists. For this example, we'll be using the "counties_border" layer inside of the "basemap.map" mapfile. This mapfile can be found in the *maps/demo/statedata* directory that comes with the GeoMOOSE demo. The template is very simple, and creates the needed to create "<option>" tags in HTML:

```
<!-- MapServer Template -->
<option value="[shpext]">[COUNTYNAME]</option>
```

The first line is the MapServer "magic string" that is required for template files starting in version 5.4. The second line, has a few important prts. The first is the "[shpext]" directive, this tells mapserver to place the feature's extent into this part of the string. "[COUNTYNAME]" tells MapServer to place the feature's title, in this case 'COUNTYNAME,' into the template. When working with a different layer, substitute "COUNTYNAME" for whatever the layer's title column name happens to be. For this example, we'll name the file, "county_dynamic_zoom.html"

2. Edit the layer's metadata.

Now we need to tell GeoMOOSE about this template file. The method for doing that is to add a METADATA entry to the mapfile. First, it is neccessary to find the LAYER entry in the mapfile, that will look something like this:

```
LAYER
        NAME county_borders
        ...
END
```

Then, check to see if there is already a METADATA section, if there is not then add one:

```
LAYER
        NAME county_borders
        ...
        METADATA
        END
END
```

Finally, the metadata entry needs to be made:

```
LAYER
        NAME county_borders
        ...
```

```
        METADATA
                'dynamic_zoom' 'county_dynamic_zoom.html'
        END
END
```

3. Add the layer to DynamicZoomConfiguration

Edit *htdocs/extensions/DynamicZoom.js*, at the very top is a Javascript Object named "DynamicZoomConfiguration." In DynamicZoomConfiguration there is an entry called layers, it is an object that contains the information for the various layers. If the file is from a default-GeoMOOSE installation then the entry for "Counties" is already there. If it is missing then adding the following lines would create it:

```
...
layers: {
        'Counties' : {
                mapbook_src: 'borders/county_borders', /* this is the src of the layer in the mapbook
                qitem: '', /* leave blank */
                string: '' /* leave blank */
        }
}
...
```

Other Options

In the above instructions "qitem" and "qstring" were intentionally left blank. However, if they are set, they will act as a filter on the dataset. This is useful if you have multiple feature types stored in a single table. By default, the filtering is case-insensitive.

6.4 Helpful How To's

6.4.1 How To Add Your Own Layers

Adding a new layer in GeoMOOSE 2 is a two step procedure. The first step is to define how GeoMOOSE should communicate with the layer (whether it is MapServer or a WMS source) and the second step defines the layer in the catalog.

Step 1: Adding a Layer

All requests start with a `<map-source>` tag with a `name` and `type` attribute. The name is the name of the source that will be used with GeoMOOSE. This name does not need to relate to the source that is going to be read. The type attribute determines what type of service will be read. The children of the `<map-source>` are determined by the `type` attribute. Currently `mapserver` and `wms` types are the two that are supported.

All `<map-source>` types have `<layer>` children. `<layer>` children have only one attribute `name`. The `name` attribute for `<layer>` children reflects the name of the layer in the MapServer Mapfile or the name of the layer served by the WMS Service. Some MapServer mapfiles will have all layers set to "default" in which case there should still be one `<layer>` entry in the `<map-source>` with the name of "all."

A Mapserver Layer

Mapserver Layers need to have the `<file>` tag filled out specifying the location of the mapfile. This path can be relative to the `mapfile_root` configuration setting or an absolute path on the file system.

Here is an example of the parcels layer from the default demo:

```
<map-source name="parcels" type="mapserver">
        <file>./demo/landrecords/parcels.map</file>
        <layer name="all"/>
</map-source>
```

A WMS Layer

WMS Layers have a `<url>` child that defines the root URL of the WMS service.

This is an example using the LMIC FSA Photography:

```
<map-source name="lmic" type="wms" tiled="false">
        <url>http://geoint.lmic.state.mn.us/cgi-bin/wms</url>
        <layer name="fsa"/>
</map-source>
```

Step 2: Adding the Layer to the Catalog

After defining how to talk to the layer we need to define how to display the layer in the interface.

Layers are defined in the `<catalog>` section of the mapbook. To add the LMIC FSA photography the `<layer>` entry would look like this:

```
<layer title="LMIC 2003 FSA Aerials" src="lmic/fsa" status="on"/>
```

The individual parts:

- `title=` - Sets the title in the mapbook to be displayed
- `src=` - This is the layer name, it is a combination of the `map-source`'s name combined with the `layer`'s name from the definitions above.
- `status=` - This determines whether the layer is either on or off by when the map loads. Omitting `status=` will leave the layer off.

6.4.2 How To Create Your Own Skin

GeoMOOSE 2 has greatly simplified the skinning and CSS required to change elements of the interface.

Step 1: Make a Copy of a Previous Existing Skin

In the current distribution the easiest to modify skin is the "blue" skin. The directory has sub-directories so it will be necessary to do a recursive copy (the default on windows):

```
cd htdocs/skins
cp -r blue myskin
```

Step 2: Rename the Skin and Edit It!

This is just a small step to keep things clean:

```
cp blue.css myskin.css
```

Open myskin.css in a text editor and change the code! After following step 3 you'll see the changes in the interface.

Step 3: Set GeoMOOSE to use your Skin

Open "geomoose.html" in a text editor. In the `<head>` section there should be a tag starting `<link...` which mentions the skin. Comment out all of the skins you will not be using and add a `<link>` tag including your skin.

This is the default geomoose.html with the *myskin* skin selected:

```
<!--
        This is where your skin is defined.
        For an example, comment out the line containing "green.css" and
        uncomment "blue.css".
-->
<!--<link type="text/css" rel="stylesheet" href="skins/green/green.css"/> -->
<!--<link type="text/css" rel="stylesheet" href="skins/blue/blue.css"/> -->
<link type="text/css" rel="stylesheet" href="skins/myskin/myskin.css"/>
```

Tips and Advice

Changing just the Header Image

While some folks may want to change the entire look of the site you may only wish to change the header image to match your website. Follow all three steps above. This creates a unique skin for your site. Then change to the skin's image directory:

```
cd htdocs/skins/myskin/images
```

In that directory is the "logo_top.jpg" image. This is the image that is used in the header, it is 670 pixels by 59 pixels. Making a replacement image of similar size (the height is especially important) will change the logo atop the website!

6.4.3 How To Change the Tabs

GeoMOOSE 2 now features a flexible tabbing capability. This allows administrators to have any number of tabs available to the user in the interface. This document will show how to take a svn-derived instance of GeoMOOSE and reduce the tabs down to two ('Layers' and 'Information').

Step 1: Define the Tabs in the HTML

HTML from SVN will typically look like this:

```
<div id="tab-container">
        <div id="catalog-tab">
        </div>

        <div id="search-tab">
                SEARCH TAB
        </div>
```

```
        <div id="results-tab">
                No Results Yet!
        </div>

        <div id="service-tab">
                Form Information Goes Here.
                <div style="width: 400px; height: 20px; background: red">
                        abcdef
                </div>
        </div>
</div>
```

This HTML exists as it serves to test a number of Javascript and CSS configuration parameters. This interface is not what is useful to deliver to users. The first task is to cut down the tabs to only the ones we want to display:

```
<div id="tab-container">
        <div id="layers-tab"></div>
        <div id="information-tab"></div>
</div>
```

If you were to reload the geomoose.html page then you would see an entirely broken interface. This is because we need to adjust the configuration parameters to support this change.

Step 2: Change the Configuration

As with most configuration changes with GeoMOOSE this requires changing the mapbook. We need to specify some basic information about where specific content should be place in the interface.

Step 2(a): Define the Tabs

GeoMOOSE uses the "tabs" configuration parameter to define the mapping of tab-names and div-ids in the interface. The default mapping looks like this:

```
<param name="tabs"><![CDATA[
 {
         'Catalog' : 'catalog-tab',
         'Search' : 'search-tab',
         'Services' : 'service-tab',
         'Results' : 'results-tab'
 }
 ]]></param>
```

With the changes made to geomoose.html earlier we now need to add this to the `<configuration>` section of the mapbook to map our new Div's to Tab names:

```
<param name="tabs"><![CDATA[
 {
         'Layers' : 'layers-tab',
         'Information' : 'information-tab'
 }
 ]]></param>
```

Step 2(b): Define What is in the Tabs

Some tabs will have custom content. This can be a custom form or a special copyright message. In this case, the HTML can be left in the tab and none of these settings will have an effect. However, there are very important settings for targetting where specific content will be placed.

- `default_tab` - This is the tab that will be selected when the interface is opened.
- `catalog_name` - This is the name of the tab that will contain the catalog.
- `show_service_settings_in` - The name of the tab where all service settings will be shown. Service settings are the options that are presented to the user when they click on a service like "Select" or "Print."
- `show_results_in` - This sets the name of the tab that will contain the results from services.

To make this example work we would need to add the following to the `<configuration>` section of the mapbook:

```
<param name="default_tab">Layers</param>
<param name="catalog_name">Layers</param>
<param name="show_service_settings_in">Information</param>
<param name="show_results_in">Information</param>
```

6.4.4 How To Put Content in the Menubar

The menubar's content can be specified by using simply HTML in the Mapbook.

In the `<configuration>` section you can add:

```
<param name="links_bar_html"><![CDATA[
        <b>THIS IS MY HTML!!!</b>
]]></param>
```

Clear the browser cache and reload the interface! The HTML should appear where the menubar once existed!

6.4.5 How To Add Text to a Tool

Some tools should have text. While an Icon is nice, sometimes you may only have a single icon that is used repetetively for a number of searches. As is often the case, it's simply easier to read, "Search Assets" instead of filing through the four or five relatively obscure looking 16 x 16 pixel icons. Tools on the toolbar are defined in the `<toolbar>` section of the mapbook.

For this example, look at the "print" tool:

```
<toolbar>
        (snip)
        <tool name="print" title="Print" type="service" service="print" selectable="false"/>
        (snip)
</toolbar>
```

When GeoMOOSE renders the tool in the toolbar it creates a unique ID based on the name of the tool. Since the `<tool>` has the name of "print" the ID will be "tool-print". Knowing this we can change the CSS properties of the tool in the toolbar. GeoMOOSE renders the text for every tool in the toolbar, however, by default that text is hidden. To display the text a few lines need to be added to *user_tools.css*:

```
#tool-print {
        /* By default the tools "width" is limited to 20 pixels */
        width: auto;
```

```
        /* This sets a custom icon for the tool */
        background-position: 2 2;
        background-image: url('../images/toolbar/printer.png');
}

#tool-print .ToolContent {
        /*
         * This must be in the code in this order, it makes the tools display
         * properly in both IE and FF.
         */

        display: -moz-inline-box;
        display: inline-block;

        /* This allows the tool to grow to fit the text */
        width: auto;
}

#tool-print .ToolText {
        /*
         * These padding settings center the text in the middle of the tool.
         * Depending on your icon this may not be neccessary.
         */
        padding-top: 3px;
        padding-right: 3px;

        /*
         * This must be in the code in this order, it makes the tools display
         * properly in both IE and FF.
         */
        display: -moz-inline-box;
        display: inline-block;
}
```

The CSS above is commented to explain the different actions of changing those specific CSS attributes, however, in a real application those comments would not need to be included.

6.4.6 How To Call a Service at Startup

GeoMOOSE is centered around its services. Services are defined in the mapbook by the `<service>` tag and they define how GeoMOOSE should communicate with external scripts. There are a number of scenarios in which it is neccessary to call a service on startup. The primary example of this functionality is the ability to zoom to and highlight a feature upon opening the interface.

Given the item query service provided with the demo:

```
<service name="search_parcels">
        <!-- The URL of the service to call -->
        <url>php/itemquery.php</url>

        <!-- most service have one or two "steps" this one is a non-spatial input -->
        <step type="input">

                <!-- the field to search against -->
                <input type="select" name="qitem" title="Search By:">
                        <option value="OWNER_NAME">Owner</option>
                        <option value="PID">Parcel ID</option>
```

```
                </input>

                <!-- this is the actual search string -->
                <input type="user" name="qstring" title=""/>

                <!-- these are the rest of the settings for the service -->
                <input type="hidden" name="layer" value="parcels/all"/>
                <input type="hidden" name="zoom_to_first" value="false"/>
                <input type="hidden" name="highlight" value="true"/>
        </step>
</service>
```

Using this service there are two important things to note:

1. The name of the service, "search_parcels"
2. The inputs of type "user", these are the inputs that would require the user to enter text. We need to bypass that with parameters for the URL. In this example, the only "user" input is "qstring".

Calling the URL

To call the service, the url must specify the "call" parameter and the appropriate user-inputs:

```
geomoose.html?call=search_parcels&qstring=Johnson
```

- The "call" parameter specifies the name of the service to call
- The "qstring" parameter is the same name of the input of type "user" and its value is set to "Johnson"

6.4.7 How To Create a Custom Printing Template

GeoMOOSE 2 uses a page-templating method for generating PDFs. Any printed map in GeoMOOSE starts as a pre-existing PDF that does not feature a map. This PDF can have inline fonts, J PEGs, watermarks, etc. In GeoMOOSE 1.X it was necessary to write PHP to change the output format of the printing utility with 2.X that is no longer needed. Creating PDF files have b ecome commonplace and can be done with almost any software with the avilability of PDF printers.

Step 1: Make Your Template PDF

The first step is to create a PDF that you wish to use as a printing template. This can be done with any software that can output to a PDF. If none of your software has an "Export PD F" option then a PDF printer may be an option on your operating system:

- Mac OS/X - A PDF Printer is built-in. See Apple's Documentation. http://docs.info.apple.com/article.html?path=Mac/10.4/en/mh1035.html
- Windows - I've had good luck with CutePDF. http://www.cutepdf.com/
- Linux - Use CUPS. http://www.linux.com/articles/61826

For purposes of this document the PDF file generated here will be referred to as "my_template.pdf". "my_template.pdf" should be saved/copied into `conf/print`.

Step 2: Define the Template for GeoMOOSE

GeoMOOSE is smart, but not that smart, it does need to be told more information about the template created in Step 1. To do this we need to create an XML template definition file. It is easiest to copy a pre-existing template and modify it. For convenience, it makes sense to name the XML file the same as the PDF file. Copy letter_landscape.xml to my_template.pdf. You should now have a file that contains:

```
<print-template>
        <!-- This is the template -->
        <template>default_template.pdf</template>

        <!-- page dimensions -->
        <page w='11' h='8.5'/>

        <!-- this is the location of the map in the template -->
        <map x="0.43" y="1.43" w="10.15" h="6.42"/>

</print-template>
```

The important element to change is the name of the PDF template file to "my_template.pdf":

```
<print-template>
        <!-- This is the template -->
        <template>my_template.pdf</template>

        <!-- page dimensions -->
        <page w='11' h='8.5'/>

        <!-- this is the location of the map in the template -->
        <map x="0.43" y="1.43" w="10.15" h="6.42"/>

</print-template>
```

It is best to have a good guess for the `<page>` and `<map>` attributes but they can be adjusted later to fine tune the template.

Step 3: Add the Template to the Print Options

In the mapbook, the print utility has an `<input>` that allows the user to select a template. To make the template selectable it is necessary to add it to the mapbook definition.

The print service will initially have the following options:

```
<service name="print" title="Print Map">
        <url>php/print.php</url>
        <input type="print_info" name="layers"/>
        <input type="extent" name="extent"/>

        <input type="select" name="template" title="Output Template: ">
                <option value="letter_landscape">Letter - Landscape</option>
                <option value="letter_portrait">Letter - Portrait</option>
                <option value="poster_landscape">Poster - Landscape</option>
                <option value="poster_portrait">Poster - Portrait</option>
        </input>
(snip)
</service>
```

In order to make "my_template" available it just needs added to the list:

```
<service name="print" title="Print Map">
        <url>php/print.php</url>
        <input type="print_info" name="layers"/>
        <input type="extent" name="extent"/>

        <input type="select" name="template" title="Output Template: ">
                <option value="my_template">MY TEMPLATE!!!!</option>
                <option value="letter_landscape">Letter - Landscape</option>
                <option value="letter_portrait">Letter - Portrait</option>
                <option value="poster_landscape">Poster - Landscape</option>
                <option value="poster_portrait">Poster - Portrait</option>
        </input>
(snip)
</service>
```

Reloading GeoMOOSE should show the new template as available and users should now be able to print using that template.

Other Notes

It is possible to add dynamic text to a map. This is done by specifying a `<text>` tag. The text tells the printing service to place text on the page and it supports some basic string substitution. Strings featuring a variable like '%abc%' will be substituted with URL values. For example if the string is:

```
Hello %name%!
```

And "name=Dave" is passed into the print script via the CGI parameters, the string will become:

```
Hello Dave!
```

An example can be found in the demo's printing templates:

```
<text x=".5" y=".75" size="48" content="%title%"/>
```

The attributes breakdown like this:

- x - The x-position in inches on the page.
- y - The y-position in inches on the page.
- size - The font-size in points.
- content - The string to place on the page.

6.4.8 How To Avoid getmapbook.php

There are installations that do not use the PHP services provided with GeoMOOSE. If the reader is truly intereted in this document then they are probably one of those exact users. This how to is written assuming you have a strong sense of how your server is configured and how the GeoMOOSE services system functions in your environment.

Why This is a Bad Idea

The mapbook is the all-knowing configuration file for GeoMOOSE. All of the built-in services use this file as the center definition for the application. There are also a few non-philosophical reasons why getmapbook.php exists:

- Central Configration. Using the default configuration, to change the current mapbook, only one line in conf/settings.ini needs to change and all of the services and the main HTML/JS application switch to using that mapbook.
- Future Security. While GeoMOOSE 2.0 has not implemented the feature yet, we are actively building a security model that uses the getMapbook() function as the central marschall for authorization. This security feature will allow a single GeoMOOSE installation to be used for either internal or external use. It also promises to allow an organization to prevent differing options to users based on a login.

[/soapbox]

Why It's Okay

As stated in the introduction, if you're doing this, you should know what you're doing.

Here are a couple of reasons to override the mapbook setting:

- If you are having difficulty getting PHP working on your server and want to experiment with GeoMOOSE
- You want to use a different server-side scripting language (such as Python or ASP) and are not interested in using the other GeoMOOSE PHP services.

Step 1: Override the 'mapbook' Setting

The 'mapbook' attribute of CONFIGURAITON is the only one that cannot be overriden after the mapbook is loaded. Thus, some small changes need to be made to the geomoose.html file. Find the line that includes "compiled.js" and add the URL to your mapbook afterwards:

```
<script type="text/javascript" src="compiled.js"></script>
<script type="text/javascript">
        CONFIGURATION['mapbook'] = 'mapbook.xml';
</script>
```

Step 2: Change the settings.ini File (PHP Only)

It is possible to use the other PHP services with a mapbook over-ride present but they still need to know the location of that mapbook. The `settings.ini` file in the `conf/` director contains such information.

conf/settings.ini:: [defaults] ; Location of the default mapbook mapbook=../htdocs/mapbook_demo.xml

6.4.9 How To Customize Your Waiting Message

Some services can take serious time to load. This is especially true of the printing service which needs to assemble many high resolution images. To help the user understand the application is running it is often useful to have a waiting message. To customize the waiting message simply change the waiting_html configuration parameter in your mapbook:

```
<param name="waiting_html"><![CDATA[
        <center>
                <b>Please wait while this loads...</b>
        </center>
]]></param>
```

6.4.10 How To Change Default Drawing Colors

GeoMOOSE 2 features client side red-lining/drawing/sketching tools. These tools allow the user to place their own markup on the map and edit that markup. Currently, the shapes drawn can have their shape, fill color, stroke color, and opacity adjusted after creation. While the default red/green combination that GeoMOOSE employs by default is *very* attractive we recognize it may be neccessary to adjust those settings. To do so there are three configuration variables that need to be over-ridden in the mapbook:

```
<param name="drawing_tools.default_fill">red</param>
<param name="drawing_tools.default_stroke">yellow</param>
<param name="drawing_tools.default_opacity">1</param>
```

The "red" and "yellow" colors are used an example. These colors can be any one of the 16 supported W3C colors, a six digit RGB color code (ex., #00FF00), or a three digit RGB color code (ex., #0F0).

6.4.11 How to set the Defaults for the Measure Tools

Units

The measure tool units are defined using configuration parameters. There is a parameter for line measuring units and area measuring units:

- measure_tool.line_units = mi,ft,m
- measure_tool.area_units = mi,ft,m,yd,acre

Translations:

- mi = Miles
- ft = Feet
- m = Meters
- yd = Yards
- acre = Acres

All of the area_units specifications are assumed to be square units (for example, mi = Square Miles, for area measurement). This example sets the default units to square Miles:

```
<param name="measure_tool.area_units">mi</param>
```

Precision

By default the measure tool will display three digits of precision. This can be changed via the `measure_tool.precision` configuration variable:

```
<param name="measure_tool.precision">4</param> <!-- measure tool will now display four digits of prec
```

6.4.12 How to use a WMS Service in a Different Projection

Not all WMS services use the projection that a GeoMOOSE application may use natively. In this case, it is necessary to reproject the request to display properly.

Step 1: Prerequisites

Before we can do the reprojection both projections must be defined! GeoMOOSE has EPSG:4326 predefined so if the WMS service uses EPSG:4326 you will not need to define this projection. Please review *Adding Projections* for instructions on creating projections and adding them to the interface.

Step 2: Configure the WMS map-source

To define a WMS map-source in a different projection the `projection` attribute needs to get set to the projection code:

```
<map-source name='nasa' type='wms' tiled='false' projection='EPSG:4326'>
        <url>http://onearth.jpl.nasa.gov/wms.cgi</url>
        <layer name="global_mosaic"/>
        <param name="format" value="image/jpeg"/>
</map-source>
```

Step 3: Add the Layer to the Catalog

This is just like any other layer:

```
<layer title='NASA Global Mosiac' src='nasa/global_mosaic'/>
```

6.4.13 How to Create Custom "Input" Types

From time to time is may be necessary to create a custom "input" type for a service with GeoMOOSE. This happens when you cannot get information about the map or from the user in a way that GeoMOOSE supports by default.

Step 1: Create an Input Type

Create the file, this one I called "MapWidth.js":

```
GeoMOOSE.Services.InputType.MapWidth = OpenLayers.Class(GeoMOOSE.Services.InputType, {
        MAPBOOK_NAME: "map_width",

        getValue: function() {
                var extent = Map.getExtent();
                return extent.right - extent.left;
        }
});
```

Step 2: Add it to the "geomoose.html"

- Open "geomoose.html" in a text editor.
- Find *</head>* in the file.
- Add the following line:

  ```
  <script type="text/javascript" src="MapWidth.js"></script>
  ```

Step 3: Add the input type to a Service

Find the service to add the type to and add the input definition:

```
<service ... >
        <input type="map_width" name="mw"/>
</service>
```

Step 4: Enjoy

This is the easiest part!

6.4.14 How to Add Google, Yahoo, or Bing Layers

Notes on All Layers

- Commercial layers can only be used as basemap layers. You cannot elevate them any higher in the list.
- Commercial layers cannot be printed. The licensing agreements for these layers do not allow them to be printed.
- You must shift your map into Web Mercator (aka "the Google Projection"). Please read *Changing the Projection*

How to Add Google Layers

Step 1: Add the Google API to geomoose.html

- Open "geomoose.html" in a text editor.
- Find the <*script*.. tag that contains 'OpenLayers.js' in the 'src' attribute.
- Then add the Google API, the code should look like this:

```
<script type="text/javascript" src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAAnfs7bKE
<script type="text/javascript" src="OpenLayers-2.8/OpenLayers.js"></script>
```

Step 2: Add the Google map-source Definitions

Now it is necessary to put the map-source definitions into the Mapbook:

```
<map-source name="google_physical" type="google" google-type="physical">
        <layer name="all"/>
</map-source>

<map-source name="google_streets" type="google" google-type="streets">
        <layer name="all"/>
</map-source>

<map-source name="google_hybrid" type="google" google-type="hybrid">
        <layer name="all"/>
</map-source>

<map-source name="google_satellite" type="google" google-type="satellite">
        <layer name="all"/>
</map-source>
```

Step 3: Add the Definitions to the Catalog

Google layers can only be used as basemap layers. So when using them as a background it is usually wise to simply create a backgrounds group and allow the user to toggle between the various layers:

```
<group title="Backgrounds" expand="true" multiple="false">
        <layer title="Streets" src="google_streets/all" status="on"/>
        <layer title="Physical" src="google_physical/all" status="off"/>
        <layer title="Hybrid" src="google_hybrid/all" status="off"/>
        <layer title="Satellite" src="google_satellite/all" status="off"/>
</group>
```

6.5 Release Notes

6.5.1 GeoMOOSE 2.2 Release

Features:

- Google, Bing, Yahoo, ArcGIS Server Cache Layers!
- A new super-query service!
- User Extensions
- Better, Customizable Input Types
- More bug fixes!
- Bug fixes to the Geocoder and Popup Services
- Addition of Feature Report service

You might be asking your self the following questions:

1) Is it worth the upgrade? Yes.

2) Is this the version that should convince me to upgrade from 1.X to 2.X? Yeah, this is definitely a worthwhile upgrade. The User Extensiosn and Custom Input types have really advanced customizations unavailable in the 1.X series of releases.

6.5.2 GeoMOOSE 2.0.1 Release

Features:

- A multitude of bug fixes.
- Sketching/Redlining Tools
- Improved Printing Configuration
- Improved Demo
- Upgrade to OpenLayers 2.8
- autotools based configuration/installation.

Install Instructions for UNIX/Linux have been added:

http://www.geomoose.org/moose/docs/install_unix.html

You might be asking your self the following questions:

1) Is it worth the upgrade? Yes.

2) Is the upgrade as painful as upgrading old GeoMOOSE 1.X.X versions? No! You should not have to change your mapbook at all. You should back up your settings.ini and verify changes there.

3) Is this the version that should convince me to upgrade from 1.X to 2.X? Probably, but 2.2 will be coming soon and that will be even more worthwhile with additional bug fixes and a few additional features.

4) Are there still bugs? Yep, we even know about a bunch of them and are working on fixing them.

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*