WEB MAPPING COOKBOOK

CVEN 8390

Jessica Garrett
November 28, 2016

Table of Contents

Chapter 1: Mapbox

Introduction

Mapbox is hosted online mapping platform for GIS professionals and developers. Based on a monthly service fee, there are various tiers of service or plans; starter, premium and enterprise. The starter addition is free and can support up to 50,000 map views or users. The premium and enterprise unlocks many other services such as paid applications, email support, encryption mapping and can support up to 1 million and 5 million views, respectively. Note that all three additions include various base maps, geocoding and direction routing applications. The starter plan is ideal for students or small companies just getting their feet wet in the web mapping industry.

Getting Started

The first step to getting started with MapBox is creating an account (you must have a valid email address). The interface is fairly user friendly; from the home screen, we can navigate to data we have uploaded, maps we have created as well as your personal account information. From a beginner's standpoint, *Mapbox Studio* is a great place to start and is how we create web maps. Navigate to the developers tab to explore more advanced level mapping tools.

Uploading Data

A *dataset* is a collage of GeoJSON features which can be uploaded from your local machine or can be created from scratch within Mapbox. Note that a dataset upload must either be in GeoJSON or CSV format. An essential aspect of a dataset is that it is *editable.* In contrast, a *tileset* is static where feature data cannot be added, edited nor deleted. We can export a dataset to a tileset, or simply uploaded a (non-editable) file from your local drive. Acceptable file formats for tilesets are BTiles, KML, GPX, GeoJSON, Shapefile (zipped) or CSV. We can upload raster data from a GeoTIFF file.

After we have created tilesets, it is time to add these to a *style*. We can think of a *style* as an individual map project. We can include built-in Mapbox tiles such as political boundaries and labeling, transportation polylines as well as natural formations like lakes and streams. To do this, just add a base map of choice then delete unwanted tiles.

Editing Symbology

One great component of Mapbox is its ability to enhance personalized cartography on the fly. With an active style, we can edit symbology to a desired appearance. Possible adjustments include color, line thickness, transparency, symbols, etc. We can even change feature presentation base on zoom-level; there are 22 levels of zoom.

Publishing

After we have created a map in Mapbox Studio, we may want to publish our work to other users. With an active style, click the **Publish** button in the top left corner. Then navigate to your map under styles and then select **Share, develop and use** on the right-hand side. Mapbox then provides us with an URL in which we can copy then open in a browser.

Chapter 2: GDAL

Introduction

Geospatial Data Abstraction Library (GDAL) is a free, open source software (FOSS) created by the Open Source Geospatial Foundation in 2000. It is a translator library used to read and write raster and vector data. Traditionally, GDAL was only compatible with raster data, and its sister software, *OpenGIS Simple Features Reference Implementation* (OGR) with vector. However, with modern releases the two are heavily integrated, which is why we often see GDAL/OGR pair together. GDAL operates in command line and does not have any associated graphical user interface (GUI), at least published by OSGF. However, many mainstream GIS software companies use GDAL in the background; ESRI ArcGIS 9.2+, Google Earth, QGIS, MapServer, GeoServer, ERDAS and many more. See the full list here.

Install

For window's 64 bit machines, navigate to http://www.gisinternals.com/query.html?content=filelist&file=release-1800-x64-gdal-2-1-0-mapserver-7-0-1.zip and find the generic installer for GDAL core components (fifth option down). Follow the steps within the installation wizard. For beginners, it is recommended to select the "complete" setup type rather than "typical" or "custom". Note that the complete option requires more disk space than typical and the custom allows user to choose which features will be installed. The program can simply by found by searching "GDAL" under All Programs. For Unix/Mac users, go to http://www.kyngchaos.com/software:frameworks for installation instructions.

Raster Data

GDAL supports over 100 raster formats. Some commonly used file types are JPEG, PNG, TIF, GIF, BMP, etc. Click here for a full list. Given a valid raster data type, we can perform various operations on our data. Please see bullets below for a few very common and useful GDAL commands. Please note that the format present is directly taken from http://gdal.org/gdal_utilities.html. The text printed in bold is the actual command, while the text in brackets will call more specific information.

- **gdalinfo** [--help-general] [-json] [-mm] [-stats] [-hist] [-nogcp] [-nomd] [-norat] [-noct] [-nofl] [-checksum] [-proj4] [-listmdd] [-mdd domain|`all`] [-sd subdataset] [-oo NAME=VALUE] * datasetname

Description: Provides information about the dataset. A few listed items are file size, coordinate system, band data types, min/max values and unit type.

- **gdal_polygonize.py** [-8] [-nomask] [-mask filename] raster_file [-b band] [-q] [-f ogr_format] out_file [layer] [fieldname]

Description: Creates polygon features from a raster file.

- **gdalcompare.py** [-sds] golden_file new_file

Description: Compares tow raster files and prints results. Note that the second file (new_file) is compared to the first (golden_file).

- **gdalmove.py** [-s_srs <srs_defn>] -t_srs <srs_defn> [-et <max_pixel_err>] target_file

Description: Transforms the coordinate system of the raster file.

Vector Data

GDAL/OGR is compatible with most vector file types, including SHP, KML, TIGER, CSV, GML, etc. There roughly 90 unique supported extensions. A complete listing can be found here. The following commands can be used on vector data.

- **ogrinfo** [--help-general] [-ro] [-q] [-where restricted_where|\@filename] [-spat xmin ymin xmax ymax] [-geomfield field] [-fid fid} [-sql statement|\@filename] [-dialect dialect] [-al] [-rl] [-so] [-fields={YES/NO}] [-geom={YES/NO/SUMMARY/WKT/ISO_WKT}] [-formats] [[-oo NAME=VALUE] ...] [-nomd] [-listmdd] [-mdd domain|`all`]* [-nocount] [-noextent] datasource_name [layer [layer ...]]

Description: Prints a summary of the dataset to the consol. The tag [-al] will list all features in the dataset. [-so] stands for summary only and will provide information like projection, feature count and extents.

- **ogr2ogr** [--help-general] [-skipfailures] [-append] [-update] [-select field_list] [-where restricted_where|\@filename] [-progress] [-sql <sql statement>|\@filename] [-dialect dialect] [-preserve_fid] [-fid FID] [-spat xmin ymin xmax ymax] [-spat_srs srs_def] [-geomfield field] [-a_srs srs_def] [-t_srs srs_def] [-s_srs srs_def] [-f format_name] [-overwrite] [[-dsco NAME=VALUE] ...] dst_datasource_name src_datasource_name

Description: Converts data features between file types.

Navigating Command Line

If you are not familiar with a command line interface, working with GDAL can be intimidating at first. There are a few commands that are very useful when running GDAL.

- cd – Change directory. This allows you not to have to type out the entire path (location) of the file.
- dir – Lists all files and folders in the current directory.
- del – Delete files or folders.
- verify – Check if files have been saved.
- cls – Clear the console screen.
- md – Create new folder or directory.

More CMD commands can be found here for a Windows 64 OS.

Chapter 3: MapQuest API's

Introduction and Getting Started

MapQuest has made great strides from the late 90's; it is more than just a traffic navigation tool. MQ offers many products and services in the big data and geospatial realm. Specifically, their developer tools provide business solutions through various APIs. Companies like Papa Johns, Home Depot, Sabre, etc. use MQ API in the background of their own platforms. To get started, navigate to https://developer.mapquest.com and create an account. A free addition comes with 15,000 transactions a month with services like mapping, geocoding, routing, traffic, spatial search, data manager and forum support. You will need to create a (free) key. This chapter will focus on a select few MQ's APIs.

JavaScript API:

After you have created a developer account, creating a web map via JavaScript API is very simple. First, download a text editor, if you don't have one already. I recommend Notepad++ or Sublime. Next, open a new HTML session. Firstly, we will want to add a block of code found here. Within the text, we can see adjustable options like initial zoom levels, center of the map (latitude and longitude), etc. The first chunk of code will produce a very basic map. To enhance features and user-interaction, we can also include a handful of different controls.   A few components to consider are large zoom, small zoom, traffic control and a drawing control. You can find all available controls here. After your code is complete, navigate to run/launch in Chrome in Notepad++ to open HTML file in browser.

Geocoding API

Arguably one of the most popular developer tools offered by MQ, geocoding matches addresses with a set of coordinates. The need for this function surfaces a lot when gathering data that may not be necessarily geared for a geospatial user. *Forward* geocoding (also called address coding) provides users with an associated set of coordinates with an address input. In contrast, *backward geocoding* generates addresses based on given coordinates. If more than one possible result is produced, MQ will list outcomes in order of confidence. A batch function call will kick back up to 100 results with just one input. You can test the Geocoding API here.

Data Management API

This API is basically cloud storage for geospatial data. Vector data is stored as *geographies* to enable the user to perform spatial queries. The data management API is easily integrated into other MapQuest's APIs. Also, address geocoding is a built-in feature with this API. Supported file types for upload are CSV, KML and even zipped SHP files. The Data Management API is equipped with security settings to provide data privacy.

Direction API

Perhaps what many MapQuest users originally used the platform for, the Direction API provides routing from departure and destination points. The API also supports multi-point requests for up to 50 unique locations. Using real-time traffic data, this tool can provide alternate routes to ensure the fastest solution is provided. Users can also request shortest distance results as well as paths specific for pedestrians or bicyclists. In addition, the API allows for special requests like avoiding ferries, highways or toll roads. Note that the input can either be in address or coordinate format. Directions API is the engine that runs MapQuest.com. See the full list of features for this API here.

Chapter 4: GeoServer

Introduction

GeoServer is a free, open source software allowing users to generate and share maps as well as edit data through a Java-based server. It is known for its interoperability with other GIS platforms such as ESRI's products, Google Earth and Maps, Yahoo Maps and Microsoft Visual Earth. GeoServer maintains standards defined by Open Geospatial Consortium (OGC). Integrating the mapping library OpenLayers, outputs are easy to generate with pleasing cartography. The GIS library, GeoTools also helps run the application. GeoServer is considered both a Web Mapping Service (WMS) and a Web Feature Service as features can be edited and shared in addition to map production and distribution via the web. The software offers Web Covering Services (WCS) as well; like the ability to query and filter data. Like most FOSS, support services are offered through email lists, forum channels as well bi-weekly scheduled Skype meeting.

Installation

Before installing and running GeoServer, confirm that your machine has Java Runtime Environment (or SDK). JRE is a backbone for all Java programs and includes the full Java Class library, among other essential tools and functions to properly run GeoServer. Go to http://geoserver.org/download/ to find the appropriate version for your machine. Navigate through the installation wizard; it is recommended that you leave the default port 90 populated, unless you believe this port to already be in use. After launching GeoServer application locally, open a Chrome or Firefox session (you may run into issues with IE) and go to http://localhost:8080/geoserver/web/ . You will need to log in using the credentials you created upon installation. Consider this to be your home page for GeoServer.

Uploading Data

There are a series of steps to store data into GeoServer. Firstly, we must create a *workspace*, located on the left-hand side under Data. This is a container which holds, commonly, similar geospatial data. This is the beginnings to a map project. You can set a default workspace to quickly access a project.

Next, we must create a new *data store* to place in the workspace. You can have multiple data stores in one workplace, or you can house all your data in just one data store. GeoSever will prompt you to declare what type of data you are uploading. There are various type of vector and raster data formats to select. Note that we can upload a directory of shapefiles all at once to a data store. Note that under Connect Parameters, shapefiles must be stored in the same directory that GeoServer lives (usually C:\Program Files (x86)\GeoServer 2.9.1\data_dir).

Lastly, data layers are published to the data store. There is where the actual data comes to life. Here, we declare coordinate and projection systems as well as boundary extends. A list of EPSG codes can be found here.

Viewing Maps

As stated previously, OpenLayers is heavily integrated into GeoServer. Under Data → Layer Preview, we have many format options for viewing data. By choosing OpenLayers, an additional browser will open with your map image. Other viewing formats are KML, GML, PNG, TIFF as well as downloadable zipped shapefiles. After making a viewing request, if you open your GeoServer command prompt, you will first see a getMap request followed by a laundry list of parameters like format (KML, etc.) color (in RGB format), datum, etc.

Editing Styles

One appealing component of working in GeoServer is the ability to add personal unique cartographic design. Under Data → Style we can edit default styles or create originals from scratch. To first familiarize ourselves with a basic template, try generating a default style (like line, point or polygon). Here we will see XML code which builds that particular default style. For example, the default symbol for a point shape file is a red square of size 6. We can either type directly into the text field or upload a file (via a text editing program). This Styled Layer Descriptor (SLD) is a standard format for editing symbology. Please find http://docs.geoserver.org/stable/en/user/styling/sld/cookbook/for GeoServer's SLD Cookbook for further direction. Editing styles is a huge component of producing maps in the GeoServer environment.

Chapter 5: Web Mapping Standards

With so many FOSS GIS platforms out there, it is essential to establish standards for web mapping. Here we will look at one specific organization and their policies aimed to bring together geospatial software tools to modern and consistence standards.

Open Geospatial Consortium

The OGC, a non-profit entity, establishes a level playing field for geospatial organizations (government, private as well as Universities). There are over 500 international entities participating in the OGC. One main concept that comes to mind is data sharing. Another is the convenience of accessibility. Overall OGC standards help people communication more effectively; establishing interoperability standards. Also, although indirectly, organizations like the OGC brings together professionals from various places within the geospatial spectrum; helping to bridge the gap between "tech people" and "geospatial people". The OGC creates geospatial related web standards including WMS, WFS, WCS, Web Map Tile Service (WMTS), Keyhole Markup Language (KML), Global Markup Language (GML), coordinate transformations, GeoAPI and so many more. They also maintain SLD formats discussed in the previous chapter. Here, we will look at WMS, WFS, WCS and WMTS in a little more depth. For downloadable documentation describing all OGC standards, go here.

Web Mapping Service

The OGC first published this standard in 1999 and provides guidance to how geospatial data is published over the web. For example, any WMS server must contain two unique request types; getCapabilities and getMap. In fact, we looked at getMap in Chapter 4 under the 4th section. getCapabilities will return to the user various parameters of the web map. Other optional requests provided by many WMS, although not required by OGC are getLegendGraphic, (GeoServer has a legend option under styles) and describeLayer. Note that a WMS provides a static map image.

Web Feature Service

A WFS defined by OGC standards, allow users to manipulate features (data) components of web maps. Specially, a WFS can query spatial and non-spatial data, create, delete and edit data. Similarly to a WMS, there are a number of functions in which must be imbedded into the platform, including getCapabilities, getFeature, and describeFeatureType.

Web Coverage Service

WMS, WFS and WCS are often grouped together. However, WCS guidelines are slightly different than its two sister services. Here, protocols define web-based retrieval of data in original format. If we want to analyze raw data, we may consider using a WCS. Common requests required for all WCS are getCoverage, describeCoverage and getCapabilities.

Web Mapping Tile Service

WMTS is a comparably recent set of standards defined by OGC in 2010. To review, mapping tiles are individual requests of mapping components, mosaiced together to display one whole map. The advantage of using tiles becomes most obvious when panning and zooming within a web map as many tiles do not have to be “re-rendered”. In sum, displaying data is much quicker as you are limiting the amount of data to be loaded for each change of view/instance. Google Maps was one of the first companies to use this technology. Specially WMTS standards describe how tiles are requested and displayed on your browser.

List of References

http://gdal.org/

http://gis.stackexchange.com/questions/216003/what-does-ogr-stand-for/216004

http://www.osgeo.org/gdal_ogr