*Current Working Version:* ***0.4.7***

Table of Contents

[Overview 2](#_Toc477350408)

[Project Structure 2](#_Toc477350409)

[Version History 2](#_Toc477350410)

[Creating a New Project 3](#_Toc477350411)

[Sub-Projects 3](#_Toc477350412)

[Working with Projects 3](#_Toc477350413)

[Adding Tables to Geocode 3](#_Toc477350414)

[Adding Existing Layers 4](#_Toc477350415)

[Spatially joining Geocoded Dots to Boundaries (e.g. Census Block) 4](#_Toc477350416)

[Styling Layers 4](#_Toc477350417)

[Sorting Layers 4](#_Toc477350418)

[Change Extents (Zoom) for Exported PDFs 4](#_Toc477350419)

[Defining Your Own Zoom Levels 4](#_Toc477350420)

[Run Outputs Multiple Times 4](#_Toc477350421)

[Adding Data in Table to Layout 4](#_Toc477350422)

[Troubleshooting 5](#_Toc477350423)

# Overview

The Mapbuilder project was developed to quickly produce standardized map outputs in PDF formats with a wide range of data, existing layers, and operations. The concept is similar to Data Driven maps in ArcGIS, but provides a more complete workflow process. Mapbuilder not only generates final PDFs, but allows for importing data from raw sources (such as Tab Delimited .txt files), geocoding, sorting layers, and setting up the Legend.

## Project Structure

The project module does most of the heavy lifting.

Testing using Python 2.7.10 install with ArcGIS 10.4.0 and 10.4.1.

Controller has logic. Bootstrap provides per-project customization by overriding MapBuilder’s config class.

# Version History

0.4.0: 8/15/2016 – Very linear, not very reusuable

0.4.3: 8/31/2016 – Major redesign to make components reusable

0.4.4: 9/6/2016 - Major functionality additions, ability to easily plug-and-play various map elements. Paying back some initial technical debt.

0.4.5: 10/17/2016 – Cleaning up code and functionality.

0.4.6: 5/1/2017 – Added ability to load existing layers to MXD and spatially join layers.

0.4.7: In Progress

0.5.0: Coming soon…

# Creating a New Project

Create a new folder with the PROJECT\_NAME value you want. Copy the bootstrap.py and config.py files from the ./Templates directory of the Mapbuilder project into the new project folder you created.

Edit the config.py and set the following values:

* PROJECT\_NAME – Defines the file names for the MXD and GDB files
* PROJECT\_AUTHOR
* PROJECT\_BASE\_PATH – Base path for all files in the project
* PROJECT\_DESCRIPTION
* OUTPUT\_HEADER\_PREFIX
* OUTPUT\_FILE\_PREFIX
* OUTPUT\_MODE
* OUTPUT\_LIST

## Project with Multiple Configurations

A single bootstrap.py can load and process multiple configuration files.

Example of project with sub-projects is: R:\RESEARCH\ArcGIS\Projects\17AY\Dorsey\_Maps\_Multiple\_Programs

This allows each configuration to have slightly different configurations, data sources, styling, etc.

The key is that in the bootstrap.py file needs to import each config file and add it to the configs list. If you use a long list of configs but want to run a quicker test, set the run\_one variable to True.

# Working with Projects

The next few sections give examples of how to achieve specific goals in a project, such as Geocoding addresses, adding existing layers, applying styles, etc.

## Adding Tables to Geocode

**WARNING: Currently this is hardcoded to a specific Geocoder path. I need to fix that, and allow custom paths. WHAT ABOUT THE ONLINE GEOCODER? FRCC Can’t use that for student addresses, but others might…**

In project’s custom config.py. Update name for the actual data file. First time running comment out the geocode\_layer\_style line and the comma in the line above. It’s normal to first just bring in the data, manually style the layer, and save that as a layer file to use for automated styling later on.

# Add Table definitions

TABLES = list()

# Coming soon...

data\_enrollments = {'name': 'GIS\_FWC\_Enrollments\_201630',

'extension': '.txt', # NOT REQUIRED FOR GDB TABLES, omit (I THINK?)

'path': None, # Set to path if not in default data path.

'geocode': True,

'geocode\_layer\_style': 'Style\_CourseEnrollments.lyr'

}

TABLES.append(data\_enrollments)

Behind the scenes, the dictionary above (data\_enrollments) is used to create “Table” objects as defined in MapBuilder.

## Adding Existing Layers

Coming soon…

To add an existing spatial layer, such as TIGER/Line Census Blocks…

Code example (for config.py):

census\_tract = {'name': 'Census Tracts',

'path':'C:\\ArcGIS\\Common Data\\Census\\TLData.gdb\\tl\_2014\_08\_ct'}

LAYERS.append(census\_tract)

## Spatially joining Geocoded Dots to Boundaries (e.g. Census Block)

COMING SOON!!!!

This step automatically adds the layer to the MXD file.

SPATIAL\_JOINS = list()

join\_enrollments\_to\_ct = {'layer\_name': ct\_layer\_name,

'layer\_path': ct\_layer\_path,

'table\_name': data\_table\_name\_geocoded

}

SPATIAL\_JOINS.append(join\_enrollments\_to\_ct)

## Styling Layers

Coming soon…

## Sorting Layers

In a project’s config.py file paste the following code, changing the values as needed for:

* move\_layer\_name
* ref\_layer\_name
* insert\_position

NOTE: Only declare the SORT variable (list) once. To sort multiple layers, just define multiple variables like “layer\_sort” and append them in order.

SORT = list()

layer\_sort = {'move\_layer\_name': 'GIS\_FBC\_Enrollments\_201630\_Geocoded',

'ref\_layer\_name': 'FRCC Locations',

'insert\_position': 'After'}

SORT.append(layer\_sort)

## Change Extents (Zoom) for Exported PDFs

Coming soon…

## Defining Your Own Zoom Levels

Coming soon…

## Run Outputs Multiple Times

I want to be able to do this w/ Different Styles or Different Layers Activated.

Coming soon…

# Adding Data in Table to Layout

Functionality not yet built.

# Troubleshooting

If there’s an error while executing the bootstrapped program, the project variables (where most troubleshooting seems to be needed) can be accessed in the active IDLE Python console. Use: c.prj.<attribute> to access.