Python Map Automation – Beyond the Basics of arcpy.mapping

Jeff Barrette
Jeff Moulds
Basic rules

• Reference an MXD using a path or “current” keyword
  - When using CURRENT
    - Always run in foreground
    - May need to refresh (e.g., RefreshActiveView)

• Uniquely name all the objects you want to reference

• Pre-author MXDs with all possible elements
  - Can’t create new objects (e.g., north arrow, data frames)
  - Author the extra elements off the page
  - No "New Map" function, so keep an empty MXD available

• This is not a replacement for ArcObjects – we are trying to draw a line in the sand

Us
You
“The Line”
Sample Application
Jeff Barrette

http://esriurl.com/5907
What’s new topic

- What’s new ‡ Mapping ‡ What’s new for automating map workflows

Map automation with Python and arcpy.mapping

There were significant additions to arcpy.mapping for the 10.1 release, including:

- Symbol properties for the following renderers can now be automated: graduated colors, graduated symbols, unique values, and classified renderers.
- An Export Report function is available that allows you to automate the generation of reports.
- ArcGIS 10.1 provides access to a LayerList properties to perform analysis over time. You can also enable time on layers.
- The arcpy.mapping module now allows you to update the individual legend items displayed in a Legend item on a page layout by using the UpdateLegend method. You can also remove legend items using the RemoveLegend method.
- Two new functions exist to automate the map service publishing in a Python script. The CreateMapOutput function was introduced to create draft service definition files. Secondly, the AnalyzeResult function has been added to analyze service definition drafts for errors that might prevent publishing.
- ArcGIS provides support for printing WebMaps from the ArcGIS web API. The ConvertWebMapToMapDocument function will convert a WebMap that you intend to print or export to a map document. Once the document is converted, the full state of the WebMap exists in the map document.
- Text elements and graphic elements on a page layout can now be placed,

A detailed list of all additions made at 10.1 is below.
Cloning elements

• You can now clone text and graphic elements
• This allows you to automate things like dynamic tables
• Example:

```python
vertl = arcpy.mapping.ListLayoutElements(
mxd, "GRAPHIC_ELEMENT", "VerticalLine")[0]
vertl.elementPositionX = xPos;
vertl.elementPositionY = 4
vertl.elementHeight = 3
for line in range(1, numColumns+1):
    vert_clone = vertLine.clone("_clone")
    xPos = xPos + colWidth
    vert_clone.elementPositionX = xPos
```
Symbology classes

- **Layer.symbologyType**  `r/o : string`
  - Returns:
    - GRADUATED_COLORS, GRADUATED_SYMBOLS, UNIQUE_VALUES
    - RASTER_CLASSIFIED, OTHER

- **Layer.symbology**  `r/o : Symbology Class`

- **Example:**
  ```python
  if lyr.symbologyType == "GRADUATED_COLORS":
      lyr.symbology.numClasses = 10
      lyr.symbology.valueField = "POP2007"
  ```

- **General notes, can NOT change:**
  - Symbology class
  - Individual symbols
  - Classification methods
    - Use `arcpy.mapping.UpdateLayer`
arcpy.mapping group  http://esriurl.com/5915
Performance tips

• Don’t keep calling list functions
  
  ```python
  import map as arcpy.mapping
  item1 = map.ListLayoutElements(mxd,wildcard="Item1")
  item2 = map.ListLayoutElements(mxd,wildcard="Item2")
  item3 = map.ListLayoutElements(mxd,wildcard="Item3")
  ```

• Call them once instead and iterate through the items
  
  ```python
  for elm in arcpy.mapping.ListLayoutElements(mxd):
      if elm.name =="Item1": item1 = elm
      if elm.name =="Item2": item2 = elm
      if elm.name =="Item3": item3 = elm
  ```
Or even better, use dictionaries

```python
dict = {}
    for elm in arcpy.mapping.ListLayoutElements(mxd):
        dict[elm.name] = elm

    dict["Item1"].text = "Dictionaries"
    dict["Item2"].text = "are really"
    dict["Item3"].text = "COOL!!!"
```
Authoring geoprocessing tasks with Python scripts

• Things to consider help topic:  http://esriurl.com/4656

• Before
  
  ```python
  import arcpy
  path = r"C:\Project\maps"
  ...
  arcpy.mapping.ExportToPDF(mxd, path + "\output.pdf")
  ```

• After
  
  ```python
  import arcpy
  path = arcpy.env.scratchWorkspace
  ...
  arcpy.mapping.ExportToPDF(mxd, path + "\output.pdf")
  arcpy.SetParameterAsText(1,path + "\output.pdf")
  ```
Functions in 10.1 for server publishing and printing

- **ConvertWebMapToMapDocument()**
  - Use with the ArcGIS web APIs for advanced web map printing workflows

- **CreateMapSDDraft()**
  - Automate publishing map documents to map services
Server printing out-of-the-box

- ArcGIS Server 10.1 and the ArcGIS web APIs support web map printing via print services.
  - Out-of-the-box print service and template maps ship with Server
  - Print services sample: http://esriurl.com/6465

Related Session: Supporting High-Quality Printing in Web Applications with ArcGIS for Server (Wednesday @ 8:30 am Room 15A)
Advanced server printing with arcpy.mapping

- Build web apps with customized versions of the 10.1 out-of-the-box print service

New arcpy.mapping method for converting Web Maps to MapDocuments:

- `ConvertWebMapToMapDocument ()`

- `ConvertWebMapToMapDocument (webmap_json, {template_mxd}, {notes_gdb}, {extra_conversion_options})`
Advanced server printing with arcpy.mapping

- Full capabilities of arcpy.mapping on the document
  - Modify content (e.g. replace service layers with local vector data for high quality printing)
  - Export using custom options
  - Export data driven pages
  - Export to PDF and insert additional pages
- Return a printer-friendly output file (PDF, PNG, etc.)
- Online help and examples http://esriurl.com/4600
Demo: High quality server printing with arcpy.mapping

- Print high quality vector layers instead of tiled image cache of service layers
  - Vector layers will be staged in template map documents
Demo: High quality server printing with arcpy.mapping

- Reference the custom arcpy.mapping based GP service

```javascript
var printUrl = "http://gilbert:6000/arcgis/rest/services/Austin/AustinPrint/GPServer/AustinPrint";
printTask = new esri.tasks PrintTask(printUrl, { async: true });
```
**Demo: High quality server printing with arcpy.mapping**

Arcpy.mapping code used in custom geoprocessing service

```python
import arcpy, os, uuid

# Input web map json
Web_Map_as_JSON = arcpy.GetParameterAsText(0)

# Input layout template
Layout_Template = arcpy.GetParameterAsText(1)

templatePath = '/gilbert/Austin/Templates'
templateMxd = os.path.join(templatePath, Layout_Template + '.mxd')

# Convert the web map to a map document
result = arcpy.mapping(ConvertWebMapToMapDocument(Web_Map_as_JSON, templateMxd)
mxd = result.mapDocument

df = arcpy.mapping.ListDataFrames(mxd, 'Webmap')[0]

# Remove the service layer
# This will leave the staged vector layers from the template
for lyr in arcpy.mapping.ListLayers(mxd, data_frame=df):
    if lyr.isServiceLayer:
        arcpy.mapping.RemoveLayer(df, lyr)

# Export the web map to PDF
output = 'WebMap_({}).pdf'.format(str(uuid.uuid1()))
Output_File = os.path.join(arcpy.env.scratchFolder, output)
arcpy.mapping.ExportToPDF(mxd, Output_File)

# Set the output parameter to be the output file of the server job
arcpy.SetParameterAsText(2, Output_File)
```
High quality server printing with arcpy.mapping

- Two tutorials:
  - Basic high-quality web map printing: [http://esriurl.com/4601](http://esriurl.com/4601)
  - Advanced web map printing: [http://esriurl.com/4602](http://esriurl.com/4602)
Publishing map services with arcpy.mapping

- arcpy.mapping CreateMapSDDraft()
- Workflow from map document to map service
- Use python for:
  - Publishing automated analysis results
  - Scheduled service upgrades
  - Batch migration from 10.0 to 10.1
Publishing map services with arcpy.mapping

Sample: `CreateMapSDDraft (arcpy.mapping)`

```python
# define local variables
wrkspc = 'C:/Project/
mapDoc = arcpy.mapping.MapDocument(wrkspc + 'counties.mxd' )
con = 'GIS Servers/arcgis on MyServer_6080 (publisher).ags'
service = 'Counties'
sddraft = wrkspc + service + '.sddraft'
sd = wrkspc + service + '.sd'
summary = 'Population Density by County'
tags = 'county, counties, population, density, census'

# create service definition draft
arcpy.mapping.CreateMapSDDraft(mapDoc, sddraft, service, 'ARCGIS_SERVER',
                                 con, True, None, summary, tags)

# analyze the service definition draft
analysis = arcpy.mapping.AnalyzeForSD(sddraft)

# stage and upload the service if the sddraft analysis did not contain errors
if analysis['errors'] == {}:
    # Execute StageService
    arcpy.StageService_server(sddraft, sd)
    # Execute UploadServiceDefinition
    arcpy.UploadServiceDefinition_server(sd, con)
else:
    # if the sddraft analysis contained errors, display them
    print analysis['errors']
```

Online help: [http://esriurl.com/4598](http://esriurl.com/4598)
Publishing other service types with python

• 10.1:
  - `arcpy.CreateMapSDDraft()`

• 10.1 sp1:
  - `arcpy.CreateGPSDDraft()`
    - Create geoprocessing services with Python
  - `arcpy.CreateImageSDDraft()`
    - Create image services with Python

• 10.2:
  - `arcpy.CreateGeocodeSDDraft()`
    - Create geocoding services with Python
Related sessions

Tuesday:
10:15 - 11:30am : Introduction to arcpy.mapping : Room 8

Wednesday:
8:30 – 9:45am : Supporting High-Quality Printing with AGS : Room 15A
10:15 - 11:30am : Introduction to arcpy.mapping : Room 1 A/B
3:15 - 4:30pm : Beyond the Basics of arcpy.mapping : Room 1 A/B

Thursday:
8:30 - 09:45am : Beyond the Basics of arcpy.mapping : Room 8
Thank you…

Please fill out the session evaluation

*First Offering ID: 1298*

*Second Offering ID: 1220*

**Online** – www.esri.com/ucsessionssurveys

**Paper** – pick up and put in drop box