Advance Map Automation With Python

Jeff Barrette

Jeff Moulds

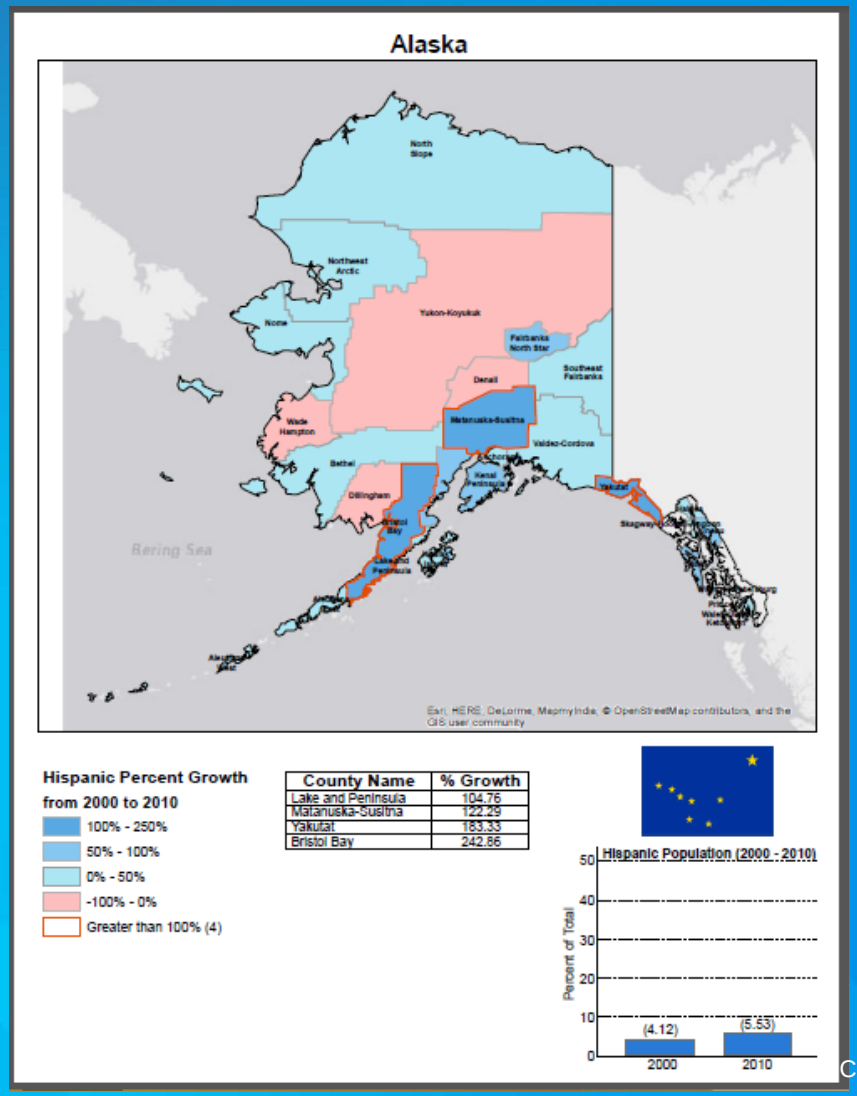
arcpy.(m)a(p)ping

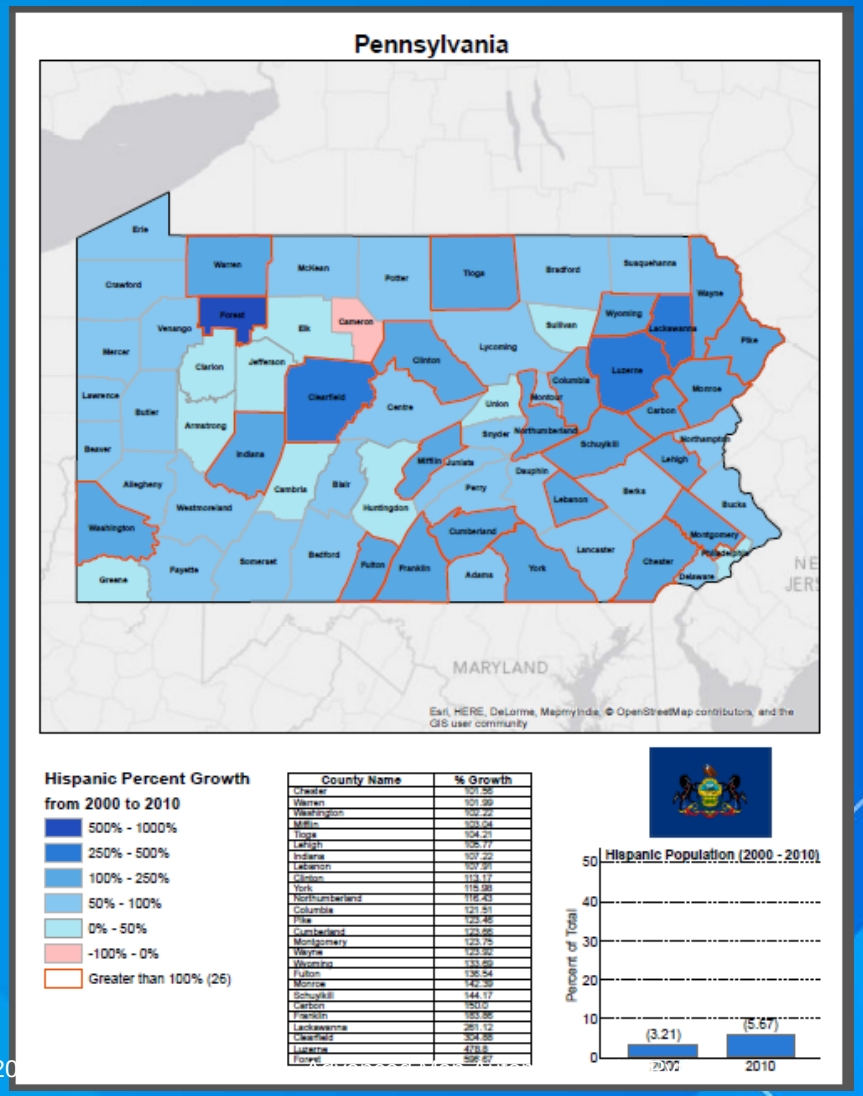
http://esriurl.com/8899

DDPwithDynamicTablesAndGraphs_10.1_v1

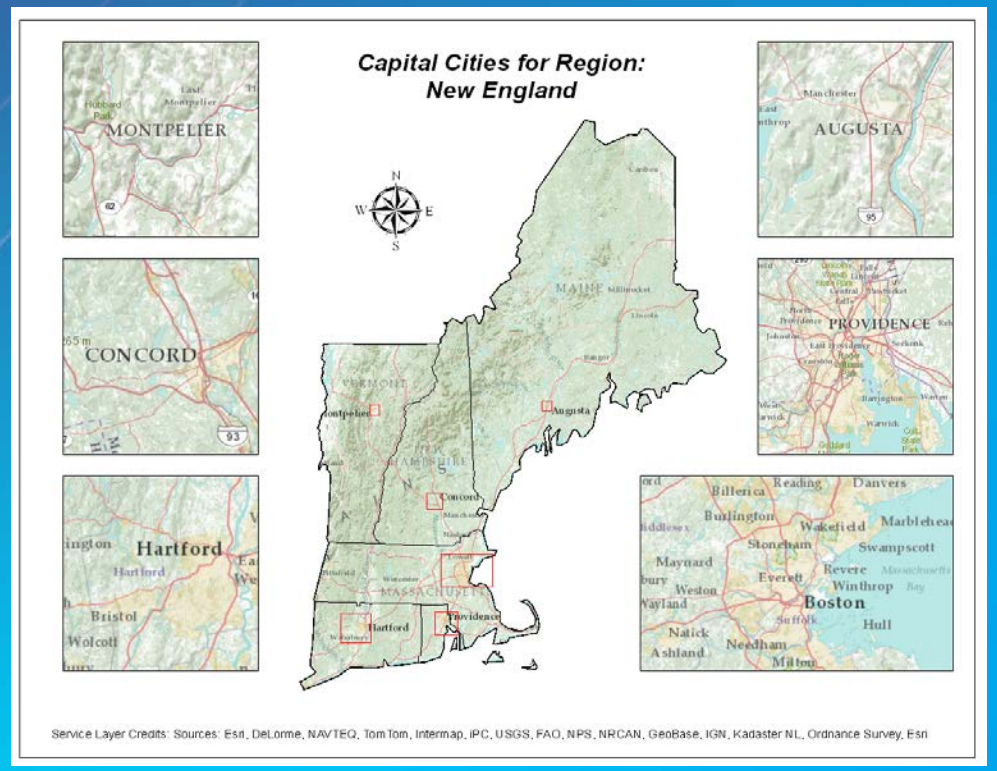
Header
Cell

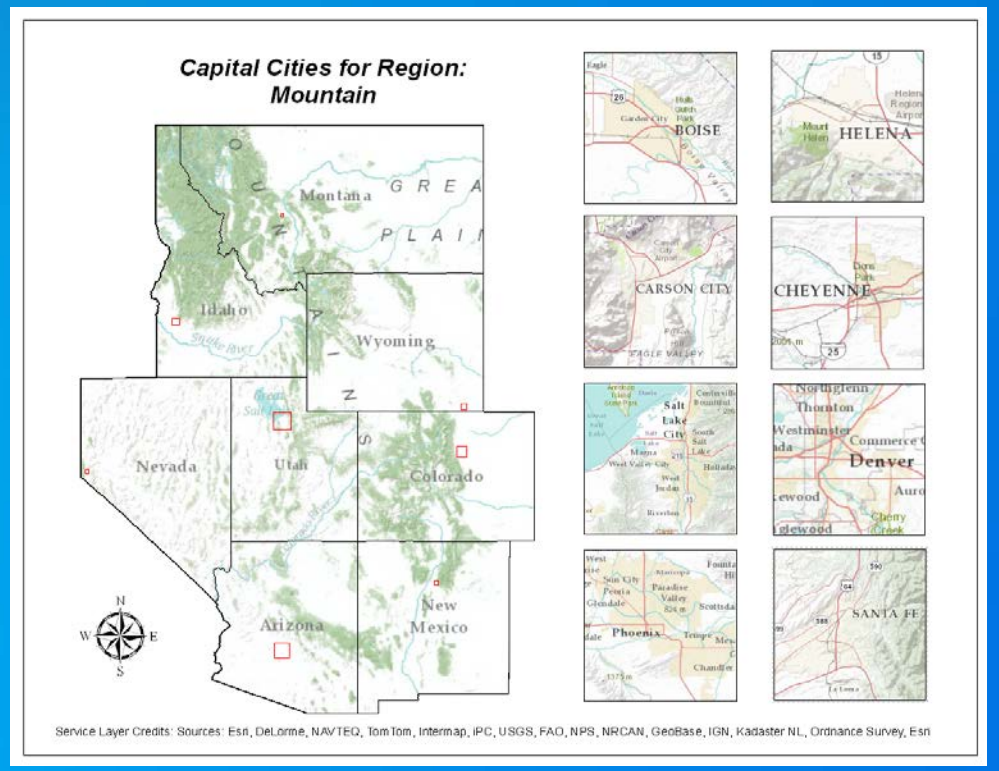
County Name	% Growth
Chester	101.56
Warren	101.99
Washington	102.22
Mifflin	103.04
Tioga	104.21
Lehigh	105.77
Indiana	107.22
Lebanon	107.91
Clinton	113.17
York	115.98
Northumberland	116.43
Columbia	121.51
Pike	123.46





MultipleElementLayoutManager_10.0_v1

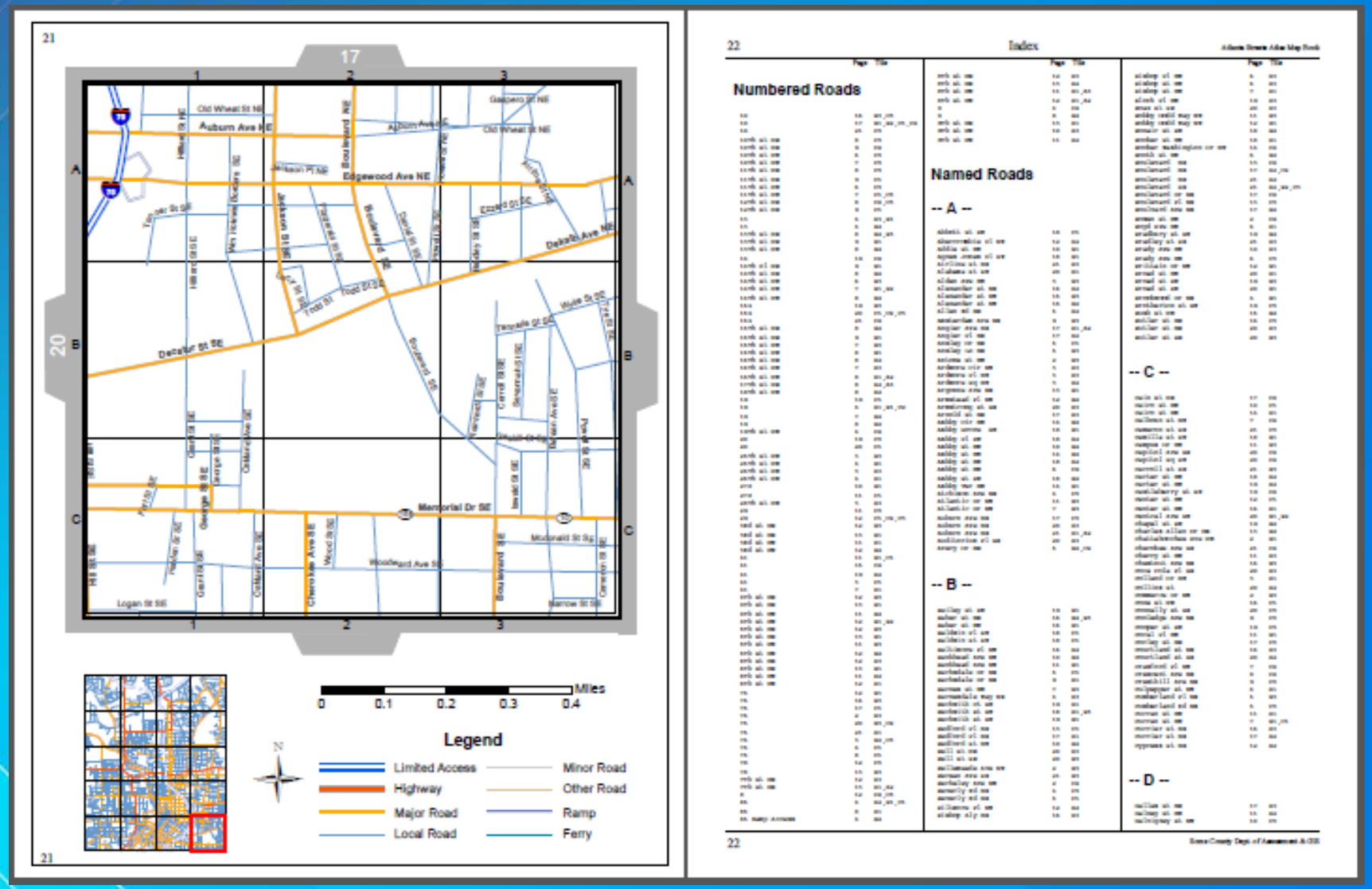




PageLayoutElements

OB	Name	MainDF	Inset1	Inset2	Inset3	Inset4	Inset5	Inset6	Inset7	Inset8	Title_NorthArrow
1	Default Template	[0.25,0.25,10.5,8.0,-4872376,15442576,-3658297,16367588,5000000]	[-2.0,6.5,1.7	[-2.0,4.5,1	[-2.0,2.5,1.	[-2.0,0.5,1.	[11.25,6.5,	[11.25,4.5,1.	[11.25,2.5,1.	[11.25,0.5,	[5.5, 7.75, 10, 1]
2	East North Central	[0.25,0.25,10.5,8.0,-10999471,4322861,-8332466,6354865,10000000]	[7.81,5.19,3.	[0.3,5.77,	[0.3,0.77,2.	[0.3,3.25,2.	[8.51,0.71,	<Null>	<Null>	<Null>	[5.29,7.85,9.61,4.18]
3	East South Central	[0.25,0.25,10.5,8.0,-10704277,3398170,-8704023,4922173,7500000]	[7.39,0.79,3.	[0.3,5.77,	[0.3,0.77,2.	[0.3,3.25,2.	<Null>	<Null>	<Null>	<Null>	[6.72,7.84,9.72,6.78]
4	Middle Atlantic	[0.25,0.25,10.5,8.0,-9355063,4655409,-8021560,5671411,5000000]	[0.5,5.87,2.2	[0.53,3.26	[0.53,0.64,	<Null>	<Null>	<Null>	<Null>	<Null>	[5.49,7.78,9.75,1.49]
5	Mountain	[0.25,0.25,10.5,8.0,-13512281,3592054,-9511773,6640060,15000000]	[6.47,6.38,1.	[6.47,4.49	[6.47,2.55,	[6.47,0.62,	[8.63,6.4,1.	[8.63,4.47,1.	[8.65,2.54,1.	[8.66,0.63,	[3.46,7.79,1.09,1.37]
6	New England	[0.25,0.25,10.5,8.0,-8659058,4932580,-7058854,6151782,6000000]	[0.5,6.0,2.25	[0.5,3.5,2.	[0.5,1.0,2.2	[8.5,6.0,2.2	[8.5,3.5,2.2	[7.15,1.0,3.6	<Null>	<Null>	[5.55,7.85,4.31,6.3]
7	Pacific	[0.25,0.25,10.5,8.0,-19482949,1675259,-6147923,11835279,50000000]	[8.5,6.0,2.25	[8.5,3.5,2.	[8.5,1.0,2.2	[4.87,4.8,2.	[1.13,1.39,	<Null>	<Null>	<Null>	[5.81,7.79,7.01,2.94]
8	South Atlantic	[0.25,0.25,10.5,8.0,-10362617,2812844,-7502229,4992187,10725927]	[0.5,6.5,3.25	[0.5,4.5,2.	[0.5,2.5,1.7	[0.5,0.5,2.	[8.75,6.5,1.	[8.75,4.5,1.7	[8.75,2.5,1.7	[8.75,0.5,1.	[6.93,3.15,7.15,1.88]

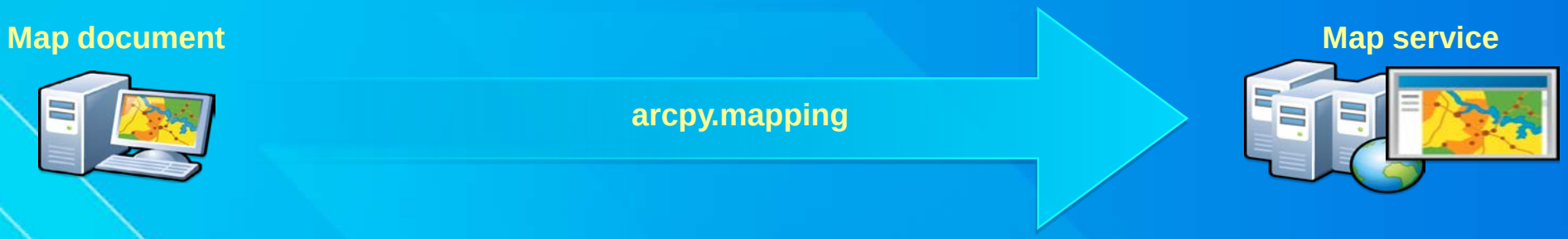
GenerateMapBookWithIndexPages_10_v2



Functions for web map printing and server publishing

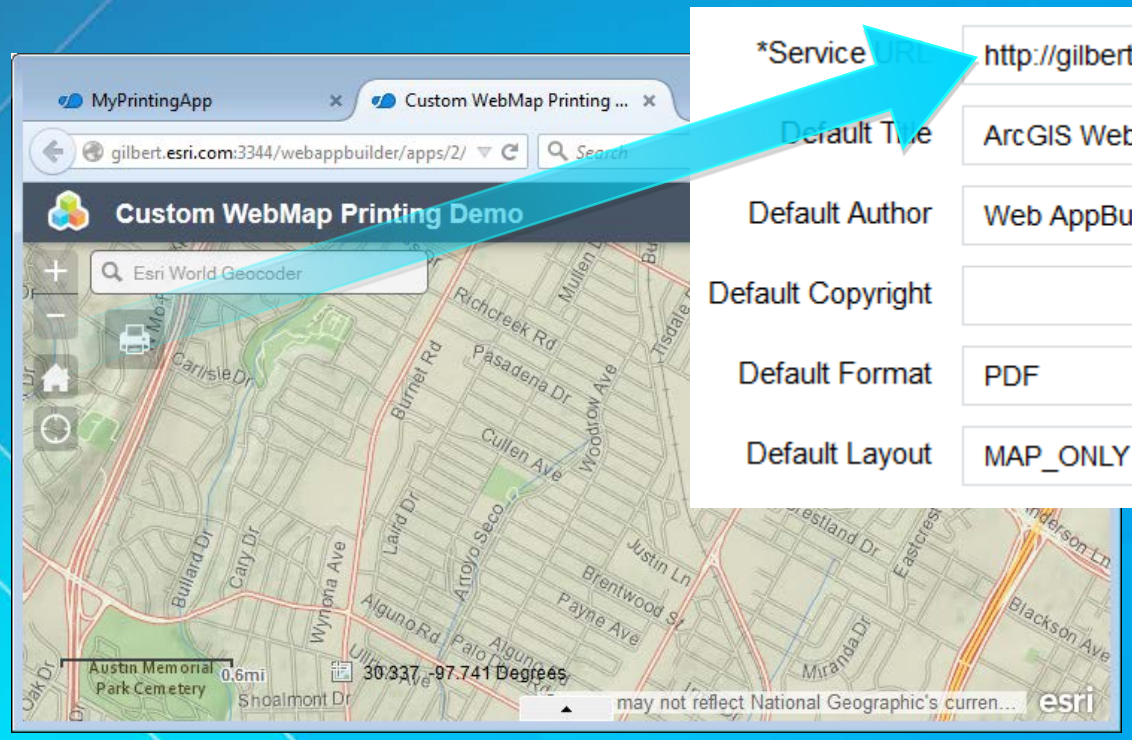
- Advanced Web Map Printing workflows

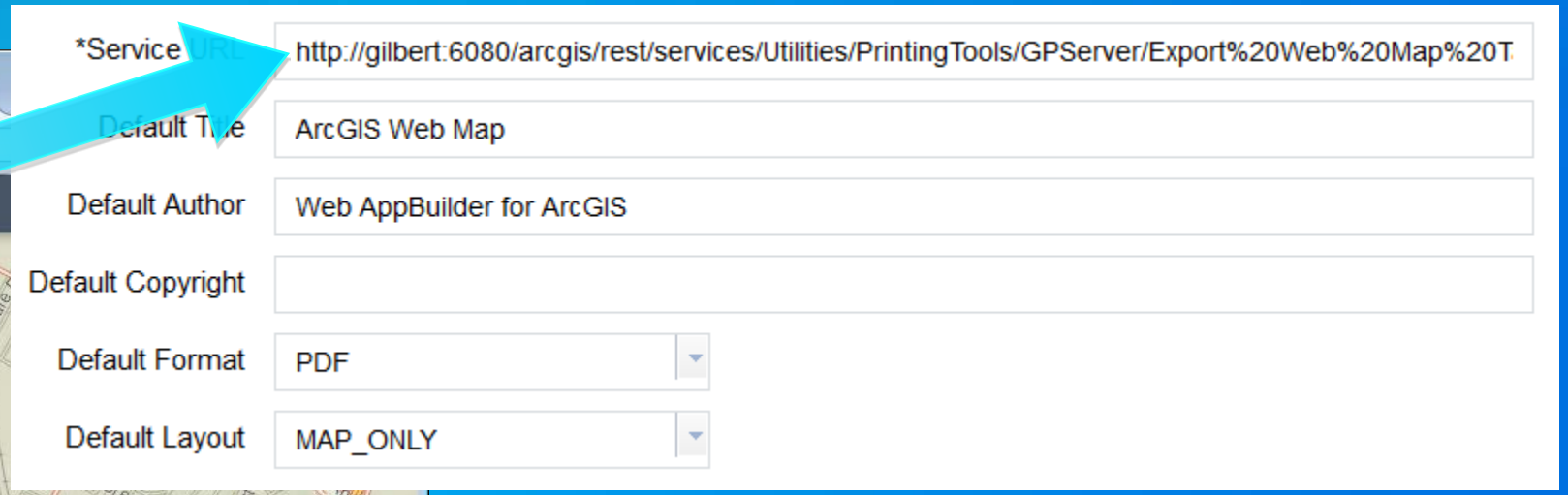
- Automate publishing map documents to map services



Server printing out-of-the-box

- **ArcGIS Server, Web App Builder and the ArcGIS web APIs support web map printing.**
  - **Out-of-the-box print service and template maps ship with Server**
  - **Hosted print service available via ArcGIS Online**
  - **Easy to use in the Web App Builder or the ArcGIS web APIs**





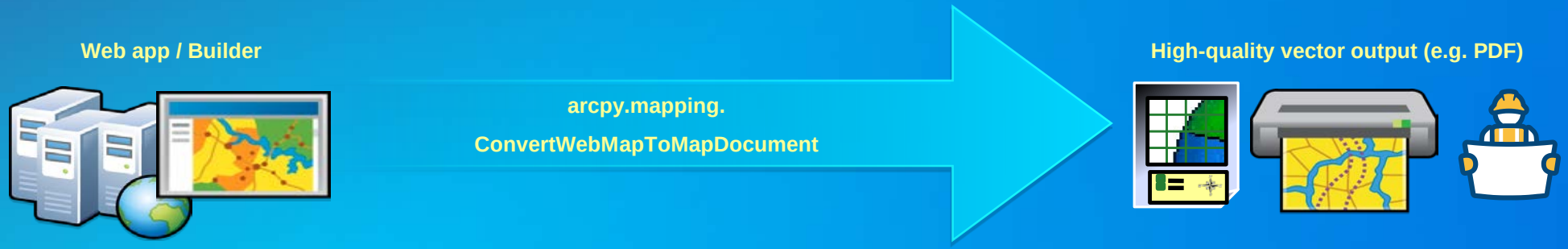
Related Session: Enabling High-Quality Printing in Web Applications (Wednesday @ 10:15am – 11:30pm Room 3)
Also search your agenda for "Web App Builder" – many sessions!

Advanced server printing with arcpy.mapping

- **Full capabilities of arcpy.mapping:**
  - **Swap out service layers for local vector data for *vector PDF output***
  - **Export using advanced options**
  - **Export data driven pages**
  - **Export to PDF and insert additional pages (title page, reports, etc.)**
  - **Controlling the appearance of the legend**
  - **Etc.**
- **Return a printer-friendly output file (PDF, PNG, etc.)**

Advanced server printing with arcpy.mapping

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



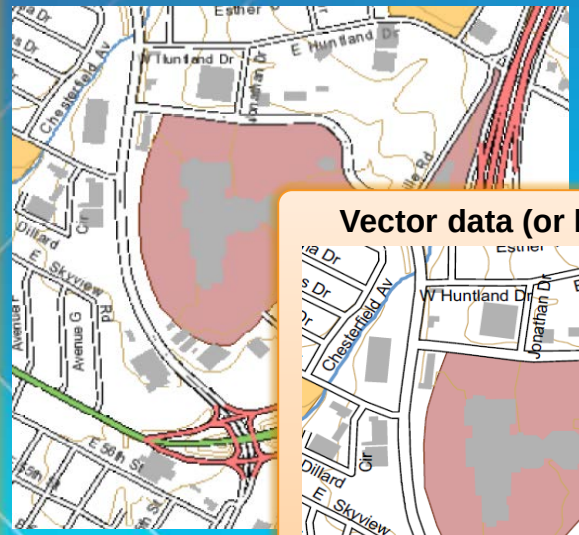
- **arcpy.mapping method for converting Web Maps to Map Documents:**
  - `ConvertWebMapToMapDocument (webmap_json, {template_mxd}, {notes_gdb}, {extra_conversion_options})`

Online help and examples http://esriurl.com/4600

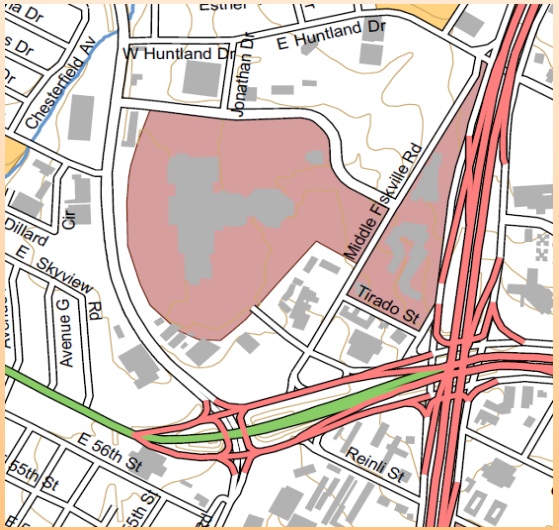
Demo: Web app to export vector PDF using arcpy.mapping

- Output or print vector layers instead of “flat” image of service layers
  - Vector layers will be staged in template map document

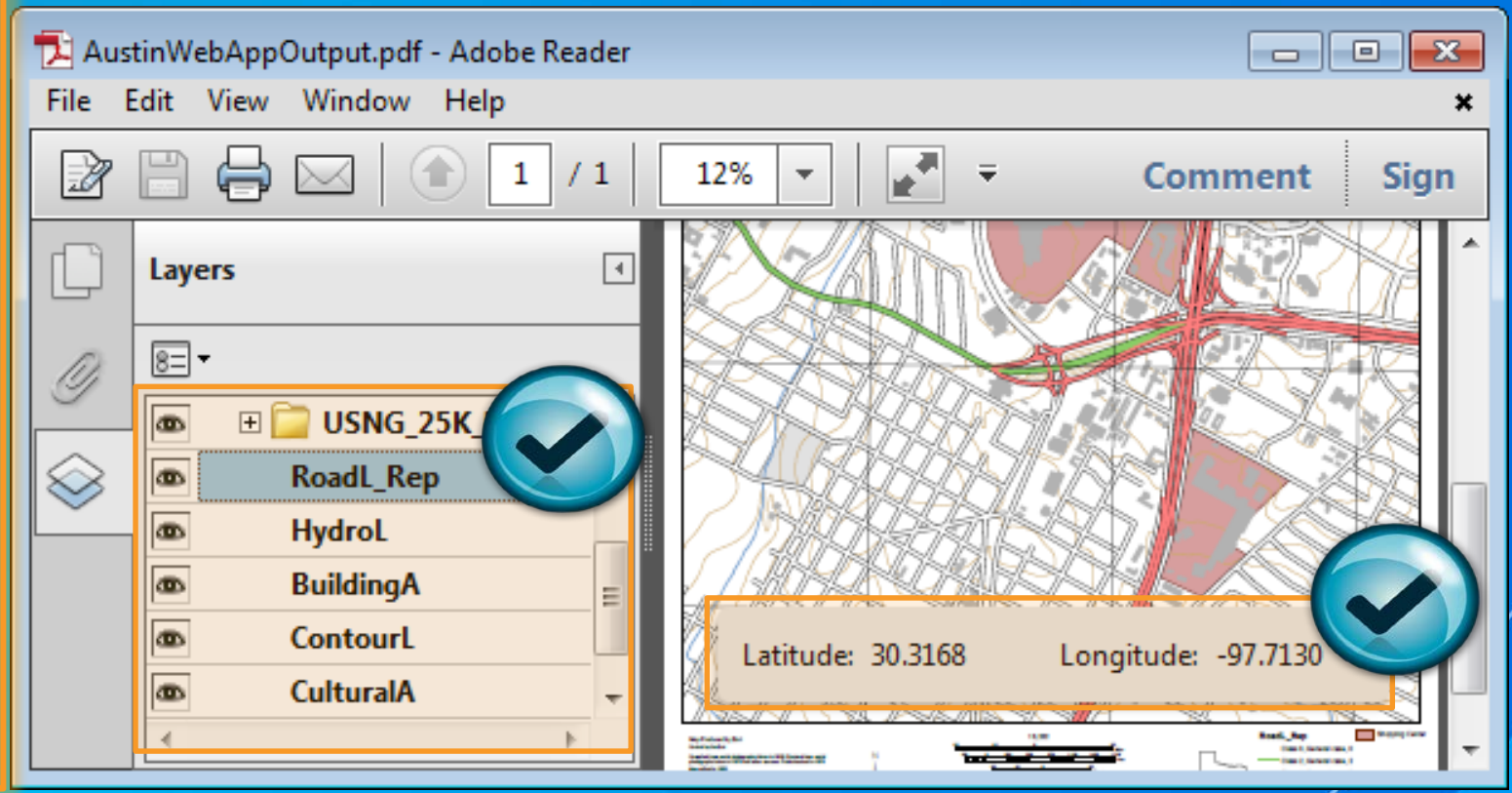
Map service tiled cache (low dpi)



Vector data (or high dpi image)



Output PDF viewed in Adobe Reader



Demo: Web app to export vector PDF using arcpy.mapping

Python code used in custom GP service

Get web map JSON

Get template MXD

Create new MXD based on web map

Remove service layers

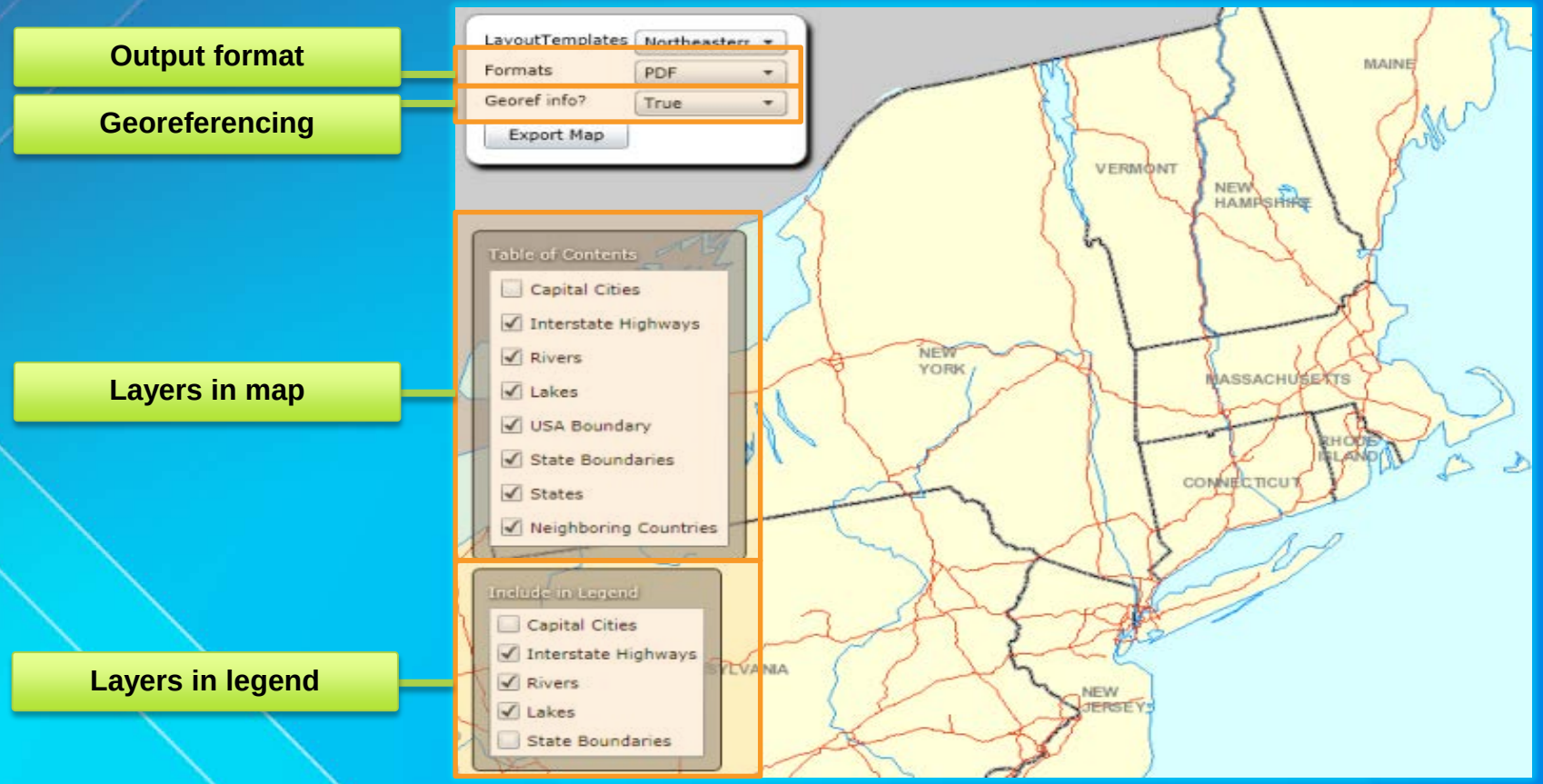
Export PDF

Output file of job

```
import arcpy, os, uuid

# Input WebMap json
Web_Map_as_JSON = arcpy.GetParameterAsText(0)

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

# The template location in the server registered folder
templatePath = '//gilbert/Austin/Templates'
templateMxd = os.path.join(templatePath, Layout_Template + '.mxd')

# Convert the WebMap 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 just leave the 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, georef_info=True)

# Set the output parameter to be the output file of the server job
arcpy.SetParameterAsText(3, Output File)
```

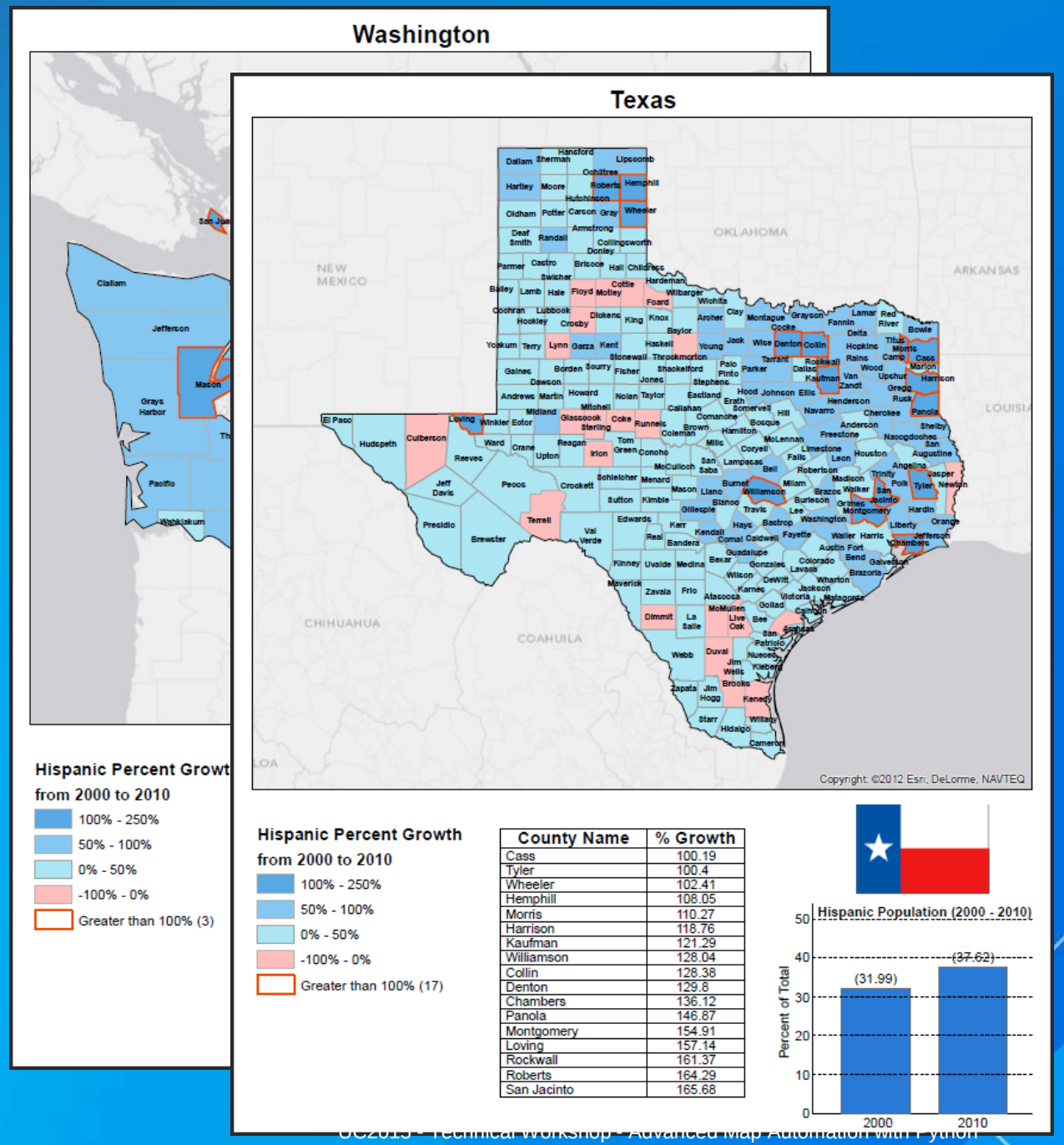
Web app to export vector PDF using arcpy.mapping

- **Two tutorials in the help:**
  - **Basic vector web map printing: http://esriurl.com/4601**
  - **Advanced web map printing: http://esriurl.com/4602**



Advanced Server Printing

- Modify arcpy.mapping scripts authored on Desktop and use them in geoprocesing and print services



County Name	% Growth
Cass	100.19
Tyler	100.4
Wheeler	102.41
Hemphill	108.05
Morris	110.27
Harrison	118.76
Kaufman	121.29
Williamson	128.04
Collin	128.38
Denton	129.8
Chambers	136.12
Panola	146.87
Montgomery	154.91
Loving	157.14
Rockwall	161.37
Roberts	164.29
San Jacinto	165.68

Advanced Server Printing: new function at 10.3

- **Layer.UpdateLayerFromJSON(json_layer_definition)**
  - **Used in web map printing applications that support changing the renderer (or other properties) of dynamic web service layers.**
  - **Will apply the renderer (or other layer properties) as specified in the webmap_json to the corresponding vector layers staged in the template map document.**

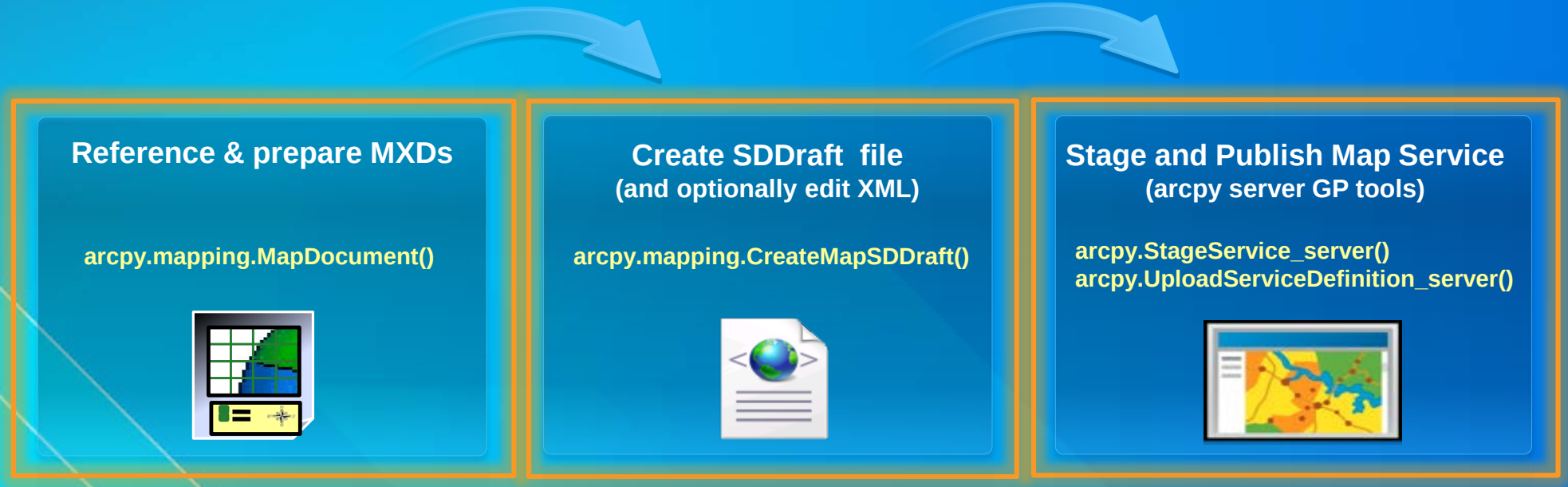
Get JSON Layer Definition from web map

Update vector layer from JSON

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

# Reference the data frame that contains the web map
df = arcpy.mapping.ListDataFrames(mxd, 'Webmap')[0]

# Reference the staged vector data that corresponds to the dynamic layer in the JSON
# This is the layer that will get updated based on the layer definition in the JSON
lyr = arcpy.mapping.ListLayers(mxd, "U.S. States (Generalized)", df)[0]

# Read the JSON and extract the layer definition
# In this case we have hardcoded it to second operational layer
data = json.loads(Web_Map_as_JSON)
layerDefinition = data["operationalLayers"][1]["layerDefinition"]

# Update the staged vector layer with the layer definition (e.g. renderer info) from the JSON
lyr.updateLayerFromJSON(layerDefinition)
```

Publishing map services with arcpy.mapping

- arcpy.mapping.CreateMapSDDraft(map_document, out_sddraft, service_name, {server_type}, {connection_file_path}, {copy_data_to_server}, {folder_name}, {summary}, {tags})
- Workflow to convert map document to map service.
- Use python scripts for:
  - Scheduled service updates. E.g. nightly.
  - Publishing automated analysis results.
  - Batch publishing.



Publishing map services with arcpy.mapping

Sample script: CreateMapSDDraft

- **Reference MXD**
- **Server connection, service properties, etc.**
- **Create and analyze sddraft for errors, warnings, etc.**
- **Stage and publish Map Service**
- **Don't publish if errors exist**

```
import arcpy

# 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 and samples: http://esriurl.com/4598

Publish and overwrite a feature service on ArcGIS.com:
http://blogs.esri.com/esri/arcgis/2014/01/24/updating-your-hosted-feature-service-for-10-2

Publishing other service types with python

- Create geoprocessing services
  - `arcpy.CreateGPSDDraft()`
- Create image services
  - `arcpy.CreateImageSDDraft()`
- Create geocoding services
  - `arcpy.CreateGeocodeSDDraft()`

Pro arcpy.mp – major changes

- arcpy.mapping renamed to arcpy.mp
- MapDocument()is now ArcGISProject()
- Fewer root level functions, more OO design
- DataFrame is replaced with:
  - Camera, MapFrame, Map
- LayerFiles work differently
- A new Layout object
- Updating data sources is very different
  * Stand-alone scripts don't use the cache

ArcGIS Pro arcpy.mp demonstration

http://esriurl.com/8948

Function for importing 10.x documents into ArcGIS Projects

- ArcGISProject.importDocument (document_path, {include_layout})

Looping through MXDs in a folder.

Reference a template APRX. Import MXD into the APRX. Save the project.

```
1  import arcpy, os
2  folder = r"C:\Project"
3  for file in [f for f in os.listdir(os.path.join(folder, 'MXDs'))
4               if os.path.splitext(f)[1].lower() == '.mxd']:
5      aprx = arcpy.mp.ArcGISProject(os.path.join(folder, 'template.aprx'))
6      aprx.importDocument(os.path.join(folder, file))
7      aprx.saveACopy(os.path.join(folder, 'APRXs', os.path.splitext(file)[0] + '.aprx'))
8      del aprx
```

Function for importing 10.x documents into ArcGIS Projects

- ArcGISProject.importDocument(document_path, {include_layout})


```
1  aprx = arcpy.mp.ArcGISProject("CURRENT")
2  aprx.importDocument(r"C:\Project\Demo\Mexico.mxd", include_layout=True)
3  aprx.importDocument(r"C:\Project\CentralColorado.mxd", include_layout=False)
4  aprx.importDocument(r"C:\Project\Yosemite.3dd")
5  aprx.importDocument(r"C:\Project\Structured.sxd")
```


DEMO

Updating Data Sources – improved usability at Pro

Project/Map/Layer/Table/LayerFile.updateConnectionProperties (current_connection_info, new_connection_info, {auto_update_joins_and_relates}, {validate})

Find this path:

```
aprx = arcpy.mp.ArcGISProject(r'C:\Projects\YosemiteNP\Yosemite.aprx')
aprx.updateConnectionProperties(r'C:\Projects\YosemiteNP\Data\Yosemite.gdb',
                                r'C:\Projects\YosemiteNP\Vector_Data\Yosemite.gdb')
```

Replace it with this path:

Updating Data Sources – improved usability at Pro

1. **Changing a folder**

```
aprx = arcpy.mp.ArcGISProject(r'C:\Projects\YosemiteNP\Yosemite.aprx')
aprx.updateConnectionProperties('Data','Vector_Data')
```

2. **Changing FGDB to SDE**

```
aprx = arcpy.mp.ArcGISProject(r'C:\Projects\YosemiteNP\Yosemite.aprx')
aprx.updateConnectionProperties(r'C:\Projects\YosemiteNP\Vector_Data\Yosemite.gdb',
                                r'C:\Projects\YosemiteNP\DBConnections\Server.sde')
```

3. **Changing PGDB to FGDB**

```
aprx = arcpy.mp.ArcGISProject(r'C:\Projects\YosemiteNP\Yosemite.aprx')
m = aprx.ListMaps("Yose*")[0]
m.updateConnectionProperties(r'Background.mdb', 'Background_fGDB.gdb')
```

Updating Data Sources advanced concepts – Layer.connectionProperties

- New at Pro
- The entire layer data source object model is exposed as a Python dictionary.
- Use if you need more fine grained control that what is available in Project/Map/Layer/Table/LayerFile.updateConnectionproperties()

Access a layer in a map.

Get layer's connection properties.

File Geodatabase layer connection properties dictionary

SDE Geodatabase layer connection properties dictionary

```
import arcpy, pprint
p = arcpy.mp.ArcGISProject('current')
m = p.listMaps()[0]
l = p.listLayers()[0]
pprint.pprint(l.connectionProperties)
```

```
{'connection_info': {'database': 'C:\\Projects\\YosemiteNP\\Data\\Yosemite.gdb'},
 'dataset': 'RangerStations',
 'workspace_factory': 'File Geodatabase'}
```

```
{'connection_info': {'authentication_mode': 'DBMS',
                     'database': 'gis',
                     'db_connection_properties': 'Esri',
                     'dbclient': 'sqlserver',
                     'instance': 'sde:sqlserver:Esri',
                     'password': '**********',
                     'server': 'gis',
                     'user': 'arcpy',
                     'version': 'sde.DEFAULT'},
'dataset': 'gis.arcpy.RangerStations',
'workspace_factory': 'SDE'}
```

Two ways to use the connection properties dictionary

1. **Write directly to the dictionary**

Get layer connection properties

Update dictionary

Set layer connection properties

```
# change the sde username and password
# ====================================
oldUser = 'Robbie'
newUser = 'Sly'
password = 'Dunbar'
# get the layer's connection properties
conProps = sdeLayer.connectionProperties
conProps['connection_info']['user'] = newUser
conProps['connection_info']['password'] = password
conProps['dataset'] = sdeLayer.name.replace(oldUser, newUser)
# apply the new connection properties
sdeLayer.connectionProperties = conProps
```

2. **UpdateConnectionProperties will also do find and replace for full and partial dictionaries**

Old database info

New database info

Set layer connection properties

```
# change the sde server
# =====================
newDatabase = 'Shakespeare'

# old database to search for
old_dict = {'connection_info': {'db_connection_properties': 'esri4',
                                'instance': 'sde:sqlserver:esri4',
                                'server': 'esri4',
                                'version': 'sde.DEFAULT'}}

# new database to replace with
new_dict = {'connection_info': {'db_connection_properties': newDatabase,
                                'instance': 'sde:sqlserver:' + newDatabase,
                                'server': newDatabase,
                                'version': 'dbo.DEFAULT'}}

# update the data sources by doing a find and replace on the project
aprx.updateConnectionProperties(old_dict, new_dict, True, True)
```

Updating Data Sources – demo

Updating Data Sources – demo source code

```
# part 1 - update from PGDB to FGDB
# ================================

aprx.updateConnectionProperties('PGDB.mdb', 'FGDB.gdb', True)

# part 2 - view layer's conection properties
# =========================================

import pprint
mexLyr = aprx.listMaps('Layers')[0].listLayers('mex_eco')[0]
conProps = mexLyr.connectionProperties
pprint.pprint(conProps)

# part 3 - update join info in layer's connection properties
# ==========================================================

conProps['foreign_key'] = 'ECO_CODE_NEW'
mexLyr.connectionProperties = conProps
```

Understanding our world.