# How to Web scrape ArcGIS Server API

Created by Annie Nguyen

2/5/20

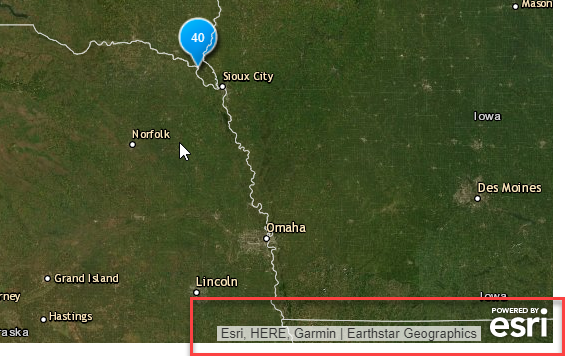
## Part 1. Identify you Data

Got an interactive map but can’t download the dataset without having to click on every individual point? If you need to scrape spatial data that’s pulled from an ArcGIS Server, then you’re in luck! This document is for you. If not, then you might need to look elsewhere to figure out how to scrape your data unfortunately ☹

The example used in this tutorial is using the observation wells in South Dakota using this link:

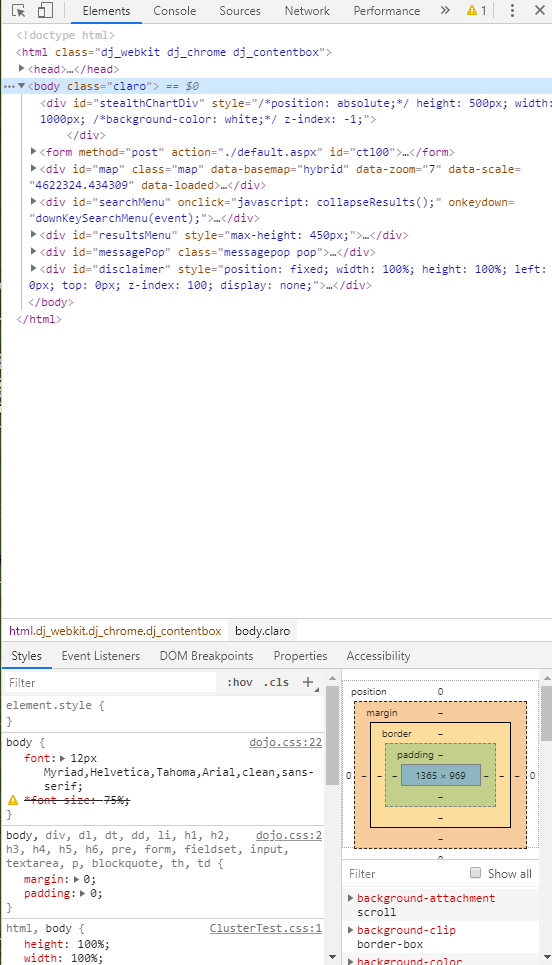
<https://apps.sd.gov/nr69obswell/default.aspx>

One way to figure out if the data you want is using ArcGIS is to look at the bottom of your interactive map:

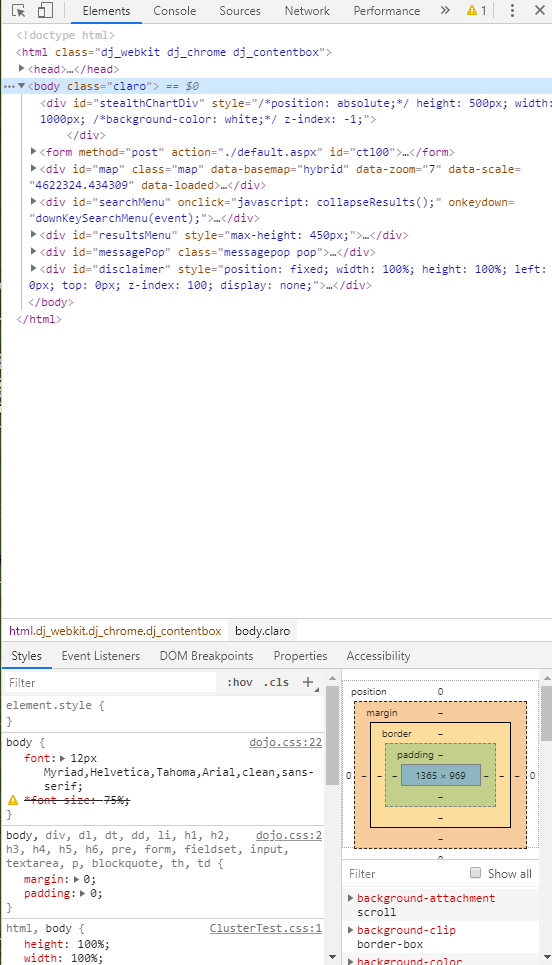


## Part 2. Getting to the treasure trove

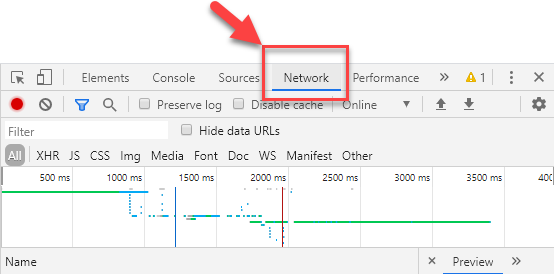
1. So now that you’ve identified that your data uses ArcGIS, go ahead to press **Ctrl + Shift + I** on your keys at the same time. This brings of the developer tools for the website:



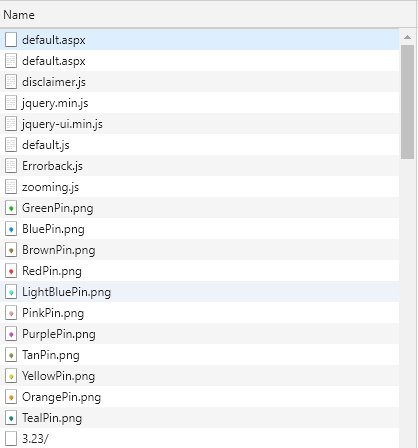
1. This is where you can look at how each element of the website is constructed using HTML (i.e. the map, the title banner, the legend, etc.) Try hovering over some of these elements and it will highlight what it’s referring to on the actual website.



1. For our purposes, we’re going to need to go to the network tab on the top here:

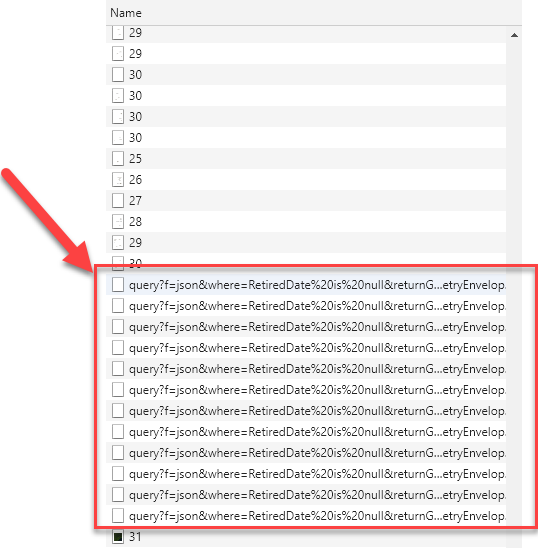


Go ahead to refresh the webpage—the developer tool panel will still be there, but now there’s a bunch of things populating the network catalog here:

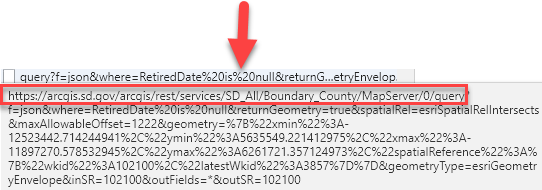


These are the objects that are created and pulled into the map—see how each pin that marks our map is actually just a .png image and that some things like disclaimer are created using javascript (.js). You can click on each one and see the preview to see what they look like. We’re going to scroll further down to where the actual data of this map lies.

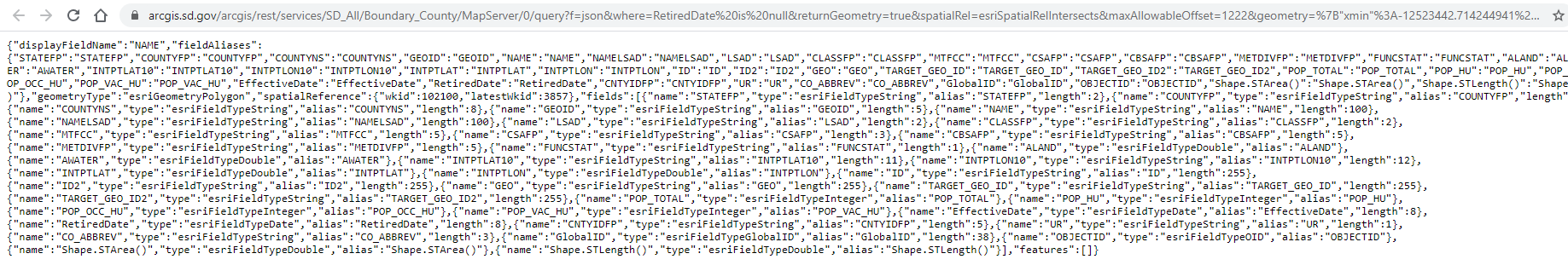
1. If you keep scrolling, you’ll come across something that begins with “query…”



This is where the website actually queries from the API (the ArcGIS server), and pulls data from this location. This means the data is stored not on this site but on a separate location. If you hover over some of these and take a look at the url:



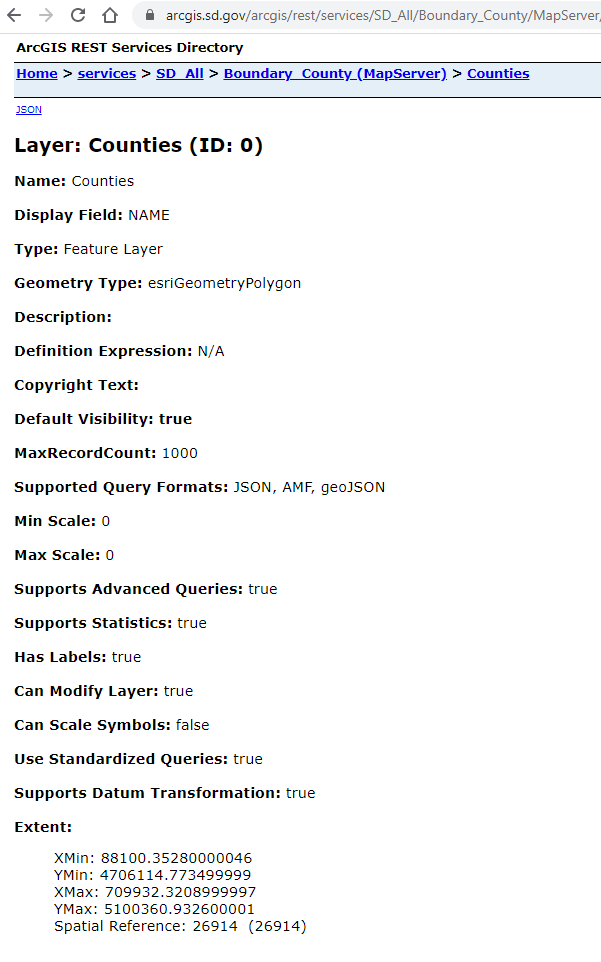
1. It can give you an idea of what it’s querying. Here it looks like this is where it pulls the county data and boundary shapefiles to create the map. Double-click on this link and it should bring you to something like this:



1. It’s pretty confusing looking, but this is all the data stored as javascript. Try going to the url and clear everything after “…MapServer/0/” so that it now looks like this:

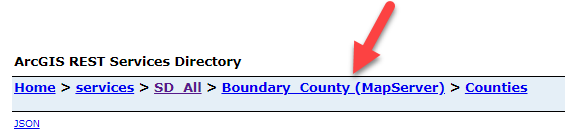


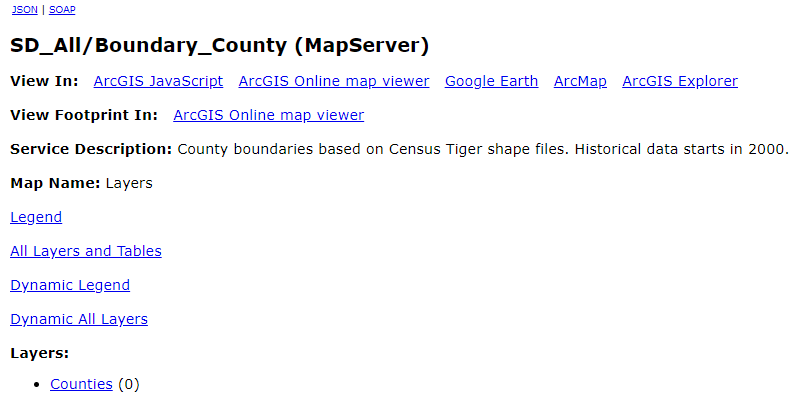
Go to that link and it should bring you to this site:



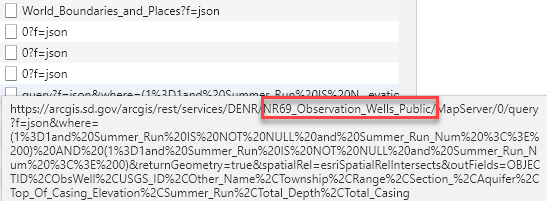
1. This is called the ArcGIS Rest Service, where the API stores its data.

If you click back on “Boundary County (Map Server)”, it will bring you to another site and show you the data that’s available, and the description:





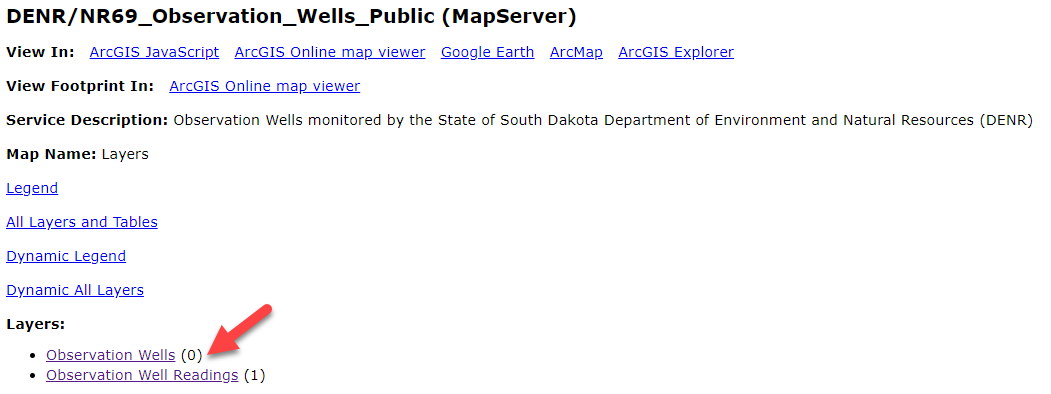
1. You can try going to SD\_all, but that’s not going to contain the observation well data we need. That’s going to be stored in a different query so if you go back to our Developer Tool panel and Network, we can try to look for the query that links to our wells. Sometimes datasets are stored in different queries and you’re going to need to hunt for what you want.



1. This looks more promising, it says Observation wells in the name! Do the same what we did for the counties, and double click on the link. Then clear everything after MapServer/. You can leave the 0/ that comes after MapServer but this will bring you straight to the inventory and description of the data.

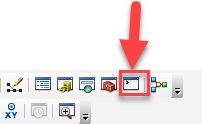


1. Go to this link and voila! We have our observation wells. We’re going to be using the first link in the diagram below, since this contains our depth data.



## Part 3. Scraping the data

1. Now that we have our data, we need to scrape it. Go onto ArcMap and open a blank document. We’re going to scrape this using ArcPy so that it can be readily stored as a shapefile.
2. Open the python command window



1. We’re going to be using this code to pull out the data we need:

import arcpy

import urllib2

import json

# Setup

arcpy.env.overwriteOutput = True

baseURL = **"http://services.gis.ca.gov/arcgis/rest/services/Environment/Wildfires/MapServer/0"**

fields = "\*"

outdata = **"H:/cal\_data/data.gdb/testdata"**

# Get record extract limit

urlstring = baseURL + "?f=json"

j = urllib2.urlopen(urlstring)

js = json.load(j)

maxrc = int(js["maxRecordCount"])

print "Record extract limit: %s" % maxrc

# Get object ids of features

where = "1=1"

urlstring = baseURL + "/query?where={}&returnIdsOnly=true&f=json".format(where)

j = urllib2.urlopen(urlstring)

js = json.load(j)

idfield = js["objectIdFieldName"]

idlist = js["objectIds"]

idlist.sort()

numrec = len(idlist)

print "Number of target records: %s" % numrec

# Gather features

print "Gathering records..."

fs = dict()

for i in range(0, numrec, maxrc):

torec = i + (maxrc - 1)

if torec > numrec:

torec = numrec - 1

fromid = idlist[i]

toid = idlist[torec]

where = "{} >= {} and {} <= {}".format(idfield, fromid, idfield, toid)

print " {}".format(where)

urlstring = baseURL + "/query?where={}&returnGeometry=true&outFields={}&f=json".format(where,fields)

fs[i] = arcpy.FeatureSet()

fs[i].load(urlstring)

# Save features

print "Saving features..."

fslist = []

for key,value in fs.items():

fslist.append(value)

arcpy.Merge\_management(fslist, outdata)

print "Done!"

The only thing you need to change for this code are the parts highlighted in Red. BaseURL needs to be the url you got from the ArcGIS Service API:

<https://arcgis.sd.gov/arcgis/rest/services/DENR/NR69_Observation_Wells_Public/MapServer/0>

And the outdata needs to be saved to where you want it to be saved for example if you want to save it to documents, you would type in: “C:/Documents/datafoldername”

Here’s an example of mine:

import arcpy

import urllib2

import json

# Setup

arcpy.env.overwriteOutput = True

baseURL = **"https://arcgis.sd.gov/arcgis/rest/services/DENR/NR69\_Observation\_Wells\_Public/MapServer/0"**

fields = "\*"

outdata = **"** **C:/Documents/Obs\_wells"**

# Get record extract limit

urlstring = baseURL + "?f=json"

j = urllib2.urlopen(urlstring)

js = json.load(j)

maxrc = int(js["maxRecordCount"])

print "Record extract limit: %s" % maxrc

# Get object ids of features

where = "1=1"

urlstring = baseURL + "/query?where={}&returnIdsOnly=true&f=json".format(where)

j = urllib2.urlopen(urlstring)

js = json.load(j)

idfield = js["objectIdFieldName"]

idlist = js["objectIds"]

idlist.sort()

numrec = len(idlist)

print "Number of target records: %s" % numrec

# Gather features

print "Gathering records..."

fs = dict()

for i in range(0, numrec, maxrc):

torec = i + (maxrc - 1)

if torec > numrec:

torec = numrec - 1

fromid = idlist[i]

toid = idlist[torec]

where = "{} >= {} and {} <= {}".format(idfield, fromid, idfield, toid)

print " {}".format(where)

urlstring = baseURL + "/query?where={}&returnGeometry=true&outFields={}&f=json".format(where,fields)

fs[i] = arcpy.FeatureSet()

fs[i].load(urlstring)

# Save features

print "Saving features..."

fslist = []

for key,value in fs.items():

fslist.append(value)

arcpy.Merge\_management(fslist, outdata)

print "Done!"

1. Once you’re done with that, copy and paste into the python window and hit enter on your keyboard. And then you should be done!

## Resources

Here’s the link to the code that I copied that you can read more about. Happy Scraping!

<https://socalgis.org/2018/03/28/extracting-more-features-from-map-services/>