Effective Batch Geocoding

Will Ibekwe (wibekwe@esri.com)

Victor Bhattacharyya (vbhattacharyya@esri.com)

Expected Audience

- Familiar with Geocoding
- Some knowledge of either Python or Javascript
- Familiar with REST protocol

Outline

- GeocodeAddresses API
- BatchGeocode API
- Advantages to each approach
- Code samples
- Demos

GeocodeAddresses API

- A REST endpoint for Batch Geocoding

- Each request geocodes MaxBatchSize records at a time
- Demands input and output of JSON recordset
- Requires credits when geocoding against the online service

For more info:
https://developers.arcgis.com/rest/geocode/api-reference/geocoding-geocode-addresses.htm

Calling GeocodeAddresses

Geocoding Service

```python
def geocode_batch(service_url, recordset, token, category, source_country, outSR):
    ''' Makes REST calls to geocodeAddresses'''
    if not outSR:
        outSR = 4326
    try:
        # Pass a token when geocoding against a sercured service
        # Both a secured local service and ArcGIS Online require a token
        if token:
            values = ([('addresses',recordset),('token',token), \
                       ('category',category), \
                       ('sourceCountry',source_country), \
                       ('outSR',outSR),('f','json')])
        else:
            values = ([('addresses', recordset),
                  ('category', category), \
                  ('sourceCountry', source_country), \
                  ('outSR',outSR),('f','json')])

        # Encode data to send to the geocoding service
        encodedData = urllib.urlencode(values).encode('utf-8')

        # Secured local service needs an http_header
        http_header = {'referer':referer}
        request = urllib2.Request(service_url,encodedData,http_header)
        # Store the response
        try:
            response = urllib2.urlopen(request,timeout=90)
        except urllib2.HTTPError:
            retried_feature_set = retry_REST_request_until_done(values)

        try:
            res = response.read()
            encoded_res = res.decode('utf-8')
            json_rep = json.loads(encoded_res)
        except Exception as e:
            arcpy.AddError('Exception in REST call: {}'.format(e))
```

Calling GeocodeAddresses

Geocoding Service

```python
def geocode_batch(service_url, recordset, token, category, source_country, outSR):
    ''' Makes REST calls to geocodeAddresses'''
    if not outSR:
        outSR = 4326
    try:
        # Pass a token when geocoding against a sercured service
        # Both a secured local service and ArcGIS Online require a token
        if token:
            values = ([('addresses',recordset),('token',token), \
                       ('category',category), \
                       ('sourceCountry',source_country), \
                       ('outSR',outSR),('f','json')])
        else:
            values = ([('addresses', recordset),
                       ('category', category), \
                       ('sourceCountry', source_country), \
                       ('outSR',outSR),('f','json')])

        # Encode data to send to the geocoding service
        encodedData = urllib.urlencode(values).encode('utf-8')

        # Secured local service needs an http_header
        http_header = {'referer':referer}
        request = urllib2.Request(service_url,encodedData,http_header)
        # Store the response
        try:
            response = urllib2.urlopen(request,timeout=90)
        except urllib2.HTTPError:
            retried_feature_set = retry_REST_request_until_done(values)

        try:
            res = response.read()
            encoded_res = res.decode('utf-8')
            json_rep = json.loads(encoded_res)
        except Exception as e:
            arcpy.AddError('Exception in REST call: {}'.format(e))
```

Calling GeocodeAddresses

Geocoding Service

```python
def geocode_batch(service_url, recordset, token, category, source_country, outSR):
    ''' Makes REST calls to geocodeAddresses'''
    if not outSR:
        outSR = 4326
    try:
        # Pass a token when geocoding against a sercured service
        # Both a secured local service and ArcGIS Online require a token
        if token:
            values = ([('addresses',recordset),('token',token), \
                       ('category',category), \
                       ('sourceCountry',source_country), \
                       ('outSR',outSR),('f','json')])
        else:
            values = ([('addresses', recordset),
                  ('category', category), \
                  ('sourceCountry', source_country), \
                  ('outSR',outSR),('f','json')])

        # Encode data to send to the geocoding service
        encodedData = urllib.urlencode(values).encode('utf-8')

        # Secured local service needs an http_header
        http_header = {'referer':referer}
        request = urllib2.Request(service_url,encodedData,http_header)
        # Store the response
        try:
            response = urllib2.urlopen(request,timeout=90)
        except urllib2.HTTPError:
            retried_feature_set = retry_REST_request_until_done(values)

        try:
            res = response.read()
            encoded_res = res.decode('utf-8')
            json_rep = json.loads(encoded_res)
        except Exception as e:
            arcpy.AddError('Exception in REST call: {}'.format(e))
```

Calling GeocodeAddresses

Geocoding Service

```python
def geocode_batch(service_url, recordset, token, category, source_country, outSR):
    ''' Makes REST calls to geocodeAddresses'''
    if not outSR:
        outSR = 4326
    try:
        # Pass a token when geocoding against a sercured service
        # Both a secured local service and ArcGIS Online require a token
        if token:
            values = ([('addresses',recordset),('token',token), \
                       ('category',category), \
                       ('sourceCountry',source_country), \
                       ('outSR',outSR),('f','json')])
        else:
            values = ([('addresses', recordset),
                  ('category', category), \
                  ('sourceCountry', source_country), \
                  ('outSR',outSR),('f','json')])

        # Encode data to send to the geocoding service
        encodedData = urllib.urlencode(values).encode('utf-8')

        # Secured local service needs an http_header
        http_header = {'referer':referer}
        request = urllib2.Request(service_url,encodedData,http_header)
        # Store the response
        try:
            response = urllib2.urlopen(request,timeout=90)
        except urllib2.HTTPError:
            retried_feature_set = retry_REST_request_until_done(values)

        try:
            res = response.read()
            encoded_res = res.decode('utf-8')
            json_rep = json.loads(encoded_res)
        except Exception as e:
            arcpy.AddError('Exception in REST call: {}'.format(e))
```

Advantages to GeocodeAddresses

- **Fast for small tables**
- **Easy to debug**
- **Lightweight**
- **Synchronous**

For more info:
https://developers.arcgis.com/rest/geocode/api-reference/geocoding-geocode-addresses.htm

BatchGeocode API

New in 10.5

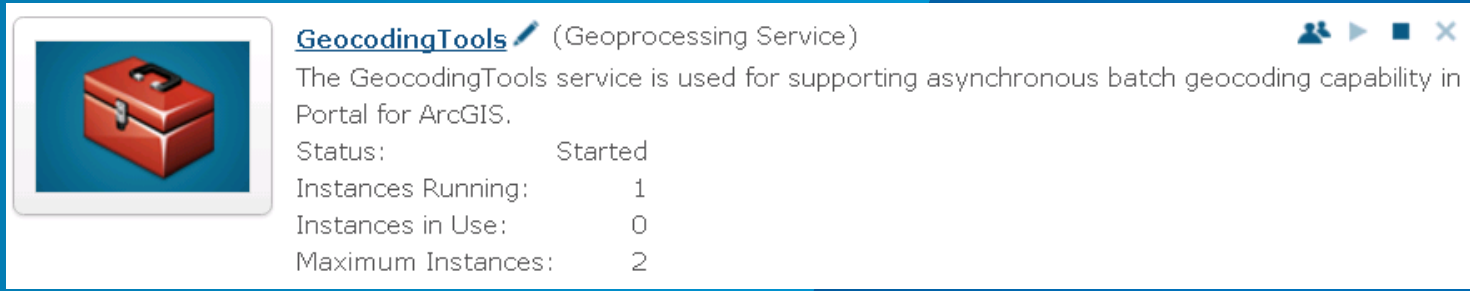
Batch Geocode API

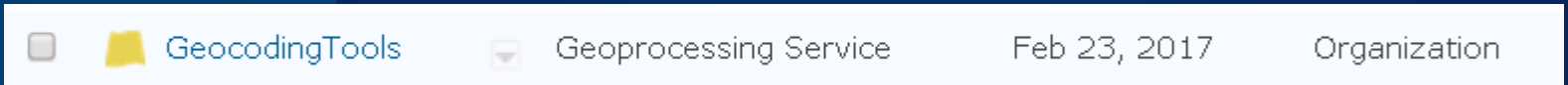
- A REST endpoint for Batch Geocoding

- One-stop-shop for geocoding large tables
- Accepts multiple file formats as input and output
  - CSV
  - XLSX/XLS
  - Hosted Tables
  - Feature layers
- Leverages multithreading for optimized performance

Batch Geocode REST Endpoint

Available in 10.5





Calling BatchGeocode

```
import arcpy
import time

arcpy.ImportToolbox("http://sampleserver1.arcgisonline.com/ArcGIS/services;Utilities/GeocodingTools")

geocode_parameters = '''
{
  "field_info": "[('SingleLine', 'TEXT', 255)]",
  "column_names": "SingleLine",
  "file_type": "csv",
  "column_delimiter": "",
  "text_qualifier": "",
  "header_row_exists": false,
  "field_mapping": "[['SingleLine', 'SingleLine']]"
}
'''

locator_service_url = "http://sampleserver1.arcgisonline.com/ArcGIS/services/World/GeocodeServer"

output_type = "csv"

file = "{'itemid':'xyz'}"

output_fields = "x,y,score,match_addr"

output_name = '''{"serviceProperties" : {"name" : "utf8_csv"}}'''

extent = '''{"outSR": {"wkid": 4326}}'''

result = arcpy.BatchGeocode_GeocodingTools(geocode_parameters, locator_service_url, output_type, "", file, "US", "POI",\
                                           output_fields, "1", output_name, extent)

while result.status < 4:
    print result.status
    time.sleep(0.2)
print "Execution Finished"

result.getMessages()
output = result.getOutput(0)
```

Calling BatchGeocode

```
import arcpy
import time

arcpy.ImportToolbox("http://sampleserver1.arcgisonline.com/ArcGIS/services;Utilities/GeocodingTools")

geocode_parameters = '''
{
  "field_info": "[('SingleLine', 'TEXT', 255)]",
  "column_names": "SingleLine",
  "file_type": "csv",
  "column_delimiter": "",
  "text_qualifier": "",
  "header_row_exists": false,
  "field_mapping": "[['SingleLine', 'SingleLine']]"
}
'''

locator_service_url = "http://sampleserver1.arcgisonline.com/ArcGIS/services/World/GeocodeServer"

output_type = "csv"

file = "{'itemid':'xyz'}"

output_fields = "x,y,score,match_addr"

output_name = '''{"serviceProperties" : {"name" : "utf8_csv"}}'''

extent = '''{"outSR": {"wkid": 4326}}'''

result = arcpy.BatchGeocode_GeocodingTools(geocode_parameters, locator_service_url, output_type, "", file, "US", "POI",\
                                           output_fields, "1", output_name, extent)

while result.status < 4:
    print result.status
    time.sleep(0.2)
print "Execution Finished"

result.getMessages()
output = result.getOutput(0)
```

Calling BatchGeocode

```
import arcpy
import time

arcpy.ImportToolbox("http://sampleserver1.arcgisonline.com/ArcGIS/services;Utilities/GeocodingTools")

geocode_parameters = '''
{
  "field_info": "[('SingleLine', 'TEXT', 255)]",
  "column_names": "SingleLine",
  "file_type": "csv",
  "column_delimiter": "",
  "text_qualifier": "",
  "header_row_exists": false,
  "field_mapping": "[['SingleLine', 'SingleLine']]"
}
'''

locator_service_url = "http://sampleserver1.arcgisonline.com/ArcGIS/services/World/GeocodeServer"

output_type = "csv"

file = "{'itemid':'xyz'}"

output_fields = "x,y,score,match_addr"

output_name = '''{"serviceProperties" : {"name" : "utf8_csv"}}'''

extent = '''{"outSR": {"wkid": 4326}}'''

result = arcpy.BatchGeocode_GeocodingTools(geocode_parameters, locator_service_url, output_type, "", file, "US", "POI",\
                                           output_fields, "1", output_name, extent)

while result.status < 4:
    print result.status
    time.sleep(0.2)
print "Execution Finished"

result.getMessages()
output = result.getOutput(0)
```

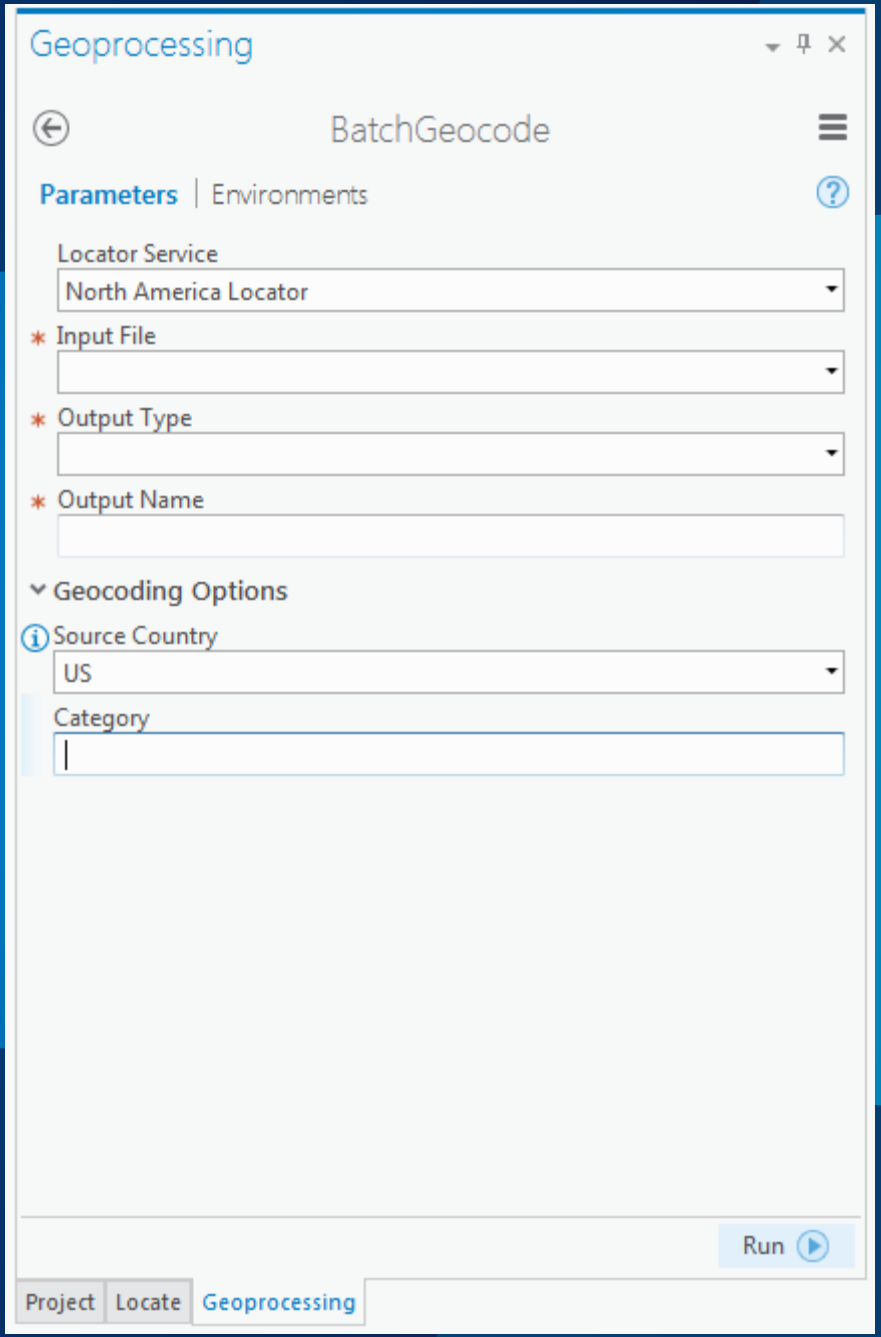
Calling BatchGeocode

```
import arcpy
import time

arcpy.ImportToolbox("http://sampleserver1.arcgisonline.com/ArcGIS/services;Utilities/GeocodingTools")

geocode_parameters = '''
{
  "field_info": "[('SingleLine', 'TEXT', 255)]",
  "column_names": "SingleLine",
  "file_type": "csv",
  "column_delimiter": "",
  "text_qualifier": "",
  "header_row_exists": false,
  "field_mapping": "[['SingleLine', 'SingleLine']]"
}
'''

locator_service_url = "http://sampleserver1.arcgisonline.com/ArcGIS/services/World/GeocodeServer"

output_type = "csv"

file = "{'itemid':'xyz'}"

output_fields = "x,y,score,match_addr"

output_name = '''{"serviceProperties" : {"name" : "utf8_csv"}}'''

extent = '''{"outSR": {"wkid": 4326}}'''

result = arcpy.BatchGeocode_GeocodingTools(geocode_parameters, locator_service_url, output_type, "", file, "US", "POI",\
                                           output_fields, "1", output_name, extent)

while result.status < 4:
    print result.status
    time.sleep(0.2)
print "Execution Finished"

result.getMessages()
output = result.getOutput(0)
```

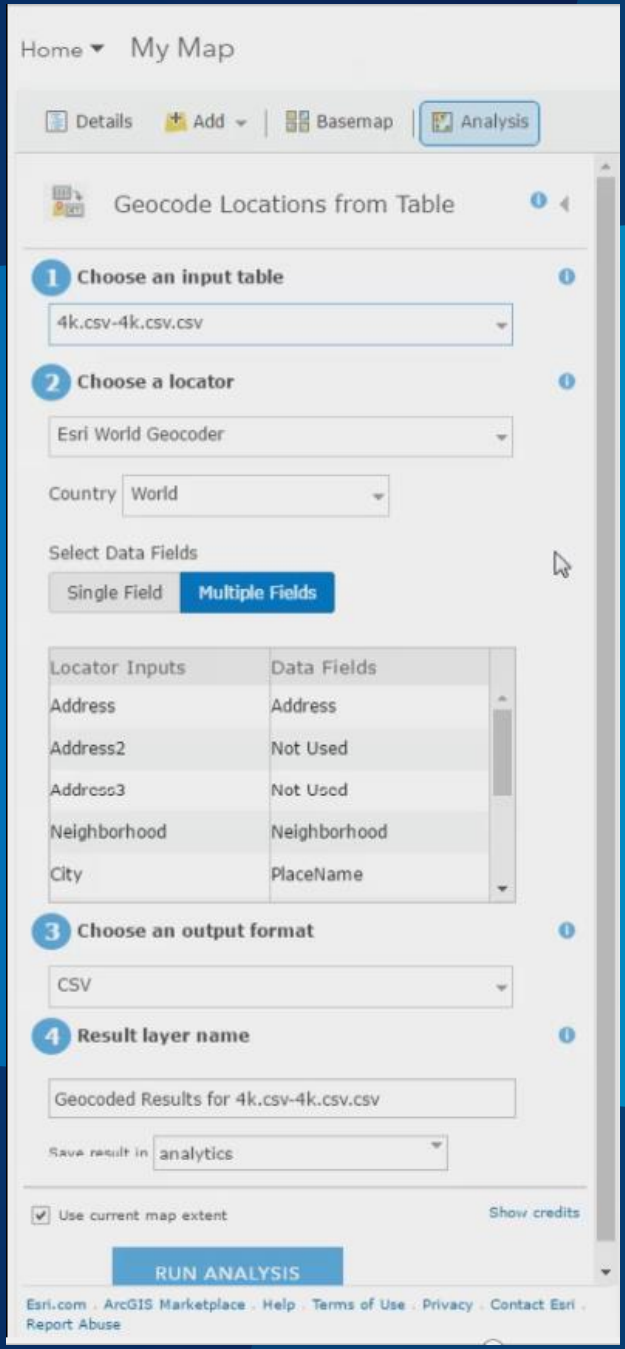
Advantages to BatchGeocode

- More input and output formats
- Built-in error handling
- Robust—can handle large inputs
- Asynchronous

Pro Script Demo

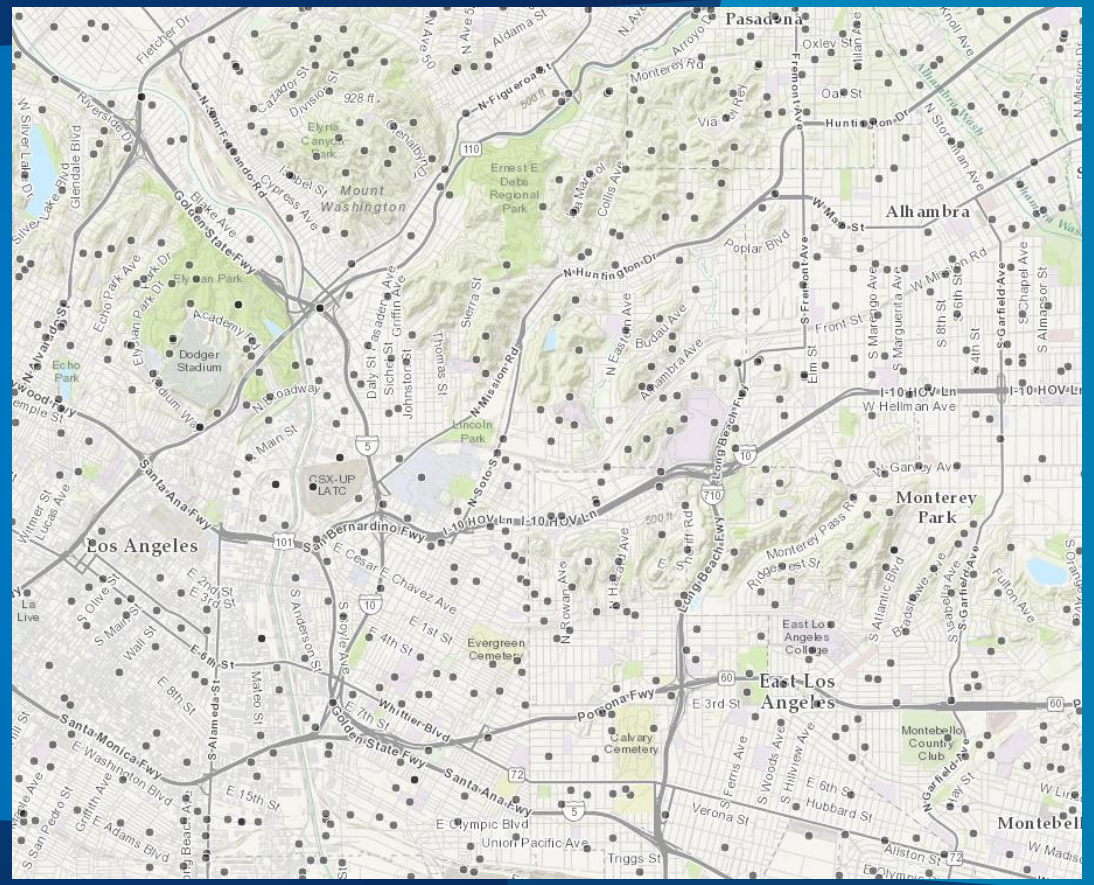


MapViewer Demo



Custom JavaScript App Demo 1

Using both GeocodeAddresses API



https://developers.arcgis.com/javascript/3/
http://resources.arcgis.com/en/help/arcgis-rest-api/index.html#/Geocode_Addresses/02r3000000s6000000/

GeocodeAddress API

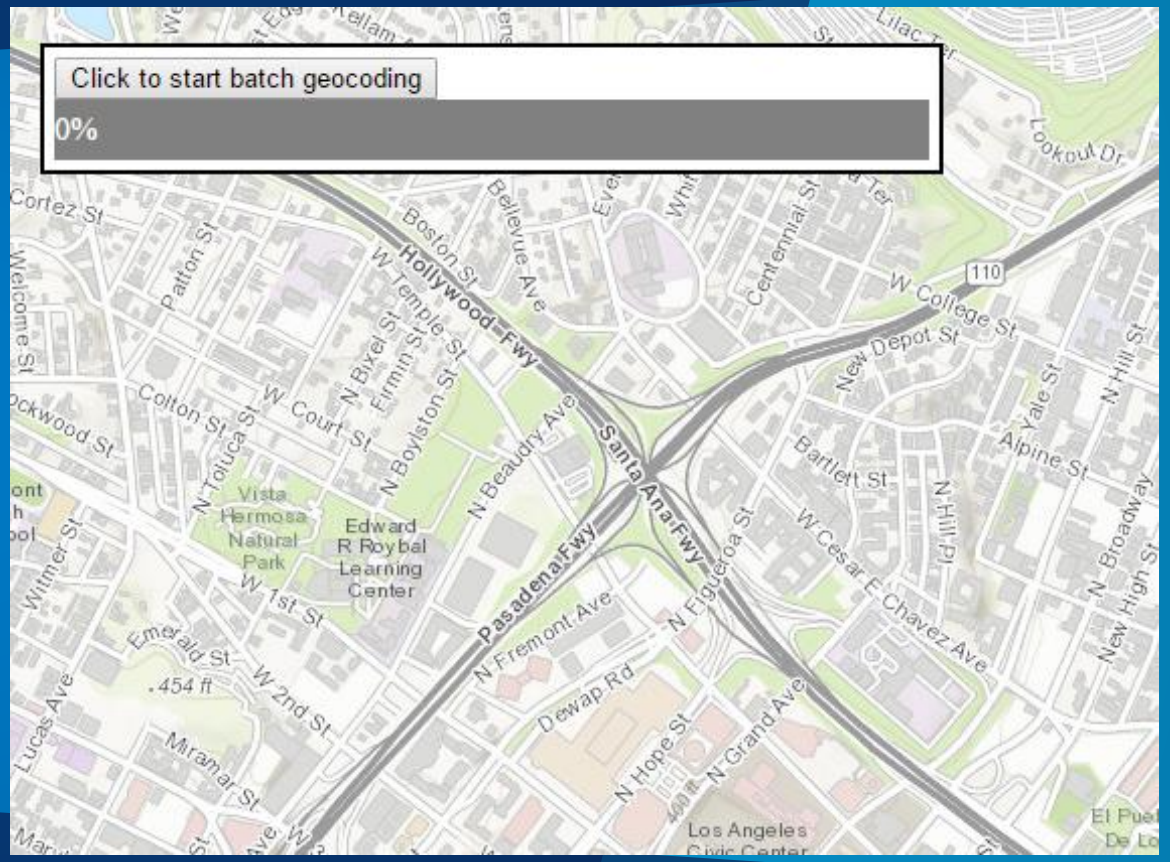
```
function batchRequestSync( evt ) {
    require(["dojo/request"], function ( request ) {
      var url = geocodeServiceURL +"/geocodeAddresses?token=" + token;
      var render = new FileReader();
      render.onload = function(evt){
          //Parse input
        var fileAddrs = render.result.split( /\r\n|\n/ );
        var count = Math.min(fileAddrs.length, 501);
          var counter = 0;

          //Traverses through the file to construct the address parameter for the GeocodeAddress request.
          // This will generate 1000 addresses at a time
        for(var j = 0; j < count; j++ ){
            var addresses = "";
            for(var i = 1; i < 1000 && i < count  ; i++ ) {
                    if(fileAddrs[i].trim() == ""){ continue; }
                    var addr = fileAddrs[i].split( ',' );
                    // Specify the field mapping. This requires knowledge of the file that was uploaded
                var format = {
                    "attributes":{
                        "OBJECTID":addr[0],
                        "Address":addr[1],
                        "City":addr[3],
                        "Region":addr[4],
                        "Postal": addr[6]
                        }
                    };
                    addresses += JSON.stringify( format ) + ",";
                }
            i = 0;
            counter = 1000 * j;
            addresses = addresses.slice( 0, -1 );
            addresses ="{records:[" + addresses + "]}";
             var options = {
                 data:{ addresses: addresses, f:"pjson", token: token }
             };
          // Submit the rest request
            request.post( url, options ).then(
                function( response ){
                    document.getElementById("dropbox").style.display = "none";
                    var results = JSON.parse( response );
                    //filtering through all the response, collecting only points and matching label
                    results = results.locations.map( function( entry ){
                        return {
                            X: entry.attributes.X,
                            Y: entry.attributes.Y,
                            addresses: entry.attributes.Match_addr
                        }
                    });
                    //plots each found point onto the map
                    plotPoints( results );
                },
                function( err ){
                    console.log( err );
                // Requires Error Handling for request that fail.
                }}}}});
```

GeocodeAddress API

```
function batchRequestSync( evt ) {
    require(["dojo/request"], function ( request ) {
      var url = geocodeServiceURL +"/geocodeAddresses?token=" + token;
      var render = new FileReader();
      render.onload = function(evt){
          //Parse input
        var fileAddrs = render.result.split( /\r\n|\n/ );
        var count = Math.min(fileAddrs.length, 501);
          var counter = 0;

          //Traverses through the file to construct the address parameter for the GeocodeAddress request.
          // This will construct 1000 addresses at a time
        for(var j = 0; j < count; j++ ){
            var addresses = "";
            for(var i = 1; i < 1000 && i < count  ; i++ ) {
                    if(fileAddrs[i].trim() == ""){ continue; }
                    var addr = fileAddrs[i].split( ',' );
                    // Specify the field mapping. This requires knowledge of the file that was uploaded
                var format = {
                    "attributes":{
                        "OBJECTID":addr[0],
                        "Address":addr[1],
                        "City":addr[3],
                        "Region":addr[4],
                        "Postal": addr[6]
                        }
                    };
                    addresses += JSON.stringify( format ) + ",";
                }
            i = 0;
            counter = 1000 * j;
            addresses = addresses.slice( 0, -1 );
            addresses ="{records:[" + addresses + "]}";
             var options = {
                 data:{ addresses: addresses, f:"pjson", token: token }
             },
      // Submit the rest request
            request.post( url, options ).then(
                function( response ){
                    document.getElementById("dropbox").style.display = "none";
                    var results = JSON.parse( response );
                    //filtering through all the response, collecting only points and matching label
                    results = results.locations.map( function( entry ){
                        return {
                            X: entry.attributes.X,
                            Y: entry.attributes.Y,
                            addresses: entry.attributes.Match_addr
                        }
                    });
                    //plots each found point onto the map
                    plotPoints( results );
                },
                function( err ){
                    console.log( err );
                // Requires Error Handling for request that fail.
                }}}}});
```

Custom JavaScript App Demo 2

Using both AnalyzeGeocodeInput and BatchGeocode APIs



https://developers.arcgis.com/javascript/3/
http://resources.arcgis.com/en/help/arcgis-rest-api/index.html - /Analyze_Geocode_Input/02r3000002zs000000/
http://resources.arcgis.com/en/help/arcgis-rest-api/index.html#/Batch_Geocode/02r3000002zr000000/

AnalyzeGeocodeInput API

Gathering parameter options

```
//Use the AnalyzeGeocodeInput call
function AsyncAnalyzeInput( evt ) {
    require(["esri/request",  "esri/config", "esri/tasks/Geoprocessor"],
        function (esriRequest,  esriConfig, Geoprocessor ) {

        itemid = prompt("Enter itemID", "");

        var inputFileParameters = {
            "fileType": "csv",
            "headerRowExists": "true",
            "columnDelimiter": "COMMA",
            "textQualifier": ""
        };

        var options = {
            geocodeServiceURL: geocodeServiceURL,
            inputFileItem: JSON.stringify({"itemid": itemid }),
            token: token,
            inputFileParameters: JSON.stringify(inputFileParameters),
            f: "pjson"
        };

        //Call request using token authentication
        esriRequest({
            url: portalURL + "/AnalyzeGeocodeInput/submitJob",
            content: options,
            handleAs: "json"
        }, {usePost: true}).then(
            //call back returns the job id of the request.
            function (response) {
                var jobid = response.jobId;

                //Check the status of teh AnalyzeGeocodeMethods Job
                var url = portalURL + "/AnalyzeGeocodeInput/jobs/" + jobid + "?token=" + token
                var gp = new Geoprocessor(url);
                console.log(url);
                gp.submitJob(null,
                        function (jobInfo) { //Job complete function
                         console.log("AnalyzeGeocodeInput method is complete");
                         getFieldMapping(jobInfo.jobId);
                    }, function (jobInfo) { // Job status change function
                         console.log(jobInfo)
                    }, function (err) { // Job failure function
                         console.log("AnalyzeGeocodeInput method is complete");
                         console.log(err)
                    });
            });
    })
}
```

AnalyzeGeocodeInput API

Starting AnalyzeGeocodeInput request

```
//Use the AnalyzeGeocodeInput call
function AsyncAnalyzeInput( evt ) {
    require(["esri/request",  "esri/config", "esri/tasks/Geoprocessor"],
        function (esriRequest,  esriConfig, Geoprocessor ) {

        itemid = prompt("Enter itemID", "");

        var inputFileParameters = {
            "fileType": "csv",
            "headerRowExists": "true",
            "columnDelimiter": "COMMA",
            "textQualifier": ""
        };

        var options = {
            geocodeServiceURL: geocodeServiceURL,
            inputFileItem: JSON.stringify({"itemid": itemid }),
            token: token,
            inputFileParameters: JSON.stringify(inputFileParameters),
            f: "pjson"
        };

        //Call request using token authentication
        esriRequest({
            url: portalURL + "/AnalyzeGeocodeInput/submitJob",
            content: options,
            handleAs: "json"
        }, {usePost: true}).then(
            //call back returns the job id of the request.
            function (response) {
                var jobid = response.jobid;

                //Check the status of teh AnalyzeGeocodeMethods Job
                var url = portalURL + "/AnalyzeGeocodeInput/jobs/" + jobid + "?token=" + token
                var gp = new Geoprocessor(url);
                console.log(url);
                gp.submitJob(null,
                        function (jobInfo) { //Job complete function
                         console.log("AnalyzeGeocodeInput method is complete");
                         getFieldMapping(jobInfo.jobId);
                    }, function (jobInfo) { // Job status change function
                         console.log(jobInfo)
                    }, function (err) { // Job failure function
                         console.log("AnalyzeGeocodeInput method is complete");
                         console.log(err)
                    });
            });
    })
}
```

AnalyzeGeocodeInput API

```
//Use the AnalyzeGeocodeInput call
function AsyncAnalyzeInput( evt ) {
    require(["esri/request",  "esri/config", "esri/tasks/Geoprocessor"],
        function (esriRequest,  esriConfig, Geoprocessor ) {

        itemid = prompt("Enter itemID", "");

        var inputFileParameters = {
            "fileType": "csv",
            "headerRowExists": "true",
            "columnDelimiter": "COMMA",
            "textQualifier": ""
        };

        var options = {
            geocodeServiceURL: geocodeServiceURL,
            inputFileItem: JSON.stringify({"itemid": itemid }),
            token: token,
            inputFileParameters: JSON.stringify(inputFileParameters),
            f: "pjson"
        };

        //Call request using token authentication
        esriRequest({
            url: portalURL + "/AnalyzeGeocodeInput/submitJob",
            content: options,
            handleAs: "json"
        }, {usePost: true}).then(
            //call back returns the job id of the request.
            function (response) {
                var jobid = response.jobId;

                //Check the status of teh AnalyzeGeocodeMethods Job
                var url = portalURL + "/AnalyzeGeocodeInput/jobs/" + jobid + "?token=" + token
                var gp = new Geoprocessor(url);
                console.log(url);
                gp.submitJob(null,
                        function (jobInfo) { //Job complete function
                            console.log("AnalyzeGeocodeInput method is complete");
                            getFieldMapping(jobInfo.jobId);
                    }, function (jobInfo) { // Job status change function
                            console.log(jobInfo)
                    }, function (err) { // Job failure function
                            console.log("AnalyzeGeocodeInput method is complete");
                            console.log(err)
                    });
            });
    })
}
```

Checking to see if request is complete

Retrieving Field Mapping Results

```
//This method will retrieve the field mapping from the BatchGeocoding response.
function getFieldMapping( jobId ){
    require([ "esri/request"], function (esriRequest) {

        var options = {
            token: token,
            f:"json"
        };

        esriRequest({
            url: portalURL + "/AnalyzeGeocodeInput/jobs/" + jobId + "/results/geocodeParameters",
            content:options
        }).then(
            function (response) {
                // This will provide the response for the recommended field mapping
                batchRequestAsync(response.value);
            },
            function (err) {
                console.log(err);
            }
        );
    })
}
```

BatchGeocode API

Gathering parameter options

```
function batchRequestAsync( inputFormat ){
    require(["esri/request", "esri/config", "esri/tasks/Geoprocessor"], function (esriRequest, esriConfig, Geoprocessor) {
        var filename = prompt("Enter the name of new file", "");
        var options = {
            token: token,
            geocodeServiceURL: geocodeServiceURL,
            geocodeParameters: JSON.stringify(inputFormat),
            outputType: "csv",
            outputName:  JSON.stringify({"serviceProperties":{"name":filename }}),
            context: JSON.stringify({"outSR":{"wkid":4326}}),
            inputFileItem:  JSON.stringify({"itemid":itemid}),
            f:"json"
        };

        esriRequest( {
            url: portalURL + "/BatchGeocode/submitJob",
            content: options
        }, {usePost:true}).then(
            function(response){
                var gp = new Geoprocessor(portalURL + "/BatchGeocode/jobs/" + response.jobId);
                gp.submitJob(null,
                    function (jobInfo) { //Job complete function
                        progressBar(100);
                        var options = { token: token, f:"json" };
                        retriveResults(jobInfo.jobid);
                },  function (jobInfo) { // Job status change function
                        if(jobInfo.messages.length > 0) {
                            var status  = jobInfo.messages[jobInfo.messages.length - 1].description.split(" ")[0];
                            if(parseFloat(status).toString() != "NaN"){
                                progressBar(status);
                            }
                        }
                },  function (err) { // Job failure function
                        console.log(err)
                });
            },
            function(err){
                console.log(err);
            }
        )
    });
}
```

BatchGeocode API

Starting BatchGeocode request

```
function batchRequestAsync( inputFormat ){
    require(["esri/request", "esri/config", "esri/tasks/Geoprocessor"], function (esriRequest, esriConfig, Geoprocessor) {
        var filename = prompt("Enter the name of new file", "");
        var options = {
            token: token,
            geocodeServiceURL: geocodeServiceURL,
            geocodeParameters: JSON.stringify(inputFormat),
            outputType: "csv",
            outputName:  JSON.stringify({"serviceProperties":{"name":filename }}),
            context: JSON.stringify({"outSR":{"wkid":4326}}),
            inputFileItem:  JSON.stringify({"itemid":itemid}),
            f:"json"
        };

        esriRequest( {
            url: portalURL + "/BatchGeocode/submitJob",
            content: options
        }, {usePost:true}).then(
            function(response){
                var gp = new Geoprocessor(portalURL + "/BatchGeocode/jobs/" + response.jobId);
                gp.submitJob(null,
                    function (jobInfo) { //Job complete function
                        progressBar(100);
                        var options = { token: token, f:"json" };
                        retriveResults(jobInfo.jobid);
                },  function (jobInfo) { // Job status change function
                        if(jobInfo.messages.length > 0) {
                            var status  = jobInfo.messages[jobInfo.messages.length - 1].description.split(" ")[0];
                            if(parseFloat(status).toString() != "NaN"){
                                progressBar(status);
                            }
                        }
                },  function (err) { // Job failure function
                        console.log(err)
                });
            },
            function(err){
                console.log(err);
            }
        )
    });
}
```

BatchGeocode API

```
function batchRequestAsync( inputFormat ){
    require(["esri/request", "esri/config", "esri/tasks/Geoprocessor"], function (esriRequest, esriConfig, Geoprocessor) {
        var filename = prompt("Enter the name of new file", "");
        var options = {
            token: token,
            geocodeServiceURL: geocodeServiceURL,
            geocodeParameters: JSON.stringify(inputFormat),
            outputType: "csv",
            outputName:  JSON.stringify({"serviceProperties":{"name":filename }}),
            context: JSON.stringify({"outSR":{"wkid":4326}}),
            inputFileItem:  JSON.stringify({"itemid":itemid}),
            f:"json"
        };

        esriRequest( {
            url: portalURL + "/BatchGeocode/submitJob",
            content: options
        }, {usePost:true}).then(
            function(response){
                var gp = new Geoprocessor(portalURL + "/BatchGeocode/jobs/" + response.jobId);
                gp.submitJob(null,
                    function (jobInfo) { //Job complete function
                        progressBar(100);
                        var options = { token: token, f:"json" };
                        retriveResults(jobInfo.jobid);
                }, function (jobInfo) { // Job status change function
                        if(jobInfo.messages.length > 0) {
                            var status  = jobInfo.messages[jobInfo.messages.length - 1].description.split(" ")[0];
                            if(parseFloat(status).toString() != "NaN"){
                                progressBar(status);
                            }
                        }
                }, function (err) { // Job failure function
                        console.log(err)
                });
            },
            function(err){
                console.log(err);
            }
        )
    });
}
```

Checking to see if request is complete

Retrieve Geocding Results

```
function retriveResults( jobID ){
    esriRequest({
        url: portalURL + "/BatchGeocode/jobs/" + response.jobId + "/results/geocodeResult",
        content:options
    }).then(
        function (response) {
            var button = document.createElement("a");
            button.href = contentURL + response.value.itemId;
            button.innerHTML = "Click here to see the file in the portal!";
            document.getElementById("file").appendChild(button);
        },
        function (err) {
            console.log(err);
        });
}
```

- GeocodeAddresses API
- BatchGeocode API
- Advantages to each approach
- Code samples Python and JavaScript
- Demos

Questions?

esri

THE SCIENCE OF WHERE