Esri International Developer Summit

Palm Springs, CA

# Creating and Working with Geoprocessing Services

Kevin Hibma

March 12, 2014

# Design your service

- Where does data come from?
  - Upload
  - Already on the server
  - Feature layer from map/feature service
- How do you want to view results?
  - Draw features with Map Service
  - Client download and draws features

## Workflow to create a Geoprocessing Service

- Create tool
- Document tool
- Run tool (*from ArcMap if possible)
- From Results Window, publish as service
- Set service name, parameters, etc in the Service Editor
- Analyze
- Publish
- Consume in a *WebApp*, ArcMap, etc.

# WebApp (JavaScript)

- Three steps…

```
var gp;
  gp = new Geoprocessor("

function computeViewShed(evt) {
  map.graphics.clear();
  var pointSymbol = new SimpleMarkerSymbol();
  pointSymbol.setSize(14);
  pointSymbol.setColor(new Color([0, 255, 0, 0.25]));
  var graphic = new Graphic(evt.mapPoint, pointSymbol);
  map.graphics.add(graphic);

  var features = [];
  features.push(graphic);
  var featureSet = new FeatureSet();

function drawViewshed(results, messages) {
  var polySymbol = new SimpleFillSymbol();
  polySymbol.setOutline(new SimpleLineSymbol(SimpleLineSymbol.STYLE_SOLID, new Color([0, 0, 0, 0.5]), 1));
  polySymbol.setColor(new Color([255, 127, 0, 0.7]));
  var features = results[0].value.features;
  for (var f = 0, fl = features.length; f < fl; f++) {
    var feature = features[f];
    feature.setSymbol(polySymbol);
    map.graphics.add(feature);
  }
  map.setExtent(graphicsUtils.graphicsExtent(map.graphics.graphics), true);
}
```

Create geoprocessor object and supply URL to the task

Gather parameters and submit or execute the task

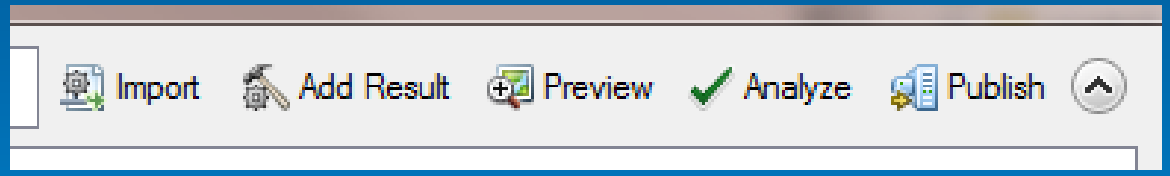
Get the response and draw it

# Creating Surfaces case 1: uploading data

- Data uploaded
- Map Service draws

# Service Editor

- Import configuration settings
- Multiple results = multiple tasks
- See how the task will look to someone consuming from Desktop
- Analyzer errors and warnings



# Data Store

- Data Store tells ArcGIS Server about your data
- Without a Data Store entry, all required data is copied to the server
- Data Store acts as a lookup table

| | | |
|---|---|---|
| • C:\data\analysis | Update Paths | • E:\fileShare\gisdata\landAnalysis |
| • SDE: sqlserver:dtuser | | • SDE: sqlserver:agsuser |
| • C:\gisdata\projects | Same Paths | • C:\gisdata\projects |
| • SDE: oracle:sdeuser | | • SDE: oracle:sdeuser |

Data Store: http://esriurl.com/datastore

# Web App: Initialize the task

- Syntax slightly different depending on the API
- Pattern remains the same, reference the URL and wire up methods

```
            gp = new
Geoprocessor("http://sampleserver6.arcgisonline.com/ArcGIS/rest/services/Elevation/ESRI_Elevation
_World/GPServer/Viewshed");
            gp.setOutputSpatialReference({
              wkid: 102100
            });
```

JavaScript

```
_geoprocessorTask = new Geoprocessor("http://serverapps101.esri.com/arcgis/rest/services/ProbabilisticViewshedModel/GPServer/ProbabilisticViewshedModel")
_geoprocessorTask.JobCompleted += GeoprocessorTask_JobCompleted;
_geoprocessorTask.Failed += GeoprocessorTask_Failed;
```

Silverlight

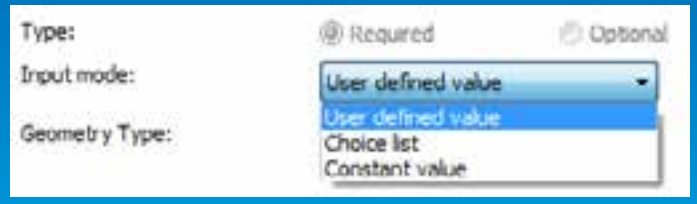
```
        <esri:Geoprocessor id="gp"
                           executeComplete="executeCompleteHandler(event)"
                           fault="faultHandler(event)"
                           outSpatialReference="{map.spatialReference}"
                           showBusyCursor="true"

url="http://sampleserver6.arcgisonline.com/arcgis/rest/services/Elevation/ESRI_Elevation_Wo
rld/GPServer/Viewshed"/>
```

Flex

# Parameter transformation

- Parameter types converted to supported types when publishing
- You can update the Input Mode depending on the parameter type
  - User Defined Value: allows the end user to interactively add features or enter text and number values
  - Choice list: allows the end user to select from a list of layers already on the server
  - Constant value: hard codes the parameter; the end user will not be able to provide input



# WebApp: Gather Parameter Input

- **Parameter names and types must match the tool definition.**

Silverlight

```
List<GPParameter> parameters = new List<GPParameter>();
parameters.Add(new GPFeatureRecordSetLayer("Input_Features", mapPoint));
parameters.Add(new GPString("Height", HeightTextBox.Text));
parameters.Add(new GPLinearUnit("Distance", esriUnits.esriMiles, Convert.ToDouble(MilesTextBox.Text)));
```

JavaScript

```
var myFeatureSet:FeatureSet = new FeatureSet([{ geometry: myMapPoint }]);
myViewshedDistance.distance = Number(viewshedDistance.text);
myViewshedDistance.units = "esriMiles";
var params:Object = {
        "Input_Observation_Point": myFeatureSet,
        "Viewshed_Distance": myViewshedDistance
    };
```

*http://sampleserver6.arcgisonline.com/arcgis/rest/services/Elevation/ESRI_Elevation_World/GPServer/Viewshed*

**Parameter:** Viewshed_Distance
**Data Type:** GPLinearUnit
**Display Name** Viewshed_Distance
**Description:** The maximum distance from the input point for which the viewshed should be calculated. The maximum allowed distance is 20000 meters.
**Direction:** esriGPParameterDirectionInput
**Default Value:** 15000.0 (esriMeters)
**Parameter Type:** esriGPParameterTypeRequired
**Category:**

# Execution Mode

- **Execution mode defines how the client interacts with service while it executes**
  - **Synchronously: the client waits for the server to finish executing and then gets the result.**
  - **Asynchronously: client must ask the server if its finished then get the result. The client is free to do other work during this time.**

  - **Can only use a Result Map Service with Async.**
  - **Synchronous services are typically fast services**

# WebApp: Execute and Submit

- Pass parameter array when calling the task
- Sync = Execute
  - Provide a call back function when the task is finished.
  - Should trap for a failed task

```
gp.execute(params, drawViewshed);
```

- Async = SubmitJob
  - Provide functions to check for jobStatus and Failure in addition to the JobComplete function

```
gp.submitJob(params, gpJobComplete, gpJobStatus, gpJobFailed);
```

# Result Map Service

- A result map service (RMS) provides an additional way to get results from the Geoprocessing Service.
- An image is returned to the client.
  - The data can still be downloaded.
- Use a RMS when:
  - Want better cartography than the client can support
  - It is impractical to render a large dataset in a client.
- Execution must be Asynchronous when using a RMS

# WebApp: Draw the result

- **Draw features or display the result map service**

```
            private function executeCompleteHandler(event:GeoprocessorEvent):void
            {
                for each (var myGraphic:Graphic in
 event.executeResult.results[0].value.features)
                {
                    myGraphic.symbol = viewshedSimpleFill;
                    myGraphicsLayer.add(myGraphic);
                }
            }
```

Flex

```
function gpJobComplete(jobinfo){

  //construct the result map service url using the id from jobinfo we'll add a new layer to the map
  var mapurl = mapserviceurl + "/" + jobinfo.jobId;
  var hotspotLayer = new esri.layers.ArcGISDynamicMapServiceLayer(mapurl,{
    "id":"HotspotLayer",
    "opacity": 0.7
  });

  //when the layer is added initialize the time slider
  dojo.connect(map,'onLayersAddResult',updateMap);

  //add the hotspot layer to the map
  map.addLayers([hotspotLayer]);
}
```

JavaScript

# Script Tools

- Paths and data handled the same as models
- Output and Intermediate paths
  - os.path.join(arcpy.env.scratchFolder, “out.shp”)
  - os.path.join(arcpy.env.scratchGDB, “out”)
  - In_memory\out

# Creating Surfaces case 2: data on the server

- Data on server
  - Referenced in the datastore
- User supplies query
- Map server draws result

# Creating surfaces Case 3: layer from a webmap

- Data from layer
  - Code inside web app handles the feature layer
- Client draws result

# Useful links

- **Quick tour of Authoring and Sharing GP Services - http://esriurl.com/gpSrvQuick**

- **Web APIs (JavaScript, Flex, Silverlight) - http://developers.arcgis.com/**


esri
Understanding our world.