Improving Access to GIS Data With Targeted Web Apps

Joyce Green, City of Norman
John McIntosh, City of Norman
Improving Access to GIS Data with Targeted Web Apps

• Background
• Objective
• Development Environment
• Development Strategy
• Challenges
• Current Status
• Future Plans
The Setting…

Oklahoma in USA…

Cleveland County in Oklahoma…

Norman in Cleveland County…
The Setting

- Third largest city in Oklahoma
  - Land Area ~ 189 Square Miles
  - Population ~ 115,000
  - Home of the University of Oklahoma
  - ESRI User Since 1992
GIS Profile – Beginning of Project

- Version of ArcGIS before project: 9.2
- Enterprise ArcSDE for SQL Server
- External web apps: ArcIMS
- No internal web apps
- Number of ArcGIS Desktop users: 80+
- Third party applications: 2
  - Cityworks by Azteca
  - CrimeView by Omega
Project Objective

- Provide easy to use targeted web apps to access GIS data
  - Replicate business practices in web browser
  - Minimize the number of ArcGIS Desktop installs
  - Maximize the number of users of GIS data
  - Minimize the amount of training required
Development Environment - System Overview

- **Two Systems**
  - **ArcGIS Production Server**
    - Internal use only
    - Supports:
      - Desktop users
      - Web maps to support City business practices
      - Third party applications
  - **DMZ**
    - External
    - Not linked to production server
    - Supports
      - Externally facing web maps
      - External access to City GIS data through map services
Development Environment - System Architecture

- **Servers – VMware vSphere**
  - **Production**
    - ArcGIS Server Enterprise 10.1 and IIS
      - Windows Server 2008 R2 Datacenter
      - 8GB RAM,
      - 80GB hard drive
    - Database Server ArcSDE on SQL Server 2008
      - Windows Server 2008 R2 Datacenter
      - 8GB RAM,
      - 80GB & 800GB hard drives
  - **DMZ**
    - ArcGIS Server Workgroup, IIS, and Sequel Express database
      - Windows Server 2008 R2 Datacenter
      - 12GB RAM
      - 80GB, 500GB, 600GB hard drives
Development Environment: Platform

- ArcGIS API for Javascript
  - Started with version 3.5 and “legacy” style coding
  - Now using version 3.9 with the AMD style coding
- Use ESRI hosted APIs for both versions
- Other libraries
  - Shp-write
    - Javascript library to read and write shapefiles
  - JSZip
    - Javascript library to read, write and edit zipfile
  - XLSX
    - Javascript library to read, write and edit Excel (xlsx) files
Development Strategy

• Server initialization
  - ESRI Jumpstart
    - Installed software
    - Assisted with initial services
    - Gave advice on data loading
  - IT Department
    - Built servers
    - Set up domain
  - Third party vendor
    - Upgraded Cityworks software from desktop to server

• Application Development
  - In-house by GIS staff
Application Development

- Design considerations
  - Customer expectations
Application Development

• Design Considerations
  - Business practices of workgroups
  - Performance
Application Development

- Design considerations
  - Focused tasks
Application Development

• **Approach**
  - Learn by example
    - Forums
    - ArcGIS API for JavaScript

• **Bluefish Editor**
  - Javascript development
  - Open source
  - Used by IT department

• **IDLE Editor**
  - Python scripts
Application Development

- Modular design
  - Shared css
  - Shared JavaScript functions
Application Development

• Example of generic JavaScript functions
Application Development

• Services built on geoprocessing tasks
  - Extract and export data
  - High quality printing
  - Export webmap

• Customized from ESRI tutorials and models
  - Clip and ship
  - Advanced high-quality web map printing/exporting
Application Development

- HTML
  - Designed for easy editing and customization.
  - User defined parameters as variables at top

```javascript
// Define map services and layers to be used
var mainMapServiceURL = "http://terra:6080/arcpin/rest/services/ServicesForWeb/PermitCounter_web/MapServer";
var printServiceURL = "http://terra:6080/arcpin/rest/services/ServicesForWeb/PrintExport_web/MapServer";
var locatorServiceURL = "http://terra:6080/arcpin/rest/services/Utilities/AddressLocator/MapServer";
var geometryServiceURL = "http://terra:6080/arcpin/rest/services/Utilities/GeometryGeometryService";
var dataExtractionServiceURL = "http://terra:6080/arcpin/rest/services/ServicesForWeb/ExtractDataTask_web/OPServer"

// Define map layers and services to be used
var printTemplate = "CurrentPlanning":

var imageLayerInfoArray = [
  "2013-Citywide", "https://map.norman.org/arcpin/rest/services/AerialPhoto2013/ImageServer",
  "2010-Citywide", "https://map.norman.org/arcpin/rest/services/AerialPhoto2010/ImageServer",
  "1997-Urban", "https://map.norman.org/arcpin/rest/services/Imagery/AerialPhotoTwoFF1997/ImageServer"
];
```
Challenges

- Asynchronous processes
- Syntax changes between legacy and AMD JavaScript APIs
- Designing modular services
- System performance
Challenges - Asynchronous Processes

• Combining asynchronous processes
  - An example – An application to estimate proposed fees stormwater fees based on the amount of impermeable surfaces on a parcel

- General Steps
  1. User selects parcel
  2. Overlapping features from a building layer, a pavement layer, a parking lot layer and a pool layer are selected
  3. The selected features are intersected with the selected parcel
  4. The impermeable area and fee are calculated when the intersections of all layers are finished
Challenges - Asynchronous Processes
Start

User Selects a parcel

Select overlapping building layer features
  - Intersect the building layer features and selected parcel
  - Add the results as a graphic to the map
  - Calculate the area of the intersection results

Select overlapping pavement features
  - Intersect the overlapping pavement features and selected parcel
  - Add the results as a graphic to the map
  - Calculate the area of the intersection results

Select overlapping parking lot features
  - Intersect the overlapping parking lot features and selected parcel
  - Add the results as a graphic to the map
  - Calculate the area of the intersection results

Select overlapping pool features
  - Intersect the overlapping pool features and selected parcel
  - Add the results as a graphic to the map
  - Calculate the area of the intersection results

Sum the intersection results area

Calculate the fee

Challenges - Asynchronous Processes
Challenges - Asynchronous Processes

- Each selection/intersection/area calculation consists of asynchronous processes.
- ESRI processes contain callbacks which allow the process to occur synchronously for each feature type.

```javascript
buildingLayer.selectFeatures(selectQuery, FeatureLayer.SELECTION_NEW);
buildingLayer.on("selection-complete", function(){
    // Do something here..
});
```
Challenges - Asynchronous Processes

- How to delay the final calculation until the processes for each feature type are completed?
- We use dojo deferred lists

```javascript
doParkingIntersectionTask = new Deferred();
doPoolIntersectionTask = new Deferred();

defered = all([getParcelAreasTask,
doBuildingIntersectionTask,
doPavementIntersectionTask,
doParkingIntersectionTask,
doPoolIntersectionTask]);
```
Challenges - Asynchronous Processes

- Add a resolve statement for the functions used to calculate the areas for each of the feature types
  
  ```javascript
  getParcelAreasTask.resolve(newArea);
  ```

- Add a deferred.then function.
  ```javascript
  deferred.then(function() {
    // Code to do the calculations and display the results..
  })
  ```
Challenges - Legacy and AMD Syntax Differences

- ArcGIS API after 3.4 written in AMD style
  - ESRI recommends AMD style although “legacy” is still supported
  - Update requires more than simple cut-and-paste
    - Dependencies are defined differently
    - Other syntax differences
Challenges - Legacy and AMD Syntax Differences

- Loading Dependencies
  
  **“Legacy”**
  
  ```javascript
  dojo.require("esri.map");
  dojo.require("esri.tasks.locator");
  
  function init(){
      //code here..
  }
  ```

  **“AMD”**
  
  ```javascript
  require(["esri/map","esri/tasks/locator",...],
    function (map, Locator,...) {
        //code here..
    });
  ```
Challenges - Legacy and AMD Syntax Differences

• connect vs .on for functions with callbacks

“Legacy”
locator = new esri.tasks.Locator(locatorServiceURL);
dojo.connect(locator, "onAddressToLocationsComplete", showLocationResults);

“AMD”
locator = new Locator(locatorServiceURL);
locator.on("address-to-locations-complete", showLocationResults);
Challenges – Modular Services

- Used working units MXDs for initial map services
  - Resulted in data being published multiple times
  - Easy to deploy, but degrades performance
- Need to redesign Apps to use modular services
Challenges – System Performance

• Monitor to determine where bottleneck is in system
• Add RAM
• Add additional machines to site
Current Status

- Eight services deployed internally
Current Status

- One application deployed externally
Current Status

• Four maps deployed on ArcGIS Online
Current Status - GIS Profile

- Version of ArcGIS before project: 10.1
- External web apps: ArcGIS Server for Workgroups 10.1
- Enterprise Database: ArcSDE 10.1 for SQL Server
- Internal web apps: ArcGIS Server 10.1 Enterprise
  - Permit Counter
  - Engineering
  - Solid Waste
  - Police Special Operations
- Number of ArcGIS Desktop users: 16
- Third party applications: 3
  - Cityworks by Azteca
  - CrimeView by Omega
  - New World Systems
Future Plans

- Upgrade to 10.2.1 or latest version that works with third party apps in October 2014
- Deploy more external applications
- Deploy mobile apps
- Make greater use of ArcGIS Online
Demo & Questions

Contact information:
Joyce.Green@NormanOK.gov
John.McIntosh@NormanOK.gov

Website:
http://maps.normanok.gov/index.html
ArcGIS Online site:
http://normanok.maps.arcgis.com/home/index.html