Extending Services in ArcGIS for Server

Sterling Quinn

@Sterling GIS

Offering ID: 276
Session overview

- Introduction to server object extensions (SOEs)
- Developing an SOE
- Deploying an SOE
- Using an SOE in a client application
Introduction to SOEs
What is an SOE?

- **Server Object Extension**

- Extends base functionality of ArcGIS Server using ArcObjects

- A class library you write in .NET or Java and deploy to your server
Enable SOEs when you create or edit a service.
Advantages of SOEs

- Functional
- Fast
- Pluggable
- Consumable through REST web services
Demo: SOEs in action

- Elevation profile
- Dynamic segmentation
Detour: Alternatives to SOEs
Building layouts for web printing

- Out-of-the-box PrintingTools service in 10.1
- Arcpy.mapping for advanced printing
Changing symbols on the fly

• “Dynamic layers” capability at 10.1
  - Specify drawing information as part of map service request

• Client-side graphics with Web APIs’ FeatureLayer
Web editing

- Feature service and the Web APIs
Geoprocessing

- ModelBuilder published as geoprocessing service
- Python script tool as geoprocessing service
- Custom geoprocessing tool with .NET, C++, etc.
Geometric calculations

- Geometry service
When do you need an SOE?

- Need functions beyond out-of-the-box services
- Can’t or don’t want to use geoprocessing framework
- Leverage fast server-side processing using ArcObjects
Developing an SOE
Today we’ll focus on REST Web service SOEs

• Easily invoked by the Esri Web APIs
  - JavaScript
  - Flex
  - Silverlight
  - iOS
  - Etc.
What you have to understand to develop an SOE

- .NET or Java technologies
- REST or SOAP communication
- ArcObjects
- Optional technologies for custom property pages
  - HTML and JavaScript for Manager pages
  - Java Swing or Windows Forms, and ArcCatalog customization, for ArcCatalog pages
Demo: Another SOE in action

Spatial Query REST
Steps for developing and deploying an SOE

1. Write the code
2. Deploy the SOE
3. Publish a service and enable the SOE on it
4. Use the SOE in a client application you develop
An SOE implements the following interfaces

- `IServerObjectExtension`
- `IOBJECTConstruct` *(put initialization logic here)*
- `IRESTRequestHandler` *

* Other interfaces are used with SOAP and DCOM SOEs
Getting started with the code

- REST SOE template in Visual Studio
- Installed with .NET ArcObjects SDK
- Has an example SOE stubbed out for you
Demo: A tour of the REST SOE template
Creating the schema

- What resources and operations will your SOE expose?
  - Resources give you back information
  - Operations do something

- Snap them together to make a schema

- You may need to diagram this
SOE capabilities (or “Operations allowed”)

- Determine which schema items clients can access
- Configured as a parameter on schema items

```java
RestResource customLayerResource = new RestResource("customLayers", true, CustomLayer, "GetInfo");
```
Demo: Exploring REST SOE schemas
Writing handler functions

• Each resource and operation has a handler

• Operation handlers are where most of your business logic is invoked
Working with JSON

• Your handlers have to...
  - Deserialize incoming JSON
  - Do something with it (often using ArcObjects)
  - Serialize the output into a JSON response

• Important class:
  ESRI.ArcGIS.SOESupport.JsonObject
Serialization methods on SOESupport.JsonObject

- **Deserialization (receiving a request)**
  - TryGetArray(string, out object[])
  - TryGetBoolean(string, out bool?)
  - TryGetDate(string, out System.DateTime?)
  - TryGetDouble(string, out double?)
  - TryGetLong(string, out long?)
  - TryGetJsonObject(string, out ESRI.ArcGIS.SOESupportJsonObject)
  - TryGetObject(string, out object)
  - TryGetString(string, out string)

- **Serialization (preparing a response)**
  - AddArray(string, object[])
  - AddBoolean(string, bool)
  - AddDate(string, System.DateTime)
  - AddDouble(string, double)
  - AddJsonObject(string, ESRI.ArcGIS.SOESupportJsonObject)
  - AddLong(string, long)
  - AddObject(string, object)
  - AddString(string, string)
Conversion methods on SOESupport.Conversion

- Conversion()
- ToGeometry(ESRI.ArcGIS.SOESupport.JsonObject, ESRI.ArcGIS.Geometry.esriGeometryType)
- ToGeometry(string, ESRI.ArcGIS.Geometry.esriGeometryType)
-ToJson(ESRI.ArcGIS.Geodatabase.IRecordSet)
-ToJson(ESRI.ArcGIS.Geometry.IGeometry)
-ToJsonObject(ESRI.ArcGIS.Geometry.IGeometry)
- ToSpatialReference(string)

- These are helpful with (de)serialization
Deserialization in the SpatialQueryREST sample
Serialization in the SpatialQueryREST sample

Diagram showing the process of converting geometries to JSON objects and handling the response.
Demo: JSON serialization
Accessing map services

• Avoid MXD-specific ArcObjects (IMap, ILayer, etc.)
  - Carto object model

• Use IMapServerDataAccess to get at underlying data of MSDs
  - IFeatureClass, IRaster, or ITable
Demo: Map service access
Deploying an SOE
Deploying and enabling an SOE

• Browse to the .SOE file in Manager to deploy

• Edit any map service properties and check the SOE

• Manager provides text boxes for typing SOE properties
  - Custom Manager and ArcCatalog property pages available
Demo: Deploying and enabling an SOE
Using an SOE in a client application
Using the SOE in a client application

- Use types designed for asynchronous HTTP requests to Web services
- Different techniques for JavaScript, Flex, and Silverlight
JavaScript clients to REST SOEs

- Use esri.request

- Set up content variable with JSON inputs

- Pass content to esri.request, then work with the response object

- Online SDK has a sample
Demo: Using an SOE in a JavaScript app
Where can I get more help?

- Video of this talk will be online
- SOE doc is in the ArcObjects SDK developer help
- Client examples are in the ArcGIS Web API helps
Download today’s demos

- esriurl.com/spatialqueryrest101
- esriurl.com/milemarker
Please fill out the survey using the DevSummit app

Offering ID: 276