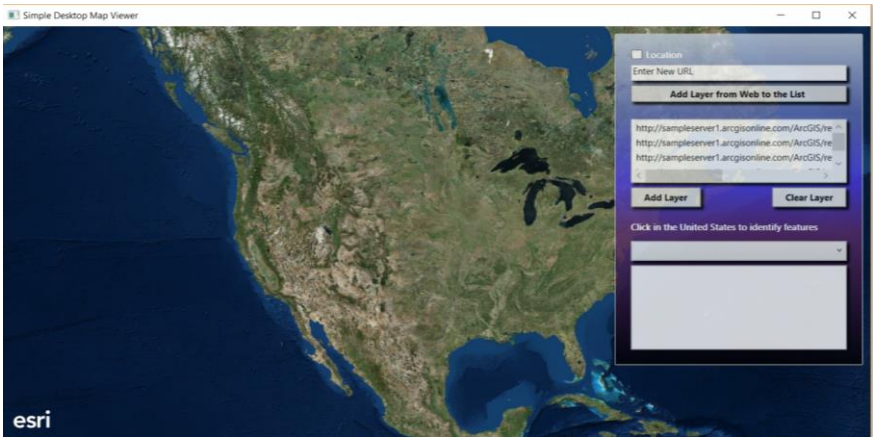


Simple Desktop Map Viewer

ArcGIS Runtime SDK for .NET 10.2.7

Sharvari Sangle

Geog 5562

Table of Contents

Introduction:

Simple Desktop Map viewer is developed using ArcGIS Runtime SDK for .Net 10.2.7. It allows you to add, remove, view, navigate through Web Services. The user can add new web services or use the existing one. It also allows identifying features. This project can be used in companies where they don't want to invest in 100s of license copies for ArcMap but still want some of their employees to have the ability to view the internal services in a very basic viewer. The app can be can customized according to the company's needs. All you need a developer's license and one-time investment in application development.

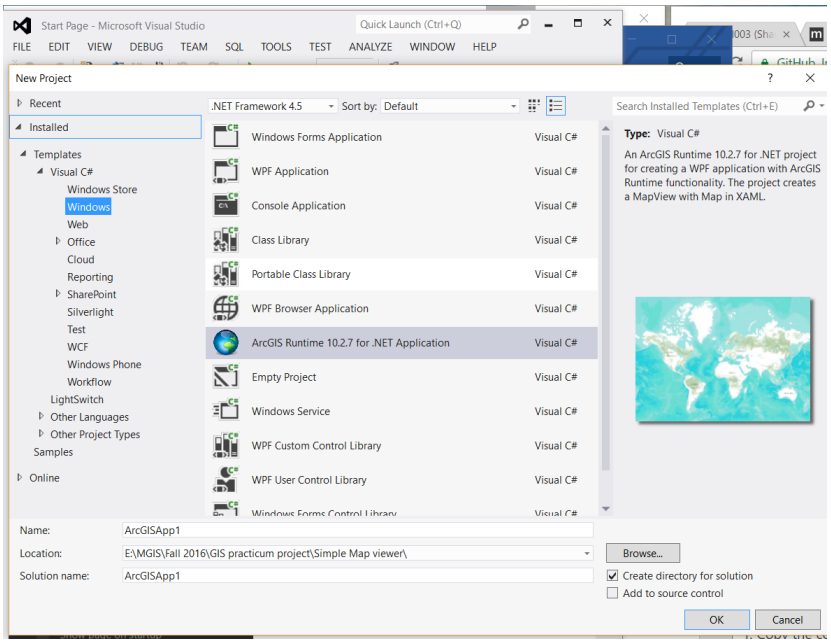
Learning and understanding the concepts:

I choose to develop in ArcGIS Runtime SDK for .NET as I wanted to revisit the concepts and refresh the C# programming. Initially, I spend time on going through the basic concepts of C# and object oriented programming like class, destructors, constructor, methods, inheritance etc. I used the MSDN for reference https://msdn.microsoft.com/en-us/library/mt656686.aspx After going through the basic concepts I went through learning about ArcGIS Runtime for .NET. The system requirements for developing a Desktop application needs the Visual Studio IDE and installation of ArcGIS Runtime SDK.

I installed Microsoft Visual Studio Professional 2012 Version 11.0.61219.00 Update 5 Microsoft .NET Framework Version 4.6.01586 and ArcGIS Runtime for .Net 10.2.7 This is little tricky as you have to first go through the installation of visual studio and then the Runtime. This allows it to add the assemblies and templates for ArcGIS Runtime in the visual studio. If you do not follow the sequence you would end uninstalling and reinstalling the setups. This is the simple part in the installation process. The next step is to setup the NuGet Packages which allows you to get additional controls and utilities. I didn't install this package however eventually I had to install it as some of the ArcGIS Runtime assemblies were missing.

Visual Studio Professional 2012 IDE:

After the install, it is simple to create a project either using a WPF template or using an ArcGIS Runtime template in the projects menu

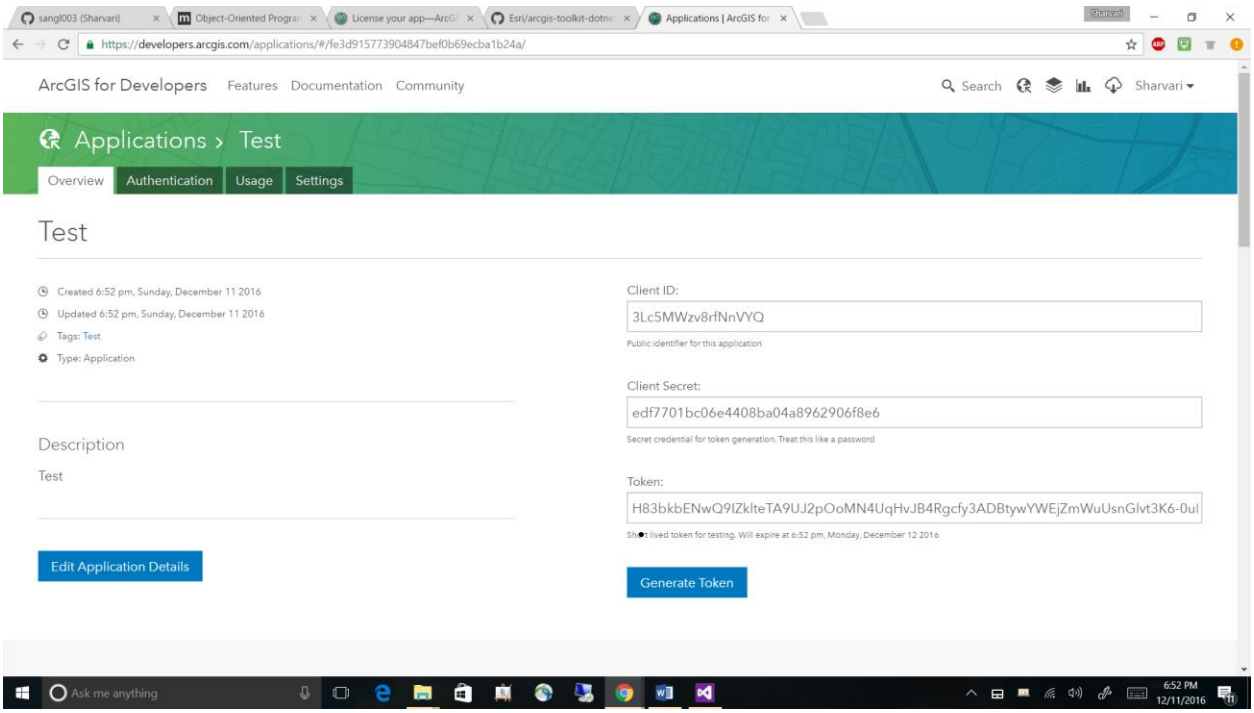


I had some familiarity with the Visual Studio IDE and the different panels like Solutions explorer, Tools, Properties etc. The IDE is really useful as you can drag and drop and design the front view of your app in the design view. This saves a lot of time. You still have the ability to code in the ".xaml" file. This file contains the Extensible Application Markup language. The backend or the logic for the buttons or text box list box etc. can be coded in the ".cs" file. This file contains the classes and different methods in the application. The Solution explorer allows to view the files in a hierarchical order and you can use this to go back and forth from different files.

Licensing the application:

The next step is to license your application using the developer's account and registering the application. You can simply register a new application and once the code is generated. This code can be used in the app.cs file. There are 3 different licensing levels Developers, Basic and Standard. I registered for the free 50 credit/month Development and testing which allows you remove the watermarks from the application and to deploy the app.

The online developer's account looks something like below image



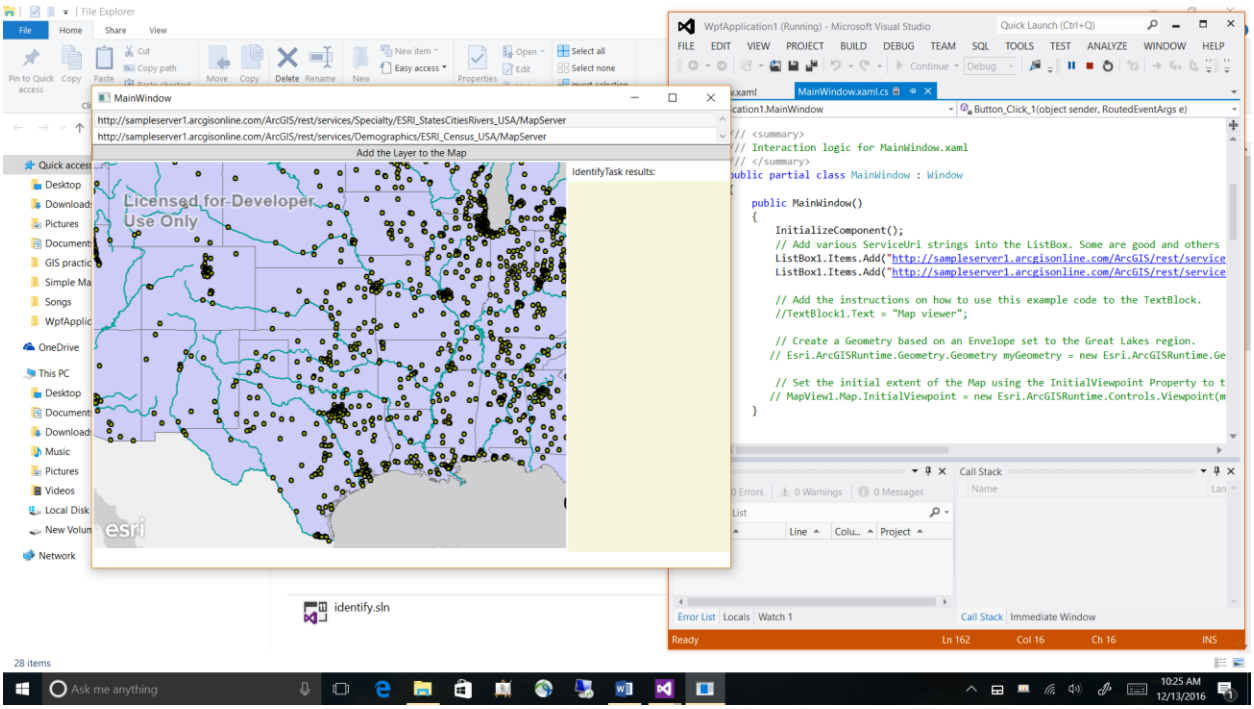
After setting the environment and licensing your app, there is nothing much to do regarding the setup. The next step is to start designing and implementing the application logic.

Designing the app:

1. Initially, I had a thought of having a lot many features than the app has right now like layer list, legend, and print. However, I started to take one step at a time and started with the first sample build your first web map. This helped me understand how the Map viewer objects work. The next step was to learn about dynamic map services and to add services dynamically to the app in runtime. Initially, I used a drop down list and passed the list of Map services. This allowed me to add the services from the list one at a time.

2. The second part was to implement identify tool, this part took a while longer than expected. I used a list of services and hard coded the pop-ups and used the x,y coordinates to access the features properties. This worked well, however when I allowed the user to add new services to the feature this logic didn't work. Now I had to find a way to keep the 'identity tool' generalized and access all the attributes of the feature using the identity operations of a service.

Below image shows the app at this point of time:



By referring the Identify tool sample in the developer's documentation I was able to access the identify operations and query on click for all the layers in the map. This allowed returning multiple features.

3. Allowing user to add web services in a text box. I was pretty clear about adding the new web services to the list box and not to the Map as it would keep the service in the instance of running application and user can still add different layers. The second scenario was to pass the input from the textbox directly to the object of Esri.ArcGISRuntime.Layers.ArcGISDynamicMapServiceLayer

Finally, I was able to add the services to the list and add them to the map. However, this made the map refresh every time and did not allow services to overlap each other. After tweaking the code, I was able to overlay layers. But as the layer is added from a list and not a check box there was no ability to remove a layer. Now I added a button to clear the layer at position 0 so that it will remove the very first layer every time you click it.

4. Adding location services was pretty simple as it uses the system location and displays it on the map. The sample code also allowed to reset the map, panning, etc. but I just used the location service as all other things were not required for this simple app.

5. After having the foundation of the application now I was working on the cosmetic changes in the app. Initially, all the components were on different border tags as I was not able to find the tag in XAML which allowed adding the components on one stack and still keep it mobile responsive. Later I realized I could use Grid tag to do so. The final look of the app is given below:

Conclusion:

This application does not have a lot of features but I have learned a lot about the structure of XAML and ArcGIS Runtime SDK and .Net during the development of this project. I would do some things differently if given a chance. I would use radio buttons or check boxes for selecting the layers. I would add more functionality for adding new data to the app like shape files or zip folders.

My aim was to gain confidence in .NET, XAML, and ArcGIS Runtime SDK so that I could use these skillsets in my professional life. I believe I have gained that knowledge to keep challenging myself and learn more about these languages. It was a great learning experience and I liked the openness of the project and the freedom to focus on what you learn than the immenseness of project.