Deploy Web Appbuidler for ArcGIS based application in MVC and Implementation of Portal for ArcGIS based OAuth 2.0 authentication

Chakresh Sahu, chakresh.sahu@esri.in (Esri India)

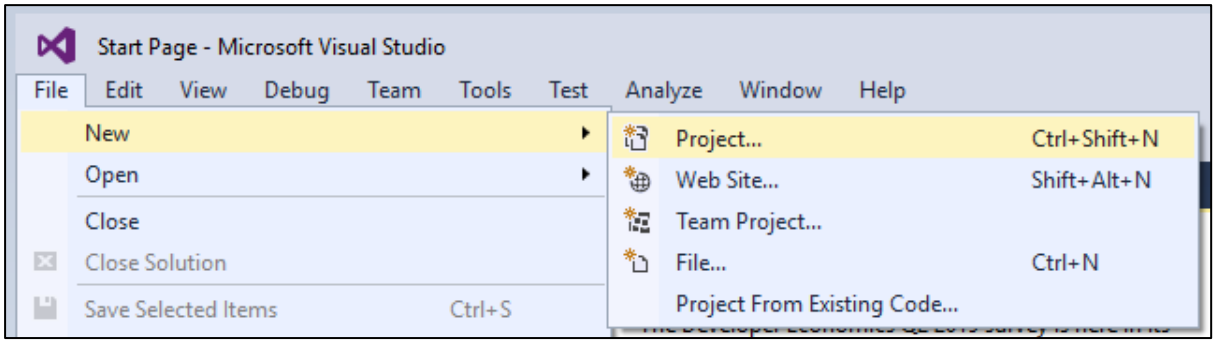
Create A New ASP.NET MVC Project In Visual Studio 2015

Step 1

Open Visual Studio.

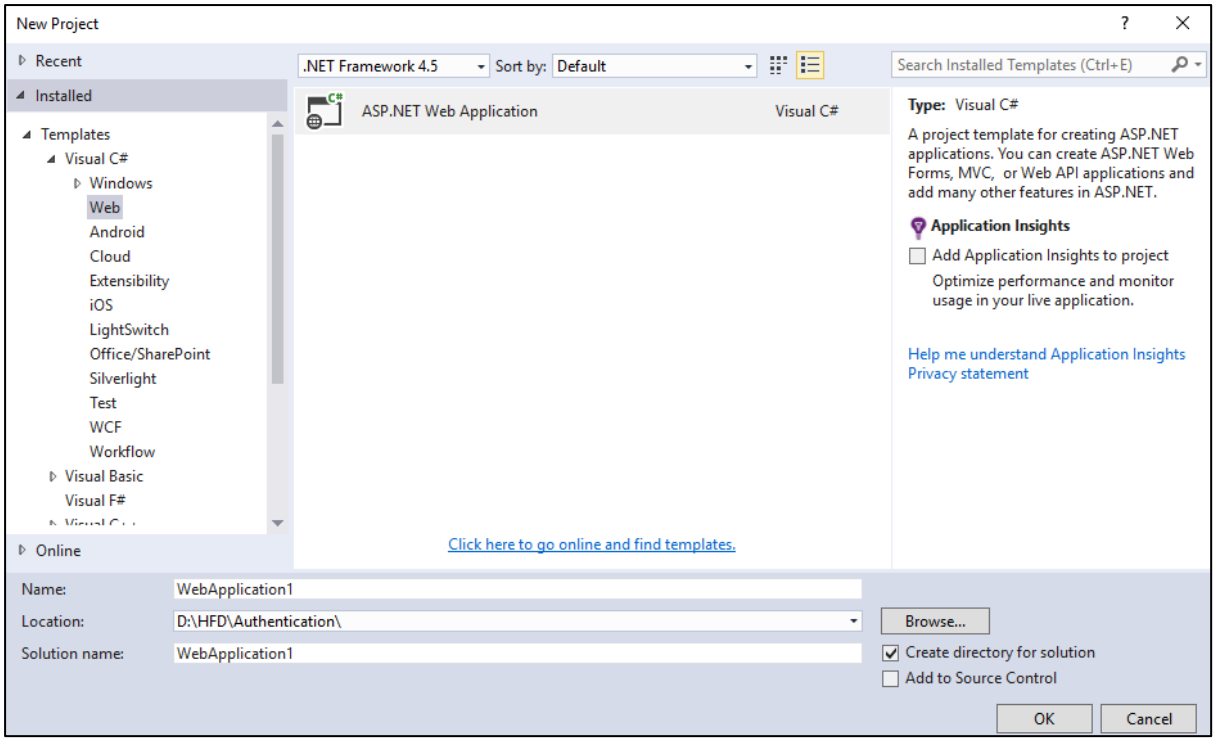
Step 2

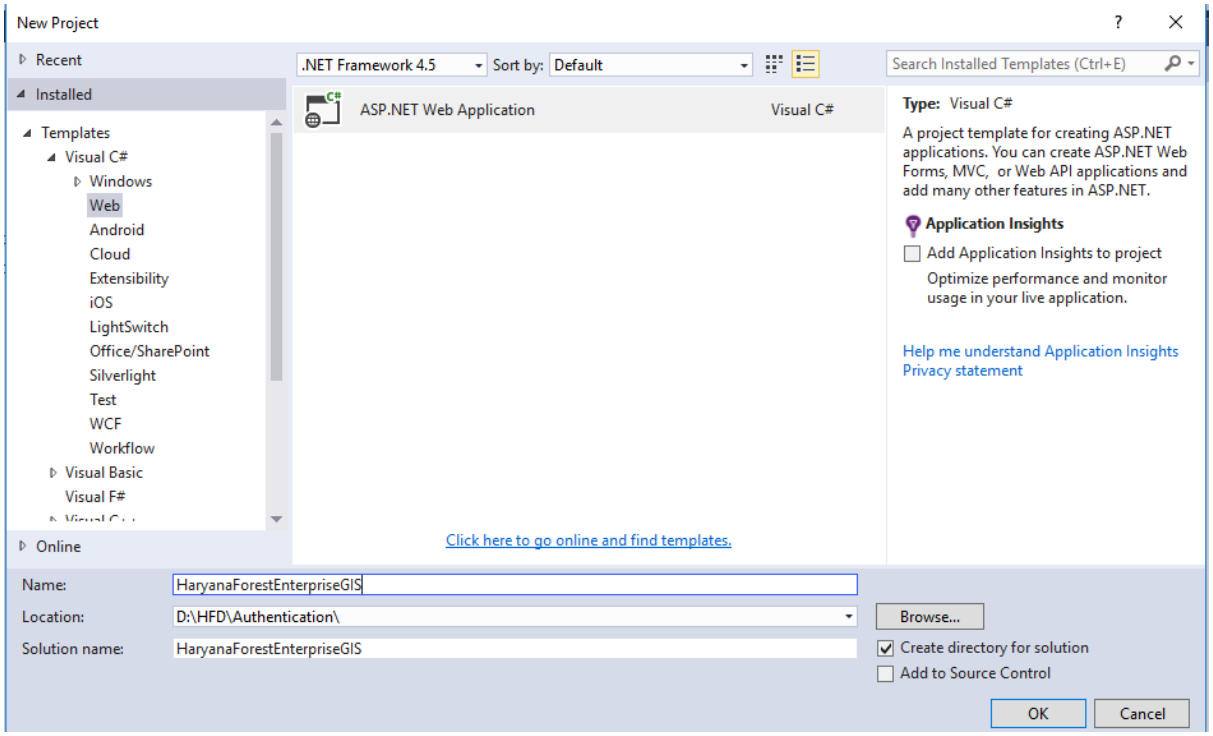
Go to File, New Menu and select "Project...".



Step 3

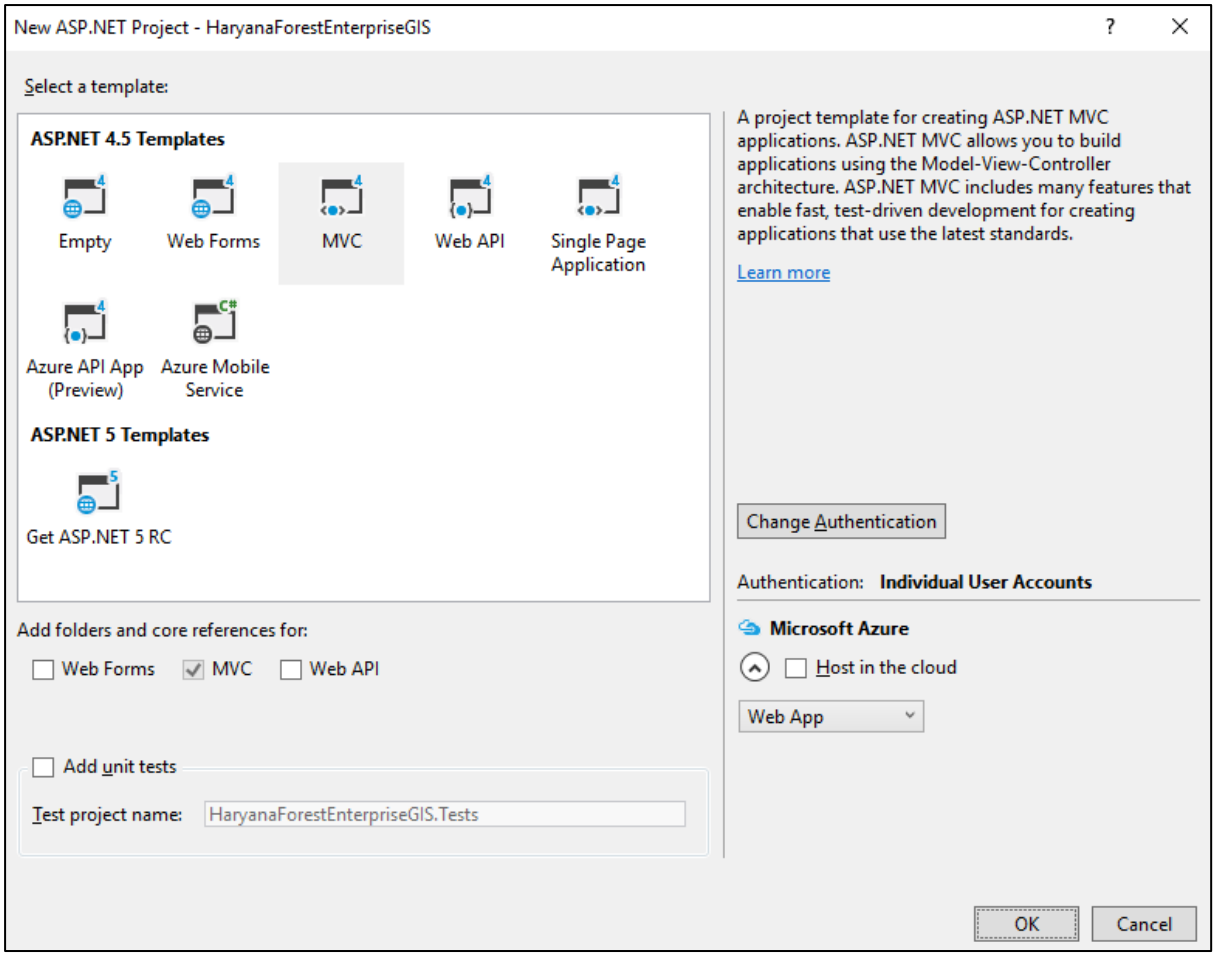
In the "New Project" dialog, select "Templates", Visual C#, then Web and select "ASP.NET Web Application". After that provide the name you want to save and click "OK".





Step 4

In the next screen select ASP.NET Template as "MVC" as shown in the following screen. This screen also offers to include "core references" for WebForms and Web API as well. If you want you can just add those by checking the checkbox. If you also want to add "Unit Tests" for this project, Visual studio will add one for you, if you check "Add unit tests" check box. Click "OK" to open the selected template.



Step 5

That's it. You have successfully created an ASP.NET MVC Web Project. It will look like below, as you can see, it will consist of all the necessary files to start the development. You can see this in the solution explorer.

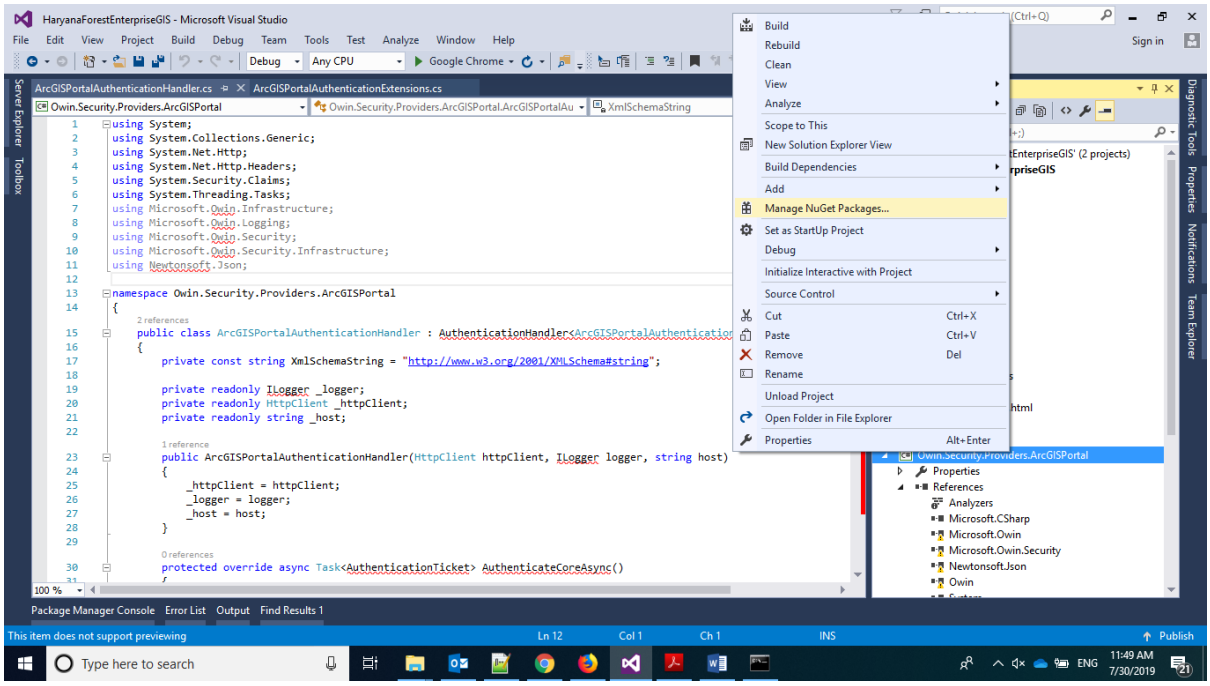
Server-side authentication using ArcGIS Portal

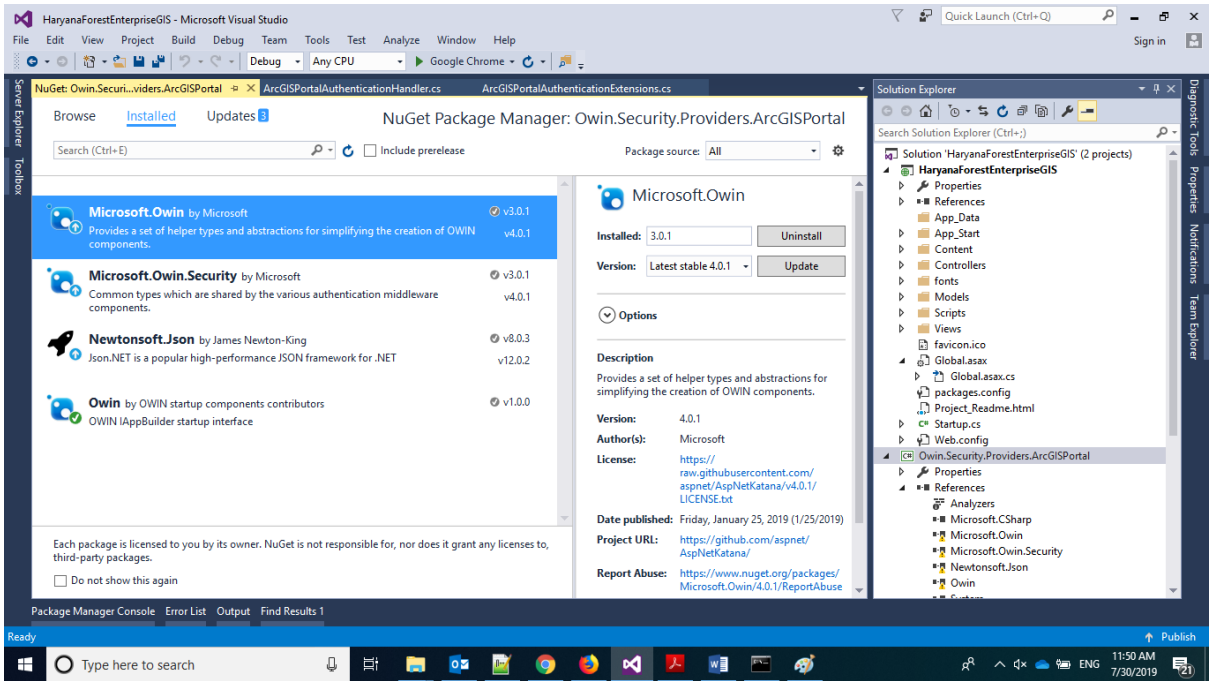
Add Owin.Security.Providers.ArcGISPortal project

1. Download and include the Owin.Security.Providers.ArcGISPortal project in your solution

https://github.com/gavinharriss/OwinOAuthProviders/tree/master/src/Owin.Security.Providers.ArcGISPortal

2. Build the solution and manage the package through NuGet Packages.





3. Add the missing references from downloaded packages.
4. Build again the project.

Enable Authentication in Solution

1. Open Startup.Auth.cs file.

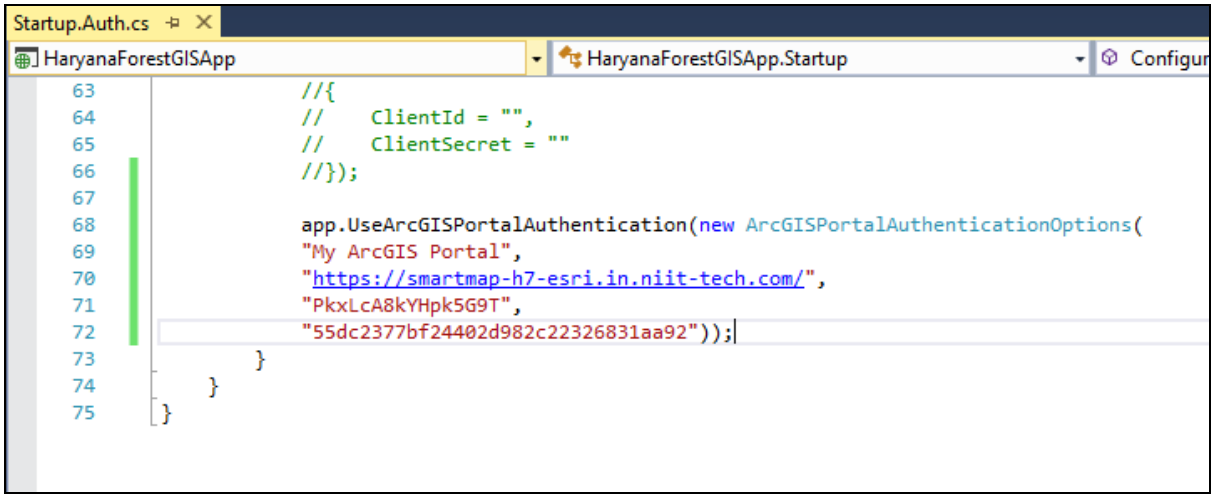
Add following reference in file

```
using Owin.Security.Providers.ArcGISPortal;
```

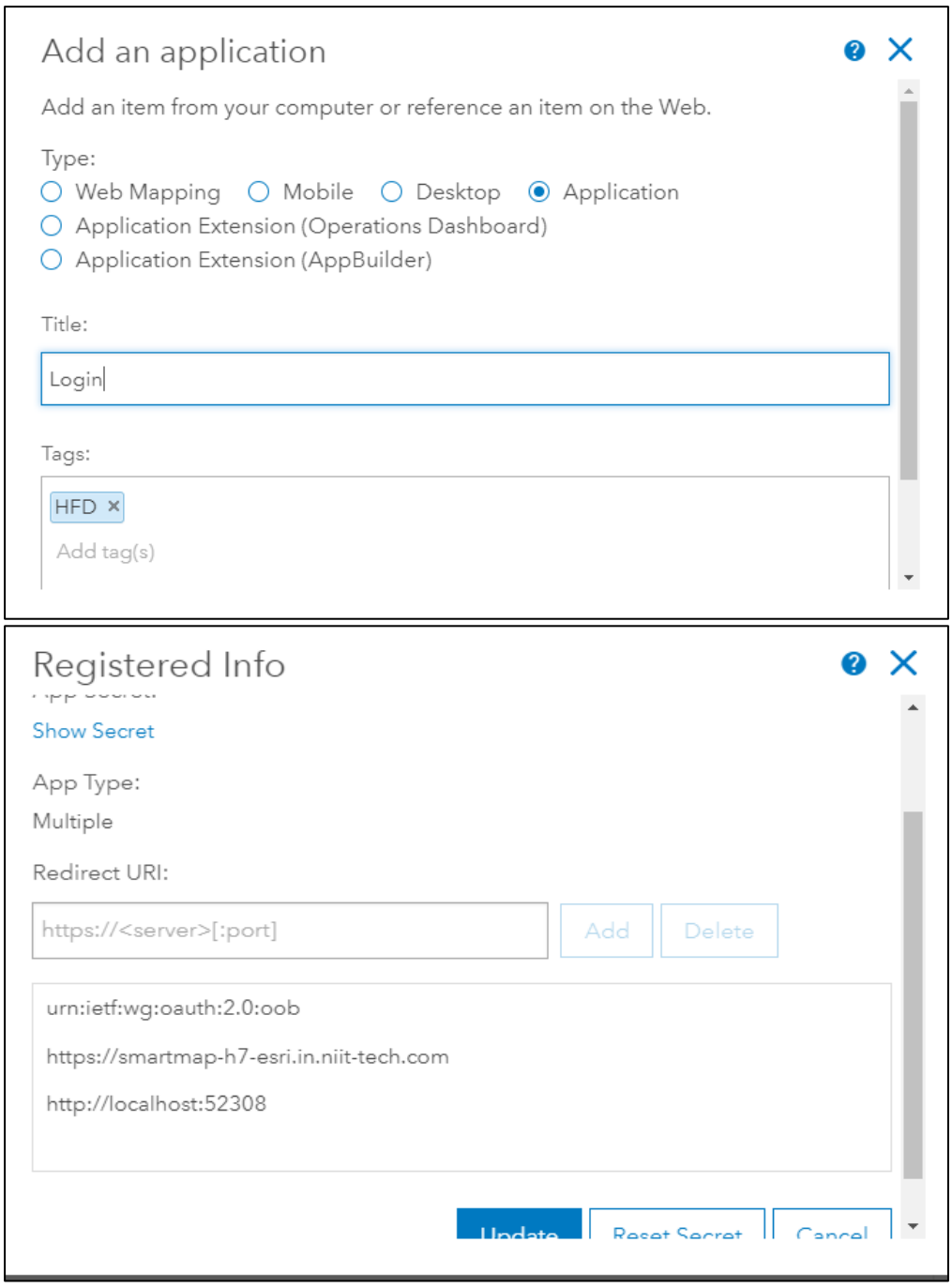
Add following code in file

```
app.UseArcGISPortalAuthentication(new ArcGISPortalAuthenticationOptions(
 "My ArcGIS Portal",
  "https://arcgisportal.mydomain.com/",
  "ClientID",
  "Client Secret"));
```

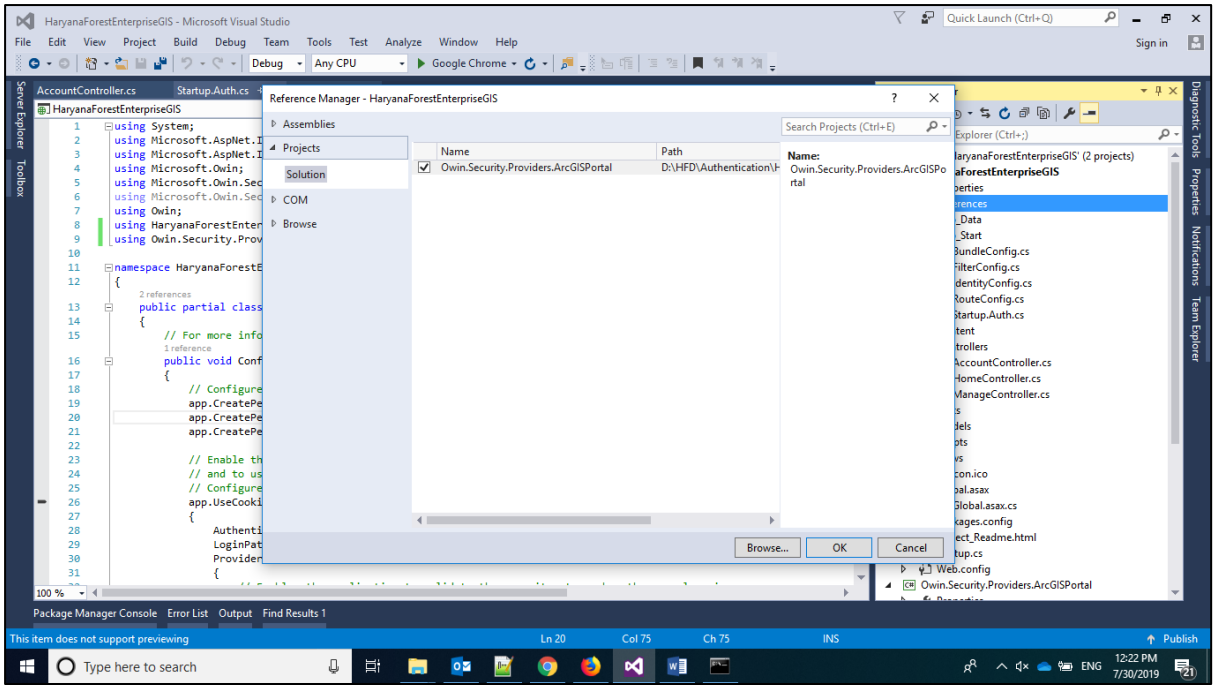
For example



2. To get Client ID and Client Secret, register the application in Portal for ArcGIS.



3. Add Owin.Security.Providers.ArcGISPortal project reference in solution.



4. To change the portal url in solution. Open ArcGISPortalAuthenticationOptions.cs file and following code according the portal url.

```
private const string AuthorizationEndPoint =
"arcgis/sharing/rest/oauth2/authorize/";
private const string TokenEndpoint = "arcgis/sharing/rest/oauth2/token/";
private const string UserInfoEndpoint = "arcgis/sharing/rest/community/self";
```

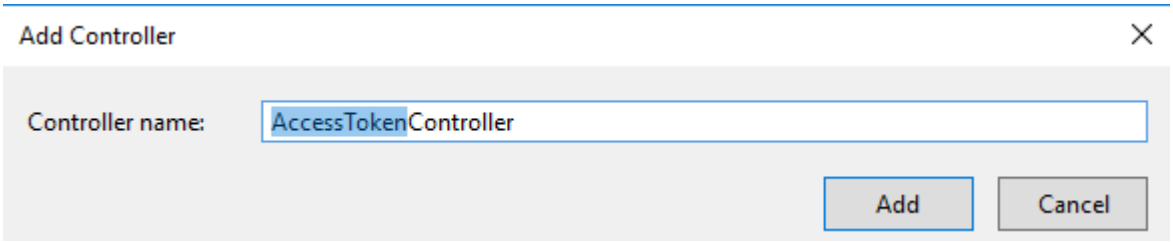
For example in our case it will be following.

Web API Controller

In order to add a Web API Controller you will need to Right Click the Controllers folder in the Solution Explorer and click on Add and then Controller.

Now from the Add Scaffold window, choose the Web API 2 Controller – Empty option as shown below.

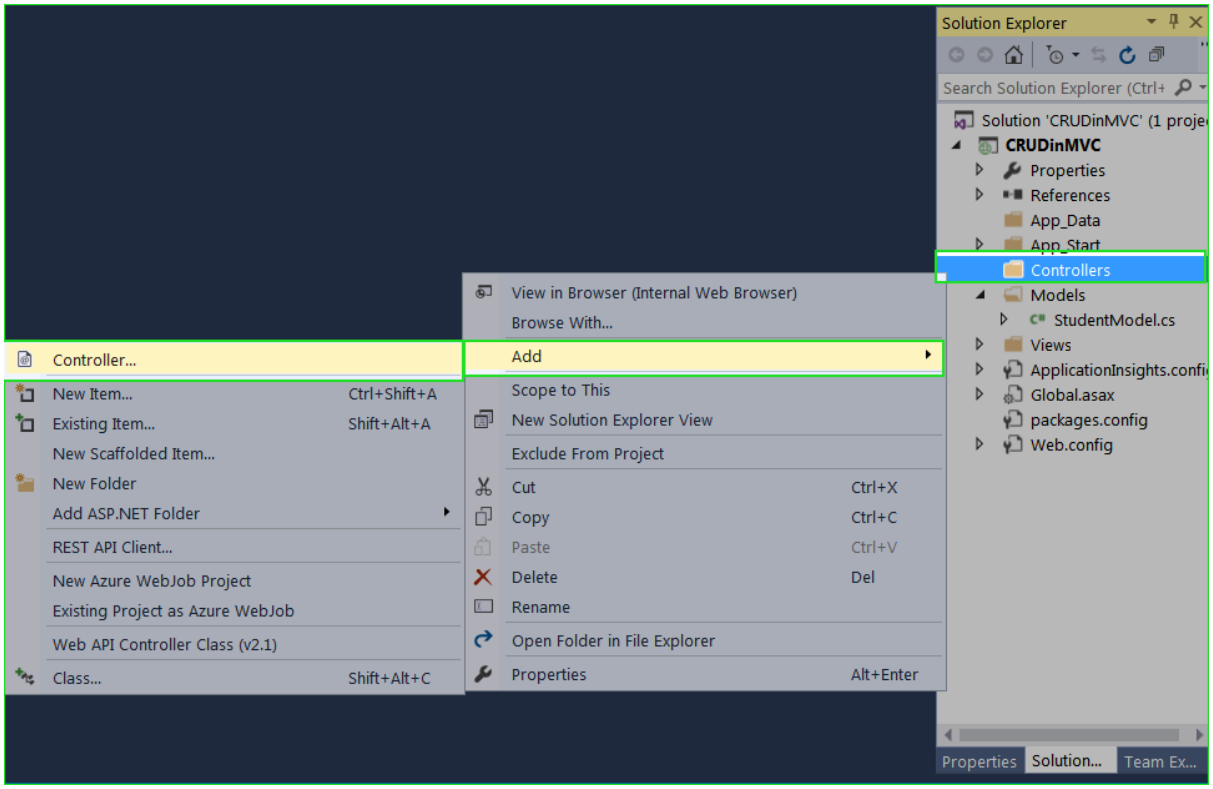
Then give it an AccessToken name and click OK



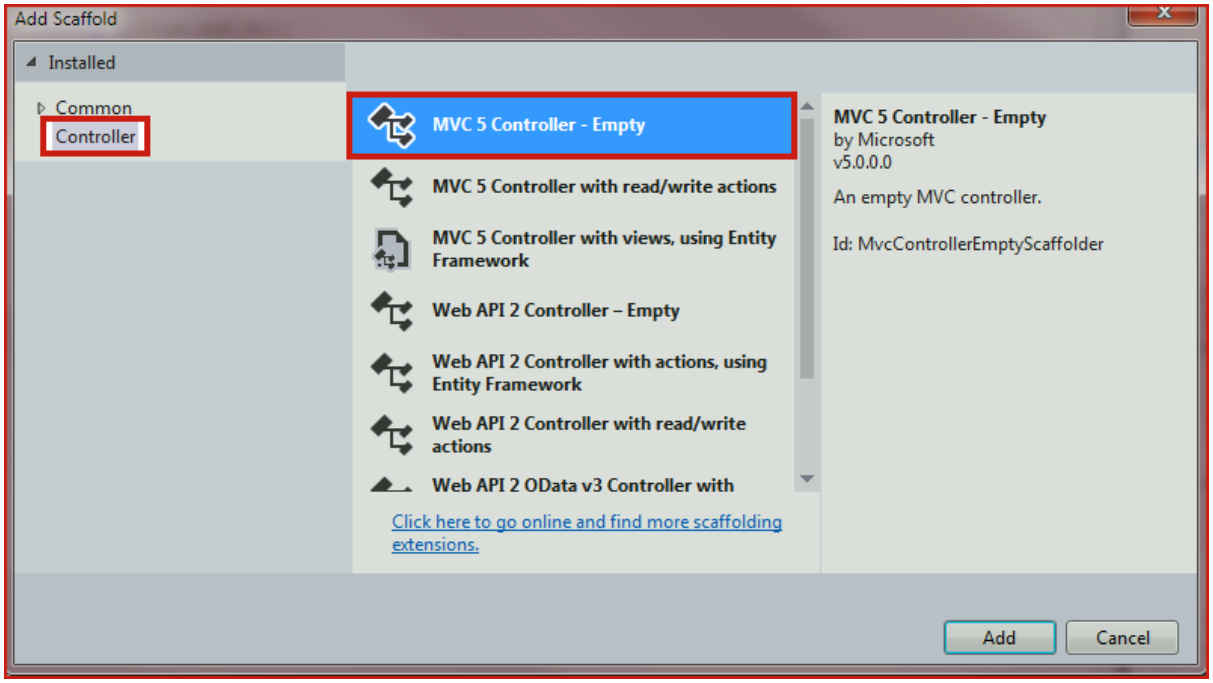
Difference between ApiController and Controller in ASP.NET MVC Note If you have worked with ASP.NET MVC, then you are already familiar with controllers. They work similarly in Web API, but controllers in Web API derive from the ApiController class instead of Controller class. The first major difference you will notice is that actions on Web API controllers do not return views, they return data.

Adding a Controller and View Page in ASP.NET MVC 5

1. Go to Solution Explorer and **Right click on Controllers** folder **Add Controller**

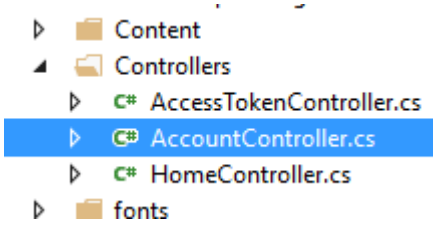


2. Select **MVC 5 Controller – Empty** and click on **Add** button



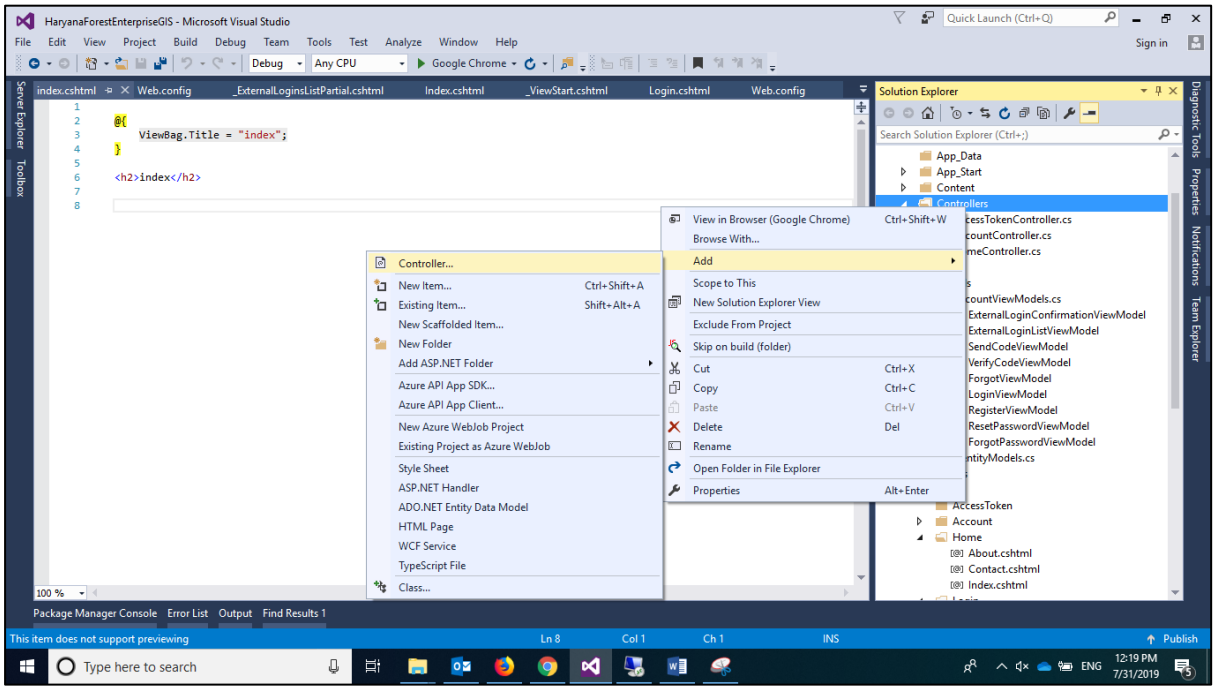
3. Give Controller Name to **AccountController**. *"Controller"* suffix must be added in controller name. Click **Add** button to add controller.

4. Your Item Controller will look like this.

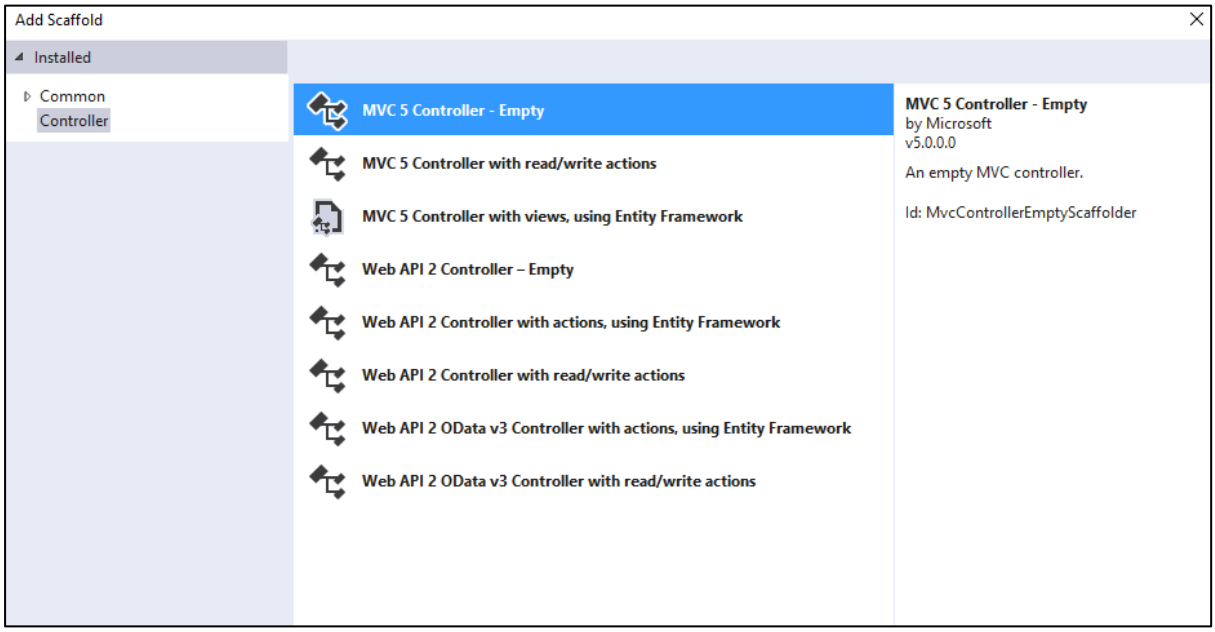


Create Login View in Application

1. Go to Solution Explorer and **Right click on Controllers** folder **Add Controller**

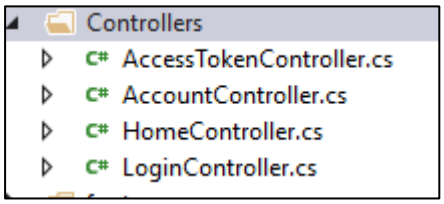


2. Select **MVC 5 Controller – Empty** and click on **Add** button



3. Give Controller Name to **LoginController**. *"Controller"* suffix must be added in controller name. Click **Add** button to add controller.

4. Your Item Controller will look like this.

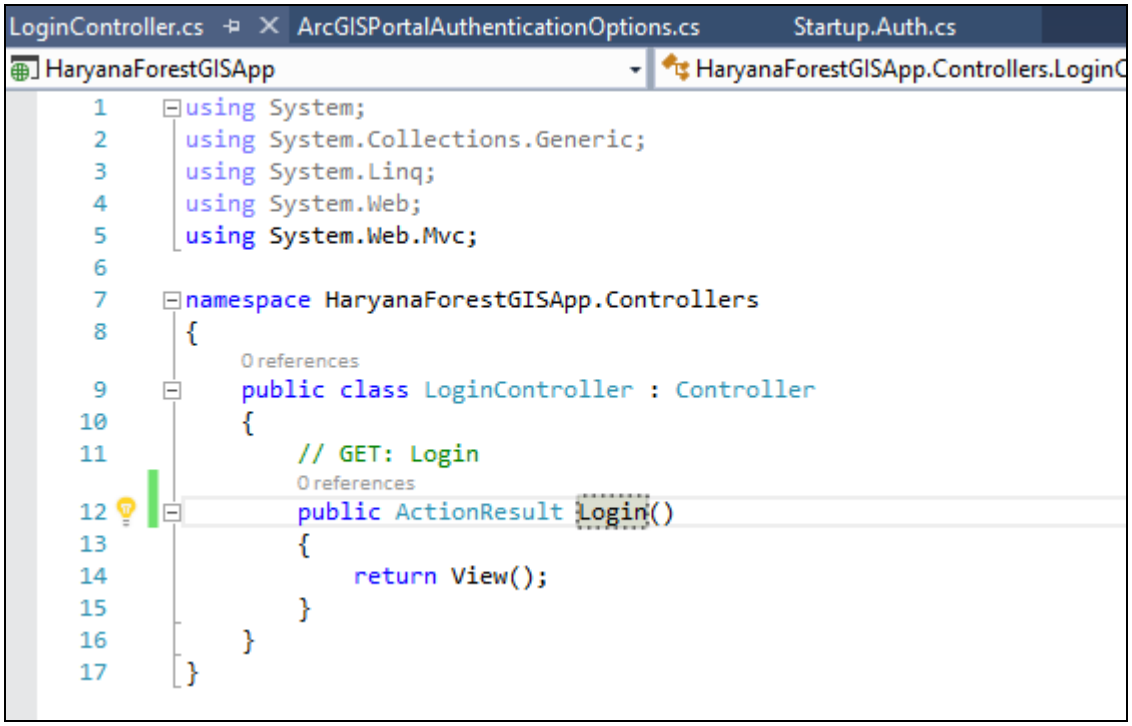


This controller will execute when user browse following url:
http://localhost:52308/Login/Index

In the above link, **Login** is `controller` and **Index** is `action method`.

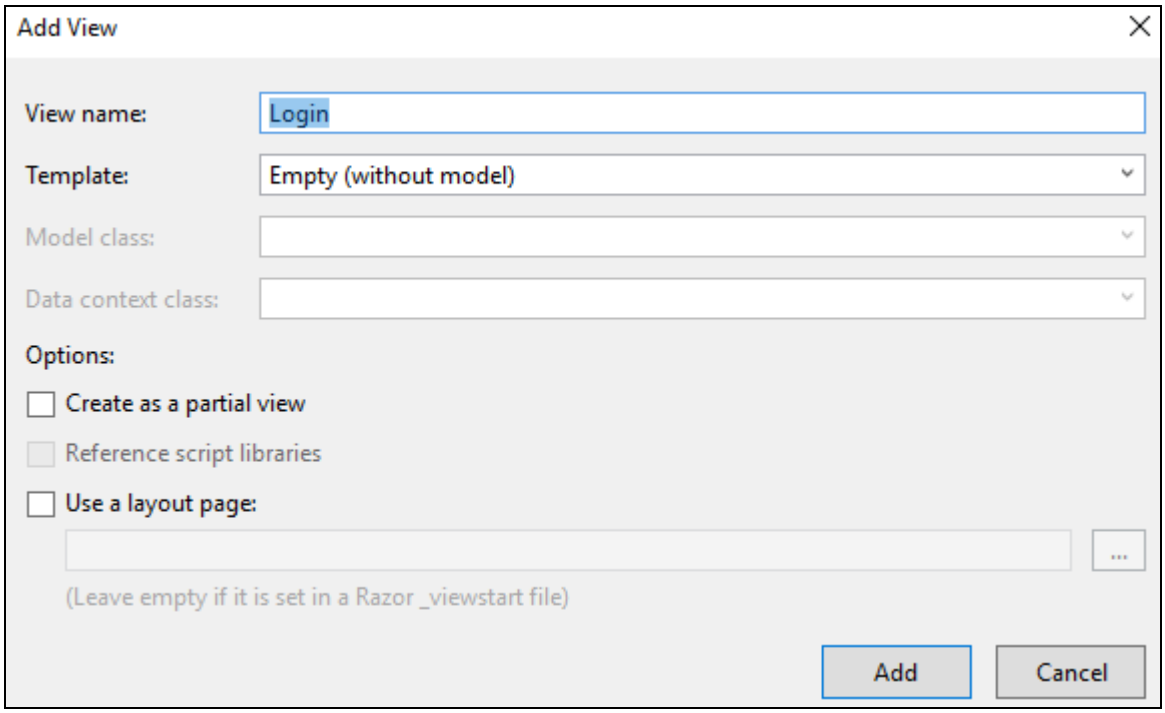
When user click on the link, it will search for `LoginController` with `Index()` Action method. Controllers keeps action method that gets executed when user needs them. There must be a **ViewPage** **`Index.cshtml`** in Item Folder, otherwise you will get error message.

Modify Index method to Login


LoginController.cs
ArcGISPortalAuthenticationOptions.cs
Startup.Auth.cs
HaryanaForestGISApp
HaryanaForestGISApp.Controllers.LoginC
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace HaryanaForestGISApp.Controllers
{
0 references
public class LoginController : Controller
{
// GET: Login
0 references
public ActionResult Login()
{
return View();
}
}
}

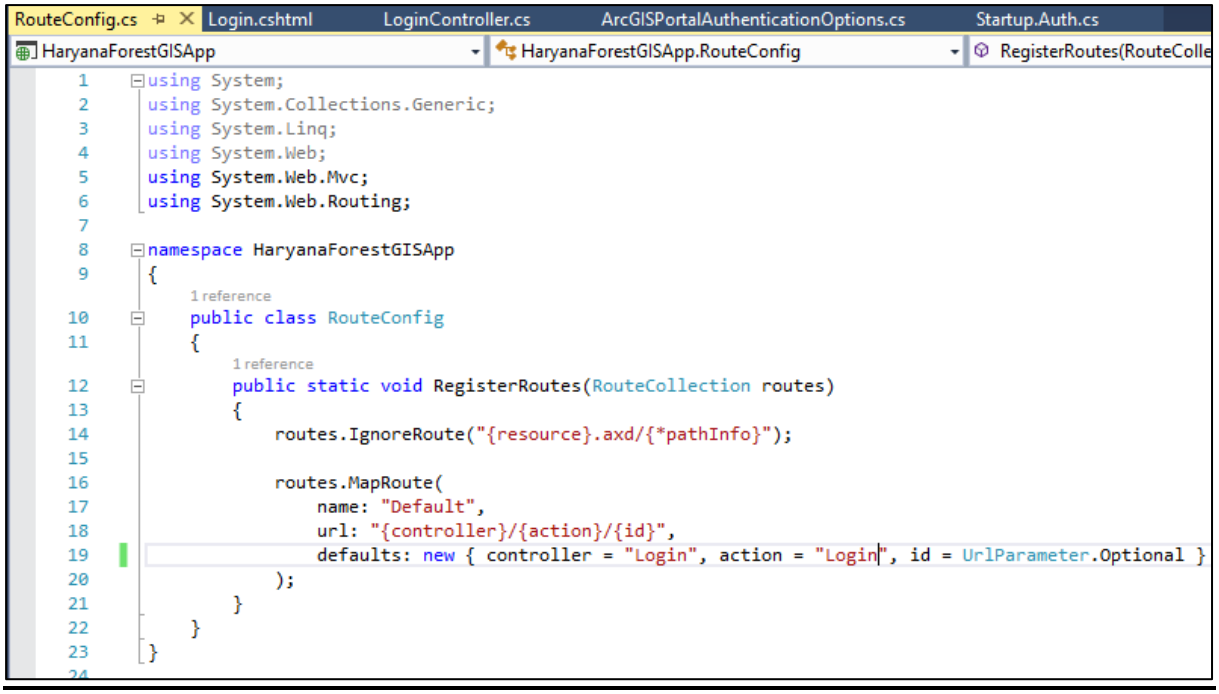
Adding a ViewPage for Login method

1. Right click on Login Action Method and select **Add View**



2. Your Index view page is added. You can see here.

3. In RouteConfig change your default route to newly created Login route.



4. Open Login.cshtml and modify code to this

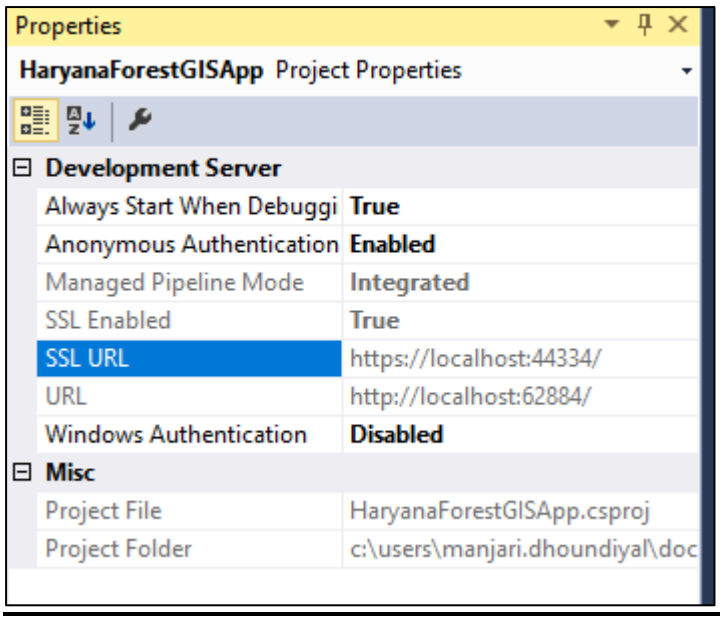
```
@using HaryanaForestEnterpriseGIS.Models
@model LoginViewModel
@{
    ViewBag.Title = "Login";
}

<div class="col-md-4">
    <section id="socialLoginForm">
        @Html.Partial("_ExternalLoginsListPartial", new ExternalLoginListViewModel {
ReturnUrl = ViewBag.ReturnUrl })
    </section>
</div>
```

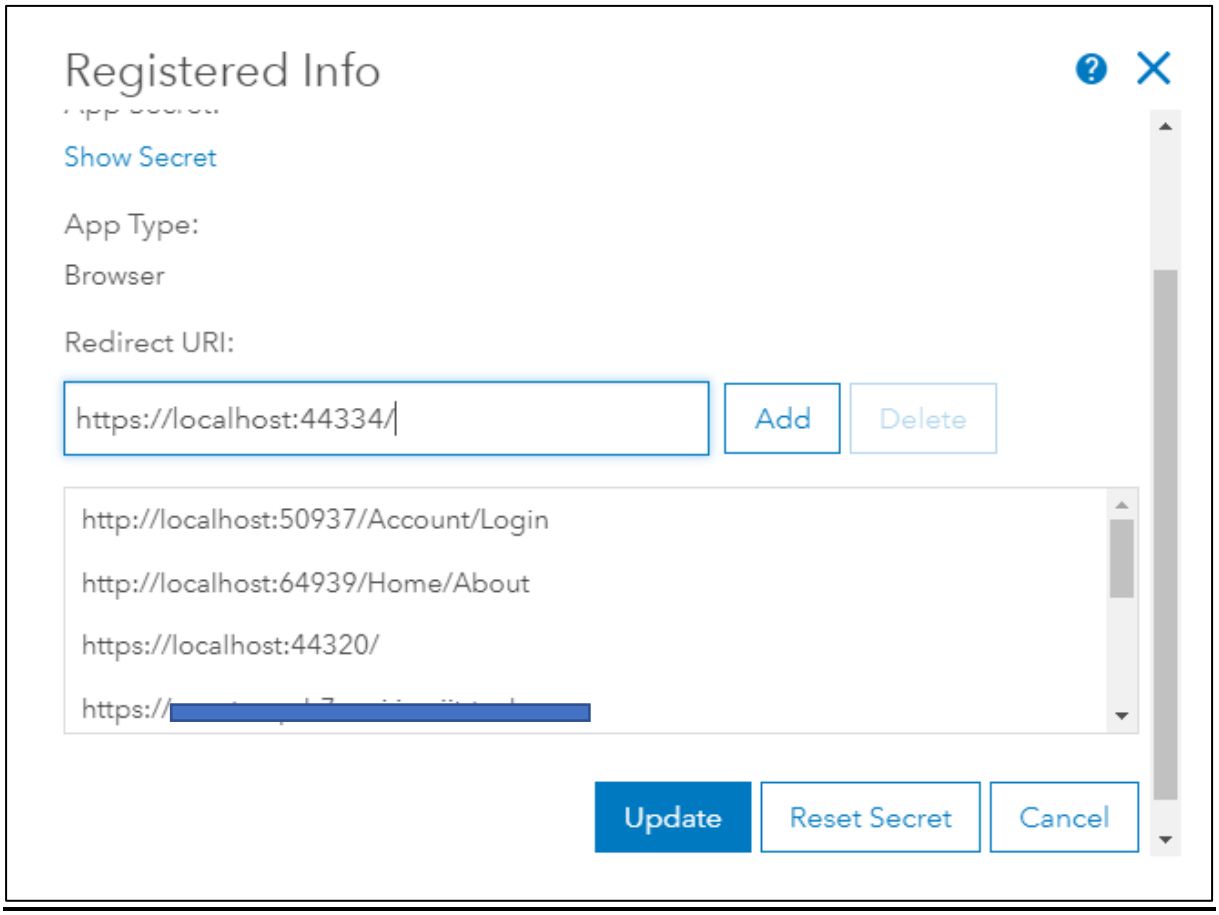
5. Copy _ExternalLoginsListPartial.cshtml file to Login folder in view.

6. Select project in Visual Studio and press F4. Project properties window will open where set ‘SSL Enabled’ to True. You will get https url after this which we will use for browsing.

In below example it is “https://localhost:44334”

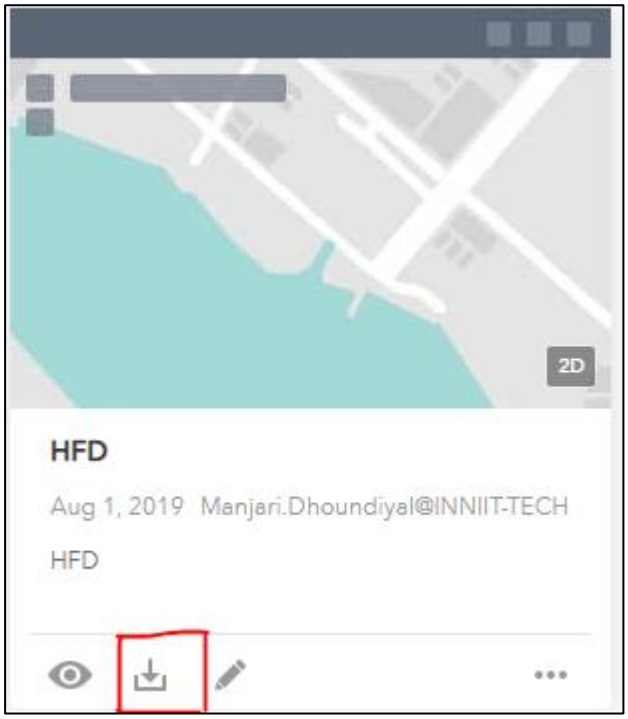


7. In registered application add this url in redirect info.



Integrating WebAppBuilder Application

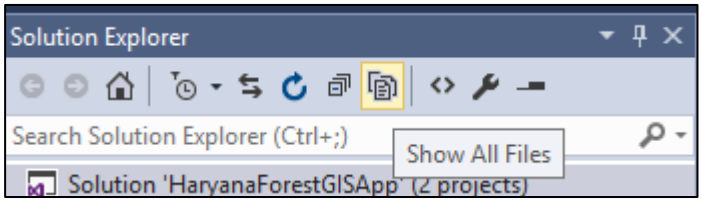
1. Download application in webAppBuilder.



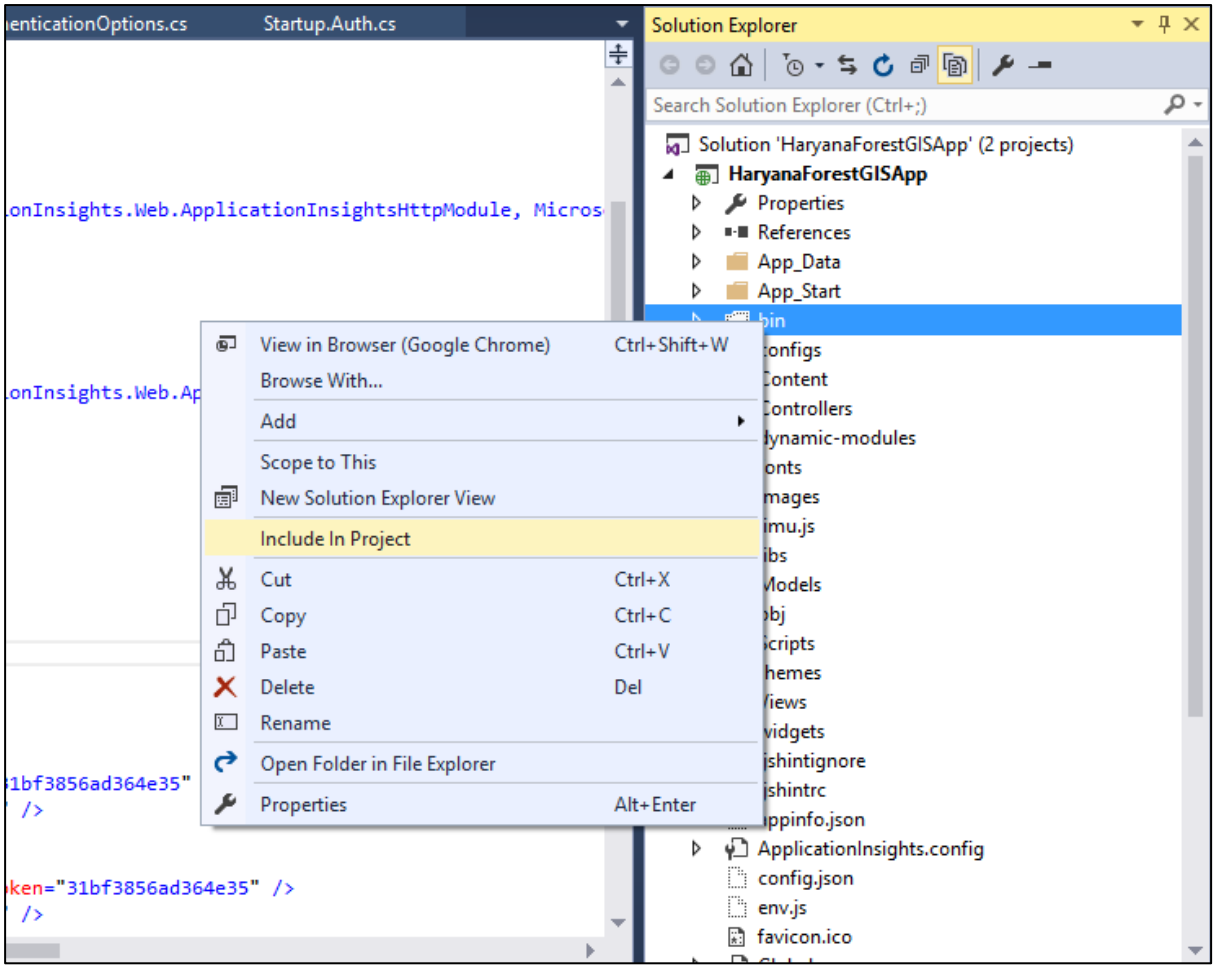
2. Extract Zip File to temporary location and copy its content in MVC project folder.
3. Do not copy web.config file in the mvc project instead open already existing web.config file and add following code inside <system.webServer>

```
        <staticContent>
          <!-- Configure site to serve JSON files -->
          <remove fileExtension=".json" />
          <mimeMap fileExtension=".json" mimeType="application/json" />
          <!-- Configure site to serve font files -->
          <remove fileExtension=".otf" />
          <mimeMap fileExtension=".otf" mimeType="font/otf" />
        </staticContent>
```

4. Click show all files as shown below.



5. Include all of the Web App Folders and files



6. Now move following files in script folder
    - Init.js
    - Env.js
    - simpleLoader.js

7. In BundleConfig.cs in App_Start add the "~/jimu.js/css/jimu-ie.css" script to an existing Css bundle as shown below

```
                    "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from. Then, when you're
            // ready for production, use the build tool at http://modernizr.com to pick only the tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
                      "~/Scripts/bootstrap.js",
                      "~/Scripts/respond.js"));

            bundles.Add(new StyleBundle("~/Content/css").Include(
                      "~/Content/bootstrap.css",
                      "~/Content/site.css",
                      "~/jimu.js/css/jimu-ie.css"));
        }
    }
}
```

8. In Index.cshtml view replace the following code

```
@{
    ViewBag.Title = "ArcGIS Web Application";
    @Styles.Render("~/Content/css") //this is the bundle config I added the
jimu-je.css to
    Layout = null; // set layout to null if you just want the pure web app,
regardless of layout.cshtml view
}
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
    <meta http-equiv="X-UA-Compatible" content="IE=EDGE" />
    <title>ArcGIS Web Application</title>
    <link rel="shortcut icon" href="images/shortcut.png">
</head>

<body class="claro jimu-main-font">
    <div id="main-loading">
        <!-- This is section you can modify to customize the loading page -->
        <div id="app-loading"></div>
        <div id="loading-gif"></div>
        <!-- ////////////////////    END    ////////////////////////////// -->
        <div id="ie-note" style="display:none;">
            <div class="hint-title">Error</div>
            <div class="hint-img">Your browser is currently not supported.</div>
            <p class="hint-text">
                <span>
                    Please note that creating presentations is not supported in
Internet Explorer versions 6, 7.
                </span>
                <br>
                <span>
                    We recommend upgrading to the latest Internet Explorer,
Google Chrome, or Firefox.
                </span>
                <br>
```

```
                <span>
                    If you are using IE 8 or later, make sure you turn off
"Compatibility View".
                </span>
            </p>
        </div>
    </div>
    <div id="main-page">
        <a role="link" id="trapLinkNode" tabindex="0">Trap Link Node</a>
        <div id="jimu-layout-manager"></div>
        <div id="skipContainer"></div>
    </div>
    @Scripts.Render("~/Scripts/env.js");
    @Scripts.Render("~/Scripts/simpleLoader.js");
    @Scripts.Render("~/Scripts/init.js");

</body>

</html>
```

9. Open env.js and do following changes in getPath() method

```
var fullPath, path;

    var url = window.location.href.split('/');
    path = url[0] + '//' + url[2];
return path;
```

For eg:

```
192
193   function getPath() {
194       var fullPath, path;
195
196       var url = window.location.href.split('/');
197       path = url[0] + '//' + url[2];
198       return path;
199   }
```

10. Open init.js file and replace window.apiUrl to target Javascript API as shown below and add '\' in window path url

```
var esriJsLib = 'https://js.arcgis.com/3.29/'
resources = resources.concat([
  esriJsLib + 'dojo/resources/dojo.css',
  esriJsLib + 'dijit/themes/claro/claro.css',
  esriJsLib + 'esri/css/esri.css',
  esriJsLib + 'dojox/layout/resources/ResizeHandle.css',
  window.path + '/jimu.js/css/jimu-theme.css',
  window.path + '/libs/caja-html-sanitizer-minified.js',
  window.path + '/libs/moment/twix.js',
  window.path + '/libs/Sortable.js',

  window.path + '/libs/cropperjs/cropperjs.js',
  window.path + '/libs/cropperjs/cropper.css',
  //because we have jimu/dijit/GridLayout dijit, so we import this css here
  window.path + '/libs/goldenlayout/goldenlayout-base.css',
  window.path + '/libs/goldenlayout/goldenlayout-light-theme.css'
]);
```

11. In init,js only update your localhost url :

```
var localhostMVC;
var url = window.location.href.split('/');
localhostMVC = "https://localhost:someportnumber/";
```

Below is the screenshot for the same

```
159         } else {
160             var localhostMVC;
161             var url = window.location.href.split('/');
162             localhostMVC = "https://localhost:44320/";
163             dojoConfig.baseUrl = window.apiUrl + 'dojo';
164             dojoConfig.packages = [{
165                 name: "widgets",
166                 location: localhostMVC + "widgets"
167             }, {
168                 name: "jimu",
169                 location: localhostMVC + "jimu.js"
170             }, {
171                 name: "themes",
172                 location: localhostMVC + "themes"
173             }, {
174                 name: "libs",
175                 location: localhostMVC + "libs"
176             }, {
177                 name: "dynamic-modules",
178                 location: localhostMVC + "dynamic-modules"
179             }, {
180                 name: "configs",
181                 location: localhostMVC + "configs"
182             }];
183
184             resources.push(window.apiUrl + 'init.js');
185         }
186
```

12. Update the _loadPolyfills() method with the code shown below

```
window.appInfo.appPath = localhostMVC;
```

13. In env.js change prePath value to your localhost as shown in screenshot below

```
prePath = "https://localhost:44334/"
tests = [{
  test: window.console,
  failure: prePath + "libs/polyfills/console.js",
  callback: completeCb
}, {
  test: ap.indexOf && ap.lastIndexOf && ap.forEach && ap.every && ap.some &&
    ap.filter && ap.map && ap.reduce && ap.reduceRight,
  failure: prePath + "libs/polyfills/array.generics.js",
  callback: completeCb
}, {
  test: fp.bind,
  failure: prePath + "libs/polyfills/bind.js",
  callback: completeCb
}, {
  test: Date.now,
  failure: prePath + "libs/polyfills/now.js",
  callback: completeCb
}, {
  test: sp.trim,
  failure: prePath + "libs/polyfills/trim.js",
  callback: completeCb
}, {
  test: false,
  failure: prePath + "libs/polyfills/FileSaver.js",
  callback: completeCb
}, {
  test: typeof Blob !== 'undefined',
```

14. In ConfigLoader.js , Line 520 change “this.configFile” value to config path

```
510         this.configFile = "https://localhost:44334/config.json";
511       return xhr(this.configFile, {handleAs: 'json'}).then(lang.hitch(this, function(appConfig){
512         tokenUtils.setPortalUrl(appConfig.portalUrl);
513         if(appConfig.portalUrl){
514           window.portalUrl = appConfig.portalUrl;
515         }
```

Now you have successfully integrated WAB application in MVC.

15. Open Home controller and add [Authorize] in Index View. This will restrict everyone to view WAB application without logging in.

```
[Authorize]
0 references
public ActionResult Index()
{
    return View();
}
```

16. In Startup.Auth.cs change the redirection path to the login page when someone tries to access WAB application without logging in

```
AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
LoginPath = new PathString("/Login/Login"),
Provider = new CookieAuthenticationProvider
```

Adding SignOut functionality through WAB Application

1. Add logout in links which is in app config in application and set it's url to starting login page as shown below :

```
"links": [
  {
    "label": "Logout",
    "url": "https://localhost:44334"
  }
],
```

2. Replace code in Login controller with following code

```
using Microsoft.AspNet.Identity;
using Microsoft.Owin.Security;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace HaryanaForestGISApp.Controllers
{
    public class LoginController : Controller
    {
        // GET: Login
        public ActionResult Login()
        {
            bool LoggedIn = (System.Web.HttpContext.Current.User != null) &&
System.Web.HttpContext.Current.User.Identity.IsAuthenticated;
            if (LoggedIn)
            {

AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);
                return View();
            }
            else
            {
                return View();
            }
        }
        private IAuthenticationManager AuthenticationManager
        {
            get
            {
                return HttpContext.GetOwinContext().Authentication;
            }
        }
    }
}
```

3. Now browse your application, you will have to select ArcGIS Portal. After that it will redirect to ArcGIS Login Page. If correct credentials are provided then it will redirect to

WAB application. There will be logout link in application, on clicking that you will be redirected back to login page.