

External Authentication Services with ASP.NET Web API (C#)

01/28/2019 • 16 minutes to read • Contributors  all

In this article

[Using External Authentication Services](#)

[Create a Sample Web Application](#)

[Enabling Facebook authentication](#)

[Enabling Google Authentication](#)

[Enabling Microsoft Authentication](#)

[Enabling Twitter Authentication](#)

[Additional Information](#)

Visual Studio 2017 and ASP.NET 4.7.2 expand the security options for [Single Page Applications](#) (SPA) and [Web API](#) services to integrate with external authentication services, which include several OAuth/OpenID and social media authentication services: Microsoft Accounts, Twitter, Facebook, and Google.

In this Walkthrough

- [Using External Authentication Services](#)
- [Creating the Sample Web Application](#)
- [Enabling Facebook Authentication](#)
- [Enabling Google Authentication](#)
- [Enabling Microsoft Authentication](#)
- [Enabling Twitter Authentication](#)
- [Additional Information](#)
 - [Combining External Authentication Services](#)
 - [Configuring IIS Express to use a Fully Qualified Domain Name](#)
 - [How to Obtain your Application Settings for Microsoft Authentication](#)
 - [Optional: Disable Local Registration](#)

Prerequisites

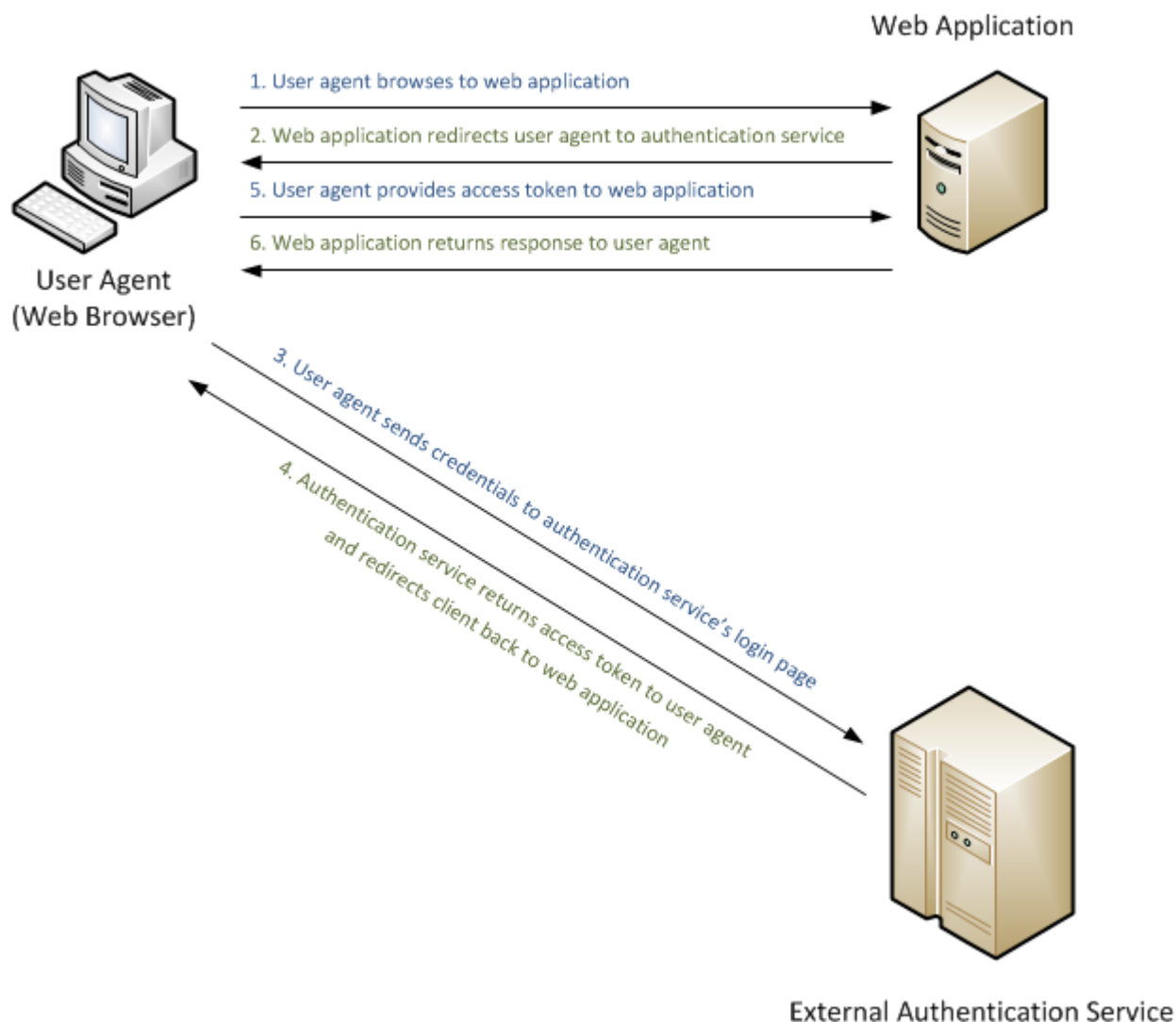
To follow the examples in this walkthrough, you need to have the following:

- Visual Studio 2017
- A developer account with the application identifier and secret key for one of the following social media authentication services:
 - Microsoft Accounts (<https://go.microsoft.com/fwlink/?LinkID=144070>)
 - Twitter (<https://dev.twitter.com/>)
 - Facebook (<https://developers.facebook.com/>)
 - Google (<https://developers.google.com/>)

Using External Authentication Services

The abundance of external authentication services that are currently available to web developers help to reduce development time when creating new web applications. Web users typically have several existing accounts for popular web services and social media websites, therefore when a web application implements the authentication services from an external web service or social media website, it saves the development time that would have been spent creating an authentication implementation. Using an external authentication service saves end users from having to create another account for your web application, and also from having to remember another username and password.

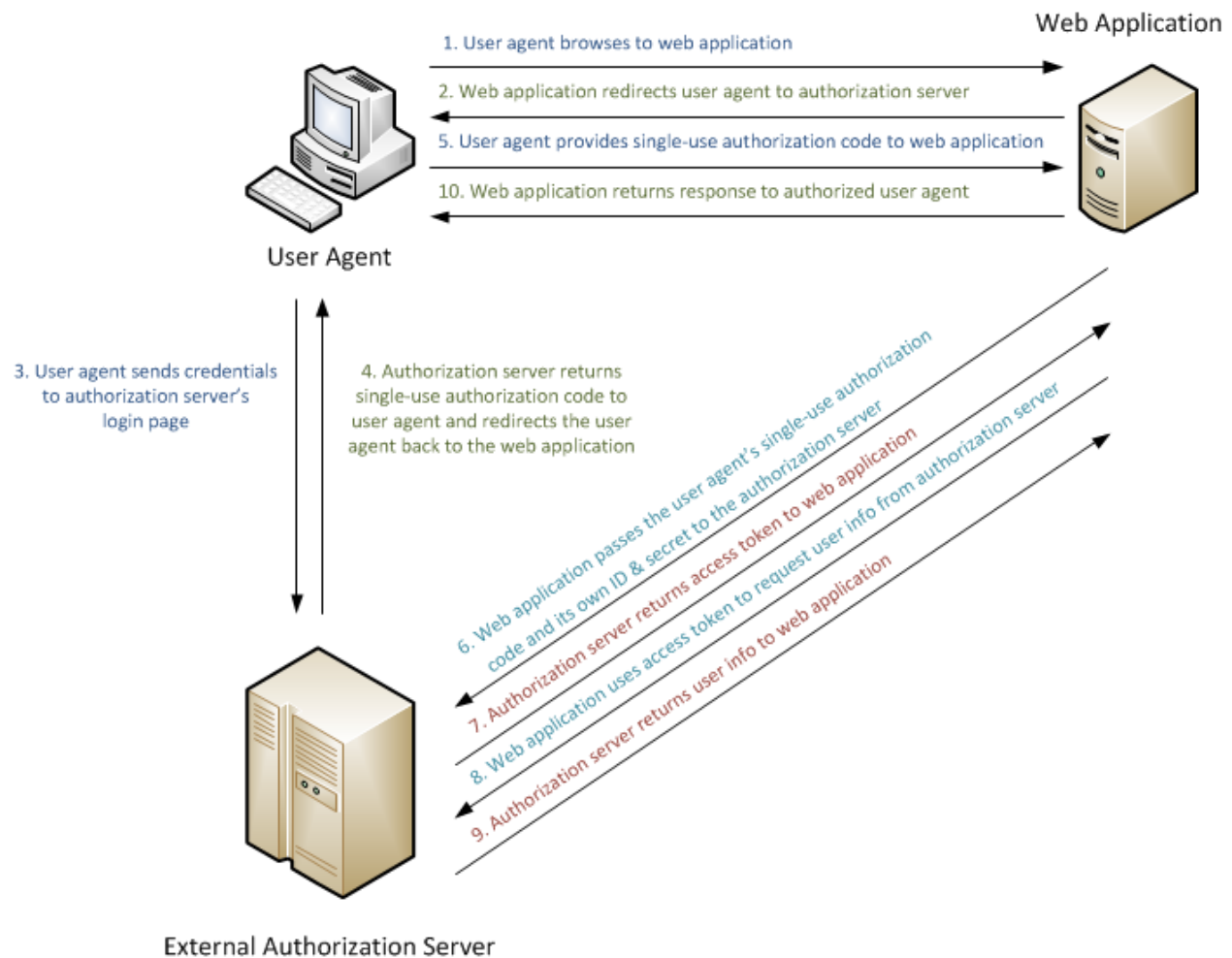
In the past, developers have had two choices: create their own authentication implementation, or learn how to integrate an external authentication service into their applications. At the most basic level, the following diagram illustrates a simple request flow for a user agent (web browser) that is requesting information from a web application that is configured to use an external authentication service:



In the preceding diagram, the user agent (or web browser in this example) makes a request to a web application, which redirects the web browser to an external authentication service. The user agent sends its credentials to the external authentication service, and if the user agent has successfully authenticated, the external authentication service will redirect the user agent to the original web application with some form of token which the user agent will send to the web application. The web application will use the token to verify that the user agent has been successfully authenticated by the external authentication service, and the web application may use the token to gather more information about the user agent. Once the application is done processing the user agent's information, the web application will return the appropriate response to the user agent based on its authorization settings.

In this second example, the user agent negotiates with the web application and external authorization server, and the web application performs additional communication with the external authorization server to retrieve additional information about the user agent:

OAuth2 Authorization Code Grant



Visual Studio 2017 and ASP.NET 4.7.2 make integration with external authentication services easier for developers by providing built-in integration for the following authentication services:

- Facebook
- Google
- Microsoft Accounts (Windows Live ID accounts)
- Twitter

The examples in this walkthrough will demonstrate how to configure each of the supported external authentication services by using the new ASP.NET Web Application template that ships with Visual Studio 2017.

ⓘ Note

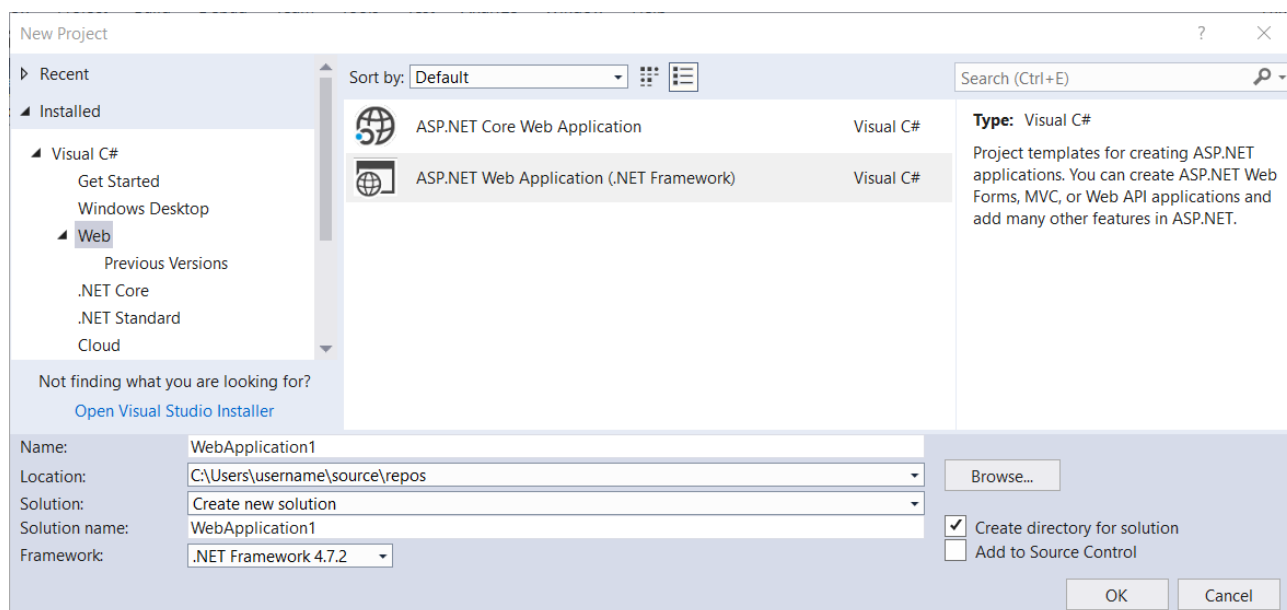
If necessary, you may need to add your FQDN to the settings for your external authentication service. This requirement is based on security constraints for some external authentication services which require the FQDN in your application settings to match the FQDN that is used by your clients. (The steps for this will vary greatly for each external authentication service; you will need to consult the documentation for each external authentication service to see if this is required and how to configure these settings.) If you need to configure IIS Express to use an FQDN for testing this environment, see the [Configuring IIS Express to use a Fully Qualified Domain Name](#) section later in this walkthrough.

Create a Sample Web Application

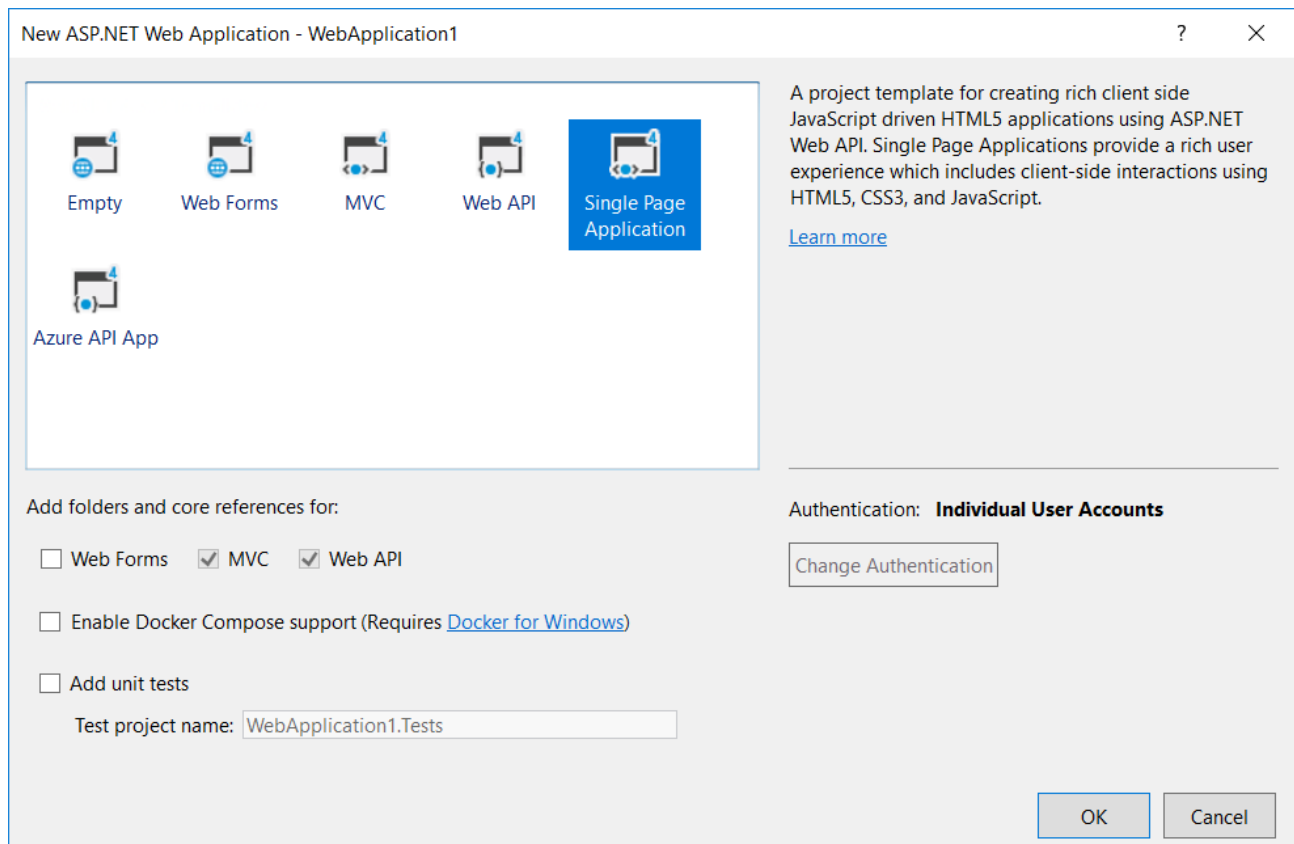
The following steps will lead you through creating a sample application by using the ASP.NET Web Application template, and you will use this sample application for each of the external authentication services later in this walkthrough.

Start Visual Studio 2017 and select **New Project** from the Start page. Or, from the **File** menu, select **New** and then **Project**.

When the **New Project** dialog box is displayed, select **Installed** and expand **Visual C#**. Under **Visual C#**, select **Web**. In the list of project templates, select **ASP.NET Web Application (.NET Framework)**. Enter a name for your project and click **OK**.



When the **New ASP.NET Project** is displayed, select the **Single Page Application** template and click **Create Project**.



Wait as Visual Studio 2017 creates your project.

When Visual Studio 2017 has finished creating your project, open the *Startup.Auth.cs* file that is located in the **App_Start** folder.

When you first create the project, none of the external authentication services are enabled in *Startup.Auth.cs* file; the following illustrates what your code might resemble, with the sections highlighted for where you would enable an external authentication service and any relevant settings in order to use Microsoft Accounts, Twitter, Facebook, or Google authentication with your ASP.NET application:

C#	Copy
<pre>using System; using Microsoft.AspNet.Identity; using Microsoft.AspNet.Identity.EntityFramework; using Microsoft.AspNet.Identity.Owin; using Microsoft.Owin; using Microsoft.Owin.Security.Cookies; using Microsoft.Owin.Security.DataProtection; using Microsoft.Owin.Security.Google; using Microsoft.Owin.Security.OAuth; using Owin; using WebApplication1.Models; using WebApplication1.Providers;</pre>	

```
namespace WebApplication1
{
    public partial class Startup
    {
        // Enable the application to use OAuthAuthorization. You can then
        secure your Web APIs
        static Startup()
        {
            PublicClientId = "web";

            OAuthOptions = new OAuthAuthorizationServerOptions
            {
                TokenEndpointPath = new PathString("/Token"),
                AuthorizeEndpointPath = new PathString("/Account/Authorize"),
                Provider = new ApplicationOAuthProvider(PublicClientId),
                AccessTokenExpireTimeSpan = TimeSpan.FromDays(14),
                AllowInsecureHttp = true
            };

            public static OAuthAuthorizationServerOptions OAuthOptions { get;
private set; }

            public static string PublicClientId { get; private set; }

            // For more information on configuring authentication, please visit
            https://go.microsoft.com/fwlink/?LinkId=301864
            public void ConfigureAuth(IAppBuilder app)
            {
                // Configure the db context, user manager and signin manager to use
                a single instance per request
                app.CreatePerOwinContext(ApplicationDbContext.Create);
                app.CreatePerOwinContext<ApplicationUserManager>
                (ApplicationUserManager.Create);
                app.CreatePerOwinContext<ApplicationSignInManager>
                (ApplicationSignInManager.Create);

                // Enable the application to use a cookie to store information for
                the signed in user
                app.UseCookieAuthentication(new CookieAuthenticationOptions
                {
                    AuthenticationType =
                    DefaultAuthenticationTypes.ApplicationCookie,
                    LoginPath = new PathString("/Account/Login"),
                    Provider = new CookieAuthenticationProvider
                    {
                        // Enables the application to validate the security stamp
                        when the user logs in.
                        // This is a security feature which is used when you change
                        a password or add an external login to your account.
                    }
                })
            }
        }
    }
}
```

```
OnValidateIdentity =
SecurityStampValidator.OnValidateIdentity<ApplicationUserManager,
ApplicationUser>(
    validateInterval: TimeSpan.FromMinutes(20),
    regenerateIdentity: (manager, user) =>
user.GenerateUserIdentityAsync(manager))
    }
});
// Use a cookie to temporarily store information about a user
logging in with a third party login provider

app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

// Enables the application to temporarily store user information
when they are verifying the second factor in the two-factor authentication
process.

app.UseTwoFactorSignInCookie(DefaultAuthenticationTypes.TwoFactorCookie,
TimeSpan.FromMinutes(5));

// Enables the application to remember the second login
verification factor such as phone or email.
// Once you check this option, your second step of verification
during the login process will be remembered on the device where you logged in
from.
// This is similar to the RememberMe option when you log in.

app.UseTwoFactorRememberBrowserCookie(DefaultAuthenticationTypes.TwoFactorRemem
berBrowserCookie);

// Enable the application to use bearer tokens to authenticate
users
app.UseOAuthBearerTokens(OAuthOptions);

// Uncomment the following lines to enable logging in with third
party login providers
//app.UseMicrosoftAccountAuthentication(
//    clientId: "",
//    clientSecret: "");

//app.UseTwitterAuthentication(
//    consumerKey: "",
//    consumerSecret: "");

//app.UseFacebookAuthentication(
//    appId: "",
//    appSecret: "");

//app.UseGoogleAuthentication(new
GoogleOAuth2AuthenticationOptions()
//{
```



```
// ClientId = "",  
// ClientSecret = ""  
//});  
}  
}  
}
```

When you press F5 to build and debug your web application, it will display a login screen where you will see that no external authentication services have been defined.

Application name

Home

API

Register

Log in

Log in.

Use a local account to log in.

Use another service to log in.

Email

Password

☐ Remember me?

Log in

[Register as a new user](#)

There are no external authentication services configured. See [this article](#) for details on setting up this ASP.NET application to support logging in via external services.

© 2019 - My ASP.NET Application

In the following sections, you will learn how to enable each of the external authentication services that are provided with ASP.NET in Visual Studio 2017.

Enabling Facebook authentication


Using Facebook authentication requires you to create a Facebook developer account, and your project will require an application ID and secret key from Facebook in order to function. For information about creating a Facebook developer account and obtaining your application ID and secret key, see <https://go.microsoft.com/fwlink/?LinkID=252166>.

Once you have obtained your application ID and secret key, use the following steps to enable Facebook authentication for your web application:

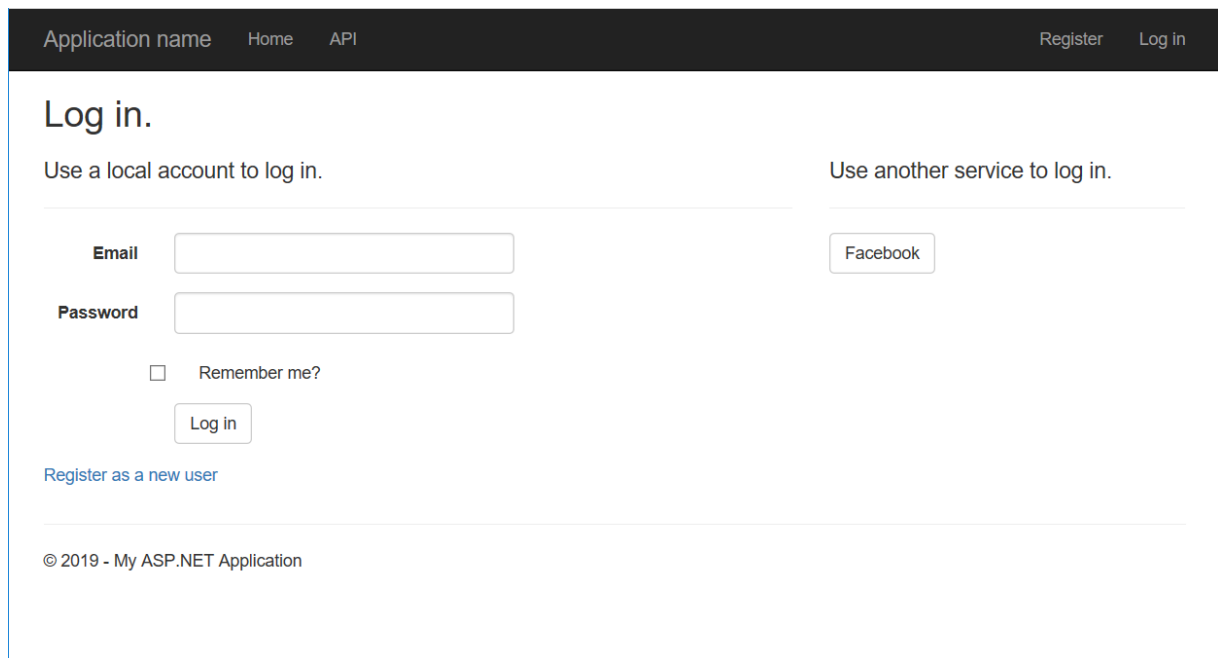
1. When your project is open in Visual Studio 2017, open the *Startup.Auth.cs* file.
2. Locate the Facebook authentication section of code:

C#	
<pre>// Uncomment the following lines to enable logging in with third party login providers //app.UseMicrosoftAccountAuthentication(// clientId: "", // clientSecret: ""); //app.UseTwitterAuthentication(// consumerKey: "", // consumerSecret: ""); //app.UseFacebookAuthentication(// appId: "", // appSecret: ""); //app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions() //{ // ClientId = "", // ClientSecret = "" //});</pre>	

3. Remove the "//" characters to uncomment the highlighted lines of code, and then add your application ID and secret key. Once you have added those parameters, you can recompile your project:

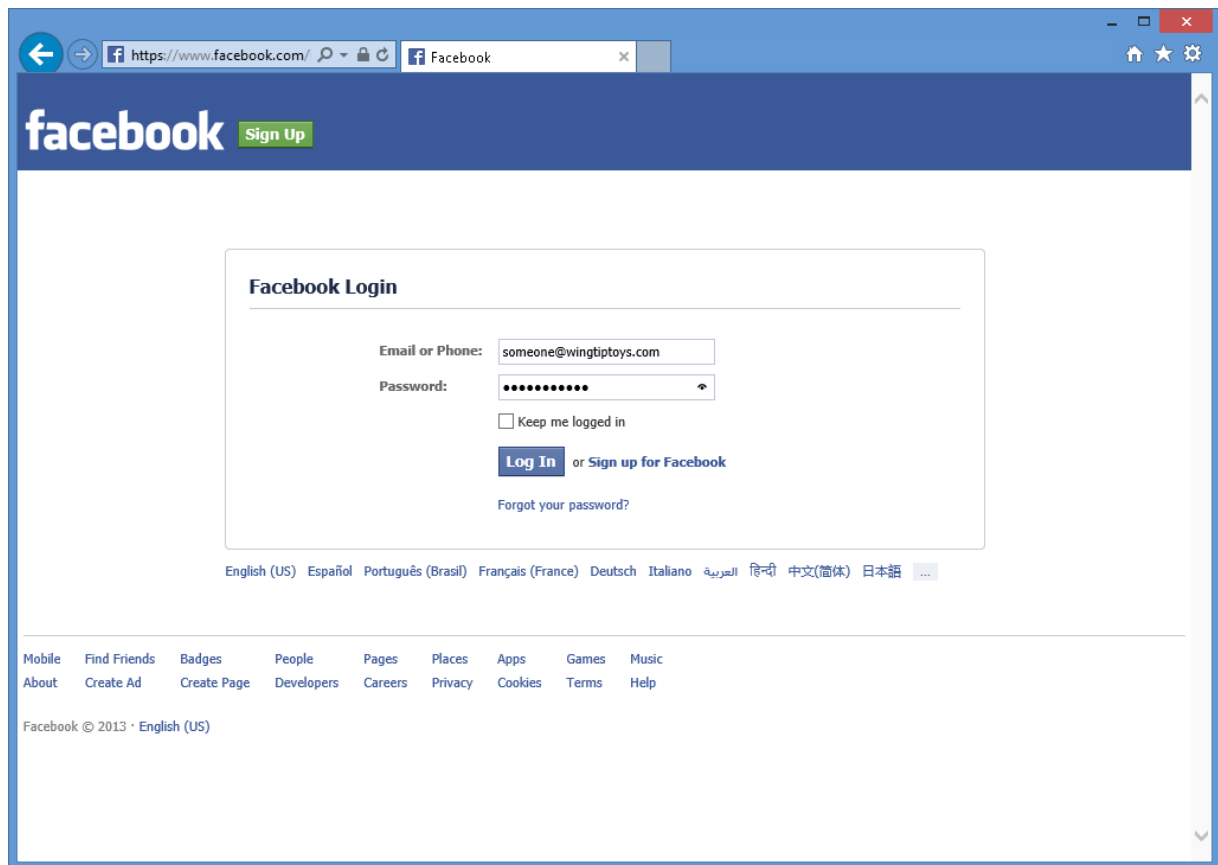
C#	
<pre>// Uncomment the following lines to enable logging in with third party login providers //app.UseMicrosoftAccountAuthentication(// clientId: "", // clientSecret: ""); //app.UseTwitterAuthentication(// consumerKey: "", // consumerSecret: ""); app.UseFacebookAuthentication(appId: "426f62526f636b73", appSecret: "57686f6120447564652c2049495320526f636b73"); //app.UseGoogleAuthentication();</pre>	

4. When you press F5 to open your web application in your web browser, you will see that Facebook has been defined as an external authentication service:



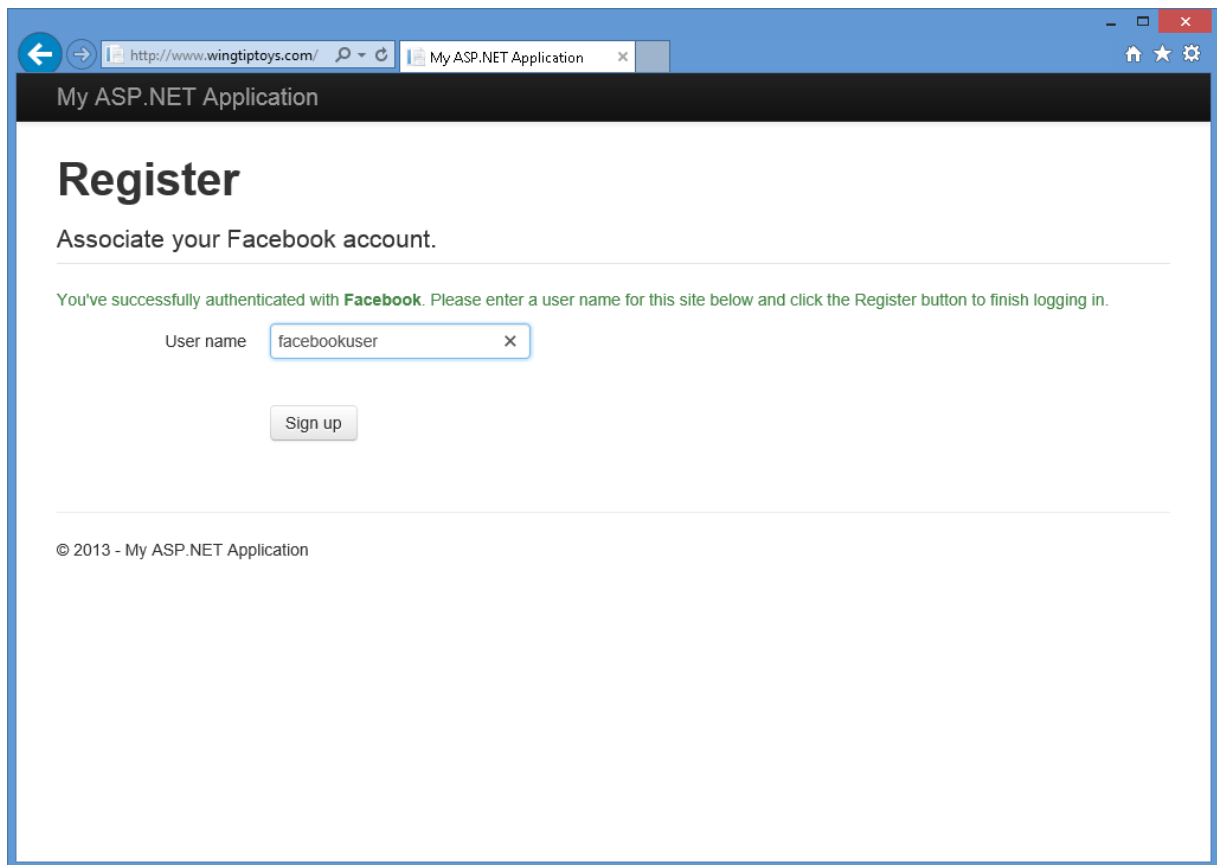
The screenshot shows a web application interface with a dark header bar containing links for 'Application name', 'Home', 'API', 'Register', and 'Log in'. The main content area is titled 'Log in.' and is divided into two sections. The left section, 'Use a local account to log in.', contains input fields for 'Email' and 'Password', a 'Remember me?' checkbox, and a 'Log in' button. Below this is a link to 'Register as a new user'. The right section, 'Use another service to log in.', features a 'Facebook' button. At the bottom, a copyright notice reads '© 2019 - My ASP.NET Application'.

5. When you click the **Facebook** button, your browser will be redirected to the Facebook login page:

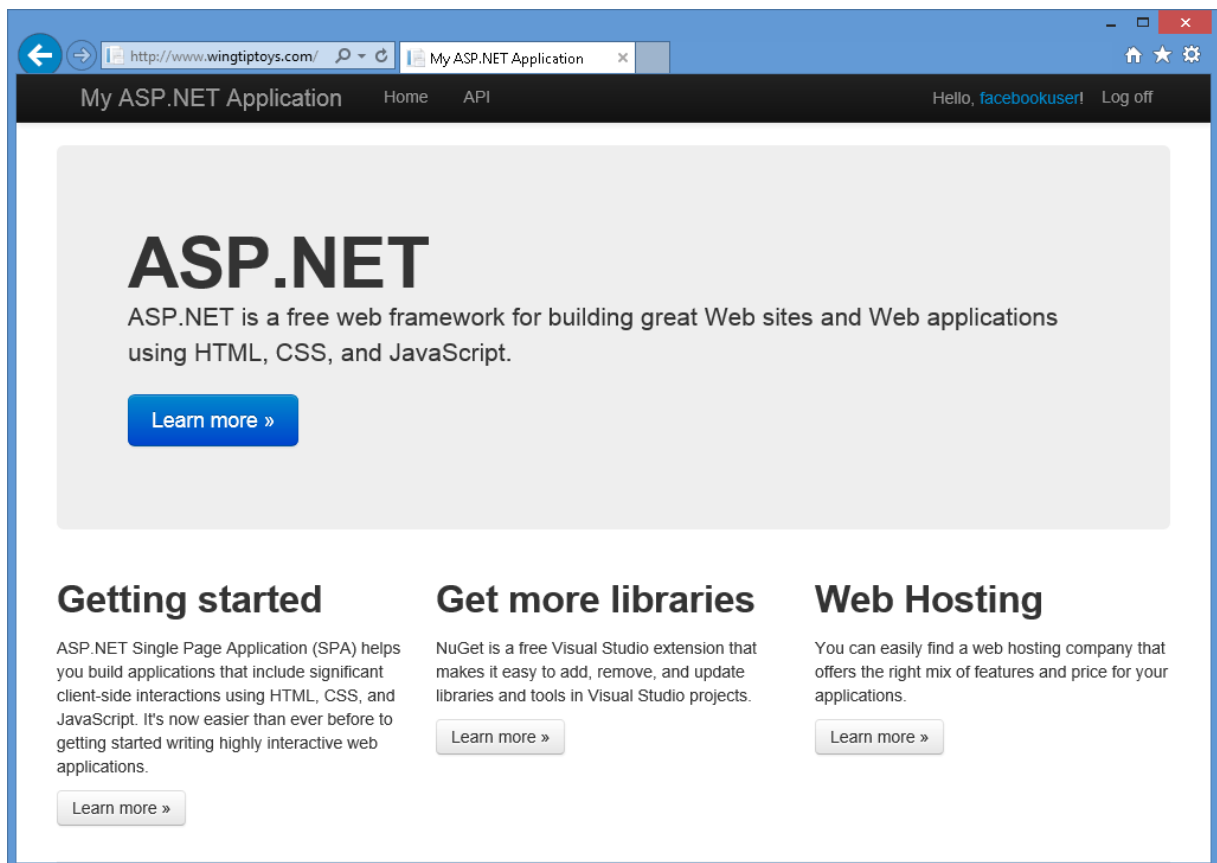


The screenshot displays the Facebook login page in a web browser. The browser's address bar shows 'https://www.facebook.com/'. The page features the Facebook logo and a 'Sign Up' button. The main content area is titled 'Facebook Login' and contains input fields for 'Email or Phone:' (with the value 'someone@wingtiptoy.com') and 'Password:'. Below these fields are a 'Keep me logged in' checkbox and a 'Log In' button, followed by a link to 'Sign up for Facebook'. A link for 'Forgot your password?' is also present. At the bottom, there is a row of language links including 'English (US)', 'Español', 'Português (Brasil)', 'Français (France)', 'Deutsch', 'Italiano', 'العربية', 'हिन्दी', '中文(简体)', '日本語', and a dropdown menu. The footer includes links for 'Mobile', 'Find Friends', 'Badges', 'People', 'Pages', 'Places', 'Apps', 'Games', 'Music', 'About', 'Create Ad', 'Create Page', 'Developers', 'Careers', 'Privacy', 'Cookies', 'Terms', and 'Help', along with the copyright notice 'Facebook © 2013 · English (US)'.

6. After you enter your Facebook credentials and click **Log in**, your web browser will be redirected back to your web application, which will prompt you for the **User name** that you want to associate with your Facebook account:



7. After you have entered your user name and clicked the **Sign up** button, your web application will display the default **home page** for your Facebook account:



Enabling Google Authentication


Using Google authentication requires you to create a Google developer account, and your project will require an application ID and secret key from Google in order to function. For information about creating a Google developer account and obtaining your application ID and secret key, see <https://developers.google.com>.

To enable Google authentication for your web application, use the following steps:

1. When your project is open in Visual Studio 2017, open the *Startup.Auth.cs* file.
2. Locate the Google authentication section of code:

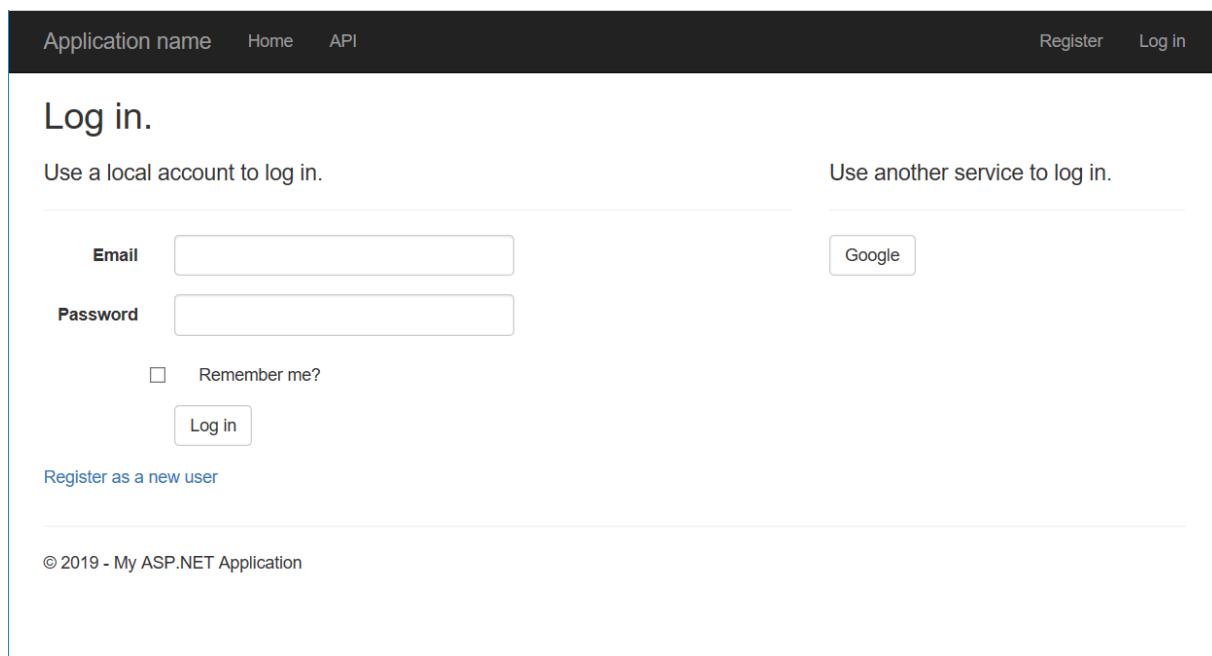
C#	 Copy
<pre>// Uncomment the following lines to enable logging in with third party login providers //app.UseMicrosoftAccountAuthentication(// clientId: "", // clientSecret: ""); //app.UseTwitterAuthentication(// consumerKey: "", // consumerSecret: ""); //app.UseFacebookAuthentication(// appId: "", // appSecret: ""); //app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions() //{ // ClientId = "", // ClientSecret = "" //});</pre>	

3. Remove the `//` characters to uncomment the highlighted lines of code, and then add your application ID and secret key. Once you have added those parameters, you can recompile your project:

C#	 Copy
<pre>// Uncomment the following lines to enable logging in with third party login providers //app.UseMicrosoftAccountAuthentication(// clientId: "", // clientSecret: "");</pre>	

```
//app.UseTwitterAuthentication(  
//  consumerKey: "",  
//  consumerSecret: "");  
  
//app.UseFacebookAuthentication(  
//  appId: "",  
//  appSecret: "");  
  
app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()  
{  
    ClientId = "477522346600.apps.googleusercontent.com",  
    ClientSecret = "gobkdpbocikdfbnfahjladnetpdkvmic"  
});
```

4. When you press F5 to open your web application in your web browser, you will see that Google has been defined as an external authentication service:



Application name Home API Register Log in

Log in.

Use a local account to log in. Use another service to log in.

Email

Password

☐ Remember me?

Log in

[Register as a new user](#)

© 2019 - My ASP.NET Application

5. When you click the **Google** button, your browser will be redirected to the Google login page:

Accounts

Localhost is asking for some information from your Google Account. To see and approve the request, sign in. [Learn more](#)

Sign in

Email

someone@wingtiptoy.com

Password

••••••••

Sign in ☒ Stay signed in

[Can't access your account?](#)

© 2013 Google [Terms of Service](#) [Privacy Policy](#) [Help](#) English (United States)

6. After you enter your Google credentials and click **Sign in**, Google will prompt you to verify that your web application has permissions to access your Google account:

Google

someone

Localhost

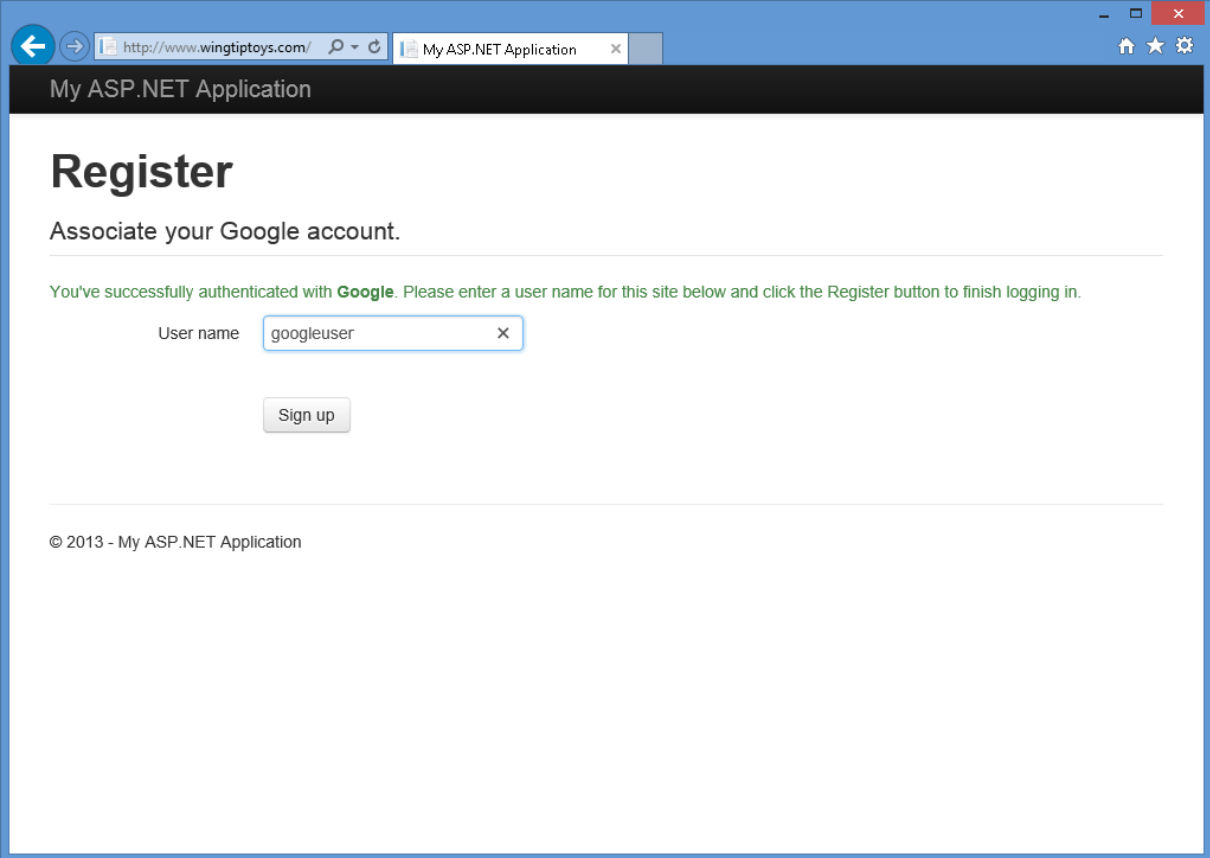
Localhost would like to:

- View basic information about your account
- View your email address

Localhost and Google will use this information in accordance with their respective terms of service and privacy policies.

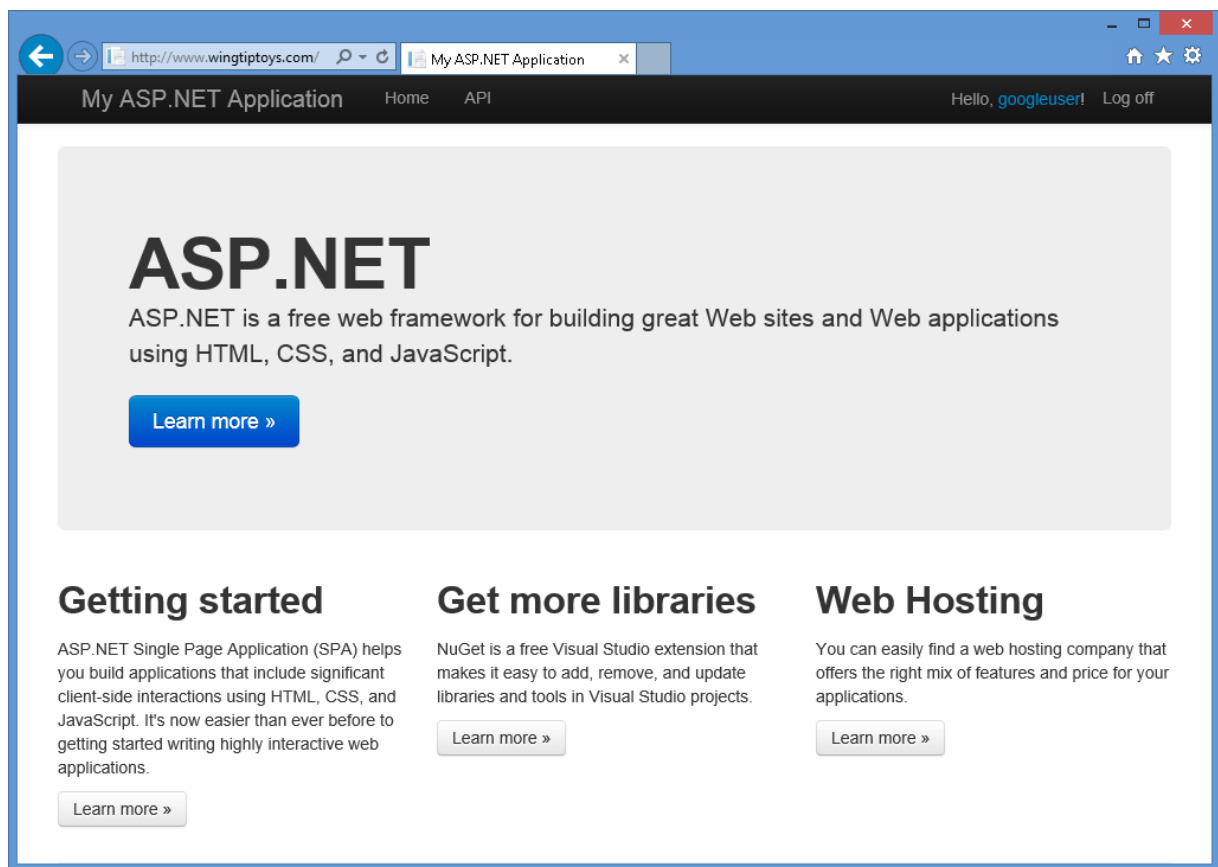
Cancel Accept

7. When you click **Accept**, your web browser will be redirected back to your web application, which will prompt you for the **User name** that you want to associate with your Google account:



The screenshot shows a web browser window with the address bar displaying `http://www.wingtiptoy.com/` and a tab titled 'My ASP.NET Application'. The page has a dark header with the text 'My ASP.NET Application'. Below the header, the main content area has a title 'Register' and a subtitle 'Associate your Google account.'. A green message states: 'You've successfully authenticated with Google. Please enter a user name for this site below and click the Register button to finish logging in.' Below this message is a text input field labeled 'User name' containing the text 'googleuser'. To the right of the input field is a small 'x' icon. Below the input field is a 'Sign up' button. At the bottom of the page, there is a copyright notice: '© 2013 - My ASP.NET Application'.

8. After you have entered your user name and clicked the **Sign up** button, your web application will display the default **home page** for your Google account:



Enabling Microsoft Authentication

Microsoft authentication requires you to create a developer account, and it requires a client ID and client secret in order to function. For information about creating a Microsoft developer account and obtaining your client ID and client secret, see <https://go.microsoft.com/fwlink/?LinkID=144070>.

Once you have obtained your consumer key and consumer secret, use the following steps to enable Microsoft authentication for your web application:

1. When your project is open in Visual Studio 2017, open the *Startup.Auth.cs* file.
2. Locate the Microsoft authentication section of code:

C#	Copy
<pre>// Uncomment the following lines to enable logging in with third party login providers //app.UseMicrosoftAccountAuthentication(// clientId: "", // clientSecret: ""); //app.UseTwitterAuthentication(</pre>	

```
// consumerKey: "",
// consumerSecret: "");

//app.UseFacebookAuthentication(
//  appId: "",
//  appSecret: "");

//app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()
//{
//  ClientId = "",
//  ClientSecret = ""
//});
```

3. Remove the "//" characters to uncomment the highlighted lines of code, and then add your client ID and client secret. Once you have added those parameters, you can recompile your project:

C#

 Copy

```
// Uncomment the following lines to enable logging in with third party
login providers
app.UseMicrosoftAccountAuthentication(
  clientId: "426f62526f636b73",
  clientSecret: "57686f6120447564652c2049495320526f636b73");

//app.UseTwitterAuthentication(
//  consumerKey: "",
//  consumerSecret: "");

//app.UseFacebookAuthentication(
//  appId: "",
//  appSecret: "");

//app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()
//{
//  ClientId = "",
//  ClientSecret = ""
//});
```

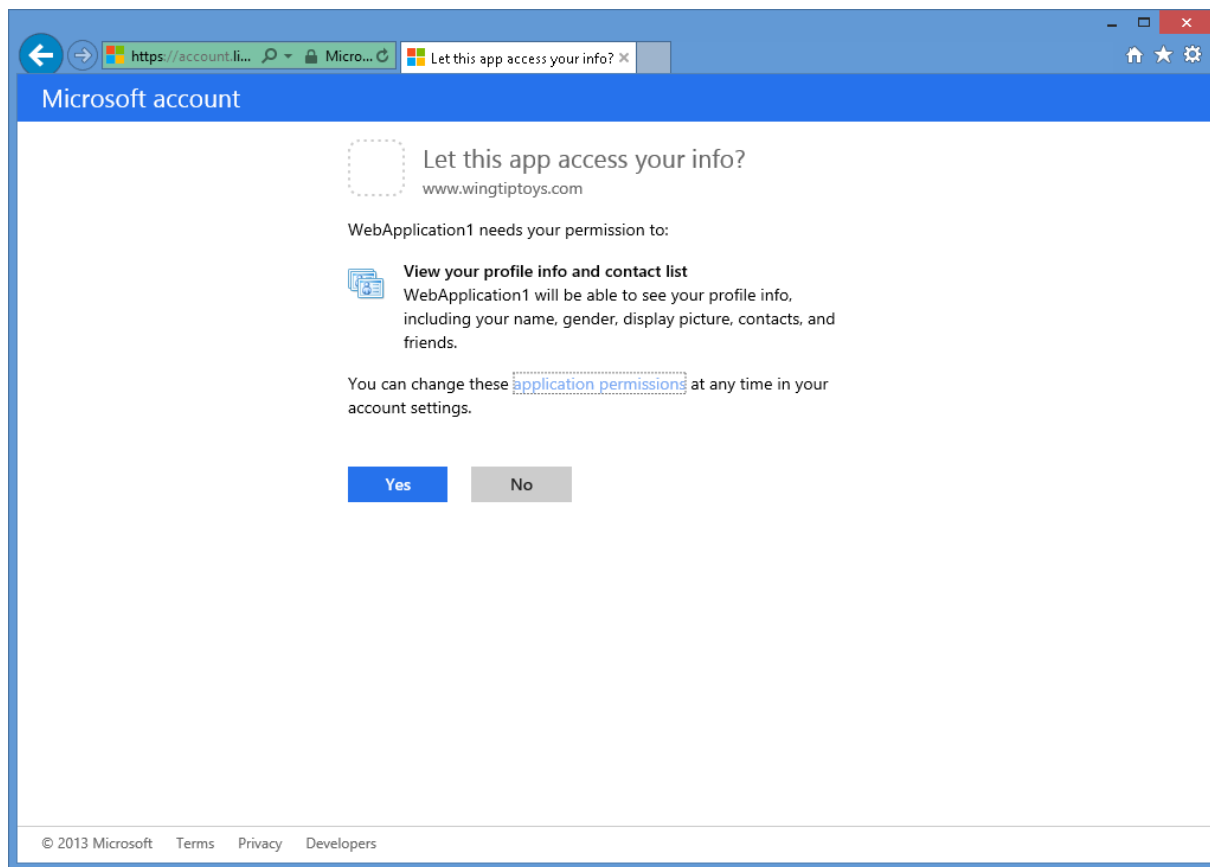
4. When you press F5 to open your web application in your web browser, you will see that Microsoft has been defined as an external authentication service:

The screenshot shows a web browser window with the address bar displaying `http://www.wingtip toys.com/` and a tab titled 'My ASP.NET Application'. The page has a dark header with the text 'My ASP.NET Application'. Below the header, the main content area is titled 'Log in'. It is divided into two sections. The left section, 'Use a local account to log in.', contains a 'User name' input field, a 'Password' input field, a 'Remember me?' checkbox, and a 'Log in' button. Below this section is a link: 'Register if you don't have a local account.' The right section, 'Log in using another service', contains the text 'Use another service to log in.' and a 'Microsoft' button. At the bottom of the page, there is a copyright notice: '© 2013 - My ASP.NET Application'.

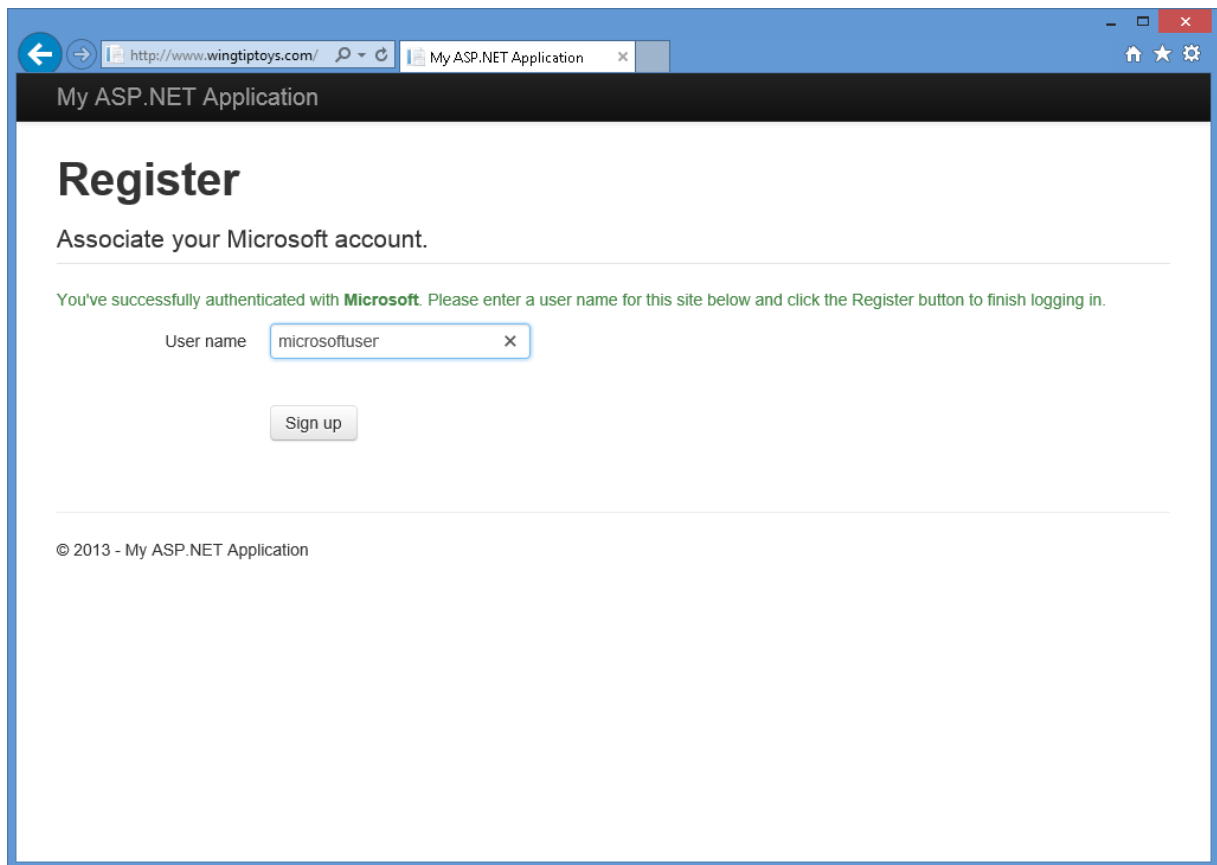
5. When you click the **Microsoft** button, your browser will be redirected to the Microsoft login page:

The screenshot shows a web browser window with the address bar displaying `https://login.live....` and a tab titled 'Sign in to your Microsoft ac...'. The page has a white background with the title 'Sign in'. Below the title, there is a message: 'Because you're accessing sensitive info, you need to verify your password.' Below this message, there is a text input field containing the email address 'someone@wingtip toys.com'. Below the email field is a password input field represented by a series of dots. Below the password field is a blue 'Sign in' button. Below the button, there are two links: 'Can't access your account?' and 'Sign in with a different Microsoft account'. At the bottom of the page, there is a copyright notice: '©2013 Microsoft' and a link to 'Privacy & Cookies'.

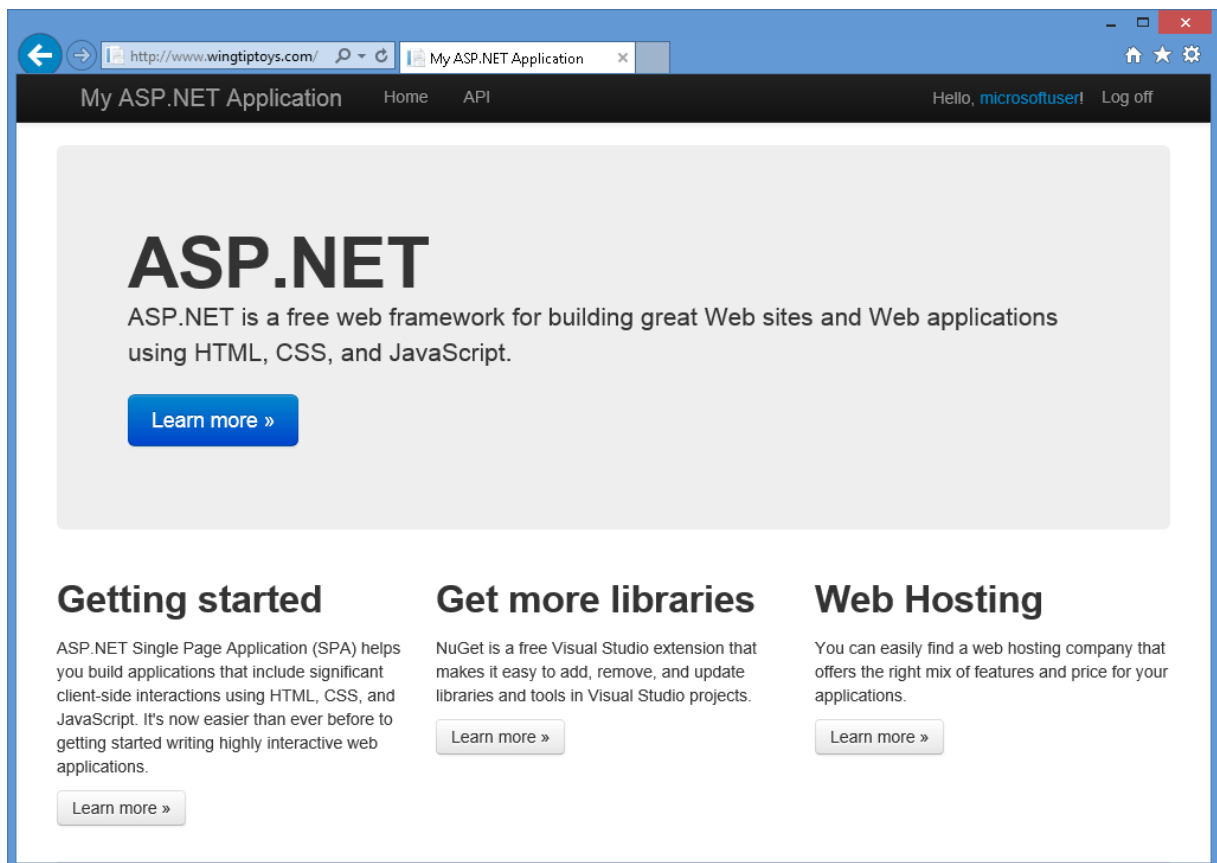
6. After you enter your Microsoft credentials and click **Sign in**, you will be prompted to verify that your web application has permissions to access your Microsoft account:



7. When you click **Yes**, your web browser will be redirected back to your web application, which will prompt you for the **User name** that you want to associate with your Microsoft account:



8. After you have entered your user name and clicked the **Sign up** button, your web application will display the default **home page** for your Microsoft account:




Enabling Twitter Authentication


Twitter authentication requires you to create a developer account, and it requires a consumer key and consumer secret in order to function. For information about creating a Twitter developer account and obtaining your consumer key and consumer secret, see <https://go.microsoft.com/fwlink/?LinkID=252166>.

Once you have obtained your consumer key and consumer secret, use the following steps to enable Twitter authentication for your web application:

1. When your project is open in Visual Studio 2017, open the *Startup.Auth.cs* file.
2. Locate the Twitter authentication section of code:

C#	 Copy
<pre>// Uncomment the following lines to enable logging in with third party login providers //app.UseMicrosoftAccountAuthentication(// clientId: "", // clientSecret: ""); //app.UseTwitterAuthentication(// consumerKey: "", // consumerSecret: ""); //app.UseFacebookAuthentication(// appId: "", // appSecret: ""); //app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions() //{ // ClientId = "", // ClientSecret = "" //});</pre>	

3. Remove the `//` characters to uncomment the highlighted lines of code, and then add your consumer key and consumer secret. Once you have added those parameters, you can recompile your project:

C#	 Copy
<pre>// Uncomment the following lines to enable logging in with third party login providers //app.UseMicrosoftAccountAuthentication(// clientId: "",</pre>	

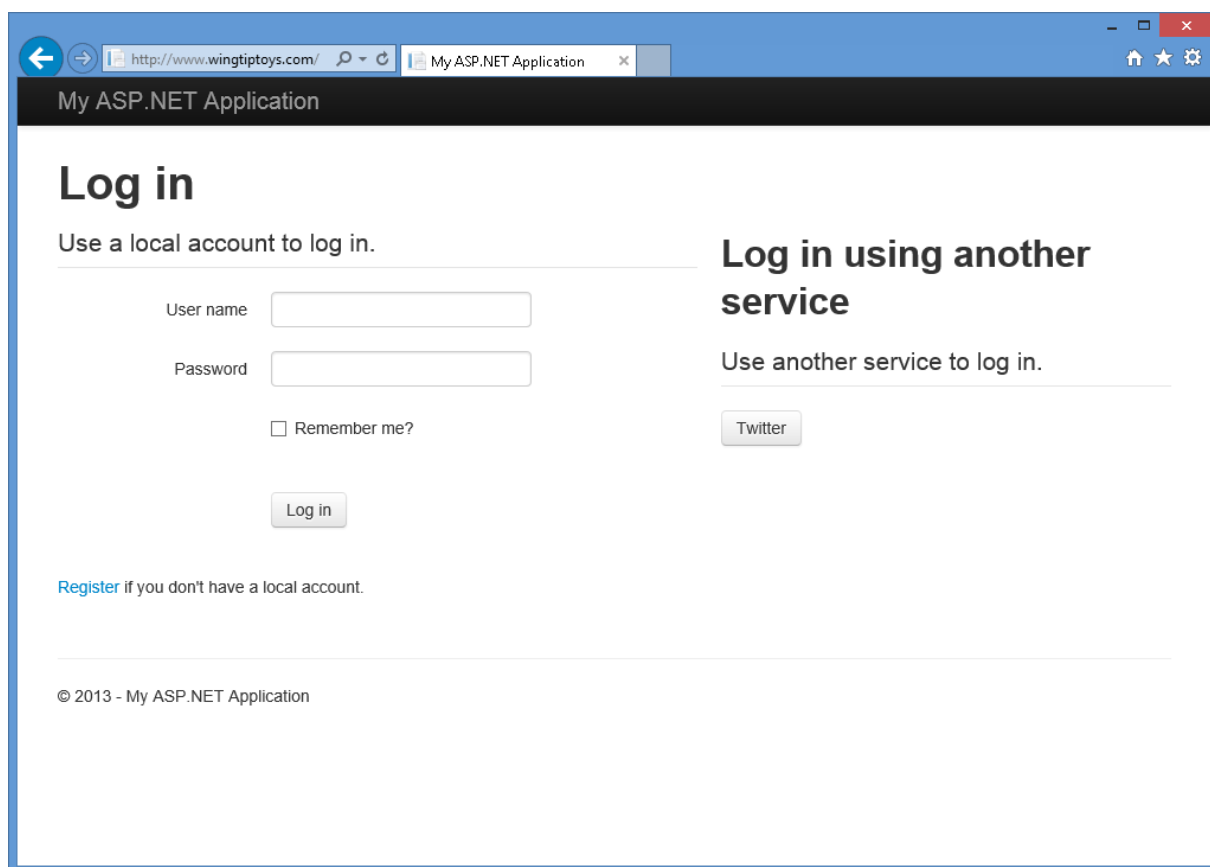
```
//    clientSecret: "");

app.UseTwitterAuthentication(
    consumerKey: "426f62526f636b73",
    consumerSecret: "57686f6120447564652c2049495320526f636b73");

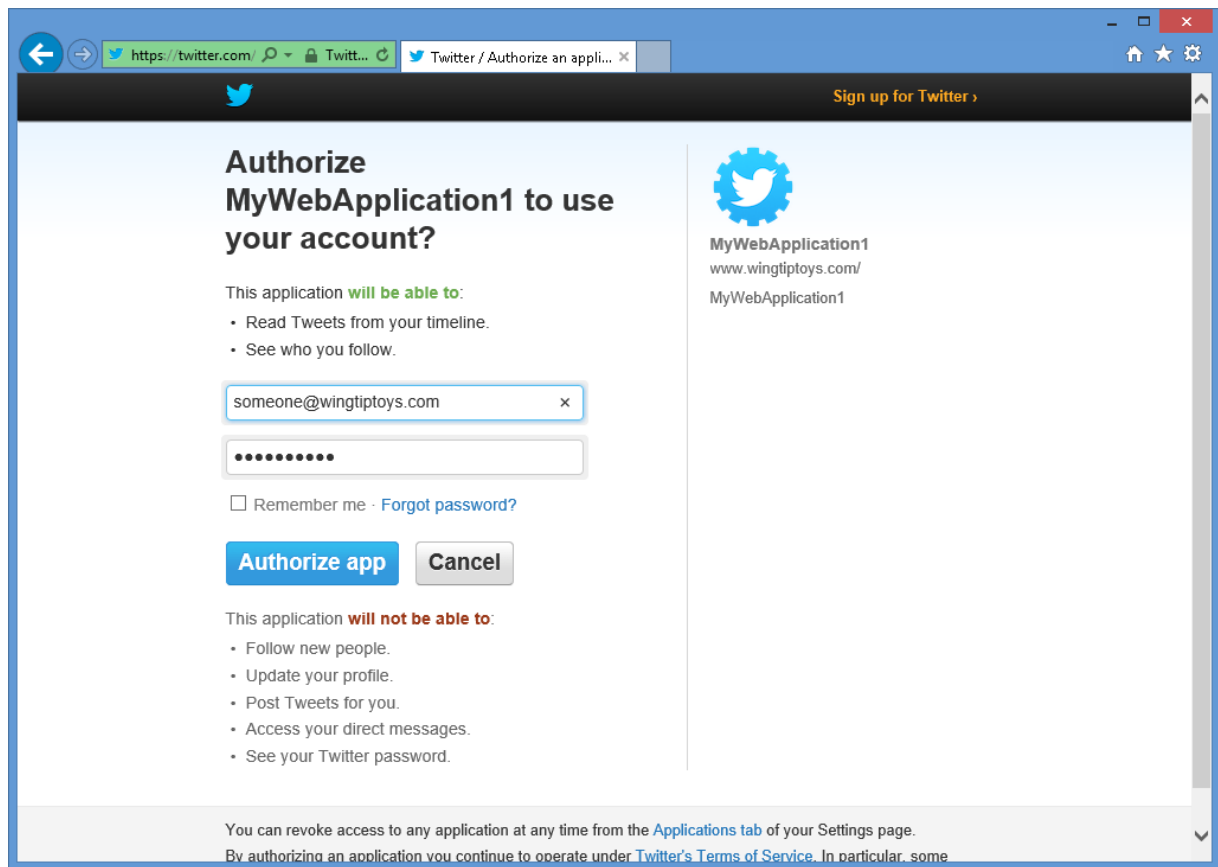
//app.UseFacebookAuthentication(
//    appId: "",
//    appSecret: "");

//app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()
//{
//    ClientId = "",
//    ClientSecret = ""
//});
```

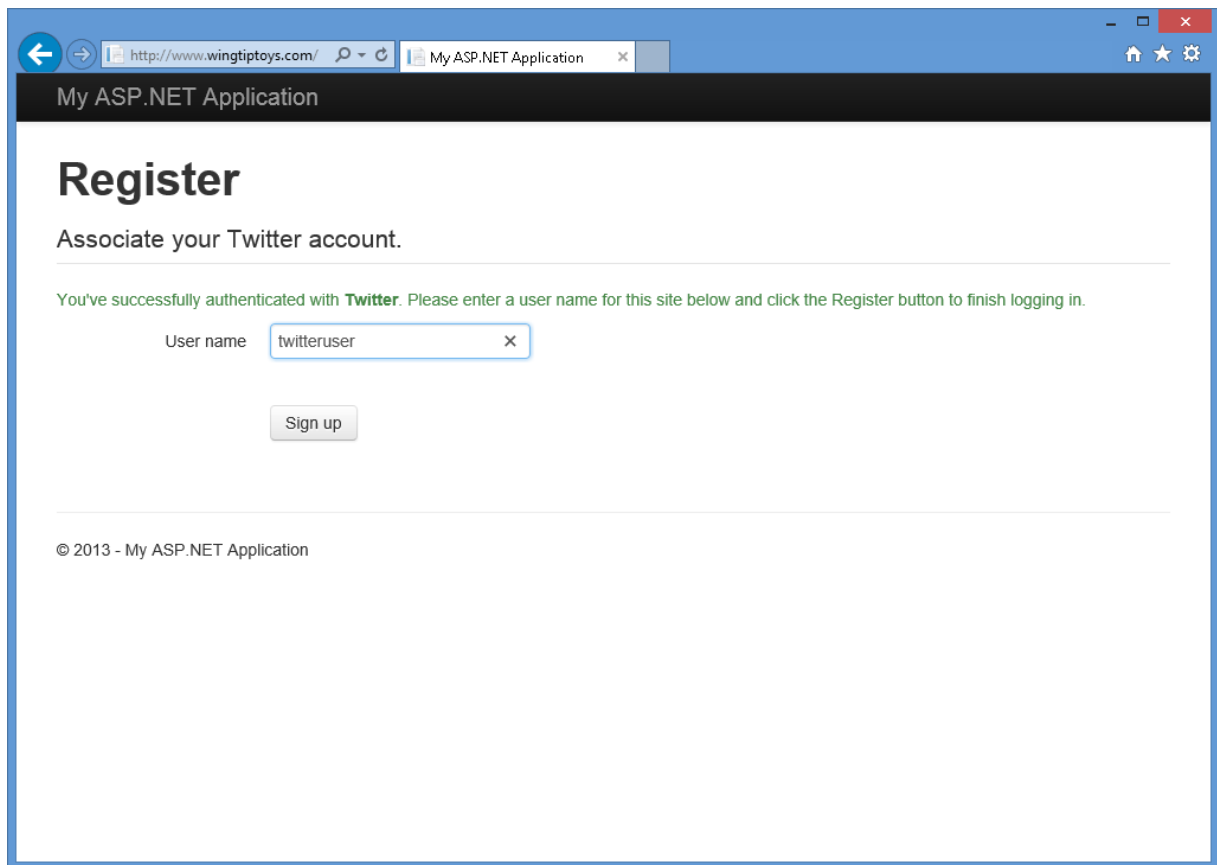
4. When you press F5 to open your web application in your web browser, you will see that Twitter has been defined as an external authentication service:



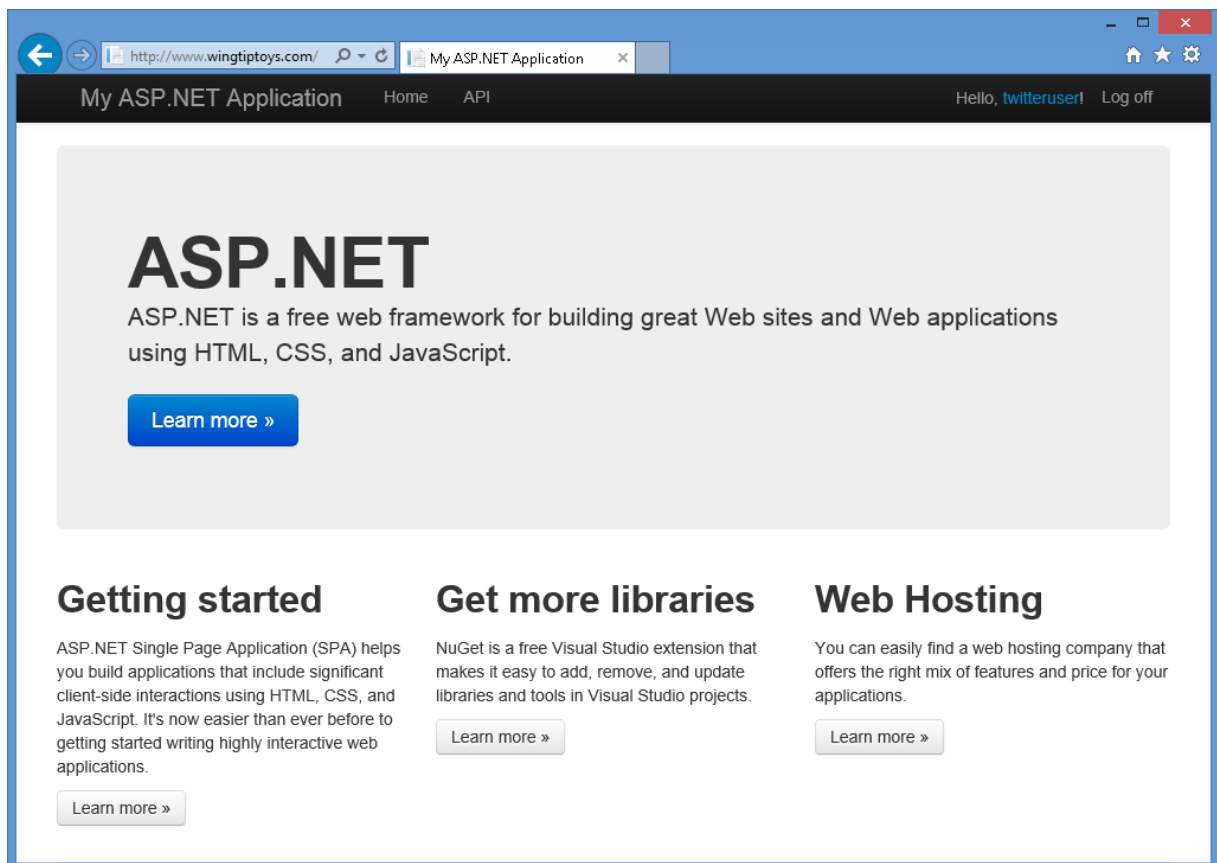
5. When you click the **Twitter** button, your browser will be redirected to the Twitter login page:



6. After you enter your Twitter credentials and click **Authorize app**, your web browser will be redirected back to your web application, which will prompt you for the **User name** that you want to associate with your Twitter account:



7. After you have entered your user name and clicked the **Sign up** button, your web application will display the default **home page** for your Twitter account:



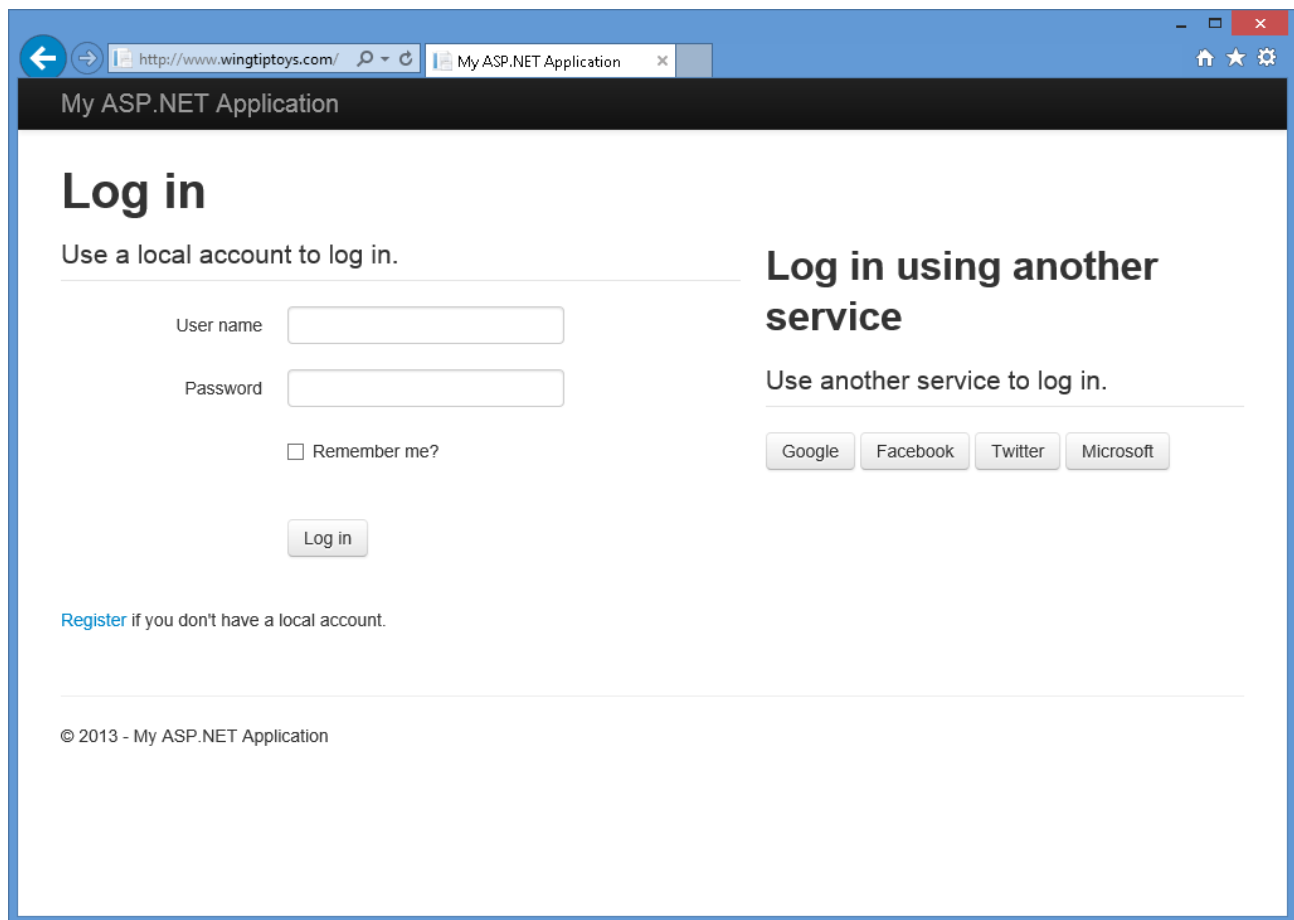
Additional Information

For additional information about creating applications that use OAuth and OpenID, see the following URLs:

- <https://go.microsoft.com/fwlink/?LinkID=252166>
- <https://go.microsoft.com/fwlink/?LinkID=243995>

Combining External Authentication Services

For greater flexibility, you can define multiple external authentication services at the same time - this allows your web application's users to use an account from any of the enabled external authentication services:



Configure IIS Express to use a fully qualified domain name

Some external authentication providers do not support testing your application by using an HTTP address like `http://localhost:port/`. To work around this issue, you can add a static Fully Qualified Domain Name (FQDN) mapping to your HOSTS file and configure

your project options in Visual Studio 2017 to use the FQDN for testing/debugging. To do so, use the following steps:

- Add a static FQDN mapping your HOSTS file:
 1. Open an elevated command prompt in Windows.
 2. Type the following command:

```
notepad %WinDir%\system32\drivers\etc\hosts
```
 3. Add an entry like the following to the HOSTS file:

```
127.0.0.1 www.wingtiptoy.com
```
 4. Save and close your HOSTS file.
 - Configure your Visual Studio project to use the FQDN:
 1. When your project is open in Visual Studio 2017, click the **Project** menu, and then select your project's properties. For example, you might select **WebApplication1 Properties**.
 2. Select the **Web** tab.
 3. Enter your FQDN for the **Project Url**. For example, you would enter

```
http://www.wingtiptoy.com
```

 if that was the FQDN mapping that you added to your HOSTS file.
 - Configure IIS Express to use the FQDN for your application:
 1. Open an elevated command prompt in Windows.
 2. Type the following command to change to your IIS Express folder:

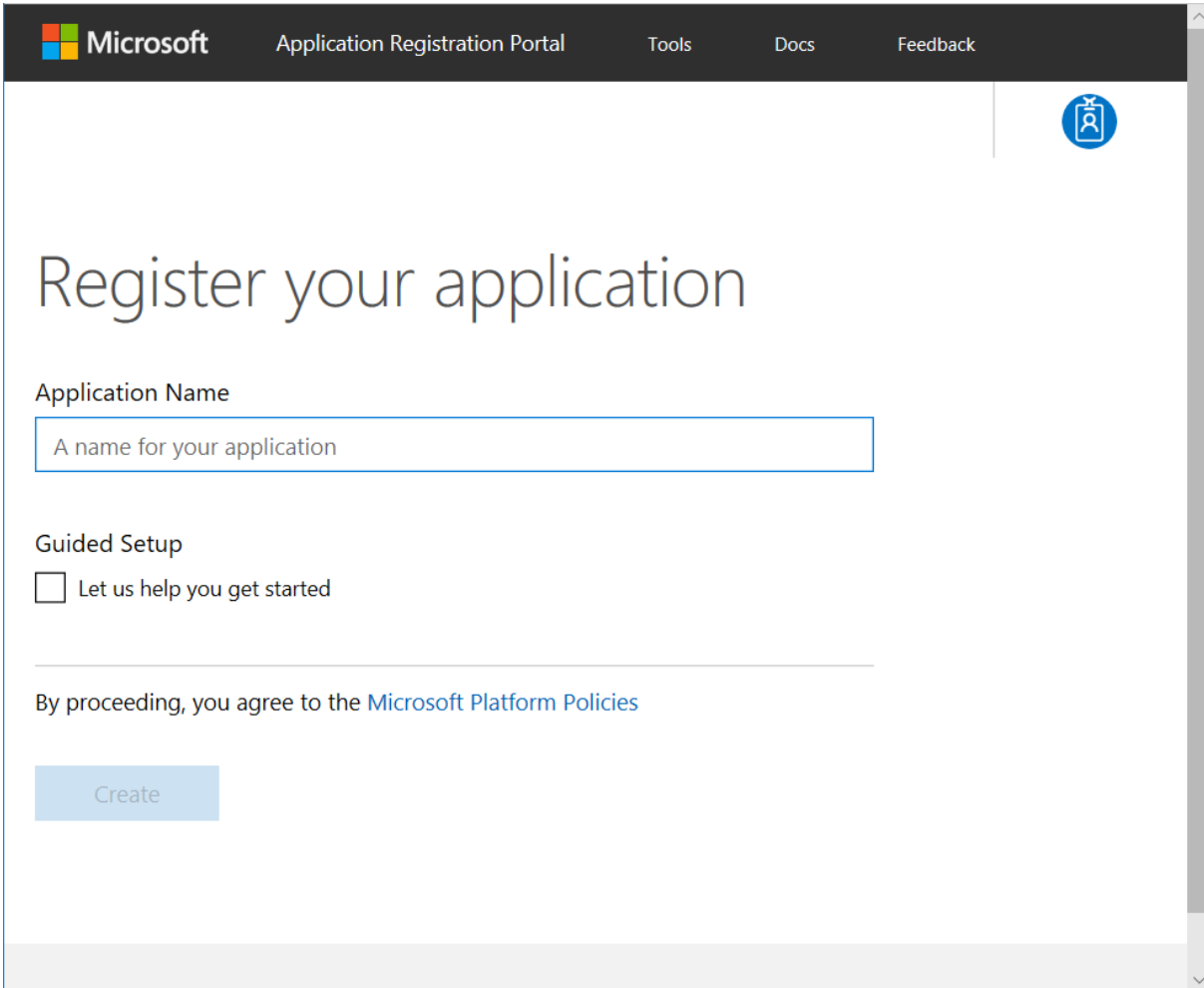
```
cd /d "%ProgramFiles%\IIS Express"
```
 3. Type the following command to add the FQDN to your application:

```
appcmd.exe set config -section:system.applicationHost/sites /+ "[name='WebApplication1'].bindings.[protocol='http',bindingInformation='*:80:www.wingtiptoy.com']" /commit:apphost
```
- Where **WebApplication1** is the name of your project and **bindingInformation** contains the port number and FQDN that you want to use for your testing.

How to Obtain your Application Settings for Microsoft Authentication

Linking an application to Windows Live for Microsoft Authentication is a simple process. If you have not already linked an application to Windows Live, you can use the following steps:

1. Browse to <https://go.microsoft.com/fwlink/?LinkID=144070> and enter your Microsoft account name and password when prompted, then click **Sign in**:
2. Select **Add an app** and enter the name of your application when prompted, and then click **Create**:

The screenshot shows the Microsoft Application Registration Portal. At the top is a dark navigation bar with the Microsoft logo, 'Application Registration Portal', 'Tools', 'Docs', and 'Feedback'. A user profile icon is in the top right. The main heading is 'Register your application'. Below it is a form for 'Application Name' with a text box containing the placeholder 'A name for your application'. Underneath is a 'Guided Setup' section with a checkbox labeled 'Let us help you get started'. A horizontal line separates this from the text 'By proceeding, you agree to the [Microsoft Platform Policies](#)'. At the bottom is a blue 'Create' button.

3. Select your app under **Name** and its application properties page appears.
4. Enter the redirect domain for your application. Copy the **Application ID** and, under **Application Secrets**, select **Generate Password**. Copy the password that appears. The application ID and password are your client ID and client secret. Select **Ok** and then **Save**.

New password generated

This is the only time when it will be displayed. Please store it securely.

lrnESTL862-~drmoOSL89)-

Ok

Optional: Disable Local Registration

The current ASP.NET local registration functionality does not prevent automated programs (bots) from creating member accounts; for example, by using a bot-prevention and validation technology like [CAPTCHA](#). Because of this, you should remove the local login form and registration link on the login page. To do so, open the *_Login.cshtml* page in your project, and then comment out the lines for the local login panel and the registration link. The resulting page should look like the following code sample:

HTML

Copy

```
<!-- ko with: login -->
<hgroup class="title">
  <h1>Log in</h1>
</hgroup>
<div class="row-fluid">
  @*<section class="span7">
    <form>
      <fieldset class="form-horizontal">
        <legend>Use a local account to log in.</legend>
        <ul class="text-error" data-bind="foreach: errors">
          <li data-bind="text: $data"></li>
        </ul>
        <div class="control-group">
          <label for="UserName" class="control-label">User
name</label>

          <div class="controls">
            <input type="text" name="UserName" data-bind="value:
```

```

userName, hasFocus: true" />
        <span class="text-error" data-bind="visible:
userName.hasError, text: userName.errorMessage"></span>
    </div>
</div>
<div class="control-group">
    <label for="Password" class="control-
label">Password</label>
    <div class="controls">
        <input type="password" name="Password" data-
bind="value: password" />
        <span class="text-error" data-bind="visible:
password.hasError, text: password.errorMessage"></span>
    </div>
</div>
<div class="control-group">
    <div class="controls">
        <label class="checkbox">
            <input type="checkbox" name="RememberMe" data-
bind="checked: rememberMe" />
            <label for="RememberMe">Remember me?</label>
        </label>
    </div>
</div>
<div class="form-actions no-color">
    <button type="submit" class="btn" data-bind="click: login,
disable: loggingIn">Log in</button>
</div>
<p><a href="#" data-bind="click: register">Register</a> if you
don't have a local account.</p>
</fieldset>
</form>
</section>*@
<section class="span5">
    <h2>Log in using another service</h2>
    <div data-bind="visible: loadingExternalLogin">Loading...</div>
    <div data-bind="visible: !loadingExternalLogin()">
        <div class="message-info" data-bind="visible: !hasExternalLogin()">
            <p>
                There are no external authentication services configured.
                See <a href="http://go.microsoft.com/fwlink/?LinkId=252166">this article</a>
                for details on setting up this ASP.NET application to
                support logging in via external services.
            </p>
        </div>
        <form data-bind="visible: hasExternalLogin">
            <fieldset class="form-horizontal">
                <legend>Use another service to log in.</legend>
                <p data-bind="foreach: externalLoginProviders">
                    <button type="submit" class="btn" data-bind="text:
name, attr: { title: 'Log in using your ' + name() + ' account' }, click:

```

```
login"></button>
        </p>
    </fieldset>
</form>
</div>
</section>
</div>
<!-- /ko -->
```

Once the local login panel and the registration link have been disabled, your login page will only display the external authentication providers that you have enabled:

