|  |
| --- |
|  |

17/3/2015

|  |
| --- |
|  |
| https://uknowit.uwgb.edu/images/group91/41011/Office365.jpg |
|  |



|  |  |
| --- | --- |
|  | Laboratorio De Autenticación y Autorización en Aplicaciones de SPO |

**Indice**

[1 Introducción 2](#_Toc414031203)

[2 Prerequisitos 2](#_Toc414031204)

[3 Exercise 1: OAuth in a Provider-Hosted App 2](#_Toc414031205)

[4 Exercise 2: OAuth with the O365 APIs 7](#_Toc414031206)

# Introducción

En este laboratorio, vamos a crear Apps que usan diferentes modelos de seguridad OAuth y examinaremos el proceso.

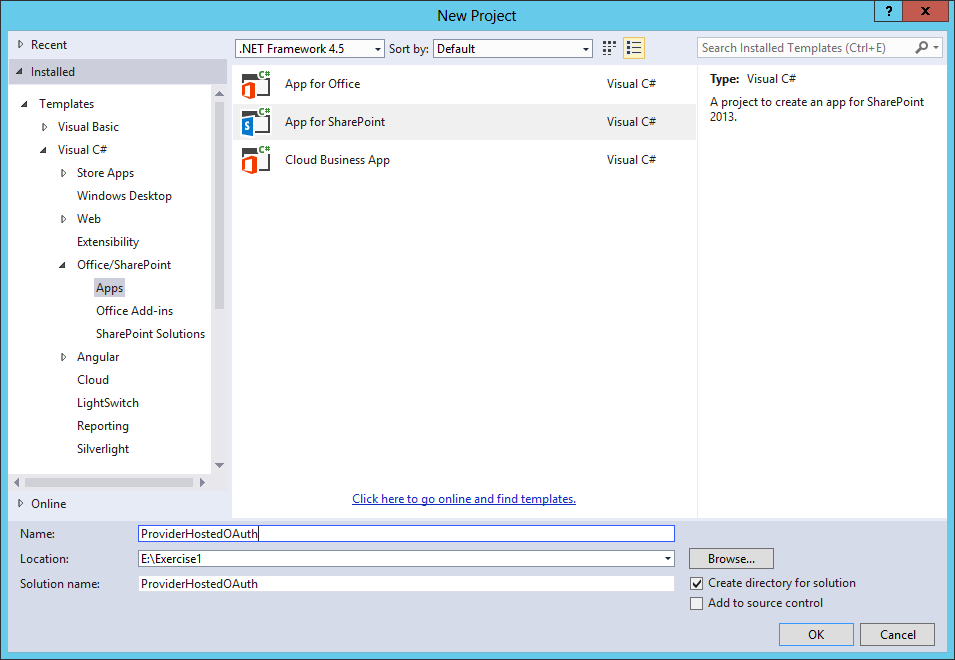
# Prerequisitos

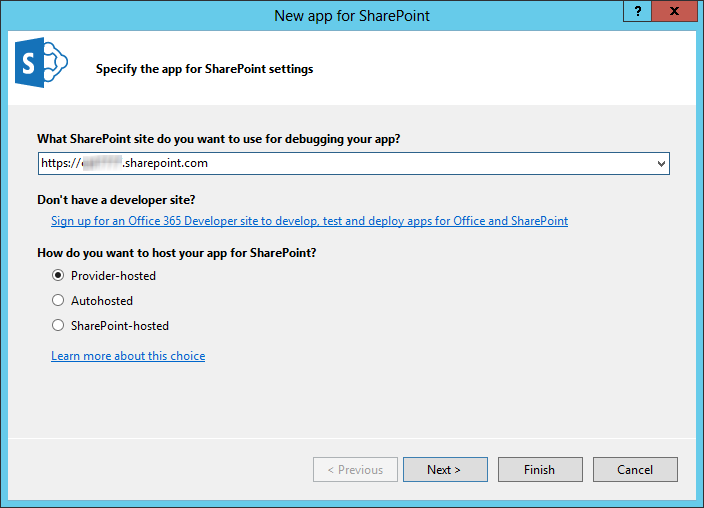
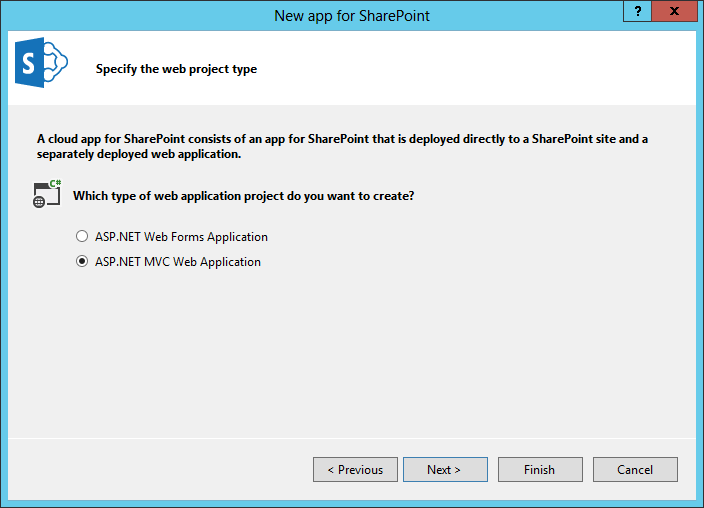
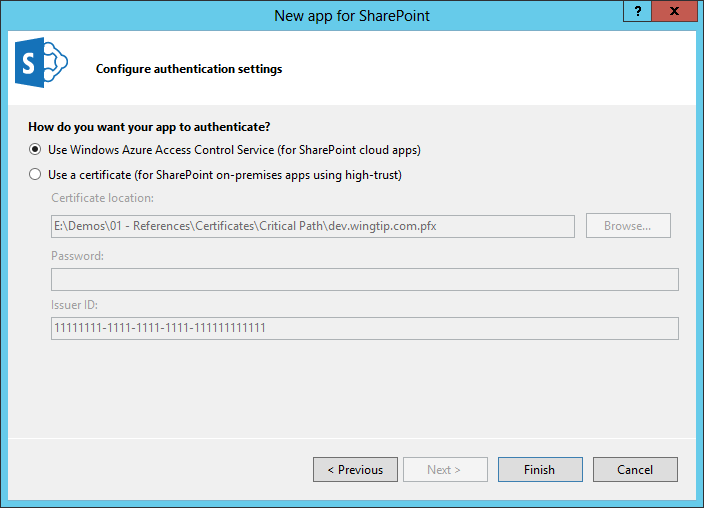
1. You must have an Office 365 tenant and Windows Azure subscription to complete this lab. If you do not have one, the lab for **O3651-7 Setting up your Developer environment in Office 365** shows you how to obtain a trial.
2. You must have [Fiddler](http://www.telerik.com/fiddler) installed.

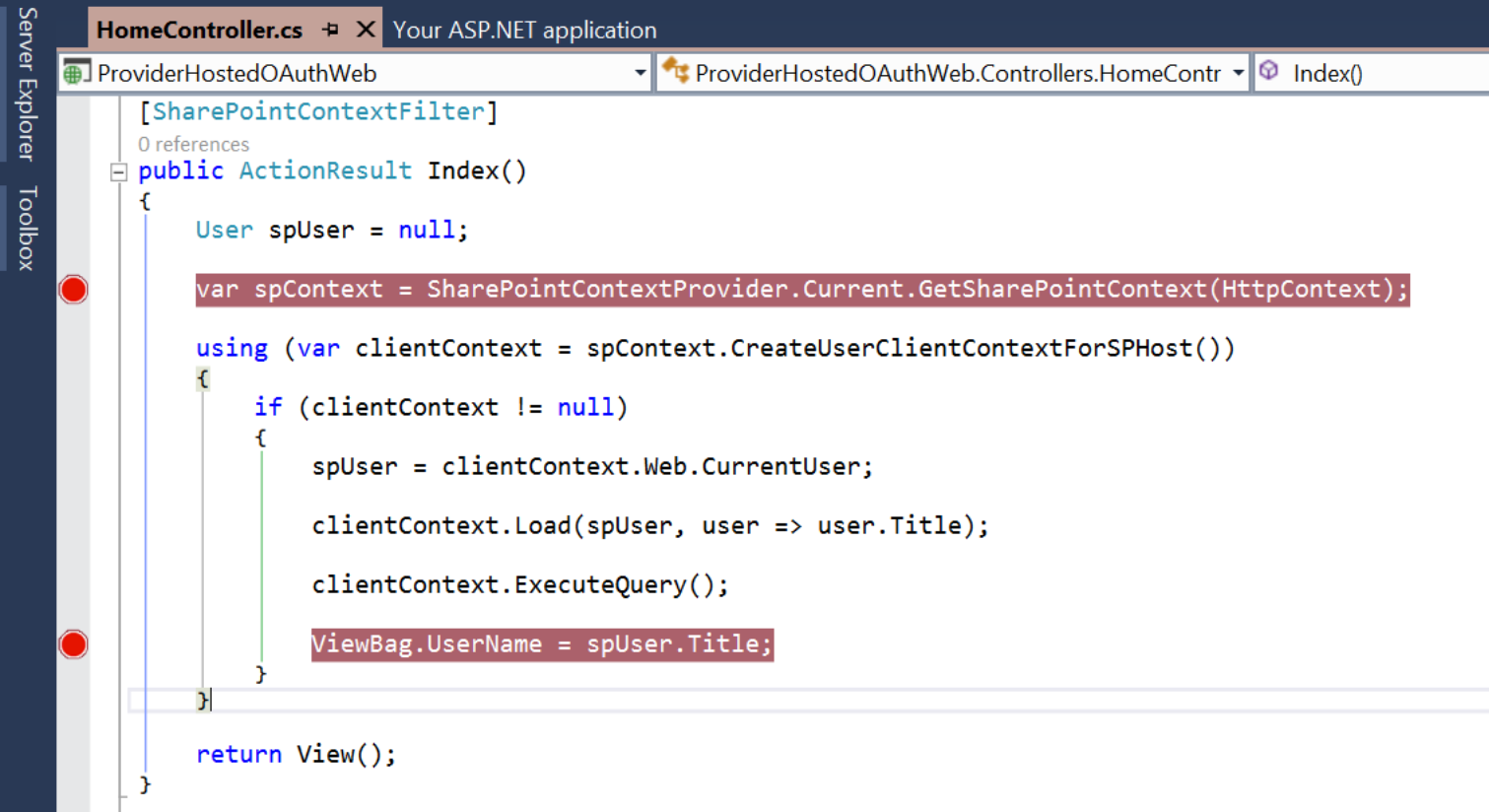
# Exercise 1: OAuth in a Provider-Hosted App

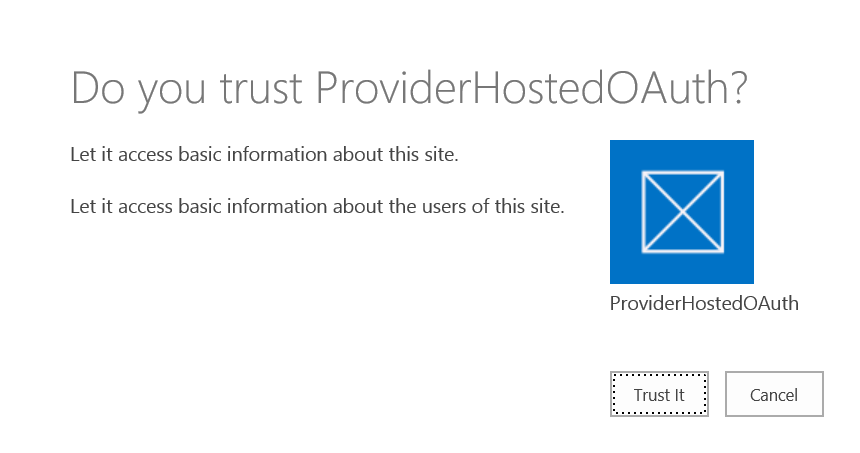
In this exercise you create a new provider-hosted app and examine the OAuth flow.

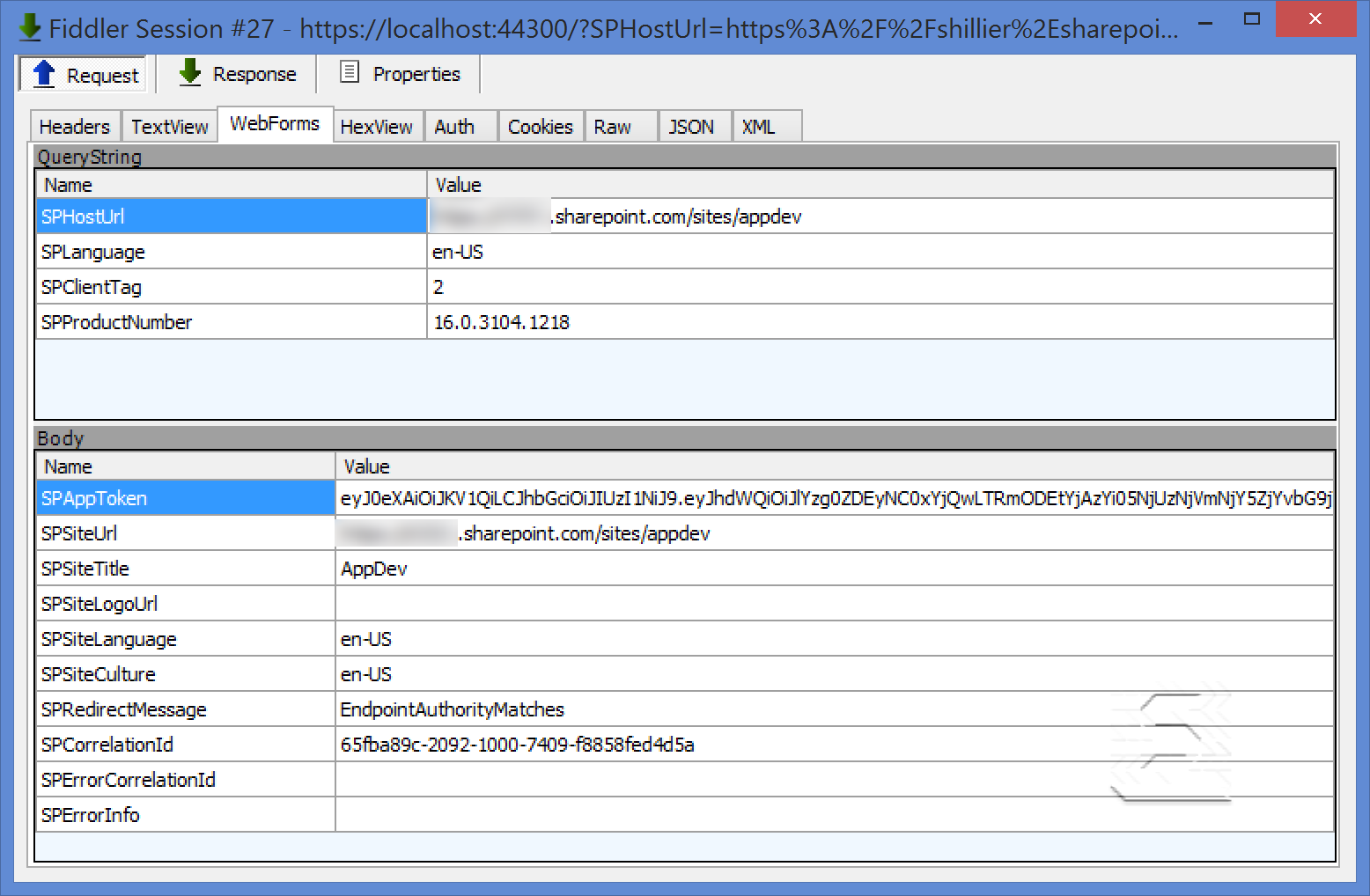
1. Create the new solution in Visual Studio 2013:
   1. Launch **Visual Studio 2013** as administrator.
   2. In Visual Studio select **File/New/Project**.
   3. In the New Project dialog:
      1. Select **Templates/Visual C#/Office/SharePoint/Apps**.
      2. Click **App for SharePoint 2013**.
      3. Name the new project **ProviderHostedOAuth** and click **OK**.

[](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/01.png?raw=true)

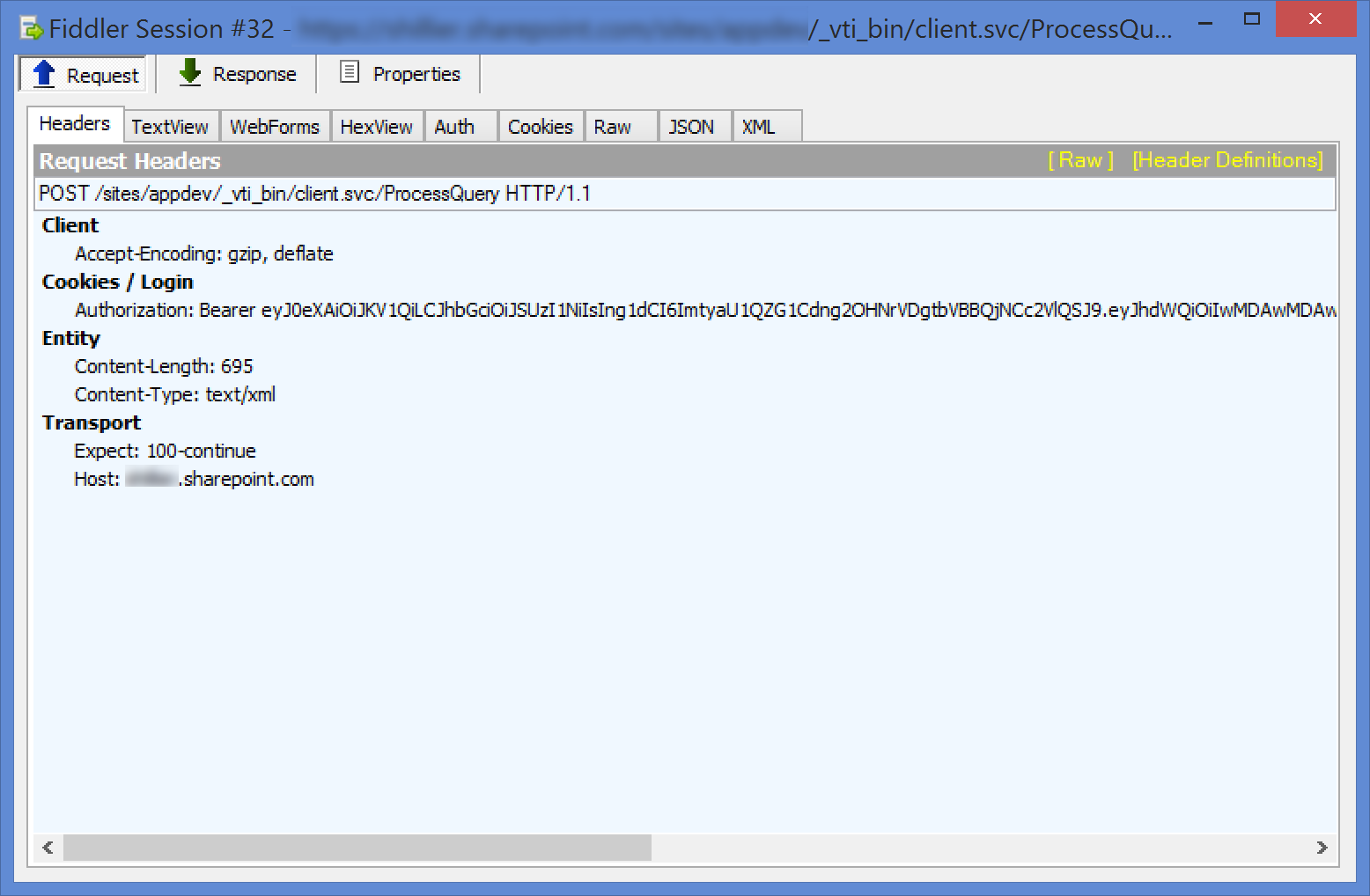
* 1. In the New App for SharePoint wizard:
     1. Enter the address of a SharePoint site to use for testing the app (***NOTE:*** The targeted site must be based on a Developer Site template)
     2. Select **Provider-Hosted** as the hosting model.
     3. Click **Next**.  
        [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/02.png?raw=true)
     4. Select **ASP.NET MVC Web Application**.
     5. Click **Next**.  
        [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/03.png?raw=true)
     6. Select the option labeled **Use Windows Azure Access Control Service (for SharePoint cloud apps)**.
     7. Click **Finish**.  
        [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/04.png?raw=true)
     8. When prompted, log in using your O365 administrator credentials.
     9. After the new project is created, set breaskpoints in **HomeController.cs** as shown

[](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/05.png?raw=true)

1. Start **Fiddler** to capture web traffic from your app.
   1. In Fiddler click **Tools/Fiddler Options**.
   2. Click **HTTPS**.
   3. Check the box entitled **Decrypt HTTPS Traffic**.
   4. When warned, click **Yes** to trust the Fiddler root certificate.
   5. Confirm any additional dialog boxes to install the certificate.
   6. Click **OK** to close the options dialog.
2. Debug the app by pressing **F5** in Visual Studio 2013.
   1. When prompted, sign into Office 365.
   2. When prompted, click **Trust It**.  
      [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/07.png?raw=true)
   3. When the first breakpoint is hit, look for the session in Fiddler near the bottom of the list.  
      [https://github.com/OfficeDev/TrainingContent/raw/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/08.png?raw=true](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/08.png?raw=true)
   4. Right click the session and select **View in New Window**.
   5. Click the **Web Forms** tab.
   6. Notice that SharePoint has included the SPHostUrl, SPLanguage, SPClientTag, and SPProductNumber query string parameters in the initial call. These are known as the **Standard Tokens**.
   7. Notice that the context token is included in the body as **SPAppToken**

. [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/09.png?raw=true)

* 1. Close the window.
  2. Return to Visual Studio, and press **F5** to continue debugging.
  3. When the second breakpoint is hit, look for the session in Fiddler near the bottom of the list.  
     [https://github.com/OfficeDev/TrainingContent/raw/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/10.png?raw=true](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/10.png?raw=true)
  4. Right click the session and select **View in New Window**.
  5. Click the **Headers** tab and examine the access token in the **Cookies/Login** section

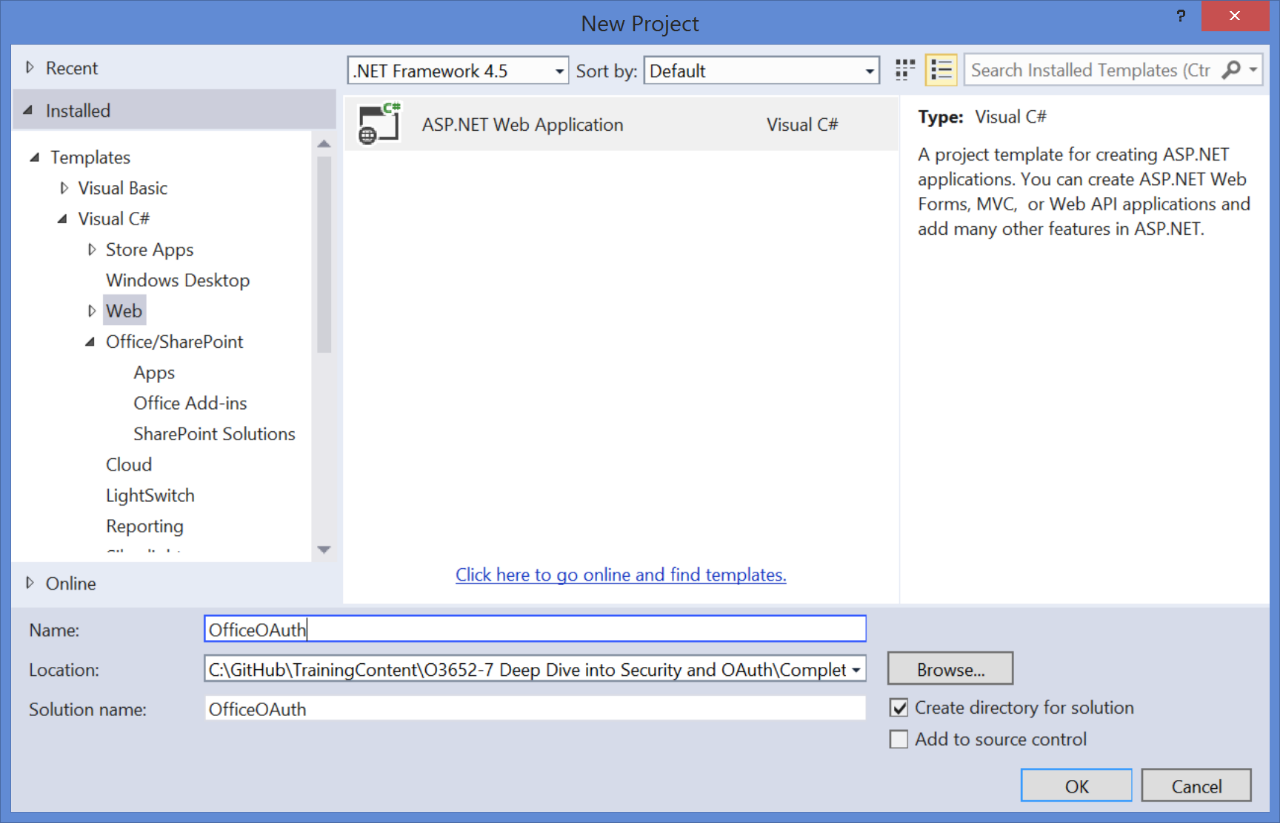
[](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/11.png?raw=true)

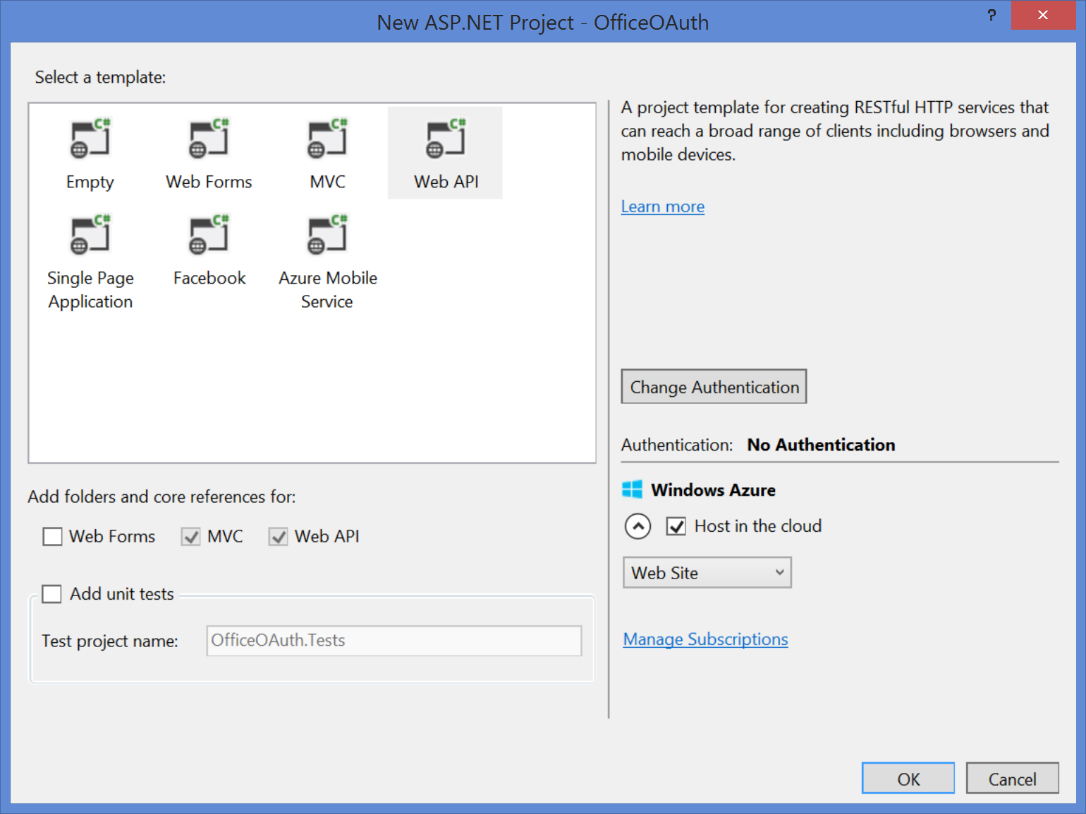
* 1. Return to Visual Studio, and press **F5** to continue debugging.
  2. With the app still running, open a new browser window to **/\_layouts/15/AppPrincipals.aspx**.
  3. Look for **ProviderHostedOAuth** in the list of registered apps to confirm that the app was registered during debugging.
  4. Stop debugging.

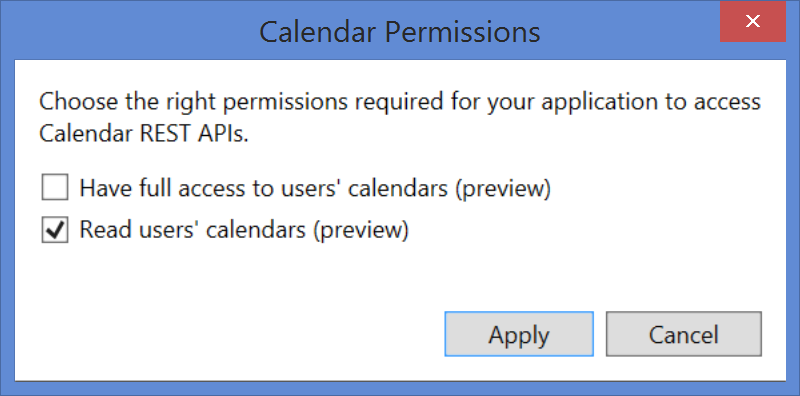
# Exercise 2: OAuth with the O365 APIs

In this exercise you create a new web applicvation and examine the OAuth flow.

1. Create the new solution in Visual Studio 2013:
   1. Launch **Visual Studio 2013** as administrator.
   2. In Visual Studio select **File/New/Project**.
   3. In the New Project dialog:
      1. Select **Templates/Visual C#/Web**.
      2. Click **ASP.NET Web Application**.
      3. Name the new project **OfficeOAuth** and click **OK**.

[](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/12.png?raw=true)

* 1. In the **New ASP.NET Project** dialog, select **Web API**.
  2. Check **Host in the Cloud**.
  3. Click **Change Authentication**.
  4. In the **Change Authentication** dialog:
     1. Click **No Authentication**.
     2. Click **OK**.
  5. Click **OK**.  
     [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/13.png?raw=true)
  6. If prompted, sign into **Windows Azure**.  
     [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/14.png?raw=true)
  7. When the **Configure Windows Azure Sites Settings** dialog appears, make appropriate selectgions for your project.
  8. Click **OK**.

1. If you do not have the **Office 365 API Tools** installed:
   1. Click **Tools/Extensions and Updates**.
   2. In the **Extensions and Updates" dialog, click \*\*Online**.
   3. Click **Visual Studio Gallery**.
   4. Type **Office 365** in the search box.
   5. Click **Office 365 API Tools - Preview**.
   6. Click **Install**.
2. Add an O365 connection
   1. Right click the **OfficeOAuth** project and select **Add/Connected Service**.
   2. In the **Services Manager** dialog, click **Sign In**.
   3. Sign in with your managed account.
   4. Click **Calendar**.
   5. Click **Permissions**.
   6. Check **Read user's calendar**.
   7. Click \*\*Apply.  
      [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/15.png?raw=true)
   8. Click **OK**.
3. Update the Home Controller.
   1. Expand the **Controllers** folder and open **HomeController.cs**.
   2. Replace the **Index** method with the following code

public async Task<ActionResult> Index()

{

IOrderedEnumerable<IEvent> events = await CalendarAPISample.GetCalendarEvents();

ViewBag.Events = events;

return View();

}

1. Update the Index View.
   1. Expand the **Views/Home** folders and open **Index.cshtml**.
   2. Replace all of tyhe code with the following

<div style="margin:25px;">

<table>

<tr>

<th>Start</th>

<th>End</th>

<th>Subject</th>

<th>Location</th>

</</tr>

@foreach (var Event in ViewBag.Events)

{

<tr>

<td>

<div style="width:200px;">@Event.Start.ToString()</div>

</td>

<td>

<div style="width:200px;">@Event.End.ToString()</div>

</td>

<td>

<div style="width:200px;">@Event.Subject</div>

</td>

<td>

<div style="width:200px;">@Event.Location.DisplayName</div>

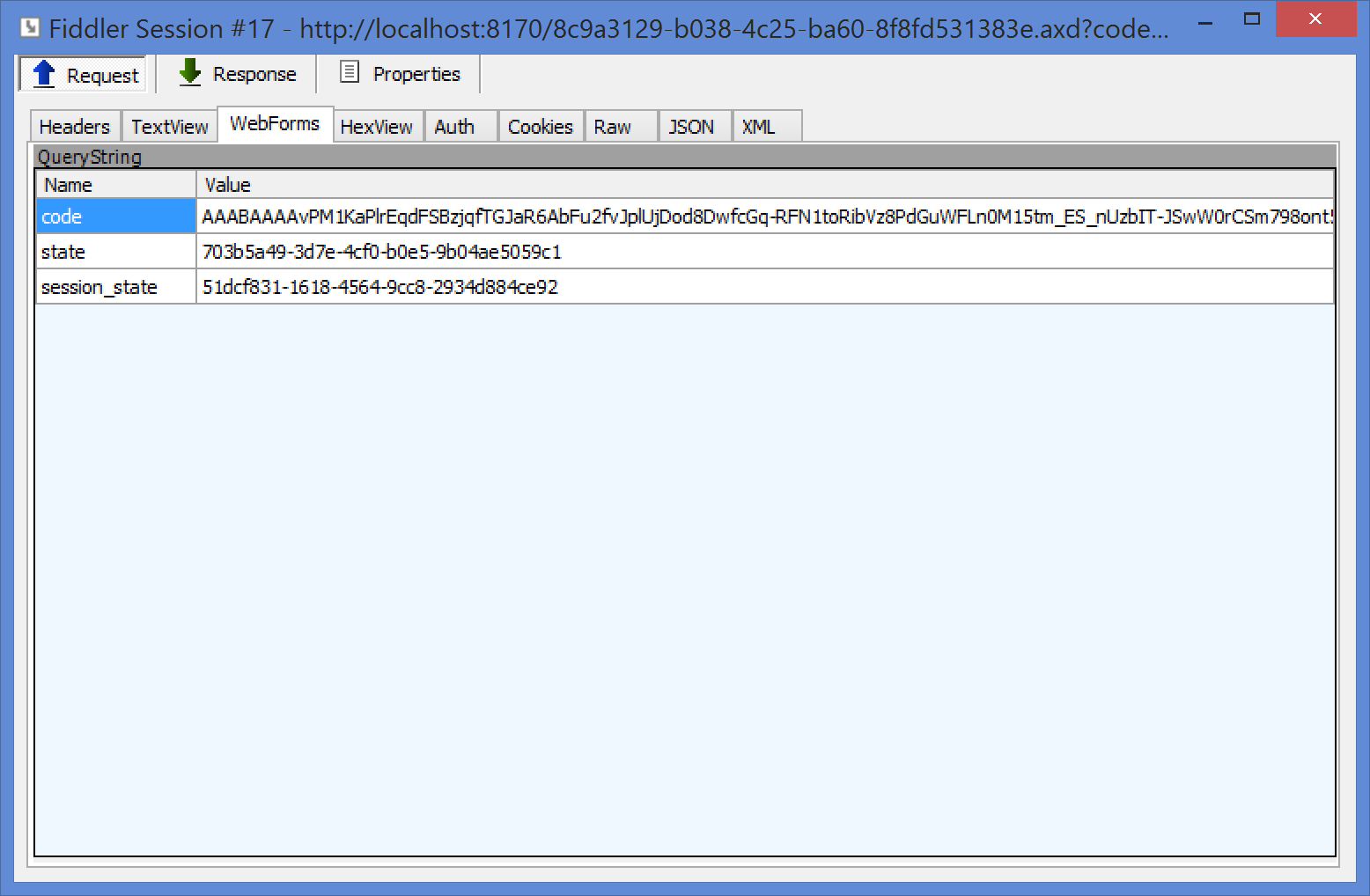
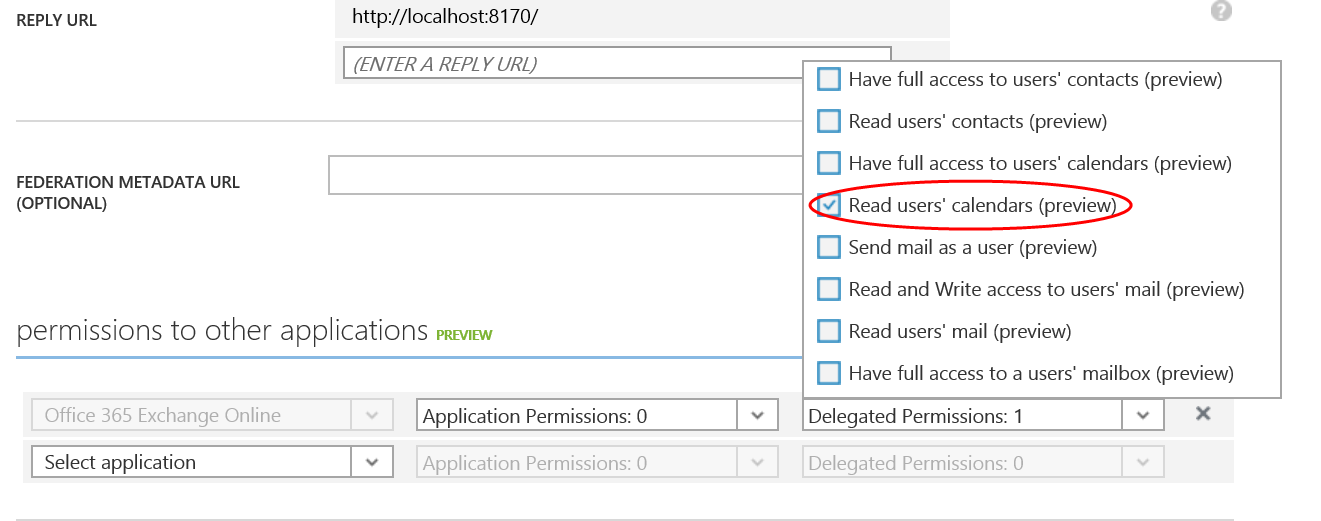
</td>

</tr>

}

</table>

</div>

1. Debug the app.
   1. Start **Fiddler**.
   2. Press **F5** in Visual Studio 2013 to debug the application.
   3. When prompted, login to Office 365 with your managed account.
   4. Verify that the application displays your calendar information.
   5. In **Fiddler**, locate the session entry containing the query string parameter **code**. This is the Authorization Code returned from Azure Access Control Services.  
      [https://github.com/OfficeDev/TrainingContent/raw/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/16.png?raw=true](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/16.png?raw=true)
   6. Right click the session and select **Inspect in New Window**.
   7. In the session window, click the **Web Forms** tab.
   8. Examine the authorization code.  
      [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/17.png?raw=true)
   9. Close the window.
   10. Stop debugging.
2. Examine the Windows Azure configurtation.
   1. Log into the [Windows Azure Portal](https://manage.windowsazure.com/)
   2. Click **Active Directory**.
   3. Select your Azure Active Directory instance.
   4. Click on the app entitled **OfficeOAuth.Office365App**. This entry was made for you by the Office 365 tools in Visual Studio.
   5. Click **Configure**.
   6. Scroll to the section entitled **Permissions to Other Applications**.
   7. Examine the **Office 365 Exchange Online** permissions. These are the permissions you granted in Visual Studio.  
      [](https://github.com/OfficeDev/TrainingContent/blob/master/O3652/O3652-7%20Deep%20Dive%20into%20Security%20and%20OAuth/Images/18.png?raw=true)