**Moving Full Trust Code to the Cloud**

In this lab, you will investigate how to use an app security principle, Windows Azure, and new client-side APIs for SharePoint Online in the Office 365 environment.

**Prerequisites**

1.          You must have an Office 365 tenant and a Windows Azure account to complete this lab. If you do not have these please raise your hand and ask a TLG for assistance.

2.          You must connect the Office 365 tenant to Azure. Perform the following steps to connect the tenant to Azure.

1.          Navigate to http://manage.windowsazure.com

2.          Sign in with your credentials for Azure.

3.          On the navigation pane on the left, click **Active Directory**. You may need to scroll down to see this entry.

4.          Click the **+NEW** button on the bottom left of the screen.

5.          On the New Object page, click **APP SERVICES**, then **ACTIVE DIRECTORY**, then **DIRECTORY**. Finally, click **CUSTOM CREATE**.

6.          In the Add Directory dialog, select **Use existing directory** from the dropdown list.

7.          Select the **I am ready to be signed out now** check box and then click the checkmark button.

8.          When the sign in page loads, use the credentials for your Office 365 tenant (user@mycompany.onmicrosoft.com) to sign in.

9.          On the confirmation page that displays click the **Continue** button. Then, click the **Sign Out Now** link.

3.          You must have a Developer site in Office 365. Perform the following steps to create a Developer site.

1.          Click the desktop shortcut named **Office 365 Portal** and sign in to your Office 365 tenant with your credentials (user@mycompany.onmicrosoft.com). Note: If you are using your own Office 365 tenant, you must sign in using credentials that have Global or SharePoint Online admin permissions to create the developer site.

2.          Locate the Admin links (the link may appear at the top or left side of the page) and click the link to manage SharePoint.

3.          In the **SharePoint admin center**, on the **Site Collections** tab, click the **New** button, and then click **Private Site Collection**.

4.          Enter information about the new Developer site and then click the **OK** button.

5.          Once the new site is created, copy the URL into a new text document by using a utility such as Notepad. You will use the URL in later steps.

**Exercise 1: Create an App Security Principle and Configure Its Permissions**

In this exercise you will create an app security principle in your Office 365 tenancy and configure it with the permissions that are required by the Console app that you will create in the next exercise.

1.          Click the desktop shortcut named **Office 365 Portal** and sign in to your Office 365 tenant with your credentials.

2.          Navigate to the new app security principle page for your developer site. If the root URL for your developer site is:  
  
https://Mysite.sharepoint.com/sites/Dev  
  
you would enter the following URL into the address bar of the browser:    
  
https://Mysite.sharepoint.com/sites/Dev/\_layouts/15/appregnew.aspx    
  
Enter the following information on the page:

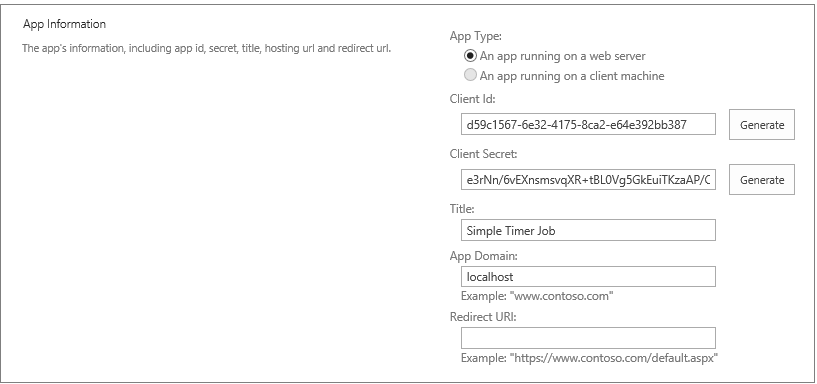
1.          Click the **Generate** button to create a new **Client Id**.

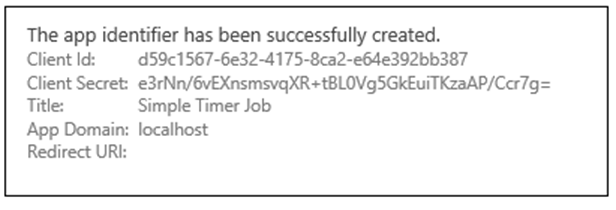
2.          Click the **Generate** button to create a new **Client Secret**.

3.          Enter **SimpleTimerJob**  as the value for **App Title**.

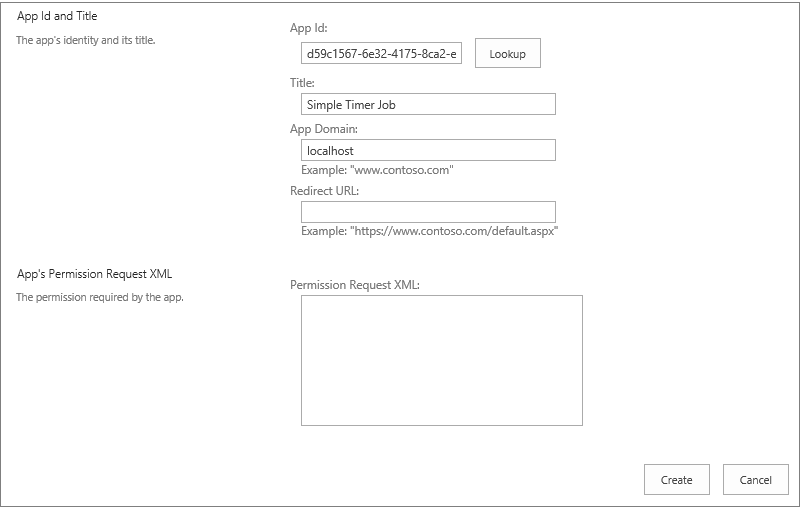
4.          Enter **localhost** as the value for **App Domain**. This allows requests to come from any URL. Note: This value should only be used in non-production environments for testing and development.

5.          Enter a **Redirect URL** of https://localhost.

6.          Click the **Create** button to create the new app security principle. 

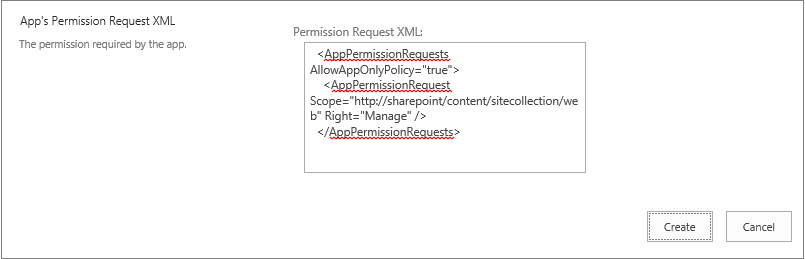
3.          After creating the new app security principle, you will see a confirmation page like the one shown in the following screenshot which lists both the **Client Id** and the **Client Secret**. Copy the **Client Id** and the **Client Secret** into a new text document by using a utility such as Notepad. You will use these values in later steps. 

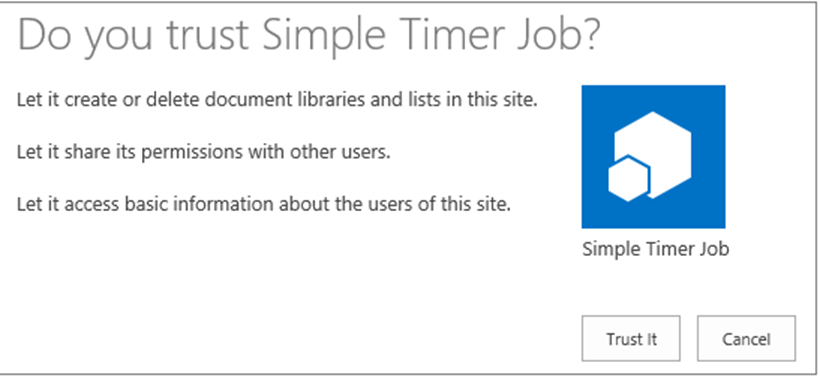
4.          Navigate to the page in your developer site that allows you to configure security principles for an existing app. If the root URL for your developer site is:  
  
https://Mysite.sharepoint.com/sites/Dev  
  
you would enter the following URL into the address bar of the browser:    
  
https://Mysite.sharepoint.com/sites/Dev/\_layouts/15/appinv.aspx

5.          Copy and paste the **Client Id** of the app security principle you created into the **App Id** textbox and click the **Lookup** button. The page will retrieve other information about the app such as the **App Title** and the **App Domain**. 

6.          Enter the following XML markup segment into the **Permission Request XML** textbox. If you would rather, you can copy-and-paste this XML from a text file named **PermissionRequestXML.txt** in the **Starter Files** folder for this lab. The Starter files are located in c:\Labfiles\O3651-4 Moving Full Trust Code to the cloud using repeatable patterns and best practices\.

1.           <AppPermissionRequests AllowAppOnlyPolicy="true">  
 < AppPermissionRequest   
   Scope="http://sharepoint/content/sitecollection/web"   
   Right="Manage" />  
</AppPermissionRequests>

7.          Click the **Create** button to save the XML markup with the app permission configuration to the SharePoint tenant. 

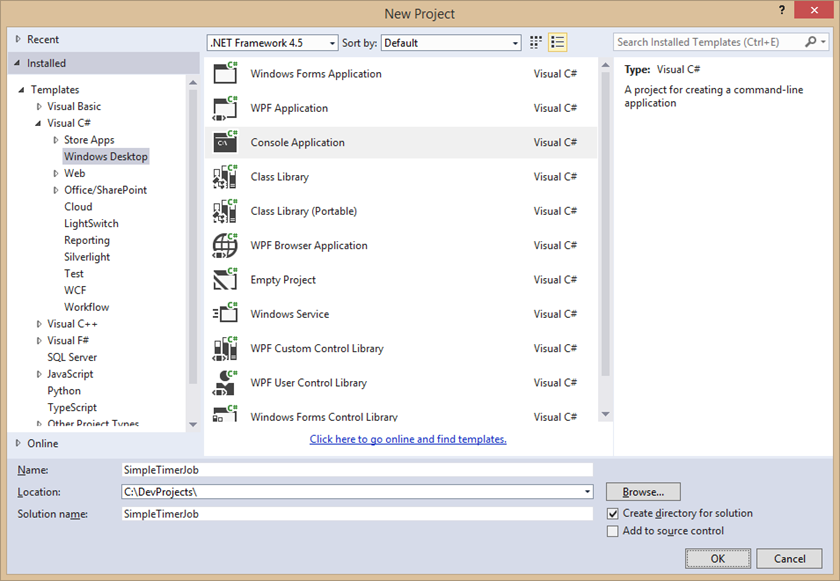
8.          When you attempt to save the app security changes, SharePoint prompts you with a page titled **Do you trust Simple Timer Job?** Click the **Trust It** buttonto confirm that you want to save the app security configuration changes. 

9.          You have now successfully created and configured a new app security principle for your Office 365 site.

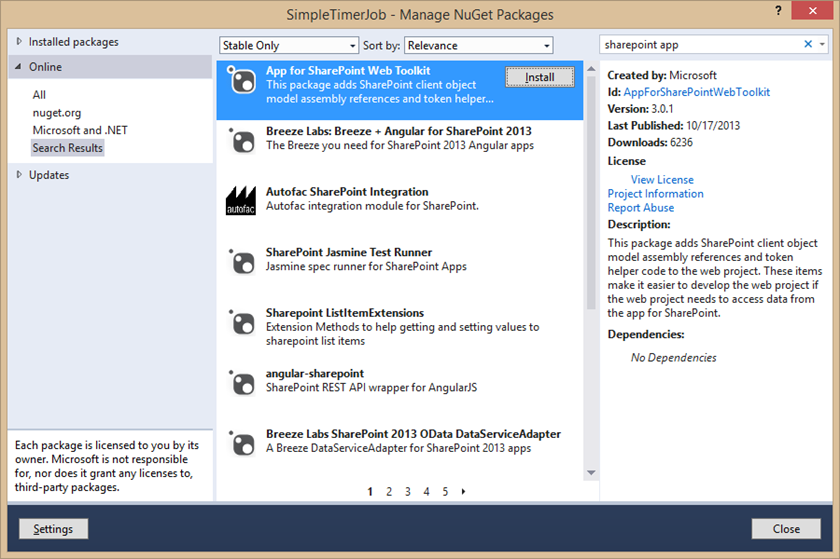
**Exercise 2: Create a Console App to Access a SharePoint Online Site**

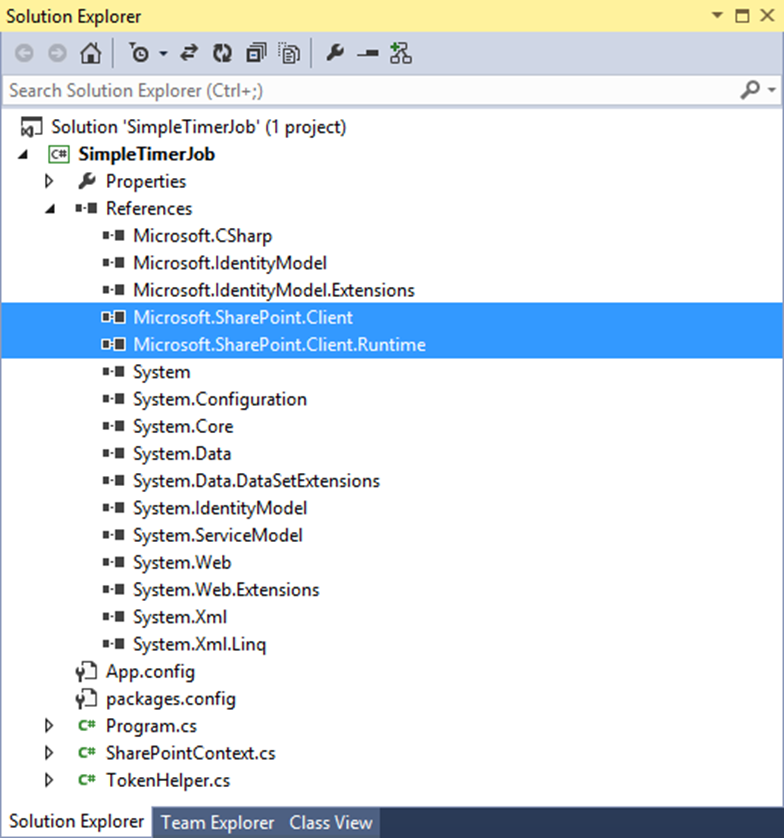
In this exercise you will use Visual Studio to create a simple console application that uses the app security principle you just created to establish its identity with your Office 365 developers site. You will write the code in the app to change the look and feel of the site by configuring it to use a different theme.

1.          Launch Visual Studio and then click **File/New/Project**.

2.          In the **New Project** dialog, select **Templates/Visual C#/Windows Desktop**. Then select **Console Application** and give the project a name of **SimpleTimerJob**. Click **OK** when you are done. 

3.          Once Visual Studio creates the **SimpleTimerJob** project, right-click the node for the project in Solution Explorer and select **Manage NuGet Packages**. Do not right-click the solution. NOTE: If Solution Explorer is not visible, you can open it from the **View** menu.

4.          In the **Manage NuGet Packages** dialog, select **Online**  and search for *"sharepoint app"* to find and install the package named **App for SharePoint Web Toolkit**.

5.          After adding the **App for SharePoint Web Toolkit** package, examine the project references and verify that the project now has references to the two core assemblies of the SharePoint Client Object Model (CSOM) named **Microsoft.SharePoint.Client.dll** and **Microsoft.SharePoint.Client.Runtime.dll** 

6.          Locate the app configuration file named **App.config** file and double-click to open the file.

7.          Add an **appSettings** section to the file. Use the values for **ClientId** and the **ClientSecret** that you recorded in exercise 1, step 3.

<appSettings>  
  <add key="ClientId" value="[[YOUR\_VALUE]]"/>  
  <add key="ClientSecret" value="[[YOUR\_VALUE]]"/>  
</appSettings>

8.          When you are done, your **app.config** file should look like the following screenshot with the exception of the values of the **ClientId** and the **ClientSecret**. Save the file.

9.          Open the **Program.cs** file.

10.      Add a **using** statement for **Microsoft.SharePoint.Client** namespace.

using Microsoft.SharePoint.Client;

11.      Inside the **Program** class, add a new static function named **URLCombine** by using the following code. If you would rather not type it in by hand, you can copy the code for this function from the **URLCombine.cs.txt** file that is in the **Starter Files** folder for this lab. The Starter files are located in c:\Labfiles\O3651-4 Moving Full Trust Code to the cloud using repeatable patterns and best practices\.

private static string URLCombine(string baseUrl, string relativeUrl) {  
  if (baseUrl.Length == 0)  
    return relativeUrl;  
  if (relativeUrl.Length == 0)  
    return baseUrl;  
  return string.Format("{0}/{1}",  
   baseUrl.TrimEnd(new char[] { '/', '\\' }),  
   relativeUrl.TrimStart(new char[] { '/', '\\' }));  
}

12.      Inside the **Program** class, add the following static function. You can copy the code for this function from the **ApplyTheme.cs.txt** that is in the **Starter Files** folder for this lab.

private static void ApplyTheme(ClientContext clientContext) {  
  Web currentWeb = clientContext.Web;  
  clientContext.Load(currentWeb);  
  clientContext.ExecuteQuery();  
  //Apply Sketch theme  
  currentWeb.ApplyTheme(  
   URLCombine(currentWeb.ServerRelativeUrl,   
   "/\_catalogs/theme/15/palette007.spcolor"),  
   URLCombine(currentWeb.ServerRelativeUrl,   
   "/\_catalogs/theme/15/fontscheme002.spfont"),  
   URLCombine(currentWeb.ServerRelativeUrl,   
   "/\_layouts/15/images/image\_bg007.jpg"),  
   false);  
   clientContext.ExecuteQuery();  
}

13.      Replace the existing **Main** function with the following code.  
  
static void Main(string[] args) {  
  Uri siteUri = new Uri("YOUR DEVELOPER SITE ROOT URL");  
  string realm = TokenHelper.GetRealmFromTargetUrl(siteUri);  
  string accessToken =   
   TokenHelper.GetAppOnlyAccessToken(TokenHelper.SharePointPrincipal,   
   siteUri.Authority, realm).AccessToken;  
   using (var clientContext =   
    TokenHelper.GetClientContextWithAccessToken(siteUri.ToString(),   
    accessToken)) {  
     ApplyTheme(clientContext);  
    }  
    Console.WriteLine("The theme has now been updated...");  
}  
  
Note: The **siteUri** address must match the root URL for the Developer site that you used to create the app security principle in Exercise 1. Replace the text “YOUR DEVELOPER SITE URL” with the URL for your developer site. For example, if your developer site uses the following URL:  
  
https://Mysite.sharepoint.com/sites/Dev  
  
you would enter the following URL to define the siteUri variable:  
  
Uri siteUri = new Uri("https://Mysite.sharepoint.com/sites/Dev");

14.      Test the app in Visual Studio by pressing the **Start** button on the toolbar to begin a debugging sessions. The app should start and then run to completion.

15.      In the browser, navigate to your Office 365 developer site. You should see that the theme of the site has been changed. This means that your code ran successfully.

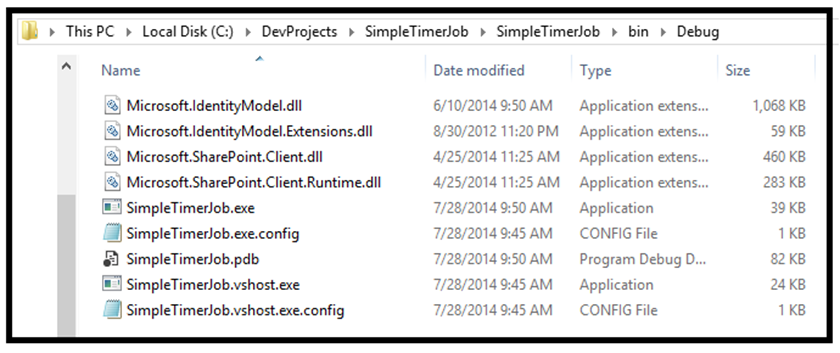
16.      In your Office 365 developer site, change the theme back to the original out-of-the-box look and feel. Accomplish this by using the **Change the look** option in the **Site Settings** menu to change the Composed Look of the site back to the original **Office** theme.

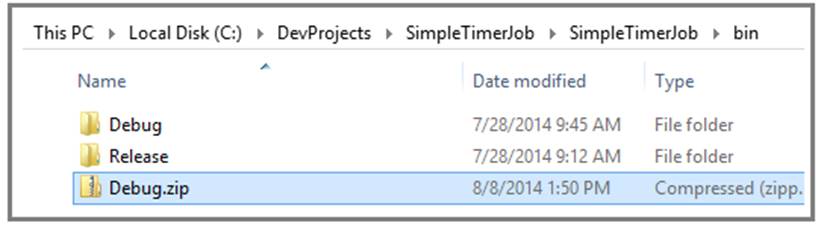
**Exercise 3: Configure the Console App to Run in Windows Azure**

In this exercise you will create a new Windows Azure Web Site and configure it to use a WebJob to run the SimpleTimerJob console application from inside the Azure cloud. You must create a zip archive for Windows Azure deployment that contains the Console app and its dependency files.

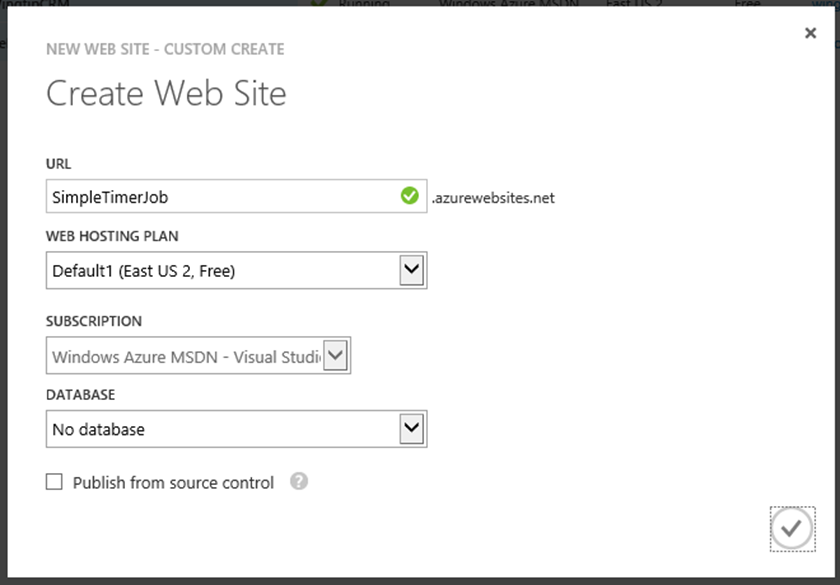
1.          Launch Windows Explorer and navigate to the root folder of the **SimpleTimerJob** project that you created in the previous exercise. The root folder is located at C:\Users\LocalAdmin\Documents\Visual Studio 2013\Projects

2.          Navigate to the child folder inside the root project folder that is named **SimpleTimerJob**.

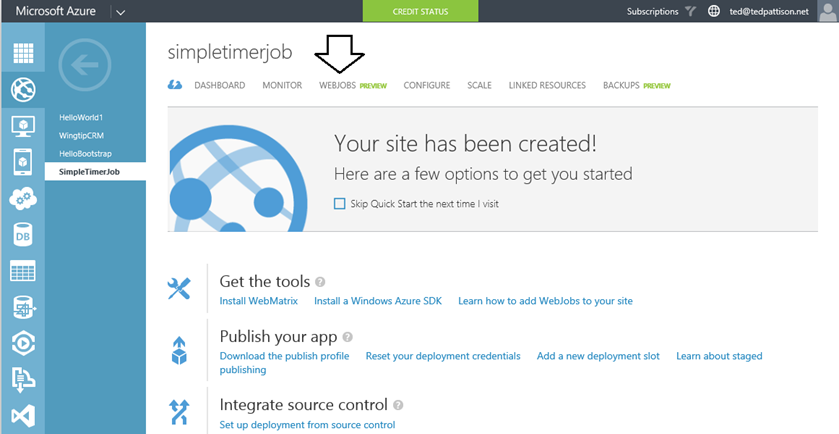
3.          Inspect the files that Visual Studio has added to the **bin\Debug** folder. You should see the main EXE file named **SimpleTimerJob.exe** and an associated set of DLLs. 

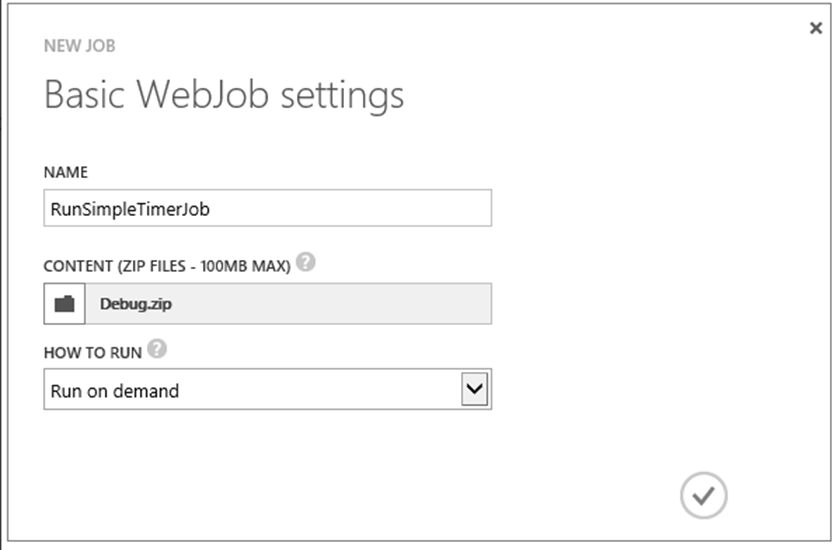
4.          In Windows Explorer, right-click on the **Debug** folder and select the **Send To > Compressed (zipped) file** command to create a new zip archive named **Debug.zip** which contains all the files required to deploy the Console app to Windows Azure. 

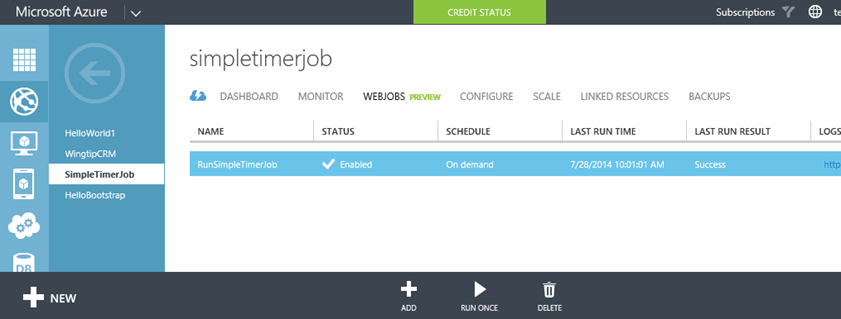
5.          Sign in to the Windows Azure manage portal at **https://manage.windowsazure.com** using your Azure account credentials.

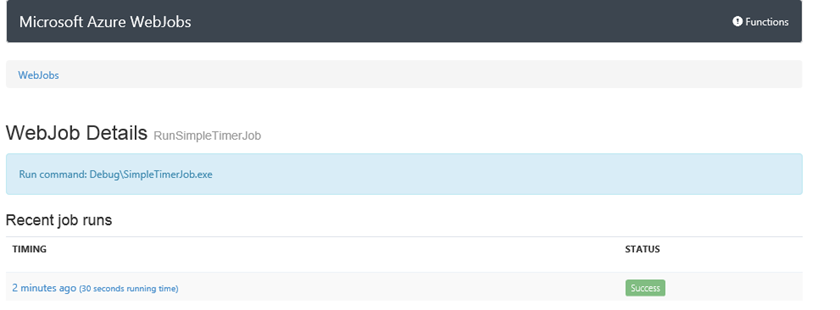
6.          Create a new Azure Web App with a URL such as **SimpleTimerJobazurewebsites.net**. Note: The name must be unique. You can add a sequential numbers like 001 to the name to guarantee uniqueness.

7.          Once the new Web App has been created, you should see the name of the Web App on the Web Apps page.  Click on the name of the new Web App.

8.          Click the **WEBJOBS** link to display the empty list of WebJobs for the new Web App. 

9.          **Add** a new WebJob named **RunSimpleTimerJob**. Add the **Debug.zip** file created earlier in this exercise as the **CONTENT** file and configure the **How To Run** setting as **Run on demand** as shown in the following screenshot. When you are done, click the button with the check mark in the bottom right of the page to save your changes and create the new WebJob. 

10.      Run the **RunSimpleTimerJob** WebJob by selecting it and clicking the **RUN ONCE** button with the right arrow icon at the bottom center of the screen. 

11.      Inside the Windows Azure portal, verify that the WebJob ran successfully. 

12.      In the browser, navigate to your Office 365 developer site. You should see that the theme of the site has been changed. That means your code ran successfully from inside Windows Azure.

13.      In your Office 365 developer site, change the theme back to the original out-of-the-box look and feel. Accomplish this by using the **Change the look** option in the **Site Settings** menu to change the site's look and feel back to the original **Office** theme.