Developing SharePoint-hosted Add-ins

Lab Time: 40 minutes

Lab Folder: C:\Student\SharePointHostedAddins\Lab

Lab Overview: In this lab you will create a new Developer site and also create two new SharePoint-hosted Add-in projects to get some experience developing and testing SharePoint-hosted add-ins and custom app parts.

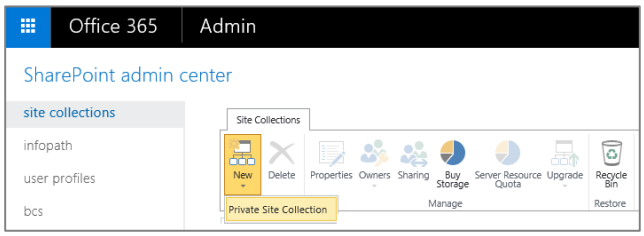
Exercise 1: Creating a New Developer Site for Testing

In this exercise you will prepare your Office 365 development environment by ensuring you have access to a SharePoint Developer site that's been created within the tenancy associated with your Office 365 developer account.

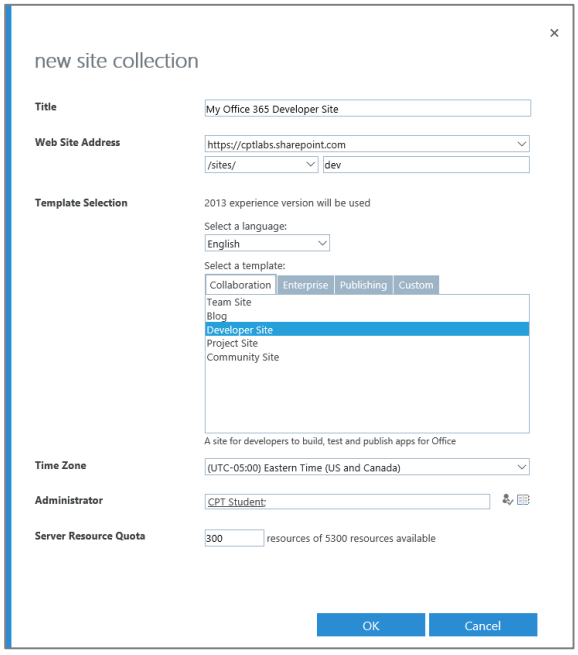
1. If you have already created a Developer site in SharePoint online, navigate to this Developer site in the browser to ensure that you can log in with your Office 365 developer account credentials and see the home page. Once you can access the Developer site with the browser, you can move ahead to exercise 2.
2. If you have not already created a Developer site in SharePoint online, you must do so before continuing to the next exercise.
   a) Using the browser, navigate to the SharePoint admin center (i.e. tenant admin site collection) for your SharePoint Onlone tenancy at the following URL.

   ```
   https://[YOUR_TENANCY].admin.sharepoint.com
   ```

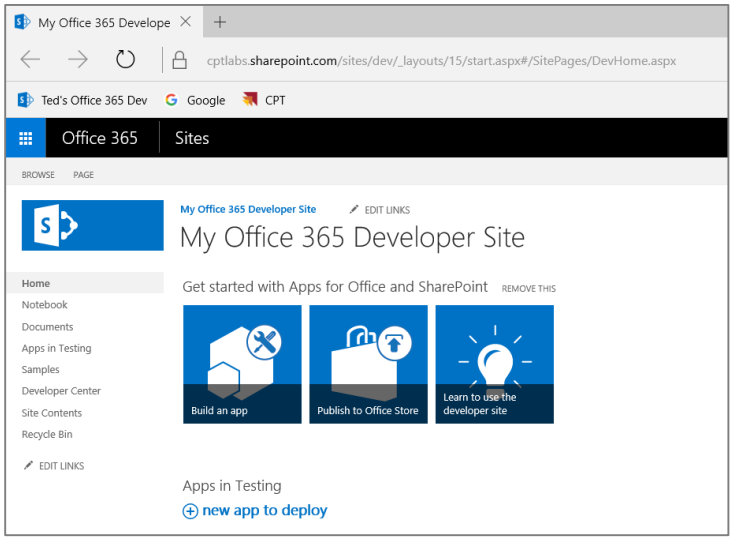
   b) With **site collections** selected on the left-side of the page, drop down the **New** menu and click **Private Site Collection**.



   c) In the **new site collection** form, fill in the required information and select the site template titled **Developer Site**. Click the **OK** button to create the new Developer site.



d) Wait for the site collection to be created. This may take a minute or two. Once the Developer site has been created, navigate to this site in the browser to ensure you can get to the home page.

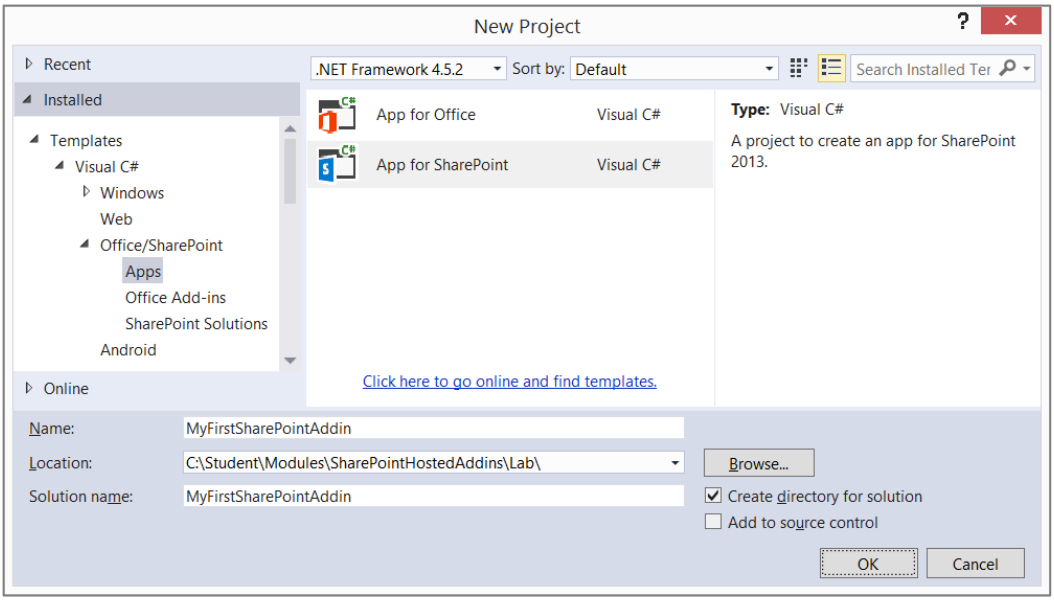


Now that you have a Developer site, you can begin to use it as you develop, test and debug SharePoint add-ins.

Exercise 2: Creating and Debugging a SharePoint-hosted Add-in

In this exercise you will create and test a very simple SharePoint-hosted add-in project.

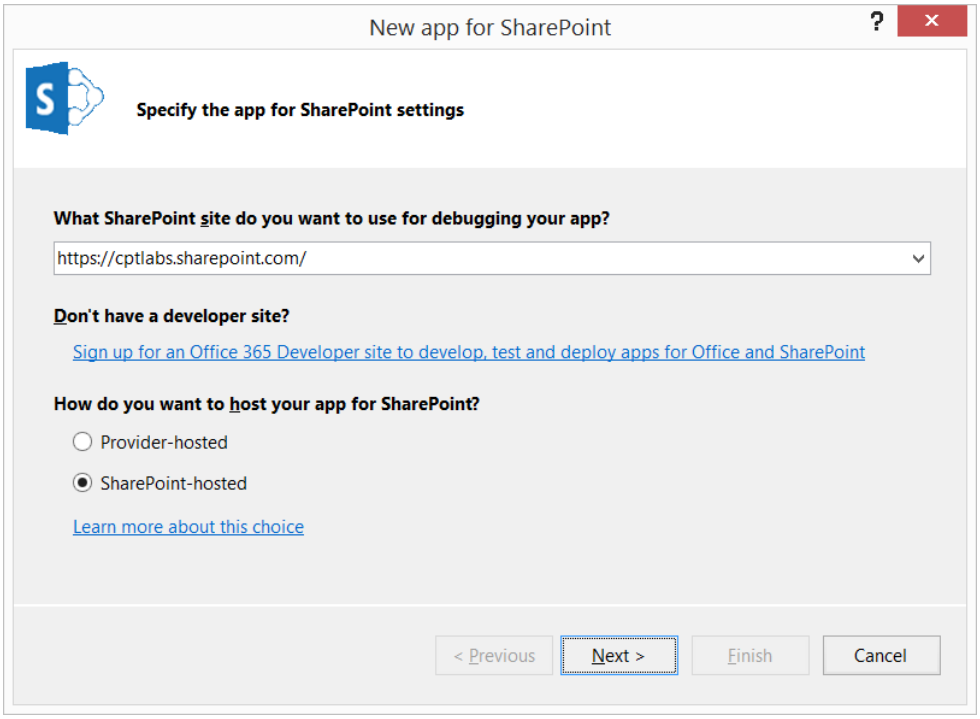
1. Create a new project in Visual Studio 2015:
   a) Launch **Visual Studio 2015** as administrator.
   b) In Visual Studio select **File → New → Project**.
   c) In the **New Project** dialog, find the **App for SharePoint** template (Templates → Visual C# → Office / SharePoint → Apps)
   d) **Name**: MyFirstSharePointHostedAddin
   e) **Location:** C:\Student\Modules\SharePointHosted\Lab
   f) **Solution name:** MyFirstSharePointHostedAddin
   g) Click **OK** to create the new project**.**



h) In the **New App for SharePoint** wizard, use the following values to complete the wizard and click **Finish**.

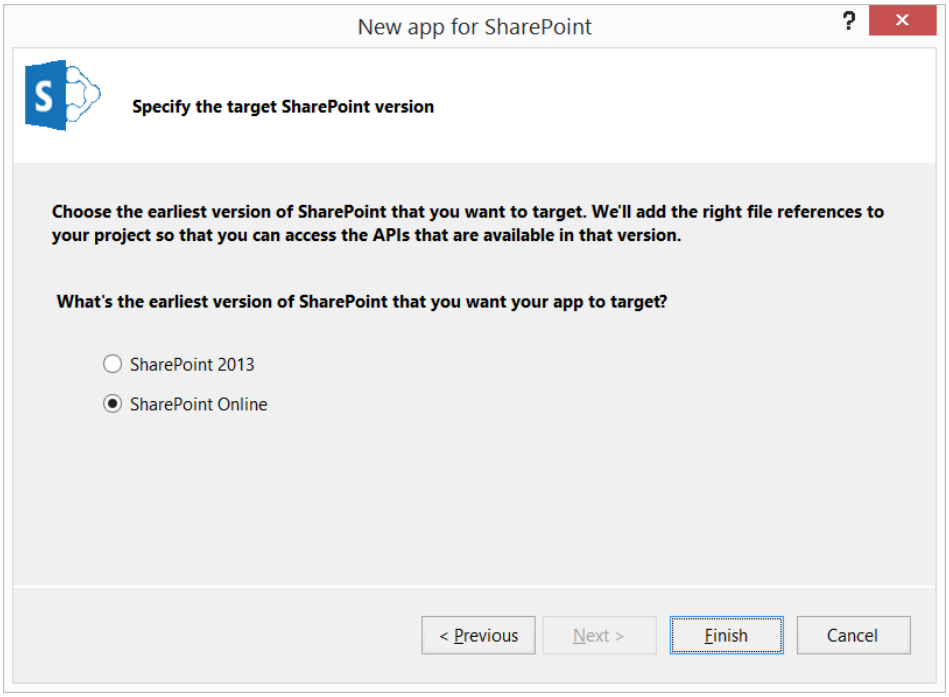
i) **What site do you want to use for debugging?** *[Enter the URL for your Developer site]*

ii) **How do you want to host your app for SharePoint?** SharePoint-hosted



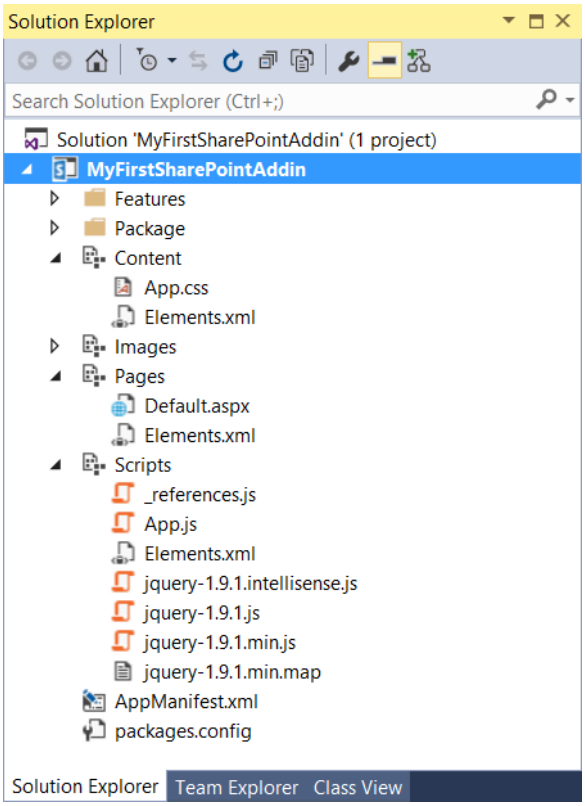
If you are prompted to log in, enter your credentials to log in using your Office 365 developer account.

i) On the **Specify the target SharePoint version** page, accept the default selection of **SharePoint Online** and click **Finish**.



j) Wait for Visual Studio to complete the process of creating the new SharePoint add-in project.

2. Take a moment and inspect the file and folder structure of the new project.



3. Here are a few things you should observe about the new SharePoint-hosted add-in project.
   a) Like a traditional SharePoint solution-based project you have a **Features** and **Packages** node.
   b) There are four folders named **Content**, **Images**, **Pages** and **Scripts** which are a special type of project item known as SharePoint Modules which provision their files to the respective folders in the add-in web during add-in installation.
   c) **Content/App.css** is an empty Cascading Style Sheet file which can be used to style the appearance of the add-in.
   d) **Images/AppIcon.png** is the default image used for the add-in.
   e) **Pages/Default.aspx** is the default start page for the add-in.
   f) **Scripts/_references.js** is an Intellisense support file for JavaScript libraries. This file isn't provisioned to SharePoint.
   g) **Scripts/App.js** is the main file that containing JavaScript code which controls the behavior and core logic for your add-in.
   h) The **Scripts** folder contains source files for the jQuery library.
   i) **AppManifest.xml** is the add-in manifest. It tells SharePoint the basic information it needs about the add-in such as:
      i) Name, Product ID, App Version Number and minimum version for the SharePoint host environment.
      ii) Security configuration and permissions.
      iii) App Title to display on app launcher tile on Site Contents page of the host web.
      iv) The URL of the app's start page.
4. Customize the start page for the SharePoint add-in:
   a) Using the **Solution Explorer** tool window, right-click the **Pages/Default.aspx** file and select **Open**.
   b) Inspect (*but do not modify*) the content placeholder with the ID of **PlaceHolderAdditionalPageHead**. You can see there are references to the jQuery library and to the **App.js** file as well as a reference to the **App.css** file as well.
   c) Locate the **PlaceHolderPageTitleInTitleArea** placeholder and update its content to match the following code listing.

```
<asp:Content ContentPlaceHolderID="PlaceHolderPageTitleInTitleArea" runat="server">
  My First SharePoint Add-in
</asp:Content>
```

   d) Locate the **PlaceHolderMain** placeholder and remove the content inside. Add content into the **PlaceHolderMain** placeholder to match the following code listing.

```
<asp:Content ContentPlaceHolderID="PlaceHolderMain" runat="server">
```

```
  <div id="toolbar">
    <input type="button" id="cmdPressMe" value="Press Me" />
  </div>

  <div id="content_box" />

</asp:Content>
```

e) Save and close **default.aspx**.

5. Add some CSS code to style the HTML elements that were added to the add-in start page.
   a) Using the **Solution Explorer** tool window, right-click the **Content/app.css** file and select **Open**.
   b) Update the contents of the app.css file to match the following code listing.

```
#toolbar {
  border: black solid 1px;
  border-radius: 8px;
  padding: 8px;
  background-color: #EEE;
}

#content_box {
  margin-top: 8px;
  font-size: 18px;
  color: blue;
}
```

   c) Save and close **app.css**.

6. Write a little JavaScript code to give your add-in some behavior:
   a) Using the **Solution Explorer** tool window, right-click the **Scripts / app.js** file and select **Open**.
   b) Update the contents of the app.css file to match the following code listing.

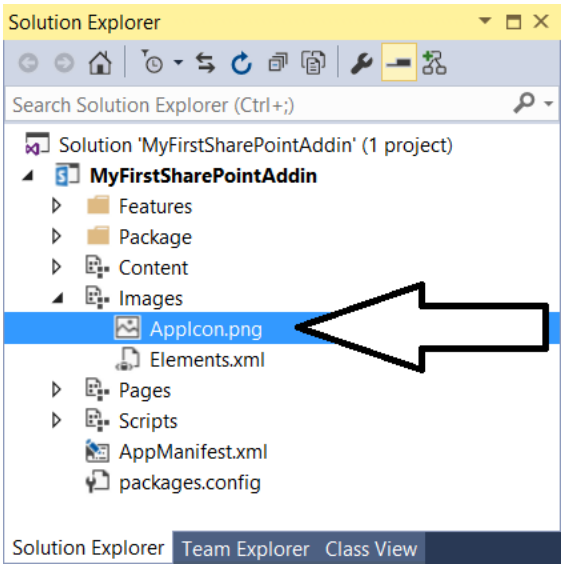
```
'use strict';

$(onPageLoad);

function onPageLoad() {
  $("#cmdPressMe").click(onPressMe);
}

function onPressMe() {
  $("#content_box").text("Hello SharePoint Add-ins");
}
```

   c) Save and close **app.js**.

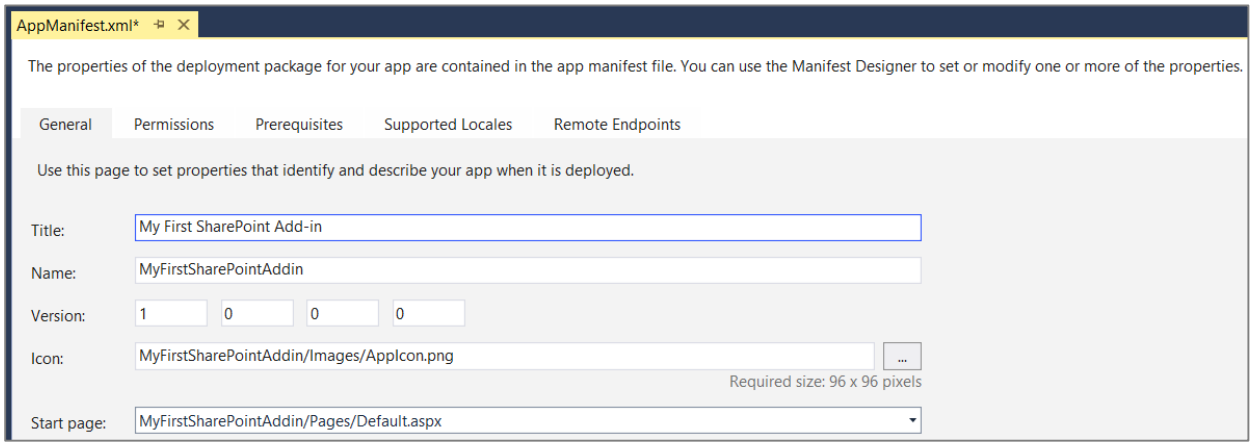
7. Replace the add-ins icon with a custom icon.
   a) Open Windows explorer and locate the following file in the **Student** folder.

```
C:\Student\ExtraStudentFiles\Images\AppIcon.png
```

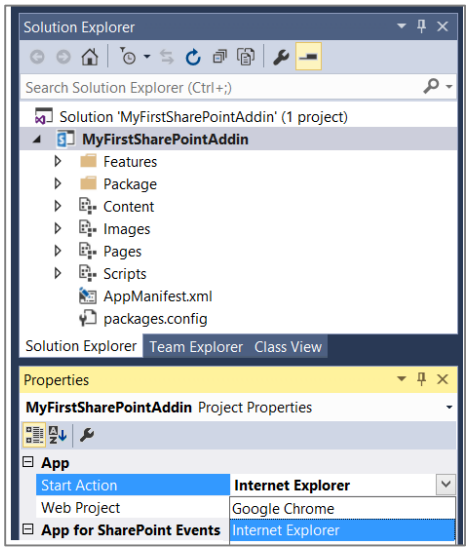
   b) Use the **AppIcon.png** file from the **Student** folder to replace the **AppIcon.png** file in the **Images** folder of the add-in project.



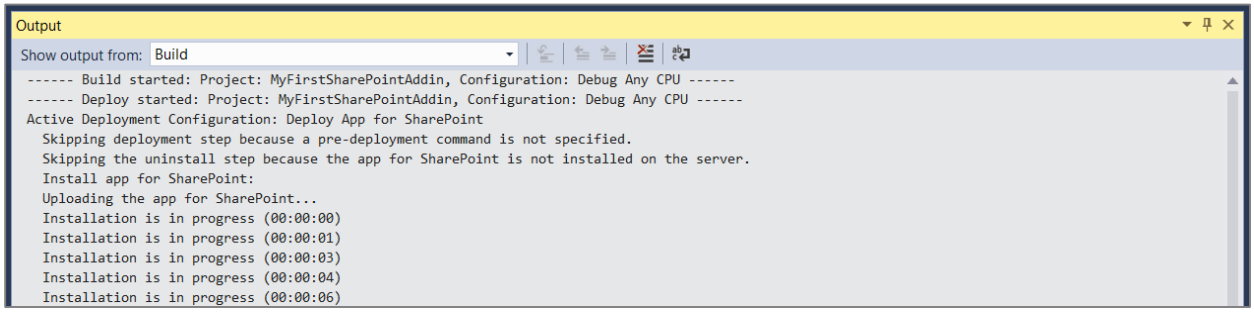
8. Modify the add-in manifest to provide a better title for the add-in:
   a) Using the **Solution Explorer** tool window, right-click the **AppManifest.xml** file and select **Open**.
   b) Update the add-in **Title** property to **My First SharePoint Add-in** so that it's more human-readable



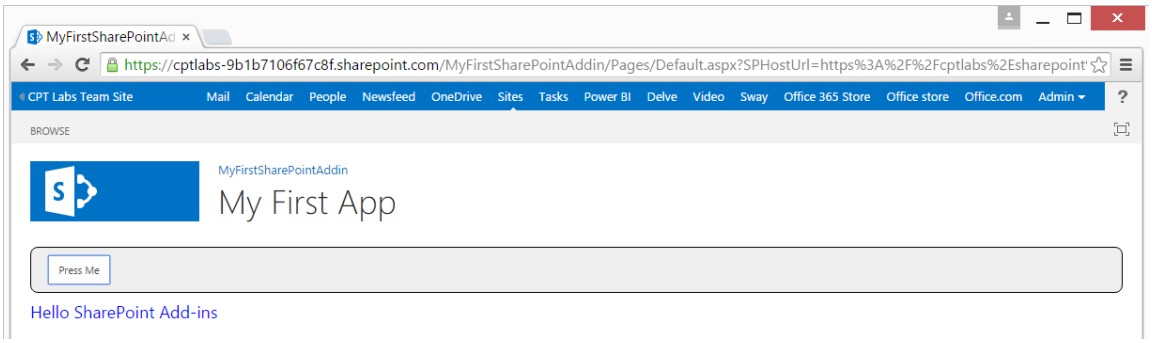
   c) Save and close **AppManifest.xml**.
9. Configure which browser to use when debugging the add-in project with the Visual Studio debugger.
   a) In Solution Explorer, select the top-level node for the **MyFirstAddinProject**.
   b) Locate the project-level **Start Action** property in the Properties window.
   c) Set the **Start Action** property to the browser of your choosing such as **Internet Explorer** or **Chrome**. Note that you can start one debugging session using Internet Explorer and then start the next debugging session using Chrome.



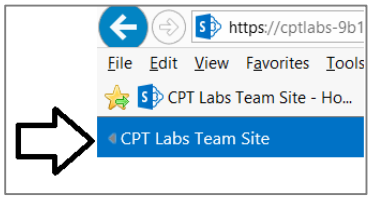
10. Build and test your application by pressing **{F5}** or **Debug → Start Debugging**.
    a) Pressing {F5} builds and add-in package and begin the add-in installation process. The installation process for an add-in can take a minute or two to complete the first time you press {F5}. You can monitor the debugger's progress in the **Output** window.



    b) Once the solution has been deployed, Visual Studio launches a browser session and you should be redirected to the add-in start page which is **default.aspx**.
    c) When the page loads, click the **Push me!** button to see your text get written to the page:

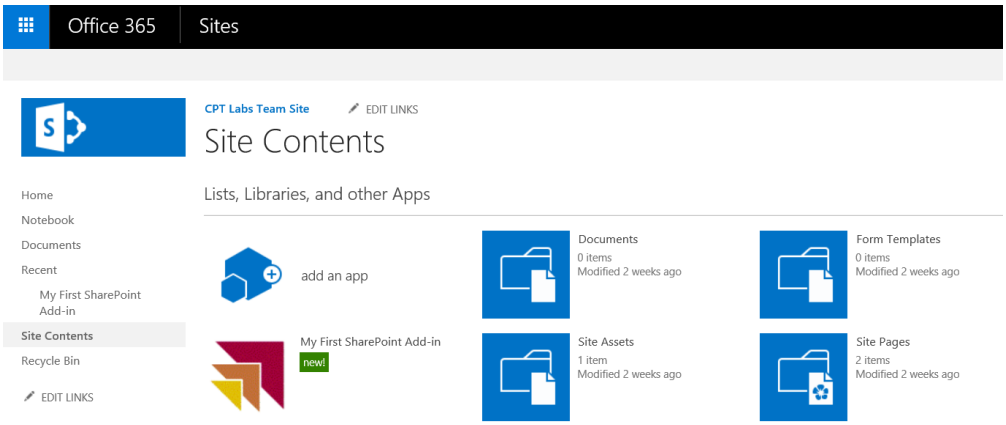


11. Return to the host web and see where SharePoint Online displays the custom logo for this add-in.
    a) Click the link on the far left of the link bar to navigate to the host web.



b) On the site's **Quick Launch** navigation (on the left side of the screen) click on **Site Contents**.

c) On the **Site Contents** page notice the icon for the app we just deployed My First SharePoint Hosted App.



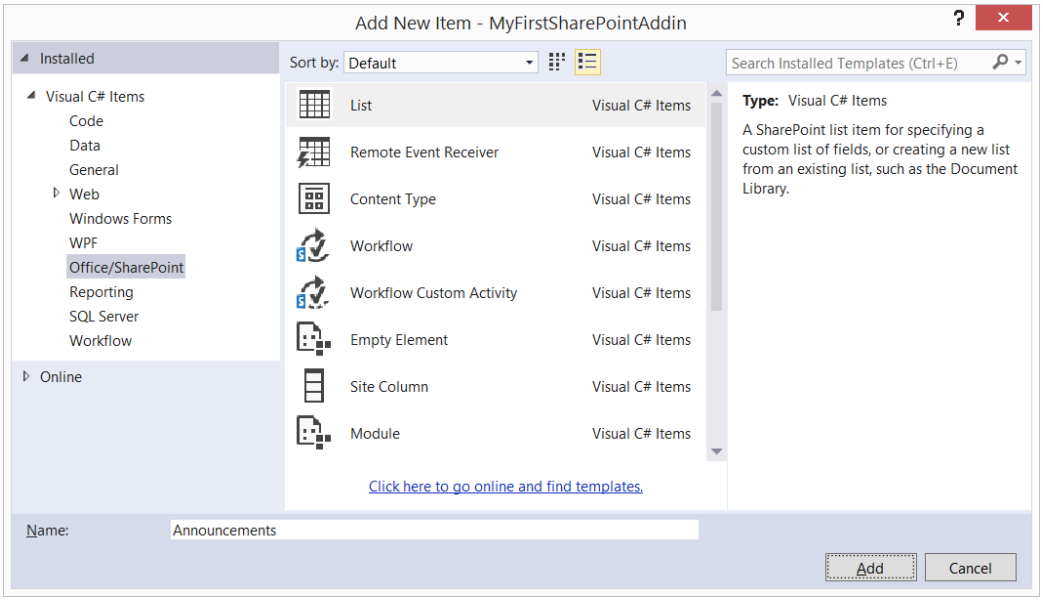
12. Close the browser to stop the debugger and return to Visual Studio.

Configure the SharePoint-hosted add-in To Create a List in the Add-in Web

13. Add a new project item to create a SharePoint list in the add-in web during installation:

a) Using the **Solution Explorer**, right-click the **My First SharePoint Hosted App** project and select **Add → New Item**.

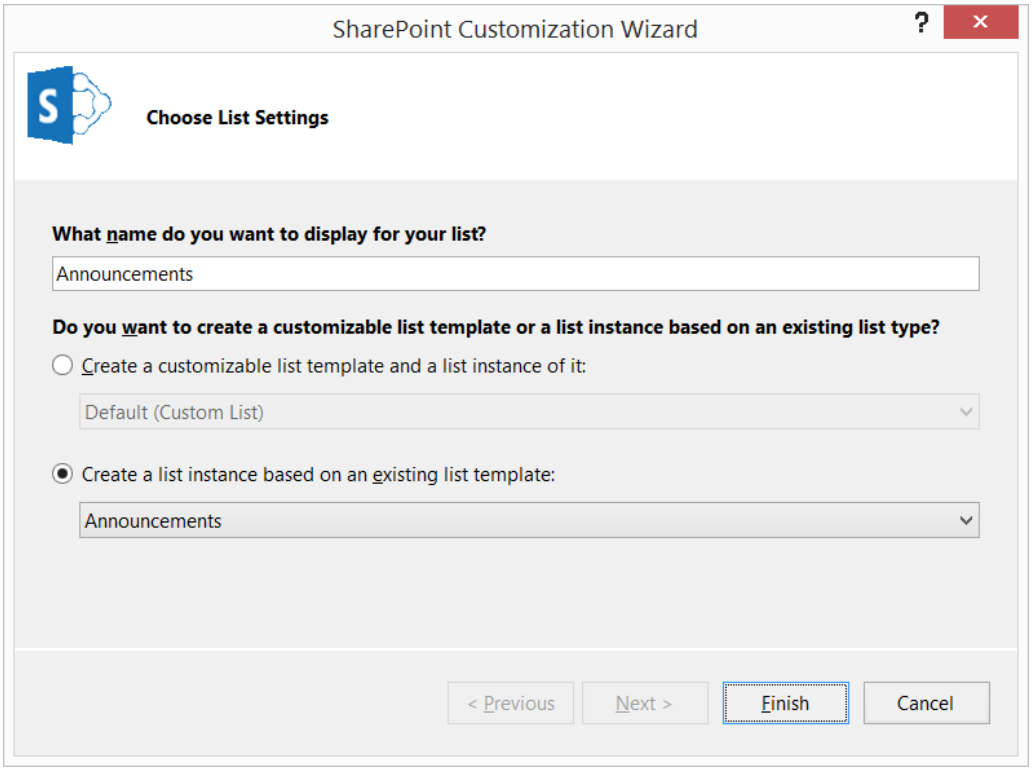
b) In the **Add New Item** dialog, select the **List** template found in the **Visual C# Items / Office / SharePoint** category. Enter a name of **Announcements** and click **Add**.



c) In the **SharePoint Customization Wizard** dialog, use the following values to complete the form and click **Finish**:

i) **What name do you want to display for your list?** Announcements

ii) **Do you want to create a customizable list template or a list instance based on an existing list type:** Create a list instance based on an existing list template: **Announcements**

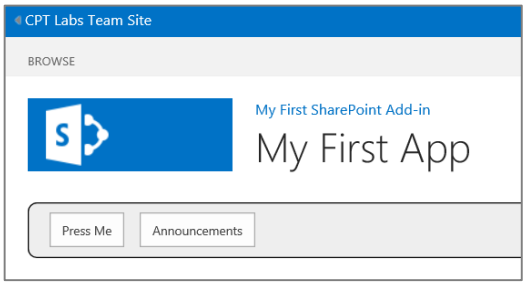


14. Save all changes: **File → Save All**.
15. Open the file default.aspx and add the following HTML link to after the ta.

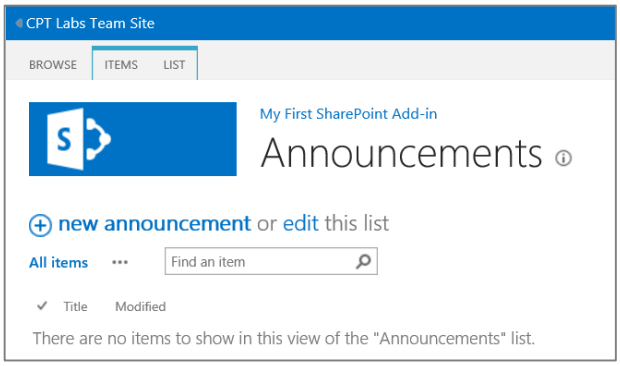
```
<div id="toolbar">
    <input type="button" id="cmdPressMe" value="Press Me" />
    <input
      type="button"
      value="Announcements"
      onclick="JavaScript: location.href = '../Lists/Announcements'" />
  </div>
```

Build and Test the Project

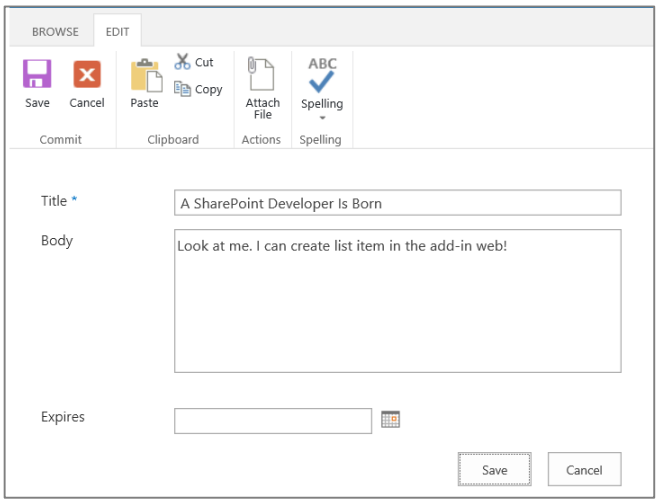
16. Build and test your application by pressing **[F5]** or **Debug → Start Debugging**.
17. Once the solution has been deployed, Internet Explorer will launch and navigate to the add-in start page.
18. Notice there is nothing in the user interface of the add-in to link to the Announcements list except for the link button that you added.



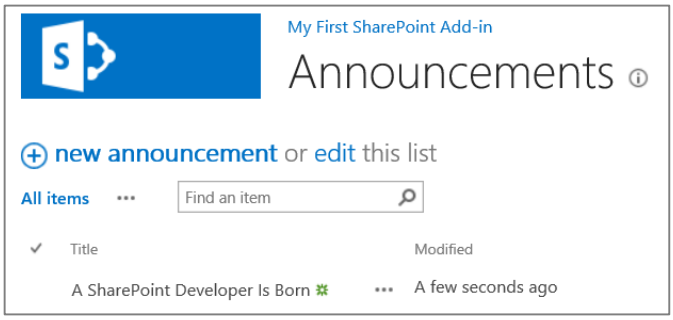
19. Click the **Announcements** link button to navigate to the **Announcements** list.



20. Click the **new announcement** link button to display a form for creating a new announcement. Create new announcement item using test date like that shown in the following screenshot. Click the **Save** button to create the new item.



21. Once you save the item, you should be able to see it listed in the default view for the **Announcements** list.



22. Close the browser to stop the debugger and return to Visual Studio.

In this exercise you created a simple SharePoint-hosted add-in and made some basic customizations to it. Later modules and labs will build upon this foundation (e.g. working with the CSOM and REST API's, customizing the user interface, and creating robust client-side code with additional permissions).

Exercise 3: Creating the Hello World App Part

In this exercise you will create a new SharePoint-hosted add-in with a simple app part.

1. Create a new SharePoint App project in Visual Studio 2015:
   a) In Visual Studio select **File → New → Project**.

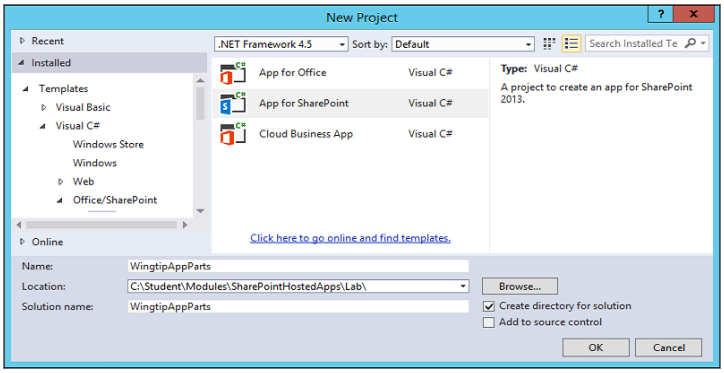
b) In the **New Project** dialog:

i) Find the **App for SharePoint** template under the **Templates → Visual C# → Office / SharePoint → Apps** section.

ii) **Name**: WingtipAppParts

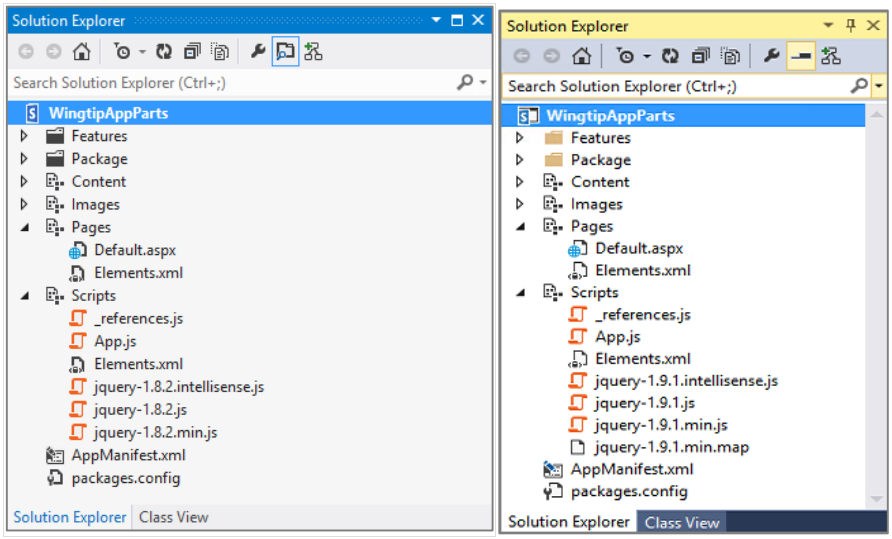
iii) **Location:** C:\Student\Modules\SharePointHostedApps\Lab

iv) Click **OK.**



c) In the **New app for SharePoint** dialog, choose the option to create a SharePoint-hosted app and make sure the URL is the one for your SharePoint Developer site.

d) Once the new project has been created, examine its structure and the source files inside.



2. Open the **AppManifest.xml** file by double clicking on it. Change the **Title** to Wingtip App Parts. Save and close the **AppManifest.xml** file.

3. Open the **App.js** file in the **Scripts** folder. Delete all the contents from **App.js** leaving it as an empty file for now. Save your changes to **App.js** and close the file.

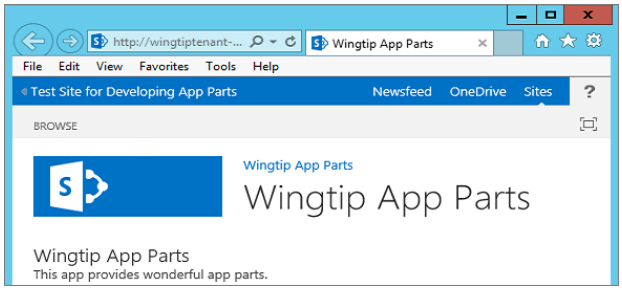
4. Open **Default.aspx** in Code View. Replace the content inside the two placeholders named **PlaceHolderPageTitleInTitleArea** and **PlaceHolderMain** with the following HTML code.

```
<asp:Content ContentPlaceHolderID="PlaceHolderPageTitleInTitleArea" runat="server">
    Wingtip App Parts
</asp:Content>

<asp:Content ContentPlaceHolderID="PlaceHolderMain" runat="server">

    <h2>Wingtip App Parts</h2>
    <div>This app provides wonderful app parts.</div>

</asp:Content>
```

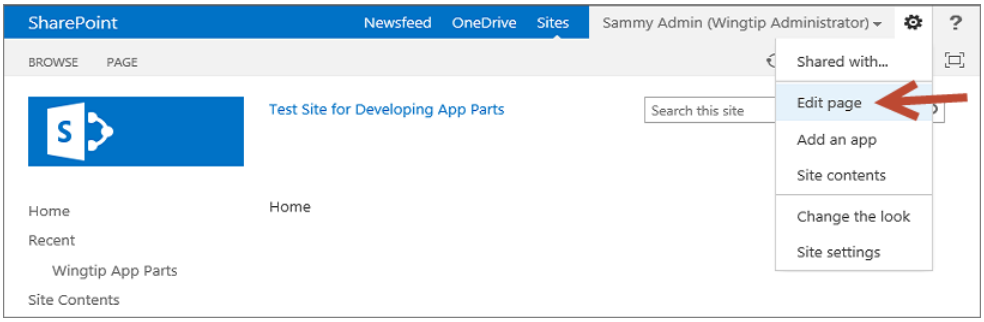
5. Save your changes to **Default.aspx** and then close this file.

Remember that the purpose of this lab exercise is to create app parts. However, the app still requires a start page even if the start page doesn't really provide any real functionality. However, the start page is helpful for testing because it provides a link back to the host web where you will be testing and debugging your app parts. When you launch a debugging session, you should become familiar with the process of redirecting from the app start page back to the host web so you can create an instance of your app parts for testing and debugging purposes.

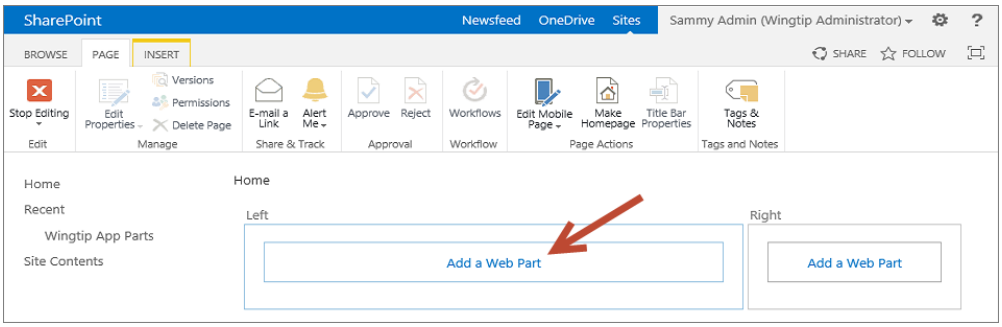
6. Test your work by pressing the **{F5}** key to launch a debugging session. When the app starts, you should see the start page appear as the one shown in the following screenshot. Leave this start page open for the next step.



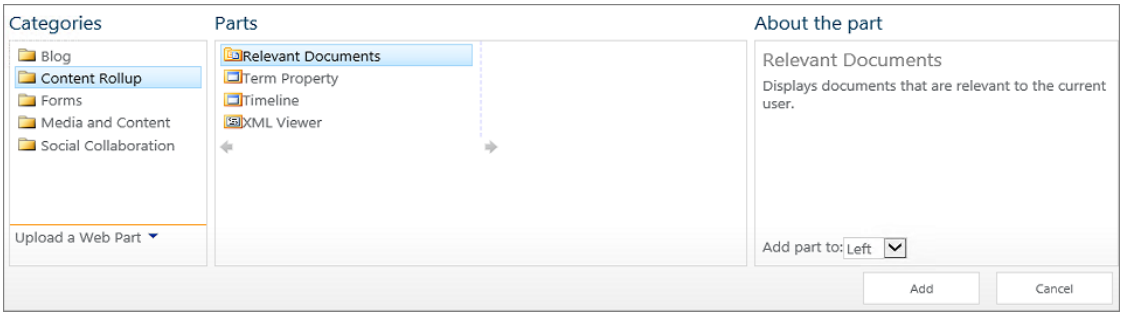
7. Navigate to the host web and see what web parts are available out of the box.
   a) Click the link in the top left corner of the start page to navigate back to the host web. This should redirect you to the home page of the Blank site at **http://apppart.wingtip.com**.
   b) Drop down the **Site Actions** menu and select the **Edit page** command to move the page into Edit Mode.



   c) Once the page is in Edit Mode, you should see two web part zones. Click the **Add a Web Part** link in the left web part zone. This action will display the web part catalog.

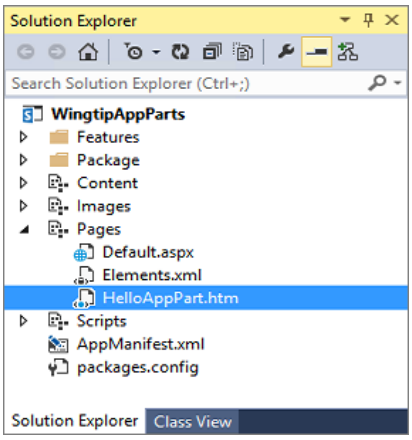


   d) While you do not need to add a web part to the page in this step, your objective is simply to see what web parts are available. In just a bit, your will see your custom app parts available in this web part catalog.

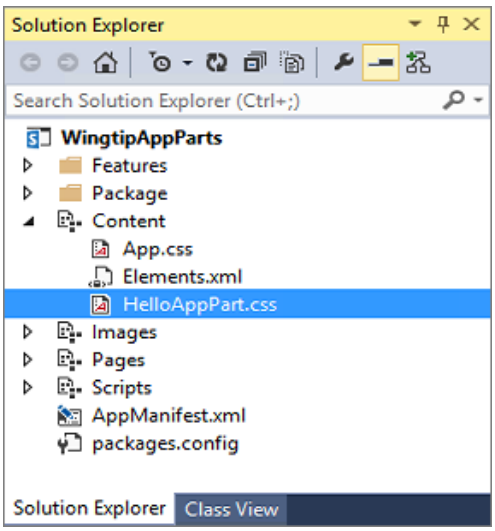


e) Once you have looked through the available set of out-of-the-box web parts, close the browser to end the debugging session.

8. Return to Visual Studio.
9. Create an HTML page for an app part.
   a) Add a new HTML page to the **Pages** folder named **HelloAppPart.htm**.
      i) In the **WingtipAppParts** project right click on the **Pages** folder and select **Add → New Item…**
      ii) In the **Add New Item** dialog box, Select **Visual C# → Web** from the categories on the left side then select **HTML Page** from the templates in the middle and give this page the name: **HelloAppPart.htm**



   b) Add a new CSS file to the **Content** folder named **HelloAppPart.css**.
      i) In the **WingtipAppParts** project right click on the **Content** folder and select **Add → New Item…**
      ii) In the **Add New Item** dialog box, Select **Visual C# → Web** from the categories on the left side then select **Style Sheet** from the templates in the middle and give this page the name: **HelloAppPart.css**



c) Modify the contents of **HelloAppPart.css** to look like the following CSS listing.

```
body {
  background-color: yellow;
}

h4 {
  color: blue;
  border-bottom: 1px solid blue;
}
```

d) Save and close **HelloAppPart.css**.

e) Open **HelloAppPart.htm** and modify the HTML contents to look like the following HTML listing. Be sure to include a link to the CSS file named **HelloAppPart.css**.

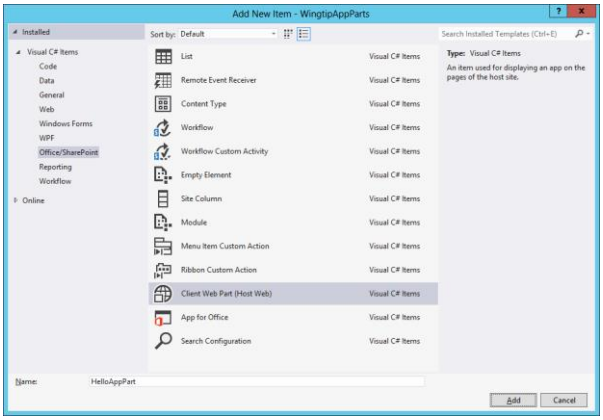
```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>
  <meta charset="utf-8" />
  <title></title>
  <link href="../Content/HelloAppPart.css" rel="stylesheet" />
</head>

<body>

  <h4>Hello App Part Content</h4>
  <div>This content lives in the app web</div>

</body>
</html>
```

f) Save and close **HelloAppPart.htm**.

10. Create a new app part which will use the page **HelloAppPart.htm** to display its contents.

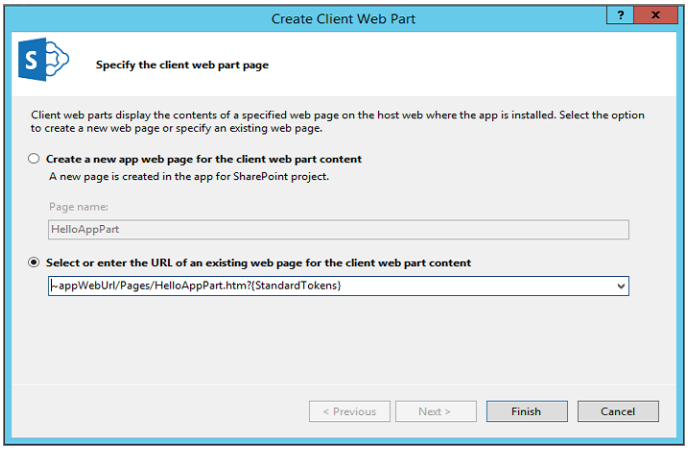
a) In the Solution Explorer, right-click on the **WingtipAppParts** project and select the **Add New Item** command.

b) In the **Add New Item** dialog, select the **Client Web Part (Host Web)** project item template and give it the name **HelloAppPart**.



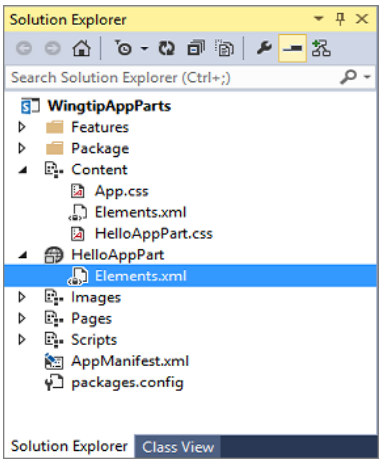
c) Click the **Add** button at the bottom right of the **Add New Item** dialog to add the new Client Web Part project item. When you click the Add button, you should then see the **Create Client Web Part** dialog.

d) In the **Create Client Web Part** dialog, select the option **Select or enter a URL for an existing web page**. Then use the drop down list to select the **HelloAppPart.htm** page in the **Pages** folder.



e) Click the **Finish** button in the **Create Client Web Part** dialog to complete the process of adding the new Client Web Part project item.

f) Once the Client Web Part project item has been created, you can see that Visual Studio has created a folder for it in the project. This folder contains a file named **elements.xml**.



g) Modify the **elements.xml** file for the new **HelloAppPart** Client Web Part to match the XML in the following code listing.

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <ClientWebPart
    Name="HelloAppPart"
    Title="The Hello App Part"
    Description="A simple little app part"
    DefaultWidth="600" DefaultHeight="200">

    <Content
      Type="html"
      Src="~appWebUrl/Pages/HelloAppPart.htm?{StandardTokens}" />

    <Properties>
    </Properties>

  </ClientWebPart>
</Elements>
```

h) Save and close the **elements.xml** file.

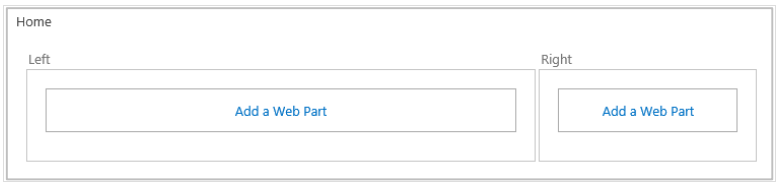
11. Test your work by adding the **HelloAppPart** app part to a web part page in the host web.

a) Press **{F5}** to begin a debugging session.

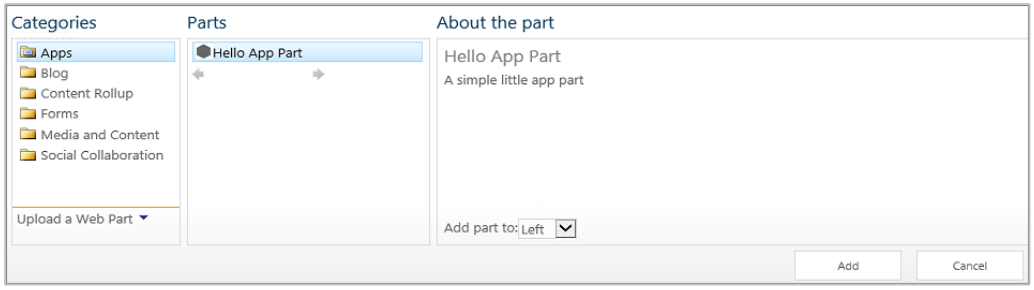
b) When you see the app's start page, click the link to redirect to the home page of the host web. (Reminder: this link is in the top left corner of the page)

c) Use the **Edit page** menu from the **Site Actions** menu to move the page into Edit Mode.

d) Once you are in Edit Mode, click the **Add a Web Part** link in the left web part zone to display the web part catalog.



e) Locate and select the app part with a title of **The Hello App Part** in the **Apps** category folder. Click the **Add** button on the bottom right-hand side of the web part catalog to add the app part to the home page of the host web.



f) In the **Ribbon Bar Page** Tab click **Stop Editing**. Now click the **Browse** Tab in the Ribbon to see your completed page with the app part.

g) After you have added the app part, you should be able to see it on the home page of the host web.

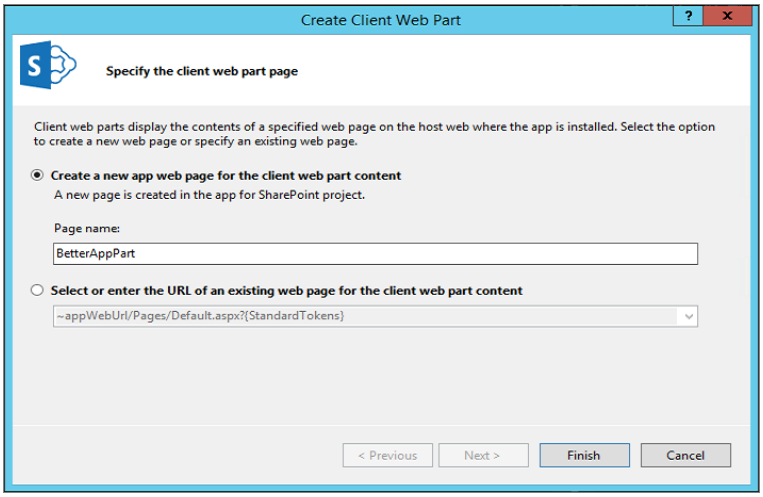
h) Close the browser to end the debugging session and then return to Visual Studio.

Now you have created and tested a simple app part based on an HTML page. Next, you will create a more complicated app part with custom app part properties which is implemented with an ASPX file instead of a simple HTML file.

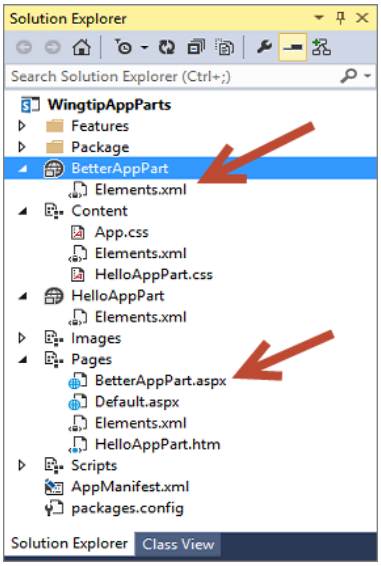
Exercise 4: Creating an App Part with Custom Properties

In this exercise you will create and test an app part with custom properties.

1. Continue working in the same **WingtipAppParts** project you created in the previous exercise.
2. Create a new app part named **BetterAppPart**,
   a) Right-click on the **WingtipAppParts** project and select the **Add New Item** command.
   b) In the **Add New Item** dialog, select the **Client Web Part (Host Web)** project item template (Located in the **Visual C# Items → Office/SharePoint** category) and give it a name of **BetterAppPart**. Click the **Add** button, you should then see the **Create Client Web Part** dialog.
   c) In the **Create Client Web Part** dialog, accept the default settings and click **Finish**.



d) Once the Client Web Part has been added, inspect what files have been added to the project. You should see that Visual Studio created a folder named **BetterAppPart** for the project item which contains an elements.xml file which defines the Client Web Part. In addition, an aspx page named **BetterAppPart** has been added to the **Pages** folder.



3. Open the **elements.xml** file in the **BetterAppPart** and modify its content to look like this.

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

  <ClientWebPart
    Name="BetterAppPart"
    Title="Better App Part"
    Description="A really nice app part"
    DefaultWidth="600"
    DefaultHeight="200">

    <Content
      Type="html"
      Src="~appWebUrl/Pages/BetterAppPart.aspx?{StandardTokens}" />

    <Properties>
    </Properties>

  </ClientWebPart>
</Elements>
```

4. Save and close the **elements.xml** file.

5. Open **BetterAppPart.aspx** in Code View. Do not make any modifications to the **Page** directive, the **Register** directives or the **WebPartPages:AllowFraming** control at the top of the page. However, modify the HTML content below in the page to look like the code following listing. (i.e. this means you will remove all the <script> tags and associated script content (except for the jquery script tag) from the <head> section of the page in addition to adding content to the <body> section)

```
<%@ Page Language="C#" Inherits="Microsoft.SharePoint.WebPartPages.WebPartPage, BLAH BLAH BLAH %>

<%@ Register TagPrefix="SharePoint" Namespace="Microsoft.SharePoint.WebControls" BLAH BLAH BLAH %>
<%@ Register TagPrefix="Utilities" Namespace="Microsoft.SharePoint.Utilities" BLAH BLAH BLAH %>
<%@ Register TagPrefix="WebPartPages" Namespace="Microsoft.SharePoint.WebPartPages" BLAH BLAH BLAH %>

<WebPartPages:AllowFraming ID="AllowFraming" runat="server" />

<html>
<head>
  <title></title>

  <script type="text/javascript" src="../Scripts/jquery-1.9.1.min.js"></script>

</head>
<body>

  <h4>Better App Part Content</h4>
  <div id="results">default contents</div>

</body>
</html>
```

   a) Save your changes to the **BetterAppPart.aspx** file. (Note: Keep this file open we will need it later)

6. Add some JavaScript code for the app part.

   a) In Windows Explorer, look inside the folder at **C:\Student\Modules\SharePointHostedApps\Lab\StarterFiles** and locate the file named **wingtip.utilities.js**. Add this file into the **WingtipAppParts** project in the **Scripts** folder. (Note: you can accomplish this by dragging the file from the **Starter Files** source folder in File Explorer into the **Scripts** destination folder in the Solution Explorer in Visual Studio)

   b) Inspect what's inside of **wingtip.utilities.js**. As you can see, it is a JavaScript Module named **Wingtip.Utilities** that is very similar to the one you created in the JavaScript programming lab.

```
'use strict';

var Wingtip = window.Wingtip || {};

Wingtip.Utilities = function () {

  var getQueryStringParameter = function (param) {
    var querystring = document.URL.split("?")[1];
    if (querystring) {
      var params = querystring.split("&");
      for (var index = 0; (index < params.length) ; index++) {
        var current = params[index].split("=");
        if (param.toUpperCase() === current[0].toUpperCase()) {
          return decodeURIComponent(current[1]);
        }
      }
    }
  }


  return {
    getQueryStringParameter: getQueryStringParameter,
  };

}();
```

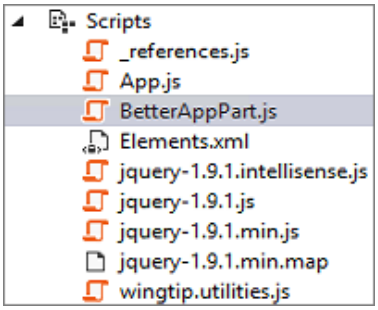
   c) Close **wingtip.utilities.js**.

   d) Add a new JavaScript file into the **Scripts** folder named **BetterAppPart.js**.

      i) Right-click on the **Scripts** folder in the **WingtipAppParts** project in **Solution Explorer** and select the **Add New Item** command.

   e) In the **Add New Item** dialog, select the **JavaScript File** template (Located in the **Visual C# Items → Web** category) and give it a name of **BetterAppPart.js**. Click the **Add** button, you should then see the **Create New JavaScript File** dialog.

f) In the **Create New JavaScript File** dialog, accept the default settings and click **Finish**



g) Add the following JavaScript code to **BetterAppPart.js**.

```
$(function () {

  $("#results").text("My dynamic content");

  $("body").css({
    "border": "2px solid #CCC",
    "padding": "8px"
  });

  $(":header").css({"border-bottom": "1px solid black"});

});
```

h) Save and close the **BetterAppPart.js** file.

7. Open **BetterAppPart.aspx** and add the following script links into the **head** section of the page:
(Note: you can do this quickly by dragging the JavaScript files from the Solution Explorer into the correct location in the BetterAppPart.aspx page)

a) The jQuery library (verify that this is already there)
(Note: the version number on this library may differ from the code below as it is frequently updated)

b) Wingtip.utilities.js

c) BetterAppPart.js

```
<head>
  <title></title>
    <script type="text/javascript" src="../Scripts/jquery-1.9.1.min.js"></script>
    <script src="../Scripts/wingtip.utilities.js"></script>
    <script src="../Scripts/BetterAppPart.js"></script>

</head>
```

d) Save and close the **BetterAppPart.aspx** file.

8. Test your work by adding the **BetterAppPart** app part to a web part page in the host web.

a) Press **{F5}** to begin a debugging session.

b) When you see the app's start page, click the link to redirect to the home page of the host web.

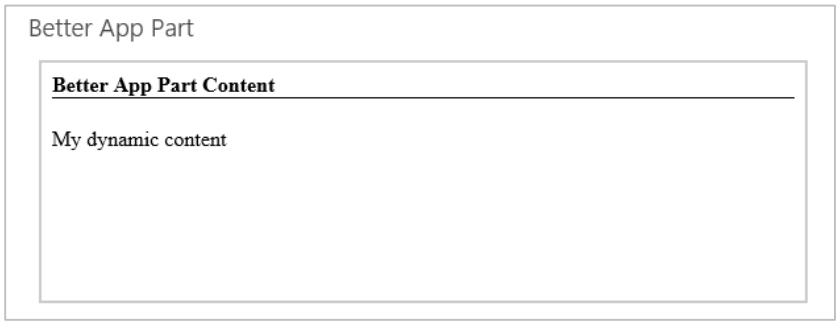
c) Use the **Edit page** menu from the **Site Actions** menu to move the page into Edit Mode.

d) Once you are in Edit Mode, click the **Add a Web Part** link in the left web part zone to display the web part catalog.

e) Locate and select the app part with a title of **Better App Part** in the **Apps** category folder. Click the **Add** button on the bottom right-hand side of the web part catalog to add the app part to the home page of the host web.

f) In the **Ribbon Bar Page** Tab click **Stop Editing**. Now click the **Browse** Tab in the Ribbon to see your completed page with the app part.

g) Once the app part is displayed, you should be able to verify that the JavaScript code executed property to add the message "My dynamic content" and to add a bottom border on the heading **Better App Part Content**.



h) Close the browser window to end the debugging session and return to Visual Studio.

9. Add two app part properties.

a) Open the **elements.xml** file for the **BetterAppPart** app part. Add the two following property definitions.

```
<Properties>

      <Property
          Name="BackgroundColor"
          WebDisplayName="Add Background Color"
          Type="boolean"
          DefaultValue="false"
          WebCategory="Custom Wingtip Properties"
          RequiresDesignerPermission="true" >
      </Property>

      <Property
          Name="HeaderColor"
          WebDisplayName="Header Color"
          Type="enum"
          DefaultValue="Black"
          WebCategory="Custom Wingtip Properties"
          RequiresDesignerPermission="true" >
        <EnumItems>
          <EnumItem WebDisplayName="Black" Value="Black"/>
          <EnumItem WebDisplayName="Blue" Value="Blue"/>
          <EnumItem WebDisplayName="Green" Value="Green"/>
        </EnumItems>
      </Property>

</Properties>
```

b) Inspect the Content element in **elements.xml**. Currently the Src attribute is defined as an URL which has a query string defined using only the dynamic token named **{StandardTokens}**.

```
<Content
  Type="html"
  Src="~appWebUrl/Pages/BetterAppPart.aspx?{StandardTokens}" />
```

c) Modify the query string in the **elements.xml** file as shown here to pass the custom property values to **BetterAppPart.aspx**.

```
BetterAppPart.aspx?BackgroundColor=_BackgroundColor_&amp;HeaderColor=_HeaderColor_&amp;{StandardTokens}
```

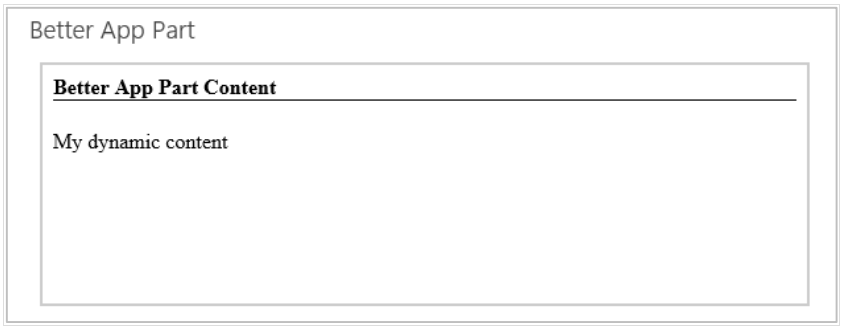
d) Save and close the **elements.xml** file.

e) Return to **BetterAppPart.js** and add some code to read the two property values from the query string.

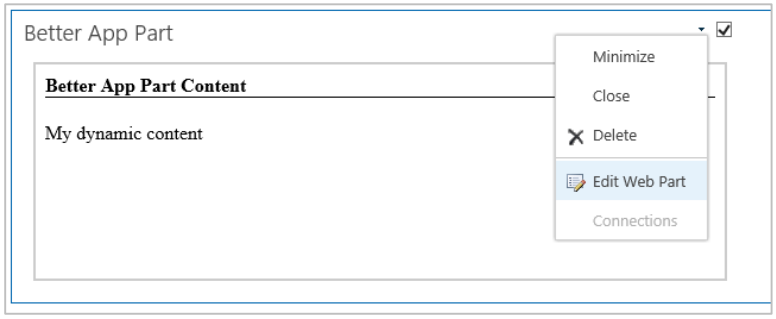
```
$(function () {

  $("#results").text("My dynamic content");

  $("body").css({
    "border": "2px solid #CCC",
    "padding": "8px"
  });

  $(":header").css({"border-bottom": "1px solid black"});

  var BackgroundColor = Wingtip.Utilities.getQueryStringParameter("BackgroundColor");
```

```
  if (BackgroundColor === "true") {
    $("body").css({ "background-color": "Yellow" });
  }

  var HeaderColor = Wingtip.Utilities.getQueryStringParameter("HeaderColor");
  if (HeaderColor) {
    $(":header").css({ "color": HeaderColor });
    $(":header").css({ "border-bottom": "1px solid " + HeaderColor });
  }

});
```

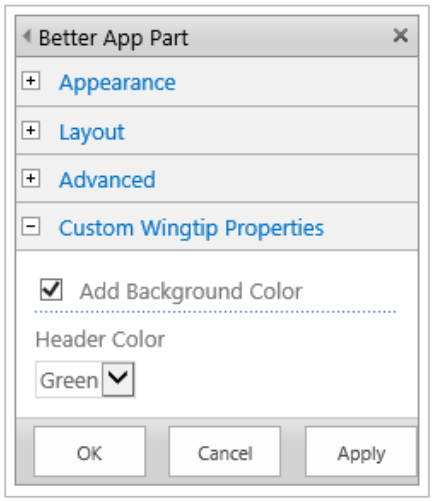
10. Save and close the **BetterAppPart.js** file.
11. Test your work.
    a) Using the Visual Studio **Build** Menu select **Rebuild WingtipAppParts** to ensure the updated code will be deployed.
    b) Press **{F5}** to begin a debugging session.
    c) When you see the app's start page, click the link to redirect to the home page of the host web.
    d) Use the **Edit page** menu from the **Site Actions** menu to move the page into Edit Mode.
    e) Once you are in Edit Mode, click the **Add a Web Part** link in the left web part zone to display the web part catalog.
    f) Locate and select the app part with a title of **Better App Part** in the **Apps** category folder. Click the **Add** button on the bottom right-hand side of the web part catalog to add the app part to the home page of the host web.
    g) Once the app part is displayed, you should be able to verify that the JavaScript code executed property to add the message "My dynamic content" and to add a bottom border on the heading **Better App Part Content**.



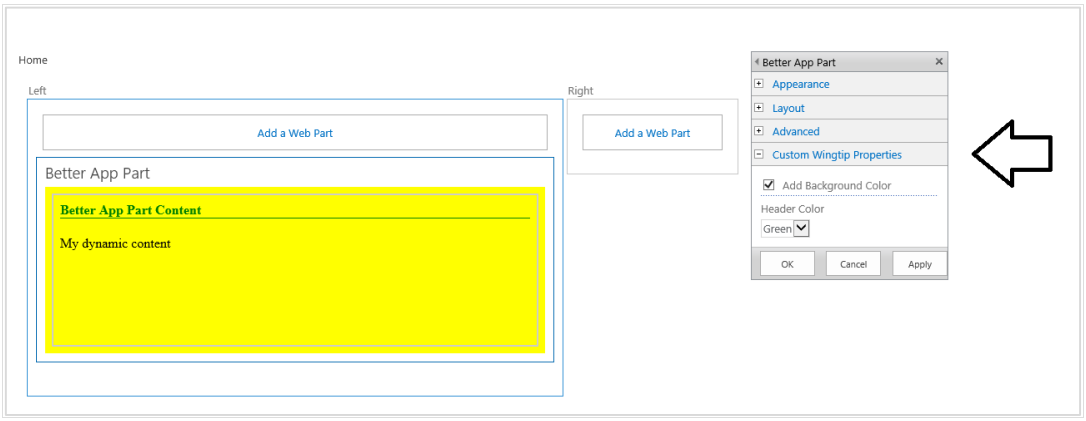
    h) As you are still in the Page Edit mode you can use the drop down app part menu in the top-right corner of the app part title bar to select the **Edit Web Part** menu. This will display the editor parts that make it possible for the user to modify app part properties.



    i) In the editor part for the **Better App Part**, locate and expand the **Custom Wingtip Properties** section.



j) Enable the option to **Add Background Color**. Change the **Header Color** property to Green and then click the **Apply** button. You should see these changes affect the display the app part.



k) When you are done with your testing, close the browser window to end the debugging session.

You have now completed this lab where you have created and tested an app part with custom properties.