Developing SharePoint-hosted Add-ins

Lab Time: 40 minutes

Lab Folder: **C:\Student\Modules\03_SharePointHostedAddins\Lab**

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

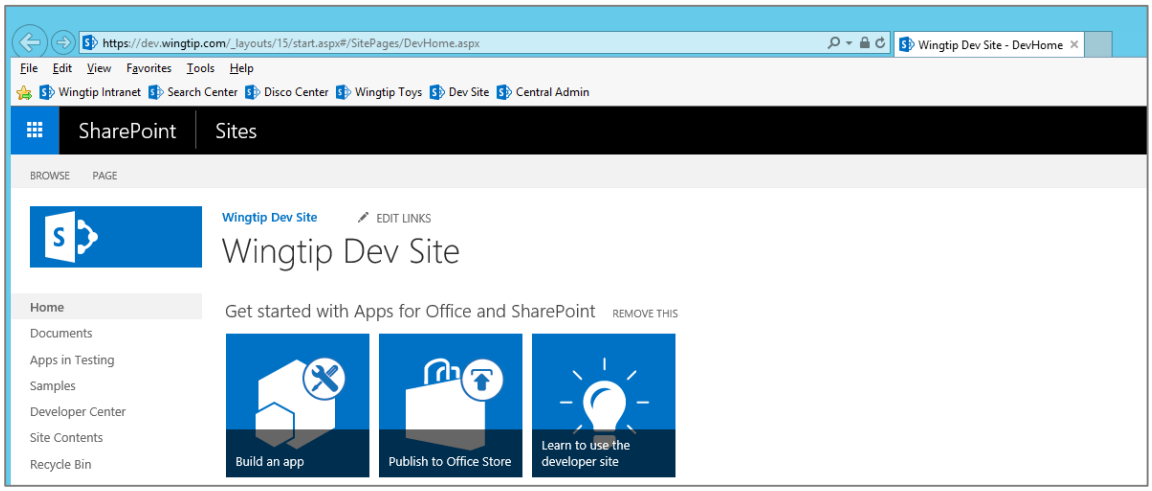
Exercise 1: Creating a New Developer Site for Testing

In this exercise you will prepare your development environment by creating a new Developer site at https://dev.wingtip.com. You will run a PowerShell script to create a new site collection using the Developer Site template which is often used during SharePoint Add-in development. If you VM already has an existing site at the URL of https://dev.wingtip.com, running the script will delete the existing site collection and create a fresh new Developer Site.

1. Create a new site collection for this lab:
   a) Ensure you are logged into the **WingtipServer** server as **WINGTIP\Administrator**.
   b) Run a PowerShell script, found in the root lab folder for this module:
      i) Right-click **SetupLab.ps1** and select **Run with PowerShell**. This file can be found in the files associated with this lab: (Note: In order to run PowerShell Scripts in this environment, you may be prompted for an Execution Policy Change. If prompted, type **Y** and press **Enter.**)

```
C:\Student\Modules\03_SharePointHostedAddins\Lab\SetupLab.ps1
```

   c) When the script completes, it will launch a new browser and navigate to the new Developer site at **https://dev.wingtip.com**.



   d) Close the PowerShell console window.

You have now completed the first step by creating a new test site for SharePoint add-in development.

Exercise 2: Creating and Debugging a SharePoint-Hosted App

In this exercise you will create a new SharePoint-Hosted App.

1. Create a new project in Visual Studio 2015:
   a) Launch **Visual Studio 2015** as administrator:
      i) Windows Keyboard Key → Right click on the Visual Studio 2013 tile and select **Run as administrator**.

b) In Visual Studio select **File → New → Project**.

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

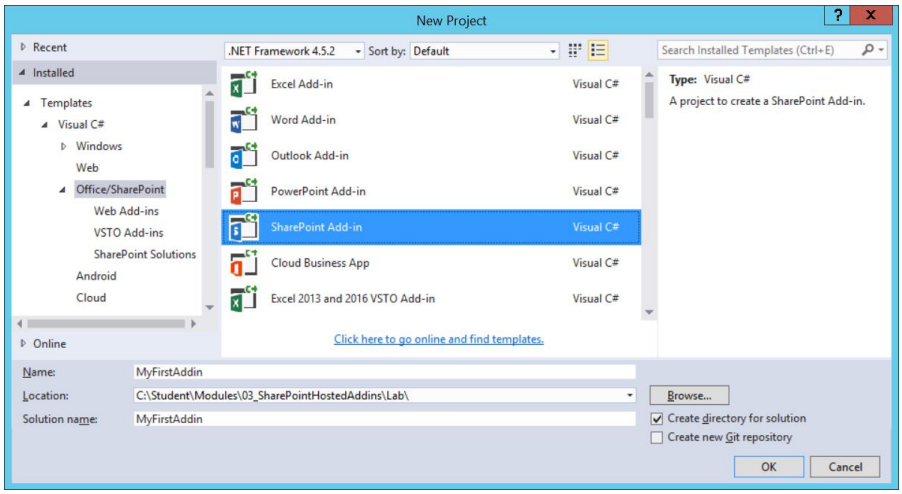
i) Find the **SharePoint Add-in** template under the **Templates → Visual C# → Office/SharePoint** section.

ii) **Name**: MyFirstAddin

iii) **Location:** C:\Student\Modules\03_SharePointHostedAddins\Lab\

iv) **Solution name:** MyFirstAddin

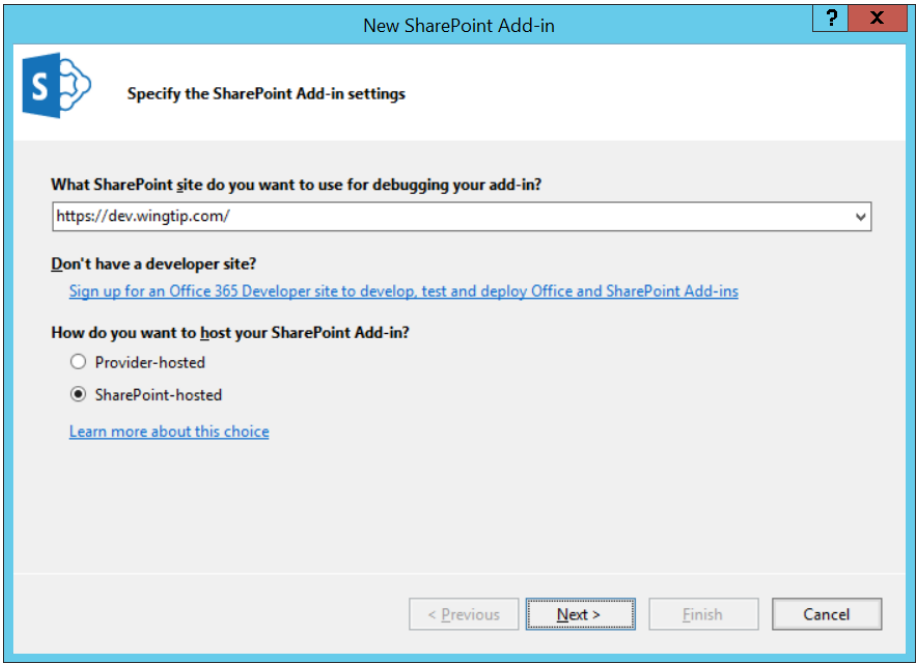
v) Click **OK.**



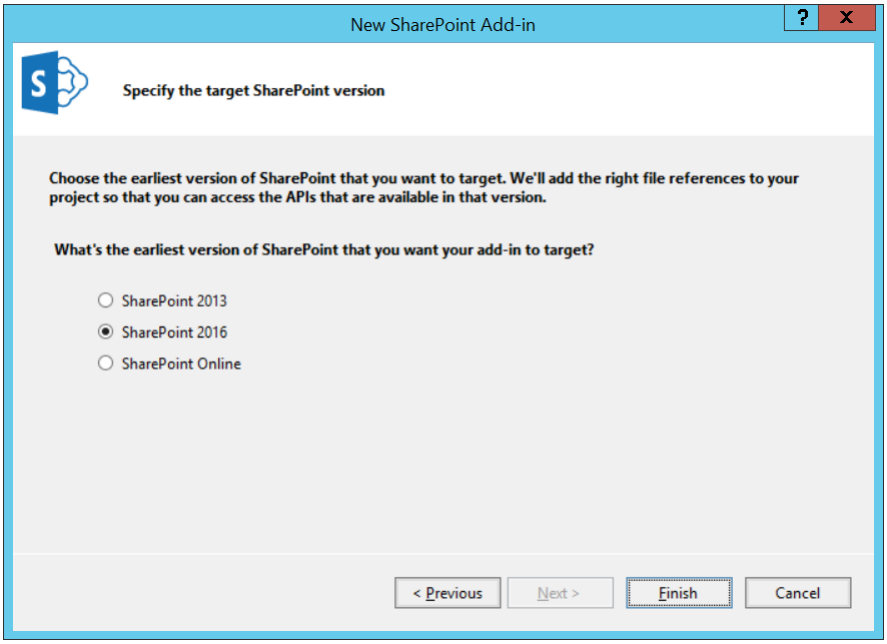
d) In the **New SharePoint Add-in** wizard, use the following values to complete the wizard and click **Finish**.

i) **What site do you want to use for debugging?** https://dev.wingtip.com

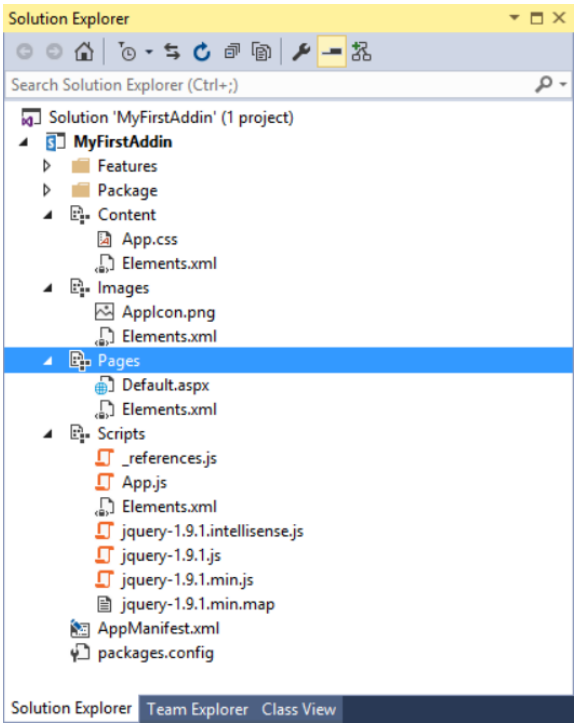
ii) **How do you want to host your app for SharePoint?** SharePoint-hosted



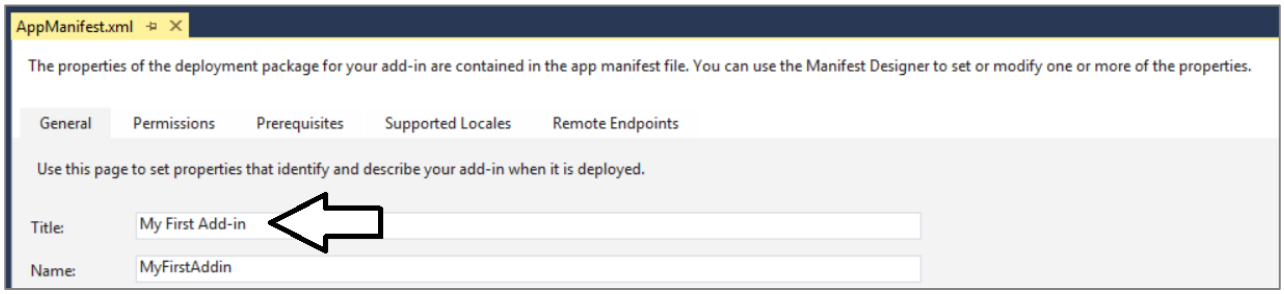
e) Specify **SharePoint 2016** as the add-in target platform and click **Next**.



2. Examine the default project setup for a SharePoint-Hosted app:
   a) Like a traditional SharePoint solution-based project you have a **Features** and **Packages** node. These work the same way they do in a solution-based project.
   b) **Content**, **Images** & **Pages**: These are actually SharePoint Project Items (SPI) that are Modules and will provision their contents to the respective folders in the app web that will be generated upon installing the app.
      i) **Content → App.css:** this is the Cascading Style Sheet used for the app
      ii) **Images → AppIcon.png:** this is the default image used for the app.
      iii) **Pages → Default.aspx:** this is the default homepage for the app.
   c) **Scripts:** This is also a SPI Module that provisions its contents to the site. Because SharePoint-Hosted apps cannot use any server-side code, all business logic is handled in the client using JavaScript.
      i) **Scripts → _references.js**: this file is not provisioned… it is a little trick file that Visual Studio uses to provide IntelliSense when writing JavaScript code. If you look inside this file you will notice triple commented references to JavaScript files which Visual Studio 2013 uses for IntelliSense.
      ii) **Scripts → App.js**: this file is the default logic file for your app. It is referenced from the Default.aspx file. You do not have to use this file for your business logic… you can replace its contents or create your own.
      iii) **Scripts → jquery[..].js**: because all logic is implemented in the client, the popular jQuery library is added to the project. The minified and non-minified versions of the library are include as well as the Visual Studio IntelliSense file (*.vsdoc.js). You are free to replace this jQuery library with a more recent release if you like.
   d) **AppManifest.xml**: every app must have this file. It tells SharePoint the basic information it needs about the app 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.



3. Update the AppManifest.xml file.
   a) Using the **Solution Explorer** tool window, right-click the **AppManifest.xml** file and select **Open**.
   b) Update the **Title** to **My First Add-in**.



4. Replace the generic app icon with a custom app icon.
   a) Using Windows Explorer, locate the icon file at the following location inside your Student folder.

```
C:\Student\Modules\03_SharePointHostedAddins\Lab\StarterFiles\AppIcon.png
```

   b) Use this **AppIcon.png** file to replace the generic **AppIcon.png** file located at the root of the **Images** folder of the **MyFirstAddin** project.

5. Update the app home page named **Default.,aspx**.
   a) Using the **Solution Explorer** tool window, right-click the **Pages/Default.aspx** file and select **Open**.
   b) Locate the three **<asp:Content>** tags on the page with the IDs of **PlaceHolderAdditionalPageHead**, **PlaceHolderPageTitleInTitleArea** and **PlaceHolderMain**.
   c) You do not need to change anything inside the **PlaceHolderAdditionalPageHead** content control but you should observe that it already contains script links to the jQuery library and the **App.js** file as well as a link to the app's CCS file named **App.css**.
   d) Update the contents of the **PlaceHolderPageTitleInTitleArea** content control to match the following listing.

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

e) Update the contents of the **PlaceHolderMain** content control to match the following listing.

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

    <div id="toolbar">
        <input id="cmdClickMe" type="button" value="Click Me" />
    </div>

    <div id="content_box" />

</asp:Content>
```

f) Save and close **Default.aspx**.

6. Update the app CSS file.
   a) Using the **Solution Explorer** tool window, right-click the **Content/App.css** file and select **Open**.
   b) Delete the contents of app.css and replace it with the following CSS code.

```
#toolbar{
    background-color: #DDD;
    padding: 8px;
    border: 1px solid #AAA;
    border-top-left-radius: 8px;
    border-top-right-radius: 8px;
}

#toolbar input {
    border-radius: 4px;
}

#content_box{
    background-color: #FFC;
    border: 1px solid #AAA;
    padding: 12px;
    min-height: 240px;
    border-bottom-left-radius: 8px;
    border-bottom-right-radius: 8px;
}
```

   c) Save and close **app.css**.
7. Update the app script file:
   a) Using the **Solution Explorer** tool window, right-click the **Scripts/App.js** file and select **Open**.
   b) Delete the contents of app.js and replace with the following JavaScript code listing.

```
'use strict';

$(function () {
  $("#cmdClickMe").click(onClickMe);
});

function onClickMe() {

  $("#content_box")
    .text("Hello SharePoint Add-ins")
    .css({"font-size": "32px", "color": "blue" })
}
```

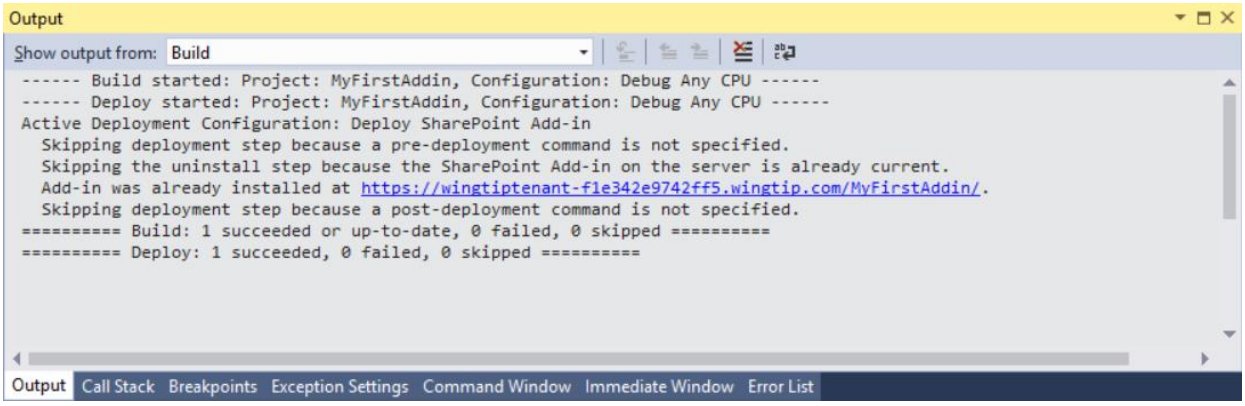
8. Save all changes: **File → Save All**.

This is the simplest SharePoint-Hosted app you could create with some business logic in it. Let's just see how things turn out before we add some more stuff to it.

Build and Test the Project

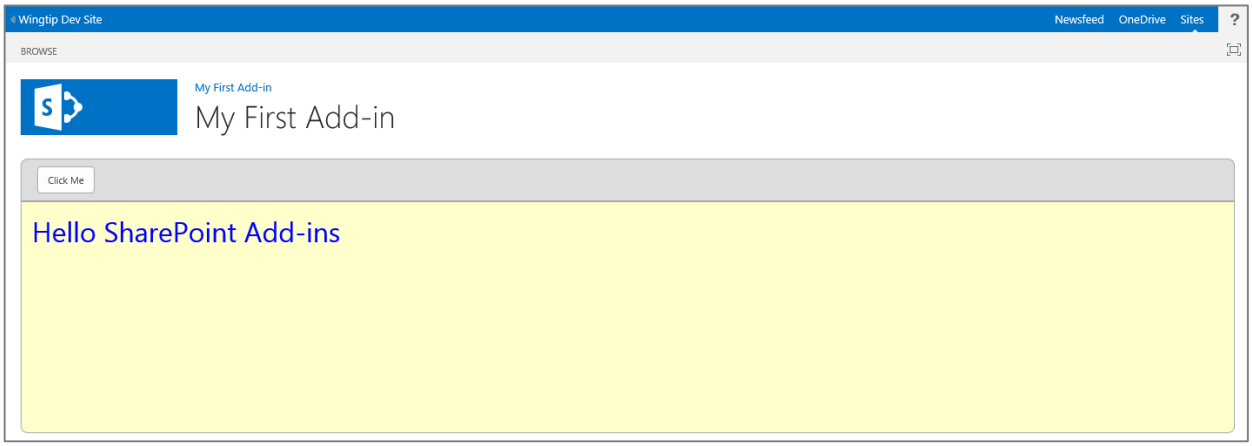
9. Build and test your application by pressing **[F5]** or **Debug → Start Debugging**.

10. The installation process for an app will take a moment to complete. If you watch the lower-left corner of Visual Studio, it will tell you what it is currently doing. If you want more information, click the **Output** tab at the bottom of Visual Studio to see a log of what is going on (if the **Output** tab isn't present, select the window from the menu in Visual Studio 2015: **View → Output**):

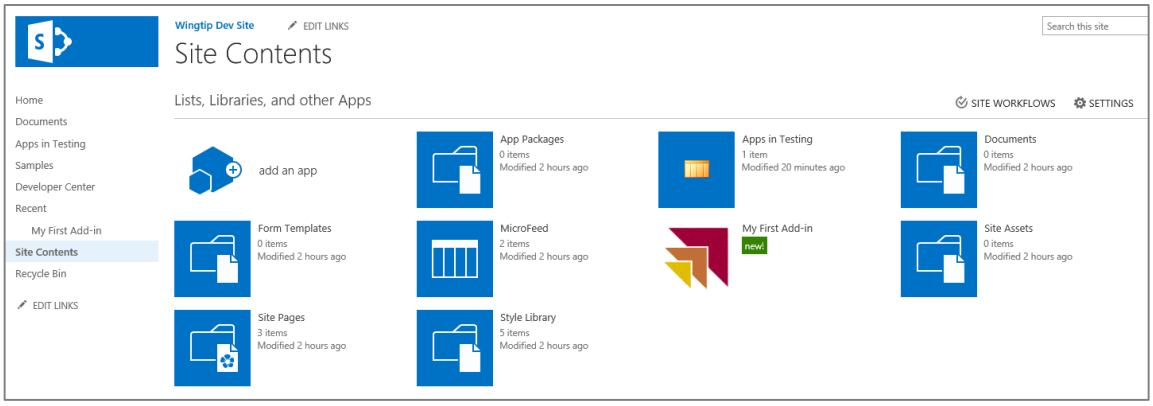


a) What you see in the screenshot is the app was compiled first and then the installation process started. Visual Studio will write a message to the Output window every second while the app is being installed.

11. Once the solution has been deployed, Internet Explorer will launch and navigate to the app's **default.aspx** page.
12. When the page loads, click the **Push me!** button to see your text get written to the page:



13. At the top of the page click the **Wingtip Dev** link to navigate back to the site which is the host web.
14. On the Sites Quick Launch navigation (on the left side of the screen) click on **Site Contents**.
15. On the Site Contents page notice the icon for the app we just deployed **My First Add-in**.



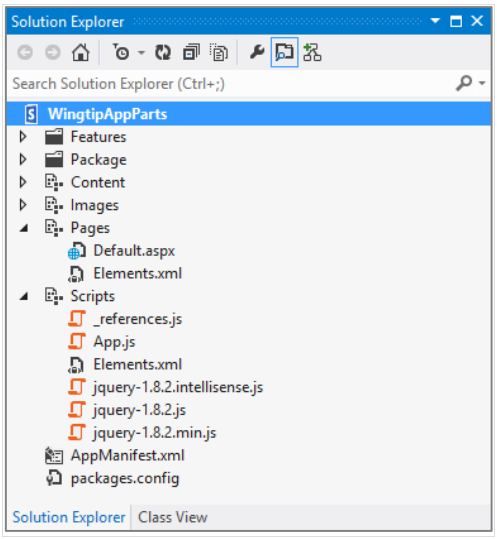
16. Close the browser to stop the debugger and go back to Visual Studio.

In this exercise you created a simple SharePoint-Hosted app 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 App 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 **SharePoint Add-in** template under the **Templates → Visual C# → Office** section.
      ii) **Name**: **WingtipAppParts**
      iii) **Location:** C:\Student\Modules\03_SharePointHostedAddins\Lab
      iv) Click **OK.**
   c) In the **New SharePoint Add-in** dialog, enter a URL of http://dev.wingtip.com for the test site and select the option to create a **SharePoint-hosted Add-in**.
   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.

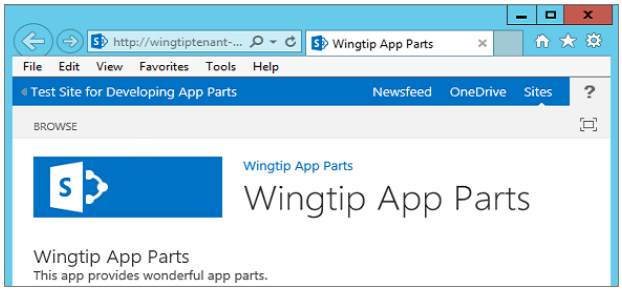
```
<asp:Content ContentPlaceHolderID="PlaceHolderPageTitleInTitleArea" runat="server">
    Wingtip App Parts
</asp:Content>

<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.

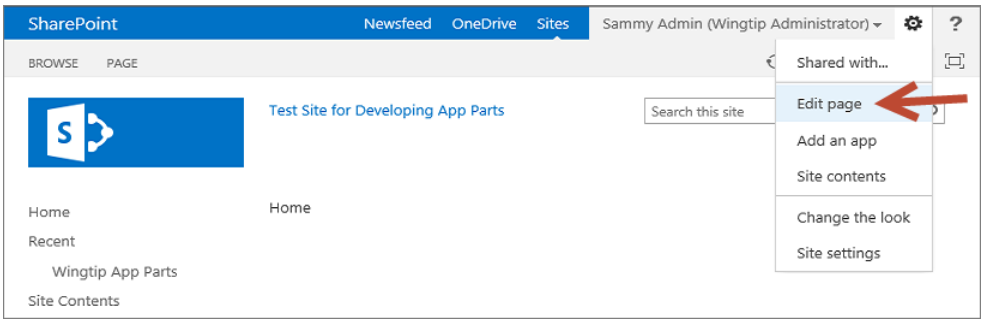
Remember that the purpose of this lab exercise is to create app parts. However, the app still requires a start page even if the start page doesn't really provide any real functionality. However, the start page is helpful for testing because it provides a link back to the

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.

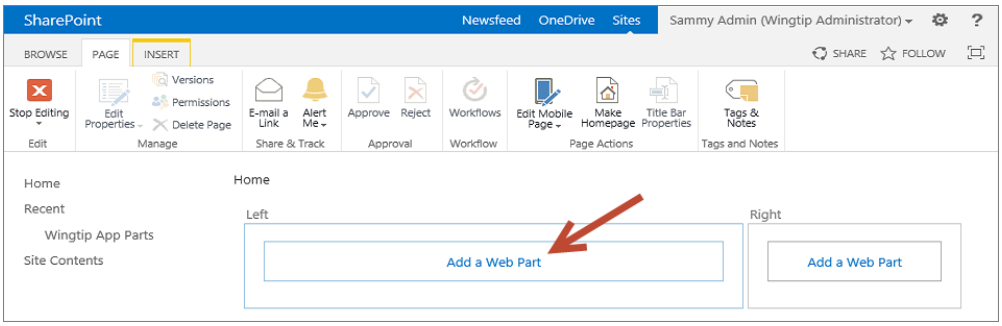
6. Test your work by pressing the **{F5}** key to launch a debugging session. When the app starts, you should see the start page appear as the one shown in the following screenshot. Leave this start page open for the next step.



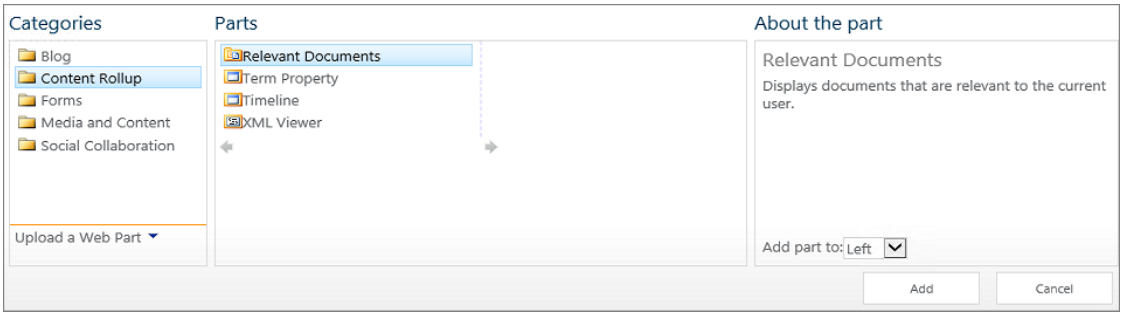
7. Navigate to the host web and see what web parts are available out of the box.
   a) Click the link in the top left corner of the start page to navigate back to the host web. This should redirect you to the home page of the Blank site at **http://apppart.wingtip.com**.
   b) Drop down the **Site Actions**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

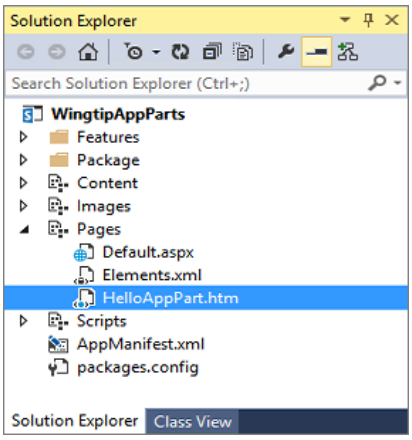


   d) While you do not need to add a web part to the page in this step, your objective is simply to see what web parts are available. In just a bit, your will see your custom app parts available in this web part catalog.

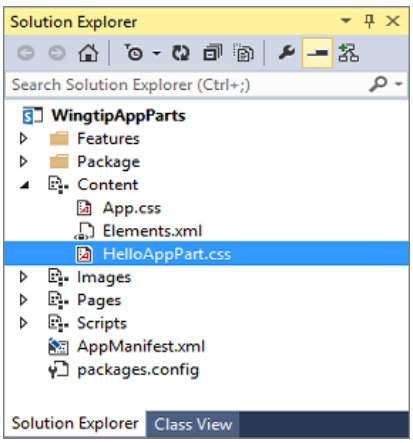


e) Once you have looked through the available set of out-of-the-box web parts, close the browser to end the debugging session.

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**.

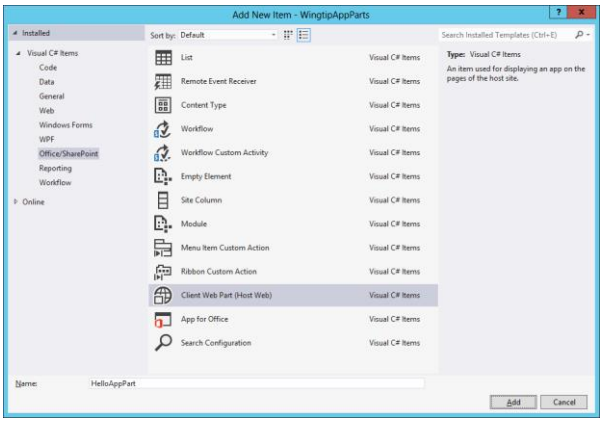
```
<!DOCTYPE html>

<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.

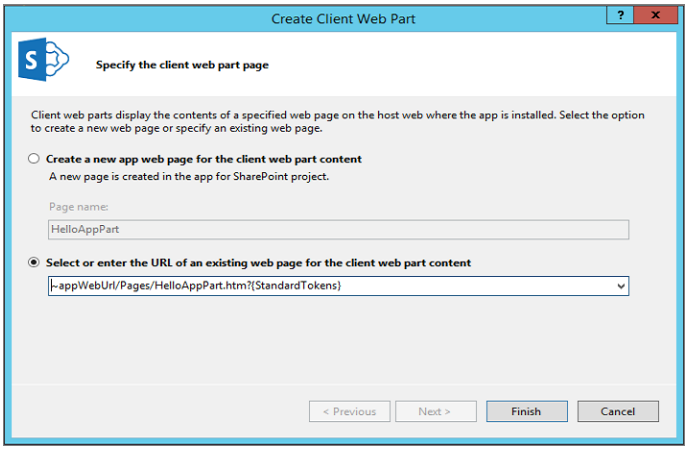
a) In the Solution Explorer, right-click on the **WingtipAppParts** project and select the **Add New Item** command.

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



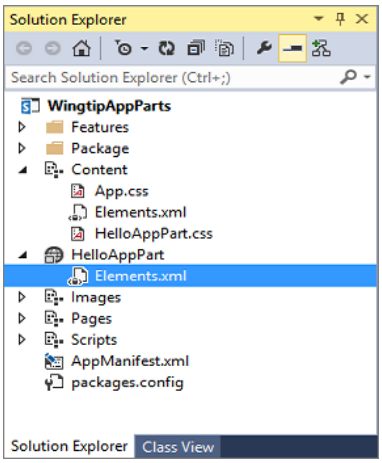
c) Click the **Add** button at the bottom right of the **Add New Item** dialog to add the new Client Web Part project item. When you click the Add button, you should then see the **Create Client Web Part** dialog.

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.

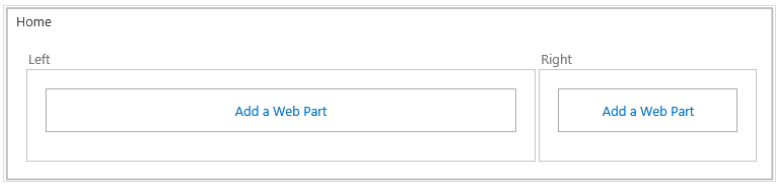
11. Test your work by adding the **HelloAppPart** app part to a web part page in the host web.

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

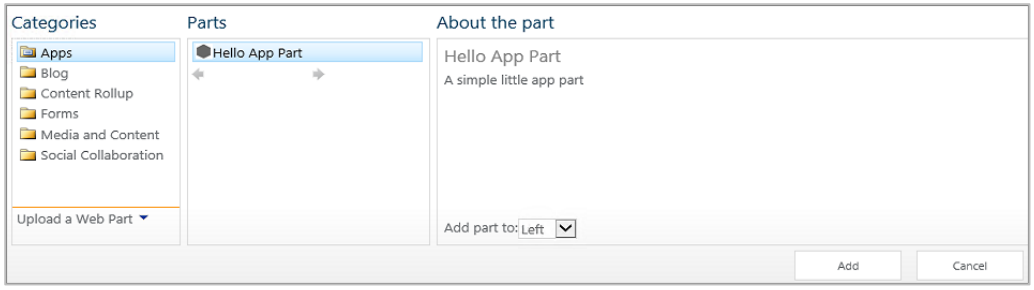
b) When you see the app's start page, click the link to redirect to the home page of the host web. (Reminder: this link is in the top left corner of the page)

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.

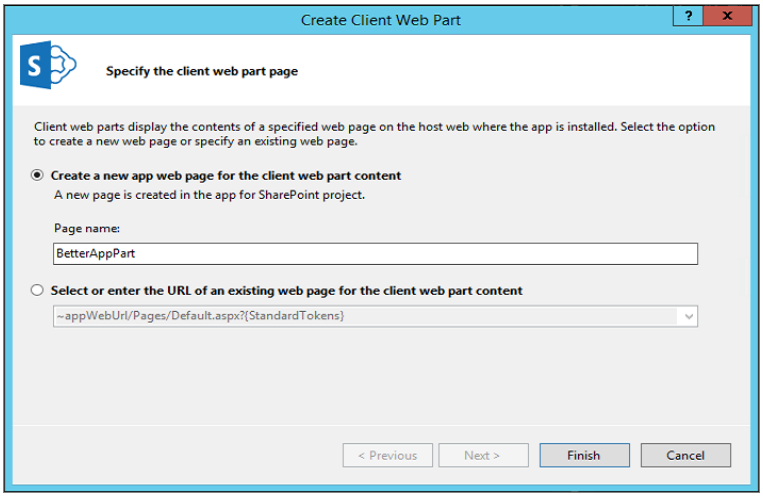
h) Close the browser to end the debugging session and then return to Visual Studio.

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.

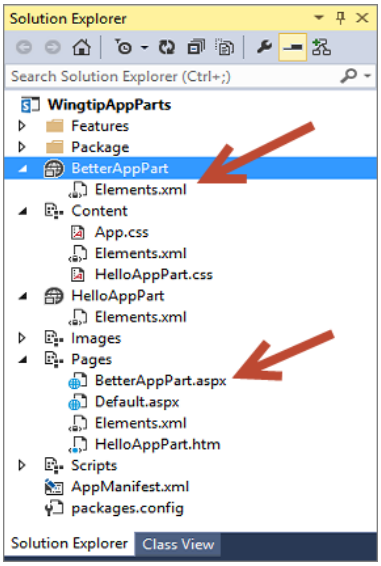
Exercise 4: Creating an App Part with Custom Properties

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,
  };

}();
```

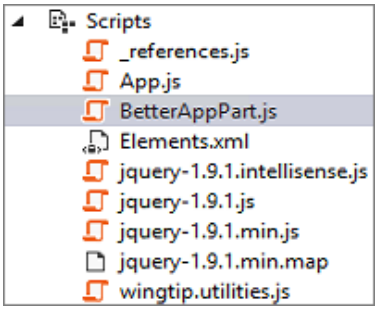
   c) Close **wingtip.utilities.js**.

   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.

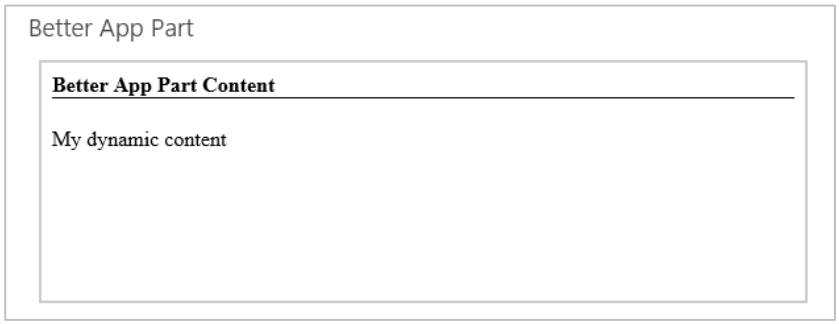
c) Use the **Edit page** menu from the **Site Actions** menu to move the page into Edit Mode.

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}
```

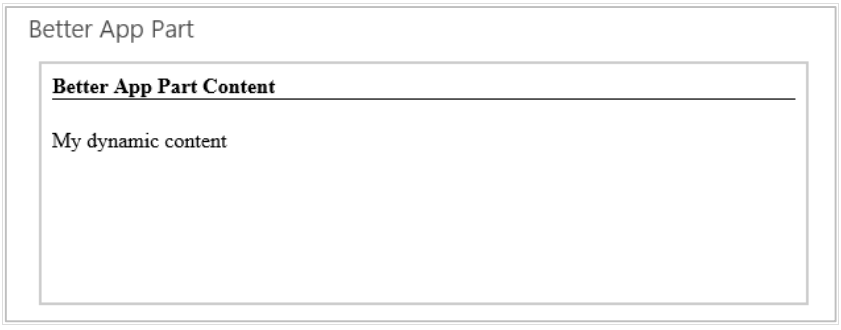
d) Save and close the **elements.xml** file.

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

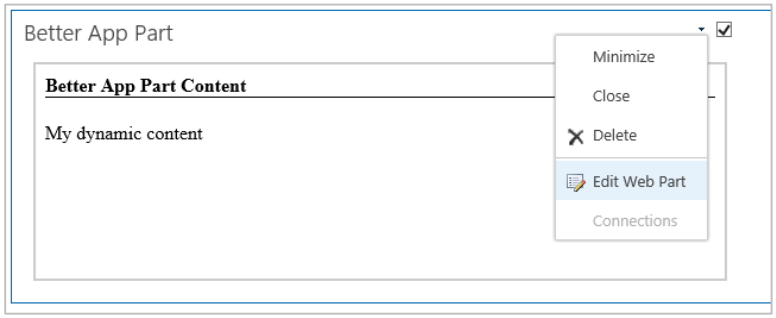
```
$(function () {

  $("#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 });
  }

});
```

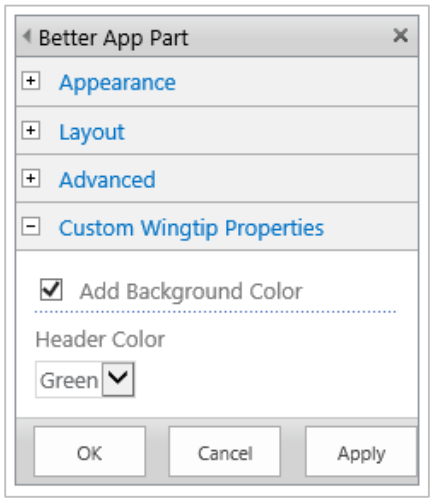
10. Save and close the **BetterAppPart.js** file.
11. Test your work.
    a) Using the Visual Studio **Build** Menu select **Rebuild WingtipAppParts** to ensure the updated code will be deployed.
    b) Press **{F5}** to begin a debugging session.
    c) When you see the app's start page, click the link 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**.



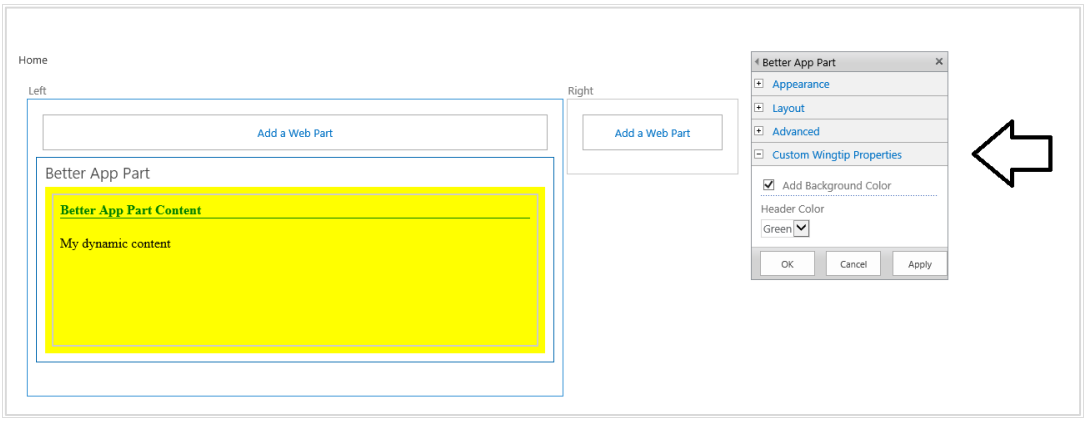
    h) As you are still in the Page Edit mode you can use the drop down app part menu in the top-right corner of the app part title bar to select the **Edit Web Part** menu. This will display the editor parts that make it possible for the user to modify app part properties.



    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.