|  |
| --- |
| Microsoft SharePoint 2013 Preview - Hands-on Lab |
| SharePoint-Hosted Apps |
| Verified Against Build 15.0.4420.1017 |

|  |
| --- |
| Microsoft  Version 1.0  August 14, 2012 |

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links are provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright © 2012 Microsoft Corporation. All rights reserved.

Microsoft and SharePoint are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

# Technical Background

This lab requires you first run a script that creates a dedicated SharePoint site collection that you will use throughout all exercises in this lab. By working in an isolated site collection, you can rest assured that your changes will not affect other sites in the environment.

## Create Lab SharePoint Site Collection

In the files provided with the hands on lab, run the batch file called SetupModule.bat by double clicking it. This file will execute a PowerShell script that will create a new site collection at [http://intranet.contoso.com/sites/](http://intranet.contoso.com/sites/IntroSpApps)SPHostedApps.

|  |
| --- |
| Description: C:\Users\vesaj\Pictures\DVD_ART36\Artwork_Imagery\Icons - Illustrations\_ SUPER VISTA STYLE\yield sign red white exclamation point.png **Important** |
| *It is important you run this batch file to create the site collection before working through any of the exercises as the exercises contain instructions for working with this specific site collection at the specific URL created by the script.* |

The script will first check to see if there is already a site collection at the specified address. If there is it will delete the site collection before creating it. Therefore if you run into problems with the lab, feel free to rerun the batch file to reset the environment and restart the exercise.

# Introduction

## Estimated time to complete this lab

60 minutes

## Objectives

After completing this lab, you will be able to:

* Learn the fundamentals of creating SharePoint-Hosted Apps
* Learn to include jQuery libraries in Apps
* Learn to use additional artifacts such as lists, Web Parts, pages, and Web services in Apps
* Learn to use the Chrome Control

## Overview of Lab

SharePoint Apps can be centered on SharePoint artifacts, cloud resources, or both. In this lab, you will create several SharePoint-Hosted Apps that primarily use SharePoint artifacts. At the end of the lab, you will have a good understanding of the techniques for creating SharePoint-Hosted Apps.

## Virtual Machine Technology

The computers in this lab are virtual machines that are implemented using Microsoft Hyper-V. Before starting each virtual machine, ensure you apply the **Start-Lab** snapshot (if supplied). When you have started a virtual machine, log on by pressing **CTRL+ALT+END** and supply the credentials listed in the lab instructions.

## Computers in this lab

This lab uses virtual machines as described in the following table. Before you begin the lab, you must start the virtual machines and then log on to the computers.

|  |  |
| --- | --- |
| **Virtual Machine** | **Role** |
| {Supplied by Instructor} | Domain Controller |
| {Supplied by Instructor} | Actual SharePoint environment with Office client and other required software. |

All user accounts in this lab use the password {Supplied by Instructor}.

## Lab Setup Script

In the files provided with the hands on lab, run the batch file called **SetupModule.bat** by double clicking it. This file will execute a PowerShell script that will create a new site collection at [http://intranet.contoso.com/sites/](http://intranet.contoso.com/sites/IntroSpApps)SPHostedApps.

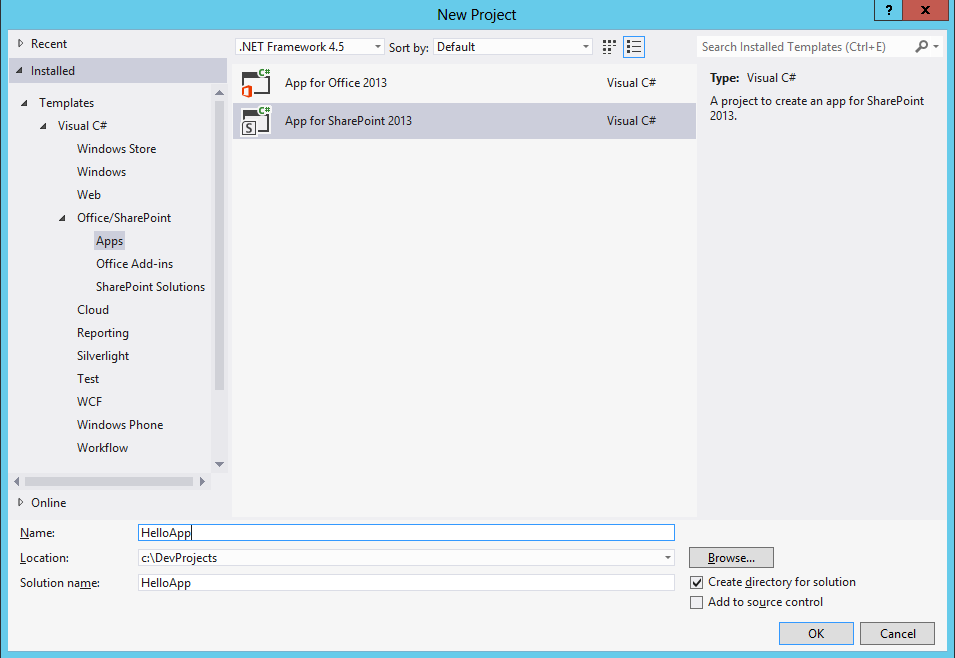
# Exercise 1: Hello, Apps

In this exercise, you will create a simple SharePoint App using Visual Studio. This exercise will familiarize you with the development process, toolset, and patterns for simple App development.

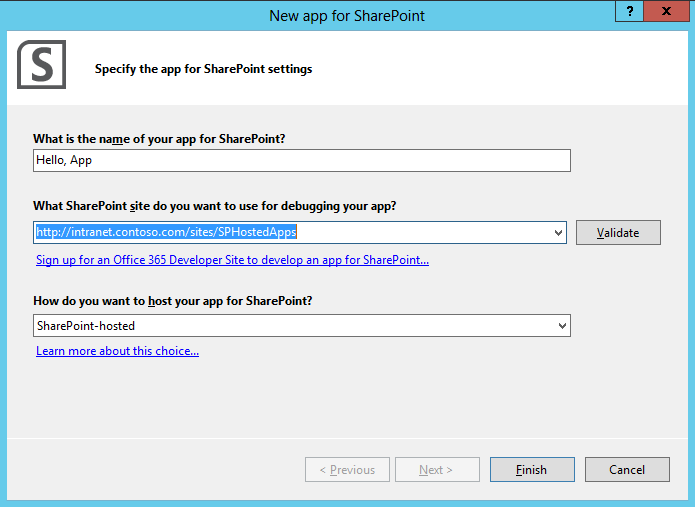
## Task 1 – Create a New App

In this task, you will develop a new App in Visual Studio

1. Open Microsoft Visual Studio 2012 and create a new SharePoint App project
   1. **Open** Microsoft Visual Studio 2012
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚Apps** node and select the **App for SharePoint 2013** project template
   4. Name the new project **HelloApp**
   5. Click the **OK** button



* + - Create new SharePoint App Project
  1. In the **New App for SharePoint Wizard**, name the new App **Hello, App**.
  2. Specify the site you will use for this lab.
  3. Select **SharePoint-hosted** as the hosting type.
  4. Click **Finish**.



* New App for SharePoint Wizard

1. Code the App
   1. Open **Pages\Default.aspx** for editing.
   2. **Add** the code below within the **PlaceHolderMain** Content Placeholder control:

<div id="displayDiv"></div>

<input type="button" value="Push me!" onclick="hello();"/>

* 1. Open **Scripts\App.js** for editing.
  2. Add the following code to the library:

function hello() {

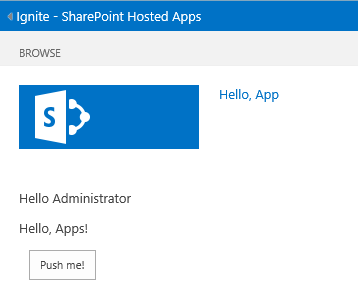
$get("displayDiv").innerHTML = "<p>Hello, Apps!</p>";

}

## Task 2 – Run the App in Debug mode

In this task, you will run the App and see it hosted in SharePoint.

1. Run the App
   1. In Visual Studio, select **Debug⮚Start Debugging**, observe the Output Window for installation messages.
   2. Launch the App
   3. In the App, click the **Push Me** button.



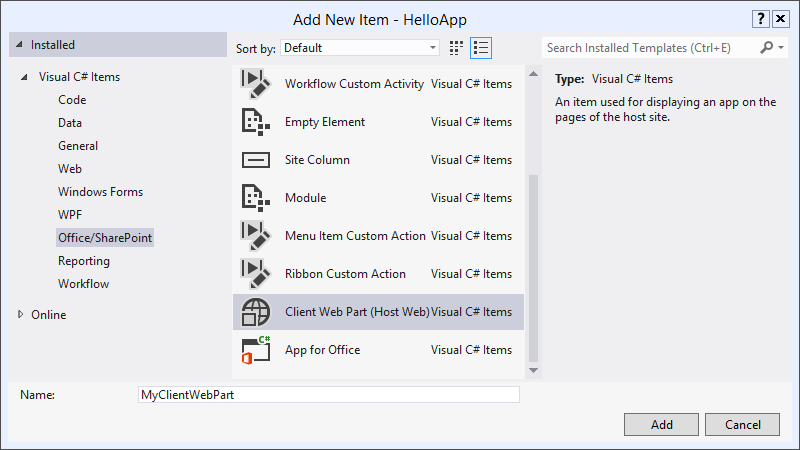
* + - Running the App

1. Shutdown the App
   1. In Visual Studio, select **Debug⮚Stop Debugging**, observe the Output Window for retraction messages.

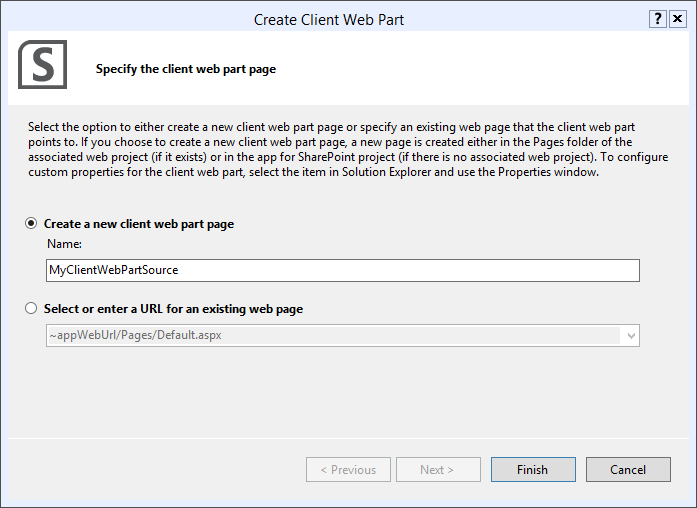
## Task 3 – Add a Client Web Part to the App

In this task, you will add a Client Web Part and code it. This web part can be used on pages outside of the App to add functionality to the hosting SharePoint site.

1. Add a Client Web Part
   1. In the **Solution Explorer**, right click the **HelloApp** project node and select **Add⮚New Item** from the context menu.
   2. In the **Add New Item** dialog, select **Client Web Part**.
   3. Name the new item **MyClientWebPart** and click **Add**.



* + - Adding a Client Web Part
  1. In the Create Client Web Part dialog, choose **Create a new client web part page** and give it the name **MyClientWebPartSource**.



* 1. **Replace** the entire contents of the **Elements** file with the following:

<?xml version="1.0" encoding="utf-8"?>

<Elements xmlns="http://schemas.microsoft.com/sharepoint/">

<ClientWebPart Name="MyClientWebPart"

Title="My Client WebPart"

Description="A simple client web part"

DefaultWidth="200"

DefaultHeight="200">

<!-- Content element identifies the location of the page that will render inside the client web part

Properties are referenced on the query string using the pattern \_propertyName\_

Example: Src="~appWebUrl/Pages/ClientWebPart1.aspx?Property1=\_property1\_" -->

<Content Type="html"

Src="~appWebUrl/Pages/MyClientWebPartSource.aspx?Message=\_DisplayMessage\_" />

<!-- Define properties in the Properties element.

Remember to put Property Name on the Src attribute of the Content element above. -->

<Properties>

<Property Name="DisplayMessage"

Type="string"

WebBrowsable="true"

WebDisplayName="Display Message"

WebDescription="A message to display"

WebCategory="Configuration"

DefaultValue="Hello, Client Web Part!"

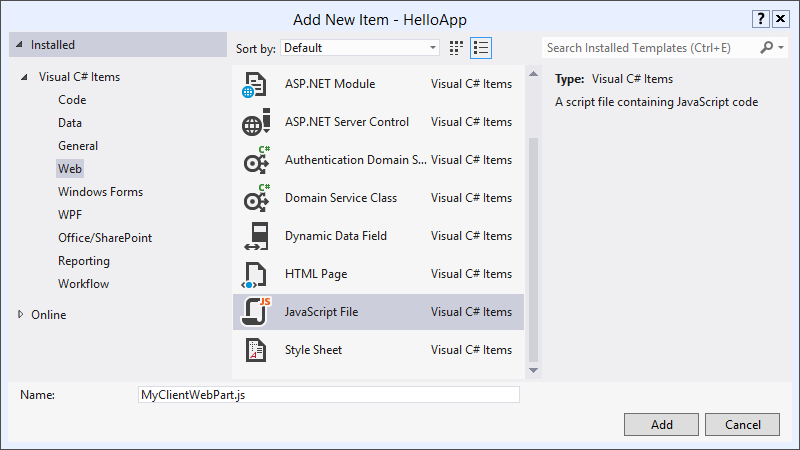
RequiresDesignerPermission="true" />

</Properties>

</ClientWebPart>

</Elements>

* 1. Add a new JavaScript file to hold the logic for the client web part.
     1. Right-click the **Scripts** folder in Visual Studio and choose **Add / New Item**. Choose **Web / JavaScript File** and give it a name of **MyClientWebPart.js.**



* 1. **Add** the following code to the new MyClientWebPart.js file.

function helloAppPart() {

var message = getQueryStringValue("Message");

document.getElementById("appPartDiv").innerHTML = "<p>" + message + "</p>";

}

function getQueryStringValue(paramName) {

var params = document.URLUnencoded.split("?")[1].split("&");

var strParams = "";

for (var i = 0; i < params.length; i = i + 1) {

var singleParam = params[i].split("=");

if (singleParam[0] == paramName)

return decodeURIComponent(singleParam[1]);

}

}

* 1. In the **Pages** folder, open the item **MyClientWebPartSource.aspx. Add** the following in the HEAD tag.

<script src="../Scripts/MyClientWebPart.js"></script>

* 1. **Replace** the contents of the BODY tag with the following.

<div id="appPartDiv"></div>

<input type="button" value="Push Me!" onclick="helloAppPart();" />

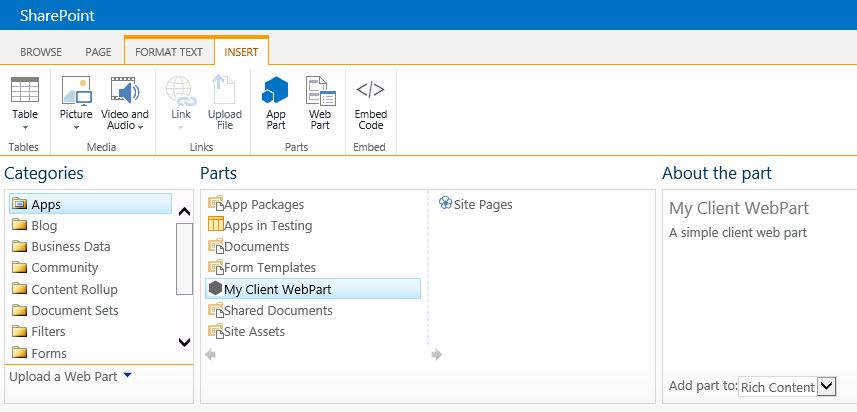
1. The result will look like the following:



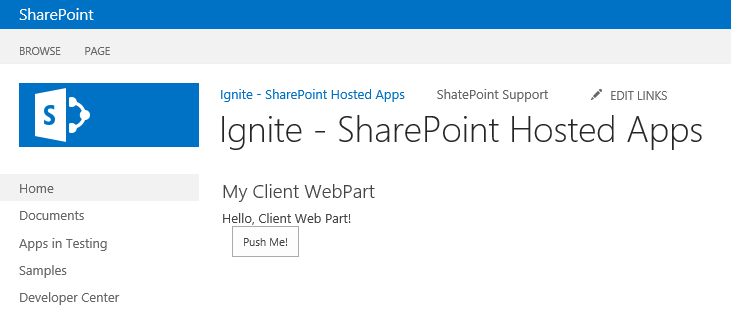
## Task 4 – Add the Client Web Part to a Page

In this task, you will run the App and add the Client Web Part to the home page of the portal.

1. Deploy the App
   1. In Visual Studio, right click the **HelloApp** node and select **Deploy** from the context menu.
   2. When the App is deployed, open your browser to the home page of the SharePoint site hosting the App (not the home page of the App itself!).
2. Add the Client Web Part
   1. In the ribbon, select **Page⮚Edit Page**.
   2. Click **Add a Web Part** in any Zone.
   3. Add the **My Client WebPart** part to the page.



* 1. Click the **Push Me** button to test the functionality.



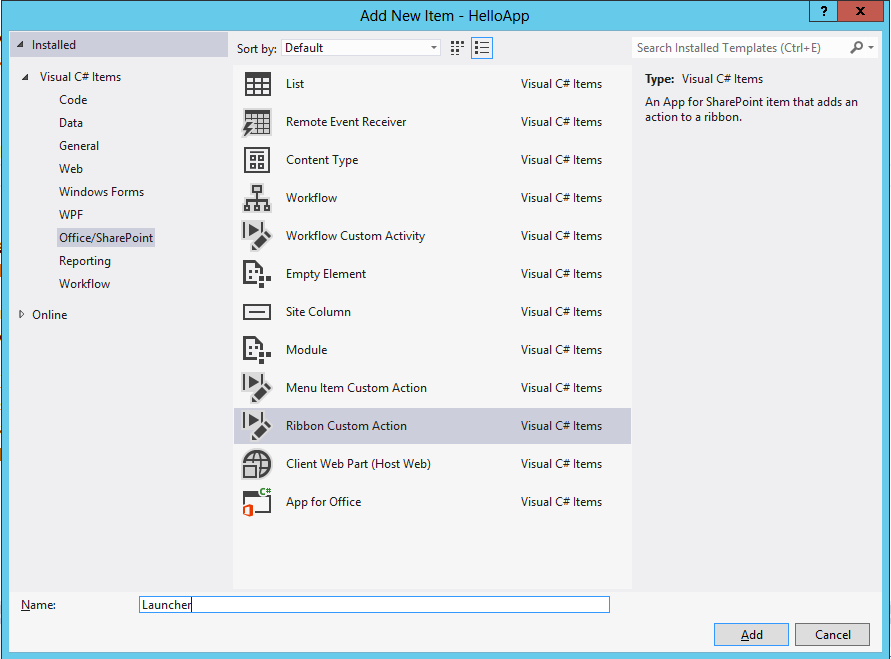
* + - Testing the Client Web Part

1. Retract the App
   1. In Visual Studio, right click the **HelloApp** node and select **Retract** from the context menu.
   2. Note that the **Client Web Part** is removed from the home page.

## Task 5 – Add a button to the Ribbon

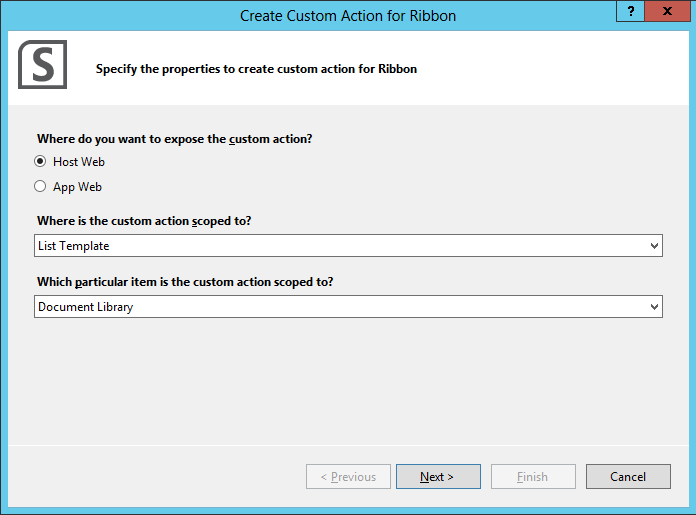
In this task, you will add a custom action to the ribbon of all Document Libraries to launch the App.

1. In Visual Studio, right click the **HelloApp** node and select **Add⮚New Item** from the Context menu.
2. In the **New Item** dialog, select **Ribbon Custom Action**.
3. Name the item **Launcher** and click **Add**.

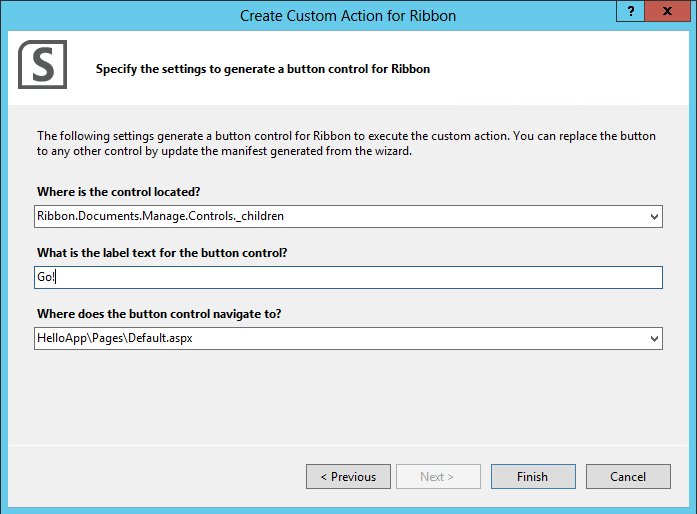


* + - Adding a UI Custom Action

1. In the Create Custom Action for Ribbon window, choose to expose the custom action to the **Host Web.** The Custom Action should be scoped to **List Template**, and the particular item the custom action is scoped to is **Document Library**.



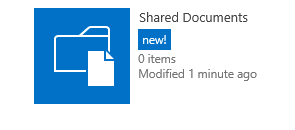
1. In the next window, we will specify settings to generate a button control for the ribbon. Use the following settings:
   1. Where is the control located? **Ribbon.Documents.Manage.Controls.\_children**
   2. What is the label text for the button control? **Go!**
   3. Where does the button control navigate to? **HelloApp\Pages\Default.aspx**



## Task 6 – Launch an App from the Ribbon

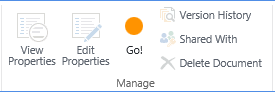
In this task, you will deploy the App and use the button on the Ribbon to launch it.

1. In Visual Studio, right click the **HelloApp** node and select **Deploy** from the Context menu.
2. After the App is deployed, open the home page of the portal where the App is hosted (not the App itself!).
3. On the home page, click **Site Contents**.
4. Click **Add an App**.
5. Select to **Add a Document Library**.
6. Name the Document Library **Shared Documents**.
7. On the Apps page, click the newly-created **Shared Documents** library.



* + - The Shared Documents Library

1. In the ribbon, click the **Files** tab.
2. **Launch** the App using the button you created.



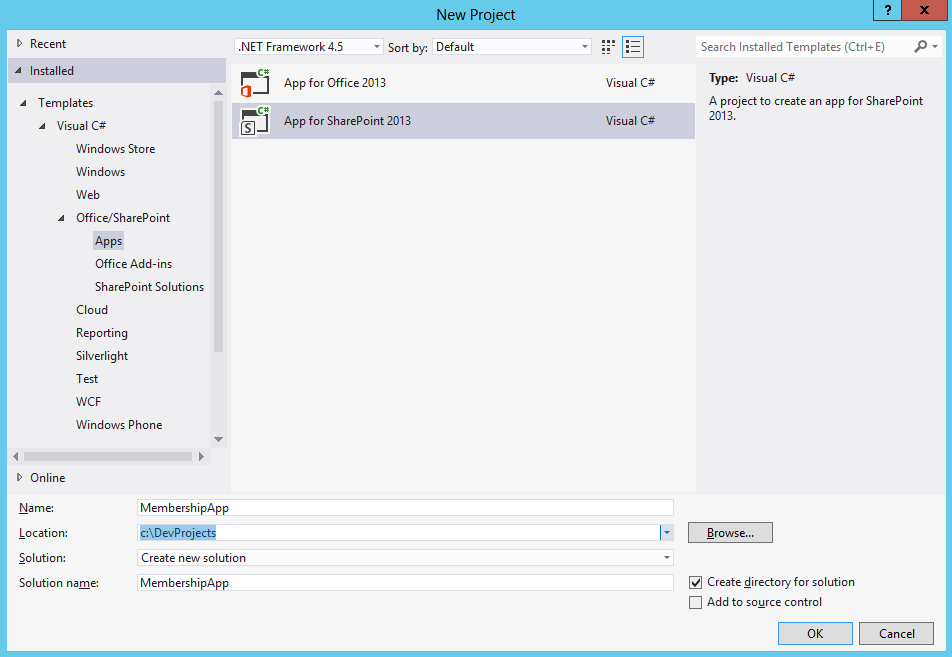
* + - Launching the App from the Ribbon

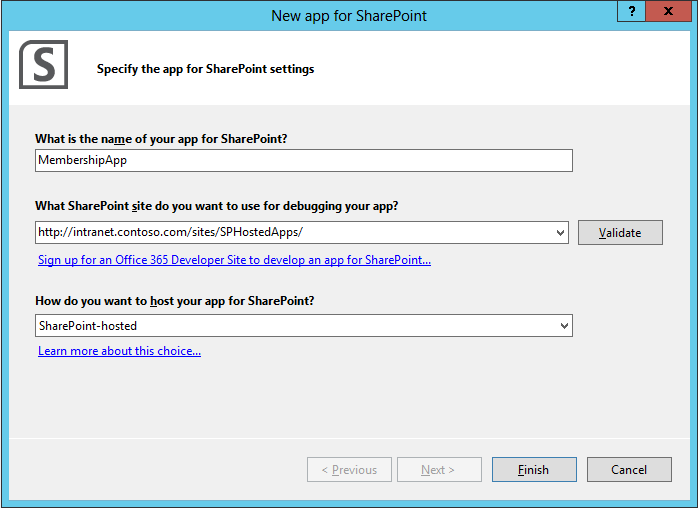
# Exercise 2: Membership App

In this exercise, you will create a SharePoint App that accesses the User Information list of the hosting site. This exercise will help you understand how to use request permissions in an App and access lists in the hosting SharePoint web. You will also make use of jQuery in the App.

## Task 1 – Create a New App

In this task, you will develop a new App in Visual Studio

1. Open Microsoft Visual Studio 2012 and create a new SharePoint App project
   1. **Open** Microsoft Visual Studio 2012
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚Apps** node and select the **App for SharePoint 2013** project template
   4. Name the new project **MembershipApp**
   5. Click the **OK** button
      * 1. 
      * Create new SharePoint App Project
   6. In the **New App for SharePoint Wizard**, name the new App **Membership App**.
   7. In the **New App for SharePoint Wizard**, specify the site you created prior to starting the lab for hosting Apps.
   8. Select **SharePoint-hosted** as the hosting type.
   9. Click **Finish**.



* + - New App for SharePoint Wizard

1. Code the App
   1. Open **Default.aspx** in Visual Studio for editing
   2. **Add** the following code in the **PlaceHolderMain** Content Placeholder control with the following:

<div id="peopleDiv"></div>

* 1. **Open** **App.js** for editing.
  2. **Add** the following code to **App.js** to query the User Information list of the hosting site:

$(document).ready(function () {

Membership = {

element: '',

url: '',

init: function (element) {

Membership.element = element;

Membership.url = \_spPageContextInfo.webAbsoluteUrl + "/\_api/site/rootweb/lists/getByTitle('User%20Information%20List')/items?$select=Title,Name";

},

load: function () {

$.ajax(

{

url: Membership.url,

method: "GET",

headers: {

"Accept": "application/json;odata=verbose",

},

success: Membership.onSuccess,

error: Membership.onError

}

);

},

onSuccess: function (data) {

var results = data.d.results;

var html = "<table>";

for (var i = 0; i < results.length; i++) {

html += "<tr><td>";

html += results[i].Title;

html += "</td><td>"

html += results[i].Name;

html += "</td><tr>";

}

html += "</table>";

Membership.element.html(html);

},

onError: function (err) {

alert(JSON.stringify(err));

}

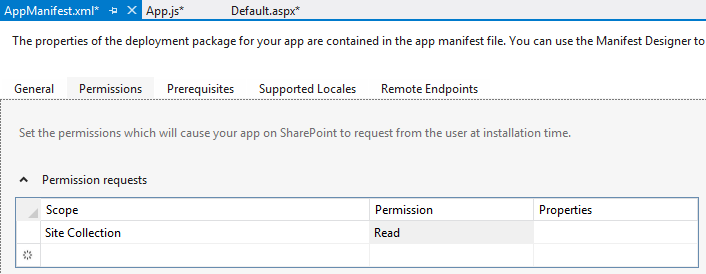
}

Membership.init($('#peopleDiv'));

Membership.load();

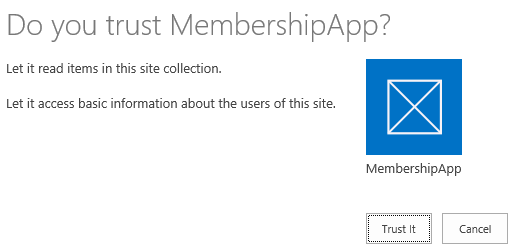
});

* 1. Open the **AppManifest.xml** designer.
  2. In the **Permission Requests** table, select **Site Collection** as the Scope.
  3. Select **Read** as the Permission.



## Task 2 – Run the App in Debug mode

In this task, you will run the App and see it hosted in SharePoint.

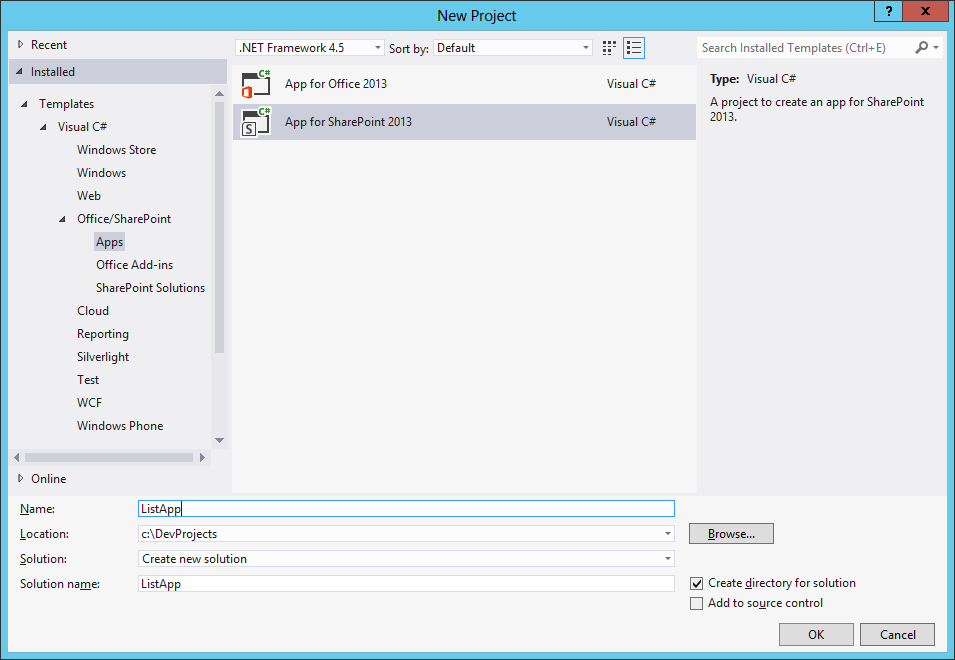
1. Run the App
   1. Select **Debug⮚Start Debugging** in Visual Studio, observe the **Output Window** for installation messages, and verify that **Internet Explorer** opens to the SharePoint Apps site you created before starting the lab.
   2. When prompted by SharePoint to grant permissions, click **Trust It**.
      * 1. 
      * Grant the App Permission
   3. From the Apps page, launch the Membership App.
   4. You should now see a list of all the members of the Site Collection hosting the app.

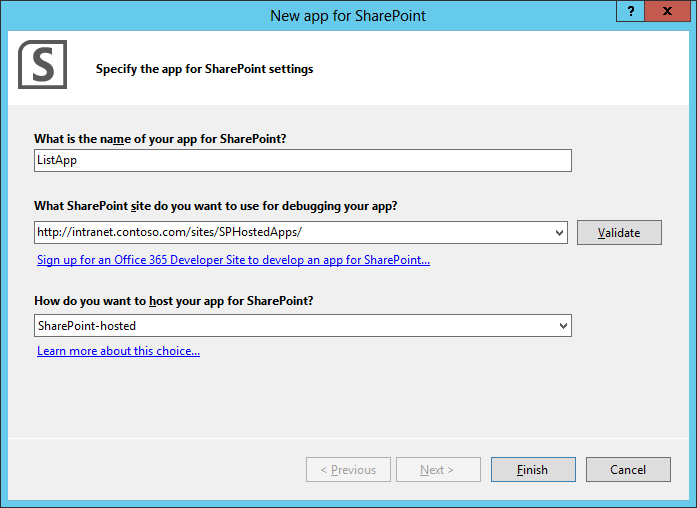
# **Exercise 3: List-Powered Apps**

In this exercise, you will create a SharePoint App that uses SharePoint lists. This exercise will familiarize you with basic SharePoint concepts and how they can be used in Apps.

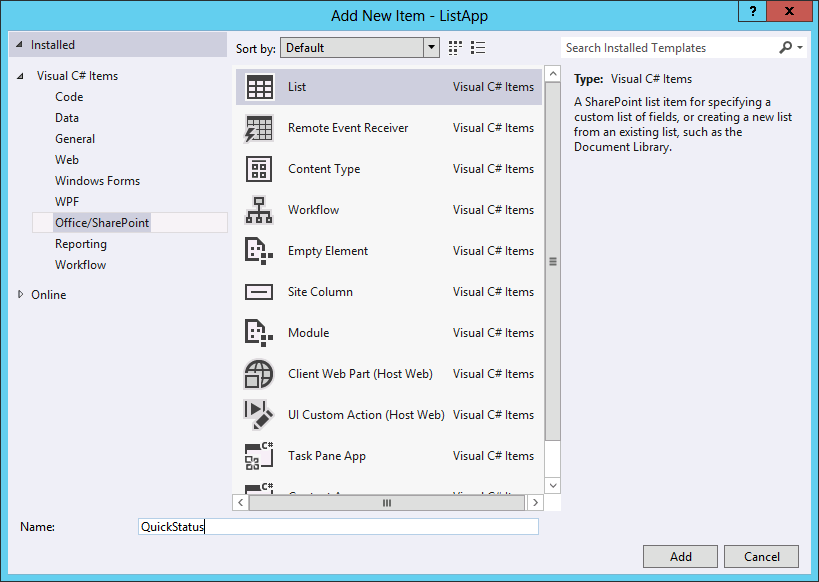
## Task 1 – Create a New App

In this task, you will develop a new App in Visual Studio

1. Open Microsoft Visual Studio 2012 and create a new SharePoint App project
   1. **Open** Microsoft Visual Studio 2012
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚Apps** node and select the **App for SharePoint 2013** project template
   4. Name the new project **ListApp**
   5. Click **OK**.
      * 1. 
      * Create new SharePoint App Project
   6. In the **New App for SharePoint Wizard**, name the App, **List App**.
   7. Specify the site you will use for this lab.
   8. Select **SharePoint-hosted** as the hosting type.
   9. Click **Finish**.

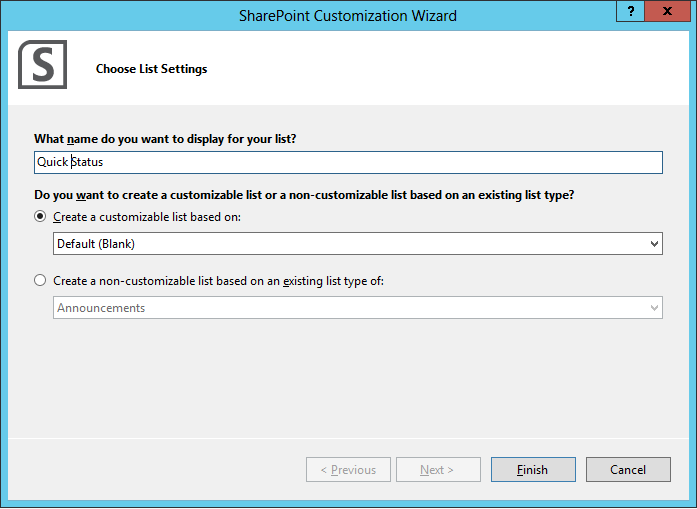
  
*New App for SharePoint Wizard*

1. In the **Solution Explorer**, Right click on the **ListApp** node, select Add, select **Add New Item.**
2. In the **Add New Item** dialog, select **List**.
3. Enter **QuickStatus** for the name and click **Add**.



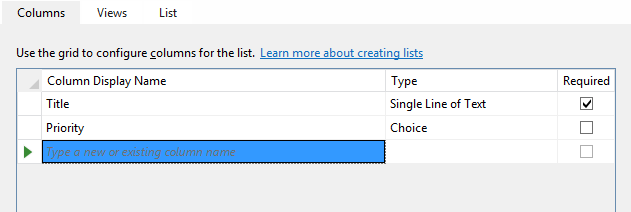
* + - 1. Add New Item Dialog

1. In the Choose List Settings dialog, change the display name to **Quick Status.**
2. Click **Finish**.



* + - 1. SharePoint Customization Wizard

1. In the **List Schema Designer**, add a field called **Priority**, and change the Type to **Choice**.



* + - 1. List Schema Designer

1. Code the App.
   1. Open **Content/App.css** in Visual Studio for editing
   2. **Add** the following:

.messages

{

margin: 0px;

padding: 0px;

float: left;

border-top: 1px #D8D8D0;

width: 300px;

}

.messages tr

{

padding: 5px 5px 10px 10px;

list-style-type: none;

background-color: #FAFAF4;

color: #000033;

}

.subtle

{

color: #999999

}

* 1. Open **Scripts/App.js** in Visual Studio for editing
  2. **Add** the following code:

var \_surlWeb = "";

$(document).ready(function()

{

\_surlWeb = \_spPageContextInfo.webServerRelativeUrl;

if (\_surlWeb.length > 0 && \_surlWeb.substring(

\_surlWeb.length - 1, \_surlWeb.length ) != "/") {

\_surlWeb += "/"; }

loadMessages();

});

function loadMessages()

{

$.ajax({

url: \_surlWeb +

"\_api/lists/getbytitle('Quick Status')/items?$select=ID,Title,Author/Title&$expand=Author&$orderby=ID desc",

headers: {

"accept": "application/json;odata=verbose",

"X-RequestDigest": $("#\_\_REQUESTDIGEST").val()

},

success: postMessageListRetrieve,

error: oops

});

}

function postMessageListRetrieve(data)

{

var items = [];

items.push("<table class='messages'>");

$.each(data.d.results, function(key, val)

{

items.push('<tr id="' + val.ID + '"><td><div>' +

val.Title + '</div><div class="subtle">' +

val.Author.Title + '</div></td></tr>');

});

items.push("</table>");

$('#messages').html(items.join(''));

}

function oops(data) {

alert(data.responseText);

}

function addMessage()

{

$.ajax({

url: \_surlWeb + "\_api/lists/getbytitle('Quick Status')/items",

type: "POST",

data: JSON.stringify(

{ '\_\_metadata': {

'type': 'SP.Data.Quick\_x0020\_StatusListItem'},

'Title': $('#messageInput').val()

}),

headers: {

"Content-Type" : "application/json;odata=verbose",

"accept": "application/json;odata=verbose",

"X-RequestDigest": $("#\_\_REQUESTDIGEST").val()

},

success: loadMessages,

error: oops

});

}

|  |
| --- |
| Description: C:\Users\vesaj\Pictures\DVD_ART36\Artwork_Imagery\Icons - Illustrations\_ SUPER VISTA STYLE\yield sign red white exclamation point.png **Important** |
| *In the above code, note the following:*  *jQuery’s $.ajax function is used to make requests to the REST APIs on the server.*  *An X-RequestDigest header is added to the request, which fulfills a requirement in SharePoint APIs to have a “form digest” that defends against XSS requests. This value is included in the page in a hidden input called \_\_REQUESTDIGEST.*  *REST queries to SharePoint are made via the following URL:*  *<webiteurl>/\_api/lists/getbytitle('Quick Status')/items?$select=ID, Title,Author/Title&$expand=Author&orderby=ID desc  The query string here chooses specific fields to return in the query, including the “title” (full name) of the user who added an item. It also sorts it by ID in descending order so that newer posts appear first.* |

* 1. Open **Pages/Default.aspx** for editing
  2. **Add** the following within the **PlaceHolderMain** Content Placeholder control:

<div>

<div id="messageInputArea">

<input type="text" id="messageInput"></input>

<input type="button" value="Add"

onclick="addMessage()"></input>

</div>

<div id="messages">

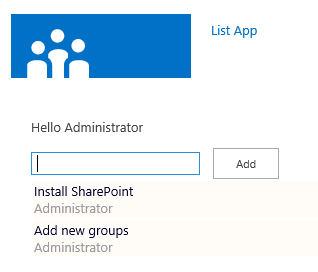
Please wait, loading...

</div>

</div>

## Task 2 – Run the App in Debug mode

In this task, you will run the App and see it hosted in SharePoint.

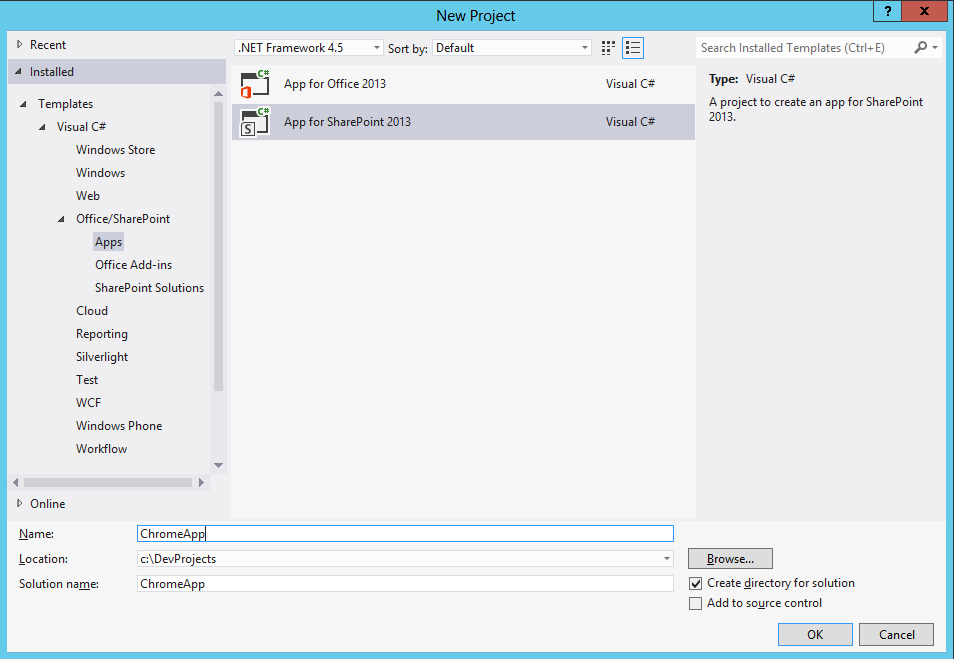
1. Run the App
   1. Select **Debug⮚Start Debugging** in Visual Studio, observe the **Output Window** for installation messages, and verify that **Internet Explorer** opens to the SharePoint Apps site you created before starting the lab.
   2. From the Apps page, launch the **List App**.
   3. Try adding new items to the list using the App.
      * 1.   
           *Running the App*

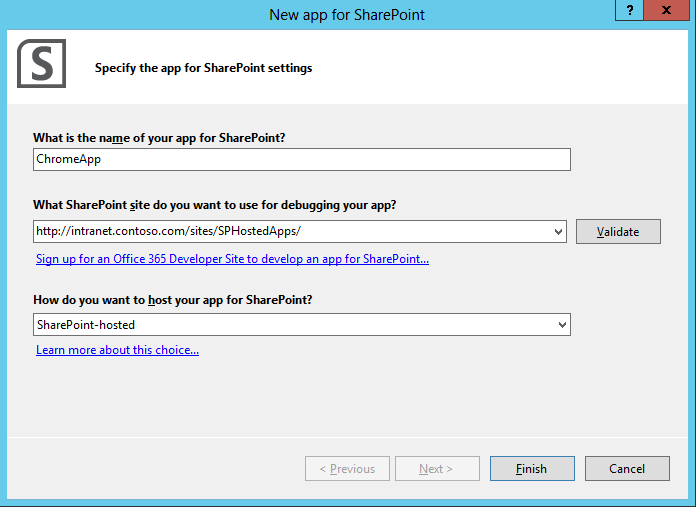
# **Exercise 4: The Chrome Control**

In this exercise, you will create a SharePoint App that uses the chrome control. The chrome control is used to brand a SharePoint App similarly to the hosting SharePoint web.

## Task 1 – Create a New App

In this task, you will develop a new App in Visual Studio

1. Open Microsoft Visual Studio 2012 and create a new SharePoint App project
   1. **Open** Microsoft Visual Studio 2012
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚Apps** node and select the **App for SharePoint 2013** project template
   4. Name the new project **ChromeApp**
   5. Click **OK**.
      * 1. 
      * Create new SharePoint App Project
   6. In the **New App for SharePoint Wizard**, name the App, **Chrome App**.
   7. Specify the site you will use for this lab.
   8. Select **SharePoint-hosted** as the hosting type.
   9. Click **Finish**.



* New App for SharePoint Wizard

## Task 2 – Code the App

In this task, you code the app to use the Chrome control using two approaches, declaratively using the chrome control and programmatically setting properties of the chrome control.

1. In the **Solution Explorer**, Right click on the **Scripts** node, select **Add Existing Item.**
   1. **Navigate** to C:\Program Files\Common Files\Microsoft Shared\web server extensions\15\TEMPLATE\LAYOUTS.
   2. In the file dialog, select the file **sp.ui.controls.js** and click **Add**.
2. In the Solution Explorer, expand the Scripts node
   1. Open **App.js** for editing.
   2. **Replace** all of the code in the file with the following:

var hostweburl;

// Load the SharePoint resources.

$(document).ready(function () {

// Load the js file and continue to the

// success handler.

$.getScript("../scripts/sp.ui.controls.js", renderChrome)

});

function renderChrome() {

// The Help, Account, and Contact pages receive the

// same query string parameters as the main page.

var options = {

"appIconUrl": "../Images/AppIcon.png",

"appTitle": "Chrome control app",

"appHelpPageUrl": "Help.html?"

+ document.URL.split("?")[1],

"settingsLinks": [

{

"linkUrl": "Account.html?"

+ document.URL.split("?")[1],

"displayName": "Account settings"

},

{

"linkUrl": "Contact.html?"

+ document.URL.split("?")[1],

"displayName": "Contact us"

}

]

};

var nav = new SP.UI.Controls.Navigation(

"chrome\_ctrl\_container",

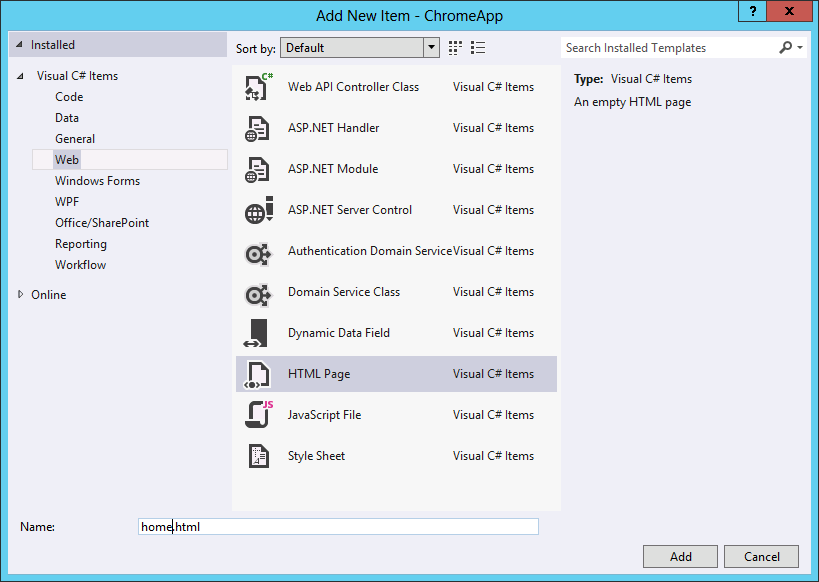
options

);

nav.setVisible(true);

}

1. In the Solution Explorer, expand the **Pages** node.
   1. Delete **Default.aspx**.
   2. Right click the **Pages** node and select **Add⮚New Item** from the context menu.
   3. In the **Add New Item** dialog, select **Web⮚HTML Page**.
   4. Name the new page **Home.html** and click **Add**.
   5. Repeat these steps to add pages **Account.html**, **Contact.html**, and **Help.html**.



* + - 1. Add New Item Dialog

1. One way to use the chrome control is to set its properties programmatically. In this example, we set properties of the chrome control using the code in app.js. Replace all of the code in **Home.html** with the following.

<!DOCTYPE html>

<html>

<head>

<title>Chrome controls declarative sample</title>

<script src="../Scripts/jquery-1.7.1.js" type="text/javascript"></script>

<script src="../Scripts/app.js" type="text/javascript"></script>

<link href="../Content/App.css" rel="stylesheet" />

</head>

<body>

<!-- Chrome control placeholder -->

<div id="chrome\_ctrl\_container"></div>

<!-- The chrome control also makes the SharePoint

Website stylesheet available to your page -->

<h1 class="ms-accentText">Main content</h1>

<h2 class="ms-accentText">The chrome control</h2>

<div id="MainContent" >

This is the page's main content.

You can use the links in the header to go to the help,

account or contact pages.

</div>

</body>

</html>

1. Another way to use the chrome control is to declaratively set properties within the HTML markup. Note that in this example, we do not set a reference to app.js. Replace all of the code in **Account.html** with the following.

<!DOCTYPE html>

<html>

<head>

<title>Account Settings</title>

<script src="../Scripts/sp.ui.controls.js" type="text/javascript">

</script>

</head>

<body>

<!-- Note the use of siteUrl to reference the site for the theming -->

<div id="chrome\_ctrl\_container" data-ms-control="SP.UI.Controls.Navigation" data-ms-options='{

"siteUrl" : "http://intranet.contoso.com/sites/SPHostedApps",

"appHelpPageUrl" : "../Pages/Help.html",

"appIconUrl" : "../Images/appIcon.png",

"appTitle" : "Chrome Control",

"settingsLinks" : [

{

"linkUrl" : "../Pages/Account.html",

"displayName" : "Account settings"

},

{

"linkUrl" : "../Pages/Contact.html",

"displayName" : "Contact us"

}

]

}'>

</div>

<div style="margin: 50px">

<h3>Account Settings Page</h3>

<table cellspacing="3" cellpadding="3" border="0">

<tr>

<td>Store Personal Infomartion</td>

<td>

<input type="checkbox" /></td>

</tr>

<tr>

<td>E-Mail Address</td>

<td>

<input type="text" value="administrator@contoso.com" /></td>

</tr>

</table>

</div>

</body>

</html>

1. Replace all of the code in **Contact.html** with the following.

<!DOCTYPE html>

<html>

<head>

<title>Contact Us</title>

<script src="../Scripts/sp.ui.controls.js" type="text/javascript">

</script>

</head>

<body>

<!-- Note the use of siteUrl to reference the site for the theming -->

<div id="chrome\_ctrl\_container" data-ms-control="SP.UI.Controls.Navigation" data-ms-options='{

"siteUrl" : "http://intranet.contoso.com/sites/SPHostedApps",

"appHelpPageUrl" : "../Pages/Help.html",

"appIconUrl" : "../Images/appIcon.png",

"appTitle" : "Chrome Control",

"settingsLinks" : [

{

"linkUrl" : "../Pages/Account.html",

"displayName" : "Account settings"

},

{

"linkUrl" : "../Pages/Contact.html",

"displayName" : "Contact us"

}

]

}'>

</div>

<div style="margin: 50px">

<h3>Contact Page</h3>

<p><a href="mailto:administrator@contoso.com">Web Master</a></p>

</div>

</body>

</html>

1. Replace all of the code in **Help.html** with the following.

<!DOCTYPE html>

<html>

<head>

<title>Help Me</title>

<script src="../Scripts/sp.ui.controls.js" type="text/javascript">

</script>

</head>

<body>

<!-- Note the use of siteUrl to reference the site for the theming -->

<div id="chrome\_ctrl\_container" data-ms-control="SP.UI.Controls.Navigation" data-ms-options='{

"siteUrl" : "http://contososerver",

"appHelpPageUrl" : "../Pages/Help.html",

"appIconUrl" : "../Images/appIcon.png",

"appTitle" : "Chrome Control",

"settingsLinks" : [

{

"linkUrl" : "../Pages/Account.html",

"displayName" : "Account settings"

},

{

"linkUrl" : "../Pages/Contact.html",

"displayName" : "Contact us"

}

]

}'>

</div>

<div style="margin: 50px">

<h3>Help Page</h3>

<p>There's really no helping you.</p>

</div>

</body>

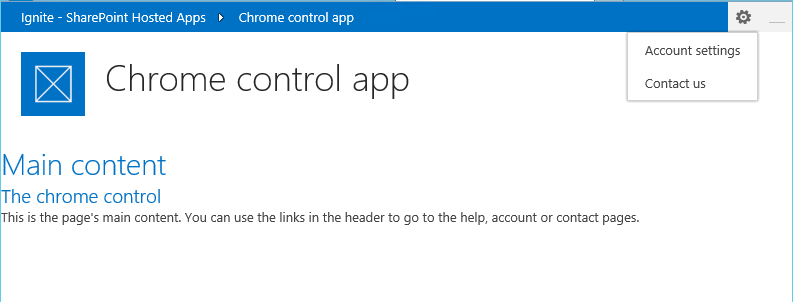
</html>

1. Open the **AppManifest.xml** designer.
   1. Set the **StartPage** element to be **Home.html**

## Task 3 – Deploy and Run the App

In this task, you will deploy the App and run it.

1. In the **Solution Explorer**, Right click on the **ChromeApp** project, select **Add Deploy.**
2. **Navigate** to the Site hosting the App.
   1. Click **Site Contents**.
   2. Click **Chrome App**.



* + - 1. Running the App

# Summary

1. This lab gave you a chance to work with SharePoint-hosted apps. You learned how to add a client web part and how to launch an app from the ribbon. You learned how to query data outside the appweb using OData, and used OData to set data in a list. Finally, you learned how to use the chrome control to provide navigation and theming for your app.