



Deep Dive into SharePoint Hosted Apps

Office 365



Hands-on lab

In this lab, you will learn how to create SharePoint Hosted Apps.

This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. 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.

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 2014 © Microsoft Corporation. All rights reserved.

Microsoft, Internet Explorer, Microsoft Azure, Microsoft Office, Office 365, SharePoint, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Exercise 1: Create a SharePoint-Hosted App

You must have an Office 365 tenant to complete this lab. To sign up for an Office 365 developer subscription:

1. Navigate to [http://msdn.microsoft.com/en-us/library/office/fp179924\(v=office.15\).aspx](http://msdn.microsoft.com/en-us/library/office/fp179924(v=office.15).aspx).
2. Under the heading **Sign up for an Office 365 Developer Site** click **Try It Free**.

▲ Sign up for an Office 365 Developer Site

You may already have access to an Office 365 Developer Site. Here are three ways to get one:

- Are you an MSDN subscriber? Visual Studio Ultimate and Visual Studio Premium with MSDN
- Do you have a midsize business and enterprise (Plan E1 or E3) Office 365 subscription? You can get a [365 subscription](#).
- You can either start with a [free 30-day trial](#), or buy an [Office 365 developer subscription](#) (with a 30-day trial).

Try it free 

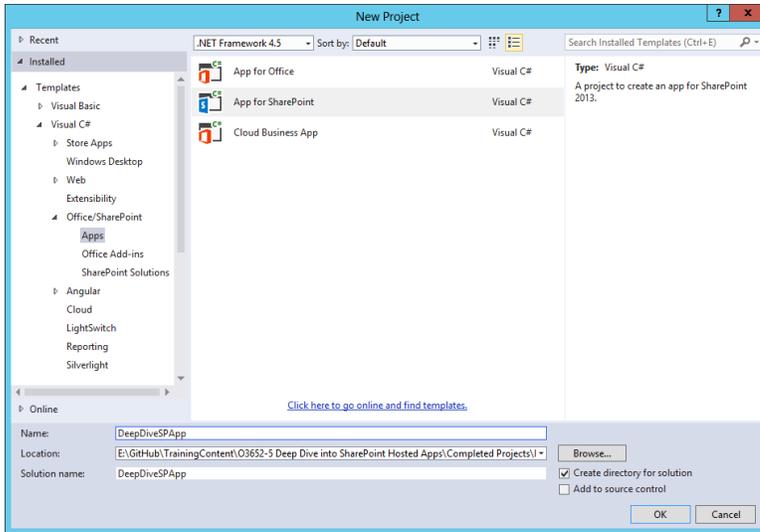
3. Fill out the form to obtain your trial Office 365 subscription.
4. When completed, you will have a developer site in the [subscription].sharepoint.com domain located at the root of your subscription (e.g. <https://mysubscription.sharepoint.com>)

In this exercise you will create a basic SharePoint-Hosted app that you can enhance in follow-on exercises.

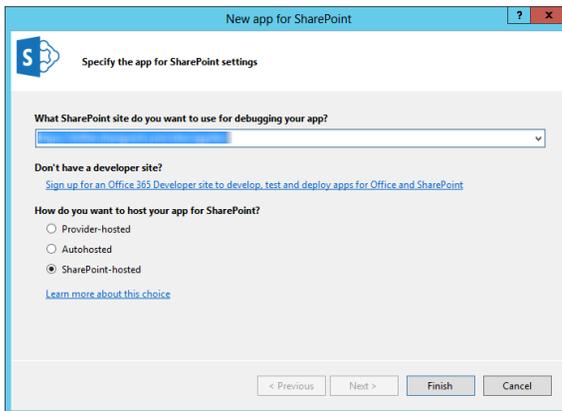
Create the new project in Visual Studio 2013.

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

Deep Dive into SharePoint Hosted Apps



4. In the New App for SharePoint wizard:
 1. Enter the address of a SharePoint site to use for testing the app (**NOTE**: The targeted site must be based on a Developer Site template)
 2. Select **SharePoint Hosted** as the hosting model.
 3. Click **Finish**.



4. When prompted, log in using your O365 administrator credentials.

Test the app

1. After the project is created, press **F5** to debug the app.
2. Verify that the app launches and greets you.

Deep Dive into SharePoint Hosted Apps



3. Stop debugging.
4. The code in the project template utilizes the client-side object model (CSOM). However, it can be rewritten to utilize the equivalent REST calls.
 1. Open the **app.js** file located in the **scripts** folder.
 2. Delete all of the JavaScript code in the file.
 3. Add the following code to the **app.js** file.

```
(function () {  
    "use strict";  
  
    jQuery(function () {  
  
        jQuery.ajax({  
            url: "../_api/web/currentuser",  
            type: "GET",  
            headers: {  
                "accept": "application/json;odata=verbose",  
            },  
            success: function (data, status, jqXHR) {  
                jQuery("#message").text("Welcome, " + data.d.Title);  
            },  
            error: function (jqXHR, status, message) {  
                jQuery("#message").text(message);  
            }  
        });  
    });  
});  
  
}());
```

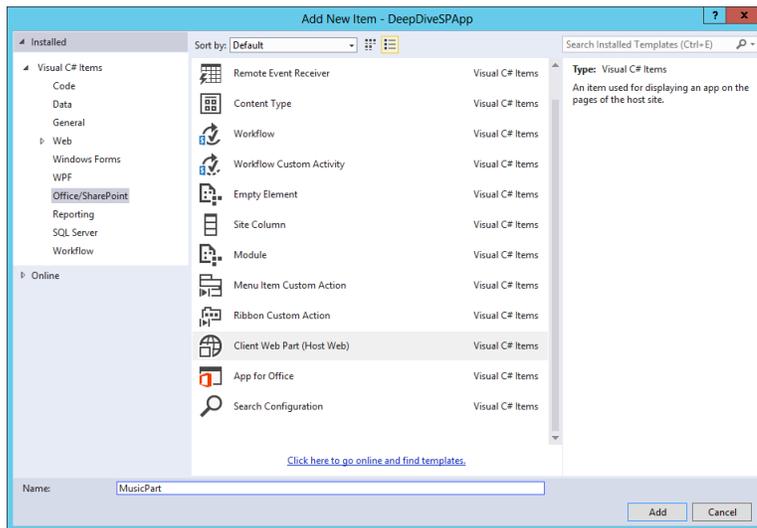
4. Press **F5** to debug the app.
5. Verify that the app launches and greets you.

Exercise 2: Create an App Part

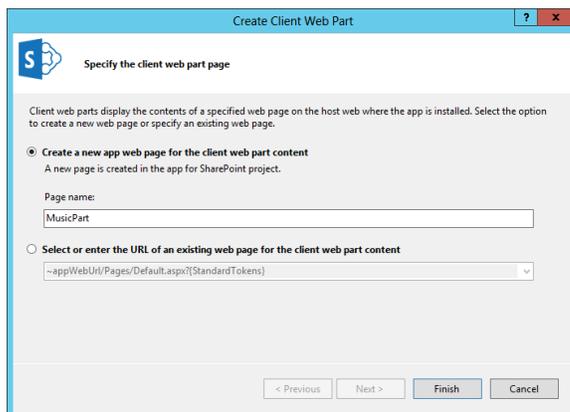
In this exercise you will add an app part to your project for displaying song titles based on a specified artist.

Add the new Client Web Part

1. In **Visual Studio 2013**, right click the **DeepDiveSPApp** project and select **Add/New Item**.
2. In the **New Item** dialog, select **Client Web Part (Host Web)**.
3. Name the new app part **MusicPart** and click **Add**.



4. In the **Specify the Client Web Part Page** dialog, select **Create a New App Web Page for the Client Web Part Content**.
5. Click **Finish**



6. In the **Element.xml** file that describes the Client web part, **replace** the **ClientWebPart** element with the following code.

Deep Dive into SharePoint Hosted Apps

```
<ClientWebPart
  Name="MusicPart"
  Title="Music App Part"
  Description="Displays songs from a specified artist"
  DefaultWidth="300"
  DefaultHeight="200">
  <Content Type="html"
    Src="~appWebUrl/Pages/MusicPart.aspx?{StandardTokens}&Artist=_Artist_" />
  <Properties>
    <Property
      Name="Artist"
      Type="string"
      WebBrowsable="true"
      WebDisplayName="Artist"
      WebDescription="The artist to search"
      WebCategory="Configuration"
      DefaultValue="artist"
      RequiresDesignerPermission="true"
      PersonalizableIsSensitive="false"
      PersonalizationScope="shared"/>
    </Properties>
  </ClientWebPart>
```

Prepare the App Part user interface

1. Open **MusicPart.aspx** for editing.
2. Add the following HTML to the body element for displaying information in the app part.

```
<div>
  <ul id="songList"></ul>
</div>
```

3. Add the following script reference to the head section to include functionality for the app part.

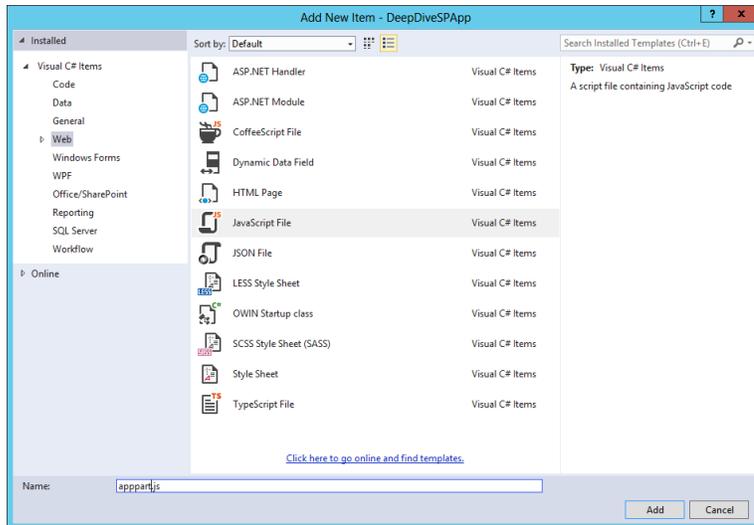
```
<script type="text/javascript" src="~/Scripts/apppart.js"></script>
```

Code the App Part

1. Right click the **scripts** node and select **Add/New Item**.

Deep Dive into SharePoint Hosted Apps

2. In the **Add New Item** dialog, select **Web** and then **JavaScript File**.
3. Name the new file **apppart.js**.
4. Click **Add**.



5. Add* the following code to **apppart.js** to call the **MusicBrainz** service and display songs for the designated artist.

```
(function () {  
    "use strict";  
  
    $(function () {  
  
        var ctx = SP.ClientContext.get_current();  
        var request = new SP.WebRequestInfo();  
  
        request.set_url(  
            "http://www.musicbrainz.org/ws/2/release-  
group?query=artist:" + artist  
        );  
        request.set_method("GET");  
        responseDocument = SP.WebProxy.invoke(ctx, request);  
        ctx.executeQueryAsync(onSuccess, onError);  
  
    });  
  
} ());  
  
var onSuccess = function () {
```

Deep Dive into SharePoint Hosted Apps

```
↪ var xmlDoc = $.parseXML(responseDocument.get_body());
↪ $(xmlDoc).find("release-group").each(function (i) {
↪     var title = $(this).children("title").first().text();
↪     $("#songList").append("<li>" + title + "</li>");
↪ });
↪ }
↪
↪ var onError = function () {
↪     alert("failed!");
↪ }
↪
↪ var getQueryStringParameter = function (p) {
↪     var params =
↪         document.URL.split("?")[1].split("&");
↪     var strParams = "";
↪     for (var i = 0; i < params.length; i = i + 1) {
↪         var singleParam = params[i].split("=");
↪         if (singleParam[0] == p)
↪             return decodeURIComponent(singleParam[1]);
↪     }
↪ }
↪
↪ var artist = getQueryStringParameter("Artist");
↪ var responseDocument = "";
```

Test the App Part

1. Open **AppManifest.xml**
2. Click **Remote Endpoints**
3. Add the endpoint <http://www.musicbrainz.org> and click **Add**

General Permissions Prerequisites Supported Locales Remote Endpoints

Specify one or more remote domains for querying a remote service. The web proxy validates whether the requests issued to the remote domains are declared in the app manifest.

Enter the valid URL of an endpoint to add to the list of endpoints:

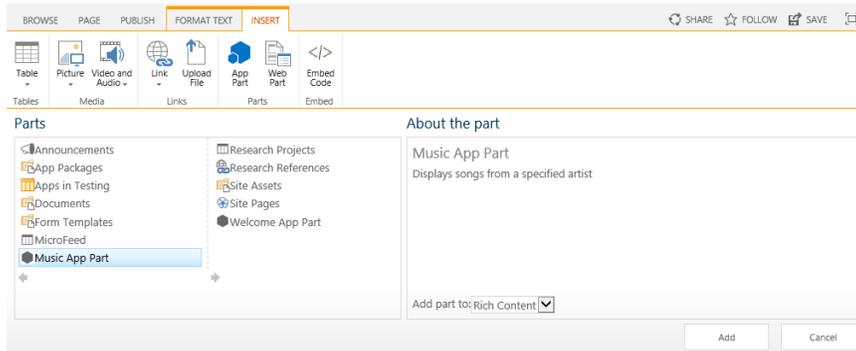
Endpoints:

http://www.musicbrainz.org

4. Press **F5** to start debugging.

Deep Dive into SharePoint Hosted Apps

- When the app launches, navigate away from the app home page to the home page of the **host web**.
- Place the page in edit mode.
- Click **Insert/App Part**.
- Select the **Music App Part** and click **Add**



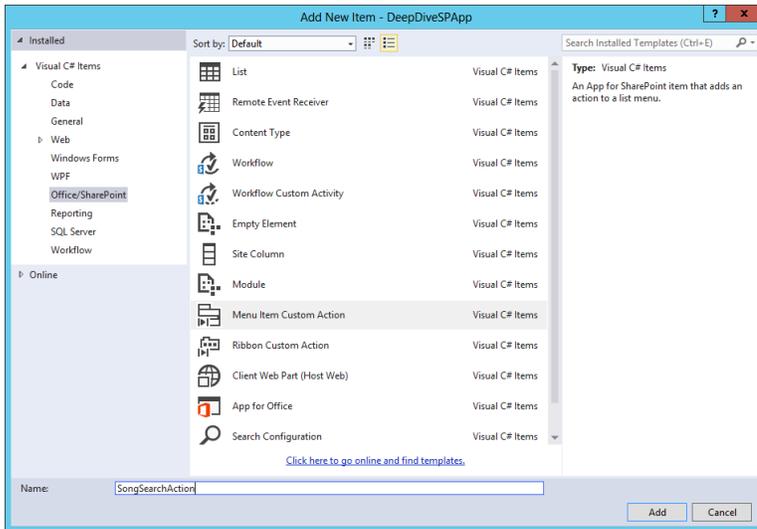
- Hover over the app part and select **Edit Web Part** from the menu.
- In the **Web Part Properties**, change the **Artist** property to a different value.
- Click **OK** and verify that songs appear in the app part.
- Stop debugging.

Exercise 3: Create a Menu Item custom action

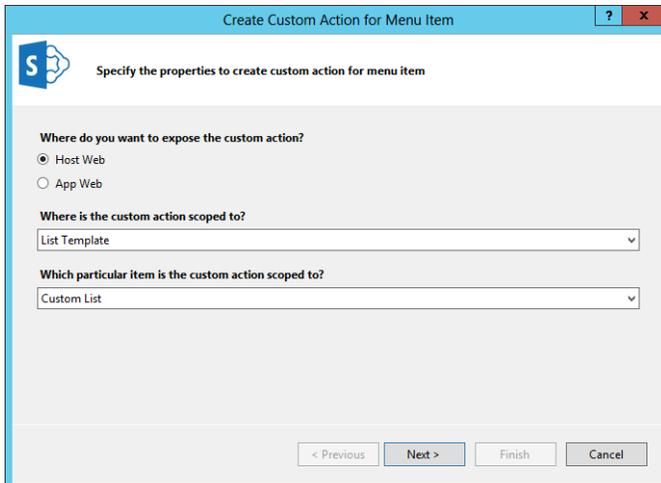
In this exercise you will add a Menu Item custom action to invoke the song search from a SharePoint list.

Add the new Menu Item Custom action

1. In **Visual Studio 2013**, right click the **DeepDiveSPApp** project and select **Add/New Item**.
2. In the **New Item** dialog, select **Menu Item Custom Action**.
3. Name the new app part **SongSearchAction** and click **Add**.

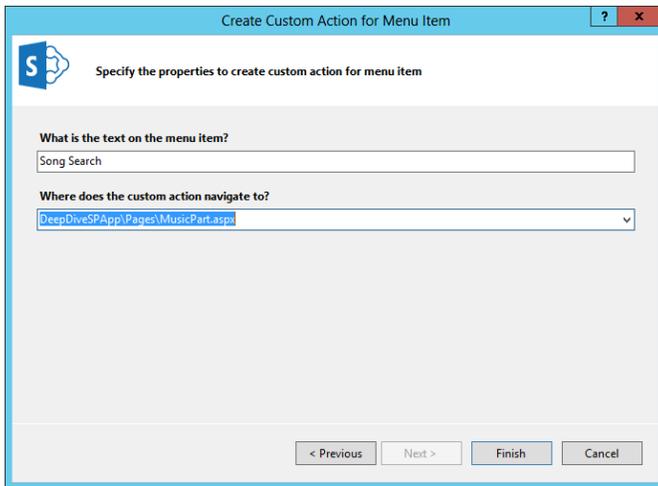


4. In the **Specify the Properties to Create Custom Action for menu Item** dialog
 1. Select **Host Web**
 2. Select **List Template**
 3. Select **Custom List**
 4. Click **Next**



Deep Dive into SharePoint Hosted Apps

5. In the **Specify the Properties to Create Custom Action for menu Item** dialog
 1. Enter **Song Search** as the title
 2. Enter **DeepDiveSPApp\Pages\MusicPart.aspx** as the target page
 3. Click **Finish**



6. Open **MusicPart.aspx** for editing.
7. Add the following script reference to the **head** section just before the **apppart.js** reference.

```
<script type="text/javascript"
  src="/_layouts/15/sp.requestexecutor.js"></script>
```

Code the Menu Item Custom Action

1. Open **apppart.js** for editing.
2. **Add** the following code to the bottom on the file to retrieve an artist name from a custom list selection.

```
var appWebUrl = getQueryStringParameter("SPAppWebUrl");
var hostWebUrl = getQueryStringParameter("SPHostUrl");
var listId = getQueryStringParameter("SPListId");
var listItemId = getQueryStringParameter("SPListItemId");

if (typeof (listId) != "undefined" && typeof (listItemId) !=
  "undefined") {
  listId = listId.substring(1, listId.length - 1);
  var executor = new SP.RequestExecutor(appWebUrl);
  executor.executeAsync({
    url: "../_api/SP.AppContextSite(@target)/web/lists(guid'" +
    listId +
    "')/getItemByStringId('" + listItemId +
```

Deep Dive into SharePoint Hosted Apps

```
↪      "?@target='" + hostWebUrl + "'",
↪      method: "GET",
↪      headers: {
↪          "accept": "application/json;odata=verbose",
↪      },
↪      success: function (data) {
↪          artist = JSON.parse(data.body).d.Title;
↪      },
↪      error: function (data) {
↪          artist = "artist";
↪      }
↪  });
↪ }
```

3. Enclose the call to MusicBranz in a **timeout** function as a simple way to ensure the artist name is retrieved from the list before the call is made. The following code shows how this is done

```
↪      setTimeout(function () {
↪
↪          var ctx = SP.ClientContext.get_current();
↪          var request = new SP.WebRequestInfo();
↪
↪          request.set_url(
↪              "http://www.musicbrainz.org/ws/2/release-
↪          group?query=artist:" + artist
↪          );
↪          request.set_method("GET");
↪          responseDocument = SP.WebProxy.invoke(ctx, request);
↪          ctx.executeQueryAsync(onSuccess, onError);
↪
↪      }, 2000)
```

Test the Menu Item custom action

1. Open **AppManifest.xml**
2. Click **Permissions**
3. Select **Web** for the **Scope**.
4. Select **Read** for the **Right**

Deep Dive into SharePoint Hosted Apps

General Permissions Prerequisites Supported Locales Remote Endpoints

Specify the permissions that your app for SharePoint will request from the user at installation time.

Allow the app to make app-only calls to SharePoint. [Learn more about app authentication policy.](#)

Scope	Permission	Properties
Web	Read	
*		

5. Press **F5** to start debugging.
6. When the app launches, navigate away from the app home page to the home page of the **host web**.
7. On the host web home page, click **Site Contents**.
8. Click **Add an App**.
9. Click **Custom List**.
10. Name the new list **Artists**.
11. Click **Create**

Adding Custom List ×

Pick a name
You can add this app multiple times to your site. Give it a unique name.

Name:

[Advanced Options](#)

12. Add a new artist name to the list.
13. Using the item menu, select **Song Search**.

Congratulations! You have completed investigating SharePoint-Hosted apps.