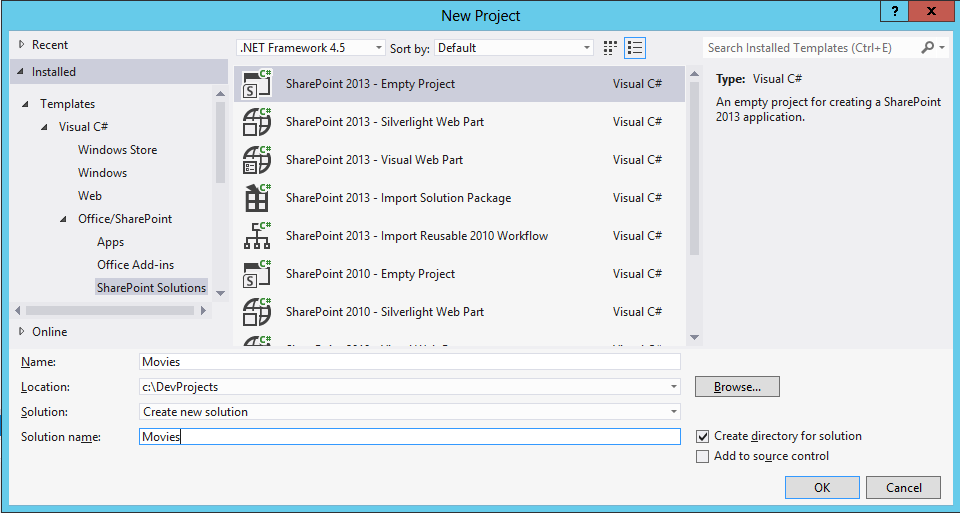
# Demo 1: Using the New List Designer

In this Demo, you will create a sandboxed solution and demo create a new list and adding custom columns using the new list designer. Before beginning the demo, make sure you have a local test site (e.g. <http://contososerver>) and also ensure that the Microsoft SharePoint Foundation Sandboxed Code Service has been started on your VM so that you will be able to test and deploy a sandbox solution.

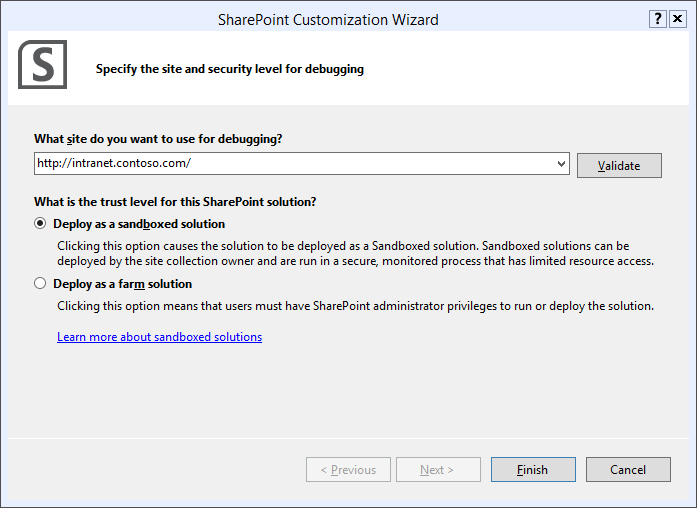
## Step 1 – Create a new sandbox solution

In this Step, you will develop a new sandbox solution in Visual Studio to create two site columns and a custom list.

1. Open Microsoft Visual Studio 2012 and create a new sandbox solution project named **Movies**. Click OK to continue.



1. In the next dialog, add the URL to your test site and select the option to Deploy as a sandbox solution.



* + - * + If you use a host header for your web site, you may need to edit your HOSTS file with an entry for your IP address in order to create a sandboxed solution. For instance:
        + 192.168.0.2 intranet.contoso.com

1. Create a new site column named **Genre**. Make it a Choice column and update the elements.xml file to look like this.

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

<Field

ID="{209e7942-5acf-4afb-8ad2-be026a81bfd4}"

Name="Genre"

DisplayName="Movie Genre"

Type="Choice"

Required="FALSE"

Group="Contoso Site Columns">

<CHOICES>

<CHOICE>Action</CHOICE>

<CHOICE>Comedy</CHOICE>

<CHOICE>Drama</CHOICE>

<CHOICE>Musical</CHOICE>

</CHOICES>

</Field>

</Elements>

1. Create a second site column named **Released** to track what year a movies was released. After creating the site column, update the elements.xml file to look like this.

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

<Field

ID="{2568b09e-1c10-416d-871d-530b4f7df776}"

Name="Released"

DisplayName="Year Released"

Type="Text"

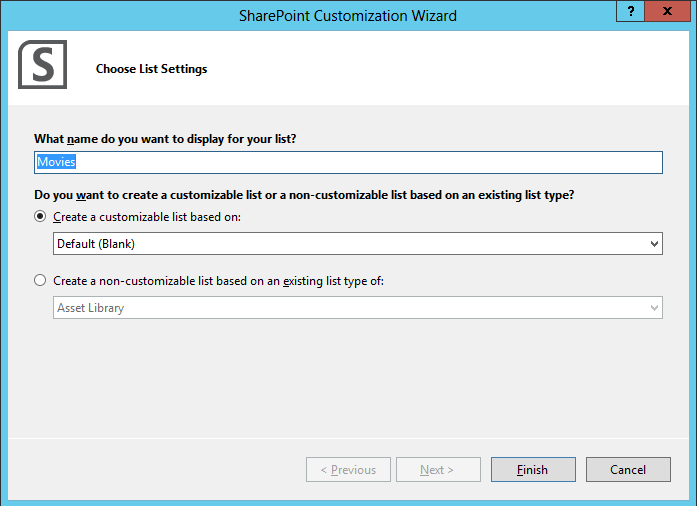
Required="FALSE"

Group="Contoso Site Columns">

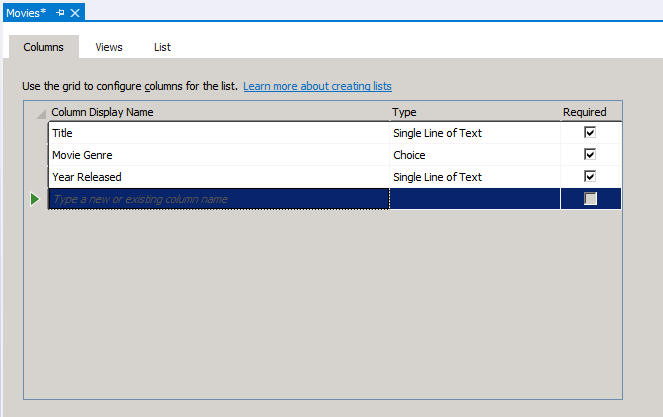
</Field>

</Elements>

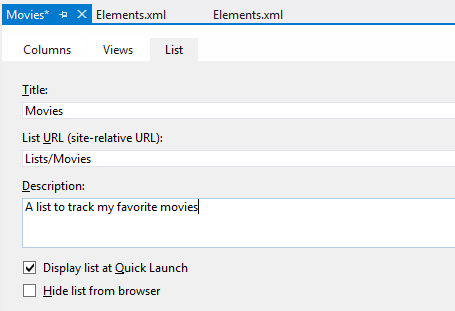
1. Run a test to deploy the sandbox solution. Now, navigate to the test site. Go to the site columns gallery and ensure that the site columns have been created.
2. Return to the Movies project in Visual Studio.
3. Create a new customizable list named **Movies** based on the **Default (Blank)** list type. Click the **Finish** button when you are done.



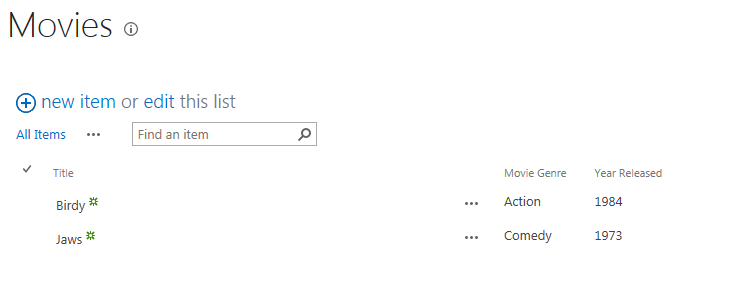
1. After the list has been created, you should see the list designer and the **Columns** tab should be displayed. Add two new columns for the two site columns you added earlier with the display names of **Movie Genre** and **Year Released**.



1. Move to the **Views** tab and show the students what is there. There is no need to change anything on this tab.
2. Move to the **List** tab and update the description.



1. Deploy the **Movies** solution to the test site.
2. Go to the test site and show student the list has been created. Add a few new items using your favorite movies to show students that the list is functional.



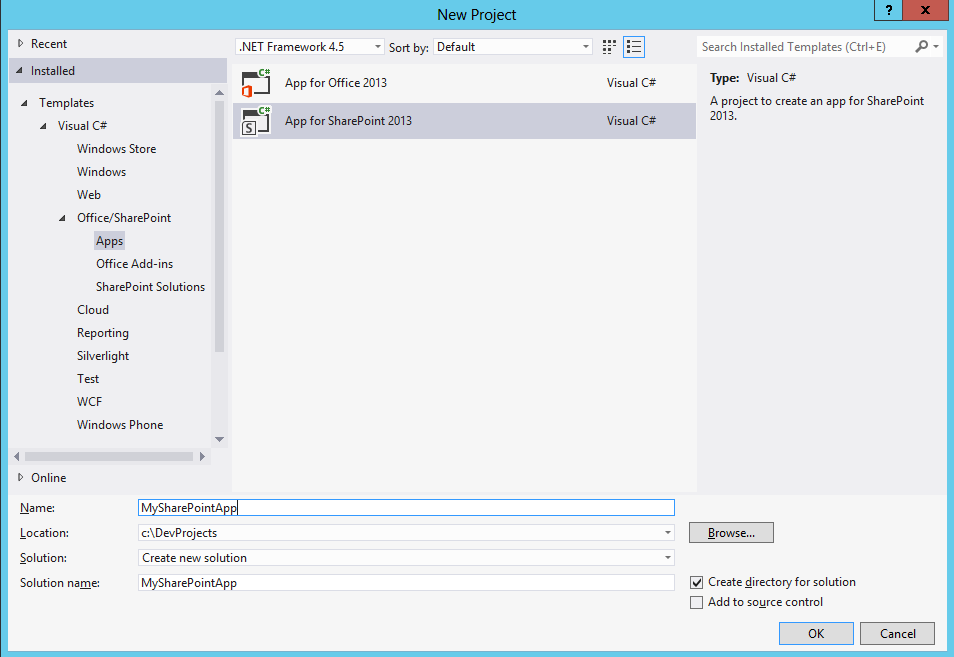
# Demo 2: Creating and Testing a New SharePoint App

In this Demo, you will create and test a simple SharePoint-hosted app.

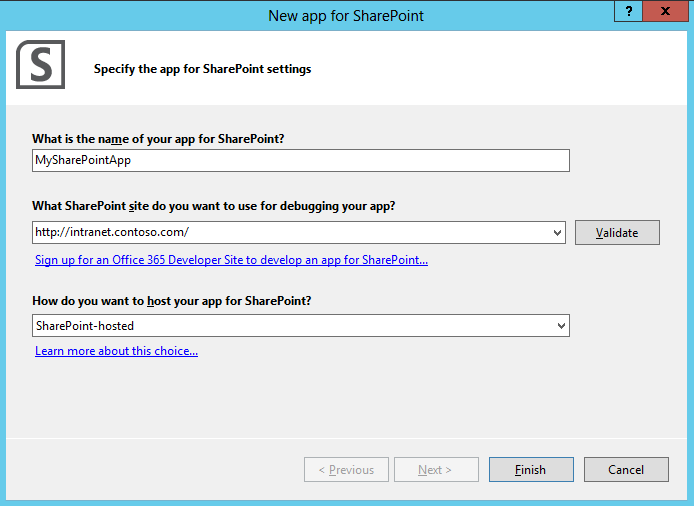
## Step 1 – Create a new SharePoint-hosted app.

In this Step, you will create a new SharePoint-hosted app and make a small number of changes to the code provided.

1. Launch Visual Studio and create a new SharePoint App project.



1. Configure the app project as a SharePoint-hosted app and ensure the test site URL is pointing to a site in your local farm.



1. Once the project has been created, give the students a quick walkthrough of the folders and files that are added to the project using the solution explorer.
2. Open the **AppManifest.xml** file in design view and show students what can be changed inside the manifest using design view.
3. Close the **AppManifest.xml** file in design view and open it in code view just so students see how that is done. Do not make any changes but mention there is sometimes a need to make changes that can only be done in code view.
4. Open **Default.aspx** and step through what is inside. Once you have showed students what’s inside **Default.aspx**, close the file without making any changes.
5. Open the JavaScript file named **App.js**. Remove all the code inside and replace it with the following code.

// This code runs when the DOM is ready

$(document).ready(function () {

SP.SOD.executeFunc('sp.js', 'SP.ClientContext', function () { sharePointReady(); });

});

function sharePointReady() {

$('#message').text('Hello from my SharePoint app');

}

1. Tell students that this syntax is based on jQuery in case they do not already know that. Make a point that jQuery is something they will likely have to learn if they want to develop SharePoint apps.
2. Save your changes to **App.js**.
3. Press the F5 key to begin debugging your app. Show the progress in the output window as the app is installed.
4. When you see the icon for the app on the Site Content page, click it to navigate to the start page of the app. Point out the custom message displayed on the page.
5. Close that browser and then return to Visual Studio and show the progress of the app being uninstalled in the Output window.
6. Make a small change to the text used for the message in App.js and press {F5} again to install and test the app. Make a point that that is the typical cycle of making changes and then testing the app.
7. Make any closing points and return to the lecture.

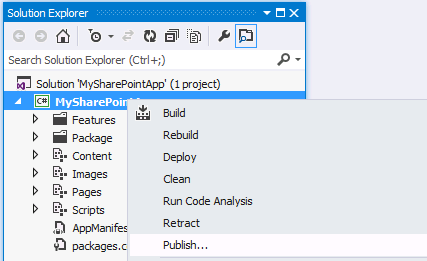
# Demo 3: Creating and Testing a New SharePoint App

In this Demo, you will create and test a simple SharePoint-hosted app.

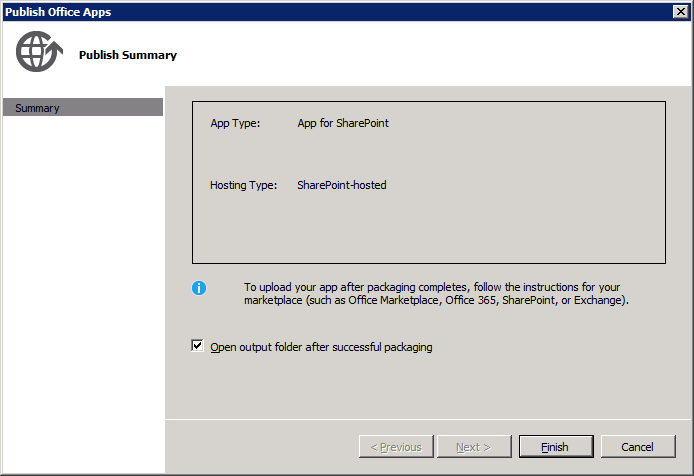
## Step 1 – Inspecting the Application Package for a SharePoint App

In this Step, you will create a new SharePoint-hosted app and make a small number of changes to the code provided.

1. Launch Visual Studio and open the MySharePointApp that you created in the previous demo.
2. Right-click on the project and click the **Publish…** menu command.



1. On the next dialog you can just click finish.



1. You should then see the output folder with a file named **MySharePointApp.app**. Change the file name to **MySharePointApp.app.zip** so Windows recognizes it as a ZIP archive.
2. Double click on the ZIP archive to show students what is inside.

