|  |
| --- |
| Microsoft SharePoint 2013 - Hands-on Lab |
| Visual Studio Tooling for SharePoint 2013 |
| 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

## 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)SharePointTools.

|  |
| --- |
| 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 to create and test site columns
* Learn to use the new list designer
* Learn when direct edits to schema.xml are required
* Learn to create and test site page templates
* Learn write, integrate and test JavaScript code behind site page templates

## Overview of Lab

This lab will give you exposure to the SharePoint Tools that are built into Visual Studio 2013. You will begin by using new features such as a project template item for creating a site column and the new list designer. You will also work with site page templates and writing client-side JavaScript code. While much of the work in other labs for this course will focus on SharePoint App development, this lab will have you working with a Sandboxed Solution so you can focus on new features and client-side development skills without the distraction of the new complexities involved in app development.

## 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. 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. |

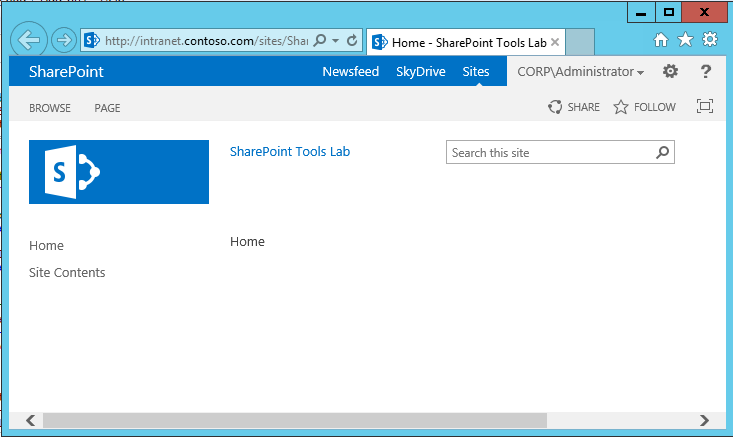
All user accounts in this lab use the password {Supplied by Instructor}.

## Lab Setup Script

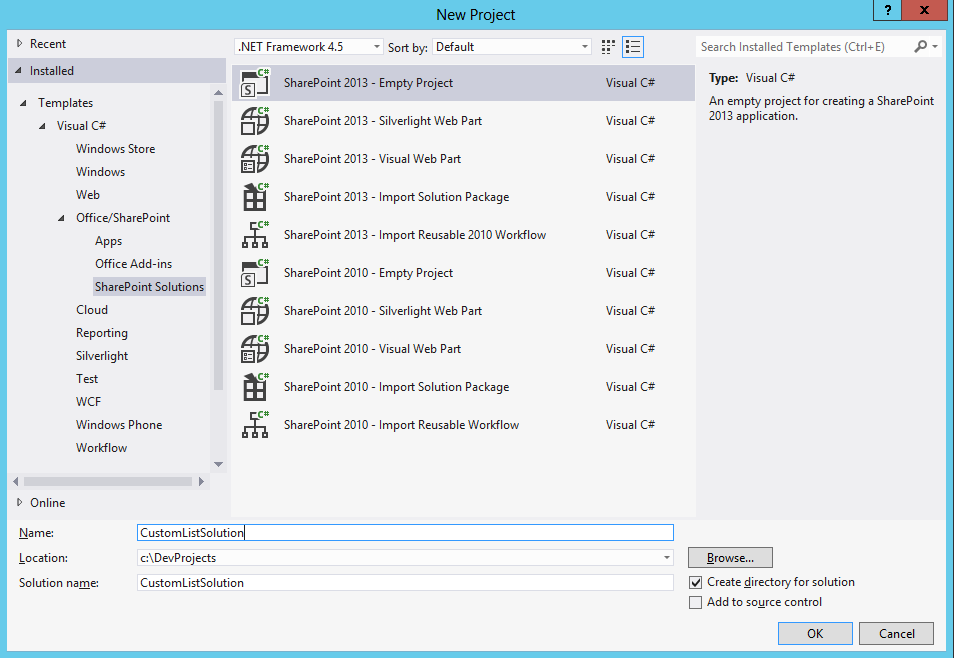
In the files provided with the hands on lab, run the batch file called **SetupModule.bat** by double clicking it. This file will execute a PowerShell script that will create a new site collection at [http://intranet.contoso.com/sites/SharePointTools](http://w15-sp/sites/SharePointTools).

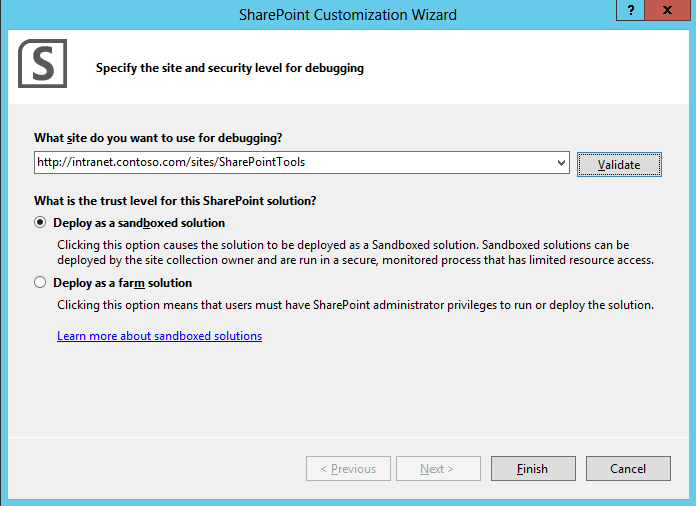
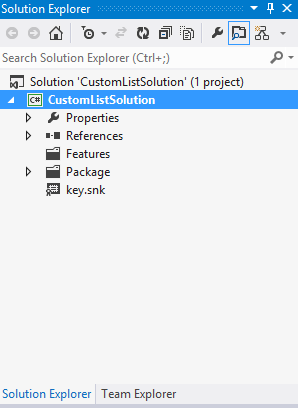
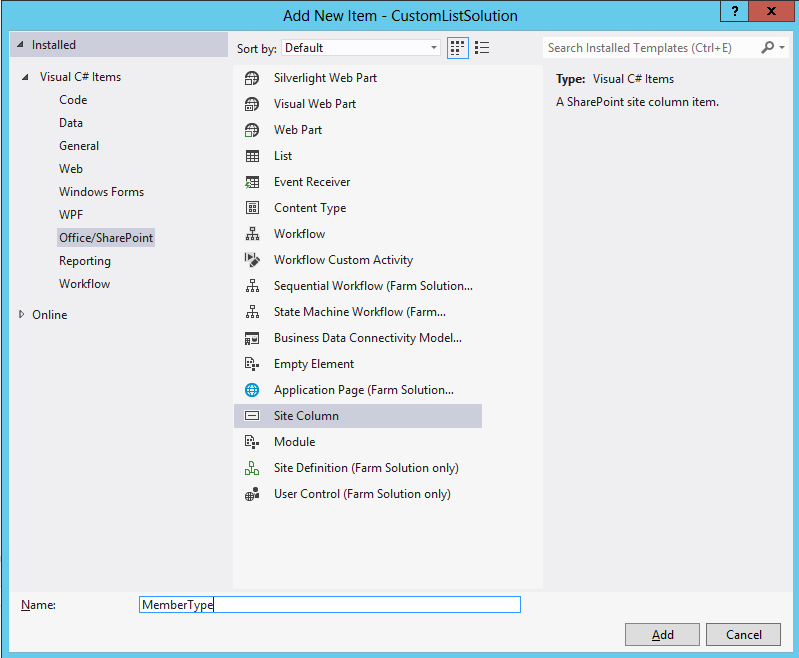
# Exercise 1: Creating a Custom List

* 1. In this exercise, you will work through the basics of creating, testing and debugging a custom list type using the new list designer in Visual Studio 2012.
* Begin this task logged on to **SP** as **CONTOSO\Administrator**
  1. Using Internet Explorer, open the site that you created using the lab setup script at   
     [http://intranet.contoso.com/sites/SharePointTools](http://w15-sp/sites/SharePointTools).



* 1. Launch Visual Studio 2012.
  2. Create a new **SharePoint 2013 – Empty Project** and name this new project **CustomListSolution**. Click OK on the New Project dialog.



* 1. In the SharePoint Customization Wizard, add the URL of the site created earlier and ensure that the option to **Deploy as a sandboxed solution** is selected. Click **Finish**.
     1. 
  2. At this point you should have created a new SharePoint project that you can test by deploying it as a sandboxed solution.
     1. 
  3. Right-click on the CustomListSolution project in the Project Explorer and click the **Add > New Item** command.
  4. Create a new project item from the **Site Column** item template named **MemberType**.
     1. 
  5. When you create a new site column, Visual Studio creates on a node inside your project which contains an **elements.xml** file with a **Field** element to provide you with a starting point for a new site column definition. This starting point includes a new GUID for the ID. Modify the **Field** element for your new site column as shown below to create a choice column with four choices. Note that you can use the GUID that is created and that you do not have to change the GUID to match the one below.

<Field

ID="{dfa867d2-79d0-496d-b379-d35b181a0c06}"

Name="MemberType"

DisplayName="Member Type"

Type="Choice"

Required="True"

Group="Contoso Site Columns">

<CHOICES>

<CHOICE>Full-time Employee</CHOICE>

<CHOICE>Part-time Employee</CHOICE>

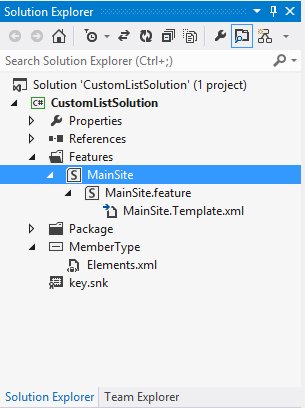
<CHOICE>Intern</CHOICE>

<CHOICE>Contractor</CHOICE>

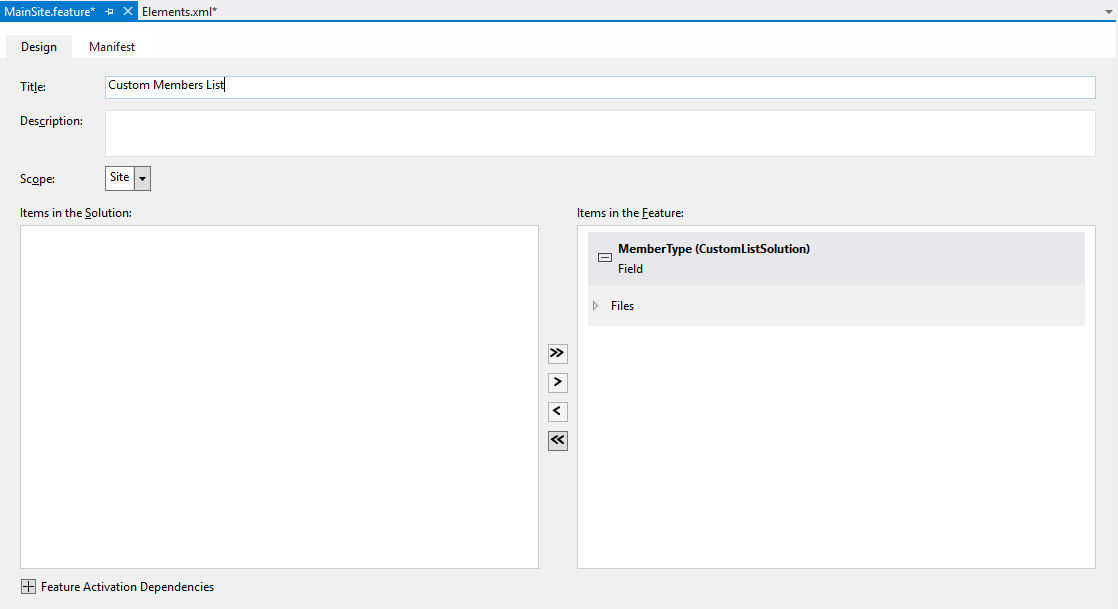
</CHOICES>

</Field>

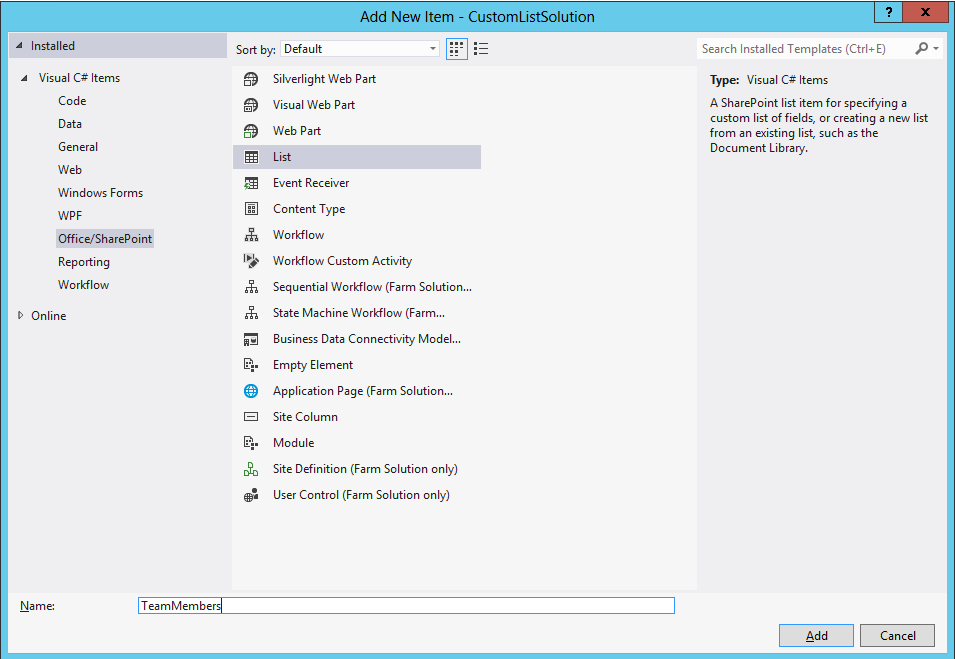
* 1. Note that creation of the Site Column should have also create a new feature that has been named **Feature1**. Right-click and the feature in the Solution Explorer and click the Rename command to rename the feature to **MainSite**.

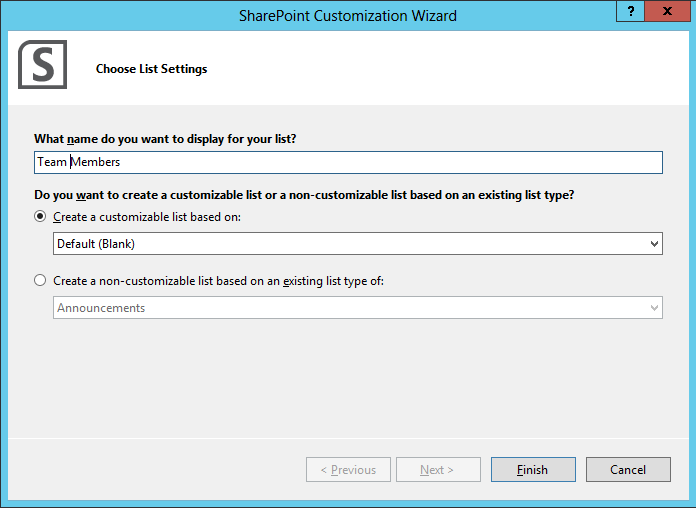
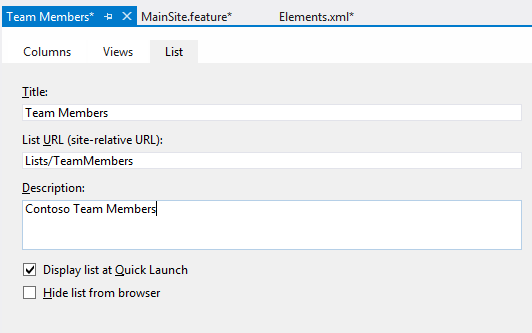
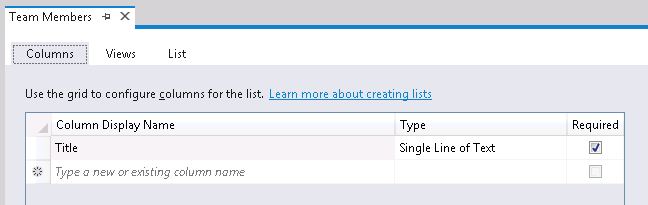
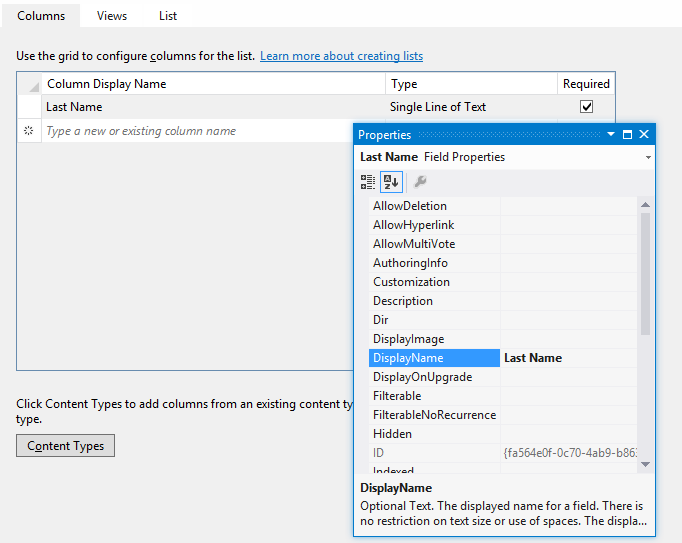
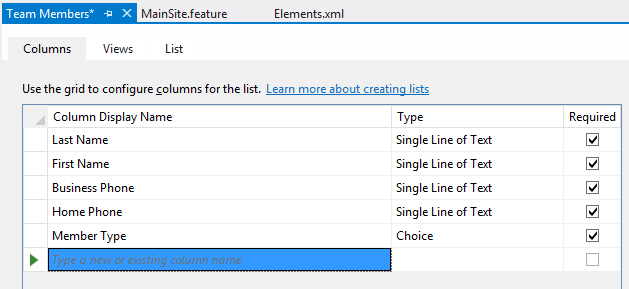


* 1. Double click the **MainSite** feature to display it inside the Feature Designer. Update the feature so it activates at the site collection level by changing the **Scope** property to a value of **Site**. Also modify the **Title** property of feature from its default value to **Custom Members** List make it look more readable.



* 1. Right-click on the **CustomListSolution** project in the Project Explorer and click the **Add > New Item** command. Create a new item from the **List** template named **TeamMembers**. Click the **Add** button to move to the next step.



* 1. On the SharePoint Customization Wizard, Add a display name of **Team Members**. Also ensure you select the option for **Customize the list based on: Default (Blank)**. Click Finish to see the new customizable list in the new **List Designer**.
     1. 
  2. In the List Designer, navigate to the **List** tab. Change the **Title** to **Team Members**. Ensure the list URL is **Lists/TeamMembers** (without a space). Also add a simple description.
     1. 
  3. In the List Designer, navigate to the **Columns** tab. You should see the list has an initial set of columns that just includes the standard **Title** site column.
     1. 
  4. Make sure you have the Title column selected in the List Designer. Now inspect that column in the standard Visual Studio property sheet. You should be able to see it displays the properties for the **Title** site column. Update the **Display Name** property to **Last Name**.
     1. 
  5. Now return to the List Designer and add four more site columns including **First Name**, **Business Phone**, **Home Phone** and **Member Type**. Note that the first three are standard site columns that are included as part of SharePoint Foundation. The fourth site column named **Member Type** is the custom site column you create earlier in this lab exercise. Be sure to configure each of these site columns so that they are required.
     1. 
  6. In this step you need to add two more site columns named **LinkTitle** and **LinkTitleNoMenu**. However, these cannot be added using the List Designer. Instead, you must make a direct edit to the **schema.xml** file for the custom list. Open the **schema.xml** file and locate the element for the **Fields** collection. You should see that the first Field element in the Fields collection is the site column with the name of **Title**.

<List>

<MetaData>

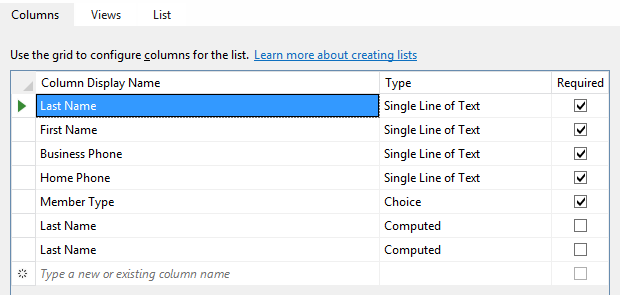
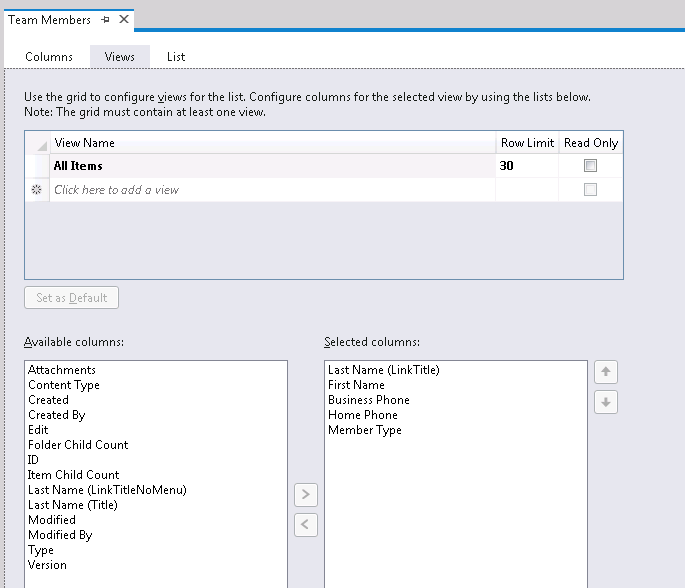
<ContentTypes> ... </ContentTypes>

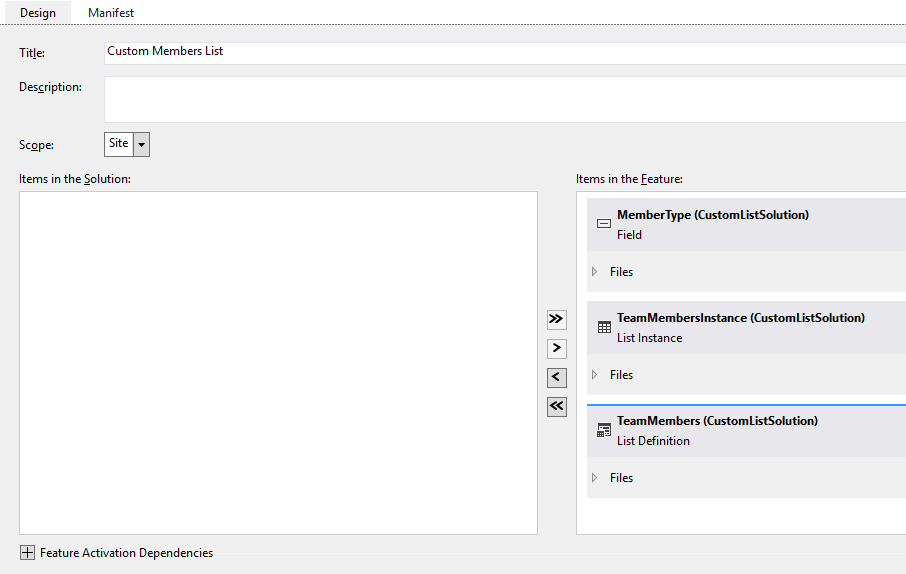
<Fields>

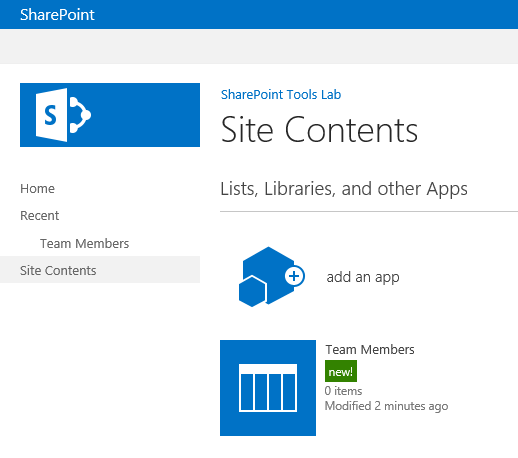
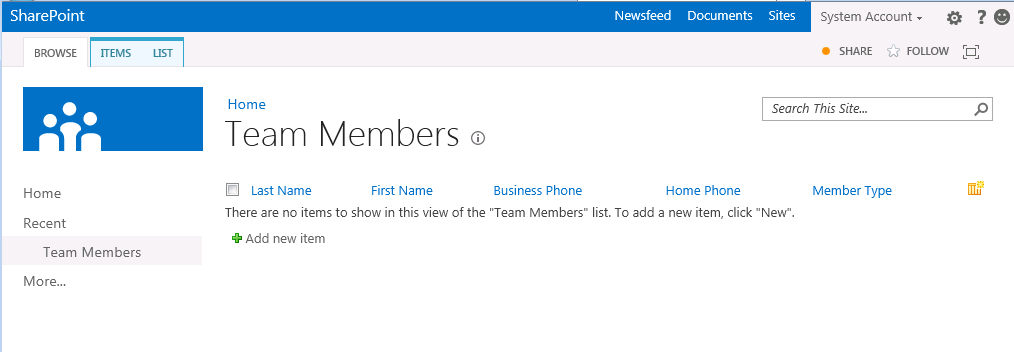
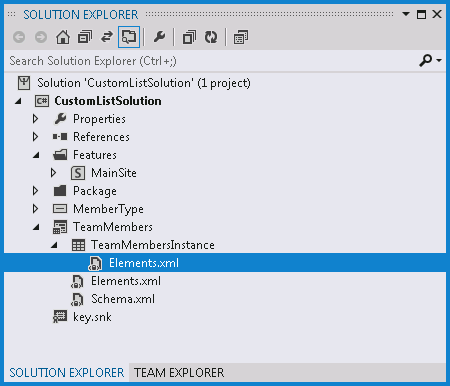
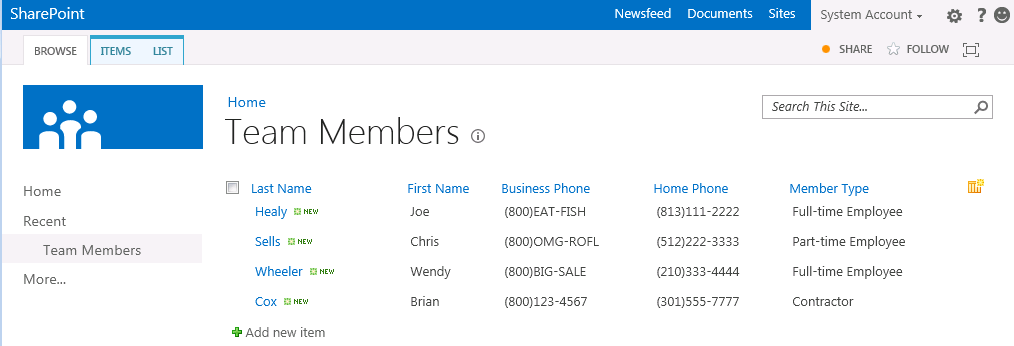
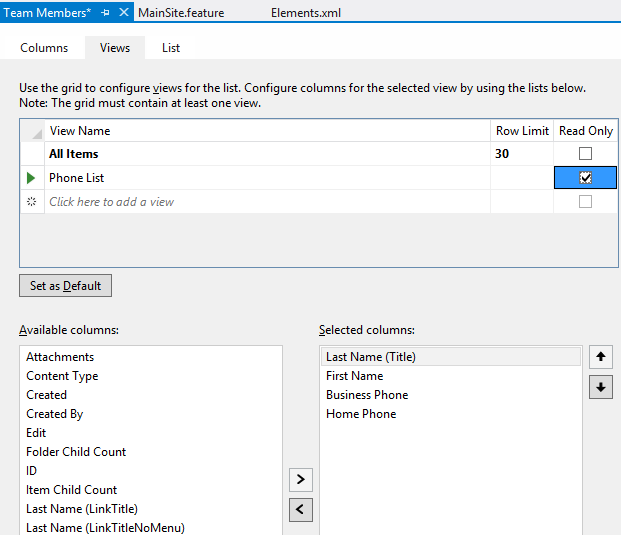
<Field Name="Title" ...

* 1. Inside **schema.xml**, add a line break directly after the Field element which defines the **Title** site column. Add the following Field elements to add the site columns LinkTitle and LinkTitleNoMenu and to configure both of them with a Display Name of Last Name. If you would rather not type all this in, you can copy and paste the XML code from the file inside the folder for this lab named **AddSiteColumns.txt**.

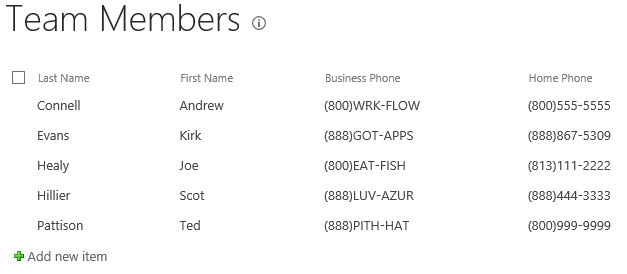
<Field  
 ID="{82642ec8-ef9b-478f-acf9-31f7d45fbc31}"   
 Name="LinkTitle"   
 DisplayName="Last Name"   
 Type="Computed" />  
  
<Field   
 ID="{bc91a437-52e7-49e1-8c4e-4698904b2b6d}"   
 Name="LinkTitleNoMenu"   
 DisplayName="Last Name"   
 Type="Computed" />

* 1. Save your changes to **schema.xml** and close the file. Inside the Project Explorer, double-click on the top-level node for the custom list to display the list in the List Designer again. You should now see the two site columns that you just added.
     1. 
  2. In the List Designer, navigate to the **Views** tab. You should see there is a single view which is named **All Items**. From the Selected columns list below, ensure the site column named **Attachments** does not appear in the **Selected columns** list.
     1. 
  3. Double-click the MainSite feature and make sure the **MemberType**, **TeamMembersInstance**, and **TeamMembers** items are included in the feature.



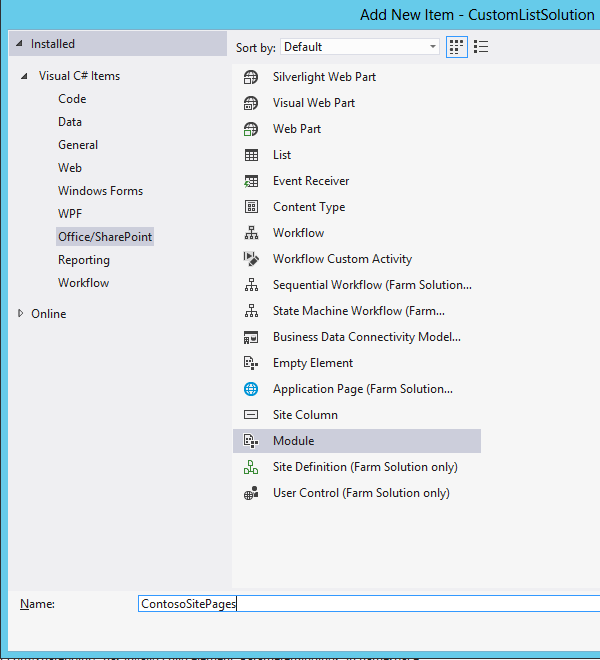
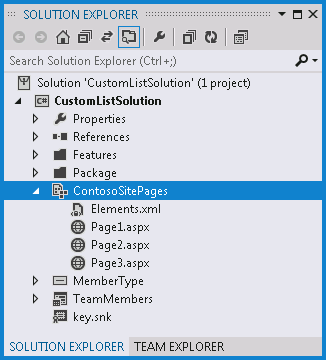
* 1. Now it is time to deploy the solution and test your work. Rick-click on the top-level project node inside the Solution Explorer and click the **Deploy** command. This should build the project into a .WSP file and then upload the .wsp file to the target site collection for your test site. After that, Visual Studio will activate the sandboxed solution which should activate the **MainSite** feature and create a new instance of your custom list. Go to the Internet Explorer and refresh the page showing the test site and verify you can see a link to the **Team Members** list.
     1. 
  2. Click on the link to the Team Members list and see what it looks like. Add one or two new items to the list to test the user experience of your custom list. 
  3. Note that each time you run the Deploy with your project, Visual Studio will delete the existing list instance and create a new one. Therefore any content you add will be deleted during your testing. Therefore, you might want to add some default content to your custom list. You will accomplish this over the next few steps
  4. Inside the node for the custom list type named **TeamMembers**, locate the child node named **TeamMembersInstance** which contains a file named **elements.xml.**
     1. 
  5. Open the **elements.xml** file from inside the **TeamMembersInstance** node and inspect the **ListInstance** element inside.
     1. <Elements xmlns="http://schemas.microsoft.com/sharepoint/">
     2. <ListInstance Title="Team Members" ... >
     3. <!-- add XML here to add default items -->
     4. </ListInstance>
     5. </Elements>
  6. You can add a default item to the list with the following code. If you would rather not type in all this XML, you can alternatively copy and paste the contents of the text file named **DefaultMemberItems.txt** inside the **ListInstance** element to create several new Members each time the list is recreated.
     1. <ListInstance Title="Team Members" ... >
     2. <Data>
     3. <Rows>
     4. <Row>
     5. <Field Name="Title">Healy</Field>
     6. <Field Name="FirstName">Joe</Field>
     7. <Field Name="WorkPhone">(800)EAT-FISH</Field>
     8. <Field Name="HomePhone">(813)111-2222</Field>
     9. <Field Name="MemberType">Full-time Employee</Field>
     10. </Row>
     11. </Rows>
     12. </Data>
     13. </ListInstance>
  7. Test your work. You should be able to deploy your project from Visual Studio and then refresh the page with the Team Members in the Internet Explorer and see you default items.
     1. 
  8. Return to the **Views** tab in the List Designer. Create a new view named **Phone List**. Create the new view to include **Last Name**, **First Name** and **Business Phone** and **Home Phone**. Save your work when you are done. Make this view read-only so that users cannot make changes to this view.
     1. 
  9. The next step will involve adding custom sorting behavior which will require making a manual edit to the **schema.xml** file. Open schema.xml in code view and find the View element for the **Phone List** view. Add an **OrderBy** clause to sort by the Title column which will have the effect of sorting by the member’s last names.
     1. <View BaseViewID="2" Name="ee64ba2b-80ef-4bba-a850-753c6220d48b"   
         DisplayName="Phone List" Type="HTML" ReadOnly="TRUE"   
         WebPartZoneID="Main" Url="Phone List.aspx">
     2. <RowLimit>50</RowLimit>
     3. <ViewFields>
     4. <FieldRef Name="Title" />
     5. <FieldRef Name="FirstName" />
     6. <FieldRef Name="WorkPhone" />
     7. <FieldRef Name="HomePhone" />
     8. </ViewFields>
     9. **<Query>**
     10. **<OrderBy>**
     11. **<FieldRef Name="Title"/>**
     12. **</OrderBy>**
     13. **</Query>**
     14. <Toolbar Type="Standard" />
     15. <XslLink Default="TRUE">main.xsl</XslLink>

</View>

* 1. Test your work. You should be able to deploy your project from Visual Studio and then refresh the page with the Team Members in the Internet Explorer. Change the view from All Items to Phone List. You should be able to see that the view has the correct set of columns and the correct sorting.
     1. 

# Exercise 2: Creating and Editing Site Pages

In this exercise you will continue working with the sandboxed solution project named **CustomListSolution** that you created in the previous exercise. You will extend this project by adding page templates to create site pages in your solution.

* 1. Make sure the **CustomListSolution** project open is open in Visual Studio.
  2. Right-click on the **CustomListSolution** project in the Project Explorer and click the **Add > New Item** command. Create a new project item using the Module template and give it a name of **ContosoSitePages**.
     + 1. 
  3. After the **ContosoSitePages** module is created, its folder will initially contain the **elements.xml** file and a text file named **Sample.txt**. Delete **Sample.txt**.
  4. Inside the **StarterFiles** folder for this lab, you will find three page templates named **Page1.aspx**, **Page2.aspx** and **Page3.aspx**. Add all three of these page templates into the **ContosoSitePages** module.
     + 1. 
  5. Open each of these page templates and take a moment to inspect the code in each one.
  6. Add a feature receiver to add node to the TopNav bar.
     1. Right-click on the node for the **MainSite** feature and click **Add Event Receiver**.
     2. Add methods for **FeatureActivated** and **FeatureDeactivating**.
     3. Add a using statement for the **Microsoft.SharePoint.Navigation** namespace.
     4. Add an implementation for **FeatureActivated** to add TopNav links for the three pages.

public override void FeatureActivated(

SPFeatureReceiverProperties properties) {

SPSite siteCollection = (SPSite)properties.Feature.Parent;

if (siteCollection != null) {

SPWeb site = siteCollection.RootWeb;

// create dropdown menu for custom site pages

SPNavigationNodeCollection topNav =

site.Navigation.TopNavigationBar;

topNav.AddAsLast(

new SPNavigationNode("Page 1", "ContosoSitePages/Page1.aspx"));

topNav.AddAsLast(

new SPNavigationNode("Page 2", "ContosoSitePages/Page2.aspx"));

topNav.AddAsLast(

new SPNavigationNode("Page 3", "ContosoSitePages/Page3.aspx"));

}

}

* 1. Add an implementation of FeatureDeactivating to delete the folder into which the site pages were created and to delete the custom links on Top Navigation.

public override void FeatureDeactivating(SPFeatureReceiverProperties properties) {

SPSite siteCollection = (SPSite)properties.Feature.Parent;

SPWeb site = siteCollection.RootWeb;

try {

// delete folder of site pages provisioned during activation

SPFolder sitePagesFolder = site.GetFolder("ContosoSitePages");

sitePagesFolder.Delete();

}

catch { }

SPNavigationNodeCollection topNav =

site.Navigation.TopNavigationBar;

for (int i = topNav.Count - 1; i >= 0; i--) {

if (topNav[i].Url.Contains("ContosoSitePages")) {

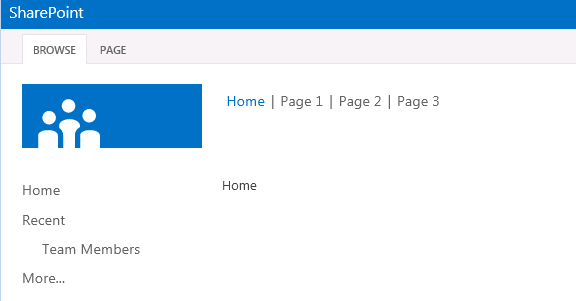
// delete node

topNav[i].Delete();

}

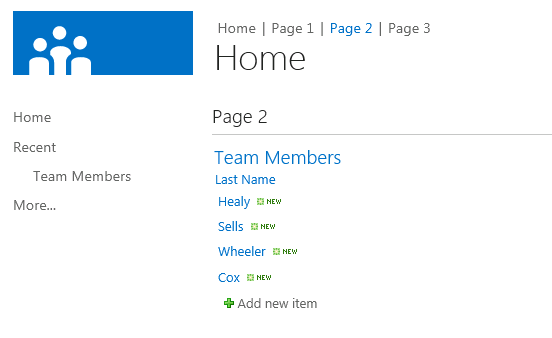
}

}

* 1. Test your work:
     1. Press the **[F5]** to start the debugger which will take you to the home page of the test site. You should see links that allow you to navigate to any of the three new site pages.
        1. 
     2. Click on each link and make sure you can successfully navigate to each page.
     3. Stop the debugger and return to Visual Studio.
  2. Now you will add some CSS style rules to Page1.aspx to make it look better.
     1. Open the source file for the page template named Page1.aspx.
     2. Add a new content tag for PlaceHolderAdditionalPageHead.
     3. Add some CSS styles to make Page1.aspx look better. You can copy and paste the text for these styles from Page1\_Styles\_Starter.txt in [[LAB FILES]]\StarterFiles folder.
     4. <asp:Content ContentPlaceHolderId="PlaceHolderAdditionalPageHead" runat="server">
     5. <style type="text/css">
     6. #MSO\_ContentTable {padding-top: 2px;padding-left:8px;}
     7. .s4-specialNavLinkList a.s4-rcycl {display:none;}
     8. .s4-specialNavLinkList LI {min-height:0px;}
     9. h2 {color:Blue;}
     10. p {color:Green;}
     11. </style>
     12. </asp:Content>
     13. Start the debugger and examine Page1.aspx in the test site test your work. You should see that the styles you have added effect what the page looks like. Feel free to experiment by adding other CSS style rules to this page.
  3. Prepopulate Page2.aspx with a Web Part which displays the Team Members list.
     1. Examine the page template named Page2.aspx. This is a simple starting point for a Web Part page. Currently, however, the page instance for Page2.aspx is created with no Web Parts on it.
     2. Open the elements.xml file for the **ContosoSitePages** Module. Locate the File element used to provision to page instance named Page2.aspx. Now open the File element so that you can add AllUserWebPart elements inside in the next step.
     3. <?xml version="1.0" encoding="utf-8"?>
     4. <Elements xmlns="http://schemas.microsoft.com/sharepoint/">
     5. <Module Name="ContosoSitePages">
     7. <File Path="ContosoSitePages\Page1.aspx" Url="ContosoSitePages/Page1.aspx" />
     9. <File Path="ContosoSitePages\Page2.aspx" Url="ContosoSitePages/Page2.aspx" >
     10. <!-- TODO: Add AllUsersWebPart elements here -->
     11. </File>
     13. <File Path="ContosoSitePages\Page3.aspx" Url="ContosoSitePages/Page3.aspx" />
     15. </Module>
     16. </Elements>
     17. Locate the starter file named AllUsersWebPart\_Starter.txt in the StarterFiles folder. Open the file up and inspected the two AllUserWebPart elements inside.
     18. Copy and paste the contents of its content of AllUsersWebPart\_Starter.txt into the File element for Page2.aspx.



* 1. Now, test your work. Start the debugger and **navigate** to Page2.aspx. You should now see a Web Part on the page showing the Team Members list.



* + 1. Stop the debugger and return to Visual Studio.
  1. Work with the JavaScript used in Page3.aspx.
     1. Open the page template for Page3.aspx. Examine the page to see how it defines JavaScript methods and wire up client-side event handlers to HTML controls defined by the DOM.
     2. Start the debugger and test Page3.aspx. When you click the two command button on the page, you should see their code execute.
  2. If you still have time, experiment by modifying the JavaScript in the two methods named MyHandler and MyOtherHandler.

Summary

* 1. This lab gave you a chance to work with the SharePoint Tools that are built into Visual Studio 2012. You created a site column and you gained experience working with the new list designer. You also worked with Visual Studio to add site page templates to a SharePoint solution.

# 