Creating SharePoint Lists and Event Receivers

Lab Time: 45 minutes

Lab Folder: C:\Student\Modules\08_ListsAndEvents\Lab

Lab Overview: In this lab you will create different types of lists and events in SharePoint 2016. First, you will create a site column and a content type and then use them to create a custom list. Next, you will learn how to create server-side events in SharePoint 2016, the traditional type of events that run on the SharePoint server. The final two exercises focus on creating remote event receivers and see how you can apply these to both app lifecycle events and list-based events.

Exercise 1: Setup Lab Environment

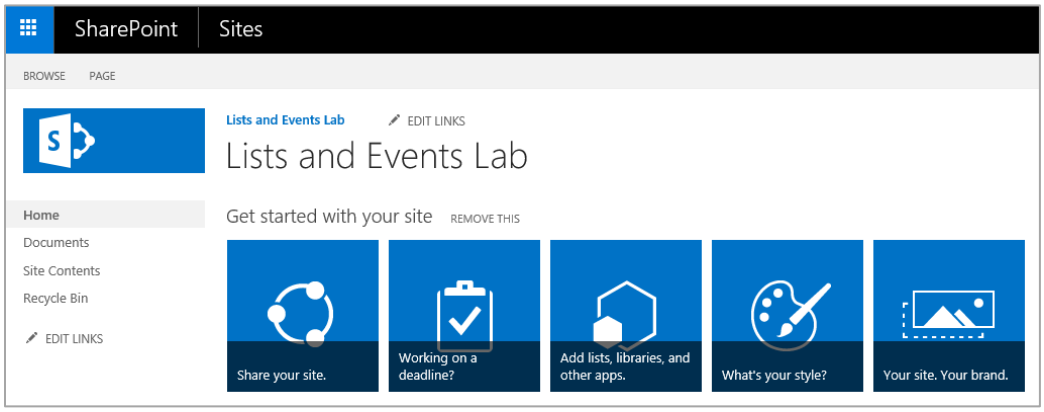
In this exercise you will setup your environment.

All exercises in this lab assume you will work in a new site collection, https://ListsAndEvents.wingtip.com.

1. Ensure you are logged into the **WingtipServer** server as **WINGTIP\Administrator**.
2. Setup a new site collection for this lab:
   a) In Windows Explorer, locate the PowerShell script located at the following path.

   ```
   C:\Student\Modules\08_ListsAndEvents\Lab\SetupLab.ps1
   ```

   b) Right-click on the file named SetupLab.ps1 and select the **Run with PowerShell** command.
   c) When the script completes, it will launch a new browser and navigate to the new test site collection for this lab.

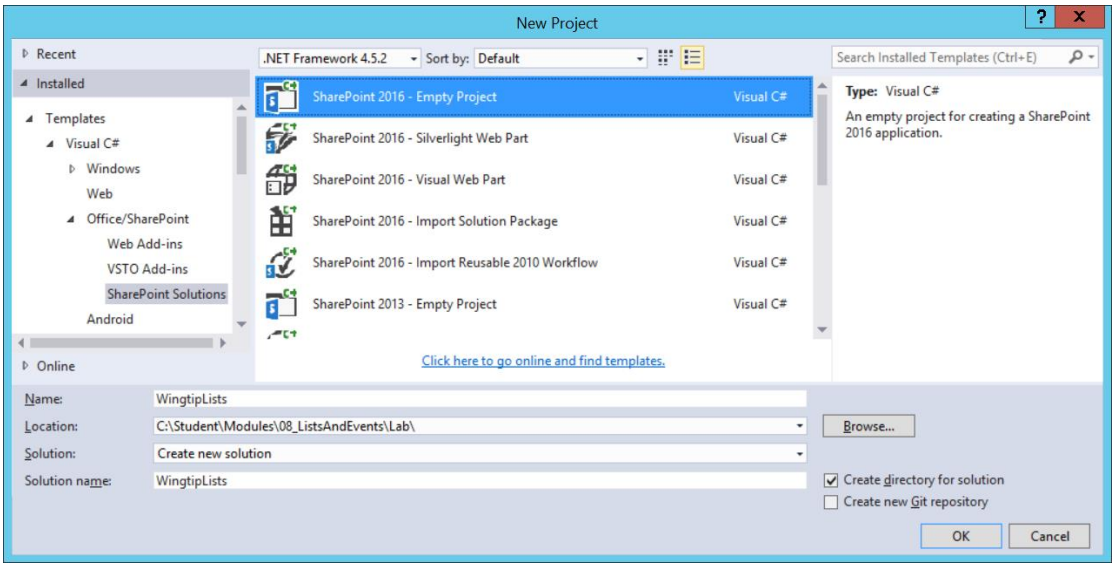


   d) Close the PowerShell console window.

Exercise 2: Creating Site Columns & Content Types

In this exercise you will create a few new site columns and content types that leverage these site columns using the Visual Studio SharePoint Tools. You will create a content type for managing contracts.

1. Create a new project in Visual Studio:
   a) Launch **Visual Studio** if it is not already started.
   b) Select **File → New → Project**.
   c) In the **New Project** dialog:
      i) Select the **SharePoint 2016 – Empty Project** template under the **Templates → Visual C# → Office/SharePoint → SharePoint Solutions** section.
      ii) **Name**: WingtipLists



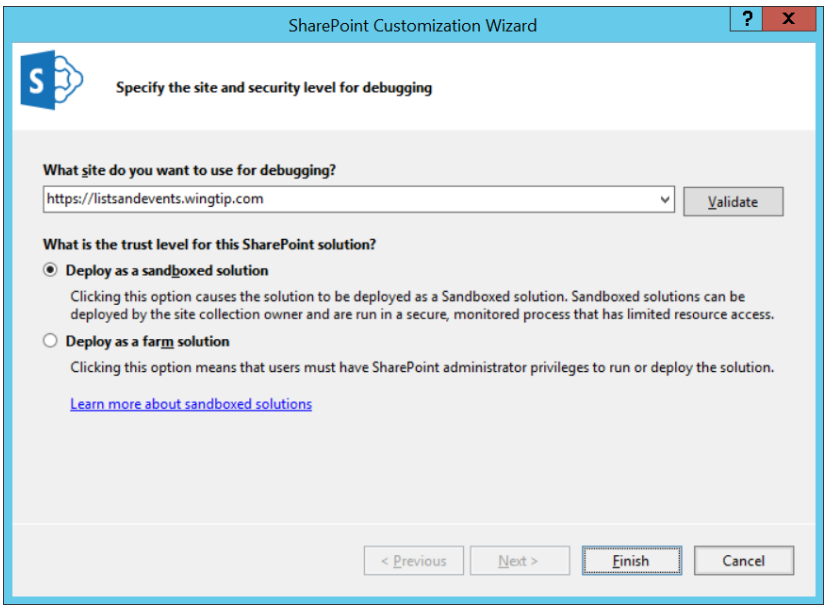
d) Click **Ok** to create the project.

e) In the **SharePoint Customization Wizard**, use the following values to complete the wizard.

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

ii) **What is the trust level for this SharePoint solution:** Deploy as sandboxed solution

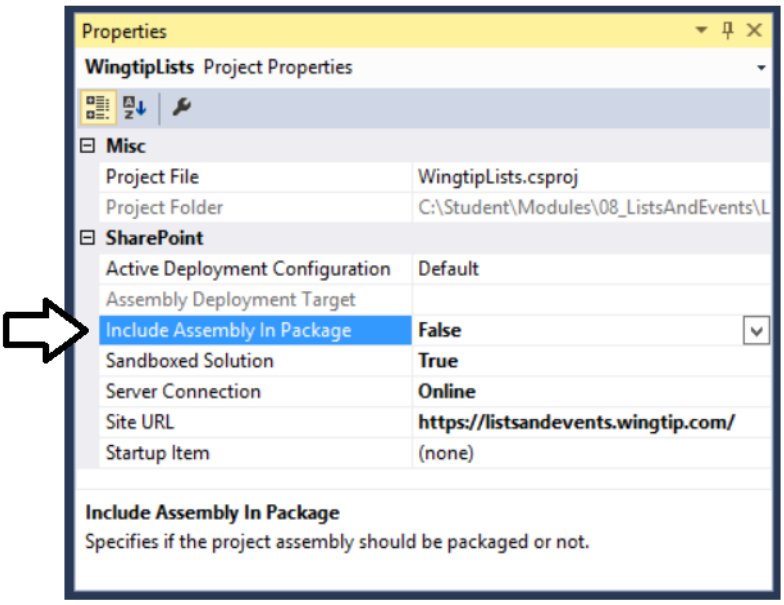
iii) Click **Finish** to create the new project.



Note that while this project has been configured as a Sandbox Solution, it will not contain any .NET code but instead only features and declarative XML elements. This means it will be a no-code sandbox solution (NCSS) which is still supported by Microsoft. A main benefit of testing the project as a NCSS is that Visual Studio will not have to perform an IISRESET operation each time you deploy as it must do when testing a farm solution. This means it is faster when you are constantly deploying and testing the CAML code for your declarative XML elements.

2. After creating a SharePoint solution project for the no-code sandbox solution, it is recommended that you declare your intentions by configuring the project output so it doesn't include a .NET DLL. Accomplish this by selecting the top-level node for the

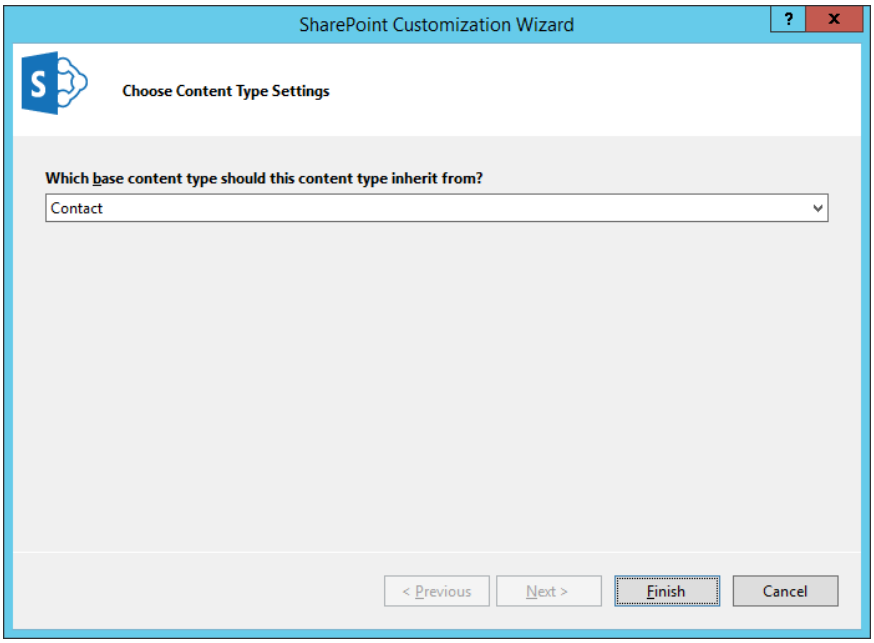
WingtipLists project in the Solution Explorer and then examining its project properties in the Visual Studio Properties Window. Locate the project property named **Include Assembly in Package** and set its value to **False**.



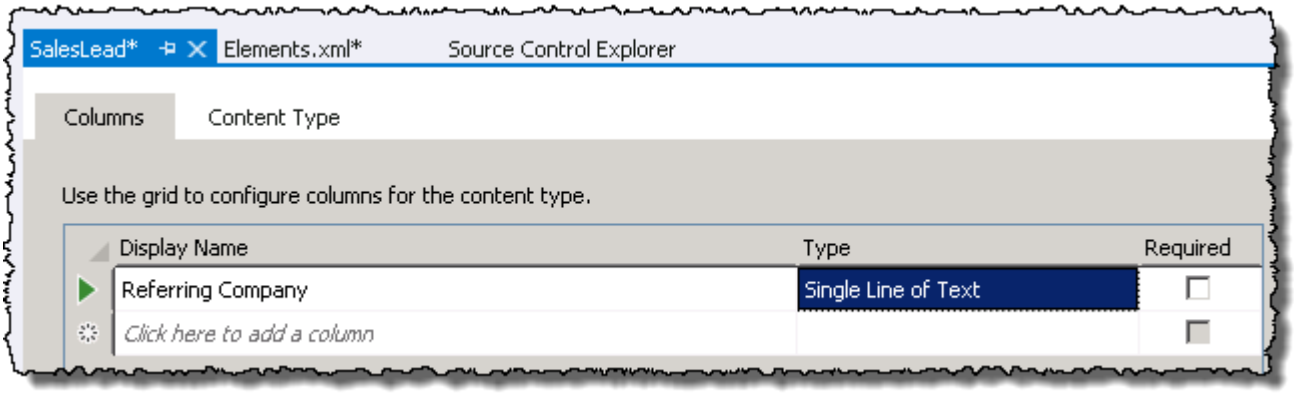
3. Add a new site column:
   a) Using the **Solution Explorer** tool window, right-click the **WingtipLists** project and select **Add → New Item**.
   b) In the Add New Item dialog, select the Site Column template from the Visual C# Items → Office / SharePoint category.
   c) Give the new site column as name of **ReferringCompany**.
   d) Click **Add**.
   e) Visual Studio will stub out the XML for the new site column. Make one change to it by changing the **Group** attribute to **Wingtip Site Columns**.

```
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <Field
       ID="{96a5fce6-fbca-4938-8139-f691025cda4a}"
       Name="ReferringCompany"
       DisplayName="Referring Company"
       Type="Text"
       Required="FALSE"
       Group="Wingtip Site Columns">
  </Field>
</Elements>
```

   f) Save and close the **Elemets.xml** file for the new **ReferringCompany** site column.
4. Add a new content type:
   a) Using the **Solution Explorer** tool window, right-click the **WingtipLists** project and select **Add → New Item**.
   b) In the **Add New Item** dialog, select the Content Type template from the Visual C# Items → Office / SharePoint category.
   c) Give the new content type a name of **SalesLead**.
   d) Click **Add**.
   e) Next, you will be prompted by the **Choose Content Type Settings** page of the **SharePoint Customization Wizard** dialog. Select a base content type of **Contact** and click **Finish**.



5. At this point, Visual Studio should display the **SalesLead** content type in the content type designer.
6. Add a new column for the **ReferringCompany** site column.



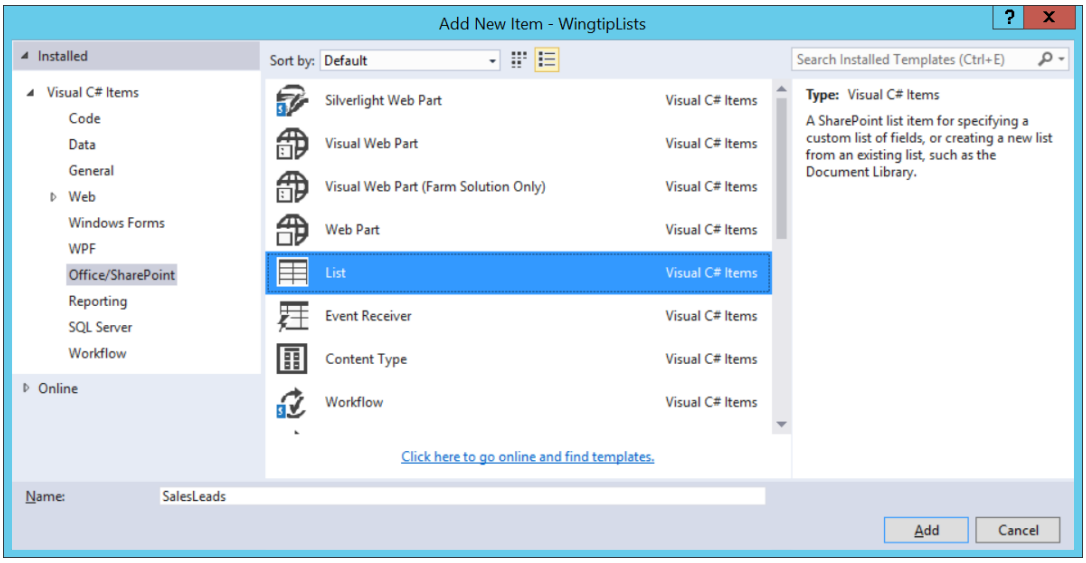
7. Save all changes: **File → Save All**.

In this exercise you created new site columns and content types that leveraged these site columns using the Visual Studio SharePoint Tools. In the next exercise you will see how to create a new list using this content type.

Exercise 3: Creating Lists with Visual Studio

In this exercise you will continue working with the **WingtipLists** project you created in the previous exercise by creating a new SharePoint list using the site column and content type you have already created.

8. Add a new list to the project:
   a) Using the **Solution Explorer** tool window, right-click the **WingtipLists** project and select **Add → New Item**.
      i) In the **Add New Item** dialog, select the **List** template from the **Visual C# Items → Office / SharePoint** category.
      ii) **Name:** References.
      iii) Click **Add**.



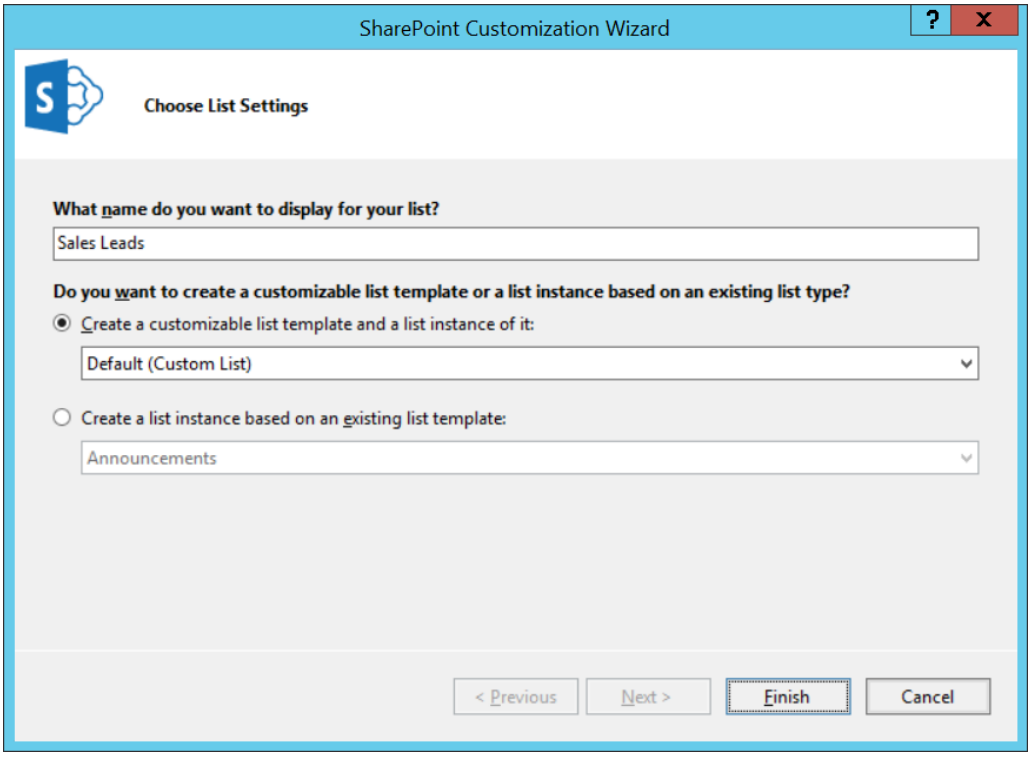
b) In the **SharePoint Customization Wizard** dialog, in **Choose List Settings**, use the following values to complete the dialog:

i) **What name do you want to display for your list?** Sales Leads

ii) Do you want to create a customizable list or a non-customizable list based on an existing list type?

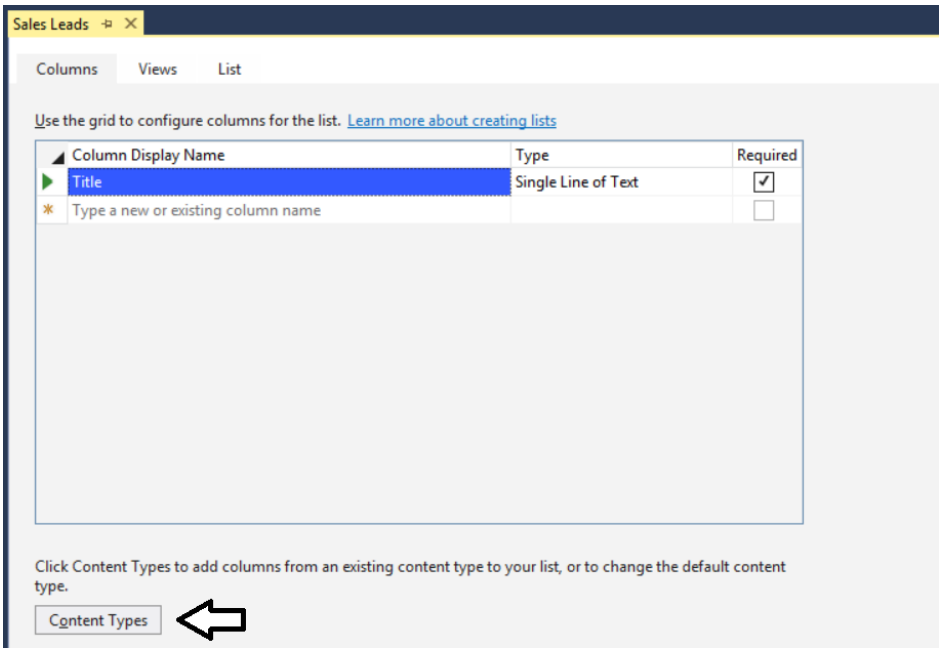
(1) **Create a customizable list based on:** Default (Blank)

c) Click **Finish**.



9. At this point, Visual Studio should display the new **Sales Leads** list in the Visual Studio list designer.

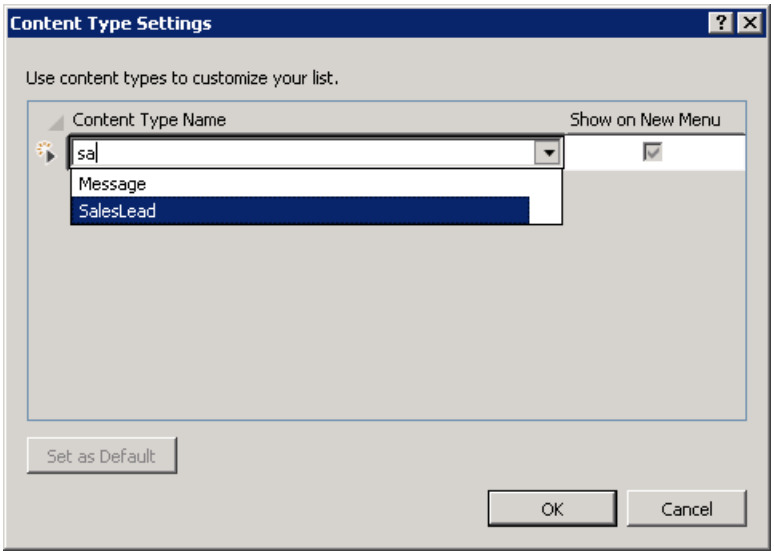
10. In the list designer, click the **Content Types** button:



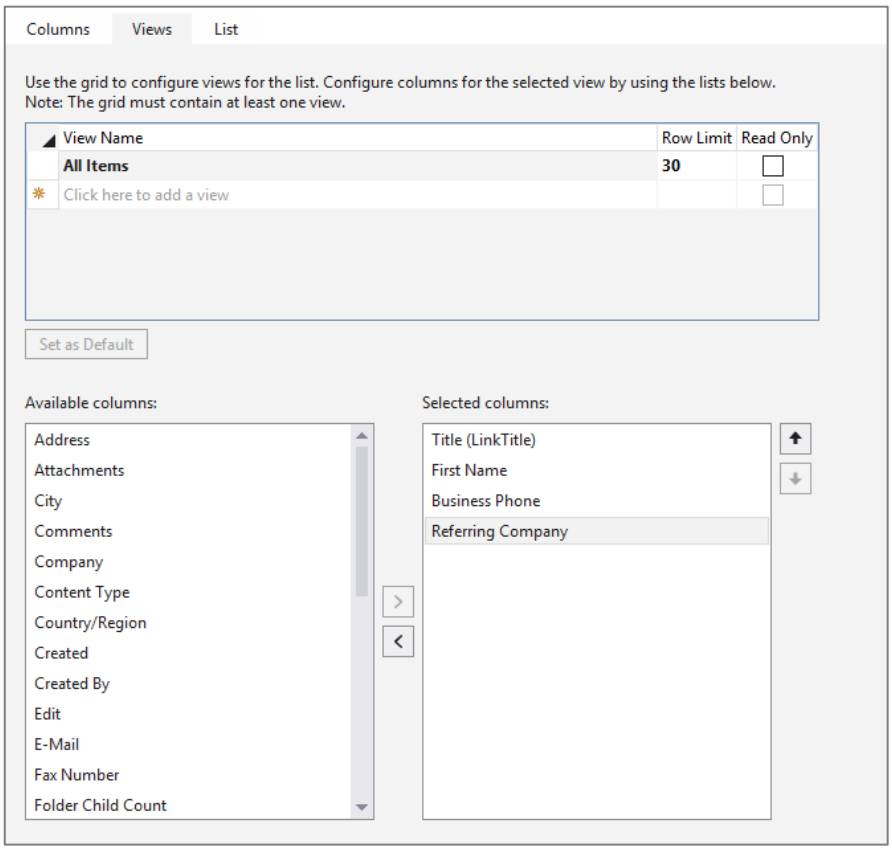
11. Remove the two existing content types by selecting each and pressing **[DELETE]**.

In order to delete it you need to select the whole row which can be done by clicking the green arrow to the left of the row.

12. Add the content type **SalesLead** which is included in our project and click **OK**.



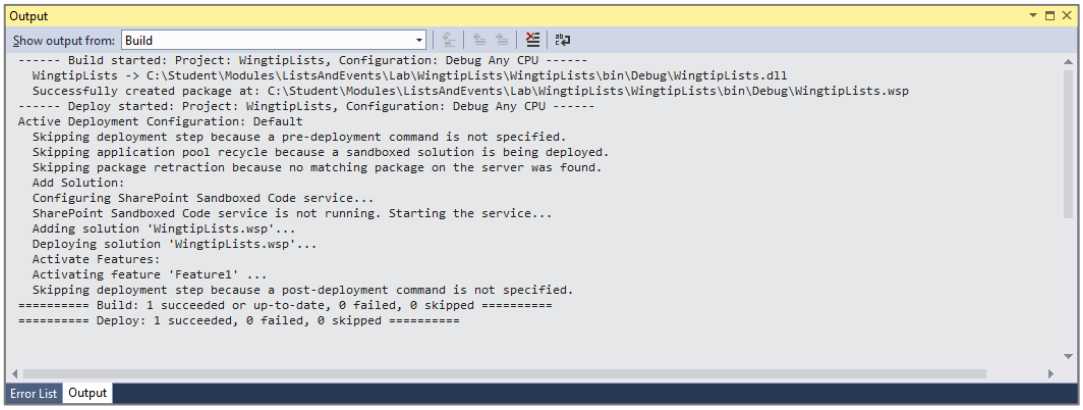
13. After adding the content type, move to the **Views** tab. You can see that the **All Items** view has too many columns to show them all on a single web page. Configure the Select columns for the **All Items** view to match the following screenshot.



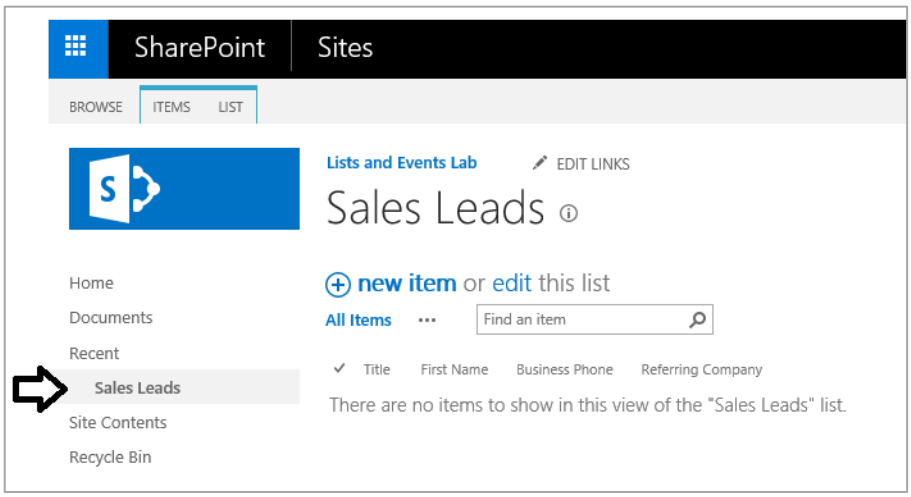
14. Save all changes: **File → Save All**.

Build and Test the Project

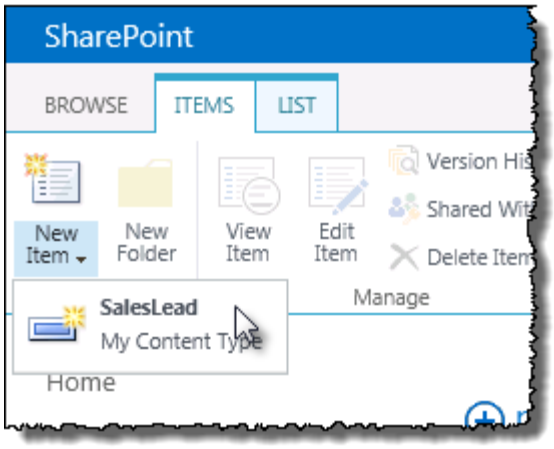
15. Make sure the Visual Studio **Output Windows** is showing. If it's not showing, display it using the **Ctrl + Alt + O** keyboard shortcut.
16. Build and test your SharePoint solution by right-clicking the top-level project node of the **WingtipTypes** project and select the **Deploy** command.
17. Monitor the Output Windows wait until the Deploy command has finished processing.



18. Using Internet Explorer, navigate to the test site at https://ListsAndEvents.wingtip.com.
19. On the home page of the site, you should be able to see a **References** link to the new list in the QuickLaunch navigation menu. Click on the **References** link to navigate to the default view of the **References** list.



20. On the **References** list, use the ribbon and select the **Items** tab. Then select the arrow below the **Items** button to show the content types available:



21. Select the **SalesLead** item.
22. On the new form, scroll to the bottom to see the column you added to the Contact content type.
23. Close the browser to stop the debugger and go back to Visual Studio.

In this exercise you used Visual Studio to create new site columns, a content type and a list instance.

Exercise 4: Create Server-Side Event for all Announcement Lists

In this exercise you will create a new server-side event for all announcement lists that will display a custom error message if the body of the announcement contains a specific string.

1. Create a new project in Visual Studio:
   a) Launch **Visual Studio**.
   b) Select **File → New → Project**.
   c) In the **New Project** dialog:
      i) Select the **SharePoint 2016 – Empty Project** template under the **Templates → Visual C# → Office / SharePoint → SharePoint Solutions** section.
      ii) **Name:** AnnouncementListEventChecker

d) Click **OK** to create the project.

e) In the **SharePoint Customization Wizard**, use the following values to complete the wizard and click Finish.

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

ii) **What is the trust level for this SharePoint solution?** Deploy as farm solution

f) Click **Finish**.

2. Add an event receiver that checks all new and updated announcement list items:

a) Using the **Solution Explorer** tool window, right-click the **AnnouncementListEventChecker** project and select **Add → New Item**.

b) In the **Add New Item** dialog, select the **Event Receiver** template from the **Visual C# Items → Office / SharePoint** category.

i) **Name:** ContosoAnnReceiver

c) Click **Add**.

d) In the **SharePoint Customization Wizard** dialog, use the following values to complete the form.

i) **What type of event receiver do you want**: List Item Events

ii) **What item should be the event source**: Announcements

iii) **Handle the following events**: (check the following)

(1) An item is being added

(2) An item is being updated

e) Click **Finish**.

3. Add code to the event receiver to check if the announcement contains the phrase “contoso” and if so, redirect the user:

a) In the **ContosoAnnReceiver.cs** file, add the following methods to the **ContosoAnnReceiver** class:

```
private void CheckForError(SPItemEventProperties properties) {

  string stringToValidate = properties.AfterProperties["Title"].ToString() +
                            properties.AfterProperties["Body"];

  if (!IsValidString(stringToValidate)){
    properties.ErrorMessage =
              "Creating announcements with the previous company name is not permitted.";
    properties.Status = SPEventReceiverStatus.CancelWithError;
  }

}

private bool IsValidString(string stringToValidate) {

  if (string.IsNullOrEmpty(stringToValidate))
    return true;

  // check if the string has "contoso" anywhere in the name
  return stringToValidate.ToLower().Contains("contoso") ? false : true;
}
```

b) Next, add a call to the **CheckForError()** method to both event receiver methods and remove the existing code. The two methods should look like the following code after the changes:

```
public override void ItemAdding(SPItemEventProperties properties) {
  CheckForError(properties);
}

public override void ItemUpdating(SPItemEventProperties properties) {
  CheckForError(properties);
}
```

4. Verify the event receiver registrations:

a) Using the **Solution Explorer** tool window, right-click the file **ContosoAnnReceiver \ Elements.xml** and select **Open**.

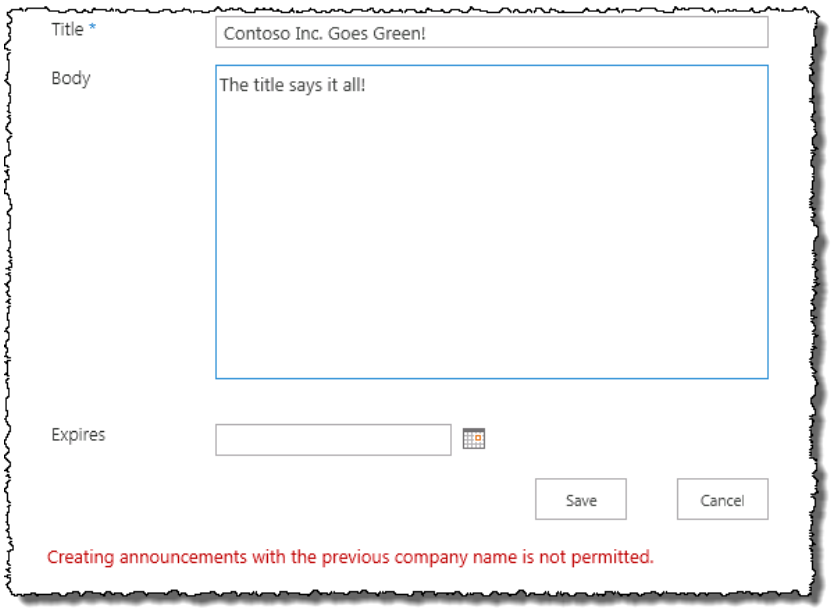
b) Notice the **<Receivers>** element is registering the event for **ListTemplateId=”104”** which is the ID of the Announcements list template. This means all the event receivers listed as children will be attached to all lists based on that type.

c) Notice the two **<Receiver>** elements contain references to the **ItemAdding** and **ItemUpdating** events.

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

Build and Test the Project

6. Build and test your application by pressing **[F5]** or **Debug → Start Debugging**.
7. Once the solution has been deployed, Internet Explorer will launch and navigate to the https://ListsAndEvents.wingtip.com site.
8. Create a new Announcements list:
   a) Using the **Quick Launch** navigation, select **Site Contents**.
   b) On the **Site Contents** page, select **Add an app**.
   c) On the **Site Contents > Your Apps**, select the **Announcement** app.
      i) Name the new app **Announcements** and click **Create**.
9. Add a valid announcement to the list:
   a) Using the **Quick Launch** navigation, select **Announcements**.
   b) On the **Announcements** page, create a new list item with the following values:
      i) **Title:** Wingtip Inc. Goes Green!
      ii) **Body:** The title says it all!
   c) Click **Save**.
   d) Notice the announcement is added to the list as normal.
10. Add an invalid announcement to the list:
   a) Using the **Quick Launch** navigation, select **Announcements**.
   b) On the **Announcements** page, create a new list item with the following values:
      i) **Title:** Contoso Inc. Goes Green!
      ii) **Body:** The title says it all!
   c) Click **Save**.
   d) Notice how an error message appears and the item isn't saved to the list:



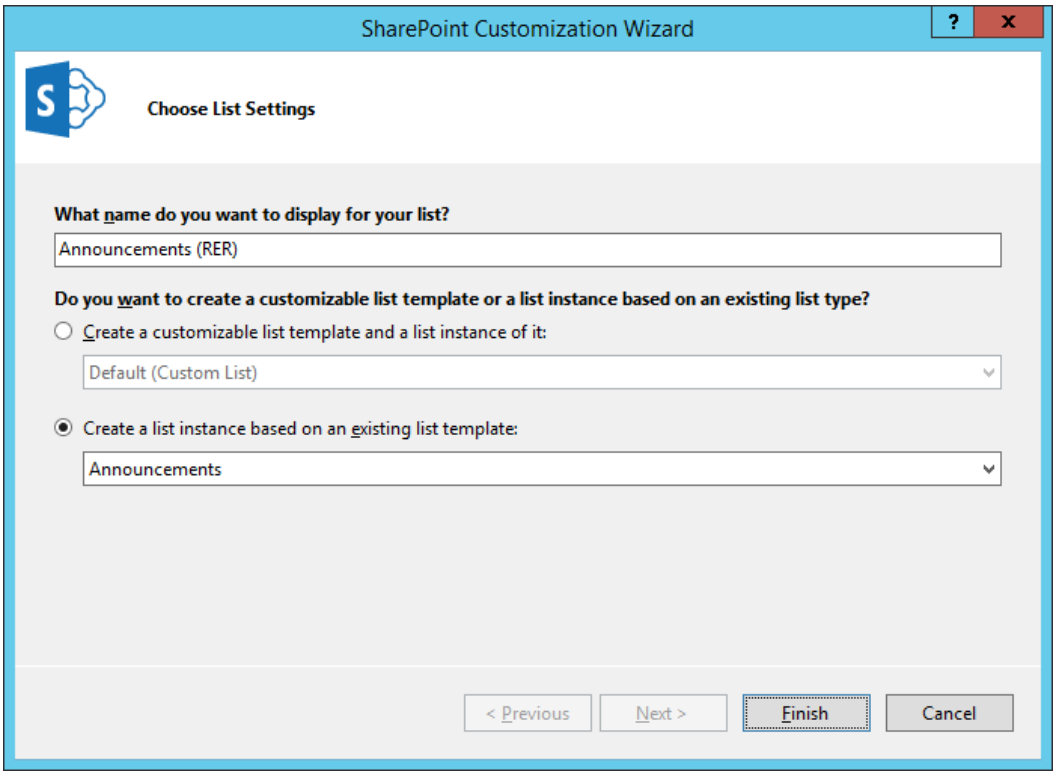
11. Close the browser to stop the debugger and go back to Visual Studio.

In this exercise you created an event receiver for all Announcement lists that displayed a custom error message when a certain string was detected.

Exercise 5: Creating List-Based Remote Event Receivers

In this exercise you will create a remote event receiver that is triggered for new announcements. All those will be done in a SharePoint-Hosted App.

1. Create a new project in Visual Studio 2013:
   a) Launch **Visual Studio 2013** as administrator: **Start → All Programs → Microsoft Visual Studio 2013 → Visual Studio 2013**.
   b) Select **File → New → Project**.
   c) In the **New Project** dialog:
      i) Select the **App for SharePoint 2013** template under the **Templates → Visual C# → Office / SharePoint → Apps** section.
      ii) **Name**: AnnRemoteEventReceiverApp
   d) Click **Ok** to create the project.
   e) In the **New App for SharePoint** wizard, use the following values to complete the wizard.
      i) **What is the name of your App for SharePoint?** Announcement RER App
      ii) **What site do you want to use for debugging?** http://ListsAndEvents.wingtip.com
      iii) **How do you want to host your app for SharePoint?** SharePoint-hosted
   f) Click **Finish**.
2. Add an Announcements list to the App:
   a) Using the **Solution Explorer** tool window, right-click the **AnnRemoteEventReceiverApp** node and select **Add → New Item**.
   b) In the **Add New Item** dialog, select the **List** template from **Visual C# Items → Office / SharePoint** category.
      i) **Name:** AnnouncementsListRER
   c) Click **Add**.
   d) In the **SharePoint Customization Wizard** dialog, use the following values to complete the form:
      i) **What name do you want to display for your list?** Announcements (RER)
      ii) Select the option **Create a list instance based on an Existing list template**.
      iii) **Select the list template named Announcements**.
   e) Click **Finish**.



3. Add the new announcement list to the homepage of the app:
   a) Using the **Solution Explorer** tool window, right-click the **Pages\Default.aspx** file and select **Open**.
   b) Locate the placeholder with the ID of **PlaceHolderPageTitleInTitleArea** and replace its contents with a better page title.

```
<asp:Content ContentPlaceHolderID="PlaceHolderPageTitleInTitleArea" runat="server">
  Lists and Events Lab
</asp:Content>
```

   c) Locate the placeholder with the ID of **PlaceHolderMain** and replace its contents with the following code to add a Web Part Zone to the page.

```
<asp:Content ContentPlaceHolderID="PlaceHolderMain" runat="server">
  <div>
    <WebPartPages:WebPartZone runat="server" FrameType="TitleBarOnly" ID="full" Title="loc:full" />
  </div>
</asp:Content>
```

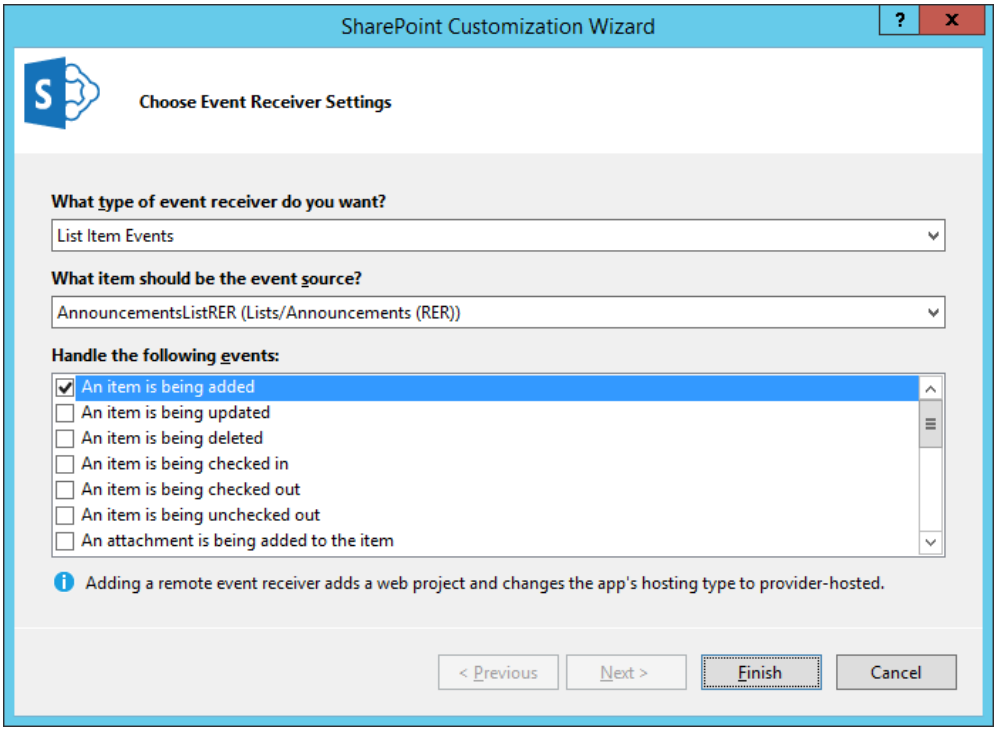
   d) Using the **Solution Explorer** tool window, right-click the **Pages\Elements.xml** file and select **Open**.
   e) Replace the **<File>** element in the **Elements.xml** file with the following to not only provision the page but also add the announcements list using an **XsltListViewWebPart** created by the app to the page. If you'd rather not type all this in by hand, you can copy and paste it from a text file named **FileElementSnippet.txt** which is located in the **StarterFiles** folder for this lab.

```
<File Path="Pages\Default.aspx" Url="Pages/Default.aspx">
  <AllUsersWebPart WebPartZoneID="Full" WebPartOrder="0">
    <![CDATA[
      <webParts>
        <webPart xmlns="http://schemas.microsoft.com/WebPart/v3">
          <metaData>
            <type name="Microsoft.SharePoint.WebPartPages.XsltListViewWebPart, Microsoft.SharePoint,
                        Version=14.0.0.0,Culture=neutral, PublicKeyToken=71e9bce111e9429c" />
            <importErrorMessage>Cannot import this Web Part.</importErrorMessage>
          </metaData>
          <data>
```

```
            <properties>
              <property name="Title" type="string">Announcements (RER)</property>
              <property name="ListDisplayName" type="string">Announcements (RER)</property>
               <property name="ChromeType" type="chrometype">TitleOnly</property>
            </properties>
          </data>
        </webPart>
      </webParts>
    ]]>
  </AllUsersWebPart>
</File>
```

Create a Remote Event Receiver

4. Add a remote event receiver to the App:
   a) Using the **Solution Explorer** tool window, right-click the **AnnRemoteEventReceiverApp** node and select **Add → New Item**.
   b) In the **Add New Item** dialog, select the **Remote Event Receiver** template from the **Visual C# Items → Office / SharePoint** category.
      i) **Name:** AnnRemoteEventReceiver
   c) Click **Add**.
   d) In the **SharePoint Customization Wizard** dialog, use the following values to complete the form:
      i) **What type of event receiver do you want**: List Item Events
      ii) **What item should be the event source**: Announcements (RER) (AnnRemoteEventReceiverApp\AnnouncementsListRER)
      iii) **Handle the following events**: (check the following)
         (1) An item is being added
   e) Click **Finish**.



Note that remote event handlers are supported in provider-hosted apps but not in SharePoint-hosted app. Therefore, adding a remote event receiver in this fashion triggers Visual Studio to convert the SharePoint app project into a provider-hosted app project on the fly.

5. First, modify the security configuration of the app to use internal security and not OAuth or a special S2S token:
   a) Using the **Solution Explorer** tool window, within the **AnnRemoteEventReceiverApp** project, right-click the **AppManifest.xml** file and select **View Code**.
   b) Locate the **<AppPrincipal>** node.
   c) Replace the contents with **<Internal />** so it looks like the following markup:

```
<AppPrincipal>
  <Internal />
</AppPrincipal>
```

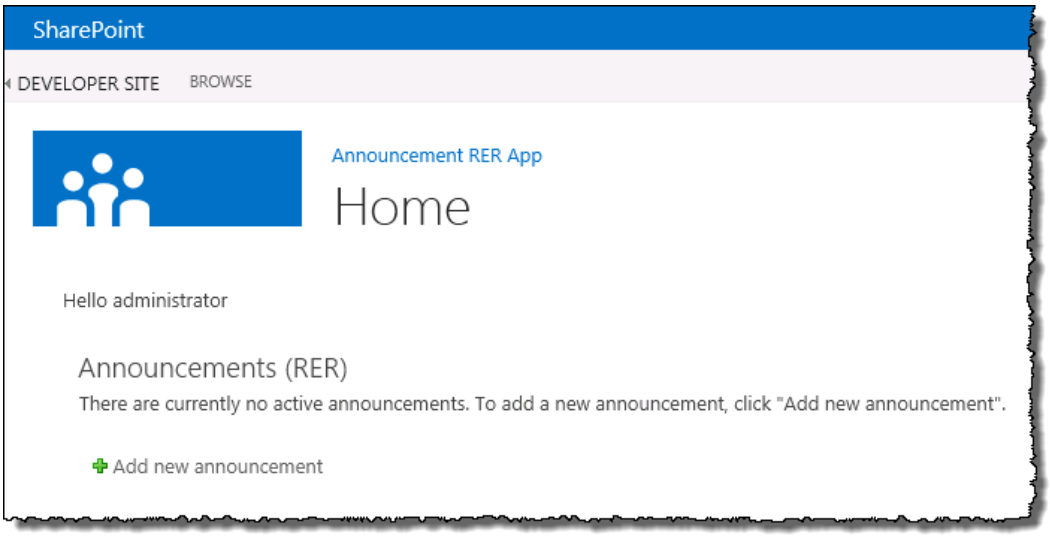
6. Now update the configuration of the remote web project's security:
   a) Using the **Solution Explorer** tool window, within the **AnnRemoteEventReceiverAppWeb** project, right-click the **web.config** file and select **Open**.
   b) Locate the two **<add>** nodes with a **key** attribute that starts with **Client** and delete them.
7. With everything configured you can now code the event receivers. You will create an event that will update the item that triggered the event and update the Body field:
   a) Using the **Solution Explorer** tool window, within the **AnnRemoteEventReceiverAppWeb** project, right-click the **AnnRemoteEventReceiver.svc\ AnnRemoteEventReceiver.svc.cs** file and select **View Code**.
   b) Locate the **ProcessEvent()** method. Within this method you will find a lot of commented code. This is helper code for reaching back into SharePoint when the site is not using internal security.
      i) Delete all the commented code in **ProcessEvent()**.
   c) Update the **ProcessEvent()** method to match following code.

```
public SPRemoteEventResult ProcessEvent(SPRemoteEventProperties properties) {
  SPRemoteEventResult result = new SPRemoteEventResult();

  if (properties.EventType == SPRemoteEventType.ItemAdding) {
    string bodyValue = properties.ItemEventProperties.AfterProperties["Body"].ToString();
    bodyValue += "\n\n\n *** CONFIDENTIAL *** \n";

    result.ChangedItemProperties.Add("Body", bodyValue);
  }

  return result;
}
```

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

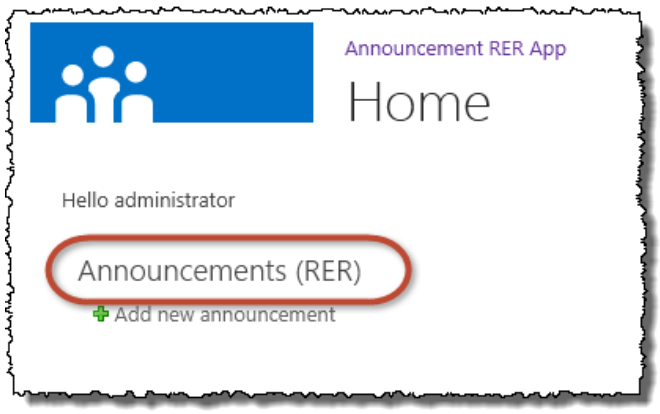
Build and Test the Project

9. Build and test your application by pressing **[F5]** or **Debug → Start Debugging**.
10. Once the solution has been deployed, Internet Explorer will launch and navigate to the http://ListsAndEvents.wingtip.com site.
11. Test the remote event receiver:
    a) Using the **Quick Launch** navigation, select **Announcements RER App**.

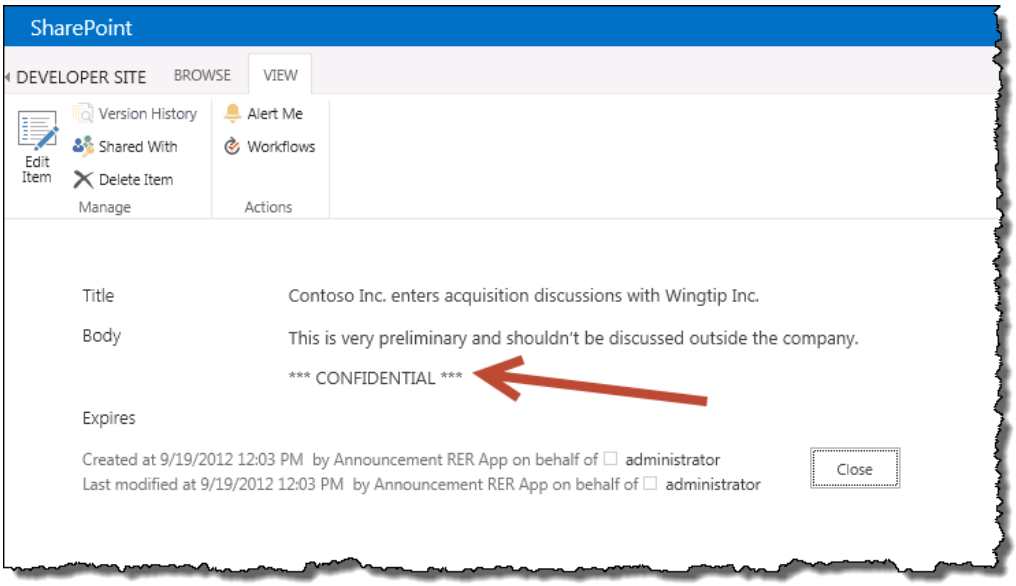


b) When the page loads, click the **Add new announcement** link and create a new announcement with the following values:
   i) **Title:** Contoso Inc. enters acquisition discussions with Wingtip Inc.
   ii) **Body:** This is very preliminary and shouldn't be discussed outside the company.

c) After creating the item, click the **Announcements (RER)** title of the XsltListViewWebPart on the page to navigate to the list:



d) Click the only announcement in the list, the one you created and notice the **Body** field has been updated:



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

In this exercise you saw how to create a new remote event receiver for a list.