## Creating SharePoint 2013 Event Receivers

**Lab Time**: 60 minutes

**Lab Folder**: [[StudentFolder]]\ListsEvents

**Lab Overview**: In this lab you will create different types of events in SharePoint 2013. First you will learn how to create server-side events in SharePoint 2013, the traditional type of events that run on the SharePoint server. You will also learn how to create remote event receivers, new in SharePoint 2013, 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, http://listevents.wingtip.com.

1. Setup a new site collection for this lab:
   1. Ensure you are logged into the **WingtipServer** server as **WINGTIP\Administrator**.
   2. Run a PowerShell script, found in the root lab folder for this module:
      1. Right-click **SetupModule.ps1** and select **Run with PowerShell**. This file can be found in the files associated with this lab:

[..]\ListsEvents

* 1. When the script completes, it will launch a new browser and navigate to the lab site collection.
  2. Close the PowerShell console window.

### Exercise 2: 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 2012:
   1. Launch **Visual Studio 2012**:**Start** 🡪 **All Programs** 🡪 **Microsoft Visual Studio 2012** 🡪 **Visual Studio 2012**
   2. Select **File 🡪 New 🡪 Project**.
   3. In the **New Project** dialog:
      1. Select the **SharePoint 2013 – Empty Project** template under the **Templates 🡪 Visual C# 🡪 Office / SharePoint 🡪 SharePoint Solutions** section.
      2. **Name:** AnnouncementListEventChecker
   4. Click **OK** to create the project.
   5. In the **SharePoint Customization Wizard**, use the following values to complete the wizard and click Finish.
      1. **What site do you want to use for debugging?** <http://listevents.wingtip.com>
      2. **What is the trust level for this SharePoint solution?** Deploy as farm solution
   6. Click **Finish**.
2. Add an event receiver that checks all new and updated announcement list items:
   1. Using the **Solution Explorer** tool window, right-click the **AnnouncementListEventChecker** project and select **Add 🡪 New Item**.
   2. In the **Add New Item** dialog, select the **Event Receiver** template from the **Visual C# Items 🡪 Office / SharePoint** category.
      1. **Name:** ContosoAnnReceiver
   3. Click **Add**.
   4. In the **SharePoint Customization Wizard** dialog, use the following values to complete the form.
      1. **What type of event receiver do you want**: List Item Events
      2. **What item should be the event source**: Announcements
      3. **Handle the following events**: (check the following)
         1. An item is being added
         2. An item is being updated
   5. Click **Finish**.
3. Add code to the event receiver to check if the announcement contains the phrase “contoso” and if so, redirect the user:
   1. 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;

}

* 1. 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);

}

1. Verify the event receiver registrations:
   1. Using the **Solution Explorer** tool window, right-click the file **ContosoAnnReceiver \ Elements.xml** and select **Open**.
   2. 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.
   3. Notice the two **<Receiver>** elements contain references to the **ItemAdding** and **ItemUpdating** events.
2. Save all changes: **File 🡪 Save All**.

#### Build and Test the Project

1. Build and test your application by pressing **[F5]** or **Debug 🡪 Start Debugging**.
2. Once the solution has been deployed, Internet Explorer will launch and navigate to the <http://listevents.wingtip.com> site.
3. Create a new Announcements list:
   1. Using the **Quick Launch** navigation, select **Site Contents**.
   2. On the **Site Contents** page, select **Add an app**.
   3. On the **Site Contents > Your Apps**, select the **Announcement** app.
      1. Name the new app **Announcements** and click **Create**.
4. Add a valid announcement to the list:
   1. Using the **Quick Launch** navigation, select **Announcements**.
   2. On the **Announcements** page, create a new list item with the following values:
      1. **Title:** Wingtip Inc. Goes Green!
      2. **Body:** The title says it all!
   3. Click **Save**.
   4. Notice the announcement is added to the list as normal.
5. Add an invalid announcement to the list:
   1. Using the **Quick Launch** navigation, select **Announcements**.
   2. On the **Announcements** page, create a new list item with the following values:
      1. **Title:** Contoso Inc. Goes Green!
      2. **Body:** The title says it all!
   3. Click **Save**.
   4. Notice how an error message appears and the item isn’t saved to the list:



1. 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.

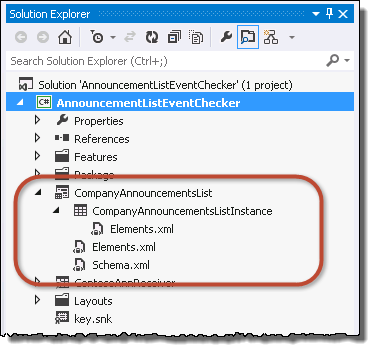
### Exercise 3: Create Server-Side Event for a Specific List

In this exercise you will take the project you were just working on and modify it so that the event receivers are added to a specific list rather than all lists of type Announcements.

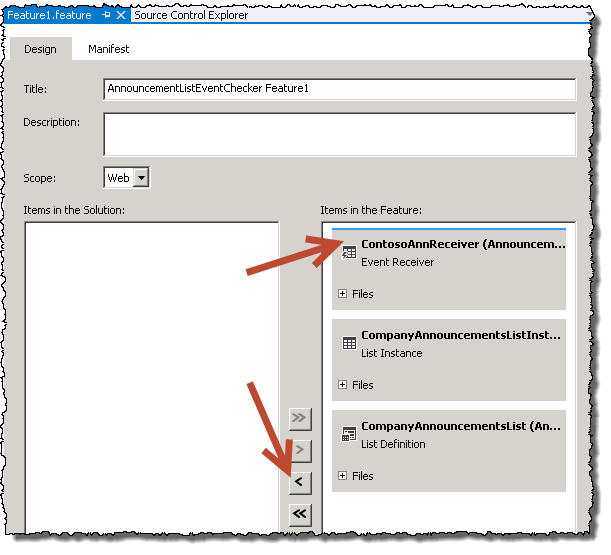
1. Open an existing starter project in Visual Studio 2012:
   1. Launch **Visual Studio 2012** as administrator: **Start 🡪 All Programs 🡪 Microsoft Visual Studio 2012 🡪 Visual Studio 2012**.
   2. Select **File 🡪 Open 🡪 Project/Solution**.
   3. In the **Open Project** dialog, select the following project provided in the files associated with this exercise:

[..]\ListsEvents\Exercises\Ex3\AnnouncementListEventChecker.sln

Notice this project has been slightly modified from the one you were using in the last exercise. A new SharePoint Project Item has been added to create an instance of an Announcement list called Company Announcements.



1. Modify the project to not attach the event receiver to every list of type Announcements list:
   1. Using the **Solution Explorer** tool window, right-click the **Features / Feature1** and select **View Designer**.
   2. In the section **Items in the Feature**, find the **ContosoAnnReceiver**, and remove it using the arrows in the middle of the two panes:



1. In order to attach the event handler to a specific list, you will need to write some custom code. The best place for this is going to be in the Feature receiver:
   1. Using the **Solution Explorer** tool window, right-click the **Features / Feature1** and select **Add Event Receiver**.
   2. Within the **Feature1.EventReceiver.cs**, find the **FeatureActivated()** method and uncomment it.
   3. Add the following code to the body of the **FeatureActivated()** method to create two new events for that specific list:

SPWeb site = properties.Feature.Parent as SPWeb;

if (site == null) { return; }

SPList list = site.Lists.TryGetList("Company Announcements");

if (list == null) { return; }

// attach event to ItemAdding

var itemAddingEvent = list.EventReceivers.Add();

itemAddingEvent.Type = SPEventReceiverType.ItemAdding;

itemAddingEvent.Synchronization = SPEventReceiverSynchronization.Synchronous;

itemAddingEvent.Assembly = System.Reflection.Assembly.GetExecutingAssembly().FullName;

itemAddingEvent.Class = "AnnouncementListEventChecker.ContosoAnnReceiver.ContosoAnnReceiver";

itemAddingEvent.Update();

// attach event to ItemUpdatinging

var itemUpdatingEvent = list.EventReceivers.Add();

itemUpdatingEvent.Type = SPEventReceiverType.ItemUpdating;

itemUpdatingEvent.Synchronization = SPEventReceiverSynchronization.Synchronous;

itemUpdatingEvent.Assembly = System.Reflection.Assembly.GetExecutingAssembly().FullName;

itemUpdatingEvent.Class = "AnnouncementListEventChecker.ContosoAnnReceiver.ContosoAnnReceiver";

itemUpdatingEvent.Update();

1. Save all changes: **File 🡪 Save All**.

#### Build and Test the Project

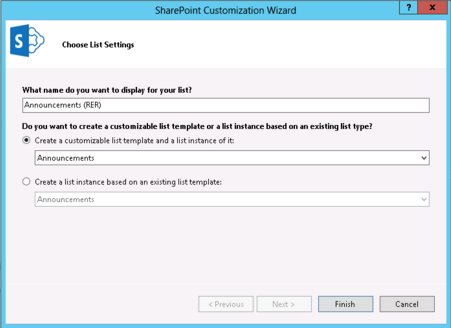
1. Build and test your application by pressing **[F5]** or **Debug 🡪 Start Debugging**.
2. Once the solution has been deployed, Internet Explorer will launch and navigate to the <http://listevents.wingtip.com> site.
3. Test the event receiver:
   1. Using the Quick Launch navigation, select **Company Announcements**.
4. Add a valid announcement to the list:
   1. Using the **Quick Launch** navigation, select **Announcements**.
   2. On the **Announcements** page, create a new list item with the following values:
      1. **Title:** Wingtip Inc. Goes Green!
      2. **Body:** The title says it all!
   3. Click **Save**.
   4. Notice the announcement is added to the list like normal.
5. Add an invalid announcement to the list:
   1. Using the **Quick Launch** navigation, select **Announcements**.
   2. On the **Announcements** page, create a new list item with the following values:
      1. **Title:** Contoso Inc. Goes Green!
      2. **Body:** The title says it all!
   3. Click **Save**.
   4. Notice how an error dialog appears and the item isn’t saved to the list.
6. Close the browser to stop the debugger and go back to Visual Studio.

In this exercise you created an event receiver for the Company Announcements list that displayed a custom error message when a certain string was detected.

### Exercise 4: 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 2012:
   1. Launch **Visual Studio 2012** as administrator: **Start 🡪 All Programs 🡪 Microsoft Visual Studio 2012 🡪 Visual Studio 2012**.
   2. Select **File 🡪 New 🡪 Project**.
   3. In the **New Project** dialog:
      1. Select the **App for SharePoint 2013** template under the **Templates 🡪 Visual C# 🡪 Office / SharePoint 🡪 Apps** section.
      2. **Name**: AnnRemoteEventReceiverApp
   4. Click **Ok** to create the project.
   5. In the **New App for SharePoint** wizard, use the following values to complete the wizard.
      1. **What is the name of your App for SharePoint?** Announcement RER App
      2. **What site do you want to use for debugging?** <http://listevents.wingtip.com>
      3. **How do you want to host your app for SharePoint?** SharePoint-hosted
   6. Click **Finish**.
2. Add an Announcements list to the App:
   1. Using the **Solution Explorer** tool window, right-click the **AnnRemoteEventReceiverApp** node and select **Add 🡪 New Item**.
   2. In the **Add New Item** dialog, select the **List** template from **Visual C# Items 🡪 Office / SharePoint** category.
      1. **Name:** AnnouncementsListRER
   3. Click **Add**.
   4. In the **SharePoint Customization Wizard** dialog, use the following values to complete the form:
      1. **What name do you want to display for your list?** Announcements (RER)
      2. **Do you want to create a customizable list template or a list instance based on an existing list type**: Create a customizable list template and a list instance of it: Announcements
   5. Click **Finish**.



1. Add the new announcement list to the homepage of the app:
   1. Using the **Solution Explorer** tool window, right-click the **Pages\Default.aspx** file and select **Open**.
   2. Add the following markup to the page, just before the closing **</asp:Content>** at the bottom of the file. This will add a Web Part Zone to the page.

<WebPartPages:WebPartZone runat="server" FrameType="TitleBarOnly" ID="full" Title="loc:full" />

* 1. Using the **Solution Explorer** tool window, right-click the **Pages\Elements.xml** file and select **Open**.
  2. 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:

<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>

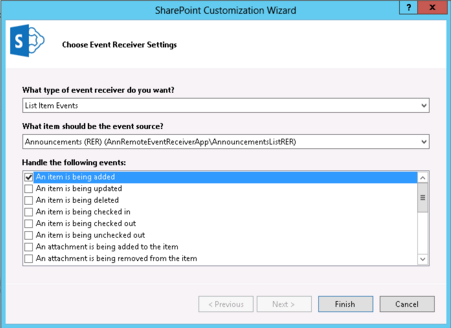
]]>

</AllUsersWebPart>

</File>

#### Create a Remote Event Receiver

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



With the event handler created, you can now create the implementation of it.

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

<AppPrincipal>

<Internal />

</AppPrincipal>

1. Now update the configuration of the remote web project’s security:
   1. Using the **Solution Explorer** tool window, within the **AnnRemoteEventReceiverAppWeb** project, right-click the **web.config** file and select **Open**.
   2. Locate the two **<add>** nodes with a **key** attribute that starts with **Client** and delete them.
2. 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:
   1. Using the **Solution Explorer** tool window, within the **AnnRemoteEventReceiverAppWeb** project, right-click the **AnnRemoteEventReceiver.svc\ AnnRemoteEventReceiver.svc.cs** file and select **View Code**.
   2. 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.
      1. Delete all the commented code in **ProcessEvent()**.
   3. Add the following code to the **ProcessEvent()** method to update the body of the item:

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);

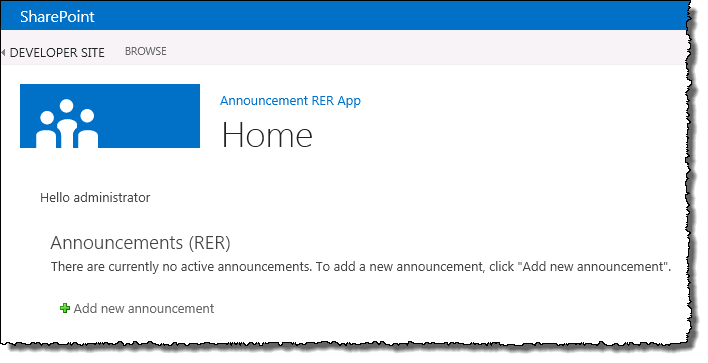
}

return result;

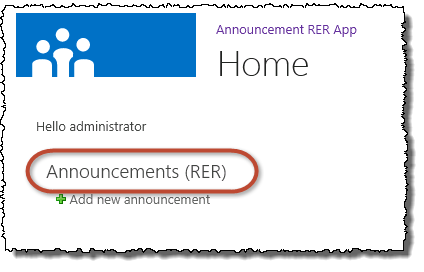
1. Save all changes: **File 🡪 Save All**.

#### Build and Test the Project

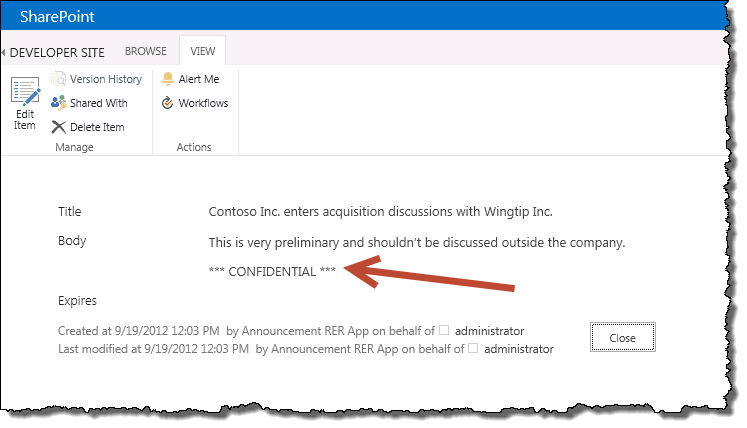
1. Build and test your application by pressing **[F5]** or **Debug 🡪 Start Debugging**.
2. Once the solution has been deployed, Internet Explorer will launch and navigate to the <http://listevents.wingtip.com> site.
3. Test the remote event receiver:
   1. Using the **Quick Launch** navigation, select **Announcements RER App**.



* 1. When the page loads, click the **Add new announcement** link and create a new announcement with the following values:
     1. **Title:** Contoso Inc. enters acquisition discussions with Wingtip Inc.
     2. **Body:** This is very preliminary and shouldn’t be discussed outside the company.
  2. After creating the item, click the **Announcements (RER)** title of the XsltListViewWebPart on the page to navigate to the list:



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



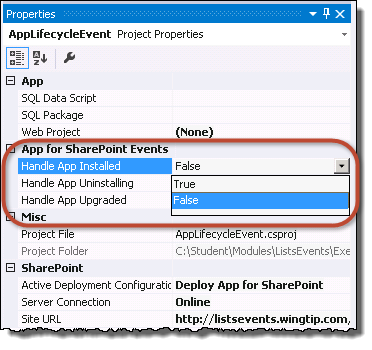
1. Close the browser to stop the debugger and go back to Visual Studio.

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

### Exercise 5: Create an App Lifecycle Remote Event Receiver

In this exercise you will create a remote event receiver that is called in one of the app lifecycle events.

1. Create a new project in Visual Studio 2012 for hosting your app:
   1. Launch **Visual Studio 2012** as administrator: **Start** 🡪 **All Programs** 🡪 **Microsoft Visual Studio 2012** 🡪 **Visual Studio 2012**
   2. Select **File 🡪 New 🡪 Project**.
   3. In the New Project dialog:
      1. Select the **App for SharePoint 2013** template under the **Templates 🡪 Visual C# 🡪 Office / SharePoint 🡪 Apps** section.
      2. **Name**: AppLifecycleEvent
   4. Click **Ok** to create the project.
   5. In the **New App for SharePoint** wizard, use the following values to complete the wizard.
      1. **What is the name of your App for SharePoint?** App Lifecycle Event
      2. **What site do you want to use for debugging?** <http://listevents.wingtip.com>
      3. **How do you want to host your app for SharePoint?** SharePoint-hosted
   6. Click **Finish**.
2. Create a new remote event receiver that will be called when the app is installed:
   1. Using the **Solution Explorer** tool window, select the **AppLifecycleEvent** project.
   2. Using the **Properties** tool window, find the **Handle App Installed** and set it to **True**:



* 1. Visual Studio 2012 will prompt you to create a new web project to host the event. Click **OK**.

1. Now code the event. You could do any number of things with the code in this event, but for demonstration purposes you will simply show that something happened by writing to the event log on the server:
   1. Using the **Solution Explorer** tool window, within the **AppLifecycleEventWeb** project, right-click the **AppEventReceiver.svc \ AppEventReceiver.svc.cs** file and select **View Code**.
   2. Locate the **ProcessEvent()** method and delete all the commented code in **ProcessEvent()**.
   3. Add the following code to the **ProcessEvent()** method to write to the event log the app was installed:

SPRemoteEventResult result = new SPRemoteEventResult();

if (properties.EventType == SPRemoteEventType.AppInstalled)

{

string source = "Lifecycle SharePoint App Demo";

// verify there's a category in the Event Log for our app

if (!System.Diagnostics.EventLog.SourceExists(source))

{

System.Diagnostics.EventLog.CreateEventSource(source, "Application");

}

// write to the log

System.Diagnostics.EventLog.WriteEntry(source, "App Installed");

}

result.Status = SPRemoteEventServiceStatus.Continue;

return result;

#### Update Security Configuration for the App

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

<AppPrincipal>

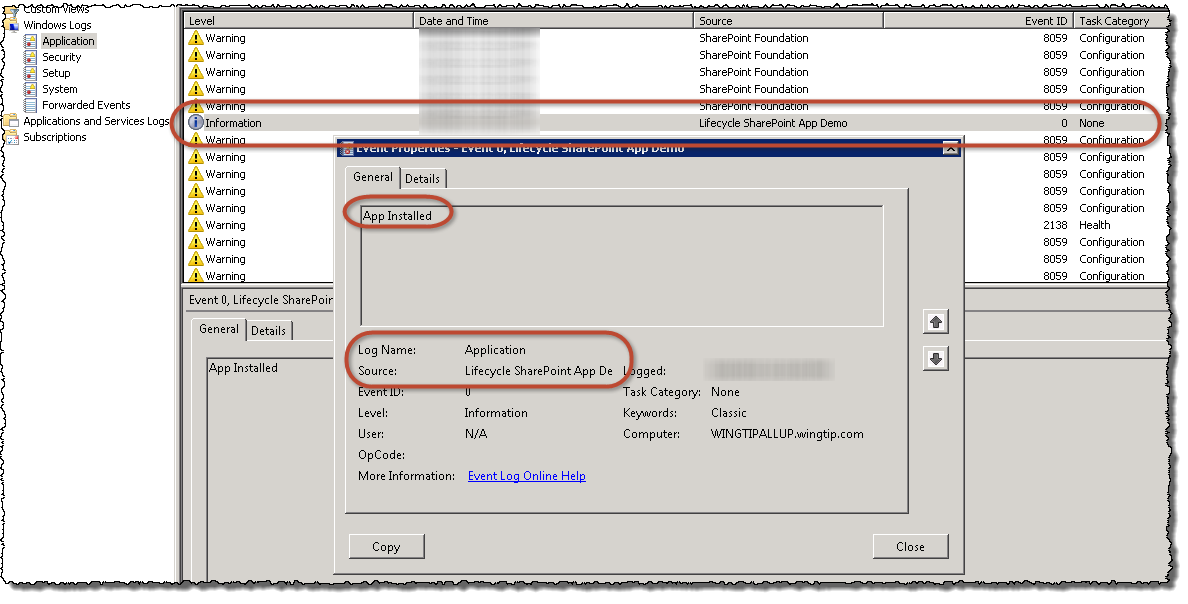
<Internal />

</AppPrincipal>

1. Now update the configuration of the remote web project’s security:
   1. Using the **Solution Explorer** tool window, within the **AppLifecycleEventWeb** project, right-click the **web.config** file and select **Open**.
   2. Locate the two **<add>** nodes with a **key** attribute that starts with **Client** and delete them.
2. Save all changes: **File 🡪 Save All**.

#### Build and Test the Project

1. Build and test your application by pressing **[F5]** or **Debug 🡪 Start Debugging**.
2. Once the solution has been deployed, Internet Explorer will launch and navigate to the <http://listevents.wingtip.com> site.
3. At this point the app has been installed. Open the **Event Viewer** on the server to see the log message: **Start 🡪 Administrative Tools 🡪 Event Viewer**.
4. When the **Event Viewer** window loads, expand the tree on the left to **Event Viewer 🡪 Windows Logs 🡪 Application**.
5. With the **Application** log open, find the event we just wrote from the app:



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

In this exercise you learned how to create a remote event receiver to handle app lifecycle events.