## Create Office 365 Developer SharePoint Site Collection

This lab requires you first create an Office 365 Developer Site. Visit <http://dev.office.com> and sign up for an Office 365 Developer Site. Once provisioned, create a new site using the Developer Site template at [**https://YourTenant.sharepoint.com/sites/Dev**](https://YourTenant.sharepoint.com/sites/Dev)**.**

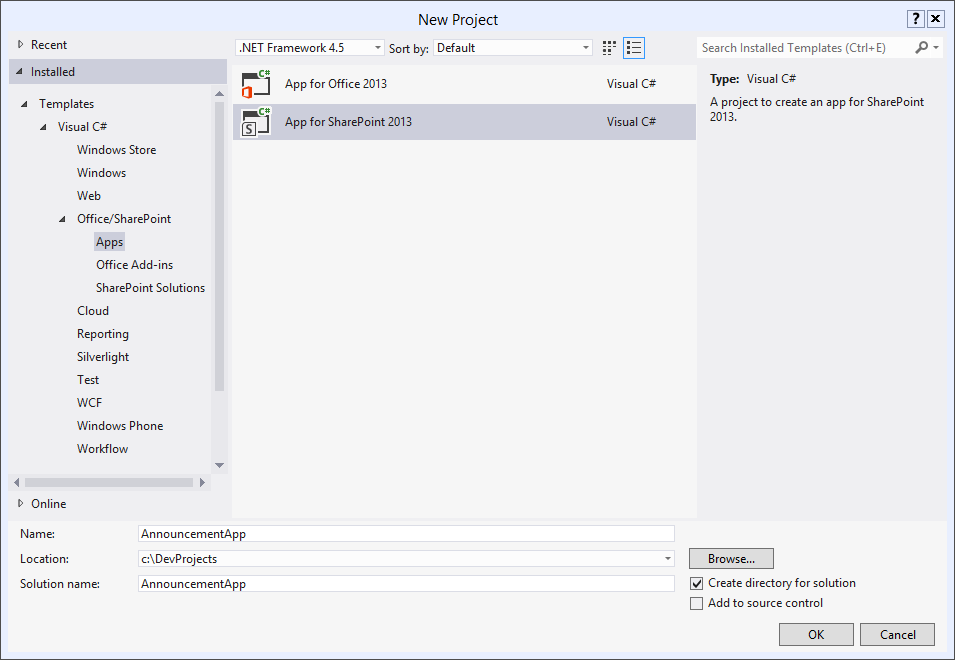
# **Demo 1: Remote Event Receivers**

In this exercise, you will create a Remote Event Receiver that handles events for an Announcements list. The Announcements will be deployed as part of an App.

## Task 1 – Create a New App

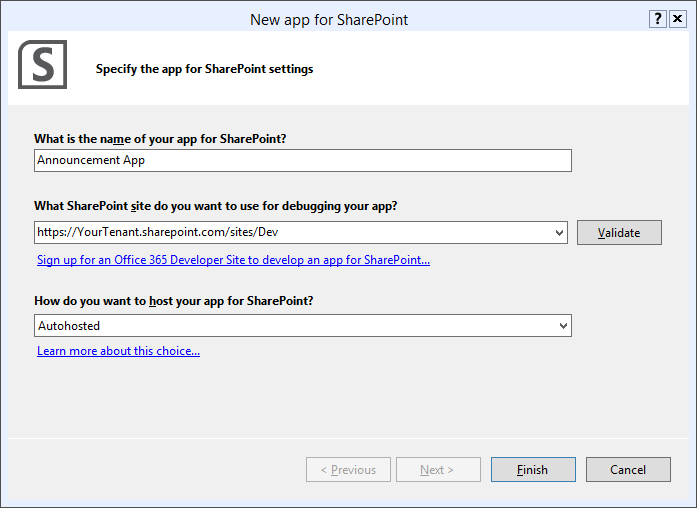
In this task, you will develop a new App in Visual Studio

1. Open Microsoft Visual Studio 2012 and create a new SharePoint App project
   1. Open **Microsoft Visual Studio 2012**
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚Apps** node and select the **App for SharePoint 2013** project template
   4. Name the new project **AnnouncementApp**
   5. Click the **OK** button



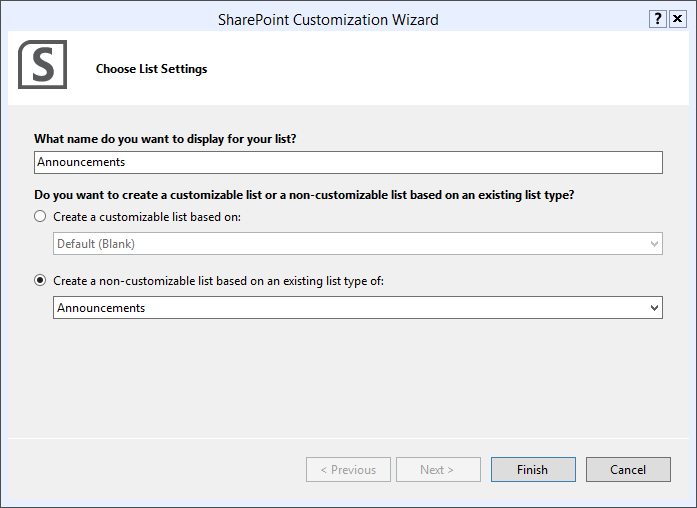
*Create new SharePoint App Project*

* 1. In the **New App for SharePoint Wizard**, name the App **Announcement App**.
  2. Specify the Office 365 developer site you will use for this lab.
  3. Select **Autohosted** as the hosting type.
  4. Click **Finish**.

**

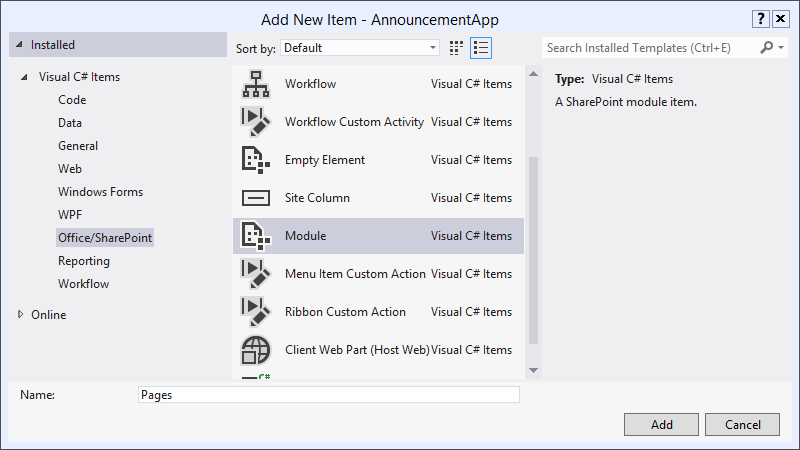
*New App for SharePoint Wizard*

1. Add an Announcements List
   1. In the Solution Explorer, right click the **AnnouncementApp** project and select **Add⮚New Item** from the Context menu.
   2. In the Add New Item dialog, select **List**.
   3. Name the new List **Announcements** and click **Add**.
   4. In the **New App for SharePoint Wizard**, select **Create a Non-Customizable List based On**.
   5. Choose to use the **Announcements** list and click **Finish**.

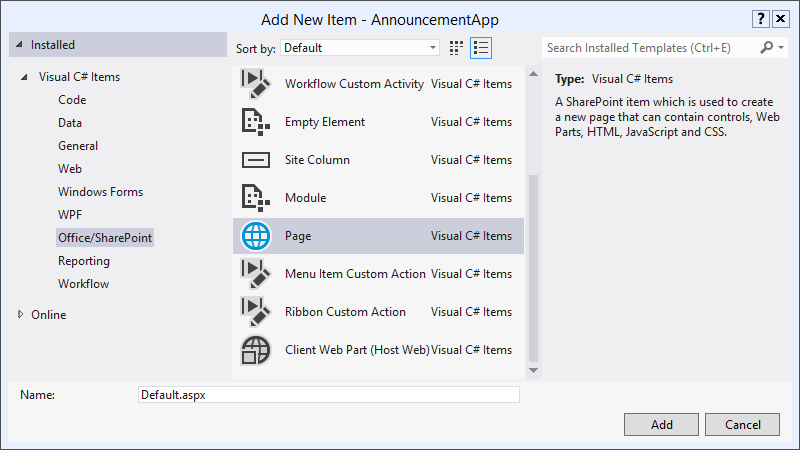
**

*New App for SharePoint Wizard*

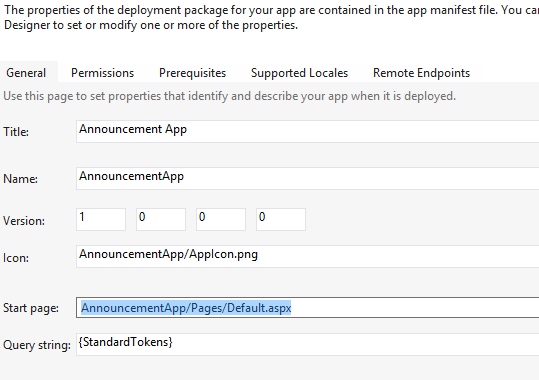
1. Add a new **Module** named **Pages**.
   1. In the Solution Explorer, right click the **AnnouncementApp** project and select **Add⮚New Item** from the Context menu.
   2. In the Add New Item dialog, select **Module**.
   3. Name the new module **Pages** and click **Add**.
   4. Delete the generated **Sample.txt** file.



1. Add a new **Page** named **Default.aspx** to the **Pages** module.
   1. In the Solution Explorer, right click the **Pages** module and select **Add⮚New Item** from the Context menu.
   2. In the Add New Item dialog, select **Page**.
   3. Name the new page **Default.aspx** and click **Add**.



1. Change the app start URL to the app web’s default.aspx page.
   1. **Double-click** the file **appmanifest.xml** to view the app manifest.
   2. In the **Start Page** drop-down, change the start page to **AnnouncementApp/Pages/Default.aspx**.



1. Add the Announcements List to the App Home page
   1. Open **Pages\Elements.xml** in Visual Studio for editing
   2. **Replace** the **File** element associated with the **Default.aspx** page to appear like the following:

<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</property>

<property name="ListDisplayName"

type="string">Announcements</property>

<property name="ChromeType"

type="chrometype">TitleOnly</property>

</properties>

</data>

</webPart>

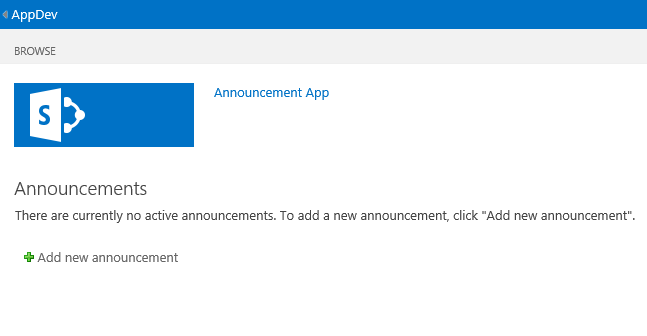
</webParts>

]]>

</AllUsersWebPart>

</File>

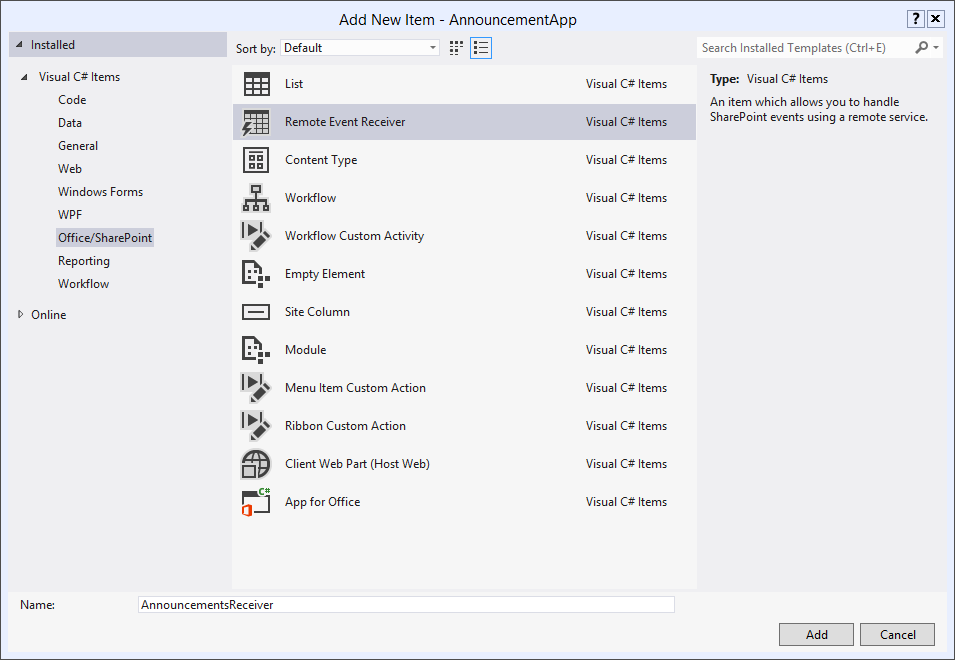
1. Test the app by hitting F5 to verify your changes.



## Task 2 – Create an Event Handler

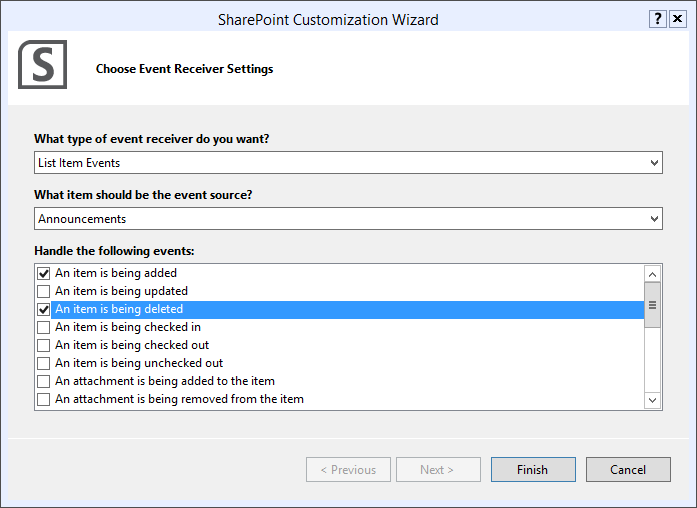
In this task, you will create an event handler for the list.

1. Add the Event Handler
   1. In the Solution Explorer, right click the **AnnouncementApp** project and select **Add⮚New Item** from the Context menu.
   2. In the **Add New Item** dialog, select **Remote Event Receiver**.
   3. Name the new item **AnnouncementsReceiver** and click **Add**.



*Adding a Remote Event Receiver*

* 1. In the Customization Wizard, select **Announcements** as the event source.
  2. Check the **An Item is Being** **Added** and **An Item is Being** **Deleted** events.
  3. Click **Finish**.



*Selecting Events*

1. Code the Event Handler
   1. In the Solution Explorer, click on the **AnnouncementWebApp** project.
   2. Open **AnnouncementsReceiver.svc.cs** for editing.
   3. **Replace** the methods in the class with the following code.

public SPRemoteEventResult ProcessEvent(SPRemoteEventProperties properties)

{

SPRemoteEventResult result = new SPRemoteEventResult();

using (ClientContext clientContext = TokenHelper.CreateRemoteEventReceiverClientContext(properties))

{

if (clientContext != null)

{

switch (properties.EventType)

{

case SPRemoteEventType.ItemAdding:

result.ChangedItemProperties.Add("Body", properties.ItemEventProperties.AfterProperties["Body"] += "\n \*\* For internal use only \*\* \n");

clientContext.Load(clientContext.Web);

clientContext.ExecuteQuery();

break;

case SPRemoteEventType.ItemDeleting:

result.ErrorMessage = "Items cannot be deleted from this list";

result.Status = SPRemoteEventServiceStatus.CancelWithError;

break;

}

}

}

return result;

}

public void ProcessOneWayEvent(RemoteEventProperties properties)

{

if (properties.EventType == RemoteEventType.ItemAdded)

{

using(Microsoft.SharePoint.Client.ClientContext ctx =

new Microsoft.SharePoint.Client.ClientContext(

properties.ItemEventProperties.WebUrl))

{

Microsoft.SharePoint.Client.List list =

ctx.Web.Lists.GetByTitle(

properties.ItemEventProperties.ListTitle);

ctx.Load(list);

Microsoft.SharePoint.Client.ListItem item =

list.GetItemById(

properties.ItemEventProperties.ListItemId);

ctx.Load(item);

ctx.ExecuteQuery();

item["Body"] +=

"\n Announcement Tracking ID: " +

Guid.NewGuid().ToString() + " \n";

item.Update();

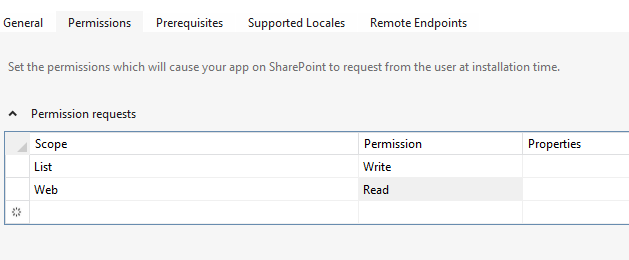
ctx.ExecuteQuery();

}

}

}

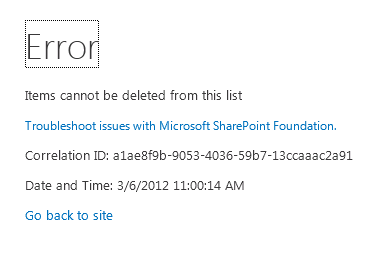
1. Set up security
   1. Open the file **AppManifest.xml**.
   2. Change the permission requests to **List = Write** and **Web = Read**.



## Task 3 – Test the App

In this task, you will run and test the App. The SharePoint 2013 developer site will not have access to call your local IIS Express development server running on localhost. We will deploy the app, which creates an Azure autohosted site that has a trust to the SharePoint developer site.

* 1. Select **Build⮚Deploy Solution** in Visual Studio, observe the **Output Window** for installation messages, and verify that **Internet Explorer** opens to the SharePoint Apps site you created before starting the lab.
  2. When the App appears, **click** the title of the list to access the detailed list view page.
  3. **Add** a new Announcement to the list.
  4. **Open** the newly added item and observe the changes to the announcement body.
  5. **Delete** an existing announcement. Observe the error message provided by our custom event receiver.



*Cancelling Events*

# **Demo 2: App Event Receivers**

In this exercise, you will create a Remote Event Receiver that handles events for an App. Using these event receivers, you can handle the Installed, Uninstalling, and Upgraded events for an App. This allows you better control over the App lifecycle.

## Task 1 – Create an Announcements List

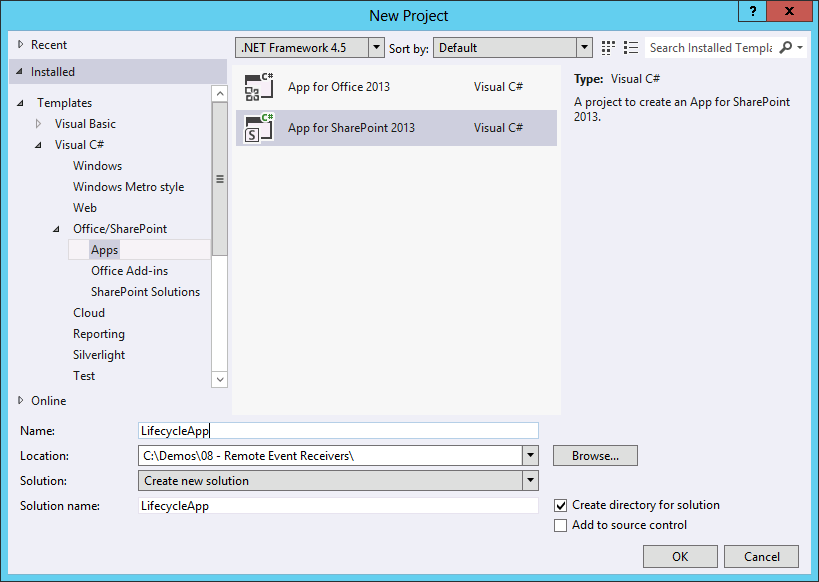
In this task, you will create a new Announcements list in your Office 365 developer site.

1. Open your browser to your developer site.
2. Click Site Contents to see the list of apps installed.
3. Click **Add an App**. Under “Apps You Can Add”, choose the **Announcements** app to create an announcements list.
4. Name the new list **Announcements**.

## Task 2 – Create a New App

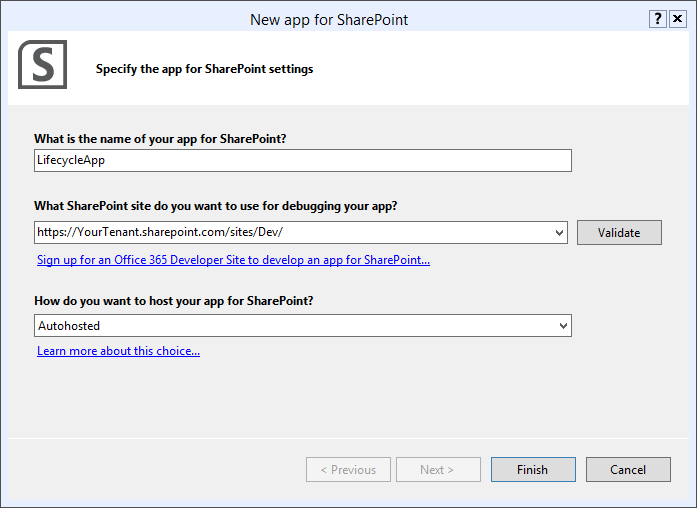
In this task, you will develop a new App in Visual Studio

1. Open **Microsoft Visual Studio 2012** and create a new SharePoint App project
   1. Open **Microsoft Visual Studio 2012**
   2. Select **File⮚New Project** from the main menu
   3. Click the **Templates⮚Visual C#⮚Office/SharePoint⮚Apps** node and select the **App for SharePoint 2013** project template
   4. Name the new project **LifecycleApp**
   5. Click the **OK** button



*Create new SharePoint App Project*

* 1. In the **New App for SharePoint Wizard**, name the App **Lifecycle App**
  2. Specify the Office 365 developer site you will use for this lab.
  3. Select **Autohosted** as the hosting type.
  4. Click **Finish**.



*New App for SharePoint Wizard*

## Task 2 – Create an Event Handler

In this task, you will create an event handler for the App.

1. Add the Event Handler
   1. In the Solution Explorer, click the **LifecycleApp** project.
   2. In the **Properties** dialog, set the **Handle App Uninstalling** property to **True**.
   3. In the Solution Explorer, click on the **LifecycleAppWeb** project.
   4. Open **AppEventReceiver.svc.cs** for editing.
   5. **Replace** the code contained in the **ProcessEvent** method with the following code:

SPRemoteEventResult result = new SPRemoteEventResult();

using (ClientContext clientContext = TokenHelper.CreateAppEventClientContext(properties, false))

{

if (clientContext != null)

{

if (properties.EventType == SPRemoteEventType.AppUninstalling)

{

List announcementsList = clientContext.Web.Lists.GetByTitle("Announcements");

ListItemCreationInformation itemCreateInfo = new ListItemCreationInformation();

ListItem newItem = announcementsList.AddItem(itemCreateInfo);

newItem["Title"] = "The app is being uninstalled!";

newItem["Body"] = "The app is being uninstalled at " + System.DateTime.Now.ToLongTimeString();

newItem.Update();

clientContext.ExecuteQuery();

clientContext.Load(clientContext.Web);

clientContext.ExecuteQuery();

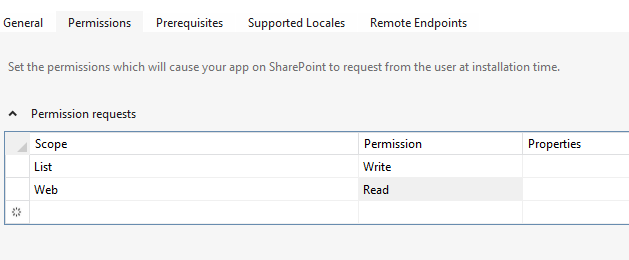
}

}

}

return result;

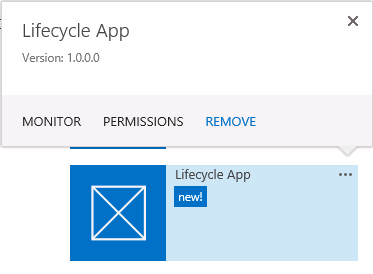
1. Set up security
   1. Open the file **AppManifest.xml**.
   2. Change the permission requests to **List = Write** and **Web = Read**.



## Task 3 – Test the App

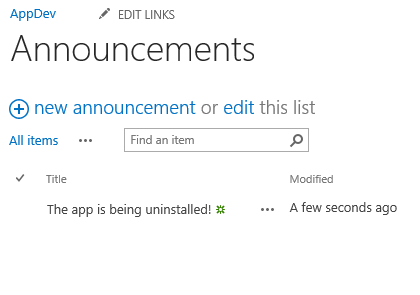
In this task, you will run and test the App. The SharePoint 2013 developer site will not have access to call your local IIS Express development server running on localhost. We will deploy the app, which creates an Azure autohosted site that has a trust to the SharePoint developer site.

1. Select **Build⮚Deploy Solution** in Visual Studio, observe the **Output Window** for installation messages, and verify that **Internet Explorer** opens to the SharePoint Apps site you created before starting the lab.
2. When the **Do you trust Lifecycle App** screen is shown, change the list dropdown to the **Announcements** list you created earlier in this lab and choose **Trust It**.
3. Now the app has been trusted and installed. The app will be displayed in the browser, click the back button once it renders.
4. Locate the App and select Remove.
   1. Click the ellipsis associated with the App, and select **Remove**.



*Remove the App*

1. Open the Announcements list and verify a new list item appears.



## Summary

In this lab, you learned how to create a Remote Event Receiver. You learned how to utilize internal security for the App to Event Receiver communication, and you learned how to create an Event Receiver for handling App Removal.