Developing SharePoint Add-ins with Remote Event Receivers

Lab Time: 60 minutes

Lab Folder: C:\Student\Modules\RemoteEventReceivers\Labs

Lab Overview: In this lab you will get hands-on experience developing a remote event receiver for a provider-hosted app. In particular, you will create a provider-hosted app which handles the **AppInstalled** all lifecycle event so that you can write custom C# code executes when the app is installed to create a list in the host web.

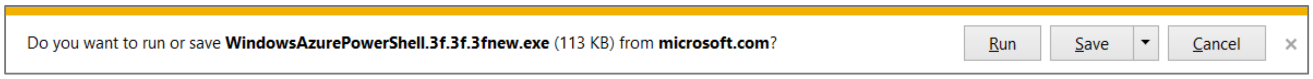
Exercise 1: Creating an Azure Service Bus Endpoint for Debugging

In this exercise, you will create a Windows Azure service bus namespace to use when you debug the remote event receiver you will create in the next exercise.

1. In order to complete your work for this lab you must install support for the PowerShell cmdlets that are provided by Microsoft Azure PowerShell. Go to the Microsoft Azure PowerShell Download Page using the following link and install Microsoft Azure PowerShell support before moving on to the next step.

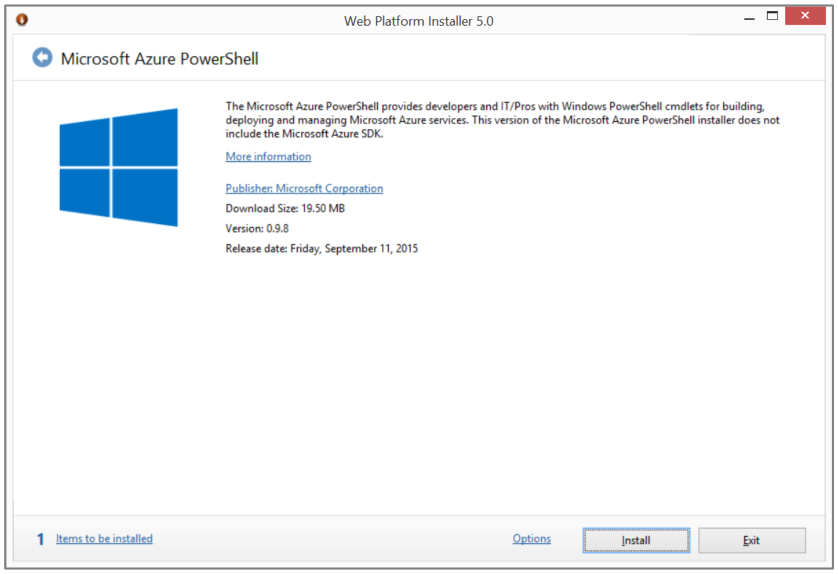
```
http://go.microsoft.com/fwlink/p/?linkid=320376&clcid=0x409
```

2. When you navigate to this link, it will download and installation file for Windows Azure PowerShell. Click Run to run the installation program.



Do you want to run or save **WindowsAzurePowerShell.3f.3f.3fnew.exe** (113 KB) from **microsoft.com**?

3. When the installation program starts, it will launch the Web Platform Installer and give you the option to install the **Microsoft azure PowerShell** module. Click the Install button to continue with the installation.



Web Platform Installer 5.0

Microsoft Azure PowerShell

The Microsoft Azure PowerShell provides developers and IT/Pros with Windows PowerShell cmdlets for building, deploying and managing Microsoft Azure services. This version of the Microsoft Azure PowerShell installer does not include the Microsoft Azure SDK.

More information

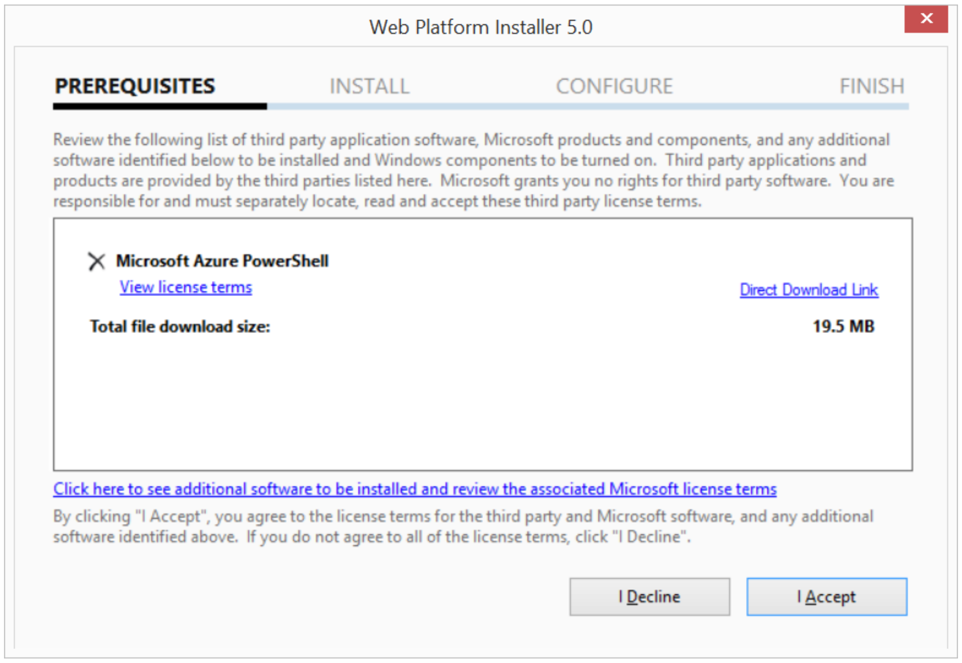
Publisher: Microsoft Corporation

Download Size: 19.50 MB

Version: 0.9.8

Release date: Friday, September 11, 2015

4. On the prerequisites page, click the **I Accept** button to continue with the installation.


Web Platform Installer 5.0

Review the following list of third party application software, Microsoft products and components, and any additional software identified below to be installed and Windows components to be turned on. Third party applications and products are provided by the third parties listed here. Microsoft grants you no rights for third party software. You are responsible for and must separately locate, read and accept these third party license terms.

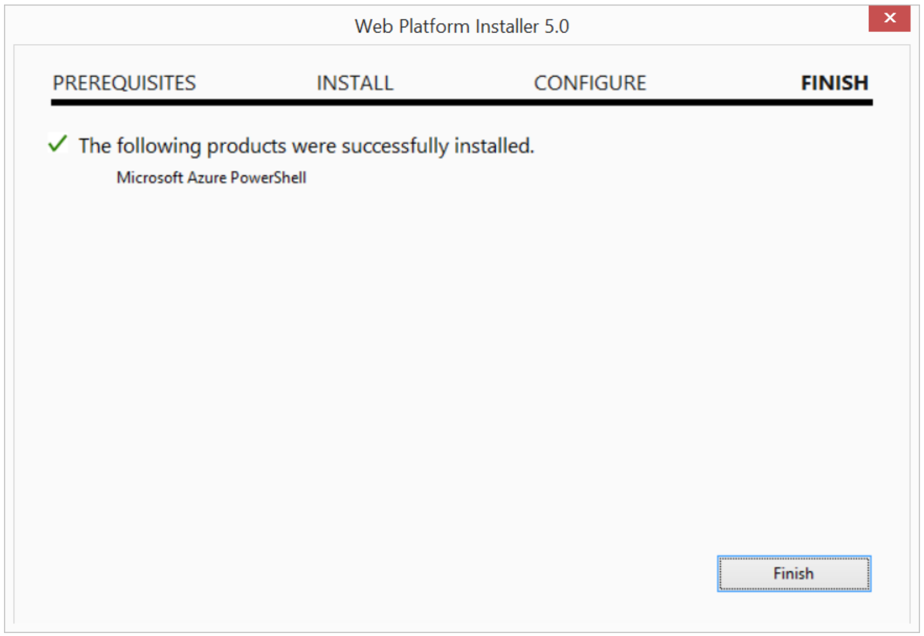
Microsoft Azure PowerShell
View license terms
Direct Download Link

Total file download size: 19.5 MB

Click here to see additional software to be installed and review the associated Microsoft license terms

By clicking "I Accept", you agree to the license terms for the third party and Microsoft software, and any additional software identified above. If you do not agree to all of the license terms, click "I Decline".

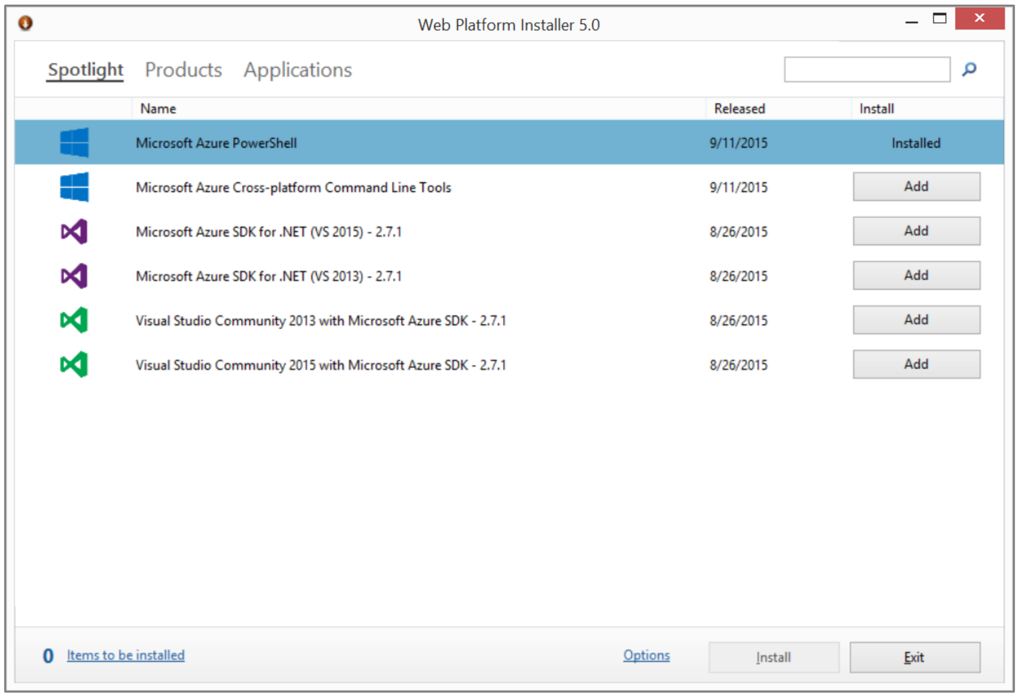
5. When you see the following dialog, click **Finish**.


Web Platform Installer 5.0

✓ The following products were successfully installed.

Microsoft Azure PowerShell

6. At this point you should have Microsoft Azure PowerShell support installed on your local developer workstation. Click the **Exit** button to exit the Web Platform Installer.



Name	Released	Install
Microsoft Azure PowerShell	9/11/2015	Installed
Microsoft Azure Cross-platform Command Line Tools	9/11/2015	Add
Microsoft Azure SDK for .NET (VS 2015) - 2.7.1	8/26/2015	Add
Microsoft Azure SDK for .NET (VS 2013) - 2.7.1	8/26/2015	Add
Visual Studio Community 2013 with Microsoft Azure SDK - 2.7.1	8/26/2015	Add
Visual Studio Community 2015 with Microsoft Azure SDK - 2.7.1	8/26/2015	Add

7. Press the **Windows** key to navigate to the Windows start page and type **"Azure PowerShell"**. Click on the **Microsoft Azure PowerShell** tile to launch the Azure PowerShell Console window.

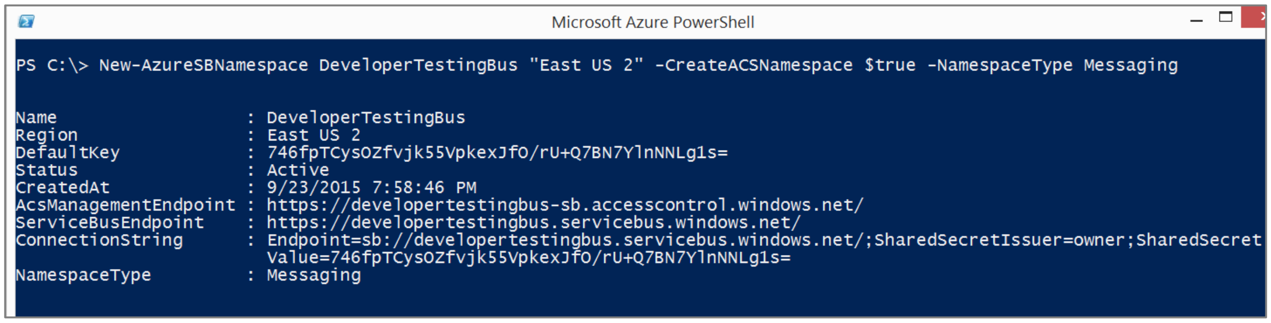
Bare careful. In addition to the **Microsoft Azure PowerShell** tile, there might also be a tile for **Windows Azure Active Directory Module for PowerShell**. In this exercise, you want to launch **Microsoft Azure PowerShell** console and not the Powershell console window which is specific to Azure Active Directory.

8. In the Azure PowerShell Console window, execute the **Add-AzureAccount** cmdlet. When you execute this cmdlet, you will be prompted for login credentials. Log in with the credentials for your organizational account.

9. In the Azure PowerShell Console window, execute the **New-AzureSBNamespace** cmdlet to create a service bus namespace with a name such as **DeveloperTestingBus**. You should replace **East US 2** with a different Azure location if that is more appropriate for your geographical location.

```
New-AzureSBNamespace DeveloperTestingBus "East US 2" -CreateACSNamespace $true -NamespaceType Messaging
```

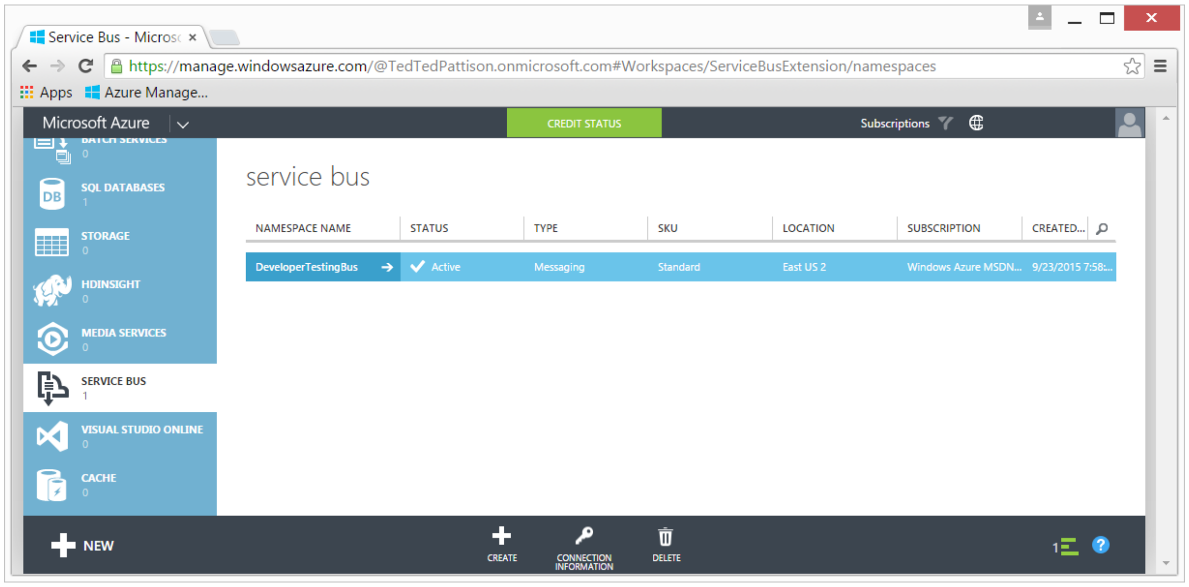
10. When you execute the **New-AzureSBNamespace** cmdlet, you should be able to observe that the cmdlet has executed successfully and created a new service bus namespace with ACS support.



```
PS C:\> New-AzureSBNamespace DeveloperTestingBus "East US 2" -CreateACSNamespace $true -NamespaceType Messaging


Name                    : DeveloperTestingBus
Region                  : East US 2
DefaultKey              : 746fpTCysOZfvjk55VpkexJfO/rU+Q7BN7YlnNNLg1s=
Status                  : Active
CreatedAt               : 9/23/2015 7:58:46 PM
AcsManagementEndpoint   : https://developertestingbus-sb.accesscontrol.windows.net/
ServiceBusEndpoint      : https://developertestingbus.servicebus.windows.net/
ConnectionString        : Endpoint=sb://developertestingbus.servicebus.windows.net/;SharedSecretIssuer=owner;SharedSecret
                          Value=746fpTCysOZfvjk55VpkexJfO/rU+Q7BN7YlnNNLg1s=
NamespaceType           : Messaging
```

11. Switch over to the browser and navigate to the Azure Management Portal at **https://manage.windowsazure.com**. Log in using the credentials associated with your organizational account. At this point, the Azure Management Portal should display the Azure objects that are associated with your Azure subscription.

12. Locate and click the **Service Bus** link in the left-hand navigation menu.

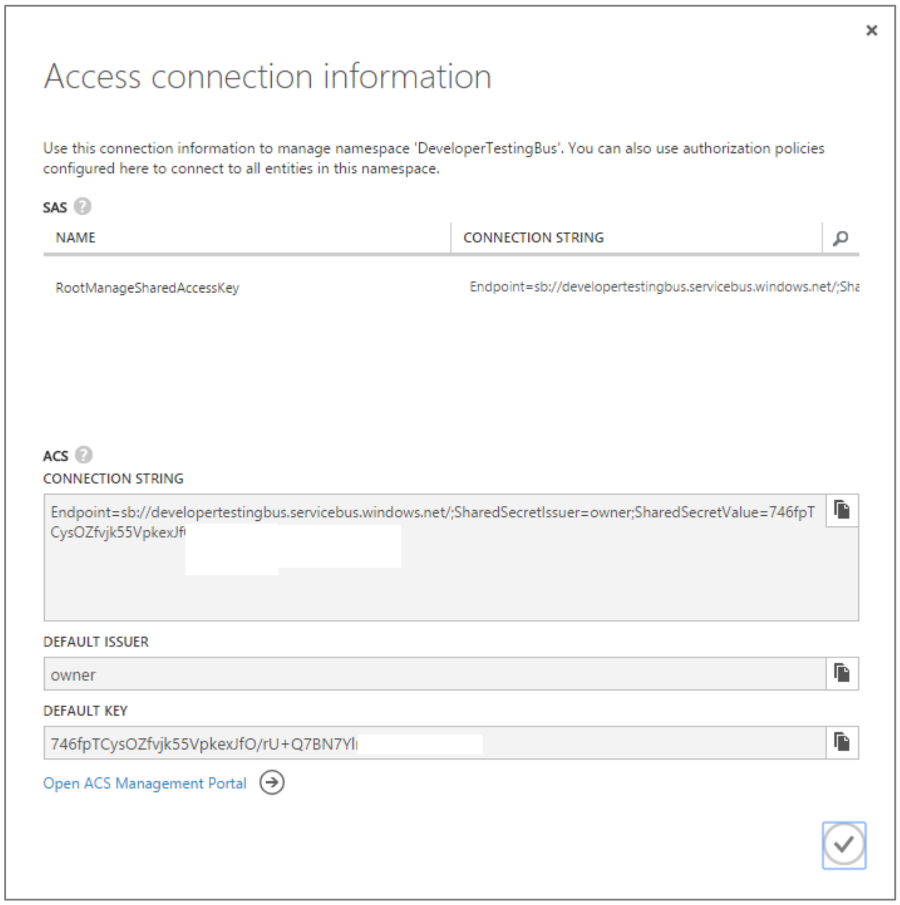


service bus

NAMESPACE NAME	STATUS	TYPE	SKU	LOCATION	SUBSCRIPTION	CREATED...
DeveloperTestingBus	Active	Messaging	Standard	East US 2	Windows Azure MSDN...	9/23/2015 7:58:...

13. On the service bus page, you should be able to see the service bus namespace you just created using Azure PowerShell. Select this namespace and then click the **CONNECTION INFORMATION** button at the bottom of the page.

14. On the **Access connection information** page, locate the **ACS CONNECTION STRING** setting and copy its value to the Windows clipboard.



Access connection information

Use this connection information to manage namespace 'DeveloperTestingBus'. You can also use authorization policies configured here to connect to all entities in this namespace.

SAS

NAME	CONNECTION STRING
RootManageSharedAccessKey	Endpoint=sb://developertestingbus.servicebus.windows.net/;Sha

ACS

CONNECTION STRING

Endpoint=sb://developertestingbus.servicebus.windows.net/;SharedSecretIssuer=owner;SharedSecretValue=746fpTCysOZfvjk55VpkexJf[redacted]

DEFAULT ISSUER

owner

DEFAULT KEY

746fpTCysOZfvjk55VpkexJfO/rU+Q7BN7Yl[redacted]

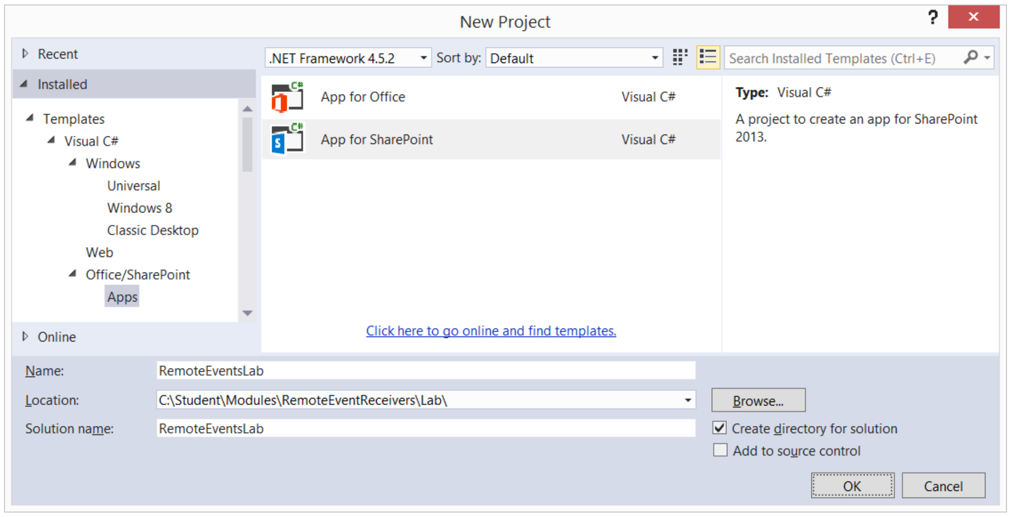
Open ACS Management Portal

15. Open **NOTEPAD.EXE** and paste the value of the **ACS CONNECTION STRING** setting into a new text file. Save this file so that you can use this value in the following exercise.

Exercise 2: Creating A Remote Event Receiver for the AppInstalled event

In this exercise you will create a provider-hosted app which includes a remote event receiver to handle the AppInstalled event.

1. Make sure your developer test site is up and running so you have a place to debug the app you are going to create.
2. Create a new project in Visual Studio 2015 by selecting File > New > Project.
   a) In the **New Project** dialog, select the **App for SharePoint 2013** template under the **Templates > Visual C# > Office / SharePoint > Apps** section. Give the new app project a name of **RemoteEventsLab** and click **OK** to create the project.



Type: Visual C#

A project to create an app for SharePoint 2013.

Name: RemoteEventsLab

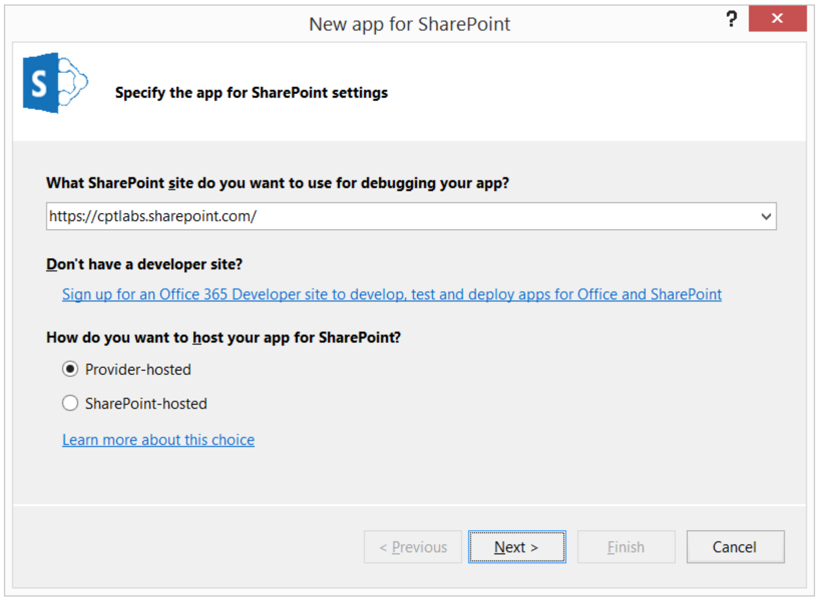
Location: C:\Student\Modules\RemoteEventReceivers\Lab\

Solution name: RemoteEventsLab

☑ Create directory for solution

☐ Add to source control

   b) Enter the URL for your developer site and make sure the project is created as a **Provider-hosted** add-in. Click **Next**.


New app for SharePoint
Sign up for an Office 365 Developer site to develop, test and deploy apps for Office and SharePoint
Learn more about this choice
< Previous
Next >
Finish
Cancel

Specify the app for SharePoint settings

What SharePoint site do you want to use for debugging your app?

https://cptlabs.sharepoint.com/

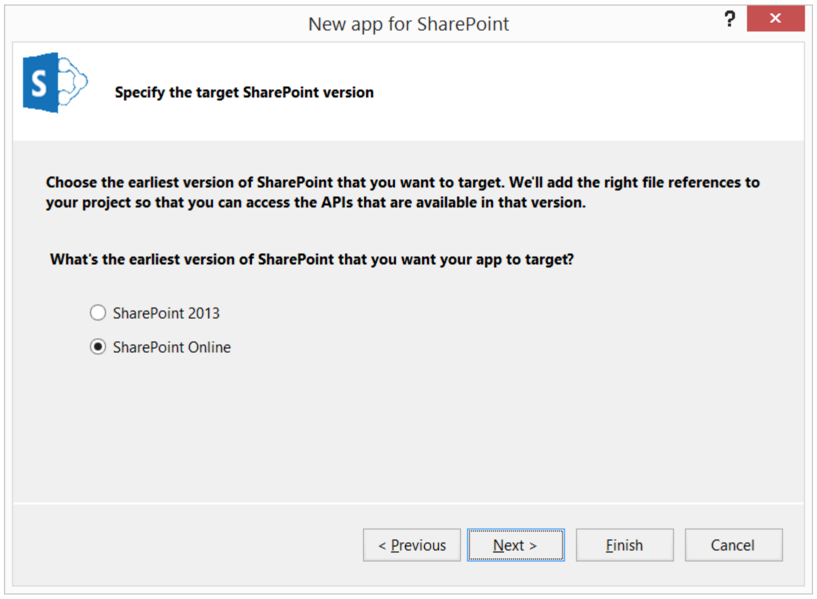
Don't have a developer site?

How do you want to host your app for SharePoint?

● Provider-hosted

○ SharePoint-hosted

   c) Select **SharePoint Online** as the target platform and click **Next**.


New app for SharePoint

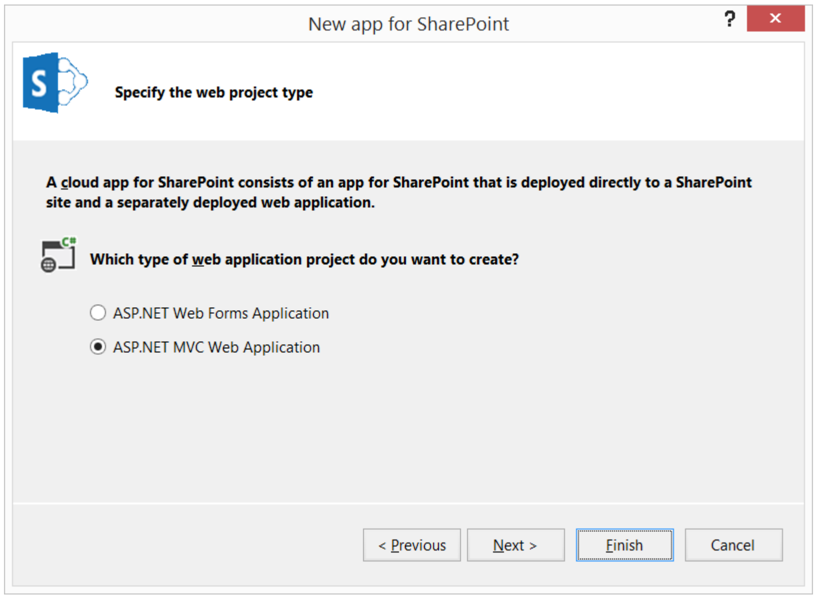
Specify the target SharePoint version

Choose the earliest version of SharePoint that you want to target. We'll add the right file references to your project so that you can access the APIs that are available in that version.

What's the earliest version of SharePoint that you want your app to target?

SharePoint 2013 [✓ SharePoint Online]

d) Select **ASP.NET MVC Web Application** and click **Next**.


New app for SharePoint

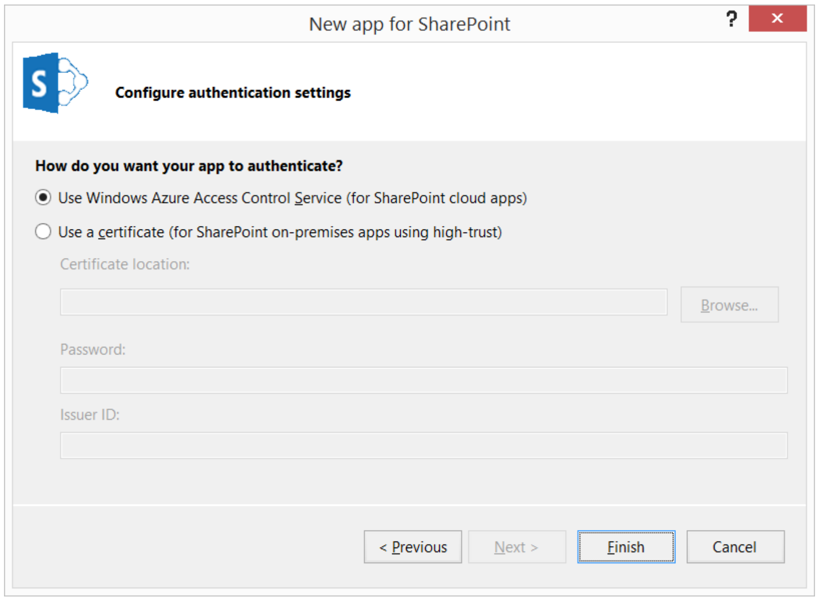
Specify the web project type

A cloud app for SharePoint consists of an app for SharePoint that is deployed directly to a SharePoint site and a separately deployed web application.

Which type of web application project do you want to create?

ASP.NET Web Forms Application [✓ ASP.NET MVC Web Application]

e) On the final page of the **New app for SharePoint** wizard, select **Use Windows azure Access Control Service** and click **Finish**.


New app for SharePoint

Configure authentication settings

How do you want your app to authenticate?

(●) Use Windows Azure Access Control Service (for SharePoint cloud apps)

( ) Use a certificate (for SharePoint on-premises apps using high-trust)

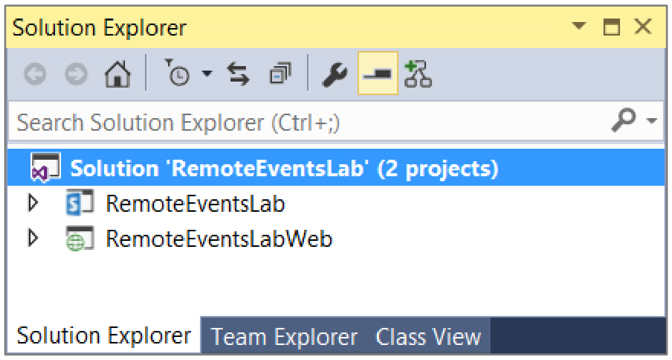
Certificate location:

Password:

Issuer ID:

f) At this point, Visual Studio should finish creating the new Visual Studio.

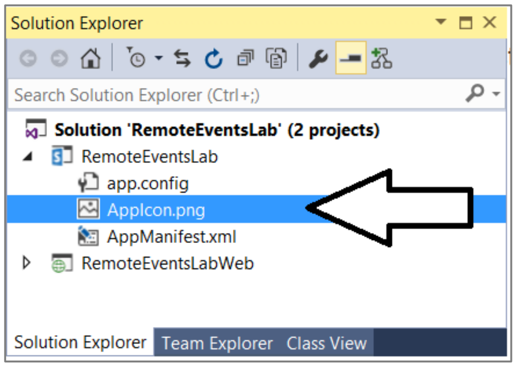
3. Visual Studio has now created a new solution with two projects named **RemoteEventsLab** and **RemoteEventsLabWeb**.


Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'RemoteEventsLab' (2 projects)
RemoteEventsLab
RemoteEventsLabWeb

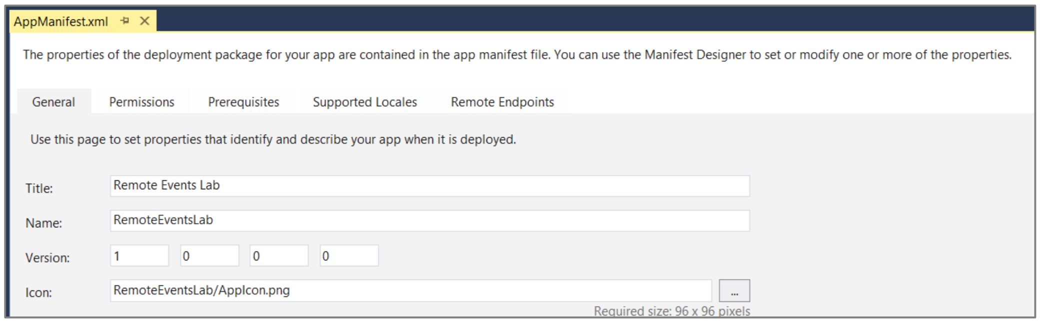
4. In the top project, locate and double-click on **AppManifest.xml** to open the app manifest in the Visual Studio app manifest designer.


Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'RemoteEventsLab' (2 projects)
RemoteEventsLab
app.config
AppIcon.png
AppManifest.xml
RemoteEventsLabWeb

5. On the **General** tab of the **App Manifest Designer**, modify the app **Title** property **Remote Events Lab**.



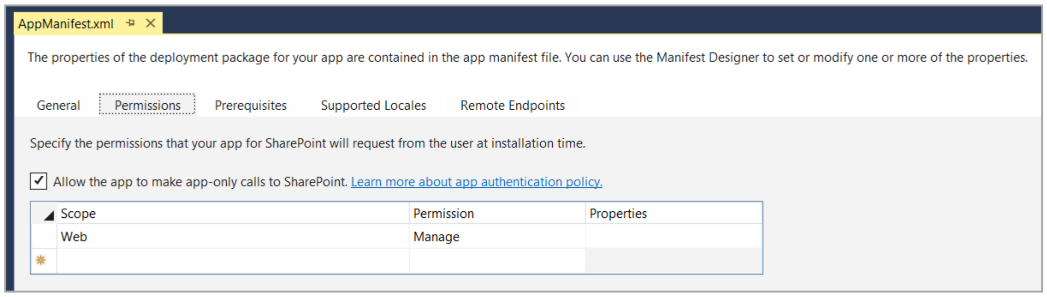
The properties of the deployment package for your app are contained in the app manifest file. You can use the Manifest Designer to set or modify one or more of the properties.

Use this page to set properties that identify and describe your app when it is deployed.

Title:	Remote Events Lab
Name:	RemoteEventsLab
Version:	1 0 0 0
Icon:	RemoteEventsLab/AppIcon.png

Required size: 96 x 96 pixels

6. On the **Permissions** tab of the App Manifest Designer, make sure the checkbox for the **Allow the app to make app-only calls** setting is checked. Add a permission request that is scoped to **Web** and configured with **Manage** permissions as shown in the following screenshot.



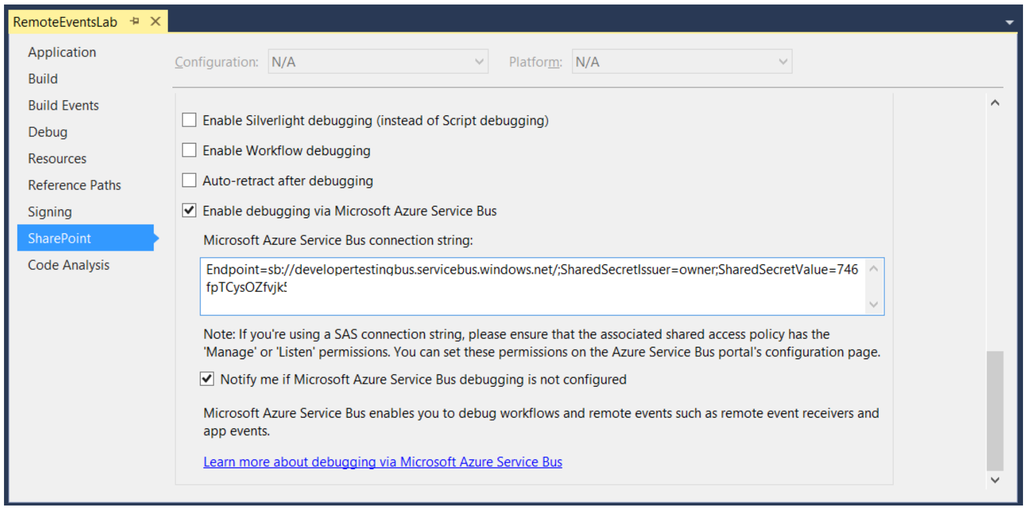
The properties of the deployment package for your app are contained in the app manifest file. You can use the Manifest Designer to set or modify one or more of the properties.

Specify the permissions that your app for SharePoint will request from the user at installation time.

☑ Allow the app to make app-only calls to SharePoint. Learn more about app authentication policy.

Scope	Permission	Properties
Web	Manage	

7. Save and close **AppManifest.xml**.
8. In the Solution Explorer, right-click on the **RemoteEventLabs** project node and select **Properties**.
9. Select the **SharePoint** tab of the **Project Properties** page and scroll down to the bottom of the page. Check the checkbox with the **Enable debugging via Microsoft Azure Service Bus** option. Paste the ACS connection string you obtained in the previous exercise into the textbox as shown in the following screenshot.



Configuration: N/A Platform: N/A

☐ Enable Silverlight debugging (instead of Script debugging)
☐ Enable Workflow debugging
☐ Auto-retract after debugging
☑ Enable debugging via Microsoft Azure Service Bus

Microsoft Azure Service Bus connection string:

```
Endpoint=sb://developertestingbus.servicebus.windows.net/;SharedSecretIssuer=owner;SharedSecretValue=746
fpTCysOZfvjk!
```

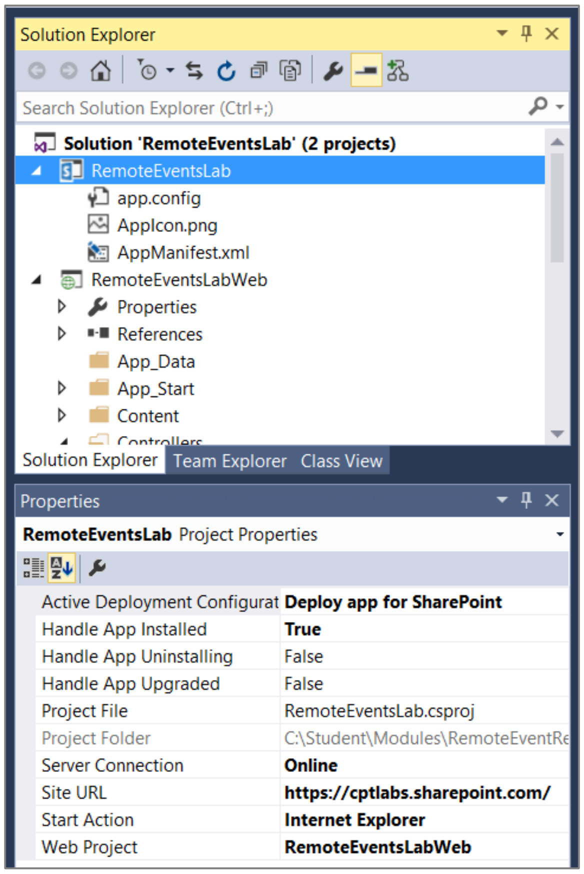
Note: If you're using a SAS connection string, please ensure that the associated shared access policy has the 'Manage' or 'Listen' permissions. You can set these permissions on the Azure Service Bus portal's configuration page.

☑ Notify me if Microsoft Azure Service Bus debugging is not configured

Microsoft Azure Service Bus enables you to debug workflows and remote events such as remote event receivers and app events.

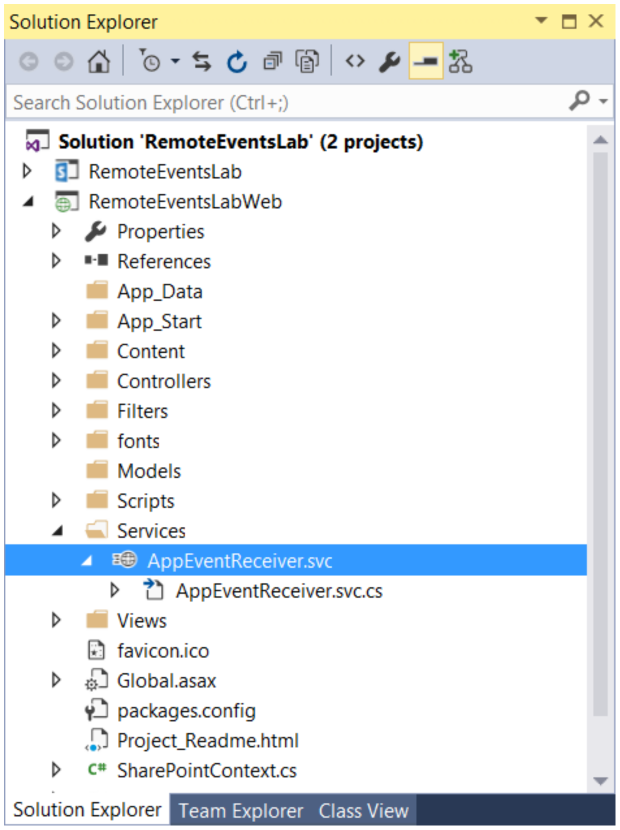
Learn more about debugging via Microsoft Azure Service Bus

10. Save and close the Project Properties for the **RemoteEventsLab** project.
11. In Solution Explorer, select the top-level node for the **RemoteEventsLab** project and then inspect the project's property sheet. Locate The **Handle App Installed** project property in the **App for SharePoint Events** section and change its value to **True**.



Active Deployment Configurat	Deploy app for SharePoint
Handle App Installed	True
Handle App Uninstalling	False
Handle App Upgraded	False
Project File	RemoteEventsLab.csproj
Project Folder	C:\Student\Modules\RemoteEventRe
Server Connection	Online
Site URL	https://cptlabs.sharepoint.com/
Start Action	Internet Explorer
Web Project	RemoteEventsLabWeb

12. Move down to the bottom project named **RemoteEventsLabWeb**. You should see the C# file named **HomeController.cs** which provides the MVC controller for the app. You should also notice that the project contains a new web service file named **AppEventReceiver.svc** along with an associated code-behind file named **AppEventReceiver.svc.cs** that were created when you configured support for the **AppInstalled** event handler.

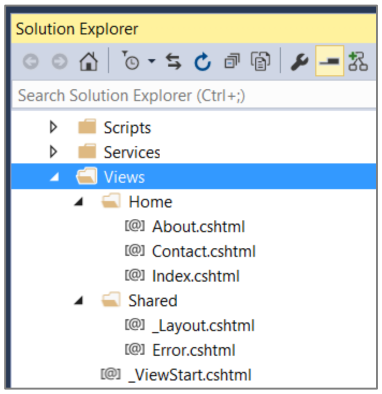


13. Open **HomeController.cs** and replace the existing implementation of the **Index** method with the following code which uses CSOM to obtain the set of non-hidden lists that exist within the host web and to pass this set of lists to the associated MVC view using the MVC **ViewBag** object.

```
[SharePointContextFilter]
public ActionResult Index() {

  var spContext = SharePointContextProvider.Current.GetSharePointContext(HttpContext);

  using (var clientContext = spContext.CreateUserClientContextForSPHost()) {
    if (clientContext != null) {
      clientContext.Load(clientContext.Web);
      ListCollection Lists = clientContext.Web.Lists;
      clientContext.Load(Lists, lists => lists.Where(list => !list.Hidden)
                                        .Include(list => list.Title,
      list => list.DefaultViewUrl));
      clientContext.ExecuteQuery();

      Dictionary < string, string> userLists = new Dictionary<string, string>();
      string siteUrl = clientContext.Web.Url;
      foreach (var list in Lists) {
        userLists.Add(siteUrl + list.DefaultViewUrl, list.Title);
      }
      ViewBag.UserLists = userLists;
    }
  }
  return View();
}
```

14. Save and close **HomeController.cs**.

15. Expand the **Views** folder and the two child folders inside so that you can see all the razor view files.



16. Open the _**Layouts.cshtml** file in the Shared view folder and replace its contents with the following razor code.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Remote Events Lab</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
</head>
<body>
  <div class="container body-content">
    <h2>Remote Events Lab</h2>
    <hr />
    @RenderBody()
  </div>
  @Scripts.Render("~/bundles/jquery")
  @Scripts.Render("~/bundles/bootstrap")
  @Scripts.Render("~/bundles/spcontext")
  @RenderSection("scripts", required: false)
</body>
</html>
```

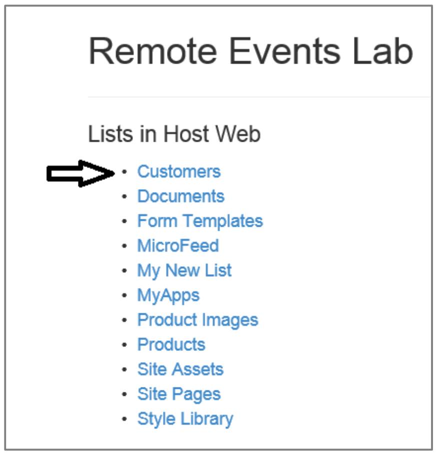
17. Save and close _**Layouts.cshtml**.
18. Open the **Index.cshtml** file in the **Home** view folder and replace its contents with the following razor code which will display the set of non-public lists from the host web.

```
<h4>Lists in Host Web</h4>

@{
  @Html.Raw("<ul>");
  foreach (var list in ViewBag.UserLists) {
    string html = "<li><a href='" + list.Key + "' >" + list.Value + "</a></li>";
    @Html.Raw(html);
  }
  @Html.Raw("</ul>");
}
```

19. Save and close **Index.cshtml**.
20. Open **AppEventReceiver.svc.cs**.
21. Replace the implementation of **ProcessEvent** with the following code.

```
public SPRemoteEventResult ProcessEvent(SPRemoteEventProperties properties) {

  SPRemoteEventResult result = new SPRemoteEventResult();
  using (ClientContext clientContext =
           TokenHelper.CreateAppEventClientContext(properties, useAppWeb: false)) {
    if (clientContext != null) {
      clientContext.Load(clientContext.Web);
      string listTitle = "Customers";

      // delete list if it exists
      ExceptionHandlingScope scope = new ExceptionHandlingScope(clientContext);
      using (scope.StartScope()) {
        using (scope.StartTry()) {
          clientContext.Web.Lists.GetByTitle(listTitle).DeleteObject();
        }
        using (scope.StartCatch()) { }
      }

      // create and initialize ListCreationInformation object
      ListCreationInformation listInformation = new ListCreationInformation();
      listInformation.Title = listTitle;
      listInformation.Url = "Lists/Customers";
      listInformation.QuickLaunchOption = QuickLaunchOptions.On;
      listInformation.TemplateType = (int)ListTemplateType.Contacts;

      // Add ListCreationInformation to lists collection and return list object
      List list = clientContext.Web.Lists.Add(listInformation);

      // modify additional list properties and update
      list.OnQuickLaunch = true;
      list.EnableAttachments = false;
      list.Update();

      // send command to server to create list
      clientContext.ExecuteQuery();

      // create a sample item in the list
      var customer1 = list.AddItem(new ListItemCreationInformation());
      customer1["Title"] = "Mike";
      customer1["FirstName"] = "Fitzmaurice";
      customer1.Update();

      // send command to server to create item
      clientContext.ExecuteQuery();
    }
  }
  return result;
}
```

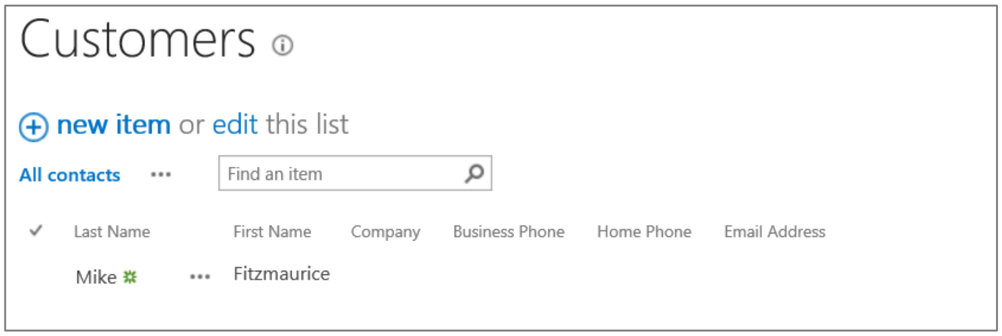
22. Test your work by pressing the **{F5}** key and launching the app in the Visual Studio debugger. When the app loads and displays the start page, you should see that **Customers** list created by the event handler for the **AppInstalled.**


Remote Events Lab

Lists in Host Web

- Customers
- Documents
- Form Templates
- MicroFeed
- My New List
- MyApps
- Product Images
- Products
- Site Assets
- Site Pages
- Style Library

23. Click on the link for the **Customers** list to navigate to the default view of this list.



Customers

new item or edit this list

All contacts

Find an item

	Last Name	First Name	Company	Business Phone	Home Phone	Email Address
	Mike	Fitzmaurice				

24. Close the browser and terminate the debugging session.