

# Automating Working with SharePoint 2013 Workflow

**Lab Time:** 60 minutes

**Lab Folder:** C:\Student\Modules\Workflow

**Lab Overview:** This hands-on lab will walk you through the experience of creating workflows using both SharePoint Designer 2013 and Visual Studio 2013. You will get first person experience in using the new activities, loops and stages capabilities in SharePoint Designer as well as creating workflows that are used in SharePoint apps. Further, you will also see how to create workflows that communicate with OData web services.

## 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://workflow.wingtip.com>.

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

**C:\Student\Modules\Workflow**
  - c) When the script completes, it will launch a new browser and navigate to the lab site collection.
  - d) Close the PowerShell console window.

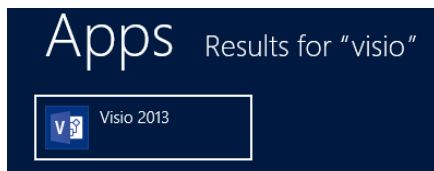
## Exercise 2: Visio 2013 & SharePoint Designer 2013 Workflows

In this exercise you will create a new workflow using Visio 2013 and SharePoint Designer 2013. In this workflow you will leverage the new support for stages.

1. Ensure you are logged into the **WingtipServer** server as **WINGTIP\administrator**.

## Model the Workflow Using Visio 2013

2. Open **Visio 2013: Windows Keyboard Key → Type “visio”** and then select the Visio 2013 application.



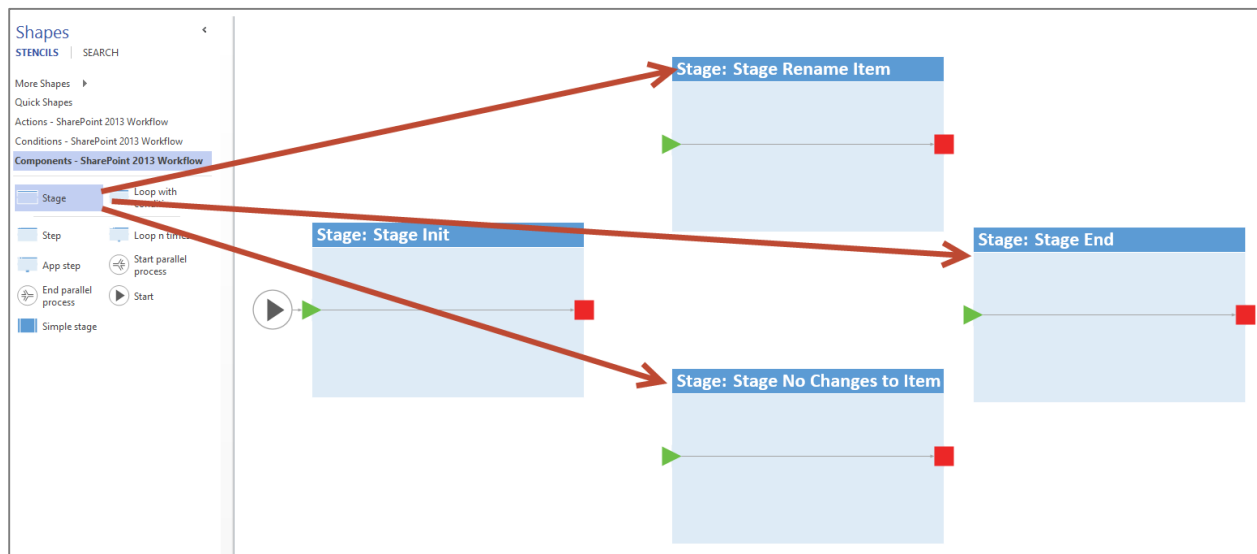
3. When **Visio** loads, click the **Flowchart** category on the **New** document page:



4. Select the template **Microsoft SharePoint 2013 Workflow**, followed by the **Create** button.



5. When the designer loads, right-click the dark blue heading of **Stage 1** and select **Edit Text**. Rename the stage to **Stage Init**.
6. Add three more stages to the designer by dragging the **Stage** shape from the **Shapes** task pane and rename them to the following stages as shown in the follow figure:
  - a) **Stage Rename Item**
  - b) **Stage No Changes to Item**
  - c) **Stage End**

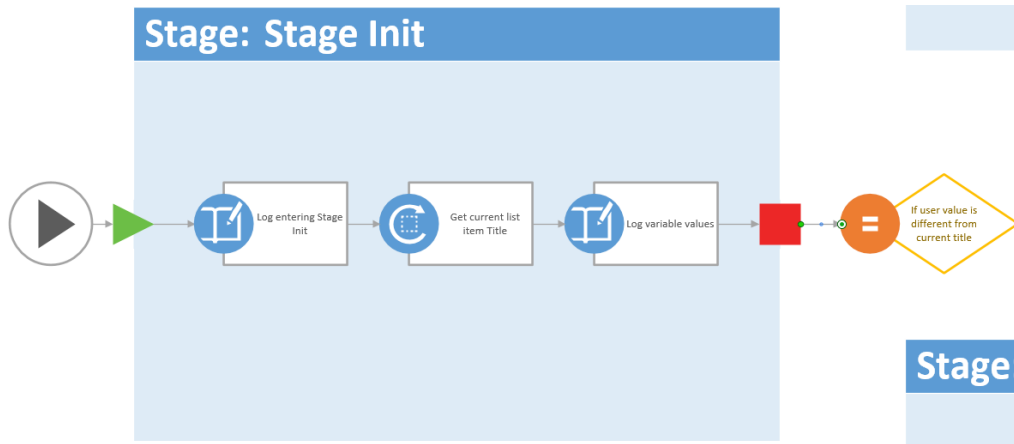


7. Add activities to **Stage Init**:
  - a) Using the **Shapes** pane, select the **Actions – SharePoint 2013 Workflow** category and add the following actions to **Stage Init**:
    - i) **Log to history list**
    - ii) **Set workflow variable**
    - iii) **Log to history list**

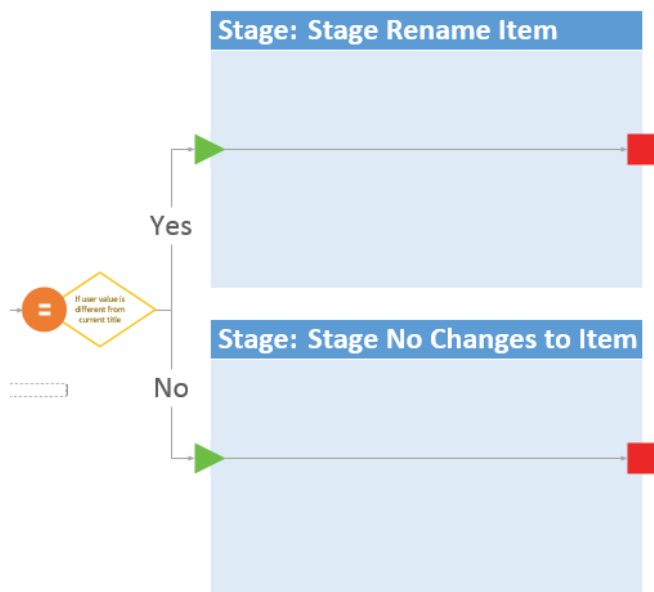
When you are dragging-and-dropping actions from the Visual Studio toolbox on the workflow designer, make sure you are dropping these actions on the connector (i.e the black line connecting the green triangle and red square) within the stage. If you don't do this, you will have to fix up the connections before saving the workflow.

8. Rename the actions by right-clicking each one and selecting **Edit Text**. Give each action a descriptive name as follows:
  - a) **Log to history list** = Log entering Stage Init
  - b) **Set workflow variable** = Get current list item Title
  - c) **Log to history list** = Log variable values

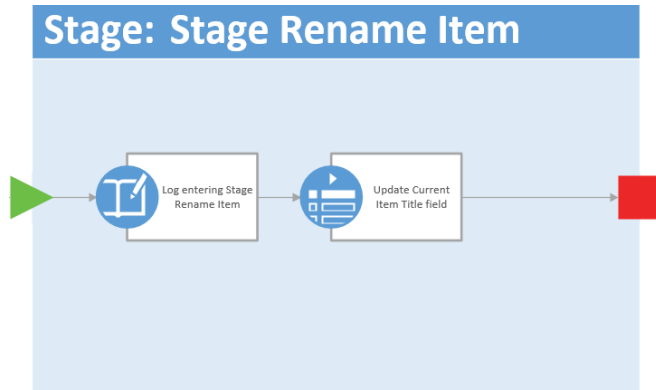
9. Using the **Shapes** pane, select the **Conditions – SharePoint 2013 Workflow** category and add an **If any value equals value** condition to the workflow immediately to the right of **Stage Init** (i.e. to the right of the red square).
  - a) Rename this condition to **If user value is different from current title**.
10. Using the **Connector** tool, found in the **Home** tab of the ribbon in the **Tools** group, connect the red box from **Stage Init** to the condition you just added.
  - a) To Connect the red box with the orange circle, click on the **Connector** tool in on the Home tab and then place your **mouse pointer near the right hand edge of the red box** (you should see a **small green square** show up
  - b) Click on the small green square on the right edge of the red box and drag towards the orange circle until you see the small green square appear on the left edge of the orange circle.
  - c) Release the left mouse button.
11. The **Stage Init** should look like the following figure:



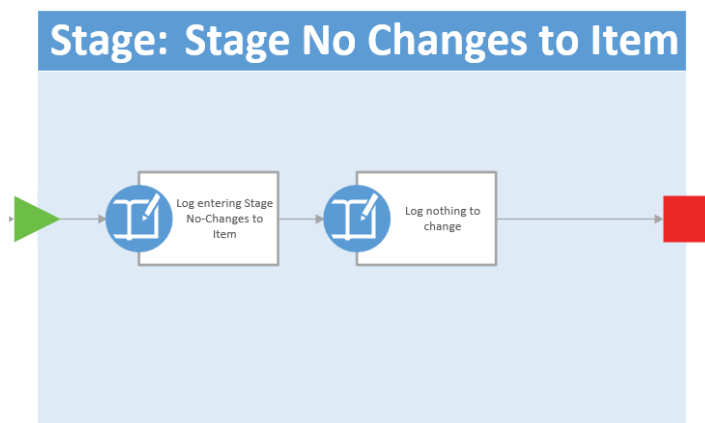
12. Using the same technique with the Connector tool, draw two connections from the condition, one to the start arrow of **Stage Rename Item** and one to the **Stage No Changes to Item**.
13. Right-click the connector to **Stage Rename Item** and select **Yes**.
14. Right-click the connector to **Stage No Changes to Item** and select **No**.
15. The condition connections should look like the following figure:



16. Use the same techniques you just applied, update **Stage Rename Item** by adding the following actions and renaming them it so it looks like the following figure:
- a) **Log to history list** = Log entering Stage Rename Item
  - b) **Set field in current item** = Update current item Title field



17. Update the Stage No Changes to Item by adding the following actions so it looks like the following figure:
- a) **Log to history list** = Log entering Stage No Changes to Item
  - b) **Log to history list** = Log nothing to change



18. Using the **Connector** tool, connect the terminating red square on the right of **Stage Rename Item** to the green arrow on **Stage End**.
19. Repeat the previous step connecting the **Stage No Changes to Item** to **Stage End**.
20. Add a **Log to history list** action to the **Stage End** stage.
- a) Rename it to **Log Stage End**
21. Save your changes selecting **File → Save**.
- a) When prompted, save the file to the **desktop** with a filename of **RenamelItemWorkflow.vsd**.

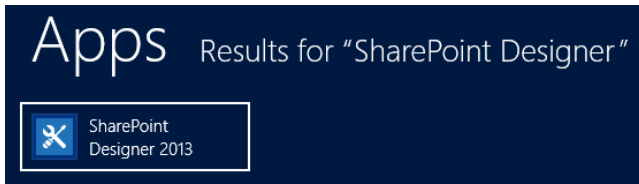
A copy of the drawing can be found in the **C:\Student\Modules\Workflow\Lab\Solution** folder in this lab's files

22. Close **Visio 2013**.

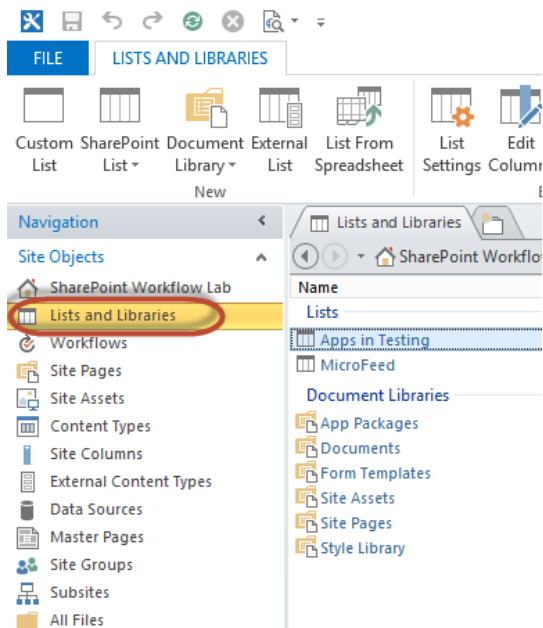
At this point you have designed your workflow using Visio 2013. This simulates the process of creating a workflow with a business user without access to SharePoint Designer 2013 or a connection to SharePoint 2013.

## Complete the Workflow in SharePoint Designer 2013

23. Open SharePoint Designer 2013: Windows Keyboard Key → Type “SharePoint Designer” and then select the SharePoint Designer 2013 application.



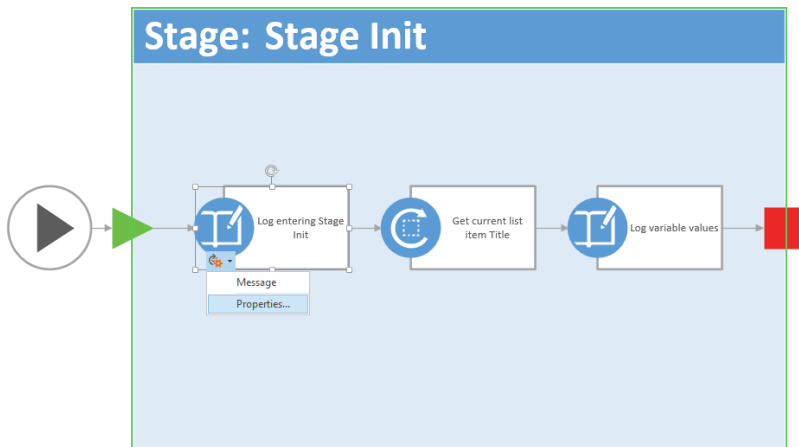
24. Open the lab site collection:
- Click the **Open Site** button.
  - In the **Open Site** dialog, enter <http://workflow.wingtip.com> in the **Site Name** box and click **Open**.
  - If prompted to login, use the credentials for **WINGTIP\administrator**.
25. When SharePoint Designer loads, create a new announcement list:
- Select **Lists and Libraries** in the **Navigation** pane.



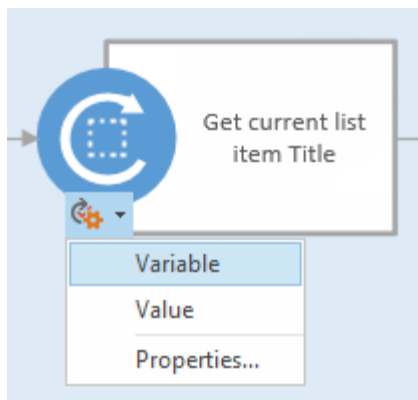
- In the ribbon, select **SharePoint List** → **Announcements**.
  - In the **Create list or document library** dialog, set the **Name** to **Announcements** and click **OK**.
26. Import the workflow created with Visio 2013:
- Select **Workflows** in the **Navigation** pane.
  - In the ribbon's **Manage** group, select **Import from Visio** → **Import Visio 2013 Diagram**.
  - Find the **RenameItemWorkflow.vsd** drawing you created on your desktop using Visio 2013 and click **Open**.
  - In the **Create Workflow** dialog, set the following values and click **OK**:
    - Name:** Rename Item Workflow
    - Workflow Type:** List Workflow
    - SharePoint List:** Announcements

When the workflow loads you'll notice it looks identical to the drawing in Visio. In fact, it almost seems like it opened Visio, but if you look at the very top of the application, you'll see you still have the <http://workflow.wingtip.com> site open in SharePoint Designer 2013.

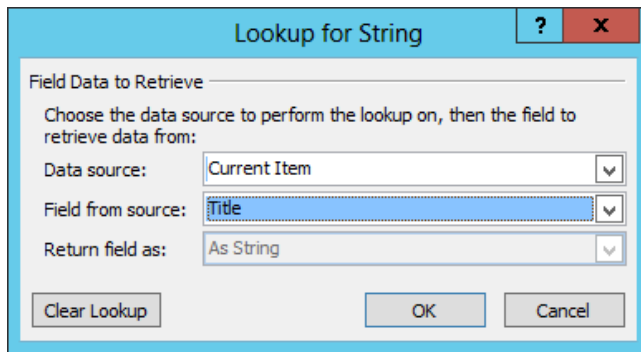
27. Before proceeding create a new variable that will trigger a form to appear when the user starts the workflow. This will prompt the user to optionally enter a new value for the title of the announcement.
- In the ribbon, click the **Initiation Form Parameters** button in the **Variables** group.
  - In the **Association and Initiation Form Parameters** dialog, click **Add**.
  - Enter the following values and click **Next**:
    - Field Name:** New Announcement Title
    - Description:** Do you want to change the title of the announcement?
  - Click **Next** and then click **Finish**.
  - Click **OK**
28. Select the first action **Log entering Stage Init** in **Stage Init**. Notice a little tile appears in the lower left corner. Click it and select **Properties**:



- In the **Log to History List Properties** dialog, set the **Message** to **Entering Stage Init...** by clicking the ... button, entering the text and clicking **OK** twice to accept your changes.
29. Open the **Get Current List Item Title** action's **Properties** dialog using the same process in the last step to get the tile to appear, except don't select **Properties...** instead select **Variable**.



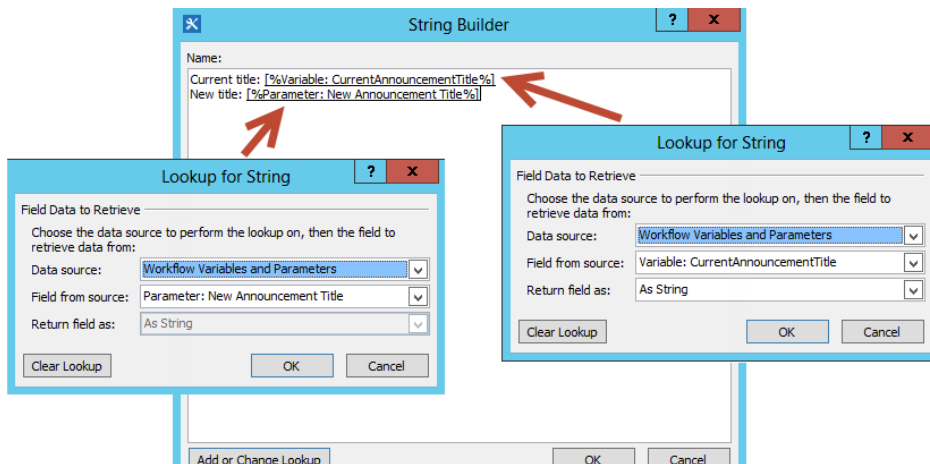
- For the **Variable** property, click the drop down and select **Create a new variable**.
  - Name:** CurrentAnnouncementTitle
  - Type:** String
  - Click **OK**
- Click into the **Value** property and click the **fx** button and select the following options:



c) Click **OK** twice

30. Open the **Log variable values** action's **Properties** dialog using the same process as before.

- Click into the **Message** field and click the builder button.
- Enter the following message, using the **Add or Change Lookup** button to add the values shown in the following figure:
  - Type in the **Current title:** and **New title:** and then click the **Add or Change Lookup** button with the cursor positioned to the right of each entry to add the **Lookup for String** entries as shown below.



c) Click **OK** twice

31. Switch from the Visual Designer view to the **Text-Based Designer** by clicking the **Views** button in the ribbon in the **Manage** group.

32. Locate the **Transition to stage** section within **Stage Init**. Update the path the workflow should take depending on the values selected:

- In the first part of the **If** statement, select the following linked phrases and update their values:
  - value** (the first one): Parameter: New Announcement Title  
(Hint: use the **fx** button to add this and select **Workflow Variables and Parameters** as your Data source).
  - equals**: Change to **is not empty**

**Stage: Stage Init**

Log Entering Stage Init to the workflow history list

then Set Variable: CurrentAnnouncementTitle to Current Item:Title

then Log Current title: [%Variable: CurrentAnn... to the workflow history list

---

**Transition to stage**

If Parameter: New Announcement Title is not empty

Go to Stage Rename Item

Else

Go to Stage No Changes to Item

33. Update the **Stage Rename Item** to reflect what you see in the following figure using the techniques you learned in the previous steps:

**Stage: Stage Rename Item**

Log Entering Stage Rename Item to the workflow history list

then Set Title to Parameter: New Announcement Title

---

**Transition to stage**

Go to Stage End

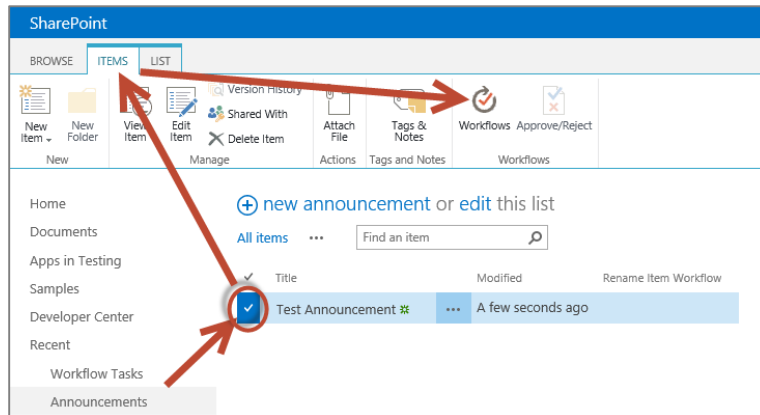
34. Update the **Stage: No Changes to Item**
- a) Set the **message** for the first **Log to history list** action to **Entering Stage No Title Changes Specified**.
  - b) Set the **message** for the second **Log to history list** action to **Nothing to change**.
35. Finally, in the **Log to history list** action within the **Stage End**, click the **message** link and enter **Workflow complete**.
36. Click the **Save** button in the ribbon group **Save** to save your workflow.

## Deploy and Test the Workflow

37. Publish the workflow to SharePoint 2013 and Workflow Manager:
- a) Click the **Publish** button in the ribbon group **Save**.
38. Create a new list item to be a test subject:
- a) Open **Internet Explorer** and browse to <http://workflow.wingtip.com>.
  - b) In the **Quick Launch** to the left of the page, select **Announcements**.
  - c) Create a new item by selecting the **new announcement** link or from the ribbon, click the **Items** tab and then the **New Item** button.
  - d) Set the **Title** of the new item to **Test Announcement** and click **Save**.
39. Start a new instance of your workflow:
- a) In the **Announcements** list, select the **Test Announcement** item (i.e. click on the **check mark area** to the left of the **Test Announcement** Item)



- b) Within the ribbon, select the **Items** tab and then click the **Workflows** button under the **Workflows** group.



- c) On the **Announcements: Workflows: Test Announcement** page, click **Rename Item Workflow** under the **Start a New Workflow** heading.
- d) A form will load prompting you to enter a new announcement title. Enter **A New Title** and click **Start**.

It will take a moment for the workflow to start and process all the way through... be patient while this happens. In the meantime continue on to the next step

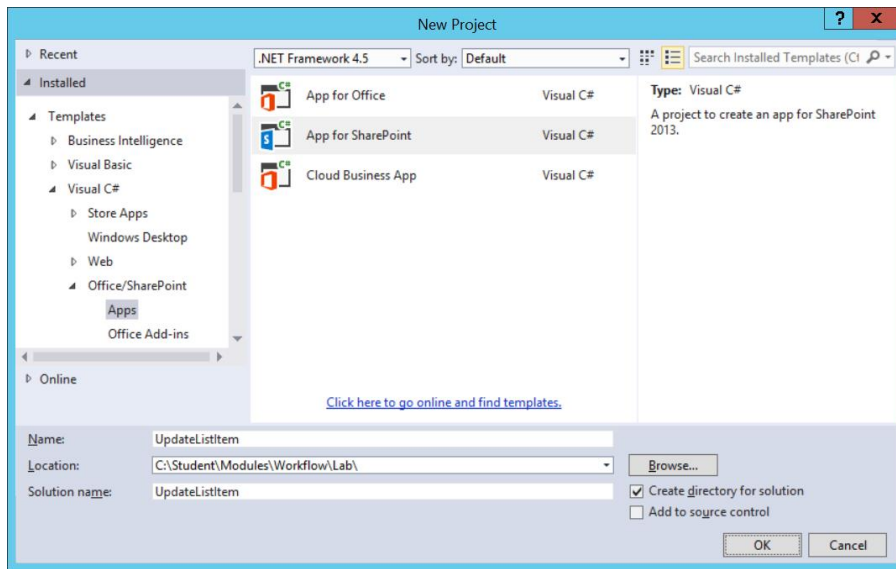
40. When SharePoint returns you back to the **Announcements** list, select the **Test Announcement** item.
41. Within the ribbon, select the **Items** tab and then click the **Workflows** button under the **Workflows** group.
42. Notice the status page will show both running and completed workflows. Click the status link for the workflow you just started to see the status page with the logging information.
43. If it hasn't completed by now, keep refreshing the page every few seconds to see if it has completed. Once it completes you should be able to see the **Internal Status** field change to **Completed**, the title updated, and the logging information for the workflow executed.

In this exercise you created a new workflow first with Visio 2013 and then completed it using SharePoint Designer 2013. You got some hands on experience with the new stages support and design tools in SharePoint 2013 and Workflow Manager.

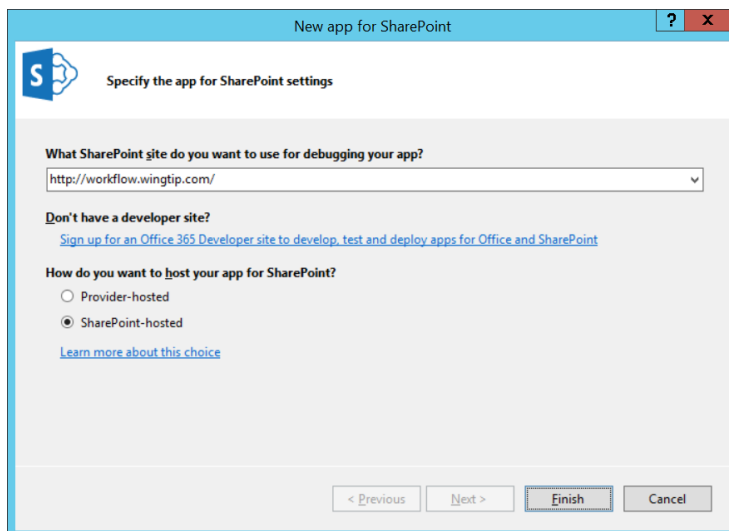
### Exercise 3: Developing a Sequential Workflow with Visual Studio 2013

In this exercise you will use Visual Studio 2013 to create a custom sequential workflow for a SharePoint 2013 site that will demonstrate calling web services from the workflow. More specifically, the custom workflow you develop will call a custom OData web service that will return data about users. If the workflow finds a matching user in the remote service it takes the user's information and updates the SharePoint list item.

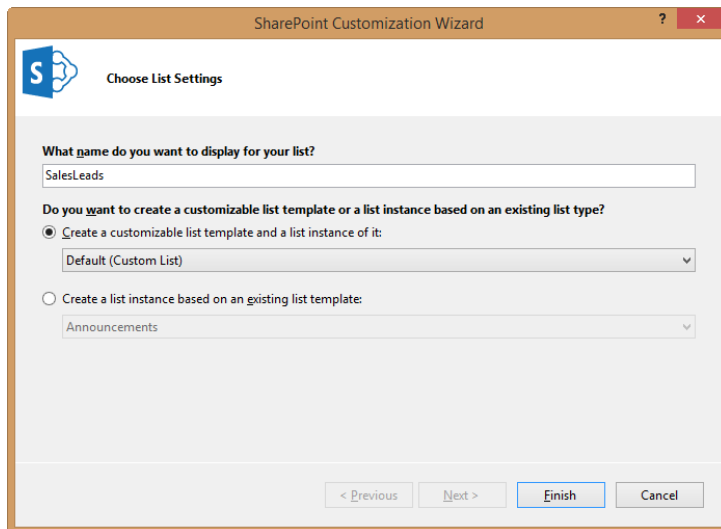
1. Create a new project in Visual Studio 2013:
  - a) Launch **Visual Studio 2013** as administrator:
2. Create a new SharePoint Hosted App project:
  - a) In Visual Studio select **File > New > Project**.
  - b) Select the **App for SharePoint** from the **Visual C# > Office-SharePoint > Apps** template category.
  - c) Set the name of the project to **UpdateListItem** and click **OK**.



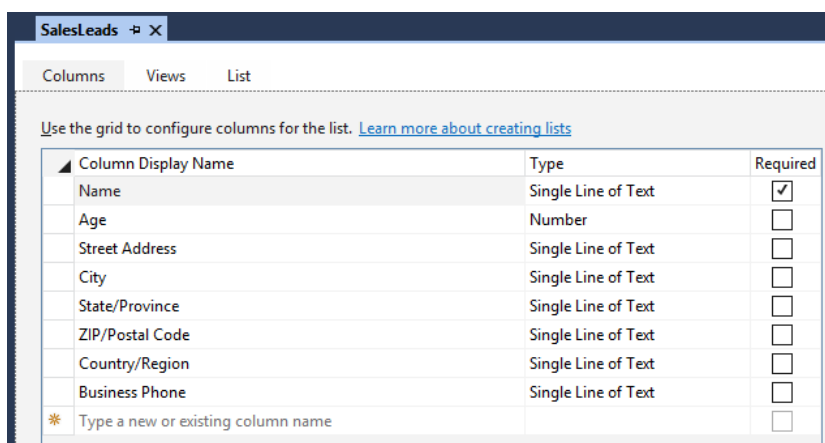
- d) In the next page of the wizard, enter the URL of <http://workflow.wingtip.com> to use for testing.
- e) Set the type of app as a **SharePoint-hosted app**.
- f) Click **Finish**.



- 3. Before creating the workflow, you need a list to attach it to. Therefore, create a list that will be used for testing the workflow:
  - a) Right-click the project **UpdateListItem** in the Solution Explorer tool widow and select **Add / New Item**.
  - b) In the **Add New Item** dialog, select **List**.
  - c) Give the list a name of **SalesLeads** and click **Add**.
  - d) On the SharePoint Customization Wizard page, set the name of the list to **Sales Leads**, select a customizable list template and pick **Default (Custom List)**.



- e) Click **Finish**.
4. After Visual Studio creates the list, it will open the list designer. Make the following changes & additions to the list schema as shown in the following screen shot.
  - a) Change **Title** to **Name**.
  - b) Add **Age** (type = **Number**).
  - c) Add **Street Address** (type = **Single Line of Text**).
  - d) Add **City** (type = **Single Line of Text**).
  - e) Add **State/Province** (type = **Single Line of Text**).
  - f) Add **Zip/Postal Code** (type = **Single Line of Text**).
  - g) Add **Country/Region** (type = **Single Line of Text**).
  - h) Add **Business Phone** (type = **Single Line of Text**).



5. Because this app will test a workflow using a SharePoint list, simplify the startup experience of the app by changing the start page.
  - a) Open the **AppManifest.xml** file.
  - b) Change the Start Page to **~appWebUrl/Lists/SalesLeads** to load the page with the lists default view when starting the app.
6. Add a new workflow to the project:
  - a) Right-click the project **UpdateListItem** in the Solution Explorer tool widow and select **Add > New Item**.
  - b) Select **Workflow** from the **Office/SharePoint** template category and set the name to **UpdateContactFromService**.

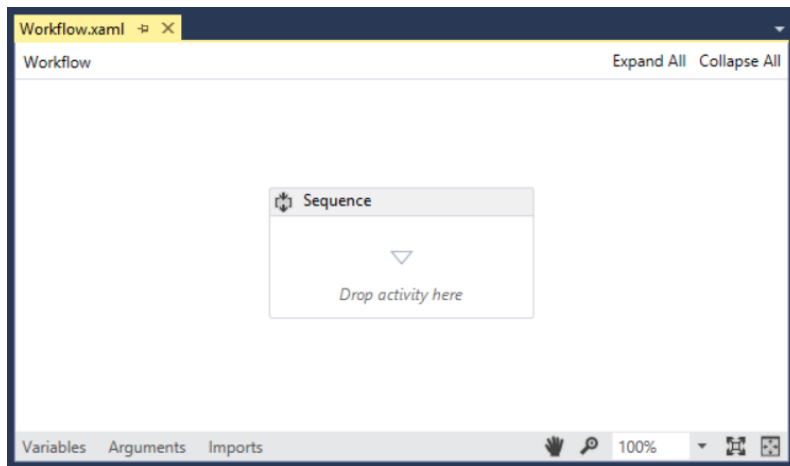
- c) Click **Add**.
- d) In the SharePoint Customization Wizard, set the name to **Update Contact From Service** and set it to a **List Workflow**, then click **Next**.

The screenshot shows the 'SharePoint Customization Wizard' window. The title bar says 'SharePoint Customization Wizard'. The main heading is 'Specify the workflow name for debugging'. Below this, there is a section 'What is the name of the workflow?' with a text box containing 'Update Contact From Service'. Below that is a section 'What type of workflow do you want to create?' with two radio buttons: 'List Workflow' (selected) and 'Site Workflow'. At the bottom, there are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

7. On the next page in the wizard, do the following:
- a) Check the box for **Would you like Visual Studio to automatically associate the workflow**.
  - b) Set **The library or list to associate your workflow with** to **SalesLeads**.
  - c) Set **The history list to display content logged by WriteToHistory activity** to **<Create New>**.
  - d) Set **The task list where the workflow will create tasks for workflow participants** to **<Create New>**.
  - e) Click **Finish**.

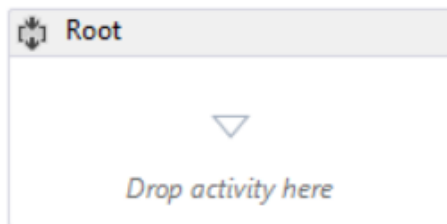
The screenshot shows the 'SharePoint Customization Wizard' window, Step 2. The title bar says 'SharePoint Customization Wizard'. The main heading is 'Select the lists you will use when debugging'. Below this, there is a section 'Would you like Visual Studio to automatically associate the workflow?' with a checked checkbox and the text 'Yes, associate this workflow with the following libraries and lists. If this step is omitted, you must manually associate the workflow after it is created.' Below this are three dropdown menus: 'The library or list to associate your workflow with:' (selected 'SalesLeads'), 'The history list to display content logged by WriteToHistory activity:' (selected '<Create New>'), and 'The task list where the workflow will create tasks for workflow participants:' (selected '<Create New>'). At the bottom, there are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

8. At this point you have an empty workflow in the design surface in Visual Studio.



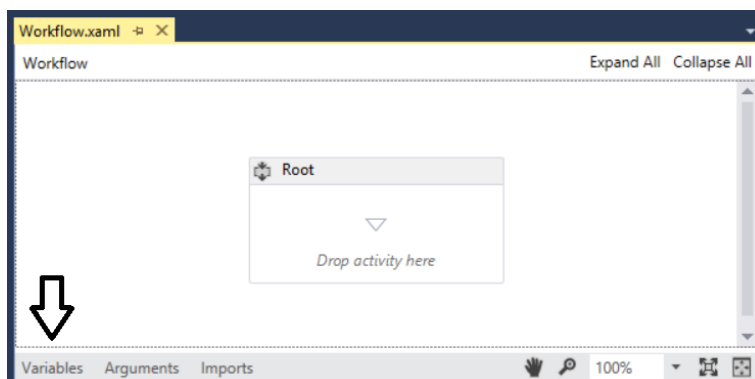
The first step in designing this workflow will be to do some high-level customizations to change the overall sequence and to add some global variables to the workflow.

9. Rename the default activity:
- Select the single Sequence activity.
  - Using the Properties tool window, change the **DisplayName** to **Root**.



It's a good idea to rename activities to make it easier to develop workflows because many workflows contain multiple nested sequences. Just above the design surface where it says workflow is a breadcrumb-like navigation that will fill up the more nested you are in the workflow. If it said Sequence multiple times, it would provide zero value. Therefore starting with ROOT is clear that this is the top-most part of the workflow.

10. Now add some variables that will be used throughout the workflow. At the bottom of the designer, click the **Variables** tab to bring up a new pane to create variables.



11. Add the following variables with the specified data types and default values:

- a) **PersonName** (string)
- b) **WebServiceResponse** (DynamicValue)

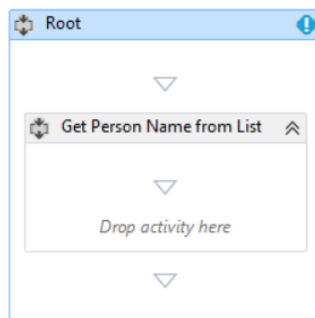
Note that the **DynamicValue** data type won't be visible in the selector automatically. Select **Browse for types** and type the data type **DynamicValue** in the search to find it.

- c) **WebServiceQuery** (string)
- d) **PersonID** (string), Default = "-1"
- e) **WebServiceUri** (string), Default = "http://services.odata.org/V3/OData/OData.svc"

Name	Variable type	Scope	Default
PersonName	String	ROOT	<i>Enter a C# expression</i>
WebServiceResponse	DynamicValue	ROOT	<i>Enter a C# expression</i>
WebServiceQuery	String	ROOT	<i>Enter a C# expression</i>
PersonID	String	ROOT	"-1"
WebServiceUri	String	ROOT	"http://services.odata.org/V3/OData/OData.svc"

12. Add the first sequence (step) to the workflow to retrieve the name of the person entered into the list item that you will search for in the remote web service:

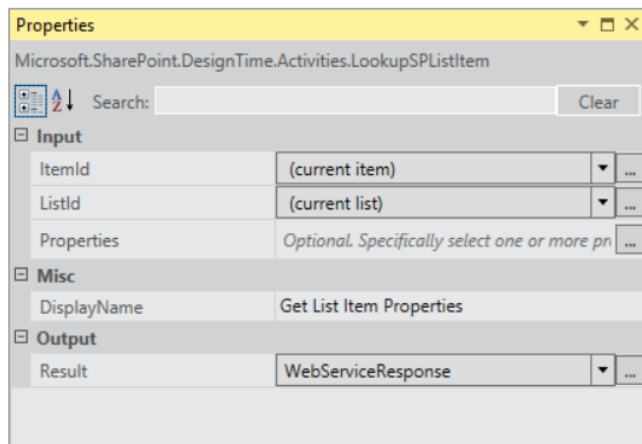
- a) Using the Toolbox, drag a **Sequence** activity to the design surface and place it into the existing **Root** sequence.
- b) Rename the new sequence to **Get Person Name from ListItem** using the technique you used to rename the default sequence.



13. Using the Toolbox, drag a **LookupSPLListItem** activity into the sequence you just created.

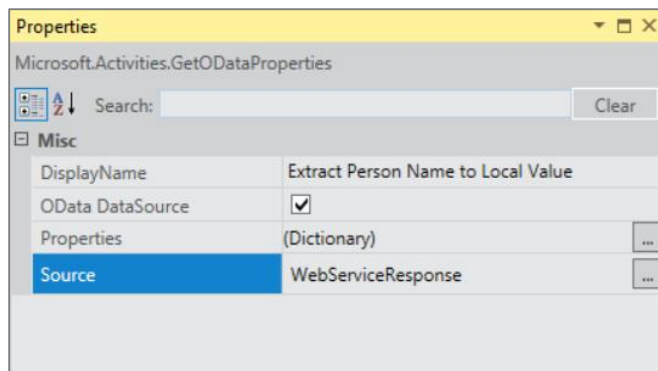
14. Select the **LookupSPLListItem** activity and using the Properties tool window, set the following values:

- a) **ItemId**: (current item)
- b) **ListId**: (current list)
- c) **DisplayName**: Get List Item Properties
- d) **Result**: WebServiceResponse



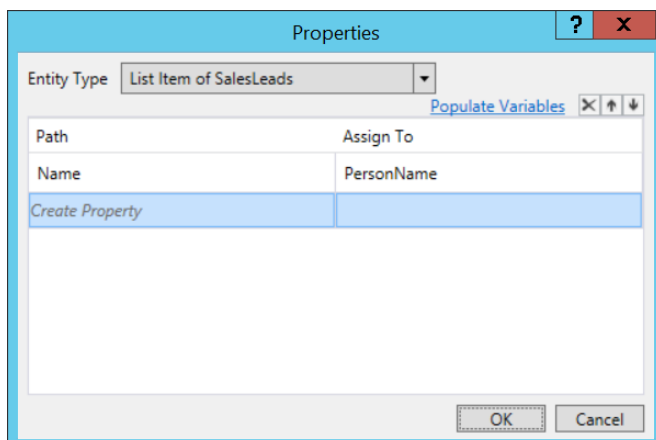
15. Drag a **GetDynamicValueProperties** activity just after the activity you just added in the last step. Using the Properties tool window, set the following values:

- DisplayName:** Extract Person Name to Local Value
- OData DataSource:** Checked
- Source:** WebServiceResponse

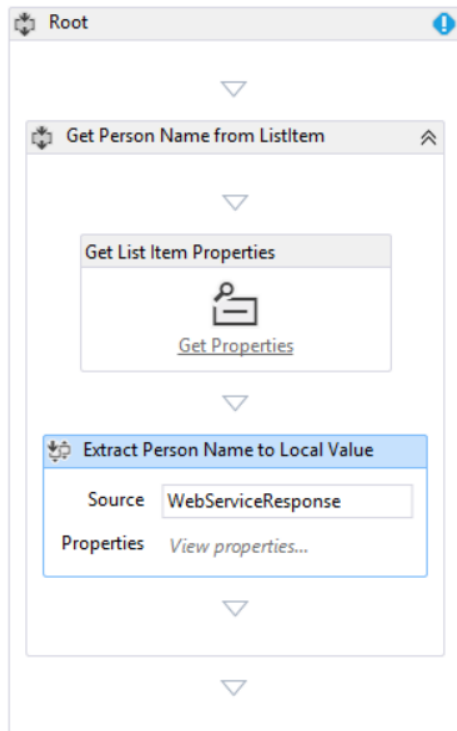


16. Click the ellipse builder button (...) for the **Properties** property.

- Set the **Entity Type** to **List Item of SalesLeads**.
- Select the property **Name** and assign it to the **PersonName** variable as shown in the following screenshot:

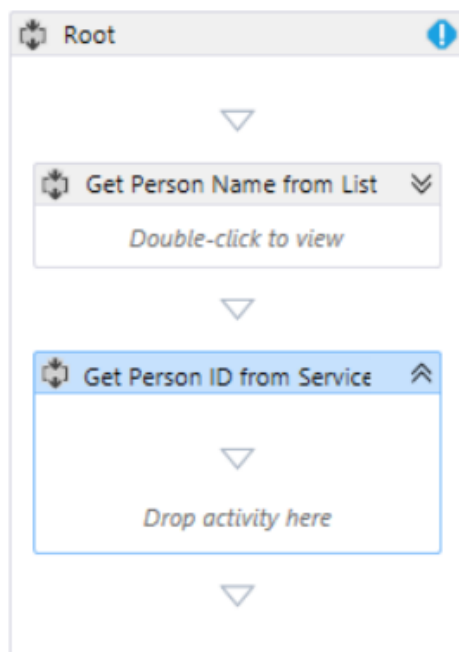


17. At this point your workflow should look like the following screenshot.



Now you will add a second sequence activity that will query the web service for a matching name entered in the list item.

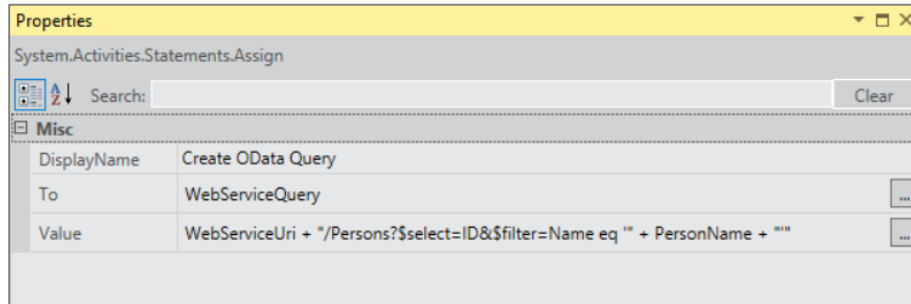
18. Add a new **Sequence** activity immediately after the existing **Get Person Name from List Item** activity.
- Rename the new activity to **Get Person ID from Service**.





19. Add a new **Assign** activity and set its properties to the following values:

- a) **DisplayName:** Create OData Query
- b) **To:** WebServiceQuery
- c) **Value:** WebServiceUri + "/Persons?\$select=ID&\$filter=Name eq '" + PersonName + "'"



Misc	
DisplayName	Create OData Query
To	WebServiceQuery
Value	WebServiceUri + "/Persons?\$select=ID&\$filter=Name eq '" + PersonName + "'"

20. To execute the web service call, add an **HttpSend** activity and set its properties to the following values.

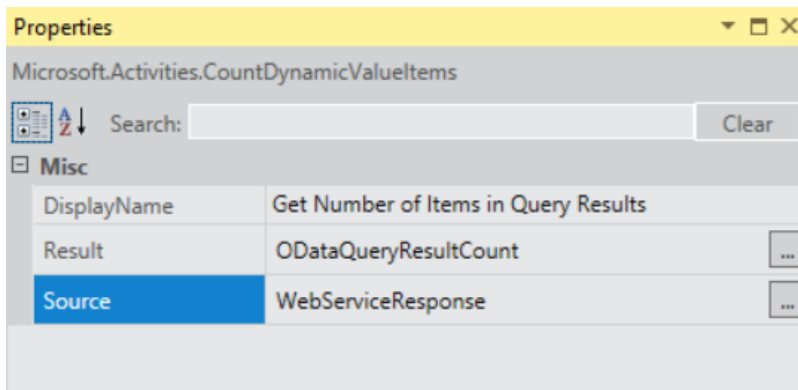
- a) **DisplayName:** Execute OData Query
- b) **Method:** GET
- c) **Uri:** WebServiceQuery
- d) **ResponseContent:** WebServiceResponse
- e) Click the ellipse button (...) on the **RequestHeaders** property.
  - i) Add a header **Accept** with the value of "**application/json;odata=verbose**".

21. Add a new variable that will hold the number of items returned by the web service query. Use the following values to create the variable using the same technique you used previously to create variables:

- a) **Name:** ODataQueryResultCount
- b) **Variable Type:** Int32
- c) **Scope:** Get Person ID from Service

22. Check if items are returned by the web service call by adding a **CountDynamicValueItems** activity with the these property values:

- a) **DisplayName:** Get Number of Items in Query Results
- b) **Result:** ODataQueryResultCount
- c) **Source:** WebServiceResponse



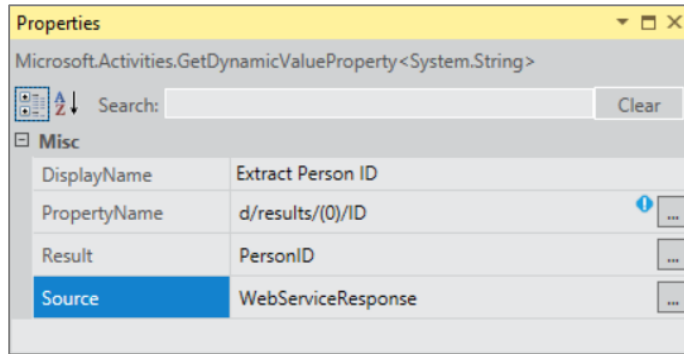
Misc	
DisplayName	Get Number of Items in Query Results
Result	ODataQueryResultCount
Source	WebServiceResponse

23. Get the ID of the person that was returned if results were returned:

- a) Add an **If** activity named **Check if Person Found in OData Query**.
- b) Set the **Condition** property of the **If** activity to **ODataQueryResultCount != 0**.

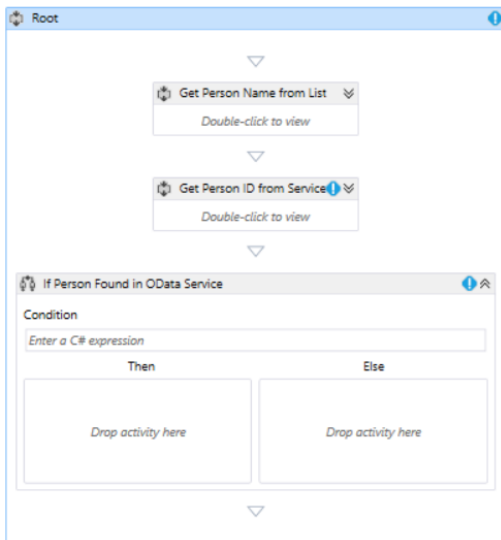
24. In the **Then** part of the **If** activity you just created, add a **GetDynamicValueProperty** activity. When you add this activity, Visual Studio will prompt you for the data type. Select **String**. Then set the following properties on the activity:

- i) **DisplayName:** Extract Person ID
- ii) **PropertyName:** d/results/(0)/ID
- iii) **Result:** PersonID
- iv) **Source:** WebServiceResponse



The last step is to add a new sequence activity that will update the item in the SharePoint list if a match was found in the web service.

25. Add an **If** activity named **If Person Found in OData Service** after the sequence activity you finished in the last step, but before the end of the ROOT sequence.

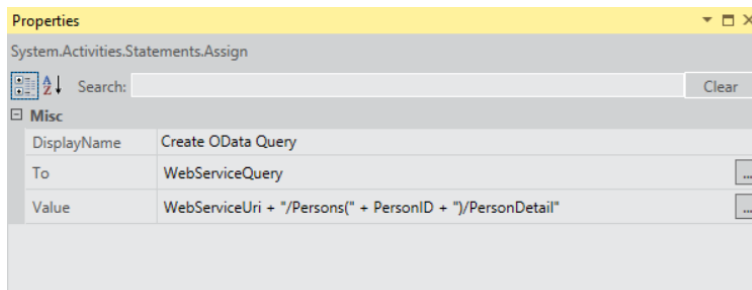


26. Inside the **If Person Found in OData Service** activity you just created, set the **Condition** property to **PersonID != "-1"**.

The **Else** side of the If activity represents the condition where no user was found in the remote service. Therefore there is nothing to do on the **Else** side.

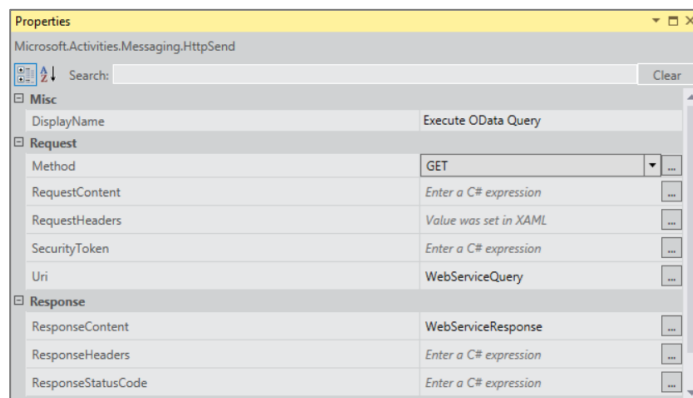
27. In the **Then** side of the If activity, add an **Assign** activity and set the following properties to the respective values:

- a) **DisplayName:** Create OData Query
- b) **To:** WebServiceQuery
- c) **Value:** WebServiceUri + "/Persons(" + PersonID + ")/PersonDetail"

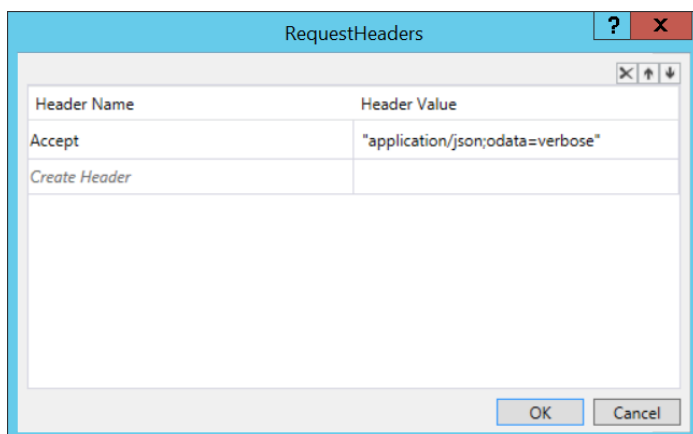


28. In the **Then** side of the **If** activity under the **Assign** activity, add a new **HttpSend** activity with the following property values:

- DisplayName:** Execute OData Query
- Method:** GET
- Uri:** WebServiceQuery
- ResponseContent:** WebServiceResponse

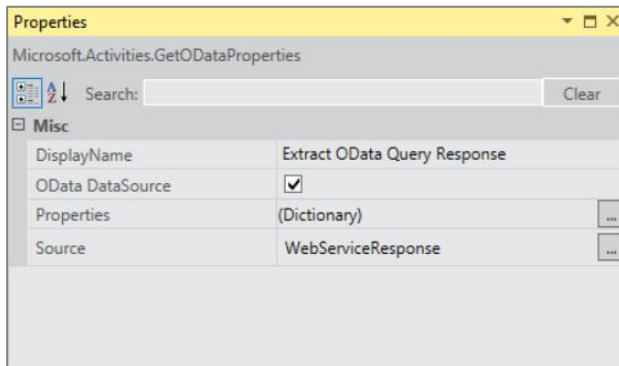


- RequestHeaders:** click the ... button and add a single header of **Accept** equal to "**application/json;odata=verbose**"



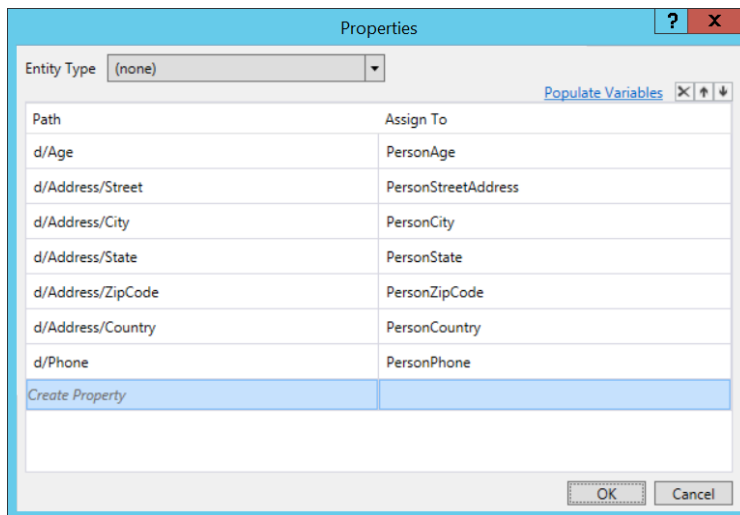
29. Add a **GetDynamicProperties** activity with the following values:

- DisplayName:** Extract OData Query Response
- OData DataSource:** Checked
- Source:** WebServiceResponse



30. In the **Extract OData Query Response** activity, click the ellipse button (...) for the **Properties** property.

- a) Leave the **Entity Type** set to **(none)** because Visual Studio does not have any context about the remove web service this workflow is calling. This means you need to manually parse the response by adding the following Paths to variables:
  - i) Path d/Age and set Assign To to PersonAge
  - ii) Path d/Address/Street and set Assign To to PersonStreetAddress
  - iii) Path d/Address/City and set Assign To to PersonCity
  - iv) Path d/Address/State and set Assign To to PersonState
  - v) Path d/Address/ZipCode and set Assign To to PersonZipCode
  - vi) Path d/Address/Country and set Assign To to PersonCountry
  - vii) Path d/Phone and set Assign To to PersonPhone



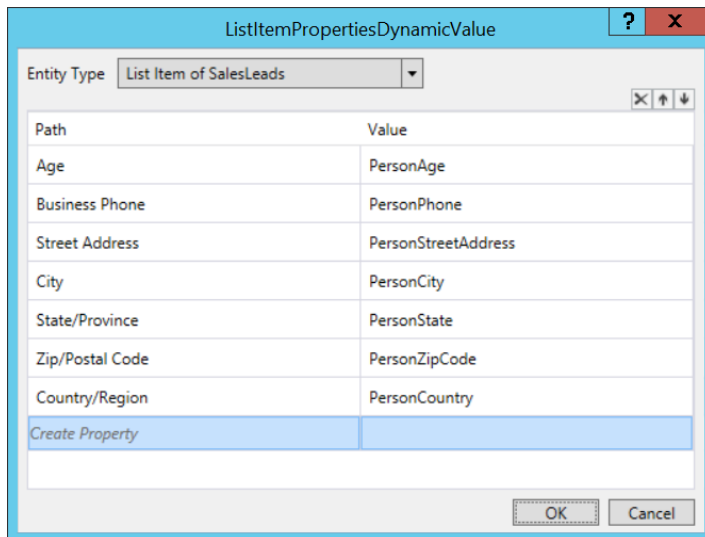
- b) Before clicking the **OK** button, click the link **PopulateVariables** to let Visual Studio create all the variables for you.
- c) Then click **OK**.

31. Add a **UpdateListItem** activity and set the following properties:

- a) **ItemId**: (current item)
- b) **ListId**: (current list)
- c) **DisplayName**: Update List Item

32. Click the ellipse button (...) on the **ListItemPropertiesDynamicValue** property.

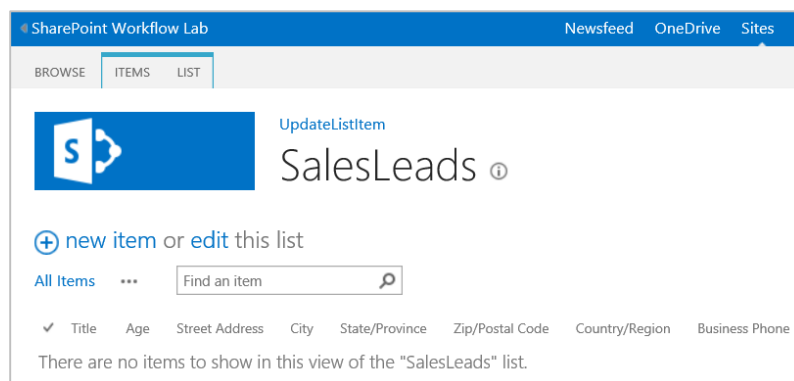
- a) Change the **Entity Type** to **List Item of SalesLeads**.
- b) For each variable you retrieved in the last activity, use the selectors to associate each field in the last with those values as shown from the following screenshot:



Path	Value
Age	PersonAge
Business Phone	PersonPhone
Street Address	PersonStreetAddress
City	PersonCity
State/Province	PersonState
Zip/Postal Code	PersonZipCode
Country/Region	PersonCountry
Create Property	

Now the workflow has been completed and it is time to test it out in the Visual Studio debugger.

33. With the workflow complete, in Visual Studio, select **Debug / Start Debugging** to launch the workflow.
  - a) Once the debugging session starts, you should be at the default view of the SalesLeads list.



SharePoint Workflow Lab Newsfeed OneDrive Sites

BROWSE ITEMS LIST

UpdateListItem

SalesLeads

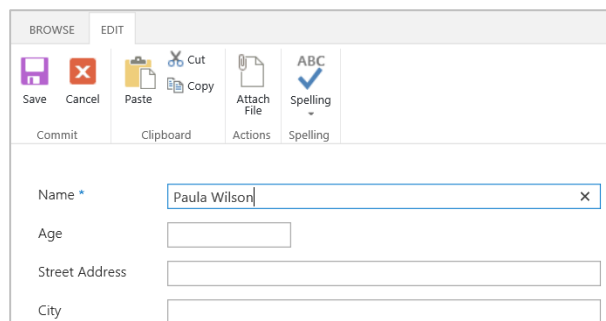
+ new item or edit this list

All Items Find an item

✓ Title Age Street Address City State/Province Zip/Postal Code Country/Region Business Phone

There are no items to show in this view of the "SalesLeads" list.

- b) Click the new item link to enter a new item to the list.
- c) Test the workflow by entering the name 'Paula Wilson' who is an existing user and click the **Save** button.



BROWSE EDIT

Save Cancel Paste Copy Attach File Spelling

Commit Clipboard Actions Spelling

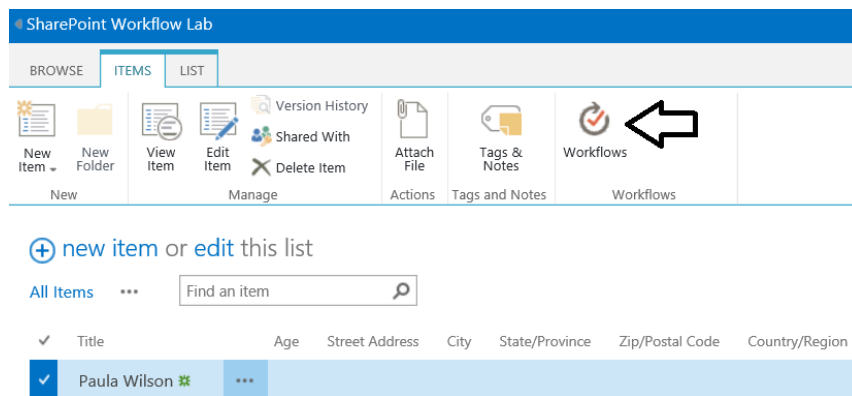
Name \* Paula Wilson

Age

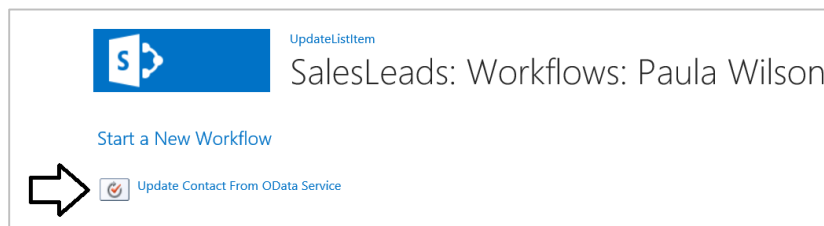
Street Address

City

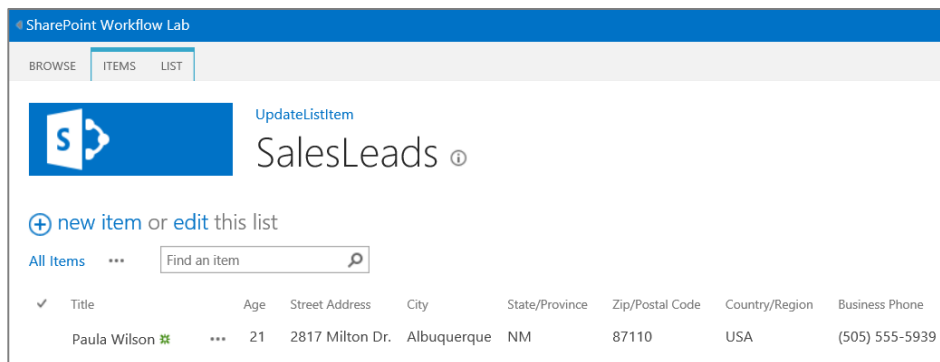
34. After creating the list item, select the item and click the **Workflows** button in the SharePoint ribbon.



35. On the Workflow page for the new SalesLead item, you will see a link to start the workflow you just created. Click this button to start your workflow.



36. Once the workflow starts, go back to the list item and keep refreshing the page... after a few moments you should see all of the user's details are updated as seen in the following figure:



Congratulations! In this exercise you created a sequential workflow that queried an OData / REST web service for a specific person & used that data to update a SharePoint list item.