# SharePoint 2013 Demo – Creating a Workflow with Visual Studio

## Estimated time to complete this demo

25 minutes

## Objectives

Demonstration of following capabilities in SharePoint 2013

* Create a workflow using Visual Studio 2012
* Leverage the new DynamicValue type

## Computers in this demo setup

This demo uses virtual machines as described in the following table. Before you begin the demo, you must start the virtual machines and then log on to the computers.

|  |  |
| --- | --- |
| **Virtual Machine** | **Role** |
| {Supplied by Instructor} | Domain Controller |
| {Supplied by Instructor} | Actual SharePoint environment with Office client and other required software. |

All user accounts in this lab use the password {Supplied by Instructor}.

## Getting started with demo

In the files provided with the hands on lab, run the batch file called **SetupModule.bat** by double clicking it. This file will execute a PowerShell script that will create a new site collection at [http://intranet.contoso.com/sites/Workflow](http://wave15-sp/sites/IntroSpApps%20) .

In addition also ensure that a user **Dan Jump** (**CORP\danj**) has been added to the **Site Collection Administrators** group of this site. You must create & run workflows under this user rather than the administrator account because workflows cannot be started using a SYSTEM account.

# Creating a Workflow with Visual Studio

* 1. In this exercise you will create a workflow using Visual Studio. This workflow will use the new HTTP and DynamicValue activities to issue a call from a sandbox solution deployed workflow that gets a description of a specified category from a publically available OData service.

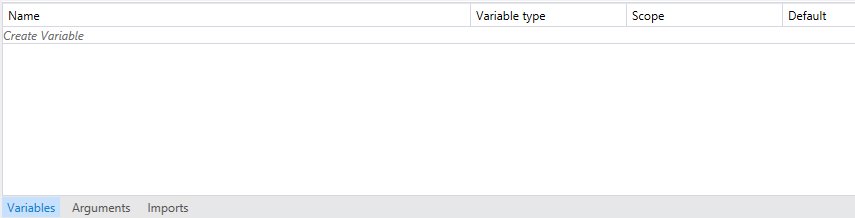
|  |
| --- |
| Description: C:\Users\vesaj\Pictures\DVD_ART36\Artwork_Imagery\Icons - Illustrations\_ SUPER VISTA STYLE\yield sign red white exclamation point.png **Important** |
| *This lab exercise involves using an HTTP activity in a workflow that will obtain content from an externally hosted Web service. Therefore the hands on lab computer running the workflow must have internet connectivity enabled and correctly configured in order to complete this lab.* |

## Task 1: Create a Workflow Project Variables

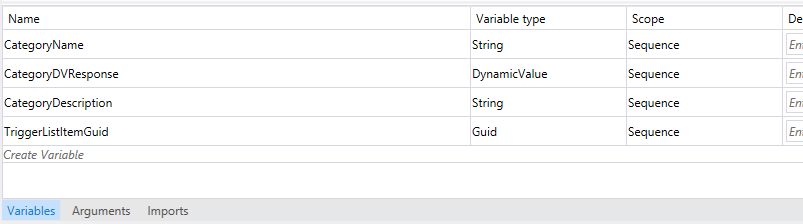
In this task you will create a workflow in Visual Studio and add some variables that will be used throughout the workflow.

* Begin this task logged on to **SP as CORP\danj**.

1. Open Visual Studio 11 as an administrator: **Start ⮚ All Programs ⮚ Microsoft Visual Studio 2012 ⮚** right-click **Visual Studio 2012** and select **Run as Administrator**.
2. Select **File ⮚ New ⮚ Project**.
3. In the **New Project** dialog, select the **Visual C# ⮚ Office/SharePoint ⮚ SharePoint Solutions** category. Pick the project template **SharePoint 2013 Project**. Give the project the name **CategoryDescriptionLookupWorkflow**.
4. In the **SharePoint Customization Wizard** dialog, select **Deploy as a sandboxed solution** and enter the URL of the workflow site (<http://intranet.contoso.com/sites/Workflow>) as the site to use in debugging, finally clicking **OK**.
5. Once the project is created, add a new workflow item to the project. Right-click the project in the **Solution Explorer** tool window and select **Add ⮚ New Item**:
6. In the Add New Item dialog, select **Visual C# Items ⮚ Office/SharePoint** category and the template **Workflow**. Give it a name of **CategoryDescriptionLookupWorkflow** and click **Add**.
7. In the SharePoint Customization Wizard, select **List Workflow** and change the name to **Category Description Lookup Workflow**.
8. On the next step of the **SharePoint Customization Wizard**, check the box for **Would you like Visual Studio to automatically associate the workflow?** as well as the following values and click Finish.
   * The library or list to associate… : **Announcements**
   * The history list to display…: **Workflow History** or **<Create New>** if Workflow History isn’t an option.
   * The task list to display…: **Workflow Tasks** or **<Create New>** if Workflow Tasks isn’t an option.
9. Visual Studio will automatically create and open in the designer the Workflow.xaml file that you will use to design the workflow. At the bottom of the designer you will find three tabs: Variables, Arguments & Imports. Click **Variables** to open the **Variables** window:



1. Once the **Variables** window it opens, click the **Sequence** activity in the designer to set the scope of the variables to the entire workflow.
2. Create three new variables that will be used throughout the workflow using the following options:
   * Variable:
     1. **Name:** CategoryName
     2. **Variable Type:** String
     3. **Scope:** Sequence
   * Variable:
     1. **Name:** CategoryDVResponse
     2. **Variable Type:** Microsoft.Activities.DynamicValue
     3. **Scope:** Sequence
   * Variable:
     1. **Name:** CategoryDescription
     2. **Variable Type:** String
     3. **Scope:** Sequence
   * Variable:
     1. **Name:** TriggerListItemGuid
     2. **Variable Type:** System.Guid
     3. **Scope:** Sequence

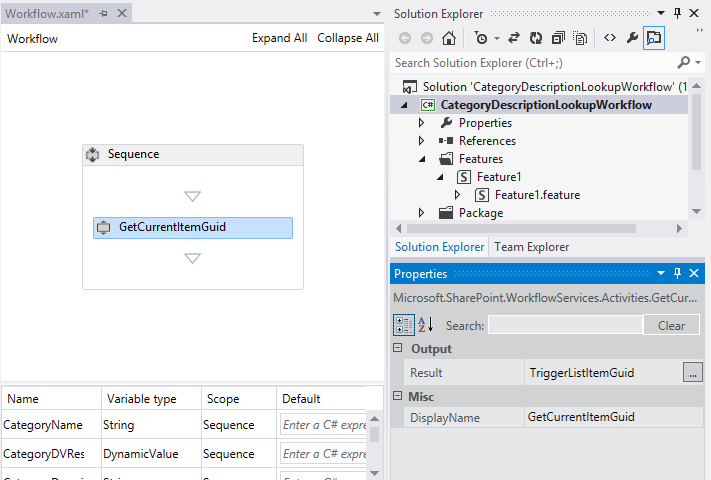


1. Save your changes by clicking **File ⮚ Save All**.

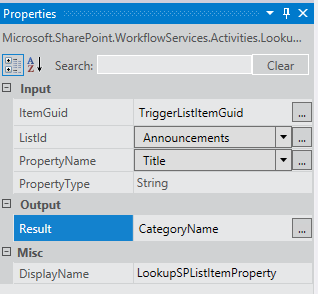
## Task 2: Add and Configure Activities to Request Data from an OData Service

* Begin this task logged on to **SP as CORP\danj**.

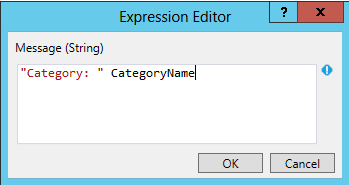
1. While still in the workflow created in the previous task, add a new **GetCurrentItemGuid** activity (found in the **Toolbox** under the **Utility Actions** category) to the workflow within the **Sequence** activity in the designer.
2. Select the **GetCurrentItemGuid** activity in the workflow, press **[F4]** to open the **Property** tool window and set the Result property to **TiggerListItemGuid**.



1. Next, add a new **LookupSPListItemProperty** activity (found in the **Toolbox** under the **SP – List Items** category) to the workflow within the **Sequence** activity in the designer.
2. Select the **LookupSPListItemProperty** activity and press **[F4]** to open the **Property** tool window. Set the following properties on the activity to extract the value in the **Title** field of the list item that triggered the workflow and store it in a local variable created in the previous task:
   * **ItemGuid:** TriggerListItemGuid
   * **ListId:** Announcements
   * **PropertyName:** Title
   * **Result:** CategoryName



1. Add another activity (**WriteToHistory** found in the **Utility Actions** category in the **Toolbox**) to the workflow immediately following the previous activity added to write the value extracted from the item to the history list.
2. Select the **WriteToHistory** activity and press **[F4]**. Use the builder to create a string that writes out the name of the title field:

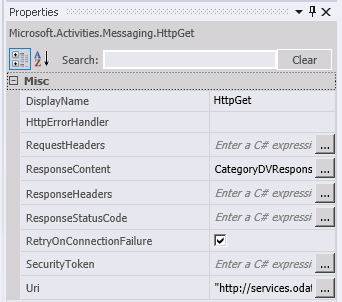


1. Save your changes by clicking **File ⮚ Save All**.

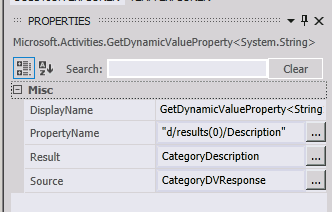
## Task 3: Call & Process the OData Service Response

* Begin this task logged on to **SP as CORP\danj**.

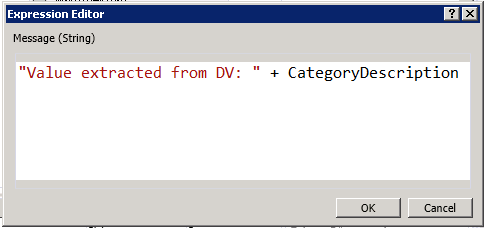
1. Now it is time to add an activity to the workflow that will make a call to the sample Northwind OData service hosted on <http://www.odata.org>. Add an **HttpSend** activity found in the **Messaging** category of the Toolbox.
2. Select the **HttpSend** activity and press **[F4]**. This activity has a few properties that you can ignore… many are used to authenticate against the service but in this exercise the service you will call is anonymous.
3. Store the results of the OData Web service call, which in the JSON format, to the **CategoryDVResult** DynamicValue variable created in a previous task by entering the variable name in the **ResponseContent** property.
4. Set the **Uri** property of the **HttpSend** activity to the following Web service address. You will have to craft the address by concatenating some strings together:
   * **Uri:** "http://services.odata.org/Northwind/Northwind.svc/Categories?$format=json&$filter=substringof('" + CategoryName + "', CategoryName) eq true&$select=Description"



1. Finally, for some additional debugging add another **WriteToHistory** activity immediately following the **HttpGet** activity and set its **Message** property to the following in order to see the raw results returned by the service in the workflow status page:
   * **Message:** "Response from service: " & CategoryDVResponse.ToString()
2. With the service request stored in a variable, the next step is to extract it to another variable for use later in the workflow. From the **DynamicValue** category in the **Toolbox**, add a **GetDynamicValueProperty<T>** activity to the end of the workflow. When prompted for the type, select **String**.
3. Modify the properties of the **GetDynamicValueProperty<String>** activity to extract the **Description** of the category requested from the **CategoryDVResponse** (the Source) variable and store it in the **CategoryDescription** (Result) variable.



1. Add another activity (**WriteToHistory** found in the **SharePoint 15 Workflow** category in the **Toolbox**) to the workflow immediately following the previous activity added to write the value extracted from the item to the history list.



1. Save your changes by clicking **File ⮚ Save All**.

## Task 4: Update SharePoint List Item

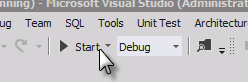
* Begin this task logged on to **SP as CORP\danj**.

1. The last step in creating this workflow is to update the list item with the results from the OData service. From the **Utility Actions** category, add a **SetField** activity to the end of the workflow.
2. Set the properties of the **SetField** workflow to update the Body of the list item (an announcement) to the value extracted from the JSON OData response:
   * **FieldName:** Body
   * **FieldValue:** CategoryDescription
3. Save your changes by clicking **File ⮚ Save All**.

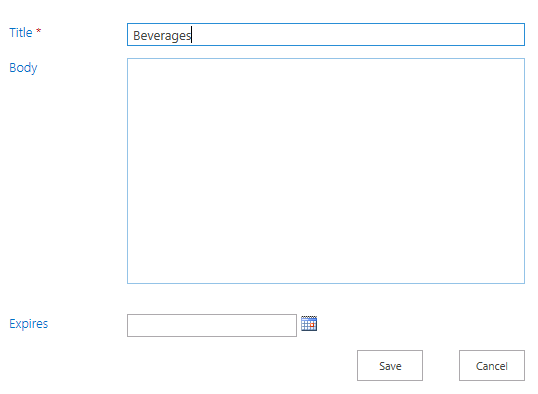
## Task 5: Deploy and Test the Workflow

* Begin this task logged on to **SP as CORP\danj**.

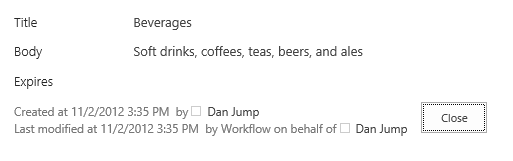
1. Start the workflow in debugging mode by clicking the **Start** button in Visual Studio.



1. If prompted by Visual Studio to attach to a process, click the **Attach** button.
2. Once the workflow has been deployed, it will launch a browser and navigate to the homepage of the test site.
3. Select the **Announcements** list in the left-hand Quick Launch menu.
4. Add an item to the **Announcements** list:
   * **Title**: Beverage
   * **Body:** <blank>



1. After adding the item, start the workflow. Do this by selecting the item and clicking the **Workflows** button under the **View** tab in the ribbon.
2. On the **Announcements: Workflows: [item title]** page, click the **CategoryDescriptionLookupWorkflow - WorkflowStart** to start the workflow.
3. After a brief delay the browser will redirect you to the Announcements list page again. The workflow will start but this may not be seen for a few seconds so be patient.
4. Click on the item in the list view to view the details page. You’ll notice that the **Body** field has been updated by the workflow.



* This is the end of the demo