

Creating and Debugging a SharePoint Workflow Solution in Visual Studio 2010

(Microsoft Corporation)

This walkthrough demonstrates how to create a basic sequential workflow template. The workflow checks a property of a shared document library to determine whether a document has been reviewed. If the document has been reviewed, the workflow finishes.

This walkthrough illustrates the following tasks:

- Creating a SharePoint list definition sequential workflow project in Visual Studio.
- Creating workflow activities.
- Handling workflow activity events.

Note: Although this walkthrough uses a sequential workflow project, the process is identical for a state machine workflow project.

Also, your computer might show different names or locations for some of the Visual Studio user interface elements in the following instructions. The Visual Studio edition that you have and the settings that you use determine these elements. For more information, see *Working with Settings*.

Prerequisites

You need the following components to complete this walkthrough:

- Supported editions of Microsoft Windows and SharePoint. For more information, see *Requirements for Developing SharePoint Solutions*.
- Visual Studio 2010.

Adding Properties to the SharePoint Shared Documents Library

To track the review status of documents in the **Shared Documents** library, we will create three new properties for shared documents on our SharePoint site: Status, Assignee, and Review Comments. We define these properties in the **Shared Documents** library.

To add properties to the SharePoint shared documents library

1. Open a SharePoint site, such as `http://<system name>/SitePages/Home.aspx`, in a Web browser.
2. On the QuickLaunch bar, click **Shared Documents**.
3. Click **Library** on the **Library Tools** ribbon and then click the **Create Column** button on the ribbon to create a new column.
4. Name the column Document Status, set its type to **Choice (menu to choose from)**, specify the following three choices, and then click **OK**:
 - Review Needed
 - Review Complete
 - Changes Requested
5. Create two more columns and name them Assignee and Review Comments. Set the Assignee column type as a single line of text, and the Review Comments column type as multiple lines of text.

Creating and Debugging a SharePoint Workflow Solution in Visual Studio 2010

(Microsoft Corporation)

Enabling Documents to be Edited without Requiring a Check Out

It is easier to test the workflow template when you can edit the documents without having to check them out. In the next procedure, you configure the SharePoint site to enable that.

To enable documents to be edited without checking them out

1. On the QuickLaunch bar, click **Shared Documents**.
2. Click **Library** on the **Library Tools** ribbon and then click the **Library Settings** button to display the **Document Library Settings** page.
3. In the **General Settings** section, click **Versioning Settings** to display the **Versioning Settings** page.
4. Verify that the setting for **Require documents to be checked out before they can be edited** is **No**. If it is not, change it to **No** and then click **OK**.
5. Close the browser.

Creating a SharePoint Sequential Workflow Project

A sequential workflow is a set of steps that executes in order until the last activity finishes. In this procedure, we create a sequential workflow that will apply to our Shared Documents list. The workflow wizard lets you associate the workflow with either the site definition or the list definition and lets you determine when the workflow will start.

To create a SharePoint sequential workflow project

1. Start Visual Studio and display the **New Project** dialog box by pointing to **New** on the **File** menu and then clicking **Project**.
2. Expand the **SharePoint** node under either **Visual C#** or **Visual Basic**, and then click **2010**.
3. In the **Templates** pane, select **Sequential Workflow**.
4. In the **Name** box, type MySharePointWorkflow and then click **OK**. The **SharePoint Customization Wizard** appears.
5. In the **Specify the site and security level for debugging** page, click **Next** to accept the default site and trust level. This step sets the trust level for the solution as farm solution, the only available option for workflow projects. For more information, see *Sandboxed Solution Considerations*.
6. In the **Specify the workflow name for debugging** page, accept the default name (**MySharePointWorkflow - Workflow1**). Keep the default workflow template type value, **List Workflow**, and then click **Next**.
7. In the **Would you like Visual Studio to automatically associate the workflow in a debug session?** page, click **Next** to accept all of the default settings. This step automatically associates the workflow with the Shared Documents library.
8. In the **Specify the conditions for how your workflow is started** page, leave the default options selected in the **How do you want the workflow to start?** section and click **Finish**. This page enables you to specify when your workflow starts. By default, the workflow starts either when a user manually starts it in SharePoint or when an item to which the workflow is associated is created.

Creating and Debugging a SharePoint Workflow Solution in Visual Studio 2010

(Microsoft Corporation)

Creating Workflow Activities

Workflows contain one or more *activities* that represent actions to perform. Use the workflow designer to arrange activities for a workflow. In this procedure, we will add two activities to the workflow: `HandleExternalEventActivity` and `OnWorkflowItemChanged`. These activities monitor the review status of documents in the **Shared Documents** list

To create workflow activities

1. The workflow should be displayed in the workflow designer. If it is not, then double-click either **Workflow1.cs** or **Workflow1.vb** in **Solution Explorer** to open it.
2. In the designer, click the **OnWorkflowActivated1** activity to select it.
3. In the **Properties** window, type `onWorkflowActivated` next to the **Invoked** property, and then press ENTER. The Code Editor opens, and an event handler method named `onWorkflowActivated` is added to the `Workflow1` code file.
4. Switch back to the workflow designer, open the toolbox, and then expand the **Windows Workflow v3.0** node.
5. Drag a **While** activity from the **Windows Workflow v3.0** node of the **Toolbox** and connect it to the line under the **onWorkflowActivated1** activity.
6. Click the **WhileActivity1** activity to select it.
7. In the **Properties** window, set **Condition** to Code Condition.
8. Expand the **Condition** property, and type `isWorkflowPending` next to the child **Condition** property, and then press ENTER. The Code Editor opens, and a method named `isWorkflowPending` is added to the `Workflow1` code file.
9. Switch back to the workflow designer, open the toolbox, and then expand the **SharePoint Workflow** node.
10. Drag a **OnWorkflowItemChanged** activity from the **SharePoint Workflow** node of the **Toolbox** and then connect it to the line inside the **whileActivity1** activity.
11. Click the **onWorkflowItemChanged1** activity to select it.
12. In the **Properties** window, set the properties as shown in the following table.

Property	Value
CorrelationToken	workflowToken
Invoked	onWorkflowItemChanged

Creating and Debugging a SharePoint Workflow Solution in Visual Studio 2010

(Microsoft Corporation)

Handling Activity Events

Finally, check the status of the document from each activity. If the document has been reviewed, then the workflow is finished.

To handle activity events

1. In Workflow1.cs or Workflow1.vb, add the following field to the top of the Workflow1 class. This field is used in an activity to determine whether the workflow is finished.

```
Boolean workflowPending = true;
```

2. Add the following method to the Workflow1 class. This method checks the value of the Document Status property of the Documents list to determine whether the document has been reviewed. If the Document Status property is set to Review Complete, then the checkStatus method sets the workflowPending field to false to indicate that the workflow is ready to finish.

```
private void checkStatus()  
{  
    if ((string)workflowProperties.Item["Document Status"] == "Review Complete")  
        workflowPending = false;  
}
```

3. Add the following code to the onWorkflowActivated and onWorkflowItemChanged methods to call the checkStatus method. When the workflow starts, the onWorkflowActivated method calls the checkStatus method to determine whether the document has already been reviewed. If it has not been reviewed, the workflow continues. When the document is saved, the onWorkflowItemChanged method calls the checkStatus method again to determine whether the document has been reviewed. While the workflowPending field is set to true, the workflow continues to run.

```
private void onWorkflowActivated(object sender, ExternalDataEventArgs e)  
{  
    // Check the status.  
    checkStatus();  
}
```

```
private void onWorkflowItemChanged(object sender, ExternalDataEventArgs e)  
{  
    // Check the status.  
    checkStatus();  
}
```

4. Add the following code to the isWorkflowPending method to check the status of the workflowPending property. Each time the document is saved, the **whileActivity1** activity calls the isWorkflowPending method. This method examines the Result property of the ConditionalEventArgs object to determine whether the **WhileActivity1** activity should continue or finish. If the property is set to true, the activity continues. Otherwise, the activity finishes and the workflow finishes.

```
private void isWorkflowPending(object sender, ConditionalEventArgs e)  
{  
    e.Result = workflowPending;  
}
```

Creating and Debugging a SharePoint Workflow Solution in Visual Studio 2010

(Microsoft Corporation)

}

5. Save the project.

Testing the SharePoint Workflow Template

When you start the debugger, Visual Studio deploys the workflow template to the SharePoint server and associates the workflow with the **Shared Documents** list. To test the workflow, start an instance of the workflow from a document in the **Shared Documents** list.

To test the SharePoint workflow template

1. In Workflow1.cs or Workflow1.vb, set a breakpoint next to the **onWorkflowActivated** method.
2. Press F5 to build and run the solution. The SharePoint site appears.
3. In the navigation pane in SharePoint, click **Shared Documents**.
4. In the **Shared Documents** page, click **Documents** on the **Library Tools** tab, and then click the **Upload Document** button to upload a document.
5. In the **Upload Document** dialog, click the **Browse** button, select any document file, click **Open**, and then click **OK**. This uploads the selected document into the **Shared Documents** list and starts the workflow.
6. In Visual Studio, verify that the debugger stops at the breakpoint next to the **onWorkflowActivated** method. Press F5 to continue execution.
7. You can change the settings for the document here, but leave them at the default values for now by clicking **Save**. This returns you to the **Shared Documents** page of the default SharePoint Web site.
8. In the **Shared Documents** page, verify that the value underneath the **MySharePointWorkflow** column is set to **In Progress**. This indicates that the workflow is in progress and that the document is awaiting review.
9. In the **Shared Documents** page, point to the document, click the down arrow, and then click **Edit Properties**.
10. Set **Document Status** to **Review Complete** and then click **Save**. This returns you to the **Shared Documents** page of the default SharePoint Web site.
11. In the **Shared Documents** page, verify that the value underneath the **MySharePointWorkflow** column is set to **Review Complete**. This indicates that workflow is finished and that the document has been reviewed.