**Microsoft Office SharePoint Server (MOSS) for Developers**

**(Part 1)**

**Topic 4: Workflows in SharePoint**

[**http://msdn.microsoft.com/rampup**](http://msdn.microsoft.com/rampup)

**Introduction**

Microsoft Windows SharePoint Services provides a robust, customizable work environment for users to create, collaborate, and store valuable business information. Now, with Microsoft Windows SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007, you can attach custom business processes to these documents or list items.

You can represent these custom business processes by using workflows. A workflow is a natural way to organize and run a set of work units, or activities, to form an executable representation of a work process. This process can control almost any aspect of an item in Windows SharePoint Services, including the life cycle of that item. The workflow is flexible enough to model both the system functions and the human actions necessary for the workflow to complete.

You can create workflows that are as simple or complex as your business processes require. You can create workflows that the user initiates, or workflows that Windows SharePoint Services automatically initiates based on some event, such as when an item is created or changed.

Suppose you need to create a simple workflow that routes a document to a series of users for approval or comments. This workflow would include actions that the system needs to perform, as well as provide interfaces for the users to interact with the workflow in prescribed ways. For example, Windows SharePoint Services would send an e-mail message to the selected users when the document was ready for review. Those users would then need to be able to notify Windows SharePoint Services when they had completed their reviews and, optionally, enter any comments. The workflow framework included in Windows SharePoint Services 3.0, and extended in SharePoint Server 2007, enables you to model such complex work processes and present them to end users in an easily understood, unobtrusive manner that guides them through each step of the process.

This article provides a high-level overview of workflows as they are implemented in Windows SharePoint Services and extended in SharePoint Server. It includes a discussion of the developer tools available to create workflows in both environments, and the respective capabilities and advantages of those tools.

**Workflow Persistence**

One of the most important services Windows SharePoint Services provides to the WF workflow engine is that of *persistence*. Workflows that include human interaction are inherently long running; even in ideal circumstances, humans require a relatively long time to complete work compared to machines. In many Microsoft Office scenarios, workflows typically require days, if not longer. Consider the example workflow that routes documents for approval. The approver might need several days to review the document.

Clearly, leaving each running workflow in memory for the entire duration of its execution is not feasible; very soon the resources required by accumulated long-running workflows would bring the system to a halt.

Instead, after a workflow instance has reached a point at which it is waiting for user input, Windows SharePoint Services unloads that workflow instance from memory and persists its data. Then, when an appropriate event, such as the user entering input, requires that the workflow instance start again, Windows SharePoint Services re-creates the workflow instance using the persisted data, so the workflow instance can receive and handle the event as necessary.

Although numerous workflow instances may run at any given time, only a fraction of those workflows are likely to be in memory and use system resources.

**Workflow Composition**

Earlier, we defined a workflow as a way to organize and run a set of work units, or activities, to form an executable representation of a work process. Therefore, each WF workflow consists of a set of related activities. An activity is the elemental unit of modeling, programmability, reuse, and execution within Windows Workflow Foundation. An activity may be performed by the system or by a user. For example, the system may send someone an e-mail message as an alert; or a person might approve a document for distribution.

Activities can also represent logical control structures that define scope and direct the execution flow of the workflow, much as code logic controls, such as **If Then** and **Do While** loops, control the program flow in code.

Activities can have properties, methods, and events. *Simple activities* perform a single unit of work, such as "delay for 1 day" or "invoke Web service." *Composite activities* contain other activities; for example, a conditional with two branches. You can also attach handlers, such as error or compensation handlers, to activities. This is especially advisable for composite activities.

In essence, the workflow itself is simply a composite activity that contains all the other activities in that workflow.

Therefore, workflow can be seen as a set of activities stored as a model that describes a real-world process. Work passes through the model from start to finish, and activities might be executed by people or by system functions. Workflow provides a way of describing the order of execution and dependent relationships between pieces of short-running or long-running work.

Windows Workflow Foundation includes a number of predefined activities that you can use in your workflows, or you can create your own custom activities. The Windows SharePoint Services workflow project template pack contains numerous activities designed explicitly for use in the Windows SharePoint Services environment. The template pack also contains the references necessary to program against the Windows SharePoint Services object model. Similarly, the SharePoint Server project template pack is customized for use in the SharePoint Server environment.

**Workflows in Windows SharePoint Services and SharePoint Server 2007**

So far, we have discussed the attributes of workflows for any application that implements the Windows Workflow Foundation. Now, let's take a closer look at workflows as they are implemented in Windows SharePoint Services specifically and discuss important aspects of that implementation.

**Workflow Templates and Instances**

In Windows SharePoint Services, workflows that are available on a site or list are referred to as workflow *templates*; a workflow that is actually running on a specific SharePoint item is called a workflow *instance*. So, on a given list, several instances of the same workflow template may be running at the same time, each on a different SharePoint item. More than one workflow can be running on a given SharePoint item at a time.

**Points of User Interaction with Workflows**

Let's look more closely at the various stages at which users can interact with workflows in Windows SharePoint Services and SharePoint Server.

**Association**

Association occurs when a site administrator makes the workflow template available on a specific document library, list, or content type. At this stage, the site administrator customizes the workflow for this specific document library, list, or content type by specifying the following parameter information:

* A unique name for the workflow
* How the workflow is applied to a given item: either automatically when the item is created or changed, or manually; and which roles, such as Administrator or Contributor, can start a workflow
* The Task list, in which the workflow can create tasks
* The History list in which the workflow can store history events, as defined by the workflow

In addition, the workflow developer can enable the site administration to further customize the workflow by setting parameter information specific to the particular workflow. The administrator may have to specify parameters, default values, and other information for the workflow as it applies to items on the list, library, or content type with which the administrator is associating it.

Associating a workflow actually occurs outside the workflow itself; no workflow instance is actually started during association. Rather, Windows SharePoint Services stores the association information in a special internal workflow association table. Then, when the workflow instance is actually started, Windows SharePoint Services uses the association data (and the initiation data, if any) to set the parameters of the new workflow instance.

**Initiation**

Whereas association deals with the workflow as it applies to a specific list, library, or content type, initiation deals with the workflow as it applies to a specific SharePoint item.

Initiation occurs when a user actually starts a workflow instance on a specific item. The user supplies information custom to that specific workflow on that item, if necessary, and then starts the workflow.

Workflow developers can use initiation to let users override or append the association parameters set by administrators, or specify additional parameters or information about the workflow as it applies to the given SharePoint item. Not all workflows need initiation. In fact, a workflow set to start automatically cannot have initiation parameters.

The Windows SharePoint Services workflow project for the Visual Studio 2008 Designer for Windows Workflow Foundation includes an activity that acts as the event handler when the workflow is initiated. This activity is the first activity in any Windows SharePoint Services workflow.

**Modification**

Modifications enable users to alter the parameters or flow of a workflow instance as it is running. As a developer, you may have points in your workflow at which you would like to enable users to alter the workflow while it is running on an item. For example, you might want to enable a user to assign a task to another person, or even add a specific task to the workflow. The options you present to users to change the workflow as it is running on an item are called *modifications*.

The Windows SharePoint Services workflow project for the Visual Studio 2008 Designer for Windows Workflow Foundation includes an activity to enable a workflow modification, as well as an activity that acts as an event handler when a workflow modification is enabled.

**Tasks**

Workflow activities that are performed by people are represented as tasks in Windows SharePoint Services workflows.

As workflow author, you can specify the task schema. For example, the task listing could include the following:

* Task title
* Name of person to whom the task is assigned
* Task status
* Task priority
* Task date due
* A link to the referenced item

The user can then edit the task as necessary. The workflow instance is notified of changes to workflow tasks and can choose to respond to them as specified in the workflow.

The Windows SharePoint Services workflow project for the Visual Studio 2008 Designer for Windows Workflow Foundation includes activities that create, delete, and update tasks, as well as activities that act as event handlers when tasks are created, updated, or deleted.

**Authoring SharePoint Workflows in Visual Studio 2008**

The Visual Studio 2008 Designer for Windows Workflow Foundation is an add-in for Visual Studio 2008. It provides a means to rapidly develop workflows by using a graphical interface that leverages a developer's knowledge of the Visual Studio development environment.

Visual Studio 2008 Designer for Windows Workflow Foundation is a tool for you to rapidly create the workflow in a way that is integrated with the development of the code encapsulating your business processes. To do this, the Visual Studio 2008 Designer for Windows Workflow Foundation provides a graphical interface with intuitive controls, hosted within the familiar Visual Studio development environment. Its features include the following:

* A drag-and-drop design surface that lets you assemble custom workflows from predefined workflow activities you drag from the **Toolbox**
* An interface that lets you work on your workflow markup using intuitive graphical tools
* Integration with the **Properties** window, so developers can configure properties of workflow activities either through the graphical interface or directly in the code-beside file, and always keeps the two in sync
* Debugging of your workflows by attaching to the Windows SharePoint Services process, including setting breakpoints in your workflow
* The ability to attach error, compensation, and event handlers to activities, and "comment out" activities graphically

To further assist your workflow development, Microsoft provides two Visual Studio project template packs for use with the Visual Studio 2008 Designer for Windows Workflow Foundation: one customized for Windows SharePoint Services workflows, the other customized for creating workflows for SharePoint Server.

The Window SharePoint Services workflow project template contains a reference to the Windows SharePoint namespace and includes custom workflow activities designed specifically for the Windows SharePoint Services environment. These custom activities enable you to perform functions common to the Windows SharePoint Services environment, such as creating, updating, completing, and deleting SharePoint tasks, sending e-mail, and enabling workflow modifications.

The SharePoint Server workflow project contains everything the Window SharePoint Services workflow project template, as well as a reference to the SharePoint Server namespace, and additional functionality for managing workflow tasks in the SharePoint Server work environment.

**The Workflow Development Process in Visual Studio**

In general, when you develop workflows for Windows SharePoint Services or Office SharePoint Server using the Visual Studio 2008 Designer for Windows Workflow Foundation, you follow these basic steps:

1. Author your workflow, including the code-beside file if needed, in the Visual Studio 2008 Designer for Windows Workflow Foundation.
2. Design and publish any forms you want to use with your workflow.
3. Author the feature definition and workflow template definition file, which contains information about the workflow assembly, and binds the forms to the workflow assembly.
4. Compile the workflow files into a .NET assembly.
5. Package the workflow assembly and workflow definition together and deploy them by using the features functionality in Windows SharePoint Services.
6. Debug the live workflow assembly using the Visual Studio 2008 Designer for Windows Workflow Foundation.
7. Recompile and deploy the workflow assembly as necessary to fix any bugs found.

The following sections provide an overview of each of these development steps.

**Creating Workflows by Using the Visual Studio 2008 Designer for Windows Workflow Foundation**

When you select a new Windows SharePoint Services or Office SharePoint Server workflow project, Visual Studio displays the Visual Studio 2008 Designer for Windows Workflow Foundation design surface (Figure 1). This design surface presents a graphical interface, where you can assemble workflows by dragging and dropping various activities from the **Toolbox**.

**Figure 1. Visual Studio 2008 Designer for Windows Workflow Foundation interface**

When you drag a specific activity onto your workflow, the Visual Studio 2008 Designer for Windows Workflow Foundation shows you the valid locations for that activity within the workflow. You cannot place an activity in a location where it would not be valid. For example, you cannot position a Send activity as the first activity in a Listen activity branch. As shown in Figure 2, Visual Studio 2008 Designer for Windows Workflow Foundation displays green plus-sign icons to denote valid positions for the specific activity.

**Figure 2. Valid positions for a Workflow activity**

As you graphically design your workflow, the Visual Studio 2008 Designer for Windows Workflow Foundation actually generates corresponding markup.

In addition, if you are working with code separation, your workflow project contains a code-behind file, in which you program the business logic of your workflow. You can switch between working in the workflow design surface and the code-beside file whenever you want.

**Setting Workflow Properties**

After you add the activity to the workflow, you must set the properties of the activity for it to be valid within the workflow. You can set these properties by using the standard Visual Studio Properties window. The type of data you can specify depends on the type of data the property itself accepts: literal values, variables, or method names.

If you want to specify a variable for the property, you can either enter the variable name in the Properties window, in which case the Visual Studio 2008 Designer for Windows Workflow Foundation automatically declares the variable in the code-behind file; or declare the variable in the code-behind file, and then select it in the Properties window.

Certain activity properties are essentially references to methods in the code-beside file that conform to a specific signature. As with variable names, you can enter the method name in the Properties window and let the Visual Studio 2008 Designer for Windows Workflow Foundation add the method signature to the code-behind file, or you can create the method in the code-behind file and then select the method in the Properties window.

You can also bind a property to the property of another activity.

**Handler Activities**

A workflow might have several potential points of failure. It is important to track the state of a workflow and to report errors as they occur, so that you can resolve problems accurately and with a minimum of effort. It is equally important for a workflow to maintain the integrity of a set of closely related actions, so that if part of an operation takes place but another does not, the entire operation can be rolled back. You can use the **FaultHandlerActivity**, **TransactionScopeActivity**, **CompensationHandlerActivity**, and **CancellationHandlersActivity** activities to handle errors, to maintain the state of a workflow, and to fix problems as they occur.

You can think of a **FaultHandlersActivity** activity as a C-language **try** block, to which you can attach an ordered set of **FaultHandlerActivity** activities that act as exception handlers. These exception handlers can be thought of as C-language **catch** blocks. If an exception is thrown during the execution of a composite activity, the WF run-time engine matches the exception against the exception types that are handled by the **FaultHandlerActivity** activities. If the run-time engine finds no matching exception handler, it then communicates the exception to the next higher composite activity, where the process repeats, and so on, until an appropriate handler is found.

You can also have **EventHandlersActivity** activities that respond to events, by adding event handlers to an **EventHandlerScopeActivity** activity. Conceptually, these event handlers are very similar to those in the C or Visual Basic .NET languages. To create an event handler, you must use **EventDrivenActivity** activities.

**CompensationHandlerActivity** activities contain code that compensates for, or rolls back, the operations of a composite activity if it does not perform successfully.

**ASP.NET Forms in Windows SharePoint Services Workflows**

As mentioned earlier, you can use ASP.NET to create the forms you use with your Windows SharePoint Services workflow. These forms are then displayed in the Windows SharePoint Services user interface at the appropriate stages in the workflow.

Workflow forms are late-bound to the workflow assembly through information you supply in the workflow template definition XML file. The workflow template definition schema contains elements for denoting the URL of the various types of forms you can use with Windows SharePoint Services workflows. This provides the ability to create elements for forms for any custom workflow modifications, as well as forms for the various types of SharePoint tasks used in the workflow.

In most cases, the workflow assembly itself contains no information about or link to the workflow forms themselves. Developers can change the workflow forms to be used just by editing the workflow definition XML, without having to recompile the workflow assembly itself. The one exception is workflow modifications; each activity that enables a workflow modification must contain the GUID of the form for that workflow modification.

**Deploying Workflows**

When you are finished specifying your workflow, you can choose to compile your workflow as a workflow, or as an activity.

After you have compiled your workflow, you can use the SharePoint Features functionality to package and deploy the workflow assembly and any necessary supporting files.

Feature packaging is a way of encapsulating Windows SharePoint Services solutions and functionality for ease of deployment. It provides a mechanism by which developers can package the files needed for a solution, such as workflows, Web Parts, lists, and site definitions, for ease of distribution and deployment. Developers package the necessary files into a .wsp file, which is essentially a .cab file containing a manifest that lists its contents. For more information about SharePoint Features, see the Microsoft Windows SharePoint Services 3.0 SDK.

Each workflow template you deploy must contain a workflow template definition file. A workflow template definition is an XML file that contains the information that Windows SharePoint Services requires to instantiate and run the workflow, such as the following:

* The name, GUID, and description of the workflow template
* Assembly
* The URLs (or URNs for IP forms) of any custom forms used with this workflow template
* Optionally, the names of the workflow, workflow engine, and host service assemblies to use when running the workflow
* Optionally, the correct classes within those assemblies to call

The workflow assembly itself must be deployed to the global assembly cache.

After it is deployed to a site, the workflow is available as a workflow template, which SharePoint administrators can associate with the document libraries and lists on that site.

**Debugging Workflows**

After you deploy your workflow assembly, you can debug your workflow by opening your workflow project and attaching to the Windows SharePoint Service *w3wp* process.

Because the Visual Studio 2008 Designer for Windows Workflow Foundation is hosted within Visual Studio, you can take full advantage of the debugging capabilities of Visual Studio. You can set breakpoints both in the code you write in your code-beside file, as well as on workflow activities in the design surface.

The Visual Studio 2008 Designer for Windows Workflow Foundation not only supports standard Visual Studio debugging features such as the Breakpoints and Call Stack windows, it also includes a range of visual indicators that provide information during the debugging process. You can also add breakpoints to a workflow activity while you are debugging the assembly.

You can perform step in, step out, and step over operations to move through workflow code.

# **Additional Resources and References**

# [Developer Introduction to Workflows for Windows SharePoint Services 3.0 and SharePoint Server 2007](http://msdn.microsoft.com/en-us/library/aa830816.aspx)