Authoring Windows SharePoint Services Workflows

What is a Windows SharePoint Services workflow? Fundamentally, it consists of two things: the forms a workflow uses to interact with its users and the logic that defines the workflow’s behavior. Understanding how Windows SharePoint Services workflows are created requires knowing something about both.

Because it communicates with users through a Web browser, a Windows SharePoint Services workflow relies on Microsoft ASP.NET to display its forms. Accordingly, those forms are defined as ASPX pages. A Windows SharePoint Services workflow can potentially display its own forms at four points in its lifecycle:

|  |  |
| --- | --- |
| • | Association: When a Windows SharePoint Services administrator associates a workflow template with a particular document library or list, he might be able to set options that will apply to every workflow instance created from this association. If a workflow author chooses to allow this, she must provide a form that lets the administrator specify this information. |
| • | Initiation: The initiator of a workflow might be allowed to specify options when he starts a running instance. In the approval scenario just described, for instance, the options included specifying the list of workflow participants and defining how long each one had to complete his or her task. If a workflow allows this, its author must provide a form to allow the initiator to set these options, as shown in step 3 of the previous diagram. |
| • | Task Completion: The running workflow instance must display a form to the participants in the workflow to let them complete their task. Shown in step 6 in the previous diagram, this form is what allowed the approvers in the earlier scenario to make comments on the document and indicate their approval or rejection. |
| • | Modification: Although it wasn’t shown in the scenario above, the creator of a workflow can allow it to be modified while it’s running. For example, a workflow might allow adding new participants after it has begun executing or extending the due date for completing tasks. If this option is used, the workflow must display a form at this point to let a participant specify what changes should be made. |

Workflows built solely on Windows SharePoint Services define their forms as .aspx pages, while those using Office SharePoint Server can also use forms created with InfoPath, as described later. In either case, a workflow’s logic is always defined as a group of activities, just as with any WF-based workflow. To specify the logic and forms for a workflow, Microsoft provides two different tools, each targeting a different audience. Software developers can use WF Workflow Designer hosted in Visual Studio 2005. Information workers, a less technical group, can use Office SharePoint Designer to create workflows without writing code. The next two sections examine how Windows SharePoint Services workflows can be created using each of these tools.

**Authoring Workflows with Visual Studio 2005 and WF Workflow Designer**

In many ways, a workflow is like a flowchart. Given this, it makes sense to provide a graphical tool that lets developers specify a workflow’s actions. For WF, this tool is Workflow Designer, part of Windows Workflow Foundation Extensions for Visual Studio 2005. Developers can use WF Workflow Designer to define graphically a workflow’s activities and the order in which those activities should be executed. The screen below shows a simple example of how this looks.

The activities available for use appear in the Toolbox on the left side of the screen. A developer can drag these activities onto the design surface to define the steps in a workflow. The properties of each activity can then be set in the Properties window that appears in the lower right corner.

WF’s Base Activity Library provides a group of fundamental activities, as described earlier. Windows SharePoint Services also provides a set of activities designed expressly for creating Windows SharePoint Services workflows. Among the most important of these are the following:

|  |  |
| --- | --- |
| • | OnWorkflowActivated: provides a standard starting point for a Windows SharePoint Services workflow. Among other things, this activity can accept information supplied by a Windows SharePoint Services administrator via the Association form when the workflow is associated with a document library or list. It can also accept information supplied via the Initiation form when the workflow is started. Every Windows SharePoint Services workflow must begin with this activity. |
| • | CreateTask: creates a task assigned to a particular user in a task list. For example, the approval workflow in the scenario described earlier used this activity to add a task to the task list used by each of the participants. This activity also has a SendEmailNotification property that, when set to true, automatically sends an email message to the person for whom this task was created. |
| • | OnTaskChanged: accepts information from the Task Completion form. The approval workflow in the earlier scenario used this activity to accept the input of each participant when the document was approved. |
| • | CompleteTask: marks a task as completed. |
| • | DeleteTask: removes a task from a task list. |
| • | OnWorkflowModified: accepts information from the Modification form, which can then be used to change how this instance of the workflow behaves. If the workflow’s creator chooses not to include any instances of this activity in the workflow, that workflow cannot be modified while it’s running. |
| • | SendEmail: sends email to a specified person or group of people. |
| • | LogToHistoryList: writes information about the workflow’s execution to a history list. The information in this list is used to let users see where a workflow is in its execution, look at the workflow’s history after it’s completed, and more. To allow this kind of monitoring, the workflow’s author must write information to a History list at appropriate points in the workflow’s execution. Because it provides its own mechanism for tracking workflows, Windows SharePoint Services doesn’t support WF’s standard tracking service. |

A typical pattern for a simple Windows SharePoint Services workflow begins with an OnWorkflowActivated activity, then uses a CreateTask activity to assign a task to a participant in the workflow. The BAL’s standard While activity might then be used to wait until the user completes the task. To learn when this has happened (perhaps the user makes multiple changes to the task, then checks a box on the Task Completion form when she’s done), an OnTaskChanged activity executes within the While, extracting whatever information the user has entered on that form. When the user has completed the task, a CompleteTask activity might execute, followed by a DeleteTask. The workflow can then go on to the next participant, using CreateTask to assign a task to him, and so on. And of course, other things can occur, such as sending email, logging information to the history list, or even including the BAL’s Code activity, which allows running arbitrary code.

All of the activities provided by Windows SharePoint Services are concerned with letting workflows operate within the Windows SharePoint Services environment. The business logic a workflow implements is entirely up to the creator of that workflow. In fact, a developer authoring a Windows SharePoint Services workflow is free to create and use her own custom activities—she’s not required to use only those provided by Windows SharePoint Services and WF.

As described earlier, Windows Workflow Foundation supports both sequential and state machine workflows. A Windows SharePoint Services workflow created with the WF Workflow Designer can also use either option. To allow this, Windows SharePoint Services adds two project types to Visual Studio 2005, one for each of these workflow styles.

Whatever style is chosen, the developer must define more than just the workflow’s logic; he must also specify the ASPX forms it should use. To do this, the developer relies on a file named workflow.xml. This file provides a template that the developer fills in to specify what form, if any, should be displayed at each of the four points at which a Windows SharePoint Services workflow is allowed to do this.

A developer must do some work to pass information between a Windows SharePoint Services workflow and the ASPX forms it uses. Windows SharePoint Services provides a namespace, Microsoft.Windows.SharePoint.Workflow, that exposes an object model for developers. Using the types in this namespace, the creator of a Windows SharePoint Services workflow can pass information from an ASPX form to the workflow and vice-versa.

Once a Windows SharePoint Services workflow and its forms have been created, the developer must package them into what Windows SharePoint Services 3.0 refers to as a feature. A Windows SharePoint Services administrator must then install this feature, which includes installing the workflow’s assemblies to the target system’s global assembly cache. The new workflow will now be visible to the administrator as a workflow template that can be associated with a document library or list.

For a software developer, creating a Windows SharePoint Services workflow using Visual Studio and the WF Workflow Designer isn’t especially hard. The developer needs to understand the specifics of working in this environment, but much of what he’s doing will be familiar. Yet software developers aren’t the only people who’d like to author Windows SharePoint Services workflows. As described next, people who aren’t professional developers can also create workflows using Office SharePoint Designer.

**Authoring Workflows with Office SharePoint Designer 2007**

Office SharePoint Designer 2007, a separately licensed component of the 2007 Office system, allows information workers and others to add application logic (implemented as a workflow) to Windows SharePoint Services sites. This is certainly a useful goal, but Office SharePoint Designer also addresses another important problem. If a developer creates a Windows SharePoint Services workflow using Visual Studio, that workflow must be installed on a Windows SharePoint Services server like any other application. Yet many Windows SharePoint Services administrators won’t allow arbitrary code to be deployed on their servers, believing that the risk of destabilizing the system is too great. Being able to create straightforward business logic tied to documents and list items is very useful, however, and it’s something that many Windows SharePoint Services users need. Along with allowing less technical people to create workflows, Office SharePoint Designer also addresses this problem by providing a safer way to define and deploy business logic on Windows SharePoint Services servers.

The workflow scenarios that Office SharePoint Designer is intended to address are different in some ways from those addressed by Visual Studio and WF Workflow Designer. While it’s certainly possible to create complex applications, the intent of Office SharePoint Designer is to let users add business logic to Windows SharePoint Services sites. For example, suppose that a site contains a list that allows its users to submit change requests. Office SharePoint Designer could be used to create a workflow that automatically informs the submitter when her change request is accepted or rejected. Similarly, a custom workflow might inform a particular group of users whenever a new document is added to a particular document library. Performing this kind of custom notification isn’t complicated—creating the workflows is easy—but it’s challenging with earlier versions of Windows SharePoint Services because of administrators’ reluctance to install user-written code.

There’s an obvious question here: why should logic created with Office SharePoint Designer be treated any differently? What makes Windows SharePoint Services administrators willing to allow workflows built with this tool to be deployed on the systems for which they’re responsible? The answer is that workflows built with Office SharePoint Designer can only use activities from an administrator-controlled list. While a site’s administrator can choose to include custom activities created by a developer on this list, he’s not required to. By defining exactly what workflows are allowed to do, a Windows SharePoint Services administrator can have more confidence that deploying logic created using Office SharePoint Designer won’t destabilize his system.

Both because it’s intended for information workers rather than developers and because it emphasizes simpler scenarios, Office SharePoint Designer uses a different model for creating workflows than the Visual Studio-hosted WF Workflow Designer. Instead of a graphical approach, Office SharePoint Designer uses a rule-based approach. It’s somewhat similar to the Rules Wizard in Outlook, a tool that’s familiar to many people. The screen below illustrates how a user of Office SharePoint Designer defines a step in a workflow.

Each step can have a condition and an action. The condition determines whether this step’s action should be executed, as in the If statements shown above. The choices for actions include things such as assigning a To-do item to a workflow participant, sending email, and many more. Each of these actions is actually carried out by some Windows SharePoint Services activity, and the activities used here are the same as with Visual Studio and WF Workflow Designer. For example, the **Send an Email** option shown above corresponds to the SendEmail activity, while **Assign a To-do Item** corresponds to a slightly specialized version of the CreateTask activity. The list of actions can also include any other activities allowed by the Windows SharePoint Services administrator for this site, including custom activities created by developers.

Even though its user interface looks quite different from the graphical approach used with Visual Studio and WF Workflow Designer, Office SharePoint Designer creates a standard WF workflow. What’s actually produced is a sequential workflow with conditions expressed using the WF rules engine. Workflows created with this tool do have some limitations, however. For example, they can’t be modified while they’re running, unlike those built using Visual Studio and WF Workflow Designer, and only sequential workflows can be created—state machines aren’t supported. Also, workflows built with this tool must be authored against a specific document library or list when they’re designed. Creating a general workflow template that can later be associated with any library or list or with a content type isn’t possible. While this does place limits on how a workflow can be used, it also makes deploying the workflow much simpler. In fact, when a user finishes authoring a workflow with Office SharePoint Designer, the tool automatically deploys the workflow to the target site. This is significantly less complicated than the multi-step deployment process required for workflows created using Visual Studio and WF Workflow Designer.

Workflows created using Office SharePoint Designer can also display customized forms. Rather than require workflow authors to create ASPX pages directly, however, the tool instead generates those pages. The author specifies details about how the generated pages should look, such as what fields they should contain, and Office SharePoint Designer takes care of the rest. Of the four points in a Windows SharePoint Services workflow’s lifecycle where forms can be used, however, only two are used with workflows created using Office SharePoint Designer: Initiation and Task Completion. Because every workflow created with this tool must be associated with a particular document library or list, there’s no need for an association step and hence no Association form. And since these workflows can’t be modified while they’re running, there’s no need for a Modification form.

Office SharePoint Designer can be used for things other than creating Windows SharePoint Services workflows. The tool can be used to create a Windows SharePoint Services site, for example, or to customize the look and feel of a site’s pages by editing the site’s master page. It can also be used to connect to external data, something that relies on the data binding support in ASP.NET. For information workers who need to create logic that executes on a Windows SharePoint Services site, however, the most important aspect of Office SharePoint Designer is surely its support for authoring workflows.

Windows SharePoint Services provides a great deal of functionality for creating document-oriented workflows. Yet ultimately, it’s a platform for development and execution. On its own, it provides no workflow functionality that’s directly usable by end users. If what’s required is out-of-the-box workflow applications—and it often will be—Windows SharePoint Services isn’t enough. Workflows created using just Windows SharePoint Services also have other restrictions, such as the inability to interact with participants via Office client applications. As described next, Office SharePoint Server offers a way to overcome all of these limitations.