The Annals of "Dunarea de Jos" University of Galati
Fascicle I – 2008. Economics and Applied Informatics. Years XIV - ISSN 1584-0409

Businesses Integration with Workflow and Web Service Technologies

Gianina RIZESCU
grizescu@ugal.ro
"Dunarea de Jos" University of Galati

Abstract. If businesses want to benefit from the power of the Internet, web sites have to evolve. It is often no longer beneficial for them to only provide static information. It is necessary for these web sites to find ways that allow them to interact with other websites, operating systems, and applications. With Web services it is finally possible to create functions that can easily be accessed over the Internet by both internal and external parties. In other words, with Web services it is possible to integrate different value chains from different organizations with ease. On the other hand, workflows provide tools and mechanisms for managing the interaction between people, systems, applications and business functions. The combination of these two technologies offers a very powerful solution for businesses integration.

Keywords: Workflow, Web Service, Integration, Windows Workflow Foundation (WF), Service Orientated Architecture (SOA)

1. Introduction

Workflows provide a mechanism for modelling and implementing a business process as a series of activities that manage the interaction between people and back-end systems to carry out a business function. Business processes (or business applications) that require intense human involvement normally are executed over long periods of time due to the complex coordination of multiple entities. Using workflows to implement such business processes provides a mechanism for coordinating, aggregating, and routing business data.

Due to the distributed nature of a business process, coordinating information coming from multiple sources across an organization, it makes sense for a workflow to be deployed as a distributed application.

Web service definitions are designed to abstract how applications communicate with each other. They are used to model the types of interactions a workflow requires in order to carry out a business function. By using Web services, it is possible to implement these interactions as distributed interoperable services that support application-to-application communications in a standard manner. These services allow the decoupling of business logic from client application code. This mechanism can be used to generalize and publish service implementations and have them consumed by multiple clients.

Combining these two technologies provides the ability to expose business processes as Web Services.

2. The power of the Web Services

Web services are a set of protocols based on XML (Extensible Mark-up Language). The following base protocols formed the initial specification for Web services [2],[11]:

- *Simple Object Access Protocol (SOAP)* - defines the runtime message that contains the service request and response. SOAP is independent of any particular transport and implementation technology.
- *Web Services Description Language (WSDL)* - describes a Web service and the SOAP Message. It provides a programmatic way to describe what a service does, preparing the way for automation.
- *Universal Discovery, Description, Integration (UDDI)* - UDDI is a cross industry initiative to create a standard for service discovery together with a registry facility that facilitates the publishing and discovery processes.

These have effectively become de facto standards, with effectively universal acceptance and widespread implementation by vendors. Figure 1 shows the way their application is typically illustrated.

The main goal of Web services is to enable software developers to easily integrate different kinds of applications and services with each other, without having to worry about the underlying protocols, interfaces, environmental conditions, etc.

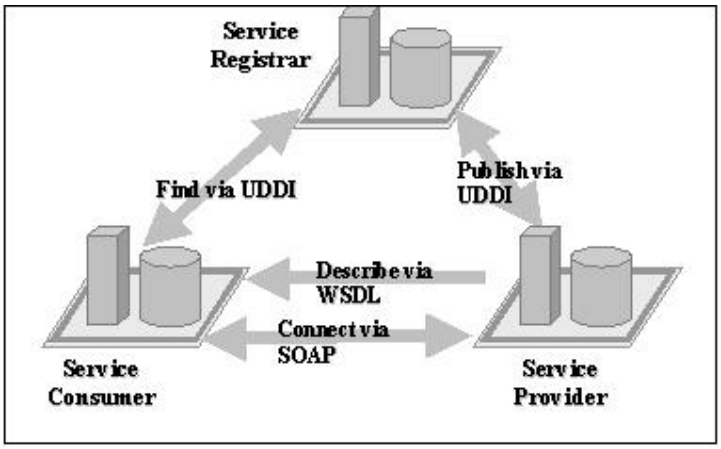

Figure 1. Base Web Service Protocols [11]

3. The power of the Workflows

In a collaborative application built of Web services, where the business process is really a set of tasks whose participants are Web services, workflow control is overwhelming and interaction of disparate workflows is inevitable.

For fulfil its role of integration, a workflow software must accomplish at least the following functions[10]:

- The workflow application must have the capability to update workflows easily when processes and organizations change. Also, a workflow application must help the enterprise adhere to government and organizational regulations by standardizing and monitoring business processes. For industries like Manufacturing for example, workflow applications must enhance production flexibility and enable production system load-balancing.
- In order to work well with application integration software APIs, a workflow application must provide flexible Java support that allows that workflow application to be integrated

with Web applications and other IT applications, and must support integration with existing business applications. For example, the workflow application must be able to support a nested workflow within an external host workflow system.
- Workflows can be made available as Web services and can also control the flow of a set of Web services that make up an application.
- Workflow applications must be the binder for collaborative applications. Collaborative applications most often refers to applications that are built of other applications/Web services required to perform tasks and that manage all of the interaction and data flow. Collaborative applications can also mean the automating and streamlining of people-based business processes so that the people involved in the workflow become more productive individually and, more significantly, as a team.

4. Integrating the Workflows with Web Services in Windows Workflow Foundation (WF)

When building applications today the call of using Web services is strong. Web services provide an easy way to build distributed applications because of their use of open protocols and data formats (generally HTTP and XML).

Service Oriented Architecture is a new way of thinking about how to build distributed applications in this new Web services world. The ideas of SOA are actually fairly simple and straightforward. They are based around four basic principles [1]:
- boundaries are explicit;
- services are autonomous;
- services share schema and contract, not class;
- service compatibility is based on policy.

Some say that SOA is missing one importance piece: Workflow. We'll explain here, how Windows Workflow Foundation can integrate workflows and Web services.

Calling a Web Service

Calling a Web service from .NET code is fairly straightforward. The .NET framework provides the *SoapHttpClientProtocol* class (found in the *System.Web.Services.Protocols* namespace). This class knows how to invoke Web services. It is able to turn a .NET call stack into a SOAP Envelope that can be sent to a Web service using HTTP, and receive a corresponding response SOAP Envelope, deserialize the response, and return it to the caller as a .NET type.

In order to call a Web service from a .NET project, a *Web Reference* must be added to the project. Adding a Reference to a project adds a .NET assembly-level reference to whatever assembly is specified.

Adding a *Web Reference* causes VS.NET to generate a code that allows it to call upon a Web service in the same way that it would call into any .NET object referenced from an external assembly. The *Add Web Reference* command in VS.NET is a code generation tool that is available from the Project menu, or from the context menu of the project node in the Solution Explorer.

The generated code allows to call upon a Web service that uses the definition provided by a *Web Service Description Language (WSDL)* file. When the *Add Web Reference* command is used, a dialog appears and an URL can be typed in the address bar. This can either be an URL to a WSDL file (which can either live on the local machine or on the network) or the URL to an ASMX Web service.

When finished, this functionality takes care of generating a new class that can invoke a Web service that implements the specified document (this is done by running a code that lives in an SDK tool named WSDL.exe—so this can be done from a command line as well).

SoapHttpClientProtocol is a class that is programmed to call any basic Web service. The derived class that is created from the *Add Web Reference* is generated to facilitate the ease of calling the Web service by creating a set of wrapper methods that closely mimic the operation or the operations that the WSDL defines.

This base class knows how to turn the call stack from the .NET method that is called from (like *Add*) into the appropriate SOAP call to the Web service. It uses the *XmlSerializer* class and other classes in the *System*.Xml namespace to accomplish this.

Using *Web References* is the general way of programming against Web services from any .NET based project in VS.NET.

Calling a Web Service from a Workflow

In order to call a Web service from a workflow, the *InvokeWebService* activity must be added to the workflow. The *InvokeWebService* activity allows to do this by building on top of this typical .NET facility for calling a Web service - the *SoapHttpClientProtocol* class. *InvokeWebService* activity requires four (or more) properties to be set for it:

- *ReturnValue*: This is a bind property that must be set to a field in workflow to keep the value that is returned from the Web service method;
- *MethodName*: Name of Web service method that can be chosen from a drop down list;
- *ProxyClassName*: Name of the proxy class for the Web service. Add Web Reference dialogue sets this property automatically but it can be created manually as well;
- *URL*: string value of Web service address;
- *Method Parameters*: The Web service may have none, one or more input parameters with different types. If it has any input parameters then, these must be specified as bind properties for the activity. There are two possibilities: to choose from existing fields (Bind to an existing member) in workflow or to create new field (Bind to a new member). If the second is chosen, then it will generate the code automatically.

When executed, the *InvokeWebService* activity is going to call the method specified on an instance of the proxy class. The call will pass as input parameters whatever the values of the variables were picked as input parameters at the time of the call, and setting the value of the variable that was picked as the return value with the return of the Web service operation. Such simple workflow looks like in Figure 2.

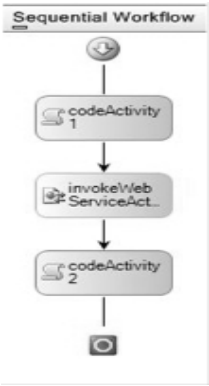

Figure 2. Invoke WebService from a Workflow

As it was shown in the above image, there are some basic steps that must be followed whenever the *InvokeWebService* activity is used:

- adding the activity to the workflow and specifying the URL to the WSDL of the Web service that is need to be invoked;
- specifying the *MethodName*, and then specifying the field or property name(s) for the parameters to that method;
- adding the code or activity before the *InvokeWebService* activity that will ensure that the input parameters will be initialised with the correct values;
- adding the code or activity after the *InvokeWebService* activity that will harvest the return value and execute the appropriate code for workflow;

The other aspect of Web services is around sessions. ASP.NET Web services (ASMX services) support cookies sessions via cookies in HTTP protocol out of the box and is not necessary to take care about these details in Web services. In workflows sometimes a Webservice needs to be called several times and needs to keep sessions during all these calls. Fortunately *InvokeWebService* activity comes with *SessionID* property.

This property helps to keep sessions for all calls to a Web service from different activities. A unique string may be putted in this property as its value for all activities (more simple, a GUID may be used to ensure its uniqueness between all activities). Windows Workflow will do the rest and makes sure that all activities use the same sessions during all calls.

Such a workflow and the *InvokeWebService* activities properties can be seen in Figure 3. The two *InvokeWebService* activities will have the same value for the *SessionID* property, and so the Web service will be able to keep session state for the workflow between those two invocations.

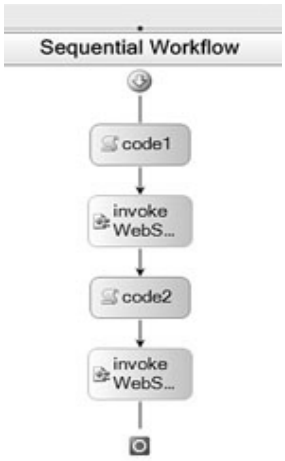

Figure 3. Invoke Web Service with sessions

Exposing a Workflow through a Web Service

The Windows Workflow Foundation framework supports Web service interoperability that includes also the ability to expose a workflow as Web service to ASP.NET clients and to other workflows. Windows Workflow Foundation supports publishing a workflow as an ASP.NET Web service on a Web server. Because Windows Workflow Foundation Web service support is based on ASP.NET 2.0, it inherits most of the features of a standard ASP.NET Web service.

The Windows Workflow Foundation base activity library contains the *WebServiceInput* and *WebServiceOutput* activities, which enable a workflow to be used as Web service end points. *WebServiceInput* enables receiving data from a Web service in a workflow and *WebServiceOutput* enables sending data to a Web service from within a workflow.

To create a workflow as a Web service, in the first place an interface must be created and expose the methods of that interface as Web service methods. Typically the interface and the workflow are hosted in a separate assembly and then referenced from within the ASP.NET Web service project. However once the interface and workflow are created, it is not necessary to manually create the ASP.NET Web service project as Visual Studio can auto-generate the ASP.NET Web service project.

There are three key activities which are going to have to work with in order to be able to expose the workflow as a Web service:

- *WebServiceInput* activity. Each workflow based Web services require at least one *WebServiceInput* activity and one or more *WebServiceOutput* activities. As the name suggests, the *WebServiceInput* activity accepts the input parameters for future processing by the subsequent steps in the workflow;
- *WebServiceOutput*. This activity completes the Web service's processing by returning the return value identified by the *MethodName* in the *WebServiceInput* activity. For this reason, the InputActivityName property must be set to one of the *WebServiceInput* activities present in the workflow.
- *WebServiceFaul*. This activity allows to bind the exceptions generated during the Web service execution and translates them into *SoapException* objects. Closely related to *WebServiceOutput* activity in that it also indicates the termination of workflow processing;

To expose a Workflow through a Web service, the following steps are needed:

1. Creating a new VS.NET Sequential Workflow Library project;
2. Creating the interface that consumers will see when they want to use the service. It defines any methods or properties that will be implemented by the Web service;
3. Dragging an instance of *WebServiceInputActivity* activity onto the workflow designer from the toolbox and configuring the properties of *WebServiceInput* activity as follows:
    - Interface - WorkflowName.InterfaceName;
    - IsActivating – True;
    - MethodName – one of the methods defined in the Interface;

The *IsActivating* property needs to be set to True for the first *WebServiceInput* activity in a workflow. When an application using this web service makes a call to this web service and executes *ProcessWorkflow*, the workflow engine will know to call this *WebServiceInput* activity.

4. Dragging *CodeActivity* from the Toolbox, and dropping it below webServiceInputActivity1. (see Figure 4) This *CodeActivity* will include de code for whatever the workflow does.
5. Dragging an instance of *WebServiceOutput* activity onto the workflow designer from the toolbox and setting the *InputActivityName* property to be *webServiceInputActivity1* defined at the step 3. A *ReturnValue* is added to the list of *webServiceOutputActivity1* properties. Now, it is necessary to bind a workflow property to the ReturnValue property of *webServiceOutputActivity1*.

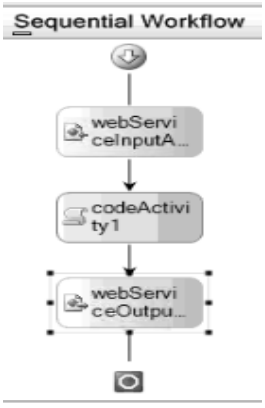

Figure 4. Exposing a Workflow through a Web Service

6. Up to this step, we presented how a workflow can be created. The next step is to expose the workflow as a service so that the client applications can execute the workflow as a service. To publish the workflow as a web service, the *Publish as Web service* property of *WorkflowProject* must be used. As a result of this, a new solution (with .asmx extension) will be created within the Solution Explorer, and all the necessary files are also created. To view the created Web service, the *View In Browser* property of the .asmx solution must be used. This opens the .asmx file in the browser where the Web service can be tested.

Conclusions

With Web Services it becomes possible to overcome the physical boundaries of existing organizations. By using Web Services there are no longer constrains such as: operating systems, object models, and programming models. We've seen in this paper that Windows Workflow Foundation offers rich integration facilities with Web services, allowing the invocation of Web services directly from the workflow using the *InvokeWebService* activity, as well as offering implementation options for exposing workflows via Web Services using the *WebServiceInput* and *WebServiceOutput* activities. We can take advantage of the rich, declarative Windows Workflow Foundation model to expose the business processes, while leveraging Web Services as a deployment technology for executing them.

References

1. *Andrew, P. et. al, Presenting Windows Workflow Foundation, Sams Publish House, 2005;*
2. *Duivestein, S. - Web Services and Workflow - Organizing Web Services, 2001, available at: http://www.webservicesarchitect.com/content/articles/sander01.asp;*
3. *Kitta, T. - Professional Windows Workflow Foundation, Wronx Press, 2007;*
4. *Morais, P. - Dynamic e-business using Web service workflow, Techtarget white paper, 2002, available at: http://searchsoa.techtarget.com/originalContent/0,289142,sid26_gci834488,00.html*
5. *Nayyeri, K. - Invoke a WebService from a Workflow, 2007, available at: http://nayyeri.net/ archive/ 2007/03/30/invoke-a-webservice-from-a-workflow.aspx;*
6. *Razi Bin Rais, Understanding the Windows Workflow Foundation WF: From a Business User's Perspective, Code Project white paper, 2006, available at: http://www.codeproject.com/ dotnet/UnderstandWWF.asp;*
7. *Rubio, D. - Windows Workflow Foundation for Web services, Techtarget white paper, 2006, available at: http://searchsoa.techtarget.com/tip/0,289483,sid26_gci1204115,00.html;*
8. *Shukla, D., Schmidt, B. - Essential Windows Workflow Foundation, Addison Wesley Publish House, 2006;*

9. *Thangarathinam, T. - Windows Workflow Foundation – Part 2, 2007, available at: http://dotnetslackers.com/ articles/wf/WindowsWorkflowFoundationPart2.aspx;*
10. *Virdell, M. - Business processes and workflow in the Web services world, white paper, 2003, available at: http://www.ibm.com/developerworks/webservices/library/ws-work.html;*
11. *Wilkes, L. - The Web Services Protocol Stack, white paper, 2005, available at: http://roadmap.cbdiforum.com/reports/protocols/*
12. **** Microsoft white paper, Deploy Distributed Business Processes With Windows Workflow And Web Services, available at: http://msdn.microsoft.com/msdnmag/issues/06/10 /webserviceworkflows/;*
13. **** WF - Exposing a workflow through a web service, white paper 2006, available at: http://community.bartdesmet.net/blogs/bart/archive/2006/09/03/4388.aspx;*
14. *http://msdn2.microsoft.com/en-us/default.aspx;*
15. *http://msdn2.microsoft.com/en-us/webservices/default.aspx;*
16. *http://msdn2.microsoft.com/en-us/netframework/;*
17. *http://www.e-workflow.org/*
18. *http://www.w3.org/2002/ws/;*
19. *http://www.wfmc.org*