The Web services insider, Part 6: Assuming responsibility

Implementing roles in WSFL

James Snell (jasnell@us.ibm.com)
Software engineer, Emerging Technologies, IBM
July 2001

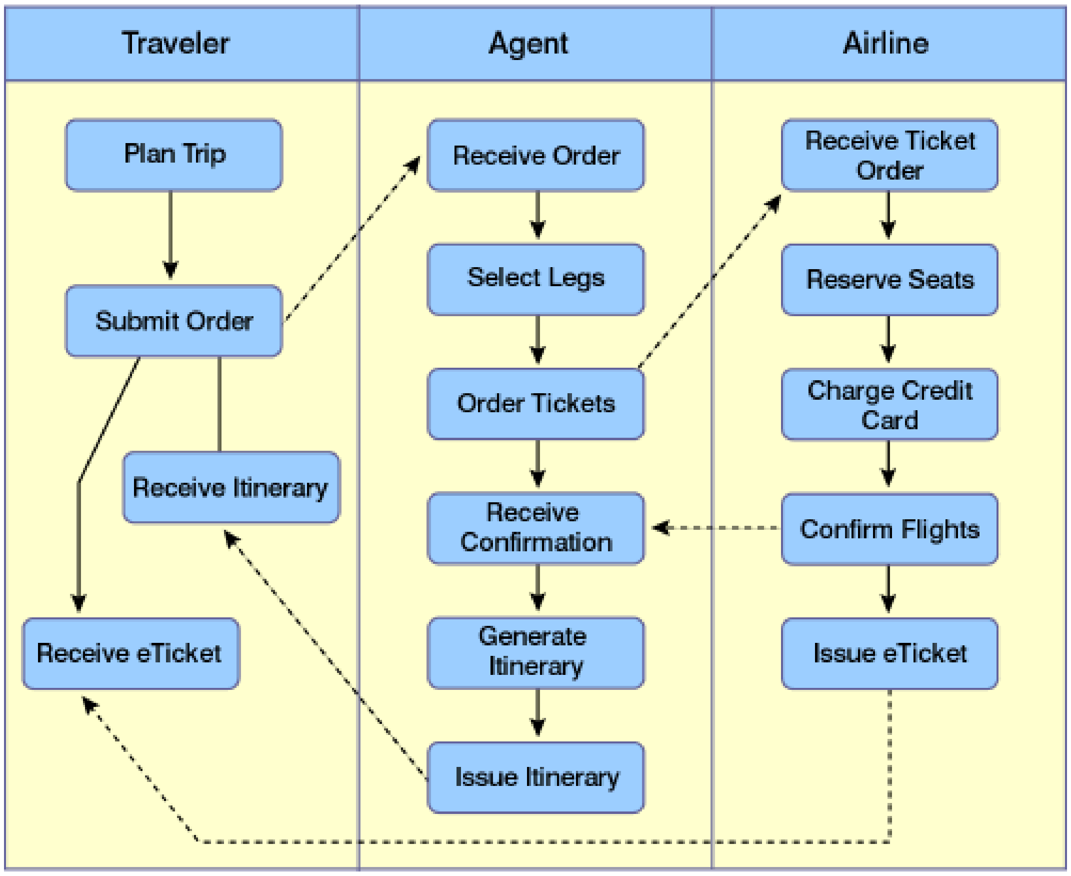
Web services offer the potential for creating highly dynamic and versatile distributed applications that span technological and business boundaries, allowing service providers and service consumers to improve the way they do business. Web Service Flow Language (WSFL) extends this potential by building a framework in which service providers and consumers can come together to implement standard business processes; a place where the "Who is doing what" is less important than the "What is being done." This framework allows anyone who properly implements the appropriate Web service interfaces to assume the various roles of business process. In this installment of the Web service insider, I continue my discussion of WSFL, focusing on how to become a service provider.

The previous Web services insider installment (see Resources) introduced business process modeling and the concept of service provider types that fulfill the various responsibilities of implementing those business processes. In this installment, I'm going to take a much more in-depth look at becoming a WSFL service provider. The good news is that it takes nothing more than properly implementing either a single WSDL-defined Web service interface or several, using any Web services-enabled platform -- whether or not that platform is WSFL or even WSDL capable. In other words, you really don't have to know anything about WSFL to assume the role of a WSFL service provider. What you do have to know is how to take a Web services interface definition and turn it into an actual Web service implementation, and that's the process that I intend to explain here.

Overview of the process

Every activity within a WSFL flow model is implemented in the form of a Web service offered by a Web service provider and fulfilling one of the defined roles within the process. Each service provider is naturally expected to properly fulfill the requirements of implementing the Web service, or set of Web services that actually execute that activity. Each activity within a WSFL flow model is linked to an actual Web service implementation using information contained in the WSFL global model (see Figure 1). Each Web service implementation points to a WSDL-based description of that implementation which is, in turn, linked to the WSDL-based interface description that describes all of the actual messages and operations that can be sent to and from the Web service.

Figure 1: The flow model for the travel reservations business process



The global model that corresponds with the above graphical representation of a business process flow model is shown in Listing 1.

Listing 1: The global model for the travel reservations business process

```
<globalModel name="bookTripNTickets" serviceproviderType="agentType">
  <!-- Defines the identity, location and implementation of the service
        provider fulfilling the role of the travel agent -->
  <serviceProvider name="travelAgent" serviceProviderType="tio:agentType">
    <export>
      <source portType="tio:tripHandler" operation="sendItinerary"/>
      <target portType="tio:tripNTicketHandler" operation="sendItinerary"/>
    </export>
    <export>
      <source portType="tio:tripHandler" operation="receiveTripOrder"/>
      <target portType="tio:tripNTicketHandler" operation="receiveTripOrder"/>
    </export>
    <locator type="..." />
  </serviceProvider>

  <!-- Defines the identity, location and implementation of the service provider
        fulfilling the role of the airline -->
  <serviceProvider name="airline" serviceProviderType="tio:airlineType">
    <export>
      <source portType="tio:ticketDelivery" operation="sendETicket"/>
      <target portType="tio:tripNTicketHandler" operation="sendETickets"/>
    </export>
  </serviceProvider>

  <!-- A plugLink links the output of an operation of one service provider to
       the input of an operation for another service provider -->
  <plugLink>
    <source serviceProvider="airline" portType="tio:ticketHandler"
operation="sendConfirmation"/>
    <target serviceProvider="travelAgent" portType="tio:tripHandler"
operation="waitForConfirmation"/>
  </plugLink>
  <plugLink>
    <source serviceProvider="travelAgent" portType="tio:tripHandler"
operation="requestTicketOrder"/>
    <target serviceProvider="airline" portType="tio:ticketHandler"
operation="receiveTicketOrder"/>
  </plugLink>
</globalModel>
```

In order to fulfill the role of a service provider within a WSFL-defined process, you must first have access to the WSDL document that defines the fundamental data types, messages, and port types that you will need your to implement in Web service. You must then design and create the application code that implements each of the operations defined within the required port types and deploy that code on a Web services platform (such as IBM WebSphere Application Server 4.0, Apache SOAP, etc).

Becoming part of the process
The good news is, if you want to become a part of the business process, all you really have to know how to do is create a Web service. Once that is done, you merely need to get plugged into the WSFL global model that links your Web service implementation to one of the roles defined within the flow model. This link can be made statically (the global model explicitly identifies your Web service as the service provider for a given role) or dynamically (the global model specifies a set of discovery rules for selecting a service provider and yours gets selected).

There are plenty of resources available that detail how to create a Web service -- I'm somewhat partial to the tutorial I wrote for the Web Services ToolKit (see Resources)-- so I won't go into any those details here. Instead, what I've offered below are a few examples that illustrate how a Web service implementation is plugged into the global model.

It all starts with you, the developer, providing a WSDL description of the Web service that you have implemented. This description provides all of the information necessary for allowing client applications to invoke your Web service. In the case of a WSFL workflow, the "clients" will be other Web services that implement the various activities. How your Web services invoke other services is dependent on the way that your flow model defines the data and control links. Once the WSDL is created, it needs to be deployed to some network accessible location, preferably on your Web server. You may then optionally publish the WSDL-defined Web service to a UDDI registry.

For now, just assume that this step is done. You have a WSDL document named `Agent.wsdl` that lives at `http://acme.com/services/agent.wsdl`. The target namespace for that WSDL document is `urn:Agent_Service_Implementation` and the name of the Web service is *AgentService*. The intent of this service is to fulfill the role of the travel agent in the business process outlined above.

As defined by the WSFL specification, there are four different ways that your Web services can be located: statically, locally, via UDDI, or dynamically while the business process in being executed.

With a static location, it is as simple as merely adding a direct link to your WSDL service definition to the WSFL global model. This will tell the WSFL workflow engine exactly where your Web service lives without any decisions being made (see Listing 2).

Listing 2: A static location for a service

```
<serviceProvider name="travelAgent" type="tio:agentType">
    <!-- the service attribute is a QName that links to
        the service definition within a WSDL document -->
    <locator type="static" service="abc:agent"/>
</serviceProvider>
```

"Local" services are Web services that are not necessarily provided through a Web service interface. Rather, they take the form of applications or Java classes that are local to the workflow engine processing the request. These applications can still be described using WSDL documents, but the way that you do it is a bit different. The WSFL specification can provide more detail about local services (see Listing 3).

Listing 3: A "local" location for the service

```
<locator type="local">
    <!-- whatever information is needed to locate the local service.  This is
        up to the WSFL implementation -->
</locator>
```

The third way in which your Web service may be found is through the use of a UDDI query. Essentially, the global model defines a UDDI `find_service` operation intending to search the UDDI registry and retrieve a list of candidate Web services that fit the bill. Using a *selection policy*, the global model decides on which Web service, amongst those returned in the search results, will be used to fulfill the role within the business process. Valid selection policies include selecting the first service in the list, selecting a service at random from the list, or user some user-defined algorithm. The global model may also indicate when the UDDI query is executed. The choices are to run the query at development time, in which case the UDDI query is replaced with a static locator when the WSFL model is deployed to the production environment, or to run the query at runtime where the UDDI query is executed the first time that the flow model is invoked (see Listing 4).

Listing 4: A UDDI-based location for the service

```
<locator type="uddi" bindTime="startup" selectionPolicy="first">
    <uddi-api:find_service businessKey="uuid_key"
        generic="1.0" xmlns:uddi-api="urn:uddi-org:api">
        ...
    </uddi-api:find_service>
</locator>
```

The use of UDDI brings a great deal of flexibility and power to WSFL, allowing multiple service providers to compete for the right to fulfill a role within a business process. However, it is the *mobility locator* mechanism that adds the highest level of flexibility to WSFL. The mobility locator allows the WSFL workflow engine to select a service provider based completely on a set of rules applied while the process is being invoked. For example, if a purchase order is submitted that is over $10,000, the workflow engine may select a different Web service implementation than it would if the purchase order was for less than $10,000.

The mobility locator specifies where in the message that is used to invoke the Web service to look for conditions that must be met in order to select which Web service implementation will be actually invoked (see Listing 5).

Listing 5: A mobility location for the service

```
<locator type="mobility" activity="getFlight"
      message="flightInfo" messagePart="airline" dataField="providerInfo">
     <!-- rules for applying the mobility locator, these
        are determined by the WSFL implementation -->
</locator>
```

Summary

WSFL's ability to allow any Web service provider to implement any of the activities defined in the business process is perhaps one of its most powerful and useful features. The ability to dynamically locate, and bind to providers based on a user-defined set of rules adds a new dimension to conducting business on the Web that did not exist prior -- dynamic federation and integration of loosely coupled application components. In the next installment of this column, I'll introduce you to another cool feature of WSFL: the ability to recursively compose new business processes from existing business processes.

Resources

- Participate in the discussion forum on this article by clicking **Discuss** at the top or bottom of the article.
- Be sure to check out the three previous installments of this column: Part 3, Part 4, and Part 5.

- Learn how to implement Web services with the Web Services Toolkit
- Read the WSFL specification.
- Check out the business process modeling resources catalogued at Google.
- Learn about SOAP, WSDL, and UDDI.
- Learn about the IBM MQSeries and its workflow components.

About the author

James Snell is both an author and a developer and is one of the newest members of the IBM Web Services development team. He comes to IBM with an extensive background in custom enterprise application development and business-to-business integration, and a passion for the cutting edge of Web technology. You can reach him at jasnell@us.ibm.com.

Discuss e-mail it!

What do you think of this article?

○ Killer! (5) ○ Good stuff (4) ○ So-so; not bad (3) ○ Needs work (2) ○ Lame! (1)

Comments? Care to share your comments? Use the forum!

Submit feedback

About IBM | Privacy | Legal | Contact