# BTSHOL14: Working with Workflow and AppFabric

Overview

So far we have only worked with BizTalk. In this lab we’ll work with related technologies such as Windows Workflow Foundation and Windows AppFabric.

Objectives

After completing this lab, you will be able to:

* Create a Workflow using Visual Studio 2010.
* Expose your Workflow as a service
* Consume other services
* Deploy and manage your service using AppFabric.

Scenario

You work for a travel booking company, who receives flight requests, and correlates these to airline companies they have partnerships with.

There have been requests for booking hotels at the same time as booking the flights. Your job is to update the service to handle requests for hotel bookings along with the already existing flight bookings.

Estimated time to complete this lab: 60 minutes

User Name: **Administrator**

Password: **pass@word1**

Prepare the environment

Before you start you need to make a modification to the Internet Information Services.

1. Open the IIS Manager by clicking *Start->Administrative Tools ->* *Internet Information Services (IIS) Manager*
2. Expand the sites, and disable the *SharePoint -80* site by right-clicking the site and select *Manage Web Site ->Stop.*
3. Enable the Default site by right-clicking the *Default Web Site* and select *Manage Web Site ->Start.*

Exercise 1

Familiarize yourself with the solution

As you haven’t yet worked with Workflow Foundation, it’s important to take your time to familiarize with the developing environment. Much of the artifacts are the same as when working with BizTalk, but the tooling is a bit different.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Open the solution, and examine the contracts | 1. Open the following solution: C:\Labs\Lab 14\Start\ Lab14.sln 2. In the Solution explorer, open TravelContracts.cs. 3. This class file includes the request (*TravelRequest*) and the response (*TravelResponse*) messages of your service. There is also a *FlightResponse* and a *HotelResponse,* which both are used in the *TravelResponse.*   These **DataContracts** are equivalent to the schemas you’ve created when developing BizTalk solutions. |
| 1. Examine the Workflow | 1. In the Solution explorer, open TravelService.xamlx. 2. In the workflow designer, mark the *Travel Booking Service* scope*.* At the bottom left corner of the designer, click *Variables.*   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML5cedd9.PNG   1. Examine the variables. The *travelRequest* and the *travelResponse* are the only “Global variables” of the WF.   WF does not distinguish variables from messages as we do when working with BizTalk Orchestrations.   1. Mark the *Call Flight Service* and examine the variables.   Just as when working with BizTalk Orchestrations, it’s a good practice to scope the variables. This will cause there to be less data to be serialized when the workflow persists. Variables which is no longer in scope, will not get persisted.   1. Double-click the *Call Flight Service.* This action will drill down into the selected activity. You can go back using the breadcrumb navigation on the top left corner.   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML699a8b.PNG |
| 1. Test the solution | 1. In the Solution Explorer, right-click the solution and select properties. 2. In the *Startup Project* tab, make sure the *Multiple startup projects* is checked, and that all projects has the Action set to *Start.* 3. Press F5 to start up the projects. Two console applications and WCF Test Client should start. If you get a warning about “The contract ‘IMetaDataExchange’”… just click *Ok.* 4. In the WCF Test Client, double-click the *MakeTravelRequest()* operation. 5. In the service pane on the right, set the value of the *TravelRequest* to *TravelService.TravelRequest:*   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML7959f4.PNG   1. Expand the TravelRequest node and fill out the travel booking request, and press *Invoke.* 2. Analyze the response, and close the WCF Test Client tool. |

Exercise 2

Update the workflow

There have been requests for booking hotels at the same time as booking the flights. Your job is to update the service to handle requests for hotel bookings along with the already existing flight bookings.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Add the Hotel Booking Activity. | 1. Browse to C:\Labs\Lab 14\Start\HotelBookingService\bin\Debug, and double click the HotelBookingService.exe to start up the back-end hotel booking service which you’ll call from the WF. 2. In Visual Studio 2010, under the *Travel Service* project, right-click *Service References* and then click *Add Service Reference.* 3. In the *Add Service Reference* dialog, set the *Address* to <http://localhost:8080/HotelBookingService> and click the *Go* button. 4. Set the namespace to *HotelBookingServiceReference*, and click Ok. 5. Build the TravelService project, and open the Tool box on the left side of the Workflow designer.   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTMLb4d867.PNG  By adding a service reference, Visual Studio 2010 automatically creates an Activity for you. You’ll use this activity soon. |
| 1. Change the workflow to call the Hotel Booking Service. | 1. In the Tool box, expand the *Control Flow* activity group, and drag a *Parallel* activity and drop it just above the *Call Flight Service* activity in the workflow designer.   You’ll change the workflow to call both the Flight and the Hotel booking service in parallel.   1. Set the *DisplayName* to *Call Booking Services.* 2. Drag and drop the *Call Flight Service* activity into the *Call Booking Services*. 3. From the tool box, drag a *Sequence* activity to the right of the *Call Flight Service* activity. 4. Set the *DisplayName* to *Call Hotel Services.* 5. Drag the *SendHotelBookingRequest* from the tool box to the *Call Hotel Services* activity.      1. Mark the *Call Hotel Service* and click variables in the lower left corner. 2. Create a new variable and set the name to *hotelRequest.* Click the variable type (String), and select *Browse for Types…* 3. Expand the *TravelService.HotelBookingServiceReference* and select *HotelBookingRequest.* 4. Create another variable with the name *hotelResponse* and set the type to *HotelBookingResponse.* 5. Select the *SendHotelBookingService* activity. In the properties window, set the *hotelBookingRequest* to *hotelRequest*, and the *SendHotelBookingRequestResult* to *hotelResponse.* |
| 1. Create transformations | 1. In the tool box, expand the *BizTalk* activity category, and drag a *Mapper* activity above the *SendHotelBookingRequest* activity. 2. In the *Select Types* dialog, select *Browse for Types…* in the *InputDataContractType.* Select the *TravelService.TravleRequest.* 3. Repeat the step for the OutputDataContractType, and set it to *TravelService.HotelBookingServiceReference.HotelBookingRequest.* 4. Click the new mapper activity, and set the *Input* parameter to *travelRequest* in the Properties. Set the *Output* parameter to *hotelRequest.*   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTMLa07909.PNG   1. Click the *Edit…* button on the mapper activity. Set the name of the map to *TravelReq\_to\_HotelReq* and click *Ok* to create and open the new map. 2. Pair the schema nodes according to the table below:  |  |  | | --- | --- | | Source | Destination | | CreditCardNumber | CreditCardNumber | | Customer | Customer | | DepartureDate | FromDate | | Notes | Notes | | NumberOfNights | NumberOfNights |  1. Save and close the mapper. 2. Drag another Mapper activity below the *SendHotelBookingRequest* activity. Set the types according to the table below:  |  |  | | --- | --- | | InputDataContractType | TravelService.HotelBookingServiceReference.HotelBookingResponse | | OutDataContractType | TravelService.HotelResponse |  1. In the Properties for the new mapper activity, set the *Input* value to hotelResponse, and the *Output*value to *travelResponse.HotelResponse.*   In difference from BizTalk, you can append an already instantiated variable/message.   1. Click the *Edit…* button on the mapper activity. Set the name of the map to *HotelResp\_to\_TravelResp* and click *Ok* to create and open the new map. 2. As the two schemas share the same structure, drag the *HotelBookingResponse* node on the left, to the *HotelResponse* on the right. In the context menu chose *Link by Name.* 3. Save and close the mapper. 4. In the *Assign* activity below the *Call Booking Services* activity, update the right side *Value* expression to:   travelResponse.FlightResponse.AmountCharged + travelResponse.HotelResponse.AmountCharged  This will aggregate the charged amount from each of the booking services to the total amount of the travel request.    C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML16138d2.PNG |
| 1. Test the solution | 1. If you have the Hotel Booking Service Console open, close it. 2. Build and run the solution using F5. 3. If you get a warning about “The contract ‘IMetaDataExchange’”… just click *Ok.* 4. Complete the booking using the WCF Test Client, and reflect upon the result. The response should have both Flight- and Hotel responses, and the total amount charged. |

Exercise 3

Deploy the solution to Windows Server AppFabric

So far you have only tested your solution hosted in Visual Studio 2010. As you’re done with your workflow you now need to deploy it to Windows Server AppFabric. AppFabric leverages the existing IIS capabilities, why deploying a workflow is no different from deploying a web site. Basically, there are two different ways to deploy a workflow solution:

1. Publish from Visual Studio 2010
2. Build a deployment package in Visual Studio 2010, and deploy it using either IIS Manager or a created bat file.

Even though publishing the solution from Visual Studio is convenient, it is seldom the preferred choice in real life. This is because the production environments usually don’t have Visual Studio installed, nor is the environment accessible from any developer environment where Visual Studio is installed.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Create a deployment package using Visual Studio 2010. | 1. In the Solution Explorer, right-click the *TravelService* project and select *Build Deployment Package*.   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML2a37e2.PNG   1. After the package was successfully built, right-click the project again, and select *Package/Publish Settings*. If we were to specify the output location, we would have changed the *Location where package will be created* setting. |
| 1. Import the deployment package to the web server. | 1. Open the IIS Manager by clicking *Start->Administrative Tools ->* *Internet Information Services (IIS) Manager.* 2. In the IIS Manager, expand the web sites in the *Connections* pane to the left. Right-click the *Default Web Site*, and select *Deploy->Import Application.*   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML35757e.PNG   1. In the *Import Application Package* dialog, click the *Browse* button and navigate to *C:\Labs\Lab 14\Start\TravelService\obj\Debug\Package\TravelService.zip.* Click Open. 2. Click *Next* and *Next* again to set the name of the *Application.* Set the name to *TravelService.*   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML3b9cfb.PNG   1. Click *Next* and *Finish.* 2. Right-click the *TravelService* Application. Click *ManageApplication ->Advanced Settings.* 3. In the Advanced Settings dialog, set the *Application Pool* to ***AppServerAppPool.*** 4. Your package has been deployed, configured and is ready to use! |
| 1. Test the deployed Workflow service. | 1. In the C:\Labs\Lab 14\Start folder, start the back-end services using the shortcuts. 2. In Visual Studio, under the tools menu, click the *WcfTestClient.* 3. Right-click the *My Service Projects* node and select *Add Service.* 4. Set the endpoint address to: <http://localhost/TravelService/TravelService.xamlx> 5. Click *Ok*, and disregard the warning dialog. 6. Continue filling in the travel information as you have done in previous tests, and submit the travel request. 7. Shut down one of the back-end services (HotelBookingService or the FlightBookingService), and submit another booking. This should fail and cause an exception. |
| 1. Monitor your service | 1. In the IIS Manager, select the TravelService Application. Then double-click the AppFabric Dashboard.   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML81620f.PNG   1. In the *WF Instance History* click the *Activations* link to see all instances in the last 24 hours. 2. There should be at least one failed instance with status *Aborted.* Right-click one of these instances, and select *View Tracked Events.* Examine the report. 3. Right-click the *TravelService* Application, and select *Manage WCF and WF Services->Configure.* 4. In the *Configure WCF and WF for Application* dialog, select *Monitoring* in the left pane. Raise the level to ***Troubleshooting.*** 5. Submit one more message using the WCF Test Client. You might need to check the *Start a new proxy* check box. 6. Go back to the AppFabric Dashboard, and select *Failures* from the WF Instance History.   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML8b6b52.PNG   1. It might take a few seconds for the new message to arrive. Keep refreshing until it turns up, then right-click and select *View Tracked Events.* 2. Examine the result. |