Microsoft

Developing Intelligent Apps

Lab 3 – Building a Bot

By Gerry O'Brien

Overview

In this lab you will develop a relatively simple Bot that makes use of FormFlow. The Bot will be a sandwich order bot and will demonstrate how to create a Bot using the Bot Framework, the Bot Builder FormFlow library, and the Bot Emulator for testing your bot.

What You'll Need

To complete this lab, you will need the following:

- Microsoft Visual Studio 2015 Community Edition
- Internet access to install necessary libraries

Get the Bot Emulator

In this exercise you will download and install the Bot Emulator. You will use this emulator for testing your Bot after you have built it.

1. Navigate to the Bot Framework web site. https://dev.botframework.com/bots
2. You will be requested to login with a Microsoft Account (MSA), log in with your MSA
3. Click the Documentation tab on the top nav bar after logging in.
4. On the left menu, select Bot Framework Downloads.
5. This lab provides instructions for use with the Windows emulator so download by selecting that option.
6. If you elect to use the Console emulator for Mac or Linux with Mono, ensure you click the link for instructions on how to use that emulator.
7. Once the emulator downloads, install it.

Get the Starter Project

In this exercise you will download a Visual Studio project that contains starter code for you to build your Bot with. The starter project saves you some time in creating this application because it already has a project structure in place with most of the components you will need. We will cover these components in the lab.

1. Connect to the GitHub repository for this course.
2. Locate and download the DAT211xLab3Starter.zip file.
3. Open the project in Visual Studio 2015.

Create the Sandwich Class

In this exercise you will create the Sandwich class that will be used as the basis for the sandwich Bot. You will be creating the class, some enumerations to hold the sandwich options, and some fields in the class.

Create the Class

You should have already completed the previous exercise to ensure that you have the starter project open in Visual Studio 2015.

1. Create a new class in the Microsoft.Bot.Sample.SimpleSandwichBot project and name it Sandwich
2. Ensure that Sandwich.cs is open in the editor.
3. Because we want to be able to use this class for communication with the Bot system, we need to be able to serialize it so decorate the class with the Serializable attribute.
4. Also, because we will use a field name of Sandwich in the code, rename the class to SandwichOrder. Leave the filename as Sandwich.cs, just update the class name in the code.

Create the Sandwich Options

You should have already the Sandwich.cs file opened. In this exercise, you will create your sandwich options using enumerations. For each sandwich option below, create a separate C# enum.

1. Create an option for sandwich type. Call the enum SandwichOptions. Populate it with sandwich types such as BLT, BlackForestHame, ColdCutCombo, etc. Make up sandwiches of your own choosing but try to have somewhere between 5 to 15 choices.
2. Create an option for sandwich length and call the enum LengthOptions.
3. Create an enum for BreadOptions and add various bread types such as Italian, Flatbread, WholeWheat etc.
4. Create an enum for CheeseOptions and add some cheese types to the enum.
5. Create a ToppingOptions enum and add sandwich toppings such as Avocado, GreenPeppers, Onion, etc.
6. Create a SauceOptions enum and add some sandwich sauce types such as Mayo, LightMayo, Mustard, Vinegar, Pepper, etc.
7. In the Sandwich class, create fields for each of these enums. For example, you would create a field for the SandwichOptions called Sandwich as shown here.

```
public SandwichOptions? Sandwich;
```

8. Create the remaining fields using the following names and the sample code in step 7:
    - Length
    - Bread

- Cheese
- Toppings
- Sauce

9. Save your work and leave Visual Studio open for the next exercise

Review the Required Libraries

In this exercise, you will review the libraries that are required for the Bot Builder, Bot Connector, and the Json handling.

1. With Visual Studio open, expand the References in the Microsoft.Bot.Sample.SimpleSandwichBot project, in Solution Explorer.
2. Note that the Microsoft.Bot.Builder and Microsoft.Bot.Connector libraries are included as references.
3. Microsoft.Bot.Builder is required for the FormFlow namespace and classes that will be used to construct the forms the user will interact with.
4. Microsoft.Bot.Connector is what will be used to connect the bot to communication services such as Skype, SMS, etc. We will be using the Bot Emulator to communicate with this bot in the lab and not Skype.
5. Newtonsoft.Json is another library that is required as the services use JSON as the data format for message exchange.
6. There are also a list of Azure libraries listed but we will not be covering these libraries in this lab. They provide the options for publishing and connecting to Azure with your bots.
7. Collapse the References folder and leave Visual Studio open for the next exercise.

Setup the Sandwich Class for FormFlow

Now that you have seen the libraries that were imported, you can add the necessary code in the Sandwich class to take advantage of the FormFlow library.

1. At the top of the Sandwich.cs file, add the using directive for the Microsoft.Bot.Builder.FormFlow namespace.
2. Now we can implement a BuildForm method that will create the sandwich order form flows for us. Paste the following code inside the SandwichOrder class, immediately following your field list:

```
public static IForm<SandwichOrder> BuildForm()
 {
     return new FormBuilder<SandwichOrder>()
             .Message("Welcome to the simple sandwich order bot!")
             .Build();
    }
```

3. IForm is an abstract class in the FormFlow library that takes a class as the type to create when a new form is generated.
4. If you would like, you can right-click on the FormBuilder class name and choose Go To Definition or Peek Definition to review the code that is in the FormBuilder class. This is what will generate the forms that the user will interact with.

If you really want to understand the FormBuilder class and how it generates the forms for your bot, place a breakpoint in the FormBuilder code while debugging your application. We do not explain the FormFlow code in this lab.

Test the Bot

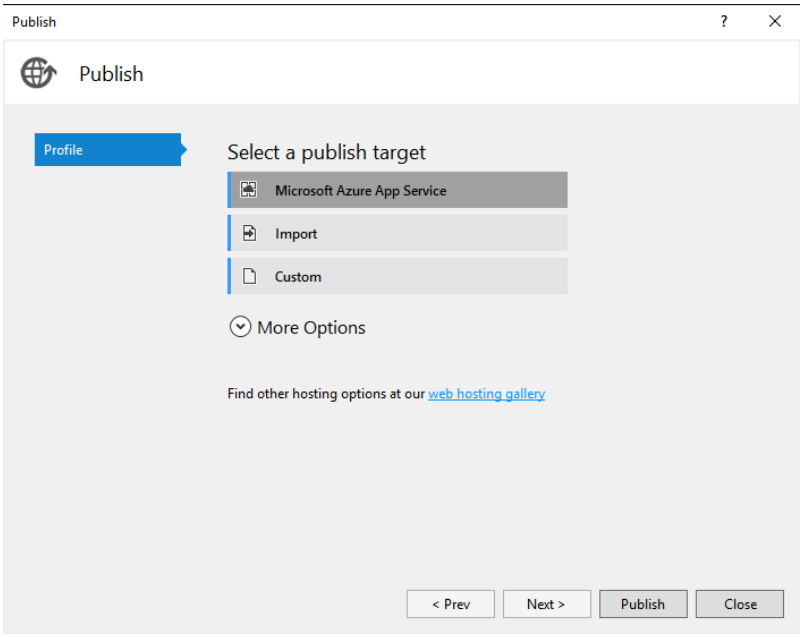
Run your application and have it open in your default browser.

1. Once the bot opens in the browser window, note the port number that is being used in the address bar.
2. Start the Bot Emulator
3. In the URL field, ensure that the address is http://localhost and then append the port number such as http://localhost:3977.
4. Append /api/messages to the URL for a full URL of http://localhost:3397/api/messages
5. In the chat entry pane at the bottom of the emulator, type in "make me a sandwich" and press enter.
6. Your bot should respond with a list of sandwich options from the enum you created in your code.
7. In the Bot Emulator, single select items in a form can be clicked with the mouse, or you can enter the number corresponding to the position of the item. For example, the second item in the list can be chosen by clicking on it or by entering 2 in the chat entry pane and pressing enter.
8. Select a length
9. Choose your bread
10. Choose the cheese
11. The toppings are a multi-select so you can no longer use the mouse. You can enter your selections as either space or comma separated values here. Choose a list of toppings and press enter.
12. Do the same for the sauce choice
13. Verify that the bot has determined all of your selections for your sandwich when it asks if this is your selection.
14. Close the emulator and stop the application.

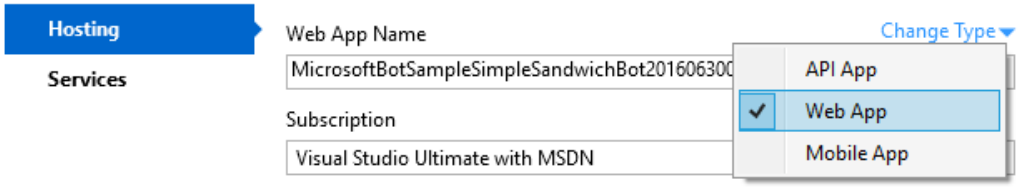
Publish your Bot

Now that you have a working bot, let's publish it in Azure. You will register your bot in the gallery in the next exercise.

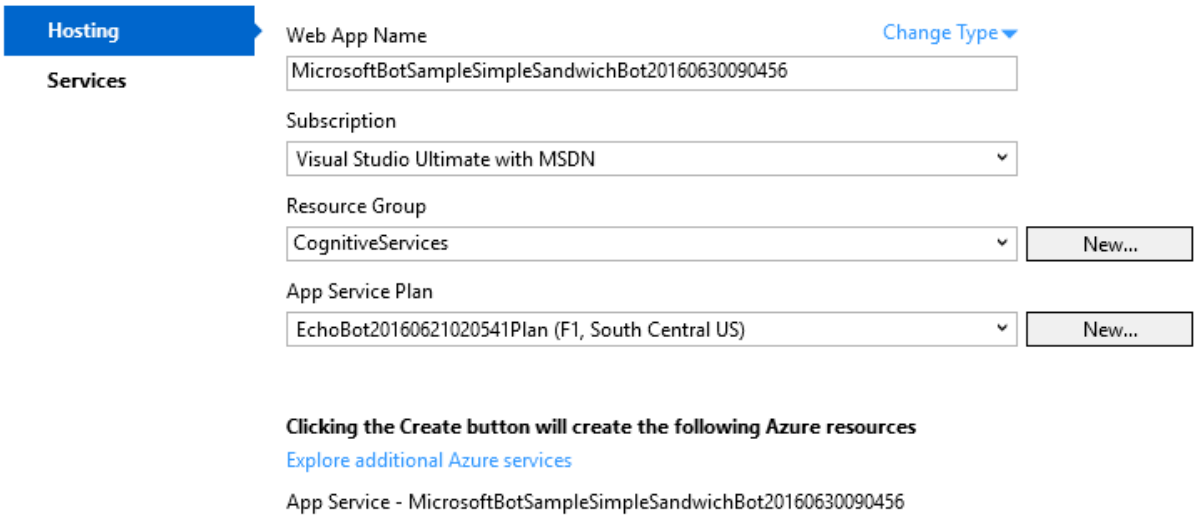
1. With your bot project open in Solution Explorer, right-click on the project folder and choose Publish.
2. In the Publish dialog, select the Microsoft Azure App Service option as shown here:



3. The App Service dialog is displayed where you will need to log in using your Azure account information.
4. Select your subscription from the drop down that you want to publish this Bot under
5. The next step in the Azure App Service publishing process is to create your App Service. Click on “New…” on the right side of the dialog to create the App Service.
6. The Create App Service dialog will be displayed, fill in the details as appropriate. Make sure to choose “Web App” from the Change Type drop down in the top right instead of “API App” (which is the default).



7. One final complexity on this dialog is the App Service Plan. This just lets you give a name to a combination of location and system size so you can re-use it on future deployments. Just put in any name, then choose the datacenter and size of deployment you want.



8. Once you hit okay on the App Service Plan, you'll have defined your App Service completely. Hit Create, and you'll be taken back to the Publish Web Wizard.
9. Now that you've returned to the Publish Web wizard copy the destination URL to the clipboard, you'll need it in a few moments. Hit "Validate Connection" to ensure the configuration is good, and if all goes well, click "Next".
10. By default your Bot will be published in a Release configuration. If you want to debug your Bot, change Configuration to Debug. Regardless, from here you'll hit "Publish" and your Bot will be published to Azure.
11. Before you can use you Bot, you will need to register it with the Microsoft Bot Framework. The next exercise will guide you through that process.

Register your Bot

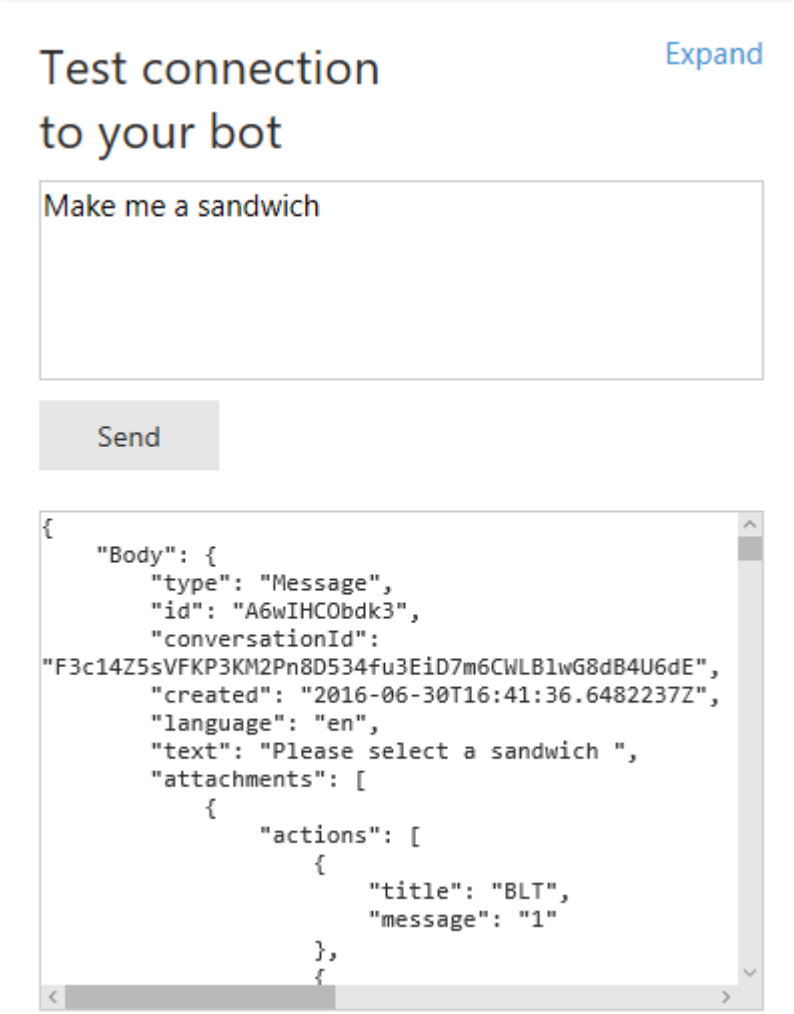
Now that you have published your bot to Azure, let's register it with the Bot Framework so you can access it. This registration process will generate the AppId and AppSecret needed for accessing the Bot.

1. Go to the Microsoft Bot Framework portal at https://www.botframework.com and sign in with your Microsoft Account.
2. Click the "Register a Bot" button and fill out the form. Many of the fields on this form can be changed later. Use the endpoint generated from your Azure deployment, and don't forget that when using the Bot Application template, you'll need to extend the URL you pasted in with the path to the endpoint at /API/Messages. You should also prefix your URL with HTTPS instead of HTTP; Azure will take care of providing HTTPS support on your bot. Save your changes by hitting "Create" at the bottom of the form.
3. Once your registration is created, Microsoft Bot Framework will have generated your AppId and AppSecrets. These are used to authenticate your Bot with the Microsoft Bot Framework
4. Click Show to display the Primary app secret value.
5. Now that the Bot is registered, you need to update the keys in the web.config file in your Visual Studio project. Change the following keys in the web.config file to match the ones

generated when you saved your registration, and you're ready to build. You need only the primary AppSecret, the secondary is used when you wish to regenerate your primary key without downtime. Clicking the "show" link will show the value, along with exposing the regenerate link if you ever need to change your AppSecret. Update your web.config, and re-publish your bot to Azure.

6. Back in the developer dashboard for your Bot there's a test chat window that you can use to interact with your Bot without further configuration, and verify that the Bot Framework can communicate with your Bot's web service.

Note that the first request after your Bot starts up can take 20-30s as Azure starts up the web service for the first time. Subsequent requests will be quick. This simple viewer will let you see the JSON object returned by your Bot.



NOTE: If you try to test your bot with the Bot Emulator, you may receive a 500 Internal Server Error message if your emulator is running on a computer that is behind a firewall. You will need to note the emulator port number and ensure that it is open on your firewall.

7. The last step is to add any channels that you wish to have interact with your Bot. This lab doesn't cover the channels as there are many channels available and each has distinct setup

and configuration steps. Choose a channel of your choice and follow the instructions found there to setup your channel.

Summary

In this lab you have:

- Downloaded and used the Bot Emulator
- Created a Bot that used FormFlow to generate a sandwich order interaction
- Tested the bot using the Bot Emulator on your local machine
- Published your bot to Microsoft Azure
- Registered your Bot with the Bot Framework in order to use channels to interact with your bot.

NOTE: It's important to note that the Bot Framework does provide a lot of functionality to your bots but it still relies on you to create the actual logic behind the intelligence of the bot. You can make use of LUIS for language understanding that can help you to create more sophisticated logic than just simply hard coding options in code, and you can also make use of the Microsoft Cognitive Services to evaluate and provide results on the text sent by the user. In this way, you can utilize these services to help with the logic of the intelligence in your bot.