

Visual Studio **LIVE!** | San Diego
EXPERT SOLUTIONS FOR .NET DEVELOPERS

W11 - Building Business Applications Using Bots

Michael Washington
Programmer
AiHelpWebsite.com
Level: Introductory/Intermediate

Code Again for the First Time!

Visual Studio 25 YEARS OF CODING INNOVATION

About Michael Washington



- Microsoft Reconnect MVP
- Microsoft Certified Professional



Agenda

- Background
 - What is the Microsoft Bot Framework V4?
 - How to get started with the Microsoft Bot Framework V4 (and how much it costs)
- Creating a Hello World! Bot
- Using Dialogs
- Using LUIS-Language Understanding Intelligent Service
- Using QnA Maker



What is the
Microsoft Bot
Framework V4?



The **Microsoft Bot Framework V4** allows you to create intelligent bots that interact naturally wherever your users are (text/SMS to Skype, Slack, Office 365 mail and other popular services).

In preview now, the **Bot Builder V4 Preview SDK** offers new features and is extensible with a pluggable middleware model.



How To Get Started



Requirements

- Visual Studio 2017 (or higher) with the following workloads:
 - ASP.NET and web development
 - Azure development
 - .NET Core cross-platform development
- Bot Builder V4 SDK Template for Visual Studio
- Bot Framework Emulator (download the latest version even if it is “Alpha”)
- A Microsoft Azure Subscription



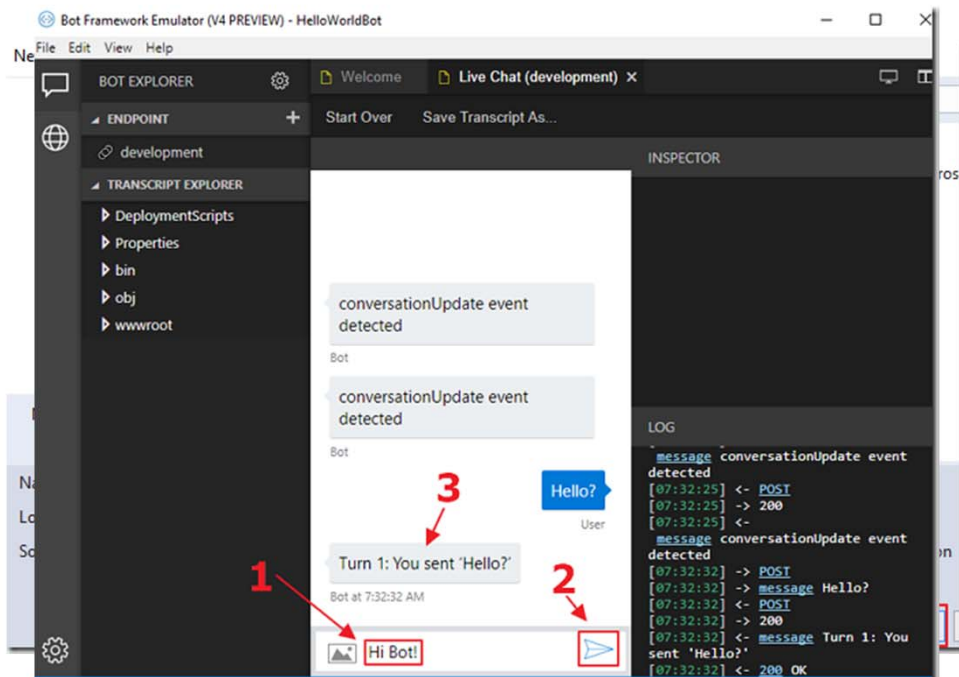
Pricing

Choose your pricing tier			
Browse the available plans			
Bot Service Premium Messages pricing includes messages sent/received via the Premium Channels. Learn more			
F0 Free	S1 Standard		
10K Premium Messages	1K Premium Msgs/Unit		
Bot Creation Tools	Bot Creation Tools		
Free Standard Channels	Free Standard Channels		
	99.9% Premium Messages SLA		
0.00 FREE	0.50 USD/1,000 MESSAGES (ESTIMATED)		

+Plus hosting for website and storage



Creating A Hello World! Bot



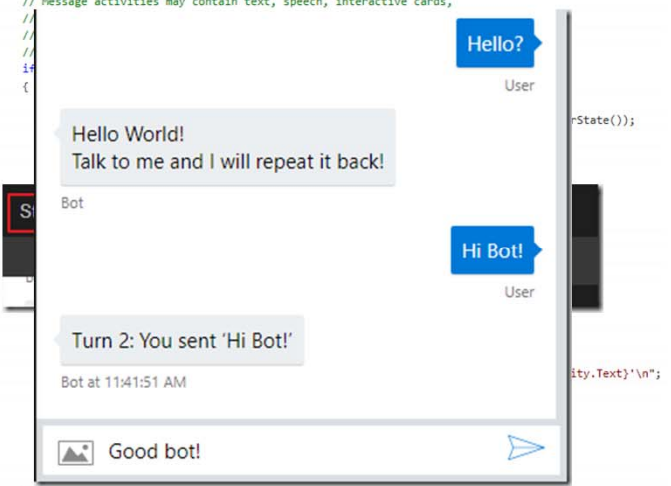
Demonstration



Update The Code



```
public async Task OnTurnAsync(ITurnContext turnContext, CancellationToken
    cancellationToken = default(CancellationToken))
{
    // Handle Message activity type, which is the main activity
    // type for shown within a conversational interface
    // Message activities may contain text, speech, interactive cards,
    //
    //
    //
    if (turnContext.Activity.Type == ActivityType.Message)
    {
        // User: Hello?
        // Bot: Hello World!
        //       Talk to me and I will repeat it back!
        // User: Hi Bot!
        // Bot: Turn 2: You sent 'Hi Bot!'
        //       Bot at 11:41:51 AM
        //       [Image] Good bot!
    }
    else
    {
        await turnContext.SendActivityAsync($"{turnContext.Activity.Type} event detected");
    }
}
```



Visual Studio 25 YEARS OF CODING INNOVATION

Demonstration

Deploy The Bot



Home > All resources > HelloWorldAzureBot - Settings
HelloWorldAzureBot - Settings

Add a bot

Bot name * Location *
Hello World (Production) C:\TEMP\HelloWorldBot\HelloWorldBot\BotConfigu Browse...

Endpoint URL *
https://HelloWorldBot.azurewebsites.net/api/messages

MSA app ID MSA app password
500e4ab0-c1fd-4e1b-a9e1-75f1

☐ Encrypt your keys

Cancel Connect

Bot Service pricing API key (User-Generated Application Insights API key)



Demonstration (Video)



Using Dialogs



A **dialog** encapsulates application logic much like a function does in a standard program. It allows you to perform a specific task, such as gathering the details of a user's profile, and then possibly reuse the code as needed. **Dialogs** can also be chained together in **DialogSets**.

The **Microsoft Bot Builder SDK** includes two built-in features to help you manage conversations using **dialogs**:

- DialogSets** - This is a collection of **Dialogs**. To use **dialogs**, you must first create a **dialog set** to add the **dialog** to. A **dialog** can contain only a single *waterfall step*, or it can consist of multiple *waterfall steps*.

- Prompts** - This provides the methods you can use to ask users for different types of information. For example, a text input, a multiple choice, or a date or number input. A prompt dialog uses at least two functions, one to prompt the user to input data, and another function to process and respond to the data.



Change the constructor of the class to the following:

In the **OnTurnAsync** method, change the code in the first "else" block to the following:

```
// Run the DialogSet - let the framework manage the current state of the dialog from the dialog stack
// the dialog stack
var dialogContext = Remove the "else" block.
var results = await dialogContext.ProcessAsync(input, cancellationToken);
// If the DialogTurnStatus is Empty, start a new dialog.
if (results.Status == DialogTurnStatus.Empty)
{
    await dialogContext.BeginDialogAsync("details", null, cancellationToken);
}

_dialogs.Add(new WaterfallDialog("details", WaterfallSteps));
_dialogs.Add(new TextPrompt("name"));
```

Bot Framework Emulator (V4 PREVIEW)

Manage User Secrets

The screenshot shows a Visual Studio code editor with a C# file. The code is for a bot's OnTurnAsync method. It uses the Bot Framework SDK to manage dialogs. A blue callout box points to the 'else' block where a new dialog is started. The code includes comments and uses DialogContext, DialogTurnStatus, and DialogSet classes. A 'Bot Framework Emulator (V4 PREVIEW)' window is visible in the background.

Demonstration



Adding User State



To this:

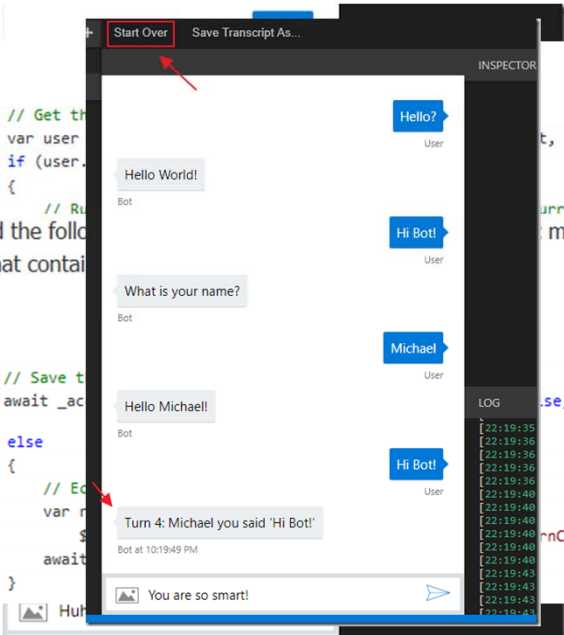
```
// Get the user profile
var userProfile = await _context.UserProfiles.FindAsync(userId);
if (userProfile == null)
{
    // Return a new user profile
    userProfile = new UserProfile();
    _context.UserProfiles.Add(userProfile);
    await _context.SaveChangesAsync();
}

// Save the user profile
await _context.SaveChangesAsync();

else
{
    // End the conversation
    var response = new Activity(ActivityType.Text, "You are so smart!");
    await _context.SaveChangesAsync();
}
```

Finally, we add the following **UserState** (that contains the current state of the dialog from the previous method) to persist the state:

```
private static UserState GetOrCreateUserStateFromConversation(
    ConversationState conversationState, CancellationToken cancellationToken)
{
    return conversationState.GetOrCreateUserStateFromConversation(cancellationToken);
}
```



The screenshot shows the Visual Studio Bot Framework interface. On the left, a chat window displays a conversation between a user and a bot. The user messages are: "Hello?", "Hi Bot!", "Michael", and "Hi Bot!". The bot messages are: "Hello World!", "What is your name?", "Hello Michael!", and "Turn 4: Michael you said 'Hi Bot!'". The chat window also shows a timestamp "Bot at 10:19:49 PM" and a status "You are so smart!". On the right, the Bot Framework Inspector is open, showing the current state of the dialog from the previous method. The inspector displays a list of messages with timestamps and the current state of the dialog. A red arrow points to the "Start Over" button in the top left corner of the chat window.

Visual Studio 25 YEARS OF CODING INNOVATION

Demonstration

Using LUIS

Language Understanding Intelligent Service



The Microsoft [Language Understanding Intelligent Service \(LUIS\)](#) allows you to create intelligent applications that allow your end-users to use natural language to perform actions.

The **Language Understanding Intelligent Service (LUIS)**, allows developers to create language understanding models that are specific to their [problem domain](#).

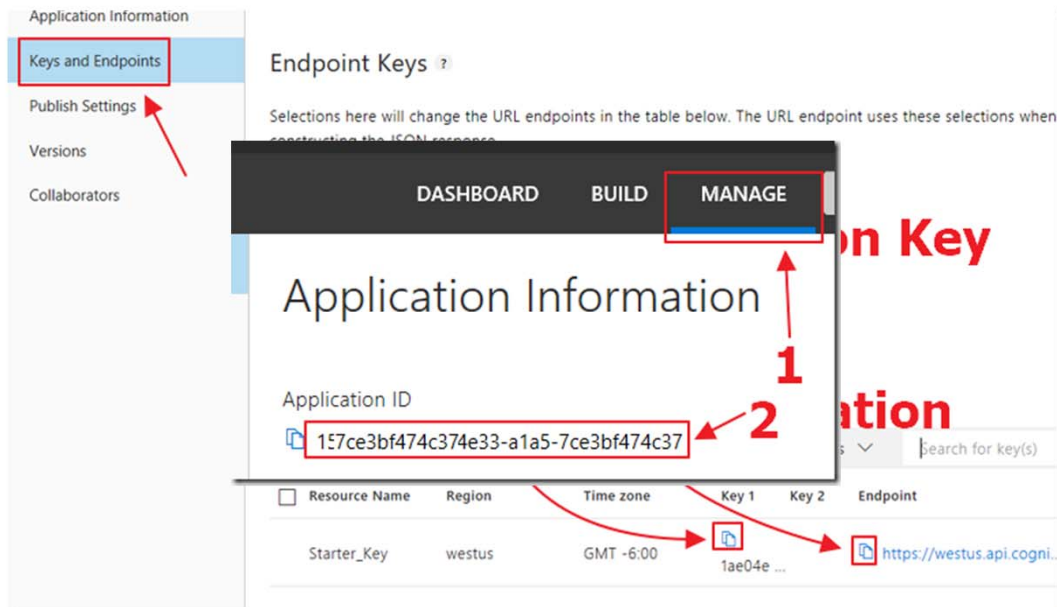
To use **LUIS**, developers log into the [LUIS website](#), enter a few **utterances** and their **labels**, and then deploy a model to an HTTP endpoint on [Azure](#).

The utterances sent to the endpoint are logged and labeled. The website allows the developer (or any application administrator) to train the application by identifying issues, which can be resolved by either adding more labels or by giving hints to the machine learner in the form of features.

A developer can create and deploy a model in minutes, and then maintain and train it as the usage of the application grows.



Create The LUIS Application



The screenshot shows the 'Keys and Endpoints' section of a LUIS application in the Azure Portal. A red box highlights the 'Keys and Endpoints' tab in the left sidebar. A red arrow points to the 'MANAGE' tab in the top navigation bar. Another red arrow points to the 'Application ID' field, which contains the value '157ce3bf474c374e33-a1a5-7ce3bf474c37'. A red box highlights this Application ID. A red arrow points to the 'Key 1' field in the table below, which contains the value '1ae04e...'. A red box highlights this key. A red arrow points to the 'Endpoint' field in the table below, which contains the value 'https://westus.api.cogni...'. A red box highlights this endpoint. The table has columns for 'Resource Name', 'Region', 'Time zone', 'Key 1', 'Key 2', and 'Endpoint'. The first row is labeled 'Starter_Key' and has the region 'westus' and time zone 'GMT -6:00'.

Resource Name	Region	Time zone	Key 1	Key 2	Endpoint
Starter_Key	westus	GMT -6:00	1ae04e ...		https://westus.api.cogni...



Update The Bot Code

A screenshot of a bot development interface. On the left, a code editor shows a C# snippet with a `public` keyword. The main area displays a chat conversation between a user and a bot. The user's messages are in blue bubbles, and the bot's responses are in grey bubbles. The chat history includes: "Hello?", "Hi Bot!", "Michael", "Can you help me?", and "Thanks!". The bot's responses are: "Hello World!", "What is your name?", "Hello Michael!", and "<here's some help>". A timestamp "Bot at 9:23:35 PM" is shown. On the right, a "LOG" window displays a detailed trace of the bot's execution, including timestamps, HTTP status codes (200), and the specific messages sent and received. The log entries show the bot's internal state and the flow of the conversation. The Visual Studio 25th Anniversary logo is in the bottom right corner.

Demonstration



Using QnA Maker



According to the [Microsoft QnA Maker website](#), the **QnA Maker** allows you to: *"Build, train and publish a simple question and answer bot based on FAQ URLs, structured documents, product manuals or editorial content in minutes"*.

This really speeds your **Bot** development because it leverages the abilities of [LUIS \(Language Understanding Intelligent Service\)](#) to understand human conversation, and combines it with a service that allows you to provide a list of questions and answers, or simply upload a product manual or provide a link to a website, and the questions and answers will be parsed for you automatically.

The end result is an incredible amount of functionality that you can add to your **Bot** easily.



Create The QnA
Maker Service



STEP 4 QnA Maker Michael

Success! Your service has been deployed.
What's next? **KnowledgeBaseId**

You can always find the deployment details in your service's settings.

Use the below HTTP request to build your bot. **Host** [Learn how.](#)

Sample HTTP request

```
POST /knowledgebases/63015717-4c5a084a1084a1084a1084a134a1/generateAnswer
Host: https://helloworldbotqna.azurewebsites.net/qnamaker
Authorization: EndpointKey aedc42ce-e28a-4a084a1084a1084a1
Content-Type: application/json
{"question": "<Your question>"}
```

EndpointKey

+ Add f

On which languages does Microsoft provide support? nine languages: English,

Visual Studio 25 YEARS OF CODING INNOVATION

Demonstration
Video

Update The Bot Code



Add the following code to the Bot class (this adds code to the Bot class):

Inside the method:

```
// Give QnA answer
var answer = await _context.QnARepository.GetQnAAsync(question);

// If no QnA found, return a default answer
if (answer == null)
{
    answer = "No QnA found. Please try again later.";
}
else
{
    // Echo back the QnA answer
    answer = "The answer to your question is: " + answer;
}

// Echo back the QnA answer
var response = $"The answer to your question is: {answer}";
await _context.QnARepository.UpdateQnAAsync(question, response);
return response;
```

What Azure support do you have?

Access to Azure technical support requires a [paid support plan](#). However, some Microsoft programs could give you access to Azure technical support via Support Benefits even if you don't have a paid support plan. The following programs include Azure Support Benefits:

- [MSDN and BizSpark](#)
- [Microsoft Partner Network](#)
- [Signature Cloud Support](#)

Each of these programs will give you different support benefits, from response times to the number of requests that can be submitted. Please check the details of each program from the links above for more information.

There are also some programs like MSDN and BizSpark Plus that give you other Microsoft Azure Benefits on top of access to Technical Support, such as monthly usage credits, free services and discounted consumption rates. Check the details of these programs for more information.

Thanks!

INSPECTOR - JSON

```
{
  "channelData": {
    "clientActivityId": "1538716819071.7477545620508967.8"
  },
  "from": {
    "id": "default-user",
    "name": "User"
  },
  "id": "6b07bb70-c85e-11e8-861e-1b441c8a411d",
  "locale": "en-US",
  "text": "What Azure support do you have?",
  "textFormat": "plain",
  "timestamp": "2018-10-05T05:20:48",
  "type": "message"
}
```

LOG

```
[22:20:37] <- 200 OK
[22:20:37] -> POST
[22:20:37] -> message Hello Bot!
[22:20:37] -> POST
[22:20:37] -> POST
[22:20:37] -> trace Luis Trace
[22:20:37] -> POST
[22:20:37] -> trace QnA Maker Trace
[22:20:37] -> POST
[22:20:37] -> 200
[22:20:37] -> message Turn 4: Michael you said 'Hello Bot!'
[22:20:37] -> 200 OK
[22:20:48] -> POST
[22:20:48] -> message What Azure support do you have?
[22:20:48] -> POST
[22:20:48] -> 200
[22:20:48] -> trace Luis Trace
[22:20:48] -> POST
[22:20:48] -> trace QnA Maker Trace
[22:20:48] -> POST
[22:20:48] -> 200
[22:20:48] -> message Access to Azure support requires a paid support plan. However, some Microsoft programs could give you access to Azure technical support via Support Benefits even if you don't have a paid support plan. The following programs include Azure Support Benefits:
[22:20:48] -> 200 OK
```

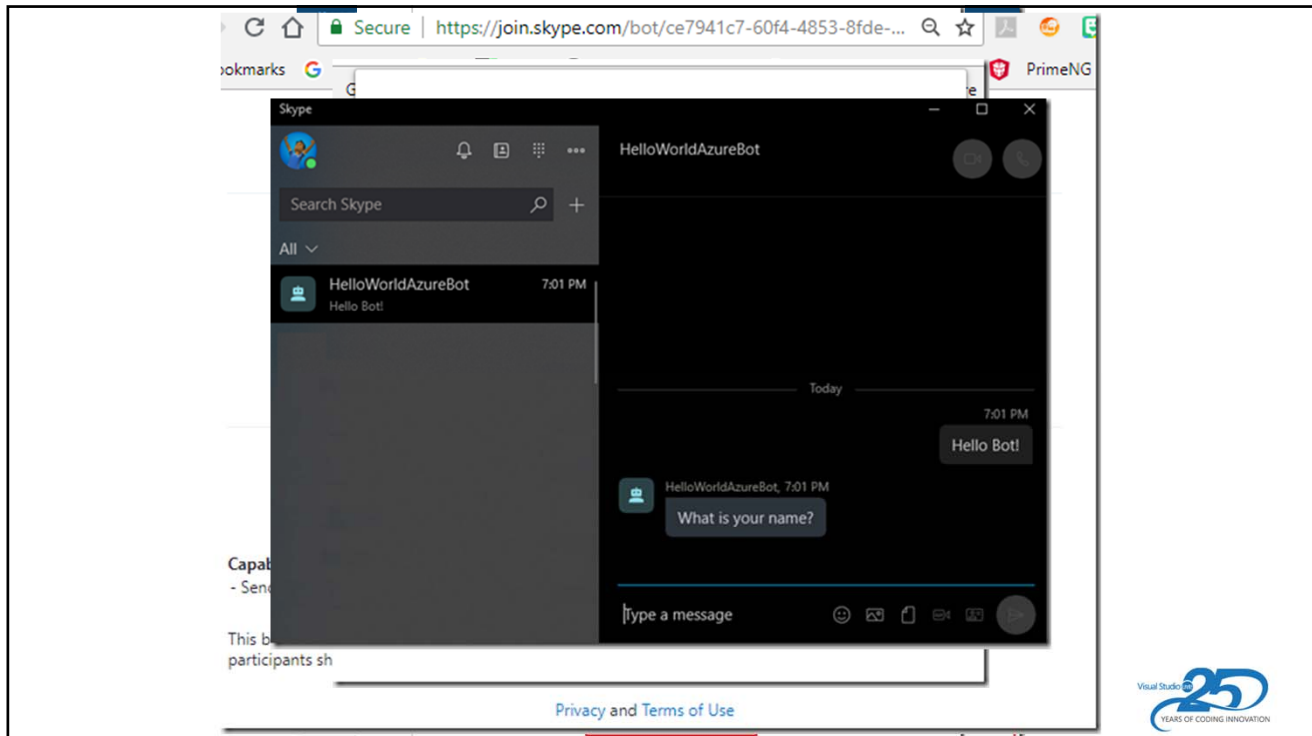


Demonstration



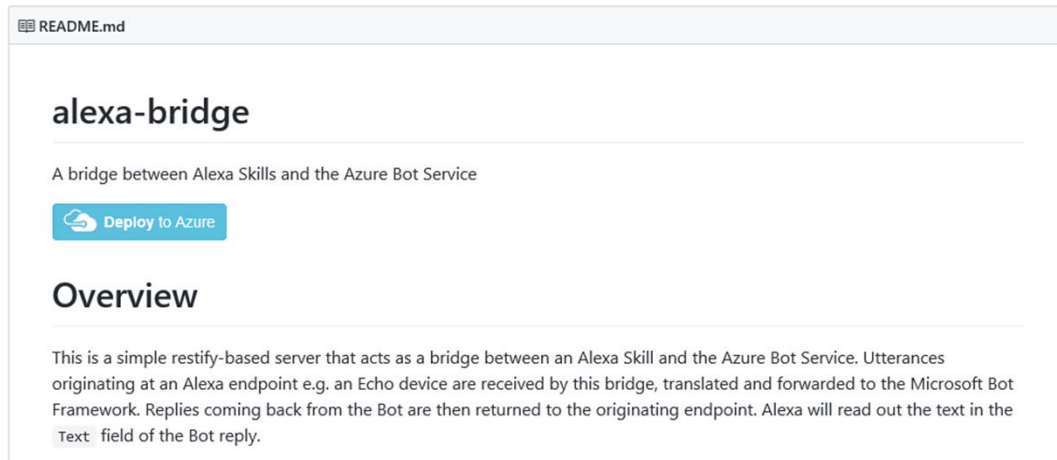
How to deploy to Skype, Cortana and Facebook





How to communicate
with users through
Amazon Alexa

<https://github.com/CatalystCode/alexa-bridge>



Resources

AI Help Website

<http://AIHelpWebsite.com>

ADefWebserver

<http://ADefWebserver.com>



Questions ?



Thank You!

