



BOT Work Shop

Eric Hullander



Microsoft Partner
Gold Data Analytics
Gold Cloud Platform



Tools You Will Need

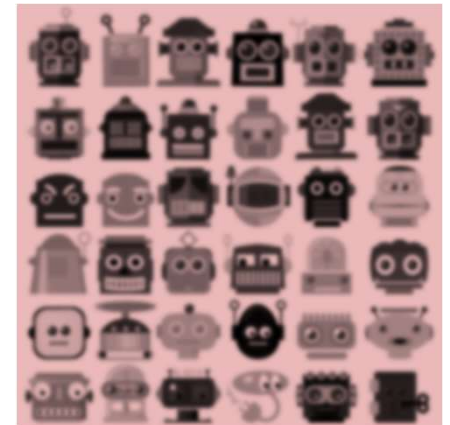
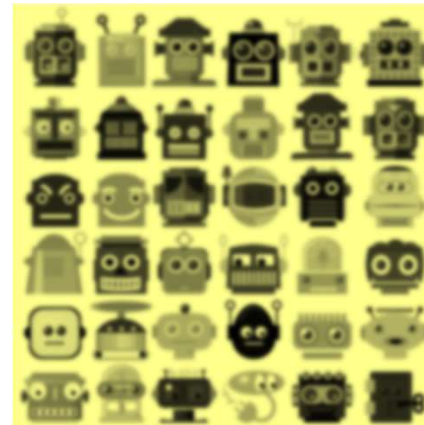
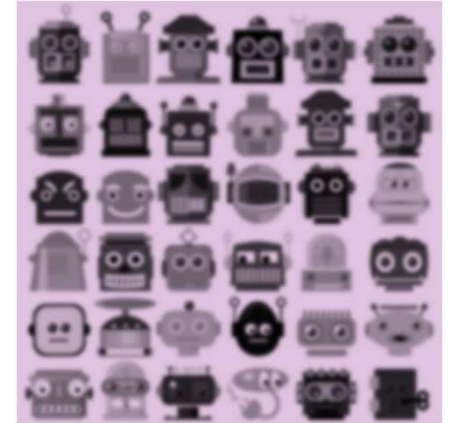
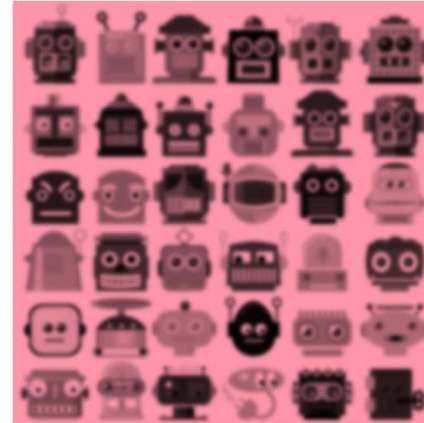
<https://docs.botframework.com/en-us/>

Required:

- Microsoft Account
- Visual Studio 2015 or higher
- Microsoft.Bot.Builder ("Manage NuGet Packages" in VS)
- Bot emulator
- Web Browser
- Skype, etc.

Optional:

- SQL Management Studio
- Excel
- Notepad
- PowerBI



Bot Framework

- Sign into Azure
- <https://dev.botframework.com/>
- Click Register a Bot
- Fill in
 - Name
 - Bot Handle
- Click Create Microsoft App ID and password
- Will leave “Messaging Endpoint” Blank

Bot Framework
My bots
Register a bot
Documentation
Bot Directory
Blog

Tell us about your bot

Bot profile

Icon
Upload custom icon
(300x300, jpeg only)

Name *
Bot handle *
Type in your Bot handle
Description *
Description

Configuration

Messaging endpoint:
https URL

Register your bot with Microsoft to generate a new App ID and password

Create Microsoft App ID and password

Paste your app ID below to continue:
Microsoft App ID from the Microsoft App registration portal

Admin

Owners:
Instrumentation key:
Instrumentation key (Azure App Insights key)

By clicking Register, you agree to the Privacy statement, Terms of use, and Code of conduct.

Register Cancel

Bot Framework

Copy the App ID to notepad for future use

Generate App ID and password

App name

EricHullandertest

App ID

c6851ae8-631c-4911-ae11-44a236edc3ad

Generate an app password to continue

☐ English (United States)

Bot Framework AppID and Password

- Copy the App ID to notepad for future use
- Click "Generate an app password..."
- Copy the Password to notepad for future use
- Click "OK"
- "Click Finish and go to Bot Framework"

App name: EricHullandertest

App ID:

App pw:

The image shows two screenshots of the Bot Framework App ID and Password generation interface. The top screenshot shows the 'Generate App ID and password' form with the App name 'EricHullandertest' and App ID 'c6851ae8-631c-4911-ae11-44a236edc3ad'. A button 'Generate an app password to continue' is visible. The bottom screenshot shows the 'New password generated' message with the password 'L4wqs7mXeAnz7OGcXWxFTm' and an 'Ok' button. Below this, the 'Generate App ID and password' form is shown again, but with the 'Finish and go back to Bot Framework' button visible instead of the 'Generate an app password to continue' button.

Generate App ID and password

App name
EricHullandertest

App ID
c6851ae8-631c-4911-ae11-44a236edc3ad

Generate an app password to continue

English (United States)

New password generated

Copy this password now, this is the only time when it will be displayed. Please store it securely and paste it into your bot configuration file.

L4wqs7mXeAnz7OGcXWxFTm

Ok

Generate App ID and password

App name
EricHullandertest

App ID
c6851ae8-631c-4911-ae11-44a236edc3ad

Password [Learn More](#)
L4w*****

Finish and go back to Bot Framework

English (United States)

Bot Framework Registration

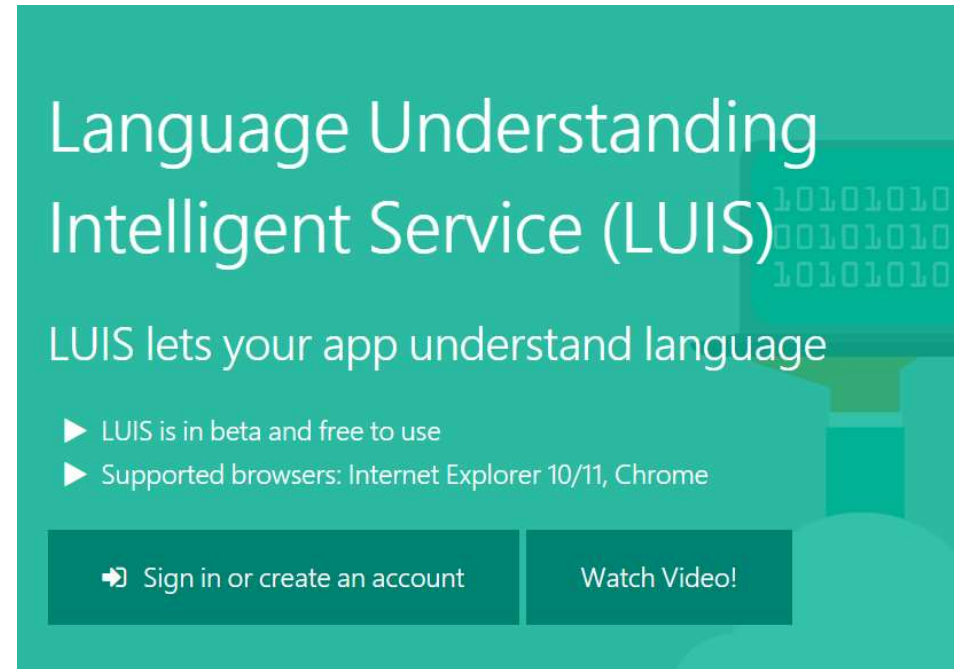
- Add the description if you haven't
- Leave endpoint blank for now
- Click if you agree to the T&Cs
- Click "Register"
- Click "OK" on "Bot created" menu
- You have just taken your first step into a much larger world...

Option: You may add an app icon now or later if you like...

The screenshot shows the Bot Framework Registration interface. The top navigation bar includes links for Bot Framework, My bots, Register a bot, Documentation, Bot Directory, and Blog. The main heading is 'Tell us about your bot'. The form is divided into three sections: Bot profile, Configuration, and Admin. The Bot profile section includes an icon upload area, Name, Bot handle, and Description fields. The Configuration section includes a Messaging endpoint field and a button to 'Manage Microsoft App ID and password'. The Admin section includes an Owners field and an Instrumentation key field. A 'Bot created' dialog box with an 'OK' button is overlaid on the right side of the form. At the bottom, there is a 'Register' button and a 'Cancel' button.

LUIS

- [Sign into Azure](#)
- <https://www.luis.ai/>
- Sign in with your Microsoft account



Language Understanding
Intelligent Service (LUIS)

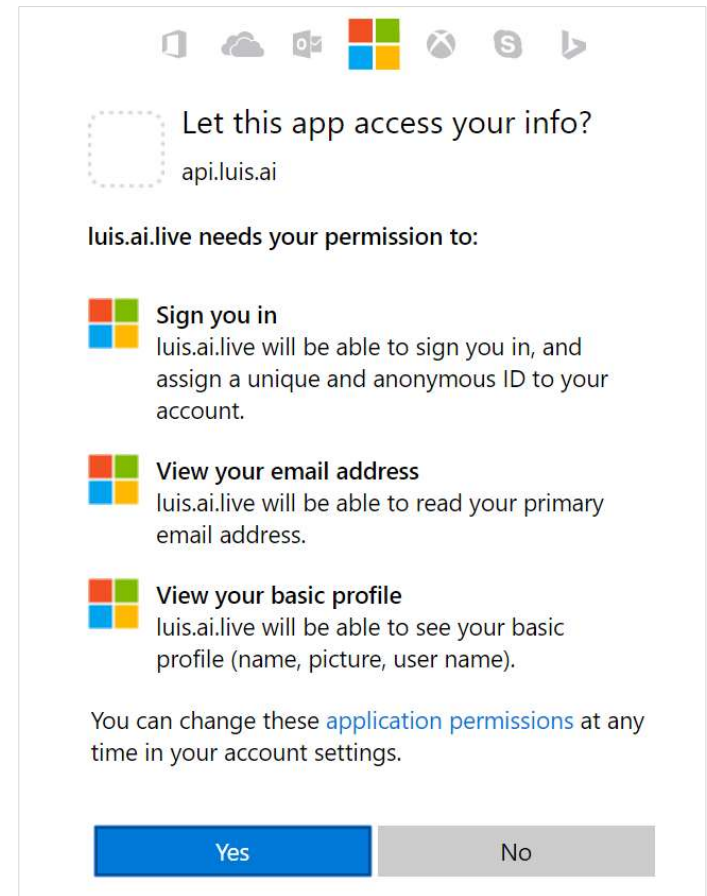
LUIS lets your app understand language

- LUIS is in beta and free to use
- Supported browsers: Internet Explorer 10/11, Chrome

[→ Sign in or create an account](#) [Watch Video!](#)

LUIS

- Sign into Azure
- <https://www.luis.ai/>
- Sign in with your Microsoft account
- "Yes"



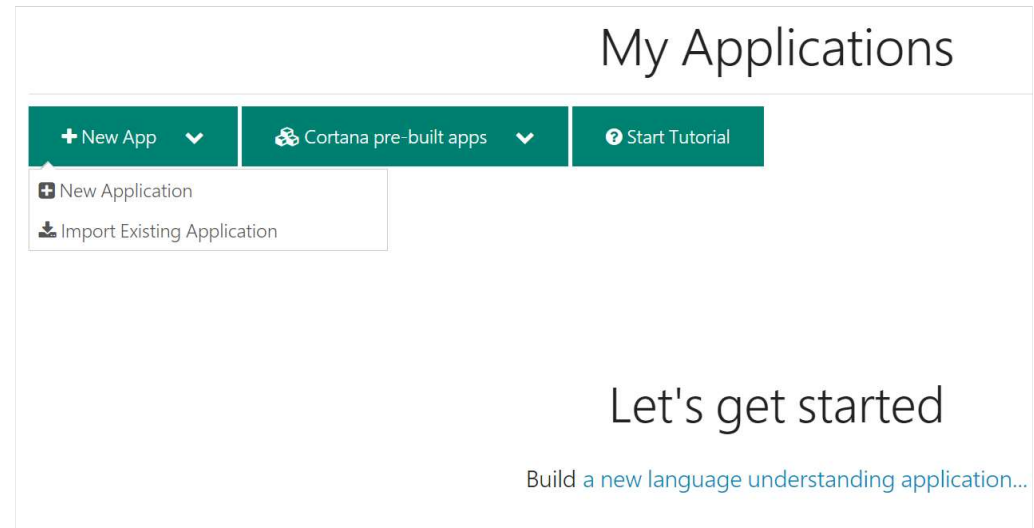
LUIS Building a LUIS Model

- New App
- Choose "Import..."

A LUIS Application (model) takes examples of many phrases (utterances) that mean the same thing and maps them to a single intent

A LUIS Application can have up to 40 unique intents

Given a new utterance, LUIS predicts which of the 40 intents is the best match



LUIS Building a LUIS Model

- New App
- Choose "Import..."

Further, entities (nouns, verbs, adjectives) can be inferred from within an utterance

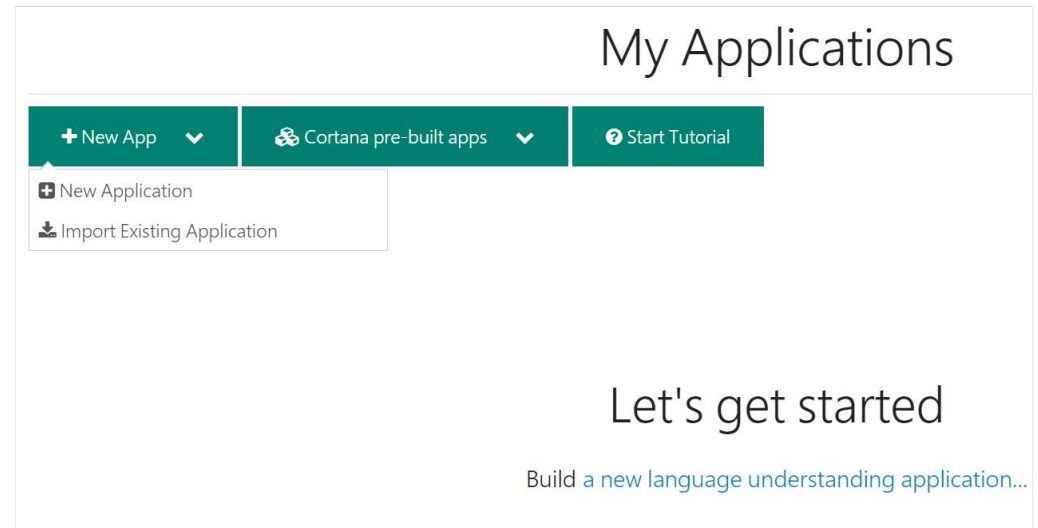
Example:

I want to buy a **house**.

I want to buy a **car**.

Might map to the intent "purchase"

The entity could be "house" or "car"

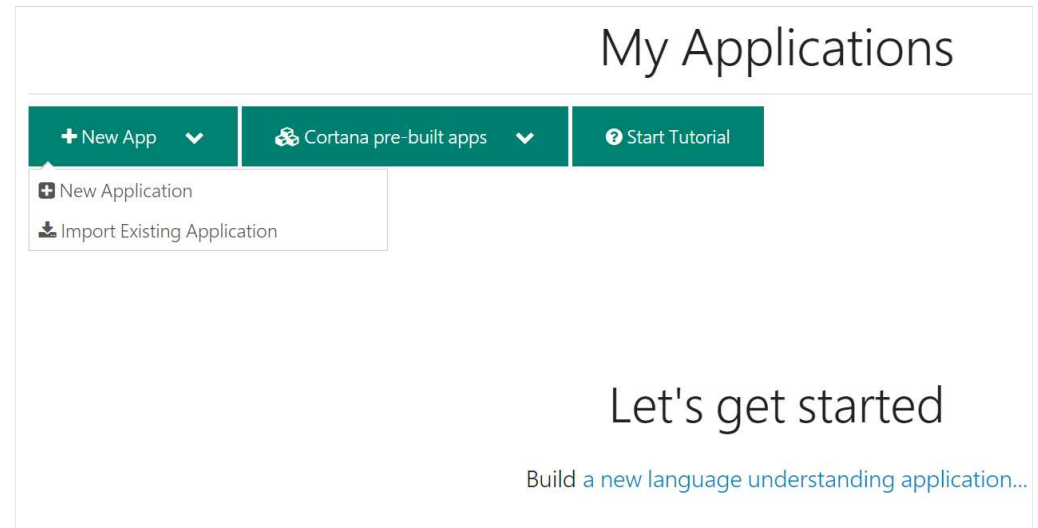


LUIS Building a LUIS Model

- New App
- Choose "Import..."

To work best, LUIS needs many examples to learn from

Fortunately, if you've already trained LUIS before, and you're application will be using the same set of intents and entities you can recycle an existing model and build on it



LUIS Building a LUIS Model

- Once a model has been trained, we can export it as a JSON file
- Click "Choose File" and Navigate to the JSON file provided
- Click "Import"



HotelExampleBot.json

Import an existing application

Choose an application JSON file to import your application. Please note that importing an application may take several minutes. You will automatically be directed to your new app when it is successfully imported.

Choose File and enter (optional) imported app name:

Choose File HotelExampleBot.json

Enter optional App name ...

Import

LUIS JSON File

- Within the JSON file you will find lists of intents and later, text and entities that map to that intent
- You may edit this JSON directly and upload it, or use the interface

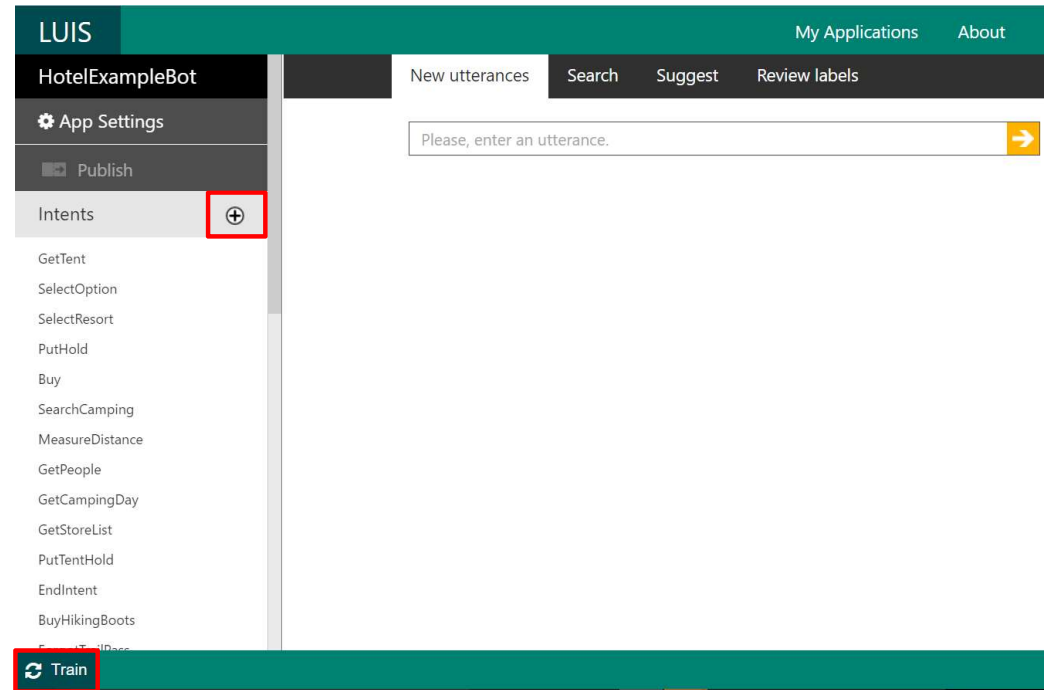


HotelExampleBot.json

```
{
  "luis_schema_version": "1.3.0",
  "name": "HotelExampleBot",
  "desc": "",
  "culture": "en-us",
  "intents": [
    {
      "name": "Buy"
    },
    {
      "name": "BuyHikingBoots"
    },
    {
      "name": "EndIntent"
    },
    {
      "name": "FindHikes"
    },
    {
      "name": "ForgotTrailPass"
    },
    {
      "text": "camp sites",
      "intent": "SearchCamping",
      "entities": []
    },
    {
      "text": "camping",
      "intent": "SearchCamping",
      "entities": []
    },
    {
      "text": "any hikes",
      "intent": "FindHikes",
      "entities": []
    },
    {
      "text": "do they have nice hikes?",
      "intent": "FindHikes",
      "entities": []
    }
  ]
}
```

LUIS Editing

- Once Loaded we see this screen
- The list on the left are the “intents” this model understands
- If we click (+), we can add new intents to the model
- Even though we have imported the file, we still need to train the algorithm to recognize utterances
- Click “Train”



LUIS Publishing

- Training can take some time. Click "Publish"
- Click "Publish web Service"

The screenshot displays the LUIS web interface. The top navigation bar includes 'LUIS', 'My Applications', and 'About'. Below this, the application name 'HotelExampleBot' is shown. The left sidebar contains a menu with 'App Settings' and 'Publish' (highlighted with a red box). The 'Intents' list is visible below. The main area shows a search bar with the placeholder text 'Please, enter an utterance.' and a yellow arrow button. A modal dialog titled 'HTTP SERVICE' is open, showing the status 'Status: service not published' and a red 'Publish web service' button. A note at the bottom of the modal states: 'Note: To enable bot integration, enable action fulfillment in one of your intents.' with a checkbox for 'Enable Action Binding using Microsoft Bot Framework'.

LUIS Publishing

- Training can take some time. Click "Publish"
- Click "Publish web Service"

The screenshot displays the LUIS web interface. The top navigation bar includes 'LUIS', 'My Applications', and 'About'. The main header shows 'HotelExampleBot' and tabs for 'New utterances', 'Search', 'Suggest', and 'Review labels'. A search input field with the placeholder 'Please, enter an utterance.' and a submit button is visible. The left sidebar contains a menu with 'App Settings' and 'Publish' (highlighted with a red box), followed by a list of intents: 'GetTent', 'SelectOption', 'SelectResort', 'PutHold', 'Buy', 'SearchCamping', 'MeasureDistance', 'GetPeople', 'GetCampingDay', 'GetStoreList', and 'PutTentHold'. A modal window titled 'HTTP SERVICE' is open at the bottom, containing the text 'Publish Current Application to URL for access via HTTP' and 'Status: service not published'. A red box highlights the 'Publish web service' button in this modal. Below the button, a note states: 'Note: To enable bot integration, enable action fulfillment in one of your intents.' followed by a checkbox labeled 'Enable Action Binding using Microsoft Bot Framework'.

LUIS Testing

- You can type an utterance into “Query” to see how the model scores it. Given an utterance it will give each possible intent in the model a score
- Example
- Query: “Find a trail”

```
{ "query": "find a trail", "topScoringIntent": { "intent": "GetTrailPass", "score": 0.9974881 }, "intents": [ { "intent": "GetTrailPass", "score": 0.9974881 }, { "intent": "None", "score": 0.5163145 }, { "intent": "ForgotTrailPass", "score": 0.0379541256 }, { "intent": "Buy", "score": 0.0100398948 }, { "intent": "PutTentHold", "score": 0.007901444 }, { "intent": "Help", "score": 0.00303268922 }, { "intent": "SelectResort", "score": 0.00238058716 },
```

HTTP service

Publish Current Application to URL for access via HTTP

Status: Published on 1/24/2017, 2:24:59 PM

[Update published application](#)

Query:

Optional Parameters

☒ Verbose

☐ Enable Bing Spell Check For This Query [?](#)

URL: [https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/02849bf2-da96-4840-825c-73d340943a4a?](https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/02849bf2-da96-4840-825c-73d340943a4a?subscription-key=)
[subscription-key=](#) [=true](#)

Note: To enable bot integration, enable action fulfillment in one of your intents.

☐ Enable Action Binding using Microsoft Bot Framework

Download web service usage logs [Download logs](#)

LUIS AppID and Key

- Click App Settings
- Copy the App Id to your notepad
- In the top banner, click on your account name

The screenshot displays the LUIS portal interface. At the top, a green banner contains 'My Applications' and 'About'. Below this, a dark grey bar shows 'HotelExampleBot' and navigation links: 'New utterances', 'Search', 'Suggest', and 'Review labels'. A search bar with the placeholder 'Please, enter an utterance.' is on the right. The left sidebar lists 'Intents' and various actions like 'GetTent', 'SelectOption', etc. The 'App Settings' dialog is open, showing 'Basic Settings' and 'Auth Settings' tabs. The 'App Id' field is highlighted with a red box, containing the value '02849bf2-da96-4840-825c-73d340943a4a'. The 'App Name' field contains 'HotelExampleBot'. Below, there are sections for 'Endpoint Key Assigned to app:', 'Save a new endpoint key:', and 'Bing Spell Check'. The bottom status bar is green and shows 'Train' and 'Last train completed: 1/24/2017, 2:09:37 PM'.

LUIS AppID and Key

- In the top banner, click on your account name
- Click Subscription Keys
- Copy the API Key to your notepad

App name: EricHullandertest

App ID: c6851ae8-631c-4911-ae11-44a236edc3ad

App pw:

LUIS App Name: HotelExampleBot

LUIS App ID: 02849bf2-da96-4840-825c-73d340943a4a

LUIS Key:



My Settings

General Information	Subscription Keys	External APIs Keys
<h3>Programmatic API Key</h3> <div><input type="text"/> (Reset Programmatic Api Key)</div> <p>Enter new key Need help getting your key?</p> <p>Copy your key from the Azure Portal ...</p> <p>Azure subscription keys (click here to buy a key from Azure)</p> <p>Application Name</p> <p>HotelExampleBot</p>		

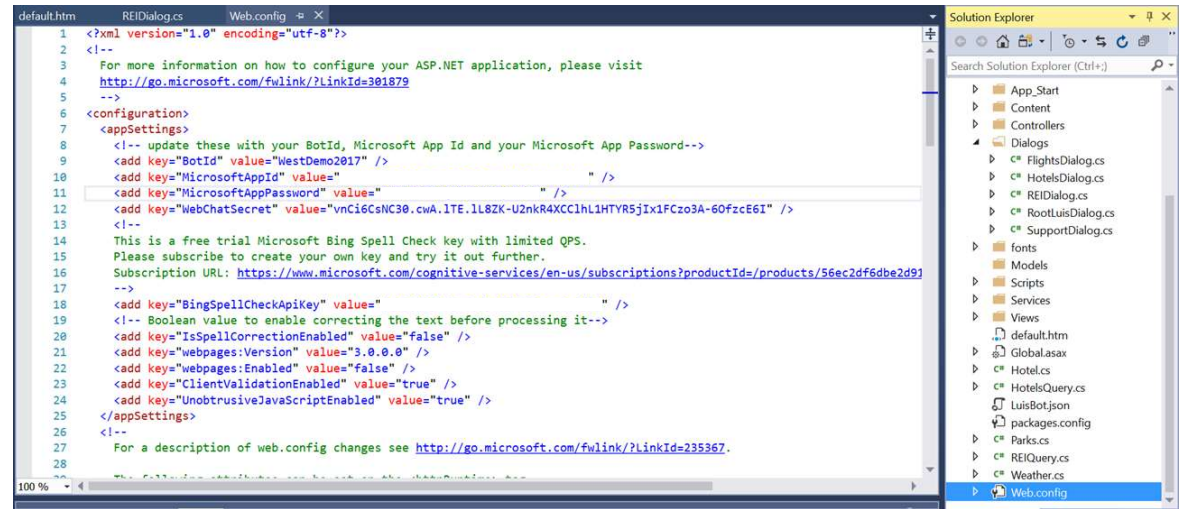
Visual Studio

- Unzip and then navigate to the .sln file in the zip file provided
- Open in Visual Studio
- This can be used as a template for future bot building and will provide a starting point for your exploration
- Some configuration is required

Name	Date modified	Type	Size
obj	1/24/2017 12:12 AM	File folder	
packages	11/28/2016 11:54 AM	File folder	
Properties	11/28/2016 11:55 AM	File folder	
Scripts	11/28/2016 11:55 AM	File folder	
Service References	11/28/2016 11:55 AM	File folder	
Services	11/28/2016 11:55 AM	File folder	
Views	11/28/2016 11:55 AM	File folder	
default.htm	1/24/2017 12:20 AM	Chrome HTML Docu...	2 KB
Global.asax	11/28/2016 11:54 AM	ASP.NET Server Appli...	1 KB
Global.asax.cs	11/28/2016 11:54 AM	Visual C# Source File	1 KB
Hotel.cs	11/28/2016 11:55 AM	Visual C# Source File	1 KB
HotelsQuery.cs	11/28/2016 11:54 AM	Visual C# Source File	1 KB
LuisBot.csproj	1/24/2017 12:12 AM	Visual C# Project file	16 KB
LuisBot.csproj.user	1/24/2017 12:12 AM	Per-User Project Opti...	2 KB
LuisBot.csproj.vspssc	11/28/2016 11:55 AM	Source Control Proje...	1 KB
LuisBot.json	11/28/2016 11:54 AM	JSON File	5 KB
LuisBot.sln	11/28/2016 11:54 AM	Visual Studio Solution	2 KB
packages.config	11/28/2016 11:54 AM	XML Configuration File	3 KB
Parks.cs	11/28/2016 11:54 AM	Visual C# Source File	1 KB
README.md	11/28/2016 11:54 AM	MD File	9 KB
REIQuery.cs	11/28/2016 11:54 AM	Visual C# Source File	1 KB
Weather.cs	11/28/2016 11:54 AM	Visual C# Source File	3 KB
Web.config	1/23/2017 11:41 PM	XML Configuration File	6 KB
Web.Debug.config	11/28/2016 11:54 AM	XML Configuration File	2 KB
Web.Release.config	11/28/2016 11:54 AM	XML Configuration File	2 KB

Visual Studio

- Unzip and then navigate to the .sln file in the the zip file provided
- Open in Visual Studio
- Open Web.config
- Fill in from your notepad
 - BotId
 - MicrosoftAppId
 - MicrosoftAppPassword

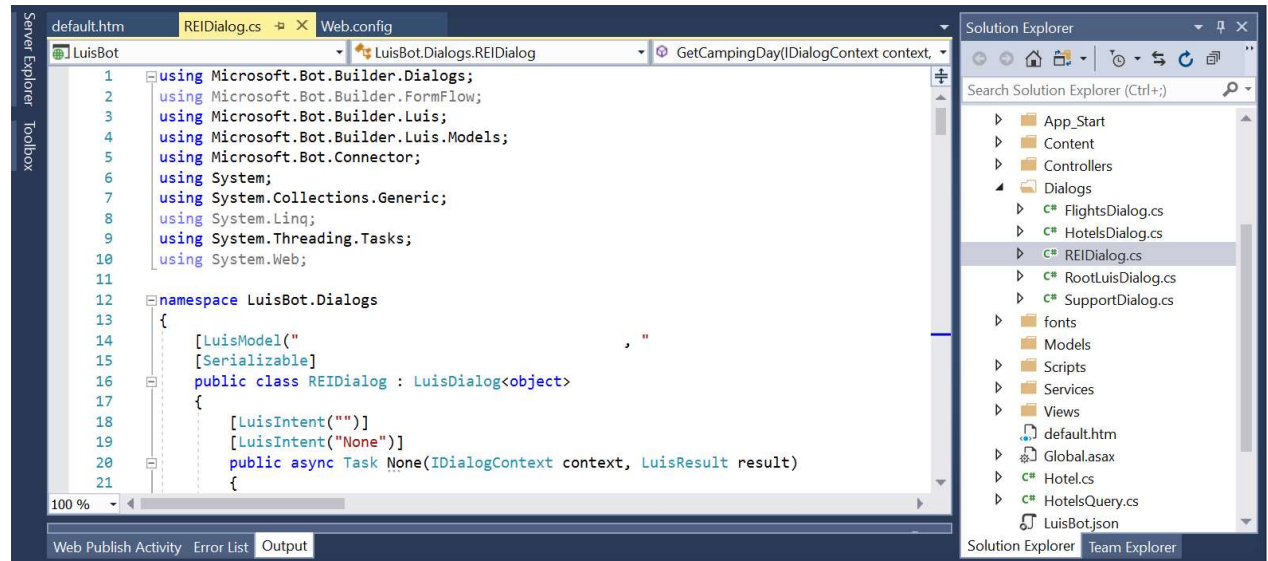


The screenshot shows the Visual Studio IDE with the 'Web.config' file open in the editor. The file contains XML configuration for an ASP.NET application, including settings for BotId, MicrosoftAppId, MicrosoftAppPassword, and various application settings. The Solution Explorer on the right shows the project structure, including folders like App_Start, Content, Controllers, Dialogs, fonts, Models, Scripts, Services, and Views, and files like default.htm, Global.asax, Hotel.cs, HotelsQuery.cs, LuisBot.json, packages.config, Parks.cs, REIQuery.cs, and Weather.cs.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
For more information on how to configure your ASP.NET application, please visit
http://go.microsoft.com/fwlink/?LinkId=301879
-->
<configuration>
  <appSettings>
    <!-- update these with your BotId, Microsoft App Id and your Microsoft App Password-->
    <add key="BotId" value="WestDemo2017" />
    <add key="MicrosoftAppId" value="" />
    <add key="MicrosoftAppPassword" value="" />
    <add key="WebChatSecret" value="vnC16CsNC30.cwA.lTE.1L8ZK-U2nkR4XCC1hL1HTYRSj1x1FCzo3A-60fzcE6I" />
    <!--
    This is a free trial Microsoft Bing Spell Check key with limited QPS.
    Please subscribe to create your own key and try it out further.
    Subscription URL: https://www.microsoft.com/cognitive-services/en-us/subscriptions?productId=/products/56ec2d6db2d97
    -->
    <add key="BingSpellCheckApiKey" value="" />
    <!-- Boolean value to enable correcting the text before processing it-->
    <add key="IsSpellCorrectionEnabled" value="false" />
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <!--
  For a description of web.config changes see http://go.microsoft.com/fwlink/?LinkId=235367.
  -->
</configuration>
```

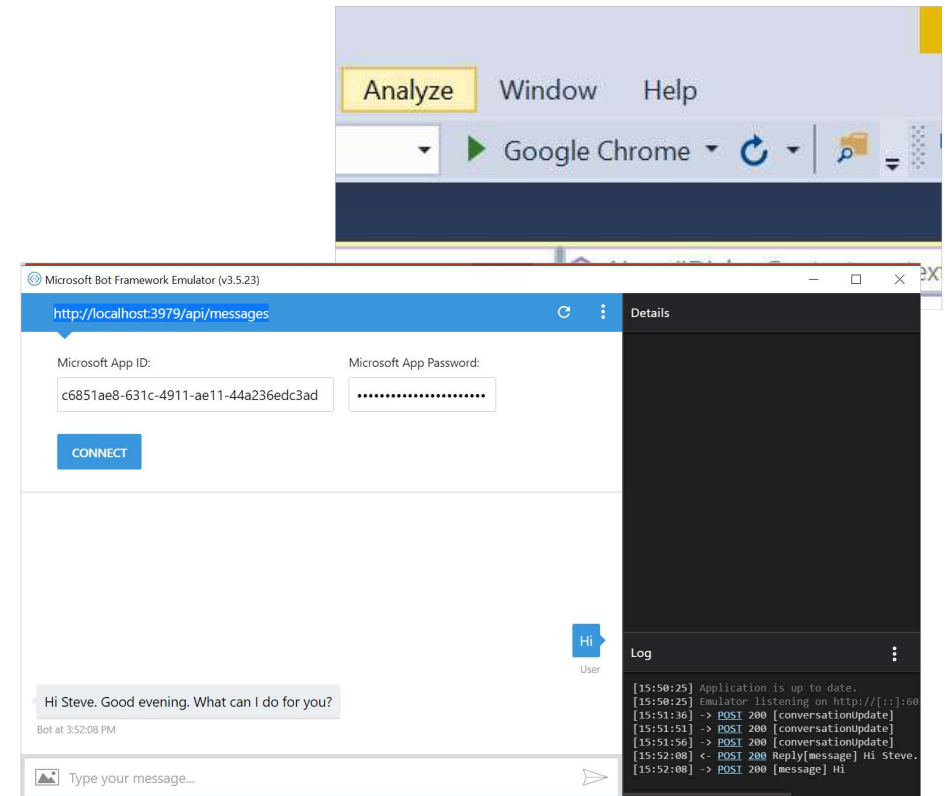
Visual Studio

- Open REIDialog.cs
- Enter the App ID and Password



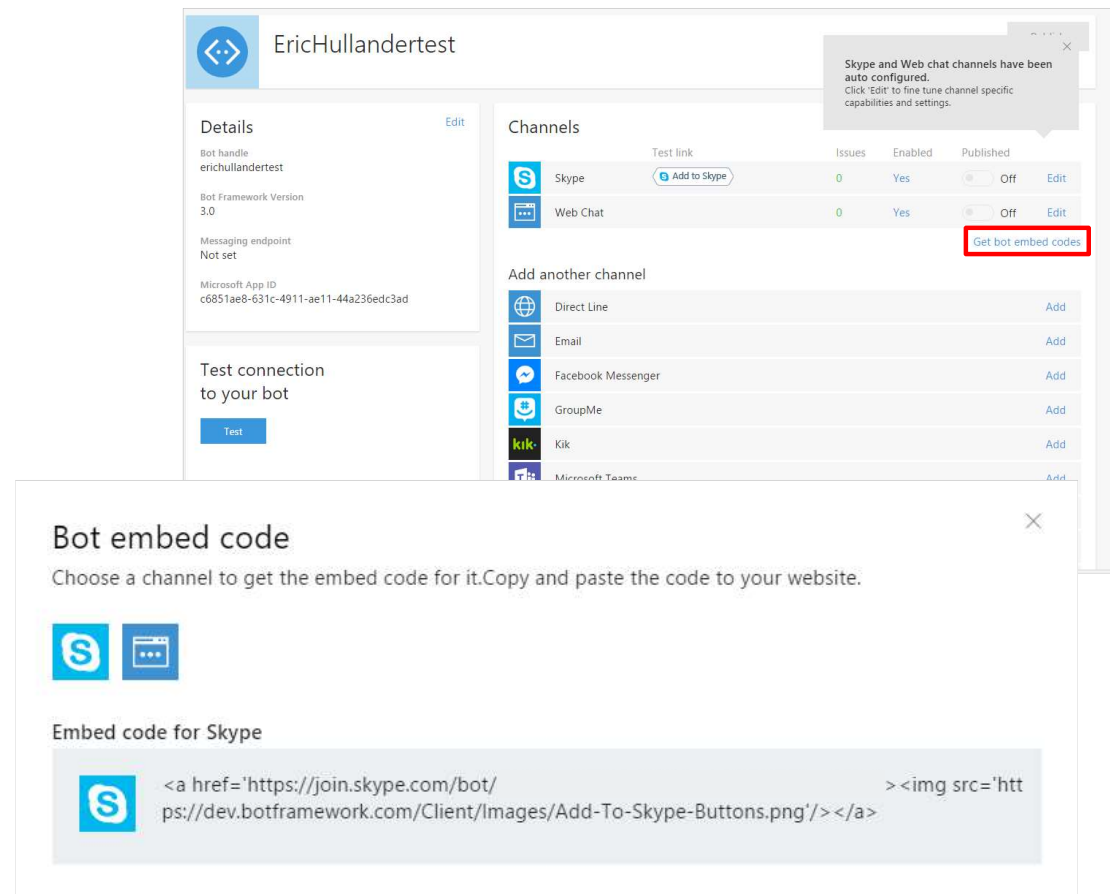
Bot Emulation

- Before we publish we will test the bot locally using the bot emulator
- Click the Play Button to have the bot run on your local host
- Launch the Bot emulator
- Input the address of your local host
- Input your AppID and PW
- Let the emulator connect
- Say "Hi"



Bot Publication Skype

- Return to <https://dev.botframework.com/bots?id=erichullandertest>
- Click the light blue Skype icon
- Click “Get bot embed codes”
- Copy the embed code to your notepad



The screenshot shows the Bot Framework Developer Portal interface for a bot named 'EricHullandertest'. The 'Details' section on the left shows the bot handle 'erichullandertest', version 3.0, and a Microsoft App ID. The 'Channels' section on the right shows the bot is configured for 'Skype' and 'Web Chat'. A red box highlights the 'Get bot embed codes' button for the 'Web Chat' channel. A modal window titled 'Bot embed code' is open, showing the embed code for Skype.

Bot embed code

Choose a channel to get the embed code for it. Copy and paste the code to your website.

Embed code for Skype

```
<a href='https://join.skype.com/bot/ps://dev.botframework.com/Client/Images/Add-To-Skype-Buttons.png'><img src='htt
```


Bot Publication Web

- Click the dark blue web icon icon
- "Click here to open the web..."

The screenshot shows the Microsoft Bot Framework portal for a bot named 'EricHullandertest'. The 'Details' section on the left shows the bot handle 'erichullandertest', framework version '3.0', and a Microsoft App ID. The 'Channels' section on the right lists 'Skype' and 'Web Chat' as configured channels. A red box highlights the 'Web Chat' channel, and a tooltip above it states: 'Skype and Web chat channels have been auto configured. Click 'Edit' to fine tune channel specific capabilities and settings.' Below the channels, there is a list of other channels to add: Direct Line, Email, Facebook Messenger, GroupMe, Kik, and Microsoft Teams.



Channel	Test link	Issues	Enabled	Published	Action
Skype	Add to Skype	0	Yes	Off	Edit
Web Chat		0	Yes	Off	Edit

Add another channel

- [Direct Line](#) [Add](#)
- [Email](#) [Add](#)
- [Facebook Messenger](#) [Add](#)
- [GroupMe](#) [Add](#)
- [Kik](#) [Add](#)
- [Microsoft Teams](#) [Add](#)

Bot embed code

Choose a channel to get the embed code for it. Copy and paste the code to your website.

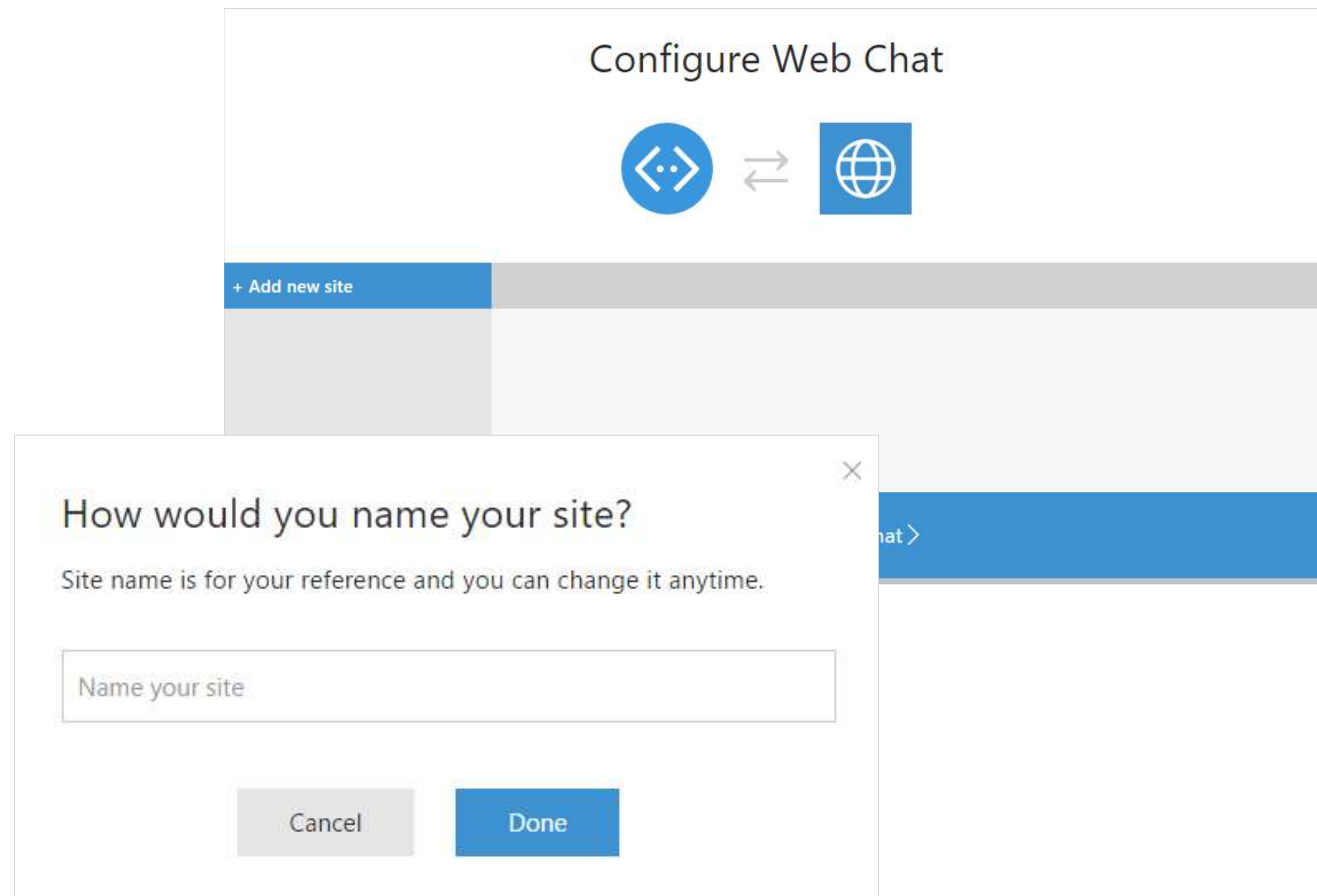
 

Embed code for Web Chat has moved

[Click here to open the Web Chat configuration page >](#)

Bot Publication Web

- Click "Add New Site"
- Give your site a name
 - "MySite"
- Click Done



The screenshot displays the 'Configure Web Chat' interface. At the top, the title 'Configure Web Chat' is centered. Below it, there are two blue icons: a code editor icon (less-than, dot, greater-than) and a globe icon, connected by a double-headed arrow. A blue button labeled '+ Add new site' is visible on the left side of the interface. In the foreground, a modal dialog box is open with the title 'How would you name your site?'. The dialog contains a text input field with the placeholder text 'Name your site'. Below the input field are two buttons: 'Cancel' and 'Done'. A small 'X' icon is in the top right corner of the dialog box. The background interface is partially obscured by the dialog box.

Bot Publication Web

- Show
- Copy Secret to notepad
- Copy Embed Code to notepad
- Replace "YOUR_SECRET_HERE" in embed code with your secret

The screenshot shows the 'Bot Framework Web Chat' publication page for a site named 'EricHullanderTest'. The interface includes a sidebar with a '+ Add new site' button and the current site name. The main content area is divided into sections: 'Secret keys' with two input fields containing placeholder text 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX' and 'Show | Regenerate' links; 'Embed code' with a text area containing an iframe snippet and a 'Copy' link; and 'Preview' with a description of the preview feature and a checked 'Enable Preview' checkbox.

+ Add new site

EricHullanderTest

☐ Disable

Secret keys

XXXXXXXXXXXXXXXXXXXXXXXXXXXX [Show](#) | [Regenerate](#)

XXXXXXXXXXXXXXXXXXXXXXXXXXXX [Show](#) | [Regenerate](#)

Embed code

`<iframe src='https://webchat.botframework.com/embed/erichullandertest?s=YOUR_SECRET_HERE'></iframe>` [Copy](#)

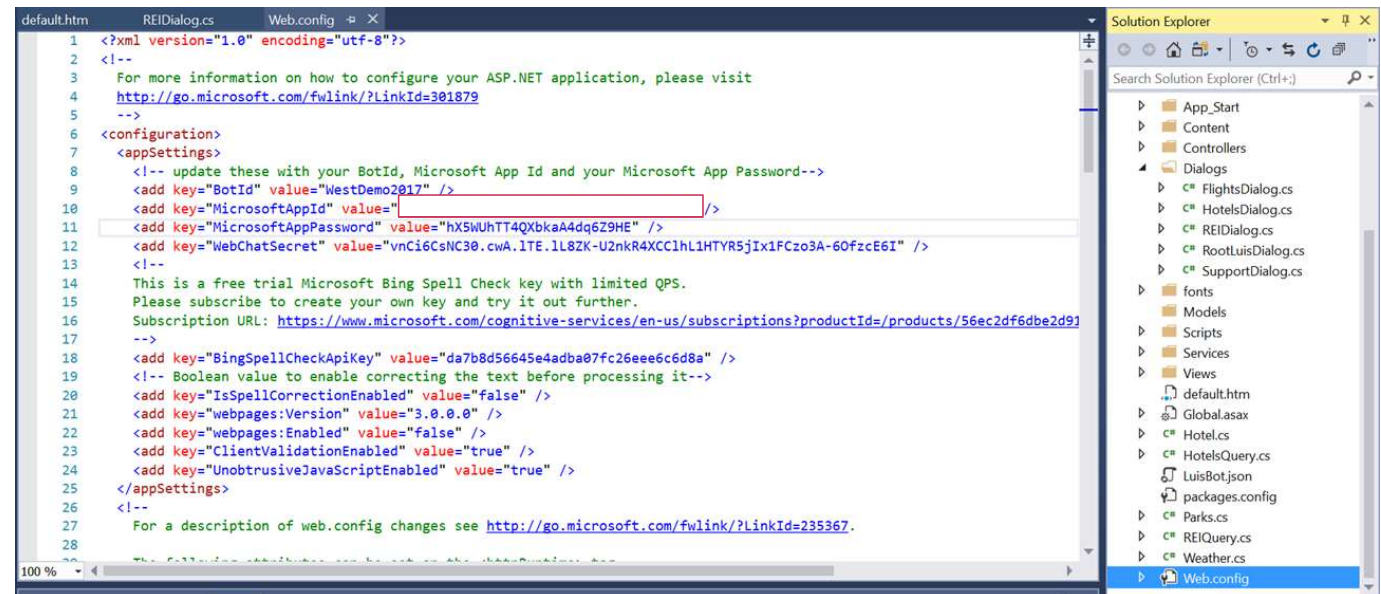
Preview

Enabling Web Chat Preview on this site will give users of your bot a preview version of the Bot Framework Web Chat control. The preview version includes new, beta features that have not yet been officially released.

☒ Enable Preview

Visual Studio

The web secret goes here too

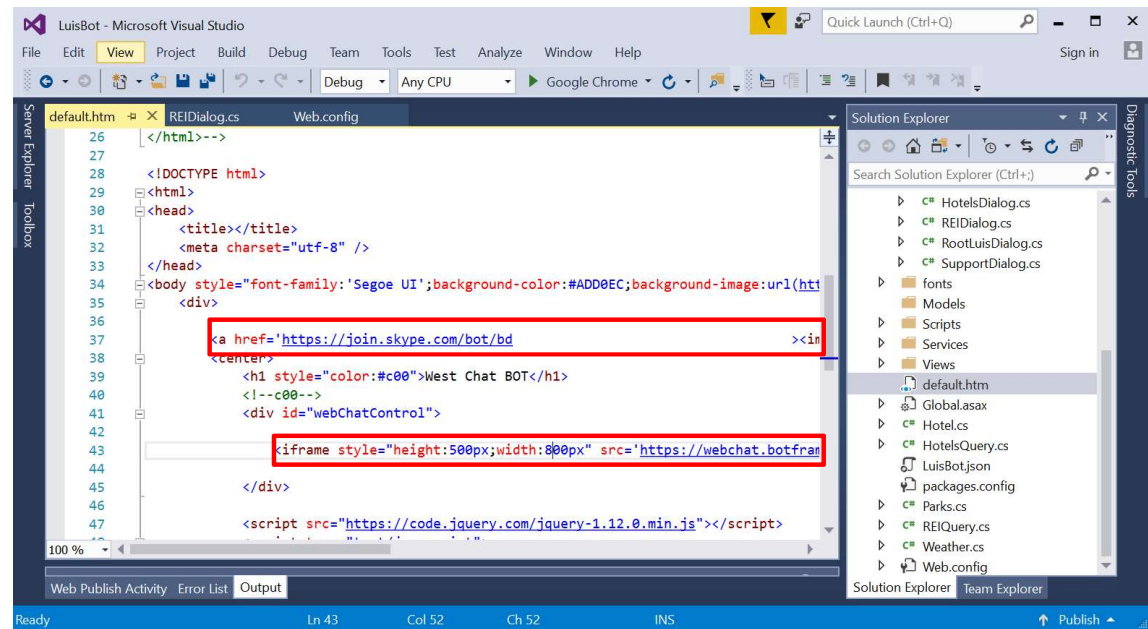


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!--
3 For more information on how to configure your ASP.NET application, please visit
4 http://go.microsoft.com/fwlink/?LinkId=301879
5 -->
6 <configuration>
7   <appSettings>
8     <!-- update these with your BotId, Microsoft App Id and your Microsoft App Password-->
9     <add key="BotId" value="WestDemo2017" />
10    <add key="MicrosoftAppId" value=" " />
11    <add key="MicrosoftAppPassword" value="hX5WUhtT4QXbkaA4dq6Z9HE" />
12    <add key="WebChatSecret" value="vnCi6CsNC30.cwA.1TE.1L8ZK-U2nkR4XCC1hL1HTYR5jIx1FCzo3A-60fzcE6I" />
13    <!--
14 This is a free trial Microsoft Bing Spell Check key with limited QPS.
15 Please subscribe to create your own key and try it out further.
16 Subscription URL: https://www.microsoft.com/cognitive-services/en-us/subscriptions?productId=/products/56ec2df6dbe2d91
17 -->
18    <add key="BingSpellCheckApiKey" value="da7b8d56645e4adba07fc26eee6c6d8a" />
19    <!-- Boolean value to enable correcting the text before processing it-->
20    <add key="IsSpellCorrectionEnabled" value="false" />
21    <add key="webpages:Version" value="3.0.0.0" />
22    <add key="webpages:Enabled" value="false" />
23    <add key="ClientValidationEnabled" value="true" />
24    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
25  </appSettings>
26 <!--
27 For a description of web.config changes see http://go.microsoft.com/fwlink/?LinkId=235367.
28 -->
29 </configuration>
```

Bot Publication Web

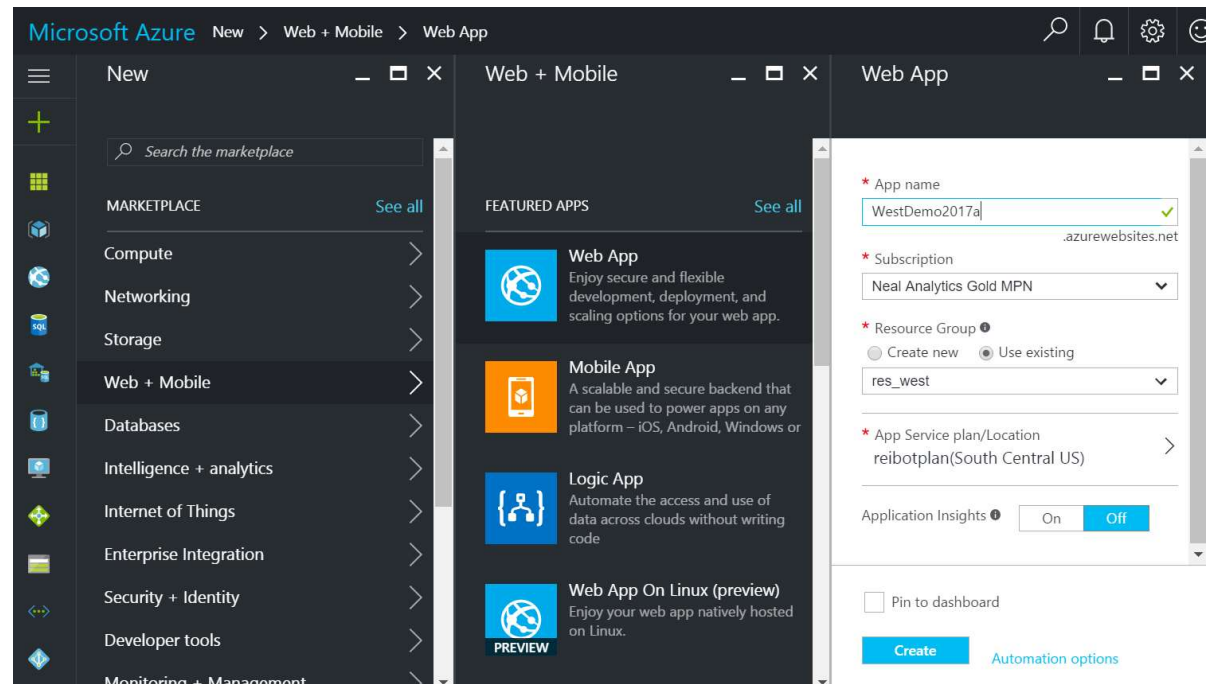
- Navigate to default.htm in Visual Studio
- Find the `<a href>` tag and replace with the `<a href>` tag for Skype in your notepad
- Find the `<iframe>` tag and replace it with the `iframe` tag in your notepad

This will enable Skype and the web interface for your bot



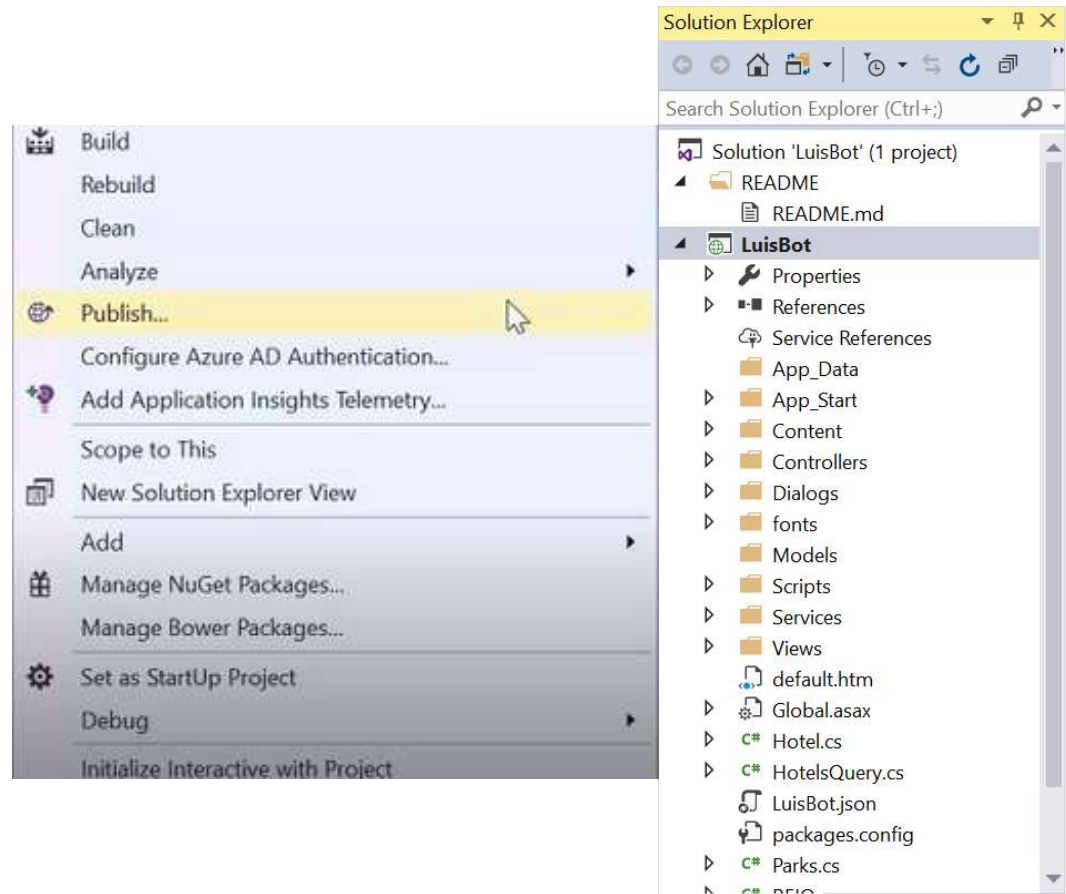
Bot Publication Azure Portal

- <https://portal.azure.com/>
- Click +
- Click "Web + Mobile"
- Web App



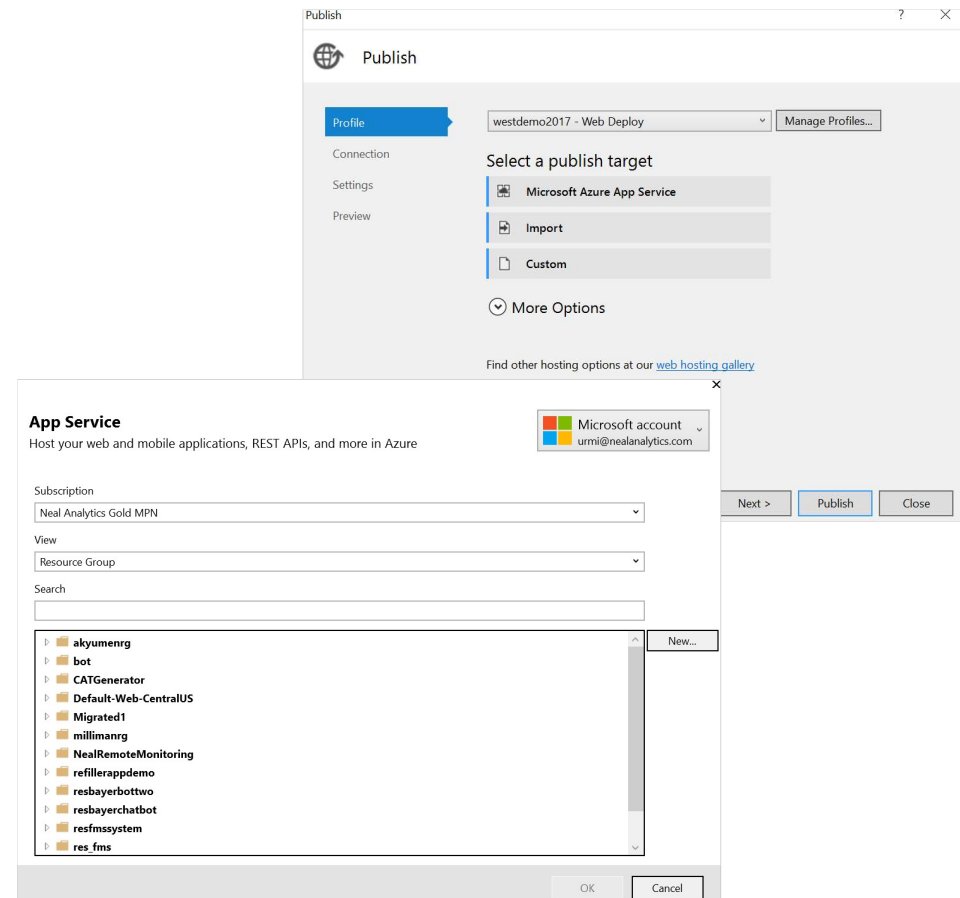
Bot Publication Web

- Right Click on your solution
- Click "Publish"



Bot Publication Web

- Click Profile
- Click Manage Profiles
- Remove any profiles (these are unique to the machine)
- Click “Microsoft Azure App Service”
- Select an account and Sign in
- Select the resource group you just created or selected



Bot Publication Web

- Click "Validate Connection"
- Click "Publish"

Security Feature: You can only publish this app from this machine from now on!!!

The screenshot shows the 'Publish' dialog box in Visual Studio. The title bar says 'Publish'. On the left, there's a sidebar with 'Profile', 'Connection' (selected), 'Settings', and 'Preview'. The main area is titled 'westdemo2017 - Web Deploy'. It contains the following fields and controls:

- Publish method:** A dropdown menu set to 'Web Deploy'.
- Server:** A text box containing 'westdemo2017.scm.azurewebsites.net:443'.
- Site name:** A text box containing 'westdemo2017'.
- User name:** A text box containing '\$westdemo2017'.
- Password:** A text box filled with dots, with a 'Save password' checkbox checked below it.
- Destination URL:** A text box containing 'http://westdemo2017.azurewebsites.net'.
- Validate Connection:** A button with a green checkmark icon next to it, indicating a successful connection.
- Navigation buttons:** At the bottom, there are four buttons: '< Prev', 'Next >', 'Publish' (highlighted in blue), and 'Close'.

Bot is Live

- Click "Validate Connection"
- Click "Publish"

Security Feature: You can only publish this app from this machine from now on!!!


- Return to <https://dev.botframework.com/bots?id=erichullandertest>
- Click "Edit"
- Specify an endpoint

The screenshot displays the Bot Framework Developer Portal interface. At the top, a warning message states: "You cannot test or publish your bot until you specify a messaging endpoint." Below this, the bot's name "Erichullandertest" is shown next to a blue icon with a white code symbol. The "Details" section lists the following information: Bot handle (erichullandertest), Bot Framework Version (3.0), Messaging endpoint (Not set), and Microsoft App ID (c6851ae8-631c-4911-ae11-44a236ec). A "Test connection to your bot" button is visible. The "Configuration" section shows the "Messaging endpoint" field with the value "http://westdemo2017.azurewebsites.net/api/messages". Below this, a button labeled "Manage Microsoft App ID and password" is present. The "Paste your app ID below to continue" section shows the App ID "c6851ae8-631c-4911-ae11-44a236ec3ad".

QnA Maker Chat BOT

Steps for creating QnA maker services (<https://qnamaker.ai>)

Add URLs of FAQ in
QnA maker services

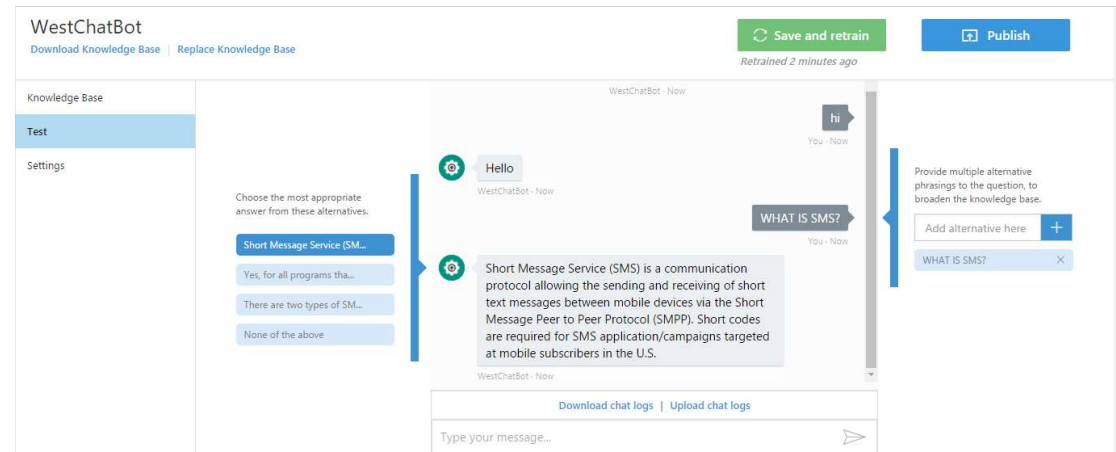


The screenshot shows the 'SETTINGS' page of the QnA Maker service. On the left, a sidebar contains 'Knowledge Base', 'Test', and 'Settings' (which is highlighted). The main area is titled 'SETTINGS' and contains a 'Service name' field with the value 'WestChatBot'. Below this is a section titled 'URLs' with a list of five input fields. The first four fields contain 'https://www.' and each has a trash icon to its right. The fifth field contains 'http://' and has a '+ Add another' link to its right.

SETTINGS	
Service name	WestChatBot
URLs	
https://www.	
https://www.	
https://www.	
https://www.	
http://	+ Add another

After train and publish QnA service we can test it

- We can add alternatives for each questions and again retrain it and publish it
- Highlighted is one that's already picked and we can change if we want
- We can download Chat logs from here



Generate knowledgebase Key and Subscription key

Use this keys in our BOT application to connect with QnA maker service

Deployment details

Sample HTTP request

```
POST /knowledgebases/26d16c8f-d19b-4549-add3-d27b883f8c21/generateAnswer
Host: https://westus.api.cognitive.microsoft.com/qnamaker/v1.0
Ocp-Apim-Subscription-Key:
Content-Type: application/json
{"question":"hi"}
```

BOT Application C# coding

QnA result class to get JSON response

4 references

```
public class QnarResult  
{
```

```
    /// <summary>  
    /// The top answer found in the QnA Service.  
    /// </summary>
```

```
    [JsonProperty(PropertyName = "answer")]
```

5 references

```
    public string Answer { get; set; }
```

```
    /// <summary>  
    /// The score in range [0, 100] corresponding to the top answer found in the QnA Service.  
    /// </summary>
```

```
    [JsonProperty(PropertyName = "score")]
```

0 references

```
    public double Score { get; set; }
```

```
}
```

Getting response based on user FAQ

```
{
    string responseString = string.Empty;
    //var arg = await argument;
    var query = activity.Text;
    var knowledgebaseId = "26d16c8f-d19b-4549-add3-d27b883f8c21"; // Use knowledge base id created.
    var qnamakerSubscriptionKey = " "; //Use subscription key assigned to you.

    //Build the URI
    Uri qnamakerUriBase = new Uri("https://westus.api.cognitive.microsoft.com/qnamaker/v1.0");
    var builder = new UriBuilder($"{qnamakerUriBase}/knowledgebases/{knowledgebaseId}/generateAnswer");

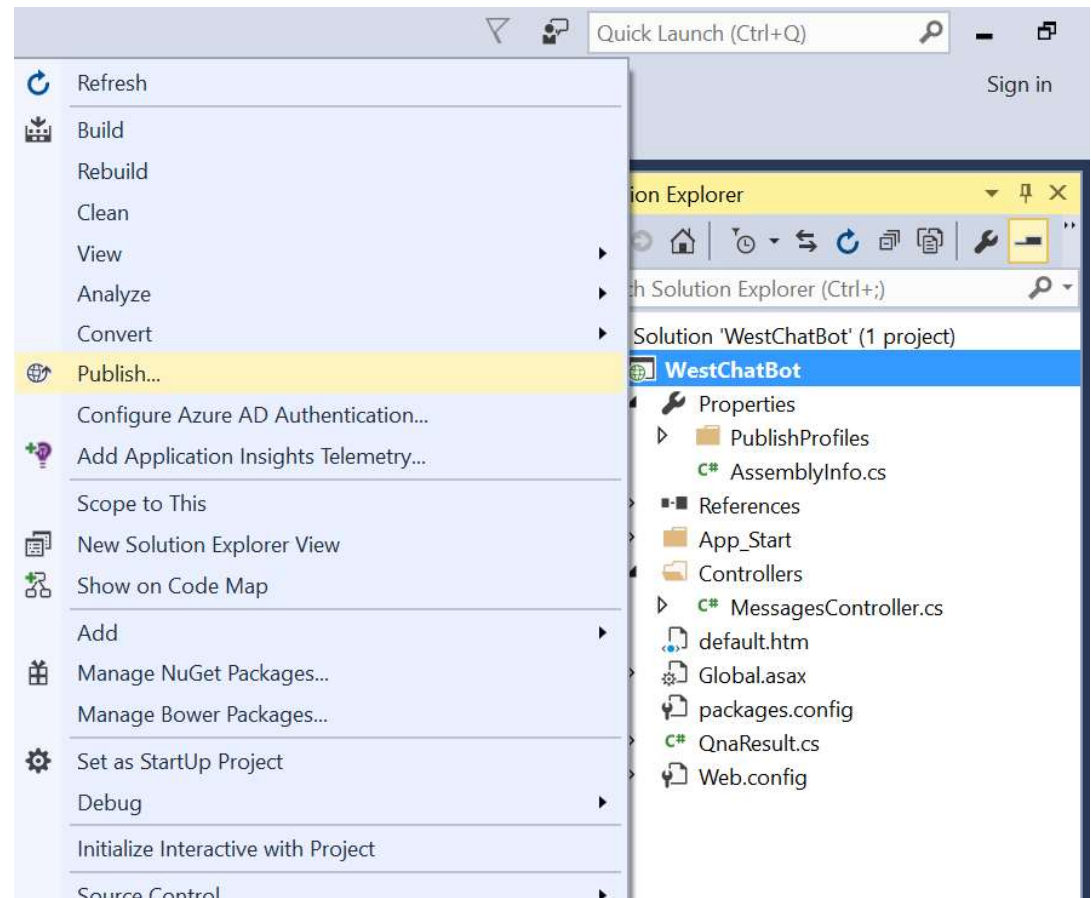
    //Add the question as part of the body
    var postBody = $"{{\"question\": \"{query}\"}}";

    //Send the POST request
    using (WebClient client = new WebClient())
    {
        //Set the encoding to UTF8
        client.Encoding = System.Text.Encoding.UTF8;

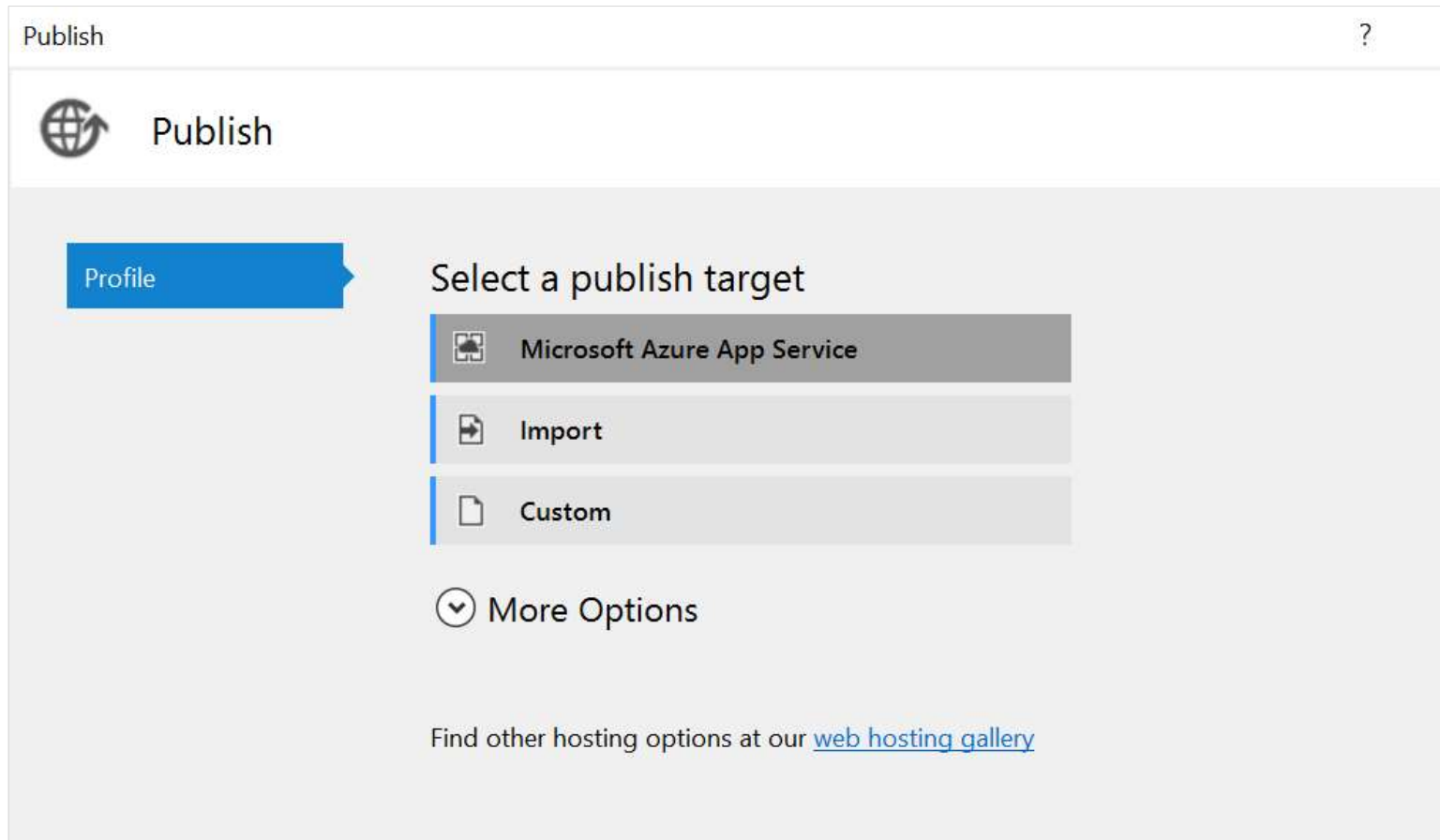
        //Add the subscription key header
        client.Headers.Add("Ocp-Apim-Subscription-Key", qnamakerSubscriptionKey);
        client.Headers.Add("Content-Type", "application/json");
        responseString = client.UploadString(builder.Uri, postBody);
    }
    QnaResult response1;
    response1 = JsonConvert.DeserializeObject<QnaResult>(responseString);
    if (response1.Answer.Equals("No good match found in the KB"))
    {
        response1.Answer = "Sorry I do not understand your question. You can reach out to our customer service representative at 425.672.8762 for further assistance.";
    }
    Activity reply = activity.CreateReply(response1.Answer);
    await connector.Conversations.ReplyToActivityAsync(reply);
}
catch (Exception e)
{
    var ex = e.ToString();
}
```


Steps for publishing BOT application on Azure websites

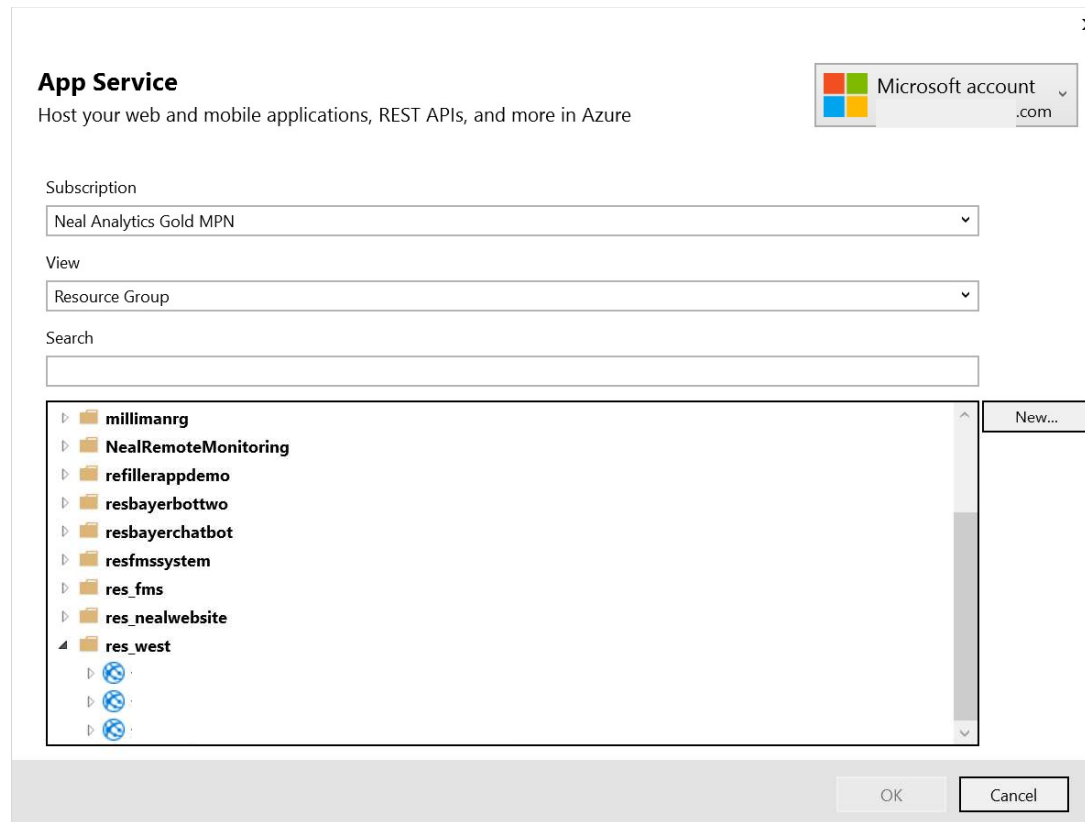
Right click on solution and select publish



Select Azure App Service from Publish dialog



Select Resource Group and Web App which you created on Azure Portal



Now it generates password and after validating it Publish it

The screenshot shows the 'Publish' dialog box in Visual Studio. The title bar says 'Publish'. On the left, there's a sidebar with 'Profile', 'Connection' (highlighted in blue), 'Settings', and 'Preview'. The main area is titled 'westchatbot - Web Deploy (2)'. It contains the following fields and controls:

- Publish method:** A dropdown menu set to 'Web Deploy'.
- Server:** A text box containing 'westchatbot.scm.azurewebsites.net:443'.
- Site name:** A text box containing 'westchatbot'.
- User name:** A text box containing '\$westchatbot'.
- Password:** A text box filled with dots, indicating a generated password.
- Save password:** A checked checkbox.
- Destination URL:** A text box containing 'http://westchatbot.azurewebsites.net'.
- Buttons:** 'Validate Connection', '< Prev', 'Next >', 'Publish' (highlighted in blue), and 'Close'.