**TRPP Hand­s-on lab**

Microsoft Bot Framework

Darren Jefford

Anthony Sadarangani

Travis Hilbert

July 2016

******Contents**

**OVERVIEW 3**

**EXERCISE 1: CREATE A BOT 5**

Task 1 – Register a Bot 5

Task 2 – Create a new Bot Application 5

Task 3 – Run a Bot in the Bot Emulator 6

**EXERCISE 2: ADD A LUIS MODEL 8**

Task 1 – Create an Application 8

Task 2 – Adding Intents, Entities, and Labels 9

**EXERCISE 3: CONNECTING THE DOTS 12**

Task 1 – Connect to LUIS 12

Task 2 – Finishing touches 13

Overview

Microsoft Bot Framework is a comprehensive offering to build and deploy high quality bots for your users to enjoy in their favorite conversation experiences. Developers writing bots all face the same problems: bots require basic I/O; they must have language and dialog skills; they must be performant, responsive and scalable; and they must connect to users – ideally in any conversation experience and language the user chooses. Bot Framework provides just what you need to build, connect, manage and publish intelligent bots that interact naturally wherever your users are talking – from text/sms to Skype, Slack, Facebook Messenger, Kik, Office 365 mail and other popular services. This lab will show you how to build a basic bot, connect it to a LUIS model, and get responses back.

**Objectives**

* 1. This lab will show you how to:
  + Create an echo chat Bot
  + Create a LUIS model
  + Integrate a LUIS model into an existing Bot
  + Use the Bot Freamwork Emulator to test your Bot

**System requirements**

* 1. You must have the following to complete this lab:
  + Windows 10
  + Microsoft Visual Studio 2015 (latest update)
  + Bot Framework Emulator
  + Bot Application Template

**Setup**

* 1. You must perform the following steps to prepare your computer for this lab:
  2. Install Microsoft Windows 10.
  3. Install Microsoft Visual Studio 2015.
     1. Update all VS extensions to their latest versions
  4. Install the [Bot Application Template](http://aka.ms/bf-bc-vstemplate) (http://aka.ms/bf-bc-vstemplate)
     1. Save in you VS template directory “%USERPROFILE%\Documents\Visual Studio 2015\Templates\ProjectTemplates\Visual C#"
  5. Install the [Bot Framework Emulator](https://aka.ms/bf-bc-emulator) (https://aka.ms/bf-bc-emulator)

**Exercises**

* 1. This Hands-on lab includes the following exercises:
  2. Create an echo Bot
  3. Create a LUIS model
  4. Conncect the LUIS model to the Bot
  5. Estimated time to complete this lab:  **30 to 45 minutes**.

Exercise 1: Create a Bot

**Task 1 – Register a Bot**

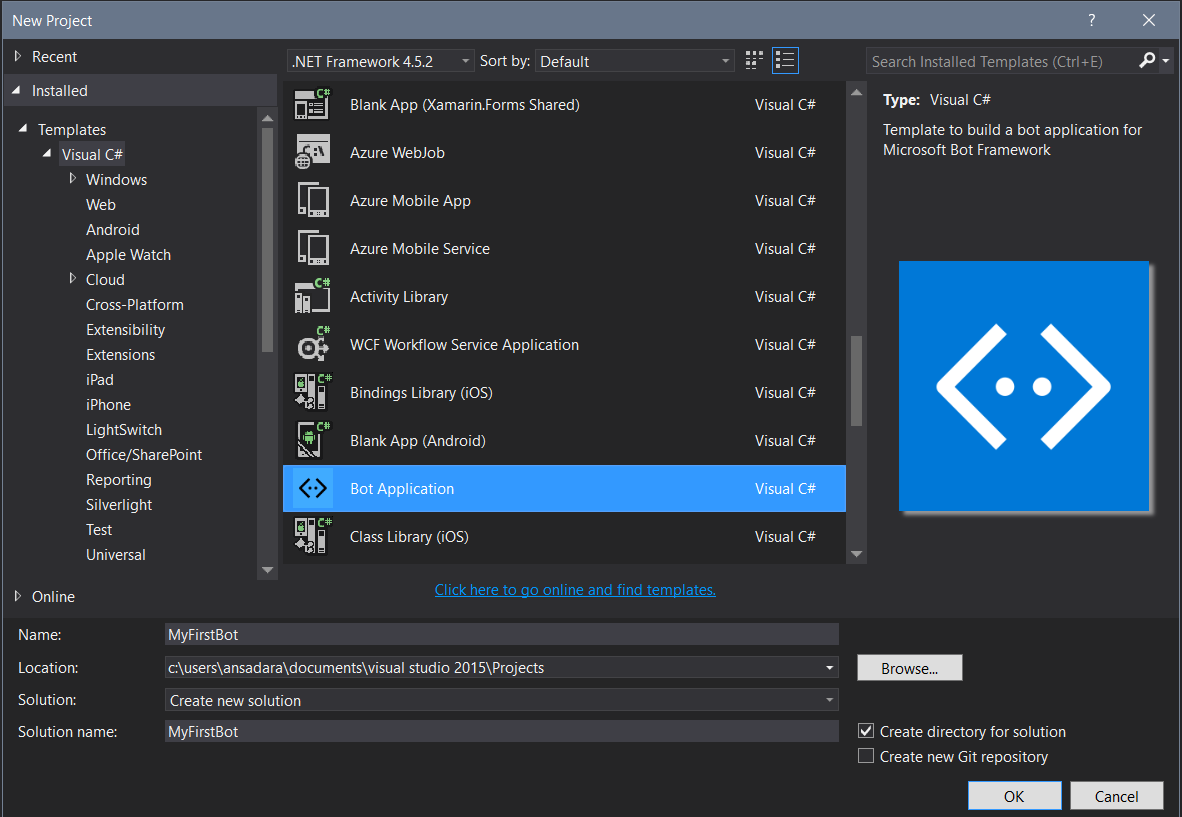
Before we begin, we need to register a new bot on the bot framework website.

* 1. Go to https://dev.botframework.com/ and click “Register a bot” at the top of the screen.
  2. Give your bot a name, a bot handle, and give it a small description.
  3. In the Configuration section, click “Create Microsoft App ID and password”. On the next page, your App ID will be displayed (write this down).
  4. Click “Generate a password to continue”. **Write down this password and keep it safe!**  **This is the only time the password will be displayed!** Click “ok” and then click “Finish and go back to Bot Framework”
  5. Fill out the rest of the required fields. Then click “Resister”.

**Task 2 – Create a new Bot Application**

We will begin by creating a project from the Bot Application template.

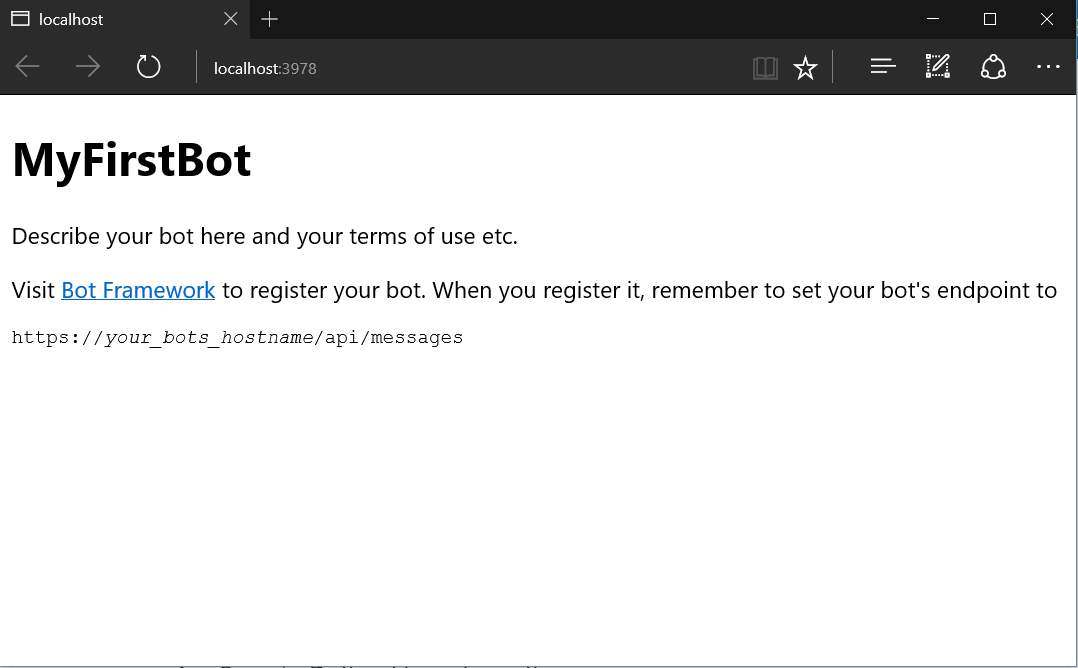
1. In a new instance of Visual Studio 2015, choose **File > New> Project** to open the New Project dialog. Navigate to **Installed > Templates > Visual C#** and select the **Bot Application** template.

Name your project **MyFirstBot** and select the file system location where you will save your solutions. Leave the options selected to **Create new solution** and **Create directory for solution**. 

1. Set your Solution Configuration to **Debug** and your Solution Platform to **Any CPU**. Select **Microsoft Edge** (or your favorite browser)from the Debug Target dropdown menu. You can choose to debug/deploy on a phone device connected via USB outside of this lab.



1. Build and run your app. You will see a blank app browser tab displaying the applications Default.htm

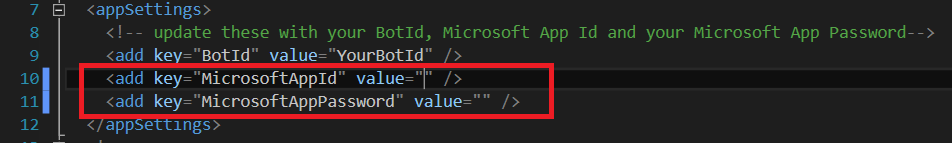
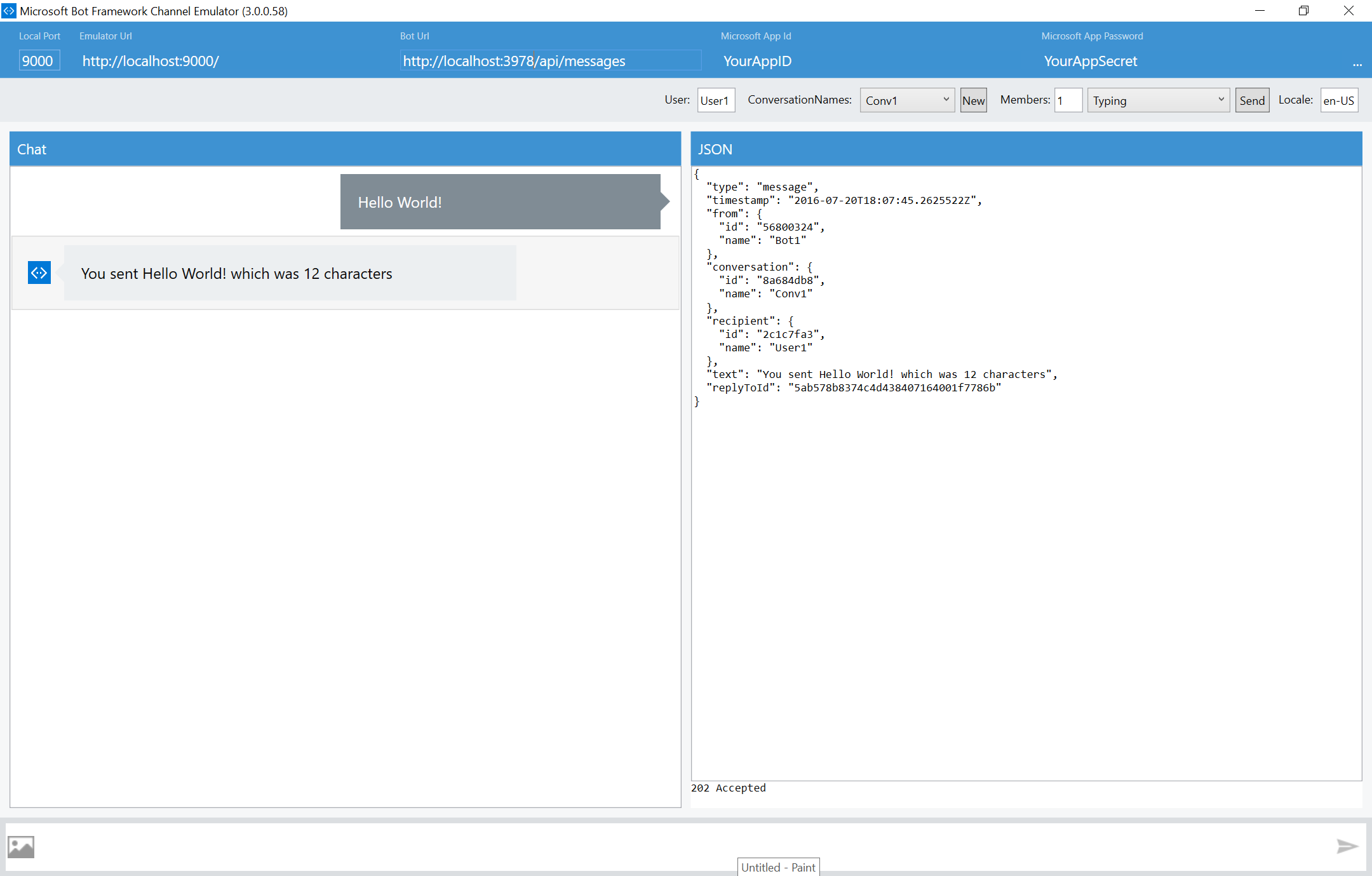


**Keep note of the port your Bot is running on as well as the API URL to be used for testing your Bot.**

1. Return to Visual Studio and stop debugging.

**Task 3 – Run a Bot in the Bot Emulator**

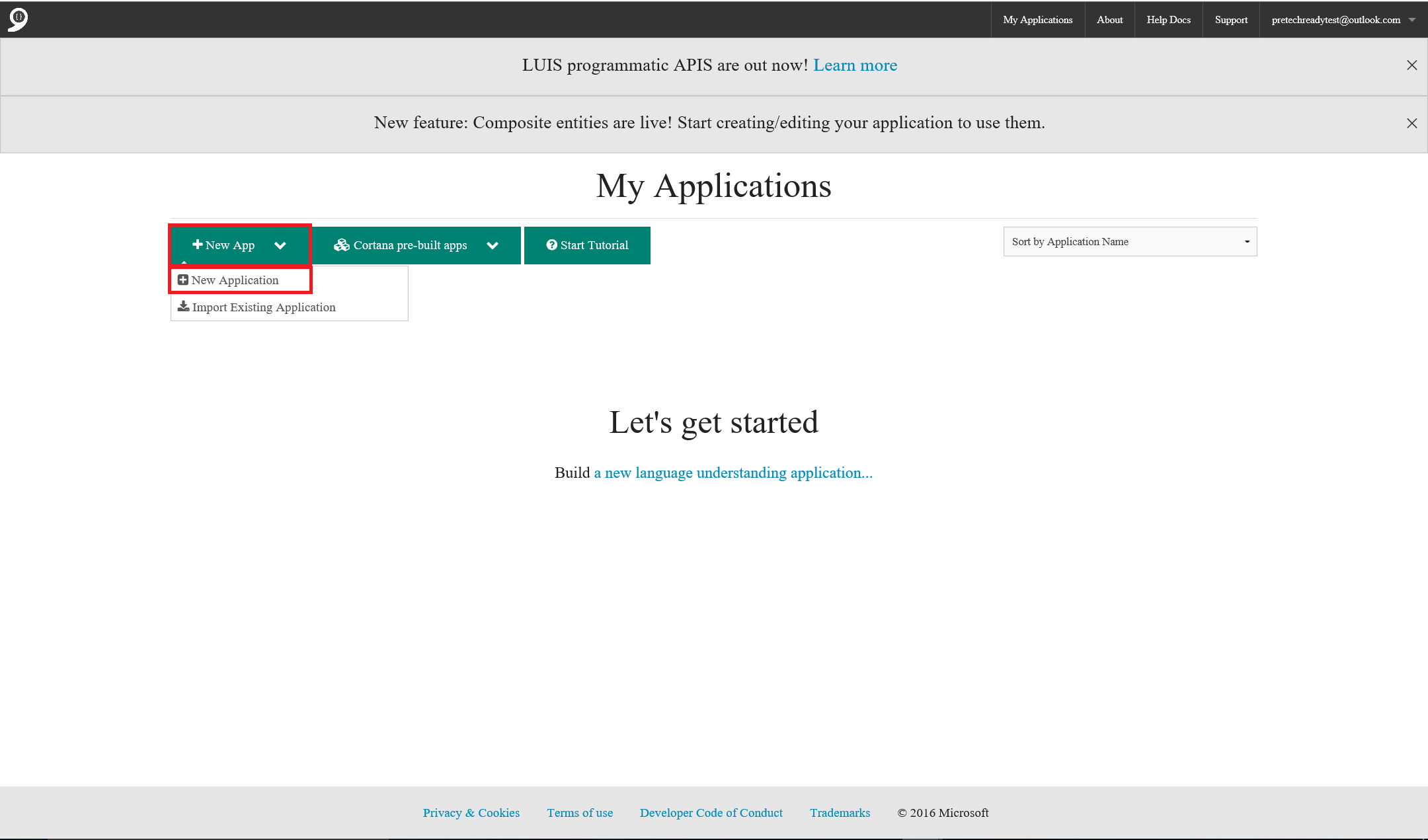
The Bot Framework provides an [emulator](#_Setup) that lets you test calls to your Bot as if it were being called by the Bot Framework cloud service.

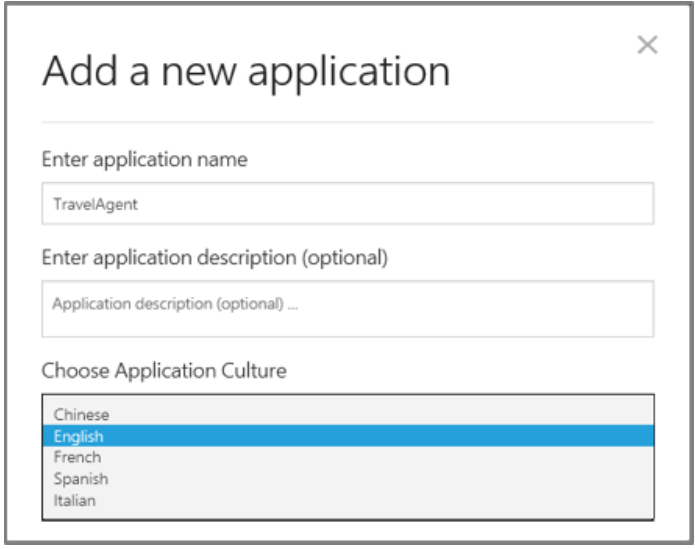
1. Keep your Bot application open in Visual Studio.
2. Open the Bot Framework Emulator.
3. Now, let’s configure the Emulator to interact with your Bot.
   1. Ensure the URL in the Emulator matches the URL displayed in your web browser and add “/api/messages” to the end of the URL.   
      For instance, **http://localhost:3978*/api/messages***
   2. Add the your AppId to the Web.Config file and to the App Id in the Emulator.
   3. Add the your AppSecret to the Web.Config file and to the App Id in the Emulator.
   4. 
   5. ****
4. Run your updated code and test your Bot by typing a message into the Emulator’s text box; located at the bottom. Tt the moment you Bot is very simple and will return the phrase sent along with the character count of the phrase.
   1. 

Exercise 2: Add a LUIS model

Making a LUIS model can be summarized in three steps; create an application, train a model, and publishing the model. In the following walkthrough, we will show you how to do just that.

**Task 1 – Create an Application**

1. Go to [www.luis.ai](http://www.luis.ai/) and log in.
2. Click “New App” in the top left corner of the page and click “New Application” from the drop down menu.
3. In the pop up, name your new LUIS model, give it a description, and choose the application culture. (At the moment Chinese, English, French, Italian, and Spanish are available). Next, click “Add App”.

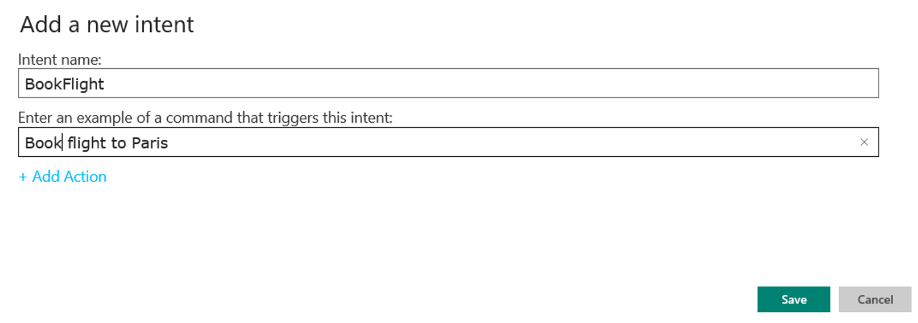


1. When your application is finished provisioning, it will take you to the main page of your new LUIS model. Next, you will train your Luis model.

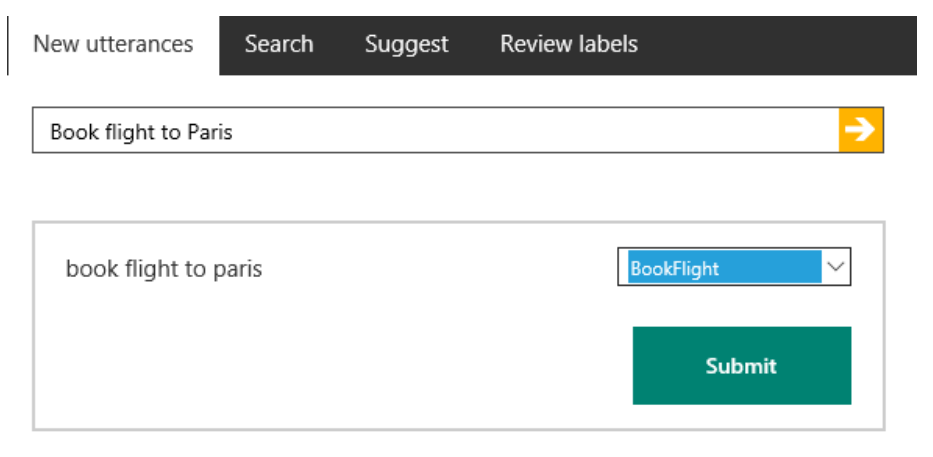
**Task 2 – Adding Intents, Entities, and Labels**

Next, we will add two intents to the application.

1. Click the “+” next to intents and make a “BookFlight” intent. Name it “BookFlight” and give it an example utterance, “Book flight to Paris”, then click save.



1. Your utterance will come up for labelling. Make sure “BookFlight” is selected in the drop down menu and click “Submit” to submit the utterance to your LUIS app.

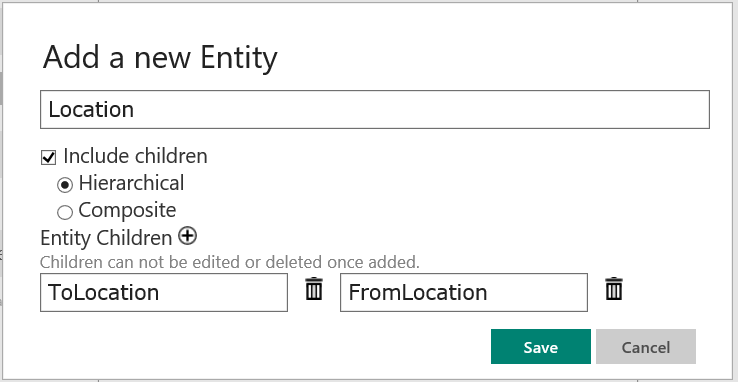


1. Now add a second intent called “GetWeather”, with the example command of “How is the weather in London”. Click “Save”, and accept the presented utterance as a “GetWeather” intent and click “Submit”.

**Defining entities/Hierarchical entities**

You can have the ability to define relationships between entities based on hereditary hierarchical patterns. The generic entity acts as the parent and the children are the specific types/sub groups under the parent, yet both share the same characteristics. An example of this could be our Location

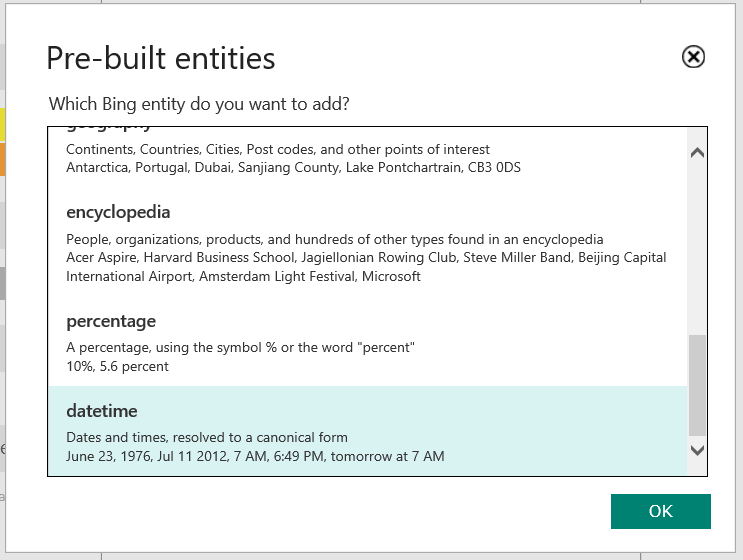
1. To add entities, click the “+” button next to the “Entities” section on the left of the screen, and name it “Location”.
2. Next, click “Include Children” and “Hierarchical”
3. Add a “ToLocation” entity.
4. Next, click “+” next to “Entity Children” and add a “FromLocation” entity.
5. When finished, click “Save”.



**Using Pre-Built Entities**

Next, we are going to add a Pre-Built datetime entity.

1. To add a Pre-Built Entity, click “+” next to “Pre-built Entities” at the left of the screen.
2. Scroll down and select “datetime” from the lift of Pre-Built entities”
3. Then click “Ok”. It’s as easy as that.



**Training your model**

Now that we have the intents and entities defined, we now need to provide more examples of utterances that illustrate these concepts. Click the text box at the top of the screen and start typing in example utterances. You will need to input and label at least five examples of each intent in order to get an accurate model.

Next click “Train”, located at the bottom left of the page, and wait for the completed message to appear. Then click “Publish” at the top left of the screen and click “Publish web service”. A URL will appear with your application id and your subscription key. Write these down and keep them safe. You will be using these later.

Now it’s time to connect the dots.

**Notes on Entity Limits**

* Up to 10 entities of each type can be used in a single application.
* Up to 10 children types for each parent entity may be used.
* When adding children, make sure you add then at the same time you are creating the parent entity.
* To delete an entity with its children, click the entity name in the left-hand panel, and then click “Delete” in the dialog box.

Exercise 3: Connecting the Dots

* 1. This next section will show you how to connect all the dots together and get your bot to recognize utterances entered into the bot emulator.
  2. **Task 1 – Connect to LUIS**
  3. Go back to the echo bot you made earlier. Add a new file and name it “TravelApp”. Inside that new file, add a new class named “LUISApp”. This is where the code for your LUIS model will live.
  4. Next, we need to add code that will handle LUIS intents that are triggered from phrases entered into the Bot Emulator. Insert the following code snippet into the LUISApp file.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.Bot.Builder.Dialogs;

using Microsoft.Bot.Builder.Luis;

using Microsoft.Bot.Builder.Luis.Models;

namespace Bot\_Application1.TravelApp

{

[LuisModel("App ID ", "Subscription Key")]

[Serializable]

public class LUISApp : LuisDialog<object>

{

//Will run when a None intent is Triggered

[LuisIntent("")]

public async Task None(IDialogContext context, LuisResult result)

{

//Add Custom code here.

string message = $"No Intent";

await context.PostAsync(message);

context.Wait(MessageReceived);

}

//Will run when a GetWeather intent is Triggered

[LuisIntent("GetWeather")]

public async Task GetWeather(IDialogContext context, LuisResult result)

{

//Add Custom code here.

string message = $"GetWeather Intent";

await context.PostAsync(message);

context.Wait(MessageReceived);

}

//Will run when a Bookflight intent is Triggered

[LuisIntent("BookFlight")]

public async Task BookFlight(IDialogContext context, LuisResult result)

{

//Add Custom code here.

string message = $"BookFlight Intent";

await context.PostAsync(message);

context.Wait(MessageReceived);

}

}

}

* 1. Input your App ID and Subscription Key from your LUIS model.
  2. **Task 2 – Finishing Touches**
  3. Next we need to edit the “MessageController.cs” file and replace your code with the following:

using System;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using Microsoft.Bot.Connector;

using Newtonsoft.Json;

using Microsoft.Bot.Builder.Dialogs;

using Bot\_Application1.TravelApp;

namespace Bot\_Application1

{

[BotAuthentication]

public class MessagesController : ApiController

{

public async Task<HttpResponseMessage> Post([FromBody]Activity activity)

{

if (activity.Type == ActivityTypes.Message)

await Conversation.SendAsync(activity, () => new LUISApp());

var response = Request.CreateResponse(HttpStatusCode.OK);

return response;

}

}

}

* 1. Rebuild your code and run it.
  2. Go to the bot emulator and start entering test queries. When a intent of “GetWeather” is triggered, the bot will return “GetWeather”, intent of “BookFlight” will return “BookFlight”, and a “None” intent will return “No Intent”.