



[BOT FRAMEWORK & LUIS 实验手册]

Abstract

Bot Framework 是微软公司所提供的简化编写对话机器人的框架，直接支持 Skype、Facebook Messenger、Slack、Kik、Office 365 邮件等对话渠道，可以通过 Direct Line 与微信公众号相连。Bot Framework SDK 支持 C# 与 Node.js，整合了 Microsoft Cognitive Services 中的语义理解服务—LUIS, 帮助开发者建立更加智能的机器人

Yuheng Ding (CSE)

目录

实验一：实现 Bot Framework 基本功能.....	2
1. 下载开发模板.....	2
2. 创建后端服务.....	3
3. 在 Bot Framework 网站注册新创建的 Bot	7
4. 更新后端服务.....	9
5. 发布并测试.....	10
实验二：将 Bot 与 LUIS 结合	11
1. 新建 LUIS 服务.....	11
2. 在 Bot 中集成 LUIS 服务	16

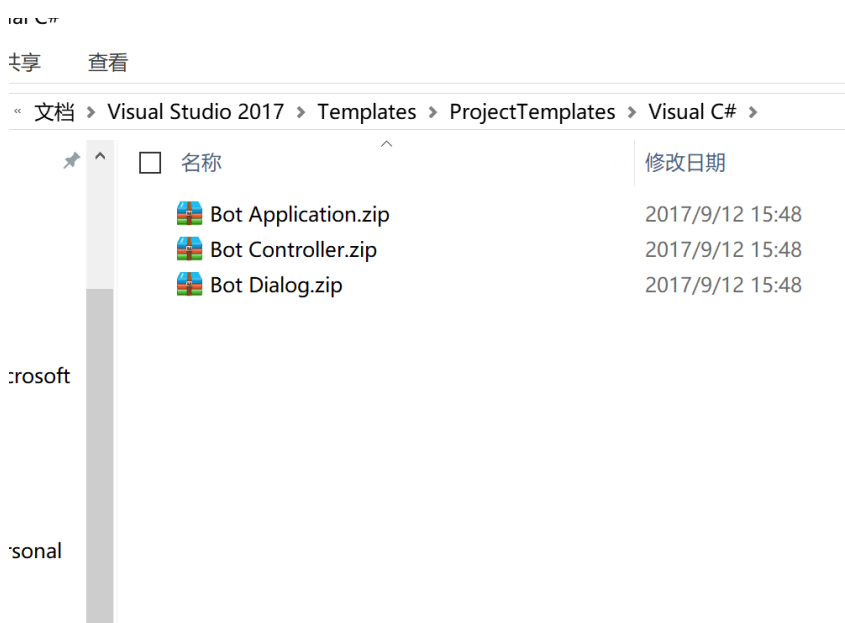
实验一：实现 Bot Framework 基本功能

1. 下载开发模板

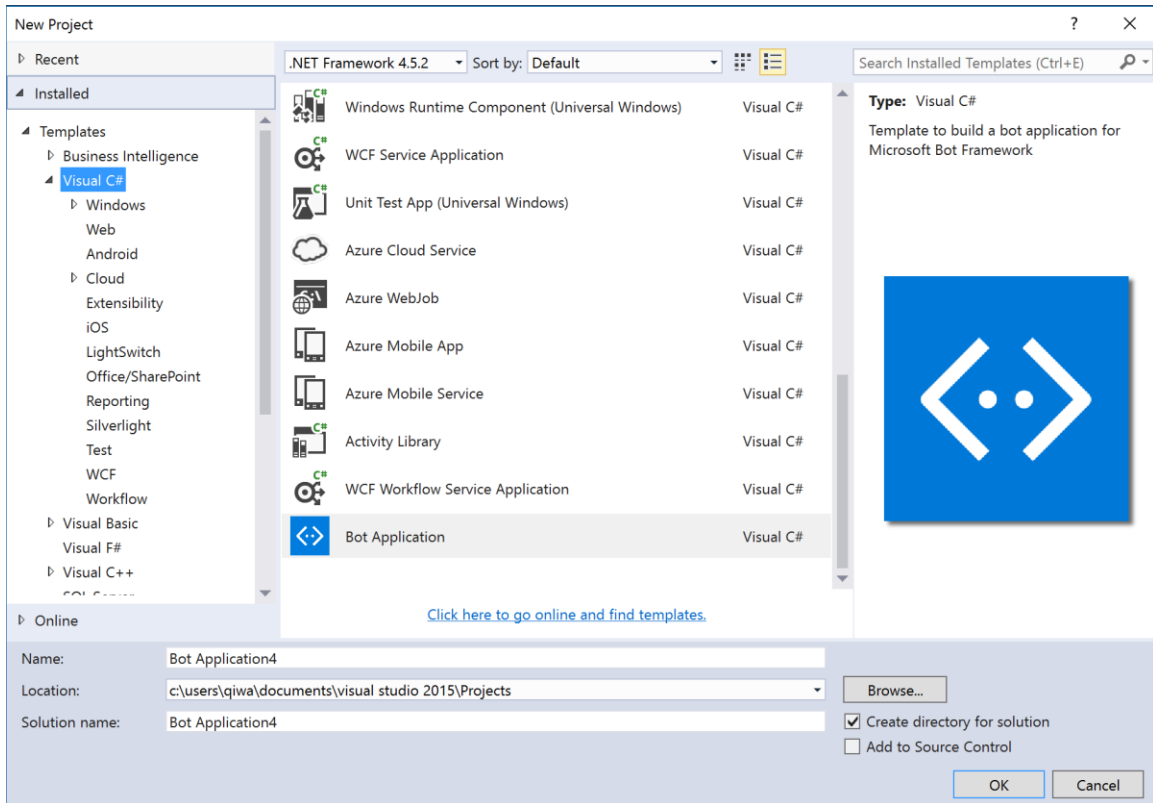
为了方便开发，我们可以下载 Bot Framework 的开发模板。下载地址如下：

[Bot Application](#), [Bot Controller](#), and [Bot Dialog](#) .zip 文件

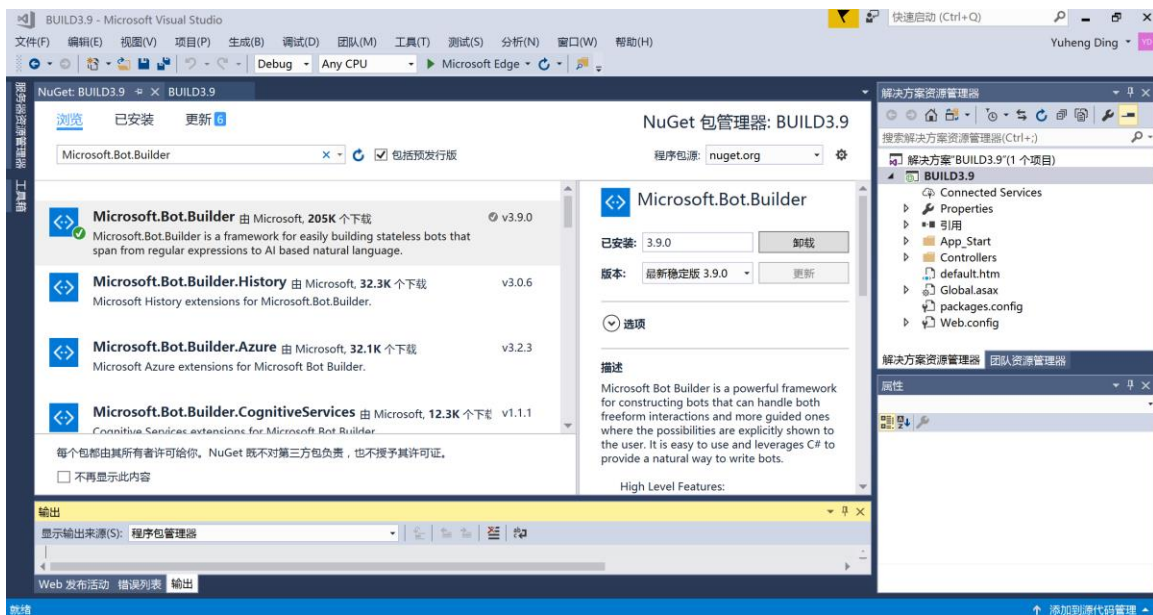
下载完成后，将这些 zip 包复制到%USERPROFILE%\Documents\Visual Studio 2017\Templates\ProjectTemplates\Visual C#\"下（无需解压缩），如下图：



打开 Visual Studio 2017，新建项目，在 Visual C#项目下可以看到 Bot Application 项目类型，选中并点击 OK，创建一个新的 Bot 应用：

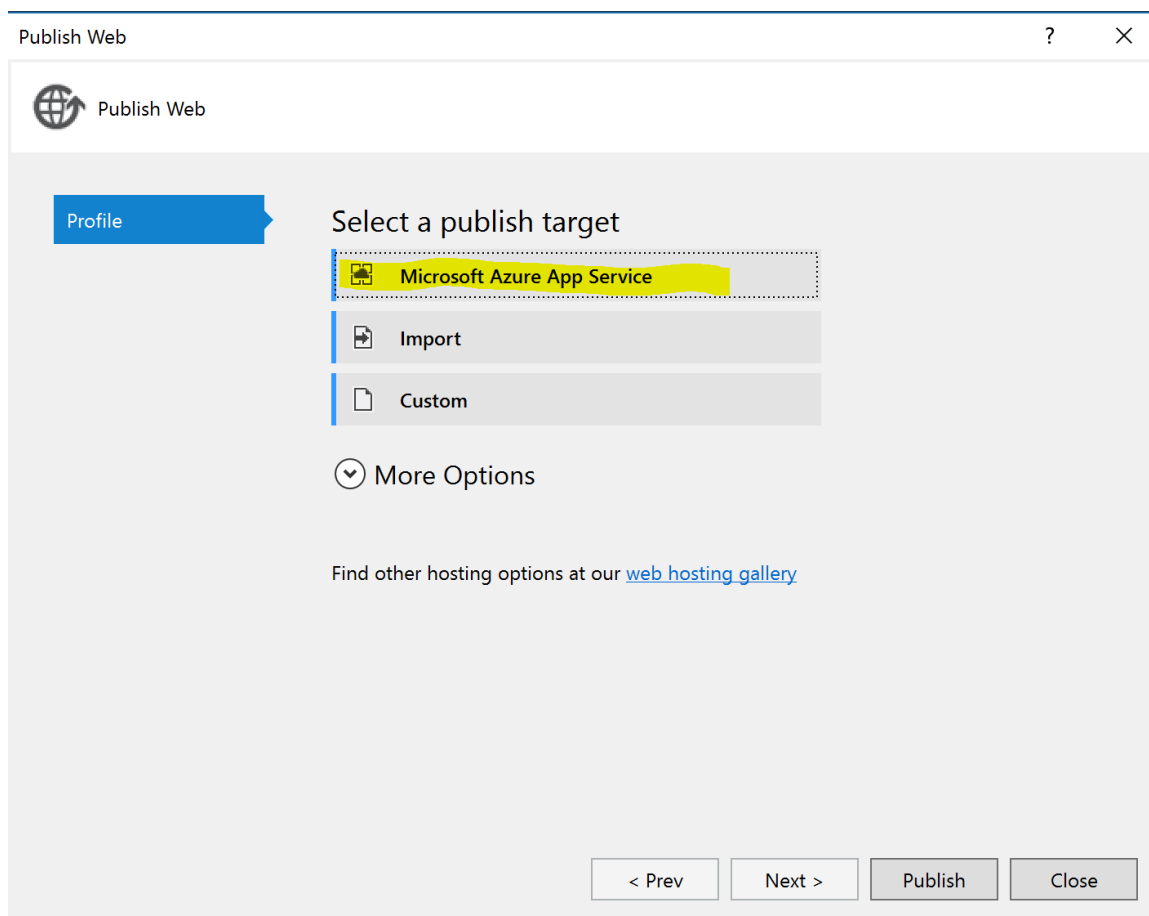


在创建好的 Bot 项目中，右击项目->管理 Nuget 程序包->点击“浏览”页面，搜索“Microsoft.Bot.Builder”，将其更新到最新版。




2. 创建后端服务


右键点击刚刚创建的网站，然后点击 Publish 将其发布到 Azure 上。选择 Microsoft Azure App Service



然后点击新建，如下图所示：

×

**App Service**
Host your web and mobile applications, REST APIs, and more in Azure

Microsoft

Subscription

Microsoft Azure 内部消耗

View

Resource Group

Search

New...

OK

Cancel

新建 Resource Group 及 Services Plan，然后点击 Create，如下图所示：

Close

Create App Service

Host your web and mobile applications, REST API

Hosting

Services

API App Name

BotApplication420160927111830

Subscription

Microsoft Azure 内部消耗

Resource Group

WQBot

App Service Plan

Clicking the Create button will c

[Explore additional Azure services](#)

App Service - BotApplication42016

If you have removed your spending limit or you are using Pay as You

[Learn More](#)

Export...

Close

Configure App Service Plan

An App Service plan is the container for your app. The App Service plan settings will determine the location, features,...

App Service Plan

BotApplication420160927111830Plan

Location

East Asia

Size

Free

OK

Cancel

Create

Cancel

记下 Destination URL, 然后点击 Publish

Publish Web

Publish Web

Profile

Connection

Settings

Preview

BotApplication420160927111830 - Web Deploy

Publish method: Web Deploy

Server: botapplication420160927111830.scm.azurewebsites.net:443

Site name: BotApplication420160927111830

User name: \$BotApplication420160927111830

Password:

☒ Save password

Destination URL: http://botapplication420160927111830.azurewebsites.net

Validate Connection

< Prev Next > Publish Close

3. 在 Bot Framework 网站注册新创建的 Bot

打开 Bot Framework 网站：<https://dev.botframework.com/>

如果没有账号，请先注册，注册完成后登陆，点击导航栏中的 Register a bot，然后填写相关信息：


"Name"：你的 bot 的名字

"Bot Handle"：随便写一串字母，其实就是你的 Bot 的 id

"Description"：你的 Bot 的描述，会在你的 publish 之后主页上显示。

Tell us about your bot

Bot profile



Icon

[Upload custom icon](#)

30K max, png only

* Name ?

WQBot

* Bot handle ?

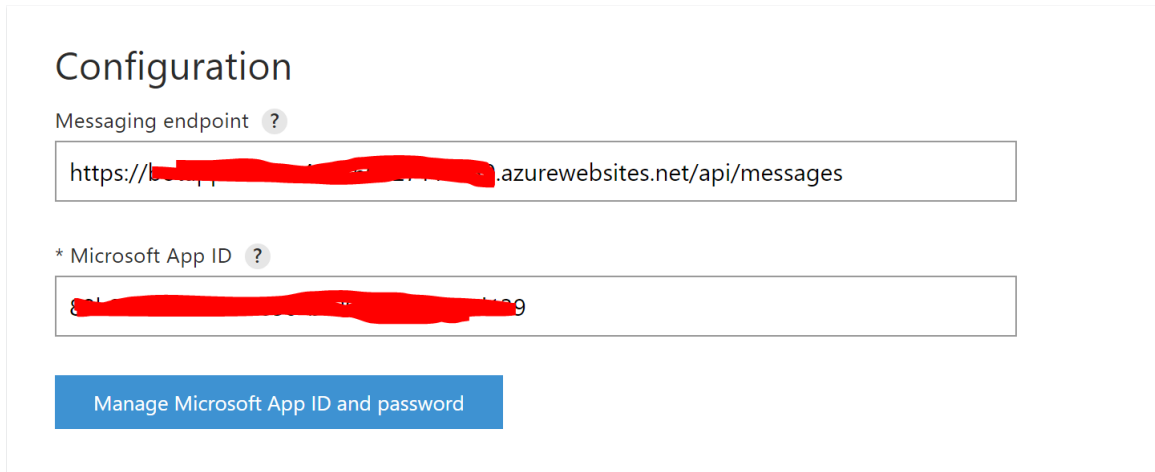
WQBot handle

* Description ?

just for demo

在 Configuration 中点击“Create Microsoft App ID and password”，用 Live ID 登陆，然后点击“Generate an app secret to continue”，记下生成的 password

在 Messaging endpoint 中输入 https://你的服务器地址/api/messages，如下图所示



Configuration

Messaging endpoint ?

https://[redacted].azurewebsites.net/api/messages

* Microsoft App ID ?

[redacted]

Manage Microsoft App ID and password

下面 Publisher profile 中的必填项随便填即可，然后点击 Register，完成 bot 注册






4. 更新后端服务

回到 Visual Studio 中，打开 Web.config 文件，填上你的 botId，刚才创建的 App ID 和 app password

```
<appSettings>
  <!-- update these with your BotId, Microsoft App Id and y
  <add key="BotId" value="WQBot" />
  <add key="MicrosoftAppId" value="82b339a3-d998-4658-bc29-
  <add key="MicrosoftAppPassword" value="OAFpM7kh0JkBuGjjkA
</appSettings>
```

在 bot framework 网站中找到 Web Chat，并点击 Edit

Channels

	Test link	Issues	Enabled	Published	
	Skype		0	Yes (Preview)	 Off Edit
	Web Chat	0	Yes	 Off Edit	

点击 Regenerate Web Chat Secret，然后将 Embed template 和 Secret 中的内容复制下来

How to

^ Web Chat secret

Credentials have been validated.

Secret	zMg7OJrQTm4.cwA.xNo._yIIOR1vMjV-8p3lcFTbgKTfOrCjyedDS-T9m5X
Embed template	<iframe src='https://webchat.botframework.com/embed/WQBotHand

Regenerate Web Chat secret

Deauthorize

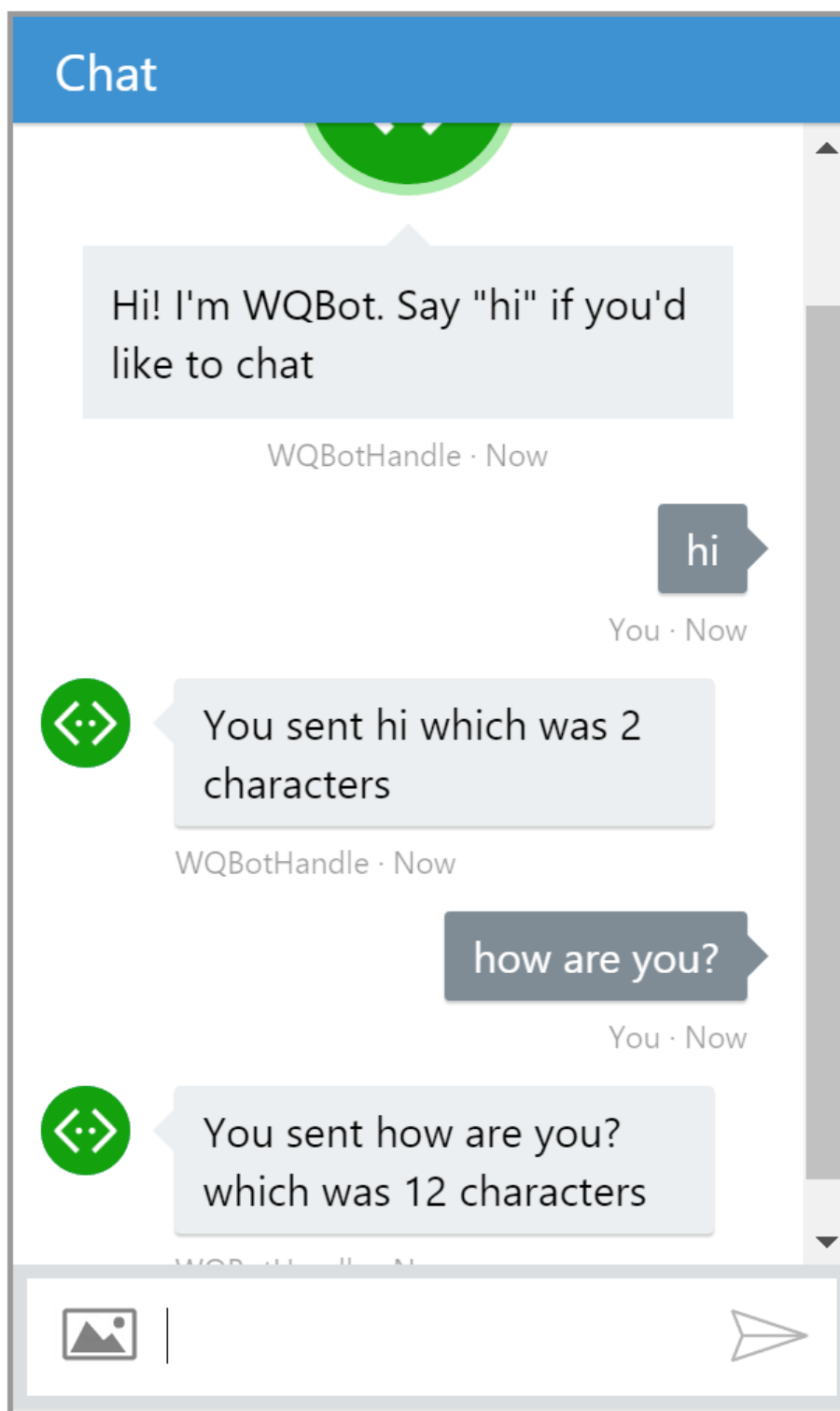
在 iframe 中添加 height 属性，并将 Secret 替换到链接中。打开 Default.htm 文件，将 iframe 嵌入到页面中

```
<body style="font-family:'Segoe UI'">  
  <h1>Bot_Application4</h1>  
  <p>Describe your bot here and your terms of use etc.</p>  
  <p>Visit <a href="https://www.botframework.com/">Bot Framework</a> to register your bot. When you register  
  <iframe style="height:500px" src="https://webchat.botframework.com/embed/WQBotHandle?s=zMg7OJrQTm4.cwA.xNo._yIIOR1vMjV-8p3lcFTbgKTfOrCjyedDS-T9m5X">
```

5. 发布并测试

右键点击 Web 应用，点击 Publish

发布完成后，即可在网页中和 bot 进行交流



实验二：将 Bot 与 LUIS 结合

1. 新建 LUIS 服务

打开链接 <https://www.luis.ai/> 并注册一个新账号，然后点击“New App”创建新应用：

My Apps

Create and manage your LUIS applications ... [Learn more](#)

New App

Import App

Cortana prebuilt apps ▾

在“Add a new application”对话框中，输入应用程序名称、用途等基本信息，然后在“Choose Application Culture”中选择中文，然后点击“Create”

在界面左侧的 Tab 中，点击“Entities”，并为新建的 Add custom entity 起名为“地点”，然后点击 Save，如下图

天气查询

Dashboard

Intents

Entities

Features

Train & Test

Publish App

Dashboard (App Id: f4675141-9af5-4aea-b293-1434df885022)

Facts & statistics about the app's data and the received endpoint hits at any period of time ... [Learn more](#)

App status

Last train: Not trained yet

Last published: Not published yet

Intent Count

Entity Count

Prebuilt Entity Count

Labeled Utterances Count

1 / 80

0 / 30

0 / 8

0

Endpoint Hits Per Period

Total Endpoint Hits

Add Entity

Entity name (REQUIRED)

地点

Entity type (REQUIRED)

Simple

Save

Cancel

点击“Intents”，在 Add Intent 界面中，输入“查询天气”作为 Intent 名称，然后输入例句，如下图所示：

天气查询

Dashboard

Intents

Entities

Features

Train & Test

Publish App

Intents

A listing of intents in the application. Click an intent to view/edit its details, or add a new intent ... [Learn more](#)

Add Intent

Search for intent ...

Intent Name

Utterances

None

0

No custom intents yet. Add your first intent now.

← Back to App list

Add Intent

Intent name (REQUIRED)

查询天气

Save

Cancel

点击 **Save** 之后，在出现的 **Utterance** 里面输入北京天气如何，点击回车，用鼠标选择例句中的“北京”二字，并将其标注为“地点”，然后点击“**Save**”，如下图所示：

<input type="checkbox"/>	Utterance text	Predicted Intent
<input checked="" type="checkbox"/>	[北京] 天气如何	Not trained

Search/Add entity ...

地点

1

点击左边的“**Train & Test**”按钮，完成训练

天气查询

Dashboard

Intents

Entities

Features

Train & Test

Publish App

← Back to App list

Test your application

Use ~~this tool to test~~ the current and published versions of your application, to check if you are progressing on the right track ... [Learn more](#)

Train Application

Please train your application before testing.

Interactive Testing

Batch Testing

☐ Enable published model

Labels view (Ctrl+E)

Entities

Reset console

Please train your application before testing.

Train

训练完成之后，就可以测试了

Use this tool to test the current and published versions of your application, to check if you are progressing on the right track ... [Learn more](#)

Train Application

Last train: Apr 13, 2017 10:47:19 AM | Last publish: **Not published yet.**

Interactive Testing Batch Testing

☐ Enable published model

Labels view (Ctrl+E) Entities [Reset console](#)

→

[地点] 天气怎么样

Current version results
Top scoring intent
查询天气 (1)
Other intents
None (0.09)

对于最终程序中使用需要“Publish”，点击“Publish App”

“

Publish App

Publish your app as a web service or as a chat bot. You can publish a new app or an updated version of a published app ... [Learn more](#)

Essentials

Latest publish: You haven't published your application yet

Endpoint Key (REQUIRED)

~~gnd10795c7-d4b20b0d-75bd2-4446d1~~

[Add a new key to your account](#)

Publish settings

Endpoint slot

Production

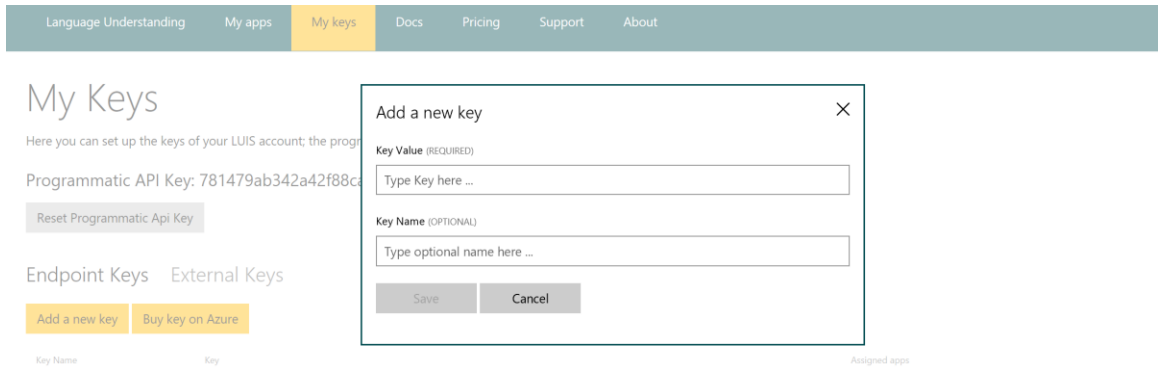
This slot has no published application.

Train

Publish

这里的 Endpoint Key 可以使用免费测试的 1000 条

点击上图中“[Add a new key to your account](#)”，打开之后复制 Programmatic API Key,点击下面 Add A NEW KEY，粘贴刚刚复制的那个 key 倒 key value 这一行，点击 save。



然后回到 Publish App 页面，endpoint key 就可以选了，选好之后就可以 publish 了。将 URL 中的 ID 和 subscription-key 记下来，后面需要用到

Publish settings

Endpoint slot

Production

Slot info

Published version Id: 0.1

Published date: Apr 14, 2017 9:43:33 AM

Endpoint url

<https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/540330cc-4621-4281-8000-000000000000?subscription-key=al-1234567890&timezoneOffset=0.0&verbose=true&q=>

注意 ID 和 Key 的位置分别为：

<https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/ID?subscription-key=KEY&timezoneOffset=0.0&verbose=true&q=>

2. 在 Bot 中集成 LUIS 服务

回到 Visual Studio，打开 Controllers\MessagesController.cs 文件。将

`public async Task<HttpResponseMessage> Post([FromBody]Activity activity)` 中的代码替换为如下内容：

```
public virtual async Task<HttpResponseMessage> Post([FromBody] Activity activity)
{
    if (activity.Type == ActivityTypes.Message)
    {
        await Conversation.SendAsync(activity, () => new WeatherDialog());
    }
    else
    {
        //add code to handle errors, or non-messaging activities
    }

    return new HttpResponseMessage(System.Net.HttpStatusCode.Accepted);
}
```

新建名为 WeatherDialog 的类，粘贴如下代码：

```
namespace WeatherSample
{
    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;
    using Newtonsoft.Json;
    using Newtonsoft.Json.Linq;
    using System.Xml;
    using System.Net.Http;

    [LuisModel("LUIS的ID", "LUIS的Key")]
    [Serializable]
    public class WeatherDialog : LuisDialog<object>
    {
        public const string Entity_location = "Location";

        [LuisIntent("")]
        public async Task None(IDialogContext context, LuisResult result)
        {
            string message = $"您好，我还年轻，目前只能提供中国地区天气查询功能";
            await context.PostAsync(message);
            context.Wait(MessageReceived);
        }

        [LuisIntent("查询天气")]
        public async Task QueryWeather(IDialogContext context, LuisResult result)
        {
            string location = string.Empty;
            string replyString = "";

            if (TryToFindLocation(result, out location))
            {
                replyString = GetWeather(location);

                JObject WeatherResult =
                (JObject)JsonConvert.DeserializeObject(replyString);
                var weatherinfo = new
                {
                    城市 = WeatherResult["weatherinfo"]["city"].ToString(),
                    温度 = WeatherResult["weatherinfo"]["temp"].ToString(),
                    湿度 = WeatherResult["weatherinfo"]["SD"].ToString(),
                    风向 = WeatherResult["weatherinfo"]["WD"].ToString(),
                    风力 = WeatherResult["weatherinfo"]["WS"].ToString()
                };
            }
        }
    }
}
```

```

        await context.PostAsync(weatherinfo.城市 + "的天气情况: 温度" +
weatherinfo.温度 + "度;湿度"+weatherinfo.湿度+";风力"+weatherinfo.风力+";风向
"+weatherinfo.风向);
    }
    else
    {

        await context.PostAsync("亲你要查询哪个地方的天气信息呢, 快把城市的名
字发给我吧");
    }
    context.Wait(MessageReceived);

}

private string GetWeather(string location)
{
    string weathercode = "";
    XmlDocument citycode = new XmlDocument();

citycode.Load("https://wqbot.blob.core.windows.net/botdemo/CityCode.xml");
    XmlNodeList xnList = citycode.SelectNodes("//province//city//county");
    foreach (XmlElement xnl in xnList)
    {
        if (xnl.GetAttribute("name").ToString() == location)
            weathercode = xnl.GetAttribute("weatherCode").ToString();
    }
    HttpClient client = new HttpClient();
    string result =
client.GetStringAsync("http://www.weather.com.cn/data/sk/" + weathercode +
".html").Result;
    return result;
}
private bool TryToFindLocation(LuisResult result, out String location)
{
    location = "";
    EntityRecommendation title;
    if (result.TryFindEntity("地点", out title))
    {
        location = title.Entity;
    }
    else
    {
        location = "";
    }
    return !location.Equals("");
}
}
}
}

```

在 Controllers\MessagesController.cs 中添加 using WeatherSample;

右键点击项目，然后点击 **Publish** 进行发布。发布完成后，即可实现与机器人的简单对话，如下图所示：

