Bot Framework & LUIS 实验手册

目录

[Bot Framework & LUIS 实验手册 1](#_Toc464479633)

[实验一：实现Bot Framework基本功能 3](#_Toc464479634)

[1. 下载开发模板 3](#_Toc464479635)

[2. 创建后端服务 4](#_Toc464479636)

[3. 在Bot Framework网站注册新创建的Bot 8](#_Toc464479637)

[4. 更新后端服务 10](#_Toc464479638)

[5. 发布并测试 11](#_Toc464479639)

[实验二：将Bot与LUIS结合 13](#_Toc464479640)

[1. 新建LUIS服务 13](#_Toc464479641)

[2. 在Bot中集成LUIS服务 17](#_Toc464479642)

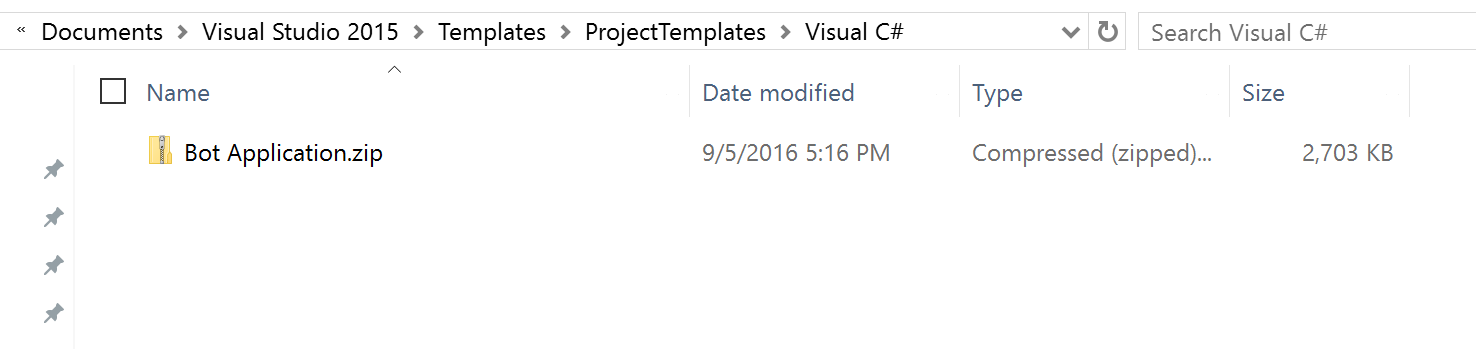
# 实验一：实现Bot Framework基本功能

1. **下载开发模板**

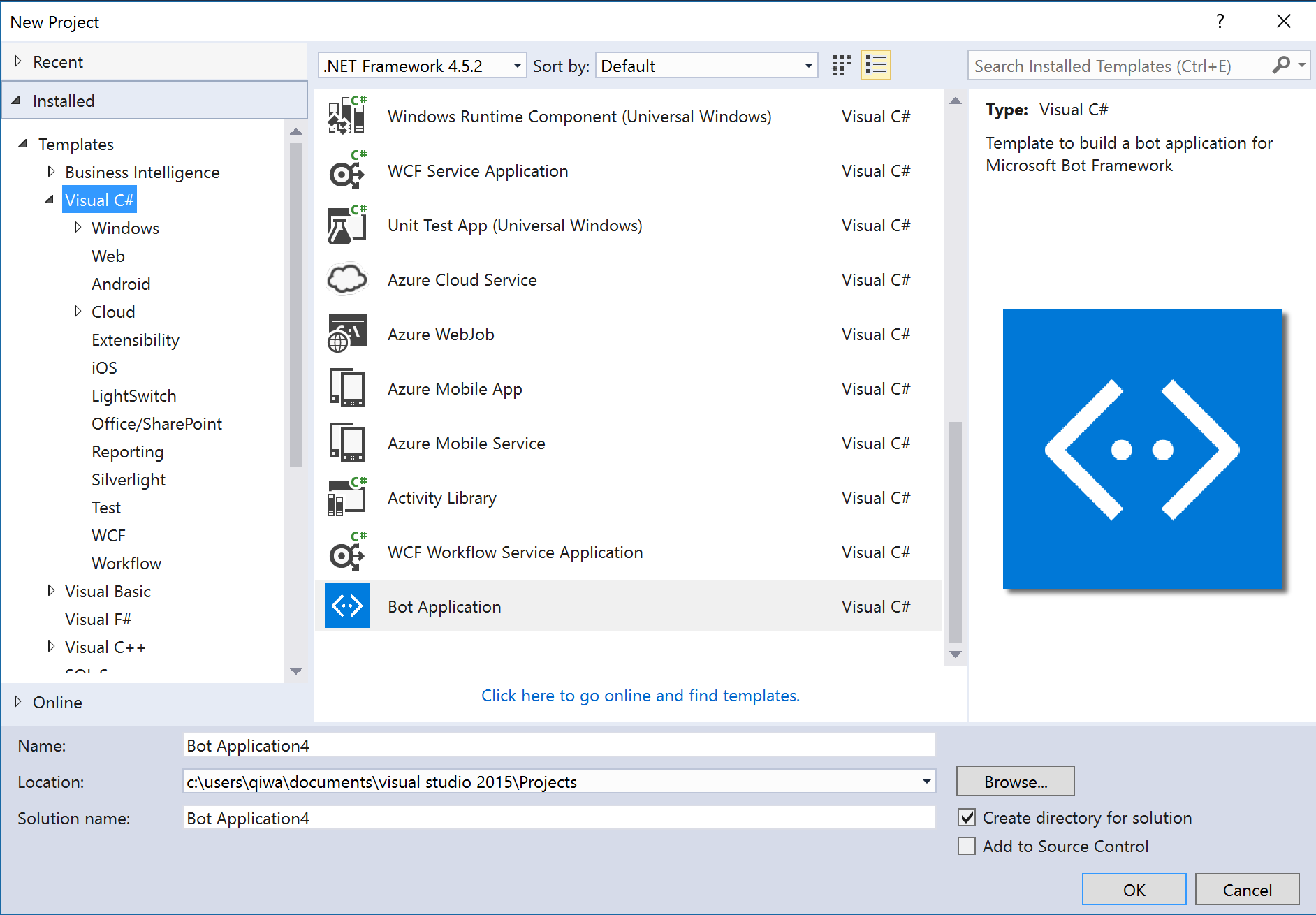
为了更方便开发，已经有人做了一个Bot Framework的开发模板。下载地址如下：

<http://aka.ms/bf-bc-vstemplate>

下载完成后，将Bot Application.zip复制到“%USERPROFILE%\Documents\Visual Studio 2015\Templates\ProjectTemplates\Visual C#"下（无需解压缩），如下图：

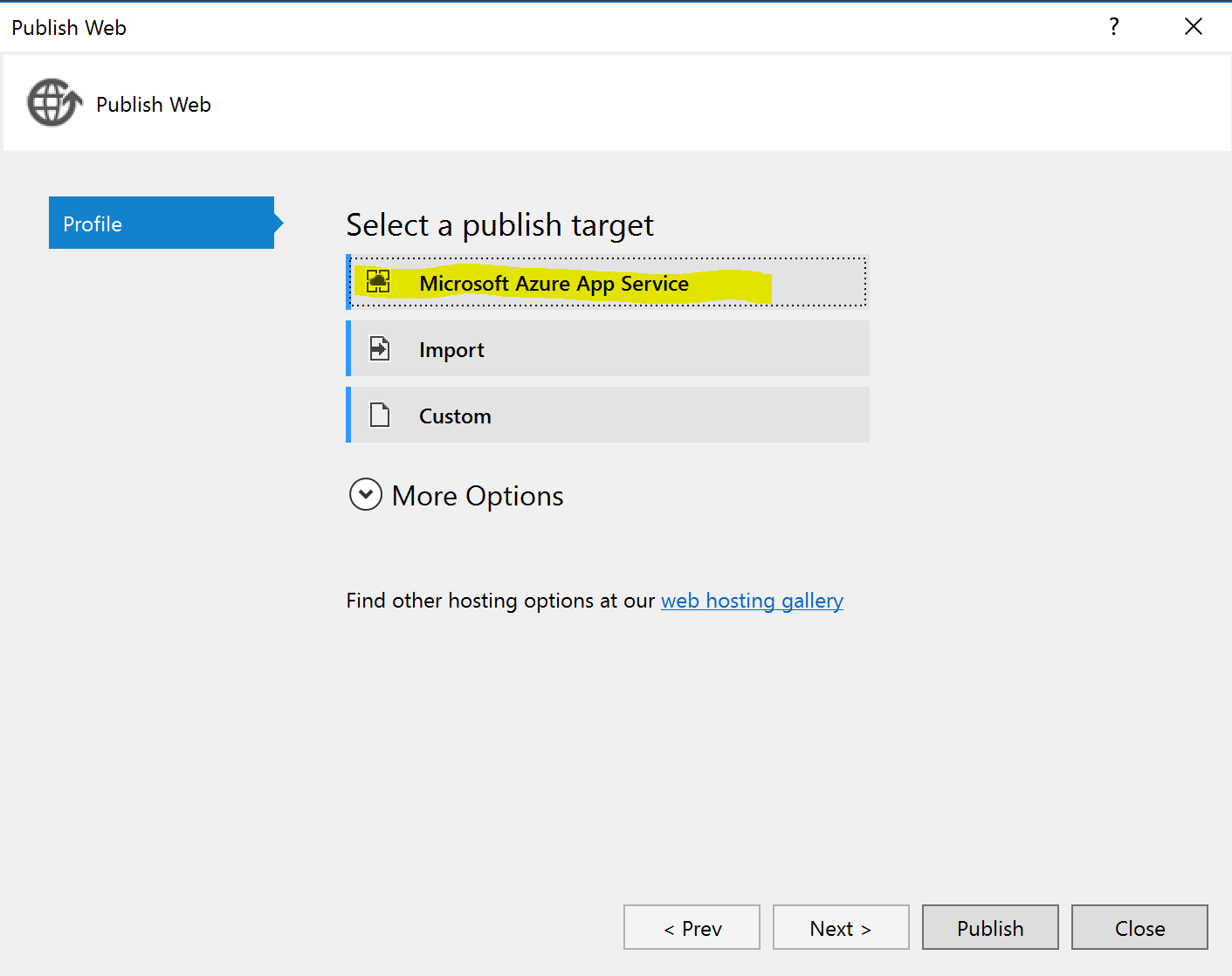


打开Visual Studio 2015，新建项目，在Visual C#项目下可以看到Bot Application项目类型，选中并点击OK，创建一个新的Bot 应用：

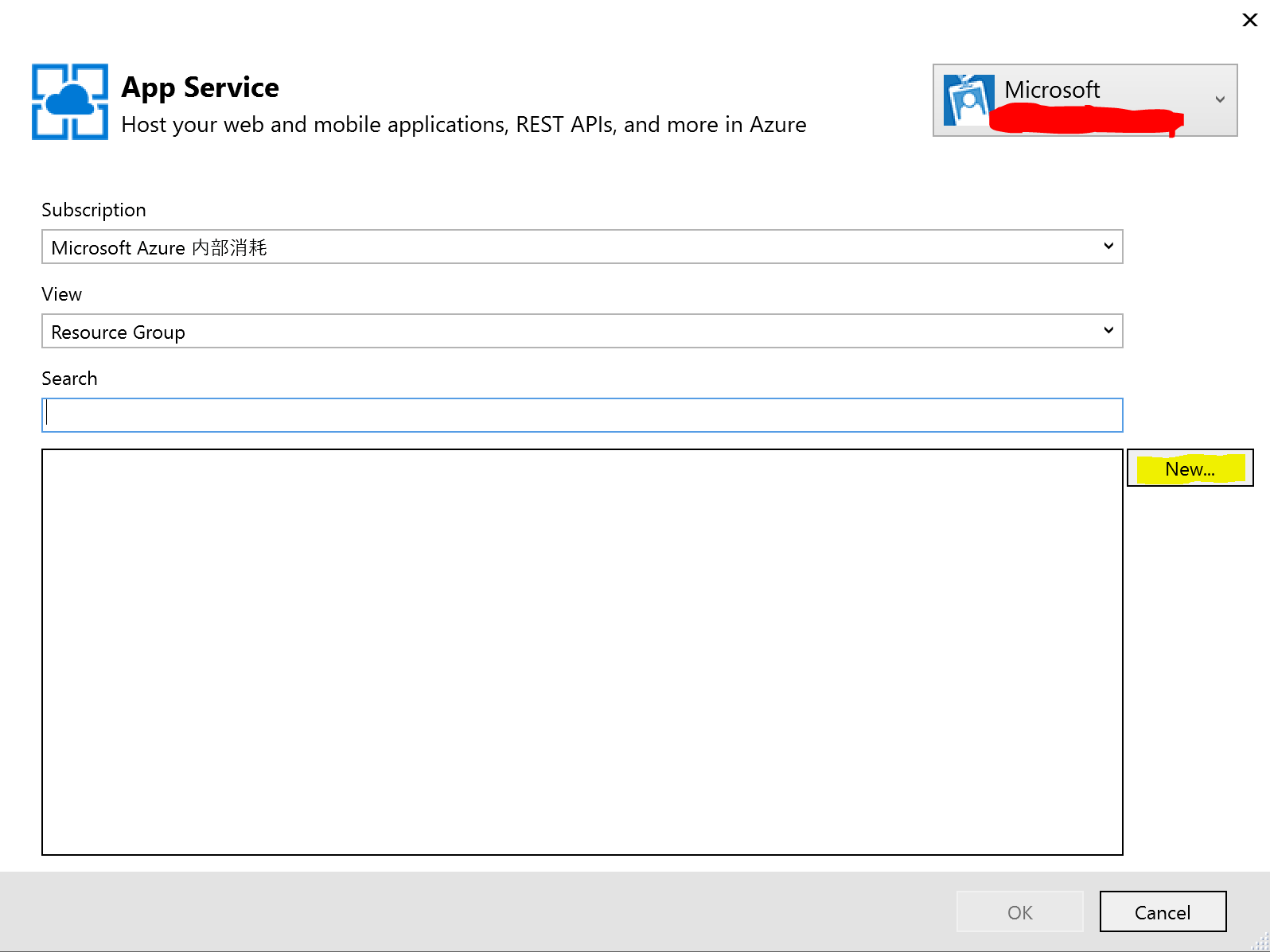


1. **创建后端服务**

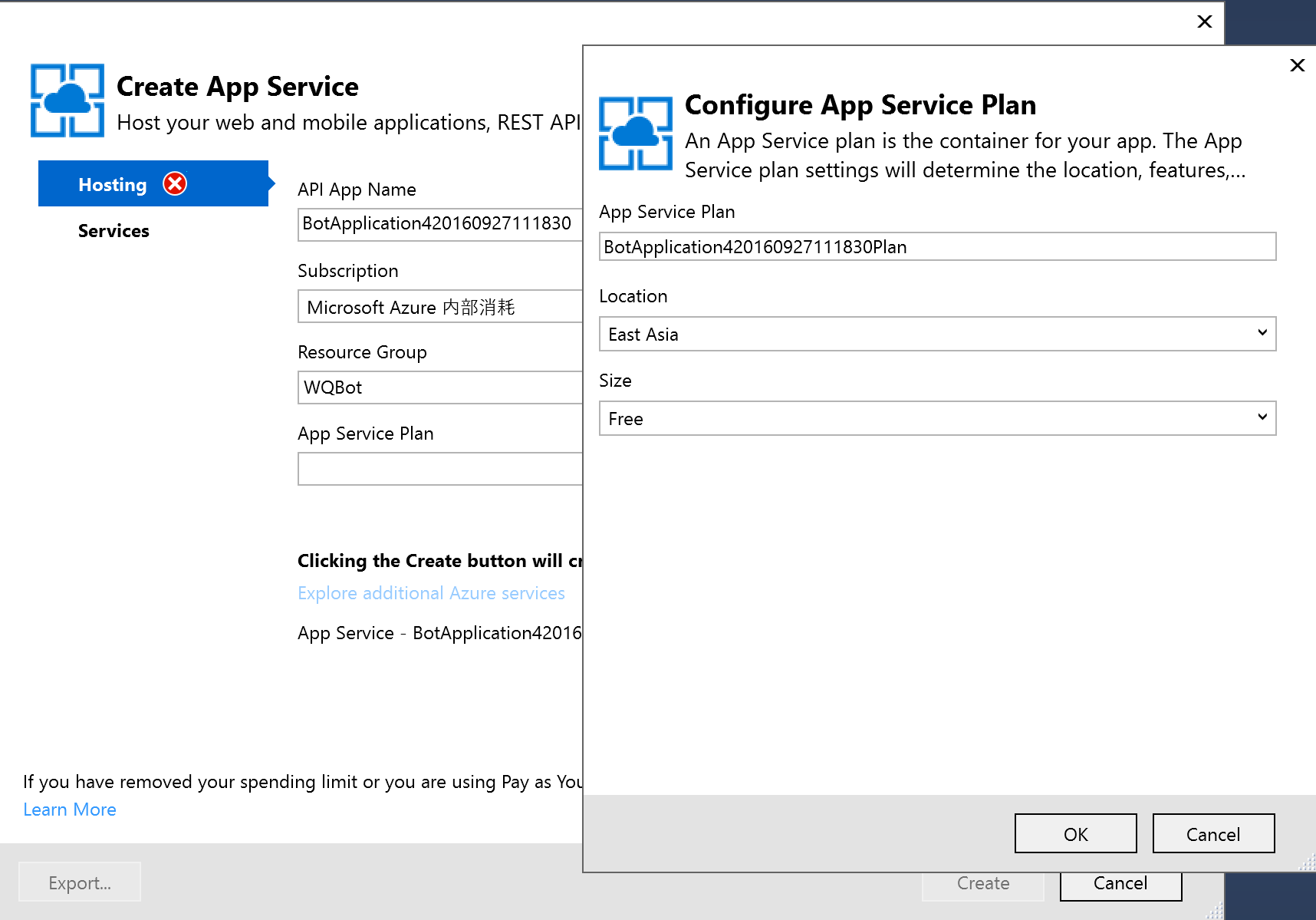
右键点击刚刚创建的网站，然后点击Publish将其发布到Azure上。选择Microsoft Azure App Service



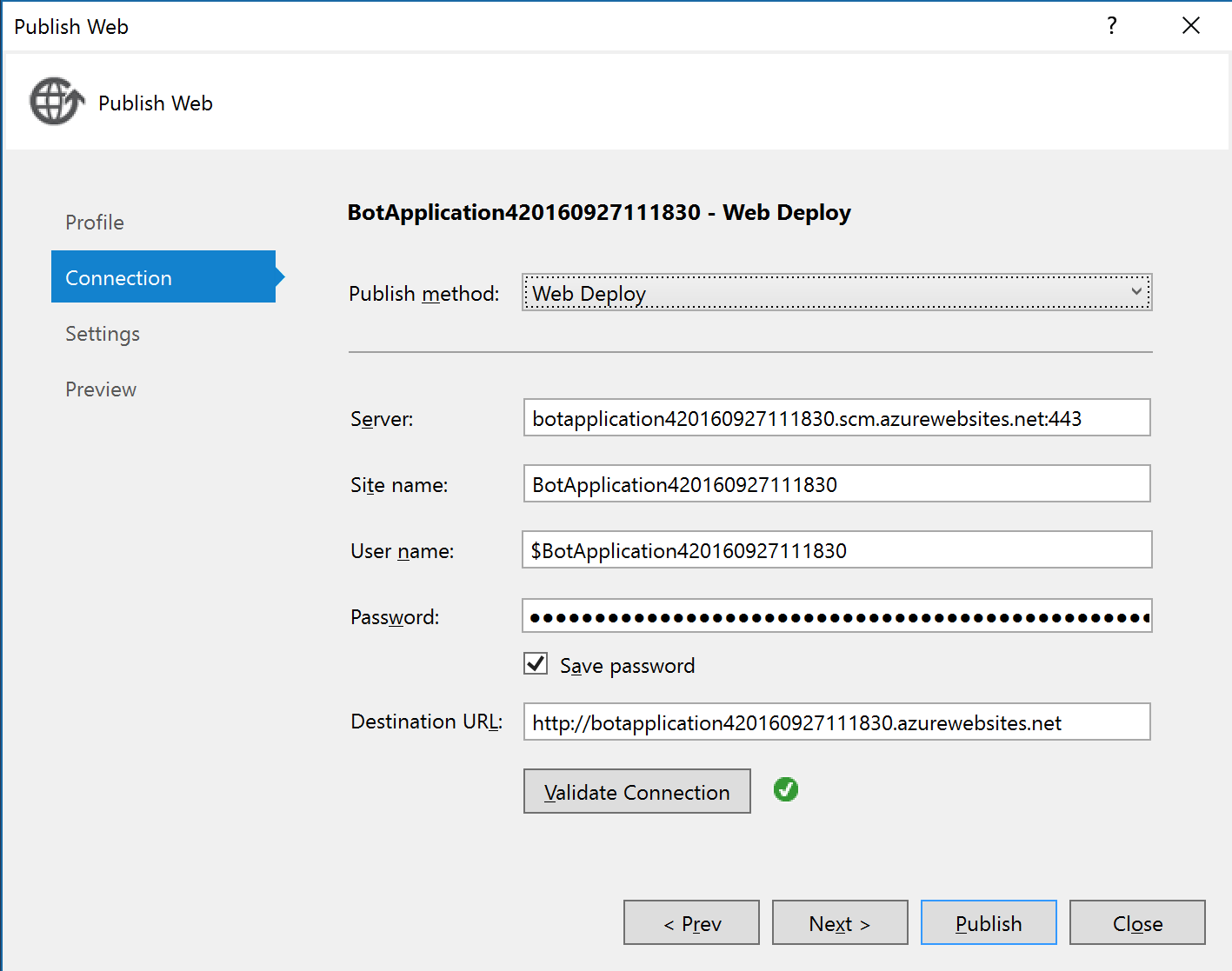
然后点击新建，如下图所示：



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



记下Destination URL，然后点击Publish



1. **在Bot Framework网站注册新创建的Bot**

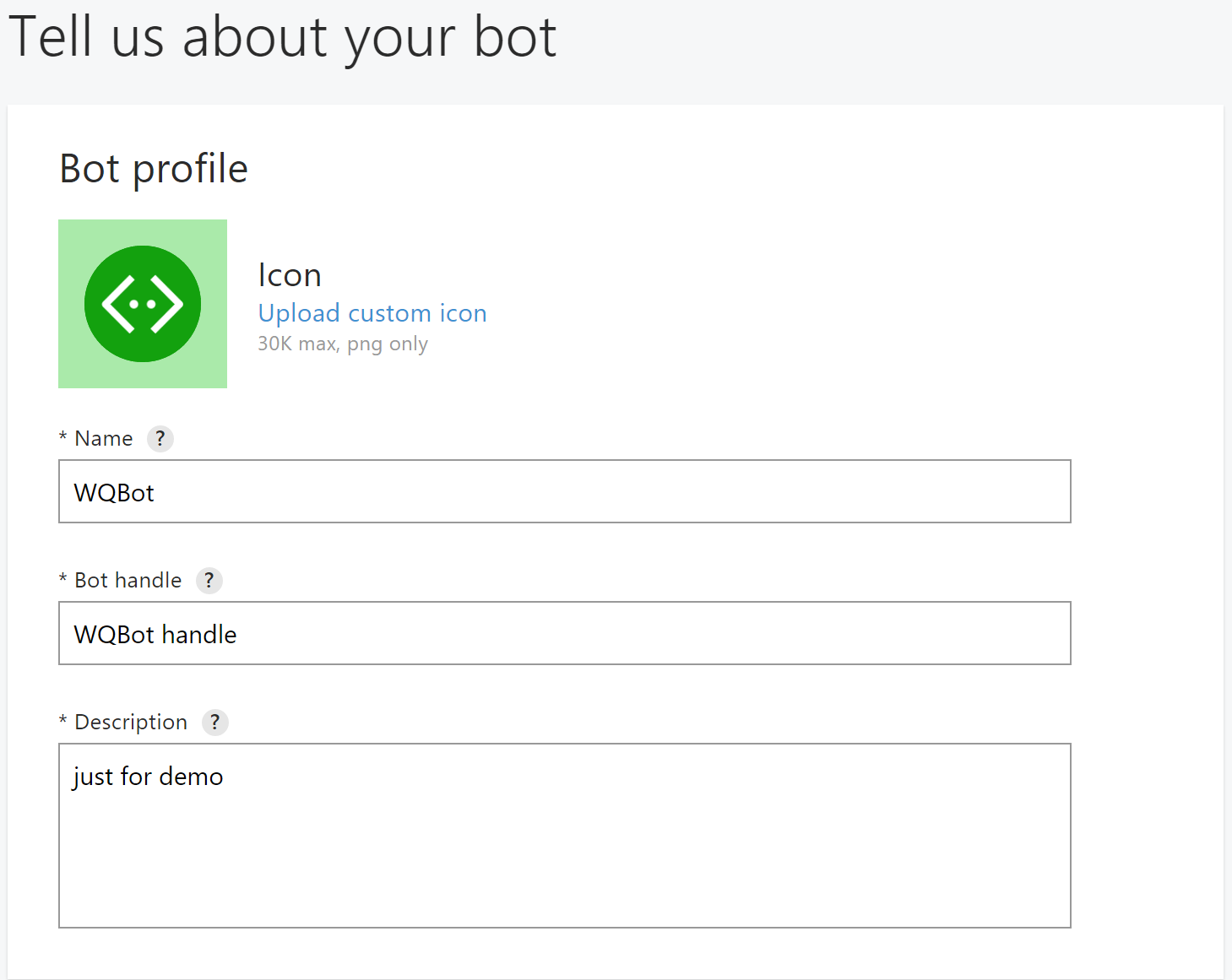
打开Bot Framework网站：<https://dev.botframework.com/>

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：

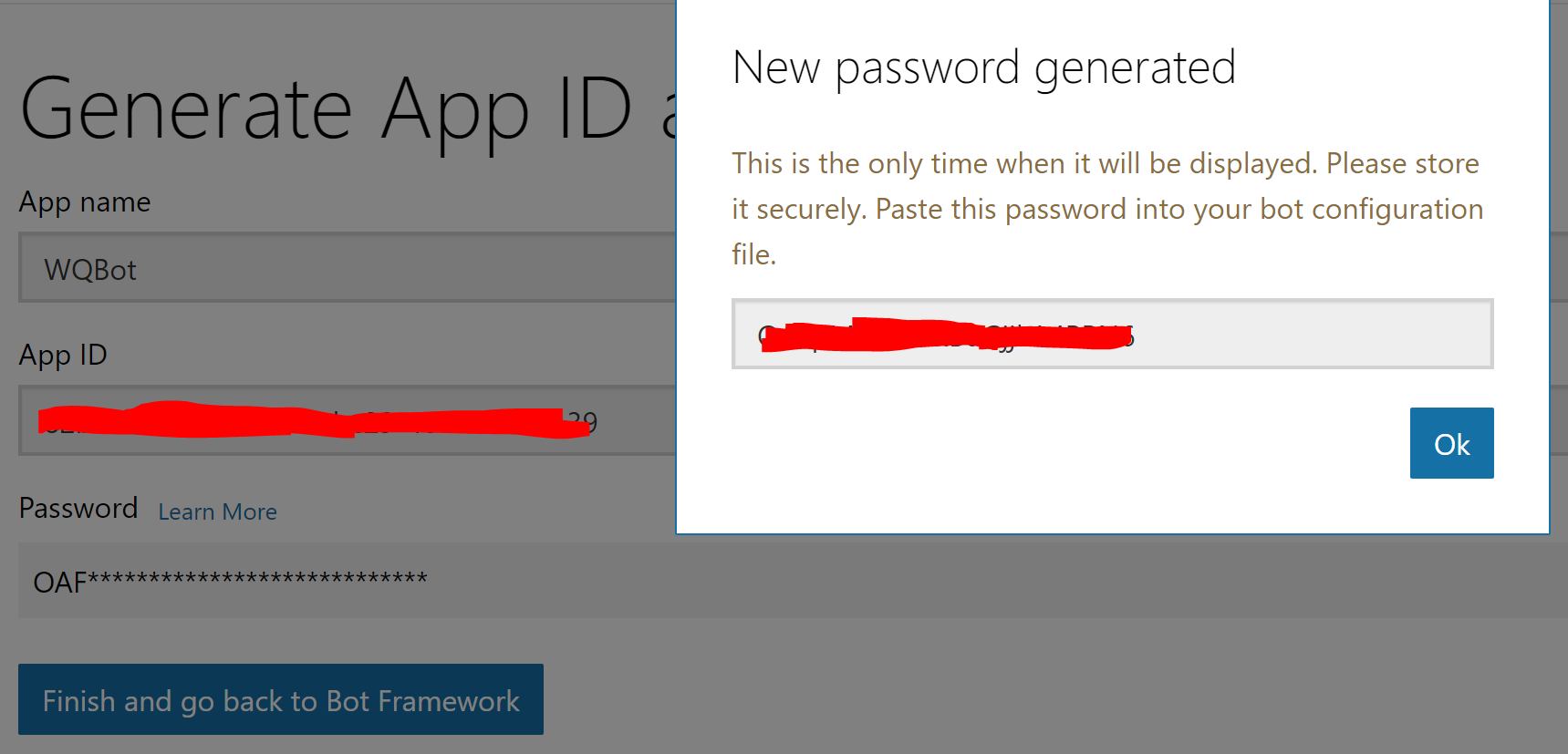
"Name"：你的bot的名字

"Bot Handle"：随便写一串字母，其实就是你的Bot的id

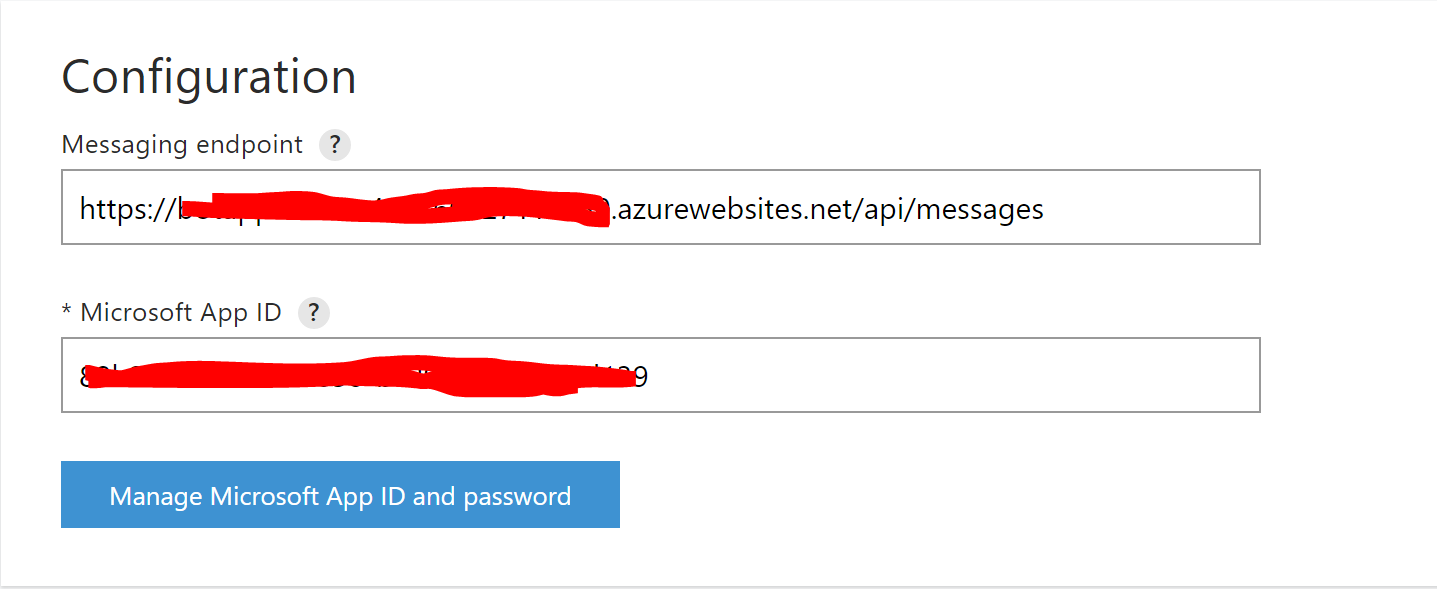
"Description"：你的Bot的描述，会在你的publish之后主页上显示。



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



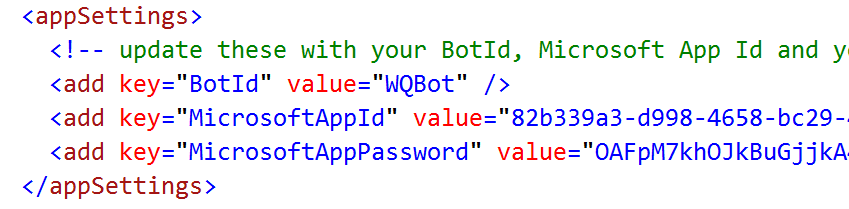
在Messaging endpoint中输入https://你的服务器地址/api/messages，如下图所示



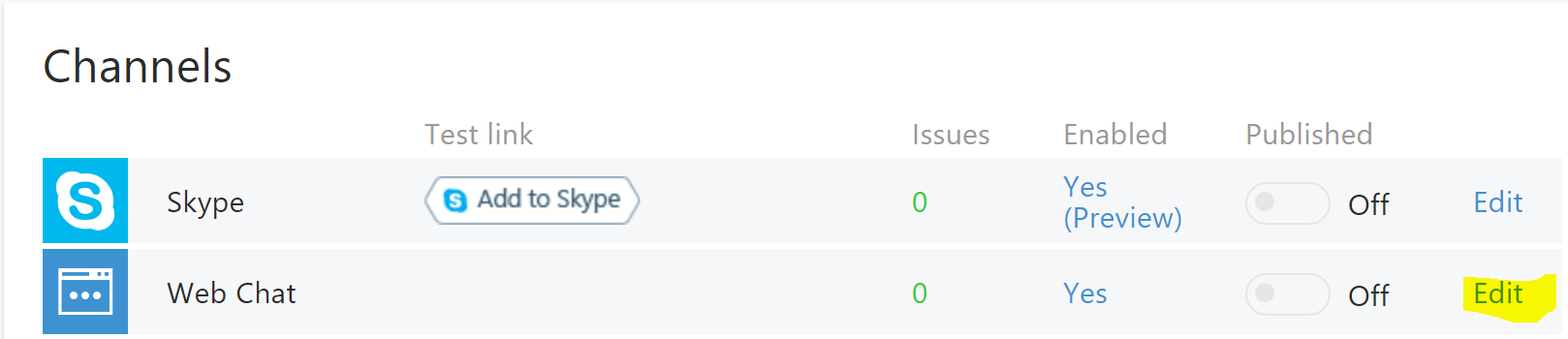
下面Publisher profile中的必填项随便填即可，然后点击Register，完成bot注册

1. **更新后端服务**

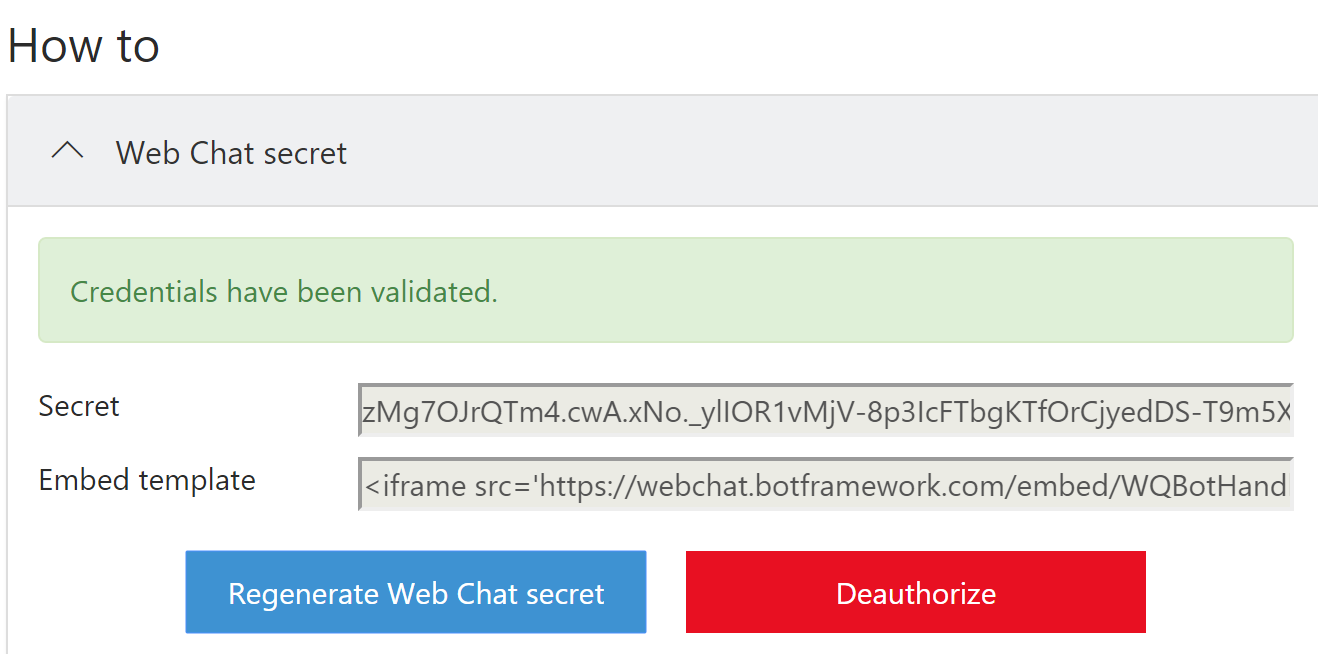
回到Visual Studio中，打开Web.config文件，填上你的botId，刚才创建的App ID和app password



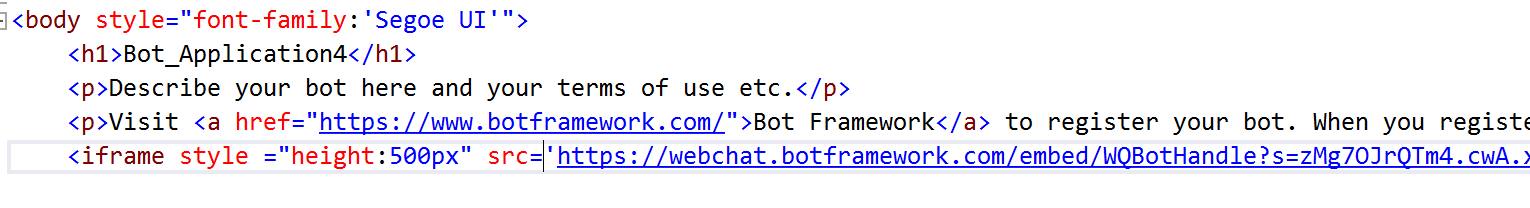
在bot framework网站中找到Web Chat，并点击Edit



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



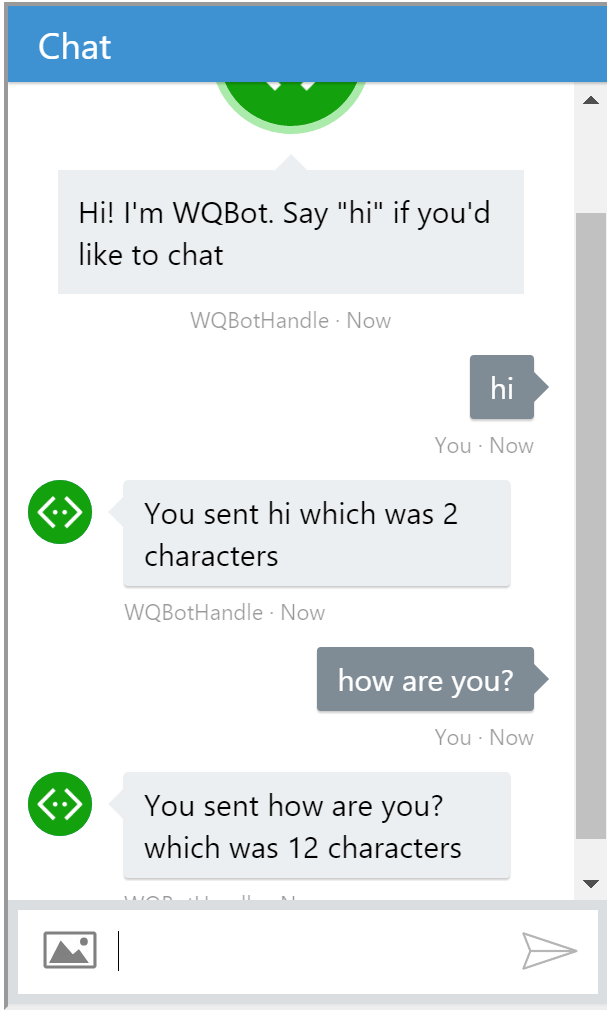
在iframe中添加height属性，并将Secret替换到链接中。打开Default.htm文件，将iframe嵌入到页面中



1. **发布并测试**

右键点击Web应用，点击Publish

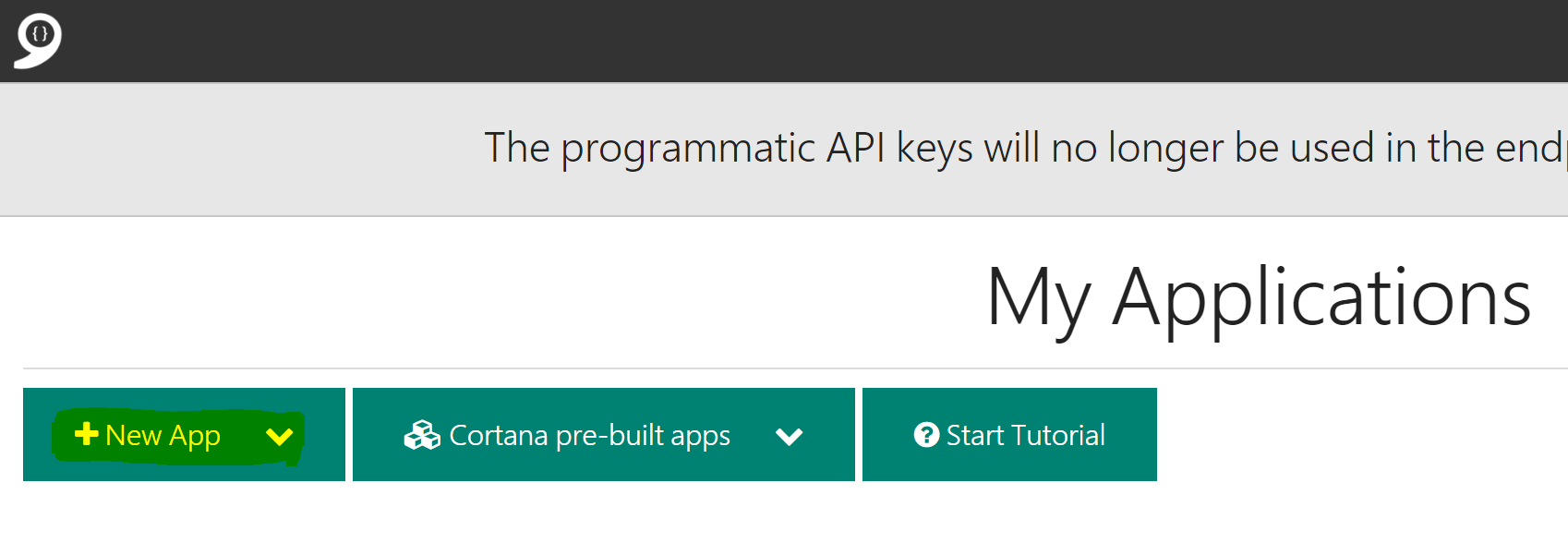
发布完成后，即可在网页中和bot进行交流



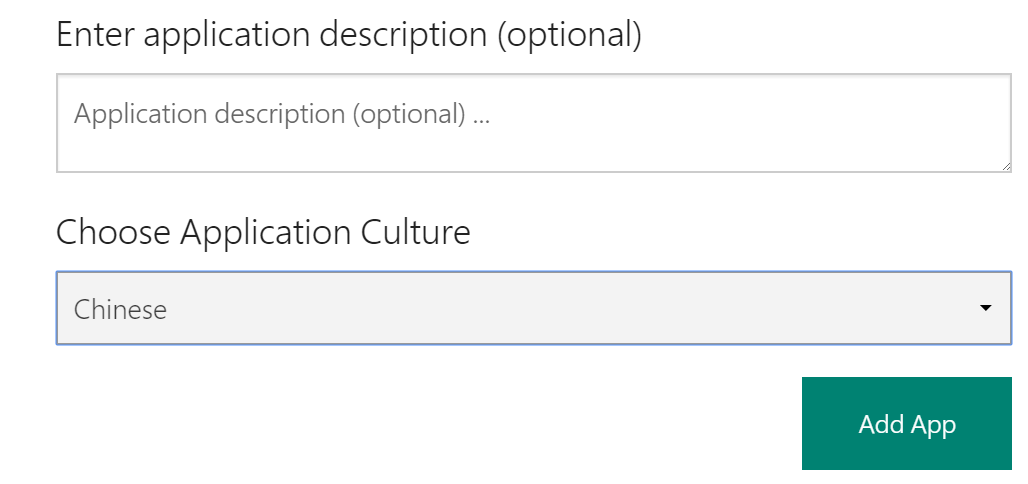
# 实验二：将Bot与LUIS结合

1. **新建LUIS服务**

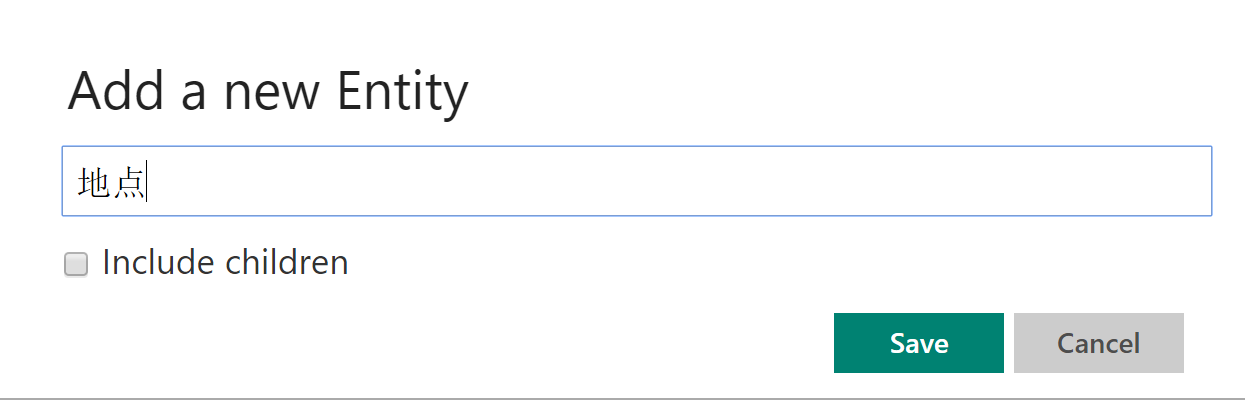
打开链接<https://www.luis.ai/> 并注册一个新账号，然后点击“New App”创建新应用：



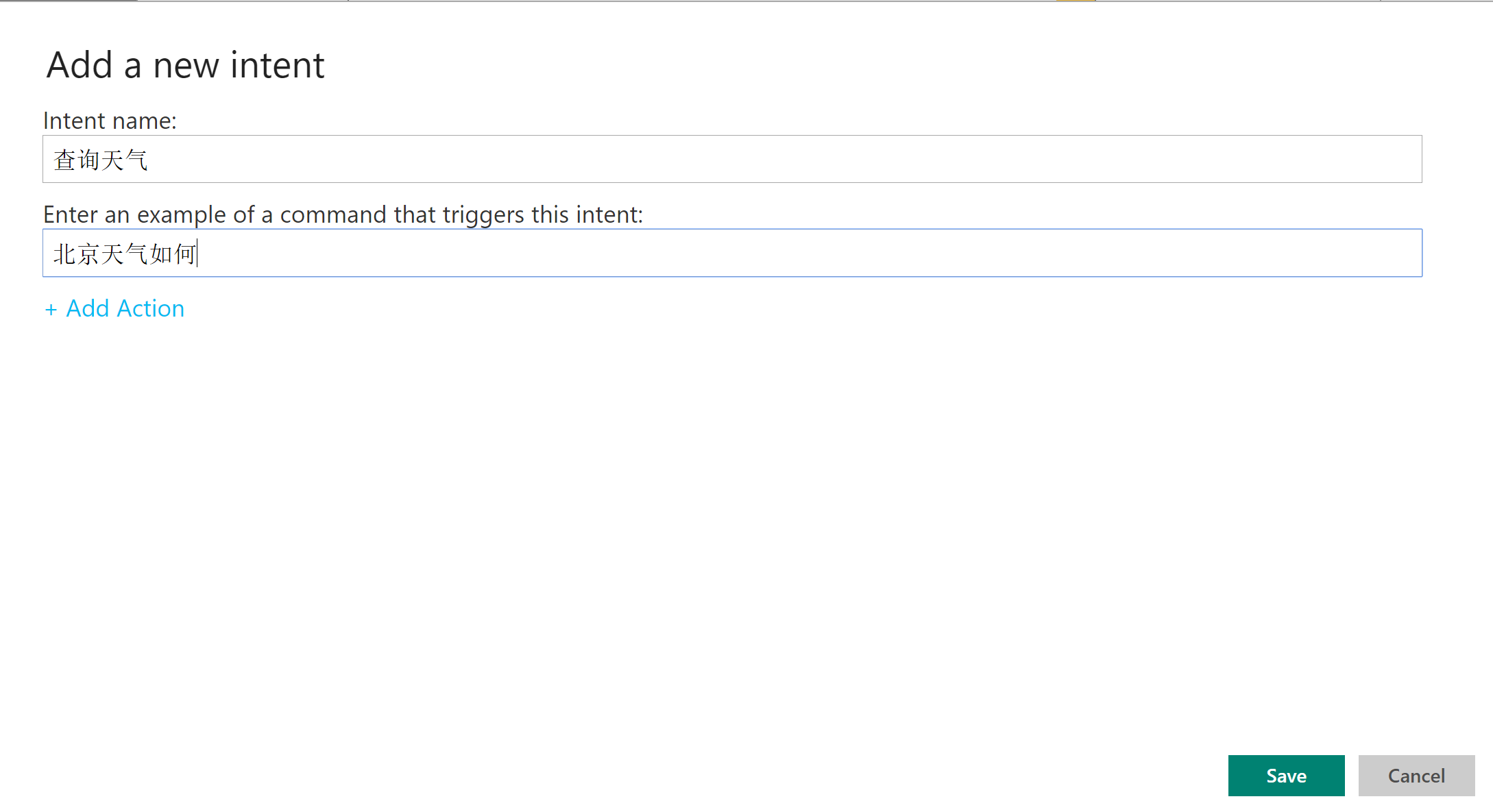
在“Add a new application”对话框中，输入应用程序名称、用途等基本信息，然后在“Choose Application Culture”中选择中文，然后点击“Add app”



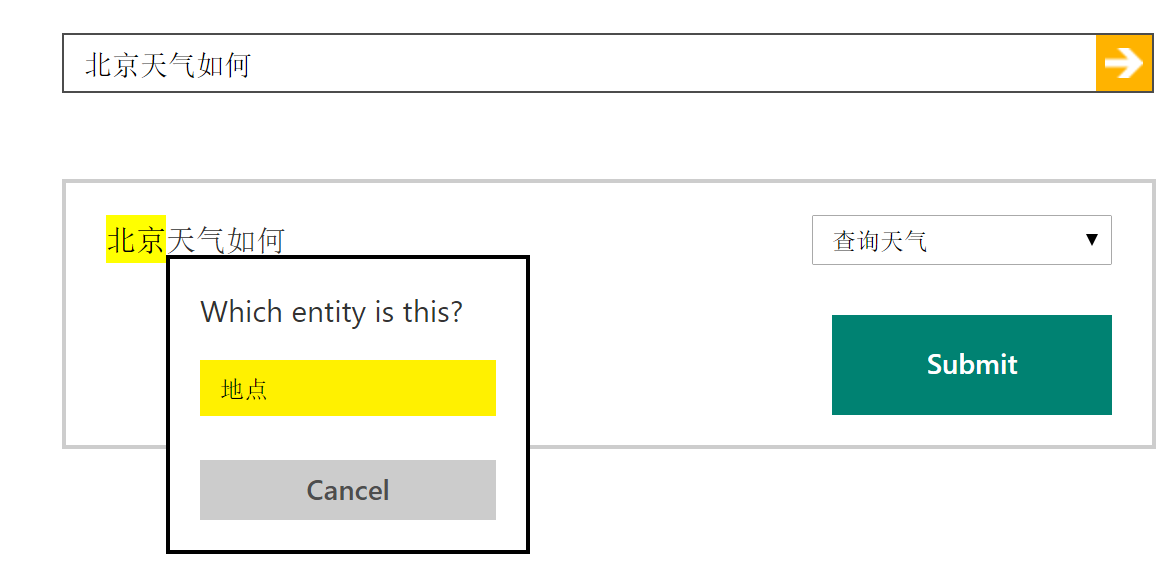
在界面左侧的Tab中，点击“Entities”旁边的+，并为新建的Entity起名为“地点”，然后点击Save，如下图



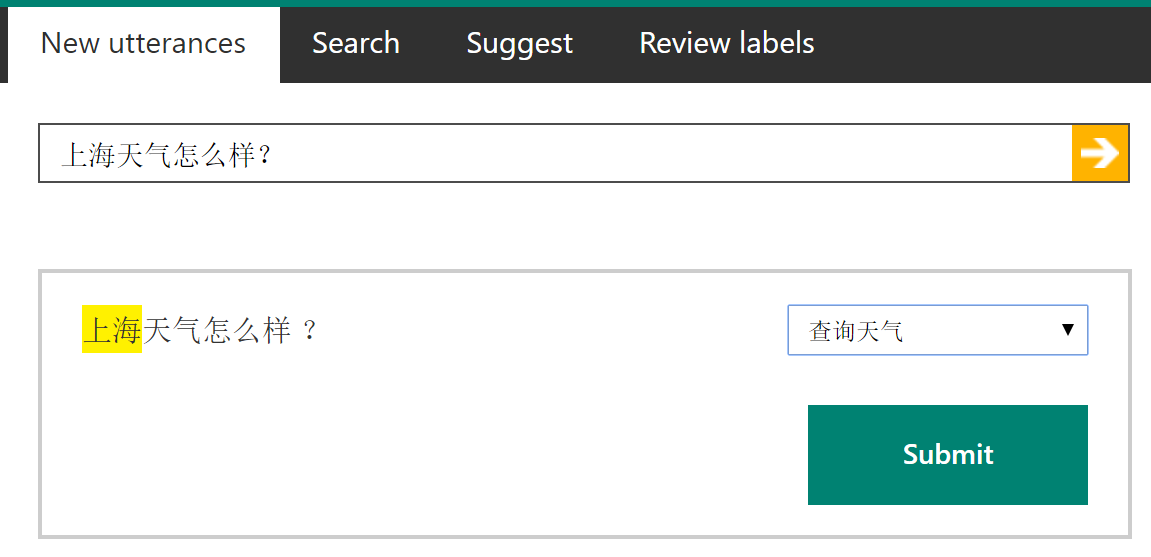
点击“Intents”旁边的+，在新建Intent界面中，输入“查询天气”作为Intent名称，然后输入例句，如下图所示：



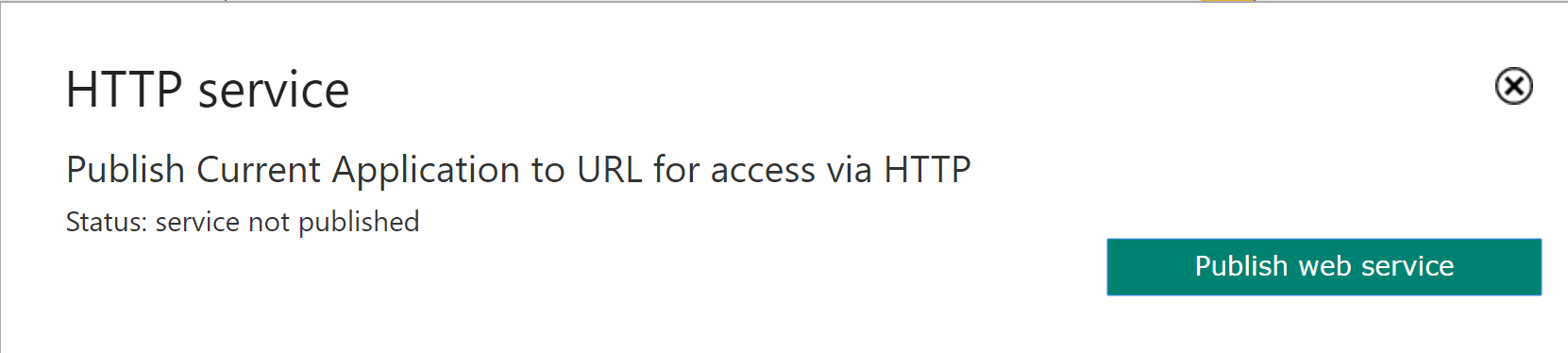
点击Save之后，用鼠标选择例句中的“北京”二字，并将其标注为“地点”，然后点击“Submit”，如下图所示：



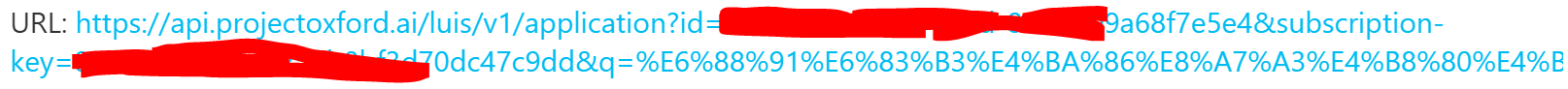
在“New utterances”标签下再输入一些查询天气的例句



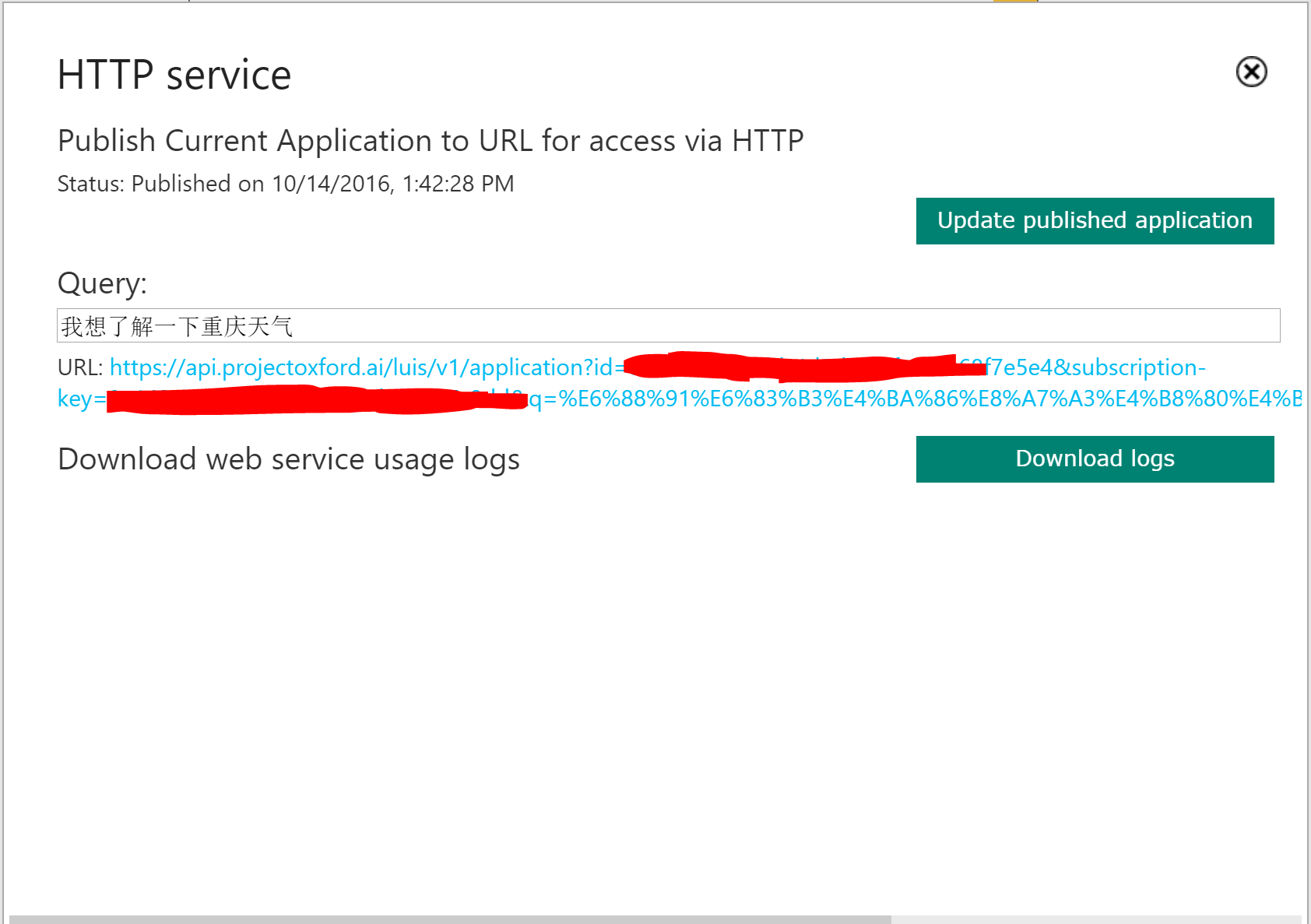
点击左下角的“Train”按钮

训练完成之后，点击左上角的“Publish”按钮，然后点击“Publish web service”“

将URL中的ID和subscription-key记下来，后面需要用到

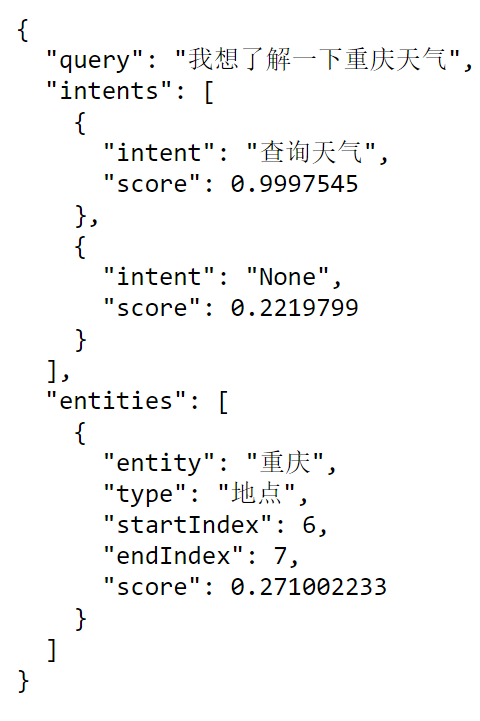


然后输入例句进行测试，并查看API调用后所返回的JSON信息，如下图所示：



备注：LUIS最近有更新，你遇到的URL可能是Version 2，类似下面这个：

URL https://api.projectoxford.ai/luis/v2.0/apps/LUIS\_ID?subscription-key=SUBSCRIPTION\_KEY



1. **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：

