



# 微软 BOT FRAMEWROK 技术白皮书

Microsoft DX

## 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

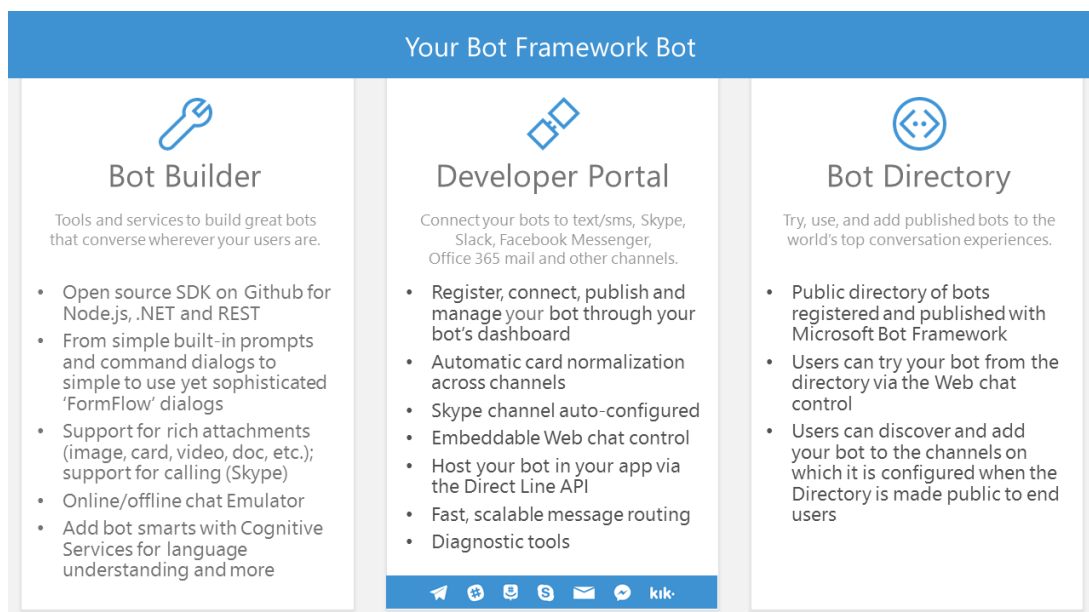
yuding@microsoft.com

## Contents

第一章 Bot Framework 概览.....	- 2 -
1. Bot Builder SDK .....	- 3 -
2. Bot Framework 开发者入口 .....	- 3 -
3. Bot Directory .....	- 4 -
第二章 使用.NET 开发 Bot .....	- 5 -
1. 安装必备程序.....	- 5 -
2. 编译你的 Bot.....	- 6 -
3. 使用 Bot 框架模拟器(Bot Framework Emulator )来测试你的 Bot 应用程序。 .....	- 8 -
第三章 将 Bot 与 LUIS 结合.....	- 10 -
1. 新建 LUIS 服务 .....	- 10 -
2. 在 Bot 中集成 LUIS 服务 .....	- 14 -
第四章 将 Bot 应用程序发布至微软 Azure.....	- 19 -
第五章 在微软 Bot 网站中注册您的 Bot .....	- 25 -
1. 测试您的 bot 的连接 .....	- 29 -
2. 配置通道.....	- 29 -
附录.....	- 32 -
参考资料： .....	- 32 -

# 第一章 Bot Framework 概览

Microsoft Bot Framework（微软对话机器人框架）是微软公司提供给开发者去开发高质量的对话机器人整体解决方案，让用户在自己的喜欢的聊天环境（如微信，skype）中直接体验与聊天机器人的交互。开发者在编写对话机器人的时候都会面临种种问题比如：机器人所需要的基础 I/O，语言选择和对话框的实现，如何编写高性能，响应及时并且扩展性强的机器人。同时，机器人如何与用户对接，尤其是在用户使用不同的聊天环境和语言的情况下。基于这种情况，微软对话机器人技术为开发者提供了一整套的解决方案去帮助开发者搭建，链接，管理和发布可以与用户自如的交流的智能机器人，不管你的用户来自 SMS 还是 Skype, Slack, Facebook Messenger, Kik, Office 365 邮件服务以及其他对话服务，比如国内的微信等。



Bot Framework 包括：Bot Builder SDK, 开发者页面 and Bot 目录 (Directory)。

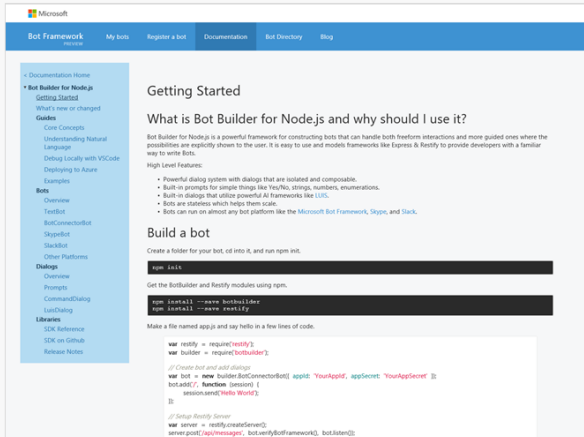
## 1. Bot Builder SDK

Bot Builder SDK 是一个在 Git 上开源的 SDK (<https://github.com/Microsoft/BotBuilder>), 支持基于 Node.js-, .NET- 或者 REST API 搭建的 bot。

### Bot Builder SDK

#### Node.js, .NET and REST

- Dialogs to model conversation
  - Dialogs are reusable
  - Types of Dialogs include:
    - Built-in prompts
    - Yes/No, String, Number, Choices
    - FormFlow and form slot filling (branching, disambiguation, multi-turn)
  - Conversations are scalable to multiple machines
- Rich interactions
  - Support for rich attachments (image, card, video, doc, etc.); support for calling (Skype)
  - Service extensions for language understanding (LUIS) and translation
- Online/offline Chat Emulator



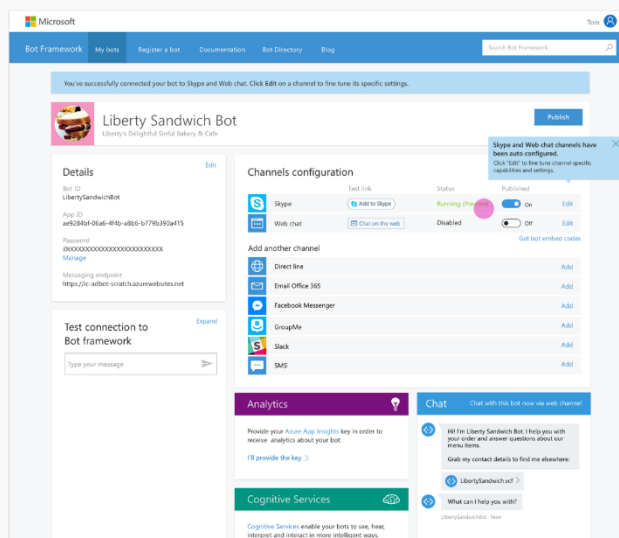
The screenshot shows the Microsoft Bot Framework documentation page for Node.js. The page has a blue header with the Microsoft logo and navigation links: 'Bot Framework', 'My bots', 'Register a bot', 'Documentation', 'Bot Directory', and 'Blog'. The left sidebar contains a table of contents with links to 'Documentation Home', 'Bot Builder for Node.js', 'Getting Started', 'Guides', 'Core Concepts', 'Understanding Natural Language', 'Debug Locally with VSCode', 'Deploying to Azure', 'Examples', 'Bots', 'Overview', 'TestBot', 'BotConnectorBot', 'SkypeBot', 'SlackBot', 'Other Platforms', 'Dialogs', 'Overview', 'Prompts', 'CommandDialog', 'TestDialog', 'Libraries', 'SDK Reference', 'SDK on GitHub', and 'Release Notes'. The main content area is titled 'Getting Started' and includes a section 'What is Bot Builder for Node.js and why should I use it?' with a list of features: 'Powerful dialog system with dialogs that are isolated and composable', 'Built-in prompts for simple things like Yes/No, strings, numbers, enumerations', 'Built-in dialogs that utilize powerful AI frameworks like LUIS', 'Bots are stateless which helps them scale', and 'Bots can run on almost any bot platform like the Microsoft Bot Framework, Skype, and Slack'. Below this is a 'Build a bot' section with instructions to create a folder and run npm init, followed by a code snippet for creating a bot and adding a dialog.

## 2. Bot Framework 开发者入口

Bot Framework 开发者框架帮助开发者将开发好的 Bot 与对话渠道进行连接比如：SMS, Skype, Slack, Facebook Messenger, Kik, Office 365 邮件服务以及其他对话服务。对于所有的连接，开发者只需要进行简单的注册，配置即可完成。对于所有注册的 Bot 机器人，微软都默认帮助开发者进行了与 Skype 和网页的连接。

## Developer Portal

- Register your bot
- Connect to channels
- Test
- Publish
- Manage
- Measure



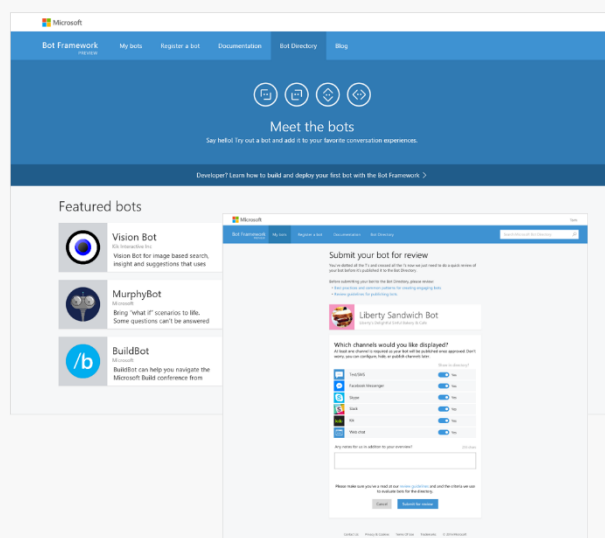
## 3. Bot Directory

Bot Directory (<https://bots.botframework.com/>) 是一个类似于手机端的 App Store 的 Bot 的总目录, 开发者需要在微软 bot 开发者入口注册并提交, 通过审核后 Bot 就可以出现在这个目录里面。普通用户可以直接添加目录的机器人。

## Bot Directory

### Public Directory of Bot Framework Bots

- Users can discover, try, and add bots to the conversation experiences on which the bot is configured (no app required)
- Bots are public at developer discretion; bots must be submitted for review in order to appear in the directory
- Searchable



## 第二章 使用.NET 开发 Bot

本章主要讲如何使用 Bot Framework Connector SDK 中的 .NET 模板进行开发。

### 1. 安装必备程序

- Visual Studio 2015（包含最新更新）- 你可以从此处免费下载社区版：

[www.visualstudio.com](http://www.visualstudio.com)

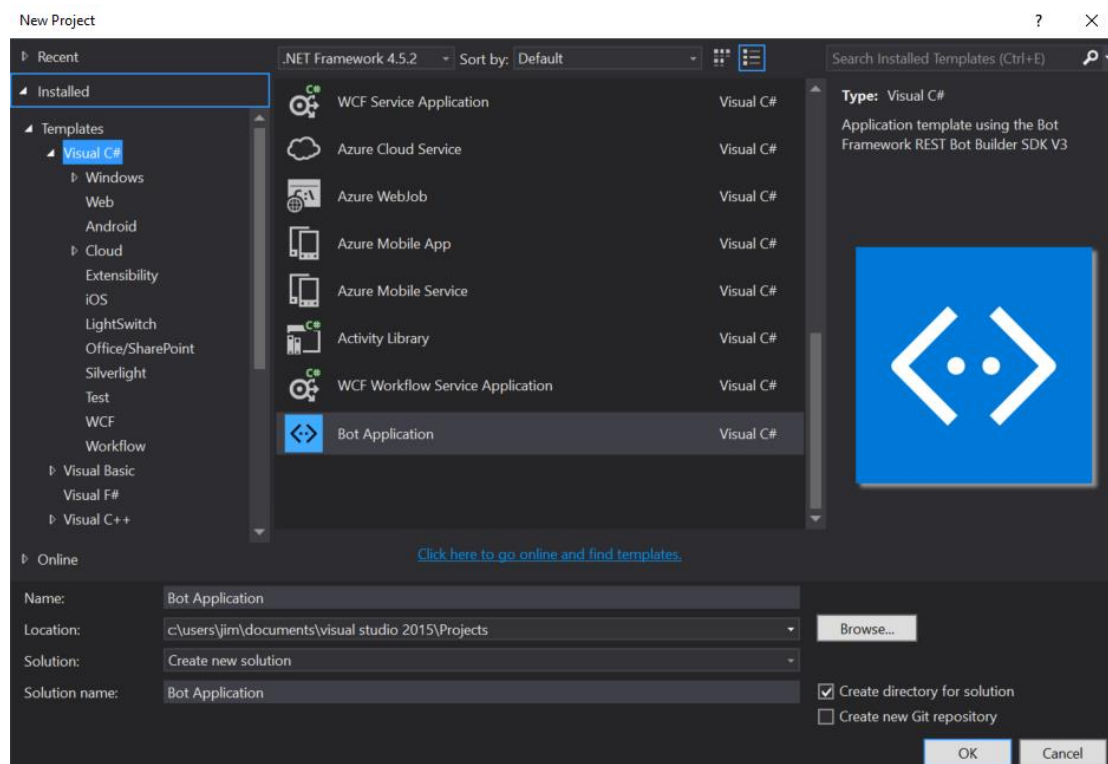
- 请确保所有的 VS 扩展都升级到最新版本：工具-扩展和更新-更新

#### 1. 下载并安装 Bot Application 模板

- 从此处下载所需文件：<http://aka.ms/bf-bc-vstemplate>
- 将下载的 zip 压缩包复制到 Visual Studio 2015 目录，一般在  
"%USERPROFILE%\Documents\Visual Studio 2015 \ Templates \  
ProjectTemplates \ Visual C#\".

#### 2. 打开 Visual Studio。

#### 3. 使用新的 Bot Application 模板创建一个新的 C#项目。



4. 该模板是一个完整功能的 Echo Bot，可以将用户输入的内容原样输出出来。

为了能正确运行：

- 该 Bot 应该使用 Bot Connector 进行注册
- 在 Bot Framework 注册页面获得的 AppId 和 AppPassword 要在项目的 web.config 中进行配置。
- 项目需要发布到服务器上。

## 2. 编译你的 Bot

Bot 模板的核心功能在 `Controllers\MessagesController.cs` 文件中，参考下面的代码。在本示例中，以下代码从用户输入取得文本信息，并使用 `CreateReplyMessage` 函数创建一个返回信息。方法上的 `BotAuthentication` 装饰器基于 HTTPS 来验证你的 Bot Connector 凭据。

[BotAuthentication]

**public class** MessagesController : ApiController

{

<summary>

POST: api/Messages

Receive a message **from** a user and reply to it

</summary>

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

{

ConnectorClient connector = **new** ConnectorClient(**new** Uri(activity.ServiceUrl));

**if** (activity.Type == ActivityTypes.Message)

{

// calculate something for us to return

**int** length = (activity.Text ?? **string**.Empty).Length;

// return our reply to the user

Activity reply = activity.CreateReply(\$"**You sent {activity.Text} which was {length}**

**characters**");

**await** connector.Conversations.ReplyToActivityAsync(reply);

}

**else**

{

HandleSystemMessage(activity);



```
}  
  
var response = Request.CreateResponse(HttpStatusCode.OK);  
  
return response;  
  
}  
  
}
```

### 3. 使用 Bot 框架模拟器(Bot Framework Emulator )来测试你的 Bot 应用程序。

Bot 框架提供了一个频道模拟器(channel emulator)来模拟 Bot 框架云服务测试调用你的 Bot。要安装 Bot 框架模拟器，请从[这儿](#)下载。

安装好 Bot 框架模拟器后你就可以测试了。首先，使用浏览器作为宿主运行前面步骤在 Visual Studio 中的 Bot 应用程序。下图是使用微软 Edge 浏览器运行：



## Bot\_Application

Describe your bot here and your terms of use etc.

Visit [Bot Framework](#) to register your bot. When you register it, remember to set your bot's endpoint to `https://your_bots_hostname/api/messages`

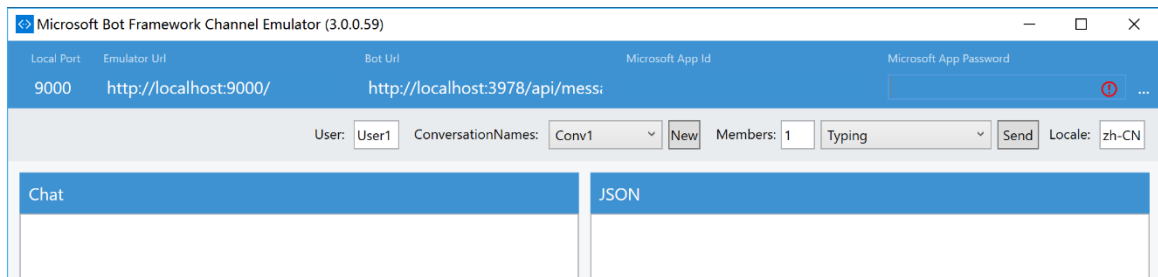
当使用模拟器来测试 Bot 应用程序时，需要记下程序运行的端口号，本例中的端口号是 3979。需要这个信息来运行 Bot 框架模拟器。

现在打开 Bot 框架模拟器。在能与 Bot 应用程序交互之前还需要做一些配置。

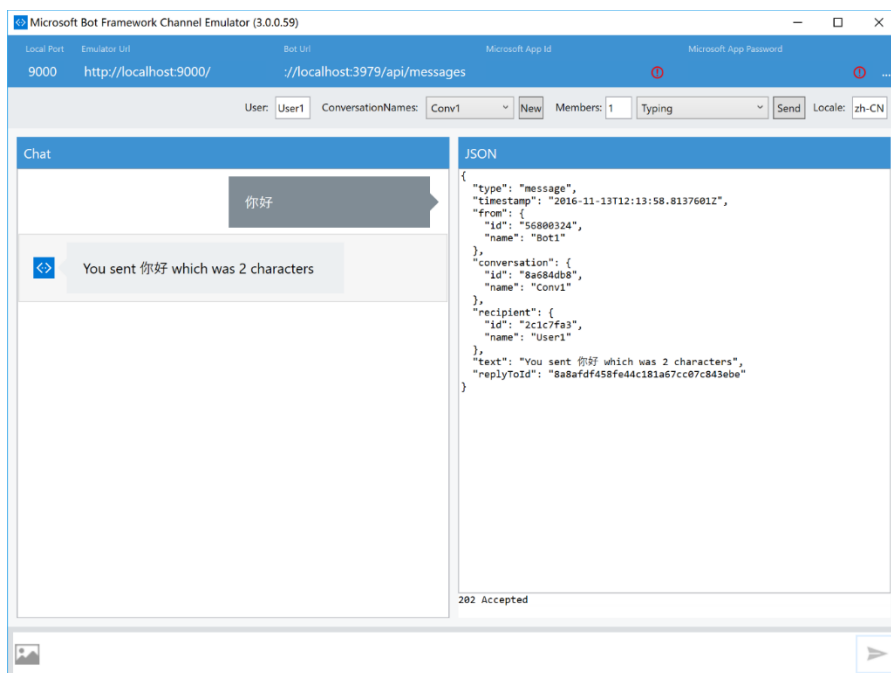
当使用模拟器配合本地运行的 Bot 时，需要做的配置如下：

- Bot 的 Url 设置为 localhost:<端口> , 其中<端口>为前面步骤记载下的端口。注意, 前面步骤使用的是 Bot 引用程序模板, Url 还需要添加“/api/messages”。
- MicrosoftAppId 字段不要填写, 保留空值就好
- MicrosoftAppPassword 字段不要填写, 保留空值就好。

这种方法仅适用于本地运行的模拟器。如果在云端运行的话, 你需要指定核实的 URL 核认证信息, 更多信息请参考[这里](#)。



现在所有万事俱备, 你可以和服务进行交互。在 Bot 框架模拟器引用的底部有个文本框, 你可以输入信息, 你输入的信息将会得到回应, 如下图所示:



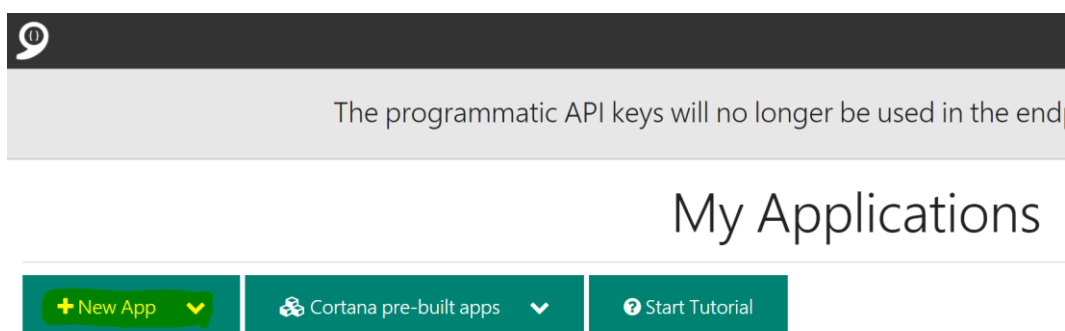
如果我们看看通过 Visual Studio 2015 Bot 应用程序模板产生的 Bot 应用程序代码，特别是名称为 MessageController.cs 这个文件，就可以看到用户输入的信息被转换成了一个回复活动，回复“You sent {activity.Text} which was {length} character”给用户。

## 第三章 将 Bot 与 LUIS 结合

微软认知服务中的语义理解服务—LUIS：即自然语言处理，是基于在线的大量语料库进行集成，可以提取出用户的意图。举个简单的例子，日常对话中很多人去打招呼，方式也各不相同：你好, hi, hello, 甚至是你吃了吗，传统的制作机器人的套路是以穷尽问题的方式去回答，让程序员在大量语料库中进行判断。微软提供的 LUIS 服务，将这些具体的问法归结为同一个意图“打招呼”，这使得我们的程序员就可以不需要考虑大量实际的问法，只是简单地针对于意图进行回复。我们可以将用户的需求通过意图提取跟我们的业务相互对应，使得用户可以通过简单对话的方式，完成之前可能需要打开网页，点击或者 touch 操作才能完成的人机交互。

### 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”

Enter application description (optional)

Application description (optional) ...

Choose Application Culture

Chinese

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

Add a new Entity

地点

☐ Include children

Save

Cancel

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：

## Add a new intent

Intent name:

Enter an example of a command that triggers this intent:

[+ Add Action](#)

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：

北京天气如何

Which entity is this?

地点

Cancel

查询天气

Submit

在“New utterances”标签下再输入一些查询天气的例句

New utterances

Search

Suggest

Review labels

上海天气怎么样?

→

上海天气怎么样?

查询天气 ▼

Submit

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”

HTTP service

ⓧ

Publish Current Application to URL for access via HTTP

Status: service not published

Publish web service

将 URL 中的 ID 和 subscription-key 记下来，后面需要用到

URL: [https://api.projectoxford.ai/luis/v1/application?id=\[redacted\]&subscription-key=\[redacted\]](https://api.projectoxford.ai/luis/v1/application?id=[redacted]&subscription-key=[redacted])

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：

示：

HTTP service

Publish Current Application to URL for access via HTTP  
Status: Published on 10/14/2016, 1:42:28 PM

Update published application

Query:  

我想了解一下重庆天气

URL: https://api.projectoxford.ai/luis/v1/application?id=[redacted]&subscription-key=[redacted]  
q=%E6%88%91%E6%83%B3%E4%BA%86%E8%A7%A3%E4%B8%80%E4%

Download web service usage logs

Download logs

```
{
  "query": "我想了解一下重庆天气",
  "intents": [
    {
      "intent": "查询天气",
      "score": 0.9997545
    },
    {
      "intent": "None",
      "score": 0.2219799
    }
  ],
  "entities": [
    {
      "entity": "重庆",
      "type": "地点",
      "startIndex": 6,
      "endIndex": 7,
      "score": 0.271002233
    }
  ]
}
```

## 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 xn1 in xnList)
    {
        if (xn1.GetAttribute("name").ToString() ==
location)
            weathercode =
xn1.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("");
}
}
}
}

```

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：



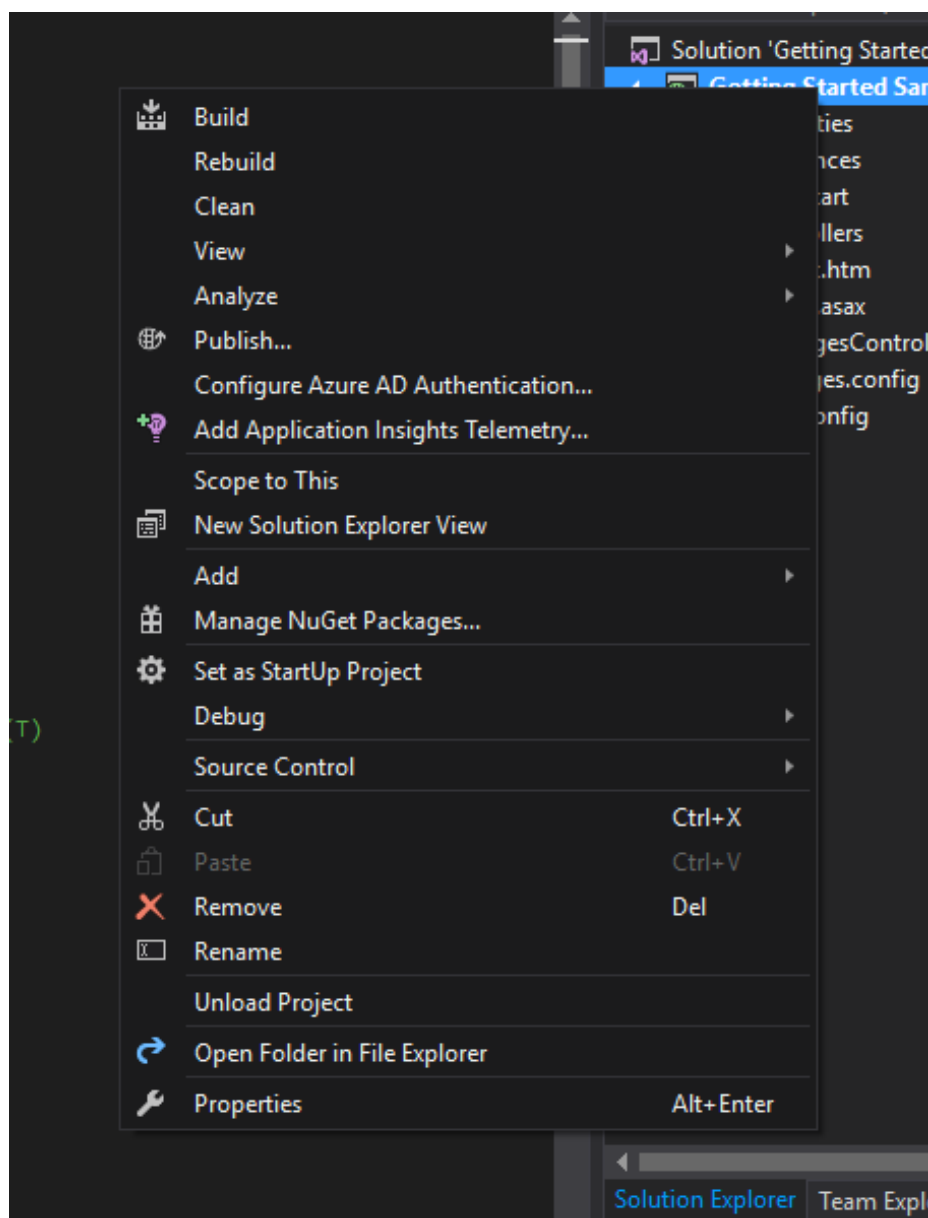
## 第四章 将 Bot 应用程序发布至微软 Azure

在本教程中，我们使用 微软 Azure 托管 Bot 应用程序。要发布 Bot 应用程序，您需要一个 Microsoft Azure 订阅。您可免费试

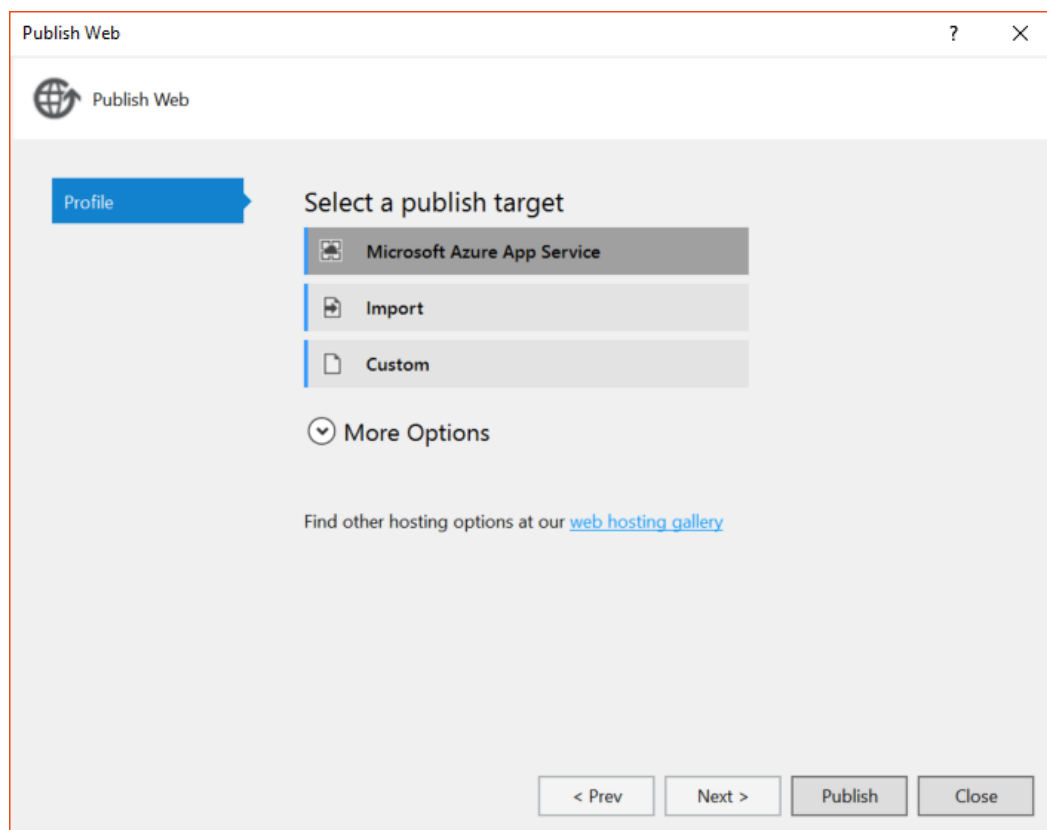
用：[azure.microsoft.com/en-us/](https://azure.microsoft.com/en-us/)

您可对项目做出您喜欢的修改然后发布。右键单击项目并选择“发布”，然后选择 Azure 订阅。默认情况下，bot 作为微软 Azure 应用程序服务发布。发布时，跟踪您选择的 URL，因为我们需要用它来更新 Bot 框架注册端点。第一次发布时有几个额外的步骤，但你只需做一次。

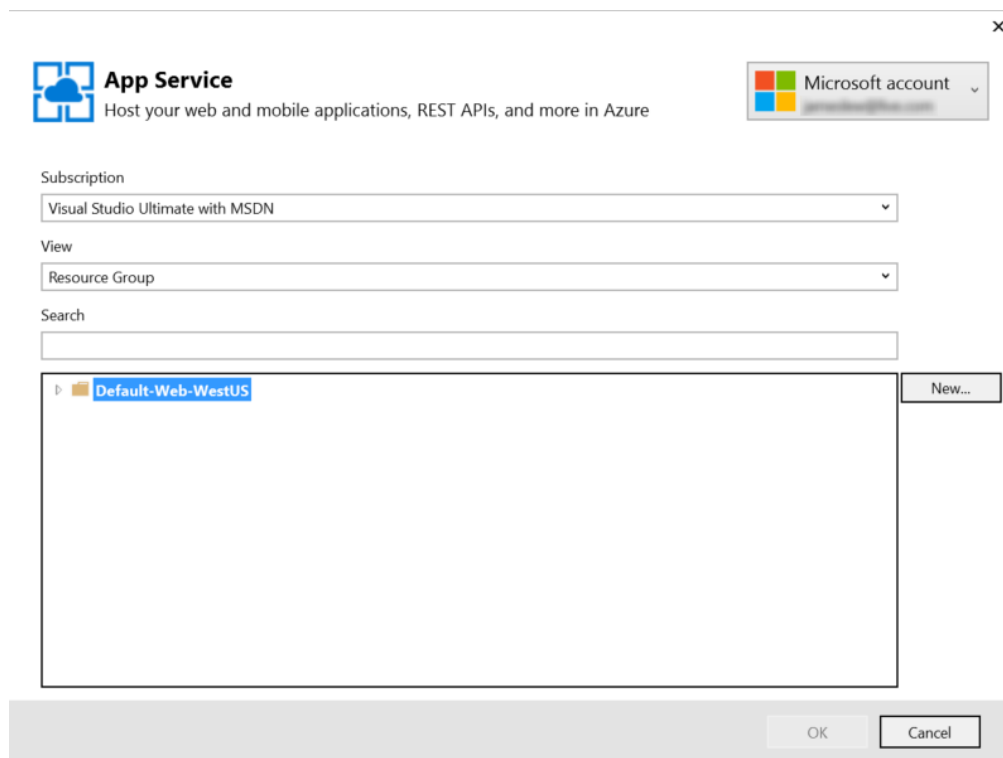
在 Visual Studio 中，右键单击解决方案资源管理器中的项目，然后选择“发布”——或者选择“构建|发布”，将显示以下对话框：



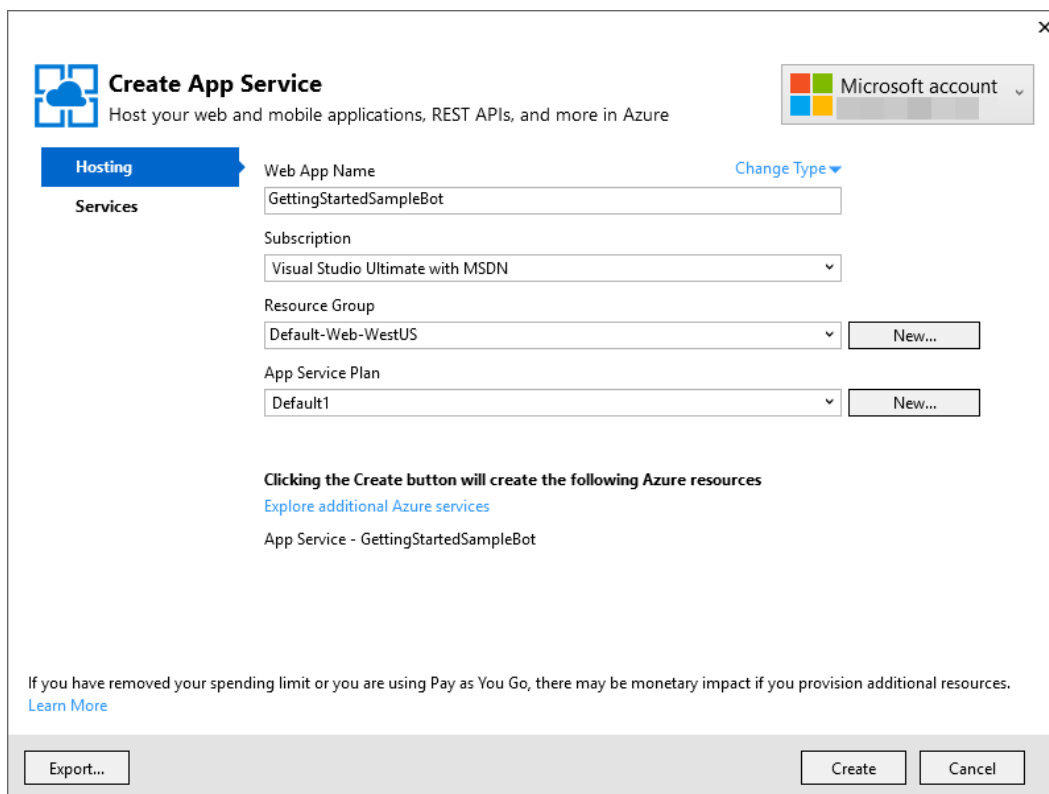
发布到 Azure 向导将启动。对于本教程，您需要选择“Microsoft Azure 应用服务”  
(Microsoft Azure App Service) 作为您的项目类型。



Azure 应用程序服务发布过程的下一步是创建应用程序服务。单击对话框右侧的“新建...”（New...）以创建应用程序服务。



然后将显示“创建应用程序服务”对话框，根据需要填写详细信息。在右上角的“更改类型”（Change Type）下拉菜单中选择“Web 应用程序”（Web App），而不是“API 应用程序”（API App 这是默认值）。



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

Microsoft account

**Hosting**  
**Services**

Web App Name [Change Type](#)  
GettingStartedSampleBot

Subscription  
Visual Studio Ultimate with MSDN

Resource Group  
Default-Web-WestUS [New...](#)

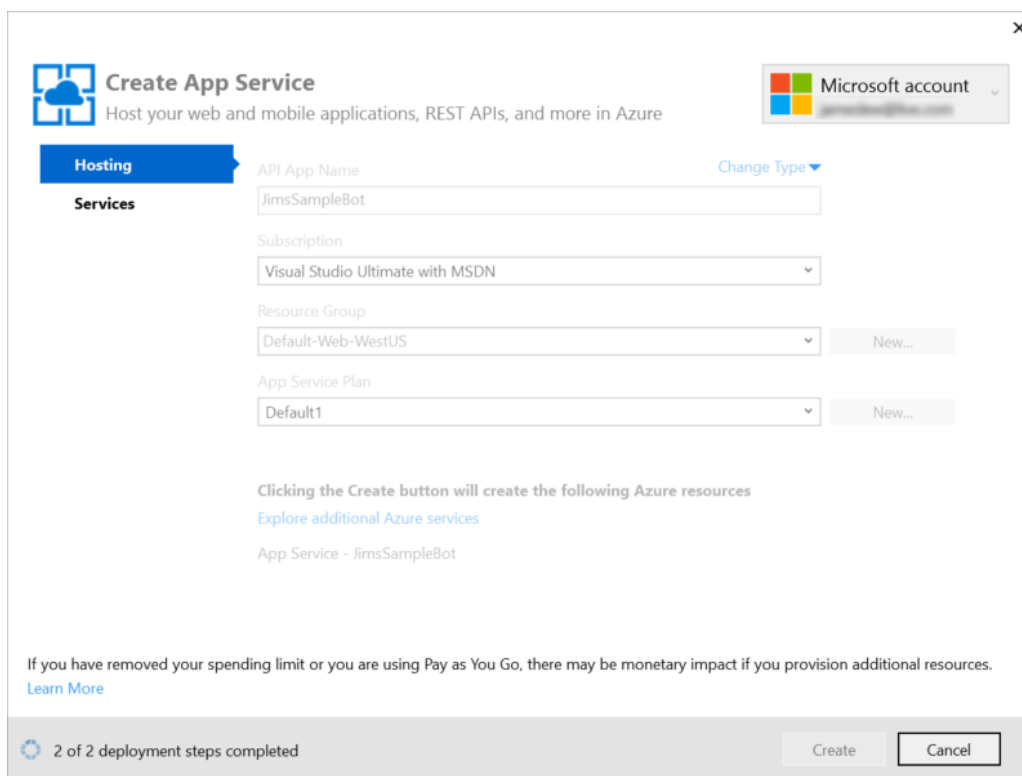
App Service Plan  
Default1 [New...](#)

Clicking the Create button will create the following Azure resources  
[Explore additional Azure services](#)  
App Service - GettingStartedSampleBot

If you have removed your spending limit or you are using Pay as You Go, there may be monetary impact if you provision additional resources.  
[Learn More](#)

[Export...](#) [Create](#) [Cancel](#)

此对话框的最后一个复杂的配置是应用程序服务计划。这里只是让您为位置和系统大小的组合提供一个名称，以便您在将来的部署中重新使用它。您只要输入任何名称，然后选择数据中心和您想要的部署大小。



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

Microsoft account

**Hosting**  
**Services**

API App Name: JimsSampleBot [Change Type](#)

Subscription: Visual Studio Ultimate with MSDN

Resource Group: Default-Web-WestUS [New...](#)

App Service Plan: Default1 [New...](#)

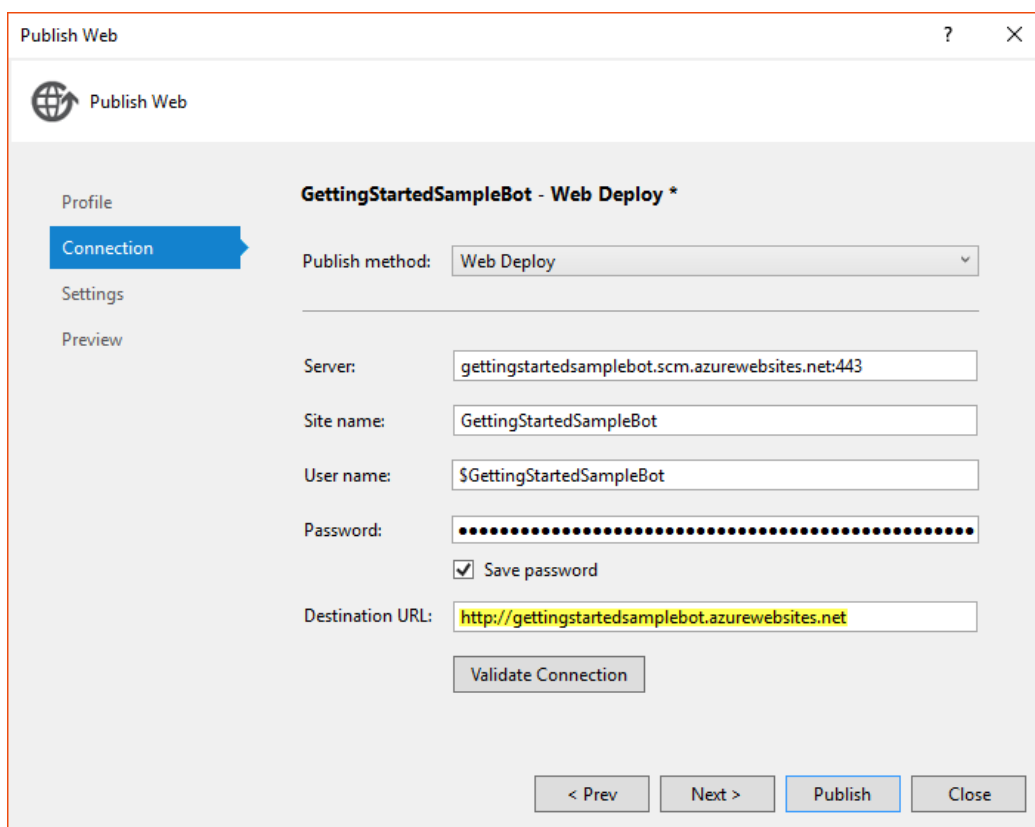
Clicking the Create button will create the following Azure resources  
[Explore additional Azure services](#)  
App Service - JimsSampleBot

If you have removed your spending limit or you are using Pay as You Go, there may be monetary impact if you provision additional resources.  
[Learn More](#)

2 of 2 deployment steps completed

Create Cancel

一旦在应用服务计划中点击“OK”，您就完成了定义应用服务的任务。点击创建，您将被带回到发布 Web 向导。



**Publish Web**

**GettingStartedSampleBot - Web Deploy \***

Profile  
**Connection**  
Settings  
Preview

Publish method: Web Deploy

Server: gettingstartedsamplebot.scm.azurewebsites.net:443

Site name: GettingStartedSampleBot

User name: \$GettingStartedSampleBot

Password:

☒ Save password

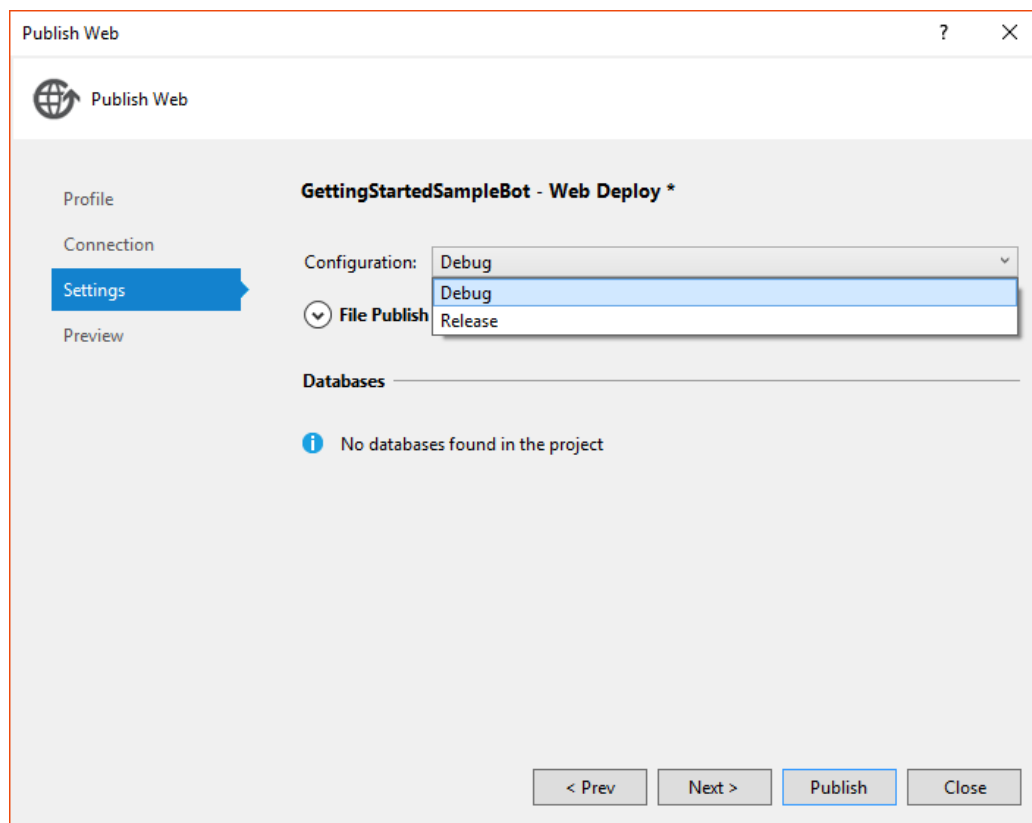
Destination URL: <http://gettingstartedsamplebot.azurewebsites.net>

[Validate Connection](#)

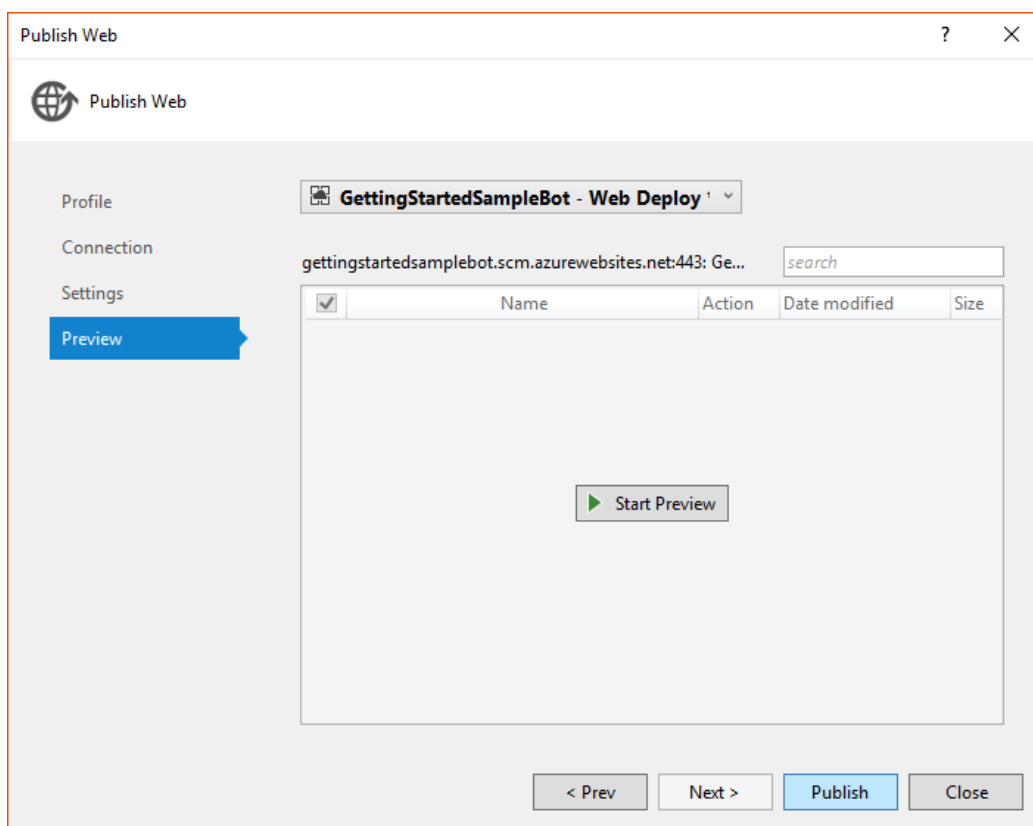
< Prev Next > Publish Close



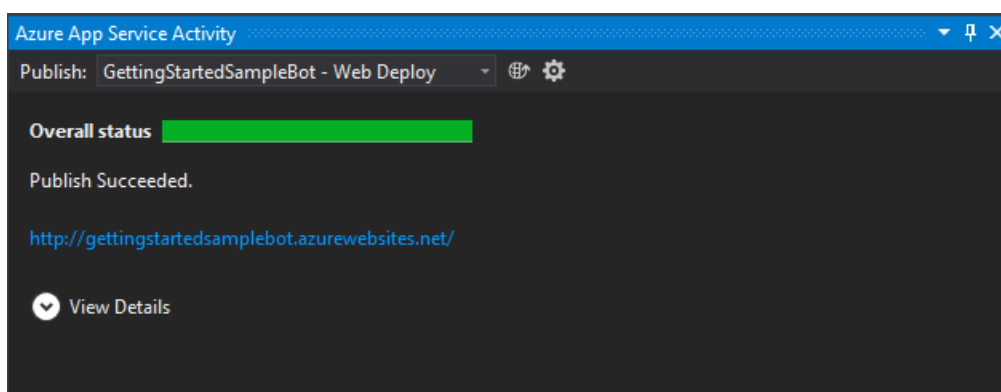
现在，您已返回到发布 Web 向导，请将目标网址复制到剪贴板，稍后您将需要它。点击“验证连接”以确保配置是好的，如果一切顺利，请单击“下一步”(Next)。



默认情况下，Bot 以 Release 的配置发布。如果要调试 Bot，请将配置 (Configuration) 更改为 Debug。无论如何，在这里你“发布” (Publish) 后，Bot 应用程序就已在 Azure 上发布。



您将在 Visual Studio 2015“输出”（Output）窗口中看到发布状态。一旦发布完成，您还将看到您的浏览器中显示的 Bot 应用程序的网页（浏览器将启动，并呈现您的 Bot 应用程序 HTML 页面），如下图所示。

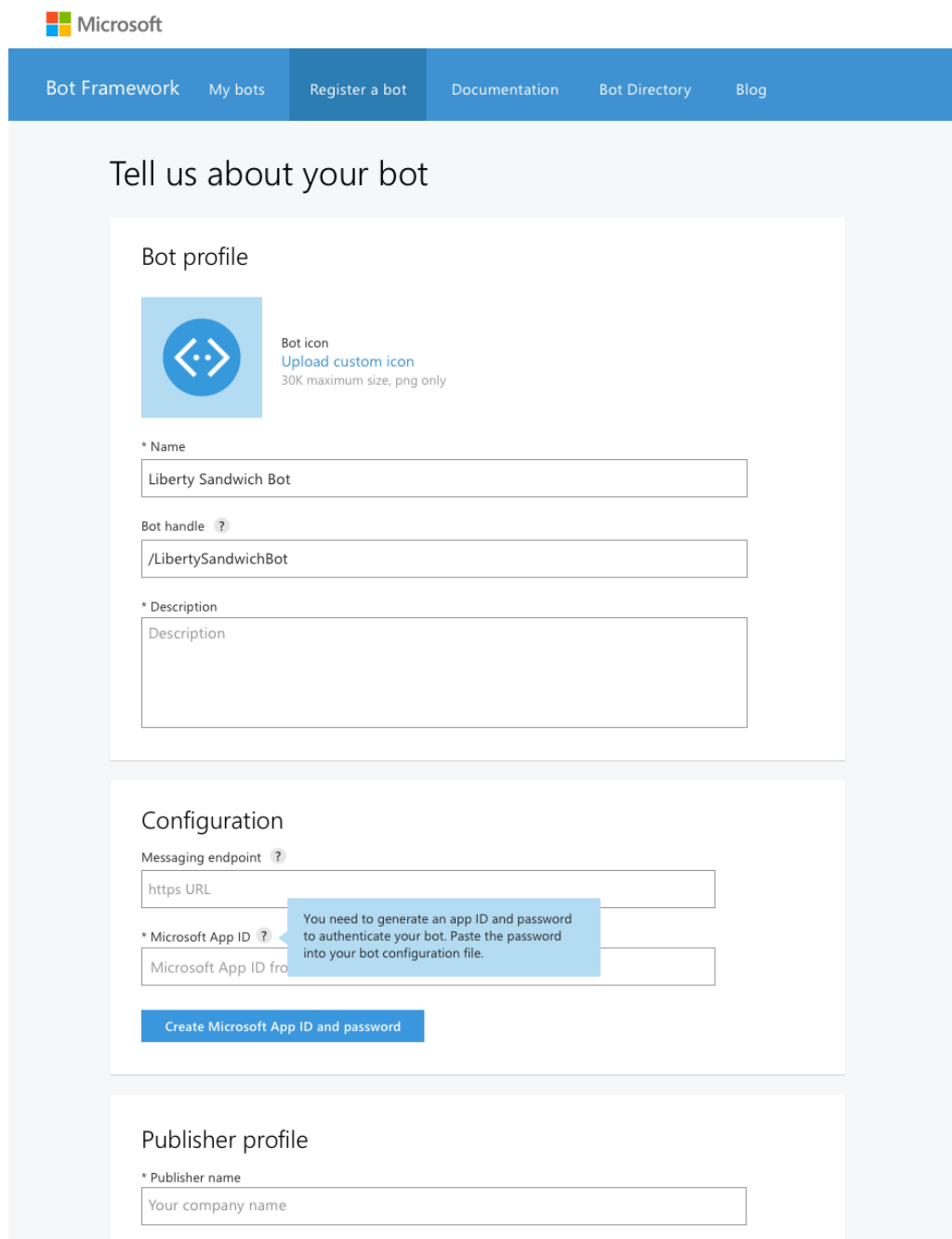


## 第五章 在微软 Bot 网站中注册您的 Bot

注册您的 Bot 配置连接器如何调用 Bot 的 Web 服务。请注意，MicrosoftAppId

和 MicrosoftAppPassword 是在您使用 微软 Bot 框架连接器 ( Microsoft Bot Framework Connector ) 注册 Bot 时生成的，MicrosoftAppId 和 MicrosoftAppPassword 用于 web 通信的身份验证，并允许开发人员用其创建通信的通道。

1. 浏览微软 Bot 框架门户网站 <https://dev.botframework.com>，并使用您的 Microsoft 帐户登录。
2. 单击“注册 Bot” (Register a Bot) 按钮并填写表单，此表单上的许多字段可以稍后更改。请记住使用从 Azure 部署生成的端点，当使用 Bot 应用程序临时变量时，您需要将粘贴的 URL 扩展到 / API / Messages 处的端点路径。请注意您该在 URL 前面添加 HTTPS 而不是 HTTP， Azure 将负责为您的 bot 提供 HTTPS 支持。点击表单底部的“创建” (Create) 保存更改。



The screenshot shows the Microsoft Bot Framework registration interface. At the top is the Microsoft logo and a navigation bar with links: Bot Framework, My bots, Register a bot, Documentation, Bot Directory, and Blog. The main heading is "Tell us about your bot". Below this is the "Bot profile" section, which includes a bot icon placeholder (a blue circle with a white robot head icon) and a link to "Upload custom icon" (30K maximum size, png only). The form fields in this section are: \* Name (Liberty Sandwich Bot), Bot handle (/?LibertySandwichBot), and \* Description (Description). Below the Bot profile section is the "Configuration" section. It includes a "Messaging endpoint" field (https URL) and a "Microsoft App ID" field. A blue tooltip points to the Microsoft App ID field, stating: "You need to generate an app ID and password to authenticate your bot. Paste the password into your bot configuration file." Below the Microsoft App ID field is a blue button labeled "Create Microsoft App ID and password". At the bottom is the "Publisher profile" section, which includes a field for \* Publisher name (Your company name).

3. 注册创建后，微软 Bot 框架将引导您生成 MicrosoftAppId 和 MicrosoftAppPassword。这些用于使用微软 Bot 框架验证您的 Bot。注意：生成 MicrosoftAppPassword 时，请务必将其记录在某处，因为您将无法再次看到它。

**Configuration**

Messaging endpoint ?

https URL

\* Microsoft App ID ?

Microsoft App ID from Microsoft App registration portal

Generate Microsoft App ID and password

现在 Bot 已注册，您需要更新 Visual Studio 项目中 web.config 文件中的配置。更改 web.config 文件中的以下配置以匹配注册时生成的值，然后可以开始编译。单击“显示”链接将显示该值，如果您需要更改您的 AppPassword，只需重新生成链接。更新您的 web.config 后重新发布您的 bot 至 Azure。

```
<? 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 appid and one of your appsecret keys-->
```

```
< add key = "MicrosoftAppId" value = "[GUID]" />
```

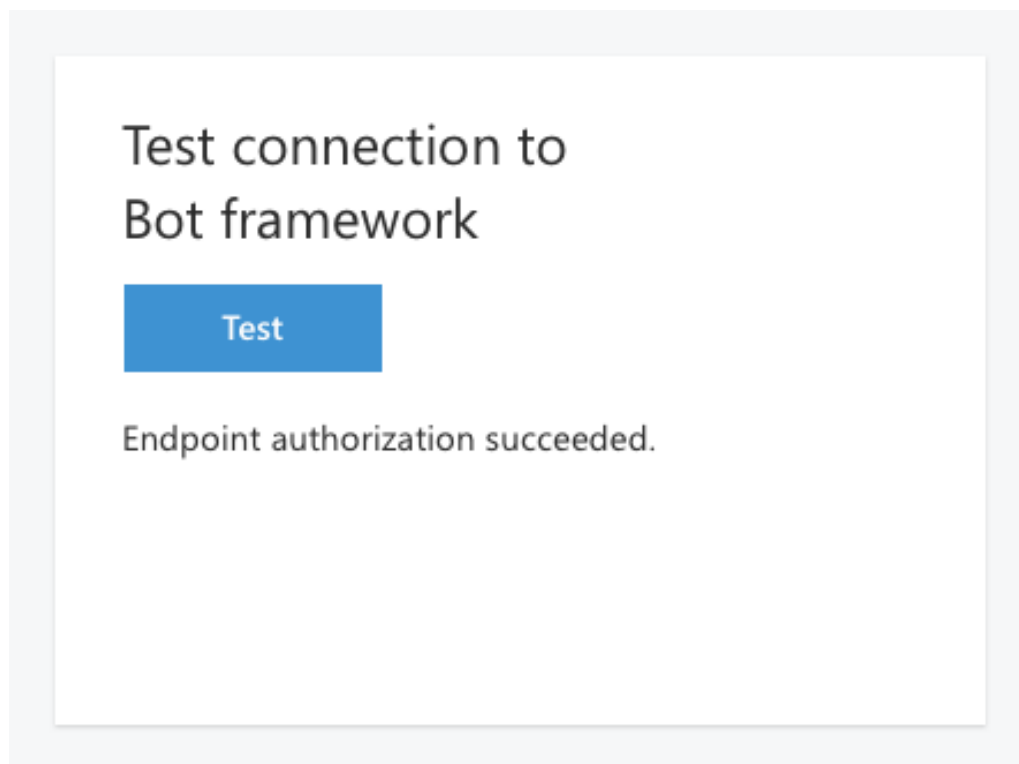
```
< add key = "MicrosoftAppPassword" value = "[PASSWORD]" />
```

```
</ appSettings >
```

## 1. 测试您的 bot 的连接

现在回到您的 Bot 开发人员信息中心，您可以使用测试窗口与 Bot 进行交互而无需任何配置，并验证 Bot 框架是否可以与您的 Bot Web 服务进行通信。

请注意，Bot 启动后的第一个请求可能需要 10 - 15 秒，因为这是 Azure 首次启动 Bot Web 服务，后续请求则很快。在此查看器中允许您查看 Bot 返回的 JSON 对象。



## 2. 配置通道

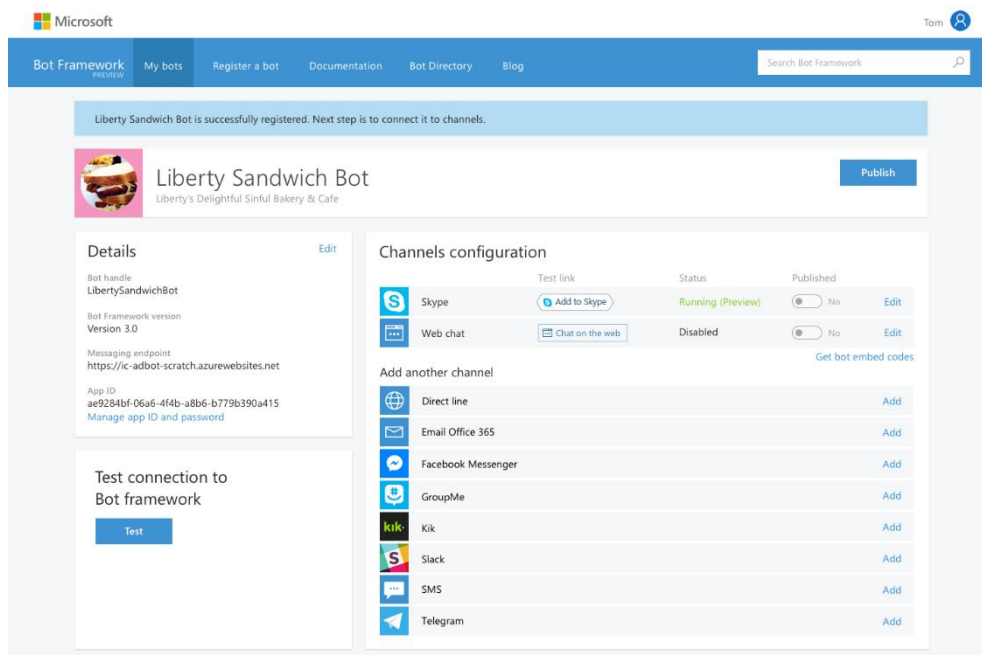
现在您的 Bot 已开始运行，您需要为用户正在使用的一个或多个渠道进行

配置。配置通道是微软 Bot 框架工作流和对话服务工作流的组合，对于您配置的每个通道是唯一的。

1. 要配置通道，请返回到 Bot 框架门户网站，其网址为

<https://www.robotframework.com>。登录后导航至 My Bots，然后



转到通道面板。



2. 选择您要配置的频道，然后点击添加。您将看到注册 Bot 的说明页面。在大多数情况下，您要将身份认证的凭据配置为目标服务的开发人员，然后注册您的应用程序，并获取一组您使用的微软 Bot 框架 OAuth 密钥。

Tom ▾

## Configure Facebook Messenger



### How to

▽ Getting Started

▽ Create a Facebook Page for your bot

▽ Create a Facebook App for your bot

▽ Copy your App ID and App Secret and save for later

▽ Set webhook callback url and verify token

▽ Generate Page Access Token

△ Enter your credentials

Credentials have not yet been validated.

Page Id

0000000000000000

App Id

0000000000000000

App Secret

0000000000000000

Page Access Token

xxxxxxxxxxxxxxxxxxxxxxxx

Resubmit

Deauthorize

☐ Enable this bot on Facebook Messenger

Enabling or disabling a channel doesn't affect its credentials.

I'm done configuring Facebook Messenger >

Contact us

Privacy & cookies

Terms of use

Code of conduct

© 2016 Microsoft

3. 完成这些步骤后，请返回开发者平台上的通道（Channel）页面，点击所选通道（如果您还没有）的复选框，然后点击“保存更改”。

所有的配置至此结束——你的 Bot 已万事俱备。当然 Bot 有自己的步骤，给予 Bot 权限参与所在的组/通道的通信，如获得短信电话号码或电子邮件的联络信息。他们可以在您 Bot 的 Bot 目录页面中执行此操作，该链接位于开发人员门户中 Bot 详细信息页面的顶部。



# 附录

## 参考资料：

Microsoft Bot Framework 资料汇总:

<https://github.com/andrewdyhhub/BotFramework>

MSDN 开发视频:

<https://channel9.msdn.com/Series/For-China-Developers/Conversation20161017A02>

Bot SDK & 示例代码：

<https://github.com/Microsoft/BotBuilder.git>

开发者入口：

<https://dev.botframework.com/>

HOL 动手实验（英文版）：

<https://docs.botframework.com/en-us/csharp/builder/sdkreference/gettingstarted.html>

详细的开发文档和介绍：

<http://docs.botframework.com/>

除了图片增加一些更多的交互

<http://blog.botframework.com/2016/05/13/BotFramework.buttons/>

BUILD 上的 Bots 的介绍课程：

<https://channel9.msdn.com/Events/Build/2016/B821>

LUIS 的介绍(提高更好的自然语言交互)

<https://www.azure.cn/cognitive-services/zh-cn/language-understanding-intelligent-service-luis>

语言分析：

<https://www.azure.cn/cognitive-services/zh-cn/linguistic-analysis-api>

与微信连接:

<http://www.cnblogs.com/sonic1abc/p/5941442.html>

示例代码：

<https://github.com/leonlj/BotDemo>