

Bot Framework

Outline

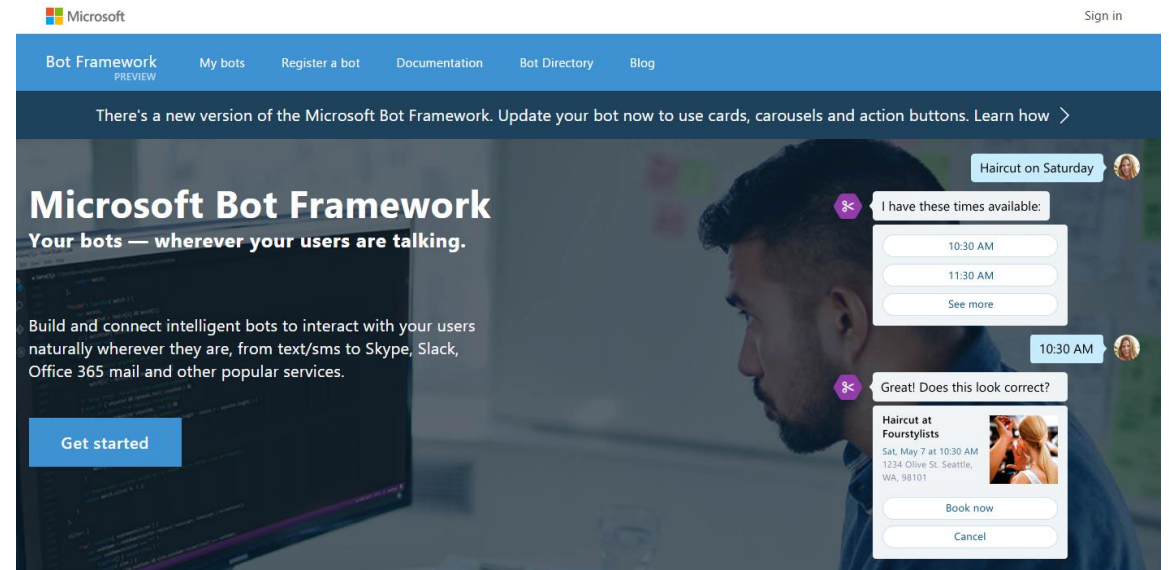
2

- ▶ 09:30 ~ 10:30 Microsoft Bot Framework 相關概念介紹
- ▶ 10:45 ~ 12:00 Bot Builder, Emulator 介紹
- ▶ 12:00 ~ 13:00 午餐
- ▶ 13:20 ~ 14:30 Hands-On Lab 1 :介紹 EchoBot
- ▶ 14:30 ~ 15:30 Hands-On Lab 2 : 介紹 SandwichBot
- ▶ 15:30 ~ 16:00 Break
- ▶ 16:00 ~ 17:30 Hands-On Lab 3 : 做出請假Bot(1 請假 ; 2 查詢請假時數)
- ▶ 17:30 ~ 18:00 Q & A

Microsoft Bot Framework 相關概念介紹

- ▶ Bot Builder
- ▶ Bot Connector
- ▶ Bot Framework Emulator

3



Bot Builder

4

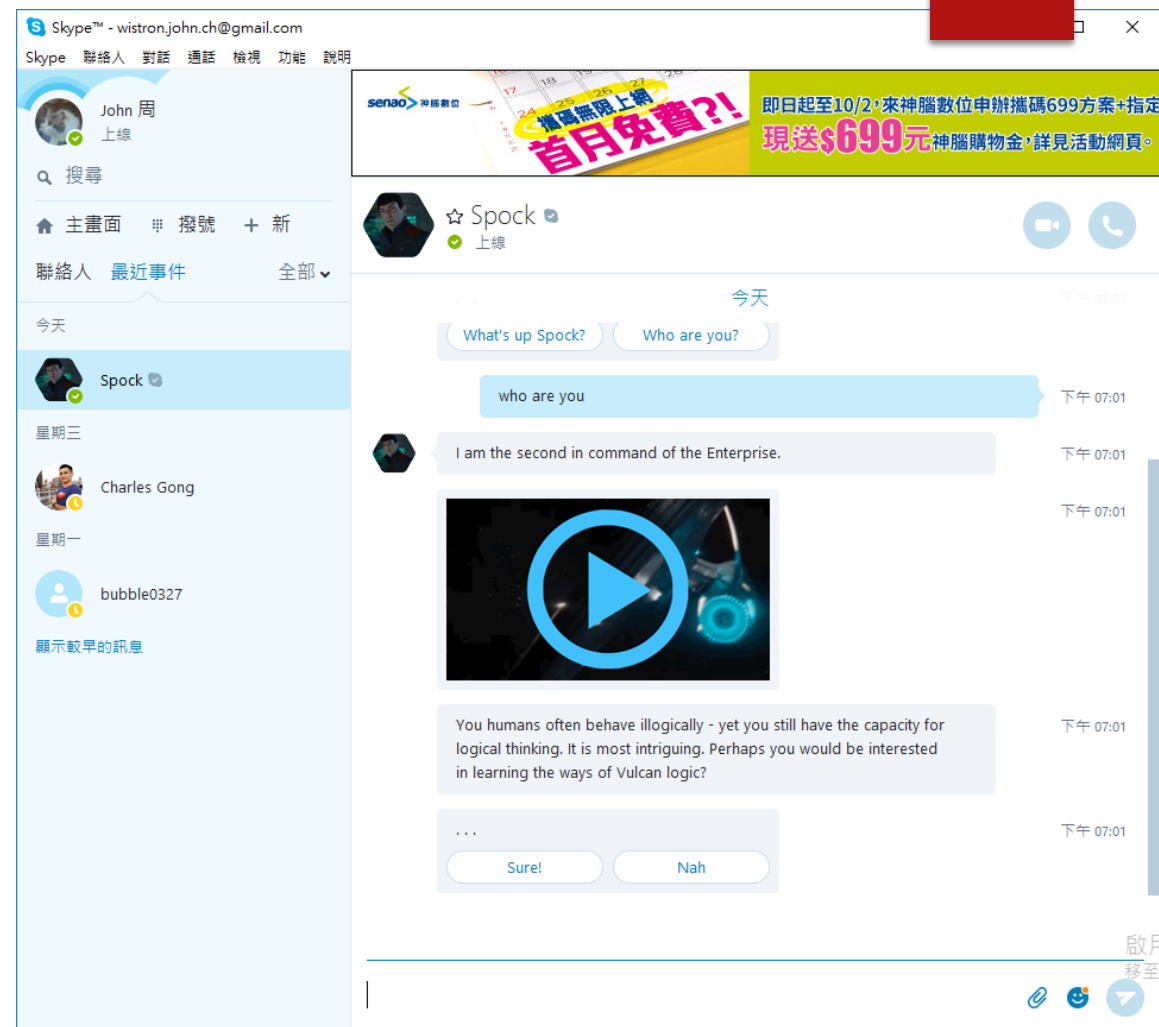
Microsoft Bot Builder is a powerful framework for constructing bots that can handle both freeform interactions and more guided ones where the possibilities are explicitly shown to the user. It is easy to use and leverages C# to provide a natural way to write bots.

- ▶ C#.NET ([gotoLink](#))
 - ▶ Dialog
 - ▶ Form Flow
 - ▶ Language Understanding Intelligent Service (LUIS)
- ▶ Node.js ([gotoLink](#))

Bot Builder .NET Dialog

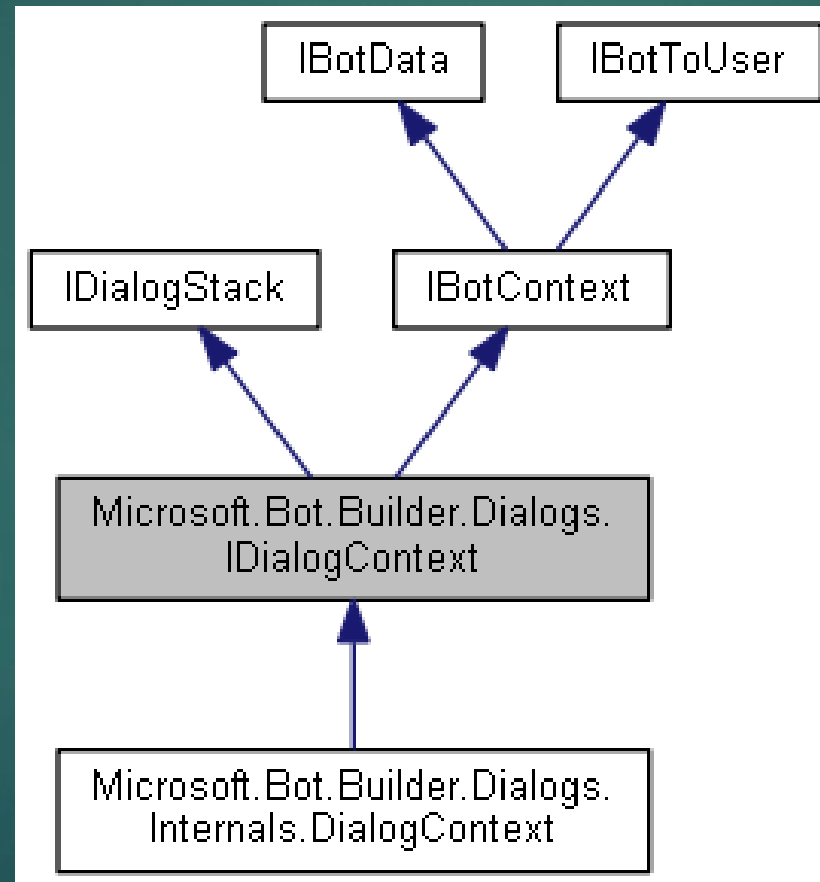
- ▶ To model a conversational process.
- ▶ Dialogs composed by one or more dialogs, and maintain a stack of dialog.
- ▶ Conversation state can be saved (like .NET website's session).

5



Dialog Stack

6



Dialog Sample

7

- ▶ Echo Bot
- ▶ Echo Bot with state

Bot Builder .NET FormFlow

- ▶ A dialog which sacrifices some of the flexibility provided by dialogs.
- ▶ Guides the user through filling in the form.

8

Please select a system

Video Only

Gps Only

Basic

1

Please enter a number for video devices

3

Please enter name

My car

Please select a model

Standard

Premium

Form flow sample

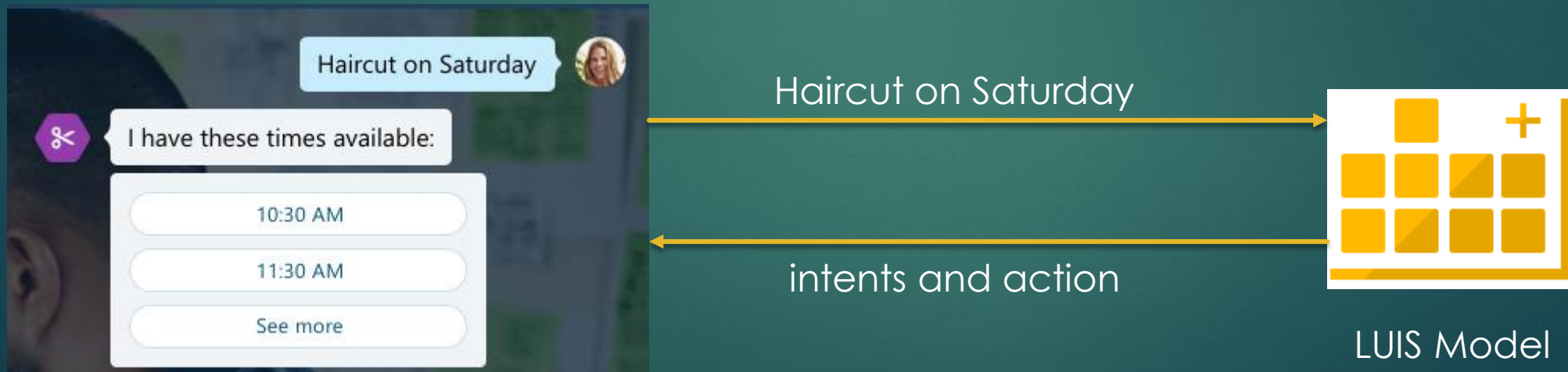
9

- ▶ Simple Sandwich Bot

Bot Builder .NET LUIS

10

- ▶ Create Language Understanding models.
- ▶ Send Message to LUIS
- ▶ Design your program to process intents and action.





Bot Connector

- ▶ It is a communication service that helps you connect your Bot with many different communication channels.
- ▶ Sending and receiving activities.
- ▶ Attachments, Cards and actions.









11

Channels

	Test link	Issues	Enabled	Published	
 Skype	Add to Skype	0	Yes (Preview)	<input type="checkbox"/> Off	Edit
 Web Chat		0	Yes	<input type="checkbox"/> Off	Edit

[Get bot embed codes](#)

Add another channel

	Direct Line	Add
	Email	Add
	Facebook Messenger	Add
	GroupMe	Add
	Kik	Add
	Slack	Add
	Telegram	Add
	Twilio (SMS)	Add

啟用 Windows
移至「設定」以啟用 Win

Bot developer Portal

- ▶ Bot management portal.
 - ▶ Connect to Channels
 - ▶ Real Bot endpoint
 - ▶ Sample test console

12

Framework
PREVIEW


My bots

Register a bot

Documentation

Bot Directory

Blog

 BotGraphDemo
WiAdvance

Details

Edit

Bot handle
BotGraphDemo

Bot Framework Version
3.0

Messaging endpoint
https://localhost:44324/api/messages

Microsoft App ID
10cad3d3-ca87-4112-b021-2af4f52c4848

Test connection
to your bot

Test



Channels

Test link

Issues






Enabled

Published


	Skype	Add to Skype	0	Yes (Preview)	<input type="checkbox"/> Off	Edit
	Web Chat		0	Yes	<input type="checkbox"/> Off	Edit


Get bot embed codes


Add another channel


	Direct Line	Add
	Email	Add
	Facebook Messenger	Add
	GroupMe	Add
	Kik	Add


My bots

 BotGraphDemo
WiAdvance

 Seven-Eleven
WiAdvance Corp.

 StockBot
John

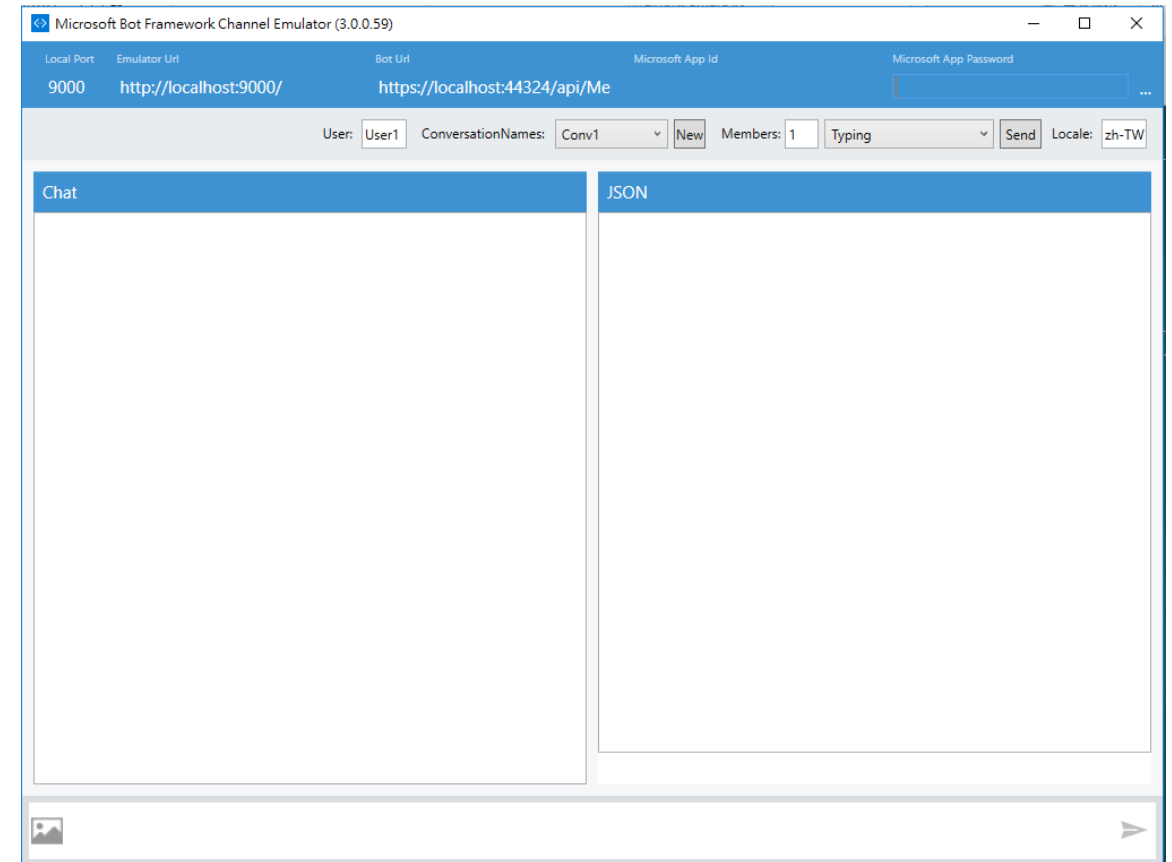
 WiAdvance.Bot.DoctorRes
WiAdvance Corp.

 WiAdvance.Bot.IMConnector
WiAdvance Corp.

Bot Framework Emulator

- ▶ An easy way to debug your bot([gotoLink](#)).

13



Bot Sample

14

- ▶ Spock (<https://join.skype.com/bot/7fa82ae6-547c-41b1-ba00-9de3b4190dac>)
- ▶ ImageBot (<https://join.skype.com/bot/79eaad73-f046-4720-9bba-3a2c8e1b0e0b>)

Lunch time

15

Hands-On Lab-prepare

16

- ▶ Visual studio Template C# ([gotoLink](#))
- ▶ Bot Framework Emulator Windows([gotoLink](#))
- ▶ Sample code ([gotoLink](#))

Hands-On Lab1

- ▶ EchoBot
- ▶ Dialog Echo Bot

17

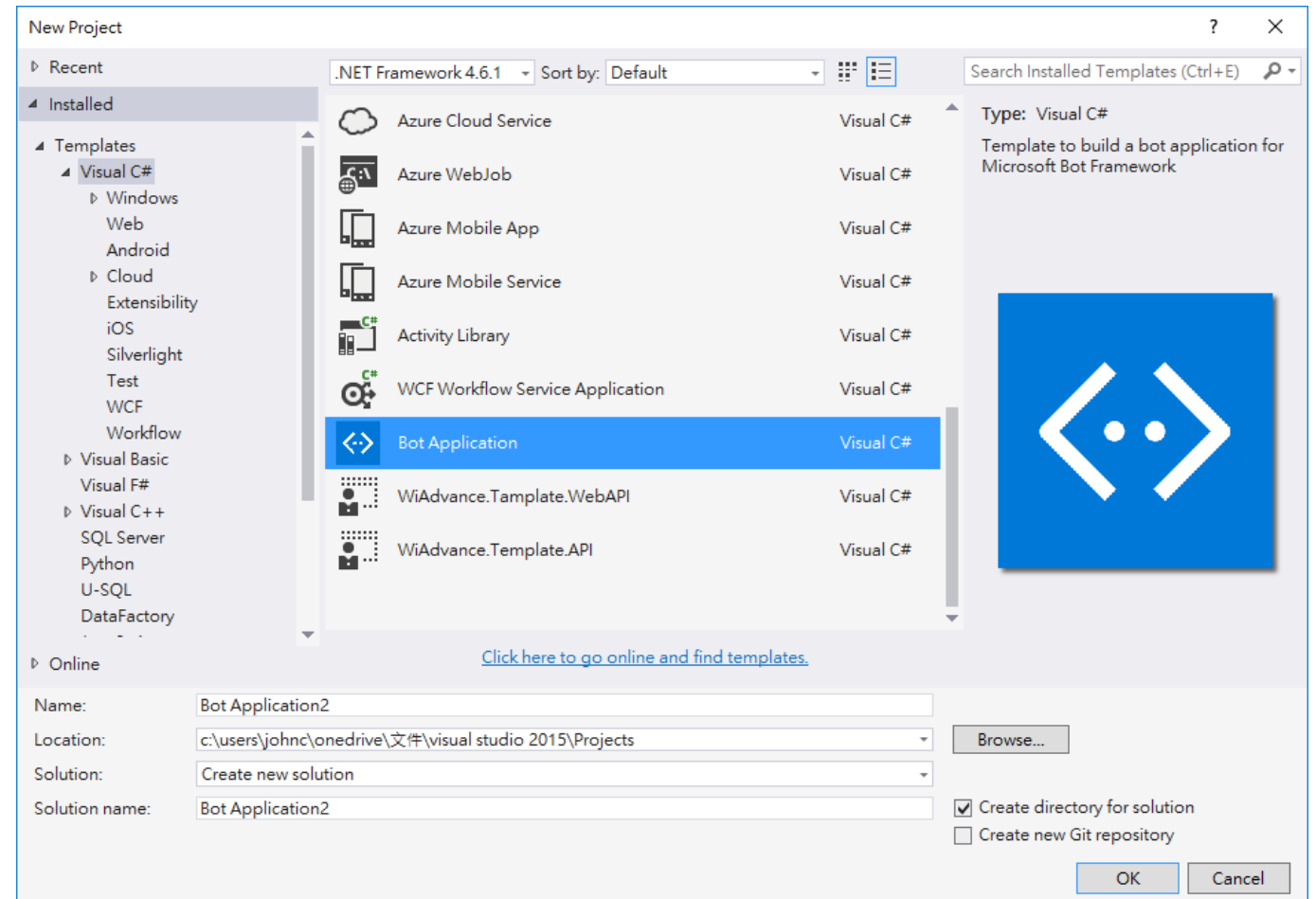
```
namespace MicrosoftGraphBot
{
    [BotAuthentication]
    public class MessagesController : ApiController
    {
        /// <summary>
        /// POST: api/Messages
        /// Receive a message from a user and reply to it
        /// </summary>
        [BotAuthentication]
        public virtual async Task<HttpResponseMessage> Post([FromBody]Activity activity)
        {
            if (activity != null && activity.GetActivityType() == ActivityTypes.Message)
            {
                await Conversation.SendAsync(activity, () => new AuthDialog());
            }
            else
            {
                this.HandleSystemMessage(activity);
            }
            return new HttpResponseMessage(System.Net.HttpStatusCode.Accepted);
        }

        private Activity HandleSystemMessage(Activity message)
        {
            if (message.Type == ActivityTypes.Ping)
            {
                Activity reply = message.CreateReply();
                reply.Type = ActivityTypes.Ping;
                return reply;
            }

            return null;
        }
    }
}
```

Create an app from template

- ▶ Use Bot Template to create a new application.
- ▶ Build & Run



Talk to your bot

19

- ▶ Open Emulator
- ▶ Change “Bot Url” to project web path.
- ▶ Type some words.

Local Port

Emulator Url

Bot Url

Microsoft App Id

Microsoft App Password

9000

http://localhost:9000/

http://localhost:3979/api/mess:



...

User:

User1

ConversationNames:

Conv1



New

Members: 1

Typing



Send

Locale:

zh-TW

Chat

hi



You sent hi which was 2 characters

JSON

```
{
  "type": "message",
  "timestamp": "2016-10-03T07:18:53.0438664Z",
  "from": {
    "id": "56800324",
    "name": "Bot1"
  },
  "conversation": {
    "id": "8a684db8",
    "name": "Conv1"
  },
  "recipient": {
    "id": "2c1c7fa3",
    "name": "User1"
  },
  "text": "You sent hi which was 2 characters",
  "replyToId": "cc514d627261441186ebaae77694ce48"
}
```

202 Accepted



Add a dialog to handle conversation context

- ▶ Add a new class call EchoDialog.cs

```
[Serializable]
public class EchoDialog : IDialog<object>
{
    public async Task StartAsync(IDialogContext context)
    {
        context.Wait(MessageReceivedAsync);
    }

    public async Task MessageReceivedAsync(IDialogContext context, IAwaitable<IMessageActivity> argument)
    {
        var message = await argument;
        await context.PostAsync("You said: " + message.Text);
        context.Wait(MessageReceivedAsync);
    }
}
```

Add a dialog to handle conversation context

- Change post method.

```
public virtual async Task<HttpResponseMessage> Post([FromBody] Activity activity)
{
    // check if activity is of type message
    if (activity != null && activity.GetActivityType() == ActivityTypes.Message)
    {
        await Conversation.SendAsync(activity, () => new EchoDialog());
    }
    else
    {
        HandleSystemMessage(activity);
    }
    return new HttpResponseMessage(System.Net.HttpStatusCode.Accepted);
}
```

Add a dialog to keep conversation context state.

23

- ▶ Declare count to keep status.
- ▶ Create a contracture.

```
protected int count = 1;

public async Task StartAsync(IDialogContext context)
{
    context.Wait(MessageReceivedAsync);
}
```

Add a dialog to keep conversation context state.

24

- Add MessageReceivedAsync to control all conversation stack.

```
public virtual async Task MessageReceivedAsync(IDialogContext context, IAwaitable<IMessageActivity> argument)
{
    var message = await argument;
    if (message.Text == "reset")
    {
        PromptDialog.Confirm(
            context,
            AfterResetAsync,
            "Are you sure you want to reset the count?",
            "Didn't get that!",
            promptStyle: PromptStyle.None);
    }
    else
    {
        await context.PostAsync(string.Format("{0}: You said {1}", this.count++, message.Text));
        context.Wait(MessageReceivedAsync);
    }
}
```


Add a dialog to keep conversation context state.

25

- ▶ AfterResetAsync: to handle delegate.
- ▶ Notice that IAwaitable should map to PromptDialog.

```
public async Task AfterResetAsync(IDialogContext context, IAwaitable<bool> argument)
{
    var confirm = await argument;
    if (confirm)
    {
        this.count = 1;
        await context.PostAsync("Reset count.");
    }
    else
    {
        await context.PostAsync("Did not reset count.");
    }
    context.Wait(MessageReceivedAsync);
}
```

Hands-On Lab2

- ▶ Sandwich Bot
 - ▶ Design a FormFlow.
 - ▶ Design return message.

26



Add a Order to help sandwich orders.

27

- ▶ Create a class SandwichOrder.cs

Add a Order to help sandwich orders.

28

- ▶ The code inside is:

```
public enum SandwichOptions
{
    BLT, BlackForestHam, BuffaloChicken, ChickenAndBaconRanchMelt, ColdCutCombo, MeatballMarinara,
    OvenRoastedChicken, RoastBeef, RotisserieStyleChicken, SpicyItalian, SteakAndCheese, SweetOnionTeriyaki, Tuna,
    TurkeyBreast, Veggie
};

public enum LengthOptions { SixInch, FootLong };
public enum BreadOptions { NineGrainWheat, NineGrainHoneyOat, Italian, ItalianHerbsAndCheese, Flatbread };
public enum CheeseOptions { American, MontereyCheddar, Pepperjack };
public enum ToppingOptions
{
    Avocado, BananaPeppers, Cucumbers, GreenBellPeppers, Jalapenos,
    Lettuce, Olives, Pickles, RedOnion, Spinach, Tomatoes
};

public enum SauceOptions
{
    ChipotleSouthwest, HoneyMustard, LightMayonnaise, RegularMayonnaise,
    Mustard, Oil, Pepper, Ranch, SweetOnion, Vinegar
};

[Serializable]
public class SandwichOrder
{
    public SandwichOptions? Sandwich;
    public LengthOptions? Length;
    public BreadOptions? Bread;
    public CheeseOptions? Cheese;
    public List<ToppingOptions> Toppings;
    public List<SauceOptions> Sauce;

    public static IForm<SandwichOrder> BuildForm()
    {
        return new FormBuilder<SandwichOrder>()
            .Message("Welcome to the simple sandwich order bot!")
            .Build();
    }
};
```

DoProcess On Complete

29

- ▶ Delegate OnCompletion method.

```
public static IForm<SandwichOrder> BuildForm()
{
    OnCompletionAsyncDelegate<SandwichOrder> processOrder = async (context, state) =>
    {
        var reply = context.MakeMessage();
        reply.Text = JsonConvert.SerializeObject(state);
        await context.PostAsync(reply);
    };
    return new FormBuilder<SandwichOrder>()
        .Message("Welcome to the simple sandwich order bot!")
        .OnCompletion(processOrder)
        .Build();
}
```

Breaks

30

Hands-On Lab3

31

- ▶ 做出請假Bot包含以下功能
 - ▶ 請假
 - ▶ 事、病、喪、生理假四種可選。
 - ▶ 同一天只能請一個假。
 - ▶ 生理假只允許女性申請。
 - ▶ 查詢請假記錄
 - ▶ 查詢個人請假總天數。
 - ▶ 查詢各假別請假總天數。
 - ▶ 列出各假別請假明細(日期)。

9000

http://localhost:9000/

http://localhost:3979/api/mess:



User:

User1

ConversationNames:

Conv1

New

Members: 1

Typing

Send

Locale:

zh-TW

Chat



Your leave apply is proccessed.

?



How can i server you?

我要請假

查詢請假記錄

沒事

我要請假

Please Select your gendar

Personal Leave

JSON

```
{
  "type": "message",
  "timestamp": "2016-10-03T16:23:38.7408179Z",
  "from": {
    "id": "56800324",
    "name": "Bot1"
  },
  "conversation": {
    "id": "8a684db8",
    "name": "Conv1"
  },
  "recipient": {
    "id": "2c1c7fa3",
    "name": "User1"
  },
  "text": "The leave apply was canceled.",
  "replyToId": "dc3eba4303ff4c24a481ea3c06792dee"
}
```

202 Accepted



References

33

- ▶ <https://dev.botframework.com/>
- ▶ <https://www.luis.ai/>
- ▶ <https://ankitbko.github.io/2016/08/ChatBot-using-Microsoft-Bot-Framework-Part-1/>
- ▶ <http://www.codeproject.com/Articles/1106457/An-Introduction-to-the-Microsoft-Bot-Framework>

Q&A

See you next section!