Developing and Deploying Intelligent Chat Bots

Microsoft

UI Controls

Objectives

How much NLP do bots need?

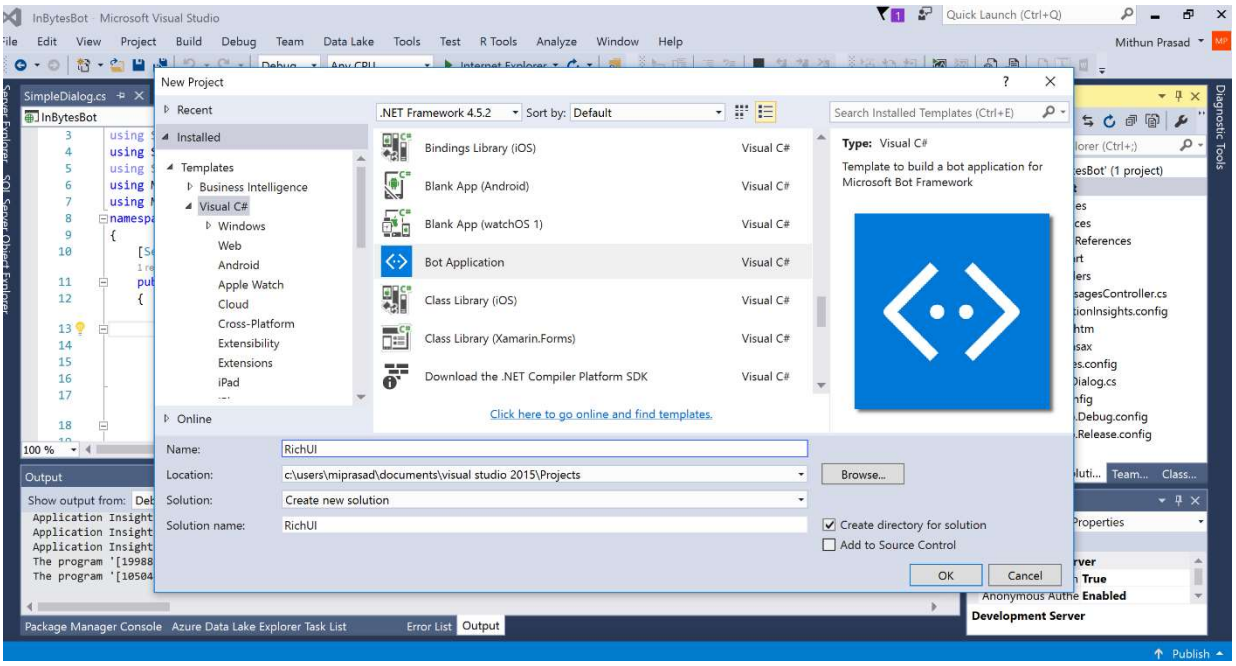
It is quite natural for users to prepare their messages using natural language. But NLP can get quite tricky and extracting semantic representations in text is still an open research. A solution around avoiding large use of text is to leverage rich User Interface (UI).

For example, you could ask a user "what kind of pizza would you like?". This would result in an infinite number of possible pizza varieties if you include all the combinations via free text. A more finite way of asking a user "what kind of pizza would you like?" is to present a list of options from which the user can select. This would avoid working with free text making the task of processing much easier for the Bot.

The aim of this lab is to demonstrate integration of user controls such as buttons and menus with bots.

Setup

Create a new project using the Bot Application template as shown below.



PromptDialog

A PromptDialog is essentially a Dialog factory for creating simple prompts. It allows you to ask the user for a response, and indicate what code will run when the response is given.

A PromptDialog can be one of the following types:

- Prompt for an attachment
- Prompt for one of a set of choices
- Ask a yes/no question
- Prompt for a long

- Prompt for a double
- Prompt for a string

Basic Confirmation

The simplest Prompt type is a basic confirmation. This prompt will allow the user to respond with Yes or No. With some bots, we will automatically get these displayed. It is important to note that although the prompts can be displayed via the emulator, depending on the channel used (i.e. SMS) this may not be available to the user.

To implement this, we can call PromptDialog.Confirm. We will need to provide the context, the name of a suitable Action and options for how we what we will display in the prompt. See example below:

```
PromptDialog.Confirm(context, Confirmed, "Do you have a question?");


public async Task Confirmed(IDialogContext context, IAwaitable<bool> argument)
  {
    bool isCorrect = await argument;
    if (isCorrect)
    { }
    else
    { }
  }
```

Custom Prompts

Although Yes/No prompts are great, sometimes you want to provide a more specific set of options. The custom prompts will allow for a finite set of options. For implementing custom prompts, we can call PromptDialog.Choice. The arguments are pretty similar to the above example except for action – we are awaiting a string and not a bool.

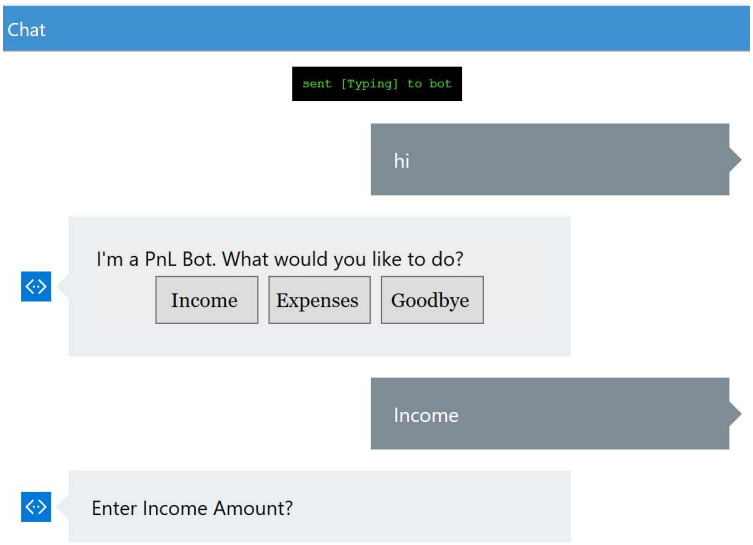
```
var PromptOptions = new string[] { "Beer?", "Coffee?", "Freshly Squeezed Juice?" };

PromptDialog.Choice(context, TestConfirm, PromptOptions, $"Are you thirsty?");

public async Task TestConfirm (IDialogContext context, IAwaitable <object> result)
{
            var  confirm = await result;
            if ((string)confirm == "Beer?")
            {
            }
            else if ((string)confirm == "Coffee?")
            {
            }
```

```
            else
            {
            }
}
```

Profit and Loss Bot

Develop a *Profit and Loss* Bot using PromptDialog.Choice in MessagesController for capturing Income and Expenses from the user. The goal of the bot is to help users compute profit/loss and help file taxes.

The prompts can be Income and Expenses as show below.



At each prompt chosen, compute the total Income and total Expenses and display to the user. For reference, you can look at the MessagesController code in Student-Resources/Labs/CSharp/UIControls folder.