# Lab for Bot Framework Developer Deep Dive .NET

## EchoBot and Dialogs

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

### A Simple EchoBot

From Bot Framework .NET template, perform the following steps to add Dialog functionality.

Add this namespace:

using Microsoft.Bot.Builder.Dialogs;

Add this class:

[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);

}

}

Replace the Post method with this one:

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

{

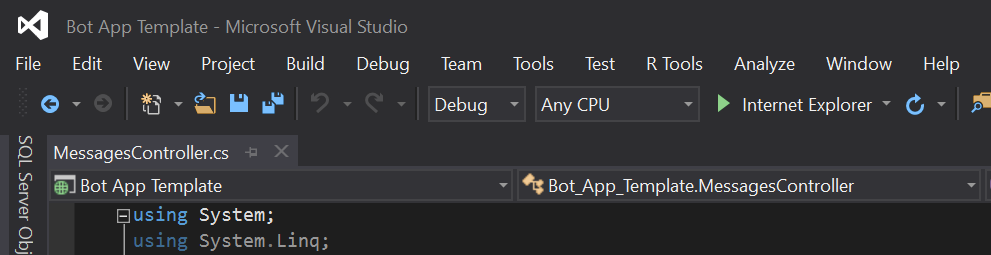
HandleSystemMessage(activity);

}

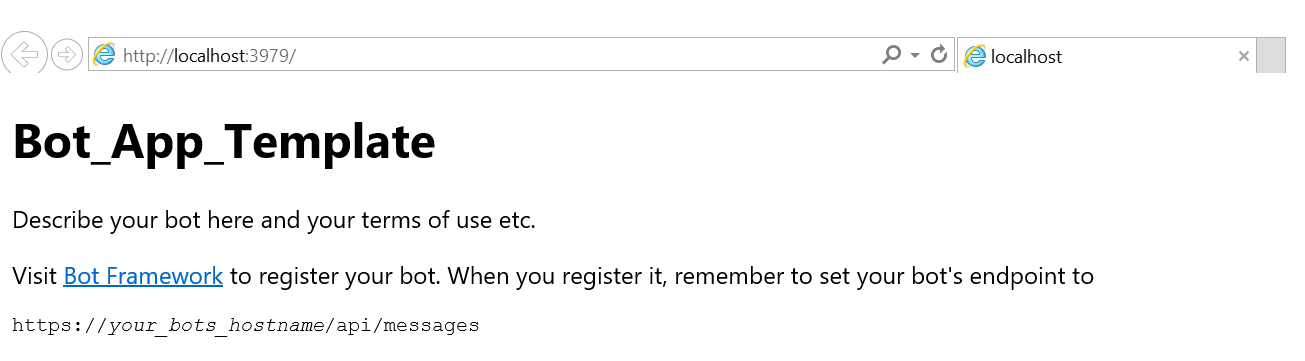
return new HttpResponseMessage(System.Net.HttpStatusCode.Accepted);

}

Run from VS



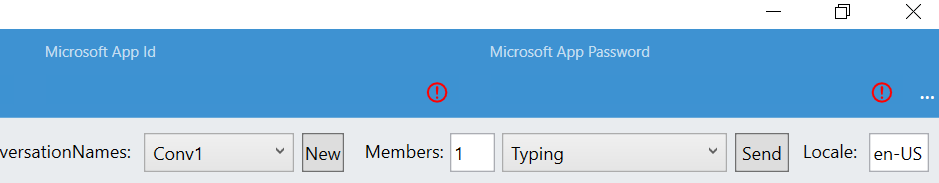
This should open a browser window with the endpoint (showing the default.htm file from project)



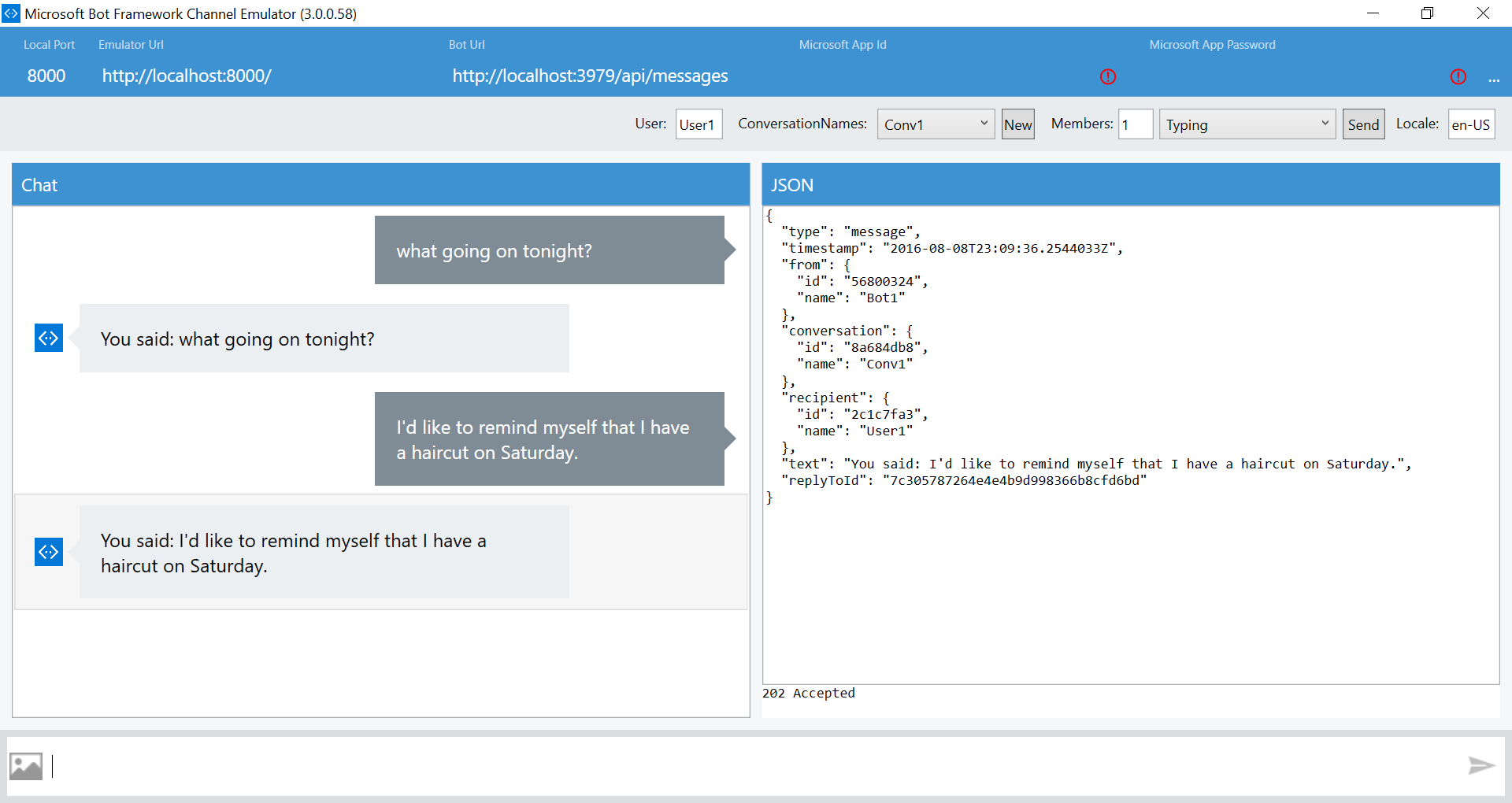
The port above should match the port in the emulator.

Open the Bot Framework Emulator application.

For local testing with the emulator make sure the Id and password fields are clear:



Test out this echo dialog bot.



### Adding State to the EchoBot

Replace the old EchoDialog with this one:

[Serializable]

public class EchoDialog : IDialog<object>

{

// the state we are persisting with this dialog on each message

protected int count = 1;

public async Task StartAsync(IDialogContext context)

{

context.Wait(MessageReceivedAsync);

}

// we have added check to see if the input was "reset"

// if that is true we use the built-in Prompts.Confirm dialog to spawn a sub-dialog

public virtual async Task MessageReceivedAsync(IDialogContext context, IAwaitable<IMessageActivity> argument)

{

var message = await argument;

if (message.Text == "reset")

{

// sub-dialog

PromptDialog.Confirm(

context,

AfterResetAsync, // result is then passed onto the AfterResetAync method

"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);

}

}

// check on the response and perform the action including sending a message back to the user

public async Task AfterResetAsync(IDialogContext context, IAwaitable<bool> argument)

{

var confirm = await argument;

if (confirm)

{

this.count = 1;

await context.PostAsync("Count reset.");

}

else

{

await context.PostAsync("Did not reset count.");

}

// final step is to do IDialogContext.Wait with

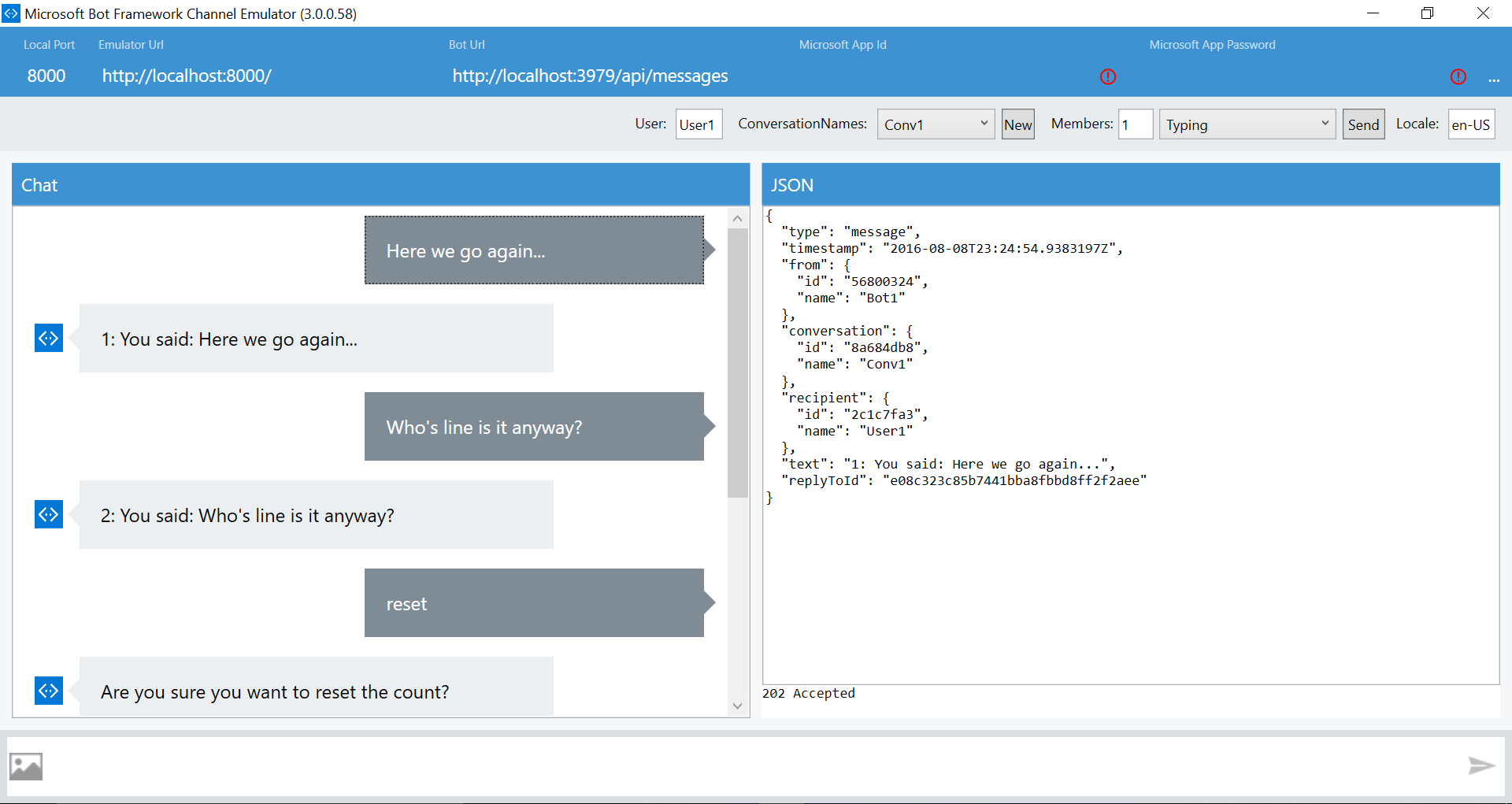
// continuation back to MessageReceivedAsync on the next message

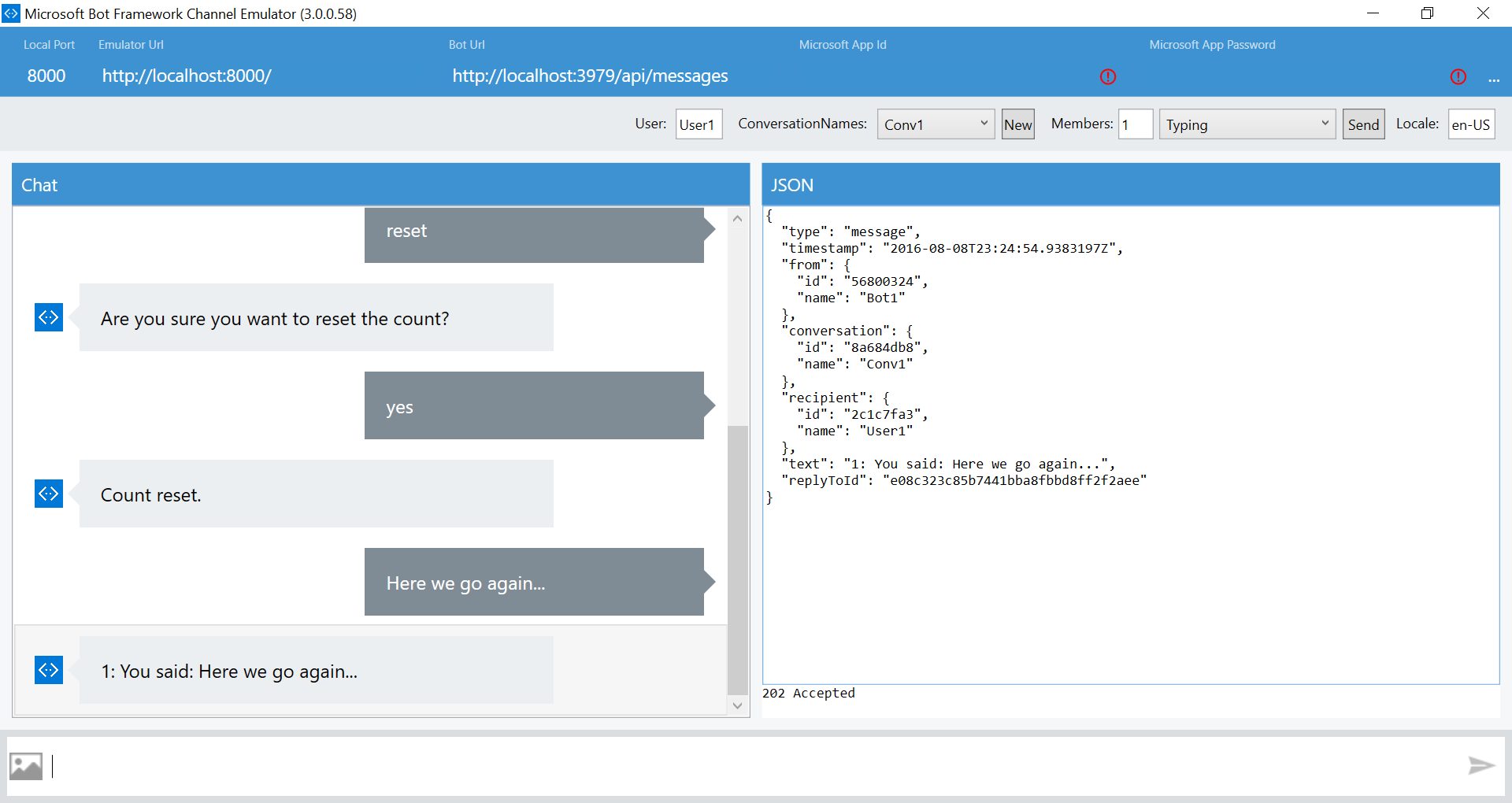
context.Wait(MessageReceivedAsync);

}

}

Again, run the code in VS and test in the bot emulator.





### Extra Credit – AlarmBot

Follow the instructions and code for adding a natural language understanding model (LUIS API) to the EchoBot above in the documentation (<https://docs.botframework.com/en-us/csharp/builder/sdkreference/dialogs.html>)