**Terminologies in Echo Bot**

<Application>.bot contains all bot related settings, such as App Id, app secret, Endpoint

**Conversational Flow**

There are major three components in bot communication

1. Bot Service - facilitate communication between bot app and channel. Every interaction between bot and user is treated as activity
   1. Activity
      1. Messages
      2. Attachment
      3. Suggested action
2. Channel

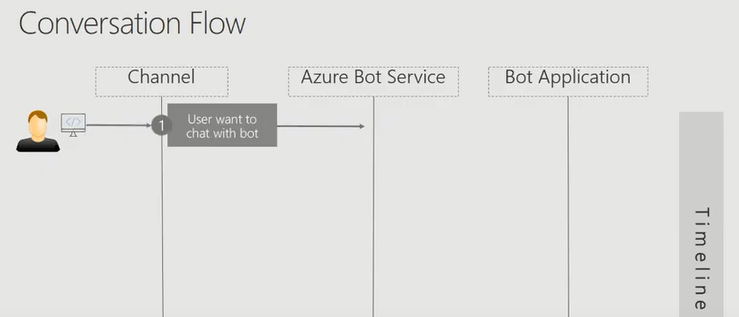
Platform used to chat with the bot, skype, messenger etc.

Each channel can include additional activity that they send across

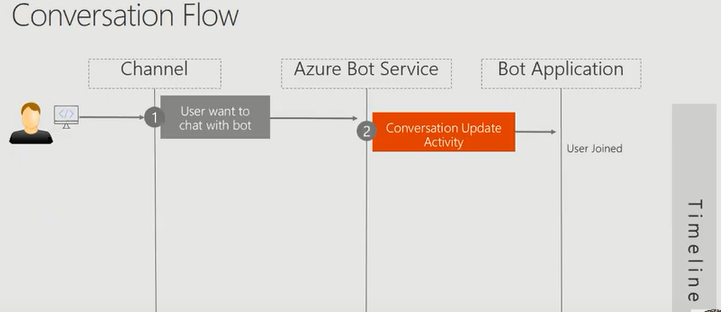
1. Last one is bot application, which processes the activity and sends a response

**Conversation Flow**

i. When User want to chat with a bot he adds it as a contact in the channel,

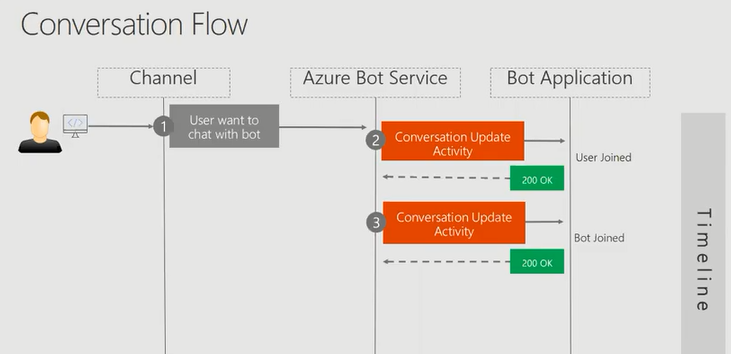


ii. In Response to that Azure Bot service sends the http post request to the bot application with an activity called Conversation Update. This contains user information in the request payload data

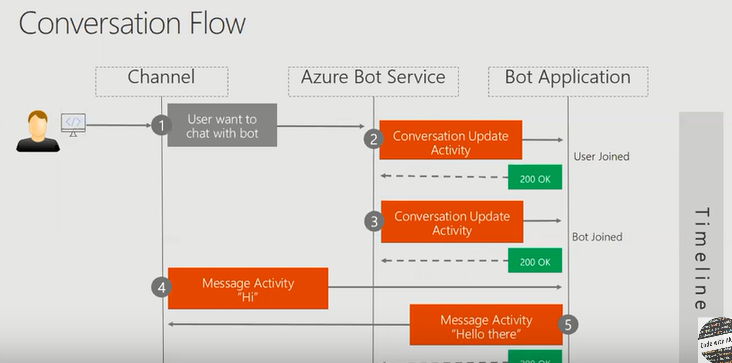


iv. After User joined the conversation, the bot application will send a conversation update activity to the

Bot service intimating that it has also joined the conversation



V.When user says Hi and the bot says hello these are the message kind of activity.



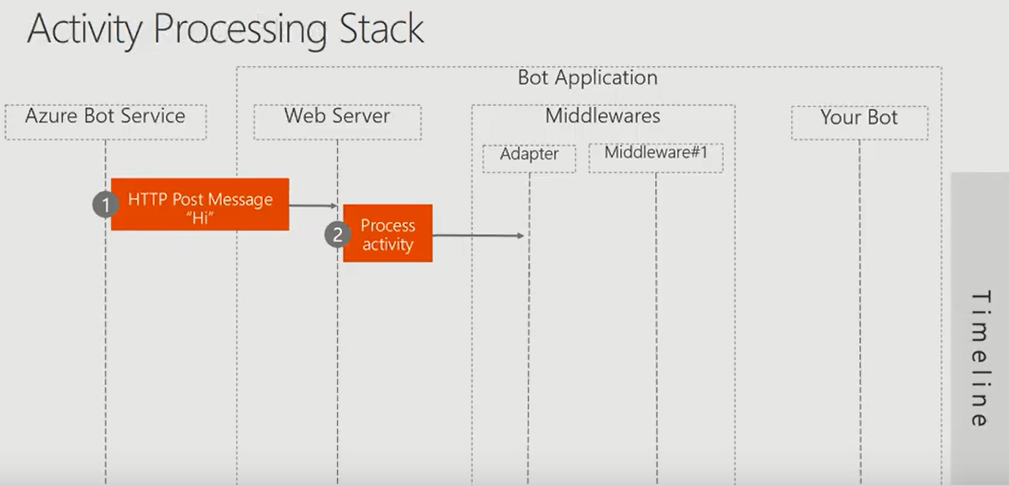
**Activity Processing Stack**

**Building Blocks**

1. Web Server
2. Middleware
3. Your Bot code

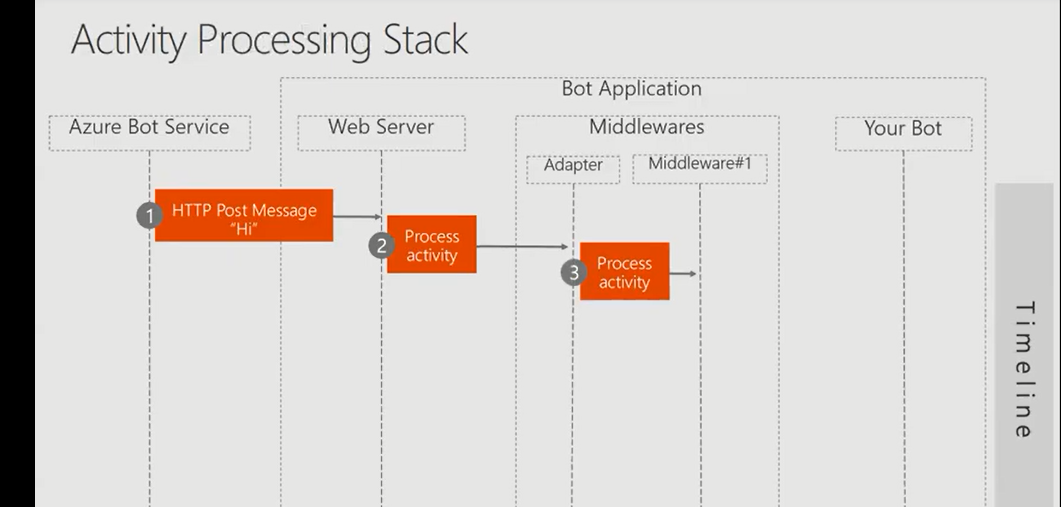
**Flow**

1. Azure bot service will send an activity of type message(sent by user) to the Webserver



ii.Web server will de serialize that activity the json data into activity object. That activity object will be given to the adapter.

iii.Adapter creates a turn context and calls the middleware. The turn consist of users incoming activity to the bot, and bots response to the user.

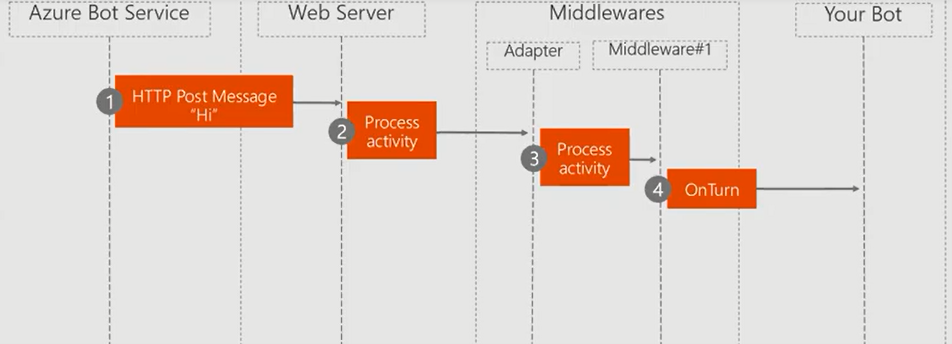


The turn context object contains

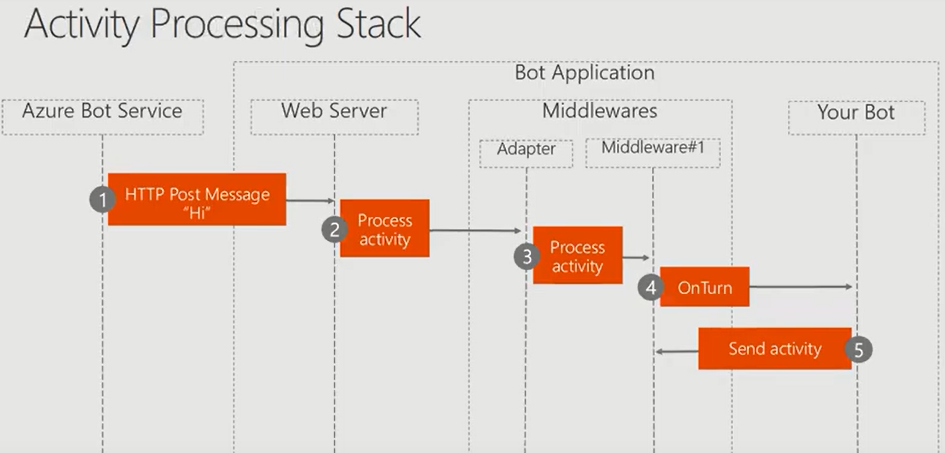
1. Sender of the activity
2. Receiver of the activity
3. Channel specific data
4. Tokens

iv.The middleware is the processor function over the incoming request as well as the outgoing response

At the end of the middleware pipeline it will call the turn handler function from your bot code. So your bot code contains the controller class. Inside the controller class you have the OnTurn Method. Turn Context will be passed after all the processing of the middleware to the OnTurn Method. I



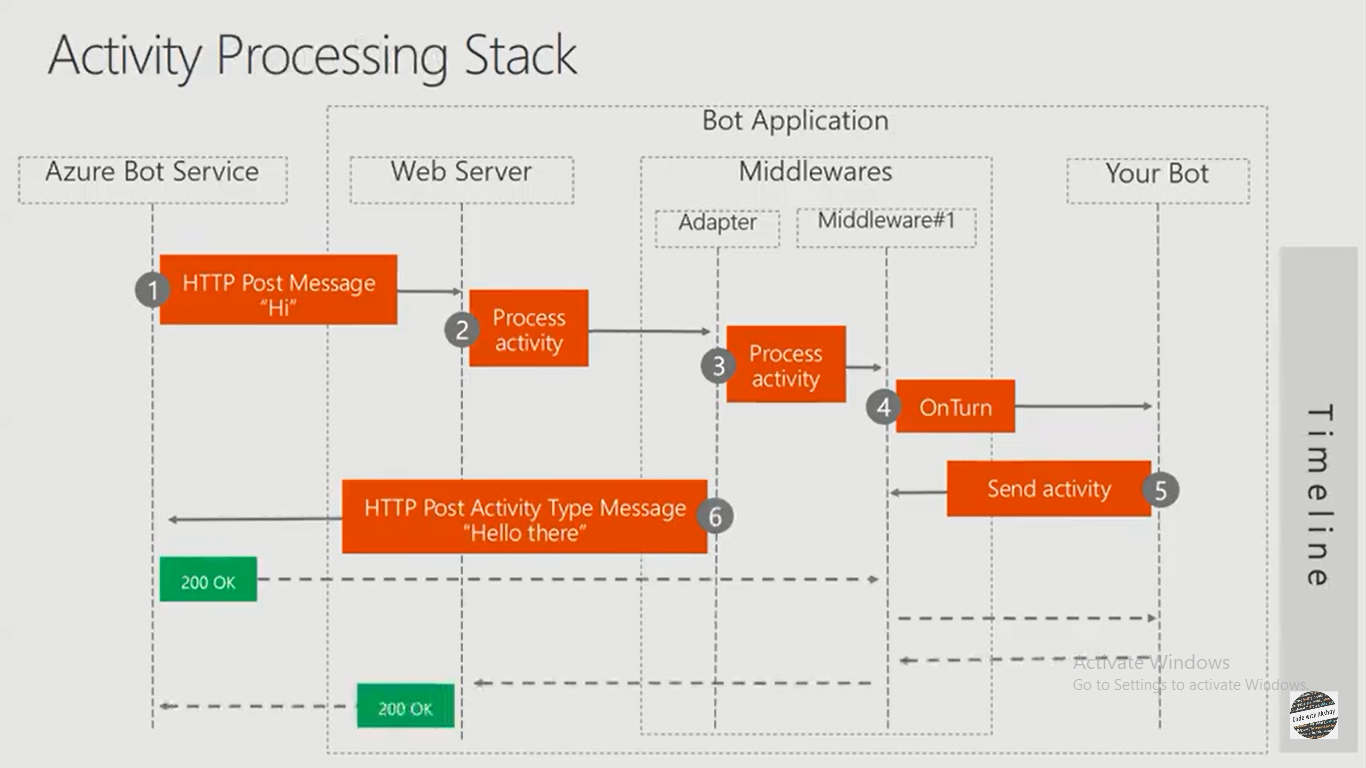
v. Inside the OnTurn Method inbound activities content will be processed, and bot will respond back as activities.



vi. Middleware processes the response back that is the outbound messages

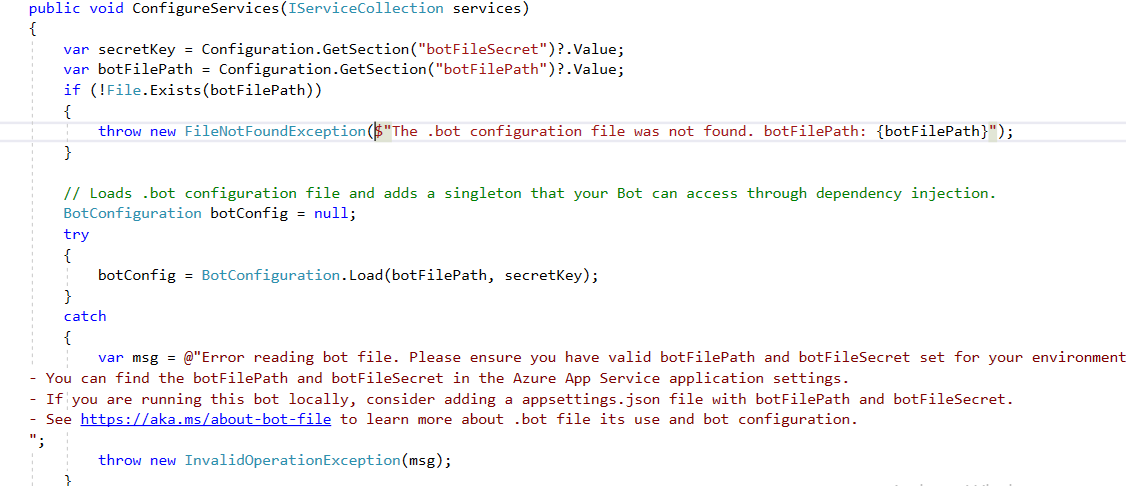
After receiving the response from the middleware the bot service sends 200 OK

to the bot application, and the bot application also send the same 200 OK to the bot service.



Code Overview

**Startup.cs**



First all the bot related settings will be loaded from the bot file. To load the settings BotConfiguration class can be used.



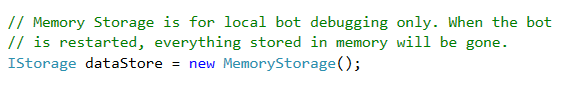
This botConfig object to the service collection for dependency injection



For maintaining the counter of the messages we are going to use the conversation State from the bot builder class



For storing the data we are using MemoryStorage.



Next part of the execution is adding and configuring the bot inside the ConfigureService

Here the MyFirstBot will be added to the service Collection object, and we configure the options inside it.



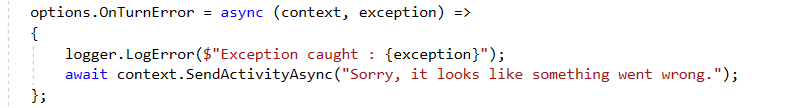
First option is to provide the AppId and AppSecret. So we can maintain different endpoint services in the bot file.



After that using the loggerFactory we are creating the logger for our bot



For global exception handling we can mention the OnTurn function handler

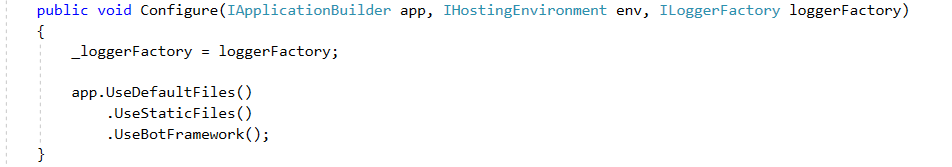


For handling any exception which has occurred in the bot file we are logging error, and we are mentioning the exception details inside this log.

We are hooking the TurnHandler as the last middleware.

Next is Configure method

It is used to setup the middleware’s that handles every http request. The ordering of the middle wares is very important because they gets executed sequentially.

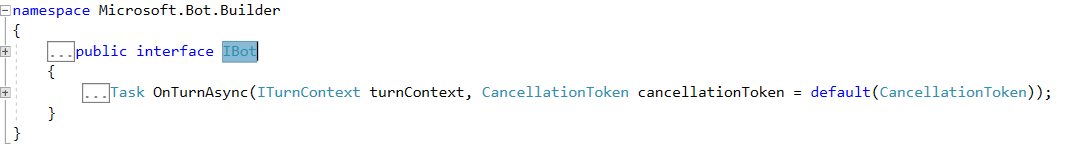


Here we have added the UseBotFramework at the end of the request execution pipeline, it will add the bot framework.

**<ApplicationName>Bot.cs**

This implements the IBot Interface, which has the OnTurnAsync Method.





Control will go to this method whenever user enters any message as it passes through all the middlewares, it calls the bot class. After the execution of the constructor execution control given to this method.

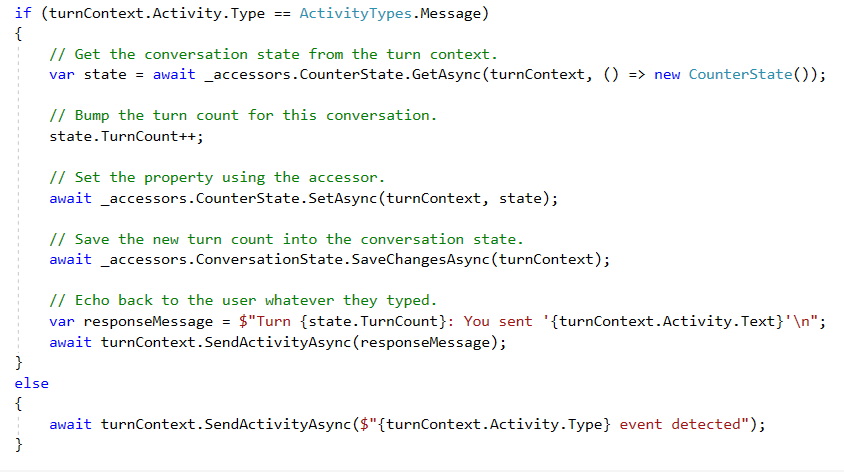
Here we are checking whether the type of an activity is a message.

If it is a message we are retrieving the counter state from the conversation state, for that we are using accessors

Here we will see how to maintain the state inside the bot and access the state variables using the accessors.

We are saving the counter state in the counter state class

As you can see we retrieve the counterState and store it into state variable. Then we increment the turnCount property of the state. Then we Set the CounterSet After setting it, we are saving the conversation state.

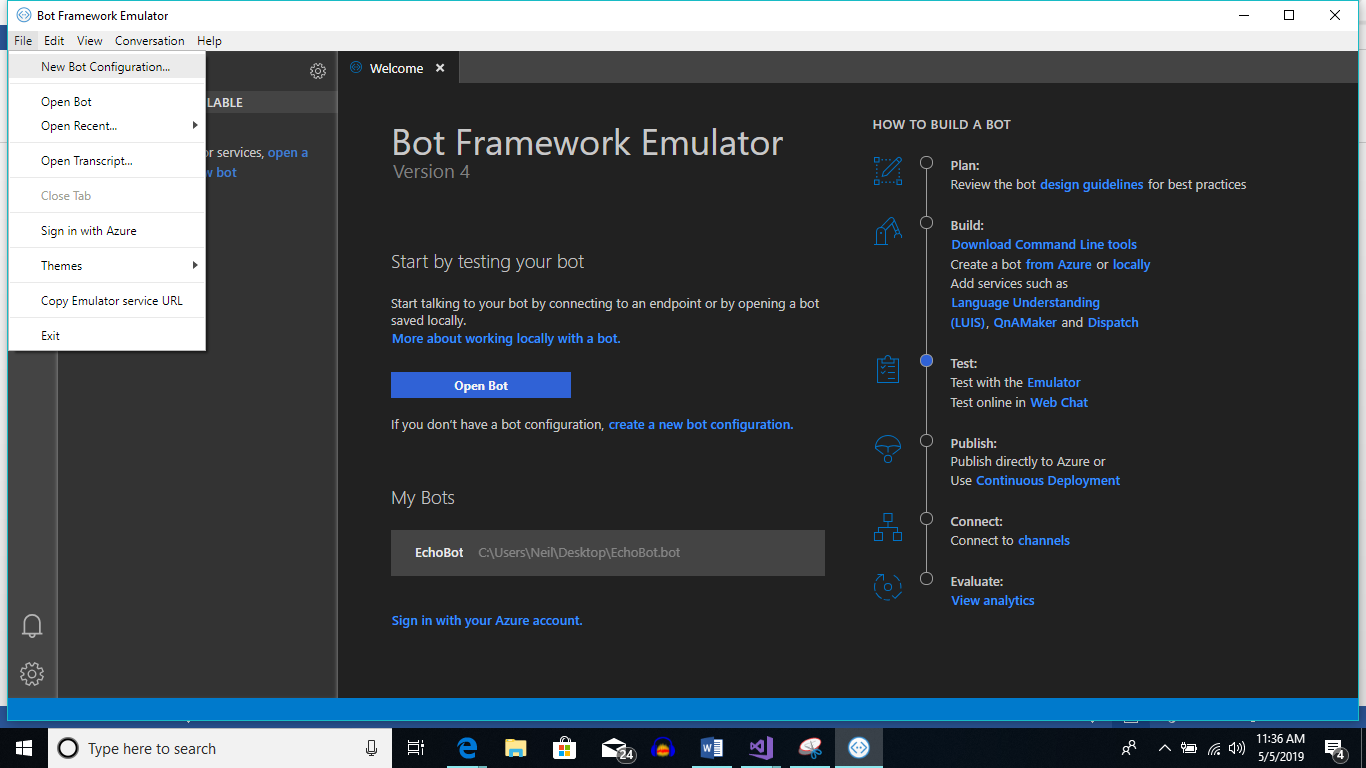


After that we are preparing the response message.

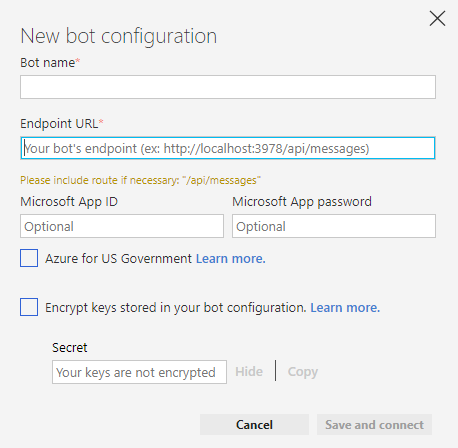
Then we send the message using SendActivityAsync method of turnContext.

**Running the bot.**

1. Run the bot in Visual studio
2. Open Bot Emulator and select File->New Bot Configuration

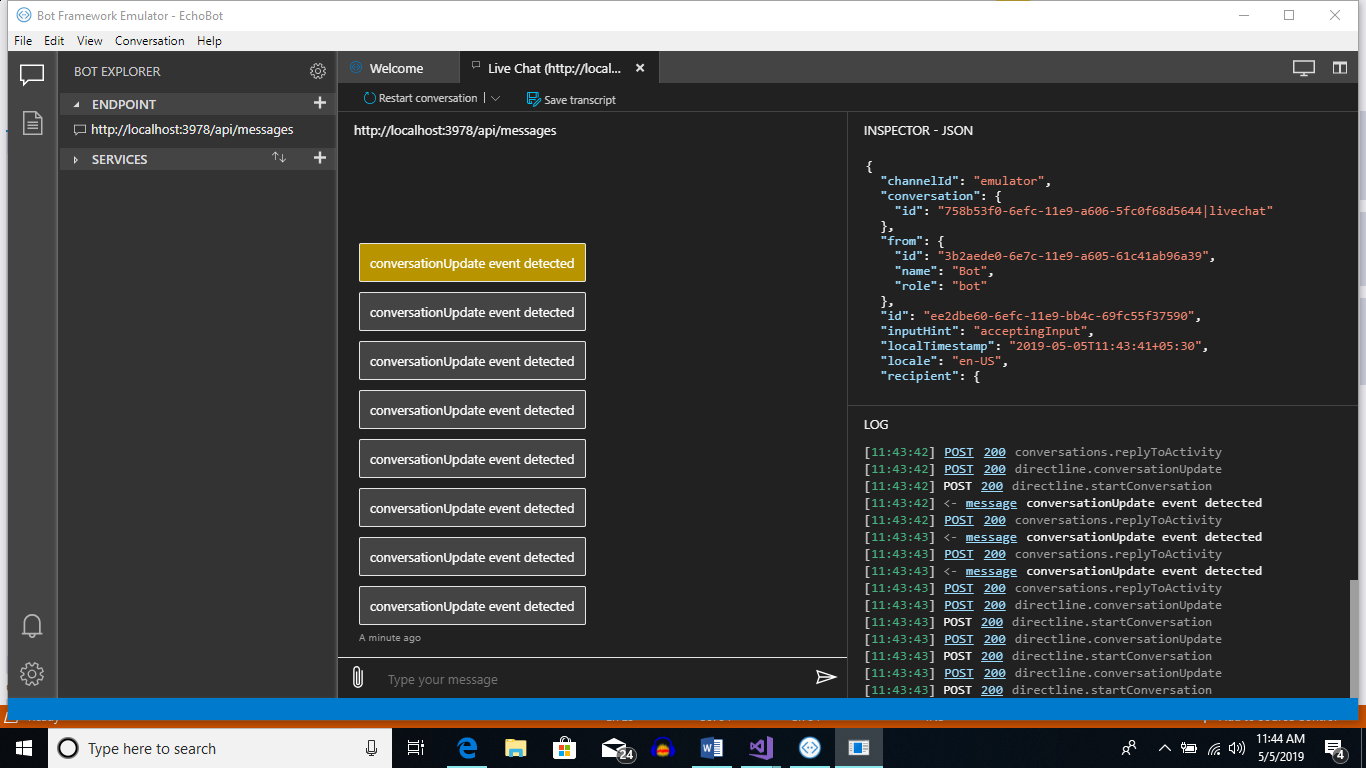


1. Put the endpoint address in configuration. Take it from .bot file





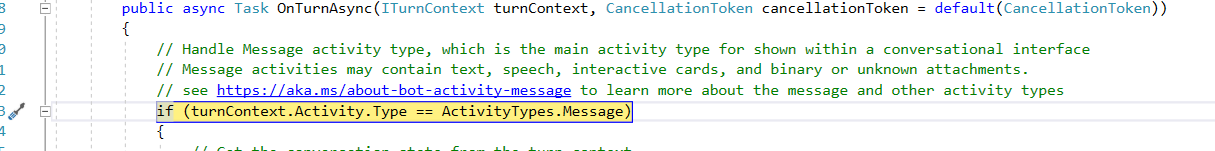
Iv. Then do a Save and connect.



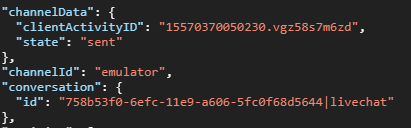
As you can see the first activity is Conversation Update when bot the parties join the chatroom. Below is the Json response which later gets de serialized into activity object



When I first type the message hi, the control comes to the OnTurnAsync method



If you look at the Json request, you can see the channel specific data



Then you can see the receiver, sender, the service url and the type of activity which is a message

