Workshop Transcript

Sunday, August 27, 2017 6:20 AM

A basic transcript of the Microsoft BOT Framework workshop held by Braindigit in association with AspNetCommunity.org at Insight Workshop on 2017-08-26

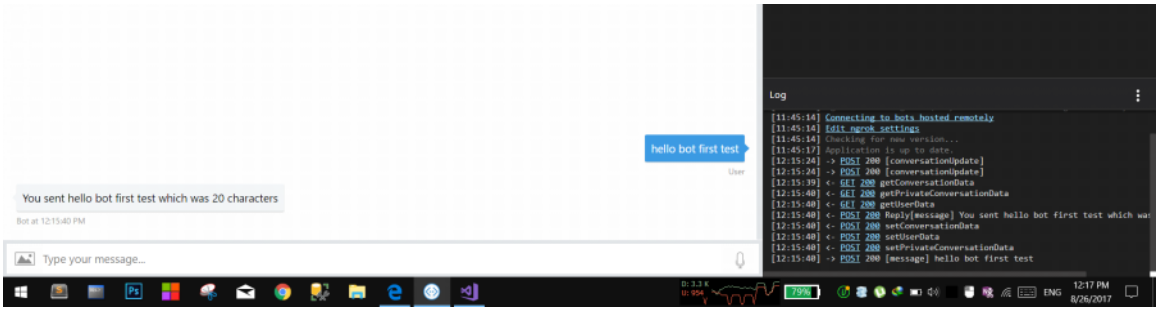
Mentor: Mr. Dev Raj Gautam
Event: https://www.eventbrite.com/e/microsoft-bot-framework-workshop-tickets-36889324964

BLOG: f5blogs.wordpress.com
|- Article
|- Source Code /w GIT

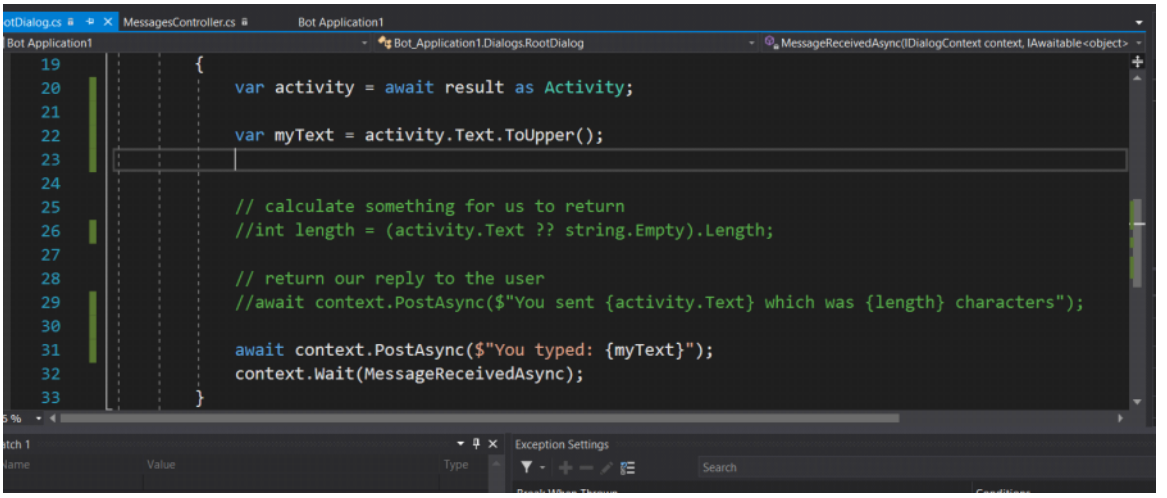
dev.botframework.com => register bot to work

use ngrok to deploy temporarily for verification

To test the bot, go to the appropriate URL in the emulator.
In this case,
http://localhost:3979/api/messages



Next task is to greet user without API initial hit



On MessagesController

```
        }
        else if (message.Type == ActivityTypes.ConversationUpdate)
        {
            // Handle conversation state changes, like members being added and removed
            // Use Activity.MembersAdded and Activity.MembersRemoved and Activity.Action for in
            // Not available in all channels
            ConnectorClient connector = new ConnectorClient(new Uri(message.ServiceUrl));
            var welcomeMessage = "Welcome to the Siddhant's News Bot";
            var reply = message.CreateReply(welcomeMessage);
            connector.Conversations.SendToConversation(reply);

        }
```

```
RootDialog.cs*   MessagesController.cs   Bot Application1
Bot Application1   Bot_Application1.Dialogs.RootDialog   newsSources
namespace Bot_Application1.Dialogs
{
    [Serializable]
    public class RootDialog : IDialog<object>
    {
        public Task StartAsync(IDialogContext context)
        {
            context.Wait(MessageReceivedAsync);

            return Task.CompletedTask;
        }

        List<String> newsSources = new List<string> { "BBC-News", "CNN", "Associated Press" };

        private async Task MessageReceivedAsync(IDialogContext context, IAwaitable<object> result)
```

ConnectorClient => in your channels, it send a reply; its an inbuilt class; used whenever you want to send a message to the user; namespace is using System

```
ConnectorClient connector = new ConnectorClient(new Uri(message.ServiceUrl));
```

Emulator hits the API twice so the ConnectorClient is hit twice and responds twice; Not a problem with messenger;
can be remedied in the Emulator also

```
//Remedy in the following way. Add this snippet with foreach() and a conditional
ConnectorClient connector = new ConnectorClient(new Uri(message.ServiceUrl));
    foreach(var member in message.MembersAdded)
    {
        if (member.Id != message.Recipient.Id) {
            var welcomeMessage = "Welcome to the Siddhant's News Bot";
            var reply = message.CreateReply(welcomeMessage);
            connector.Conversations.SendToConversation(reply);
        }
    }
```

Change

```
private async Task MessageReceivedAsync(IDialogContext context, IAwaitable<object> result)
```

to

```
private async Task MessageReceivedAsync(IDialogContext context, IAwaitable<IMessageActivity>
result)
```

The code is:(on //RootDialog)

```
    private async Task MessageReceivedAsync(IDialogContext context, IAwaitable<IMessageActivity>
    result)
  {
     var message = await result;
     PromptDialog.Choice(
        context: context,
        resume: GetNewsAsync,
        options: newsSources,
        prompt: "Please select your preferred news source:",
        retry: "I support the aforementioned BBC, CNN and AP at this time. Please select one of those
threee options"
        );
  }
```

Generate an API key from newsapi.org

Gen API Key = 7f29daf2862c4e3ba2956c754689df07

Scroll down on : https://newsapi.org/ and copy/paste the RESPONSE JSON

CONVERT THIS on json2csharp.com (beware of hyphens -)


```
{
"status": "ok",
"source": "the-next-web",
"sortBy": "latest",

"articles": [

{
"author": "Rachel Kaser",
"title": "The internet is losing its mind over Half-Life 3 (again)",
"description": "One of the main writers on the Half-Life series released a staid \"Epistle\"
depicting his version of the next game in the series -- and, in a reaction as Pavlovian as it ...",
"url": "https://thenextweb.com/gaming/2017/08/26/internet-losing-mind-over-half-life-3-
again/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2012/01/valve-
image-2-social.jpeg",
"publishedAt": "2017-08-26T00:16:44Z"
},

{
"author": "Tristan Greene",
"title": "These programmers taught an AI how to understand tattoos",
"description": "Artificial Intelligence is getting hip by learning about tattoos. A couple of
developers for the app Tattoodo wanted a better way to categorize all the tat pics they
receive, so they ...",
"url": "https://thenextweb.com/artificial-intelligence/2017/08/25/these-programmers-
taught-an-ai-how-to-understand-tattoos/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2017/08/AItattoo-
social.jpg",
```

"publishedAt": "2017-08-25T23:50:01Z"
},

{
"author": "Bryan Clark",
"title": "Confirmed: Apple to add wireless charging to the iPhone 8",
"description": "As we draw nearer to Apple's September iPhone event, the rumor mill only grows louder. And one of the loudest rumors, one we've been hearing since February, is that the next generation ...",
"url": "https://thenextweb.com/apple/2017/08/25/confirmed-next-iphone-to-offer-full-speed-wireless-charging/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2017/08/iPhone-8-encased-social.jpg",
"publishedAt": "2017-08-25T20:44:25Z"
},

{
"author": "Rachel Kaser",
"title": "Sorry technophiles, I really don't want a self-driving car",
"description": "Autonomous cars are coming, regardless of how I feel -- and sooner than I think. Elon Musk's lofty prediction that we'll have driverless cars by 2021 sounds absurd, but the evidence ...",
"url": "https://thenextweb.com/cars/2017/08/25/sorry-technophiles-i-really-dont-want-a-self-driving-car/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2016/05/Google-Self-Driving-Car-social.jpg",
"publishedAt": "2017-08-25T20:08:10Z"
},

{
"author": "Tristan Greene",
"title": "The trailer for Netflix's Black Mirror season 4 is a techno-horror delight",
"description": "Netflix today released the trailer for Black Mirror's fourth season. The 50-second long tour de force features the titles of all six upcoming episodes and a few tasty seconds from each.\r\n\r\nhttps://www.youtube.com/watch?v=oH85obU350E\r\n\r\nThe ...",
"url": "https://thenextweb.com/distract/2017/08/25/the-trailer-for-netflixs-black-mirror-season-4-is-a-techno-horror-delight/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2017/08/blackmirror-social.jpg",
"publishedAt": "2017-08-25T19:55:34Z"
},

{
"author": "Abhimanyu Ghoshal",
"title": "Discover the Game of Thrones universe with this handy interactive map",
"description": "Six seasons in and I'm still not great with the names of characters and places in the Game of Thrones universe. That makes me a horrible companion to binge-watch the show with: I'm ...",
"url": "https://thenextweb.com/shareables/2017/08/25/discover-game-thrones-universe-handy-interactive-map/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2016/07/Quartermaester-map-social.jpg",

"publishedAt": "2017-08-25T19:25:54Z"
},

{
"author": "Napier Lopez",
"title": "Facebook wants to fill up your notifications with even more memories",
"description": "Facebook is one of the easiest ways to take a stroll down memory lane, what with its endless collection of your ancient photos and comments. But while Facebook usually highlights memories ...",
"url": "https://thenextweb.com/facebook/2017/08/25/facebook-wants-fill-notifications-even-memories/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2017/08/SmartSelectImage_2017-08-25-10-15-58-social.png",
"publishedAt": "2017-08-25T17:23:53Z"
},

{
"author": "TNW Deals",
"title": "Become a pro coder without going back to school",
"description": "Kids are heading back to school — and you may want to think about joining them. Well, maybe not scrunched into those tiny desks or at the cafeteria lunch tables, but the pursuit of ...",
"url": "https://thenextweb.com/offers/2017/08/25/become-a-pro-coder-without-going-back-to-school/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2017/08/ac9O1fs-1-social.jpg",
"publishedAt": "2017-08-25T16:01:00Z"
},

{
"author": "The New School",
"title": "Reanimate your career through the power of creativity",
"description": "Sometimes when you've "arrived" in your career, you've simply arrived. Landing that dream job in a high-level, professional role is an impressive and noteworthy accomplishment—but ...",
"url": "https://thenextweb.com/insider/2017/08/25/reanimate-your-career-through-the-power-of-creativity/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2017/08/imagination2.jpg",
"publishedAt": "2017-08-25T15:00:10Z"
},

{
"author": "Boris Veldhuijzen van Zanten",
"title": "This hidden iOS function shows how much time you're wasting on which apps",
"description": "We all feel like we spend too much time on our phones. Notifications keep nagging us, we need to see what happens on Facebook or Twitter, and we feel a constant need to post to Instagram ...",
"url": "https://thenextweb.com/insider/2017/08/25/hidden-ios-function-how-much-time-wasting-apps/",
"urlToImage": "https://cdn0.tnwcdn.com/wp-content/blogs.dir/1/files/2017/08/image-5-2-social.png",
"publishedAt": "2017-08-25T13:55:36Z"

```
}
]
}
```

CONVERTS TO:

```
public class Article
{
   public string author { get; set; }
   public string title { get; set; }
   public string description { get; set; }
   public string url { get; set; }
   public string urlToImage { get; set; }
   public string publishedAt { get; set; }
}

public class RootObject
{
   public string status { get; set; }
   public string source { get; set; }
   public string sortBy { get; set; }
   public List<Article> articles { get; set; }
}
```

Now, back to VS2017. Add this to the BotApplication1

make these changes to the RootDialog:

```
//arko async method call gareko vayera this needs to change function type to async
    private async Task GetNewsAsync(IDialogContext context, IAwaitable<String> result)
    {
      string newsSource = await result;
      var newsArticles = await GetNews(newsSource.ToString());
      if (newsArticles.Status == "ok") {
        List<Attachment> list = new List<Attachment>();
        foreach (Article articleResponse in newsArticles.Articles) {
          var articleCard = new HeroCard() {
            Title = articleResponse.Title,
            Subtitle = articleResponse.Author+', '+articleResponse.PublishedAt,
            Text = articleResponse.Description,
            Images = new List<CardImage>{ new
CardImage(articleResponse.UrlToImage.AbsoluteUri)},
            Buttons = new List<CardAction> { new CardAction(ActionTypes.OpenUrl,"View Full Story",
value: articleResponse.Url.ToString())}

          };
        }
      }
    }


 private async Task<LiveResponse> GetNews(string newsSource)
    {
```

```
        LiveResponse liveResponse = new LiveResponse();
        try
        {
          var client = new HttpClient();
          var response = await client.GetAsync(string.Format("https://newsapi.org/v1/articles?
source={0}&apiKey=7f29daf2862c4e3ba2956c754689df07", newsSources));
          var content = await response.Content.ReadAsStringAsync();
          liveResponse = JsonConvert.DeserializeObject<LiveResponse>(content);
          return liveResponse;
        }
        catch (Exception)
        {
          return liveResponse;
        }
      }
```

The aforementioned is the herocard setup

AFTER CREATING AND TESTING BOT ON EMULATOR:

go to dev.botframework.com
register an account
you need an https domain to handle requests (ngrok)
Create new bot
Make a bot handle/BotUN
This creates an application ID that you can use in the Web Config of your BOT project folder
Configure likewise appropriately

You can get the gist of the workshop /w the materials in the following webpages. These blog posts closely follow the workshop(in order):

1. https://f5blogs.wordpress.com/2017/07/31/getting-started-with-bots/
2. https://f5blogs.wordpress.com/2017/07/31/getting-started-with-bots-ii-get-started-with-dialog-form-flow/
3. https://f5blogs.wordpress.com/2017/07/31/get-started-with-bots-iii-get-started-with-hero-card/

MICROSOFT BOTS

Getting Started With Bots -I ! (Create Your Basic Bot)

Date: July 31, 2017 0 Comments

In this series of 3 article, I will explain about getting started with Bots. You will learn about How to build bot using Microsoft Bot Framework. We will learn about getting started with bot and advance to build a bot that displays the Headlines from New site. You should be able to build a bot after completing the three articles.

Disclaimer: I have tried to keep the codes simple as possible hence might have skipped some good practices of programming

Download the codes Github URL **here and complete this 3 articles.**

Create Your Basic Bot: **will teach you the basics of bot and create a bot that echoes the user input. You will learn to add simple salutations to user of News Bot and understand basic codes.**

Get Started with Dialog & Form Flow: **You will learn and understand about the Root Dialog, Prompt Dialog and Form Flow. We will use prompt dialog to capture user input of the news source they want headlines of.**

Get Started with Hero Card: **We will make a web service call to News API and display the results in the Hero Card. We will learn about hero card and display the news in the bot.**

Setting up the environment

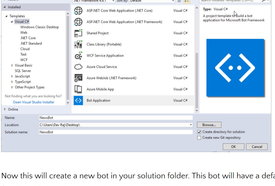
Download the latest Bot Template from the following link, you wont be able to see a bot template unless you have it installed properly.

Download Bot Framework Emulator Here . Bot Framework emulator enables you to test and debug the bots on desktop environment, you can test bots from local host or remote sites. This emulator also enables you to see the actual JSON calls made and received in your bot.

Getting Started With the Bot "News Bot"

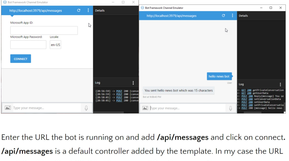
Create a new bot application Named "NewsBot" in visual studio.

File>New Project > Templates> Bot Application



Now this will create a new bot in your solution folder. This bot will have a default functionality to return your Message back to you. Lets see how to use an emulator to see how the bot works.

Press F5 or run the bot application. you would be able to see the application running in the browser selected in the Visual Studio. In my case the bot application was running in the URL **http://localhost:3979/** . You would need this URL to connect to bot emulator later. First lets connect the bot to emulator that you have downloaded and installed. Open the bot framework emulator and input details to connect to our local bot



Enter the URL the bot is running on and add **/api/messages** and click on connect. **/api/messages** is a default controller added by the template. In my case the URL was http://localhost:3979/api/messages. After the connection you would see HTTP 200 status **POST 200 [conversationUpdate]** in the log of the emulator.

Type some messages you would like and the bot will return **the characters you have entered and the length of the characters you have entered.**

Now that you have a bot application and already tested in the emulator lets start by modifying the code. **/api/messages** that we connected via emulator has some default code, open the **MessageController.CS** in you solution and see the following codes. let's try to understand the code.

```
public async Task Post([FromBody]Activity activity)
 {
 if (activity.Type == ActivityTypes.Message)
 {
 await Conversation.SendAsync(activity, () => new Dialogs.RootDialog());
 }
 else
 {
 HandleSystemMessage(activity);
 }
 var response = Request.CreateResponse(HttpStatusCode.OK);
 return response;
 }
```

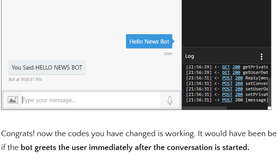
The default post method in this controller is receiving all messages from the user and invoking the Root Dialog inside **Dialogs\RootDialog.cs** that will process the message. For all other activities like *Message Being typed by user, User Joining the Conversation and other system events* are forwarded to HandleSystemMessage Method in the same controller, we will explore this later in this article. For now lets change the codes in root dialog to return the input text in Uppercase back to user i.e if user types "Hello News Bot" it would be returned as "You Said HELLO NEWS BOT".

open the **Dialogs\RootDialog.cs** and navigate to the **MessageReceivedAsync** method with following line of codes

```
private async Task MessageReceivedAsync(IDialogContext context,
IAwaitable result) { var activity = await result as Activity;// 
calculate something for us to return int length = (activity.Text ??
string.Empty).Length;// return our reply to the user await
context.PostAsync($"You sent {activity.Text} which was {length}
characters");context.Wait(MessageReceivedAsync); }Change the codes to
convert the received message to upper case and echoes back to the user.
```

```
private async Task MessageReceivedAsync(IDialogContext context,
IAwaitable result) { var activity = await result as Activity;// convert
the user message to uppercase string message = (activity.Text ??
string.Empty).ToUpper();// return our reply to the user await
context.PostAsync(String.Format("You Said
{0}",message));context.Wait(MessageReceivedAsync); }
```

Ok now we are pretty done with the first modification of the bots code. Go back to emulator and click on **Start New Conversation button (refresh)** and type some messages. You will see that you messages are being echoed back in Upper Case.



Congrats! now the codes you have changed is working. It would have been better if the **bot greets the user immediately after the conversation is started.**

Lets change the code in **HandleSystemMessage of MessageController.CS .** Change the following block **ActivityTypes.ConversationUpdate** to the following codes

```
else if (message.Type == ActivityTypes.ConversationUpdate)
  {
  foreach (var members in message.MembersAdded)
  {
  if (members.Id != message.Recipient.Id)
  {
  string welcomeMessage = "Welcome to News Bot, I will find the best News
Headlines For you";
  ConnectorClient connector = new ConnectorClient(new
Uri(message.ServiceUrl));
  var reply = message.CreateReply(welcomeMessage);
  connector.Conversations.SendToConversation(reply);

  }
  }
  }
```

ConversationUpdateis sent when the conversation's properties change, for example the topic name, or when user joins or leaves the group.

You might have noticed this line of code **members.Id != message.Recipient.Id** will make sure that the welcome message are only sent to the users that are connecting for the first time. Emitting this line ill cause the welcome message being sent 2 times.

The **Connector service** class uses industry-standard REST and JSON over HTTPS to send and receive messages to channels configured in Bot Framework.

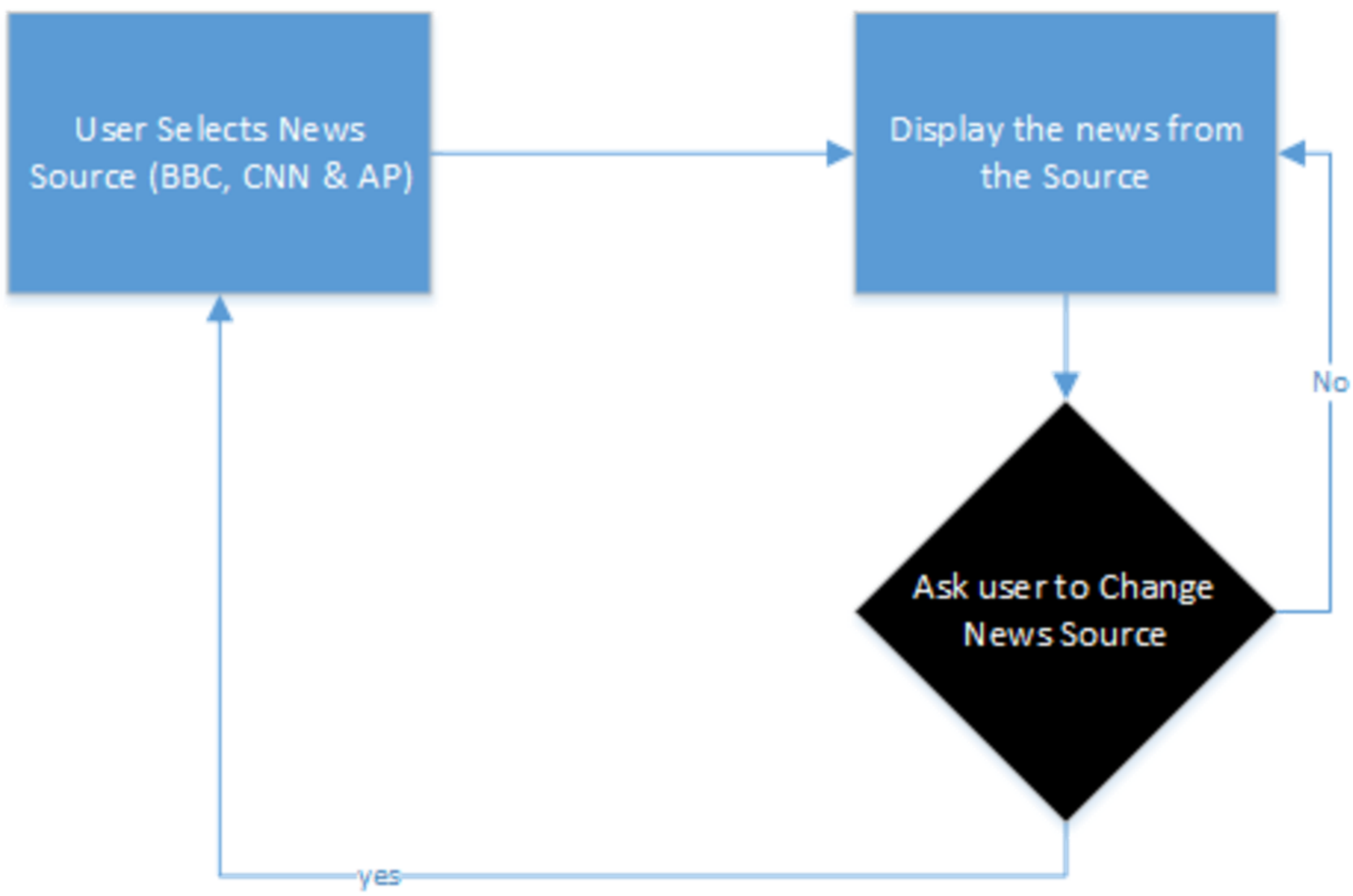
CreateReply will take a message and create a reply message for it with the routing information set up to correctly route a reply to the source message.
SendToConversation method allows you to send an activity to the end of a conversation.

MICROSOFT BOTS

Getting Started With Bots -II! (Get Started with Dialog & Form Flow)

Date: July 31, 2017 1 Comment

So now we have already a Basic Bot Now lets transform it to a News Bot that will allow users to read major articles from the bot. The flow of the bot would be



After the User will be greeted user will be asked to type "start" to a get started! You could also use your terms and conditions and other content here, could have also used some dialog form but to keep this article simple upto this stage lets make user to type "Start". Please change the Welcome message we created in Article 1 in **HandleSystemMessage of MessageController.cs**

```
string welcomeMessage = @"Welcome to News Bot
```

```
string welcomeMessage =@ Welcome to News Bot,
 I will find the best News Headlines For you.
 Please Type Start to get Started";
```

Let's change the code of our root dialog to evaluate the user's input is start or not. First Please change the Definition of **MessageReceivedAsync** Method in **RootDialog.cs** to the following.

```
private async Task MessageReceivedAsync(IDialogContext context,
IAwaitable<IMessageActivity> result)
 {
 var message = await result;
 if (message.Text.ToLower() == "start")
 {
 await context.PostAsync(message.Text);
 }
 else
 {

 await context.PostAsync("Please Type Start to Begin");
 }

 }
```

This Code will Simply Echo the "Start" text of the user, but provides a message to type start if inputted otherwise.

Now let's build a prompt that will ask for the user to choose the News source among the BBC, CNN & AP. user will be greeted and asked for the selection of the news source. We will use **a list of string and PromptDialog** provided by the Microsoft bot framework to create a **Prompt.** Let's create a new List<String>()

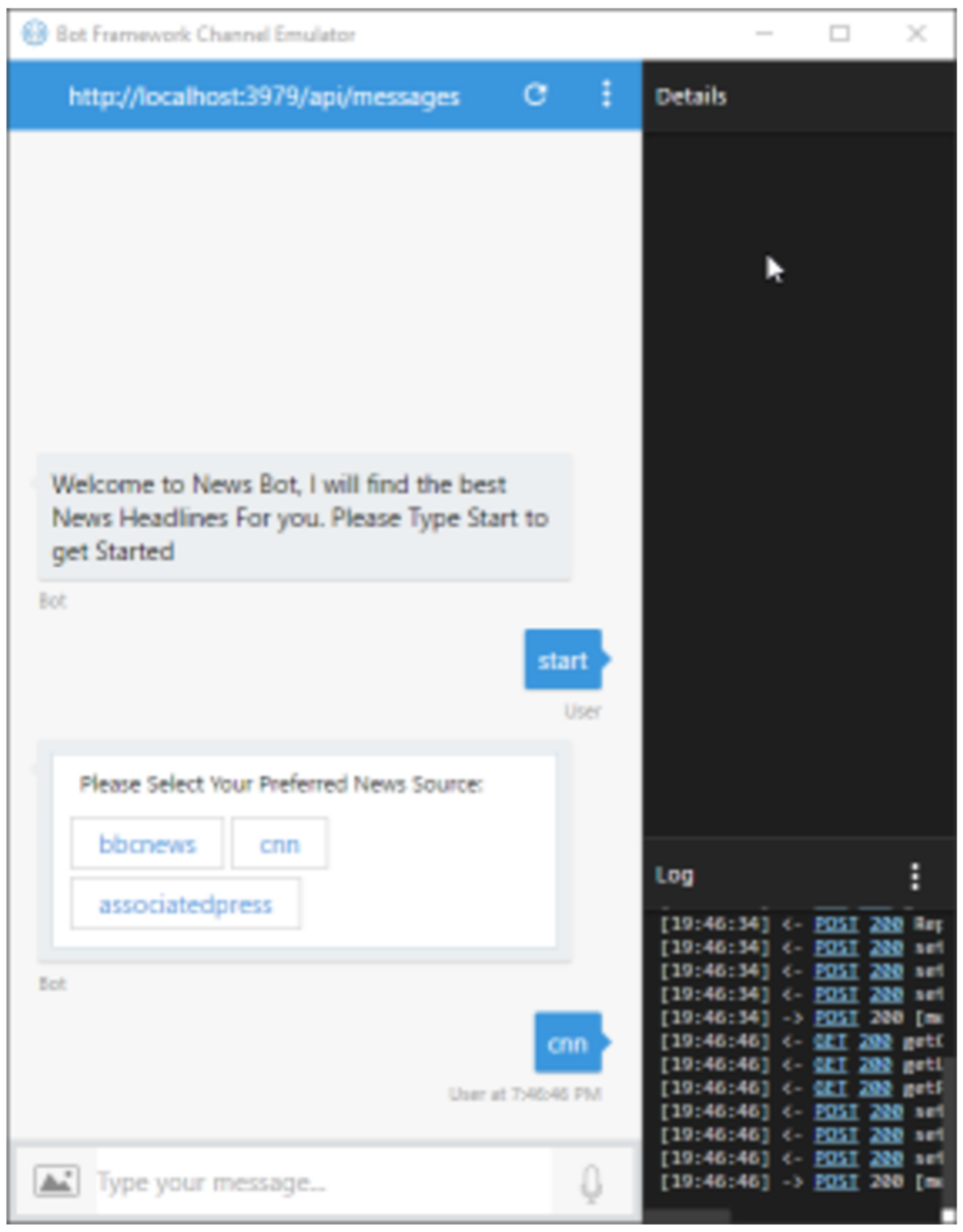
```
public List<string> NewsSources = new List<string>() {
 "bbc-news",
 "cnn",
 "associated-press"};
```

Now user will provided an a selection prompt to choose betwen bbcnews, cnn

Now user will provided an a selection prompt to choose between bbcnews, cnn, associatedpress. Change the code of the **MessageReceivedAsync** Method in **RootDialog.cs** to the following.

```
var message = await result;
 if (message.Text.ToLower() == "start")
 {
 PromptDialog.Choice(
 context: context,
 resume: GetNewsAsync,
 options: NewsSources,
 prompt: "Please Select Your Preferred News Source:",
 retry: "Currently I support only one of the News Source among:.");
 }
 else
 {
 await context.PostAsync("Please Type Start to Begin");
 }
```

Also create a new method called **GetNewsAsync** and keep it empty for now. Lets Go ahead and see what the application does.



So that's the way we wanted we have asked the user to select the news source and we have also taken the input from the user. We will further now pull the news

and we have also taken the input from the user. We will further now pull the news and display in the bot. First lets try to understand the codes here.

First we changed the **IAwaitable<object> result** parameter **to IAwaitable<IMessageActivity> result.** IMessageActivity basically lets u access the Message in the conversation. It has several properties and method and we used one of it by accessing the text similarly you could access attachment and other properties.

```
var message = await result;
message.Text.ToLower();
```

```
context: The dialog context.

resume: Resume handler.

prompt: The prompt to show to the user.

retry: What to show on retry

Options: Options to be provided to the user
```

MICROSOFT BOTS

Get Started With Bots- III (Get Started with Hero Card)

Date: July 31, 2017 1 Comment

Now to display the news we will use a pretty cool API that comes to you free. After the user has selected the news source **we will make a web service call to the https://newsapi.org/ and display the results via Hero Card in bots.** Please get your own key by registering in the site https://newsapi.org/.

Sample call for the API is as follows, please replace your key with the key you have received by registering in newsapiorg.

```
GET
https://newsapi.org/v1/articles?source=bbc-news&sortBy=top&apiKey=yourkey
```

Let's add 2 model class in the project to serialize the results . I have created a new folder called Models and a class LiveResponse.CS & Article.CS with following definition .

```
namespace NewsBot.Models
{
 public class Article
 {
 public string Author { get; set; }
 public string Title { get; set; }
 public string Description { get; set; }
 public Uri Url { get; set; }
 public Uri UrlToImage { get; set; }
 public string PublishedAt { get; set; }
 }
}
```

```
namespace NewsBot.Models
{
 public class LiveResponse
 {
 public string Status { get; set; }
 public string Source { get; set; }
 public string SortBy { get; set; }
 public List<Article> Articles { get; set;}
 }
}
```

Now Lets Create a method that will make the web service all and return the result in the list. Please add the following method to your Root dialog.

```
private async Task<LiveResponse> GetNews(string newsSource)
 {
 LiveResponse liveResponse = new LiveResponse();
 try
 {
 var client = new HttpClient();
 var response = await
client.GetAsync(string.Format("https://newsapi.org/v1/articles?source={0}&apiKey=94f312c927104585a9e98ebcfce44403", newsSource));
 var content = await response.Content.ReadAsStringAsync();
 liveResponse = JsonConvert.DeserializeObject<LiveResponse>(content);
 return liveResponse;
 }
 catch (Exception)
 {

 return liveResponse;
 }
 }
```

Please don't forget to change the apiKey in the URL. This method will populate a List of LiveResponse by making the web service call.
Now lets **bind the list results into HeroCard** of bot framework to display in the bot. Please Change the code of the GetNewsAsync Method we created earlier to the following codes.

```
public async Task GetNewsAsync(IDialogContext context,
IAwaitable<string> result)
 {
 string newsSource = await result;
 var newsArticles =await GetNews(newsSource.ToString());
 if (newsArticles.Status == "OK")
 {
 List<Attachment> list = new List<Attachment>();
 foreach (Article articleResponse in newsArticles.Articles)
 {
 var articleCard = new HeroCard()
 {
 Title = articleResponse.Title,
 Subtitle = articleResponse.Author + ',' + articleResponse.PublishedAt,
 Text = articleResponse.Description,
 Images = new List<CardImage> { new
CardImage(articleResponse.UrlToImage.AbsoluteUri) },
 Buttons = new List<CardAction> { new CardAction(ActionTypes.OpenUrl,
"View Full Story", value: articleResponse.Url.ToString()) }
 };
 list.Add(articleCard.ToAttachment());
 }
 var reply = context.MakeMessage();
 reply.AttachmentLayout = AttachmentLayoutTypes.Carousel;
 reply.Attachments = list;
 await context.PostAsync(reply);

 }
 else
 {
 await context.PostAsync("An Error Occured Calling NewsAPI.");
 }
 PromptDialog.Choice(
 context: context,
 resume: GetNewsAsync,
 options: NewsSources,
 prompt: "Do you want more News?,Please Select Your Preferred News
Source:",
 retry: "Currently I support only one of the News Source among:.");

 }
```

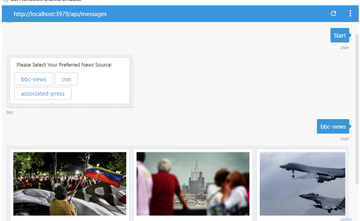
In the above code we make a call to **GetNews** method to fetch the list of news , and if the status is **"ok"** we further proceed to create **HeroCards and Show in Carousel**. HeroCard is initialized for each articles received. Let's explain Properties used in hero card

Title: Title of the card

Subtitle: Subtitle appears just below Title field, differs from Title in font styling only.

Text: Text field appears just below subtitle, differs from Subtitle in font styling only.

Images: Messaging supports all media formats: audio, video, images and thumbnails as well to optimize content download.

Buttons: Set of actions applicable to the current card. Not all channels support buttons or cards with buttons. Some channels may choose to render the buttons using a custom keyboard.

Here you go the news bot is ready 🙂 Congrats!!