International Journal of Computer Trends and Technology
ISSN: 2231-2803 / https://doi.org/10.14445/22312803/IJCTT-V68I1P101

Volume 68 Issue 1, 1-4, January 2020

Original Article

The Architecture of Bot Framework

Chandra Kudumula

Software Architect, Designer, and Developer. 1806 Rivendell Way, Edison, New Jersey, USA.

Received Date: 02 November 2019
Revised Date: 25 December 2019
Accepted Date: 29 December 2019

Abstract *- Even though the bot concept started decades ago, it came to the spotlight recently because the major players in the technology created robust frameworks to build fully-featured bots quickly and easily. One of the major players is Microsoft, and they created the Bot Framework.*

Microsoft bot framework has the Framework SDK and Framework portal, which makes bot framework a great tool to build chatbots. Framework SDK provides libraries to build the chatbot, and the Framework portal is the admin panel to register or manage bots. Significant components of Bot Framework are channels, bot connector, and Bot. The channel represents a place where a user can interact with a bot, and *Bot Connector enables Bot to exchange messages with users on channels. A bot is an app that interacts with users in a conversational way. Microsoft allows us to create bots using the Azure bot service also, but this paper covers Bot Framework SDK explicitly.*

Keywords *- Chatbot, Microsoft Bot Framework, Bot Framework SDK.*

I. INTRODUCTION

In general, any software framework provides classes and functions, which allows the developers to build the applications quickly and easily without worrying about building the systems from scratch. To effectively use the framework to build the applications, we have to know what features the framework is offering and its architecture of it.

Most recently, we have been listening to many frameworks to build chatbots, and one of those frameworks is Bot Framework, which was created by Microsoft. Bot Framework allows developers or coders to build applications called bots or chatbots. It is vital to understand the architecture of the Bot Framework to build an efficient chatbot using the Bot Framework and significant components in it and also the communication between the components. We will cover the overview of significant elements in the following sections.

II. DEFINITION OF CHATBOT

A Chabot is an application, often available via messaging platforms and using some form of intelligence that interacts with a user via a conversational user interface. This definition means, Chatbot application mostly takes input from a user, processes it, and responds to the user. Chatbots are available via messaging platforms like Skype, Messenger, Slack, Twilio, and others. Chatbots can leverage some form of artificial intelligence services available via Microsoft cognitive services and many other third parties. Chatbots are distinguished from traditional applications because they interact with the user via the conversational user interface. Conversation can be text and/or voice.

III. MICROSOFT BOT FRAMEWORK

Microsoft bot framework has two core parts that make it a great tool to build chatbots. They are:

- The Bot Framework SDK
- The Bot Framework Portal

The Bot Framework SDK is the set of language libraries (Microsoft.Bot.Builder, Connector, Configuration, etc.) that will help us to build the chatbot with a minimal amount of effort. The SDKs are available in .NET and Node.js platforms. The SDK is open source and hosted on GitHub at https://github.com/microsoft/botframework-sdk.

To test and debug bots built using the Bot Framework SDK, Microsoft provided a test tool called the Bot Framework Emulator for bot developers. The Emulator is the stand-alone app that provides not only a chat interface but also interrogation and debugging tools to help understand how and why your Bot does what it does. We can use the Bot Framework Emulator to test bots running either locally on our machine or connect to bots running remotely through a tunnel. The emulator repository is available at https://github.com/microsoft/BotFramework-Emulator.

The bot framework portal is the admin panel to register and manage chatbots. The portal lets us configure the chatbot to make it work with multiple channels or messaging applications. The link to register the Bot is https://dev.botframework.com/bots/new or App Studio. The bot framework portal currently supports the following channels: Facebook, GroupMe, Kik, LINE, Microsoft Teams, Skype, Skype for Business, Slack, Telegram, Twilio, WeChat, Email, Web Chat, and Cortana.

The coolest part of the framework is that it supports building bots across multiple channels using the same code base. What this means is when a message comes in from Facebook, it looks exact Same as when it comes in from slack. That way, we don't have to account for so many different variations of data, which makes our lives very much easier.

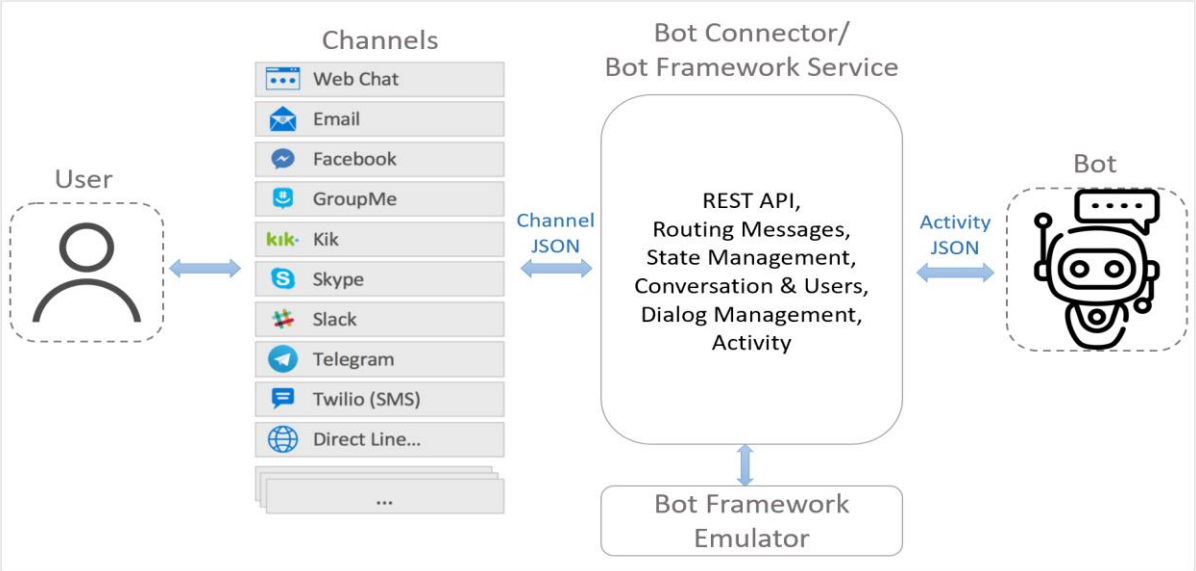

Fig. 1 Microsoft Bot Framework

Along with making the data coming in easier to read, it also makes the data going out easier to create with the support of rich attachments. This means if I want to send a carousel to slack, it will be the same code as the image carousel I build for Facebook. In addition to messaging applications, the Bot Framework supports email, SMS, and websites. The Bot Framework offers a web chat control that resides on a web page.

Apart from these features, Microsoft also provides a set of services called Cognitive Services, which we can use to add intelligence to our Chatbot. Cognitive services provide 21 different APIs, and they are separated into five domains like vision, speech, language, knowledge, and search based on their functionality. They are

1) Vision – APIs under the vision domain allows our bots to understand images and video content. They will retrieve information about faces, feelings, and other visual content.

2) Speech – APIs allow our bots to hear and speak to the users. APIs can filter noise and identify speakers. Based on the recognized intent, they can drive further actions.

3) Language – Allows bots to process natural language and learn how to recognize what users want.

4) Knowledge – Using these APIs, bots can explore different knowledge from the web or from academia or from our own data.

5) Search – These APIs give the ability to make bots more intelligent with the power of Bing and access data from billions of web pages, images, news articles, and videos.

IV. COMPONENTS OF BOT FRAMEWORK

Significant components of Bot Framework are channels, bot connector, and Bot.

A. Channel

The channel represents a place where a user can interact with a Chatbot. In other words, the channel is a funnel through which users will communicate with the Bot. A channel is often associated with a messaging app, like Skype, Slack, Facebook messenger and others. A channel can also be any program that sends and receives messages to and from the Bot Connector. Microsoft integrates with several third-party apps and has its own channels. Bot Framework supports building custom channels and connecting them with Direct Line API.

B. Bot Connector

Bot Connector service enables Bot to exchange messages with users on channels that are configured in the Bot Framework Portal. The service uses industry-standard REST and JSON over HTTPS and enables authentication with JWT Bearer tokens.

a) Activity

The bot connector uses the activity object to pass information back and forth between Bot and channel (user). The most common type of activity is the message, but there are other activity types, like Conversation update, Event, Invoke, Trace, Typing activity, etc. can be used to communicate various types of information to a bot or channel. The Bot adapter introduced in SDK version 4, which handles Bot Framework authentication, and the adapter is part of the activity processing stack. The adapter handles incoming and outgoing traffic between a channel and our Bot's turn handler, encapsulating the calls to the Bot Framework Connector.

b) Routing

The bot connector sends messages to and receives messages from channels, then sends and receives messages with the chatbot. Bot Connector facilitates communication between channels and chatbots, which is called rooting.

c) State management

Bot State management automates the reading and writing of Bot's state to the underlying storage layer. The storage layer types are memory storage, Azure blob, or cosmos DB storage.

d) Dialogues

Dialogs provide a useful way to manage a conversation with the user — each dialogue perform a specific task in a particular order. We can define the order of individual dialogues to invoke them in different ways and guide the conversation and - sometimes in response to a user, sometimes in response to some outside events, or from other dialogues. There are three dialogue types: prompt dialogues, waterfall dialogues, and component dialogues. Fig 2 shows the dialogue types and class hierarchy.

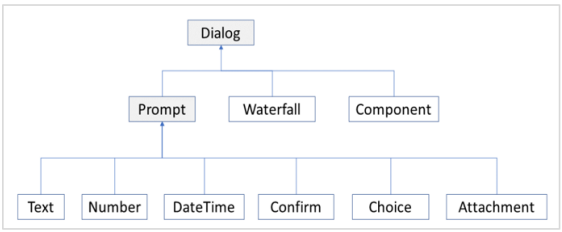

Fig. 2 Dialog types and class hierarchy

- The Prompt dialogue provides an easy way to ask the user for information and evaluate their response.
- The Waterfall dialogue is a specific implementation of a dialogue that is commonly used to guide the users through a series of tasks or collect information from the users.
- The Component dialogue provides a strategy for creating independent dialogues to handle specific scenarios, breaking a large dialogue set into more manageable pieces.

e) FormFlow

A FormFlow is a different dialogue. FormFlow is more declarative rather than procedural logic. Essentially, we specify what we want the conversation to be rather than how to perform the conversion. It will streamline building a bot that collects information from the user. For example, a bot that takes sandwich orders must collect several pieces of information from the user, such as type of bread, choice of toppings, size, and so on. Given basic guidelines, FormFlow can automatically generate the dialogues necessary to manage a guided conversation.

C. Bot

A bot is an app that interacts with users in a conversational way, using text, graphics (such as cards or images), or speech. The chatbot component is something the developer builds. The chatbot serves whatever purpose its creator decides. There's currently a growing list of chatbots for nearly any imaginable domain. E.g., entertainment, information, retail, gaming, team management, and more.

From a technology perspective, a bot is nothing more than an API endpoint that accepts data and returns a response. The messages that are sent by received to be proxied through the Microsoft Bot connector are JSON, which makes them human-readable and easy to log and debug.

From the view of a delivered product, a bot puts some tools in place to allow customers or staff to access business intelligence or issue commands relevant to business needs. It will interpret what the user is saying, look data up in the data stores, make decisions through the logic created, perform tasks on behalf of users or the system, and ultimately help a user to accomplish the task.

a) Bot's Life Cycle

Bot Framework provides an integrated set of tools and services for building bots. The framework provides tools for various stages of bot development to design and develop bots. Fig 3 shows the Bot's life cycle.

Fig. 3 Bot's life cycle

1) Plan

Before starting building or writing software for a bot, first understand the goals, processes thoroughly, and user needs. These are important to create a successful bot.

2) Build

Bot is a web service that implements a conversational interface and communicates with the Bot connector or Bot Framework Service to send and receive messages and events. A bot can be developed using the Bot Framework SDK or Azure portal.

Bot Framework offers additional features or components to extend Bot's functionality:

Feature	**Description**
Add natural language processing	Enable Bot to understand natural language, use speech, understand spelling errors, and recognize the user's intent.
Answer questions	We can add a knowledge base to answer questions users ask in a more conversational, natural way.
Manage different models	If more than one model is used, such as for QnA Maker and LUIS, intelligently determine when to use which one during Bot's conversation.
Add cards and buttons	Increase the user experience with media other than text, such as cards, graphics, and menus.

3) Test

Emulator is useful to test the Bot locally. The Bot Framework Emulator is the stand-alone app that provides not only a chat interface but also interrogation and debugging tools to help understand how and why Bot does what it does. The Framework Emulator can be run locally alongside your in-development bot application.

4) Publish

When the Bot is ready to be available on the web, publish the Bot to Azure or any data centre or web service. To have an address on the public Internet is the first step to Bot coming to life on the website or inside chat channels.

5) Connect

After publishing the Bot, connect it to channels such as Facebook, Kik, Messenger, Skype, Slack, Microsoft Teams, Telegram, text/SMS, Twilio, and Cortana. The Bot Framework does most of the work necessary to send and receive messages from all of these different platforms. The bot application receives a unified, normalized stream of messages regardless of the number and type of channels it is connected to.

6) Evaluate

Evaluate the data collected and improve the capabilities and performance of the Bot.

Finally, to ensure the Bot Framework Connector can only access the Bot's endpoint, configure the Bot's endpoint to use the only HTTPS and enable Bot Framework authentication by registering the Bot to acquire its appID and password.

V. COMPONENTS COMMUNICATION

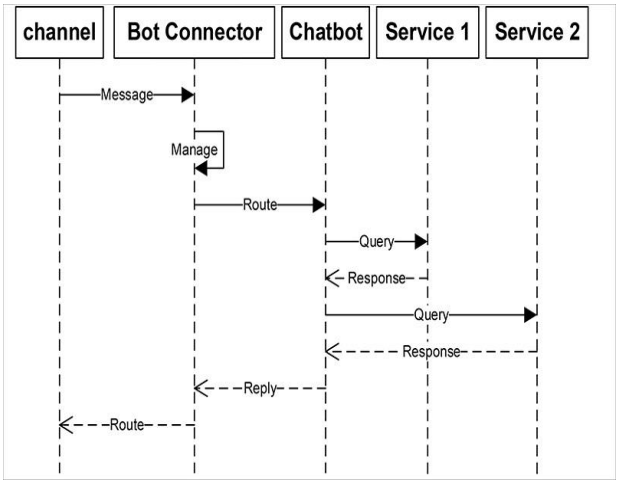

Fig. 4 communication paths between channel, bot connector, chatbot, and additional services

Fig 4 shows the communication paths between channel, Bot Connector, Chatbot, and additional services. Chatbot's are distributed applications. Each of the arrows between objects in Fig 4 represents a message being passed between components, and, more specifically, each of those arrows represents communication across the Internet. The communication occurring between channels, Bot Connector, and Chatbot is the normal part of the Bot Framework architecture. If the user has a slow Internet connection, their experience suffers. Alternatively, if the Internet connection is good, the user experience is better.

Fig 4 also contains an interesting set of features that developers must be aware of with services (labelled Service 1 and Service 2) that the Chatbot uses. It is normal to call external services, like Language Understanding Intelligence Service (LUIS), for natural language processing (NLP). There might be a couple of other services of interest, such as those offered by Microsoft Cognitive Services. If those services are necessary, by all means, use them. However, be aware of their nature and their potential to affect a Chatbot's performance and scalability.

VI. CONCLUSION

To date, the Microsoft Bot Framework is one of the best open-source Chatbot platforms, and it has everything to build an enterprise-level Chatbot. We understood some of the significant components and architecture of the Bot Framework. I hope this article will help someone who wants to understand the bot framework architecture and its components better.

REFERENCES

[1] Programming the Microsoft Bot Framework: A Multiplatform Approach to Building Chatbots by Joe Mayo.
[2] http://jameschambers.com/2018/03/bakebot/bots-day-0-Building-Bots-With-Microsofts-Bot-Framework/.
[3] https://dev.botframework.com/bots
[4] https://github.com/Microsoft/botframework-sdk/blob/master/specs/botframework-activity/botframework-activity.md
[5] https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-concept-state?view=azure-bot-service-4.0&viewFallbackFrom=azure-bot-service-3.0
[6] https://docs.microsoft.com/en-us/azure/bot-service/bot-builder-concept-dialog?view=azure-bot-service-4.0
[7] https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overview-introduction?view=azure-bot-service-4.0
[8] https://www.luis.ai/home
[9] Building Chatbots with Microsoft Bot Framework and Node.js by Akshay Kulkarni
[10] Learning Azure Cognitive Services by Leif Larsen.