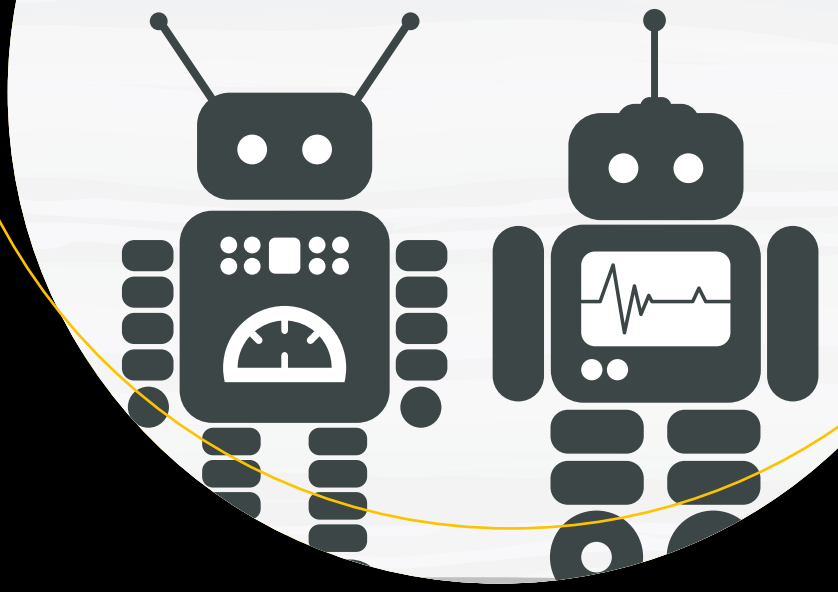

Developing Bots with Microsoft Bots Framework

Create Intelligent Bots using MS Bot Framework and Azure Cognitive Services

Srikanth Machiraju
Ritesh Modi

Apress®

Developing Bots with Microsoft Bots Framework

Create Intelligent Bots using MS Bot Framework and Azure Cognitive Services

Srikanth Machiraju
Ritesh Modi

Apress®

Developing Bots with Microsoft Bots Framework

Srikanth Machiraju
Hyderabad, Andhra Pradesh, India

Ritesh Modi
Hyderabad, Andhra Pradesh, India

ISBN-13 (pbk): 978-1-4842-3311-5
ISBN-13 (electronic): 978-1-4842-3312-2
https://doi.org/10.1007/978-1-4842-3312-2

Library of Congress Control Number: 2017962439

Cover image by Freepik (www.freepik.com)

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Nikhil Karkal
Development Editor: James Markham
Technical Reviewer: Puneet Jindal
Coordinating Editor: Sanchita Mandal
Copy Editor: April Rondeau
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, email orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please email rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-3311-5. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Contents

About the Authors

Srikanth Machiraju has over nine years of experience as a developer, architect, technical trainer, and community speaker. He is currently working with Microsoft Hyderabad designing and preaching modern application development using cutting-edge platforms and technologies. Prior to Microsoft, he worked with BrainScale as a corporate trainer and senior technical analyst on application design, development, and migration to cloud. He is a tech-savvy developer who is passionate about embracing new technologies and sharing his learning via blogs or community engagements. He has also authored *Learned Windows Server Containers*, blogs at https://vishwanathsrikanth.wordpress.com, and is active on LinkedIn at https://www.linkedin.com/in/vishsrik/.

Ritesh Modi is an ex-Microsoft Commercial software engineering senior technology evangelist. He is an architect, senior evangelist, cloud architect, published author, speaker, and a leader known for his contributions to Datacenter, Azure, Bots, Blockchain, cognitive services, DevOps, artificial intelligence, and automation. Currently, he is working as principle consultant with Infront Consulting. He is the author of multiple books, including `Azure for Architects` and `DevOps with Windows Server 2016`—both available on Amazon and B&N. He co-authored another book titled `Introducing Windows Server 2016 Technical Preview` with the Windows Server team. He has spoken at more than 15 conferences, including TechEd and PowerShell Asia conference, and is a published author for *MSDN* magazine.

He has more than a decade of experience in building and deploying enterprise solutions for customers. He has more than 25 technical certifications. His interests and hobbies include writing books, playing with his daughter, watching movies, and learning new technologies. His blog is available at https://automationnext.wordpress.com and his Twitter handle is `@automationnext`. He is based out of Hyderabad, India.

About the Technical Reviewer

Yogesh Sharma is an IT consultant based in Pune, India. He specializes in helping organizations with quality and cost improvement via migration and automation to various cloud offerings. His recent interest lies in collaborative learning. He would like to acknowledge Prachi, Celestin, and the entire editorial team for the opportunity and support.

Introduction

Bots are the new face of the user experience. Conversational user interfaces (CUI) provide a plethora of options to make the user experience richer, innovative, and appealing with Email, SMS, Image, Voice, or Video to communicate with the application. Modern web or desktop applications will soon be replace or augmented with intelligent bots that can be connected from anywhere using any device. Bots can use artificial intelligence and user data to provide richer insights and a personalized experience.

The Microsoft Bot framework has made the bot-building experience easy, and by using this framework we can build rich, scalable, and intelligent bots that can be connected from anywhere using an impressively vast list of platforms like Email, Skype, SMS, Facebook Chat and so on. This book explains how to develop intelligent bots using the Microsoft BOT framework, Visual Studio, Microsoft Azure, and Microsoft Cognitive Services. The preliminary chapters of the book deal with helping developers learn the basics of bot development using Visual Studio, .Net, C#, and Node.js. You will learn basic development and debugging skills, publishing to Azure, and configuring bots using the bot developer portal. The advanced section of the book deals with building intelligent bots using scalable storage, conversation flows, Microsoft Cognitive Services like LUIS, Bing Search, Vision and Voice API. This section also explains configuring analytics and other common Bot Operations.

The book is divided into the following sections:

Part 1 focuses on the need for a new communication platform and how conversation user interfaces (CUIs) break the barriers of building user interfaces; it also describes the current trends and future focus of this upcoming CUI and bot platform. This part also focuses on salient features of the MS Bot framework, available versions and features, and how the industry is embracing the change, and compares it with other competitive technologies and roadmap.

Part 2 teaches how to design and develop simple Skype bots on the Windows platform using the MS Bot framework, Skype, Visual Studio, .NET, and Azure.

In this part, the readers will also learn how to build bots using open source platforms like Node JS and VS Code. Readers will be shown how to manage the complete lifecycle of a Skype bot, like design, development, testing, and pushing to production.

Part 3 goes into prospective features of the Microsoft Bot framework, like channels and channel data and using rich text, buttons, media, and actions in chat messages. It also explains building a bot using the Skype Calling API and speech-to-text conversion, managing user data using Bot State Service, and using different conversation flows, like dialog model and form-flow model.

Part 4 delves into the details of building bots by integrating with Azure Cognitive Services, like Bing Search, OCR, and LUIS. We also focus on how to perform common operations, analytics, and diagnostics on bots in production environments.

Target Audience

The target audience for this book is C#/Node.js developers and architects who design and build modern applications using a Microsoft stack like Azure Cloud, Visual Studio, and code. Developers who want to get up to speed by learning the cutting-edge technologies that enrich the user experience and cater to multiple form factors. Architects/developers who wish to learn to build and design scalable and reliable messaging platforms that offer rich conversation experiences with the use of attachments, rich text, and voice for communication with enterprise applications. Developers can learn to integrate bots with machine learning and Cognitive Services offered by Azure. Business analysts and UX specialists can also learn to design trendy user interfaces by using bots and Azure ML that can be connected using any device and provide an enriched user experience to end customers. The target audience of this book do not need any prior bot-building experience.

CHAPTER 1

Conversations as Platforms

Have you ever had the experience of ordering pizza using an application that remembers your favorite pizza and orders to your current location automatically? Or have you ever booked a cab just by typing in a chat window or by using voice inputs and had a cab show up at your door step? If you have seen either of those, what you have experienced is the new generation of smart applications called bots (a short form of robots). Bots provide richer and more personalized experiences in our day-to-day activities, thereby making our lives much better. If you have not experienced this firsthand, you have yet to witness the next revolution in IT after the worldwide web, mobile, and data. Bots are much smarter than mobile applications; in some cases they can be smarter than you. Bots are designed to perform human-like interactions and exhibit human-like intelligence. Chat bots are not new; we have had platforms that help build chat-based applications for quite a few years (like Skype SDK), but what makes the new generation of bots special is their integration with artificial intelligence.

Over the years, smart devices and smart phones have become such an integral part of our life that they now hold lots of useful personalized information, like your favorite color, calendar, contacts, favorite restaurants, and so on. New-generation bots are designed to use the data and context surrounding the data with machine-learning (ML) and deep-learning technologies to give you a richer experience. A few decades ago, using ML or deep-learning technologies in a commercial application was highly complicated because they involve lots of new learnings and come with heavy computing and memory requirements. With the advent of cloud and serverless computing, the use of machine learning, data analytics, and advanced algorithms like facial recognition, voice recognition, and search is just a click away. The focus of this chapter will be on introducing the benefits of building conversations as a platform for all kinds of business needs; the Microsoft Bot framework, one of the top-class, end-to-end suites for building smarter, richer bots; and the various bot intelligence services and platforms available.

The following topics will be discussed in this chapter:

- Types of user interfaces
- Drawbacks of conventional user interface
- Conversations as platform
- Introduction to Microsoft Bot framework
- Meet a few bots

S. Machiraju and R. Modi, *Developing Bots with Microsoft Bots Framework*,
https://doi.org/10.1007/978-1-4842-3312-2_1

- Azure Bot Service
- Direct Line
- IOT and bot scenarios
- Other bot frameworks
- Bot abuse

Types of User Interfaces

User interfaces represent the face of any application. They should describe what the application can do and should be easy to use. It can be as simple as a command line where the customer interacts via commands or much more sophisticated, like a mobile application that can accept voice inputs from the user. Historically, user interfaces were more command-line or custom-input devices with a pre-defined set of commands printed on them; they could only perform a limited set of operations, like the interface shown in Figure 1-1. These types of interfaces had a limited set of responses and were not designed with the intelligence to respond to an unexpected random request. Some of them do not even retain any context of the user's previous conversations, which could be used to improve conversations with the returning user.

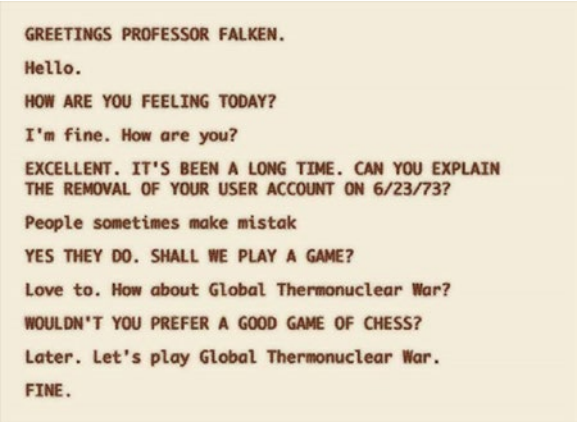

Figure 1-1. *WarGames: David Lightman talking with Joshua*

The most common user interface that we see today is called a graphical user interface (GUI), which was popularized by Xerox, Apple, and Microsoft during the 1980s.

Figure 1-2 shows the first GUI with a bitmapped screen. It was developed by Xerox in the year 1973 and was called Xerox PARC. In 1981, Xerox introduced Star and Workstation, which adapted Xerox PARC and influenced future innovations.

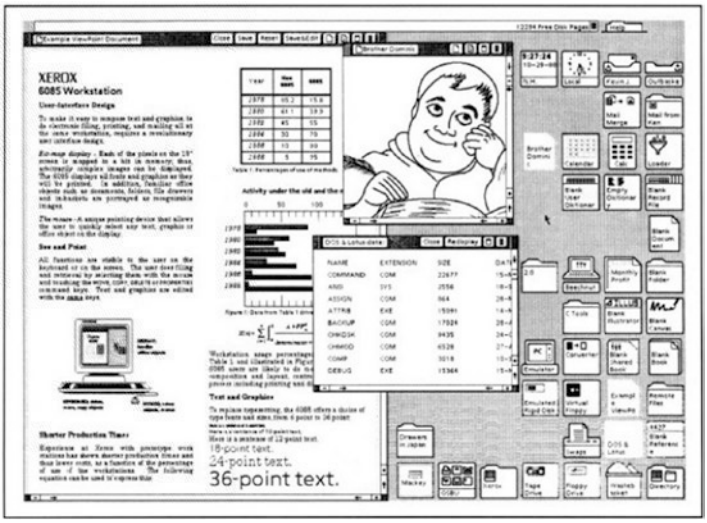

Figure 1-2. *Xerox Star user interface*

GUIs have evolved over the past few decades and are more sophisticated and easy to develop today. Modern applications are based on GUI, which are designed for desktop operating systems, web browsers, mobile, or kiosks. This, so far, is the easiest and most user-friendly form of interface, where the user interacts via button clicks (or touch) on a clickable element and uses an alpha-numeric keyboard, which can be used to input text, numbers, or symbols. Users are accustomed to these types of user interfaces, but haven't they become monotonous? Why would you want to ask the user to enter personal information like address, phone number, or favorite color every time; why would you need a call-center operator to answer the same questions from different customers? Why would you want to ask the user to install a mobile application or log in to your web application to perform a daily task like booking a cab, ordering your favorite food etc? Bots help you bring in a customized experience with natural language recognition and artificial intelligence—like voice-, image-, or video-based communication—to an application you are already using.

Drawbacks of Conventional UI

There was a reason we moved away from command-based interfaces to GUI, but what made us shift back to the old way? GUIs have their own shortcomings, and it is a challenge to build an effective user interface, be it for mobile or desktop. Not every GUI-based application makes optimum use of the screen space. For example, the GUIs in Figure 1-3 try to show all the available features or options on one screen, and for a first time user this can be overwhelming. It takes a while to figure out where to click and how to get the work done. These types of user interfaces force the business to include a readme document or 24/7 customer service, which can help users with queries about the portal's usage.

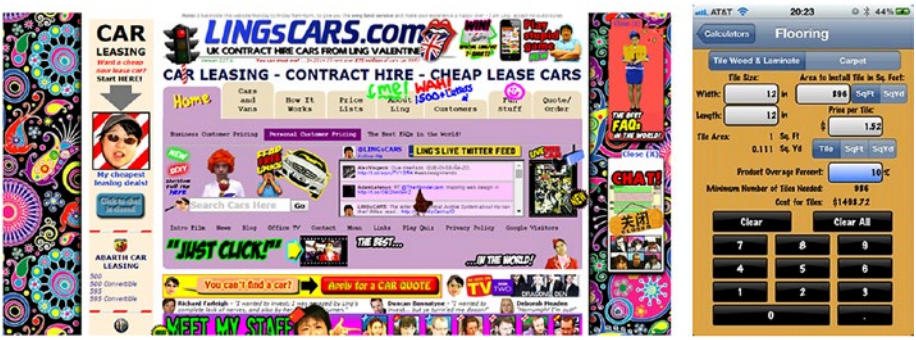

Figure 1-3. *Example of suboptimal screen design in a web and mobile application*

Mobile applications are constrained by screen space. The small screen size poses a challenge to building an effective application that is self-explanatory and covers all the operational aspects of the business. This restriction forces the stakeholders to design the application with only a few features on mobile and in parallel run a web version with a full feature set. It is also difficult to go through 20 clicks and a form-filling process on mobile for a few kinds of businesses. Web-based GUIs also have their limitations. Normally, the team involved in building these applications is huge and involves a UX designer, who works on the HTML + CSS, a development team, which builds the application logic, and database teams, which design the database schema. Bots, on the other hand, gel into existing chat interfaces like Skype, Slack, or Facebook chat, thus reducing the effort put into the application UI design so that the team can focus on building the application logic. For desktop applications, there is the additional challenge of the underlying software; this could be Java runtime, .NET, or node.js, which needs to be installed on the user's machine for the application to run. Desktop applications always target a specific operating system and version of runtime. It is highly difficult to build an application that works across a multitude of operating systems. Also, in a typical GUI application, the user moves from one screen to another; sometimes this makes it difficult for the user to understand and accept the flow and structure of the application (Figure 1-4). The infrastructure costs that are incurred when setting up the machines that run these applications or backend support systems is also huge. Chat bots do not need an enormous machine that hosts the interface; they can be designed to use existing applications to interact with business, reducing the application's running costs.

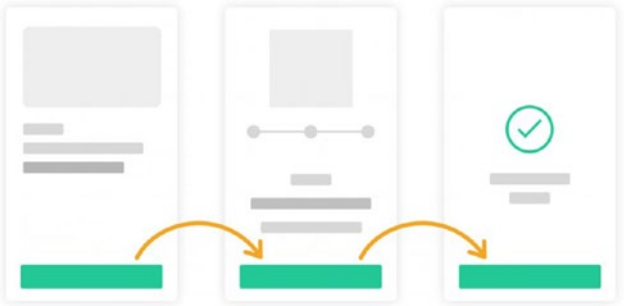

Figure 1-4. *Example of user experience in a typical GUI-based application*

Source: https://yourstory.com/2016/12/2017-year-conversational-user-interface/

The processing capabilities of today's computers have increased, most of the computations can be done at a large scale in real-time, and almost every task is achievable just by using a mobile phone or device. Why not change the way we interact with applications? Can we interact with these applications in a completely different way just by providing voice or text as inputs in a chat window, like you are talking to a friend? Can we build intelligent application interfaces that can use the context of my location, history, and personal preferences to finish tasks on my behalf; for example, a smart application that can track the upcoming football games of my favorite team and reserve a seat for me based on how my calendar is scheduled. Modern chat-based applications called bots are the key to the preceding questions. Bots are intelligent, context aware, and can hold more human-like conversations with the user.

Conversations as Platform

A conversational user interface (CUI) is any user interface that allows you to perform human-like interactions (Figure 1-5). What exactly is a human-like interaction? Human-like interactions use more natural language, which can be text, image, voice, or a combination of those. Human-like interactions are a conversational approach in which the application can understand and respond to what the user is saying regardless of the language used in the input; it can be in your own regional language in the most natural way, like the one shown in Figure 1-5. Conversation-based user interfaces make us realize that it is rather easy to book a flight or buy a shoe using more human-like interactions with a bot that is aware of my preferences and can even complete the payment on my behalf. It is also easier to build a conversation-based application because it does not contain any rich text, styling, or images to woo the user; all we need is to build an intelligent and simple conversation pattern that resembles a human.

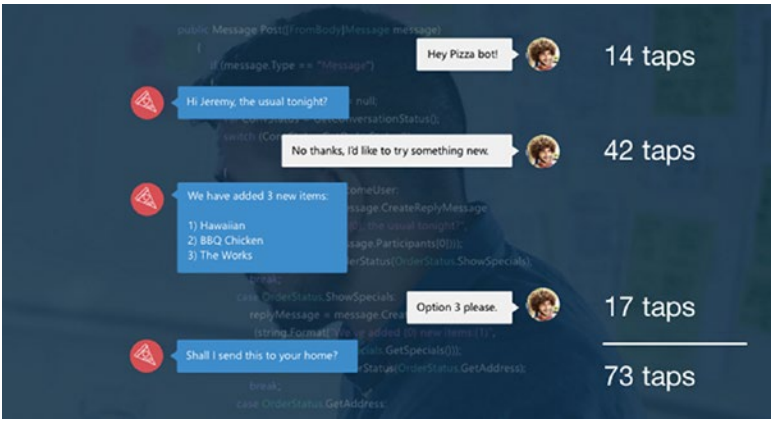

Figure 1-5. *Example of typical conversation-based user interface*

The most selling feature of CUIs is that everything happens right in front of you. You do not have to switch screens, navigate, or even scroll up and down to figure out a way to interact with the application. A simple gesture like Yes, No, or OK can get things done for you. CUIs are expected to be intelligent and context aware, and it makes no or less sense to have a CUI with limited set of responses.

A bot is a CUI-based software application that is automated to do a predefined set of tasks using human interactions (text and voice) through any medium, like a browser, desktop application, or phone. Bots are here to stay and are going to replace mobile apps. Like mobile applications replaced a lot of web-based applications, bots might replace mobile applications in the future with applications that can be interacted with using voice, text, or image. There are two categories of bots that are being built today: chat bots and AI bots. Chat bots are generally rule based. It is easier to build chat bots than AI bots, and they also consume less infrastructure and have lower costs. Most chat bots serve a single purpose, like a bot that imitates a pizza-ordering helpline with a limited set of menus.

The second type of bots are the artificial intelligence or AI bots. These bots are like chat bots, with the only difference being that they are backed by an artificial intelligence algorithm(s) that can predict your next action in an application based on usage pattern, or can recommend a similar item based on current selection and by analyzing what other users have bought together. Imagine just saying, "Repeat pizza order," and having a pizza delivered to your current location from your favorite pizza store. Most AI bots contain natural language processing and deep-learning capabilities. AI bots can sense the tone of the conversation and respond accordingly, which chat bots cannot do, as they are programmed with default responses irrespective of the tone and context. If you ask a chat bot an unrelated or random question, it might just deny the request or not respond at all, but an AI bot would try to analyze the question and answer as human. It would learn from the interactions so that it could respond to similar random questions. One more key distinguishing feature of AI bots is the source of data that drives decision making. Let us say you have a meeting scheduled for tomorrow 9 a.m. at Place A and you received an invite for lunch at Place B, which is about 10 miles away. An AI bot should be intelligent

enough to reject the invite because the traffic conditions will prevent you from reaching the lunch on time. It should also be able to auto-respond on your behalf based on the decision taken by the bot.

Bots are still emerging. As of today, there are only a few bots, which mimic web applications or mobile apps. The beauty of bots is that we do not have to build something from scratch or worry about installing it on a client's machine. Bots can be integrated into existing message platforms like Skype, Facebook Messenger, Slack, and so on. A few years ago, technologies like artificial intelligence and machine learning were out of reach to most developers because of the effort involved in learning the language and the semantics. Microsoft Cognitive Services has helped us overcome these challenges by introducing a multitude of intelligence-based APIs that are effective and easy to consume.

Introduction to Microsoft Bot Framework

The Microsoft Bot framework is a complete suite to build intelligent and intuitive bots that will be reachable via familiar communication tools like Skype, Slack, Teams, Office 365 Email, and other popular ones without any additional effort required. Communicating with a bot resembles a human-to-human communication and occurs in various forms, like sending a text or an email. The framework consists of a powerful Bot Builder SDK, Bot connector service, a developer portal, and a bot directory. The Microsoft Bot SDK is available for both Node.js and C# developers; for other languages, developers can use the REST API to build intelligent bots. The framework provides support for user management, session management, state management, authentication, and conversation models like dialogs, activities, cards, or attachments. The Bot SDK is open source.

To add more human-like conversation features to your bot, you can integrate with Microsoft Cognitive Services, which provides vision APIs for image processing, speech APIs for voice-to-text translation and vice versa, language APIs for language conversation, Search APIs for including Bing search in the results, and knowledge APIs for building recommendations based on a user's previous usage.

When you are done building your bot, you register it with the Bot registry and configure connectivity with a variety of channels before you finally publish. Figure 1-6 shows a few channels currently supported by the Microsoft Bot framework.

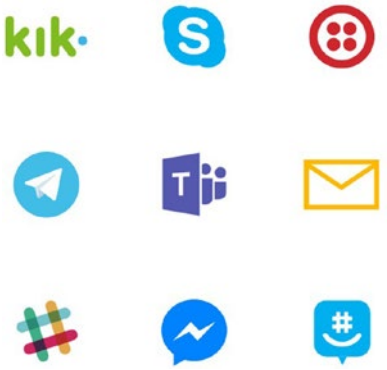

Figure 1-6. *Bot channels*

Microsoft provides a bot developer portal to connect your bot to various channels and test on those channels. The Bot Connector service uses the REST API and JSON schema to communicate with the Bot API. Once you have configured your channels and published your bot for testing, the bot ends up in a bot directory (Figure 1-7). A bot directory is a list of bots published by developers from across the globe. You can search for and connect to any bot available in the directory by using your favorite channel.

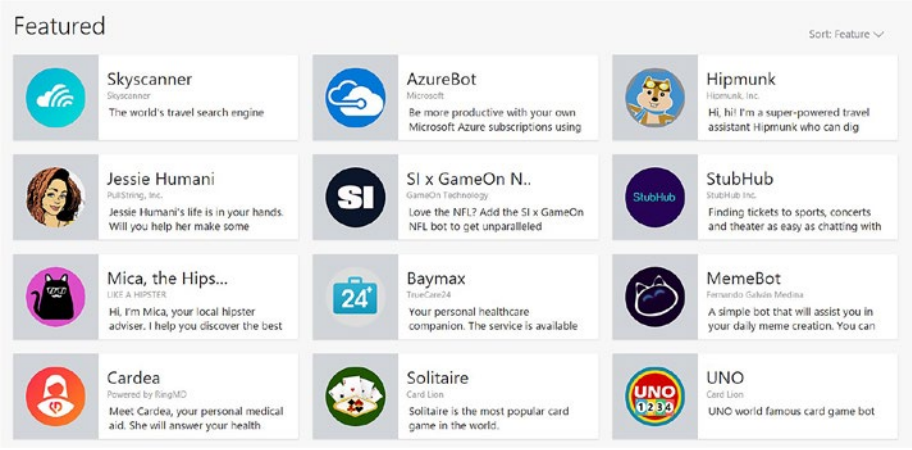

Figure 1-7. *Bot directory*

Every bot in a directory is configured with supported channels, so you can just click on any bot and connect using the configured channel. For example, the UNO Bot only contains Skype as a channel, so you can click on UNO Bot and then click on Add to Skype to add the bot to your Skype bot contacts list. The latest version of Skype, called Skype Preview, isolates the bots into a separate bot contacts category. You can also search

for bots using the Skype Search feature like any other user. The latest version of Skype Preview can be downloaded from the Windows App Store.

Meet a Few Bots

The bot directory consists of many featured bots that are designed for a specific cause—and some bots are just fun to chat with (specially the AI bots)! In this section, let us meet a few interesting bots and get a feel for the arena before we start designing bot applications. The bot directory is available under the Bot Directory tab in the Bot Developer Portal, found at `https://bots.botframework.com/`.

Summarize

If you are feeling too lazy or could not find the time to read a lengthy blog, Summarize Bot can help by providing a summary of the blog or any web page. The user can paste the link of the blog, and Summarize Bot will summarize the blog and list the important points. There are a couple of ways you can interact with Summarize Bot. From Skype Preview, you can search for Summarize Bot and add it to the contacts list, or you can visit the bot directory and use the web chat window. Figure 1-8 shows a sample conversation with Summarize Bot using Skype Preview.

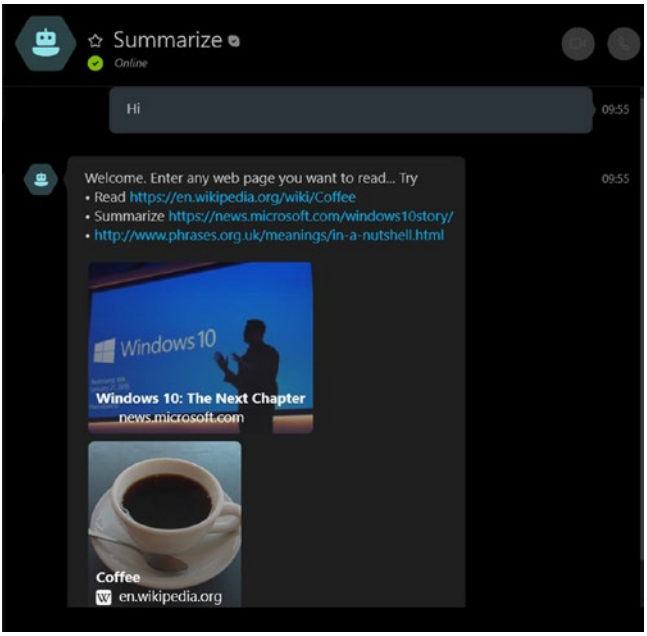

Figure 1-8. *Summarize Bot*

Figure 1-9 shows the response from Summarize Bot when any link—for example, `https://docs.botframework.com/en-us/azure-bot-service`—is sent as a request.

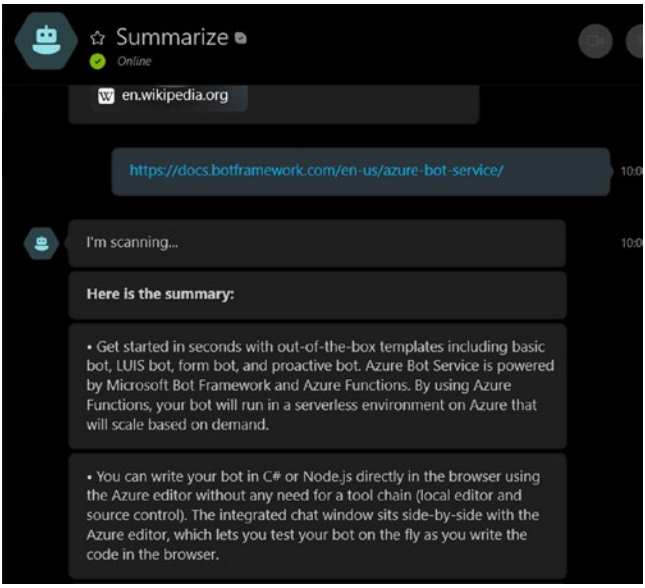

Figure 1-9. *Summarize Bot response*

Feedback from users is critical if an application is to be improved. Summarize Bot follows best practices by seeking information from the user as to whether it did well.

Your Face

Your Face is an AI-powered bot that uses Microsoft Cognitive Services to assess the face in the image and predict the age. **Your Face** is available on a variety of channels, like Skype, Telegram, Kik, email, GroupMe, and Facebook Messenger. Figure 1-10 shows an interaction with the bot using Microsoft Outlook. You can send any picture with a face as an attachment to `yourface_bot@outlook.com`.

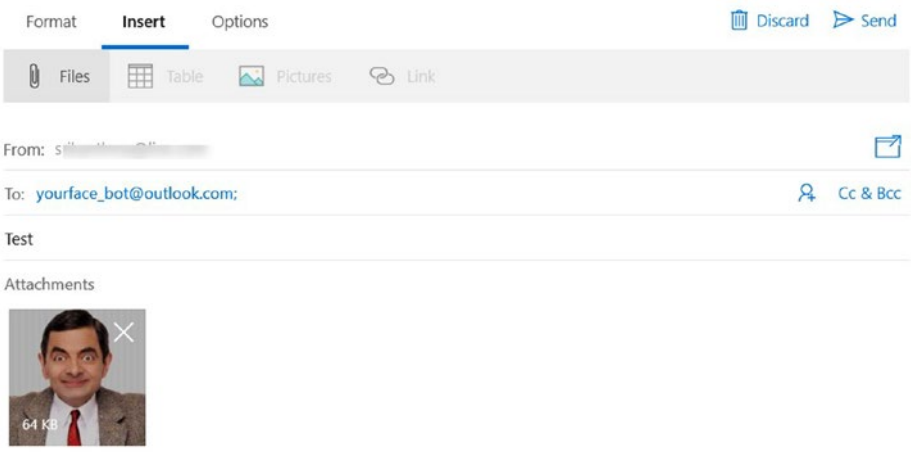

Figure 1-10. *Interacting with Your Face AI bot using email client*

Within a couple of minutes, the bot responds with an email. As you can see in Figure 1-11, the bot puts a name to the face since it is a familiar one; at the same time, it predicts the age of the face. You can try with any of your personal images and have some fun.

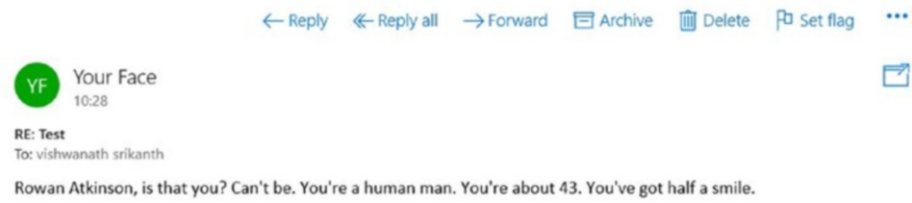

Figure 1-11. *Sample response from Your Face AI bot*

Azure Bot Service

Azure Bot Service is a PaaS (Platform as a Service) offering from Microsoft that is available as part of the Azure subscription. Azure Bot Service enables rapid application development powered by the Microsoft Bot framework and runs in a serverless environment on Azure. Azure Bot Service allows your bots to scale on demand and pay only for resources you consume. To create an Azure Bot Service, one would need an Azure subscription. You can buy one or create a free trial account for learning purpose from here: `https://azure.microsoft.com/en-us/free`.

Azure Bot Service provides an integrated development environment that helps you register the bot right from the Azure portal and allows you to author code with boilerplate templates. Figure 1-12 shows the bot configuration on Azure Bot Service.

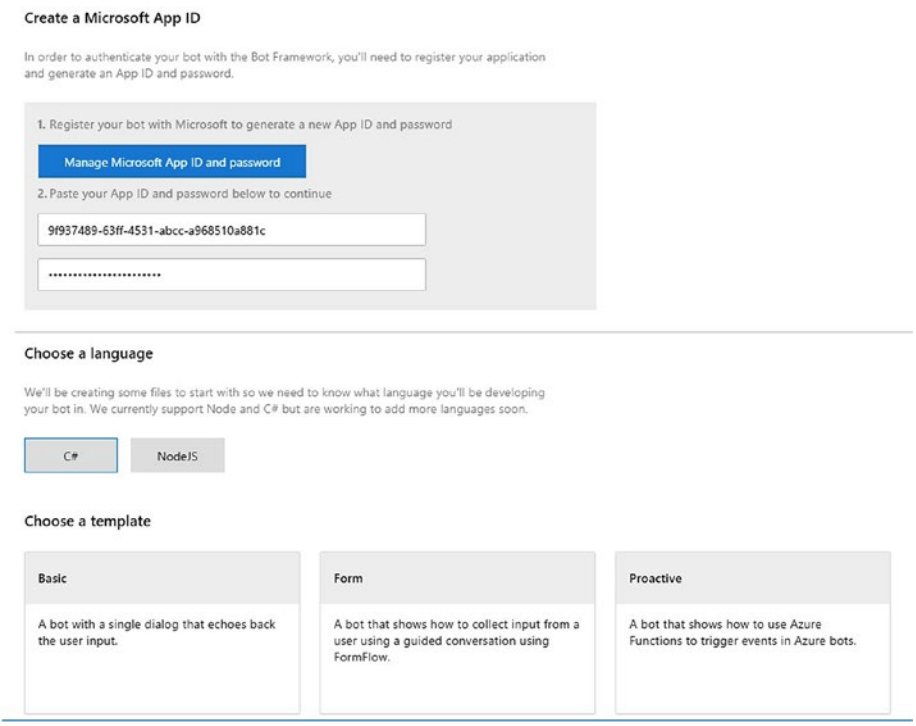

Figure 1-12. *Azure Bot Service*

You can select your favorite language and start developing bots by choosing any existing template. Azure Bot Service offers boilerplate templates, from simple bots to intelligent ones like those with natural language processing, proactive bots, and question and answer bots. Since it is on Azure, there is no overhead when managing servers or even patching. The bot can be scheduled to scale based on events powered by Azure Functions. By using Azure Functions, your bot runs on a completely serverless environment that scales on demand. Azure Bot Service provides in one place all the required resources for development, channel configuration, bot settings, a web-chat interface for testing, and a publishing service for publishing, as shown in Figure 1-13.

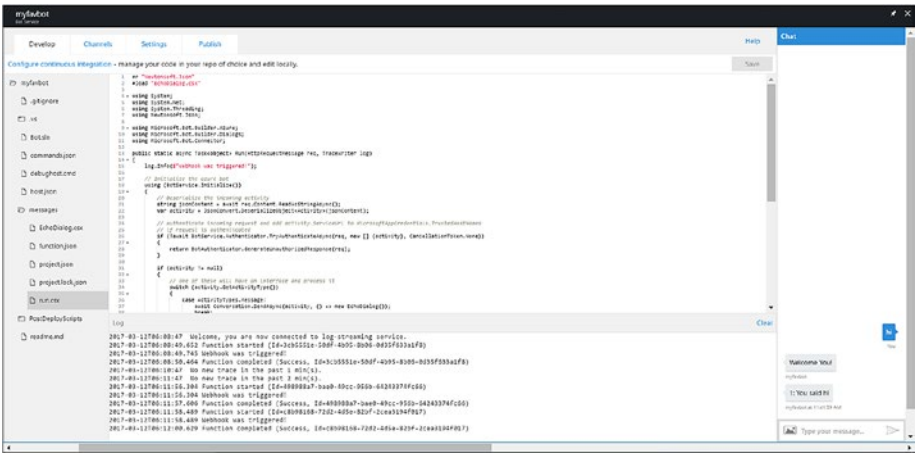

Figure 1-13. *Azure Bot Service development experience*

From Azure Portal, we can set up Continuous Integration from GitHub, Visual Studio Team Services, and many more. We can use the code from the web interface as a start, but remember that we cannot modify the code in Azure after setting up Continuous Integration. Continuous Integration and Delivery options let you deliver the application code at a rapid pace so the code gets built and deployment for every commit from the developer.

The following are a few intelligent bot templates available on Azure Bot Service.

LUIS Bot

For an intelligent bot, it is important to understand the user's conversation in the natural language. LUIS, which stands for *Language Understand Intelligent Service*, helps you identify the intent and entities in the conversation and map them to pre-defined HTTP endpoints. For example, in a statement like "get score about India versus England cricket," the bot should be able to get the intent, which is "get score" and the entities which are India, England, and Cricket LUIS enables you to design HTTP endpoints and map the user conversation to HTTP endpoints. For more information on integrating with LUIS, please visit `https://www.microsoft.com/cognitive-services/en-us/luis-api/documentation/home`. You will learn more about building bots with LUIS integration in Chapter 9.

QnA Bot

Most businesses have a QnA or FAQ section that helps users find answers to repetitive questions on the business model or on how to use the application. The bot template for QnA allows you to quickly create a FAQ or QnA bot that can answer a user's queries about your business via various channels using existing FAQ content as the knowledge base. QnA Maker (`https://qnamaker.ai/`) lets you ingest your existing FAQ content and

expose it as an HTTP endpoint. You can also build a new bot with an empty knowledge base. Figure 1-14 shows a sample QnA bot made from a FAQ URL knowledge base.

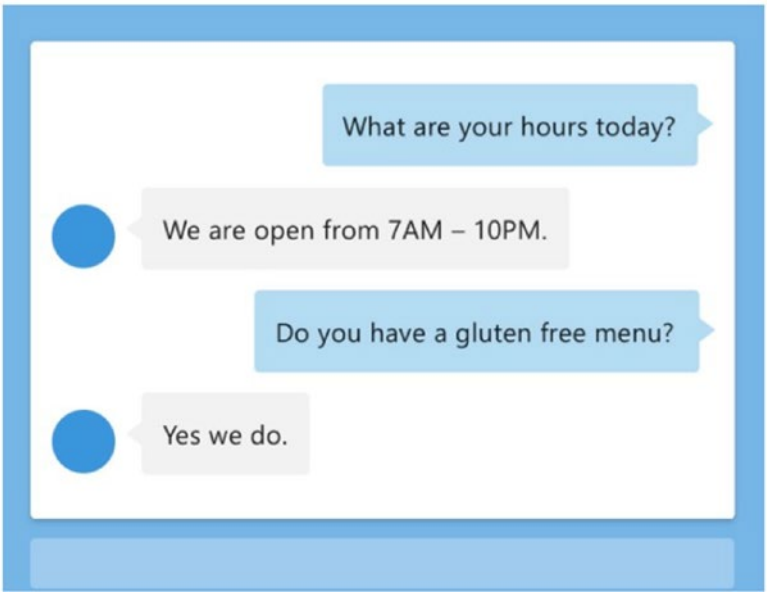

Figure 1-14. *Sample QnA bot*

Proactive Bot

The Proactive Bot template helps you in scenarios where you want the bot to initiate a conversation. The bot can initiate a conversation based on some triggered event, lengthy job, or external event like updating a cricket score on completion of a bowler's over. The Proactive Bot template uses Azure Functions to trigger an event when there is a message in the queue. Azure Functions then alerts the bot via the Direct Line API. The proactive bot template creates all the Azure resources you need for enabling the scenario. Figure 1-15 shows an overview of how multiple Azure resources communicate to trigger an proactive conversation.

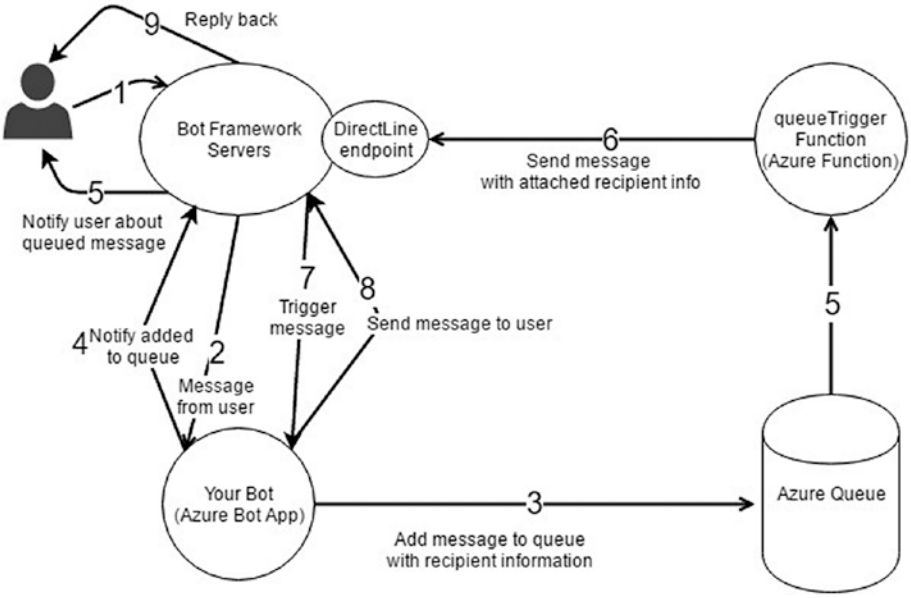

Figure 1-15. *Proactive Bot design*

Direct Line

The Direct Line API is a simple REST API that allows you to initiate a conversation with a Bot. This helps developers write their own client applications for web, mobile, or service-to-service. The API has the ability to authenticate using secret tokens or token patterns. Using the Direct Line API, you can send messages from your client to your bot via HTTP post messages. In addition, you can receive messages from the bot using Web Socket Stream or by polling mechanism. If you are planning to build any custom smart device that responds to a user's voice inputs or gestures, you can use the Direct Line API to send and receive messages from the bot, backed up by AI services from a custom IOT device (Figure 1-16).

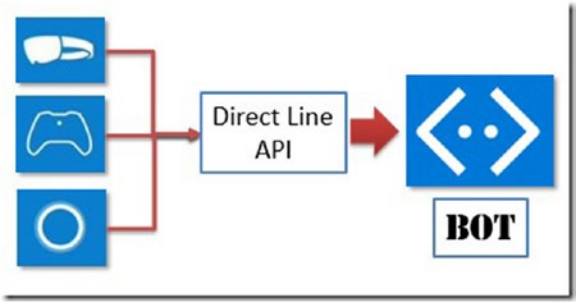

Figure 1-16. *High-level design for designing bots with custom clients*

IOT and Bots

Bots can integrate with IOT (Internet of Things) devices to make our lives much more sophisticated. Microsoft Azure provides services for building smart IOT devices, like Azure IOT hubs that can process millions of IOT assets. For example, imagine you are running a restaurant. A bot can proactively ping you with an image of a guest with the details like name, gender, age, recurring customer, favorite food (by using the Bing Search API and facial-recognition APIs) using a IOT device installed at the entrance. This model can be further extended by using the Azure ML, Stream Analytics, and Power BI to build a recommendations-based menu for recurring guests. Such is the power of Azure and the services provided by Microsoft as part of Azure.

Microsoft Cortana, Apple Siri, and Amazon Echo are a few examples of smartness being built into your handheld devices using various sources of information. With the Azure Bots framework and IOT hubs, you can use any existing device to build smart apps that accept voice input and respond with a voice or text response that includes information from a wide variety of sources.

Other Bot Frameworks

It is worth mentioning the other bot frameworks that are like the MS Bot framework and provide a similar feature set; for example, Wit.ai, API.ai, and Viv. The April 2016 Facebook Bot engine release is based on Wit.ai, which it acquired in 2015. Wit.ai runs on a server hosted on the cloud. The Facebook Bot engine is a wrapper that helps developers build bots for Facebook Messenger only. Wit.ai works like LUIS, as it helps extract intents and entities and it also comes with few pre-defined entities. API.ai also works along the lines of defining entities and intents, though the defining feature of API.ai is its reachability. API.ai bots can be exported as modules and integrated in Facebook, Kik, Slack, Alexa, and even Cortana. Viv.ai is from the authors of Apple Siri, and its focus is to process complex queries using a flexible mechanism. Viv follows the SQL query-processing approach as application logic, which involves breaking the request into constituents and combining them into an execution plan. Viv, as claimed by the makers, is more suitable for building virtual assistants than an enterprise bot.