Creating a Virtual Assistant

IBM HANDS-ON CODELAB

Miguel Crisanto and Miriam Oglesby

Codemotion Developer Conference, Berlin – Nov. 13th 2019

Introduction

This hands-on tutorial guides you through the implementation of a Virtual Assistant using the Watson Assistant service provided within the IBM Cloud Public platform.

The tutorial includes two parts which you may complete in the allocated time according to your pace. The first part provides the steps to implement an *Assistant for store location and opening hours information*. The second part uses the components created in the first part and extends the exercise to allow for *Processing delivery orders*.

Part 1: Assistant for store location and opening hours information

1. In order to access the Watson services, you need to create and log into an IBM Cloud account.

   Start by going to https://ibm.biz/BdzkyV

   Register or login to the IBM Cloud with your IBM ID credentials.

   Once you have logged in, you will see your *Dashboard*.

2. To create an instance of the Watson Assistant, click **Create Resource** in the upper right corner.

   From here, you can see all the various tools and services that are offered in the IBM Cloud catalog. Click the **AI** link, which will filter out all the other services.

   Find and select the **Watson Assistant** service.

3. The screen that follows provides some information about the service, images, pricing plans, and a button to create the service on your Dashboard.

   Select **Frankfurt** as the region where your Service should be created.

   If you scroll down to the **Pricing Plans**, you can see what the features and prices are for each level of service. The **Lite** plan offers everything we need to get started, so make sure that option is selected.

The **Service name** field needs to be unique across all users, so try adding some characters to the end if you customize the name and get an error that the name is already in use.

Click the **Create** button on the right.

4. The next page presents the instance of the service (it's also accessible through your Dashboard).

   From this page, you can view the instance details of your service and you can also review the credentials that are required to access that instance through the REST API.

   Other helpful links on this page include a tutorial to help you get started, and a link to the API reference.

   The three vertical dots in the upper right corner provides you with a way to access the service documentation.

5. We want to start building a Watson Assistant, so click **Launch Watson Assistant**.

   The *IBM Watson Assistant* home page provides access to the major elements for creating the Watson Assistant: *Assistants* and *Skills*.

   "*My first assistant"* is provided to start creating your assistant. Click the three vertical dots in the upper right corner to display the options. Click on **Settings** to rename the assistant and (optionally) provide a description. Press **Enter**.

6. Each assistant requires at least one *Skill* (previous service versions define *Workspaces* instead of *Skills*). Each *Skill* acts as a collection of intents, entities, and dialog that define the capabilities of the assistant.

   Click on the **My first skill** square to start creating the artifacts for your assistant.

7. Before we begin building anything, let's quickly review our user scenario. We are creating an assistant that can answer questions about our stores. Take a moment to think about what kind of store you want to use in your example. Maybe you sell groceries, pet supplies, or electronics. Consider your customers and create a mental picture of the assistant that you think best relates to those customers. When you write a dialog, think about how your assistant (your store representative) would talk to a customer.

   There are two requirements we have for our assistant. First, it needs to be able to tell the store locations, and second it needs to be able to give the store hours.

   Each of these requirements is a great example of an *intent*, meaning that a customer is likely to ask where the stores are and what the hours are.

   Let's start by creating an *intent* for the store locations. Click **Create intent**.

8. An important aspect of using *Watson Assistant* is to have a good naming convention for your intents, entities, and environment variables, which will make it easier for you to find and use them as your *Skills* space grows.

   You may want to name our first intent *store_location*. The description should be the golden standard of what this intent means, so you can ensure your user examples remain consistent. As description you may use *Things users say to find a store*.

9. Watson Assistant learns each intent through *examples*. These examples should be representative of what real users of the system would say. Your examples should also contain logical variations. For example, someone looking for a pet store might ask, *"where can I get a dog"* rather than say they are looking for a store.

   We need a minimum of five examples to train the intent. More examples are generally better, but the number you need is really dependent on how many different ways your users will state the intent. Punctuation and capitalization are ignored for training purposes. Try creating your own examples specific to your store.

   Add each example by typing it into the **User example** field and pressing **Enter** or clicking **Add example**. The examples I'll use for this intent are:
   Where is your store?
   How do I get to your store?
   Where are you located?
   I need directions.
   What street are you on?

   Notice that as you add examples, they are alphabetized so they are easier to find later.

   Also notice that *"I need directions"* is not a question. Intents just need to match what a user might say to convey an idea.

10. When you are done, click the **left arrow** in the upper left corner to return to the intents page.

11. Our second intent is to help a user find the store hours. Click **Create intent**.

    Enter an intent name of *store_hours*.
    For description, use *Things users say to get the store hours*. Click **Create intent**.

12. When asking about store hours, a user might ask when a store opens or when it closes. If this were an office, they may ask if it is closed for lunch. Think about what your users would ask, and then enter five examples. I'll use the following examples:

Are you open on weekends?
What are your hours?
What time do your doors open?
When do you close?
When do you open?

Click the **left arrow** to return to the intents page.

13. We now have our two intents with 5 examples for each. Next, we'll add some *entities*. Click the **Entities** link on the left.

14. Entities are contained in a user example and can be used to refine a user's intent. In our example, we have a store on Elm street and a store on Maple street. We can use entities to tell when a user is talking about either of these locations.

15. Entity names are designated with an @ symbol. In the **Entity name** field, enter *street*.

    The entity name is not used for matching, so it can be completely different from any of the associated values. The values are what get matched, and we want to match *Elm* and *Maple*. Enter the *values* one after the other pressing **Add value** after each of them.

    Sometimes an entity has a synonym you want to recognize. For example, a user might say Elm, or Elm Street, or Elm Road. We want to recognize those as all being the same location, so we'll set up some synonyms. Click **Elm**.
    Add synonyms for *Elm Street* and *Elm Road*.

    Now do the same for *Maple*.

    Synonyms are helpful for matching exact strings. In some cases, you may want to match patterns or use fuzzy matching to allow for misspellings. You can get information about Fuzzy Matching or turn on the feature using the option in the upper right corner. If you enter an entity as a *pattern* (click the triangle right of **Synonyms**), then you define it with regular expressions. This type of entity is helpful if you want to recognize telephone numbers, IP addresses or other specially formatted text.

    Now that we have our entities defined, click the **left arrow** to return to the main entities page.

16. In addition to your custom entities, there are also built-in system entities. Click the **System entities** link.

    System entities are defined by IBM, so they cannot be edited. Using them is as simple as turning on the ones you want to use and checking for their use in your *dialog* nodes. They have a reserved naming convention that starts with *sys-*.

Some typical uses for system entities include checking for dates, currency, or numbers. Check the documentation to see the full list of system entities available for use.

17. Go back to **My entities**.

18. Now that we have added some intents and entities, let's make sure they are working. Open the **Try it** panel in the upper right corner. We'll be going back to this panel from time to time to make sure things are working right. The *Try it* panel is a tool that helps you configure your virtual assistant and is not a UI available to end users. You will need to create your own user interface or connect your assistant to an existing web page or chat service to use it in a production environment.

19. First, let's test our intents using the *Try it* panel.

    Start by asking one of the questions you trained on, exactly as it was entered into your user examples.
    Write: "*Where is your store?"*
    The Assistant was able to recognize this intent as #store_location.
    How about, "*Are you open on weekends?"*
    The Assistant was able to recognize this intent as #store_hours.
    We would expect the Assistant to know the examples it was trained on, but the real power of the AI is asking in ways it hasn't been trained for. Think of a few more ways to ask for a location or hours using new terms.
    I'll try, "*Where are you?"*
    It correctly guessed that I'm looking for the store location.
    I'll try, "*I want to find your store"*
    Again, it guessed #store_location.

    Let's try something unlike any examples it was trained on. *"Will you let me in at 4:45?"*
    With the #store_hours intent trained on only five examples, and none of them mentioning letting me in or a specific time, the Assistant was able to correctly understand that I was asking about the store hours.

20. If the Assistant does get an intent wrong in the *Try it* panel, fixing the error is as simple as clicking the drop-down box and selecting the intent you want to change it to.

21. Now let's look at the entities.
    Enter *"Elm"*, *"Elm Street"*, *"Maple"* and *"Maple Road"*
    For each of these examples, the intent is marked **Irrelevant** because there is not enough information to determine an intent. Each of the entities, however, is correctly identified.

22. For one more example, let's test an intent with an entity. Ask, *"Where is your Elm Street location?"*

The intent is #store_location, and the entity is @street:Elm. It looks like everything is working.

Close the **Try it** panel.

23. While you are on the **My first skill** interface, click the **Content Catalog** link on the left.

    Watson Assistant includes several commonly used categories of intents that can be added to your *skill* panel. These additional intents can help you save time when building Assistants in the various listed domains, like banking, customer care, eCommerce or Telco. Click **General** to see what is included in this category.

24. Every Intent included in each category has multiple user examples so that you can immediately start using that intent in your dialog nodes. The *General* category includes such common intents as *ending the conversation*, *requesting a human agent* or *providing positive* or *negative feedback*. Add this category to your *skill* by clicking **Add to skill** in the upper right corner.

25. Click the **left arrow** to exit this page.

26. Click the **Intents** link.

    From this page, you can see the intents that were imported from the catalog. They all start with the word *#General_*. Now that these have been imported, you can edit the examples, add your own or delete the intents you do not intend to use.

27. Click the **Dialog** link.

    Dialog is used to control the flow of the conversation. Each of the white boxes you see here is an individual *dialog* node. By default, your dialog will contain two nodes. The *Welcome* node is activated only the first time a conversation is started. The *Anything else* node is last in the list and gets activated when all other nodes fail to activate.

28. Click the **Welcome** node to select it.

    When you select a *node* or *folder*, the editor appears on the right side.

    Nodes have several features that determine how they function. The first section contains the *node name*. The name is a label to help you determine the function of the node and to provide a destination when jumping from one node to another.

    The second section contains *conditional statements*. If the conditional statements for a node evaluate to TRUE, the node will respond with whatever is in the third section of the node, called the *response*. The *Welcome node* has a special condition called *welcome*. The variable called *welcome* is only true the first time the conversation is started.

After the response, the fourth section of the node determines whether the node waits for the next user input or jumps to another node in the dialog.

In this *Welcome* node, the Assistant will start by saying, *"Hello. How can I help you?"*. Let's customize this node to better reflect our use case. Change the introduction to whatever works best for your store.

I'm going to make it say, **"Hi, I'm the Watson Assistant, and I'm here to tell you about our stores."** The reason I start with that line is because I want the users to understand what the bot can do without forcing them to ask.

There is also space to add variations of the response. I'll add an alternate of **"My name is Watson. What can I tell you about our stores?"**

When you have variations of responses, you can determine if the responses are sequential, or if they are randomly chosen. Varying responses can prevent a conversation from feeling repetitive and can also provide alternate ways of stating something if it wasn't clear the first time.

Click the **Workspace** background on the left to deselect the node.

29. In order to better manage our *dialog*, we will create a *folder* for each intent. Folders are a way to keep related segments of dialog in collapsible containers that keep the build area from getting cluttered.

    Click the **Add folder** square.

    Name the folder **Hours of operation**.

    Click **Customize** on the right.

30. Folders can be customized to handle *digressions*. Expand the option to see the configuration.

    Digressions enable the Assistant to break away from a point of conversation, and optionally return when the digression is complete. In our scenario, it is likely that a customer would ask the hours of operation in order to complete another transaction, so we will enable the option to **Return after digression**.

    Click **Apply**.

31. For the conversation to flow into this folder, we want to activate if the #store_hours intent is detected.

    Go to the second section, *If assistant recognizes*, and click the **Enter condition** field. If bot recognizes intent (scroll down until you see) #store_hours and click that.

Now it's configured that if the bot recognizes the #store_hours intent, the conversation will flow into this folder.

32. Now let's create a folder for the store locations. For this folder, we will leave the digression settings to default, and activate when the bot recognizes #store_location.

33. Dialog flow starts at the top of the tree when a conversation begins. The conversation flow will evaluate each node in order until it reaches a true condition. Because the *Welcome* node is always true at the start of a conversation, its contents will be displayed first.

    The dialog will flow in one of three ways. First, if there are child nodes of the current node, then the flow will process those in turn. Second, if there are no more child nodes, or if the next sibling node evaluates to false, then dialog will continue to flow downward to evaluate the next sibling node until one is true. Finally, the flow can branch to a node in a different part of the conversation if the current node tells Dialog to go to it.

34. It's time to start adding dialog. Click the **Hours of operation** folder.

    The dialog will enter this folder if the #store_hours intent is detected. In order to talk about this topic, we need to add a child node. Click **Add child node**.

    The new node is automatically selected. Name this node **"Store hour response"**. For this example, all our stores have the same hours. There are no other conditions we want to check for, so set the condition to **true**. For the response, enter something that makes sense for your store. Press enter when you are done. I'll use "**Both of our stores are open Monday through Friday from 8am to 5pm**." As a variation, I'll enter, "**We are open weekdays from 8:00am to 5:00pm**."

    Open the **Try it** panel and ask for the store hours. Ask again to see the alternate response. Close the **Try it** panel.

35. Let's add some responses for the store locations. Add a child node to the *Store locations* node and call it **Elm location**. For the condition, check for an entity called Street > is > Elm (or write **@street:Elm**). The response should be directions for your Elm street store. I'll use, "**Our Elm Street store is in the Elm Grove shopping center beside Jaynemart**."

    We need to add a sibling node to the Elm location, so click Add node. We'll call this one **Maple location**. Add a condition for the entity @street:Maple, and a matching response. I'll use, "**Our Maple Street store is located at 123 Maple Street, across from the water tower.**"

We need to add one more sibling node in case the user didn't mention which store they are looking for. This time please click the menu option for the Maple location and add a node below.

Call this node **Which location?**, make the condition **true**, and respond with, "**We have stores on Maple and Elm streets. Which one are you interested in?**"

36. Now click the **Try it** panel.

    Click **Clear** to start over.

    Start by asking, "**Where is your maple street store?**"

    You can see that both the intent and entity were correctly identified, and the response is correct.

    Just like in real life where users make typos, so your testing should include some.

    Try asking, "**Where is em road store?**"

    The Assistant knows that the user is asking about a store location but it doesn't recognize the location. So, it is now asking for clarification. Type "**elm**".

    It looks like the entity was correctly identified, but there appears to be a problem with our dialog. The issue we have here is that the dialog will continue from where we left off instead of going back to evaluate the new entity. To fix this problem, we need to change the dialog flow. Click the "**The assistant should**" option and change it to jump to the Elm location after getting the user's input.

37. Click Clear and try again.
    Ask it "**where is the em road store?"**
    When it asks for clarification, enter "**elm**"
    Now we are getting the response we expect. When troubleshooting your dialog, start with the node order, check the conditions, and then look at the "The assistant should" section.

38. Click the **folder icon** to collapse your nodes.

39. Our Assistant currently has two main features, so if a customer asks for help, let's make sure they know what the bot is for. Create a new node below Store locations and call it **Help**. Give it a condition of *#General_Agent_Capabilities*. Respond with, "**I am a Watson Assistant that can tell you our store locations and hours.**"

    You'll notice that we didn't put this node into its own folder, but we could have. Since it is a single node with no branching dialog, we can leave it like this with the option of creating a folder later if the dialog grows.

40. Finally, add one more node that lets the user end the conversation. Create a node called **Goodbye** with a condition of *#General_Ending*. Give it a proper response. I'll say, "**Check back later, as I'm training for new capabilities. Goodbye!**"

41. Clear the **Try it** panel and test your *Help* and *Goodbye* nodes.

42. Notice that the *Goodbye* node does not end the conversation. That functionality is going to be part of your UI, as you may need to also do things like log the user out of multiple back end systems and restart the UI for the next user.

43. Another important feature of Watson Assistant is the ability to have it collect multiple pieces of required information before continuing past a certain point of the dialog. We call this feature *Slots*. One common use for Slots would be to collect all the information you need to process an order.

Part 2: Processing delivery orders

As documented for our project, we met our stated objectives of being able to have the Assistant answer questions about the location and hours of our stores. Now we're going to add a bonus capability of collecting a user's address and order information so we can make deliveries.

Start by going to the **Content Catalog** and adding the *eCommerce* intents to your skill space.

In addition to the new intents, we're also going to need to use some *system entities*. Click **Entities** on the left, then click the **system entities**.

Let's **turn on** the system entities for the location, the date and the number.

Now we have the intents and entities we need for an order.

44. Next go to the **Dialog** link.

45. Now create a *folder* called **Placing an order**. Place it under the Store locations folder.

    Set this new folder to activate if the bot recognizes an intent that corresponds to the new ecommerce value for placing an order.

    Now add a child node so that we can begin taking the order.

    Call this node "**Gather order info**" and click **customize**.

    Turn on Slots for this node so we can collect all the information we need from this one node. If you wanted to prompt the user to provide you all the information at

once, you can select the option, "**Prompt for everything**". From this customization panel, there is also a feature allowing you to handle multiple inputs and outputs from a single node instead of using branching dialog. We'll leave that off.

Click the **Digressions** tab.

You should remember that when we created the dialog for the *store hours*, we turned on digressions so that we could branch to that node and then return back to our conversation. In order to use that functionality, we need to allow this node to branch away. Expand the first option and **allow digressions to go away from this node**.

If someone is placing an order, we don't want random digressions in the conversation to stop the order process, so also enable the option to **allow only digressions that return.**

Click **Apply**.

46. The first thing you should notice is that after the bot checks for a condition, it also starts checking for slot items. If it finds the entity, it will store that value as a context variable for later reference. If it doesn't find it, you will provide a prompt to ask the user for that information. The Type field is used for specifying if the slot is mandatory or optional.

    Any time you have a slot that has an empty field for "*If not present, ask*", the slot is optional. Filling that field makes it mandatory.

    We always want this node to activate, so set the condition to **true**.

47. For our ordering process, we'll need an address, a quantity and a delivery date. We're going to accept local deliveries only, so we'll use our existing *@street* entity to detect the address. Any valid delivery street for this demo needs to be part of an existing entity in order to be detected. If you are considering a national delivery system, instead of entering each street name into an entity field, you would instead use a custom entity with regular expressions to detect the address. Feel free to try that on your own after the exercises, but it's beyond the scope of this course demo.

48. When filling in slots based on user utterances, the values are obtained according to the order in which they are received. Because of this requirement, you will want to make sure the instructions to the user are clear and that you thoroughly test your dialog before putting it into production. The only time it could cause confusion, is when you are collecting multiple instances of the same type of entity, such as a *@sys-number* for the house number and another for the quantity.

49. Fill 4 slots in the "*Then check for*" section. Enter the following for individual slot and press **Add slot** as needed:
    1) Enter an entity of *@street*. If not present, ask, "**What is your address?**".

2) Enter an entity of *@sys-number*. If not present, ask, "**What is your house number?**"
3) Enter an entity of *@sys-number*. Save it as **$quantity**. If not present, ask, "**How many do you want?**"
4) Enter an entity of *@sys-date*. If not present, ask, "**When do you want this delivered?**"

50. For the *Assistant responds* section, enter:
"**Thanks for placing your order. $quantity items will be delivered to $number $location on $date.**"

51. Now let's **Try it**.

52. At the prompt, enter, "**I want to place an order**."

    For the address, enter **123 Elm Street** (only Elm and Maple streets are known).

    For the quantity, enter **2**.

    When asked about the *delivery date*, ask "**what are your hours?**"
    Notice that the dialog digresses to answer the question and then asks again when to deliver.

    Tell it you want delivery on **Monday**.

    The dialog will confirm the information you entered.

53. At the top of the screen (in the *Try it out* panel), click "**Manage Context**".

    Every time a variable is set in dialog, you can use it in conditions and responses in your dialog nodes. Variables can also be passed back from the service to the calling application so that they can be acted on externally.

54. When your Assistant is ready to be deployed to a test or production environment, you can use the **Integrations** section, accessible thru the **Assistants** icon, to assist with that task. Click on **Add integration** to start with the deployment configuration.

    The purpose of this deployment is to provide bidirectional communication between your messenger platform or application with the *Watson Assistant* service so that the solution can be accessed from your platforms of choice. Setting up the communication usually requires administrative access from the application side, as well as authentication credentials from the *Watson Assistant* side. The deploy options in the service interface provide you with step-by-step instructions to configure both sides to communicate with each other.

    A few tips for deploying include:

1. Before deploying, make sure you have an IBM Cloud service slot available for the connector. If you don't have a free slot, you will get an error that points you to a troubleshooting page. The free IBM Cloud accounts are limited to 4 services, so plan accordingly.
2. Make sure you have administrator access to the environment you are deploying to. Without that access, you will not have authority to make the necessary configurations.
3. Follow the instructions precisely and in order. If you encounter an error, fix it before moving to the next step.
4. Keep a record of all your credentials and configured URLs for reference later if you decide to change the configuration.
5. Configure your bot with the lowest necessary permissions required to perform its job. Don't give it elevated privileges because it does not require them.

55. After you have deployed your Assistant, the service will begin collecting data on its usage, and you can inspect that data on the **Analytics** page. The data does not reflect the *Try it* panel, only your deployed instance.

56. The **Overview** tab gives you a view of how active your Assistant is by the number of conversations and average messages per conversation. It also shows performance by highlighting any instances of weak intent understanding.

57. Scrolling down shows more graphs, as well as top intents and entities.

58. Selecting the **User conversations** tab gives you the opportunity to look at individual conversations and user utterances.

59. You can filter the conversations for specific intents, entities, or user statements. So, I can use the page to look for the word "**where**" and see that there is an instance of a user statement that didn't resolve the way I wanted. I can take that information and make corrections to my intent user examples. These tools are helpful for identifying candidates for new intents, entities, and topics for your Assistant to address in the future, as well as quickly seeing which topics your users are interested in.

Further Resources

Why create a Chatbot with IBM Cloud? – Video (4 mins)

Wie erstellen Sie einen Chatbot? – Video in German (4 mins)

Was ist ein Chatbot? – Information in German