Develop and Deploy AI-Powered Intelligent Bots

Create a Bot with Intents, Entities, Flows, Components and Channels

In this hands on workshop, you’ll create a simple financial Bot, configure its artifacts, test it and deploy it to a sample web application.

You will create *intents* (you can think of an intent as the meaning behind what the user wants to do), add *utterances* (sample phrases to help your bot reference intents when it parses the user input), add *entities* (extra content to enable your bot to complete a user request), configure system components (built-in components that allow you to perform typical interactions with the users, control the flow, perform language detection and manipulate variables) and custom components (REST services that allow the Bot to interface with external APIs) and finally configure messaging channels where you enable a messaging platform to use your bot.

The Oracle Intelligent Bots platform has many other capabilities but we won’t be able to explore all of them during this lab.

What Do You Need?

For this lab, you’ll need the following files from the *labfiles.zip* provided:

- FirstBotYAML.txt
- FirstBot-Intents.csv
- CustomComponentURL.txt
- CustomPrintBalance.txt
- CustomStartPayment.txt
- CustomTrackSpending.txt
Lab 1: Create a Simple Banking Bot

In this section, you create a simple banking Bot and examine the main artifact types.

1. Log in to your instance of Oracle Intelligent Bots and then click **New Bot**.

2. In the Create Bot dialog, enter *FirstBot_XX*, where *XX* are your initials. Next, add a description and then click **Create**.

3. In the left navbar, you can see a list of icons that you use to navigate to your intents, entities, flows, resource bundles, components, settings and quality.
Develop and Deploy AI-Powered Intelligent Bots

reports. The left navbar is what you are going to use to navigate through the Bots UI during the labs.

4. By default, the Intents page is open, but as of this moment, you don’t have any intents.

5. Click Entities (the second icon down) and notice that it’s pre-populated system entities. These are standard entities that you can use in your Bot without having to explicitly define them.
6. Next, click the Flows icon. Notice that it’s pre-populated with code that enables the Bot to output a “hello” message. Don’t worry about the code for the flow right now -- you'll make modifications to it later.
7. The next option on the list is the Resource Bundles. Resource Bundles are used to localize your Bot based on the language set for the messaging channel currently in use.

8. Now, click the Components icon. Components is the area where you configure external REST services for your Bot to interact with.

9. Finally, click the Settings icon. Notice its tabs: General and Channels.

10. The General tab contains general details about the Bot and some properties that influence how the Bot is trained. You’ll find out more about that later. The Channels tab is where you’ll publicize your bot by hooking it up to the Web Messenger. That too is something that you’ll do later in the lab.
In the next section, you will add artifacts to make the Bot work.
Lab 2: Adding Intents and Entities to Your Bot

In this lab, you will create intents and add utterances to them. Then, you will create and add entities to the intents.

1. Click the Intent icon in the left navbar and then click the green Add Intent button.

2. This intent will be used to find out your banking balance, so replace Intent1 in the Name field with Balances and then provide a description (i.e. Query my account balance). These values are saved automatically, so you do not need to explicitly save them. As you create artifacts, you may notice a message in the lower right corner that tells you that your work has been saved.
3. Now that you have an intent, you need some example phrases, or utterances, that express what a checking balance means. In the Examples area add the following text: *How much money do I have in my checking account?* and then press Return.

4. Add the following list of utterances to your intent, each followed by a return. Keep in mind that the examples don’t need to be in the form of a question; they can be a statement.
Develop and Deploy AI-Powered Intelligent Bots

- How much do I owe on all of my credit cards?
- How much money did I save last year?
- How much money do I have in all of my accounts?
- What's my savings balance?
- What's my available credit on my Visa?
- What's my balance?
- What's the current balance on my cc?

5. While you can manually add the intents like you did in the previous step, you can also add intents quickly by importing them from a CSV file. Here is how to do it.

In the middle of the Intents page, click More and then Import Intents button. Select the FirstBot-Intents.csv file found in the labfiles directory.

6. Next, click Open.

7. Three intents should be imported or updated: Balances, Send Money, and Track Spending. Each intent has its own set of utterances.
Develop and Deploy AI-Powered Intelligent Bots

8. To get a better idea of the how the language used in these utterances differentiates each of the intents, click each intent (Balances, Send Money and Track Spending) and take a look at their respective example phrases.

9. Now that you’ve created all the intents for your Bot, you can add Entities to it. Entities are special pieces of information that help the Bot break the user’s sentence apart and extract the relevant parts of it.

For example, if you want to request the balance of an account, you would probably need to know the kind of account that returns that balance (i.e. checking, savings, etc). To extract that information, you would use an entity that defines different types of accounts.

10. Click the Entities icon in the left navbar. Click the green Add Entity button, replace Entity1 in the Name field with AccountType and then add a short description (i.e. The type of bank account to be used).
11. Now that you have an entity, you need to provide some values that it will use to identify key words from the user input. In the case of the account type, you need to add values that represent the various types of accounts that you could query for a balance.

12. In the Configuration area, be sure that the Type property is set to Value List and then click the green Add Value button.

13. In the popup dialog, enter checking as the value and check as a synonym. Press return/Enter. Make sure you use lowercase values here.

14. Click Create.
15. Next, add a second value called `savings` and then add `save` as the synonym. Don’t forget to press return/Enter.

16. Finally, add a third value named `credit card`. Enter `AMEX`, `Visa`, and `card` as its synonyms. When you’ve finished, your entity definition should look like the image below.
17. Now, using the tables below, add a couple more custom entities.

   a. The first one, **ToAccount**, is for the recipients of money transfers.

<table>
<thead>
<tr>
<th><strong>Entity Name</strong></th>
<th><strong>Values</strong></th>
<th><strong>Synonyms</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>ToAccount</td>
<td>Lauren, Shea, Mom</td>
<td>sister, daughter, mother</td>
</tr>
<tr>
<td></td>
<td>Chase Preferred, the baby sitter</td>
<td>Chase, babysitter</td>
</tr>
</tbody>
</table>
b. The second custom entity, **TrackSpendingCategory**, defines the categories used to track spending. The values on this entity have no synonyms.

<table>
<thead>
<tr>
<th>Entity Name</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrackSpendingCategory</td>
<td>gas, retail, travel, uber, restaurants, coffee, grocery</td>
</tr>
</tbody>
</table>
18. Now that you've got all required Intents and the Entities they will work with, you need to associate them. Don’t worry – it’s easy!

a. Click the Intents icon in the left navbar. Select the **Balances** intent. Locate the Intent Entities area at the right of the page.

b. Click the green **Add Entity** button and then select **AccountType** from the list.
c. Now, select the **Send Money** intent. Use the green **Add Entity** button to select the entity and associate the **Send Money** intent with its entities according to the table below.

<table>
<thead>
<tr>
<th>Intent</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send Money</td>
<td>AccountType</td>
</tr>
<tr>
<td></td>
<td>CURRENCY (system entity)</td>
</tr>
<tr>
<td></td>
<td>ToAccount</td>
</tr>
</tbody>
</table>
d. Finally, associate the Track Spending intent with its entities according to the table below.

<table>
<thead>
<tr>
<th>Intent</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrackSpending</td>
<td>DATE (system entity)</td>
</tr>
<tr>
<td></td>
<td>DURATION (system entity)</td>
</tr>
<tr>
<td></td>
<td>TrackSpendingCategory</td>
</tr>
</tbody>
</table>

In the next section, you will customize the Bot dialog flow code.
Lab 3: Customizing the Bot Dialog Flow

In this section, you will customize the Bot code, called BotML, and make it ready to respond to user’s requests.

1. Click in the **Flows** icon on the left navbar.

2. There’s BotML code in the editor that displays “hello” when you run the Bot. However, we’re not going to use this code. Instead we’re going to add our own flow.

3. From the **labfiles** directory in your system, open the **FirstBotYAML.txt** file in your text editor of choice.

4. Take a look at the BotML code. Under the **context** node near the top, notice that this flow definition names the AccountType entity as a variable (**accountType**) and further down in the intent state, names your intent (Balances) as one of the actions. Because flow definition includes the **accountType** variable in the **startBalances**
state, the conversation flow proceeds to `askBalancesAccountType` and then finally onto the `printBalance` state, which displays the balance. When the `accountType` variable is not set, then the `askBalancesAccountType` state will prompt you for the account type using the value list values that belong to the `AccountType` entity. It then moves to the `printBalance` state.

5. Copy the contents from the `FirstBotYAML.txt` file into the Bot Flow editor, replacing all of the code. To prevent YAML formatting issues, paste any code from the beginning on Column 1.
Finally, click the **Validate** button in the upper right. You should see a message that there were no problems found in your Bot. Just in case your BotML doesn’t get validated, click on the bug icon in the top right corner of the screen. The line number and the error message and the will be displayed to you in the lower panel.

6. Next, let’s train and test the Bot with what you have done so far.
Develop and Deploy AI-Powered Intelligent Bots

**Lab 4: Train and Test Your Bot**

In this lab, you will use the training tool on the Bot. Training your Bot enables it to understand phrases other than the utterances that you’ve defined for its intents. In other words, training allows your Bot to understand user input that’s similar to the utterances, but not exactly the same.

1. In the upper right, click the **Train** button. This will kick off a process that will run an algorithm that takes your example utterances and builds the model that will be used to ascertain the intents and entities. Whenever the Bots platform recognizes that your Bot needs to be trained or re-trained, it will display an exclamation point in the train button. The training process may take a few minutes, so be patient. Once the training is complete, the exclamation point is grayed out.

7. To test the Bot, click the **Play** icon in the upper right of the page. This opens the Tester where you can see two tabs: Bot and Intent.

8. Click the Bot tab in the Tester to test the Bot. Remember that what you type into the Message area is what gets sent to the Bot when you click the **Send** button.
9. Let's start out simple to test the Bot:

a. In the Message area, type in *What’s my balance?* and then click the **Send** button.
b. Since the account type wasn't specified, the Bot presents you with three options: checking, savings, credit card.

c. Click one of the three options. The Bot outputs text showing the chosen account and its balance.
10. Now let's try a message that includes the account type. In the Tester, click the **Intent** tab.

11. In the message area, type in *What is the balance on my Amex?* and then click **Send**.

The Tester displays the level of confidence, expressed as a percentage, that the intent can resolve the user input. You also see that the Account Type entity is recognized as a credit card.

**Important:** To avoid the confusion that can arise from an incomplete flow from a previous session by completing each round of requests and responses, or start a new session by clicking the **Reset** button.

In the previous labs, you've created a Bot, added an intent, an entity (and its values) and validated the BotML code. You also trained your Bot and tested it. It’s time for you to test the intent resolution.
Lab 5: Test the Intent Resolution

In this lab, you will test the intents that you’ve just created and adjust the intent resolution of your Bot.

1. Click the Play icon to open the Tester (if it’s not open yet). Click the Intent tab in the Tester. Click Reset if it’s already open.

2. Then in the Message area, enter *What’s the balance in my checking account?* and then click Send.

   ![Tester Interface](image)

   The Tester displays a list of all of the intents that you’ve added, each with a confidence percentage.

3. Notice how the Balances intent is first in the list of intents because the message that you just sent is specifically about balances.
4. Now, click JSON (located above the message area) to see what has been returned by the algorithms. Using the slider bar to scroll down, you can see the account type and intent matches.

5. Click the Reset button.
Develop and Deploy AI-Powered Intelligent Bots

6. Now let's try a different message: What’s on the credit card.

7. Now at this point, you may actually encounter an issue where the A.I. engine identifies an intent that you didn’t expect as the more likely candidate to resolve the input. For example, in the following image, you can see that the Balances intent is rated higher than the Track Spending intent for the input, What’s on the credit card.

8. When this happens, you can increase the confidence level and the intent accuracy by first selecting the radio button by the correct intent and then by clicking the Add Example button. Doing this adds the text from the Message area as an utterance for the selected intent (Notice you might see different results in the confidence level while testing the intents).

   Be sure that the radio button by the Track Spending intent is selected and then click the Add Example button.
9. Next, train your Bot again with this new example phrase.

10. Click **Reset** and then enter the same statement (*What's on the credit card*) again. Click **Send**.

   The Track Spending intent should now be at the top of list because you added the new utterance and retrained the Bot. By testing it with additional values, you can increase the pool of example utterances that your intent uses, making it more accurate.
Develop and Deploy AI-Powered Intelligent Bots
Lab 6: Connect Your Bot to a Backend System

In this lab, you will configure a Bot component/service to use custom components that connect your Bot to a backend system that handles retrieving the user’s account balance, transferring money to someone in the user’s list and tracking user’s spending.

So far, you’ve been using the built-in system components (the ones that begin with System.*). In Oracle Intelligent Bots, you provide the business logic through custom components. Custom components are REST services that Bot developers create and deploy onto any infrastructure that can expose the components on the Internet. Once the components are available, Bot developers can then configure their Bots to interact with them.

We’ve provided a set of custom components that implements the business logic for your Bot. This lab will walk you through the steps of configuring your Bot so it can access them. You will then modify your Bot's dialog flow to invoke these custom components.

1. In this step, you'll add the service that will let your Bot connect to a backend APIs. When you're done, your Bot will then include the definitions for the custom component state.
   To get started select the Components icon in the left navbar.

2. Next, click Add Service button.
You should see the Create Service screen which you are going to fill up in the next step.

3. Configure the component service using the values listed in the `CustomComponentsURL.txt` provided in the `labfiles` directory. You should end up with something similar to the following.
Click Create.

4. Your brand new component service should be created successfully and its details are now displayed in the Components page.
5. To see the various components that this service provides for your Bot, click the arrow next to the service name.

6. You are ready to add those custom components to your BotML code. Go back to the BotML code by selecting the Flows icon in the left navbar.

7. Using the CustomPrintBalance.txt file provided in the labfiles directory make the following two changes to the printBalance state to update it from using a system component to the custom component, BalanceRetrieval.
   a. First, change the component type from System.Output to BalanceRetrieval.
   b. Now that you are using the custom component to call the API, you need to pass along the accountType. Remove the text property and replace it with accountType: "${accountType.value}".

The printBalance state should look like the following:
8. Click the **Validate** button to make sure everything is correct.

9. Next, you will work with the **Send Money** intent. This one is a little more complex because it requires new states to be added to the Bot flow. All of the new states being added are System components except for the **doPayment** state which uses the **Payments** custom component that you have added above.

10. Go to the top of the BotML code and add two new variables under the **context → variables** section as follows:

    ```
    toAccount: "ToAccount"
    paymentAmount: "CURRENCY"
    ```

    Your **variables** section of the BotML code should look like this:

    ```
    metadata:
        platformVersion: "1.0"
        main: true
        name: "FirstBot_MJ"
        context:
            variables:
                accountType: "AccountType"
                toAccount: "ToAccount"
                paymentAmount: "CURRENCY"
                iResult: "nlpresult"
    ```

11. Go to the **intent** state and add the following new action under the state’s transitions

    ```
    Send Money: "startPayments"
    ```

    Your **intent** state should look like this:
12. Copy the content of the CustomStartPayment.txt file, go back to your Bot's Flow and paste the code right after the printBalance state and before the unresolved state. Your BotML code should look like the following:

```
states:
  intent:
    component: "System.Intent"
    properties:
      variable: "iResult"
      confidenceThreshold: 0.4
    transitions:
    actions:
      Balances: "startBalances"
      Send Money: "startPayments"
      unresolvedIntent: "unresolved"

printBalance:
  component: "BalanceRetrieval"
  properties:
    accountType: "$(accountType.value)"
  transitions:
  return: "printBalance"

startPayments:
  component: "System.SetVariable"
  properties:
    variable: "accountType"
    value: "$(iResult.value.entityMatches['AccountType'][0])"
  transitions: ()
resolveToAccount:
  component: "System.SetVariable"
  properties:
    variable: "toAccount"
    value: "$(iResult.value.entityMatches['ToAccount'][0])"
  transitions: ()
askFromAccountType:
  component: "System.List"
  properties:
    options: "$(accountType.type.enumValues)"
    prompt: "From which account do you want to make a payment?"
    variable: "accountType"
  transitions: ()
askToAccount:
  component: "System.List"
  properties:
    options: "$(toAccount.type.enumValues)"
    prompt: "To which account do you want to make a payment?"
    variable: "toAccount"
  transitions: ()
askPaymentAmount:
  component: "System.Text"
  properties:
    prompt: "What's the payment amount?"
    variable: "paymentAmount"
  transitions: ()
doPayment:
  component: "Payments"
  properties:
    fromAccountType: "$(accountType.value)"
    toAccount: "$(toAccount.value)"
    amount: "$(paymentAmount.value.totalCurrency)"
  transitions:
  return: "doPayment"

unresolved:
  component: "System.Output"
  properties:
    text: "Unable to resolve intent!"
  transitions:
  return: "unresolved"
```
Develop and Deploy AI-Powered Intelligent Bots

13. Click the Validate button to make sure everything is correct.

14. Finally, let’s look at how you are going to update the Track Spending intent. Go to the top of the BotML code and add a new variable under context → variables as follows:

   spendingCategory: "TrackSpendingCategory"

Your variables section of the BotML code should look like this:

   metadata:
     platformVersion: "1.0"
   main: true
   name: "FirstBot_MJ"
   context:
     variables:
       accountType: "AccountType"
       toAccount: "ToAccount"
       paymentAmount: "CURRENCY"
       spendingCategory: "TrackSpendingCategory"
       iResult: "nlpresult"

15. Go to the intent state and add the following new action under the state’s transitions

   Track Spending: "startTrackSpending"

Your intent state should look like this:

   states:
     .
     intent:
       component: "System.Intent"
       properties:
         variable: "iResult"
         confidenceThreshold: 0.4
       transitions:
         actions:
           Balances: "startBalances"
           Send Money: "startPayments"
           Track Spending: "startTrackSpending"
           unresolvedIntent: "unresolved"
Develop and Deploy AI-Powered Intelligent Bots

16. Copy the content of the CustomTrackSpending.txt file, go back to your Bot’s Flow and paste the code right after the doPayment state and before the unresolved state. Your BotML code should look like the following:

```botml
16. Copy the content of the CustomTrackSpending.txt file, go back to your Bot’s Flow and paste the code right after the doPayment state and before the unresolved state. Your BotML code should look like the following:

```botml
17. Click the Validate button to make sure there are no errors in the BotML code. Next, you will test your Bot to see how the custom components make a difference in what’s returned.

18. Let’s start with the Balances intent. To test the flow code that uses custom components, click the Play button in the upper right to open the Tester. If it’s already open, click Reset to start a new session.

19. Next, enter What’s my balance? in the Message area and then click Send. You should see a list of all the accounts you’ve included in the System.List component.
20. Next, select an account. The balance appears, which is a value returned by the `BalanceRetrieval` custom component.
Develop and Deploy AI-Powered Intelligent Bots

21. Try out some other messages, including some with the account in the message text to see how the Bot responds (i.e. *What’s my balance in checking?*)

22. Now let’s test the Send Money intent flow. First, click the Reset button. Next, enter *I would like to send money* and click Send.

When prompted, either type the account to send the money from or select it from the list.

23. Next, you’re prompted for a person to send the money to. Select a person to receive the money.
24. Finally, you are prompted for an amount to send. Enter $50 and click **Send**. The Bot will confirm the recipient, the amount and account to transfer the funds from.

25. Let’s now test the flow of the **Track Spending** intent, so click **Reset**. You need to specify a spending category to test this intent. So, let’s start off by entering **How much did I spend on gas?** and then click **Send**.
26. Now click Reset and try another one. For example, How much did I spend on travel? and then click Send.

Congratulations! You have just completed this lab. Next up, you will integrate your Bot with a Web front-end.
Lab 7: Setup and Run Your Client Application

In this lab, you will configure and publish your Bot through a Web channel, download and install the sample client application, run and test it against your own Bot.

Because this allows users to access your Bot through a Web page, it’s within easy reach of a multitude of users.

Before You Begin

To complete this lab, make sure you have Node.js (https://nodejs.org) downloaded and installed on your machine.

To start this lab, configure a new Web channel for your Bot.

1. In the Oracle Intelligent Bots UI, click the Settings icon in the left navbar and then click the Channels tab. Click Add Channel.

2. In the Create Channel dialog, give the channel a name and a short description.

3. Select Web as the Channel Type.

4. Switch on the Channel Enabled toggle and then click Create.
5. In the Channel definition screen, copy the **App Id** to a text file. You'll use it later on this lab while running the sample client application.

7. Extract the zip file and open a terminal / command prompt session in the ChatSample directory.

8. Run the following command:

    `npm install`

9. To start the sample ChatSample app, run the following command:

    `node server.js`

    If your server starts correctly, you should see a message indicating that the server is listening on port 3000.

10. Open a new browser window and navigate to ‘http://localhost:3000’

11. Enter the **App Id** generated when you created a new Web channel.
12. In the next screen, click the ‘Chat with your Bot’ button.

13. The Web Messenger widget should appear in the right lower corner of your web browser window.
14. You can now send messages to your Bot using the chat window. You can ask it about balance information, track spending and money transfer.
Feel free to test the Send Money and Track Spending intents by sending other messages to your Bot (i.e. I’d like to send money or How much did I spend on travel?)

Hooray! You have now introduced your Bot to its public by hooking it up to a web interface. Even more impressive -- you’ve completed the lab!