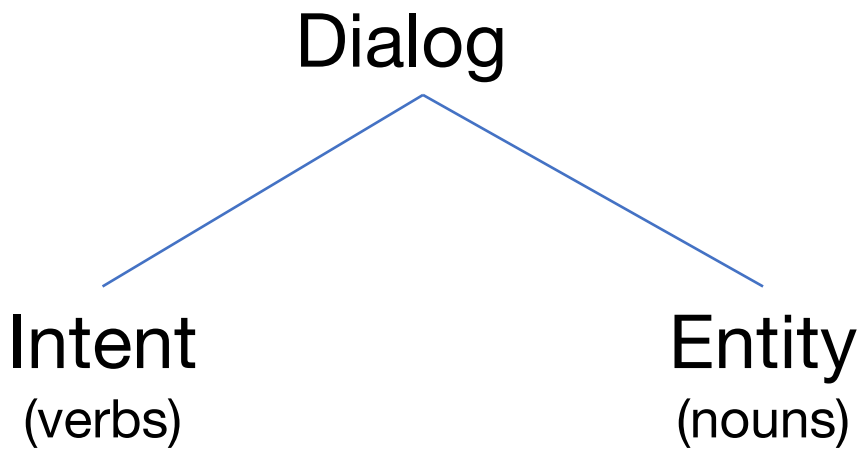
Watson Conversation

Lecture Deck

Components of Training Watson

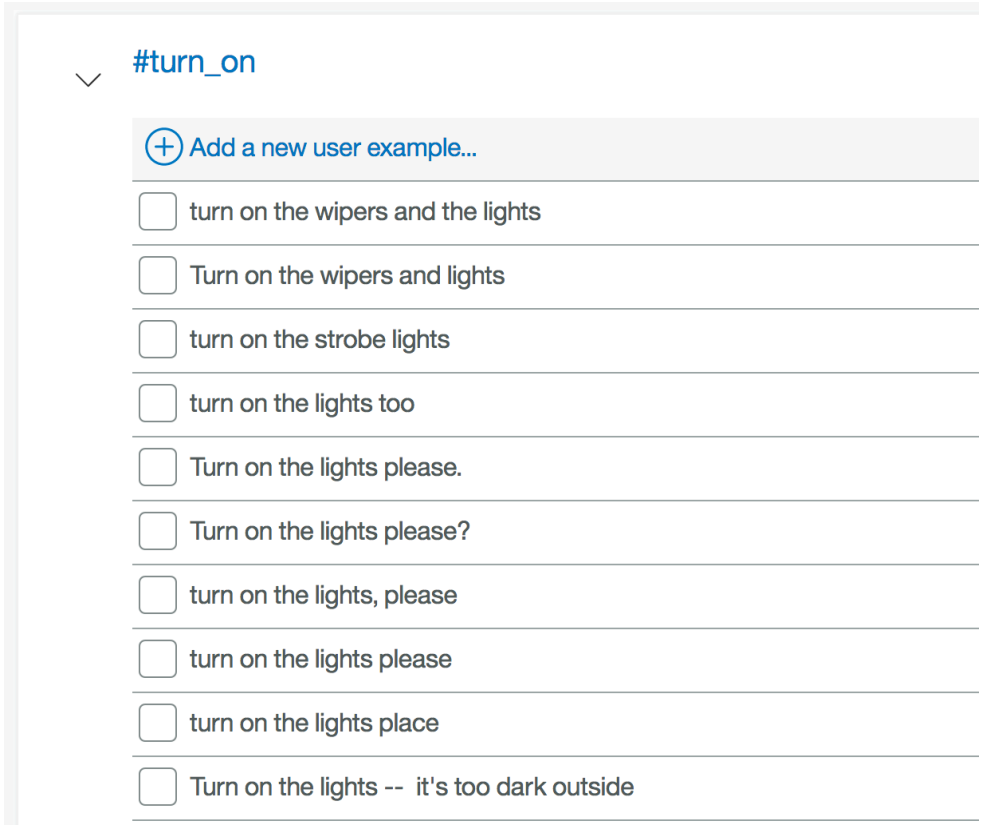
Create a dialog such as a chat bot, customer service agent, smart product, that responds to natural language



Intents

An *intent* is a purpose or goal expressed in a customer's input, such as answering a question or processing a bill payment.

Provide at least five examples for each intent.

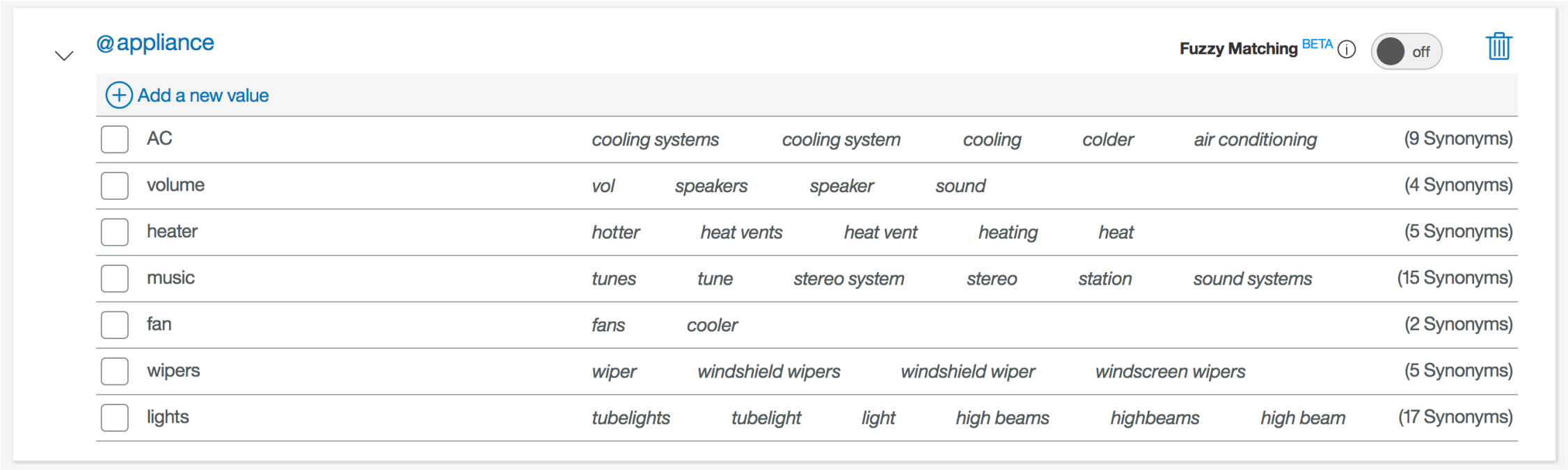


Designing Intents

1. Gather as many actual customer questions, commands, or other utterances as possible. Using input from real users gives a better picture of the expected input than having experts create lists of possible utterances. Remember that customers might phrase the same kind of request in many different ways.
2. After you have a list of examples, sort them into categories based on the capabilities you want your application to support. These categories represent the intents you will define, and the examples will help your application to identify the intents in new input.
3. Continue to refine your intents and examples as needed. Do not think of your set of intents as a finished product. It is likely that when you design your dialogs, you will identify additional intents that you need to add.

@ Entities

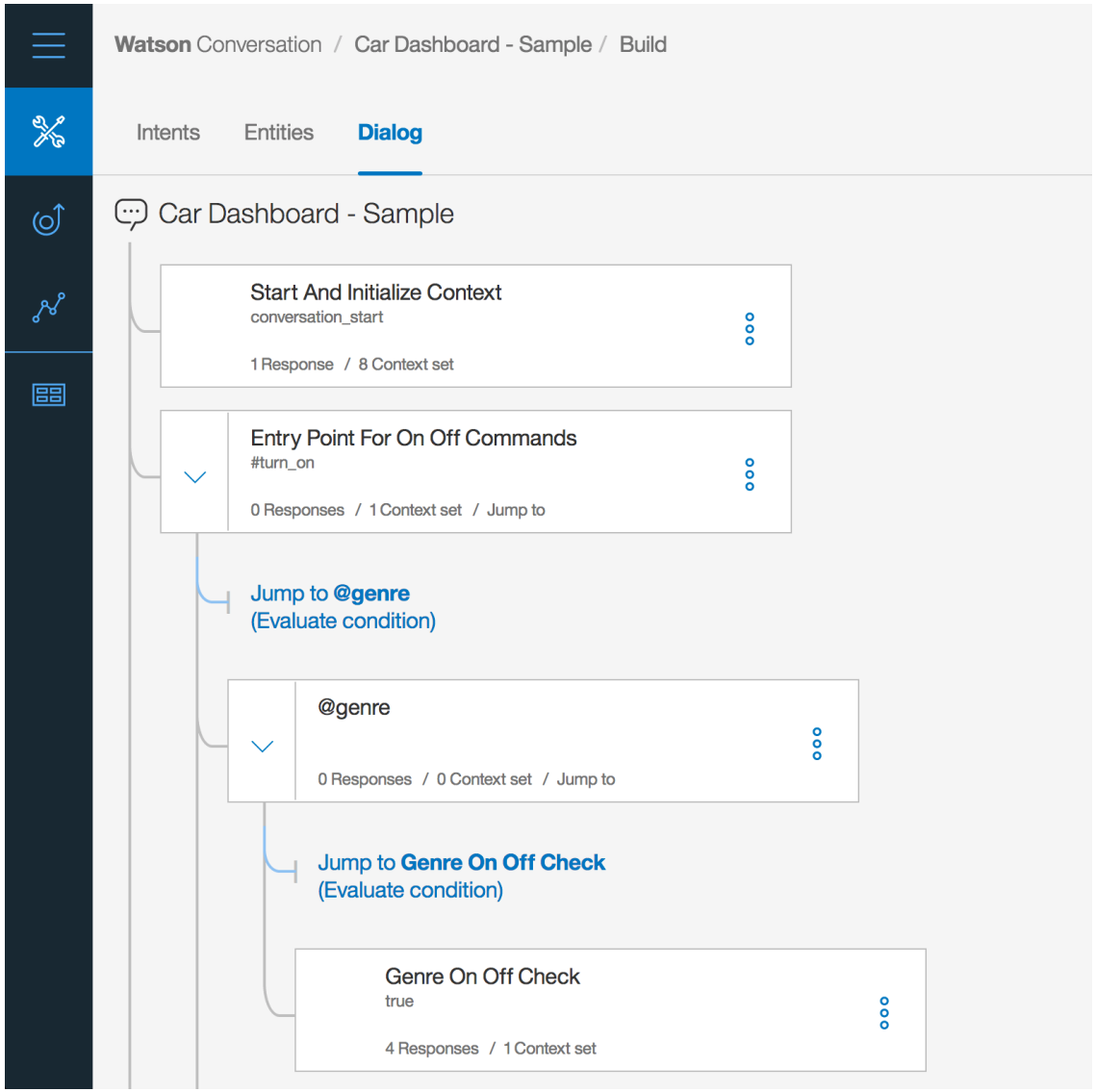
An *entity* represents a class of object or a data type that is relevant to a user's purpose. By recognizing the entities that are mentioned in the user's input, the Conversation service can choose the specific actions to take to fulfill an intent.



Dialog

The dialog component of the Conversation service provides responses to users based on the identified intents and examples.

A dialog uses the intent and entity that have been identified, plus context from the application, to interact with the user and ultimately provide a response.



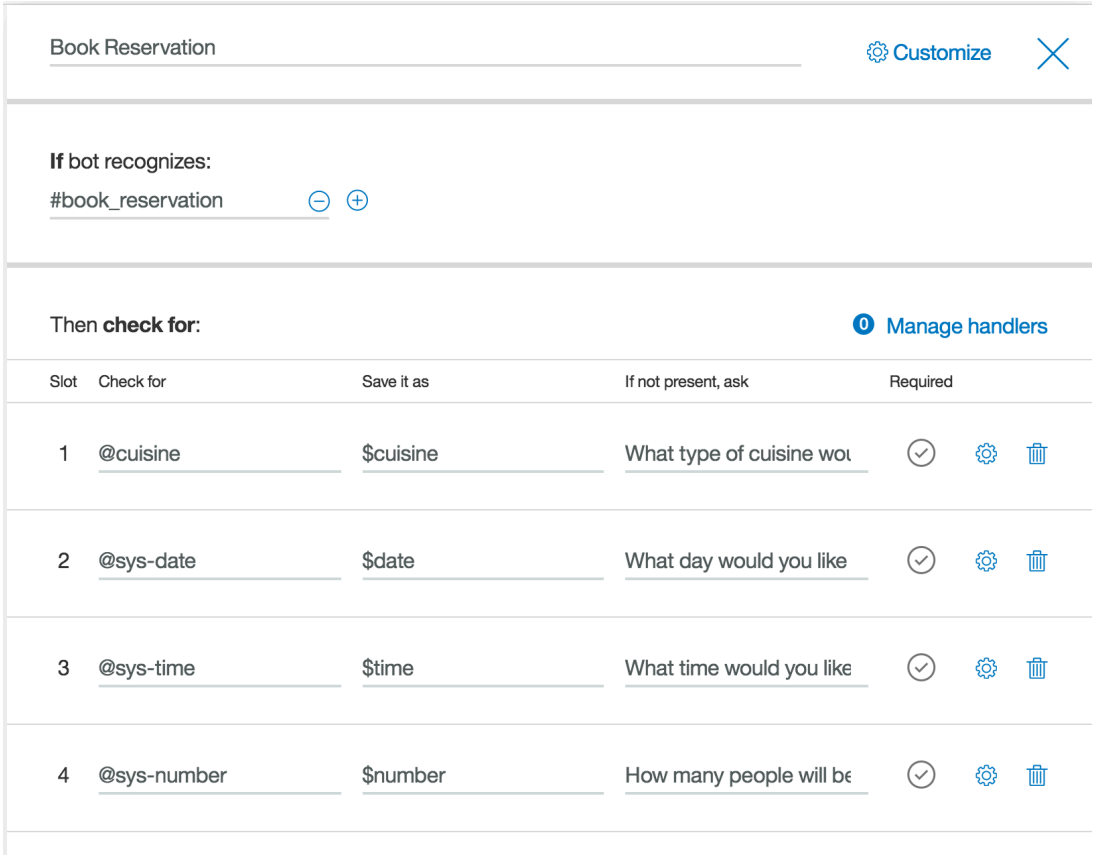
Designing Dialog Conditions

Condition	What it does…
welcome	Evaluated as true during the first dialog turn (when the conversation starts), only if the **initial request from the application does not contain any user input**.
conversation_start	Initial output. Unlike **welcome**, it is true whether or not the **initial request from the application contains user input**.
@cuisine:Chinese	Process node if an entity toppings with value **Chinese** is present
#{intent name}	Process node if intent is present
entities.size() > 1	Test the number of entities input has
true	Catch-all within a **child tree level**
anything_else	Catch-all within **top tree level**. We've got a problem we can't solve.

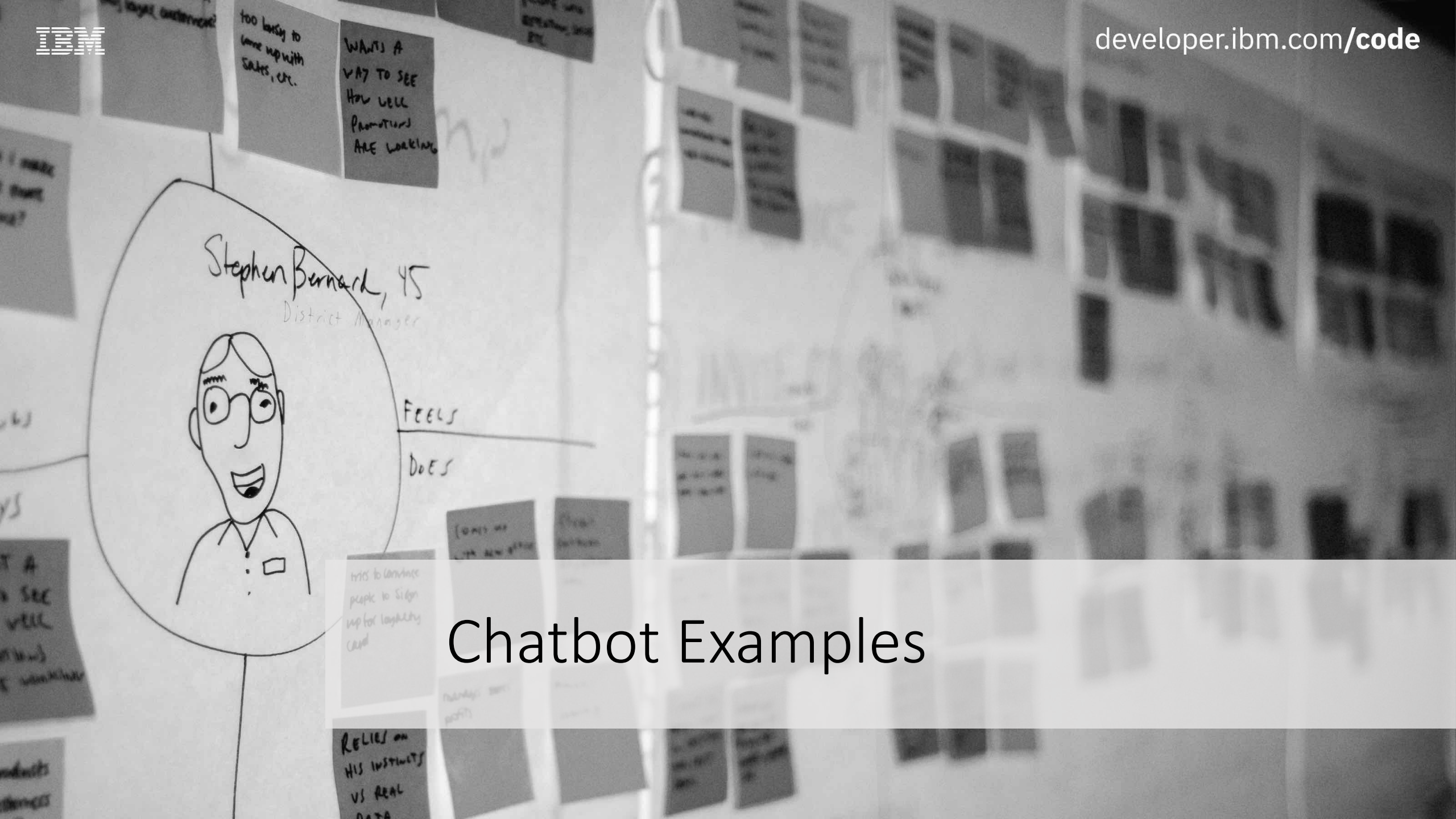
Slots

Add slots to a dialog node to gather multiple pieces of information from a user within that node.

Slots collect information at the users' pace. Details the user provides upfront are saved, and the service asks only for the details they do not.



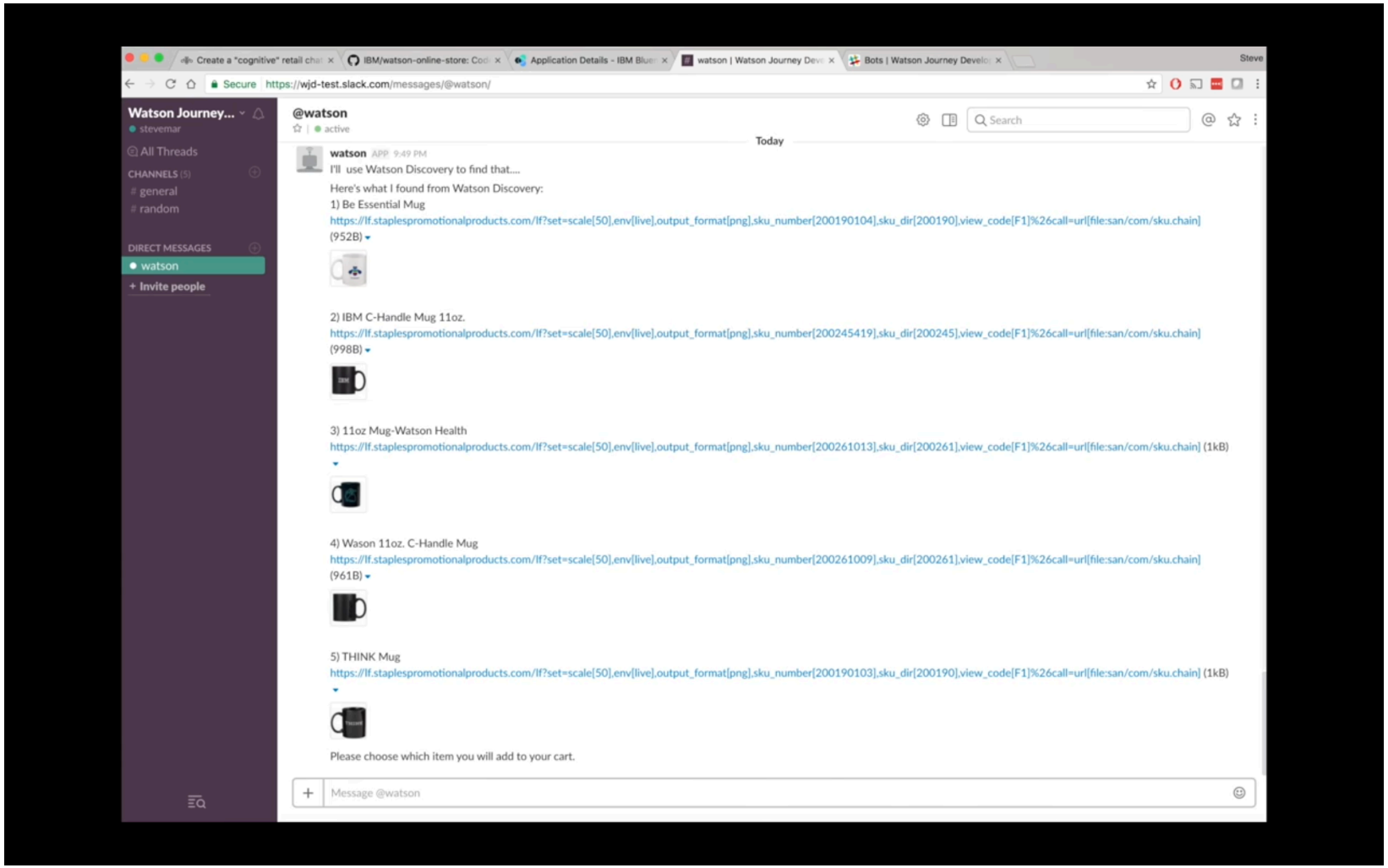
https://github.com/IBM/watson-conversation-slots-intro



Chatbot Examples

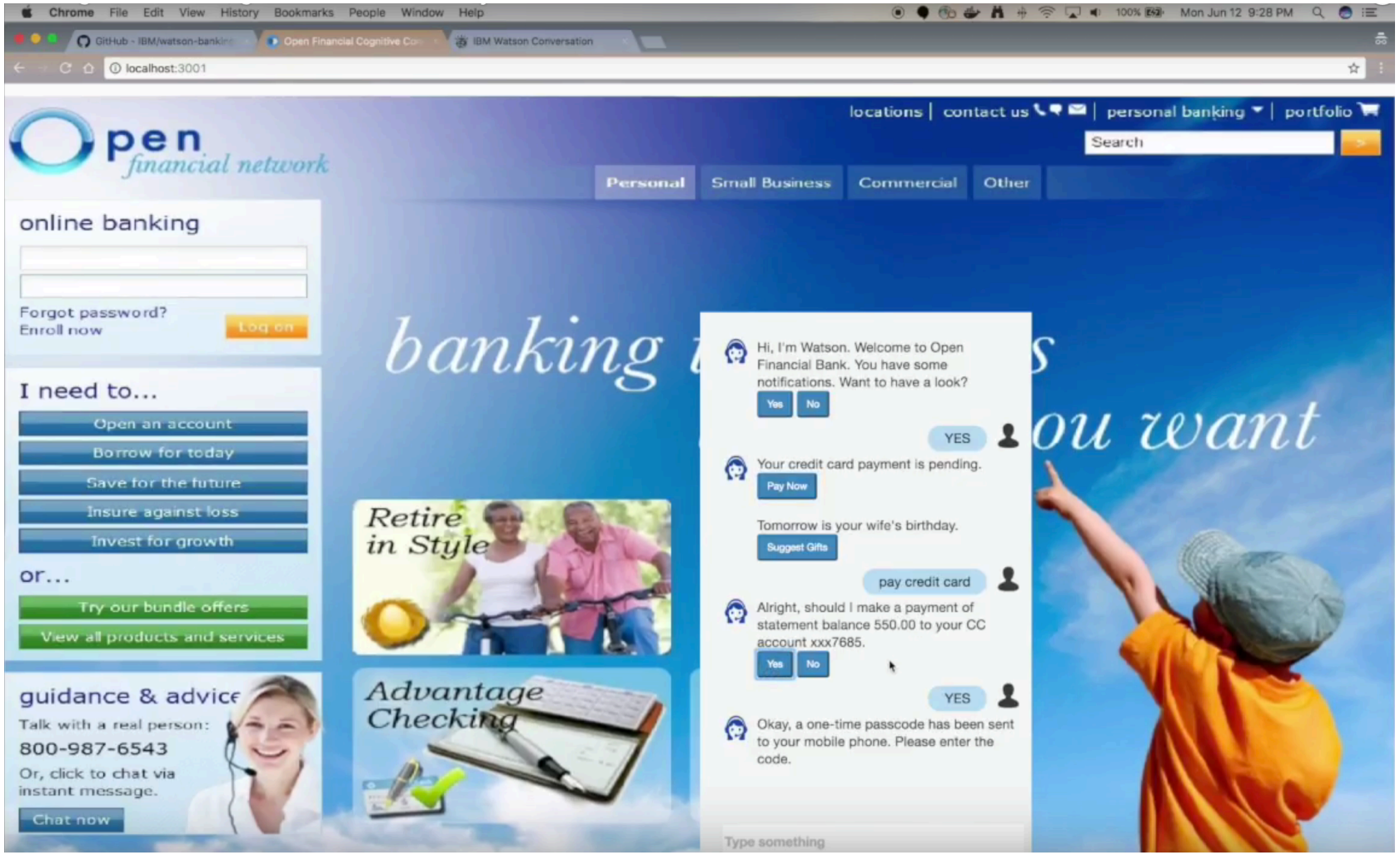
Configurable, retail-ready chatbot

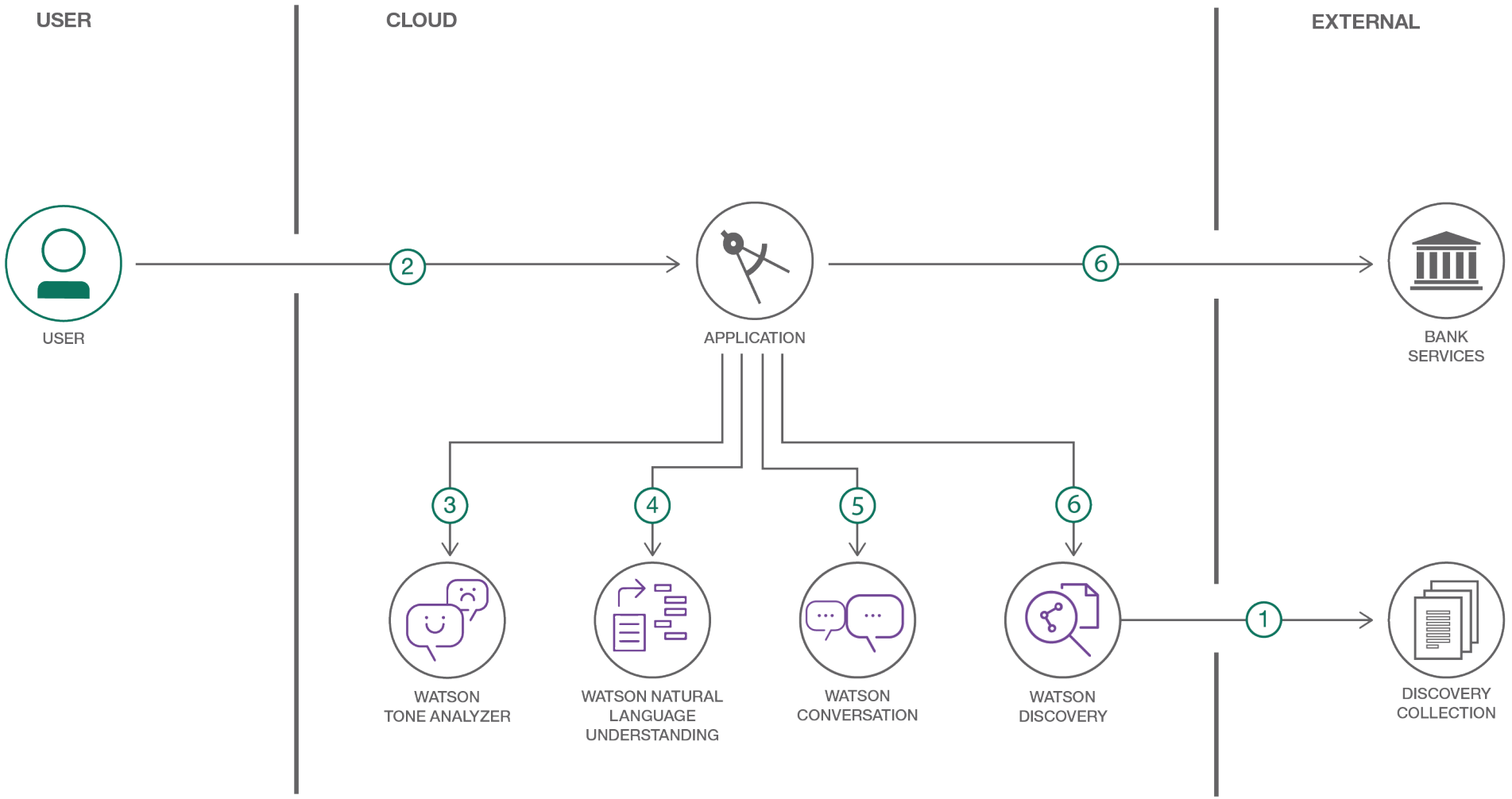
ibm.biz/code/create-cognitive-retail-chatbot



Create a cognitive banking chatbot

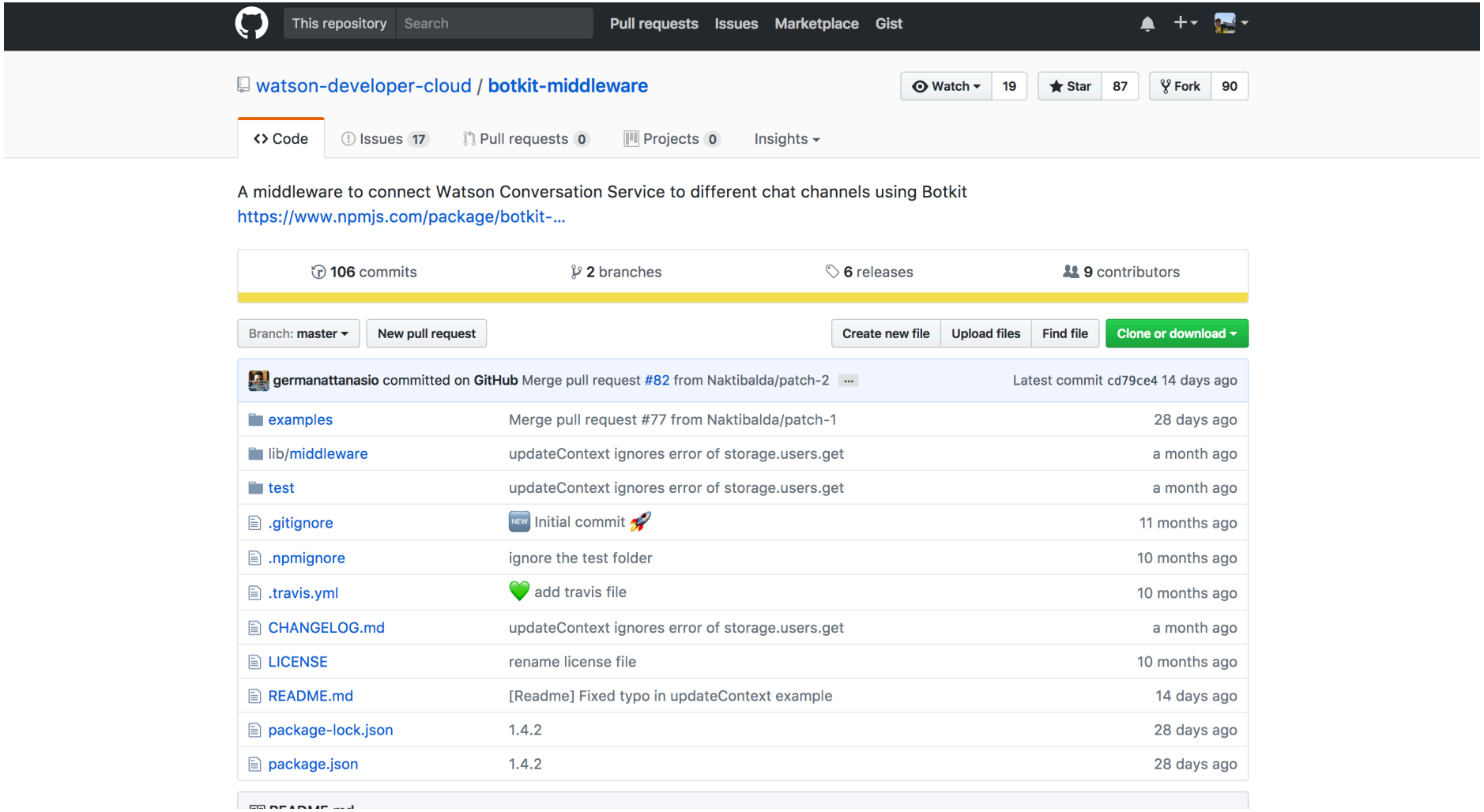
ibm.biz/create-cognitive-banking-chatbot



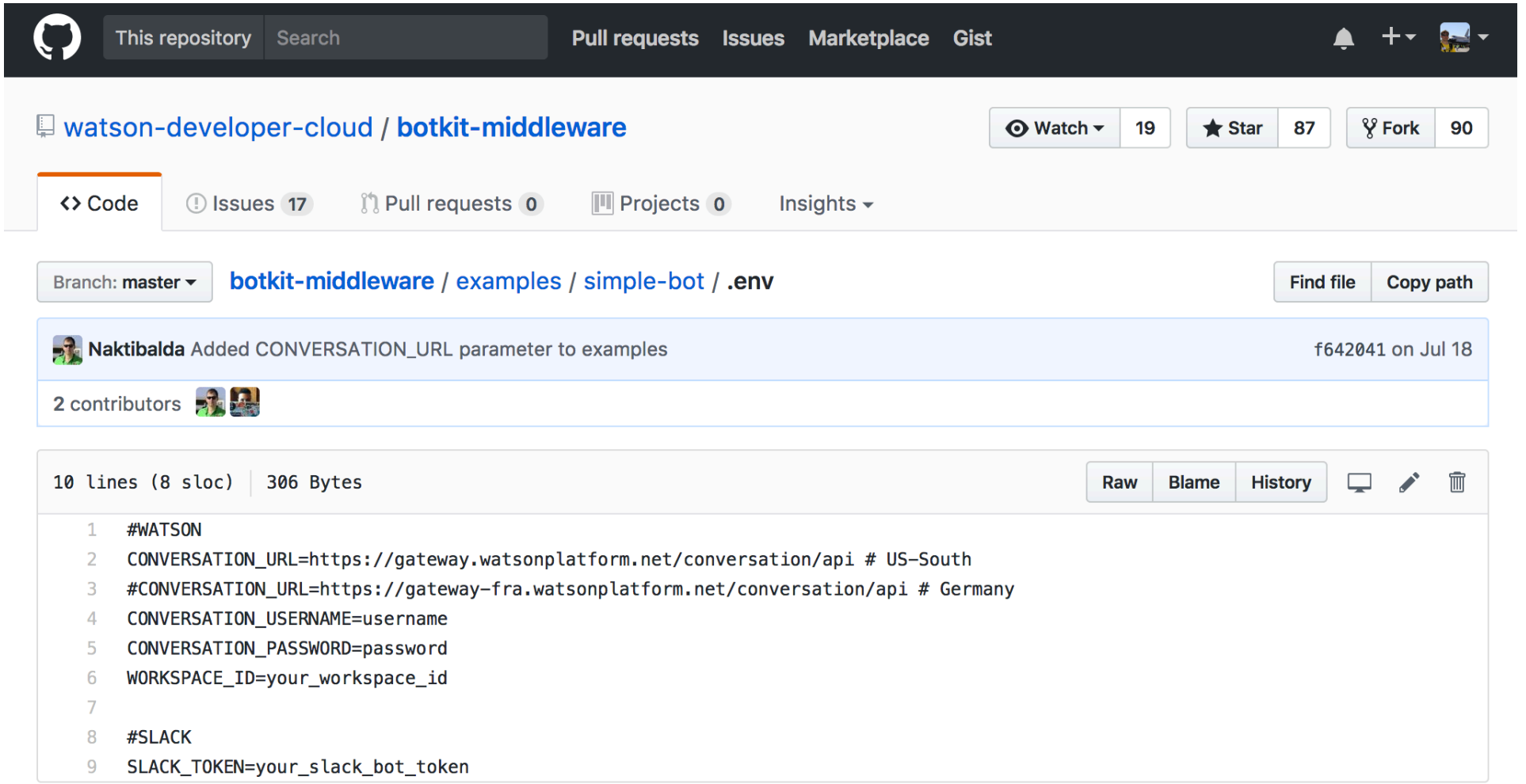

USER
CLOUD
EXTERNAL
USER
2
APPLICATION
6
BANK SERVICES
3
4
5
6
WATSON TONE ANALYZER
WATSON NATURAL LANGUAGE UNDERSTANDING
WATSON CONVERSATION
WATSON DISCOVERY
1
DISCOVERY COLLECTION

Connect to An Application

BotKit Middleware



Connecting An Application



Make API calls to third-party data sources, databases, and microservices to get data.

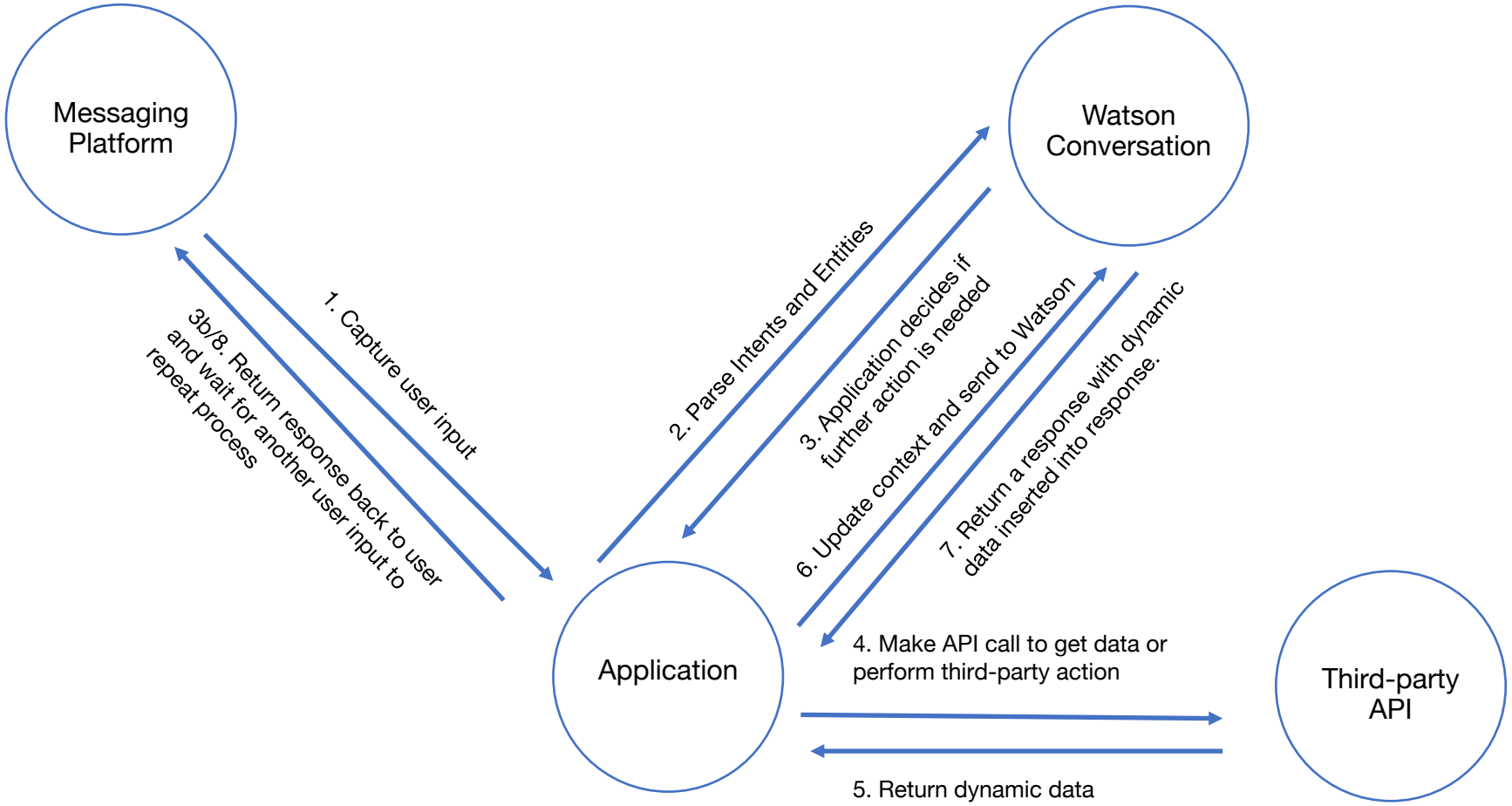
Handles each message received via Slack

If Conversation service returns output from a Dialog node, responds back to the Slack channel

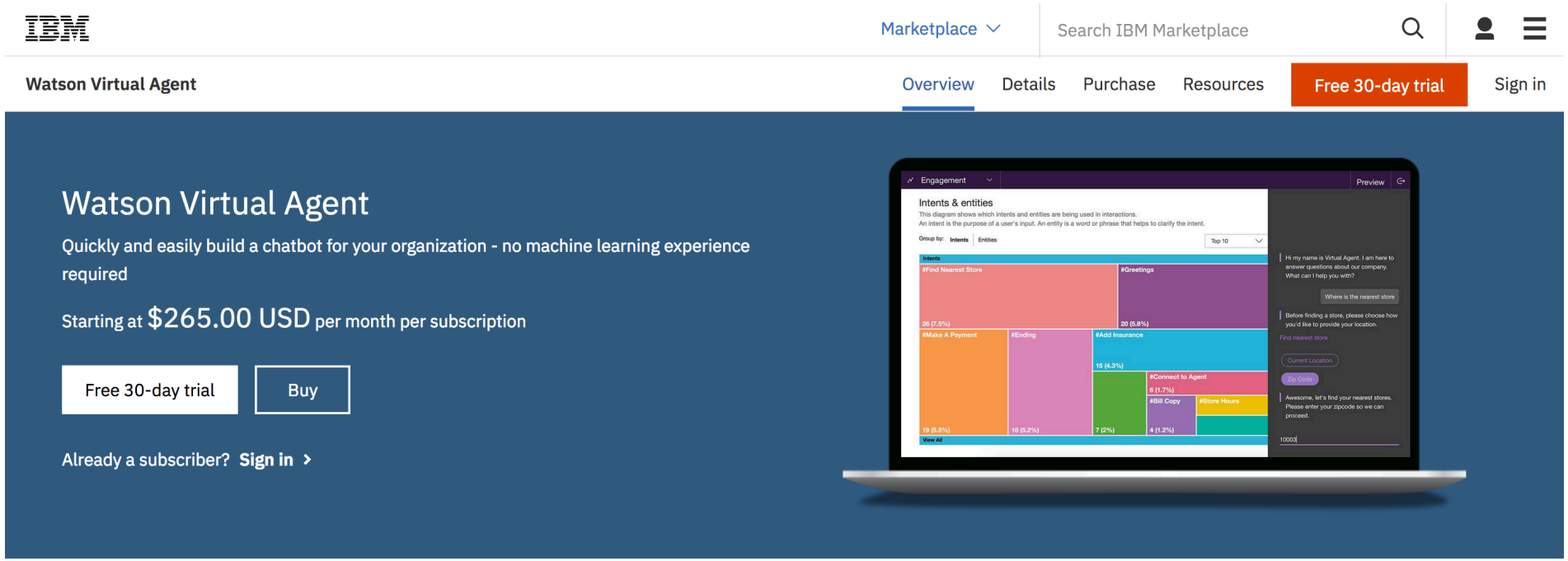
Application specific action to check balance and use it in the response back to the user.

```
function checkBalance(watsonResponse, callback) {
  //middleware.after function must pass a complete Watson respose to callback
  watsonResponse.context.validAccount = true;
  watsonResponse.context.accountBalance = 95.33;
  callback(null, watsonResponse);
}

watsonMiddleware.after = function(message, watsonResponse, callback) {
  //real action happens in middleware.after
  if (typeof watsonResponse !== 'undefined' && typeof watsonResponse.output !== 'undefined') {
    if (watsonResponse.output.action === 'check_balance') {
      return checkBalance(watsonResponse, callback);
    }
  }
  callback(null, watsonResponse);
};

var processWatsonResponse = function(bot, message) {
  if (message.watsonError) {
    return bot.reply(message, "I'm sorry, but for technical reasons I can't respond to your message");
  }

  if (typeof message.watsonData.output !== 'undefined') {
    //send "Please wait" to users
    bot.reply(message, message.watsonData.output.text.join('\n'));

    if (message.watsonData.output.action === 'check_balance') {
      var newMessage = clone(message);
      newMessage.text = 'balance result';
      //send to watson
      watsonMiddleware.interpret(bot, newMessage, function() {
        //send results to user
        processWatsonResponse(bot, newMessage);
      });
    }
  }
};

controller.on('message_received', processWatsonResponse);
```

Connecting Third-Party APIs



Pre-trained Virtual Agents



What it can do for your business

Watson Virtual Agent is a new way to provide automated services to your customers. It offers a cognitive, conversational self-service experience that can provide answers and take action. You can easily customize your Watson Virtual Agent to fit your specific business needs, provide custom content and match your business brand. Additionally, deep analytics provide insights on your customer's engagement with the Watson Virtual Agent and help with the understanding of your constantly changing customer's needs.

Resources

- https://www.ibm.com/watson/services/conversation/
- https://conversation-demo.mybluemix.net
- https://github.com/watson-developer-cloud
- https://www.ibm.com/us-en/marketplace/cognitive-customer-engagement
- http://developer.ibm.com/code/