

Google Home Final Presentation

Mehmet Kardan, Hanna Köb, Mathias Meinschad, Daniel Linter

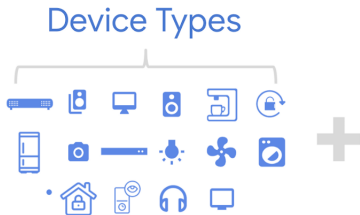
University of Innsbruck - STI

June 22, 2020

- 1 Introduction to Google Home
 - Device Types & Traits
 - Execution Path
- 2 Introduction to Dialogflow
 - Intents
 - Entities
 - Architecture
- 3 Implementation
 - Entities Extraction
- 4 Queries, Intents and Response Handling
 - Intents
 - Queries
 - Response Handling
- 5 Problems
- 6 Live Demo



- Founded by Google in 2016
- Development through Googles developer console and Dialogflow
- Creating skills pretty easy
- No programming skills required



Traits

Attributes - SYNC

Defines configuration options for traits.

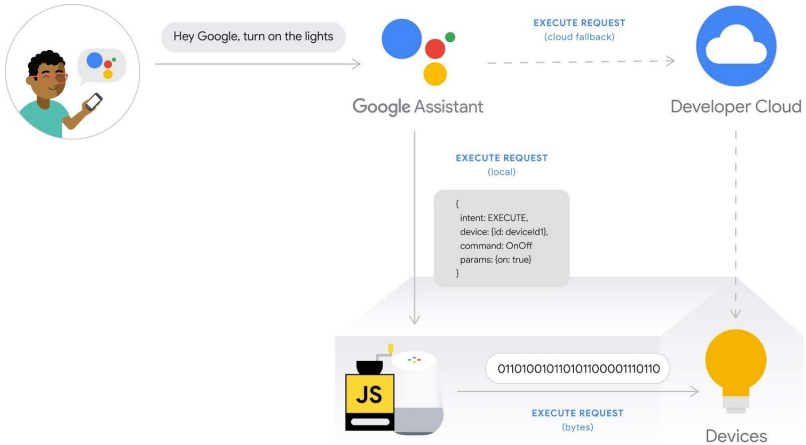
States - QUERY & EXECUTE

Defines the real-time state of the device.

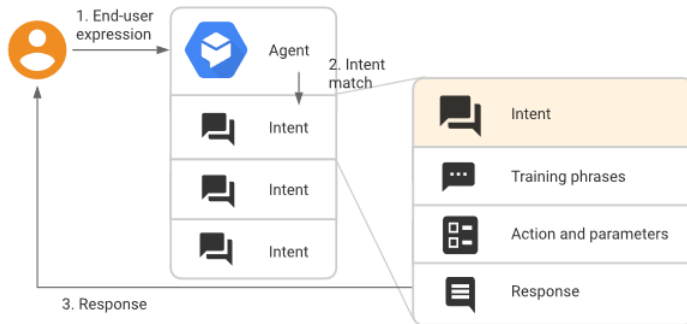
Commands - EXECUTE

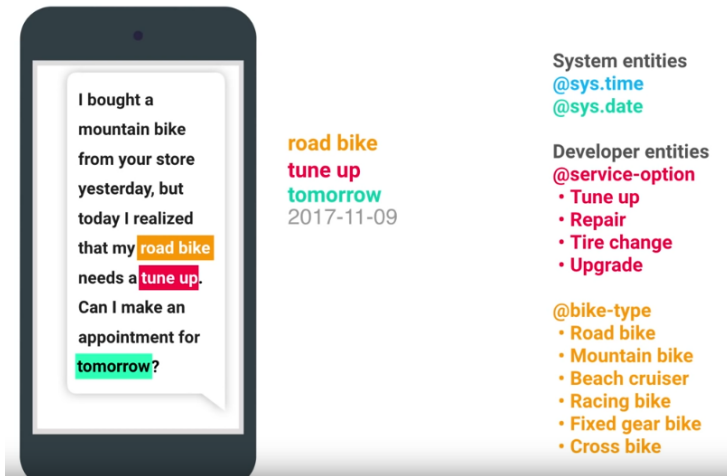
Used to change the state or perform a function on the device.

- Various device types (air purifier to yogurt maker)
- Capabilities of a device \Rightarrow traits



- Service developed and provided by Google
- Natural language tool to create conversational user interfaces for apps, chatbots, etc.
- By adding 'Training phrases' Dialogflow automatically trains the machine learning model





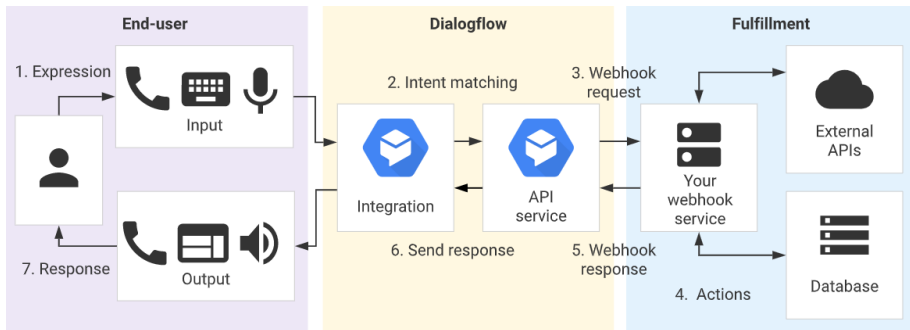
I bought a mountain bike from your store yesterday, but today I realized that my **road bike** needs a **tune up**. Can I make an appointment for **tomorrow**?

road bike
tune up
tomorrow
2017-11-09

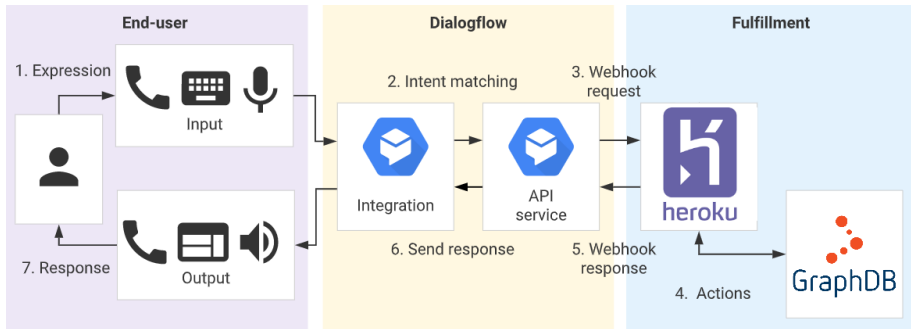
System entities
@sys.time
@sys.date

Developer entities
@service-option
• Tune up
• Repair
• Tire change
• Upgrade

@bike-type
• Road bike
• Mountain bike
• Beach cruiser
• Racing bike
• Fixed gear bike
• Cross bike



- Coding in JavaScript



- An entity for each class in the knowledge graph is created

```
SELECT DISTINCT ?class
WHERE {
    ?s a ?class .
}
```

- Then schema.org's property *name* is used to fill the entities with values

```
PREFIX sc: <http://schema.org/>
SELECT ?name
WHERE {
    ?resource a [entityTypeName] .
    ?resource sc:name ?name .
}
```

Concept



SAVE



☒ Define synonyms ☐ Regexp entity ☐ Allow automated expansion ☒ Fuzzy matching

Search entries



1

OF 3



Knowledge Generation Tool

Knowledge Generation Tool

Manual Editing Tool

Manual Editing Tool

Manual Editing

Manual Editing

Unstructured Content Extraction
Technique

Unstructured Content Extraction Technique

Small Data Set

Small Data Set

Queries, Intents and Response Handling

- Divided questions into the Intents.
- Defined a query for specific intent type.
- Response handling function for the queries.

- 6 question types defined in Dialogflow
 - Difference Type Question
 - Example Type Questions
 - List Type Questions
 - Narrower Type Question
 - Step Type Questions
 - What is Type Questions

- Queries are encoded with Query String module.
- For each intent type, we defined the following queries.

Query for "What is Type Question"

```
PREFIX schema: <http://schema.org/>
PREFIX kgs: <http://knowledgegraphbook.ai/schema/>
select ?description ?purpose where {
  {
    ?Concept schema:name ?name.
    OPTIONAL { ?Concept schema:description ?description . }
    OPTIONAL { ?Concept kgs:purpose ?purpose . }
    filter (LCASE(?name) = LCASE("${parameter}"))
  }
  union
  {
    ?Concept schema:alternateName ?name.
    OPTIONAL { ?Concept schema:description ?description . }
    OPTIONAL { ?Concept kgs:purpose ?purpose . }
    filter (LCASE(?name) = LCASE("${parameter}"))
  }
}
```


Query for "Difference Type Questions"

```
PREFIX schema: <http://schema.org/>
PREFIX kgbs: <http://knowledgegraphbook.ai/schema/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

select ?description where {
  ?Concept schema:name ?name.
  ?relatesTo rdf:type ?type.
  ?type rdfs:label ?labelcheck.
  ?relatesTo schema:name ?check_name.
  ?relatesTo schema:description ?description.
  filter (LCASE(?name) = LCASE("${first_parameter}") || LCASE(?name) = LCASE("${second_parameter}"))
  filter(?labelcheck = "Difference")
  filter (contains (LCASE(?check_name),LCASE("${first_parameter}")) || contains (LCASE(?
check_name),LCASE("${second_parameter}")))
}
```

Query for "List Type Questions"

```
PREFIX schema: <http://schema.org/>
PREFIX kgbs: <http://knowledgegraphbook.ai/schema/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
select ?description where {
  {
    ?Concept schema:name ?name
    OPTIONAL {?Concept skos:narrower ?specialization.}
    OPTIONAL {?specialization schema:name ?description.}
    filter (LCASE(?name) = LCASE("${parameter}")) .
  }
  union
  {
    ?Concept schema:alternateName ?name
    OPTIONAL {?Concept skos:narrower ?specialization.}
    OPTIONAL {?specialization schema:name ?description.}
    filter (LCASE(?name) = LCASE("${parameter}")) .
  }
}
```

Query for "Example Type Questions"

```
PREFIX schema: <http://schema.org/>
```

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
```

```
select ?name ?description where {  
  ?Concept schema:name ?target.  
  optional { ?Concept skos:example ?example . }  
  optional { ?example schema:name ?name . }  
  optional { ?example schema:description ?description . }  
  filter (LCASE(?target) = LCASE("${parameter}"))  
}
```

```
PREFIX schema: <http://schema.org/>
PREFIX kgsb: <http://knowledgegraphbook.ai/schema/>

select ?description where {
  ?Concept schema:name ?name .
  ?Concept schema:step: ?Object .
  OPTIONAL { ?Object schema:text ?description . }
  filter contains (LCASE(?name), LCASE("${parameter}")) .
}
```

Query for "Narrower Type Questions"

```
PREFIX schema: <http://schema.org/>
PREFIX kgs: <http://www.knowledgegraphbook.ai/schema/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

select ?name where {
    ?Concept schema:name ?target.
    ?Concept skos:narrower ?example .
    ?example schema:name ?name .
    filter contains(LCASE(?target), LCASE("${parameter}"))
}
```

Response Handling

```
function collectResponseDataFromGraphDb(response) {  
  var ret_array = []  
  for (i = 0; i < response.data.results.bindings.length; i++) {  
    if ('purpose' in response.data.results.bindings[i]) {  
      ret_array[i] = response.data.results.bindings[i].purpose.value;  
    }  
    else if ('description' in response.data.results.bindings[i]) {  
      ret_array[i] = response.data.results.bindings[i].description.value;  
    }  
    else if ('name' in response.data.results.bindings[i]) {  
      ret_array[i] = response.data.results.bindings[i].name.value;  
    }  
    else {  
      ret_array[i] = "No description or purpose found in result of Graph DB."  
    }  
  }  
  return ret_array;  
}
```

- 3 different types of response will be handled.
- Default text for error handling.

- Changing namespaces of GraphDB
- No JavaScript library for GraphDB with authentication

Live Demo

Thank you for your
attention!