Mycroft Simple Documentation

Release 0.1.0

Matthew Scholefield

Jun 23, 2017

Contents:

Mycroft, made simple

This repository contains experimental code restructing with things like Padatious integration.

CHAPTER 1

Subpackages

mycroft.clients package

mycroft.clients.mycroft_client module

class mycroft.clients.mycroft_client.**MycroftClient**(*query_manager*)
Bases: object

Provides common behavior like sending and receiving queries Examples clients include the voice client and text client

on_response(*format_manager*)
Called after send_query. Use format_manager to get outputted response

quit()
Should send a signal to stop the main thread of the client

run()
Executes the main thread for the client

send_query(*query*)
Ask a question and trigger on_response when an answer is found

mycroft.clients.text_client module

class mycroft.clients.text_client.**TextClient**(**args*, ***kwargs*)
Bases: *mycroft.clients.mycroft_client.MycroftClient*

Interact with Mycroft via a terminal

on_response(*format_manager*)

quit()

run()

mycroft.engines package

mycroft.engines.intent_engine module

class mycroft.engines.intent_engine.**IntentEngine**(*path_manager*)
Bases: object

Interface for intent engines

calc_intents(*query*)
Run the intent engine to determine the probability of each intent against the query :param query: input sentence as a single string :return: dict of intent: intent_data where

Example return data: { 'name': 'TimeSkill:time.ask', 'confidence': '0.65', 'matches': {'location': 'new york'} }

on_intents_loaded()
Override to run code when all intents have been registered

try_register_intent(**args*, ***kwargs*)
Attempt to register intent with given arguments :rtype str :returns intent name if parsed parameters, otherwise ""

mycroft.engines.intent_engine.**extract_intent_name**(*namespaced_name*)
Ex. TimeSkill:time.ask -> time.ask

mycroft.engines.intent_engine.**extract_skill_name**(*namespaced_name*)
Ex. TimeSkill:time.ask -> TimeSkill

mycroft.engines.intent_engine.**make_namespaced**(*intent_name*, *skill_name*)
Mangle the intent name so that it doesn't conflict and to save the skill name in the same string

mycroft.engines.padatious_engine module

class mycroft.engines.padatious_engine.**PadatiousEngine**(*path_manager*)
Bases: *mycroft.engines.intent_engine.IntentEngine*

Interface for Padatious intent engine

GIT_BRANCH = 'feature/mycroft-simple'

GIT_URL = 'https://github.com/MatthewScholefield/padatious-mycroft.git'

HOST = '127.0.0.1'

PORT = 8014

calc_intents(*query*)

on_intents_loaded()

try_register_intent(*skill_name*, *intent_name*)

mycroft.formats package

mycroft.formats.dialog_format module

class mycroft.formats.dialog_format.**DialogFormat**(*path_manager*)
Bases: *mycroft.formats.mycroft_format.MycroftFormat*

Format data into sentences

generate(*name*, *results*)

mycroft.formats.mycroft_format module

class mycroft.formats.mycroft_format.**MycroftFormat**(*path_manager*)
Bases: object

Base class to provide an interface for different types of "formats"

Formats are modes to display key-value data. For instance, the DialogFormat puts data into sentences The EnclosureFormat could put data into visual faceplate animations

generate(*name*, *results*)
Internally format the data from the results Depending on the format, this can be accessed different ways

Parameters

- **name** – namespaced intent name
- **results** – dict containing all data from the skill

Returns nothing

mycroft.managers package

mycroft.managers.client_manager module

class mycroft.managers.client_manager.**ClientManager**(*client_classes*, **args*, ***kwargs*)
Bases: object

Holds all clients to start and stop them

quit()
Sends a signal to all clients to quit. Does not wait for clients to exit (non blocking)

start()
Starts all clients in different threads (non blocking)

mycroft.managers.format_manager module

class mycroft.managers.format_manager.**FormatManager**(*path_manager*)
Bases: object

Holds all formats and provides an interface to access them

as_dialog
Get data as a sentence

generate(*name*, *results*)

mycroft.managers.intent_manager module

class mycroft.managers.intent_manager.**IntentManager**(*path_manager*)
Bases: object

Used to handle creating both intents and intent engines

calc_results(*query*)
Find the best intent and run the handler to find the results

Parameters **query** – input sentence

Returns name, results

Rtype name string (namespaced intent)

Rtype results dict

on_intents_loaded()

register_fallback(*handler*)
Register a function to be called as a general knowledge fallback

Parameters **handler** – function that receives query and returns a dict of results, one of which is 'confidence' note: register_fallback in the MycroftSkill base class automatically manages results

register_intent(*skill_name*, *intent*, *handler*)
Register an intent via the corresponding intent engine It tries passing the arguments to each engine until one can interpret it correctly

Parameters

- **skill_name** –
- **intent** – argument used to build intent; can be anything
- **handler** – function that receives intent_data and returns a dict of results note: register_intent in the MycroftSkill base class automatically manages results

Returns nothing

mycroft.managers.path_manager module

class mycroft.managers.path_manager.**PathManager**(*base_path*)
Bases: object

Retreives directories and files used by Mycroft

dialog_dir(*skill_name*)

intent_dir(*skill_name*)

mod_path

padatious_exe
The locally compiled Padatious executable

skill_dir(*skill_name*)

skills_dir

vocab_dir(*skill_name*)

mycroft.managers.query_manager module

class mycroft.managers.query_manager.**QueryManager**(*intent_manager*, *format_manager*)
Bases: object

Launches queries in separate threads

on_response(*callback*)
Assign a callback to be run whenever a new response comes in

send_query(*query*)
Starts calculating a query in a new thread

mycroft.managers.skill_manager module

class mycroft.managers.skill_manager.**SkillManager**(*intent_manager*, *path_manager*)
Bases: object

Dynamically loads skills

load_skills()
Looks in the skill folder and loads the CamelCase equivalent class of the snake case folder This class should be inside the skill.py file. Example:

skills/

time_skill/ skill.py - class TimeSkill(MycroftSkill):

weather_skill/ skill.py - class WeatherSkill(MycroftSkill):

mycroft.skills package

Subpackages

mycroft.skills.duck_duck_go_skill package

mycroft.skills.duck_duck_go_skill.skill module

class mycroft.skills.duck_duck_go_skill.skill.**DuckDuckGoSkill**(**args*, ***kwargs*)
Bases: *mycroft.skills.mycroft_skill.MycroftSkill*

Fallback skill that queries DuckDuckGo's instant answer API

fallback(*query*)

fallback_no_question(*query*)

mycroft.skills.quit_skill package

mycroft.skills.quit_skill.skill module

class mycroft.skills.quit_skill.skill.**QuitSkill**(**args*, ***kwargs*)
Bases: *mycroft.skills.mycroft_skill.MycroftSkill*

mycroft.skills.time_skill package

mycroft.skills.time_skill.skill module

class mycroft.skills.time_skill.skill.**TimeSkill**(**args*, ***kwargs*)
Bases: *mycroft.skills.mycroft_skill.MycroftSkill*

date(*intent_data*)

time(*intent_data*)

mycroft.skills.unknown_skill package

mycroft.skills.unknown_skill.skill module

class mycroft.skills.unknown_skill.skill.**UnknownSkill**(**args*, ***kwargs*)
Bases: *mycroft.skills.mycroft_skill.MycroftSkill*

calc_results(*intent_data*)

mycroft.skills.mycroft_skill module

class mycroft.skills.mycroft_skill.**MycroftSkill**(*intent_manager*)
Bases: object

Base class for all Mycroft skills

add_result(*key*, *value*)

Adds a result from the skill. For example:

self.add_result('time', '11:45 PM') Except, of course, '11:45 PM' would be something generated from an API

Results can be both general and granular. Another example: self.add_result('time_seconds', 23)

create_handler(*handler*, *skill_name=None*)
Wrap the skill handler to return added results

register_fallback(*handler*)
Same as register_intent except the handler only receives a query and is only activated when all other Mycroft intents fail

register_intent(*name*, *handler*)
Set a function to be called when the intent called 'name' is activated In this handler the skill should receive a dict called intent_data and call self.add_result() to add output data. Nothing should be returned from the handler

CHAPTER 2

mycroft.mycroft_thread module

mycroft.mycroft_thread.**quit**()

mycroft.mycroft_thread.**set_quit_action**(*callback*)

CHAPTER 3

mycroft.util module

mycroft.util.**split_sentences**(*text*)
Turns a string of multiple sentences into a list of separate ones As a side effect, .?! at the end of a sentence are removed

mycroft.util.**to_camel**(*snake*)
time_skill -> TimeSkill

mycroft.util.**to_snake**(*camel*)
TimeSkill -> time_skill

Python Module Index

m

Index

O

P

Q

R

S

T

U

V