go-jsbox-location Documentation

Release 0.1.1

Praekelt Foundation

October 06, 2014

Contents

Contents:

CHAPTER 1

LocationState Class

class `LocationState`(*name*, *opts*)

A state which requests a location from the user, and sets the user data through getting the location data from the Google Maps API. It may also request a second prompt from the user to further refine the location if the first prompt wasn't clear enough.

Arguments

- **name** (*string*) – name used to identify and refer to the state
- **opts.question** (*string_or_LazyText*) – The question to first display to the user. Defaults to `What is your address?`.
- **opts.refine_question** (*string_or_LazyText*) – The question to display to the user when selecting a location from a list if the first search query wasn't clear enough. Defaults to `Please select your location from the following:`
- **opts.error_question** (*string_or_LazyText*) – The question to display to the user when no locations are found for their search term. It will keep requesting until results are found. Defaults to `Error: No results for your search term. Please try another search term.`
- **opts.continue_session** (*boolean*) – whether or not this is the last state in a session. Defaults to `true`.
- **opts.send_reply** (*boolean*) – whether or not a reply should be sent to the user's message. Defaults to `true`.
- **opts.next** (*function_or_string_or_object*) – The state that the user should visit after this state. May either be the name of the next state, an options object representing the next state, or a function of the form `f(content)` returning either, where `content` is the input given by the user. If `next` is `null` or not defined, the state machine will be left in the current state. See `State.set_next_state()`. Defaults to `null`
- **opts.options_per_page** (*integer*) – The maximum limit for the amount of choices on each page. Defaults to `8`.
- **opts.characters_per_page** (*integer*) – The maximum limit for the amount of characters on each page. Defaults to `160`. Whichever one of characters_per_page or option_per_page is reached first will be chosen.
- **opts.next_text** (*string*) – The text to display for the next page option. Defaults to `Next`.
- **opts.previous_text** (*string*) – The text to display for the previous page option. Defaults to `Previous`.

- **opts.store_fields** (*array*) – An array of field names from the google maps API results that should be stored. Defaults to `['formatted_address']`. Data is stored as a string representation of the object in the `location` field in the contact store.
- **opts.namespace** (*string*) – The namespace to use when storing the contact details, ie. `location:.....`. Defaults to `location`.
- **opts.events** (*object*) – Optional event name-listener mappings to bind.

Example:

```
self.states.add('states:example-locationState', function(name){
    return new LocationState(name, {
        question: ["Welcome to the location app.",
            "What is your current address?"].join("\n"),
        next: "states:end",
        previous_text: "Prev",
        store_fields: ["geometry.location", "formatted_address"]
    });
});
```

CHAPTER 2

Testing Class

This class is used to automatically create the fixtures required to test LocationState states.

add_location(*opts*)

Adds location data to the fixtures

Arguments

- **opts.request_url** (*string*) – URL for the HTTP request. Defaults to `"http://maps.googleapis.com/maps/api/geocode/json"`
- **opts.request** (*string*) – The address that is to be queried. Defaults to `"Friend Street, South Africa"`.
- **opts.address_list** (*array_of_strings*) – An array of the list of *formatted_address*'s that should be sent in the response. If response_data is included, this will be ignored. Defaults to `["Friend Street, Cape Town 7925, South Africa"]`
- **opts.response_data** (*array_of_objects*) – An array of objects that represents the response from the gmaps API

Usage: Create an instance of

```
locations = LocationState.testing()
```

and then add the fixtures in

```
locations.fixtures
```

to the fixtures list beforeEach test. Then add locations during testing:

```
locations.add_location({
    request:"New Street",
    address_list:["New Street 1", "New Street 2"]
});
```

add_locations(*opts*)

Adds an array location data to the fixtures.

Arguments

- **opts_array** (*array_of_objects*) – An array of opts

Related: `add_location()`

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

A

L