DESIGN DOCUMENT

SEARCH WIDGET

April 16, 2016

Vineet Purswani: 12807813
Ankit Pachouri: 12123
IIT Kanpur
Computer Science and Engineering

INTRODUCTION

Search widget is a smart object designed by combining three already existing widgets - button, entry, hover. This widget will enable developers to inject list of labels along with their corresponding callbacks and to get a search bar functioning over the data provided in their applications. The sections below will get more into design and implementation of the widget, individually covering the following topics - Sub-widgets, Communication among components, Control Flow, API Documentation, Sample Application and References.

SUB-WIDGETS

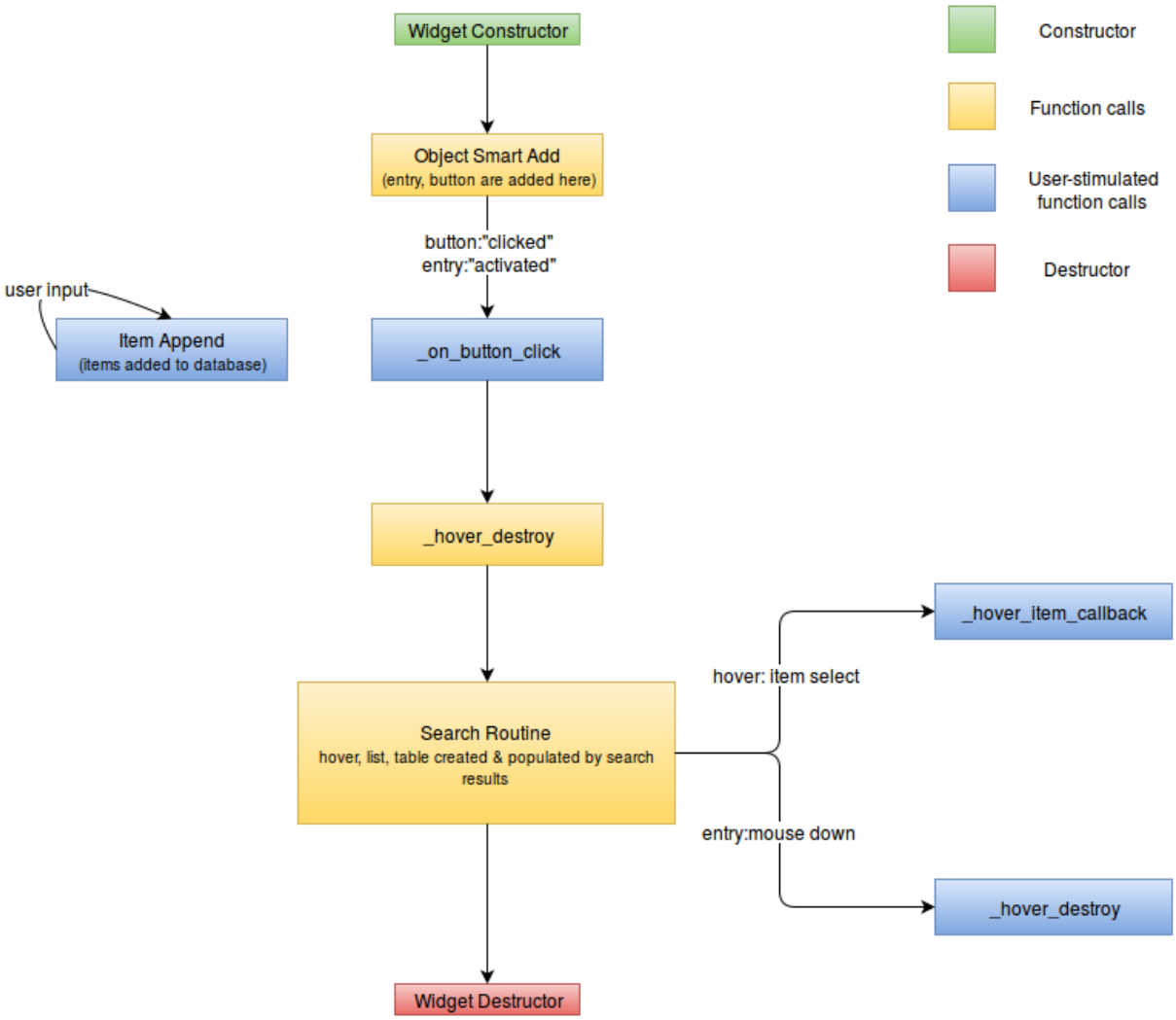
What and Why

- **Entry** - Entry widget is used to get the users input, a substring that is to be searched into the given database (i.e. list of labels).
- **Button** - Button widget enables users to get results once pressed. It is kept next to entry widget just to give a feel of a search bar. Button press event is mapped to a callback which implements the actual logic of searching the entry text in the given database.
- **Hover** - Hover widget is used to display the search result, where every item is mapped to a callback provided by the developer. This hover object in itself has list widget contained into a table widget to give it a better look.

Communication

- **Entry:Button** - There is only once that these two widgets communicate, on button "clicked" event. A function callback takes entry widget's text and runs a substring search in the given label list.
- **Entry:Hover** - There is a two way communication between these two widgets, one when return key is pressed and "activated" event callback shoots up (same as button "clicked" event, search routine) and another when mouse down event is called for entry widget, where hover list is destroyed.
- **Button:Hover** - There is one way communication between these two widgets, i.e. when button "clicked" event takes place. The existing hover, if any, is destroyed and a new one is produced to fulfill the users search query.

Control Flow



API EXPOSED

- **elm_status_item_append**()
  Append an item to search widget's database with or without a callback.

  Arguments
  obj - search widget object
  label - item label
  func - callback function
  func_data - callback function data

SAMPLE APPLICATION

Sample application included with the project gives a very basic demonstration of a possible use of search widget in any application. It has an entry and a search widget. While adding items to search widget a callback is passed with each label which updates the entry widget with the hover list item text.

FURTHER WORK

- *Gengrid* or *genlist* could be used in place of list widget for functionalities and aesthetics purposes.
- More string matching options like case sensitivity could be given for better text control.
- Substring matching algorithms could be implemented for faster real-time results on large datasets. We would need algorithm specific data structures for best performance.

REFERENCES

Combobox Widget <https://git.enlightenment.org/devs/ami/smartobjects.git/tree/eo/combobox>

Status Widget <https://git.enlightenment.org/devs/ami/smartobjects.git/tree/status>

List Widget example application <https://git.enlightenment.org/core/elementary.git/tree/src/examples/list_example_03.c>