<MentionNotifier>

Object Design

<1.0>

<07/01/2018>

<Berk Ergun – 210CS2013>

Prepared for

CSE490 Software Engineering



Table of Contents

[1. Introduction 1](#_Toc503179872)

[1.1. Object Design Trade-offs 1](#_Toc503179873)

[1.2. Interface Documentation Guidelines 1](#_Toc503179875)

[i) Naming Conventions 1](#_Toc503179876)

[1.3. Definitions, Acronyms, and Abbreviations 2](#_Toc503179877)

[1.4. References 2](#_Toc503179878)

[2. Packages 3](#_Toc503179879)

[2.1. MentionNotifier.py 3](#_Toc503179880)

[2.2. GoogleSearchIter.py 3](#_Toc503179881)

[2.3. IntervalFuncTimer.py 3](#_Toc503179882)

[2.4. ArchiverComperator.py 3](#_Toc503179883)

[3. Class Interfaces 3](#_Toc503179884)

[3.1. MentionNotifier.py package 3](#_Toc503179885)

[3.2. GoogleSearchIter.py package 6](#_Toc503179886)

[3.3. IntervalFuncTimer.py package 6](#_Toc503179887)

[3.4. ArchiverComperator.py package 7](#_Toc503179888)

OBJECT DESIGN DOCUMENT [1]

# Introduction

This section of this document describes the general trade-offs made by the developer (e.g., buy vs. build, memory space vs. response time), guidelines and conventions (e.g., naming conventions).

## Object Design Trade-offs

## Delivery Time vs. Reliability

The application retrieves page sources from all kinds of pages that the Google Web Search engine brings out and the error handling for such a variety of code is out of the scope of this project. For Delivery Time’s sake, such error handling was left to exceptions.

* + 1. Delivery Time vs. Clarity

As the delivery time nears, we drifted further and further away from proper interface design and code clarity. While further supportability was not one of the requirements for our project, it still is a measure we had to take.

## Interface Documentation Guidelines

For documenting our project, we are using Sphinx to generate documentation from inline blocks of comments in our source code. We are using the reStructeredText syntax to represent our technical documentation.

## Naming Conventions

|  |  |  |
| --- | --- | --- |
| **Identifier Type** | **Rules for Naming** | **Examples** |
| **Packages** | The prefix of a unique package name is always written in mixed case ASCII letters, with the first letter of each word capitalized. | ArchiverComperator.py  MentionNotifier.py |
| **Classes** | Class names should be nouns, in mixed case with the first letter of each word capitalized. | class ResultBundle  class Row |
| **Methods** | Our Methods are verbs, either written in lowercase, with words separated by underscores ("\_"), or in mixed case with the first letter of each internal word capitalized. | google\_this()  addField() |
| **Variables** | Except for variables, all instance, class, and class constants are in lowercase with words separated by underscores ("\_"). | int i |

## Definitions, Acronyms, and Abbreviations

RAD: Requirements Analysis Document

SDD: Software Design Document

ODD: Object Design Document

Interface: A shared boundary between two or more components of a system that exchange information. [2]

Pickle: Python module implementing binary protocols for serializing and de-serializing a Python object structure. [3]

Kivy: Kivy is an open source Python library for developing mobile apps and other multitouch application software with a natural user interface (NUI). It can run on Android, iOS, Linux, OS X, and Windows. Distributed under the terms of the MIT license, Kivy is free and open source software. [4][5]

## References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. *IEEE 100: the authoritative dictionary of IEEE standards terms*. Institute of Electrical and Electronics Engineers, 2000
3. “12.1. pickle - Python object serialization¶.” *12.1. pickle - Python object serialization — Python 3.6.4 documentation*, docs.python.org/3/library/pickle.html.
4. “API Reference¶.” *API Reference — Kivy 1.10.1.dev0 documentation*, kivy.org/docs/api-index.html.
5. “Kivy (Framework).” *Wikipedia*, Wikimedia Foundation, 2 Jan. 2018, en.wikipedia.org/wiki/Kivy\_(framework).
6. “Sphinx Domains¶.” *Sphinx Domains — Sphinx 1.6.6 documentation*, www.sphinx-doc.org/en/stable/domains.html#the-python-domain.
7. Abenassi. “Abenassi/Google-Search-API.” *GitHub*, 17 Feb. 2017, github.com/abenassi/Google-Search-API/tree/master/google.
8. Jones, Richard. *A ReStructuredText Primer*, docutils.sourceforge.net/docs/user/rst/quickstart.html.
9. MentionNotifier - Requirements Analysis Document
10. MentionNotifier – Software Design Document

# Packages

## MentionNotifier.py

This package holds the Kivy Application of our project and mainly holds the functions used by the UI elements and the User.

## GoogleSearchIter.py

This package holds the modified Google-Search-API by github.com user, abenassi. Holds the functions required in creating the GoogleResult objects used by our system.

## IntervalFuncTimer.py

This package holds the IntervalFuncTimer class, a modified Timer object recursively calling upon itself. Used to make continuous searches.

## ArchiverComperator.py

This package holds the functions related to the creation, archival, retrieval and comparison of link:hash dictionaries both new and old.

# Class Interfaces

## MentionNotifier.py package

*class*MentionNotifier.**GoogleResultsList**(*\*\*kwargs*)

Custom UI element created by inheriting from Kivy class BoxLayout. Used as the layout for Row class elements to be listed.

**float\_updated**()

Brings the result Rows that are updated to the top of self.

|  |  |
| --- | --- |
| **Returns:** |  |

**list\_results**(*selection\_list*)

Function listing the results of a ResultBundle as a Row in self.

|  |  |
| --- | --- |
| **Parameters:** | **selection\_list** – List of searches. Used to pick the one that is selected. |
| **Returns:** | Returns None if no search is selected. |

**set\_blue**(*link*, *selection\_list*)

Marks a Row as checked if it’s link was used. This makes it’s Updated text color blue, instead of red.

|  |  |
| --- | --- |
| **Parameters:** | * **link** – Link that was just reacehd by the user. * **selection\_list** – List of searches. Used to pick the one that is selected. |
| **Returns:** |  |

**stop\_search**(*selection\_list*)

Sets a SearchBundle to cancel it’s thread and removes that SearchBundle from the list bundles.

|  |  |
| --- | --- |
| **Parameters:** | **selection\_list** – List of searches. Used to pick the one that is selected. |
| **Returns:** |  |

*class*MentionNotifier.**MentionNotifier**(*\*\*kwargs*)

Base UI element for the application inheriting from the kivy TabbedPanel class. The TabbedPanel widget manages different widgets in tabs, with a header area for the actual tab buttons and a content area for showing the current tab content.

**addField**(*text*, *selected=False*)

Function for adding a keyword in to the list of keywords.

|  |  |
| --- | --- |
| **Parameters:** | * **text** – Keyword to be added. * **selected** – Denotes whether the keyword t obe created is selected or not. Not accessible through user, for auxiliary purposes. Defaults to False. |
| **Returns:** | Returns empty if text == “”. |

**google\_this**(*search\_keywords*, *pages=2*)

Function repeatedly called by a ResultBundle’s intervalFuncTimer thread. Makes sure that the ResultBundle corresponding to it’s query is still in the bundles list, then retrieves it. Then retrieves the GoogleResult instances for the google query with given page depth. Then sets the ResultBundle update\_index by comparing it to a pickled iteration of the search, by matching the two’s queries.

|  |  |
| --- | --- |
| **Parameters:** | * **search\_keywords** – The google query to be searched. * **pages** – Depth of the google search to be done. |
| **Returns:** |  |

**google\_thread**(*pages*, *interval*)

Function for starting a search. Makes sure that a keywords is selected, then creates a google query string out of it. Then it creates a ResultBundle and creates it’s thread with the given interval parameter. The thread calls on to the google\_this function with the google query created and the number of pages to be searched. The function then adds this ResultBundle into the ResultBundle.bundles list and then starts the ResultBundle thread.

**removeField**()

Function for removing selected a keyword from the keywords list.

|  |  |
| --- | --- |
| **Returns:** |  |

*class*MentionNotifier.**MentionNotifierApp**(*\*\*kwargs*)

Custom class inheriting from the Kivy App class. The base for our application, holds the entry point for the Kivy run loops.

**build**()

Initializes the application; it will be called only once. Returns the MentionNotifier widget (tree), that will be used as the root widget and added to the window.

|  |  |
| --- | --- |
| **Returns:** | MentionNotifier instance |

**on\_stop**()

Event function fired when the application stops. Useed to cancel each search thread running.

|  |  |
| --- | --- |
| **Returns:** |  |

*class*MentionNotifier.**ResultBundle**(*thread*)

Class holding the data for each ongoing search task.

|  |  |
| --- | --- |
| **Variables:** | **bundles** (*list*) – List holding the ResultBundle instances. |

*class*MentionNotifier.**Row**(*\*\*kwargs*)

Custom UI element created by inheriting from Kivy class BoxLayout. Used as the view class to represent each GoogleResult column in the ResultsList.

|  |  |
| --- | --- |
| **Variables:** | * **name** (*str*) – Kivy property that represents a string value. Title of a google result. * **link** (*str*) – Kivy property that represents a string value. Link of a google result. * **description** (*str*) – Kivy property that represents a string value. Description text of a google result. * **was\_updated** (*bool*) – Kivy property that represents a boolean value. Denotes whether the result was updated or not. * **last\_updated** (*str*) – Kivy property that represents a string value. Date when the object was last updated. * **time\_updated** (*str*) – Kivy property that represents a string value. Time when the object was last updated. * **was\_checked** (*str*) – Kivy property that represents a string value. Denotes whether the result was checked by the user or not. |

*class*MentionNotifier.**SelectableLabel**(*\*\*kwargs*)

Custom selectable UI element created by inheriting from two Kivy classes. Used for representing and selecting each keyword or search.

**apply\_selection**(*rv*, *index*, *is\_selected*)

Applies a selection to the data dictionary of the RecycleView object in the background.

|  |  |
| --- | --- |
| **Parameters:** | * **rv** – Recycleview instance. * **index** – Index of self. * **is\_selected** – Whether if self is selected. |
| **Returns:** |  |

**on\_touch\_down**(*touch*)

Internal Widget Kivy class function overrridden. When a new touch is registered, tests if it collides with the

bounding box of another known gesture. Used for click detection for our purposes.

|  |  |
| --- | --- |
| **Parameters:** | **touch** – Touch Kivy object used to initialize the gesture container. |
| **Returns:** |  |

**refresh\_view\_attrs**(*rv*, *index*, *data*)

Internal RecycleDataView Kivy class function overridden. Syncs the view and brings it up to date with the data.

|  |  |
| --- | --- |
| **Parameters:** | * **rv** – RecycleView instance. * **index** – The index of the SelectableLabel. * **data** – Updated data dictionary kept in the back. |
| **Returns:** | Calls the function onto self. |
| **Variables:** | **index** – Sets the index as self’s index within a list. |

*class*MentionNotifier.**SelectableRecycleBoxLayout**(*\*\*kwargs*)

Custom UI element created by inheriting from multiple Kivy classes. Used for listing Keywords and Searches. The FocusBehavior mixin class provides keyboard focus behavior. (instances of self can be cycled by pressing tab.) LayoutSelectionBehavior adds selection behavior to layouts. RecycleBoxLayout provides a BoxLayout type layout for use with RecycleView widget.

## GoogleSearchIter.py package

*class*GoogleSearchIter.**GoogleResult**

Represents a google search result.

GoogleSearchIter.**GoogleResultGen**(*li*, *page*, *index*)

Aux function to be called for when creating a GoogleResult.

GoogleSearchIter.**dummy\_func**()

Aux dummy function.

GoogleSearchIter.**search\_iter**(*bundle*, *iterFunc=None*, *pages=1*, *lang='en'*, *void=True*)

Returns a list of GoogleResult.

|  |  |
| --- | --- |
| **Parameters:** | * **query** (*str*) – String to search in google. * **pages** (*int*) – Number of pages where results must be taken. |
| **Returns:** | A GoogleResult object. |

## IntervalFuncTimer.py package

*class*IntervalFuncTimer.**intervalFuncTimer**(*t*, *func*, *xargs*)

Class representing a perpetual python Timer object. Calls a it’s function at initialization and then keeps calling it in set intervals. Can be stopped by setting the running bool flag as False.

|  |  |
| --- | --- |
| **Variables:** | **thread\_count** (*int*) – Number of intervalFuncTimer instances running. |

**cancel**()

Function that sets self as not running. As there is no thread termination in Python threads, the thread will be running till it’s nex iteration.

**handle\_function**()

Function deciding whether the self.func function should be called by checking the running flag. If it does, initializes and starts a new Timer for self that will call upon self.func at initialization.

**start**()

Function that sets self as running and starts self’s Timer thread.

## ArchiverComperator.py package

ArchiverComperator.**compare\_dicts**(*dict1*, *bundle*)

The comparator function for a new and old site\_dict dictionary values.

* If a link exists in both dicts and their hashes match, it’ll do nothing.
* If a link exists in both dicts but their hashes don’t match, then it’s and marked as updated.
* If the old dict throws a KeyError, meaning it cannot find a link value in the new site\_dict, it’ll pass.

After that, the links that are NOT in the old site\_dict but are in the new site\_dict are marked as updated.

|  |  |
| --- | --- |
| **Parameters:** | * **dict1** – Old site\_dict retrieved from a pickle stored. * **bundle** – Bundle from which it’s site\_dict will be retrieved from and it’s last\_updated and was\_checked values |

will be set.

|  |  |
| --- | --- |
| **Returns:** | update\_count, update\_index |

ArchiverComperator.**iterThis**(*res*, *bundle*)

Function called for while iterating over GoogleResult’s to hash them. Get’s the source of a link (with a 3 second timmeout), pulls the <body> part from it and hashes that string. If body can’t be retrieved, it uses the whole page instead. And if the page cannot be retreived at all, the hash is saved as an all-zero hash. Then stores it in the ResultBundle’s site\_dict as a value for the link’s key.

|  |  |
| --- | --- |
| **Parameters:** | * **res** – GoogleResult that the link of will be used. * **bundle** – ResultBundle that it’s site\_dict will be used. |
| **Returns:** |  |

ArchiverComperator.**pickle\_dict**(*bundle*)

Function called for when storing and comparing old and new site\_dict values. Sets the update index, last\_updated and was\_checked values to default if they don’t exist, then retrieves the pickle file corresponding to the bundle. Then it will compare the site\_dict’s of the old and the new and return the update\_index which’ll show whether a GoogleResult was updated or not. Then it’ll save the new\_dict into a pickle file, overwriting any existing one if it exists. Finally it’ll file a notification announcing the number of updates within the bundle.

|  |  |
| --- | --- |
| **Parameters:** | **bundle** – ResultBundle |
| **Returns:** |  |