iosr-crawler Documentation

Release 1.0.0

Grzegorz Miejski, Mateusz Radko, Krzysztof Trzepla

June 08, 2015

Contents

Contents:

CHAPTER 1

Crawler Engine

Contents:

1.1 Crawler Engine

class engine.CrawlerEngine.**CrawlerEngine**

class **CustomSpider**(**a*, ***kw*)

allowed_domains = ['en.wikipedia.org']

config = {'start_urls': 'http://en.wikipedia.org/wiki/Programming_language', 'allowed_domains': 'en.wikipedia.o

config_file = <closed file '/home/docs/checkouts/readthedocs.org/user_builds/iosr-crawler/checkouts/latest/src/e

config_path = '/home/docs/checkouts/readthedocs.org/user_builds/iosr-crawler/checkouts/latest/src/engine/conf.c

crawler

handles_request(*request*)

log(*message*, *level=10*, ***kw*)
Log the given messages at the given log level. Always use this method to send log messages from your spider

make_requests_from_url(*url*)

name = 'spider'

parse(*response*)

static parse_page(*response*)

parse_start_url(*response*)

process_results(*response*, *results*)

rules = (<scrapy.contrib.spiders.crawl.Rule object at 0x7fb1ab416450>,)

set_crawler(*crawler*)

settings

start_requests()

start_urls = ['http://en.wikipedia.org/wiki/Programming_language']

CrawlerEngine.**add_query**(*user_id*, *query*)
Add crawling query for given user.

Parameters

- **user_id** (*int*) – ID of user associated with the query.
- **query** (*str*) – User's query.

CrawlerEngine.**get_urls**(*query*)
Retrieves all URLs associated with given query form database.

Returns list of URLs.

CrawlerEngine.**get_user_queries**(*user_id*)
Retrieves user queries form database.

Parameters **user_id** (*int*) – Id of user associated with the query.

Returns list of user queries.

static CrawlerEngine.**notify_agents**()
Notifies agent about new crawling query.

CrawlerEngine.**start_crawling**()
Notifies all agents and if crawling process in not started, starts it.

1.2 DB Engine

class engine.db_engine.DbEngine.**DbEngine**

add_keywords(*query*, *keywords*, *bucket_name='keywords'*)
Adds keywords for given query to database.

Parameters

- **query** (*str*) – Query associated with keywords.
- **keywords** (*list*) – List of keywords produced from the query.

add_query(*user_id*, *query*, *bucket_name='user_queries'*)
Adds query to database.

Parameters

- **user_id** (*int*) – Id of user associated with the query.
- **query** (*str*) – Query to be saved into database.

add_url(*query*, *url*, *bucket_name='urls'*)
Adds url for given query to database.

Parameters

- **query** (*str*) – Query associated with url.
- **url** (*str*) – URL of page satisfying search requirements.

get_all_queries(*bucket_name='all_queries'*)
Retrieves all queries form database.

Returns list of all queries.

get_keywords(*query*, *bucket_name='keywords'*)
Retrieves all keywords associated with given query form database.

Returns list of keywords.

get_urls(*query*, *bucket_name='urls'*)
Retrieves all URLs associated with given query form database.

Returns list of URLs.

get_user_queries(*user_id*, *bucket_name='user_queries'*)
Retrieves user queries form database.

Parameters **user_id** (*int*) – Id of user associated with the query.

Returns list of user queries.

1.3 Search Engine

class engine.search_engine.SearchEngine.**SearchEngine**

reload_queries()
Reloads queries from database.

search(*content*)
Iterates over all queries and returns those for which number of found keywords satisfies search threshold.

Parameters **content** (*str*) – content of web page associated with the URL.

Returns list of queries for which search threshold was satisfied.

search_in_url(*url*, *content*)
Search web page content in order to find keywords.

Parameters

- **url** (*str*) – URL of web page being crawled.
- **content** (*str*) – content of web page associated with the URL.

CHAPTER 2

Extractor

Contents:

2.1 Extractor

class nlp.extractor.**NLPExtractor**

build_stop_word_regex()
Creates stop word regex.

Returns stop word pattern.

static calculate_word_scores(*phrase_list*)
Calculates words scores based on their frequency and degree.

Parameters **phrase_list** (*list*) – List of phrases to be processed.

Returns mapping between word and its score.

static generate_candidate_keyword_scores(*phrase_list*, *word_score*)
Generates scores for candidate keywords.

Parameters

- **phrase_list** (*list*) – List of phrases to be processed.
- **word_score** (*map*) – Mapping between word and its score.

Returns mapping between phrases and their scores.

static generate_candidate_keywords(*sentence_list*, *stopword_pattern*)
Generates list of keywords candidates.

Parameters

- **sentence_list** (*list*) – List of sentences to be processed.
- **stopword_pattern** (*str*) – Stop words pattern.

Returns list of keywords

static is_number(*word*)
Checks whether word is a number.

Parameters **word** (*str*) – Word to be checked.

Returns True or False

load_stop_words()
Utility function to load stop words from a file and return as a list of words.

Returns list A list of stop words.

run(*text*)
Extracts keywords from the text.

Parameters **text** (*str*) – Text to be processed.

Returns list of keywords.

static separate_words(*text*, *min_word_return_size*)
Utility function to return a list of all words that are have a length greater than a specified number of characters.

Parameters

- **text** (*str*) – The text that must be split in to words.
- **min_word_return_size** (*int*) – The minimum no of characters a word must have to be included.

Returns list of separated words.

static split_sentences(*text*)
Utility function to return a list of sentences.

Parameters **text** (*str*) – The text that must be split in to sentences.

Returns sentences List of sentences created due to split.

CHAPTER 3

User Interface

Contents:

3.1 Forms

class ui.forms.**QueryForm**(*data=None*, *files=None*, *auto_id=u'id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*)

add_error(*field*, *error*)
Update the content of *self._errors*.

The *field* argument is the name of the field to which the errors should be added. If its value is None the errors will be treated as NON_FIELD_ERRORS.

The *error* argument can be a single error, a list of errors, or a dictionary that maps field names to lists of errors. What we define as an "error" can be either a simple string or an instance of ValidationError with its message attribute set and what we define as list or dictionary can be an actual *list* or *dict* or an instance of ValidationError with its *error_list* or *error_dict* attribute set.

If *error* is a dictionary, the *field* argument *must* be None and errors will be added to the fields that correspond to the keys of the dictionary.

add_initial_prefix(*field_name*)
Add a 'initial' prefix for checking dynamic initial values

add_prefix(*field_name*)
Returns the field name with a prefix appended, if this Form has a prefix set.

Subclasses may wish to override.

as_p()
Returns this form rendered as HTML <p>s.

as_table()
Returns this form rendered as HTML <tr>s – excluding the <table></table>.

as_ul()
Returns this form rendered as HTML s – excluding the .

base_fields = OrderedDict([('query', <django.forms.fields.CharField object at 0x7fb1aaf4cb10>)])

changed_data

clean()
Hook for doing any extra form-wide cleaning after Field.clean() has been called on every field. Any ValidationError raised by this method will not be associated with a particular field; it will have a special-case association with the field named '__all__'.

declared_fields = OrderedDict([('query', <django.forms.fields.CharField object at 0x7fb1aaf4cb10>)])

errors
Returns an ErrorDict for the data provided for the form

full_clean()
Cleans all of self.data and populates self._errors and self.cleaned_data.

has_changed()
Returns True if data differs from initial.

has_error(*field*, *code=None*)

hidden_fields()
Returns a list of all the BoundField objects that are hidden fields. Useful for manual form layout in templates.

is_multipart()
Returns True if the form needs to be multipart-encoded, i.e. it has FileInput. Otherwise, False.

is_valid()
Returns True if the form has no errors. Otherwise, False. If errors are being ignored, returns False.

media

non_field_errors()
Returns an ErrorList of errors that aren't associated with a particular field – i.e., from Form.clean(). Returns an empty ErrorList if there are none.

visible_fields()
Returns a list of BoundField objects that aren't hidden fields. The opposite of the hidden_fields() method.

3.2 Models

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

e

n

u

A

B

C

D

E

F

G

H