

---

# **eWRT - Extensible Web Retrieval Toolkit Documentation**

***Release 0.1***

**Mister X**

**Jul 11, 2017**



---

## Contents

---

|          |                            |           |
|----------|----------------------------|-----------|
| <b>1</b> | <b>Installation</b>        | <b>3</b>  |
| <b>2</b> | <b>eWRT Package</b>        | <b>5</b>  |
| 2.1      | eWRT Package . . . . .     | 5         |
| 2.2      | Subpackages . . . . .      | 5         |
| <b>3</b> | <b>Features</b>            | <b>23</b> |
| <b>4</b> | <b>Indices and tables</b>  | <b>25</b> |
|          | <b>Python Module Index</b> | <b>27</b> |



Knowledge capture in the age of massive Web data requires robust and scalable mechanisms to acquire, consolidate and pre-process large amounts of heterogeneous data. The Extensible Web Retrieval Toolkit (eWRT) is modular open-source Python API that addresses this requirement. It retrieves social data from Web sources such as Delicious, Flickr, Yahoo! and Wikipedia, including various helper classes for effective caching and data management. The toolkit provides components for content acquisition and caching, low-level natural language processing functionalities such as language detection, phonetic string similarity measures, and methods for string normalization.

eWRT has been jointly developed by researchers from [MODUL University Vienna](#), [webLyzard technology](#), the [University of Applied Sciences Chur](#), and the [Vienna University of Economics and Business](#). The library is currently being extended as part of the [uComp Project](#), which investigates Embedded Human Computation for Knowledge Extraction and Evaluation.

Checkout: eWRT on [gitweb](#).

Contents:



# CHAPTER 1

---

## Installation

---

todo: Finish this part

Access to the Repository

Clone the eWRT git repository::

```
git clone http://git.semanticlab.net/eWRT.git
```

Or directly install using pip:

```
pip install git+http://git.semanticlab.net/eWRT.git
```

Dependencies

- python 2.6 or higher
- python-nose >=0.11



### eWRT Package

@package eWRT

The easy Web Retrieval Toolkit

### Subpackages

#### access Package

##### db Module

##### file Module

Convenient methods for file access

@package eWRT.access.file Created on Dec 6, 2012

@author: albert

**class** eWRT.access.file.**CompressedFile** (*fname, mode='rb'*)

Bases: `object`

An intelligent file object that transparently opens compressed files.

**COMPRESSION\_EXT** = ('bz2', 'gz')

**classmethod** **get\_extension\_list** (*fname*)

**Parameters** **fname** – the file name to analyze

**Return type** a list of file extensions of this file ignoring extensions indicating file compression.

e.g. 'x/y/test.awp.csv.bz2' -> ['csv', 'awp']

```
classmethod open (fname, mode='rb')
class eWRT.access.file.TestFile
    Bases: object
    TESTSTRING = 'this is a test'
    test_formats ()
```

## http Module

## config Package

### config Package

@package eWRT.config evaluates ~/.eWRT/siteconfig.py and publishes the values

## input Package

### input Package

@package eWRT.input

Modules supporting data input, conversion and cleanup.

### Subpackages

#### clean Package

##### clean Package

@package eWRT.input.clean

data cleanup modules.

#### text Module

#### conv Package

##### conv Package

@package eWRT.input.conv

Classes for converting various input formats into each other.

#### doc Module

@package eWRT.input.conv.doc converts Microsoft Word documents into text

```
class eWRT.input.conv.doc.HtmlToText
```

```
    Bases: object
```

```
    converts HTML into text requires a converter
```

```
    static getText (word_document_content)
```

```
        @param[in] word_document_content the content of the html page to convert @param[in] encoding the
        document encoding @returns the text representation of the Web page
```

## html Module

```
@package eWRT.input.conv.html converts HTML pages into text
```

```
class eWRT.input.conv.html.HtmlToText
```

```
    Bases: object
```

```
    converts HTML into text requires a converter
```

```
    static getText (html_content, encoding='utf8')
```

```
        @param[in] html_content the content of the html page to convert @param[in] encoding the document
        encoding @returns the text representation of the Web page
```

```
class eWRT.input.conv.html.TestHtmlToText
```

```
    Bases: object
```

```
    testBorderCases ()
```

```
    testConversion ()
```

## pdf Module

```
@package eWRT.input.conv.pdf converts PDF documents into text
```

```
class eWRT.input.conv.pdf.HtmlToText
```

```
    Bases: object
```

```
    converts HTML into text requires a converter
```

```
    static getText (pdf_content)
```

```
        @param[in] pdf_content the content of the html page to convert @param[in] encoding the document en-
        coding @returns the text representation of the Web page
```

## Subpackages

### cxl Package

### cx1 Package

### corpus Package

## Subpackages

### bbc Package

### bbc Package

@package DetectLanguage.reuters a generic method to retrieve text from the reuters corpus

```
class eWRT.input.corpus.bbc.BBCGetCorpus (filePattern='*')
    Bases: object
    An iterator over all documents
    static getTitle (text)
        returns the title of a given text
    next ()
```

### reuters Package

### reuters Module

### stock Package

### stock Package

### Subpackages

### lib Package

### lib Package

### Result Module

```
class eWRT.lib.Result.Result (id, name)

    getAttributes ()
        todo: return attributes of Result
    getId ()
    getName ()
```

### ResultSet Module

```
class eWRT.lib.ResultSet.ResultSet (id=None, name=None, content=None)

    addContent (newContent)
    getId ()
    getName ()
    next ()
    static printRS (resultSet, filler=0)
    refresh ()
```

**setContent** (*content*)

## Webservice Module

**class** eWRT.lib.Webservice.**Webservice**

Bases: `object`

**login** ()

implement me

## apihelber Module

eWRT.lib.apihelber.**info** (*object, spacing=10, collapse=1*)

Print methods and doc strings. Takes module, class, list, dictionary, or string.

## Subpackages

### thirdparty Package

#### Subpackages

#### advas Package

#### advas Package

source code from the AdvaS Advanced Search project version: 0.2.3

## phonetics Module

eWRT.lib.thirdparty.advas.phonetics.**caverphone** (*term*)

returns the language key using the caverphone algorithm 2.0

eWRT.lib.thirdparty.advas.phonetics.**metaphone** (*term*)

returns metaphone code for a given string

eWRT.lib.thirdparty.advas.phonetics.**nysiis** (*term*)

returns New York State Identification and Intelligence Algorithm (NYSIIS) code for the given term

eWRT.lib.thirdparty.advas.phonetics.**soundex** (*term*)

Return the soundex value to a string argument.

## ontology Package

### ontology Package

@package eWRT.ontology.eval Evaluates ontologies based on their `_internal_` characteristics such as term coherence, internal integrity, ...

For evaluations against a reference ontology @see eWRT.ontology.compare

## Subpackages

compare Package

## Subpackages

relations Package

relations Package

relationtypes Package

relationtypes Package

eval Package

## Subpackages

terminology Package

terminology Package

visualize Package

visualize Package

## output Package

## Subpackages

plot Package

plot Package

radar\_chart Module

## stat Package

stat Package

@package eWRT.stat

## Subpackages

coherence Package

## coherence Package

@package eWRT.ws.stat.coherence Determines how strongly two terms are connected to each other

**class** eWRT.stat.coherence.Coherence (dataSource, cache=True)

Bases: object

@class Coherence abstract class for computing the coherence between terms

**static** getCoherence (nx, ny, nt)

@param[in] nx counts of term1 @param[in] ny counts of term2 @param[in] nt counts of term1 together with term2 @returns the coherence

**getTermCoherence** (t1, t2)

@param[in] t1 term1 @param[in] t2 term2 @returns the coherence between these two terms

**class** eWRT.stat.coherence.DiceCoherence (dataSource, cache=True)

Bases: eWRT.stat.coherence.Coherence

@class DiceCoherence computes the dice coherence for the given terms

**static** getCoherence (nx, ny, nt)

@param[in] nx counts of term1 @param[in] ny counts of term2 @param[in] nt counts of term1 together with term2 @returns the coherence

**class** eWRT.stat.coherence.PMICoherence (dataSource, cache=True)

Bases: eWRT.stat.coherence.Coherence

@class PMICoherence computes the coherence based on the pointwise mutual information (PMI)

**static** getCoherence (nx, ny, nt)

@param[in] nx counts of term1 @param[in] ny counts of term2 @param[in] nt counts of term1 together with term2 @returns the coherence

**class** eWRT.stat.coherence.TestCoherence (methodName='runTest')

Bases: unittest.case.TestCase

**testDice** ()

tests the computation of the dice coefficient based on the example in

[http://en.wikipedia.org/wiki/Dice's\\_coefficient](http://en.wikipedia.org/wiki/Dice's_coefficient)

**testPMI** ()

tests the computation of the PMI based on the results from wilson's paper

**testPMIZero** ()

tests the handling of PMI values of no counts are found

## eval Package

### metrics Module

@package eWRT.ws.stat.eval.metrics Standard IR evaluation metrics such as

- precision
- recall
- F1 measure

```
class eWRT.stat.eval.metrics.TestEvaluationMetrics
    Bases: object
    tests the evaluation metrics

    a = set([8, 1, 3, 9])
    b = set([1, 10, 3, 12])
    c = set([1, 3])
    d = set([10, 11])
    testFMeasure()
    testPrecision()
    testRecall()

eWRT.stat.eval.metrics.fMeasure(p, r, beta=1.0)
    returns the F-measure for the given precision and recall @param[in] p precision @param[in] r recall @param[in]
    beta weight used to compute the f mesure @returns the F-Measure

eWRT.stat.eval.metrics.precision(relevant, retrieved)
    returns the precision of the given sets @param[in] relevant set of relevant terms @param[in] retrieved set of
    retrieved terms @returns the precision

eWRT.stat.eval.metrics.recall(relevant, retrieved)
    returns the recall of the given sets @param[in] relevant set of relevant terms @param[in] retrieved set of retrieved
    terms @returns the recall
```

## language Package

### language Package

```
@package eWRT.stat.language
language detection

class eWRT.stat.language.DetectLanguageTest(methodName='runTest')
    Bases: unittest.case.TestCase

    test_detect_language()
        tests the language detection based on examples

    test_exceptions()
        results for empty strings

eWRT.stat.language.detect_language(text)
    detects the most probable language for the given text

eWRT.stat.language.get_lang_name(fname)

eWRT.stat.language.read_wordlist(fname)
    reads a language wordlist from a file
```

## string Package

### string Package

## spelling Module

## util Package

### advLogging Module

```
class eWRT.util.advLogging.SNMPHandler (moduleName)
    Bases: logging.Handler

    Logging handler for sending SNMP traps

    emit (record)
        sends the message

class eWRT.util.advLogging.TestHandler (methodName='runTest')
    Bases: unittest.case.TestCase

    testHandler ()
        tests the handler

eWRT.util.advLogging.sendSNMPTrap (message, module, level)
    sends a SNMP message @param message: String that should be sent @param level: SNMP levels ('ok', 'warn-
    ing', 'critical', 'unknown')
```

### assert Module

@package eWRT.util.assert Assertion based counters

Examples: see unittests

```
class eWRT.util.assert.AssertReturnValue (evalExpression, counterNameTrue, counterName-False)
    Bases: object

    decorator class used to time functions

class eWRT.util.assert.TestAssertReturnValue
    Bases: object

    testAssertCounter ()
        verifies the assertion counters
```

### async Module

@package eWRT.util.async asynchronous procedure calls

@warning this library is still a draft and might change considerable

```
class eWRT.util.async.Async (cache_dir, cache_nesting_level=0, cache_file_suffix='',
                             max_processes=8, debug_dir=None)
    Bases: eWRT.util.cache.DiskCache

    Asynchronous Call Handling

    fetch (cache_file)

    getPostHashfile (cmd)
        returns an identifier representing the object which is compatible to the identifiers returned by the
        eWRT.util.cache.* classes.
```

**has\_processes\_limit\_reached()**

closes finished processes and verifies whether we have already reached the maximum number of processes

**post(*cmd*)**

checks whether the given command is already cached and calls the command otherwise. @param[in] cmdline command to call @returns the hash required to fetch this object

**class** eWRT.util.async.**TestAsync**

Bases: *object*

unittests covering the class async

**TEST\_CACHE\_DIR** = *'./test-async'*

**setUp()**

**tearDown()**

**testDebugMode()**

tests the debug mode

**testMaxProcessLimit()**

tests the max process limit

## cache Module

@package eWRT.util.cache caches arbitrary objects

**class** eWRT.util.cache.**Cache** (*fn=None*)

Bases: *object*

An abstract class for caching functions

**fetch** (*fetch\_function, \*args, \*\*kwargs*)

Fetches a object from the cache or computes it by calling the *fetch\_function*. The *objectId* is computed based on the function arguments

**fetchObjectId** (*key, fetch\_function, \*args, \*\*kwargs*)

Fetches a object from the cache or computes it by calling the *fetch\_function*. The *key* helps to determine whether the object is already in the cache or not.

**static getKey** (*\*args, \*\*kwargs*)

returns the key for a set of function parameters

**static getObjectId** (*obj*)

returns an identifier representing the object

**class** eWRT.util.cache.**DiskCache** (*cache\_dir, cache\_nesting\_level=0, cache\_file\_suffix='', fn=None*)

Bases: *eWRT.util.cache.Cache*

@class DiskCache Caches arbitrary functions based on the function's arguments (*fetch*) or on a user defined key (*fetchObjectId*)

@remarks This version of DiskCached is threadsafe

**fetch** (*fetch\_function, \*args, \*\*kwargs*)

**fetches the object with the given id, querying**

1. the cache and
2. the *fetch\_function*

if the `fetch_function` is called, the functions result is saved in the cache

::param `fetch_function`: function to call if the result is not in the cache ::param `args`: arguments ::param `kargs`: optional keyword arguments

::returns: the object (retrieved from the cache or computed)

**fetchObjectId** (*key, fetch\_function, \*args, \*\*kargs*)

**fetches the object with the given id, querying**

- the cache and
- the `fetch_function`

if the `fetch_function` is called, the functions result is saved in the cache

::param `key`: key to fetch ::param `fetch_function`: function to call if the result is not in the cache ::param `args`: arguments ::param `kargs`: optional keyword arguments

::returns: the object (retrieved from the cache or computed)

**getCacheStatistics** ()

returns statistics regarding the cache's hit/miss ratio

**class** `eWRT.util.cache.DiskCached` (*cache\_dir, cache\_nesting\_level=0, cache\_file\_suffix=''*)

Bases: `object`

Decorator based on `Cache` for caching arbitrary function calls usage:

`@DiskCached('/.cache/myfunction')` `def myfunction(*args):`

`@remarks` This version of `DiskCached` is threadsafe

**cache**

**class** `eWRT.util.cache.IterableCache` (*cache\_dir, cache\_nesting\_level=0, cache\_file\_suffix='', fn=None*)

Bases: `eWRT.util.cache.DiskCache`

caches arbitrary iterable content identified by an identifier

**fetchObjectId** (*key, function, \*args, \*\*kargs*)

**fetches the object with the given id, querying**

1. the cache and
2. the function

if the function is called, the functions result is saved in the cache

::param `key`: key to fetch ::param `function`: function to call if the result is not in the cache ::param `args`: arguments ::param `kargs`: optional keyword arguments

::returns: the object (retrieved from the cache or computed)

**next** ()

**class** `eWRT.util.cache.MemoryCache` (*max\_cache\_size=0, fn=None*)

Bases: `eWRT.util.cache.Cache`

`@class` `MemoryCached`

Caches arbitrary functions based on the function's arguments (`fetch`) or on a user defined key (`fetchObjectId`)

**fetch** (*fetch\_function, \*args, \*\*kargs*)

**fetchObjectId** (*key, fetch\_function, \*args, \*\*kargs*)

**garbage\_collect\_cache()**  
removes the object which have not been in use for the longest time

**max\_cache\_size**

**class** eWRT.util.cache.**MemoryCached**(arg)  
Bases: eWRT.util.cache.MemoryCache

Decorator based on MemoryCache for caching arbitrary function calls usage:

@MemoryCached or @MemoryCached(max\_cache\_size) def myfunction(\*args): ...

**class** eWRT.util.cache.**RedisCache**(max\_cache\_size=0, fn=None, host='localhost', port=6379, db=0)  
Bases: eWRT.util.cache.Cache

**fetch**(fetch\_function, \*args, \*\*kwargs)

**fetchObjectId**(key, fetch\_function, \*args, \*\*kwargs)

**garbage\_collect\_cache()**  
removes the object which have not been in use for the longest time

**class** eWRT.util.cache.**RedisCached**(arg)  
Bases: eWRT.util.cache.RedisCache

Decorator based on MemoryCache for caching arbitrary function calls usage:

@MemoryCached or @MemoryCached(max\_cache\_size) def myfunction(\*args): ...

eWRT.util.cache.**get\_unique\_temp\_file**(fname)

## exception Module

@package eWRT.util.exception

**exception** eWRT.util.exception.**SNMPException**(module\_name, msg, level='warning')  
Bases: exceptions.Exception

reports an exception to snmp

**class** eWRT.util.exception.**TestSNMPException**  
Bases: object

**testQuoting()**

**testSNMPException()**  
tests an SNMP exception

## execute Module

@package eWRT.util.execute Helpers for executing third party modules

**class** eWRT.util.execute.**TestExecute**  
Bases: object

**test\_pipe\_content()**

eWRT.util.execute.**pipe\_content**(cmd, stdin=None)  
Pipes the content through the given command. @param[in] cmd: command to be executed @param[in] stdin: standard input @return: (exit\_status, stdout)

## monitoring Module

The class NSCA-Service helps the enables to send test-results over the NSCA service to Nagios. This is a more reliable way of sending messages directly from programs than with SNMP. On the one hand because they are not only associated to single service, but also as the configuration is easier

== Configuration ==

- Install the package nsca

aptitude install nsca

- On the monitoring system:

\*\* edit the file `"/etc/nsca.cfg"`: \*\* set a password and an appropriate encryption method

- On the host:

\*\* edit the file `"/etc/send_nsca.cfg"` \*\* enter the above password and encryption method

**class** eWRT.util.monitoring.NSCA

Bases: `object`

class NSCA Service helps sending test results to Nagios

**static send** (*message*, *status*, *service\_name*, *src\_host=None*, *performance=[]*, *monitoringServer="tor.wu.ac.at"*)  
sends the

**class** eWRT.util.monitoring.Performance (*label*, *value*, *unit=''*, *warn=''*, *critical=''*, *min=''*, *max=''*)

Bases: `object`

## pickleIterator Module

pickelIterator

**class** eWRT.util.pickleIterator.AbstractIterator (*fname*, *file\_mode=None*)

Bases: `object`

Abstract Iterator class used to implement ReadPickleIterator and WritePickleIterator

**close** ()

**classmethod get\_filename** (*fname*)

**next** ()

Python 2 compatibility

**class** eWRT.util.pickleIterator.ReadPickleIterator (*fname*)

Bases: `eWRT.util.pickleIterator.AbstractIterator`

provides an iterator over pickeled elements

**class** eWRT.util.pickleIterator.WritePickleIterator (*fname*)

Bases: `eWRT.util.pickleIterator.AbstractIterator`

writes pickeled elements (available as iterator) to a file

**dump** (*obj*)

dumps the following object to the pickle file

## profile Module

@package eWRT.util.profile google like profiling :)  
@warning this library is still a draft and might change considerable  
eWRT.util.profile.**profile** (*fn*, *logfile='profile.awi'*)  
    profile function  
eWRT.util.profile.**profileFunction** ()  
    function to profile  
eWRT.util.profile.**testProfile** ()  
    test the profiling

## timing Module

@package eWRT.util.timing timing of arbitrary method calls  
Examples: see unittests  
**class** eWRT.util.timing.**Timed** (*f*)  
    Bases: *object*  
    decorator class used to time functions  
    **clear** ()  
**class** eWRT.util.timing.**TimedTest** (*methodName='runTest'*)  
    Bases: *unittest.case.TestCase*  
    **testCallStatistics** ()  
    **testClearCallStatistics** ()  
    **testReturnValue** ()

## ws Package

### ws Package

@package eWRT.ws Web Service access.  
**class** eWRT.ws.**AbstractIterableWebSource**  
    Bases: *eWRT.ws.AbstractWebSource*  
    web source implementing several calls to the API iterating over search terms and over API-specific maximal number of results restriction  
    **DEFAULT\_COMMAND** = None  
    **DEFAULT\_FORMAT** = None  
    **DEFAULT\_MAX\_RESULTS** = None  
    **DEFAULT\_START\_INDEX** = None  
    **RESULT\_PATH** = None  
    **invoke\_iterator** (*search\_terms*, *max\_results*, *from\_date=None*, *to\_date=None*, *command=None*, *output\_format=None*)  
        iterator: iterates over search terms and API requests

**process\_output** (*results, path*)

results' post-processor: iterates over the API responses and calls the output convertor

**request** (*search\_term, current\_index, max\_results, from\_date=None, to\_date=None, command=None, output\_format=None*)

calls the web source's API

**search\_documents** (*search\_terms, max\_results=None, from\_date=None, to\_date=None, \*\*kwargs*)

calls iterator and results' post-processor

**class** eWRT.ws.**AbstractWebSource**

Bases: `object`

a raw Web Source object

**MAPPING** = `None`

**NAME** = `None`

**SUPPORTED\_PARAMS** = `None`

**post\_search** (*search\_params*)

override this function to perform post-run tasks

**pre\_search** (*search\_params*)

override this function to perform pre-run tasks

**search\_documents** (*search\_terms, max\_results=None, from\_date=None, to\_date=None, \*\*kwargs*)

runs the actual search / calls the webservice / API ...

## TagInfoService Module

**class** eWRT.ws.TagInfoService.**TagInfoService**

Bases: `object`

Class for fetching assigned tags

**getRelatedTags** (*tags, retrieveTagInfo=False*)

returns a the count of related tags @param list/tuple of tags @param retrieveTagInfo determines whether we will retrieve the tagInfo for the related tags @returns list of related tags with a count of their occurrence

**getTagInfo** (*tags*)

returns the count for the given tags @param list/tuple of tags @returns number of counts

## WebDataSource Module

**class** eWRT.ws.WebDataSource.**WebDataSource**

Bases: `object`

A WebDataSource performs search queries against Web Sources such as Youtube, Twitter, ...

**search** (*search\_terms*)

returns the count for the given tags @param search\_terms: a list of search terms @returns a list of matching documents

## Subpackages

### amazon Package

amazon Package

example Module

delicious Package

delicious Package

test Module

facebook Package

facebook Package

fbBatchRequest Module

flickr Package

flickr Package

test Module

geoLyzard Package

geoLyzard Package

test Module

geonames Package

geonames Package

Subpackages

gazetteer Package

gazetteer Package

exception Module

util Package

georesolve Module

google Package

google Package

plus Module

googletrends Package

googletrends Package

test Module

opencalais Package

opencalais Package

test Module

rest Package

rest Package

rss Package

rss Package

technorati Package

technorati Package

test Module

twitter Package

twitter Package

wikipedia Package

wikipedia Package

descriptor Module

distance Module

Subpackages

initialize Package

`initialize` Package

`create-distance-db` Module

`wot` Package

`wot` Package

`yahoo` Package

`yahoo` Package

`term_extractor` Module

`youtube` Package

`youtube` Package

eWRT provides the following features:

- content acquisition components for the Web (see `eWRT.access.http`) and different social media sources (see `eWRT.ws`)
- low-level natural language processing, e.g.
  - language detection (`eWRT.stat.language`)
  -
- content caching (see `eWRT.util.cache`)



## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### e

- eWRT, [5](#)
- eWRT.access.file, [5](#)
- eWRT.config, [6](#)
- eWRT.input, [6](#)
- eWRT.input.clean, [6](#)
- eWRT.input.conv, [6](#)
- eWRT.input.conv.doc, [6](#)
- eWRT.input.conv.html, [7](#)
- eWRT.input.conv.pdf, [7](#)
- eWRT.input.corpus.bbc, [8](#)
- eWRT.lib, [8](#)
- eWRT.lib.apihelber, [9](#)
- eWRT.lib.Result, [8](#)
- eWRT.lib.ResultSet, [8](#)
- eWRT.lib.thirdparty.advas, [9](#)
- eWRT.lib.thirdparty.advas.phonetics, [9](#)
- eWRT.lib.Webservice, [9](#)
- eWRT.ontology, [9](#)
- eWRT.stat, [10](#)
- eWRT.stat.coherence, [11](#)
- eWRT.stat.eval.metrics, [11](#)
- eWRT.stat.language, [12](#)
- eWRT.util.advLogging, [13](#)
- eWRT.util.assert, [13](#)
- eWRT.util.async, [13](#)
- eWRT.util.cache, [14](#)
- eWRT.util.exception, [16](#)
- eWRT.util.execute, [16](#)
- eWRT.util.monitoring, [17](#)
- eWRT.util.pickleIterator, [17](#)
- eWRT.util.profile, [18](#)
- eWRT.util.timing, [18](#)
- eWRT.ws, [18](#)
- eWRT.ws.TagInfoService, [19](#)
- eWRT.ws.WebDataSource, [19](#)



## A

a (eWRT.stat.eval.metrics.TestEvaluationMetrics attribute), 12  
 AbstractIterableWebSource (class in eWRT.ws), 18  
 AbstractIterator (class in eWRT.util.pickleIterator), 17  
 AbstractWebSource (class in eWRT.ws), 19  
 addContent() (eWRT.lib.ResultSet.ResultSet method), 8  
 AssertReturnValue (class in eWRT.util.assert), 13  
 Async (class in eWRT.util.async), 13

## B

b (eWRT.stat.eval.metrics.TestEvaluationMetrics attribute), 12  
 BBCGetCorpus (class in eWRT.input.corpus.bbc), 8

## C

c (eWRT.stat.eval.metrics.TestEvaluationMetrics attribute), 12  
 Cache (class in eWRT.util.cache), 14  
 cache (eWRT.util.cache.DiskCached attribute), 15  
 caverphone() (in module eWRT.lib.thirdparty.advas.phonetics), 9  
 clear() (eWRT.util.timing.Timed method), 18  
 close() (eWRT.util.pickleIterator.AbstractIterator method), 17  
 Coherence (class in eWRT.stat.coherence), 11  
 CompressedFile (class in eWRT.access.file), 5  
 COMPRESSION\_EXT (eWRT.access.file.CompressedFile attribute), 5

## D

d (eWRT.stat.eval.metrics.TestEvaluationMetrics attribute), 12  
 DEFAULT\_COMMAND  
   (eWRT.ws.AbstractIterableWebSource attribute), 18  
 DEFAULT\_FORMAT (eWRT.ws.AbstractIterableWebSource attribute), 18

## DEFAULT\_MAX\_RESULTS

(eWRT.ws.AbstractIterableWebSource attribute), 18

## DEFAULT\_START\_INDEX

(eWRT.ws.AbstractIterableWebSource attribute), 18

detect\_language() (in module eWRT.stat.language), 12  
 DetectLanguageTest (class in eWRT.stat.language), 12  
 DiceCoherence (class in eWRT.stat.coherence), 11  
 DiskCache (class in eWRT.util.cache), 14  
 DiskCached (class in eWRT.util.cache), 15  
 dump() (eWRT.util.pickleIterator.WritePickleIterator method), 17

## E

emit() (eWRT.util.advLogging.SNMPHandler method), 13  
 eWRT (module), 5  
 eWRT.access.file (module), 5  
 eWRT.config (module), 6  
 eWRT.input (module), 6  
 eWRT.input.clean (module), 6  
 eWRT.input.conv (module), 6  
 eWRT.input.conv.doc (module), 6  
 eWRT.input.conv.html (module), 7  
 eWRT.input.conv.pdf (module), 7  
 eWRT.input.corpus.bbc (module), 8  
 eWRT.lib (module), 8  
 eWRT.lib.apihelber (module), 9  
 eWRT.lib.Result (module), 8  
 eWRT.lib.ResultSet (module), 8  
 eWRT.lib.thirdparty.advas (module), 9  
 eWRT.lib.thirdparty.advas.phonetics (module), 9  
 eWRT.lib.Webservice (module), 9  
 eWRT.ontology (module), 9  
 eWRT.stat (module), 10  
 eWRT.stat.coherence (module), 11  
 eWRT.stat.eval.metrics (module), 11  
 eWRT.stat.language (module), 12  
 eWRT.util.advLogging (module), 13

eWRT.util.assert (module), 13  
eWRT.util.async (module), 13  
eWRT.util.cache (module), 14  
eWRT.util.exception (module), 16  
eWRT.util.execute (module), 16  
eWRT.util.monitoring (module), 17  
eWRT.util.pickleIterator (module), 17  
eWRT.util.profile (module), 18  
eWRT.util.timing (module), 18  
eWRT.ws (module), 18  
eWRT.ws.TagInfoService (module), 19  
eWRT.ws.WebDataSource (module), 19

## F

fetch() (eWRT.util.async.Async method), 13  
fetch() (eWRT.util.cache.Cache method), 14  
fetch() (eWRT.util.cache.DiskCache method), 14  
fetch() (eWRT.util.cache.MemoryCache method), 15  
fetch() (eWRT.util.cache.RedisCache method), 16  
fetchObjectId() (eWRT.util.cache.Cache method), 14  
fetchObjectId() (eWRT.util.cache.DiskCache method), 15  
fetchObjectId() (eWRT.util.cache.IterableCache method), 15  
fetchObjectId() (eWRT.util.cache.MemoryCache method), 15  
fetchObjectId() (eWRT.util.cache.RedisCache method), 16  
fMeasure() (in module eWRT.stat.eval.metrics), 12

## G

garbage\_collect\_cache() (eWRT.util.cache.MemoryCache method), 15  
garbage\_collect\_cache() (eWRT.util.cache.RedisCache method), 16  
get\_extension\_list() (eWRT.access.file.CompressedFile class method), 5  
get\_filename() (eWRT.util.pickleIterator.AbstractIterator class method), 17  
get\_lang\_name() (in module eWRT.stat.language), 12  
get\_unique\_temp\_file() (in module eWRT.util.cache), 16  
getAttributes() (eWRT.lib.Result.Result method), 8  
getCacheStatistics() (eWRT.util.cache.DiskCache method), 15  
getCoherence() (eWRT.stat.coherence.Coherence static method), 11  
getCoherence() (eWRT.stat.coherence.DiceCoherence static method), 11  
getCoherence() (eWRT.stat.coherence.PMICoherence static method), 11  
getId() (eWRT.lib.Result.Result method), 8  
getId() (eWRT.lib.ResultSet.ResultSet method), 8  
getKey() (eWRT.util.cache.Cache static method), 14  
getName() (eWRT.lib.Result.Result method), 8  
getName() (eWRT.lib.ResultSet.ResultSet method), 8

getObjectId() (eWRT.util.cache.Cache static method), 14  
getPostHashfile() (eWRT.util.async.Async method), 13  
getRelatedTags() (eWRT.ws.TagInfoService.TagInfoService method), 19  
getTagInfo() (eWRT.ws.TagInfoService.TagInfoService method), 19  
getTermCoherence() (eWRT.stat.coherence.Coherence method), 11  
getText() (eWRT.input.conv.doc.HtmlToText static method), 7  
getText() (eWRT.input.conv.html.HtmlToText static method), 7  
getText() (eWRT.input.conv.pdf.HtmlToText static method), 7  
getTitle() (eWRT.input.corpus.bbc.BBCGetCorpus static method), 8

## H

has\_processes\_limit\_reached() (eWRT.util.async.Async method), 13  
HtmlToText (class in eWRT.input.conv.doc), 6  
HtmlToText (class in eWRT.input.conv.html), 7  
HtmlToText (class in eWRT.input.conv.pdf), 7

## I

info() (in module eWRT.lib.apihelper), 9  
invoke\_iterator() (eWRT.ws.AbstractIterableWebSource method), 18  
IterableCache (class in eWRT.util.cache), 15

## L

login() (eWRT.lib.WebService.WebService method), 9

## M

MAPPING (eWRT.ws.AbstractWebSource attribute), 19  
max\_cache\_size (eWRT.util.cache.MemoryCache attribute), 16  
MemoryCache (class in eWRT.util.cache), 15  
MemoryCached (class in eWRT.util.cache), 16  
metaphone() (in module eWRT.lib.thirdparty.advas.phonetics), 9

## N

NAME (eWRT.ws.AbstractWebSource attribute), 19  
next() (eWRT.input.corpus.bbc.BBCGetCorpus method), 8  
next() (eWRT.lib.ResultSet.ResultSet method), 8  
next() (eWRT.util.cache.IterableCache method), 15  
next() (eWRT.util.pickleIterator.AbstractIterator method), 17  
NSCA (class in eWRT.util.monitoring), 17  
nysiis() (in module eWRT.lib.thirdparty.advas.phonetics), 9

## O

open() (eWRT.access.file.CompressedFile class method), 5

## P

Performance (class in eWRT.util.monitoring), 17  
 pipe\_content() (in module eWRT.util.execute), 16  
 PMICoherence (class in eWRT.stat.coherence), 11  
 post() (eWRT.util.async.Async method), 14  
 post\_search() (eWRT.ws.AbstractWebSource method), 19  
 pre\_search() (eWRT.ws.AbstractWebSource method), 19  
 precision() (in module eWRT.stat.eval.metrics), 12  
 printRS() (eWRT.lib.ResultSet.ResultSet static method), 8  
 process\_output() (eWRT.ws.AbstractIterableWebSource method), 18  
 profile() (in module eWRT.util.profile), 18  
 profileFunction() (in module eWRT.util.profile), 18

## R

read\_wordlist() (in module eWRT.stat.language), 12  
 ReadPickleIterator (class in eWRT.util.pickleIterator), 17  
 recall() (in module eWRT.stat.eval.metrics), 12  
 RedisCache (class in eWRT.util.cache), 16  
 RedisCached (class in eWRT.util.cache), 16  
 refresh() (eWRT.lib.ResultSet.ResultSet method), 8  
 request() (eWRT.ws.AbstractIterableWebSource method), 19  
 Result (class in eWRT.lib.Result), 8  
 RESULT\_PATH (eWRT.ws.AbstractIterableWebSource attribute), 18  
 ResultSet (class in eWRT.lib.ResultSet), 8

## S

search() (eWRT.ws.WebDataSource.WebDataSource method), 19  
 search\_documents() (eWRT.ws.AbstractIterableWebSource method), 19  
 search\_documents() (eWRT.ws.AbstractWebSource method), 19  
 send() (eWRT.util.monitoring.NSCA static method), 17  
 sendSNMPTrap() (in module eWRT.util.advLogging), 13  
 setContent() (eWRT.lib.ResultSet.ResultSet method), 8  
 setUp() (eWRT.util.async.TestAsync method), 14  
 SNMPException, 16  
 SNMPHandler (class in eWRT.util.advLogging), 13  
 soundex() (in module eWRT.lib.thirdparty.advas.phonetics), 9  
 SUPPORTED\_PARAMS (eWRT.ws.AbstractWebSource attribute), 19

## T

TagInfoService (class in eWRT.ws.TagInfoService), 19

tearDown() (eWRT.util.async.TestAsync method), 14  
 TEST\_CACHE\_DIR (eWRT.util.async.TestAsync attribute), 14  
 test\_detect\_language() (eWRT.stat.language.DetectLanguageTest method), 12  
 test\_exceptions() (eWRT.stat.language.DetectLanguageTest method), 12  
 test\_formats() (eWRT.access.file.TestFile method), 6  
 test\_pipe\_content() (eWRT.util.execute.TestExecute method), 16  
 testAssertCounter() (eWRT.util.assert.TestAssertReturnValue method), 13  
 TestAssertReturnValue (class in eWRT.util.assert), 13  
 TestAsync (class in eWRT.util.async), 14  
 testBorderCases() (eWRT.input.conv.html.TestHtmlToText method), 7  
 testCallStatistics() (eWRT.util.timing.TimedTest method), 18  
 testClearCallStatistics() (eWRT.util.timing.TimedTest method), 18  
 TestCoherence (class in eWRT.stat.coherence), 11  
 testConversion() (eWRT.input.conv.html.TestHtmlToText method), 7  
 testDebugMode() (eWRT.util.async.TestAsync method), 14  
 testDice() (eWRT.stat.coherence.TestCoherence method), 11  
 TestEvaluationMetrics (class in eWRT.stat.eval.metrics), 11  
 TestExecute (class in eWRT.util.execute), 16  
 TestFile (class in eWRT.access.file), 6  
 testFMeasure() (eWRT.stat.eval.metrics.TestEvaluationMetrics method), 12  
 TestHandler (class in eWRT.util.advLogging), 13  
 testHandler() (eWRT.util.advLogging.TestHandler method), 13  
 TestHtmlToText (class in eWRT.input.conv.html), 7  
 testMaxProcessLimit() (eWRT.util.async.TestAsync method), 14  
 testPMI() (eWRT.stat.coherence.TestCoherence method), 11  
 testPMIZero() (eWRT.stat.coherence.TestCoherence method), 11  
 testPrecision() (eWRT.stat.eval.metrics.TestEvaluationMetrics method), 12  
 testProfile() (in module eWRT.util.profile), 18  
 testQuoting() (eWRT.util.exception.TestSNMPException method), 16  
 testRecall() (eWRT.stat.eval.metrics.TestEvaluationMetrics method), 12  
 testReturnValue() (eWRT.util.timing.TimedTest method), 18  
 TestSNMPException (class in eWRT.util.exception), 16  
 testSNMPException() (eWRT.util.exception.TestSNMPException

method), [16](#)

TESTSTRING (eWRT.access.file.TestFile attribute), [6](#)

Timed (class in eWRT.util.timing), [18](#)

TimedTest (class in eWRT.util.timing), [18](#)

## W

WebDataSource (class in eWRT.ws.WebDataSource), [19](#)

Webservice (class in eWRT.lib.Webservice), [9](#)

WritePickleIterator (class in eWRT.util.pickleIterator), [17](#)