Swish Indexing

Introduction

The swish extension provides the bindings for Swish-e API. Swish-e stands for "Simple Web Indexing System for Humans - Enhanced" and is an open source system for indexing and search. Swish-e itself is licensed under GPL license, but uses a clause that allows applications to link against the library if every copy of the combined work is accompanied by the URL to Swish-e source code. Here it is: » http://swish-e.org.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension?including the names of its functions and any other documentation surrounding this extension?may change without notice in a future release of PHP. This extension should be used at your own risk.

Installing/Configuring

Requirements

PECL/swish requires PHP 5.1.3 or newer.

Installation

Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases, downloads, source files, maintainer information, and a CHANGELOG, can be located here: » http://pecl.php.net/package/swish.

The latest PECL/swish Win32 DLL can be downloaded here: » php_swish.dll.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Swish::META_TYPE_UNDEF (integer)

Swish::META_TYPE_STRING (integer)

Swish::META_TYPE_ULONG (integer)

Swish::META_TYPE_DATE (integer)

Swish::IN_FILE_BIT (integer)

Swish::IN_TITLE_BIT (integer)

Swish::IN_HEAD_BIT (integer)

Swish::IN_BODY_BIT (integer)

Swish::IN_COMMENTS_BIT (integer)

Swish::IN_HEADER_BIT (integer)

Swish::IN_EMPHASIZED_BIT (integer)

Swish::IN_META_BIT (integer)

Swish::IN_FILE (integer)

Swish::IN_TITLE (integer)

Swish::IN_HEAD (integer)

Swish::IN_BODY (integer)

Swish::IN_COMMENTS (integer)

Swish::IN_HEADER (integer)

Swish::IN_EMPHASIZED (integer)

Swish::IN_META (integer)

Swish::IN_ALL (integer)

Examples

Basic usage

Example #1 - Basic search query

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $results = $swish->query("test OR text");

   echo "Found ", $results->hits, " results\n";
   while ($result = $results->nextResult()) {
       var_dump($result);
       break; //break after the first result
   }

} catch (SwishException $e) {
   echo "Error: ", $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
Found 9 results
object(SwishResult)#3 (8) {
 ["swishreccount"]=>
 int(1)
 ["swishrank"]=>
 int(1000)
 ["swishfilenum"]=>
 int(10)
 ["swishdbfile"]=>
 string(13) "index.swish-e"
 ["swishdocpath"]=>
 string(23) "README.SUBMITTING_PATCH"
 ["swishtitle"]=>
 NULL
 ["swishdocsize"]=>
 int(4557)
 ["swishlastmodified"]=>
 int(1072136752)
}
```

Swish Functions

Swish::__construct

Swish::__construct -- Construct a Swish object

Description

void **Swish::__construct** (string `$index_names`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`index_names`
The list of index files separated by spaces.

Return Values

No value is returned.

Errors/Exceptions

Throws SwishException on error.

Examples

Example #2 - A Swish::__construct() example

```
<?php

try {
   $swish = new Swish("index1 index2");
} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

foreach ($swish->indexes as $index) {
   var_dump($index["name"]);
   var_dump($index["headers"]["Total Words"]);
}
```

```
?>
```

The above example will output something similar to:

```
string(6) "index1"
int(1888)
string(6) "index2"
int(2429)
```

Swish->getMetaList

Swish->getMetaList -- Get the list of meta entries for the index

Description

array **Swish->getMetaList** (string `$index_name`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`index_name`
The name of the index file.

Return Values

Returns an array of meta entries for the given index.

Examples

Example #3 - Basic Swish->getMetaList() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   var_dump($swish->getMetaList("index.swish-e"));

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
array(1) {
 [0]=>
 array(3) {
```

```
    ["Name"]=>
    string(12) "swishdefault"
    ["Type"]=>
    int(0)
    ["ID"]=>
    int(1)
  }
}
```

Swish->getPropertyList

Swish->getPropertyList -- Get the list of properties for the index

Description

array **Swish->getPropertyList** (string `$index_name`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`index_name`
The name of the index file.

Return Values

Returns an array of properties for the given index.

Examples

Example #4 - Basic Swish->getPropertyList() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $properties = $swish->getPropertyList("index.swish-e");
   foreach ($properties as $prop) {
       echo $prop["Name"],"\n";
   }

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
swishreccount
swishrank
swishfilenum
swishdbfile
swishdocpath
swishtitle
swishdocsize
swishlastmodified
```

Swish->prepare

Swish->prepare -- Prepare a search query

Description

object **Swish->prepare** ([string `$query`])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Prepare and return a search object, which you can later use for unlimited number of queries.

Parameters

query
: Optional query string. The query can be also set using SwishSearch->execute() method.

Return Values

Returns SwishSearch object.

Errors/Exceptions

Throws SwishException on error.

Examples

Example #5 - Basic Swish->prepare() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare("search query");
   $results = $search->execute();
   echo "Found: ", $results->hits, " hits\n";
```

```
   $results = $search->execute("new search");

   echo "Found: ", $results->hits, " hits\n";
} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
Found: 2 hits
Found: 5 hits
```

Swish->query

Swish->query -- Execute a query and return results object

Description

object **Swish->query** (string `$query`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

A quick method to execute a search with default parameters.

Parameters

`query`
Query string.

Return Values

Returns SwishResults object.

Errors/Exceptions

Throws SwishException on error.

Examples

Example #6 - Basic Swish->query() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $results = $swish->query("test query");

   echo "Found: ", $results->hits, " hits\n";

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
```

```
}

?>
```

The above example will output something similar to:

```
Found: 1 hits
```

SwishResult->getMetaList

SwishResult->getMetaList -- Get a list of meta entries

Description

array **SwishResult->getMetaList** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Return Values

Returns the same array as swish->getmetalist(), but uses the index file from the result handle.

SwishResult->stem

SwishResult->stem -- Stems the given word

Description

array **SwishResult->stem** (string `$word`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Stems the word based on the fuzzy mode used during indexing. Each result object is linked with its index, so the results are based on this index.

Parameters

word
The word to stem.

Return Values

Returns array containing the stemmed word variants (usually just one).

Errors/Exceptions

Throws SwishException on error.

Examples

Example #7 - Basic SwishResult->stem() example

```
<?php

try {

   $swish = new Swish("ext/swish/tests/index.swish-e");
   $results = $swish->query("testing OR others");

   if ($result = $results->nextResult()) {
       var_dump($result->stem("testing")); //the results fully depend on the
stemmer used in the index
```

```
        var_dump($result->stem("others"));
    }

} catch (SwishException $e) {
    echo "Error: ", $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
array(1) {
 [0]=>
 string(4) "test"
}
array(1) {
 [0]=>
 string(5) "other"
}
```

SwishResults->getParsedWords

SwishResults->getParsedWords -- Get an array of parsed words

Description

array **SwishResults->getParsedWords** (string `$index_name`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`indexi_name`
The name of the index used to initialize Swish object.

Return Values

An array of parsed words with stopwords removed. The list of parsed words may be useful for highlighting search terms in the results.

Examples

Example #8 - Basic SwishResults->getParsedWords() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $results = $swish->query("'some characters' and numbers");

   var_dump($results->getParsedWords("index.swish-e"));
   var_dump($results->indexes[0]['parsed_words']); //same result in a
different way

} catch (SwishException $e) {
   echo "Error: ", $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
array(4) {
 [0]=>
 string(4) "some"
 [1]=>
 string(10) "characters"
 [2]=>
 string(3) "and"
 [3]=>
 string(7) "numbers"
}
array(4) {
 [0]=>
 string(4) "some"
 [1]=>
 string(10) "characters"
 [2]=>
 string(3) "and"
 [3]=>
 string(7) "numbers"
}
```

SwishResults->getRemovedStopwords

SwishResults->getRemovedStopwords -- Get an array of stopwords removed from the query

Description

array **SwishResults->getRemovedStopwords** (string `$index_name`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`index_name`
The name of the index used to initialize Swish object.

Return Values

Returns array of stopwords removed from the query.

SwishResults->nextResult

SwishResults->nextResult -- Get the next search result

Description

object **SwishResults->nextResult** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Return Values

Returns the next SwishResult object in the result set or **FALSE** if there are no more results available.

Examples

Example #9 - Basic SwishResults->nextResult() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare();

   $results = $search->execute("lost");
   while($result = $results->nextResult()) {
       /* do something with the result object */
   }

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

SwishResults->seekResult

SwishResults->seekResult -- Set current seek pointer to the given position

Description

int **SwishResults->seekResult** (int `$position`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`position`
Zero-based position number. Cannot be less than zero.

Return Values

Returns new position value on success.

Errors/Exceptions

Throws SwishException on error.

Examples

Example #10 - Basic SwishResults->seekResult() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare();

   $results = $search->execute("lost");

   var_dump($results->seekResult(0)); //this will succeed
   var_dump($results->seekResult(100)); //this will fail

} catch (SwishException $e) {
```

```
   echo "Error: ", $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
int(0)
Error: No more results
```

SwishSearch->execute

SwishSearch->execute -- Execute the search and get the results

Description

object **SwishSearch->execute** ([string `$query`])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Searches the index file(s) based on the parameters set in the search object.

Parameters

`query`
: The query string is an optional parameter, it can be also set using Swish->prepare() method. The query string is preserved between executions, so you can set it once, but execute the search multiple times.

Return Values

Returns SwishResults object.

Errors/Exceptions

Throws SwishException on error.

Examples

Example #11 - Basic SwishSearch->execute() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare();

   $results = $search->execute("query");
```

```
   echo "First query found: ", $results->hits, " hits\n";

   $results = $search->execute("new OR query");
   echo "Second query found: ", $results->hits, " hits\n";

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
First query found: 2 hits
Second query found: 12 hits
```

SwishSearch->resetLimit

SwishSearch->resetLimit -- Reset the search limits

Description

void **SwishSearch->resetLimit** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Reset the search limits previous set by SwishSearch->setLimit.

Return Values

No value is returned.

Examples

Example #12 - Basic SwishSearch->resetLimit() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare();

   $results = $search->execute("time");
   echo "First query found: ", $results->hits, " hits\n";

   $search->setLimit("swishdocsize", "3000", "6000"); //limit by document
size, from 3000 to 6000 bytes
   $results = $search->execute("time");
   echo "Second query found: ", $results->hits, " hits\n";

   $search->resetLimit();
   $results = $search->execute("time");
   echo "Third query found: ", $results->hits, " hits\n";

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
First query found: 5 hits
Second query found: 2 hits
Third query found: 5 hits
```

SwishSearch->setLimit

SwishSearch->setLimit -- Set the search limits

Description

void **SwishSearch->setLimit** (string `$property`, string `$low`, string `$high`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`property`
Search result property name.

`low`
The lowest value of the property.

`high`
The highest value of the property.

Return Values

No value is returned.

Errors/Exceptions

Throws SwishException on error.

Examples

Example #13 - Basic SwishSearch->setLimit() example

```
<?php
try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare();
```

```
   $results = $search->execute("time");
   echo "First query found: ", $results->hits, " hits\n";

   $i = 0;
   while($result = $results->nextResult()) {
       echo "Hit #", ++$i, " - ", $result->swishdocsize, " bytes\n";
   }

   $search->setLimit("swishdocsize", "3000", "6000"); //limit by document
size, from 3000 to 6000 bytes
   $results = $search->execute("time");
   echo "Second query found: ", $results->hits, " hits\n";

   $i = 0;
   while($result = $results->nextResult()) {
       echo "Hit #", ++$i, " - ", $result->swishdocsize, " bytes\n";
   }

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
First query found: 5 hits
Hit #1 - 4261 bytes
Hit #2 - 37937 bytes
Hit #3 - 7126 bytes
Hit #4 - 15427 bytes
Hit #5 - 4768 bytes
Second query found: 2 hits
Hit #1 - 4261 bytes
Hit #2 - 4768 bytes
```

SwishSearch->setPhraseDelimiter

SwishSearch->setPhraseDelimiter -- Set the phrase delimiter

Description

void **SwishSearch->setPhraseDelimiter** (string `$delimiter`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`delimiter`
Phrase delimiter character. The default delimiter is double-quotes.

Return Values

No value is returned.

Examples

Example #14 - Basic SwishSearch->setPhraseDelimiter() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare();

   $results = $search->execute('"every time"'); //looking for "every time"
   echo "First query found: ", $results->hits, " hits\n";

   $search->setPhraseDelimiter("'");
   $results = $search->execute("'every time'"); //the same query, but using
different delimiter
   echo "Second query found: ", $results->hits, " hits\n";

   $search->setPhraseDelimiter('"');
   $results = $search->execute("'every time'"); //looking for "every" and
"time"
```

```
   echo "Third query found: ", $results->hits, " hits\n";

   //let's look at parsed words
   var_dump($results->getParsedWords("index.swish-e"));

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
First query found: 1 hits
Second query found: 1 hits
Third query found: 2 hits
array(2) {
 [0]=>
 string(5) "every"
 [1]=>
 string(4) "time"
}
```

SwishSearch->setSort

SwishSearch->setSort -- Set the sort order

Description

void **SwishSearch->setSort** (string `$sort`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`sort`
: Sort order of the results is a string containing name of a result property combined with sort direction ("asc" or "desc"). Examples: "swishrank desc", "swishdocpath asc", "swishtitle asc", "swishdocsize desc", "swishlastmodified desc" etc.

Return Values

No value is returned.

Examples

Example #15 - Basic SwishSearch->setSort() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare();

   $results = $search->execute("time");
   echo "First query found: ", $results->hits, " hits\n";

   $i = 0;
   while($result = $results->nextResult()) {
       echo "Hit #", ++$i, " - ", $result->swishdocsize, " bytes\n";
   }

   $search->setSort("swishdocsize desc"); //sort by document size
```

```
   $results = $search->execute("time");
   echo "Second query found: ", $results->hits, " hits\n";

   $i = 0;
   while($result = $results->nextResult()) {
       echo "Hit #", ++$i, " - ", $result->swishdocsize, " bytes\n";
   }

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
First query found: 5 hits
Hit #1 - 4261 bytes
Hit #2 - 37937 bytes
Hit #3 - 7126 bytes
Hit #4 - 15427 bytes
Hit #5 - 4768 bytes
Second query found: 5 hits
Hit #1 - 37937 bytes
Hit #2 - 15427 bytes
Hit #3 - 7126 bytes
Hit #4 - 4768 bytes
Hit #5 - 4261 bytes
```

SwishSearch->setStructure

SwishSearch->setStructure -- Set the structure flag in the search object

Description

void **SwishSearch->setStructure** (int `$structure`)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, its name, and surrounding documentation may change without notice in a future release of PHP. This function should be used at your own risk.

Parameters

`structure`
: The structure flag a bitmask is used to limit search to certain parts of HTML documents (like title, meta, body etc.). Its possible values are listed below. To combine several values use bitwise OR operator, see example below.

- **Swish::IN_FILE**
- **Swish::IN_TITLE**
- **Swish::IN_HEAD**
- **Swish::IN_BODY**
- **Swish::IN_COMMENTS**
- **Swish::IN_HEADER**
- **Swish::IN_EMPHASIZED**
- **Swish::IN_META**

Return Values

No value is returned.

Examples

Example #16 - Basic SwishSearch->setStructure() example

```
<?php

try {

   $swish = new Swish("index.swish-e");
   $search = $swish->prepare();

   $results = $search->execute("time");
   echo "First query found: ", $results->hits, " hits\n";

   $search->setStructure(Swish::IN_TITLE|Swish::IN_HEAD); //search in title
and head
   $results = $search->execute("time");
   echo "Second query found: ", $results->hits, " hits\n";

   $search->setStructure(Swish::IN_ALL); //search in whole document, the
default value
   $results = $search->execute("time");
   echo "Third query found: ", $results->hits, " hits\n";

} catch (SwishException $e) {
   echo $e->getMessage(), "\n";
}

?>
```

The above example will output something similar to:

```
First query found: 5 hits
Second query found: 0 hits
Third query found: 5 hits
```