

Options

1.1

Generated by Doxygen 1.5.8

Tue Feb 16 00:54:58 2010

Contents

1	Options	1
1.1	Introduction	1
1.2	Author	2
1.3	Bugtracking	2
1.4	Dependencies	2
1.5	Installation	2
1.6	Error handling	2
1.7	Example	3
2	Class Index	5
2.1	Class List	5
3	Class Documentation	7
3.1	Options Class Reference	7
3.1.1	Detailed Description	7
3.1.2	Member Function Documentation	8
3.1.2.1	delWhitespaces	8
3.1.2.2	getCompleteConfig	8
3.1.2.3	getProperty	8
3.1.2.4	new	8
3.1.2.5	splitCfgLine	9

Chapter 1

Options

1.1 Introduction

`Options` is a class giving you the ability to easily parse optionfiles in your perl programmms.

With `Options` you can implement a hierarchical option structure which this class can parse into a single anon perl hash structure or you can extrace single options.

To implement a hierarchic structure you must seperate options attributes using a ".".

The main syntax of an optionline is:

```
attribute[.attribute[.]] = value
```

Blank lines or lines starting with a # (hash-sign) will be ignored during parsing.

Lets construct an example optionfile for configuring cars and their attributes.

We call our optionfile "cars.properties" and it looks like this:

```
audi.tt.doors = 2
audi.tt.ps = 250
audi.a4.doors = 4
audi.a4.ps = 130
porsche.cayenne.doors = 4
porsche.boxter.doors = 2
```

Now you got two options.

- Parse a single option value
- Parse whole optionsfile into a anon perl hash

Lets consider you want to parse a single value: audi.a4.ps

```
my $psAudiA4 = $caropts->getProperty("audi.a4.ps");
```

`$psAudiA4` will now hold the value "130".

Lets consider you want to parse the whole option file.

```
my $caroptsStruct = $caropts->getCompleteConfig();
```

No \$caroptsStruct will have the following layout:

```
$caroptsStruct->{audi}->{tt}->{doors} = 2
$caroptsStruct->{audi}->{tt}->{ps} = 250
$caroptsStruct->{audi}->{a4}->{doors} = 4
$caroptsStruct->{audi}->{a4}->{ps} = 130
$caroptsStruct->{porsche}->{cayenne}->{doors} = 4
$caroptsStruct->{porsche}->{boxter}->{doors} = 2
```

1.2 Author

Markus Mazurczak <coding@markus-mazurczak.de>

1.3 Bugtracking

If you find any bugs or want to have a particular feature included register at <http://mantis.markus-mazurczak.de>

1.4 Dependencies

- strict - Should be a core module
- warnings - Should be a core module
- File::Basename - Should be a core module
- File::Spec - Should be a core module
- Merror - Download it from <http://markus-mazurczak.de/?p=40>

1.5 Installation

Download and unpack [Options](#) into a directory your perl programm can access and also put Merror.pm into the same directory as Options.pm

1.6 Error handling

[Options](#) uses the Merror class to implement error handling with a stacktrace function. The Merror object will be accessible through the MERROR member of the [Options](#) object.

E.g.: \$sopts->{MERROR}->error();

1.7 Example

Lets create the following directory structure: \$HOME/src/test

- lib/
 - Merror.pm
 - Options.pm
- properties/
 - cars.properties
- test.pl

And test.pl looks the following:

```
#!/usr/bin/perl
BEGIN {
    use strict;
    use warnings;
    use File::Spec;
    use File::Basename;
    use Data::Dumper;

    push(@INC, File::Spec->rel2abs(dirname($0)."/lib");
}

use Merror;
use Options;

my $optfile = File::Spec->rel2abs(dirname($0)."/properties/cars.properties");

my $options = Options->new($optfile);
if($options->{MERROR}->error()) {
    print("Error code: " . $options->{MERROR}->ec . "\n");
    print("Error description: " . $options->{MERROR}->et . "\n");
    print("Stacktrace:\n");
    $options->{MERROR}->stacktrace;
    exit(0);
}

my $fullopts = $options->getCompleteConfig();

print Dumper($fullopts);

my $audiA4Doors = $options->getProperty("audi.a4.doors");

print("Audi A4 doors: $audiA4Doors\n");
```

The output will be the following:

```
nexus@hoare: ~/src/perl/test$ ./test.pl
$VAR1 = {
    'porsche' => {
        'boxter' => {
            'doors' => '2'
        },
        'cayenne' => {
            'doors' => '4'
        }
    },
    'audi' => {
        'a4' => {
```

```
        'doors' => '4',
        'ps' => '130'
    },
    'tt' => {
        'doors' => '2',
        'ps' => '250'
    }
}
};
Audi A4 doors: 4
```

Now lets assume that we got no read permissions on cars.properties
than the output will look like the following:

```
Error code: -1
Error description: Could not read option file: /home/nexus/src/perl/test/properties/cars.properties
Stacktrace:
Level: 0 -- File: /home/nexus/src/perl/test/lib/Merror.pm -- Pkg: Merror -- Sub: Merror::fillstack -- Line:
Level: 1 -- File: /home/nexus/src/perl/test/lib/Options.pm -- Pkg: Options -- Sub: Merror::error -- Line:
Level: 2 -- File: /home/nexus/src/perl/test/lib/Options.pm -- Pkg: Options -- Sub: Options::checkOptfile --
Level: 3 -- File: ./test.pl -- Pkg: main -- Sub: Options::new -- Line: 17
```

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Options (Options implements a class for parsing option files)	7
---	---

Chapter 3

Class Documentation

3.1 Options Class Reference

[Options](#) implements a class for parsing option files.

Public Functions

- public [Options](#) [new](#) (string optfile)
Creates a new [Options](#) object checking optfile.
- public anonHash [getCompleteConfig](#) ()
Parses the whole configuration file and returns it as a anonymous perl hash.
- public string [getProperty](#) (string attributeName)
Searches attributeName in the configuration file and returns the value pre and post whitespace deleted or sets Merror.

Private Functions

- private void [splitCfgLine](#) (string line, anonHash hashRef)
Splits a configuration line referenced by line into a hierarchic anonHash hashRef.
- private string [delWhitespaces](#) (string what)
Deletes all leading and trailing whitespaces from what.
- private void [checkOptfile](#) ()
Checks if a optionfile is given and if it is readable.

3.1.1 Detailed Description

[Options](#) implements a class for parsing option files.

3.1.2 Member Function Documentation

3.1.2.1 private string Options::delWhitespaces (string *what*)

Deletes all leading and trainling whitespaces from what.

Parameters:

what String where to delete

Returns:

A leading and trainling whitespaces cleaned string

3.1.2.2 public anonHash Options::getCompleteConfig ()

Parses the whole configuration file and returns it as a anonymous perl hash.

Returns:

cfgHash A anonymous perl hash mapping the hierarchic options structure

3.1.2.3 public string Options::getProperty (string *attributeName*)

Searches attributeName in the configuration file and retunes the value pre and post whitespace deleted or sets Merror.

Parameters:

attributeName The attribute name

Returns:

The value of attribute referenced through attributeName or an empty string

3.1.2.4 public Options Options::new (string *optfile*)

Creates a new [Options](#) object checking optfile.

Parameters:

optfile String defining the path to the optionfile to use

Returns:

[Options](#) Returns a new [Options](#) object

3.1.2.5 private void Options::splitCfgLine (string *line*, anonHash *hashRef*)

Splits a configuration line referenced by *line* into a hierarchic anonHash *hashRef*.

Parameters:

line A string holding the actual configurationfile line

hashRef A hash reference where the structure will be mapped

The documentation for this class was generated from the following file:

- Options.pm

Index

delWhitespaces
Options, [8](#)

getCompleteConfig
Options, [8](#)

getProperty
Options, [8](#)

new
Options, [8](#)

Options, [7](#)
delWhitespaces, [8](#)
getCompleteConfig, [8](#)
getProperty, [8](#)
new, [8](#)
splitCfgLine, [8](#)

splitCfgLine
Options, [8](#)