PHP class Indentation

Author: Roger Baklund

Wed Feb 10 2016

Contents

1 Introduction

Generic text indentation utilities.

This class has methods for manipulating indentation in blocks of text. You can read current indentation without making changes, you can indent or unindent a block of text, you can split blocks of text based on the indentation, and you can modify indentation for individual lines in a text.

TAB characters are recognized, but they count as one, which makes it hard to see the correct level when used in combination with spaces. For this reason you should use either spaces or `TAB` characters, not both mixed.

The `$lines` argument can be a string with linefeeds (`LF` or `CRLF`) or an array of lines, such as the one returned from `file()`.

NOTE: `CR` characters are removed, the output strings will allways use `LF`.

Version

1.3 LGPL

Author

Roger Baklund `roger@baklund.no`

Version history:

- 1.3 2016-01-29
  - unindent() return empty string for no/false/NULL input
- 1.2 2014-03-19
  - $ws argument added for set_indents()
- 1.1 2014-03-16
  - Improved line handling
  - Improved documentation
- 1.0 2014-02-10 (initial version)

2 Indentation

A few methods for working with text indentation.

```
blocks($lines)                  -- Input multiline string or array of strings, returns
                                   array of blocks paired with line number reference
indent($lines,$size=2,$ws=' ') -- Add spaces at the start of each line, by default 2
unindent($lines)                -- Remove indentation equally from all lines until at
                                   least one line starts at position 1
get_indents($lines)             -- Get the number of leading WS characters for each line
                                   in a multiline string
set_indents($lines,$arr,$append=false,$ws=' ') -- Set individual indentation for each line
```

Each method is described in more detail below.

blocks()

Splits a string into blocks based on indentation.

```
function blocks($lines)
```

This method takes a single argument, a string or an array of strings, like output from `file()`. The string is split into lines, the indentation of each line is considered, and if it is indented compared to a previous line, it is considered to "belong" to that previous line, the two (or more) forms a "block". Empty lines are ignored. Note that the lines are *not* made into a single line, the `LF` is preserved, and the indentation is *not* removed. This means you can apply `blocks()` to it's own output to read multiple levels of indentation.

The `blocks()` method returns an array of arrays. The inner array has two items: the line number from the input source as an integer, and the block as a single string.

```
$blocks = Indentation::blocks(file('data.txt'));
$blocks = Indentation::blocks(file_get_contents('data.txt'));
$blocks = Indentation::blocks($LinesArray);
$blocks = Indentation::blocks($String);
```

Note that using constant multiline strings in the PHP source code requires zero indentation for the root items, otherwise it would be seen as a single multiline block. This results in ugly source code:

```
class foo {
  function bar() {
    if($somthing) {
      $blocks = Indentation::blocks(
        "First
           indented
Second
  indented
Third
  indented");
      foreach($blocks as $chunk) {
        list($LineNo,$block) = $chunk;
        Process($LineNo,$block);
      }
    }
  }
}
```

Use the `unindent()` method to avoid this problem:

```
$blocks = Indentation::blocks(
          Indentation::unindent(
   "First
      indented
    Second
      indented
    Third
      indented"));
foreach($blocks as $chunk) ...
```

indent()

Injects a number of WS characters at the start of each line in the string.

```
function indent($lines,$size=2,$ws=' ')
```

This method takes one, two or three arguments. The first is required, it is a multiline string or an array of lines. The second is optional, the number of whitespace characters to inject, the default value is 2. The third argument is also optional, it is which whitespace character to use, by default it will be a space. You could provide `"\\t"` or `chr(9)` if you wanted to inject TABs, or you could use `" "` for HTML output.

unindent()

Remove excessive indentation.

```
function unindent($lines)
```

This method takes a single argument, a multiline string or an array of lines, and removes excessive whitespace from the start of each line. The first line is a special case: any indentation is removed and ignored. The first line with characters is allways considered to be at the "root level" of the collection of "blocks".

What is excessive indentation?

If *all lines* in the input (ignoring the first) have `X` or more whitespace characters at the start of the line, then `X` spaces will be removed from all lines, so that at least one (the first) line will start at position one on the line.

For example, this:

```
Indentation::unindent("
          First
            indented
          Second
            indented");
# the lines above has 9 and 11 characters indentation
```

...results in this:

```
First
  idented
Second
  indented
```

Note that the structure is kept, the entire block of text is shifted to the left. "First" and "Second" has no indentation, lines two and four ("indented") are still indented, but now with only two space characters.

Sometimes the structure *is* changed, but only for the first line. Any leading whitespace will be removed and ignored when considering the indentation levels.

For example, this:

```
    First
  indented
Second
  indented
```

...is converted into this:

```
First
  indented
Second
  indented
```

The difference is only in the indentation for the first line, which is handled in this special way for practical reasons.

The `blocks()` method is used to split a string into blocks based on the indentation. There is a special case when there is only one such block.

Suppose you wanted Second and Third to be indented below First. This will not work:

```
Indentation::unindent('First
  Second
  Third');
Indentation::unindent(
  'First
   Second
   Third');
Indentation::unindent(
  'First
     Second
     Third');
```

The first two are identical, the linefeed *before* the ' does not matter. First, Second and Third are considered to be three items on the same level. The third example is less intuitive, the indentation is increased for Second and Third, but it is the same as the two above. All indentation for Second and Third is removed in all cases.

Even this next example is the same as the above, because the indentation for the first line is ignored:

```
Indentation::unindent('
    First
  Second
  Third');
```

If you wanted to keep the indentation in this case, you would have to leave the first line blank *and* Second and Third must have more indentation than First:

```
Indentation::unindent('
First
  Second
  Third');
Indentation::unindent('
    First
      Second
      Third');
```

In the last two examples, Second and Third are indentated and 'belongs' to First, only the indentation for First and the initial blank line is removed.

Note that the problem descibed above only occurs when there is a single item with indented lines, both of the below examples will see this as two blocks; First with Second and Third indented, and Fourth:

```
Indentation::unindent('First
  Second
  Third
Fourth');
Indentation::unindent('
  First
    Second
    Third
  Fourth');
```

Even the next example represents the same two blocks, because indentation for the first line is **ignored:**

```
Indentation::unindent('
      First
    Second
    Thirds
  Fourth');
```

get_indents()

Get indentation count for each line in a string with multipe lines.

```
function get_indents($lines)
```

This method takes a single argument; the text to analyze in the form of a multiline string or an array of lines. It will return an array of integers. Length of array corresponds to number of lines in the input, and each value represents the number of leading whitespace characters used for that line.

Note: TAB characters counts as one, just like space characters. If they are mixed the numbers does not reflect the indentation you can see in an editor.

Note: In addition to TAB and space characters, ASCII 0, 11 and 13 will also count as one.

set_indents()

Set indentation on individual lines in a multiline string.

```
function set_indents($lines,$arr,$append=false,$ws=' ')
```

This method takes two, three or four arguments. The first two are required. The first is a multiline string or an array of lines. This is the text with or without indentation. The second is an array of integers, each represents the amount of whitespace you wish to set or append for each line. The third, optional argument decides if the whitespace should be appended to existing whitespace in the input sting, or if they represent the final indentation you want in the result. The latter is the default, set it to `true` if you want it to append. The fourth, optional parameter is which whitespace character to use, by default it will use a space character (ASCII 32).

Examples

Example using `unindent()` and `blocks()`:

```
$str = "
      This is a test
        string spanning
        multiple lines.
      It contains three
        sentences, each are
        split on three lines.
      The indentation decides
        which lines belongs
          to which sentence.";
$str = Indentation::unindent($str);
echo '<pre>'.$str.'</pre>';
foreach(Indentation::blocks($str) as $idx => $block) {
  list($LineNo,$Sentence) = $block;
  echo '<p>'.
       'Sentence '.($idx+1).' from line '.$LineNo.': '.
       htmlentities($Sentence).
       '</p>';
}
```

This would output something like this:

```
This is a
  string spanning
  multiple lines.
It contains three
  sentences, each are
  split on three lines.
The indentation decides
  which lines belongs
    to which sentence.

Sentence 1 from line 1: This is a string spanning multiple lines.

Sentence 2 from line 4: It contains three sentences, each are split on three lines.

Sentence 3 from line 7: The indentation decides which lines belongs to which sentence.
```

You can use the `get_indents()` method to analyze the indentation of a multiline string.

```
$indents = Indentation::get_indents($String);
$lines = count($indents);
$smallest = min($indents);
$biggest = max($indents);
```

Empty lines in the input would be returned as 0 indented. you can not distinguish it from a line with text and no indentation without inspecting the line. Note that the `blocks()` method automatically removes empty lines, eliminating this problem.

3 Class Documentation

3.1 Indentation Class Reference

Static Public Member Functions

- static blocks ($lines)
- static unindent ($lines)
- static indent ($lines, $size=2, $ws=' ')
- static set_indents ($lines, $arr, $append=false, $ws=' ')
- static get_indents ($lines)

Static Private Member Functions

- static lines ($lines)

3.1.1 Detailed Description

Definition at line 38 of file Indentation.class.php.

3.1.2 Member Function Documentation

3.1.2.1 static Indentation::blocks (*$lines*) [static]

Concatinate indented line with the previous line.

Returns array of arrays with pairs of linenumbers and blocks.

NOTE: Will also remove `CR` and blank lines in the process, but these lines are still counted, the reported line numbers are correct according to the full input. You can detect removed blank lines by checking the linenumber plus the number of lines in a block and compare it with the linenumber in the next block.

Parameters

string | array	*$lines*	Multiline string or array of strings

Returns

array Array of arrays (Linenumber, Lines as one LF concatinated string)

Definition at line 66 of file Indentation.class.php.

```
                                {
    $blocks = array();
    $lineno = 0;
    foreach(self::lines($lines) as $line) {
      $lineno = $lineno + 1;
      if(!strlen(trim($line))) continue; # skip blank lines, but accept "0"
```

```
      if($blocks && in_array($line[0],array(' ',"\t"))) {
        $blocks[count($blocks)-1][1] .= "\n".rtrim($line); # append to last line
        continue;
      }
      $blocks[] = array($lineno,rtrim($line));
    }
    return $blocks;
  }
```

3.1.2.2 static Indentation::get_indents (*$lines*) [static]

Get the number of leading whitespace for each line.

Will count spaces, tabs and even occurrences of ASCII 0 and ASCII 11.

Parameters

string | array	*$lines*	A string or array of strings

Returns

array A list of integers, one for each line in the input

Definition at line 149 of file Indentation.class.php.

```
                                    {
    $res = array();
    foreach(self::lines($lines) as $l)
      $res[] = strlen($l) - strlen(ltrim($l));
    return $res;
  }
```

3.1.2.3 static Indentation::indent (*$lines*, *$size* = 2, *$ws* = ' ') [static]

Indent a multiline string by prepending each line with WS characters.

By default it will inject space characters, but you can provide `"\\t"` or `chr(9)` as the third parameter if you want to indent with TAB. There is however no special handling of TAB, each character counts as one.

Parameters

string | array	*$lines*	The string or array of strings to indent
int	*$size*	The size of the indentation, default is 2
string	*$ws*	The whitespace character to use, default space (ASCII 32)

Returns

string The multiline string indented

Definition at line 116 of file Indentation.class.php.

```
                                                {
    $lines = self::lines($lines);
    foreach($lines as $idx=>$l)
      $lines[$idx] = str_repeat($ws,$size).$l;
    return implode("\n",$lines);
  }
```

3.1.2.4 static Indentation::lines (*$lines*) [static],[private]

Normalize input to array of lines.

Accepts both CRLF and LF line endings. Will return an array unchanged.

This method is private, it used by all the other methods.

Parameters

string | array	*$lines*	Multiline string or array of strings

Returns

array Array of strings

Definition at line 48 of file Indentation.class.php.

```
                                        {
  if(is_string($lines))
    $lines = explode("\n",str_replace("\r\n","\n",$lines));
  return $lines;
}
```

3.1.2.5 static Indentation::set_indents (*$lines*, *$arr*, *$append* = `false`, *$ws* = `' '`) [static]

Set individual indentation for each line.

This method is similar to indent(), except you provide an array of int values specifying the indentation for each line in the multiline string. You can append to existing indentation, and you can provide the character to use, by default it will be a space.

Parameters

string | array	*$lines*	A string or array of strings you want to indent
array	*$arr*	Array of integers, specific indentation for each line
bool	*$append*	Set to true if you want to append the given numbers to existing indentation, be default it will replace existing indentation.
string	*$ws*	The whitespace character to use, default space (ASCII 32)

Returns

string The multiline string indented

Definition at line 136 of file Indentation.class.php.

```
                                                                {
  $lines = self::lines($lines);
  foreach($lines as $idx=>$l)
    $lines[$idx] = str_repeat($ws,$arr[$idx]).($append?$l:ltrim($l));
  return implode("\n",$lines);
}
```

3.1.2.6 static Indentation::unindent (*$lines*) [static]

Remove unwanted indentation.

Removes preceeding spaces equally from all lines until at least one line starts at position one.

For practical reasons, the first line is not considered, any whitespace at the beginning is removed.

Parameters

string | array	*$lines*	A string or array of strings

Returns

string The string unchanged or unindented, some leading spaces removed

Definition at line 91 of file Indentation.class.php.

```
                                {
    $lines = self::lines($lines);
    if(!$lines) return '';
    $first = array_shift($lines);
    $min = PHP_INT_MAX;
    foreach($lines as $l) { # find smallest indentation
      $ind = strlen($l) - strlen(ltrim($l));
      if($ind < $min)
        $min = $ind;
    }
    foreach($lines as $idx=>$l)
      $lines[$idx] = substr($l,$min);
    return trim($first."\n".implode("\n",$lines));
  }
```

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

- /www/Indentation/Indentation.class.php

4 File Documentation

4.1 /www/Indentation/Indentation.class.php File Reference

Classes

- class Indentation

4.2 /www/Indentation/README.md File Reference

Index