ClickBlocks\Utils\PHP\Tokenizer

General information

Inheritance	no
Child classes	no
Interfaces	Iterator
Source	Framework/utils/php/tokenizer.php

The class allows to iterate tokens of php-code. Also it contains several useful static methods for getting different information about tokens.

Public static methods

getTokenCount()

```
public static integer getTokenCount()
```

Returns the number of available PHP tokens.

isEqual()

```
public static boolean isEqual(mixed $token1, mixed $token2)
```

$token1	mixed	the first token for comparison.
$token2	mixed	the second token for comparison.

Returns TRUE if the both tokens (which can be returned by **token_get_all()**) are equal and FALSE otherwise.

isSemanticToken()

```
public static boolean isSemanticToken(mixed $token)
```

$token	mixed	the token info or token ID.

Returns TRUE if the given token is not a comment, whitespace character or an open/close tag. Otherwise, it returns FALSE. Example:

```
echo (int)Tokenizer::isSemanticToken(T_COMMENT);                          // shows 0
echo (int)Tokenizer::isSemanticToken([T_FUNCTION, 'file_get_contents', 13]); // shows 1
echo (int)Tokenizer::isSemanticToken(';');                                // shows 1
```

parse()

```
public static array parse(string $code)
```

$code	string	the PHP code for tokenization.

Parses string into tokens. Returns an array of token identifiers. Example:

```
print_r(Tokenizer::parse('$y = ($x = 1) + $x++ * 3;'));
```

The execution result of the above script will be as follows:

```
Array
(
    [0] => Array
        (
            [0] => 310
            [1] => $y
            [2] => 1
        )

    [1] => Array
        (
            [0] => 377
            [1] =>
            [2] => 1
        )

    [2] => =
    [3] => Array
        (
            [0] => 377
            [1] =>
            [2] => 1
        )

    [4] => (
    [5] => Array
        (
            [0] => 310
            [1] => $x
            [2] => 1
        )

    [6] => Array
        (
            [0] => 377
            [1] =>
            [2] => 1
        )

    [7] => =
    [8] => Array
        (
            [0] => 377
            [1] =>
            [2] => 1
        )

    [9] => Array
        (
            [0] => 306
            [1] => 1
            [2] => 1
        )

    [10] => )
    [11] => Array
        (
            [0] => 377
            [1] =>
            [2] => 1
        )

    [12] => +
    [13] => Array
```

```
        (
            [0] => 377
            [1] =>
            [2] => 1
        )

    [14] => Array
        (
            [0] => 310
            [1] => $x
            [2] => 1
        )

    [15] => Array
        (
            [0] => 298
            [1] => ++
            [2] => 1
        )

    [16] => Array
        (
            [0] => 377
            [1] =>
            [2] => 1
        )

    [17] => *
    [18] => Array
        (
            [0] => 377
            [1] =>
            [2] => 1
        )

    [19] => Array
        (
            [0] => 306
            [1] => 3
            [2] => 1
        )

    [20] => ;
)
```

Public non-static methods

__construct()

```
public void __construct(string $source)
```

$source	string	the PHP code for tokenization.

Constructor. Sets source PHP code to tokenize.

reset()

```
public void reset()
```

Sets internal pointer to the first character of source code. Example:

```
$code = '$y = ($x = 1) + $x++ * 3;';
```

```
$tz = new Tokenizer($code);
print_r($tz->token()); // gets the first token.
print_r($tz->token()); // gets the second token.
$tz->reset();          // returns back the initial state of the tokenizer.
print_r($tz->token()); // gets the first token.
print_r($tz->token()); // gets the second token.
```

token()

```
public mixed token()
```

Returns the current token and move forward to next one. Returns FALSE if the internal token pointer points beyond the end of the tokens list or the tokens list is empty. Example:

```
$code = '$y = ($x = 1) + $x++ * 3;';
$tz = new Tokenizer($code);
// Successively displays all tokens.
while ($token = $tz->token()) print_r($token);
```

current()

```
public mixed current()
```

Returns the current token. Returns FALSE if the internal pointer points beyond the end of the tokens list. It does not move the internal token pointer in any way. Example:

```
$code = '$y = ($x = 1) + $x++ * 3;';
$tz = new Tokenizer($code);
// Successively displays all tokens.
while ($tz->valid())
{
  // Shows the current token.
  print_r($tz->current());
  // Advances the internal token pointer.
  $tz->next();
}
```

key()

```
public integer key()
```

Return the position of the current token. Returns positive integer on success, or -1 on failure. Example:

```
$code = '$y = ($x = 1) + $x++ * 3;';
$tz = new Tokenizer($code);
// Successively displays all token indexes.
while ($tz->valid())
{
  // Shows index number of the current token.
  print_r($tz->key());
  // Advances the internal token pointer.
  $tz->next();
}
```

The above script will output:

```
01234567891011121314151617181920
```

valid()

```
public boolean valid()
```

Checks if current position is valid. Returns TRUE on success or FALSE on failure.

rewind()

```
public void rewind()
```

Rewinds the tokenizer to the first token. This is the first method called when starting a foreach loop. It will not be executed after foreach loops. Example:

```
$code = '$y = ($x = 1) + $x++ * 3;';
$tz = new Tokenizer($code);
// Successively displays all tokens.
foreach ($tz as $token) print_r($token);
```

next()

```
public void next()
```

Moves forward the internal token pointer to the next token.

Protected non-static properties

source

```
protected string $source
```

The PHP code for tokenization.

length

```
protected integer $length
```

Length of the given PHP code.

token

```
protected mixed $token
```

The current token.

pos

```
protected integer $pos = -1
```

Position of the current token.