

Dokumentacja kodu PHP

Spis treści

- 1 Dokumentacja kodu PHP - czyli phpDocumentor
 - 1.1 Dokumentacja kodu źródłowego - o co chodzi?
 - 1.1.1 Pamięć ludzka jest zawodna
 - 1.1.2 Dokumentacja dla innych
 - 1.2 Narzędzie phpDocumentor
 - 1.3 Podstawy phpDocumentor
 - 1.3.1 DocBlock
 - 1.3.2.1 @abstract
 - 1.3.2.2 @access
 - 1.3.2.3 @author
 - 1.3.2.4 @category
 - 1.3.2.5 @copyright
 - 1.3.2.6 @deprecated
 - 1.3.2.7 @final
 - 1.3.2.8 @global
 - 1.3.2.9 @ignore
 - 1.3.2.10 @license
 - 1.3.2.11 @link
 - 1.3.2.12 @package
 - 1.3.2.13 @param
 - 1.3.2.14 @return
 - 1.3.2.15 @since
 - 1.3.2.16 @static
 - 1.3.2.17 @staticvar
 - 1.3.2.18 @todo
 - 1.3.2.19 @var
 - 1.3.2.20 @version
 - 1.4 Używanie phpDocumentora
 - 1.5 Zakończenie

Dokumentacja kodu PHP - czyli phpDocumentor

Dokumentacja kodu źródłowego - o co chodzi?

Gdy stworzysz projekt sam dla siebie, ewentualnie dla niewielkiej grupy znajomych możesz myśleć, że prawidłowe opisanie i udokumentowanie wszystkich tworzonych funkcji i klas jest głupie. "To przecież tak dużo roboty!". Jest to oczywiście prawdą. Jest to bardzo czasochłonne zajęcie, ale może przynieść korzyści. A jest już niezmiernie ważne, gdy piszesz kod, który ma być dostępny szerszemu gronu osób.

Pamięć ludzka jest zawodna

Gdy do własnego, zagmatwanego kodu siądziesz po długim czasie, to bez porządných komentarzy nie rozeznasz się szybko do czego służy dana funkcja. Nie musisz pamiętać czy jako parametry przyjmuje tablice czy ciągi. A może wartości logiczne? A co zwraca? Liczby czy ciągi?

Udokumentowanie kodu, czy to w PHP, czy w jakimkolwiek innym języku, jest ważne, gdy sam, po jakimś czasie

musisz rozeznąć się w napisanym kodzie.

Ale jeśli sam nie rozumiesz swojego kodu, to pomyśl, jak inni sobie z tym nie radzą.

Dokumentacja dla innych

Gdy bierzesz udział w projekcie Open Source być może w regułach pisania kodu projektu są ustalone takie rzeczy jak nazywanie zmiennych, opisywanie ich komentarzami, zaznaczanie modyfikowanych fragmentów.

Gdy sam piszesz coś, co będziesz udostępniał szerszemu gronu osób jako projekt Open Source (czy poprzez sprzedaż kodu!), należy opisać także, czym zajmują się dane funkcje, klasy. Tak, by inni wiedzieli, jak daną funkcję wykorzystać. Większość osób, gdy sięga po jakiś kod Open Source, by wykorzystać go we własnym projekcie, ma nadzieję, że szybko się rozeznają, jak należy użyć danej funkcji. Jeśli kod będzie nieczytelny i nieopisany - poszukają czegoś innego. Zwłaszcza, gdy naglą ich terminy, nie będą chcieli tracić czasu na rozgryzanie Twoich funkcji.

Gdy piszesz kod, który kiedyś może być modyfikowany, bądź ulepszany, warto zadbać, aby inni nie męczyli się z ponownym rozgryzaniem funkcji. Jedną ze "złotych zasad", jakie znalazłem w Internecie, jest:

"Pisz kod tak, jakby osoba, która go po Tobie przejmie, była uzbrojonym psychopatą znającym Twój adres".

Nie utrudniaj innym pracy. Kod musi być komentowany i opisywany. A dokumentację kodu można sobie ułatwić, dzięki różnorodnym narzędziom.

Narzędzie phpDocumentor

Jak programiści Javy mają JavaDoc, jak Pascalowcy mają PasDoc, tak PHP-owcy mają phpDocumentor. Wszystkie te narzędzia oparte są na podobnych zasadach. Przed każdym elementem do opisania musi znaleźć się komentarz, w którym są specjalne znaczniki. Automatyczny program następnie analizuje kod oraz te znaczniki, tworząc dokumenty w języku HTML zawierające opis klas, plików, funkcji.

phpDocumentor jest narzędziem typu Open Source, udostępnianym na zasadach licencji GPL. Dostępny jest w serwisie SourceForge, pod adresem <http://phpdocu.sourceforge.net>.

Podstawy phpDocumentor

DocBlock

Wszystkie komentarze, jakie phpDocumentor ma uwzględnić, muszą znaleźć się w specjalnych znacznikach, nazywanych "DocBlock". Sekcje DocBlock (documentation block) wyglądają w ten sposób:

```
/**
 *
 */
```

Wszystkie linie nie zaczynające się od "/*" będą ignorowane.

Aby udokumentować funkcję, klasę, zmienną czy stałą, należy DocBlock umieścić przed nią. Na przykład w taki sposób:

```
/**
 * Przekracza szybkość światła i ratuje świat... przed śniadaniem!
 */
function foo()
```

```
{
...
}
```

DocBlock zawierają trzy elementy:

- * krótki opis
- * długi opis
- * znaczniki

```
/**
 * Funkcja, która jest genialna
 *
 * Funkcja, którą mógłbym długo opisywać, bo to jest pole na długi opis
 * ale jest tak zagmatwana, że mi się nie chce.
 *
 * TUTAJ ZNACZNIKI
 */
```

Znaczniki

Znaczniki (ang. *tags*) to krótkie słowa poprzedzane znakiem @. Informują one narzędzie, jak wyświetlać dokumentację. Mogą oznaczać zmienne jako statyczne, metody jako prywatne, oznaczać autora bądź licencjonowanie.

Przykład użycia znaczników:

```
/**
 * Ładowanie (lub tworzenie) pliku logu
 * @param string $file_name Ścieżka do pliku logu
 * @param string $mode Tryb otwarcia logu (append, write, read)
 * @return bool Zwraca czy operacja się powiodła
 * @author Ktos <ktos@ktos.info>
 * @license http://www.example.com/gnugpl2/ GNU GPL v2
 * @access public
 */
function load_log_file($file_name, $mode = 'write')
{
    ...
}
```

Następujące znaczniki phpDoc są dostępne:

@abstract

Oznacza metodę, zmienną lub klasę, która musi być ponownie zdefiniowana w klasie potomnej.

Uwaga! Prawidłowe tylko dla PHP4, PHP5 posiada słowo kluczowe abstract.

@access

Kontrola dostępu do elementu. Jeśli używamy @access private to domyślnie phpDoc nie dołączy tego elementu do dokumentacji.

Dozwolone wartości: public, private, protected

@author

Oznacza autora elementu. Jeśli dodamy adres e-mail w nawiasach trójkątnych (na przykład: @autor Joe Schmoe <jschmoe@example.com>) to phpDoc zamieni ten adres na klikalny link mailto.

@category

Znacznik kategorii używany jest do grupowania razem elementów @package rozproszonych po różnych plikach.

@copyright

Oznacza prawa autorskie do elementu.

@deprecated

Oznacza element przestarzały, który nie powinien być używany, jako, że może zostać usunięty w następnych wersjach.

@final

Oznacza element, który nie powinien być nadpisywany przez klasy potomne.

Uwaga! Prawidłowe tylko dla PHP4, PHP5 posiada słowo kluczowe final.

@global

Definiuje zmienną globalną.

Są dwie metody użycia:

```
/**  
 * @global typ_danych $nazwa_zmiennej  
 * @global typ_danych opis zmiennej  
 */
```

Typ danych musi być typem danych PHP albo mixed.

@ignore

Oznacza element ignorowany

@license

Wyświetla adres internetowy do dokumentu z licencją kodu.

Składnia: @license URL nazwa_licencji

@link

Tworzy hiperłącze do adresu internetowego.

Składnia:

@link URL

@link URL Tekst opisujący

Druga wersja utworzy link z opisem w postaci `Opis`.

@package

Grupuje razem klasy i funkcje.

@param

Oznacza parametr funkcji.

Składnia: @param typ_danych \$nazwa_parametru Opis

@return

Określa typ danych zwracanych przez funkcję.

Składnia: @return typ_danych Opis

@since

Oznacza, od której wersji projektu jest w niej dany element.

@static

Oznacza metodę klasy jako statyczną (dostęp z zewnątrz bez tworzenia instancji klasy).

@staticvar

Oznacza zmienną klasy jako statyczną.

Składnia: @staticvar typ_danych Opis

@todo

Opis czynności, jakie zostały zaplanowane dla danego elementu, ale jeszcze nie wykonane.

@var

Opisuje zmienną.

Składnia: @var typ_danych Opis

@version

Opisuje wersję danego elementu.

Dygresja: Osobiście zwykle używam w tym miejscu znacznika `$Revision$`, dzięki któremu serwer CVS automatycznie wpisuje w to miejsce aktualną wersję pliku.

Używanie phpDocumentora

Gdy już opisaliśmy wszystkie elementy naszego kodu, możemy powierzyć aplikacji zadanie stworzenia dokumentacji. Ja będę opisywał to na podstawie interfejsu przez przeglądarkę WWW - aby go użyć wystarczy skopiować pliki PHP aplikacji do folderu, który może wyświetlić nasz serwer WWW.

Gdy naszym oczom ukaże się strona główna aplikacji, na górze zauważamy pasek kilku kart. Tam dokonujemy konfiguracji programu.



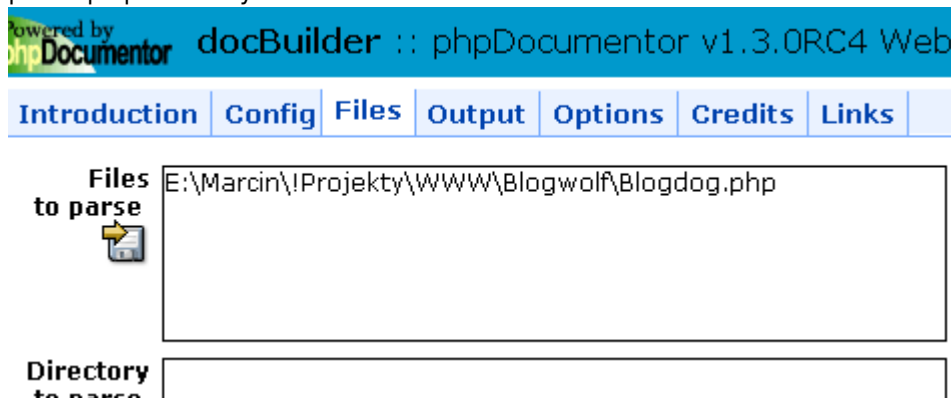
Welcome to **docBuilder**.

This is the new web-interface for running, in our opinion, the best in-cc **phpDocumentor**.

What's new in this release? Heaps of things, but here are the headlines

- Much greater support for PHP on both windows and Linux

Najważniejsza dla nas jest sekcja Files, gdzie podajemy pliki, jakie mają zostać poddane dokumentacji. Możemy też podać po prostu cały folder.




W sekcji Output podajemy katalog, gdzie ma znaleźć się dokumentacja. Możemy także wybrać styl, w jakim ma

zostać utworzona (jeśli zdecydujemy się na HTML), bądź też w ogóle format - od HTML do CHM przez DocBook.

Powered by
phpDocumentor docBuilder :: phpDocumentor v1.3.0

Introduction	Config	Files	Output	Options	Credits
--------------	--------	-------	--------	---------	---------

Target  E:\Marcin\!Projekty\WWW\Blogwolf\docs\

Output Format HTML:Smarty:default


Output type:Converter name:template name
HTML:Smarty:PHP

Po wybraniu odpowiedniego stylu/formatu zobaczymy przykład wyglądu.

Powered by
phpDocumentor docBuilder :: phpDocumentor v1.3.0


Output type:Converter name:template name
HTML:Smarty:PHP

[Add the converter in the help box.](#)



W sekcji Options możemy zdefiniować tytuł, domyślną kategorię, czy program ma wpisywać elementy oznaczone jako private...

Ostatnim krokiem jest kliknięcie przycisku "Create".



create

create (new window)

W konsoli programu powinniśmy ujrzeć wiele komunikatów oraz, na końcu, ten najbardziej interesujący.

```
Conversion time: 1 seconds

Total Documentation Time: 1 seconds
done

Operation Completed!!
```

Zakończono

I w zadanym folderze mamy stworzone pliki z dokumentacją naszych skryptów PHP, w zadanym formacie.

Te pliki możemy teraz rozpowszechniać, bądź umieścić na stronie internetowej naszego skryptu, co pozwoli innym na szybkie rozeznanie się w naszym kodzie.

Zakończenie

phpDocumentor pozwala ułatwić pracę przy projektach wieloosobowych - używanie narzędzi tego typu bądź nawet proste (ale dokładne) "zwykłe" komentowanie kodu jest niezbędne, gdy wiele osób ma dostęp do kodu i go nieustannie poprawia.

W firmach phpDoc jest stosowany właśnie w tym celu, by kolejna osoba, która siędzie do kodu, mogła zorientować się, co dany fragment dokładnie robi - dzięki świetnej strukturze dokumentów, nawigacja po kolejnych elementach jest banalna, nie trzeba ręcznie przeglądać ogromnej rzeszy plików.

Źródło: 4programmers.net. Treść udostępniona na zasadach licencji [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/)

Autor: 4programmers.net

Artykuł pobrano ze strony eioba.pl