

Tomasz "Zyx" Jędrzejewski

Dokumentowanie kodu PHP w TypeFriendly

Wersja 1.0 (02.7.2009)



Szczegółowe informacje o licencji znajdują się pod artykułem.

www.zyxist.com

Dokumentowanie kodu PHP w TypeFriendly

Aplikacja to nie tylko kod źródłowy, ale i szereg dodatkowych zależności, których nie widać na pierwszy rzut oka, przeglądając listingi. Nawet najlepiej napisany skrypt będzie stanowić czarną magię dla nowych użytkowników, jeśli nie wprowadzimy ich w zasady działania oraz nie pokażemy, jak z niego korzystać. W poprzednim artykule, „Dokumentowanie kodu PHP w phpDocumentor” zajmowaliśmy się generowaniem dokumentacji API, przydatnej przy wyszukiwaniu konkretnych klas i funkcji, zaś teraz skupimy się na budowie podręczników użytkownika, które można sobie przeczytać w wolnym czasie, jak książkę, po kolei poznając wszystkie aspekty naszego projektu.

PhpDocumentor jest narzędziem koncentrującym się przede wszystkim na opisie API, zaś możliwość dołączania zwykłych artykułów traktowana jest jako dodatek funkcjonalny. Przyjęte rozwiązania nawigacyjne nie pozwalają stworzyć tam czytelnej struktury rozdziałów, dlatego w wielu projektach podręczniki użytkownika opracowywane są przy pomocy innych narzędzi. Pomysłów jest kilka. Niekiedy na ten cel adaptuje się systemy wiki, dzięki czemu prace można prowadzić on-line. Wadami wiki jest konieczność ręcznego tworzenia całej nawigacji oraz brak przenośności. Najczęściej nie mamy możliwości wygenerowania statycznej wersji HTML, którą można by dołączyć do archiwum z projektem lub opublikować na innej stronie WWW. Czytelnicy uzależnieni są zatem od sprawności naszych serwerów.

Do tworzenia dokumentacji bardzo często adaptowana jest przemysłowa aplikacja XML DocBook, wprowadzona już przy okazji omawiania phpDocumentora. Przykładami rozbudowanych podręczników stworzonych w tej technologii jest np. dokumentacja do PHP czy Zend Frameworka. Jednak każdy, kto próbował uzyskać podobne efekty, wie, że jest to bardzo trudne zadanie. DocBook to jedynie format publikacji, natomiast my potrzebujemy swobodnego frameworka, który będzie potrafił nam wszystko przetworzyć na formaty wyjściowe, a także umożliwi modularyzację projektu i podział źródłowego tekstu na wiele plików. Zbudowanie takiego narzędzia od podstaw wymaga bardzo dobrej znajomości technicznych niuansów funkcjonowania parserów XML oraz XSLT, a także sporo czasu. Wspomniane już PHP oraz Zend Framework posiadają własne systemy generowania dokumentacji z DocBooka, jednak czy naprawdę konieczne jest tracenie setek godzin czasu, by móc napisać podręcznik i podzielić się nim z użytkownikami?

W tym artykule wprowadzone zostanie narzędzie TypeFriendly będące właśnie tego typu gotowym frameworkiem do pisania podręczników, inspirowane możliwościami „prywatnych” systemów dokumentacyjnych zbudowanych w oparciu o DocBook. W TypeFriendly można tworzyć zarówno podręcznik użytkownika, jak i dokumentację API, jednak – odwrotnie niż w phpDocumentorze – główny nacisk położony jest na pierwszą część. W dodatku nie jest on związany z żadnym konkretnym językiem programowania, przez co można go wykorzystywać nie tylko do opisywania kodu PHP.

Zasada działania

TypeFriendly buduje podręcznik na podstawie zbioru plików tekstowych sformatowanych przy pomocy składni Markdown i opisanych dodatkowymi tagami. Korzystając z samych nazw plików, potrafi określić zależności między poszczególnymi rozdziałami, ułożyć je w należyтым porządku i wygenerować całą nawigację. Tagi pozwalają na określenie tych atrybutów rozdziału, których nie da się zakodować w samej nazwie pliku. Podczas formatowania tekstu rozdziałów, skrypt automatycznie odnajduje fragmenty kodu źródłowego, dodając kolorowanie składni oraz rozmaite ramki informacyjne pozwalające wyróżnić istotne informacje.

Instalacja

W chwili pisania tego tekstu najnowszą dostępną wersją TypeFriendly jest 0.1.2. Działa on wyłącznie na PHP5 i można go zainstalować manualnie. W tym celu odwiedzamy stronę <http://www.invenzzia.org/> i pobieramy najnowszy dostępny pakiet z sekcji „Download”. Ściągnięte archiwum będzie zawierać katalog z plikami projektu, które należy rozpakować w dowolne miejsce na twardym dysku. TypeFriendly obsługiwany jest wyłącznie z linii komend, dlatego do korzystania z niego nie jest potrzebne posiadanie serwera WWW, jednak z drugiej strony musimy znać ścieżkę do pliku wykonywalnego PHP (w systemach uniksowych przeważnie */usr/bin/php*). Najlepiej dodać ją do zmiennej środowiskowej *PATH*, aby nie trzeba było jej pisać za każdym

razem, gdy chcemy wygenerować dokumentację. Pod Windowsem można to zrobić w Panelu Sterowania → System → Zaawansowane → Zmienne środowiskowe.

Pierwsze użycie

Pora przetestować działanie TypeFriendly. Wraz z projektem dostajemy też źródłową wersję jego własnej dokumentacji, która służy jednocześnie jako przykład, na którym możemy sprawdzić czy wszystko zostało poprawnie zainstalowane. Uruchamiamy systemową konsolę i poleceniem `cd` przechodzimy do katalogu, gdzie znajduje się projekt:

```
Windows:
C:
cd sciezka\do\TypeFriendly
php typefriendly.php build ./docs/

Linux:
cd /sciezka/do/TypeFriendly
php typefriendly.php build ./docs/
```

Po chwili w katalogu `/docs/output` powinna pojawić się dokumentacja projektu w formacie HTML, z którą warto się zapoznać. Możemy także podać trzeci argument: `-l pl`, aby wygenerować dokumentację w języku polskim.

Dokumentowanie projektu

TypeFriendly jest bardziej procesorem tekstu, niż generatorem dokumentacji. Można w nim tworzyć różne publikacje tekstowe w formacie HTML, nawet zupełnie niezwiązane z programowaniem. Nasza dokumentacja podzielona jest na rozdziały i podrozdziały, które musimy jedynie wypełnić treścią. Najpierw jednak musimy rozpocząć nowy projekt przy pomocy kreatora. Zakładamy jakiś pusty katalog i w konsoli systemowej wydajemy polecenie:

```
php ./typefriendly.php create ./sciezka/do/pustego/katalogu
```

Zostaniemy zapytani o kilka podstawowych informacji dotyczących naszej publikacji takich, jak tytuł, rodzaj i licencja, zaś w katalogu pojawi się kilka pozycji:

- `/input` – katalog na wersję źródłową publikacji
- `/output` – katalog na wersję HTML publikacji
- `settings.ini` – konfiguracja projektu
- `sort_hints.txt` – konfiguracja kolejności rozdziałów

Wejźdźmy do katalogu `input`, w którym domyślnie umieszczony będzie jeden podkatalog: `en`. Oznacza on wersję językową, ponieważ TF wspiera tworzenie publikacji wielojęzycznych. Jeśli potrzebujesz, możesz spokojnie przemianować go na `pl`, jednak musisz wtedy także zajrzeć do pliku `settings.ini` i analogicznie zmienić wartość opcji `baseLanguage`. W katalogu danego języka umieszczamy pliki rozdziałów.

Pojedynczy rozdział to zwykły plik tekstowy, który można tworzyć i edytować nawet w najprostszym edytorze – należy jedynie upewnić się, że wspiera on kodowanie Unicode, gdyż właśnie w nim musi być zapisany tekst. Wszystkie rozdziały posiadają swoje identyfikatory zbudowane z małych liter, podkreśleń i pauz. Za ich pomocą można tworzyć odnośniki między różnymi stronami publikacji i ogólnie odróżniać jedne rozdziały od innych. Nazwa pliku rozdziału używana jest do zakodowania informacji o identyfikatorze oraz o rozdziałach nadrzędnych. Przykładowo, chcemy stworzyć strukturę taką, jak zaprezentowana poniżej:

1. Instalacja (ID: `installation`)
 1. Instalacja podstawowa (ID: `basic`)
 2. Instalacja rozszerzeń (ID: `extensions`)

Rozdział „Instalacja” zapisujemy wtedy w pliku `installation.txt`, zaś jego podrozdziały odpowiednio do

installation.basic.txt oraz *installation.extensions.txt*. Jak widać, identyfikatory w nazwie pliku oddzielamy kropką, a system jest prosty i intuicyjny. Wrócimy do niego w późniejszej części artykułu.

Kreator publikacji utworzył nam jeden domyślny rozdział: *preface.txt*, domyślnie przeznaczony na wstęp. Rzućmy okiem na jego budowę:

```
Title: Preface

---

This documentation was auto-generated by TypeFriendly. You can fill it
with your content now.
```

Rozdziały składają się z dwóch części: nagłówek oraz treści oddzielonych poziomą linią z kilku pauz. W nagłówku podajemy tzw. *tagi* opisujące różne parametry dokumentu. Tutaj jedynym tagiem jest *Title*, dzięki któremu możemy nadać rozdziałowi tytuł, ale ich zbiór jest dużo większy. Treść dokumentu to zwykły tekst sformatowany w składni Markdown. Jej cechą jest naturalność i czytelność. Jeśli pisałeś kiedyś README jako plik tekstowy, jest duże prawdopodobieństwo, że parser Markdowna dałby radę go poprawnie wyświetlić.

Spróbujmy nieco rozbudować wstęp o kilka informacji dotyczących projektu:

```
Title: Wstęp

---

Dziękujemy za skorzystanie z [naszego projektu] (http://www.example.com/).
Posiada on szereg unikalnych właściwości, które poprawiają jakość Twojej
pracy:

1. Prostota użycia
2. Obszerna ilość dodatków
3. Rozbudowana dokumentacja i pomoc
4. Dodatkowe wsparcie techniczne

Życzymy owocnej pracy!
```

W Markdownie paragrafy oddzielamy od siebie pojedynczą, pustą linią. Listy numerowane można tworzyć w sposób naturalny, aczkolwiek nie trzeba przejmować się poprawnością numeracji. W razie czego Markdown oferuje w takich miejscach alternatywną składnię. Na odnośnik składają się zawsze dwie pary nawiasów. Pierwsza z nich obejmuje tekst, który chcemy uczynić klikalnym, druga zaś podaje miejsce docelowe, przy czym od rodzaju nawiasów zależy rodzaj odnośnika – w okrągłych możemy podawać adresy URL, w kwadratowych – identyfikatory innych rozdziałów naszej publikacji.

Spróbujmy przyjrzeć się teraz, jak opisać instalację skryptu (*installation.basic.txt*):

```
Title: Instalacja podstawowa
SeeAlso:
  - installation.extensions

---

Aby zainstalować skrypt, należy pobrać najnowszą wersję z [naszej strony
WWW][nasza-www] i rozpakować ją na naszym dysku twardym. Upewnij się, że
posiadasz PHP w wersji co najmniej 5.2[^1]. W poniższym opisie będziemy
używać następujących określeń:

katalog roboczy
:   Katalog, z którego korzystać będzie skrypt do przechowywania i
    obróbki swoich danych.
```

```
katalog instalacyjny
: Katalog, w którym skrypt będzie zainstalowany. Musi on być dostępny
dla serwera WWW.
```

Skopiuj pliki projektu do katalogu instalacyjnego na swoim dysku twardym i uruchom z linii komend skrypt instalacyjny:

```
~~~~
```

```
[console]
$ install.sh
~~~~
```

Przeprowadzi Cię on przez proces instalacyjny. Zalecane jest po jego zakończeniu przejrzeć plik konfiguracyjny `config.php` i sprawdzić niektóre ustawienia:

```
~~~~
```

```
[php]
<?php
// Ustawienie, które coś robi
$ustawienie1 = 'wartość';

// Ustawienie, które coś robi
$ustawienie2 = 'wartość';
~~~~
```

Następnie musisz gdzieś na swoim twardym dysku utworzyć katalog roboczy, któremu nadajesz uprawnienia do odczytu i zapisu.

```
> [warning]
> Upewnij się, że PHP ma odblokowany dostęp do podanego katalogu w
dyrektywie `open_basedir`.
```

```
[nasza-www]: http://www.example.org/download "Dział download na naszej
stronie WWW"
```

```
[^1]: Dzięki zastosowaniu paru sztuczek, da się uruchomić skrypt również
na PHP 5.1, ale nie jest to zalecane.
```

Przeanalizujmy plik po kawałku:

1. Tag *SeeAlso* pozwala na stworzenie pod dokumentem sekcji „Zobacz także”, gdzie pojawiają się odnośniki do innych rozdziałów. Wymieniamy w nim ich identyfikatory.
2. W pierwszym zdaniu występuje nowa metoda na tworzenie odnośników z użyciem referencji. W dłuższych opracowaniach umieszczanie adresów URL bezpośrednio w tekście może sprawiać problemy z ich nadzorem i czytelnością dokumentu. Dlatego zamiast tego możemy w dodatkowym nawiasie kwadratowym zdefiniować jakąś nazwę, a resztę odnośnika zdefiniować w końcowej części dokumentu (przedostatnia linijka, z tekstem „*[nasza-www]:*”. Zwróć uwagę, że tej samej referencji możesz używać wielokrotnie, a jeśli jej nie zdefiniujesz, TypeFriendly będzie próbował zamienić *[tekst][identyfikator]* na odnośnik do rozdziału *identyfikator.txt*.
3. Nieco dalej napotykamy symbol *[^1]*. To nic innego, jak mechanizm tworzenia przypisów. W podobny sposób, jak referencje przy odnośnikach, możemy pod dokumentem dodać treść przypisu.
4. Pod pierwszym katalogiem znajduje się tzw. lista definicji umożliwiająca proste definiowanie różnych pojęć. Pojęcia oddzielone są pustą linijką i składają się z:
 1. Nazwy pojęcia
 2. Jednej lub większej ilości definicji pisanych w kolejnych linijkach i rozpoczynanych dwukropkiem.
5. Linijki zawierające ciąg tyld definiują blok kodu, czyli obszar, w którym składnia Markdown nie

obowiązuje i który zostanie wyświetlony przy użyciu czcionki o stałej szerokości znaków. Można go używać do podawania fragmentów kodu źródłowego. Ponadto, TypeFriendly wspiera kolorowanie składni. W pierwszej linijce bloku kodu możemy umieścić w nawiasach kwadratowych nazwę języka, w jakim napisany jest podany kod. Za kolorowanie odpowiedzialny jest znany pakiet GeSHi i w TF dostępne są wszystkie obsługiwane przez niego języki. Dodatkowo można używać specjalnego kolorowania o nazwie *console* służącego do wklejania listingów z konsoli systemowej.

6. Odwrócone apostrofy umożliwiają umieszczanie w tekście pisanim wstawek pisanych czcionką o stałej szerokości znaków, np. nazw plików, funkcji czy klas.
7. Na samym dole dokumentu widoczne są linijki cytatu, mające podobne zastosowanie, jak w wiadomościach e-mail. Ich funkcjonalność jest jednak dodatkowo rozszerzona na ramki informacyjne, które można wykorzystać w różnych celach. Aby stworzyć ramkę informacyjną, w pierwszej linijce cytatu należy w nawiasach kwadratowych określić jej rodzaj. *Warning* sprawi, że wiadomość pojawi się w ramce z ikonką wykrzyknika obok tekstu, przykuwając uwagę czytelnika. Wewnątrz bloków cytatu wciąż można korzystać ze składni Markdown.

Ostateczny rezultat prezentuje poniższa ilustracja:

2.1. Instalacja podstawowa

Aby zainstalować skrypt, należy pobrać najnowszą wersję z [naszej strony WWW](#) i rozpakować ją na naszym dysku twardym. Upewnij się, że posiadasz PHP w wersji co najmniej 5.2¹. W poniższym opisie będziemy używać następujących określeń:

katalog roboczy
☞ Katalog, z którego korzystać będzie skrypt do przechowywania i obróbki swoich danych.

katalog Instalacyjny
☞ Katalog, w którym skrypt będzie zainstalowany. Musi on być dostępny dla serwera WWW.

Skopiuj pliki projektu do katalogu instalacyjnego na swoim dysku twardym i uruchom z linii komend skrypt instalacyjny:

```
$ install.sh
```

Przeprowadź Cię on przez proces instalacyjny. Zalecane jest po jego zakończeniu przejrzeć pliku konfiguracyjnego `config.php` i sprawdzenie niektórych ustawień:

```
<?php
// Ustawienie, które coś robi
$ustawienie1 = 'wartość';

// Ustawienie, które coś robi
$ustawienie2 = 'wartość';
```

Następnie musisz gdzieś na swoim twardym dysku utworzyć katalog roboczy, któremu nadajesz uprawnienia do odczytu i zapisu.

⚠ Upewnij się, że PHP ma odblokowany dostęp do podanego katalogu w dyrektywie `open_basedir`.

1. Dzięki zastosowaniu paru sztuczek, da się uruchomić skrypt również na PHP 5.1, ale nie jest to zalecane. ↩

Zobacz także:

- [2.2. Instalacja rozszerzeń](#)

Jak widać, Markdown oferuje całą gamę narzędzi przydatnych przy formatowaniu tekstu i ułatwiających opracowywanie podręczników użytkownika dla skryptów i innych aplikacji. Jej opanowanie nie zajmuje zbyt wiele czasu, a pozwala niemal natychmiast uzyskiwać bardzo ładne i satysfakcjonujące efekty.

Przy tworzeniu plików rozdziałów konieczne jest zwrócenie uwagi na kilka szczegółów:

1. Gdy tworzymy podrozdział, np. *installation.basic.txt*, niezbędne jest także stworzenie rozdziału nadrzędnego: *installation.txt*
2. Identyfikatory odpowiadających sobie rozdziałów w różnych wersjach językowych muszą być identyczne. Jeśli planujesz tworzyć później tłumaczenia na inne języki, najlepiej przyjmij angielskie identyfikatory, aby np. Amerykanie nie byli raczeni w adresach URL plikami typu *instalacja.podstawowa.html*, ponieważ program powszechnej nauki polskiego jest jeszcze w powijakach.
3. Staraj się, aby identyfikator rozdziału mówił coś o jego zawartości i był czytelny. Przykładowo, rozdział *Using feature XYZ* mógłby zostać zapisany w pliku *using-xyz.txt*.

4. Stopień zagnieżdżenia podrozdziałów zależy tylko od ograniczeń systemowych oraz ograniczeń języka HTML.

Zarządzanie kolejnością rozdziałów

W porządku, nasza dokumentacja liczy już sobie kilka rozdziałów:

```
preface.txt
installation.txt
installation.basic.txt
installation.extensions.txt
```

Jednak gdy wydamy teraz komendę

```
php typefriendly.php build /sciezka/do/dokumentacji
```

otrzymamy niezbyt miły komunikat:

```
Processing the files.
Not all base chapters are defined in the sorting hint list. Missing:
"installation". I don't know, what to do.
```

Problem w rzeczywistości jest bardzo prosty do rozwiązania. TypeFriendly musi wiedzieć, w jakiej kolejności ułożyć wszystkie rozdziały, gdyż nie da się tego wywnioskować z samej nazwy pliku. W tym celu posiłkuje się dodatkowym plikiem *sort_hints.txt*, który został automatycznie wygenerowany przez kreator. Ma on bardzo prostą budowę – w kolejnych liniach podajemy identyfikatory rozdziałów w takiej kolejności, w jakiej mają one być ułożone. W tej chwili znajduje się tam jedynie linijka z rozdziałem *preface*, a to jest stanowczo zbyt mało. TypeFriendly wymaga, aby w obrębie tego samego poziomu mieć podaną kolejność wszystkich albo żadnego rozdziału (w tym drugim przypadku aplikowana jest kolejność alfabetyczna). My, oprócz *preface*, na tym samym poziomie mamy całkiem obszerny rozdział *installation*. Wystarczy więc go dopisać:

```
preface
installation
```

I tym razem nasza dokumentacja zostanie prawidłowo wygenerowana. Możemy także zdefiniować kolejność dla podrozdziałów, anulując tym samym domyślne sortowanie alfabetyczne:

```
preface
installation
installation.basic
installation.extensions
```

Ważne jest, aby podrozdziały wymieniać **po** podaniu ich rozdziału nadrzędnego.

Dokumentujemy kod

Podobnie jak omówiony w poprzednim artykule phpDocumentor, również TypeFriendly pozwala na tworzenie dokumentacji kodu źródłowego, z tym że w przeciwieństwie do tamtej aplikacji, tutaj nie jest to zadanie priorytetowe i musimy mieć świadomość, że wszystkie podstrony będziemy musieli tworzyć i dostosowywać do zmieniającej się struktury kodu ręcznie. Z tego powodu zalecane jest wykorzystywanie tej właściwości głównie do prezentowania API dla końcowego użytkownika, które nie ulega zbyt częstym zmianom, lub też w sytuacjach, gdy nie da się zastosować automatycznych generatorów dokumentacji (np. tworzymy własny język skryptowy i chcemy opisać jego bibliotekę standardową). Przyjrzyjmy się, co TypeFriendly potrafi w dziedzinie opisu kodu na przykładzie metody klasy:

```
Title: jakasMetoda()
ShortTitle: klasa::jakasMetoda()
VersionSince: 1.2.0
VersionTo: 1.5.0
Construct: method
Visibility: public
Reference: string klasa::jakasMetoda(int $arg1, int $arg2)
Arguments:
```

```

- Name: $arg1 | EType: int | Desc: Opis argumentu 1
- Name: $arg2 | EType: int | Desc: Opis argumentu 2
Returns: Metoda zwraca tekst
EThrows:
- OutOfRangeException

---

Metoda wykonuje pewną operację na dwóch liczbach i zwraca tekst. Przykład
użycia:

~~~~
[php]
$klasa = new klasa;
echo $klasa->jakasMetoda(5, 3);
~~~~

```

Tym razem duża część zabawy odbywa się w tagach, których jest tutaj cała masa. Ich zastosowanie:

1. *Title* – ten tytuł wyświetla się w nawigacji oraz w menusach
2. *ShortTitle* – ten tytuł wyświetla się przy tworzeniu odnośników. Jeśli nie jest podany, w to miejsce używana jest wartość tagu *Title*
3. *VersionSince* – od której wersji omawiany element jest dostępny
4. *VersionTo* – do której wersji omawiany element był dostępny
5. *Construct* – określa rodzaj omawianego elementu. TypeFriendly rozpoznaje tutaj całkiem dużą rodzinę identyfikatorów i automatycznie tłumaczy je na język polski, a także używa do kontroli poprawności tagów, aczkolwiek korzystanie z własnych określeń nie jest zabronione.
6. *Visibility* – widoczność elementu (publiczny, prywatny, chroniony itd.)
7. *Reference* – definiuje prototyp funkcji/metody, w sposób symboliczny podając zwracane wartości oraz niezbędne argumenty.
8. *Arguments* – opis poszczególnych argumentów.
9. *Returns* – opis tego, co funkcja zwraca.
10. *Throws* – lista rzucanych wyjątków.

Wiele tagów związanych z programowaniem stosuje następującą konwencję:

1. W tagu o zwykłej nazwie, np. *Throws* podajemy odnośniki do innych rozdziałów dokumentacji. Stosowane jest to, gdy rzucony wyjątek czy klasa pochodna jest omawiana w naszej dokumentacji i pragniemy zastosować klikalny odnośnik.
2. Jeśli nazwa tagu poprzedzona jest literą „E”, np. *EThrows*, wymieniamy po prostu nazwy elementów, gdyż w domyśle nie są one zdefiniowane w naszej dokumentacji (np. mogą być częścią biblioteki standardowej języka). „E” jest skrótem od słowa „External”.

Istnieje także spora grupa tagów umożliwiających definiowanie relacji między klasami.

Kontrola wersji

Zaletą podręczników w formacie tekstowym jest możliwość ich umieszczenia w systemie kontroli wersji, dzięki czemu zyskujemy dokładny rejestr zmian i rozwoju dokumentacji. Systemy takie, jak SVN, umożliwiają umieszczenie w zarządzanych przez siebie plikach specjalnych słów kluczowych informujących o tym, w jakiej rewizji dany plik był ostatnio modyfikowany i przez kogo. TypeFriendly udostępnia specjalny tag, w którym takie słowa można umieścić. Poniższy przykład pochodzi z jednego z moich własnych projektów:

```
Title: Instructions
```

```
VCSKeywords: $Id: syntax.instructions.txt 128 2009-06-12 12:19:41Z zyxist
$
----
An instruction is any XML tag in one of the registered namespaces that...
(dalsza treść)
```

Co więcej, w konfiguracji publikacji (*settings.ini*) możemy zdecydować, czy chcemy, aby taka informacja była także wyświetlana w publikacji wynikowej (przydatne przy rozwojowych snapshotach) czy nie:

```
versionControlInfo = false ; nie pokazuj informacji SVN-a
```

Pliki multimedialne

Gdy piszemy podręcznik użytkownika bądź programisty, często niezbędnym elementem są diagramy czy screenshoty. System dokumentacji musi wspierać proste osadzanie elementów graficznych w tekście. Tak też jest w TypeFriendly – zaczynamy od stworzenia katalogu */input/pl/media*, do którego kopiujemy żądane grafiki. Następnie osadzamy je w dokumencie przy pomocy konstrukcji niemal identycznej, jak odnośniki:

```
Title: Wybory do parlamentu
---
W roku 2007 odbyły się wybory do parlamentu. Podział mandatów między
ugrupowaniami przedstawia poniższy wykres:

![Podział mandatów w parlamencie] (media/podzial_mandatow.png)
**Podział mandatów w parlamencie**

Partia XYZ jest zdecydowanym zwycięzcą...
```

Jedyna różnica to wykrzyknik poprzedzający całość. Zauważmy, że katalog z plikami multimedialnymi znajduje się w tym samym miejscu, co rozdziały, dlatego przy tworzeniu publikacji wielojęzycznych każdy język może przechowywać swoje własne grafiki.

Wielokrotne wykorzystanie tekstów

W TypeFriendly 0.1.2 pojawił się załączek mechanizmu szablonów umożliwiający zapisanie często powtarzających się fragmentów tekstu w jednym miejscu i doklejanie ich do różnych rozdziałów. W tej wersji ograniczony jest on wyłącznie do semantycznego tagu *FeatureInformation* pozwalającego na dodatkowe scharakteryzowanie funkcjonalności omawianego elementu. Przypuśćmy, że posiadamy dwie funkcje: jedna z nich jest dopiero w stadium eksperymentalnym i jej implementacja może zawierać poważne błędy, zaś druga jest powoli wycofywana. Chcielibyśmy, aby były one oznakowane dodatkowym komunikatem dla czytelnika. W tym celu tworzymy katalog */input/pl/templates*, a w nim dwa pliki:

deprecated.txt:

```
> [warning]
> Ta funkcja jest nieaktualna i będzie wycofywana z przyszłych wydań. Nie
zalecamy jej używania w nowopowstających projektach.
```

experimental.txt:

```
> [warning]
> Ta funkcja ma status eksperymentalny. Może ona zawierać błędy, a
szczególnie jej implementacji mogą być bez ostrzeżenia zmienione w
najbliższych wydaniach.
```

Teraz możemy dokleić taki komunikat do rozdziału dokumentującego wybraną funkcję:

```
Title: eksperymentalnaFunkcja
```

```
ShortTitle: eksperymentalnaFunkcja
Construct: function
FeatureInformation: experimental

---

Opis...
```

Nad tekstem pojawi się w dokumencie wynikowym ramka informacyjna zdefiniowana w szablonie *experimental.txt*. Dalsze poszerzenie możliwości szablonów będzie implementowane w kolejnych wersjach TypeFriendly wraz z rozwojem parsera Markdown.

Porady praktyczne

Rozdział ten jest kontynuacją rozważań z poprzedniego artykułu, „Dokumentowanie kodu PHP w phpDocumentor”. W chwili obecnej mamy już omówione wszystkie narzędzia, jakie są potrzebne, by stworzyć kompleksową i wyczerpującą pomoc dla naszego projektu. Niestety, jest to tylko połowa sukcesu. Reszta zależy od Twoich własnych umiejętności przekazywania informacji oraz takiego ich zorganizowania, aby czytelnik nie pogubił się w połowie lektury. Najprawdopodobniej Twoje podręczniki będą stopniowo ewoluować pod wpływem sugestii czytelników, gdyż nie da się jednoznacznie określić, jaką formę najbardziej oni preferują. Przygotuj się na to i pamiętaj, że chociaż opracowywanie pomocy dla projektu jest równie trudnym zadaniem, jak sam projekt, może ono mieć decydujący wpływ na jego przyszłość i popularność.

Pierwszą czynnością powinno być przyjęcie pewnych konwencji odnośnie narzędzia. Zapoznaj się z oferowanymi możliwościami i zdecyduj, w jakich sytuacjach będziesz ich używać. Najlepiej poinformuj o swoich wyborach czytelnika we wstępie tak, jak to czyni wielu autorów książek:

```
W niniejszym podręczniku przyjęliśmy kilka konwencji typograficznych, aby
ułatwić zrozumienie tekstu. Nazwy plików i funkcji będziemy oznaczać
czcionką o stałej szerokości znaków, np. `plik.txt`. O często popełnianych
błędach będziemy informować przy pomocy następującej ramki:

> [error]
> Taka ramka opisuje często popełniany błąd.

Inne stosowane przez nas ramki to...
```

Dzięki temu unikniesz nieporozumień i poprawisz jakość odbioru tego, co chcesz przekazać.

Kolejnym elementem jest rozplanowanie treści. W podręcznikach można wyróżnić rozdziały, których głównym celem jest objaśnienie wszystkich funkcji programu, oraz takich, gdzie chcemy wprowadzić czytelnika w jakieś zagadnienie. Oba wymagają nieco innego podejścia. Rozdziały opisujące program są skierowane przede wszystkim dla osób, które w jakiś sposób już go poznały i szukają pomocy odnośnie konkretnych funkcji. Tutaj kluczem powinna być łatwa nawigacja, prostota dotarcia do potrzebnej informacji i zwięzły, ale rzeczowy opis. Nie trzeba chyba wspominać, że powinien on być zgodny z rzeczywistością. W przypadku kodu źródłowego zadanie takie realizuje phpDocumentor, natomiast elementy takie, jak interfejs użytkownika, mogą być opisane w TypeFriendly.

Wprowadzenie nowicjuszy w świat programu to nieco inne zadanie. Tutaj musimy wcielić się w rolę nauczyciela. Zaczynamy od opracowania sposobu przedstawienia głównych funkcji programu użytkownikowi. Obejmuje to m.in. kolejność prezentowania kolejnych partii informacji związaną z zależnościami między nimi oraz ponownie – przyjęcia pewnych konwencji. Podczas opisywania nie powinniśmy się starać przedstawiać danego zagadnienia dogłębnie, od początku do końca; nawet jest to wysoce niewskazane w przypadku bardziej złożonej funkcjonalności. Naszym zadaniem jest sprawienie, aby czytelnik „poczuł” projekt i się w nim odnalazł, a zalewanie go tysiącami informacji i skomplikowanymi terminami na pewno mu w tym nie pomoże.

Kluczem do zrozumienia są przykłady – możemy zakładać, że ludzie są inteligentni, dlatego jeden trafny fragment kodu z krótkim komentarzem może być dużo bardziej zrozumiały, niż wyczerpujący opis. Nie obawiaj się angażować różnych zmysłów czytelnika, ubogacając opisy w graficzne ilustracje i wyróżniki, a także powtarzać tych samych informacji. Nie wszystko da się zapamiętać podczas czytania, a przypomnienia służą

uporządkowaniu materiału i jego utrwaleniu. Z drugiej strony, dobrym pomysłem jest pozwolenie na wykazanie się czytelnikowi: omówienie zagadnienia, przykład i lista krótko omówionych opcji, z którymi może poeksperymentować. Taki zabieg podnosi dodatkowo wartość merytoryczną danego rozdziału i sprawia, że staje się on przydatny nie tylko podczas zapoznawania się z programem.

Ostatnia sprawa to język i styl wypowiedzi. Tego niestety nie da się nauczyć przez lekturę artykułu, ale przez praktykę w hurtowych ilościach. Pamiętaj o tym, kto jest adresatem Twojego tekstu, jaki jest jego cel oraz zwracaj uwagę na ortografię i prawidłowe użycie interpunkcji. Nieużywanie przecinków, kropek, dużych liter czy polskich znaków diaktrycznych odbierane jest przez wielu czytelników jako przejaw ignorancji i na pewno pozostawi w nich niemiłe pierwsze wrażenie, a tego przecież chcemy uniknąć.

Zakończenie

Dokumentowanie projektu jest równie ważne, jak sam projekt, tymczasem często spotyka się z lekceważeniem ze strony programistów. Jeden z powodów to brak wiedzy o narzędziach, jakie można wykorzystać, drugi – pisanie dokumentacji wymaga pewnych zdolności humanistycznych i umiejętności redagowania zrozumiałego tekstu. Jednak wszystko da się wypracować, odpowiednio ćwicząc. Dzięki tej serii artykułów poznałeś kilka narzędzi, które możesz wykorzystać i skupić się w całości na stronie merytorycznej. PhpDocumentor pozwala na szybkie tworzenie i aktualizację dokumentacji kodu pomagającej w rozwoju projektu i stanowiącej główne źródło informacji o tym, jak jest on zbudowany i jak działa. TypeFriendly umożliwia stworzenie eleganckich podręczników użytkownika/programisty niezbędnych do pokazania sposobów korzystania z projektu i tych informacji, których nie da się tak prosto wywnioskować, analizując strukturę kodu. Jak dużo będziesz w stanie z tego wyciągnąć, zależy tylko od Ciebie.

Tomasz "Zyx" Jędrzejewski

Aktualna wersja artykułu zawsze na **www.zyxist.com**

Licencja

Artykuł rozpowszechniany jest na licencji **Creative Commons Uznanie autorstwa-Użycie niekomercyjne-Bez utworów zależnych 2.5 Polska**.

Wolno:

- kopiować, rozpowszechniać, odtwarzać i wykonywać utwór

Na następujących warunkach:

1. *Uznanie autorstwa*. Utwór należy oznaczyć w sposób określony przez Twórcę lub Licencjodawcę*.
2. *Użycie niekomercyjne*. Nie wolno używać tego utworu do celów komercyjnych.
3. *Bez utworów zależnych*. Nie wolno zmieniać, przekształcać ani tworzyć nowych dzieł na podstawie tego utworu.

W celu ponownego użycia utworu lub rozpowszechniania utworu należy wyjaśnić innym warunki licencji, na której udostępnia się utwór.

Każdy z tych warunków może zostać uchylony, jeśli uzyska się zezwolenie właściciela praw autorskich.

Powyższe postanowienia w żaden sposób nie naruszają uprawnień wynikających z dozwolonego użytku ani żadnych innych praw.

Internetowa skrócona wersja licencji:

<http://creativecommons.org/licenses/by-nc-nd/2.5/pl/>

Pełen tekst licencji:

<http://creativecommons.org/licenses/by-nc-nd/2.5/pl/legalcode>

Podczas publikacji należy podać następujące informacje:

1. Link do internetowej skróconej wersji licencji.
2. Informację o wersji publikowanego tekstu.
3. Informacje o autorze w następujący sposób:

Tomasz "Zyx" Jędrzejewski

Aktualna wersja artykułu zawsze na **www.zyxist.com**