

# Tworzenie pluginów do Wordpressa

Wordpress to najpopularniejszy i według wielu programistów i użytkowników najlepszy na świecie system do tworzenia blogów i nie tylko. Na wordpresie powstaje coraz więcej stron korporacyjnych. Czasem można się nawet spotkać ze sklepami opartymi na tym systemie, które możemy tworzyć w prosty sposób za pomocą różnego typu pluginów, czy jak kto woli wtyczek. Dzisiaj zaprezentuję sposób na tworzenie własnych pluginów. Zaczniemy od prostego dodatku o nazwie Hello World, by dowiedzieć się na jakiej zasadzie zbudowane są pluginy do Wordpressa, a skończymy na pluginie zamieszczającym we wpisie informację o podobnych artykułach które mogą spodobać się czytelnikom naszego bloga.

## Nowa wersja Wordpress

Całkiem niedawno, bo w czerwcu, wyszła nowa trzecia już wersja Wordpress'a, w której dokonano ponad 1200 zmian, wprowadzono nowe kolory do panelu administracyjnego, zmieniono domyślną skórkę na Twenty Ten, ale przede wszystkim wbudowano system wordpress MU, który pozwala za pomocą jednego wordpress'a utworzyć kilka innych blogów. Zmiana ta jest w szczególności przydatna jeśli prowadzimy bloga i wydaje nam się że treści należało by podzielić, na przykład oddzielając prywatne przemyślenia od portfolio, czy tworząc sieć blogów. Nowa wersja wprowadziła też możliwość dodawania niestandardowych typów wpisów, dokładny menadżer menu, możliwość ustalenia hasła administratora bezpośrednio podczas instalacji i kilka innych ciekawych zmian, które można odkryć osobiście.

## Instalacja wordpressa

Skoro już wszyscy doskonale wiemy czym jest wordpress, czas na rozpoczęcie zabawy z jego programowaniem. Zaczniemy od programu zajmującego się wypisaniem w wybranym przez nas miejscu napisu „Hello World”.

Aby móc zabrać się za jego pisanie najpierw musimy ściągnąć wordpress'a, znajdziemy go na stronie wordpress.org. Po rozpakowaniu paczki i umieszczeniu jej w katalogu naszego lokalnego serwera, należy ją zainstalować. Potrzebujemy do tego serwera z php4 i mysql. Przed zainstalowaniem musimy stworzyć bazę danych, na której będzie umieszczony nasz blog. Sam proces instalacji jest niezwykle prosty wystarczy kolejno podać:

- Nazwę bazy danych (Database Name)
- Nazwę użytkownika bazy danych (User Name)
- Hasło do bazy danych (Password). Jeżeli użytkownik nie ma hasła możemy zostawić to pole puste, jednak dla bezpieczeństwa najlepiej je ustawić.
- Adres bazy danych (Database Host). Najczęściej będzie to localhost.
- Prefix tabel (Table Prefix). Prefix umożliwia nam instalację kilku wordpress'ów na jednej bazie danych.

Instalację możemy przeprowadzić także za pomocą pliku konfiguracyjnego. Plik, jaki należy utworzyć to wp-config.php. Do paczki z wordpressem dołączany jest zawsze plik wp-config-sample.php zawierający przykładową konfigurację. Należy go skopiować i zmienić jego nazwę na wp-config.php, po czym wypełnić zgodnie z informacjami podanymi w komentarzach. Z plikiem wp-config.php warto się zapoznać, bo gdy będziemy chcieli przenieść naszego wordpressa np. z serwera lokalnego na serwer w sieci, łatwiej będzie

wykonać kopię bazy danych i umieścić ją na serwerze, niż dodawać wszystkie dane od nowa.

## Zacznijmy tworzyć

Aby utworzyć nasz pierwszy plugin musimy stworzyć plik, zawierający informacje o autorze, wersji, nazwie itp. Wszystkie te informacje, podobnie jak w przypadku tworzenia templatek, muszą być podane jako komentarz. Utwórzmy plik pod nazwą `hi.php` w katalogu `wp-content/plugins/`. Listing 1 zawiera poprawnie wypełniony plik z informacjami o pluginie. Wtyczki w wordpress działają tylko wtedy, jeśli zostały włączone. Po dodaniu pliku z kodem z listingu 1 powinniśmy otrzymać możliwość włączenia naszej wtyczki w panelu administracyjnym. Jak na razie nie wykonuje ona żadnej akcji, żeby to zmienić do pliku z pluginem powinniśmy dodać funkcję `add_action`, która w parametrach przyjmuje nazwę akcji przed jaką ma wykonać się parametr podany w drugim argumencie oraz opcjonalnie priorytet (liczba całkowita od 1 do 10) i ilość argumentów przyjmowanych przez naszą funkcję. Wszystkie nazwy akcji, czyli miejsc w którym napisana przez nas funkcja się wykona można znaleźć na stronie [http://codex.wordpress.org/Plugin\\_API](http://codex.wordpress.org/Plugin_API) jest ich ponad 80, więc warto, ją odwiedzić. Wracając jednak do tworzenia naszego pluginu, zakładamy że tekst „Witaj w wordpress” powinien się wyświetlić na dole, przed załadowaniem pliku `footer.php` z templatki. Aby tak się stało musimy jako pierwszy parametr funkcji `add_action` podać `"get_footer"` i utworzyć funkcję np. `hello_wordpress()`, która będzie wyświetlać nasz tekst i której nazwę podamy jako 2 parametr. To wystarczy, jak widać, ładnie i szybko uporaliśmy się z naszym pierwszym pluginem. Cały jego kod znajduje się na listingu 2.

## Funkcje wbudowane w wordpress

Czas poznać kilka najważniejszych, wbudowanych w wordpress'a funkcji, klas i obiektów pomocniczych. Ten najważniejszy, który przydaje się prawie zawsze, to obiekt bazy danych. W wordpress za bazę danych odpowiada zmienna globalna `$wpdb` - instancja klasy o tej samej nazwie. Używając jej będziemy mogli używać naszej bazy danych bez łączenia się i rozłączania z niej (zajmuje się tym konstruktor i destruktor). Aby wykonać zapytanie do bazy danych będziemy potrzebować prefiksu tabel zapisanego w polu `$wpdb->prefix` oraz odpowiedniej metody. Zależnie od tego, co chcemy dokonać, możemy użyć między innymi metod: „`query`” - zapytanie SQL, „`insert`” - dodanie danych do tabeli, „`select`” - pobranie danych, lub „`replace`” - zmiana danych. Kod klasy znajduje się zawsze w katalogu `wp-includes` w pliku `wp-db.php`, warto się z nim zapoznać bliżej, jak również przejrzeć jej dokumentację na oficjalnej stronie wordpress [http://codex.wordpress.org/Function\\_Reference/wpdb\\_Class](http://codex.wordpress.org/Function_Reference/wpdb_Class). Naszą następną wtyczką, jaką napiszemy do wordpress'a będzie wtyczka, która po włączeniu się doda nam do bazy danych nowy komentarz. Zacznijmy od wpisania naszego podstawowego kodu z informacją o pluginie. Tym razem będziemy chcieli wykonać jego zadanie tylko raz. Do tego celu nie użyjemy już funkcji `add_action`, lecz `register_activation_hook`, funkcja ta uruchamia podaną przez nas funkcję, kiedy plugin jest aktywowany. Nasz obecny kod powinien wyglądać mniej więcej tak jak na listingu 3. Wywoływanie pluginu podczas odpalenia jest szczególnie przydatne, kiedy na przykład chcemy by nasz plugin zbierał dane do tabeli która mogła nie zostać jeszcze utworzona.

Teraz czas zapisać realne działanie naszego pluginu. Zgodnie z naszymi ustaleniami miał on dodawać do naszej bazy danych komentarz. Ustalmy że ma to być pierwszy wpis, czyli automatycznie dodany do wordpress'a, że autorem komentarza ma być „User”, a jego

treścią „123 testujemy pluginy”. Pierwsze co musimy zrobić to dodać do naszego kodu wykorzystywaną przez nas później zmienną globalną \$wpdb, po czym użyć metodę insert z podanymi na listingu 4 parametrami. Kolejno będzie to nazwa tabeli do której dodajemy dane (w naszym przypadku tabela \$wpdb->prefix.'comments'), dane do dodania (kolejno kolumna => dane), oraz format danych, jeśli chcemy je sprawdzić, przyjmujący jedną z 3 opcji %d - dla liczb całkowitych, %s - dla stringów i %f dla liczb typu float. Ostatni parametr nie będzie nas dotyczyć. Do naszej funkcji start\_once, powinniśmy teraz dodać zawartość listingu 4. Po włączeniu naszego pluginu wystarczy wejść na stronę z komentarzami, a zobaczymy że nasz komentarz rzeczywiście został dodany. Jak widać, narzędzia do obsługi bazy danych są zbudowane całkiem przyjemnie.

## O filtrowaniu treści słów kilka

Kolejną z dość przydatnych funkcji wordpress'a są tak zwane filtry. Ich zadaniem jest stosowanie zmian w treści według określonych przez pluginy zasad. Jedną z głównych zalet filtrów jest to że dane, jakie mamy zamiar filtrować, takie jak na przykład komentarze, czy też treści, nie są zapisywane w bazie danych, lecz są stosowane zaraz przed ich użyciem. Dzięki temu administratorzy serwisu mają dostęp do oryginalnych treści i w przypadku wprowadzania zmian mogą powrócić do nich komentując zaledwie jedną linijkę kodu, zawierającego zastosowania filtrów. Filtry aktywujemy bardzo podobnie jak akcje. Za dodanie funkcji filtrujących odpowiada funkcja add\_filter. Podajemy w niej kolejno, analogicznie jak w add\_action, nazwę filtra do zastosowania, nazwę funkcji obsługującej, a także opcjonalnie priorytet i akceptowane argumenty. Nazw wszystkich filtrów, jest nawet więcej niż nazw akcji, dlatego tu także odsyłam do dokumentacji, a konkretniej listy filtrów, znajdującej się pod adresem: [http://codex.wordpress.org/Plugin\\_API/Filter\\_Reference](http://codex.wordpress.org/Plugin_API/Filter_Reference). Aby zaprezentować bezpośrednio możliwości korzystania z filtrów dopiszemy do naszego pluginu z listingu 2 filtr eliminujący wulgaryzmy z komentarzy. Na początek ustawmy w naszym pluginie funkcję add\_filter podając jako pierwszy jej parametr comment\_text, tak żeby wszystkie komentarze tekstowe były automatycznie sprawdzane przed ich wyświetleniem. Jako kolejny parametr podajemy nazwę funkcji która wykona pracę za nas. W naszym przypadku będzie to nice\_words. Oczywiście w funkcję wpisujemy odpowiednie zasady, które w najprostszej postaci mogą wyglądać tak jak na listingu 5. Jeśli odpalimy nasz plugin i w komentarzu wpisujemy „kurde”, to tekst ten zamieni się na „motyla noga”. Oczywiście dzięki filtrom możemy nie tylko eliminować przekleństwa, ale też słowa które są niepożądane czy też np. wykorzystać możliwość dodania tagów HTML do komentarza i zamiany linków do youtube na odtwarzacze filmowe, wszystko zależy od chęci i pomysłu programisty.

## Łączymy wszystko w jedną całość

Skoro znamy już całkiem nieźle teorię tworzenia wtyczek do wordpressa, czas na zrobienie czegoś konkretnego. Wpadłem na pomysł że możemy napisać plugin, doradzający naszym użytkownikom przeczytanie innego naszego wpisu. Algorytm będzie banalnie prosty, na początek ustalimy w aktywnym wpisie słowa kluczowego (podanego w polach opcjonalnych lub w nazwie) i wyszukamy go w innych wpisach z kategorii. Jeśli niczego nie znajdziemy spróbujemy wyszukać to słowo w nazwie wszystkich wpisów, jeśli to też nie przyniesie nam pożądanego rezultatu podamy link do poprzedniego lub wcześniejszego wpisu z kategorii, i znowu, jeśli takiego nie ma to wyeliminujemy kategorię i podamy inny, dowolny wpis. Całkiem proste. Jak wiadomo, żeby się do tego zabrać będziemy musieli dodać do naszego wordpress'a kilka testowych wpisów. Dobrze było by

przynajmniej dla 2 z nich ustalić tą samą kategorię, żebyśmy byli pewni że wszystko działa na 100%. Zaczniemy od stworzenia nowego pliku wtyczki napisania komentarzy do niego. To co podamy w treści nie ma jakiegoś wielkiego znaczenia, więc można to dostosować do swoich potrzeb. Kolejnym krokiem będzie ustalenie, kiedy chcemy uruchomić nasz plugin. Ja postanowiłem zrobić to, stosując akcję `comment_form`, która spowoduje wyświetlenie tekstów pod formularzem do komentowania. Nie umożliwi ona dostępu do treści posta, a jedynie wyświetlenie, dlatego właśnie najpierw wywołamy filtr czytający nazwę wpisu, a potem w wywołanej przez niego funkcji, jeśli treść znajduje się na stronie typu `single` (`is_single()`), czyli takiej z wyświetlonym tylko jednym wpisem, dodamy odpowiednie linki. Aby aktywować filtr na nazwie wpisu musimy użyć `the_title`. Nasz obecny kod powinien teraz wyglądać mniej więcej tak jak na listingu numer 6. Możemy wyświetlać treść pod polem do komentowania i mamy dostęp do nazwy wpisu. Skoro już mamy dostęp do nazwy naszego wpisu, należy ją wysłać dalej, tak by `hello_wordpress` przejęła całkowitą inicjatywę. Zrobimy to za pomocą zmiennej globalnej. Wystarczy ją zadeklarować, po czym podać w jej miejsce treść naszej nazwy - listing 7. Oczywiście mogli byśmy użyć funkcji `get_the_title`, ale chcę pokazać inną ścieżkę. Możliwość `get_the_title` zaprezentuje nam inna wbudowana funkcja - `get_the_ID()`; która pobiera identyfikator obecnego wpisu. Użyjemy ją do pobrania dostępnych nam pól opcjonalnych (Custom Fields), za pomocą funkcji `get_post_meta` przyjmującej w parametrach kolejno `$post_id` - identyfikator postu, `$key` - nazwa pola, oraz opcjonalnie czy rezultat ma być podany osobno - `$single`, standardowo jest w tablicy. Po pobraniu danych, połączymy je z nazwą a następnie podzielimy na tablicę, używając spacji jako separatora. Z tak przygotowanej tablicy możemy już stworzyć proste zapytania. Użyjemy, może nieco nieładnej metody przeszukiwania bazy danych, ale za to jednej z najprostszej - REGEXP. Aby zachować pewność że nie wykonamy zbyt ciężkiego zapytania usuńmy z naszej tablicy wszystkie wyrazy mające mniej niż 4 znaki i skróćmy nieco słowa by nie mieć problemów z odmianą, dodatkowo, by nie męczyć się zbytnio z zapytaniem dodajmy już elementy z zapytania. W zmiennej `$sql` zapiszemy nasze zapytanie, do tabeli z treściami (`wp_posts`), po czym wyświetlimy wyniki. Nasz końcowy plugin powinien wyglądać tak jak na listingu 8.

## Dla fanów open source

Jeśli tworzenie pluginów dla wordpressa Ci się spodobało, możesz je tworzyć i udostępniać na [wordpress.org](https://wordpress.org). Mam nadzieję że ten artykuł pozwolił Ci zrozumieć jak działają pluginy w wordpress i że Ci się podobał. Pozdrawiam Marcin Lenkowski - [lenkowski.net](https://lenkowski.net).

## O autorze

Marcin Lenkowski, student, programista php, wolny strzelec. Twórca bezpłatnego narzędzia do tworzenia wycen - [Softime.pl](https://softime.pl), na co dzień blogger znany jako MWL, którego wpisy znaleźć można na [lenkowski.net](https://lenkowski.net)

## Listing 1 - Komentarz z informacjami o pluginie

```
<?php
/*
Plugin Name: Nazwa Pluginu
Plugin URI: http://adres_pluginu.com/
Description: Mały opis.
Author: Imię Nazwisko
Version: 0.0.1
Author URI: http://nasz_blog.com/
*/
?>
```

## Listing 2 - Pierwszy plugin w wordpress

```
<?php
/*
Plugin Name: Nazwa Pluginu
Plugin URI: http://adres_pluginu.com/
Description: Mały opis.
Author: Imię Nazwisko
Version: 0.0.1
Author URI: http://nasz_blog.com/
*/

function hello_wordpress()
{
    echo "Witaj w wordpress";
}

add_action("get_footer", "hello_wordpress");
?>
```

## Listing 3 - Plugin uruchamiający się podczas włączenia

```
<?php
/*
Plugin Name: Nazwa Pluginu
Plugin URI: http://adres_pluginu.com/
Description: Mały opis.
Author: Imię Nazwisko
Version: 0.0.1
Author URI: http://nasz_blog.com/
*/

function start_once()
{
    // tutaj możemy dodać kod, który będzie działał podczas włączenia pluginu
}

register_activation_hook(__FILE__, "start_once");
?>
```

## Listing 4 - Kod dodający do bazy danych z wordpressem nasz komentarz

```
<?php
function start_once()
{
    global $wpdb;
    $wpdb->insert($wpdb->prefix."comments", array("comment_post_ID"=>"1",
"comment_author"=>"User", "comment_date"=>date("Y-m-d H:i:s"),
"comment_content"=>"123 testujemy pluginy"));
}
?>
```

### Listing 5 - Kod zajmujący się eliminowaniem brzydkich słów z komentarzy

```
<?php
function nice_words($comment)
{
    //$comment
    $return = str_ireplace(array('cholera', 'kurde'), 'motyla noga', $comment);
    return $return;
}

add_filter('comment_text', 'nice_words');
?>
```

### Listing 6 - Pierwsze kroki w pisaniu naszego pluginu „Suggest”

```
<?php
/*
Plugin Name: Suggest
Description: Plugin wyświetlający wszystkie podobne artykuły, napisany dla
Software Development Journal.
Author: Marcin Lenkowski
Version: 0.0.1
Author URI: http://lenkowski.net/
*/

function hello_wordpress()
{
    echo "Witaj w wordpress";
}

function run_suggest($title)
{
    if(is_single()) add_action("comment_form", "hello_wordpress");

    return $title;
}

add_filter('the_title', 'run_suggest');
?>
```

### Listing 7 - Deklaracja zmiennej globalnej do późniejszego wykorzystania

```
function run_suggest($title)
{
    global $post_title;
    $post_title = $title;
    ...
}
```

### Listing 8 - Plugin Suggest, podsuwający podobne wpisy

```
<?php
/*
Plugin Name: Suggest
Description: Plugin wyświetlający wszystkie podobne artykuły, napisany dla
Software Development Journal.
Author: Marcin Lenkowski
Version: 0.0.1
Author URI: http://lenkowski.net/
*/

function hello_wordpress()
{
    global $post_title;
}
```

```

global $wpdb;

$meta = get_post_meta(get_the_ID(), 'key', true);
$words = explode(' ', $meta['post_title']);
$array = array();

foreach($words as $word)
{
    if(strlen($word)<4) continue;
    $word = " `post_title` REGEXP '".strtolower(substr($word, 0, round(strlen($word)*0.75)))."' OR";

    if(!in_array($word, $array))
    {
        $array[] = $word;
        $return .= $word;
    }
}

$return = substr($return, 0, -3);

$sql = "SELECT `post_title`, `guid` FROM `{$wpdb->prefix}posts` WHERE
({$return}) AND `post_status`='publish' AND `ID`!='".$get_the_ID()";

$query = $wpdb->get_results($sql, ARRAY_A);

if(sizeof($query)>2) { shuffle($query); }

if(sizeof($query)>0)
{
    echo 'Przeczytaj także:<br />';
    echo '<a href="'. $query[0]['guid']. "'>". $query[0]['post_title']. '</a>';
    echo '<br />';
    echo '<a href="'. $query[1]['guid']. "'>". $query[1]['post_title']. '</a>';
}
}

function run_suggest($title)
{
    global $post_title;
    $post_title = $title;

    if(is_single()) add_action("comment_form", "hello_wordpress");

    return $title;
}

add_filter('the_title', 'run_suggest');

?>

```