Technologia i rozwiązania

Projektowanie systemów CMS przy użyciu PHP i jQuery

Zbuduj CMS na miarę swoich potrzeb!

- Jak zaprojektować jądro systemu CMS?
- Jak zarządzać użytkownikami?
- Jak przygotować instalator?

Helion

Kae Verens

[PACKT] PUBLISHING open source*

Tytuł oryginału: CMS Design Using PHP and jQuery

Tłumaczenie: Łukasz Piwko

ISBN: 978-83-246-3365-4

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/psycms.zip

Materiały graficzne na okładce zostały wykorzystane za zgodą iStockPhoto Inc.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/psycms
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

ROZDZIAŁ 3

Zarządzanie stronami — część pierwsza

W tym rozdziale stworzymy formularz do zarządzania stronami i zbudujemy system pozwalający przenosić strony metodą przeciągania.

Omówione zostaną następujące tematy:

- Żądanie i generowanie stron
- Wyświetlanie listy stron w panelu administracyjnym
- Administracja stronami

System zarządzania stronami zostanie ukończony dopiero w kolejnym rozdziale, w którym opiszę zapisywanie stron w bazie danych oraz dodam edytor tekstu i menedżera plików.

Strony w systemie CMS

Zgodnie z tym, co napisałem w rozdziale 1., strona to zasadniczo treść, która powinna zostać wyświetlona w odpowiedzi na żądanie określonego adresu URL.

W serwisach nieopartych na systemach CMS pojęcie strony jest bardzo wyraźne, ponieważ każdy adres URL zwraca konkretny plik HTML. Natomiast w systemie CMS strony są generowane dynamicznie i mogą być wzbogacone o rozmaite wtyczki, rodzaje prezentacji zależne od tego, czy użytkownik czegoś szuka, czy zastosowano opcję podziału na podstrony, itd.

W większości witryn internetowych stronę można w uproszczeniu zdefiniować jako duży obszar treści pośrodku okna przeglądarki. Czasami jednak trudno zaakceptować taką definicję, gdyż to, co widać na ekranie, może w istocie być składanką fragmentów treści z różnych miejsc serwisu.

Problem tych różnic rozwiązujemy poprzez zastosowanie „typów" stron, z których każdy może być zaprezentowany w inny sposób. Wśród typów stron można wymienić galerie, formularze, strony z wiadomościami, wyniki wyszukiwania itd.

W tym rozdziale pokażę, jak utworzyć najprostszy typ strony, o nazwie *normalny*. W panelu administracyjnym formularz służący do tworzenia tego typu stron będzie zawierał duży obszar tekstowy na treść, która po zatwierdzeniu będzie wyświetlana na stronie. Można zastosować też inną nazwę, taką jak np. „domyślny", ale ponieważ z CMS-ów często korzystają też osoby niemające wiedzy technicznej, lepiej zastosować słowo, którego znaczenie jest raczej oczywiste. Niejednokrotnie klienci pytali mnie, co oznacza słowo „default" (domyślny), natomiast słowo „normal" (normalny) jeszcze nikogo nie dziwiło.

Jak zapewne pamiętasz, w rozdziale 1. napisałem, co powinno wchodzić w skład jądra systemu, a co należałoby udostępnić w postaci wtyczki.

Każdy system CMS powinien umożliwiać tworzenie przynajmniej jednego najprostszego rodzaju stron. Dlatego typ normalny będzie wbudowany w jądro, a nie udostępniany jako dodatek.

Wyświetlanie listy stron w panelu administracyjnym

Zaczniemy od dodania odnośnika *Strony* do menu administracyjnego. W tym celu otwórz plik */ww.admin/header.php* i dodaj do niego poniższy pogrubiony wiersz kodu:

```
<ul>
    <li><a href="/ww.admin/pages.php">Strony</a></li>
    <li><a href="/ww.admin/users.php">Użytkownicy</a></li>
```

Jeszcze jedno: po zalogowaniu do sekcji administracyjnej użytkownik powinien być automatycznie przenoszony do sekcji zarządzania stronami, gdyż większość czynności związanych z zarządzaniem systemem dotyczy właśnie stron serwisu.

W związku z tym zmienimy zawartość pliku */ww.admin/index.php*, aby zawierał ten sam kod, co plik */ww.admin/pages.php*. Zamień zawartość pliku */ww.admin/index.php* na następujący kod:

```
<?php
require 'pages.php';
```

Teraz zajmiemy się budową sekcji *Strony*.

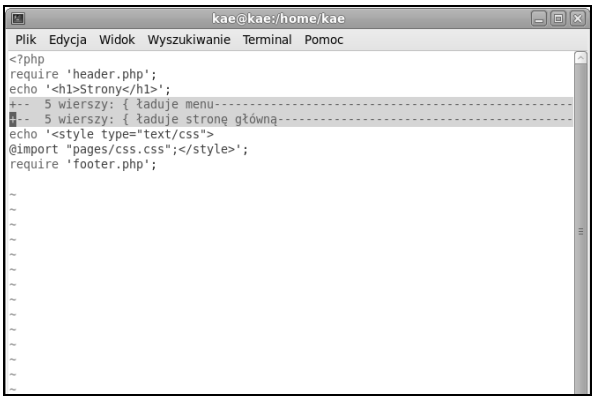
Zaczniemy od utworzenia pliku */ww.admin/pages.php*:

```
<?php
require 'header.php';
echo '<h1>Pages</h1>';
// { ładuje menu
echo '<div class="left-menu">';
require 'pages/menu.php';
echo '</div>';
// }
// { ładuje stronę główną
echo '<div class="has-left-menu">';
require 'pages/forms.php';
echo '</div>';
// }
echo '<style type="text/css">
  @import "pages/css.css";</style>';
require 'footer.php';
```

Zwróć uwagę na specjalne komentarze, w które ująłem bloki kodu (początek bloku oznaczony jest znakami //{, a koniec znakami //}).

Komentarze te dodałem dlatego, gdyż niektóre edytory oferują tzw. **funkcję zwijania**, która pozwala zwijać bloki kodu ujęte w specjalne znaczniki, tak aby widoczny był tylko pierwszy wiersz.

Przykładowo u mnie w edytorze Vim powyższy fragment kodu wyświetlony jest tak, jak widać na rysunku:



Zadaniem skryptu zapisanego w pliku *pages.php* jest wczytanie nagłówków, menu, formularza i stopki. W dalszej części rozdziału jeszcze do niego wrócimy.

Teraz utwórz katalog */ww.admin/pages*, a w nim stwórz plik o nazwie */ww.admin/pages/forms.php*:

```
<h2>TU BĘDZIE FORMULARZ</h2>
```

Możemy przejść do tworzenia menu strony. Utwórz plik */ww.admin/pages/menu.php* o następującej treści:

```
<?php
echo '<div id="pages-wrapper">';
$rs=dbAll('select id,type,name,parent from pages order by ord,name');
$pages=array();
foreach($rs as $r){
  if(!isset($pages[$r['parent']]))$pages[$r['parent']]=array();
  $pages[$r['parent']][]=$r;
}
function show_pages($id,$pages){
  if(!isset($pages[$id]))return;
  echo '<ul>';
  foreach($pages[$id] as $page){
    echo '<li id="page_'.$page['id'].'">'
      .'<a href="pages.php?id='.$page['id'].'"'>'
      .'<ins> </ins>'.htmlspecialchars($page['name'])
      .'</a>';
    show_pages($page['id'],$pages);
    echo '</li>';
  }
  echo '</ul>';
}
show_pages(0,$pages);
echo '</div>';
```

Skrypt ten generuje nieuporządkowaną listę stron.

Zwróć uwagę na użycie pola `parent`. W większości witryn strony są rozmieszczone w strukturze hierarchicznej z relacjami typu „rodzic – dziecko". Każda strona jest „dzieckiem" jakiejś innej strony lub „korzenia" witryny. W polu `parent` zapisany jest identyfikator strony będącej nadrzędną wobec danej strony.

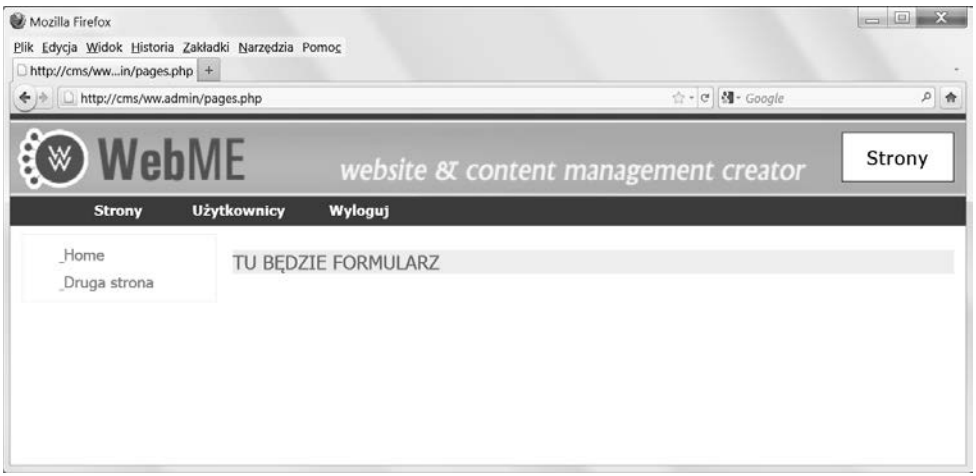
Stronę główną (tzn. tę, która zostanie wyświetlona po wpisaniu w przeglądarce adresu *http://cms/* bez wskazania konkretnej strony) można wyznaczyć na dwa sposoby.

1. Tworząc w bazie danych jedną stronę z polem `parent` o wartości `0`, oznaczającej, że ta strona nie ma rodzica — to ta strona zostanie wyświetlona po wpisaniu w oknie przeglądarki adresu *http://cms/*. Wówczas wszystkie strony typu *http://cms/nazwastrony* będą miały w polu `parent` identyfikator strony, która ma w tym polu wartość `0`.
2. Tworząc wiele stron z polem `parent` o wartości `0`, z których każda będzie tzw. stroną najwyższego poziomu. Jedna z nich musiałaby wówczas mieć specjalną wartość w polu `special`, wskazującą, że to jest strona główna. W takim przypadku strony typu *http://cms/nazwastrony* będą miały rodzica `0`, a strona *http://cms/* może być przechowywana w dowolnym miejscu bazy danych.

Pierwsze rozwiązanie ma jedną poważną wadę: aby zmienić stronę główną, trzeba przenieść bieżącą stronę główną pod jakąś inną stronę (albo ją usunąć), a następnie przesunąć wszystkie strony potomne bieżącej strony głównej tak, aby miały w polu `parent` identyfikator nowej strony głównej. Może to być bardzo skomplikowane, jeśli nowa strona główna ma już jakieś podstrony — zwłaszcza jeśli nazwy niektórych się powtarzają.

Drugie rozwiązanie jest o wiele lepsze, ponieważ pozwala na bezproblemową zmianę strony głównej.

Teraz nasz serwis wygląda tak, jak widać na poniższym rysunku (do budowy użyte zostały strony wprowadzone do bazy w rozdziale 1.).



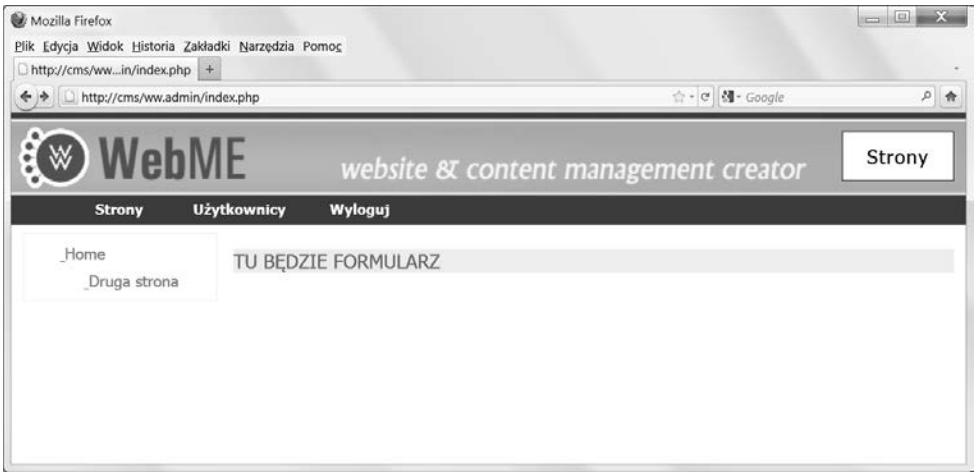
Widok hierarchii stron

Zmodyfikujemy nieco bazę danych, aby można było wyświetlić hierarchię stron.

W tym celu otwórz konsolę MySQL i zmień definicję drugiej strony tak, aby w jej polu `parent` znajdował się identyfikator strony głównej:

```
mysql> select id,name,parent from pages;
+----+-------------+--------+
| id | name        | parent |
+----+-------------+--------+
| 24 | Home        |      0 |
| 25 | Druga strona|      0 |
+----+-------------+--------+
2 rows in set (0.00 sec)
mysql> update pages set parent=24 where id=25;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Po dokonaniu zmian odśwież stronę w przeglądarce:



Jak widać, wpis **drugiej strony** jest nieco wcięty, gdyż jest ona potomkiem **strony głównej** i w kodzie HTML znajduje się w zagnieżdżonym elemencie `ul`.

Widok ten możemy poprawić.

Istnieje wtyczka jQuery o nazwie `jstree`, która przetwarza drzewa stworzone przy użyciu elementów `ul` w interfejsy podobne do interfejsów menedżerów plików.

Ponadto pozwala ona na przeciąganie węzłów drzewa i wiązanie zdarzeń z kliknięciami tych węzłów.

Skorzystamy z tych funkcji później, kiedy będziemy pracować nad mechanizmem tworzenia i usuwania stron oraz zmiany hierarchii stron poprzez ich przeciąganie.

Utwórz w katalogu głównym witryny folder o nazwie */j/*.

Przypomnę, że w rozdziale 1. została podjęta decyzja, iż wszystkie nazwy katalogów w systemie, które są dłuższe niż dwa znaki, będą zawierać kropkę.

Powodem, dla którego katalog ten nazwałem */j/* zamiast np. */ww.javascript/*, jest mała liczba znaków pozwalająca zaoszczędzić kilka bajtów transferu. To może być ważne, jeśli z serwisu będą korzystać użytkownicy dysponujący łączami o niskich parametrach, takich jakie są np. w smartfonach.

Skrócenie jednej nazwy może na niewiele się zda, ale jeśli wyrobisz sobie nawyk stosowania takich krótkich nazw, to skumulowany efekt pozwoli zaoszczędzić sekundę lub dwie przy pobieraniu stron.

Jedna drobna optymalizacja nie ma znaczenia, ale połączenie dużej liczby takich optymalizacji to już całkiem co innego.

W każdym bądź razie tworzymy folder */j/* i pobieramy skrypt `jstree` ze strony *http://jstree.com* oraz wypakowujemy zawartość archiwum tak, aby ścieżka do pliku *jquery.tree.js* była następująca: */j/jquery.jstree/jquery.tree.js*.

Do budowy tego CMS-a użyta została wersja 0.9.9a skryptu.

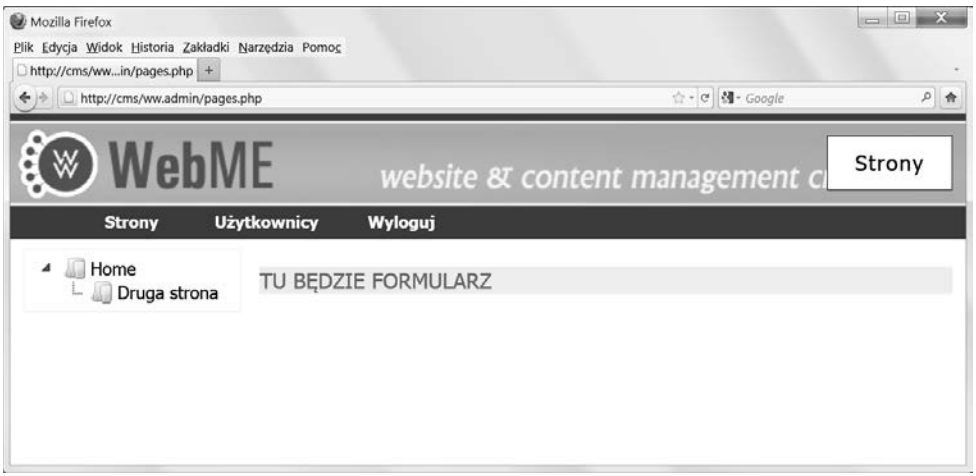
Teraz otwórz plik */ww.admin/pages/menu.php* i dodaj do niego poniższe wiersze kodu zaznaczone pogrubieniem.

```
<script src="/j/jquery.jstree/jquery.tree.js"></script>
<script src="/ww.admin/pages/menu.js"></script>
<?php
```

Następnie utwórz plik */ww.admin/pages/menu.js*:

```
$(function(){
  $('#pages-wrapper').tree();
});
```

Od razu na ekranie pojawi nam się piękne drzewo reprezentujące hierarchię stron serwisu, jak pokazano na poniższym rysunku:



Nazwy stron można przeciągać w różne miejsca, które w czasie operacji są zaznaczane specjalnym symbolem, tak jak widać na dwóch kolejnych zrzutach ekranu:

Zanim przejdziemy do rzeczywistej pracy nad stronami, dokonamy jeszcze jednej poprawki tego menu. Stworzymy przycisk pozwalający dodawać strony najwyższego poziomu oraz zaimplementujemy funkcję zapamiętywania zdarzeń przeciągania stron, aby miały one trwały skutek.

Otwórz plik */ww.admin/pages/menu.js* i zmień jego zawartość na następującą:

```
$(function(){
 $('#pages-wrapper').tree({
  callback:{
   onchange:function(node,tree){
    document.location='pages.php?action=edit&id='
      +node.id.replace(/.*_/,'');
   },
   onmove:function(node){
    var p=$.tree.focused().parent(node);
    var new_order=[],nodes=node.parentNode.childNodes;
    for(var i=0;i<nodes.length;++i)
      new_order.push(nodes[i].id.replace(/.*_/,''));
    $.getJSON('/ww.admin/pages/move_page.php?id='
      +node.id.replace(/.*_/,'')+'&parent_id='
      +(p==-1?0:p[0].id.replace(/.*_/,''))
      +'&order='+new_order);
   }
  }
 });
 var div=$(
   '<div><i>Prawy przycisk myszy wyświetla opcje</i><br /><br /></div>');
 $('<button>Dodaj stronę główną</button>')
  .click(pages_add_main_page)
  .appendTo(div);
 div.appendTo('#pages-wrapper');
});
function pages_add_main_page(){}
```

Za pomocą tego kodu wzbogaciliśmy drzewo stron o kilka funkcji.

Po pierwsze, została dodana funkcja zwrotna `onchange`.

Kliknięcie węzła drzewa (nazwy jednej ze stron) powoduje przekierowanie przeglądarki na stronę *pages.php?edit=* z identyfikatorem kliknięte strony na końcu. Podczas tworzenia elementu `ul` reprezentującego drzewo dla każdego podelementu `li` zdefiniowaliśmy identyfikator w taki sposób, że nazwa strony o identyfikatorze `24` w bazie danych znajduje się na stronie w elemencie `li` o identyfikatorze `page_24`.

Zatem kiedy zostanie kliknięty jeden z węzłów (element `li`), musimy tylko usunąć część `page_` identyfikatora i reszty łańcucha użyć do otwarcia strony *pages.php*.

Po drugie, dodaliśmy funkcję zwrotną o nazwie `onmove`. Jej wywołanie następuje po zakończeniu operacji przeciągania elementu.

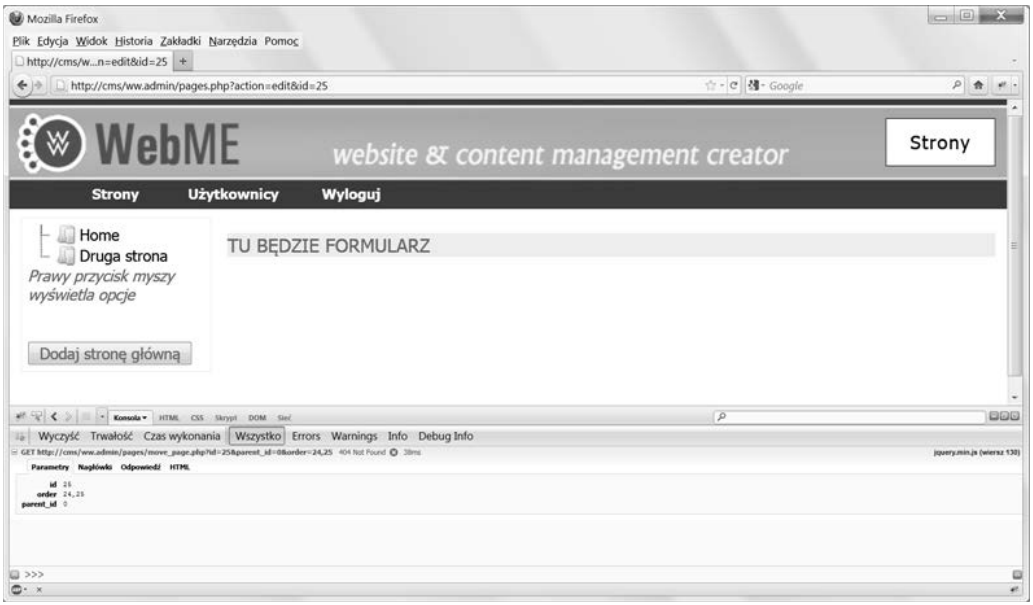
Jej treść jest nieco bardziej skomplikowana niż poprzedniej. Pobiera identyfikator nowego rodzica i tworzy tablicę, w której zapisuje identyfikatory wszystkich bezpośrednich podstron. Następnie wszystkie te dane wysyła do skryptu */ww.admin/pages/move_page.php*, który niebawem stworzymy.

Na końcu dodaliśmy informację, że kliknięcie prawym przyciskiem myszy powoduje wyświetlenie dodatkowych opcji, którymi zajmiemy się później, oraz utworzyliśmy przycisk do tworzenia stron najwyższego poziomu, którego obsługą również zajmiemy się w dalszej części tego rozdziału. Aby kod ten nie powodował błędów, konieczne jest dodanie atrapy funkcji. Później zastąpimy ją prawdziwą definicją.

Aranżacja hierarchii stron

W tej chwili przeciągnięcie strony w inne miejsce na drzewie powoduje wykonanie wywołania Ajax do skryptu */ww.admin/pages/move_page.php* i przekazanie do niego pewnych informacji.

Oto zrzut ekranu (widoczne okienko dodatku Firebug), na którym widać, jakie dane są przesyłane podczas takiej operacji przeciągania:



W tym wywołaniu zostały przesłane następujące informacje: identyfikator strony `25`, identyfikator nowego rodzica `0` oraz nowa kolejność stron, których rodzic ma identyfikator `0` (`25,24`).

Czas na napisanie skryptu */ww.admin/pages/move_page.php*:

```
<?php
require '../admin_libs.php';
$id=(int)$_REQUEST['id'];
$to=(int)$_REQUEST['parent_id'];
$order=explode(',',$_REQUEST['order']);
dbQuery('update pages set parent='.$to.' where id='.$id  );
for($i=0;$i<count($order);++$i){
  $pid=(int)$order[$i];
  dbQuery("update pages set ord=$i where id=$pid");
  echo "update pages set ord=$i where id=$pid\n";
}
```

To proste! Ten skrypt zapisuje te informacje, które są przesyłane w wywołaniu Ajax.

Administracja stronami

Mamy już listę stron serwisu. Czas na dodanie funkcji umożliwiających ich modyfikowanie.

Kod formularza do tworzenia stron jest dość długi, dlatego będę go pokazywać i omawiać po kawałku. Otwórz plik */ww.admin/pages/forms.php* i zastąp jego zawartość poniższym kodem:

```
<?php
if(isset($_REQUEST['id']))$id=(int)$_REQUEST['id'];
else $id=0;
if($id){  // sprawdzenie, czy strona o takim identyfikatorze istnieje
  $page=dbRow("SELECT * FROM pages WHERE id=$id");
  if($page!==false){
    $page_vars=json_decode($page['vars'],true);
    $edit=true;
  }
}
if(!isset($edit)){
  $parent=isset($_REQUEST['parent'])?
    (int)$_REQUEST['parent']:0;
  $special=0;
  if(isset($_REQUEST['hidden']))$special+=2;
  $page=array('parent'=>$parent,'type'=>'0','body'=>'',
    'name'=>'','title'=>'','ord'=>0,'description'=>'',
    'id'=>0,'keywords'=>'','special'=>$special,
    'template'=>'');
  $page_vars=array();
  $id=0;
  $edit=false;
}
```

Kod ten inicjuje dwie tablice: `$page` do przechowywania najważniejszych informacji o stronie i `page_vars` na dane nienależące do głównej tabeli danych stron, np. informacje zapisane przez jakąś wtyczkę.

Jeśli w adresie URL zostanie wysłany identyfikator, to skrypt wczyta dane odpowiadającej mu strony.

Przykładowo: jeśli dodam do kodu wiersz `var_dump($page);`, a następnie wpiszę w przeglądarce adres */ww.admin/pages.php?action=edit&id=25* (taka strona znajduje się w mojej bazie danych), to zobaczę następujący wynik:

Zostały wyświetlone wszystkie informacje o stronie, jakie znajdują się w tabeli bazy danych.

Gdyby w adresie URL został przekazany identyfikator `0` lub jakikolwiek inny, którego nie ma aktualnie w bazie danych, to tablica `$page` i tak zostałaby zainicjowana, ale pustymi wartościami.

Ponieważ strony mogą być bardzo skomplikowane, zwłaszcza gdy będzie wiele ich rodzajów dodanych przez wtyczki, podzielimy formularz na kilka zakładek.

Formularz tworzenia stron normalnego typu będzie się składał z dwóch zakładek — opcji ogólnych i opcji zaawansowanych.

Na pierwszej z wymienionych zakładek znajdą się często zmieniane informacje, czyli np. nazwa strony, treść strony itp.

Natomiast na zakładce opcji zaawansowanych umieścimy rzadziej używane elementy, takie jak znaczniki `meta`, szablony itp. Zastosujemy nazwę *Zaawansowane* dlatego, iż nazwa *Rzadko używane opcje* byłaby zbyt długa, oraz dlatego, że niektórzy administratorzy mogą nie wiedzieć, do czego służą niektóre z zawartych na tej karcie opcji.

Dodajemy menu z zakładkami do pliku */ww.admin/pages/forms.php*:

```
// { jeśli strona jest niewidoczna w nawigacji, to ma zostać wyświetlony stosowny komunikat
if($page['special']&2)
  echo '<em>UWAGA: ta strona jest aktualnie niewidoczna w nawigacji. Aby ją
  ↪przywrócić do widoku, skorzystaj z opcji na karcie Zaawansowane.</em>';
// }
echo '<form id="pages_form" method="post">';
echo '<input type="hidden" name="id" value="',$id,'" />'
    ,'<div class="tabs"><ul>'
    ,'<li><a href="#tabs-common-details">Opcje ogólne</a></li>'
    ,'<li><a href="#tabs-advanced-options">Zaawansowane</a></li>'
    ;
// tu będą zakładki wtyczek
echo '</ul>';
```

Jeśli dana strona nie jest widoczna w menu nawigacyjnym witryny, to nad tym formularzem będzie wyświetlona odpowiednia informacja. Strona nie jest uwzględniana w menu nawigacyjnym wówczas, gdy w polu `special` ma wartość `2`.

Maski bitowe są przydatne w sytuacjach, gdy mamy wartości typu „tak” i „nie” i nie chcemy zajmować całego pola w bazie danych na przechowywanie każdej z nich.

Dalej zaczyna się kod generujący formularz.

Zwróć uwagę na brak parametru `action`. Mimo iż w specyfikacji W3C języka HTML 4.01 parametr ten jest wymagany, to żadna przeglądarka nie wymusza jego stosowania. Kiedy go brak, to przeglądarki do przetwarzania danych z formularza stosują domyślnie ten sam plik.

To samo dotyczy elementu `style`, którego typ jest domyślnie ustawiany na `text/css`, i elementu `script`, którego typ jest domyślnie ustawiany na `javascript`.

Za formularzem mamy menu zakładek zbudowane na bazie elementu listy.

W przedostatnim wierszu kodu znajduje się komentarz informujący o zakładkach wtyczek. Kiedy w dalszej części rozdziału zaczniemy zajmować się wtyczkami, niektóre z nich mogą mieć na tyle dużo opcji, że trzeba będzie je wydzielić do osobnej karty. Zajmiemy się tym później.

Teraz dodamy zakładkę opcji ogólnych (pracujemy cały czas na tym samym pliku):

```
// { opcje ogólne
echo '<div id="tabs-common-details"><table
style="clear:right;width:100%;"><tr>';
// { nazwa
echo '<th width="5%">nazwa</th><td width="23%">
  <input
    id="name" name="name"
    value="',htmlspecialchars($page['name']),'" /></td>';
// }
// { tytuł
echo '<th width="10%">tytuł</th><td width="23%">
  <input
    name="title"
    value="',htmlspecialchars($page['title']),'" /></td>';
// }
// { url
echo '<th colspan="2">';
if($edit){
  $u='/'.str_replace(' ','-',$page['name']);
  echo '<a style="font-weight:bold;color:red" href="',$u,'"
target="_blank">PODGLĄD</a>';
}
else echo ' ';
echo '</th>';
// }
echo '</tr><tr>';
// { typ
echo '<th>typ</th><td><select name="type"><option value="0">normalna</option>';
// tutaj wstaw typy dodane jako wtyczki
echo '</select></td>';
// }
// { rodzic
```

```
echo '<th>rodzic</th><td><select name="parent">';
if($page['parent']){
  $parent=Page::getInstance($page['parent']);
  echo '<option value="',$parent->id,'">',htmlspecialchars($parent->name),
  ↪'</option>';
}
else echo '<option value="0"> -- ','brak',' -- </option>';
echo '</select>',"\n\n",'</td>';
// }
if(!isset($page['associated_date']) || !preg_match('/^[0-9]{4}-[0-9]{2}-
↪[0-9]{2}$/',$page['associated_date']) || $page['associated_date']==
↪'0000-00-00')$page['associated_date']=@date('Y-m-d');
echo '<th>Data</th><td><input name="associated_date" class="date-human"
↪value="',$page['associated_date'],'" /></td>';
echo '</tr>';
// }
// { dane dotyczące typu strony
echo '<tr><th>treść</th><td colspan="5">';
echo '<textarea name="body">',htmlspecialchars($page['body']),'</textarea>';
echo '</td></tr>';
// }
echo '</table></div>';
// }
```

Ten skrypt wyświetla wartości najczęściej zmienianych pól tabeli bazy danych:

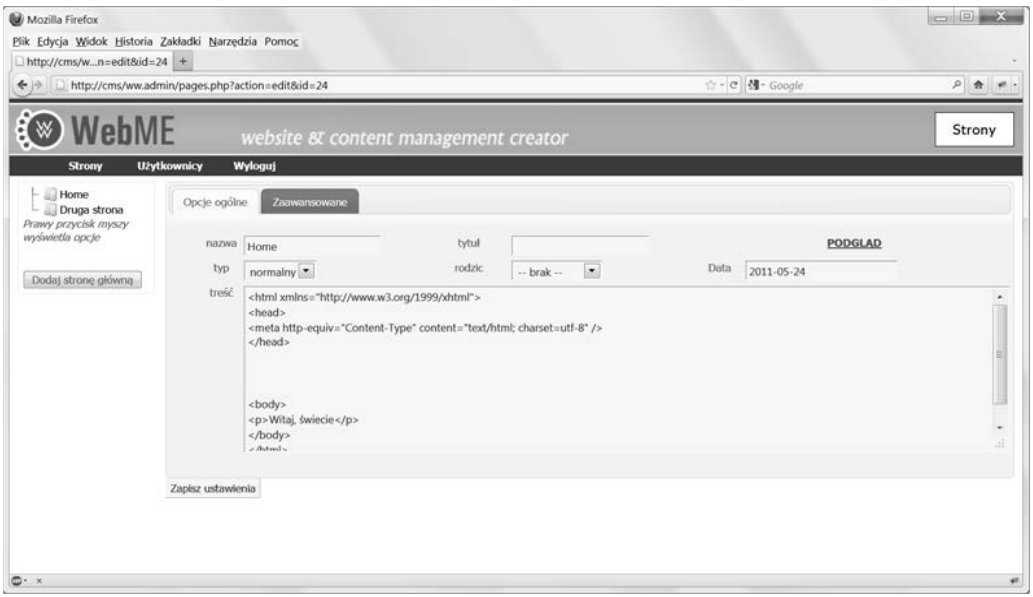
- `name` (nazwa)
- `title` (tytuł)
- `type` (typ)
- `parent` (rodzic)
- `associated_date` (data)
- `body` (treść)

Kilkoma z nich jeszcze się zajmiemy, kiedy skończymy z formularzem. A na razie należy poczynić kilka uwag na temat formularza i zawartych w nim opcji:

- **URL**: podczas modyfikowania strony dobrze jest mieć ją wyświetloną w innym oknie lub na innej karcie. Aby to umożliwić, dodaliśmy odnośnik do strony w widoku przeznaczonym dla użytkowników. Jego kliknięcie powoduje otwarcie nowej karty lub nowego okna.
- **Typ**: domyślnie stronom nadawany jest typ normalny, który jak na razie jest jedyną opcją do wyboru. Później, kiedy dodamy wtyczki, tych możliwości będzie więcej.
- **Rodzic**: to strona, w której znajduje się aktualnie redagowana strona. Wyświetlony jest tylko bieżący rodzic i nie ma żadnych dodatkowych opcji. Jest ważny powód, aby tak było; stanie się on jasny po zakończeniu pracy nad formularzem.

- **Data**: ze stroną związanych jest kilka dat. Wewnętrznie rejestrowane są daty utworzenia i ostatniej modyfikacji (które mogą być przydatne wtyczkom), ale czasami administrator chce określić datę strony. Sytuacja taka może mieć miejsce np. wówczas, gdy strona stanowi część systemu wiadomości. Pole daty jeszcze rozbudujemy, gdy skończymy pracę nad formularzem.
- **Treść**: zawartość, która będzie wyświetlona na froncie strony. W tym polu należy wpisywać zwykły kod HTML. Oczywiście przeciętny administrator niekoniecznie zna język HTML, dlatego trzeba będzie nad tym jeszcze trochę popracować.

Tak teraz wygląda zawartość pierwszej karty (na zrzucie widać już ukończone karty stworzone przy użyciu jQuery — jak to zrobiłem, dowiesz się w dalszej części rozdziału).



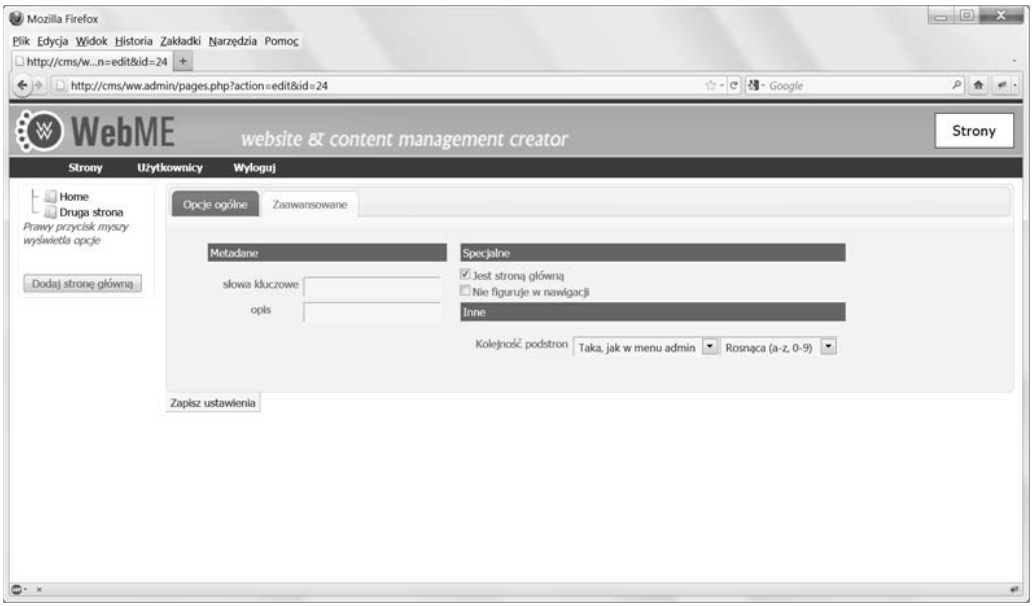
Jak widać, pole daty jest dość duże. Nie jest tak bez powodu, o czym przekonasz się w następnym rozdziale.

Kod źródłowy drugiej karty będzie nieco krótszy. Dodaj poniższy kod do pliku */ww.admin/pages/forms.php*:

```
// {opcje zaawansowane
echo '<div id="tabs-advanced-options">';
echo '<table><tr><td>';
// { metadane
echo '<h4>Metadane</h4><table>';
echo '<tr><th>słowa kluczowe</th><td>
  <input name="keywords"
    value="',htmlspecialchars($page['keywords']),'"
  /></td></tr>';
```

```
echo '<tr><th>opis</th><td>
  <input name="description"
    value="',htmlspecialchars($page['description']),'"
  /></td></tr>';
  // { szablon
// odpowiedni kod zostanie dodany w następnym rozdziale
// }
echo '</table>';
// }
echo '</td><td>';
// { specjalne
echo '<h4>Specjalne</h4>';
$specials=array('Jest stroną główną',
    'Nie figuruje w nawigacji');
for($i=0;$i<count($specials);++$i){
  if($specials[$i]!=''){
    echo '<input type="checkbox" name="special[',$i,']"';
    if($page['special']&pow(2,$i))echo ' checked="checked"';
    echo ' />',$specials[$i],'<br />';
  }
}
// }
// { inne
echo '<h4>Inne</h4>';
echo '<table>';
// { kolejność podstron
echo '<tr><th>Kolejność podstron</th><td><select name="page_
vars[order_of_sub_pages]">';
$arr=array('Taka, jak w menu admin','Alfabetyczna','by associated date');
foreach($arr as $k=>$v){
  echo '<option value="',$k,'"';
  if(isset($page_vars['order_of_sub_pages']) &&
      $page_vars['order_of_sub_pages']==$k)
      echo ' selected="selected"';
  echo '>',$v,'</option>';
}
echo '</select>';
echo '<select name="page_vars[order_of_sub_pages_dir]">
    <option value="0">Rosnąca (a-z, 0-9)</option>';
echo '<option value="1"';
if(isset($page_vars['order_of_sub_pages_dir']) &&
    $page_vars['order_of_sub_pages_dir']=='1')
    echo ' selected="selected"';
echo '>Malejąca (z-a, 9-0)</option></select></td></tr>';
// }
echo '</table>';
// }
echo '</td></tr></table></div>';
// }
```

Nie ma tu wiele do wyjaśniania. Można by było dodać jeszcze kilka zaawansowanych opcji, które byłyby przydatne, gdyby system był bardziej rozwinięty (dodane wtyczki, ukończony system szablonów i motywów itd.).



Najpierw dodane zostały **pola tekstowe** na **słowa kluczowe** i opis (elementy `meta keywords` i `description`). Większość ludzi pozostawia je nietknięte i dlatego znalazły się na karcie *Zaawansowane*.

Szablony i motywy zostaną dodane w następnym rozdziale, a na razie w ich zastępstwie w kodzie znajduje się tylko komentarz informujący o przyszłych planach.

Dalej znajduje się lista „specjalnych" cech. Na razie są tylko dwie: pole wyboru pozwalające oznaczyć stronę jako główną oraz pole umożliwiające usunięcie strony z nawigacji.

Na końcu znajdują się dwie listy rozwijane służące do określania kolejności podstron bieżącej strony w nawigacji frontowej. Możesz przykładowo zechcieć posortować alfabetycznie listę autorów albo aby nowe elementy pojawiały się w kolejności malejącej według daty. Najczęściej jednak stosuje się tę samą kolejność, co w menu administracyjnym (można ją zmienić, przeciągając nazwy stron w menu nawigacyjnym po lewej stronie).

Czas na dokończenie formularza i dodanie kodu odpowiedzialnego za wyświetlanie zakładek.

Warto dodać jeszcze jedną sekcję do formularza — niektóre wtyczki mogą wymagać dodawania własnych zakładek, ale do tego wrócimy kiedy indziej.

Dodaj do pliku */ww.admin/pages/forms.php* poniższy kod:

```
echo '</div><input type="submit" name="action" value="',
    ($edit?'Zapisz ustawienia':'Wprowadź ustawienia')
    ,'" /></form>';
echo '<script>window.currentpageid='.$id.';</script>';
echo '<script src="/ww.admin/pages/pages.js"></script>';
```

Ponadto utwórz następujący plik */ww.admin/pages/pages.js*:

```
$(function(){
  $('.tabs').tabs();
});
```

Zmienna `window.currentpageid` będzie potrzebna w następnym podrozdziale.

Podstawowa wersja formularza jest już gotowa.

Teraz zajmiemy się rozszerzaniem funkcjonalności uprzednio wyróżnionych pól.

Asynchroniczne wypełnianie listy rodziców

W bardzo dużych serwisach proces wczytywania formularza może być bardzo powolny z powodu długiej listy rozwijanej *rodzic* informującej serwer, której strony potomkiem jest dana strona.

Jeśli listę tę będzie się wypełniać podczas wczytywania formularza, to ilość kodu HTML do pobrania może być bardzo pokaźna.

Rozwiązanie tego problemu opracowałem na potrzeby mojej poprzedniej książki (*jQuery 1.3 with PHP*). Zaimplementowałem je w postaci wtyczki jQuery, której teraz użyjemy.

Pobierz wtyczkę `remoteselectoptions` ze strony *http://plugins.jquery.com/project/remoteselect-options* i wypakuj ją do katalogu */j/*.

Wtyczka ta działa następująco: przy wczytywaniu strony do listy rozwijanej pobierana jest tylko jedna opcja, a pozostałe zostają dobrane dopiero wtedy, gdy są rzeczywiście potrzebne, tzn. gdy zostanie kliknięty element listy.

Aby zmusić ją do takiej współpracy z polem *rodzic*, otwórz plik */ww.admin/pages/pages.js* i zmień jego zawartość następująco:

```
$(function(){
  $('.tabs').tabs();
  $('#pages_form select[name=parent]').remoteselectoptions({
    url:'/ww.admin/pages/get_parents.php',
    other_GET_params:currentpageid
  });
});
```

Jako że ta wtyczka może być przydatna w wielu miejscach panelu administracyjnego, dodamy ją do pliku */ww.admin/header.php* (wyróżniony fragment):

```
<script src="http://ajax.googleapis.com/ajax/libs
    /jqueryui/1.8.0/jquery-ui.min.js"></script>
<script src="/j/jquery.remoteselectoptions/jquery.remoteselectoptions.js">
↪</script>
<link rel="stylesheet" href="http://ajax.googleapis.com/ajax
    /libs/jqueryui/1.8.0/themes/south-street/jquery-ui.css"
    type="text/css" />
```

Z treści pliku *pages.js* wynika, że do budowy listy nazw stron potrzebny jest jeszcze jeden plik — */ww.admin/pages/get_parents.php*. Oto jego zawartość:

```
<?php
require '../admin_libs.php';
function page_show_pagenames($i=0,$n=1,$s=0,$id=0){
  $q=dbAll('select name,id from pages where parent="'.$i.'" and id!="'.$id.'"
  ↪order by ord,name');
  if(count($q)<1)return;
  foreach($q as $r){
    if($r['id']!=''){
      echo '<option value="'.$r['id'].'" title=
      ↪"'.htmlspecialchars($r['name']).'"';
      echo($s==$r['id'])?' selected="selected">':'>';
      for($j=0;$j<$n;$j++)echo ' ';
      $name=$r['name'];
      if(strlen($name)>20)$name=substr($name,0,17).'...';
      echo htmlspecialchars($name).'</option>';
      page_show_pagenames($r['id'],$n+1,$s,$id);
    }
  }
}
$selected=isset($_REQUEST['selected'])
    ?$_REQUEST['selected']:0;
$id=isset($_REQUEST['other_GET_params'])
    ?(int)$_REQUEST['other_GET_params']:-1;
echo '<option value="0"> --  brak  -- </option>';
page_show_pagenames(0,0,$selected,$id);
```

Wtyczka `remoteselectoptions` wysyła do tej strony żądanie z dwoma parametrami: identyfikatorem aktualnie wybranego rodzica i identyfikatorem aktualnej strony.

Powyższy skrypt tworzy listę opcji oraz uniemożliwia administratorowi wstawienie strony w niej samej ani w stronie, która znajduje się niżej od niej w hierarchii. Gdyby tak się stało, strona zniknęłaby z wszystkich menu nawigacyjnych, także w panelu administracyjnym.

W tej chwili oznacza to, że jedyne dostępne opcje to albo *brak* (jeśli strona jest najwyższego poziomu), albo *Druga strona*, tak jak w naszym przykładzie. Aktualnie są tylko dwie strony i oczywiście nie można umieścić strony *Home* pod stroną *Home*.

Wykonaliśmy kawał dobrej roboty, więc możesz sobie zrobić przerwę, aby nabrać sił na dokończenie mechanizmu tworzenia stron w następnym rozdziale.

Podsumowanie

W tym rozdziale zostały stworzone podstawy systemu zarządzania stronami. Utworzyliśmy formularz do zarządzania nimi oraz za pomocą narzędzi jQuery zaimplementowaliśmy prosty mechanizm zamiany stron miejscami i optymalizacji wyświetlania długich list opcji.

W następnym rozdziale dokończymy sekcję zarządzania stronami i zbudujemy proste frontowe menu nawigacyjne, dzięki któremu będzie można wchodzić na różne strony.

Skorowidz

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

Ż

Projektowanie **systemów CMS** przy użyciu **PHP i jQuery**

Na rynku systemów do zarządzania treścią jest ogromny wybór różnorodnych rozwiązań, od Joomli! zaczynając, a na Drupalu i Wordpressie kończąc. Warto jednak zastanowić się, czy nie lepiej stworzyć własne, skrojone na miarę rozwiązanie, idealnie dopasowane do potrzeb i możliwości. Zamiast przedzierać się przez zawiłą konfigurację, przygotowywać szablony i próbować pogodzić ze sobą wtyczki, być może lepiej tę energię spożytkować na napisanie własnego CMS-a?

Jeżeli zdecydujesz się na to, w niniejszej książce znajdziesz szczegółowy przewodnik, jak dokonać tego przy użyciu najpopularniejszych narzędzi: łącząc język PHP i bibliotekę jQuery. W trakcie lektury dowiesz się, jak zaprojektować jądro swojego nowego systemu, moduł zarządzania użytkownikami oraz własny mechanizm szablonów. Ponadto nauczysz się zarządzać treścią, tworzyć hierarchię stron oraz edytować je przy użyciu wygodnych narzędzi, takich jak CKEditor. Na koniec zbudujesz instalator swojego CMS-a oraz wzbogacisz go o obsługę wtyczek i widżetów. Książka ta jest pozycją obowiązkową dla każdego webmastera pragnącego stworzyć własny, unikatowy system zarządzania treścią.

Zbuduj swój autorski system CMS, wykorzystując niezastąpiony duet PHP i jQuery!

Główne zagadnienia omówione w książce:

- jądro systemu CMS,
- panel administracyjny,
- struktura bazy danych oraz katalogów,
- zarządzanie użytkownikami — role, rodzaje użytkowników,
- logowanie użytkowników,
- procedura odzyskiwania hasła,
- wyświetlanie listy stron,
- tworzenie, edycja i usuwanie stron,
- wykorzystanie narzędzia CKEditor,
- tworzenie i obsługa szablonów,
- rozszerzanie funkcjonalności za pomocą wtyczek,
- budowa instalatora.

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

ISBN 978-83-246-3365-4

Cena 59,00 zł

Informatyka w najlepszym wydaniu