WYŻSZA SZKOŁA BIZNESU
NATIONAL-LOUIS UNIVERSITY

Złożenie pracy online:
2010-03-01 22:50:16
Kod pracy:
2570

WYDZIAŁ INFORMATYKI
KIERUNEK: INFORMATYKA
SPECJALNOŚĆ: INŻYNIERIA OPROGRAMOWANIA

Robert Podwika
(Nr albumu: 10275*INF/LIC)

System Zarządzania Treścią

Content Management System

Praca inżynierska

Promotor: **dr inż. Bogdan Batko**

Nowy Sącz, 2010

Wstęp

W poniższej pracy inżynierskiej omawiam zagadnienie Systemu Zarządzania Treścią w sieci Web. Zagadnienie jest istotne z uwagi na bardzo dynamiczny rozwój globalnej sieci – Internetu oraz technologii z nim powiązanych. Ludzie coraz częściej korzystają z narzędzi dostarczanych im przez webmasterów w postaci stron WWW. Każda strona WWW zawiera treść, którą trzeba zarządzać. Dlatego tak istotnym jest przyjrzenie się zagadnieniu systemów, które zarządzają informacjami na stronach WWW.

W pracy będę chciał napisać system w pełni oparty o programowanie obiektowe, z wykorzystaniem najnowszych technologii i bibliotek, które ułatwiają programowanie aplikacji webowych.

Praca zawiera pełny opis zastosowanych bibliotek, frameworków, technologii takich jak: Zend Framework, Smarty, Jquery. W pierwszym rozdziale pracy opisuje zagadnienie systemu zarządzania treścią, jego zadań, historii wraz z przykładami. Drugi rozdział poświęcony jest technologiom, które zostały użyte w pracy, z wytłumaczeniem, dlaczego wybrałem te, a nie inne rozwiązania. W trzecim rozdziale opisuje krok po kroku aplikację, którą napisałem, wraz z fragmentami kodu, zrzutami ekranu oraz wyjaśnieniem funkcjonalności. Jeżeli interesują Cię zagadnienia programowania aplikacji internetowych używając, php, koniecznie powinieneś zagłębić się w lekturę dalszych rozdziałów pracy.

1. System zarządzania treścią

System zarządzania treścią(ang. Content Management System) – jest to aplikacja, która pozwala zarządzać serwisem internetowym przez osobę, nieposiadającą wiedzy technicznej. Najczęściej poprzez różnego rodzaju formularze, kreatory zawartości oraz moduły. Modyfikacja treści odbywa się najczęściej poprzez łatwe w obsłudze edytory(bardzo często typu WYSIWYG[1]). CMS są zazwyczaj oparte na językach skryptowych działających po stronie serwera(PHP, ASP, JSP, RUBY, PYTHON) lub też środowisku działającym w platformie FLASH np. FLEX.

1.1 Zadania

System zarządzania treścią ma za zadanie oddzielić warstwę prezentacyjna witryny od zarządzania nią. Osoba, która ma uprawienia do wprowadzania treści(stosownie regulowane poprzez moduł uprawnień) wprowadza dane do systemu, który przetwarza je i zapisuje w bazie danych. W zależności od stopnia zaawansowania CMS, osoba zarządzająca nim może ustawiać elementy, które są generowane dynamicznie. Bloki, sekcje mogą być przestawiane przez administratora lub inną osobę, która jest do tego uprawiona. Warstwa prezentacyjna systemu ma za zadanie przetworzyć w odpowiedni sposób informacje pobrane z systemu oraz wyświetlić je w przyjaznej dla użytkownika formie. Głównym zadaniem systemu zarządzania treścią jest ułatwienie generowania treści, zachowania jej oraz przekazania użytkownikowi.

[1] WYSIWG(ang. What you see is what you get – to co widzisz jest tym co otrzymujesz) jest skrótem stosowanym w informatyce, który określa pewien rodzaj metod i narzędzi pozwalający na edycje treści w czasie rzeczywistym, w taki sposób by treść w edytorze była identyczna lub bardzo zbliżona do tej opublikowanej.

1.2 Historia

Historia systemów zarządzania treścią jest co najmniej tak stara jak historia treści. Odkąd powstała pierwsza treść ludzie szukali sposobów by nią efektywnie zarządzać. Podczas rozwoju cywilizacji ludzie wymyślali coraz lepsze i szybsze sposoby zarządzania treścią. W epoce digitalizacji treść zaczęła być zapisywania na cyfrowych nośnikach pamięci, które począwszy od kart paskowych, taśm magnetycznych zaczęły ewoluować, do twardych dysków, dostępu do danych poprzez World Wide Web, lub też zaawansowane systemy baz danych. Oczywistym stała się potrzeba wykorzystywania coraz to rożnych sposobów na zarządzanie treścią.[2]

1.3 Główne moduły definiujące CMS

Wśród listy modułów, które powinny znaleźć się w systemie zarządzania treścią możemy wyodrębnić:

- Repozytorium danych w systemie plików lub bazie danych
- Separacja zawartości od warstwy prezentacyjnej
- Edycja w trybie WYSIWG
- Kontrola wersji systemu CMS
- Odpowiednie tagowanie meta oraz zaawansowana nawigacja z użyciem np. modułów Apache mod rewrite.
- Wielojęzyczność
- Personalizacja
- Separacja tworzenia treści od jej wyświetlania

[2] Martin Brampton, PHP5 CMS Framework Development, Birmingham, Packt Publishing Ltd. 2008, s. 7 I nast.

1.4 Systemy WCMS

W mojej pracy użyłem systemu WCMS(Web Content Management System), czyli system zarządzania treścią zaimplementowany, jako aplikacja sieci WEB dla zarządzania i przetwarzania stron HTML. System WCMS jest używany do zarządzania oraz kontroli dużych, dynamicznych zbiorów danych(przeważnie są to dokumenty HTML, filmy, pliki FLASH). Głównymi zadaniami systemu WCMS jest dostarczenie funkcji, które umożliwiają efektywne zarządzanie treścią, edycję oraz wszelkie czynności, które składają się na konserwację systemu.

1.4.1 Narzędzia

Narzędzia, które dostarcza system są tak skonstruowane, aby osoby nietechniczne z niewielką wiedzą mogły tworzyć i zarządzać treścią nie napotykając przy tym większych problemów. Większość systemów WCMS wykorzystuje bazę danych by przechowywać treść. Innymi metodami przechowywania zasobów są pliki(głownie XML). By przyspieszyć działanie serwisu, systemy zarządzania treścią wykorzystują mechanizmy pamięci podręcznej oraz kompilacji stron. Gdy informacje z bazy danych nie są często zmieniane stosuje się mechanizm zapamiętywania treści na określony czas. Natomiast, gdy jesteśmy pewni, że treść nie ulegnie zmianie, możemy skorzystać z technologii kompilacji szablonów, którą udostępnia na przykład biblioteka SMARTY, z której skorzystałem w mojej aplikacji.

1.4.2 Cechy WCMS

Charakterystycznymi cechami systemu WCMS są:

- Łatwo edytowalna treść – edycja treści odbywa się za pomocą intuicyjnych edytorów typu WYSISWG. Warstwa prezentacyjna jest oddzielona od innych warstw systemu, co sprawia, że osoba, która edytuje treść nie musi zagłębiać się w logikę systemu.
- Modularna budowa - większość systemu WCMS posiada budowę modularną, co sprawia, że są łatwe w rozbudowie. Moduły mogą być z łatwością dodawane przez różnych programistów, a całość jest zarządzana z poziomu administracyjnego systemu.
- Zgodność z standardami – bardzo ważnym czynnikiem jest zgodność z standardami, którymi zarządza organizacja „World Wide Web Consortium(W3C)".
- Zarządzanie uprawnieniami – w dobrym systemie zarządzania treścią w sieci Web powinien znaleźć się moduł odpowiedzialny za uprawnienia. Z reguły uprawienia są realizowane na zasadzie grup uprawnień, zasobów, do których chcemy uzyskać uprawienia. System, który stworzyłem posiada takie rozwiązanie. Do zarządzania uprawieniami w Zend Framework służy komponent o nazwie Zend Access Control List(Zend_ACL).
- Zarządzanie przepływem treści - przepływ treści jest to proces kreowania treści, która po utworzeniu musi być zaakceptowana przez osobę do tego uprawnioną, która może nanieść poprawki. Na przykład twórca treści wysyła artykuł, który nie jest publikowany, aż do momentu, kiedy zostanie zaakceptowany przez moderatora, administratora lub inną osobę posiadającą stosowne uprawnienia.
- Komentowanie treści - systemy zarządzania treścią bardzo często pozwalają użytkownikom na komentowanie zdjęć, artykułów, plików lub innych składowych strony. Komentowanie treści sprzyja rozwijaniu społeczności na stronie, co prowadzi do wzrostu popularności witryny.
- Dystrybucja zasobów - dobry system powinien zawierać moduły, które ułatwiają użytkownikom integrację z serwisem. Jednym z sposobów, żeby to zrobić są kanały RSS(Really Simple Dyndicaction). W wyższym poziomie integracji użytkowników z serwisem powinny się znaleźć tak zwane Web Service. Są to narzędzia, które służą do

zdalnego wywoływania procedur, które zwracają nam pewną zawartość strony. Takowe narzędzia są bardzo często oparte na protokole RPC(Remote Procedure Call) stworzonym przez firmę SUN. Najczęściej wykorzystywanymi protokołami są XML-RPC(Extensible Markup Language- Reamote Procedure Call) oraz SOAP(Simple Object Acess Protocol). Serwis powinien udostępnić specjalne api dla osób, które chcą skorzystać z usług, które system oferuje.

1.4.3 Typy systemów WCMS

Istnieją trzy główne typy systemu WCMS: zarządzanie treścią offline, online oraz systemy hybrydowe. Te trzy kategorie opisują, w jaki sposób jest generowana treść oraz kiedy jest ona publikowana na stronach.

- Zarządzanie treścią w trybie online - są to systemy, których treść jest generowana w locie. Treść jest tworzona przez użytkowników systemu. Do takich systemów najczęściej należą fora internetowe, blogi, galerie zdjęć, systemy zarządzania plikami i tym podobne. Systemy te dodawane są najczęściej do istniejących stron. Wiele z nich jest udostępniane, jako projekty Open Source na licencjach GPL lub LGPL.
- Zarządzanie treścią w trybie offline - w systemach tego typu treść jest przygotowywana statycznie i potem wgrywana do systemu. Rozwiązanie tego typu możemy spotkać na przykład na stronach firm, które chcą wykorzystać witrynę wyłącznie do celów prezentacyjnych bez interakcji z użytkownikiem.
- Systemy hybrydowe - są to najczęściej spotykane systemy zarządzania treścią w sieci internetowej. Charakteryzują się tym, ze treść jest generowana zarówno statycznie jak i dynamicznie. Języki, które są najczęściej wykorzystywane do tworzenia takich systemów to: JSP, ASP, PHP, ColdFusion, Perl, Python.

1.4.4 Historia oraz najbardziej popularne systemy WCMS

Zarządzanie treścią w sieci Web narodziło się wraz powstaniem stron internetowych. Pierwsze komercyjne systemy WCMS powstały latach ’90. Około roku 2005 nastąpił gwałtowny rozwój systemów, który trwa aż do dziś. Poniższa lista przedstawia popularne systemy zarządzania treścią w sieci Web.

- Agility – http://agilitycms.com
- Apache Lennya – http://lenya.apache.org
- AxCMS.net – http://axcms.net
- Bigace – http://bigace.de
- BitWeaver – http://bitweaver.org
- Bloofox – http://bloofox.com
- Bricolage – http://bricolage.cc
- Campsite – http://campware.org
- Clickability - http://clickability.com
- CMSBox - http://www.cmsbox.com
- CMS Made Simple - http://www.cmsmadesimple.org
- CMSimple http://cmsimple.dk
- CompactCMS - http://www.compactcms.nl
- Concrete5.org – http://concrete5.org
- ConceptCMS – http://conceptcms.com[3]

[3] Źródło: http://ww.cmsstrict.com/pages/cms-list.html (data odczytu 10.01.2010)

2. Narzędzia użyte przy projekcie

W tym rozdziale opisuje narzędzia, które zostały użyte w pracy takie jak JQuery, Zend Framework oraz Smarty.

2.1 Jquery

Jquery to biblioteka JavaScript, która w znacznym stopniu ułatwia programowanie w tym języku. Jest bardzo szybka, zajmuje niewiele miejsca(19 KB). Jquery rozwiązuje za nas problemy różnej interpretacji kodu JavaScript poprzez przeglądarki. Bardzo dużą zaletą jest również zgodność z standardami W3C, wsparcie dla CSS 1-3. Jquery znalazło uznanie wśród takich firm jak: Google, Dell, lub Mozzila.

2.1.1 Cechy

Jquery charakteryzuje się prostotą i lekkością kodu jednakże to nie wszystkie zalety tej biblioteki. Jquery wyróżnia się tym, że:

- programista nie musi martwić się o różną interpretacje JavaScript przez przeglądarki
- Jquery zajmują mniej miejsca niż standardowe rozwiązania z JavaScript
- ogromny zasób funkcji pozwalający w prosty sposób tworzyć zaawansowane aplikacje
- pełne wsparcie dla AJAX
- dostęp do elementów poprzez DOM(Document Object Model)
- dostęp do obiektów poprzez selektory znane z CSS
- bardzo duża społeczność oferująca duże zasoby darmowych rozszerzeń dla biblioteki
- separacja kodu JavaScript od HTML[4]

[4] Źródło: http://jquery.com/ (data odczytu 10.01.2010)

2.2 Zend Framework

Zend Framework jest to Framework do PHP5 stworzony przez firmę Zend(w tej firmie pracują osoby bezpośrednio odpowiedzialne za rozwijanie samego języka PHP) oraz rozwijany w znacznym stopniu poprzez społeczność. Zend Framework jest oparty na prostocie, w pełni wykorzystuje zalety programowania zorientowanego na obiekty. Dzięki narzędziu można tworzyć potężne aplikacje WEB 2.0, które mogą używać API do takich serwisów jak Google, Amazon, Yahoo! Zend Framework charakteryzuje się prostotą rozszerzalności kodu oraz możliwością zaprojektowania dowolnej architektury. Podstawowym celem systemu było sprawienie by pisanie aplikacji w PHP było proste, łatwe i szybsze. Zend Framework dostarcza zestaw klas, które są najczęściej wykorzystywane w aplikacjach PHP. Framework posiada również swój własny stos, dzięki, któremu możemy uzyskać dostęp do zarejestrowanych w nim elementów. Zestaw komponentów i klas, które oferuje nam Zend Framework jest ogromny. Począwszy od zarządzanie portalem, model MVC, zarządzanie sesją, bazą do generowania plików PDF, grafik, mechanizmu pamięci podręcznej, generowania RSS, łączenia się z Web Service, tworzenia Web Service integrację z Jquery i wiele innych ciekawych funkcji[5]

2.2.1 Cechy

Zend Framework cechuje się tym, że:

- Posiada bardzo duży wybór klas i elementów
- Biblioteki Zend mogą być bardzo łatwo rozszerzalne, przeważnie wystarcza dziedziczenie po klasie, którą chcemy rozszerzyć. Komponenty, które nie są częścią Frameworka mogą być łatwo dodane, społeczność osób używających ZF jest bardzo duża.
- Jest spójny. Zend posiada bardzo zwartą strukturę, która sprawia, że programista może bardzo szybko się w nim odnaleźć. Dodanie nowego modułu sprowadza się do stworzenia kontrolera dla tego modułu, umieszczenia w nim akcji(np. wyświetl,

[5] Źródło: http://framework.zend.com/about/overview (data odczytu 10.01.2010)

edytuj, usuń), utworzenia plików widoków dla tych akcji. Zend Framework wie, że gdy adres witryny to http://aquacms.pl/article/edit/id/1 to musi szukać kontrolera ArticleController.php w katalogu „Controllers", w tym kontrolerze musi wywołać metodę editAction(), która może pobrać i przetworzyć parametr $id poprzez $this->_request->getParam('id'); i na końcu zmienne, które są w szablonie ma wyświetlić w pliku edit.html widoku, który szuka w katalogu „views/article/".

- Jakość dokumentacji jest na wysokim poziomie. Zend Framework posiada bardzo dobrą i obszerną dokumentacje klas wraz z przykładami, którą można znaleźć na stronie http://framework.zend.com/
- Framework jest stale aktualizowany. Zwykle co parę tygodni.
- Każdy komponent jest sprawdzany przez grupę profesjonalnych programistów przed dopuszczeniem go do publikacji. Ponadto, komponenty z uwagi na otwarty kod są sprawdzane przez programistów, którzy ich używają.

System zarządzania treścią, który napisałem jest oparty o model MVC, który jest wspierany przez Zend Framework poprzez klasy: Zend_Application, Zend_Application_Bootstrap, Zend_Application_Module, Zend_Application_Resource, Zend_Controller_Front, Zend_Controller_Action, Zend_Controller_Dispatcher, Zend_Controller_Plugin, Zend_Controller_Router, Zend_Form, Zend_Layout, Zend_View, Zend_View_Filter, Zend_View_Helper.

Do obsługi bazy danych użyłem adapterów. Poziom abstrakcji sprawia, iż możliwe jest wykorzystanie różnych silników baz danych w systemie. Do obsługi bazy danych użyte zostały klasy: Zend_Db, Zend_Db_Adapter, Zend_Db_Profiler, Zend_Db_Select, Zend_Db_Table. Zend Framework wspiera również wielojęzyczność. Poprzez klasę Zend_Locale, automatycznie wykryłem język przeglądarki, której używa osoba przebywająca na stronie CMS-a.

Procesy autoryzacji, autentykacji są zarządzane poprzez listy dostępowe(klasa Zend_ACL) oraz klasę obsługującą autoryzacje(Zend_Auth).

2.3 Smarty

Smarty jest to technologia odpowiedzialna za warstwę prezentacji witryny oparty o system szablonów. Biblioteka Smarty dostarcza programiście oraz osobie, która odpowiada za warstwę prezentacyjną serwisu technologie potrzebne do zautomatyzowania oraz oddzielenia kodu PHP od kodu HTML aplikacji.

2.3.1 Cechy

Podstawowe cechy SMARTY to:

- Pamięć podręczna - biblioteka posiada system cachowania szablonów. Programista ma opcje zdefiniowania, które pliki HTML lub części stron powinny zostać poddane mechanizmowi pamięci podręcznej.
- Bezpieczeństwo - szablony HTML, nie zawierają kodu PHP. Dlatego też, osoba, która na przykład tnie grafikę dla danego szablonu lub projektuje stronę, nie musi znać PHP by to zrobić.
- Łatwość użycia - szablony nie zawierają kodu PHP. Składnia smarty niewiele różni się od kodu HTML. Szablon wygenerowany nie różni się w znaczącym stopniu kodem HTML od tego zawartego w oryginalnym pliku.
- Modyfikatory zmiennych - zawartość zmiennych może być łatwo zmodyfikowana poprzez funkcje na przykład zmiana łańcucha znaków na duże lub małe, formatowanie daty, generowanie segmentów kodu takich jak tabele, okienka pop-up, listy i tym podobne. Modyfikatory umożliwiają również iterowanie tablic z danymi, formatowanie oraz filtrowanie tekstu.
- Filtry - system umożliwia pełną filtrację szablonów przed po oraz po wygenerowaniu szablonu.
- Wtyczki - system Smarty może być łatwo rozszerzalny za pomocą wtyczek. Pluginy mogą być znalezione na stronie smarty, wiele wtyczek zostało stworzone przez społeczność internautów.

- System wykrywania błędów - programista może łatwo zobaczyć, jakie zmienne są w tym momencie dostępne w szablonie.[6]

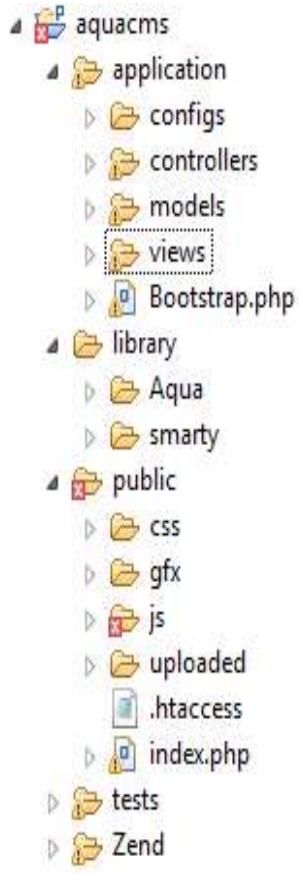
Smarty jest bardzo często wybierane przez programistów, ponieważ posiada wiele zalet i ułatwia prace. Główną zaletą smarty jest podział pracy pomiędzy programistów oraz osoby, które zajmują się Frontend. Programiści nie ingerują w wygląd aplikacji, a zajmują się tylko stroną logistyczną. Osoby, które zajmują się wyglądem nie muszą znać logiki systemu i co najważniejsze nie zepsują kodu aplikacji. Smarty posiada bardzo wiele wbudowanych funkcji odpowiedzialnych za bezpieczeństwo, toteż prawdopodobieństwo narażenia systemu na lukę w bezpieczeństwie poprzez designera jest bardzo niskie.

[6] Źródło: http://www.smarty.net/whyuse.php (data odczytu 10.01.2010)

3. System Aqua Cms

System, który napisałem nazwałem AquaCms. Jest on w pełni oparty o strukturę MVC. Do napisania go użyłem technologii, które opisałem w wcześniejszej części pracy.

3.1 Struktura katalogów



System zarządzania treścią, który napisałem posiada strukturę MVC(Model-Widok-Kontroler). Kontrolery umieszczone są w katalogu controllers. Modele, w katalogu models oraz widoki w katalogu views. W katalogu application oprócz wspomnianych wyżej rzeczy znajdują się katalogi configs, który zawiera pliki konfiguracyjne aplikacji oraz plik bootstrap.php, który odpowiada za start systemu. W Zend Framework klasy nazywane są w zależności od tego, w jakim katalogu się znajdują. Przykładowo, jeżeli szukamy klasy Aqua_Registry_Example powinniśmy szukać w katalogu library/Aqua/Registry/Example.php Wprowadzenie takiego sposobu nazewnictwa plików i klas sprawia, iż kod jest bardziej zwarty oraz nie musimy długo szukać plików klas, których potrzebujemy. Katalog library zawiera biblioteki, które zostały użyte w systemie. W podkatalogu Smarty znajduje się biblioteka smarty pobrana z strony http://smarty.com

Rys. 1. Struktura katalogów portalu, źródło: opracowanie własne

3.2 Rejestr

Klasa Aqua Registry dziedziczy z Zend_Registry, który jest kontenerem trzymającym różne wartości przez aplikację. Poprzez zapisanie wartości w Zend_Registry, mamy do niej dostęp w całej aplikacji. Stałe wartości CONFIG, DB, SESSION, USER zostały użyte w aplikacji do zapisania kolejno konfiguracji, połączenia z baza danych, sesji oraz danych użytkownika.

```php
<?php
    class Aqua_Registry extends Zend_Registry
    {
        const CONFIG     =    1;
        const DB         =    2;
        const SESSION    =    3;
        const USER       =    4;
    }
?>
```

Rys. 2. Klasa Aqua_Registry, źródło: opracowanie własne

Przykład użycia rejestru do zapisania danych w pliku startowym systemu

```php
public function run()
{
   $siteConfig = new Zend_Config_Ini(APPLICATION_PATH.'/configs/site.ini');
   Zend_Session::rememberMe($siteConfig->session->expiration_time);
   Zend_Session::start();
   Aqua_Registry::set(Aqua_Registry::CONFIG,$siteConfig);
   $session = new Zend_Session_Namespace($siteConfig->session->namespace-
>user);
   Aqua_Registry::set(Aqua_Registry::SESSION, $session);
   $locale = new Zend_Locale(Zend_Locale::BROWSER);
   $db = Zend_Db::factory($siteConfig->database);
   $db->getConnection();
   Aqua_Registry::set(Aqua_Registry::DB,$db);
   Zend_DB_Table_Abstract::setDefaultAdapter($db);

   $session = Aqua_Registry::get(Aqua_Registry::SESSION);
   parent::run();
 }
```

3.3 Konfiguracja aplikacji internetowej

Dane konfiguracyjne aplikacji przechowywane są w plikach o rozszerzeniach ini, xml lub jako pliki php.

Rysunek przedstawia fragment pliku konfiguracyjnego aplikacji AquaCms

```
[site]
    title               =   "Aqua CMS 1.0"
    keywords            =   "Content management system"
    defaultLanguage     =   "pl"
    url                 =   "http://127.0.0.1/aquacms/public"
[captcha]
    imgdir              =   APPLICATION_PATH "\..\public\gfx\captcha"
    font                =   APPLICATION_PATH "\..\public\gfx\comic.ttf"
    url                 =   "/gfx/captcha"

[users]
    newLimit            =   5;
[session]
    namespace.user      =   "frontend"
    namespace.admin     =   "backend"
    expiration_time     =   3600

    ;settings connected with handling session by database
    name                =   "session"

    primary[]           =   "session_id"
    primary[]           =   "save_path"
    primary[]           =   "name"

    primaryAssignment[] =   "idSession"
    primaryAssignment[] =   "savePath"
    primaryAssignment[] =   "name"
```

Rys. 3. Fragment pliku konfiguracyjnego CMS, źródło: opracowanie własne

Aby uzyskać dostęp w aplikacji do danych konfiguracyjnych należy wywołać statyczną metodę Get klasy Aqua_Registry.
Przykład uzyskania dostępu do zmiennej określającej liczbę wyświetlanych nowych użytkowników.

Aqua_Registry::get(Aqua_Registry::config)->users->newLimit; lub w widoku $this->config->users->newLimit.

Innymi metodami, które mogą być wykorzystane w Zend Framework do przechowywania danych konfiguracyjnych są Zend_Xml lub zapisanie zmiennych, jako tablica w PHP.

Przykład:

```
1. // Given an array of configuration data
2. $configArray = array(
3.     'webhost'  => 'www.example.com',
4.     'database' => array(
5.         'adapter' => 'pdo_mysql',
6.         'params'  => array(
7.             'host'     => 'db.example.com',
8.             'username' => 'dbuser',
9.             'password' => 'secret',
10.            'dbname'   => 'mydatabase'
11.        )
12.    )
13. );
14.
15. // Create the object-oriented wrapper upon the configuration data
16. $config = new Zend_Config($configArray);
17.
18. // Print a configuration datum (results in 'www.example.com')
19. echo $config->webhost;
20.
21. // Use the configuration data to connect to the database
22. $db = Zend_Db::factory($config->database->adapter,
23.                        $config->database->params->toArray());
```

Rys. 4. Fragment pliku konfiguracyjnego CMS,
źródło: http://framework.zend.com/manual/1.10/en/zend.config.introduction.html (data odczytu 10.01.2010)

Większość komponentów Zend Framework przyjmuje w konstruktorze dane, które są pobierane z plików konfiguracyjnych lub bazy danych.

3.4 Model, Widok, Kontroler

W celu oddzielenia od siebie warstw aplikacji: modelu danych, warstwy prezentacji oraz logiki sterowania systemem, zastosowałem model MVC(Model View Controler). Model ten jest architektonicznym wzorcem projektowym w informatyce, którego głównym celem jest separacja logiki systemu od prezentacji danych użytkownikowi.[7]

3.4.1 Widok

Do obsługi widoku w systemie użyłem biblioteki Smarty, która została adoptowana do Zend Framework. Utworzona klasa Zend_Smarty_View odpowiada za obsługę widoków w aplikacji. Klasa ta posiada kilkanaście. Chciałbym zwrócić uwagę na kilka z nich:

- __set

```
97    public function __set($key, $val)
98    {
99        $this->_smarty->assign($key, $val);
100   }
101
```

Rys. 5. Kod metody __set, źródło: opracowanie własne

Jest to tak zwana metoda magiczna, która wywoływana jest w momencie, kiedy chcemy przypisać nieistniejącej zmiennej klasowej jakąś wartość. W Zend_Smarty_View wykorzystane jest to by przypisać dowolną zmienną do szablonu w dość prosty sposób. Na przykład, gdy chcemy mieć w widoku w zmiennej $test, wartość „Ala ma kota” powinniśmy napisać poniższy kod:

$this->view->test = „Ala ma kota”

Podczas wywoływania tego kodu interpreter PHP nie znajdzie zmiennej test w klasie Zend_Smarty_View, więc zostanie wywołana magiczna metoda __set z parametrami

[7] Martin Brampton, PHP5 CMS.., op. cit., s. 224.

$key=”test”, $val = “Ala ma kota”. Parametry te zostaną przykazane do metody assign, która odpowiada za przypisanie zmiennej do szablonu smarty w danym widoku.

3.4.2 Kontroler

Kontroler jest najważniejszym elementem w całym systemie MVC. To on oddziela warstwę logiczną od warstwy prezentacyjnej. Każde żądanie jest niego przetwarzane i odpowiednio wysyłane do innych kontrolerów.

W systemie AquaCms rolę głównego kontrolera pełni klasa Zend_Controller_Action_Aqua. W klasie tej znajduje się metoda init(), która odpowiada za wczytanie konfiguracji, utworzenie obiektów widoku, wczytanie konfiguracji oraz ustawienie gdzie Zend powinien szukać plików widoków dla danej akcji kontrolera.

```
public function init()
{
        $this->config = new Zend_Config_Ini(APPLICATION_PATH.'/configs/site.ini');
        Zend_Registry::set('config',$this->config);
        $view = new Zend_View_Smarty(APPLICATION_PATH.'\views',array('compile_dir' => APPLICATION_PATH.'/views/compile',
                                                                     'cache_dir'   => APPLICATION_PATH.'/views/cache'
                                                                    )
                                     );
        $this->view = $view;
        $viewRenderer = Zend_Controller_Action_HelperBroker::getStaticHelper('viewRenderer');
        $viewRenderer
            ->setView($view)
            ->setViewScriptPathSpec( APPLICATION_PATH . '/views/:controller/:action.:suffix')
            ->setViewSuffix('html');

        $options = array(
                'viewSuffix' => 'html',
                'layoutPath' => APPLICATION_PATH . '/views/',
                );

        $this->layout = Zend_Layout_Aqua::startMvc($options);
        $this->view->layout     =     $this->layout ;
        $this->view->url        =     $this->config->site->url;
        $session = Aqua_Registry::get(Aqua_Registry::SESSION);
        $this->session = $session;
        if($session->user!=null) $this->view->user = $session->user;

}
```

Rys. 6. Kod metody init klasy Zend_Controller_Action_Aqua, źródło: opracowanie własne

3.4.3 Model

Modelem w Zend Framework są głownie klasy dziedziczące po klasie abstrakcyjnej Zend_Db_Table. Klasa ta daje nam interfejs obiektowy dla tabel w bazie danych. Zend_Db_Table dostarcza wiele metod, które są używane dla najczęściej wykonywanych operacjach na bazie danych. Na przykład wstawianie, aktualizowanie lub też usuwanie rekordów. Zend_Db_Table umożliwia również zdefiniowanie relacji pomiędzy tabelami. By klasa współpracowała z bazą danych musimy podać dwa podstawowe parametry: $_name oraz $_primary. Pierwszy z nich określa, z jakiej tabeli model korzysta, drugi określa nazwę indeksu klucza podstawowego. Poniższy rysunek przedstawia model dla tabeli użytkownicy wraz z przykładową metodą zwracającą nowych użytkowników, którzy aktywowali swoje konto.

```php
<?php
    class Models_User extends Zend_Db_Table_Abstract
    {
        protected $_name          = 'user';
        protected $_primary       = 'idUser';
        const FRIEND_ACCEPTED     = 3;
        const FRIEND_DENIED       = 2;
        const FRIEND_INVITED      = 1;
        const USER_ACTIVE         = 1;
        const USER_INACTIVE       = 0;

        /**
         * Funkcja zwraca nowych użytkowników
         *
         * @param int $limit
         * @return Zend_Db_Statement
         */
        public function getNew($limit)
        {
            $limit = $limit ? $limit : Aqua_Registry::get(Aqua_Registry::CONFIG)->users->newLimit;
            $st  = $this->getDefaultAdapter()->select()
                                    ->from(array('u'=>'user'),array('login'))
                                    ->order('u.idUser DESC')
                                    ->joinUsing('user_info','idUser')
                                    ->where('u.isActive=?', self::USER_ACTIVE)
                                    ->limit($limit);
            return $st->query();
        }
```

Rys. 7. Przykład klasy modelu dla tabeli users, źródło: opracowanie własne

3.5 Funkcje serwisu

W tym rozdziale przedstawione są podstawowe funkcje serwisu takie jak logowanie, zarządzanie profilem użytkownika, rejestracja oraz funkcjonalność stopki witryny.

3.5.1 Logowanie

Mechanizm logowania wykorzystuje CAPTCHA(Completely Automated Public Turing test to tell Computers and Humans Apart), który zabezpiecza system przed oprogramowaniem, które umieszcza spam na stronach.

Rys. 8. Okno logowania systemu Aqua Cms, źródło: opracowanie własne

Captcha jest tworzona za pomocą klasy Zend_Captcha_Image, w której ustawiłem długość słowa, czcionkę, miejsce zapisywania obrazków, czas, po którym captcha zostanie uznana za nieważną oraz nazwę.

```php
public function loginboxAction()
{
        $captcha = new Zend_Captcha_Image(array(
                                                'name' => 'captcha',
                                                'wordLen' => 5,
                                                'timeout' => 3000,
                                            ));

        $captcha->setImgDir($this->config->captcha->imgdir.'/');
        $captcha->setFont($this->config->captcha->font);
        $url = $this->config->site->url.$this->config->captcha->url.'/';
        $captcha->setFontSize(30);
        $captcha->setImgUrl($url);
        $captcha->setImgAlt('Przepisz tekst z obrazka');
        try{
            $id = $captcha->generate();
        }catch(Exception $e){
            echo $e->getMessage();
        }
        $this->view->captcha = $captcha;
        echo $this->view->render('boxes/login.html');
        exit;
}
```

Rys. 9. Metoda odpowiadająca za wyświetlenie okna logowania, źródło: opracowanie własne

Całość jest wywoływana poprzez AJAX w specjalnym okienku za pomocą funkcji showDialog oraz login.

```javascript
function showDialog(title,id,url)
{
        var id = (id==undefined ? 0 : id);
        if(url!=undefined)
        {
            $("#dialog-window-" + id).load(url);
        }
        var cnt = $("#dialog-windows").html();
        if(title!='' && title!=undefined) $("#dialog-window-" + id).attr('title',title);


        $("#dialog-window-" + id).dialog({bgiframe: true, modal: true,buttons: {
                Ok: function(){
                    $(this).dialog('close');
                }
            }
        });
        $("#dialog-windows").html(cnt);
        return false;
}
```

Rys. 10. Funkcja wyświetlająca dowolną treść pobieraną ajaxowo w okienku dialogowym, źródło: opracowanie własne

```
77 function login()
78 {
79     showDialog('Panel logowania do systemu',0,'{/literal}{$url}{literal}/auth/loginbox');
80     return false;
81 }
```

Rys. 11. Funkcja wyświetlająca okno logowania, źródło: opracowanie własne

Po poprawnym zalogowaniu zostajemy przeniesieni na główną stronę. Gdy jesteśmy zalogowani mamy dostęp do edycji profilu oraz skrzynki odbiorczej, możemy się również wylogować używając odnośnika „wyloguj”.

Rys. 12. Nagłówek strony po poprawnym zalogowaniu, źródło: opracowanie własne

3.5.2 Profil użytkownika

Profil użytkownika jest obsługiwany poprzez kontroler o nazwie „user”. Na stronie widoku profilu użytkownika znajdują się informację o nim takie jak imię, nazwisko, język, a także znajomi, których posiada. Oprócz tego są też tam przyciski, które w przyszłej wersji CMS będą umożliwiać dodanie użytkownika do znajomych, jeżeli jeszcze go w nich nie ma oraz napisanie użytkownikowi wiadomości.

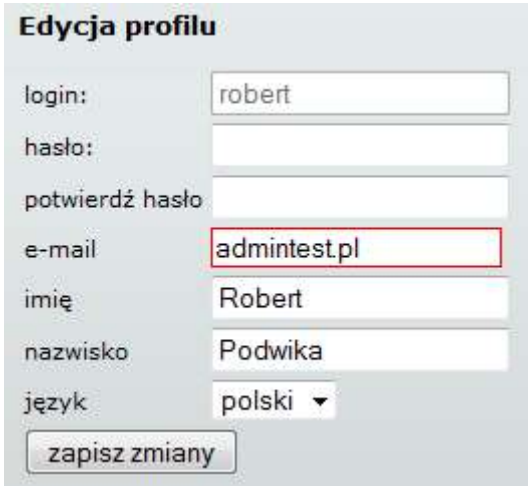
Rys. 13. Widok profilu użytkownika, źródło: opracowanie własne

3.5.3 Edycja profilu

Edycja profilu jest obsługiwana poprzez kontroler o nazwie „profile”. Klikając w link „edytuj profil” zostajemy przeniesieni na stronę, która umożliwia zmianę danych profilowych.

Rys. 14. Edycja profilu, źródło: opracowanie własne

Każde pole w formularzu jest odpowiednio walidowane, gdy popełnimy błąd zostaniemy o tym poinformowani poprzez czerwoną ramkę wokół błędnie wypełnionego pola. Na poniższym obrazku został pokazany błąd w polu e-mail.



Rys. 15. Przykład zakomunikowania błędu przez system, źródło: opracowanie własne

Poprawność danych formularza edycji profilu sprawdzana jest przez klasy pochodne dla „Zend_Validate", które posiadają metodę „isValid" sprawdzającą czy dany ciąg znaków jest poprawny. Na poniższym obrazku zostały użyte obiekty Zend_Validate dla sprawdzenia zgodności haseł, długości ciągu znaków, poprawności adresu e-mail oraz sprawdzenia czy

znaki są alfanumeryczne. W linii numer 40 zmienna $errors jest wyzerowana. W liniach 42-47 oraz 57, 58 są sprawdzane kolejne zmienne, w przypadku niezgodności w tablicy błędów dla danego pola ustawiana jest wartość „true”.

```
32      protected function doSave()
33      {
34          $info              = $this->_request->getParam('info');
35          $passwordValidator = new Zend_Validate_Identical($this->_request->getParam('password'));
36          $emailValidator    = new Zend_Validate_EmailAddress();
37          $nameValidator     = new Zend_Validate_Alpha(false);
38          $lenghtValidator   = new Zend_Validate_StringLength(3,15);
39
40          $errors = null;
41
42          if(!$emailValidator->isValid($info['email']))
43              $errors['email']     = implode(':',$emailValidator->getErrors());
44          if(!$nameValidator->isValid($info['firstname']) || !$lenghtValidator->isValid($info['firstname']))
45              $errors['firstname']    = true;
46          if(!$nameValidator->isValid($info['lastname']) || !$lenghtValidator->isValid($info['lastname']))
47              $errors['lastname']     = true;
48
49          //Zend_Debug::dump($errors);
50
51          $insertData = array("email"         =>    $info['email'],
52                              "firstname"     =>    $info['firstname'],
53                              "lastname"      =>    $info['lastname'],
54                              "lang"          =>    $info['lang']);
55          if($info['password']!='' || $this->_request->getParam('password',null) != null)
56          {
57                  if(!$passwordValidator->isValid($info['password']) || !$lenghtValidator->isValid($info['password']))
58                      $errors['password']  = implode(':',$passwordValidator->getErrors());
59                  else
60                      $insertData['password'] = $info['password'];
61          }
```

Rys. 16. Kod funkcji walidującej i zapisującej dane profilowe, źródło: opracowanie własne

Jeżeli po aplikacja po dojściu do linii numer 62 nie posiada żadnych błędów dane profilu zostają zaktualizowane w bazie, w przeciwnym przypadku do widoku zostają przekazane błędy oraz dane formularza.

```
62              if($errors == null)
63              {
64                  $ui_view = new Models_Usersview();
65                  $ui_view->update((int)$this->session->user->idUser,$insertData);
66              }
67              else
68              {
69                  $this->view->error    = $errors;
70                  $this->view->info     = $info;
71              }
```

Rys. 17. Fragment kodu odpowiedzialnego za zapisanie danych profilowych, źródło: opracowanie własne

W widoku każde pole jest sprawdzane pod kątem posiadania błędów, jeżeli takowe istnieją zostaje dodana klasa arkuszy styli kaskadowych o nazwie „error”, która tworzy ramkę wokół pola.

Rys. 18. Kod odpowiedzialny za wyświetlenie ramki wokół źle wypełnionego pola, źródło: opracowanie własne

3.5.3 Rejestracja

Moduł rejestracji podobnie jak logowanie jest wywoływany okienku dialogowym.

Rys. 19. Okno rejestracji użytkownika, źródło: opracowanie własne

Gdy klikniemy na odnośnik „Zarejestruj” pokaże się okienko z danymi, które są potrzebne przy rejestracji. Po poprawnym wypełnieniu formularza i kliknięciu przycisku „zarejestruj” zostaniemy przeniesieni na stronę, która potwierdza poprawną rejestrację.

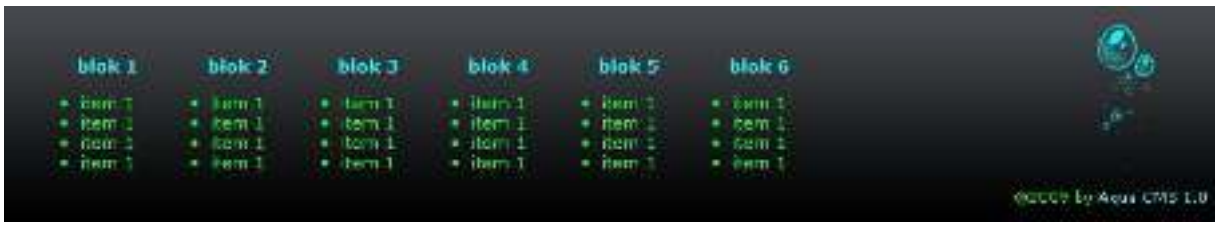
Rys. 20. Komunikat po poprawnej rejestracji, źródło: opracowanie własne

W przypadku, gdy popełnimy błąd pojawi się formularz, który umożliwia poprawę danych. Na poniższym rysunku źle został wprowadzony adres e-mail oraz hasła, które zostały wprowadzone nie były identyczne. Po poprawnym wprowadzeniu danych i naciśnięciu przycisku „zarejestruj" zostaniemy przeniesieni na stronę, która informuje nas o poprawnej rejestracji.

Rys. 21. Komunikat po błędnej rejestracji, źródło: opracowanie własne

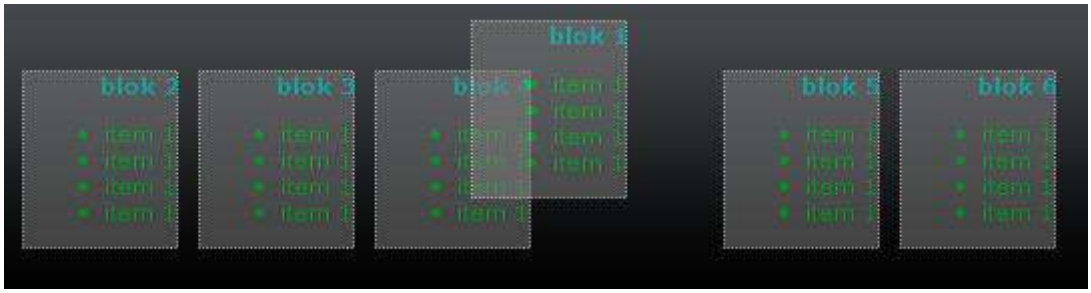
3.5.4 Stopka witryny

Stopka umożliwia dynamiczne przesuwanie bloków, które się w niej znajdują. W wersji finalnej administrator systemu AquaCms będzie mógł przesuwać bloki, a one będą zapamiętywane.



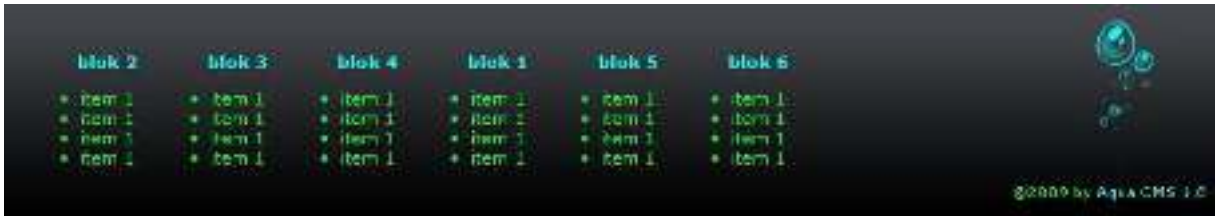
Rys. 22. Stopka portalu, źródło: opracowanie własne

Przesuwanie polega na najechaniu na blok, naciśnięciu lewego przycisku myszki i trzymając go przesunięciu bloku w wybrane miejsce docelowe. Gdy lewy zostanie wybrany element, który chcemy przesunąć inne bloki zmienią kolor na szary, staną się przeźroczyste oraz wokół nich pojawi się obramowanie.



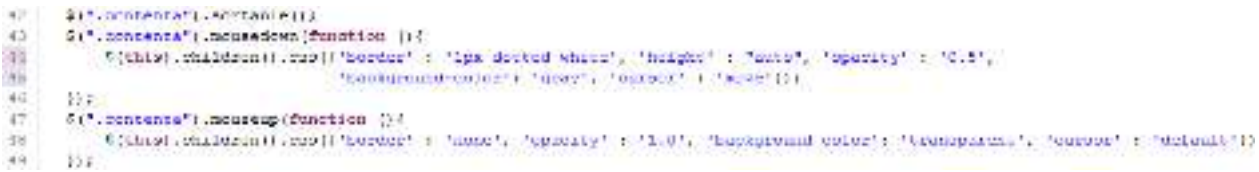
Rys. 23. Przykład przeniesienia bloków, źródło: opracowanie własne

Po puszczeniu bloku style bloków zostaną przywrócone, a element pozostanie w miejscu, do którego go przenieśliśmy. Na rysunku poniżej blok 1 został wstawiony pomiędzy bloki 4 i 5.



Rys. 24. Układ bloków po przeniesieniu, źródło: opracowanie własne

Przesuwanie bloków jest realizowane za pomocą biblioteki JQuery UI. Dynamiczna zmiana styli jest zaprogramowana w JQuery



Rys. 25. Kod funkcji odpowiadającej za przesuwanie bloków, źródło: opracowanie własne

W linii numer 42 na elemencie listy, która zawiera bloki zastosowana jest metoda „sortable”, która umożliwia ich przemieszczanie. W linii 43-46 do zdarzenia naciśnięcia klawisza myszki na elemencie listy przypisywana jest funkcja, która wszystkim dzieciom tej listy zmienia style na zadane w skrypcie. W linii 47-49 analogicznie jak w przykładzie wyżej następuje zmiana styli, ale wtedy, kiedy puścimy klawisz myszki.

Podsumowanie

Pisząc pracę postawiłem sobie za cel napisanie w aplikacji działającej na modelu MVC. Chciałem, żeby kod był oparty o programowanie zorientowane obiektowo. Ważnym dla mnie było oddzielenie warstwy prezentacyjnej od warstwy aplikacji. Udało mi się zrealizować wszystkie z założonych celów. Używając takich bibliotek jak Smarty, Zend, Jquery napisałem nowoczesną aplikację, która zarządza treścią na stronie. Aplikacje, która z uwagi na rozbudowaną bazę danych i obiektowy model programowania może być łatwo rozwijana przez grupę programistów.

AquaCms posiada moduły, które z łatwością mogą być przetłumaczone na różne języki z automatycznym wykrywaniem języka przeglądarki. Dodanie modułu jest bardzo łatwe i sprowadza się do dodania odpowiedniego kontrolera oraz jego widoków. Dzięki Jquery UI i funkcjom, które napisałem każda treść może zostać wyświetlona w okienku. Poprzez użycie smarty osoby nietechniczne mogą edytować kod html strony bez obawy o to, iż coś może zostać popsute.

Używanie najnowszych bibliotek i frameworków oraz programowanie zorientowane obiektowo ułatwia pisanie aplikacji. Zastosowanie wzorca MVC wprowadza strukturalność aplikacji, sprawia, że kod nie jest chaotyczny. Uważam, że w swojej pracy osiągnąłem wszystkie zamierzone cele i jestem pewny, że aplikacja, którą napisałem będzie rozwijana i stanie się w pełni funkcjonalnym systemem WCMS.

Bibliografia

1. http://jquery.com
2. Martin Brampton.: PHP5 CMS Framework Development, Birmingham, Packt Publishing Ltd. 2008
3. http://smarty.com
4. http://www.cmsstrict.com
5. http://www.survivethedeepend.com
6. http://zend.com

Spis Rysunków