Krótki kurs tworzenia stron w technologii PHP

Mateusz Gałażyn, Jeremi Niedziela

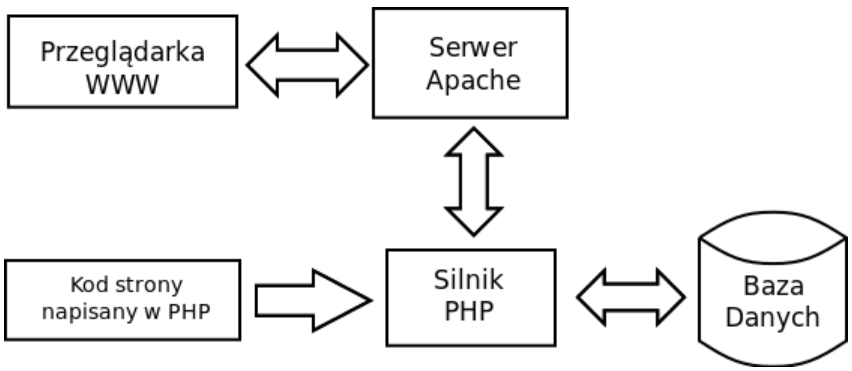
1 maja 2012

1 Wstęp

Ten krótki kurs ma na celu przedstawienie sposobu pisania stron w przy użyciu technologii Apache + MySQL + PHP (w skrócie AMP) - jednego z najpopularniejszych zestawów oprogramowania służącego do uruchamiania serwisów WWW. Ten kurs jest bardzo okrojony i zawiera tylko wybrane elementy języka PHP oraz MySQL dlatego aby w pełni zrozumieć metodykę tworzenia aplikacji w oparciu o te języki dobrze jest się wspierać dokumentacjami technicznymi, innymi kursami dostępnymi w internecie, a także specjalistycznymi forami dyskusyjnymi.
Dokumentacja PHP: http://www.php.net/manual/pl/
Dokumentacja MySQL: http://dev.mysql.com/doc/refman/5.5/en/index.html
Dokumentacja Apache: http://httpd.apache.org/docs/2.4/
Kurs PHP @ Wikibooks: http://pl.wikibooks.org/wiki/PHP

1.1 Mechanizm działania systemu AMP

Gdy użytkownik strony uruchamia przeglądarkę i wpisuje w pasek adresu, adres szukanego serwisu WWW, przeglądarka nawiązuje połączenie z serwerem na którym są uruchomione usługi umożliwiające dostęp do strony.
Apache - jest to najszerzej stosowany w internecie serwer HTTP
PHP - jeden z najpopularniejszych języków programowania używany do tworzenia stron WWW.
MySQL - system zarządzania bazami danych za pomocą języka SQL



Rysunek 1: Schemat komunikacji zestawu AMP

Żądanie otrzymane od przeglądarki jest przechwytywane przez serwer Apache, który przetwarzając je uruchamia kod strony napisany w języku PHP. Następnie silnik PHP komunikuje się z bazą danych, pobiera dane, przetwarza je i zamienia na kod HTML, który zwraca serwerowi Apache. W kolejnym kroku serwer Apache wysyła kod HTML razem z obrazami umieszczonymi na stronie i stylami do przeglądarki, która renderuje i wyświetla stronę.

2 Instalacja środowiska AMP

2.1 Instalacja na systemie Windows

2.1.1 Instalacja serwera Apache

2.1.2 Instalacja PHP

2.1.3 Instalacja serwera MySQL

2.2 Instalacja na systemie Linux

2.2.1 Instalacja serwera Apache

2.2.2 Instalacja PHP

2.2.3 Instalacja serwera MySQL

2.3 Konfiguracja

2.3.1 Konfiguracja Apache

2.3.2 Konfiguracja PHP

2.3.3 Konfiguracja MySQL

2.4 Instalacja phpMyAdmin

3 Projekt forum dyskusyjnego

Naszym celem jest jest zbudowanie prostego forum dyskusyjnego. Musimy dać możliwość rejestrowania się poszczególnym użytkownikom, dodawania własnych wątków oraz odpowiadania na już utworzone. Potrzebne będzie też konto administratora, który będzie miał prawo moderacji odpowiedzi w tematach (postów).

3.1 Układ podstron

Gdy zostanie już ustalona lista wymaganych funkcjonalności projektowanego serwisu, następnym etapem jest stworzenie układu poszczególnych podstron.

Strona główna Tutaj trzeba wyświetlić listę tematów (wraz z odnośnikami do podstrony, na którym będzie pojedynczy wątek wyświetlany w całości) które zostały założone na forum, umieścić odnośniki do podstrony z formularzem używanym do zalogowania się i rejestracji dla użytkowników oraz odnośnik do podstrony umożliwiającej założenie nowego tematu.

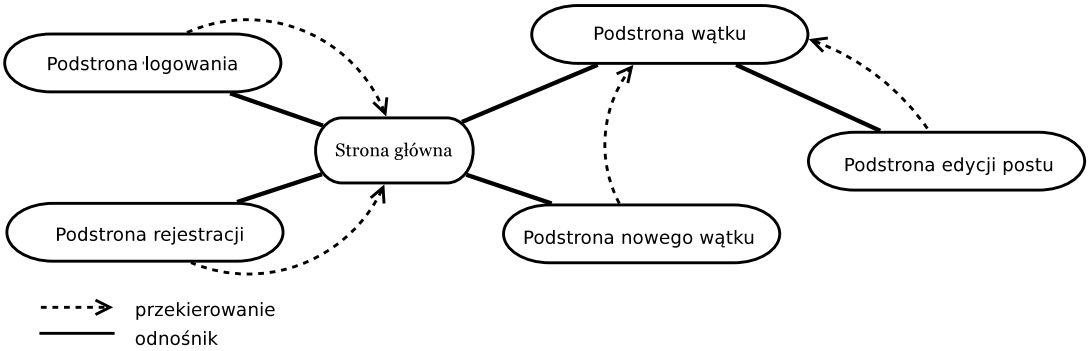
Podstrona wątku W tym miejscu trzeba wyświetlić wszystkie posty w danym wątku, oraz formularz dający możliwość odpowiedzi w tym wątku. Gdy administrator wejdzie na tą podstronę, trzeba dać także dodatkowe możliwości edycji i usuwania poszczególnych postów.

Podstrona logowania Zawierać będzie formularz logowania umożliwiający uwierzytelnienie użytkownika wchodzącego stronę. Uwierzytelnienie będzie polegało na porównaniu nazwy użytkownika i hasła z obecnymi w bazie danych. Po zalogowaniu się, użytkownik zostanie przekierowany z powrotem na stronę główną. Podstrona dostępna tylko dla niezalogowanych użytkowników.

Podstrona rejestracji Odpowiedzialna za obsługę rejestracji nowego użytkownika. Trzeba będzie wyświetlić formularz rejestracji, odebrać i zweryfikować dane oraz dodać je do bazy danych.

Podstrona nowego wątku Podstrona z formularzem umożliwiającym dodanie nowego wątku. Po odebraniu danych od użytkownika trzeba będzie dokonać ich weryfikacji i dodać do bazy danych. Na koniec trzeba przekierować użytkownika do podstrony wątku.

Podstrona edycji postu Podstrona z formularzem umożliwiającym edycję własnych postów, a także w przypadku administratora - edycji każdego postu. Przekierowuje z powrotem do podstrony wyświetlającej cały wątek.



Rysunek 2: Układ podstron forum

3.2 Projekt bazy danych

Informacje w bazie danych są przechowywane w postaci tabel, podobnie jak w arkuszach kalkulacyjnych dostępnych w popularnych pakietach biurowych. W bazie projektowanego forum będziemy przechowywać listę użytkowników, listę tematów oraz listę postów. Dla każdej z tych list trzeba utworzyć oddzielną tabelę: users, threads, posts. W każdej z tych tabel konieczne jest stworzenie odpowiednich kolumn służących do przechowywania danych we właściwych typach. Ustawienie typu danej przechowywanej w każdej kolumnie przyśpiesza pracę serwera MySQL poprzez automatyczną optymalizację realizowanych zapytań do bazy oraz optymalizację sposobu przechowywania tabel na dysku.

3.2.1 Typy danych MySQL

W języku MySQL jest wiele typów danych (dokładny opis: http://dev.mysql.com/doc/refman/5.5/en/data-type-overview.html), te najważniejsze to:

Typy znakowe:

CHAR Przechowuje ciąg znaków do 255 elementów.

VARCHAR Tak jak **CHAR** przechowuje ciąg znaków do 255 elementów. Główną różnicą pomiędzy nimi jest to, że **VARCHAR** zmienia swój rozmiar w zależności od przyjętej liczby znaków, a **CHAR** zawsze rezerwuje miejsce 255 znaków niezależnie od długości zapisywanego ciągu.

TEXT Przechowuje ciąg do 65535 znaków.

MEDIUMTEXT Przechowuje ciąg do 16777215 znaków.

Typy liczbowe:

TINYINY Przechowuje liczby od -128 do 127 (0 do 255 w przypadku **UNSIGNED**).

SMALLINT Przechowuje liczby od -32768 to 32767 (0 do 65535 w przypadku **UNSIGNED**).

MEDIUMINT Przechowuje liczby od -8388608 do 8388607 (0 do 16777215 w przypadku **UNSIGNED**).

FLOAT Małe liczby zmiennoprzecinkowe.

DOUBLE Liczby zmiennoprzecinkowe o podwojonej precyzji.

Każdy typ liczbowy występuje także w wersji **UNSIGNED**, która pozwala przechowywać dwa razy większe liczby bez przechowywania informacji o znaku.

Inne typy:

DATE Data w formacie YYYY-MM-DD

DATETIME Data w formacie YYYY-MM-DD HH:MM:SS

TIMESTAMP Znacznik czasu w formacie: YYYYMMDDHHMMSS

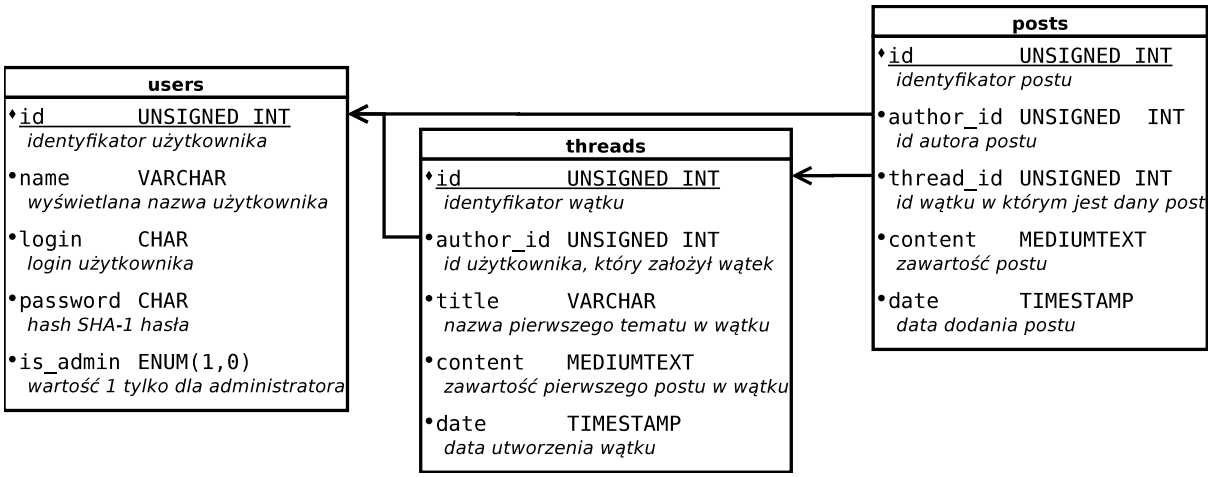
TIME Czas w formacie: HH:MM:SS

ENUM Podobnie jak w C++, definiuje listę określonych wartości jakie może przyjąć to pole.

SET Podobnie jak **ENUM**, lecz pole w tabeli może przyjąć kilka wartości na raz.

3.2.2 Projekty tabel

Tabela *users* powinna zawierać kolumny przechowujące informacje o użytkowniku takie jak: nazwa użytkownika, login używany przy identyfikacji na stronie, hash SHA-1[1] hasła oraz pole determinujące czy dany użytkownik ma uprawnienia administracyjne (wartość 1, gdy użytkownik jest administratorem).
Tabela *threads* powinna zawierać kolumny przechowujące tytuł wątku, zawartość pierwszego posta oraz data utworzenia wątku.
Tabela *posts* powinna zawierać kolumny przechowujące treść posta i datę jego utworzenia.
Dodatkowo w każdej z tabel powinna znajdować się kolumna *id* identyfikująca w każdej tabeli pojedynczy rekord. To rozwiązania ułatwia odwoływanie się do poszczególnych wpisów (rekordów) w tabeli, bez konieczności ponownego przeszukiwania całej tabeli. W tabelach *posts* i *threads* oprócz informacji o postach i wątkach musimy utworzyć tak zwane relacje - kolumny wiążące poszczególne posty z użytkownikami oraz wątkami. Jest to tak zwana relacja **jeden do wielu**. Zagadnienie to można rozwiązać w prosty sposób: w tabeli *threads* tworzymy kolumnę *author_id*, którą będzie zawierała id rekordu w tabeli *users* odpowiadającego użytkownikowi, który stworzył dany wątek. Analogiczną operację trzeba wykonać dla tabeli posts: tworzymy kolumnę *author_id* oraz dodatkową *thread_id*, która definiuje wątek do którego przynależy dany post.



Rysunek 3: Schematy tabel i ich wzajemne relacje

Optymalizacja Oprócz precyzowania typów danych dla każdej kolumny, aby ułatwić sobie i serwerowi pracę można zdefiniować dodatkowe atrybuty dla poszczególnych kolumn. Kolumna *id* występująca w każdej z tabel musi być zdefiniowana z atrybutami *PRIMARY KEY* (co jest zaznaczone na schemacie podkreśleniem) - oznacza to, że kolumna jest wykorzystywana do identyfikacji rekordów w tabeli, *AUTO_INCREMENT* - każdy dodawany rekord ma ustawiane automatycznie *id* o 1 większe niż największe *id* w tabeli, *NOT NULL* - nie może mieć wartości 0.

[1] W bazach danych **NIGDY** nie powinno się przechowywać haseł w jawnej postaci tekstu. W przypadku gdy osoba trzecia uzyska nieautoryzowany dostęp do bazy danych, będzie mogła skopiować listę haseł wszystkich użytkowników, co nie powinno mieć nigdy miejsca. Zamiast tego przechowuje się specjalne hashe wygenerowane na podstawie haseł. Więcej informacji na temat hashy: pl.wikipedia.org

3.2.3 Kod SQL

Nadszedł czas na stworzenie zaprojektowanej przez nas struktury tabel. Uruchamiamy serwer MySQL, a następnie poleceniem (dla systemu Linux):

```
mysql -u root -p
```

uruchamiamy wiersz poleceń MySQL. Na systemie Windows aby uruchomić wiersz poleceń, wpierw trzeba uruchomić terminal, a w nim wpisać:

```
C:> cd C:\sciezka\do\katalogu\mysql\bin
C:\sciezka\do\katalogu\mysql\bin> mysql.exe -u root -p
```

Następnie po wpisaniu hasła, aby stworzyć bazę danych wydajemy polecenie:

```
CREATE DATABASE forum;
```

Aby przystąpić do tworzenia tabel, trzeba wybrać nowo utworzoną bazę danych:

```
USE forum;
```

Możemy teraz utworzyć tabelę *users* wydając poniższe polecenie:

```
1  CREATE TABLE `forum`.`users` (
2          `id` INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
3          `name` VARCHAR( 255 ) NOT NULL ,
4          `login` CHAR( 255 ) NOT NULL ,
5          `password` CHAR( 255 ) NOT NULL ,
6          `id_admin` ENUM( '1', '0' ) NOT NULL ,
7          UNIQUE (
8                  `name` ,
9                  `login`
10         )
11 );
```

Ważne jest umieszczenie na końcu wyrażenia średnika, ponieważ informuje to interpreter MySQL że to jest koniec komendy. Bez średnika interpreter będzie czekał na dalsze wprowadzanie komend, dopóki nie napotka średnika. W linii 1 podajemy nazwę bazy danych, a po kropce jest nazwa tworzonej tabeli. Nazwy bazy, tabel i kolumn są wzięte w ` `. W liniach 2-6 zdefiniowane są kolumny tabeli oddzielone przecinkami. Składnia jest następująca:

```
`nazwa_kolumny` TYP NULL [dodatkowe atrybuty] ,
```

Na początku podajemy nazwę kolumny, potem nazwę typu, jeżeli jest to typ liczbowy bez znaku, za nazwą typu dodajemy *UNSIGNED*. Jeżeli jest to typ znakowy *CHAR* lub *VARCHAR* to wpisujemy w nawiasie maksymalną długość ciągu znaków. Jeżeli jest to typ *ENUM*, lub *SET* w nawiasie podajemy dopuszczalne wartości. Następnie podajemy informację czy w tą kolumnę może zostać wpisana wartość pusta - parametr *NULL*, lub gdy nie: *NOT NULL*. Jeżeli chcemy ustawić dodatkowe opcje dla kolumny wpisujemy je oddzielając spacją. Dla kolumny *id* zostały ustawione opcje *AUTO_INCREMENT* oraz *PRIMARY KEY*. Po wymienionych kolumnach można zdefiniować dodatkowe indeksy, w linii 7 zdefiniowany jest indeks *UNIQUE*: dla kolumn umieszczonych w nawiasie wartości w pól w rekordach nie mogą się powtarzać.

Tabela *threads*:

```
1  CREATE TABLE `forum`.`threads` (
2          `id` INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
3          `author_id` INT UNSIGNED NOT NULL ,
4          `title` VARCHAR( 255 ) NOT NULL ,
5          `content` MEDIUMTEXT NOT NULL ,
6          `date` TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
7  );
```

Kod jest prawie identyczny jak w poprzednim przypadku. Różnica pojawia się jedynie w linii 6. Po typie danych *TIMESTAMP* zdefiniowana jest opcja *ON UPDATE CURRENT_TIMESTAMP*, która mówi, że po aktualizacji rekordu silnik bazy danych automatycznie umieści w kolumnie *date* aktualny znacznik czasu. Następnie jest podana informacja, że wartość pola nie może być wartością *NULL*. Parametr *DEFAULT CURRENT_TIMESTAMP* mówi nam, że gdy przy wpisywaniu wartości do bazy danych nie podamy wartości tego pola, serwer MySQL automatycznie wypełni to pole aktualną godziną.

Tabela *posts*:

```
CREATE TABLE `forum`.`posts` (
`id` INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`author_id` INT UNSIGNED NOT NULL ,
`thread_id` INT UNSIGNED NOT NULL ,
`content` MEDIUMTEXT NOT NULL ,
`date` TIMESTAMP ON UPDATE CURRENT_TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

Składnia jest analogiczna jak w poprzednim wypadku. Powyższe trzy zapytania *CREATE* tworzą już kompletną, ale jeszcze pustą strukturę naszej bazy danych.

4 Kod strony