helion.pl

Idź do

- Spis treści
- Przykładowy rozdział

Katalog książek

- Katalog online
- Zamów drukowany katalog

Twój koszyk

- Dodaj do koszyka

Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

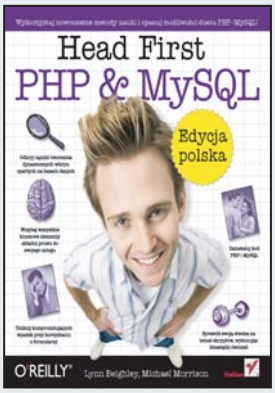
Czytelnia

- Fragmenty książek online

Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl
© Helion 1991-2008

Head First PHP & MySQL. Edycja polska



Autorzy: Lynn Beighley, Michael Morrison
Tłumaczenie: Łukasz Schmidt
ISBN: 978-83-246-2113-2
Tytuł oryginału: Head First PHP & MySQL
Format: 200×230, stron: 800

- Odkryj tajniki tworzenia dynamicznych witryn opartych na bazach danych
- Uniknij kompromitujących wpadek przy korzystaniu z formularzy
- Wczytaj wszystkie kluczowe elementy składni prosto do swego mózgu
- Zainstaluj kod PHP i MySQL
- Sprawdź swą wiedzę na temat skryptów, wykonując dziesiątki ćwiczeń

PHP wraz z MySQL stanowią najpopularniejszy zespół, służący do szybkiego tworzenia aplikacji internetowych o różnym stopniu złożoności. Dzięki dużym możliwościom, wydajności oraz optymalnemu podejściu do wielu zagadnień tworzą prawdopodobnie najpopularniejszą platformę do wprowadzania atrakcyjnych rozwiązań. Na temat wykorzystania możliwości PHP i MySQL napisano już wiele książek, jednak ta jest wyjątkowa – należy do popularnej i przyjaznej Czytelnikowi serii Head First!

„Head First PHP & MySQL. Edycja polska" nie jest kolejnym trudnym i nudnym podręcznikiem do nauki PHP i MySQL. Autorzy wykorzystują tu innowacyjne oraz niezwykle skuteczne techniki przyswajania wiedzy szybko i bezboleśnie. Z pomocą licznych ilustracji i ciekawych skojarzeń nauczą Cię, jak stosować PHP wraz z MySQL w Twojej codziennej pracy. Dzięki temu bez najmniejszych problemów przygotujesz formularz, przetworzysz dane wprowadzone przez użytkownika, a następnie zapiszesz je w bazie MySQL. Ponadto zostaniesz wtajemniczony w szczegóły języka PHP oraz SQL. Zapoznasz się z zagrożeniami oraz dowiesz się, jak chronić swoją aplikację przed atakami z zewnątrz. To wszystko sprawi, że pewnym krokiem wejdziesz w świat profesjonalnych aplikacji internetowych!

- Tworzenie i obsługa formularzy
- Zastosowanie zmiennej $_POST
- Wysyłanie wiadomości e-mail z poziomu PHP
- Wykonywanie zapytań SQL
- Pobieranie i wykorzystywanie danych z MySQL w PHP
- Elementarz języka PHP
- Przesyłanie plików
- Bezpieczeństwo danych w PHP
- Zastosowanie sesji oraz ciasteczek
- Sortowanie wyników
- Obsługa kanałów RSS
- Zamieszczanie materiałów multimedialnych
- Zastosowanie formatu XML
- Instalacja i konfiguracja serwera Apache

Wykorzystaj nowoczesne metody nauki i opanuj możliwości duetu PHP–MySQL!

Spis treści (skrócony)

	Wprowadzenie	27
1	Ożywianie statycznych stron. *To żyje!*	39
2	Łączenie się z bazą MySQL. *Jak wszystko wiąże się ze sobą?*	97
3	Tworzenie i zapełnianie bazy danych. *Tworzenie własnych danych*	141
4	Realistyczne i praktyczne aplikacje. *Twoja aplikacja w sieci WWW*	195
5	Używanie danych przechowywanych w plikach. *Kiedy baza danych nie wystarcza*	259
6	Zabezpieczanie aplikacji. *Załóżmy, że każdy stanowi zagrożenie*	329
7	Tworzenie spersonalizowanych aplikacji sieciowych. *Pamiętasz mnie?*	379
7½	Eliminowanie powtórzeń w kodzie. *Współdzielenie oznacza troskę*	451
8	Kontroluj dane — kontroluj swój świat. *Zbieranie danych*	461
9	Funkcje niestandardowe i do obsługi łańcuchów znaków. *Dzięki funkcjom żyje się lepiej*	533
10	Wyrażenia regularne. *Reguły zastępowania*	593
11	Wizualizowanie danych i inne zagadnienia. *Dynamiczne dodawanie grafiki*	637
12	Rozpowszechnianie danych i usługi sieciowe. *Interfejs do komunikacji ze światem*	687
A	Pozostałości. *Dziesięć najważniejszych tematów (których nie poruszyliśmy)*	743
B	Konfigurowanie środowiska programistycznego. *Miejsce do zabawy*	761
C	Rozszerzanie PHP. *Jeszcze więcej możliwości*	779

Spis treści (z prawdziwego zdarzenia)

Wprowadzenie

Twój mózg a PHP i MySQL. Podczas gdy *Ty* próbujesz się czegoś *nauczyć, mózg* wyświadcza Ci przysługę i dba o to, abyś *niczego nie zapamiętał*. Twój mózg myśli sobie: „Lepiej zostawić miejsce na ważniejsze informacje, na przykład o dzikich zwierzętach, których należy unikać, i o tym, że ćwiczenie jogi pod wodą nie jest dobrym pomysłem". Jak więc możesz *przechytrzyć* mózg i przekonać go, że Twoje życie zależy od znajomości technologii PHP i MySQL?

Dla kogo przeznaczona jest ta książka?	28
Kto prawdopodobnie powinien zrezygnować z tej książki?	28
Wiemy, co sobie myślisz	29
Wiemy, co sobie myśli Twój mózg	29
Metapoznanie: myślenie o myśleniu	31
Oto, co MY zrobiliśmy	32
A oto, co TY możesz zrobić, aby zmusić mózg do posłuszeństwa	33
Przeczytaj koniecznie	34
Zespół recenzentów technicznych	36
Podziękowania	37

Ożywianie statycznych stron

1 To żyje!

Umiesz tworzyć świetne strony za pomocą HTML-a i odrobiny stylów CSS. Zauważyłeś jednak, że użytkownicy Twoich witryn mogą tylko pasywnie przeglądać zawartość witryn. Komunikacja jest jednostronna i planujesz to zmienić. ***Chcesz poznać opinie użytkowników***. Aby dowiedzieć się, o czym myślą internauci, trzeba umożliwić im **wprowadzanie danych za pomocą formularzy**. Musisz też umieć **przetworzyć** i **pobrać te informacje**. Wygląda na to, że podniesienie poziomu witryn wymaga użycia czegoś więcej niż samego kodu HTML.

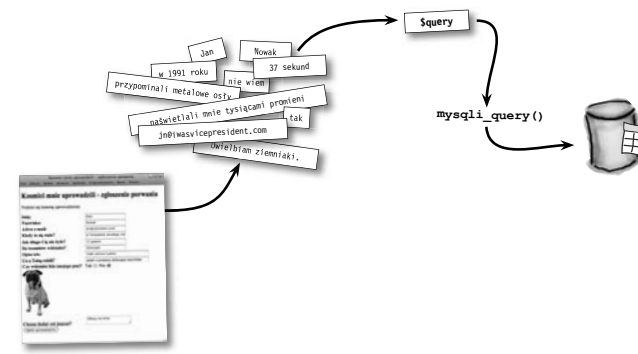
Strony HTML są statyczne i nudne	40
PHP ożywia strony WWW	41
Formularz pomoże Markowi poznać całą historię	43
Formularze składają się z kodu HTML	44
Formularz HTML sprawia problemy	46
HTML działa po stronie KLIENTA	48
PHP działa po stronie SERWERA	49
Skrypty PHP działają na serwerze	50
Dostęp do danych z formularza w kodzie PHP	54
Skrypty PHP muszą działać na serwerze!	56
Przenoszenie skryptów PHP na serwer	57
Serwer przekształca kod PHP na HTML	60
Analiza skryptu PHP Marka	62
Kilka reguł pisania kodu związanych z językiem PHP	63
Ustalanie idealnych nazw dla zmiennych	64
Zmienne służą do przechowywania danych w skryptach	69
$_POST to specjalna zmienna na dane z formularza	71
Zmienna $_POST przesyła dane z formularza do skryptu	72
Tworzenie treści listu w kodzie PHP	82
Nawet zwykły tekst można (trochę) sformatować	84
Znaki nowego wiersza trzeba umieścić w cudzysłowach	85
Zbuduj e-mail do Marka	86
Zmienne przechowują fragmenty e-maila	87
Wysyłanie e-maili przy użyciu PHP	88
Marek zaczyna otrzymywać e-maile	91
Marek zaczyna gubić e-maile	92

Łączenie się z bazą MySQL

2 Jak wszystko wiąże się ze sobą?

Ustalenie powiązań między elementami przed rozpoczęciem pracy to dobry pomysł. Utworzyłeś pierwszy działający skrypt PHP, jednak otrzymywanie danych z formularzy w formie listów elektronicznych już nie wystarcza. Potrzebny jest sposób na **zapisanie takich informacji**, aby można je ***przechowywać*** i w odpowiednim momencie ***pobierać***. Możesz użyć do tego **bazy danych MySQL**. Aby zapisać w niej dane, musisz uzyskać dostęp do bazy MySQL z poziomu skryptów PHP.

Formularz PHP Marka działa dobrze — aż za dobrze… 98
MySQL doskonale nadaje się do przechowywania danych 99
Marek potrzebuje bazy danych MySQL 100
Utwórz bazę danych i tabelę MySQL 102
Instrukcja INSERT w akcji 105
Użyj instrukcji SELECT do pobrania danych z tabeli 108
Użyj PHP do obsługi żmudnych instrukcji SQL 111
PHP umożliwia przepływ danych z formularza Marka 112
Połącz się z bazą danych w skrypcie PHP 114
Wstawianie danych za pomocą skryptu PHP 115
Użyj funkcji PHP do komunikacji z bazą danych 116
Łączenie się z bazą za pomocą mysqli_connect() 118
Budowanie zapytań INSERT w kodzie PHP 123
Kierowanie zapytań do bazy MySQL w skryptach PHP 124
Zamykanie połączenia za pomocą funkcji mysqli_close() 125
Zmienna $_POST udostępnia dane z formularza 129
Marek potrzebuje pomocy przy filtrowaniu danych 134
Marek jest na dobrej drodze do znalezienia Kła 136

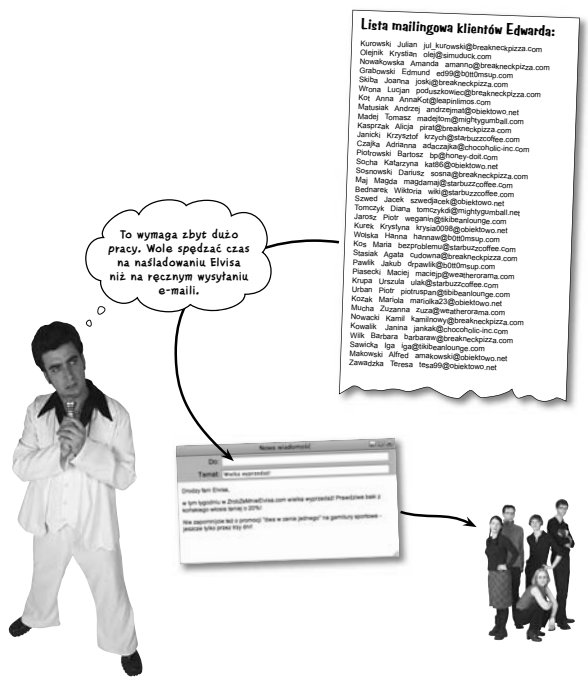


Tworzenie i zapełnianie bazy danych

3 Tworzenie własnych danych

Potrzebne informacje nie zawsze są dostępne. Czasem trzeba najpierw *utworzyć dane*. Nieraz należy też *przygotować tabele* do przechowywania informacji, a nawet *zbudować bazę danych*. Czujesz się zagubiony? Wkrótce wszystkiego się dowiesz. Zobaczysz, jak samodzielnie tworzyć bazy danych i tabele. Jeśli to wciąż za mało, przy okazji rozwiniesz swą pierwszą aplikację opartą na technologiach PHP i MySQL.

Sklep poświęcony Elvisowi otwiera podwoje 142
Edward potrzebuje aplikacji 143
Wizualizacja projektu aplikacji Edwarda 144
Wszystko zaczyna się od tabeli 147
Nawiąż kontakt z serwerem MySQL 148
Tworzenie bazy danych na adresy Edwarda 149
Tworzenie tabeli w bazie danych 150
Musimy zdefiniować dane 151
Poznaj wybrane typy danych języka MySQL 152
Tworzenie tabel przy użyciu zapytań 155
Wskaż bazę danych przed jej użyciem 158
Instrukcja DESCRIBE opisuje strukturę tabeli 161
Edward jest gotowy do zapisywania danych 163
Tworzenie skryptu do dodawania adresów 164
Druga strona aplikacji Edwarda 171
Mechanizmy skryptu do wysyłania wiadomości 172
Zacznijmy od początku — pobieranie danych 173
Funkcja mysqli_fetch_array() pobiera wyniki zapytania 174
Pętle WHILE 177
Przechodzenie po danych za pomocą pętli while 178
Otrzymałeś wiadomość od Edwarda! 183
Czasem użytkownicy chcą zrezygnować 184
Usuwanie danych za pomocą instrukcji DELETE 185
Użyj klauzuli WHERE, aby usunąć konkretne dane 186
Minimalizowanie ryzyka przypadkowych usunięć 187
ZrobZeMnieElvisa.com to aplikacja internetowa 192



Realistyczne i praktyczne aplikacje

4 Twoje aplikacje w sieci WWW

Czasem trzeba podejść do projektu realistycznie i jeszcze raz go przemyśleć. Możesz też poświęcić więcej czasu na zaplanowanie witryny na początku prac. Kiedy aplikacja znajdzie się już w sieci WWW, może się okazać, że projekt nie był doskonały. Rozwiązania, które na pozór powinny działać, czasem nie sprawdzają się w praktyce. W tym rozdziale rozwiążesz kilka *rzeczywistych problemów*, które mogą pojawić się przy **przenoszeniu aplikacji ze środowiska testowego na serwer WWW**. Przy okazji poznasz kilka ważnych fragmentów kodu PHP i SQL.

Niektórzy klienci Edwarda są zirytowani	196
Zabezpiecz Edwarda przed nim samym	199
Sprawdzaj poprawność danych z formularza	200
Walidacja w skrypcie sendemail.php	201
Kod podejmuje decyzje dzięki instrukcji IF	202
Sprawdzanie warunku	203
Instrukcja IF sprawdza nie tylko równość	204
Walidacja w skrypcie sendemail.php	207
Funkcje PHP do sprawdzania zawartości zmiennych	208
Sprawdzanie wielu warunków przy użyciu I oraz LUB	215
Użytkownicy formularza potrzebują informacji zwrotnych	219
Wygodne otwieranie i zamykanie bloków PHP	229
Użyj flagi, aby uniknąć powielania kodu	230
Napisz kod formularza jeden raz	231
Formularz autoreferencyjny	235
Wskaż skrypt w atrybucie action	236
Sprawdzanie, czy użytkownik przesłał formularz	238
Niektórzy użytkownicy nadal są niezadowoleni	242
Wiersze tabeli powinny mieć niepowtarzalne identyfikatory	244
Klucze główne wymuszają niepowtarzalność	246
Od pól wyboru do identyfikatorów klientów	251
Przechodzenie po elementach tablicy za pomocą foreach	252

Używanie danych przechowywanych w plikach

5 Kiedy baza danych nie wystarcza

Nie wierz przechwałkom — przynajmniej tym dotyczącym baz danych. To prawda, świetnie nadają się do przechowywania danych tekstowych, ale ***co z danymi binarnymi***, na przykład **rysunkami w formacie JPEG** lub **dokumentami PDF**? Czy ma sens zapisywanie zdjęć kolekcji rzadkich gitar w tabelach bazy danych? Zwykle nie. Jednak możesz jednocześnie mieć wirtualne ciastko i je zjeść. W tym rozdziale dowiesz się, jak **wykorzystać pliki i bazę danych do utworzenia aplikacji PHP**, która używa danych binarnych.

Wirtualni gitarzyści lubią współzawodnictwo 260
Rysunki to dowód prawdziwości wyniku 261
Aplikacja musi przechowywać rysunki 262
Planowanie przesyłania plików w aplikacji Gitarowe wojny 267
Trzeba zmodyfikować bazę rekordów za pomocą instrukcji ALTER 268
Jak pobrać rysunek od użytkownika? 272
Wstaw rysunki (nazwy plików) do bazy danych 274
Określanie nazwy przesłanego pliku 275
Gdzie znajdują się przesłane pliki? 280
Utwórz miejsce na przesłane pliki graficzne 284
Współużytkowane dane trzeba współdzielić 290
Dołączanie danych współużytkowanych przez skrypty 291
Traktuj instrukcję require_once jak polecenie „wstaw” 292
W przypadku rekordowych wyników najważniejszy jest porządek 294
Wyróżnianie najlepszego gitarowego wojownika 297
Sformatuj najlepszy wynik za pomocą kodu HTML i CSS 298
Przepuszczamy tylko małe rysunki 303
Walidacja pliku zwiększa niezawodność aplikacji 304
Projektowanie strony administracyjnej 308
Generowanie odsyłaczy do usuwania wyników na stronie administracyjnej 311
Skrypty mogą komunikować się ze sobą 312
O żądaniach GET i POST 314
GET, POST i usuwanie wyników 316
Znajdowanie usuwanych wyników 319
Kontrolowanie liczby usuwanych wierszy za pomocą klauzuli LIMIT 320

Zabezpieczanie aplikacji

6 Załóżmy, że każdy stanowi zagrożenie

Rodzice mieli rację — nie należy rozmawiać z nieznajomymi. A przynajmniej nie wolno im ufać. Z pewnością *nie dawaj im kluczy do danych z aplikacji z nadzieją*, że nie wykorzystają ich do złych celów. Świat bywa okrutny i nikomu nie należy ufać. Programista aplikacji sieciowych musi być nieco cyniczny i bardzo podejrzliwy. Tak, ludzie są przeważnie źli i z pewnością stanowią zagrożenie! W porządku, może trochę przesadzamy, jednak koniecznie **traktuj bezpieczeństwo poważnie** i **projektuj aplikacje tak, aby zabezpieczyć je** przed potencjalnymi napastnikami.

Dzień, w którym umarła muzyka	330
Gdzie się podziały wyniki?	331
Zabezpieczenia przed dzikimi hordami	333
Zabezpieczanie strony administracyjnej Gitarowych wojen	334
Uwierzytelnianie HTTP wymaga nagłówków	336
Kontrolowanie nagłówków za pomocą PHP	339
Uwierzytelnianie za pomocą nagłówków	340
Tworzenie skryptu do obsługi uwierzytelniania	348
Gitarowe wojny. Część II: Atak klonów rekordów	352
Odejmowanie przez dodawanie	353
Bezpieczeństwo wymaga interwencji człowieka	354
Planowanie obsługi moderowania w Gitarowych wojnach	355
Użyj instrukcji ALTER, aby zrobić miejsce na zatwierdzanie wyników	356
Niezatwierdzone wyniki nie są potrzebne	361
Oszustwo za milion punktów	364
Czy jest tu moderator?	365
Jak Ela to zrobiła?	367
Oszukiwanie bazy MySQL za pomocą komentarzy	368
Do formularza do dodawania wyników wstrzyknięto kod SQL	369
Chroń dane przed wstrzyknięciem kodu SQL	370
Bezpieczniejsza instrukcja INSERT (z parametrami)	371
Walidacji formularza nigdy za wiele	373
Wstrzymać ogień!	375

Tworzenie spersonalizowanych aplikacji sieciowych

7 Pamiętasz mnie?

Nikt — a zwłaszcza użytkownik aplikacji sieciowych — nie lubi, kiedy się o nim zapomina. Jeśli odwiedzający wchodzą w spersonalizowane interakcje z witryną, aplikacja powinna ich zapamiętywać. Przedstawianie się rodzinie po każdym powrocie do domu to dziwaczna perspektywa. Dzięki wspaniałej funkcji zwanej **pamięcią** nie musimy tego robić, jednak *aplikacje sieciowe nie zapamiętują automatycznie użytkowników*. To zmyślny programista musi użyć dostępnych narzędzi (na przykład PHP i MySQL), aby **utworzyć spersonalizowaną aplikację, która potrafi rozpoznawać odwiedzających**.

Podobno przeciwieństwa się przyciągają	380
Niedopasowanie opiera się na prywatnych danych	381
Witryna Niedopasowani potrzebuje loginów	382
Przygotowanie bazy do zapisywania loginów	385
Tworzenie interfejsu logowania	387
Szyfrowanie haseł za pomocą funkcji SHA()	388
Porównywanie haseł	389
Uwierzytelnianie użytkowników za pomocą HTTP	392
Logowanie się użytkowników za pomocą uwierzytelniania HTTP	395
Formularz do rejestracji nowych użytkowników	399
Co zawierają pliki cookie?	409
Zastosuj ciasteczka w kodzie PHP	410
Modyfikowanie przebiegu logowania	413
Logowanie oparte na plikach cookie	414
Wylogowanie to efekt usunięcia plików cookie	419
Sesje nie są zależne od klienta	423
Śledzenie danych sesji	425
Usprawnij aplikację Niedopasowani za pomocą sesji	426
Wylogowywanie przy użyciu sesji	427
Dokończ wprowadzanie sesji	432
Użytkownicy nie czują się mile witani	438
Sesje nie żyją długo…	440
…ale pliki cookie mogą trwać wiecznie!	441
Sesje + pliki cookie = większa trwałość logowania	443

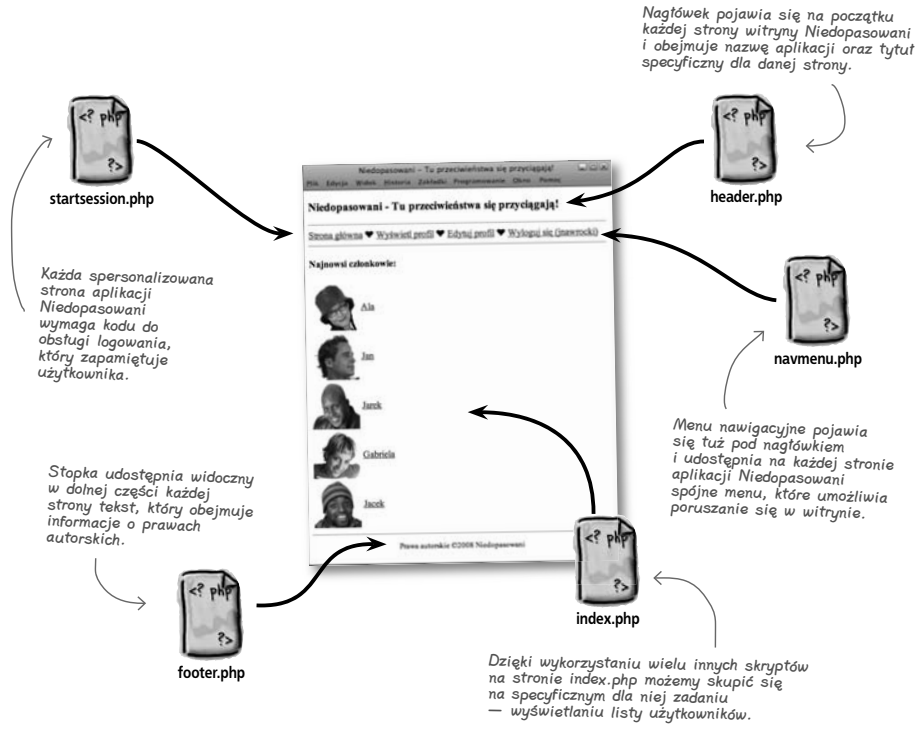
Eliminowanie powtórzeń w kodzie

7½

Współdzielenie oznacza troskę

Parasole to nie jedyna rzecz, którą można się dzielić. W każdej aplikacji sieciowej natrafisz *na kod, który powtarza się w kilku miejscach*. Nie tylko jest to niepotrzebne, ale ponadto prowadzi do ***problemów z konserwacją***, ponieważ w przyszłości z pewnością będziesz musiał wprowadzić zmiany w wielu różnych miejscach aplikacji. Rozwiązanie polega na **wyeliminowaniu powtórzeń w kodzie przez jego współużytkowanie**. Oznacza to, że należy umieścić powielany fragment w jednym miejscu, a następnie korzystać z tego kodu za każdym razem, kiedy jest potrzebny. Eliminacja powtarzającego się kodu pozwala tworzyć aplikacje, którą są **wydajniejsze**, **łatwiejsze w konserwacji** i **bardziej niezawodne**.

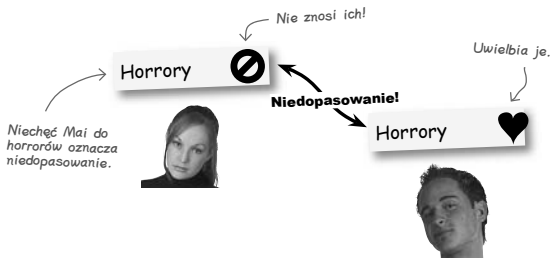
Aplikacja Niedopasowani w kawałkach 455
Przebudowywanie aplikacji Niedopasowani przy użyciu szablonu 456
Przebudowa aplikacji Niedopasowani z wykorzystaniem szablonów 458
Zupełnie nowa i dużo lepiej uporządkowana aplikacja Niedopasowani 460

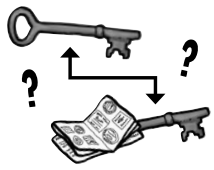


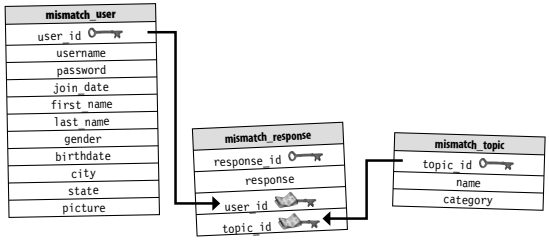
Kontroluj dane — kontroluj swój świat

8 Zbieranie danych

Nie ma nic lepszego niż udane żniwa danych. Możesz uzyskać w ten sposób dostęp do mnóstwa informacji gotowych do ***analizowania***, ***sortowania***, ***porównywania***, ***łączenia*** i wykonywania innych zadań obsługiwanych przez Twoją fantastyczną aplikację sieciową. Przyjemne? O tak. Jednak **uzyskanie kontroli nad danymi w bazie MySQL** — podobnie jak prawdziwe żniwa — wymaga ciężkiej pracy i dużej wiedzy. Użytkownicy wymagają czegoś więcej niż przestarzałych i nudnych danych. Chcą informacji, które wzbogacają ich wiedzę, dają satysfakcję i są ważne. Na co więc czekasz? Uruchom traktor MySQL i ruszaj do pracy!







Idealnie niedopasowani	462
Aplikacja Niedopasowani opiera się na danych	463
Utwórz model bazy danych za pomocą schematu	465
Łączenie kilku tabel	470
Klucze zewnętrzne w praktyce	471
Jeden wiersz pasuje do jednego wiersza	472
Jeden wiersz łączy się z wieloma	473
Wiersze w relacji wiele do wielu	474
Tworzenie kwestionariusza na potrzeby aplikacji Niedopasowani	479
Zapisywanie odpowiedzi w bazie danych	480
Możemy sterować formularzem za pomocą danych	484
Generowanie formularza z kwestionariuszem w aplikacji Niedopasowani	490
Droga do normalności	496
W czasie normalizacji myśl w kategoriach atomów	497
Trzy kroki do znormalizowanej bazy danych	499
Modyfikowanie bazy aplikacji Niedopasowani	503
Czy baza aplikacji Niedopasowani jest naprawdę znormalizowana?	504
Zapytanie w zapytaniu wewnątrz zapytania…	506
Złączmy tabele	507
Łączenie za pomocą kropek	508
To jeszcze nie wszystkie możliwości złączeń wewnętrznych	509
Pseudonimy tabel i kolumn	511
Pomoc ze strony złączeń	512
Pięć kroków do udanego niedopasowania	519
Porównywanie osób pod kątem niedopasowania	521
Potrzebujemy pętli FOR	522

Funkcje niestandardowe i do obsługi łańcuchów znaków

9 Dzięki funkcjom żyje się lepiej

Funkcje pozwalają znacznie podnieść poziom aplikacji. Używałeś już funkcji języka PHP do wykonywania różnych zadań. W tym rozdziale poznasz kilka innych **funkcji wbudowanych**. Następnie nauczysz się tworzyć własne, **niestandardowe funkcje**, które pozwolą Ci uzyskać niewyobrażalne efekty. Może nie uda Ci się wyhodować w ten sposób laserowych rekinów, ale niestandardowe funkcje usprawniają kod i umożliwiają jego wielokrotne wykorzystanie.

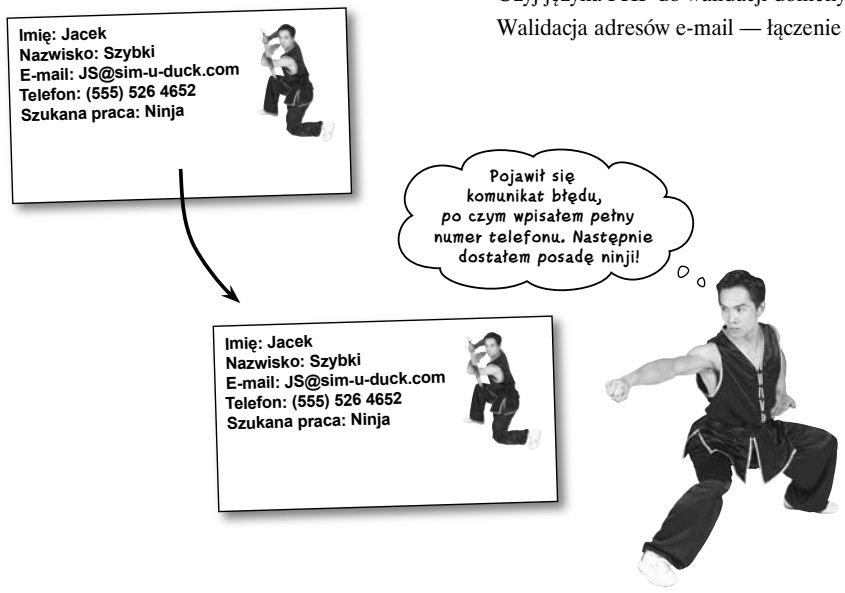
Trudno jest znaleźć dobrą ryzykowną pracę	534
Wyszukiwanie nie daje możliwości popełnienia błędu	536
Dzięki słowu kluczowemu LIKE zapytania SQL mogą być elastyczne	537
Podziel łańcuch znaków na pojedyncze słowa	542
Funkcja implode() tworzy łańcuch znaków przy użyciu podłańcuchów	545
Wstępne przetwarzanie szukanego łańcucha znaków	551
Zastępowanie niepotrzebnych znaków w szukanym tekście	552
Potrzebujemy poprawnych szukanych łańcuchów	556
Kopiowanie niepustych elementów do nowej tablicy	557
Czasem potrzebny jest tylko fragment łańcucha	560
Pobieranie podłańcuchów z obu stron tekstu	561
Można posortować wyniki w kilku zapytaniach	564
Funkcje umożliwiają wielokrotne wykorzystanie kodu	568
Zbuduj zapytanie za pomocą niestandardowej funkcji	569
SWITCH obsługuje więcej decyzji niż IF	574
Sortowanie w funkcji build_query()	577
Możemy podzielić wyniki na strony	580
Pobieranie tylko potrzebnych wierszy dzięki klauzuli LIMIT	581
Zarządzanie odnośnikami związanymi z klauzulą LIMIT	582
Kontrolowanie danych potrzebnych do obsługi stronicowania	583
Konfigurowanie zmiennych używanych do stronicowania	584
Poprawianie zapytania, które pobiera stronicowane wyniki	585
Generowanie odnośników do nawigacji po stronach	586
Tworzenie kompletnego skryptu do wyszukiwania ofert	589
Kompletny skrypt do wyszukiwania ofert — ciąg dalszy	590

Wyrażenia regularne

10 Reguły zastępowania

Funkcje do obsługi łańcuchów znaków są bardzo przydatne, ale mają pewne ograniczenia. Oczywiście, potrafią określić długość tekstu, skrócić go i zastąpić wybrane znaki innymi. Jednak czasem potrzebne są bardziej złożone operacje związane z tekstem. Wtedy pomocne są **wyrażenia regularne**. Przy ich użyciu można **precyzyjnie modyfikować łańcuchy znaków** na podstawie **zestawu reguł**, a nie pojedynczego kryterium.

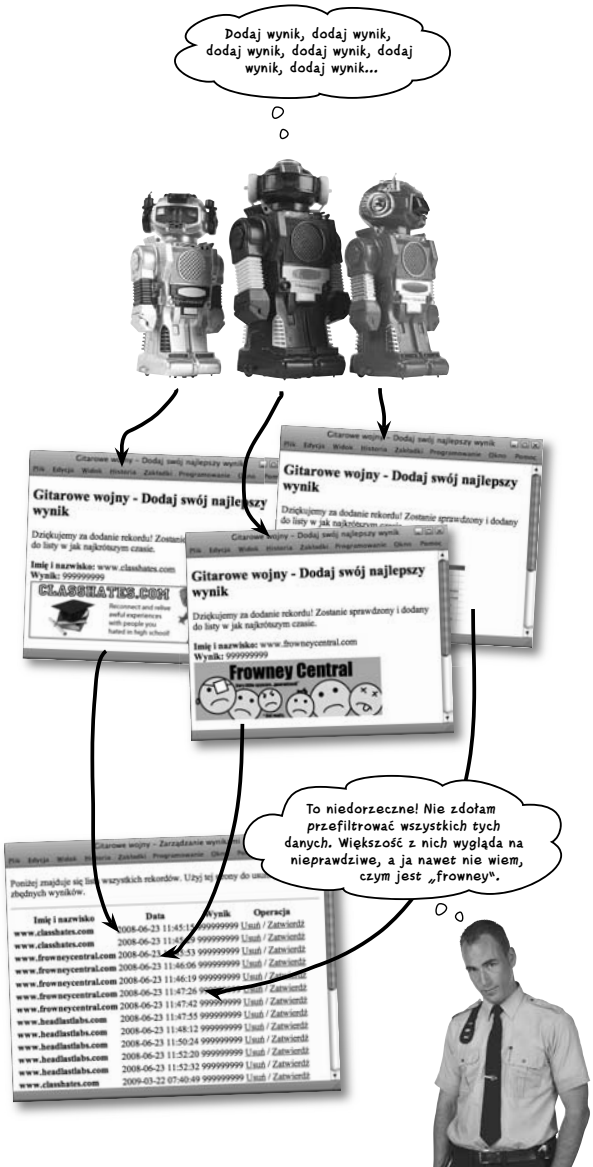
Użytkownicy serwisu Ryzykowne prace mogą zamieszczać w nim życiorysy 594
Określ, jaki format powinny mieć dane 598
Utwórz wzorzec numerów telefonów 601
Dopasuj dane do wzorca za pomocą wyrażeń regularnych 602
Tworzenie wzorców za pomocą metaznaków 604
Dopracuj wzorce za pomocą klas znaków 611
Wyszukiwanie wzorców za pomocą funkcji preg_match() 616
Ustandaryzuj numery telefonów 623
Usuwanie niepożądanych znaków 624
Dopasowywanie adresów e-mail może być trudne 628
Przyrostki domen są wszędzie 630
Użyj języka PHP do walidacji domeny 631
Walidacja adresów e-mail — łączenie wszystkich elementów 632



Wizualizowanie danych i inne zagadnienia

11 Dynamiczne dodawanie grafiki

Wszyscy wiemy, jakie możliwości dają odpowiednie zapytania i zbiory wyników. Jednak dane nie zawsze są w pełni zrozumiałe. Czasem warto przedstawić je w innym, bardziej wizualnym świetle. PHP umożliwia wyświetlanie **graficznej reprezentacji danych z bazy** w postaci ***wykresów kołowych***, ***wykresów słupkowych***, ***diagramów Venna***, ***plam Rorschacha*** i tak dalej. Pomaganie użytkownikom w zrozumieniu danych to swego rodzaju gra. Jednak nie wszystkie wartościowe grafiki w aplikacjach PHP pochodzą z bazy danych. Czy wiedziałeś, że **za pomocą dynamicznie generowanych rysunków można powstrzymać roboty spamujące**?

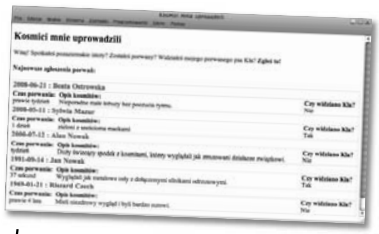


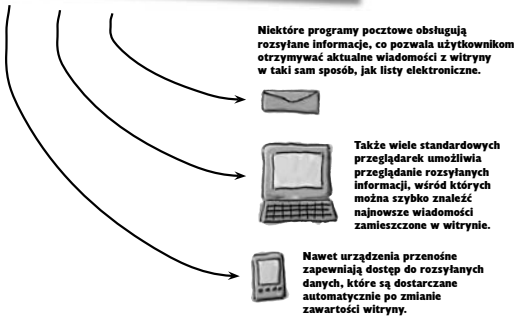
Powrót Gitarowych wojen: narodziny maszyn	638
Żaden formularz na dane wejściowe nie jest bezpieczny	639
Musimy oddzielić ludzi od maszyn	640
Możemy pokonać automatyzację automatyzacją	643
Generowanie tekstu hasła w teście CAPTCHA	645
Generowanie rysunku z testem CAPTCHA	646
Funkcje graficzne biblioteki GD	648
Wyświetlanie tekstu za pomocą innych czcionek	652
Generowanie losowych rysunków z testami CAPTCHA	655
Przywracanie normalności w Gitarowych wojnach	657
Dodawanie testu CAPTCHA do skryptu do dodawania wyników	659
Pięć wymiarów niedopasowania	662
Diagramy niedopasowania	663
Zapisywanie danych na potrzeby wykresu	664
Od jednej tablicy do drugiej	668
Tworzenie tablicy niedopasowanych tematów	670
Plan tworzenia wykresu	671
Analizowanie kategorii	672
Określanie liczby kategorii	673
Podstawy rysowania wykresów	676
Rysowanie i wyświetlanie rysunku z wykresem słupkowym	679
Osobne rysunki z wykresami dla wszystkich	682
Użytkownicy serwisu Niedopasowani polubili wykresy	685

Rozpowszechnianie danych i usługi sieciowe

12 Interfejs do komunikacji ze światem

Świat jest wielki, czego nie można ignorować w czasie tworzenia aplikacji sieciowych. Nie chcesz przecież, aby użytkownicy zignorowali Twoją witrynę. Doskonałym sposobem na dostrojenie aplikacji do świata jest rozpowszechnianie danych. Technika ta sprawia, że użytkownicy mogą subskrybować informacje i otrzymywać je bez konieczności odwiedzania witryny. Ponadto aplikacja może stanowić interfejs do innych programów (służą do tego usługi sieciowe) i korzystać z danych z zewnętrznych serwisów, co otwiera przed programistą dodatkowe możliwości.





Marek musi poinformować świat o zaginięciu Kła	688
Przesyłanie danych o porwaniach do użytkowników	689
RSS pozwala przesyłać zawartość witryny do użytkowników	690
RSS to w rzeczywistości XML	691
Z bazy danych do czytnika kanałów RSS	696
Wizualizowanie kodu RSS	699
Dynamiczne generowanie kanału RSS	702
Odnośnik do kanału RSS	706
Jeden film jest wart miliona słów	708
Pobieranie materiałów od innych	710
Rozpowszechnianie filmów z YouTube	711
Przesyłanie żądania danych o filmach do serwisu YouTube	712
Marek jest gotów do utworzenia żądania REST	716
YouTube komunikuje się w języku XML	720
Analiza kodu XML odpowiedzi z serwisu YouTube	724
Wizualizowanie danych XML z opisem filmów	725
Dostęp do danych XML za pomocą obiektów	726
Od elementów XML do obiektów PHP	727
Pobieranie danych XML za pomocą obiektów	728
Nie zapominaj o przestrzeniach nazw!	729
Obserwatorzy zauważyli Kła	731
Rozmieść wyświetlane filmy	732
Formatowanie danych o filmach w celu ich wyświetlenia	733

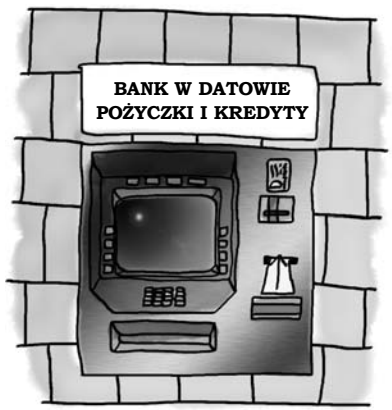
Pozostałości

A

Dziesięć najważniejszych tematów (których nie poruszyliśmy)

Nawet po omówieniu wszystkiego zawsze znajdą się jakieś dodatkowe informacje. Warto, żebyś poznał jeszcze kilka innych zagadnień. Uznaliśmy, że nie można ich pominąć, choć wymagają tylko krótkiej wzmianki. Dlatego zanim odłożysz książkę, zapoznaj się z tymi krótkimi, ale ważnymi ciekawostkami na temat języków PHP i MySQL. Ponadto skoro już doszedłeś do tego miejsca, zostało Ci do końca tylko kilka niedługich dodatków… indeks… może kilka reklam… i to będzie już naprawdę koniec. Obiecujemy!

Numer 1. Dopasowywanie kodu do języka PHP 4 i funkcji mysql 744
Numer 2. Uprawnienia użytkowników w MySQL 746
Numer 3. Zgłaszanie błędów w MySQL 748
Numer 4. Obsługa wyjątków w kodzie PHP 749
Numer 4. Obsługa wyjątków w PHP (ciąg dalszy) 750
Numer 5. Obiektowy język PHP 751
Numer 5. Obiektowy język PHP (ciąg dalszy) 752
Numer 6. Zabezpieczanie aplikacji PHP 753
Numer 6. Zabezpieczanie aplikacji PHP (ciąg dalszy) 754
Numer 7. Zabezpieczanie aplikacji przed atakami XSS 755
Numer 7. Zabezpieczanie aplikacji przed atakami XSS (ciąg dalszy) 756
Numer 8. Pierwszeństwo operatorów 757
Numer 9. Czym różni się język PHP 5 od PHP 6? 758
Numer 9. Czym różni się język PHP 5 od PHP 6? (ciąg dalszy) 759
Numer 10. Wykorzystanie kodu PHP innych osób 760



Konfigurowanie środowiska programistycznego

Miejsce do zabawy

Potrzebujesz miejsca, w którym będziesz mógł przetestować nowe umiejętności z obszaru technologii PHP i MySQL bez narażania danych na ataki w sieci WWW. Zawsze warto mieć bezpieczny obszar do rozwijania aplikacji PHP przed udostępnieniem ich światu. W tym dodatku dowiesz się, jak zainstalować serwer WWW, bazę danych MySQL i język PHP, aby przygotować bezpieczne miejsce do pracy oraz ćwiczeń.

Utwórz środowisko do programowania w języku PHP 762
Sprawdź dostępne komponenty 762
Czy masz serwer WWW? 763
Czy masz PHP? W której wersji? 763
Czy masz zainstalowany serwer MySQL? Którą wersję? 764
Zacznij od serwera WWW 765
Instalowanie serwera Apache — zakończenie 766
Instalowanie PHP 766
Etapy instalowania PHP 767
Etapy instalowania PHP — zakończenie 768
Instalowanie MySQL 768
Etapy instalowania serwera MySQL w systemie Windows 769
Włączanie PHP w systemie Mac OS X 772
Etapy instalowania MySQL w systemie Mac OS X 772
Przenoszenie kodu ze środowiska produkcyjnego do publicznie dostępnej witryny 774
Zrób zrzut danych (i tabel) 775
Przygotowania do użycia zrzutu danych 775
Przenoszenie zrzuconych danych na serwer publiczny 776
Łączenie się z serwerem publicznym 777

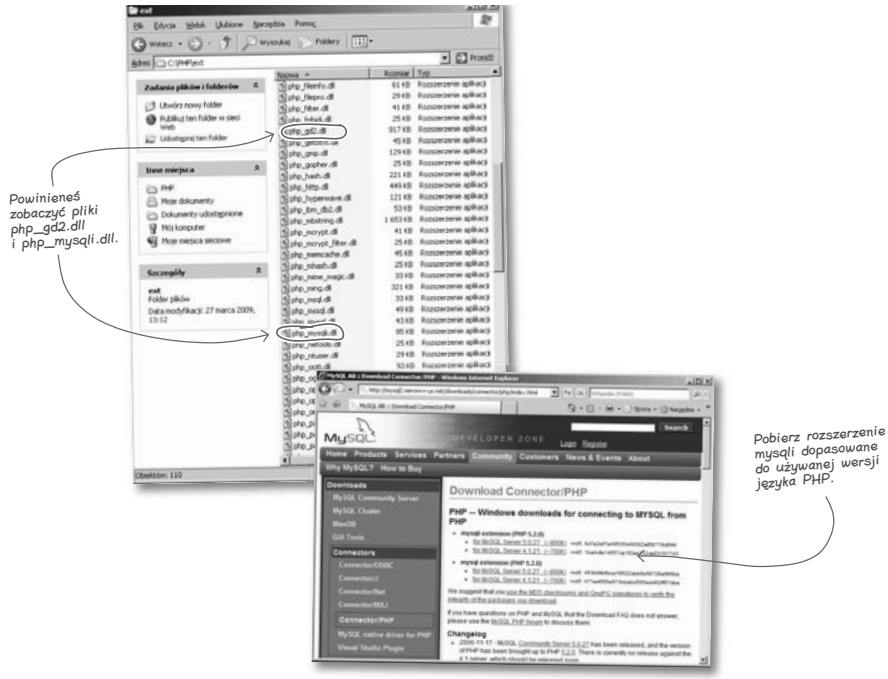
Rozszerzanie PHP

C

Jeszcze więcej możliwości

To prawda, że za pomocą PHP i MySQL możesz tworzyć doskonałe aplikacje sieciowe. Jednak wiesz, że musi istnieć coś jeszcze. Rzeczywiście tak jest. W tym krótkim dodatku zobaczysz, jak zainstalować rozszerzenie mysqli i bibliotekę graficzną GD. Opiszemy też kilka innych wartościowych dodatków do języka PHP. Czasem nie ma nic złego w tym, że chcemy czegoś więcej.

Rozszerzanie języka PHP 780
A w komputerach Mac… 783



S

Skorowidz

785

Dzięki funkcjom żyje się lepiej

Funkcje pozwalają znacznie podnieść poziom aplikacji. Używałeś już funkcji języka PHP do wykonywania różnych zadań. W tym rozdziale poznasz kilka innych **funkcji wbudowanych.** Następnie nauczysz się tworzyć własne, **niestandardowe funkcje**, które pozwolą Ci uzyskać niewyobrażalne efekty. Może nie uda Ci się wyhodować w ten sposób laserowych rekinów, ale niestandardowe funkcje usprawniają kod i umożliwiają jego wielokrotne wykorzystanie.

Trudno jest znaleźć dobrą ryzykowną pracę

Nowy serwis internetowy RyzykownePrace.biz ma pomóc firmom w znalezieniu odpowiednich pracowników do wykonywania niebezpiecznych zajęć. Model biznesowy witryny jest prosty — otrzymujemy od przedsiębiorstw prowizję za każdego znalezionego kandydata. Im więcej pracowników zapewnimy firmom, tym większe będą zyski.

Serwis wymaga usprawnienia mechanizmu wyszukiwania ofert pracy. Obecnie mamy bazę danych pełną ryzykownych stanowisk, które tylko czekają na odkrycie przez odpowiednie osoby. Przyjrzyjmy się formularzowi wyszukiwania informacji w serwisie Ryzykowne prace i bazie danych z dostępnymi ofertami.

Ten prosty formularz wywołuje skrypt, który przeszukuje tabelę riskyjobs.

Formularz wyszukiwania w serwisie Ryzykowne prace uruchamia zapytanie do tabeli riskyjobs, które pobiera odpowiednie oferty.

Tabela riskyjobs zawiera nazwy stanowisk i ich opisy, a także miejsce wykonywania pracy oraz datę udostępnienia każdej oferty.

riskyjobs

job_id	title	description	city	state	zip	company	date_posted
1	Matador	Rozwijająca się mleczarnia…	Szczecin	ZP	70-950	Mleczarnia szalone krowy	2008-03-11 10:51:24
2	Paparazzi	Czołowa firma fotografująca…	Opole	OP	45-087	W pogoni za gwiazdami	2008-03-24 10:51:24
3	Treser rekinów	Uczenie rekinów ciekawych…	Gliwice	SL	44-100	Rekinarium	2008-04-28 03:12:45
4	Strażak	Miejscowość Katowice …	Katowice	SL	40-012	Miasto Katowice	2008-05-22 12:34:17
5	Kontroler napięcia	Praca w terenie, która…	Radom	MZ	26-610	Szok S.A.	2008-06-28 11:16:30
6	Dentysta krokodyli	Czy kochasz zwierzęta?…	Warszawa	MZ	02-200	Żarłoczne gady	2008-07-14 10:51:24
7	Chodzenie po kremie	Potrzebujemy osób…	Kalisz	WP	62-800	Kremowe technologie	2008-07-14 10:54:05
8	Serwisant elektrycznych byków	Bar u Henia poszukuje…	Olsztyn	WM	10-254	Bar u Henia	2008-07-27 11:22:28
			…				

Każda oferta ma niepowtarzalny identyfikator w postaci klucza głównego job_id.

Wyświetlanie wyników wyszukiwania!

Zaostrz ołówek

Przy przesyłaniu formularza w witrynie Ryzykowne prace łańcuch znaków z zapytaniem jest zapisywany w zmiennej `$user_search`. Skrypt wstawia tę zmienną do poniższego zapytania SQL, które uruchamia wyszukiwanie. Napisz, ile wierszy z bazy `riskyjobs` z poprzedniej strony zostanie zwróconych przez zapytanie Antoniego.

```
$search_query = "SELECT job_id, title, state, description FROM riskyjobs " .
  "WHERE title = '$user_search'";
$result = mysqli_query($dbc, $search_query);
```

Tu wpisz odpowiedź! .....

Zaostrz ołówek: Rozwiązanie

Przy przesyłaniu formularza w witrynie Ryzykowne prace łańcuch znaków z zapytaniem jest zapisywany w zmiennej `$user_search`. Skrypt wstawia tę zmienną do poniższego zapytania SQL, które uruchamia wyszukiwanie. Napisz, ile wierszy z bazy `riskyjobs` z poprzedniej strony zostanie zwróconych przez zapytanie Antoniego.

Klauzula WHERE ze znakiem = oznacza, że dwa porównywane łańcuchy muszą być dokładnie takie same.

Ta zmienna zawiera dane wpisane w polu tekstowym.

```
$search_query = "SELECT job_id, title, state, description FROM riskyjobs " .
  "WHERE title = '$user_search'";
$result = mysqli_query($dbc, $search_query);
```

Nic, zero, brak wyników! Problem polega na tym, że zapytanie jest zbyt wymagające — użytkownik musi wpisać dokładną nazwę stanowiska.

0

Wyszukiwanie nie daje możliwości popełnienia błędu

Zapytanie `SELECT` w skrypcie witryny Ryzykowne prace jest bardzo ścisłe i zwraca wyniki tylko wtedy, jeśli porównywane łańcuchy są **identyczne**. Utrudnia to przeszukiwanie ofert. Użytkownicy powinni móc uzyskać wyniki także wtedy, jeżeli nie wpiszą dokładnej nazwy stanowiska.

Wróćmy do wyszukiwania Antoniego. Wpisane dane powodują uruchomienie zapytania, które szuka w kolumnie `title` tabeli `riskyjobs` tekstu „pogromca byków matador”.

Wielkość znaków w szukanym tekście nie ma znaczenia, ponieważ klauzula WHERE języka MySQL domyślnie nie uwzględnia tej cechy.

```
SELECT job_id, title, description FROM riskyjobs

WHERE title = 'pogromca byków matador'
```

Operator = wymaga, aby porównywane łańcuchy znaków były identyczne.

Dostrzegasz problem? Zapytanie znajdzie w tabeli tylko te wiersze, w których tabela `title` zawiera tekst „pogromca byków matador”. Nie zwróci stanowisk o nazwie „matador” lub „serwisant elektrycznych byków”. To prawda, ta ostatnia posada nie interesuje Antoniego, ale wyszukiwanie nie działa tak, jak powinno. Problemu nie stanowi wielkość znaków (proces wyszukiwania danych w języku MySQL domyślnie **nie uwzględnia wielkości liter**). Przyczyną jest fakt, że do nazwy stanowiska musi pasować cały łańcuch znaków, co wynika z użycia w klauzuli `WHERE` operatora równości (`=`).

Dzięki słowu kluczowemu LIKE zapytania SQL mogą być elastyczne

Potrzebny jest sposób na przeszukiwanie bazy danych w celu dopasowania tylko fragmentu podanego łańcucha. SQL udostępnia do tego słowo kluczowe `LIKE`, które zwiększa elastyczność zapytań z klauzulą `WHERE`. Możesz traktować to słowo jak bardziej „wyrozumiałą" wersję operatora `=`. Przyjrzyj się poniższemu zapytaniu. Użyliśmy w nim słowa kluczowego `LIKE` do dopasowania wierszy, w których w **dowolnym miejscu** kolumny `title` pojawia się słowo „kier".

```
SELECT job_id, title, description FROM riskyjobs

WHERE title LIKE '%kier%'
```

Słowo kluczowe LIKE pozwala znaleźć dane, które nie są identyczne ze słowem podanym w apostrofach. Wyszukiwanie także tu nie uwzględnia wielkości znaków.

% to symbole wieloznaczne, które zastępują dowolne znaki występujące przed danym słowem lub po nim.

Słowo kluczowe `LIKE` znacznie ułatwia wyszukiwanie, zwłaszcza jeśli chcesz dopasować tekst, który jest częścią dłuższego słowa lub zwrotu. Przyjrzyj się przykładowym łańcuchom, które pasują do powyższego zapytania:

Kierownik

Bankier

Cukiernik

W klauzuli `LIKE` zwykle pojawiają się **symbole wieloznaczne**, które zastępują znaki w dopasowywanych danych. W języku SQL symbol wieloznaczny w postaci znaku procentów (`%`) oznacza dowolną grupę znaków. Umieszczenie tego symbolu w zapytaniu przed szukanym tekstem lub po nim (jak we wcześniejszej instrukcji `SELECT`) informuje SQL, że należy pobrać dane, w których pojawia się dany fragment. Nie jest istotne, ile znaków znajduje się przed nim lub po nim.

Dla zaawansowanych

SQL udostępnia też inny symbol wieloznaczny, którego można użyć w klauzuli LIKE. Jest to podkreślenie (_), które reprezentuje pojedynczy znak. Przyjrzyj się poniższej klauzuli LIKE:

```
LIKE '%straż_ _'
```

Klauzula ta oznacza: „Znajdź poprzedzony dowolnymi symbolami łańcuch »straż«, po którym następują dwa znaki". Wyrażenie to pasuje do słów „strażak" i „starszy strażak", ale już nie do wyrazu „strażnik".

Przerwa! Poświęć chwilę na zapoznanie się z bazą serwisu Ryzykowne prace i uruchom kilkakrotnie wyszukiwanie.

Pobierz z FTP wydawnictwa Helion (*ftp://ftp.helion.pl/przyklady/hfphms.zip*) plik *riskyjobs.sql* aplikacji Ryzykowne prace. Zawiera on instrukcje SQL, które utworzą tabelę `riskyjobs` i zapełnią ją przykładowymi danymi.

Po uruchomieniu w narzędziu do obsługi MySQL instrukcji z pliku *riskyjobs.sql* wypróbuj kilka zapytań, aby zasymulować wyszukiwanie stanowisk. Poniżej znajdziesz kilka poleceń, od których możesz zacząć.

```
SELECT * FROM riskyjobs
```

To zapytanie pobiera wszystkie kolumny wszystkich wierszy z ofertami z tabeli riskyjobs.

```
SELECT job_id, title, description FROM riskyjobs
WHERE title = 'matador pogromca byków'
```

To zapytanie pobiera identyfikator oferty, nazwę i opis stanowiska z wierszy, w których kolumna title ma wartość „matador pogromca byków".

```
SELECT job_id, title, description FROM riskyjobs
WHERE description LIKE '%zwierz%'
```

W tym zapytaniu użyliśmy klauzuli LIKE, aby znaleźć stanowiska, w których opisie pojawia się słowo „zwierz".

Pobierz pliki!

Kompletny kod źródłowy aplikacji Ryzykowne prace znajdziesz na FTP wydawnictwa Helion:

ftp://ftp.helion.pl/przyklady/hfphms.zip

Magnesiki z klauzulą LIKE

Do lodówki przyczepionych jest kilka klauzul `LIKE`. Czy potrafisz dopasować je do odpowiednich wyników? Które magnesiki nie odpowiadają żadnej z podanych klauzul `LIKE`?

Niektórym klauzulom może odpowiadać kilka wyników.

LIKE '%er'

LIKE '% T%'

LIKE 'c%'

LIKE '%ator %'

LIKE '%Tipper Cow%'

LIKE '%do_'

LIKE '%ma%'

Operator Armaty

Komandos

Młodszy Tester

Klown na Rodeo

Matador

Polityk

Chomików Treser

Chytry Bankier

Czujny Oficer

Szmugler

Podwodny Ankieter

Magnesiki z klauzulą LIKE. Rozwiązanie

Do lodówki przyczepionych jest kilka klauzul LIKE. Czy potrafisz dopasować je do odpowiednich wyników? Które magnesiki nie odpowiadają żadnej z podanych klauzul LIKE?

Do tego zapytania pasują wszystkie nazwy z końcówką „er".

LIKE '%er'

- Młodszy Tester
- Podwodny Ankieter
- Chomików Treser
- Chytry Bankier
- Szmugler
- Czujny Oficer

W pasujących nazwach przed literą „T" musi wystąpić odstęp.

LIKE '% T%'

- Młodszy Tester
- Chomików Treser

Wielkość znaków nie ma znaczenia.

LIKE '%ator %'

- Operator Armaty

W zapytaniach SQL wielkość znaków jest nieistotna, dlatego to zapytanie pasuje do nazw z małą i dużą literą „c".

LIKE 'c%'

- Chomików Treser
- Chytry Bankier
- Czujny Oficer

Po literach „do" może pojawić się tylko jeden znak.

LIKE '%do_'

- Matador
- Komandos

W dowolnym miejscu pasującej nazwy muszą pojawić się litery „ma".

LIKE '%ma%'

- Operator Armaty
- Komandos
- Matador

Te nazwy nie pasują do żadnego zapytania.

- Klown na Rodeo
- Polityk

Ta klauzula LIKE nie odpowiada żadnej nazwie.

LIKE '%Treser Chomików%'

Ostatnia klauzula LIKE, LIKE '%Treser Chomików%', nie pasuje do żadnego stanowiska, ponieważ słowa „Treser" i „Chomików" nigdzie nie występują razem w tej kolejności. Wygodny byłby podział szukanych wyrażeń na słowa i wyszukiwanie poszczególnych wyrazów.

Zapytanie będzie działało dużo lepiej, jeśli będzie wyszukiwało pojedyncze słowa „Treser" i „Chomików" zamiast cały łańcuch „Treser Chomików".

To naprawdę wygodne! Musimy tylko ustalić, jak wyszukać każde z wpisanych słów.

Dokładne dopasowywanie całego tekstu z pola wyszukiwania aplikacji Ryzykowne prace nie zawsze pozwala uzyskać pożądane wyniki. Aby zwiększyć skuteczność aplikacji, należy wyszukiwać poszczególne słowa, a nie całe łańcuchy znaków. Jak jednak znaleźć wiersze danych na postawie kilku odrębnych wyrazów? Możemy zapisać każdy z nich w tablicy, a następnie zmodyfikować zapytanie SELECT, aby wyszukiwało dane na podstawie poszczególnych słów.

Podziel łańcuch znaków na pojedyncze słowa

Aby mechanizm wyszukiwania w aplikacji Ryzykowne prace był skuteczniejszy, należy podzielić wpisany przez użytkownika łańcuch znaków, jeśli składa się z kilku słów. Osoby szukające pracy wprowadzają dane tekstowe, co oznacza, że możemy użyć funkcji wbudowanych języka PHP do ich przetworzenia. Jednym z przydatnych narzędzi tego typu jest funkcja `explode()`. Dzieli ona łańcuch znaków na fragmenty. Oto przykład jej działania:

Funkcja explode() dzieli łańcuch znaków na tablicę podłańcuchów.

Funkcja explode() dzieli łańcuch znaków na tablicę podłańcuchów na podstawie separatora (nazywanego też ogranicznikiem).

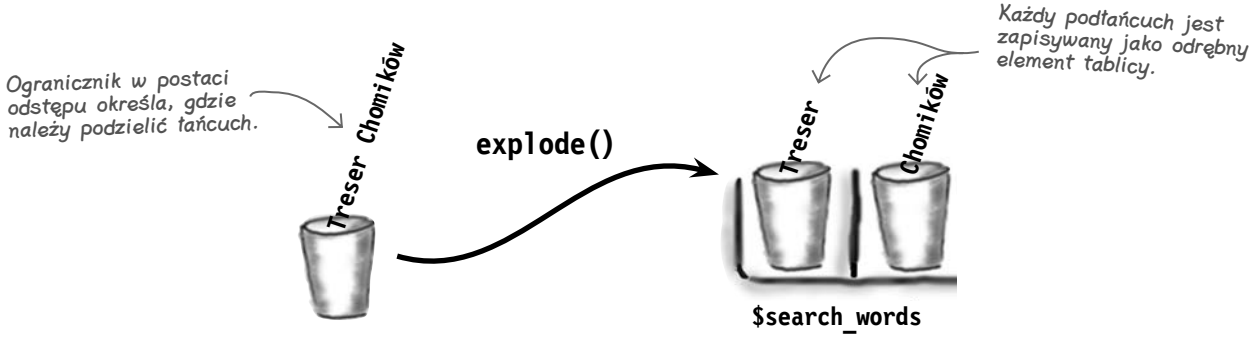
Ten parametr to tekst, który chcemy podzielić.

```
$search_words = explode(' ', 'Treser Chomików');
```

Zmienna $search_words zawiera tablicę szukanych słów, której użyjemy w zapytaniu SQL.

Ten parametr informuje funkcję explode(), co oddziela podłańcuchy w tekście (tu jest to odstęp). Można podać tu kilka znaków. Noszą one nazwę ograniczników.

Funkcja `explode()` wymaga podania dwóch parametrów. Pierwszy z nich to **ogranicznik**, czyli znak (lub ich zbiór), który określa, **gdzie** należy podzielić łańcuch. W przedstawionym kodzie separatorem jest odstęp, co oznacza, że funkcja podzieli tekst w miejscach wystąpienia spacji. Same ograniczniki są pomijane w wynikowych podłańcuchach. Drugi parametr to dzielony tekst.



Utworzenie tablicy szukanych słów w aplikacji Ryzykowne prace wymaga dodania wiersza kodu przed uruchomieniem zapytania. Teraz jeśli użytkownik wpisze w polu wyszukiwania tekst „Treser Chomików”, kod podzieli go na dwa słowa i zapisze każde z nich w tablicy `$search_words`.

Funkcja explode() zapisuje każde słowo ze zmiennej $user_search w tablicy $search_words.

```
$user_search = $_GET['usersearch'];
$search_words = explode(' ', $user_search);
```

Ćwiczenie

Aby wyszukać poszczególne słowa w bazie aplikacji Ryzykowne prace, trzeba umieścić każde z nich w zapytaniu SELECT języka SQL. Można to zrobić za pomocą słów kluczowych LIKE i OR. Tak powinno wyglądać zapytanie, jeśli Antoni wpisze tekst „matador pogromca byków".

Teraz przeszukujemy opisy stanowisk zamiast nazw, ponieważ w opisie można znaleźć więcej informacji.

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%matador%' OR description LIKE '%pogromca%' OR
  description LIKE '%byków%'
```

Załóżmy, że użyliśmy poniższego kodu PHP do utworzenia tego zapytanie na podstawie danych wpisanych przez użytkownika w formularzu wyszukiwania w aplikacji Ryzykowne prace:

```
$search_query = "SELECT * FROM riskyjobs";
$where_clause = '';
$user_search = $_GET['usersearch'];
$search_words = explode(' ', $user_search);
foreach ($search_words as $word) {
  $where_clause .= " description LIKE '%$word%' OR ";
}

if (!empty($where_clause)) {
  $search_query .= " WHERE $where_clause";
}
```

Zapisz zapytanie SQL wygenerowane przez ten kod, kiedy Antoni wpisze w polu wyszukiwania tekst „pogromca byków matador". Opisz, jakie problemy mogą się pojawić.

.....

.....

.....

Ćwiczenie: Rozwiązanie

Aby wyszukać poszczególne słowa w bazie aplikacji Ryzykowne prace, trzeba umieścić każde z nich w zapytaniu SELECT języka SQL. Można to zrobić za pomocą słów kluczowych LIKE i OR. Tak powinno wyglądać zapytanie, jeśli Antoni wpisze tekst „matador pogromca byków".

Teraz przeszukujemy opisy stanowisk zamiast nazw, ponieważ w opisie można znaleźć więcej informacji.

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%matador%' OR description LIKE '%pogromca%' OR
  description LIKE '%byków%'
```

Załóżmy, że użyliśmy poniższego kodu PHP do utworzenia tego zapytanie na podstawie danych wpisanych przez użytkownika w formularzu wyszukiwania w aplikacji Ryzykowne prace:

```
$search_query = "SELECT * FROM riskyjobs";
$where_clause = '';
$user_search = $_GET['usersearch'];
$search_words = explode(' ', $user_search);
foreach ($search_words as $word) {
  $where_clause .= " description LIKE '%$word%' OR ";
}
if (!empty($where_clause)) {
  $search_query .= " WHERE $where_clause";
}
```

Każda klauzula LIKE kończy się słowem OR, co pozwala dodać następny warunek. Technika ta działa dobrze do czasu dodania ostatniego słowa.

Przed dołączeniem klauzuli WHERE do zapytania należy sprawdzić, czy nie jest ona pusta.

Ten operator dodaje łańcuch znaków na końcu innego łańcucha.

Zapisz zapytanie SQL wygenerowane przez ten kod, kiedy Antoni wpisze w polu wyszukiwania tekst „pogromca byków matador". Opisz, jakie problemy mogą się pojawić.

*SELECT * FROM riskyjobs*

WHERE description LIKE '%pogromca%' OR description LIKE '%byków%' OR

description LIKE '%matador%' OR

Na końcu zapytania znajduje się dodatkowe słowo OR, co spowoduje, że instrukcja nie zadziała!

Funkcja implode() tworzy łańcuch znaków przy użyciu podłańcuchów

Kod powinien umieszczać w klauzuli WHERE słowo OR między instrukcjami LIKE, **ale nie po ostatniej z nich**. Jak można to zrobić? Może dodamy warunek w pętli, który sprawdzi, czy kod doszedł do ostatniego elementu, co pozwoli pominąć słowo OR? To rozwiązanie zadziała, ale jest nieeleganckie. Dużo bardziej przejrzyste będzie wywołanie polecenia, które odwróci działanie funkcji `explode()`. Funkcja `implode()` przyjmuje tablicę łańcuchów i tworzy przy ich użyciu pojedynczy łańcuch znaków.

To ogranicznik, który funkcja wstawi między podłańcuchy przy ich scalaniu.

```
$where_clause = implode(' OR ', $where_list);
```

Funkcja implode() zwraca pojedynczy łańcuch znaków.

Tu trzeba podać tablicę łańcuchów znaków, które chcemy scalić.

Jak pomoże to w rozwiązaniu problemu nadmiarowego słowa OR w zapytaniu? W funkcji `implode()` można określić ogranicznik, który znajdzie się między scalanymi łańcuchami. Jeśli jako separatora użyjemy sekwencji `' OR '`, utworzymy klauzulę WHERE, w której słowo OR znajdzie się tylko **pomiędzy** klauzulami LIKE.

Zaostrz ołówek

Zmodyfikuj kod PHP, który generuje zapytanie SELECT w aplikacji Ryzykowne prace. Użyj funkcji `implode()`, aby rozwiązać problem nadmiarowego słowa OR.

Zaostrz ołówek: Rozwiązanie

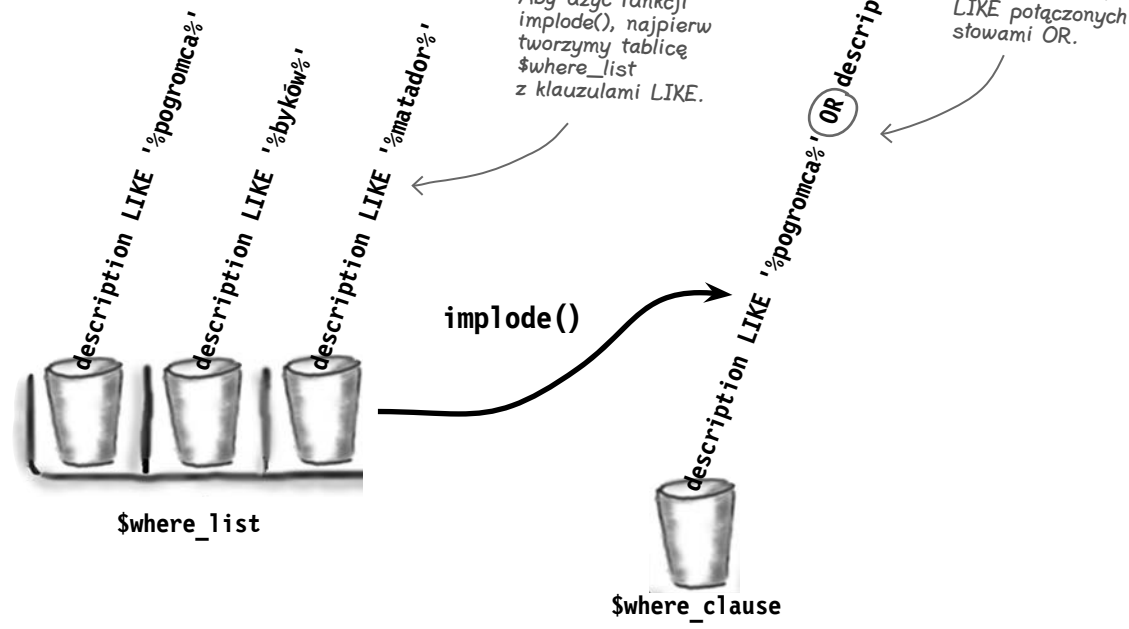
Zmodyfikuj kod PHP, który generuje zapytanie SELECT w aplikacji Ryzykowne prace. Użyj funkcji `implode()`, aby rozwiązać problem nadmiarowego słowa OR.

```
$search_query = "SELECT * FROM riskyjobs";
$where_list = array();
$user_search = $_GET['usersearch'];
$search_words = explode(' ', $user_search);
foreach ($search_words as $word) {
  $where_list[] = "description LIKE '%$word%'";
}
$where_clause = implode(' OR ', $where_list);

if (!empty($where_clause)) {
  $search_query .= " WHERE $where_clause";
}
```

Ponieważ funkcja implode() przyjmuje tablicę scalanych łańcuchów, trzeba utworzyć tablicę klauzul LIKE.

Operator [] w tym kontekście działa tak samo jak funkcja array_push() — dodaje nowy element na końcu tablicy.

Jako ogranicznik należy przekazać do funkcji implode() słowo „OR" z odstępami po obu stronach.



Aby użyć funkcji implode(), najpierw tworzymy tablicę $where_list z klauzulami LIKE.

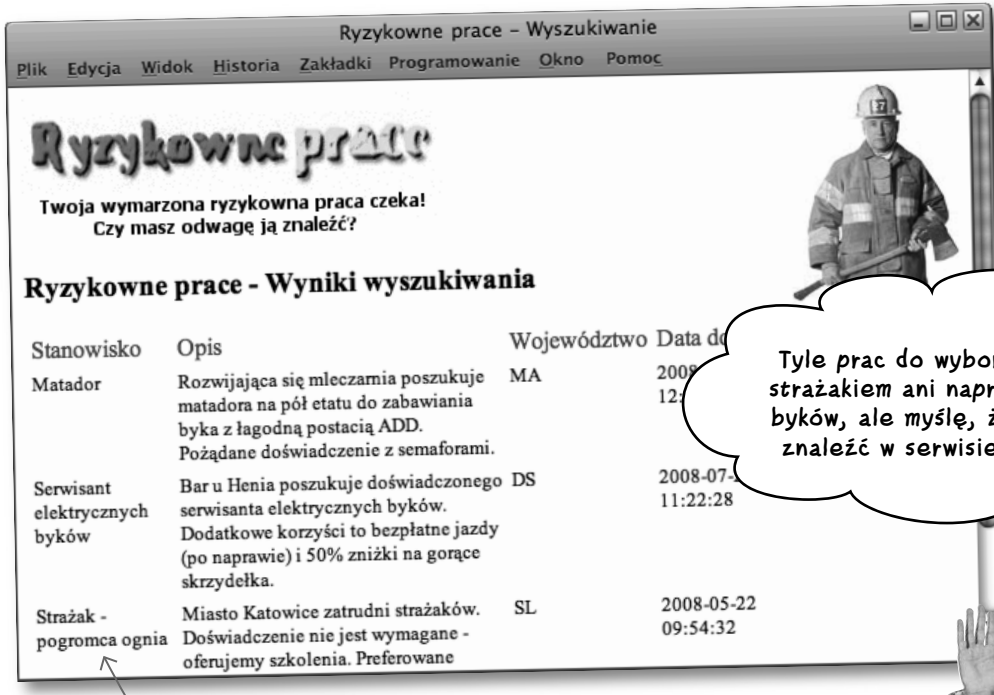
Efekt scalania to kilka klauzul LIKE połączonych słowami OR.

Jazda próbna

Wypróbuj formularz wyszukiwania w aplikacji Ryzykowne prace.

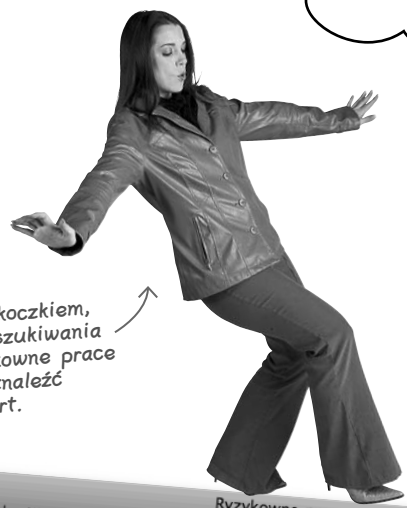
Pobierz kod aplikacji Ryzykowne prace z FTP wydawnictwa Helion (*ftp://ftp.helion.pl/przyklady/hfphms.zip*). Skrypt *search.php* zawiera kod do generowania zapytania, który wcześniej omówiliśmy, i służy do przetwarzania danych wpisanych w formularzu na stronie *search.html*.

Prześlij wspomniany skrypt i inne pliki aplikacji Ryzykowne prace na serwer WWW, a następnie otwórz w przeglądarce formularz wyszukiwania (strona *search.html*). Wpisz kilka różnych wyrażeń, aby zobaczyć, jak sprawuje się kod do generowania zapytań. Koniecznie wpisz zapytanie Antoniego „pogromca byków matador". Jest to dobry test dla nowego kodu, w którym użyliśmy funkcji `implode()`.



Tyle prac do wyboru! Nie chcę zostać strażakiem ani naprawiać elektrycznych byków, ale myślę, że kiedyś uda mi się znaleźć w serwisie wymarzoną pracę.

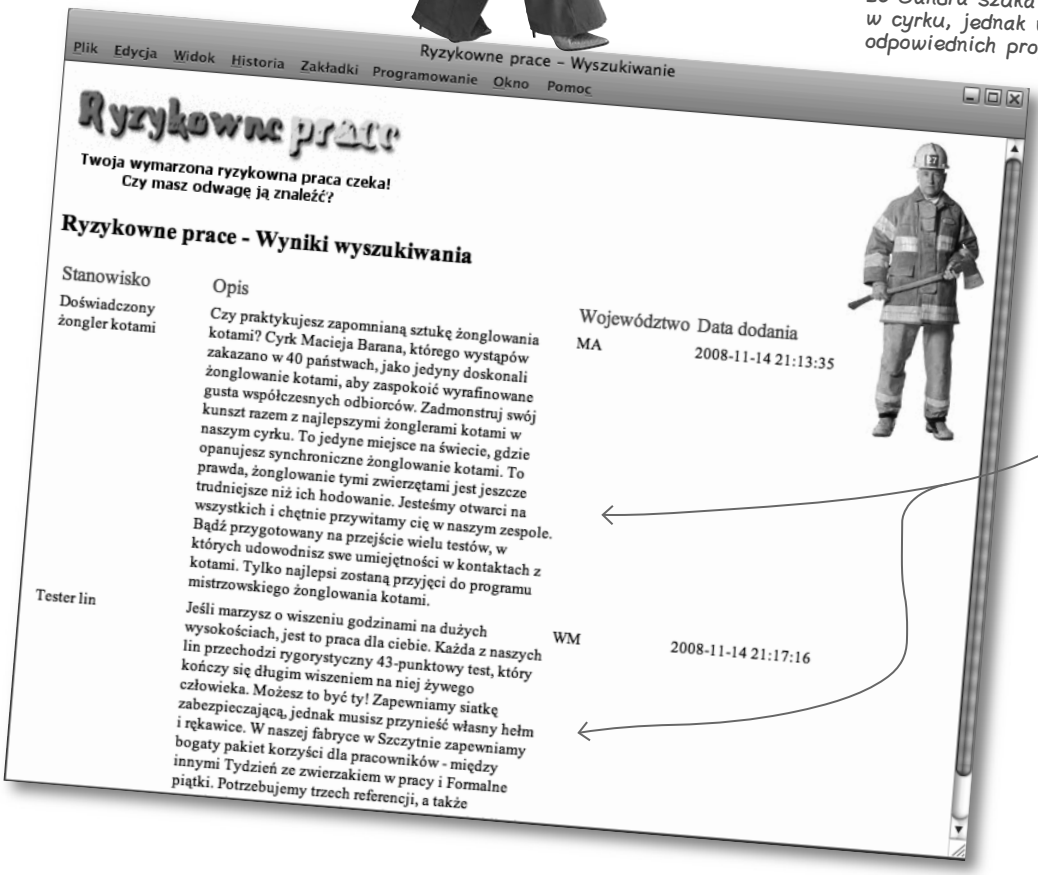
Antoni nie znalazł od razu swej wymarzonej ryzykownej pracy, ale widoczne są wyraźne postępy w działaniu skryptu wyszukiwania, który sprawdza teraz poszczególne słowa.



Jestem linoskoczkiem. Odwiedziłam wasz serwis i nie znalazłam żadnych ofert pracy, choć wpisałam poprawne słowa.

Sandra jest linoskoczkiem, ale formularz wyszukiwania w aplikacji Ryzykowne prace nie pozwolił jej znaleźć odpowiednich ofert.

Szukane słowa wyraźnie określają, że Sandra szuka pracy linoskoczka w cyrku, jednak wyniki nie zawierają odpowiednich propozycji.



Czy problem stanowią szukane słowa, czy może naprawdę nie ma pracy dla linoskoczków?

Zaostrz ołówek

Zapisz zapytanie SQL wygenerowane po wpisaniu przez Sandrę tekstu „chodzenie, lina, cyrk" w polu wyszukiwania. Wyjaśnij, z czego może wynikać problem.

.....

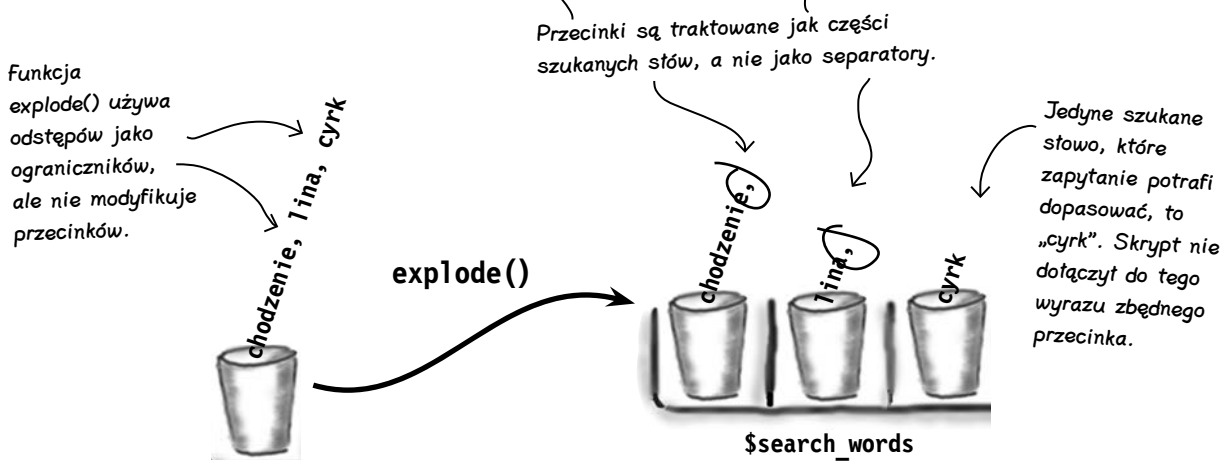
.....

.....

Zaostrz ołówek: Rozwiązanie

Zapisz zapytanie SQL wygenerowane po wpisaniu przez Sandrę tekstu „chodzenie, lina, cyrk" w polu wyszukiwania. Wyjaśnij, z czego może wynikać problem.

```
SELECT * FROM riskyjobs
WHERE description LIKE '%chodzenie,%' OR description LIKE '%lina,%' OR
description LIKE '%cyrk%'
```



Nie widzę, w czym tkwi problem. Wystarczy dwukrotnie wywołać funkcję explode() — najpierw aby usunąć odstępy, a następnie w celu pozbycia się przecinków.

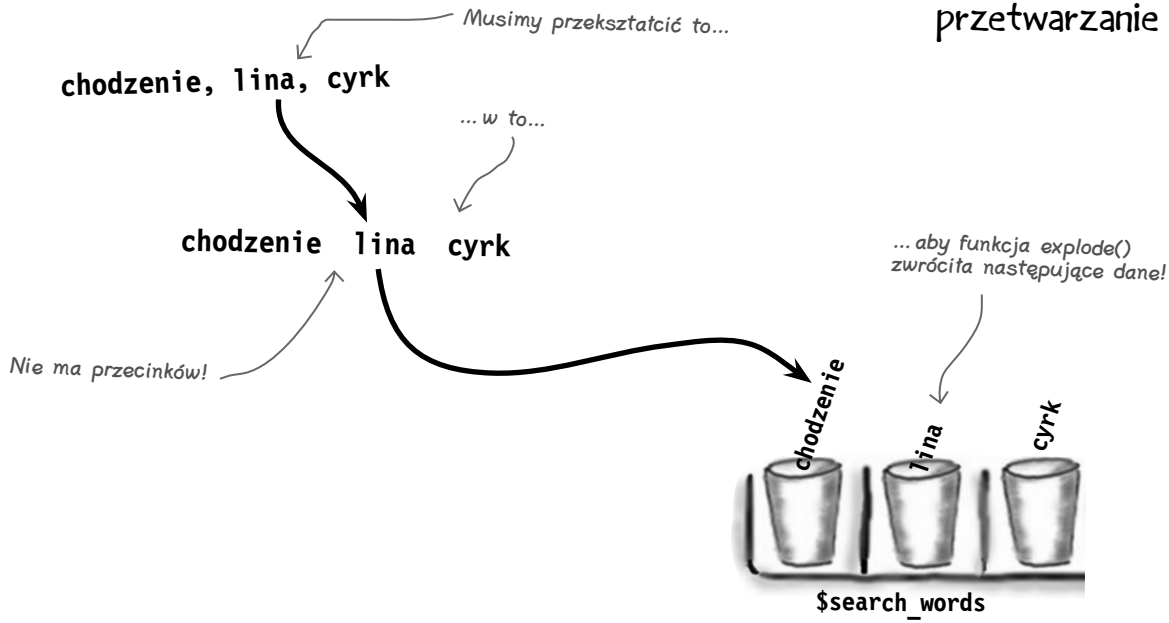
Funkcja `explode()` pozwala podzielić pojedynczy łańcuch na podłańcuchy, jednak tu już wcześniej je utworzyliśmy.

Pierwsze wywołanie funkcji `explode()` tworzy **zbiór łańcuchów znaków** zapisany w tablicy, dlatego nie mamy pojedynczego łańcucha, który można podzielić. Próba rozbicia na fragmenty każdego łańcucha z tablicy prawdopodobnie spowoduje dalsze problemy. Zamiast próbować rozwiązać problem ograniczników za pomocą kilku wywołań funkcji `explode()`, należy **wstępnie przetworzyć** szukany łańcuch znaków, aby przed uruchomieniem tej funkcji zawierał tylko **jeden separator**. Wtedy można ją wywołać, aby zrobiła to, co potrafi najlepiej — podzieliła łańcuch przy użyciu jednego ogranicznika.

Wstępne przetwarzanie szukanego łańcucha znaków

Chcemy przekazać do funkcji `explode()` łańcuch znaków, który można prawidłowo podzielić za jednym razem. Jak to zrobić? Należy się upewnić, że funkcja musi uwzględnić tylko jeden ogranicznik (na przykład odstęp). Oznacza to, że trzeba **wstępnie przetworzyć** szukany łańcuch znaków, aby jego poszczególne słowa były rozdzielone odstępami także wtedy, jeśli użytkownik wpisał przecinki.

Wstępne przetwarzanie pozwala usunąć zbędne znaki i ułatwić przetwarzanie danych.



Nie istnieją głupie pytania

P: Czy przy podziale łańcucha możemy użyć jako ogranicznika więcej niż jednego znaku?

O: Tak, można użyć do tego dowolnej liczby znaków, jednak funkcja nie obsługuje różnych **ograniczników**, dlatego nie rozwiąże to omawianego problemu.

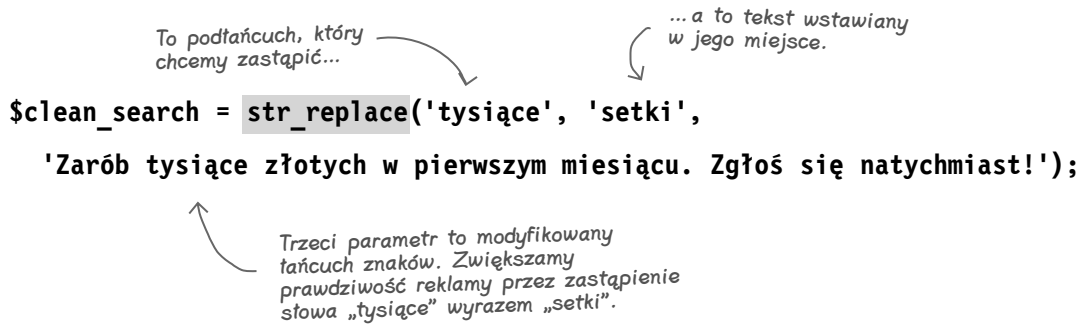
Jeśli wywołamy w celu rozbicia łańcucha instrukcję `explode (', ', $user_search)`, funkcja użyje jako ogranicznika sekwencji przecinek – odstęp, co zadziała, jeśli użytkownik wpisze tekst „chodzenie, lina, cyrk". Jednak funkcja nie podzieli wtedy łańcucha „chodzenie lina cyrk". Wtedy skrypt spróbuje dopasować jeden długi łańcuch, czego chcemy uniknąć.

P: Czy możemy po prostu usunąć przecinki zamiast przekształcać je na odstępy?

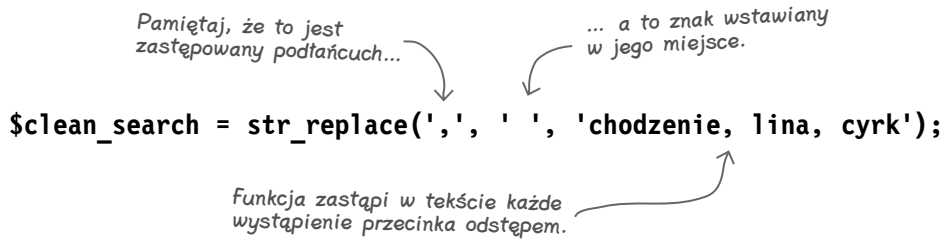
O: To rozwiązanie zadziała, jeśli użytkownik rozdzieli szukane słowa sekwencją przecinek – odstęp, czego jednak nie możemy założyć. Jeśli skrypt będzie usuwał przecinki, może przekształcić tekst „chodzenie,lina" na „chodzenielina", co prawdopodobnie nie pozwoli znaleźć żadnych ofert w bazie serwisu Ryzykowne prace.

Zastępowanie niepotrzebnych znaków w szukanym tekście

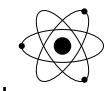
Jeśli się nad tym zastanowić, wstępne przetwarzanie szukanego tekstu w aplikacji Ryzykowne prace przypomina funkcję wyszukiwania i zastępowania w edytorach tekstu. W omawianym skrypcie chcemy znaleźć przecinki i zastąpić je odstępami. Można to zrobić za pomocą funkcji `str_replace()` języka PHP. Należy przekazać do niej trzy parametry: wyszukiwany tekst, zastępujące go znaki i pierwotny łańcuch. Oto przykład zastosowania tej funkcji:



Wróćmy do przecinków w szukanym tekście. Funkcja `str_replace()` pozwala zastępować także pojedyncze znaki:



Po uruchomieniu tego kodu zmienna `$clean_string` będzie zawierać łańcuch „chodzenie lina cyrk".



WYSIL SZARE KOMÓRKI

Czy widzisz coś podejrzanego w efekcie działania funkcji `str_replace()`? Czy uważasz, że zastąpienie przecinków odstępami rozwiąże problem?

Ćwiczenie

Na podstawie poniższego kodu PHP określ, co znajdzie się w tablicy $search_words po przetworzeniu poszczególnych szukanych łańcuchów znaków. Zapisz dane w odpowiednich elementach tablicy i wykreśl pozycje, które nie będą potrzebne.

```
$clean_search = str_replace(',', ' ', $user_search);
$search_words = explode(' ', $clean_search);
```

byk,matador płachta

$search_words

Trzy odstępy!

byk matador płachta

$search_words

byk , matador płachta

$search_words

Dwa odstępy!

byk,matador, płachta

$search_words

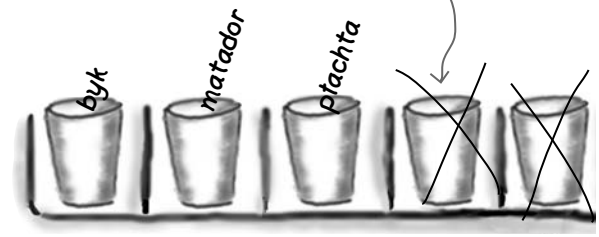
Ćwiczenie: Rozwiązanie

Na podstawie poniższego kodu PHP określ, co znajdzie się w tablicy $search_words po przetworzeniu poszczególnych szukanych łańcuchów znaków. Zapisz dane w odpowiednich elementach tablicy i wykreśl pozycje, które nie będą potrzebne.

```
$clean_search = str_replace(',', ' ', $user_search);
$search_words = explode(' ', $clean_search);
```

W tej tablicy znajdą się tylko trzy elementy.

byk,matador płachta

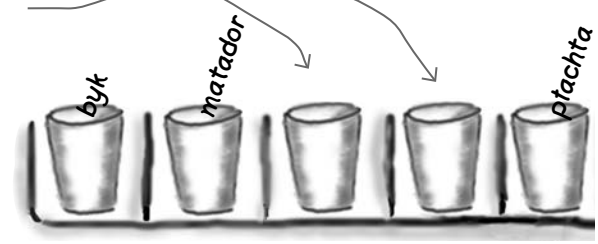


$search_words

Te dwa elementy są puste z uwagi na dwa dodatkowe odstępy między słowami „matador" i „płachta" w szukanym tekście.

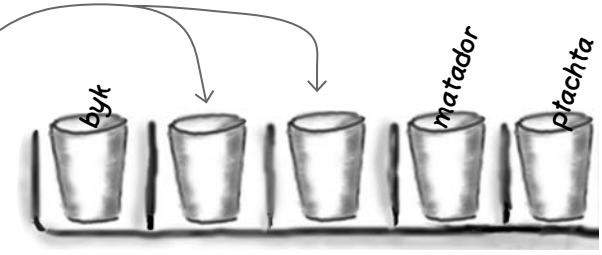
Trzy odstępy!

byk matador płachta



$search_words

byk , matador płachta

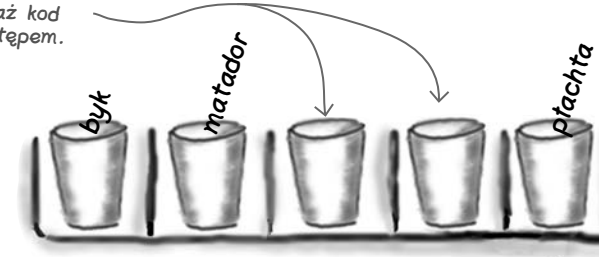


$search_words

Także tu występują dwa puste elementy, ponieważ kod zastępuje przecinek odstępem.

Dwa odstępy!

byk,matador, płachta



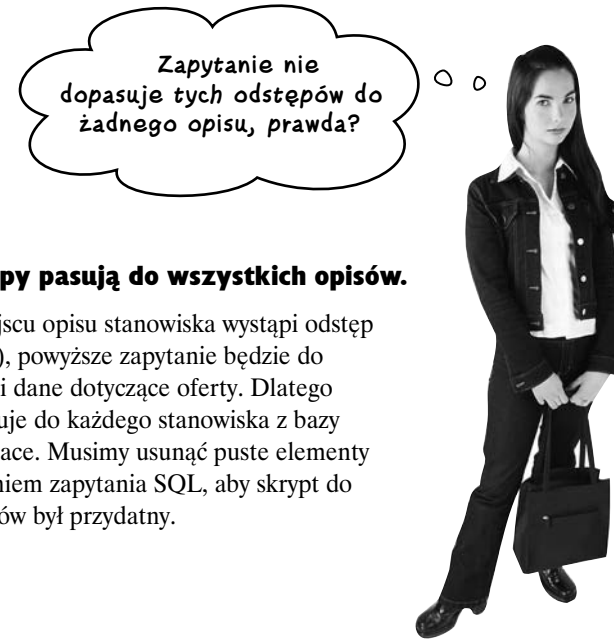
$search_words

Nie do końca. Wstępne przetwarzanie usuwa zbędne znaki, ale niestety nie prowadzi do utworzenia tablicy, w której wszystkie szukane słowa są poprawne.

Pamiętaj, że nasz cel to utworzenie łańcucha znaków, w którym wszystkie słowa są rozdzielone tym samym ogranicznikiem — odstępem. Zobaczmy, co stało się w trzech ostatnich przykładach z poprzedniej strony. Niektóre elementy tablicy `$search_words` są puste. Jeśli spróbujemy utworzyć klauzulę `WHERE` przy użyciu pustych elementów, może powstać zapytanie podobne do poniższego:

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%byk%' OR
  description LIKE '%matador%' OR
  description LIKE '% %' OR
  description LIKE '% %' OR
  description LIKE '%płachta%'
```

Te odstępy pasują do wszystkich spacji w każdym opisie stanowiska. To poważny problem.



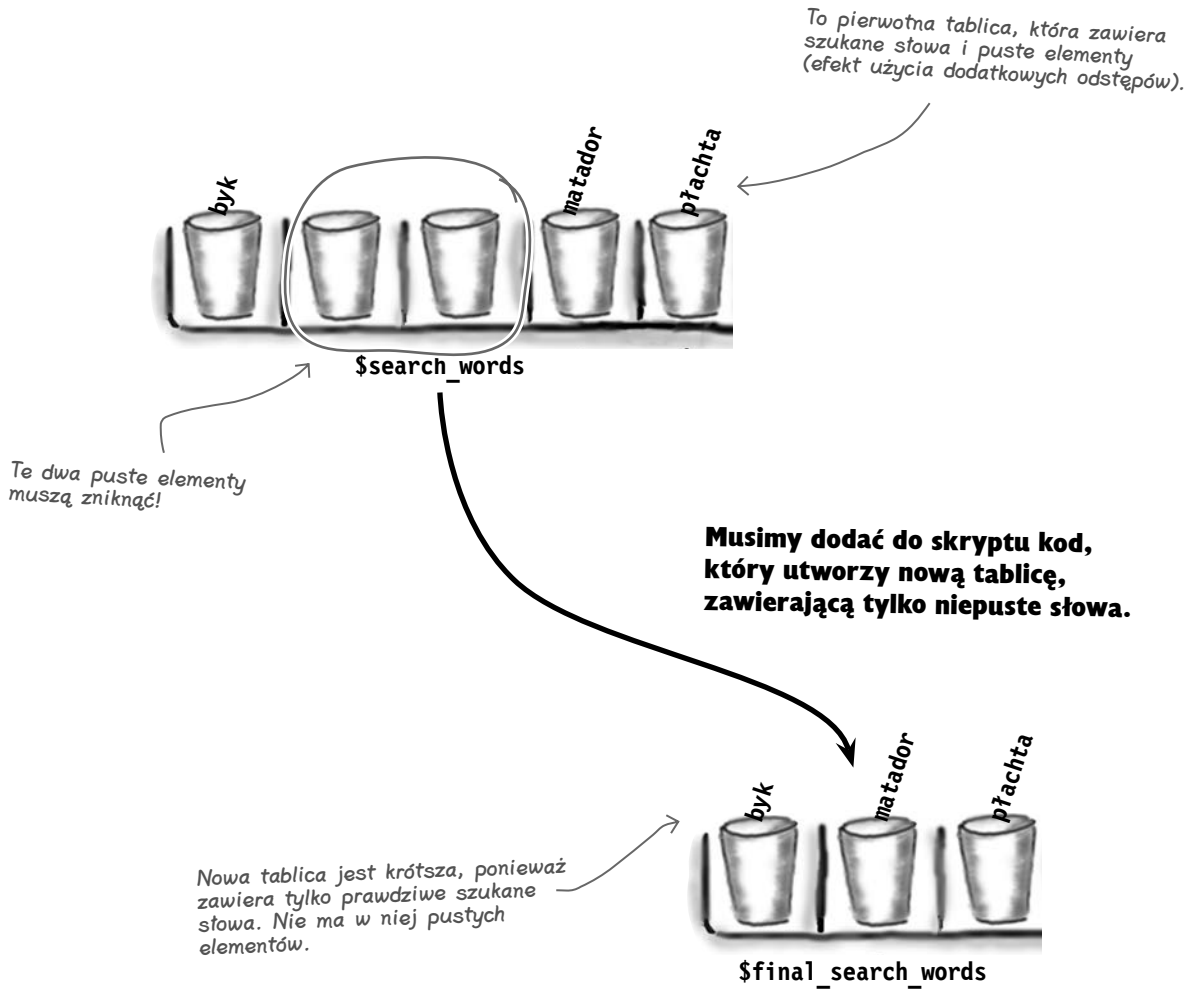
Nieprawda! Odstępy pasują do wszystkich opisów.

Jeśli w dowolnym miejscu opisu stanowiska wystąpi odstęp (co jest niemal pewne), powyższe zapytanie będzie do niego pasować i zwróci dane dotyczące oferty. Dlatego podana instrukcja pasuje do każdego stanowiska z bazy serwisu Ryzykowne prace. Musimy usunąć puste elementy tablicy przed utworzeniem zapytania SQL, aby skrypt do wyszukiwania ofert znów był przydatny.

Potrzebujemy poprawnych szukanych łańcuchów

Dobra wiadomość jest taka, że można łatwo przefiltrować szukane słowa przed użyciem ich w zapytaniu. Trzeba utworzyć nową tablicę, która będzie zawierać tylko potrzebne łańcuchy. Dlatego należy skopiować wszystkie niepuste elementy z pierwszej tablicy do drugiej, a następnie użyć nowej struktury do wygenerowania zapytania `SELECT`.

Aby utworzyć nową tablicę, możemy użyć pętli `foreach` i przejść po wszystkich elementach pierwotnej tablicy, wyszukując przy tym niepuste słowa za pomocą instrukcji `if`. Jeśli kod znajdzie poprawny łańcuch, powinien dodać go do nowej tablicy. Proces ten wygląda następująco:



Kopiowanie niepustych elementów do nowej tablicy

Przyjrzyjmy się teraz fragmentowi kodu, który kopiuje niepuste elementy z tablicy `$search_words` do nowej tablicy `$final_search_words`.

```
$search_query = "SELECT * FROM riskyjobs";

// Pobieranie szukanych słów do tablicy.
$clean_search = str_replace(',', ' ', $user_search);
$search_words = explode(' ', $clean_search);
$final_search_words = array();
if (count($search_words) > 0) {
  foreach ($search_words as $word) {
    if (!empty($word)) {
      $final_search_words[] = $word;
    }
  }
}
```

To nic nowego — zastępujemy przecinki odstępami za pomocą funkcji str_replace().

Przejście w pętli po wszystkich elementach tablicy $search_words. Jeśli element nie jest pusty, należy zapisać go w tablicy $final_search_words.

Po sprawdzeniu, czy w tablicy `$search_words` znajduje się choć jedno słowo, kod przechodzi po niej w pętli `foreach` i wyszukuje niepuste elementy. Jeśli je znajdzie, dodaje je za pomocą operatora `[]` na końcu tablicy `$final_search_words`. W ten sposób powstaje nowa tablica.

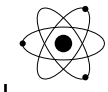
Co dzieje się potem? Kod generuje zapytanie `SELECT` w prawie taki sam sposób jak wcześniej. Jedyna różnica polega na użyciu tablicy `$final_search_words` zamiast `$search_words`:

```
// Generowanie klauzuli WHERE przy użyciu wszystkich szukanych słów.
$where_list = array();
if (count($final_search_words) > 0) {
  foreach($final_search_words as $word) {
    $where_list[] = "description LIKE '%$word%'";
  }
}
$where_clause = implode(' OR ', $where_list);

// Dodawanie do zapytania klauzuli ze słowem kluczowym WHERE.
if (!empty($where_clause)) {
  $search_query .= " WHERE $where_clause";
}
```

To ten sam kod, którego użyliśmy do utworzenia klauzuli WHERE we wcześniejszym zapytaniu, jednak tym razem zastosowaliśmy tablicę $final_search_words, pozbawioną pustych elementów.

Ten kod tworzy zapytanie, które nie obejmuje pustych elementów. Oto nowa instrukcja utworzona na podstawie szukanego tekstu „byk, matador, płachta”:

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%byk%' OR
  description LIKE '%matador%' OR
  description LIKE '%płachta%'
```



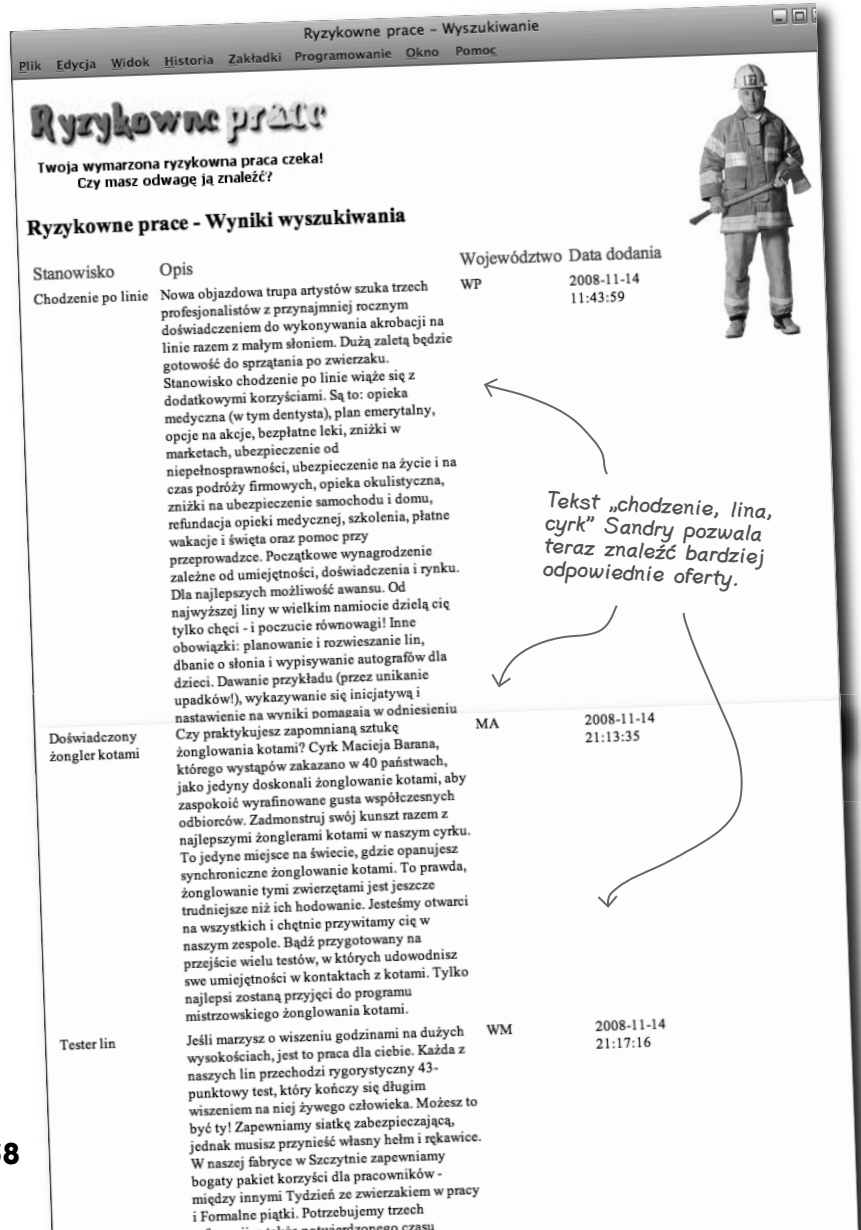
WYSIL SZARE KOMÓRKI

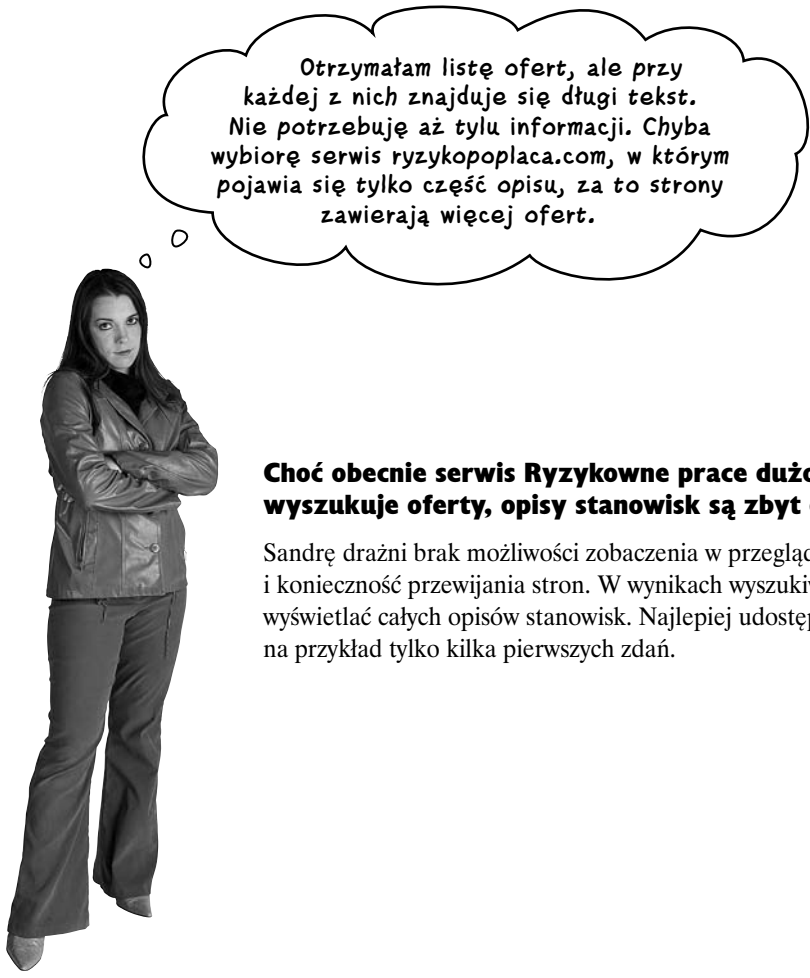
Czy to zapytanie zwróci użytkownikom oferty, których szukają?

Jazda próbna

Zmodyfikuj skrypt do wyszukiwania ofert, aby wstępnie przetwarzał szukany łańcuch znaków.

Zmodyfikuj skrypt *search.php*. Użyj w nim funkcji `explode()` i `implode()`, by wstępnie przetworzyć szukany tekst i wygenerować bardziej niezawodne zapytanie `SELECT`. Następnie prześlij plik na serwer WWW i uruchom kilkakrotnie wyszukiwanie.





Choć obecnie serwis Ryzykowne prace dużo lepiej wyszukuje oferty, opisy stanowisk są zbyt długie.

Sandrę drażni brak możliwości zobaczenia w przeglądarce wielu ofert i konieczność przewijania stron. W wynikach wyszukiwania nie trzeba wyświetlać całych opisów stanowisk. Najlepiej udostępnić ich fragmenty, na przykład tylko kilka pierwszych zdań.

Napisz, jak możemy skrócić opisy stanowisk, aby zajmowały mniej miejsca w wynikach wyszukiwania:

.....

.....

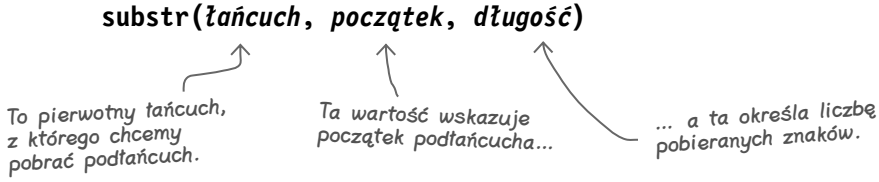
.....

Czasem potrzebny jest tylko fragment łańcucha

Ponieważ opisy stanowisk w bazie serwisu Ryzykowne prace mają różną długość i niektóre zajmują dużo miejsca, można uporządkować wyniki wyszukiwania przez skrócenie pełnego tekstu. Po każdym obciętym opisie warto dodać trzy kropki (…), aby użytkownicy wiedzieli, że nie jest to cały tekst.

Funkcja substr() języka PHP umożliwia pobranie fragmentu łańcucha znaków.

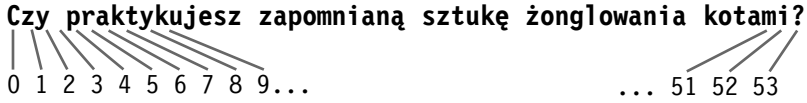
Funkcja `substr()` języka PHP doskonale nadaje się do pobierania fragmentów łańcuchów znaków. Należy przekazać do tej funkcji pierwotny tekst i dwie liczby całkowite. Pierwsza z nich to indeks początku pobieranego podłańcucha, a druga — liczba jego znaków. Składnia tej funkcji wygląda następująco:



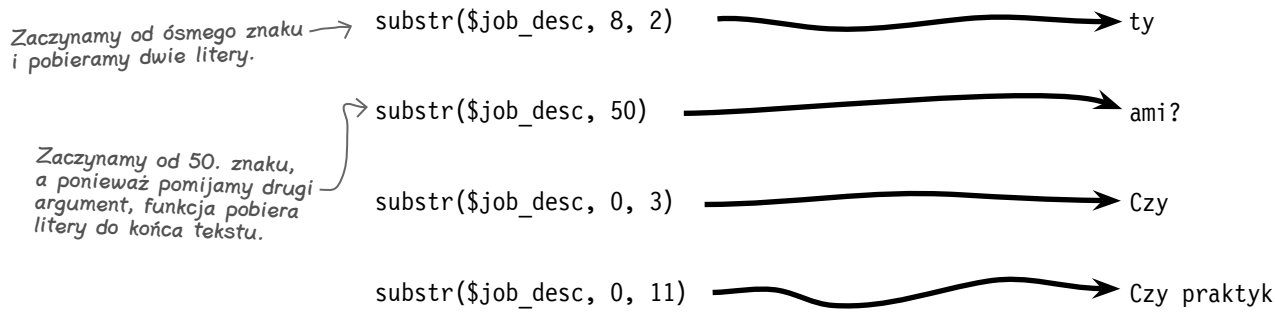
Przy korzystaniu z funkcji `substr()` można traktować łańcuch jak tablicę, w której każdy znak to odrębny element. Przyjrzyj się poniższemu łańcuchowi:

```
$job_desc = 'Czy praktykujesz zapomnianą sztukę żonglowania kotami?';
```

Podobnie jak elementy w tablicy, tak i każdy znak tego łańcucha ma indeks (od 0 do liczby znaków w tekście).

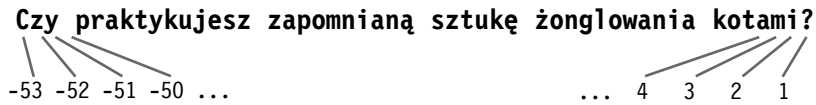


Możemy użyć indeksów w funkcji substr(), aby pobrać fragmenty łańcucha:

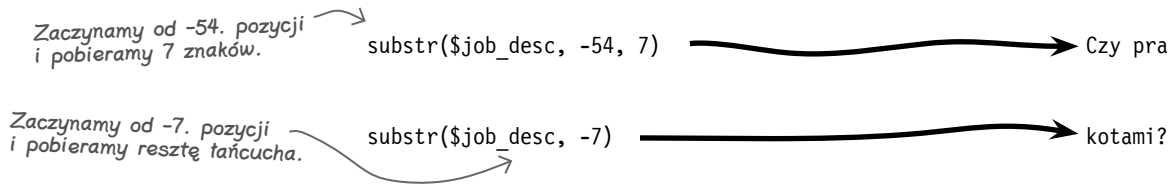


Pobieranie podłańcuchów z obu stron tekstu

Funkcja `substr()` pobiera podłańcuchy nie tylko na podstawie indeksów liczonych od początku tekstu. Możesz też wskazać litery, licząc od końca łańcucha. Pobieranie także wtedy działa od lewej do prawej, jednak do określenia początku podłańcucha trzeba użyć liczby ujemnej.



Oto kilka przykładów:



Zaostrz ołówek

Poniżej znajduje się kod PHP, który generuje tabelę HTML z wynikami wyszukiwania ofert w serwisie Ryzykowne prace. Dokończ brakujący kod, którego zadaniem jest skrócenie opisu stanowiska do 100 znaków, a daty dodania oferty do roku, miesiąca i dnia.

```
echo '<table border="0" cellpadding="2">';
echo '<td>Stanowisko</td><td>Opis</td><td>Województwo</td><td>Data dodania</td>';
while ($row = mysqli_fetch_array($result)) {
  echo '<tr class="results">';
  echo '<td valign="top" width="20%">' . $row['title'] . '</td>';
  echo '<td valign="top" width="50%">' . ............................ . '...</td>';
  echo '<td valign="top" width="10%">' . $row['state'] . '</td>';
  echo '<td valign="top" width="20%">' . ............................ . '</td>';
  echo '</tr>';
}
echo '</table>';
```

Zaostrz ołówek: Rozwiązanie

Poniżej znajduje się kod PHP, który generuje tabelę HTML z wynikami wyszukiwania ofert w serwisie Ryzykowne prace. Dokończ brakujący kod, którego zadaniem jest skrócenie opisu stanowiska do 100 znaków, a daty dodania oferty do roku, miesiąca i dnia.

```
echo '<table border="0" cellpadding="2">';
echo '<td>Stanowisko</td><td>Opis</td><td>Województwo</td><td>Data dodania</td>';
while ($row = mysqli_fetch_array($result)) {
  echo '<tr class="results">';
  echo '<td valign="top" width="20%">' . $row['title'] . '</td>';
  echo '<td valign="top" width="50%">' . substr($row['description'], 0, 100) . '...</td>';
  echo '<td valign="top" width="10%">' . $row['state'] . '</td>';
  echo '<td valign="top" width="20%">' . substr($row['date_posted'], 0, 10) . '</td>';
  echo '</tr>';
}
echo '</table>';
```

Dodajemy wielokropek na końcu tekstu, aby podkreślić, że jest to tylko część opisu.

Dane w polu date_posted rozpoczynają się od sekwencji RRRR-MM-DD, która zajmuje dokładnie 10 znaków.

Dla zaawansowanych

Można zrezygnować z funkcji `substr()` języka PHP i skrócić opis stanowiska w zapytaniu SQL. Służy do tego funkcja `SUBSTRING()` języka MySQL, która przyjmuje te same argumenty, co polecenie `substr()`. Jedyna różnica polega na tym, że w języku MySQL indeks początkowy to 1, a nie 0. Dlatego aby pobrać 100 pierwszych znaków opisu stanowiska, należy użyć następującego kodu:

```
SELECT SUBSTRING(job_description, 1, 100)
  FROM riskyjobs;
```

Zastosowanie funkcji języka PHP ma tę zaletę, że daje dostęp zarówno do skróconego, jak i pełnego opisu. Jeśli użyjemy funkcji języka MySQL, uzyskamy tylko niepełny tekst i będziemy musieli pobrać cały opis za pomocą następnego zapytania.

Nie istnieją głupie pytania

P: Czy funkcja `substr()` działa też dla liczb?

O: Nie, polecenie to obsługuje tylko łańcuchy. Jednak jeśli przechowujesz liczby w kolumnach typu `CHAR`, `VARCHAR` lub `TEXT`, po pobraniu ich za pomocą języka SQL będą traktowane w kodzie PHP jak tekst, dlatego można użyć do nich funkcji `substr()`.

P: Co się stanie, jeśli podana liczba znaków jest większa niż długość łańcucha? Czy funkcja zwróci tekst z odstępami, aby dopasować jego długość do żądania?

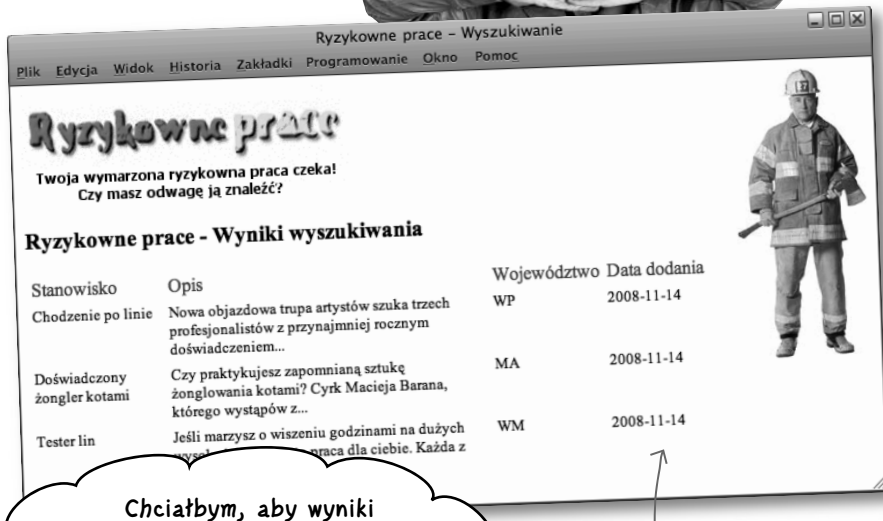
O: Funkcja zwróci cały łańcuch, jednak nie uzupełni go odstępami, aby zmienić długość tekstu. Na przykład kod `substr('pies', 0, 10)` zwróci łańcuch „pies".

Jazda próbna

Popraw skrypt do wyszukiwania ofert, aby skrócić wyświetlane opisy i daty.

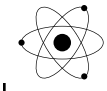
Zmodyfikuj skrypt *search.php*. Użyj w nim funkcji `substr()` języka PHP, aby skrócić opisy i daty wyświetlane w wynikach wyszukiwania. Prześlij skrypt na serwer WWW i uruchom kilkakrotnie wyszukiwanie.

Sandra jest zadowolona, ponieważ może zobaczyć wyniki wyszukiwania bez konieczności przewijania długich opisów stanowisk.



Czas dodania oferty jest bardziej czytelny, ponieważ nie obejmuje godziny, a jedynie datę.

Chciałbym, aby wyniki były posortowane według województwa lub daty dodania oferty. Interesuje mnie praca matadora w Małopolsce.

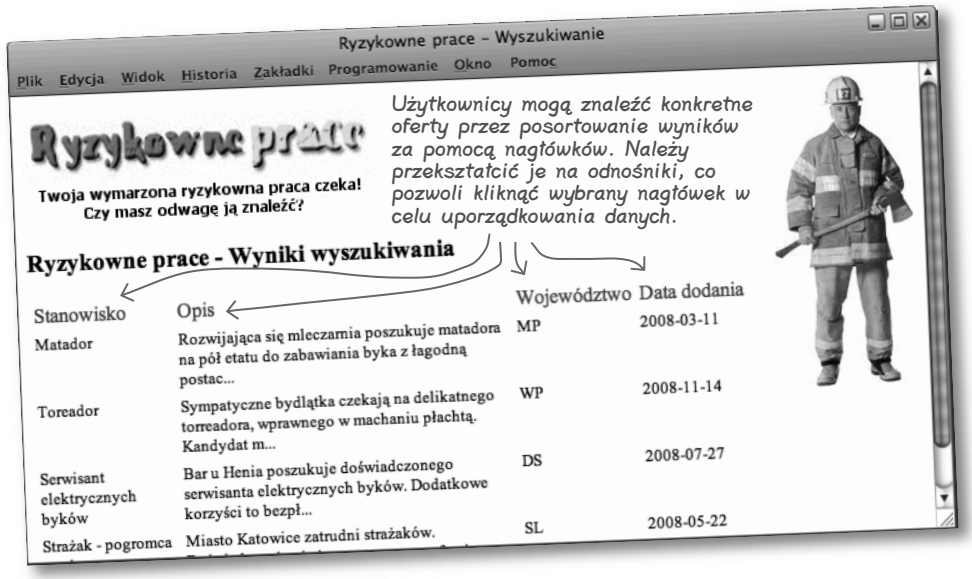


WYSIL SZARE KOMÓRKI

Jak należy zmienić układ strony i zapytanie, aby umożliwić sortowanie danych według województwa, stanowiska lub daty dodania oferty?

Można posortować wyniki w kilku zapytaniach

Aby umożliwić użytkownikom posortowanie wyników wyszukiwania, potrzebny jest mechanizm do określania pożądanego uporządkowania danych. Może użyjemy do tego formularza lub przycisków? Nie — istnieje prostszy sposób. Wystarczy użyć kodu HTML i przekształcić nagłówki kolumn w tabeli wyników w odnośniki. Użytkownicy będą mogli kliknąć te odsyłacze, aby wskazać, której kolumny chcą użyć do posortowania ofert.



Tych odnośników możemy użyć do odświeżenia tego samego skryptu i uruchomienia w nim zapytania, które posortuje wyniki na podstawie wybranego odsyłacza. Wiesz już, jak użyć klauzuli `ORDER BY` do uporządkowania danych zwracanych przez zapytanie. Jeśli utworzymy różne zapytania SQL, które sortują wyniki według poszczególnych kolumn, umożliwimy użytkownikom uporządkowanie danych alfabetycznie (według stanowisk, opisów i województw) lub chronologicznie (według dat zamieszczenia ofert).

Poniższe zapytanie SQL sortuje wyniki alfabetycznie według opisu stanowiska:

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%pogromca%' OR description LIKE '%byków%' OR
  description LIKE '%matador%'
  ORDER BY description
```

Ta klauzula sortuje wyniki zapytania w rosnącym porządku alfabetycznym według opisów stanowisk.

Zaostrz ołówek

Napisz trzy różne zapytania, które posortują oferty z serwisu Ryzykowne prace według nazwy stanowiska, województwa i daty zamieszczenia oferty. Przyjmij, że użytkownik wpisał tekst „mycie, okna, wieżowce".

Jak należy zmodyfikować te zapytania, aby wyświetlić wyniki malejąco według nazw stanowisk i województw? Jak pokazać oferty uporządkowane od najnowszej z nich?

Zaostrz ołówek: Rozwiązanie

Napisz trzy różne zapytania, które posortują oferty z serwisu Ryzykowne prace według nazwy stanowiska, województwa i daty zamieszczenia oferty. Przyjmij, że użytkownik wpisał tekst „mycie, okna, wieżowce".

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%mycie%' OR description LIKE '%okna%' OR
  description LIKE '%wieżowce%'
  ORDER BY title
```

Domyślnie klauzula ORDER BY sortuje dane rosnąco, co odpowiada instrukcji ORDER BY job_title ASC.

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%mycie%' OR description LIKE '%okna%' OR
  description LIKE '%wieżowce%'
  ORDER BY state
```

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%mycie%' OR description LIKE '%okna%' OR
  description LIKE '%wieżowce%'
  ORDER BY date_posted
```

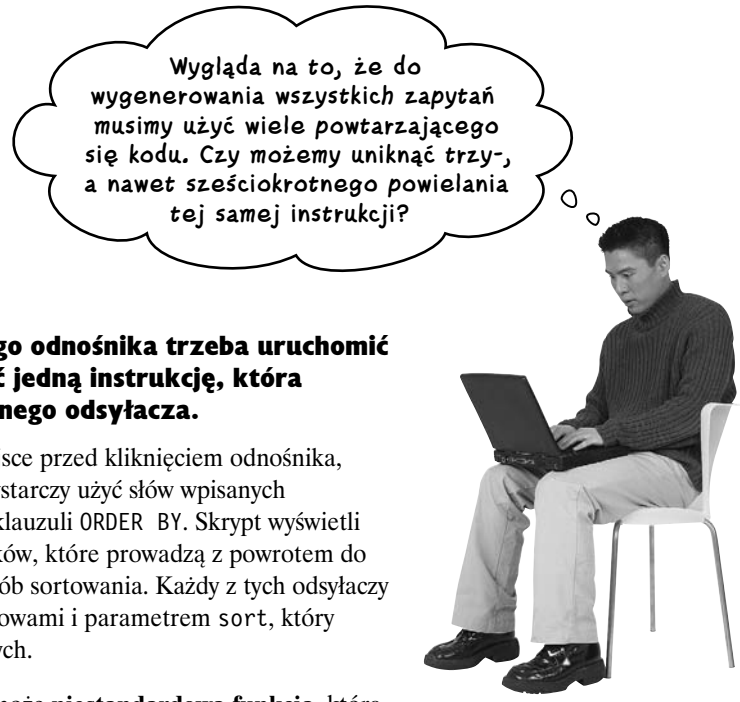
Jak należy zmodyfikować te zapytania, aby wyświetlić wyniki malejąco według nazw stanowisk i województw? Jak pokazać oferty uporządkowane od najnowszej z nich?

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%mycie%' OR description LIKE '%okna%' OR
  description LIKE '%wieżowiec%'
  ORDER BY title DESC
```

Można użyć tych zapytań, jeśli użytkownik uporządkował dane według jednej z kolumn i ponownie kliknął jej nagłówek, aby odwrócić kolejność ofert.

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%mycie%' OR description LIKE '%okna%' OR
  description LIKE '%wieżowiec%'
  ORDER BY state DESC
```

```
SELECT * FROM riskyjobs
  WHERE description LIKE '%mycie%' OR description LIKE '%okna%' OR
  description LIKE '%wieżowiec%'
  ORDER BY date_posted DESC
```



Jest to możliwe. Choć dla każdego odnośnika trzeba uruchomić inne zapytanie, można utworzyć jedną instrukcję, która dostosowuje działanie do wybranego odsyłacza.

Pierwsze wyświetlenie wyników ma miejsce przed kliknięciem odnośnika, dlatego nie trzeba sortować danych. Wystarczy użyć słów wpisanych w formularzu i utworzyć zapytanie bez klauzuli `ORDER BY`. Skrypt wyświetli wyniki z nagłówkami w formie odnośników, które prowadzą z powrotem do skryptu, ale pozwalają wybrać inny sposób sortowania. Każdy z tych odsyłaczy składa się z adresu URL z szukanymi słowami i parametrem `sort`, który określa pożądane uporządkowanie danych.

W zastosowaniu tej techniki bardzo pomoże **niestandardowa funkcja**, która będzie przyjmować informacje o sposobie sortowania ofert, a następnie zwróci łańcuch znaków z odpowiednimi klauzulami `WHERE` i `ORDER BY`. Nowa funkcja sprawdzi parametr `sort`, aby określić, jak należy uporządkować wyniki wyszukiwania. Funkcja ta powinna wykonywać następujące zadania:

1. Wstępne przetwarzanie szukanych słów i zapisywanie ich w tablicy.
2. Opcjonalne przyjmowanie parametru `sort`, informującego funkcję, która kolumna posłuży do sortowania.
3. Usuwanie pustych słów.
4. Tworzenie klauzuli `WHERE` z wszystkimi szukanymi słowami.
5. Sprawdzanie, czy parametr `sort` ma wartość. Jeśli tak, należy dodać ją do klauzuli `ORDER BY`.
6. Zwracanie utworzonego zapytania.

Może się wydawać, że przygotowanie tej funkcji będzie wymagać wiele pracy, jednak większość kodu jest już gotowa. Musimy tylko przekształcić go w funkcję. Jednak wcześniej zobaczmy, jak powstają funkcje niestandardowe…

Funkcje umożliwiają wielokrotne wykorzystanie kodu

Funkcja to odrębny blok kodu, który można uruchomić w dowolnym miejscu skryptu. Do tej pory używałeś **funkcji wbudowanych**, które są częścią języka PHP. Polecenia `explode()`, `substr()` i `mysqli_query()` to zdefiniowane w języku PHP funkcje, które można wywoływać w każdym skrypcie.

Możesz też tworzyć własne **funkcje niestandardowe**, aby uruchamiać operacje, których język nie udostępnia. Dzięki utworzeniu funkcji niestandardowej można wielokrotnie używać własnego kodu bez powielania go w skrypcie. Aby uruchomić ten kod, wystarczy wywołać funkcję przez podanie jej nazwy.

Poniższy kod to przykładowa funkcja niestandardowa `replace_commas()`, która zastępuje przecinki odstępami w łańcuchu znaków:

Funkcje niestandardowe pozwalają nadać nazwę fragmentowi kodu PHP, co ułatwia jego wielokrotne użycie.

Aby utworzyć funkcję niestandardową, należy zacząć od słowa „function".

Tu możesz wpisać dowolną nazwę funkcji. Powinna być ona jak najbardziej opisowa.

Po nazwie funkcji znajduje się para nawiasów. Do funkcji można przekazywać wiele wartości (argumentów) rozdzielonych przecinkami. Tu używamy tylko jednej takiej wartości.

```
function replace_commas($str) {
  $new_str = str_replace(',', ' ', $str);
  return $new_str;
}
```

Nawiasy klamrowe — podobnie jak w pętlach i instrukcjach if — określają miejsce, w którym należy podać kod.

Funkcja może zwracać wartość w miejscu jej wywołania. Ta funkcja zwraca zmodyfikowany łańcuch znaków.

Kiedy niestandardowa funkcja jest już gotowa, wystarczy wywołać ją za pomocą nazwy i wpisać w nawiasach oczekiwane wartości. Jeśli funkcja zwraca wartość, można przypisać ją do nowej zmiennej:

Przekazujemy do funkcji łańcuch „chodzenie, lina, cyrk".

```
$clean_search = replace_commas('chodzenie, lina, cyrk');
```

Funkcja zwraca nowy łańcuch, w którym przecinki zastąpiono odstępami.

Zbuduj zapytanie za pomocą niestandardowej funkcji

Napisaliśmy już większą część kodu potrzebnego do utworzenia niestandardowej funkcji, która wygeneruje zapytanie do przeszukiwania bazy serwisu Ryzykowne prace. Teraz wystarczy przenieść ten kod do szkieletu funkcji PHP. Oto niestandardowa funkcja `build_query()`:

```
function build_query($user_search) {
  $search_query = "SELECT * FROM riskyjobs";

  // Zapisywanie szukanych słów w tablicy.
  $clean_search = str_replace(',', ' ', $user_search);
  $search_words = explode(' ', $clean_search);
  $final_search_words = array();
  if (count($search_words) > 0) {
    foreach ($search_words as $word) {
      if (!empty($word)) {
        $final_search_words[] = $word;
      }
    }
  }

  // Generowanie klauzuli WHERE przy użyciu wszystkich szukanych słów.
  $where_list = array();
  if (count($final_search_words) > 0) {
    foreach($final_search_words as $word) {
      $where_list[] = "description LIKE '%$word%'";
    }
  }
  $where_clause = implode(' OR ', $where_list);

  // Dodawanie do zapytania klauzuli WHERE z szukanymi słowami.
  if (!empty($where_clause)) {
    $search_query .= " WHERE $where_clause";
  }

  return $search_query;
}
```

Przekazujemy do funkcji tablicę $user_search, którą utworzyliśmy na podstawie danych wpisanych w formularzu wyszukiwania.

W funkcji nie ma prawie nic nowego!

Dodaliśmy tę instrukcję. W tym miejscu zwracamy nowe zapytanie, aby kod, który wywołał funkcję, mógł go użyć.

Funkcja `build_query()` zwraca kompletne zapytanie SQL, oparte na szukanym tekście, który przekazaliśmy za pomocą argumentu `$user_search`. Aby użyć funkcji, wystarczy przekazać szukane dane, które wprowadził użytkownik, a następnie zapisać zwrócony wynik w nowym łańcuchu znaków — `$search_query`:

```
$search_query = build_query($user_search);
```

Ta zmienna pozwala zapisać zwróconą przez funkcję wartość, którą tu jest nowe zapytanie.

To wartość z formularza wyszukiwania przesłanego przez użytkownika.

Wszystko o niestandardowej funkcji

Rozmowa tygodnia:
Czy niestandardowe funkcje naprawdę są niestandardowe?

Head First: Wiele osób zastanawia się, co złego jest w powtarzającym się kodzie. W końcu wystarczy go napisać, a następnie skopiować, wkleić i gotowe.

Niestandardowa funkcja: Nie chcę już słyszeć o powtórzeniach w kodzie. Są brzydkie i zmniejszają czytelność skryptów. Już samo to wystarczy, aby ich unikać, a istnieją też przecież o wiele ważniejsze przyczyny.

Head First: Na przykład?

Niestandardowa funkcja: Co się stanie, jeśli kod będzie wymagał zmian? Dzieje się to dość często.

Head First: I co z tego? Aplikacja zmienia się cały czas. Trzeba po prostu wprowadzić poprawki.

Niestandardowa funkcja: A jeśli zmiany dotyczą kodu, który powtarza się w wielu miejscach aplikacji?

Head First: Nie widzę problemu. Wystarczy znaleźć wszystkie te miejsca i zmodyfikować kod.

Niestandardowa funkcja: Dobrze, a jeżeli programista zapomni wprowadzić poprawkę w jednym z tych miejsc? W końcu jest tylko człowiekiem. Jeśli pominie jedną zmianę, będzie mu bardzo trudno wykryć błąd.

Head First: Rzeczywiście, jest to prawdopodobne. W czym jednak ty możesz pomóc?

Niestandardowa funkcja: W tym właśnie tkwi moja wartość. Jeśli kod znajduje się w funkcji, wystarczy zmodyfikować go jednokrotnie. Tylko jedna zmiana i gotowe!

Head First: Muszę przyznać, że to atrakcyjna wizja. Jednak wciąż nie rozumiem, dlaczego warto cię stosować. W końcu masz poważne ograniczenia, prawda? Możesz używać tylko łańcuchów znaków.

Niestandardowa funkcja: To nieprawda! Mogę przyjmować dane dowolnego typu. Jeśli tylko kod, który zawieram, obsługuje informacje w prawidłowy sposób, akceptuję każde dane. W końcu w ostatnim przykładzie używałam tablic. Moim zdaniem to dość skomplikowany typ danych.

Head First: Ale zwracałaś łańcuch znaków.

Niestandardowa funkcja: Mogę zwracać dowolne dane. Wszystko sprowadza się do wykorzystania moich możliwości i prawidłowego zastosowania mnie.

Head First: Wiąże się z tym następne pytanie. Jesteś bardzo wymagająca i trzeba przekazywać ci dane.

Niestandardowa funkcja: Skąd ten niedorzeczny pomysł? Jeśli programista odpowiednio mnie skonfiguruje, będzie mógł wywoływać mnie bez podawania zmiennych. Jeśli nie chcesz przekazywać do mnie danych, nie wpisuj zmiennych w nawiasach po nazwie w mojej definicji. Nie rozumiem jednak, dlaczego ktoś miałby rezygnować z przekazywania do mnie informacji albo przyjmowania danych, które zwracam za pomocą instrukcji return.

Head First: Nasz czas się kończy — dziękuję za rozmowę.

Niestandardowa funkcja: Nie ma za co. Istnieję po to, żeby służyć. A może służę, aby istnieć? Albo służyć istnieniu? Coś w tym rodzaju.

Jazda próbna

Zmodyfikuj skrypt do przeszukiwania bazy przez użycie w nim funkcji `build_query()`.

Utwórz w skrypcie *search.php* nową funkcję `build_query()`. Zastąp pierwotny kod wywołaniem tej funkcji. Prześlij skrypt na serwer WWW i sprawdź w przeglądarce, czy wyszukiwanie działa prawidłowo.

Nowa niestandardowa funkcja build—query() jest wygodna, ale na razie nie określa kolejności wyników wyszukiwania. Czy możemy dodać nowy parametr, który uruchomi sortowanie?

Oczywiście. Do funkcji `build_query()` możemy przekazać dwa parametry zamiast jednego.

Przekazaliśmy już do funkcji argument `$user_search`, który zawiera szukane słowa. Teraz potrzebujemy drugiego argumentu, `$sort`, który określi, jak sortować dane. Nowy argument będzie kontrolował kolejność danych zwracanych przez zapytanie. Istnieje sześć możliwości, które wymieniliśmy na stronie 566: sortowanie według kolumn `title`, `state` i `date_posted` tabeli `riskyjobs` na **dwa** sposoby — rosnąco i malejąco.

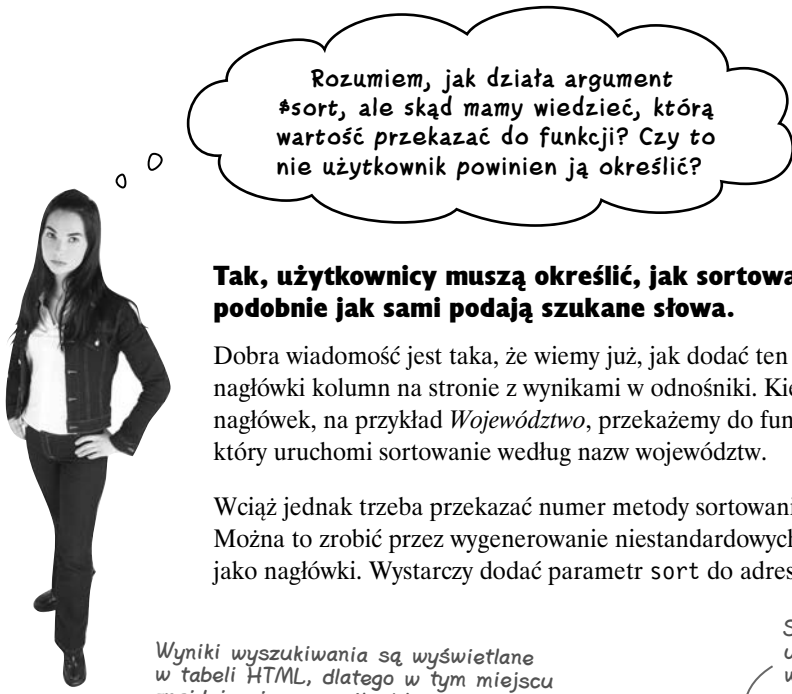
Aby określić metodę sortowania, można zapisać łańcuchy z klauzulami `ORDER BY` w zmiennej `$sort`. Inne rozwiązanie to użycie liczb od 1 do 6, które będą reprezentować poszczególne sposoby porządkowania danych:

```
$sort == 1 ➡ ORDER BY title
$sort == 2 ➡ ORDER BY title DESC
$sort == 3 ➡ ORDER BY state
$sort == 4 ➡ ORDER BY state DESC
$sort == 5 ➡ ORDER BY date_posted
$sort == 6 ➡ ORDER BY date_posted DESC
```

Sortowanie według opisu stanowiska nie jest zbyt przydatne, ponieważ kolejność alfabetyczna nie ma w tym przypadku większego znaczenia.

Te liczby i ich znaczenie ustaliliśmy w pełni arbitralnie. Trzeba jedynie pamiętać, aby używać tych wartości konsekwentnie.

Czy jednak liczby nie są mniej zrozumiałe w trakcie analizowania kodu? To prawda, bez odpowiednich komentarzy są mniej czytelne, jednak zastosowanie liczb całkowitych wynika z istotnej przyczyny. Jeśli użyjemy łańcuchów z klauzulą `ORDER BY`, dane będą częścią nagłówka-odnośnika i pojawią się w adresie URL skryptu. Spowoduje to ujawnienie nazw kolumn, czego z uwagi na bezpieczeństwo należy unikać.



Tak, użytkownicy muszą określić, jak sortować wyniki wyszukiwania, podobnie jak sami podają szukane słowa.

Dobra wiadomość jest taka, że wiemy już, jak dodać ten mechanizm — przekształcimy nagłówki kolumn na stronie z wynikami w odnośniki. Kiedy użytkownik kliknie dany nagłówek, na przykład *Województwo*, przekażemy do funkcji `build_query()` numer, który uruchomi sortowanie według nazw województw.

Wciąż jednak trzeba przekazać numer metody sortowania z odnośnika do skryptu. Można to zrobić przez wygenerowanie niestandardowych odnośników wyświetlanych jako nagłówki. Wystarczy dodać parametr `sort` do adresu URL:

Wyniki wyszukiwania są wyświetlane w tabeli HTML, dlatego w tym miejscu znajduje się znacznik <td>.

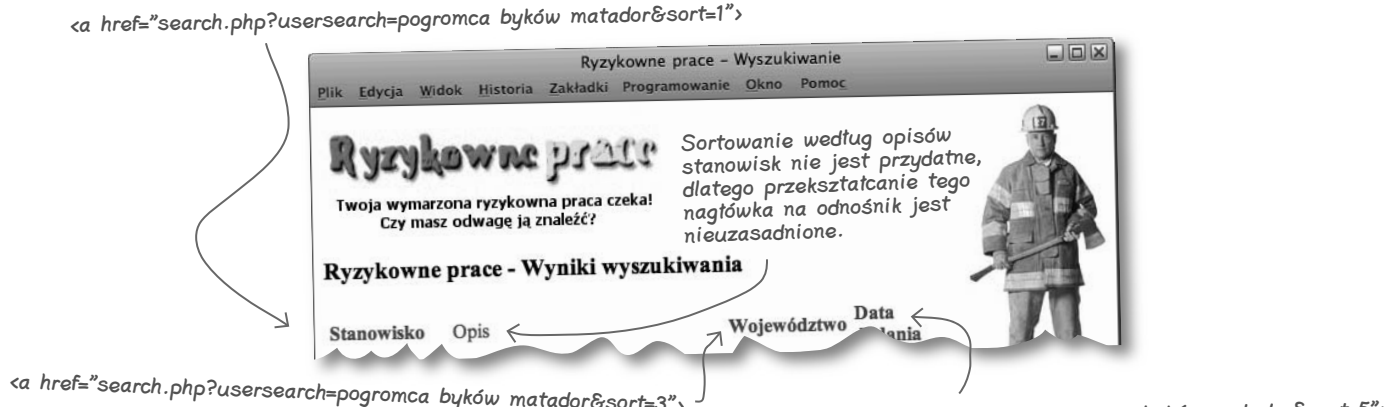
Skrypt powinien odświeżać stronę, kiedy użytkownik kliknie nagłówek kolumny w celu posortowania wyników, dlatego należy utworzyć formularz autoreferencyjny.

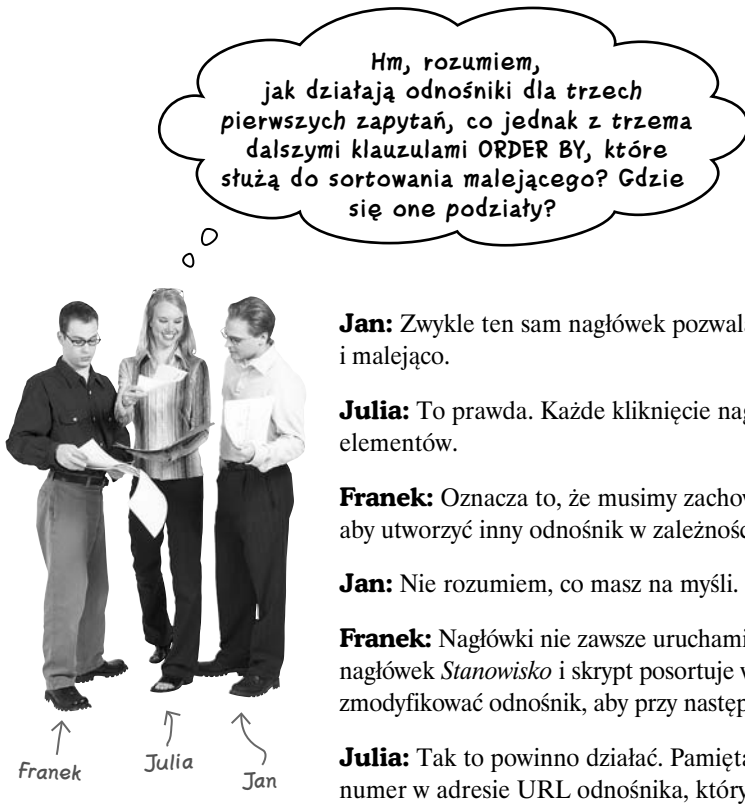
```
$sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] .
  '?usersearch=' . $user_search . '&sort=3">Województwo</a></td>';
```

Funkcja build_query(), która służy do wyświetlenia wyników, potrzebuje słów szukanych przez użytkownika, dlatego przekazujemy je w adresie URL.

Przekazujemy numer metody sortowania, aby określić pożądaną metodę porządkowania wyników. Ponieważ jest to odnośnik Województwo, zmienna „sort" ma wartość 3.

Na wygenerowanej stronie z wynikami każdy nagłówek-odnośnik (oprócz tytułu *Opis stanowiska*) ma niestandardowy adres URL, który obejmuje wartość zmiennej `sort`. Określa ona sposób sortowania ofert.





Jan: Zwykle ten sam nagłówek pozwala użytkownikom sortować dane rosnąco i malejąco.

Julia: To prawda. Każde kliknięcie nagłówka powoduje odwrócenie kolejności elementów.

Franek: Oznacza to, że musimy zachować stan nagłówka po każdym kliknięciu, aby utworzyć inny odnośnik w zależności od aktualnej zawartości danego odsyłacza.

Jan: Nie rozumiem, co masz na myśli.

Franek: Nagłówki nie zawsze uruchamiają to samo sortowanie. Na przykład: jeśli klikniesz nagłówek *Stanowisko* i skrypt posortuje wyniki rosnąco według nazw stanowisk, trzeba zmodyfikować odnośnik, aby przy następnym kliknięciu uporządkował dane malejąco.

Julia: Tak to powinno działać. Pamiętaj, że każdej metodzie sortowania odpowiada numer w adresie URL odnośnika, który informuje skrypt o pożądanym uporządkowaniu danych. Ponieważ generujemy te odsyłacze, możemy kontrolować liczby, które się w nich znajdą.

Jan: Teraz rozumiem. Zadanie polega więc na zbudowaniu kodu tak, aby generował właściwy odnośnik na podstawie ostatnio użytej metody sortowania, prawda?

Franek: Czy nie możemy użyć do tego kilku instrukcji `if`? Struktura ta dobrze nadaje się do wykonywania takich operacji.

Jan: Tak, to rozwiązanie zadziała, jednak potrzebnych jest kilka warunków opartych na tych samych danych — numerze metody sortowania. Warto znaleźć lepszy sposób na wybór liczby niż używanie wielu zagnieżdżonych instrukcji `if-else`.

Julia: To bardzo trafna uwaga. Mamy świetną okazję do wypróbowania nowej instrukcji, którą niedawno poznałam. Struktura `switch` pozwala podejmować różne decyzje — więcej niż dwie — na podstawie jednej wartości.

Franek: Wygląda to bardzo atrakcyjnie. Wypróbujmy ją.

Jan: Zgadzam się. Zróbmy wszystko, aby uniknąć skomplikowanych instrukcji `if-else`. Przerażają mnie!

Julia: Mnie też. Myślę, że instrukcja `switch` będzie idealnym rozwiązaniem…

SWITCH obsługuje więcej decyzji niż IF

Instrukcja switch umożliwia wydajne sprawdzanie wyrażenia i uruchamianie na podstawie jego wartości jednego z kilku bloków kodu. Aby uzyskać podobny efekt, czasem — zwłaszcza przy większej liczbie możliwości — potrzeba naprawdę wielu instrukcji if-else.

Aby sprawdzić wszystkie możliwe wartości, nie trzeba tworzyć zagnieżdżonych instrukcji if-else. W zamian należy zbudować instrukcję switch i umieścić w niej etykiety case odpowiadające poszczególnym wartościom. Na końcu każdego bloku case należy umieścić polecenie break;, które nakazuje skryptowi PHP opuszczenie całej instrukcji switch i pominięcie pozostałych etykiet. Gwarantuje to, że skrypt uruchomi kod z co najwyżej jednego bloku case.

Instrukcja SWITCH zawiera szereg etykiet CASE, które uruchamiają różne bloki kodu w zależności od wartości zmiennej.

Przyjrzyjmy się przykładowi użycia instrukcji switch:

```
switch ($benefit_code) {
case 1:
  $benefits = 'Poważna choroba – 10 dni zwolnienia';
  break;
case 2:
  $benefits = 'Tylko w przypadku śmierci lub kalectwa – miesięczna odprawa';
  break;
case 3:
case 4:
  $benefits = 'Powodzenia!';
  break;
default:
  $benefits = 'Brak.';
}
echo 'Oferujemy cztery korzystne pakiety pomocowe';
echo 'Wybrany plan: ' . $benefits;
```

Instrukcja switch sprawdza tę wartość. Zmienna ta kontroluje wszystkie bloki.

Ten kod jest uruchamiany tylko wtedy, jeśli zmienna $benefit_code ma wartość 1.

Instrukcja break informuje skrypt PHP, że należy wyjść z instrukcji switch.

Jeśli chcesz uruchomić te same operacje dla kilku wartości, instrukcję break umieść tylko po kodzie ostatniego bloku z grupy.

Jeżeli zmienna $benefit_code ma wartość różną od 1, 2, 3 lub 4, skrypt uruchomi domyślny kod.

To nie do końca prawda. Są tylko trzy pakiety, ponieważ wartościom 3 i 4 odpowiada ten sam kod, co wynika z braku instrukcji break po bloku dla liczby 3.

Ćwiczenie

Aplikacja Ryzykowne prace zawiera nową funkcję, `generate_sort_links()`, która umożliwia sortowanie wyników wyszukiwania przez kliknięcie nagłówków. Niestety, w kodzie brakuje kilku ważnych fragmentów. Uzupełnij tę funkcję. Nie zapomnij numerów metod sortowania: 1 = rosnąco według stanowisk, 2 = malejąco według stanowisk, 3 = rosnąco według województw, 4 = malejąco według województw, 5 = rosnąco według daty dodania, 6 = malejąco według daty dodania.

```
..... generate_sort_links($user_search, $sort) {
  $sort_links = '';
  ..... ($sort) {
  case 1:
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search . '&sort= .... ">Stanowisko</a></td><td>Opis</td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Województwo</a></td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search . '&sort= ....">Data dodania</a></td>';
    ..........
  case 3:
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Stanowisko</a></td><td>Opis</td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Województwo</a></td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Data dodania</a></td>';
    ..........
  case 5:
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Stanowisko</a></td><td>Opis</td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Województwo</a></td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Data dodania</a></td>';
    ..........
  ..........
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Stanowisko</a></td><td>Opis</td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Województwo</a></td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= .... ">Data dodania</a></td>';
  }

  return ............... ;
}
```

To domyślne nagłówki, które pojawią się, jeśli użytkownik nie wybierze metody sortowania.

Ćwiczenie: Rozwiązanie

Aplikacja Ryzykowne prace zawiera nową funkcję, `generate_sort_links()`, która umożliwia sortowanie wyników wyszukiwania przez kliknięcie nagłówków. Niestety, w kodzie brakuje kilku ważnych fragmentów. Uzupełnij tę funkcję. Nie zapomnij numerów metod sortowania: 1 = rosnąco według stanowisk, 2 = malejąco według stanowisk, 3 = rosnąco według województw, 4 = malejąco według województw, 5 = rosnąco według daty dodania, 6 = malejąco według daty dodania.

```
function generate_sort_links($user_search, $sort) {
  $sort_links = '';
  switch ($sort) {
  case 1:
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search . '&sort= 2 ">Stanowisko</a></td><td>Opis</td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 3 ">Województwo</a></td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search . '&sort= 5 ">Data dodania</a></td>';
    break;
  case 3:
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 1 ">Stanowisko</a></td><td>Opis</td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 4 ">Województwo</a></td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 5 ">Data dodania</a></td>';
    break;
  case 5:
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 1 ">Stanowisko</a></td><td>Opis</td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 3 ">Województwo</a></td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 6 ">Data dodania</a></td>';
    break;
  default;
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 1 ">Stanowisko</a></td><td>Opis</td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 3 ">Województwo</a></td>';
    $sort_links .= '<td><a href = "' . $_SERVER['PHP_SELF'] . '?usersearch=' .
      $user_search .'&sort= 5 ">Data dodania</a></td>';
  }

  return $sort_links ;
}
```

Jeśli zmienna $sort ma wartość 1, posortowaliśmy już oferty rosnąco według nazw stanowisk, dlatego teraz trzeba uporządkować je w kolejności malejącej.

Jeśli zmienna $sort nie została ustawiona lub ma wartość 2, 4 albo 6, należy wyświetlić wyjściowe odnośniki, które sortują dane rosnąco.

Sortowanie w funkcji build_query()

Mamy już dwie funkcje do obsługi wyszukiwania w serwisie Ryzykowne prace. Funkcja `build_query()` tworzy zapytanie SQL na podstawie szukanych słów, które wpisał użytkownik, a funkcja `generate_sort_links()` generuje nagłówki-odnośniki, umożliwiające sortowanie ofert. Jednak funkcja `build_query()` nie jest jeszcze w pełni gotowa, ponieważ zapytanie, które generuje, na razie nie sortuje wyników. Funkcja powinna dodawać do zapytania klauzulę `ORDER BY`. Jednak musi być to **właściwa** klauzula tego typu, określona na podstawie nowego argumentu `$sort`:

```
function build_query($user_search, $sort) {
  $search_query = "SELECT * FROM riskyjobs";

  ...

  // Dodawanie klauzuli ze słowem kluczowym WHERE do zapytania wyszukującego dane.
  if (!empty($where_clause)) {
    $search_query .= " WHERE $where_clause";
  }
  // Określanie sortowania w zapytaniu na podstawie zmiennej $sort.
  switch ($sort) {
  // Rosnąco według stanowisk.
  case 1:
    $search_query .= " ORDER BY title";
    break;
  // Malejąco według stanowisk.
  case 2:
    $search_query .= " ORDER BY title DESC";
    break;
  // Rosnąco według województw.
  case 3:
    $search_query .= " ORDER BY state";
    break;
  // Malejąco według województw.
  case 4:
    $search_query .= " ORDER BY state DESC";
    break;
  // Rosnąco według dat dodania (najstarsze na początku).
  case 5:
    $search_query .= " ORDER BY date_posted";
    break;
  // Malejąco według dat dodania (najnowsze na początku).
  case 6:
    $search_query .= " ORDER BY date_posted DESC";
    break;
  default:
    // Nie określono metody sortowania, dlatego funkcja nie porządkuje danych.
  }
  return $search_query ;
}
```

Do funkcji oprócz argumentu $user_search przekazujemy teraz zmienną $sort.

To nowy kod funkcji build_query(). Instrukcja switch sprawdza wartość zmiennej $sort i dodaje odpowiednią instrukcję ORDER BY na końcu zapytania wyszukującego dane.

Jeśli użytkownik wczyta stronę z ofertami bez kliknięcia nagłówka kolumny, zmienna $sort będzie pusta, dlatego domyślnie funkcja w ogóle nie sortuje wyników.

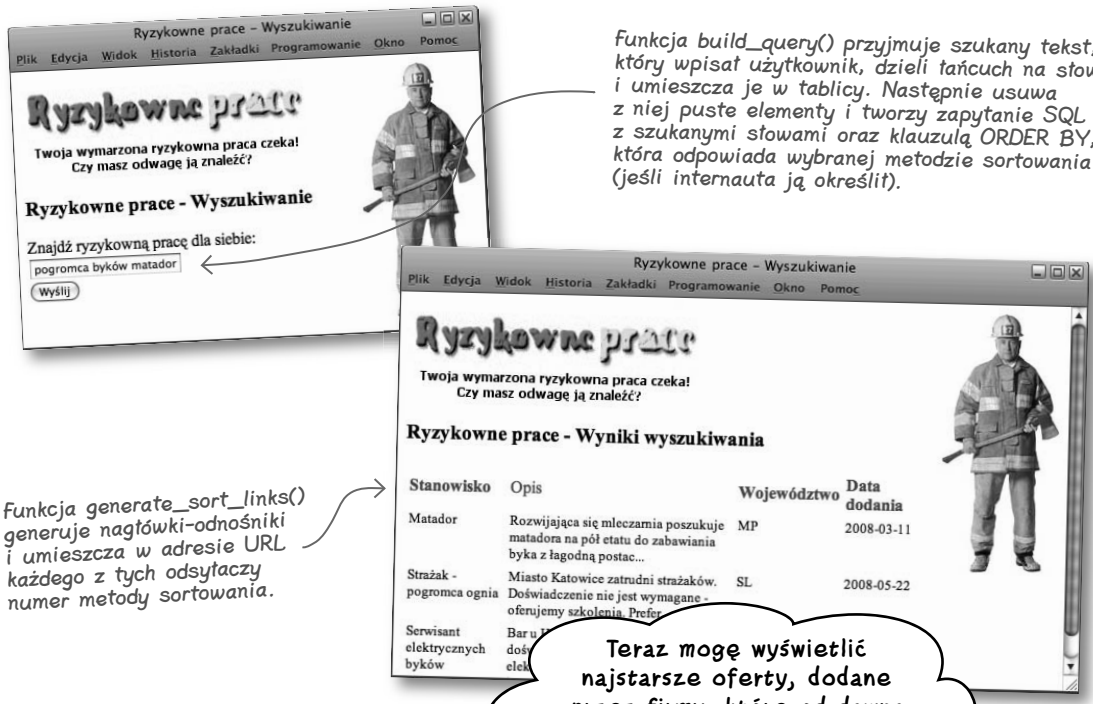
Podobnie jak wcześniej funkcja zwraca zmienną $search_query, jednak tym razem kończy się ona klauzulą ORDER BY.

Jazda próbna

Usprawnij skrypt do wyszukiwania ofert przez dodanie dwóch nowych niestandardowych funkcji.

Utwórz w skrypcie *search.php* funkcję `generate_sort_links()`, a następnie dodaj nowy kod do funkcji `build_query()`, aby generowała zapytanie, które posortuje wyniki. Nie zapomnij wywołać funkcji `generate_sort_links()` w miejscu, w którym powinny pojawić się nagłówki kolumn z wynikami.

Prześlij skrypt na serwer WWW, otwórz w przeglądarce stronę *search.html* i uruchom wyszukiwanie. Kliknij nagłówki nad wynikami zapytania, aby posortować oferty według różnych danych. Kliknij ten sam nagłówek dwukrotnie, aby zmienić uporządkowanie z rosnącego na malejące.



Funkcja build_query() przyjmuje szukany tekst, który wpisał użytkownik, dzieli łańcuch na słowa i umieszcza je w tablicy. Następnie usuwa z niej puste elementy i tworzy zapytanie SQL z szukanymi słowami oraz klauzulą ORDER BY, która odpowiada wybranej metodzie sortowania (jeśli internauta ją określił).

Funkcja generate_sort_links() generuje nagłówki-odnośniki i umieszcza w adresie URL każdego z tych odsyłaczy numer metody sortowania.

Ryzykowne prace – Wyszukiwanie

Plik Edycja Widok Historia Zakładki Programowanie Okno Pomoc

Ryzykowne prace

Twoja wymarzona ryzykowna praca czeka!
Czy masz odwagę ją znaleźć?

Ryzykowne prace - Wyniki wyszukiwania

Stanowisko	Opis	Województwo	Data dodania
Chodzenie po kremie	Potrzebujemy osób, które zechcą przetestować teorię dotyczącą możliwości chodzenia po kremi...	WP	2008-07-24
Treser rekinów	Uczenie rekinów ciekawych sztuczek prezentowanych w nowym parku wodnym. Będziesz samodzielnie p...	SL	2008-04-28
Kontroler napięcia	Praca w terenie, polegająca na sprawdzaniu napięcia w zakresie od 3 do ponad 250 voltów. Otrzymas...	MZ	2008-06-28
Instalator anten	Będziesz instalował anteny i inne metalowe narzędzia odbiorcze na dachach najwyższych budynków ...	MZ	2008-09-04
	Potrzebny doświadczony proktolog, który chce pracować z dużymi ...nie w naszym kr...	MA	2008-07-29
	...vców wymagają ...trzebujemy osoby o ...ach do usuwania rdzy...	WP	2008-08-17
Matador	...mleczarnia poszukuje matadora na pół etatu do zabawiania byka z łagodną postac...	MP	2008-03-11
	Czołowa firma fotografująca znane	MZ	2008-03-24

Ofert jest zbyt wiele, aby wyświetlać je na jednej stronie.

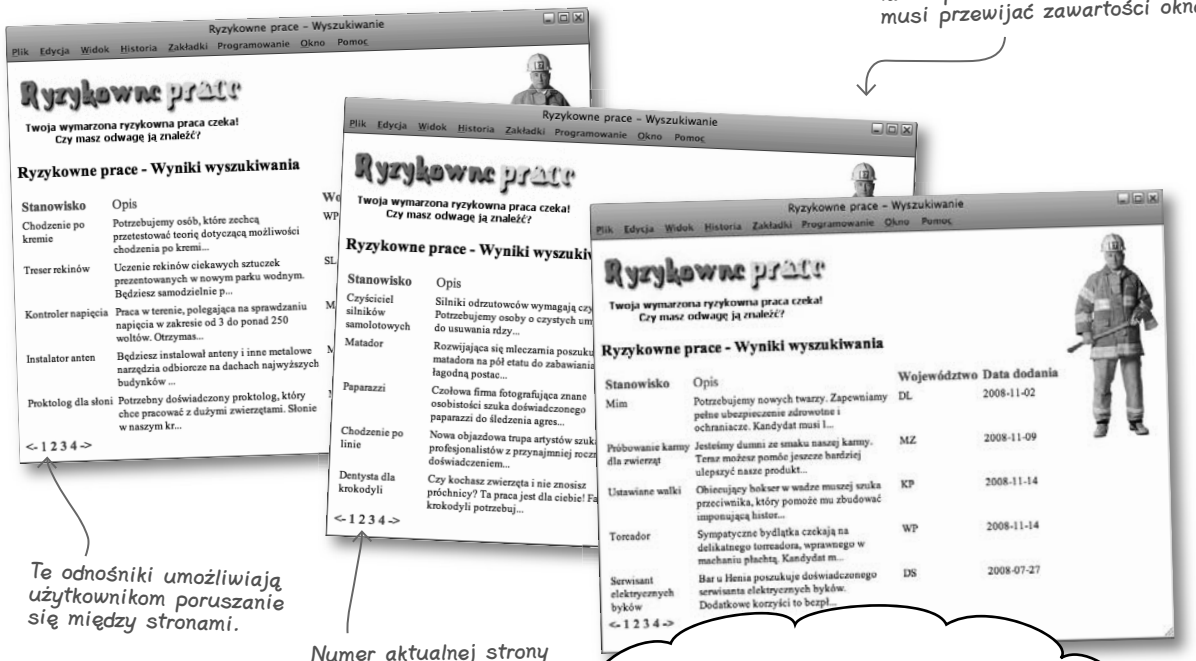
WYSIL SZARE KOMÓRKI

Jak w innych witrynach rozwiązano problem wyświetlania wielu wyników wyszukiwania na jednej stronie?

Możemy podzielić wyniki na strony

Obecnie wyświetlamy wszystkie wyniki na jednej stronie, co jest niewygodne, jeśli zapytanie pasuje do wielu ofert. Zamiast zmuszać użytkownika do przewijania długiej strony w górę i w dół, możemy wyświetlać dane za pomocą techniki zwanej **stronicowaniem**. Wymaga to podzielenia znalezionych ofert na grupy i wyświetlenia każdego zbioru na odrębnej stronie:

Każda strona obejmuje pięć wyników i odnośniki, które zapewniają dostęp do innych porcji danych. Użytkownik może łatwo przechodzić między stronami i nie musi przewijać zawartości okna.



Świetnie, ale jak podzielimy wyniki na grupy? Zapytanie SQL zwraca wszystkie oferty pasujące do szukanych słów.

Stronicowanie polega na podziale wyników zapytania na grupy i wyświetlaniu każdej z nich na odrębnej stronie.

Potrzebujemy zapytania, które zwróci tylko podzbiór wyników, a nie wszystkie.

Na szczęście język SQL udostępnia potrzebny mechanizm — klauzulę `LIMIT`. Przyjrzyjmy się jej ponownie, aby zobaczyć, jak użyć jej do podziału wyników na pięcioelementowe grupy…

Pobieranie tylko potrzebnych wierszy dzięki klauzuli LIMIT

Kluczem do wskazywania wierszy wyświetlanych na danej stronie jest nowa klauzula w zapytaniu — `LIMIT`. Aby pobrać nie więcej niż pięć wierszy, należy umieścić na końcu zapytania polecenie `LIMIT 5`:

```
SELECT * FROM riskyjobs
  ORDER BY title
  LIMIT 5
```

Bez klauzuli WHERE zapytanie zwróci wszystkie oferty z bazy, co odpowiada przeszukiwaniu danych bez szukanego łańcucha znaków.

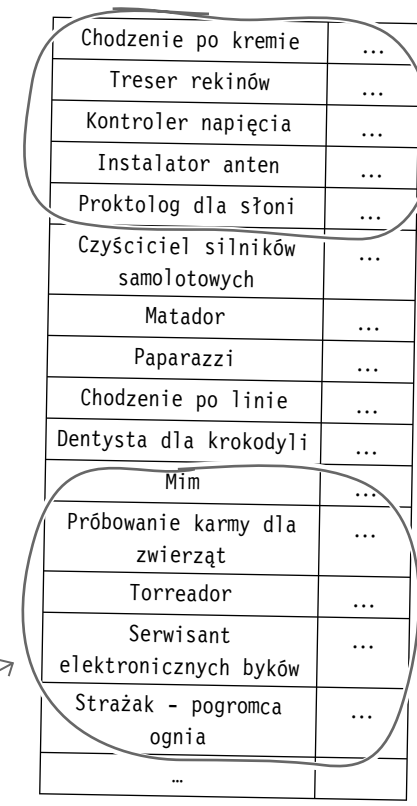
Niezależnie od liczby znalezionych ofert zapytanie zwróci tylko pięć pierwszych wyników.

Klauzula LIMIT określa, ile wierszy zwróci zapytanie SQL.

Prawdopodobnie przypominasz sobie, że do tworzenia zapytania w aplikacji Ryzykowne prace użyliśmy niestandardowej funkcji `build_query()`. Aby to zapytanie wyświetlało tylko pięć pierwszych ofert, należy po jego utworzeniu dołączyć do niego polecenie `LIMIT 5`:

```
$query = build_query($user_search, $sort);
$query = $query . " LIMIT 5";
```

Dodanie klauzuli LIMIT na końcu zapytania ogranicza liczbę zwracanych wierszy (tu do pięciu).



To rozwiązanie zadziała dla pięciu pierwszych wierszy wyników, co jednak z dalszymi grupami? Aby pobrać następne oferty ze zbioru wyników, trzeba zmodyfikować klauzulę `LIMIT`. Ale jak to zrobić? Polecenie `LIMIT 10` zwróci pierwszych 10 wierszy, dlatego nie jest dobrym rozwiązaniem. Potrzebne są wiersze od 6. do 10., które pobierzemy za pomocą innej składni klauzuli `LIMIT`. Jeśli podasz dwa argumenty, pierwszy określi zakres pomijanych elementów, a drugi — liczbę pobieranych wyników. Poniższe zapytanie pobierze wiersze od 11. do 15., czyli trzecią stronę wyników:

```
$query = build_query($user_search, $sort);
$query = $query . " LIMIT 10, 5";
```

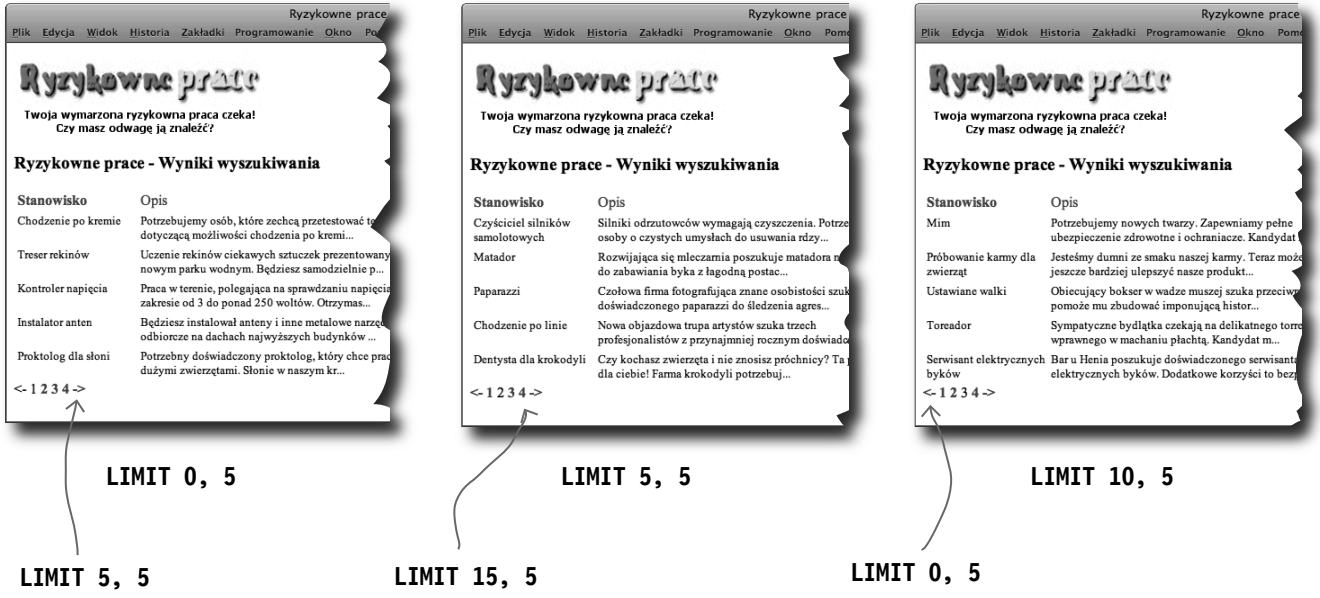
Pierwszy argument klauzuli LIMIT określa, ile wierszy zapytanie ma pominąć. Tu jest to 10 pierwszych wyników.

Drugi argument określa liczbę pobieranych wierszy. Podobnie jak wcześniej zapytanie zwróci pięć wyników.

Zarządzanie odnośnikami związanymi z klauzulą LIMIT

Ważnym aspektem stronicowania jest udostępnianie odnośników, które umożliwiają użytkownikom poruszanie się po stronach z wynikami. Do utworzenia odsyłaczy, które pojawiają się w dolnej części każdej strony z ofertami, możemy użyć klauzuli `LIMIT`. Na przykład odnośniki *Następna* i *Poprzednia* powinny mieć różne klauzule tego rodzaju. To samo dotyczy odnośników z numerami, które umożliwiają przejście bezpośrednio do danej strony wyników.

Oto klauzule `LIMIT` dla trzech pierwszych stron wyników wyszukiwania oraz wybranych odnośników:



To proste. Musimy tylko napisać zestaw zapytań z różnymi klauzulami LIMIT w każdym z nich, prawda?

Mniej więcej. Potrzebujemy różnych klauzul `LIMIT` dostosowanych do strony i odnośnika, jednak możemy je wygenerować zamiast ręcznie zapisywać liczne zapytania.

Wystarczy nieco zmodyfikować funkcję `build_query()` i umieścić właściwą klauzulę `LIMIT` na końcu tworzonego zapytania.

Kontrolowanie danych potrzebnych do obsługi stronicowania

Aby dodać mechanizm stronicowania do funkcji `build_query()`, musimy ustawić i kontrolować kilka zmiennych. Posłużą one do określania, które wyniki należy pobrać i wyświetlić na danej stronie. Te zmienne pomogą także w generowaniu odnośników nawigacyjnych, które znajdują się w dolnej części strony.

`$cur_page`

Numer aktualnej strony (zmienna `$cur_page`) pobieramy za pomocą tablicy `$_GET` z adresu URL skryptu. Jeśli adres nie zawiera numeru, należy ustawić tę zmienną na pierwszą stronę (1).

`$results_per_page`

Liczba wyników na stronę. Można ją określić na podstawie wyglądu i stylu strony oraz tego, ile elementów się na niej zmieści. Ta wartość to drugi argument klauzuli `LIMIT`.

`$skip`

Zmienna `$skip` zawiera liczbę wierszy, które należy pominąć, aby wyświetlić wyniki z danej strony. Ta zmienna wyznacza początkowy wynik widoczny na stronie i jest pierwszym argumentem klauzuli `LIMIT`.

`$total`

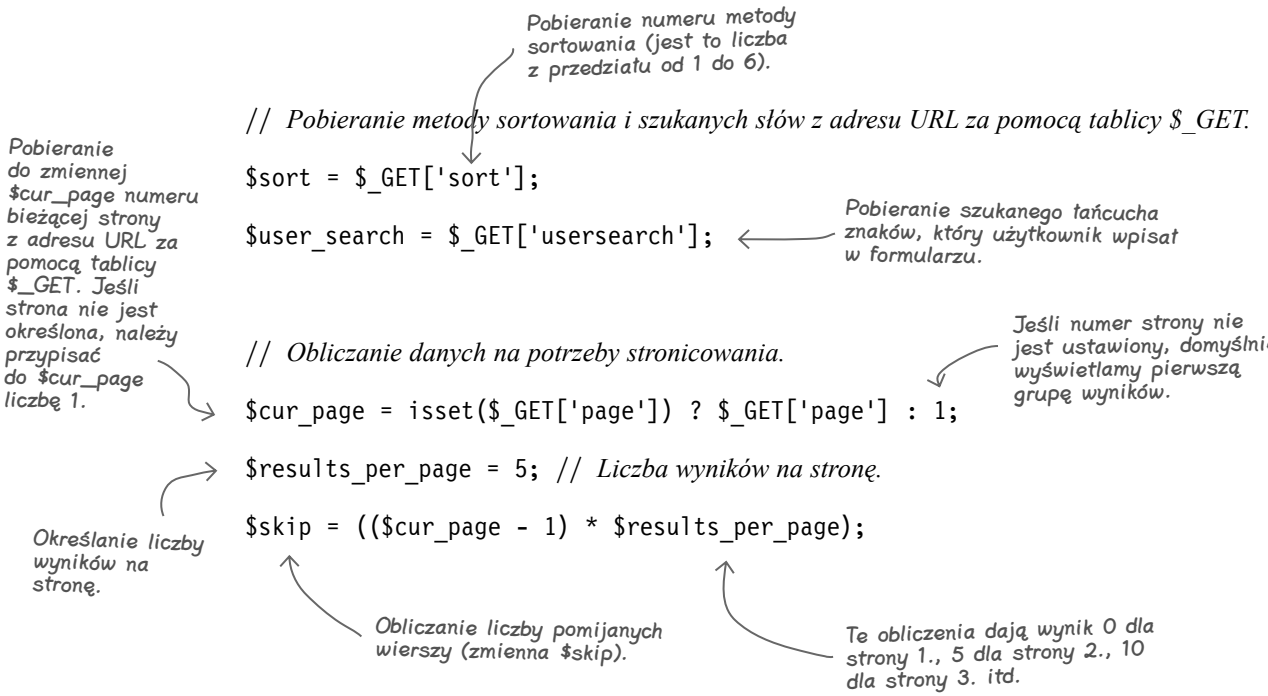
Uruchom zapytanie bez klauzuli `LIMIT`, aby pobrać wszystkie wiersze. Zapisz liczbę wyników w zmiennej `$total`. Określa ona sumę wszystkich zwróconych wierszy.

`$num_pages`

Aby obliczyć liczbę stron (`$num_pages`), podziel zmienną `$total` przez `$results_per_page`. Każde zapytanie zwraca wiersze, których liczbę zapisujemy w zmiennej `$total`. Wyniki trzeba wyświetlić na stronach, a liczbę wierszy na każdej z nich zawiera zmienna `$results_per_page`. Jest więc `$num_pages` stron, a obecną grupę wyników wskazuje zmienna `$cur_page`.

Konfigurowanie zmiennych używanych do stronicowania

Większość potrzebnych zmiennych można ustawić na podstawie informacji z adresu URL, które pobierzemy ze zmiennej superglobalnej `$_GET`. Bezpośrednio z żądania `GET` pochodzą na przykład wartości zmiennych `$sort`, `$user_search` i `$cur_page`. Następnie możemy użyć tych zmiennych, aby określić liczbę pomijanych wierszy (zmienna `$skip`). Inaczej rzecz się ma ze zmienną `$results_per_page`, w której należy zapisać, ile wyników ma pojawić się na stronie. Liczbę tę dobiera programista na podstawie układu strony z wynikami.



Nadal brakuje kilku ważnych zmiennych — `$total` i `$num_pages`. Te zmienne można ustawić dopiero po wywołaniu początkowego zapytania, które pozwala ustalić, ile wyników znaleziono w bazie. Po określeniu liczby wierszy można określić wartości tych zmiennych i ograniczyć wyniki za pomocą klauzuli `LIMIT`.

Poprawianie zapytania, które pobiera stronicowane wyniki

Po przygotowaniu zmiennych musimy poprawić skrypt do wyszukiwania ofert, aby zamiast zwracać wszystkie wyniki, pobierał tylko ich część, potrzebną na wyświetlanej stronie. Najpierw trzeba uruchomić zapytanie, by ustawić zmienną `$total` i obliczyć wartość zmiennej `$num_pages`. Następnie należy utworzyć drugie zapytanie i użyć w nim zmiennych `$skip` oraz `$results_per_page` do wygenerowania klauzuli `LIMIT`, która znajdzie się na końcu tego zapytania. Oto poprawiona część skryptu *search.php* (wyróżniliśmy w niej nowe fragmenty kodu):

Funkcja mysqli_num_rows() zwraca liczbę wszystkich wierszy pobranych przez zapytanie.

To zapytanie nie ma klauzuli LIMIT, dlatego pobiera wszystkie wiersze.

```
// Zapytanie, które określa liczbę wierszy.
$query = build_query($user_search, $sort);
$result = mysqli_query($dbc, $query);
$total = mysqli_num_rows($result);
$num_pages = ceil($total / $results_per_page);

// Drugie zapytanie, które pobiera podzbiór wyników.
$query = $query . " LIMIT $skip, $results_per_page";
$result = mysqli_query($dbc, $query);
while ($row = mysqli_fetch_array($result)) {
  echo '<tr class="results">';
  echo '<td valign="top" width="20%">' . $row['title'] . '</td>';
  echo '<td valign="top" width="50%">' . substr($row['description'], 0, 100) .
    '...</td>';
  echo '<td valign="top" width="10%">' . $row['state'] . '</td>';
  echo '<td valign="top" width="20%">' . substr($row['date_posted'], 0, 10) . '</td>';
  echo '</tr>';
}
echo '</table>';
```

Zapisywanie liczby wierszy za pomocą wywołania funkcji mysqli_num_rows().

Określanie liczby stron. Skrypt dzieli sumę wierszy przez liczbę wyników na stronę i zaokrągla wynik.

Funkcja ceil() zaokrągla wartość do najbliższej liczby całkowitej.

Należy pominąć tyle wierszy...

...i zwrócić tyle wyników.

Drugie zapytanie — tym razem klauzula LIMIT ogranicza pobierane wiersze do ofert z bieżącej strony.

Generowanie odnośników do nawigacji po stronach

Ustawiliśmy kilka zmiennych i utworzyliśmy nowe zapytanie SQL, która zwraca tylko wiersze wyświetlane na danej stronie. Teraz wystarczy wygenerować odnośniki nawigacyjne, które pojawią się w dolnej części strony. Są to: odsyłacz do poprzedniej strony, odnośniki z numerami, które prowadzą do poszczególnych stron, i odsyłacz do następnej grupy wyników. Mamy już wszystkie informacje potrzebne do utworzenia tych odnośników. Warto przyjrzeć się tym danym jeszcze raz, aby dokładnie zrozumieć, do czego służą.

`$user_search`

Każdy odnośnik musi zawierać informacje o tekście wpisanym przez użytkownika, dlatego w adresie URL trzeba umieścić szukane słowa.

`$cur_page`

Działanie odnośników nawigacyjnych zależy od bieżącej strony, dlatego trzeba podać jej numer w każdym adresie URL.

`$num_pages`

Trzeba ustalić, ile jest stron z wynikami, aby wygenerować odnośnik do każdej z nich.

`$sort`

W odnośnikach do stron ważny jest też numer metody sortowania, ponieważ jeśli skrypt nie zachowa kolejności wierszy, stronicowanie nie będzie miało sensu.

Wiemy już, jakie informacje są potrzebne do wygenerowania odnośników nawigacyjnych, dlatego możemy napisać kod PHP, który je utworzy. Nowy fragment można umieścić bezpośrednio w skrypcie *search.php*, jednak możemy też utworzyć niestandardową funkcję. Dzięki temu kod skryptu do generowania listy ofert będzie dużo prostszy, a do utworzenia odnośników wystarczy jeden wiersz z wywołaniem nowej funkcji `generate_page_links()`.

Jedyny problem polega na tym, że skrypt nie powinien wywoływać tej funkcji, jeśli istnieje tylko jedna strona wyników. Dlatego przed uruchomieniem funkcji `generate_page_links()` trzeba sprawdzić liczbę stron. Poniższy kod określa tę liczbę i wywołuje funkcję. Potrzebne informacje skrypt przekazuje do funkcji za pomocą argumentów:

```
if ($num_pages > 1) {
  echo generate_page_links($user_search, $sort, $cur_page, $num_pages);
}
```

Najpierw należy sprawdzić, czy istnieje więcej niż jedna strona wyników. Jeśli nie, nie trzeba generować odsyłaczy.

Przekazywanie szukanego tekstu, numeru metody sortowania, numeru aktualnej strony i liczby wszystkich stron. Wartości te posłużą do wygenerowania odnośników.

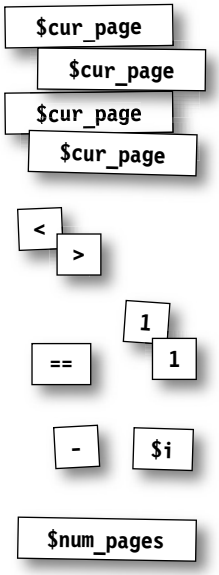
Magnesiki z kodem PHP i MySQL

Funkcja `generate_page_links()` jest już prawie gotowa, jednak brakuje kilku jej fragmentów. Użyj magnesików do uzupełnienia kodu, aby umożliwić generowanie odnośników nawigacyjnych w aplikacji Ryzykowne prace.

```
function generate_page_links($user_search, $sort, $cur_page, $num_pages) {
  $page_links = '';

  // Jeśli nie jest to pierwsza grupa wyników, należy wygenerować odnośnik do poprzedniej strony.
  if ( ........................... ) {
    $page_links .= '<a href="' . $_SERVER['PHP_SELF'] .
      '?usersearch=' . $user_search .
      '&sort=' . $sort .
      '&page=' . ( ..................... ) . '"><-</a> ';
  }
  else {
    $page_links .= '<- ';
  }

  // Przejście w pętli po stronach i wygenerowanie odnośników z numerami.
  for ($i = 1; $i <= $num_pages; $i++) {
    if ( ....................... ) {
      $page_links .= ' ' . $i;
    }
    else {
      $page_links .= ' <a href="' . $_SERVER['PHP_SELF'] .
        '?usersearch=' . $user_search .
        '&sort=' . $sort .
        '&page=' . $i . '"> ' . $i . '</a>';
    }
  }

  // Jeśli nie jest to ostatnia grupa wyników, należy wygenerować odnośnik do następnej strony.
  if ( ............................ ) {
    $page_links .= ' <a href="' . $_SERVER['PHP_SELF'] .
      '?usersearch=' . $user_search .
      '&sort=' . $sort .
      '&page=' . ($cur_page + 1) . '">-></a>';
  }
  else {
    $page_links .= ' ->';
  }

  return $page_links;
}
```

Odnośnik do poprzedniej strony ma postać strzałki skierowanej w lewo: „<—”.

Odnośnik do następnej strony to strzałka skierowana w prawo: „—>”.



Magnesiki z kodem PHP i MySQL. Rozwiązanie

Funkcja `generate_page_links()` jest już prawie gotowa, jednak brakuje kilku jej fragmentów. Użyj magnesików do uzupełnienia kodu, aby umożliwić generowanie odnośników nawigacyjnych w aplikacji Ryzykowne prace.

```
function generate_page_links($user_search, $sort, $cur_page, $num_pages) {
  $page_links = '';

  // Jeśli nie jest to pierwsza grupa wyników, należy wygenerować odnośnik do poprzedniej strony.
  if ( $cur_page > 1 ) {
    $page_links .= '<a href="' . $_SERVER['PHP_SELF'] .
      '?usersearch=' . $user_search .
      '&sort=' . $sort .
      '&page=' . ( $cur_page - 1 ) . '"><-</a> ';
  }
  else {
    $page_links .= '<- ';
  }

  // Przejście w pętli po stronach i wygenerowanie odnośników z numerami.
  for ($i = 1; $i <= $num_pages; $i++) {
    if ( $cur_page == $i ) {
      $page_links .= ' ' . $i;
    }
    else {
      $page_links .= ' <a href="' . $_SERVER['PHP_SELF'] .
        '?usersearch=' . $user_search .
        '&sort=' . $sort .
        '&page=' . $i . '"> ' . $i . '</a>';
    }
  }

  // Jeśli nie jest to ostatnia grupa wyników, należy wygenerować odnośnik do następnej strony.
  if ( $cur_page < $num_pages ) {
    $page_links .= ' <a href="' . $_SERVER['PHP_SELF'] .
      '?usersearch=' . $user_search .
      '&sort=' . $sort .
      '&page=' . ($cur_page + 1) . '">-></a>';
  }
  else {
    $page_links .= ' ->';
  }

  return $page_links;
}
```

W każdym odnośniku trzeba przekazać szukane słowa i numer metody sortowania.

Odnośnik do poprzedniej strony ma postać strzałki skierowanej w lewo: „<—”.

Ta zmienna gwarantuje, że każdy odsyłacz prowadzi do tego samego skryptu. W każdym odnośniku zmieniamy jedynie numer strony.

Odsyłacz do konkretnej strony to po prostu jej numer.

Odnośnik do następnej strony to strzałka skierowana w prawo: „—>”.

Tworzenie kompletnego skryptu do wyszukiwania ofert

Wreszcie doszliśmy do kompletnej wersji skryptu do wyszukiwania ofert w serwisie Ryzykowne prace. Skrypt ten wyświetla odpowiednie wyniki wyszukiwania na podstawie tekstu wpisanego przez użytkownika, generuje nagłówki-odnośniki, które umożliwiają posortowanie ofert, dzieli wyniki na grupy i generuje odnośniki nawigacyjne, wyświetlane w dolnej części strony.

```
<?php
  // Ta funkcja tworzy zapytanie na podstawie szukanych słów i numeru metody sortowania.
  function build_query($user_search, $sort) {
    ...

    return $search_query;
  }

  // Ta funkcja tworzy nagłówki-odnośniki na podstawie numeru metody sortowania.
  function generate_sort_links($user_search, $sort) {
    ...

    return $sort_links;
  }

  // Ta funkcja tworzy odnośniki nawigacyjne na podstawie numeru bieżącej
  // strony i ich łącznej liczby.
  function generate_page_links($user_search, $sort, $cur_page, $num_pages) {
    ...

    return $page_links;
  }

  // Pobieranie numeru metody sortowania i szukanych słów z adresu URL za pomocą tablicy $_GET.
  $sort = $_GET['sort'];
  $user_search = $_GET['usersearch'];

  // Obliczanie wartości zmiennych na potrzeby stronicowania.
  $cur_page = isset($_GET['page']) ? $_GET['page'] : 1;
  $results_per_page = 5; // Liczba wyników na stronę.
  $skip = (($cur_page - 1) * $results_per_page);

  // Rozpoczęcie generowania tabeli z wynikami.
  echo '<table border="0" cellpadding="2">';

  // Generowanie nagłówków-odnośników.
  echo '<tr class="heading">';
  echo generate_sort_links($user_search, $sort);
  echo '</tr>';
```

Już wcześniej utworzyliśmy te funkcje, dlatego nie ma potrzebny powtarzać tu wszystkich wierszy kodu.

Pobieranie numeru metody sortowania i szukanego tekstu, które skrypt przekazuje jako żądania GET w adresie URL.

Inicjowanie zmiennych na potrzeby stronicowania, które będą potrzebne do ograniczenia działania zapytania i utworzenia odnośników do poszczególnych stron.

Wywołanie funkcji generate_sort_links() w celu utworzenia nagłówków-odnośników i wyświetlenia ich.

To jeszcze nie wszystko!

search.php

Kompletny skrypt do wyszukiwania ofert — ciąg dalszy

```
// Łączenie się z bazą danych.
require_once('connectvars.php');
$dbc = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

// Zapytanie, które pobiera liczbę wszystkich wierszy.
$query = build_query($user_search, $sort);
$result = mysqli_query($dbc, $query);
$total = mysqli_num_rows($result);
$num_pages = ceil($total / $results_per_page);

// Drugie zapytanie, które pobiera tylko podzbiór wyników.
$query = $query . " LIMIT $skip, $results_per_page";
$result = mysqli_query($dbc, $query);
while ($row = mysqli_fetch_array($result)) {
  echo '<tr class="results">';
  echo '<td valign="top" width="20%">' . $row['title'] . '</td>';
  echo '<td valign="top" width="50%">' . substr($row['description'], 0, 100) .
    '...</td>';
  echo '<td valign="top" width="10%">' . $row['state'] . '</td>';
  echo '<td valign="top" width="20%">' . substr($row['date_posted'], 0, 10) . '</td>';
  echo '</tr>';
}
echo '</table>';

// Generowanie odnośników nawigacyjnych, jeśli wyniki zajmują więcej niż jedną stronę.
if ($num_pages > 1) {
  echo generate_page_links($user_search, $sort, $cur_page, $num_pages);
}

mysqli_close($dbc);
?>
```

Wywołanie funkcji build_query(), która tworzy zapytanie SQL wyszukujące oferty.

To klauzula LIMIT, którą utworzyliśmy, aby pobrać tylko podzbiór znalezionych ofert.

A to kod, który przy użyciu funkcji substr() skraca opis stanowiska i datę dodania oferty.

Wywołanie funkcji generate_page_links() w celu wygenerowania odnośników do stron z wynikami i wyświetlenia ich.

Należy zamknąć połączenie z bazą, aby „posprzątać" po działaniu skryptu.

Nie istnieją głupie pytania

P: Czy naprawdę musimy przekazywać do funkcji `generate_page_links()` szukany tekst, metodę sortowania i dane potrzebne przy stronicowaniu?

O: Tak. Wynika to z tego, że dobrze zaprojektowane funkcje nie powinny manipulować danymi, które znajdują się poza ich kodem. Dlatego w funkcji należy korzystać wyłącznie z danych przekazanych jako argumenty i modyfikować tylko zwracane informacje.

P: A co z wyświetlaniem danych? Dlaczego funkcja `generate_page_links()` nie pokazuje odnośników?

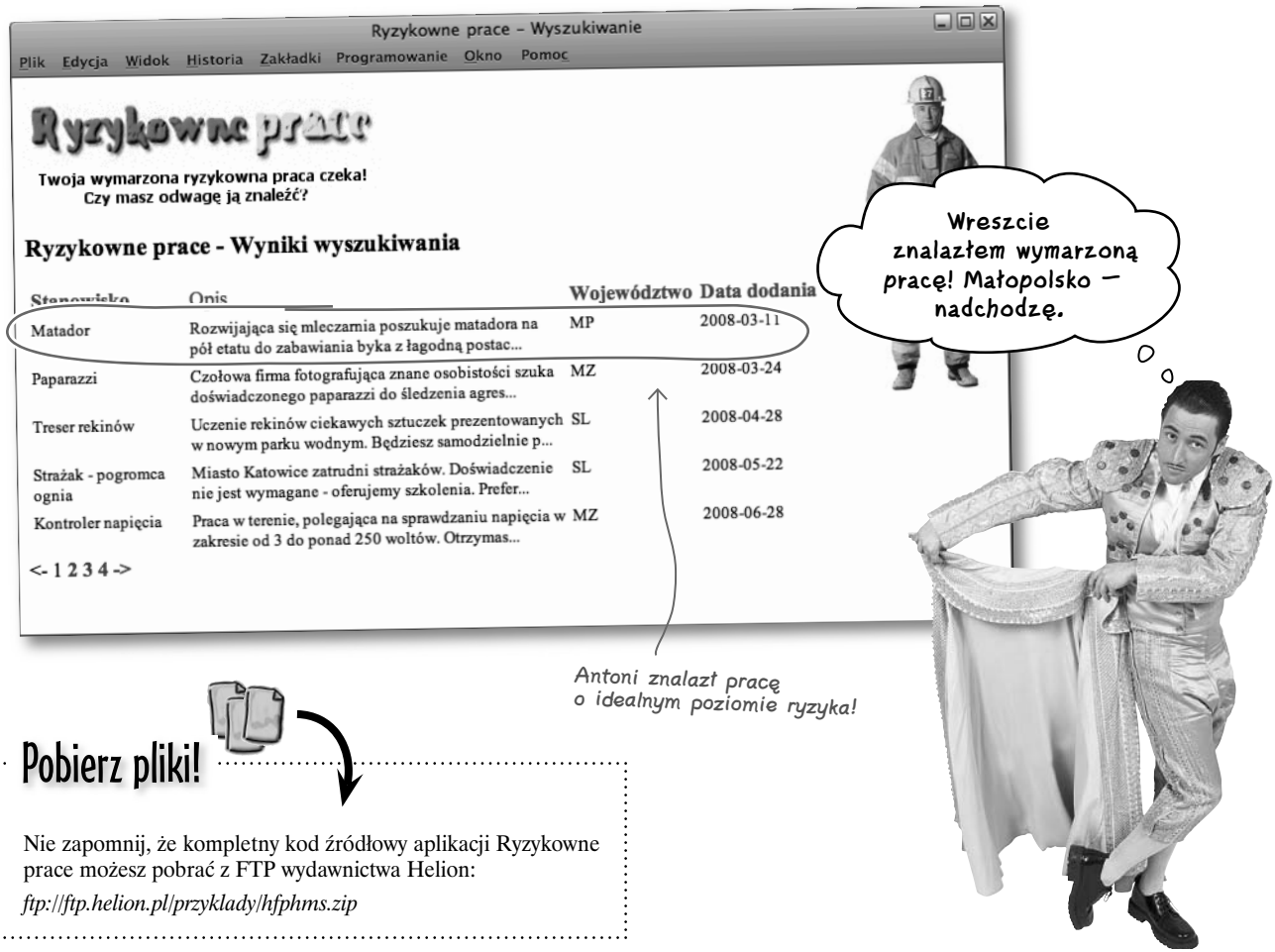
O: Przyczyna jest ta sama. Przekazanie danych do przeglądarki przez funkcję powoduje wprowadzenie zmian poza jej kodem. Dużo trudniej jest diagnozować i konserwować funkcję, jeśli nie wiadomo, które dane modyfikuje. Rozwiązanie polega na zwracaniu danych wygenerowanych w funkcji i używaniu tych informacji **poza nią**.

Jazda próbna

Dokończ skrypt do wyszukiwania ofert w serwisie Ryzykowne prace.

Dodaj do skryptu *search.php* funkcję `generate_page_links()`. Nie zapomnij, że kod powinien wywoływać ją po sprawdzeniu, czy zapytanie zwróciło więcej niż jedną stronę wyników. Trzeba też utworzyć i zainicjować zmienne, które posłużą jako argumenty tej funkcji, a także zaktualizować zapytanie za pomocą klauzuli `LIMIT`, co pozwoli pobrać dla każdej strony właściwy podzbiór wyników.

Po ukończeniu tych zadań prześlij nowy skrypt *search.php* na serwer WWW i otwórz w przeglądarce stronę *search.html*. Uruchom wyszukiwanie kilka razy i wpisz tekst, który zwróci wiele wyników, aby wypróbować mechanizm stronicowania. Najwięcej ofert zobaczysz po wysłaniu pustego formularza wyszukiwania.



Pobierz pliki!

Nie zapomnij, że kompletny kod źródłowy aplikacji Ryzykowne prace możesz pobrać z FTP wydawnictwa Helion:

ftp://ftp.helion.pl/przyklady/hfphms.zip

Niezbędnik programisty PHP i MySQL

Skrypt do wyszukiwania ofert w serwisie Ryzykowne prace wymagał zastosowania kilku nowych technik z języków PHP i MySQL. Przypomnijmy najważniejsze z nich.

LIKE

Użyj słowa LIKE do znalezienia za pomocą zapytania SQL danych bez konieczności ich dokładnego dopasowania. Przed i po szukanym tekście można dodać symbol %, aby określić, że wokół wyrażenia mogą znajdować się inne znaki.

explode(), implode()

Funkcja explode() języka PHP rozbija łańcuch na tablicę podłańcuchów rozdzielonych specyficznym ogranicznikiem (na przykład odstępem lub przecinkiem). Funkcja implode() działa odwrotnie — tworzy jeden łańcuch na podstawie tablicy podłańcuchów i wstawia między nie separator.

substr()

Ta funkcja języka PHP zwraca fragment łańcucha na podstawie podanych argumentów. Można pobrać początek tekstu, jego koniec lub znaki z jego środka.

str_replace()

Wywołaj tę funkcję PHP, aby znaleźć i zastąpić pojedyncze znaki lub ich sekwencje innym tekstem.

Funkcja niestandardowa

Jest to fragment kodu PHP w formie nazwanej jednostki, którą można wywołać wielokrotnie. Celem stosowania funkcji jest wyodrębnienie części skryptu, która wykonuje określone zadanie. Pozwala to wygodnie korzystać z danego fragmentu i uniknąć powtarzania kodu.

switch-case

Ta struktura języka PHP służy do podejmowania decyzji i pozwala wykonać jeden z wielu bloków kodu w zależność od wartości jednej zmiennej. Często grupę zagnieżdżonych instrukcji if-else można przekształcić na bardziej wydajną instrukcję switch.

LIMIT

Klauzula LIMIT pozwala precyzyjnie określić liczbę wierszy zwracanych przez zapytanie SQL. Ponadto można jej użyć do pominięcia wierszy w zbiorze wyników, co umożliwia wyodrębnienie ich podzbioru.