Przeszukiwanie katalogu

"Czego szukasz?" To pytanie często zadawane podczas odwiedzania sklepu detalicznego. Oferowanie pomocy w znalezieniu produktów, których szukają klienci, może przynieść firmie znaczne zyski i ta zasada dotyczy również sklepów internetowych. W tej części dodamy funkcję wyszukiwania produktów do naszego TShirtShop, która pomoże odwiedzającym znaleźć produkty, których szukają. Zobaczysz, jak łatwo jest dodać tę funkcję do TShirtShop, integrując nowe komponenty z istniejącą architekturą. W tej Częsci

• Przeanalizujes różne sposoby przeszukiwania katalogu produktów.

• Utworzysz niezbędne struktury danych MySQL, które wspierają wyszukiwanie produktów.

• Napiszesz dane i warstwy biznesowe używane do implementacji funkcji wyszukiwania.

• Zbudujesz interfejs użytkownika dla funkcji wyszukiwania w katalogu przy użyciu szablonów składowych Smarty.

Wybór sposobu przeszukiwania katalogu

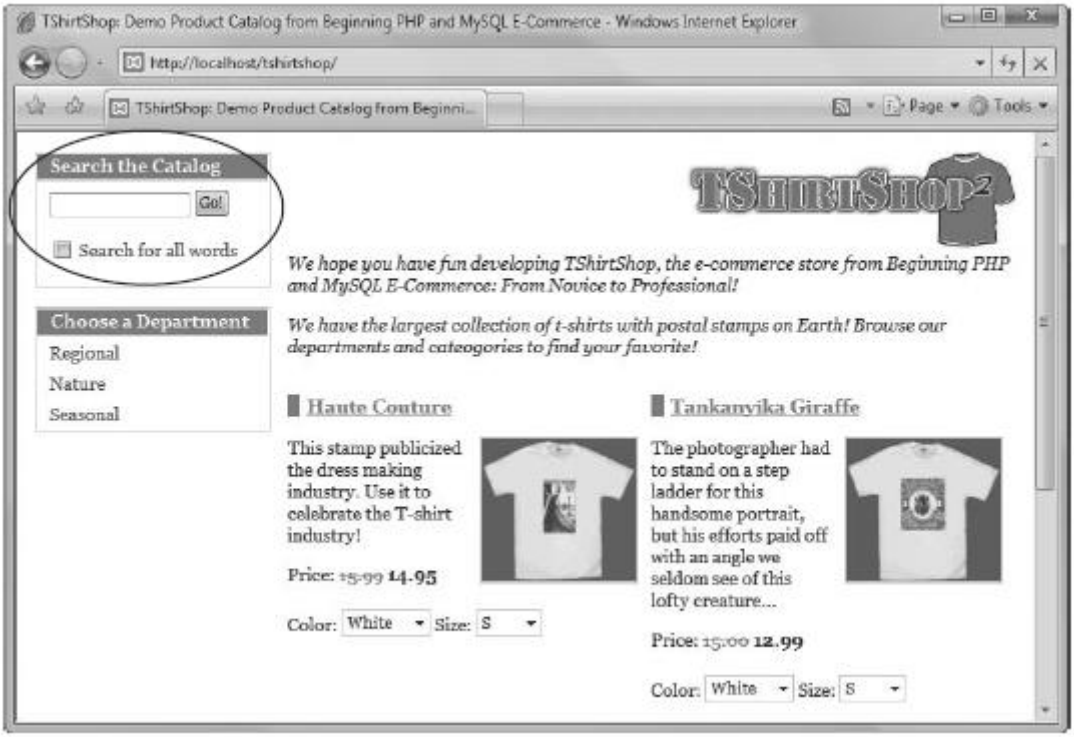
Jak zawsze, jest kilka rzeczy, o których musimy pomyśleć przed rozpoczęciem kodowania. Projektując każdą nową funkcję, zaczynamy od jej analizy z perspektywy użytkownika końcowego. W wizualnej części funkcji wyszukiwania w katalogu użyjemy pola tekstowego, w którym odwiedzający może wpisać jedno lub więcej słów do wyszukania w nazwach i opisach produktów. Tekst wprowadzony przez odwiedzającego można wyszukiwać na kilka sposobów:

* Wyszukiwanie dokładne: jeśli odwiedzający wprowadzi ciąg wyszukiwania składający się z więcej niż jednego słowa, zostaną one wyszukane w katalogu w takim stanie, w jakim są, bez rozdzielania słów i wyszukiwania ich osobno.

* Wyszukiwanie ze wszystkimi słowami: ciąg wyszukiwania wprowadzony przez odwiedzającego jest podzielony na pojedyncze słowa, powodując wyszukiwanie każdego produktu zawierającego wszystkie słowa wprowadzone przez odwiedzającego. Jest to podobne do wyszukiwania z dopasowaniem dokładnym, ponieważ nadal wyszukuje wszystkie wprowadzone słowa, ale w tym przypadku kolejność słów nie jest ważna.

* Wyszukiwanie dowolnych słów: ten rodzaj wyszukiwania zwraca produkty, które zawierają co najmniej jedno słowo z ciągu wyszukiwania.

Ta prosta klasyfikacja nie jest w żaden sposób kompletna. Wyszukiwarka może być tak złożona, jak ta oferowana przez nowoczesne wyszukiwarki internetowe, które zapewniają wiele opcji i funkcji oraz pokazują uporządkowaną listę wyników, lub może być tak prosta, jak przeszukiwanie bazy danych w celu znalezienia dokładnego ciągu podanego przez odwiedzającego. TShirtShop obsługuje tryby wyszukiwania ze wszystkimi słowami i ze wszystkimi słowami. Nie uwzględniamy wyszukiwania z dokładnym dopasowaniem, ponieważ nie jest ono przydatne w przypadku naszych witryn internetowych. Ta decyzja prowadzi do wizualnego projektu funkcji wyszukiwania.



Zgodnie z oczekiwaniami znajduje się tam pole tekstowe wraz z polem wyboru, które pozwala odwiedzającemu wybrać między wyszukiwaniem wszystkich słów a wyszukiwaniem według dowolnych słów. Musisz także zdecydować, jak mają być wyświetlane wyniki wyszukiwania. Jak powinna wyglądać strona wyników wyszukiwania? W końcu chcesz wyświetlić listę produktów spełniających kryteria wyszukiwania. Najprostszym rozwiązaniem wyświetlania wyników wyszukiwania byłoby ponowne użycie złożonego ze składowych szablonu products_list, który zbudowałeś w poprzednim rozdziale. Przykładowa strona wyszukiwania będzie wyglądać jak pokazana na rysunku

Powyższy rysunek pokazuje również adresy URL używane do stron wyników wyszukiwania. Jest to bardziej optymalizacja dla użytkowników niż optymalizacja pod kątem wyszukiwarek, ponieważ ograniczymy wyszukiwarkom przeglądanie stron wyników wyszukiwania, aby uniknąć problemów z duplikacją treści. Jednak te adresy URL mogą być łatwo dodawane do zakładek przez odwiedzających i można je łatwo zhakować (odwiedzający może ręcznie edytować adres URL w pasku adresu) – oba szczegóły sprawiają, że przeglądanie witryny na żywo przez odwiedzającego jest przyjemniejsze. Ostatnim szczegółem, jaki można zauważyć na rysunku , jest to, że witryna wykorzystuje stronicowanie. Jeśli jest dużo wyników wyszukiwania, zaprezentujesz tylko stałą (ale konfigurowalną) liczbę produktów na stronie i umożliwisz odwiedzającemu przeglądanie stron za pomocą linków nawigacyjnych. Zacznijmy od implementacji funkcjonalności, jak zwykle, od warstwy danych.

Uczenie bazy danych do samodzielnego wyszukiwania Masz dwie główne opcje implementacji wyszukiwania w bazie danych:

• Zaimplementuj wyszukiwanie za pomocą funkcji WHERE i LIKE.

• Wyszukiwanie przy użyciu funkcji wyszukiwania pełnotekstowego w MySQL.

Przeanalizujmy te opcje.

Wyszukiwanie za pomocą opcji WHERE i LIKE

Proste rozwiązanie, często używane do implementacji wyszukiwania, polega na użyciu LIKE w klauzuli WHERE instrukcji SELECT. Rzućmy okiem na prosty przykład, który zwróci produkty, które mają gdzieś w opisie słowo „kwiat”:

SELECT nazwa FROM produkt WHERE opis LIKE '%flower%'

Operator LIKE dopasowuje części ciągów i procent symbol wieloznaczny (%) jest używany do określenia dowolnego ciągu składającego się z zera lub większej liczby znaków. Dlatego w poprzednim przykładzie wzorzec %flower% pasuje do wszystkich rekordów, których kolumna opisu zawiera gdzieś słowo „kwiat". W tym wyszukiwaniu nie jest rozróżniana wielkość liter. Jeśli chcesz pobrać wszystkie produkty zawierające słowo „kwiat" gdzieś w nazwie lub opisie produktu, zapytanie będzie wyglądać tak:

```
SELECT name

FROM product

WHERE description LIKE '%flower%' OR name LIKE '%flower%';
```

Ta metoda wyszukiwania ma tę wielką zaletę, że działa na każdym typie tabel MySQL (takich jak typ tabeli InnoDB), ale ma trzy ważne wady:

Szybkość: Ponieważ musimy szukać tekstu gdzieś wewnątrz pól opisu i nazwy, cała baza danych musi być przeszukiwana dla każdego zapytania. Nazywa się to skanowaniem całej tabeli, ponieważ silnik bazy danych nie może używać żadnych zwykłych indeksów, aby przyspieszyć proces wyszukiwania wyników. Może to znacznie spowolnić ogólną wydajność, zwłaszcza jeśli w bazie danych znajduje się duża liczba produktów.

Jakość wyników wyszukiwania: ta metoda nie ułatwia wdrażania różnych zaawansowanych funkcji, takich jak zwracanie pasujących produktów posortowanych według trafności wyszukiwania.

Zaawansowane funkcje wyszukiwania: ta metoda nie pozwala odwiedzającym na przeprowadzanie wyszukiwań z użyciem operatorów logicznych (AND, OR), odmienionych form słów (takich jak liczba mnoga i różne czasy czasowników) lub słów znajdujących się blisko siebie.

Jak więc możesz przeprowadzać lepsze wyszukiwania, które implementują te funkcje? Jeśli masz dużą bazę danych, która musi być często przeszukiwana, jak możesz przeszukiwać tę bazę danych bez zabijania serwera? Odpowiedzią jest użycie funkcji wyszukiwania pełnotekstowego MySQL.

Wyszukiwanie za pomocą funkcji wyszukiwania pełnotekstowego MySQL

Wyszukiwanie za pomocą LIKE, jak wyjaśniono wcześniej, jest bardzo nieefektywne ze względu na operację skanowania całej tabeli, którą musi wykonać baza danych podczas wyszukiwania słowa. Jeśli szukasz słowa „kwiat" w opisach produktów, każdy opis produktu jest czytany i analizowany. To najgorszy scenariusz, jeśli chodzi o operacje na bazach danych.

Porada : Typowe indeksy tabel stosowane w kolumnach tekstowych (takich jak varchar) poprawiają wydajność wyszukiwań szukających dokładnej wartości lub ciągów rozpoczynających się od określonej litery lub słowa. Dzieje się tak, ponieważ typowy indeks działa poprzez sortowanie ciągów w kolejności alfabetycznej, analizując je od lewej do prawej – tak jak sortowane są na przykład nazwiska w książce telefonicznej. Indeksy te przyspieszają wyszukiwanie, gdy znasz litery (lub znaki), od których zaczyna się ciąg wyszukiwania, ale są one bezużyteczne, gdy szukasz słów znajdujących się w ciągu.

Dobrą wiadomością jest to, że MySQL ma funkcję o nazwie indeksy FULLTEXT, które zostały specjalnie zaprojektowane, aby umożliwić wydajne i wydajne wyszukiwanie tekstowe. Indeksy FULLTEXT są podobne do normalnych indeksów, ale analizują całą zawartość kolumn ciągów (takich jak nazwy i opisy produktów). Indeks FULLTEXT znacznie przyspieszy operacje wyszukiwania określonego słowa (lub zestawu słów) na przykład w opisie produktu. Ten indeks umożliwia wykonywanie takich operacji bez wykonywania skanowania pełnej tabeli, które ma miejsce, gdy używana jest funkcja LIKE.

Wyszukiwanie pełnotekstowe MySQL jest znacznie szybsze i mądrzejsze niż wspomniana wcześniej metoda (za pomocą operatora LIKE). Oto jego główne zalety:

• Wyniki wyszukiwania są uporządkowane według trafności wyszukiwania.

• Małe słowa są ignorowane. Słowa, które nie mają co najmniej czterech znaków, takie jak „i", „tak" itd., są domyślnie usuwane z zapytania.

• Zaawansowane funkcje, takie jak wyszukiwanie pełnotekstowe MySQL, mogą być również wykonywane w trybie logicznym. Ten tryb umożliwia wyszukiwanie słów na podstawie kryteriów ORAZ/LUB, takich jak „+piękny +kwiat", co powoduje wyszukanie wszystkich wierszy zawierających zarówno słowa „piękny", jak i „kwiat".

• Możliwe są szybsze wyszukiwania. Ze względu na użycie specjalnych indeksów wyszukiwania operacja wyszukiwania jest znacznie szybsza niż przy użyciu metody LIKE.

Jak wyjaśniono w części 4, główną wadą funkcji wyszukiwania pełnotekstowego jest to, że działa ona tylko z typem tabeli MyISAM. Alternatywnym typem tabeli, którego możesz użyć, jest InnoDB, który jest bardziej zaawansowany i obsługuje funkcje, takie jak klucze obce, transakcje ACID i inne, ale nie obsługuje funkcji pełnego tekstu.

Uwaga: ACID to akronim opisujący cztery podstawowe właściwości transakcji bazodanowych: niepodzielność, spójność, izolacja i trwałość. W tej książce nie będziemy używać transakcji bazodanowych, ale możesz dowiedzieć się o nich więcej z innych źródeł, takich jak Przewodnik programisty po SQL.

Na kolejnych kilku stronach najpierw utworzysz indeksy PEŁNOTEKSTOWE w swojej bazie danych, a następnie nauczysz się ich używać do przeszukiwania katalogu.

Tworzenie struktur danych umożliwiających wyszukiwanie

W naszym scenariuszu tabela, której użyjemy do wyszukiwania, to produkt, ponieważ właśnie tego będą szukać nasi odwiedzający. Zanim będzie można ją przeszukiwać za pomocą indeksów FULLTEXT, musisz upewnić się, że typ tabeli to MyISAM (powinno to mieć miejsce, jeśli prawidłowo postępowałeś zgodnie z instrukcjami). Jeśli użyłeś innego typu tabeli podczas jej tworzenia, przekonwertuj go teraz, wykonując tę instrukcję SQL po połączeniu się z bazą danych tshirtshop:

```
ALTER TABLE product ENGINE = MYISAM;
```

Aby umożliwić przeszukiwanie tabeli produktów, musimy dodać indeks pełnotekstowy do pary kolumn (nazwa, opis), w następujący sposób:

1. Załaduj phpMyAdmin, wybierz bazę danych tshirtshop z pola Baza danych i kliknij kartę SQL.

2. W formularzu wpisz następujące polecenie, które dodaje nowy indeks pełnotekstowy o nazwie idx_ft_nazwa_produktu_opis:

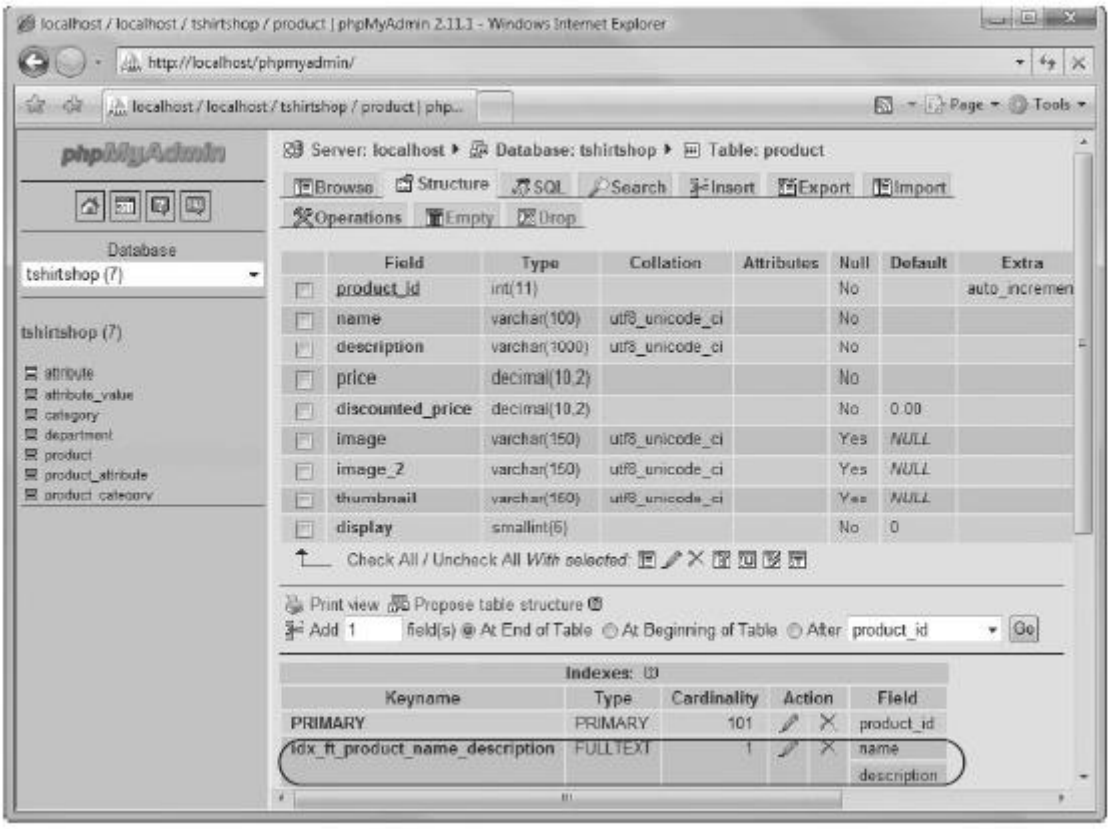
```
-- Create full-text search index

CREATE FULLTEXT INDEX `idx_ft_product_name_description`

ON `product` (`name`, `description`);
```

Po kliknięciu przycisku Go powinieneś zostać poinformowany, że polecenie zostało wykonane pomyślnie. Ponieważ chcemy, aby TShirtShop umożliwiał odwiedzającym wyszukiwanie produktów zawierających określone słowa w ich nazwach lub opisach, utworzyliśmy indeks pełnotekstowy na

parze pól (nazwa, opis) w tabeli produktów (jest to coś innego niż posiadanie dwóch pełnych -indeksy tekstowe, jeden na nazwę i jeden na opis). Utworzenie tego indeksu pełnotekstowego umożliwia wyszukiwanie pełnotekstowe w indeksowanych polach. Aby phpMyAdmin potwierdził istnienie nowego indeksu pełnotekstowego, kliknij kartę Struktura i kliknij ikonę Struktura dla tabeli produktów. W nowym oknie, w sekcji Indeksy , zobaczysz teraz nowy indeks typu FULLTEXT w kolumnach name i description.



Wskazówka: warto zauważyć, że phpMyAdmin potwierdza, że mamy pojedynczy indeks FULLTEXT w dwóch kolumnach tabeli, a nie dwa oddzielne indeksy FULLTEXT.

Uczenie MySQL do wyszukiwania dowolnych słów

Ogólna składnia MySQL do wykonywania wyszukiwania pełnotekstowego wygląda następująco:

SELECT <column_list>

FROM <table>

WHERE MATCH <column or list of columns> AGAINST <search criteria>

Kolumna lub lista kolumn, w których przeprowadzasz wyszukiwanie, musi być indeksowana pełnotekstowo. Jeśli istnieje lista kolumn, musi istnieć indeks pełnotekstowy odnoszący się do tej grupy kolumn, tak jak nasz indeks idx_ft_nazwa_produktu_opis dotyczy zarówno nazwy, jak i opisu. W jaki sposób można użyć tego indeksu pełnotekstowego do wyszukiwania dowolnych słów w swoich produktach? Załóżmy, że chcesz wyszukać słowa „piękny” i/lub „kwiat” w ich parze (nazwa, opis). Osiąga to następująca instrukcja SQL:

SELECT name, description FROM product

WHERE MATCH (name, description) AGAINST ("beautiful flower");

Wykonanie tego zapytania, gdy baza danych tshirtshop zawiera przykładowe dane, zwróciłoby 33 rekordy produktów. Przeprowadzając takie wyszukiwania, zwykle chcesz pobrać wyniki posortowane w kolejności malejącej według trafności. Można to zrobić za pomocą klauzuli ORDER BY i podając jako argument regułę MATCH. Zawsze pamiętaj o korzystaniu z opcji DESC, aby najtrafniejszy wynik znalazł się na górze.

SELECT name, description FROM product

WHERE MATCH (name, description) AGAINST ("beautiful flower")

ORDER BY MATCH (name, description) AGAINST ("beautiful flower") DESC

Zapytanie ma 33 wyniki, korzystając z naszych przykładowych danych, pokazanych częściowo na rysunku.

name	description
Uruguay Flower	The Indian Queen Anahi was the ugliest woman ever seen. But instead of living a slave when captured by the Conquistadores, she immolated herself in a fire and was reborn the most beautiful of flowers: the ceibo, national flower of Uruguay. Of course, you won't need to burn to wear this T-shirt, but you may cause some pretty hot glances to be thrown your way!
Poland Flower	A beautiful and sunny T-shirt for both spring and summer!
Costa Rica Flower	This national flower of Costa Rica is one of our most beloved flower T-shirts (you can see one on Jill, above). You will surely stand out in this T-shirt!
Afghan Flower	This beautiful image was issued to celebrate National Teachers Day. Perhaps you know a teacher who would love this T-shirt?
Albania Flower	Well, these crab apples started out as flowers, so that's close enough for us! They still make for a uniquely beautiful T-shirt.
Romania Flower	Also known as the spring pheasant's eye, this flower belongs on your T-shirt this summer to help you catch a few eyes.
Israel Flower	This plant is native to the rocky and sandy regions of the western United States, so when you come across one, it really stands out. And so will you when you put on this beautiful T-shirt!
Bulgarian Flower	For your interest (and to impress your friends), this beautiful stamp was issued to honor the George Dimitrov state printing works. You'll need to know this when you wear the T-shirt.
Congo Flower	The Congo is not at a loss for beautiful flowers, and we've picked a few of them for your T-shirts.
Austria Flower	Have you ever had nasturtiums on your salad? Try it--they're almost as good as having them on your T-shirt!
Ghana Flower	This is one of the first gingers to bloom in the spring--just like you when you wear this T-shirt!

Wyniki reprezentują rekordy uporządkowane na podstawie wartości trafności wyszukiwania, przy czym najtrafniejsze wyniki są wyświetlane jako pierwsze (lista na rysunku 8-4 została wygenerowana przez wykonanie zapytania i kliknięcie linku „Wydrukuj widok (z pełnymi tekstami)”, który pojawia się pod adresem na dole strony phpMyAdmin). Na przykład produkty zawierające zarówno słowa „piękny”, jak i „kwiat” (lub zawierające więcej ich wystąpień) pojawiają się wyżej na liście niż produkty zawierające tylko jedno z tych słów.

DOSTRAJANIE WYSZUKIWANIA PEŁNOTEKSTOWEGO MYSQL

Domyślnie słowa, które nie mają co najmniej czterech znaków, nie są indeksowane (w rezultacie nigdy nie są uwzględniane w żadnych wyszukiwaniach), ale możesz zmienić to zachowanie, jeśli chcesz. Minimalna długość słów, które mają być uwzględnione w indeksach FULLTEXT, jest określana przez zmienną serwerową ft_min_word_len. Na przykład, jeśli chcesz, aby trzyznakowe słowa były dostępne do przeszukiwania, wystarczy ustawić zmienną ft_min_word_len w pliku konfiguracyjnym serwera MySQL w następujący sposób:

[mysqld]

ft_min_word_len=3

Plik konfiguracyjny, w którym należy przechowywać to ustawienie, to zazwyczaj /opt/lampp/etc/my.cnf w systemie Unix i C:\xampp\mysql\bin\my.cnf lub C:\Windows\php.ini w systemie Windows.

Po zmianie wartości ft_min_word_len musisz zrestartować serwer MySQL. Po ponownym uruchomieniu serwera możesz zapytać serwer MySQL o wartości zmiennych, aby upewnić się, że zmiany odniosły skutek, używając zapytania takiego jak SHOW VARIABLES LIKE 'ft_%'; Po zmianie wartości ft_min_word_len musisz również przebudować indeksy FULLTEXT. Możesz to zrobić, upuszczając i ponownie tworząc indeks lub używając REPAIR TABLE w następujący sposób:

REPAIR TABLE product QUICK;

Pamiętaj, że musisz NAPRAWIĆ tylko tabele, w których masz indeksy FULLTEXT. Jeśli z jakiegoś powodu wolisz odtworzyć indeks (ale zalecamy użycie REPAIR TABLE), możesz to zrobić w następujący sposób:

ALTER TABLE product

DROP INDEX idx_ft_product_name_description;

CREATE FULLTEXT INDEX idx_ft_product_name_description

ON product (name, description);

Uczenie MySQL, jak wyszukiwać za pomocą wszystkich słów

Widzieliśmy już, że wyszukiwanie dowolnych słów zwróci wszystkie produkty, które w nazwie lub opisie zawierają słowo „kwiat” lub „piękne” (lub oba słowa). Z drugiej strony wyniki wyszukiwania zawierającego wszystkie słowa powinny zawierać tylko produkty zawierające wszystkie słowa, których szukasz (w tym przypadku „piękny” i „kwiat”). W przypadku wyszukiwania zawierającego wszystkie słowa należy użyć trybu Boolean funkcji wyszukiwania pełnotekstowego, który pozwala na użycie logiki AND/OR w kryteriach wyszukiwania. Nowe zapytanie wyglądałoby tak:

SELECT name, description FROM product

WHERE MATCH (name, description) AGAINST ("+beautiful +flower" IN BOOLEAN MODE)

ORDER BY MATCH (name, description) AGAINST ("+beautiful +flower" IN BOOLEAN MODE)

DESC;

Sortowanie w kolejności malejącej według wartości dopasowania nie jest wymagane, ale jest bardzo pożądane, ponieważ zwykle chcesz otrzymywać wyniki wyszukiwania w kolejności malejącej według trafności. Wiodący znak plus oznacza wymagane słowa, więc należy go dodać dla każdego słowa w wyszukiwaniu wszystkich słów. W porównaniu z wyszukiwaniem przy użyciu dowolnych słów, to zapytanie zwraca tylko siedem produktów, co widać na rysunku .

name	description
Afghan Flower	This beautiful image was issued to celebrate National Teachers Day. Perhaps you know a teacher who would love this T-shirt?
Albania Flower	Well, these crab apples started out as flowers, so that's close enough for us! They still make for a uniquely beautiful T-shirt.
Bulgarian Flower	For your interest (and to impress your friends), this beautiful stamp was issued to honor the George Dimitrov state printing works. You'll need to know this when you wear the T-shirt.
Congo Flower	The Congo is not at a loss for beautiful flowers, and we've picked a few of them for your T-shirts.
Israel Flower	This plant is native to the rocky and sandy regions of the western United States, so when you come across one, it really stands out. And so will you when you put on this beautiful T-shirt!
Poland Flower	A beautiful and sunny T-shirt for both spring and summer!
Uruguay Flower	The Indian Queen Anahi was the ugliest woman ever seen. But instead of living a slave when captured by the Conquistadores, she immolated herself in a fire and was reborn the most beautiful of flowers: the ceibo, national flower of Uruguay. Of course, you won't need to burn to wear this T-shirt, but you may cause some pretty hot glances to be thrown your way!

WIĘCEJ DOSTRAJANIA PEŁNOTEKSTOWEGO: ROZSZERZENIE ZAPYTANIA

Wyszukiwanie pełnotekstowe z funkcją rozszerzania zapytań zostało wprowadzone w MySQL 4.1.1 i umożliwia MySQL znajdowanie produktów, które pasują nie tylko do wyszukiwanych słów, ale także do słów pokrewnych. Podczas wyszukiwania z rozszerzeniem zapytań MySQL przeprowadza wyszukiwanie dwukrotnie za kulisami. Najpierw znajduje słowa, które są najbardziej odpowiednie dla tych, których szukasz. Następnie te słowa są dołączane do początkowego ciągu zapytania i wyszukiwanie jest wykonywane ponownie. Ta metoda zwiększa liczbę wyników wyszukiwania, ale także zwiększa szansę na uzyskanie nietrafnych produktów. Aby włączyć rozszerzanie zapytań, musisz dodać Z ROZSZERZENIEM ZAPYTANIA do kryteriów wyszukiwania, jak pokazano w następującym fragmencie kodu:

SELECT name, description FROM product

WHERE MATCH (name, description) AGAINST ("flower" WITH QUERY EXPANSION)

ORDER BY MATCH (name, description) AGAINST ("flower" WITH QUERY EXPANSION) DESC;

Wyszukiwanie hasła „kwiat” z rozszerzeniem zapytania zwraca 75 wyników. Wykonując to samo wyszukiwanie bez rozszerzania zapytania, otrzymujesz tylko 15 wyników. Kod warstwy danych przedstawiony na następnych stronach nie korzysta z rozszerzania zapytań, ale możesz go bardzo łatwo dodać, jeśli chcesz.

Pisanie procedur składowanych dla funkcji wyszukiwania

Jesteś teraz gotowy do wdrożenia funkcji wyszukiwania na swojej stronie internetowej.

Ćwiczenie: Pisanie kodu przeszukiwania bazy danych

1. Użyj phpMyAdmin, aby wykonać następujący kod, który utworzy procedurę składowaną catalog_count_search_result w bazie danych sklepu tshirtshop. Nie zapomnij ustawić $$ jako ogranicznika przed wykonaniem kodu.

-- Create catalog_count_search_result stored procedure

CREATE PROCEDURE catalog_count_search_result(

```
IN inSearchString TEXT, IN inAllWords VARCHAR(3))

BEGIN

IF inAllWords = "on" THEN

PREPARE statement FROM

"SELECT count(*)

FROM product

WHERE MATCH (name, description) AGAINST (? IN BOOLEAN MODE)";

ELSE

PREPARE statement FROM

"SELECT count(*)

FROM product

WHERE MATCH (name, description) AGAINST (?)";

END IF;

SET @p1 = inSearchString;

EXECUTE statement USING @p1;

END$$
```

2. Teraz wykonaj tę samą procedurę, aby wykonać ten kod, który tworzy zapisaną procedurę catalog_search w bazie danych sklepu tshirtshop:

```
-- Create catalog_search stored procedure

CREATE PROCEDURE catalog_search(

IN inSearchString TEXT, IN inAllWords VARCHAR(3),

IN inShortProductDescriptionLength INT,

IN inProductsPerPage INT, IN inStartItem INT)

BEGIN

IF inAllWords = "on" THEN

PREPARE statement FROM

"SELECT product_id, name,

IF(LENGTH(description) <= ?,

description,

CONCAT(LEFT(description, ?),

'...')) AS description,
```

```
price, discounted_price, thumbnail
FROM product
WHERE MATCH (name, description)
AGAINST (? IN BOOLEAN MODE)
ORDER BY MATCH (name, description)
AGAINST (? IN BOOLEAN MODE) DESC
LIMIT ?, ?";
ELSE
PREPARE statement FROM
"SELECT product_id, name,
IF(LENGTH(description) <= ?,
description,
CONCAT(LEFT(description, ?),
'...')) AS description,
price, discounted_price, thumbnail
FROM product
WHERE MATCH (name, description) AGAINST (?)
ORDER BY MATCH (name, description) AGAINST (?) DESC
LIMIT ?, ?";
END IF;
SET @p1 = inShortProductDescriptionLength;
SET @p2 = inSearchString;
SET @p3 = inStartItem;
SET @p4 = inProductsPerPage;
EXECUTE statement USING @p1, @p1, @p2, @p2, @p3, @p4;
END$$
```

Jak to działa: funkcja przeszukiwania katalogów

W tym ćwiczeniu utworzono funkcję bazy danych obsługującą logikę warstwy biznesowej wyszukiwania produktów, która składa się z dwóch procedur składowanych:

• Catalog_count_search_result: ta procedura składowana zlicza liczbę wyników wyszukiwania. Jest to wymagane, aby warstwa prezentacji wiedziała, ile stron wyników wyszukiwania ma wyświetlić.

• Catalog_search: ta procedura składowana wykonuje rzeczywiste wyszukiwanie produktów.

Chociaż na początku mogą wyglądać onieśmielająco, głównie ze względu na swój rozmiar, te przechowywane procedury po prostu dobrze wykorzystują teorię wyszukiwania pełnotekstowego, którą poznałeś wcześniej w tym rozdziale. Dodatkowo catalog_search implementuje techniki stronicowania, które są również obecne w innych procedurach zwracających listy produktów.

Wdrażanie poziomu biznesowego

Warstwa biznesowa funkcji wyszukiwania składa się z jednej metody o nazwie Search, która wywołuje dwie zaimplementowane wcześniej procedury składowane w celu pobrania listy wyszukiwanych produktów. Najpierw zaimplementujmy tę metodę, a potem omówimy, jak to działa.

Ćwiczenie: Wdrażanie warstwy biznesowej

1. Funkcja wyszukiwania pełnotekstowego MySQL ignoruje słowa, które są krótsze niż określona długość. Chcemy poinformować odwiedzającego, jakie słowa zostały użyte do wyszukiwania, a które zostały zignorowane. Aby wesprzeć tę funkcję, najpierw musimy znaleźć długość, która jest już określona w naszej bazie danych, odpytując zmienną MySQL ft_min_word_len. Użyj phpMyAdmin, aby wykonać następującą instrukcję SQL w bazie danych tshirtshop:

SHOW VARIABLES LIKE 'ft_min_word_len';

Jeśli nie zmieniłeś wartości, ft_min_word_len powinno wynosić 4

Variable_name	Value
ft_min_word_len	4

2. Zapiszmy teraz znalezioną właśnie wartość ft_min_word_len w pliku konfiguracyjnym, dzięki czemu będzie on łatwo dostępny z kodu aplikacji. Otwórz include/config.php i dodaj następujący kod:

```
/* Minimum word length for searches; this constant must be kept in sync

with the ft_min_word_len MySQL variable */

define('FT_MIN_WORD_LEN', 4);
```

3. Otwórz business/catalog.php i dodaj kod metody Search():

```
// Search the catalog

public static function Search($searchString, $allWords,

$pageNo, &$rHowManyPages)

{

//The search result will be an array of this form

$search_result = array ('accepted_words' => array (),

'ignored_words' => array (),

'products' => array ());

// Return void if the search string is void

if (empty ($searchString))
```

```
return $search_result;
// Search string delimiters
$delimiters = ',.; ';
/* On the first call to strtok you supply the whole
search string and the list of delimiters.
It returns the first word of the string */
$word = strtok($searchString, $delimiters);
// Parse the string word by word until there are no more words
while ($word)
{
// Short words are added to the ignored_words list from $search_result
if (strlen($word) < FT_MIN_WORD_LEN)
$search_result['ignored_words'][] = $word;
else
$search_result['accepted_words'][] = $word;
// Get the next word of the search string
$word = strtok($delimiters);
}
// If there aren't any accepted words return the $search_result
if (count($search_result['accepted_words']) == 0)
return $search_result;
// Build $search_string from accepted words list
$search_string = '';
// If $allWords is 'on' then we append a ' +' to each word
if (strcmp($allWords, "on") == 0)
$search_string = implode(" +", $search_result['accepted_words']);
else
$search_string = implode(" ", $search_result['accepted_words']);
// Count the number of search results
$sql = 'CALL catalog_count_search_result(:search_string, :all_words)';
$params = array(':search_string' => $search_string,
```

```
':all_words' => $allWords);

// Calculate the number of pages required to display the products

$rHowManyPages = Catalog::HowManyPages($sql, $params);

// Calculate the start item

$start_item = ($pageNo - 1) * PRODUCTS_PER_PAGE;

// Retrieve the list of matching products

$sql = 'CALL catalog_search(:search_string, :all_words,

:short_product_description_length,

:products_per_page, :start_item)';

// Build the parameters array

$params = array (':search_string' => $search_string,

':all_words' => $allWords,

':short_product_description_length' =>

SHORT_PRODUCT_DESCRIPTION_LENGTH,

':products_per_page' => PRODUCTS_PER_PAGE,

':start_item' => $start_item);

// Execute the query

$search_result['products'] = DatabaseHandler::GetAll($sql, $params);

// Return the results

return $search_result;

}
```

Jak to działa: metoda wyszukiwania na poziomie biznesowym

Metoda Search() usuwa słowa, które są krótsze niż długość określona przez ft_min_word_len. Minimalna długość słowa jest przechowywana w stałej FT_MIN_WORD_LEN. T_MIN_WORD_LEN powinien być zsynchronizowany ze zmienną serwera MySQL ft_min_word_len. Powodem, dla którego warto znać tę wartość w naszym kodzie PHP, jest to, że chcemy powiedzieć odwiedzającemu, które słowa zostały usunięte podczas wyszukiwania. Najpierw dowiadujemy się, które słowa są usuwane, porównując ich długość z FT_MIN_WORD_LEN. W ten sposób dzielimy wyszukiwane słowa na dwa zestawy — akceptowane i ignorowane — i umieszczamy je w tablicy asocjacyjnej, w której również będziemy przechowywać wyniki wyszukiwania i zwracać je jako wynik metody. Metoda Search() warstwy biznesowej jest wywoływana z warstwy prezentacji z następującymi parametrami (zauważ, że wszystkie z wyjątkiem pierwszego są takie same jak parametry metody warstwy danych Search()):

• $searchString zawiera ciąg wyszukiwania wprowadzony przez odwiedzającego.

• $allWords jest „włączone” dla wyszukiwań zawierających wszystkie słowa.

• $pageNo reprezentuje stronę żądanych produktów.

• $rHowManyPages reprezentuje całkowitą liczbę stron wyników.

Funkcja rozpoczyna się od zadeklarowania tablicy asocjacyjnej o nazwie $search_result, w której będą przechowywane wyniki wyszukiwania produktów.

```
// Search the catalog

public static function Search($searchString, $allWords,

$pageNo, &$rHowManyPages)

{

//The search result will be an array of this form

$search_result = array ('accepted_words' => array (),

'ignored_words' => array (),

'products' => array ());
```

Jak widać, tablica zawiera trzy inne tablice z samoopisowymi nazwami. Tablica accept_words będzie przechowywać słowa, które będą używane do wyszukiwania, a ignored_words zostanie wypełnione słowami, które zostaną zignorowane podczas wyszukiwania. Warstwa prezentacji użyje tych danych, aby poinformować odwiedzającego, które słowa zostały wyszukane, a które zignorowane. Tablica produktów będzie przechowywać listę produktów pasujących do wyszukiwanego ciągu. Następnie Search() sprawdza, czy wyszukiwany ciąg nie jest pusty. Jeśli tak, zwracana jest pusta tablica $search_result, co oznacza, że nie znaleziono wyników wyszukiwania:

```
// Return void if the search string is void

if (empty ($searchString))

return $search_result;
```

Po upewnieniu się, że wyszukiwany ciąg nie jest pusty, zaczynamy filtrować słowa.Słowa krótsze niż FT_MIN_WORD_LEN są wysyłane do tablicy ignored_words i są ignorowane przez MySQL podczas wyszukiwania. Pozostałe słowa są zapisywane w tablicy accept_words, array; będą używane do wyszukiwania. Ponieważ MySQL nie informuje, które słowa zostały użyte do wyszukiwania, a które zostały zignorowane, musimy sprawdzić ignorowane i akceptowane słowa w warstwie biznesowej. Dzieląc wyszukiwany ciąg na pojedyncze słowa, bierzemy pod uwagę, że możliwe separatory słów to przecinek (,), kropka (.), średnik (;) i spacja. Każdy inny znak wpisany przez użytkownika w polu wyszukiwania będzie uważany za część słowa. Podświetlony kod, który następuje, to ten, który dzieli wyszukiwany ciąg na słowa i zapisuje je w tablicach akceptowanych słów i ignorowanych słów:

```
// Search string delimiters

$delimiters = ',.; ';

/* On the first call to strtok you supply the whole

search string and the list of delimiters.

It returns the first word of the string */
```

```
$word = strtok($searchString, $delimiters);

// Parse the string word by word until there are no more words

while ($word)

{

// Short words are added to the ignored_words list from $search_result

if (strlen($word) < FT_MIN_WORD_LEN)

$search_result['ignored_words'][] = $word;

else

$search_result['accepted_words'][] = $word;

// Get the next word of the search string

$word = strtok($delimiters);

}
```

Po wykonaniu tej operacji ponownie sprawdzamy, czy nadal mamy słowa do wyszukania. Tym razem sprawdzamy, czy tablica accept_words nie jest pusta. Jeśli tak, to nie mamy czego szukać, więc zwracamy pustą tablicę $search_result:

```
// If there aren't any accepted words return the $search_result

if (count($search_result['accepted_words']) == 0)

return $search_result;
```

Teraz, gdy wiemy, że mamy co najmniej jedno słowo do wyszukania, czas przygotować ciąg wyszukiwania, który wyślemy do bazy danych. Jak wiecie z teorii wyszukiwania pełnotekstowego, podczas wyszukiwania za pomocą wszystkich słów używamy wyszukiwania logicznego i musimy poprzedzić każde wyszukiwane słowo znakiem plus (+). W przypadku wyszukiwania hasła „piękny kwiat” za pomocą wszystkich słów typu logicznego, ciąg wyszukiwania, który wysyłamy do bazy danych, to „+piękny +kwiat”. W przypadku wyszukiwania dowolnych słów po prostu wysyłamy „piękny kwiat”. Poniższy kod tworzy i wypełnia zmienną łańcuchową $search_, która zawiera dokładnie ten ciąg znaków, którego MySQL musi wyszukać:

```
// Build $search_string from accepted words list

$search_string = '';

// If $allWords is 'on' then we append a ' +' to each word

if (strcmp($allWords, "on") == 0)

$search_string = implode(" +", $search_result['accepted_words']);

else $search_string = implode(" ", $search_result['accepted_words']);
```

Poniższy kod w Search() jest typowym kodem, który żąda strony produktów i znasz go z rozdziałów, w których utworzyłeś katalog produktów. Funkcja kończy się zwróceniem $search_result, który zawiera

wyniki wyszukiwania oraz listy akceptowanych i ignorowanych słów, które reprezentują wszystkie dane, które chcesz pokazać odwiedzającemu podczas wyszukiwania:

```
// Execute the query

$search_result['products'] = DatabaseHandler::GetAll($sql, $params);

// Return the results

return $search_result;

}
```

Wdrażanie poziomu prezentacji

Funkcja przeszukiwania katalogu ma dwa oddzielne elementy interfejsu, które należy zaimplementować:

• Składnikowy szablon o nazwie search_box, którego rolą jest zapewnienie środków do wprowadzenia ciągu wyszukiwania dla odwiedzającego

• Podzielony szablon o nazwie search_results, który wyświetla produkty pasujące do kryteriów wyszukiwania

Utworzysz dwa szablony składowe w dwóch oddzielnych ćwiczeniach.

Tworzenie pola wyszukiwania

Postępuj zgodnie z instrukcjami w ćwiczeniu, aby zbudować skomponentowany szablon search_box i zintegrować go z TShirtShop.

Ćwiczenie: Tworzenie szablonu składowego search_box

1. Utwórz nowy plik szablonu o nazwie search_box.tpl w folderze prezentacji/szablonów i dodaj do niego następujący kod:

```
{* search_box.tpl *}

{load_presentation_object filename="search_box" assign="obj"}

{* Start search box *}

<div class="box">

<p class="box-title">Search the Catalog</p>

<form class="search_form" method="post" action="{$obj->mLinkToSearch}">

<p>

<input maxlength="100" id="search_string" name="search_string"

value="{$obj->mSearchString}" size="19" />

<input type="submit" value="Go!" /><br />

</p>

<p>
```

```
<input type="checkbox" id="all_words" name="all_words"
{if $obj->mAllWords == "on"} checked="checked" {/if}/>
Search for all words
</p>
</form>
</div>
{* End search box *}
```

2. Utwórz obiekt prezentacji SearchBox w pliku o nazwie search_box.php w folderze prezentacji z następującym kodem:

```
<?php
// Manages the search box
class SearchBox
{
// Public variables for the smarty template
public $mSearchString = '';
public $mAllWords = 'off';
public $mLinkToSearch;
// Class constructor
public function __construct()
{
$this->mLinkToSearch = Link::ToSearch();
if (isset ($_GET['Search']))
{
$this->mSearchString = trim($_POST['search_string']);
$this->mAllWords = isset ($_POST['all_words']) ?
$_POST['all_words'] : 'off';
// Clean output buffer
ob_clean();
// Redirect 302
header('HTTP/1.1 302 Found');
header('Location: ' .
```

```
Link::ToSearchResults($this->mSearchString, $this->mAllWords));
// Clear the output buffer and stop execution
flush();
ob_flush();
ob_end_clean();
exit();
}
elseif (isset ($_GET['SearchResults']))
{
$this->mSearchString = trim(str_replace('-', ' ', $_GET['SearchString']));
$this->mAllWords = isset ($_GET['AllWords']) ? $_GET['AllWords'] : 'off';
}
if (isset ($_GET['ProductId']) &&
isset ($_SESSION['link_to_continue_shopping']))
{
$continue_shopping =
Link::QueryStringToArray($_SESSION['link_to_continue_shopping']);
if (isset ($continue_shopping['SearchResults']))
{
$this->mSearchString =
trim(str_replace('-', ' ', $continue_shopping['SearchString']));
$this->mAllWords = $continue_shopping['AllWords'];
}
}
}
}
?>
```

3. Otwórz Presentation/link.php i dodaj następujące dwie metody do klasy Link:

```
// Create link to the search page
public static function ToSearch()
{
```

```
return self::Build('index.php?Search');

}

// Create link to a search results page

public static function ToSearchResults($searchString, $allWords,

$page = 1)

{

$link = 'search-results/find';

if (empty($searchString))

$link .= '/';

else

$link .= '-' . self::CleanUrlText($searchString) . '/';

$link .= 'all-words-' . $allWords . '/';

if ($page > 1)

$link .= 'page-' . $page . '/';

return self::Build($link);

}
```

4. W tym samym pliku zmodyfikuj następującą metodę CheckRequest(), tak jak zaznaczono. Zaimplementowałeś tę funkcję w części 7, aby przekierować przychodzące żądania do ich właściwych wersji, ale nie chcemy, aby weryfikowała ona zapytania lub strony wyników wyszukiwania, które nie mają „właściwej” wersji.

```
// Redirects to proper URL if not already there

public static function CheckRequest()

{

$proper_url = '';

if (isset ($_GET['Search']) || isset($_GET['SearchResults']))

{

return ;

}

// Obtain proper URL for category pages

elseif (isset ($_GET['DepartmentId']) && isset ($_GET['CategoryId']))

{

if (isset ($_GET['Page']))
```

5. Dodaj następujące style, potrzebne w szablonie search_box, do styles/tshirtshop.css:

```
div.yui-b div form {
padding: 5px 10px;
}
input, select, textarea {
font-family: "georgia";
font-size: 85%;
}
```

6. Zmodyfikuj plik Presentation/templates/store_front.tpl, aby załadować szablon search_box:

```
...
<div class="yui-b">
{include file="search_box.tpl"}
{include file="departments_list.tpl"}
{include file=$obj->mCategoriesCell}
</div>
...
```

7. Załaduj swój projekt do przeglądarki, a zobaczysz pole wyszukiwania ładnie odpoczywające na swoim miejscu

Jak to działa: skomponowany szablon search_box

W tym momencie Twoje pole wyszukiwania działa. Jednak zanim będziesz mógł w pełni przetestować funkcję wyszukiwania, musisz również zaimplementować stronę wyników wyszukiwania. Zrobisz to w następnym ćwiczeniu; do tego czasu upewnij się, że rozumiesz, jak działa pole wyszukiwania. Szablon search_box.tpl zawiera układ HTML pola — nie ma tu żadnych tajemnic. Z drugiej strony na uwagę zasługuje obiekt prezentacji, który obsługuje ten szablon. Klasa SearchBox w search_box.php ma dwie funkcje:

• Gdy odwiedzający wpisze ciąg wyszukiwania i prześle formularz, pole wyszukiwania, poprzez kod stanu 302, przekierowuje żądanie do adresu URL, który odwiedzający może dodać do zakładek, na przykład http://localhost/tshirtshop/search-results/ znajdź-piękny-kwiat/wszystkie-słowa wyłączone/. O ile taki mechanizm nie zostanie zaimplementowany, wyniki wyszukiwania będą wyświetlane na stronie, którą odwiedzał odwiedzający, gdy przeprowadzał wyszukiwanie.

• Jeśli aktualnie załadowana strona jest stroną wyników wyszukiwania lub stroną szczegółów produktu, do której osiągnięto dostęp ze strony wyników wyszukiwania, pole wyszukiwania jest wypełniane słowami kluczowymi wyszukiwanymi przez odwiedzającego.

Lokalizacja, do której następuje przekierowanie żądania podczas wyszukiwania, jest obliczana za pomocą metody Link::ToSearchResults(). Zauważ, że używane jest przekierowanie 302. Jest to kod statusu, który określa tymczasową relokację strony, chociaż w tym przypadku użyty kod statusu nie ma

tak naprawdę znaczenia, ponieważ wyszukiwarki i tak nie przesłałyby formularza. Celem przekierowania nie jest optymalizacja pod kątem wyszukiwarek, ale umożliwienie użytkownikom dodawania do zakładek stron wyników wyszukiwania i oferowania innym witrynom środków do łączenia się z tymi stronami:

```
if (isset ($_GET['Search']))

{

$this->mSearchString = trim($_POST['search_string']);

$this->mAllWords = isset ($_POST['all_words']) ?

$_POST['all_words'] : 'off';

// Clean output buffer

ob_clean();

// Redirect 302

header('HTTP/1.1 302 Found');

header('Location: ' .

Link::ToSearchResults($this->mSearchString, $this->mAllWords));

// Clear the output buffer and stop execution

flush();

ob_flush();

ob_end_clean();

exit();

}
```

Oczywiście, wdrażając stronę wyników wyszukiwania w następnym ćwiczeniu, musimy nauczyć się interpretować linki wyników wyszukiwania. Obecnie link do wyszukiwania hasła „piękny kwiat” przy użyciu dowolnych słów to: http://localhost/tshirtshop/search-results/find-beautiful-flower/all-words-off/. Gdyby to było wyszukiwanie obejmujące wszystkie słowa, ostatni fragment adresu URL byłby włączony, a nie wyłączony. Druga strona wyników zawierałaby /page-2/ na końcu adresu URL. Jeśli jesteśmy na stronie wyników wyszukiwania, pobieramy ciąg wyszukiwania i wartość AllWords z ciągu zapytania. Zostaną one odczytane i wyświetlone przez szablon search_box.tpl:

```
elseif (isset ($_GET['SearchResults']))

{

$this->mSearchString = trim(str_replace('-', ' ', $_GET['SearchString']));

$this->mAllWords = isset ($_GET['AllWords']) ? $_GET['AllWords'] : 'off';

}
```

Na koniec SearchBox weryfikuje, że bieżąca strona jest stroną szczegółów produktu, do której osiągnięto stronę wyników wyszukiwania. W takim przypadku pole wyszukiwania jest również wypełnione słowami kluczowymi wyszukiwanymi przez odwiedzającego:

```
if (isset ($_GET['ProductId']) &&
isset ($_SESSION['link_to_continue_shopping']))
{
$continue_shopping =
Link::QueryStringToArray($_SESSION['link_to_continue_shopping']);
if (isset ($continue_shopping['SearchResults']))
{
$this->mSearchString =
trim(str_replace('-', ' ', $continue_shopping['SearchString']));
$this->mAllWords = $continue_shopping['AllWords'];
}
}
```

Wyświetlanie wyników wyszukiwania

W następnym ćwiczeniu utworzysz szablon z elementami, który będzie wyświetlał wyniki wyszukiwania. Aby ułatwić Ci życie, ponownie użyjemy skomponentowanego szablonu product_list, aby wyświetlić listę produktów. Jest to złożony szablon, którego obecnie używamy do tworzenia listy produktów dla strony głównej, dla działów i dla kategorii. Oczywiście, jeśli chcesz, aby wyniki wyszukiwania były wyświetlane w innym formacie, musisz utworzyć inny szablon składowy.

Będziesz musiał zmodyfikować plik logiki szablonów listy produktów (products_list.php), aby rozpoznać, kiedy jest wywoływany w celu wyświetlenia wyników wyszukiwania i wyświetlenia prawidłowej listy produktów. W poniższym ćwiczeniu utwórzmy szablon search_results i zaktualizujmy logikę szablonów w skomponentowanym szablonie products_list.

Ćwiczenie: Tworzenie szablonu składowego search_results

1. Utwórz nowy plik szablonu w katalogu Presentation/templates o nazwie search_results.tpl i dodaj do niego następujący kod:

```
{* search_results.tpl *}
<h1>Search results</h1>
{include file="products_list.tpl"}
```

2. Zmodyfikuj plik Presentation/products_list.php, dodając następujące wiersze na początku konstruktora klasy ProductsList, __construct():

```
// Retrieve the search string and AllWords from the query string
if (isset ($_GET['SearchResults']))
```

```
{
$this->mSearchString = trim(str_replace('-', ' ', $_GET['SearchString']));
$this->mAllWords = isset ($_GET['AllWords']) ? $_GET['AllWords'] : 'off';
}
```

3. Dodaj elementy $mSearchDescription, $mAllWords i $mSearchString do klasy ProductsList, znajdującej się w tym samym pliku:

```
<?php
class ProductsList
{
// Public variables to be read from Smarty template
public $mPage = 1;
public $mrTotalPages;
public $mLinkToNextPage;
public $mLinkToPreviousPage;
public $mProductListPages = array();
public $mProducts;
public $mSearchDescription;
public $mAllWords = 'off';
public $mSearchString;
// Private members
private $_mDepartmentId;
private $_mCategoryId;
```

4. Zmodyfikuj metodę init() w klasie ProductsList w następujący sposób:

```
public function init()
{
/* If searching the catalog, get the list of products by calling
the Search business tier method */
if (isset ($this->mSearchString))
{
// Get search results
$search_results = Catalog::Search($this->mSearchString,
```

```
$this->mAllWords,
$this->mPage,
$this->mrTotalPages);
// Get the list of products
$this->mProducts = $search_results['products'];
// Build the title for the list of products
if (count($search_results['accepted_words']) > 0)
$this->mSearchDescription =
'<p class="description">Products containing <font class="words">'
. ($this->mAllWords == 'on' ? 'all' : 'any') . '</font>'
. ' of these words: <font class="words">'
. implode(', ', $search_results['accepted_words']) .
'</font></p>';
if (count($search_results['ignored_words']) > 0)
$this->mSearchDescription .=
'<p class="description">Ignored words: <font class="words">'
. implode(', ', $search_results['ignored_words']) .
'</font></p>';
if (!(count($search_results['products']) > 0))
$this->mSearchDescription .=
'<p class="description">Your search generated no results.</p>';
}
/* If browsing a category, get the list of products by calling
the GetProductsInCategory business tier method */
elseif (isset ($this->_mCategoryId))
$this->mProducts = Catalog::GetProductsInCategory(
$this->_mCategoryId, $this->mPage, $this->mrTotalPages);
...
```

5. Kontynuuj, aktualizując podświetloną metodę init(), aby dodać łącza nawigacyjne na stronach wyników wyszukiwania:

```
...
```

```
/* If there are subpages of products, display navigation
controls */
if ($this->mrTotalPages > 1)
{
// Build the Next link
if ($this->mPage < $this->mrTotalPages)
{
if (isset($_GET['SearchResults']))
$this->mLinkToNextPage =
Link::ToSearchResults($this->mSearchString, $this->mAllWords,
$this->mPage + 1);
elseif (isset($this->_mCategoryId))
$this->mLinkToNextPage =
Link::ToCategory($this->_mDepartmentId, $this->_mCategoryId,
$this->mPage + 1);
elseif (isset($this->_mDepartmentId))
$this->mLinkToNextPage =
Link::ToDepartment($this->_mDepartmentId, $this->mPage + 1);
}
// Build the Previous link
if ($this->mPage > 1)
{
if (isset($_GET['SearchResults']))
$this->mLinkToPreviousPage =
Link::ToSearchResults($this->mSearchString, $this->mAllWords,
$this->mPage - 1);
elseif (isset($this->_mCategoryId))
$this->mLinkToPreviousPage =
Link::ToCategory($this->_mDepartmentId, $this->_mCategoryId,
$this->mPage - 1);
elseif (isset($this->_mDepartmentId))
```

```
$this->mLinkToPreviousPage =
Link::ToDepartment($this->_mDepartmentId, $this->mPage - 1);
}
// Build the pages links
for ($i = 1; $i <= $this->mrTotalPages; $i++)
if (isset($_GET['SearchResults']))
$this->mProductListPages[] =
Link::ToSearchResults($this->mSearchString, $this->mAllWords, $i);
elseif (isset($this->_mCategoryId))
$this->mProductListPages[] =
Link::ToCategory($this->_mDepartmentId, $this->_mCategoryId, $i);
elseif (isset($this->_mDepartmentId))
$this->mProductListPages[] =
Link::ToDepartment($this->_mDepartmentId, $i);
else
$this->mProductListPages[] = Link::ToIndex($i);
}
...
/* 404 redirect if the page number is greater than
the total number of pages */
if ($this->mPage > $this->mrTotalPages && !empty($this->mrTotalPages))
{
// Clean output buffer
ob_clean();
...
```

6. Dodaj następujące wiersze na początku prezentacji/templates/products_list.tpl:

```
{* products_list.tpl *}
{load_presentation_object filename="products_list" assign="obj"}
{if $obj->mSearchDescription != ""}
<p class="description">{$obj->mSearchDescription}</p>
{/if}
```

7. Otwórz .htaccess i wpisz następujące wiersze RewriteRule, które przekierowują strony wyników wyszukiwania:

```
# Redirect department pages
RewriteRule ^.*-d([0-9]+)/page-([0-9]+)/?$ index.php?DepartmentId=$1&Page=$2 [L]
RewriteRule ^.*-d([0-9]+)/?$ index.php?DepartmentId=$1 [L]
# Redirect search results
RewriteRule ^search-results/find-(.*)/all-words-(on|off)/page-([0-9]+)/?$
index.php?SearchResults&SearchString=$1&AllWords=$2&Page=$3 [L]
RewriteRule ^search-results/find-?(.*)/all-words-(on|off)/?$
index.php?SearchResults&SearchString=$1&AllWords=$2&Page=1 [L]
# Redirect subpages of the home page
RewriteRule ^page-([0-9]+)/?$ index.php?Page=$1 [L]
```

8. Zmodyfikuj plik Presentation/store_front.php, aby załadować skomponentowany szablon search_results podczas wyszukiwania, dodając podświetlone wiersze kodu w metodzie init():

```
...
// Load product details page if visiting a product
if (isset ($_GET['ProductId']))
$this->mContentsCell = 'product.tpl';
// Load search result page if we're searching the catalog
elseif (isset ($_GET['SearchResults']))
$this->mContentsCell = 'search_results.tpl';
// Load the page title
$this->mPageTitle = $this->_GetPageTitle();
...
```

9. Ponadto w store_front.php dodaj podświetlone wiersze kodu w metodzie _GetPageTitle(), aby wyświetlić niestandardowy tytuł podczas przeglądania strony wyników wyszukiwania:

```
...
elseif (isset ($_GET['ProductId']))
{
$page_title = 'TShirtShop: ' .
Catalog::GetProductName($_GET['ProductId']);
}
```

```
elseif (isset ($_GET['SearchResults']))
{
$page_title = 'TShirtShop: "';
// Display the search string
$page_title .= trim(str_replace('-', ' ', $_GET['SearchString'])) . '" (';
// Display "all-words search " or "any-words search"
$all_words = isset ($_GET['AllWords']) ? $_GET['AllWords'] : 'off';
$page_title .= (($all_words == 'on') ? 'all' : 'any') .
'-words search';
// Display page number
if (isset ($_GET['Page']) && ((int)$_GET['Page']) > 1)
$page_title .= ', page ' . ((int)$_GET['Page']);
$page_title .= ')';
}
else
{
if (isset ($_GET['Page']) && ((int)$_GET['Page']) > 1)
$page_title .= ' - Page ' . ((int)$_GET['Page']);
}
return $page_title;
}
...
```

10. Otwórz Presentation/product.php i dodaj podświetlony kod do metody init() klasy Product. Umożliwia to funkcję kontynuowania zakupów, gdy dotrzemy do strony produktu ze strony wyników wyszukiwania.

```
elseif (isset ($continue_shopping['DepartmentId']))
$this->mLinkToContinueShopping =
Link::ToDepartment((int)$continue_shopping['DepartmentId'], $page);
elseif (isset ($continue_shopping['SearchResults']))
$this->mLinkToContinueShopping =
Link::ToSearchResults(
trim(str_replace('-', ' ', $continue_shopping['SearchString'])),
```

$continue_shopping['AllWords'], $page);

else

$this->mLinkToContinueShopping = Link::ToIndex($page);

}

11. Dodaj następujący styl do pliku styles/tshirtshop.css:

.words {

color: #ff0000;

}

12. Załaduj swój projekt do ulubionej przeglądarki, wyszukaj słowa „boże narodzenie”, przejdź do drugiej strony wyników i spodziewaj się zobaczyć stronę taką jak na rysunku

Jak to działa: przeszukiwalny katalog produktów

Gratulacje, masz przeszukiwalny katalog produktów! Było sporo do napisania, ale kod nie był aż tak skomplikowany, prawda? Lista produktów jest wyświetlana w szablonie products_list utworzonym w rozdziale 5, ale teraz rozpoznaje, czy element Search znajduje się w ciągu zapytania, w którym to przypadku używa metody Search() warstwy biznesowej, aby uzyskać listę produktów dla gościa. Metoda Search() warstwy biznesowej zwraca tablicę zawierającą, oprócz listy zwróconych produktów, listę słów użytych do wyszukiwania oraz listę słów, które zostały zignorowane (słowa krótsze niż określona liczba znaków) . Te szczegóły są pokazywane odwiedzającemu. Nowością w tym ćwiczeniu są reguły .htaccess. Jak wiesz, gdy odwiedzający prześle nowe zapytanie, żądanie jest przekierowywane do adresu URL bogatego w słowa kluczowe, który reprezentuje stronę wyników, którą odwiedzający mogą dodać do zakładek. Reguły przepisywania odczytują adresy URL bogate w słowa kluczowe do ich dynamicznych odpowiedników:

Redirect search results

RewriteRule ^search-results/find-(.*)/all-words-(on|off)/page-([0-9]+)/?$

index.php?SearchResults&SearchString=$1&AllWords=$2&Page=$3 [L]

RewriteRule ^search-results/find-?(.*)/all-words-(on|off)/?$

index.php?SearchResults&SearchString=$1&AllWords=$2&Page=1 [L]

Pierwsza z tych reguł pasuje do adresów URL stronicowanych wyników wyszukiwania, a druga do adresów URL niestronicowanych wyników wyszukiwania.

PodsumowanieSELECT name

FROM product

WHERE description LIKE '%flower%' OR name LIKE '%flower%';

W tym rozdziale zaimplementowałeś funkcję wyszukiwania w TShirtShop, używając funkcji wyszukiwania pełnotekstowego MySQL. Mechanizm wyszukiwania bardzo dobrze integrował się z obecną strukturą witryny internetowej i funkcjonalnością stronicowania wbudowaną w Części 4. Najbardziej interesującym nowym szczegółem, którego nauczyłeś się w tym rozdziale, było wykonywanie wyszukiwania pełnotekstowego za pomocą MySQL. Był to również pierwszy raz, gdy warstwa biznesowa miała pewne własne funkcje, zamiast po prostu przekazywać dane tam iz powrotem między warstwą danych a warstwą prezentacji. W części 9 dowiesz się, jak sprzedawać swoje produkty za pomocą PayPal.