Search Engine Optimization

Po dodaniu atrybutów produktów do swojej witryny, prawdopodobnie chcesz również dodać nowe funkcje, takie jak przyjmowanie płatności użytkowników, wyszukiwanie produktów lub koszyk. Zrobimy to wkrótce, obiecujemy! Zanim to nastąpi, trzeba zadbać o jeden szczegół: musimy przygotować fundament strony internetowej, aby wesprzeć nasze wysiłki w zakresie optymalizacji pod kątem wyszukiwarek. To ważny temat, bo bezpośrednio wpływa na rentowność strony internetowej, więc zaczynajmy! Optymalizacja pod kątem wyszukiwarek, lub po prostu SEO, odnosi się do praktyk stosowanych w celu zwiększenia liczby odwiedzających witrynę internetową z bezpłatnych (bezpłatnych) stron wyników wyszukiwania. Dzisiaj wyszukiwarka jest najważniejszym narzędziem, którego ludzie używają do znajdowania informacji i produktów w Internecie. Nie trzeba dodawać, że dobra pozycja Twojej witryny e-commerce dla odpowiednich słów kluczowych pomoże przyciągnąć odwiedzających do Twojej witryny i zwiększyć szanse, że odwiedzający będą kupować od Ciebie, a nie od konkurencji! Chociaż nie (jeszcze) rakieta naukowa, optymalizacja w wyszukiwarkach jest złożonym tematem w jeszcze szerszym temacie marketingu w wyszukiwarkach. W tym rozdziale zaktualizujemy TShirtShop, aby jego podstawowa architektura była przyjazna dla wyszukiwarek, co pomoże marketerom w ich wysiłkach.

Optymalizacja TShirtShop

Co więc można ulepszyć w TShirtShop, aby był bardziej przyjazny dla wyszukiwarek? Cóż, może cię to zaskoczyć, ponieważ TShirtShop jest tak mały i młody, ale już jest wiele do ulepszenia! Na szczęście zaprojektowaliśmy jego strukturę w taki sposób, aby dodawanie nowych funkcji było bezbolesne, a nowo stworzona struktura przetrwała.

• Implementuj adresy URL zawierające słowa kluczowe za pomocą przepisywania adresów URL i modułu mod_rewrite Apache. W ten sposób, zamiast żądać index.php przy użyciu różnych parametrów ciągu zapytania, nasza witryna będzie obsługiwać adresy URL, które wyglądają lepiej zarówno dla ludzi, jak i dla wyszukiwarek, takie jak http://www.example.com/koszulka-myszki-miki .html. Na tym etapie witryna musi zostać zaktualizowana, aby wewnętrznie korzystać z nowych łączy.

• Prawidłowe przekierowywanie starych lub błędnie wpisanych adresów URL na prawidłowe adresy URL. Jest to szczególnie ważne, jeśli Twoje stare adresy URL są już od jakiegoś czasu online i zawierają linki do nich. Ten krok pomoże zapewnić, że nie stracisz żadnych rankingów, które strony w Twojej starej witrynie mogły już uzyskać (ich udział w linkach) ani nie poniesiesz kar za fałszywe zduplikowane strony w Twojej nowej witrynie. Zapewniamy również, że bez względu na sposób wpisywania adresu URL — www.twojawitryna.com lub twojawitryna.com — wyszukiwarki zrozumieją, że odmiany Twojego adresu URL to w rzeczywistości ta sama witryna. Dla człowieka mogą wyglądać jak ta sama strona internetowa, ale dla wyszukiwarki mogą to być dwie różne! Ten proces konwersji różnych form adresu URL do postaci standardowej nazywa się kanonizacją adresu URL i omówimy go później.

• Przekieruj żądania do index.php i index.html do /. Jest to ważne, ponieważ nie chcemy, aby ta sama treść była powielana w różnych adresach URL Twojej witryny. Jak się dowiesz, może to prowadzić do niejawnych lub wyraźnych kar dla wyszukiwarek.

• Dynamicznie generuj tytuły stron, aby odzwierciedlić zawartość ich stron. W tej chwili wszystkie strony mają ten sam <title>, co jest niedopuszczalne, jeśli rankingi i użyteczność w wyszukiwarkach mają jakiekolwiek znaczenie (i oczywiście są).

• Zaktualizuj paginację na wszystkich stronach katalogowych (takich jak strony kategorii i działów), aby zawierała linki do wszystkich podstron, a nie tylko linki Poprzednie i Następne. Dzięki temu strony produktów są łatwiej dostępne zarówno dla wyszukiwarek, jak i dla ludzi.

• Użyj poprawnie kodów stanu 404 (nie znaleziono strony) i 500 (błąd serwera), aby odzwierciedlić problemy ze stronami w witrynie.

W dalszej części podejmiemy więcej działań związanych z SEO, ale dzięki tym zmianom omówimy podstawy. Pomaga w tym fakt, że strona jest już odpowiednio skonstruowana. Następujące szczegóły związane z SEO zostały już zaimplementowane:

• Prawidłowo zastosowaliśmy nagłówki stron i inne znaczniki, dzięki czemu wyszukiwarki będą w stanie zidentyfikować ważny egzemplarz strony.

• Nie mamy zduplikowanych treści. Katalog nie zawiera identycznych stron, ani fragmentów stron, za które mogą zostać nałożone kary dla wyszukiwarek.

• Strony produktów, działów i kategorii są łatwo dostępne.

• Do generowania treści nie użyliśmy (i nie będziemy używać) technologii takich jak Flash i AJAX, których zawartość jest nieczytelna dla wyszukiwarek. W dalszej części tej książki zobaczysz przykład używania AJAX w sposób, który nie wpływa na widoczność w wyszukiwarkach.

Obsługa adresów URL bogatych w słowa kluczowe

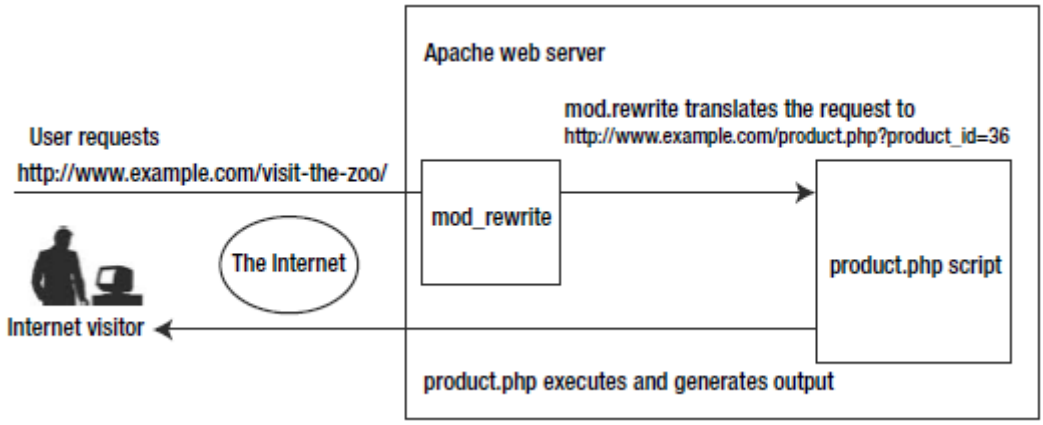
Spójrz na następujące adresy URL i wybierz ten, który Ci się bardziej podoba:

• http://localhost/tshirtshop/index.php?index.php?product_id=87

• http://localhost/tshirtshop/christmas-tree.html

Oczywiście chcesz mieć takie adresy URL w swojej witrynie internetowej. Nie tylko zawierają słowa kluczowe, które są odpowiednie dla treści strony, co może wpływać na rankingi w wyszukiwarkach, ale są również bardziej atrakcyjne dla ludzkiego odwiedzającego. Pierwszy adres URL nazwiemy dynamicznym adresem URL, a drugi adresem URL zawierającym słowa kluczowe. Krótko mówiąc, chcemy, aby nasza witryna zawierała adresy URL zawierające wiele słów kluczowych. W świecie Apache i PHP jest to zaimplementowane za pomocą modułu Apache o nazwie mod_rewrite, który może przechwytywać adresy URL zgodne z określonym wzorcem i przepisywać żądania na inny adres URL, który może być przetwarzany przez Twoją aplikację. Ten proces nazywa się przepisywaniem adresów URL.

Nawet podczas przepisywania adresów URL, końcowym plikiem, który zostanie wykonany, jest skrypt, taki jak w naszym przypadku index.php. Ale dostęp do tego samego skryptu można uzyskać za pomocą ładniej wyglądającego adresu URL. Rysunek pokazuje prosty przykład przepisywania adresów URL.



Jak widać, moduł mod_rewrite przechwytuje adres URL żądany przez odwiedzającego i przepisuje go na dynamiczny adres URL, który może zrozumieć aplikacja. Skrypt PHP jest wykonywany, a wyniki są odsyłane do naszych użytkowników, którzy (oczywiście) są bardzo zadowoleni z otrzymania pożądanej treści.

Istnieje kilka sposobów przetłumaczenia adresu URL bogatego w słowa kluczowe na jego wersję dynamiczną w celu wykonania. W TShirtShop zastosujemy technikę, która jest zarówno skuteczna, jak i prosta do wdrożenia: ukryjemy identyfikator produktu w bogatej w słowa kluczowe wersji adresów URL, na przykład w http://localhost/tshirtshop/visit-the- zoo-p36/. Ten bogaty w słowa kluczowe adres URL zawiera ukryty wewnątrz identyfikator produktu w sposób, który nie utrudnia jego czytelności zarówno dla ludzi, jak i wyszukiwarek. W tym rozdziale zaimplementujemy obsługę typów adresów URL wymienionych w tabeli. W każdym przypadku „X” jest używany jako symbol zastępczy dla działu, kategorii, identyfikatorów produktów, a „P” to numer strony.

Typ adresu URL: Format adresu URL

Adres URL strony głównej : http://www.example.com/

Adres URL strony głównej, jeśli został podzielony na strony : http://www.example.com/page-P/

Adres URL działu : http://www.example.com/nazwa-działu-dX/

Adres URL działu w przypadku podziału na strony : http://www.example.com/nazwa-działu-dX/strona-P/

Adres URL kategorii : http://www.example.com/nazwa-oddzialu-dX/nazwa-kota-cX/

Adres URL kategorii w przypadku podziału na strony : http://www.example.com/nazwa-działu-dX/nazwa-kota-cX/strona-P/

Adres URL produktu : http://www.example.com/nazwa-produktu-pX/

Jest więcej szczegółów, których musisz się nauczyć, aby w pełni obsługiwać takie adresy URL, ale robimy to pojedynczo. Wykonaj ćwiczenie, aby zaimplementować obsługę adresów URL zawierających słowa kluczowe, a szczegóły omówimy później.

Ćwiczenie: Obsługa adresów URL bogatych w słowa kluczowe

1. Pierwszym krokiem jest upewnienie się, że mod_rewrite jest włączony w twojej instalacji Apache. Jeśli pracujesz z kontem hostingowym, najprawdopodobniej mod_rewrite jest już włączony. Jeśli sam zainstalowałeś Apache, może być konieczne ręczne włączenie mod_rewrite. Na szczęście jest to proste

zadanie. Otwórz plik konfiguracyjny Apache, httpd.conf i upewnij się, że poniższa linia nie jest skomentowana przez poprzedzenie znakiem hash (#). Jeśli tak, usuń skrót, zapisz plik i uruchom ponownie Apache.

Moduły LoadModule rewrite_module/mod_rewrite.so

Wskazówka: Plik konfiguracyjny Apache znajduje się domyślnie w C:\xampp\apache\conf\ podczas instalacji XAMPP w systemie Windows. W typowej instalacji Linuksa znajdziesz go w /opt/lampp/etc/. Nie zapomnij zrestartować Apache po wprowadzeniu jakichkolwiek zmian w httpd.conf! Jeśli pojawią się jakieś błędy, sprawdź dzienniki błędów Apache, aby znaleźć więcej szczegółów na temat błędu.

2. Utwórz plik o nazwie .htaccess w folderze głównym projektu (C:\tshirtshop) i wpisz następujący kod (omówimy go szczegółowo później):

<IfModule mod_rewrite.c>

Enable mod_rewrite

RewriteEngine On

Specify the folder in which the application resides.

Use / if the application is in the root.

RewriteBase /tshirtshop

Rewrite to correct domain to avoid canonicalization problems

RewriteCond %{HTTP_HOST} !^www\.example\.com

RewriteRule ^(.*)$ http://www.example.com/$1 [R=301,L]

Rewrite URLs ending in /index.php or /index.html to /

RewriteCond %{THE_REQUEST} ^GET\ .*/index\.(php|html?)\ HTTP

RewriteRule ^(.*)index\.(php|html?)$ $1 [R=301,L]

Rewrite category pages

RewriteRule ^.*-d([0-9]+)/.*-c([0-9]+)/page-([0-9]+)/?$ index.php?Depart

mentId=$1&CategoryId=$2&Page=$3 [L]

RewriteRule ^.*-d([0-9]+)/.*-c([0-9]+)/?$ index.php?DepartmentId=$1&Cate

goryId=$2 [L]

Rewrite department pages

RewriteRule ^.*-d([0-9]+)/page-([0-9]+)/?$ index.php?DepartmentId=$1&Pag

e=$2 [L]

RewriteRule ^.*-d([0-9]+)/?$ index.php?DepartmentId=$1 [L]

Rewrite subpages of the home page

RewriteRule ^page-([0-9]+)/?$ index.php?Page=$1 [L]

Rewrite product details pages

RewriteRule ^.*-p([0-9]+)/?$ index.php?ProductId=$1 [L]

</IfModule>

Wskazówka: jeśli nie masz przyjaznego edytora kodu, utworzenie pliku, który nie ma nazwy, a jedynie rozszerzenie, takie jak .htaccess, może okazać się problematyczne w systemie Windows. Najłatwiejszym sposobem utworzenia tego pliku jest otwarcie Notatnika, wpisanie zawartości, przejście do opcji Zapisz jako i wpisanie „.htaccess" jako nazwy pliku, w tym cudzysłowów. Cytaty uniemożliwiają edytorowi automatyczne dodawanie domyślnego rozszerzenia pliku, takiego jak .txt dla Notatnika.

3. W tej chwili Twoja witryna powinna poprawnie obsługiwać adresy URL zawierające słowa kluczowe w formie opisanej przed rozpoczęciem tego ćwiczenia. Na przykład spróbuj załadować http://localhost/tshirtshop/nature-d2/. Wynik powinien przypominać stronę pokazaną na rysunku

Jak to działa: obsługa adresów URL bogatych w słowa kluczowe

W tej chwili możesz przetestować wszystkie rodzaje adresów URL bogatych w słowa kluczowe, które są obecnie znane w Twojej witrynie: strony działów i podstrony, strony i podstrony kategorii, stronę główną i jej podstrony oraz linki do szczegółów produktu. Pamiętaj jednak, że linki aktualnie generowane przez Twoją witrynę internetową są nadal starymi, dynamicznymi adresami URL. Aktualizacja linków w Twojej witrynie będzie tematem następnego ćwiczenia. Rdzeń funkcjonalności, którą właśnie zaimplementowałeś, znajduje się w pliku .htaccess. Wykorzystaliśmy ten plik konfiguracyjny Apache oparty na folderze Apache do przechowywania reguł przepisywania dla mod_rewrite. Można również użyć pliku konfiguracyjnego httpd.conf Apache, ale wybraliśmy .htaccess, ponieważ wiele scenariuszy hostingu nie pozwala na modyfikację pliku httpd.conf. Ponadto modyfikacja .htaccess nie wymaga ponownego uruchomienia serwera WWW, aby nowe ustawienia

zaczęły obowiązywać, ponieważ plik jest analizowany przy każdym żądaniu, co czyni go idealnym do celów programistycznych. Pierwsze polecenie w .htaccess to ta, która włącza silnik przepisywania. Jeśli nie skonfigurowałeś poprawnie mod_rewrite, ta linia spowoduje błąd:

RewriteEngine On

Następnie użyliśmy polecenia RewriteBase, aby określić nazwę folderu tshirtshop. Zauważ, że jeśli trzymasz swoją aplikację w folderze głównym, powinieneś zastąpić /tshirtshop /.

RewriteBase /tshirtshop

Wtedy zaczyna się prawdziwa zabawa. Poniżej znajduje się szereg poleceń RewriteRule, które zasadniczo opisują, jakie adresy URL powinny zostać przepisane i do czego powinny zostać przepisane. Czasami poleceniom RewriteRule towarzyszy RewriteCond, który określa warunek, który musi zostać spełniony, aby następujące polecenie RewriteRule zostało wykonane. Polecenie RewriteRule zawiera co najmniej dwa parametry. Pierwszy ciąg następujący po RewriteRule to wyrażenie regularne opisujące strukturę pasujących przychodzących adresów URL. Drugi opisuje, do czego należy przepisać adres URL.

mod_rewrite i wyrażenia regularne

Wyrażenia regularne to jeden z tych tematów, które programiści albo kochają, albo nienawidzą. Wyrażenie regularne, powszechnie nazywane regex, to ciąg tekstowy, który używa specjalnego formatu do opisania wzorca tekstowego. Wyrażenia regularne służą do definiowania reguł, które dopasowują lub przekształcają grupy ciągów i reprezentują jedno z najpotężniejszych dostępnych obecnie narzędzi do manipulacji tekstem. Znajdź kilka szczegółów na ich temat na stronie Wikipedii pod adresem http://en.wikipedia.org/wiki/Regular_expression.

Wyrażenia regularne są szczególnie przydatne w sytuacjach, gdy trzeba manipulować ciągami, które nie mają dobrze zdefiniowanego formatu (jak na przykład dokumenty XML) i nie można ich analizować ani modyfikować przy użyciu bardziej wyspecjalizowanych technik. Na przykład wyrażenia regularne mogą służyć do wyodrębniania lub sprawdzania poprawności adresów e-mail, znajdowania poprawnych dat w ciągach, usuwania zduplikowanych wierszy tekstu, określania liczby wystąpień słowa lub litery we frazie, znajdowania lub walidacji adresów IP, i tak dalej. W poprzednim ćwiczeniu użyłeś reguł mod_rewrite przy użyciu wyrażeń regularnych, aby dopasować przychodzące adresy URL bogate w słowa kluczowe i uzyskać ich przepisane, dynamiczne wersje. Nieco później w tym rozdziale użyjemy wyrażenia regularnego, które przygotowuje ciąg znaków do umieszczenia w adresie URL, zastępując nieobsługiwane znaki myślnikami i eliminując zduplikowane znaki rozdzielające. Wyrażenia regularne są obsługiwane przez wiele języków i narzędzi, w tym język PHP i moduł mod_rewrite Apache, a implementacje są podobne. Wyrażenie regularne, które działa w PHP, przez większość czasu będzie działać w Javie lub C# bez modyfikacji. Jeśli chcesz wykonać operację opartą na wyrażeniach regularnych, zwykle musisz podać co najmniej trzy kluczowe elementy:

• Ciąg źródłowy, który należy przeanalizować lub zmanipulować

• Wyrażenie regularne do zastosowania w ciągu źródłowym

• Rodzaj operacji do wykonania, która może polegać na uzyskaniu pasujących podciągów lub zastąpieniu ich czymś innym

Wyrażenia regularne używają specjalnej składni opartej na znakach regularnych, które są interpretowane dosłownie, oraz metaznakach, które mają specjalne właściwości dopasowania. Znak regularny w wyrażeniu regularnym pasuje do tego samego znaku w ciągu źródłowym, a sekwencja

takich znaków pasuje do tej samej sekwencji w ciągu źródłowym. Jest to podobne do wyszukiwania podciągów w ciągu. Na przykład, jeśli dopasujesz „lub” w „ulubionym kolorze”, znajdziesz dla niego dwa dopasowania. Wyrażenie regularne może zawierać metaznaki, które mają specjalne właściwości, a ich moc i elastyczność sprawia, że wyrażenia regularne są tak użyteczne. Na przykład metaznak znaku zapytania (?) określa, że poprzedzający znak jest opcjonalny. Więc jeśli chcesz dopasować „kolor” i „kolor”, Twoim wyrażeniem regularnym będzie kolor. Jak wskazano wcześniej, wyrażenia regularne mogą stać się niezwykle złożone, gdy zagłębisz się w ich bardziej subtelne szczegóły. W tej sekcji znajdziesz wyjaśnienia dotyczące wyrażeń regularnych, których używamy, i sugerujemy kontynuowanie treningu wyrażeń regularnych za pomocą specjalistycznej książki lub samouczka. Tabela 7-2 zawiera opis najpopularniejszych metaznaków wyrażeń regularnych. Możesz użyć tej tabeli jako odniesienia do zrozumienia zasad przepisywania.

Metaznak: Opis

^ : Dopasowuje początek linii. W naszym przypadku zawsze będzie pasował do początku adresu URL. Nazwa domeny nie jest uważana za część adresu URL, jeśli chodzi o RewriteRule. Przydatne jest myślenie o ^ jako zakotwiczeniu znaków, które występują na początku ciągu, czyli o zapewnieniu, że są one pierwszą częścią.

. : Dopasowuje dowolny pojedynczy znak.

* : Określa, że poprzedzający znak lub wyrażenie może być powtórzone zero lub więcej razy, czyli wcale nie do nieskończoności.

+ : Określa, że poprzedzający znak lub wyrażenie może być powtórzone raz lub więcej razy. Innymi słowy, poprzedzający znak lub wyrażenie musi pasować co najmniej raz.

? : Określa, że poprzedzający znak lub wyrażenie może być powtórzone zero lub jeden raz. Innymi słowy, poprzedzający znak lub wyrażenie jest opcjonalne.

{m,n} : określa, że poprzedzający znak lub wyrażenie może być powtórzone od m do n razy; m i n są liczbami całkowitymi, a m musi być mniejsze od n.

( ) : Nawiasy służą do definiowania przechwyconego wyrażenia. Łańcuch pasujący do wyrażenia w nawiasach można następnie odczytać jako zmienną. Nawiasy mogą być również używane do grupowania zawartych w nich treści, jak w matematyce, oraz operatorów takich jak *, + lub ? można następnie zastosować do wynikowego wyrażenia.

[ ] : Używany do definiowania klasy znaków. Na przykład [abc] dopasuje dowolny ze znaków a, b lub c. Łącznika (-) można użyć do zdefiniowania zakresu znaków. Na przykład [a-z] pasuje do dowolnej małej litery. Jeśli myślnik ma być interpretowany dosłownie, powinien to być ostatni znak przed nawiasem zamykającym,]. Wiele metaznaków traci swoją specjalną funkcję, gdy są ujęte w nawiasy i są interpretowane dosłownie.

[^] : Podobny do [ ], ale pasuje do wszystkiego oprócz wspomnianej klasy znaków. Na przykład [^a-c] dopasowuje wszystkie znaki z wyjątkiem a, b i c.

$: Dopasowuje koniec linii. W naszym przypadku zawsze będzie pasował do końca adresu URL. Przydatne jest myślenie o tym jako o zakotwiczeniu poprzednich znaków na końcu ciągu, czyli o zapewnieniu, że są one ostatnią częścią.

\ : Odwrotny ukośnik służy do zmiany znaczenia następującego znaku. Służy do ucieczki przed metaznakami, gdy trzeba je wziąć ze względu na ich dosłowną wartość, a nie ich specjalne znaczenie.

Na przykład, \. dopasuje kropkę, a nie dowolny znak (typowe znaczenie kropki w wyrażeniu regularnym). Ukośnik odwrotny może również uciec sam - więc jeśli chcesz dopasować C: \ Windows, musisz odnieść się do niego jako C: \ \ Windows.

Aby zrozumieć, jak te metaznaki działają w praktyce, przeanalizujmy jedną z reguł przepisywania w TShirtShop: tę, która przepisuje adresy URL stron kategorii. W przypadku przepisywania stron kategorii mamy dwie reguły - jedną, która obsługuje kategorie stronicowane, a drugą, która obsługuje kategorie niestronicowane. Poniższa reguła przepisuje kategorie ze stronami:

Redirect category pages

RewriteRule ^.*-d([0-9]+)/.*-c([0-9]+)/page-([0-9]+)/?$

index.php?DepartmentId=$1&CategoryId=$2&Page=$3 [L]

To wyrażenie regularne jest przeznaczone do dopasowania adresów URL, takich jak http://localhost/tshirtshop/regional-d1/french-c1/page-2 i wyodrębnienia identyfikatora działu, identyfikatora kategorii i numeru strony z tych adresów URL . W prostym języku angielskim reguła wyszukuje ciągi, które zaczynają się od pewnych znaków, po których następuje -d i liczba (będąca identyfikatorem działu), po których następuje ukośnik, kilka innych znaków, -c i inna liczba (będąca identyfikatorem kategorii), po którym następuje /page- i numer, który jest numerem strony. Korzystając z tabeli jako odniesienia, przeanalizujmy wyrażenie regularne pod względem technicznym. Wyrażenie zaczyna się od znaku ^, pasującego do początku żądanego adresu URL (adres URL nie zawiera nazwy domeny). Znaki .* odpowiadają dowolnemu ciągowi składającemu się z zera lub większej liczby znaków, ponieważ kropka oznacza dowolny znak, a gwiazdka oznacza, że poprzedzający znak lub wyrażenie (które jest kropką) może się powtórzyć zero lub więcej razy. Kolejne znaki, -d([0-9]+), wyodrębniają identyfikator działu. Bit [0-9] pasuje do dowolnego znaku z zakresu od 0 do 9 (czyli dowolnej cyfry), a znak +, który następuje, wskazuje, że wzór może się powtórzyć jeden lub więcej razy, więc możesz mieć liczbę wielocyfrową, a nie tylko pojedynczą cyfra. Otaczające nawiasy wokół [0-9]+ wskazują, że aparat wyrażeń regularnych powinien przechowywać pasujący ciąg (który będzie identyfikatorem działu) wewnątrz zmiennej o nazwie $1. Ta zmienna będzie potrzebna do skomponowania przepisanego adresu URL. Ta sama zasada jest używana do zapisywania identyfikatora kategorii i numeru strony w zmiennych $2 i $3. Wreszcie masz /?, który określa, że adres URL może kończyć się ukośnikiem, ale ukośnik jest opcjonalny. Wyrażenie regularne kończy się znakiem $, który pasuje do końca ciągu.

Uwaga : gdy musisz użyć symboli, które mają znaczenie metaznakowe jako ich wartości dosłowne, musisz je zastąpić odwrotnym ukośnikiem. Na przykład, jeśli chcesz dopasować index.php, wyrażenie regularne powinno czytać index\.php. \ to znak ucieczki, który wskazuje, że kropka powinna być traktowana jako kropka dosłowna, a nie jako dowolny znak (co ma znaczenie metaznaku kropki).

Drugi argument RewriteRule, index.php? DepartmentId=$1&CategoryId=$2&Page=$3, podłącza zmienne wyodrębnione za pomocą wyrażenia regularnego do przepisanego adresu URL. Zmienne $1, $2 i $3 są zastępowane wartościami dostarczonymi przez wyrażenie regularne, a adres URL jest ładowany przez naszą aplikację. Reguła przepisywania może również zawierać trzeci argument, który składa się ze specjalnych flag wpływających na sposób obsługi przepisywania. Te argumenty są specyficzne dla polecenia RewriteRule i nie są powiązane z wyrażeniami regularnymi. Tabela 7-3 zawiera listę możliwych argumentów reguły RewriteRule. Te flagi przepisywania muszą być zawsze umieszczane w nawiasach kwadratowych na końcu pojedynczej reguły.

Opcja RewriteRule : Znaczenie : Opis

R : Redirect : Wysyła przekierowanie HTTP.

F : Forbidden : Zabrania dostępu do adresu URL.

G : Gone : Oznacza brak adresu URL.

P : Proxy : Przekazuje adres URL do mod_proxy.

L : Last : Zatrzymuje przetwarzanie dalszych reguł.

N : Next : Rozpoczyna przetwarzanie ponownie od pierwszej reguły, ale przy użyciu aktualnego przepisanego adresu URL.

C : Chain : Łączy bieżącą regułę z następną.

T : Typ : Wymusza wspomniany typ MIME.

NS : Nosubreq : Ma zastosowanie tylko wtedy, gdy żadne wewnętrzne podżądanie nie jest wykonywane.

NC : Nocase : Dopasowywanie adresów URL nie uwzględnia wielkości liter.

QSA : Qsappend : Dołącza część ciągu zapytania do nowego adresu URL zamiast go zastępować.

PT : Passthrough : Przekazuje przepisany adres URL do innego modułu Apache w celu dalszego przetwarzania.

S : Skip : Pomija następną regułę.

E : Env : Ustawia zmienną środowiskową.

Polecenia RewriteRule są przetwarzane w kolejności sekwencyjnej, tak jak są zapisywane w pliku konfiguracyjnym. Jeśli chcesz się upewnić, że reguła jest ostatnią przetwarzaną w przypadku znalezienia dla niej dopasowania, musisz użyć flagi [L]. Ta flaga jest szczególnie przydatna, jeśli masz długą listę poleceń RewriteRule, ponieważ użycie [L] poprawia wydajność i zapobiega przetwarzaniu przez mod_rewrite wszystkich poleceń RewriteRule, które następują po znalezieniu dopasowania. To jest to, czego chcesz, niezależnie od tego. Nasza ostatnia uwaga na temat reguł .htaccess dotyczy następującego kodu:

```
# Redirect to correct domain to avoid canonicalization problems

#RewriteCond %{HTTP_HOST} !^www\.example\.com

#RewriteRule ^(.*)$ http://www.example.com/$1 [R=301,L]
```

Jak widać, polecenia RewriteCond i RewriteRule są zakomentowane za pomocą znaku #. Skomentowaliśmy te linie, ponieważ powinieneś zmienić www.example.com na lokalizację swojej witryny internetowej przed ich odkomentowaniem (pracując na localhost, zostaw te zasady wykomentowane). RewriteCond to polecenie mod_rewrite, które umieszcza warunek dla następującej reguły. W takim przypadku chcesz sprawdzić, czy witryna była dostępna przez www.example.com. Jeśli tak nie jest, wykonujesz przekierowanie 301 do www.example.com. Ta technika implementuje kanonizację nazw domen. Jeśli do Twojej witryny można uzyskać dostęp za pośrednictwem wielu nazw domen (takich jak www.example.com i example.com), ustaw jedną z nich jako domenę główną i przekieruj do niej wszystkie pozostałe, unikając nakładania przez wyszukiwarki kar za powielanie treści. Więcej o przekierowaniach 301 dowiesz się nieco później.

Tworzenie adresów URL bogatych w słowa kluczowe

W poprzednim ćwiczeniu osiągnąłeś świetną rzecz: zacząłeś obsługiwać adresy URL zawierające słowa kluczowe w TShirtShop! Pamiętaj jednak, że

• Twoja witryna obsługuje również dynamiczne adresy URL.

• Wszystkie łącza w Twojej witrynie internetowej korzystają z dynamicznych wersji adresów URL.

Przy tych dwóch wadach sam fakt, że obsługujemy adresy URL bogate w słowa kluczowe, nie przynosi żadnych znaczących korzyści. To prowadzi nas do drugiego ćwiczenia związanego z naszymi adresami URL. Tym razem zmienimy dynamiczne linki w naszej witrynie na adresy URL zawierające wiele słów kluczowych. We wcześniejszych rozdziałach byliśmy wystarczająco mądrzy, aby użyć scentralizowanej klasy o nazwie Link, która generuje wszystkie linki do witryny. Oznacza to, że teraz aktualizacja wszystkich linków w naszej witrynie to tylko kwestia aktualizacji tej klasy Link. Będziemy również musieli zbudować infrastrukturę warstwy danych i warstwy biznesowej, aby obsługiwać nową funkcjonalność, która składa się z metod zwracających nazwę działu, kategorii lub produktu, jeśli dostarczymy identyfikator.

Ćwiczenie: Generowanie adresów URL bogatych w słowa kluczowe

1. Użyj phpMyAdmin, aby połączyć się z bazą danych tshirtshop i wykonaj następujący kod, który tworzy trzy procedury składowane. Są to proste procedury, które zwracają nazwę działu, kategorii lub produktu po podaniu jego identyfikatora. Nie zapomnij ustawić $$ jako ogranicznika przed wykonaniem kodu.

```
-- Create catalog_get_department_name stored procedure

CREATE PROCEDURE catalog_get_department_name(IN inDepartmentId INT)

BEGIN

SELECT name FROM department WHERE department_id = inDepartmentId;

END$$

-- Create catalog_get_category_name stored procedure

CREATE PROCEDURE catalog_get_category_name(IN inCategoryId INT)

BEGIN

SELECT name FROM category WHERE category_id = inCategoryId;

END$$

-- Create catalog_get_product_name stored procedure

CREATE PROCEDURE catalog_get_product_name(IN inProductId INT)

BEGIN

SELECT name FROM product WHERE product_id = inProductId;

END$$
```

2. Dodamy teraz kod warstwy biznesowej, który uzyskuje dostęp do utworzonych wcześniej procedur składowanych. Dodaj następujący kod do klasy Catalog w business/catalog.php:

```
// Retrieves department name
public static function GetDepartmentName($departmentId)
{
// Build SQL query
$sql = 'CALL catalog_get_department_name(:department_id)';
// Build the parameters array
$params = array (':department_id' => $departmentId);
// Execute the query and return the results
return DatabaseHandler::GetOne($sql, $params);
}
// Retrieves category name
public static function GetCategoryName($categoryId)
{
// Build SQL query
$sql = 'CALL catalog_get_category_name(:category_id)';
// Build the parameters array
$params = array (':category_id' => $categoryId);
// Execute the query and return the results
return DatabaseHandler::GetOne($sql, $params);
}
// Retrieves product name
public static function GetProductName($productId)
{
// Build SQL query
$sql = 'CALL catalog_get_product_name(:product_id)';
// Build the parameters array
$params = array (':product_id' => $productId);
// Execute the query and return the results
return DatabaseHandler::GetOne($sql, $params);
}
```

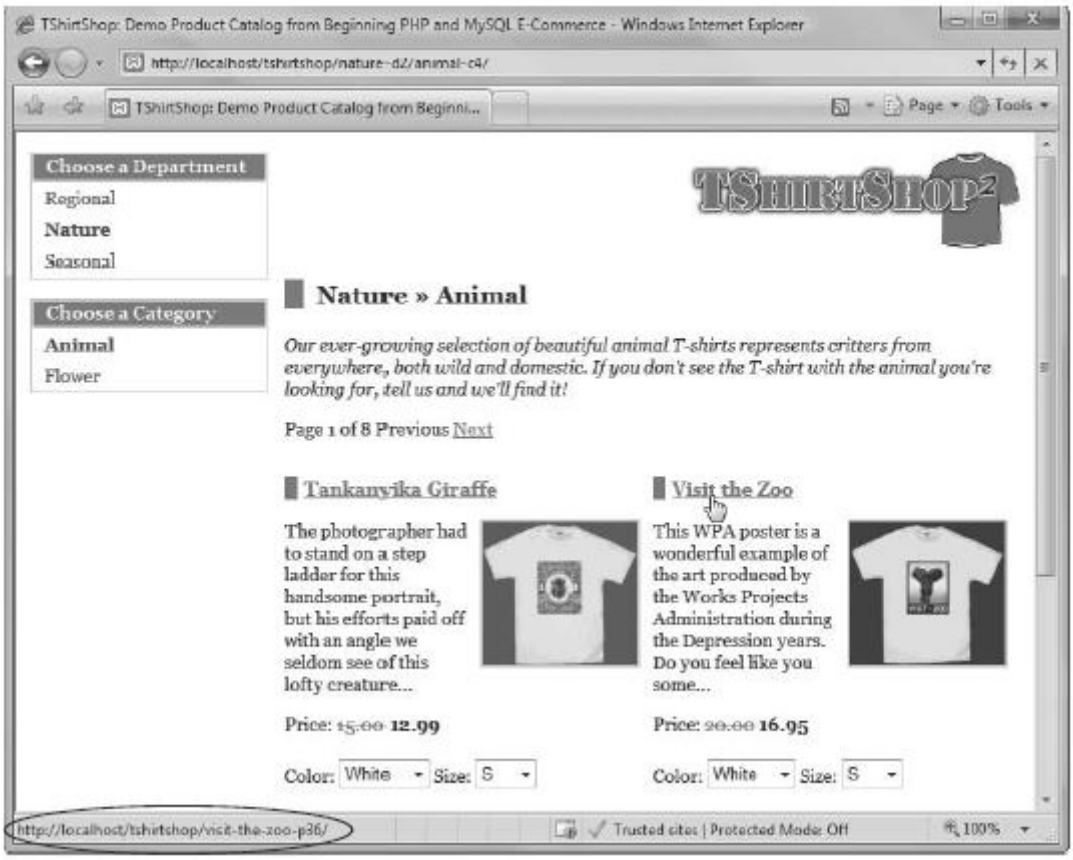
3. Otwórz prezentację/link.php i zmodyfikuj jej kod w następujący sposób:

```
public static function ToDepartment($departmentId, $page = 1)
{
$link = self::CleanUrlText(Catalog::GetDepartmentName($departmentId)) .
'-d' . $departmentId . '/';
if ($page > 1)
$link .= 'page-' . $page . '/';
return self::Build($link);
}
public static function ToCategory($departmentId, $categoryId, $page = 1)
{
$link = self::CleanUrlText(Catalog::GetDepartmentName($departmentId)) .
'-d' . $departmentId . '/' .
self::CleanUrlText(Catalog::GetCategoryName($categoryId)) .
'-c' . $categoryId . '/';
if ($page > 1)
$link .= 'page-' . $page . '/';
return self::Build($link);
}
public static function ToProduct($productId)
{
$link = self::CleanUrlText(Catalog::GetProductName($productId)) .
'-p' . $productId . '/';
return self::Build($link);
}
public static function ToIndex($page = 1)
{
$link = '';
if ($page > 1)
$link .= 'page-' . $page . '/';
return self::Build($link);
}
```

4. Kontynuuj pracę nad klasą Link, dodając następującą metodę CleanUrlText(), która jest wywoływana przez zaktualizowane wcześniej metody w celu usunięcia złych znaków z łączy:

```
// Prepares a string to be included in an URL

public static function CleanUrlText($string)

{

// Remove all characters that aren't a-z, 0-9, dash, underscore or space

$not_acceptable_characters_regex = '#[^-a-zA-Z0-9_ ]#';

$string = preg_replace($not_acceptable_characters_regex, '', $string);

// Remove all leading and trailing spaces

$string = trim($string);

// Change all dashes, underscores and spaces to dashes

$string = preg_replace('#[-_ ]+#', '-', $string);

// Return the modified string

return strtolower($string);

}
```

5. Załaduj TShirtShop i zwróć uwagę na nowe linki. Na rysunku link do produktu Odwiedź Zoo, http://localhost/tshirtshop/visit-the-zoo-p36/, jest widoczny na pasku stanu Internet Explorera.



Jak to działa: generowanie adresów URL bogatych w słowa kluczowe

W tym ćwiczeniu zmodyfikowano metody ToIndex(), ToDepartment(), ToCategory() i ToProduct() klasy Link w celu utworzenia adresów URL zawierających słowa kluczowe zamiast dynamicznych adresów URL. W celu obsługi tej funkcji utworzono kod infrastruktury (metody warstwy biznesowej i procedury składowane w bazie danych), który pobiera nazwy działów, produktów i kategorii z bazy danych. Zaimplementowałeś również metodę o nazwie CleanUrlText(), która używa wyrażeń regularnych do zastępowania znaków, których nie chcemy umieszczać w adresach URL, myślnikami. Ta metoda przekształca ciąg, taki jak „Odwiedź zoo” na ciąg przyjazny dla adresu URL, taki jak „odwiedź zoo”. Upewnij się, że wszystkie linki w Twojej witrynie są teraz przyjazne dla wyszukiwarek i przejdźmy do następnego zadania.

Korekta adresu URL z przekierowaniami 301

Jednym z potencjalnych problemów z naszą witryną jest to, że do tej samej strony można dotrzeć za pomocą wielu różnych linków. Weźmy na przykład następujące adresy URL:

http://localhost/tshirtshop/nature-d2/

http://localhost/tshirtshop/TYPO-d2/

Ponieważ zawartość jest pobierana na podstawie ukrytego identyfikatora w linkach, który w tych przykładach wynosi 2, oba linki ładują dział Nature, którego prawidłowy link to http://localhost/tshirtshop/nature-d2/.

Ta elastyczność może mieć potencjalnie niekorzystny wpływ na rankingi w wyszukiwarkach. Jeśli z jakiegoś powodu wyszukiwarki dotrą do tej samej strony za pomocą różnych linków, pomyślą, że masz

wiele różnych stron z identyczną zawartością w Twojej witrynie i mogą błędnie założyć, że masz witrynę ze spamem. W takim skrajnym przypadku Twoja witryna jako całość lub tylko jej część może zostać ukarana. Nawet w przypadku braku wyraźnej kary ze strony wyszukiwarek, posiadanie kapitału treści podzielonego przez wiele adresów URL może samo z siebie obniżyć rankingi wyszukiwarek. Rozwiązaniem, które zalecamy, aby uniknąć kar, jest prawidłowe użycie kodów statusu HTTP w celu przekierowania wszystkich stron o identycznej treści do jednego, standardowego adresu URL.

KODY STANU HTTP

Kody statusu HTTP to kody, które są wysyłane jako odpowiedź na żądanie sieciowe wraz z żądaną treścią i wskazują stan żądania. Jako programista stron internetowych prawdopodobnie znasz kod stanu 200, który wskazuje, że żądanie zakończyło się powodzeniem, oraz kod 404, który wskazuje, że nie można znaleźć żądanego zasobu. Wśród kodów stanu HTTP jest kilka, które konkretnie dotyczą problemów z przekierowaniem. Najczęstszym z tych kodów stanu przekierowania jest 301, który wskazuje, że żądany zasób został na stałe przeniesiony do nowej lokalizacji, oraz 302, który wskazuje, że przemieszczenie jest tylko tymczasowe. Gdy przeglądarka internetowa lub wyszukiwarka wysyła żądanie, którego odpowiedź zawiera kod stanu przekierowania, kontynuują przeglądanie we wskazanej lokalizacji. Przeglądarka internetowa zażąda nowego adresu URL i zaktualizuje pasek adresu, aby odzwierciedlał nową lokalizację. Domyślny kod stanu przekierowania to 302. Warto to wiedzieć, ponieważ podczas optymalizacji pod kątem wyszukiwarek zwykle będziesz chciał użyć przekierowań 301. Jeśli chodzi o SEO, przekierowania 301 są preferowane, ponieważ (powinny) również przenosić kapitał linków ze starego adresu URL na nowy adres URL. Oznacza to, że jeśli stary adres URL miał dobrą pozycję w rankingu dla niektórych słów kluczowych, jeśli zostanie użyty 301, nowy adres URL będzie miał taką samą pozycję jak stary, po tym, jak wyszukiwarki zauważą przekierowanie. W praktyce nadużywanie 301 nie jest pożądane, ponieważ nie ma gwarancji, że kapitał linków zostanie całkowicie przeniesiony – a nawet jeśli tak się stanie, może minąć trochę czasu, zanim ponownie uzyskasz dobrą pozycję w rankingu dla pożądanych słów kluczowych. Bardziej subtelne szczegóły dotyczące przekierowań i kodów statusu HTTP można znaleźć w artykule Professional Search Engine Optimization with PHP: A Developer's Guide to SEO autorstwa Cristiana Darie i Jaimie Sirovicha. Naszym celem w następnym ćwiczeniu jest stworzenie standardowej („właściwej") wersji adresu URL dla każdej strony w naszej witrynie. Po załadowaniu strony porównujemy znany, standardowy adres URL strony z adresem żądanym przez użytkownika. Jeśli się nie zgadzają, wykonujemy przekierowanie 301 do odpowiedniej wersji adresu URL. Jak wspomniano wcześniej, korekta adresu URL jest przydatna, gdy ktoś wpisze adres URL z literówką, na przykład http://localhost/tshirtshop/natureTYPO-d2/, lub gdy zmienisz nazwę produktu, kategorii lub działu, co powoduje Zmienia się również adres URL.

Ćwiczenie: Implementacja korekty adresu URL

1. Korekta adresów URL i inne funkcje, które zaimplementujemy w tym rozdziale, obejmują pracę z nagłówkami HTTP. Aby uniknąć problemów z ustawianiem nagłówków, musimy dokonać tej zmiany w index.php. Dodaj następujący podświetlony kod do pliku index.php:

```
<?php

// Activate session

session_start();

// Start output buffer

ob_start();
```

```
// Include utility files
require_once 'include/config.php';
require_once BUSINESS_DIR . 'error_handler.php';
```

2. Na końcu index.php dodaj następujący kod:

```
// Close database connection
DatabaseHandler::Close();
// Output content from the buffer
flush();
ob_flush();
ob_end_clean();
?>
```

3. Dodaj metodę CheckRequest() do klasy Link w pliku Presentation/link.php:

```
// Redirects to proper URL if not already there
public static function CheckRequest()
{
$proper_url = '';
// Obtain proper URL for category pages
if (isset ($_GET['DepartmentId']) && isset ($_GET['CategoryId']))
{
if (isset ($_GET['Page']))
$proper_url = self::ToCategory($_GET['DepartmentId'],
$_GET['CategoryId'], $_GET['Page']);
else
$proper_url = self::ToCategory($_GET['DepartmentId'],
$_GET['CategoryId']);
}
// Obtain proper URL for department pages
elseif (isset ($_GET['DepartmentId']))
{
if (isset ($_GET['Page']))
$proper_url = self::ToDepartment($_GET['DepartmentId'],
```

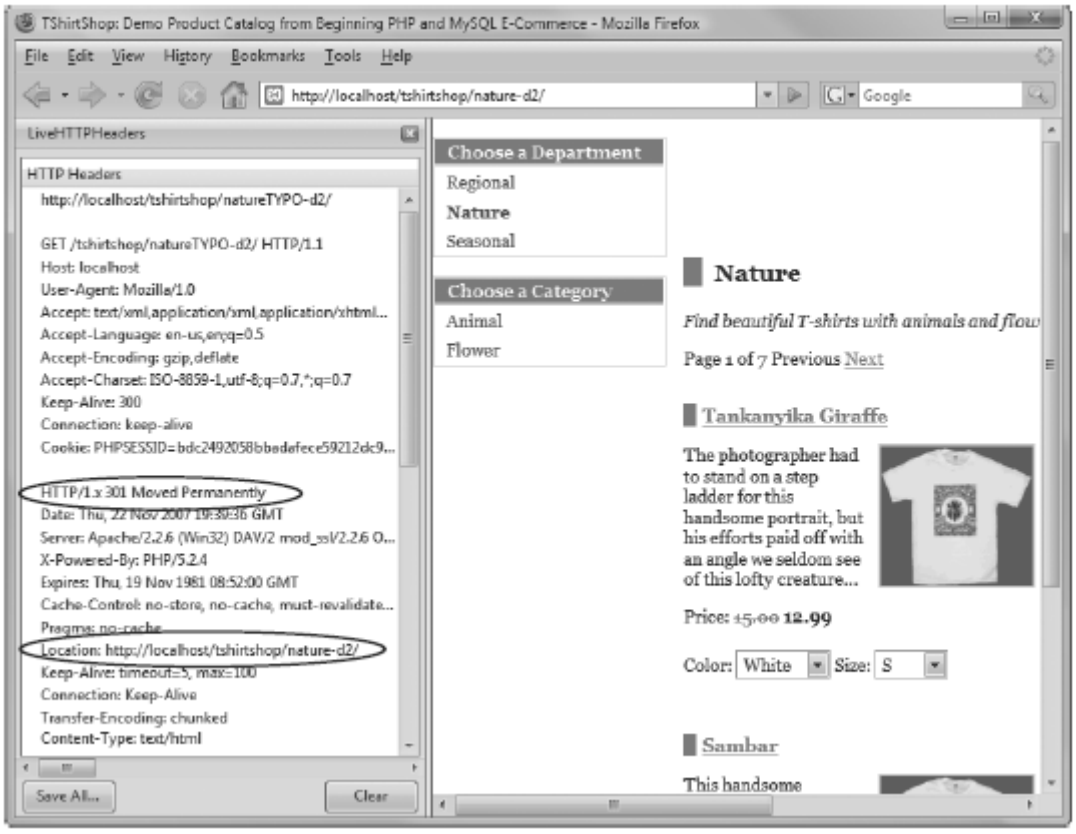
```
$_GET['Page']);
else
$proper_url = self::ToDepartment($_GET['DepartmentId']);
}
// Obtain proper URL for product pages
elseif (isset ($_GET['ProductId']))
{
$proper_url = self::ToProduct($_GET['ProductId']);
}
// Obtain proper URL for the home page
else
{
if (isset($_GET['Page']))
$proper_url = self::ToIndex($_GET['Page']);
else
$proper_url = self::ToIndex();
}
/* Remove the virtual location from the requested URL
so we can compare paths */
$requested_url = self::Build(str_replace(VIRTUAL_LOCATION, '',
$_SERVER['REQUEST_URI']));
// 301 redirect to the proper URL if necessary
if ($requested_url != $proper_url)
{
// Clean output buffer
ob_clean();
// Redirect 301
header('HTTP/1.1 301 Moved Permanently');
header('Location: ' . $proper_url);
// Clear the output buffer and stop execution
flush();
```

```
ob_flush();
ob_end_clean();
exit();
}
}
```

4. Otwórz index.php i wywołaj tę metodę w ten sposób:

```
// Load the database handler
require_once BUSINESS_DIR . 'database_handler.php';
// Load Business Tier
require_once BUSINESS_DIR . 'catalog.php';
// URL correction
Link::CheckRequest();
// Load Smarty template file
$application = new Application();
// Display the page
$application->display('store_front.tpl');
// Close database connection
DatabaseHandler::Close();
```

5. Załaduj http://localhost/tshirtshop/natureTYPO-d2/ i zauważ, że strona przekierowuje do http://localhost/tshirtshop/nature-d2/. Używając narzędzia takiego jak rozszerzenie LiveHTTPHeaders Firefox (http://livehttpheaders.mozdev.org/), możesz zobaczyć, że typ przekierowania to 301; patrz Rysunek.



Uwaga : Inne narzędzia, których możesz użyć do wyświetlenia nagłówków HTTP, to Pomocnik Web Development Helper i Fiddler dla Internet Explorera i FireBug lub wtyczka Web Developer dla Firefoksa.

Jak to działa: używanie 301 do przekierowywania treści

Kod podąża za prostą logiką, aby wykonać zadanie. Metoda CheckRequest() klasy Link weryfikuje, czy żądanie powinno zostać przekierowane na inny adres URL, a jeśli tak, to wykonuje przekierowanie 301. Sposobem wykonania przekierowania w PHP jest ustawienie nagłówka HTTP w następujący sposób:

```
// 301 redirect to the proper URL if necessary

if ($requested_url != $proper_url)

{

// Clean output buffer

ob_clean();

// Redirect 301

header('HTTP/1.1 301 Moved Permanently');

header('Location: ' . $proper_url);

// Clear the output buffer and stop execution

flush();

ob_flush();
```

```
ob_end_clean();

exit();

}
```

Wywołujemy CheckRequest() w index.php, aby upewnić się, że sprawdza wszystkie przychodzące żądania. Zmieniliśmy także index.php, dodając kod kontroli wyjścia, aby zapewnić, że będziemy mogli opróżnić dane wyjściowe i zmienić nagłówki wyjściowe w razie potrzeby, ponieważ nagłówków nie można zmienić po wysłaniu jakichkolwiek danych wyjściowych do klienta. Przeczytaj więcej o funkcjach kontroli wyjścia PHP na http://php.net/outcontrol. Przydatny artykuł na ten temat można znaleźć pod adresem http://www.phpit.net/article/output-buffer-fun-php/.

Dostosowywanie tytułów stron

Jednym z typowych błędów popełnianych przez twórców stron internetowych jest ustawienie tego samego tytułu dla wszystkich stron w witrynie internetowej. Szkoda, bo tytuł strony jest, zdaniem wielu autorytetów SEO, najważniejszym czynnikiem w algorytmie rankingu wyszukiwarek. Potwierdza to artykuł pod adresem http://www.seomoz.org/article/search-ranking-factors. W tej chwili wszystkie strony w TShirtShop mają ten sam tytuł, który jest zdefiniowany w site.conf. W poniższym ćwiczeniu zobaczysz, że łatwo jest zaktualizować witrynę, aby wyświetlać dostosowane tytuły stron dla każdego obszaru witryny.

Ćwiczenie: Generowanie niestandardowych tytułów stron

1. Otwórz Presentation/store_front.php i dodaj podświetlony element do klasy StoreFront:

```
<?php

class StoreFront

{

public $mSiteUrl;

// Define the template file for the page contents

public $mContentsCell = 'first_page_contents.tpl';

// Define the template file for the categories cell

public $mCategoriesCell = 'blank.tpl';

// Page title

public $mPageTitle;
```

2. W tej samej klasie StoreFront dodaj następujący kod na końcu metody init():

```
// Load product details page if visiting a product

if (isset ($_GET['ProductId']))

$this->mContentsCell = 'product.tpl';

// Load the page title

$this->mPageTitle = $this->_GetPageTitle();
```

```
}
```

3. Kontynuuj aktualizację klasy StoreFront, dodając następującą metodę prywatną:

```
// Returns the page title
private function _GetPageTitle()
{
$page_title = 'TShirtShop: ' .
'Demo Product Catalog from Beginning PHP and MySQL E-Commerce';
if (isset ($_GET['DepartmentId']) && isset ($_GET['CategoryId']))
{
$page_title = 'TShirtShop: ' .
Catalog::GetDepartmentName($_GET['DepartmentId']) . ' - ' .
Catalog::GetCategoryName($_GET['CategoryId']);
if (isset ($_GET['Page']) && ((int)$_GET['Page']) > 1)
$page_title .= ' - Page ' . ((int)$_GET['Page']);
}
elseif (isset ($_GET['DepartmentId']))
{
$page_title = 'TShirtShop: ' .
Catalog::GetDepartmentName($_GET['DepartmentId']);
if (isset ($_GET['Page']) && ((int)$_GET['Page']) > 1)
$page_title .= ' - Page ' . ((int)$_GET['Page']);
}
elseif (isset ($_GET['ProductId']))
{
$page_title = 'TShirtShop: ' .
Catalog::GetProductName($_GET['ProductId']);
}
else
{
if (isset ($_GET['Page']) && ((int)$_GET['Page']) > 1)
$page_title .= ' - Page ' . ((int)$_GET['Page']);
```

```
}

return $page_title;

}
```

4. Zaktualizuj prezentację/templates/store_front.tpl w następujący sposób:

```
<html>

<head>

<title>{$obj->mPageTitle}</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link href="{$obj->mSiteUrl}styles/tshirtshop.css" type="text/css"

rel="stylesheet" />

</head>
```

5. Załaduj stronę inną niż strona główna w TShirtShop i zwróć uwagę na jej nowy, dostosowany tytuł strony, który jest podświetlony na rysunku (tytuł strony głównej pozostaje taki sam).

Jak to działa: tworzenie tytułów stron

W tym ćwiczeniu zaktualizowaliśmy klasę StoreFront, aby używała danych zebranych za pomocą funkcji GetDepartmentName(), GetCategoryName() i GetProductName() klasy Catalog do tworzenia pożądanych tytułów dla stron działów, kategorii i produktów. Szablon Smarty został również zaktualizowany, aby wyświetlać nowo utworzony tytuł zamiast domyślnego. Nie będziemy omawiać szczegółów, ponieważ kod jest dość prosty.

Aktualizowanie stronicowania katalogu

Podobnie jak wyszukiwarki zakładają, że strony, które nie są dobrze połączone ze źródłami zewnętrznymi, są mniej ważne niż te, które są, mogą zakładać, że strony ukryte w wewnętrznej strukturze linków witryny internetowej nie są zbyt ważne. Nasz obecny system nawigacji po stronach produktów jest doskonałym przykładem zakopywania stron w hierarchii linków. Aby nawigować między stronami produktów, obecnie oferujemy tylko linki Poprzednie i Następne. Nie ułatwia to odwiedzającym nawigowania bezpośrednio do różnych stron produktów, a także nie ułatwia wyszukiwarkom. Rozważmy przykład czwartej strony produktów z kategorii Regional. Obecnie do tej strony mogą dotrzeć ludzie lub wyszukiwarki takie jak ta:

Home -> Regional -> Page 2 -> Page 3 -> Page 4

Czwarta strona produktów jest trudniej dostępna nie tylko dla ludzi (którzy muszą kliknąć co najmniej cztery razy), ale także dla wyszukiwarek. Rozwiążmy ten problem, wykonując krótkie ćwiczenie.

Ćwiczenie: Paginacja SEO

1. W klasie ProductsList z pliku Presentation/products_list.php zmodyfikuj metodę init() klasy ProductList, jak pokazano:

```
/* If there are subpages of products, display navigation
controls */
if ($this->mrTotalPages > 1)
{
// Build the Next link
if ($this->mPage < $this->mrTotalPages)
{
if (isset($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($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; $i++)
if (isset($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);
}
```

2. Otwórz plik Presentation/templates/products_list.tpl i zmodyfikuj go zgodnie z podświetleniem:

```
{* products_list.tpl *}
{load_presentation_object filename="products_list" assign="obj"}
{if count($obj->mProductListPages) > 0}
<p>
{if $obj->mLinkToPreviousPage}
<a href="{$obj->mLinkToPreviousPage}">Previous page</a>
{/if}
{section name=m loop=$obj->mProductListPages}
{if $obj->mPage eq $smarty.section.m.index_next}
<strong>{$smarty.section.m.index_next}</strong>
{else}
<a href="{$obj->mProductListPages[m]}">{$smarty.section.m.index_next}</a>
```

{/if}

{/section}

{if $obj->mLinkToNextPage}

mLinkToNextPage}">Next page

{/if}

</p>

{/if}

{if $obj->mProducts}

<table class="product-list" border="0">

<tbody>

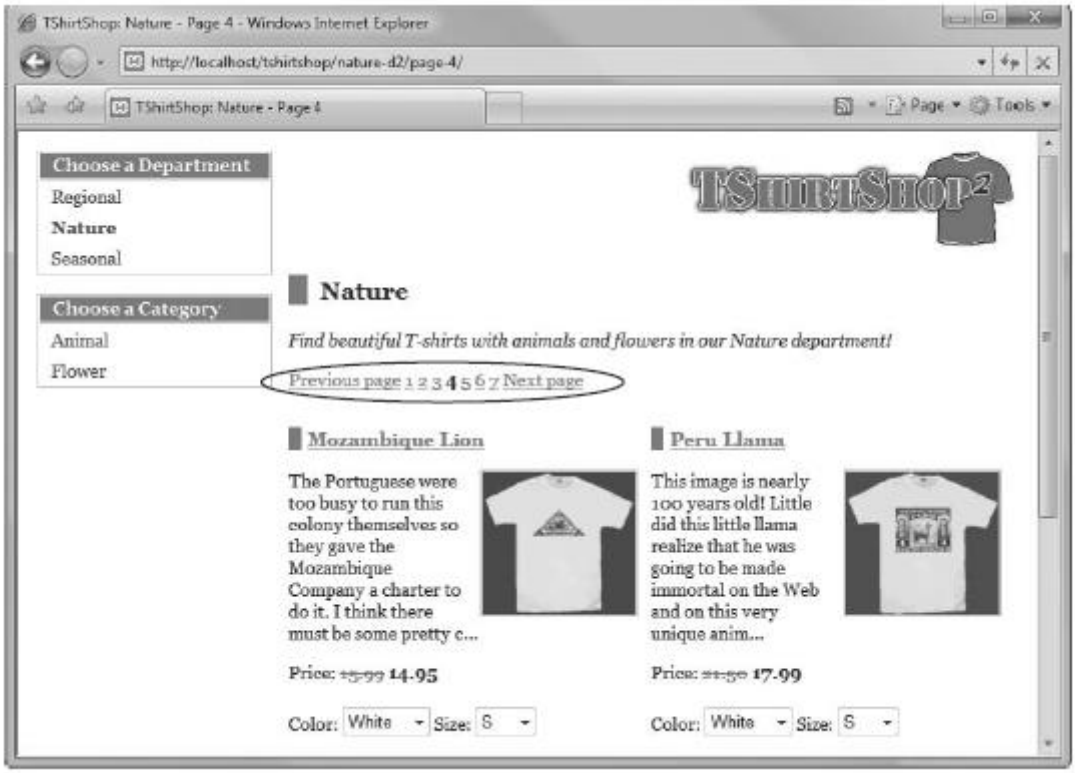
{section name=k loop=$obj->mProducts}

{if $smarty.section.k.index % 2 == 0}

<tr>

{/if}

3. Załaduj TShirtShop i przejdź do działu regionalnego. Na rysunku widać nowe łącza stronicowania.



Jak to działa: paginacja

Dzięki zaimplementowaniu tej małej sztuczki Twój katalog jest teraz łatwo dostępny zarówno dla osób odwiedzających, jak i odwiedzających elektronicznych. Użytkownicy z pewnością docenią pomoc w

szybkim przechodzeniu do poszczególnych stron produktów, a wyszukiwarki znacznie łatwiej je znajdą i zindeksują.

Prawidłowa sygnalizacja błędów 404 i 500

Ważne jest, aby użyć poprawnego kodu statusu HTTP, gdy coś wyjątkowego dzieje się z żądaniem odwiedzającego. Widziałeś już, że podczas wykonywania przekierowań znajomość kodów stanu HTTP może mieć istotny wpływ na wysiłki związane z optymalizacją wyszukiwarek. Tym razem porozmawiamy o 404 i 500. Kod stanu 404 służy do poinformowania odwiedzającego, że zażądał strony, która nie istnieje w docelowej witrynie internetowej. Przeglądarki i serwery internetowe mają szablony, które użytkownicy otrzymują, gdy wyślesz takie żądanie – wiesz, że je widziałeś. Usługi hostingowe pozwalają określić niestandardową stronę, która ma być wyświetlana, gdy wystąpi taki błąd 404. Jest to oczywiście korzystne dla Twojej witryny, ponieważ możesz przekazać odwiedzającemu niestandardowe informacje zwrotne w zależności od tego, czego szukał. Czasami jednak kod stanu 404 nie jest ustawiany automatycznie, więc musisz to zrobić w swoim skrypcie 404. Jeśli z jakiegoś powodu Twoja witryna zareaguje na błędy 404, wysyłając strony z kodem stanu 200 OK, wyszukiwarki pomyślą, że masz wiele różnych adresów URL zawierających tę samą treść, a Twoja witryna może zostać ukarana. Komunikat o stanie 500 służy do informowania o błędach wewnętrznych serwera WWW lub aplikacji. W poniższym ćwiczeniu dostosujemy TShirtShop, aby poprawnie używał kodów stanu 404 i 500.

Ćwiczenie: Używanie kodu statusu HTTP 500

1. Otwórz business\error_handler.php i zmodyfikuj metodę Handler(), jak pokazano w poniższym fragmencie kodu:

```
/* Warnings don't abort execution if IS_WARNING_FATAL is false

E_NOTICE and E_USER_NOTICE errors don't abort execution */

if (($errNo == E_WARNING && IS_WARNING_FATAL == false) ||

($errNo == E_NOTICE || $errNo == E_USER_NOTICE))

// If the error is nonfatal ...

{

// Show message only if DEBUGGING is true

if (DEBUGGING == true)

echo '<div class="error_box"><pre>' . $error_message . '</pre></div>';

}

else

// If error is fatal ...

{

// Show error message

if (DEBUGGING == true)

echo '<div class="error_box"><pre>' . $error_message . '</pre></div>';
```

```
else
{
// Clean output buffer
ob_clean();
// Load the 500 page
include '500.php';
// Clear the output buffer and stop execution
flush();
ob_flush();
ob_end_clean();
exit();
}
// Stop processing the request
exit();
}
}
```

2. W folderze głównym aplikacji utwórz plik o nazwie 500.php i wpisz następujący kod:

```
<?php
// Set the 500 status code
header('HTTP/1.0 500 Internal Server Error');
require_once 'include/config.php';
require_once PRESENTATION_DIR . 'link.php';
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>
TShirtShop Application Error (500): Demo Product Catalog from
Beginning PHP and MySQL E-Commerce
</title>
```

```
<link href="<?php echo Link::Build('styles/tshirtshop.css'); ?>"
type="text/css" rel="stylesheet" />
</head>
<body>
<div id="doc" class="yui-t7">
<div id="bd">
<div id="header" class="yui-g">
<a href="<?php echo Link::Build(''); ?>">
<img src="<?php echo Link::Build('images/tshirtshop.png'); ?>"
alt="tshirtshop logo" />
</a>
</div>
<div id="contents" class="yui-g">
<h1>
TShirtShop is experiencing technical difficulties.
</h1>
<p>
Please
<a href="<?php echo Link::Build(''); ?>">visit us</a> soon,
or <a href="<?php echo ADMIN_ERROR_MAIL; ?>">contact us</a>.
</p>
<p>Thank you!</p>
<p>The TShirtShop team.</p>
</div>
</div>
</div>
</body>
</html>
```

3. Dodaj podświetlone wiersze kodu na końcu pliku .htaccess:

```
</IfModule>
# Ustaw domyślną stronę 500 dla błędów Apache
```

ErrorDocument 500 /tshirtshop/500.php

Uwaga: pamiętaj o zmodyfikowaniu adresu URL do lokalizacji pliku 500.php.

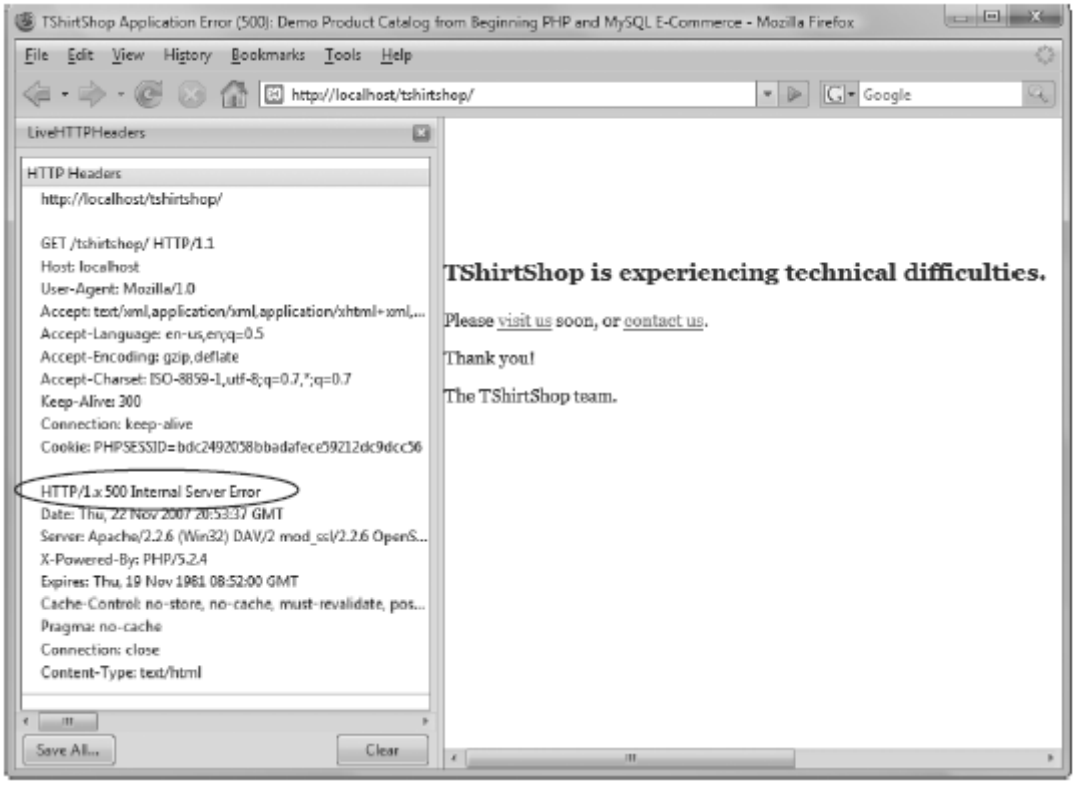
4. Przetestujmy nasz nowy plik 500.php, tworząc błąd na naszej stronie internetowej. Otwórz include\config.php i ustaw DEBUGGING const na false, aby wyłączyć tryb debugowania (w przeciwnym razie nasza strona nie zwróci 500 błędów):

```
// These should be true while developing the web site

define('IS_WARNING_FATAL', true);

define('DEBUGGING', false);
```

5. Następnie otwórz index.php i dodaj odwołanie do nieistniejącego pliku:

```
// URL correction

Link::CheckRequest();

require_once('inexistent_file.php');
```

6. Teraz załaduj swoją aplikację. Jeśli wszystko działa zgodnie z oczekiwaniami, powinieneś otrzymać stronę 500 pokazaną na rysunku.



Jak to działa: obsługa 500 błędów

Jak widać, jeśli wystąpi błąd aplikacji, odwiedzającemu zostanie wyświetlona właściwa strona błędu. Kod stanu jest prawidłowo ustawiony na 500, więc wyszukiwarki będą wiedziały, że witryna ma problemy i nie zindeksuje strony błędu 500. Zamiast tego, poprzednio zindeksowana wersja Twojej strony, która rzekomo zawiera poprawną treść, jest przechowywana w indeksie. Jest to bardzo ważne,

ponieważ jeśli kod statusu 500 nie zostanie użyty prawidłowo, cała witryna może zostać wymazana z indeksu wyszukiwarki, zastępując wszystkie strony tekstem widocznym na rysunku.

Uwaga: przed przejściem do następnego ćwiczenia, upewnij się, że ustawiłeś stałą DEBUGGING z powrotem na true, aby TShirtShop pokazywał dane debugowania, gdy wystąpi błąd, zamiast wyrzucać stronę 500. Usuń także odniesienie do inexistent_file.php.

Ćwiczenie: Używanie kodu stanu HTTP 404

1. Zmodyfikuj metodę CheckRequest() w Presentation/link.php, dodając podświetlony kod:

```
/* Remove the virtual location from the requested URL
so we can compare paths */
$requested_url = self::Build(str_replace(VIRTUAL_LOCATION, '',
$_SERVER['REQUEST_URI']));
// 404 redirect if the requested product, category or department
// doesn't exist
if (strstr($proper_url, '/-'))
{
// Clean output buffer
ob_clean();
// Load the 404 page
include '404.php';
// Clear the output buffer and stop execution
flush();
ob_flush();
ob_end_clean();
exit();
}
// 301 redirect to the proper URL if necessary
if ($requested_url != $proper_url)
{
```

2. Otwórz Presentation/products_list.php i dodaj następujący kod do funkcji init():

```
elseif (isset($this->_mDepartmentId))
$this->mProductListPages[] =
Link::ToDepartment($this->_mDepartmentId, $i);
```

```
else
$this->mProductListPages[] = Link::ToIndex($i);
}
/* 404 redirect if the page number is larger than
the total number of pages */
if ($this->mPage > $this->mrTotalPages)
{
// Clean output buffer
ob_clean();
// Load the 404 page
include '404.php';
// Clear the output buffer and stop execution
flush();
ob_flush();
ob_end_clean();
exit();
}
// Build links for product details pages
for ($i = 0; $i < count($this->mProducts); $i++)
{
```

3. W folderze tshirtshop utwórz plik o nazwie 404.php i wpisz następujący kod:

```
<?php
// Set the 404 status code
header('HTTP/1.0 404 Not Found');
require_once 'include/config.php';
require_once PRESENTATION_DIR . 'link.php';
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
```

```
<title>
TShirtShop Page Not Found (404): Demo Product Catalog from
Beginning PHP and MySQL E-Commerce
</title>
<link href="<?php echo Link::Build('styles/tshirtshop.css'); ?>"
type="text/css" rel="stylesheet" />
</head>
<body>
<div id="doc" class="yui-t7">
<div id="bd">
<div id="header" class="yui-g">
<a href="<?php echo Link::Build(''); ?>">
<img src="<?php echo Link::Build('images/tshirtshop.png'); ?>"
alt="tshirtshop logo" />
</a>
</div>
<div id="contents" class="yui-g">
<h1>
The page that you have requested doesn't exist on TShirtShop.
</h1>
<p>
Please visit the
<a href="<?php echo Link::Build(''); ?>">TShirtShop catalog</a>
if you're looking for T-shirts,
or <a href="<?php echo ADMIN_ERROR_MAIL; ?>">email us</a>
if you need further assistance.
</p>
<p>Thank you!</p>
<p>The TShirtShop team.</p>
</div>
</div>
```

</body>

</html>

4. Zmodyfikuj .htaccess, dodając ten podświetlony kod:

Set the default 500 page for Apache errors

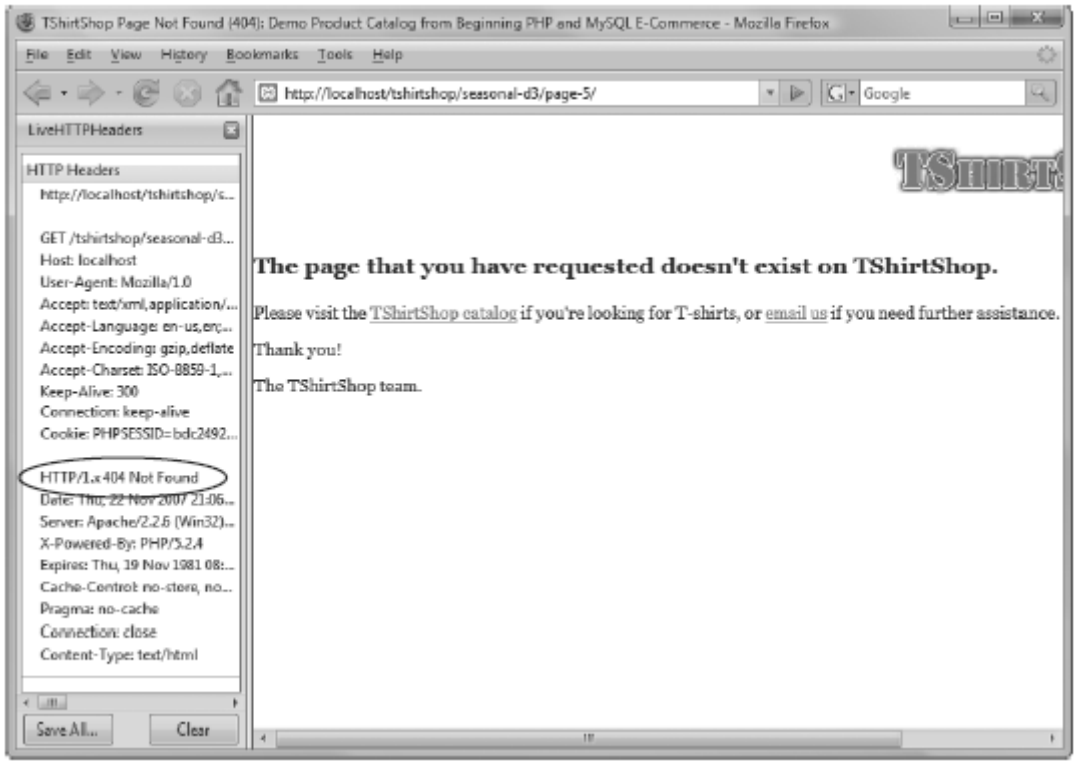
ErrorDocument 500 /tshirtshop/500.php

Set the default 404 page

ErrorDocument 404 /tshirtshop/404.php

Uwaga: upewnij się, że są to prawidłowe lokalizacje plików 404.php i 500.php.

5. Załaduj http://localhost/tshirtshop/seasonal-d3/page-5/. Ponieważ dział sezonowy ma tylko cztery strony produktów, TShirtShop powinien wyświetlić stronę 404, jak pokazano na rysunku.



Jak to działa: 404 i 500

W tym i poprzednim ćwiczeniu nauczyłeś się pracować z kodami stanu 404 i 500 przy użyciu pliku konfiguracyjnego .htaccess oraz kodu PHP. W przypadku 404 użyteczność obu technik jest bardziej oczywista. Jeśli użytkownik zażąda strony, która nie pasuje do żadnej istniejącej lokalizacji Twojej witryny, Apache użyje strony 404 skonfigurowanej w .htaccess. Jeśli jednak użytkownik zażąda strony technicznie poprawnej, ale takiej, której zawartość nie istnieje, na przykład podstrony kategorii, której wartość Page jest większa niż największa istniejąca strona, musimy sami wyrzucić stronę 404 za pomocą kodu PHP. Aby przetestować pierwszy scenariusz, po prostu załaduj stronę, taką jak

http://localhost/tshirtshop/does_not_exist.php. Drugi scenariusz został przetestowany w ostatnim kroku ćwiczenia, a wyniki przedstawiono na rysunku powyżej.

Podsumowanie

Jesteśmy pewni, że podobała Ci się ta część! Dzięki zaledwie kilku zmianom w kodzie, TShirtShop jest teraz gotowy do stawienia czoła konkurencji online, z solidnym fundamentem zoptymalizowanym pod kątem wyszukiwarek. Oczywiście wysiłki związane z optymalizacją wyszukiwarek na tym się nie kończą. Dodając każdą nową funkcję strony internetowej, będziemy przestrzegać ogólnych wytycznych SEO, więc gdy uruchomimy stronę internetową, wyszukiwarki będą naszymi przyjaciółmi, a nie wrogami. W kolejnych rozdziałach będziemy kontynuować wprowadzanie drobnych ulepszeń SEO. Na razie położono fundamenty i jesteśmy gotowi do dalszego wdrażania kolejnej ekscytującej funkcji w TShirtShop: wyszukiwania produktów!