Zalecenia dotyczące produktów

Jedną z najważniejszych zalet sklepu internetowego w porównaniu do tradycyjnej lokalizacji jest możliwość dostosowania strony internetowej dla każdego odwiedzającego w oparciu o jego preferencje lub preferencje oparte na danych zebranych od podobnych odwiedzających. Jeśli Twoja witryna internetowa wie, jak w sprytny sposób sugerować odwiedzającym dodatkowe produkty, mogą oni kupić więcej niż początkowo planowali. Niewątpliwie widziałeś już tę strategię w działaniu na wielu udanych witrynach e-commerce i jest ku temu powód - zwiększa zyski. Tu wdrożysz prosty, ale skuteczny system rekomendacji produktów w swoim sklepie internetowym TShirtShop.

• Dodaj rekomendacje produktów do stron szczegółów produktu. Rekomendacje te będą promować dodatkowe produkty, które zostały zamówione razem z danym produktem.

• Dodaj rekomendacje produktów do stron koszyka. Rekomendacje te będą promować produkty, które zostały zamówione razem z produktami w koszyku.

Wdrożenie tych funkcji nie jest szczególnie trudne, ale znalezienie idealnego mechanizmu rekomendacji jest zawsze ciekawym zadaniem.

Zwiększanie sprzedaży dzięki dynamicznym rekomendacjom

System rekomendacji produktów możesz wdrożyć na kilka sposobów, w zależności od rodzaju sklepu. Oto kilka popularnych:

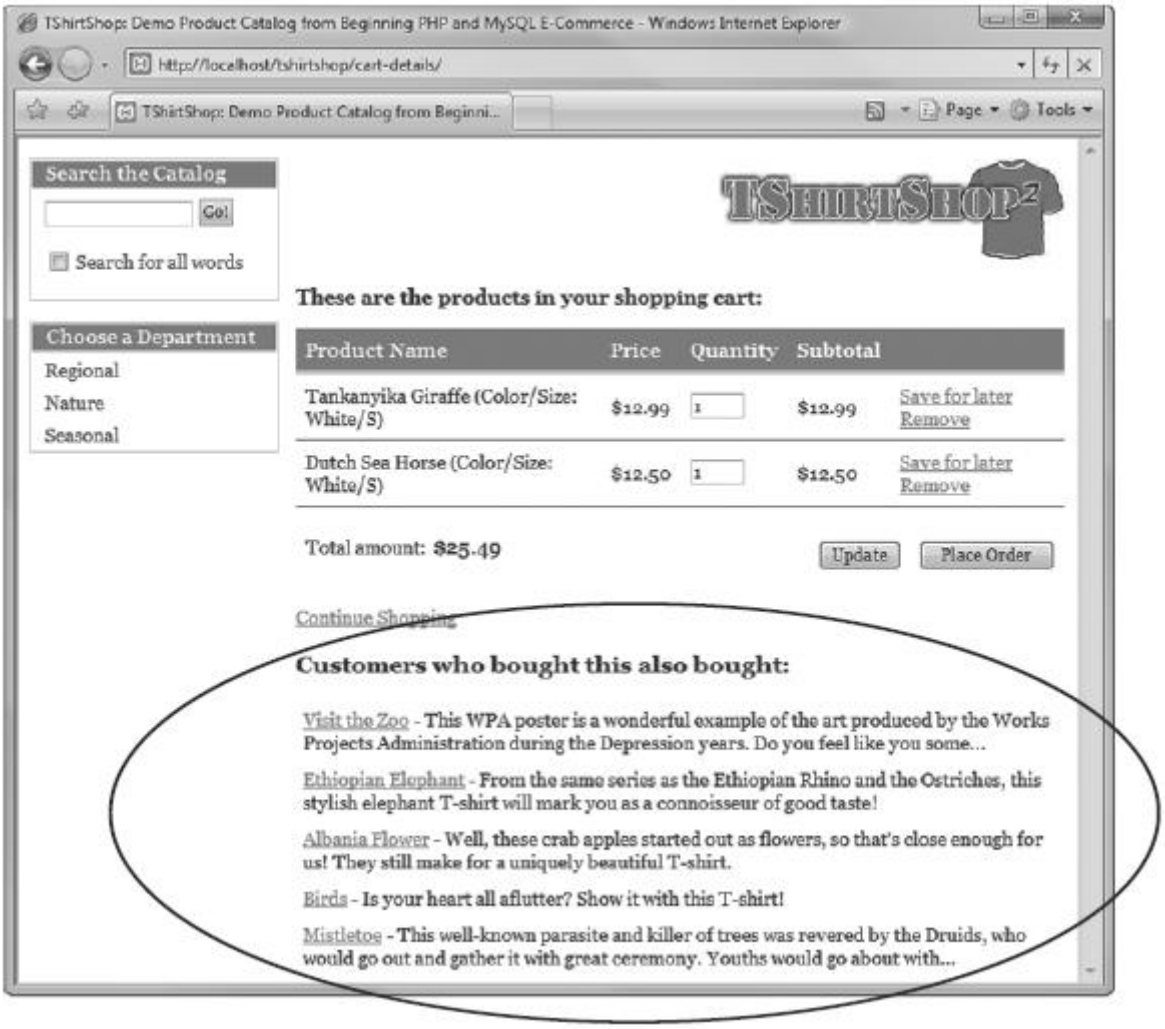
Up-Selling: Up-selling to strategia oferowania konsumentom możliwości zakupu uaktualnienia lub czegoś dodatkowego w zależności od żądanego zakupu. Być może najsłynniejszy przykład up-sellingu – „Czy chciałbyś to zwiększyć?" - jest wymieniana klientom, gdy zamawiają posiłek w McDonald's. To pozornie niewinne pytanie znacznie zwiększa marżę firmy.

Cross-Selling: Cross-selling to praktyka oferowania klientom produktów komplementarnych. Kontynuując analogię z McDonaldem, kiedy ktoś zamawia hamburgera, zawsze usłyszysz zdanie: „Czy chciałbyś do tego frytki?" Ponieważ powszechnie wiadomo, że frytki pasują do burgerów, a konsument zamawia burgera, jest prawdopodobne, że konsument również lubi frytki – sama wzmianka o frytkach może wygenerować nową sprzedaż.

Wyróżnione produkty na stronie głównej: TShirtShop umożliwia już administratorowi witryny wybór produktów prez

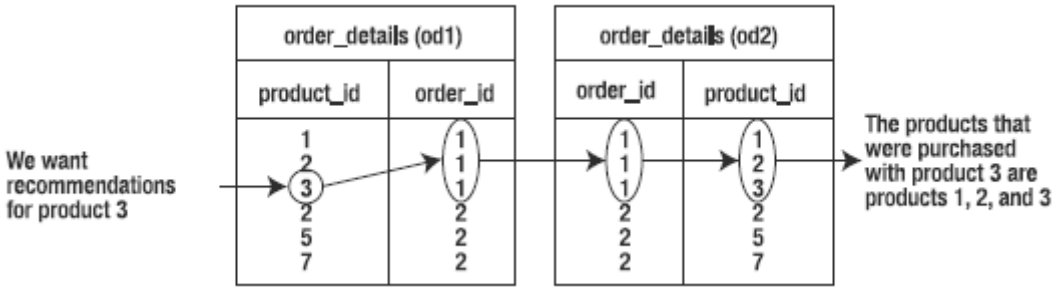
Tu wdrożysz dynamiczny system rekomendacji obejmujący zarówno strategie up-sellingowe, jak i cross-sellingowe. Ponieważ w tym momencie TShirtShop zachowuje, które produkty zostały sprzedane, zaimplementujesz „klienci, którzy kupili ten produkt kupili również . . ". w tym rozdziale. Ten system ma tę zaletę, że nie wymaga ręcznej konserwacji. Nasza strona automatycznie pomoże nam zwiększyć nasze zyski bez dalszej interwencji! Jak wspomniano wcześniej, wdrożymy system dynamicznych rekomendacji w koszyku odwiedzającego oraz na stronie szczegółów produktu. Po dodaniu nowych bitów do sklepu, strona szczegółów produktu będzie zawierać listę rekomendacji produktów na dole strony, jak pokazano na rysunku .

Strona koszyka na zakupy otrzymuje podobny dodatek, jak pokazano na rysunku.



Wybór rekomendacji z bazy danych

Wyzwanie techniczne związane z implementacją nowej funkcji polega na pisaniu zapytań do bazy danych, które zwracają rekomendacje produktów. Pisanie zapytania SELECT, które wyszukuje produkty, które zostały zamówione razem z innym produktem (w celu wygenerowania rekomendacji produktowych) lub z innym zestawem produktów (w celu wygenerowania rekomendacji koszyka), nie jest zbyt skomplikowane ale też nie trywialne. Zacznijmy od rekomendacji produktów. Musisz dowiedzieć się, jakie inne produkty kupili klienci, którzy kupili produkt, dla którego wyliczasz rekomendacje (innymi słowy, określ informację „klient, który kupił ten produkt, kupił również…"). Nasza tabela order_detail zawiera dane dla każdego z naszych zamówień. Aby określić, jakie inne produkty zostały zamówione razem z konkretnym produktem, połączylibyśmy dwa wystąpienia tabeli order_detail w ich polach order_id. Łączenie wielu instancji jednej tabeli jest jak łączenie niezależnych tabel danych zawierających te same dane (przejrzyj sekcję „Łączenie tabel danych" w rozdziale 5, aby szybko przypomnieć sobie temat łączeń tabel). Łączymy dwa wystąpienia order_detail — które nazwaliśmy od1 i od2 — w ich polach order_id i filtrujemy według wartości product_id w od1 dla produktu, którego szukamy. W ten sposób po stronie od2 relacji będziemy mieć wszystkie produkty, które zostały zamówione w zamówieniach, które zawierają również produkt, który filtrujemy. Aby łatwiej to zrozumieć, spójrz na diagram na rysunku.



Na diagramie mamy dwa zamówienia (1 i 2), z których każde ma trzy produkty (odpowiednio 1, 2, 3 i 2, 5, 7). Śledząc zależność między od1 i od2 widać, że dość łatwo jest znaleźć, jakie produkty zostały zamówione z konkretnym produktem.

Zobaczmy, jak możemy zaimplementować tę relację z kodem SQL. Poniższe zapytanie pobiera wszystkie produkty, które zostały zamówione razem z produktem zidentyfikowanym przez product_id równy 4:

```
SELECT od2.product_id

FROM order_detail od1

JOIN order_detail od2

ON od1.order_id = od2.order_id

WHERE od1.product_id = 4;
```

Ten kod zwraca długą listę produktów, która obejmuje produkt o identyfikatorze produktu równym 4, na przykład ten:

```
product_id
----------
         4
         5
        10
        43
         4
         5
        10
        23
        25
        28
         4
        10
        12
        14
        43
```

Ten wynik jest dobry na początek, ale nie jest idealny. Po pierwsze, pomocne byłoby posortowanie listy produktów według częstotliwości zamawiania produktów wraz z naszym produktem. Im więcej razy dwa produkty są zamawiane razem, tym lepsza będzie rekomendacja sprzedaży krzyżowej. Za chwilę pokażemy, jak sobie z tym poradzić. Po drugie, lista nie powinna zawierać produktu, dla którego wyliczamy rekomendacje; zostało już zamówione. Ten problem jest prosty do rozwiązania, dodając jeszcze jedną regułę do klauzuli WHERE:

WHERE clause:

```
SELECT od2.product_id

FROM order_detail od1

JOIN order_detail od2

ON od1.order_id = od2.order_id

WHERE od1.product_id = 4 AND od2.product_id != 4;
```

Nic dziwnego, że wykonując to zapytanie otrzymujesz listę produktów podobną do poprzedniej, z tą różnicą, że nie zawiera już produktu o identyfikatorze product_id równym 4:

```
product_id
----------
         5
        10
        43
         5
        10
        23
        25
        28
        10
        12
        14
        43
```

Teraz lista zwracanych produktów jest znacznie krótsza, ale zawiera wiele wpisów dla kilku produktów (produkty, które zostały zamówione w kilku zamówieniach, które również zawierają identyfikator produktu 4). Aby uzyskać najtrafniejsze rekomendacje, musimy dowiedzieć się, które produkty pojawiają się na tej liście częściej. Robimy to, grupując wyniki poprzedniego zapytania według identyfikatora produktu i sortując w porządku malejącym według tego, ile razy każdy produkt pojawia się na liście (liczba ta jest podana w kolumnie obliczonej rangi w poniższym fragmencie kodu):

```
SELECT od2.product_id, COUNT(od2.product_id) AS rank

FROM order_detail od1

JOIN order_detail od2

ON od1.order_id = od2.order_id

WHERE od1.product_id = 4 AND od2.product_id != 4

GROUP BY od2.product_id
```

ORDER BY rank DESC;

* **Wskazówka** : Umieszczenie spacji między COUNT a następującym wyrażeniem powoduje, że MySQL generuje błąd, więc bądź ostrożny.

Zapytanie zwraca teraz listę z produktami o najwyższym rankingu wymienionymi jako pierwsze:

product_id	rank
10	3
5	2
43	2
23	1
25	1
28	1
12	1
14	1

Jeśli nie potrzebujesz widzieć rangi, możesz przepisać to zapytanie za pomocą funkcji agregującej COUNT bezpośrednio w klauzuli ORDER BY. Możesz również użyć słowa kluczowego LIMIT, aby określić, ile rekordów Cię interesuje. Jeśli chcesz otrzymać pięć pierwszych produktów z listy, to zapytanie załatwia sprawę:

```
SELECT od2.product_id

FROM order_detail od1

JOIN order_detail od2

ON od1.order_id = od2.order_id

WHERE od1.product_id = 4 AND od2.product_id != 4

GROUP BY od2.product_id

ORDER BY COUNT(od2.product_id) DESC

LIMIT 5;
```

Wyniki tego zapytania to

product_id
10
5
43
23
25

* **Uwaga** : Pamiętaj, że możesz uzyskać różne wyniki, jeśli spróbujesz tego zapytania przy użyciu swojej bazy danych, nawet jeśli masz te same dane, co my. Jeśli kryteria sortowania określone za pomocą funkcji ORDER BY nie są wystarczająco szczegółowe, aparat bazy danych wybiera najwygodniejszą dla siebie metodę sortowania, co w rzeczywistości oznacza, że otrzymujesz rekordy, które są najszybsze i najłatwiejsze do pobrania przez bazę danych. Na przykład, jeśli chodzi o poprzednie zapytanie, widać, że istnieje pięć identyfikatorów produktów, dla których ranga wynosi 1 (12, 14, 23, 25, 28). Kiedy

wykonaliśmy zapytanie w naszej testowej bazie danych, otrzymaliśmy 23 i 25, ale Twoja baza danych może „preferować” inne produkty. Aby uzyskać rekomendacje produktowe, jedynym kryterium, które nas interesuje, jest wyliczana przez nas ranga; gdy mamy wiele produktów o tej samej randze, można pozwolić, aby baza wybrała za nas.

Ponieważ ta lista liczb nie ma większego sensu dla ludzkiego oka, warto również poznać nazwę i opis polecanych produktów. Poniższe zapytanie wyodrębnia nazwy, łącząc listę zalecanych identyfikatorów produktów z tabelą produktów:

```
SELECT od2.product_id, od2.product_name

FROM order_detail od1

JOIN order_detail od2 ON od1.order_id = od2.order_id

JOIN product p ON od2.product_id = p.product_id

WHERE od1.product_id = 4 AND od2.product_id != 4

GROUP BY od2.product_id

ORDER BY COUNT(od2.product_id) DESC

LIMIT 5;
```

Na podstawie danych z poprzednich fikcyjnych wyników to zapytanie zwraca coś takiego:

```
product_id    name
----------    ----------------------
        10    Haute Couture
         5    Marianne
        43    Equatorial Rhino
        23    Italian Airmail
        25    Romulus & Remus
```

Alternatywnie możesz chcieć obliczyć rekomendacje produktów tylko na podstawie danych z zamówień złożonych w ciągu ostatnich n dni. W tym celu potrzebne jest dodatkowe sprzężenie z tabelą zamówień, która zawiera pole created_on. Poniższe zapytanie oblicza rekomendacje produktów na podstawie zamówień złożonych w ciągu ostatnich 30 dni:

```
SELECT od2.product_id, od2.product_name

FROM order_detail od1

JOIN order_detail od2 ON od1.order_id = od2.order_id

JOIN orders o ON od1.order_id = o.order_id

WHERE od1.product_id = 4 AND od2.product_id != 4

AND DATE_SUB(NOW(), INTERVAL 30 DAY) < o.created_on

GROUP BY od2.product_id

ORDER BY COUNT(od2.product_id) DESC

LIMIT 5;
```

* **Uwaga** : W tej chwili nie będziemy używać tej sztuczki w TShirtShop, ale ważne jest, aby o tym pamiętać. Z biegiem czasu, w miarę jak sprzedajesz coraz więcej za pośrednictwem swojej strony, będziesz musiał ograniczyć liczbę zamówień, które są wykorzystywane do obliczania rekomendacji - w przeciwnym razie Twoje rekomendacje będą nie tylko datowane, ale będą obliczane na coraz większych zestawach danych.

Taka sytuacja zmniejsza skuteczność rekomendacji i odbija się na wydajności witryny. W końcu czy naprawdę chcemy polecić świąteczne koszulki w maju? A jak Twoi kupujący będą się czuć, że musieli czekać, aż Twój serwer będzie łamał się na przestarzałych rekomendacjach? Można się założyć, że zaczną całkowicie ignorować twoje zalecenia, jeśli są zbyt często nieistotne. Jeśli zdecydujesz się skorzystać z tej techniki, dodanie indeksu w polu created_on zwiększy wydajność zapytania podczas uzyskiwania rekomendacji koszyka na zakupy stosujemy w zasadzie tę samą technikę, co w przypadku rekomendacji pojedynczych produktów, z wyjątkiem tego, że dodajemy składnik koszyka na zakupy. Zamiast pozyskiwać produkty, które zostały zamówione razem z innym produktem, otrzymujemy produkty, które zostały zamówione razem z dowolnym produktem z koszyka. Jest to rzeczywiście bardziej złożona kalkulacja, ale zwrot z inwestycji czasowej jest tego wart dla potencjalnej dodatkowej sprzedaży! Ostatnią rzeczą, na którą należy zwrócić uwagę przed napisaniem kodu, jest to, że nasza proponowana implementacja jest jedną z wielu możliwych. Generalnie w programowaniu, a szczególnie w świecie SQL, rezultat można osiągnąć na wiele sposobów. Zapytanie zwracające rekomendacje produktów można przepisać na wiele sposobów, z których najbardziej oczywistym jest użycie podzapytań zamiast łączenia tabel. Wdrażamy logikę warstwy danych, którą właśnie wyjaśniliśmy, w dwóch procedurach składowanych: catalog_get_recommendations i shopping_cart_get_recommendations. Oprócz zapytań przedstawionych wcześniej, procedury składowane będą również pobierać nazwy i opisy produktów, które należy wyświetlić odwiedzającemu.

Ćwiczenie: Dodawanie procedur przechowywanych rekomendacji produktów

1. Użyj phpMyAdmin, aby utworzyć procedurę składowaną opisaną w poniższych krokach. Nie zapomnij ustawić ogranicznika $$ przed wykonaniem kodu na każdym kroku.

2. Wykonaj ten kod, który utworzy procedurę składowaną catalog_get_recommendations w bazie danych sklepu tshirtshop:

```
-- Create catalog_get_recommendations stored procedure

CREATE PROCEDURE catalog_get_recommendations(

IN inProductId INT, IN inShortProductDescriptionLength INT)

BEGIN

PREPARE statement FROM

"SELECT od2.product_id, od2.product_name,

IF(LENGTH(p.description) <= ?, p.description,

CONCAT(LEFT(p.description, ?), '...')) AS description

FROM order_detail od1

JOIN order_detail od2 ON od1.order_id = od2.order_id

JOIN product p ON od2.product_id = p.product_id
```

```
WHERE od1.product_id = ? AND

od2.product_id != ?

GROUP BY od2.product_id

ORDER BY COUNT(od2.product_id) DESC

LIMIT 5";

SET @p1 = inShortProductDescriptionLength;

SET @p2 = inProductId;

EXECUTE statement USING @p1, @p1, @p2, @p2;

END$$
```

3. Wykonaj następujący kod, który utworzy procedurę przechowywaną shopping_cart_get_recommendations w Twojej bazie danych tshirtshop:

```
-- Create shopping_cart_get_recommendations stored procedure

CREATE PROCEDURE shopping_cart_get_recommendations(

IN inCartId CHAR(32), IN inShortProductDescriptionLength INT)

BEGIN

PREPARE statement FROM

"-- Returns the products that exist in a list of orders

SELECT od1.product_id, od1.product_name,

IF(LENGTH(p.description) <= ?, p.description,

CONCAT(LEFT(p.description, ?), '...')) AS description

FROM order_detail od1

JOIN order_detail od2

ON od1.order_id = od2.order_id

JOIN product p

ON od1.product_id = p.product_id

JOIN shopping_cart

ON od2.product_id = shopping_cart.product_id

WHERE shopping_cart.cart_id = ?

-- Must not include products that already exist

-- in the visitor's cart

AND od1.product_id NOT IN
```

```
(-- Returns the products in the specified

-- shopping cart

SELECT product_id

FROM shopping_cart

WHERE cart_id = ?)

-- Group the product_id so we can calculate the rank

GROUP BY od1.product_id

-- Order descending by rankORDER BY COUNT(od1.product_id) DESC

LIMIT 5";

SET @p1 = inShortProductDescriptionLength;

SET @p2 = inCartId;

EXECUTE statement USING @p1, @p1, @p2, @p2;

END$$
```

Jak to działa: Pobieranie rekomendacji produktów z bazy danych

Procedury składowane, które właśnie napisałeś, stosują po prostu techniki, których nauczyłeś się w pierwszej części rozdziału. W innych wydaniach tej książki (wykorzystujących PostgreSQL i SQL Server) w tym miejscu przedstawiliśmy również alternatywne implementacje tych procedur składowanych, które wykorzystywały podzapytania zamiast łączenia tabel w celu uzyskania niezbędnych danych. W MySQL użycie podzapytań nie jest możliwe ze względu na pewne ograniczenia używania LIMITu, które są udokumentowane pod adresem http://dev.mysql.com/doc/refman/5.1/en/subquery-errors.html. Jeśli nowsze wersje MySQL rozwiążą te problemy, gdy wydajność stanie się problemem, możesz rozważyć wykonanie testów przy użyciu przepisanych wersji tych procedur składowanych. W niektórych przypadkach wersje korzystające z podzapytań oferują lepszą wydajność.

Wdrażanie rekomendacji dotyczących produktów i koszyka

W poniższym ćwiczeniu napiszesz kod dla warstw biznesowych i prezentacyjnych systemu rekomendacji produktów. Warstwa biznesowa systemu rekomendacji produktów składa się z dwóch metod o nazwie GetRecommendations. Jeden z nich znajduje się w klasie Catalog i pobiera rekomendacje dla strony szczegółów produktu, a drugi znajduje się w klasie ShoppingCart i pobiera rekomendacje do wyświetlenia w koszyku odwiedzającego.

Ćwiczenie: Wdrażanie zaleceń dotyczących produktów i koszyka

1. Dodaj następującą metodę do klasy Catalog w pliku business/catalog.php:

```
// Get product recommendations

public static function GetRecommendations($productId)

{

// Build the SQL query
```

```
$sql = 'CALL catalog_get_recommendations(
:product_id, :short_product_description_length)';
// Build the parameters array
$params = array (':product_id' => $productId,
':short_product_description_length' =>
SHORT_PRODUCT_DESCRIPTION_LENGTH);
// Execute the query and return the results
return DatabaseHandler::GetAll($sql, $params);
}
```

2. Otwórz plik shopping_cart.php znajdujący się w folderze biznesowym i dodaj następujący kod:

```
// Get product recommendations for the shopping cart
public static function GetRecommendations()
{
// Build the SQL query
$sql = 'CALL shopping_cart_get_recommendations(
:cart_id, :short_product_description_length)';
// Build the parameters array
$params = array (':cart_id' => self::GetCartId(),
':short_product_description_length' =>
SHORT_PRODUCT_DESCRIPTION_LENGTH);
// Execute the query and return the results
return DatabaseHandler::GetAll($sql, $params);
}
```

3. Teraz aktualizujemy warstwę prezentacji, modyfikując szablony składowe produktu i cart_details, aby wyświetlić rekomendacje produktów. Otwórz plik Presentation/product.php i dodaj członka o nazwie $mRecommendations do klasy Product:

```
// Public variables to be used in Smarty template
public $mProduct;
public $mProductLocations;
public $mLinkToContinueShopping;
public $mLocations;
public $mEditActionTarget;
```

public $mShowEditButton;

public $mRecommendations;

4. Następnie musisz pobrać dane rekomendowanych produktów w $mRecommendations i utworzyć linki do ich stron produktów. Zmodyfikuj metodę init() klasy Product, jak pokazano tutaj:

$this->mLocations = Catalog::GetProductLocations($this->_mProductId);

// Create the Add to Cart link

$this->mProduct['link_to_add_product'] =

Link::ToCart(ADD_PRODUCT, $this->_mProductId);

// Get product recommendations

$this->mRecommendations =

Catalog::GetRecommendations($this->_mProductId);

// Create recommended product links

for ($i = 0; $i < count($this->mRecommendations); $i++)

$this->mRecommendations[$i]['link_to_product'] =

Link::ToProduct($this->mRecommendations[$i]['product_id']);

// Build links for product departments and categories pages

for ($i = 0; $i < count($this->mLocations); $i++)

{

5. Aby zakończyć wdrażanie systemu rekomendacji dla strony szczegółów produktu, musimy zaktualizować szablon product.tpl. Dodaj następujące wiersze na końcu prezentacji/templates/product.tpl. Następnie możesz załadować TShirtShop i załadować stronę szczegółów produktu. Jeśli ten produkt został zamówiony z innymi produktami, pojawią się na liście rekomendacji.

{if $obj->mRecommendations}

<h2>Customers who bought this also bought:</h2>

{section name=m loop=$obj->mRecommendations}

{strip}

mRecommendations[m].link_to_product}">

{$obj->mRecommendations[m].product_name}

{/strip}

 - {$obj->mRecommendations[m].description}

{/section}

{/if}

6. Otwórz tshirtshop.css i dodaj następujący styl:

.list { color: #000000; }

7. Teraz zmodyfikujmy szablon z komponentami cart_details, aby wyświetlać rekomendacje produktów na stronie koszyka. Otwórz cart_details.php znajdujący się w folderze prezentacji, aby dodać element $mRecommendation do klasy CartDetails:

<?php

// Class that deals with managing the shopping cart

class CartDetails

{

// Public variables available in smarty template

public $mCartProducts;

public $mSavedCartProducts;

public $mTotalAmount;

public $mIsCartNowEmpty = 0; // Is the shopping cart empty?

public $mIsCartLaterEmpty = 0; // Is the 'saved for later' list empty?

public $mLinkToContinueShopping;

public $mUpdateCartTarget;

public $mRecommendations;

8. Następnie musisz pobrać dane rekomendowanych produktów w $mRecommendations i utworzyć linki do ich stron produktów. Dodaj podświetlony fragment kodu na końcu metody init():

for ($i = 0; $i < count($this->mSavedCartProducts); $i++)

{

$this->mSavedCartProducts[$i]['move'] =

Link::ToCart(MOVE_PRODUCT_TO_CART,

$this->mSavedCartProducts[$i]['item_id']);

$this->mSavedCartProducts[$i]['remove'] =

Link::ToCart(REMOVE_PRODUCT,

```
$this->mSavedCartProducts[$i]['item_id']);
}
// Get product recommendations for the shopping cart
$this->mRecommendations =
ShoppingCart::GetRecommendations();
// Create recommended product links
for ($i = 0; $i < count($this->mRecommendations); $i++)
$this->mRecommendations[$i]['link_to_product'] =
Link::ToProduct($this->mRecommendations[$i]['product_id']);
}
}
?>
```

9. Na koniec zaktualizuj szablon cart_details, aby wyświetlić listę rekomendacji. Dodaj następujące wiersze na końcu prezentacji/templates/cart_details.tpl:

```
{if $obj->mRecommendations}
<h2>Customers who bought this also bought:</h2>
<ol>
{section name=m loop=$obj->mRecommendations}
<li>
{strip}
<a href="{$obj->mRecommendations[m].link_to_product}">
{$obj->mRecommendations[m].product_name}
</a>
{/strip}
<span class="list"> - {$obj->mRecommendations[m].description}</span>
</li>
{/section}
</ol>
{/if}
```

10. Załaduj TShirtShop, złóż zamówienia, a następnie sprawdź strony szczegółów produktu i koszyka; wyświetlają rekomendacje na podstawie zamówionych produktów! Wyniki powinny wyglądać tak, jak na rysunkach 15-1 i 15-2 pokazanych wcześniej w tym rozdziale.

* **Uwaga** :Pamiętaj, aby złożyć niektóre zamówienia testowe, które zawierają wiele produktów, aby zapewnić mechanizmowi rekomendacji pewne dane do pracy.

Podsumowanie

Pokazywanie rekomendacji produktów to świetny sposób na zachęcenie do sprzedaży i udało nam się wdrożyć tę funkcjonalność w tym krótkim rozdziale. Największym wyzwaniem było zbudowanie zapytania SQL, które pobiera listę rekomendowanych produktów, a my krok po kroku przeanalizowaliśmy sposób jej tworzenia. Nasza strona jest gotowa do zwiększenia sprzedaży bez dalszych wysiłków z naszej strony! W następnym rozdziale wejdziemy w III fazę rozwoju, dodając kolejną usługę dla naszych klientów (i nas) - funkcjonalność kont klientów.