Voorstel: Birkenstockshop

Birkenstockshop koppeling

AUTEUR: ***NAV APPAIYA - NOTIVE***
DATUM: 10/03/2015
VERSIE: ***1.0.0***

Inleiding

De BIRKENSTOCK-STORE Magento webshop wordt omgezet naar het SEOshop platform.
Om dit mogelijk te maken moet o.a. de koppeling tussen Magento en het kassa systeem opnieuw worden gebouwd. Hiervoor is Notive benaderd, waar ShopMonkey de SEOshop opbouwt, zal Notive zich focussen op de koppeling.

De producten kunnen geïmporteerd worden op basis van naam. Onder elk product hangen de diverse varianten zoals kinder of dames. Deze varianten kunnen ieder hun eigen prijs, sku en allerlei extra informatie bevatten.

Daarnaast kunnen we het afhandelen zoals het nu in de Magento webshop is, per variant een eigen SKU.

Het project kan worden onderverdeeld in 4 delen:
- Uitlezen en parsen XML
- Doorzetten data SEOShop API
- Data uitlezen vanuit SEOShop
- Nieuwsbrief

Voorstel

Probleem	Oplossing
Uitlezen en parsen XML	Dataset / structuur uit het kassasysteem nog te ontvangen
Doorzetten data SEOShop API	Via de rest API van Seoshop producten aan kunnen maken. Met behulp van Guzzle een POST request maken naar /products.json met elk product.
Data uitlezen vanuit SEOShop	Nog geen details over ontvangen
Nieuwsbrief	Nog geen details over

Stappenplan:

1. Abstracte Seoshop model maken voor de Notive SeoShopBundle

Als uitbreiding op de Notive seoshopbundle moet er een model geschreven worden die een seoshop presenteerd. Deze is dan vervolgens te extenden door andere entiteiten/classes in andere bundels.

Een aantal eisen aan dit model:
- Moet een abstract model zijn (te extenden en gebruiken in onze andere bundles)
- Moet alleen de nodige (generieke) attributen bevatten die bij een seoshop horen (dus geen specificaties
- Moet zowel voor partner apps als voor api_keys bruikbaar zijn.

Vervolgens kunnen we dit model gebruiken in onze appbundle onder entities door dit model te extenden.

2. Entity product maken

Een entity aanmaken aan hand van de benodigde velden voor de SeoShop API. De entiteit zou alle velden moeten bevatten die we naar seoshop gaan synchroniseren via de API. Hier gaan we de producten uit de xml dataset in importeren.

3. Uitlezen en parsen XML data in ons producten tabel
Een generiek product moet gevuld worden met de XML data vanuit het kassasysteem.

4. Doorzetten data SEOShop API
In het productmodel voegen we een extra kolom toe waar we het product_id van seoshop opslaan. Dit krijgen we terug bij het aanmaken van een product. Indien deze is ingevuld, betekend dat het product al is gesynchroniseerd met Seoshop.

1. Schrijven van een controller die alle producten ophaalt zonder seoshop product id
2. Product in een create product request naar seoshop verwerken
3. Afvangen van het nieuwe seoshop product id en opslaan bij het gesynchroniseerd product.

5. Controller inrichten om webhooks te installeren
Om webhooks te installeren voor een webshop moet er een controller ingericht worden. Deze controller zorgt ervoor dat we handmatig webhooks kunnen installeren voor een specifieke shop. De controller neemt een value (shopId) als parameter en zal met bijbehorende shop-keys de webhook installeren. We installeren een webhook onder Orders op het moment van bestellen (order created)

Seoshop app vs api keys
Bij de overweging om te kiezen voor App_keys of Api_keys is het voor deze situatie aannemelijker om te kiezen voor api_keys. Producten voorraad word gebruikt door meerdere seoshops dus de producten kunnen vanuit 1 algemene database gesynchroniseerd worden met SeoShop.