

Pflichten- und Lastenheftabgabe

Autor: Dr. David Urbansky

Version und Datum: 1.3 / 01.12.2014

Ontologie

Die folgenden fünf Typen wurden angelegt:

1. **Sensorische Attribute:** süß, salzig, sauer, scharf und bitter
2. **16 Themenkörbe** (http://www.emmas-enkel.de/index.php?cl=start_shoplist)
3. **Saisonale Attribute:** Sommer, Winter, Frühling, Herbst
4. **Medizinische Attribute:** gegen Fieber, gut für Verdauung... + weitere
5. **Allgemeine Attribute:** locker, fruchtig, herb, deftig...+ weitere

Suchfeldanpassung


Die Suchvorschläge enthalten neben dem Namen und dem Bild vorgeschlagener Produkte zusätzlich noch:


1. Den Preis des Produkts.
2. Eine verstellbare Anzahl des vorgeschlagenen Produkts.
3. Ein Warenkorbformular womit das Produkt in den Warenkorb gelegt werden kann.
4. Einen Link um das Produkt auf den Merkzettel zu verschieben.


Folgender Screenshot zeigt die Vorschläge beispielhaft.


wagner pizza

Produkte

 wagner pizza Steinofen Speciale
€2.5 | Stück: [in den Warenkorb](#) | [Auf den Merkzettel](#)

 wagner Steinofen pizza Salami
€2.5 | Stück: [in den Warenkorb](#) | [Auf den Merkzettel](#)

 wagner Big pizza Hawaii
€2.4 | Stück: [in den Warenkorb](#) | [Auf den Merkzettel](#)

 wagner Big pizza Thunfisch
€2.4 | Stück: [in den Warenkorb](#) | [Auf den Merkzettel](#)

Einbau des Suchfelds

Das Suchfeld kann mit den beiden Dateien **unibox.min.css** und **unibox.min.js** und folgender Weise eingebunden werden. Die beiden Dateien müssen im `<head>` der Seite verlinkt werden. Voraussetzung ist das Vorhandensein von jQuery > 1.5. Der import der Dateien kann folgendermaßen aussehen:

```
<link rel="stylesheet" href="css/unibox.min.css">
<script src="js/jquery.min.js"></script>
<script src="js/unibox.min.js"></script>
```

Am Ende der Seite sollte die Konfiguration des Scriptes und die Initialisierung erfolgen.

```
<script type="text/javascript">
var settings = {
    // Die URL zu den Suchvorschlägen (siehe API Dokumentation).
    suggestUrl: 'http://api.semfox.com/queries/suggest?apiKey=
apiKey=kftruanreiotsdaifaiseapeiorsdafb customerId=6&query=',

    // Falls die Suche Bilder zur Visualisierung nutzt, kann hier eine Überschrift
    // angelegt werden.
    queryVisualizationHeadline: 'Ihre Suche Visualisiert',

    // Der event der passiert, wenn man enter auf einem Vorschlag drückt.
    enterCallback: function(text,link) {console.log(text);},

    // Wo die Bilder der Suchvisualisierung angezeigt werden sollen "top", "bottom", "all"
    // oder "none".
    instantVisualFeedback: 'none',

    // Ob der gefundene Suchstring im Vorschlag hinterlegt werden soll.
    highlight: true,

    // Extra HTML für die Anzeige vom Merkzettel, des Preises, der Anzahl und des
    // Warenkorbes. Hier müssen die URLs für das Formular und den Merkzettel angepasst werden.
    extraHtml: '##price## | Stück: <form action="http://emmasenkel.de/warenkorb.php"
style="display: inline-block;"><input type="hidden" name="pid" value="##articleNumber##">
<input type="number" name="quantity" min="1" max="99" value="1"> <input type="submit"
value="in den Warenkorb"></form> | <a
href="http://emmasenkel.de/merkzettel?product=##articleNumber##">Auf den Merkzettel</a>',

    // Zeit zwischen Tastenanschlägen in Millisekunden, in denen kein Vorschlag aufgerufen
    // werden soll.
    throttleTime: 50,
}

// Initialisieren der Suchbox mit der ID des Suchinputs, z.B. wenn das input <input
// id="searchBox" type="text"> heisst:
$('#searchBox').unibox(settings);
</script>
```

API Dokumentation

Einleitung

Dieses Dokument beschreibt die SEMFOX RESTful API für Emmas-Enkel.de

BASE: <http://api.semfox.com/>

An alle Anfragen muss die "customerId=6" und der "apiKey=kftruanreiotsdaifaiseapeiorsdafb" angefügt werden.

Schnittstellen

Suchvorschläge:

Eingabe (GET): Suchanfrage von Emmas-Enkel Webshop

Ausgabe: Eine JSON-formattierte Ausgabe mit Informationen für das Eingabefeld.

Endpunkt: [BASE + queries/suggest?query=Pizza&limit=10&numSuggests=10](#)

Beispielantwort:

```
{
  "suggests": {
    "Produkte": [
      {
        image: "http://bilder.imwalking.de/pool/iaw_format_da/9089941.jpg",
        name: "ara Sandalette",
        price: "€1.99",
        articleNumber: "AZ-4345-38"
      }
    ],
    "_": [
      {
        "name": "pizza mit salami"
      },
      {
        "name": "bio pizza"
      }
    ]
  },
  "words": [
    {
      "image": "images/cat-PIZZA.png",
      "name": "pizza"
    }
  ]
}
```

Die Anfrageparameter:

query: Die Suchanfrage

limit: Die maximale Anzahl der Produktvorschläge, default: 8

numSuggests: Die maximale Anzahl der Suchanfragenvorschläge, default: 5

Suche:

Eingabe (GET): Suchanfrage von Emmas-Enkel Webshop

Ausgabe: Eine JSON-formattierte Ausgabe mit den Artikelnummern von Emmas-Enkel Produkten.

Die Darstellung und Verknüpfung der gelieferten Informationen ist Aufgabe von Emmas-Enkel.

Endpunkt: [BASE + products?query=Pizza&offset=0&limit=10](#)

Beispielantwort:

```
{
```

```

    expires: 1409646631244,
    searchResults: [
      [
        {
          ean: "4046800110064",
          id: 1429,
          groupId: 1429,
          articleNumber: "3072",
          name: "Mondamin Pizza Teig",
          image: "http://www.emmas-enkel.de/mondamin-pizza-teig-460g.jpg"
        },
      ]
    ],
    interpretedQuery: {
      tags: [ ],
      category: "pizza",
      corrected: "bio pizza",
      explanation: "Wir haben <span class='sx-highlight'>575 Pizzen</span> für Sie
finden können.",
      confidence: 0,
      normalizedCategory: "PIZZA",
      rankingValues: {
        1429: ["EUR 2.99"],
        1430: ["EUR 3.49"],
      },
    },
    processingTime: "418ms"
  }

```

Die Anfrageparameter:

query: Die Suchanfrage

offset: Die zu überspringenden Resultate vom Start.

limit: Die maximale Anzahl der Suchresultate

cache: Ob der Cache genutzt werden darf, default: 1

Die Antwort besteht hauptsächlich aus zwei Blöcken:

1. **searchResults:** Dies ist eine Liste von Listen von Suchresultaten. Meistens ist in jeder Resultatliste nur ein Eintrag, es kann aber sein, dass Produkte gruppiert werden möchten und daher mehrere Einträge unter dem gleichen Suchresultat gelistet werden (Beispielsweise wenn es Kerrygold Butter in 200g und 250g Pack gibt, was aber immernoch die gleichen Produkte sind). Für jedes Suchresultat gibt es Basisdaten die für die Anzeige dienen.
2. **interpretedQuery:** Dieser Block umfasst die Erläuterung wie die Anfrage verstanden wurde. Nicht alle der folgenden Felder sind immer enthalten.
 - tags:** Bestimmte schlüsselworte die erkannt wurden
 - category:** Die Produktkategorie die angefragt wurde
 - normalizedCategory:** Die erkannte Kategorie normalisiert.
 - corrected:** Die korrigierte Anfrage, sollte sich der Nutzer verschrieben haben
 - explanation:** Ein generierter Antwortsatz, der dem Nutzer angezeigt werden kann.
 - confidence:** Ein Wert der beschreibt wie sicher sich SEMFOX ist, dass es die Anfrage gut beantworten konnte. Je höher der Wert, desto besser. Derzeitiges Maximum ist 3
 - rankingValues:** Optional werden auch sog. Ranking values mitgegeben. Dies sind Werte, die sich auf die Anfrage beziehen und für die Anzeige mit den Produkten mit den gegebenen ids verknüpft werden können. Beispielsweise kann „günstige Pizza“ gesucht werden und in ranking values befindet sich der jeweilige Preis zu jedem Suchresultat, verknüpft über die id des Produkts.

Ontologieinformationen abrufen:

Eingabe (GET): Senden einer Artikelnummer eines Produkts

Ausgabe: Angereicherte Informationen (JSON-formattiert) zu dem Produkt. Die Fülle der Informationen hängt direkt und stark von den Eingangsdaten ab.

Endpunkt: [BASE + products/data?articleNumber=#ARTICLE_NUMBER#](#)

ARTICLE_NUMBER ist die Shop-spezifische Artikelnummer des Produkts.

Beispielausgabe

```
{
  Name: "Pizza",
  Preis: "2.99",
  Beschreibung: "super lecker",
  bio: "ja"
}
```

Produkt Update:

Diese Methode stellt sicher, dass auch neue Produkte über die Suche gefunden werden können. Neue oder geänderte Produkte müssen über diese Schnittstelle hinzugefügt werden.

Eingabe (PUT): Ein Produkt mit Stammdaten im Json-Format

Ausgabe: leer

Endpunkt: [BASE + products](#)

Parameter: **productsJsonArray** - json formattierte Liste von Produkten mit Stammdaten, beispielsweise:

```
[
  {
    name: "Pizza",
    ean: "2.99",
    articleNumber: "546786",
    image: "http://emmas-enkel.de/pic1.jpg",
    category: "Tiefkühlkost",

    bio: "ja",
    beschreibung: "Pizzen sind lecker",
    ...
  },...
]
```

Die Felder **name**, **ean**, **articleNumber**, **image** und **category** sind verpflichtend. Alle weiteren Felder können frei benannt werden (am besten deutsch natürlichsprachig wie „Beschreibung“ oder „Bio“).

Produkt Löschung:

Diese Methode erlaubt es Produkte, die nicht mehr verkauft werden aus der Suche auszuschliessen. Eingabe (DELETE): Die Artikelnummer des zu löschenden Produkts
Ausgabe: leer

Endpunkt: [BASE + products?articleNumber=#ARTICLE_NUMBER#](#)

ARTICLE_NUMBER ist die Shop-spezifische Artikelnummer des Produkts.