# So funktioniert's: Affiliate-Daten per API mit dem Content verschmelzen

*Die Nutzung von Affiliate-Programmen und -Netzwerken gehört für viele Webseiten- und Portalbetreiber bereits zum Alltag. Grafisch vorgefertigte Werbemittel, Karusell-Widgets und Deeplink-Generatoren gehören zu den meist genutzten Funktionen. Jedoch bieten die APIs ein größeres (bisher selten genutztes) Potenzial.*

Für viele Webseiten- und Portal-Betreiber stellt das Affiliate-Marketing ein wichtiges Instrument zur Monetarisierung dar. Dabei versucht der Betreiber mittels geeigneter Partner-Programme zu seinem Content passende Werbemittel in die Webseite einzubinden. Hierbei handelt es sich dann oft um standardisierte Formate in Bild- oder Textform. Bei dieser Art der Integration hat der Webseiten-Betreiber mit der s.g. Banner-Blindness (http://en.wikipedia.org/wiki/Banner_blindness) zu kämpfen, d.h. erfahrende Internetnutzer ignorieren diese standardisierten Formate und es kommt somit zu keiner Vergütung.

Erfolgsversprechender scheint die Empfehlung von Produkten in Artikelform zu sein. Dabei wird über das Produkt in einem Text-Review berichtet und an entsprechenden Stellen im Text ein Deeplink gesetzt. Diese Art des Affiliate-Marketings bedeutet jedoch einen hohen redaktionellen Aufwand.

Eine aus meiner Sicht effektivere Form des Affiliate-Marketings ist das Einbinden von Affiliate-Produktdaten über APIs (Application Programming Interface). Dazu bieten einige Affiliate-Anbieten, wie z.B. Amazon, Schnittstellen für verschiedene Programmiersprachen an, um auf die Produktdaten zuzugreifen. Diese Schnittstellen ermöglichen den Zugriff auf wesentlich mehr Produktinformationen und der Detaillierungsgrad wie Produktdaten auf der Webseite angezeigt werden, lässt sich individuell und auch mehrstufig festlegen. Zudem ist der redaktionelle Aufwand sehr gering, da die Daten aktuell über die API ausgelesen werden. Wie das Ganze im Detail aussehen kann, zeige ich am Beispiel einer Buch-Integration.

**Amazon Buchempfehlungen unter Verwendung der Amazon API**

Zunächst wird ein Account beim Amazon-Partner-Programm benötigt. Die Anmeldung dazu erfolgt unter https://partnernet.amazon.de. Nach erfolgreicher Anmeldung muss einer Beantragung zur Freischaltung der Product Advertising API erfolgen (https://partnernet.amazon.de/gp/flex/advertising/api/sign-in.html).

Bevor es nun an die Implementierung geht, sollte man beachten, dass Amazon in der Dokumentation beschreibt, wie ein Zugriff auf die Daten zu erfolgen hat. Dabei sind folgende Punkte von besonderer Bedeutung:

1. Es wird ein Private und Public Key benötigt (https://aws-portal.amazon.com/gp/aws/developer/account/index.html?ie=UTF8&action=access-key)

2. Es muss ein signierter Aufruf erfolgen (http://docs.amazonwebservices.com/AWSECommerceService/latest/DG/index.html?Query_QueryAuth.html)
3. Die Anzahl der API-Zugriffe sind beschränkt

Das Schlüsselpaar stellt Amazon aus. Der signierte Aufruf lässt sich anhand der Dokumentation (http://docs.amazonwebservices.com/AWSECommerceService/latest/DG/index.html?Query_QueryAuth.html) von Amazon realisieren oder unter Verwendung bereits vorhandener Skripte. Ich nutze dazu die Lösung AWS Signed Query von Ulrich Mierendorff (http://mierendo.com/software/aws_signed_query/). Um die Anzahl der Zugriffe gering zu halten, empfiehlt es sich ein Caching-Mechanismus zu implementieren. Dazu lässt sich bspw. der PEAR Cache nutzen. Für PHP kann die Implementierung wie folgt aussehen:

```
<?php
#include Pfad für die PEAR-Installation setzen
ini_set("include_path",  Pfad_zur_PEAR_Installation"/_PEAR/PEAR".
PATH_SEPARATOR . ini_get("include_path") );

#PEAR Cache
require_once 'Cache/Output.php';

#Function aws_signed_query von Ulrich Mierendorff
require_once 'Pfad_zum_Script/aws_signed_query.php';
?>
```

Nachdem alle notwendigen Skripte „geladen“ wurden, kann nun der eigentliche API-Aufruf erfolgen. Dabei wird zwischen ItemSearch und ItemLookup unterschieden. ItemLookup ermöglicht den Zugriff auf ein bestimmtes Produkt von Amazon unter Verwendung ASIN (Amazon Standard Identification Number). Um den API Aufruf später einfacher zu nutzen, wird als Funktion umgesetzt.

```
<?php
function getAmazonBookPropertiesByAsin($asin){

        $public_key = "xxxxxxxxxxxxxx";
        $private_key = " xxxxxxxxxxxxxx ";

        #PEAR File Caching
        $opts = array('cache_dir' => 'Pfad_zum_CacheDir/_cache/');
        $cache = new Cache_Output('file', $opts);
        $cache->fileLocking=true;
        $id = $cache->generateID($asin);

        #Pruefung ob Object bereits im Cache
        if(!$pxml = $cache->start($id)){

                #Amazon API Aufruf fuer deutschen Amazon Store und allen
                Produktinformationen (ResponseGroup=Large), sowie Publisher ID
                (AssociateTag) fuer Umsatzverguetung
                $pxml = aws_signed_request("de",
```

```
            array("Operation"=>"ItemLookup","ItemId"=>$asin,"AssociateTag"=>"xx
            xxxxxxx","ResponseGroup"=>"Large"), $public_key, $private_key);

            $cache->endPrint(86400);

        }

        if ($pxml === False){

            echo "Did not work.\n";

        }

        else{

            $book = array();
            $book['title']  = (string) $pxml->Items->Item->ItemAttributes->Title;
            $book['image']   = (string) $pxml->Items->Item->ImageSets->ImageSet-
            >SmallImage->URL;
            $book['author']  = (string) $pxml->Items->Item->ItemAttributes->Author;
            $book['url']  = (string) $pxml->Items->Item->DetailPageURL;
            $book['listprice']   = (string) $pxml->Items->Item->ItemAttributes-
            >ListPrice->FormattedPrice;

            return $book;

        }

}

?>
```

Mit diesem Funktionsaufruf lassen sich nun die verschiedenen Daten zu einem Buch direkt in die Webseite integrieren und grafisch an die Webseite anpassen. Um bspw. ein kleines Buchcover, den Autor, den Buchtitel und den Preis auszugeben und direkt mit dem Amazon Store zu verlinken, ist folgende Vorgehensweise möglich:

```
<?php

echo "<h2>Literatur-Tipp</h2>\n";
echo "<a href=\"".$book_details['url']."\"><img title=\"Details
zu:".$book_details['title']."\" src=\"".$book_details['image']."\" border=\"0\"
alt=\"".$book_details['title']."\"/></a>\n";
echo "<span
class=\"relatedAuthor\">".utf8_decode($book_details['author'])."</span><br/>\n";
echo "<strong class=\"book\">".$book_details['title']."</strong><br/>\n";
echo "<strong class=\"bookPrice\">".$book_details['listprice']."</strong> <a
href=\"".$book_details['url']."\">Weitere Details</a>\n";

?>
```

Das Resultat könnte dann wie auf dem Screenshot dargestellt aussehen.

**Abbildung 1: Literatur-Tipp über Amazon API auf free-mac-software.com**

Statt den „Literatur-Tipp“ direkt mit der Amazon Webseite zu verlinken, ist auch die Weiterleitung auf eine „Buch-Detailseite“ denkbar. Auf dieser könnten dann weitere Informationen dargestellt und wiederum mit zusätzlichem Content verknüpft werden. Somit ließe sich auch ein Shop abseits der Shop-Widgets abbilden, der mehr Flexibilität erlaubt und mit weiterem Content angereichert werden könnte.

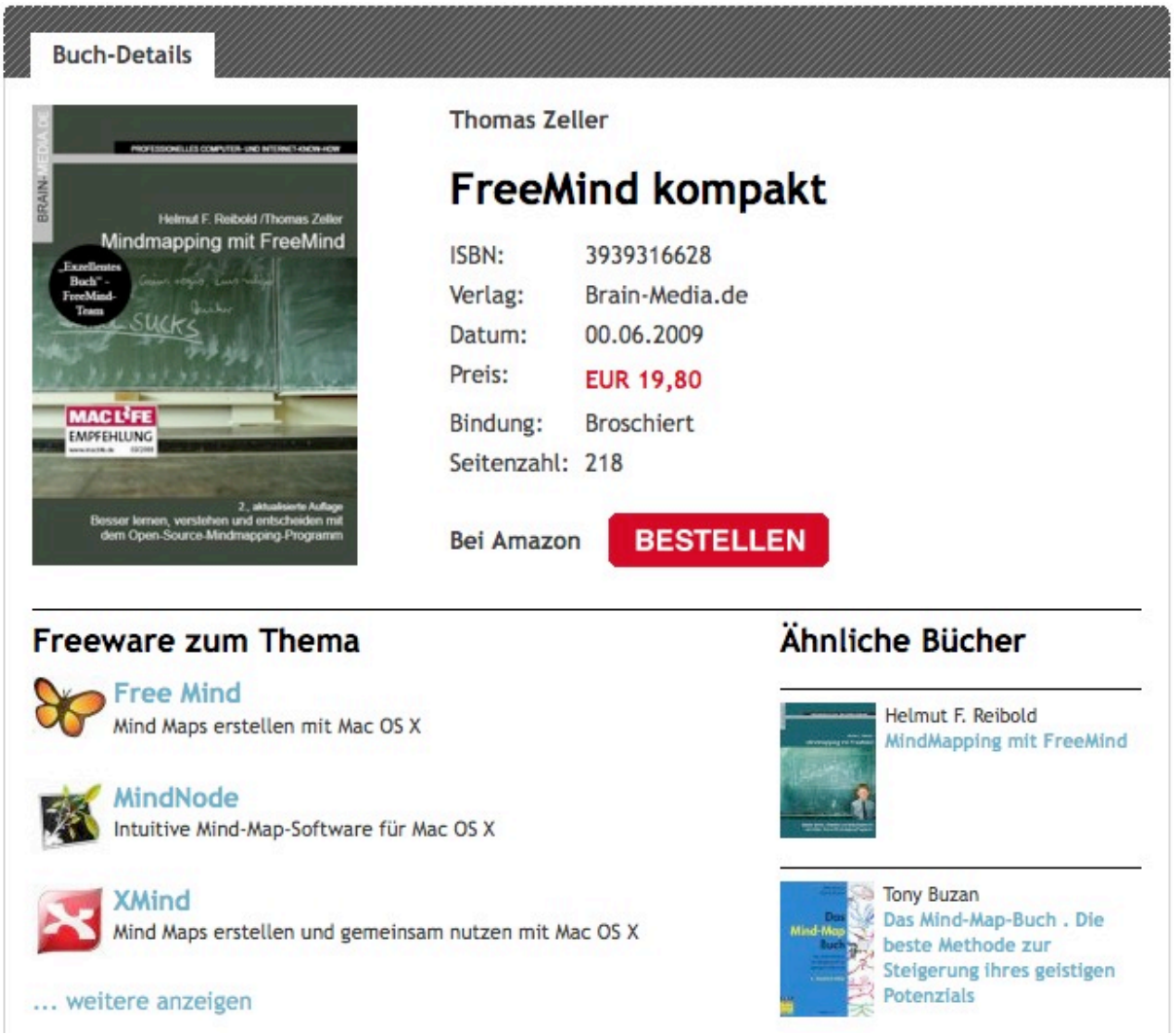


Abbildung 2: Buchdetailseite über Amazon API auf free-mac-software.com

Diese Beispiele lassen sich noch beliebig erweitern indem über die Amazon API auch Warenkorb- und Wunschlisten-Funktionen genutzt würden.

Neben Amazon bieten eine Reihe weitere Partner-Programme ebenfalls solche APIs, wodurch vielfältige Produktdaten und Integrationsmöglichkeiten zur Verfügung stehen.

## Der Autor

Andreas Witt ist wissenschaftlicher Mitarbeiter an der Fachhochschule Brandenburg, unter dem Stichwort WebManagement als selbstständiger Projekt-Manager (www.andreas-witt.net) tätig und Betreiber des Portals Free Mac Software. Außerdem engagiert er sich als Mitglied im webEdition e.V. (www.webedition.org) bei der Weiterentwicklung des Open Source CMS webEdition.