How to Use a FusePump Data Feed

©2009 FusePump Ltd | All right reserved
Email. publications@fusepump.com

Contents

Introduction

There are many ways to use product data feeds in affiliate marketing, which makes them a very powerful tool for increasing affiliate revenues from your site. By displaying detailed product information on your sites, you can improve the user experience as well as increasing conversion rates.

How you choose to display product information on your site is up to you, but in order to do this you'll first need to be able to load the data from a feed and store it on your server so that it can be output and displayed to your users.

The steps involved are as follows:

1. Download the feed.
2. Process the feed and store product data in a database.
3. Display product data on your site.

Furthermore, if you want to keep the product data on your site up to date, you'll need to automate steps 1 and 2 above so that they occur every time the feed is updated remotely by the feed provider.

There are many technologies that can be used to automate the download and processing of XML data feeds, however, by far the most commonly used is PHP.

Likewise, there are several technologies that can be used to store product data in a database, however, by far the most commonly used is MySQL.

The reason PHP and MySQL are so popular is that they are offered as standard by the vast majority of low cost web hosting providers. This is good news, as it means if you're already running a web site, you most likely have these two pieces of software installed on your server already.

What will I find out?

This tutorial will explain:

1. How to use PHP to automate the download and processing of an XML product feed
2. How to store product data from the feed in a MySQL database
3. How to read product data from the MySQL database and display it on your affiliate site

What do I need to know already?

This tutorial will assume limited prior knowledge of PHP and MySQL, although it would be good if you were familiar with each of these technologies before starting. If you've used PHP and MySQL before, you should find it easy to understand.

If you're not familiar with PHP and MySQL, it would be worth reviewing the following resources before you delve too deeply into this tutorial:

- PHP
  - http://se2.php.net/tut.php – a great introductory tutorial for PHP newbies.
- MySQL
  - http://dev.mysql.com/doc/refman/5.1/en/tutorial.html – a good introduction to MySQL.

What will I need?

You'll need to have PHP and MySQL installed and configured on your server. If you're using a major web hosting provider, its likely these will already be available to you. Otherwise, you may need to contact your web host for advice on how to set these up.

You will also need to have access to a FusePump data feed to work with. If you have'nt yet downloaded (or configured) a FusePump data feed, please see the tutorial "How to Download a FusePump Data Feed".

A Quick Look at The Feed

Text.

```
<?xml version="1.0" encoding="utf-8"?>
<products>
      <product>
            <price>1054.00</price>
            <buyurl>http://www.jdoqocy.com/click-1234-10675895</buyurl>
            <image>http://media.thomson.co.uk/asset/lpp/753/218.jpg</image>
            <destination>Algarve</destination>
            <resort>Balaia</resort>
            <accommodation>Holiday Village Algarve</accommodation>
            <departuredate>15/09/2009</departuredate>
            <returndate>22/09/2009</returndate>
            ...
      <product>
      ...
<products>
```

©2009 FusePump Ltd | All right reserved
Email. publications@fusepump.com

Creating the Database

Before product data from the feed can be displayed on the site, it must be stored in a database table.

First, however, we will need to create a database ("holiday_demo") to store our table and a user account ("holiday_user") for accessing the database.

Open up the MySQL console[1] and issue the following SQL statements:

```
mysql> CREATE DATABASE holiday_demo;
mysql> GRANT ALL PRIVILEGES ON holiday_demo.* TO
'holiday_user'@'localhost' IDENTIFIED BY 'holidays';
```

Don't forget to hit the enter key after typing each line in order the execute the statement.

Note: The "mysql> " text at the start of each line represents the mysql prompt. If you're copying and pasting each line, you will not need to include this text.

Creating a Table

Every time a product is displayed on the site, we will need to show the following details from the feed:

- destination
- resort
- accommodation
- departuredate
- returndate
- price
- buyurl
- image

We will therefore need to create a database table called "holidays" which contains one field for each of these feed outputs. Holiday data will be stored to this table when the feed is processed and subsequently read from this table when we need to display it on the site.

Issue the following statement in the MySQL console:

[1] In order to open the MySQL console, you will need to type the command "mysql" at either the command prompt (in Windows) or the terminal (in Linux). If mysql is password protected, you will need to supply a root username when you enter the mysql command e.g. "mysql -u {username} -p". You will then be asked to enter your password.

```
mysql> CREATE TABLE `holiday_demo`.`products` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`destination` VARCHAR( 255 ) NOT NULL ,
`resort` VARCHAR( 255 ) NOT NULL ,
`accommodation` VARCHAR( 255 ) NOT NULL ,
`departureDate` VARCHAR( 255 ) NOT NULL ,
`returnDate` VARCHAR( 255 ) NOT NULL ,
`price` VARCHAR( 255 ) NOT NULL ,
`buyurl` VARCHAR( 255 ) NOT NULL ,
`image` VARCHAR( 255 ) NOT NULL
);
```

The statement above creates a text field of 255 characters length in which to store each of the holiday outputs. In addition, an “id” field is created and set to auto increment to ensure that each holiday in our database has a unique identifier assigned to it. This makes it easy to reference, update and remove details for specific holidays in the future, should we need to.

Processing the Feed

Products in the feed are updated once a day, which means we'll also want to update our database once a day to make sure its as accurate as possible. This will require the feed to be downloaded and processed using a PHP script. This PHP script can be set up to run automatically at a particular time each day, using a CRON[2] job.

So, what does the script look like? Well, the script needs to perform three basic functions:

1. Download the feed
2. Read products from the feed
3. Write products to the database

We will define each of these functions below, and then put them together to create the complete script.

Downloading the Feed

Having configured a static URL for the feed using the FusePump FeedCreator, we will need to download data from this URL using PHP.

We will download the feed using the "file_put_contents" method[3], as follows:

```
//set the URL from which to download the feed
$url = "http://fusepumpaffiliates.co.uk/feed-distribution/thomson/grab.php?
fpid=793ed2a50ee6";

//set the filename where the downloaded feed will be stored
$filename = "feed.xml";

//download the feed
$file = file($url);

//save the feed
file_put_contents($filename, $file);
```

Reading Products from the Feed

Now that we've downloaded the feed, we need to interpret the XML data, one product at a time. There are several libraries in PHP for parsing XML data, and the most common of these is SimpleXML whose name reflects the fact that it is very easy to

[2] CRON is a Linux tool that enables users to schedule jobs (commands or shell scripts) to run automatically at a certain time or date. For more information, please see http://www.howtoforge.com/a-short-introduction-to-cron-jobs.

[3] If you're having problems using this method, it probably means your PHP script doesn't have write access to its folder. Try checking your PHP configuration and folder permissions.

use. SimpleXML allows you build an array of data (which can be easily read from using PHP) directly from an XML file and is great when you are working with small pieces of XML data which you need to translate quickly.

For larger XML files however, SimpleXML is unsuitable as the memory required to store the data from an entire feed in a single array can be very large. As we're working with a large feed, we'll need to use an XML parser that enables us to read products from the feed one at a time instead. Each time we read a product from the feed, we will quickly store it to the database (using the "saveProduct" function which is defined in the next section of this tutorial) and then discard it completely before moving onto the next one. Although its a little more complicated, this approach will give us great performance and allow us to work with arbitrarily large feeds.

In order to do this, we will need to use the XMLReader library in PHP, which is available as standard in all recent versions. The section of code below illustrates how to read products from the feed and retrieve the fields we're interested in for each one. If you would like to understand how this code works in more detail, please refer to the XMLReader documentation[4] on the PHP web site.

```
//open feed using XMLReader
$xmlReader = new XMLReader();
$xmlReader->open($filename);

//process products
while ($xmlReader->read()) {
      if ($xmlReader->name == "product") {
            $product = array();
            while ($xmlReader->read()) {
                  $name = $xmlReader->name;
                  if ($name == "product")
                        break;
                  switch($name) {
                        case "destination":
                        case "resort":
                        case "accommodation":
                        case "departuredate":
                        case "returndate":
                        case "price":
                        case "buyurl":
                        case "image":
                           if (!isset($product[$name]))
                              $product[$name] = $xmlReader->readString();
                              break;
                  }
            }
            saveProduct($product);
      }
}
```

[4] http://us3.php.net/manual/en/book.xmlreader.php

Saving Products to the Database

As described above, we need to be able to store products from the feed to the database we created earlier. In order to do this using PHP, we'll first need to open a connection to the database using the "`mysql_connect`" function. We'll also need to close the database connection once we've finished, for good practice. The code for opening the connection will need to be placed somewhere near the start of your PHP script, ensuring the connection to the database is open before you try to make any queries. The code for closing the connection will need to be placed at the end, after you've finished making queries.

```
//connect to database
$db = mysql_connect("localhost", "holiday_user", "holidays");
mysql_select_db("holiday_demo");

...

//close database connection
mysql_close($db);
```

Each product will be entered into the products table using a MySQL `INSERT` statement. The function outlined below takes an array containing all of the required data for a single product as its argument and places this data within an insert query before calling `mysql_query` to execute the query. The `mysql_real_escape_string` is used to prevent characters within the data causing problems with MySQL when they are added to our query.

```
function saveProduct($product) {
      $result = mysql_query(
            "INSERT INTO products SET " .
            "`destination`='" . escape($product["destination"]) . "', " .
            "`resort`='" . escape($product["resort"]) . "', " .
            "`accommodation`='" . escape($product["accommodation"]) . "', " .
            "`departuredate`='" . escape($product["departuredate"]) . "', " .
            "`returndate`='" . escape($product["returndate"]) . "', " .
            "`price`='" . escape($product["price"]) . "', " .
            "`buyurl`='" . escape($product["buyurl"]) . "', " .
            "`image`='" . escape($product["image"]) . "'"
      );
      if (!$result) {
            print mysql_error();
      }
}

function escape($string) {
      return mysql_real_escape_string($string);
}
```

©2009 FusePump Ltd | All right reserved
Email. publications@fusepump.com

The Complete Script

The complete script for reading, processing and storing data from the feed automatically is shown below.

```
<?php

//connect to database
$db = mysql_connect("localhost", "holiday_user", "holidays");
mysql_select_db("holiday_demo");

//set the URL from which to download the feed
$url = "http://fusepumpaffiliates.co.uk/feed-distribution/thomson/grab.php?
fpid=793ed2a50ee6";

//set the filename where the downloaded feed will be stored
$filename = "feed.xml";

//download the feed
$file = file($url);

//save the feed
file_put_contents($filename, $file);

//open feed using XMLReader
$xmlReader = new XMLReader();
$xmlReader->open($filename);

//process products
while ($xmlReader->read()) {
      if ($xmlReader->name == "product") {
            $product = array();
            while ($xmlReader->read()) {
                  $name = $xmlReader->name;
                  if ($name == "product")
                        break;
                  switch($name) {
                        case "destination":
                        case "resort":
                        case "accommodation":
                        case "departuredate":
                        case "returndate":
                        case "price":
                        case "buyurl":
                        case "image":
                              if (!isset($product[$name]))
                                    $product[$name] = $xmlReader-
>readString();
                              break;
                  }
            }
            saveProduct($product);
      }
}

function saveProduct($product) {
      $result = mysql_query(
            "INSERT INTO products SET " .
            "`destination`='" . escape($product["destination"]) . "', " .
            "`resort`='" . escape($product["resort"]) . "', " .
            "`accommodation`='" . escape($product["accommodation"]) . "', " .
            "`departuredate`='" . escape($product["departuredate"]) . "', " .
            "`returndate`='" . escape($product["returndate"]) . "', " .
            "`price`='" . escape($product["price"]) . "', " .
            "`buyurl`='" . escape($product["buyurl"]) . "', " .
            "`image`='" . escape($product["image"]) . "'"
      );
      if (!$result) {
            print mysql_error();
      }
}

function escape($string) {
      return mysql_real_escape_string($string);
}

//close database connection
mysql_close($db);

?>
```

Displaying Products

Creating a Product Table in HTML

In order to display products from the database on our web page using PHP, we will need to use the `print` statement to output snippets of HTML.

First of all, lets see what the table will look like:

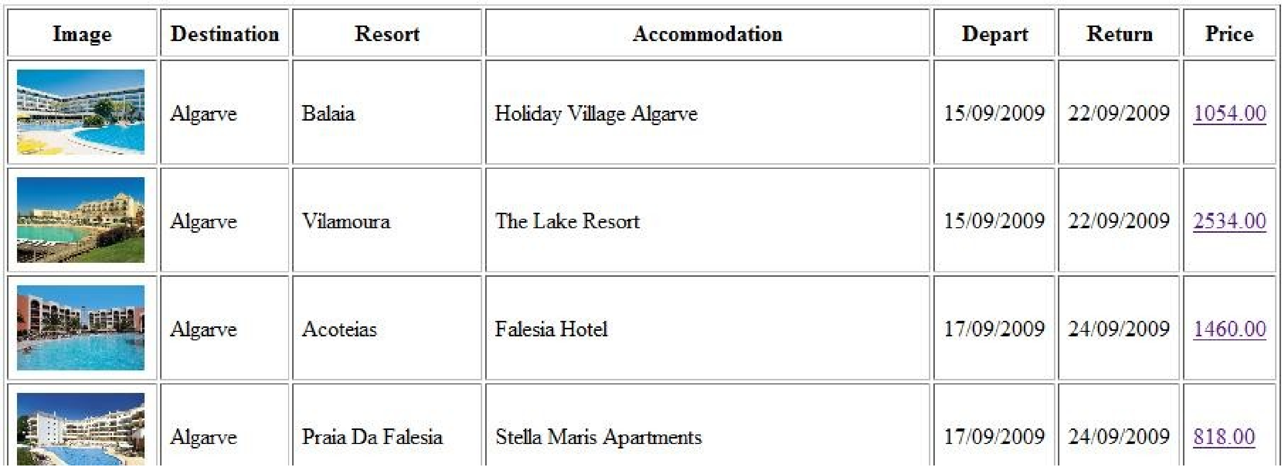

Image	Destination	Resort	Accommodation	Depart	Return	Price
	Algarve	Balaia	Holiday Village Algarve	15/09/2009	22/09/2009	1054.00
	Algarve	Vilamoura	The Lake Resort	15/09/2009	22/09/2009	2534.00
	Algarve	Acoteias	Falesia Hotel	17/09/2009	24/09/2009	1460.00
	Algarve	Praia Da Falesia	Stella Maris Apartments	17/09/2009	24/09/2009	818.00

Now lets look at the HTML required to create the table:

```
<table border='1' cellpadding='6'>
      <tr>
          <th>Image</th>
          <th>Destination</th>
          <th>Resort</th>
          <th>Accommodation</th>
          <th>Depart</th>
          <th>Return</th>
          <th>Price</th>
      </tr>
      <tr>
          <td>
              <img height='60' src='http://thomson.co.uk//753/218.jpg'>
          </td>
          <td>Algarve</td>
          <td>Balaia</td>
          <td>Holiday Village Algarve</td>
          <td>15/09/2009</td>
          <td>22/09/2009</td>
          <td>
              <a href='http://www.jdoqocy.com/click-1234-4563'>1054.00</a>
```

©2009 FusePump Ltd | All right reserved
Email. publications@fusepump.com

```
          </td>
      </tr>

      ...

</table>
```

The HTML for the table is output in PHP as follows:

```
//output start of table
print "<table border='1' cellpadding='6'>";
print "<tr>";
print "<th>Image</th>";
print "<th>Destination</th>";
print "<th>Resort</th>";
print "<th>Accommodation</th>";
print "<th>Depart</th>";
print "<th>Return</th>";
print "<th>Price</th>";
print "</tr>";

...

//output end of table
print "</table>";
```

Reading Products from the Database

Once again, we'll need to connect to the database before we can read data from the products table. We'll also need to close the database connection once we've finished, for good practice. The code for opening the connection will need to be placed somewhere near the start of your PHP script, ensuring the connection to the database is open before you try to make any queries. The code for closing the connection will need to be placed at the end, after you've finished making queries.

```
//connect to database
$db = mysql_connect("localhost", "holiday_user", "holidays");
mysql_select_db("holiday_demo");

...

//close database connection
mysql_close($db);
```

Data is read from the products table using a MySQL `SELECT` query. In PHP, the results of a MySQL query are processed by looping through the result of the `mysql_query` function. The mysql_fetch_array function returns a single line of data (in our case, a product) from the database as an array. Each product will be output using the `outputProduct` method, which is defined in the next section of this tutorial.

```
//output table lines
$result = mysql_query("SELECT * FROM products");
if ($result) {
      while ($row = mysql_fetch_array($result)) {
            outputProduct($row);
      }
}
```

Outputting Products as HTML

Now we need to define a method to output the necessary HTML for each product. For every product, we will output a table row `<tr>` and for each data field (e.g. destination, returndate), we will output a table cell `<td>`.

To display an image for each product, we will need to output an `<img>` tag, using the image field for the product as the `src` attribute.

In order to encourage the user to buy a holiday, we will create a hyperlink for each price element using an `<a>` tag. We will need to set the `href` of the tag to the url of the product to ensure the user is directed to correct page when they click the link.

```
function outputProduct($product) {
      print "<tr>";
      print "<td><img height='60' src='" . $product["image"] . "'></td>";
      print "<td>" . $product["destination"] . "</td>";
      print "<td>" . $product["resort"] . "</td>";
      print "<td>" . $product["accommodation"] . "</td>";
      print "<td>" . $product["departuredate"] . "</td>";
      print "<td>" . $product["returndate"] . "</td>";
      print "<td>";
      print "<a href='" . $product["buyurl"] . "'>";
      print $product["price"];
      print "</a>";
      print "</td>";
      print "</tr>";
}
```

The Complete Script

The complete script for displaying products from the database is shown below.

```
<?php

//connect to database
$db = mysql_connect("localhost", "holiday_user", "holidays");
mysql_select_db("holiday_demo");

//output start of table
print "<table border='1' cellpadding='6'>";
print "<tr>";
print "<th>Image</th>";
print "<th>Destination</th>";
print "<th>Resort</th>";
print "<th>Accommodation</th>";
print "<th>Depart</th>";
print "<th>Return</th>";
print "<th>Price</th>";
print "</tr>";

//output table lines
$result = mysql_query("SELECT * FROM products");
if ($result) {
      while ($row = mysql_fetch_array($result)) {
            outputProduct($row);
      }
}

//output end of table
print "</table>";

function outputProduct($product) {
      print "<tr>";
      print "<td><img height='60' src='" . $product["image"] . "'></td>";
      print "<td>" . $product["destination"] . "</td>";
      print "<td>" . $product["resort"] . "</td>";
      print "<td>" . $product["accommodation"] . "</td>";
      print "<td>" . $product["departuredate"] . "</td>";
      print "<td>" . $product["returndate"] . "</td>";
      print "<td>";
      print "<a href='" . $product["buyurl"] . "'>";
      print $product["price"];
      print "</a>";
      print "</td>";
      print "</tr>";
}

//close database connection
mysql_close($db);

?>
```

Conclusion

We hope you have enjoyed this tutorial and found it useful. If you would like to learn more about FusePump, please visit www.fusepump.com.

If you still have questions, please feel free to email us on publications@fusepump.com.