OCAPI Training Exercise: Product search / add to cart using PHP & jQuery

Overview:

This exercise provides a simple use-case for demonstrating Demandware’s Shop API. It consists of three basic parts:

1. A product search is conducted and the results are shown using PHP to access the OCAPI product search resource and process the results directly on the server.
2. Products may be added to the cart using a combination of PHP and a Javascript XHTTP request with jQuery to the OCAPI basket resource.
3. Contents of the basket are displayed on the page following a request to the OCAPI basket resource and parsing the JSON result via jQuery in the browser.

Requirements:

In order to run this exercise there are a few environmental considerations and requirements:

1. A working PHP environment is necessary in order to make the calls into OCAPI. For the purpose of the exercise it is recommended this be installed locally through MAMP or WAMP, which provide simple PHP environments without setting up a standalone web server, etc.
2. A working Demandware sandbox instance that is configured to accept traffic from the PHP environment, and has a properly configured Site Genesis demo site available.

PHP Environment Setup:

1. Install PHP using WAMP or MAMP:

We recommend setting up MAMP (Mac OSX), or WAMP (Windows) on your local machine in order to make running PHP simple and easy to configure / develop against.

MAMP may be found here : <http://www.mamp.info/en/> and the free version is fine for this purpose.

WAMP may be found here: <http://www.wampserver.com/en/>

After performing the standard install for your local PHP software, it is recommended you familiarize yourself with the PHP environment, and accessing files locally using your <http://localhost> address when the server is running. This document will not attempt to provide detailed insight into this process, as there are many readily available tutorials that do so.

In order to run this exercise, we will be using “CURL”, to make our http calls. This is installed by default with MAMP / WAMP, but if you want to make sure you have CURL installed, you can find out by inspecting the phpinfo in a browser.

<http://php.net/manual/en/function.phpinfo.php>

1. Configure local host file:

OCAPI will not permit access from “localhost” for security purposes. Therefore, in order to use a local PHP server, it is necessary to configure your system’s local hosts file in order to map your localhost ip address to a dummy domain name. The default configuration for localhost ip address is 127.0.0.1. In your machine’s host file, map this ip address to a dummy domain name (e.g., [www.localstore.com](http://www.localstore.com)). You can test that this is working by pinging that domain name from a terminal window. If it returns the correct ip address you are ready to use it to access OCAPI resources.

When browsing your local PHP resources, replace “localhost” with your dummy domain name. Using MAMP for example, you would normally access your php files on the server like so:

<http://localhost:8888/folder/phpfile.php>

Now you will use:

<http://www.localstore.com:8888/folder/phpfile.php>

Once you have confirmed that this is working properly using a test PHP or HTML file (e.g., a php version of ‘hello world’), you can configure your OCAPI settings in Business Manager to permit traffic from that dummy domain name as you would any other site.

1. Install the exercise PHP files into your PHP environment

Place the 3 included PHP files in a folder within your PHP environment . In MAMP, this is in Applications/MAMP/htdocs (though this can be customized if you prefer).

1. Create an empty Cookie file

Using Demandware’s OCAPI basket requires that a session cookie be created to store the contents of the cart. CURL does not write cookies automatically like a browser does. In order to use cookies with CURL we must create an empty text file on the file system and provide CURL with a path to this file. When we specify the location of this file for CURL it will write cookie data to it.

For simplicity’s sake, in the exercise, I have placed the cookie file in the same folder as the php files will be run from. In MAMP, this is in Applications/htdocs/dw (or wherever you placed your exercise in the previous step). This line in the exercise php files “ocapi\_getcart” and “ocapi\_addtocart”

$ckfile = "/Applications/MAMP/htdocs/dw/cookies.txt";

should be modified so that the path is correct. For WAMP, this will be a different path, but the idea is the same – the cookie file needs to be readily accessible to the PHP script. The PHP script will check for the existence of this file before running. If it does not exist, it will fail.

\*\*NOTE: If you wish to empty your cart after running the exercise for testing purposes, simply delete the contents of this cookies.txt file (not the file itself!). Clearing your browser’s cookies will have no effect on CURL.

1. Running the exercise

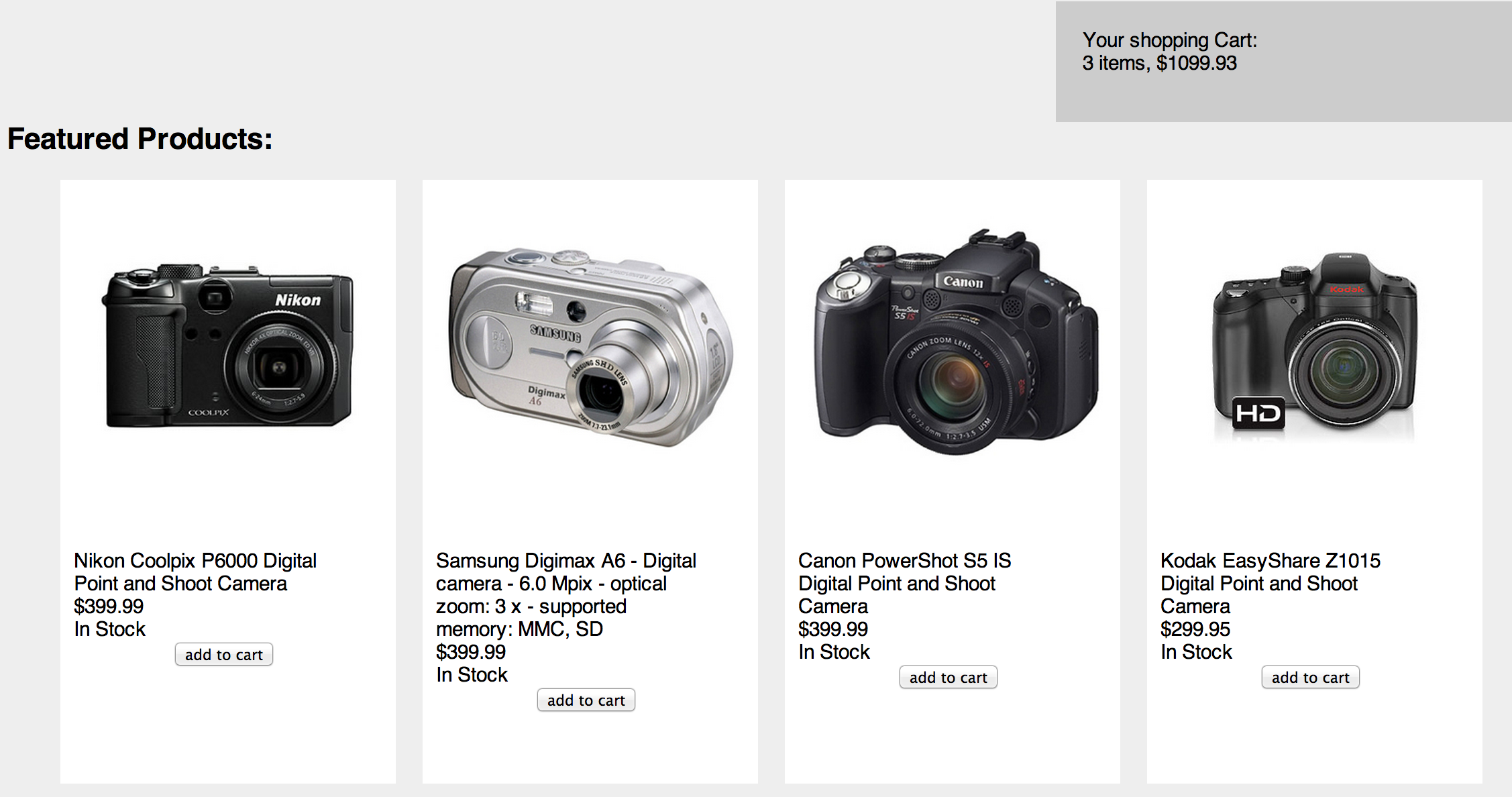
When your environment is setup, open ocapi\_productsearch.php in your browser using the dummy domain name you created, and provide http parameters in the query string for the search keyword and number of results as follows:

<http://localstore:8888/ocapiexercise/ocapi_productsearch.php?q=camera&num=4>

“q” is the parameter name for the search keyword, e.g. ‘camera’

“num” is the parameter name for the results to show (defaults to 3)

You should receive a page that displays products, including imagery, pricing, name, etc. (see screenshot). Clicking “Add to cart” should result in the cart area in the top right being refreshed with an item total.



1. Exercise explained:
   1. Ocapi\_productsearch.php

The purpose of this file is to demonstrate how to perform a simple shop api request using PHP and then process the results directly in PHP. The result from OCAPI is converted from JSON into an associative array and then accessed using PHP. HTML is output by the PHP directly.

In addition to the PHP, javascript events are assigned to the elements on the page to show how javascript might be used in conjunction with OCAPI for other actions.

* 1. Ocapi\_addtocart.php

This php file takes a product ID and adds a product to the cart using the shop api’s basket resource. In order to maintain the session state, cookies are used by CURL. In the exercise, this file is called through a javascript XHTTP request using JQuery. This is a very likely use case, and provides immediate feedback to the user. PHP is used to make the call to OCAPI rather than using javascript directly, as javascript is not able to properly set the headers that OCAPI expects for the POST action (it is designed to be server-server, not client-server). Instead of returning the raw JSON from OCAPI, this script also simplifies it and returns either “Success” in the event of a 200, or the error code if the call fails. \*\*I should note that the basket add action in the shop api returns the contents of the cart, and therefore the ocapi\_getcart.php file is actually unnecessary in this use case. In a real-world situation it is much more likely the developer would use the results from the call directly. I have opted for using a subsequent call and php script in order to demonstrate how the cookie makes the session data persistent across multiple actions / instances of CURL.

* 1. Ocapi\_getcart.php

This php script simply returns the contents of the basket in the original JSON format so that it can be processed and used by javascript in the browser. This is a very practical usage of OCAPI. This script also demonstrates how the cookie loads the proper session data so that it is persistent for multiple calls.