Chapter 3 - First steps - Access the Web service and list client

Table of content

First steps - Access the Web service and list client

Preparation

1. Configure your PHP installation so that it has the cURL extension installed and activated:
   - With Windows: Place this line in your `php.ini` file:

     ```
     extension=php_curl.dll
     ```

   - Linux/Mac: install the cURL extension:

     ```
     sudo apt-get install php5-curl
     ```

2. Copy the `PSWebServiceLibrary.php` file at the root of your Web server.

   ⚠ You can also do this tutorial on a local server even while your shop is on the Internet.

3. Create a `list_the_clients.php` file at the root of the Web server that you have chosen.
4. Specify where to find the web server in your file:

   ```
   require_once( './ PSWebServiceLibrary.php' );
   ```

Configured this way, your file should be found in the same folder as `PSWebServiceLibrary.php`.

Accessing the web service

In this section we will see how to access the web service using the PHP library.

First, you must create an instance of the `PrestaShopWebservice` object, which takes 3 parameters in its constructor:

- The store's root path (ex: `http://example.com/`).
- The authentication key (ex: ZR92FNY5UFRERNI3O9Z5QDHWKTP3YIIT).
- A boolean value, indicating whether the Web service must use its debug mode.

If you do not understand the terms of object-oriented programming such as instance, method, or constructor, that's okay for the rest of the tutorial. Here's how you create a Web service call:

```
$webService = new PrestaShopWebservice('http://example.com/', 'ZR92FNY5UFRERNI3O9Z5QDHWKTP3YIIT', false);
```

Once the instance is created, you can access the following methods:

Method	HTTP equivalent
get()	GET
add()	POST
edit()	PUT
delete()	DELETE

We will explain how to use of these methods in other parts of the tutorial.

Handling errors

It is essential that you understand how to handle errors with the web service library. By implementing error-catch method early, you will more easily detect issues, and be able to correct them on the go.

Error handling with the web service library is done using PHP exceptions. If you do not know about them, you should read http://php.net/manual/en/language.exceptions.php, as exceptions are an essential part of good coding practice.

How it works

The error handling is done within a `try..catch` block, with the web service processing being done in the `try` section, the `catch` one containing the error handling code.

```
try {
  // Execution ( if an error occurs in this code, stops and goes in the catch block)
}
catch {
  // Error handling (tries to catch the error or the error display)
}
```

Example

That means each creation or use of the library must be located within a "try" block. The "catch" block can then handle the error if it occurs during the execution of the try block.
Now we'll see how to list all customers via the web service, and then we will see the four CRUD methods.

In the following code sample, we want to to get the list of all customers:

1. In the `try` block, we instantiate the `PrestaShopWebservice` object with the necessary parameters and retrieve the `customer` resource XML in a variable.
2. In the `catch` block, we put code to display the PHP error message, if anything wrong happens in the `try` block.

```
try {
        // creating web service access
        $webService = new PrestaShopWebservice('http://example.com/', 'ZR92FNY5UFRERNI3O9Z5QDHWKTP3YIIT',
false);

        // call to retrieve all clients
        $xml = $webService->get(array('resource' => 'customers'));
}
catch (PrestaShopWebserviceException $ex) {
        // Shows a message related to the error
        echo 'Other error: <br />' . $ex->getMessage();
}
```

Listing clients

Let's now see how to view a full list of client IDs. We could display more information and customize it, but that's for another part of this tutorial

As we saw in the previous code sample, we need the `get()` method to retrieve an XML file containing all the customers. The parameter has to be a key-value array, where we define the resource we want:

Key	Value
resource	customers

```
// The key-value array
$opt['resource'] = 'customers';

Retrieving the XML data
$xml = $webService->get($opt);
```

The value defines the resource that the web service will use in a future call. The value could be carrier types, countries or any other type of resource that can be found in the "Web service" menu of your back-office.

Result

Launching the code above will return a `SimpleXML` object containing all the client IDs.

```
<?xml>
<prestashop>
  <customers>
    <customer>
      Client ID
    </customer>
    <customer>
      Client ID
    </customer>
    <customer>
      Client ID
    </customer>
...Other client tags
  </customers>
</prestashop>
```

Now we need to access the tags that interest us in the XML file.

Structure

The data returned by calling `$webService->get` puts us at the root of the document.

To access the fields of clients who are children from the Customers tag, we only need to retrieve all fields in an associative array in SimpleXML like this:

```
$resources = $xml->customers->children();
```

From there, we can access client IDs easily. Here's an example with a path from identifiers:

```
foreach ($resources as $resource)
        echo $resource->attributes() . '<br />';
```

Thanks to these elements, we can create a HTML table containing all the client IDs. Try that before reading the next chapter.

You can use the "Customers" menu in the back-office to find the IDs of all customers. If you encounter difficulties, have a look at the example file named `0-CustomersList.php`, to see the results you should get.