Exemple d'utilisation du service web PrestaShop : CRUD

Ce tutorial vous montre comment utiliser le service web de PrestaShop avec la librairie PHP en créant un « CRUD ».

Prérequis :
- Une boutique PrestaShop 1.4 installée
- un serveur XAMP sous PHP5

Qu'est ce que CRUD ?

CRUD est un acronyme anglais qui signifie "Create, Read, Update, Delete" pour Créer, Récupérer, Mettre à jour et Supprimer.

Ce sont les 4 opérations de base qui permettent de gérer des données dans une application.

Le service web de PrestaShop utilise une architecture REST afin d'être disponible sur un maximum de plateforme, en effet les protocoles HTTP et XML sont présent sur un nombre incalculable de plateforme.

Qu'est ce que REST ?

REST définit une architecture représentant un ensemble des bonnes méthode à pratiquer sur le web. Elle définit plusieurs règles, dont une que nous allons décrire car elle s'apparente à CRUD.

Dans le protocole HTTP nous retrouvons 4 méthodes principales qui permettent d'effectuer des traitement sur les données et qui sont définit dans l'architecture REST, nous pouvons d'ailleurs faire cette correspondance avec CRUD :

- GET -> Retrieve (Récupérer)
- POST -> Create (Créer)
- PUT -> Update (Modifier)
- DELETE -> Delete (Supprimer)

Nous allons voir ensemble comment créer une petite application permettant d'effectuer ces 4 opérations sur les clients ("customers" en anglais).

Les chapitres 1, 2 et 3 sont obligatoires.

Vous allez voir dans les chapitres suivant comment interagir avec le service web avec chacune des opérations de CRUD pour vous donner les clés pour faire un CRUD complet.

- Si vous ne voulez que récupérer des données, par exemple dans l'élaboration d'une application web de notification des commandes alors vous pouvez ne vous interesser qu'au chapitre 4.

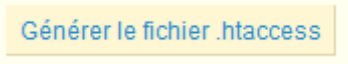
- Si vous préferez développer une application plus complète, les chapitre 4 à 7 vous interesseront.

Sommaire

Chapitre 1 - Mise en place : Création des accès dans le Back Office

Dans un premier temps nous allons créer un accès au service web.
Pour ce faire il suffit d'aller dans votre Back Office dans l'onglet Outils/Service Web.
Dans un premier temps séléctionnez "Activer le service web" et cliquez sur enregistrer afin d'activer le service.

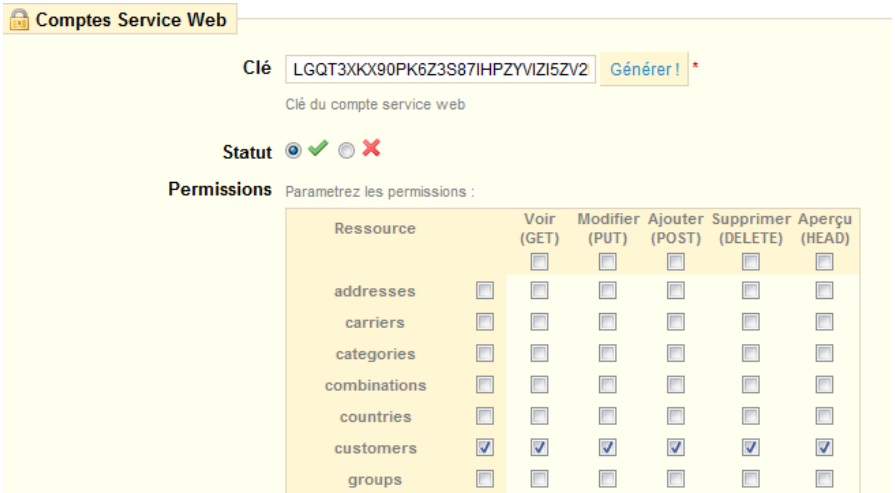
Génération d'un fichier .htaccess :
Afin que le service web puisse fonctionner, vous devez générer/régénérer un fichier .htaccess.
Toujours dans le Back Office, rendez vous dans l'onglet Outils/Générateurs puis cliquez sur :



Création de l'accès :
Retournez dans Outils/Service Web

- Cliquez sur "Nouveau", vous accèdez à la page de permission et de définition de la "Clé".
- Cliquez sur "Générer", Cela vous générera une clé d'authentification

C'est grâce à cette clé d'authentification qu'il sera possible d'accéder au service web.
Ensuite vous pouvez créer des droits pour chacunes des ressources auxquels vous souhaitez accéder.



Dans la liste des permissions, le bouton de gauche vous permet de définir l'ensemble des droits pour une ressource donnée. Sélectionnez les ressources que vous devez manipuler depuis votre application, dans notre cas cochez la première case de la ligne "customers" puis :
- Appuyez sur Enregistrer

Note sur la Clé d'authentification :
⚠ Afin que la clé ne puisse être devinée, veuillez utilisez le bouton "Générer".
Si vous definissez vous même la clé assurez vous qu'elle soit suffisement sécurisé et que ces droit sont limités.

Chapitre 2 - Découverte : Tester l'accès au service web avec le navigateur

Afin de tester si vous avez correctement configuré votre accès au service web, vous allez accéder à la page `http://Ma clé d'authentification@maboutique.com/api/` où « Ma clé d'authentification » est à remplacé par votre clé, vous y accéderez de préférence avec Mozilla Firefox.

> Note :
> Une autre méthode est d'accéder directement à la page suivante :
> `http://maboutique.com/api/`
> Celle ci devrait vous demander un identifiant ainsi qu'un mot de passe, l'identifiant à rentrer est la **clé d'authentification** et il n'y a **pas de mot de passe.**

Vous accédez alors à la liste des ressources que vous avez configuré dans votre back office avec l'ensemble des permissions accordées.

À l'aide de "**XLink**", vous allez pouvoir accéder à vos différentes ressources.

Qu'est-ce que "**XLink**" ?

Xlink permet d'associer un fichier XML à un autre fichier XML via un lien.

Dans la balise customers, vous devriez obtenir ces attributs :

```
<customers xlink:href="http://maboutique.com/api/customers" get="true"
put="true" post="true" delete="true" head="true">
```

les attributs get, put, post et delete ont comme valeur "true" (vrai), ce qui signifie que vous avez bien configuré la ressource "customers" et qu'elle est accessible.

Vous pouvez desormais utiliser le "XLink" qui pointe sur l'URL "http://maboutique.com/api/customers" et vous y rendre.

Une fois la liste des clients affichée via "http://exemple.com/boutique/api/customers", vous pourrez accéder aux XLink correspondant à chacun des clients.

Exemple :

le fichier XML situé dans "http://maboutique.com/api/customers/1" dont on retrouve le lien dans l'ensemble des clients (cf : lien précédent) vous donnera les propriétés du client ayant pour ID 1.

Ainsi vous naviguez dans le service web pour accéder à toutes les ressources en XML.

Chapitre 3 - Premiers pas : Accéder au service web et lister les clients

Préparation :

Configurez votre installation de PHP pour qu'elle ait l'extension CURL installée et activée :
Windows : Placer dans votre fichier php.ini la ligne suivante :
extension=php_curl.dll
Linux/Mac : installez l'extension CURL
sudo apt-get install php5-curl

Copiez le fichier fournit PSWebServiceLibrary.php à la racine de votre serveur WEB, c'est cette librairie dont nous allons expliquer l'utilisation dans ce tutorial.

> Note :
> Vous pouvez faire ce tutorial en local alors que votre boutique se trouve sur internet.

Créez un fichier lister_les_clients.php à la racine du serveur WEB que vous aurez choisi.
Spécifiez où se trouve le service web dans votre fichier :

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

Configuré de cette façon, votre fichier doit se trouver dans le même dossier que PSWebServiceLibrary.php.

3.1 Accéder au service web

Dans cette partie, nous allons voir comment accéder au service web via la librairie PHP.

Dans un premier temps, vous devez créer une instance de PrestaShopWebservice qui prend dans son constructeur 3 paramètres :

- Chemin racine de la boutique (ex : http://boutique.com/)
- La clé d'authentification (ex : ZR92FNY5UFRERNI3O9Z5QDHWKTP3YIIT)
- Un booléen indiquant si le service web doit utiliser son mode debug

Si vous ne comprenez pas les termes de la programmation orientée objet tel que instance, constructeur ou méthode ce n'est pas grave pour la suite du tutorial, voici comment vous devez créer un appel au service web :

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

Une fois l'instance créée vous pouvez accéder aux méthodes suivantes :
get (GET)
add (POST)
edit (PUT)
delete (DELETE)

Nous développerons l'utilisation de ces méthodes dans les différentes parties du tutorial.

3.2 Gestion des erreurs

L'apprentissage de la gestion des erreurs avec la librairie est essentielle pour débuter, si vous mettez en place directement cette vérification vous détecterez immédiatement d'où l'erreur provient ainsi que d'autres d'informations.

Pour se faire, la gestion d'erreur avec la librairie PHP du service web se fait à l'aide d'exceptions.

Principe : Les traitements liés au service web de prestashop doivent se situer dans un bloc try qui lui même doit être suivit d'un bloc catch permettant de récupérer les erreurs et si possible de les rattraper.

Illustration :

```
try
{
	// Execution (s'arrête et va dans le bloc catch si une erreur survient)
}
catch
{
	// Traitement des erreurs (tenter de rattraper l'erreur ou afficher l'erreur)
}
```

Exemple :

```
try
{
	// Création d'un accès au service web
	$webService = new PrestaShopWebservice(
				'http://maboutique.com/',
				'ZR92FNY5UFRERNI3O9Z5QDHWKTP3YIIT',
				false);
	// Appel de récupération de tous les clients
	$xml = $webService->get(array('resource' => 'customers'));
}
catch (PrestaShopWebserviceException $ex)
{
	$trace = $ex->getTrace();		// Récupère toutes les Informations sur l'erreur
	$errorCode = $trace[0]['args'][0]; // Récupération du code d'erreur
	if ($errorCode == 401)
		echo 'Bad auth key';
	else
		echo 'Other error : <br />'.$ex->getMessage();
					// Affiche un message associé à l'erreur
}
```

Cela signifie que chaque création ou utilisation de la librairie doit se situer dans un bloc "try", le bloc "catch" permet ensuite de gérer l'erreur si elle survient lors de l'execution du bloc try.

Maintenant, nous allons voir comment lister tous les clients via le service web puis nous verrons les 4 méthodes de CRUD.

3.3 Lister les clients

Nous allons voir ici comment afficher la liste des ID des clients, nous aurions pu afficher plus d'information et personnaliser cela mais nous verront cela plus tard dans le tutorial.

Afin de récupérer un fichier XML contenant l'ensemble des clients nous devons utiliser la méthode "get" qui prends en premier argument un tableau défini comme suit :

Clé	Valeur
resource	customers

La valeur définit la ressource que le service web va utiliser dans son futur appel, elle aurait pu être de type carriers, countries ou tout autre type de ressource que l'ont peux trouver dans l'onglet service web du « Back-Office ».

Exemple :

```
$opt['resource'] = 'customers';
$xml = $webService->get($opt);
```

L'appel à la méthode nous renverra un objet **SimpleXML** contenant l'ensemble des identifiants des clients.
Maintenant nous devons accéder aux balises qui nous interessent dans le fichier XML :

Structure:

```
<?xml>
<prestashop>
<customers>
      <customer>
      ID du client
      </customer>
...   Autres balises client
</customers>
</prestashop>
</xml>
```

En récupérant le retour de "$webService->get", nous sommes à la racine du document.
Afin d'accéder aux champs des clients qui sont enfants (« children » en anglais) de la balise customers, il nous suffit en SimpleXML de récupérer l'ensemble des champs dans un tableau associatif comme ceci :

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

A partir de là nous pouvons accéder aux identifiants des clients facilement, exemple avec un parcours des identifiants :

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

Grâce à ces éléments, vous allez créer un tableau (HTML) contenant tous les ID des clients avant de passer au chapitre suivant.
Vous pouvez vous aider du Back Office dans l'onglet "Clients" afin de retrouver les ID de tous les clients.
Si vous rencontrez des difficultés, n'hésitez pas à regarder le code du fichier "0-CustomersList.php", c'est le résultat que vous devez obtenir.

Chapitre 4 - Récuperer des données : Récupérer un client

Objectif : Une application WEB permettant de lister et d'afficher les informations d'un client
Difficulté : *
Problématique : Comment créer un système qui permet à partir des identifiants des clients de récupérer les fiches des clients ?

Préparation :

Dupliquez le fichier lister_les_clients.php de l'étape précédente vers un fichier nommé R-CRUD.php à la racine de votre serveur WEB.
Si vous n'avez pas réussi l'étape précédente, dupliquez le fichier 0-CustomersList.php vers un fichier nommé R-CRUD.php.

Dans le fichier **XML** contenant la liste des clients, nous retrouvons l'ensemble des XLink permettant d'accéder aux informations d'un client.

Exemple :

```
<customers>
<customer id="1" xlink:href="http://exemple.com/boutique/api/customers/1" />
</customers>
```

Ici, on voit que la balise "customer" dont l'ID est 1 a pour xlink :
"http://maboutique.com/api/customers/1"
Ce lien nous amène à un fichier XML contenant les informations sur le client ayant comme ID 1.

Pour ce tutorial, afin de gérer l'accès aux différents clients vous allez procéder de manière à associer les pages aux identifiants des clients via un paramètre GET nommé "id".

Exemple :
La page "http://maboutique.com/R-CRUD.php?id=1" nous affichera la fiche du client 1.

Modifiez votre tableau créé dans le chapitre précédent pour lui ajouter un lien vers les futurs fiches client.

Vous allez devoir isoler l'affichage de la liste de l'affichage d'un client en particulier.

Pour se faire, il vous faut isoler l'affichage de votre liste en vérifiant à l'aide de isset si le paramètre GET "id" n'est bien pas présent lors de l'affichage de votre liste.

L'appel au service web est exactement le même que pour l'affichage de la liste à l'exception prêt qu'il faut ajouter un élément id au tableau et ayant pour valeur l'id d'un client.

Nous sommes ici dans une utilisation de la ressource « customers » ou « client », si nous aurions été en train de modifier la ressource pays « countries » cette id aurait été un id de pays.

```
$opt['resource'] = 'customers';
$opt['id'] = $_GET['id'];
$xml = $webService->get($opt);
```

Conseil : Utiliser « isset » avant de définir un ID vous permettra de réaliser facilement ce chapitre.

L'accès aux ressources se fait comme précédemment pour l'affichage de la liste car les balises qui nous intéressent sont enfants de la balise « customers ».

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

Le parcours quand à lui se fait d'une autre manière (ici dans un tableau HTML) :

```
foreach ($resources as $key => $resource)
    echo 'Nom du champ : '.$key.' - Valeur : '.$resource.'<br />';
```

Vous avez désormais tout le nécessaire pour réaliser un script permettant à la fois de lister et à la fois d'afficher les informations d'un client en particulier.

Essayez de créer ce script "R-CRUD.php", si vous rencontrez des difficultés prenez exemple sur le fichier "1-Retrieve.php" qui correspond au résultat que vous devriez obtenir.

Nous verrons dans un autre tutorial comment filtrer, trier et limiter le nombre d'éléments affichés dans la liste.
Si vous êtes pressé d'implémenter ces fonctionnalités, vous trouverez davantage d'information au chapitre 8.

Chapitre 5 - Modification : Mettre à jour un client

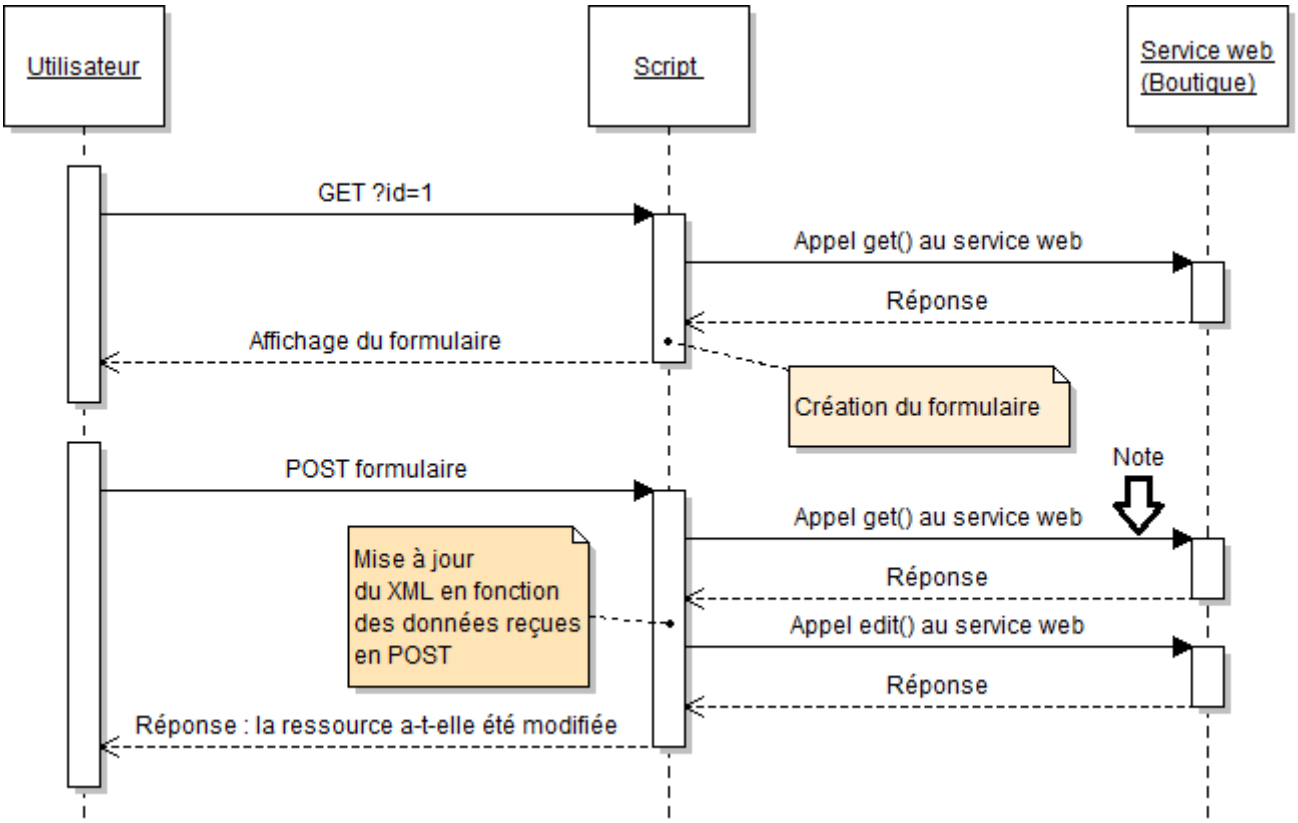
Objectif : Une application WEB permettant de lister et de mettre à jour les informations d'un client.
Difficulté : ***

Préparation :

Dupliquez le fichier lister_les_clients.php de la section 3.3 vers un fichier nommé U-CRUD.php à la racine de votre serveur WEB.

La mise à jour des ressources via le service web est complexe, nous allons donc d'abord expliquer son fonctionnement.

Diagramme de séquence représentant la mise à jour d'une ressource :



Nous pouvons voir que le diagramme se décompose en 2 étapes :
- Récupération de la ressource à un id définit (1 dans le diagramme) et création du formulaire.
- Mise à jour de la ressource.

Note (Flèche vers le bas sur le diagramme) :
A l'endroit de la flèche, nous trouvons un « get », qui correspond à une récupération de ressource.

Cette étape est importante car il nous est nécessaire de récupérer à nouveau le fichier XML afin de le faire correspondre avec les données envoyées par le formulaire avant d'appeler « edit » pour mettre à jour la ressource.

Notez que nous aurions pu faire autrement en envoyant un XML modifié à l'aide de javascript et ainsi ne pas avoir de « get » dans ce processus.

Etape 1 : Récupération et création du formulaire

Récupération du fichier XML et affichage du formulaire :

```
// Définition de la ressource
$opt = array('resource' => 'customers');
// Définition de l'ID de ressource à modifier
$opt['id'] = $_GET['id'];
// Appel du service web, récupération du fichier XML
$xml = $webService->get($opt);
// Récupération des éléments de la ressource dans une variable (tableau)
$resources = $xml->children()->children();
// Formulaire client
```

Ici , L'appel est similaire à la récupération de données, c'est cette appel qui va nous permettre de créer le formulaire.

Nous allons générer le formulaire de mise à jour automatiquement.

Pour ce formulaire, utilisons des balises HTML « input » ayant comme « name » le nom de l'attribut et comme « value » la valeur de celui ci.

Afin de ne pas perdre l'id pour la 2ème étape selon le diagramme, le formulaire va pointer sur :

?id= « Id du client »

Ainsi nous le récupèreront comme ceci :

$_GET['id'] ;

Nous aurions pu faire autrement comme de passer également cette ID en POST, mais vous verrez que cela va simplifier le traitement par la suite.

Etape 2 : Mise à jour de la ressource

Dans un premier temps, comme vous pouvez le voir dans le diagramme là ou se trouve la flèche « Note », nous allons récupérer le fichier XML, pour ceci, vous allez réaliser le même appel que pour la création du formulaire.

Si vous avez bien spécifié comme indiqué précédemment la destination du formulaire avec un id, votre appel devrait déjà se faire et le formulaire se réafficher.

Aide pour la création du formulaire :

```
foreach ($resources as $key => $resource)
{
        echo '<tr><th>'.$key.'</th><td>';
        echo '<input type="text" name="'.$key.'" value="'.$resource.'"/>';
        echo '</td></tr>';
}
```

Une fois le fichier XML récupéré il nous faut modifier les nouvelles données avec les données reçu en POST.

Parcours des clés dans le fichier XML et mise à jours des valeurs :

```
foreach ($resources as $nodeKey => $node)
{
        $resources->$nodeKey = $_POST[$nodeKey];
}
```

Nous disposons maintenant d'un fichier XML mis à jour, il ne nous reste plus qu'a l'envoyer

Exemple d'une mise à jour :

```
$opt = array('resource' => 'customers');// Définition de la ressource
$opt['putXml'] = $xml->asXML();          // Définition du fichier XML
    // L'appel de asXML() retourne une chaine correspondant au fichier
$xml = $webService->edit($opt);          // Appel
```

Essayez maintenant de créer dans votre script "U-CRUD.php" la modification d'un client avec un ID définit dans le code, puis pour tous les clients.

Vérifiez à l'aide de "R-CRUD.php" que les informations ait bien été modifiées puis rendez dynamique l'ID du client.

Si vous rencontrez des difficultés, regardez le code de 2-Update.php.

Chapitre 6 - Création : Formulaire d'ajout à distance

Objectif : Une application WEB permettant de lister et de créer un nouveau client.
Difficulté : **

Préparation :

Dupliquez le fichier lister_les_clients.php de la section 3.3 vers un fichier nommé C-CRUD.php à la racine de votre serveur WEB.

L'ajout de ressource peut s'apparenter à une mise à jour à partir d'un élément vide.

Mais comment récupérer un XML formaté en tant que client vide ?

Dans le service web, il existe une méthode pour récupérer un XML vide, elle est accessible grâce à une URL formatée comme suit :

```
http://maboutique.com/api/[Nom de la ressource]?schema=blank
```

> Note :
> Il est possible de remplacer la valeur du paramètre schema « blank » par « synopsis » afin de récupérer davantage d'informations sur les champs de la ressource.

Comme nous l'avons vu dans la section 3.3 (Lister les clients) il est possible de passer comme tableau de paramètre à "get", "resource" et "id", il est également possible de ne spécifier qu'une url de cette façon :

```
$xml = $webService->get(array('url' =>
'http://maboutique.com/api/customers?schema=blank'));
```

Ici, nous récuperons dans la variable XML l'intégralité d'un client vide.

Début du fichier XML récupéré :

```
<prestashop>
<customer>
<id_default_group/>
etc...
```

Nous pouvons ensuite grâce aux nom des champs que nous avons, créer un formulaire associé.

Récupération de l'ensemble des champs :

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

Parcours de l'ensemble des champs et partie de la création dynamique des champs du formulaire dans un tableau :

```
foreach ($resources as $key => $resource)
{
        echo '<tr><th>'.$key.'</th><td>';
        echo '<input type="text" name="'.$key.'" value=""/>';
        echo '</td></tr>';
}
```

Une fois les données passées en POST, nous allons associé les données envoyé avec le fichier XML vierge, cette technique est la même que pour la mise à jour de données.

```
foreach ($resources as $nodeKey => $node)
{
        $resources->$nodeKey = $_POST[$nodeKey];
}
```

L'appel au service web est quand à lui ressemblant à ce que nous avons pu voit précédemment :

```
$opt = array('resource' => 'customers');
$opt['postXml'] = $xml->asXML();
$xml = $webService->add($opt);
```

Réalisez maintenant un script qui permet d'ajouter un client, pensez que certains champs sont obligatoires, il ne faut pas l'oublier.

Si vous rencontrez des difficultés, regardez le code de "3-Create.php".

Chapitre 7 - Suppression : Retirer des comptes client de la base

Objectif : Une application WEB permettant de lister et de supprimer des clients.
Difficulté : *

Préparation :

Dupliquez le fichier lister_les_clients.php de la section 3.3 vers un fichier nommé D-CRUD.php à la racine de votre serveur WEB.

Pour cette dernière partie, nous allons voir la suppression de ressource.
Voici l'appel complet détaillé que vous devez faire pour supprimer un client :

```
try
{
	$webService = new PrestaShopWebservice('http://maboutique.com/',
	'ZR92FNY5UFRERNI3O9Z5QDHWKTP3YIIT', false);// Création d'une
                                                    // instance
	$opt['resource'] = 'customers';		// Ressource à utiliser
	$opt['id'] = 3;				// ID à utiliser
	$webService->delete($opt);		// Suppression
	echo 'Client '.3.' supprimé avec succès !';	// Si nous avons pu afficher le
                                                    // message c'est que nous n'avons
                                                    // pas quitté le bloc try.
}
catch  (PrestaShopWebserviceException $ex)
{
	$trace = $ex->getTrace();		// Récupère toutes les
                                                    // informations sur l'erreur
	$errorCode = $trace[0]['args'][0];	// Récupération du code d'erreur
	if ($errorCode == 401)
		echo 'Bad auth key';
	else
		echo 'Other error : <br />'.$ex->getMessage();
                                                    // Affiche un message d'erreur
}
```

Ce code permet la suppression d'un client ayant pour ID 3, comme vous pouvez le remarquer la suppression ne diffère que de peu de la récupération d'une resource. En effet la seule chose différence au niveau du code se situe au niveau de la méthode appelée.

Nous n'appelons plus la méthode « get » mais la méthode « delete », tout simplement !

Vous devez maintenant remplacer l'ID du client par un ID définit dynamiquement.

Réalisez maintenant l'ensemble le script qui permettra d'afficher la liste des ID des clients et de supprimer un client au choix.

Encore une fois, si vous rencontrez des difficultés regardez le code de "4-Delete.php".

Chapitre 8 – Utilisation avancée

Option de rendu

Inclure tous les champs de la ressource produit « products »

URL :

```
« URL de la boutique » /api/products/?display=full
```

PHP :

```
$opt = array('resource' => 'products', 'display' => 'full');
```

N'inclure que l'ID de tous les transporteurs « carriers »

URL :

```
« URL de la boutique » /api/products/
```

PHP :

```
$opt = array('resource' => 'products');
```

N'inclure que les champs « name » et « value » de la ressource « configurations »

URL :

```
« URL de la boutique » /api/configurations/?display=[name,value]
```

PHP :

```
$opt = array('resource' =>'configurations', 'display' => '[name,value]');
```

Filtres de rendu

N'inclure que les noms et prénoms des clients « customers » ayant l'id 1 et 5

```
« URL de la boutique » /api/customers/?display=[firstname,lastname]&filter[id]=[1|5]
```

PHP :

```
$opt = array('resource' =>'customers', 'display' => '[firstname,lastname]', 'filter[id]' =>
'[1|5]');
```

N'inclure que les noms des clients « customers » ayant un id compris entre 1 et 10

```
« URL de la boutique » /api/customers/?display=[lastname]&filter[id]=[1,10]
```

PHP :

```
$opt = array('resource' =>'customers', 'display' => '[lastname]', 'filter[id]' => '[1,10]');
```

N'inclure que la date de naissance du client ayant pour nom « John » et prénom « DOE »

```
« URL de la boutique »
/api/customers/?display=[birthday]&filter[firstname]=[John]&filter[lastname]=[DOE]
```

PHP :

```
$opt = array('resource' =>'customers', 'display' => '[birthday]', 'filter[firstname]' => '[John]',
'filter[lastname]' => '[DOE]');
```

N'inclure que les noms des constructeurs « manufacturers » dont le nom commence par « Appl »

```
« URL de la boutique » /api/manufacturers/?display=[name]&filter[name]=[appl]%
```

PHP :

```
$opt = array('resource' => 'manufacturers', 'display' => '[name]', 'filter[name]' => '[appl]%');
```

Filtres de tri

Trier les clients « customers » en ordre alphabétique du nom

```
« URL de la boutique » /api/customers?display=full&sort=[lastname_ASC]
```

PHP :

```
$opt = array('resource' => 'customers', 'display' => 'full', 'sort' => '[lastname_ASC]');
```

Filtres de limitation de rendu

N'inclure que les 5 premiers états « states »

```
« URL de la boutique » /api/states/?display=full&limit=5
```

PHP :

```
$opt = array('resource' => 'states', 'display' => 'full', 'limit' => '5');
```

N'inclure que les 5 éléments à partir du 10ème élément de la ressource état « states »

```
« URL de la boutique » /api/states/?display=full&limit=9,5
```

PHP :

```
$opt = array('resource' => 'states', 'display' => 'full', 'limit' => '9,5');
```

Mémento : Notions énoncées dans ce tutorial

Méthode

Afin de vous aider dans vos premiers pas avec le service web, voici un petit mémo des techniques utilisés dans ce tutorial.

			Paramètre(s) de la méthode				
	REST	Méthode	url	resource	id	postXml	putXml
C	POST	add	X	X		X	
R	GET	get	X	X	X		
U	UPDATE	edit	X	X	X		X
D	DELETE	delete	X	X	X		

Si le paramètre url est spécifié, aucun autre paramètre ne peut être utilisé et vice versa.

Options

Clé	Suffixe de clé		préfix	Valeur	Suffixe	Description
display				[champ1,champ2 …]		N'afficher que les champs entre crochet
display				full		Afficher tous les champs

Clé	Suffixe de clé		préfix	Valeur	Suffixe	Description
filter	[champ]			[valeur1|valeur2]		Filtrer « champ » par valeur comprise entre « valeur1 » et « valeur2 »
filter	[champ]			[valeur]		Filtrer champ par la valeur « valeur »
filter	[champ]			[valeur1,valeur2…]		Filtrer champ pour les valeurs spécifiées entre crochet
filter	[champ]		%	[valeur]	%	Filtrer « colonne » pour les valeurs contenant « valeur »

Clé	Suffixe de clé		préfix	Valeur	Suffixe	Description
sort				[champ1_ASC,champ2 _DESC,champ3_ASC]		Trier par champ avec le suffixe _ASC ou _DESC selon l'ordre souhaité
sort				full		Afficher tous les champs

Clé	Suffixe de clé		préfix	Valeur	Suffixe	Description
limit				Nombre		Limiter le résultat à « Nombre »
limit				Index de départ, Nombre		Limiter le résultat à « Nombre » à partir de « Index »