JSON

**JSON** (*JavaScript Object Notation*) est un format de données textuelles, générique, dérivé de la notation des objets du langage ECMAScript. Il permet de représenter de l’information structurée comme le permet XML par exemple. Créé par Douglas Crockford, il est décrit par la RFC 4627 de l’IETF. C’est un format léger d’échange de données. Il est facile à lire ou à écrire pour des humains. Il est aisément analysable ou générable par des machines.

Mon tuteur de stage a reçu une demande de la part de l’entreprise commercialisant des carnets de chèque cadeaux. Le client souhaitait qu’un module structure des informations extraite de la base de données au format JSON. Ce module sera appelé par un autre serveur, qui se servira des données généré par le module pour pouvoir être traité ensuite. Lors de l’appel par le serveur de ce module, il doit communiquer les dates de début et de fin sur lesquelles doit porter l’extraction de données.

La génération des informations au format JSON est relativement simple. En effet une fonction native de php, nommé « json\_code », dans laquelle on passe un tableau associatif, permet de créer le JSON de ce que contient ce tableau. Le tableau que je passe en paramètre est l’extraction du tableau dans le tableau retournée par la fonction « result\_array » de la framework codeigniter.

Pour la communication le serveur appelant le module que je viens de créer, mon tuteur de stage me conseille de regarder ce qu’est le « REST » (REpresentational State Transfer). C’est un style d’architecture pour les systèmes hypermédia distribués, créé par Roy Fielding en 2000 dans le chapitre 5 de sa thèse de doctorat. REST n’est pas un protocole (tel que HTTP) ou un format. Ce style d'architecture est particulièrement bien adapté au World Wide Web mais n'en est pas dépendant. Les contraintes, telles que définies par Roy Fielding, peuvent s'appliquer à d'autres protocoles d'application que HTTP.

Après de longue de recherche, et une difficile compréhension, je me demande un peu d’aide auprès de mon tuteur. Alors il a demandé à collaborateur d’Archriss qui se trouve à Annecy, par le biais de skype. Il lui renvoie un code qu’il a réalisé et me le communique. En étudiant ce code, je comprends que l’utilisation de la fonction « file\_get\_contents » (Lit tout un fichier dans une chaîne) en passant en paramètre une page web, permet de récupérer tout le code HTML de la page. Il suffit alors d’afficher, à l’aide d’un « echo » ou « print\_r » pour que le serveur réaffiche les données en utilisant un navigateur. Mais la compréhension du « REST » n’est pas claire pour moi. Après lecture de ce [site](http://www.figer.com/publications/REST.htm), les éléments commence à se mettre en place dans ma tête, et la situation commence à de décanter.

Suite à quoi, je comprends que c’est une communication client/serveur, et que le module que je crée est la partie serveur. Mais pour la réalisation de mes tests, je devais avoir un script client qui appelait mon script serveur. Mon tuteur de stage me conseille de réadapter le script qu’il m’envoyé, et d’utiliser le partie « CURL ». PHP supporte libcurl, une bibliothèque créée par Daniel Stenberg, qui vous permet de me connecter et de communiquer avec différents types de serveurs, et ce, avec différents types de protocoles. Dans mon script coté serveur, j’affiche toujours mes informations extraire de la base, au format JSON. Les tests avec le « CURL » sont concluants.

Il fallait à présent communiquer les dates au serveur pour extraire les commandes concernées. Ces dates sont très variables, et c’est le script client qui décide de cette plage de date. Pour cela je communique les informations des dates dans l’URL. En effet, je respecte la nomenclature permettant de communiquer des informations dans une URL, c’est-à-dire :

* <http://www.example.com/scriptServeur.php?nomVal1=valeur1&nomVal2=valeur2>

Je commence l’ajout des informations par un « ? », donne un nom à la valeur, saisi la valeur elle-même, puis je sépare chaque info par un « & ».

Une fois que je communique correctement mes informations, il fallait que je les récupère dans le script du côté serveur. Comme je passe les informations dans l’url, j’espérai pouvoir les récupérer dans la variable globale « $\_GET ». Mais pour une raison inconnue, cela ne fonctionne pas. Grâce à la variable globales « $\_SERVER », j’arrive à récupérer l’URL, contenant les informations. J’ai donc créé une fonction qui retravaille cette chaîne, avec l’utilisation des fonctions « strpos » et « substr ». Une fois les informations extraites de l’URL, je les stocke dans un tableau, que la fonction retourne. Ce tableau sera utilisé lors de l’appel de la fonction « query » de la framework « codeigniter », qui prend en paramètre un tableau des valeurs utilisé par la requête. Après des tests concluants, mon script coté serveur renvoie bien les commandes concernées au format JSON.