TP 1 - WebServices Consommateur

Contexte

Rendez-vous sur le site REST Countries (**`https://restcountries.com/`**).

Il s'agit d'un site hébergeant un Web Service public sur les pays du monde. Le site contient aussi une description de l'API de consultation. Elle fonctionne en mode REST.

Dans ce TP, nous utiliserons la version **`3.1`** de l'API.

Exercice 1 - Navigateur

Premières requêtes

Puisqu'il ne s'agit que de requêtes **`GET`** (car vous ne pouvez rien modifier sur ce site, évidemment), vous pouvez donc consommer cette API directement avec un navigateur Web car celui-ci est capable de faire de telles requêtes **`GET`**, et c'est d'ailleurs à peu près tout ce dont il est capable...

Consultez la documentation pour récupérer la liste des pays du monde et affichez-là dans un nouvel onglet de votre navigateur.

Le résultat est formaté en **`JSON`** mais il est très peu pratique à lire dans le navigateur car il n'est pas mis en forme pour des raisons de compacité et de gain de bande passante. Nous allons y remédier.

Notez déjà que si vous utilisez Firefox, il est capable de formater proprement du **`JSON`**. Certains navigateurs offrent peut-être aussi cette fonctionnalité, nativement ou à l'aide d'un plugin, mais voici une autre possibilité.

Sélectionnez l'intégralité de l'affichage **`JSON`** (**`CTRL`**+**`A`**), copiez-le (**`CTRL`**+**`C`**) et collez-le (**`CTRL`**+**`V`**) dans une fenêtre vierge du logiciel **VSC**.

Sauvez le contenu de votre fenêtre (de **VSC**) avec un nom portant une extension **`.json`**.

Si le document n'est pas mis en forme automatiquement, rendez-vous dans le Menu **`Affichage > Palette de commandes`** et tapez **`Format`** dans la zone de saisie. Vous devriez y voir un choix **`Mettre le document en forme`**.

Si vous ne trouvez pas cette option dans le menu, installez l'extension **`Prettier`** et recommencez (Menu **`Affichage > ...`**).

Ainsi remis en forme, le fichier doit être plus lisible et accessible.

Requêtes supplémentaires

Maintenant que vous avez réussi à afficher proprement la liste des pays du monde, faites de même avec ces requêtes, en cherchant dans la documentation :

- La liste des pays ayant l'Euro comme monnaie (le code de l'Euro est **`eur`**)
- La liste des pays parlant le français
- La liste des pays d'Asie
- La liste des pays ayant Paris comme capitale

Chrome - Outils de développement

Pour cet exercice, vous devez utiliser le navigateur **Chrome**. Ce qu'on va expérimenter peut certainement se faire avec d'autres navigateurs mais les instructions qui suivent sont prévues pour Chrome.

En principe, les outils de développement de Chrome sont déjà installés. Ils sont disponibles via le menu **`Afficher > Options pour les développeurs > Outils de développement`**. Affichez-les.

Les outils qui nous intéressent ici sont placés dans l'onglet **`Réseau`** (il est peut-être caché dans le menu **`>>`** des outils de développement que vous avez fait apparaître).

Relancez l'affichage de la liste des pays ayant Paris comme capitale.

Vous devez voir apparaître le résultat et dans liste de l'onglet **`Réseau`**, vous devez aussi voir apparaître plusieurs lignes, dont une nommée **`paris`** qui est la page de résultat (**`JSON`**).

Cliquez sur la ligne **`paris`** et observez le contenu de la fenêtre des **`En-têtes`** dans vos outils de développement. Identifiez les éléments suivants :

- Mode de requête (méthode)
- Code d'état (code de **`HTTP`** de la réponse)
- Content-type (format de la réponse)

Notez au passage que vous disposez aussi ici d'une fenêtre (un onglet) **`Aperçu`**, qui vous présente les données bien formatées et facilement accessibles sans obligatoirement avoir besoin de passer dans **VSC**.

Refaites la même expérience en remplaçant **Paris** par **Lannion** et regardez ces mêmes éléments d'**`En-Têtes`**, notamment le Code d'état.

Exercice 2 - Postman

Un navigateur Web, c'est bien pour faire des **`GET`** mais vous allez être rapidement bloqués pour les autres actions (**`POST`**, **`PUT`**, **`DELETE`** etc) avec d'autres API de WS.

Heureusement, il existe un outil très pratique que vous allez utiliser à la place de votre navigateur pour tester les WS. Il s'agit de **Postman** (**`https://www.postman.com`**).

Il est, en principe, déjà présent sur vos ordinateurs. Si ce n'est pas le cas, téléchargez et installez l'application **Postman** vous-même. Vous n'avez pas besoin de créer de compte sur le site, ni pour télécharger, ni pour l'utiliser ensuite, mais si vous voulez en créer un, libre à vous.

Suivez les étapes suivantes pour tester, dans Postman, la récupération de la liste des pays du monde (voir **Exercice 1**) :

1. Cliquez sur le bouton **New**
2. Choisissez **Collection** et donnez-lui un nom (**RESTcountries** par exemple)
3. Cliquez encore sur le bouton **New**
4. Choisissez **HTTP Request**
5. La **Request** s'affiche alors et vous pouvez la paramétrer. Choisissez la méthode **`GET`** (qui doit être la méthode par défaut déjà choisie)
6. Saisissez l'**URL** que vous avez utilisée à l'**Exercice 1** pour récupérer la liste des pays du monde.
7. Faites **Save** pour sauver votre **Request** en choisissant de la placer dans votre collection nouvellement créée. Donnez-lui un nom (**Countries** par exemple).
8. Puis faites **Send**
9. Vous devriez avoir l'affichage de la réponse en **`JSON`**

Observez comment est construit cet objet **`JSON`**.

Les **`[]`** représentent un tableau

Les **`{}`** représentent un objet (avec ses attributs à l'intérieur).

Les attributs sont de la forme **`key: value`**

Observez aussi le code de retour qui doit être **`200 OK`**.

Faites un essai avec une requête récupérant la liste des pays ayant **Lannion** comme capitale. Observez le code de retour et la réponse.

Notez que cette API renvoie aussi un **`body`** en cas d'erreur (**`code != 200`**). Ce n'est pas le cas de toutes les API et il est assez commun de n'avoir qu'un simple code pour identifier une erreur.

Postman est très utile pour tester des API. N'hésitez pas à en faire usage dès que possible !

Dans les exercices suivants, si l'accès au site **`https://restcountries.com/`** est lent, et uniquement dans cette situation, vous pouvez utiliser notre image Docker, clone du site officiel. Voici comment procéder dans un Terminal :

```
docker image pull r401-restcountries:1.0
docker container run --rm -p 8081:8080 r401-restcountries:1.0
```

Sur vos PC personnels, utilisez l'image **`bigpapoo/r401-restcountries:1.0`**

Dans un navigateur Web, testez que **`http://localhost:8081`** affiche bien le site cloné localement. Dans vos scripts, utilisez cette URL au lieu de celle vers le site officiel.

Vous remplacerez ensuite **`https://restcountries.com`** par **`http://localhost:8081`**

Exercice 3

Il y a une fonction très utile qui évite l'ouverture, la lecture et la fermeture d'un fichier et qui permet de lire l'intégralité d'un fichier en un seul appel de fonction. Il s'agit de **`file_get_contents()`**. Consultez la documentation **PHP** pour voir ce qu'elle fait et comment l'utiliser.

En outre, **PHP** sait "ouvrir" une URL comme s'il ouvrait un fichier présent physiquement sur disque en utilisant les mêmes fonctions que pour les fichiers !

Ainsi, il est possible d'utiliser **`file_get_contents()`** pour "ouvrir" une URL et lire le contenu d'une page Web directement dans une chaîne de caractères.

Vous allez tester ça avec l'URL qui affiche les pays du monde (**Exercice 1** et **2**).

Question 1

Écrivez un script **PHP** qui charge la liste des pays et affiche simplement ce qui est ainsi récupéré. Ça doit être difficilement lisible, c'est normal.

Ce qui suit à propos du proxy n'est à prendre en compte que si vous accédez au site **`https://restcountries.com/`**. Si vous utilisez l'option d'un conteneur Docker en local, ne faites rien et sautez cette partie.

A l'IUT, pour que les fonctions réseau de **PHP** comme, par exemple, **`file_get_contents()`** puissent utiliser le proxy, il faut ajouter ceci en tête de script :

```
stream_context_set_default([
    'http'=>[
        'proxy'=>'129.20.239.9:3128'
    ]
]);
```

Question 2

Ce que vous récupérez est du **JSON**, pas du format **PHP**. Vous allez maintenant convertir cette chaîne **JSON** en quelque chose au format **PHP**. La fonction qui va vous y aider est **`json_decode()`**. Consultez l'aide de cette fonction. Identifiez aussi comment faire en sorte que **`json_decode()`** retourne un tableau associatif.

Testez cela en prenant soin de placer l'affichage entre **`<pre>...</pre>`** pour plus de lisibilité du résultat dans votre navigateur.

Question 3

Ecrivez maintenant un script **PHP** qui affiche tous les pays (ou les X premiers si vous préférez) dans une **`<TABLE>...<TABLE>`** par exemple. Vous afficherez le champ "Nom du pays", le code "CCA3" et la "Capitale".

Exercice 4

Sur la base de l'**Exercice 3**, placez un formulaire en début de page avec un champ "Nom du pays" et un bouton "Chercher". L'appui sur le bouton doit soumettre le formulaire pour afficher les informations du pays demandé.

Exercice 5

Améliorez votre page **`HTML`** de l'**Exercice 3** pour afficher en plus l'image du drapeau. Il y a un champ avec une URL vers l'image du drapeau dans le **`JSON`** récupéré.

Exercice 6

Sur la base de l'**Exercice 5**, proposez un formulaire de filtrage de la liste par langue ou par devise. Faites en sorte que la liste qui s'affiche soit paginée par blocs de 10, c'est à dire qu'elle affiche aussi un bouton **Suivant** et un **Précédent** pour se déplacer de 10 en 10 (et un paramètre dans le code est mieux qu'une valeur en dur).

Les critères de filtrage n'ont pas à être combinés car vous ne disposez d'aucune API le permettant dans ce WS. Vous pourriez choisir de tout récupérer et de faire votre propre filtrage mais ce n'est pas l'objectif visé dans cet exercice, ni la finalité d'une API WS.

Au final, vous devez donc permettre de filtrer soit par nom de pays, soit par devise, soit par langue.