TD 8 – JSON et C#

Préambule [JSON]

Un document JSON (*c.f.* `https://fr.wikipedia.org/wiki/JavaScript_Object_Notation`) est un document texte qui a pour fonction de représenter de l'information structurée à l'aide d'étiquettes permettant d'en interpréter les divers éléments...
Ceci dans la même idée que XML, mais les balises et leurs contenus sont remplacés par des paires noms/valeurs (entre accolades, séparés du caractère deux-points) ; de plus il est possible en JSON de définir des ensembles (listes) de paires noms/valeurs (tableaux entre [ ]).

Partie I : Les documents JSON

Exercice 1 : Ecrire un document JSON

Soient les informations suivantes issues d'un agenda papier un peu brouillon (le même que celui du TD6)

Noé, Léa et Charles DUPONT habitent au 42 rue de la Paix 75001 Paris, 01 23 45 67 89. Léa dispose aussi d'un mobile : 06 11 22 33 44. Paul MARTIN et Jeanne DURANT, épouse MARTIN, habitent 1 rue de la gare à Trifouilly-les-oies, 03 54 76 98 21. Paul pour le boulot (numéro de fax 04 04 04 04 04) habite de temps en temps au 7 rue du parc à Lyon, il est joignable au 06 76 54 32 10 ; Jeanne a pour tél mobile le 07 88 99 11 22. Marie NICOLAS habite 13 rue de la République à 92400 Courbevoie, 01 54 07 64 82, 06 94 88 72 13.

On vous demande d'organiser sous forme d'un document JSON les données de l'agenda décrit ci-dessus.
Il s'agit (comme au TD6) d'un agenda composé de contacts multiples
Vous utiliserez le logiciel JSONView pour éditer un tel document JSON (fichier.json) ;
=> menu ''File" > "New" pour accéder à l'editeur

Dans l'éditeur de JSONView vous disposerez des boutons
- *Validate JSON* - pour valider la conformité (vis à vis des règles d'écriture JSON) du document
- *Format JSON* - pour mettre en forme le document suivant les règles JSON

Exercice 2 : Exploiter un document JSON

Vous trouverez sur Moddle le fichier : ***itineraireOptimize.json***

Ce fichier est la réponse de l'api GoogleMap à la requête suivante :
`https://maps.googleapis.com/maps/api/directions/json?origin=courbevoie,SA&destination=meudon&waypoints=optimize:true|Vanves|Cachan|Fresnes|Montrouge&key=AIzaSyBgToCfsJqVC9kfeSJem25nMY9hoA`

Cette requête calcule l'itinéraire optimisé entre Courbevoie et Meudon en passant par les villes de Vanves, Cachan, Fresnes, Montrouge.
L'optimisation est une application du problème du voyageur de commerce basée sur le temps de trajet comme facteur principal. L'ordre optimisé des points de passage est founi (avec une valeur de base zéro) dans un champ waypoint_order.

En utilisant les outils de visualisation à votre disposition, quels sont les numéros et les noms des villes constituant le trajet dans l'ordre optimisé
(les numéros des villes intermédiaires correspondent à l'ordre dans lequel elles sont saisies dans la requête GoogleMap.

Exercice 3 : Les outils online

1. **Validateurs JSON**
   Vous disposez sur le net de nombreux outils permettant de valider la conformité d'un document JSON. Par exemple `https://jsonlint.com/`

   => Utilisez ce validateur pour corriger le fichier JSON (fourni sur Moodle) :
   couleurs.txt

   Ouvrir le fichier dans un éditeur de votre choix (Notepad+ ou Wordpad) et recopier son contenu dans le validateur JSON.

2. **Conversion XML <-> JSON**
   Par exemple `http://www.utilities-online.info/xmltojson`

   => Utilisez ce convertisseur sur le carnet d'adresses XML fourni sur Moodle :
   adresses.xml (c'est celui généré lors du TD6 sur XML)

   Ouvrir le fichier dans un éditeur de votre choix (Notepad+ ou Wordpad) et recopier son contenu dans le convertisseur XML<=>JSON.

   Effectuez sa conversion en fichier JSON et comparez avec le fichier généré dans l'exercice 1.

3. **JSONPath evaluator**
   Vous trouverez aussi des évaluateurs de requête JSONPatlh.
   par exemple : `http://www.jsonquerytool.com`

   attention : Utiliser l'implémentation (le Query Type) JSONPath de Goessner 0.8.0

   Vous utiliserez comme document JSON le document fourni sur Moodle :
   chiens.txt

   Ouvrir le fichier dans un éditeur de votre choix (notepad + ou Wordpad), recopier son contenu dans la fenêtre JSON de l'évaluateur. Vos requêtes sont à écrire dans la fenêtre Extraction Query.

Syntaxe JsonPath dans le Framework Json de Goessner 0.0.0 :

$	l'élément root
@	l'élément courant
.	child operator
[ ]	child operator recusif
..	recursive descent
*	tous les éléments (indépendament de leur nom)
[ ]	subscript operator , utilisé pour itérer(récursivement) les éléments d'un tableau ex : livres [2] ou avec un filtre ex : livres [?(.....)]
[start : end : step]	slice operator
?(predicat)	expression d'un filtre, ex : ?(titre == "C facile")
prédicat	utilisent ==, <, >, &&, ||

=>Liste de requêtes à écrire et à exécuter

1. le document complet
2. les clés/valeurs du tableau chiens
3. le nom du 2ème chien de la liste (Figaro pour vous permettre de vérifier votre réponse)
4. le nom du dernier chien de la liste
5. tous les noms de chien (essayer les opérateurs . , [ ], .. et .[ ])
6. les informations sur le chien Taiga
7. idem sur le chien Figaro
8. le nom du propriétaire de Taiga
9. le nom des chiens femelle
10. toutes les infos sur les chiots de Loukhoum
11. le nom des chiots de Loukhoum
12. le nom des chiens qui ont des chiots
13. le nom des chiens qui n'ont pas de chiots
14. le nom de tous les chiots
15. les noms des seuls chiens (4) de la liste, pas les chiots (3 versions : avec slice operator, joker ou filtre)
16. le nom des chiens nés avant l'année 2000
17. le nom des chiens femelles nés avant l'année 2000

Partie II : Lecture/écriture depuis C#

Dans cette partie du TD, vous avez à créer un projet C# dans Visual Studio

Dans ce projet, vous installerez le framework Newtonsoft.Json de Newton-King qui apportera dans C# les instructions permettant la manipulation du format Json.

Puis vous coderez les 3 méthodes demandées.
- **LectureTokenJson**
- **AfficherPrettyJson (string nomFichier)**
- **EcritureJson**

Dans le dossier **"pour partie 2"** vous trouverez :
- le fichier***chien.json*** qui doit être recopié dans le dossier **debug** de votre projet `C#`
- un squelette de méthode EcritureFichierJson pour l'exercice 7 (écriture de fichier Json)

Exercice 4 : Installation du framework Json

JSON est un format de données natif de Javascript. Dans C#, il existe de nombeux framework pemettant son utilisation. Durant ce TD nous allons utiliser le framework JSON de Newton-King : ***newtonsoft.Json***

Son installation se fait à partir de VisualStudio en utilisant le gestionnaire de package de Microsoft NuGet. Pour les informations techniques : `https://www.newtonsoft.com/json`

1. Créer un projet Console dans Visual Sudio
2. Aller dans Menu => outil=>gestionnaire de package NuGet=>gérer les packages NuGet, parcourir la liste, choisir Newtonsoft.Json et l'installer
3. Vérifier son installation dans l'explorateur de solution
4. si ce n'est pas le cas afficher l'explorateur de solution : dans le menu=>Affichage=>explorateur de solution
5. dans l'explorateur de solution, déplier : nomDuProjet=> reférences. Il doit être dans la liste.

ϟAttention : pour l'utilisation, vous devez importer dans votre projet les librairies suivantes :

using System.IO ; pour les lecture/ecriture de fichier
using Newtonsoft.Json ; pour l'utilisation du format JSON

Exercice 5 : lecture d'un fichier Json

Afficher le contenu du document Json ***chien.json*** à l'aide du code founit ci-dessous.

Le fichier ***chien.json*** a été téléchargé sur Moodle dans le répertoire "pour partie 2" et recopié dans le dossier
monProjet\monProjet\bin\debug de votre projet C#).

Vous utiliserez un StreamReader pour le lecture en mode texte et un JsonTextReader pour l'interprétation du flux de texte obtenu par le StreamReader.

Ecrire votre code sous la forme d'une méthode de signature
void LectureTokenJson()

```
StreamReader reader = new StreamReader("chiens.json");
JsonTextReader jreader = new JsonTextReader(reader);
while (jreader.Read())
{
  //il y a deux sortes de token : avec une valeur associée ou non
  if (jreader.Value != null)
  {
    Console.WriteLine(jreader.TokenType.ToString() + "   " + jreader.Value);
  }
  else
  {
    Console.WriteLine(jreader.TokenType.ToString());
  }
}
jreader.Close();
reader.Close();
```

Exercice 6 : lecture d'un fichier Json

Le programme de cet exercice sera écrit sous la forme d'un méthode de signature
void AfficherPrettyJson (string nomFichier)
afin de pouvoir être réutilisé dans la suite des exercices

En utilisant les éléments de code du programme précédent, écrivez la méthode demandée pour afficher le contenu d'un document Json sous une forme propre et lisible. Puis utilisez-la (AfficherPrettyJson (chien.json)) pour afficher proprement ***chien.json***.

Par exemple :

```
Nouvel objet
--------------
chiens : Liste

Nouvel objet
--------------
nom : neptune
race : boxer
sexe : male
annee : 1990
proprietaire : Chapelot
email : chapelot@gmail.com
-------------

Nouvel objet
--------------
nom : figaro
race : husky
sexe : male
annee : 1995
proprietaire : Levasseur
email : levasseur@gmail.com
-------------
```

Pour lire les Jtoken : voir l'exercice précédent
Vous aurez à gérer les JToken : if (jreader.TokenType.ToString() == "StartObject")
- StartObject
- PropertyName
- EndObject
- StartArray
- EndArray

Et à afficher les jreader.Value

Exercice 7 : Ecriture d'un fichier Json

Ecrire votre programme sous la forme d'un méthode de signature
void EcritureFichierJSON ()
Vous trouverez cette méthode déjà partiellement écrite dans le squelette de programme fourni sur Moodle. Dans la partie déjà écrite, vous trouverez 4 tableaux contenant les informations nom, race, sexe et propriétaires pour 3 chats.

Vous avez à rédiger le code correspondant à l'écriture d'un fichier Json contenant les informations concernant ces 3 chats. Vous pourrez donc vous inspirez de l'affichage obtenu dans l'exercice 1 pour la structure du fichier chat.json.

Une fois le fichier chat.json construit vous utiliserez la méthode ***void AfficherPrettyJson (string nomFichier)*** écrite précédemment pour l'afficher.

```
1  //informations sur les chats
2  string[] nom = {"Bambou", "Taz", "Leloo" } ;
3  string[] race = {"europeen", "europeen", "siamois" } ;
4  string[] sexe = {"femelle", "male", "femelle" } ;
5  string[] proprietaire = {"Jules", "Alain", "Luc" } ;
6
7  //instanciation des "writer"
8  // -----------
9  // à compléter
10 // -----------
11
12 //debut du fichier Json
13 jwriter.WriteStartObject();
14 jwriter.WritePropertyName("chats");
15
16 //ecriture du tableau Json
17 // -----------
18 // à compléter
19 // -----------
20
21 //fin du fichier Json
22 jwriter.WriteEndObject();
23
24 //fermeture des "writer"
25 jwriter.Close();
26 writer.Close();
27
28 //relecture du fichier créé
29 Console.WriteLine("lecture des informations de chats.json");
30 AfficherPrettyJson("chats.json");
```

Consultez les méthodes de l'objet jwriter (de type JsonTexWriter)....
Vous aurez besoin des méthodes suivantes :
- WriteStartArray() ;

- WriteStartObject() ;
- WritePropertyName("nom") ;
- WriteValue(chat.Nom) ;
- WriteEndArray() ;
- WriteEndObject() ;