



Cégep de Saint-Hyacinthe
Département d'informatique

Programmation orientée objet

420-2DP-HY

(3-3-3)

Initiation à JSON

(1 heure)



Version 1.2

ENSEIGNANTS

Martin Lalancette, bureau B-2335, poste 2843, MIO

Comprendre les éléments suivants:

- Qu'est-ce que le JSON
- Règles de syntaxe
- Utilisation de **System.Text.Json**

Table des matières

Préparation de base	3
Qu'est-ce que le JSON?	3
Règles de syntaxe	3
Une donnée et type.....	4
Sérialisation vs Désérialisation d'un objet.....	5
Sérialisation et désérialisation d'un objet	6
Sérialisation et désérialisation d'une liste d'objets (même type)	6
Définir les options avec JsonSerializerOptions.....	7
Attributs à connaître	7
Bibliographie.....	8

Préparation de base

Pour bien suivre les instructions qui vont être mentionnées tout au long des séquences d'apprentissage de ce cours, une préparation de base s'impose. Il est important de créer un répertoire de travail (sur votre C: ou clé USB). Voici une suggestion d'arborescence:



Préparation: Créer la solution **Initiation à JSON** et le projet de type Console nommé **Exercice01**.

Qu'est-ce que le JSON?



Figure 1 - Auteur (Wikipédia - L'encyclopédie libre, 2020)

JSON, dont l'acronyme est *JavaScript Object Notation*, est un format texte permettant d'emmagasiner et d'échanger de l'information. Ce format a été créé par Douglas Crockford entre 2002 et 2005 et est une alternative au XML. C'est un langage indépendant. Il est plus compact que le XML et intégré au langage

JavaScript.

Règles de syntaxe

Voici un résumé des règles de syntaxe attribuées à JSON :

1. Une donnée est composée d'un nom de champ et d'une valeur
2. Un objet possède une ou plusieurs données entourées de { et }
3. Chaque donnée est séparée par une virgule
4. Un objet peut contenir un tableau d'objets en utilisant les crochets [et]
5. Un objet peut être composé d'un ou plusieurs sous-objets

Maintenant, voyons plus en détail chacun des éléments.

Une donnée et type

Les types implicites possibles sont :

Type	Exemple(s)
Chaîne de caractères (<i>string</i>)	"Tremblay" "c"
Nombre (<i>number</i>)	10 5.6
Booléen (<i>boolean</i>)	true false
Objet (<i>object</i>)	{...}
Tableau (<i>array</i>)	[...]
Nul	null

Nomenclature d'une donnée :

{"*Champ*" : "*Valeur*"}

↙ ↘
Objet

Exemples :

- "Nom" : "Tremblay"
- "Salaire" : 10.75
- "Fumeur" : false
- "Classe" : "A"

Voici des exemples de code représentant une Personne:

Code	Explications
Un objet et 1 champ. <pre>{ "Nom": "Tremblay" };</pre>	En mémoire vous obtiendrez un objet avec un champ <i>Nom</i> contenant " <i>Tremblay</i> ".
Un objet et plusieurs champs. <pre>{ "Nom": "Tremblay", "Prénom": "Marc", "Fumeur": false, "Age": 35 };</pre>	Plusieurs champs avec les valeurs correspondantes séparées par une virgule .
Un tableau (objet) et 5 valeurs. <pre>[10, 20, 34, 53, 90];</pre>	Un tableau contenant des valeurs numériques.

Un tableau (objet) et 2 objets.

```
[
  {
    "Nom": "Tremblay",
    "Prénom": "Marc",
    "Fumeur": false,
    "Age": 35
  },
  {
    "Nom": "Savard",
    "Prénom": "Lise",
    "Fumeur": true,
    "Age": 23
  }
];
```

Ici l'objet contient deux personnes grâce aux crochets.

À partir de la version .NET Core 3.1 (.NET 5.0), la gestion de la syntaxe JSON a été intégrée directement dans la bibliothèque [System.Text.Json](#). C'est cette composante que nous allons utiliser. Espaces de noms à utiliser:

```
using System.Text.Json;
using System.Text.Json.Serialization;
```

Sérialisation vs Désérialisation d'un objet

Avant de pouvoir utiliser correctement cette bibliothèque, il faut comprendre les deux actions suivantes :

Sérialiser

Permet de convertir (sérialiser) un **objet en mémoire vers la structure JSON** (format texte).

En mémoire :

Nom	Valeur	Type
client	[testerFormulairesWPF.Client]	testerFormulairesWPF.Client
Naissance	[2017-03-01 19:08:35]	System.DateTime
Nom	"Tremblay"	string
Prénom	"Marc"	string
Téléphone	"450-123-1234"	string



En texte (string) format JSON :

```
{
  "Nom": "Tremblay",
  "Prénom": "Marc",
  "Téléphone": "450-123-1234",
  "Naissance": "2017-03-01T19:08:35"
}
```

Désérialiser

Permet d'effectuer l'opération inverse, c.-à-d. convertir (désérialiser) du texte structuré en **JSON vers objet en mémoire**.

En texte (string) format JSON :

```
{
  "Nom": "Tremblay",
  "Prénom": "Marc",
  "Téléphone": "450-123-1234",
  "Naissance": "2017-03-01T19:08:35"
}
```

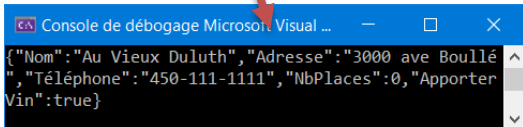
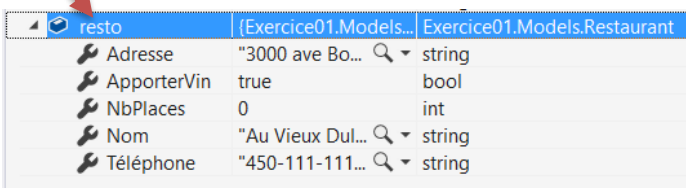


En mémoire :

Nom	Valeur	Type
client	[testerFormulairesWPF.Client]	testerFormulairesWPF.Client
Naissance	[2017-03-01 19:08:35]	System.DateTime
Nom	"Tremblay"	string
Prénom	"Marc"	string
Téléphone	"450-123-1234"	string

Sérialisation et désérialisation d'un objet

Cette section consiste à expliquer l'utilisation de **JsonSerializer** sur un seul objet par des exemples de codes :

Sérialiser	Désérialiser
<pre>// Déclaration des variables locales. string sJSON = ""; Restaurant resto = new Restaurant() { Nom = "Au Vieux Duluth", Adresse = "3000 ave Boullé", Téléphone = "450-111-1111", ApporterVin = true }; // Convertir l'objet en format JSON. sJSON = JsonSerializer.Serialize(resto); // Afficher le texte. Console.WriteLine(sJSON);</pre> 	<pre>// Déclaration des variables locales. string sJSON = "{ \"Nom\": \"Au Vieux Duluth\", \"Adresse\": \"3000 ave Boullé\", \"Téléphone\": \"450-111-1111\", \"NbPlaces\": 0, \"ApporterVin\": true }"; Restaurant resto = null; // Convertir le format JSON en objet. resto = JsonSerializer.Deserialize<Restaurant>(sJSON);</pre> 

Exercice 1. : Créer un projet de type Console et ajouter une classe nommée **Restaurant** avec les propriétés : **Nom**, **Adresse**, **Téléphone** et **ApporterVin**. Effectuer les exemples précédents. Suivre les instructions de l'enseignant.

Sérialisation et désérialisation d'une liste d'objets (même type)

Cette section consiste à expliquer l'utilisation de **JsonSerializer** sur une liste d'objets (même type) par des exemples de codes :

Sérialiser	Désérialiser
<pre>// Déclaration des variables locales. List<Automobile> liste = new List<Automobile>(); string sJson = ""; // Ajouter des automobiles à la liste. ... // Convertir object en format JSON. sJson = JsonSerializer.Serialize(liste);</pre>	<pre>// Déclaration des variables locales. List<Automobile> liste = null; ... // Convertir le format JSON en liste d'objets. liste = JsonSerializer.Deserialize<List<Automobile>>(sJson);</pre>

Définir les options avec JsonSerializerOptions

Vous pouvez définir des options à passer à l'objet en utilisant [JsonSerializerOptions](#). Voici un exemple qui permet de [supporter ou non la casse](#) dans les noms des propriétés:

```
var options = new JsonSerializerOptions
{
    PropertyNameCaseInsensitive = true
};
liste = JsonSerializer.Deserialize<List<Automobile>>(sJson, options);
```

Vous trouverez la liste des propriétés pouvant être définies avec cet objet ici: [Définition de la classe JsonSerializerOptions](#).

Attributs à connaître

Voici quelques attributs à utiliser dans certaines situations: [JsonIgnoreAttribute](#), [JsonIncludeAttribute](#), [JsonPropertyNameAttribute](#).

Exercice 2. : Modifier le programme précédent pour concevoir le menu suivant et effectuer les opérations demandées:

```
1) Afficher la liste des restaurants
2) Ajouter un restaurant
3) Retirer un restaurant
4) Sauvegarder la liste
5) Charger la liste
Q) Quitter
Votre choix : _
```

Pour les opérations Charger et Sauvegarder, utiliser un fichier JSON.

Bibliographie

Microsoft - Docs Comment sérialiser et désérialiser (marshaler et démarshaler) JSON dans .NET [En ligne] // Microsoft - Docs. - Microsoft, 02 04 2022. - 04 04 2022. - <https://docs.microsoft.com/fr-ca/dotnet/standard/serialization/system-text-json-how-to?pivots=dotnet-5-0>.

Présentation de JSON [En ligne] // www.json.org. - 19 08 2016. - <http://www.json.org/json-fr.html>.

Wikipédia - L'encyclopédie libre Douglas Crockford [En ligne] // Wikipédia. - 08 05 2020. - 03 04 2022. - https://fr.wikipedia.org/wiki/Douglas_Crockford.