

Reflection – An example JsonConvert

Mission

To understand the possibilities of getting metadata information at run-time, and to build your own JsonConvert.

Background

- Theory:
C#Note OOProg04 pp.23-28,
<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection>
- Wiki: [https://en.wikipedia.org/wiki/Reflection_\(computer_programming\)](https://en.wikipedia.org/wiki/Reflection_(computer_programming))
- MS References:
<https://docs.microsoft.com/en-us/dotnet/api/system.object.gettype?view=netcore-3.0>
<https://docs.microsoft.com/en-us/dotnet/api/system.type.gettype?view=netcore-3.0>
- Examples: <https://www.dotnetperls.com/reflection>
- JSON specification: <https://www.json.org/json-en.html>

Assignment 1- The first feeling

The first step is to try the principle of reflections.

To do this work you are to create a Library 'ReflectionLib', which have three classes:

- An **abstract class Person** (properties Name, BirthOfYear, constructor and a Get property Age, to return the age based on BirthOfYear and current Year)
- A **class Clerk inherit from Person** (properties Skills (a list of strings))
- A **class Manager inherit from Person** (Properties Employees (a list of Persons))

Next step creates a console application.

Define two objects one of the Clerk (add some skills) and one of the Manager (add some clerks).

Implement a Method like:

```
public void TryReflection(Object obj)
```

In this method do the following:

- Get the type of the object (obj.GetType())
- From the type found out these information's of the obj.
 - Name of object
 - Type of object (interface, abstract, class)
 - The properties of the object
 - The methods of the object

Extra: Do this recursively for the base-class, until the base-class is the Object-class

Assignment 2- JsonConvert - Serialize

You are to implement a new library 'MyJsonLib' with a static class 'MyJsonConverter', having two static generic methods:

1. `public static String Serialize<T>(T obj)`
2. `public static T Deserialize<T>(String json)`

Before you start implement this method, install the NuGet Newtonsoft in the Console App in previous assignment. Serialize the clerk object and printout the json-string, whereby you have something to compare with. *(NB the C# build-in json serialize do not always work!)*

Implementing the Serialize method

You start with Serialize in following steps:

1. make a StringBuilder for the json string // this is faster than solely working with strings
2. Get all the properties
3. The properties (`prop.PropertyType`) are either a **simple type** or an **object**
 - a. If it is an object call the methods itself (recursive)
 - b. If it is a simple type, append the property name and value to the StringBuilder.

Be careful with your '{' and '}' they are important for the json-string they surrounds objects. E.g.

```
{ "Name" : "Peter" , "BirthOfYear" : 1958 , "GetAge" : 64 }
```

Or

```
{ "Mother" : { "Name" : "Vibeke" , "BirthOfYear" : 1980 , "GetAge" : 40 } , "Father" :  
{ "Name" : "Peter" , "BirthOfYear" : 1978 , "GetAge" : 42 } , "Name" : "Anders" ,  
"BirthOfYear" : 2018 , "GetAge" : 2 }
```

Where '[' and ']' surrounds lists

Extra A- JsonConvert - Deserialize

When making the Deserializing it is the opposite way round. If you meet a '{' it is an object otherwise it is a simple type. Perhaps you can use regular expressions and get capture groups to find objects or lists

Create an object of the generic Type. For the name in the Json string find the property in the object, for each property find the value in the json-string, and set the value.