Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
2015

JSON (*JavaScript Object Notation*) é um formato de dados textual para representação de objectos. O JSON é independente da linguagem de programação e a sua notação usa convenções familiares ao idioma C, C#, Java ou Javascript. Baseia-se num subconjunto da linguagem de programação JavaScript, Standard ECMA-262.

A classe `Jsonfier` disponibiliza um método `ToJson` que retorna uma representação JSON para o objecto recebido por parâmetro, conforme o exemplo seguinte:

```
using Jsonzai.Reflect;

Student expected = new Student(27721, "Ze Manel");
string json = Jsonfier.ToJson(expected);
Console.WriteLine(json); // -> {"nr": 27721, "name": "Ze Manel"}

int [] expected = { 4, 5, 6, 7};
json = Jsonfier.ToJson(expected);
Console.WriteLine(json); // -> [4,5,6,7]
```

O método `ToJson` é recursivo na serialização de membros de objectos e de elementos de *arrays*, tendo como condição terminal o argumento ser `null`, `string` ou um tipo primitivo.

A implementação é flexível na adaptação do **critério de selecção** dos membros. Ou seja, existe a possibilidade de modificar qual a estratégia de selecção dos membros a usar na representação JSON. Por exemplo, na representação JSON de objectos podem ser considerados os valores dos campos do objecto, ou das suas propriedades, ou dos métodos sem parâmetros e retorno *non* `void`, ou outro critério qualquer. O objectivo é que a modificação do **critério de selecção** seja feita com o mínimo de alterações ao código.

A classe `Jsoninstr` tem uma API semelhante à de `Jsonfier` mas uma implementação distinta que melhora o seu desempenho. O objectivo é que na segunda serialização (e seguintes) de objectos do **mesmo tipo** se aproveite o processamento feito na primeira serialização.

Para tal, por cada tipo serializado pretende-se emitir código de um serializador com uma implementação específica para esse tipo. A emissão do código para cada serializador deve usar o suporte de `System.Reflection.Emit`. Os serializadores implementam um contracto comum para que possam ser guardados numa estrutura de dados que associa descritores de tipos a serializadores, como representado na figura seguinte: