

**Não deixe seu Front-End
preso ao Back-End use
Json.**

Caio Oliveira

Caio@javacia.com.br

Introdução a JSON.

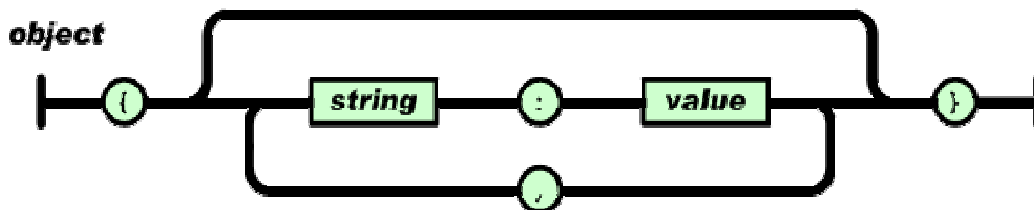
Json (Notação de Objetos JavaScript) é uma forma leve de troca de dados. Para pessoas e máquinas é fácil de interpretar e gerar. É baseado em um subconjunto da linguagem JavaScript com a especificação ECMA-262 (<http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf>). **Json** é gerado em formato texto, completamente independente de linguagem, pois usa convenções que são parecidas com as linguagens da família “C”, como por exemplo: **C++**, **C#**, **Java** entre outras. Isso faz com que o **Json** seja uma das melhores formas de troca de dados.

JSON está constituído em duas estruturas:

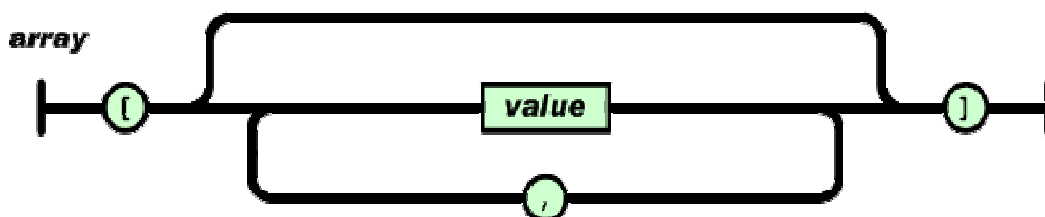
- Uma coleção de pares nome/valor. Em várias linguagens, isto é caracterizado como um *Object*, *Record*, *Struct*, *Dicionário*, *Hash Table*, *Keyed List*, ou *Arrays Associativas*.
- Uma lista ordenada de valores. Na maioria das linguagens, isto é caracterizado como uma *Array*, *Vetor*, *Lista* ou *Sequência*.

Em JSON, os dados são apresentados desta forma:

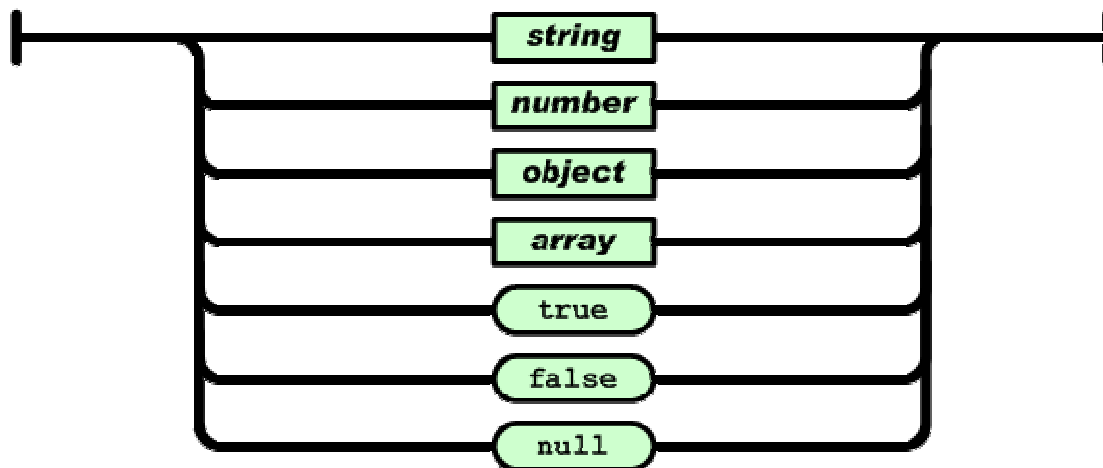
Um objeto é um conjunto desordenado de pares nome/valor. Um objeto começa com “{” (chave de abertura) e termina com “}” (chave de fechamento). Cada nome é seguido por “:” (dois pontos) e os pares nome/valor são seguidos por “,” (vírgula).



Uma array é uma coleção de valores ordenados. O array começa com “[” (conchete de abertura) e termina com “]” (conchete de fechamento). Os valores são separados por “,” (vírgula).



Um valor (value, na imagem acima) pode ser uma cadeia de caracteres (string), número, true, false, null, objeto ou uma array. Estas estruturas podem estar aninhadas.

value

Colocando a mão na massa.

Serialização de objetos ActionScript.

Agora que já entendemos o **Json**, vamos mostrar na prática como a serializar um objeto **ActionScript**.

O Google disponibilizou uma biblioteca "as3corelib" (<http://code.google.com/p/as3corelib/downloads/list>) que faz a serialização de objetos ActionScript. Baixe a biblioteca e adicione no seu projeto.

Adicione o código abaixo no arquivo principal da aplicação.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">

    <mx:Script>
        <![CDATA[
            import mx.collections.ArrayCollection;
            import mx.controls.Alert;
            import com.adobe.serialization.json.JSON;

            public function clickbBtnActionToJsonHandler():void{
                var obj:Object = new Object();
                obj._string = "javacia";
                obj._int = 1;
                obj._decimal = 1.10;

                obj._array = new
                Array("javacial", "javacia2", "javacia3");

                obj._object = new
                ArrayCollection(["javacial", "javacia2", "javacia3"]);
                txtRet.text = JSON.encode(obj);
            }

            public function clickBtnJsonToActionHandler():void{
                var obj:Object = JSON.decode(txtRet.text);
                if(obj!=null){
                    Alert.show(obj._string);
                }
            }
        ]]>
    </mx:Script>

    <mx:Button id="btnActionToJson" x="10" y="10" label="ActionToJson"
    click="{clickbBtnActionToJsonHandler()}" />

    <mx:TextArea width="526" height="241" id="txtRet" horizontalCenter="0"
    verticalCenter="0"/>

    <mx:Button id="btnJsonToAction" x="120" y="10" label="JsonToAction"
    click="{clickBtnJsonToActionHandler()}" />

</mx:Application>
```

Interpretando Código

Perceba que a classe “*com.adobe.serialization.json.JSON*” é quem faz todo o trabalho sujo de serialização. Quando clicamos no botão “*btnActionToJson*” é criado um objeto que é passado para o método “*encode*” da classe JSON, esse método retornará a string abaixo, essa string é o **Json**.

```
{
  "_string": "javacia",
  "_int": 1,
  "_decimal": 1.1,
  "_array": [
    "javacia1", "javacia2", "javacia3"
  ],
  "_object": {
    "source": [
      "javacia1", "javacia2", "javacia3"
    ],
    "filterFunction": null,
    "list": {
      "source": [
        "javacia1", "javacia2", "javacia3"
      ],
      "length": 3,
      "uid": "B14185F1-0F2B-38EF-385D-C4AFA6D64BB9"
    },
    "length": 3,
    "sort": null
  }
}
```

Já no botão “*btnJsonToAction*” é feito o processo inverso, a partir de um **Json** criamos um objeto *ActionScript*. Isso é feito com o método “*decode*” da classe JSON, após a criação do objeto podemos acessar todos os atributos do objeto que tínhamos criado no clique do botão “*btnActionToJson*”.

Assim como o *ActionScript* existem bibliotecas para grande parte das linguagens atuais. A grande sacada é mandar essa string **Json** via *Http* para “Serviços” de outra linguagem. Assim fica mais fácil você mudar de Back-End, pois seu Front-End só fará uma requisição e não terá que se preocupar com quem está fazendo o processo, se preocupando somente com o **Json** que será retornado.

No Link abaixo tem um exemplo de integração de Flex & Java usando **Json**.

Exemplo: http://www.javacia.com.br/Jc_JavaFlex_Json/

Source : http://www.javacia.com.br/blog/downloads_post/Jc_JavaFlex_Json.rar

No exemplo, será gerado um **Json** a partir de um objeto *ActionScript*, ele será mandado para um *Servlet* em Java que criará um objeto com o **Json**, será feita uma verificação com os atributos do objeto, após a verificação será retornado um **Json** para o *ActionScript*.