Unidade 9 - Tipos de dados JSON

Apresentação

O JSON (JavaScript Object Notation) é mundialmente reconhecido por sua leveza e simplicidade, por isso é muito utilizado no intercâmbio de dados entre *web services* e em aplicações com grande fluxo de dados.

Nesta Unidade de Aprendizagem, você irá identificar os diferentes tipos de dados em JSON e utilizá-los para implementar modelos JSON. Além disso, também irá implementar uma aplicação utilizando modelos JSON.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Identificar os diferentes tipos de dados em JSON.
- Implementar modelos JSON usando os tipos de dados.
- Implementar uma aplicação com modelos JSON.

Infográfico

JSON (JavaScript Object Notation – Notação de Objetos JavaScript) é uma formatação leve de troca de dados. Para seres humanos, é fácil de ler e escrever. Para sistemas de informação, é fácil de interpretar e gerar, superando assim o XML. Sua construção é baseada em um subconjunto da linguagem de programação JavaScript.

A cada dia tem crescido o número de soluções que utilizam JSON; isso ocorre, basicamente, porque é um formato leve de troca de informações e dados entre sistemas, sendo de fácil compreensão e assimilação por diversas plataformas e sistemas de informação.

No Infográfico a seguir, você verá os principais tipos de dados JSON e suas subdivisões de maneira esquematizada.

Conteúdo interativo disponível na plataforma de ensino!

Conteúdo do Livro

O JSON é um formato de intercâmbio para troca de dados entre sistemas derivado da linguagem JavaScript. É uma forma textual de representação de dados estruturados em uma coleção de pares no formato chave/valor.

No capítulo Tipos de dados JSON, da obra *Programação Back End III*, você irá estudar os tipos de dados JSON, vendo os principais modelos utilizados no formato JSON, além de implementar um exemplo de aplicação.

Boa leitura.

PROGRAMAÇÃO BACK END III

Pedro Henrique Chagas Freitas

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Tipos de dados JSON

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar os diferentes tipos de dados JSON.
- Implementar modelos JSON usando os diferentes tipos de dados.
- Implementar uma aplicação com o modelo JSON.

Introdução

Neste capítulo, você conhecerá a utilização dos diferentes tipos de dados JSON, além de aprender a implementar modelos a partir dos diferentes tipos de dados e, por fim, iremos demonstrar uma aplicação com o modelo JSON.

Contextualização e identificação dos tipos de dados

JSON é um padrão oriundo da linguagem de programação JavaScript. No JSON, assim como no JavaScript e em outras linguagens de programação, temos tipos de dados ou representações de dados por meio de estruturas, as quais abordaremos neste capítulo. Este é um formato simples, que visa referenciar a linguagem humana; logo, apresenta como principais tipos de dados os objetos, os *arrays* e os tipos de dados primitivos, como *string*, número, booleano e nulo.

Para o melhor entendimento deste assunto, devemos conhecer um pouco sobre a lógica de programação, tendo em vista que todos os tipos de dados JSON buscam atender a um conjunto pré-ordenado de comandos que resultam em um programa. Assim, podemos definir que JSON é uma forma textual de representação de dados estruturados em uma coleção de pares, no formato de chave/valor. A chave sempre é descrita como texto e o valor pode ser expresso como tipos de dados (texto, número, booleano, nulo, objeto ou uma sequência ordenada de valores, ou seja, *arrays*). O padrão JSON é muito utilizado no

intercâmbio de informações por ser independente de uma linguagem de programação, além de ser fácil de criar, manipular e analisar (TONSIG, 2008).

Apesar destes benefícios, esse formato possui limitações. O JSON não possui suporte a controles hipermídia, além de apresentar certa dificuldade para integrar dados de diferentes fontes, considerando que pode haver conflitos e ambiguidades nos pares chave/valor dos tipos de dados JSON (SOMMERVILLE, 2007).

Utilizamos estes tipos de dados para executar a lógica de programação dentro do padrão JSON. Em outras palavras, ocorre uma espécie de desencadeamento de pensamentos em sequência, visando atingir um objetivo específico, composto por instruções que formam um conjunto de normas pré-estabelecidas para realizar determinada tarefa, indicando ao computador uma ação elementar que deve ser executada. Assim, pode haver diversos tipos de artefatos de lógica de programação. Na Figura 1, observamos um exemplo de manipulação de dados com o JSON.

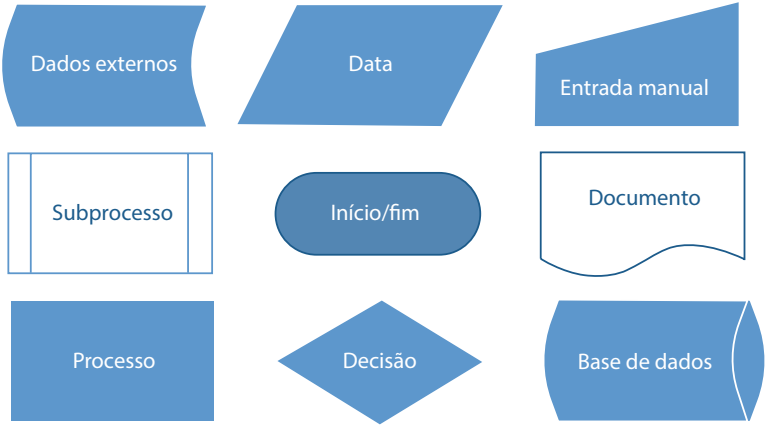

Figura 1. Exemplo de manipulação de dados com o formato JSON.

Diante disso, é possível verificar que no formato JSON existem dois tipos de dados: os dados elementares e os dados estruturados. Os dados elementares também são conhecidos como simples, básicos, nativos ou primitivos. Já os dados estruturados também são chamados de compostos.

Os tipos de dados elementares no formato JSON não podem ser decompostos. Por exemplo, se eu disser que João tem 20 anos, é possível decompor o valor de sua idade? Como sabemos que isto não é possível, este é um exemplo de dado elementar.

Existem diversos tipos de dados elementares. Os principais para o formato JSON, de acordo com Pressman (2011), estão listados a seguir.

- **Caractere**. Também conhecido como *char* ou literal, são representações de letras, dígitos e símbolos que podem ser utilizados no formato JSON. Quando colocados em conjunto, formam um tipo estruturado chamado *string* ou cadeia de caracteres. Exemplo: 'b', '$', '9', 'K' etc.
- **Inteiro**. Também conhecido como *integer*, são similares aos números inteiros da matemática, sem parte fracionária. Podem ser positivos, negativos ou nulos. Exemplo: -10% de crescimento do PIB; 895 km de distância; 32ºC de temperatura etc.
- **Lógico**. Também conhecido como *boolean*, são representações de valores lógicos, como verdadeiro/falso, ligado/desligado, sim/não etc. São extremamente importantes na programação, principalmente na verificação de condições.
- **Real**. Também conhecido como *float* ou *floating point* (ponto flutuante), são similares aos números reais da matemática e possuem parte fracionária. Exemplo: 3,7432.

Os tipos estruturados são aqueles que podem ser decompostos no formato JSON. Por exemplo, se eu disser que o meu nome é Carlos, é possível decompor esse nome? Sim, basta dividi-lo em caracteres ou em sílabas: 'C', 'a', 'r', 'l', 'o', 's'; Car-los. Há infinitos tipos estruturados, pois estes são a combinação de vários outros. O mais comum no JSON é a cadeia de valores, ou *string*, que são representações de sequências de caracteres, incluindo, ou não, símbolos. Estas sequências podem ser uma palavra, uma frase etc., como: "Assim como o homem se vê no seu coração, assim ele é".

No exemplo a seguir, veremos um trecho simples de código JSON, que visa demonstrar a utilização do tipo de dados *string* pela linguagem JavaScript.

```
{
"stringname" : "João Ribeiro",
"stringhomepage" : "http://joaoribeiro.com.br/" ,
"stringimage" : "http://joaoribeiro.com.br/joaoferias.png"
}
```

Modelos JSON e os tipos de dados

São vários os modelos de dados JSON que podem ser originados dos tipos de dados. Isto ocorre porque não existe um limite para a manipulação no formato JSON. Vejamos alguns dos principais modelos para o intercâmbio de dados, segundo os exemplos disponíveis em json.org (JSON EXAMPLE, [2007], documento *on-line*). Para facilitar a visualização e a compreensão da estrutura simplificada do JSON, vamos mostrar o mesmo modelo em XML.

Neste modelo temos uma representação de página de um glossário com troca de mensagens, utilizando JSON e o tipo de dados *string* pela linguagem JavaScript.

```
{
    "glossary": {
        "title": "exemplo de glossário",
       "GlossDiv": {
            "title": "Glossário JSON",
              "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                             "SortAs": "SGML",
                             "GlossTerm": "Standard Generalized
Markup Language",
                             "Acronym": "SGML",
                             "Abbrev": "ISO 8879:1986",
                             "GlossDef": {
                         "para": "JSON utiliza tipos de dados
String.",
                                    "GlossSeeAlso":  ["GML",
"XML"]
                    },
                             "GlossSee": "markup"
                }
            }
        }
    }
}
```

Agora, vemos a implementação do modelo de um glossário para troca de dados, em XML (JSON EXAMPLE, [2007], documento *on-line*):

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

```
{"widget": {
    "debug": "on",
    "window": {
                "title":  "Sample  Konfabulator  Widget",
                                "name":   "main_window",
                                        "width":    500,
        "height": 500
                                                        } ,
    "image": {
        "src": "Images/Sun.png",
                                    "name":    "sun1",
                                    "hOffset":    250,
                                    "vOffset":    250,
        "alignment": "center"
                                                        } ,
    "text": {
        "data": "Click Here",
        "size": 36,
                                   "style":    "bold",
                                   "name":    "text1",
                                    "hOffset":    250,
                                    "vOffset":    100,
        "alignment": "center",
       "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

Respectivamente, outro modelo difundido é a apresentação de modelos de *label* com tipos de dados booleanos, comuns em páginas Web de busca (JSON EXAMPLE, [2007], documento *on-line*):

```
{"menu": {
"booleano: true",
"booleano: false",
    "header": "SVG Viewer",
    "items": [
        {"id": "Open"},
        {"id": "OpenNew", "label": "Open New"},
        null,
        {"id": "ZoomIn", "label": "true"},
        {"id": "ZoomOut", "label": "false"},
        {"id": "OriginalView", "label": "Original View"},
        null,
        {"id": "Quality"},
        {"id": "Pause"},
        {"id": "Mute"},
        null,
        {"id": "Find", "label": "true"},
        {"id": "FindAgain", "label": "false"},
        {"id": "Copy"},
        {"id": "CopyAgain", "label": "true"},
        {"id": "CopySVG", "label": "false"},
        {"id": "ViewSVG", "label": "true"},
        {"id": "ViewSource", "label": "false"}

    ]
}}
```

Aplicação com o modelo JSON

Para facilitar nossa exemplificação, vamos trabalhar com uma aplicação simples, que utiliza o modelo JSON em conjunto com a linguagem JavaScript. Assim, teremos uma classe `BMW` que será o nosso POJO (*Plain Old JavaScript Object*, algo como "bom e velho objeto JavaScript"), e a classe `AplicacaoJSON` que será o nosso construtor *main*.

```
package br.com.json;
public class BMW {
    private Long id;
    private String modelo;
    private String placa;

    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getModelo() {
        return modelo;
    }
    public void setModelo(String modelo) {
        this.modelo = modelo;
    }
    public String getPlaca() {
        return placa;
    }
    public void setPlaca(String placa) {
        this.placa = placa;
    }
  impressão com o System.out.println()
    @Override
    public String toString() {
        return "[id=" + id + ", modelo=" + modelo + ", placa="
+ placa
                + "]";
    }
}
```

Na classe `BMW` temos os atributos `Id`, `Modelo` e `Placa`, representando inteiros e *strings*, que são nossos tipos de dados a serem usados no formato JSON. Agora, vamos implementar nossa classe `AplicacaoJSON`.

```
package br.com.json;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class AplicacaoJSON {
   public static void main(String[] args) throws JSONException {
        adicaoSimplesDeDados();
    }
}
```

Agora, vamos criar o método `adicaodeDadosJSON()`, que conterá a identificação das nossas `BMWs`.

```
private static void adicaoDeDadosJSON () throws JSONException {
    BMW bmw = new BMW();
    bmw.setId(1l);
    bmw.setModelo("320i");
    bmw.setPlaca("SFT1456");

    //Criação do objeto bmwJson
    JSONObject bmwJson = new JSONObject();
   //Inserção dos valores da bmw no objeto JSON
    bmwJson.put("id", bmw.getId());
    bmwJson.put("Modelo", bmw.getModelo());
    bmwJson.put("Placa", bmw.getPlaca());

    //Impressão do objeto JSON
    System.out.println(bmwJson);
}
```

Teremos, então, impresso o `ID`, o `modelo` e a `placa` da nossa `BMW`, por meio da migração da classe Java para JSON, mostrando:

```
{"id":1,"Modelo":"320i","Placa":"SFT1456"}
```

Podemos observar a importância da utilização dos tipos de dados JSON para a implementação do funcionamento lógico das aplicações. Por apresentar um formato simples, o JSON, que tem como objetivo referenciar a linguagem humana, apresenta em sua estrutura tipos de dados que, conforme vimos, são objetos, *arrays* e tipos de dados primitivos, como *string*, número, booleano, nulo. Portanto, a compreensão dos tipos de dados é essencial para o estudo do JSON.

Referências

JSON EXAMPLE. *Introducing JSON*, [*S. l.*], [2007]. Disponível em: https://json.org/example.html. Acesso em: 23 set. 2019.

PRESSMAN, R. S. *Engenharia de software*: uma abordagem profissional. 7. ed. Porto Alegre: AMGH; Bookman, 2011. 780 p.

SOMMERVILLE, I. *Engenharia de software*. 8. ed. São Paulo: Pearson Prentice Hall, 2008. 552 p.

TONSIG, S. L. *Engenharia de software*: análise e projeto de sistemas. 2. ed. Rio de Janeiro: Ciência Moderna, 2008. 319 p.

Leituras recomendadas

AQUILES, A. JSON e Objeto JavaScript são a mesma coisa? *Alura Cursos Online*, São Paulo, 4 mar. 2016. Disponível em: https://www.alura.com.br/artigos/json-e-objeto-javascript-sao-a-mesma-coisa. Acesso em: 23 set. 2019.

GONÇALVES, E. C. JSON Tutorial: Uma introdução ao JSON. *DevMedia*, Rio de Janeiro, 2012. Disponível em: https://www.devmedia.com.br/json-tutorial/25275. Acesso em: 23 set. 2019.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Dica do Professor

O formato JSON (JavaScript Object Notation) é um formato aberto usado como alternativa ao XML para a transferência de dados estruturados entre um servidor de *web* e uma aplicação *web*. Sua lógica de organização tem semelhanças com o XML, mas notação diferente.

Nesta Dica do Professor, você verá um exemplo de aplicação que armazena localidades em JSON com tipos de dados *string* e booleano.

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Exercícios

1) **JSON é um formato para troca de dados entre aplicações, que compete diretamente com a linguagem XML. Apesar dos inúmeros benefícios do formato JSON, ele apresenta limitações.**

Entre elas, temos que JSON não possui suporte a:

A) intercâmbio de dados.

B) indexação de dados.

C) controle de hipermídias.

D) *strings*.

E) objetos.

2) **O formato JSON foi pensado em meados de 2000, para ser uma forma simples de intercambiar dados.**

Dessa forma, os tipos de dados no formato JSON se dividem em dados:

A) elementares e estruturados.

B) procedurais e fundamentais.

C) primitivos e avançados.

D) simples e decompostos.

E) inteiros e booleanos.

3) **Os tipos de dados no formato JSON abarcam duas categorias.**

A categoria elementares tem tipos de dados que podem ser de vários formatos; entre eles, os booleanos, que também são conhecidos como:

A) reais.

B) *chars*.

C) inteiros.

D) *float*.

E) lógicos.

4) Os tipos de dados no formato JSON abarcam duas categorias.

A categoria elementares tem tipos de dados que podem ser de vários formatos; entre eles, os literais, que também são conhecidos como:

A) inteiro.

B) caractere.

C) real.

D) booleano.

E) *float*.

5) Há infinitos tipos estruturados no formato JSON, pois são a combinação de vários outros.

O mais comum no JSON é a cadeia de valores, que são representações de sequências de caracteres, também conhecidas como:

A) *char*.

B) *integer*.

C) *string*.

D) *float*.

E) *boolean*.

Na prática

O formato JSON ganhou popularidade em serviços da Web, como clientes de *e-mail* e *sites* de compras, pois consegue transmitir uma grande quantidade de informações entre o cliente e o servidor usando uma quantidade menor de caracteres.

Na prática, há diversos *sites* de compras na *internet* que utilizam JSON para realizar vendas *online*, o que colabora cada vez mais com a difusão e empoderamento do formato JSON frente à linguagem de intercâmbio de dados XML. Exemplo disso são os *sites* de venda de comida, que realizam o intercâmbio de dados de seus menus por meio de JSON. Veja mais a seguir.

Hoje, a maioria das pessoas utiliza ***smartphones***. Por consequência, muitas das transações de compra de pratos realizadas ocorrem em **aplicativos *mobile***, o que também colabora com a **utilização do formato JSON**, tendo em vista sua simplicidade de implementação.

Os desenvolvedores que trabalham em empresas como iFood desenvolvem menus para vários tipos de restaurantes utilizando o formato JSON.

Nesse caso, os tipos de dados em JSON são de fácil assimilação pelos programadores, tendo em vista a aproximação com a linguagem humana, o que ajuda também na reutilização das aplicações para outros pratos e outros menus de outros restaurantes, bem como na evolução de novas funcionalidades no intercâmbio de dados (por exemplo, tráfego de voz em videochamadas usando aplicativos mobile).

Temos, a seguir, um exemplo de menu para iFood, desenvolvido em formato JSON:

```
{
"fantasy-name": "Japa Food",
"phone": "061 345-6723",
"website": "www.ifood.com.br/japafood",
"category": ["Fast Food"],
"kitchen": ["Japonesa"]
"menu": [
{
"name": "Sushi",
"photo": "Restaurant photo",
"description": "Restaurant description",
"price": 10,
},
{
"name": "Sashimi",
"photo": "Restaurant photo",
"description": "Restaurant description",
"price": 20,
},
{
"name": "Mochi",
"photo": "Restaurant photo",
"description": "Restaurant description",
"price": 30,
}
]
}
```

Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.