



NOVO! Panorama #JOBS

favorito (10)

imprimir



anotar

marcar como lido

tirar dúvidas

Introdução: JSON

Veja neste artigo sobre JSON onde começaremos pelo básico sobre e avançaremos para exemplos mais complexos nos próximos artigos.

 (14)  (0)

Fala pessoal!

Esta semana percebi que muitos ainda não conhecem ou continuam na dúvida sobre o que é JSON e resolvi contribuir com mais um artigo.

Afinal, o que é JSON?

JSON é basicamente um formato leve de troca de informações/dados entre sistemas. Mas JSON significa **JavaScript Object Notation**, ou seja, só posso usar com JavaScript correto? Na verdade não e alguns ainda caem nesta armadilha.

O JSON além de ser um formato leve para troca de dados é também muito simples de ler. Mas quando dizemos que algo é simples, é interessante compará-lo com algo mais complexo para entendermos tal simplicidade não é? Neste caso podemos comparar o JSON com o formato XML.

Vamos visualizar esta diferença?

XML

```
<id>
  Alexandre Gamanome>
  R. Qualquerendereco>
```

JSON

```
{"id":1,"nome":"Alexandre Gama", "endereco":"R. Qualquer"}
```

Bom, é notável a diferença. Visualmente o segundo trecho (em JSON) é mais fácil de ler. Mas só existe essa diferença entre os dois? Na verdade não. Podemos listar algumas outras vantagens:

Vantagens do JSON:

- Leitura mais simples
- Analisador(parsing) mais fácil

JSON é o formato de troca de dados mais utilizado

CÓDIGOS ▾

ASSINE MVP

Baixe o APP

Login

- Arquivo com tamanho reduzido
- Quem utiliza? Google, Facebook, Yahoo!, Twitter...

Estas são algumas das vantagens apresentadas pelo JSON. Agora vamos ao que interessa: Código! Vamos fazer um exemplo extremamente simples nesta primeira parte e avançaremos no próximo artigo, inclusive falando sobre JSON em páginas Web.

Qual biblioteca usar?

Existem diversas bibliotecas para trabalharmos com JSON e Java. Usaremos no nosso estudo o json.jar que você pode baixar tranquilamente neste link

O nosso caso de estudo será simples: Teremos uma classe **Carro** que será a nossa classe POJO e a classe **EstudoJSON** que terá o nosso famoso método **main**.

Classe **Carro**

```
package br.com.json;

public class Carro {
    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 String getPlaca() {
        return placa;
    }
}
```

CÓDIGOS ▾ ASSINE MVP

Baixe o APP

Login

```
public String getPlaca() {
    return placa;
}
```

```
public void setPlaca(String placa) {
    this.placa = placa;
}

//Aqui fizemos o Override do método toString() para visualizar a impressão com o System.c
@Override
public String toString() {
    return "[id=" + id + ", modelo=" + modelo + ", placa=" + placa
        + "]";
}
}
```

Esta é uma classe simples, onde temos os atributos Id, Modelo e Placa.

Agora teremos a classe **EstudoJSON**

```
package br.com.json;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

CÓDIGOS ▼ ASSINE MVP

Baixe o APP

Login

```
adicaoSimplesDeDados();
}
}
```

Repare que criamos o método *adicaoSimplesDeDados()* que conterá o código de exemplo:

```
private static void adicaoSimplesDeDados() throws JSONException {  
    //Criação do objeto carro e atribuição dos valores  
    Carro carro = new Carro();  
    carro.setId(11);  
    carro.setModelo("Celta");  
    carro.setPlaca("AAA1234");  
  
    //Criação do objeto carroJson  
    JSONObject carroJson = new JSONObject();  
    //Inserção dos valores do carro no objeto JSON  
    carroJson.put("id", carro.getId());  
    carroJson.put("Modelo", carro.getModelo());  
    carroJson.put("Placa", carro.getPlaca());  
  
    //Impressão do objeto JSON  
    System.out.println(carroJson);  
}
```

Se executarmos este código, veremos que foi impresso no console o seguinte:

CÓDIGOS ▾ ASSINE MVP

Baixe o APP

Login

Você desenvolvedor mais atento vai reparar que existe um objeto impresso: Um *Long*! Isso mesmo! Como vimos, o JSON consegue armazenar objetos! Podemos inclusive armazenar um objeto do tipo *Carro* mesmo:

Modificamos o nosso método main:

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

E adicionamos o método *adicaoDeUmObjeto()*:

```
private static void adicaoDeUmObjeto() throws JSONException {  
    Carro carro = new Carro();  
    carro.setId(11);  
    carro.setModelo("Celta");  
    carro.setPlaca("AAA1234");  
  
    JSONObject carroJson = new JSONObject();  
    //Adição do objeto carro  
    carroJson.put("Carro", carro);  
  
    System.out.println(carroJson);  
}
```

Neste caso foi impresso no console o seguinte:

CÓDIGOS ▾ ASSINE MVP

Baixe o APP

Login

Simples não?

Como o JSON trabalha com coleção de pares nome/valor, podemos imprimir um valor específico simplesmente chamando o nome que desejamos.

```
System.out.println(carroJson.get("Modelo"));
```

Veja que neste caso queremos somente o modelo do carro, bastando fazer a chamada `get("nome que desejamos")`!

Conclusão

É isso pessoal! Vimos as vantagens do JSON e vimos como é simples começar a trabalhar com ele. Nos próximos artigos veremos exemplos mais complexos e veremos algo mais real onde faremos chamadas à API do Facebook!


Você também pode encontrar este código no meu Gist Público.

Abraços!

Alexandre Gama

twitter.com/alexandregamma

<http://alexandregama.wordpress.com/>

 Publicado no [Canal Java](#)



por Alexandre Gama

Expert em Java e programação Web

CÓDIGOS ▾

ASSINE MVP

Baixe o APP

Login

Ajude-nos a evoluir: você gostou do post?



(14)



(0)

Compartilhe:

Ficou com alguma dúvida?



Post aqui sua dúvida ou comentário que nossa equipe responderá o mais rápido possível.



Sinval Júnior

Como fazer o get no caso de ser um objeto?

Ex: `System.out.println(carroJson.get("Carro"));`

há +1 ano



Douglas Claudio

Olá Sinval, obrigado pelo seu comentário.

Enviamos sua solicitação ao Alexandre e estamos no aguardo de um feedback do mesmo.

Um abraço.

CÓDIGOS ▼

ASSINE MVP

Baixe o APP

Login



Diogo Souza

Olá Sinval,

Neste caso não é permitido porque você já tem o objeto em mãos. Se você tivesse um objeto dentro do outro ou uma lista de objetos em formato JSON aí poderia usar normalmente o código que postou..

há +1 ano

Adicionar um comentário...

Mais posts

Video aula

Refatorando injeção de dependências - Curso Java Avançado - Aula 35

Video aula

Aplicando a extração de métodos na refatoração - Curso Java Avançado - Aula 34

Video aula

Top 10 Refactorings Java - Curso Java Avançado - Aula 33

Video aula

Refatorando em Java - Curso Java Avançado - Aula 32

CÓDIGOS ▾

ASSINE MVP

Baixe o APP

Login

Trabalhando com Enums - Curso de Java Avançado - Aula 31

Listar mais conteúdo



Publique | Assine | Fale conosco

Hospedagem web por **Porta 80 Web Hosting**

CÓDIGOS ▼

ASSINE MVP

Baixe o APP

Login