Residência em Software

Residência em Tecnologia da Informação e Comunicação

Arquivos JSON

Professor:

Alvaro Degas Coelho

Java

INSTITUIÇÃO EXECUTORA
cepedi UESC

COORDENADORA
MCTI FUTURO Softex

APOIO
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

Residência em Software

JSON

- Um problema: (potencialmente) queremos que QUALQUER sistema integre com QUALQUER sistema
  - Java com C? C com Python? Python com Ruby, Ruby com Cobol? Cobol com Java?
- Tipos incompatíveis
  - Tecnologias, estruturas de dados, representação de tipos
    - Não existe date em Cobol. Não existe herança múltipla em Java...
- Ideia geral: um formato de dados “padrão”
  - Que permita a estruturação dos dados tal como os vemos
- MODELO de CLASSES (ou de Entidades e Relacionamentos)

Residência em Software

Bancos de Dados Hierárquicos

- Proposta antiga
  - Anterior aos modelos em rede e, principalmente, o modelo relacional que é o padrão atual
- Ideia geral: criar entidades subordinar a elas os atributos
  - Além de, eventualmente, outras entidades
- Sistemas
  - ADABAS (questionável)
  - IMS Database (usado para dar suporte às missões Apolo)

Residência em Software

Como?

- Exemplo
  - Venda1
    - Data: 23/12/2032
    - Itens
      - Item1
        - Desc: Ervilha em lata
        - Preço: 4,5
        - Quantidade: 5
      - Item 2
        - Desc: Manteiga: 500g
        - Preço: 10,5
        - Quantidade: 2
- ...

- Exemplo
  - Venda2
    - Data: 24/12/2032
    - Itens
      - Item1
        - Desc: Manteiga 500g
        - Preço: 10,5
        - Quantidade: 4
      - Item 2
        - Desc: Cerveja (pack c/12)
        - Preço: 36,0
        - Quantidade: 2
- ...

Residência em Software

BD Hierárquicos estão Ultrapassados

- Pouca flexibilidade
  - Impossível normalizar
  - Operações DML: um pesadelo!
    - Tenta mudar o preço da Margarina!
- Superados pelos BD Relacionais
  - Relações (Tabelas / Views)
  - Chaves Primárias e estrangeiras
  - SQL

Residência em Software

BD Hierárquico: ruim mesmo?

- Para operações de manipulação de dados: péssimo!
- Para consultas?
- Para transporte de dados?
- Modelos XML e JSON
  - Mesmo conceito dos BDs hierárquicos
  - Sintaxe mais flexível

Residência em Software

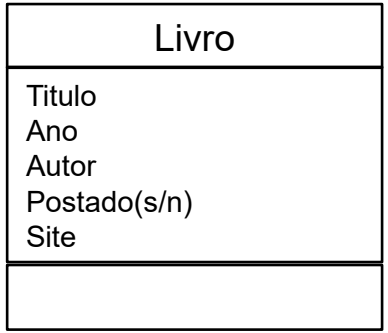
Um arquivo JSON

- Sintaxe simples
- O JSON é sempre uma lista de Objetos
  - Cada objeto com uma lista de atributos
  - Cada atributo com seu valor
- Formato
  - **{ }** (chaves) para delimitar um objeto;
  - **:** (dois pontos) para separar os atributos de valores;
  - **,** (vírgula) para separar os atributos chave/valor;
  - **[ ]** (colchetes) para delimitar um array.
    - Sim... JSON não respeita nem a 1ª Forma Normal!

Residência em Software

Exemplos

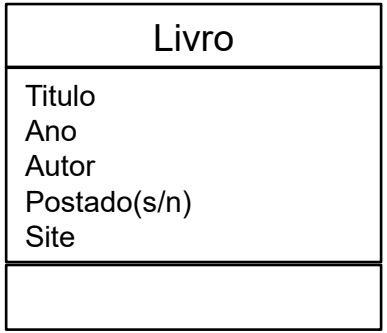
- Suponha a classe ao lado
- Atributos
  - Titulo
  - Ano
  - Autor
  - Postado(s/n)
  - Site



Residência em Software

Exemplos

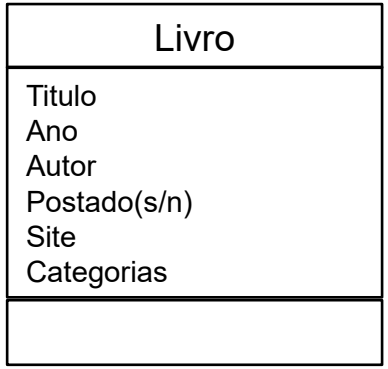
- Suponha a classe ao lado
- Atributos
  - Titulo
  - Ano
  - Autor
  - Postado(s/n)
  - Site



Um objeto deste tipo num arquivo JSON
{ “titulo”: “O que é Lasanha”, “ano”: 2009, “autor”: “Degas”, “postado”: true, “site”: www.degas.com.br}

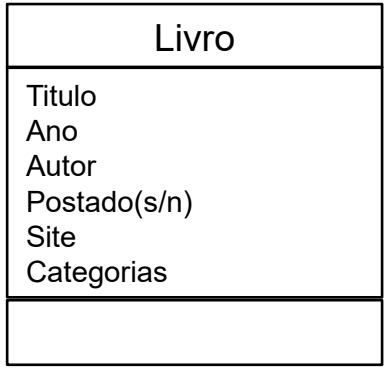
Sofisticando o Exemplo

- Suponha um atributo multivalorado (alô, 1FN?) categorias
- Atributos
  - Titulo
  - Ano
  - Autor
  - Postado(s/n)
  - Site
  - Categorias



Sofisticando o Exemplo

- Suponha um atributo multivalorado (alô, 1FN?) categorias
- Atributos
  - Titulo
  - Ano
  - Autor
  - Postado(s/n)
  - Site
  - Categorias



Um objeto deste tipo num arquivo JSON

{ “titulo”: “O que é Lasanha”, “ano”: 2009, “autor”: “Degas”, “postado”: true, “site”: www.degas.com.br , “categoria”: [“culinária”, “fitness”, “vida saudável”]}

Residência em Software

Melhor visualizar de forma hierárquica

- Mas trata-se apenas da visualização! O que importa é a sintaxe!

```
{
    “titulo”: “O que é Lasanha”,
    “ano”: 2009,
    “autor”: “Degas”,
    “postado”: true,
    “site”: www.degas.com.br ,
    “categoria”: [
                “culinária”,
                 “fitness”,
                 “vida saudável”
                 ]
}
```

Residência em Software

Tipos de dados JSON

- Típicos
  - **string:** Separados por aspas duplas ou simples. Ex: “Degas ou ‘Degas’;
  - **numérico:** Sem o uso das aspas, podendo ser inteiro ou real. Ex: inteiro (2009) ou real (2.009);
  - **booleano:** tipo lógico normal, podendo ser true ou false;
  - **nulo:** valor nulo. Ex: “resumo”: null.
- Objetos complexos devem ser construídos a partir dos tipos primitivos
- Implementações diversas podem conter mais tipos
  - Não é recomendável por conta da integração!

Residência em Software

JSON em Java

- JSON é um arquivo texto
- Você pode abrir e recuperar os dados sem grande dificuldade
  - Você pode criar uma biblioteca sua para isso
- Ou você pode usar bibliotecas de Terceiros

Residência em Software

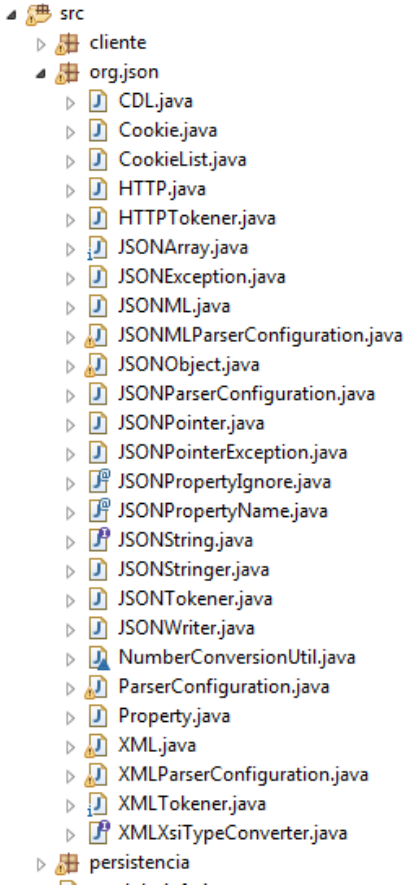
org.json

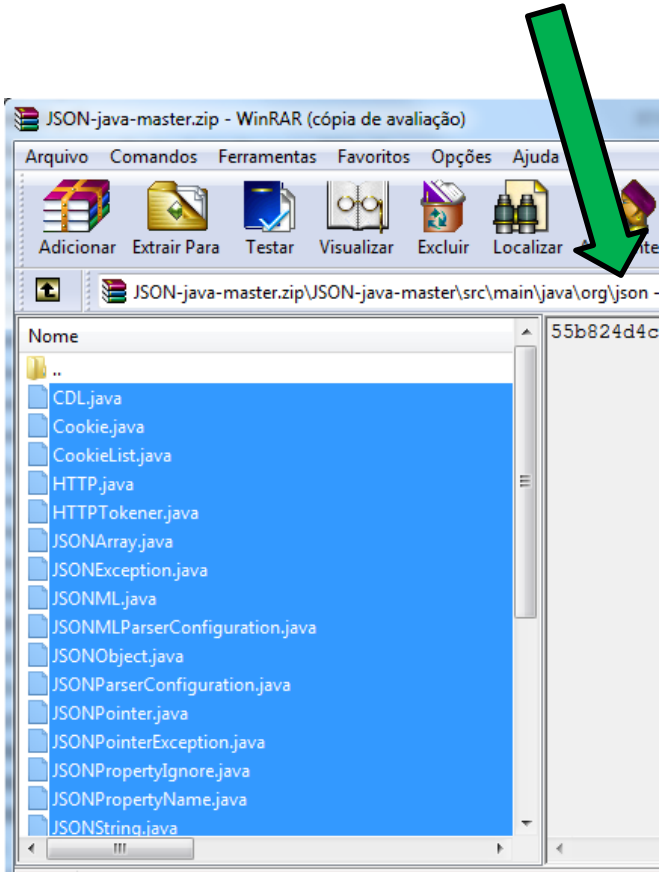
- Uma proposta de padrão
  - Google
- Não é a única
- Não é a mais usada em contextos WEB
- Prática e permite manipular objetos JSON no mundo Java

Residência em Software

Como?

- São classes dentro de uma package
- Crie uma package org.json
- Baixe o .zip com a estrutura
    - Está disponível no nosso drive
- Descompacte as classes dentro do Package





Residência em Software

Objetos do tipo JSON

- Dentro da biblioteca org.json
  - //instancia um novo JSONObject
  - JSONObject my_obj = **new** JSONObject();
- my_obj é a representação de um json
- Podemos acrescentar atributos e valores
  - my_obj.put("titulo", "JSON x XML: a Batalha Final");
  - my_obj.put("ano", 2012);
  - my_obj.put("genero", "Ação");

Residência em Software

Objetos do tipo JSON

- Podemos gerar uma entrada JSON com o objeto json (útil para salvar em qrquivos)
  - my_obj.toString();
  - {"ano":2012,"genero":"Ação","titulo":"JSON x XML: a Batalha Final"}
- O método put() também serve para alterações
  - my_obj.put("titulo", "JSON x XML: o Confronto das Linguagens");
- Podemos recuperar atributos específicos (observando o tipo!)
  - String titulo = my_obj.getString("titulo");
  - Integer ano = my_obj.getInt("ano");
  - String genero = my_obj.getString("genero");

Residência em Software

Parsing

- O objeto JSONObject é uma representação do dado JSON
  - Que tipicamente é lido de um arquivo de texto
  - String
- É possível fazer um parse
- Criar um objeto a partir de uma String que esteja gramaticalmente correta
- Observe o uso do caracter de escape (\)
  - String json_str = "{\"titulo\":\"Os Arquivos JSON\",\"ano\":1998,\"genero\":\"Ficção\"}";
  - //instancia um novo JSONObject passando a string como entrada
  - JSONObject my_obj = **new** JSONObject(json_str);

Residência em Software

Construindo Objetos JAVA a partir de JSON

- Supondo que tenhamos um objeto Filme (Titulo, Ano, Genero)
  - Filme filme = new Filme();
  - filme.setTitulo(my_obj.getString("titulo"));
  - filme.setAno(my_obj.getInt("ano"));
  - filme.setGenero(my_obj.getString("genero"));
- Temos um objeto Java que foi obtido a partir de um objeto JSON

Residência em Software

JSON Array

- Para dados multivalorados
  - JSONArray my_genres = **new** JSONArray();
  - my_genres.put("aventura");
  - my_genres.put("ação");
  - my_genres.put("ficção");
- O array precisa ser inserido (como um campo) dentro do JSON
  - my_obj.put("generos", my_genres);
- O JSON (string)
  - {"ano":2012,"generos":["aventura","ação","ficção"],"titulo":"JSON x XML: a Batalha Final"}
- O array pode ser recuperado (como um campo) a partir do JSON
  - JSONArray meuArray = my_obj.getJSONArray("generos");

Residência em Software

Recuperando dados de um ARRAY

- Podemos recuperar os elementos do Array
  - `for (int i = 0; i < elenco.length(); i++) {`
  - `    System.out.println("(" + i + ") " + elenco.get(i));`
  - `}`
- Podemos usar isto para reconstruir um atributo multivalorado
- Supondo um JSONArray com as categorias de um livro
  - `Livro livro = new Livro();`
  - `...`
  - `for (int i = 0; i < elenco.length(); i++) {`
  - `    livro.getCtegorias().put(categorias.get(i));`
  - `}`

Residência em Software

Um pouco mais sobre JSOn Array

- É possível se manipular elementos do array
- Inserir
  - elenco.put("Michael Java Fox");
- Alterar
  - elenco.put(0, "Jennifer Json Leigh");
- Excluir
  - elenco.remove(2);

Residência em Software

Exercício

- Crie uma classe Cliente com atributos nome, cpf e cnh
  - Getters e Setters!!!
- Crie uma package persistencia
  - Dentro dela crie uma classe ClienteJSON
    - Métodos salvaClientes(Lista de clientes, nome do arquivo)
    - recuperaClientes(nome do arquivo)
  - Programe esta classe (atenção nos Exceptions)
- Crie um método main em Cliente para testar a funcionalidade

Residência em Software

Exercício 2

- Acrescente um atributo dtNascimento ao seu cliente
  - Get e Set
- Altere os métodos salvaClientes e recuperaClientes
  - Lembre que a data terá que ser armazenada no JSON utilizando tipos primitivos!
  - Método para converter e restaurar datas!
- Altere o método main em Cliente para testar a funcionalidade

Residência em Software

Exercício 3

- Acrescente um atributo telefones ao seu cliente
  - Uma Lista (aArrayList?) de String
  - Get e addTelefone(fone)
- Altere os métodos salvaClientes e recuperaClientes
  - Lembre que a data terá que ser armazenada no JSON utilizando tipos primitivos!
  - Vai ter que criar um JSONArray para colocar os telefones
    - Na escrita do arquivo: gerar o JSONArray a partir do ArrayList
    - Na leitura: gerar o ArrayList a partir do JSONArray
- Altere o método main em Cliente para testar a funcionalidade