


Java Avançado

Os Formatos XML e JSON




Softblue
cursos online

Tópicos Abordados



- Formato XML
 - Introdução ao XML
 - XML x HTML
 - Características
 - Validação com DTD e XML Schema
 - Manipulação de documentos com as APIs SAX e DOM
- Formato JSON
 - Funcionamento
 - Manipulação de dados com a API Gson

Introdução



- XML
 - **E**xtensible **M**arkup **L**anguage
- Linguagem de marcação para descrever dados estruturados

Exemplo de documento XML



```
<Clientes>
  <Cliente id="1">
    <Nome>João da Silva</Nome>
    <Email>joao@email.com</Email>
  </Cliente>

  <Cliente id="2">
    <Nome>Maria Silva</Nome>
    <Email>maria@email.com</Email>
  </Cliente>
</Clientes>
```

XML x HTML



- Assim como no HTML, XML é baseado em tags
- Em XML, as tags podem ser definidas pelas aplicações
- XML é muito mais rígido quanto ao formato do documento
 - Todos os elementos devem estar definidos de acordo com a especificação

Características



- XML possibilitou fácil integração de sistemas multiplataforma
 - É um padrão aberto e independente de fornecedor e dispositivo
- O sucesso do XML também se deve à grande quantidade de código pronto
 - Facilita a manipulação de documentos

Validação



- A validação do documento pode ser feita no momento da sua leitura
- É possível validar documentos XML usando
 - DTD
 - XML Schema

Validação: DTD



- A DTD descreve como um documento XML deve ser
- Utiliza uma linguagem própria
- Caiu em desuso, pois foi substituída pelo XML Schema

Validação: XML Schema



- O XML Schema descreve como um documento XML deve ser
- Utiliza a própria linguagem XML para especificar documentos XML
- É o modo de validação mais utilizado atualmente
- É mais poderoso que o DTD

Manipulação de Documentos XML



- A manipulação de documentos pode ser feita pelas APIs
 - DOM
 - SAX
- A partir do Java 1.4, implementações de ambas as APIs estão disponíveis no JDK

DOM



- **D**ocument **O**bject **M**odel
- API fácil e intuitiva
- Cria uma árvore de objetos em memória
- Esta árvore representa a estrutura do documento XML
- Desvantagem
 - Ocupa muita memória

SAX



- Não é tão intuitiva quanto a DOM
- SAX é baseada em eventos
 - Funções de callback são invocadas durante o parsing do documento
- Como não cria uma estrutura em memória, é aconselhada para documentos XML extensos

O formato JSON



- **J**ava**S**cript **O**bject **N**otation
- Formato de representação de dados
 - Baseado em texto
 - Independente de linguagem
 - Mais simples que o formato XML
- Baseado em duas estruturas
 - Pares de chave e valor
 - Sequência ordenada de elementos

Exemplos de dados em JSON



Cada dado é separado do outro por vírgula (,)

```
{ "id":1, "nome":"João da Silva", "email":"joao@email.com" }
```

Um conjunto de dados é delimitado por chaves ({ e })

A chave e o valor de um dado é separada por dois pontos (:)

[10, true, "A"]

Sequências de dados são delimitadas por colchetes ([e])

A API Gson



- Uma das APIs disponíveis para manipular dados no formato JSON em Java
 - <http://code.google.com/p/google-gson>
- O ponto forte do Gson é a conversão de objetos Java em dados no formato JSON e vice-versa
- Utilização da API
 - Baixar o código-fonte na página oficial
 - Adicionar o JAR da API ao classpath da aplicação

Classes importantes da API



- *Gson*
 - Principal classe da API
 - Possui métodos para manipular os dados

```
Gson gson = new Gson();

Cliente c = new Cliente(1, "João da Silva",
    "joao@email.com");

String jsonStr = gson.toJson(c);

Cliente c = gson.fromJson(jsonStr, Cliente.class);
```

Converte para JSON

Converte para objeto

Classes importantes da API



- *JsonElement*
 - Representa um elemento JSON
 - Subclasses
 - *JsonObject*, *JsonArray*, *JsonPrimitive* e *JsonNull*

```
JsonObject obj = new JsonObject();
obj.add("id", new JsonPrimitive(1));
obj.add("nome", new JsonPrimitive("João da Silva"));
obj.add("email", new JsonPrimitive("joao@email.com"));

String jsonStr = gson.toJson(obj);
```

Adiciona os elementos

Converte para JSON



6