VOLTAR

Metadados para Troca de Dados (XML e JSON)

Este tópico apresenta os principais conceitos relacionados ao uso de metadados em arquivos de formato XML e JSON.

NESTE TÓPICO

Os vários modelos de arquitetura existentes no mercado, apesar de não dizerem explicitamente que pode ocorrer a comunicação entre sistemas desenvolvidos com diferentes tecnologias de desenvolvimento, mas isso é algo bastante comum no desenvolvimento de aplicações de grande porte, como os sistemas bancários, ou os sistemas para gestão empresarial. Para permitir a comunicação entre os sistemas desenvolvidos com tecnologias diferentes, uma das formas mais usuais é o uso de metadados.

Um metadado (ou metainformação) são informações a respeito de outros dados, e eles existem os anos 1970. Inicialmente, para o armazenamento de dados usados pelos aplicativos, e posteriormente, para comunicação entre aplicativos, e/ou compartilhamento de informação com outras aplicações. Normalmente, os metadados são escritos em arquivos simples de texto, pois é o formato mais simples de ser processado e transmitido pelas redes de comunicação. Ao longo dos anos, foram criados vários tipos de metadados, para os mais variados tipos de aplicações, algumas vezes sendo modelados de acordo com determinados padrões públicos, outras vezes não.

Nos anos 1990, as aplicações para a Internet trouxeram novos tipos de metadados, baseados em padrões abertos, como os formatos XML e JSON. A principal vantagem no uso desses tipos de arquivos se deve ao fato de que são arquivos que podem ser escritos e lidos pro rotinas escritas usando qualquer linguagem de programação. Além disso, ambos os padrões definem apenas como os metadados devem ser formatados e referenciados. Os nomes

dos metadados podem ser definidos pelos próprios desenvolvedores, e os arquivos em formato JSON ou XML podem ser usados como protocolos ou interfaces comunicação.

O uso desses formatos é tão popular e difundido, que várias plataformas de desenvolvimento, como o Java EE ou o .Net possuem mecanismos que já escrevem ou extraem as informações desses tipos de arquivos, usando como referência classes que representam a estrutura dos metadados.

XML (e*X*tensible *M*arkup *L*anguage)

O padrão XML é uma recomendação do consórcio W3C (World Wide Web Consortium) que define uma linguagem de marcação projetada para ser incrementada de acordo com os seguintes princípios:

- Separar o conteúdo a ser apresentado, da formatação do conteúdo (arquivos conhecidos como DTD);
- Ser uma estrutura que possa ser facilmente entendida por máquinas e seres humanos;
- Não haver limitação na quantidade de tags que podem ser criadas num arquivo XML;
- Pode servir de agente de transporte de informações entre sistemas e bases de dados distintos;
- Manter o foco do desenvolvedor no conteúdo a ser transmitido, e não em sua aparência.

Esses princípios existem pois há uma variedade muito grande de aplicações que existem no mercado, e cada uma deles com suas próprias características e demandas, que precisam ser consideradas pelos desenvolvedores. Então, o desenvolvedor precisa ter a liberdade necessária para criar um arquivo de acordo com suas necessidades, e a sua estrutura pode ser definida de acordo com o que o desenvolvedor precisa implementar. Por isso, ele se tornou bastante popular, não só na área de tecnologia da informação, mas também nas áreas de telecomunicações, aplicações de saúde, aplicações governamentais, entre outras. Tanto que outros padrões de tecnologia foram definidos com base nesse formato, como o XHTML, os protocolos de envio de mensagens MMS, e os serviços web (conhecidos como Web Services).

O padrão XML permite ao desenvolvedor definir um modelo de metadados baseado em tags, e cada nome do metadado recebe um nome significativo que pode ser facilmente identificado. Esse padrão define uma tag que indica o início do arquivo xml, a versão xml sendo usado, e qual codificação de caractere está sendo usada. Depois, as demais tags indicam as informações que são necessárias ao aplicativo, como o exemplo da listagem a seguir, que apresenta um arquivo XML usado para armazenar os dados pessoais de uma pessoa.

```
Listagem 1 - Exemplo de Arquivo XML usado para configurar um projeto na IDE NetBeans

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://www.netbeans.org/ns/project/1">
    <type>org.netbeans.modules.web.project</type>
    <configuration>
        <data xmlns="http://www.netbeans.org/ns/web-project/3">
            <name>ExemploJPAWeb</name>
            <minimum-ant-version>1.6.5</minimum-ant-version>
            <web-module-libraries>
                <library dirs="200">
                    <file>${libs.hibernate4-persistence.classpath}</file>
                    <path-in-war>WEB-INF/lib</path-in-war>
                </library>
                <library dirs="200">
                    <file>${file.reference.derbyclient.jar}</file>
                    <path-in-war>WEB-INF/lib</path-in-war>
                </library>
                <library dirs="200">
                    <file>${file.reference.jandex-2.0.2.Final.jar}</file>
                    <path-in-war>WEB-INF/lib</path-in-war>
                </library>
            </web-module-libraries>
            <web-module-additional-libraries/>
            <source-roots>
                <root id="src.dir"/>
            </source-roots>
            <test-roots>
                <root id="test.src.dir"/>
            </test-roots>
        </data>
    </configuration>
</project>
```

Como pode ser observado na listagem, após a tag *xml*, o desenvolvedor define outras tags, que são escritas de modo hierárquico, e sempre há um par de tags indicando o início e o fim da informação. Por exemplo, para indicar que a tag se refere a um nome, ele poderia escrever algo como Bilbo Bolseiro. A mesma regra vale para coleções de dados. Por exemplo, é possível que o aplicativo use uma lista de usuários, e para ter acesso aos vários nomes de usuário existentes nessa lista, o desenvolvedor poderia tags chamadas e para armazenar valores como Bilbo Bolseiro; Frodo Bolseiro; e Legolas Folha Verde.

A desvantagem do padrão XML é a quantidade de bytes necessária para escrever os nomes das tags, se comparado ao número de bytes usados para escrever a informação. Assim, a tag nome que contém o valor Bilbo Bolseiro possui treze bytes para as tags, e catorze bytes para o conteúdo. Isso significa que dos vinte e sete bytes existentes no texto a ser processado, somente catorze bytes são a informação que realmente importa. Até certo ponto, os bytes usados para compor os nomes da tags podem ser considerados como sendo uma informação inútil, uma vez que ela é usada apenas para identificar o tipo da informação a ser processada, e será rapidamente descartada pelo aplicativo. Esses bytes adicionais podem ser considerados inúteis, principalmente em situações em que o aplicativo opera sobre uma rede de transmissão de dados em que a tarifação é feita sobre o volume transmitido de bytes. No exemplo acima, praticamente cinquenta por cento do valor tarifado seria baseado em bytes usados apenas

para indicar o uso da informação, e não a informação que o sistema realmente precisa processar. No caso, esse custo pode ser considerado como um empecilho ao desenvolvimento do software.

Outra desvantagem é a necessidade de processar as tags que indicam o início e o fim da informação, tornando a extração dos dados mais lenta, e deixando o desenvolvimento das rotinas que geram os arquivos XML, e que extraem as informações deles, seja um pouco mais demorado.

Listagem 2 - Exemplo de arquivo XML usado para armazenar dados pessoais relacionados aos empregados de uma empresa

```
<?xml version="1.0" encoding="UTF-8"?>
<Empregados>
<item>
<DadosPessoais>
   <anoMesCompetencia>201302</anoMesCompetencia>
   <siglaEmpresa>SIGLA</siglaEmpresa>
   <cnpjEmpresa>33333333333333</cnpjEmpresa>
   <nomeEmpresa>EMPRESA XXXXXXXX</nomeEmpresa>
   <cpfEmpregado>33333333333</cpfEmpregado>
   <pisPasep>12345678900</pisPasep>
   <nomeMae>JOANA DOE</nomeMae>
   <carteiraTrabalho>01011</carteiraTrabalho>
   <serieCarteiraTrabalho>00099</serieCarteiraTrabalho>
</DadosPessoais>
</item>
</Empregados>
```

JSON (*J*ava*S*cript *O*bject *N*otation)

Uma alternativa bastante popular em relação ao XML é o formato JSON, foi criado para servir como um padrão com baixo custo de processamento que permitisse a transmissão de informações entre os lados cliente e servidor de um aplicativo web. Ele tem como características principais:

- É uma coleção de dados no formato *nome da informação : valor da informação*. Esse formato foi definido assim pois era a forma mais usual para definir as estruturas, ou uniões, de dados em aplicativos escritos nas linguagens C e C++, e que era facilmente tratada na forma de um vetor de dados, ou array;
- Ele tem uma estrutura de informações que pode ser facilmente entendida por seres humanos e máquinas;
- Apesar de ser um subconjunto da linguagem de programação JavaScript, a forma como a informação é tratada é totalmente independente da linguagem de programação;

Por tudo isso, ele se tornou bastante popular entre os desenvolvedores, pois ele tem as mesmas vantagens do XML, mas permite um uso menor de bytes para transmissão da informação, por não ser baseado em tags. Além disso, é mais simples escrever um interpretador de JSON do um interpetador XML.

No início, ele foi usado como substituto ao XML em aplicações web que usam Ajax, mas devido às suas vantagens sobre o padrão XML, com o passar do tempo, passou a ser usado para troca de informações em outros

aplicativos. Atualmente, portais como Google, Flickr, Yahoo e Facebook permitem que várias pesquisas sejam feitas em suas bases de dados por aplicativos externos, desde que o aplicativo seja capaz de entender a forma de se comunicar ao provedor de informações usando os protocolos baseados em JSON que esses portais definem.

A listagem a seguir apresenta um arquivo JSON que formata os dados pessoais de uma pessoa. Perceba que similar a algumas linguagens de programação, o escopo dos dados inicia com o *{* e termina com o *}*. Quando há informações repetitivas como os vários números de telefone de uma pessoa, esses blocos de informação são iniciados com *[* e terminados com *]*. Entre esses símbolos, há sempre o par *nome da informação : valor da informação*, como na linha *"primeiroNome": "Joao"*, e as informações são separadas por vírgula.

Listagem 3 - Exemplo de arquivo em formato JSON para transmitir dados sobre uma pessoa

```
{
    "primeiroNome": "Joao",
    "ultimoNome": "Smith",
    "idade": 25,
    "endereco": {
        "rua": "Rua Assis Brasil, 1000",
        "cidade": "Blumenau",
        "estado": "SC"
    },
    "telefones": [
        "5555-5555",
        "9999-9999"
    ],
    "emails": [
        {
            "tipo": "pessoal",
            "endereco": "joao@joao.com"
        },
        {
            "tipo": "profissional",
            "endereco": "joao.smith@algumaempresa.com"
        }
    ]
}
```

Quiz

Exercício Final

Metadados para Troca de Dados (XML e JSON)

INICIAR

Referências

CHOWDHURY, A., CHAUDHARY, P. *JAX - JAVA APIs For XML*. SAMS PUBLISHING. 2002

FRIESEN, F. *JAVA XML And JSON*. APRESS. 2016

Avalie este tópico

ANTERIOR
Servidores de Aplicação

Índice

PRÓXIMO
RMI, SOAP (Web Services) e REST

Ajuda? (https://ava.uninove.br/seu/AVA/topico/topico.php?idCurso=)