Extensible Markup Language (XML)

Wrocław, 15.03.2019 - Java - technologie zaawansowane

Wprowadzenie

XML jest językiem znaczników (ang. “markup language) używanym do definiowania zbioru zasad rozmieszczenia elementów na dokumencie w taki sposób, aby był łatwy do odczytu zarówno dla człowieka jak I maszyny.

Język XML zaprojektowano jako uproszczoną wersję języka SGML z myślą o zastosowaniach w Internecie. Jak często się zdarza, prostsze okazało się lepsze i język XML spotkał się z entuzjastycznym przyjęciem oraz szybko zdobył duże grono zwolenników, co nie udało się językowi SGML.

Przykładowy plik XML

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <title>
      <font>
         <name>Helvetica</name>
         <size>36</size>
      </font>
   </title>
   <body>
      <font>
         <name>Times Roman</name>
         <size>12</size>
      </font>
   </body>
   <window>
      <width>400</width>
   </window>
   <color>
      <red>0</red>
      <green>50</green>
      <blue>100</blue>
   </color>
   <menu>
      <item>Times Roman</item>
      <item>Helvetica</item>
      <item>Goudy Old Style</item>
   </menu>
</configuration>
```

Jak łatwo zauważyć, format XML pozwala wyrazić hierarchiczną strukturę danych i powtórzenia

elementów bez niepotrzebnych kombinacji.

Format pliku XML jest przejrzysty i zbliżony do plików HTML. Języki XML i HTML stanowią

bowiem pochodne języka SGML (*Standard Generalized Markup Language*).

Podobieństwo dwóch języków

Mimo że języki XML i HTML posiadają wspólne korzenie, to istnieje między nimi wiele różnic.

- W przeciwieństwie do języka HTML język XML uwzględnia wielkość znaków. Dlatego też na przykład znaczniki `<H1>` i `<h1>` będą w nim rozróżniane.

- W języku HTML możemy opuścić znaczniki końca akapitu `</p>` lub końca elementu Listy `</li>`, jeśli z kontekstu wynika, w którym miejscu powinny one wystąpić. W języku XML nie wolno opuszczać znaczników końca.

- W języku XML elementy posiadające pojedynczy znacznik, któremu nie odpowiada żaden znacznik końca, muszą kończyć się znakiem ukośnika, na przykład `<img src="coffeecup.png"/>`. Dzięki temu parser języka XML uzyskuje informację, że nie powinien szukać znacznika końca.

Struktura dokumentu XML

Dokument XML powinien rozpoczynać się nagłówkiem w postaci

```
<?xml version="1.0"?>
```

lub

```
<?xml version="1.0" encoding="UTF-8"?>
```

Chociaż użycie nagłówka jest opcjonalne, to jednak ze wszech miar zalecane.

Struktura dokumentu XML

Każdy z elementów może zawierać ***element podrzędny*** i (lub) tekst. W przykładzie z poprzedniego slajdu element `font` posiada dwa elementy podrzędne, `name` i `size`. Element `name` zawiera tekst `"Helvetica"`.

Struktura dokumentu XML

Elementy mogą też posiadać atrybuty, na przykład:
<size unit="pt">36</size>

Pomiędzy projektantami języka XML nie ma zgody co do tego, kiedy należy stosować elementy, a kiedy ich atrybuty. Wydaje się na przykład, że czcionkę łatwiej opisać za pomocą atrybutów

niż za pomocą elementów

<name>Helvetica</name>
<size>36</size>

Jednak atrybuty okazują się mniej uniwersalne od elementów. Załóżmy na przykład, że opis rozmiaru czcionki chcemy wyposażyć w jednostkę. Korzystając z atrybutów, umieścimy ją w wartości atrybutu.
<font name="Helvetica" size="36

Parsowanie dokumentów XML

Aby przetworzyć dokument XML, musimy najpierw go *parsować*. Parser jest programem, który wczytuje zawartość pliku, stwierdza poprawność jej formatu i rozkłada ją na elementy składowe, które udostępnia programiście. Istnieją dwa rodzaje parserów XML:

- **parsery drzewiaste, jak na przykład DOM (*Document Object Model*), które umieszczają elementy dokumentu XML w strukturze drzewa,**
- **parsery strumieniowe, jak na przykład SAX (*Simple API for XML*), które generują zdarzenia podczas czytania dokumentu XML.**

W większości technologi parsowanie jest już zaimplementowane co pozwala na szybkie i wygodne parsowanie plików XML.

Kontrola poprawności dokumentów XML

Załóżmy na przykład, że wczytaliśmy

poniższy element.

```
<font>
<name>Helvetica</name>
<size>36</size>
</font>
```

Przed parsowaniem dokumentu Musimy sprawdzić, czy nazwą jego znacznika jest "name"

a później, czy posiada on pojedynczy węzeł podrzędny typu Text. Ten ostatni warunek musimy

sprawdzić dla wszystkich węzłów podrzędnych, ponieważ autor dokumentu mógł zmienić

porządek tych węzłów lub dodać nowe. Napisanie odpowiedniego kodu jest pracochłonne

i łatwo w nim o pomyłkę.

Kontrola poprawności dokumentów XML

Jednak jedną z podstawowych zalet stosowania parsera dokumentów XML jest to, że może on automatycznie sprawdzić poprawność struktury dokumentu. Dzięki temu późniejsza jej analiza jest znacznie łatwiejsza. Jeśli na przykład element `font` przeszedł pomyślnie weryfikację, to możemy od razu pobrać dwa jego elementy podrzędne drugiego stopnia, rzutować je na typ `Text` i pobrać łańcuchy znaków bez żadnej dodatkowej kontroli.

Kontrola poprawności dokumentów XML

Aby określić strukturę dokumentu, dostarczamy plik definicji typu dokumentu (DTD) lub
definicję w języku XML Schema. Plik DTD lub definicja XML Schema zawiera reguły
opisujące format dokumentu przez określenie dopuszczalnych elementów podrzędnych i atrybutów
każdego z elementów. Przykładowa reguła może wyglądać następująco:
<!ELEMENT font (name,size)>

Kontrola poprawności dokumentów XML

Reguła ta wymaga, aby element `font` posiadał zawsze dwa elementy podrzędne — `name` i `size`.
W języku XML Schema te same ograniczenia moglibyśmy wyrazić w poniższy sposób:

```
<xsd:element name="font">
<xsd:sequence>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="size" type="xsd:int"/>
</xsd:sequence>
</xsd:element>
```

Kontrola poprawności dokumentów XML

Tabela 2.1. *Reguły określające zawartość elementów*

Reguła	Znaczenie
E*	0 lub więcej wystąpień E
E+	1 lub więcej wystąpień E
E?	0 lub 1 wystąpień E
$E_1 \mid E_2 \mid \ldots \mid E_n$	jeden z elementów $E_1, E_2, \ldots, E_n$
$E_1, E_2, \ldots, E_n$	E_1, po którym następują $E_2, \ldots, E_n$
#PCDATA	tekst
(#PCDATA$\mid E_1 \mid E_2 \mid \ldots \mid E_n$)*	0 lub więcej wystąpień tekstu, po którym występują $E_1, E_2, \ldots, E_n$ w dowolnym porządku (zawartość mieszana)
ANY	dowolny element podrzędny
EMPTY	nie jest dozwolony jakikolwiek element podrzędny

Następująca reguła określa, że element `menu` zawiera 0 lub więcej elementów `item`.

```
<!ELEMENT menu (item)*>
```

Literatura

- Cay S. Horstmann, Gary Cornell - Java. Techniki zaawansowane. Wydanie VIII, Wydawnictwo: Helion

- How to read XML File in Java Tutorial: https://www.journaldev.com/898/read-xml-file-java-dom-parser