Interacțiunea Om-Calculator

Laborator 5

DOM vs SAX

Cuprins

1 Obiective laborator

- Familiarizarea cu diverse metode de parsare a unui fișier XML

2 Introducere

După cum am arătat în primul laborator, XML se foloseşte pentru definirea unor fişiere de configurare, pentru implementarea unor limbaje (XSL, SVG, RSS), pentru definirea unor protocoale (de exemplu SOAP). Pentru a implementa instrumente care să prelucreze aceste documente rapid şi eficient au fost dezvoltate mai multe metode în funcție de particularitățile de prelucrare aferente fiecărui caz. Principalele metode de prelucrare ale fişierelor XML sunt DOM (Document Object Model) şi SAX (Simple API for XML). În cele ce urmează vom discuta aceste două metode de prelucrare, cazuri specifice de utilizare, precum şi o analiză comparativă a celor 2 abordări.

În acest context, de interes aparte sunt rezultatele comparative între diverse implementări de parsere [5], precum şi la proiectul Tilt [6], inovativ la nivel de vizualizare a structurii DOM aferentă unei pagini web.

3 SAX – Simple API for XML

SAX este un parser care prelucrează fișierele XML în mod serial declanșând evenimente la întâlnirea elementelor fișierului XML. Cu un parser de tip SAX dezvoltatorul nu are acces la structura efectivă a documentului, fiind necesar să implementeze mecanisme pentru a se asigura că prelucrează doar acele elemente pe care le dorește analizate. Dezvoltatorul trebuie să implementeze handlere (funcții asociate evenimentelor declanșate) pentru fiecare element de interes – element pe care dorește să-l prelucreze.

Principalele avantaje ale utilizării SAX sunt legate de faptul că permite prelucrarea stream-urilor XML pe măsură ce se primesc, consumă mult mai puțină memorie decât parserele DOM și din acest motiv permite prelucrarea unor documente XML care nu pot fi încărcate complet în memorie și care deci nu pot fi prelucrate cu DOM.

Ca dezavantaje putem menționa faptul că validarea unui fișier XML cu un parser SAX este mai dificilă și necesită câteodată parcurgerea întregului fișier sau, în anumite situații, multiple iterații pe aceeași sursă de date (de exemplu în cazul în care se folosește ID și IDREF trebuie parsat tot fișierul XML pentru a se ști dacă un IDREF nu punctează către un ID ce nu există)

Funcțiile generice utile pentru procesarea unui document sunt:

- **startDocument** – funcție apelată automat la întâlnirea elementului rădăcină;
- **endDocument** – funcție ce va fi apelată la întâlnirea marcajului de sfârșit al elementului rădăcină;
- **startElement** – funcție ce primește ca parametrii URI-ul namespace-ului documentului, numele local al elementului (numele ce nu include prefixul spațiului de nume), numele elementului ce include și prefixul spațiului de nume și care se găsește în documentație sub numele de „qualified name” sau „qname” și lista efectivă de atribute. În funcție de starea curentă a parserului și de acești parametri se decide comportamentul pe care trebuie să-l aibă parserul când întâlnește un nou element;
- **endElement** – ce are aceiași parametri cu startElement, mai puțin lista de atribute;
- **characters** – primește ca parametri un șir de caractere și datele referitoare la ce caractere din acest șir sunt utile; această funcție se folosește pentru tratarea conținutului efectiv al fișierului XML.

Adițional, comentariile și instrucțiunile de procesare pot fi tratate independent în cadrul parcurgerii secvențiale.

4 DOM – Document Object Model

Spre deosebire de SAX, DOM este un standard W3C independent de platformă. De asemenea, parsarea unui document XML folosind DOM presupune încărcarea întregului document în memorie. După ce documentul este încărcat, navigarea printre nodurile documentului se poate face în orice ordine folosind relațiile de ordine din interiorul arborelui XML. Astfel, se pot accesa rapid părinții, copiii și frații oricărui element. Aceste avantaje date de accesul rapid la conținut sunt compensate de consumul mare de memorie necesar de încărcarea documentelor XML complet în memorie. Totodată, în cazul unor documente XML foarte mari sau în cazul unor dispozitive cu resurse de memorie foarte limitate este posibil ca o prelucrare bazată pe DOM să nu poată fi făcută pentru că nu documentul nu încape în memorie.

Parserele de tip DOM întorc elemente de tip generic ”Document” și oferă o serie de metode standard prin care i se pot accesa elementele, textul și atributele.

Cele mai importante metode oferite la nivel de document (întreg document sau subarbore) sunt următoarele:

- **getDocumentElement** – oferă acces la elementul rădăcină al documentului;
- **getElementByID** – întoarce elementul ce are ID-ul egal cu șirul de caractere primit ca parametru de funcție;
- **getElementsbyTagName** – întoarce elementele ce au numele egal cu șirul de caractere primit ca parametru de funcție, în ordinea în care sunt întâlnite în document;

Aceste funcții întorc un obiect sau o listă de obiecte de tipul ”Element”, interfața care reprezintă un element oarecare din XML și care permite în principal următoarele operații:

- **getAttribute** – primește numele unui atribut și întoarce un șir de caractere ce reprezintă valoarea atributului
- **getChildNodes** – întoarce o lista cu elemente de tip ”Node” care sunt copii elementului curent. Interfața Node reprezintă interfața extinsă de Element și conține toate tipurile de noduri dintr-ul fișier XML. Pentru a identifica tipul unui anumit nod în scopul prelucrării sale se poate folosi funcția getNodeType;

De asemenea, pentru gestiunea eficientă a documentelor XML se recomandă utilizarea și a următoarelor funcții:

- **createElementNS** – primește ca atribute un spațiu de nume și numele unui element și întoarce un nou element
- **appendChild** – se adaugă un copil nodului (Document, Element) current
- **createAttributeNS** – adaugă un atribut elementului current
- **createTextNode** – construiește un nod de tip text

DOM oferă mecanisme relativ simple pentru creearea și prelucrarea fișierelor XML. Principiile sunt aproape identice în limbaje diferite de programare deoarece se bazează pe standarde comune.

5 SAX vs DOM

Criteriu	DOM	SAX
Standardizare	Da, definit formal şi stardard W3C	Generat de necesitatea în sine şi definit de comunitate
Dimensiunea fişierului	Recomandat pentru fişiere de dimensiuni mai mici, în special pe dispozitive cu memorie limitată	Suportă prelucrarea rapidă a unor fişiere de dimensiuni mult mai mari
Viteză	Mai lent	Mult mai rapid, având la bază o singură parcurgere secvențială
Memoria utilizată	Volum ridicat întrucât întreg arborele de parsare este păstrat în memorie	Mult mai puțină
Navigabilitate	Mult mai sporită, având la dispoziție întreaga structură de date	Acces secvențial, poate necesita inclusiv parcurgeri multiple
Uşurință în implementare	La latitudinea dezvoltatorului	
Aplicabilitate	Depinde de problema în sine, dar există situații în care intrinsec structura întregului arbore de parsare este necesară (XSLT, XPATH)	

6 Concluzii

Prelucrarea fişierelor XML este deosebit de facilă, existând un număr mare de API-uri în majoritatea limbajelor de programare de nivel înalt. Există modalități de prelucrare diferite şi alegerea unei metode se bazează pe mai multe aspecte dintre care cele mai importante ar fi următoarele:

- dimensiunea fişierului vs. cantitatea de resurse de memorie disponibile
- tipul de prelucrare efectuat (este o singură parcurgere suficientă sau necesare prelucrări succesive?)
- navigabilitatea dorită
- preferințele personale ale dezvoltatorului vizavi de facilitatea şi rapiditatea implementării unui parser.

7 Referinte

[1] Standardul DOM pe site-ul W3C - http://www.w3.org/DOM/

[2] Pagina oficială SAX - http://www.saxproject.org/

[3] Tutorial Oracle DOM - http://download.oracle.com/javaee/1.4/tutorial/doc/JAXPDOM.html

[4] Tutorial Oracle SAX - http://download.oracle.com/javaee/1.4/tutorial/doc/JAXPSAX.html

[5] Comparație implementări DOM vs SAX - http://www.devx.com/xml/article/16922/1954

[6] Tilt – plugin Firefox pentru vizualizarea structurii unei pagini web http://blog.mozilla.com/tilt/