

① Két parserrel foglalkoztunk:

SAX parser és a DOM parser

Sax: Egyszerű programozható API felület
szekvenciális feldolgozása
csak olvasási műveletet támogat.

Az XML-t szövegfájlként kezeli

Feldolgozás menete: először parseolás
majd elemenkénti feldolgozás

DOM: Document Object Model

A SAX-hoz képest ez már objektum orientált
támogat olvasást és írást is.

~~ez is szekvenciálisan halad~~

A parseolás után létrehoz egy DOM fát
és ezen a fa struktúrán lépésről lépésre dolgozzuk
fel az XML-t.

Az egységei az elemek attribute és a node.

PÓROS PÉTER

LHS256

2021.01.12.

② XPath

Az XPath alapvetően az XML fájlokat feldolgozást segíti dokumentumréteket kifejezésre hozni. Számos módon lehet navigálni az XML dokumentumban ezt a kifejezést: self, child, descendant, parent, ancestor, following, attribute.

self: . ; parent: .. ; descendant: //

attribute: @;

hasonló műveletek találhatók általában aggregációs művelet és az alapművelet is támogatja

count(); position(); last(); name();

concat(); contains(); sum(); round(); és a matematikai operátorok: +; -; *; div; mod; /; and; or; not();

(3) DOM

Rugalmasan programozható API felület; objektum orientált
Alapvető működési elv: Beolvas egy XML dokumentumot
majd abból előállít fát, és ezen a fán keresztül tudjuk
kezelni a dokumentumot. Az objektumok közötti relációs kapcsolat
írasi, olvasási műveletek támogat
navigációs műveletek alapszik

Elterjedt,

Implementálása nehezebb mint a SAX-é viszont alapjaitban a
SAX-ra épül.

Nagy a memória igénye.

Több függvényekhez is implementált.

Feldolgozás lépései:

(1) Dokumentum olvasó létrehozása:

DocumentBuilderFactory → DocumentBuilder → Document

(2) Parse olvas

(3) Documentből elemeket létrehozása.

~~szó~~ megkezdésül a gyökerelemet és utána Dom fán keresztül
navigációval dolgozunk fel a Dokumentumot.

interface:

3 fő osztály: Document, Element, Node

A node a legkisebb elem a hierarchiában. Node-ból kétféleképpen
típus hozzájárulással Elementet.

Főbb módszerek:

createElement(); createAttribute(); getDocument(); createTextNode();
getAttributes(); appendChild(); getChildNode();
getNodeValue();

④ XML Schema

Szerepe: XML ~~fej~~ dokumentum validálása
integritás ellenőrzése, szemlélteti az adatrendszer jelentését
eggyeztetíti a hűelő programokat, támogatja az automatizált
adattárolást.

Főbb típusok: Statikus, csak szerkezet és értelmezés

Dinamikus, a művelet köré is korlátozás

globális elemre épülő séma: minden név egy elemet azonosít
lokális elemre épülő séma: egy névhez több elem a különböző
szekvenciákban.

Nomane séma: a szerkezetnek nincs neve

Nemesített séma: a szerkezetek nemesítéssel, újra felhasználhatók

Szigori séma: kötelező, nem kerülhető meg.

Optionális séma: nem kötelező

DTD: elemi séma; korlátozott integritási elemek; nincs adattípus
megjelölés; globális nyelvezet; nomane típusok; szoros kapcsolat
az XML-hez; nincs névtér kezelés; entity támogatás

<! Doctype rootnode > - séma megadási nyel

<! Element ~~new~~, szerkezet > - elem, ~~struktúra~~

<! Attrib elem attr. típus > - attribútum

Element: igazodó minden "tag" elemet ide vonat

az összetett elemek reprezentációját egy üres tartalommal elemet
és az alá rendelt elemekből áll

minden elemnél lehet előfordulási korlátozás.

⑤ XSL

Document - fa modell de nem DOM szerkezetű
- nem módosítható



XML transzformációkat tudunk vele csinálni, szűréseket, aggregációs műveleteket állapokban.

A forrás XMLből egy XSLT XML fájl segítségével + transzformálással az XSLT processzor segítségével a cél XML se.

A forrás XML tartalmazza az adatokat

Az XSLT XML a műveleteket tartalmazza XPath szabványra építve

Cél: A forrás XMLből egy részeltávolítás kiírása a cél XML se.

1. XML beolvasása
2. fa modell létrehozása
3. a fa gyökerétől elindul a feldolgozás
ha van meghatározott elem:
- igen - pontos végrehajtás
- nem - implicit pontos

Tehát alapvetően mintaillesztéssel dolgozza fel az XML fájlt.