DOM szabványy

DOM szabvány

Az előadás anyaga

Prof. Dr. Kovács László: Adatkezelés XML környezetbe és

Kollár Lajos, Sterbinszky Nóra: Programozási technológiák

jegyzete és további irodalom alapján készült el

Témakör kérdései

1. Az XML-DOM modell szerepe
2. DOM API elemei
3. DOM API program felépítése
4. Mintafeladat

Igényelt kompetenciák

- Az XML-DOM modell megismerése
- DOM API elemeinek áttekintése
- DOM API program felépítésének elsajátítása
- Környezet: XML szerkesztő (Oxygen, EditIX, Eclypse, ….)

Mi az XML? – ismétlés

- *„Szűkebb értelemben*:

  – Szintaxis *strukturált dokumentumok ábrázolására*, mely lehetővé teszi azok automatikus feldolgozását (elektronikus dokumentum formátum).

- *Tágabb értelemben:*

  – Egy sereg *közös tőről fakadó specifikációt jelent*, melyeket összefoglaló néven *XML családnak* is neveznek.

XML adatok kezelési lehetőségei – ismétlés

Magával az XML-lel kapcsolatos specifikációk:

– Az XML lehetőségeit bővítik.

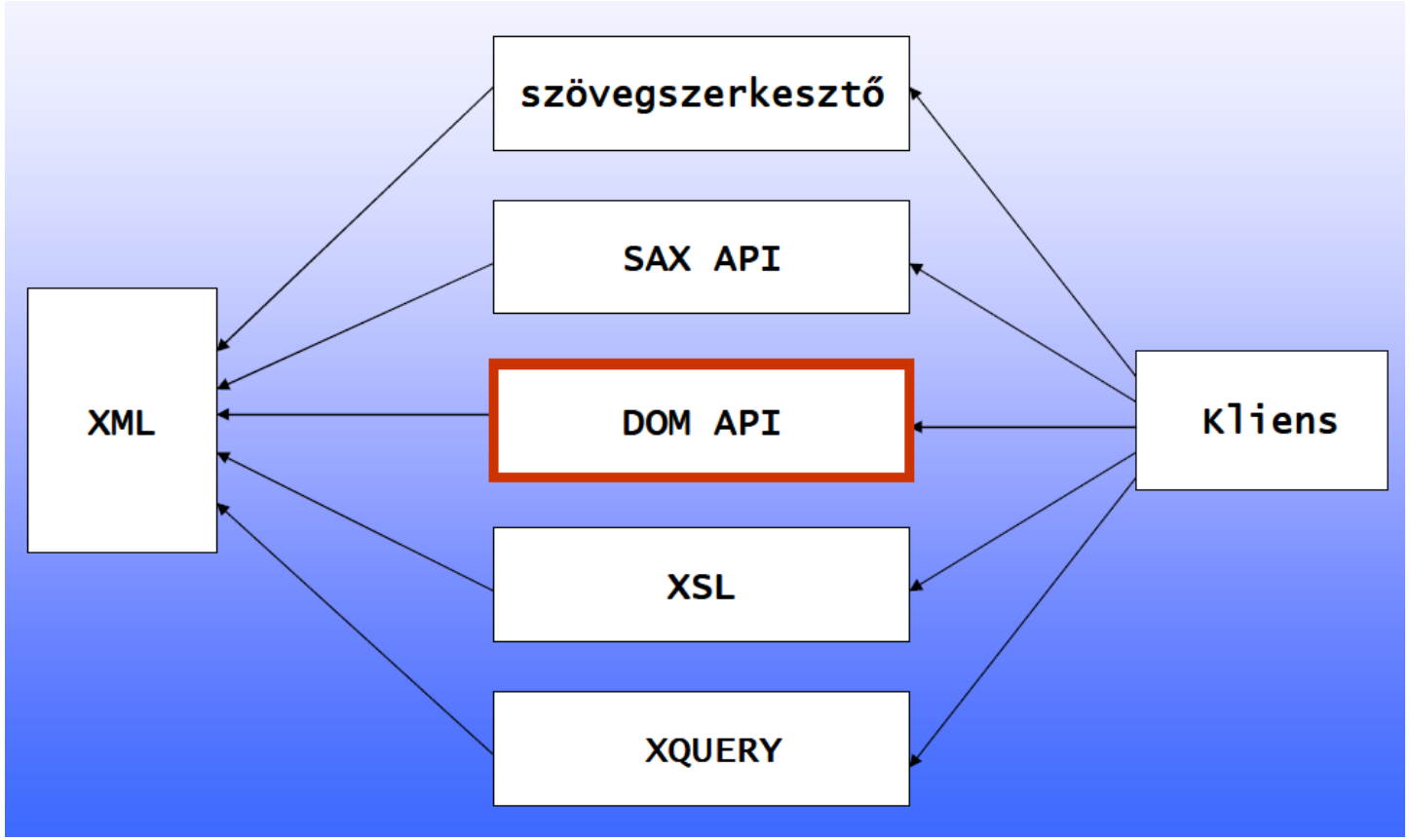
– Lehetővé teszik XML dokumentumok *szerkezetére* és *tartalmára vonatkozó megszorítások kifejezését* (*XML sémanyelvek).*

– Lehetővé teszik XML dokumentumokból *információ kinyerését (lekérdező nyelvek).*

– Lehetővé teszi XML dokumentumok *más formába alakítását (transzformációs nyelvek).*

XML adatok kezelési lehetőségei – ismétlés

Több XML-API szabvány fejlődött ki.



Java XML – Parsers – ismétlés

Mi az XML-parser?

- Az *XML-parser* lehetőséget nyújt az XML dokumentum adatainak *elérésére* vagy *módosítására*.
- A *Java több lehetőséget* kínál az *XML dokumentumok elemzésére.*
- XML dokumentumok elemzésére általánosan használt különféle értelmezők:

Java XML - Parsers

- *SAX parser* - az *eseményalapú API, callback mechanizmust* használ – nem épít fát. Ilyen a SAX.
- *DOM parser* - a dokumentumot egy *belső fa struktúrába képeznek le*, majd lehetővé az adott fában navigáljon. **DOM**.
- *JDOM parser* - A DOM egy nyílt forráskódú, *Java alapú könyvtár* az XML dokumentumok elemzésére.

Java-gyűjteményeket használ, mint a List és a Arrays.

Java XML - Parsers

StAX parser - Streaming API for XML processing.

Ez is része a standard Java könyvtáraknak.

A SAX PUSH modell - az elemző hívja a kezelő függvényeit,

StAX PULL modell – a kezelő hívja az elemző függvényeit.

DOM és a SAX előnyei ötvöződnek.

Példa: StaxNEPTUNKOD

Java XML - Parsers

- *XPath Parser* - XML dokumentumot értelmezése ún. *kifejezés alapján* történik.
- *DOM4J parser* - Java alapú könyvtár az XML, az XPath és az XSLT elemzéséhez.

  Ez egy rendkívül *rugalmas és memóriatakarékos API.*

  Támogatást nyújt a DOM, a SAX és a JAXP számára.

Java XML - Parsers

DOM4J parser - külső függőségre van szükség.

A `pom.xml` fájlba kell beleírni a következő sorokat

```
<dependency>
    <groupId>org.dom4j</groupId>
    <artifactId>dom4j</artifactId>
    <version>2.1.0</version>
</dependency>
```

Java XML - Parsers

JAX-B - Java Architecture for XML Binding (Java Architektúra XML kötéshez):

- Nem kifejezetten DOM és SAX típusú feldolgozó.
- Képes Java objektumokat XML formátumban elmenteni és onnan visszaolvasni.
- Nincs szükség saját feldolgozó implementációt írni.
- A JAX-B Java SE 1.6 verziója óta létezik.

Java XML - Parsers

Jackson

Előnye, eredetileg *JSON feldolgoz*áshoz lett tervezve, arra is használható ugyan ezen az elven.

Nem tartalma az alap Java-nak, külső függőségként kell hozzáadni a *pom.xml*-hez.”

```
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
    <version>2.9.8</version>
</dependency>
```

Document Object Model – DOM

„A *dokumentum objektummodell*, vagyis a DOM *objektummodell* a szó klasszikus, objektumorientált értelmében.

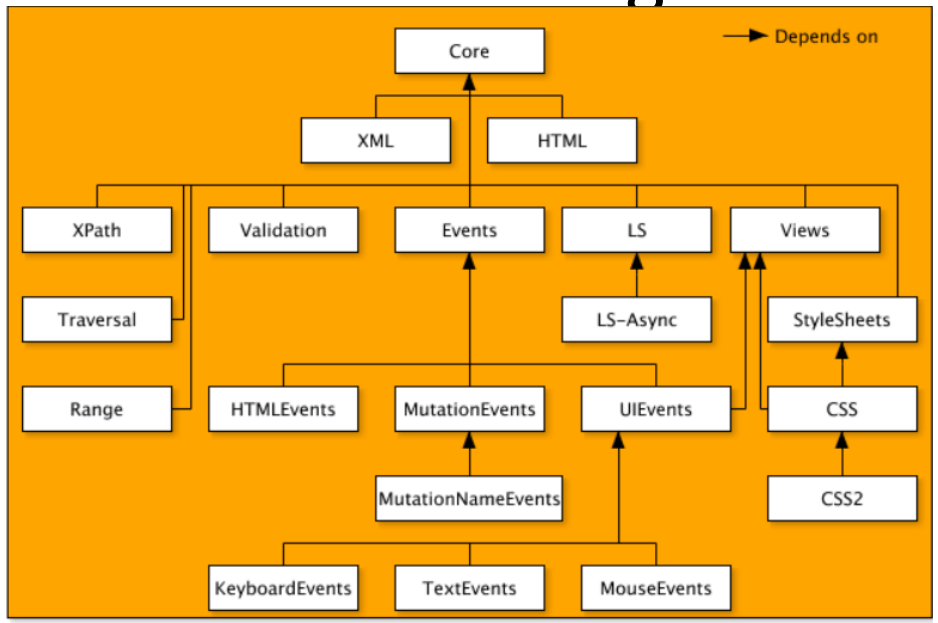
A DOM API elemei az *org.w3c.dom* csomagban helyezkednek el.

Lásd: Package org.w3c.dom

https://docs.oracle.com/javase/7/docs/api/org/w3c/dom/package-summary.html

DOM modulok

- A DOM *felépítése moduláris*, az egyes modulok jól körülhatárolható funkciót valósítanak meg.
- **DOM architektúrája**



https://www.w3.org/TR/DOM-Level-3-Core/Overview.html#contents

DOM modulok

- A legfontosabb modul *Core modul*, az összes többi modul *közvetlenül* vagy *közvetve* kapcsolódik a *Core*-hez.
- Ez attól függ milyen *objektumokat*, *interfészeket* és azok *szolgáltatásait* veszik igénybe.

További információk:

Lásd: Document Object Model (DOM) Level 3 Core Specification

https://www.w3.org/TR/DOM-Level-3-Core/Overview.html#contents

DOM modell szerepe

A DOM API kezelő felület különféle *funkciót, műveleteket* is lehetővé tesz (pl.: elemzés, lekérdezés, új létrehozása, módosítás).

DOM (Document Object Model) tulajdonsága:

- a *kezelő felület* a memóriában felépíti a *teljes dokumentum objektumot*,
- a rendelkezésre álló *metódusokon keresztül elvégezhető* a *tartalom átolvasása és módosítása* is.

DOM modell szerepe

A DOM modell az XML dokumentumot *fa ábrázolásával* írja le. Hogyan?

A DOM API az XML fa kezelésére *elemeket* tartalmaz.

Az *objektum* jelleg arra utal, hogy *mind a teljes fa*, mint annak *elemei* egy-egy ***objektumként kezelhető***.

Igy a *teljes fát ábrázoló dokumentum mellett* megjelennek *az elemeket*, *a szövegrészeket*, az *elemjellemzőket leíró* ***objektumok*** is.

DOM modell szerepe

A fa kezelési logikája *navigációs jellegű*.

Ez azt jelenti, hogy a ***fa bejárása*** során egy *feldolgozási pozíció* jelenik meg, amely megmutatja, hogy a *fa melyik része*, mely *objektuma* a *feldolgozás alatt álló elem*.

Az *elem váltáskor* az *aktuális elem szomszédaira lehet átlépni, navigálni.*

A *módosítás műveleteknél* az *aktuális elem környezetét*

lehet módosítani.

DOM API szabvány

A DOM API **osztályok gyűjteménye**, melyben a

- *adattagok* – az *elemek jellemzőit* írják le,
- *metódusok* – a *navigációra* és a *tartalom kezelésére* szolgálnak.

A *programozó feladata:* olyan *programkódot* írni, melyben a *példányosítja a kezelő osztályokat* és *elvégzi a fa módosítását* a rendelkezésre álló *metódusok* segítségével.

DOM API szabvány

A *DOM modell nagyobb végrehajtási költséggel* jár, mint a SAX modell.

Oka: hogy a DOM első lépésben *átolvassa a megadott bemeneti XML dokumentumot*, majd a memóriában felépíti a hozzá kapcsolódó *objektum hierarchiát.*

A DOM egyik kritikus pontja: hogy a rendelkezésre álló *memória véges*, ezért a hatékonyan feldolgozható *XML dokumentum méret is korlátos*.

DOM API szabvány

DOM API szabvány több verzióban jelent meg:

- az induló verzió 1998-ban készült el,
- jelenleg a 2004-es DOM 2 verzió a legfrissebb kiadás.

A DOM szabvány a *funkciókat szintekre osztja*:

- Az *alap (Level 1) szint:* a *fa struktúra* és a *tartalom navigációs parancskészletet* foglalja magába.
- *Level 2* szint: bevezetésre kerültek a *névterek* és *események kezelése.*

DOM API szabvány

- *Level 3* az *XPath* és a *validáció* támogatása.

A DOM szabványt különböző feldolgozók (parser) is támogatják.

DOM támogatást biztosító termékek:

- Internet Explorer 4.0-tól,
- Netscape 4.X,
- JAXP (Java API for XML Processing),
- MSXML (Microsoft XML-értelmező).”

DOM API szabvány

„Fejlesztő környezet a *JDeveloper.*

A *JDeveloper* az Oracle cég ingyenes *IDE fejlesztő környezete*,

mely a *Java és XML támogatás* mellett szervesen integrálja az *SQL és PL/SQL környezetek* is.

Lásd: **Oracle JDeveloper 12c (12.2.1.4.0)**

https://www.oracle.com/tools/downloads/jdeveloper-12c-downloads.html

DOM API elemei

A DOM API *egy faszerkezetet épít* fel a memóriában az XML dokumentumban *található elemekből*, amelyek két leggyakoribb típusa:

- az *elem csomópont* és a
- *szöveges csomópont.*

A DOM API használata lehetővé teszi: *csomópontok létrehozását*, *törlését,* valamint a *csomópontok tartalmának* és magának a d*okumentumszerkezetnek* a megváltoztatását.

DOM jellemzői

- Document Object Model,
- W3C standard,
- Platform és nyelv független interfész,
- A dokumentum elemeinek a *tulajdonságait*, *objektumait* és *metódusokat* adja meg,
- Fa hierarchiát kezel (bejárás és módosítás),
- Egyszerűbb implementálni (fabejárási algoritmusok),

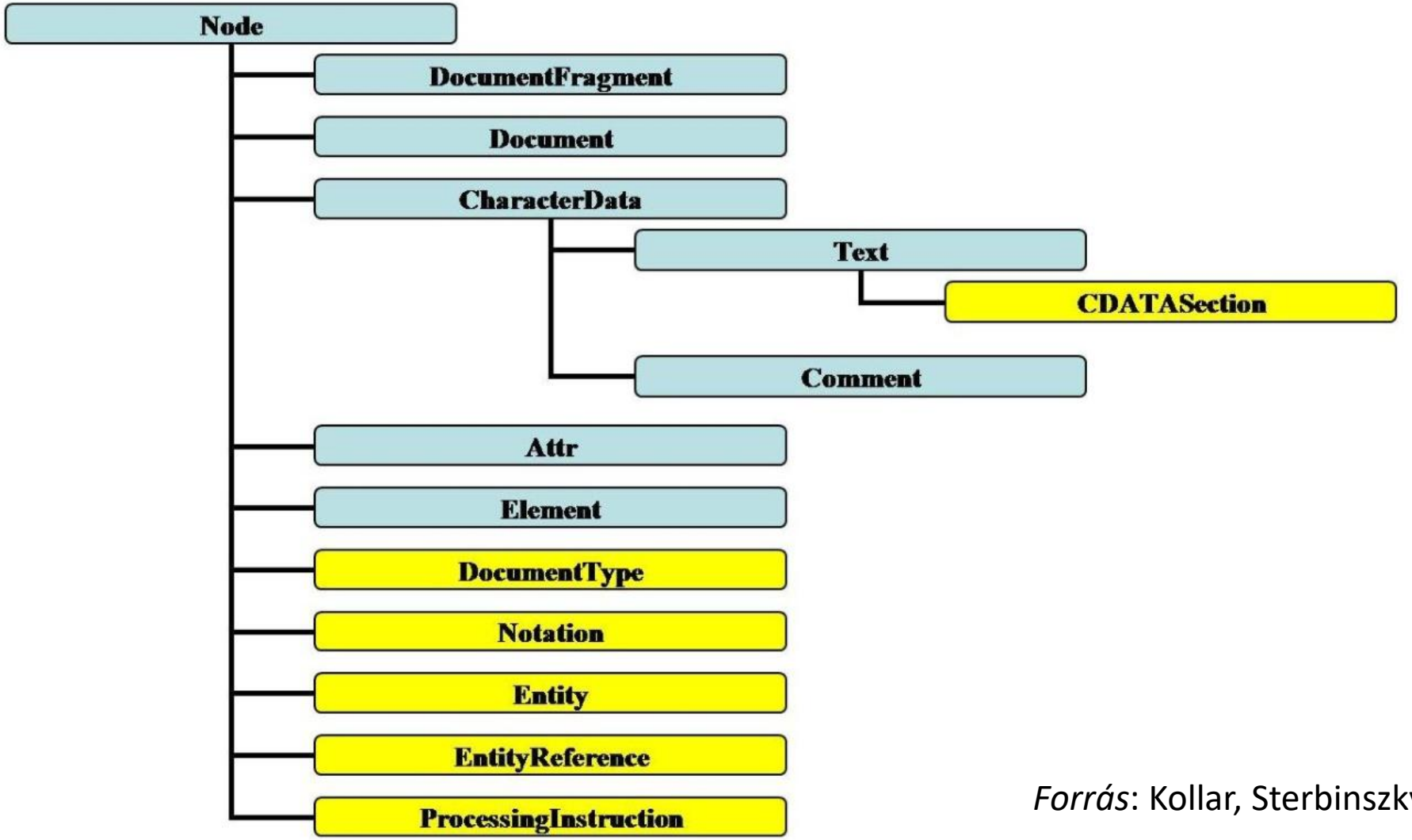
DOM jellemzői

- Nagy memóriaigény.
- A teljes dokumentum *egy dokumentum csomópont.*
- Minden XML elem *egy elem csomópont*.
- Az XML elemekben lévő *szöveg*, mind *szöveg csomópont*.
- Minden attribútum *egy attribútum csomópont*.
- A megjegyzések *megjegyzés csomópontok*.
- Interfész-hierarchia, API.

DOM jellemzői

- A *dokumentumot logikailag faként kezeli,*
- *Objektummodell a klasszikus OO értelemben*,
- *Alkalmas a dokumentumok*:
  - létrehozására, felépítésére,
  - szerkezetének bejárására,
  - elemek ill. tartalom *hozzáadásár*a, *módosítására*, *törlésére*.
- Modulokból áll (kékkel a DOM Core , sárgával az XML DOM).

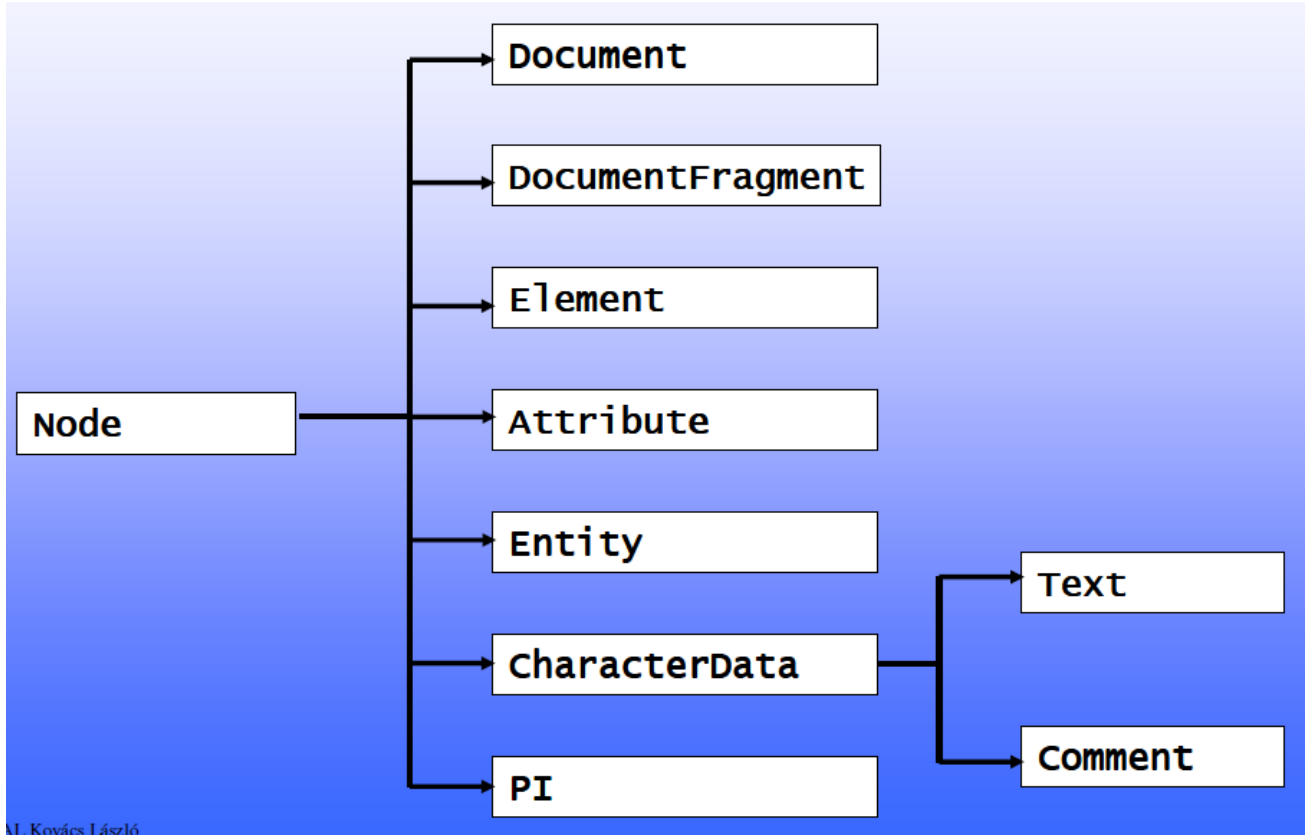
DOM jellemzői – moduljai (részlet)



Forrás: Kollar, Sterbinszky

DOM kezelő felület

A DOM *osztály-kapcsolat* modellje



Forrás: KovácsL

DOM node types

A *fában* az alábbi *csomópont típusokat* különbözteti meg:

- **`dokumentum:`** a teljes fát reprezentálja,
- **`dokumentum séma leírás:`** a fa szerkezetére vonatkozó *megkötések*, *integritási szabályok* csomópontja,
- **`tartalomelem csomópont`**: egy *általános üres* vagy *összetett elem csomópont*,
- **`szöveg csomópont,`**
- **`megjegyzés csomópont,`**

Forrás: KovácsL

DOM node types

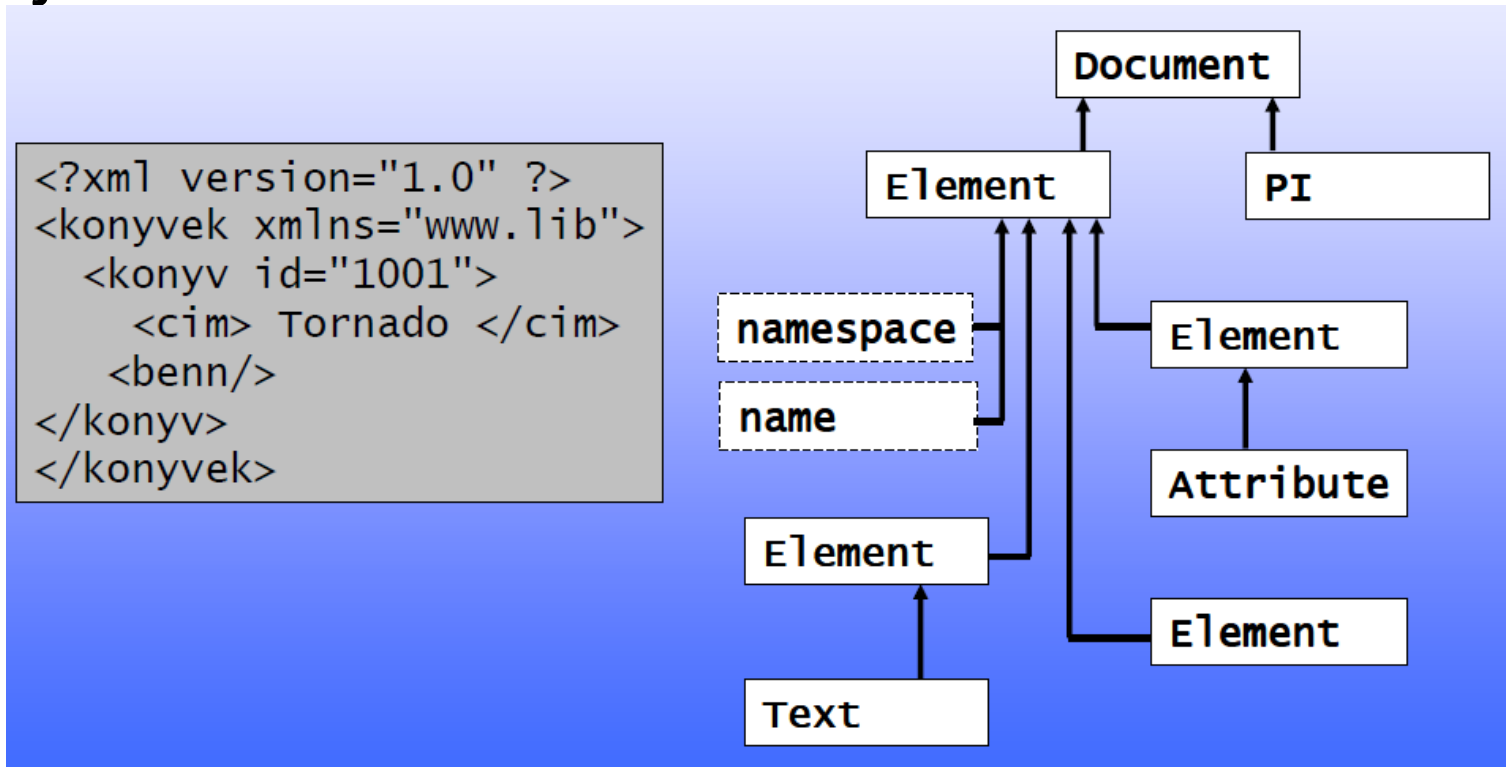
- **`direktíva csomópont,`**
- **`CDATA csomópont,`**
- **`elem hivatkozás csomópont,`**
- **`elem jellemző csomópontja.`**

Az egyes elemek között *tartalmazási reláció állítható* fel.

A DOM *értelmező, feldolgozó* a bemeneti XML dokumentumhoz előállítja az *XML-fát*.

DOM kezelő felület

Az objektumok között *tartalmazási reláció* áll fenn.



Forrás: KovácsL

DOM interfaces

- **`Node`** - a DOM alap adattípusa.
- **`Element`** - egy *elem* **Element** képviseli.
- **`Attr`** - egy *elem* attribútumát képviseli.
- **`Text`** - az *elem* vagy az *Attr* tényleges tartalma.
- **`Document`**- a teljes XML dokumentumot képviseli. A Document objektumot gyakran *DOM fának nevezik*.

Common DOM methods

- **`Document.getDocumentElement()`** - visszaadja a dokumentum gyökérelemét.
- **`Node.getFirstChild()`** - az adott csomópont *első gyermekét* adja vissza.
- **`Node.getLastChild()`** - egy adott csomópont *utolsó gyermekét* adja vissza.
- **`Node.getNextSibling()`** - egy adott csomópont *következő testvérét* adják vissza.

Common DOM methods

- **`Node.getPreviousSibling()`** - egy adott csomópont *előző testvérét* adják vissza.
- **`Node.getAttribute (attrName)`** - egy adott csomópont esetében a *kért névvel adja vissza* az *attribútumot.*

Minta XML dokumentum - mintapélda

Minta XML dokumentumból – példa – DOM fa létrehozása - jegyzetbe.

```
<?xml version="1.0" ?>
<!-- mintapélda -->
<konyvek xmlns="www.lib.org/books">
  <konyv id="1001">
    <cim> Tornado a godorben </cim>
    <ibook:isbn xmlns:ibook="www.book.org/codes">
      1156511-568-89 </ibook:isbn>
    <ar pnem="Ft"> 2455 </ar>
    <benn/>
    <kiado idref="K354" />
    <szerzo> Kiss Péter </szerzo>
    <szerzo> Kovacs Lajos </szerzo>
  </konyv>
</konyvek>
```

Az XML-fa DOM objektum modellje

Példa mutatja, hogy egész kis XML dokumentumok

esetében is elég *terebélyes DOM-fa jön* létre.

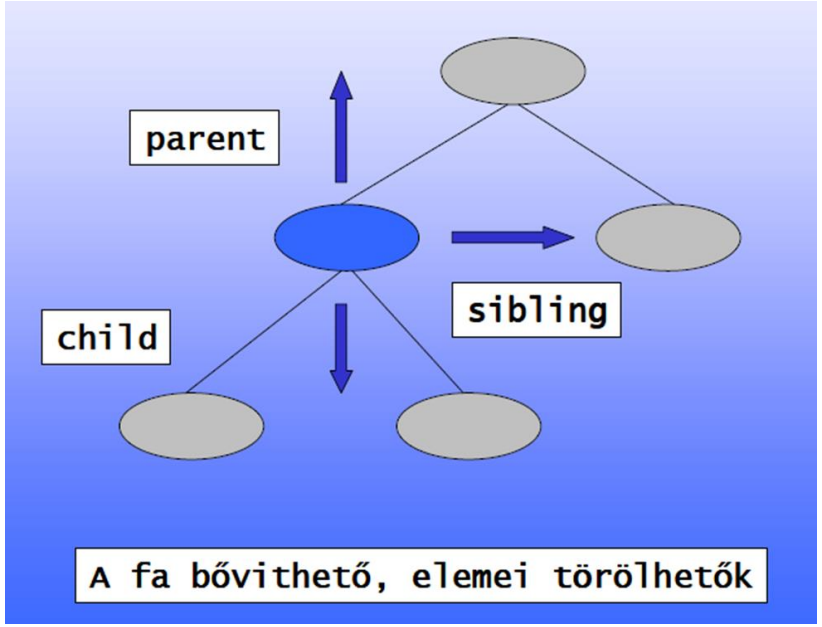
A *feldolgozás sorá*n a fában, ezen *csomópontok mentén* lehet *navigálni*.

A *módosításokat* az *aktuális csomóponthoz kapcsolódóan* lehet elvégezni.

Az XML-fa DOM objektum modellje

A modellben az alábbi *navigációs lépések* állnak rendelkezésre:

- mozgás a *gyerekek fele,*
- navigáció a *testvér csomópontok* fele,
- átlépés a *szülő csomópontra*.



Forrás: KovácsL

DOM API elemei

DOM API szabvány i*nterface-eket* definiál.

A DOM API tartalmazza:

- az egyes *csomópont típusokat* megvalósító *interface-ket*,
- *összetettebb funkciókat ellátó elemek* is megjelentek a szabványban.

DOM API elemei - Node interfész

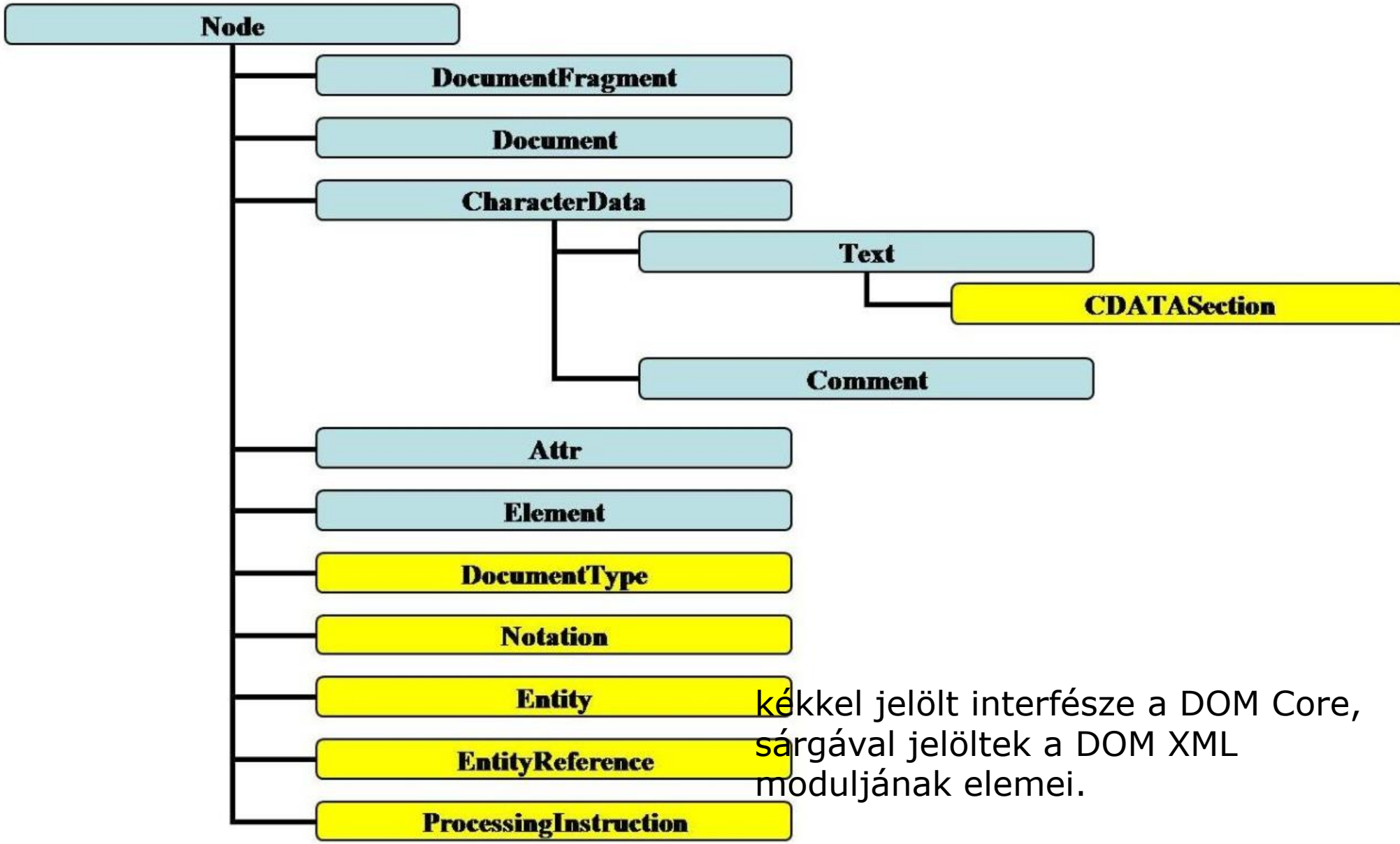
A DOM az XML-dokumentumot *faként kezeli,* amely *Node típusú* csomópontokból épül fel.

A Node interfész un. *alinterfészei* az *XML-specifikáció által megengedett egységeket* írják le.

Pl.: az Element, a DocumentType, a CharacterData, stb.

DOM A

DOM API interfészei



kékkel jelölt interfésze a DOM Core, sárgával jelöltek a DOM XML moduljának elemei.

DOM API elemei - Document

DOM API fontosabb interface elemei:

`Document`*:* a dokumentum csomópont.

Adattagok:

- *documentElement:* a fa gyökér eleme - *Element típ*usú,
- d*octype*: séma leíró - *DocType típusú*.

A dokumentum *objektum szerepe*: a teljes dokumentum *egységbe zárása* és az alkotó *elemek teljes körű kezelése*.

DOM API elemei - Document

Az *interface főbb metódusai*:

- `Element createElement (String elemnev):`

  új elem objektum létrehozása.

- `Text createTextNode(String szöveg):`

  szöveg csomópont létrehozása.

- `Comment createComment (String szöveg):`

  megjegyzés csomópont létrehozása.

DOM API elemei - Document

- `Attr createAttribute` (String attr. neve): *elemjellemző létrehozása*.
- `NodeList getElementsbyTagName` (String elemnév): a fa megadott névvel *rendelkező csomópontjainak összegyűjtése*.

DOM API elemei - Node adattagok

Node: általános fa csomópont. Fontosabb **adattagok:**

- *`nodeType`*: csomópont *típusa* (pl. elem, szöveg), típusa: short
- *`nodeName`*: csomópont *neve*, típusa String
- *`nodeValue`*: csomópont *értéke*, típusa String
- *`parentNode`*: szülő *csomópont, típusa* Node
- *`childNodes`*: gyerek *csomópontok halmaza*, típusa: NodeList
- *`nextSibling:`* testvér *csomópont, típusa*: Node
- *`Attributes:`* jellemzők *halmaza, típusa*: NamedNodeMap

DOM API elemei – Node metódusok

- *ownerDocument: tulajdonos* dokumentum objektum - típusa: Document

Az *objektumhoz tartozó metódusok*:

- `Node removeChild(Node elem)`: a megadott *gyermek csomópont* kitörlése a fából,
- `Node appendChild(Node elem)`: a megadott *csomópont beszúrása a fába* az aktuális csomópont alá gyermekként,

DOM API elemei - Node metódusok

- `Node replaceChild(Node elemuj, Node elemregi)`: a *megadott gyermek csomópont helyettesítése* egy másik elemmel,
- `Node insertBefore(Node elemuj, Node elemregi)`: a megadott csomópont beszúrása a fába *az aktuális csomópont alá gyermekként*, a megadott létező gyermekkel.

DOM API elemei - CharacterData

`CharacterData`: általános szövegrész, a *Node* interface-ből származtatott.

Két adattagja van:

- `data:` szöveges adat leírása - típusa String,
- `length:` a szöveg hossza - típusa long.

A *szövegkezelő metódusok listája:*

- `String substringData` (long kezdet, long hossz): szövegrész kiemelése

DOM API elemei - CharacterData

`deleteData` (long kezdet, long hossz): szövegrész kitörlése,

`insertData` (long kezdet, String szöveg): szövegrész beszúrása a megadott pozícióra.

DOM API elemei - Attr

`Attr:` *elemjellemző leírása*, a Node interface-ből származtatott.

Adattagok:

- `name:` az jellemző azonosító neve - típusa String,
- `specified:` van-e értéke a jellemzőnek vagy sem - típusa: Boolean,
- `value:` a jellemző értéke - típusa: String,

Az objektumhoz ***nem tartozik saját metódus***.

DOM API elemei - Element

`Element`: *tartalom elem* csomópontja, a Node interface-ből származtatott.

Speciális metódusok:

- `String getAttribute` (String név): az *attr. értékének lekérdezése az attr. neve* alapján,
- `setAttribute` (String név, String érték): a megadott *attr. értékének beállítása*,

DOM API elemei - Element

- `removeAttribute(String nev)`: megadott *attr. eltávolítása a fából,*
- `Attr getAttributeNode (String nev)`: a megadott névvel rendelkező *attr. csomópont objektum lekérdezése,*
- `NodeList getElementsbyTagName(String nev)`: a megadott névvel *rendelkező leszármazott csomópontok listájának előállítása,*

DOM API elemei - NodeList

`NodeList:` *csomópontok halmaza*, *adattagja* és *egy metódusa van:*

- `length`: a tartalmazott csomópontok darabszáma, típusa: long,
- `Node item(long index):` a megadott sorszámú csomópont lekérése.

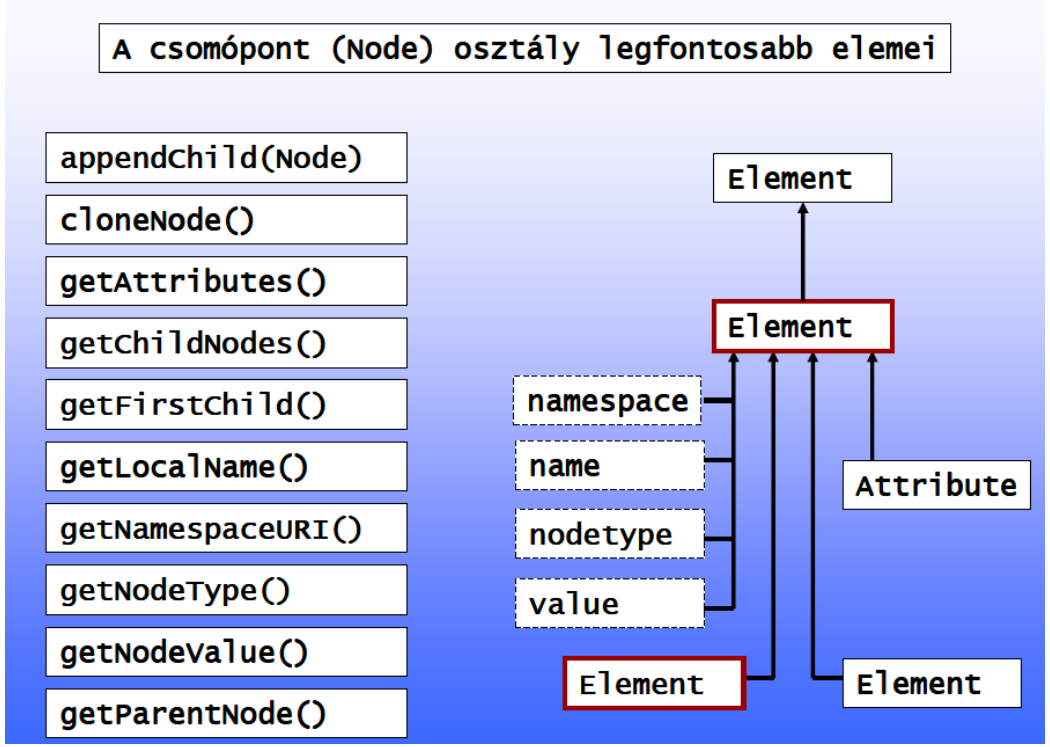
DOM API elemei - `Exception`

`DOMException:` DOM *kezelő felület által generált hiba.*
Egyetlen attribútuma van, melynek neve:

- `code:` hibakód, short típusú

DOM kezelő felület

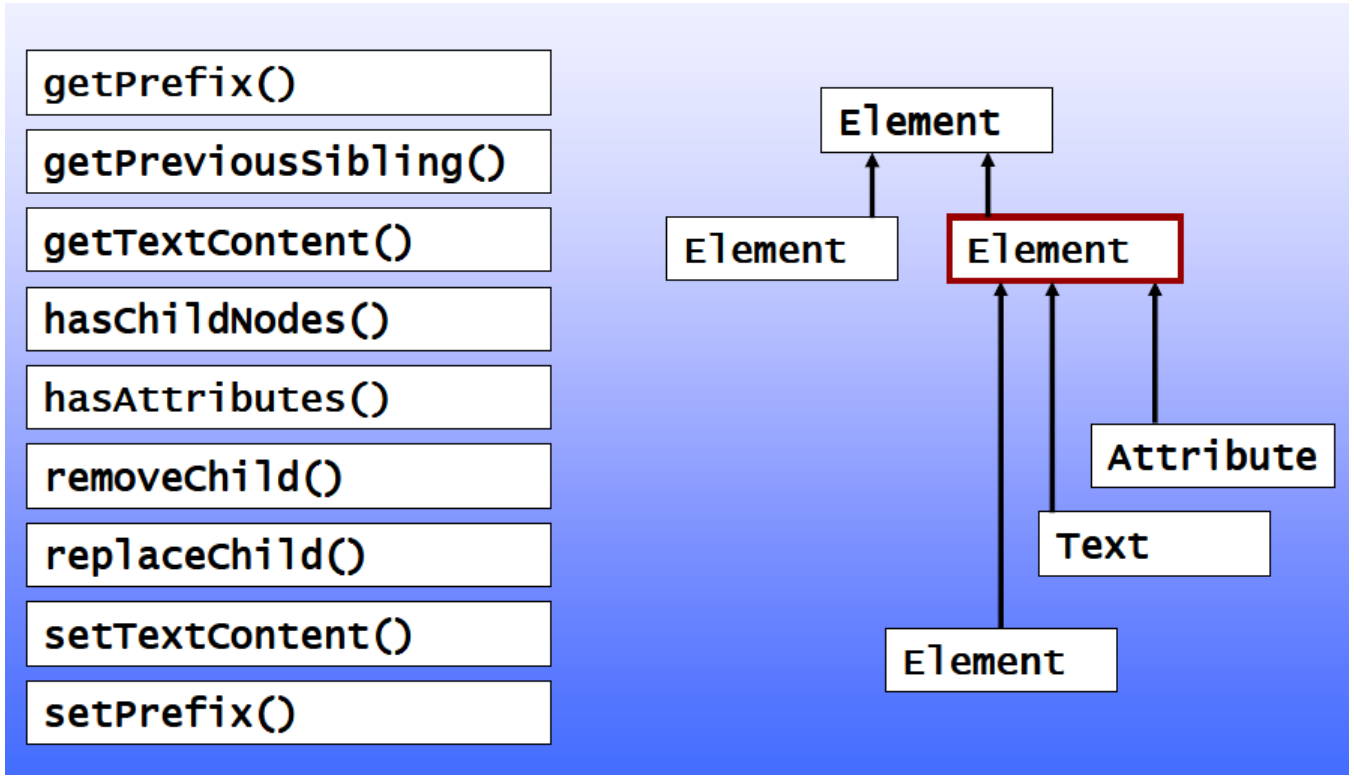
A *csomópont* (Node) osztály legfontosabb elemei.



Forrás: KovácsL

DOM kezelő felület

A csomópont (Node) osztály legfontosabb elemei



Forrás: KovácsL

DOM feldolgozás fontosabb lépései

1. Dokumentumolvasó létrehozása,

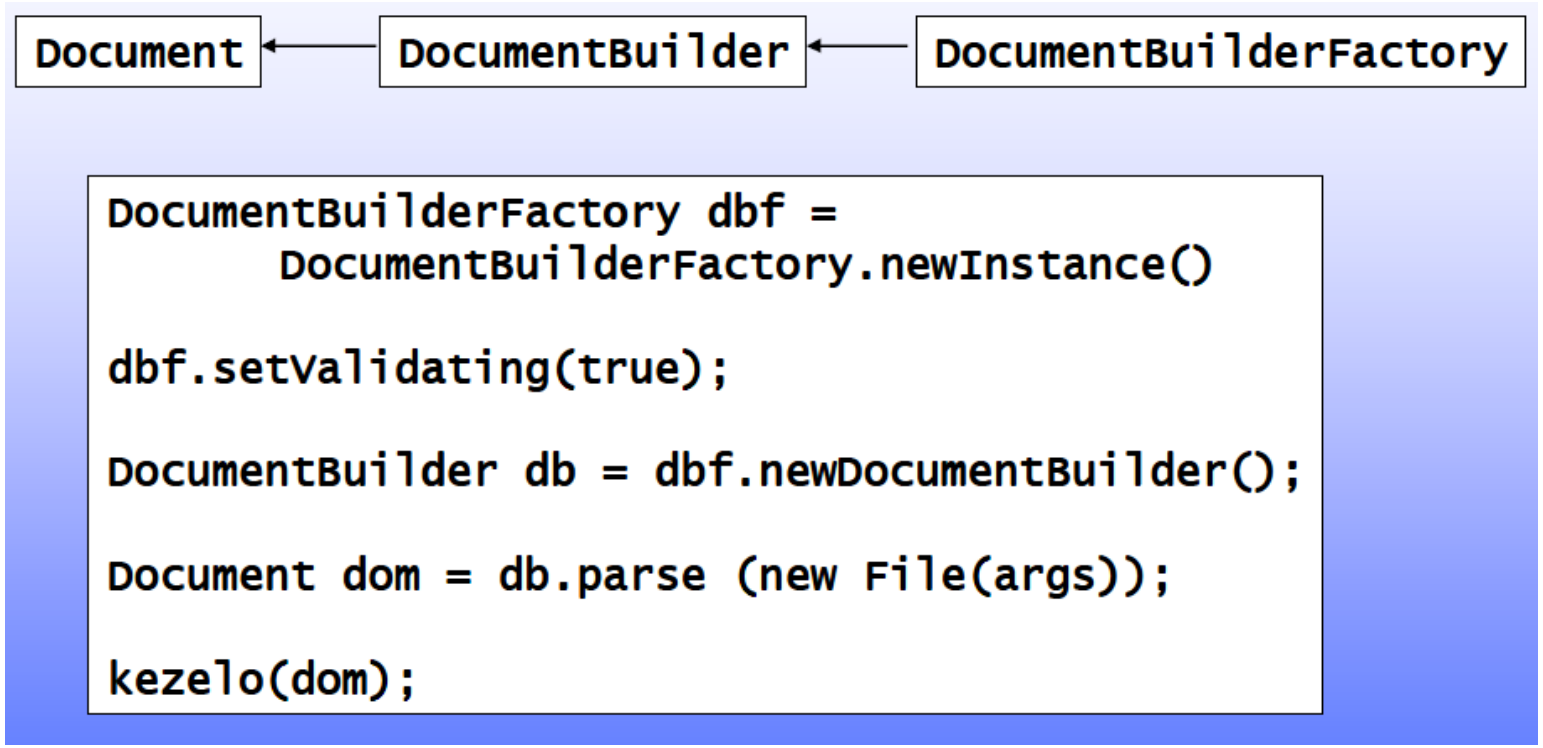
2. Objektumfa létrehozása,

3. Dokumentum *tagelemek* elérése,

4. Dokumentum metódusok meghívása,

5. Dokumentum szerializálása.

Dokumentum olvasó létrehozása

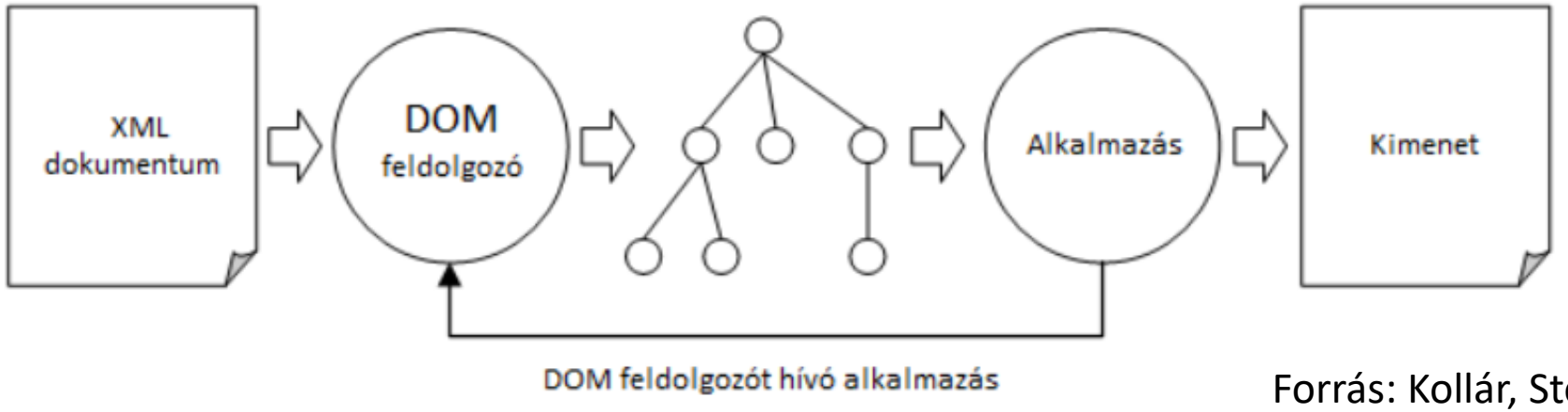


Forrás: KovácsL

Feldolgozás sematikus modellje

Az XML dokumentumok *DOM stílusú feldolgozásának sematikus modellje.*

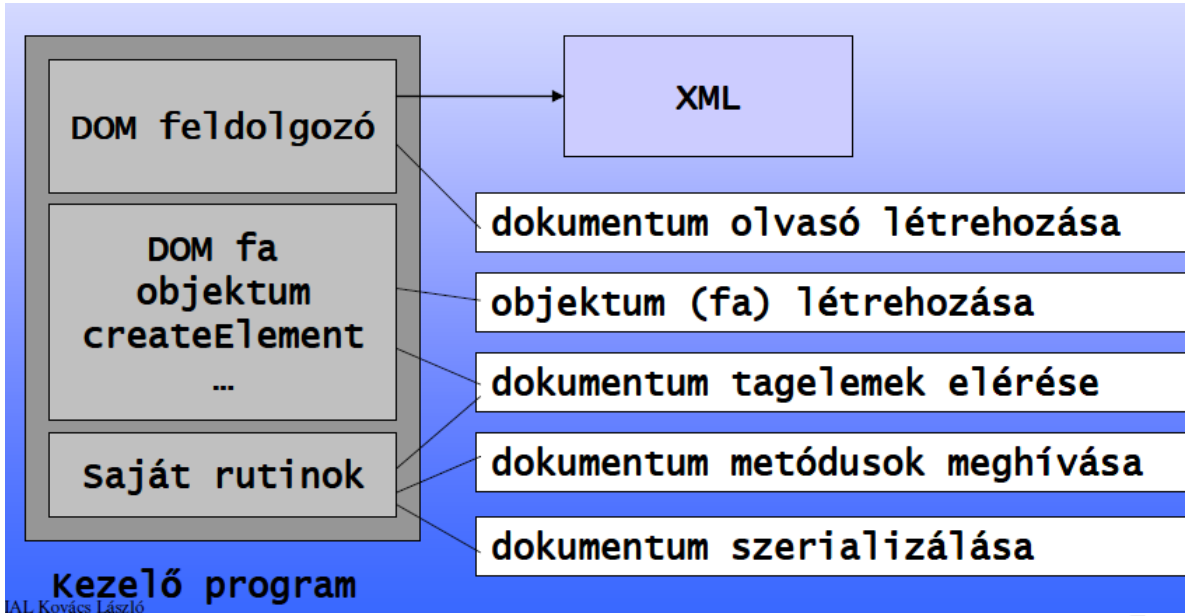
- *Előnye:* a közvetlen hozzáférés a csomópontokhoz, ami általában jól illeszkedik a Java oldali megközelítéshez.



Forrás: Kollár, Sterbinszky

DOM kezelő felület

A feldolgozó létrehoz egy *DOM objektumo*t, melyen keresztül lehet az XML dokumentum *tartalmát lekérdezni* és módosítani.



Forrás: KovácsL

DOM API program felépítése

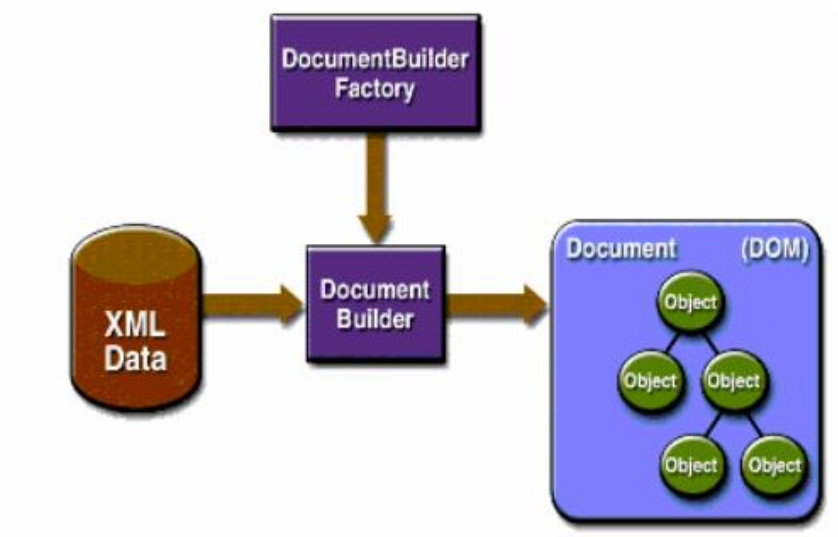
A `javax.xml.parsers.DocumentBuilderFactory` osztályt használjuk egy `DocumentBuilder` objektumpéldány létrehozására, amely egy `Document` interfészt megvalósító objektum előállítására szolgál.

- A *DocumentBuilder* osztályban:
- egy új, üres dokumentum létrehozására a `newDocument` *metódus,*

DOM API program felépítése

A *DocumentBuilder* osztályban:

- létező dokumentumok feldolgozására `a parse metódusok` szolgálnak.



forrás: Kollar, Sterbinszky

DOM API program felépítése

Először *létre kell hozni egy objektumot (*`DocumentBuilder`*),* amely alkalmas a *dokumentum DOM modell alapú* kezelésére.

Ezt a `DocumentBuilderFactory` típusú objektum `new DocumentBuilder` metódusa hozza létre.

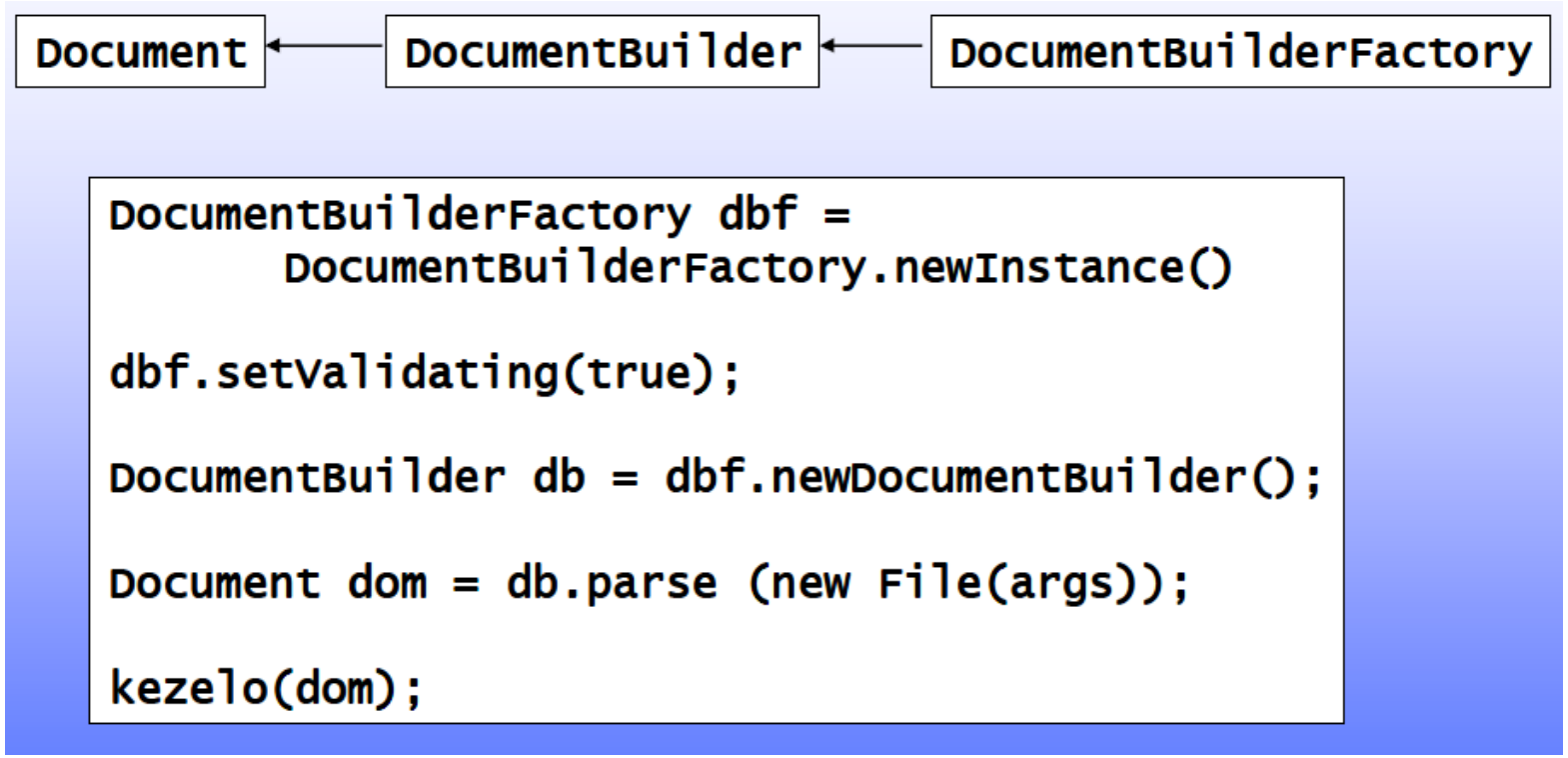
Példányosításra a `DocumentBuilderFactory` osztály statikus `newInstance()` metódusa szolgál.

DOM API program felépítése

A két utasítássor, mely minden JAXP (Java API for XML Processing) program elejére kerül a következő:

```
DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
DocumentBuilder db = dbf.newDocumentBuilder();
```

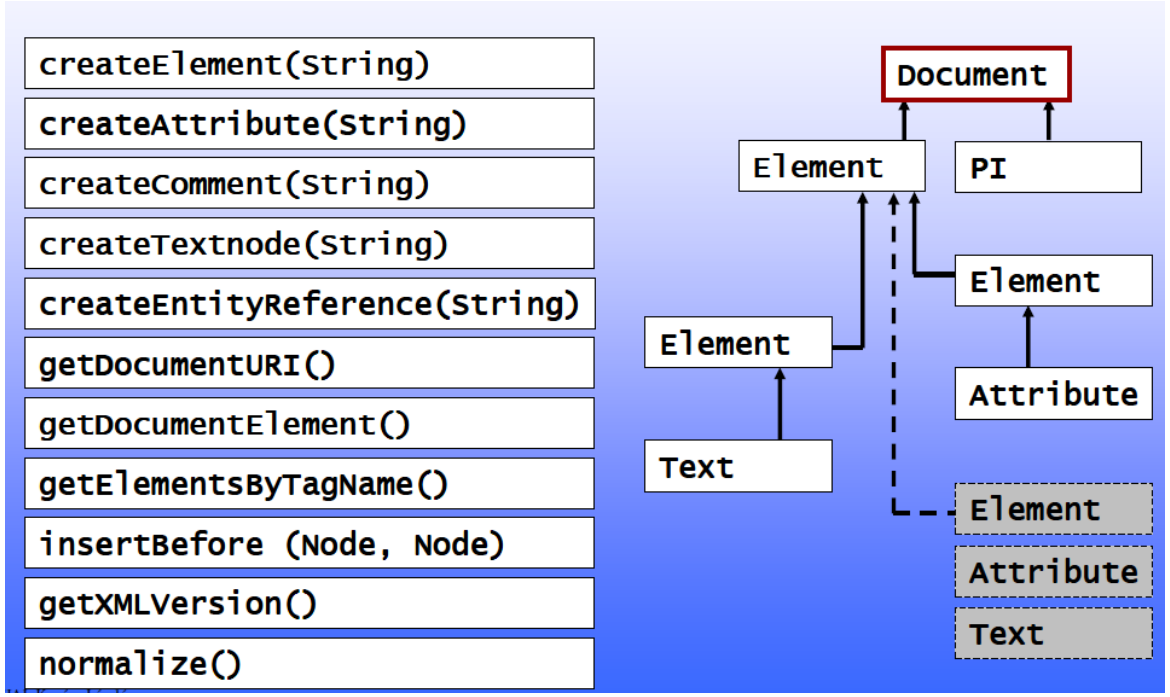
Dokumentum olvasó létrehozása



Forrás: KovácsL

Dokumentum olvasó létrehozása

A dokumentum (Document) legfontosabb elemei



Forrás: KovácsL

DOM API program felépítése

A létrejött *DOM értelmező* már alkalmas a *DOM fa előállítására.*

Ehhez `a DocumentBuilder` objektum `parse()` metódusát kell meghívni.

A `parse() metódus` az igényelt `Document` objektumot adja vissza:

```
Document dom = db.parse(new File(args));
```

Dokumentumolvasó létrehozása

```
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder builder =
factory.newDocumentBuilder();
```

Objektumfa létrehozása

A `parse() metódus` az igényelt *Document objektumot* adja vissza.

```
Document document = builder.parse(new File("XML dokumentum"));
```

Gyökérelem elérése

Gyökérelem elérése:

```
Element element = document.getDocumentElement();
```

DOM kezelő felület

Gyökérelem lekérdezése:

```
Document dok;
Node root = dok.getDocumentElement();
```

Gyerekelemek bejárása:

```
Node elem, gyerek;
for (p=0; p<elem.getChildNodes().getLength(); p++) {
      gyerek = elem.getChildNodes().item(p);
       …
}
```

Elem nevének és tipusának lekérdezése:

```
Node elem;
if (elem.getNodeType()==Node.ELEMENT_NODE) {
      if (elem.getNodeName().compareTo("ar") == 0) {}
         …
}
```


Forrás: KovácsL

DOM kezelő felület

Elemjellemzők bejárása:

```
Node elem;
for (p=0; p<elem.getAttributes().getLength();p++) {
       Node elemuj = elem.getAttributes().item(p);
        …
}
```

Szövegérték növelése:

```
Node elem;
ear = Integer.parseInt(elem.getTextContent());
String ujszov = String.valueOf(ear+nov);
elem.setNodeValue(ujszov);
```

Forrás: KovácsL

DOM kezelő felület

Új elem felvitele

```
Document dok;
Node ujelem, elem;
Node ujelem = dok.createElement("konyv");
elem.appendChild(ujelem);
ujelem.setTextContent("Babits");
```

Elem törlése

```
Node elem, gyerek;
for (p=0; p<elem.getChildNodes().getLength(); p++) {
            gyerek = elem.getChildNodes().item(p);
      elem.removeChild(gyerek);
}
```

Forrás: KovácsL

DOM API program felépítése - ellenőrzés

A séma ellenőrzést: a `dbf.setValidating(true);`

Mivel a fordítónak ismernie kell az **osztályok, metódusok szignatúráját**, a *felhasznált API elemeket deklarálni kell a program elején.*

A Java környezetben erre az *import utasítás* szolgál.

DOM kezelő felület

XML dokumentum validálása

A feldolgozó opcionálisan elvégzi a bejövő dokumentum validálását is a kapcsolódó séma alapján

```
DocumentBuilderFactory dbf =
        DocumentBuilderFactory.newInstance();
dbf.setValidating(true);

DocumentBuilder db = dbf.newDocumentBuilder();
db.setErrorHandler(new hibakez());

Document dom = db.parse(new File(fnev));
```

Az XML dokumentum gyökérelemében megadott sémára ellenőriz

Forrás: KovácsL

DOM kezelő felület

A saját hibakezelő osztály megadása

```
public class hibakez implements ErrorHandler {
    public hibakez() { }

    public void warning(SAXParseException exception) {
            System.out.println(exception.getMessage());
    }

    public void error(SAXParseException exception) {
            System.out.println(exception.getMessage());
    }

    public void fatalError(SAXParseException exception) {
            System.out.println(exception.getMessage());
    }
}
```

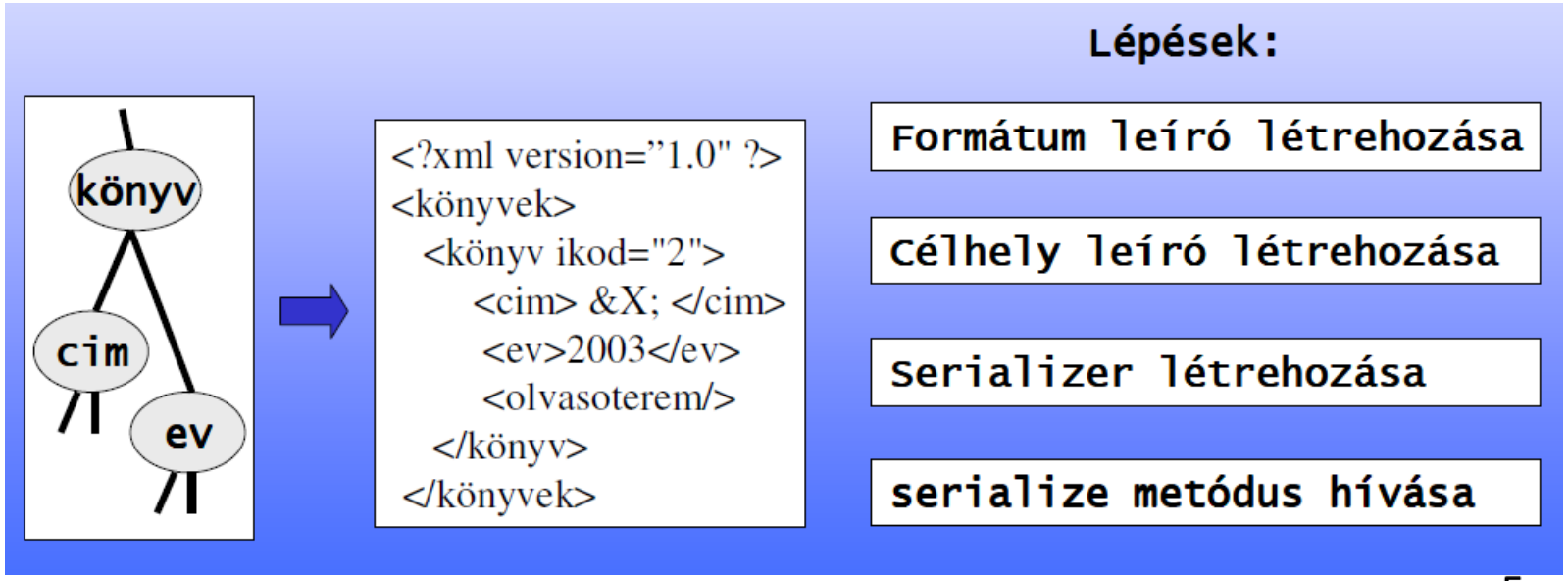
Forrás: KovácsL

DOM kezelő felület

XML dokumentum kiírása állományba.

A szabvány ezen funkciót nyitottan hagyta.

Xerces serialization: egyik megvalósítási lehetőség



Forrás: KovácsL

DOM kezelő felület

XML dokumentum kiírása állományba

```
String fnev2 =
     "c:\\users\\kovacs_l\\jegyzetxml\\proba_ki.xml";

Document dom = db.parse(new File(fnev));

OutputFormat f = new OutputFormat(dom);

XMLSerializer outp = new XMLSerializer
    ( new FileOutputStream (new File(fnev2)) , f);

outp.serialize(dom);
```

Forrás: KovácsL

DOM API program felépítése - példa

A programok fontos része a fellépő hibák felderítése és kezelése.

A DOM specifikus hibákat külön szeretnénk kezelni, akkor a DOM kezelő részt egy olyan `try...catch blokkba` tesszük, melyhez csatoljuk az alábbi *hiba típus észlelését*:

`SAXException:` a dokumentum átolvasása során keletkezett hiba.

Mintafeladat

Mintafeladat-ban megtalálható az adatkezelés négy legfontosabb alapművelete:

- adott kulcsú *rekord keresése*,
- új *rekord felvitele*,
- adott *kulcsú rekord törlése*,
- adott tulajdonságú *rekordok tartalmának módosítása*.

Mintafeladat

A minta XML doku
melyben az egyes k

Minden *rekord két*
elemekkel valósít m

A két mező a könyv

```
<?xml version="1.0" ?>
<konyvek>
  <konyv>
    <cim>Google1</cim><ar>20233</ar>
  </konyv>
  <konyv>
    <cim>Boomi</cim><ar>2233</ar>
  </konyv>


</konyvek>
```

Mintafeladat

A programban az alábbi *egyedi metódusok szerepelnek*:

- `query1(Document dom)`: a fa teljes tartalmának ki listázása,
- `query2(Document dom, int ar)` : a megadott árnál drágább könyvek ki listázása,
- `update1(Document dom, String nev, int ar)` : a megadott című könyvek árának módosítása a megadott ár értékre,
- `delete1 (Document dom, String név, int ár)` : a megadott című és áru könyvek kitörlése,

Mintafeladat

`insert1 (Document dom, String név, int ár)` : új könyv egyed létrehozása a megadott címmel és értékkel.

A mintafeladat a jegyzetbe!!!

DOM – DomRead mintapélda

Készítse el a következő *DomReadNEPTUNKOD* feladatot, amely egy *usersNeptunkod.xml* dokumentumot olvas be, majd a fa teljes tartalmának kilistázza konzolra.

Projekt név: DomReadNEPTUNKOD

Package: domneptunkod1026

DOM file name: DomReadNEPTUNKOD.java

XML name: users.xml

https://docs.oracle.com/javase/tutorial/jaxp/dom/index.html

DOM – DomWrite mintapélda

Készítse el a következő *DOMWriteNEPTUNKOD* feladatot, amely egy *usersNeptunkod.xml* dokumentum tartalmát fa struktúra formában kiírja a *konzolra* és egy *user1Neptunkod.xm*l fájlba.

Projekt név: DomWriteNEPTUNKOD

Package: domneptunkod1026

DOM file name: DomWriteNEPTUNKOD.java

XML name: users1Neptunkod.xml

DOM – DomQuery mintapélda

Készítse el a következő *DOMQueryNEPTUNKOD* feladatot, melyben egy *autokNeptunkod.xml* superautok gyerek elemek esetén a *Super* és a *Luxus* autó nevét és típusát kérdezze le.

Projekt név: DomQueryNEPTUNKOD

Package: domneptunkod1026

DOM file name: DomQueryNEPTUNKOD.java

Minta XML file: autokNeptunkod.xml

DOM – DomModify mintapélda

Készítse el a következő *DOMModifyNEPTUNKOD* feladatot, amely egy *autokNeptunkod.xml* superautok gyerek elemének pl.: egyik auto nevét módosítsa egy másik autó nevére, majd a módosított eredményt írja ki a konzolra a minta alapján. A *luxusauto* elemet törölje le.

Projekt név: DomModifyNEPTUNKOD

Package: domneptunkod1026

DOM file name: DomModifyNeptunkod.java

Minta XML file: autokNeptunkod.xml”

.Net (C#) és XML

Alap névtér: System.XML

Elemei:

- System.XMl.Schema
- System.XML.Serialization
- System.XML.XPath
- System.XMl.Xsl

.Net (C#) és XML

Alaposztály: XMLNode

Alosztályok:

- XMLDocument
- XMLDocumentFragment
- XMLEntity
- XMLAttribute
- XMLElement
- XMLText
- XMLComment

.Net (C#) és XML

Feldolgozó környezetek:

- XMLTextReader: SAX-hez hasonló
- XMLValidatingReader: Séma kezelés
- XMLTextWriter: dokumentum szerializálása
- XMLDocument : DOM alapú

.Net (C#) és XML

TextReader működése

- Nincs callback mechanizmus.
- Közvetlenül kérdezhető a következő esemény.

```
XmlTextReader tr = new XmltextReader(”aaa”);
while (tr.read()) {
if (tr.NodeType == XmlNodeType.Element) {
}
}
tr.close();
```

.Net (C#) és XML

TextWriter működése

```
XmlTextWriter ki = new XmlTextWriter(”file”,null);
ki.WriteStartDocument();
ki.WriteStartElement(”enév”);
ki.WriteAttributeString(”név”,”ertek”);
ki.WriteString(”szöveg”);
ki.WriteEndElement();
ki.WriteEndDocument();
ki.close();
```

.Net (C#) és XML

XmlDocument működése:

```
XmlDocument doc = new XmlDocument();
doc.Load (”file”);
XmlNode root = doc.DocumentElement();
XmlNodeList list =
root.SelectNodes(”query”);
doc.Save(”file”);”
```

Felhasznált irodalom

- Kovács László: Adatkezelés XML környezetbe (lásd: moodle)
- Kollár Lajos, Sterbinszky Nóra: Programozási technológiák – Jegyzet, 2014.
  https://gyires.inf.unideb.hu/GyBITT/21/ch06.html#idp46260096
- Package org.w3c.dom
  https://docs.oracle.com/javase/7/docs/api/org/w3c/dom/package-summary.html
- Jeszenszky Péter: XML, DE, 2019

Felhasznált irodalom

- DOM modules

  https://www.w3.org/TR/DOM-Level-3-Core/introduction.html
- Document Object Model (DOM) Level 3 Core Specification

  https://www.w3.org/TR/DOM-Level-3-Core/Overview.html#contents
- Oracle JDeveloper 12c (12.2.1.4.0)

  https://www.oracle.com/tools/downloads/jdeveloper-12c-downloads.html