**Webes adatkezelő környezetek**

**Féléves feladat  
XML dokumentumok DOM API alapú feldolgozása**

**Készítette:** Ilyó-Kovács Levente  
**Neptunkód:** NLFUA8

**Dátum:** 2024. december 2.

## Tartalomjegyzék

1. [Bevezetés](#bevezetés)
2. [Első feladat: XML dokumentum előkészítése](#első-feladat-xml-dokumentum-előkészítése)
   1. [ER modell](#er-modell)
   2. [XDM modell](#xdm-modell)
   3. [XML dokumentum](#xml-dokumentum)
   4. [XMLSchema](#xmlschema)
3. [Második feladat: DOM API programozás](#második-feladat-dom-api-programozás)
   1. [Adatolvasás](#adatolvasás)
   2. [Adatírás](#adatírás)
   3. [Adatlekérdezés](#adatlekérdezés)
   4. [Adatmódosítás](#adatmódosítás)
4. [Következtetés](#következtetés)

## Bevezetés

A modern informatikai rendszerek egyik legfontosabb alapköve a strukturált adatkezelés, különös tekintettel az adatok tárolására és cseréjére. Az XML (Extensible Markup Language) széles körben alkalmazott szabvány, amely lehetővé teszi az adatok hierarchikus szervezését, interoperabilitását és rugalmasságát.

A DOM (Document Object Model) API a webes technológiák egyik legismertebb programozási interfésze, amely az XML dokumentumok fa struktúráját dolgozza fel. A projekt során a DOM API segítségével négy fő feladatot oldottunk meg:

* Az XML dokumentum olvasása,
* Az XML dokumentum írása,
* Az adatok lekérdezése,
* Az adatok módosítása.

A projekt központi témája egy **gépjárműkölcsönző rendszer**, amely az XML segítségével tárolja és kezeli az adatokat. A rendszer modelljét először ER modell formájában terveztem meg, majd XDM modellé alakítottam, és végül XML dokumentumként implementáltam. Az adatstruktúra validációját egy XSD séma biztosítja, amely az adatok integritását garantálja.

## Első feladat: XML dokumentum előkészítése

### ER modell

Az adatbázis ER modellje egy **Gépjármű** központi entitás köré szerveződik, amelyhez kapcsolódnak a karbantartási rekordok, kölcsönzések, vásárlók és kölcsönző cégek adatai.

#### **Kapcsolatok:**

* **Gépjármű – Karbantartás**: Egy gépjárműhöz több karbantartás tartozhat (1:N).
* **Karbantartás – Szerelő**: Egy karbantartás egy szerelőhöz kötött (1:N).
* **Gépjármű – Kölcsönzés**: Egy gépjármű több kölcsönzésben szerepelhet (1:N).
* **Kölcsönzés – Vásárló**: Egy vásárló több kölcsönzéshez kapcsolódhat (N:1).
* **Kölcsönzés – Kölcsönző cég**: Egy kölcsönzés egy adott cégnél történik (N:1).

#### **Attribútumok:**

* **Gépjármű**: JárműID, Rendszám, Márka, Típus, GyártásiHely, Évjárat, MegtettKilométer.
* **Karbantartás**: KarbantartásiSzint, MegtettKilométer.
* **Szerelő**: SzerelőID, Név, SSN, Lakcím, Telefonszám.
* **Kölcsönzés**: KölcsönzésID, Időpont, NapokSzáma, Ár.
* **Vásárló**: VásárlóID, Név, Lakcím, Telefonszám.
* **Kölcsönző cég**: CégID, Név, AutókSzáma, Alapterület.

#### **ER modell ábra**:



### XDM modell

Az XDM modell az ER modell hierarchikus átalakításával jött létre. Az átalakítás során az entitások elemekké váltak, míg a kapcsolatok attribútumokon keresztül valósultak meg. A modellezés során a fő elem az <AutoRendszer>, amely tartalmazza a rendszer összes többi komponensét.

#### A diagram of a company Description automatically generated**XDM modell ábra**:

### XML dokumentum

Az XDM modell alapján készült XML dokumentum a gépjárműkölcsönző rendszer példadatait tartalmazza.

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

<AutoRendszer>

<Gepjarmu *JarmuID*="1">

<Rendszam>ABC-123</Rendszam>

<JarmuTipus>Személyautó</JarmuTipus>

<GyartasiHely>Budapest</GyartasiHely>

<Marka>Volkswagen</Marka>

<Evjarat>2015</Evjarat>

<MegtettKilometer>120000</MegtettKilometer>

</Gepjarmu>

<Gepjarmu *JarmuID*="2">

<Rendszam>XYZ-456</Rendszam>

<JarmuTipus>Teherautó</JarmuTipus>

<GyartasiHely>Győr</GyartasiHely>

<Marka>Mercedes</Marka>

<Evjarat>2018</Evjarat>

<MegtettKilometer>90000</MegtettKilometer>

</Gepjarmu>

<Gepjarmu *JarmuID*="3">

<Rendszam>DEF-789</Rendszam>

<JarmuTipus>Kisbusz</JarmuTipus>

<GyartasiHely>Szeged</GyartasiHely>

<Marka>Ford</Marka>

<Evjarat>2020</Evjarat>

<MegtettKilometer>45000</MegtettKilometer>

</Gepjarmu>

<Karbantartas *JarmuID*="1" *SzereloID*="10">

<KarbantartasiSzint>Közepes</KarbantartasiSzint>

<MegtettKilometer>100000</MegtettKilometer>

</Karbantartas>

<Karbantartas *JarmuID*="2" *SzereloID*="11">

<KarbantartasiSzint>Magas</KarbantartasiSzint>

<MegtettKilometer>80000</MegtettKilometer>

</Karbantartas>

<Karbantartas *JarmuID*="3" *SzereloID*="12">

<KarbantartasiSzint>Alacsony</KarbantartasiSzint>

<MegtettKilometer>40000</MegtettKilometer>

</Karbantartas>

<Szerelo *SzereloID*="10">

<Nev>Kovács István</Nev>

<SSN>123456789</SSN>

<Lakcim>Budapest, Fő utca 1.</Lakcim>

<Telefonszam>+36123456789</Telefonszam>

<Mentorok>

<Mentor>20</Mentor>

<Mentoralt>11</Mentoralt>

</Mentorok>

</Szerelo>

<Szerelo *SzereloID*="11">

<Nev>Szabó Péter</Nev>

<SSN>987654321</SSN>

<Lakcim>Győr, Arany János utca 10.</Lakcim>

<Telefonszam>+36981234567</Telefonszam>

</Szerelo>

<Szerelo *SzereloID*="12">

<Nev>Nagy János</Nev>

<SSN>456789123</SSN>

<Lakcim>Szeged, Kossuth Lajos tér 5.</Lakcim>

<Telefonszam>+36201234567</Telefonszam>

</Szerelo>

<Kolcsonzes *KolcsonzesID*="100" *JarmuID*="1" *VasarloID*="200" *KolcsonzoCegID*="300">

<KolcsonzesAra>15000</KolcsonzesAra>

<KolcsonzesIdopontja>2024-01-01</KolcsonzesIdopontja>

<ElteltNapok>5</ElteltNapok>

</Kolcsonzes>

<Kolcsonzes *KolcsonzesID*="101" *JarmuID*="2" *VasarloID*="201" *KolcsonzoCegID*="301">

<KolcsonzesAra>20000</KolcsonzesAra>

<KolcsonzesIdopontja>2024-02-15</KolcsonzesIdopontja>

<ElteltNapok>3</ElteltNapok>

</Kolcsonzes>

<Kolcsonzes *KolcsonzesID*="102" *JarmuID*="3" *VasarloID*="202" *KolcsonzoCegID*="302">

<KolcsonzesAra>18000</KolcsonzesAra>

<KolcsonzesIdopontja>2024-03-10</KolcsonzesIdopontja>

<ElteltNapok>7</ElteltNapok>

</Kolcsonzes>

<Vasarlo *VasarloID*="200">

<Nev>Tóth Anna</Nev>

<Lakcim>Debrecen, Petőfi utca 10.</Lakcim>

<Telefonszam>+36201234567</Telefonszam>

</Vasarlo>

<Vasarlo *VasarloID*="201">

<Nev>Varga Béla</Nev>

<Lakcim>Pécs, Széchenyi tér 12.</Lakcim>

<Telefonszam>+36701234567</Telefonszam>

</Vasarlo>

<Vasarlo *VasarloID*="202">

<Nev>Kiss Éva</Nev>

<Lakcim>Miskolc, Ady Endre utca 20.</Lakcim>

<Telefonszam>+36401234567</Telefonszam>

</Vasarlo>

<KolcsonzoCeg *CegID*="300">

<Nev>AutóRent Kft.</Nev>

<AutokSzama>50</AutokSzama>

<Alapterulet>500m2</Alapterulet>

<Cim>Pécs, Király utca 5.</Cim>

</KolcsonzoCeg>

<KolcsonzoCeg *CegID*="301">

<Nev>SpeedyCar Bt.</Nev>

<AutokSzama>30</AutokSzama>

<Alapterulet>300m2</Alapterulet>

<Cim>Győr, Szabadság utca 15.</Cim>

</KolcsonzoCeg>

<KolcsonzoCeg *CegID*="302">

<Nev>PremiumCar Zrt.</Nev>

<AutokSzama>70</AutokSzama>

<Alapterulet>800m2</Alapterulet>

<Cim>Debrecen, Kossuth utca 20.</Cim>

</KolcsonzoCeg>

</AutoRendszer>

*(Teljes fájl mellékelve: XMLNLFUA8.xml)*

### XMLSchema

Az XSD séma biztosítja az XML dokumentum validációját az alábbi módon:

* Minden entitás attribútumainak típusait meghatározza.
* A kulcsokat (key) és referencia-kulcsokat (keyref) definiálja.
* <?xml *version*="1.0" *encoding*="UTF-8"?>
* <xs:schema *xmlns*:*xs*="http://www.w3.org/2001/XMLSchema" *elementFormDefault*="qualified">
* <xs:element *name*="AutoRendszer">
* <xs:complexType>
* <xs:sequence>
* <xs:element *name*="Gepjarmu" *maxOccurs*="unbounded"/>
* <xs:element *name*="Karbantartas" *maxOccurs*="unbounded"/>
* <xs:element *name*="Szerelo" *maxOccurs*="unbounded"/>
* <xs:element *name*="Kolcsonzes" *maxOccurs*="unbounded"/>
* <xs:element *name*="Vasarlo" *maxOccurs*="unbounded"/>
* <xs:element *name*="KolcsonzoCeg" *maxOccurs*="unbounded"/>
* </xs:sequence>
* </xs:complexType>
* </xs:element>
* <xs:element *name*="Gepjarmu">
* <xs:complexType>
* <xs:sequence>
* <xs:element *name*="Rendszam" *type*="xs:string"/>
* <xs:element *name*="JarmuTipus" *type*="xs:string"/>
* <xs:element *name*="GyartasiHely" *type*="xs:string"/>
* <xs:element *name*="Marka" *type*="xs:string"/>
* <xs:element *name*="Evjarat" *type*="xs:int" *minOccurs*="0"/>
* <xs:element *name*="MegtettKilometer" *type*="xs:int"/>
* </xs:sequence>
* <xs:attribute *name*="JarmuID" *type*="xs:int" *use*="required"/>
* </xs:complexType>
* </xs:element>
* <xs:element *name*="Karbantartas">
* <xs:complexType>
* <xs:sequence>
* <xs:element *name*="KarbantartasiSzint" *type*="xs:string"/>
* <xs:element *name*="MegtettKilometer" *type*="xs:int"/>
* </xs:sequence>
* <xs:attribute *name*="JarmuID" *type*="xs:int" *use*="required"/>
* <xs:attribute *name*="SzereloID" *type*="xs:int" *use*="required"/>
* </xs:complexType>
* </xs:element>
* <xs:element *name*="Szerelo">
* <xs:complexType>
* <xs:sequence>
* <xs:element *name*="Nev" *type*="xs:string"/>
* <xs:element *name*="SSN" *type*="xs:string"/>
* <xs:element *name*="Lakcim" *type*="xs:string"/>
* <xs:element *name*="Telefonszam" *type*="xs:string"/>
* <xs:element *name*="Mentorok" *minOccurs*="0">
* <xs:complexType>
* <xs:sequence>
* <xs:element *name*="Mentor" *type*="xs:int" *maxOccurs*="unbounded"/>
* <xs:element *name*="Mentoralt" *type*="xs:int" *maxOccurs*="unbounded"/>
* </xs:sequence>
* </xs:complexType>
* </xs:element>
* </xs:sequence>
* <xs:attribute *name*="SzereloID" *type*="xs:int" *use*="required"/>
* </xs:complexType>
* </xs:element>
* <!-- *Kölcsönzés* -->
* <xs:element *name*="Kolcsonzes">
* <xs:complexType>
* <xs:sequence>
* <xs:element *name*="KolcsonzesAra" *type*="xs:int"/>
* <xs:element *name*="KolcsonzesIdopontja" *type*="xs:date"/>
* <xs:element *name*="ElteltNapok" *type*="xs:int"/>
* </xs:sequence>
* <xs:attribute *name*="KolcsonzesID" *type*="xs:int" *use*="required"/>
* <xs:attribute *name*="JarmuID" *type*="xs:int" *use*="required"/>
* <xs:attribute *name*="VasarloID" *type*="xs:int" *use*="required"/>
* <xs:attribute *name*="KolcsonzoCegID" *type*="xs:int" *use*="required"/>
* </xs:complexType>
* </xs:element>
* <xs:element *name*="Vasarlo">
* <xs:complexType>
* <xs:sequence>
* <xs:element *name*="Nev" *type*="xs:string"/>
* <xs:element *name*="Lakcim" *type*="xs:string"/>
* <xs:element *name*="Telefonszam" *type*="xs:string"/>
* </xs:sequence>
* <xs:attribute *name*="VasarloID" *type*="xs:int" *use*="required"/>
* </xs:complexType>
* </xs:element>
* <xs:element *name*="KolcsonzoCeg">
* <xs:complexType>
* <xs:sequence>
* <xs:element *name*="Nev" *type*="xs:string"/>
* <xs:element *name*="AutokSzama" *type*="xs:int"/>
* <xs:element *name*="Alapterulet" *type*="xs:string"/>
* <xs:element *name*="Cim" *type*="xs:string"/>
* </xs:sequence>
* <xs:attribute *name*="CegID" *type*="xs:int" *use*="required"/>
* </xs:complexType>
* </xs:element>
* </xs:schema>

*(Teljes fájl mellékelve: XMLSchemaNLFUA8.xsd)*

### Második feladat: DOM API programozás

A második feladat során a DOM API-t használva négy különböző programot valósítottam meg, amelyek az XML dokumentum feldolgozásának különböző aspektusait fedik le. A feladatok: adatolvasás, adatírás, adatlekérdezés és adatmódosítás. Minden program a megfelelő Java osztályban került implementálásra, a specifikációk szerint.

### 2.1 Adatolvasás

A DOMReadNeptunkod.java osztály célja, hogy az XML dokumentumot beolvassa, és annak tartalmát hierarchikus fa struktúrában a konzolra írja. A program a DOM API segítségével dolgozza fel az XML elemeit és attribútumait.

#### **Fő lépések:**

1. Az XML dokumentum betöltése a DocumentBuilder segítségével.
2. A gyökérelem és alárendelt elemek rekurzív feldolgozása.
3. Az elemek attribútumainak és értékeinek kiírása blokk formában.
4. A dokumentum tartalmának mentése egy új XML fájlba.

#### **Fontos kód:**

// *Root elem kiírása*

System.*out*.*println*("Root element: " + doc.*getDocumentElement*().*getNodeName*());

// *Összes gépjármű kiíratása*

NodeList nodeList = doc.*getElementsByTagName*("Gepjarmu");

*for* (int i = 0; i < nodeList.*getLength*(); i++) {

Node node = nodeList.*item*(i);

*if* (node.*getNodeType*() == Node.*ELEMENT\_NODE*) {

Element element = (Element) node;

System.*out*.*println*("Jármű ID: " + element.*getAttribute*("JarmuID"));

System.*out*.*println*("Rendszám: " + element.*getElementsByTagName*("Rendszam").*item*(0).*getTextContent*());

System.*out*.*println*("Típus: " + element.*getElementsByTagName*("JarmuTipus").*item*(0).*getTextContent*());

System.*out*.*println*("Márka: " + element.*getElementsByTagName*("Marka").*item*(0).*getTextContent*());

System.*out*.*println*("Kilométer: " + element.*getElementsByTagName*("MegtettKilometer").*item*(0).*getTextContent*());

System.*out*.*println*();

}

}

#### **Kimenet a konzolon:**

Root element: AutoRendszer

Jármű ID: 1

Rendszám: ABC-123

Típus: Személyautó

Márka: Volkswagen

Kilométer: 120000

Jármű ID: 2

Rendszám: XYZ-456

Típus: Teherautó

Márka: Mercedes

Kilométer: 90000

Jármű ID: 3

Rendszám: DEF-789

Típus: Kisbusz

Márka: Ford

Kilométer: 45000

Jármű ID: 1

Rendszám: ABC-123

Jármű ID: 2

Rendszám: XYZ-456

Jármű ID: 3

Rendszám: DEF-789

*(A teljes kód mellékelve: DOMReadNeptunkod.java)*

### 2.2 Adatírás

A DOMWriteNeptunkod.java osztály egy új XML dokumentumot generál a DOM API segítségével, és azt fájlba menti. A program lehetővé teszi új elemek és attribútumok hozzáadását a meglévő hierarchiához.

#### **Fő lépések:**

1. Egy új DOM dokumentum létrehozása.
2. A gyökérelem (AutoRendszer) létrehozása és hozzáadása.
3. Új jármű adatok (Gepjarmu) hozzáadása, attribútumokkal és alárendelt elemekkel.
4. Az eredmény mentése egy XML fájlba.

#### **Fontos kód:**

// *XML mentése fájlba*

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transformer = transformerFactory.*newTransformer*();

DOMSource source = *new* *DOMSource*(doc);

StreamResult result = *new* *StreamResult*(*new* *File*("XMLNLFUA8\_Updated.xml"));

transformer.*transform*(source, result);

#### **Kimenet fájlban:**

<?xml *version*="1.0" *encoding*="UTF-8" *standalone*="no"?>

<AutoRendszer>

<Gepjarmu *JarmuID*="4">

<Rendszam>GHI-321</Rendszam>

<JarmuTipus>SUV</JarmuTipus>

</Gepjarmu>

</AutoRendszer>

*(A teljes kód mellékelve: DOMWriteNeptunkod.java)*

### 2.3 Adatlekérdezés

A DOMQueryNeptunkod.java osztály legalább négy különböző lekérdezést valósít meg az XML dokumentum adataiból. A lekérdezések nem használják az XPath kifejezéseket, hanem a DOM API elemeire épülnek.

#### **Lekérdezések:**

1. Összes jármű márkája
2. Járművek tipusai
3. Összes karbantartási rekord listázása.
4. Melyik városban készültek a járművek
5. Legtöbb km-t megtett jármű adatai

#### **Fontos kód:**

// *5. Lekérdezés: Legtöbbet megtett kilométerű jármű adatai*

System.*out*.*println*("5. Lekérdezés: Legtöbbet megtett kilométerű jármű adatai");

int maxKilometer = 0;

Element maxKilometerCar = null;

*for* (int i = 0; i < nodeList.*getLength*(); i++) {

Element element = (Element) nodeList.*item*(i);

int kilometer = Integer.*parseInt*(element.*getElementsByTagName*("MegtettKilometer").*item*(0).*getTextContent*());

*if* (kilometer > maxKilometer) {

maxKilometer = kilometer;

maxKilometerCar = element;

}

}

*if* (maxKilometerCar != null) {

System.*out*.*println*("- Rendszám: " + maxKilometerCar.*getElementsByTagName*("Rendszam").*item*(0).*getTextContent*());

System.*out*.*println*("- Típus: " + maxKilometerCar.*getElementsByTagName*("JarmuTipus").*item*(0).*getTextContent*());

System.*out*.*println*("- Kilométer: " + maxKilometer);

}

#### **Kimenet a konzolon:**

1. Lekérdezés: Összes jármű márkája

- Márka: Volkswagen

- Márka: Mercedes

- Márka: Ford

2. Lekérdezés: Melyik járművek típusai

- Típus: Személyautó

- Típus: Teherautó

- Típus: Kisbusz

3. Lekérdezés: Karbantartás szintjei járművenként

- Jármű ID: 1

Karbantartási szint: Közepes

- Jármű ID: 2

Karbantartási szint: Magas

- Jármű ID: 3

Karbantartási szint: Alacsony

4. Lekérdezés: Melyik városokban készültek a járművek

- Gyártási hely: Budapest

- Gyártási hely: Győr

- Gyártási hely: Szeged

5. Lekérdezés: Legtöbbet megtett kilométerű jármű adatai

- Rendszám: ABC-123

- Típus: Személyautó

- Kilométer: 120000 *(A teljes kód mellékelve: DOMQueryNeptunkod.java)*

### 2.4 Adatmódosítás

A DOMModifyNeptunkod.java osztály az XML dokumentum meglévő elemeit módosítja. Legalább négy különböző módosítást valósít meg.

#### **Módosítások:**

1. Egy jármű rendszámának frissítése
2. Második jármű megtett kilométerének növelése
3. Egy új jármű hozzáadása.
4. Az első jármű gyártási helyének megváltoztatásaEgy szerelő nevének frissítése.

#### **Fontos kód:**

// *1. Módosítás: Egy jármű rendszámának frissítése*

NodeList nodes = doc.*getElementsByTagName*("Gepjarmu");

Element car = (Element) nodes.*item*(0);

car.*getElementsByTagName*("Rendszam").*item*(0).*setTextContent*("ZZZ-999");

#### **Kimenet fájlban (részlet):**

<?xml *version*="1.0" *encoding*="UTF-8" *standalone*="no"?>

<AutoRendszer>

<Gepjarmu *JarmuID*="1">

<Rendszam>ZZZ-999</Rendszam>

<JarmuTipus>Személyautó</JarmuTipus>

<GyartasiHely>Budapest</GyartasiHely>

<Marka>Volkswagen</Marka>

<Evjarat>2015</Evjarat>

<MegtettKilometer>120000</MegtettKilometer>

</Gepjarmu>

</AutoRendszer>

*(A teljes kód mellékelve: DOMModifyNeptunkod.java)*

## Következtetés

A feladat megoldása során alaposan megismerhettem az XML szabvány és a DOM API lehetőségeivel. A projekt során az alábbiakat tanultam:

* Hogyan lehet egy relációs adatbázist XML formátumba alakítani (ER és XDM modellek).
* Hogyan lehet a DOM API segítségével az XML dokumentumot olvasni, írni, lekérdezni és módosítani.
* Az XSD séma validációjának fontosságát és alkalmazását.

A feladat kiváló alapot adott az XML és a kapcsolódó technológiák mélyebb megértéséhez, valamint a DOM API hatékony használatához.

**Mellékletek:**

* ER modell ábra: ERNLFUA8.png
* XDM modell ábra: XDMNLFUA8.png
* XML dokumentum: XMLNLFUA8.xml
* XML séma: XMLSchemaNLFUA8.xsd
* Java kódok: DOMReadNeptunkod.java, DOMWriteNeptunkod.java, DOMQueryNeptunkod.java, DOMModifyNeptunkod.java