JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Étterem felépítése

Készítette: Kola Sándor

Neptunkód: B8GU52

Dátum: 2024.12.05

Tartalom

[1. Feladat leírása 3](#_Toc153142244)

[2. Az ER modell konvertálása XDM modellé 4](#_Toc153142245)

[3. XML dokumentum készítése 4](#_Toc153142246)

[4. XMLSchema készítése 7](#_Toc153142247)

[5. DOM adatolvasás 12](#_Toc153142248)

[6. DOM adatmódosítás 14](#_Toc153142249)

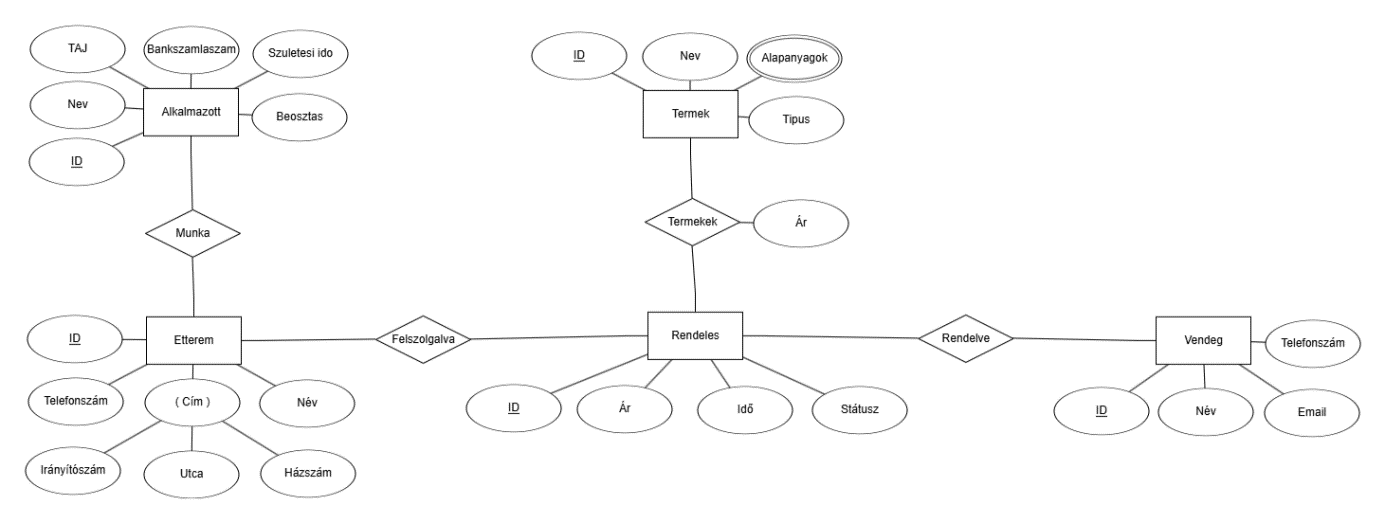
[7. DOM adatlekérdezés 16](#_Toc153142250)

[8. DOM adatírás 18](#_Toc153142251)

# Feladat leírása

A beadandó témája egy olyan adatbázis, amely egy étterem, és annak dolgozóinak illetve rendeléseinek és egyéb hozzá tartozó adatait tartja számon. Lehetőség van egyes termékek, illetve vendégek, vagy akár az adott helyszín adatainak lekérdezésére.

* Étterem egyed tulajdonságai
  + id: a kulcs
  + Telefonszám: az étterem telefonszáma
  + Név: az étterem neve
  + Cím: összetett tulajdonság, az étterem címe
* Alkalmazott egyed tulajdonságai
  + id: a dolgozó egyedi kulcsa
  + Név: a dolgozó teljes neve
  + TAJ: a dolgozó orvosi száma
  + Bankszámlaszám: a dolgozó számlaszáma
  + Születési idő: a dolgozó születési dátuma
  + Beosztás: vezető beosztásban van-e
* Rendelés egyed tulajdonságai
  + id: az rendelés egyed kulcsa
  + státusz: aktív e még a rendelés
  + Idő: rendelés időpontja
  + Ár: a rendelés végösszege
* Vendég egyed tulajdonságai
  + id: a vásárló egyedi azonosítója
  + Név: a vásárló teljes neve
  + email: a vásárló email címe
  + Telefonszám: a vásárló telefonszáma
* Termék egyed tulajdonságai
  + id: termék kulcsa
  + Név: a termék neve
  + Alapanyagok: a termék készítéséhez felhasznált alapanyagok, mely lehet többértékű
  + típus: a termék típusa

A feladat ER modellje:

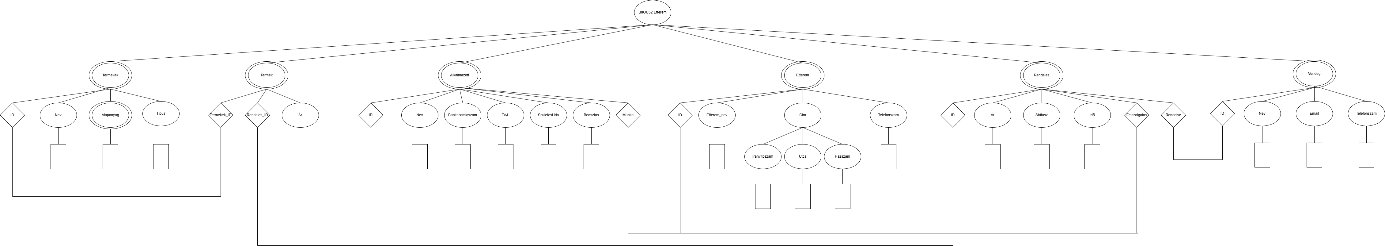
Az egyedek közötti kapcsolatok:

* Alkalmazott és Étterem közötti kapcsolat: Munka
  + Egy-több kapcsolat van közöttük, mivel egy alkalmazott csak egy étteremnél dolgozhat.
* Étterem és Rendelés közötti kapcsolat: Felszolgálva
  + Egy-több kapcsolat van közöttük, mivel egy étteremhez tartozhat több rendelés is, de egy adott rendelés nem tartozhat több étteremhez.
* Rendelés és Termék közötti kapcsolat: Termékek
  + Több-több kapcsolat van közöttük, mivel egy termék tartozhat több rendeléshez is, illetve egy rendeléshez is tartozhat több termék.
* Rendelés és Vendég közötti kapcsolat: Rendelve
  + Egy-egy kapcsolat van közöttük, mivel csak egy rendelés lehet aktív egy vevőnek, és egy rendelés csak egy vevőhöz tartozhat.

# Az ER modell konvertálása XDM modellé

Az XDM modell elkészítése során háromféle jelölést használtam: ellipszist, rombuszt és téglalapot. A modellben minden egyed ellipszis formában jelenik meg, és mivel többértékűek, ezeket duplán kell jelölni. A kulcsfontosságú tulajdonságokat rombuszokkal, míg a szövegeket téglalapokkal ábrázoltam.

Az XDM modell a következőképpen épül fel:



# XML dokumentum készítése

A következő lépés az XML dokumentum elkészítése volt az XDM modell alapján. Az összes dupla ellipszissel jelölt elemhez legalább három példányt hoztam létre. A kulcselemek gyerekelemek attribútumaként is megjelennek, valamint az idegen kulcsok szintén beépítésre kerültek.

Kód:

<B8GU52\_Étterem xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="XMLSchema.xsd">

<!-- Étterem példányok -->

<Étterem id="E001">

<Név>Gourmet Bistro</Név>

<Telefonszám>06301234567</Telefonszám>

<Cím>

<Irányítószám>12345</Irányítószám>

<Utca>Elm Avenue</Utca>

<Házszám>34</Házszám>

</Cím>

</Étterem>

<Étterem id="E002">

<Név>Pepper's Grill</Név>

<Telefonszám>06309876543</Telefonszám>

<Cím>

<Irányítószám>54321</Irányítószám>

<Utca>Oak Street</Utca>

<Házszám>12</Házszám>

</Cím>

</Étterem>

<Étterem id="E003">

<Név>Sunrise Café</Név>

<Telefonszám>06701239876</Telefonszám>

<Cím>

<Irányítószám>11223</Irányítószám>

<Utca>Willow Lane</Utca>

<Házszám>89</Házszám>

</Cím>

</Étterem>

<!-- Alkalmazott példányok -->

<Alkalmazott id="A001" Munka="E001">

<Név>Emily Watson</Név>

<TAJ>M789</TAJ>

<Bankszámlaszám>DEF987654</Bankszámlaszám>

<SzületésiIdő>1985-03-10</SzületésiIdő>

<Beosztás>Nem</Beosztás>

</Alkalmazott>

<Alkalmazott id="A002" Munka="E002">

<Név>Michael Brown</Név>

<TAJ>M456</TAJ>

<Bankszámlaszám>DEF123456</Bankszámlaszám>

<SzületésiIdő>1992-11-20</SzületésiIdő>

<Beosztás>Igen</Beosztás>

</Alkalmazott>

<Alkalmazott id="A003" Munka="E003">

<Név>Sarah Johnson</Név>

<TAJ>M789</TAJ>

<Bankszámlaszám>DEF678901</Bankszámlaszám>

<SzületésiIdő>2000-06-15</SzületésiIdő>

<Beosztás>Nem</Beosztás>

</Alkalmazott>

<!-- Rendelés példányok -->

<Rendelés id="R001" Felszolgálva="E001" Rendelve="V001">

<Ár>3000</Ár>

<Idő>2024.01.01</Idő>

<Státusz>Aktív</Státusz>

</Rendelés>

<Rendelés id="R002" Felszolgálva="E002" Rendelve="V002">

<Ár>12000</Ár>

<Idő>2024.02.20</Idő>

<Státusz>Kiszállítva</Státusz>

</Rendelés>

<Rendelés id="R003" Felszolgálva="E003" Rendelve="V003">

<Ár>5800</Ár>

<Idő>2024.03.15</Idő>

<Státusz>Kiszállítva</Státusz>

</Rendelés>

<!-- Termék példányok -->

<Termék id="T001">

<Név>Deluxe Pizza</Név>

<Típus>Főétel</Típus>

<Alapanyag>pepperoni</Alapanyag>

<Alapanyag>mozzarella</Alapanyag>

<Alapanyag>olívabogyó</Alapanyag>

</Termék>

<Termék id="T002">

<Név>Veggie Delight</Név>

<Típus>Előétel</Típus>

<Alapanyag>spenót</Alapanyag>

<Alapanyag>paradicsom</Alapanyag>

</Termék>

<Termék id="T003">

<Név>Tropical Salad</Név>

<Típus>Köret</Típus>

<Alapanyag>saláta</Alapanyag>

<Alapanyag>mangó</Alapanyag>

<Alapanyag>uborka</Alapanyag>

</Termék>

<!-- Vendég példányok -->

<Vendég id="V001">

<Név>Laura Smith</Név>

<Email>laura@example.com</Email>

<Telefonszám>06305556677</Telefonszám>

</Vendég>

<Vendég id="V002">

<Név>Daniel Carter</Név>

<Email>daniel@example.com</Email>

<Telefonszám>06701233445</Telefonszám>

</Vendég>

<Vendég id="V003">

<Név>Sophia White</Név>

<Email>sophia@example.com</Email>

<Telefonszám>06307778899</Telefonszám>

</Vendég>

<!-- Melyik termék -->

<Termékek termék\_id="T001" rendelés\_id="R001">

<Ár>3200</Ár>

</Termékek>

<Termékek termék\_id="T002" rendelés\_id="R002">

<Ár>2800</Ár>

</Termékek>

<Termékek termék\_id="T003" rendelés\_id="R003">

<Ár>4500</Ár>

</Termékek>

</B8GU52\_Étterem>

# XMLSchema készítése

Ezután elkészítettem az XML Schema-t az XML dokumentum validációjához. Először definiáltam az egyszerű típusokat, amelyeket később hivatkozásokhoz használhattam. Ezt követték az egyedi típusok, amelyek meghatározásához reguláris kifejezéseket és felsorolásokat (enumeration) is alkalmaztam. A következő lépés a séma tényleges struktúrájának felépítése volt. A root elem szülőelemeit complexType segítségével definiáltam, és ezen belül a gyerekelemeknél az előzőleg létrehozott egyszerű típusokra hivatkoztam. Végül meghatároztam a kulcsokat, idegen kulcsokat és az egyediségre (unique) vonatkozó feltételeket is.

Az XMLSchema kód

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<!-- Egyszerű elemek -->

<xs:element name="Név" type="xs:string" />

<xs:element name="Telefonszám" type="phoneType" />

<xs:element name="Irányítószám" type="xs:int" />

<xs:element name="Utca" type="xs:string" />

<xs:element name="Házszám" type="xs:int" />

<xs:element name="TeljesNév" type="xs:string" />

<xs:element name="TAJ" type="xs:string" />

<xs:element name="Bankszámlaszám" type="xs:string" />

<xs:element name="SzületésiIdő" type="timeType" />

<xs:element name="Beosztás" type="BeosztásTípus" />

<xs:element name="Ár" type="xs:int" />

<xs:element name="Idő" type="timeType" />

<xs:element name="Státusz" type="StátuszTípus" />

<xs:element name="Típus" type="xs:string" />

<xs:element name="Alapanyag" type="xs:string" />

<xs:element name="Email" type="emailType" />

<!-- Saját típusok -->

<xs:simpleType name="BeosztásTípus">

<xs:restriction base="xs:string">

<xs:enumeration value="Igen" />

<xs:enumeration value="Nem" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="StátuszTípus">

<xs:restriction base="xs:string">

<xs:enumeration value="Aktív" />

<xs:enumeration value="Kiszállítva" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="phoneType">

<xs:restriction base="xs:string">

<xs:pattern value="(06(20|30|31|50|60|70)\d{7})"/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="timeType">

<xs:restriction base="xs:string">

<xs:pattern value="([12]\d{3}.(0[1-9]|1[0-2]).(0[1-9]|[12]\d|3[01]))" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="emailType">

<xs:restriction base="xs:string">

<xs:pattern value="[\w\.]+@([\w]+\.)+[\w]{2,4}" />

</xs:restriction>

</xs:simpleType>

<!-- Felépítés -->

<xs:element name="Éttermek">

<xs:complexType>

<xs:sequence>

<xs:element name="Étterem" minOccurs="1" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element ref="Név" />

<xs:element ref="Telefonszám" />

<xs:element name="Cím" minOccurs="1" maxOccurs="1">

<xs:complexType>

<xs:sequence>

<xs:element ref="Irányítószám" />

<xs:element ref="Utca" />

<xs:element ref="Házszám" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

<xs:attribute name="id" type="xs:string" />

</xs:complexType>

</xs:element>

<xs:element name="Alkalmazott" minOccurs="1" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element ref="TeljesNév" />

<xs:element ref="TAJ" />

<xs:element ref="Bankszámlaszám" />

<xs:element ref="SzületésiIdő" />

<xs:element ref="Beosztás" />

</xs:sequence>

<xs:attribute name="id" type="xs:string" />

<xs:attribute name="Munka" type="xs:string" />

</xs:complexType>

</xs:element>

<xs:element name="Rendelés" minOccurs="1" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element ref="Ár" />

<xs:element ref="Idő" />

<xs:element ref="Státusz" />

</xs:sequence>

<xs:attribute name="id" type="xs:string" />

<xs:attribute name="Felszolgálva" type="xs:string" />

<xs:attribute name="Rendelve" type="xs:string" />

</xs:complexType>

</xs:element>

<xs:element name="Termék" minOccurs="1" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element ref="Név" />

<xs:element ref="Típus" />

<xs:element ref="Alapanyag" minOccurs="1" maxOccurs="unbounded" />

</xs:sequence>

<xs:attribute name="id" type="xs:string" />

</xs:complexType>

</xs:element>

<xs:element name="Vendég" minOccurs="1" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element ref="TeljesNév" />

<xs:element ref="Email" />

<xs:element ref="Telefonszám" />

</xs:sequence>

<xs:attribute name="id" type="xs:string" />

</xs:complexType>

</xs:element>

<xs:element name="Termékek" minOccurs="1" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element ref="Ár" />

</xs:sequence>

<xs:attribute name="termék\_id" type="xs:string" />

<xs:attribute name="rendelés\_id" type="xs:string" />

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- Kulcsok -->

<xs:key name="étterem\_kulcs">

<xs:selector xpath="Étterem" />

<xs:field xpath="@id" />

</xs:key>

<xs:key name="alkalmazott\_kulcs">

<xs:selector xpath="Alkalmazott" />

<xs:field xpath="@id" />

</xs:key>

<xs:key name="rendelés\_kulcs">

<xs:selector xpath="Rendelés" />

<xs:field xpath="@id" />

</xs:key>

<xs:key name="termék\_kulcs">

<xs:selector xpath="Termék" />

<xs:field xpath="@id" />

</xs:key>

<xs:key name="vendég\_kulcs">

<xs:selector xpath="Vendég" />

<xs:field xpath="@id" />

</xs:key>

<!-- Idegen kulcsok -->

<xs:keyref refer="étterem\_kulcs" name="alkalmazott\_idegen\_kulcs">

<xs:selector xpath="Alkalmazott" />

<xs:field xpath="@Munka" />

</xs:keyref>

<xs:keyref refer="étterem\_kulcs" name="rendelés\_étterem\_idegen\_kulcs">

<xs:selector xpath="Rendelés" />

<xs:field xpath="@Felszolgálva" />

</xs:keyref>

<xs:keyref refer="vendég\_kulcs" name="rendelés\_vendég\_idegen\_kulcs">

<xs:selector xpath="Rendelés" />

<xs:field xpath="@Rendelve" />

</xs:keyref>

<xs:keyref refer="termék\_kulcs" name="termékek\_termék\_idegen\_kulcs">

<xs:selector xpath="Termékek" />

<xs:field xpath="@termék\_id" />

</xs:keyref>

<xs:keyref refer="rendelés\_kulcs" name="termékek\_rendelés\_idegen\_kulcs">

<xs:selector xpath="Termékek" />

<xs:field xpath="@rendelés\_id" />

</xs:keyref>

</xs:schema>

# DOM adatolvasás

Először meghatároztam, hogy melyik fájlból kell elvégezni a beolvasást. Ehhez létrehoztam a szükséges változókat és egy metódust, amely segít kiírni a tartalmat a konzolra. A metódus először ellenőrzi, hogy az adott Node rendelkezik-e attribútumokkal, és ha igen, akkor kiírja azokat. Ezen kívül megnézi, hogy az adott elemnek vannak-e gyerekelemek. Ha igen, akkor rekurzívan feldolgozza és kiírja őket is.

A kód a következőképpen néz ki:

package DomParseB8GU52;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import java.io.File;

public class DOMReadB8GU52 {

public static void main(String[] args) {

try {

File xmlFile = new File("XMLB8GU52.xml");

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

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

printElementContent(doc.getDocumentElement(), "");

} catch (Exception e) {

e.printStackTrace();

}

}

public static void printElementContent(Node node, String indent) {

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

// Kiírja az aktuális elem nevét

System.out.println(indent + "Element: " + node.getNodeName());

// Kiírja az attribútumokat (ha vannak)

if (node.hasAttributes()) {

NamedNodeMap attributes = node.getAttributes();

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

Node attribute = attributes.item(i);

System.out.println(indent + " Attribute - " + attribute.getNodeName() + ": " + attribute.getNodeValue());

}

}

// Gyermekelemek feldolgozása

NodeList children = node.getChildNodes();

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

Node child = children.item(i);

if (child.getNodeType() == Node.ELEMENT\_NODE) {

printElementContent(child, indent + " ");

} else if (child.getNodeType() == Node.TEXT\_NODE) {

String textContent = child.getTextContent().trim();

if (!textContent.isEmpty()) {

System.out.println(indent + " Text: " + textContent);

}

}

}

}

}

}

# DOM adatmódosítás

A dinamikus adatmódosítást úgy valósítottam meg, hogy a felhasználótól kértem be a módosítandó elem nevét, az elem attribútumait és kulcsát, valamint a módosítani kívánt gyerekelemet és annak új értékét. A felhasználó által megadott információk alapján a program módosítja az XML fájl tartalmát, és elmenti a változtatásokat.

Kód:

package DomParseB8GU52;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import javax.xml.transform.\*;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

import java.util.Scanner;

public class DOMModifyB8GU52 {

public static void main(String[] args) {

try {

File xmlFile = new File("XMLB8GU52.xml");

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

Scanner sc = new Scanner(System.in);

System.out.println("Enter the name of the element to modify: ");

String elementName = sc.nextLine();

System.out.println("Enter the ID of the element: ");

String elementID = sc.nextLine();

System.out.println("Enter the name of the property to modify: ");

String propertyName = sc.nextLine();

System.out.println("Enter the new value: ");

String newValue = sc.nextLine();

sc.close();

if (modifyElementByID(doc, elementName, elementID, propertyName, newValue)) {

saveToFile(doc, "Modified\_XML\_B8GU52.xml");

System.out.println("Modification successful. Saved to Modified\_XML\_B8GU52.xml");

} else {

System.out.println("Modification failed.");

}

} catch (Exception e) {

e.printStackTrace();

}

}

public static boolean modifyElementByID(Document doc, String elementName, String elementID, String propertyName, String newValue) {

try {

NodeList nodeList = doc.getElementsByTagName(elementName);

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

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

if (element.getAttribute("id").equals(elementID)) {

element.getElementsByTagName(propertyName).item(0).setTextContent(newValue);

return true;

}

}

} catch (Exception e) {

e.printStackTrace();

}

return false;

}

public static void saveToFile(Document doc, String filename) {

try {

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filename));

transformer.transform(source, result);

} catch (Exception e) {

e.printStackTrace();

}

}

}

# DOM adatlekérdezés

A lekérdezést dinamikussá tettem, hasonlóan az adatmódosításhoz, úgy hogy a felhasználó megadja, melyik elem adatait szeretné lekérdezni, és az elemhez tartozó ID-t. Az XPath segítségével hajtottam végre a lekérdezést, amely lehetővé teszi különböző típusú lekérdezések végrehajtását a megfelelő XPath kifejezés módosításával.

A kód a következőképpen néz ki:

package DomParseB8GU52;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import javax.xml.xpath.\*;

import java.io.File;

import java.util.Scanner;

public class DOMQueryB8GU52 {

public static void main(String[] args) {

try {

File xmlFile = new File("XMLB8GU52.xml");

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

Scanner sc = new Scanner(System.in);

System.out.println("Enter the name of the element: ");

String elementName = sc.nextLine();

System.out.println("Enter the ID of the element: ");

String elementID = sc.nextLine();

sc.close();

queryElementByID(doc, elementName, elementID);

} catch (Exception e) {

e.printStackTrace();

}

}

public static void queryElementByID(Document doc, String elementName, String elementID) {

try {

XPathFactory xPathfactory = XPathFactory.newInstance();

XPath xpath = xPathfactory.newXPath();

String xpathQuery = "//" + elementName + "[@id='" + elementID + "']";

XPathExpression expr = xpath.compile(xpathQuery);

Node node = (Node) expr.evaluate(doc, XPathConstants.NODE);

if (node != null) {

System.out.println("Found element: " + node.getNodeName());

NodeList children = node.getChildNodes();

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

Node child = children.item(i);

if (child.getNodeType() == Node.ELEMENT\_NODE) {

System.out.println(child.getNodeName() + ": " + child.getTextContent());

}

}

} else {

System.out.println("Element not found.");

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

# DOM adatírás

A kód beolvassa az XML fájlt, végigiterál rajta, és kiírja annak elemeit, attribútumait és gyermekeit a konzolra. Ezután a módosított dokumentumot egy új fájlba menti a Transformer és a StreamResult segítségével. A writeNode metódus a dokumentum elemeit dolgozza fel, míg a saveToFile metódus az új fájlba történő mentést végzi.

A kód a következőképpen néz ki:

package DomParseB8GU52;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import javax.xml.transform.\*;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

public class DOMWriteB8GU52 {

public static void main(String[] args) {

try {

File xmlFile = new File("XMLB8GU52.xml");

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

System.out.println("Writing XML tree structure:");

writeNode(doc.getDocumentElement(), "");

saveToFile(doc, "Output\_XML\_B8GU52.xml");

System.out.println("Saved to Output\_XML\_B8GU52.xml");

} catch (Exception e) {

e.printStackTrace();

}

}

public static void writeNode(Node node, String indent) {

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

System.out.println(indent + node.getNodeName());

if (node.hasAttributes()) {

NamedNodeMap attributes = node.getAttributes();

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

Node attribute = attributes.item(i);

System.out.println(indent + " - " + attribute.getNodeName() + ": " + attribute.getNodeValue());

}

}

NodeList children = node.getChildNodes();

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

writeNode(children.item(i), indent + " ");

}

}

}

public static void saveToFile(Document doc, String filename) {

try {

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File(filename));

transformer.transform(source, result);

} catch (Exception e) {

e.printStackTrace();

}

}

}