JEGYZŐKÖNYV

Adatkezelés XML környezetben
féléves feladat

Online vásárlás

Készítette: **Salamon István**

Neptunkód: **FA6VDV**

Dátum: 2023. 12. 12.

Tartalomjegyzék

Bevezetés

Az online vásárlásokat ábrázoló ER modell egy olyan struktúrát ír le, amely segítségével könnyen megérthetők és modellezhetők az online kiskereskedelmi rendszerek működésének alapvető elemei. Az ER modell az entitások közötti kapcsolatokat és az entitásokat magukat ábrázolja, amelyek kulcsfontosságúak a rendszer felépítése és adatkezelése szempontjából.

1. Feladat Leírása:

Ez a modell célja, hogy részletesen bemutassa az online vásárlás folyamatában keletkező adatok struktúráját és kapcsolatait. Az entitások, mint például a felhasználók, rendelések, termékek és fizetések közötti összefüggések és attribútumok szemléltetése kiemeli az adatok összetettségét.

Az ER modell segítségével mutatom be a fő entitások közötti kapcsolatokat, hozzájárulva a rendszer összképének kialakításához. Az XML alkalmazásával a létrehozott adatok egyszerűen hozzáférhetővé válnak, lehetőséget teremtve azok hatékony és strukturált kezelésére.

Az XML séma gondoskodik arról, hogy az adatok struktúrája megfeleljen a megbízhatóság és biztonság szempontjából elvárható követelményeknek. Ezáltal garantálva van, hogy az adatok konzisztensek és érvényesek maradnak a feldolgozás során.

A létrehozott XML dokumentumot egy Java programon keresztül mutatom be, ahol az olvasás, módosítás, lekérdezések és új XML dokumentumok létrehozása is részletesen bemutatásra kerül. Ez a gyakorlatias megközelítés segít a Java programozási nyelv használatának megértésében a valós adatokkal történő interakció során

1.1 Az adatbázis ER modell tervezése

Az ER modellben található fő entitások:

OnlineVasarlas: A fő entitás, amely magában foglalja az összes kapcsolódó információt az online vásárlásokkal kapcsolatban. A rendelések, felhasználók, profilok, fizetések, termékek és vásárlások összekapcsolódnak ezen a szinten.

Rendeles: Az online vásárlásokat reprezentáló entitás, amely tartalmazza a rendelés dátumát, a szállítási címet, valamint az összeget. Ez kapcsolatban áll a Felhasznalo, Profil, Fizetes, Termek és Vasarlas entitásokkal.

Felhasznalo: Az online vásárlókat reprezentáló entitás, amely tartalmazza a felhasználónevüket, e-mail címüket és regisztrációs dátumukat.

Profil : A felhasználók profiljait reprezentáló entitás, amely tartalmazza a felhasználó nevét, telefonszámát és utolsó belépési dátumát.

Fizetes: Az online fizetéseket reprezentáló entitás, amely tartalmazza az összeget, a tranzakció dátumát és a fizetési módot.

Termek: Az eladó termékeket reprezentáló entitás, amely tartalmazza a termék nevét, árát és készletét.

Vasarlas: Az online vásárlásokat összekapcsoló kapcsolat entitás, amely tartalmazza, hogy egy vásárlás személyes vagy online volt.

Az ER modellben található elsődleges kulcsok:

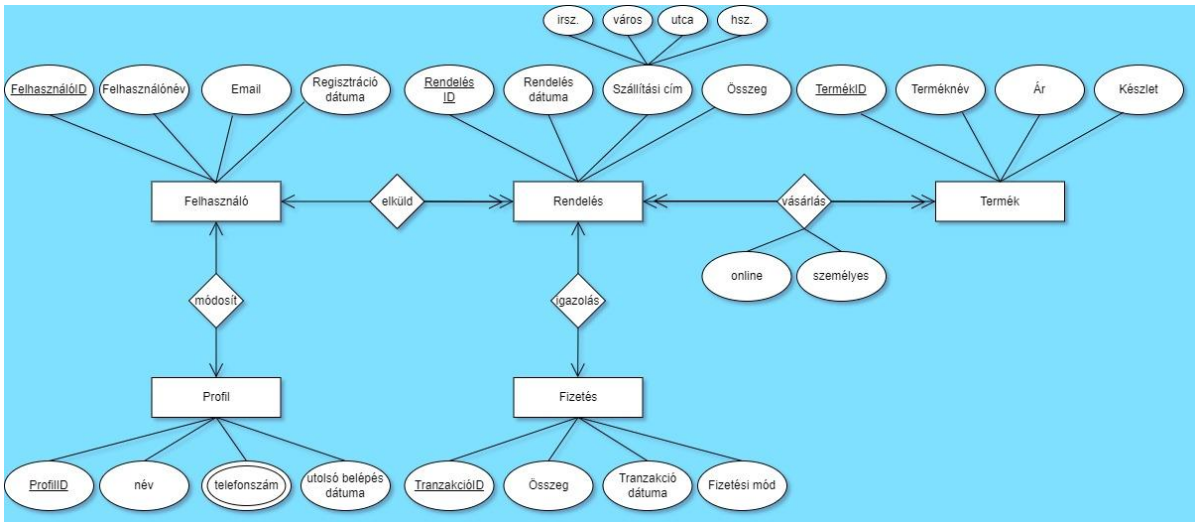
- rendeles_kulcs:
  - rendelesid
- felhasznalo_kulcs:
  - felhasznaloid
- profil_kulcs:
  - profilid
- fizetes_kulcs:
  - fizetesid
- termek_kulcs:
  - termekid

Idegen Kulcsok:

Rendelés és Felhasználó: R-V-K kapcsolat

Termék és Rendelés: T-V-K kapcsolat

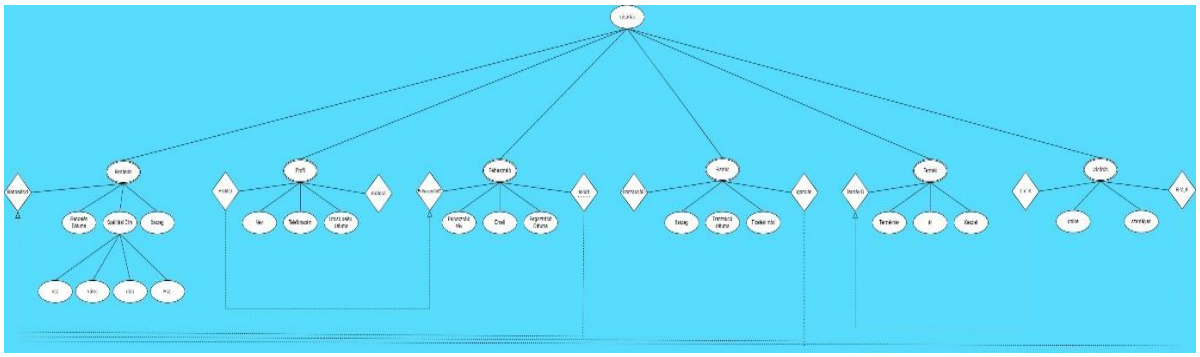
ER modell:



1.2 Az adatbázis konvertálása XDM modellre

Az adatbázis konvertálása XDM (XML Data Model) modellre egy folyamat, amelynek célja az adatok strukturált és szemantikus XML formátumban történő reprezentációja. Az XDM modell lehetővé teszi az adatok hierarchikus és rugalmas tárolását, amely kiválóan alkalmas az XML-alapú adatcserére és integrációra. Az XDM modellben az entitások adatainak strukturált tárolására szolgálnak. Az ER modellben lévő táblák az XDM modellben entitásokká válnak, és a táblák oszlopai az entitások tulajdonságait jelentik. Az entitások közötti kapcsolatokat a szaggatott vonalak segítségével kötjük össze az modellben.

XDM modell:



1.3 Az XDM modell alapján XML dokumentum készítése:

Az XDM (XML Data Model) modell alapján XML dokumentum készítése során az adatok struktúráját és hierarchiáját az XDM modell definíciója alapján kell követni.

XML dokumentum kódrész:

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

<onlinevasarlas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="xdmschema_FA6vdv.xsd">

    <rendeles rendelesid="R1" R-V_K="RVK1">
        <rendeles_datuma>2023-01-01</rendeles_datuma>
        <szallitasi_cim>
            <irsz>1328</irsz>
            <varos>Budapest</varos>
            <utca>Kossuth utca</utca>
            <hsz>12</hsz>
        </szallitasi_cim>
        <osszeg>14990</osszeg>

    </rendeles>

    <rendeles rendelesid="R2" R-V_K="RVK2">
```

```
    <rendeles_datuma>2023-01-01</rendeles_datuma>
    <szallitasi_cim>
        <irsz>3300</irsz>
        <varos>Eger</varos>
        <utca>Dobó </utca>
        <hsz>56</hsz>
    </szallitasi_cim>
    <osszeg>8990</osszeg>
</rendeles>


<felhasznalo felhasznaloid="F1">
    <felhasznalonev>szabi</felhasznalonev>
    <email>szabi.lantos@freemail.com</email>
    <regisztracio_datuma>2022-12-02</regisztracio_datuma>
</felhasznalo>

<felhasznalo felhasznaloid="F2">
    <felhasznalonev>evike</felhasznalonev>
    <email>evike@citromail.com</email>
    <regisztracio_datuma>2016-04-15</regisztracio_datuma>
</felhasznalo>

<profil profilid="E1">
    <nev>Lantos Szabolcs</nev>
    <telefonszam>+36205555444</telefonszam>
    <utolso_belepes_datuma>2023-10-02</utolso_belepes_datuma>
</profil>

<profil profilid="E2">
    <nev>Nagy Éva</nev>
    <telefonszam>+36209453264</telefonszam>
    <utolso_belepes_datuma>2023-09-30</utolso_belepes_datuma>
</profil>

<fizetes fizetesid="T1">
    <osszeg>14990</osszeg>
    <tranzakcio_datuma>2023-10-02</tranzakcio_datuma>
    <fizetesimod>Kártya</fizetesimod>
</fizetes>

<fizetes fizetesid="T2">
    <osszeg>8990</osszeg>
    <tranzakcio_datuma>2023-09-30</tranzakcio_datuma>
    <fizetesimod>Kártya</fizetesimod>
</fizetes>

<termek termekid="P1" T-V_K="TVK1">
    <termeknev>Egér</termeknev>
    <ar>4990</ar>
    <keszlet>10</keszlet>
</termek>

<termek termekid="P2" T-V_K="TVK2">
    <termeknev>Billentyűzet</termeknev>
    <ar>12000</ar>
    <keszlet>25</keszlet>
</termek>

<termek termekid="P3" T-V_K="TVK3">
    <termeknev>ssd 256</termeknev>
```

```
        <ar>8990</ar>
        <keszlet>5</keszlet>
    </termek>

    <termek termekid="P4" T-V_K="TVK4">
        <termeknev>usb kábel</termeknev>
        <ar>500</ar>
        <keszlet>8</keszlet>
    </termek>


    <vasarlas >
        <szemelyes>0</szemelyes>
        <online>2</online>
    </vasarlas>

</onlinevasarlas>
```

1.4 Az XML dokumentum alapján XMLSchema készítése

Az XML Schema egy strukturált és szabványos módszert biztosít az adatok tárolására és kezelésére, és a leírás segít megérteni az egyes entitásokat, azok tulajdonságait és azok közötti kapcsolatokat.

Elemek és Attribútumok Típusainak Meghatározása: Az egyes XML elemek és attribútumok típusait meg kell határozni, figyelembe véve azok lehetséges értékeit, kötelezőségüket, és az esetleges referenciákat más típusokra.

Ref és Key Kulcsszavak Használata: Az XSD-ben a ref kulcsszó segítségével hivatkozhatunk más típusokra, így elkerülhetjük a redundanciát és elősegíthetjük az újrafelhasználhatóságot. A key és keyref kulcsszavak segítségével definiálhatunk egyedi kulcsokat és ezek referenciait.

Az XML Schema kódja:

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

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

    <!-- Felepites -->

    <xs:element name="onlinevasarlas">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="rendeles" type="rendelesTipus"
maxOccurs="unbounded"/>
                <xs:element name="felhasznalo" type="felhasznaloTipus"
maxOccurs="unbounded"/>
```

```
                <xs:element name="profil" type="profilTipus"
maxOccurs="unbounded"/>
                <xs:element name="fizetes" type="fizetesTipus"
maxOccurs="unbounded"/>
                <xs:element name="termek" type="termekTipus"
maxOccurs="unbounded"/>
                <xs:element name="vasarlas" type="vasarlasTipus"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>



    <!-- elsodleges kulcsok -->
    <xs:key name="rendeles_kulcs">
        <xs:selector xpath="rendeles" />
        <xs:field xpath="@rendelesid" />
    </xs:key>

    <xs:key name="felhasznalo_kulcs">
        <xs:selector xpath="felhasznalo" />
        <xs:field xpath="@felhasznaloid" />
    </xs:key>

    <xs:key name="profil_kulcs">
        <xs:selector xpath="profil" />
        <xs:field xpath="@profilid" />
    </xs:key>

    <xs:key name="fizetes_kulcs">
        <xs:selector xpath="fizetes" />
        <xs:field xpath="@fizetesid" />
    </xs:key>

    <xs:key name="termek_kulcs">
        <xs:selector xpath="termek" />
        <xs:field xpath="@termekid" />
    </xs:key>


    <!-- idegen kulcsok -->

     <xs:keyref refer="rendeles_kulcs" name="rendeles_idegen_kulcs">
        <xs:selector xpath="vasarlas" />
        <xs:field xpath="@R-V_K" />
    </xs:keyref>

    <xs:keyref refer="termek_kulcs" name="termek_idegen_kulcs">
        <xs:selector xpath="vasarlas" />
        <xs:field xpath="@T-V_K" />
    </xs:keyref>


    <!-- 1 : 1 kapcsolatok -->
    <xs:unique name="unique_profil">
        <xs:selector xpath="profil" />
        <xs:field xpath="@profilid" />
    </xs:unique>

</xs:element>
```

```
<!-- ComplexType Tipusok -->

<xs:complexType name="rendelesTipus">
            <xs:sequence>
                <xs:element name="rendeles_datuma" type="xs:date" />
                <xs:element name="szallitasi_cim" type="cimTipus" />
                <xs:element name="osszeg" type="xs:integer" />
            </xs:sequence>
            <xs:attribute name="rendelesid" type="xs:string"
use="required" />
            <xs:attribute name="R-V_K" type="xs:string"  use="required" />
</xs:complexType>

<xs:complexType name="cimTipus">
            <xs:sequence>
                <xs:element name="irsz" type="xs:integer" />
                <xs:element name="varos" type="xs:string" />
                <xs:element name="utca" type="xs:string" />
                <xs:element name="hsz" type="xs:integer" />
            </xs:sequence>
</xs:complexType>

<xs:complexType name="felhasznaloTipus">
            <xs:sequence>
                <xs:element name="felhasznalonev" type="xs:string" />
                <xs:element name="email" type="xs:string" />
                <xs:element name="regisztracio_datuma" type="xs:date" />
            </xs:sequence>
            <xs:attribute name="felhasznaloid" type="xs:string"
use="required" />
</xs:complexType>

<xs:complexType name="profilTipus">
            <xs:sequence>
                <xs:element name="nev" type="xs:string" />
                <xs:element name="telefonszam" type="xs:string" />
                <xs:element name="utolso_belepes_datuma" type="xs:date" />
            </xs:sequence>
            <xs:attribute name="profilid" type="xs:string"  use="required"
/>
</xs:complexType>

<xs:complexType name="fizetesTipus">
            <xs:sequence>
                <xs:element name="osszeg" type="xs:integer" />
                <xs:element name="tranzakcio_datuma" type="xs:date" />
                <xs:element name="fizetesimod" type="xs:string" />

            </xs:sequence>
            <xs:attribute name="fizetesid" type="xs:string"  use="required"
/>
</xs:complexType>

<xs:complexType name="termekTipus">
            <xs:sequence>
                <xs:element name="termeknev" type="xs:string" />
                <xs:element name="ar" type="xs:integer" />
                <xs:element name="keszlet" type="xs:integer" />
            </xs:sequence>
            <xs:attribute name="termekid" type="xs:string"  use="required"
/>
```

```
            <xs:attribute name="T-V_K" type="xs:string"  use="required" />

</xs:complexType>

    <xs:complexType name="vasarlasTipus">
        <xs:sequence>
            <xs:element name="szemelyes" type="xs:string" />
            <xs:element name="online" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

2. feladat

2.1 adatolvasás

Az DomReadNeptunkod.java kód egy DOM alapú XML feldolgozást valósít meg. A kód egy XML fájlt olvas be (XMLFA6vdv.xml), majd kiírja a fájl tartalmát a konzolra, strukturáltan.

Két metódusból áll:

processDocument Metódus:

Egy Document objektumot kap paraméterként, és feldolgozza az összes gyermek elemét.

Lekéri a dokumentum gyökér elemeit, majd végigiterál rajtuk.

Minden gyerekelem esetén meghívja a processElement metódust.

processElement Metódus:

Egy Node-ot kap paraméterként és feldolgozza az attribútumait, tartalmát és gyerekelemeit.

Kiírja az elem nevét, attribútumait és tartalmát.

Mind az elem, mind a szöveg csomópontokkal foglalkozik.

DOMReadFA6VDV.java forráskódja:

```
package hu.domparse.fa6vdv;

import org.w3c.dom.*;
import javax.xml.parsers.*;
```

```
import java.io.*;

public class DOMReadFA6VDV {

    public static void main(String[] args) {
        try {
            // XML fájl elérési útvonala
            String xmlfile = "XMLFA6VDV.xml";

            // XML fájl beolvasása és DOM objektum létrehozása
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new File(xmlfile));

            // Dokumentum feldolgozása
            processDocument(document);
            //processElement(document);
            //document.getDocumentElement().normalize();


        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void processDocument(Document document) {
        // Az összes elem feldolgozása
        NodeList rootNodes = document.getChildNodes();
        for (int i = 0; i < rootNodes.getLength(); i++) {
            Node rootNode = rootNodes.item(i);
            if (rootNode.getNodeType() == Node.ELEMENT_NODE) {
                // Az összes gyerekelem feldolgozása
                processElement(rootNode);
            }
        }
    }

    private static void processElement(Node element) {


        if (element.getNodeType() == Node.ELEMENT_NODE) {
            //Elem neve
            System.out.println("Elem: " + element.getNodeName());

            // Elem attribútumai
            NamedNodeMap keys = element.getAttributes();
            for (int i = 0; i < keys.getLength(); i++) {
                Node attribute = keys.item(i);
                System.out.println("kulcs: " + attribute.getNodeName() + "
= " + attribute.getNodeValue());
            }

            // Elem tartalma
            NodeList children = element.getChildNodes();
            for (int i = 0; i < children.getLength(); i++) {
                Node child = children.item(i);
                if (child.getNodeType() == Node.ELEMENT_NODE) {
                    // feldolgozás a gyerekelemeknél
                    processElement(child);
                } else if (child.getNodeType() == Node.TEXT_NODE &&
```

```
!child.getNodeValue().trim().isEmpty()) {
                    //tartalom kiírása
                    System.out.println("tartalom: " +
child.getNodeValue().trim());
                }
            }

            System.out.println();// Üres sor elvalasztáshoz
        }
    }
}
```

2.2 Adatmódosítás

DOMModifyNeptunkod.java program módosításokat hajt végre egy XML fájlban a DOM (Document Object Model) API segítségével. A program példa módosításokat tartalmaz, például egy rendelés összegének növelése, egy felhasználó nevének megváltoztatása, egy termék árának csökkentése, termék készlet módosítása, fizetési tranzakció dátum módosítását tartalmazza.

DOMModifyFa6vdv.java forráskódja:

```
package hu.domparse.fa6vdv;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

public class DOMModifyFa6vdv {

    public static void main(String[] args) {
        try {
            // XML fájl elérési útvonala
            String xmlFile = "XMLFA6VDV.xml";

            // XML fájl beolvasása és DOM objektum létrehozása
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new File(xmlFile));

            // Módosítások végrehajtása
            performModifications(document);

            // Módosított dokumentum kiírása a konzolra
            printDocument(document);

        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }

    private static void performModifications(Document document) {

        //  Módosítás: rendelés összegének növelése
        modifyOrderAmount(document, "R1", 35000);

        //  Módosítás: felhasználó nevének megváltoztatása
        modifyUserName(document, "F1", "Szabolcs Lantos");

        //  Módosítás: termék árának csökkentése
        modifyProductPrice(document, "P2", 2000);

        //  Módosítás: fizetési dátum átírása
        modifyTransactionDate(document, "T1", "2023-08-05");

        //  Módosítás: készlet változtatás
        modifyProductStock(document, "P1", 51);
    }

    //  Módosítás: rendelés összegének növelése
    private static void modifyOrderAmount(Document document, String
orderID, int amountIncrease) {
        NodeList rendelesNodeList =
document.getElementsByTagName("rendeles");
        // végig megyünk NodeList-en
        for (int i = 0; i < rendelesNodeList.getLength(); i++) {
            Node rendelesNode = rendelesNodeList.item(i);

            // Ellenőrzés, hogy a NodeList aktuális eleme egy ELEMENT_NODE
típusú elem-e
            if (rendelesNode.getNodeType() == Node.ELEMENT_NODE) {
                Element rendelesElement = (Element) rendelesNode;
                String currentOrderID =
rendelesElement.getAttribute("rendelesid");
                // Ellenőrzés, hogy az aktuális orderid azonosítója
megegyezik-e a keresettel
                if (currentOrderID.equals(orderID)) {
                    int currentAmount =
Integer.parseInt(rendelesElement.getElementsByTagName("osszeg").item(0).get
TextContent());
                    int newAmount = currentAmount + amountIncrease;
                    //módosítás végrehajtása

rendelesElement.getElementsByTagName("osszeg").item(0).setTextContent(Integ
er.toString(newAmount));
                    break;
                }
            }
        }
    }
    //  Módosítás: felhasználó nevének megváltoztatása
    private static void modifyUserName(Document document, String userID,
String newName){
        //NodeList létrehozása
        NodeList felhasznaloNodeList =
document.getElementsByTagName("felhasznalo");

            for (int i = 0; i < felhasznaloNodeList.getLength(); i++) {
                Node felhasznaloNode = felhasznaloNodeList.item(i);
```

```
                if (felhasznaloNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element felhasznaloElement = (Element) felhasznaloNode;
                    String currentUserID =
felhasznaloElement.getAttribute("felhasznaloid");

                    if (currentUserID.equals(userID)) {

felhasznaloElement.getElementsByTagName("felhasznalonev").item(0).setTextCo
ntent(newName);
                        break;
                    }
                }
            }
        }

    //  Módosítás: termék árának csökkentése
        private static void modifyProductPrice(Document document, String
productID, int priceDecrease) {
        NodeList termekNodeList = document.getElementsByTagName("termek");

        for (int i = 0; i < termekNodeList.getLength(); i++) {
            Node termekNode = termekNodeList.item(i);

            if (termekNode.getNodeType() == Node.ELEMENT_NODE) {
                Element termekElement = (Element) termekNode;
                String currentProductID =
termekElement.getAttribute("termekid");

                if (currentProductID.equals(productID)) {
                    int currentPrice =
Integer.parseInt(termekElement.getElementsByTagName("ar").item(0).getTextCo
ntent());
                    int newPrice = Math.max(0, currentPrice -
priceDecrease);

termekElement.getElementsByTagName("ar").item(0).setTextContent(Integer.toS
tring(newPrice));
                    break;
                }
            }
        }
    }

    //  Módosítás: fizetési dátum átírása
    private static void modifyTransactionDate(Document document, String
transactionID, String newDate) {
        // Fizetési tranzakciók NodeList létrehozása
        NodeList fizetesNodeList =
document.getElementsByTagName("fizetes");
        // végig megyünk NodeList-en
        for (int i = 0; i < fizetesNodeList.getLength(); i++) {

            Node fizetesNode = fizetesNodeList.item(i);
            // Ellenőrzés, hogy a NodeList aktuális eleme egy ELEMENT_NODE
típusú elem-e
            if (fizetesNode.getNodeType() == Node.ELEMENT_NODE) {
                // Az aktuális elem konvertálása Element típusúvá
                Element fizetesElement = (Element) fizetesNode;
                // Az aktuális fizetési tranzakcióhoz tartozó azonosító
lekérése
```

```
                String currentTransactionID =
fizetesElement.getAttribute("fizetesid");
                // Ellenőrzés, hogy az aktuális fizetési tranzakció
azonosítója megegyezik-e a keresettel
                if (currentTransactionID.equals(transactionID)) {
                    // A fizetési tranzakció dátum elemének lekérése és a
módosítás végrehajtása

fizetesElement.getElementsByTagName("tranzakcio_datuma").item(0).setTextCon
tent(newDate);
                    break;
                }
            }
        }
    }

    //  Módosítás: készlet változtatá
    private static void modifyProductStock(Document document, String
productID, int stockChange) {
        NodeList termekNodeList = document.getElementsByTagName("termek");

        for (int i = 0; i < termekNodeList.getLength(); i++) {
            Node termekNode = termekNodeList.item(i);

            if (termekNode.getNodeType() == Node.ELEMENT_NODE) {
                Element termekElement = (Element) termekNode;
                String currentProductID =
termekElement.getAttribute("termekid");

                if (currentProductID.equals(productID)) {
                    int currentStock =
Integer.parseInt(termekElement.getElementsByTagName("keszlet").item(0).getT
extContent());
                    int newStock =  currentStock + stockChange;

termekElement.getElementsByTagName("keszlet").item(0).setTextContent(Intege
r.toString(newStock));
                    break;
                }
            }
        }
    }


    private static void printDocument(Document document) {
        try {
            // Kiíratás a konzolra
            Transformer transformer =
TransformerFactory.newInstance().newTransformer();
            transformer.transform(new DOMSource(document), new
StreamResult(System.out));
        } catch (TransformerException e) {
            e.printStackTrace();
        }
    }
}
```

2.3 adatlekérdezés

DOMQueryFa6VDV.java program lekérdezéseket hajt végre egy XML fájlban a DOM API segítségével. A program példákat mutat be különböző lekérdezésekre, például felhasználónevek kiírása, összesített összeg kiírása, legnagyobb készletű termék nevének kiírása, legutóbbi regisztrációk kiírása és személyes vásárlások kiírása.

A DOMQueryFa6VDV.java forrás kódja:

```
package hu.domparse.fa6vdv;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;

public class DOMQueryFA6VDV {


    public static void main(String[] args) {

        // Az XML dokumentum betöltése
        Document document = loadXMLDocument("XMLFA6VDV.xml");

        // Példa lekérdezések
        printUserNames(document);
        printTotalAmountSpent(document);
        printMaxStockProduct(document);
        printRecentRegistrations(document);
        printOfflinePurchases(document);


        if (document != null) {
            // A dokumentum sikeresen betöltve
            // Itt folytathatod a további műveleteket
        } else {
            System.out.println("Hiba a dokumentum betöltésekor.");
        }

    }


    public static Document loadXMLDocument(String filePath) {
        try {
            // Létrehozunk egy DocumentBuilderFactory objektumot
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();

            // Létrehozunk egy DocumentBuilder objektumot a factory
segítségével
            DocumentBuilder builder = factory.newDocumentBuilder();

            // Betöltjük az XML dokumentumot
            File file = new File(filePath);
```

```
            Document document = builder.parse(file);

            // Opcionális: Normalizáljuk a dokumentumot
            document.getDocumentElement().normalize();

            // Visszaadjuk a betöltött dokumentumot
            return document;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }




    // Lekérdezés: Felhasználónevek kiírása
    private static void printUserNames(Document document) {
        NodeList users = document.getElementsByTagName("felhasznalo");
        for (int i = 0; i < users.getLength(); i++) {
            Node userNode = users.item(i);
            if (userNode.getNodeType() == Node.ELEMENT_NODE) {
                Element userElement = (Element) userNode;
                String userName =
userElement.getElementsByTagName("felhasznalonev").item(0).getTextContent()
;
                System.out.println("Felhasználónév: " + userName);
            }
        }
    }

    // Lekérdezés: Összesített összeg kiírása
    private static void printTotalAmountSpent(Document document) {
        NodeList payments = document.getElementsByTagName("fizetes");
        int totalAmount = 0;
        for (int i = 0; i < payments.getLength(); i++) {
            Node paymentNode = payments.item(i);
            if (paymentNode.getNodeType() == Node.ELEMENT_NODE) {
                Element paymentElement = (Element) paymentNode;
                int amount =
Integer.parseInt(paymentElement.getElementsByTagName("osszeg").item(0).getT
extContent());
                totalAmount += amount;
            }
        }
        System.out.println("Összesített összeg: " + totalAmount);
    }

    // Lekérdezés: Legnagyobb készletű termék nevének kiírása
    private static void printMaxStockProduct(Document document) {
        NodeList products = document.getElementsByTagName("termek");
        int maxStock = -1;
        String maxStockProductName = "";
        for (int i = 0; i < products.getLength(); i++) {
            Node productNode = products.item(i);
            if (productNode.getNodeType() == Node.ELEMENT_NODE) {
                Element productElement = (Element) productNode;
                int stock =
Integer.parseInt(productElement.getElementsByTagName("keszlet").item(0).get
TextContent());
```

```
                if (stock > maxStock) {
                    maxStock = stock;
                    maxStockProductName =
productElement.getElementsByTagName("termeknev").item(0).getTextContent();
                }
            }
        }
        System.out.println("Legnagyobb készletű termék neve: " +
maxStockProductName);
    }

    // Lekérdezés: Legutóbbi regisztrációk kiírása
    private static void printRecentRegistrations(Document document) {
        NodeList users = document.getElementsByTagName("felhasznalo");
        for (int i = 0; i < users.getLength(); i++) {
            Node userNode = users.item(i);
            if (userNode.getNodeType() == Node.ELEMENT_NODE) {
                Element userElement = (Element) userNode;
                String registrationDate =
userElement.getElementsByTagName("regisztracio_datuma").item(0).getTextCont
ent();
                System.out.println("Felhasználó regisztráció dátuma: " +
registrationDate);
            }
        }
    }

    // Lekérdezés:személyes vásárlások kiírása
    private static void printOfflinePurchases(Document document) {
        NodeList purchases = document.getElementsByTagName("vasarlas");
        for (int i = 0; i < purchases.getLength(); i++) {
            Node purchaseNode = purchases.item(i);
            if (purchaseNode.getNodeType() == Node.ELEMENT_NODE) {
                Element purchaseElement = (Element) purchaseNode;
                String offlinePurchase =
purchaseElement.getElementsByTagName("szemelyes").item(0).getTextContent();
                if (offlinePurchase.equals("0")) {
                    System.out.println("szemelyes vásárlás");
                }
            }
        }
    }
}
```

2.4 adatírás

DOMWriteFA6VDV.java program létrehoz egy XML dokumentumot a DOM (Document Object Model) API segítségével, majd adatokat ad hozzá, és végül kiírja az XML fájlt egy másik fájlba XMLFA6VDV_1.xml néven menti.

DOMWriteFA6VDV.java forráskódja:

```
package hu.domparse.fa6vdv;

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 DOMWriteFA6VDV {

    public static void main(String[] args) throws
ParserConfigurationException, TransformerException {
        // XML dokumentum létrehozása
        Document document = createXMLDocument();

        // Adatok hozzáadása a dokumentumhoz (a példa adatok alapján)
        addDataToDocument(document);

        // XML fájlba írás
        writeXMLDocument(document, "XMLFA6VDV_1.xml");
    }

    private static void writeXMLDocument(Document document, String
filePath) {
        try {
        // Létrehoz egy TransformerFactory-t
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();

        // Létrehoz egy Transformer objektumot
        Transformer transformer = transformerFactory.newTransformer();

        // Beállítja a kimeneti fájl karakterkódolását
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");

        // Beállítja a kimeneti XML fájl tartalmazzon behúzást
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-
amount", "2");

        // Létrehoz egy DOMSource-t a dokumentumhoz
        DOMSource source = new DOMSource(document);

        // Létrehoz egy StreamResult-t a kimeneti fájlhoz
        StreamResult result = new StreamResult(new File(filePath));
```

```
            // Végrehajtja a transzformációt és kiírja az XML-t a fájlba
            transformer.transform(source, result);

            System.out.println("Az XML dokumentum kiírása sikeres volt.");
        } catch (Exception e) {
            e.printStackTrace();
        }
        }

    private static Document createXMLDocument() throws
ParserConfigurationException {
            // XML dokumentum létrehozása
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.newDocument();

            // Gyökérelem létrehozása
            Element onlinevasarlas = document.createElement("onlinevasarlas");
            onlinevasarlas.setAttribute("xmlns:xsi",
"http://www.w3.org/2001/XMLSchema-instance");
            onlinevasarlas.setAttribute("xsi:noNamespaceSchemaLocation",
"xdmschema_FA6vdv.xsd");
            document.appendChild(onlinevasarlas);

            return document;
        }

    private static void addDataToDocument(Document document) {
            // adatok hozzáadása a dokumentumhoz
            Element rendeles1 = createRendelesElement(document, "R1", "RVK1",
"2023-01-01", "1328", "Budapest", "Kossuth utca", "12", "14990");
            Element rendeles2 = createRendelesElement(document, "R2", "RVK2",
"2023-01-01", "3300", "Eger", "Dobó", "56", "8990");

            Element felhasznalo1 = createFelhasznaloElement(document, "F1",
"szabi", "szabi.lantos@freemail.com", "2022-12-02");
            Element felhasznalo2 = createFelhasznaloElement(document, "F2",
"evike", "evike@citromail.com", "2016-04-15");

            Element profil1 = createProfilElement(document, "P1", "John Doe",
"06204455788", "2023-04-22");
            Element profil2 = createProfilElement(document, "P2", "Jane Doe",
"06301455789", "2023-02-01");

            Element termek1 = createTermekElement(document, "T1", "Laptop",
"250000", "20", "TVK1");
            Element termek2 = createTermekElement(document, "T2", "Mobil
telefon", "80000", "50","TVK2");


            // Gyökérelem kiválasztása
            Element onlinevasarlas = document.getDocumentElement();

            // elemek hozzáadása a gyökérelemhez
            onlinevasarlas.appendChild(rendeles1);
            onlinevasarlas.appendChild(rendeles2);
            onlinevasarlas.appendChild(felhasznalo1);
            onlinevasarlas.appendChild(felhasznalo2);
            onlinevasarlas.appendChild(profil1);
```

```
        onlinevasarlas.appendChild(profil2);
        onlinevasarlas.appendChild(termek1);
        onlinevasarlas.appendChild(termek2);

    }

    // Az alábbiakban olyan függvények találhatók, amelyek létrehozzák a
különböző típusú elemeket (rendelés, felhasználó).

    private static Element createRendelesElement(Document document, String
rendelesid, String RVK, String rendelesDatum, String irsz, String varos,
String utca, String hsz, String osszeg) {
        // Létrehoz egy "rendeles" elemet
        Element rendeles = document.createElement("rendeles");

        // Beállítja az "rendelesid" attribútumot
        rendeles.setAttribute("rendelesid", rendelesid);

        // Beállítja az "R-V_K" attribútumot
        rendeles.setAttribute("R-V_K", RVK);

        // Létrehoz és hozzáadja a "rendeles_datuma" elemet
        Element rendelesDatumElem =
document.createElement("rendeles_datuma");

rendelesDatumElem.appendChild(document.createTextNode(rendelesDatum));
        rendeles.appendChild(rendelesDatumElem);

        // Létrehozza a "szallitasi_cim" elemet és hozzáadja az alárendelt
elemeket
        Element szallitasiCim = document.createElement("szallitasi_cim");
        szallitasiCim.appendChild(createElement(document, "irsz", irsz));
        szallitasiCim.appendChild(createElement(document, "varos", varos));
        szallitasiCim.appendChild(createElement(document, "utca", utca));
        szallitasiCim.appendChild(createElement(document, "hsz", hsz));
        rendeles.appendChild(szallitasiCim);

        // Létrehoz és hozzáadja az "osszeg" elemet
        Element osszegElem = document.createElement("osszeg");
        osszegElem.appendChild(document.createTextNode(osszeg));
        rendeles.appendChild(osszegElem);

        // Visszaadja az elkészült "rendeles" elemet
        return rendeles;
    }

    // Segédfüggvény az egyszerű XML elemek létrehozásához
    private static Element createElement(Document document, String tagName,
String textContent) {
        Element element = document.createElement(tagName);
        element.appendChild(document.createTextNode(textContent));
        return element;
    }
    private static Element createFelhasznaloElement(Document document,
String felhasznaloid, String felhasznalonev, String email, String
regisztracioDatum) {
        // Létrehoz egy "felhasznalo" elemet
        Element felhasznalo = document.createElement("felhasznalo");

        // Beállítja a "felhasznaloid" attribútumot
        felhasznalo.setAttribute("felhasznaloid", felhasznaloid);
```

```
        // Létrehoz és hozzáadja a "felhasznalonev" elemet
        Element felhasznalonevElem =
document.createElement("felhasznalonev");

felhasznalonevElem.appendChild(document.createTextNode(felhasznalonev));
        felhasznalo.appendChild(felhasznalonevElem);

        // Létrehoz és hozzáadja az "email" elemet
        Element emailElem = document.createElement("email");
        emailElem.appendChild(document.createTextNode(email));
        felhasznalo.appendChild(emailElem);

        // Létrehoz és hozzáadja a "regisztracio_datuma" elemet
        Element regisztracioDatumElem =
document.createElement("regisztracio_datuma");

regisztracioDatumElem.appendChild(document.createTextNode(regisztracioDatum
));
        felhasznalo.appendChild(regisztracioDatumElem);

        // Visszaadja az elkészült "felhasznalo" elemet
        return felhasznalo;
    }

    private static Element createProfilElement(Document document, String
profilid, String nev, String telefonszam, String utolso_belepes_datuma) {
        // Létrehozza a "profil" elemet
        Element profil = document.createElement("profil");
        profil.setAttribute("profilid", profilid);

        // Létrehozza és hozzáadja a "nev" elemet
        Element nevElement = document.createElement("nev");
        nevElement.appendChild(document.createTextNode(nev));
        profil.appendChild(nevElement);

        // Létrehozza és hozzáadja a "telefonszam" elemet
        Element telefonszamElement = document.createElement("telefonszam");

telefonszamElement.appendChild(document.createTextNode(telefonszam));
        profil.appendChild(telefonszamElement);

        // Létrehozza és hozzáadja az "utolso_belepes_datuma" elemet
        Element utolsoBelepesElement =
document.createElement("utolso_belepes_datuma");

utolsoBelepesElement.appendChild(document.createTextNode(utolso_belepes_dat
uma));
        profil.appendChild(utolsoBelepesElement);

        // Visszaadja a létrehozott "profil" elemet
        return profil;
    }

    private static Element createTermekElement(Document document, String
termekid, String termeknev, String ar, String keszlet, String tVK) {
        Element termek = document.createElement("termek");
        termek.setAttribute("termekid", termekid);
        termek.setAttribute("T-V_K", tVK);

        Element termeknevElem = createElement(document, "termeknev",
```

```
termeknev);
        Element arElem = createElement(document, "ar", ar);
        Element keszletElem = createElement(document, "keszlet", keszlet);

        termek.appendChild(termeknevElem);
        termek.appendChild(arElem);
        termek.appendChild(keszletElem);

        return termek;
    }



}
```