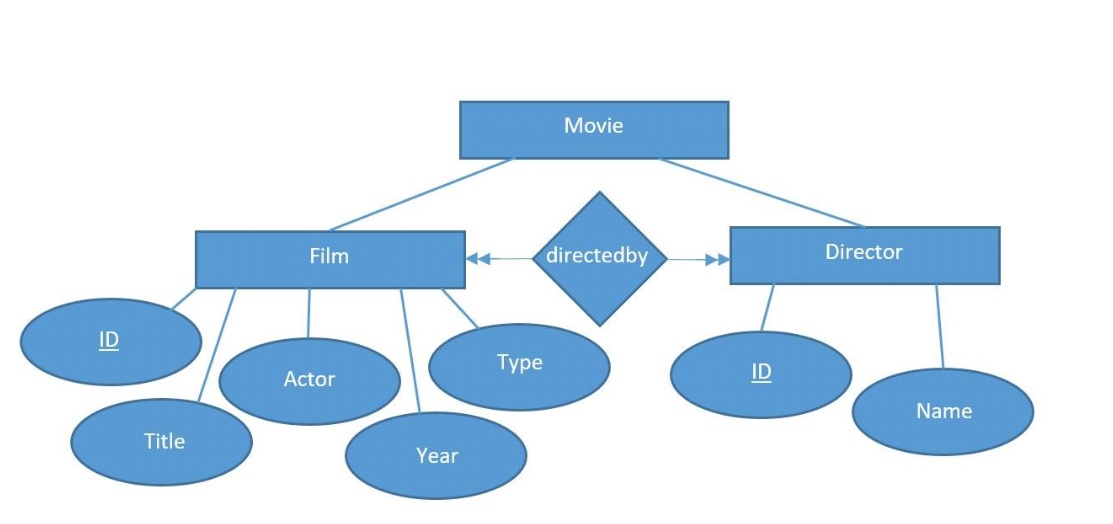
JEGYZŐKÖNYV   
Adatkezelés XML környezetben   
Féléves feladat

Készítette: Ispán Dávid   
Neptunkód: F03HHD

****  
**1a) Az adatbázis ER modell: (Legyen legalább 4 egyed, többféle kapcsolat, mindenféle   
tulajdonság! Csak programmal rajzolt ábra megfelelő!))**

**1b) Az adatbázis konvertálása XDM modellre: (Rövid leírás! Csak programmal rajzolt ábra   
megfelelő!)**

**1c) Az XDM modell alapján XML dokumentum készítése: (Ide kerül az XML kódja!)**

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

**<!DOCTYPE movie[**

**<!ELEMENT movie (film\*, director\*)>**

**<!ELEMENT film (title, actor, year, type)>**

**<!ATTLIST film filmID CDATA #REQUIRED**

**directedby CDATA #REQUIRED>**

**<!ELEMENT title (#PCDATA)>**

**<!ELEMENT actor (#PCDATA)>**

**<!ELEMENT year (#PCDATA)>**

**<!ELEMENT type (#PCDATA)>**

**<!ELEMENT director (name)>**

**<!ATTLIST director directID CDATA #REQUIRED**

**created CDATA #REQUIRED>**

**<!ELEMENT name (#PCDATA)>**

**]>**

**<movie>**

**<film filmID="F1" directedby="direct1">**

**<title>Alien</title>**

**<actor>Sigourney Weaver</actor>**

**<year>1979</year>**

**<type>sci-fi</type>**

**</film>**

**<film filmID="F2" directedby="direct2">**

**<title>Terminator</title>**

**<actor>Arnold Schwarzenegger</actor>**

**<year>1984</year>**

**<type>action</type>**

**</film>**

**<film filmID="F3" directedby="direct2">**

**<title>Titanic</title>**

**<actor>Leonardo DiCaprio</actor>**

**<year>1997</year>**

**<type>drama</type>**

**</film>**

**<film filmID="F4" directedby="direct3, direct4 ">**

**<title>Avengers 4</title>**

**<actor>Robert Downey Jr.</actor>**

**<year>2019</year>**

**<type>comic</type>**

**</film>**

**<director directID="direct1" created="F1">**

**<name>Ridley Scott</name>**

**</director>**

**<director directID="direct2" created="F2, F3">**

**<name>James Cameron</name>**

**</director>**

**<director directID="direct3" created="F4">**

**<name>Anthony Russo</name>**

**</director>**

**<director directID="direct4" created="F4">**

**<name>Joe Russo</name>**

**</director>**

**</movie>**

**1d) Az XML dokumentum alapján XMLSchema készítése (saját típusok): (Ide kerül az   
XML Schema kódja!)**

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

**<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"**

**xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning" elementFormDefault="qualified"**

**vc:minVersion="1.0" vc:maxVersion="1.1">**

**<xs:element name="movie">**

**<xs:complexType>**

**<xs:sequence>**

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

**<xs:complexType>**

**<xs:sequence>**

**<xs:element name="title" type="string"/>**

**<xs:element name="actor" type="string"/>**

**<xs:element name="year" type="string"/>**

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

**</xs:sequence>**

**<xs:attribute name="filmID" type="string"/>**

**<xs:attribute name="directedby" type="string"/>**

**</xs:complexType>**

**</xs:element>**

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

**<xs:complexType>**

**<xs:sequence>**

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

**</xs:sequence>**

**<xs:attribute name="directID" type="string"/>**

**<xs:attribute name="created" type="string"/>**

**</xs:complexType>**

**</xs:element>**

**</xs:sequence>**

**</xs:complexType>complexType>**

**</xs:element>**

**</xs:schema>**

2. feladat

**A feladat egy DOM program készítése az XML dokumentum adatainak adminisztrálása alapján: (ide kerül a kód - comment együtt)**

**2a) adatolvasás (kód – comment együtt) - fájlnév**

**package domparseF03HHD;**

**import org.w3c.dom.Document;**

**import org.w3c.dom.Element;**

**import org.w3c.dom.Node;**

**import org.w3c.dom.NodeList;**

**import org.xml.sax.SAXException;**

**import javax.xml.parsers.DocumentBuilder;**

**import javax.xml.parsers.DocumentBuilderFactory;**

**import javax.xml.parsers.ParserConfigurationException;**

**import java.io.File;**

**import java.io.IOException;**

**import java.io.InputStream;**

**public class DomReadF03HHD {**

**private static final String FILENAME = "XMLF03HHD.xml";**

**public static void main(String[] args) {**

**DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();**

**try {**

**dbf.setFeature(XMLConstants.FEATURE\_SECURE\_PROCESSING, true);**

**DocumentBuilder db = dbf.newDocumentBuilder();**

**Document doc = db.parse(new File(FILENAME));**

**doc.getDocumentElement().normalize();**

**NodeList list1 = doc.getElementsByTagName("film");**

**for (int temp = 0; temp < list1.getLength(); temp++) {**

**Node node = list1.item(temp);**

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

**Element element = (Element) node;**

**String id1 = element.getAttribute("filmID");**

**String id2 = element.getAttribute("directedby");**

**String title = element.getElementsByTagName("title").item(0).getTextContent();**

**String actor = element.getElementsByTagName("actor").item(0).getTextContent();**

**String year = element.getElementsByTagName("year").item(0).getTextContent();**

**String type = element.getElementsByTagName("type").item(0).getTextContent();**

**System.out.println("Film ID : " + id1);**

**System.out.println("Title : " + title);**

**System.out.println("Director : " + id2);**

**System.out.println("Actor: " + actor);**

**System.out.println("Year : " + year);**

**}**

**}**

**NodeList list2 = doc.getElementsByTagName("director");**

**for (int temp = 0; temp < list2.getLength(); temp++) {**

**Node node = list2.item(temp);**

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

**Element element = (Element) node;**

**String id1 = element.getAttribute("directID");**

**String id2 = element.getAttribute("created");**

**String name = element.getElementsByTagName("name").item(0).getTextContent();**

**System.out.println("Director ID: " + id1);**

**System.out.println("Director : " + name);**

**}**

**} catch (ParserConfigurationException | SAXException | IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**2b) adatmódosítás (kód – comment együtt) – fájlnév**

package domparseF03HHD;

import java.io.File;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerConfigurationException;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.TransformerFactoryConfigurationError;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

public class DomModifyF03HHD {

public static void main(String[] args) {

String filePath = "XMLF03HHD.xml";

File xmlFile = new File(filePath);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder;

try {

dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

delete(doc);

newAdd(doc);

writeXMLFile(doc);

} catch (SAXException | ParserConfigurationException | IOException | TransformerException e1) {

e1.printStackTrace();

}

}

private static void writeXMLFile(Document doc)

throws TransformerFactoryConfigurationError, TransformerConfigurationException, TransformerException {

doc.getDocumentElement().normalize();

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

StreamResult result = new StreamResult(new File("modifiedmovie.xml"));

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

transformer.transform(source, result);

System.out.println("XML modified");

}

private static void newAdd(Document doc) {

NodeList movie = doc.getElementsByTagName("Film");

Element emp = null;

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

emp = (Element) movie.item(i);

Element awardElement = doc.createElement("Award");

awardElement.appendChild(doc.createTextNode("Oscar Award"));

emp.appendChild(awardElement);

}

}

private static void delete(Document doc) {

NodeList movie = doc.getElementsByTagName("Film");

Element film = null;

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

film = (Element) movie.item(i);

Node yearNode = film.getElementsByTagName("Year").item(0);

film.removeChild(yearNode);

}

}

}