SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Hrvoje Lesar

RAD S XML U JAVI

PROJEKT

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Hrvoje Lesar

Matični broj: 0016133479

Studij: Organizacija poslovnih sustava

RAD S XML U JAVI

PROJEKT

Mentor :

Izv. prof. dr. sc. Sandro Gerić

Varaždin, Svibanj 2023.

Sadržaj

1. Uvod

2. XML

XML je standard za označavanje dokumenata koji podržava W3C organizacija. Kratica XML stoji za Extensible Markup Language. Kroz XML se definira generička sintaksa korištena za označavanje podataka jednostavnim, za ljude čitljivim oznakama [1]. Ovakav format dokumenta omogućuje visoku razinu kostumizacije načina definiranja i prikaza podataka. Neki od primjera gdje se koristi su vektorska grafika, elektronička razmjena podataka, serijalizacija objekata (pretvaranje Java objekta u XML dokument), poziva udaljenih procedura.

2.1. Struktura

Podatci u XML dokumentu su zapisani kao tekst. Podatci su okruženi tekstualnim oznakama koje opisuju podatak. Osnovna jedinica podatka i oznake naziva se element. U nastavku će biti definirane ostale vrste čvorova dostupnih u XML sintaksi. XML specifikacija definira točnu sintaksu koja se u dokumentu mora poštivati, definira kako su elementi orkuženi oznakama (tagovima), kako izgleda tag, koji nazivi tagova su prihvatljivi, gdje se atributi nalaze i njihovu sintaksu.

Isječak koda 1: Vrste čvorova

```
<?xml version="1.0" encoding="UTF-8"?>
<Hrvatska>
    <!-- Prognoza za datum 20.05.2023. -->
    <DatumTermin>
        <Datum>20.05.2023</Datum>
        <Termin>20</Termin>
    </DatumTermin>
    <Grad autom="0">
        <GradIme>žVaradin</GradIme>
        <Lat> 46.28</Lat>
        <Lon> 16.36</Lon>
        <Podatci>
            <Temp> 19.7</Temp>
            <Vlaga>75</Vlaga>
            <Tlak>1015.2</Tlak>
            <TlakTend>-</TlakTend>
            <VjetarSmjer>NE</VjetarSmjer>
            <VjetarBrzina> 3.5</VjetarBrzina>
            <Vrijeme>žpreteno čoblano</Vrijeme>
            <VrijemeZnak>4</VrijemeZnak>
        </Podatci>
    </Grad>
</Hrvatska>
```

U isječku je prikazan jedan XML dokument. Dokument prikazuje prognozu vremena na određeni dan. Kroz dokument možemo vidjeti različite vrste čvorova koje XML sintaksa podržava tj. možemo definirati 5 vrsta:

1. Uputa za obradu (meta element) - u prvoj liniji definirani je meta element koji opisuje

dokument. Prikazani element definira verziju XML-a koju prikazuje dokument i vrstu kodiranja teksta. Meta elementi se koriste za opis dokumenta i ne mogu imati sadržavati druge elemente.

2. Korijenski element - u primjeru je korijenski element prikazan tagom `Hrvatska`. Pravilni XML dokument mora sadržavati točno jedan korijenski element u kojemu su sadržani svi ostali elementi dokumenta.

3. Element - XML element s nazivom, atributima i listom elemenata djece. Primjer elementa može biti bilo koji od elemenata definiranih poslije korijenskog. Element `Grad` je dobar primjer elementa koji sadrži više elemenata djece, sadrži jedan atribut nazvan `autom` s vrijednošću `0`.

4. Tekst - niz znakova između otvarajućeg i zatvarajućeg taga. `Varaždin` je primjer teksta, vrijednosti koju sadrđi tag `GradIme`.

5. Komentar - koristan za ostavljanje napomena ili komentara u dokumentu. Kod parsiranja ovaj element se preskače. U primjeru se komentar nalazi u trećoj liniji, komentar ne može sadržavati druge elemente, sadržaj komentara su podaci komentara.

2.2. Pravila

Pravilno oblikovanje dokumenta je minimalni zahtjev za XML dokument [2]. Dokument koji nije pravilno oblikovan nije XML dokument. Parseri ga ne mogu pročitati i nije im dozvoljeno popravljati neispravan dokument, ne može pretpostaviti za koju svrhu je autor namijenio dokument te ispraviti dokument, kao što je moguće kod HTML-a. Kako bi XML dokument bio pravilno oblikovan mora sadržavati [3]:

- točno jedan korijenski element,
- svi početni tagovi moraju imati završne tagove,
- vrijednosti u atributima moraju biti u navodnicima,
- sadržaj mora biti definiran unutar character seta,
- tagovi mogu biti ugniježđeni ali se ne smiju preklapati.

3. XML u javi

U ovom poglavlju ćemo proći kroz nekoliko načina učitavanja i parsiranja XML datoteke u programskom jeziku Java. Prikazati kako su pristupi parsiranju različiti te koje su prednosti i nedostaci kod korištenja određene vrste parsera.

3.1. XSD

XSD je kratica za XML Schema Definition. XSD se koristi za definiranje strukture XML dokumenta, sama shema je pisana u XML-u. Svi podaci koji se u java projektu učitavaju prvo se validiraju prema XSD zadanoj shemi. U projektu su korišteni tri različiti XML dokumenti što znači da ujedno postoje i tri različite XSD sheme.

Isječak koda 2: Primjer XSD dokumenta za sedmodnevnu prognozu vremena

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

    <xs:complexType name="grad">
        <xs:sequence>
            <xs:element name="GradIme" type="xs:string" />
            <xs:element name="Tmax" type="xs:string" />
            <xs:element name="Tmin" type="xs:string" />
            <xs:element name="Tmin5" type="xs:string" />
            <xs:element name="Obor" type="xs:double" />
            <xs:element name="Snijeg" type="xs:string" />
            <xs:element name="VlagaMax" type="xs:integer" />
            <xs:element name="VlagaMin" type="xs:integer" />
            <xs:element name="Sunce" type="xs:string" />
            <xs:element name="Tna5Max" type="xs:string" />
            <xs:element name="Tna5Min" type="xs:string" />
            <xs:element name="Tna20Max" type="xs:string" />
            <xs:element name="Tna20Min" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="podaci">
        <xs:sequence>
            <xs:element name="Grad" type="grad" minOccurs="0" maxOccurs="unbounded"
                />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="agroMeteoroloskiPodaci">
        <xs:sequence>
            <xs:element name="Naslov" type="xs:string" />
            <xs:element name="Podaci" type="podaci" />
        </xs:sequence>
    </xs:complexType>

    <xs:element name="AgroMeteoroloskiPodaci" type="agroMeteoroloskiPodaci" />
```

```
</xs:schema>
```

Kroz priloženi dokument je definirana struktura XML dokumenta koji će sadržavati podatke o sedmodnevnoj prognozi vremena. Shema definira jedan korijenski element koji ima prilagođeni tip podatka. `AgroMeteoroloskiPodaci` je korijenski element koji mora biti prisutan u XML dokumentu koji ova shema validira. Korijenski element mora se sastojati od dva druga elementa, a to su `Naslov` i `Podaci`. `Naslov` je jednostavni element koji sadrži vrijednost naslova, dok su `Podaci` element koji se sastoji od više podelemenata tipa `Grad`.

Isječak koda 3: Klasa za validaciju XML datoteke prema XSD shemi

```
public class XSDValidator {
    public static void validiraj(File xmlDat, File xsdDat) throws SAXException,
        IOException {
        SchemaFactory schemaFactory = SchemaFactory.newInstance(XMLConstants.
            W3C_XML_SCHEMA_NS_URI);
        Schema schema = schemaFactory.newSchema(xsdDat);
        Validator validator = schema.newValidator();
        validator.validate(new StreamSource(xmlDat));
    }
}
```

Klasa sadrži jednu metodu koja kao ulaz prima XML datoteku i XSD datoteku. U metodi se kreira nova instanca `SchemaFactory` koja kao arugment prima putanju do sheme koja će se koristiti za validaciju XML dokumenta. U ovom slučaju se koristi ugrađena konstana `XMLConstants.W3C_XML_SCHEMA_NS_URI` koja je zapravo putanja na imenski prostor (namespace) XSD sheme koji definira pravila sheme. Potom se kreira nova instanca sheme koja za ulazni argument uzima datoteku sheme, poslijednje se kreira objekt validator kojemu se predaje XML datoteka koju želimo validirati prema zadanoj shemi. U slučaju da nije moguće validirati ulaznu datoteku validator postavlja iznimku i valja grešku kod validiranja dokumenta.

3.2. SAX

SAX je prva vrsta parsera koju ćemo pregledati. SAX prasira dokument redak po redak te pritom nailazka na otvarajući, zatvarajući tag ili podatke pokreće događaj (event). Prednost SAX-a je što kroz parsiranje koristi minimalno memorije tj. ne učitava cijelu XML datoteku u memoriju nego inkrementalno prolazi kroz datuteku, te korisnik po potrebi izvlači podatke koje treba iz datoteke. Tako da je ovaj parser vrlo pogodan za prasiranje velikih XML dokumenata.

Kako bi mogli parsirati dokument korištenjem ovog parsera prvo je potrebno napraviti klasu koja nasljeđuje klasu `DefaultHandler` iz SAX knjižnice. Kako bi parser mogao pokretati događaje potrebno ih je implementirati u klasi. Neki od najvažnijih metoda koje se mogu implementirati su `startElement(...)`, `characters(...)`, `endElement(...)`. `startDocument()` i `endDocument()`.

Isječak koda 4: Primjer implementacije metoda za prihvačanje događaja

```
public class SAXPrognoza extends DefaultHandler {
```

```
@Override
public void startElement(String uri, String localName, String qName, Attributes
    attributes) {
    switch (qName) {
        case "izmjena": {
            this.izmjena = new Izmjena();
            this.izmjena.attrRun = Integer.parseInt(attributes.getValue("run"));
            break;
        }
        case "grad": {
            this.trenutniGrad = new Grad();
            this.trenutniGrad.attrIme = attributes.getValue("ime");
            this.trenutniGrad.attrCode = attributes.getValue("code");
            break;
        }
        case "dan": {
            this.trenutniDan = new Dan();
            this.trenutniDan.attrDatum = attributes.getValue("datum");
            this.trenutniDan.attrDtj = attributes.getValue("dtj");
            this.trenutniDan.attrSat = Integer.parseInt(attributes.getValue("sat
                "));
            break;
        }
    }
}

@Override
public void characters(char[] ch, int start, int lenght) {
    this.trenutnaVrijednost = new String(ch, start, lenght);
}

@Override
public void endElement(String uri, String localName, String qName) {
    switch (qName) {
        case "izmjena": {
            this.izmjena.vrijednost = this.trenutnaVrijednost;
            break;
        }
        case "t_2m": {
            this.trenutniDan.t2m = Integer.parseInt(this.trenutnaVrijednost);
            break;
        }
        case "simbol": {
            this.trenutniDan.simbol = this.trenutnaVrijednost;
            break;
        }
        case "vjetar": {
            this.trenutniDan.vjetar = this.trenutnaVrijednost;
            break;
        }
        case "oborina": {
            this.trenutniDan.oborina = Double.parseDouble(this.
                trenutnaVrijednost);
```

```
                break;
            }
            case "dan": {
                this.trenutniGrad.dan.add(this.trenutniDan);
                break;
            }
            case "grad": {
                gradovi.add(this.trenutniGrad);
                break;
            }
        }
    }
}
```

Isječak prikazuje metode koje su implementirane u klasi `SAXPrognoza` koja se u projektu koristi za parsiranje sedmodnevne vremenske prognoze. U isječku nije prikazana potpuna klasa, fale neki članovi klase, no istaknute su metode korištene za parsiranje. Metoda `startElement` se poziva kad SAX parser naiđe na početni XML element. Ovisno o nazivu elementa kreira se novi objekt `Izmjena`, `Grad` ili `Dan`. Istovremeno se za svaki element su parsirani atributi te dodani objektu, ako element sadrži neke atribute. Metoda `characters` se poziva kada parser uspejšno parsira vrijednost između dva elementa. U klasi vrijednost spremamo za kasnije korištenje. Zadnja korištena metoda je `endElement` koja se poziva dok parser naiđe završni element. U metodi se prema nazivu završnog elementa zapisuje vrijednost tog elementa u odgovarajući objekt. Vrijednost se po potrebi pretvara u tip podataka koji objekt traži.

Sad kad imamo definiran način kreiranja objekata iz XML dokumenta možemo kreirati instancu SAX parsera i predati mu dokument koji će parsirati i pretvoriti u objekte koje je moguće koristiti u programu.

Isječak koda 5: Kreiranje instance SAX parsera i parsiranje XML datoteke

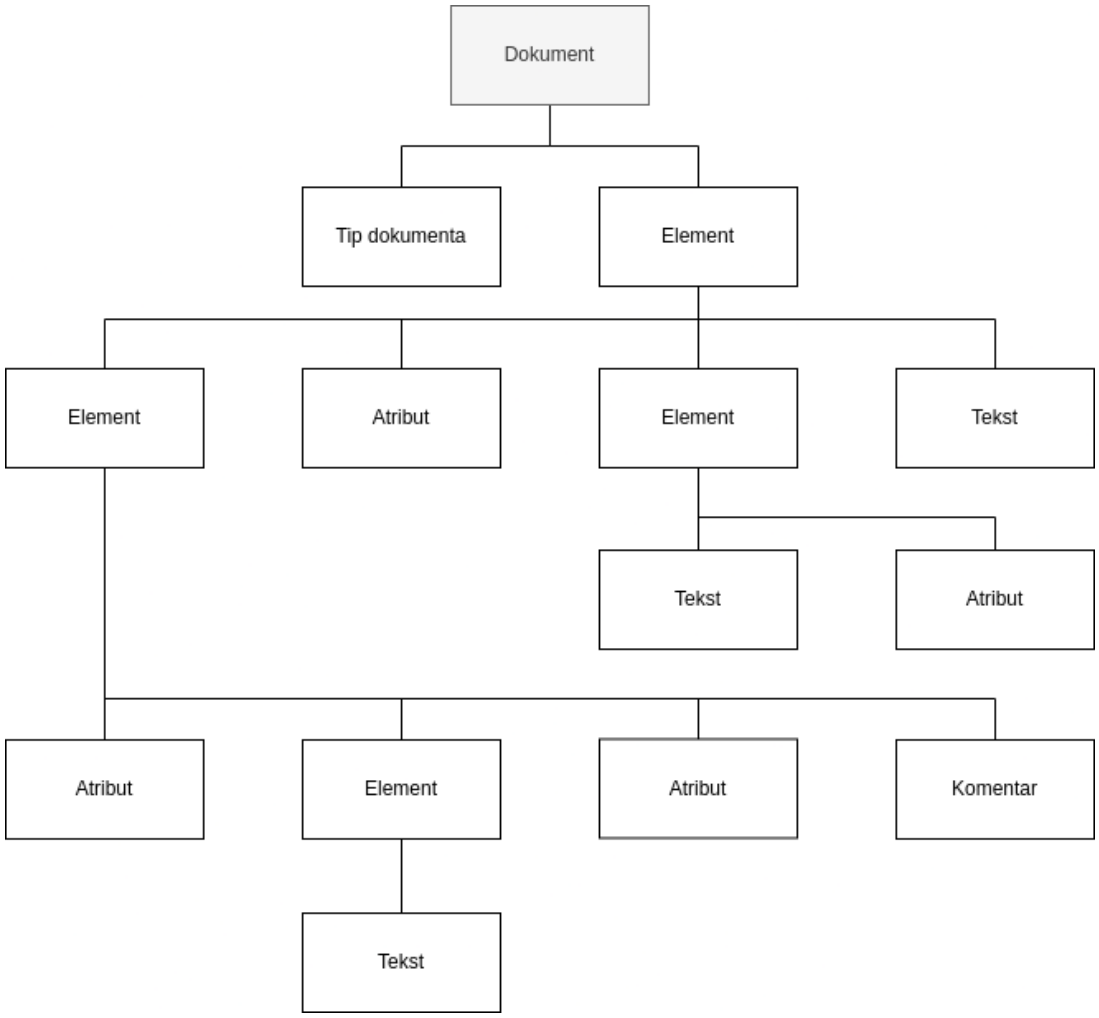
```
SAXParserFactory saxFactory = SAXParserFactory.newInstance();
SAXParser saxParser = saxFactory.newSAXParser();
SAXPrognoza prognoza = new SAXPrognoza();
saxParser.parse(DemoDatoteke.demoTrodnevnaPrognozaXML(), prognoza);
```

U ovom isječku je vidljivo kak se kreira instancira SAX parser. Najzanimljiviji dio je u četvrtoj liniji. Metoda `parse` kao argumente prima XML datoteku i klasu koja ima implementiran `DefaultHandler`. U ovom slučaju klasa `SAXPrognoza` implementira potrebne metode te `parse` koristi iste kod parsiranja tj. kod pokretanja događaja.

3.3. DOM

DOM parser, Data Object Model parser pretvara XML datoteku u DOM reprezentaciju. DOM se prikazuje kao u modelu stabla. Korijen stabla je čvor dokument iz kojeg se granaju svi ostali elementi. U kontekstu XML-a na istoj razini se nalazi element, atributi, vrijednost elementa (ako ima vrijednost) [4]. Sljedeća slika prikazuje primjer stabla koje je kreirano nakon

parsiranja XML dokumenta. Za razliku od SAX parsera DOM učitava cijeli dokument u memoriju i manje je prikladan za velike dokumente.



Slika 1: DOM stablo

Isječak koda 6: Dohvačanje vrijednosti iz DOM stabla

```
public class DOMPrognoza {

    public String ispisPrognoze() {
        Element korijenskiElement = this.xmlDocument.getDocumentElement();

        Element prognozaDatum = (Element) korijenskiElement.getElementsByTagName("
            DatumTermin").item(0);
        String datum = prognozaDatum.getElementsByTagName("Datum").item(0).
            getTextContent();

        ...

        String imeGrada = grad.getElementsByTagName("GradIme").item(0).
            getTextContent();
        Element podatciElement = (Element) grad.getElementsByTagName("Podatci").item
            (0);
```

```
        double temperatura = Double.parseDouble(podatciElement.getElementsByTagName(
            "Temp").item(0).getTextContent());
        String vrijeme = podatciElement.getElementsByTagName("Vrijeme").item(0).
            getTextContent();

        ...
    }

    Element pronadiGrad(String imeGrada, Element pocetniElement) throws Exception {
        NodeList gradCvorovi = pocetniElement.getElementsByTagName("Grad");
        int len = gradCvorovi.getLength();
        for (int i = 0; i < len; i++) {
            Node gradCvor = gradCvorovi.item(i);
            if (gradCvor.getNodeType() == Node.ELEMENT_NODE) {
                Element gradElement = (Element) gradCvor;
                if (gradElement.getElementsByTagName("GradIme").item(0).
                    getTextContent().equals(imeGrada)) {
                    return gradElement;
                }
            }
        }
        throw new Exception("Grad␣nije␣bio␣đpronaen");
    }
```

U isječku se nalazi kod koji iz DOM stabla izvlači potrebne podatke za ispis vremenske prognoze. Metoda `ispisPrognoze` počinje s dohvaćanjem korijenskog elementa iz DOM stabla, korijenski element se dohvaća metodom `getDocumentElement`. Slijedeće se iz korijenskog elementa dohvaća datum. To se radi na način da na korijenskom elementu pretražimo sve elemente koji imaju tag `DatumTermin`. Metoda `getElementsByTagName` vraća listu elemenata s određenim nazivom. Kako smo sigurni da nema više od jednog elementa koristi se metoda `item(0)` da se izvadi prvi element u listi. Na isti način se dobiva sam datum iz DatumTermin, jedina razlika je što na kraju pozivamo metodu `getTextContent` koja vraća datum u tekstualnom obliku. Na sličan način se dobivaju ostali podaci kao što su `imeGrada`, `temperatura`, `vrijeme`.

Druga metoda koja je prikazana u isječku je `pronadiGrad`. Metoda prolazi kroz sve elemente koji imaju tag `Grad`. Za svaki pregledava ako je taj čvor zapravo tipa element, ako je taj uvjet zadovoljen pretvara trenutni grad iz tipa `Node` u tip `Element` kako bi bilo moguće pregledati podatke o elementu. Slijedeće pregledava vrijednost sadržanog elementa i ako je vrijednost jednaka ulaznom argumentu `imeGrada` metoda vraća pronađeni element.

3.4. JAXB

JAXB (Java Architecture for XML Binding) pruža API za jednostavno pretvaranje objekata u XML i iz XML-a u objekte. JAXB je vrlo jednostavan za koristiti i zahtjeva najmanje pisanja koda. Funkcionira kroz korištenje anotacija na postojećim objektima. Anotacijama se definiraju koji dijelovi objekta spadaju u XML dokument, koji dijelovi su elementi, koji dio je korijenski element, koje dijelove treba preskočiti jer nisu dio XML kojega generiramo ili učitavamo.

Za definiranje korijenskog elementa na klasu se postavlja anotacija `@XmlRootElement`, sami elementi se definiraju postavljanjem anotacije `@XmlElement` na članove klase, atributi se definiraju anotacijom `@XmlAttribute`. Po potrebi se mogu dodavati anotacija `@XmlTransient` koja označava da se član klase ne uključuje u Xml dokument ili `@XmlElementWrapper` u slučaju kad trebamo definirati da će sljedeći element biti sadržan u drugom elementu.

Isječak koda 7: Primjer JAXB anotacija na klasama

```
@XmlRootElement(name = "AgroMeteoroloskiPodaci")
public class MeteoroloskiPodaciTjedanDana {
    @XmlElement(name = "Naslov")
    public String naslov;

    @XmlElementWrapper(name = "Podaci")
    @XmlElement(name = "Grad")
    public ArrayList<Grad> podaci;
}

class Grad {
    @XmlElement(name = "GradIme")
    String gradIme;

    @XmlElement(name = "Tmax")
    String tMax;

    ...

    @XmlElement(name = "Sunce")
    String sunce;

    ...
}
```

Isječak prikazuje sav kod koji je potreban za parsiranje ili kreiranje XML dokumenta iz objekta `MeteoroloskiPodaciTjedanDana`. Za JAXB dovoljno je postaviti anotacije i odmah imamo mogućnos rada s XML-om.

Isječak koda 8: Pretvaranje XML dokumenta u Java objekte korištenjem JAXB

```
JAXBContext jaxbContext = JAXBContext.newInstance(MeteoroloskiPodaciTjedanDana.class
    );
Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();
MeteoroloskiPodaciTjedanDana meteoroloskiPodaci = (MeteoroloskiPodaciTjedanDana)
    unmarshaller.unmarshal(DemoDatoteke.demoSedmodnevniPodaciXML());
```

Kako bi pretvorili XML dokument u java objekt JAXBContext-u se pridaje klasa izlaznih podataka, kreira se objekt Unmarshaller koji kod korištenja metode `unmarshal` prima XML datoteku i direktno je parsira i mapira u zadani objekt. Suprotno se može postići korištenjem Marshallera koji pretvara postojeći objekt u XML.

4. Zaključak

Popis literature

[1] E. R. Harold i W. S. Means, *XML in a Nutshell*, en, 2. izdanje. Sebastopol, CA: O'Reilly Media, lipanj 2002.

[2] J. P. C. M. Sperberg-McQueen Tim Bray, ur. „Extensible Markup Language (XML) 1.0 (Fifth Edition)." (26. studenoga 2008.), adresa: `https://www.w3.org/TR/xml/`.

[3] E. R. Harold, *Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAX*, en. Boston, MA: Addison Wesley, studeni 2002.

[4] B. McLaughlin, *Java and XML*, en, 2. izdanje. Sebastopol, CA: O'Reilly Media, rujan 2001.

Popis slika