**ЛАБОРАТОРНАЯ РАБОТА**

XML: ОСНОВЫ СОЗДАНИЯ И ОБРАБОТКИ СТРУКТУРИРОВАННЫХ ДАННЫХ

**Цель работы:** освоить принципы создания, валидации и обработки XML документов, развить навыки анализа данных и работы с инструментами для обработки структурированных форматов.

**Класс для работы с XML:**

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

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 java.io.File;

import java.text.DecimalFormat;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

*// 1. Чтение и парсинг XML-файла*

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

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

*// 2. Вывод списка всех книг*

System.out.println("\nСписок всех книг в библиотеке:");

printAllBooks(doc);

*// 3. Вычисление средней цены*

calculateAveragePrice(doc);

*// 4. Фильтрация книг*

System.out.print("\nВведите жанр для фильтрации: ");

String genreFilter = scanner.nextLine();

filterBooksByGenre(doc, genreFilter);

System.out.print("\nВведите год для фильтрации: ");

String yearFilter = scanner.nextLine();

filterBooksByYear(doc, yearFilter);

scanner.close();

} catch (Exception e) {

e.printStackTrace();

}

}

*// Метод для вывода всех книг*

private static void printAllBooks(Document doc) {

NodeList bookList = doc.getElementsByTagName("book");

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

Node bookNode = bookList.item(i);

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

Element bookElement = (Element) bookNode;

System.out.println("\nID: " + bookElement.getAttribute("id"));

System.out.println("Название: " + bookElement.getElementsByTagName("title").item(0).getTextContent());

System.out.println("Автор: " + bookElement.getElementsByTagName("author").item(0).getTextContent());

System.out.println("Год: " + bookElement.getElementsByTagName("year").item(0).getTextContent());

System.out.println("Жанр: " + bookElement.getElementsByTagName("genre").item(0).getTextContent());

System.out.println("Цена: " + bookElement.getElementsByTagName("price").item(0).getTextContent());

}

}

}

*// Метод для вычисления средней цены*

private static void calculateAveragePrice(Document doc) {

NodeList priceList = doc.getElementsByTagName("price");

double sum = 0;

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

sum += Double.parseDouble(priceList.item(i).getTextContent());

}

double average = sum / priceList.getLength();

DecimalFormat df = new DecimalFormat("#.##");

System.out.println("\nСредняя цена книг: " + df.format(average));

}

*// Метод для фильтрации книг по жанру*

private static void filterBooksByGenre(Document doc, String genre) {

NodeList bookList = doc.getElementsByTagName("book");

boolean found = false;

System.out.println("\nКниги в жанре '" + genre + "':");

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

Node bookNode = bookList.item(i);

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

Element bookElement = (Element) bookNode;

String bookGenre = bookElement.getElementsByTagName("genre").item(0).getTextContent();

if (bookGenre.equalsIgnoreCase(genre)) {

found = true;

System.out.println("\nID: " + bookElement.getAttribute("id"));

System.out.println("Название: " + bookElement.getElementsByTagName("title").item(0).getTextContent());

System.out.println("Автор: " + bookElement.getElementsByTagName("author").item(0).getTextContent());

System.out.println("Год: " + bookElement.getElementsByTagName("year").item(0).getTextContent());

System.out.println("Цена: " + bookElement.getElementsByTagName("price").item(0).getTextContent());

}

}

}

if (!found) {

System.out.println("Книг в жанре '" + genre + "' не найдено.");

}

}

*// Метод для фильтрации книг по году*

private static void filterBooksByYear(Document doc, String year) {

NodeList bookList = doc.getElementsByTagName("book");

boolean found = false;

System.out.println("\nКниги, изданные в " + year + " году:");

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

Node bookNode = bookList.item(i);

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

Element bookElement = (Element) bookNode;

String bookYear = bookElement.getElementsByTagName("year").item(0).getTextContent();

if (bookYear.equals(year)) {

found = true;

System.out.println("\nID: " + bookElement.getAttribute("id"));

System.out.println("Название: " + bookElement.getElementsByTagName("title").item(0).getTextContent());

System.out.println("Автор: " + bookElement.getElementsByTagName("author").item(0).getTextContent());

System.out.println("Жанр: " + bookElement.getElementsByTagName("genre").item(0).getTextContent());

System.out.println("Цена: " + bookElement.getElementsByTagName("price").item(0).getTextContent());

}

}

}

if (!found) {

System.out.println("Книг, изданных в " + year + " году, не найдено.");

}

}

}

**Класс для валидации XML по XSD схеме:**

import javax.xml.XMLConstants;

import javax.xml.transform.Source;

import javax.xml.transform.stream.StreamSource;

import javax.xml.validation.Schema;

import javax.xml.validation.SchemaFactory;

import javax.xml.validation.Validator;

import java.io.File;

public class XmlValidator {

public static void main(String[] args) {

try {

*// 1. Создание фабрики для XSD схемы*

SchemaFactory factory =

SchemaFactory.newInstance(XMLConstants.W3C\_XML\_SCHEMA\_NS\_URI);

*// 2. Компиляция схемы*

File schemaFile = new File("library.xsd");

Schema schema = factory.newSchema(schemaFile);

*// 3. Создание валидатора*

Validator validator = schema.newValidator();

*// 4. Валидация XML файла*

Source xmlFile = new StreamSource(new File("library.xml"));

validator.validate(xmlFile);

System.out.println("XML файл валиден по XSD схеме.");

} catch (Exception e) {

System.out.println("Ошибка валидации: " + e.getMessage());

}

}

}

**Файла library.xml:**

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

<library>

<book id="b001">

<title>Война и мир</title>

<author>Лев Толстой</author>

<year>1869</year>

<genre>Роман</genre>

<price>25.99</price>

</book>

<book id="b002">

<title>Преступление и наказание</title>

<author>Фёдор Достоевский</author>

<year>1866</year>

<genre>Психологический роман</genre>

<price>22.50</price>

</book>

<book id="b003">

<title>Мастер и Маргарита</title>

<author>Михаил Булгаков</author>

<year>1967</year>

<genre>Фантастика</genre>

<price>30.00</price>

</book>

</library>

**Файла library.xsd:**

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

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

<xs:element name="library">

<xs:complexType>

<xs:sequence>

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

<xs:complexType>

<xs:sequence>

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

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

<xs:element name="year" type="xs:integer"/>

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

<xs:element name="price" type="xs:decimal"/>

</xs:sequence>

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

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

**Исследование и сравнение:**

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

<rss version="2.0">

<channel>

<title>Новости технологий</title>

<link>https://example.com/tech</link>

<description>Последние новости из мира технологий</description>

<language>ru</language>

<item>

<title>Новый смартфон выходит на рынок</title>

<link>https://example.com/tech/123</link>

<description>Компания X представила новый флагманский смартфон...</description>

<pubDate>Wed, 15 Mar 2023 12:00:00 GMT</pubDate>

<guid>https://example.com/tech/123</guid>

</item>

<item>

<title>Искусственный интеллект в медицине</title>

<link>https://example.com/tech/124</link>

<description>Новые разработки в области ИИ помогают диагностировать...</description>

<pubDate>Tue, 14 Mar 2023 09:30:00 GMT</pubDate>

<guid>https://example.com/tech/124</guid>

</item>

</channel>

</rss>

**Анализ структуры:**

1. Документ начинается с объявления XML и корневого элемента <rss> с атрибутом версии.
2. Основной контейнер - <channel>, содержащий метаданные (название, ссылка, описание, язык).
3. Новости представлены элементами <item>, каждый из которых содержит заголовок, ссылку, описание, дату публикации и уникальный идентификатор.
4. Вложенность: rss → channel → item (множество элементов).
5. Атрибуты используются только для указания версии RSS.
6. Документ использует стандартные теги RSS 2.0.
7. Предназначение: распространение новостей в стандартизированном формате для агрегаторов и читателей RSS.

В ходе лабораторной работы были освоены принципы создания и валидации XML-документов, разработана XSD-схема для проверки структуры данных, а также написана программа для обработки XML на Python. Полученные навыки позволяют эффективно работать со структурированными данными в формате XML, что важно для обмена информацией между различными системами.