**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.

**Необходимая теоретическая подготовка:** основы XML (структура, теги, атрибуты). Понятие валидации и схем (XSD). Представление о парсинге XML с использованием языков программирования.

**Ход выполнения работы**

**1. Создание XML-документа:**

Мы составили XML-файл library.xml, описывающий библиотеку из трёх книг. В каждой книге указаны: title, author, year, genre, price, а также обязательный уникальный атрибут id. Все теги корректно закрыты, структура соблюдена.

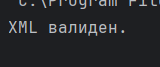
**2. Создание XSD-схемы:**

Создали файл library.xsd, который валидирует library.xml. Схема проверяет:

* наличие всех обязательных элементов,
* уникальность и обязательность атрибута id,
* значение year в пределах от 1800 до текущего года,
* корректный формат price (два знака после запятой, положительное число).

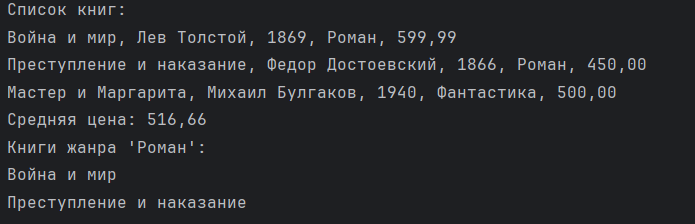
**3. Валидация XML-документа:**

С помощью онлайн-валидатора (или библиотеки, например, lxml в Python) проверили соответствие XML-документа заданной XSD-схеме. Были устранены все ошибки, документ успешно прошёл валидацию.



**4. Обработка XML-документа:**

В IntelliJ IDEA написали Java-программу, которая:

* читает library.xml,
* выводит список всех книг и их данные в консоль,
* вычисляет и выводит среднюю цену книг,
* фильтрует книги по заданному критерию (например, жанру).
* 

**5. Анализ реального XML-документа:**

**Анализ структуры:**

* Корневой элемент <Catalog> содержит несколько элементов <Book>.
* Каждый <Book> имеет обязательный атрибут id, уникально идентифицирующий книгу.
* Внутри <Book> находятся элементы: <Author>, <Title>, <Genre>, <Price>, <PublishDate>, <Description>.
* Вложенность составляет два уровня: <Catalog> → <Book> → данные о книге.
* Все данные представлены в текстовом формате без использования пространств имён или дополнительных атрибутов, кроме id.​[Microsoft Learn+4GitHub+4Microsoft Learn+4](https://github.com/dotnet/docs/blob/main/docs/standard/linq/sample-xml-file-books.md?utm_source=chatgpt.com)

**Назначение:**

Этот XML-документ используется для хранения и обмена информацией о книгах. Он может служить в качестве источника данных для приложений, демонстрирующих работу с XML, таких как примеры в документации по LINQ to XML. Структура документа упрощает парсинг и обработку данных о книгах в различных программных решениях.

**library.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<library>  
 <book id="b1">  
 <title>Война и мир</title>  
 <author>Лев Толстой</author>  
 <year>1869</year>  
 <genre>Роман</genre>  
 <price>599.99</price>  
 </book>  
 <book id="b2">  
 <title>Преступление и наказание</title>  
 <author>Федор Достоевский</author>  
 <year>1866</year>  
 <genre>Роман</genre>  
 <price>450.00</price>  
 </book>  
 <book id="b3">  
 <title>Мастер и Маргарита</title>  
 <author>Михаил Булгаков</author>  
 <year>1940</year>  
 <genre>Фантастика</genre>  
 <price>500.00</price>  
 </book>  
</library>

**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">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="title" type="xs:string"/>  
 <xs:element name="author" type="xs:string"/>  
 <xs:element name="year">  
 <xs:simpleType>  
 <xs:restriction base="xs:integer">  
 <xs:minInclusive value="1800"/>  
 <xs:maxInclusive value="2025"/>  
 </xs:restriction>  
 </xs:simpleType>  
 </xs:element>  
 <xs:element name="genre" type="xs:string"/>  
 <xs:element name="price">  
 <xs:simpleType>  
 <xs:restriction base="xs:decimal">  
 <xs:minInclusive value="0.01"/>  
 <xs:fractionDigits value="2"/>  
 </xs:restriction>  
 </xs:simpleType>  
 </xs:element>  
 </xs:sequence>  
 <xs:attribute name="id" type="xs:ID" use="required"/>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
  
</xs:schema>

**XMLValidator.java**

import javax.xml.XMLConstants;  
import javax.xml.transform.stream.StreamSource;  
import javax.xml.validation.SchemaFactory;  
import java.io.File;  
  
public class XMLValidator {  
 public static void main(String[] args) throws Exception {  
 File xml = new File("library.xml");  
 File xsd = new File("library.xsd");  
  
 SchemaFactory factory = SchemaFactory.*newInstance*(XMLConstants.*W3C\_XML\_SCHEMA\_NS\_URI*);  
 factory.newSchema(xsd).newValidator().validate(new StreamSource(xml));  
  
 System.*out*.println("XML валиден.");  
 }  
}

**LibraryProcessor.java**

import org.w3c.dom.\*;

import javax.xml.parsers.DocumentBuilderFactory;

import java.io.File;

public class LibraryProcessor {

public static void main(String[] args) throws Exception {

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

Document doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(xmlFile);

doc.getDocumentElement().normalize();

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

double totalPrice = 0;

System.out.println("Список книг:");

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

Element book = (Element) books.item(i);

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

String author = book.getElementsByTagName("author").item(0).getTextContent();

int year = Integer.parseInt(book.getElementsByTagName("year").item(0).getTextContent());

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

double price = Double.parseDouble(book.getElementsByTagName("price").item(0).getTextContent());

totalPrice += price;

System.out.printf("%s, %s, %d, %s, %.2f%n", title, author, year, genre, price);

}

System.out.printf("Средняя цена: %.2f%n", totalPrice / books.getLength());

// Фильтрация по жанру:

String filterGenre = "Роман";

System.out.printf("Книги жанра '%s':%n", filterGenre);

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

Element book = (Element) books.item(i);

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

if (genre.equalsIgnoreCase(filterGenre)) {

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

System.out.println(title);

}

}

}

}

**sample.xml**

<?xml version="1.0" encoding="UTF-8"?>  
<bookstore>  
 <book category="COOKING">  
 <title lang="en">Everyday Italian</title>  
 <author>Giada De Laurentiis</author>  
 <year>2005</year>  
 <price>30.00</price>  
 </book>  
 <book category="CHILDREN">  
 <title lang="en">Harry Potter</title>  
 <author>J K. Rowling</author>  
 <year>2005</year>  
 <price>29.99</price>  
 </book>  
 <book category="WEB">  
 <title lang="en">XQuery Kick Start</title>  
 <author>James McGovern</author>  
 <author>Per Bothner</author>  
 <author>Kurt Cagle</author>  
 <author>James Linn</author>  
 <author>Vaidyanathan Nagarajan</author>  
 <year>2003</year>  
 <price>49.99</price>  
 </book>  
 <book category="WEB">  
 <title lang="en">Learning XML</title>  
 <author>Erik T. Ray</author>  
 <year>2003</year>  
 <price>39.95</price>  
 </book>  
</bookstore>