 Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики

Кафедра автоматики и процессов управления

**ОТЧЕТ**

по лабораторной работе №4

по дисциплине «СМиСПИС»

**«Технология XML связывания данных с помощью JAXB»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 5371 |  | Ильясов Е. |
| Студентка гр. 5371 |  | Есенбаев Ч. |
| Студент гр. 5371 |  | Нургазы Б. |
| Преподаватель |  | Кораблев Ю.А. |

Санкт-Петербург

2020

1. **Цель работы**

Изучить технологию XML связывания данных с помощью API связывания – JAXB.

1. **Задание на лабораторную работу №3. Вариант №3.**

Создать приложение, которое читает данные списка книг библиотеки из XML, а также записывает данные обратно в XML.

1. **Выполнение лабораторной работы**

**Настройка сборщика Maven**

Так как на нашем компьютере установлена Open JDK 8, а библиотека JAXB была исключена из JDK дистрибутивов начиная с JDK версии 6 и мы используем сборщик Maven, то для начала необходимо добавить Maven () зависимости на эту библиотеку.

Пропишем в pom.xml – конфигурационный файл сборщика Maven зависимости на JAXB:

<dependency>

<groupId>javax.xml.bind</groupId>

<artifactId>jaxb-api</artifactId>

<version2.3.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.jaxb</groupId>

<artifactId>jaxb-runtime</artifactId>

<version>2.3.1</version>

</dependency>

Так же добавим зависимости на библиотеку для Unit тестирования приложений:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<scope>test</scope>

<version>${junit.jupiter.version}</version>

</dependency>

Добавим плагины к сборщику Maven, которые будут автоматически запускать наши тестовые методы, помеченные аннотацией @Test при сборке приложения:

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

<plugin>

<artifactId>maven-failsafe-plugin</artifactId>

<version>2.22.2</version>

</plugin>

**Написание DTD документа валидации XML**

Был написан DTD документ telephone\_book.dtd валидации нашего XML файла:

<!DOCTYPE library [  
 <!ELEMENT bookInfo EMPTY>  
 <!ELEMENT author EMPTY>  
 <!ELEMENT book (author, bookInfo)>  
 <!ELEMENT library (book\*)>  
  
 <!ATTLIST bookInfo  
 bookName CDATA #REQUIRED  
 graduationYear CDATA #REQUIRED  
 >  
  
 <!ATTLIST author  
 firstName CDATA #REQUIRED  
 lastName CDATA #REQUIRED  
 >  
 ]>

**Написание XSD схемы валидации XML**

Был написана схема telephone\_book\_scheme.xsd валидации нашего XML файла:

<?xml version="1.0"?>  
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="https://github.com/Hitsugaya7/java-labs/lab4">  
 <element name="library">  
 <complexType>  
 <sequence>  
 <element name="book" maxOccurs="unbounded" minOccurs="0">  
 <complexType>  
 <sequence>  
 <element name="author">  
 <complexType>  
 <attribute name="firstName" type="string" use="required"/>  
 <attribute name="lastName" type="string" use="required"/>  
 </complexType>  
 </element>  
 <element name="bookInfo">  
 <complexType>  
 <attribute name="bookName" type="string" use="required"/>  
 <attribute name="graduationYear" type="string" use="required"/>  
 </complexType>  
 </element>  
 </sequence>  
 </complexType>  
 </element>  
 </sequence>  
 </complexType>  
 </element>  
</schema>

**Вид валидного XML документа**

Ниже представлен пример валидного XML документа, содержащего данные телефонного справочника:

<?xml version="1.0" encoding="UTF-8"?>  
<directory:library xmlns:directory="https://github.com/Hitsugaya7/java-labs/lab4"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="https://github.com/Hitsugaya7/java-labs/lab4 ">  
 <book>  
 <author firstName="Yerdos" lastName="Ilyassov"/>  
 <bookInfo bookName="The Lord of the Rings: chapter one" graduationYear="1997"/>  
 </book>  
 <book>  
 <author firstName="Chingis" lastName="Yessenbayev"/>  
 <bookInfo bookName="The Lord of the Rings: chapter two" graduationYear="1996"/>  
 </book>  
 <book>  
 <author firstName="Baurzhan" lastName="Nurgazy"/>  
 <bookInfo bookName="The Lord of the Rings: chapter three" graduationYear="1998"/>  
 </book>  
</directory:library>

**Разработка классов для отображения XML документа**

UML диаграмма разработанных классов изображена на рис.1

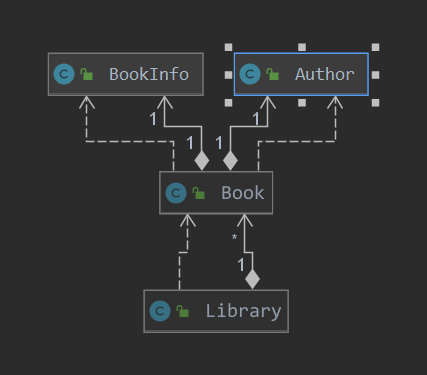


Рисунок 1 – UML диаграмма классов для отображения XML документа

**Тестирование маршаллизации и демаршаллизации**

Для тестирования маршаллизации и демаршаллизации был написан класс MarshallingDemarshallingTest, который содержит два метода, проаннотированные аннотацией @Test:

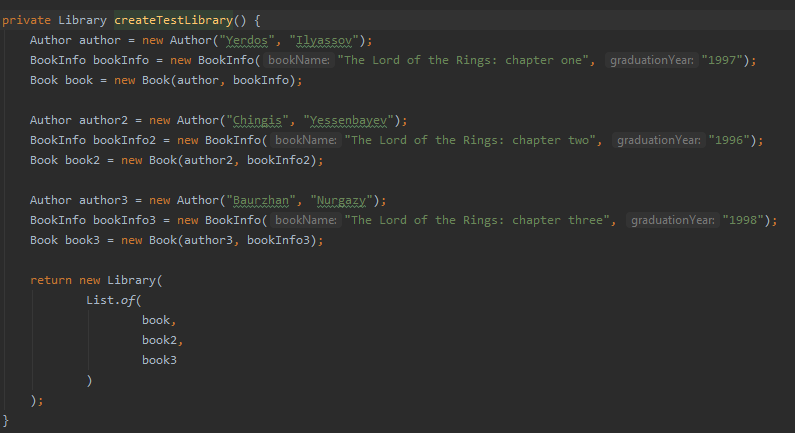
marshallingTest() - для тестирования маршаллизации. Метод создает предзаполненный данными класс TelephoneDirectory, затем маршаллизует его и сравнивает полученную строку в формета XML с эталонным значением из прочитанного файла library\_test.xml.

demarshallingTest – для тестирования демаршаллизации. Метод читает XML документ library\_test.xml, демаршаллизует его в объект TelephoneDirectory и сравнивает с эталонным объектом TelephoneDirectory, который создается программно в тестовом методе.

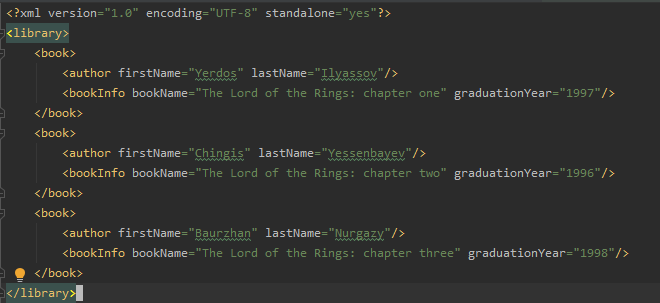
1. **Пример работы программы**

1. Тестирование маршаллизации

Создаем три объекта Author, BookInfo, Book. Формируем из них список и создаем объект Library:

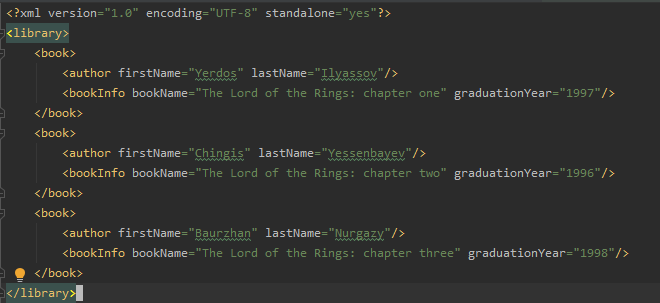


После выполнения маршаллизации получаем следующий результат:



2. Тестирование демаршаллизации

Создаем xml-файл со следующим содержанием:



После выполнения демаршаллизации получаем следующий результат:



1. **Исходный код**

**Класс Author**

package com.dto.information;  
  
import javax.xml.bind.annotation.XmlAttribute;  
import javax.xml.bind.annotation.XmlType;  
import java.util.Objects;  
  
@XmlType(name = "author")  
public class Author {  
  
 private String firstName;  
  
 private String lastName;  
  
 public Author() {  
 }  
  
 public Author(final String first, final String last) {  
 this.firstName = first;  
 this.lastName = last;  
 }  
  
 @XmlAttribute(name = "firstName")  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(final String firstName) {  
 this.firstName = firstName;  
 }  
  
 @XmlAttribute(name = "lastName")  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(final String lastName) {  
 this.lastName = lastName;  
 }  
  
 @Override  
 public boolean equals(final Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 final Author name = (Author) o;  
 return Objects.*equals*(firstName, name.firstName) &&  
 Objects.*equals*(lastName, name.lastName);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(firstName, lastName);  
 }  
}

**Класс BookInfo**

package com.dto.information;  
  
  
import javax.xml.bind.annotation.XmlAttribute;  
import javax.xml.bind.annotation.XmlType;  
import java.util.Objects;  
  
@XmlType(name = "bookInfo")  
public class BookInfo {  
  
 private String bookName;  
  
 private String graduationYear;  
  
 public BookInfo() {  
 }  
  
 public BookInfo(final String bookName, final String graduationYear) {  
 this.bookName = bookName;  
 this.graduationYear = graduationYear;  
 }  
  
 @XmlAttribute(name = "bookName")  
 public String getBookName() {  
 return bookName;  
 }  
  
 public void setBookName(final String bookName) {  
 this.bookName = bookName;  
 }  
  
 @XmlAttribute(name = "graduationYear")  
 public String getGraduationYear() {  
 return graduationYear;  
 }  
  
 public void setGraduationYear(final String graduationYear) {  
 this.graduationYear = graduationYear;  
 }  
  
 @Override  
 public boolean equals(final Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 final BookInfo number = (BookInfo) o;  
 return Objects.*equals*(bookName, number.bookName) &&  
 Objects.*equals*(graduationYear, number.graduationYear);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(bookName, graduationYear);  
 }  
}

**Класс Book**

package com.dto;  
  
import com.dto.information.Author;  
import com.dto.information.BookInfo;  
  
import javax.xml.bind.annotation.XmlType;  
import java.util.Objects;  
  
@XmlType(name = "Book")  
public class Book {  
  
 private Author author;  
  
 private BookInfo bookInfo;  
  
  
 public Book() {  
 }  
  
 public Book(final Author author, final BookInfo bookInfo) {  
 this.author = author;  
 this.bookInfo = bookInfo;  
 }  
  
 public Author getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(final Author author) {  
 this.author = author;  
 }  
  
 public BookInfo getBookInfo() {  
 return bookInfo;  
 }  
  
 public void setBookInfo(final BookInfo number) {  
 this.bookInfo = number;  
 }  
  
  
 @Override  
 public boolean equals(final Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 final Book contact = (Book) o;  
 return Objects.*equals*(author, contact.author) &&  
 Objects.*equals*(bookInfo, contact.bookInfo);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(author, bookInfo);  
 }  
}

**Класс Library**

package com.dto;  
  
import javax.xml.bind.annotation.XmlRootElement;  
import java.util.List;  
import java.util.Objects;  
  
@XmlRootElement(name = "library")  
public class Library {  
  
 private List<Book> book;  
  
 public Library() {  
 }  
  
 public Library(final List<Book> books) {  
 this.book = books;  
 }  
  
 public List<Book> getBook() {  
 return book;  
 }  
  
 public void setBook(final List<Book> book) {  
 this.book = book;  
 }  
  
 @Override  
 public boolean equals(final Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 final Library that = (Library) o;  
 return Objects.*equals*(book, that.book);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(book);  
 }  
}

**Класс MarshallingDemarshallingTest**

package com.dto;  
  
import com.dto.information.Author;  
import com.dto.information.BookInfo;  
import org.junit.Assert;  
import org.junit.jupiter.api.Test;  
  
import javax.xml.bind.JAXBContext;  
import javax.xml.bind.JAXBException;  
import javax.xml.bind.Marshaller;  
import javax.xml.bind.Unmarshaller;  
import java.io.File;  
import java.io.IOException;  
import java.io.StringWriter;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class MarshallingDemarshallingTest {  
  
 @Test  
 public void marshallingTest() throws JAXBException, IOException {  
 Library telephoneDirectory = createTestLibrary();  
  
 JAXBContext context = JAXBContext.*newInstance*(Library.class);  
 Marshaller marshaller = context.createMarshaller();  
 marshaller.setProperty(Marshaller.*JAXB\_FORMATTED\_OUTPUT*, Boolean.*TRUE*);  
  
 StringWriter writer = new StringWriter();  
 marshaller.marshal(telephoneDirectory, writer);  
  
 List<String> demarshallingResultByLine = Arrays.*stream*(writer.toString().split("\n"))  
 .map(line -> line.trim())  
 .collect(Collectors.*toList*());  
  
 Path pathToTestFile = Paths.*get*("src/test/resources/data/library\_test.xml");  
 List<String> expectedValue = Files.*readAllLines*(pathToTestFile)  
 .stream()  
 .map(line -> line.trim())  
 .collect(Collectors.*toList*());  
  
 Assert.*assertEquals*(expectedValue, demarshallingResultByLine);  
 }  
  
 @Test  
 public void demarshallingTest() throws JAXBException, IOException {  
 File file = new File("src/test/resources/data/library\_test.xml");  
 JAXBContext jaxbContext = JAXBContext.*newInstance*(Library.class);  
  
 Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();  
 Library actualTelephoneDirectory = (Library) jaxbUnmarshaller.unmarshal(file);  
 Library expectedTelephoneDirectory = createTestLibrary();  
  
 Assert.*assertEquals*(expectedTelephoneDirectory, actualTelephoneDirectory);  
 }  
  
 private Library createTestLibrary() {  
 Author author = new Author("Yerdos", "Ilyassov");  
 BookInfo bookInfo = new BookInfo("The Lord of the Rings: chapter one", "1997");  
 Book book = new Book(author, bookInfo);  
  
 Author author2 = new Author("Chingis", "Yessenbayev");  
 BookInfo bookInfo2 = new BookInfo("The Lord of the Rings: chapter two", "1996");  
 Book book2 = new Book(author2, bookInfo2);  
  
 Author author3 = new Author("Baurzhan", "Nurgazy");  
 BookInfo bookInfo3 = new BookInfo("The Lord of the Rings: chapter three", "1998");  
 Book book3 = new Book(author3, bookInfo3);  
  
 return new Library(  
 List.*of*(  
 book,  
 book2,  
 book3  
 )  
 );  
 }  
}

1. **Вывод**

В процессе выполнения лабораторной работыбыл изучен формат данных XML, а так же способы его валидации. Изучены маршлаллизация и демаршаллизация Java объектов с помощью библиотеки JAXB.

1. **Зачетное задание**

Расширить XML-файл списка книг из библиотеки тегами, дающими возможность отслеживать книги из библиотеки, взятые конкретным читателем.

Добавлен новый класс:

package com.dto.information;  
  
import javax.xml.bind.annotation.XmlAttribute;  
import java.util.Objects;  
  
public class Reader {  
  
 private String firstName;  
  
 private String lastName;  
  
 public Reader() {  
 }  
  
 public Reader(final String first, final String last) {  
 this.firstName = first;  
 this.lastName = last;  
 }  
  
 @XmlAttribute(name = "firstName")  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(final String firstName) {  
 this.firstName = firstName;  
 }  
  
 @XmlAttribute(name = "lastName")  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(final String lastName) {  
 this.lastName = lastName;  
 }  
  
 @Override  
 public boolean equals(final Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 final Reader name = (Reader) o;  
 return Objects.*equals*(firstName, name.firstName) &&  
 Objects.*equals*(lastName, name.lastName);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(firstName, lastName);  
 }  
}

Был модифицирован класс Book:

package com.dto;  
  
import com.dto.information.Author;  
import com.dto.information.BookInfo;  
import com.dto.information.Reader;  
  
import javax.xml.bind.annotation.XmlType;  
import java.util.Objects;  
  
@XmlType(name = "Book")  
public class Book {  
  
 private Author author;  
  
 private BookInfo bookInfo;  
  
 private Reader reader;  
  
  
 public Book() {  
 }  
  
 public Book(final Author author, final BookInfo bookInfo) {  
 this.author = author;  
 this.bookInfo = bookInfo;  
 }  
  
 public Book(Author author, BookInfo bookInfo, Reader reader) {  
 this.author = author;  
 this.bookInfo = bookInfo;  
 this.reader = reader;  
 }  
  
 public Author getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(final Author author) {  
 this.author = author;  
 }  
  
 public BookInfo getBookInfo() {  
 return bookInfo;  
 }  
  
 public void setBookInfo(final BookInfo number) {  
 this.bookInfo = number;  
 }  
  
 public Reader getReader() {  
 return reader;  
 }  
  
 public void setReader(Reader reader) {  
 this.reader = reader;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Book)) return false;  
 Book book = (Book) o;  
 return Objects.*equals*(getAuthor(), book.getAuthor()) &&  
 Objects.*equals*(getBookInfo(), book.getBookInfo()) &&  
 Objects.*equals*(reader, book.reader);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(getAuthor(), getBookInfo(), reader);  
 }  
}

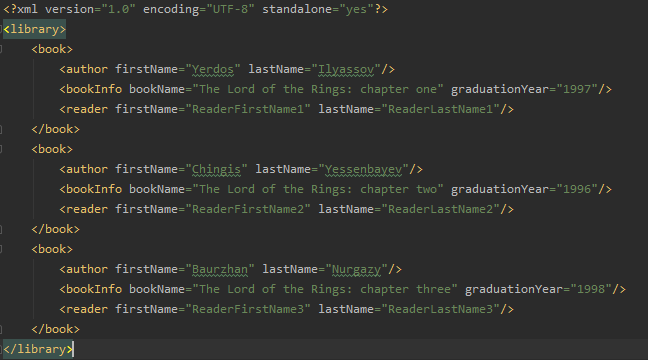
Результаты работы программы:

1. Тестирование маршаллизации

Создаем три объекта Author, BookInfo,Reader, Book. Формируем из них список и создаем объект Library:

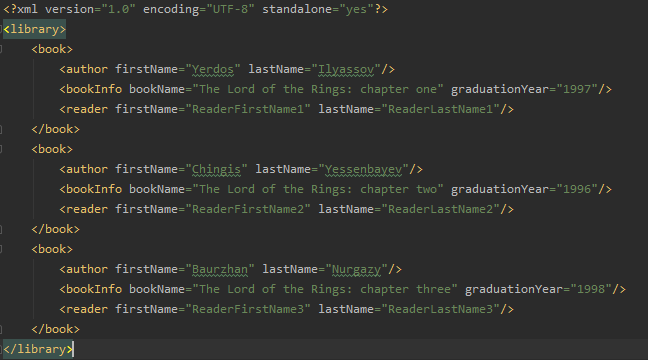


После выполнения маршаллизации получаем следующий результат:



2. Тестирование демаршаллизации

Создаем xml-файл со следующим содержанием:



После выполнения демаршаллизации получаем следующий результат:

