Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Лабораторная работа №5

«Паттерны проектирования»

Выполнил: Пристром Р.М.

Студент группы № 814301

Проверил: Лыщик А.П.

Минск 2020

**Цель**: ознакомиться и освоить практические навыки использования паттернов проектирования.

**Задание**: проектировать (а в идеале – реализовать) каркас для обработки данных из XML файла (использовать наработки ЛР №4). В программе должна присутствовать обработка входящих данных, это может быть подсчёт статистики, вывод графиков, объединение в группы и другое в зависимости от темы. Тема – интернет магазин для компьютерной техники.

Наша программа считывает XML-файл со списком книг и печатает их свойства. Сам XML-файл выглядит вот так:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <?xml version="1.0" encoding="UTF-8"?>  <BookCatalogue>      <Book>          <Title>Yogasana Vijnana: the Science of Yoga</Title>          <Author>Dhirendra Brahmachari</Author>          <Date>1966</Date>          <ISBN>81-40-34319-4</ISBN>          <Publisher>Dhirendra Yoga Publications</Publisher>          <Cost currency="INR">11.50</Cost>      </Book>      <Book>          <Title>The First and Last Freedom</Title>          <Author>J. Krishnamurti</Author>          <Date>1954</Date>          <ISBN>0-06-064831-7</ISBN>          <Publisher>Harper &amp; Row</Publisher>          <Cost currency="USD">2.95</Cost>      </Book>  </BookCatalogue> |

Cама программа для разбора и печати:

package edu.javacourse.xml;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

public class DomExample {

    public static void main(String[] args) {

        try {

            // Создается построитель документа

            DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();

            // Создается дерево DOM документа из файла

            Document document = documentBuilder.parse("BookCatalog.xml");

            // Получаем корневой элемент

            Node root = document.getDocumentElement();

            System.out.println("List of books:");

            System.out.println();

            // Просматриваем все подэлементы корневого - т.е. книги

            NodeList books = root.getChildNodes();

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

                Node book = books.item(i);

                // Если нода не текст, то это книга - заходим внутрь

                if (book.getNodeType() != Node.TEXT\_NODE) {

                    NodeList bookProps = book.getChildNodes();

                    for(int j = 0; j < bookProps.getLength(); j++) {

                        Node bookProp = bookProps.item(j);

                        // Если нода не текст, то это один из параметров книги - печатаем

                        if (bookProp.getNodeType() != Node.TEXT\_NODE) {

                            System.out.println(bookProp.getNodeName() + ":" + bookProp.getChildNodes().item(0).getTextContent());

                        }

                    }

                    System.out.println("===========>>>>");

                }

            }

        } catch (ParserConfigurationException ex) {

            ex.printStackTrace(System.out);

        } catch (SAXException ex) {

            ex.printStackTrace(System.out);

        } catch (IOException ex) {

            ex.printStackTrace(System.out);

        }

    }

}

Предлагаю пример, в котором мы считаем наш XML-файл с книгами и добавим еще одну книгу в нашу структуру. После этого мы сохраним XML в файл. Самым важным тут будет метод **addNewBook** — именно в нем мы создаем унифицированные объекты, которые потом вставляются в наше дерево. Также мы используем наследника класса **Node** — класс **Element**. Этот класс предназначен именно для тэгов. Ему можно устанавливать имя и атрибуты.

package edu.javacourse.xml;

import java.io.FileOutputStream;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.Transformer;

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.DOMException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.xml.sax.SAXException;

public class DomExample2 {

    public static void main(String[] args) {

        try {

            // Создается построитель документа

            DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();

            // Создается дерево DOM документа из файла

            Document document = documentBuilder.parse("BookCatalog.xml");

            // Вызываем метод для добавления новой книги

            addNewBook(document);

        } catch (ParserConfigurationException ex) {

            ex.printStackTrace(System.out);

        } catch (SAXException ex) {

            ex.printStackTrace(System.out);

        } catch (IOException ex) {

            ex.printStackTrace(System.out);

        }

    }

    // Функция добавления новой книги и записи результата в файл

    private static void addNewBook(Document document) throws TransformerFactoryConfigurationError, DOMException {

        // Получаем корневой элемент

        Node root = document.getDocumentElement();

        // Создаем новую книгу по элементам

        // Сама книга <Book>

        Element book = document.createElement("Book");

        // <Title>

        Element title = document.createElement("Title");

        // Устанавливаем значение текста внутри тега

        title.setTextContent("Incredible book about Java");

        // <Author>

        Element author = document.createElement("Author");

        author.setTextContent("Saburov Anton");

        // <Date>

        Element date = document.createElement("Date");

        date.setTextContent("2015");

        // <ISBN>

        Element isbn = document.createElement("ISBN");

        isbn.setTextContent("0-06-999999-9");

        // <Publisher>

        Element publisher = document.createElement("Publisher");

        publisher.setTextContent("Java-Course publisher");

        // <Cost>

        Element cost = document.createElement("Cost");

        cost.setTextContent("499");

        // Устанавливаем атрибут

        cost.setAttribute("currency", "RUB");

        // Добавляем внутренние элементы книги в элемент <Book>

        book.appendChild(title);

        book.appendChild(author);

        book.appendChild(date);

        book.appendChild(isbn);

        book.appendChild(publisher);

        book.appendChild(cost);

        // Добавляем книгу в корневой элемент

        root.appendChild(book);

        // Записываем XML в файл

        writeDocument(document);

    }

    // Функция для сохранения DOM в файл

    private static void writeDocument(Document document) throws TransformerFactoryConfigurationError {

        try {

            Transformer tr = TransformerFactory.newInstance().newTransformer();

            DOMSource source = new DOMSource(document);

            FileOutputStream fos = new FileOutputStream("other.xml");

            StreamResult result = new StreamResult(fos);

            tr.transform(source, result);

        } catch (TransformerException | IOException e) {

            e.printStackTrace(System.out);

        }

    }

}

Поиск книги по определенным критениям:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120 | package edu.javacourse.xml;    import java.io.IOException;  import javax.xml.parsers.DocumentBuilder;  import javax.xml.parsers.DocumentBuilderFactory;  import javax.xml.parsers.ParserConfigurationException;  import javax.xml.xpath.XPath;  import javax.xml.xpath.XPathConstants;  import javax.xml.xpath.XPathExpression;  import javax.xml.xpath.XPathExpressionException;  import javax.xml.xpath.XPathFactory;  import org.w3c.dom.DOMException;  import org.w3c.dom.Document;  import org.w3c.dom.Node;  import org.w3c.dom.NodeList;  import org.xml.sax.SAXException;    public class XPathExample {        public static void main(String[] args) {          try {              DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();              Document document = documentBuilder.parse("BookCatalog.xml");                printCost(document);              printCost2(document);              printCost3(document);              printCost4(document);              printCost5(document);            } catch (XPathExpressionException | ParserConfigurationException | SAXException | IOException ex) {              ex.printStackTrace(System.out);          }      }        // Печать всех элементов Cost      private static void printCost(Document document) throws DOMException, XPathExpressionException {          System.out.println("Example 1 - Печать всех элементов Cost");          XPathFactory pathFactory = XPathFactory.newInstance();          XPath xpath = pathFactory.newXPath();            // Пример записи XPath          // Подный путь до элемента          //XPathExpression expr = xpath.compile("BookCatalogue/Book/Cost");          // Все элементы с таким именем          //XPathExpression expr = xpath.compile("//Cost");          // Элементы, вложенные в другой элемент          XPathExpression expr = xpath.compile("//Book/Cost");            NodeList nodes = (NodeList) expr.evaluate(document, XPathConstants.NODESET);          for (int i = 0; i < nodes.getLength(); i++) {              Node n = nodes.item(i);              System.out.println("Value:" + n.getTextContent());          }          System.out.println();      }        // Печать элемента Cost у которого атрибут currency='USD'      private static void printCost2(Document document) throws DOMException, XPathExpressionException {          System.out.println("Example 2 - Печать элемента Cost у которого атрибут currency='USD'");          XPathFactory pathFactory = XPathFactory.newInstance();          XPath xpath = pathFactory.newXPath();          XPathExpression expr = xpath.compile("BookCatalogue/Book/Cost[@currency='USD']");          NodeList nodes = (NodeList) expr.evaluate(document, XPathConstants.NODESET);          for (int i = 0; i < nodes.getLength(); i++) {              Node n = nodes.item(i);              System.out.println("Value:" + n.getTextContent());          }          System.out.println();      }        // Печать элементов Book у которых значение Cost > 4      private static void printCost3(Document document) throws DOMException, XPathExpressionException {          System.out.println("Example 3 - Печать элементов Book у которых значение Cost > 4");          XPathFactory pathFactory = XPathFactory.newInstance();          XPath xpath = pathFactory.newXPath();          XPathExpression expr = xpath.compile("BookCatalogue/Book[Cost>4]");          NodeList nodes = (NodeList) expr.evaluate(document, XPathConstants.NODESET);          for (int i = 0; i < nodes.getLength(); i++) {              Node n = nodes.item(i);              System.out.println("Value:" + n.getTextContent());          }          System.out.println();      }        // Печать первого элемента Book      private static void printCost4(Document document) throws DOMException, XPathExpressionException {          System.out.println("Example 4 - Печать первого элемента Book");          XPathFactory pathFactory = XPathFactory.newInstance();          XPath xpath = pathFactory.newXPath();          XPathExpression expr = xpath.compile("BookCatalogue/Book[2]");          NodeList nodes = (NodeList) expr.evaluate(document, XPathConstants.NODESET);          for (int i = 0; i < nodes.getLength(); i++) {              Node n = nodes.item(i);              System.out.println("Value:" + n.getTextContent());          }          System.out.println();      }        // Печать цены книги у которой Title начинается с Yogasana      // Варианты доступа к относительным узлам:      // ancestor , ancestor-or-self, descendant, descendant-or-self      // following, following-sibling, namespace, preceding, preceding-sibling      private static void printCost5(Document document) throws DOMException, XPathExpressionException {          System.out.println("Example 5 - Печать цены книги у которой Title начинается с 'Yogasana'");          XPathFactory pathFactory = XPathFactory.newInstance();          XPath xpath = pathFactory.newXPath();          XPathExpression expr = xpath.compile("BookCatalogue/Book/Cost"                  + "[starts-with(preceding-sibling::Title, 'Yogasana')"                  + " or "                  + "starts-with(following-sibling::Title, 'Yogasana')]");          NodeList nodes = (NodeList) expr.evaluate(document, XPathConstants.NODESET);          for (int i = 0; i < nodes.getLength(); i++) {              Node n = nodes.item(i);              System.out.println("Value:" + n.getTextContent());          }          System.out.println();      }    } |