Задание за решения на задачи по Упражнение - Модул 6: DOM, SAX и StAX

Изготвил: Марина Дабова, ФН: 62503, група 3, специалност СИ

Коментарите са обозначени с червен цвят

#### **Задача 1: Като използвате SAX, принтирайте дадения по-долу XML документ в стандартната конзола, спазвайки следните условия:**

1. Йерархията на елементите да бъде запазена
2. Всеки елемент да бъде разпечатан на нов ред заедно с включените в него атрибути
3. Текстовите стойности на елементите да бъдат разпечатани с главни букви

import org.xml.sax.InputSource;

import org.xml.sax.XMLReader;

import org.xml.sax.helpers.XMLReaderFactory;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.StringReader;

import org.xml.sax.Attributes;

import org.xml.sax.ContentHandler;

import org.xml.sax.Locator;

import org.xml.sax.SAXException;

public class Main {

public static void main(String[] args) {

OutputStreamWriter outputStreamWriter = new OutputStreamWriter(System.out);

try {

XMLReader parser = XMLReaderFactory.createXMLReader();

InputSource source = new InputSource("rss.xml");

//InputSource source = new InputSource(new StringReader("")); //for using in https://ideone.com/ instead new InputSource("rss.xml")

parser.setContentHandler(new SAXHandler(outputStreamWriter)

);

parser.parse(source);

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

outputStreamWriter.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

class SAXHandler implements ContentHandler {

Locator locator;

Integer indent;

OutputStreamWriter outputStreamWriter;

private final Integer TAB\_SIZE = 4;

public SAXHandler(OutputStreamWriter outputStreamWriter) {

this.outputStreamWriter = outputStreamWriter;

indent = 0;

}

@Override

public void setDocumentLocator(Locator locator) {

this.locator = locator;

}

@Override

public void startDocument() throws SAXException {

printIndented("<?xml version=\"1.0\" encoding=\"UTF-8\"?>", false, false);

}

@Override

public void endDocument() throws SAXException {

// ...

}

@Override

public void startElement(String uri, String localName, String qName, Attributes atts) throws SAXException {

printIndented(String.format("<%s", qName), true, false);

printAttributes(atts);

printIndented(">\r\n", false, true);

++indent;

}

@Override

public void endElement(String uri, String localName, String qName) throws SAXException {

--indent;

printIndented(String.format("</%s>", qName), true, false);

}

@Override

public void characters(char[] chars, int start, int length) throws SAXException {

String newStr = new String(chars, start, length).toUpperCase().trim();

int size=newStr.length();

if (size > 0) {

printIndented(newStr, false, false);

}

/\*За да разпечатаме текстовите стойности на елементите с главни букви трябва да допишем тази функция characters. За целта си слагаме подадения масив в стринга newStr,като подаваме и сътветно подаденото начало и дължина.Чрез toUpperCase() го правим с главни букви, а .trim() е за да премахнем whitespace-овете. Ако size, което е равно на дължитаната е по-голямо от 0, то принтираме. \*/

}

@Override

public void startPrefixMapping(String prefix, String uri) throws SAXException {

// ...

}

@Override

public void endPrefixMapping(String prefix) throws SAXException {

// ...

}

@Override

public void ignorableWhitespace(char[] chars, int start, int length) throws SAXException {

// ...

}

@Override

public void processingInstruction(String target, String data) throws SAXException {

// ...

}

@Override

public void skippedEntity(String name) throws SAXException {

// ...

}

private void printIndented(String what, boolean isEndOfElement, boolean isElement) {

try {

if(isEndOfElement) {

outputStreamWriter.write("\r\n");

}

if (indent > 0 && !isElement) {

outputStreamWriter.write(String.format("%1$" + (indent \* TAB\_SIZE) + "s", ""));

}

outputStreamWriter.write(what);

outputStreamWriter.flush();

} catch (IOException e) {

e.printStackTrace();

}

}

private void printAttributes(Attributes atts) {

if (atts.getLength() > 0) {

++indent;

int size=atts.getLength();

for (int i = 0; i < size; i++) {

printIndented(String.format("%s = \"%s\"", atts.getQName(i), atts.getValue(i)), false, true);

}

--indent;

}

/\* За да бъде разпечатан всеки елемент заедно с неговите атрибути на нов ред както е посочено в условието, трябва да допишем тази функция printAttributes. Като завъртаме цикъл, в който обхождаме всеки атрибут на сътветния елемент.Създала съм променлива size=atts.getLength(), която пази размера. Чрез atts.getQName(i) взимаме името на атрибута, а чрез atts.getValue(i)- стойността. Като чрез функцията printIndented отпечатваме. %s иползваме за да го интерпретираме като стринг, имаме "=", за да си се разпечата както трябва. \*/

}

}

#### **Задача 2: Като използвате SAX, имплементирайте валидация на XML документът от задача 1, която проверява дали са изпълнени следните условия:**

* 1. Всеки елемент item трябва да съдържа едно множество от под-елементите title, link и description, всеки от тях срещащ се точно един път
* 2. Стойността на атрибута version (принадлежащ на елемента rss) трябва да бъде цяло положително число
* 3. Елементът channel трябва да съдържа поне 2 и неповече от 10 под-елемента item
* 4. Разпечатайте информация за мястото (номер на ред и колона), на което грешката се среща

import org.xml.sax.InputSource;

import org.xml.sax.XMLReader;

import org.xml.sax.helpers.XMLReaderFactory;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.StringReader;

import org.xml.sax.Attributes;

import org.xml.sax.ContentHandler;

import org.xml.sax.Locator;

import org.xml.sax.SAXException;

public class Main {

public static void main(String[] args) {

OutputStreamWriter outputStreamWriter = new OutputStreamWriter(System.out);

try {

XMLReader parser = XMLReaderFactory.createXMLReader();

InputSource source = new InputSource("rss.xml");

//InputSource source = new InputSource(new StringReader(""));

parser.setContentHandler(new SAXHandler(outputStreamWriter)

);

parser.parse(source);

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

outputStreamWriter.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

class SAXHandler implements ContentHandler {

Locator locator;

Integer indent;

OutputStreamWriter outputStreamWriter;

private final Integer TAB\_SIZE = 4;

public SAXHandler(OutputStreamWriter outputStreamWriter) {

this.outputStreamWriter = outputStreamWriter;

indent = 0;

}

public class SAXValidator implements ContentHandler {

Locator locator;

OutputStreamWriter outputStreamWriter;

private final Integer TAB\_SIZE = 4;

private String currentElementName;

private boolean titleDetected = false;

private int countOfTitle = 0;

private boolean linkDetected =false;

private int countOfLink = 0;

private boolean descriptionDetected = false;

private int countOfDescription = 0;

private boolean itemDetected = false;

private int countOfItem = 0;

public SAXValidator(OutputStreamWriter outputStreamWriter) {

this.outputStreamWriter = outputStreamWriter;

}

@Override

public void setDocumentLocator(Locator locator) {

this.locator = locator;

}

@Override

public void startDocument() throws SAXException {

//..

}

@Override

public void endDocument() throws SAXException {

// ...

}

@Override

public void startElement(String uri, String localName, String qName, Attributes atts) throws SAXException {

currentElementName = qName;

validateVersion(atts);

//валидираме

if (qName.equals("item")) {

titleDetected = false;

linkDetected = false;

descriptionDetected = false;

countOfTitle = 0;

countOfDescription = 0;

countOfLink = 0;

itemDetected = true;

++countOfItem;

}

if (qName.equals("title")) {

titleDetected = true;

++countOfTitle;

}

if (qName.equals("link")) {

linkDetected = true;

++countOfLink;

}

if (qName.equals("description")) {

descriptionDetected = true;

++countOfDescription;

}

/\*Всеки елемент item трябва да съдържа едно множество от под-елементите title, link и description, всеки от тях срещащ се точно един път \*/

}

@Override

public void endElement(String uri, String localName, String qName) throws SAXException {

if (localName.equals("item")) {

if (!(titleDetected && linkDetected && descriptionDetected && countOfTitle == 1 && countOfLink == 1 && countOfDescription == 1)) {

reportError("Item must have one subset of the sequence: title, link, description.");

}

/\*Всеки елемент item трябва да съдържа едно множество от под-елементите title, link и description, всеки от тях срещащ се точно един път, ако не е така-връщаме грешка \*/

}

if (localName.equals("channel")) {

if (!(itemDetected && countOfItem >= 2 && countOfItem <= 10)) {

reportError("Number of elements item must be between 2 and 10: ");

}

}

/\*Елементът channel трябва да съдържа поне 2 и неповече от 10 под-елемента item като ако не е вярно това-връщаме грешка. \*/

}

@Override

public void characters(char[] chars, int start, int length) throws SAXException {

//..

}

@Override

public void startPrefixMapping(String prefix, String uri) throws SAXException {

// ...

}

@Override

public void endPrefixMapping(String prefix) throws SAXException {

// ...

}

@Override

public void ignorableWhitespace(char[] chars, int start, int length) throws SAXException {

// ...

}

@Override

public void processingInstruction(String target, String data) throws SAXException {

// ...

}

@Override

public void skippedEntity(String name) throws SAXException {

// ...

}

private void printIndented(String what) {

try {

outputStreamWriter.write(what);

outputStreamWriter.flush();

} catch (IOException e) {

e.printStackTrace();

}

}

private void reportError(String cause) {

printIndented(String.format("\r\nError: %s on line %d column %d.", cause, locator.getLineNumber(), locator.getColumnNumber()));

/\* Чрез locator.getLineNumber() отпечатваме реда, а чрез locator.getColumnNumber() -отпечатваме колона, където се среща грешката \*/

}

private void validateVersion(Attributes atts) {

if (atts.getLength() > 0) {

try {

if (currentElementName.equals("rss") && (Integer.parseInt(atts.getValue("version")) < 0)) {

reportError("Attribute version is expected to have a positive integer value: ");

}

} catch (NumberFormatException e) {

reportError(String.format("Wrong value for version: %s (Attribute version is expected to have a positive integer value):", atts.getValue("version")));

}

}

}

/\* Проверява дали стойността на атрибута version, който принадлежи на елемента rss,е цяло положително число, ако не връща грешка. \*/

}

#### **Задача 3: Като използвате SAX, трансформирайте XML документът от задача 1 в HTML документ, използвайки за изход стандартната конзола. Новият HTML документ трябва да съдържа таблица със следното съдържание:**

* 1. Три колони с имена title, link и description
* 2. По един ред за всеки елемент item със стойностите на под-елементите му title, link и description

import org.xml.sax.InputSource;

import org.xml.sax.XMLReader;

import org.xml.sax.helpers.XMLReaderFactory;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.StringReader;

import org.xml.sax.Attributes;

import org.xml.sax.ContentHandler;

import org.xml.sax.Locator;

import org.xml.sax.SAXException;

public class Main {

public static void main(String[] args) {

OutputStreamWriter outputStreamWriter = new OutputStreamWriter(System.out);

try {

XMLReader parser = XMLReaderFactory.createXMLReader();

InputSource source = new InputSource("rss.xml");

//InputSource source = new InputSource(new StringReader("")); //for using in https://ideone.com/ instead new InputSource("rss.xml")

parser.setContentHandler(new SAXHandler(outputStreamWriter)

);

parser.parse(source);

} catch (Exception e) {

e.printStackTrace();

} finally {

try {

outputStreamWriter.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

class SAXHandler implements ContentHandler {

Locator locator;

Integer indent;

OutputStreamWriter outputStreamWriter;

private final Integer TAB\_SIZE = 4;

private class Item {

String title;

String link;

String description;

}

private String currentElement;

private Item currentItem;

boolean inItem = false;

public SAXHandler(OutputStreamWriter outputStreamWriter) {

this.outputStreamWriter = outputStreamWriter;

indent = 0;

}

@Override

public void setDocumentLocator(Locator locator) {

this.locator = locator;

}

@Override

public void startDocument() throws SAXException {

printIndented("<!DOCTYPE html>");

printIndented("<html>");

++indent;

printIndented("<body>");

++indent;

printIndented("<table border='1px'>");

++indent;

printIndented("<thead><tr><th>Title</th><th>Link</th><th>Description</th></tr></thead>");

printIndented("<tbody>");

++indent;

/\* Началото на документа го правим както един html документ- пишем си "<!DOCTYPE html>","<html>","<body>".

Отваряме си и таблицата,която има три колони с имена- title, link и description

Чрез "<thead><tr><th>Title</th><th>Link</th><th>Description</th></tr></thead>" задаваме заглавния ред\*/

}

@Override

public void endDocument() throws SAXException {

--indent;

printIndented("</tbody>");

--indent;

printIndented("</table>");

--indent;

printIndented("</body>");

--indent;

printIndented("</html>");

/\* Края на документа го правим отново както един html документ

Затваряме си таблицата и завършваме документа както е редно\*/

}

@Override

public void startElement(String uri, String localName, String qName, Attributes atts) throws SAXException {

currentElement = qName;

if ("item".equals(currentElement)) {

currentItem = new Item();

inItem = true;

}

/\* За всеки елемент,където срещнем item, казваме че е в йерархията му -inItem = true \*/

}

@Override

public void endElement(String uri, String localName, String qName) throws SAXException {

if ("item".equals(localName)) {

printIndented("<tr><td>" + currentItem.title + "</td><td>" + currentItem.link + "</td><td>" + currentItem.description + "</td></tr>");

inItem = false;

}

/\*За всеки елемент принтираме ред със съответните стойности на title, link, description.

Чрез inItem = false казваме, че сме излезли от йерархията на item \*/

}

@Override

public void characters(char[] chars, int start, int length) throws SAXException {

String s = new String(chars, start, length).trim();

if (inItem && s.length() > 0) {

if ("title".equals(currentElement)) {

currentItem.title = s;

}

if ("link".equals(currentElement)) {

currentItem.link = s;

}

if ("description".equals(currentElement)) {

if(currentItem.description == null) {

currentItem.description = s;

} else {

currentItem.description += s;

}

}

}

/\* Като сме в йерархията и s.length() > 0, тогава присвояваме стойността за съответно title, link, description. \*/

}

@Override

public void startPrefixMapping(String prefix, String uri) throws SAXException {

// ...

}

@Override

public void endPrefixMapping(String prefix) throws SAXException {

// ...

}

@Override

public void ignorableWhitespace(char[] chars, int start, int length) throws SAXException {

// ...

}

@Override

public void processingInstruction(String target, String data) throws SAXException {

// ...

}

@Override

public void skippedEntity(String name) throws SAXException {

// ...

}

private void printIndented(String what) {

try {

if (indent > 0) {

outputStreamWriter.write(String.format("%1$" + (indent \* TAB\_SIZE) + "s", ""));

}

outputStreamWriter.write(what + "\r\n");

outputStreamWriter.flush();

} catch (IOException e) {

e.printStackTrace();

}

}

/\* Принтиращата функция, чрез която си принтираме данните сътоветно където сме я посочили.\*/

}

#### **Задача 4: Решете задача 1 като вместо SAX използвате DOM**

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Attr;

import org.w3c.dom.Document;

import org.w3c.dom.NamedNodeMap;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.InputSource;

public class Main {

private static boolean skipNL;

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

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

dbf.setValidating(false);

DocumentBuilder builder = dbf.newDocumentBuilder();

InputSource source = new InputSource("rss.xml");

//InputSource source = new InputSource(new StringReader(""));

Document document = builder.parse(source);

System.out.println(printXML(document.getDocumentElement()));

}

private static String printXML(Node rootNode) {

String tab = "";

skipNL = false;

return(printXML(rootNode, tab));

}

private static String printXML(Node rootNode, String tab) {

String print = "";

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

print += "\n"+tab+"<"+rootNode.getNodeName();

NamedNodeMap attributes = rootNode.getAttributes();

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

Attr attr = (Attr) attributes.item(j);

if(attr != null) {

print += " " + attr.getNodeName() + "=\"" + attr.getNodeValue() + "\"";

//append to print attr.getNodeName() and attr.getNodeValue()

}

}

print += ">";

}

NodeList nl = rootNode.getChildNodes();

if(nl.getLength()>0) {

//append to print all nl.item(i) in a cycle until nl.getLength()

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

print += printXML(nl.item(i), tab );

}

/\*Ако има наследници-nl.getLength()>0, прибавяме рекурсивно данните(съдържанието) на всеки едино от поделементите на дадения ни елемемент(текущия)\*/

} else {

if(rootNode.getNodeValue()!=null) {

print = rootNode.getNodeValue().toUpperCase();

/\* Ако няма наследници,но ако има съдържание,то трябва да се направи да е с главни букви и затова прибавяме и .toUpperCase() \*/

}

skipNL = true;

}

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

if(!skipNL) {

print += "\n"+tab;

}

skipNL = false;

print += "</"+rootNode.getNodeName()+">";

}

return(print);

}

/\*Ако е края на елемента добавяме затварящия таг\*/

}

#### **Задача 5: Използвайки DOM, променете XML документът от задача 1, по следния начин:**

* 1. Превърнете под-елемента link на елемента item в негов атрибут
* 2. Запазете първите 10 item елементи, а всички останали ги изтрийте
* 3. Добавете нов под-елемент sponsor на елемента channel

import java.io.File;

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.NodeList;

import org.xml.sax.InputSource;

public class Main {

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

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

dbf.setValidating(false);

DocumentBuilder builder = dbf.newDocumentBuilder();

InputSource source = new InputSource("rss.xml");

//InputSource source = new InputSource(new StringReader(""));

Document document = builder.parse(source);

processTree(document);

TransformerFactory tf = TransformerFactory.newInstance();

Transformer writer = tf.newTransformer();

writer.setOutputProperty(OutputKeys.ENCODING, "utf-8");

writer.transform(new DOMSource(document), new StreamResult(new File("rss\_new.xml")));

}

private static void processTree(Document doc) {

NodeList linkList = doc.getElementsByTagName("link");

NodeList itemList = doc.getElementsByTagName("item");

for (int i = itemList.getLength() - 1; i >= 10; --i) {

Element item = (Element)itemList.item(i);

item.getParentNode().removeChild(item);

}

/\*Трябва да запазим първите 10 item елемента и всички останали да бъдат изтрити. Затова като се върти цикъла се започна от размера-1 и се върти до i>=10., и се трият чрез item.getParentNode().removeChild(item);\*/

for (int i = linkList.getLength() - 1; i >= 0; --i) {

Element link = (Element)linkList.item(i);

Element item = (Element)link.getParentNode();

if("item".equals(item.getNodeName())) {

item.setAttribute("link", link.getTextContent().trim());

item.removeChild(link);

}

/\*Превръщаме под-елемента link на елемента item в негов атрибут item.setAttribute("link", link.getTextContent().trim()) като чрез item.removeChild(link) изтриваме нали под-елемента(защото вече е атрибут) \*/

}

Element sponsor = doc.createElement("sponsor");

sponsor.setTextContent("Marvel");

doc.getElementsByTagName("channel").item(0).appendChild(sponsor);

/\*Добавяме нов под-елемент sponsor на елемента channel чрез doc.createElement("sponsor") \*/

}

}