Nom : MATHEY-APOSSAN
Prénom : Maté Ulrich Graciano
Option : GL
N° carte : 551078

Rapport TP 7 XML

Classe FactbookContentHandler1

```
import java.util.ArrayList;
import java.util.List;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.helpers.DefaultHandler;

public class FactbookContentHandler1 extends DefaultHandler {

	/**
	* Actions à réaliser lors de la détection d'une balise ouvrante
	*/
	private String bretagnePopulation;
	private List<String> fleuvesLongs = new ArrayList<>();
	private int maxSize = 0;
	private String largestIslandName = "";
	private int oldestDate = Integer.MAX_VALUE;
	private String oldestOrganizationName = "";

	@Override
	public void startElement(String nameSpace, String localName, String qName,
	Attributes attrs) throws SAXException {
	if (qName.equals("province") && attrs.getValue("name").equals("Bretagne")) {
	bretagnePopulation = attrs.getValue("population");
	}

	// Rivières
	if (qName.equals("river")) {
	String lengthStr = attrs.getValue("length");
	if (lengthStr != null) {
		int length = Integer.parseInt(lengthStr);
		if (length >= 6000) {
```

```
        String riverName = attrs.getValue("name");
        fleuvesLongs.add(riverName);
        }
}
}

// Island name
if (qName.equalsIgnoreCase("island")) {
String area = attrs.getValue("area");
if (area != null) {
        int islandSize = Integer.parseInt(area);
        if (islandSize > maxSize) {
        maxSize = islandSize;
        largestIslandName = attrs.getValue("name");
        }
}
}

// Organisation
if (qName.equalsIgnoreCase("organization")) {
String established = attrs.getValue("established");
if (established != null) {
        String[] mots = established.split(" ");
        if (mots.length == 3) {
        try {
                int jj = Integer.parseInt(mots[0]);
                int mm = Integer.parseInt(mots[1]);
                int aaaa = Integer.parseInt(mots[2]);
                int date = (aaaa * 12 + mm) * 31 + jj;
                if (date < oldestDate) {
                oldestDate = date;
                oldestOrganizationName = attrs.getValue("name");
                }
        } catch (NumberFormatException ignored) {
        }
        }
}
}

}

/**
* Actions à réaliser lors de la détection de la fin d'un élément
*/
```

```
	@Override
	public void endElement(String nameSpace, String localName, String qName)
	throws SAXException {
	}

	@Override
	public void endDocument() throws SAXException {
	System.out.println("La plus grande île est : " + largestIslandName);
	System.out.println("La plus ancienne organisation est : " + oldestOrganizationName);
	}

	/**
	* gestionnaire d'erreur *
	*/
	@Override
	public void error(SAXParseException e) {
	System.err.println("Erreur ligne " + e.getLineNumber()
		+ " colonne " + e.getColumnNumber());
	System.err.println(e.getMessage());
	}

	/**
	* gestionnaire d'erreur fatale *
	*/
	@Override
	public void fatalError(SAXParseException e) {
	System.err.println("Erreur fatale ligne " + e.getLineNumber()
		+ " colonne " + e.getColumnNumber());
	System.err.println(e.getMessage());
	}

	public String getBretagnePopulation() {
	return bretagnePopulation;
	}

	public List<String> getFleuvesLongs() {
	return fleuvesLongs;
	}

}
```

Classe FactbookContentHandler2

```
import org.xml.sax.Attributes;
```

```
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;


public class FactbookContentHandler2 extends DefaultHandler {

	// gestion des états de l'automate
	private enum Etat {
	INIT, PAYS, LANG
	};
	private Etat EtatCourant;

	// texte en cours de lecture, utiliser texte.toString() pour sa valeur
	private StringBuilder texte = new StringBuilder();

	// nom du pays courant, null s'il a déjà été affiché
	private String pays = null;

	/**
	* Actions à réaliser lors de la détection d'une balise ouvrante
	*/
	@Override
	public void startElement(String nameSpace, String localName, String qName,
	Attributes attrs) throws SAXException {
	// selon l'état courant
	switch (EtatCourant) {

	case INIT:
		if ("country".equals(qName)) {
		// mémoriser le nom du pays
		pays = attrs.getValue("name");
		// état suivant
		EtatCourant = Etat.PAYS;
		}
		break;

	case PAYS:
		if ("languages".equals(qName)) {
		// état suivant
		EtatCourant = Etat.LANG;
		}
		break;

	default:
```

```
// on ne fait rien, on ignore cette balise
}

// vider le StringBuilder à chaque balise
texte.setLength(0);
}

/**
* Actions à réaliser lors de la détection de la fin d'un élément
*/
@Override
public void endElement(String nameSpace, String localName, String qName)
throws SAXException {
switch (EtatCourant) {

case LANG:
	if ("languages".equals(qName)) {
	// si le pays n'a pas déjà été affiché, le faire maintenant
	if (pays != null) {
		System.out.print(pays + " : ");
		// noter qu'il a été affiché, voir le else
		pays = null;
	} else {
		// le pays a déjà été affiché, donc n'afficher qu'une virgule
		System.out.print(", ");
	}
	// afficher la langue
	System.out.print(texte.toString());
	// remonter
	EtatCourant = Etat.PAYS;
	}
	break;

case PAYS:
	if ("country".equals(qName)) {
	// sauter à la ligne si le pays a été affiché
	if (pays == null) {
		System.out.println();
	}
	EtatCourant = Etat.INIT;
	}
	break;

default:
```

```
        // on ne fait rien, on ignore cette balise
        }

        // effacer le texte à chaque balise
        texte.setLength(0);
        }

        /**
        * Actions à réaliser sur les textes
        */
        @Override
        public void characters(char[] text, int debut, int lng) {
        // si on est entre <languages> et </languages>, on stocke le texte
        if (EtatCourant == Etat.LANG) {
        texte.append(new String(text, debut, lng));
        }
        }

}
```

Classe FactbookContentHandler3

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;


public class FactbookContentHandler3 extends DefaultHandler {

        // gestion des états de l'automate
        private enum Etat {
        INIT, PAYS, LANG
        };
        private Etat EtatCourant;

        // texte en cours de lecture, utiliser texte.toString() pour sa valeur
        private StringBuilder texte = new StringBuilder();

        // nom du pays courant, null s'il a déjà été affiché
        private String pays = null;

        /**
        * Actions à réaliser lors de la détection d'une balise ouvrante
        */
```

```
@Override
public void startElement(String nameSpace, String localName, String qName,
Attributes attrs) throws SAXException {
// selon l'état courant
switch (EtatCourant) {

case INIT:
	if ("country".equals(qName)) {
	// mémoriser le nom du pays
	pays = attrs.getValue("name");
	// état suivant
	EtatCourant = Etat.PAYS;
	}
	break;

case PAYS:
	if ("languages".equals(qName)) {
	// état suivant
	EtatCourant = Etat.LANG;
	}
	break;

default:
// on ne fait rien, on ignore cette balise
}

// vider le StringBuilder à chaque balise
texte.setLength(0);
}

/**
* Actions à réaliser lors de la détection de la fin d'un élément
*/
@Override
public void endElement(String nameSpace, String localName, String qName)
throws SAXException {
switch (EtatCourant) {

case LANG:
	if ("languages".equals(qName)) {
	// si le pays n'a pas déjà été affiché, le faire maintenant
	if (pays != null) {
		System.out.print(pays + " : ");
		// noter qu'il a été affiché, voir le else
```

```
                pays = null;
            } else {
                // le pays a déjà été affiché, donc n'afficher qu'une virgule
                System.out.print(", ");
            }
            // afficher la langue
            System.out.print(texte.toString());
            // remonter
            EtatCourant = Etat.PAYS;
            }
            break;

        case PAYS:
            if ("country".equals(qName)) {
            // sauter à la ligne si le pays a été affiché
            if (pays == null) {
                System.out.println();
            }
            EtatCourant = Etat.INIT;
            }
            break;

        default:
        // on ne fait rien, on ignore cette balise
        }

        // effacer le texte à chaque balise
        texte.setLength(0);
        }

        /**
        * Actions à réaliser sur les textes
        */
        @Override
        public void characters(char[] text, int debut, int lng) {
        // si on est entre <languages> et </languages>, on stocke le texte
        if (EtatCourant == Etat.LANG) {
        texte.append(new String(text, debut, lng));
        }
        }
        }
}
```

Classe Main1

```
/**
 *
 * @author time
 */
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class Main1 {

        public static void main(final String[] args) {
        try {

        // ouvrir et traiter le fichier
        UtilXML.lireDocumentSAX("factbook.xml", new FactbookContentHandler1());

        } catch (Exception e) {
        // afficher où ça s'est planté
        e.printStackTrace();
        }
        }
}
```

Classe Main2

```
/**
 *
 * @author time
 */
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class Main2 {

        public static void main(final String[] args) {
        try {
```

```
        // ouvrir et traiter le fichier
        UtilXML.lireDocumentSAX("factbook.xml", new FactbookContentHandler2());

        } catch (Exception e) {
        // afficher où ça s'est planté
        e.printStackTrace();
        }
        }
}
```

Classe 3

```
/**
 *
 * @author time
 */
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class Main3 {

        public static void main(final String[] args) {
        try {

        // ouvrir et traiter le fichier
        UtilXML.lireDocumentSAX("factbook.xml", new FactbookContentHandler3());

        } catch (Exception e) {
        // afficher où ça s'est planté
        e.printStackTrace();
        }
        }
}
```