

Parcours de fichiers XML avec XPath

par Joseph Pachod (ZedroS)

Date de publication : 31/03/2007

Dernière mise à jour : 29/06/2007

L'article qui suit est une introduction à l'utilisation de la technologie **XPath** depuis **Java**, grâce aux bibliothèques **JDom** et **Jaxen**.

- I - Introduction
- II - Présentation
 - II-A - Qu'est ce que XPath ?
 - II-B - Solutions Java pour XPath
 - II-C - Environnement requis
- III - Principe de fonctionnement
 - III-A - Vue d'ensemble
 - III-B - La syntaxe XPath
 - III-C - XPath appliqué au Java
- IV - Exemple de mise en oeuvre
 - IV-A - Le fichier XML considéré
 - IV-B - La classe principale
- V - Conclusion
- VI - Remerciements

I - Introduction

XPath permet de parcourir un fichier XML d'une façon à la fois simple et puissante. De la sorte, en peu de temps, un développeur peut rapidement et aisément extraire les informations qui l'intéressent, même dans un document qui en comporte bien plus. On peut par exemple :

- récupérer le contenu d'une balise précise
- récupérer du contenu en fonction de la valeur d'un attribut d'une balise
- récupérer un ensemble de balises avec leur contenu et les parcourir

C'est ce que nous allons détailler dans ce tutoriel. Après une courte présentation d'XPath, de quelques solutions Java existantes, nous passerons rapidement à la pratique avec des explications et exemples utilisant JDom et Jaxen. Pour finir, vous trouverez des codes sources et d'autres liens intéressants en fin d'article.

II - Présentation

II-A - Qu'est ce que XPath ?

Pour cela, rien de tel que de parcourir la [FAQ XML \(XPath\)](#), qui nous dit :

W3C XPath est un langage avec une syntaxe non XML, permettant d'adresser les différents noeuds ou groupes de noeuds particuliers d'un document XML. XPath voit le document XML comme un arbre de noeuds, qu'il permet de parcourir selon des axes (fils, parent, ancêtre, descendant, ...) et en sélectionnant les noeuds par leur nom. Les expressions XPath permettent également de représenter des chaînes de caractères, des nombres et des booléens.

XPath est intensément utilisé par d'autres langages et technologies XML, tels que XSLT, les W3C XML Schema, XLink, XForms...

D'autres définitions sont également disponibles dans la [FAQ Java/XML \(XPath\)](#) ou directement sur le [site du W3C](#).

II-B - Solutions Java pour XPath



Comme souvent en Java, le choix est vaste. Voici quelques unes des APIs XPath disponibles pour la plateforme Java :

- **APIs standard** : Depuis J2SE 5.0 (Tiger), l'API standard contient le package `javax.xml.xpath`, qui permet de travailler avec XPath.
- **Jaxen** : Cette API est capable d'évaluer des expressions XPath sur de nombreux modèles (JDOM, DOM4J ou encore DOM).
- **Saxon** : Pour finir, on peut citer Saxon, qui a la particularité de permettre de travailler avec la spécification XPath 2.0.

Pour cet article, le choix s'est porté sur **Jaxen**, et plus précisément l'implémentation que l'on trouve dans **JDom**.

II-C - Environnement requis

jaxen Comme précisé plus haut, les librairies utilisées sont JDom et Jaxen. Voici les jars dont on aura besoin :

- **jdom.jar** : définition des interfaces
- **jaxen-core.jar** : pour le parsing
- **jaxen-jdom.jar** : implémentation d'XPath
- **saxpath.jar** : pour construire le fichier Dom

Pour obtenir ces jars, rien de plus simple : il suffit de se rendre sur le site de **Jdom**. Dans **Downloads/binaries** vous avez le lien pour Jdom 1.0. Téléchargez alors l'ensemble Jdom au format de votre choix puis décompressez le. On y trouve :

- sous \build : jdom.jar

- sous \lib : les autres jars qui nous intéressent, à savoir jaxen-core.jar, jaxen-dom.jar et saxpath.jar

III - Principe de fonctionnement

III-A - Vue d'ensemble

La bibliothèque JDom propose une interface qui définit une façon d'utiliser XPath en Java pour parcourir des fichiers XML. Le jar jaxen-jdom implémente cette interface.

Le fonctionnement est le suivant : le document est entièrement parsé et mis en mémoire, via un Document JDom. A partir de ce document, on va appliquer des chemins XPath pour trouver les éléments qui nous concernent. Ces chemins XPath permettent d'aller à l'essentiel, en évitant de décrire le chemin exact suivi. On va pouvoir demander de chercher tel attribut ou telle balise à partir de l'endroit courant ou sur l'ensemble du Document.

Bien sûr, ce parcours initial, et surtout la constitution du Document, n'est pas l'approche la plus rapide (voir SAX pour cela).

Par contre, côté développeur, le gain de temps dans le parcours du fichier XML est considérable. Plus besoin de chercher le détail de l'enchaînement des balises, on va directement à l'essentiel !

III-B - La syntaxe XPath

XPath propose en fait un langage de localisation, dont les éléments de base sont les suivants :

Élément syntaxique	Type de chemin	Exemples de chemin	Exemples de chemin répondant au chemin indiqué à gauche
balise	Nom d'élément	hopital	<hopital>...</hopital>
/	Sépare enfants directs	hopital/personne	<hopital> <personne>...</personne> </hopital>
//	Descendant	hopital//relations ---- //nom	<hopital> <pensionnaires> <patient> <relations>...</relations> </patient> </hopital> ---- <hopital> <personne>

			<code><nom></nom></code> <code></personne></code> <code><hopital></code> => recherche n'importe où dans le schéma
*	"wildcard"	<code>*/title</code>	<code><bla><title>..</title></code> ou <code><bli><title>...</title></code>
	opérateur "ou"	<code>title head</code> ---- <code>* / @*</code>	<code><title>...</title></code> ou <code><head></code> <code>...</head></code> ---- (tous les éléments: les enfants, la racine et les attributs de la racine)
.	élément courant	.	
../	élément supérieur	<code>../problem</code>	<code><project></code>
@	nom d'attribut	<code>@id</code> ---- <code>project/@id</code>	<code><xyz</code> <code>id="test">...</</code> <code>xyz></code> ---- <code><project</code> <code>id="test" ...> ...</code> <code></project></code>
<code>@attr='type'</code>	Valeur d'attribut	<code>list[@type='ol']</code>	<code><list type="ol"></code> <code></list></code>

Pour plus de renseignements, je vous enjoint à regarder cet excellent pdf : [Introduction technique à XSLT](#), dont le tableau ci-dessus est fortement inspiré (avec l'accord des auteurs de ce document).

Le résultat d'une recherche peut prendre plusieurs formes :

- la valeur du premier nœud répondant à l'expression,
- le premier nœud répondant à l'expression,
- l'ensemble des nœuds répondant à la requête.

Par nœud, il faut entendre l'emplacement d'une balise précise dans le fichier XML. Ainsi, à l'aide du « . » ou du « ../ », il est également possible de faire une recherche à partir du ou des nœuds trouvés dans une recherche précédente.

Comme vous pouvez le constater, la puissance d'expression de ce langage est conséquente.

III-C - XPath appliqué au Java

Il faut dans un premier temps construire le document DOM :

```
//On crée une instance de SAXBuilder
SAXBuilder sxb = new SAXBuilder();
org.jdom.Document document = sxb.build(new File("exempleXml.xml"));
//On initialise un nouvel élément racine avec l'élément racine du
// document.
racine = document.getRootElement();
```

Il faut ensuite déclarer le chemin à chercher, on procède ainsi :

```
XPath xpa = XPath.newInstance("chemin");
```

On applique alors ce chemin à un nœud de notre choix, par exemple la racine. Nous pouvons choisir soit :

- d'obtenir la valeur retournée :

```
String retour = xpa.valueOf(currentNode);
```

- d'obtenir le premier nœud répondant à notre requête :

```
Element monNoeud = (Element) xpa.selectSingleNode(racine);
```

- d'obtenir tous les nœuds répondant à notre requête :

```
List listeNoeuds = xpa.selectNodes(racine);
```

IV - Exemple de mise en oeuvre

Un exemple vaut mieux que de nombreux discours, voyons donc ce que cela donne.

IV-A - Le fichier XML considéré

Pour cela, nous allons nous baser sur un fichier XML un peu complexe, construit ainsi :



hopital.xml

```
<?xml version= "1.0" encoding= "ISO-8859-1"?>
<hopital>
  <personne id="idPersonnel">
    <nom>Hugo</nom>
    <prenom>Victor</prenom>
  </personne>
  <personne id="idPersonne2">
    <nom>Victor</nom>
    <prenom>Paul Emile</prenom>
  </personne>
  <personne id="idPersonne3">
    <nom>Kontz</nom>
    <prenom>Dean R.</prenom>
  </personne>
  <personne id="idPersonne4">
    <nom>Sand</nom>
    <prenom>Georges</prenom>
  </personne>
  <personne id="idPersonne5">
    <nom>Clark</nom>
    <prenom>Marie Higgins</prenom>
  </personne>
  <personne id="idPersonne6">
    <nom>Weis</nom>
    <prenom>Margaret</prenom>
  </personne>
  <personne id="idPersonne7">
    <nom>Fielding</nom>
    <prenom>Helen</prenom>
  </personne>
  <personne id="idPersonne8">
    <nom>de Troyes</nom>
    <prenom>Chrétien</prenom>
  </personne>
  <pensionnaires>
    <patient personneId="idPersonne3">
      <filiation>
        <parent personneId="idPersonnel" type="naturel">
          <contact type="journalier" ddebut="1989/02/16"
            dfin = "2001/12/21">Enfance</contact>
          <contact type="épisodique" ddebut="2001/12/22"
            dfin = "..">Divorce des parents, baisse des contact
          </contact>
        </parent>
        <parent personneId="idPersonne4" type="naturel" >
          <contact type="régulier" ddebut="1989/02/16"
            dfin = "2001/12/21">Enfance de Dean R.</contact>
          <contact type="aucun" ddebut="2001/12/22"
            dfin = "..">
            Rupture entre les parents, Dean R. ne parle plus à sa mère.
          </contact>
        </parent>
      </filiation>
    </patient>
  </pensionnaires>
</hopital>
```

hopital.xml

```

<parent personneId="idPersonne7" type="tuteur" >
  <contact type="fréquent" ddebut="2002/02/12"
    dfin = "..">Nouveau tuteur légal</contact>
</parent>
</filiation>
<internements>
  <sejour ddebut="2002/01/11" dfin="2002/01/14">
    Crise d'appendicite</sejour>
</internements>
</patient>
<patient personneId="idPersonne2">
  <filiation>
    <parent personneId="idPersonne1" type="naturel">
      <contact type="épisodique" ddebut="1985/03/21"
        dfin = "..">Père souvent en déplacement</contact>
    </parent>
    <parent personneId="idPersonne5" type="naturel" >
      <contact type="journalier" ddebut="1985/03/21"
        dfin = "1999/08/11">Paul Emile vit chez sa mère</contact>
      <contact type="aucun" ddebut="1999/08/12"
        dfin = "..">Décès</contact>
    </parent>
  </filiation>
  <internements>
    <sejour ddebut="1998/03/22" dfin="1998/04/12">
      Crise de rire à répétition</sejour>
  </internements>
</patient>
<patient personneId="idPersonne6">
  <filiation>
    <parent personneId="idPersonne8" type="naturel">
      <contact type="régulier" ddebut="1982/12/30"
        dfin = "..">RAS</contact>
    </parent>
    <parent personneId="idPersonne4" type="naturel">
      <contact type="régulier" ddebut="1982/12/30"
        dfin = "..">RAS</contact>
    </parent>
  </filiation>
  <internements>
    <sejour ddebut="1998/03/22" dfin="1998/03/25">Bras cassé
  </sejour>
  </internements>
</patient>
</pensionnaires>
</hopital>

```

Je précise que ce fichier n'est qu'à but purement éducatif. Il ne représente pas un exemple de bonne construction XML, notamment au niveau des balises <personne>. Ce fichier est disponible [là](#).

IV-B - La classe principale


ExempleXPath.java

```

package org.zedros.xpathxmlreader;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Iterator;
import java.util.List;
import org.jdom.Element;

```

ExempleXPath.java

```
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.xpath.XPath;

/**
 *
 * @author ZedroS
 */
public class ExempleXPath {

    /** Creates a new instance of ExempleXPath */
    public ExempleXPath() {

    }

    /** Parse le fichier passé en entrée pour lire le nom de chaque patient.
     * Pour cela, on cherche les patients puis, à partir de leur identifiant, on cherche leur nom.
     *
     * @ params Nom du fichier à lire.
     */
    void parse(File _FilePath) {
        org.jdom.Document document = null ;
        try {
            /* On crée une instance de SAXBuilder */
            SAXBuilder sxb = new SAXBuilder();
            document = sxb.build(_FilePath);
        } catch (IOException e) {
            System.out.println("Erreur lors de la lecture du fichier "
+ e.getMessage() );
            e.printStackTrace();
        } catch (JDOMException e){
            System.out.println("Erreur lors de la construction du fichier JDOM "
+ e.getMessage() );
            e.printStackTrace();
        }

        try {
            /* On initialise un nouvel élément avec l'élément racine du
            document. */
            Element racine = document.getRootElement();

            /* On va dans un premier temps rechercher l'ensemble des noms
            des patients de notre hôpital. */

            /* Recherche de la liste des patients */
            XPath xpa = XPath.newInstance("//patient");

            /* On récupère tous les noeuds répondant au chemin //patient */
            List results = xpa.selectNodes(racine) ;

            Iterator iter = results.iterator() ;

            Element noeudCourant = null;
            String personneId = null ;
            while (iter.hasNext()){
                /* Pour chaque patient nous allons chercher son nom puis l'afficher */
                noeudCourant = (Element) iter.next();

                /* On récupère l'identifiant de la personne
                Noter le . en début du chemin : on part de la position courante
                le @ indique que l'on cherche un attribut */
                xpa = XPath.newInstance("./@personneId");
                personneId = xpa.valueOf(noeudCourant);

                /* A partir de là on récupère les infos dans la balise personne correspondante
                On spécifie que l'on recherche une balise en fonction de la valeur
                d'un de ses attributs :*/
                xpa = XPath.newInstance("//personne[@id='" + personneId + '"]");
            }
        }
    }
}
```

ExempleXPath.java

```
        noeudCourant = (Element) xpa.selectSingleNode(noeudCourant);

        /* Nous cherchons à présent la valeur de la balise nom : */
        xpa = XPath.newInstance("./nom");
        System.out.println("Valeur : " + xpa.valueOf(noeudCourant));
    }
} catch (JDOMException e) {
    System.out.println("Erreur JDOM " + e.getMessage());
    e.printStackTrace();
}
}
```

Ce fichier est téléchargeable [là](#).

Et enfin, la classe Main pour lancer tout ça :

Main.java



```
package org.zedros.xpathxmlreader;

import java.io.File;
import org.jdom.JDOMException;

/**
 *
 * @author ZedroS
 */
public class Main {

    /** Creates a new instance of Main */
    public Main() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws JDOMException {
        File entryFile = new File("d:/Java/xml/hopital01.xml");
        ExempleXPath exemple = new ExempleXPath();
        exemple.parse(entryFile);
    }
}
```

Ce fichier est téléchargeable [là](#).

V - Conclusion




J'espère que ce petit exemple vous a convaincu de l'intérêt et de la faisabilité de la mise en oeuvre d'XPath via du code Java. Pour ma part, j'utilise désormais cette approche dès que je dois parcourir du XML sans avoir de contraintes de performances.

- Sources et FAQs :**
- [Sources](#) **Fichiers de cet article**
 - [FAQ](#) **FAQ XML**

Vous trouverez ci-contre les sources de cet article ainsi que d'autres ressources intéressantes sur les technologies Java, XML et XPath.

- [FAQ](#) **La FAQ Java/XML**
- Cours et tutoriels :**

Pour finir, vous pouvez toujours vous appuyer sur les forums de developpez.com si vous avez d'autres questions : [forums XML](#) et [forums Java](#).

-  [Manipuler des données XML avec Java et JDOM](#)
-  [Introduction technique à XSLT](#)
-  [Rubrique XML de developpez.com](#)

Spécifications et API :

-  [XPath 1.0 et XPath 2.0](#)
-  [API JDom](#)
-  [API Jaxen](#)

VI - Remerciements

Un grand merci à toute l'équipe Java de développez.com et plus particulièrement à Ricky81 et Ioan pour leurs commentaires et suggestions.

Merci également à TheGzD, Abdelmonam KOUKA et gibson700 pour leurs retours suite à utilisation !

