

XML

Introduction

XML signifie eXtensible Markup Language (Langage de balisage extensible).

Le langage est standardisé par la spécification W3C XML 1.0 du 10/02/98.

Lien : [cours XML](#)

Un document XML **bien formé** signifie que le texte XML obéit aux règles syntaxiques de XML. Le document considéré comme **valide** (facultatif) signifie que le texte XML est bien formé et répond à une structure définie par une DTD (*Definition Type Document*).

Remarque : XML n'est pas un langage de programmation mais de description de document.

Les avantages d'XML sont nombreux :

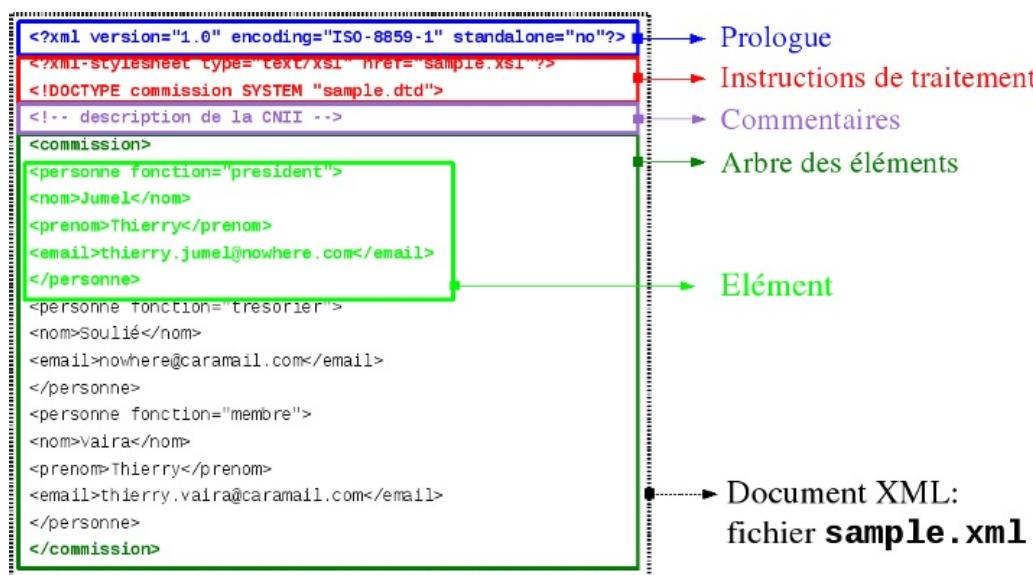
- Lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML (il est autodescriptif et extensible)
- Adaptabilité : sa structure arborescente permet de modéliser la majorité des problèmes informatiques
- Universalité et portabilité : fichier texte ASCII avec le support de l'encodage UNICODE
- Déployable : facilement distribué par n'importe quels protocoles capables de transporter du texte (comme HTTP)
- Exensibilité : ce métalangage le rend utilisable dans tous les domaines d'applications

Un document XML est structuré en trois parties :

- un **prologue** : qui indique la version XML utilisée, le jeu de caractères (*encoding*) et la présence d'une DTD (*standalone*).
- des **instructions** de traitement (IT) ou "*processing instruction*" (PI) : instructions relatives à des applications qui traiteront le document (feuille de style, transformation, ...)
- l'**arbre des éléments** : le contenu du document

Remarque : par ailleurs, une feuille XML peut contenir des commentaires de la même manière qu'un document HTML.

Un document XML est composé d'**éléments** désignés par des **balises** et structuré sous la forme d'un **arbre** avec un et un seul élément **racine** (*root*).



Remarque : Les éléments sont aussi appelés **noeuds** ou **nodes** (par référence à la théorie des graphes).

Un prologue contient systématiquement une déclaration qui spécifie la version de XML utilisée : `<?xml version="1.0" ?>`. Il existe également deux autres attributs : *encoding* (pour définir le type de codage du jeu de caractères) et *standalone* (pour préciser si une DTD est utilisée avec *no*).

De plus en plus de logiciels utilisent le format **XML** (*Extensible Markup Language*) pour leurs fichiers de données (Microsoft Office, LibreOffice/OpenOffice, ...).

Android

Comme dans d'autres environnements, il existe plusieurs manières de manipuler des données XML. On retrouve deux API souvent utilisés :

- DOM (*Document Object Model*) : charge l'ensemble du document XML et permet de manipuler l'arbre des éléments
- SAX (*Simple API for XML*) : le document est interprété ligne par ligne

SAX et DOM adoptent des stratégies différentes pour analyser (*parser*) un document XML et s'utilisent donc dans des contextes différents.

DOM

Un document est géré par une instance de [Document](#). Une instance s'obtient en appelant la méthode `parse()` (avec les données XML à analyser) de la classe [DocumentBuilder](#). On récupère une instance de [DocumentBuilder](#) en utilisant la classe [DocumentBuilderFactory](#).

On peut récupérer les données XML sous forme de chaîne de caractères (`String`) en utilisant un objet [Transformer](#).

Lire des données XML

On va commencer avec les données XML suivantes :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">MyApplicationXML</string>
    <string name="auteur">Thierry Vaira</string>
</resources>
```

Ce premier exemple permet de créer un [Document](#) et d'afficher les données XML qu'il contient :

```
public class MainActivity extends AppCompatActivity
{
    private final String TAG = "XML";
    private TextView texteView;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        texteView = findViewById(R.id.texteView);
        texteView.setText("");

        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        try
        {
            DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
            Document document = documentBuilder.parse(getInputStreamXML());

            afficher(document);
        }
        catch (Exception e)
        {
```

```

        texteView.setText("Document XML invalide !");
        e.printStackTrace();
    }

    private InputStream getInputStreamXML()
    {
        String s = null;
        s = new String("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
                      "<resources>\n" +
                      "  <string name=\"app_name\">MyApplicationXML</string>\n" +
                      "  <string name=\"auteur\">Thierry Vaira</string>\n" +
                      "</resources>");
    };

    InputStream inputStream = new ByteArrayInputStream(s.getBytes(Charset.forName("UTF-8")));

    return inputStream;
}

public void afficher(Document xml) throws Exception
{
    Transformer transformer = TransformerFactory.newInstance().newTransformer();
    transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    Writer writer = new StringWriter();
    transformer.transform(new DOMSource(xml), new StreamResult(writer));
    Log.v(TAG, writer.toString());
    texteView.setText(writer.toString());
}
}

```

Traiter des données XML

Chaque élément de l'arbre est un noeud encapsulé dans un `Node`. La méthode `getNodeType()` permettra de connaître le type du noeud. Le type de noeud peut être par exemple :

- `ELEMENT_NODE` : un élément
- `ATTRIBUTE_NODE` : un attribut
- `TEXT_NODE` : du texte

On peut récupérer un `NodeList` qui une liste ordonnée de noeuds `Node` (suivant l'ordre du document XML). On utilisera ces deux méthodes :

- `getLength()` qui retourne le nombre de noeuds
- `item()` qui retourne le noeud dont l'index est fourni en paramètre

À partir d'un `Document`, il est possible de récupérer l'élément **racine** `Element` avec `getDocumentElement()` ou la liste des noeuds (dont le nom est fourni en paramètre) avec `getElementsByTagName()`.

Un `Element` définit des méthodes pour manipuler un élément (dont ses attributs). Pour rappel, un élément dans un document XML est définie par une balise (un *tag*). On utilisera les méthodes :

- `getAttribute()` qui retourne la valeur de l'attribut dont le nom est fourni en paramètre
- `getTagName()` qui retourne le nom de la balise

On reprend les mêmes données XML :

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">MyApplicationXML</string>
  <string name="auteur">Thierry Vaira</string>
</resources>

```

On traite les données XML de la manière suivante :

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
try
{
    DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
    Document document = documentBuilder.parse(getInputStreamXML());

    afficher(document);

    Element root = document.getDocumentElement();
    Log.v(TAG, "racine = " + root.getTagName());
    if(root.hasChildNodes())
    {
        NodeList childNodes = root.getChildNodes();
        for (int i = 0; i < childNodes.getLength(); i++)
        {
            // Un noeud
            Node node = childNodes.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE)
            {
                Log.v(TAG, "noeud " + " -> type : " + getNomType(node.getNodeType()));
                Log.v(TAG, "noeud " + " -> nom : " + node.getNodeName());

                // Les attributs du noeud
                for(int j = 0; j < node.getAttributes().getLength(); j++)
                {
                    Node attribut = node.getAttributes().item(j);
                    Log.v(TAG, "    attribut " + (j+1) + " -> type : " + getNomType(attribut.getNodeType()));
                    Log.v(TAG, "    attribut " + (j+1) + " -> nom : " + attribut.getNodeName());
                    Log.v(TAG, "    attribut " + (j+1) + " -> valeur : " + attribut.getNodeValue());
                }

                if(node.hasChildNodes())
                {
                    Node child = node.getFirstChild();
                    Log.v(TAG, "    contenu " + " -> type : " + getNomType(child.getNodeType()));
                    Log.v(TAG, "    contenu " + " -> valeur : " + child.getNodeValue());
                }
            }
        }
    }
    Log.v(TAG, "-----");
}

String nomElement = "string";
NodeList items = root.getElementsByTagName(nomElement);
Log.v(TAG, "Nb d'éléments \"" + nomElement + "\" = " + items.getLength());
for(int i = 0; i < items.getLength(); i++)
{
    // Le noeud
    Node item = items.item(i);
    Log.v(TAG, "élément " + (i+1) + " -> type : " + getNomType(item.getNodeType()));
    Log.v(TAG, "élément " + (i+1) + " -> nom : " + item.getNodeName());

    // ou l'élément :
    //Element element = (Element)items.item(i); // Element hérite de Node
    //Log.v(TAG, "élément " + (i+1) + " -> balise : " + element.getTagName());

    // Les attributs du noeud
    for(int j = 0; j < item.getAttributes().getLength(); j++)
    {
        Node attribut = item.getAttributes().item(j);
        Log.v(TAG, "    attribut " + (j+1) + " -> type : " + getNomType(attribut.getNodeType()));
        Log.v(TAG, "    attribut " + (j+1) + " -> nom : " + attribut.getNodeName());
        Log.v(TAG, "    attribut " + (j+1) + " -> valeur : " + attribut.getNodeValue());
```

```

        }

        // ou directement si on connaît le nom de l'attribut, ici name :
        //String name = item.getAttributes().getNamedItem("name").getNodeValue();
        //Log.v(TAG, "    attribut name : " + name);

        if(item.hasChildNodes())
        {
            Node child = item.getFirstChild();
            Log.v(TAG, "    contenu " + " -> type : " + getNomType(child.getNodeType()));
            Log.v(TAG, "    contenu " + " -> valeur : " + child.getNodeValue());
        }
    }
    Log.v(TAG, "-----");
}

catch (Exception e)
{
    texteView.setText("Document XML invalide !");
    e.printStackTrace();
}

```

On obtient ceci :

```

03-04 18:53:02.196 26721-26721/com.example.myapplicationxml V/XML: racine = resources
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: noeud -> type : ELEMENT_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: noeud -> nom : string
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> type : ATTRIBUTE_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> nom : name
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> valeur : app_name
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: contenu -> type : TEXT_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: contenu -> valeur : MyApplicationXML
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: noeud -> type : ELEMENT_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: noeud -> nom : string
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> type : ATTRIBUTE_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> nom : name
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> valeur : auteur
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: contenu -> type : TEXT_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: contenu -> valeur : Thierry Vaira
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: -----
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: Nb d'éléments "string" = 2
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: élément 1 -> type : ELEMENT_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: élément 1 -> nom : string
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> type : ATTRIBUTE_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> nom : name
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> valeur : app_name
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: contenu -> type : TEXT_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: contenu -> valeur : MyApplicationXML
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: élément 2 -> type : ELEMENT_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: élément 2 -> nom : string
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> type : ATTRIBUTE_NODE
03-04 18:53:02.197 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> nom : name
03-04 18:53:02.198 26721-26721/com.example.myapplicationxml V/XML: attribut 1 -> valeur : auteur
03-04 18:53:02.198 26721-26721/com.example.myapplicationxml V/XML: contenu -> type : TEXT_NODE
03-04 18:53:02.198 26721-26721/com.example.myapplicationxml V/XML: contenu -> valeur : Thierry Vaira
03-04 18:53:02.198 26721-26721/com.example.myapplicationxml V/XML: -----

```

Créer des données XML

L'objectif est de créer les données XML suivantes :

```

<?xml version="1.0" encoding="utf-8"?>
<utilisateurs>
    <utilisateur uid="500">toto</utilisateur>
    <utilisateur uid="501">titi</utilisateur>

```

```
</utilisateurs>
```

Pour créer un élément, on utilisera la méthode `createElement()` puis on l'ajoutera à l'arbre XML avec `appendChild()`. Pour ajouter un attribut, on utilisera `setAttributeNode()` ou `setAttribute()`. Et pour ajouter un contenu texte à un élément, on utilisera `createTextNode()`.

```
DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
try
{
    DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
    Document document = documentBuilder.newDocument();

    Element root = document.createElement("utilisateurs");
    document.appendChild(root);

    Element element1 = document.createElement("utilisateur");
    element1.appendChild(document.createTextNode("toto"));
    Attr attribut = document.createAttribute("uid");
    attribut.setValue("500");
    element1.setAttributeNode(attribut);
    root.appendChild(element1);

    Element element2 = document.createElement("utilisateur");
    element2.appendChild(document.createTextNode("titi"));
    element2.setAttribute("uid", "501");
    root.appendChild(element2);

    afficher(document);
}
catch (Exception e)
{
    texteView.setText("Document XML invalide !");
    e.printStackTrace();
}
```

On obtient :

```
03-04 19:06:50.498 28252-28252/com.example.myapplicationxml V/XML: <?xml version="1.0" encoding="UTF-8"?><utilisateurs>
<utilisateur uid="500">toto</utilisateur>
<utilisateur uid="501">titi</utilisateur>
</utilisateurs>
```

Écriture dans un fichier XML

```
String xmlFile = "exemple.xml";
String filePath = getBaseContext().getFilesDir().getPath() + "/" + xmlFile;
File file = new File(filePath);
if(!file.exists())
{
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    try
    {
        DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.newDocument();

        Element root = document.createElement("utilisateurs");
        document.appendChild(root);

        Element element1 = document.createElement("utilisateur");
        element1.appendChild(document.createTextNode("toto"));
        Attr attribut = document.createAttribute("uid");
```

```

attribut.setValue("500");
element1.setAttributeNode(attribut);
root.appendChild(element1);

Element element2 = document.createElement("utilisateur");
element2.appendChild(document.createTextNode("titi"));
element2.setAttribute("uid", "501");
root.appendChild(element2);

TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
transformer.setOutputProperty(OutputKeys.INDENT, "yes");
DOMSource source = new DOMSource(document);

String chemin = getBaseContext().getFilesDir().getPath().toString() + "/" + xmlFile;
StreamResult fichier = new StreamResult(new File(filePath));
transformer.transform(source, fichier);

}
catch (Exception e)
{
    e.printStackTrace();
}
}

```

Lecture d'un fichier XML

```

String xmlFile = "exemple.xml";
String filePath = getBaseContext().getFilesDir().getPath().toString() + "/" + xmlFile;
File file = new File(filePath);
if(file.exists())
{
    DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
    try
    {
        DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
        Document document = null;
        InputStream stream = null;
        try
        {
            stream = getBaseContext().openFileInput(xmlFile);
            InputSource inputSource = new InputSource(stream);
            document = documentBuilder.parse(inputSource);
        }
        catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }

        afficher(document);

        // ...
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
else
{
    Log.v(TAG, "Fichier introuvable !");
}

```

SAX

```

SAXHandler parser = new SAXHandler();

SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
SAXParser saxParser = null;
try
{
    saxParser = saxParserFactory.newSAXParser();
    XMLReader xmlReader = saxParser.getXMLReader();
    xmlReader.setContentHandler(parser);
    xmlReader.parse(getInputSourceXML());
}
catch (ParserConfigurationException e)
{
    e.printStackTrace();
}
catch (SAXException e)
{
    e.printStackTrace();
}
catch (IOException e)
{
    e.printStackTrace();
}

private InputSource getInputSourceXML()
{
    String s = null;
    s = new String("<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
        "<resources>" +
        "<string name=\"app_name\">MyApplicationXML</string>" +
        "<string name=\"auteur\">Thierry Vaira</string>" +
        "</resources>");
}

BufferedReader bufferedReader = new BufferedReader(new StringReader(s));
InputSource inputSource = new InputSource(bufferedReader);

return inputSource;
}

public class SAXHandler extends DefaultHandler
{
    private String balise = "";

    // détection d'un nouvel élément
    public void startElement(String nameSpace, String localName, String qName, Attributes attr) throws SAXException
    {
        balise = localName;
        Log.v(TAG, "début élément = " + localName);
    }

    // détection de la fin d'un élément
    public void endElement(String nameSpace, String localName, String qName) throws SAXException
    {
        balise = "";
        Log.v(TAG, "fin élément = " + localName);
    }

    // début du document
    public void startDocument()
    {
        Log.v(TAG, "Début document XML");
    }

    // fin du document XML
}

```

```

public void endDocument()
{
    Log.v(TAG, "Début document XML");
}

// les données
public void characters(char[] caracteres, int debut, int longueur) throws SAXException
{
    String donnees = new String(caracteres, debut, longueur);

    if (!balise.equals("")) && !donnees.isEmpty())
    {
        if (!Character.isISOControl(caracteres[debut]))
        {
            Log.v(TAG, "contenu = " + donnees);
        }
    }
}

```

On obtient :

```

03-04 19:38:17.204 31530-31530/com.example.myapplicationxml V/XML: Début document XML
03-04 19:38:17.204 31530-31530/com.example.myapplicationxml V/XML: début élément = resources
03-04 19:38:17.205 31530-31530/com.example.myapplicationxml V/XML: début élément = string
03-04 19:38:17.205 31530-31530/com.example.myapplicationxml V/XML: contenu = MyApplicationXML
03-04 19:38:17.205 31530-31530/com.example.myapplicationxml V/XML: fin élément = string
03-04 19:38:17.205 31530-31530/com.example.myapplicationxml V/XML: début élément = string
03-04 19:38:17.205 31530-31530/com.example.myapplicationxml V/XML: contenu = Thierry Vaira
03-04 19:38:17.205 31530-31530/com.example.myapplicationxml V/XML: fin élément = string
03-04 19:38:17.205 31530-31530/com.example.myapplicationxml V/XML: fin élément = resources
03-04 19:38:17.205 31530-31530/com.example.myapplicationxml V/XML: Début document XML

```

© Thierry Vaira <tvaira@free.fr>