# Introduction à SAX

**S**imple**A**PI for**X**ML ouSAX est une API générale pour la lecture d'un flux XML.

Il existe des implémentations de cette API dans tous les langages (en tout cas l'implémentation existe pour C++, C#, Java, Pascal, Perl, PHP…) car XML est largement utilisé aujourd'hui dans le monde du logiciel pour l'échange d'informations.

Il existe deux grandes API pour relire les flux XML : DOM et SAX.

SAX est événementielle (c'est celle que nous allons étudier) alors que DOM transforme l'arborescence XML en arborescence du langage cible. L'intérêt majeur de DOM est donc la possibilité qu'il offre d'aller et venir à votre gré dans l'arborescence, son inconvénient majeur reste la lourdeur du traitement, puisqu’il doit traiter l’ensemble du document.

En effet, SAX étant événementiel, son traitement se fait au fil du flux entrant.

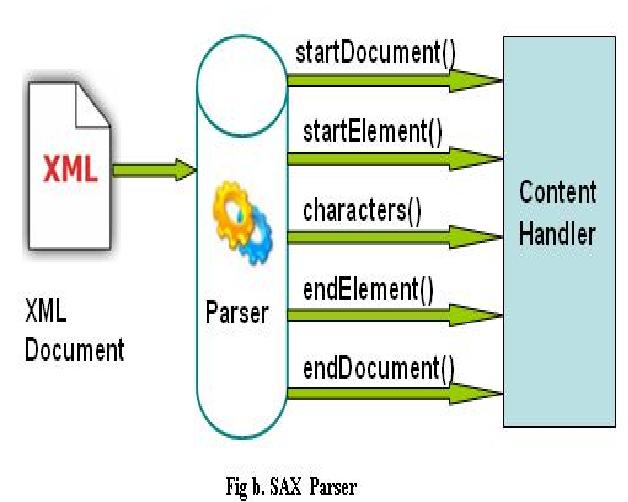
**A quoi correspondent les "évènements" dans un fichier XML ?**

Si l'on considère le flux XML entrant, il devient plus facile de découvrir les événements : une ***balise ouvrante*** est un ***événement***, une ***balise fermante un autre événement***…

Voilà pour tout dire l'essentiel de cette API, d'où son nom ***Simple*** API for XML, une fois que vous savez répondre à ces deux événements, vous pouvez déjà traiter un flux XML donc utiliser un fichier de configuration XML.

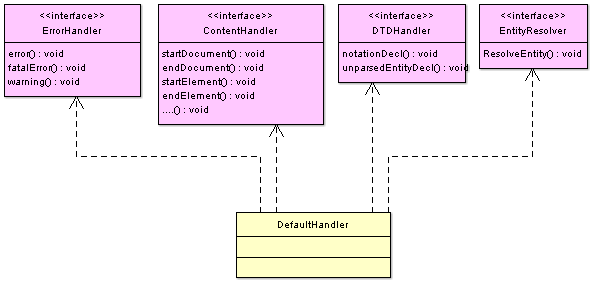
C'est étonnamment simple, et finalement extrêmement puissant comme nous allons le découvrir.

L'avantage d'une configuration XML est son aptitude contextuelle, laquelle est difficile à représenter dans un fichier de propriétés. En effet sous la forme de paires ***nom/valeur,***la notion contextuelle est généralement mal représentée par des clefs ayant une forme de chemin (***mon.chemin.vers.ma.propriété***).



### ContentHandler (gestionnaire de contenu)

Description de la classe : source openclassroom



Voyons pour commencer comment nous allons pouvoir réagir à ces fameux événements SAX générés par le parser. Pour ce faire, il nous suffira en fait

1. d'implémenter l'interface ContentHandler qui se trouve dans le package org.xml.sax, puis

2. d'instancier le parser en lui indiquant quel gestionnaire de contenu il devra utiliser (c'est-à-dire le nôtre).

Voici donc présenté un aperçu de l'interface centrale de vos développements SAX : le ContentHandler ou gestionnaire de Contenu. Ce gestionnaire est en effet celui qui fait le véritable travail d'analyse tandis que les autres interfaces de l'API font des travaux satellites comme gérer les erreurs, repérer le "curseur" dans le flux pendant l'analyse. Ces interfaces satellites ne sont pas à négliger, mais ont un rôle moins central dans votre développement.

Pour l'utiliser il faut donc définir une classe qui va implémenter ces méthodes particulières qui sont:

|  |  |  |
| --- | --- | --- |
| **startElement** | appelé lorsque le parser rencontre une nouvelle balise XML | les balises d'ouverture <maBalise> |
| **endElement** | appelé lorsque le parser rencontre une balise fermante XML | les balises de fermeture </maBalise> |

C'est essentiellement ces 2 méthodes qui vont nous intéresser ici mais il y en a d'autres qui permettent de traiter d'autres évènement:

|  |  |  |
| --- | --- | --- |
| startDocument | appelé lorsque le parser débute la lecture du document XML |  |
| endDocument | appelé lorsque le parser termine la lecture du document XML |  |
| startPrefixMapping | appelé lorsque le parser rencontre le début d'un espace de nommage (namespace) |  |
| endPrefixMapping | appelé lorsque le parser rencontre la fin d'un espace de nommage (namespace) |  |
| **characters** | appelé lorsque le parser rencontre des caractères à l'intérieur d'un élément | pour les caractères "mes caractères" dans |
| ignorableWhitespace | appelé lorsque le parser rencontre des caractères blancs (espaces) qui peuvent être ignorés | les espaces utilisés pour indenter le XML |
| processingInstruction | appelé lorsque le parser rencontre une instruction dans le flux XML |  |
| skippedEntity | appelé lorsque le parser saute une entité |  |

Il existe également des méthodes pour la gestion des erreurs:

|  |  |
| --- | --- |
| warning | appelé lorsque le parser rencontre un warning |
| error | appelé lorsque le parser rencontre une erreur non bloquante |
| fatalError | appelé lorsque le parser rencontre une erreur de parsing fatale |

Donc voilà pour le principe maintenant pour utiliser le parser il faut définir son propre handler, c'est à dire une classe qui va implémenter les méthodes dont nous avons besoin.

### Localisation du document: setDocumentLocator

Un locator vous permet de localiser "le curseur" pendant le traitement du flux vous permettant par exemple de connaître le numéro de ligne et de colonne en cours d'analyse. Ceci est une fonctionnalité certes très intéressante au moment du débogage, mais qu'il faut à tout prix éviter. Dans les Helperde l'API SAX, il vous est fourni une implémentation par défaut de toutes les interfaces. Si l'implémentation par défaut du ContentHandler est proprement inutile, celle du Locator devrait amplement suffire à 99 % des développements, on ne reviendra donc pas dessus.

### startDocument

Cette méthode est appelée par le parser une et une seule fois au démarrage de l'analyse de votre flux XML. Elle est appelée avant toutes les autres méthodes de l'interface, à l'exception unique, évidemment, de la méthode setDocumentLocator. Cet événement doit permettre d'initialiser tout ce qui doit l'être avant le début du parcours du document.

### endDocument

Cette méthode est complméentaire de startDocument et donc appelée à la fin du parcours du flux après toutes les autres méthodes. Il peut alors être utile à ce moment de notifier à d'autres objets du fait que le travail est terminé.

### processingInstruction

Cet événement est levé pour chaque instruction de fonctionnement rencontrée. Ces instructions sont celles que vous trouvez hors de l'arbre XMLlui-même comme les instructions concernant les DTD ou plus simplement la déclaration :

<?xml version="1.0" encoding="ISO-8859-1" ?>

### startPrefixMapping

Cet événement est lancé à chaque fois qu'un mapping préfixé, c'est-à-dire une balise située dans un espace de nommage***(name space),***est rencontré.

### endPrefixMapping

Son événement contraire évidemment, c'est-à-dire la fin du traitement dans un espace de nommage.

### startElement

Démarrage d'un élément XML…

Nous allons approfondir cet événement fondamental

 startElement (String namespaceUri, String localName, String rawName, Attributs atts);

* où nameSpaceUri est la chaîne de caractères contenant l'URI complète de l'espace de nommage du tag ou une chaîne vide si le tag n'est pas compris dans un espace de nommage,
* localName est le nom du tag sans le préfixe s'il y en avait un,
* rawName est le nom du tag version XML 1.0 c'est-à-dire $prefix :$localname,
* enfin attributs est la liste des attributs du tag que l'on étudiera un peu plus loin.

### endElement

Événement inverse de signature beaucoup plus simple puisque seul le nom complet du tag a besoin d'être connu. En effet, à la fermeture de la balise XML, aucun attribut n'est requis.

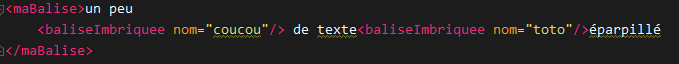
### characters

Tout ce qui est dans l'arborescence, mais n'est pas partie intégrante d'un tag, déclenche la levée de cet événement. En général, cet événement est donc levé tout simplement par la présence de texte entre la balise d'ouverture et la balise de fermeture comme dans l'exemple suivant :

<maBalise>un peu de texte</maBalise>

La présence de "un peu de texte" provoque la levée de l'événement characters.

Attention : il est à noter que l'API SAX n'impose rien quant à l'implémentation de cet événement. Dans le cas d'un texte épars autour de balises filles de la balise en cours, les réactions peuvent être diverses. Ainsi le flux XML suivant :



Il peut soit donner lieu à trois événements contenant respectivement le texte "un peu", " de texte", "éparpillé" soit donner un seul événement contenant l'intégralité du texte à savoir "un peu de texte éparpillé". Comme l'API ne fixe rien, ce sera à vous de penser au fait que le parser que vous avez sous la main ne sera peut-être pas celui de vos clients et d'agir en conséquence, c'est-à-dire en gérant les deux types de réactions possibles de telle sorte qu'elles fournissent le même comportement final dans les deux cas.

## Références

Les liens suivants proposent davantage de détails sur la manipulation des documents XML.

<https://www.baeldung.com/java-xpath>

<https://www.baeldung.com/java-xml-libraries>

Concernant la librairie Sax : <https://openclassrooms.com/fr/courses/2654406-java-et-le-xml/2686485-un-peu-de-sax>