# SAX

# SAX

- SAX (Simple API for XML) est une recommandation de fait.
- Collaboration de développeurs de parseurs XML, liste de diffusion : XML-Dev
- Objectif : fournir une API simple et standard pour accéder au contenu d'un document.
- Principe : Associer des événements aux balises

# SAX

- SAX, contrairement à DOM, n'implémente pas d'arbre en mémoire
- Orienté traitement de gros document XML
- Les spécifications de SAX, qui ne sont pas gérés par le W3C

http://www.megginson.com/SAX/index.html

- Il existe deux versions de SAX :
  - SAX1 (11 Mai 1998) supportée par tous les parseurs XML en Java, mais sans espaces de noms,
  - SAX2 (05 Mai 2000), ajoute espaces de noms et configuration des parseurs SAX et son utilisation en Java

# SAX

- SAX est basé sur un modèle événementiel,
  - l'analyseur appelle automatiquement une méthode lorsqu'un événement est détecté
- Voici les 5 évènements détectés par SAX ainsi que les méthodes qui sont appelées :
  - Détection d'une balise de début ***startElement()***
  - Détection d'une balise de fin ***endElement()***
  - Détection de données entre deux balises ***characters()***
  - Début du traitement du document XML ***startDocument()***
  - Fin du traitement du document XML ***endDocument()***

# SAX

**Quand utiliser SAX ?**

- Document XML volumineux
- Lecture rapide et efficace
  - SAX gère les fichiers XML comme un flux de données (lecture séquentiel : pas de retour ni saut)
- Economie de mémoire

# SAX

- Java possède depuis le JDK 1.4 deux API permettant la gestion  de fichiers XML regroupées dans les packages JAXP
- Comment utiliser SAX
  - Importer les classes
  - Implémenter l'interface ContentHandler
  - Paramétrer l'analyseur
  - Manipuler les événements de contenu

# SAX

**Importer les classes**

- Il faut importer les classes suivantes afin d'utiliser l'API SAX pour l'accès au fichier XML :

  org.xml.sax.*
  org.xml.sax.helpers.DefaultHandler
  javax.xml.parsers.SAXParserFactory
  javax.xml.parsers.ParserConfigurationException
  javax.xml.parsers.SAXParser

# SAX

**Importer les classes**

- Le package **org.xml.sax** définit les interfaces concernant l'analyseur SAX.
- La classe **SAXParserFactory** crée l'instance à utiliser.
- **ParserConfigurationException** exception levée si échec de création d'un analyseur qui correspond à l'option de configuration.

# SAX

**Implémenter l'interface ContentHandler**

- nécessite certaines méthodes que l'analyseur SAX invoque lorsque certains événements sont détectés pendant l'analyse.
- les méthodes les plus utilisées pour réagir aux événements sont ***startDocument()***, ***endDocument()***, ***startElement()***, ***endElement()***, et ***characters()***.

# SAX

**Implémenter l'interface ContentHandler**

- Implémentation facile : hériter de la classe **DefaultHandler** définie dans le package **org.xml.sax.helpers**.
  - Cette classe fournit des méthodes vides pour tous les événements du **ContentHandler**.

# SAX

**Implémenter l'interface ContentHandler**

- Sinon : il faut redéfinir les méthodes suivantes :
  - ***public void setDocumentLocator(Locator value)***
  - ***public void startDocument() throws SAXException***
  - ***public void endDocument() throws SAXException***
  - ***public void startPrefixMapping(String prefix, String URI) throws SAXException***
  - ***public void endPrefixMapping(String prefix) throws SAXException***
  - ***public void startElement(String nameSpaceURI, String localName, String rawName, Attributes attributs) throws SAXException***
  - ***public void endElement(String nameSpaceURI, String localName, String rawName) throws SAXException***
  - ***public void characters(char[] ch, int start, int end) throws SAXException***
  - ***public void ignorableWhitespace(char[] ch, int start, int end) throws SAXException***
  - ***public void processingInstruction(String target, String data) throws SAXException***
  - ***public void skippedEntity(String arg0) throws SAXException***

# SAX

**Implémenter l'interface ContentHandler**

- Toutes les méthodes à implémenter doivent lever une **SAXException**
- Quand une balise de début ou de fin est rencontré, son nom est passé comme chaîne de caractères aux méthodes ***startElement()*** ou ***endElement()***.
- Quand une balise de début est rencontré, les attributs qu'elle possède sont aussi passés dans une liste d'attribut (**Collection**).
- Les caractères trouvés dans l'élément sont passés en tant que tableau de caractères avec le nombre de caractères et le décalage (offset) dans le tableau qui pointe sur le premier caractère.

# SAX

**Paramétrer l'analyseur**

```
...
   DefaultHandler handler =  new AnalyseSAX();
   SAXParserFactory factory;
    factory = SAXParserFactory.newInstance();


   SAXParser saxParser = factory.newSAXParser();
   saxParser.parse( new File(args[0]), handler);
...
```

# SAX

**Manipuler les événements de contenu**

- **Les événements de document**

```
public void startDocument() throws SAXException {
        …
}

public void endDocument() throws SAXException {
        …
}
```

# SAX

**Manipuler les événements de contenu**

- **Les événements des éléments**

```
public void startElement(       String namespaceURI,
                                String sName, String qName,
                                Attributes attrs) throws SAXException {
        ...
}
public void endElement(         String namespaceURI,
                                String sName, String qName      )
                                throws SAXException {
         ...
}
```

# SAX

**Manipuler les événements de contenu**

- **Les événements des éléments**
- signification des arguments :
  - uri 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,
  - sName est le nom du tag sans le préfixe s'il y en avait un,
  - qName est le nom du tag version XML 1.0 c'est à dire $prefix:$sName,
  - attributs est la liste des attributs du tag

# SAX

**Manipuler les événements de contenu**

- **Les événements des éléments**
- Pour les attributs :
  - ***int getLength()*** : Retourne le nombre d'attributs dans la liste.
  - ***String getLocalName(int index)*** : Récupère le nom d'un attribut à partir de l'index
  - ***String getValue(int index)*** : Récupère la valeur de l'attribut à partir de l'index.

# SAX

**Manipuler les événements de contenu**

- **Les événements de caractère**

```
public void characters(char[] ch, int start, int length)
                                                throws SAXException {
                ...
}
```

- Signification des arguments :
  - *ch* : le tableau de caractères
  - *Start :* L'indice de départ dans le tableau de caractères
  - *Length* : Le nombre de caractères à récupérer du tableau.

# SAX : exemple

- Soit un fichier XML

```
<personne>
    <nom>Dupond</nom>
    <adresse>
        <numero>3</numero>
        <rue>rue de la paix</rue>
        <ville>Paris</ville>
        <codePostal>75001</codePostal>
    </adresse>
</personne>
```

# SAX : exemple

```
// Parse un document XML en JAVA avec l'Api SAX
//on importe les API necessaires
//pour l'analyse du XML
import org.xml.sax.*;
import org.xml.sax.helpers.DefaultHandler;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
//pour SOP
import java.io.*;
```

# SAX : exemple

```
Public class AnalyseSAX extends DefaultHandler {

…
    public void startDocument () throws SAXException {
            System.out.println("début du document");
        }
    public void endDocument () throws SAXException {
            System.out.println("fin du document");
        }
```

# SAX : exemple

```
Public class AnalyseSAX extends DefaultHandler {

…
    public void startElement (String namespaceURI,String
     simpleName,String qualifiedName,Attributes attrs) throws
     SAXException {
            String nomElement = simpleName;
            if (nomElement.equals("")) nomElement = qualifiedName;
            System.out.println("startElement : "+ nomElement);
        }
```

# SAX : exemple

```
Public class AnalyseSAX extends DefaultHandler {

…
 public void endElement (String namespaceURI,String
    simpleName,String qualifiedName) throws SAXException {
          String nomElement = simpleName;
          if (nomElement.equals("")){
             nomElement = qualifiedName;
          }
          System.out.println("endElement : "+ nomElement);
       }
```

# SAX : exemple

```
Public class AnalyseSAX extends DefaultHandler {

…

    public void characters (char buf [], int offset, int len)
            throws SAXException {
            String s = new String(buf, offset,len);
            System.out.println (s);
        }
```

# SAX : exemple

```
Public class AnalyseSAX extends DefaultHandler {

    …
        DefaultHandler handler = new AnalyseSAX();
        SAXParserFactory factory =
    SAXParserFactory.newInstance();
        try {
            SAXParser saxParser = factory.newSAXParser();
            saxParser.parse( new File(args[0]), handler );
        } catch (Throwable t) {
            t.printStackTrace ();
            System.exit (1);
        }
        System.exit (0);
      }
```

# SAX : exemple

*début du document*
*startElement : personne*
*startElement : nom*
*Dupond*
*endElement : nom*
*startElement : adresse*
*startElement : numero*
*3*
*endElement : numero*
*startElement : rue*
*rue de la paix*
*endElement : rue*
*startElement : ville*
*Paris*
*endElement : ville*
*startElement : codePostal*
*75001*
*endElement : codePostal*
*endElement : adresse*
*endElement : personne*
*fin du document*

```
<personne>
        <nom>Dupond</nom>
        <adresse>
                <numero>3</numero>
                <rue>rue de la paix</rue>
                <ville>Paris</ville>

        <codePostal>75001</codePostal>
        </adresse>
</personne>
```