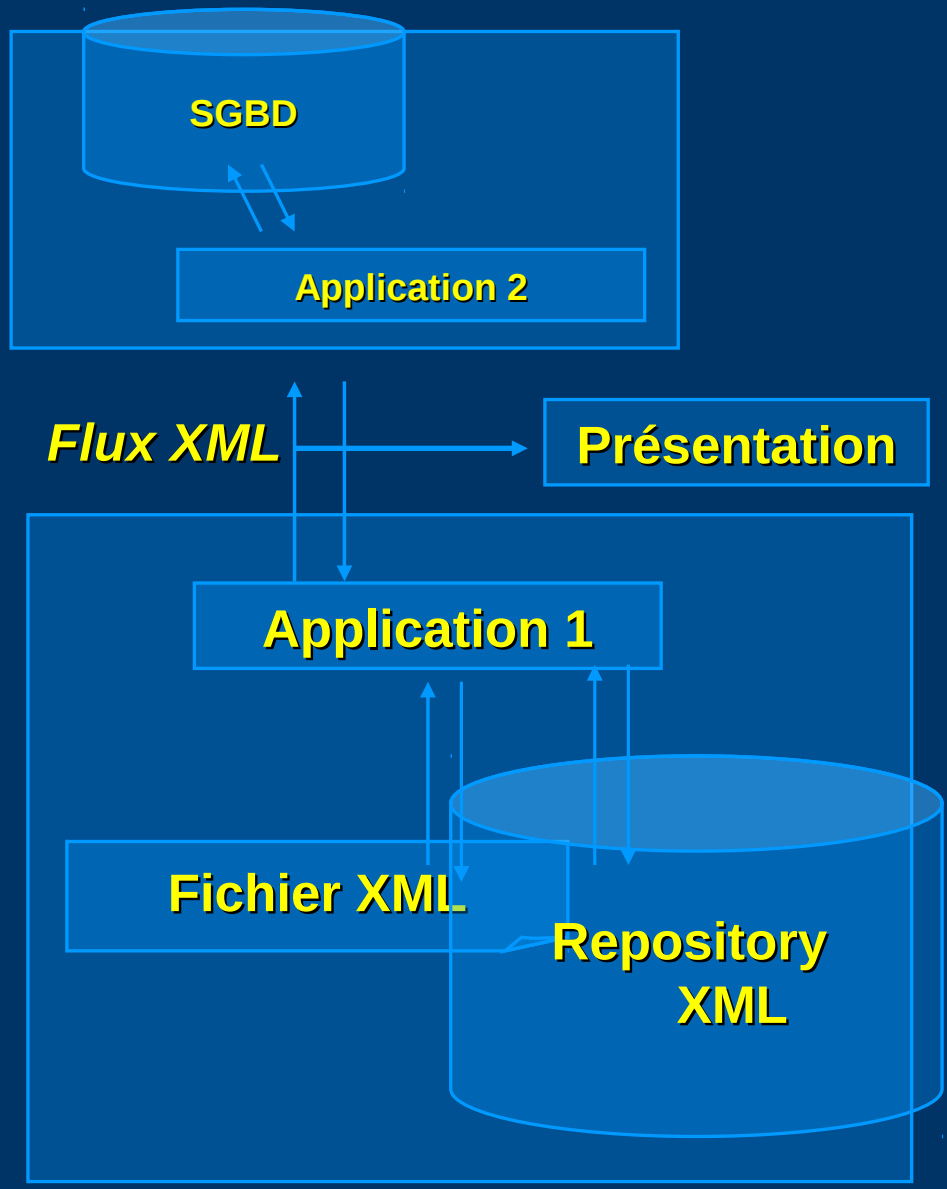
Programmation XML

- Pour quoi faire ?
  - Format de stockage
    - Plus puissant que les fichiers fixes
    - Plus standard que les fichiers formatés
    - Plus souple et plus « simple » qu’une base de données
  - Format d’échange
    - Entre applications
    - Vers un format standard

Un programme utilisant XML

- XML comme format d'échange
- Stockage de documents XML
  - Fichiers textes
  - Repository natif
- Manipulation de données XML



Objectifs

- Analyser un document XML
- Manipuler un ou plusieurs documents XML
  - Représenter un document
  - Localiser et des fragments
  - Créer des fragments
- Créer un document XML

The Simple API for XML

http://www.saxproject.org/

Présentation de SAX

- API évènementielle de parsing de documents XML
- Développée en collaboration par les membres de la liste XML-DEV
- A l'origine uniquement disponible pour Java
  - Disponible aussi avec Perl, C, …
- SAX2: version plus récente supportant les namespaces
- Nombreux parseurs publics
  - XP de James Clark, Aelfred, Saxon
  - MSXML4 de Microsoft
  - Xerces et Crimson de Apache

API évènementielle

- Rapporte les évènements liés à l'analyse syntaxique:
  - ouverture d'élément
  - Fin d'élément
  - …
- L'arbre du document n'est pas construit
  - Exemple : Recherche les éléments `Personne` contenant le texte « `durand` »

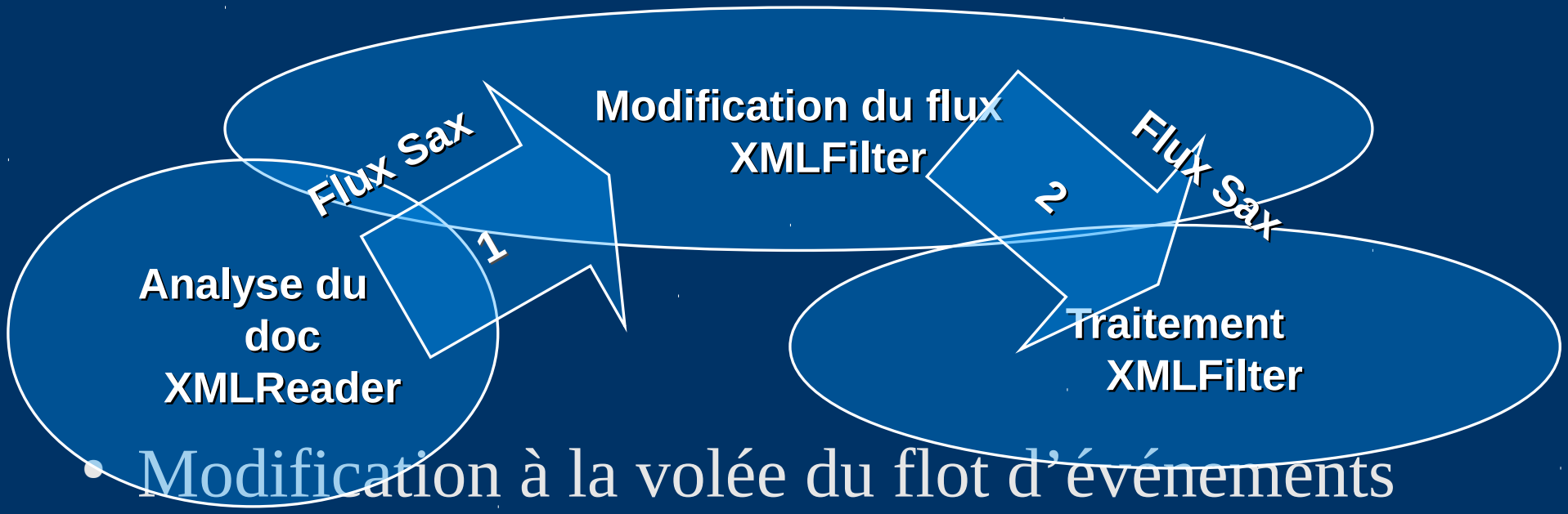
Les interfaces essentielles (SAX2)

- XMLReader
  - setContentHandler
  - setErrrorHandler
  - parse
- ContentHandler
  - startDocument
  - endDocument
  - startElement
  - endElement
  - characters
- InputSource
- ErrorHandler
  - fatalError
  - error
  - warning

Utilisation de SAX

- Définir une classe qui indique comment réagir aux évènements de l'analyse du document
  - Sous-classe de **ContentHandler**
- Utiliser cette classe de la manière suivante:
  - Instancier la classe **XMLReader**
    - (choix d'une implantation de sax)
  - Instancier la **InputSource** pour accéder au document XML source
  - Associer une instance de la sous-classe de **ContentHandler** au **XMLReader**
  - L'analyse est lancée avec la méthode ***parse(inputSource)***
- Les méthodes de la classe sont appelées par le XMLReader au cours du traitement

Les filtres SAX



- Modification à la volée du flot d'événements
- Fonctionnement en chaîne

Exemple d’analyse avec SAX

- ```
  <?xml version="1.0"?>
   <doc>
     <para>Hello, world!</para>
   </doc>
  ```

- ```
  start document
  start element: doc
  start element: para
  characters: Hello, world!
  end element: para
  end element: doc
  end document
  ```

Implantation en Java

- ```
  import java.io.FileReader; import org.xml.sax.XMLReader; import org.xml.sax.Attributes;
  import org.xml.sax.InputSource; import org.xml.sax.helpers.XMLReaderFactory;
  import org.xml.sax.helpers.DefaultHandler;

  public MySAXApp () { super(); }

  public class MySAXApp extends DefaultHandler {
    public static void main (String args[]) throws Exception {
      XMLReader xr = XMLReaderFactory.createXMLReader();
      MySAXApp handler = new MySAXApp();
      xr.setContentHandler(handler);
      xr.setErrorHandler(handler);
      FileReader r = new FileReader(args[0]);
      xr.parse(new InputSource(r)); } }

  public void startDocument () { System.out.println("Start document"); }

  public void endDocument () { System.out.println("End document"); }

  public void startElement (String uri, String name, String qName, Attributes atts) {
              if ("".equals (uri)) System.out.println("Start element: " + qName);
              else System.out.println("Start element: {" + uri + "}" + name); }

  public void endElement (String uri, String name, String qName) {
              if ("".equals (uri)) System.out.println("End element: " + qName);
              else System.out.println("End element: {" + uri + "}" + name); }

  public void characters (char ch[], int start, int length) {
              System.out.print("Characters: \"");
              System.out.print(ch[i]);
              System.out.print("\"\n"); }
  }
  ```
  ```

Conclusion sur SAX

- Interfaces événementielles simples
  - Fonctionnement en pipe-line
  - Faible coût en mémoire
  - Bonnes performances
- Difficile à utiliser si
  - Traitement multiple sur un document
  - Utilisation de la structure arborescente
    - Analyse avec un automate (Changement d’états)

Document Object Model (DOM)

Document Object Model Level 2
(W3C Recommendation)
13 November 2000

Document Object Model Level 2 Core
Document Object Model Level 2 Views
Document Object Model Level 2 Events
Document Object Model Level 2 Style
Document Object Model Level 2 Traversal and Range

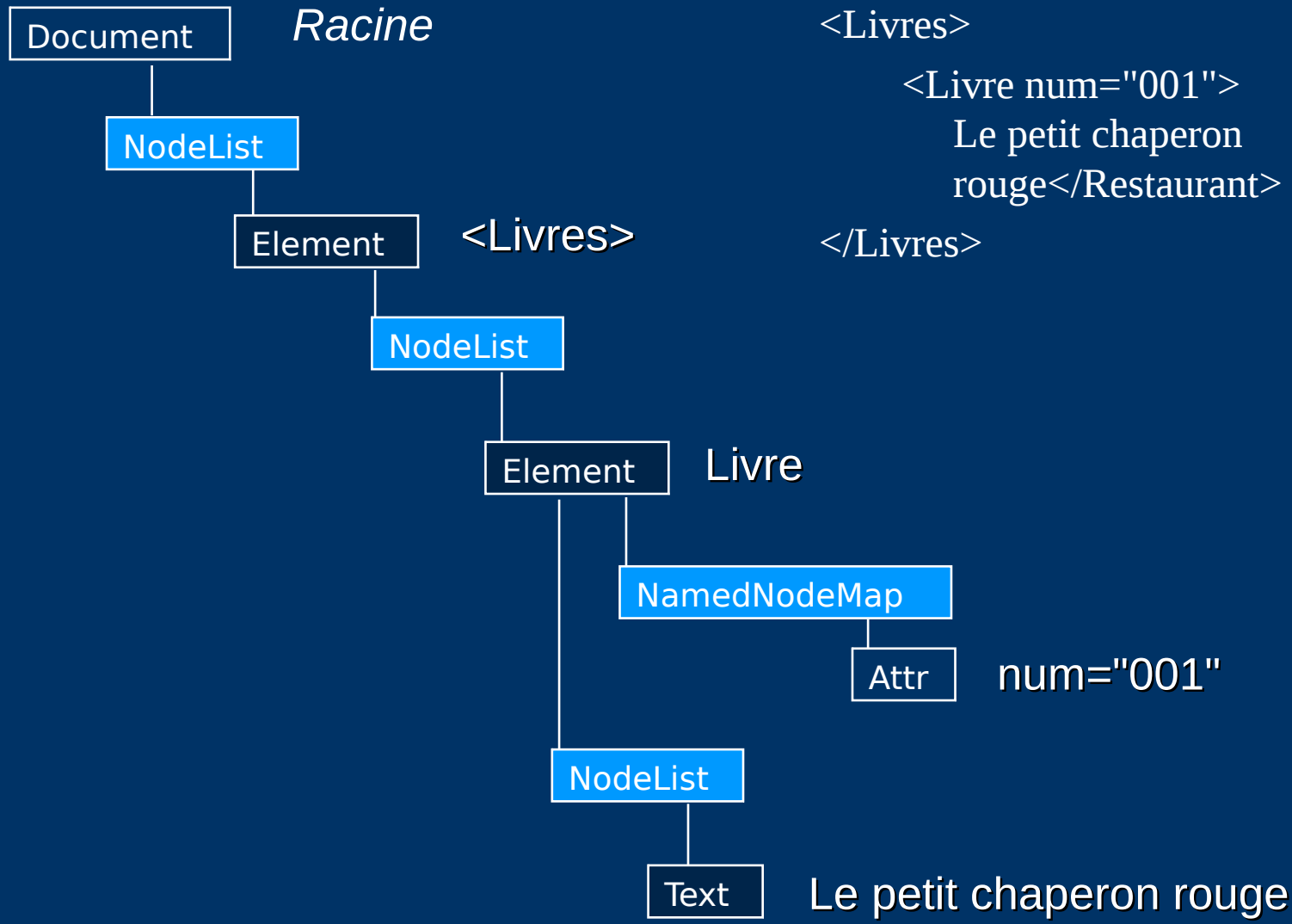
Introduction au DOM

- Interface standard d’accès à un document XML
- Indépendant
  - Du langage de programmation
  - De la plateforme
- Différents bindings: Java, C++, …
- Définit une interface:
  - Un modèle arborescent (ensemble de nœuds)
  - Une API

DOM une API pour documents

- Standard W3C
- Fonctionne avec HTML et XML
- Propose un modèle pour représenter un document
  - Produit par un "parser"
- Interface de navigation
  - Définie en IDL CORBA
  - Peut être utilisée en:
    - Java, C++
    - C#, VB
    - Python

Un d'arbre DOM



DOM, une forêt de nœuds

- Navigation via un arbre générique de nœuds
  - **Node**
  - **NodeList** (Parent/Child)
  - **NamedNodeMap** (attributs)
- Tout nœud hérite de **Node**

Nœuds DOM et leurs fils

- **Document** → Element (maximum of one), ProcessingInstruction, Comment, DocumentType (maximum of one)
- **DocumentFragment** → Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- **DocumentType** → no children
- **EntityReference** → Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- **Element** → Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference
- **Attr** → Text, EntityReference
- **ProcessingInstruction** → no children
- **Comment** → no children
- **Text** → no children
- **CDATASection** → no children
- **Entity** → Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- **Notation** → no children

Le Dom définit aussi une structure de liste de nœuds : NodeList.

Les interfaces DOM

- Interfaces fondamentales
  - DOMImplementation
  - Document
  - Comment
  - DocumentFragment
  - Element
  - Attr(ibute)
  - NamedNodeMap
  - CharacterData
    - Comment
    - Text
- Interfaces étendues XML
  - ProcessingInstruction
  - DocumentType
  - CDATASection
  - Notation
  - Entity
  - EntityReference

Mise en oeuvre - Navigation

- A partir de l'instance de Document
  - getDocumentElement() *obtenir l'élément racine*
- Pour un élément
  - getName(), getLocalName(), getNamespaceURI() : *obtenir le nom et l'espace de nom de l'élément*
  - getChildNodes() : *obtenir la liste des noeuds fils*
  - getFirstNode(), getLastNode(), getNextSibling() : *navigation*
  - getAttributes() : *obtenir la table de hachage des attributs*
- Pour les attributs
  - getName() : *obtenir le nom de l'attribut*
  - getValue() : *obtenir la valeur de l'attribut*
- Pour les noeuds texte
  - getNodeValue() : obtenir *valeur du noeud texte*

Méthodes de parcours dans DOM

- NodeIterator

```
NodeIterator iter= ((DocumentTraversal)document).createNodeIterator(
     root,
     NodeFilter.SHOW_ELEMENT, null);
while (Node n = iter.nextNode()) printMe(n);
```

- TreeWalker

```
processMe(TreeWalker tw) {
     Node n = tw.getCurrentNode();
     nodeStartActions(tw);
     for (Node child=tw.firstChild(); child!=null; child=tw.nextSibling())
             { processMe(tw); }
     tw.setCurrentNode(n);
     nodeEndActions(tw);
}
```

- NodeFilters

```
NamedAnchorFilter myFilter = new NamedAnchorFilter();
NodeIterator iter=
     ((DocumentTraversal)document).createNodeIterator( node,
     NodeFilter.SHOW_ELEMENT, myFilter);
```

Mise en œuvre - Construction

- Créer une instance de **Document**
  - En général, dépend de l'implémentation
- Construire les nœuds de l'arbre
  - L'instance de **Document** sert d'usine à objets (*factory*)
  - *createElementNS(namespaceURI, qName), createAttributeNS(…)* …
- Ajouter des nœuds en fixant des liens avec les nœuds existants.
  - *appendChild(node), replaceChild(node1, node2), insertBefore(node)*
  - *setAttributeNS(…)*

Bilan DOM

- Une interface objet standard
  - Navigation
  - Construction
- Des concepts simple
  - Interface vaste mais intuitive
- Performances actuellement limitées
  - Coût mémoire important

Interface IDL pour le nœud document

```
interface Document : Node {
  readonly attribute DocumentType     doctype;
  readonly attribute DOMImplementation  implementation;
  readonly attribute Element          documentElement;
  Element           createElement(in DOMString tagName)
                                raises(DOMException);
  DocumentFragment   createDocumentFragment();
  Text              createTextNode(in DOMString data);
  Comment           createComment(in DOMString data);
  CDATASection      createCDATASection(in DOMString data)
                                raises(DOMException);
  ProcessingInstruction createProcessingInstruction(in DOMString target,
                                        in DOMString data)
                                raises(DOMException);
  Attr              createAttribute(in DOMString name)
                                raises(DOMException);
  EntityReference    createEntityReference(in DOMString name)
                                raises(DOMException);
  NodeList          getElementsByTagName(in DOMString tagname);
  // Introduced in DOM Level 2:
  Node              importNode(in Node importedNode,
                                in boolean deep)
                                raises(DOMException);
  // Introduced in DOM Level 2:
  Element           createElementNS(in DOMString namespaceURI,
                                in DOMString qualifiedName)
                                raises(DOMException);
  // Introduced in DOM Level 2:
  Attr              createAttributeNS(in DOMString namespaceURI,
                                in DOMString qualifiedName)
                                raises(DOMException);
  // Introduced in DOM Level 2:
  NodeList          getElementsByTagNameNS(in DOMString namespaceURI,
                                in DOMString localName);
  // Introduced in DOM Level 2:
  Element           getElementById(in DOMString elementId);
};
```

Interface IDL pour les nœuds element

```
interface Element : Node {
  readonly attribute DOMString       tagName;
  DOMString          getAttribute(in DOMString name);
  void               setAttribute(in DOMString name,
                                  in DOMString value)
                                     raises(DOMException);
  void               removeAttribute(in DOMString name)
                                     raises(DOMException);
  Attr               getAttributeNode(in DOMString name);
  Attr               setAttributeNode(in Attr newAttr)
                                     raises(DOMException);
  Attr               removeAttributeNode(in Attr oldAttr)
                                     raises(DOMException);
  NodeList           getElementsByTagName(in DOMString name);
  // Introduced in DOM Level 2:
  DOMString          getAttributeNS(in DOMString namespaceURI,
                                    in DOMString localName);
  // Introduced in DOM Level 2:
  void               setAttributeNS(in DOMString namespaceURI,
                                    in DOMString qualifiedName,
                                    in DOMString value)
                                     raises(DOMException);
// Introduced in DOM Level 2:
  void               removeAttributeNS(in DOMString namespaceURI,
                                       in DOMString localName)
                                       raises(DOMException);
  // Introduced in DOM Level 2:
  Attr               getAttributeNodeNS(in DOMString namespaceURI,
                                        in DOMString localName);
  // Introduced in DOM Level 2:
  Attr               setAttributeNodeNS(in Attr newAttr)
                                        raises(DOMException);
  // Introduced in DOM Level 2:
  NodeList           getElementsByTagNameNS(in DOMString namespaceURI,
                                            in DOMString localName);
  // Introduced in DOM Level 2:
  boolean            hasAttribute(in DOMString name);
  // Introduced in DOM Level 2:
  boolean            hasAttributeNS(in DOMString namespaceURI,
                                    in DOMString localName);
```