

Programmation XML

Yves Roos
Bureau 220 - M3.ext
yves.roos@ifil.fr

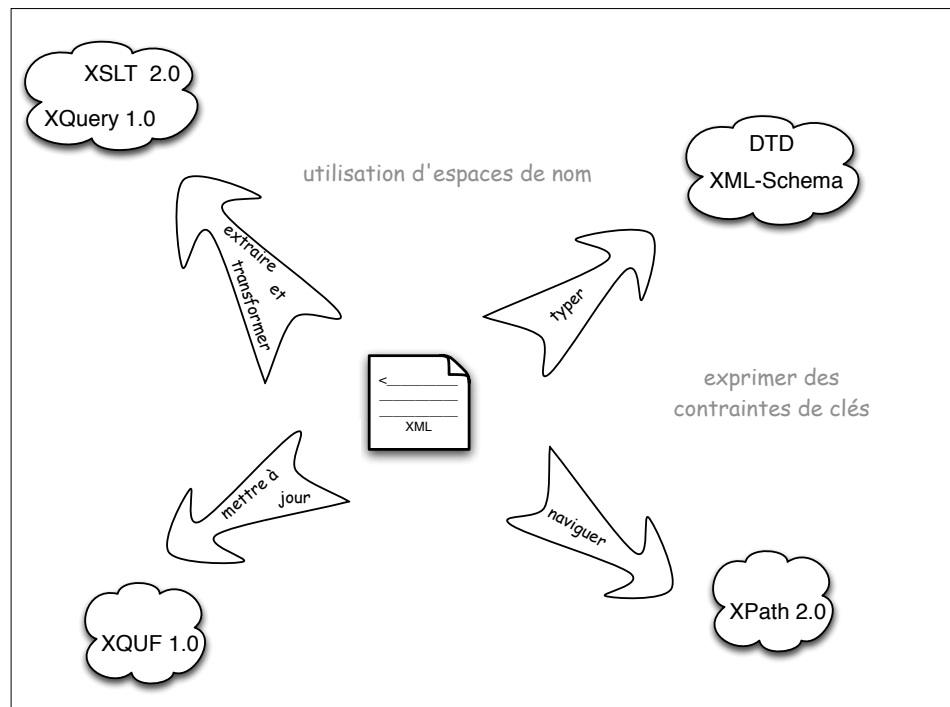
Master MIAGE M1 2011-2012

XML est human readable

```
<?xml version="1.0" encoding="UTF-8"?>  
<library>  
  <items>  
    <bd uuid="8B5F233B-0B45-4907-A106-AA72840F639B">  
      <illustrator>Roba</illustrator>  
      <numberInSeries>5</numberInSeries>  
      <pages>60</pages>  
      <publisher>Dupuis</publisher>  
      <series>Boule et Bill -1-</series>  
      <title>60 gags de Boule et Bill n°5</title>  
    </bd>  
    <bd uuid="28F20F96-C938-41BA-96AB-BAFADC22D80E">  
      <illustrator>Mezières</illustrator>  
      <numberInSeries>12</numberInSeries>  
      <pages>48</pages>  
      <publisher>Dargaud</publisher>  
      <series>Valérian</series>  
      <title>Les foudres d'Hypsis</title>  
    </bd>  
    <bd uuid="DCA1F83F-ADA9-45BD-AB19-EA991120DF53">  
      <illustrator>Magniaux</illustrator>  
      <pages>48</pages>  
      <publisher>Editions Williams</publisher>  
      <series>Charlot (Williams)</series>  
      <title>La ruée vers l'or</title>  
    </bd>
```

XML est human readable

```
<?xml version="1.0" encoding="UTF-8"?> <library> <items> <book uuid="8B5F233B-0B45-4907-A106-AA72840F639B" boxWidthInInches="40E-1" boxHeightInInches="1.17E1" used="0" boxLengthInInches="8.27E0" illustrator="Roba" fullTitle="60 gags de Boule et Bill n°5" numberInSeries="5" rating="00E0" published="1969-01-01T11:00:00" series="Boule et Bill -1-" location="BouleEtBill5w_30012005.jpg" hasExperienced="0" minutes="0" listened="0" boxWeightInPounds="00E0" mediaccount="1" signed="0" read="0" country="fr" watched="0" netrating="00E0" rare="0" title="60 gags de Boule et Bill n°5" pages="60" edition="Dupuis" upc="000000008020" played="0" players="0" lastLookupTime="286237024"></book> <book uuid="D1DDD65E-7E7C-4E51-9B76-5BFCAB88EEFB" boxWidthInInches="40E-1" boxHeightInInches="1.17E1" used="0" boxLengthInInches="8.27E0" illustrator="&lt;Collectif>" fullTitle="Nous, Tintin" numberInSeries="0" rating="00E0" published="1987-01-01T11:00:00" series="Tintin - Divers" location="NousTintin.jpg" hasExperienced="0" minutes="0" listened="0" boxWeightInPounds="00E0" mediaccount="1" signed="0" read="0" country="fr" watched="0" netrating="00E0" rare="0" title="Nous, Tintin" pages="0" edition="Les éditions du Lion" upc="0000000007566" played="0" players="0" lastLookupTime="286236999"></book> <book uuid="28F20F96-C938-41BA-96AB-BAFADC22D80E" boxWidthInInches="40E-1" boxHeightInInches="1.17E1" used="0" boxLengthInInches="8.27E0" illustrator="Mezières" fullTitle="Les foudres d'Hypsis" numberInSeries="12" rating="00E0" published="1985-01-10T11:00:00" series="Valérian" location="valeriancouv12.jpg" hasExperienced="0" minutes="0" listened="0" boxWeightInPounds="00E0" mediaccount="1" signed="0" read="0" country="fr" watched="0" netrating="00E0" rare="0" title="Les foudres d'Hypsis" pages="48" edition="Dargaud" upc="9782205030327" played="0" players="0" lastLookupTime="286236581"></book> <book uuid="DCA1F83F-ADA9-45BD-AB19-EA991120DF53" boxWidthInInches="40E-1" boxHeightInInches="1.17E1" used="0" boxLengthInInches="8.27E0" illustrator="Magniaux" fullTitle="La ruée vers l'or" numberInSeries="2" rating="00E0" published="1974-01-04T11:00:00" series="Charlot (Williams)" location="charlotwilliamsruéesverslor.jpg" hasExperienced="0" minutes="0" listened="0" boxWeightInPounds="00E0" mediaccount="1" signed="0" read="0" country="fr" watched="0" netrating="00E0" rare="0" title="La ruée vers l'or" pages="48" edition="Editions Williams" upc="0000000007795" played="0" players="0" lastLookupTime="286237014"></book>
```



Plan (prévisionnel)

- Les principes de bases de XML
- Typer des données avec des DTD
- Faire des requêtes avec XPATH
- Typer des données avec XML-Schema
- XML-Schema et les espaces de noms
- Transformer des données avec XSLT 1.0
- XSTL 2.0
- XPath 2.0
- Xquery 1.0
- XQUF et XQuery 3.0

Format pédagogique

- Cours hebdomadaire : 1h30
- TD sur machine hebdomadaire : 1h30
- Logiciels utilisés : Editix, BaseX

Évaluation

- Une note sur 20 TD (interrogation écrite)
- Une note sur 20 TP (remises des TP sur P.R.O.F.)
- Une note sur 20 EX (examen de fin de trimestre)

La note finale sur 20 PXML est calculée comme une moyenne pondérée de ces trois notes :

$$PXML = \frac{TP + \sup\left(EX, \frac{2 \times EX + TD}{3} \right)}{4}$$

PXML Master MIAGE M1 2011-2012

Cours 1 : Les principes de base de XML

Les principes de base de XML

- XML permet de décrire des documents **structurés hiérarchiquement**, utilisant des **balises**.
- XML = **eXtensible Markup Language**. eXtensible signifie qu'on définit soit même le langage des balises. On parle alors de dialecte XML :
 - XHTML pour les pages web
 - MathML pour les formules mathématiques
 - DocBook pour la documentation technique
 - NewsML pour les dépêches de presse
 - SVG (Scalable Vector Graphics) graphiques vectoriels 2D
- **Format texte** : facile à modifier, à échanger. Est devenu de fait le **principal format d'échange** entre applications et sur le web.

Document ou donnée ?

- format d'échange, texte, feuilles de style (CSS), transformation (XSLT)
- typage de données (DTD, XML-Schema)
- langages de requêtes (XPath, XQuery)
- langages de mise à jour (XQuery Update Facility)

Document ou donnée ?

- format d'échange, texte, feuilles de style (CSS), transformation (XSLT)
- typage de données (DTD, XML-Schema)
- langages de requêtes (XPath, XQuery)
- langages de mise à jour (XQuery Update Facility)

donnée

- transformation (XSLT)
- typage de données (DTD, XML-Schema)
- langages de requêtes (XPath, XQuery)
- langages de mise à jour (XQuery Update Facility)

Historique

- 1979-1986 : [SGML](#), description de documents techniques
- 1991 : [HTML](#), inventé pour le web
- 1996 : création d'un groupe de travail du [W3C](#) dont les objectifs sont de définir un langage plus facile que SGML et plus générique que HTML i.e. qui permet de définir plusieurs familles de langages de balises.
- 1998 : [XML 1.0](#). Version simplifiée de SGML et plus adaptée au Web (e.g. support natif des différents codages internationaux).

Document bien formé et document valide

- Bien formé = suit les règles syntaxiques de XML
 - Bon [parenthésage](#) des balises ouvrantes et fermantes
 - Un élément [racine](#) contient tous les autres (on parle d'arbre d'éléments)
- Valide = bien formé + conforme à un schéma (DTD , XML-Schema,...)
 - La validité n'est pas requise
 - Définir un schéma permet de manipuler plus facilement des documents, de les traiter par des programmes
 - Il existe plusieurs langages qui permettent de définir des schémas ou types de documents : Document Type Definition (DTD), [XML-schema](#), [Relax-NG](#)

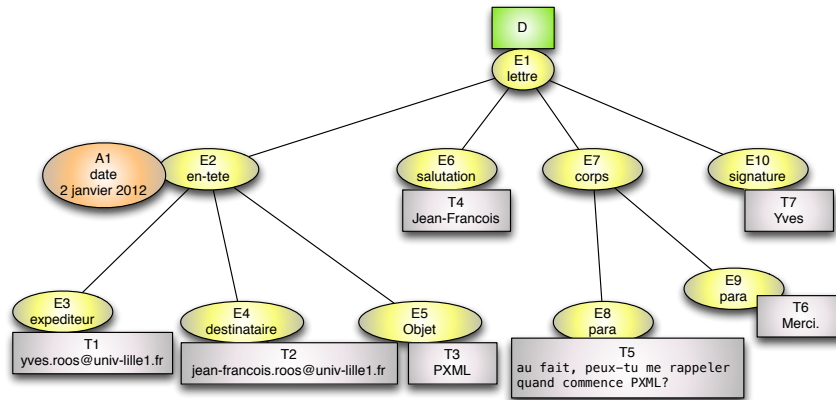
Exemple

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet href="courrier.css"
  type="text/css"?>
<!-- exemple de fichier xml -->
<lettre>
  <en-tete date="2 janvier 2012">
    <expediteur>
      yves.roos@univ-lille1.fr
    </expediteur>
    <destinataire>
      jean-francois.roos@univ-lille1.fr
    </destinataire>
    <objet>PXML</objet>
  </en-tete>
  <salutation>Jean-Francois</salutation>
  <corps>
    <para>
      au fait, peux-tu me rappeler
      quand commence PXML?
    </para>
    <para>Merci.</para>
  </corps>
  <signature>Yves</signature>
</lettre>
```

Structure d'un document XML

- un [prologue](#) (facultatif mais conseillé) déclaration de type de document
`<?xml version="1.0" encoding="ISO-8859-1" ?>`
Par défaut, le codage des caractères est `utf-8`.
- un [arbre d'éléments](#). C'est le contenu du document.
- des [commentaires](#) : `<!-- commentaire ... -->`
- des [instructions de traitement](#) qui peuvent apparaître dans le prologue et dans l'arbre d'éléments. Les instructions de traitement permettent aux documents de contenir des instructions destinées aux applications.
`<?php . . . ?>`
`<?xml-stylesheet href="courrier.css " type="text/css" ?>`

Arbre d'éléments



Les éléments

- **élément** = balise d'ouverture + contenu + balise de clôture
`<destinataire>jean-francois.roos@univ-lille1.fr</destinataire>`
- cas particulier : un **élément vide** contient ces trois choses en une seule balise.
`<eltVide/>`
- **nom** des éléments :
 - sensible à la casse : `<Hello>` ≠ `<hello>`
 - peut comporter des lettres, des chiffres, des caractères `-` `_` `.` `:` et ne peut commencer par les caractères `-` et `.`. Le caractère `:` ne devrait être utilisé que pour séparer les espaces de noms. Les noms commençant par `xml` sont réservés

Les éléments : attributs

- Dans la **balise d'ouverture** d'un élément, on peut définir des **attributs** qui définissent des **propriétés** de l'élément.

```
<rapport langue="FR" date-modif="2012-01-10">
```

- la valeur d'un attribut est une **chaîne**
`date-modif="2012-01-10"` ou `date-modif='2012-01-10'`

- les attributs **ne sont pas ordonnés**
`<rapport langue="FR" date-modif="2012-01-10">`
`<rapport date-modif="2012-01-10" langue="FR">`

- pour un élément donné, **chaque nom d'attribut est unique**
`<rapport langue="FR" langue="EN" >`

Les éléments : contenu

- Un élément peut contenir d'**autres éléments**, des **données**, des **instructions de traitement**,...
- Une donnée est un flot de caractères qui **ne contient pas** les caractères `<`, `>` et `&`
- On peut mettre dans la donnée des **entités** prédéfinies :

Entité	Caractère
<code>&lt;</code>	<code><</code>
<code>&gt;</code>	<code>></code>
<code>&amp;</code>	<code>&</code>
<code>&quot;</code>	<code>"</code>
<code>&apos;</code>	<code>'</code>

Les éléments : contenu

- [section littérale](#) :

```
<exemple>
  <![CDATA[<auteur>Dupond & al.</auteur>
</exemple>
```

Une section CDATA peut contenir n'importe quelle chaîne sauf]]

Mise en forme de XML avec CSS

Une [règle de feuille CSS](#) s'écrit sous la forme générale suivante :

```
sélecteur
{
  propriété-1: valeur ;
  propriété-2: valeur ;
  ...
}
```

Le [sélecteur](#) permet d'indiquer [à quel nœud](#) s'applique la liste de propriétés. Par exemple, pour écrire le contenu de tous les éléments nom en rouge :

```
nom
{
  color: red ;
}
```

Mise en forme de XML avec CSS : sélecteurs

Sélecteur	Éléments ciblés
*	tous les éléments
elt	les éléments elt
elt1 , elt2	les éléments elt1 et elt2
elt1 elt2	les elt2 qui descendent d'un elt1
elt1 > elt2	les elt2 enfants d'un elt1
elt[attr]	les elt possédant un attribut attr
elt[attr="valeur"]	idem avec l'attribut valant valeur
elt[attr1] [attr2]	les elt possédant un attribut attr1 et un attribut attr2
elt1 + elt2	les elt2 frères immédiats d'un elt1
elt:hover	les elt lors du survol de la souris

Mise en forme de XML avec CSS : propriétés

Propriétés	Valeurs
display	inline , block, none
color	aqua, black , blue, gray, green, ...
font-style	italic, normal
text-decoration	underline, overline, linethrough, none
font-variant	normal , small-caps
font-family	serif , sans-serif, monospace, ...
font-weight	normal , bold
...	..., ..., ...

Mise en forme de XML avec CSS : courrier.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet href="courrier.css"
  type="text/css"?>
<!-- exemple de fichier xml -->
<lettre>
  <en-tete date="2 janvier 2012">
    <expediteur>
      yves.roos@univ-lille1.fr
    </expediteur>
    <destinataire>
      jean-francois.roos@univ-lille1.fr
    </destinataire>
    <objet>PXML</objet>
  </en-tete>
  <salutation>Jean-Francois</salutation>
  <corps>
    <para>
      au fait, peux-tu me rappeler
      quand commence PXML?
    </para>
    <para>Merci.</para>
  </corps>
  <signature>Yves</signature>
</lettre>
```

Mise en forme de XML avec CSS : courrier.css

```
/* Type d'affichage */
lettre, en-tete, expediteur, destinataire ,
objet , signature , salutation {display:block}
para {display:inline}

/* Mise en forme */
objet: hover {text-decoration:underline}
destinataire, expediteur {font-weight:bold}
para + para {color:red}

/* Contenu additionnel */
lettre:before {content:"-----"}
en-tete:before {content:"Le : " attr(date)}
expediteur:before {content:"de : " ; font-weight:normal}
destinataire:before {content:"à : " ; font-weight:normal}
objet:before {content:"A propos de : "}
salutation:after {content:", "}
para:before {content:" "}
lettre:after {content:"-----"}
```

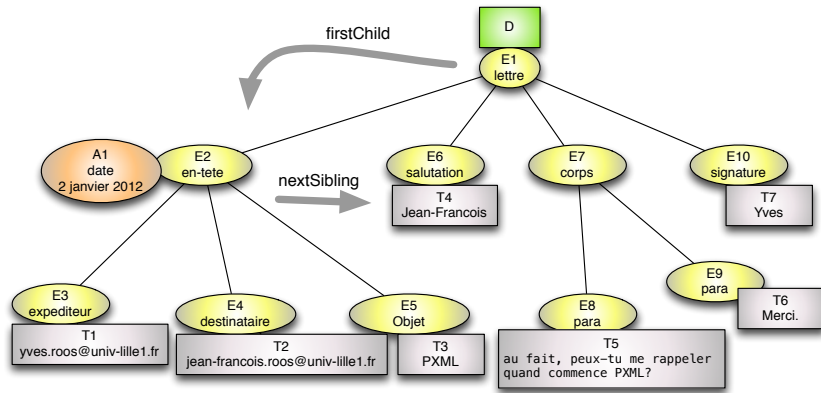
Mise en forme de XML avec CSS : exemple



Lecture de fichier XML : APIs java

- DOM
- SAX

Document Object Model



Document Object Model

Method	Description
<code>appendChild(Node newChild)</code>	Adds the node <code>newChild</code> to the end of the list of children of this node.
<code>cloneNode(boolean deep)</code>	Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.
<code>compareDocumentPosition(Node other)</code>	Compares the reference node, i.e.
<code>getAttributes()</code>	A <code>NamedNodeMap</code> containing the attributes of this node (if it is an <code>Element</code>) or <code>null</code> otherwise.
<code>getBaseURI()</code>	The absolute base URI of this node or <code>null</code> if the implementation wasn't able to obtain an absolute URI.
<code>getChildNodes()</code>	A <code>NodeList</code> that contains all children of this node.
<code>getFeature(String feature, String version)</code>	This method returns a specialized object which implements the specialized APIs of the specified feature and version, as specified in .
<code>getFirstChild()</code>	The first child of this node.
<code>getLastChild()</code>	The last child of this node.
<code>getLocalName()</code>	Returns the local part of the qualified name of this node.
<code>getNamespaceURI()</code>	The namespace URI of this node, or <code>null</code> if it is unspecified (see).
<code>getNextSibling()</code>	The node immediately following this node.
<code>getNodeName()</code>	The name of this node, depending on its type; see the table above.
<code>getNodeType()</code>	A code representing the type of the underlying object, as defined above.
<code>getNodeValue()</code>	The value of this node, depending on its type; see the table above.

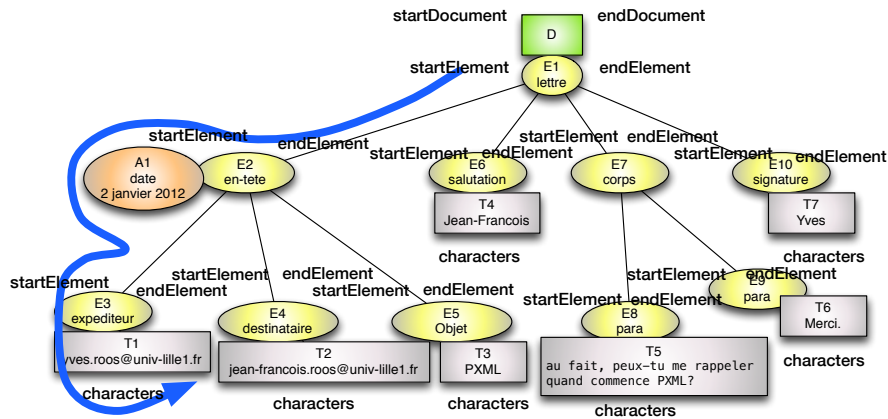
Document Object Model

Method	Description
<code>appendChild(Node newChild)</code>	Adds the node <code>newChild</code> to the end of the list of children of this node.
<code>cloneNode(boolean deep)</code>	Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.
<code>compareDocumentPosition(Node other)</code>	Compares the reference node, i.e.
<code>getAttributes()</code>	A <code>NamedNodeMap</code> containing the attributes of this node (if it is an <code>Element</code>) or <code>null</code> otherwise.
<code>getBaseURI()</code>	The absolute base URI of this node or <code>null</code> if the implementation wasn't able to obtain an absolute URI.
<code>getChildNodes()</code>	A <code>NodeList</code> that contains all children of this node.
<code>getFeature(String feature, String version)</code>	This method returns a specialized object which implements the specialized APIs of the specified feature and version, as specified in .
<code>getFirstChild()</code>	The first child of this node.
<code>getLastChild()</code>	The last child of this node.
<code>getLocalName()</code>	Returns the local part of the qualified name of this node.
<code>getNamespaceURI()</code>	The namespace URI of this node, or <code>null</code> if it is unspecified (see).
<code>getNextSibling()</code>	The node immediately following this node.

Document Object Model

Method	Description
<code>getOwnerDocument()</code>	The document object associated with this node.
<code>getParentNode()</code>	The parent of this node.
<code>getPrefix()</code>	The namespace prefix of this node, or <code>null</code> if it is unspecified.
<code>getPreviousSibling()</code>	The node immediately preceding this node.
<code>getTextContent()</code>	This attribute returns the text content of this node and its descendants.
<code>getUserData(String key)</code>	Retrieves the object associated to a key on a this node.
<code>hasAttributes()</code>	Returns whether this node (if it is an element) has any attributes.
<code>hasChildNodes()</code>	Returns whether this node has any children.
<code>insertBefore(Node newChild, Node refChild)</code>	Inserts the node <code>newChild</code> before the existing child node <code>refChild</code> .
<code>isDefaultNamespace(String namespaceURI)</code>	This method checks if the specified <code>namespaceURI</code> is the default namespace or not.
<code>isEqualNode(Node arg)</code>	Tests whether two nodes are equal.
<code>isSameNode(Node other)</code>	Returns whether this node is the same node as the given one.
<code>isSupported(String feature, String version)</code>	Tests whether the DOM implementation supports the specified feature and version.

Simple Api for XML



Simple Api for XML

[org.w3c.dom.bootstrap](#)
[org.w3c.dom.events](#)
[org.w3c.dom.ls](#)
[org.xml.sax](#)
[org.xml.sax.ext](#)
[org.xml.sax.helpers](#)

[org.xml.sax](#)

[Interfaces](#)
[AttributeList](#)
[Attributes](#)
[ContentHandler](#)
[DocumentHandler](#)
[DTDHandler](#)
[EntityResolver](#)
[ErrorHandler](#)
[Locator](#)
[Parser](#)
[XMLFilter](#)
[XMLReader](#)
[Classes](#)
[HandlerBase](#)
[InputSource](#)
[Exceptions](#)
[SAXException](#)
[SAXNotRecognizedException](#)
[SAXNotSupportedException](#)
[SAXParseException](#)

Method Summary

void	characters (char[] ch, int start, int length)	Receive notification of character data.
void	endDocument ()	Receive notification of the end of a document.
void	endElement (String uri, String localName, String qName)	Receive notification of the end of an element.
void	endPrefixMapping (String prefix)	End the scope of a prefix-URI mapping.
void	ignorableWhitespace (char[] ch, int start, int length)	Receive notification of ignorable whitespace in element content.
void	processingInstruction (String target, String data)	Receive notification of a processing instruction.
void	setDocumentLocator (Locator locator)	Receive an object for locating the origin of SAX document events.
void	skippedEntity (String name)	Receive notification of a skipped entity.
void	startDocument ()	Receive notification of the beginning of a document.
void	startElement (String uri, String localName, String qName, Attributes atts)	Receive notification of the beginning of an element.
void	startPrefixMapping (String prefix, String uri)	Begin the scope of a prefix-URI namespace mapping.

Simple Api for XML

```

package pxml ;

import org.xml.sax.*;
import org.xml.sax.helpers.* ;
import java.io.IOException;

/**
 * @author yves.roos
 *
 * Exemple d'implementation d'un ContentHandler.
 */
public class PXMLHandler extends DefaultHandler {

    /**
     * Evenement envoye au demarrage du parse du flux
     * @throws SAXException en cas de probleme quelc
     * se lancer dans l'analyse du document.
     * @see org.xml.sax.ContentHandler#startDocument
     */
    public void startDocument() throws SAXException
        System.out.println("Debut du document");
    }

```

Simple Api for XML

```

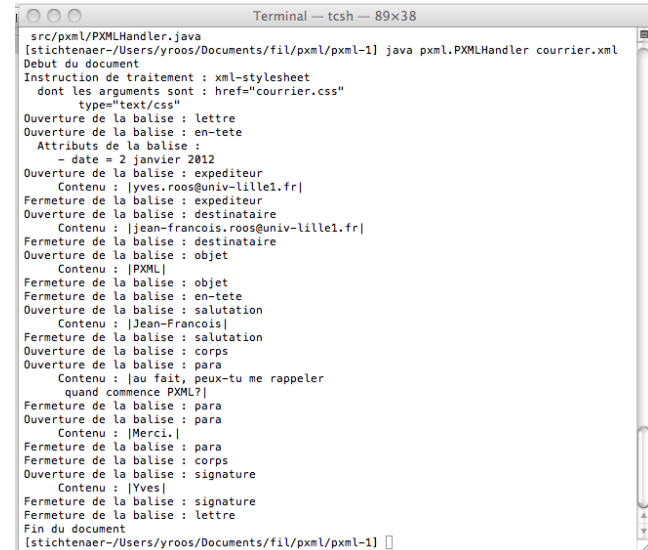
public static void main(String[] args) {
    try {
        XMLReader saxReader = XMLReaderFactory.c
        saxReader.setContentHandler(new PXMLHanc
        saxReader.parse(args[0]);
    } catch (Exception t) {
        t.printStackTrace();
    }
}

```

Simple Api for XML : exemple

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet href="courrier.css"
  type="text/css"?>
<!-- exemple de fichier xml -->
<lettre>
  <en-tete date="2 janvier 2012">
    <expediteur>
      yves.roos@univ-lille1.fr
    </expediteur>
    <destinataire>
      jean-francois.roos@univ-lille1.fr
    </destinataire>
    <objet>PXML</objet>
  </en-tete>
  <salutation>Jean-Francois</salutation>
  <corps>
    <para>
      au fait, peux-tu me rappeler
      quand commence PXML?
    </para>
    <para>Merci.</para>
  </corps>
  <signature>Yves</signature>
</lettre>
```

Simple Api for XML : exemple



```
Terminal — tch — 89x38
src/pxml/PXMLHandler.java
[stichtenaer-/Users/yroos/Documents/fil/pxml/pxml-1] java pxml.PXMLHandler courrier.xml
Debut du document
Instruction de traitement : xml-stylesheet
  dont les arguments sont : href="courrier.css"
  type="text/css"
Ouverture de la balise : lettre
Ouverture de la balise : en-tete
  Attributs de la balise :
    - date = 2 janvier 2012
Ouverture de la balise : expediteur
  Contenu : |yves.roos@univ-lille1.fr|
Fermeture de la balise : expediteur
Ouverture de la balise : destinataire
  Contenu : |jean-francois.roos@univ-lille1.fr|
Fermeture de la balise : destinataire
Ouverture de la balise : objet
  Contenu : |PXML|
Fermeture de la balise : objet
Fermeture de la balise : en-tete
Ouverture de la balise : salutation
  Contenu : |Jean-Francois|
Fermeture de la balise : salutation
Ouverture de la balise : corps
  Contenu : |au fait, peux-tu me rappeler
  quand commence PXML?|
Fermeture de la balise : para
Ouverture de la balise : para
  Contenu : |Merci.|
Fermeture de la balise : para
Fermeture de la balise : corps
Ouverture de la balise : signature
  Contenu : |Yves|
Fermeture de la balise : signature
Fermeture de la balise : lettre
Fin du document
[stichtenaer-/Users/yroos/Documents/fil/pxml/pxml-1] □
```