PXML

Yves Roos

Bureau 220 – M3.ext
yves.roos@lifl.fr

Master MIAGE M1 2010/2011

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.
- Document ou donnée ?
  1. format d'échange, texte, feuilles de style (CSS), transformation (XSLT)
  2. schéma, typage de données, langages de requête (XPath, XQuery)

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.
- Document ou donnée ?
  1. format d'échange, texte, feuilles de style (CSS), transformation (XSLT)
  2. schéma, typage de données, langages de requête (XPath, XQuery)

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.
- Document ou donnée ?
  1. format d'échange, texte, feuilles de style (CSS), transformation (XSLT)
  2. schéma, typage de données, langages de requête (XPath, XQuery)

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.
- Document ou donnée ?
  1. format d'échange, texte, feuilles de style (CSS), transformation (XSLT)
  2. schéma, typage de données, langages de requête (XPath, XQuery)

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.
- Document ou donnée ?
  1. format d'échange, texte, feuilles de style (CSS), transformation (XSLT)
  2. schéma, typage de données, langages de requête (XPath, XQuery)

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.
- Document ou donnée ?
  1. format d'échange, texte, feuilles de style (CSS), transformation (XSLT)
  2. schéma, typage de données, langages de requête (XPath, XQuery)

Plan du Cours

Plan du Cours

Plan du Cours

Plan du Cours

Plan du Cours

Plan du Cours

Plan du Cours

1 Les principes de bases de XML

- Historique de XML
- Document bien formé
- Structure d'un document XML
- Les éléments
- Mise en forme de XML avec CSS
- Java et XML

Historique

- Langages de balises :
  - SGML, description de documents techniques, normalisé en 1986 (début en 1979).
  - HTML, inventé pour le web 1991
- 1996 : création d'un groupe de travail du W3C dont les objectifs sont de définir un langage
  1. plus facile que SGML
  2. plus général 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).

- 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 les 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

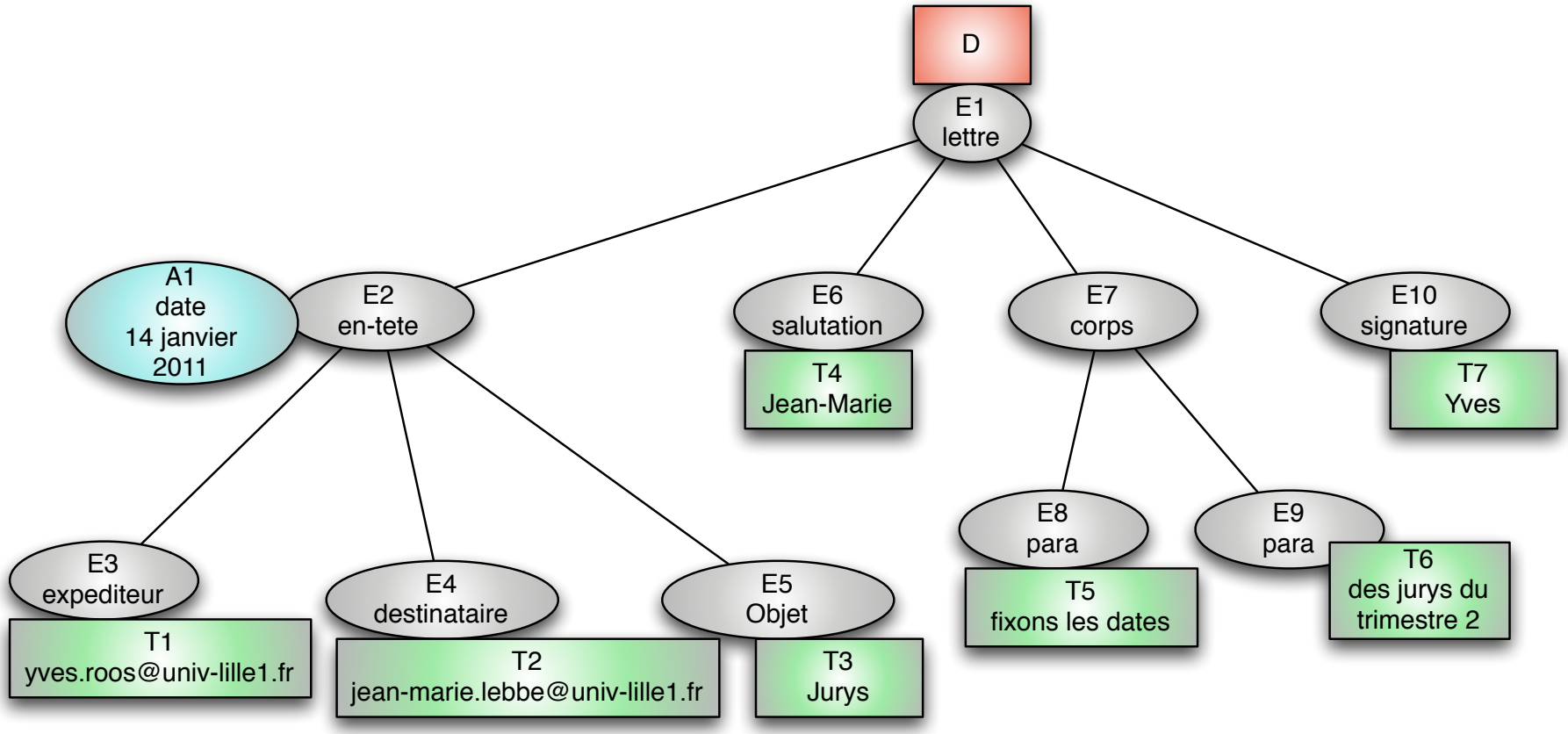
Exemple

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="courrier.css" type="text/css"?>
<lettre>
  <en-tete date="14 janvier 2011">
    <expediteur>yves.roos@univ-lille1.fr</expediteur>
    <destinataire>jean-marie.lebbe@univ-lille1.fr</
        destinataire>
    <objet>Jurys</objet>
  </en-tete>
  <salutation>Jean-Marie</salutation>
  <corps>
    <para>fixons les dates</para>
    <para>des Jurys du trimestre 2</para>
  </corps>
  <signature>Yves</signature>
</lettre>
```

- 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.
- d'un arbre d'éléments. C'est le contenu du document.
- de commentaires, un commentaire se note :
  `<!-- commentaire ... -->`
- d'instructions de traitement qui peuvent apparaître dans le prologue et dans l'arbre d'éléments. et ne peut pas contenir la chaîne `--`. 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



- elément = balise d'ouverture + contenu + balise de clôture.
  `<nom>Yves Roos</nom>`
- cas particulier, un élément vide comprends ces 3 choses en une seule balise : `<eltvide/>`
- Nom d'élément :
  - la casse est importante,
  - il peut comporter des lettres, des chiffres, des caractères `-` `_` `:` Le caractère deux-points (:) ne devrait être utilisé que pour séparer des espaces de noms. Les noms commençant par `xml` sont réservés.
- 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="2006-11-30">`
  - La valeur d'un attribut est une chaîne ('...' ou "...").
  - Les attributs ne sont pas ordonnés.
  - Pour un élément donné, chaque nom d'attribut est unique.

Contenu d'un élément

- 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 `<` `>` `&`
- On peut mettre dans la données des entités prédéfinies :

Entité	Caractère
`<`	<
`>`	>
`&`	&
`&quot;`	"
`&aquot;`	'

Contenu d'un élément

- section littérale :

```
<ex>
  <![CDATA[<auteur>Dupond \&amp; al. </auteur>]]>
</ex>
```

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

Mise en forme de XML avec CSS

Mise en forme de XML avec CSS

Une règle de feuille CSS s'écrit sous la forme générale suivante :

```
sélecteur
  {
    propriété1: valeur;
    propriété2: 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;
  }
```

CSS

Sélécteur	Eléments ciblés
`*`	
`elt`	
`elt1 , elt2`	
`elt1 elt2`	
`elt1 > elt2`	
`elt[attr]`	
`elt[attr="valeur"]`	
`elt[attr1][attr2]`	
`elt1 + elt2`	
`elt:hover`	

CSS

Sélécteur	Eléments 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 `attr1` et un `attr2`
`elt1 + elt2`	Les `elt2` frères immédiats d'un `elt1`
`elt:hover`	Les `elt` lors du survol de la souris

CSS

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

Le fichier courrier.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="courrier.css" type="text/css"?>
<lettre>
  <en-tete date="14 janvier 2011">
    <expediteur>yves.roos@univ-lille1.fr</expediteur>
    <destinataire>jean-marie.lebbe@univ-lille1.fr</
        destinataire>
    <objet>Jurys</objet>
  </en-tete>
  <salutation>Jean-Marie</salutation>
  <corps>
    <para>fixons les dates</para>
    <para>des jurys du trimestre 2</para>
  </corps>
  <signature>Yves</signature>
</lettre>
```

Le fichier courrier.css

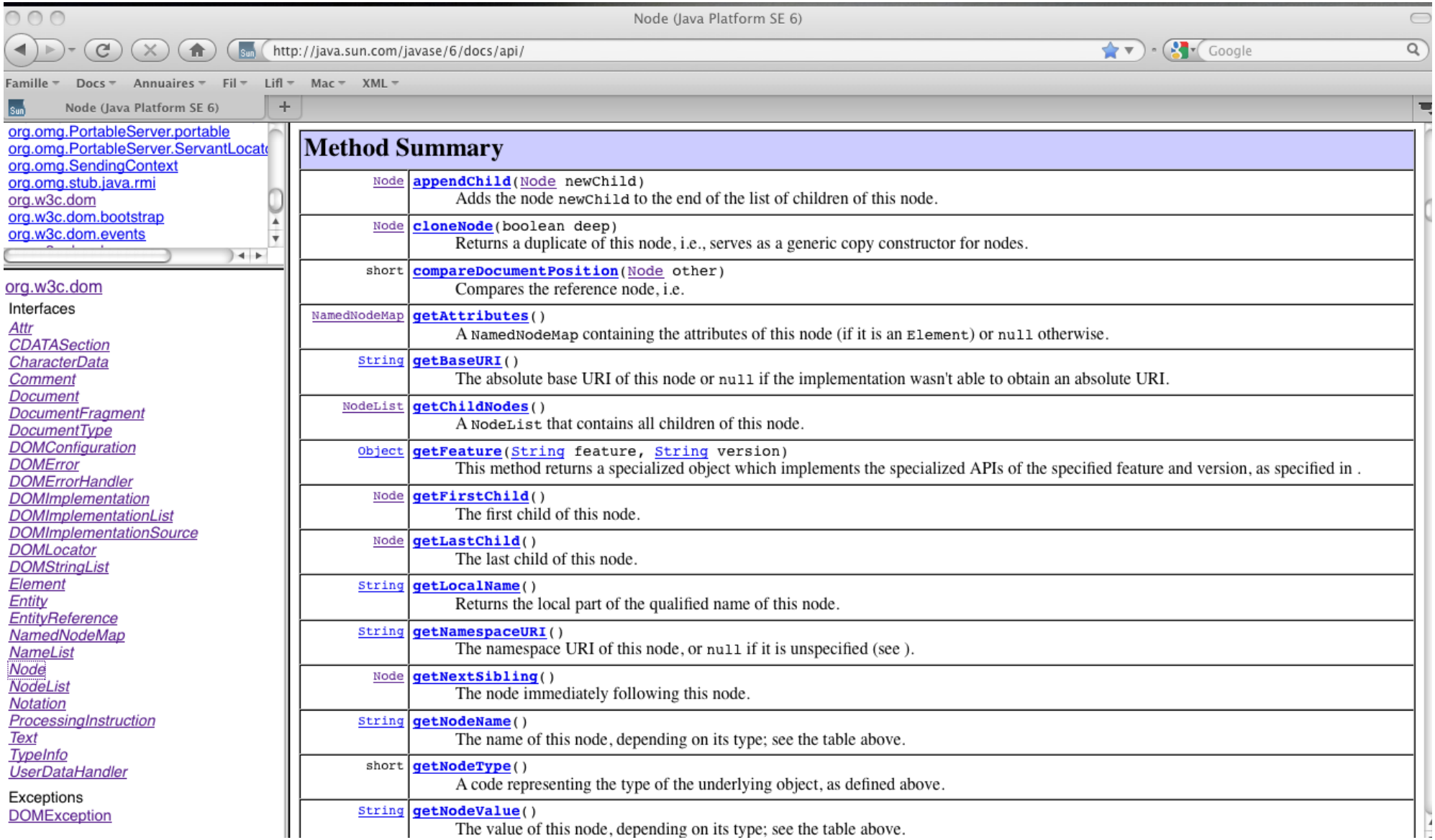
```
/* Type d'affichage */
lettre, en-tete, expediteur,
destinataire , objet , signature {display:block}
salutation , para {display:inline}
/* Mise en forme */
objet:hover {text-decoration:underline}
destinataire, expediteur {font-weight:bold}

/* 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:"----------"}
```

Résultat de la mise en forme

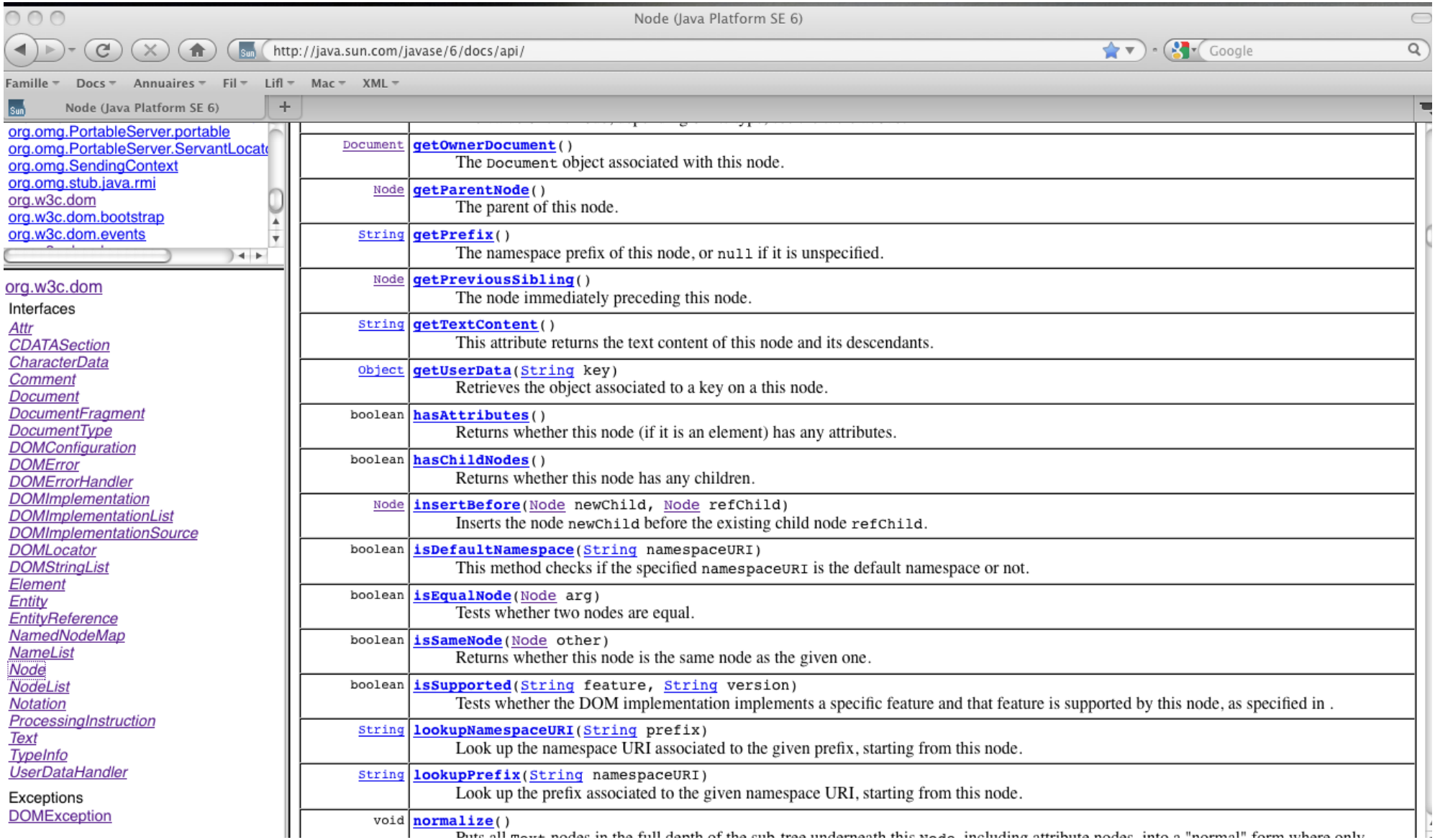
Java et XML

1. DOM
2. SAX

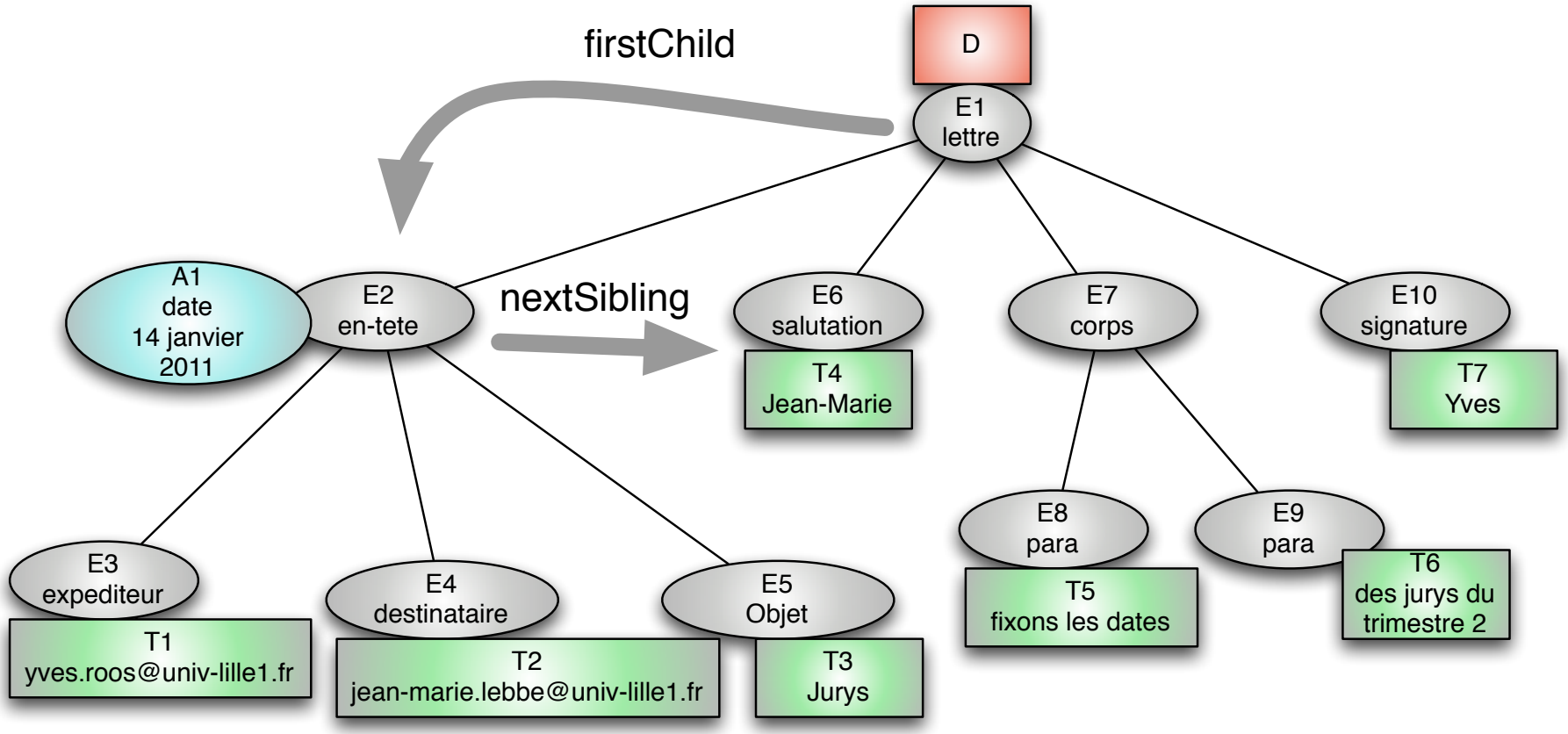
DOM


Node (Java Platform SE 6)
http://java.sun.com/javase/6/docs/api/
Famille Docs Annuaires Fil Lifl Mac XML
Node (Java Platform SE 6)
org.omg.PortableServer.portable
org.omg.PortableServer.ServantLocat
org.omg.SendingContext
org.omg.stub.java.rmi
org.w3c.dom
org.w3c.dom.bootstrap
org.w3c.dom.events
org.w3c.dom
Interfaces
Attr
CDATASection
CharacterData
Comment
Document
DocumentFragment
DocumentType
DOMConfiguration
DOMError
DOMErrorHandler
DOMImplementation
DOMImplementationList
DOMImplementationSource
DOMLocator
DOMStringList
Element
Entity
EntityReference
NamedNodeMap
NameList
Node
NodeList
Notation
ProcessingInstruction
Text
TypeInfo
UserDataHandler
Exceptions
DOMException
Method Summary
Node appendChild(Node newChild)
Adds the node newChild to the end of the list of children of this node.
Node cloneNode(boolean deep)
Returns a duplicate of this node, i.e., serves as a generic copy constructor for nodes.
short compareDocumentPosition(Node other)
Compares the reference node, i.e.
NamedNodeMap getAttributes()
A NamedNodeMap containing the attributes of this node (if it is an Element) or null otherwise.
String getBaseURI()
The absolute base URI of this node or null if the implementation wasn't able to obtain an absolute URI.
NodeList getChildNodes()
A NodeList that contains all children of this node.
Object getFeature(String feature, String version)
This method returns a specialized object which implements the specialized APIs of the specified feature and version, as specified in .
Node getFirstChild()
The first child of this node.
Node getLastChild()
The last child of this node.
String getLocalName()
Returns the local part of the qualified name of this node.
String getNamespaceURI()
The namespace URI of this node, or null if it is unspecified (see).
Node getNextSibling()
The node immediately following this node.
String getNodeName()
The name of this node, depending on its type; see the table above.
short getNodeType()
A code representing the type of the underlying object, as defined above.
String getNodeValue()
The value of this node, depending on its type; see the table above.

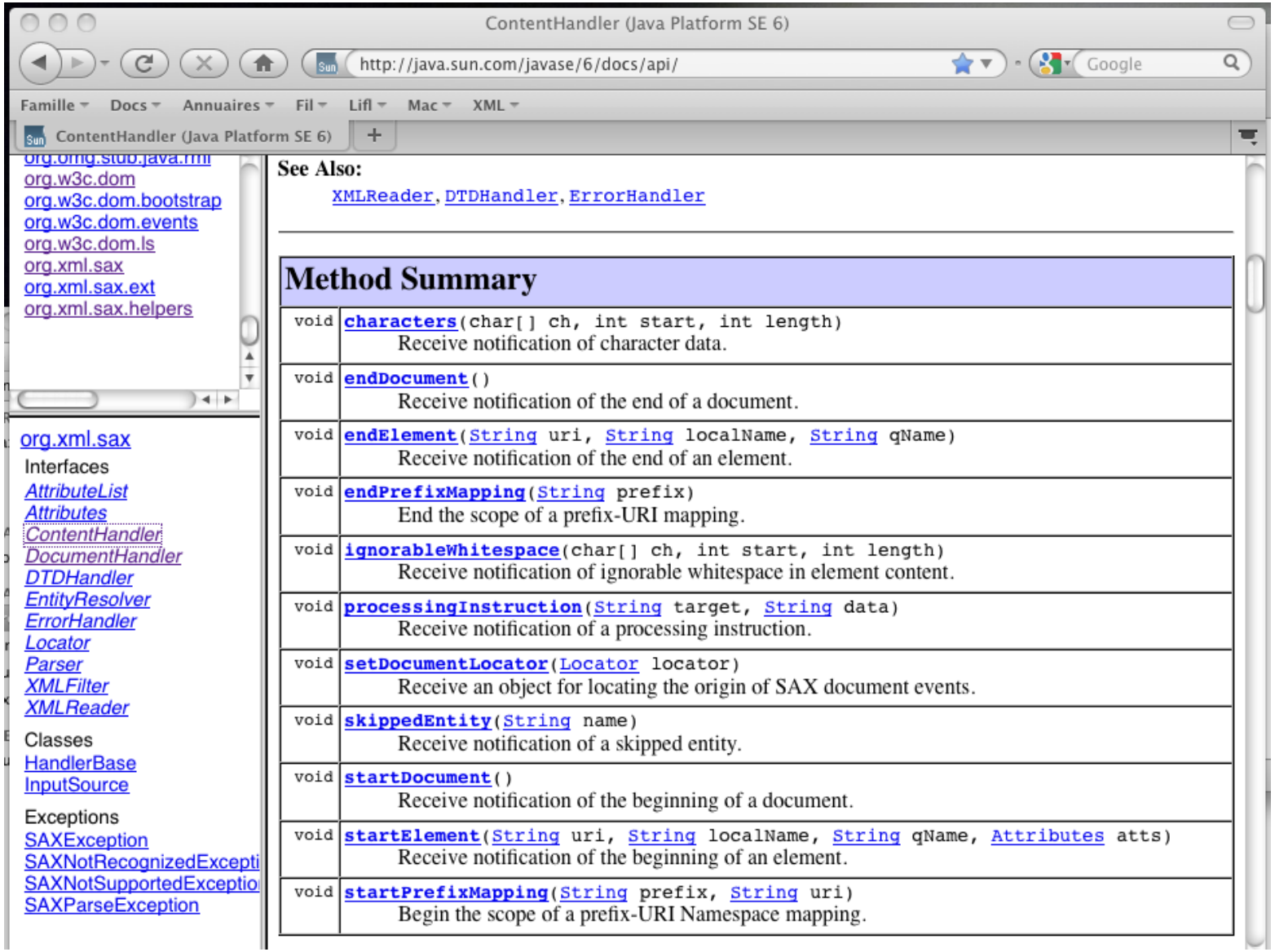
DOM


Node (Java Platform SE 6)
http://java.sun.com/javase/6/docs/api/
Famille Docs Annuaires Fil Lifl Mac XML
Node (Java Platform SE 6)
org.omg.PortableServer.portable
org.omg.PortableServer.ServantLocat
org.omg.SendingContext
org.omg.stub.java.rmi
org.w3c.dom
org.w3c.dom.bootstrap
org.w3c.dom.events
org.w3c.dom
Interfaces
Attr
CDATASection
CharacterData
Comment
Document
DocumentFragment
DocumentType
DOMConfiguration
DOMError
DOMErrorHandler
DOMImplementation
DOMImplementationList
DOMImplementationSource
DOMLocator
DOMStringList
Element
Entity
EntityReference
NamedNodeMap
NameList
Node
NodeList
Notation
ProcessingInstruction
Text
TypeInfo
UserDataHandler
Exceptions
DOMException
Document getOwnerDocument()
The Document object associated with this node.
Node getParentNode()
The parent of this node.
String getPrefix()
The namespace prefix of this node, or null if it is unspecified.
Node getPreviousSibling()
The node immediately preceding this node.
String getTextContent()
This attribute returns the text content of this node and its descendants.
Object getUserData(String key)
Retrieves the object associated to a key on a this node.
boolean hasAttributes()
Returns whether this node (if it is an element) has any attributes.
boolean hasChildNodes()
Returns whether this node has any children.
Node insertBefore(Node newChild, Node refChild)
Inserts the node newChild before the existing child node refChild.
boolean isDefaultNamespace(String namespaceURI)
This method checks if the specified namespaceURI is the default namespace or not.
boolean isEqualNode(Node arg)
Tests whether two nodes are equal.
boolean isSameNode(Node other)
Returns whether this node is the same node as the given one.
boolean isSupported(String feature, String version)
Tests whether the DOM implementation implements a specific feature and that feature is supported by this node, as specified in .
String lookupNamespaceURI(String prefix)
Look up the namespace URI associated to the given prefix, starting from this node.
String lookupPrefix(String namespaceURI)
Look up the prefix associated to the given namespace URI, starting from this node.
void normalize()

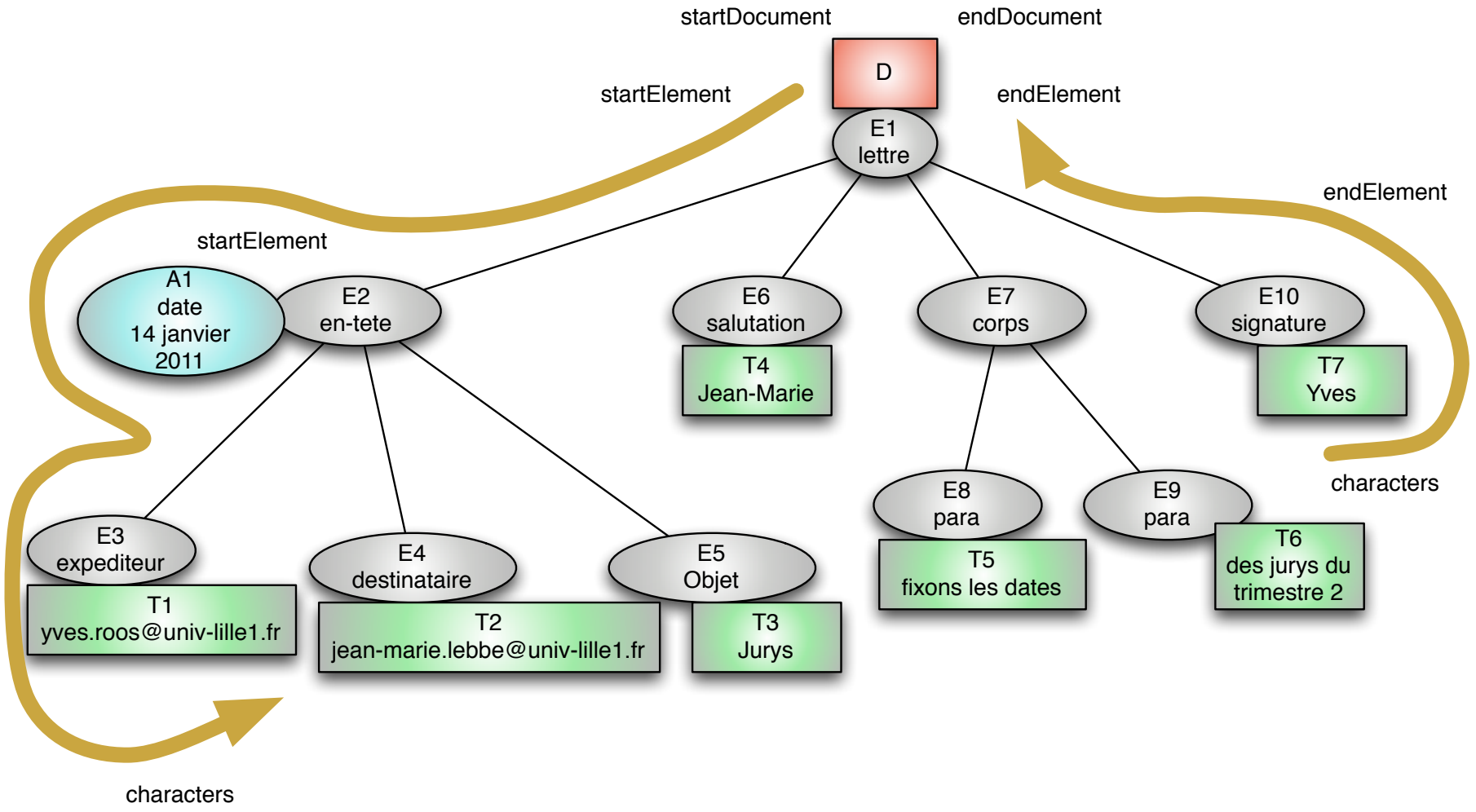
DOM


firstChild
D
E1
lettre
A1
date
14 janvier
2011
E2
en-tete
nextSibling
E6
salutation
T4
Jean-Marie
E7
corps
E10
signature
T7
Yves
E3
expediteur
T1
yves.roos@univ-lille1.fr
E4
destinataire
T2
jean-marie.lebbe@univ-lille1.fr
E5
Objet
T3
Jurys
E8
para
T5
fixons les dates
E9
para
T6
des jurys du
trimestre 2

SAX


ContentHandler (Java Platform SE 6)
http://java.sun.com/javase/6/docs/api/
Famille Docs Annuaires Fil Lifl Mac XML
ContentHandler (Java Platform SE 6)
org.omg.stub.java.rmi
org.w3c.dom
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
SAXNotRecognizedExcepti
SAXNotSupportedExceptio
SAXParseException
See Also:
XMLReader, DTDHandler, ErrorHandler
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.

SAX


startDocument
endDocument
D
startElement
endElement
E1
lettre
endElement
startElement
A1
date
14 janvier
2011
E2
en-tete
E6
salutation
T4
Jean-Marie
E7
corps
E10
signature
T7
Yves
characters
E8
para
E9
para
T5
fixons les dates
T6
des jurys du
trimestre 2
E3
expediteur
T1
yves.roos@univ-lille1.fr
E4
destinataire
T2
jean-marie.lebbe@univ-lille1.fr
E5
Objet
T3
Jurys
characters

SAX

```
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 {
```

SAX

```
/**
* Evenement envoye au demarrage du parse du flux xml.
* @throws SAXException en cas de probleme
*/
        public void startDocument() throws
            SAXException {
          System.out.println("Debut du document");
        }
/**
* Evenement envoye a la fin de l'analyse du flux xml.
*/
 public void endDocument() throws SAXException {
        System.out.println("Fin du document" );
 }
```

SAX

```
/**
* balise xml ouvrante.
* @param nameSpaceURI l'url de l'espace de nommage.
* @param localName le nom local de la balise.
* @param rawName nom de la balise en version 1.0
*/
public void startElement(String nameSpaceURI, String
    localName,
  String rawName, Attributes attributs) throws
      SAXException {
  System.out.println("Ouverture de la balise : " +
      localName) ;
   if (attributs.getLength() != 0)
     System.out.println("Attributs de la balise: ");
   for (int index = 0; index < attributs.getLength();
       index++) {
     System.out.println("      - " +
       attributs.getLocalName(index) + " = "
       + attributs.getValue(index));
```

SAX

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

Le fichier courrier.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="courrier.css" type="text/css"?>
<lettre>
  <en-tete date="14 janvier 2011">
    <expediteur>yves.roos@univ-lille1.fr</expediteur>
    <destinataire>jean-marie.lebbe@univ-lille1.fr</
        destinataire>
    <objet>Jurys</objet>
  </en-tete>
  <salutation>Jean-Marie</salutation>
  <corps>
    <para>fixons les dates</para>
    <para>des jurys du trimestre 2</para>
  </corps>
  <signature>Yves</signature>
</lettre>
```

SAX

```
[stichtenaer-/Users/yroos/Documents/fil/pxml/tp1] 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 = 14 janvier 2011
Ouverture de la balise : expediteur
      Contenu : |yves.roos@univ-lille1.fr|
Fermeture de la balise : expediteur
Ouverture de la balise : destinataire
      Contenu : |jean-marie.lebbe@univ-lille1.fr|
Fermeture de la balise : destinataire
Ouverture de la balise : objet
```

SAX

```
     Contenu : |Jurys|
Fermeture de la balise : objet
Fermeture de la balise : en-tete
Ouverture de la balise : salutation
     Contenu : |Jean-Marie|
Fermeture de la balise : salutation
Ouverture de la balise : corps
Ouverture de la balise : para
     Contenu : |fixons les dates|
Fermeture de la balise : para
Ouverture de la balise : para
     Contenu : |des jurys du trimestre 2|
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
```