XML

SÉANCE 4 : XML ET JAVA

2ÈME INFO

CHRISTOPHE DAMAS

PARSEUR XML

- **Permet de lire ou manipuler les données d'un document XML dans un langage de programmation comme Java.**
- **2 types de parseur XML**
  - Approche hiérarchique
    - Construction d'un arbre représentant le document
    - L'API la plus utilisée: DOM (Document Object Model)
  - Approche événementielle
    - Lecture séquentielle du document et génération d'événements (comme le début d'un élément, la fin d'un élément)
    - L'API la plus utilisée: SAX (Simple API for XML)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<carnet>
        <personne titre="Mme">
                <nom>Leconte</nom>
                <prenom>Emmeline</prenom>
                <contact>
                        <tel>027644688</tel>
                        <bureau>028</bureau>
                        <email>emmeline.leconte&#64;ipl.be</email>
                </contact>
        </personne>
        <personne titre="Mr">
                <nom>Debacker</nom>
                <prenom>Michel</prenom>
                <contact>
                        <tel>027644653</tel>
                        <bureau>045</bureau>
                        <email>michel.debacker&#64;ipl.be</email>
                </contact>
        </personne>
</carnet>
```

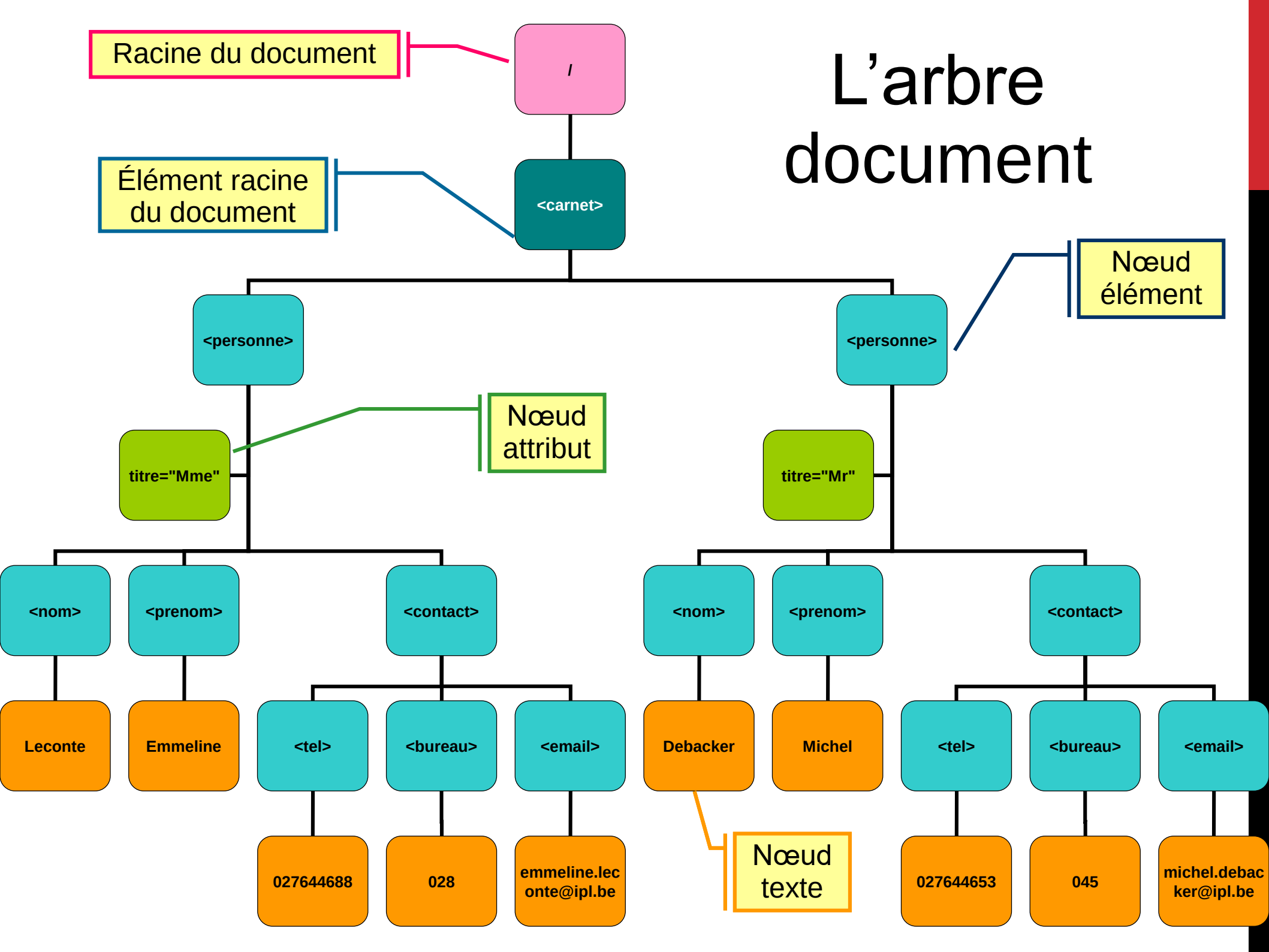
PROGRAMME SOUHAITÉ

```
Mme Leconte
telephone: 027644688
--------------------
Mr Debacker
telephone: 027644653
--------------------
```

DOM

- **Représente un document XML sous la forme d'arbre**
- **Permet la manipulation de cette représentation: parcours, recherche et mise à jour**
- **DOM est indépendant de tout langage de programmation.**
  - spécification du W3C
- **En Java, utiliser les classes du package `org.w3c.dom`**

L'arbre document



```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<carnet>
        <personne titre="Mme">
                <nom>Leconte</nom>
                <prenom>Emmeline</prenom>
                <contact>
                        <tel>027644688</tel>
                        <bureau>028</bureau>
                        <email>emmeline.leconte&#64;ipl.be</email>
                </contact>
        </personne>
        <personne titre="Mr">
                <nom>Debacker</nom>
                <prenom>Michel</prenom>
                <contact>
                        <tel>027644653</tel>
                        <bureau>045</bureau>
                        <email>michel.debacker&#64;ipl.be</email>
                </contact>
        </personne>
</carnet>
```

DOM EN JAVA

- **Le document XML = arbre dont tous les nœuds héritent de `Node`**
  - Interface `Element`
    - hérite de `Node`
    - définit des méthodes pour manipuler un élément et ses attributs.
  - Interface Document
    - hérite de `Node`
    - définit les caractéristiques pour un objet qui sera la racine de l'arbre DOM.
  - …
- **Plus de détails dans la javadoc**

OBTENIR UN ARBRE DOM

```
import java.io.File;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class DOMParseExample {
        public static void main(String[] args) {
                try {    File xmlFile = new File("carnet.xml");
                        DocumentBuilderFactory dbFactory=
                                DocumentBuilderFactory.newInstance();
                        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
                        Document doc = dBuilder.parse(xmlFile) ;
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

PARCOURIR UN ARBRE DOM

```
NodeList personnes = doc.getElementsByTagName("personne");
for (int i = 0; i < personnes.getLength(); i++) {
    Node nPersonne = personnes.item(i);
    Element ePersonne = (Element) nPersonne;

    System.out.println(ePersonne.getAttribute("titre") + " "
    + ePersonne.getElementsByTagName("nom").item(0).getTextContent());
    System.out.println ("telephone: "+ ePersonne
        .getElementsByTagName("tel").item(0).getTextContent());
    System.out.println("----------------");
}
```

OUTPUT DU PROGRAMME

```
Mme Leconte
telephone: 027644688
--------------------
Mr Debacker
telephone: 027644653
--------------------
```

CREER UN ARBRE DOM

```
import java.io.File;
import javax.xml.*;
import org.w3c.dom.*;


public class DomCreateExample {
     public static void main(String argv[]) {
          try {
               DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
               DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
               Document doc = dBuilder.newDocument();
               Element rootElement = doc.createElement("carnet");
               doc.appendChild(rootElement);

               //  creation element personne
              Element personne = doc.createElement("personne");
              rootElement.appendChild(personne);
```

CREER UN ARBRE DOM (2)

```
        // ajout de l'attribut titre
        Attr attr = doc.createAttribute("titre");
        attr.setValue("Mme");
        personne.setAttributeNode(attr);

        // ajout element nom
        Element nom = doc.createElement("nom");
        nom.appendChild(doc.createTextNode("Leconte"));
        personne.appendChild(nom);
         // ajout element prenom et contact
        …
        // enregistrer dans un fichier
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File("output.xml"));
        transformer.transform(source, result);
        } catch (Exception e) {e.printStackTrace();}
    }
}
```

AVANTAGES ET INCONVENIENTS DE DOM

- **Totalité de l'arborescence XML chargée en mémoire!**
  - Avantage:
    - Facile à utiliser: on peut naviguer vers n'importe quel nœud (parent, enfant).
  - Inconvénient:
    - Coute très cher en mémoire.

- **Utiliser DOM quand**
  - Besoin de modification/création d'une structure xml
  - Peu de données

SAX

- **SAX lit le document XML de manière linéaire du début jusqu'à la fin**
- **SAX ne garde rien en mémoire (pas d'arbre)**
- **SAX génère des événements**
  - L'utilisateur peut ajouter des Handler pouvant réagir à certains événements qui l'intéressent
  - Un Handler va par exemple être appelé quand un élément commence ou se termine

- **En Java, il faut utiliser les classes du package org.xml.sax.**

DEFINITION DES HANDLERS EN JAVA

- **Tout Handler devra implémenter l'interface `ContentHandler`.**
  - `ContentHandler` définit des méthodes que SAX utilisera pour avertir le programme qu'un composant du document XML a été lu.
    - `void startDocument()`
    - `void endDocument()`
    - `void startElement(String uri, String localName, String qName, Attributes atts)`
    - `void endElement(String uri, String localName,String qName)`
    - `void characters(char[] ch, int start, int length)`

- **`DefaultHandler` implémente `ContentHandler`**
  - fournit une implémentation par défaut pour réagir à tous les événements.

```
<?xml version="1.0" encoding="UTF-8"?>
<carnet>
        <personne titre="Mme">
                <nom>Leconte</nom>
                <prenom>Emmeline</prenom>
                <contact>
                        <tel>027644688</tel>
                        <bureau>028</bureau>
                        <email>emmeline.leconte&#64;ipl.be</email>
                </contact>
        </personne>
        <personne titre="Mr">
                <nom>Debacker</nom>
                <prenom>Michel</prenom>
                <contact>
                        <tel>027644653</tel>
                        <bureau>045</bureau>
                        <email>michel.debacker&#64;ipl.be</email>
                </contact>
        </personne>
</carnet>
```

DEFINITION DES HANDLERS

```
import org.xml.sax.* ;

public class SAXHandler extends DefaultHandler {

        String titre; boolean bNom = false; boolean bTel = false;

        @Override

        public void startElement(String uri, String localName, String qName, Attributes attributes)
throws SAXException {

                if (qName.equalsIgnoreCase("personne")) {titre = attributes.getValue("titre");

                } else if (qName.equalsIgnoreCase("nom")) {bNom = true;

                } else if (qName.equalsIgnoreCase("tel")) {bTel = true;}

        }

        @Override

        public void endElement(String uri, String localName, String qName) throws SAXException {

                if (qName.equalsIgnoreCase("personne")) {System.out.println("-------------------");}

        }

        @Override

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

                if (bNom) {

                        System.out.println(titre + " " + new String(ch, start, length));

                        bNom = false;

                } else if (bTel) {

                        System.out.println("telephone: " + new String(ch, start, length));

                        bTel = false;}

        }

}
```

PROGRAMME PRINCIPAL

```
import java.io.File;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

public class SAXParseExample {
   public static void main(String[] args){

      try {
         File inputFile = new File("carnet.xml");
         SAXParserFactory factory = SAXParserFactory.newInstance();
         SAXParser saxParser = factory.newSAXParser();
         SAXHandler userhandler = new SAXHandler();
         saxParser.parse(inputFile, userhandler);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

AVANTAGES ET INCONVENIENTS DE SAX

- **L'utilisation de SAX est conseillée quand**
  - on peut lire le document XML de manière linéaire de haut en bas.
  - on veut lire un très gros document XML
  - le problème à résoudre implique seulement une petite partie du document XML.
- **Désavantage**
  - non-présence d'une structure d'arbre qui permettrait d'obtenir facilement les éléments parents et enfants.