# XML : données d'une Application

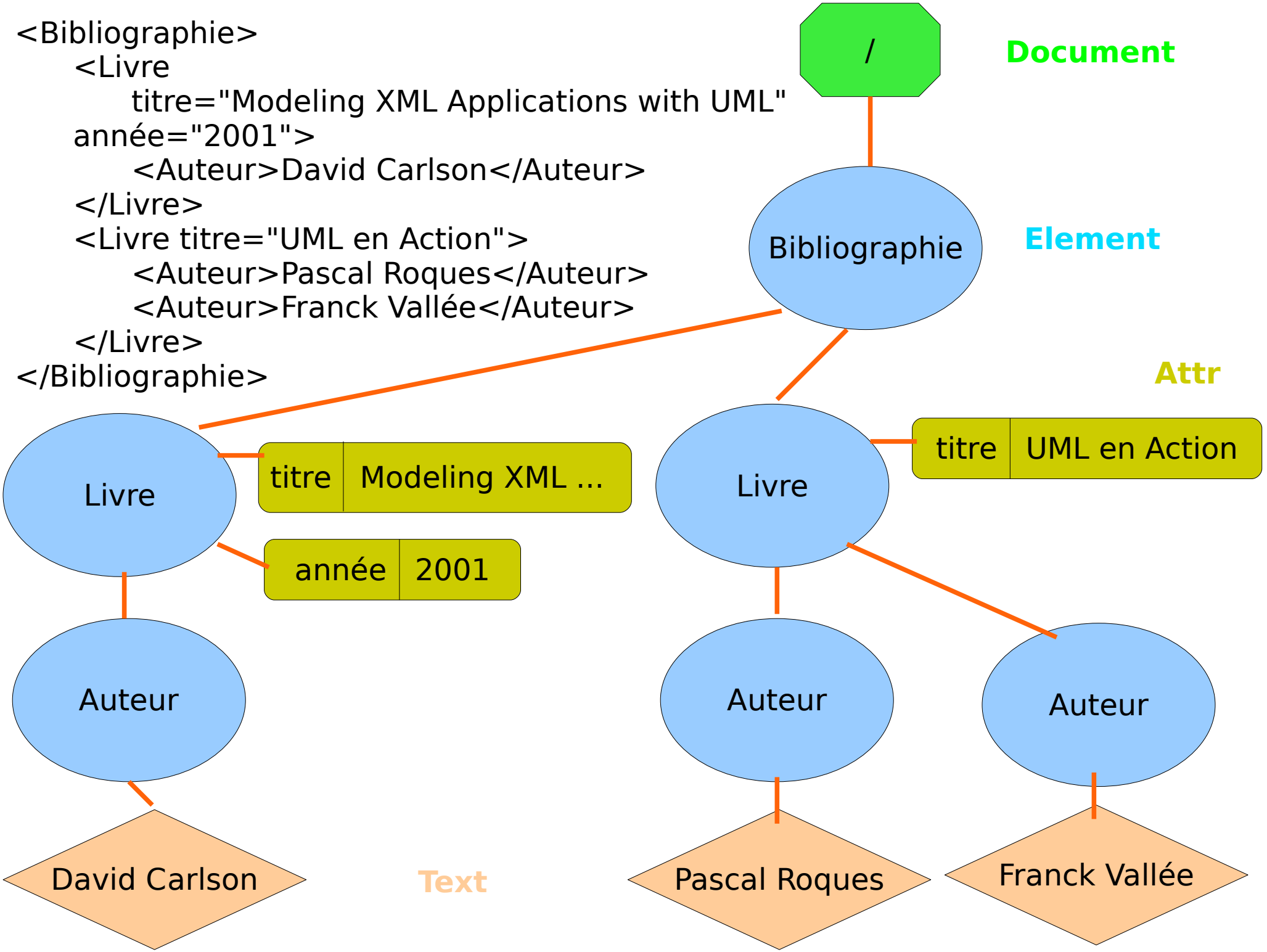
- Une application nécessite souvent de manipuler des données XML pour profiter des avantages de ce format.
- Exemples : fichier de configuration au démarrage, données à traiter
- Ces besoins ont amené la création et la normalisation d'outils pour pouvoir manipuler des données XML dans une application classique.
- Parmi ceux-ci, les deux plus connus sont :
  - **SAX** : permet de parcourir une fois un fichier XML et de faire un **traitement** approprié sur chaque élément.
  - **DOM** : permet de charger un fichier XML en **mémoire** sous la forme d'un arbre.
- DOM permet donc de manipuler à plusieurs reprises une donnée XML pendant la durée de l'application aussi bien en lecture qu'en modification

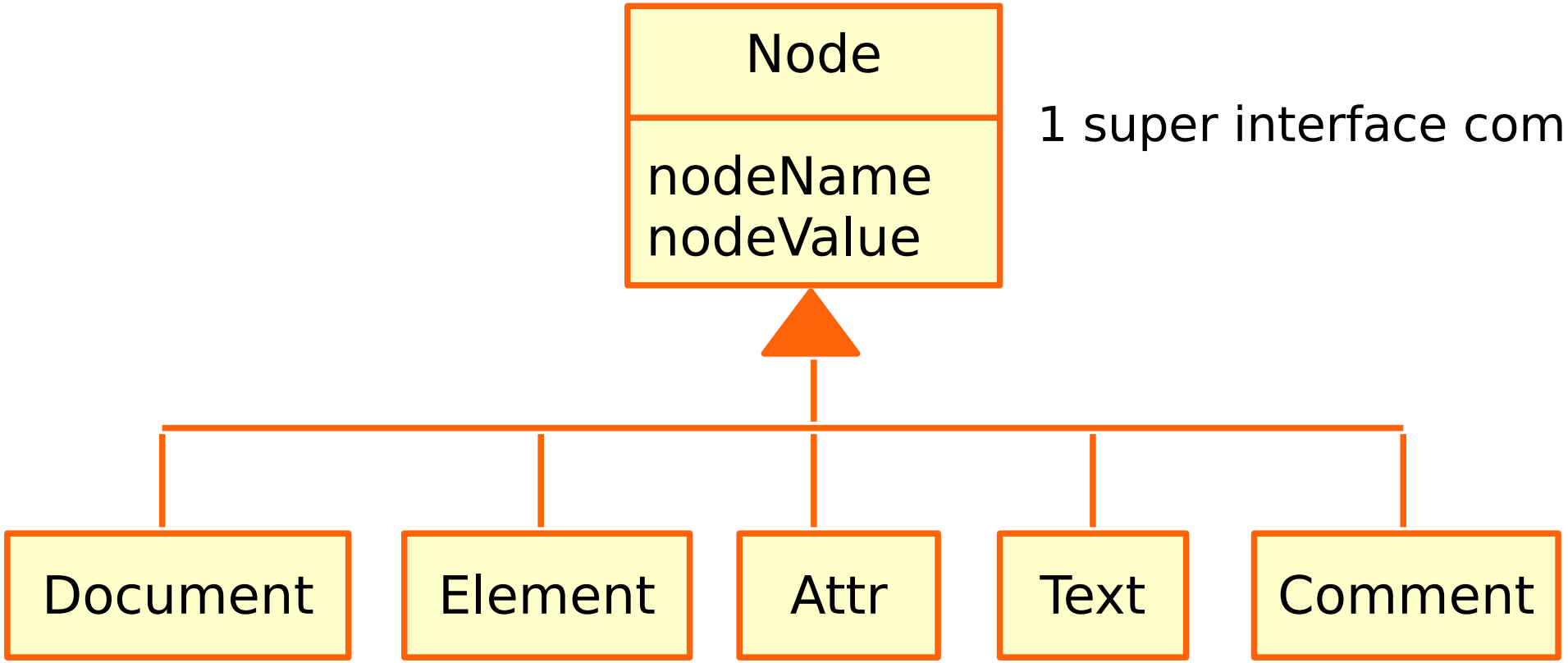
• DOM est une normalisation du consortium W3 : http://www.w3.org
• DOM est implémenté dans de nombreux langages informatiques dont Java et Javascript.
• Dans le langage Java, le package org.w3c.dom est inclus dans le JDK (il existe également une autre version opensource JDom)
• DOM est un ensemble d'interfaces qui permettent de :
  • représenter sous la forme d'un **arbre** un fichier XML
  • **naviguer** dans l'arbre
  • faire une **recherche** dans un arbre
  • **modifier** un arbre

package org.w3c.dom

| | | |
|---|---|---|
| interface Document | <-> | arbre XML |
| interface Element | <-> | élément XML |
| interface Attr | <-> | attribut d'un élement XML |
| interface Text | <-> | texte d'un élément XML |
| interface Comment | <-> | commentaire XML |

```
<Bibliographie>
    <Livre
        titre="Modeling XML Applications with UML"
    année="2001">
        <Auteur>David Carlson</Auteur>
    </Livre>
    <Livre titre="UML en Action">
        <Auteur>Pascal Roques</Auteur>
        <Auteur>Franck Vallée</Auteur>
    </Livre>
</Bibliographie>
```

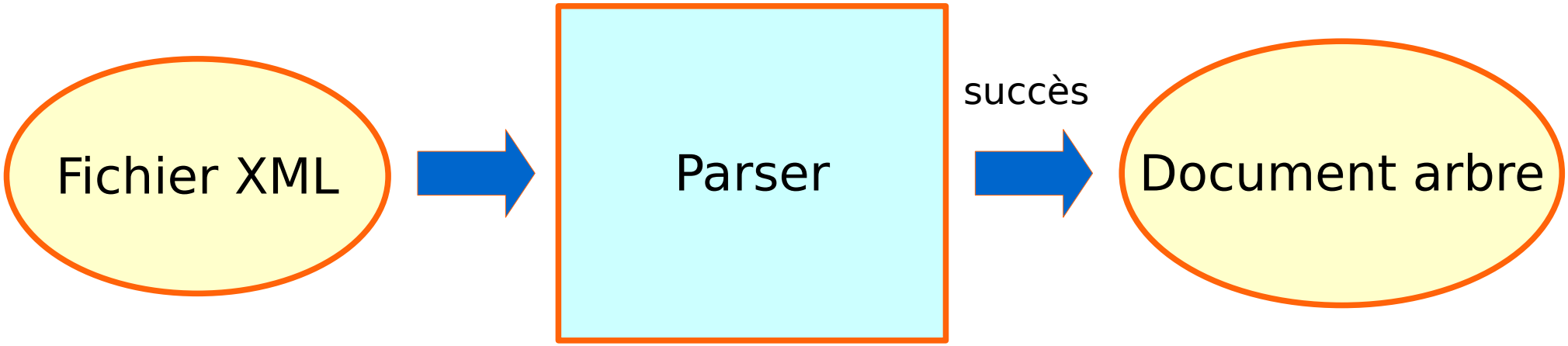





| | nodeName | nodeValue |
|---|---|---|
| Document | "#document" | null |
| Element | nom de balise | null |
| Attr | nom attribut | valeur attribut |
| Text | "#text" | texte |
| Comment | "#comment" | commentaire |

# Chargement d'un document XML :

- analyse lexicale et syntaxique :
    - le fichier est-il correctement formé ?
- production d'un arbre de la classe Document




En TD : classe DomUtils avec la fonction openXML

# Navigation dans un document DOM

Pour un document :
- getDocumentElement() : élément racine de l'arbre

Pour un élément :
- getChildNodes() : liste des fils d'un élément de type
    - Element
    - Text
    - Comment
- getAttributes() : liste des attributs d'un élément

Pour un noeud quelconque :
- getNodeName() : son nom (cf tableau)
- getNodeValue() : sa valeur (cf tableau)
- getNodeType() : son type (ELEMENT_NODE, TEXT_NODE, etc ...)

# Recherche dans un document XML

## A partir d'un document

- getElementById :
  - obtenir un élément par la valeur de son attribut ID unique
- getElementsByTagName :
  - obtenir une liste d'éléments par leur nom
- getElementsByTagNameNS :
  - obtenir une liste d'éléments par leur nom qualifié par un espace de nom (ex: xs:stylesheet)

## A partir d'un élément

- getElementByTagName / getElementByTagNameNS :
  - recherche uniquement à partir de cet élément
- getAttribute / getAttributeNS :
  - obtenir la valeur d'un attribut à partir de son nom
- getAttributeNode / getAttributeNodeNS :
  - obtenir un attribut à partir de son nom

# Modification d'un document XML

Pour un noeud quelconque :

- Remplacer la valeur du noeud :
  - setNodeValue(nouvelleValeur)

Pour un élément :

- Ajouter un fils
  - appendChild(nouveauFils)
- Supprimer un fils
  - removeChild(fils)
- Remplacer un fils
  - replaceChild(nouveauFils, ancienFils)
- Ajouter un attribut / Remplacer la valeur d'un attribut :
  - setAttribute(nomAttribut, valeurAttribut)
- Supprimer un attribut :
  - removeAttribute(nomAttribut)

Pour un document :

- createElement / createTextNode / createComment

=> un noeud est toujours créé dans le contexte de son document, puis rattaché au bon endroit dans l'arbre avec les méthodes ci-dessus

# Sauvegarde d'un document XML

Une fois le document XML traité par l'application sous la forme d'un arbre DOM,celui-ci peut-être sauvegardé dans un fichier XML (voir classe DomUtils en TD).


N'oubliez pas de consulter l'API Java pour connaître les détails de chaque méthodes et pour en découvrir d'autres !