haute école
neuchâtel berne jura
arc ingénierie
saint-imier le locle delémont

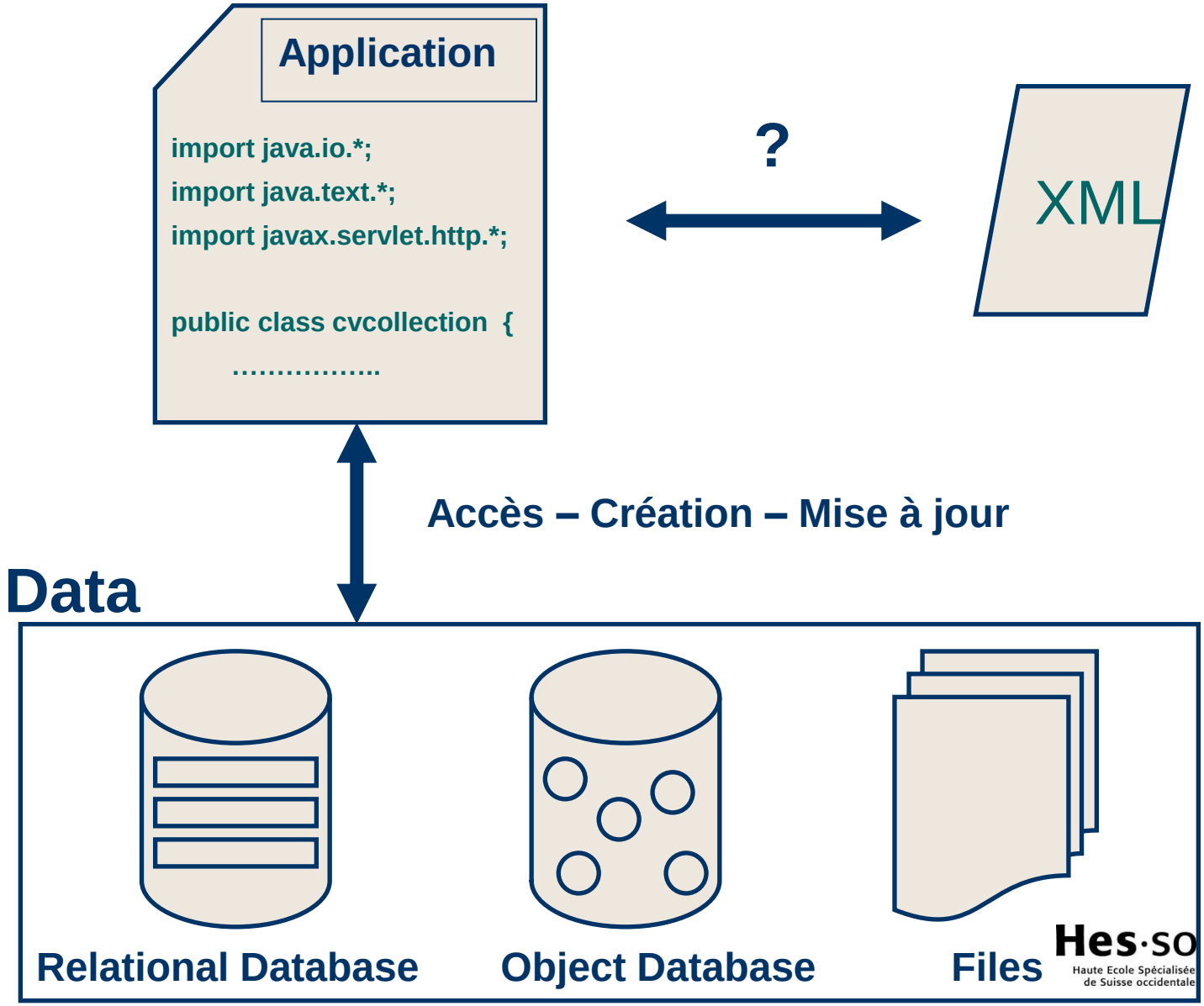
Séance 6

Parsing des documents XML

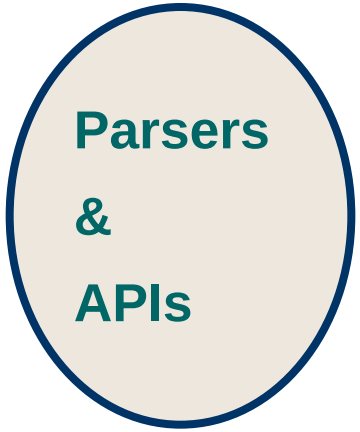
Prof. Yassin Aziz REKIK
Yassin.rekik@he-arc.ch

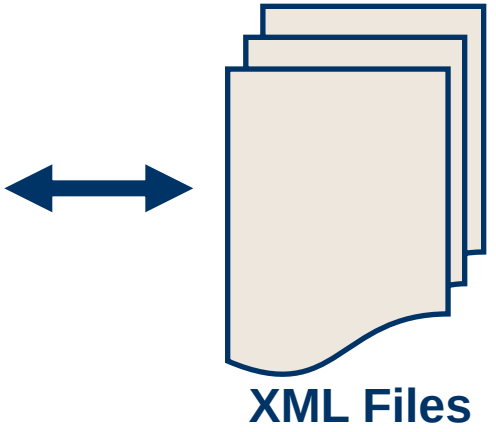
Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

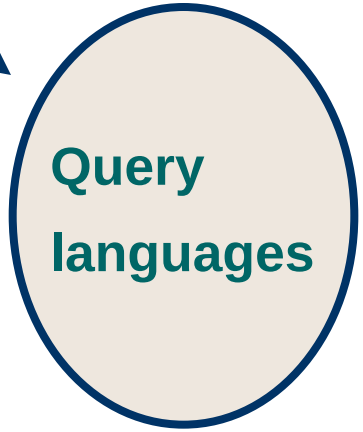
Manipulation des documents XML

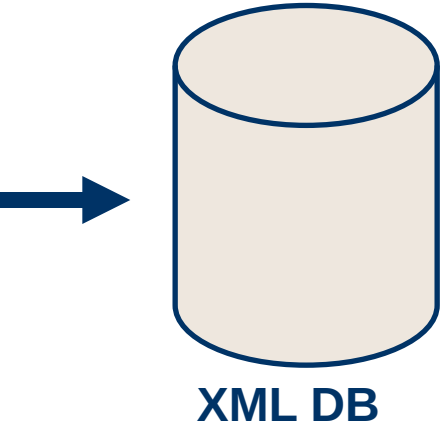


Deux approches de traitement

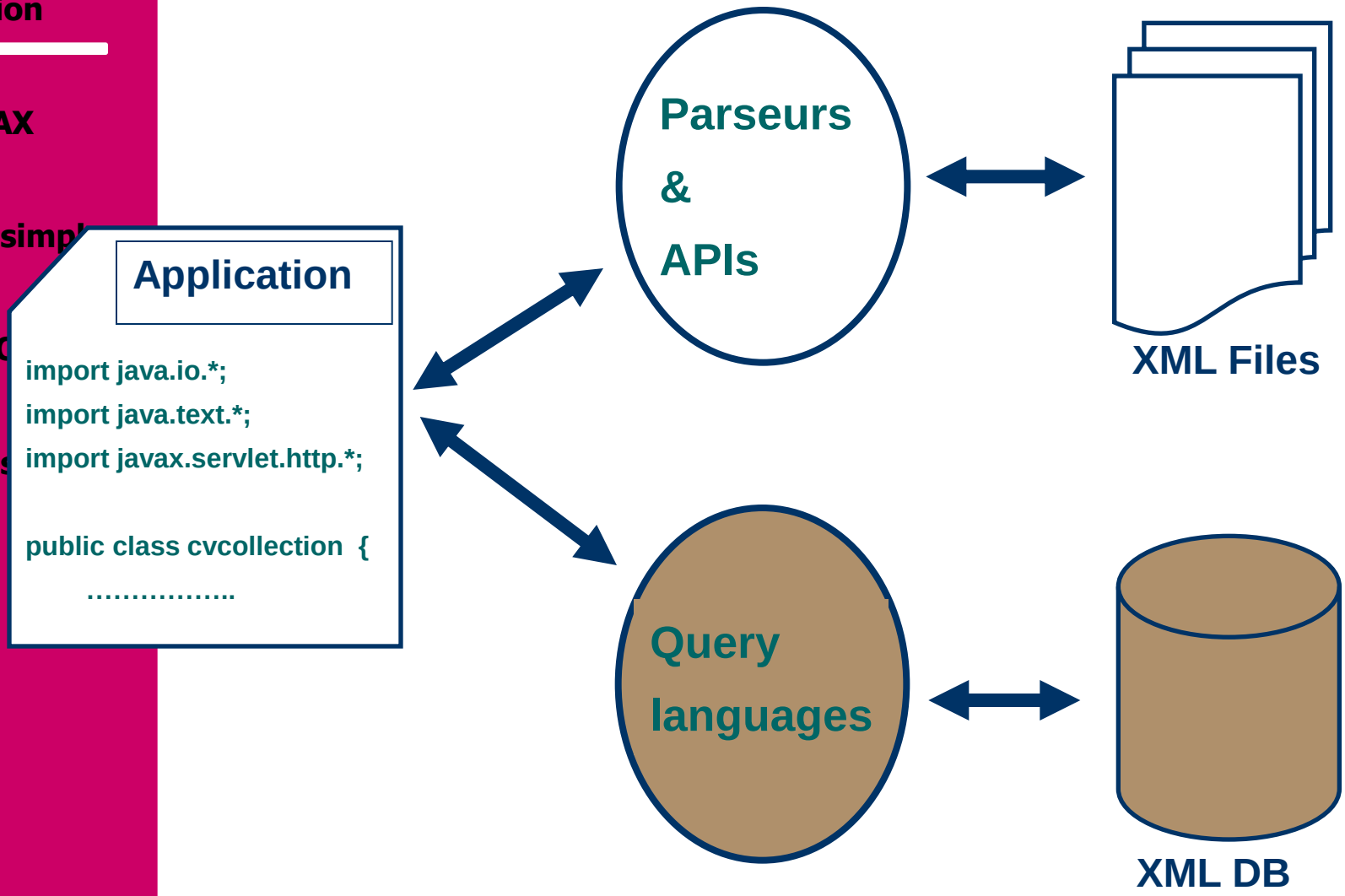






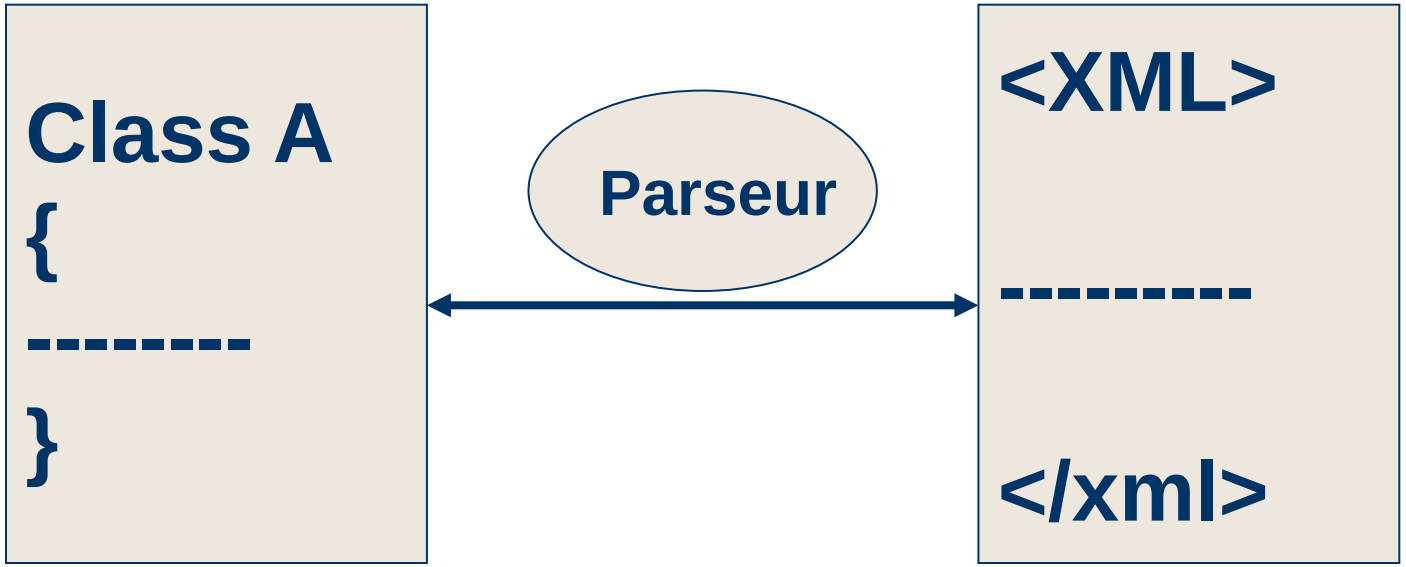


Approche basée parsing



Parseur XML

- **Un parseur XML est un outil (package, librairie, programme, …) permettant d'accéder, de créer et de mettre à jour des documents XML à travers un code.**



Parseur XML : fonctions

- **Un parseur doit assurer les fonctions suivantes:**
  - Analyse lexicale et syntaxique
  - Vérification de la conformité du document à la norme
  - Substitution des entités
  - Validation par rapport à une DTD ou un schéma
  - Manipulation de base : modifier, créer, supprimer, …
  - Etc. ….

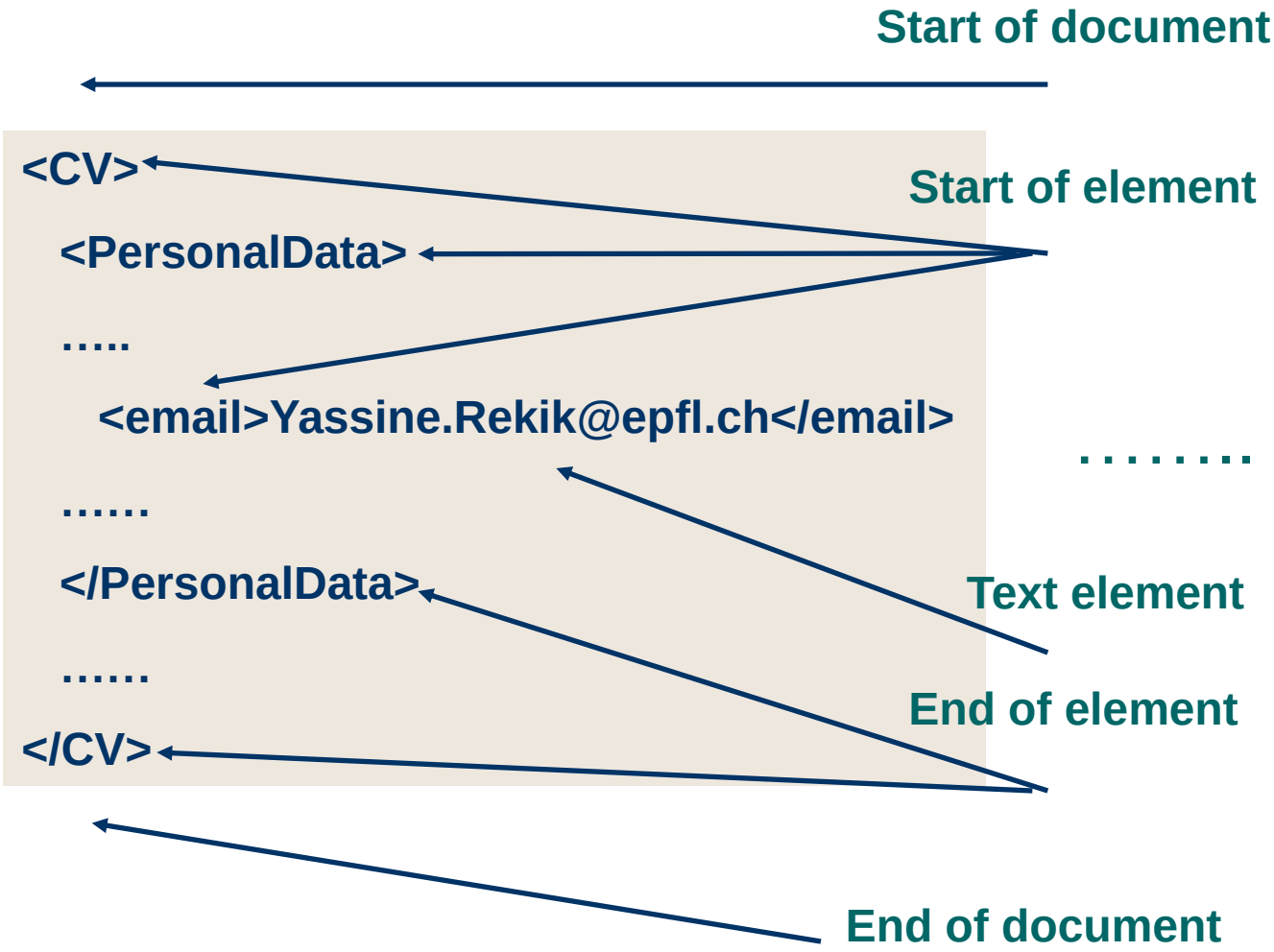
Un parseur est plus qu'un compilateur

Les modèles de parsing

- **Deux modèles de parsing XML**
  - Modèle événementiel - Event-based model
  - Modèle par compilation ou basé arbre - Tree-based model
- **Modele événementiel**
  - Standard de facto : SAX (Simple API for XML)
- **Modèle arborescent**
  - W3C standard : DOM (Document Object Model)
- **Les parseurs SAX et DOM existent pour presque tous les langages de programmation**
  - Java , C , C++ , Perl , TCL Tk , VB , …… , Prolog

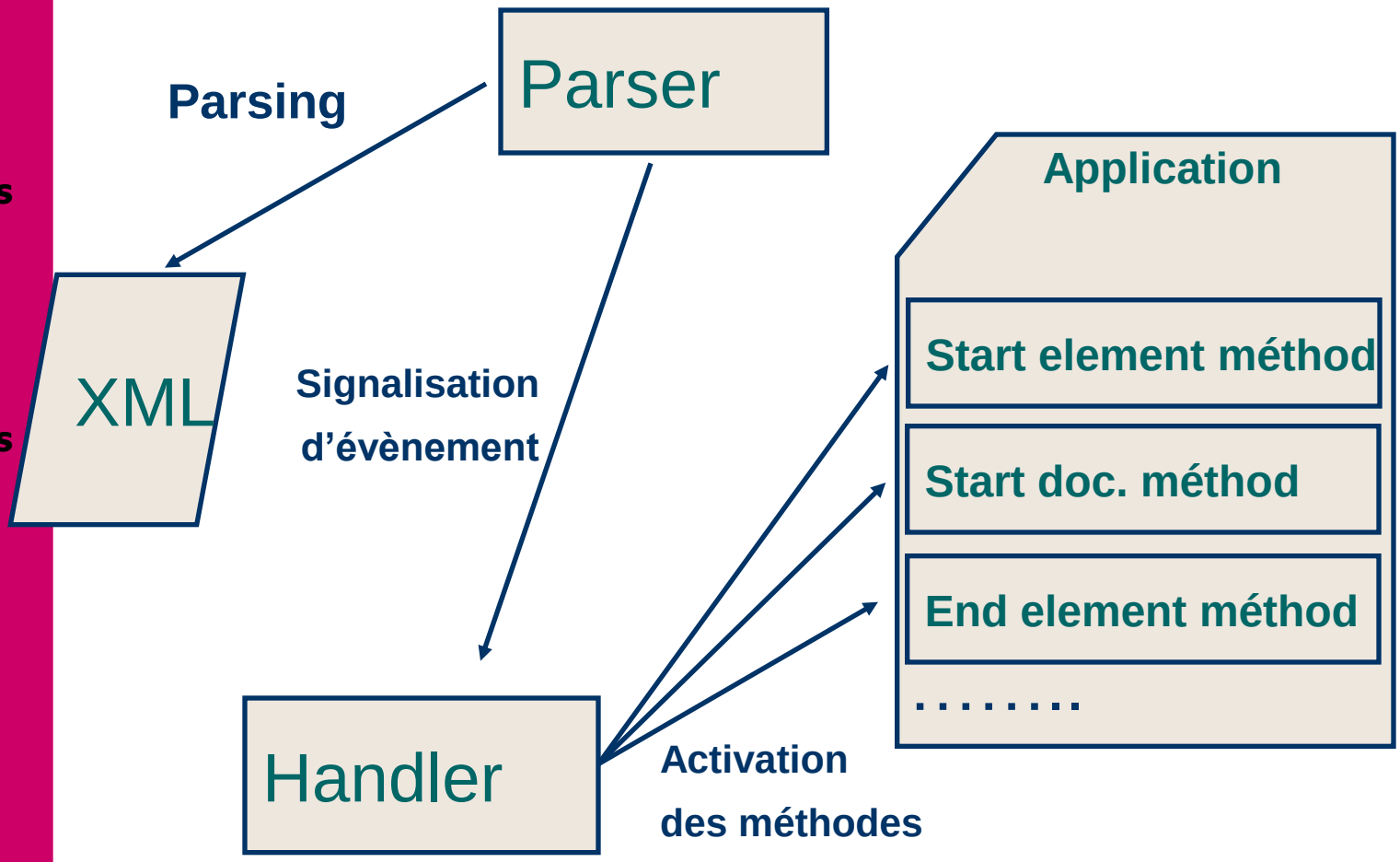
Parseurs et API SAX

- **Modèle événementiel**



Parseurs et API SAX

- **Modèle événementiel**



Parseurs et API SAX

- **Applications orientées SAX**
  - Décomposition de document
  - Translation de document
  - Extraction de fragments
  - ….

Accès et manipulation séquentiels

Parseurs et API SAX

- **Versions du SAX**
  - SAX 1 : introduit en Mai 1998
  - SAX 2.0 : introduit en Mai 2000, il supporte :
    - *Les namespaces*
    - *Des filtres de chaînes*
    - *Positionner et interroger des paramêtres sur le parsing*

Parseurs et API SAX

- **Implémentation en Java**
  - Extension de la classe HandlerBase

```
Import org.xml.sax.*;

import org.xml.sax.helpers.ParserFactory;


Public class SaxApplication extends HandlerBase
{
        public static void main(String args[]) {
        }
}
```

Parseurs et API SAX

- **Implémentation en Java**
  - Création du parseur

```
public static void main(args[]) {
        String parserName =
"org.apache.xerces.parsers.SAXParser";
        try {
                SaxApplication app = new
SaxApplication();
                Parser parser =
        ParserFactory.makeParser(parserName);
                parser.setDocumentHandler(app);
                parser.setErrorHandler(app);
                parser.parse(new InputSource(args[0]));
        } catch (Throwable t) {
                // Handle exceptions
        }
}
```

Parseurs et API SAX

- **Implémentation en Java**
  - Redéfinir, par surcharge, les évènements voulus

```
Public class SaxApplication extends HandlerBase {
        public void main (String args[]) {
                // stuff missing
        }

        public void startElement(String name,
AttributeList attrs) {
                // Process this element
        }
}
```

Parseurs et API SAX

- **Implémentation en Java**
- **Surcharger les autres évènements si besoin**
  - startDocument()
  - endDocument()
  - startElement()
  - endElement()
  - Characters
  - error()
  - …

Parseurs et API SAX

- **Exemples avec les CVs:**
  - Décomposer le documents CVs
  - Compter le nombre de candidats mariés
  - Compter le nombre de candidats femmes
  - Afficher en séquence un CV
  - …

Parseurs et API SAX

- **Analyse d'un exemple SAX**

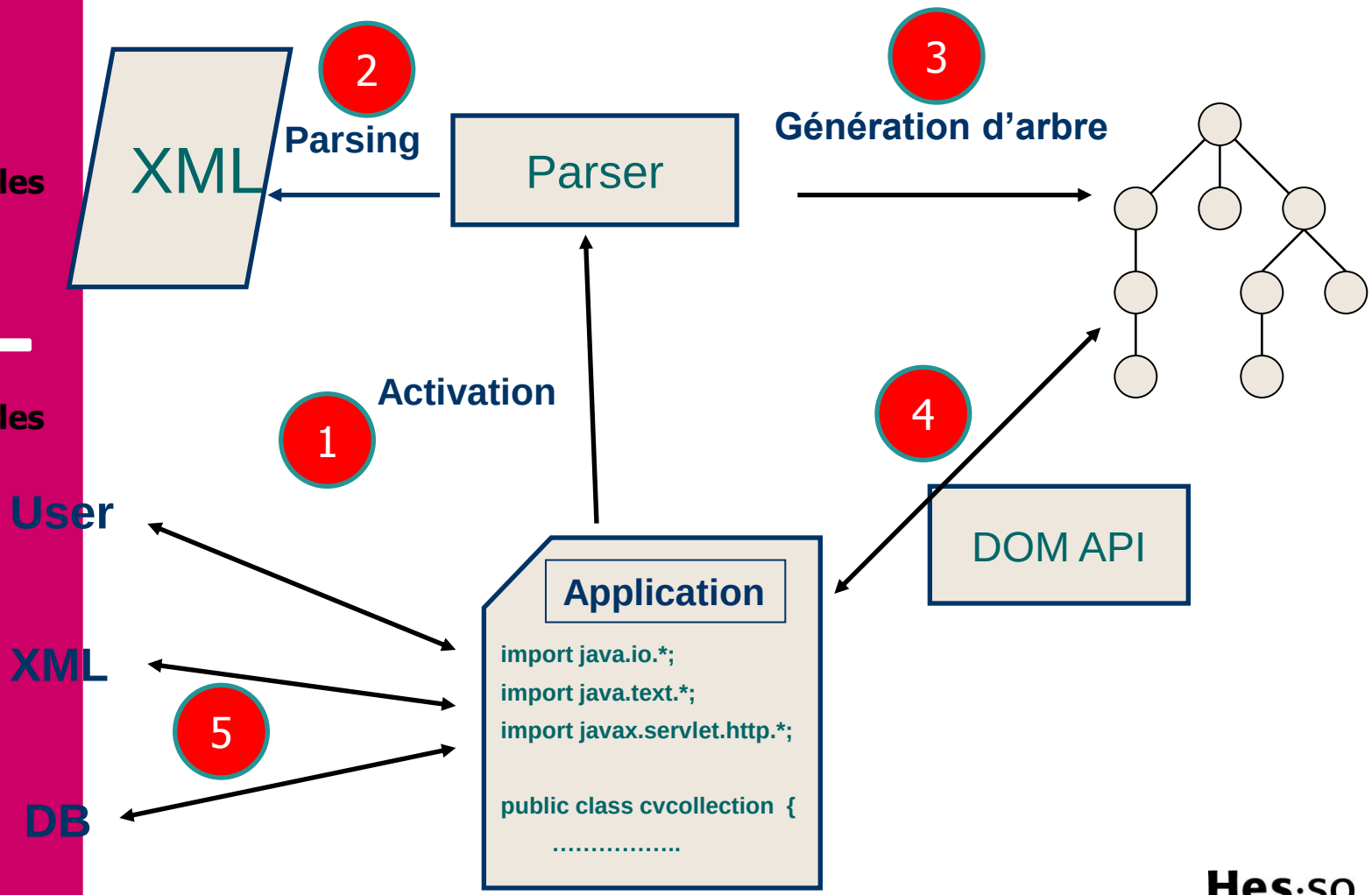
Code SAX

Exercice SAX

- **Utiliser un parsing SAX pour parser le document «CV.xml » et générer un document « CV.html »**
  - Pour simplifier, vous pouvez générer le html uniquement sur la sortie standard, pas dans un fichier.

Parseurs et API DOM

- **Modèle basé arbre**



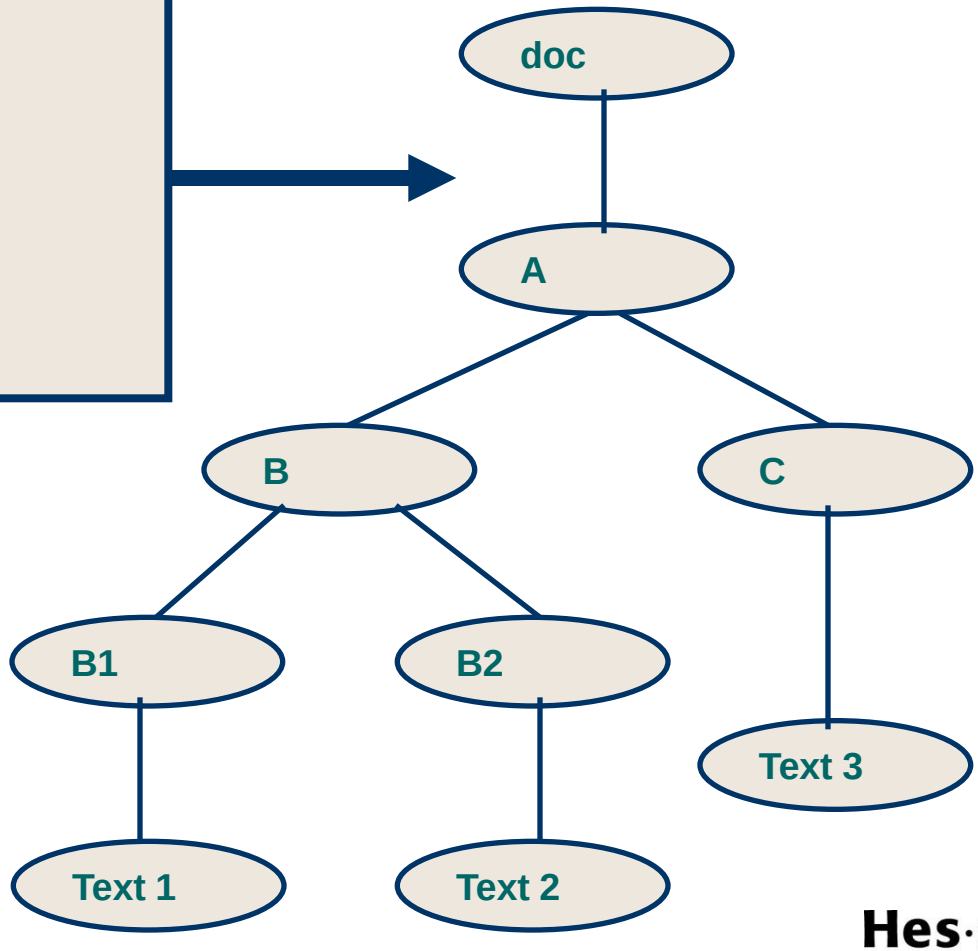
Parseurs et API DOM

- **Génération de l'arbre DOM**

```
<A>
  <B>
    <B1>Text 1</B1>
    <B2>Text 2</B2>
  </B>
  <C>text 3</C>
</A>
```

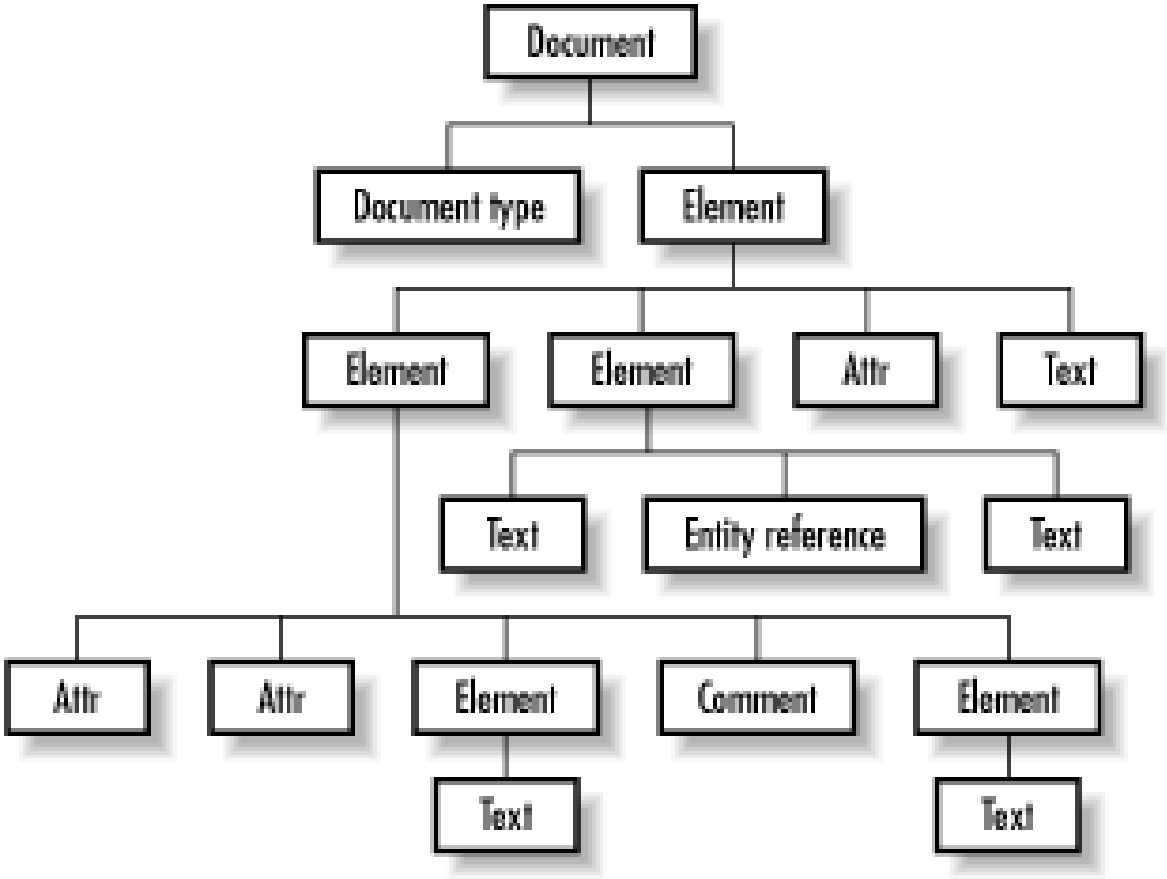
Document XML

Arbre DOM



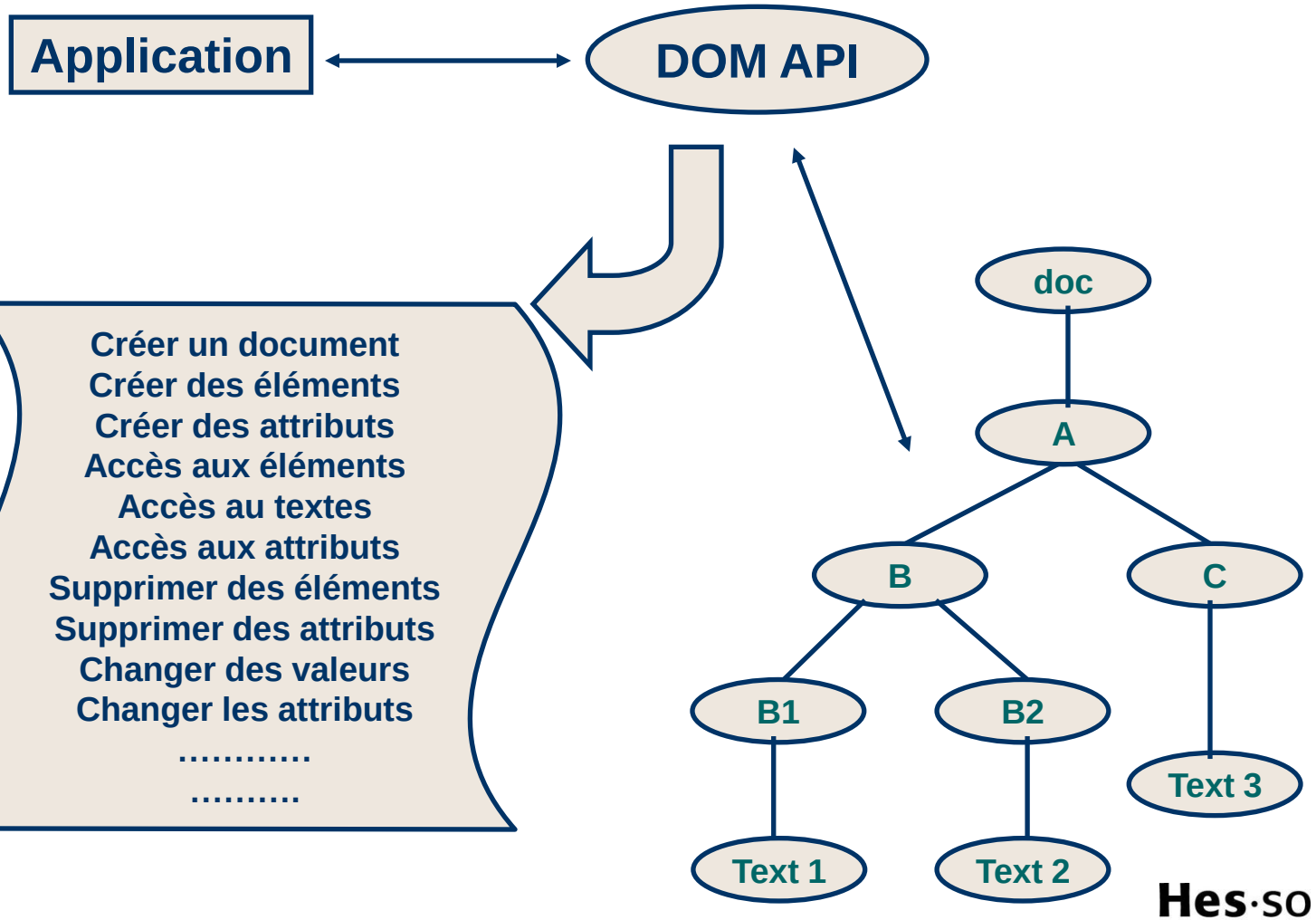
Parseurs et API DOM

- **Arbre DOM**

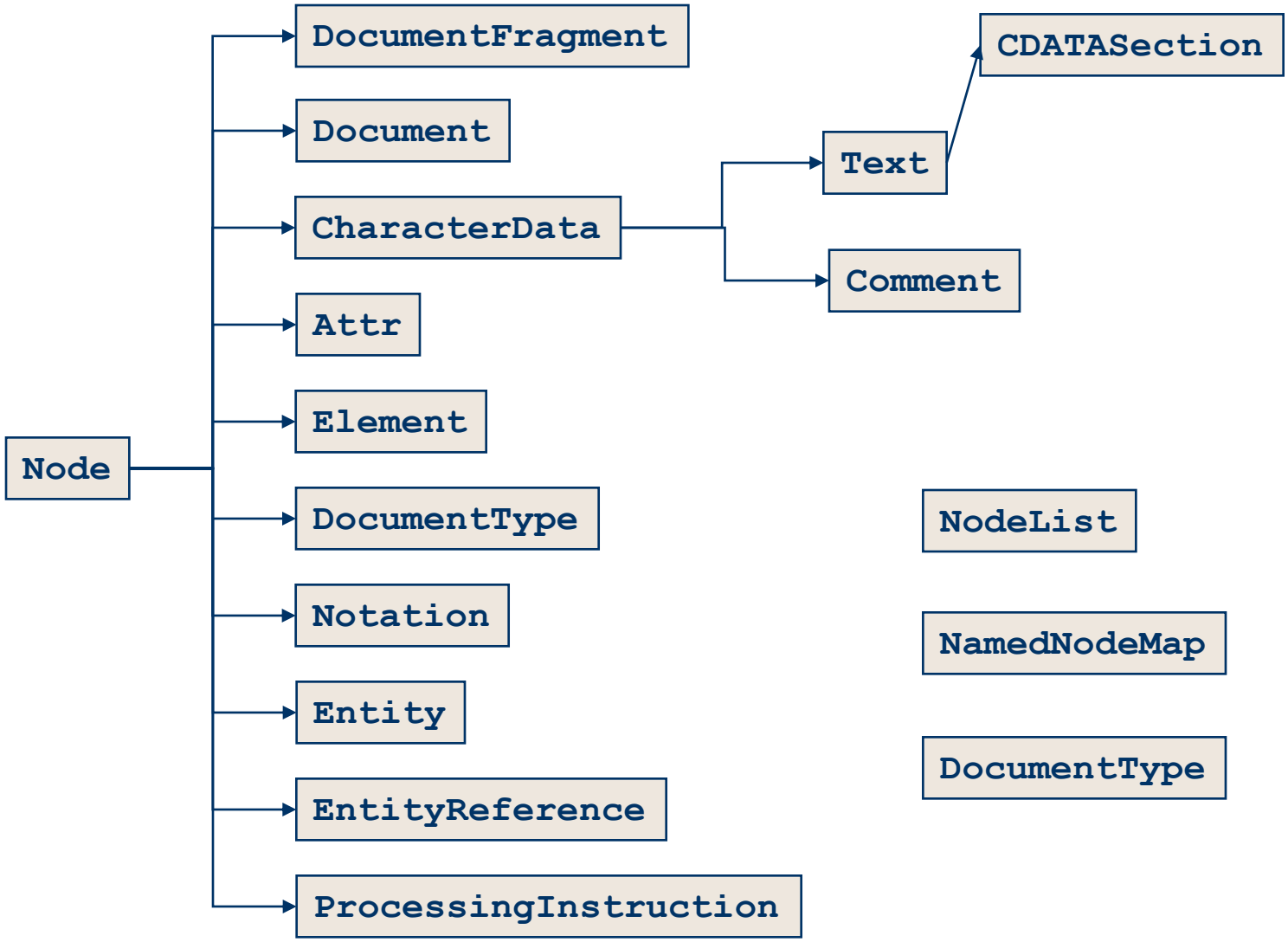


Parseurs et API DOM

- **API DOM**



API DOM



DOM : les interfaces

- **Le DOM définie plusieurs interfaces Java**
  - Node : Le type de base de tous les objets DOM
  - Element : Représente les éléments du document
  - Attr : Représente les attributs
  - Text : Le contenu des attributs ou des terminaux
  - Document : Représente le document XML en entier

DOM : Interface Node

- **Objet de base du DOM**
- **Les nodes décrivent:**

Elements	Entity declarations
Attributes	Entity references
Text	Notation declarations
Comments	Entire documents
CDATA sections	Processing instructions

- **Node collections**
  - NodeList, NamedNodeMap, DocumentFragment

DOM : Les méthodes Node

- **Trois catégories de méthodes:**
  - Node characteristics
    - *name, type, value*
  - Accès relatif et situation dans l'arbre
    - *parents, siblings, children, ancestors, descendants*
  - Modification de la Node
    - *Edit, delete, re-arrange child nodes*

DOM : Les méthodes Node


```
short       getNodeType();

String      getNodeName();

String      getNodeValue()
                                  throws DOMException;

void setNodeValue(String value)
                                  throws DOMException;

boolean     hasChildNodes();

NamedNodeMap getAttributes();

Document    getOwnerDocument();
```

DOM : Les types de Node

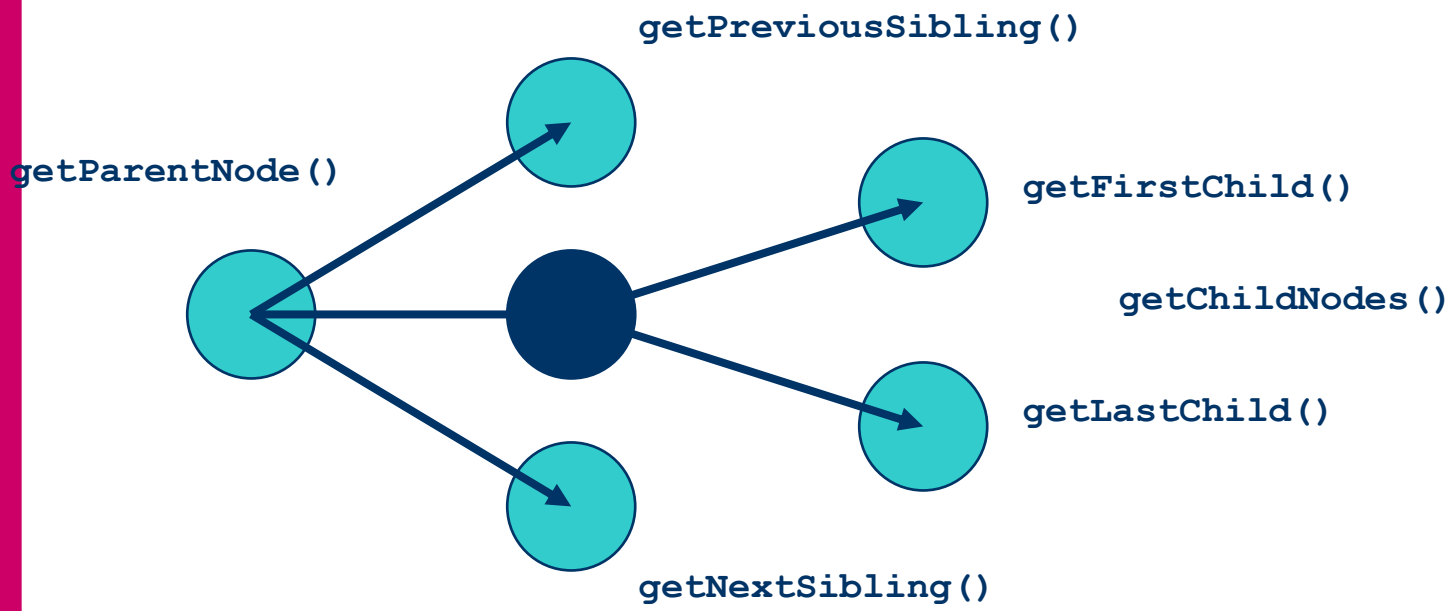

```
ELEMENT_NODE                = 1
ATTRIBUTE_NODE              = 2
TEXT_NODE                   = 3
CDATA_SECTION_NODE          = 4
ENTITY_REFERENCE_NODE       = 5
ENTITY_NODE                 = 6
PROCESSING_INSTRUCTION_NODE =  7
COMMENT_NODE                =  8
DOCUMENT_NODE               =  9
DOCUMENT_TYPE_NODE          = 10
DOCUMENT_FRAGMENT_NODE      = 11
NOTATION_NODE               = 12
```

```
if (myNode.getNodeType() == Node.ELEMENT_NODE) {
  //process node
  ...
}
```

DOM : Les méthodes de navigation

- **Chaque Node possède un emplacement spécifique dans l'arbre DOM**
- **L'interface Node spécifie des méthodes pour accéder aux environts d'un node :**
  - Node getFirstChild();
  - Node getLastChild();
  - Node getNextSibling();
  - Node getPreviousSibling();
  - Node getParentNode();
  - NodeList getChildNodes();

DOM : Les méthodes de navigation




```
Node parent = myNode.getParentNode();
if (myNode.hasChildren()) {
  NodeList children = myNode.getChildNodes();
}
```

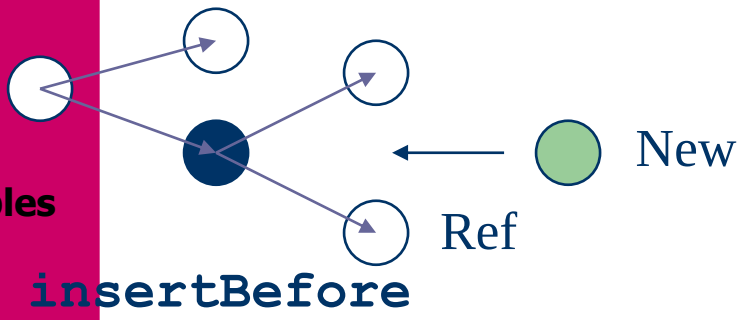
Figure from “The XML Companion” - Neil Bradley

DOM : Les méthodes de manipulation

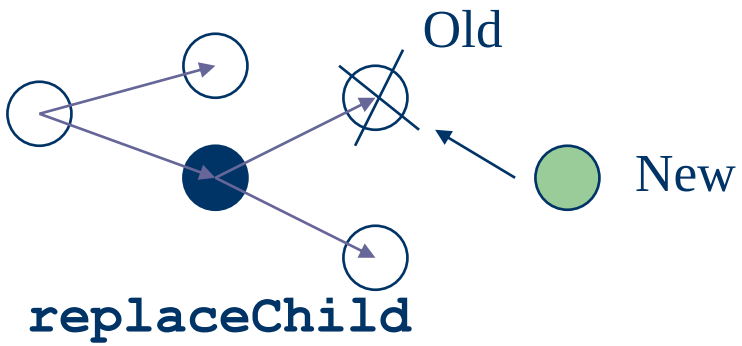
- **Les nodes fils dans un arbre DOM peuvent être manipulés: ajouter, éditer, supprimer, déplacer, copier, etc.**

```
Node removeChild(Node old)
                throws DOMException;

Node insertBefore(Node new, Node ref)
                throws DOMException;

Node appendChild(Node new)
                throws DOMException;

Node replaceChild(Node new, Node old)
                throws DOMException;

Node cloneNode(boolean deep);
```

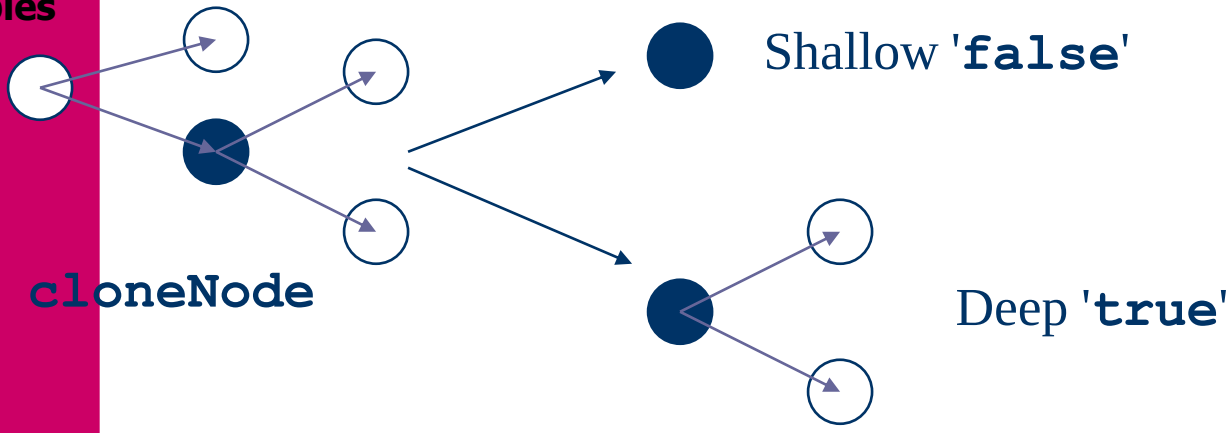
DOM : Les méthodes de manipulation



`insertBefore`



`replaceChild`



`cloneNode`

Figure from “The XML Companion” - Neil Bradley

DOM : Interface Document

- Réprésente le document entier : Root élément
- Méthodes

```
//Information from DOCTYPE - See 'DocumentType'
DocumentType      getDocumentType();

//Returns reference to root node element
Element           getDocumentElement();

//Searches for all occurrences of 'tagName' in nodes
NodeList          getElementsByName(String tagName);
```

DOM : Interface Document

- **Méthodes de création de nodes**

```
Element createElement(String tagName)
                                throws DOMException;

DocumentFragment createDocumentFragment();

Text createTextNode(String data);

Comment createComment(String data);

CDATASection createCDATASection(String data)
                                throws DOMException;

ProcessingInstruction createProcessingInstruction(
        String target, String data) throws DOMException;

Attr createAttribute(String name) throws DOMException;

EntityReference createEntityReference(String name)
                                throws DOMException;
```

DOM : Interface Element

- **Deux catégories de méthodes**
  - Méthodes générales sur les éléments

```
String   getTagName();
NodeList getElementsByTagName();
void     normalize();
```

  - Méthodes pour la gestion des attributs

```
String   getAttribute(String name);
void     setAttribute(String name, String value)
                              throws DOMException;
void     removeAttribute(String name)
                              throws DOMException;
Attr     getAttributeNode(String name);
void     setAttributeNode(Attr new)
                              throws DOMException;
void     removeAttributeNode(Attr old)
                              throws DOMException;
```

DOM : Interface Attr

- **Interface sur les nodes Attributes**

```
//Get name of attribute
String getName();

//Get value of attribute
String getValue();

//Change value of attribute
void setValue(String value);

//if 'true' - attribute defined in element,
//else in DTD
boolean getSpecified();
```

DOM : Interface CharacterData

- Permet de manipuler les éléments textuel
- Utilisée sur les élément text et comment

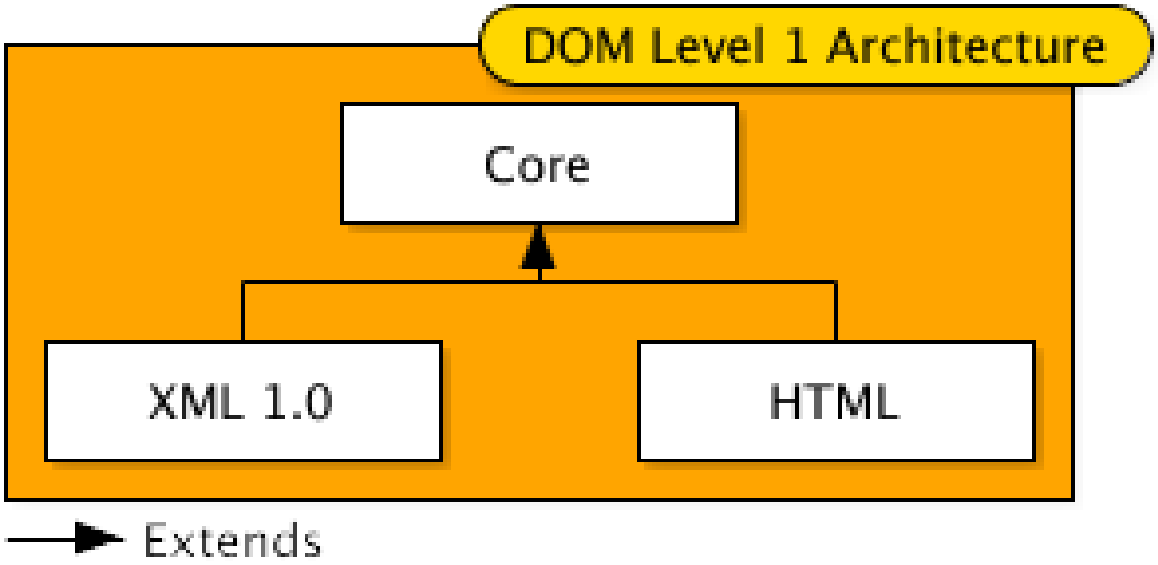
```
String getData()
                throws DOMException;
void   setData(String data)
                throws DOMException;
int    getLength();
void   appendData(String data)
                throws DOMException;
String substringData(int offset, int length)
                throws DOMException;
void   insertData(int offset, String data)
                throws DOMException;
void   deleteData(int offser, int length)
                throws DOMException;
void   replaceData(int offset, int length,
String data)
                throws DOMException;
```

DOM : Interface Text

- **Représente le contenu textuel de `Element` ou `Attr`**
  - Généralement, se sont des fils de ces éléments
- **Toujours des objets terminaux**
- **Méthode ajouter à l'interface Character Data**
  - `Text splitText(int offset) throws DOMException`
- **Factory method dans `Document` pour la création**
- **La méthode `normalize()` sur `Element` concatène les `Text` objets de ce dernier**

Parseurs et API DOM

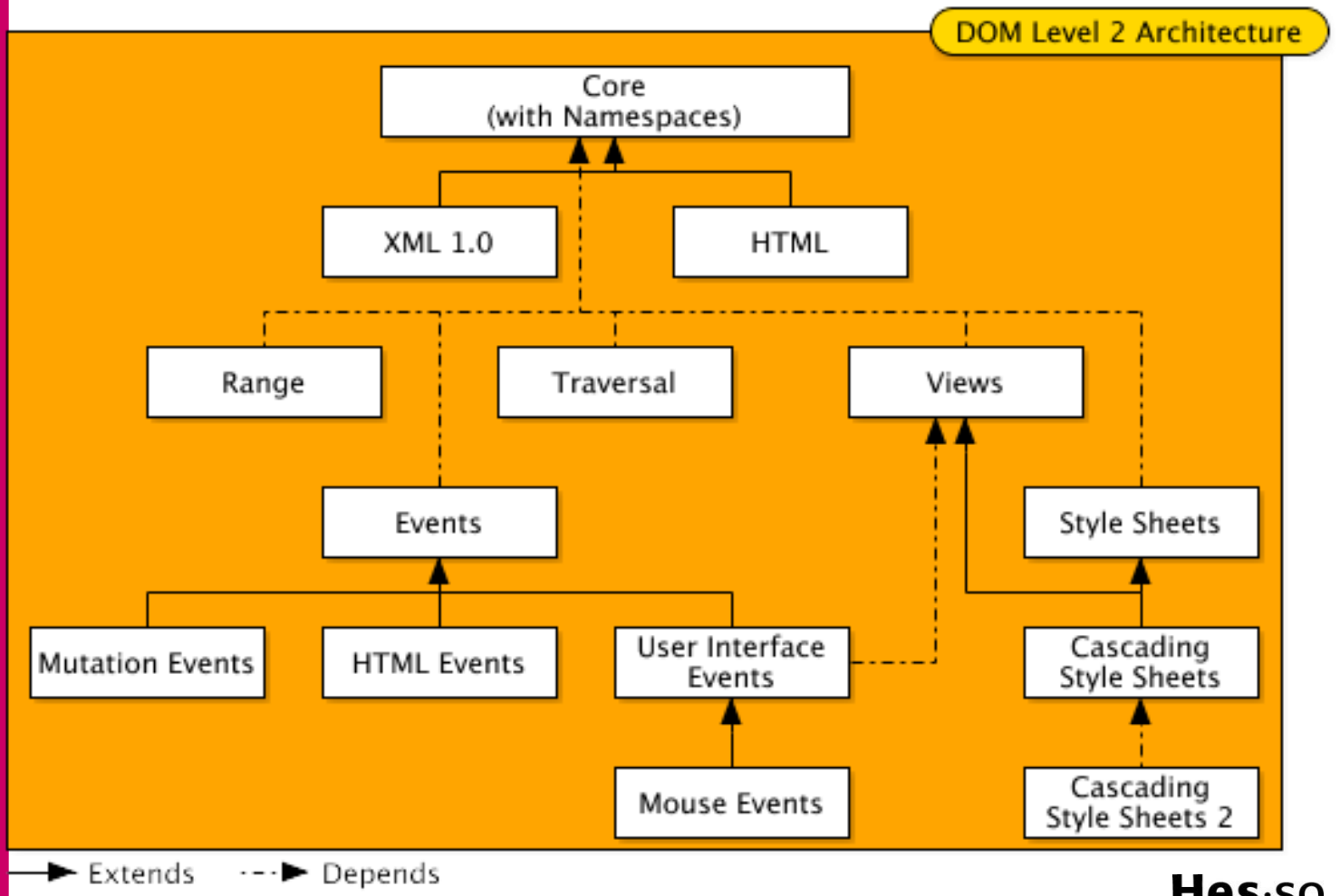
- **Evolution du DOM : Level 1**



W3C's figure

Parseurs et API DOM

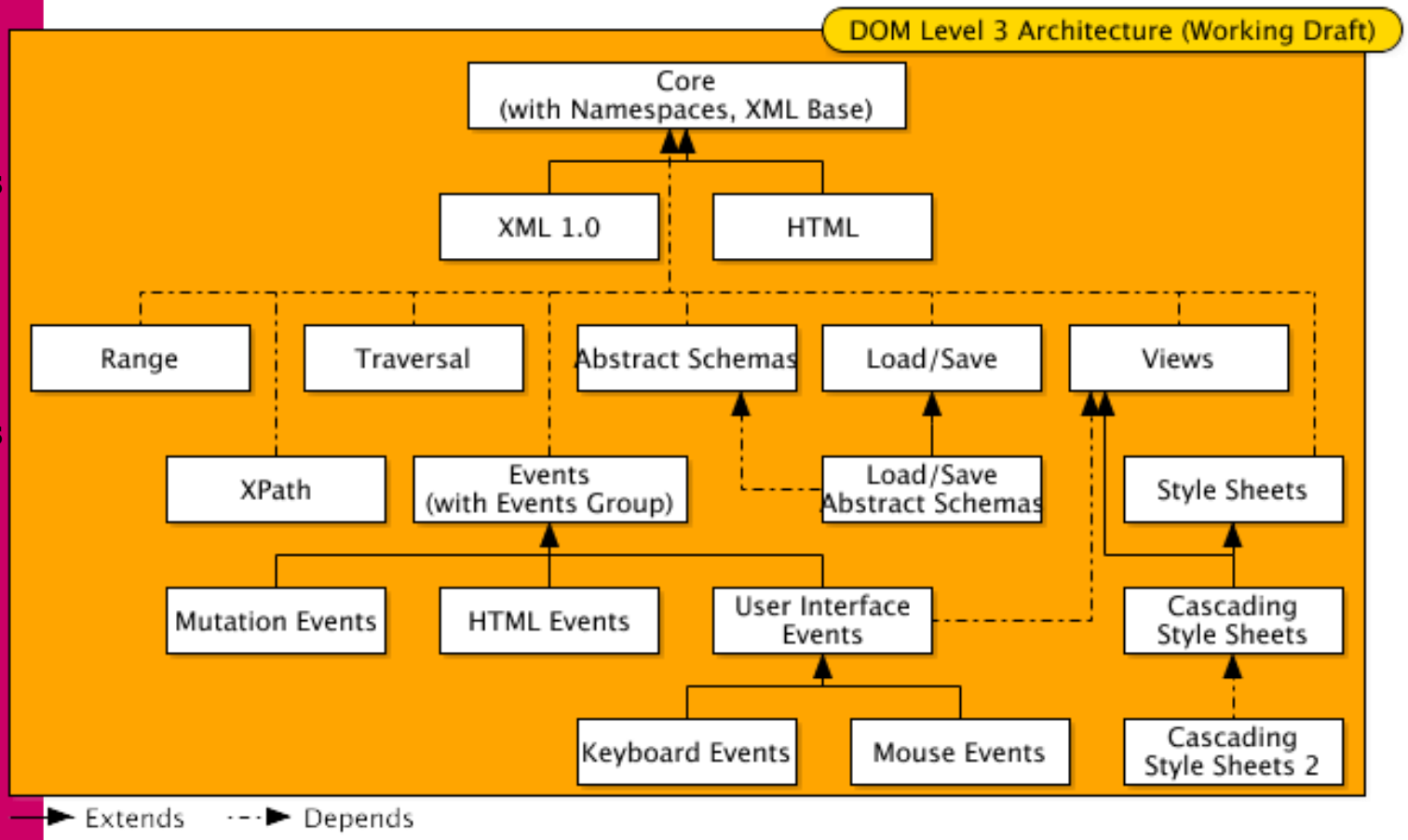
- **Evolution du DOM : Level 2**



W3C's figure

Parseurs et API DOM

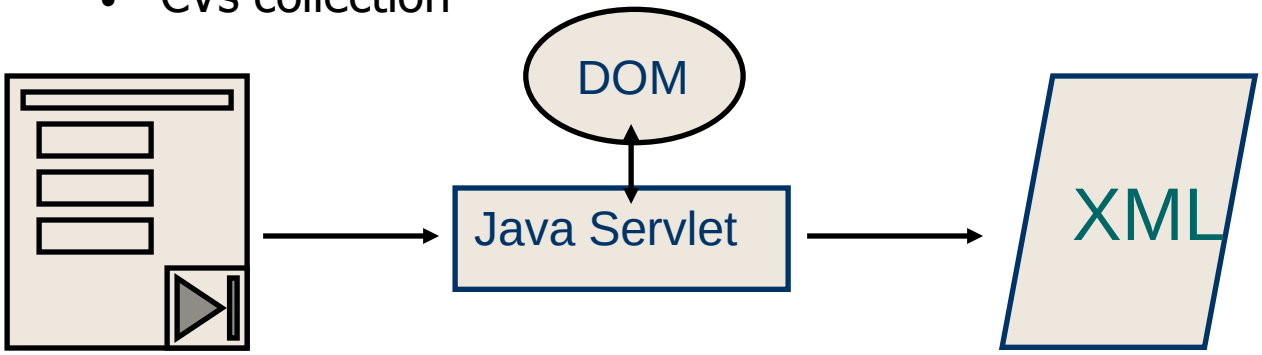
- **Evolution du DOM : Level 3**



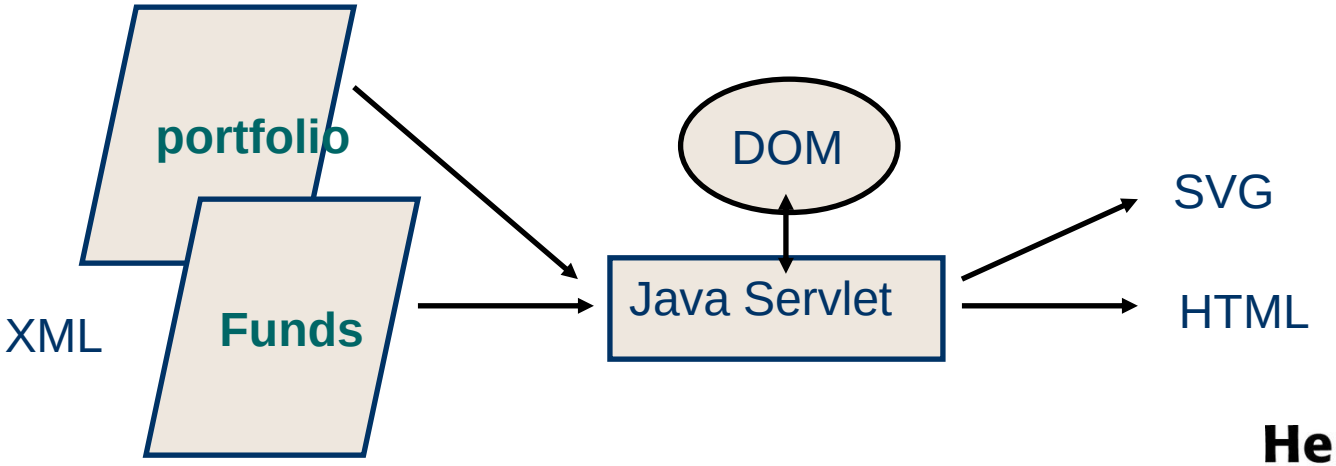
W3C's figure

Parseurs et API DOM

- **Exemples DOM**
  - CVs collection



  - Portfolio Management



Parseurs et API DOM

- Analyse d'exemple de code DOM

Code DOM

Exercice DOM

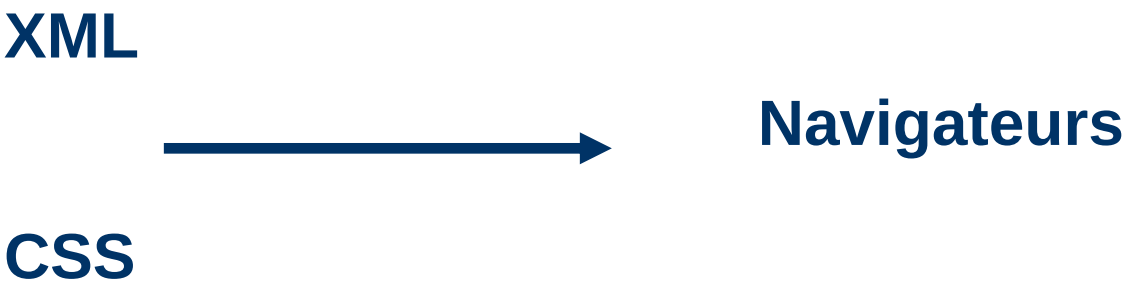
- **Utiliser un parsing DOM pour parser le document «CV.xml » et générer un document « CV.html »**
  - Pour simplifier, vous pouvez générer le html uniquement sur la sortie standard, pas dans un fichier.

XML et Web

- **Applications basées sur le Web : croissance**
- **Manipulation des documents XML via le Web**
  - Besoin courant
- **Types de manipulations**
  - Accès et publication : consommation
  - Création et mise à jour : production

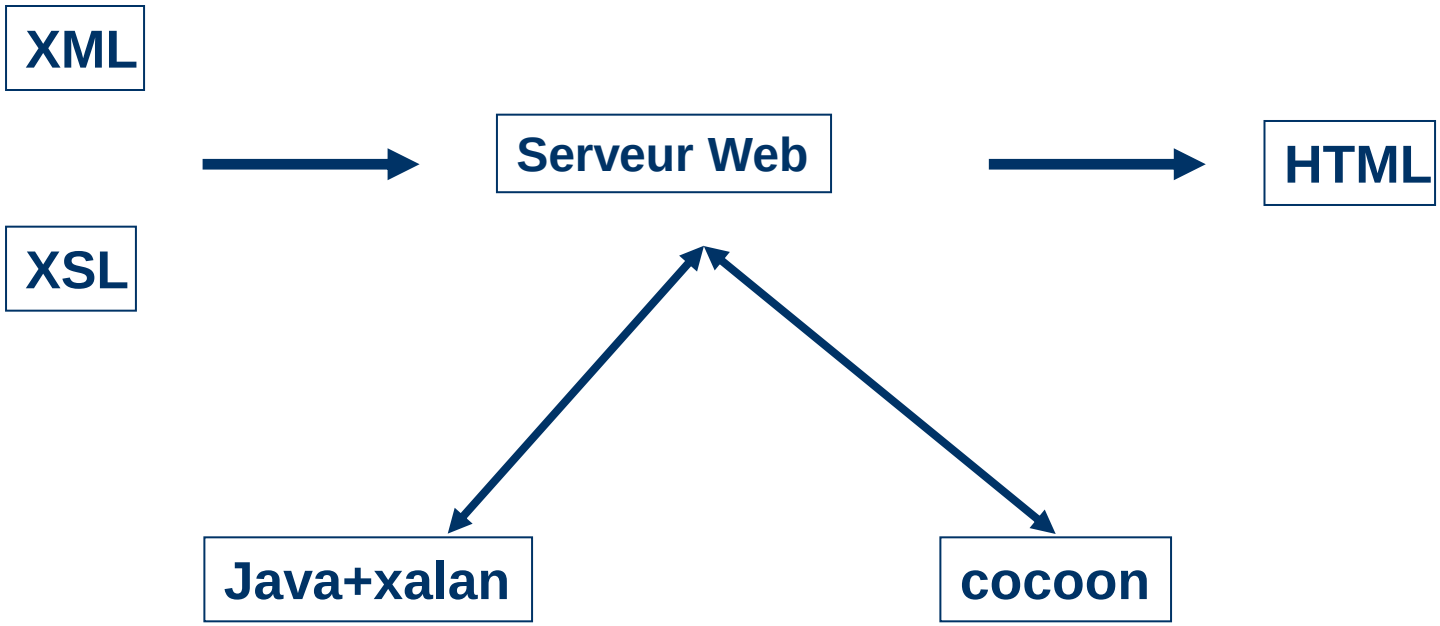
XML et Web : consommation

- **Architecture 1 : statique**



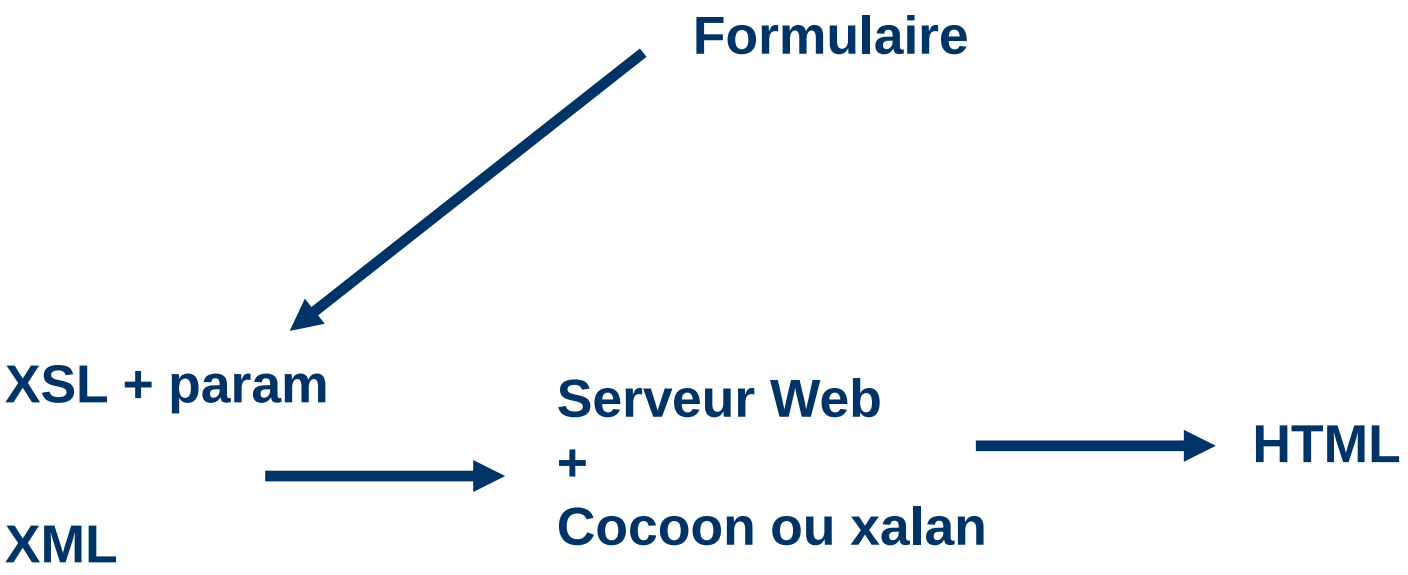
XML et Web : consommation

- **Architecture 2 : statique**



XML et Web : consommation

- **Architecture 3 : paramétrable**

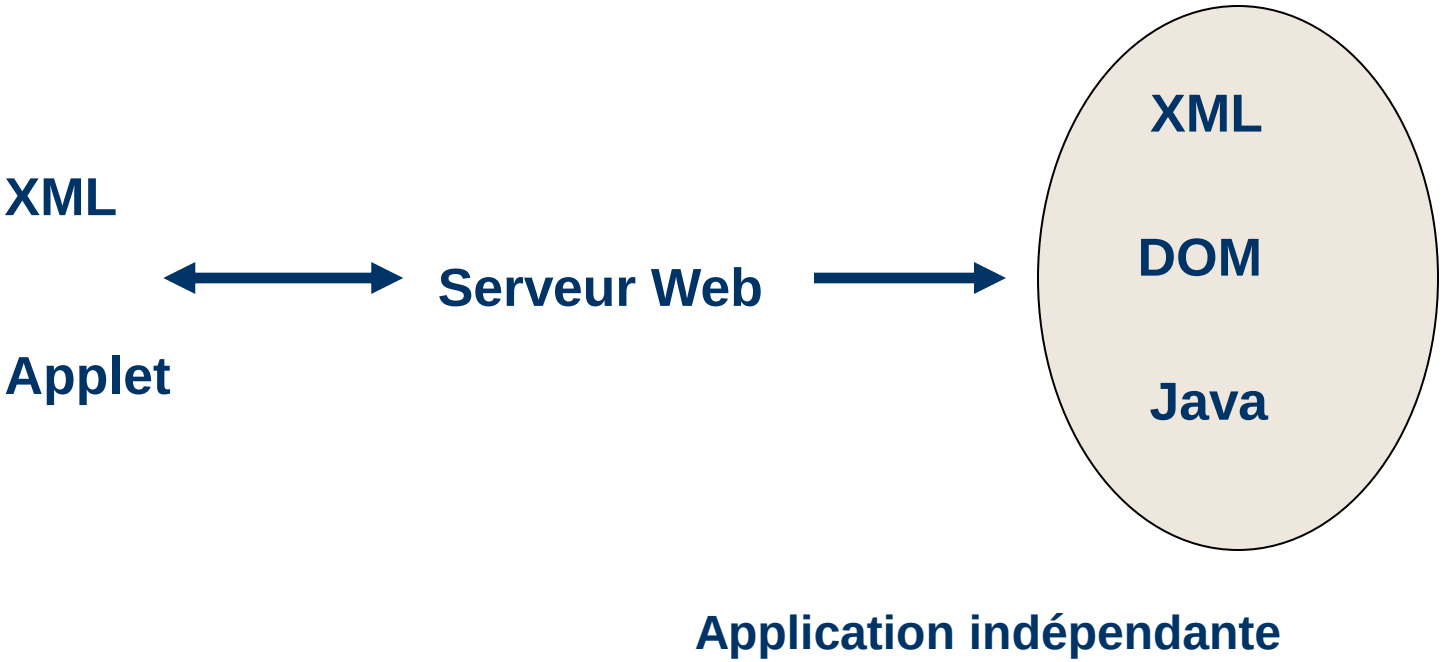


XML et Web : consommation

- **Architecture 4 : dynamique**

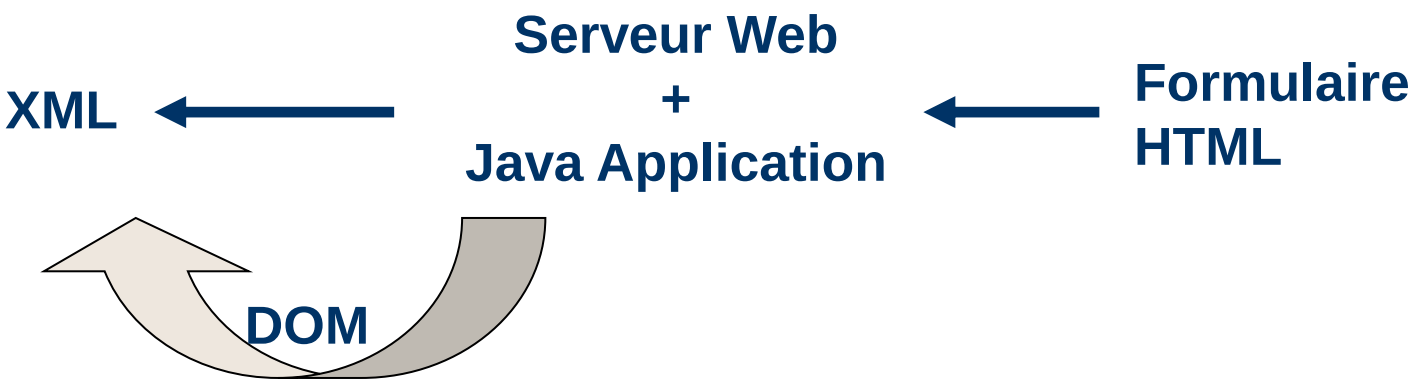
XML et Web : consommation

- **Architecture 5 : XML coté client**



XML et Web : production

- **Architecture 1 : Application serveur**



XML et Web : production

- **Architecture 2 : XML crée coté client**