DOM

DOM

- *"The W3C Document Object Model (DOM) is a platform and language neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.“*
- DOM spécifie un ensemble d’objets pour la description de documents XML ou HTML et une interface pour accéder et manipuler ces documents

DOM

DOM est séparé en trois parties :(Core, XML, HTML) et différents niveaux (level) : 1, 2, 3

- Core DOM : définie un ensemble d'objets pour un document structuré
- XML DOM : définie un ensemble d'objets pour un document XML
- HTML DOM : définie un ensemble d'objets pour un document HTML

DOM

Les spécifications existent en 3 niveaux :

- DOM level 1
  - Core : interface de base et leurs applications aux documents XML
  - HTML: interfaces applicables aux documents HTML (HTML 4)
- DOM level 2 :
  - Complète la version Core pour la gestion des espaces de noms
  - DOM CSS : prise en charge des feuilles de style
  - DOM Events : Modèle d'événements
  - DOM Filters and Iterators : filtrage d'éléments et traitements itératifs
  - DOM Range : isoler et traiter des fragments de documents
- DOM level 3 :

  intègre une gestion d'événements à la SAX pour générer un seul objet pour des éléments multiples (ce qui permettra de mapper un document XML sur des objets applicatifs complexe).

DOM

- Avec le Modèle Objet de Document (DOM), les programmeurs peuvent:
  - construire des documents
  - naviguer dans leur structure
  - et ajouter, modifier, ou supprimer soit des éléments, soit du contenu.
  - Tout ce qui peut être trouvé dans un document HTML ou XML peut être accédé, changé, détruit, ou ajouté en utilisant DOM

DOM

- Le DOM fournit une interface de programmation objet (API) indépendante des langages de programmation
- Des implémentations DOM peuvent être écrites dans le langage de votre choix
- Le DOM est en général ajouté comme couche intermédiaire entre le parseur XML et l'application qui a besoin des Informations du document
- Le parseur lit les données du document XML, les transmet à un DOM qui va être utilisé par une application de plus haut niveau.

DOM

- En utilisant DOM le document n'est plus lu de façon linéaire mais il est parcouru plusieurs fois par l'analyseur et stocké entièrement en mémoire sous forme d'un arbre.
- Contrairement à SAX, cette API présente l'intérêt de pouvoir tenir compte de la hiérarchie des éléments, et d'ajouter, supprimer ou modifier des éléments de cet arbre.

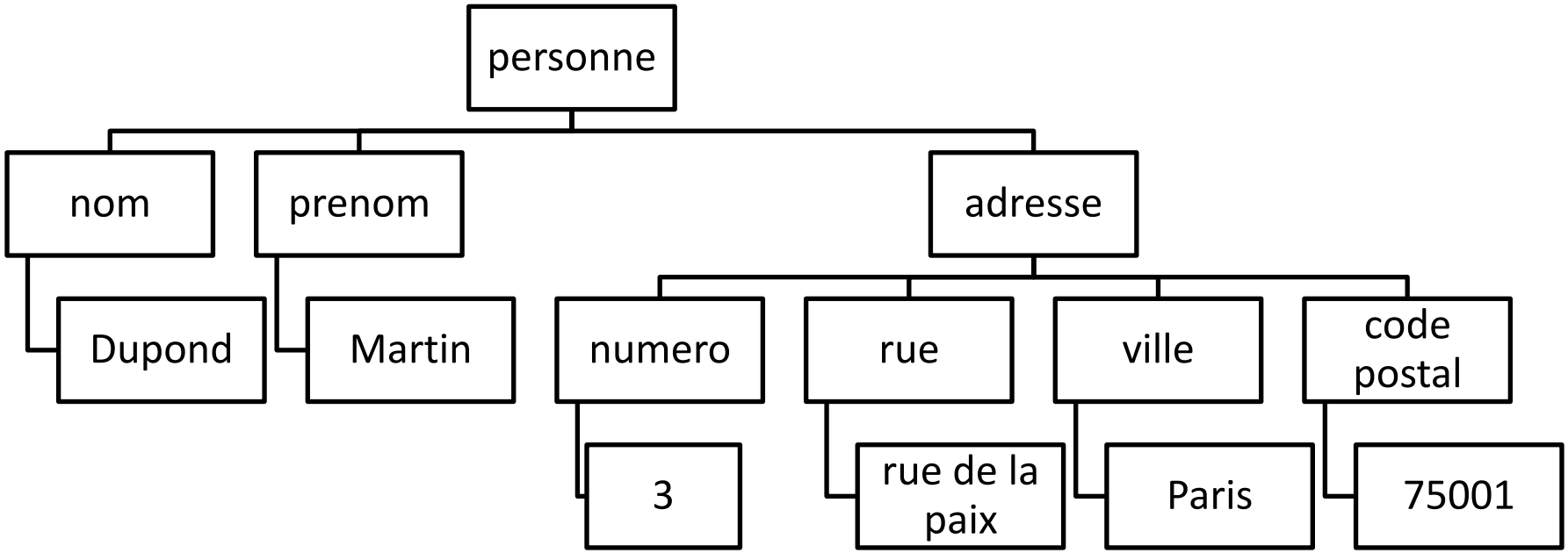
DOM

Ainsi le fichier XML suivant :

```
<personne>
    <nom>Dupond</nom>
    <prenom>Martin</prenom>
    <adresse>
        <numero>3</numero>
        <rue>rue de la paix</rue>
        <ville>Paris</ville>
        <codePostal>75001</codePostal>
    </adresse>
</personne>
```

DOM

peut être transformé en arbre suivant:



DOM

Quand utiliser DOM ?

- Pour les documents XML qui ne sont pas trop gros (sinon utilisation de mémoire et cpu importante)
- Lorsqu'on doit créer ou modifier un fichier XML
- Lorsque la hiérarchie des balises est importante ou que l'on doit parcourir différents niveaux dans l'arbre

DOM

Création d'un arbre DOM

- Il existe deux façons pour créer un arbre DOM :
  - Soit on créé un arbre vide qui ne contient que la racine, puis on ajoute des branches et des sous branches pour obtenir un arbre complet
  - Soit on génère un arbre en parsant un flux XML à partir d'un fichier par exemple.

DOM

Création d'un arbre DOM

- Pour ces deux alternatives :
  - il faut tout d'abord initialiser une instance de **DocumentBuilderFactory**.
  - Puis à partir de cette instance on créé un **DocumentBuilder** qui va permettre de créer un **Document** qui sera un arbre vide ou un arbre généré à partir d'un flux.

DOM

Parcours d'un arbre DOM

- Pour accéder aux éléments dans un arbre, il existe plusieurs méthodes de la classe **Node** qui permettent de parcourir l'arbre:
  - ***Node getFirstChild()***
  - ***Node getLastChild()***
  - ***Node getNextSibling()***
  - ***Node getPreviousSibling()***
  - ***Node getParentNode()***
  - ***NodeList getChildNodes()***
  - ***boolean hasChildren()***

DOM

Modification d'un arbre DOM

- Les Nodes fils dans un arbre DOM peuvent être manipulés :
  Les méthodes suivantes permettent d'ajouter, éditer, supprimer, déplacer et copier un nœud.
  - ***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)***