Plan

Modélisation

- Modélisation et Preuve de programmes
  - Logique des propositions : CM1, CM2, TD1, TD2, TP1
  - Logique des prédicats : CM3, TD3, TP2
  - Preuve de programmes fonctionnels : CM4, TD4, TP3
  - Preuve de programmes impératifs : CM5, TD5, TP4
- Modélisation des langages : CM6, CM7, TD6, TD7, TP5, TP6, TP7

Modélisation et Analyse des Informations Structurées

- Communication = Echange d'informations
- Besoins :
  - Représenter les informations possibles
  - Reconnaitre une information
  - Exploiter une information
- Organisation stratifiée : information structurée
- Informatique : Science du traitement de l'information
- Computer science : Science de la « machine à calculer »
- Essentiel :
  - Description et manipulation de l'information (langage),
  - Traitement d'une information quelconque,
  - Traitement d'une manipulation quelconque
- D'où :
  - Description formelle du langage
  - Génération automatique des outils de manipulation

Références bibliographiques

- Stern, Fondements mathématiques de l'informatique, McGraw-Hill, 1990.
- Hopcroft, Ullman, Introduction to automata theory, languages and computation, Addison-Wesley, 1979.
- Aho, Sethi, Ullman, Compilateurs : Principes, Techniques et Outils, InterEditions, 1989.
- Fisher, Leblanc, Crafting a compiler in ADA/in C, Benjamin Cummings, 1991.
- Wilhem, Maurer, Les compilateurs : Théorie, construction, génération, Masson, 1994.
- WWW Consortium : `http://www.w3c.org`.
- E-R. Harold, W-S. Means, XML in a nutshell, O'Reilly, 2001.
- JSON : `http://json.org`

Exemple : fichier /etc/hosts

- Fichier tel qu'il est affiché :

```
# Ceci est un commentaire

127.0.0.1       →hal9000 localhost

# En voici un autre

147.127.18.144       →phoenix.enseeiht.fr
```

- Informations brutes : caractères

```
0000000    #  sp   C   e   c   i  sp   e   s   t  sp   u   n  sp   c   o
0000020    m   m   e   n   t   a   i   r   e  nl  nl   1   2   7   .   0
0000040    .   0   .   1  ht   h   a   l   9   0   0   0  sp   l   o   c
0000060    a   l   h   o   s   t  nl  nl   #  sp   E   n  sp   v   o   i
0000100    c   i  sp   u   n  sp   a   u   t   r   e  nl  nl   1   4   7
0000120    .   1   2   7   .   1   8   .   1   4   4  ht   p   h   o   e
0000140    n   i   x   .   e   n   s   e   e   i   h   t   .   f   r  nl
0000160
```

Analyse lexicale

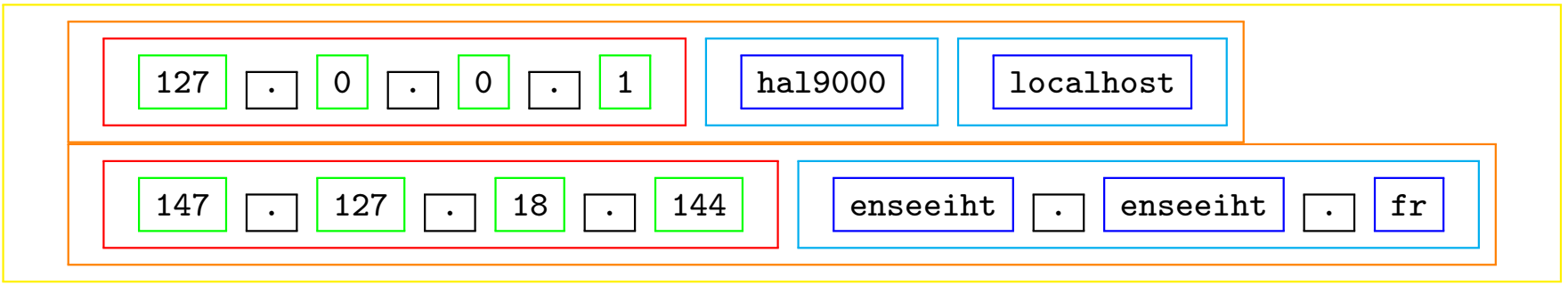
- Informations élémentaires : commentaire, nombre, identificateur, `.` (unités lexicales)
- Résultat de l'analyse lexicale :

`# Ceci est un commentaire` `127` `.` `0` `.` `0` `.` `1`
`hal9000` `localhost` `# En voici un autre` `147` `.`
`127` `.` `18` `.` `144` `enseeiht` `.` `enseeiht` `.` `fr`

- Spécification des unités lexicales : Expressions régulières
  - Commentaire : $\#[\hat{}\backslash n]^{\star}\backslash n$
  - Nombre : $[0-9]^{+}$
  - Identificateur : $[a-bA-B][a-bA-B0-9]^{\star}$

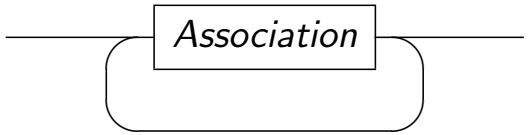
Analyse syntaxique

- Informations structurées (unités syntaxiques) :
  - Premier niveau : adresse IP, nom qualifié
  - Deuxième niveau : assocation
  - Troisième niveau : document

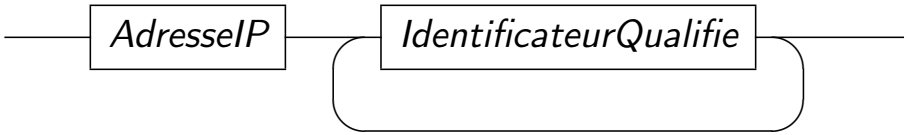


- Spécification des unités syntaxiques : Grammaires (notation de Conway)

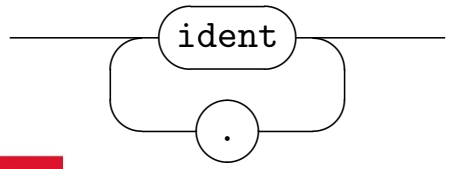
Document



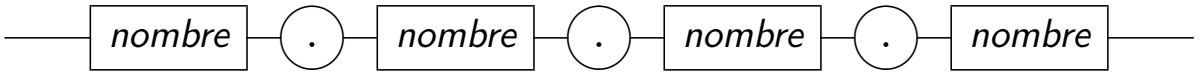
Association



IdentificateurQualifie



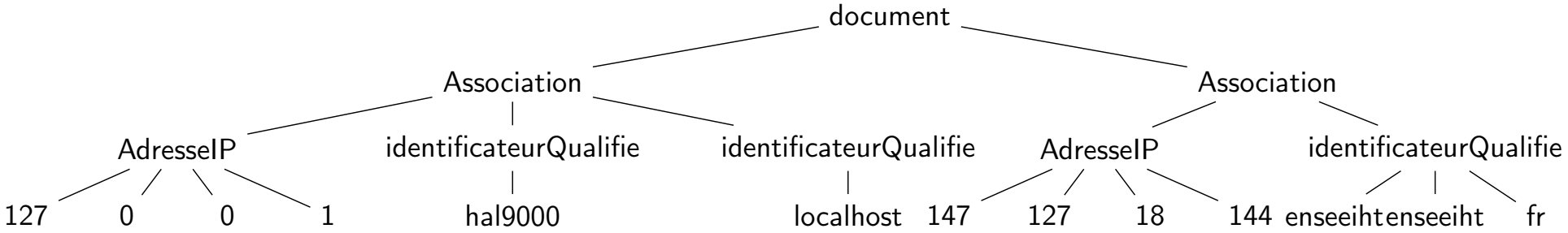
AdresseIP



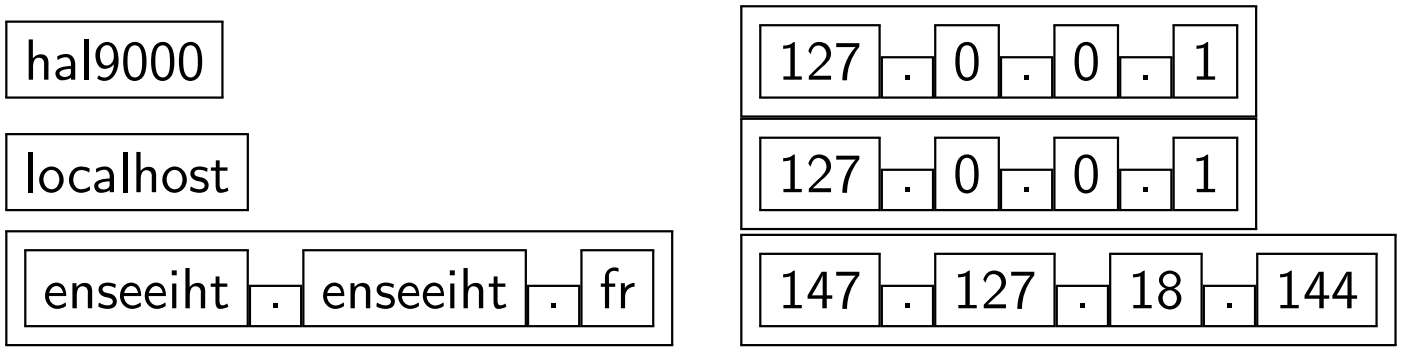
Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

Analyse sémantique

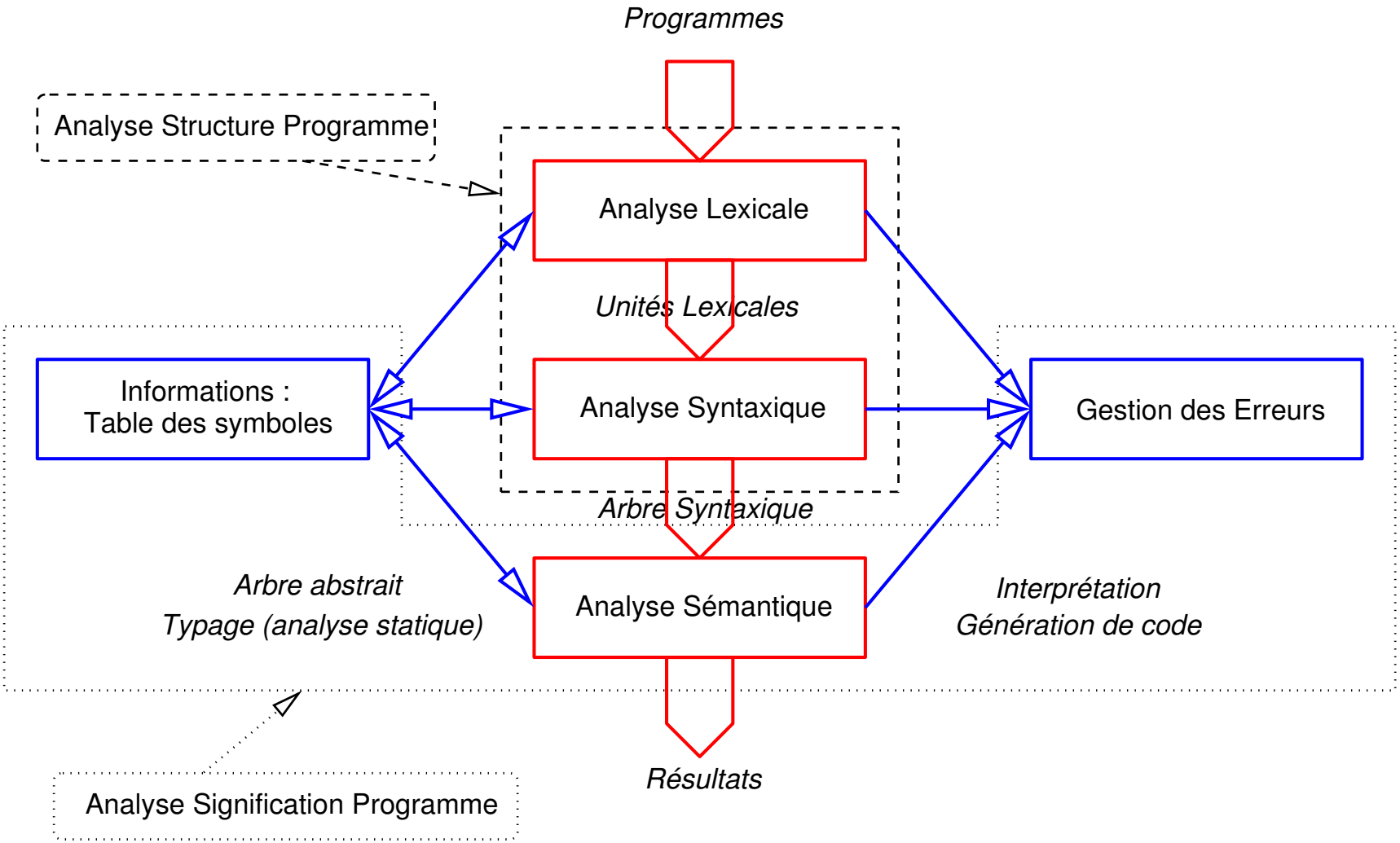
- Structure arborescente associée :



- Exploitation des informations : association nom qualifié/adresse IP (unités sémantiques)



Structure d'un outil



Définitions

- Caractère/Symbole : Unité élémentaire d'information
- Unité lexicale (lexème, mot) : Séquence de caractères
- Unité syntaxique (arbre syntaxique, syntème, phrase) : Arbre d'unités lexicales
- Unité sémantique : diverses (arbre abstrait, table des symboles, type, code généré, résultat évaluation, . . .)

Comment organiser les informations ?

- Objectif : Exploitation des informations
- Règle : Choisir le bon niveau de précision
- Unité lexicale : Bloc élémentaire d'information pertinente
- Unité syntaxique : Elément structurant de l'information

Approche XML pour les informations structurées

- Signification : eXtensible Markup Langage (langage à balises évolutif)
- Objectif : format universel pour décrire des documents arborescents structurés
- Evolutif : définition de nouveaux éléments (DTD, XML Schemas)

Points importants

- Document bien formé : Structure arborescente
- Document valide : Respect arborescence particulière (famille de documents – DTD, XML Schemas, types de documents)
- Outil de description de données portables
- Approches :
  - Donnée : structure rigoureuse
  - Littéraire : structure partielle/libre
- Structure sémantique et pas figurative ($\neq$ HTML)
- Important :
  - Ni un langage de programmation (peut représenter arbre abstrait)
  - Ni un protocole de transport (utilise HTTP)
  - Ni une base de donnée

Historique

- Années 70 : Systèmes de Gestion de Données Techniques (SGDT)
- IBM Generalized Markup Langage (GML) normalisé en 1986 (SGML)
- Application : Famille de données similaires représentée par une DTD
- Exemple : HTML (HyperText Markup Language), description pages WEB
- exclusivement figuratif (présentation sans sémantique) extension impose changement norme
- Solution : XML = SGML «allégé» proposé et standardisé par W3C

Outils et Applications

- Outils d'analyse :
  - SAX : Sample Api for Xml parsing
  - DOM, DOM4J, JDOM : Document Object Model
- Format autre qu'XML :
  - XPath : filtrage de documents XML (extraction de parties)
  - CSS (Cascading Style Sheets) : feuilles de style hiérarchiques
  - DTD : Document Type Definition

Outils et Applications (bis)

- Format XML :
  - XFragment : document composé de plusieurs autres documents
  - XMLQuery : filtrage de documents
  - XLinks : liens entre documents XML et non XML
  - XPointers : liens au sein d'un document
  - NameSpace : organisation modulaire
  - XHTML : Version structurée de HTML
  - XSLT (eXtensible Stylesheet Language-Transformation) : transformations de documents XML
  - XSL-FO (XSL-Formating Objects) : aspect figuratif
  - XML Schemas : Description structure documents
  - SOAP (Sample Object Access Protocol), XML-RPC
  - SVG (Scalable Vector Graphics), RDF (Ressource Description Framework), SMIL (synchronisation, multimédia), MathML (formule mathématiques), ...

Format d'un document XML

- Contrainte : Format textuel
- Balise : début `<ident>` fin `</ident>`
- Elément vide : `<ident />` ≡ `<ident></ident>`
- Identificateur : lettre, chiffre, _, -, ., : (espace nommage)
- Attribut : `ident = "chaîne"`
- Référence entité : `&ident;`
- Exemple caractères spéciaux :

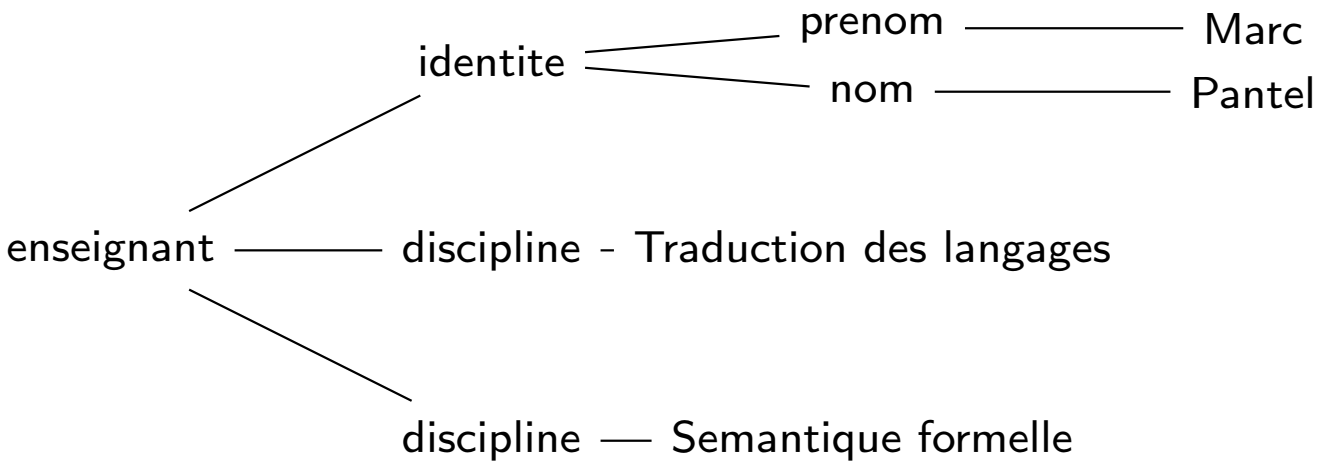
<	>	&	"	'	...
`<`	`>`	`&`	`&quot;`	`&apos;`	...

Exemple de document XML sans attributs

- Document :

```
<enseignant>
      <identite>
            <prenom>Marc</prenom>
            <nom>Pantel</nom>
      </identite>
      <discipline>Traduction des langages</discipline>
      <discipline>Sémantique formelle</discipline>
</enseignant>
```

- Structure arborescente associée :

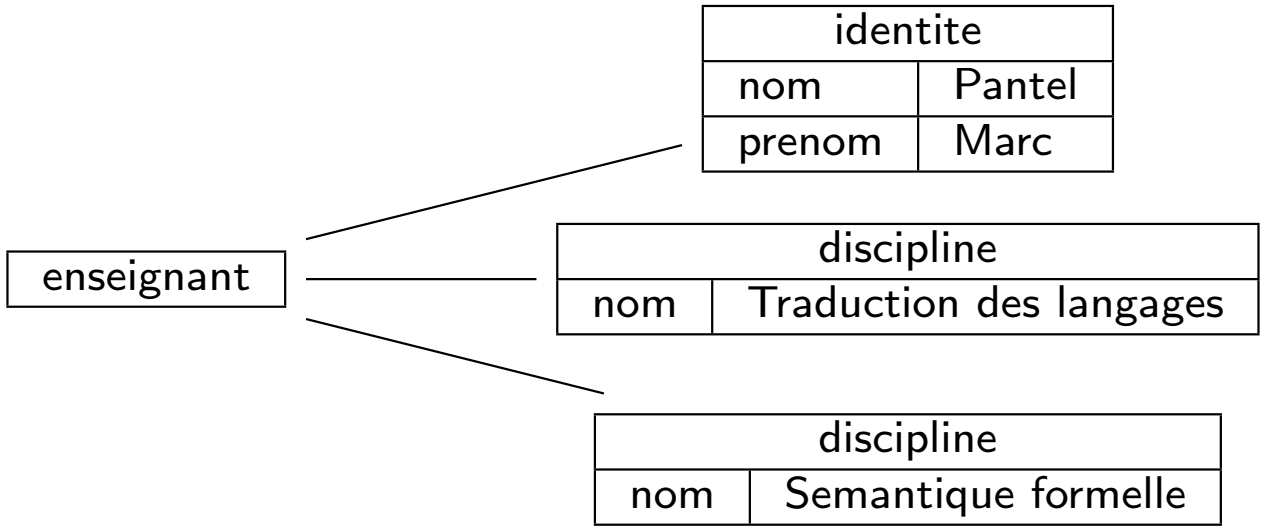


Exemple de document XML avec attributs

- Document :

```
<enseignant>
     <identite prenom="Marc" nom="Pantel"/>
     <discipline nom="Traduction des langages"/>
     <discipline nom="Sémantique formelle"/>
</enseignant>
```

- Structure arborescente associée :



Contenu du document

- Texte brut : `<![CDATA[texte]]>`
- Commentaires : `<!- texte ->`
- Ordre applicatif : `<? cible texte ?>`
- Exemple : `<?php ...?>`
- Déclaration : ordre applicatif XML

```
<?xml version="..." encoding="..." standalone="..." ?>
          1.0      défaut UTF8     yes ou no
```

Document bien formé

- Chaque balise de début a une balise de fin
- Pas de recouvrement d'éléments distincts
- Une seule racine
- Valeurs des attributs entre `"..."`
- Unicité des attributs par éléments
- Pas de commentaires dans les balises
- Pas de caractères <, >, &, ", ' dans le texte

Représentation en JavaScript des informations structurées

- Signification : JavaScript Object Notation (représentation des objets JavaScript)
- Objectif : format texte pour échanger des données entre client et serveur en JavaScript
- Différence majeure avec XML : Pas de familles de documents (DTD/Schema – Typage de documents)

Exemple de document JSON

- Document JSON :

```
{
   "prenom" : "Marc",
   "nom" : "Pantel",
   "naissance" : {
      "jour" : 4,
      "mois" : "mai",
      "annee" : 1966
   },
   "matiere" : [ "Modelisation", "Genie Logiciel", "Semantique" ]
}
```

- Struture arborescente associée :