Contraindre le XML : présentation de la DTD

Pourquoi Contraindre la XML-TEI avec une DTD?

Lorsque nous travaillons avec des documents XML, comme ceux codés en TEI (Text Encoding Initiative), il est important de s'assurer que les documents sont non seulement bien formés, mais aussi valides. La raison pour laquelle nous "contraignons" un document XML-TEI avec une DTD (Document Type Definition) ou tout autre schéma de validation (comme Relax NG ou XML Schema) est multiple :

- **Assurer la Cohérence :** Dans un projet collaboratif ou un corpus de documents, il est important que tous les documents respectent la même structure et les mêmes règles. Cela garantit que le balisage est cohérent à travers tous les fichiers, facilitant ainsi l'interopérabilité et la réutilisation des documents.
- **Éviter les Erreurs :** En imposant des contraintes via une DTD, on peut détecter et prévenir les erreurs lors de la création ou de la modification des documents. Par exemple, si une balise obligatoire est omise ou mal utilisée, le document sera signalé comme invalide.
- **Faciliter le Traitement Automatisé :** Les outils qui traitent les documents XML (comme les parsers, les transformateurs XSLT, etc.) s'attendent souvent à des structures spécifiques. La validation contre une DTD garantit que ces outils peuvent traiter les documents de manière fiable.

Un document valide

Un document XML-TEI qui respecte un schéma de validation comme une DTD est dit "valide", ce n'est pas la même chose qu'un document dit "bien formé".

Document Bien Formé :
Un document XML est considéré comme "bien formé" lorsqu'il respecte les règles syntaxiques de base de XML, indépendamment de la DTD ou de tout autre schéma de validation.

- Exigences de base :
  - 1 balise ouvrante = 1 balise fermante
  - respect de l'imbrication
  - un attribut du même nom par balise
  - un seul élément racine

Exemple d'un document bien formé:

```
<book>
  <title>Les Misérables</title>
  <author ref="https://fr.wikipedia.org/wiki/Victor_Hugo">Victor
Hugo</author>
</book>
```

Explication : Ce document est bien formé car toutes les balises sont correctement ouvertes et fermées, le contenu de l'attribut est entouré de guillements et il y a un seul élément racine (`<book>`).

Document Valide :
Un document XML est "valide" lorsqu'il est non seulement bien formé, mais qu'il respecte aussi les contraintes définies par une DTD ou un autre schéma de validation.

- Validation avec une DTD : La DTD définit quelles balises sont permises, comment elles peuvent être imbriquées, et quels attributs sont autorisés ou obligatoires. Si le document respecte toutes ces règles, il est considéré comme valide.

Le Document Type Definition (DTD)

Une DTD se présente sous la forme d'une **liste de déclaration** qui définissent la structure que devront suivre les documents qui s'y réfèrent.
La DTD qui sert de modèle à un document doit être déclarée au début de celui-ci, **avant l'ouverture de l'élément racine**.

Déclaration de la DTD

La déclaration de la DTD peut être faite au **début du document** XML dont elle contraint l'encodage, ou dans un **fichier externe** dont il est fait référence dans le document XML

Déclaration de la DTD Interne
Une DTD interne peut être définie directement à l'intérieur du document XML. Toutes les règles de la DTD sont écrites au début du document, entre les balises `<!DOCTYPE>` et `]>`.

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE book [
  <!ELEMENT book (title, author)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ATTLIST author ref CDATA #IMPLIED>
]>

<book>
  <title>Les Misérables</title>
  <author ref="https://fr.wikipedia.org/wiki/Victor_Hugo">Victor
Hugo</author>
</book>
```

Explication :

- La DTD interne est définie entre `<!DOCTYPE book [` et `]>`.
- Les éléments `<book>`, `<title>`, et `<author>` sont définis, ainsi que l'attribut `ref` pour l'élément `<author>`.
- Cette DTD interne s'applique uniquement à ce document XML.

Déclaration de la DTD Externe
Une DTD externe est stockée dans un fichier séparé. Le document XML fait référence à ce fichier en utilisant une déclaration `DOCTYPE` qui pointe vers le fichier DTD.

Exemple d'une DTD externe :

Fichier `book.dtd` :

```
<!ELEMENT book (title, author)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ATTLIST author ref CDATA #IMPLIED>
```

Document XML faisant référence à la DTD externe :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "book.dtd">

<book>
  <title>Les Misérables</title>
  <author ref="https://fr.wikipedia.org/wiki/Victor_Hugo">Victor
Hugo</author>
</book>
```

Explication :

- La DTD est stockée dans un fichier séparé appelé `book.dtd`.
- Le document XML fait référence à cette DTD externe via la déclaration `<!DOCTYPE book SYSTEM "book.dtd">`.
- Cette approche est pratique lorsque plusieurs documents XML doivent se conformer à la même DTD, car cela permet de centraliser les règles de validation.

Avantages et Inconvénients

- DTD Interne :
  - *Avantage* : Tout est contenu dans un seul fichier, ce qui peut être pratique pour de petits documents ou pour des tests rapides.
  - *Inconvénient* : La DTD ne peut pas être réutilisée par d'autres documents.
- DTD Externe :
  - *Avantage* : La DTD peut être réutilisée par plusieurs documents XML, ce qui facilite la maintenance et assure la cohérence entre les documents.
  - *Inconvénient* : Nécessite un fichier supplémentaire

Contenu d'une DTD

Une DTD contient des déclarations concernant :

- **Éléments**: Définit les éléments qui peuvent être utilisés, leur nombre, leur imbrication, leur ordre, etc.
- **Attribus**: Définit les attributs autorisés pour un élément précis, leur valeurs et le type de valeur autorisé
- **Entités**: Définit les entités qui pourront être utilisées dans le document XML

L'<élément> dans la DTD

1. Définir un Élément dans une DTD

La définition d'un élément dans une DTD se fait avec la déclaration `<!ELEMENT>`. Cette déclaration permet de spécifier le nom de l'élément, son type de contenu, ainsi que les règles de validation qui s'appliquent à cet élément.

Syntaxe Générale :

```
<!ELEMENT nom_de_l_element type_de_contenu>
```

Exemple :

```
<!ELEMENT title (#PCDATA)>
```

Dans cet exemple, l'élément `<title>` est défini pour contenir du texte brut (#PCDATA signifie "Parsed Character Data", c'est-à-dire du texte qui sera interprété par le parseur XML).

Donc par exemple dans le document XML-TEI :

```
<title>Les Misérables</title>
```

2. Types de contenu des éléments

Expression	Signification
`EMPTY`	Contenu vide
`ANY`	Contenu quelconque
`(#PCDATA)`	Contenu textuel
`(elt)`	Un seul élément
`(elt1, elt2, ..., eltn)`	Séquence d'éléments pris dans cet ordre
`(elt1 | elt2 | ... | eltn)`	Un des éléments au choix

1. `EMPTY` - Contenu Vide

Signification: Un élément défini comme `EMPTY` ne peut contenir aucun contenu textuel ni sous-élément. Il est utilisé pour des éléments qui ne portent que des attributs ou servent de marqueurs sans contenu.

Exemple DTD:

```
<!ELEMENT linebreak EMPTY>
```

Exemple XML:

```
<linebreak/>
```

Ici, l'élément `<linebreak>` est défini comme vide, donc il ne contient aucun texte ou autre élément. Il est souvent utilisé pour les éléments de structure comme les sauts de ligne ou les balises de séparation.

Bien sûr ! Remplaçons la balise `<wildcard>` par une balise plus couramment utilisée, comme `<note>`, qui est souvent utilisée pour ajouter des annotations ou des commentaires dans un texte.

2. `ANY` - Contenu Quelconque

Signification: Un élément défini comme `ANY` peut contenir n'importe quel contenu, que ce soit du texte, d'autres éléments, ou être vide. Cela donne une grande flexibilité mais est rarement utilisé en pratique en raison de sa permissivité.

Exemple DTD:

```
<!ELEMENT note ANY>
```

Exemple XML:

```
<note>Cette note contient simplement du texte.</note>
<note>
  <paragraph>Cette note contient un paragraphe structuré.</paragraph>
</note>
<note/>
```

Dans cet exemple :

- La première balise `<note>` contient du texte simple : "Cette note contient simplement du texte."
- La seconde balise `<note>` contient un sous-élément `<paragraph>` qui structure le texte à l'intérieur.
- La troisième balise `<note>` est vide, ce qui est également permis par la définition `ANY`.

Cela montre que la balise `<note>`, définie comme `ANY`, peut contenir une variété de contenus, ce qui peut être pratique mais peut aussi compliquer la validation et la cohérence du document.

3. `#PCDATA` - Contenu Textuel

Signification: `#PCDATA` (Parsed Character Data) indique que l'élément peut contenir du texte brut. Il ne permet pas l'inclusion d'éléments enfants mais peut contenir du texte.

Exemple DTD:

```
<!ELEMENT title (#PCDATA)>
```

Exemple XML:

```
<title>Introduction to XML</title>
```

L'élément `<title>` est défini pour contenir du texte uniquement, sans sous-éléments. Cela est typique pour des éléments tels que les titres, les noms, etc.

4. `(elt)` - Un Seul Élément

Signification: Cette expression indique que l'élément doit contenir exactement un sous-élément spécifié.

Exemple DTD:

```
<!ELEMENT book (title)>
```

Exemple XML:

```
<book>
  <title>Learning XML</title>
</book>
```

Dans cet exemple, l'élément `<book>` doit contenir exactement un `<title>`. Toute autre combinaison de contenu ou absence de `<title>` entraînerait une invalidité du document XML.

5. `(elt1, elt2, ..., eltn)` - Séquence d'Éléments

Signification: Cette expression spécifie une séquence exacte d'éléments qui doivent apparaître dans l'ordre indiqué.

Exemple DTD:

```
<!ELEMENT book (title, author)>
```

Exemple XML:

```
<book>
  <title>XML Basics</title>
  <author>John Doe</author>
</book>
```

Dans cet exemple, l'élément `<book>` doit contenir un `<title>` suivi directement d'un `<author>`. L'ordre des éléments est important et doit être respecté.

6. `(elt1 | elt2 | ... | eltn)` - Un Des Éléments au Choix

Signification: Cette expression permet à l'élément de contenir un seul parmi une liste d'éléments définis. Cela signifie que l'élément doit choisir un des éléments listés mais pas tous.

Exemple DTD:

```
<!ELEMENT content (paragraph | image)>
```

Exemple XML:

```
<content>
  <paragraph>This is a paragraph.</paragraph>
</content>
```

Ou

```
<content>
  <image src="image.jpg"/>
</content>
```

Ici, l'élément `<content>` peut contenir soit un `<paragraph>`, soit un `<image>`, mais pas les deux simultanément. Cela permet une certaine flexibilité tout en imposant une restriction sur le contenu.

Conclusion

En utilisant ces expressions dans une DTD, vous pouvez définir des structures XML complexes avec des règles claires sur ce que chaque élément peut contenir. Les types de contenu comme `EMPTY`, `ANY`, `#PCDATA`, ainsi que les opérateurs d'occurrence comme les séquences et les choix, vous permettent de créer des définitions de structure qui garantissent la validité et la cohérence des documents XML.

Exercice

Lier un document XML à une DTD

Au sein d'un même dossier, essayer ça :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE body SYSTEM "schema.dtd">

<body>
    <text>blabla</text>
</body>
```

```
<!ELEMENT body (text)>
<!ELEMENT text (#PCDATA)>
```

Puis essayer un exemple avec chaque type de contenus.

3. Les opérateurs d'occurrence

Les opérateurs d'occurrence en DTD définissent le nombre de fois qu'un élément peut apparaître à l'intérieur d'un autre élément.

Opérateur	Signification
?	0 ou 1 occurrence
*	0 ou plusieurs occurrences
+	1 ou plusieurs occurrences
(rien)	1 seule occurrence

Voici des exemples simples pour chaque opérateur d'occurrence en DTD :

1. ? - 0 ou 1 Occurrence

DTD:

```
<!ELEMENT person (name, nickname?)>
```

XML Valide:

```xml
<person>
  <name>John Doe</name>
  <nickname>Johnny</nickname>
</person>
```

XML Valide (sans `nickname`):

```
<person>
  <name>John Doe</name>
</person>
```

2. `*` - 0 ou Plusieurs Occurrences

DTD:

```
<!ELEMENT shoppingList (item*)>
```

XML Valide (avec plusieurs `item`):

```
<shoppingList>
  <item>Bread</item>
  <item>Milk</item>
</shoppingList>
```

XML Valide (sans `item`):

```
<shoppingList/>
```

3. `+` - 1 ou Plusieurs Occurrences

DTD:

```
<!ELEMENT paragraph (sentence+)>
```

XML Valide:

```
<paragraph>
  <sentence>This is the first sentence.</sentence>
  <sentence>This is the second sentence.</sentence>
</paragraph>
```

XML Invalide (pas de `sentence`):

```
<paragraph/>
```

4. `(rien)` - 1 Seule Occurrence

DTD:

```
<!ELEMENT book (title, author)>
```

XML Valide:

```
<book>
  <title>Les Misérables</title>
  <author>Victor Hugo</author>
</book>
```

XML Invalide (manque l'élément `author`):

```
<book>
  <title>Les Misérables</title>
</book>
```

Ces exemples montrent comment les opérateurs d'occurrence définissent les règles sur la fréquence d'apparition des éléments dans un document XML.

Exercice

Exemples avec des opérateurs d'occurence

4. Exercice : Rédiger des déclarations d'éléments

Poésie

Rédiger les déclarations d'éléments pour décrire l'encodage du poème *Mon rêve familier* de Verlaine.

```
<?xml version="1.0" encoding="UTF-8"?>
<poeme>
    <titre>Mon rêve familier</titre>
    <auteur>Paul Verlaine</auteur>
    <strophe>
        <vers>Je fais souvent ce rêve étrange et pénétrant</vers>
        <vers>D'une femme inconnue, et que j'aime, et qui m'aime</vers>
        <vers>Et qui n'est, chaque fois, ni tout à fait la même</vers>
        <vers>Ni tout à fait une autre, et m'aime et me comprend.</vers>
    </strophe>
    <strophe>
```

```
        <vers>Car elle me comprend, et mon coeur, transparent</vers>
        <vers>Pour elle seule, hélas ! cesse d'être un problème</vers>
        <vers>Pour elle seule, et les moiteurs de mon front blême,</vers>
        <vers>Elle seule les sait rafraîchir, en pleurant.</vers>
    </strophe>
    <strophe>
        <tercet>
            <vers>Est-elle brune, blonde ou rousse ? - Je l'ignore.</vers>
            <vers>Son nom ? Je me souviens qu'il est doux et sonore</vers>
            <vers>Comme ceux des aimés que la Vie exila.</vers>
        </tercet>
        <tercet>
            <vers>Son regard est pareil au regard des statues,</vers>
            <vers>Et, pour sa voix, lointaine, et calme, et grave, elle a
</vers>
            <vers>L'inflexion des voix chères qui se sont tues.</vers>
        </tercet>
    </strophe>
</poeme>
```

La DTD proposée :

```
<!DOCTYPE poeme [
      <!ELEMENT poeme (titre, auteur, strophe*)>
      <!ELEMENT titre (#PCDATA)>
      <!ELEMENT auteur (#PCDATA)>
      <!ELEMENT strophe ((vers, vers, vers, vers) | (tercet, tercet))>
      <!ELEMENT tercet (vers, vers, vers)>
      <!ELEMENT vers (#PCDATA)>
]>
```

Théâtre

Pour *La Malade Imaginaire* de Molière.

```
<?xml version="1.0" encoding="UTF-8"?>
<piece>
    <titre>Le Malade Imaginaire</titre>
    <auteur>Molière</auteur>

    <acte>
        <titre>Acte I</titre>
        <scene>
            <titre>Scène 1</titre>
            <replique>
                <personnage>Argan</personnage>
                <texte>Hé bien, Monsieur Purgon, que me conseillez-vous de
faire ?</texte>
            </replique>
```

```
            <didascalie>(Il se tourne vers son médecin avec anxiété.)
</didascalie>
            <replique>
                <personnage>Monsieur Purgon</personnage>
                <texte>Je vous conseille de continuer votre régime
strictement.</texte>
            </replique>
        </scene>
        <scene>
            <titre>Scène 2</titre>
            <replique>
                <personnage>Toinette</personnage>
                <texte>Ah, Monsieur Argan, vous ne vous portez que trop
bien !</texte>
            </replique>
            <didascalie>(Toinette, en aparté, fait signe qu'elle
plaisante.)</didascalie>
        </scene>
    </acte>

    <acte>
        <titre>Acte II</titre>
        <scene>
            <titre>Scène 1</titre>
            <replique>
                <personnage>Argan</personnage>
                <texte>Mon coeur, vous êtes cruelle !</texte>
            </replique>
            <didascalie>(Il pose la main sur sa poitrine, feignant un
malaise.)</didascalie>
            <replique>
                <personnage>Angélique</personnage>
                <texte>Père, je n'ai que de l'amour pour vous.</texte>
            </replique>
        </scene>
    </acte>
</piece>
```

La DTD proposée :

```
<!DOCTYPE piece [
    <!ELEMENT piece (titre, auteur, acte+)>
    <!ELEMENT titre (#PCDATA)>
    <!ELEMENT auteur (#PCDATA)>
    <!ELEMENT acte (titre, scene+)>
    <!ELEMENT scene (titre, (replique | didascalie)+)>
    <!ELEMENT replique (personnage, texte)>
    <!ELEMENT personnage (#PCDATA)>
    <!ELEMENT texte (#PCDATA)>
    <!ELEMENT didascalie (#PCDATA)>
]>
```

L'attribut dans la DTD

1. Définir un Attribut dans une DTD

La définition d'un attribut dans une DTD se fait avec la déclaration `<!ATTLIST>`. Cette déclaration permet de spécifier le nom de l'élément auquel l'attribut est associé, le nom de l'attribut, son type, et une valeur par défaut ou une contrainte.

Syntaxe Générale :

```
<!ATTLIST nom_de_l_element nom_de_l_attribut type_de_l_attribut
valeur_par_défaut_ou_contrainte>
```

Exemple :

```
<!ELEMENT title (#PCDATA)>
<!ATTLIST title lang CDATA #IMPLIED>
```

Dans cet exemple, l'élément `<title>` peut avoir un attribut `lang` qui spécifie la langue du titre. `CDATA` indique que la valeur de l'attribut est du texte brut, et `#IMPLIED` signifie que cet attribut est optionnel.

Exemple de Document XML-TEI :

```
<title lang="fr">Les Misérables</title>
```

Ici, l'élément `<title>` inclut un attribut `lang` avec la valeur `"fr"`, indiquant que le titre est en français. L'attribut `lang` n'est pas obligatoire, mais lorsqu'il est utilisé, il ajoute une information supplémentaire sur la langue du contenu textuel.

2. Type de contenu des attributs

Expression	Signification
CDATA	chaîne de caractères ne comprenant pas de balises
(val1 | val2 | ...)	liste de valeurs à utiliser
ENTITY / ENTITIES	entité déclarée dans la DTD (ou liste séparée par des espaces)
ID	pour identifier l'élément
IDREF / IDREFS	ID d'un autre élément (ou liste séparée par des espaces)

1. `CDATA` - Chaîne de caractères ne comprenant pas de balises

DTD:

```
<!ELEMENT note (#PCDATA)>
<!ATTLIST note content CDATA #REQUIRED>
```

XML Valide:

```
<note content="This is a simple note.">Take note of this information.
</note>
```

Dans cet exemple, l'attribut `content` de l'élément `<note>` est une chaîne de caractères simple.

2. `(val1 | val2 | ...)` - Liste de valeurs à utiliser

DTD:

```
<!ELEMENT book (#PCDATA)>
<!ATTLIST book genre (fiction | non-fiction | poetry) "fiction">
```

XML Valide:

```
<book genre="poetry">The Road Not Taken</book>
```

Dans cet exemple, l'attribut `genre` de l'élément `<book>` doit être l'une des valeurs spécifiées : `fiction`, `non-fiction`, ou `poetry`.

3. `ENTITY / ENTITIES` - Entité déclarée dans la DTD (ou liste séparée par des espaces)

DTD:

```
<!ENTITY author "Victor Hugo">
<!ELEMENT book (#PCDATA)>
<!ATTLIST book author ENTITY #IMPLIED>
```

XML Valide:

```
<book author="author">Les Misérables</book>
```

Dans cet exemple, l'attribut `author` de l'élément `<book>` doit correspondre à une entité déclarée dans la DTD, ici `"author"` qui correspond à `"Victor Hugo"`.

4. `ID` - Pour identifier l'élément

DTD:

```
<!ELEMENT chapter (#PCDATA)>
<!ATTLIST chapter id ID #REQUIRED>
```

XML Valide:

```
<chapter id="chap1">Introduction</chapter>
```

Dans cet exemple, l'attribut `id` de l'élément `<chapter>` doit être unique dans le document et sert à identifier cet élément.

5. `IDREF / IDREFS` - ID d'un autre élément (ou liste séparée par des espaces)

DTD:

```
<!ELEMENT book (chapter)>
<!ELEMENT chapter (#PCDATA)>
<!ATTLIST book id ID #REQUIRED>
<!ATTLIST chapter bookRef IDREF #REQUIRED>
```

XML Valide:

```
<book id="book1">
    <chapter bookRef="book1">Les Misérables</chapter>
</book>
```

Dans cet exemple, l'attribut `bookRef` de l'élément `<chapter>` doit faire référence à l'ID d'un autre élément `<book>` dans le document, ici `"book1"`.

3. Type d'attributs

Opérateur	Signification
`#REQUIRED`	valeur requise dans l'élément
`#IMPLIED`	valeur facultative
`#FIXED "valeur"`	valeur fixe pour l'attribut
`"valeur"`	valeur par défaut de l'attribut (on peut la remplacer)

1. `#REQUIRED` - Valeur requise dans l'élément

DTD:

```
<!ELEMENT book (#PCDATA)>
<!ATTLIST book author CDATA #REQUIRED>
```

XML Valide:

```
<book author="Victor Hugo">Les misérables</book>
```

Dans cet exemple, l'attribut `author` est obligatoire pour l'élément `<book>`. Si cet attribut est omis, le document ne sera pas valide.

2. `#IMPLIED` - Valeur facultative

DTD:

```
<!ELEMENT book (#PCDATA)>
<!ATTLIST book author CDATA #IMPLIED>
```

XML Valide:

```
<book author="Victor Hugo">Les misérables</book>
```

Ou sans l'attribut, toujours valide :

```
<book>Les misérables</book>
```

Dans cet exemple, l'attribut `title` est facultatif. Il peut être présent ou non sans affecter la validité du document.

3. `#FIXED "valeur"` - Valeur fixe pour l'attribut

DTD:

```
<!ELEMENT book (#PCDATA)>
<!ATTLIST book lang CDATA #FIXED "fr">
```

XML Valide:

```
<book lang="fr">Les Misérables</book>
```

Dans cet exemple, l'attribut `lang` doit toujours avoir la valeur `"fr"`. Toute autre valeur rendra le document non valide.

4. `"valeur"` - Valeur par défaut de l'attribut (remplaçable)

DTD:

```
<!ELEMENT book (#PCDATA)>
<!ATTLIST book lang CDATA "fr">
```

XML Valide avec valeur par défaut :

```
<book>Les Misérables</book>
```

Ou en remplaçant la valeur par défaut :

```
<book lang="en">Les Misérables</book>
```

Dans cet exemple, l'attribut `lang` a par défaut la valeur `"fr"`, mais cette valeur peut être remplacée par une autre, comme `"en"`.

4. Exercice : Rédiger des déclarations d'attributs

Poésie Rédiger les déclarations pour décrire l'encodage du poème *Mon rêve familier* de Verlaine

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE poeme SYSTEM "Mon_reve_familier.dtd">

<poeme type="sonnet">
    <titre>Mon rêve familier</titre>
    <auteur>Paul Verlaine</auteur>
    <strophe type="quatrain" n="1">
        <vers>Je fais souvent ce rêve étrange et pénétrant</vers>
        <vers>D'une femme inconnue, et que j'aime, et qui m'aime</vers>
        <vers>Et qui n'est, chaque fois, ni tout à fait la même</vers>
        <vers>Ni tout à fait une autre, et m'aime et me comprend.</vers>
    </strophe>
    <strophe type="quatrain" n="2">
        <vers>Car elle me comprend, et mon coeur, transparent</vers>
        <vers>Pour elle seule, hélas ! cesse d'être un problème</vers>
        <vers>Pour elle seule, et les moiteurs de mon front blême,</vers>
```

```
        <vers>Elle seule les sait rafraîchir, en pleurant.</vers>
    </strophe>
    <strophe type="sizain">
        <tercet n="1">
            <vers>Est-elle brune, blonde ou rousse ? - Je l'ignore.</vers>
            <vers>Son nom ? Je me souviens qu'il est doux et sonore</vers>
            <vers>Comme ceux des aimés que la Vie exila.</vers>
        </tercet>
        <tercet n="2">
            <vers>Son regard est pareil au regard des statues,</vers>
            <vers>Et, pour sa voix, lointaine, et calme, et grave, elle a
</vers>
            <vers>L'inflexion des voix chères qui se sont tues.</vers>
        </tercet>
    </strophe>
</poeme>
```

La DTD attendue :

```
<!DOCTYPE poeme [
    <!ELEMENT poeme (titre, auteur, strophe+)>
    <!ELEMENT titre (#PCDATA)>
    <!ELEMENT auteur (#PCDATA)>
    <!ELEMENT strophe ((vers, vers, vers, vers) | (tercet, tercet))>
    <!ELEMENT vers (#PCDATA)>
    <!ELEMENT tercet (#PCDATA | vers)*>

    <!ATTLIST poeme type CDATA #FIXED "sonnet">
    <!ATTLIST strophe type (quatrain | sizain) #REQUIRED>
    <!ATTLIST strophe n (1 | 2) #IMPLIED>
    <!ATTLIST tercet n (1 | 2) #REQUIRED>
]>
```

Théâtre Rédiger les déclarations pour décrire l'encodage de la pièce *Le Malade Imaginaire* de Molière

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE piece SYSTEM "theatre.dtd">

<piece>
    <titre>Le Malade Imaginaire</titre>
    <auteur>Molière</auteur>

    <acte numero="1">
        <titre>Acte I</titre>
        <scene numero="1">
            <titre>Scène 1</titre>
            <replique personnageID="p1">
                <personnage id="p1">Argan</personnage>
                <texte>Hé bien, Monsieur Purgon, que me conseillez-vous de
```

```
faire ?</texte>
            </replique>
            <didascalie>(Il se tourne vers son médecin avec anxiété.)
</didascalie>
            <replique personnageID="p2">
                <personnage id="p2">Monsieur Purgon</personnage>
                <texte>Je vous conseille de continuer votre régime
strictement.</texte>
            </replique>
        </scene>
        <scene numero="2">
            <titre>Scène 2</titre>
            <replique personnageID="p3">
                <personnage id="p3">Toinette</personnage>
                <texte>Ah, Monsieur Argan, vous ne vous portez que trop
bien !</texte>
            </replique>
            <didascalie>(Toinette, en aparté, fait signe qu'elle
plaisante.)</didascalie>
        </scene>
    </acte>

    <acte numero="2">
        <titre>Acte II</titre>
        <scene numero="1">
            <titre>Scène 1</titre>
            <replique personnageID="p1">
                <personnage id="p1">Argan</personnage>
                <texte>Mon coeur, vous êtes cruelle !</texte>
            </replique>
            <didascalie>(Il pose la main sur sa poitrine, feignant un
malaise.)</didascalie>
            <replique personnageID="p4">
                <personnage id="p4">Angélique</personnage>
                <texte>Père, je n'ai que de l'amour pour vous.</texte>
            </replique>
        </scene>
    </acte>
</piece>
```

La DTD attendue :

```
<!ELEMENT piece (titre, auteur, acte+)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT acte (titre, scene+)>
<!ATTLIST acte
    numero CDATA #REQUIRED>
<!ELEMENT scene (titre, (replique | didascalie)+)>
<!ATTLIST scene
    numero CDATA #REQUIRED>
<!ELEMENT replique (personnage, texte)>
```

```
<!ATTLIST replique
    personnageID IDREF #REQUIRED>
<!ELEMENT personnage (#PCDATA)>
<!ATTLIST personnage
    id ID #REQUIRED>
<!ELEMENT texte (#PCDATA)>
<!ELEMENT didascalie (#PCDATA)>
```

L'entité dans la DTD

1. Définir une Entité dans une DTD

Les entités permettent de définir des fragments de texte réutilisables ou des caractères spéciaux dans une DTD. Une entité peut être déclarée à l'aide de `<!ENTITY>`, et elle peut ensuite être utilisée dans les attributs ou le contenu du document XML.

Syntaxe Générale :

```
<!ENTITY nom_de_l_entité "définition_de_l_entité">
```

Exemple :

```
<!ENTITY author "Victor Hugo">
<!ELEMENT book (#PCDATA)>
<!ATTLIST book author ENTITY #REQUIRED>
```

Dans cet exemple, l'entité `author` est définie comme `"Victor Hugo"`. L'élément `<book>` a un attribut `author` qui doit utiliser cette entité. Le type `ENTITY` indique que l'attribut doit référencer une entité définie dans la DTD.

Exemple de Document XML-TEI :

```
<!DOCTYPE book [
  <!ENTITY author "Victor Hugo">
]>
<book author="author">Les Misérables</book>
```

Dans cet exemple, l'élément `<book>` utilise l'attribut `author` pour référencer l'entité `author` définie dans la DTD. Le texte de l'attribut `author` est remplacé par `"Victor Hugo"` à la validation, car l'entité `author` est définie pour cette valeur.

Voici comment cela fonctionne en pratique :

- **Déclaration de l'entité** : Dans la DTD, vous déclarez l'entité `author` avec une valeur spécifique.

- **Utilisation de l'entité** : Dans le document XML, vous utilisez le nom de l'entité comme valeur d'attribut. Lors de la validation ou de l'affichage, cette valeur est remplacée par le texte défini dans la DTD.

Ce mécanisme permet de maintenir la cohérence dans le texte et d'utiliser des valeurs standardisées dans plusieurs parties du document.

2. Utiliser des Entités prédéfinies

Il existe des entités déjà prédéfinies :

Entité	Caractère
<	<
>	>
'	'
"	"
&	&

Exemple avec l'esperluette :

```
<text>Il est important de comprendre les signes < et > dans un document
XML. Utilisez &amp; pour représenter le caractère &.</text>
```

Dans cet exemple, `&` est utilisé pour représenter le caractère `&` dans le texte XML. Les entités prédéfinies permettent d'inclure des caractères spéciaux dans le texte sans provoquer des erreurs de parsing ou d'interprétation.

3. Définir ses Entités

Il est possible de définir ses propres entités.

DTD :

```
<!ENTITY uvsq "Université de Versailles Saint-Quentin-en-Yvelines">
<!ELEMENT p (#PCDATA)>
```

Exemple de Document XML

Document XML :

```
<!DOCTYPE university [
  <!ENTITY uvsq "Université de Versailles Saint-Quentin-en-Yvelines">
```

```
]>
<p>Bienvenue à &uvsq; !</p>
```

Dans cet exemple, l'entité `&uvsq;` est utilisée directement dans le texte de l'élément `<university>`. Lors de l'affichage ou de la validation, `&uvsq;` sera remplacé par `"Université de Versailles Saint-Quentin-en-Yvelines"`.

Le résultat affiché sera :

```
<p>Bienvenue à Université de Versailles Saint-Quentin-en-Yvelines !</p>
```

Cela montre comment les entités peuvent simplifier l'inclusion de textes récurrents ou longs dans un document XML.

Exercice

Proposez une DTD pour le document exemple-complexe.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE TEI SYSTEM "schema.dtd">
<TEI xmlns="http://www.tei-c.org/ns/1.0">
    <teiHeader>
        <fileDesc>
            <titleStmt>
                <title>Exemple de texte TEI avec une DTD complexe</title>
                <author>Théo Burnel</author>
                <editor>Jean Dupont</editor>
            </titleStmt>
            <publicationStmt>
                <publisher>Éditions XML</publisher>
                <pubPlace>Paris</pubPlace>
                <date when="2024-10-08"/>
                <idno type="isbn">978-1234-5678-90</idno>
            </publicationStmt>
            <sourceDesc>
                <bibl>
                    <title>Manuscrit original</title>
                    <author>Anonyme</author>
                    <pubPlace>Bibliothèque Nationale</pubPlace>
                    <date when="1850"/>
                </bibl>
            </sourceDesc>
        </fileDesc>
    </teiHeader>

    <text>
        <body>
            <div type="chapter" xml:id="ch1">
                <head>Chapitre 1: Introduction</head>
```

```
                <p>
                    Ce texte est un exemple de <term>XML-TEI</term> avec
une DTD complexe. Il contient des références croisées
                    comme le <ref target="ch2">chapitre suivant</ref>, et
des éléments structurés tels que des citations, des
                    annotations, et des métadonnées avancées.
                </p>
                <cit>
                    <quote xml:lang="la">Lorem ipsum dolor sit amet,
consectetur adipiscing elit.</quote>
                    <bibl>
                        <title>Aeneis</title>
                        <author>Vergilius</author>
                        <date when="19BCE"/>
                    </bibl>
                </cit>
            </div>

            <div type="chapter" xml:id="ch2">
                <head>Chapitre 2: Développement</head>
                <p>
                    Ce chapitre contient des annotations linguistiques.
                    <app>
                        <lem>Le texte de base</lem>
                        <rdg wit="ms1">Une première variante</rdg>
                        <rdg wit="ms2">Une seconde variante</rdg>
                    </app>
                    Vous pouvez aussi y trouver des figures et des tables.
                </p>
                <figure xml:id="fig1">
                    <figDesc>Un exemple de schéma XML.</figDesc>
                    <graphic url="schema.png"/>
                </figure>
                <table>
                    <row>
                        <cell>Élément</cell>
                        <cell>Description</cell>
                    </row>
                    <row>
                        <cell>&lt;div&gt;</cell>
                        <cell>Division du texte</cell>
                    </row>
                    <row>
                        <cell>&lt;p&gt;</cell>
                        <cell>Paragraphe</cell>
                    </row>
                </table>
            </div>

            <div type="chapter" xml:id="ch3">
                <head>Chapitre 3: Conclusion</head>
                <p>
                    En conclusion, ce document démontre l'usage de
<term>TEI</term> avec des éléments complexes, y compris
```

```
                    des références internes et externes, des annotations,
et des structures hiérarchiques.
                </p>
                <list type="gloss">
                    <item>
                        <term>TEI</term>
                        <def>Text Encoding Initiative</def>
                    </item>
                    <item>
                        <term>XML</term>
                        <def>Extensible Markup Language</def>
                    </item>
                </list>
            </div>
        </body>
    </text>
</TEI>
```

Solution :

```
<!ELEMENT TEI (teiHeader, text)>
<!ATTLIST TEI xmlns CDATA #REQUIRED>

<!ELEMENT teiHeader (fileDesc)>
<!ELEMENT fileDesc (titleStmt, publicationStmt, sourceDesc)>
<!ELEMENT titleStmt (title, author?, editor?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT editor (#PCDATA)>
<!ELEMENT publicationStmt (publisher, pubPlace, date, idno)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT pubPlace (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ATTLIST date when CDATA #IMPLIED>
<!ELEMENT idno (#PCDATA)>
<!ATTLIST idno type CDATA #IMPLIED>

<!ELEMENT sourceDesc (bibl)>
<!ELEMENT bibl (title, author, pubPlace?, date)>

<!ELEMENT text (body)>
<!ELEMENT body (div+)>
<!ELEMENT div (head, (p|cit|figure|table|list)*)>
<!ATTLIST div type CDATA #REQUIRED xml:id ID #IMPLIED>
<!ELEMENT head (#PCDATA)>

<!ELEMENT p (#PCDATA | ref | term | app)*>
<!ELEMENT ref (#PCDATA)>
<!ATTLIST ref target IDREF #IMPLIED>
<!ELEMENT term (#PCDATA)>

<!ELEMENT cit (quote, bibl)>
```

```
<!ELEMENT quote (#PCDATA)>
<!ATTLIST quote xml:lang CDATA #IMPLIED>

<!ELEMENT app (lem, rdg+)>
<!ELEMENT lem (#PCDATA)>
<!ELEMENT rdg (#PCDATA)>
<!ATTLIST rdg wit ID #IMPLIED>

<!ELEMENT figure (figDesc, graphic)>
<!ATTLIST figure xml:id ID #REQUIRED>
<!ELEMENT figDesc (#PCDATA)>
<!ELEMENT graphic EMPTY>
<!ATTLIST graphic url CDATA #REQUIRED>

<!ELEMENT table (row+)>
<!ELEMENT row (cell+)>
<!ELEMENT cell (#PCDATA)>

<!ELEMENT list (item+)>
<!ATTLIST list type CDATA #IMPLIED>
<!ELEMENT item (term, def)>
<!ELEMENT def (#PCDATA)>
```