Web des données
TP n° 2 – Validation des documents XML

Romain LELONG – romain.lelong@gmail.com

Avant de commencer ...

Outre le présent sujet, les fichiers suivants doivent être récupérés avant de débuter le TP : `my-xml-ko.xml`, `my-xml-ok.xml`, `my-dtd.dtd`, `heros.xml`.

Les différentes parties de ce TP ne sont pas indépendantes. Elle doivent par conséquent être effectuée dans l'ordre.

Ce TP ne sera pas noté et aucun compte-rendu n'est demandé. Vous pouvez le réaliser en binôme ou seul. Les divers éléments abordés dans celui-ci pourront nénanmoins faire l'objet d'une micro-évaluation prochaine.

1 *Bien formé* vs. *Valide*

Dans cette section, on s'intéresse uniquement aux fichiers `my-xml-ok.xml`, `my-xml-ko.xml` et `my-dtd.dtd`.

Étape n° 1 : Les fichiers `my-xml-ok.xml`, `my-xml-ko.xml` sont-ils bien formés ?

Étape n° 2 : Tester chacune la ligne de commande suivante avec chacun des deux fichiers `my-xml-ok.xml` et `my-xml-ko.xml` :

`xmllint --noout` *file.xml*

Étape n° 3 : Cette commande permet-elle :

1. De vérifier si le document *file.xml* est bien formé ?
2. De vérifier si le document *file.xml* est valide ?

Justifier !

Étape n° 4 : À quoi sert l'option `-noout` ?

Étape n° 5 : À l'aide de l'une des lignes de commande suivantes, mettre en évidence la non validité de `my-xml-ok.xml` vis à vis de la **D**ocument **T**ype **D**efinition (DTD) `my-dtd.dtd` :

Ligne de commande n° 1 : `xmllint --noout` *file.xml* `-valid`

Ligne de commande n° 2 : `xmllint --noout` *file.xml* `-dtdvalid` *file.dtd*

Étape n° 6 : Corriger le document `my-xml-ok.xml` afin qu'il soit validé par `my-dtd.dtd`.

Étape n° 7 : De quelle ligne du fichier `my-xml-ok.xml` l'utilisation de la ligne de commande n° 2 permet-elle de se passer ?

2 Construction d'une DTD

Dans cette section, on s'intéresse au fichier **E**xtensible **M**arkup **L**anguage (XML) `heros.xml`.

Étape n° 8 : Construire une DTD `heros.dtd` permettant de valider le document XML `heros.xml`. On veillera a construire les règles DTD permettant de contraindre le plus finement possible le document XML `heros.xml`. Notamment, on s'assurera que l'attribut `genre` ne puisse prendre que les valeurs `F`, `M` ou `NA`.
Remarque : *Il est conseillé de procéder progressivement en partant d'une version épurée du fichier* ***`heros.xml`***.

3 Construction d'un schéma XML

Dans cette section, on s'intéresse encore au fichier XML `heros.xml`. Cependant on souhaite cette fois construire une **XML S**chema **D**efinition (XSD) et non une DTD. Avant de débuter, on rappel que divers outils permettent de valider un document XML vis à vis d'un schéma XSD. En voici un petit aperçus :

command `line XML tool (xmllint)` : Par l'intermédiaire de la ligne de commande :

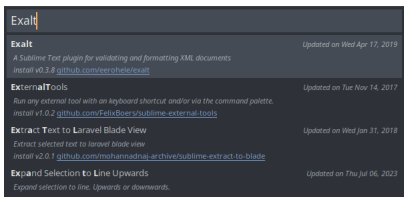
`xmllint --noout` *file.xml* `-schema` *file.xsd*

Sublime Text : Par l'intermédiaire du plugin Exalt [1] :

Ctrl + ⇧ + P

puis

Package Control:Install Package » Exalt » Redémarrage de Sublime Text



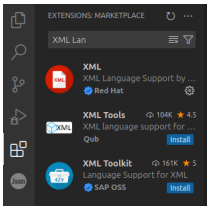
[1] `https://packagecontrol.io/packages/Exalt`

Visual Studio Code : Par l'intermédiaire du plugin *XML Language Support by Red Hat*[2]

Ctrl + ⇧ + X

puis

XML Language Support by Red Hat » install



Étape n° 9 : Construire un fichier `heros.xsd`, contenant un schéma XML permettant de valider le document `heros.xml`. On veillera notamment à :

1. créer et utiliser un *type nommé* pour chacun des *éléments XML* suivants : `powers`, `power`, `hero`, `identity`, `aliases`, `movies`
2. imposer que la valeur de l'attribut `birth-year` soit nécessairement une années.
3. imposer que le contenu de l'élément `abilities` fasse référence à des identifiant XML (en l'occurrence ici les attributs `id` des éléments XML `power`) ;
4. imposer que l'attribut `genre` ait toujours une valeurs et qu'il ne puisse pas prendre d'autres valeurs que `M`, `F` ou `NA`.
5. imposer que l'attribut `height` ne puisse prendre que des valeurs au format `n.nn` ou les n sont des entiers.
6. imposer que les films contenus dans l'élément `movies` soit impérativement au format `c[nnnn]` où `c` est une chaîne de caractères et les `n` des entiers.

Étape n° 10 : Vérifier la validité du document `heros.xml` vis à vis du schéma `heros.xsd` à l'aide de l'outil `xmllint`.

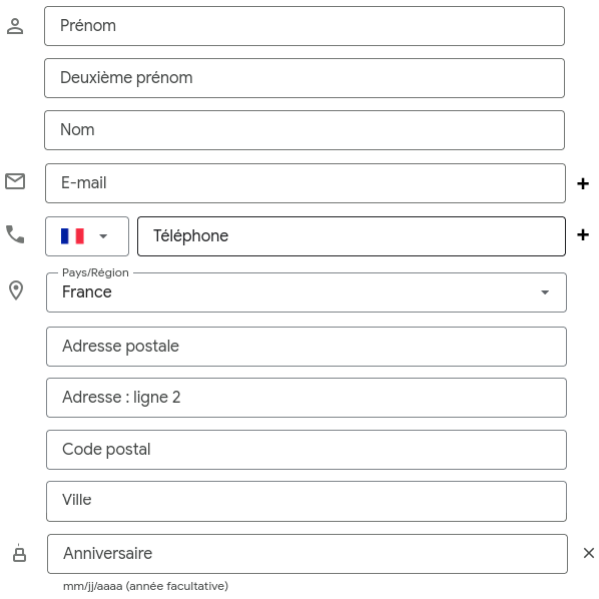
Étape n° 11 : Vérifier la validité du document `heros.xml` vis à vis du schéma `heros.xsd` à l'aide du validateur en ligne `https://www.freeformatter.com/xml-validator-xsd.html`.

4 Répertoire de contact (Facultative)

Étape n° 12 : On souhaite ici permettre la gestion d'un répertoire de contact.
L'objectif est à terme de pouvoir stocker les informations relatives à chacune des personnes de ce dernier au sein d'un fichier XML (i.e. un fichier par personne).

Bien entendu on souhaite « normaliser » ces fichiers afin de pouvoir les traiter informatiquement par la suite. À terme ces fichiers pourront être ajoutés, modifiés ou consultés à l'aide d'un formulaire HTML similaire à celui donné ci-contre.

Proposer une solution technique adaptée à cette problématique. Utiliser des commentaires pour justifier ou expliquer vos choix.
Indication : *Il n'est pas demandé de développer le formulaire HTML mais seulement de proposer une solution technique pour la partie XML.*



[2] `https://marketplace.visualstudio.com/items?itemName=redhat.vscode-xml`