Séance 3 - XML, des documents numériques

- Comment définit-on le vocabulaire (noms des éléments et des attributs) dans une dtd ?
- Comment définit-on, avec une dtd, l'arborescence d'un document xml ?

Le vocabulaire est définit après les indactions <!ELEMENT ou <!ATTLIST pour les attributs.
L'arborescence est représentée dans l'élément parent par le nom de des élément entre parenthèses.

- Indiquer les différents corrections effectuées et donner les **résultats** corrigés (rappeler vous qu'utiliser XML n'est utile que si l'on utilise des données multiples, merci de mettre plusieurs bons de commandes dans votre fichier XML par exemple).

Des corrections de syntaxe ont été effectuées.
La hiérarchie a été modifié pour utilisé l'attribut « idref » afin d'éviter la redondance.

A ce stade les dtd et xml correspondent, selon la vérification faite avec xmllint, voir codes :

Fichier dtd des Cds :

https://github.com/MaxPDev/Interop_rendus/blob/master/Seance3/def.dtd

Fichier XML correspondant :

https://github.com/MaxPDev/Interop_rendus/blob/master/Seance3/cd.xml

Fichier dtd bon de commandes :

https://github.com/MaxPDev/Interop_rendus/blob/master/Seance3/bonDeCommande.dtd

Fichier xml bon de commandes :

https://github.com/MaxPDev/Interop_rendus/blob/master/Seance3/bonDeCommande.xml

Séance 3 - Interlude : open data partout...

- A votre avis, quel.s type.s d'utilisateur.s privilégie.nt quel.s format.s ?
- Quel peut-être l'intérêt de changer le format des données ?

Le CSV est répandu, plus facile à appréhender pour un non-développeur, à extraire ou importer depuis un tableur.

Le format JSON sera privilégié par la plupart des développeur d'application web, sa manipulation étant aisé à intégrer à la POO, léger et lisible.

Le XML pourrait-être préféré pour ses possibilités de structuration et le contrôle sur celle-ci qu'il permet. Peut-être un meilleurs choisir pour interagir des programmes de taille importante, à la structuration des données plus complexes.

On peut être amené à changer le format selon des critères de lisibilité, de structuration, et selon la facilité et/ou la pertinence de l'intégration du format au reste de « l'éco-système » du programme.

Une question ouverte : les exemples de la première séance de td passaient par vos navigateurs.

- Pourquoi cette utilisation ?

  Par simplicité, l'usage étant devenu quasi universel.

- Que signifie-t-elle en terme d'utilisation du réseau, de requêtes ?

  Les ressources (lecture, création, maj, suppression) sont accessibles des requête HTTP, la mise en forme étant gérée par le client.

Séance 3 - Et l'interopérabilité ?

- Ce fichier est-il bien formé (donner la commande `xmllint` et son résultat).
- Ce fichier est-il valide (donner la commande `xmllint` et son résultat).

xmllint --dtdvalid films.dtd films.xml : aucune erreur

xmllint films.xml : aucune erreur

Question : Quelle transformation du fichier xml effectue la feuille de styles xsl ?

Cette première feuille de style n'effectue aucune transformation.

Questions :

- Que fait cette nouvelle feuille de styles
- Quel rôle joue la règle (*template*) sur les éléments de type *text()* (ligne 28). Commentez la pour comparer.

Cette nouvelle de style utilise la structure xml pour n'en afficher en output que les balises prénoms, fils de scénario.

text() cible le texte contenu entre les balises d'un élément. En appliquant un template qui ne fait rien, il n'effectue aucun action sur celui-ci en output, le text n'apparaît donc pas, et réapparaît si on comment la ligne.

Feuille de style générique :

https://github.com/MaxPDev/Interop_rendus/blob/master/Seance3/generiqueXsl.xsl

Que vous apporte l'expression `<xsl:strip-space elements="*"/>`, à placer juste après `xsl:output` ?

L'expression permet de supprimer les nœuds d'espace blanc, issus entre autre des retours chariot, sauts de ligne, tabulations etc...