1. Introduction

Commencez par créer un dossier `tp3` pour ce TP et travaillez dedans.

2. XPath - Tutoriel initial

Nous allons écrire des requêtes XPath sur différents fichiers. Téléchargez le document `albums_mod.xml`. NB: il s'agit d'une version altérée de `albums.xml` dans laquelle il manque certains attributs et éléments :

- L'attribut `numero="2"` de L'album *Tintin au Congo* a été supprimé.
- Il manque l'élément `<date>...</date>` dans l'album *Les Cigares du Pharaon*.
- L'album *Le Lotus bleu* a un 2e titre.
- L'attribut `numero` et l'élément `mois` ont été enlevés de *L'Oreille cassée*.
- L'attribut `numero` de l'album *L'Île Noire* est vide.

Pour l'exécution d'une requête, vous avez le choix, soit de travailler en ligne de commande, soit avec XML Copy Editor, soit éventuellement avec ce formulaire.

- en ligne de commande, pour afficher des éléments :

  ```
  xmllint --xpath 'expression' albums_mod.xml
  ```

- en ligne de commande, l'affichage des textes est un peu meilleur :

  ```
  xmlstarlet sel --template --value-of 'expression' albums_mod.xml
  ```

- avec XML Copy Editor :
  Chargez le fichier. La touche `F9` permet de saisir une expression XPath. Les résultats sont affichés dans un nouvel onglet. Fermez cet onglet avant de saisir une nouvelle expression, sinon elle s'appliquera à cet onglet.

Voici maintenant des requêtes de complexité croissante à essayer et à comprendre, `exemples_xpath`. À vous d'exprimer en français ce que calculent ces requêtes. Il n'est pas demandé de rédiger les réponses. C'est seulement pour comprendre XPath. N'hésitez pas à demander des explications si quelque chose vous semble étrange.

- Requêtes de base
  - `/albums/album/titre`
  - `/albums/album/titre/text()`
  - `//titre`
  - `/albums/album/@numero`
  - `/albums/album/@numero/text()` ne marche pas car un attribut n'a pas de texte
- Requêtes simples avec des conditions
  - `/albums/album[@numero=6]`
  - `/albums/album[@numero=8]/date/annee`
  - `/albums/album[@numero>22]/titre`
  - `/albums/album[titre="Tintin au Tibet"]`
  - `/albums/album[titre="Tintin au Tibet"]/@numero`
  - `/albums/album[@serie="Tintin" and @numero=5]`
  - `/albums/album[@serie="Tintin"][@numero=5]`

  - `/albums/album[date/annee=1963]`
  - `/albums/album/date[annee=1963]`
  - `//album[.//annee=1954]` le point fait repartir de l'album considéré
  - `//album[//annee=1954]` attention, `//` fait repartir de la racine, et comme cette condition est vraie (l'un des albums est de 1954), tous les albums sont affichés
  - `//album[date/mois="octobre"]/titre`
  - `//date[mois="août"]/annee`
- Requêtes sur la position des éléments
  - `//album[position()=1]/titre`
  - `//album[1]/titre` 1 est pris comme un indice dans la liste des albums, les indices commencent à 1
  - `//album[5]/titre` ce sont les sous-éléments `<titre>` du 5e album du fichier, ici, on a deux titres pour le même album
  - `//album[position()=last()]/titre`
  - `//album[last()]/titre` Dans cet exemple, `last()` retourne le numéro du dernier élément de la liste, on peut s'en servir comme indice
  - `//album[@serie="Tintin" and last()]/titre` (attention, *last()* employé ainsi veut dire qu'on sélectionne ces albums s'il y a un dernier de la liste quelque part – évidemment il y en a un, donc tous sont sélectionnés)
  - `//album[@serie="Tintin"][last()]/titre` (on peut mettre deux filtres)
  - `//album[@serie="Tintin" and position()=last()]/titre` (ne retourne rien car le dernier album du document n'est pas un Tintin)
  - `//album[@serie="Tintin"][position()=last()]/titre` (le second filtre s'applique sur la sélection faite par le premier)
  - `//titre[last()]` : piège car il retourne les éléments `<titre>` qui sont les derniers *dans leur parent*, et non pas le dernier `<titre>` du document. Les éléments ne sont pas détachés de leur parent. Ça peut compliquer certaines requêtes.
  - `//album[last()]/titre[last()]` : dernier titre du derrnier album
- Présence ou absence d'attributs et de sous-éléments
  - `//album[not(date)]` cela affiche l'album qui n'ont pas l'élément `date`.
  - `//album[not(date/mois)]` cela affiche les albums qui n'ont soit pas l'élément `mois`, soit carrément pas `date` du tout.
  - `//album[date and not(date/mois)]` cela affiche l'album sans élément `mois` mais avec l'élément `date`.
  - `//album[not(@numero)]`
  - `//album[@numero=""]`
  - `//album[not(@numero) or @numero=""]`
  - `//album[@numero]`
  - `//album[@numero!=""]`
- Utilisation des axes
  - `//date[mois="août"]/..`
  - `//mois[.="août"]/../..`
  - `//annee[.=1966]/../../titre`
  - `//date[mois="août"]/parent::album`
  - `//mois[.="août"]/ancestor::album`
  - `/descendant::titre`
  - `//album[titre="On a marché sur la Lune"]/preceding-sibling::album[1]`

  - `//album[titre="Les Sept Boules de cristal"]/following-sibling::album[1]`
  - `/descendant::album[attribute::numero="5"]/child::titre`
- Sous-requêtes
  - `//album[date/mois=//album[titre="On a marché sur la Lune"]/date/mois]`
- Fonctions
  - `//album[count(titre) > 1]`
  - `//album[string-length(titre) < 10]/titre/text()`
  - `//album[contains(date/mois, "ars")]`
  - `//album[starts-with(titre, "Tintin au")]` NB: il n'y a pas la fonction `ends-with`, elle n'est disponible qu'en XPath2.0
  - `//album[substring(titre, string-length(titre) - string-length("Soleil") + 1) = "Soleil"]` voila comment faire `ends-with` en XPath1.0

3. XPath - Exercices

Nous allons travailler avec le fichier `spatial.xml` (`spatial.dtd` si nécessaire) qui permet de faire des requêtes assez variées.

Téléchargez ReponsesTP3.txt. C'est là que vous devrez enregistrer chacune de vos réponses aux questions. Si vous voyez plusieurs solutions vraiment différentes ou que vous n'avez pas tout à fait la bonne réponse mais plusieurs propositions proches, vous pouvez les mettre l'une après l'autre. Par exemple, mais ce ne sont pas du tout les bonnes réponses :

```
Question 4.1.a spatial.xml : nations qui ont un programme spatial
/programmes//astronaute[@role="directeur"]
//attribute::role[.="directeur"]/parent::nation

Question 4.1.b spatial.xml : astronautes commandants
//objectif[@role="pilote"]/ancestor::nation
```

IMPORTANT : ne changez ni le nom, ni la structure du fichier `ReponsesTP3.txt`. Vous devez seulement placer vos réponses en dessous des noms des questions. N'insérez surtout pas de retour à la ligne dans une requête si elle est trop longue. Laissez-la comme elle est, sur une seule ligne, sinon elle serait considérée comme deux requêtes différentes pour la même question, toutes deux fausses car pas complètes.

Si vous avez un message pour le correcteur, placez-le seul sur une ligne commençant par `##`.

Pour vérifier ce que vous allez déposer sur Moodle, on vous conseille d'utiliser CheckTP3.html. C'est une page HTML contenant un script capable d'analyser vos réponses. Il ne vous donnera pas votre note, mais vous saurez si vos réponses ont une chance d'être correctes.

3.1. Fichier spatial.xml

Écrire les requêtes XPath répondant aux questions suivantes.

a. Afficher les noms des nations qui ont un programme spatial : `nom="USA"`, `nom="URSS"`, ou seulement `"USA"`, `"URSS"`, comme vous voulez (et c'est pareil pour les questions suivantes).

b. Afficher les noms des astronautes dont le rôle a été au moins une fois pilote (il se peut que certains soient en double).
c. Afficher les noms des missions effectuées par Gordon Cooper.
d. Afficher les dates des missions dont le but est absent ou vide.
e. Afficher l'élément XML `astronaute` du pilote de la mission Apollo 11.
f. Afficher les noms des missions ayant au moins un but en commun avec ceux de Gemini 12.
g. Afficher les éléments XML buts des missions du programme Apollo avec Gene Cernan.
h. Afficher le nom des missions non habitées et sans événement (aucun élément `evenement` dedans).
i. Afficher l'élément XML de la dernière mission automatique russe (la dernière mission automatique du dernier programme, tel que dans le fichier, ne pas se baser sur la date).
j. Afficher l'élément XML de la mission qui précède Soyouz 4 dans le fichier.
k. Afficher les noms des missions qui ne comptent qu'un seul astronaute.

3.2. Fichier arbres.xml

Téléchargez maintenant le fichier `arbres.xml` (`arbres.dtd` si nécessaire). C'est une adaptation en XML du document arbres.csv disponible sur opendata. Les éléments décrivent des arbres remarquables de Paris : position GPS, genre, espèce, famille, année de plantation, hauteur, circonférence, lieu, etc. Les questions suivantes sont mises au pluriel, mais il est possible qu'il n'y ait qu'une seule réponse.

a. Afficher les noms des lieux du 5e arrondissement dans lesquels il y a un arbre.
b. Afficher les numéros d'arrondissement où se trouvent des arbres du genre *Taxus*.
c. Afficher le nom du lieu des arbres dont l'adresse est vide.
d. Afficher les noms communs des arbres du Bois de Vincennes.
e. Afficher les noms des lieux d'arbres de plus de 600cm de circonférence.
f. Afficher les hauteurs des arbres plantés entre 1800 et 1850 compris.
g. Afficher les noms communs des arbres de la même espèce que celui planté en 1897.
h. Afficher les identifiants des arbres dont l'adresse est vide.
i. Afficher les années des arbres du genre *Fagus* dans le 16e arrondissement.
j. Afficher l'identifiant du premier arbre (du fichier) situé dans le 19e arrondissement.

4. Travail à rendre

Déposez seulement `ReponsesTP3.txt` sur Moodle, dans la page L4IN121T Formats et traitements de données internet. Vous pouvez vérifier ce qui sera noté avec CheckTP3.html.