XPath

- / : chemin absolue
- // : chemin relatif $\equiv$ **`descendant-or-self::node()
- element[cond] : filtrage selon condition dans crochet
    - **Rmq : si un element de l'ensemble verifie la condition, la condition est vrai**
        - ex : 15 5) cf. ci- dessous
    - si dans le crochet y'a qu'un seul `element` cela veut dire il existe au moins un element
        - ex : `//ville[not(plusBeauMonument)]` avec `plusBeauMonument` dans le [] veut dire il existe au moins 1 plus beau Monument
        - ici dans l'exemple nous avons **les villes sans plus beau monument**
    - utilisation de `not` : cf. question 14
- les axes XPath ◦ child : les enfants du nœud contextuel.
  ◦ self : le nœud courant.
  ◦ parent : le parent du nœud contextuel.
  ◦ descendant : les descendants du nœud contextuel
  ◦ ancestor :les éléments ancêtres du nœud contextuel.
  ◦ following-sibling : tous les nœuds frères (nœuds qui ont le même parent)
  qui suivent le nœud contextuel.
  ◦ preceding-sibling :tous les frères qui précèdent le nœud contextuel.
  ◦ following: tous les nœuds qui sont visité après le nœud contextuel dans
  l'ordre du document, à l'exclusion des descendants.
  ◦ preceding: tous les prédécesseurs du nœud contextuel, à l'exclusion des
  ancêtres.
  ◦ attribute : tous les attributs du nœud contextuel
    - exemple :
        - Syntaxe abrégée et
          syntaxe étendue
          Exemples :
          film/annee = child::film/child::annee
          /cinema//artiste = /child::cinema/descendant-or-self::node()/child::artiste
          // "etoile"/artiste/role = /descendant-or-self::node()/child::"etoile
          "/child::artiste/child::role
          film/../film/@lang = child::film/parent::node()/child::film/attribute::lang

TD

Exercice 1

Xpath qui s'applique sur un element de l'ensemble question 1

1. tous les menus à moins de 50 EUR `/base/restaurant/menu[@prix <50]` **chemin absolue**
   `//menu[@prix<50]` **chemin relatif**
   `restaurant/menu[@prix < 50]`
2. les menus des restaurants 2 ou 3 étoiles
   `//restaurant[@etoile = 2 or @etoile = 3]` **utilisation de or**

3. le nom des villes dans le département 69 `//ville[@departement = 69]/@nom` **condition sur ville renvoyant l'attribut nom de element ville**
4. le nom des restaurants à Lyon
   `//restaurant[@ville = 'Lyon']/@nom`
5. le nom des restaurants dans le département 75
   `//restaurant[@ville = //ville[@departement = 75 ]/@nom]`
   A = B signifie $$\exists x \in A , \exists y \in B, x=y $$
   c'est a dire il existe une ville du departement 75 tq son nom = la valeur de l'attribut ville du restaurant
   autre solution:
   `//restaurant[@ville = ../ville[@departement = 75]/@nom]/@nom`
   .. permet de remonter vers l'element base contenant les villes et les restaurants
6. Le plus beau monument des villes ayant au moins 1 restaurant 3 étoiles
   `//ville[@nom = //restaurant[@etoile =3]/@ville]/plusBeauMonument`
   autre solution
   `id(//restaurants[@etoile = 3]/@ville)/plusBeauMonument`
7. les villes avec au moins un restaurant qui a au moins 4 menus
   `//ville[@nom = //retaurant[count(menu)>=4]/@ville]`
   autre solution sans count () un restaurant pour lequel il existe un 4eme menu (position() = 4)
   `//ville[@nom = //restaurant[menu[position() = 4]]/@ville`
   on peut ecrire `menu[4]` a la place de `menu[position() = 4]`
8. les restaurants 3 étoiles fermés le dimanche
   `//retaurant[@etoile = 3 and contains(./fermeture,'dimanche']`
   contains(A, B) vrai si A contien B
9. les restaurants ayant au moins un menu contenant le nom de la ville
   `//restaurant[menu[contains(@nom , ../@ville)]`
10. a. le 2ème menu de chaque restaurant (notion d'axe)
    `//menu[2]`
    `//restaurant/menu[2]`
    `descendant-or-self::node()/child::menu[2]`
    `descendant-or-self::node()` equivalent a // b.
    `//descedant::menu[5]` attention l'expression `//menu[5]` n'est pas une expression equivalente .
11. le nombre d'étoiles des restaurants qui se trouvent dans la troisième ville du documents
    `//restaurant[@ville = /descendant :: ville[3]/@nom]/@nom` rmq : tous les elements ville sont des fils directs de base, donc il est possible ici de remplacer `/descendant :: ville[3]` par `//ville[3]`
12. (a) le 2ème menu à moins de 150 EUR de chaque restaurant.
    `//restaurant/menu[@prix <150][2]`
    position 2 dans le resultat de `//restaurant/menu[@prix <150]`
13. (b)Le 2ème menu de chaque restaurant s'il vaut moins de 150 EUR.
    `//restaurant/menu[position() = 2 and @prix <=150]`
    ou
    `//restaurant/menu[position() = 2][@prix <=150]`
14. les villes sans restaurant 3 étoiles (negation avec not())
    `//ville[@nom = //restaurant[not(@etoile =3)]/@ville]` => faux donne les villes avec aumoins 1 restaurant qui n'a pas 3 etoile

bonne solution : `ville[not(@nom = //restaurant[@etoile= 3]/@ville)]` il n'existe pas de ville d'un restau 3 etoile egale au nom de la ville recherchee

15. (a) les villes sans plus beau monument
    `//ville[not(plusBeauMonument)]` avec `plusBeauMonument` dans le [] veut dire il existe au moins 1 plus beau Monument
16. (b) les restaurants dans une ville sans plus beau monument
    `//restaurant[@ville = //ville[not(plusBeauMonument)]/@nom]` reponse incorrecte : `//restaurant [@ville != //ville[plusBeauMonument]/@nom]` car si un element de l'ensemble verifie la condition la condition est vrai
17. les noms des restaurants dont tous les menus coûtent moins cher que les menus du restaurant "Les quatre saisons".

TME

XPath Notebook

(: 1. Tous les titres de films.) `//FILM/TITRE! string()`

(: Les titres des films d'horreur.) `//FILM[GENRE='Horreur']/TITRE! string()`

(: Le résumé d'Alien.) `//FILM[TITRE = 'Alien']/RESUME! string()`

(: Titre des films avec James Stewart.) `//FILM[ROLES/ROLE[NOM = 'Stewart' and PRENOM = 'James']]/TITRE ! string()`

(: Titre des films avec James Stewart et Kim Novak.) `//FILM[ROLES/ROLE[NOM = 'Stewart' and PRENOM = 'James'] and ROLES/ROLE[NOM = 'Novak' and PRENOM = 'Kim']]/TITRE ! string()`

(: Quels films ont un résumé ?) `//FILM[RESUME] ! string()`

(: Quels films n'ont pas de résumé ?) `//FILM[not(RESUME)] ! string()`

(: Quel est l'identifiant du metteur en scène du film Vertigo?) `//FILM[TITRE = 'Vertigo']/MES/@idref ! string()`

(: Quel rôle joue Harvey Keitel dans le film Reservoir dogs ?) `//FILM[TITRE = 'Reservoir dogs']/ROLES/ROLE[NOM = 'Keitel' and PRENOM = 'Harvey']/INTITULE ! string()`

(:Quel est le dernier film du document ?) `//FILM[last()] ! string()`

(: Quel est le titre du film qui précède immédiatement le film Shining (dans l'ordre du document).
`//FILM[TITRE = 'Shining']/preceding-sibling::FILM[1]/TITRE ! string()`

(: Donnez les titres des films qui contiennent un 'V' (utiliser la fonction contains))
`//FILM[contains(TITRE, 'V')]/TITRE/text() ! string()`

(: Donner les noeuds qui ont exactement trois descendants (utiliser la fonction count).) `//descendant-or-self::node()[count(descendant::*) = 3] ! string()`

(: Donner les noeuds dont le nom contient la chaîne 'TU' (fonction name)) `/descendant-or-self::node()[contains(name(), 'TU')]` (: or) `//*[contains(name(), 'TU')]`