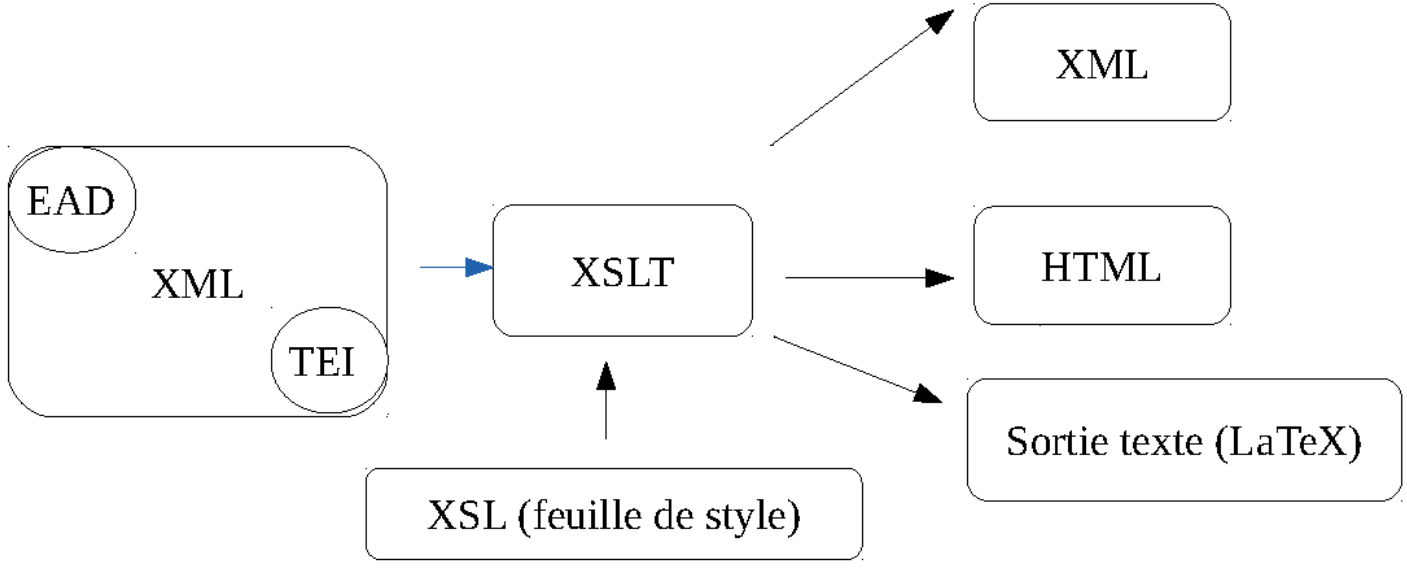
Cours Cosme 2019

XPath et fondamentaux du langage XSLT

Ariane Pinche

Lyon, 23 avril 2019

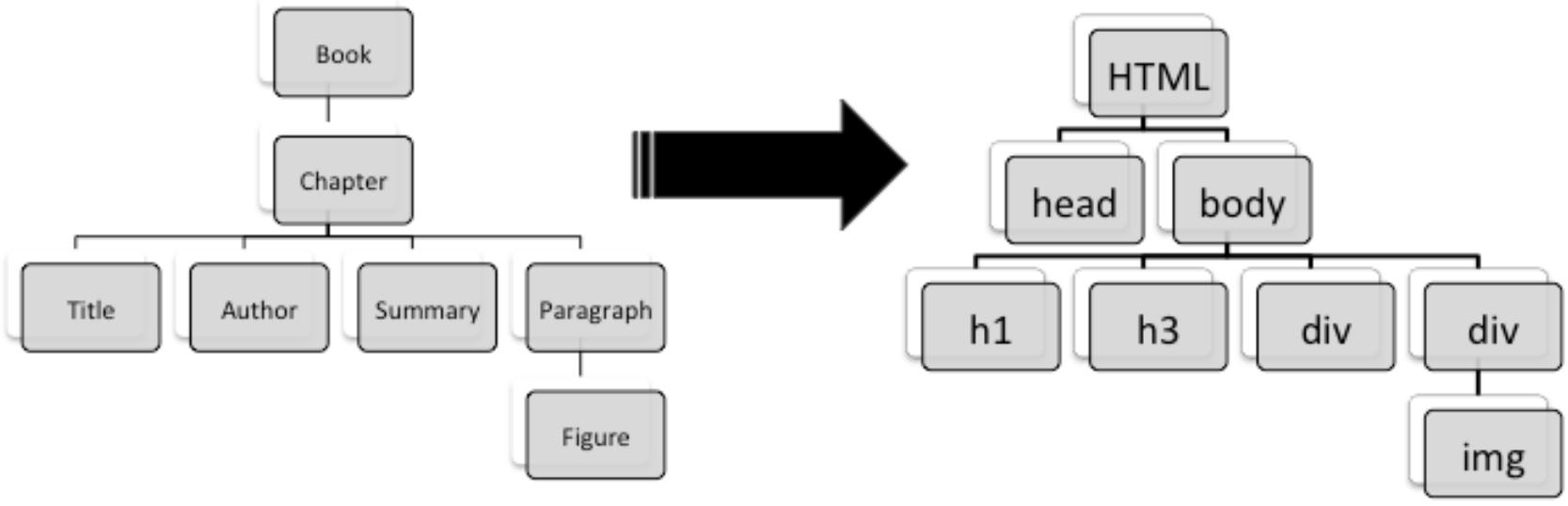
I- XSLT et l'environnement XML

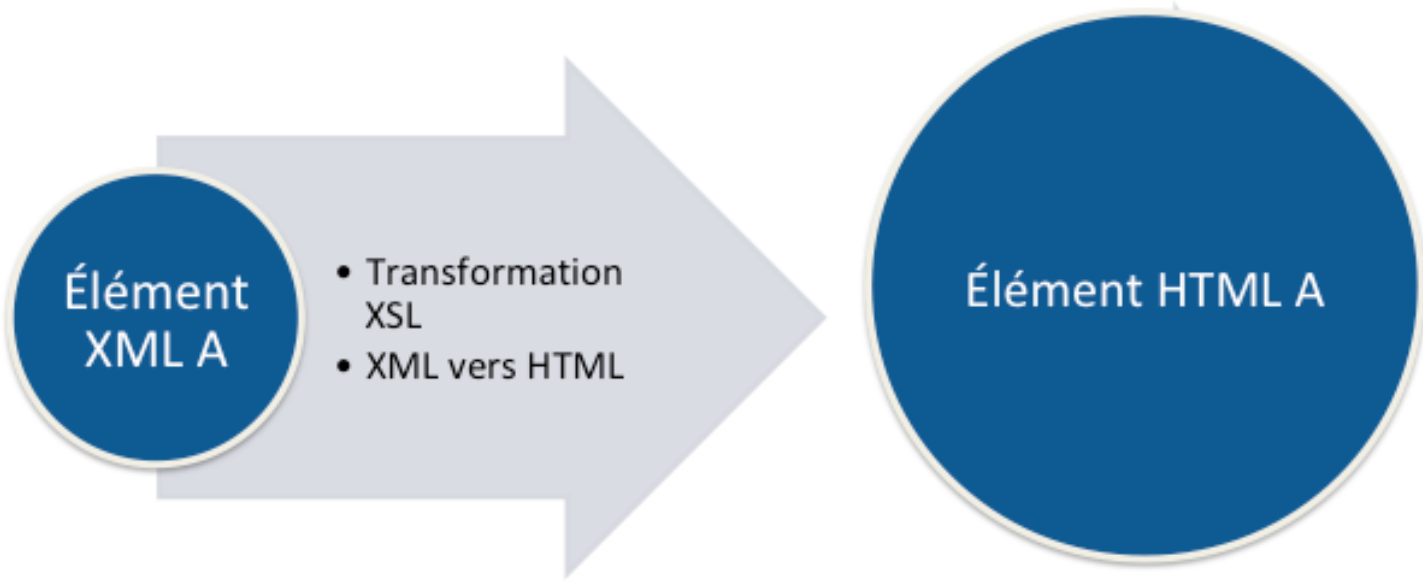


« **XSLT** (extensible Stylesheet Language Transformations) est un langage de programmation fonctionnel utilisé pour spécifier comment un document XML doit être transformé en un autre document qui peut, mais qui n'est pas nécessairement, un autre document XML.

Un processeur XSLT lit un arbre XML en entrée et une feuille de style XSL et produit un arbre résultat en sortie. »

Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet [et al.], *XML en concentré*, Paris, O'Reilly, 2005, p. 519.


Book
Chapter
Title
Author
Summary
Paragraph
Figure
HTML
head
body
h1
h3
div
div
img


Élément XML A
• Transformation XSL
• XML vers HTML
Élément HTML A

II- XPath

- Xpath 2.0 est publié par le *World Wide Web Consortium* (W3C) et s'inscrit dans la famille des standards XML;
- Xpath est un langage de requêtes qui permet de parcourir un arbre XML;
- Xpath a été conçu comme un langage intégré, et non pas un langage autonome;
- **Attention** : Xpath est un « read-only » langage.

Pour aller plus loin : Michael R. Kay, *XPath 2.0 programmer's reference*, Indianapolis, IN, Wrox Press, 2004. p.1-5.

1- Bases de la syntaxe Xpath

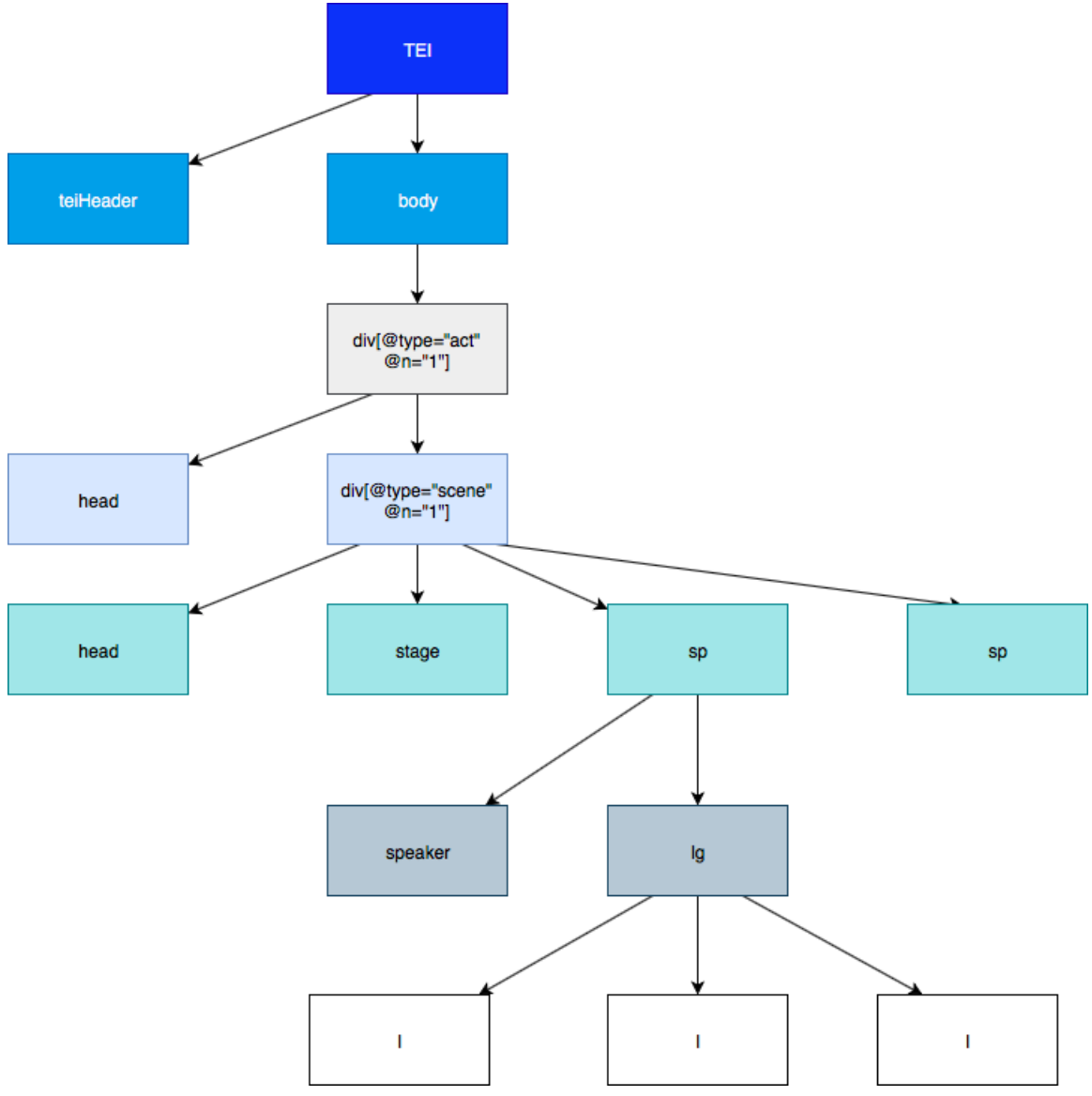
- Xpath permet d'exprimer un chemin vers un élément d'un arbre.
- Une expression de chemin correspond à une séquence d'étapes séparées par l'opérateur « / ».
    - *TEI/body/div/head*

Sans indication particulière, la relation se fait d'un élément parent vers un élément enfant.

Pour en savoir plus :

Michael R. Kay, *XPath 2.0 programmer's reference*, Indianapolis, IN, Wrox Press, 2004, p.215-216.

Arbre simplifié du document XML Andromaque



Exercices

Quels éléments sont sélectionnés par les chemins suivants ?

- TEI/body/div/div;
- TEI/body/div/div/head;
- TEI/body/div/div/sp.

À partir de l'arbre XML simplifié d'*Andromaque* donner les chemins suivants en partant de l'élément racine TEI

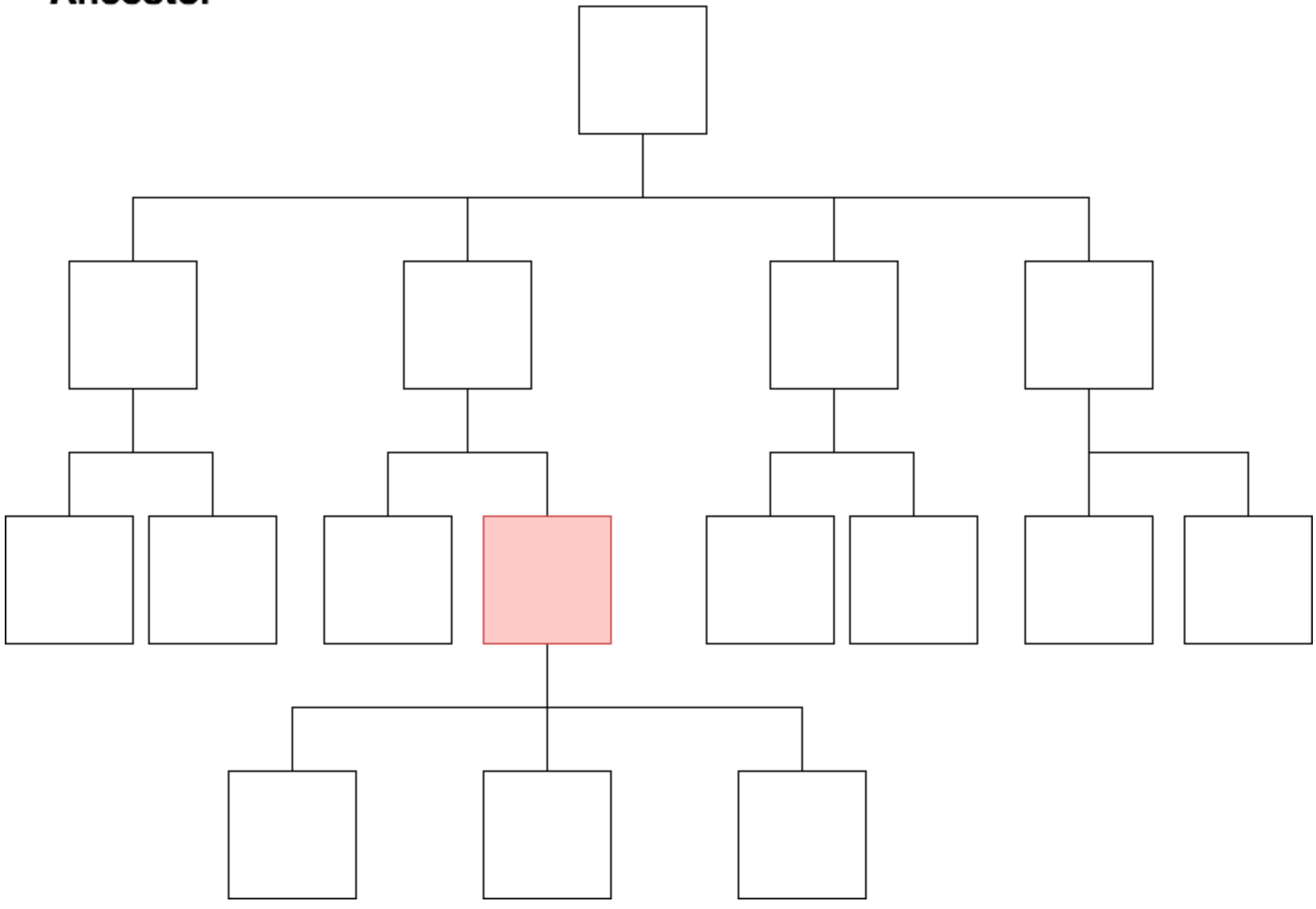
- Donner le chemin vers "body";
- Donner le chemin vers la "div" dont l'attribut type est égal à "act";
- Donner le chemin vers "stage";
- Donner le chemin vers "speaker";
- Donner le chemin vers les "l".

2- Les principaux axes Xpath

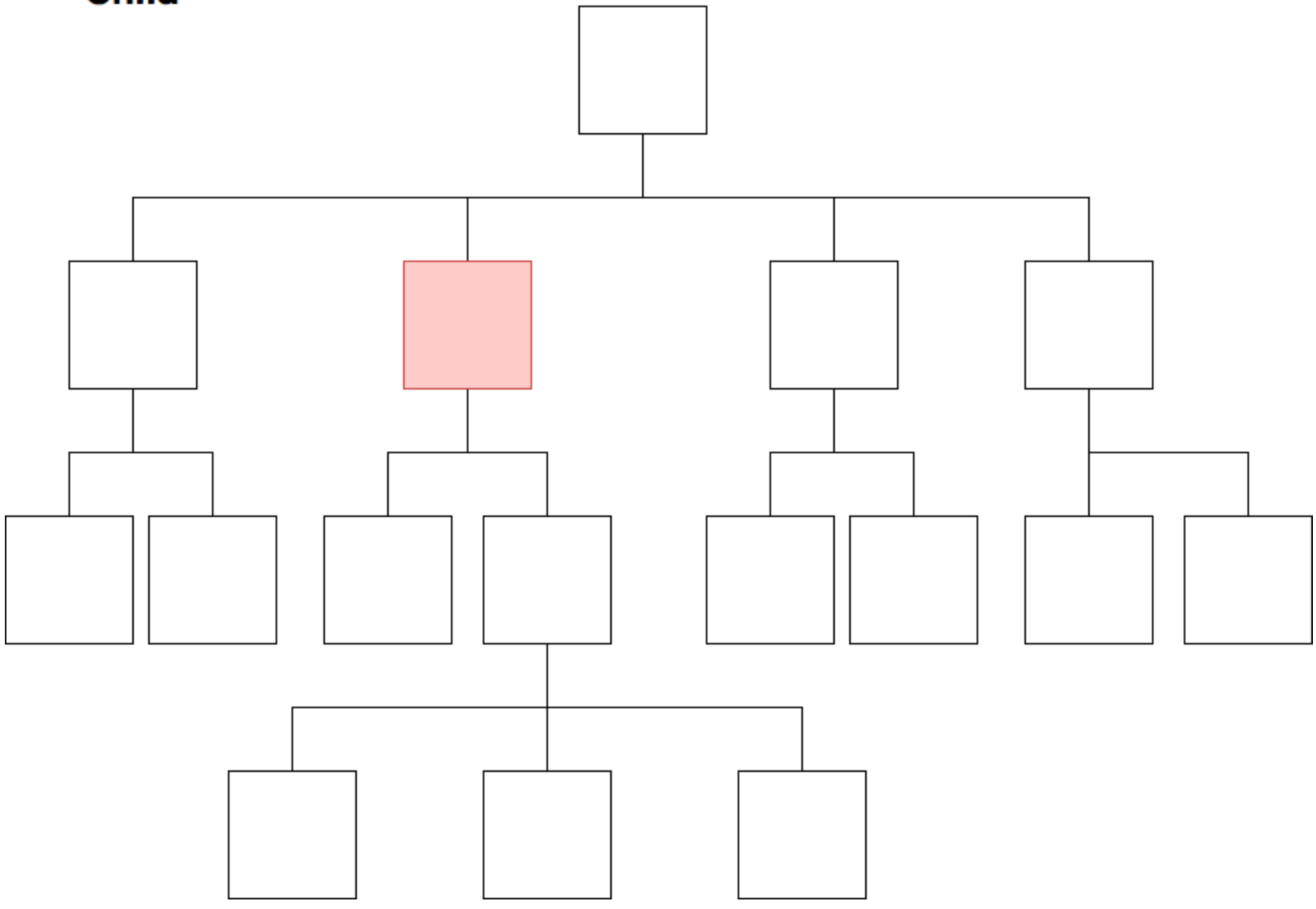
- *child axis* : sélectionne les éléments et les textes des nœuds enfants
    - body/teiHeader
- *descendant axis* : sélectionne les noeuds descendants directs ou indirects d'un élément
    - body//div
- *parent axis* : sélectionne l'élément parent d'un élément
    - div/parent::body
- ancestor axis : sélectionne tous les éléments parents directs ou indirects en remontant jusqu'à l'élément racine
    - l/ancestor::sp
- *attribute axis* : sélectionne l'attribut d'un élément
    - div/@type

Exercices

Ancestor



Child



À partir de l'arbre XML simplifié d'*Andromaque*, donner les chemins suivants avec la syntaxe la plus concise possible :

- Donner le chemin vers la "div" dont l'attribut type est égal à "act" depuis la racine;
- Donner le chemin vers "stage";
- Donner le chemin depuis "l" vers "sp";

3-Les prédicats Xpath

Pour affiner sa requête Xpath, on peut ajouter des **prédicats**.
Le prédicat est noté entre crochets droits après l'élément auquel il se rapporte.

- On peut ainsi spécifier la position d'un élément :
  - l[1]
- Une propriété de l'élément :
  - div[@type="scene"]

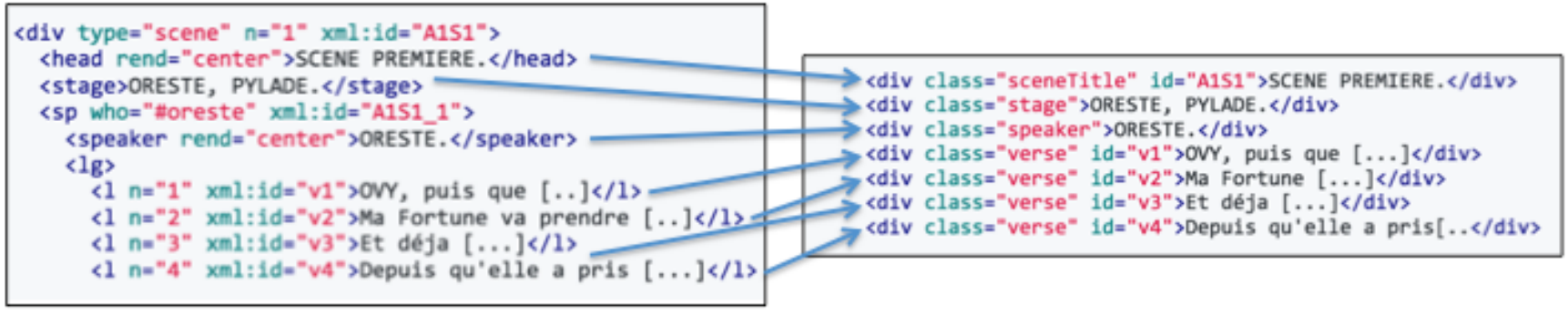
Exercices

À partir du fichier andromaque_c.xml, donner les chemins suivants avec la syntaxe la plus concise possible :

- Récupérer tous les vers de la première réplique;
- Récupérer tous les noms des personnages déclarés dans le teiHeader;
- Récupérer le premier vers de la pièce.

III-XSLT

Transformation XSLT vers HTML



Syntaxe XSLT

```
<xsl:template match="div[@type='play']">
    <h1><xsl:value-of select="head"/></h1>
    <xsl:apply-templates select="div[@type='act']"/>
</xsl:template>

<xsl:template match="div[@type='act']">
    <div class="actTitle" id="{@xml:id}">
        <xsl:value-of select="head"/>
    </div>
    <xsl:apply-templates select="div"/>
</xsl:template>

<xsl:template match="div[@type='scene']">
    <div class="sceneTitle" id="{@xml:id}">
        <xsl:value-of select="head"/>
    </div>
    <div class="stage">
        <xsl:value-of select="stage"/>
    </div>
    <xsl:apply-templates select="sp"/>
</xsl:template>
```

Légende
XSLT
Xpath
HTML

Initier un fichier XSLT

- Ouvrir le fichier XML *andromaque_c.xml*
- Ouvrir un nouveau fichier de format XSLT
- Ce fichier contient l'en-tête suivante :

  ```
  <xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs" version="2.0">
  ```

  [...]

  ```
  </xsl:stylesheet>
  ```

- Rajouter : `xpath-default-namespace="http://www.tei-c.org/ns/1.0"` après la déclaration *xmlns:xs*

Observer

1. Ecrire dans le feuille XSLT la règle suivante et l'appliquer :

```
<xsl:template match="/">
    <xsl:apply-templates/>
</xsl:template>
```

2. Appliquer la règle suivante :

```
<xsl:template match="text()"/>
```

3. Sélectionner un élément du document source et copier son contenu

```
<xsl:template match="div[@type='play']/head">
  <h1><xsl:value-of select="."/></h1>
</xsl:template>
```

Mettre en place une règle

- Pour donner des instructions XSL, on utilise des templates grâce à l'élément :

```
<xsl:template>
```

- On utilise l'attribut match pour sélectionner grâce à un chemin Xpath un élément de l'arbre XML :

```
<xsl:template match="mon_element_xml">
```

On peut ajouter dans la règle du texte,

```
<xsl:template match="div[@type='play']/head">
    Ici, il y avait mon élément
</xsl:template>
```

mais aussi des balises, ou un motif qui permet de récupérer certains éléments de l'arbre XML soit tels quels, soit transformés ou triés par des fonctions XSL ou Xpath.

```
<xsl:template match="div[@type='play']/head">
  <h1>
      <xsl:value-of select="."/>
  </h1>
</xsl:template>
```

NB: La fonction XSLT *value-of* permet de récupérer le contenu textuel d'un balise.

Principe de fonctionnement du langage de XSLT

- « Par défaut, un processeur XSLT lit le document XML d'entrée de haut en bas, commençant à l'élément racine et descendant dans l'arborescence en suivant l'ordre d'apparition des éléments. Les règles modèles sont activées dans l'ordre de rencontre des éléments. Ceci signifie qu'une règle modèle pour un élément sera activée avant les règles modèles correspondant à ses sous-éléments. »

Elliotte Rusty Harold, W. Scott Means, Philippe Ensarguet [et al.], *XML en concentré*, Paris, O'Reilly, 2005, p. 164.

- **XSLT est un langage XML**, il possède la même logique d'imbrication. Ainsi une règle XSLT s'applique à l'élément sélectionné et ses enfants. Si on veut que les règles écrites pour un élément enfant s'appliquent, il faut en autoriser l'application en ajoutant dans la règle de l'élément parent la fonction XSLT *apply-templates*.

```
<xsl:template match="div[@type='act']">
        <div class="actTitle" id="{@xml:id}">
            <xsl:value-of select="head"/>
        </div>
        <xsl:apply-templates/>
</xsl:template>
```

Exercice guidé

Transformer le fichier XML *andromaque_c.xml* pour créer une page HTML simple.

Mettre en place la structure HTML

1- Ouvrir le fichier XML;
2- Ouvrir le fichier XSL *XSL_HTML_structure.xsl*
3- Compléter la première règle pour structurer les informations dans une page HTML minimale :

- Indiquer qu'on n'appliquera que les règles qui porteront uniquement sur les éléments enfants de `div[@type='play']`

Ajout d'informations depuis le XML source vers le fichier HTML de sortie

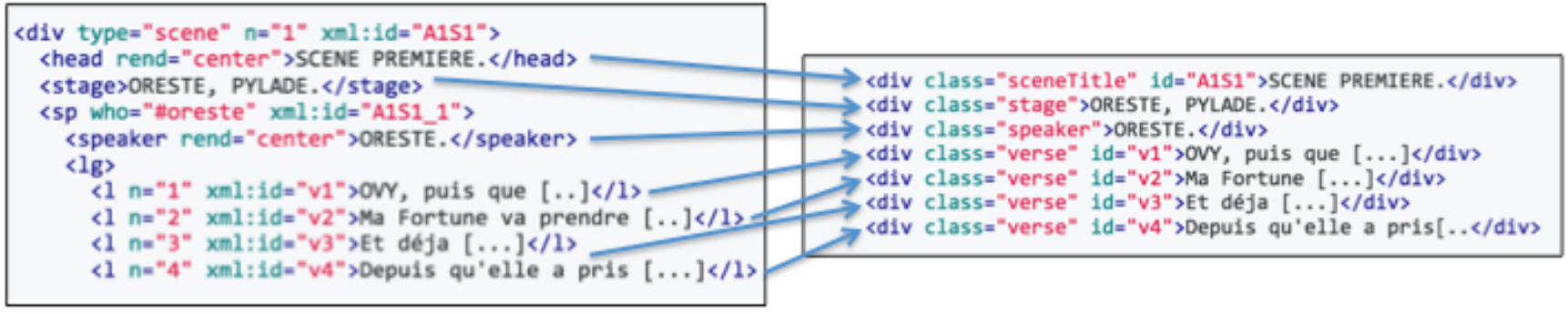
1- Écrire une nouvelle règle :

- Récupérer la valeur de la balise `<head>` contenue dans la `div[@type='play']`
- Englober le texte ainsi récupéré dans une balise HTML `<h1>`

2- Écrire une nouvelle règle pour traiter l'élément `div[@type='act']`

- Récupérer la valeur de la balise `<head>` contenue dans la `div[@type='act']`
- Englober le texte ainsi récupéré dans une balise HTML `<div>`
- Ajouter sur la balise HTML `div` un attribut *id*
- Récupérer la valeur de @xml:id de `div[@type='act']` pour remplir la valeur de l'attribut HTML *id*.
- Le reste des règles définies pour les éléments enfants de `div[@type='act']` devra s'appliquer après l'élément div que nous venons de créer.

3- Écrire une règle pour l'élément `div[@type='scene']` afin d'effectuer la transformation suivante.



PS : l'ajout du nom du "speaker" et des vers seront traités à l'aide d'une autre règle.

4-Compléter le fichier HTML :

- Créer une règle pour récupérer le nom du speaker (voir la transformation ci-dessus)
- Créer une règle pour récupérer les vers (voir la transformation ci-dessus)
- Observer le texte des vers, quels problèmes identifiez-vous ?