Humanités numériques : structuration des données et des documents textuels

E. Rouquette

Cours 6 – 13 février 2024

Le schéma EAD pour la description archivistique : brève présentation

Encoding Archival Description

- un schéma XML qui implémente le modèle l'ISAD(G) – Norme générale et internationale de description archivistique – pour produire des instruments de recherche numériques
- s'inspire de la TEI

Implémentation de l'EAD en France

Dans les archives

- implémentation progressive
- traduction française du *Dictionnaire de balises EAD* par l'AFNOR (Association Française de Normalisation) en 2004

Implémentation de l'EAD en France

Dans les archives

- implémentation progressive
- traduction française du *Dictionnaire de balises EAD* par l'AFNOR (Association Française de Normalisation) en 2004

Dans les bibliothèques

- implémentation due à des décisions politiques, avec la volonté d'informatiser les catalogues de manuscrits : choix du comité de pilotage du format EAD en 2002
- *Guide des bonnes pratiques de l'EAD en bibliothèque* en 2008
- Description des manuscrits et fonds d'archives modernes et contemporains en bibliothèque (DeMarch) en 2010.
  DeMarch : « recommandation AFNOR dont l'objet est de donner des règles permettant la description de manuscrits ou de fonds d'archives conservés dans les bibliothèques »
  `https://www.bnf.fr/fr/description-des-manuscrits-et-fonds-darchives-modernes`

Objectifs :

- Numérisations des documents : accès distant, valorisation des collections, nouvelles fonctionnalités
- Interopérabilité entre les différents catalogues et bases

Utilisation de l'EAD

- Rétroconversion d'instruments de recherche dans les archives
- Production d'IR nativement numériques dans les archives

Certaines notices bibliographiques sont structurées en TEI ; par exemple `http://www.europeanaregia.eu/fr/manuscrits/geneve-bibliotheque-geneve-ms-fr-178/fr`

Utilisation de l'EAD

Notices de manuscrits et d'archives en bibliothèques :

- BnF Archives et Manuscrits

  {BnF Archives et manuscrits

- catalogue en ligne des archives et manuscrits de l'Enseignement supérieur (Calames)

  calames

- sous-domaine Manuscrits du CCFr (Catalogue collectif de France)

  CCfr

→ Environ deux tiers des notices sont encodées en EAD

Schéma de l'EAD

- La DTD EAD possède 146 éléments dont 8 sont obligatoires.
  $\rightarrow$ Il s'agit donc d'un schéma qu'il faut lier aux documents XML.
- Le schéma est téléchargeable sur le site de l'EAD :
  `http://www.loc.gov/ead/ead2002a.html`
- La liste des éléments et attributs valides, ainsi que des valeurs conseillées sont décrites dans le dictionnaire des balises :
  `https://francearchives.fr/file/0def64f5a10f3f1ae03fdea59399a3e0755ef157/static_1066.pdf`

Structure d'un document EAD

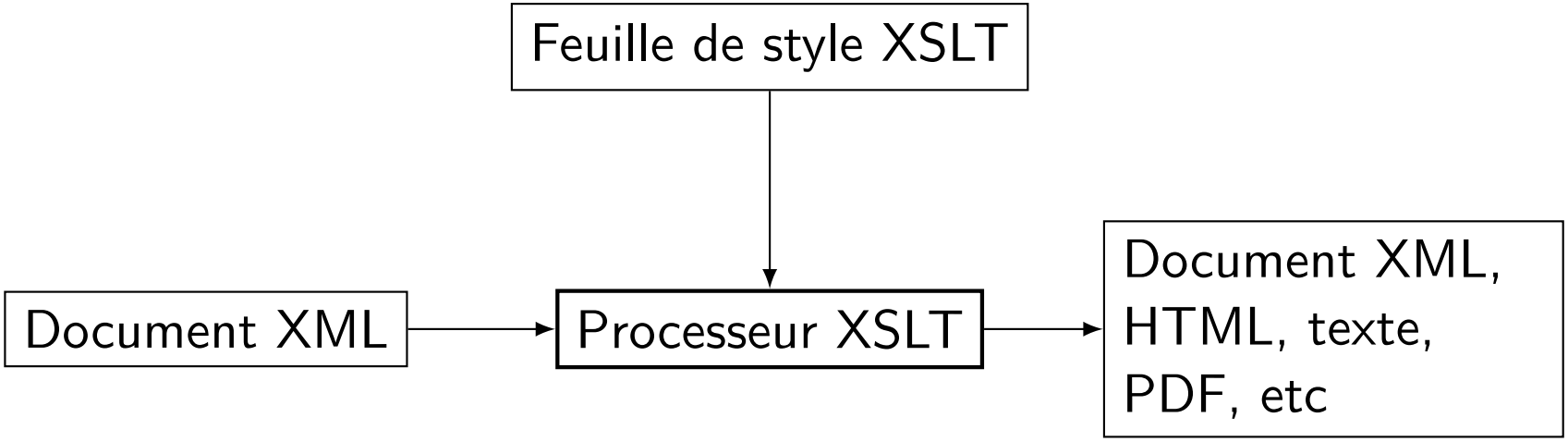
```
<ead>
 <eadheader>
  <eadid>identifiant du document numérique (s'il existe)</e
  <filedesc>
   <titlestmt>
    <titleproper></titleproper>
   </titlestmt>
  </filedesc>
 </eadheader>
 <archdesc level="fonds">
  <did>
  <unitid>identifiant de l'unité documentaire</unitid>
  </did>
 </archdesc>
</ead>
```

Exemples

- Archives nationales
- BnF Manuscrit
- Calame

XSLT : extraire et formater les informations

La transformation d'un document XML



Principes du langage XSLT

- Une feuille XSLT est un document XML contenant des balises spécifiques du type `<xsl:XXX>`
- Ces balises indiquent quels éléments sélectionner, et quel traitement effectuer sur ces éléments
- à partir des éléments sélectionnés, on peut :
  1. produire du texte
  2. produire de nouvelles balises
  3. reproduire des informations contenues dans des sous-éléments

Principes du langage XSLT

- XSLT établit des règles (`<xsl:template>`) sur les nœuds du document source XML
- chacune des règles provoque un effet sur le(s) nœud(s) appelé(s)
- un élément pour lequel aucune règle n'est définie produira son contenu textuel

$\rightarrow$ il faut bien connaître la structure du document pour pouvoir rédiger une XSL et le transformer

Dans une règle (template), on peut :

- Insérer du texte (tel quel)
- Insérer des balises (telles quelles)
- Insérer le contenu de sous-éléments avec :
  `<xsl:value-of select="sous-élément"/>`

Exemple court

```
1  <xsl:template match="p">
2   <xsl:apply-templates/>
3   <br/>
4  </xsl:template>
```

Explications :

1. Sur tous les éléments `<p>`
2. appliquer les règles (si pas de règles sur les nœuds enfants : imprime le texte qui se trouve à l'intérieur de l'élément concerné)
3. puis mettre l'élément HTML `<br/>` qui fait un retour à la ligne

Exemple court

```
1  <xsl:template match="teiHeader">
2   <h1><xsl:apply-templates/></h1>
3  </xsl:template>
```

Explications :

1. Sur l'éléments `<teiHeader>` (et ses enfants)
2. Imprimer le contenu entre des balises `<h1>` (balise html mettant titre de niveau 1)

Présentation de XPath

XPath est la syntaxe (non XML) pour sélectionner un nœud ou ensemble de nœuds utilisée dans les règles de transformation XSLT.

Un noeud peut être :

elementName Element (balises et contenu)

@attributName Attribut

text() Texte (contenu dans un élément, sans sa balise)

comment() Contenu d'un commentaire

Pour sélectionner un nœud, **on indique son chemin** avec la syntaxe XPAth

Les chemins XPath

Expression d'un chemin

`node/childNode`

- Une expression de chemin correspond à une **séquence d'étapes** séparées par l'opérateur `/`
- Sans indication particulière, on progresse d'un élément parent vers un élément enfant.
- Chaque étape devient le nœud courant (nœud de contexte) pour l'étape suivante
- On peut « sauter des étapes » en remplaçant l'opérateur `/` par `//`

Les chemins XPath

Tester un chemin XPAth avec oXygen

Dans oXygen, ouvrir le fichier `solution_Montaigne.xml`. Taper les expressions XPath suivantes dans la fenêtre dédiée :

- `//div1/head/text()`
- `//p/@xml:id`

Chemin absolu / chemin relatif

- Un chemin de localisation absolu commence au nœud racine : il commence par une barre oblique `/`
- Les chemins de localisation relatifs commencent à un nœud de l'arborescence. Ils sont utilisés notamment au sein d'un autre chemin Xpath

`.` désigne le nœud courant

`..` désigne le nœud parent

`*` désigne n'importe quel nœud

Trois éléments pour créer un chemin :

1. `nœud` un nœud
2. `axe ::` un axe (direction pour la suite du chemin)
3. `[...]` un prédicat (un test, un filtre)

Trois éléments pour créer un chemin :

Exemple d'expressions associant nœud et prédicat :

- p[title] les éléments <p> qui contiennent un élément <title>
- p[3] le troisième élément <p>
- p[@xml :id] les éléments <p> qui contiennent un attribut @xml :id
- p[@att="xx"] les éléments <p> ayant un attribut @att qui a pour valeur "xx"
- p[last()] le dernier élément <p>

Tester dans oXygen dans le fichier `solution_Montaigne.xml` :

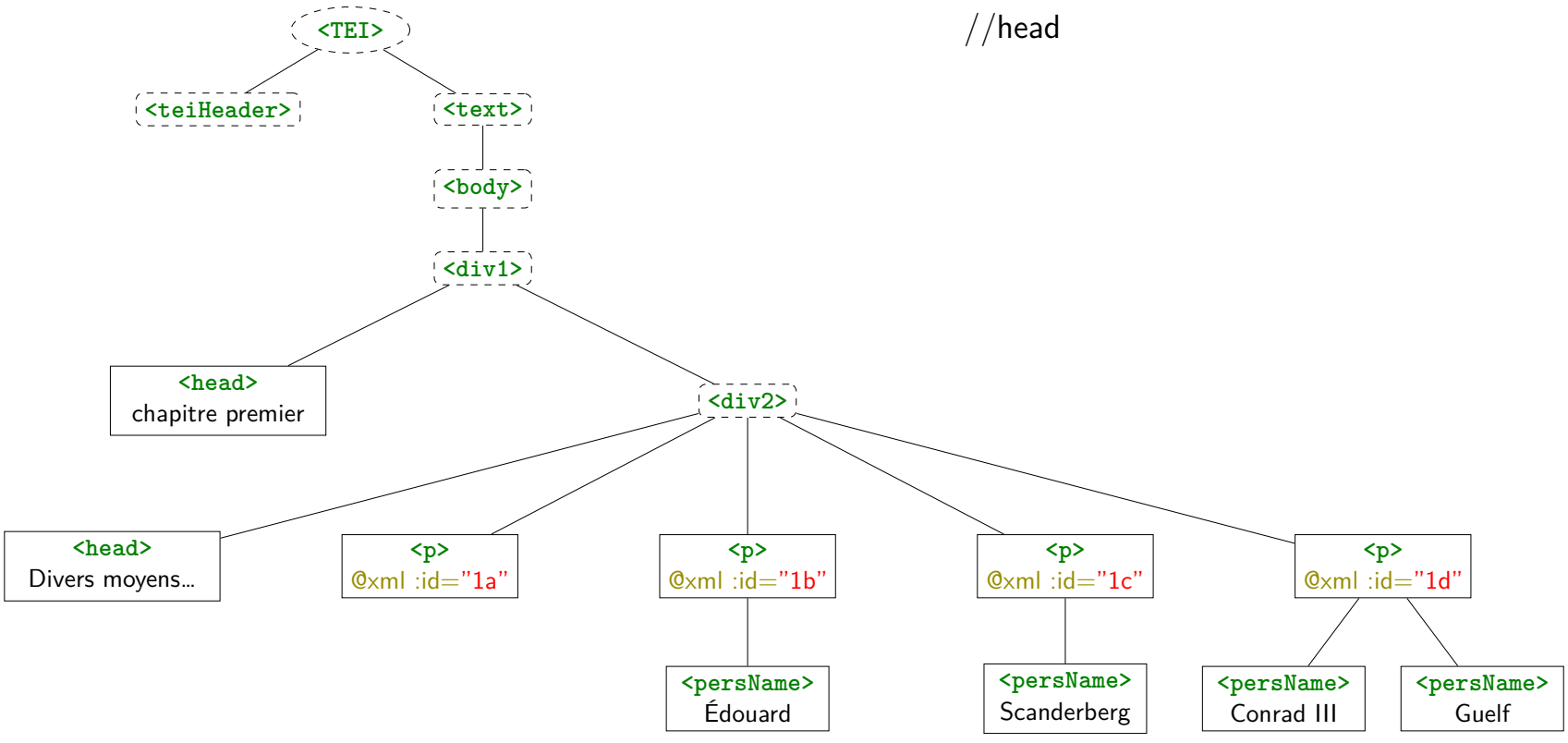
- `//p[3]`
- `//persName[@xml:id="edouard"]`

Exercice : Indiquer des chemins avec XPath (1)

Que sélectionnent les chemins suivants ?

Chemins absolus :

//head

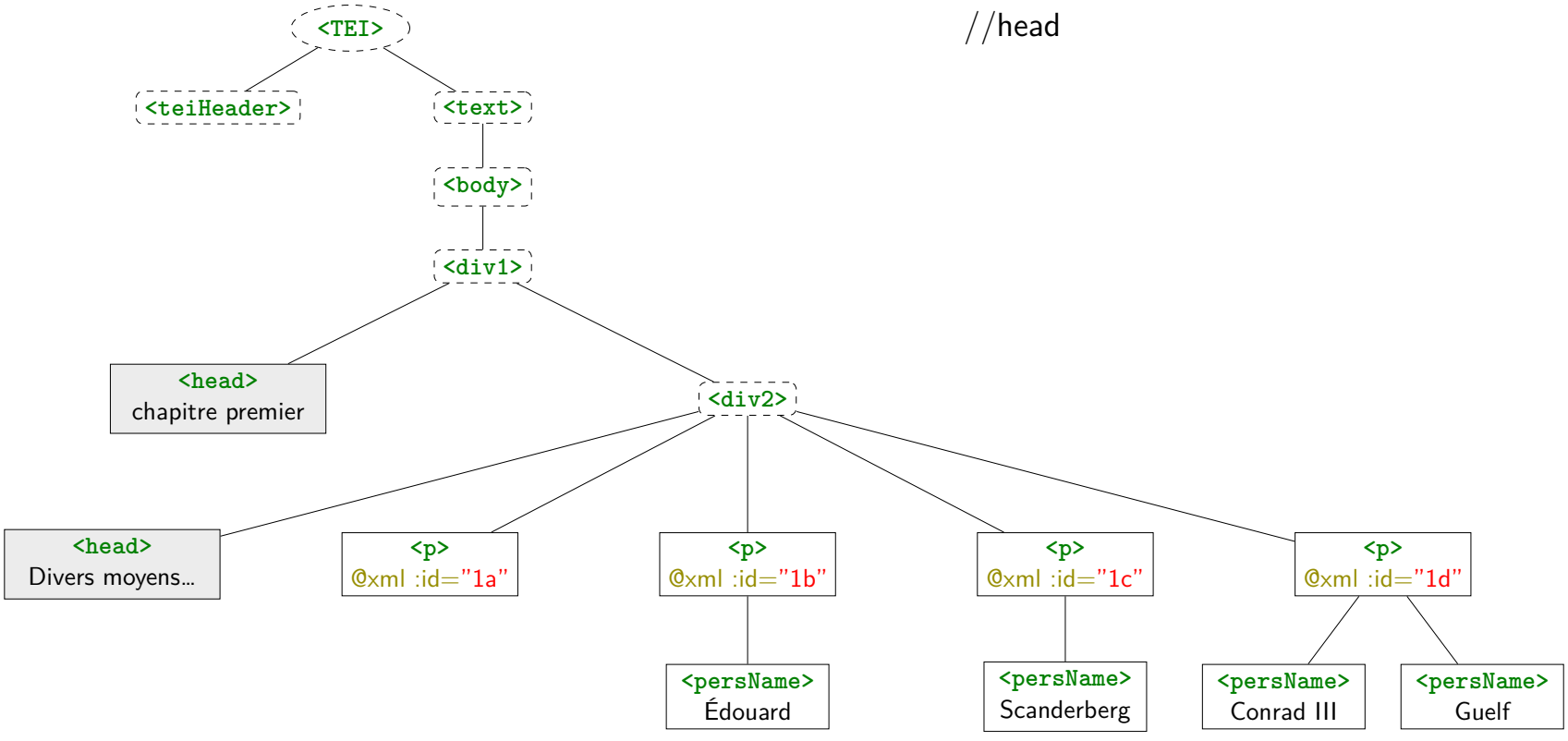


Exercice : Indiquer des chemins avec XPath (1)

Que sélectionnent les chemins suivants ?

Chemins absolus :

//head

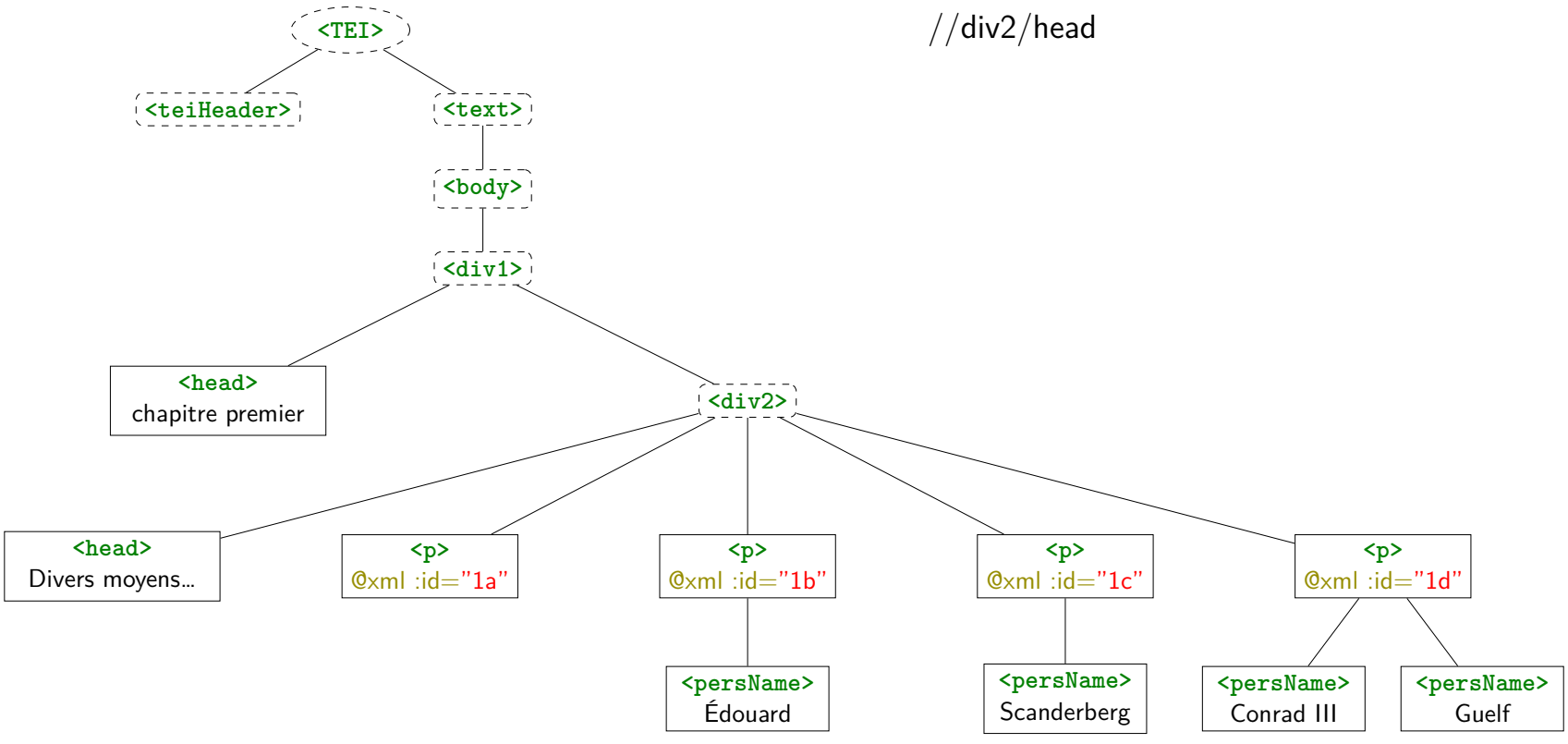


Exercice : Indiquer des chemins avec XPath (1)

Que sélectionnent les chemins suivants ?

Chemins absolus :

//div2/head

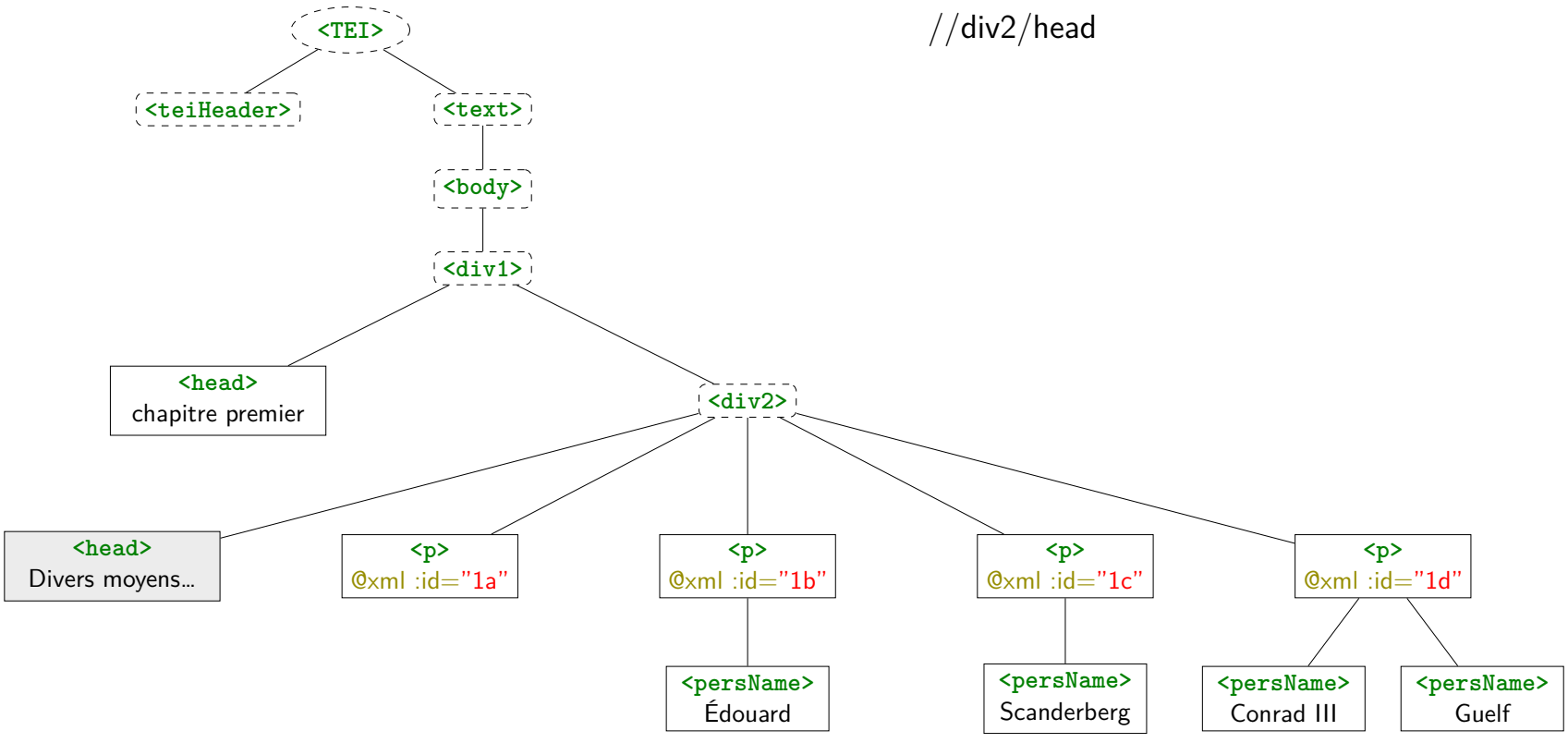


Exercice : Indiquer des chemins avec XPath (1)

Que sélectionnent les chemins suivants ?

Chemins absolus :

//div2/head

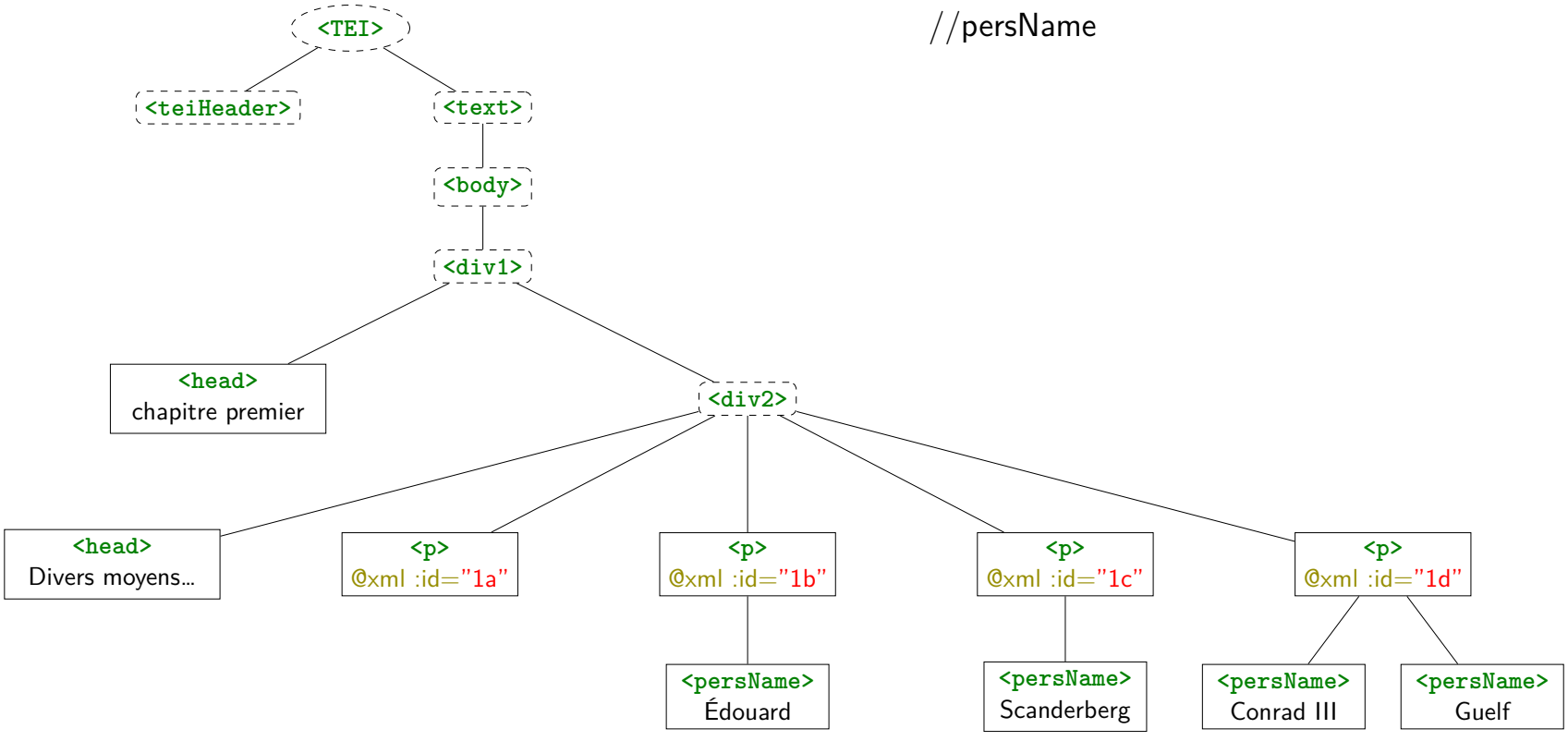


Exercice : Indiquer des chemins avec XPath (1)

Que sélectionnent les chemins suivants ?

Chemins absolus :

//persName

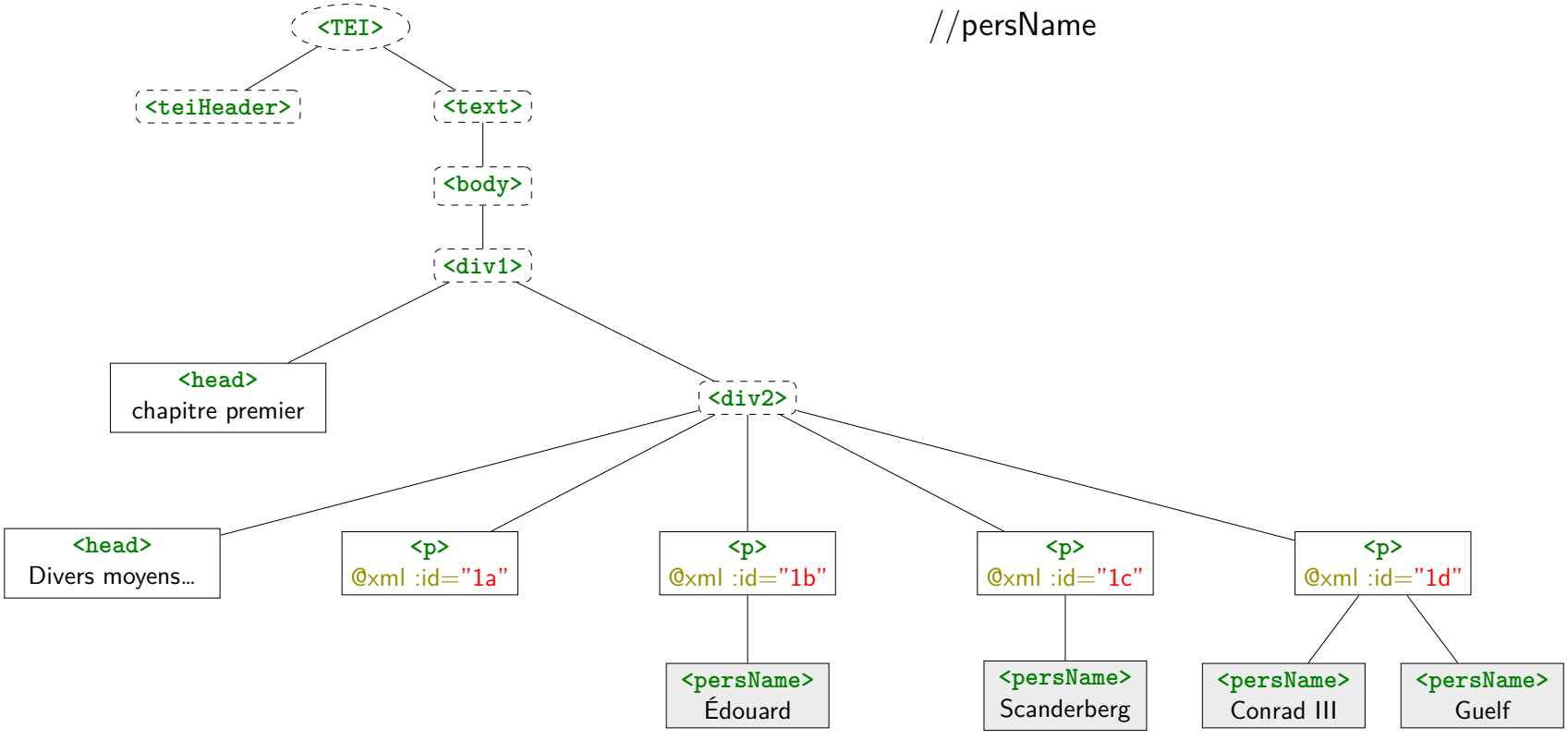


Exercice : Indiquer des chemins avec XPath (1)

Que sélectionnent les chemins suivants ?

Chemins absolus :

//persName

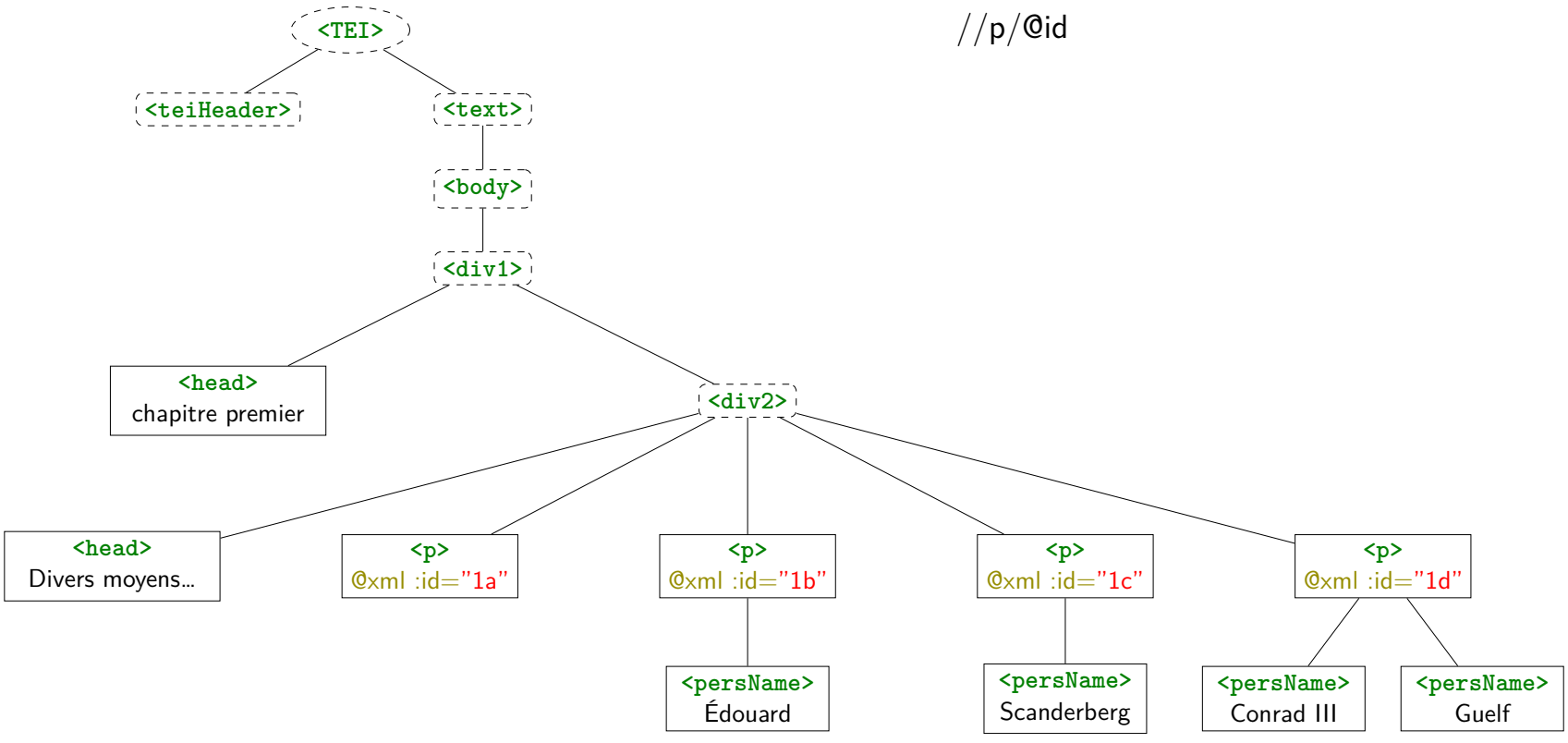


Exercice : Indiquer des chemins avec XPath (1)

Que sélectionnent les chemins suivants ?

Chemins absolus :

//p/@id

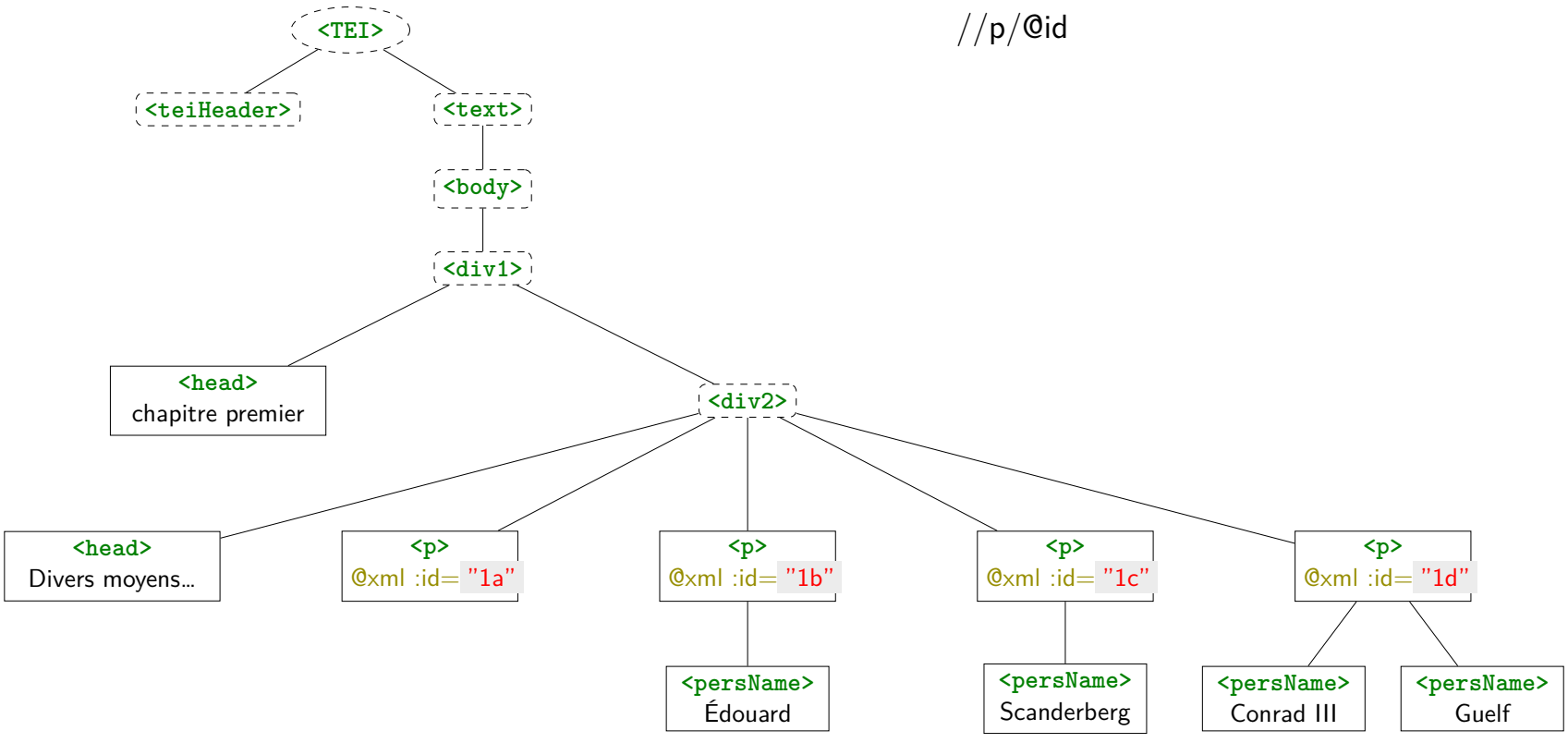


Exercice : Indiquer des chemins avec XPath (1)

Que sélectionnent les chemins suivants ?

Chemins absolus :

//p/@id

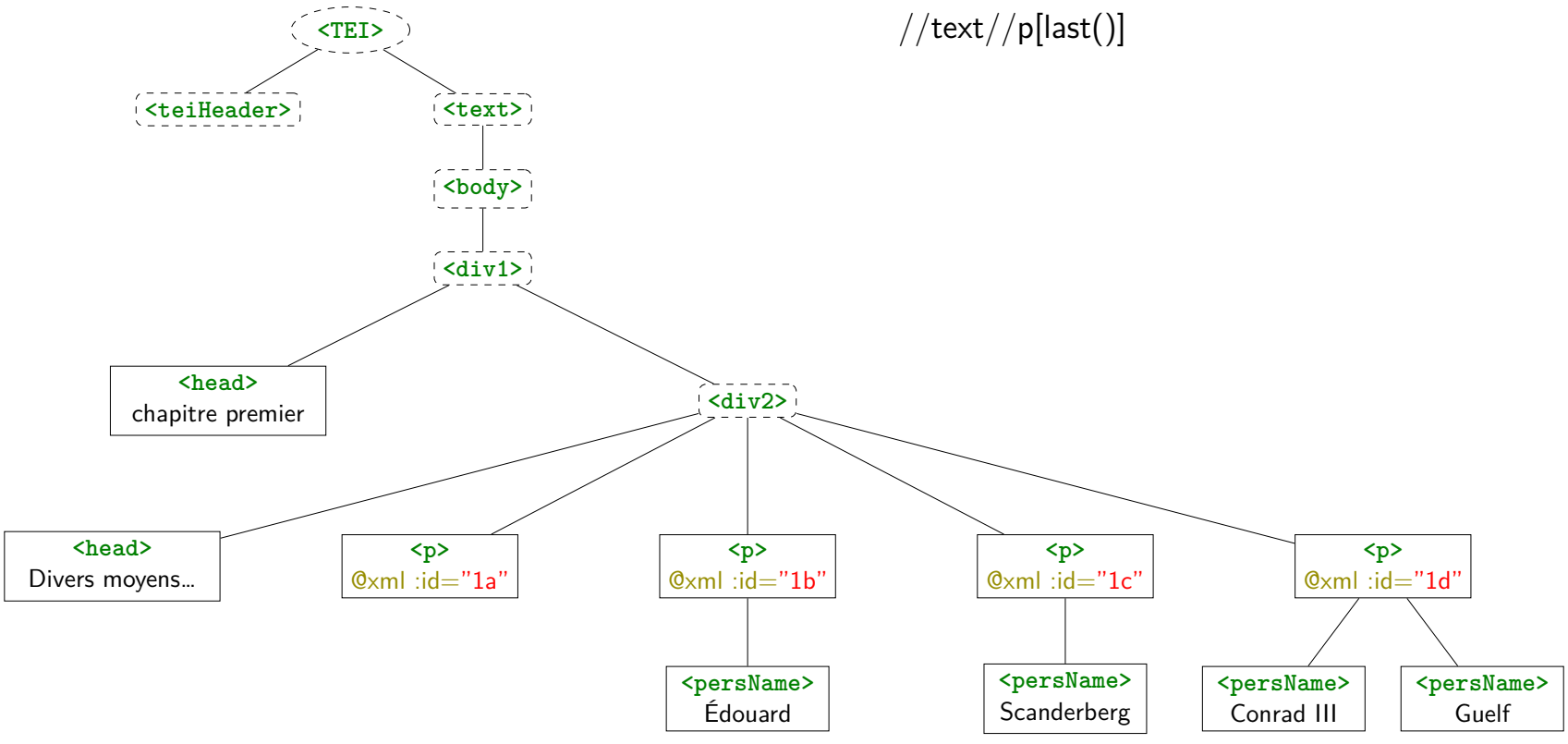


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins absolus :

//text//p[last()]

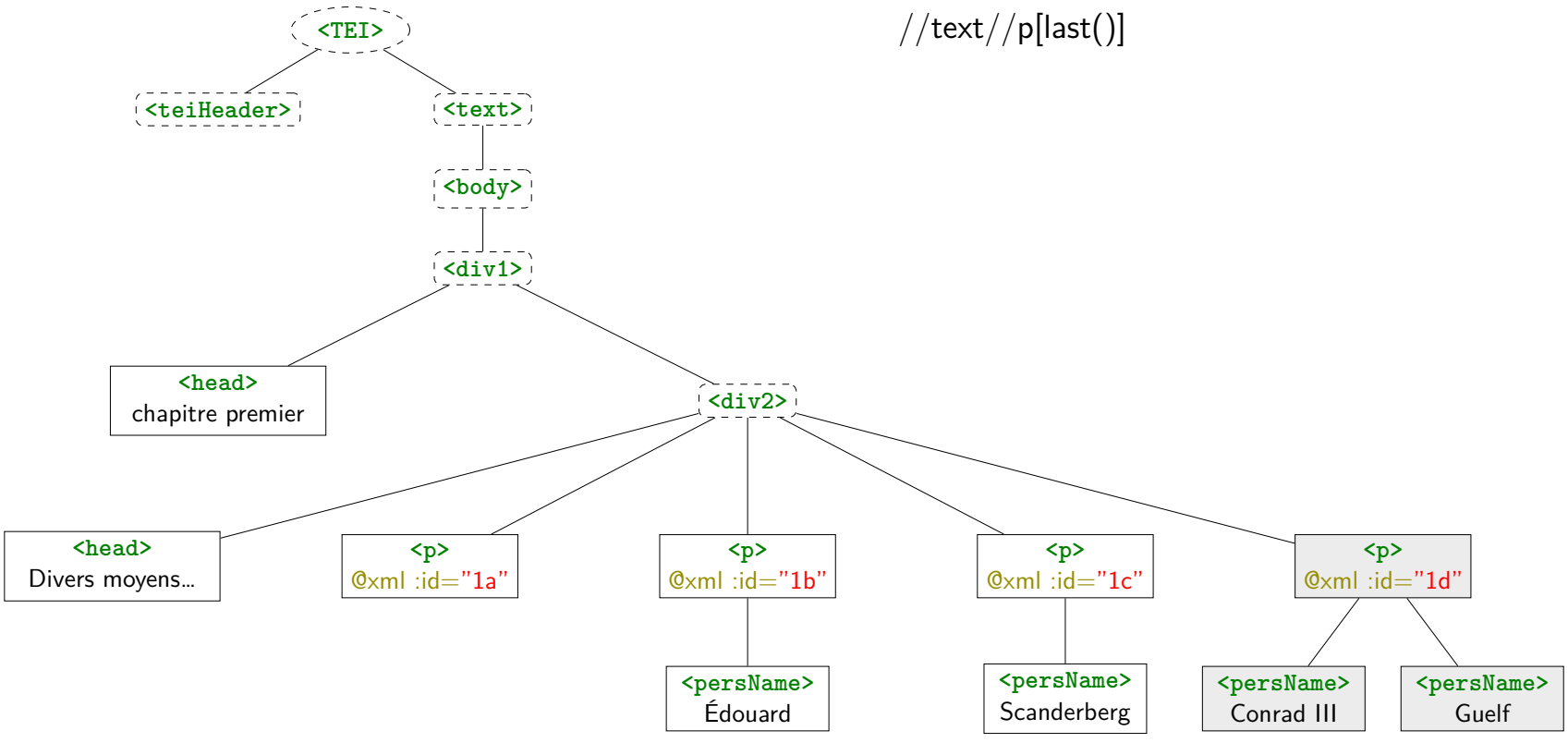


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins absolus :

//text//p[last()]

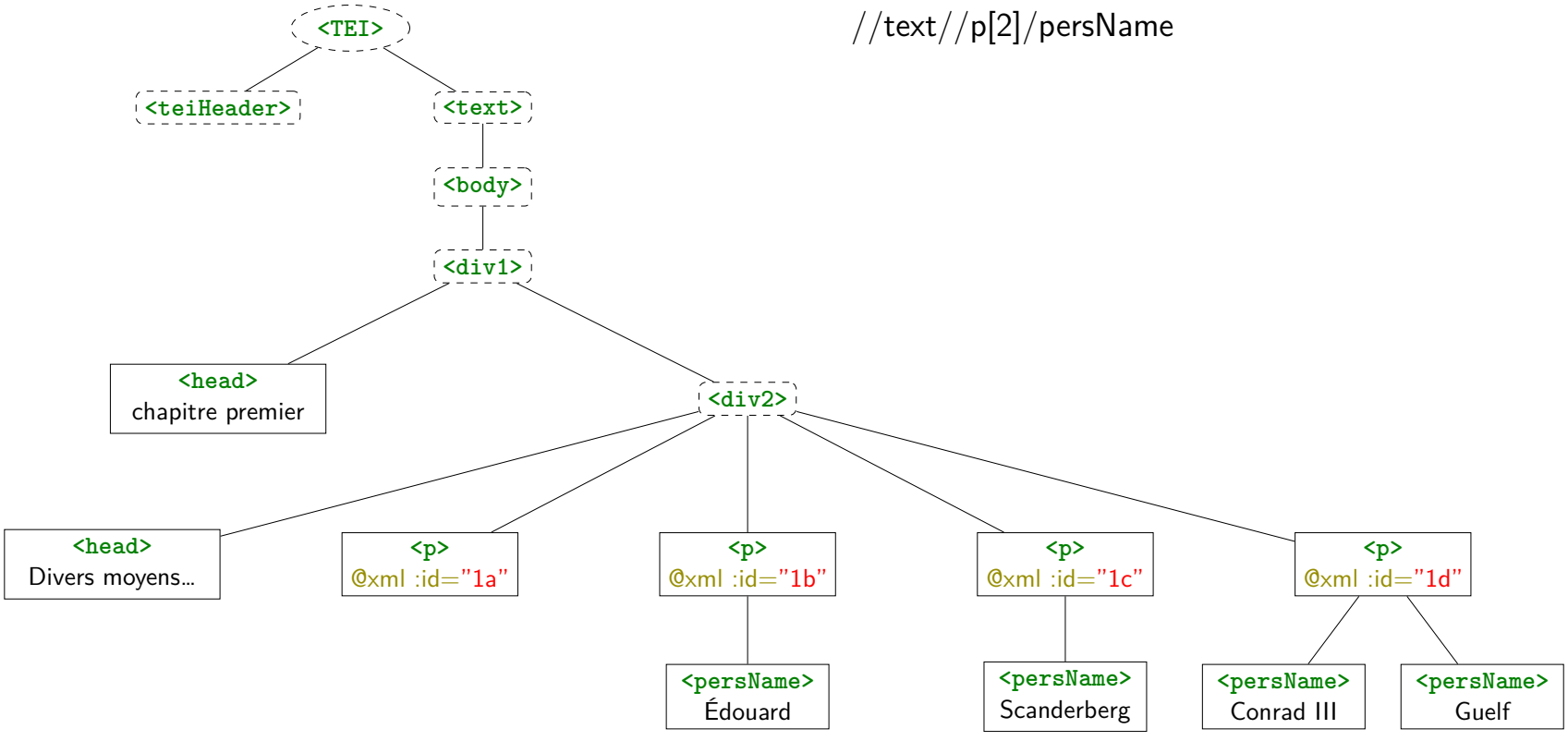


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins absolus :

//text//p[2]/persName

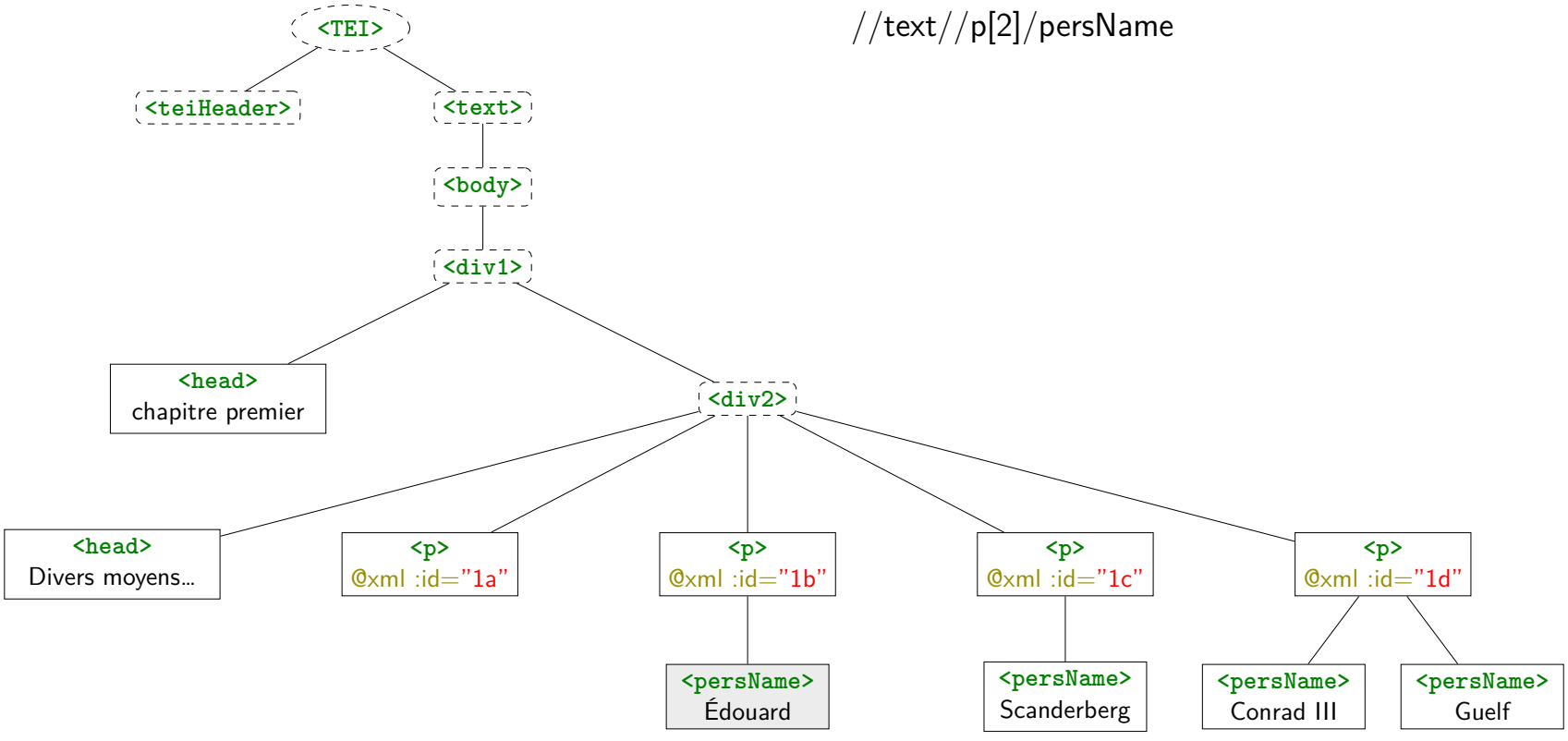


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins absolus :

//text//p[2]/persName

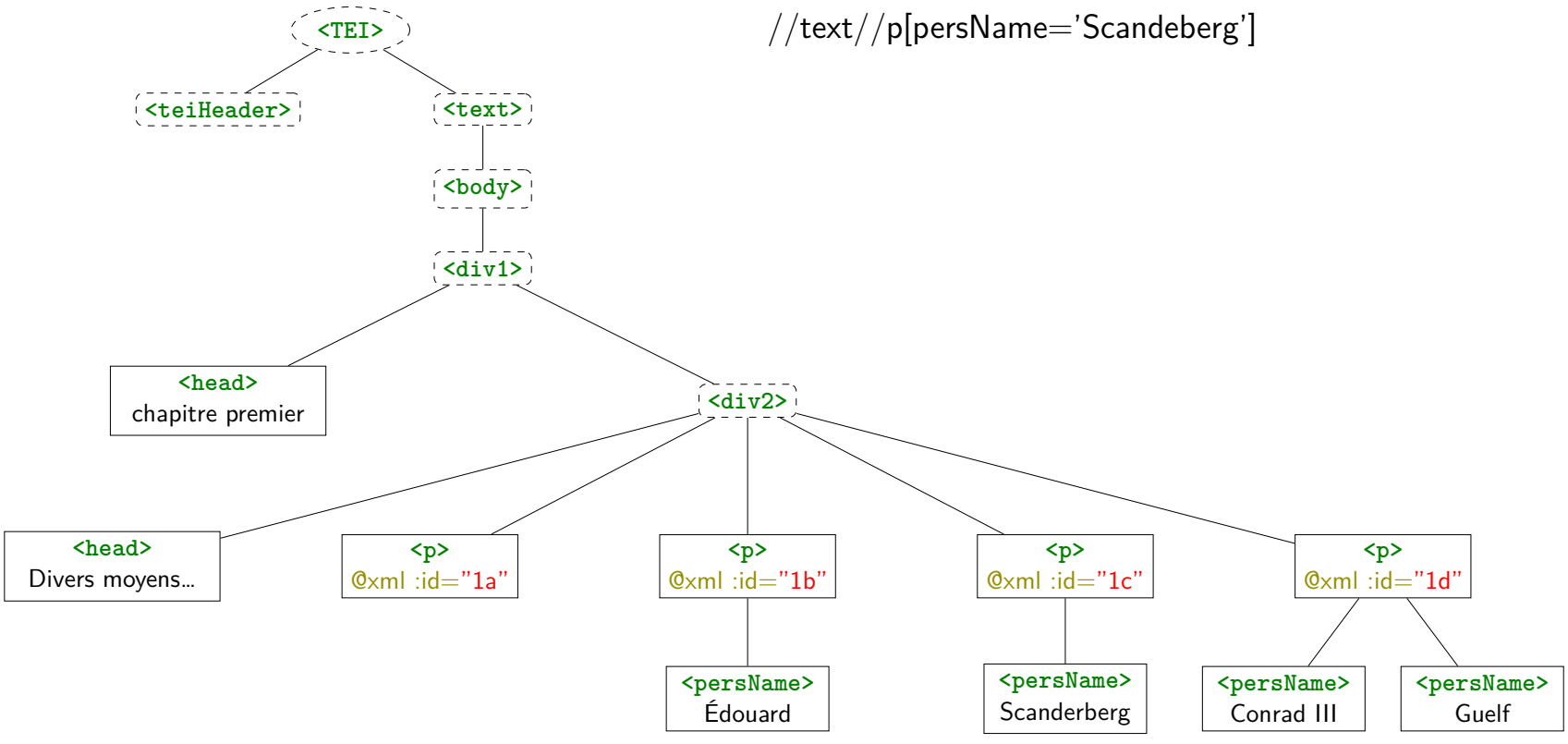


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins absolus :

//text//p[persName='Scandeberg']

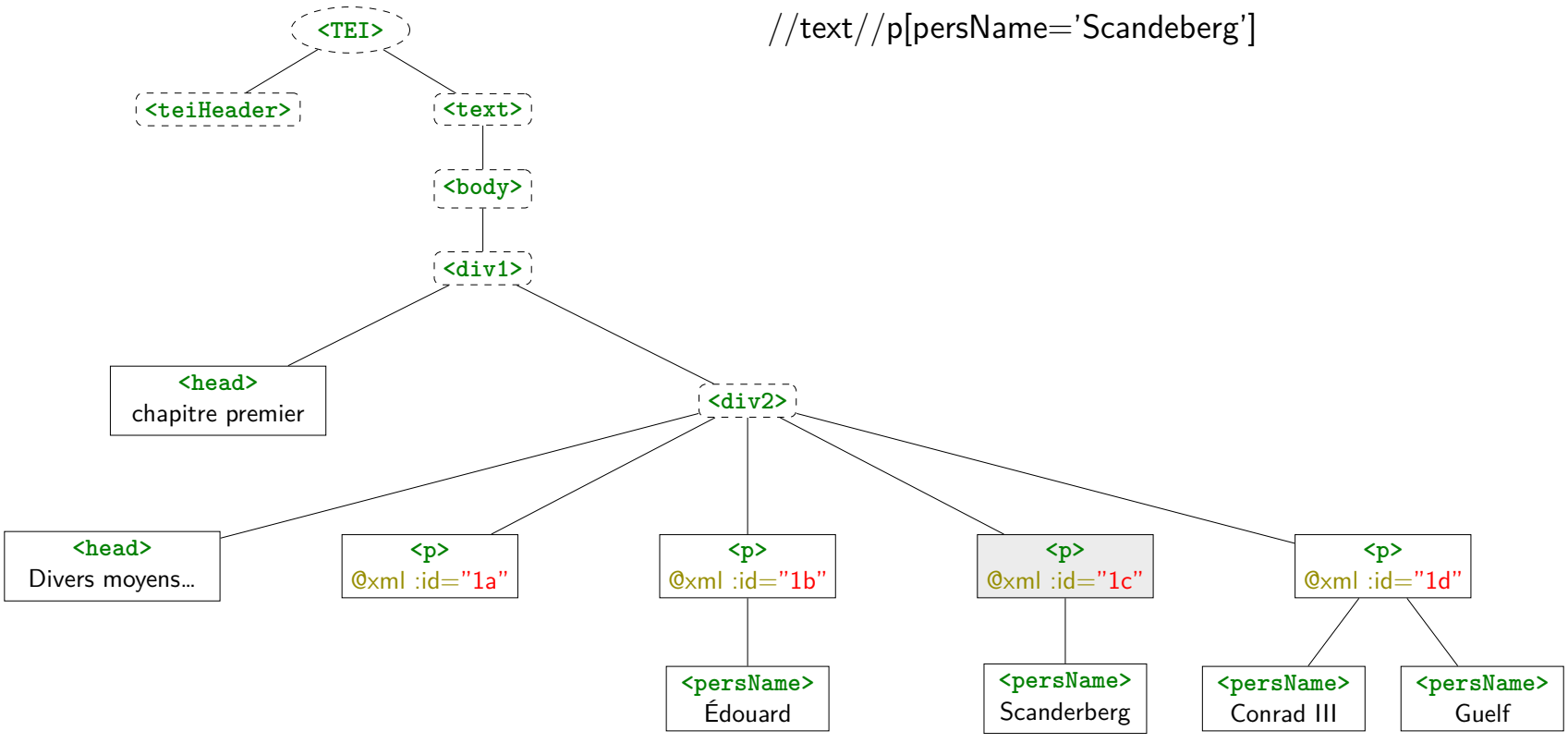


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins absolus :

//text//p[persName='Scandeberg']

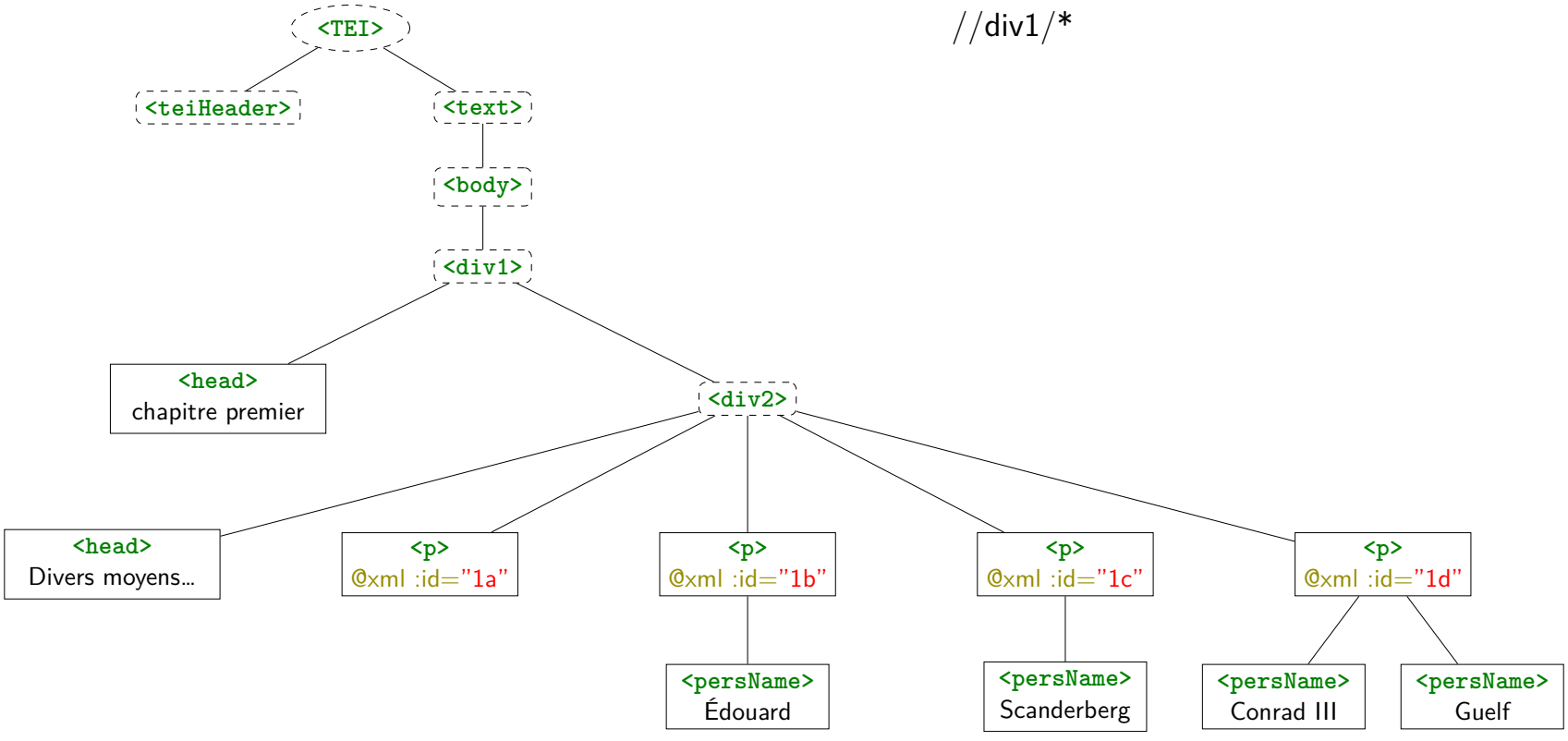


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins absolus :

//div1/*

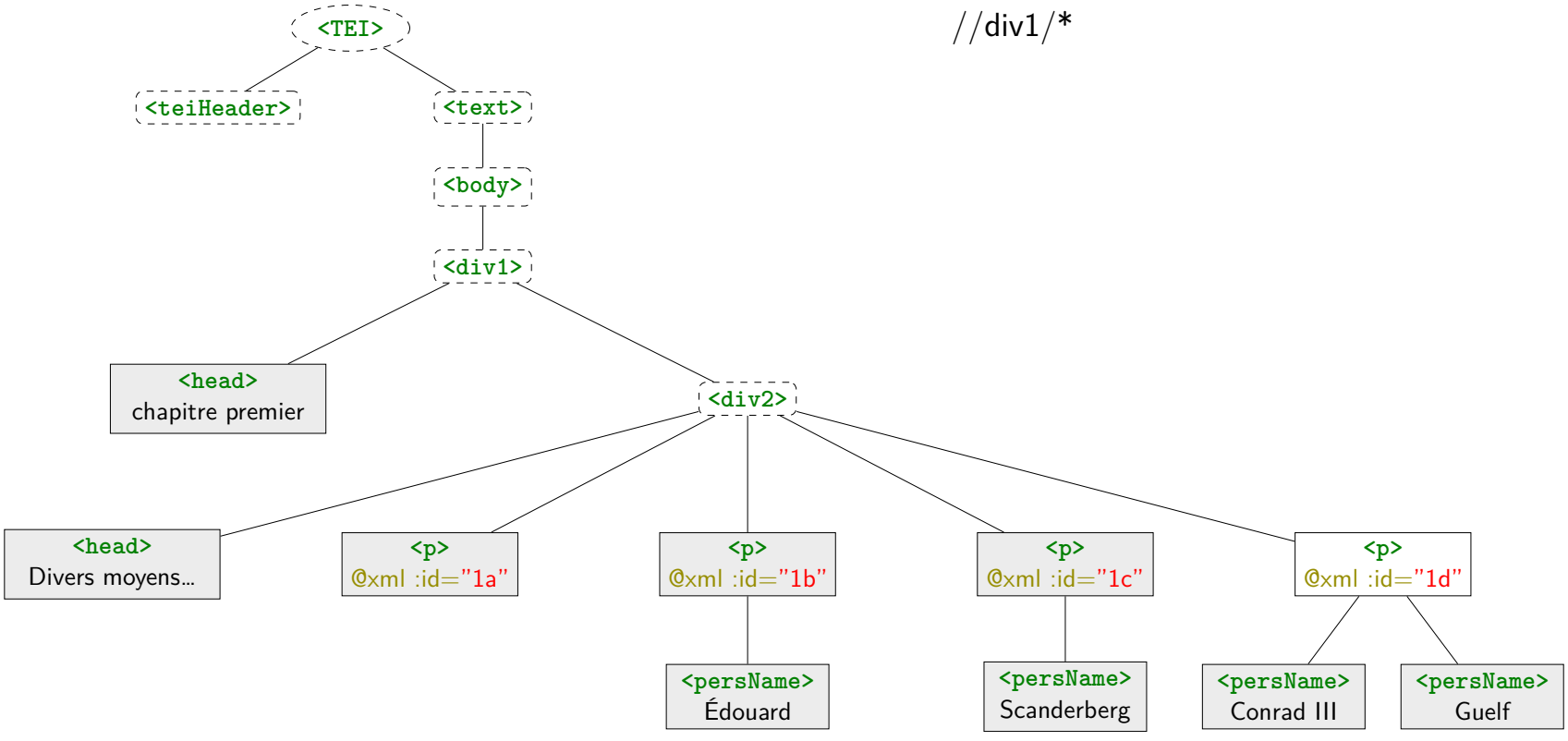


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins absolus :

//div1/*



Les axes

À partir d'un nœud, on peut indiquer une direction dans le chemin (un axe). S'il l'axe n'est pas précisé, il s'agit de `child ::` (les nœuds enfants). Quelques axes :

ancestor ::xx Sélectionne tous les ancêtres xx du nœud courant

attribute ::xx Sélectionne tous les attributs xx du nœud courant

descendant ::xx Sélectionne tous les descendants xx du nœud courant

following ::xx Sélectionne tous les nœuds du document après la balise fermante du nœud courant

following-sibling ::xx Sélectionne tous les nœuds xx suivants qui sont au même niveau que le nœud courant

parent ::xx Sélectionne les nœuds parent du nœud courant

preceding-sibling ::xx Sélectionne tous les nœuds xx précédents qui sont au même niveau que le nœud courant

Exemple : tester dans oXygen, dans le fichier `solution_Montaigne.xml` :

- `//persName/parent::p`
- `//p/descendant::placeName`
- `//p/descendant::text()`

Les fonctions

Les fonctions Xpath sont utilisées dans les prédicats (entre crochets) ou à la place d'un noeud. Elle permettent de manipuler les données à tester. Quelques fonctions :

count(x)	Compte le nombre d'occurrences de x
not(x)	teste l'absence de x
last()	Dernier nœud du document correspondant au prédicat

Exemple : tester dans oXygen, dans le fichier `solution_Montaigne.xml` :

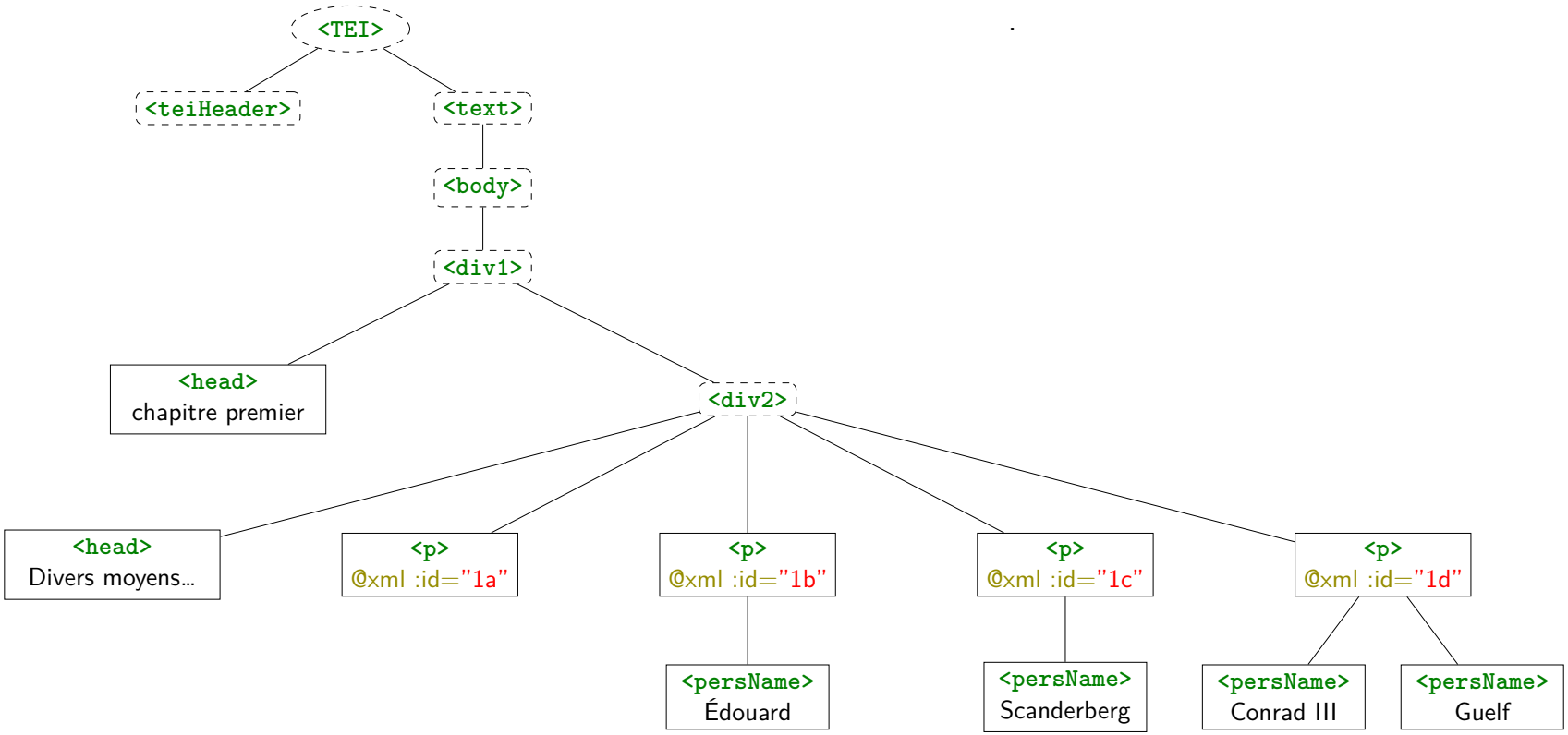
- `//p[2]/count(descendant::placeName)`
- `//p[not(persName)]`

Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//persName`

.

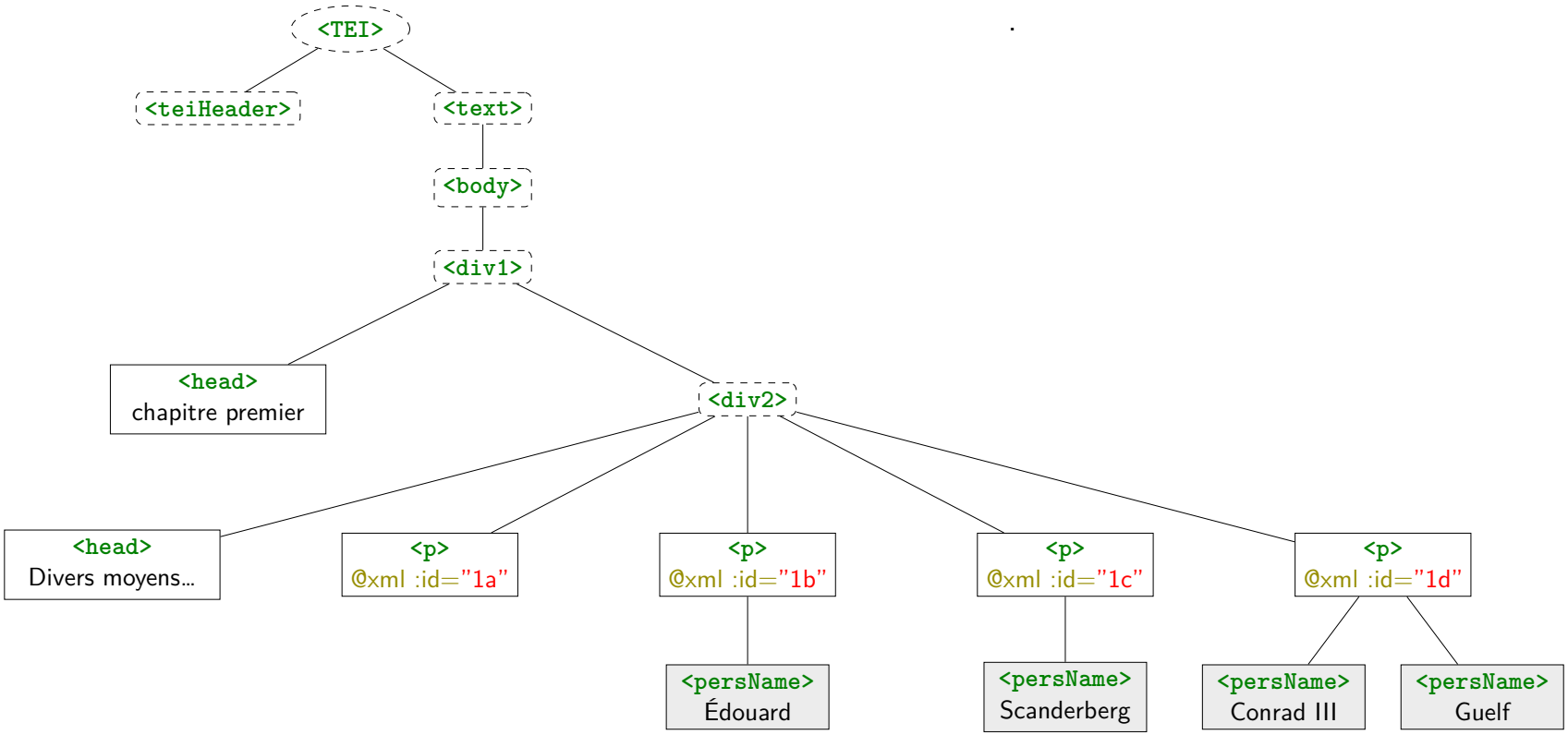


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//persName`

.

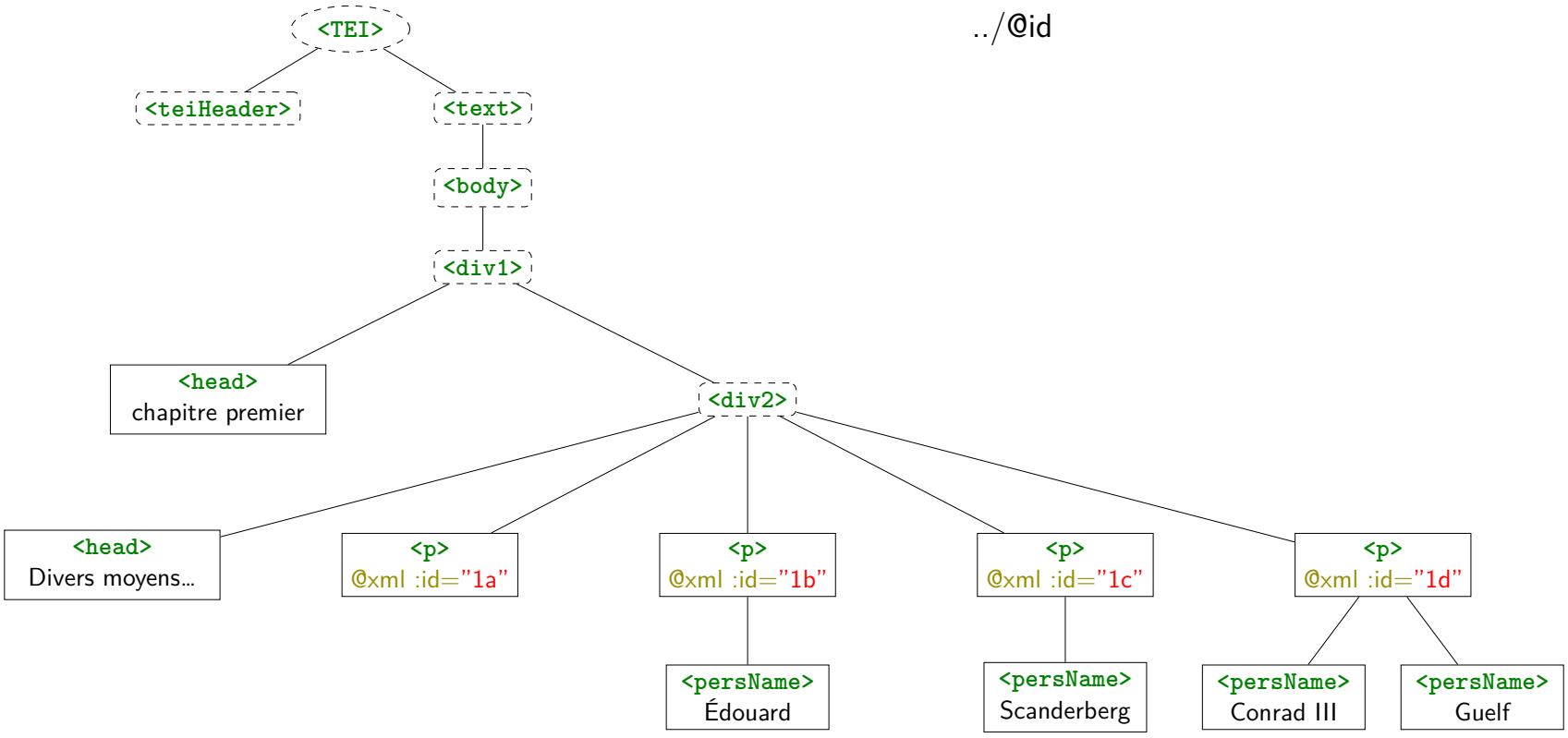


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//persName`

../@id

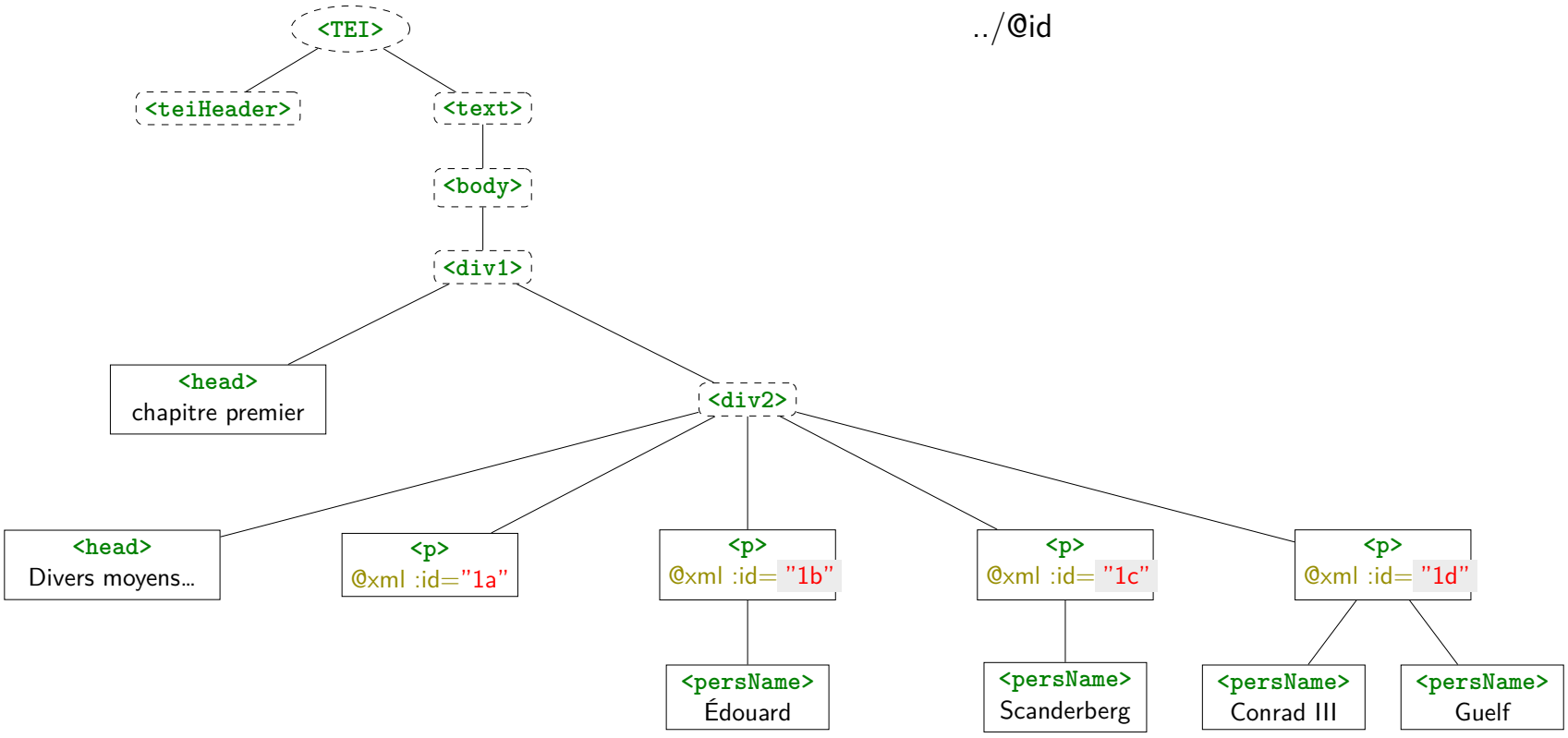


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//persName`

../@id

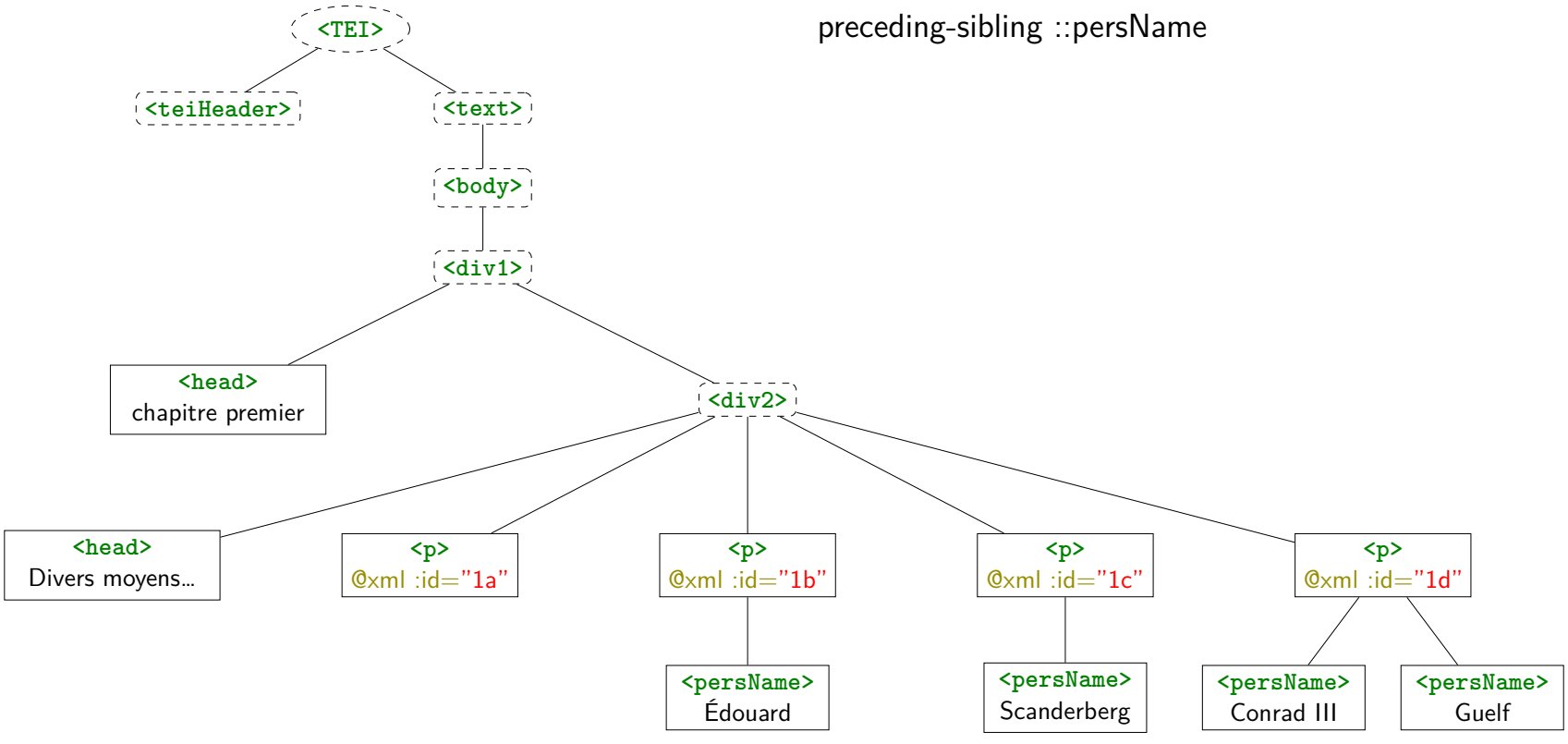


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//persName`

preceding-sibling ::persName

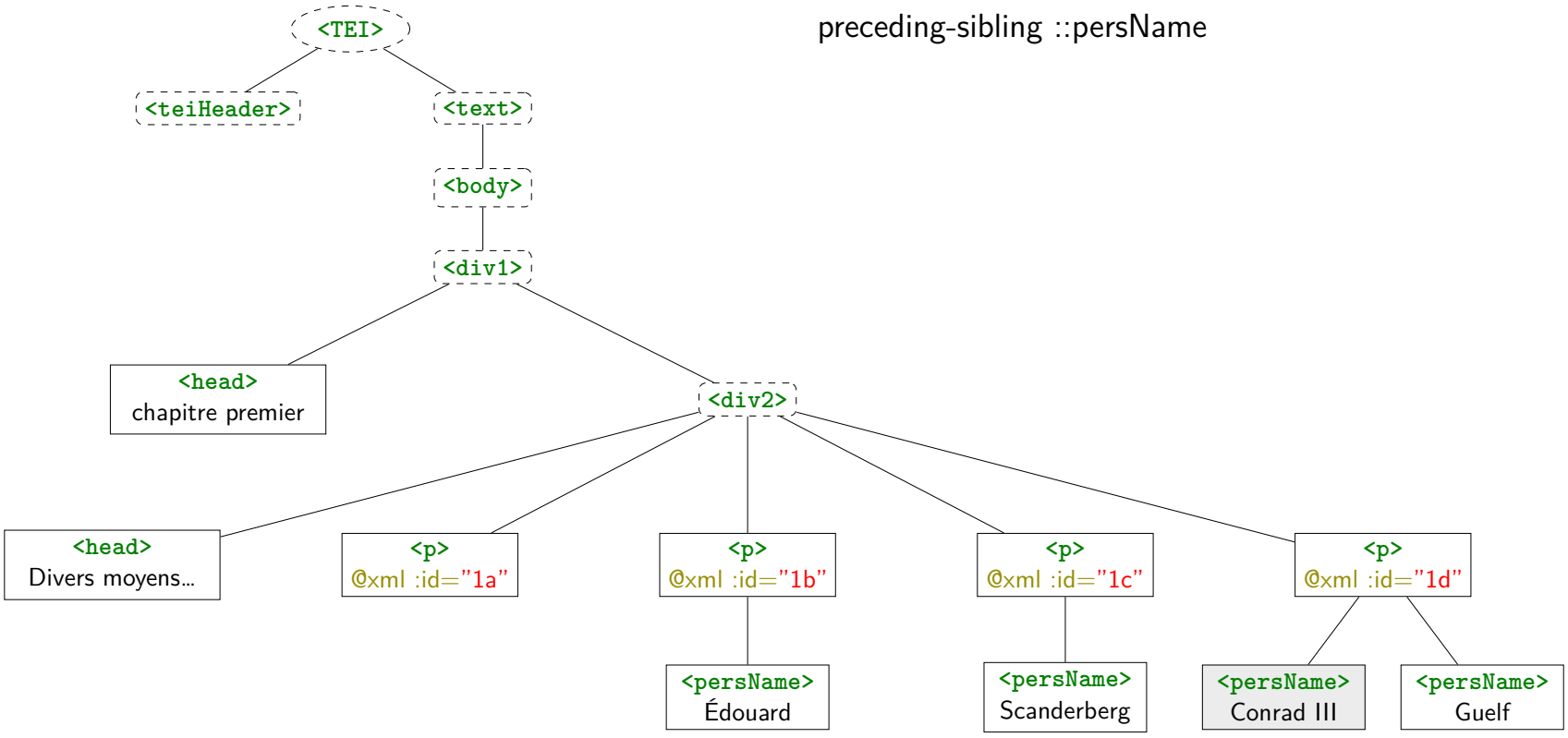


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//persName`

preceding-sibling ::persName

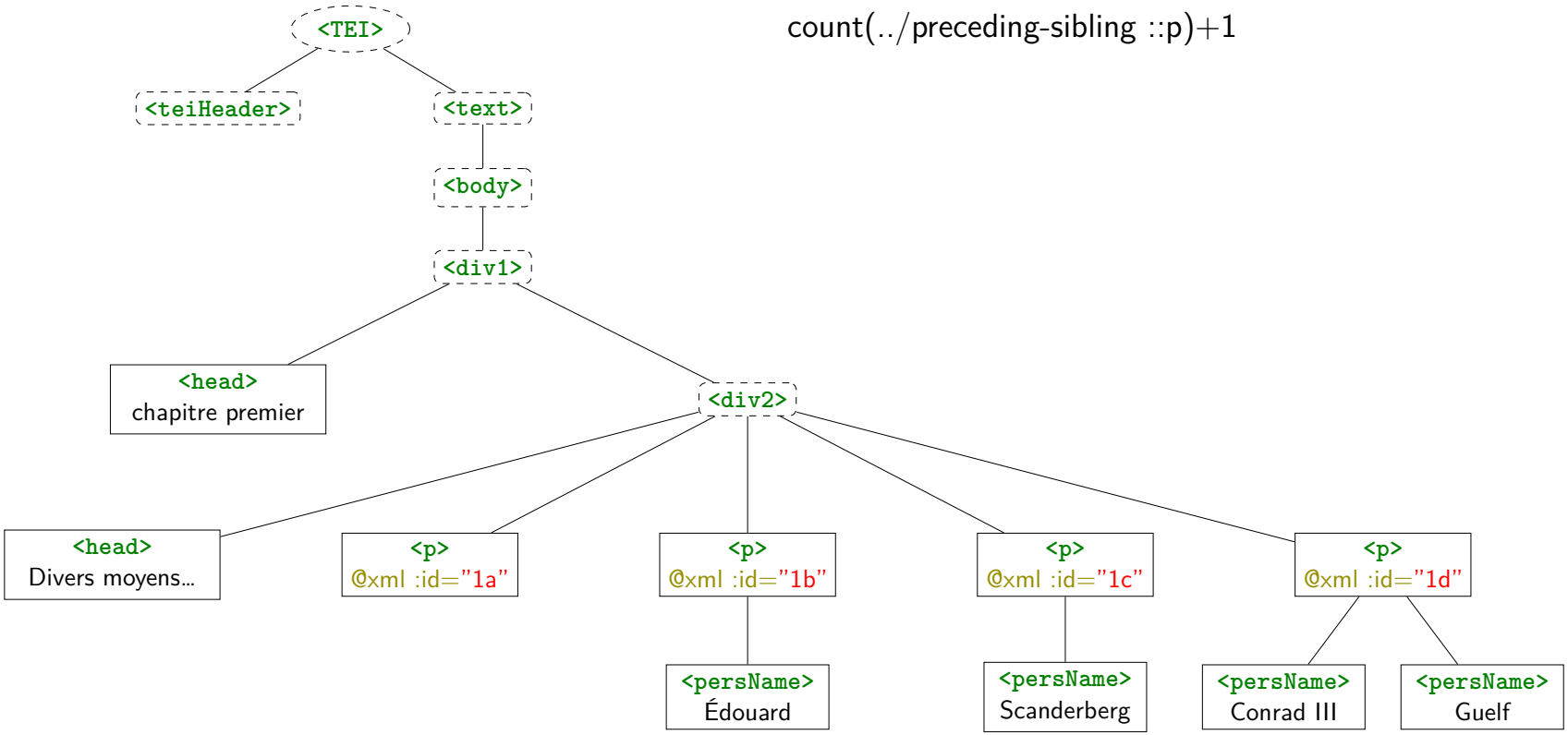


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//persName`

count(../preceding-sibling ::p)+1

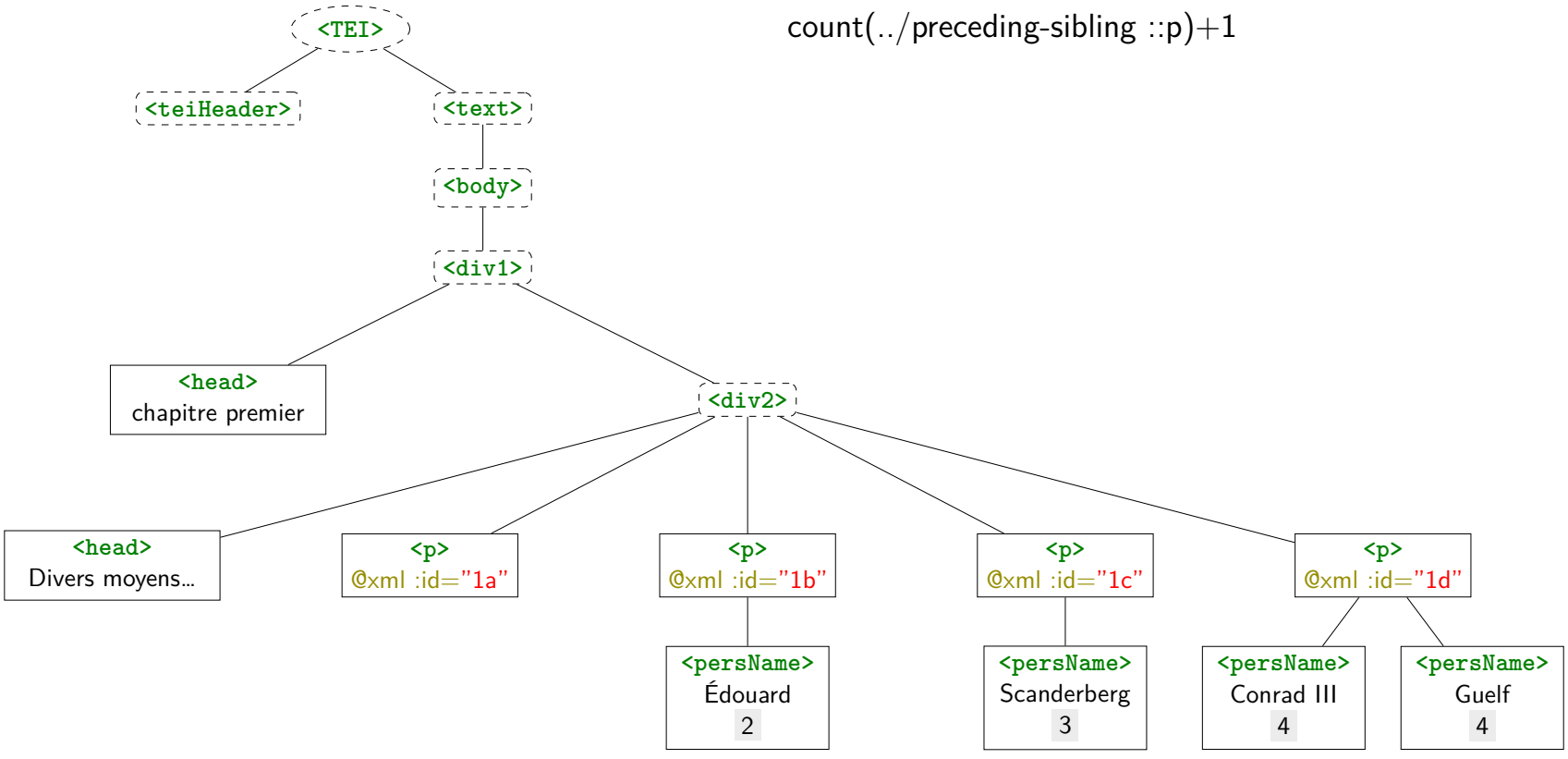


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//persName`

count(../preceding-sibling ::p)+1

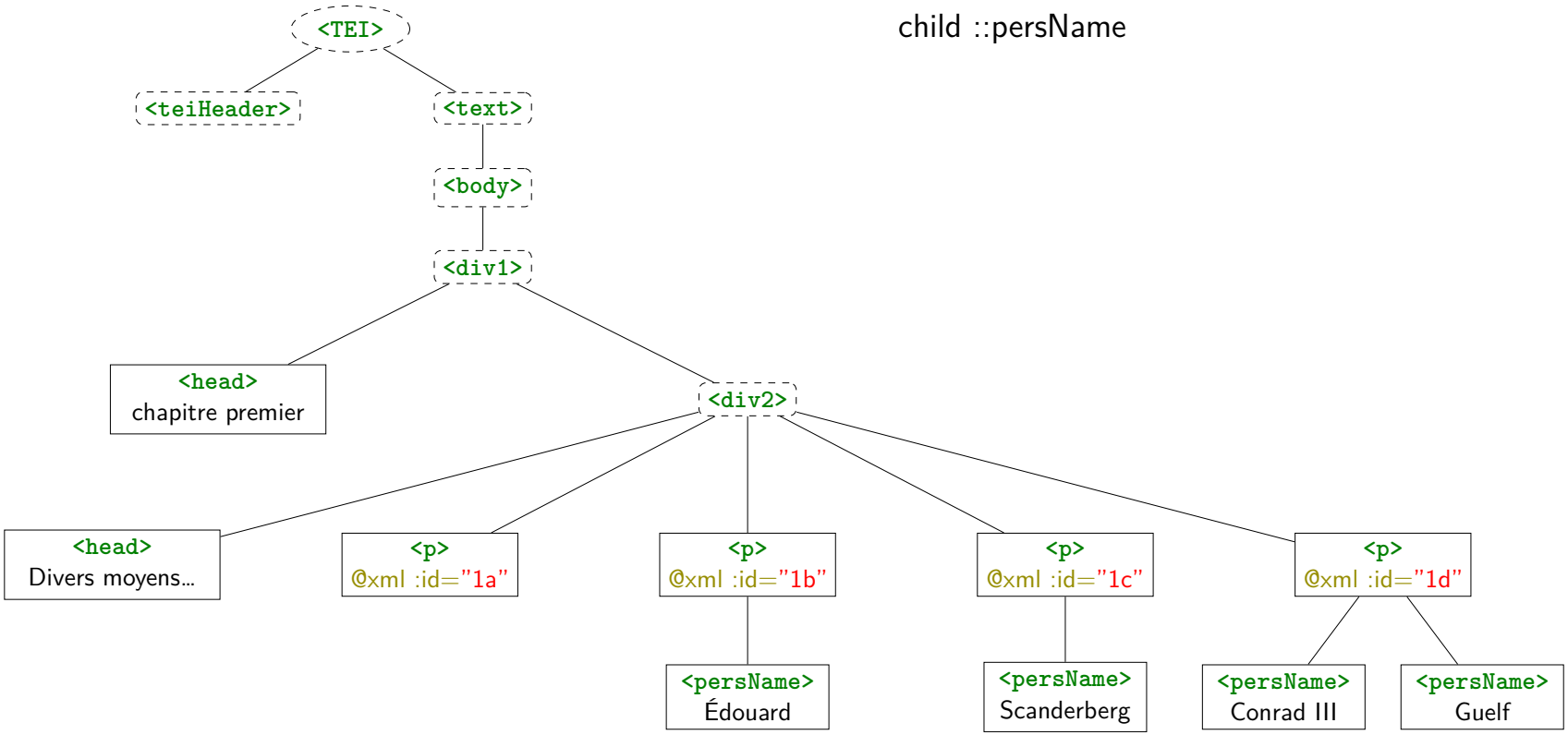


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//p[persName]`

child ::persName

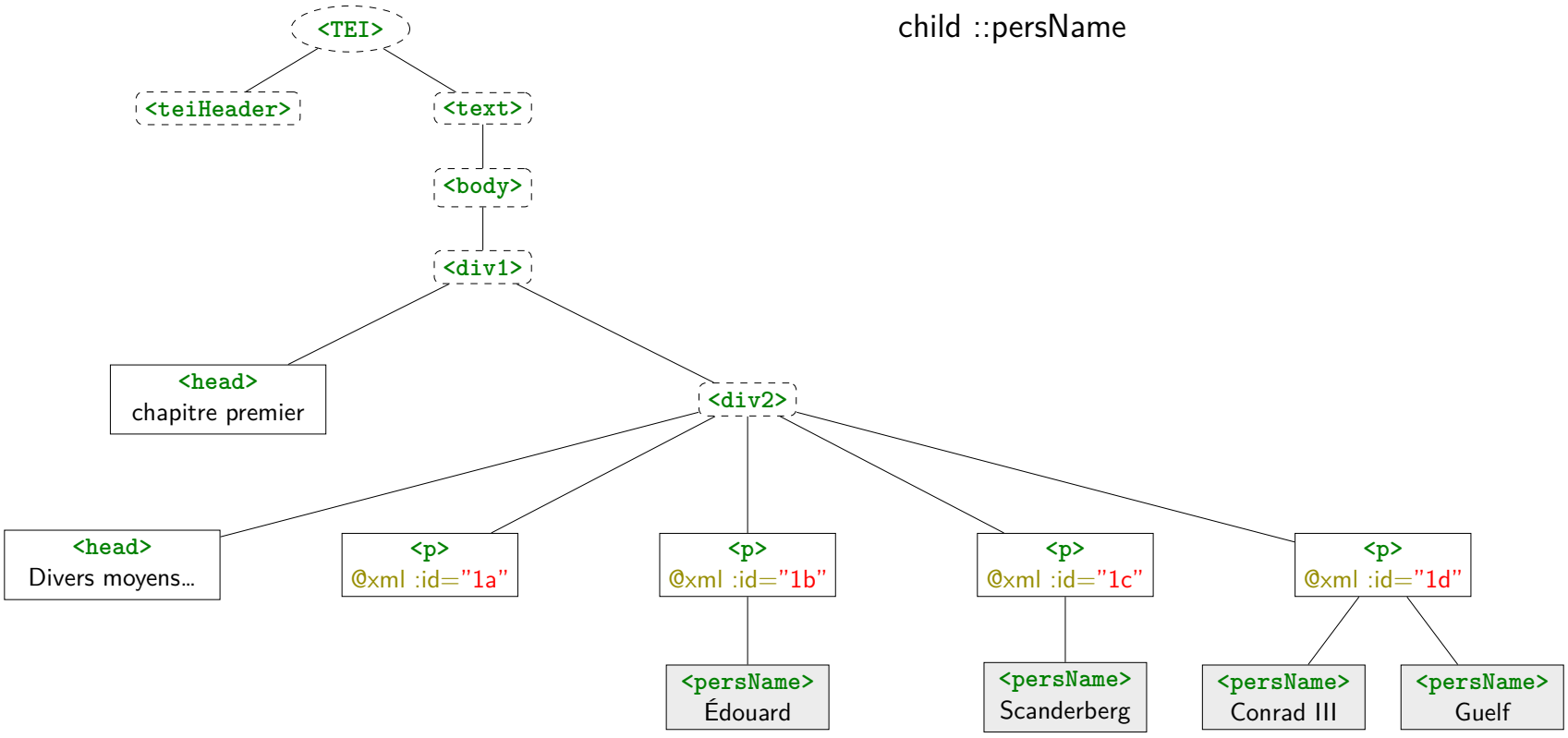


Indiquer des chemins avec XPath

Que sélectionnent les chemins suivants ?

Chemins relatifs – contexte : `//p[persName]`

child ::persName



Notre premier fichier XSLT

Deux méthodes pour créer une règle de transformation (template)

1. En indiquant sa portée

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

Explication : pour chaque élément `<p>`, mettre un retour à la ligne

nb : C'est cette méthode que nous allons utiliser ici

Deux méthodes pour créer une règle de transformation (template)

2. En lui donnant un nom et en l'appelant ensuite

```
<xsl:template name="nom">
 ... contenu du template
</xsl:template>

<xsl:template match="/>
 <xsl:call-template name="nom"/>
</xsl:template>
```

Créer/utiliser un fichier xslt avec oXygen

Pour créer un fichier xslt : Fichier - Nouveau - Nouveau document - XSLT

Créer/utiliser un fichier xslt avec oXygen

Pour cet exercice, nous utilisons le fichier `templates_flaubert.xsl`

- Télécharger le fichier
- Ouvrir le fichier `solution_Flaubert.xml` (ou votre propre encodage s'il est valide)
- Associer à ce fichier xml le fichier xsl : épingle verte, associer une feuille de style xslt/css

- Clef à molette : configurer le(s) scénarios de transformation : éditer - sortie - ouvrir dans l'éditeur Choisir : afficher dans la vue comme XML - XHTML

- tester la transformation en cliquant sur Appliquer les scenarios de transformation

Regardons l'élément racine :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Trans
3   xmlns:xs="http://www.w3.org/2001/XMLSchema"
4   xmlns:tei="http://www.tei-c.org/ns/1.0"
5   exclude-result-prefixes="xs tei"
6   version="2.0">
```

1. un fichier XSLT est un fichier XML
2. on appelle l'espace de noms XSL, les éléments auront le préfixe **:xsl**
3. on appelle l'espace de noms XMLSchéma, les éléments auront le préfixe **:xs**
4. on appelle l'espace de noms TEI : pour faire référence à des nœuds qui sont des éléments TEI, nous les préfixerons avec **:tei**
5. les préfixes xs et tei ne seront pas reportés dans le document résultant de la transformation
6. Il s'agit de la version 2.0 de XSLT

Regardons l'élément racine :

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Trans
3   xmlns:xs="http://www.w3.org/2001/XMLSchema"
4   xmlns:tei="http://www.tei-c.org/ns/1.0"
5   exclude-result-prefixes="xs tei"
6   version="2.0">
```

→ Par rapport à un fichier créé avec oXygen, nous avons ajouté l'espace de nom TEI et la valeur tei dans exclude-result-prefixes

Les templates de notre premier fichier : explication

```
6  <xsl:template match="//tei:form">
7   <b><xsl:apply-templates/></b>
8  </xsl:template>
```

Les éléments `<form>` sont mis dans la balise html `<b>`, qui dans un fichier html sert à mettre en gras

Les templates de notre premier fichier : explication

```
10  <xsl:template match="//tei:entry">
11   <xsl:apply-templates/><br/>
12  </xsl:template>
```

Après chaque entrée (`<entry>`), on insère un saut de ligne avec la balise html `<br/>`

Les templates de notre premier fichier : explication

```
15  <xsl:template match="/">
16   <html>
17    <head>
18     <title>Dictionnaire des idées reçues</title>
19    </head>
20    <body>
21      <xsl:apply-templates select="//tei:text">
22    </body>
23   </html>
24  </xsl:template>
```

- On met l'ensemble du texte de sortie (`match="/"`) dans un document html : pour obtenir un document valide, on met les balises obligatoires dans un tel document
- `<xsl:apply-templates select="//tei:text">` Dans le corps du document html, on applique les règles de transformation à l'élément `<text>` (et ses descendants) → ce qui permet d'exclure le contenu du `<teiHeader>`.

Les templates de notre premier fichier : explication

```
15  <xsl:template match="/">
16   <html>
17    <head>
18     <title>Dictionnaire des idées reçues</title>
19    </head>
20    <body>
21     <xsl:apply-templates select="//tei:text">
22    </body>
23  </html>
24  </xsl:template>
```

- Testez l'utilité de cette sélection avec `<xsl:apply-templates select="//tei:text">` en mettant `<xsl:apply-templates/>` à la place.

Exercice : manipuler les templates

En observant comment sont construits les templates appliqués aux éléments `<form>` et `<entry>`, ajouter un template permettant de mettre les épigraphes dans la balise `<blockquote>` ; tester le résultat

Exercice : manipuler les templates

Solution :

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

Exercice : Créer un tableau à partir d'un fichier XML

But de l'exercice : Mettre dans un tableau à deux colonnes la liste des personnes apparaissant dans le texte de Montaigne, et l'ID du paragraphe correspondant

Télécharger le fichier `templates_montaigne.xsl` ; déclarez-le comme fichier xslt pour le fichier `solution_Montaigne.xml`

Exercice : Créer un tableau à partir d'un fichier XML

- Dans le `<xsl:template match="/">` se trouve la balise `<table>` qui permet de créer un tableau en html
- Le template `<xsl:template match="tei:persName">` contient `<td>`, qui crée une ligne dans un tableau, et `<tr>`, qui crée une cellule dans une ligne de tableau
- l'élément `<xsl:value-of select="XXX"/>` imprime la valeur du noeud sélectionné

Exercice : remplacer "XXX" par le bon chemin XPath pour obtenir dans la colonne de gauche le `<persName>`, dans celle de droite la valeur de l'attribut xml :id du `<p>`
Le contexte est `<persName>`, le chemin doit donc être adapté en fonction

Exercice : Créer un tableau à partir d'un fichier XML

Solution :

```
<xsl:template match="tei:persName">
 <tr>
  <td><xsl:value-of select="."/></td>
  <td><xsl:value-of select="../@xml:id"/></td>
 </tr>
</xsl:template>
```