INU3011 Documents structurés

Cours 9

Introduction à XSLT

Plan

- Cours C08-XPath, diapos 70 et +
- Introduction à XSLT
  - Statut normatif et normes afférentes
  - Exemples disponibles
  - XSLT dans les navigateurs Web
  - XSLT dans oXygen
  - XSLT : le langage lui-même
    - 240-Ex-XSLT-pas-a-pas

Introduction à XSLT

Statut normatif et normes afférentes

Un (petit) imbroglio de normes (1/2)

- XSL = Extensible Stylesheet Language
- XSLT = XSL Transformations
- XSL-FO = XSL Formatting Objects
- XPath = XML Path Language
- Selon <http://www.w3.org/Style/XSL/> :
  - XPath ∈ XSLT
  - XSL = XSLT + XSL-FO

Un (petit) imbroglio de normes (2/2)

- Dans les faits :
  - Il n'y a *pas* de spécification séparée pour XSL-FO
  - La spécification intitulée XSL :
    - pointe furtivement à XSLT
    - consiste à 95% en la définition de XSL-FO

XSL-FO : statut normatif

- Extensible Stylesheet Language (XSL) Version 1.0
  - W3C Recommendation 15 October 2001
- Extensible Stylesheet Language (XSL) Version 1.1
  - W3C Recommendation 05 December 2006

XSLT: statut normatif

- XSL Transformations (XSLT) Version 1.0
  - W3C Recommendation 16 November 1999
- XSL Transformations (XSLT) Version 2.0
  - W3C Recommendation 23 January 2007
  - W3C Recommendation 30 March 2021 (2e éd.)
- XSL Transformations (XSLT) Version 3.0
  - W3C Recommendation 8 June 2017

Exemples utilisés

Apprendre XSLT par des exemples (1/3)

- 220-Ex-XSLT-en-classe exemples-jouets
- 240-Ex-XSLT-pas-a-pas (1 et 2)
  - Aussi vus en classe
- Aussi :
  - 180-Ex-multimedia
  - Feuilles XSLT de la BD "vins"
  - Stylage des éléments vides en XSLT
    - Lecture obligatoire au plan de cours

Apprendre XSLT par des exemples (2/3)

- Facultatifs :
  - 250-Ex-divers-XSLT (3 exemples divers)
  - 260-Ex-XSLT-attribut
    - Méthode générale pour mettre des attributs dans l'extrant de la transformation XSLT
  - 270-Ex-jointure-XSLT
    - Accéder à des informations dans un document externe séparé à partir de "clés" d'accès présentes dans le document XML qui est l'objet de la transformation

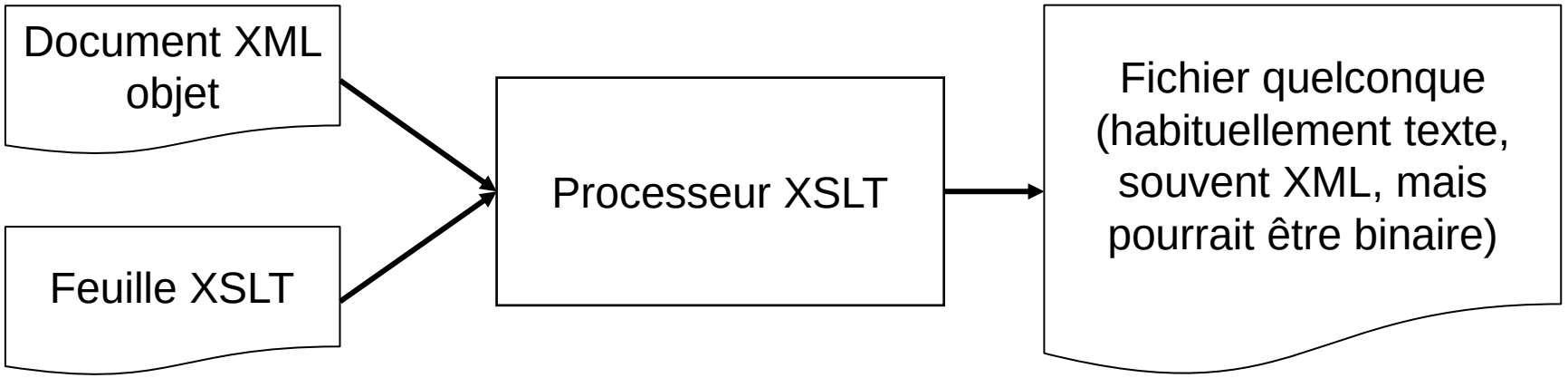
Apprendre XSLT par des exemples (3/3)

- Pour tous ces exemples :
  - Lire les éventuels fichiers « Lisez-Moi »
  - Lire les éventuels commentaires *à l'intérieur* des:
    - Feuilles de styles XSLT (.xsl)
    - Documents XML (.xml)
    - DTD (.dtd)

XSLT

Concepts et fonctionnement

Transformation XSLT en général (1/4)

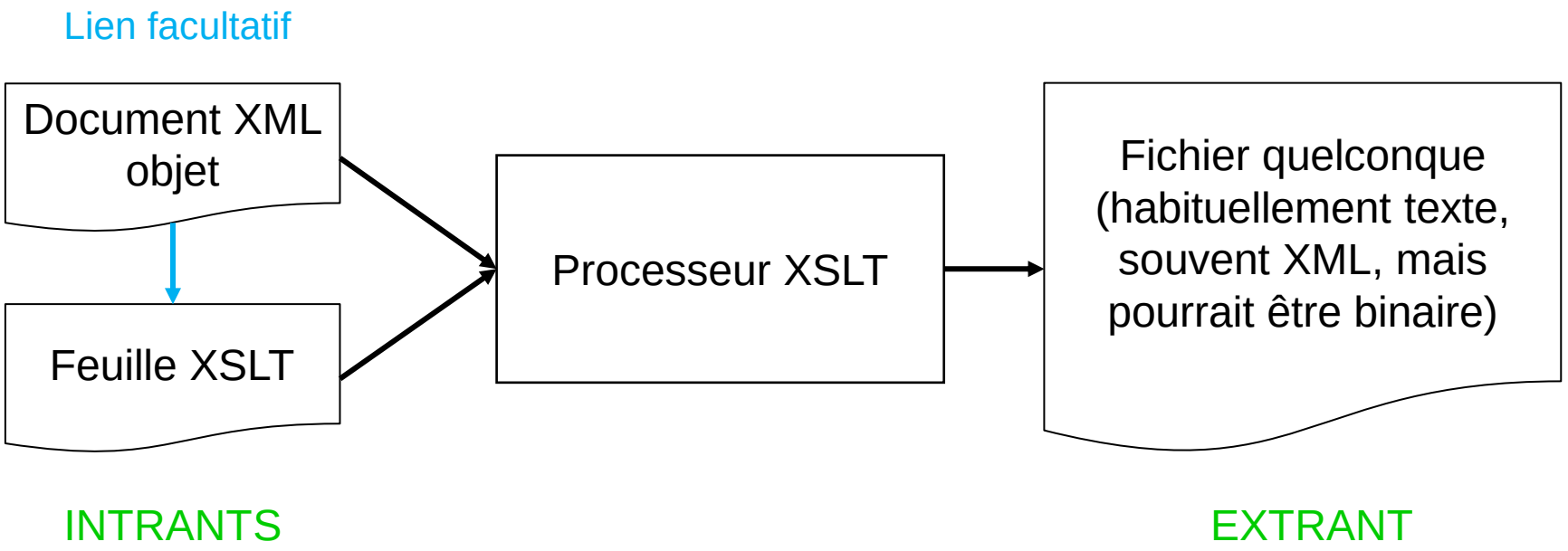


Transformation XSLT en général (2/4)

- Les 2 intrants peuvent être complètement indépendants (non liés)
- Mais, on peut aussi, comme avec CSS en HTML, "lier" le document XML à une feuille de style XSLT
  - Par une *instruction de traitement* dans le prologue du document, de la forme :

```
<?xml-stylesheet type="text/xsl" href="maFeuille.xsl" ?>
```

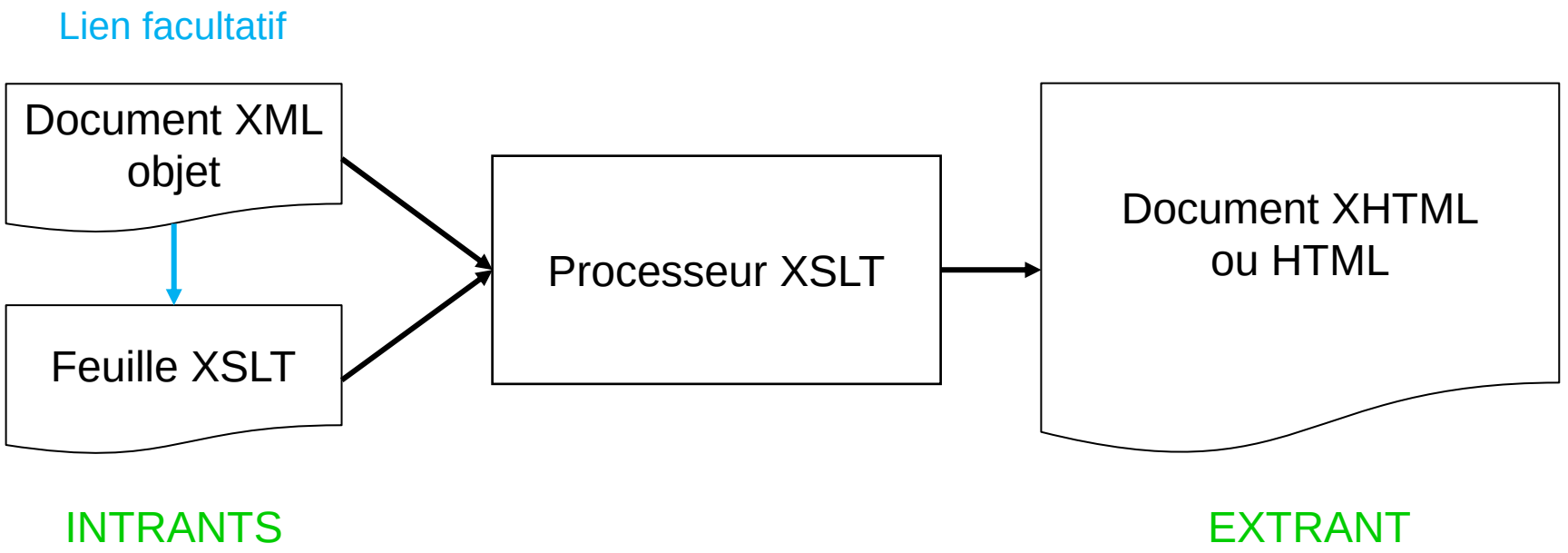

Transformation XSLT en général (3/4)



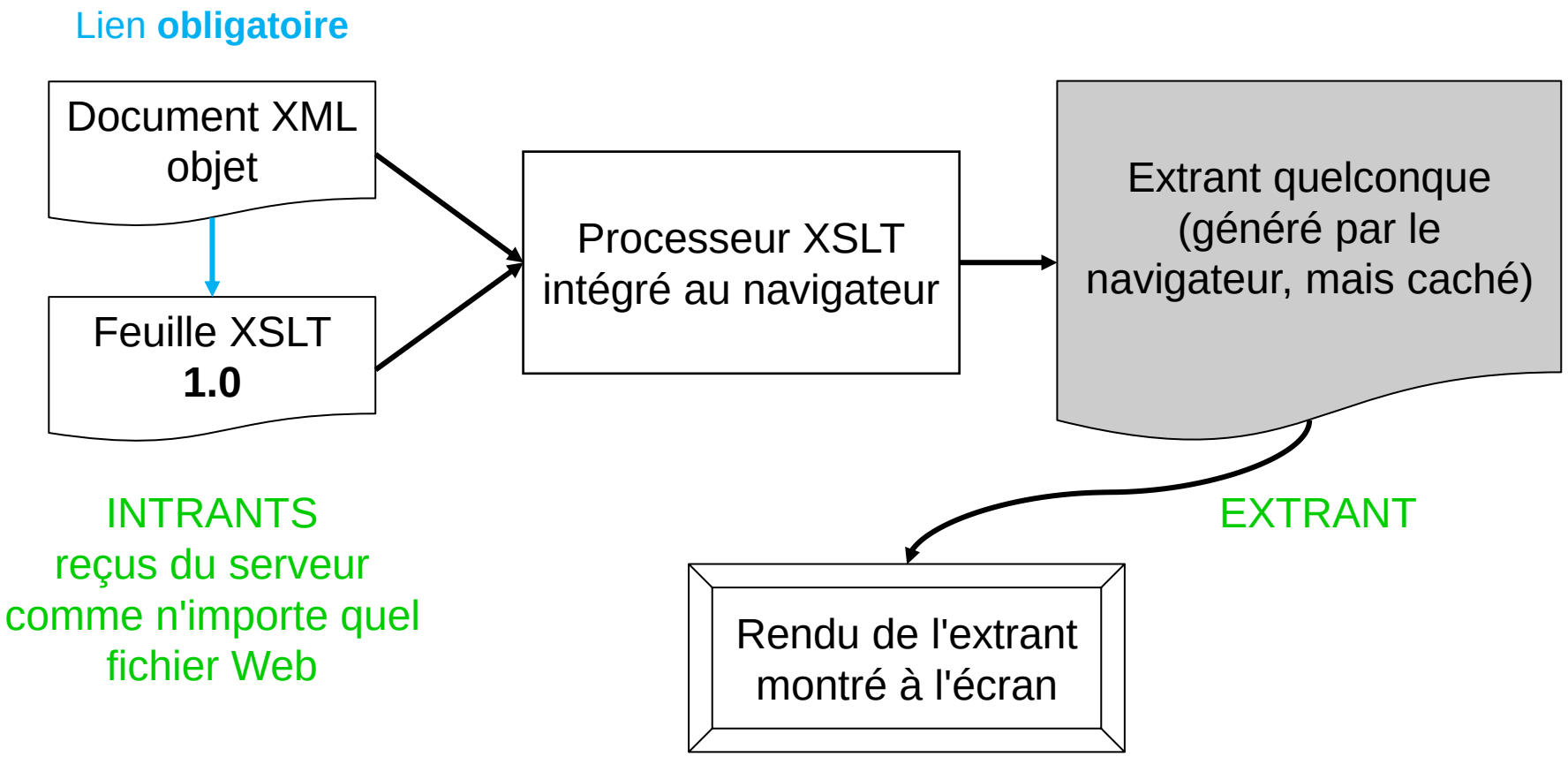
Transformation XSLT en général (4/4)

- Même si le document est lié à une feuille XSLT, l'application qui pilote la transformation peut habituellement, sur demande, faire transformer le document XML avec une *autre* feuille XSLT
  - C'est le cas d'oXygen, mais pas des navigateurs Web

Transformation XSLT vers HTML ou XHTML



Transformation XSLT en navigateur Web (1/3)



Transformation XSLT en navigateur Web (2/3)

- Les 2 cas de figure intéressants sont :
  - L'extrant est textuel
    - Le texte est rendu à l'écran tel quel
  - L'extrant est du HTML ou du XHTML
    - L'extrant est rendu à l'écran exactement comme si le HTML ou XHTML avait été reçu directement du serveur
    - C'est le cas de loin le plus fréquent

Transformation XSLT en navigateur Web (3/3)

- Noter que "Afficher source" fait afficher la source XML du document objet, et non la source de l'extrant
- La source de l'extrant de la transformation XSLT elle-même est *invisible* pour l'internaute
  - Seul son rendu à l'écran est visible

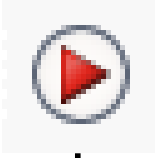
XSLT dans oXygen (1/4)

- oXygen reconnaît aussi un lien de la forme `<?xml-stylesheet type="text/xsl" href="`*`mon.xsl`*`" ?>` dans le prologue d'un document XML
- Cependant, la transformation n'est effectuée que sur demande
- Lorsque déclenchée, l'opération suppose par défaut une transformation vers du HTML; plus précisément…

XSLT dans oXygen (2/4)

- L'extrant est sauvegardé :
  - Dans le même dossier que le document XML
  - Sous le même nom de fichier, mais avec l'extension `.html`
- Juste après sa sauvegarde, cet extrant est également ouvert en navigateur Web :
  - Dans un nouvel onglet du navigateur par défaut du système d'exploitation

XSLT dans oXygen (3/4)

- Pour lancer cette transformation :
  - Sélectionner l'onglet du document à traiter
  - Cliquer sur l'icône 
    - "Appliquer scénario de transformation"
  - N.B.: La 1ère fois, il faut confirmer l'utilisation de l'instruction de traitement `xml-stylesheet`

XSLT dans oXygen (4/4)

- On peut définir d'autres "scénarios de transformation", qui permettent :
  - D'appliquer d'autres feuilles XSLT que celle pointée par le lien `xml-stylesheet`
  - Sauvegarder l'extrant dans un autre dossier que celui du fichier XML et avec une autre extension
- Ces possibilités serviront dans le TP 5

XSLT : le langage lui-même

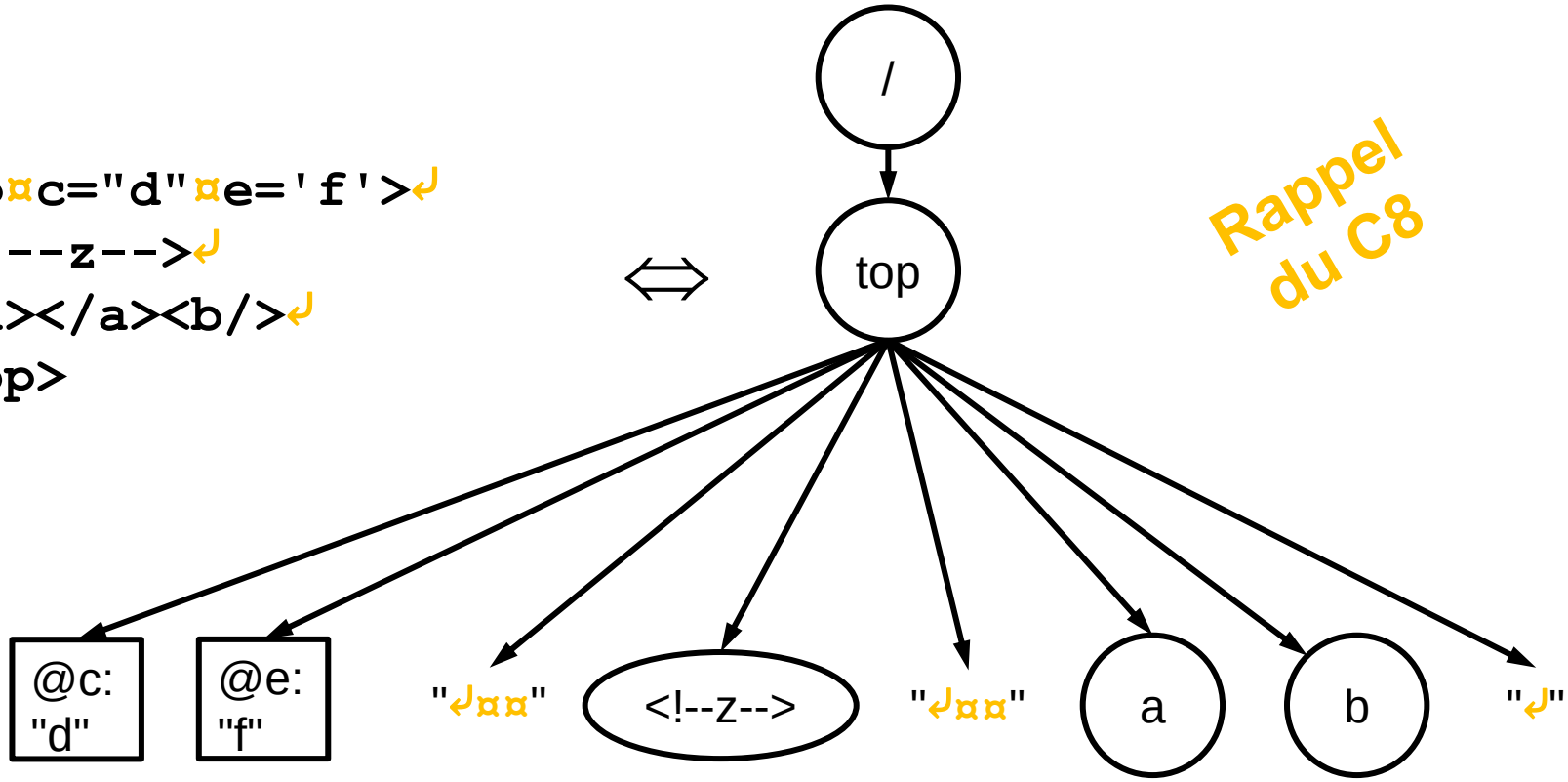
Note: On ne voit qu'une partie de XSLT 1.0 dans le cours

XSLT : la base

- Une feuille de style XSLT est un document XML *bien formé* (non valide)
- Elle contient une *suite de gabarits*
- Chaque gabarit indique au processeur XSLT *comment traiter certaines parties du document objet*
- Le fonctionnement des gabarits est défini en termes du *modèle de données XPath*

Modèle de données : XPath

```
<top¤c="d"¤e='f'>↲
¤¤<!--z-->↲
¤¤<a></a><b/>↲
</top>
```



Rappel du C8

Structure d'une feuille XSLT
(forme la plus simple)

Balise de début:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Corps (série de gabarits ou *templates*):

```
<xsl:template match="exemple">
  ...
</xsl:template>
...
```

Balise de fin:

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


Notion de base : gabarit

- Gabarit = *template* = règle
- Chaque gabarit indique au processeur XSLT comment traiter *certains* des nœuds du document objet
- Exemple :


```
<xsl:template match="//général">
  <p><em>Accessible au public</em></p>
</xsl:template>
```


Applicabilité d'un gabarit (1/2)

- L’attribut "match=" du gabarit :
  - Est une expression XPath indiquant l'*applicabilité* du gabarit (à quels nœuds il peut être appliqué)
  - Si l'expression est relative, elle est interprétée comme si elle était précédée de "//"
  - Elle ne peut pas comporter d'étapes ".."
    - Techniquement, c'est un "pattern", un sous-ensemble des chemins XPath valides

Applicabilité d'un gabarit (2/2)

- Dans un premier temps, on suppose *qu'au plus un gabarit* de la feuille XSLT est applicable à chaque nœud du document
  - Nous verrons sous peu comment ce qui se passe si ce n'est pas le cas

Cas simple d'applicabilité

`match="`*`identificateur-générique`*`"`

- Par exemple :

```
<xsl:template match="auteur">
  <p><em>Auteur: </em><xsl:value-of select="." /></p>
</xsl:template>
```

- Le gabarit est applicable à tous les éléments ayant comme nom cet identificateur générique
  - Dans l'exemple : `auteur`

Action du processeur XSLT (1/2)

- Recherche parmi les gabarits présents dans la feuille XSLT un qui s'applique au nœud racine (`match="/"`)
  - Si aucun, utilise plutôt un gabarit fixe prévu dans la norme (spécification) XSLT
- Applique ce gabarit au nœud racine
- L'extrant produit par l'application du gabarit au nœud racine constitue l'extrant de la transformation XSLT
- C'est TOUT

Action du processeur XSLT (2/2)

- L'action globale ne consiste donc qu'en **une seule application de gabarit**…
- Mais l'application d'un gabarit peut en déclencher une ou plusieurs autres, lesquelles peuvent à leur tour en déclencher d'autres en cascade
- C'est ainsi qu'une transformation XSLT comportera typiquement un grand nombre d'applications de gabarits

Extrant d'une application de gabarit

« L'extrant d'une application de gabarit est une chaîne de caractères égale au *contenu* (au sens XML) du gabarit, dans lequel les "instructions XSLT" (s'il y en a) sont remplacées par l'extrant de leur *exécution*. »

- N.B.: L'extrant *peut contenir des balises. C'est pourquoi une transformation XSLT peut produire du XML ou du HTML.*

Cas particulier (1/2)

- Cas où le gabarit ne contient *aucune instruction XSLT* (rare, mais possible)
  - L'extrant d'une application du gabarit est alors le contenu intégral du gabarit, sans aucune modification (puisqu'il ne contient aucune instruction XSLT)
  - Cet extrant est donc *toujours le même*, peu importe le nœud auquel le gabarit est appliqué !

Cas particulier (2/2)

- Exemple :

```
<xsl:template match="/"><p>Allô!</p></xsl:template>
```

- Une feuille XSLT qui contiendrait ce gabarit produirait toujours le même extrant, peu importe le document auquel on l'applique :

```
<p>Allô!</p>
```

- *Essayons-le !*

220-XSLT-en-classe
evo0

Que peut contenir un gabarit?

- Fragments bien formés XML (HTML ou autre) : texte et/ou balises
  - Ces fragments se retrouvent **tels quels** dans l'extrant du gabarit
- Des instructions XSLT, qui sont remplacées par l'extrant de leur exécution pour produire l'extrant de l'application du gabarit

Instructions XSLT

- Les deux principales instructions XSLT :

```
<xsl:value-of select="expr-XPath" />

<xsl:apply-templates select="expr-XPath" />
```

- Dans les deux cas, si ***expr-XPath*** est relative, elle est exécutée avec comme nœud courant *celui auquel le gabarit est en train d'être appliqué*

`<xsl:value-of select="`*`expr-XPath`*`" />`

- Retourne comme extrant :
  - Si *expr-XPath* est un chemin :
    - Le texte du *premier nœud* retourné par le chemin
      - Ou la chaîne vide si le chemin ne retourne aucun nœud
  - Si *expr-XPath* n'est pas un chemin :
    - La valeur retournée par l'expression, transformée en texte
      - nombre, chaîne, booléen ("true" ou "false")
- Exemples :

  `select="."` retourne le texte du nœud courant (i.e. son contenu textuel, s'il s'agit d'un élément)

  `select="@type"` retourne la valeur de l'attribut `type` de l'élément courant

  `select="nom"` retourne le contenu textuel du ***premier*** enfant du nœud courant qui est un élément `nom`

Modifions notre exemple

- Exemple :

```
<xsl:template match="/">
  <h1><xsl:value-of select="*/@titre" /></h1>
</xsl:template>
```

- Maintenant, l'extrant du gabarit - et donc de la feuille XSLT - dépend du document auquel on l'applique, en l'occurrence de l'attribut `titre` de son EPHN !
- *Essayons-le !*

220-XSLT-en-classe
evo1

`<xsl:apply-templates select="`*chemin-XPath*`" />`

- C'est l'instruction qui déclenche d'autres applications de gabarit
- S'exécute comme suit :
  - Ordonner les nœuds retournés par *chemin-XPath* par ordre d'apparition dans le document
  - Pour chacun des nœuds :
    - Appliquer le gabarit applicable à ce nœud
    - Recueillir l'extrant de cette application de gabarit
  - L'extrant du `xsl:apply-templates` est la *concaténation* des extrants ainsi recueillis
    - Ou la chaîne vide si *chemin-XPath* ne retourne aucun nœud

Pour illustrer le `apply-templates`

- Nous avons besoin d'un 2e gabarit :

```
<xsl:template match="nom">
  <li><xsl:value-of select="." /></li>
</xsl:template>
```

- Nous pouvons maintenant ajouter un `apply-templates` dans le gabarit pour "/" :

```
<xsl:template match="/">
  <h1><xsl:value-of select="*/@titre" /></h1>
  <ol><xsl:apply-templates select="*/nom" /></ol>
</xsl:template>
```

- *Essayons-le !*

220-XSLT-en-classe
evo2…4

XSLT Exemples pas-à-pas 1

240-Ex-XSLT-pas-a-pas

XSLT-ex-pas-a-pas-1

`<xsl:apply-templates />` (1/2)

- Forme "non sélective" (pas d'attribut `select`)
- Fait comme la forme sélective, mais avec tous les enfants du nœud courant (celui pour lequel le gabarit est en train d'être exécuté) *sauf* les nœuds attributs
  - Traite donc : les nœuds éléments, les nœuds textuels et les nœuds commentaires

`<xsl:apply-templates />` (2/2)

- C'est l'instruction XSLT de prédilection pour traiter les éléments qui ont un *modèle de contenu mixte*
- En effet, leur contenu peut consister en des unités de contenu textuelles et des sous-éléments entremêlés
  - Autrement dit, leurs enfants sont des nœuds textuels et des nœuds-éléments entremêlés

XSLT Exemples pas-à-pas (2/2)

240-Ex-XSLT-pas-a-pas

XSLT-ex-pas-a-pas-2

Gabarits par défaut (1/3)

- Des *gabarits par défaut fixes*, sont prédéfinis dans la norme XSLT pour tous les types de nœuds
- S'appliquent par défaut si *(et seulement si)* la feuille XSLT ne spécifie pas d'autre gabarit pour un nœud donné

Gabarits par défaut (2/3)

- Pour les éléments et la racine :

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

  - C'est-à-dire : traiter les enfants éléments, texte et commentaires

- Pour les commentaires :

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

  - C'est-à-dire : commentaires ignorés

Gabarits par défaut (3/3)

- Pour les nœuds texte et les attributs :

```
<xsl:template match="text()"
              ><xsl:value-of select="."/></xsl:template>
<xsl:template match="@*"
              ><xsl:value-of select="."/></xsl:template>
```

  - C'est-à-dire : leur texte

- Ces gabarits par défaut font qu'*une feuille XSLT vide* fait afficher le *contenu textuel* du document, et rien d'autre :
  - En particulier: aucun attribut ni commentaire

Bonne pratique

- Laisser le gabarit par défaut s'occuper du nœud racine "/"
- Utiliser un gabarit pour l'EPHN du modèle comme gabarit "principal" de la feuille XSLT
  - Notamment, y rattacher l'infrastructure HTML lorsqu'il s'agit d'une transformation vers HTML

220-XSLT-en-classe
evo5

Plusieurs gabarits applicables à un nœud (1/5)

- Qu'arrive-t-il si plusieurs gabarits de la feuille XSLT sont applicables à "/" ?
- Ou si une instruction xsl:apply-templates demande l'application d'un gabarit à un nœud et que plusieurs gabarits de la feuille XSLT sont applicables à ce nœud ?
- Réponse : c'est le gabarit avec *la plus haute priorité* de tous les gabarits applicables qui est choisi

Plusieurs gabarits applicables à un nœud (2/5)

- Pour assigner une priorité à un gabarit :
  - On utilise l'attribut `priority` de l'élément `xsl:template`
  - En l'absence de l'attribut, `priority="1"` est tenu pour acquis
  - Toute valeur numérique > 1 donne priorité au gabarit de notre choix

Plusieurs gabarits applicables à un nœud (3/5)

- Voir la lecture obligatoire :
  - Stylage des éléments vides en XSLT
  - pour un exemple commenté d'utilisation de la priorité des gabarits

Plusieurs gabarits applicables à un nœud (4/5)

- En l'absence d'attribut `priority`, sur un des gabarits applicables, le processeur XSLT choisira celui dont l'attribut `match` retourne en général le moins de nœuds
- Exemple : 3 gabarits, avec `match=`

```
nom
nom[1]
cc/nom[1]
```


Plusieurs gabarits applicables à un nœud (5/5)

- Puisque, en général :

  ```
  //nom
  ```

  retourne plus de nœuds que :

  ```
  //nom[1]
  ```

  qui en retourne plus que :

  ```
  //cc/nom[1]
  ```

  c'est donc le gabarit avec `match="cc/nom[1]"` qui sera choisi

Feuille XSLT comme application XML

- Une transformation XSLT est une application *spécialisée*, mais *non validante*
- Comme application non validante, elle peut être appliquée à *tout document XML bien formé*
- Comme application spécialisée, cependant, son résultat n'est prévisible que si on l'applique à un document valide selon un modèle spécifique, en fonction duquel elle a été conçue

Une feuille XSLT est un document bien formé

- La feuille XSLT étant un document XML, elle doit elle-même être bien-formée
- Donc, tout élément ouvert dans un gabarit doit être fermé dans ce même gabarit
  - En particulier, les balises "non XSLT " (p.ex. HTML) que l'on trouve dans un gabarit doivent "balancer" à l'intérieur du gabarit, même si elles sont simplement copiées vers l'extrant

Fonctions XSLT dans XPath

- Quand XPath est utilisé à l'intérieur de XSLT, des fonctions *additionnelles* peuvent être utilisées dans les expressions XPath
  - Liste ici
  - Les plus utiles:
    - document(*URL*)
    - current()
    - format-number(…)

(suite)

- Pour des exemples d'utilisation de :
  - document(*URL*) et
  - current()

  voir le dossier 270-Ex-jointure-XSLT (facultatif)

Aspects non vus

- Il est possible de trier (`xsl:sort`), de faire des calculs complexes, de comparer, de transformer du texte de différentes façons (p.ex. remplacer des codes par des intitulés plus longs), etc.
- En fait, XSLT est un langage de programmation complet

StudiUM

