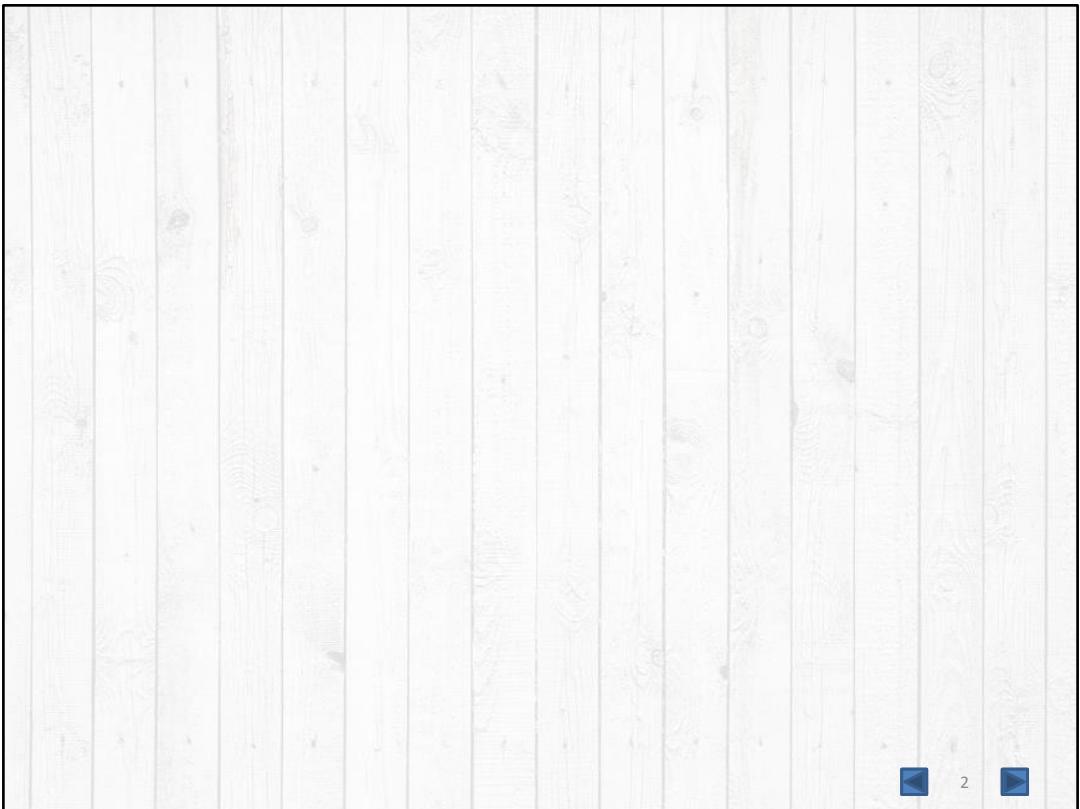


ORSYS
formation

TRANSFORMEZ VOS XML

Développer avec XSL



◀

2

▶



PRÉSENTATION LOGISTIQUE

ORSYS & VOUS



Bienvenue chez Orsys

Présentation



- Indépendant
 - Prestataire d'Orsys
 - Depuis 2011
- Animateur
 - Monsieur DESORBAIX
Alexandre
- Développeur
 - software, client lourd ,serveur
 - » c, c++, c#, java, ...
 - web, client léger
 - » html, php, sql, js, css,...
- blogger sous Wordpress



Bienvenue chez Orsys
logistique



- Nombre de jours :
 - 4 jours
- Horaires
 - Début : 9h
 - Fin : 17h30 approx.
- Pauses
 - 10h30 approx. Matin
 - 12h30 approx. Midi
 - 15h30 approx. Apres-midi



Au sommaire

- Au programme :

- Mise en forme du XML avec les feuilles de style CSS et enjeux du langage XSLT sur la transformation XML.
 - Mise en page simple de contenu XML
- L'exploration avec XPath 1.0 et 2.0 dans les données XML.
 - Parcours simple et complexe des arbres
- Le langage de transformation XSL-T 1.0 et 2.0 : construction d'arbres, restructuration,
 - Reformatage de documents XML/XML
- Les feuilles de style XSL-T : templates, structures. Visualisation brute sur les navigateurs.
 - Transformation de documents XML/HTML
- Génération multi-formats : XHTML, PDF/RTF (XSL-Fo).
 - Transformation de document xml pour les éditions particulières
- Génération d'éléments graphiques dynamique
 - Création de dessins ou animation a partir de données XML
- Cumul de langage XSL-T+SVG, XSL-FO+SVG
 - Édition de rapport riche contenant des graphiques a partir de données XML
- L'exploitation des XSL
 - Types et contexte d'exploitation XSL
- TP



- Qu'est-ce que le XML ?
- La galaxie XML
- XML et pages Web
- XML et accessibilité Web
- Xml et l'offre bureautique
- Parseurs
- Editeurs
- Espaces de nom

1. RAPPELS XML

Qu'est-ce que le XML ?

• Histoire

- **60's** : création de SGML (Standard Generalized Markup Language)
- **80's** : utilisation forte de SGML dans les métiers de l'édition
- **90's** : développement de standards issus de SGML pour le Web
- **11/96** : 1er brouillon (draft) de spécification XML
- **02/98** : XML 1.0 devient une recommandation officielle du W3C
- **02/04** : apparition de XML 1.1
- **08/06** : 4e édition de XML 1.0 (et 2e de XML)



La galaxie XML

- La galaxie XML

- XML (contenu et exports / imports BDD)
- XHTML (contenu Web)
- DTD, XSD (grammaires)
- XSL-T (Transformation) et XSL-FO (Formatage)
- RSS, Atom (syndication de contenu)
- SVG (graphiques), XLink
- SOAP (Web services... WS-*)
- XForms (formulaire)
- XQuery (requêtes), XPath (extraction)
- XUL (Mozilla), XAML (Microsoft Silverlight), MXML (Adobe Flash)



1.Rappels XML

I) Qu'est-ce que le XML ?

- Le XML

- XML (eXtensible Markup Language) est un Métalangage, donc un langage pour décrire (définir, inventer) d'autres langages
- XML est décrit en EBNF (Extended Backus-Naur Form) et sa dernière spécification est consultable à l'adresse :
<http://www.w3.org/TR/REC-xml/>

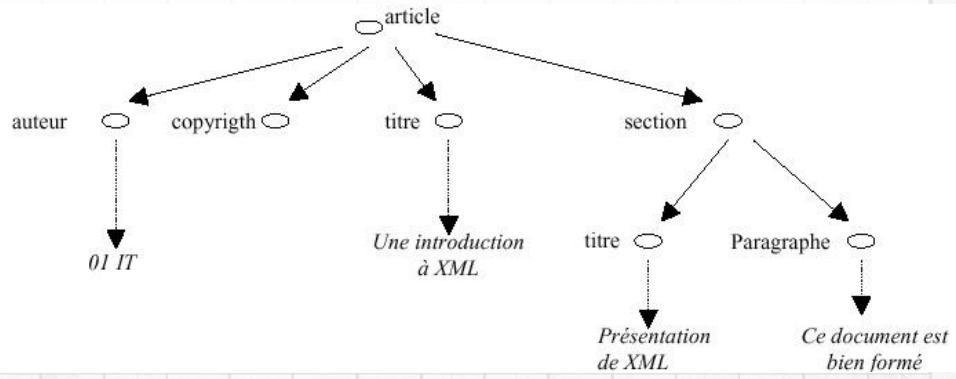
- Ses objectifs

- Faciliter l'échange de données structurées entre des SI différents
→ remplacer les formats inadaptés (CSV, SSV, tab, EDI, binaire...)
- Offrir un standard ouvert, libre indépendant des formats propriétaires
- Définir en son sein des mécanismes de contrôle de la validité des données



Qu'est-ce que le XML ?

- Structure arborescente en XML
 - les nœuds



Qu'est-ce que le XML ?

- Exemple de structure XML

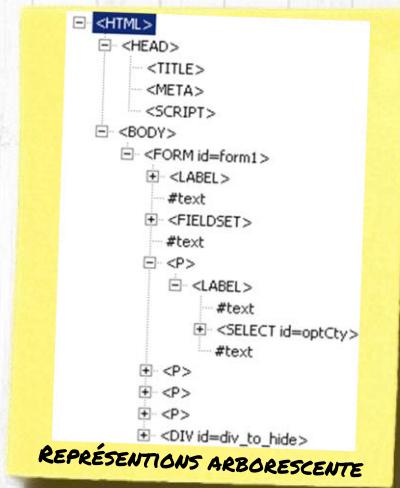
```
<?xml version="1.0" encoding="iso-8859-1" standalone="no" ?>
<rss version="2.0">
  <channel>
    <title>01 Information Technologies</title>
    <link>http://www.01-it.com</link>
    <description>Conseils-formateurs en systèmes d'information</description>
    <item>
      <title>Formation : conception et mise en œuvre XML</title>
      <link>http://www.01-
it.com/formation/xml_conception_application.html</link>
      <description>Concevoir et réaliser des applications XML...</description>
    </item>
    <item>
      <title>Formation : conception et modélisation XML</title>
      <link>http://www.01-
it.com/formation/xml_conception_application.html</link>
      <description>Créer des structures XML...</description>
    </item>
  </channel>
</rss>
```



Qu'est-ce que le XML ?

• Le DOM (Document Object Model)

- Le DOM (Document Object Model) est un modèle de document indépendant du langage et de la plateforme
- Une représentation de l'arbre XML utilisée par certains parseurs et processeurs pour l'explorer et le modifier
- Le DOM fournit une interface d'accès et de modifications des éléments du document.
- Le DOM prend en compte tous les éléments du document y compris les whitespaces entre les balises



« well formedness/ Bienformité »

- Les règles de « bienformité » sont les règles minimal de syntaxe pour qu'un fichier XML puisse être fonctionnel
- Pour être valide le fichier doit avant le dtd/xsd être bien formé
- Il existe 8 règles en 1.0 et 6 en 1.1

- | | |
|--|---|
| <ul style="list-style-type: none">❑ Processing instruction XML en 1^{er} caractère❑ Une seul balise racine❑ Pas de balise non fermé❑ Pas de balises croisées❑ Pas d'attribut sans valeur même null (attr="") | <ul style="list-style-type: none">❑ Le nom des balises commencent par une lettre ou '_' (underscore)❑ Le nom des attributs commencent par une lettre ou un '_' (underscore)❑ Les valeurs d'ID commencent par une lettre ou un '_' (underscore) nécessite un dtd/xsd |
|--|---|

XML et pages Web classiques

- Erreurs communes en XHTML
 - oubli de déclaration du type MIME XML sur le serveur
 - oubli de la balise de fin dans un élément vide :
`
 ➔
`
 - balises non-imbricables :
`<p><p></p></p> ➔ <p></p><p></p>`
 - balises entremêlées (ou croisées) :
`<i></i>`
 - oubli des guillemets pour les valeurs :
`width=30 ➔ width="30"`
 - utilisation impropre du & [< et >] :
`<title>html & xhtml</title> ➔ &`
Attention: cette règle est valable même dans les URL
 - oubli que les noms sont sensibles à la casse :
`BODY ➔ body`
 - abréviations d'attributs :
`disabled ➔ disabled="disabled"`
 - oubli du CDATA (Character DATA) :
`<![read & fill ➔ <![CDATA[read...`



XML et pages Web classiques

- **HTML vs XHTML**

- Rappels : HTML langage à balises peu structuré (ex: croisées ou vides)
- Norme HTML4 pour faire le ménage entre autres au niveau sémantique
 - le fond, la forme et l'interactivité sont séparés
 - les .css et les fichiers .js doivent se trouver dans des fichiers séparés
→ gain de productivité à la création, lors de l'entretien et de BP à l'utilisation
 - Les balises de mise en forme sont proscrites, ex : et <i> remplacées par et
- XHTML extension de HTML4 basée sur XML
- But W3C : Web + dynamique. HTML = static est un frein
- XHTML → traitements automatiques (langage naturel vs automatique)
- XHTML 1.0 → + de restrictions sur la forme, la casse (case sensitivity & minuscules) sur couples variable=valeur et références aux règles (DTD)
- Navigateurs pas compatibles XHTML1.0 (cf. MSIE 7)
- W3C recommande XHTML1.1/1.0 et HTML 4.01 plutôt que [X]HTML5.0



Précisions : initialement le HTML est un langage à balises, peu structuré.

Ex1: les balises html <i> peuvent être croisées ex: « ceci est un texte simple ceci est un texte gras <i>ici il est en gras et italique maintenant en italique seulement</i> et simple à nouveau ! »

Ex2: balises vides

La norme HTML4 a fait le ménage dans les mauvaises spécifications

XHTML (eXtended HTML) est une extension de HTML4 basée sur XML.

But du W3C : rendre le Web plus dynamique. Le HTML, trop static, est un frein à l'essor de pages Web réellement dynamiques.

Mieux structuré, XHTML permet les traitements automatiques (séparation sémantique contenu et mise en forme, unobtrusive JS).

XHTML 1.0 a donc surtout ajouté des restrictions sur la forme des documents (bien formés...) sur la casse (case sensitivity & minuscules) sur les couples variable=valeur et la référence à des règles (DTD).

Attention, certains navigateurs ne sont pas totalement compatibles XHTML1.0 (comme MSIE 7)

Bien que la norme [X]HTML5.0 soit bien avancée, W3C continue à recommander l'utilisation de XHTML1.1/1.0 et de HTML 4.01

XML et pages Web classiques

- HTML vs XHTML

- XHTML 1.0 propose 3 DTD fonction de la compatibilité désirée avec HTML4.01:
 - Strict : identique à HTML4.01 avec règles syntaxiques de XML
 - Transitional : idem XHTML1.0 avec attributs de présentation HTML3.2-
 - Frameset : identique à HTML4.01 pour l'usage des cadres <frame>
- XHTML[Modularization]1.1 version modulaire de XHTML1.0. Tags HTML bannis et présentation exclusivement en CSS
- XHTML offre 3 normes pour mobiles XHTML
 - "Basics" (allégée)
 - "Mobile Profile"
 - "+ Voice"



Xhtml 2.0 → html 5

XHTML 1.0 propose 3 DTD

En fonction de la compatibilité désirée avec HTML 4.01:

Strict : identique à HTML4.01 mais suit les règles syntaxiques de XML

Transitional : identique à XHTML1.0 mais préserve quelques attributs de présentation de HTML3.2- tels que <u> <strike> <applet>

Frameset : identique à HTML4.01 pour l'usage des cadres <frame>

Remarque: Accessibles via Dreamweaver: [Fichier – convertir]

XHTML[Modularization]1.1 est une version plus modulaire de XHTML1.0

Les tags purement HTML encore supportés sont bannis et la présentation ne se fait plus que par CSS. Une seconde version de XHTML1.1 et XHTML2.0 est en cours de développement par le W3C.

Notez que XHTML1.1 fait référence à des schémas XML (xsd) alors que XHTML1.0 utilise des DTD

- XHTML offre 3 normes pour mobiles XHTML "Basics" (version allégée), "Mobile Profile" (téléphones), "+ Voice" (XML : intéractions visuelle et vocale). Mais, les dernières innovations de Macintosh (iPhone) font penser que l'on ne va plus développer spécifiquement pour les mobiles.

XML et accessibilité Web

- Accessibilité et handicap
 - L'accessibilité = facilité d'accès au plus grand nombre ou facilité d'accès d'une entité à une fonctionnalité (vision qui ouvre à l'accessibilité des handicapés).
- WCAG, WAI et W3C
 - En IT, l'accessibilité est la faculté d'accéder à l'information et aux services à moindre coût, en offrant une interface accessible au plus grand nombre.
 - Le WAI (*Web Accessibility Initiative*) prône que :
 1. les outils de prod. de contenu doivent être accessibles à tous (ATAG)
 2. le contenu mis en ligne doit être accessible (WCAG)
 3. les outils de consultation (navigateurs) doivent être utilisables par tous

Nous sommes intéressés par le 2e point : l'accessibilité des sites Web.
 - Le WCAG 1.0 (*Web Content Accessibility Guidelines*)



XML et accessibilité Web

- WCAG, WAI et W3C suite...
 - Niveaux de priorité du WCAG :
 - 1.A (must) : obligatoire
 - 2.AA (should) : souhaitable
 - 3.AAA (may) : optionnel
 - Un des critères de niveau A, par exemple, est le respect de HTML4.01 strict
 - Outils de reporting de conformité au WCAG (ex: Watchfire WebXACT)
(Remarque: plus disponible librement depuis le 1er février 2008)

20

20



Précisions :

A l'instar des notations boursières, le WCAG fournit 3 niveaux de l'accessibilité d'un site appelé niveaux de priorité et identifie :

1. l'indispensable (must) [A] : les concepteurs/réaliseurs de sites doivent s'y conformer sous peine d'interdire l'accès à certains groupes.
2. le souhaitable (should) [AA] : les développeurs devraient s'y conformer sous peine de rendre l'accès difficile à certains groupes (niv. mini USA).
3. l'utile (may) [AAA] : les développeurs peuvent s'y conformer pour faciliter l'accès de leur site à certains groupes de personnes

Un des critères de niveau A, par exemple, est le respect de HTML4.01 strict

Des outils tels que Watchfire WebXACT permettent de passer gratuitement un site au crible selon les normes WCAG et d'éditer un rapport. IBM a décidé d'en retirer l'accès public depuis le 1er février 2008 et l'a intégré à son logiciel « IBM Rational Policy Tester Accessibility Edition » (anciennement « WebXM Accessibility Module »).

XML et accessibilité Web

- Accessibilité et gouvernements

- Aux USA, via la Section 508 de la loi américaine Federal Rehabilitation Act (www.section508.gov), le gouvernement prône le développement web selon les normes minimales WAI-AA et la validation XHTML & CSS, l'insertion de touches raccourcis, une version hautement contrastée du site, la présence d'un média alternatif pour tout contenu (ex: texte + son).
- En France, la loi 2005-102 du 11 février 2005 (art. 47) pour "l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées", dite loi "handicap", rédigée par l'ADAE (Association pour le Développement de l'Administration Électronique, prévoit l'accessibilité physique généralisée (sans rupture de déplacement), l'accessibilité renforcée des ERP... ainsi que des sanctions et des dates butoirs pour les mises aux normes (phase de diagnostic en 2011 et échéance en 2015). (cf. <http://www.legifrance.gouv.fr>)

Même les particuliers



Les parseurs XML du marché

- Types de parseurs

- On trouve des parseurs côté client (navigateurs, API MSXML...), côté serveur (API, bibliothèques, composants...)
- Les navigateurs Web les plus évolués (MSIE, MFF) intègrent des parseurs et des processeurs (ex: XSL-T)

- Exemples de parseurs

- IBM XML parser for Java
- Xerces Java Parser (SAX 2 et DOM 2)
- XP parser SAX utilisé entre autres par XT
- SimpleXML parseur XML pour PHP
- XML parser pour Adobe Flash
- JDOM, XMLBeans, StAX, JAXB, XOM...

22



Pour avoir une vue d'ensemble sur les Parsers :

http://www.xml.com/pub/rg/XML_Parsers

Panorama des éditeurs XML

- Les éditeurs XML

- **Altova XMLSpy** est l'un des meilleurs outils de production XML
- **oXygen** est un éditeur multiplateformes XML supportant aussi Relax NG, les Schematrons, SOAP, WSDL et permettant le débogage de XSL-T et XQuery
- **EditiX** est un éditeur similaire à oXygen orienté purement XML
- **Liquid XML Studio** propose une version libre et une payante très orientée programmation avec facilité d'intégration de composants Microsoft Visual Studio
- **Adobe Dreamweaver CS3+** dispose de fonctionnalités avancées de XML (génération XSL-T, intégration de fichiers XML via son framework Ajax, interpréteur XPath... mais ces outils souffrent de quelques carences)
- On note aussi l'existence d'éditeurs plus basiques tels que "**Antenna House XML Editor**", "**Open XML Editor**", "**XML Notepad**", "**Notepad++**" qui offrent une gamme de fonctionnalités de la simple coloration syntaxique XML à l'auto-complétion en passant par la validation et la génération de schémas et DTD



Pour en savoir plus :

<http://www.Altova.com/XMLSpy>

<http://www.oxygenxml.com>

<http://www.editix.com>

<http://www.liquid-technologies.com>

http://www.dmoz.org/Computers/Data_Formats/Markup_Languages/XML/Tools/Editors/Basic_Editors/

L'IDE eclipse

- Gratuit & Open-Source
- Auto-completion
- Version PHP
 - Prise en charge composants Web / html /css / dtd, déjà intégré
- Plugins XML
 - Modeling
 - XSD - XML Schema Definition SDK
 - Web, XML, Java EE and OSGi Enterprise Development
 - Eclipse XSL Developer Tools
- Debugger XSL

<input checked="" type="checkbox"/>	 Modeling	2.11.0.v20150806-0404
<input checked="" type="checkbox"/>	 Web, XML, Java EE and OSGi Enterprise Development	1.3.401.v201409111855

Adresse de téléchargement : <https://eclipse.org/downloads/>

Les espaces de nom

- Utilisation des espaces de nom
 - l'attribut « xmlns » précise le namespace auquel appartiennent les balises
`<html xmlns="http://www.w3.org/1999/xhtml">` (un exemple de default ns)
 - Sur le Web, le namespace prend généralement la forme d'une URI ou URN (mais ce n'est pas obligatoire)

Code XML :
`<bk:book xmlns:bk="..." xmlns:isbn="...">
 <bk:title...>... </bk:title>
 <isbn:number...>... </isbn:number>
</bk:book>`

- Les préfixes

- Utilisation des préfixes : `<html:html xmlns:html="...">`
`<html:head...> <html:body...>`
- Préfixes multiples :



Les espaces de nom

- La portée d'un espace de nom :
 - Portée (scoping): de la balise de déclaration à celle de fin (sauf autres zones ns)
 - Default namespace :
 - balises préfixées pas attributs
- Unicité du ns : 2 ns ayant la même valeur sont considérés identiques.
ex:

```
<book xmlns="..." xmlns:isbn="...">
    <title...>... </title>
    <isbn:number...>... </isbn:number>
</book>
```

```
<x xmlns:ns1="http://www.w3.org" xmlns:ns2="http://www.w3.org">
<bad ns1:a="1" ns2:a="2" />
<good a="1" ns2:a="2" />
```

- (a non-préfixé car attribut)



Les espaces de nom

- Choisir le NS par défaut
 - attention au choix du NS par défaut, il n'est pas anodin
 - il est préférable de l'appliquer au document de sortie final en cas de transformation
 - ou, certains parseurs ne gérant pas bien les espaces de nom, il est préférable d'appliquer le NS par défaut à ce parseur



❑ dans cette partie nous aurons vus

- ✓ Qu'est-ce que le XML & où l'utiliser?
- ✓ Ses rapports avec le web et ses usages
- ✓ Qu'il étais important de laisser une accessibilité minimum
- ✓ Comment il étais parcourus par les Parseurs
- ✓ et les outils pour l'éditer



- Introduction
- Syntaxe
- Liaison xml
- Disposition
- Sélecteur
- Héritage
- Types de propriétés*
- Propriétés
 - Disposition*
 - Unitées*
 - Couleurs*
 - Texte*
- Limites CSS

2. CSS

Css Introduction

- Qu'est-ce qu'un CSS
 - CSS (feuilles de style en cascade) langage de présentation applicable en cascade ! (c'est la technologie de mise en page utilisée initialement par XML)
 - Principe de séparation du contenu et de la mise en forme (plus accessible, pratique et rapide)
 - Les CSS externes liées (link), intégrées (head) ou inline ou spécifiées côté client (accessibilité des personnes handicapées)
 - Syntaxe, issue de javascript, est simple.
- Utilité des CSS
 - Réutilisation des mises en forme existantes facilement et rapidement, changement complet d'apparence simplement en changeant les CSS
 - Donc économie de BP car le formatage n'est pas défini pour chaque élément
 - Homogénéisation et cohérence de la présentation du site

30



Précisions : CSS (feuilles de style en cascade) est un langage servant à décrire la présentation de langages à balises (HTML, XML...). On peut appliquer plusieurs feuilles de style en cascade sur un élément et leurs effets se combinent (Arial+gras+12px & italique+souligné -> Arial 12 GIS) d'où l'utilisation du terme en cascade.

- Il a été créé sur le principe de séparation du contenu d'un document et de sa mise en forme. Ce qui rend le contenu plus accessible et la mise en forme plus facile à concevoir, modifier et réutiliser.
- Les CSS peuvent être mises dans un fichier et liées à la page ou intégrées au document (head) ou encore utilisées en ligne (inline : à éviter pour l'accessibilité) dans un élément ou enfin spécifiées par l'utilisateur dans son navigateur (ex: liens soulignés). Mais ces procédés sont à éviter car ils cassent l'intérêt de la séparation (structure, mise en forme, dynamique) et sont néfastes à l'accessibilité.
- La syntaxe de CSS (issue de javascript) est très simple. Un CSS contient un ensemble de règles constituées d'un sélecteur (nom de la "classe") ou d'une balise [a] et d'un block de déclarations du type "propriété:valeur;" .

Externes : les CSS peuvent également être contenus dans un fichier externe (mon_style.css) lié au document (X)HTML par la balise <link> (<link rel="stylesheet" type="text/css" href="mon_style.css">) . On peut avoir plusieurs CSS. Cette méthode est recommandée pour plusieurs raisons évidentes:

- économie de temps de développement (réutilisabilité / entretien site)
- économie de temps de chargement (CSS chargé une fois)
- indispensable pour une vraie séparation contenu / mise en forme
- utile pour le travail collaboratif et le respect des chartes graphiques

Il est possible de combiner les différentes manières de faire, pour des questions de priorité notamment.

Priorités : les moteurs de rendu CSS peuvent évaluer les priorités des styles. Un élément a une priorité de 1 , une classe de 10, un identifiant unique de 100. Une mention spéciale (!important) augmente la priorité de 1000.

Syntaxe Css

- Syntaxe de Base
 - sélecteur { propriété: valeur; }
- Sélecteurs multiples

```
Code css :
h1, h2, ... h6 {
    font-family : "MS Trebuchet", Verdana, Arial, sans-serif;
    font-weight : bold;
}
```

- Encapsulation des éléments

```
/* s'applique aux images contenues dans un lien*/
a img { border : none; }
/* s'applique aux paragraphes dans les balises class maClasse*/
.maClasse p { margin : 0.5em 0; }
```



CSS Histoire et normes

- Les feuilles de styles (cf. éditeurs de textes évolués) ont été inventées dans les années 70 au début de la mise en place du SGML, mais ce n'est qu'en 1994 que la norme CSS1 telle qu'on la connaît est apparue.
- La norme CSS2 est apparue en 97 afin d'intégrer de nouvelles fonctionnalités (ombres, direction, positionnement) mais suite à des problèmes d'intégration sur les navigateurs, elle a été transformée (plus stricte) en CSS2.1 en 2005. Aujourd'hui aucun navigateur n'est encore totalement compatible avec la norme CSS2 (même pas Firefox).
- La norme CSS3 débutée en 98 est toujours en cours de développement, elle prévoit la modularisation (cf. XHTML1.1)

Précisions : Il est possible et même recommandé d'aérer le code css. CSS permet une forme de type bloc utilisé dans beaucoup de langages de programmation. En effet des délimiteurs de bloc ({})) autorisent la formulation d'une instruction par ligne.

Sélecteurs multiples : On peut déclarer des instructions CSS pour des sélecteurs multiples. Il suffit de séparer ses sélecteurs par des virgules.

Encapsulation des éléments : On note l'encapsulation des éléments, on les classe en les séparant par un espace. Par exemple: a img signifie que le bloc de propriétés s'appliquera aux images contenues dans un lien.

Liaisons CSS / XML

- Formatage XML avec CSS
 - On peut formater les fichiers XML avec des CSS
 - On peut lier statiquement les fichiers XML au fichier CSS avec

```
<?xml-stylesheet href="xml_css.css" type="text/css"?>
```

- Comme en html tout élément XML (balise) peut être formaté, exemple :

```
Code xml:  
<personnes>  
    <personne>...</personne>  
    <personnes>  
  
Code css:  
personne { border: 1px black dashed;  
background-color: #F0F0E0}
```



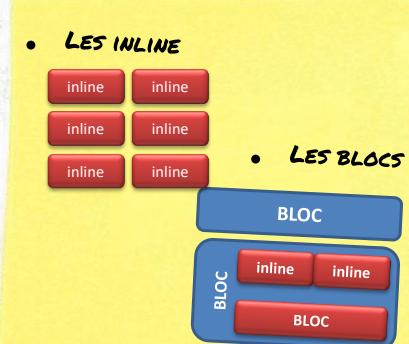
Disposition Css

- Types d'éléments
 - Bloc (bloc : div, ul, p, h1)
 - En ligne (inline : span, a, em, strong)
 - Affichage par défaut (display)
- Imbrication des éléments selon leur type
 - Bloc peut contenir un ou plusieurs blocs et inline

```
<div>
  <p>Paragraphe <em>1</em></p>
  <p>Paragraphe 2</p>
</div>
```

- Inline ne peut contenir QUE un ou plusieurs inline

```
<span><a href="">Cliquez
<strong>ici</strong></a></span>
```



33

Types d'éléments : En CSS les éléments de base sont soit des blocs, c'est-à-dire qu'il provoque un retour à la ligne (pour donner une idée), soit des éléments en-ligne (inline), c'est-à-dire qu'ils s'inscrivent dans la continuité.

Bloc : Les éléments blocs sont des conteneurs. Il s'agit d'éléments qui forment le squelette du document. Les titres (h1, h2... h6), les paragraphes (p), les listes (ul - non ordonnée, ol - ordonnée, dl - définition), les blocs (div), etc...

En-ligne : Les éléments inline sont des éléments liés au contenu. Ils permettent en général de l'enrichir : mise en emphase (em), renforcer du texte (strong), ajouter un lien (a). CSS permet de donner des instructions pour ces éléments

Display : Il est possible de changer le comportement CSS de base des éléments grâce à la propriété display !

Imbrication des éléments selon leur type : Il existe des exceptions pour les règles d'imbrication des divers éléments. Elles sont relativement peu nombreuses : un élément a, ne peut pas contenir d'autre élément a, body doit forcément comporter un élément bloc, etc...

Disposition Css

- Positionnement et notion flux
- Flux naturel (margin, padding)
- Flottant (Float)
- Position (position, z-index, rendu délicat)
 - Relative (relative, dans ou hors flux), Fixe (fixed, hors flux), Absolu (absolute, hors flux)
- Rôle particulier de la balise div
 - Une mise en page HTML traditionnelle (table) vs CSS (div)
 - Rôle de conteneur
 - Positionnement absolute (x et y) ou relative (voir flux / flot continu, superposition et z.index, conteneur, inherit)
 - Problèmes de restitution dans les navigateurs et affichage parfois « aléatoire »

34



Positionnement : c'est l'art de positionner les éléments. C'est une des tâches les plus difficiles en CSS du fait de l'incompatibilité des navigateurs (en particulier le mauvais usage des « margins » par MSIE). Le positionnement se fait selon le flux, c'est-à-dire l'ordre d'apparition des éléments dans le code source de la page. La structure du document (X)HTML détermine l'ordre d'affichage des éléments : c'est le **flux** (notion de frère importante pour le positionnement).

Flux naturel : agencement normal, utilisant margin (marges externes) et padding (marges internes). Départ en haut à gauche du conteneur parent dans toute la largeur. Les conteneurs frères se positionnent en rapport (par défaut en dessous).

Flottant : c'est la propriété float. Elle permet de positionner un élément (bloc ou inline) à gauche ou à droite d'un parent. Attention à la gestion de la hauteur (height) ! On peut ainsi positionner des blocs côté à côté.

Position : les éléments avec une propriété position, sont assez difficiles à positionner. Leur rendu est délicat et ne permet pas une restitution multi-résolution s'ils sont utilisés avec des valeurs statiques (px). Il faut leur préférer des valeurs proportionnelles (em, %). Cette technique permet de gérer la superposition des éléments (z-index).

• **Relatif** : permet de décaler un élément dans le flux. Les valeurs affectées aux propriétés top et left règlent le décalage.

• **Fixe** : sort l'élément du flux naturel et le positionne selon les indications données (top, left) par rapport à son parent ou dernier ancêtre positionné.

• **Absolu** : mis hors flux, l'élément est positionné par rapport au coin supérieur gauche de son parent selon les valeurs (top, left). Sans ces dernières, il se place dans le flux.

Remarque : le positionnement hors flux est très difficile à gérer. Il faut l'éviter au maximum, surtout pour du contenu afin de limiter les dégâts en cas d'intéraction utilisateur (agrandissement de la police).

Rôle de la balise <div> : Une mise en page CSS utilise le format de feuilles de style CSS, au lieu de tables ou de cadres HTML traditionnels, pour organiser le contenu d'une page Web. Les éléments de base de la mise en forme CSS est la balise div (conteneur de texte, images et autres éléments).

A la différence des cellules de tableau, qui ne peuvent exister qu'à l'intérieur de lignes d'un tableau, les balises div peuvent figurer n'importe où.

Vous pouvez positionner des balises div de façon absolue (x et y) ou de façon relative (distance par rapport à d'autres éléments).

Du fait de l'intégration imparfaite de CSS1&2 dans les navigateurs, il existe des problèmes de restitution sur différents navigateurs. L'affichage est parfois aléatoire avec les CSS. Voir [z.index](#) & [Débord](#) !

Sélecteur Css

- Eléments

```
p { font-size: 90%; margin: 0.5em; padding: 2px; }
```

- Classe

```
.important { color: red; }
```

- Identifiant

```
#menuGauche { margin: 0; padding: 0.5em; }
```

- Pseudo-classe

```
a:hover {  
border-top : 1px solid black;  
border-bottom: 1px solid black; }
```

35

Élément : un élément correspond à une balise (X)HTML telle qu'on la trouvera dans la structure du document, notée sans <> (h1, p, a). On peut ainsi définir des instructions CSS pour des éléments et façonnez un document de base.

Classe : lors de sa définition, une classe est identifiée par un point (.) précédent son nom (.maClasse). Vous pouvez choisir le nom que vous voulez, sans espaces. Elle pourra être utilisée sur différents éléments (X)HTML en leur ajoutant l'attribut class et la valeur de la classe entre guillemets sans le point (class="maClasse").

Identifiant : un identifiant fonctionne comme une classe. Il est référencé par un dièse (#) avant le nom. Un identifiant sert pour un élément (X)HTML unique en se basant sur l'attribut id des balises.

Pseudo-classe : les pseudo-classes permettent de mettre en forme des parties intégrantes du DOM. Il en existe peu :

- :active (élément actif),
- :focus (élément en focus),
- :hover (élément survolé),
- :link (non visité),
- :visited (élément visité) ,
- :first-child (premier élément fils),
- :lang (permet de spécifier une langue)

Héritage css

- Parent-Enfant
- Héritage (imbrication)
 - inherit
 - ex1: body → div → p ex2: table → tr → td
- Héritage déclaré (encapsulation)
 - #menuGauche → ul → li
- Classes multiples
 - class="Classe1 Classe2"

36



Parent-Enfant : c'est un concept fréquent grâce auquel on lie des éléments ou des objets en programmation, par un lien de "parenté". Dans le cas qui nous intéresse cela se traduit ainsi: table → tr → td, <http://giminik.developpez.com/xhtml/>. On peut pousser ce raisonnement en ajoutant les notions ancêtre et frère : ancêtre → parent → enfant--frère

Héritage : CSS utilise le système parent-enfant et donc permet l'héritage de propriétés. Une propriété héritable d'un parent (font-size par exemple) est transmise aux enfants qui n'ont pas défini la propriété ou si elle est fixée à « inherit ». Cela permet d'éviter de nombreuses répétitions. On peut annuler cet effet en redéfinissant une propriété dans sélecteur enfant. Les pseudo-classes sont des enfants : a:hover hérite les propriétés de a.

Imbrication des éléments : il est possible de spécifier des instructions pour des éléments en fonction de leur hiérarchie dans la structure du document. Ainsi il n'est pas obligatoire de créer systématiquement des classes. On pourra mettre en forme un lien (a) contenu dans un paragraphe (p) de classe (.maClasse) de façon spécifique et différente d'un lien courant. Encore un argument pour bien structurer vos documents, avec le (X)HTML bien sûr !

Classes multiples : il est possible de jongler avec les différents sélecteurs. On peut ainsi faire bénéficier un élément de classes différentes et lui appliquer les instructions de chacune de ces classes.

Propriétés Css

- Certaines propriétés proposent plusieurs modes de définition
 - Propriété spécifique
 - padding-top: 20px;
 - border-left-color: white;
 - Propriété globale ou super-propriété
 - padding: 0px 20px;
 - border: 1px dashed black;

37



Propriété spécifique : une propriété spécifique permet d'affecter de façon précise un élément. Par exemple une bordure à gauche. Ce procédé est utile lorsque l'on souhaite affiner un rendu. On peut ainsi affecter une couleur différente à une bordure gauche (border-left-color: silver;) et une bordure droite (border-right-color: black;).

Propriété globale : Comme pour les éléments, CSS est capable de gérer le rapport parent-enfant pour les propriétés. Ainsi une valeur affectant une propriété parente sera héritée par les enfants. border → border-top, border-bottom, border-left, border-right → border-top-color, border-top-style... Pour une propriété globale l'ordre a peu d'importance :
border : 1px dotted #e3e3e3; = border: dotted #e3e3e3 1px;

Sauf pour les positionnement : ils peuvent être déclarés de plusieurs façons :

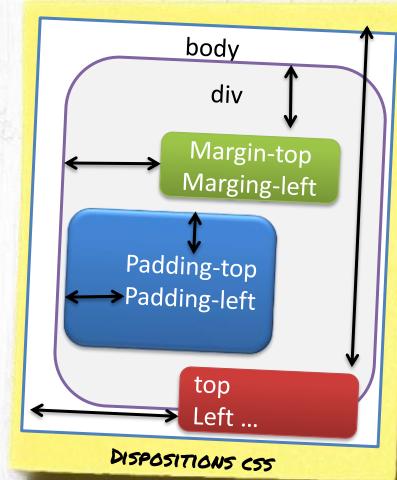
- 1 valeur: affectée partout (top, bottom, left, right).
- 2 valeurs : Valeur1 → top/bottom - Valeur2 → left/right
- 3 valeurs : Valeur1 → top - Valeur2 → left/right - Valeur3 → bottom
- 4 valeurs : Valeur1 → top - Valeur2 → right - Valeur3 → bottom - Valeur4 → left

Les schémas sont assez logiques et simples à retenir. Les valeurs multiples sont séparées par des espaces et ne sont pas forcément du même type (statique et proportionnelle mixables)

Ex : margin : 5px 2% 2em auto;

Les différentes propriétés Dispositions

- Hauteur & Largeur
 - Height: auto ;
 - Width : auto ;
- Positionnement
 - Positionnement absolu
 - Top:
 - Left
 - Right
 - Bottom
 - Marges extérieures:
Placement relatif les uns par rapport aux autres
 - Margin: (super propriété)
 - Margin-top:5px;
 - Margin-bottom:5px;
 - ...
 - Marges intérieures:
Placement dans le bloc
 - Padding: (super propriété)
 - Padding-left:8px;
 - ...



38

Les différentes propriétés-disposition

Positionnement absolue :

Il permet de se positionner par rapport au premier conteneur, quel que soit le niveau généalogique où il est déclaré ex de parent de référence :body dans html

Marges:

-Le margin : Il sert à espacer les blocs les uns par rapport aux autres, il se décompose en une super propriété et plusieurs déclinaisons telles que margin-left, -right, -top, -bottom

-Le padding, sert quant à lui à créer une marge à l'intérieur du bloc sans décaler le début du bloc; cette propriété se décline comme margin, une super propriété et d'autres propriétés spécifiques. Les cotés seront nommés aussi comme margin ex padding-left.

Les différentes propriétés

b) Les unités

- Il existe différentes unités de mesure en css

px → Pixel

em → Représente la valeur de la police
d'écriture du contenu

cm → Centimètre

mm → Millimètre

% → Pourcentage du conteneur parent

in → Inch

pt → Point 1pt=1/72 inch

pc → Pica 1pc=12pt



Pour permettre l'utilisation d'unités dans différents contextes, le css connaît et admet différentes unités.

-% le pourcentage est pratique pour que les pages soient toujours adaptées au mieux quel que soit la taille d'affichage (il est important avec % de prévoir que de petits écrans pourraient avoir accès aux pages donc essayer de faire en sorte que les données contenues aient suffisamment d'espace pour être lisible, les propriétés min-width, min-height, max-width et max-height, permettent de faire cela)

-cm, mm, in, pt, pc : ils sont souvent présents dans les fichiers css étant lié à des fichiers qui seront voués à l'impression

-px: Très souvent utilisé dans le cas de fichiers liés pour un affichage à l'écran

-em : Très utilisé par les web designer pour définir des blocs avec les hauteurs nécessaires pour écrire

Les différentes propriétés

c) Couleurs

- Modifier la couleur d'écriture
 - color:BLUE;
 - color:#**HEXVAL**;
 - color:rgb(**123,123,123**);
- Modifier couleur de fond
 - Backgroung-color: # **HEXVAL**;
 - Backgroung-color: color:rgb(**123,123,123**);
 - Backgroung-color:BLUE;
- Image fond
 - Backgroung-Image:url('http://.../...jpg');
- Interdire/forcer repetition
 - background-repeat: **no-repeat** ;
- Fond fixe
 - background-attachment: **fixed** ;
- opacité
 - Opacity : **1** ;

Les valeurs de propriétés :

Background-repeat :

repeat-x;	→répétition horizontale
repeat-y ;	→répétition verticale
no-repeat;	→pas de répétition
repeat;	→répétition horizontale & verticale

Opacity :

Valeur contenue entre 0 & 1

0 est totalement transparent

1 est totalement opaque

0.5 est entre deux

Background-attachement :

fixed ; →L'arrière plan ne défilera pas

scroll ; →valeur par défaut l'arrière plan défilera avec la page

différentes propriétés

c) Couleurs

- Modifier la couleur d'écriture

- color:BLUE;
- color:#**HEXVAL**;

#000000 équivalent rgb(0,0,0)
#FFFFFF équivalent rgb(255,255,255)

- color:rgb(000,128,255); ou hsl(0,128,255)

rgb = Rouge Vert Bleu **hsl**=Teinte Saturation Lumière
Les 2 propriétés existe avec la couche alpha **rgba** ou **hsla**

- Modifier couleur de fond

- Backgroung-color:BLUE;
- backgroung-color:#codecouleur;

41



Les valeurs de propriétés :

Background-repeat :

repeat-x;	→répétition horizontale
repeat-y ;	→répétition verticale
no-repeat;	→pas de répétition
repeat;	→répétition horizontale & verticale

Opacity :

Valeur contenue entre 0 & 1

0 est totalement transparent

1 est totalement opaque

0.5 est entre deux

Background-attachement :

fixed ; → L'arrière plan ne défilera pas

scroll ; → valeur par défaut l'arrière plan défilera avec la page

rgba différent de opacity

background-color:rgba(123,123,123,0.2)

```
div div div {  
    background-color: rgba(123,123,123,1);  
    border:3px solid rgba(0,0,0,1);}  
div {  
    background-color: rgba(123,123,123,0.2);  
    border : 3px solid rgba(0,0,0,0.2);}
```

The following div element's opacity is 0.5.
Note that both the text and the background-color are affected by the opacity level:

 Lorem ipsum dolor sit

 amet,

 consectetur adipiscing

 elit. Etiam semper diam at erat pulvinar, at
 pulvinar felis blandit...

opacity:0.2

```
div#principale {opacity:0.2;}  
div div div {opacity:1;}
```

The following div element's opacity is 0.5.
Note that both the text and the background-color are affected by the opacity level:

 Lorem ipsum dolor sit

 amet,

 consectetur adipiscing

 elit. Etiam semper diam at erat pulvinar, at
 pulvinar felis blandit...

42

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
div {  
    width: 300px;  
    text-align:center;  
    padding:15px;  
    color:black;  
    background-color: rgba(123,123,123,0.2);  
    box-shadow: 10px 10px 25px GREY;  
    border-radius:15px;  
    border:3px solid rgba(0,0,0,0.2);  
    margin-bottom:15px;  
}  
  
</style>  
</head>  
<body>  
<h1>The opacity Property</h1>  
  
<div id="a">The following div element's opacity is 0.5. Note that both the text and the  
background-color are affected by the opacity level:  
  
<div>Lorem ipsum dolor sit </div><div id="b">amet, <div>consectetur adipiscing</div> elit.  
Etiam semper diam at erat pulvinar, at pulvinar felis blandit...</div></div>  
  
</body>  
</html>
```

différentes propriétés d) textes

- **text-align**
 - Définit l'alignement horizontal du texte dans le conteneur parent
- **vertical-align :**
 - Définit l'alignement vertical dans le conteneur parent

43



Les propriétés de texte:

text-align : value ;

right	➔ alignement à droite
left	➔ alignement à gauche (valeur initiale)
center	➔ centrer
justify	➔ le texte est étendu à l'espace qui lui est attribué
inherit	➔ héritage de la valeur de propriété du parent

text-shadow: h-shadow v-shadow blur color

h-shadow	➔ position horizontale de l'ombre (obligatoire)
v-shadow	➔ position verticale de l'ombre (obligatoire)
blur	➔ distance de flou (optionnel)
couleur	➔ couleur de l'ombre (optionnel)

vertical-align: Value; (les plus courant)

espace	➔ monte ou redescend un élément de l'espace inscrit en unité
text-top	➔ alignement haut
middle	➔ alignement centré vertical
text-bottom	➔ alignement bas
inherit	➔ par héritage de la valeur du parent

différentes propriétés

e)shadow

- **text-shadow :**

- Ombre du texte (non compatible MS IE)

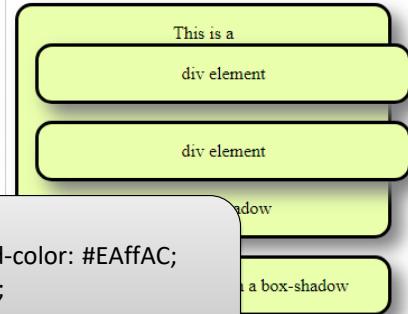
- exemple : `text-shadow:5px 5px 10px black;`

- **box-shadow**

- ombre d'un bloc html

- exemple :

```
div {  
    text-align:center; padding:15px;background-color: #EAffAC;  
    border-radius:15px; border:3px solid BLACK;  
    box-shadow: 10px 10px 25px GREY;  
}
```



44

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
div {  
    width: 300px;  
    text-align:center;  
    padding:15px;  
    background-color: #EAffAC;  
    box-shadow: 10px 10px 25px GREY;  
    border-radius:15px;  
    border:3px solid BLACK;  
    margin-bottom:15px;  
}  
</style>  
</head>  
<body>  
  
<h1>The box-shadow Property</h1>  
  
<div>This is a <div>div element </div>  
<div>div element </div>with a box-shadow</div>  
<div>This is a div element with a box-shadow</div>  
  
</body>  
</html>
```

Limites CSS

- Les limites de CSS
 - Impossible d'afficher les attributs*
 - Impossible de réordonner ou de numérotter dynamiquement des listes (ordre alphabétique par ex)
 - Impossible d'utiliser des affichages conditionnels (ou en conjugaison avec JS)*
 - D'où l'intérêt de recourir à un langage de style propre à XML le XSL qui se sert d'un sous-ensemble appelé XPath (étudions donc d'abord ce sous-ensemble)

*dispo
avec CSS 3.0



CSS 3.0 sélecteurs

- Sélecteur nouvelle génération

- Adjacents

```
ul + p { color: red; }
```

<p> directement précédé de

- Enfants

```
div#container > ul {border: 1px solid black;}
```

Uniquement les enfants contenu dans div#container

- Présence d'attribut

```
a[title] { color: green; }
```

Lien comportant un attribut 'title'

- Valeur d'attribut

```
a[title="greenLink"] { color: green; }
```

Lien comprenant l'attribut 'title' avec 'greenLink' comme valeur

Et plein d'autres : <http://code.tutsplus.com/fr/tutorials/the-30-css-selectors-you-must-memorize--net-16048>

Le CSS / Media

- **Media**

- screen, print, speech, all



```
@media not|only mediatype and  
      (media feature and|or|not mediafeature) {  
    /*CSS-Code;*/  
}
```

- **Syntaxe**

```
@media screen and (min-width: 400px) {  
  body {background-color: lightgreen;}  
}
```

- **exemple css pure ou intégration media HTML**

```
<link rel="stylesheet" media="screen and (min-width: 900px)" href="widescreen.css">
```



Les CSS pour mobiles

- @media (media queries) : responsive web design
 - CSS qui ne s'applique qu'au mode portrait
 - @media all and () {

```
body {...}
.tel {...}
{}
```
- CSS spécifiques
 - webkit-tap-highlight-color
 - webkit-user-select: none; (pas de sélection de texte possible)
 - -webkit-touch-callout: none; (évite l'apparition du menu contextuel)

48

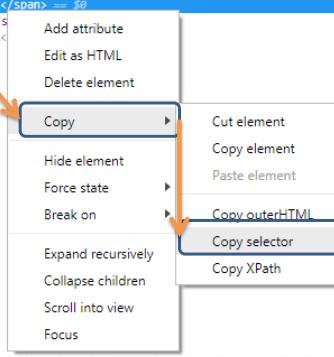


Pour en savoir plus sur @media : https://developer.mozilla.org/en/css/media_queries

Trucs et Astuces

- Sous chrome dans la console on peu obtenir le chemin **absolue** CSS d'un objet en un clic droit

```
<div class="pretty-print">
  <div class="collapsible" id="collapsible0">
    <div class="expanded">...</div>
    <div class="collapsed hidden">
      <div class="line">
        <span class="button expand-button"></span>
        <span class="html-tag"></span> == $0
        <span class="text">...</span>
        <span class="html-tag">...</span>
      </div>
    </div>
  </div>
</body>
</html>
```



sélecteur css et js

- Sélection css sans *jQuery* en **JavaScript**
- Sélection objet unique
 - `document.querySelector('Sélecteur CSS');`
- Sélection liste d'objets
 - `document.querySelectorAll(' Sélecteur CSS');`



❑ Dans cette partie nous aurons vu :

- ✓ La syntaxe et la structure du css
- ✓ La méthode de sélection et les nouveaux sélecteurs
- ✓ Les notions de dispositions
- ✓ Les couleurs en css
- ✓ Quelques propriétés css
- ✓ Comment être *@media ready*



- Exploration
- Syntaxe
 - A. Sélection
 - B. Axes
 - C. Calculs & Opération
- Fonctions

3. XPATH

XPath définition

- Définition

- XPath est un langage utilisé pour naviguer dans un document XML, en extraire des informations et effectuer des calculs et opérations simples ou complexes
- Standard du W3C utilisé dans XQuery (BdD), XPointer (pointeurs d'hyperliens sur des fragments XML) et surtout un élément majeur de XSLT

XPath définition

- XML Path Language (XPath) 1.0
 - **recommandation** du W3C du 16 Novembre 1999
 - Version non maintenue
- XML Path Language (XPath) 2.0
 - **recommandation** du W3C du 14 Decembre 2010
- XML Path Language (XPath) 3.x
 - **recommandation** du W3C du 8 avril 2014
 - 3.1 candidat du W3C du 17 Decembre 2015



Syntaxe

a) sélection

- Syntaxe : Sélection

Code xPath:

qName	→ Sélectionne tous les nœuds fils du nœud
/	→ Nœud racine
//	→ Tous les nœuds depuis le nœud courant _(pas child)
.	→ Nœud courant
..	→ Nœud parent du nœud courant.
@	→ Attribut
*	→ N'importe quel nœud élément
@*	→ N'importe quel nœud attribut
text()	→ N'importe quel nœud texte
node()	→ N'importe quel nœud
[expr]	→ Expr : [n] à nième élément last()/position(), [exprBool] vérifiée
	→ Union de requête
last(), position()	→ renvoie le dernier et la position courante



Précisions : Dans XPath la position n'est pas un index, la 1ère position est donc 1 (first() n'existe pas). Notez que le code « <xsl:if test="position()=last()"> » ajouté est d'une simplicité enfantine.

Syntaxe

a) sélection

- Syntaxe : Sélection – Exemples

Code xPath:

document	→ Sélectionne tous les nœuds fils de document
/document	→ Sélectionne document
document/html	→ Sélectionne tous les html fils de document
//img	→ Sélectionne les nœuds img descendants de doc...
body//input	→ Sélectionne tous les input descendants de body
//@id	→ Sélectionne tous les attributs nommés id
/*	→ Sélectionne tous les éléments du document
/document/*	→ Sélectionne tous les fils de document
//input[1]	→ Sélectionne le premier input de document(<small>balise</small>)
//img[@src]	→ Sélectionne les nœuds img ayant l'attribut src
//span[@lang='fr']	→ Sélectionne les nœuds span où lang = 'fr'
body/h1[position()<3]	→ Sélectionne les 2 premiers h1 fils de body



Syntaxe

b) Axes

- Syntaxe : Axes

- On peut préciser des axes de recherche par rapport au nœud courant et aux relations qu'il entretient avec les autres à l'aide de « nomAxe::nodeTest[expr] »
- axes: ancestor (tous les ancêtres), ancestor-or-self (idem + le node), attribute (tous attributs), child (tous), descendant[-or-self], following[-sibling], namespace, preceding[-sibling], self
- ex: child::* , descendant::text(), ancestor-or-self::body, child::*/child::name (noms des petits enfants du nœud courant)



Exemple d'utilisation d'un axe en XSL-T :

```
<xsl:value-of select="bibliotheque/livre//chapitre[1]/following-sibling::*[1]" />
```

Syntaxe

b) Axes

- Les Axes de parcours :

- Les axes de parcours permettent les sélections d'ensembles en rapport à la position courante
- Self
 - Permet la sélection du nœud courant
- **Ancestor[-or-self]**
 - Sélectionne tous les nœuds ancêtres (parents, grands parents,...) [et le nœud courant]
- **Descendant[-or-self]**
 - Sélectionne tous les nœuds descendant (enfants, petits -enfants,...) [et le nœud courant]
- **Child, Parent**
 - Sélectionne le nœud parent ou enfant le plus proche
- **Following[-sibling]**
 - Sélectionne tous les nœuds suivant le nœud courant [de la même fratrie]
- **Preceding[-sibling]**
 - Sélectionne tous les nœuds suivant le nœud courant [de la même fratrie]
- **Namespace**
 - Sélectionne tous les nœuds dans le namespace de la balise courante



Syntaxe

c) calcul & Operateurs

- **Calcul et Opérateurs**

- Xpath nous permet d'effectuer des calculs plus ou moins complexes sur les valeurs des différents nœuds, des attributs ou même lors de filtre Xpath ex : `/element[position()mod2=0]`

- Les opérateurs de calcul de XPath sont :

- + , - → addition , soustraction
- * , div, mod → multiplication , division, modulo (reste de division euclidienne)

- Les opérateurs de comparaison, test

- = , != → égal à / différent de
- < , > → inférieur à, supérieur à
- <= , >= → Inférieur ou égal / supérieur ou égal
- or , and → Cumul de conditions OU / ET
- | → union (cumul) exemple : /element[2] | /element[5]

59



=
pas (double) ==
pour comparer

Fonctions Xpath

- Xpath nous offre un panel assez vaste de fonctions permettant d'effectuer :
 - des traitements sur les chaînes (**string**),
 - des calculs mathématiques poussés avec les fonctions **math**
 - des opérations **boolean** ,...



Fonctions Xpath

- Mais aussi connaître sa position et des informations machines telles que la date*, l'heure*, et la timeZone*,...
- La version 2.0 de XPATH étend son panel de fonctions du panel de XPATH 1.0

Malgres le portage des fonctions de XPATH 1.0 vers 2.0 & 3.0



Le typage des donnees deviens fort

-les « **node?** »N'acceptent plus de liste

-les cast de types doivent etre definit

61



La gestion du temps des Xpath 2.0

Les fonctions Xpath

- Les fonctions Xpath (préfixe courant fn:*) :

- **Math numeric:**

- Abs(num) **valeur absolue** de num
 $\text{abs}(-345)=345$
 - Number(arg) **cast** en number de arg
 - Ceiling(num) **entier supérieur** le plus proche
 $\text{ceiling}(5.003)=6$
 - Floor(num) **entier inférieur**
 $\text{floor}(5.6)=5$
 - Round(num) arrondi avec **l'entier le plus proche**
 $\text{round}(3.4)=3$



Les fonctions Xpath

- Les fonctions Xpath (préfixe courant fn:*) :

- **String**

- codepoints-to-string(*int,int,...*)
 - string-to-codepoints(*string*)
 - Conversion de chaîne en suite de valeur de caractère et inversemement
 - compare(*comp1,comp2*)
 - Compare 2 chaînes renvoie 0 si égales
 - concat(*string,string,...*)
 - Concatène 2 chaînes ou plus
 - string-join((*string,string,...*), ' ')
 - Concatène 2 chaînes ou plus en séparant par sep



Les fonctions Xpath

- Les fonctions Xpath :

- String(suite)

- `substring(string,start[,len])`
 - Renvoie une partie de la chaîne commencer a start
 - `string-length(string)`
 - Renvoie la longueur de la chaîne
 - `Contains(string1,string2)`
 - `starts-with(string1,string2)`
 - `ends-with(string1,string2)`
 - `match(string1,string2)`
 - Renvoie vrai si présent (integral ou partiel, suivant type de recherche)
 - `replace(string,pattern,replace)`
 - Remplace le partern contenu dans string par replace



Les fonctions Xpath

- Les fonctions Xpath :

- Boolean

- True()
 - Renvoie vrai pour true pour vrai
 - False()
 - Renvoie vrai pour faux pour false
 - Not(boolean)
 - Renvoie l'inverse de boolean
 - Boolean(*arg*)
 - Cast en type boolean de arg



Les fonctions Xpath

- Les fonctions Xpath :

- Les fonctions de contexte nœud et machine

- Position(), Last()
 - Renvoie la position relative de l'élément en cours et last renvoie la dernière position de l'élément en cours

- **Uniquement Xpath 2+**

- Current-dateTime(), Current-time(), Current-date()
 - Renvoie la date actuelle au format date, time, datetime toujours avec Timezone

- Les fonctions d'agrégat

- count(node)
 - Renvoie le nombre d'élément node

- sum(arg,arg[,...]),
 - Renvoie la somme des arg contenues ou la moyenne

- **Uniquement Xpath 2+**

- max(arg,arg[,...]), min(arg,arg[,...]), avg(arg[,arg,...])
 - Renvoie la valeur max, min ou moyenne des arg contenues

66



sum(node) div count (node) équivalent a : Avg(arg,arg[,...]) ->fonction Xpath 2.0

Les fonctions Xpath

- Les fonctions Xpath :

- Les fonctions d'agrégat

- Count(node)
 - Renvoie le nombre d'élément node

- Les fonctions d'agrégat **Xpath 2.0**

- Max(arg,arg[...])

- Min(arg,arg[...])

- Renvoie la valeur max ou min des arg contenus

- Avg(arg)

- Renvoie la valeur max ou min des arg contenus

- Sum(arg,arg[...])

- Renvoie la somme des arg contenus



sum(node) div count (node) équivalent a : Avg(arg,arg[...]) ->fonction Xpath 2.0

Trucs & astuces

- Pour chaque expression Xpath plusieurs questions à se poser pour bien écrire :

– Ou suis-je ?

- Dois-je partir d'ici ou de la racine
 - » / ou // en début de recherche

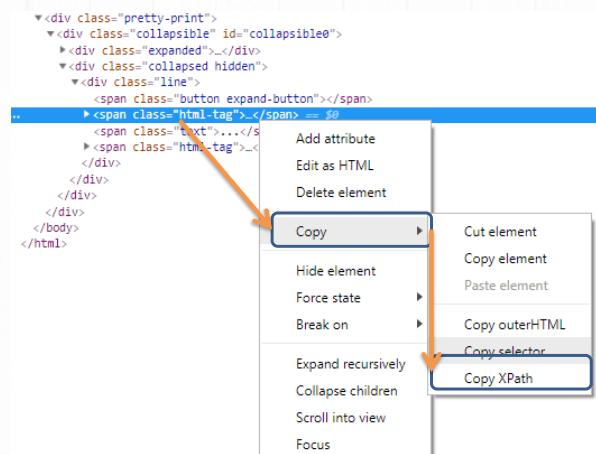
– Ou vais-je ?

- Poser les conditions du tronc vers les feuilles
 - » dégrossir les filtres [] dans le sens de parcours



Trucs et Astuces

- Sous chrome dans la console on peu obtenir le chemin **absolue** XPath d'un objet en un clic droit



❑ Dans ce chapitre nous avons abordés

- ✓ La syntaxe Xpath
- ✓ Les filtres
- ✓ Les fonctions Xpath
- ✓ Et les manières de l'utiliser



- Définition
- Compatibilité
- Outils
- Liaison XML
- Templates
- Variables
- Templates paramétrés
- Balises
 - xsl:output
 - Xsl:processing-instruction
 - Xsl:value-of
 - Xsl:element / xsl:attribut
 - Xsl:message / xsl:fallback
 - Xsl:if / Xsl:choose

4. XSL-T

Définition, le langage XSL-T

- Définition

- Le XSL (eXtensible Stylesheet Language) est un langage basé sur XML qui sert à transformer un document XML en un autre document (XML ou non)
-
- Pour simplifier : CSS = styles pour HTML et XSL = styles pour XML
- Initialement pour la présentation, mais aussi pour la transformation (XSL/T Transform, l'exploration et les calculs avec XPath)
- XSL-T se sert d'abord d'un parser pour analyser l'arbre XML avant d'utiliser son processeur XSL-T pour restructurer le document (gare aux parsing errors !)

72



Précisions sur le XSL-T : Le processus de transformation XSLT comprend un ou plusieurs documents XML (modules sources des données), un ou plusieurs modules XSLT (modules de mise en page) eux même décrits en XML, un outil d'interprétation de modèles XSLT (le processeur XSLT) et, au final, un ou plusieurs document résultats de l'opération de transformation.

En général, le processeur reçoit en entrée le document source (XML) et celui de mise en forme (XSLT) et crée un document en sortie. On a déjà vu ce type de fonctionnement lors de la transformation de types de documents (ex: XHTML en HTML) ou lors de l'import de document XML dans un modèle.

Il se sert de XPath (langage basé sur XML) pour identifier les branches des arbres XML (+ calculs et fonctionnalités) et de XQuery un langage de requêtes similaire à SQL créé pour manipuler de grandes collections de données XML et des sources de données pouvant être perçues comme du XML (ex: SGBDR).

Il existe des processeurs XSLT côté client (ex: MS IE intègre depuis 1999 un processeur XSLT conforme aux spécifications du W3C) ou côté serveur (ex: Oracle XML parser).

XSLT est maintenu par le W3C

Compatibilité XSL

- Compatibilité des navigateurs
 - Nombre de navigateurs ont d'un processeur XSLT natif (MFF1.02+, Opera9) alors que MSIE6+ utilise un ActiveX avec des fonctions propriétaires pour implémenter ses parsers (XML) et processeur XSLT
 - MSIE5 peut être rendu compatible avec quelques lignes de code de plus !
 - Malgré cette forte compatibilité forte, il ne faut pas oublier la compatibilité CSS avec les navigateurs(cf chapitre CSS)



Compatibilité XSL

- Compatibilité des navigateurs (détail) :
 - **Mozilla Firefox**
 - Firefox supporte XML, XSLT et XPath depuis la version 3.
 - **Internet Explorer**
 - Internet Explorer supporte XML, XSLT et XPath depuis la version 6.
 - Internet Explorer 5 n'est **PAS** compatible avec les recommandations officielles du W3C XSL.
 - **Google Chrome**
 - Chrome supporte XML, XSLT et XPath depuis la version 1.
 - **Opera**
 - Opera supporte XML, XSLT, et XPath depuis la version 9.
 - Opera 8 supporte uniquement XML + CSS.
 - **Apple Safari**
 - Safari supporte XML et XSLT depuis la version 3.



Outils XSL

- Difficulté de mise en œuvre à la main
 - XSL-Fo n'a pas de spécification
 - SVG est supporté différemment selon la visualisatrice
 - Tous les navigateurs ne se comportent pas de la même manière
 - Il est donc souvent plus simple d'utiliser des éditeurs spécialisés en création de documents XSL-T (et XSL-Fo...)
 - On gagne du temps, on évite les problèmes, on peut se concentrer sur l'aspect métier des développements
- Altova StyleVision
 - C'est un éditeur graphique de feuilles de style pour XML (XBRL et BdD) afin de le formater en (X)HTML, RTF, PDF, Microsoft Word 2007 et des e-forms !
 - Il supporte XSL-T 1.0/2.0, XSL-Fo, CSS, JS ainsi que de nombreuses bases de données



Pour en savoir plus sur Altova StyleVision :

http://www.altova.com/products/stylevision/xslt_stylesheet_designer.html

Liaison XML XSL en transformation client

- Lier le fichier XSL au XML (visualisation client)
 - Pour consulter le rendu d'un fichier XML passé par un processeur XSLT côté client statiquement, indiquez la liaison dans le fichier XML, comme suit :

```
<?xmlstylesheet href="personnes.xsl" type="text/xsl"?>
```

- Si la liaison est créée dynamiquement (JS), inutile de procéder de la sorte
- Si elle est effectuée côté serveur, vous recevrez le document déjà transformé



Liaison XSL-T (transformation) / XML (contenu) : Cette opération de liaison du fichier XSL au fichier XML fait perdre un peu l'avantage de la modularité au processus, mais n'oublions pas que nous plaçons, ici, la transformation XSL du côté client. Aussi, si on dispose d'un processeur XSLT côté serveur, le fichier XSL sera associé dynamiquement au fichier XML (cf. liaison XSL-T dynamique en javascript).

N'oubliez pas que seuls les navigateurs récents et puissants disposent d'un processeur XSLT intégré, aussi, cette manière de procéder ne fonctionnera pas avec des versions anciennes de navigateurs (IE5-, Netscape...).

Attention : l'attribut de version est obligatoire

namespace & préfixe XSL

- Utilisation des namespaces
 - Le document XSL qui est un doc XML commence par une déclaration XML, ex:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

- Puis, pour distinguer les commandes XSL des autres contenus, on préfixe les mots réservés XSL par un namespace déclaré (souvent xsl), ex :

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

→ « <xsl:if test="position()=last()"> » (« xsl:transform » eq.
« xsl:stylesheet »)



Liaison XSL-T (transformation) / XML (contenu) : Cette opération de liaison du fichier XSL au fichier XML fait perdre un peu l'avantage de la modularité au processus, mais n'oublions pas que nous plaçons, ici, la transformation XSL du côté client. Aussi, si on dispose d'un processeur XSLT côté serveur, le fichier XSL sera associé dynamiquement au fichier XML (cf. liaison XSL-T dynamique en javascript).

N'oubliez pas que seuls les navigateurs récents et puissants disposent d'un processeur XSLT intégré, aussi, cette manière de procéder ne fonctionnera pas avec des versions anciennes de navigateurs (IE5-, Netscape...).

Attention : l'attribut de version est obligatoire

Templates xsl name, match, ...

- **Template**

- La balise template définit le modèle du document utilisé pour construire la page ou un de ses éléments et match indique la cible à l'aide d'une expression XPath
 - `<xsl:template match="/">`
 - on peut appeler un template particulier avec
 - `<apply-templates select="node" />`
 - ou tous les templates dans l'ordre de parcours
 - `<apply-templates />`
- On peut aussi créer et appeler explicitement un template nommé

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

78

Exemple d'apply-templates :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output type="html" doctype="...">

<xsl:template match="/">
<html>
<head> ... </head>
<body>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>

<xsl:template match="personne">
<p>
<xsl:apply-templates select="nom"/>
<xsl:apply-templates select="prenom"/>
</p>
</xsl:template>

<xsl:template match="nom">
    Nom : <span style="color:#ff0000"><xsl:value-of select=". /></span><br />
</xsl:template>

<xsl:template match="prenom">
    <span><xsl:value-of select=". /></span><br />
</xsl:template>

</xsl:stylesheet>
```

Variables globales & constantes

- Utilisation de constantes et variables globales

- On peut définir et utiliser des paramètres globaux

- Pour définir un paramètre, on utilise `<param>`, ex :

```
<xsl:param name="node_path_adr"  
[select="personne/adresse"] />
```

- Pour l'utiliser on le préfixe d'un \$, ex :

```
<xsl:apply-templates select="$node_path_adr" />
```

- Les paramètres sont des variables globales qui, une fois instanciées ne peuvent pas être modifiées, sauf dans le cas d'un appel récursif ou plusieurs instances du template seront créer avec les mêmes nom de param



Les « param » sont souvent utilisés lors de déclaration de template pour définir un paramètres d'entrées d'un template et détruit à la sortie du template

Variables locales, globales & constantes

- Il existe un autre mode de déclaration local (a l'endroit où l'on en a besoin)
 - On peut aussi utiliser <variable>

```
<xsl:variable name="nb_boucles"  
select="position()" />
```

- les variables peuvent être définies localement (ce n'est pas le cas de paramètres sauf dans le cas des templates paramétrés cf. ci-après)



Les types variable et param sont des
constantes

Une seul affectation lors de la définition

80



Variables locales, globales & constantes

- Il existe 2 modes de définition
 - <xsl:variable> avec **select="XpathExprssion"**
 - <xsl:variable> **sans*** select=""

```
<xsl:variable name="nb_boucles"  
select="position() />
```

```
<xsl:variable name="maVar" >  
<balise><xsl:value-of .../></balise>  
</variable>
```

*Pas supporté
sous tous les
moteurs XSL
pour refaire des arbres XML



Les types variable et param sont des
constantes

Une seul affectation lors de la définition

Templates XSL paramétrés

- Templates paramétrés

- En XSL-T, il peut être nécessaire d'utiliser des fonctions (ex : redondance du code dans les traitements conditionnels).

- Pour cela XSL-T prévoit un mécanisme de templates paramétrés, on les crée comme suit :

```
<xsl:template name="ma_fonction">
<xsl:param name="prmBgcColor" />
<tr bgcolor="#{$prmBgcColor}">
<td><xsl:value-of select="nom" /></td>
<td><xsl:value-of select="prenom" /></td>
</tr>
</xsl:template>
```

- Et on les utilise comme suit :

```
<table><tr bgcolor="#aaa"> <th>Nom</th><th>Prénom</th></tr>
<xsl:for-each select="personne">
<xsl:call-template name="ma_fonction">
<xsl:with-param name="prmBgcColor" select="FFFFAA" />
</xsl:call-template>
</xsl:for-each></table>
```



xsl:output, format de sortie

- Le format de sortie
 - On utilise la balise <output /> pour préciser un format de sortie
 - ex : `<xsl:output method="text" />`
 - On peut préciser text, xml, html, cependant des traitements différents seront opérés selon le processeur XSL-T, par exemple :
 - sous MFF2.0.0.20, des balises <html> sont insérées même dans le cas d'une sortie purement texte demandée (la balise html <pre> simule une sortie texte à l'affichage dans le navigateur)
 - sous certaines versions MFF3 et MFF2 et MSIE6 les balises <html><head><body> ne sont ajoutées que pour une sortie html
 - sous MSIE7 la plupart du temps, aucune balise n'est ajoutée mais le code en sortie n'est considéré comme du HTML que si la sortie est précisée en méthode html (même si le document html est parfait)
 - Il vaut donc mieux utiliser des design patterns de fichier XSL-T générant un document XHTML correctement interprété pour les navigateurs visés
 - Attention l'import de fichiers d'entités n'est pas possible sous de nombreuses versions de MFF d'où l'obligation de redéfinir les entités dans le fichier XSL-T

83



Exemple de design pattern de XHTML en XSL-T pour MSIE6/7 et MFF2/3 :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE xsl:stylesheet [
    <!ENTITYnbsp "amp;nbsp;"...
    les imports de ".ent" en entités paramètres ne sont pas supportées par MF
]>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" encoding="iso-8859-1" version="1.0" standalone="no" doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN" doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />

<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Design Pattern XHTML pour XSL-T </title>
<!-- <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" /> -->
<link href="design_pattern_html.css" rel="stylesheet" type="text/css" />
</head>
<body>
<h1>Titre : &nbsp;&nbsp;&nbsp;<xsl:value-of select="racine/titre" /></h1>
<h2><xsl:value-of select="racine/soustitre" /></h2>
<xsl:for-each select="racine/paragraphe">
<p><xsl:value-of select="text()" /></p>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

xsl:output, format de sortie

- XSL-T dispose de valeurs d'attribut disponible servant à définir la manière dont va être généré le document de sortie
 - Les éléments non-XSL-T (à ne pas transformer) s'expriment dans leur langage habituel (xml, html, svg, texte, CSV/SSV...)
- Définition du type de sortie:

```
<xsl:output method=" type" />
```

- Types :
 - Html → génère un flux html
 - Txt → génère un flux texte brut
 - Xml → génère un flux xml (ex fo, rss, svg, xml sous une autre structure)
 - Xhtml → (uniquement en XSL 2.0)(XML en XSL1.0)



xsl:processing-instruction

- Les instructions processeur
 - Les instructions processeur insérées dans le XSL-T sont prises en compte lors de l'interprétation du XSL-T. Or, il est parfois nécessaire d'en insérer pour le document de sortie, par exemple pour lier un fichier CSS ou préciser un encoding
 - exemple d'insertion d'un encoding

```
<xsl:processing-instruction name="xml">
    encoding="iso-8859-1"
</xsl:processing-instruction>
```

- exemple d'insertion d'une relation à une feuille de style

```
<xsl:processing-instruction name="xmlstylesheet">
    href="my_styles.css" type="text/css"
</xsl:processing-instruction>
```

- Balises particulières
 - Il existe aussi des instructions spécifiques aux traitements XSL-T, telles que :
 - les commentaires : `<xsl:comment> tarifs au format français </xsl:comment>`
 - les whitespaces : `<xsl:strip-space elements="*" />` ou `<xsl:preserve-space elements="description" />`



xsl:Value-of, valeur de nœud

- Récupérer la valeur d'un nœud
 - On utilise <value-of select>
 - `<xsl:value-of select="personnes/personne/text()" />`
 - Remarque : par défaut, value-of ne peut pas interpréter les expressions conditionnelles dans les requêtes XPath (vous recevez toujours le 1er élément d'un ensemble de nœuds)
 - Ou la notation avec {} à l'intérieur d'un attribut

```
<select id="productId">
  <option value="{produit/@id}">
    <xsl:value-of select="produit/nom" /></option>
</select>
```

 - Remarque : dans cette dernière notation, si des calculs interviennent, il faut les effectuer à l'intérieur de l'accordéon.
 - Par exemple, l'expression : `<svg:rect x="10+20*{position()}px" />` produirait, pour la position 1, le code suivant : `<svg:rect x="10+20*1px" />` (non-interprétable en SVG)
 - l'écriture correcte serait donc : `<svg:rect x="{10+20*position()}px" />` qui produirait, pour la position 1, le code suivant : `<svg:rect x="30px" />` (interprétable en SVG)
 - On peut générer dynamiquement des éléments et attributs (cf. commentaires)
 - Enfin, on peut aussi utiliser la notation par **key** ou **id** pour chercher un nœud

86

86



On peut générer dynamiquement des éléments

Par exemple ici les éléments sont créés en fonction de leur attribut @typeParagraphe (div, p, span, td...)

```
<xsl:element name="{paragraphe/@typeParagraphe}"><xsl:value-of select="paragraphe/text()" /></xsl:element>
```

Où des attributs (dans des éléments existants) :

Par exemple ici l'attribut style="font-weight:bold;" est ajouté au td si l'attribut bold de <paragraphe> est à true

```
<td><xsl:attribute name="style"><xsl:if test="paragraphe/@bold='true'"><xsl:text>font-weight:bold;</xsl:text></xsl:if></xsl:attribute></td>
```

On peut même générer des ensembles d'attributs :

Définition :

```
<xsl:attribute-set name="table_title">
  <xsl:attribute name="align" >center</xsl:attribute>
  <xsl:attribute name="style" >font-weight:bold;font-size:18px;</xsl:attribute>
  <xsl:attribute name="bgcolor" >#888888</xsl:attribute>
</xsl:attribute-set>
```

Utilisation : <td xsl:use-attribute-sets="table_title"><xsl:value-of select="titre" /></td>

Exemple de recherche par clé (l'usage de id est encore plus simple) :

Définition de la clé : <xsl:key name="keyParagraphe" match="paragraphe" use="@bold" />

Utilisation de la clé : <xsl:for-each select="key('keyParagraphe','true')">

xsl:element, xsl:attribut

- Les balises importantes en XSL pour sortie XML:

- **Element** sert à inscrire une nouvelle balise vers la sortie
 - `name="name"`
- **Attribute** inscrit l'attribut défini dans le parent
 - `name="source"`

- Usages :

```
<xsl:element name="singer">
    <xsl:value-of select="artist" />
    <xsl:attribute name="source">
        <xsl:value-of select="images/name" />
    </xsl:attribute>
</xsl:element>
```

Nous donnerons le morceau d'arbre suivant :

`<singer source="nameVal" >artistValeur</singer>`



xsl:message, xsl:fallback

- Gestion de message vers la sortie

- Une balise message nous permet d'envoyer vers la sortie des informations. Cette balise est pratique pour signifier par exemple des erreurs de parcours de documents et informer les raisons de l'échec et forcer la fin du traitement

```
<xsl:message terminate="yes/no">  
    message d'erreur  
</xsl:message>
```

permet l'arrêt ou non du processor lors de l'envoi du message

- Gestion de prise en charge processor

- Lorsque qu'une balise n'est pas prise en charge par un processor, on peut définir une alternative à cette balise exemple avec loop qui sera remplacée par for-each si la balise loop n'est pas prise en charge par notre processor.

```
<xsl:loop select="title">  
    <xsl:fallback>  
        <xsl:for-each select="title">  
            <xsl:value-of select=". />  
        </xsl:for-each>  
    </xsl:fallback>  
</xsl:loop>
```

88



USAGE

Exemple:

```
<xsl:if test="artist="">  
    <xsl:message terminate="yes">  
        Error: Artist is an empty string!  
    </xsl:message>  
</xsl:if>
```

xsl:if, xsl:choose, test

- Traitements conditionnels

- On utilise <if> (équivalent du if classique mais sans else)

- <xsl:if test="*XPath_expr*">

- pour filtrer en fonction d'une opération logique,
ex: <xsl:if test="position()<10">...code à analyser / reproduire...</xsl:if> (top 10 !)

- Ou <choose><when><otherwise> (équivalent du switch, case, default) :

```
<xsl:choose>
  <xsl:when test="prix &gt; 10">
    <td bgcolor="#ff0000"><xsl:value-of select="price"/></td>
  </xsl:when>
  <xsl:when test="price &lt; 8">
    <td bgcolor="#ffff88"><xsl:value-of select="price"/></td>
  </xsl:when>
  <xsl:otherwise>
    <td><xsl:value-of select="price"/></td>
  </xsl:otherwise>
</xsl:choose>
```

Remarque : les fichiers XML devant être bien formés, l'usage d'expressions conditionnelles nécessite souvent l'emploi de code redondant



xsl:for-each, boucles & traitement itératif

- Traitements itératifs (boucles)

- Pour simuler les boucles XSL-T on utilise la balise `<for-each>`, exemple :
 - `<xsl:for-each select="groupe/personne">` traite toutes les personnes du groupe
- For-each possède la faculté de placer le contexte actuel au nœud en cours de parcours pour les appels Xpath par chemins relatifs
- Rappel : On peut aussi procéder par `<template match="">` et `<apply-templates>` avec des templates nommés ou non !

Sortable avec xsl:sort
en 1ères balises



- Les conteneurs à **contenu variable**

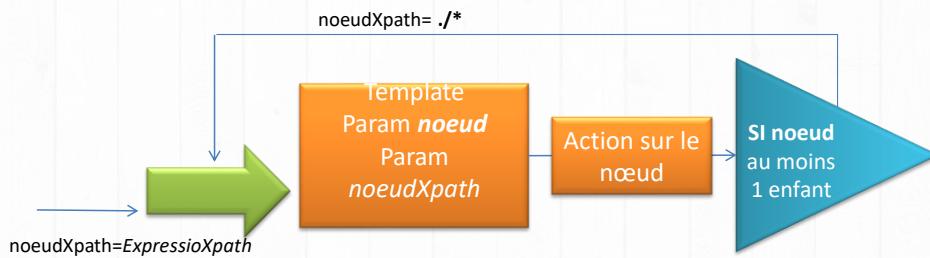
- Au cours de transformations (ex: XSLT) un document peut voir le contenu de ses conteneurs modifié par les **boucles de répétition** (ex: `<xsl:for-each ...>`)
- Dans ce cadre, il faut s'assurer que le code générant le document de sortie à bien connaissance des obligations liées au schéma afin d'éviter l'invalidation du document de sortie

```
<xsl:for-each select="/node">
    <tr><td>...</td>    </tr>
</for-each>
```



Boucles récursives & traitement itératif

- Boucles récursives par parcours
 - Le but est de parcourir tous les descendants d'un nœud sélectionné par une expression *Xpath* passée en **param** d'un template



92

USAGE :

```
<xsl:template name="recurs">
    <xsl:param name="compteur"/>
    <xsl:param name="max"/>
    <!-- traitement -->

    <xsl:if test="$compteur < $max">
        <xsl:call-template name="recurs">
            <xsl:with-param name="max" select="$max"/>
            <xsl:with-param name="compteur" select="$compteur + 1"/>
        </xsl:call-template>
    </xsl:if>
</xsl:template>
```

Xsl:sort, trie de branches

- XSL nous offre dès la version 1.0 une balise pour trier les arbres et permettre un parcours ordonné
- Plusieurs modes de tri sont disponibles suivant les types de données afférentes au tri
- On peut trier les nœuds sélectionnés sur un ou plusieurs critères à l'aide de la balise `<xsl:sort>`. On peut trier sur plusieurs champs en mettant plusieurs `<sort>`

- ex:

```
<xsl:for-each select="emprunteurs/personne">
  <xsl:sort select="adresse/cp" order="descending" />
  <xsl:sort select="nom" />
  <xsl:sort select="prenom" />
  <tr>
    <td><xsl:value-of select="adresse/cp"/></td>
    <td><xsl:value-of select="nom"/></td>
    <td><xsl:value-of select="prenom"/></td>
  </tr>
</xsl:for-each>
```

93



Prototype :

```
<xsl:sort select="nœud" data-type="dtype" order="orderVal" lang="nmtoken" case-order="cOrderValue" />
```

Nœud : correspond au nœud à trier

dtype : type de donnée pour le tri

-text	➔ type texte
-number	➔ type numérique
-elt	

orderValue : ordre de tri

-ascending	➔ croissant
-descending	➔ décroissant

cOrderValue : choix de tri pour la case (string)

-upper-first	
-lower-first	

Xsl:sort, trie de branches

- Sur les **<xsl:for-each>**
- mais aussi sur les **<xsl:apply-template/>**
 - Le template sera exécuté en recevant les nœuds dans l'ordre choisi et non l'ordre de parcours xml ...
- ex:

```
<xsl:apply-templates select="clients/client">
  <xsl:sort select="nom" order="descending" />
  <xsl:sort select="prenom" />
</xsl:apply-templates>
```

94



Prototype :

```
<xsl:sort select="nœud" data-type="dtype" order="orderVal" lang="nmtoken" case-order="cOrderValue" />
```

Nœud : correspond au nœud à trier

dtype : type de donnée pour le tri

- text ➔ type texte
- number ➔ type numérique
- elt

orderValue : ordre de tri

- ascending ➔ croissant
- descending ➔ décroissant

cOrderValue : choix de tri pour la case (string)

- upper-first
- lower-first

xsl:number

• Les nombres en XSL-T

- On peut utiliser `<number>` pour numérotter automatiquement des noeuds selon un format préétabli (ex: création de sommaire, de table des matières ou d'index)

```
<xsl:for-each select="bibliotheque/livre">
  <tr>
    <td><xsl:number level="single" count="livre" from="/" format="A"/></td>
    <td><xsl:number level="single" count=" livre" from="/" format="1"/></td>
    <td><xsl:number level="multiple" count="livre|personnage" from="/" format="A.1"/></td>
    <td><xsl:value-of select="titre" /></td>
    <td><xsl:value-of select="date_emprunt" /></td>
  </tr>
</xsl:for-each>
```

- On peut aussi formater les nombres à l'affichage :
 - exemple de définition et de formatage de type monétaire français

```
<!--définition-->
<xsl:decimal-format name="curFR" decimal-separator="," grouping-separator=" " />

<!--utilisation-->
<xsl:value-of select="format-number(price, '# #0,00', 'curFR')"/>
```



xsl:import, xsl:include

- **Modularisation via la balise <import>,<include>**

on peut importer du xsl (ex: bibliothèques de templates nommés et paramétrés, avec liste de paramètres). Il existe 2 manières de déclarer du contenu xsl externe à la page en cours, l'import et l'include. Ces 2 méthodes ont une légère nuance de fonctionnement

- **Import** : Priorité basse, en cas de conflit, les éléments définis dans la feuille appelante restent prioritaires

- Ex import :

```
<xsl:import href="import_rules.xsl" />
```

- **Include** : Priorité haute en cas de conflit, la feuille appelée sera prioritaire

- Ex include :

```
<xsl:include href="import_rules.xsl" />
```



Fonctions XSL/xpath

- Les fonctions pour XSLT

- **current()** renvoie le nœud courant

```
<xsl:value-of select="current()"/>
```

- **document('doc.xml')[,NodeSet]**) renvoie le nœud racine ou nodeset du document doc.xml

```
<xsl:value-of select="document('doc.xml')[, NodeSet] " />
```

- **element-available(str)** renvoie vrai si le processsor prend en charge la balise nommée par str

```
<xsl:if test="element-available('xsl:delete')">  
    Prise en charge du xsl:delete  
</xsl:if>
```

- **function-available(str)** idem que element-available mais pour les fonctions

- **generate-id(NodeSet)** permet de générer un id unique pour le NodeSet



XSL Applications courante

- Il est très courant que plusieurs fichiers xml soient nécessaires au traitement xsl que vous souhaitez effectuer. Ex: liste d'éléments faisant référence à des éléments présents sur un autre fichier xml que celui chargé
- Pour garder un coté logique a l'appel de transformation nous allons dissocié les données en plusieurs types les fichiers xml :
 - Données référencées
 - Document d'appel unique
- Dans le xsl nous accéderons aux nœuds grâce au document(uriXmlLocation.xml) qui nous place a la racine du document placé à uriXmlLocation

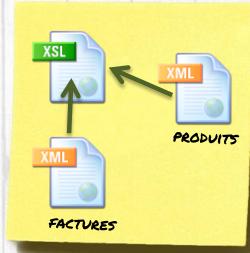


- Transformation de structure xml → rss
- Imaginons 2 bases de données contenant les mêmes données mais avec une structure différente, auxquelles nous voulons permettre l'intégration par xml des données de l'un vers l'autre et inversement.
- Les balises suivantes remplissent cette mission
 - **Xsl:element** : permet de créer une balise de sortie en ne se souciant ni de l'ouverture, ni de la fermeture
 - **Xsl:attribut** : (enfant de xs:element) permet de fixer un attribut à l'élément dont il est l'enfant



XSL - Exercice multi documents

- Créer une feuille de facture basée sur 2 xml
 - Document facture avec les idclients
 - Document client avec toutes les informations clients
 - L'appel de transformation par le navigateur se fera par ou avec le document facture.xml
 - le xsl chargera le fichier grâce à la fonction document(uri)
 - afficher les infos clients pour chaque factures



100



Trucs & Astuces

- Où puis je mettre une expression Xpath?
 - Pour bien identifier les attributs ou une expression Xpath est attendu en xsl
 - 1. **Je suis pas dans un attribut?**
 - » xsl:value-of, xsl:for-each,...
 - 2. **La balise est une balise XSL?**
 - » **OUI**, → question suivante
 - » NON, mettre des **accolades** pour évaluer la valeur vers la sortie
 - 3. **L'attribut est :**
 - **select=..**
 - **test=..**
 - **match=..**
 - » **Xpath supporter, ex... action XSL**
 - **un-autre-attrib=..**
 - » utiliser des accolades pour évaluer l'expression

SEUL CERTAINS ATTRIBUTS REQUIÈRENT OU ACCEPTENT DU XPATH !!!



Le langage XSL-T

- On peut aussi utiliser XSL-T pour générer des XSD dynamiques (règles dynamiques selon le contenu)
- Les modèles de contenu extensible
 - il existe des design patterns pour les schémas classiques et des catalogues, mais aussi des modèles de contenu extensibles (en XSL-T ou via un Schematron)
- Étendre un schéma XML à l'aide d'un Schematron*
 - Schematron : langage permettant d'effectuer des assertions concernant la validité d'un document XML à l'aide des expressions XPath
 - Inconvénients : le Schematron rend la spécification de la structure d'un document XML difficile à lire et à gérer



XSL & TRAITEMENT GÉNÉRIQUE DE NOEUDS

◀ 103 ▶

Gérer le match=« * »

- Dans certains XML les noms des noeuds sont agénérique (chaque noeud possède un nom propre)
 - Difficulté de traitement générique
 - » Ex:
 - <balise1Text type=«txt»/>
 - <balise2Rad type=«rad»/>
 - Match complexe plein d'unions ‘|’(pipe)
 - » Match=«balise1Text | bailse2Txt»
 - Maintien difficile
 - Généricité proche de zéro



Gérer le match=« * »

- Dans certains XML les noms des noeuds sont agénérique (chaque noeud possède un nom propre)
 - Difficulté de traitement générique
 - » Ex:
 - <balise1Text type=«txt»/>
 - <balise2Rad type=«rad»/>
 - Traitement générique du nom de balise
 - » match=«*[contains(name(),'text')]»
 - Maintient plus facile (mais pas optimum)
 - Dépend d'une bonne convention de nommage



Gérer le match=« * »

- Dans certains XML les noms des noeuds sont agénérique (chaque noeud possède un nom propre)
 - Difficulté de traitement générique
 - » Ex:
 - <balise1Text type=«txt»/>
 - <balise2Rad type=«rad»/>
 - Chercher des cohérences dans un attribut commun aux balises
 - » Match=« *[@type='txt'] »
 - Maintient **plus facile**
 - **Dépend d'attribut** pour la généricté du contenu



XSL → OFFICE OPEN XML

XML et l'offre bureautique

- XML et suites logicielles
 - **Open Office** utilise la norme XML "**OpenDocument**" pour décrire ses fichiers
 - **Microsoft Office** utilise la norme "**OpenXML**" (.docx = fichiers XML zippés)
- Pourquoi l'utilisation de XML en bureautique ?
 - Reproches de faible portabilité et automatisation des formats propriétaires (en particulier compilés)
 - L'utilisation de XML facilite l'interopérabilité des programmes (standard ouvert, API existantes et facilité d'import-export entre formats différents)
 - Les versions compressées de XML sont souvent moins lourdes que leurs concurrents (plus facile à stocker, échanger...)
 - L'exploitation directe des ressources bureautiques en XML permet d'éviter la duplication des ressources (et donc les risques d'erreurs de reproduction)



XML et l'offre bureautique

- XML et bureautique chez Microsoft

- Office 2003 :

- WordProcessingML ou WordML (Word)
 - SpreadsheetML (Excel)
 - Documents XML plain text monolithique (volumineux)

- Office 2007

- Office Open XML version ECMA (ECMA-376)
 - SP2 proposera ODF (lecture, écriture et choix de format par défaut)

- Office 14

- Véritable implémentation d'Office Open XML version ISO (ISO/IEC IS 29500)



XML et l'offre bureautique

- L'offre Office de Microsoft : Office Open XML (Open XML ou OOXML)
 - Créé par Microsoft
 - Standard validé par l'ECMA (2006)
 - Adopté comme norme ISO (2008)
 - Concurrent de la norme OpenDocument
 - Développé à la base pour remplacer les formats binaires de la suite MS Office
 - Open XML est l'appellation commune pour OOXML
- Quelques caractéristiques d'OOXML
 - Possibilité d'extension avec des schémas XML
 - Utilisation forte d'éléments y compris pour les métadonnées
 - Pas d'éléments mixtes



XML et l'offre bureautique

- Critiques d'OOXML

- Un format standard bureautique existe déjà
- DrawingML utilisé plutôt que le standard SVG
- Open Math ML plutôt que le standard MathML
- Pas de langage de macro
- Complexité et longueur parfois injustifiées
(définition de couleur et alignement différent pour les documents texte, les tableurs et les présentations)



XML et l'offre bureautique

- XML Open Document Format (ODF)
 - Premier format standard ouvert de bureautique publié par OASIS
 - Norme ISO (ISO26300:2006) depuis 2006
 - Inspiré par la suite OpenOffice (concurrente de MS Office)
 - Contrôlé par un organisme indépendant
 - Extension de fichier : .odt (texte), ods (tableur)...
 - Séparation données (content.xml), présentation (styles.xml) et métadonnées (meta.xml) forte
 - Prévu pour améliorer l'interopérabilité entre bureautique et système d'information



XML et l'offre bureautique

- Critiques d'ODF

- Pas de langage de macro spécifié (différent selon les éditeurs)
- Pas de tableaux dans les présentations
- Standard parfois imprécis
- Pas de signature numérique dans la norme

- Exercice de compréhension

- Créez un document texte simple (quelques phrases) à l'aide d'Open Office Writer (.odt) ou de Microsoft Office Word 2007 (.docx)
- Renommez-le puis décompressez-le !
- Analysez la structure et le contenu du fichier content.xml
- Jetez un œil aux filtres XSL sous Open Office et testez les transformations de formats



XML et l'offre bureautique

- OOXML vs ODF
 - Tous les deux basés sur des fichiers XML séparés et archive Zip
 - Normes assez récentes (2006) opposant des géants du logiciel
 - Problèmes de deux normes ISO pour un même but
 - Véritable guerre des formats
 - Enjeux considérables suivant l'adoption d'un format
 - Le format ODF aurait gagné la bataille
 - ODF plus lisible humainement (proche de HTML)
 - ODF utilise d'autres standard : MathML, SVG, XLink
 - OOXML : balises plus courtes, moins d'attributs
 - Gain de place et rapidité de traitement (à relativiser à cause du Zip)
 - Plus de balises nécessaires et peu d'imbrication d'éléments permises

114



Pour en savoir plus sur la comparaison :

http://www.freesoftwaremagazine.com/articles/odf_ooxml_technical_white_paper?page=0%2C0

XML et l'offre bureautique

- Dissociation des types de contents dans différents fichiers XML
 - Style
 - Text
 - Images / multimedia
- Relations par ID
- Prise en charge des formules pour Excel
 - Plages de formules relatives ou absolues
- Compression global du contenu
 - Methode zip



XSL Office, Structure Word

- Structure du fichier de contenu

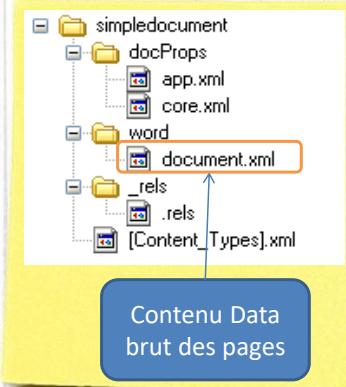
- *document.xml*

- Balise ROOT : *<w:document>*
 - *Corps de page: <w:body>*
 - *Balise de bloc de text : <w:p>*

- *Medias dans document.docx/word/media*

- *Definition des id des medias dans document.docx/word/_rels/document.xml.rels*

Ex: <Relationship Id="rId8" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="media/image2.png"/>



116

Balise *<w:document>* avec les espaces de nom nécessaires : *<w:document xmlns:ve="http://schemas.openxmlformats.org/markup-compatibility/2006" xmlns:o="urn:schemas-microsoft-com:office:office" xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships" xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:wp="http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing" xmlns:w10="urn:schemas-microsoft-com:office:word" xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main" xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml">*

XSL pour Ms Word © OpenXML SDK & .NET (C#/VB)

- Les « *using* »

- Fonctionnalités XML/XSL 1.0 (full), 2.0(partiel)

Code C#:

```
using System.Xml;  
using System.Xml.Xsl;
```

- Fonctionnalités de documents OpenXML Ms Word

Code C#:

```
using DocumentFormat.OpenXml.Packaging;  
using DocumentFormat.OpenXml.Wordprocessing;
```



XSL pour Ms Office

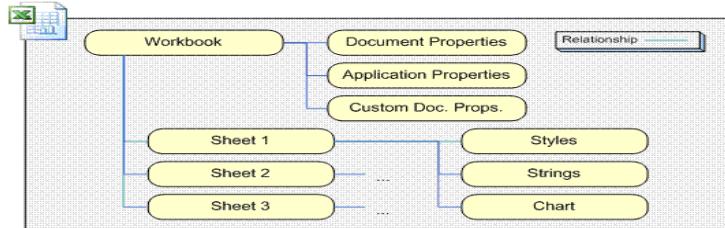
Logique de transformation XMLdocx

- Edition d'un model à peupler
 - Modèle vide pour création des styles dans le docx
- Peuplement du document avec xsl
 - XSL 1.0
 - Contenu du fichier document.xml dans `<template match="/">`
 - *Dynamisation des informations par*
 - *Value-of* Expressions Xpath
 - *For-each* ...
- Compilation du document de sortie
 - C# / VB
 - .NET(4.0 min)

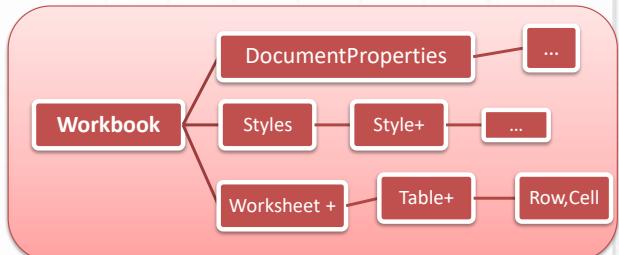


5.XSL pour Excel XML

- Structure d'un document Office Excel® xlsx



- Structure d'un document Office Excel® xml



Msdn : [https://msdn.microsoft.com/fr-fr/library/Aa338205\(v=office.12\).aspx](https://msdn.microsoft.com/fr-fr/library/Aa338205(v=office.12).aspx)

XSL pour Excel XML

- Propriétés du document
- Meta données

Code xml :

```
<DocumentProperties  
    xmlns="urn:schemas-microsoft-com:office:office">  
        <Author>Someone</Author>  
        <LastAuthor>champix</LastAuthor>  
        <Created>2015-03-15T23:04:04Z</Created>  
        <Company>Eaton Corporation</Company>  
        <Version>12.00</Version>  
</DocumentProperties>
```

120



XSL pour Excel XML

- La syntaxe des styles

```
<Styles>
    <Style ss:ID="Default" ss:Name="Normal">
        <Alignment ss:Vertical="Bottom" />
        <Borders />
        <Font ss:FontName="Arial" />
        <Interior />
        <NumberFormat />
        <Protection />
    </Style>
</Styles>
```



XSL pour Excel XML

- Feuille du classeur

```
<Worksheet ss:Name="{$sheetName}">
    <Table>
        <Row ss:Index="3">
            <Cell ss:Index="2">cell C3</Cell>
        </Row>
    </Table>
</Worksheet>
```



Références des feuilles, une feuille non
referenciee ne seras pas visible dans
excel

XSL pour Excel XML

- Insertion de formules :

```
<Cell  
ss:Index="11"  
ss:Formula="=SUM(R[-5]C:R[-3]C)">
```

- Positionnement relatif à une cellule

```
R[offset]
```

Conclusion

❑ Dans cette partie vous avez vu :

- ✓ La logique Excel, word powerpoint
- ✓ Comment créer un document office xml
- ✓ Comment sont structurés les fichiers pptx, docx, xlsx
- ✓ Comment mettre en place du xsl pour excel, word et powerpoint



- Définition
- Syntaxe
- Régions
- Structure
- Format de page
- Fo:page-sequence
- Fo:inline / fo:block
- Tableaux
- APACHE FOP

6. XSL-FO

Le langage XSL-Fo

Définition

- Définition
 - XSL-FO = eXtensible Stylesheet Language - Formatting
 - Langage de formatage des données (travaux de presse)
 - Recommandation du W3C
 - Pas de DTD ni de Schéma précisés d'où une confusion (cf. XSD XEP)
- Fonctionnement
 - Un document XML (données) fusionne avec un document XSL-T via un processeur XSL-T qui génère un fichier XSL-FO qui passe par un processeur XSL-FO (ex: FOP) pour être traduit en document formaté (ex: PDF)
 - Le fonctionnement dépend donc fortement du processeur FO (ex: FOP)



Incompatibilité des processeurs XSL-FO :

Du fait du manque de DTD/XSD pour XSL-FO, les processeurs XSL-FO sont très différents. Il en ressort que même le plus utilisé FOP est incapable de formater le même code XSL-FO avec 2 versions différentes de processeur FO. Ainsi, les modèles XSL-FO doivent être créés spécifiquement pour chaque processeur et ne sont en aucun cas interchangeables. D'où une nécessité de forte adaptation ou de spécialisation des programmeurs.

Syntaxe fo

- Syntaxe
 - Précisez `<?xml encoding="iso-8859-1" ... ?>` (problèmes d'accents)
 - L'extension des fichiers XSL-FO est ".fo"
 - Le namespace est : `xmlns:fo="http://www.w3.org/1999/XSL/Format"`
 - La racine XSL-FO est `<fo:root>`
 - On utilise des **flux** `<flow>` pour envoyer des données, comprenant :
 - Des éléments `<block>` équivalent de `<div>` (éléments block en CSS : p, div...)
 - Des éléments `<inline>` équivalent des lignes simples (élément inline en CSS)
 - Attention à l'attribut flow-name qui prend la valeur "xsl-[nom_de_region]" ex: xsl-region-body au lieu de simplement region-body
 - On peut utiliser des `<listes>`, `<table>` ou `<caption>` et/ou `<table-and-caption>`, `<table-cell>`, `<table-row>`, `<table-column>`, `<table-header>`, `<table-body>`...
 - Pour générer du XSL-FO a partir du XSL-T on utilise simplement leurs namespaces respectifs et on génère le XSL-FO à partir du XSL-T et du XML

127

Exemple de page simple XSL-FO :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

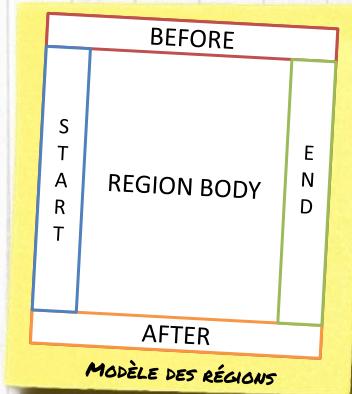
<fo:layout-master-set>
  <fo:simple-page-master master-name="my_page_ref" margin-top="1.8cm" margin-bottom="1.8cm" margin-left="2.5cm" margin-right="1.8cm">
    <fo:region-body margin="1cm" />
    <fo:region-before extent="2cm" />
    <fo:region-after extent="2cm" />
    <fo:region-start extent="2cm" />
    <fo:region-end extent="2cm" />
  </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference=" my_page_ref ">
  <fo:flow flow-name="xsl-region-body">
    <fo:block font-size="12pt" font-family="sans-serif" color="blue" space-before="5mm">Hello World !</fo:block>
  </fo:flow>
</fo:page-sequence>
</fo:root>
```

Régions fo

• Régions

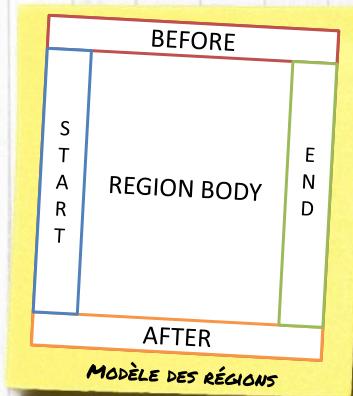
- Notons qu'un document XSL-FO contient en général plusieurs modèles de page (ex: page de garde, sommaire, page simple, annexe...)
- Chaque page créée est découpée en régions
 - **Body** : corps de la page
 - **Before** : l'en-tête de page
 - **After** : le pied de page
 - **Start** : bordure de gauche (ou de droite en bidimode)
 - **End** : bordure de droite
- Pour chacune de ces régions, on peut définir des marges, des bordures, un fond et un contenu
- Mais rappelons-nous que tous ces éléments, sont fortement dépendants du processeur.



Régions fo

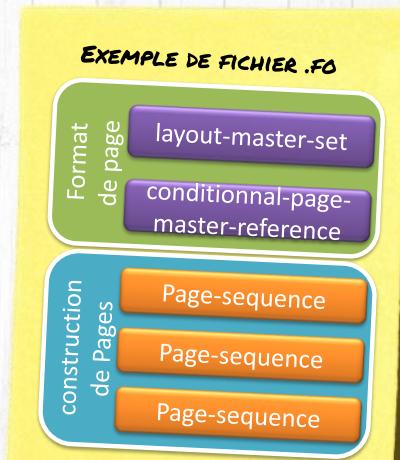
Trucs & astuces

- J'écris de haut en bas
 - **Before** : Avant de commencer ma page en haut
 - **After** : Apres avoir remplis ma page, en **bas**
- J'écris de gauche à droite
 - **Start** : je Démarre une ligne, à **gauche**
 - **End** : je Finis ma ligne, a **droite**
- **Body** est (par défaut) l'ensemble de la page régions comprises



structure fo

- Il existe un ordre particulier à la constitution d'un document FO.
- Une fois les différents **formats de pages** ainsi que leurs spécificités **déclarés**,
- les **différentes pages** de notre document feront nécessaire **référence au format de page** précédemment créé.



Format de pages FO

- Déclaration des formats de pages
 - On déclare une suite de formats de page avec **Layout-master-set** puis différents formats de pages
 - **Simple-page-master-set** : format de page simple
 - » **master-name="NomFormatPage" page-width="210mm" page-height="297mm"**
 - **Page-sequence-master** : séquence de formats de page
 - » **master-name="NomDeLaSequenceDeFormat"**
 - **Repeatable-page-master-alternatives** séquence de formats alternatif et conditionnel (cf. déclaration de séquence de format page conditionnel)
 - **conditional-page-master-reference**
 - » **master-reference="NomDeFormatRef" odd-or-even="even" page-position="last" blank-or-not-blank="blank"**

131



Usage :

```
<fo:layout-master-set>
  <fo:simple-page-master page-height="297mm" page-width="210mm" master-name="A4portraitCouv">
    <fo:region-body background-image="url('file:///C:/XMLPHOTO/img/1R.jpg')"/>
    <fo:region-before extent="1cm"/>
    <fo:region-after extent="1cm"/>
  </fo:simple-page-master>
  <fo:simple-page-master page-height="297mm" page-width="210mm" master-name="A4PortraitReport">
    <fo:region-body/>
  </fo:simple-page-master>
  <fo:simple-page-master page-height="297mm" page-width="210mm" master-name="A4portraitLeft">
    <fo:region-body background-image="url('1R.jpg')"/>
    <fo:region-before extent="5mm"/>
    <fo:region-after extent="5mm"/>
    <fo:region-start background-color="GREY" extent="1cm"/>
  </fo:simple-page-master>
  <fo:simple-page-master page-height="297mm" page-width="210mm" master-name="A4portraitRight">
    <fo:region-body background-image="url('file:///C:/XMLPHOTO/img/1R.jpg')"/>
    <fo:region-before extent="5mm"/>
    <fo:region-after extent="5mm"/>
    <fo:region-end background-color="GREY" extent="1cm"/>
  </fo:simple-page-master>
  <fo:page-sequence-master master-name="A4PortraitAvecAnneaux">
    <fo:repeatable-page-master-alternatives>
      <fo:conditional-page-master-reference blank-or-not-blank="blank" page-position="last" odd-or-even="even" master-reference="A4PortraitCouv"/>
        <fo:conditional-page-master-reference odd-or-even="even" master-reference="A4portraitRight"/>
        <fo:conditional-page-master-reference odd-or-even="odd" master-reference="A4portraitLeft"/>
    </fo:repeatable-page-master-alternatives>
  </fo:page-sequence-master>
</fo:layout-master-set>
```

Format de pages

- Déclaration des régions dans les formats de page
 - region-body sert à définir la région body
 - *Region-name=<> NomDeLaRegion <> margin=<>10mm<>*
padding=<>1mm<>
 - region-before, region-after, region-start, region-end
 - *Region-name extent=<> 1cm <>*
 - Différents attributs communs aux régions
 - Arrière-plan : *background ; background-image ; background-repeat;...*
 - Mise en page :
 - *margin[| -left | -right | -top | bottom]*
 - *padding[| -left | -right | -top | bottom]*
 - *border[| -left | -right | -top | bottom];*
 - *border-color; border-style*

132

-La région body doit être définie en premier avant les différentes régions

-region-before, region-after, region-start, region-end : leur taille est définie par la valeur de l'attribut extent

-L'attribut region-name sert à appeler directement une région par son nom lors de l'écriture du flux dans la page.

Format de pages fo

- Déclaration de séquence de format page conditionnel
 - La déclaration de séquence de pages permet par exemple de recréer le comportement des marges d'un livre classique en définissant une page sur 2 la marge à gauche étendue puis la marge à droite étendue pour prendre en considération le format livre et l'espace central perdu
 - L'alternance se définit dans la balise `conditional-page-master-reference` par l'attribut suivant :
 - **Odd or even : odd** = page impaire & even = page paire
 - Les attributs suivants permettent de compléter la condition :
 - **page-position="last"** => si la page est en dernière position
 - **blank-or-not-blank="blank"** => si la page doit être vide

133

Usage :

Cf. commentaire déclaration de formats page pour les formats initiaux en référence dans la séquence

```
<fo:page-sequence-master master-name="A4PortraitAvecAnneaux">
  <fo:repeatable-page-master-alternatives>
    <fo:conditional-page-master-reference blank-or-not-blank="blank"
      page-position="last" odd-or-
      even="even"
      master-
      reference="A4PortraitCouv"/>
      <fo:conditional-page-master-reference odd-or-even="even" master-
      reference="A4portraitRight"/>
      <fo:conditional-page-master-reference odd-or-even="odd" master-
      reference="A4portraitLeft"/>
  </fo:repeatable-page-master-alternatives>
</fo:page-sequence-master>
```

fo:page-sequence

- Définition des pages générées par xsl-fo
 - Pour créer une page on la déclare avec : **page-sequence**
 - » **master-reference=« NomDuFormatDeReference »** sert à définir le format de page à utiliser pour la page en cours
 - Pour écrire dans les régions non-body : static-content
 - Pour écrire un flux de block dans le body : **flow**
 - Ces 2 balises nécessitent le nom standard ou défini de la région où insérer les blocks
 - Régions définies par xsl: xsl-region(-body|-start|-end|-before|-after
 - » **Ex : flow-name=« xsl-region-before »**

134

- 1 seul flow est attendu dans la page séquence (règle de l'espace nom et à la structure associé)
- Les écritures dans les régions : start, before, end, after seront définies dans la balise static-content et non dans un flow (sauf si elle est la seule région écrite)
- Le static-flow est attendu avant le flow d'écriture

fo:inline vs fo:block

- Tout comme le HTML et ses « div » le xsl-fo n'échappe pas à l'empilement des ensembles.
- L'équivalent en fo de div, span : **block, inline**
 - » block : disposition empilé, peut contenir des éléments bloc et en-ligne
 - » inline : disposition en ligne, ne peut contenir que des éléments en-ligne, pas d'élément bloc
- Les ensembles block et inline peuvent s'imbriquer
 - Block peut contenir des éléments enfants de type block ou de type in-line
 - Inline ne peut contenir que des éléments inline

135



Usage :

```
<fo:block>
<fo:block margin-top="1cm" margin-left="10%">
  <fo:inline height="auto" width="60%" margin-left="10%">
    ...
  </fo:inline>
  <fo:inline height="auto" width="60%" margin-left="10%">
    <fo:block>... </fo:block>
    <fo:block>...</fo:block>
  </fo:inline>
</fo:block>
</fo:block>
```

Fonts de texte

- Soulignement
text-decoration="underline"
- Hauteur de ligne
line-height = "15pt"
- Taille police
font-size="30pt"
- Poids de font
font-weight="100->900 | normal | bold | bolder"
- Style de font
font-style="italic| oblique | normal"
- Famille de font
font-family="Calibri | fontName"
- Font *(super)*
font = "italic bold 12pt Calibri"

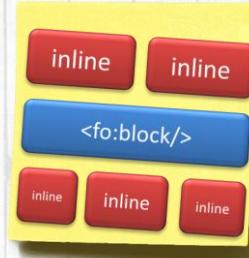


Les sauts

- Saut de ligne

`<fo:block/>`

- Un block vide provoque un saut de ligne



- Saut de page

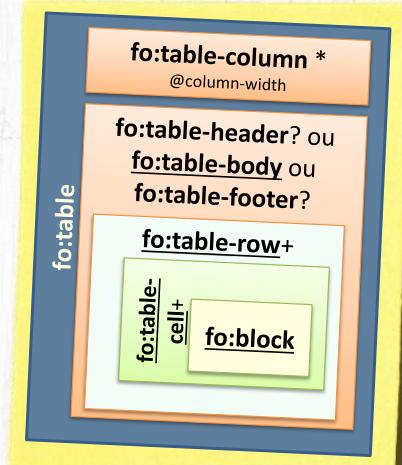
`page-break-after=""`

`page-break-before=""`

- ***always*** : saute une page
- ***avoid*** : évite les sauts de pages
- ***left*** ou ***right***: définit la position gauche ou droite de la page

Tableaux

- Les tableaux :
 - Le comportement des balises de tableaux est proche du comportement des balises du html
 - structure
 - **table-column** sert à définir les colonnes et leurs tailles
 - column-width="25mm"
 - **Table-header** sert à définir l'en tête du tableau
 - répéter sur chaque nouvelle pages physique pour le tableau
 - ↳ On y insère une ou des lignes
 - **Table-body** sert à définir le corps du tableau
 - Obligatoire
 - ↳ On y insère une ou des lignes
 - **Table-footer** sert à définir le pied du tableau
 - répéter sur chaque nouvelle pages physique pour le tableau
 - ↳ On y insère une ou des lignes
 - Création de lignes
 - » **Table-row** : définit une ligne
 - ↳ On y insère un ou des **cells**
 - » **Table-cell** : définit une cellule
 - ↳ On y insère un **block**



138

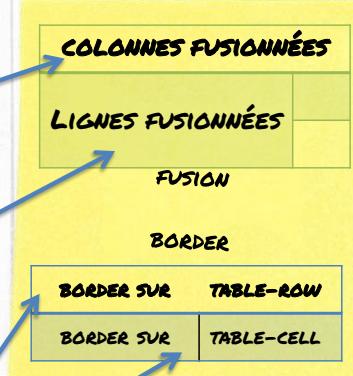
Exemple de tableau en XSL-FO:

```
<fo:table-and-caption>
<fo:table>
  <fo:table-column column-width="25mm"/>
  <fo:table-column column-width="25mm"/>
  <fo:table-header>
    <fo:table-row>
      <fo:table-cell>
        <fo:block font-weight="bold">Car</fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block font-weight="bold">Price</fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-header>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell>
        <fo:block>Volvo</fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>$50000</fo:block>
      </fo:table-cell>
    </fo:table-row>
    <fo:table-row>
      <fo:table-cell>
        <fo:block>SAAB</fo:block>
      </fo:table-cell>
      <fo:table-cell>
        <fo:block>$48000</fo:block>
      </fo:table-cell>
    </fo:table-row>
  </fo:table-body>
</fo:table>
</fo:table-and-caption>
```

FOP gère mal le table-and-caption et seul l'ensemble contenu du noeud table est nécessaire
Nous n'utiliserons donc que table

Tableaux

- Les tableaux :
 - Le comportement des balises de tableaux est proche du comportement des balises du html
 - Fusion des cellules
 - **number-columns-spanned="nbCol"** sert à définir le nombre de colonnes pour étendre de la cellule horizontalement
 - ↳ On y insère un chiffre
 - **number-rows-spanned=""** sert à définir le nombre de colonnes pour étendre de la cellule verticalement
 - ↳ On y insère un chiffre
 - Border placées sur :
 - » **fo:table** définit les bordures autour du tableau
 - » **fo:table-row**: définit les bordures autour de la ligne
 - » **fo:table-cell** : définit les bordures autour de la cellule



139

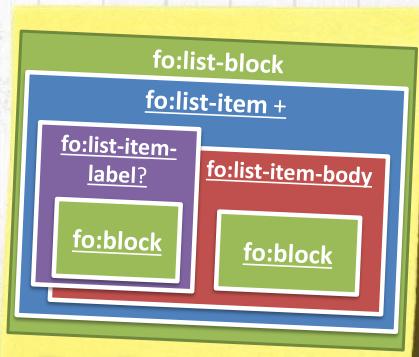
Exemple de tableau en XSL-FO:

```
<fo:table-and-caption>
  <fo:table>
    <fo:table-column column-width="25mm"/>
    <fo:table-column column-width="25mm"/>
    <fo:table-header>
      <fo:table-row>
        <fo:table-cell>
          <fo:block font-weight="bold">Car</fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block font-weight="bold">Price</fo:block>
        </fo:table-cell>
      </fo:table-row>
    </fo:table-header>
    <fo:table-body>
      <fo:table-row>
        <fo:table-cell>
          <fo:block>Volvo</fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block>$50000</fo:block>
        </fo:table-cell>
      </fo:table-row>
      <fo:table-row>
        <fo:table-cell>
          <fo:block>SAAB</fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block>$48000</fo:block>
        </fo:table-cell>
      </fo:table-row>
    </fo:table-body>
  </fo:table>
</fo:table-and-caption>
```

FOP gère mal le table-and-caption et seul l'ensemble contenu du noeud table est nécessaire
Nous n'utiliserons donc que table

Listes, type puces

- Les tableaux :
 - Le comportement des balises de tableaux est proche du comportement des balises du html
 - structure
 - **list-block** sert à définir la structure principale de la liste
 - ↳ on y insère un ou des **list-item**
 - **list-item** sert à définir la structure d'un seul élément
 - répéter sur chaque nouvelle pages physique pour le tableau
 - ↳ On y insère une un label (ou non) et un body
 - Création d'un item
 - » **list-item-label** : définit la puce en elle même
 - ↳ On y insère un **block**
 - » **list-item-body**: définit une cellule
 - On y insère un **block**



Le **label** et le **body** sont par défaut superposés l'un sur l'autre sur la gauche mais sont frères XML

140

Exemple de tableau en XSL-FO:

```
<fo:table-and-caption>
  <fo:table>
    <fo:table-column column-width="25mm"/>
    <fo:table-column column-width="25mm"/>
    <fo:table-header>
      <fo:table-row>
        <fo:table-cell>
          <fo:block font-weight="bold">Car</fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block font-weight="bold">Price</fo:block>
        </fo:table-cell>
      </fo:table-row>
    </fo:table-header>
    <fo:table-body>
      <fo:table-row>
        <fo:table-cell>
          <fo:block>Volvo</fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block>$50000</fo:block>
        </fo:table-cell>
      </fo:table-row>
      <fo:table-row>
        <fo:table-cell>
          <fo:block>SAAB</fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block>$48000</fo:block>
        </fo:table-cell>
      </fo:table-row>
    </fo:table-body>
  </fo:table>
</fo:table-and-caption>
```

FOP gère mal le table-and-caption et seul l'ensemble contenu du noeud table est nécessaire
Nous n'utiliserons donc que table

page-number & citation

- Numéro de la page en courante
 - *fo:page-number*
- Numéro de la page de séquence
 - *fo:page-number-citation-last*
- Référence de dernier block

```
<fo:page-sequence master-reference="A4P" id="mainSequence">  
...  
</fo:page-sequence>
```

ou `<fo:block id="mainSequence"/>`

- Accès à la citation de fin

```
<fo:page-number-citation-last  
    page-citation-strategy="all"  
    ref-id="mainSequence"/>
```



<http://stackoverflow.com/questions/19267255/how-to-show-page-number-n-of-n-using-xslt-in-pdf-report>

https://www.w3.org/TR/xslfo20/#fo_page-number-citation-last

Le langage XSL-Fo

IVb) Enregistrement Fonts

- Fichier de config FOP

- **fop.xconf**

- Fichier de config FOP

- **Auto-detect**

- permet l'auto-detection
 - des fonts sur le **system** exécutant fop

- déclaration de fonts lambda

- Nécessite la **métrique XML** de la font

- <font metrics-url="file:///c:/xml/fonts/garamond-bold.xml"
kerning="yes" embed-url="file:///c:/Windows/fonts/GARABD.TTF">
 - <font-triplet name="Garamond" style="normal"
weight="bold"/>
 -

```
<renderers>
<renderer mime="application/pdf">
  <fonts>
    <!--declaration des fonts-->
    <auto-detect/>
  </fonts>
</renderer>
</renderers>
```



```
java -cp "../fop-0.93/build/fop.jar;../fop-0.93/lib/serializer-version.jar;\ ..../fop-0.93/lib/commons-logging-version.jar;../fop-0.93/lib/commons-io-version.jar" \
org.apache.fop.fonts.apps.TTFFReader \ /WINDOWS/FONTS/GARA.TTF
fonts/garamond.xml
```

Your Java CLASSPATH must include the fop.jar and the other jar files shown in the example. These files are included with the FOP distribution.

Apache FOP

- FOP
 - Pour générer le fichier, on peut ensuite utiliser FOP ou autre (ex: XEP...)
 - Pour cela, il faut installer une Java Virtual Machine (ex: JVM de Sun)
 - Puis télécharger FOP et l'installer
 - Puis ouvrir une fenêtre de commande et lancer la commande :

```
fop myfile.fo myfile.pdf
```

ou avec les options de fusion xml, xsl et pdf

```
fop -xml x.xml -xsl x.xsl -pdf x.pdf
```

- Intégration sur le site Internet
 - On peut lancer un appel de FOP sur un site en PHP avec system() (éventuellement execute(), exec()...)
 - Ou utiliser des fonctions spécifiquement conçues pour PHP
 - Voir si vous utilisez Java, faire l'installation des composants FOP pour Java



Exemple d'appel de FOP comme programme JAVA externe en PHP :

```
function exec_fop ($f_src, $f_dest=null, $retour=false) {  
    if (is_null ($f_dest)) $f_dest = "C:\...\fop C:\...\myfile.fo ...";  
    $cmd = "java org.apache.fop.apps.Fop ".escapeshellarg ($f_src)." ".escapeshellarg ($f_dest);  
    system ($cmd, $retval);  
    if ($retval == 0) { if ($retour) return file_get_contents ($f_dest); else return true; }  
    return false;  
}
```

Apache FOP alternatives

- Antenna HOUSE
- Ecrion xf servant
 - Partie integrante de XF designer
- nFop
 - Moteur FOP en C#
 - <https://www.codeproject.com/Articles/12682/Creating-PDF-with-nFOP>



Conclusion

❑ Dans cette partie vous avez pu voir comment à partir de données et de xsl, fabriquer un fichier pdf, rtf, ...

- ✓ sa syntaxe, sa structure
- ✓ Les différents concepts de Formatage de pages
- ✓ Tableaux, les listes
- ✓ APACHE FOP



- Définition
- Structure
- Formes basiques
- Formes complexes
- Groupes
- Texte, liens, symboles, images
- Filtres
- Textures
- Animations
- Multi formats

SVG

Définition

- Scalable Vector Graphics (SVG)
 - Le SVG, langage de graphiques vectoriels, sert à générer des graphiques dynamiquement, à les animer et à les rendre dynamiques.
 - C'est un standard du W3C basé sur XML
 - Utilise les styles CSS pour la mise en forme
 - Contrairement à JPEG ou PNG, il n'est pas trame
 - Son principal concurrent est aujourd'hui Flash
 - Il fonctionne correctement sur les navigateurs récents (plugin MSIE natif MFF)
 - Il existe plusieurs normes SVG, la 1.0 et 1.1 sont les plus utilisées avec une préférence pour cette dernière



SVG Définition

- Avantages de SVG
 - Les graphiques peuvent être liés à d'autres ressources via xlink
 - Les SVG sont animés (avec SMIL)
 - Et interactifs car scriptable (on peut faire de véritables interfaces)
 - Les transformations géométriques sont simples (vectoriel) et s'effectuent sans perte : déplacements, agrandissements, fondus, rotations...
 - A un document SVG on peut appliquer plusieurs feuilles de style pour un rendu totalement différent (routes, courbes de niveau, stations, lacs...)
 - SVG intègre le DOM (attention pas le DOMHtml !)



SVG Définition

- Inconvénients de SVG

- Les données XML sont verbeuses, les fichiers SVG sont de grande taille mais avec une compression Zlib,
 - » le format est plus léger que ses concurrents
 - » lisible nativement compressé



<https://zlib.net/>

149 A set of small blue navigation icons typically used in presentation software like Beamer. They include symbols for back, forward, search, and table of contents.

HOW TO : https://zlib.net/zlib_how.html

ZLIB : <https://zlib.net/>

Structure SVG

• Structure SVG

- Le fichier porte l'extension ".svg" et sa DTD est :

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

- La racine des documents SVG est **<svg>**
 - Le xmlns est obligatoire: <http://www.w3.org/2000/svg>
 - **viewBox** définit la fenêtre de contenu
 - **width** et **height** définissent la taille du SVG dans la page

```
<svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%"  
viewBox="0 0 500 500">
```

- Certains navigateurs nécessitent de préciser la version="1.1"
- La section <defs> sert aux définitions (variables, effets, symboles...)

150

Exemple simple de code SVG :

```
<svg width="200" height="200" viewBox="0 0 500 500" version="1.1">  
  <title> exemple SVG </title>  
  <desc> Ceci est un exemple de SVG de base </desc>  
  <defs></defs>  
</svg>
```

svg & css

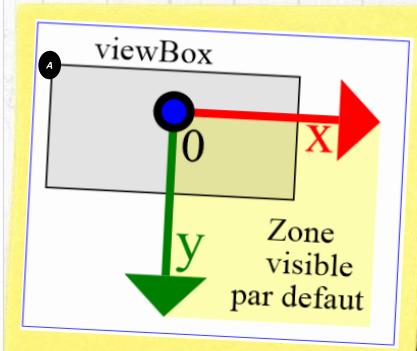
- La balise `<style type="text/css">`
- capacité `@media`
- sélection CSS classique `½ & 3*` si navigateur compatible
 - » `#premierGroup rect{ ... }`
- propriétés css pour déclarer un attribut svg
 - avec le même nom que l'attribut svg
 - exemple :
 - » en css : `fill : red;`
 - » en attribut svg : `fill="red"`

151



SVG & Système de coordonnées

- Point de référence et viewBox
- '0' référence est le coin supérieur gauche
- Pas de d'unité = ordre de grandeur
- viewBox permet d'afficher sur des positions négatives
- **viewBox="Ax Ay width height"**



Formes basiques SVG

- Le dessin svg est assimilable à un jeu de gommettes avec des formes et des tailles variées permettant une fois assemblées de créer des formes complexes
- Les formes de base sont les suivantes :

```
<circle r="100"  
       fill="url(#MyGradient2)"  
       stroke="black"  
       stroke-width="5"  
       cx="150" cy="150"/>
```



```
<ellipse rx="100" ry="150"  
        fill="url(#MyGradient2)"  
        stroke="black"  
        cx="150" cy="200"/>
```



```
<rect  
    width="150" height="200"  
    fill="url(#MyGradient)"  
    stroke="BLACK"  
    x="100" y="100"/>
```



```
<line x1="150" y1="50"  
      y2="200" x2="50"  
      stroke="url(#MyGradient)"  
      stroke-width="10" />
```



Formes SVG de base :

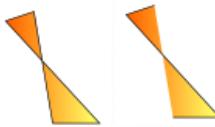
```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">  
<svg width="500" height="500" viewBox="0 0 500 500" version="1.1"  
xmlns="http://www.w3.org/2000/svg">  
  <title> exemple SVG </title>  
  <desc> Ceci est un exemple de SVG de base </desc>  
  <circle r="100px" cx="350px" cy="350px" />  
  <rect x="50" y="50" width="200" height="100" style="fill:red; stroke:blue; stroke-width:10" />  
  <ellipse cx="100" cy="300" rx="50" ry="100" style="fill:#00ff00; stroke:black; stroke-width:1" />  
  <line x1="400" y1="50" x2="350" y2="300" style="stroke:red; stroke-width:3" />  
</svg>
```

Formes complexes SVG

- Il existe aussi des balises pour des formes non définies ou pour des besoins particuliers:

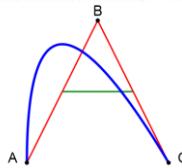
- Le polygone / polylines : forme pleine coordonnées infini

```
<polygon points="170,110  
140,120 240,230 190,230"  
fill="url(#MyGradient)"  
stroke="black"  
stroke-width="1"/>
```



```
<polyline points="170,110  
140,120 240,230 190,230"  
fill="url(#MyGradient)"  
stroke="black"  
stroke-width="1"/>
```

- Le path est un chemin vectoriel aussi appelé trajet ou simplement vecteur. Il permet d'indiquer des mouvements ou des courbures de texte par exemple.



```
<path  
d="M 100 350 q 0 -500 300 0"  
stroke="blue" stroke-width="5"  
fill="none" />
```

154

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">  
<svg width="500" height="500" viewBox="0 0 500 500" version="1.1"  
xmlns="http://www.w3.org/2000/svg">  
    <title> exemple SVG </title>  
    <desc> Ceci est un exemple de SVG de base </desc>  
    <polygon points="170,170 220,170 220,220 170,220" style="fill:blue; stroke:black; stroke-width:1" />  
    <polyline points="225,225 275,225 225,275 275,225" style="fill:none; stroke:black; stroke-width:1" />  
    <path style="fill:red; opacity:0.5; stroke:black; stroke-width:1;" d="M100,100 H200  
        C153 334 151 334 151 334  
        C151 339 153 344 156 344  
        C164 344 171 339 171 334  
        C171 322 164 314 156 314  
        C142 314 131 322 131 334  
        C131 350 142 364 156 364  
        C175 364 191 350 191 334  
        C191 311 175 294 156 294  
        C131 294 111 311 111 334  
        C111 361 131 384 156 384  
        C186 384 211 361 211 334  
        C211 300 186 274 156 274" />  
</svg>
```

Remarques :

Notez le code de l'expression d (data) de <path>, il s'agit de coordonnées de courbes de Bézier (définition de points et tangentes) !

Remplissage et traits

- Remplissage *fill*

- Couleur css

`fill="LIGHTBLUE"`



- Gradient svg

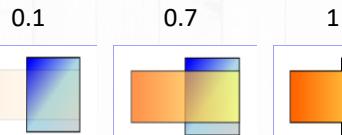
`fill="url(#gradientId)"`



- Opacité

`opacity="value"`

0.1



0.7

1

- Bordures *stroke*

- Couleur

`stroke="BLACK"`



`stroke="#2E8B57"`



- Forme

`stroke-dasharray="value"`

"40,10"



"30,5,10,20"



- Opacité

`stroke-opacity="value"`



≈1

≈0



- Epaisseur

`stroke-width="value"`



155

Les Groupes SVG

- Le groupe **<g>**

- On peut regrouper certains éléments à l'aide de la balise **<g>** (group) pour des raisons de facilité
 - de déplacement
 - d'application d'effets
 - de sélection de groupe

👉 Mettre un **id** et/ou une **class** pour l'identifier au css



Texte, liens, symboles, images

- Textes
 - On peut utiliser la balise **<text>** pour insérer du texte
- Liens
 - On peut créer des liens à l'aide de xlink (pas seulement des liens **<a>**)
 - **<a xlink:href="..."** avec **<svg xmlns:xlink="http://www.w3.org/1999/xlink"...**
- Symboles
 - Pour gagner en temps (load) et poids des fichiers on peut utiliser des symboles
 - On définit un symbole dans **<defs>** avec **<symbol id="mysymbol">** et **<g>**
 - On l'utilise avec **<use xlink:href="mysymbol" x="..." y="..." />**
- Images
 - On peut insérer des images avec **<image xlink:href="myimage.jpg" ... />**

Exemple de création et d'utilisation de symbole :

```
<defs>
  <symbol id="croix" height="25" width="25">
    <g fill="red" stroke="none">
      <rect x="10" width="10" height="25"/>
      <rect y="10" width="25" height="10"/>
    </g>
  </symbol>
</defs>

<use xlink:href="# croix " x="150" y="100"/>
<use xlink:href="# croix " x="250" y="0"/>
<use xlink:href="# croix " x="300" y="300"/>
```

Textures

- Textures
 - Pour remplir les formes avec des motifs évolués, SVG propose les gradients
 - *linearGradient* dégradé linéaire
 - *radialGradient* dégradé radial
 - On les définit dans la partie <defs>
 - On l'applique sur les objets à l'aide de l'attribut **style="fill:url(#mygradient);"**
- Rappel RVB (RGB)
 - RVB = Rouge Vert Bleu (RGB en anglais)
 - Ce sont des spectres d'additions (rouge+vert=jaune et pas vert+jaune=bleu), qui combinées, permettent d'obtenir les couleurs de bases
 - Chaque intensité de couleur va de 0 à 255 (0 = éteint et 255 = allumé au max)

158



Exemples de gradient :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="500" height="500" viewBox="0 0 500 500" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<title> exemple SVG </title>
<desc> Ceci est un exemple de SVG de base </desc>

<defs>
<linearGradient id="lingrad" x1="30%" y1="0%" x2="90%" y2="0%">
<stop offset="0%" style="stop-color:rgb(255,255,0); stop-opacity:1"/>
<stop offset="100%" style="stop-color:#FF0000; stop-opacity:1"/>
</linearGradient>
</defs>

<circle r="100px" cx="350px" cy="350px" style="fill:url(#lingrad);"/>
<rect x="50" y="50" width="200" height="100" style="fill:url(#lingrad); stroke:blue; stroke-width:10;" />
</svg>
```

Exemples de codes combinés :

http://www.w3schools.com/svg/svg_examples.asp

Pour connaître presque tous les éléments SVG :

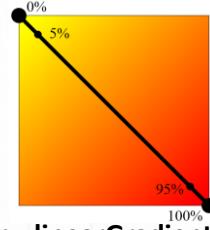
http://www.w3schools.com/svg/svg_reference.asp

Gradient SVG (dégradés)

- Dégradés

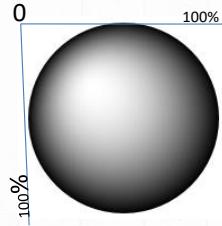
- Transition de couleur et/ou d'opacité est possible grâce aux gradient soit linéaire, soit circulaire (dit radiaux)

```
<defs>
  <lineargradient x1="0" x2="100%"
    y1="0" y2="100%" id="MyGradient">
    <stop offset="5%" stop-color="yellow"/>
    <stop offset="95%" stop-color="red"/>
  </lineargradient>
</defs>
```



• **linearGradient**

- **radialgradient**



```
<defs>
  <radialGradient id="grad1" cx="50%" cy="50%" r="50%" fx="33%" fy="33%">
    <stop offset="0%" stop-color="white" stop-opacity="0"/>
    <stop offset="100%" stop-color="darksilver" stop-opacity="1"/>
  </radialGradient>
</defs>
```

Filtres SVG

- Filtres

- Les filtres sont des effets appliqués à une forme, par exemple un flou, une déviation, une ombre portée... On note, par exemple, les filters suivants :

• feBlend	→ mode d'intersection (cf. logiciels de graphisme)
• feDiffuseLightening	→ lumière diffuse
• feFlood	→ remplissage
• feGaussianBlur	→ flou circulaire
• feMerge	→ fusion
• feOffset	→ décalage de coordonnées dx, dy de la forme de base

- On les définit dans la partie **<defs>** avec **<filter id="myfilter"> <feABC />**
- On l'applique sur les objets à l'aide de l'attribut
style="filter:url(#myfilter);"

160

Exemples de filtres :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="500" height="500" viewBox="0 0 500 500" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<title> Filtres SVG </title>
<desc> Ceci est un exemple de filtres SVG ! </desc>
<defs>
<filter id="blured">
  <feGaussianBlur in="SourceGraphic" stdDeviation="5" />
  <feOffset dx="5" dy="5"/>
</filter>
</defs>
<ellipse cx="100" cy="300" rx="50" ry="100" style="opacity:50%;fill:black; stroke:black; stroke-width:1;filter:url(#blured);"/>
<polyline points="225,225 275,225 275,275 225,275" style="fill:none; stroke:black; stroke-width:1;filter:url(#blured);"/>
</svg>
```

Animations

- Animations

- Pour déclencher des effets visuels, on utilise **<animate>** avec les attributs :
 - "attributeType" permet de préciser sur quel langage on va jouer (ex: CSS | XML)
 - "attributeName" indique quel paramètre on va modifier (ex: **opacity**)
 - "from" et "to" indiquent les valeurs de départ et d'arrivée (ex: 1 à 0)
 - "dur" indique la durée de l'effet (ex: 3s)
 - "begin" pour retarder / synchroniser les animations (ex: 2s)
 - "repeatCount" permet de répéter l'animation un nombre de fois précis (ex: 3) avec les paramètres de l'exemple on aurait un effet de fondu qui se lancerait 3x après 2s
- Pour animer une forme on utilise **<animateMotion>** avec les attributs :
 - "path" permet de préciser le chemin (vecteur, trajet) à suivre en un temps donné par "dur"
- Avec **<animateColor>** on peut aussi passer d'une couleur (from) à une autre (to)
- Et on fait des transformations à l'aide de l'attribut **transform="translate(x,y)"** ou de **<animateTransform>** (ex: **type="rotate | scale | translate | skewX/Y..."**)

161



Exemple d'effet visuel de fondu :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%" version="1.1" xmlns="http://www.w3.org/2000/svg">
<rect x="20" y="20" width="250" height="250" style="fill:blue">
<animate attributeType="CSS" attributeName="opacity" from="1" to="0" dur="3s"
repeatCount="3" />
</rect>
</svg>
```

Exemple de code d'animation :

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%" version="1.1" xmlns="http://www.w3.org/2000/svg">
<g transform="translate(80,150)">
<text id="TextElement" x="0" y="0" style="font-family:Verdana;font-size:18"> C'est du SVG !
<animateMotion path="M 0 0 L 80 150" dur="5s" fill="freeze"/>
</text>
</g>
</svg>
```

Multiformats

- Génération multi format : **XHTML/Fo + SVG en XSLT**
 - On remarque qu'avec les possibilités d'**XSLT** et de **SVG** on peut extraire des données XML et créer **dynamiquement** des **graphiques** **SVG**
 - Donc **générer des images à partir de données**
 - C'est très utile en particulier pour les **outils statistiques** tels que les outils d'aide à la décision, les agrégations peuvent être représentées graphiquement à la volée
 - De plus, le tout peut être **embarqué** dans une page **xHTML** générée ou **XSL-Fo**, elle aussi **dynamiquement** en **XSL-T**, ceci nécessite quelques précautions, dont :
 - l'utilisation de namespaces pour éviter la confusion des langages utilisés
 - sous MFF, l'usage de certaines extensions ("xhtml" au lieu de ".html" ou ".htm")
 - sous MSIE, le recours à une balise object pour forcer l'utilisation de la visualisatrice SVG même pour du SVG créé à la volée en texte mêlé au XHTML

162



Exemple de génération multi-formats :

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml" version="1.0" encoding="iso-8859-1" doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN" doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"/>
<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:svg="http://www.w3.org/2000/svg" xml:lang="fr">
<head>
<title>GENERATION DE SVG ET XHTML EN XSLT</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<object id="AdobeSVG" classid="clsid:78156a80-c6a1-4bbf-8e6a-3cd390eeb4e2"></object>
<xsl:processing-instruction name="import">
namespace="svg" implementation="#AdobeSVG"
</xsl:processing-instruction>
</head>
<body>
<h1>Chiffre d'affaires par livre</h1>
<svg:svg width="500" height="500" viewBox="0 0 500 500" version="1.1">
<svg:title> SVG + XSL </svg:title>
<svg:desc> Ceci est un exemple de SVG de base </svg:desc>
<xsl:for-each select="bibliotheque/livre">
<svg:rect x="100px" y="{20*position()}px" width="{@ca}" height="20" style="fill:red; stroke:black; stroke-width:1">
<svg:animate attributeName="width" attributeType="XML" begin="0s" dur="5s" fill="freeze" from="0" to="{@ca}"/>
</svg:rect>
<svg:text x="10px" y="{20*position() + 16}px" font-family="Arial" font-size="16px" font-weight="bold"><xsl:value-of select="titre/text()" /></svg:text>
</xsl:for-each>
</svg:svg>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Conclusion

□ Dans cette partie vous avez vu comment mettre en œuvre des SVG statique pour faire des dessins ou des graphiques dans vos PDF ou vos HTML:

- ✓ qu'est ce que SVG?
- ✓ Sa structure
- ✓ ses différentes formes natives
- ✓ différentes balises
 - Groupes, Texte, liens, symboles, images
- ✓ Des effets graphiques
- ✓ Des animations
- ✓ Les Multi formats et CSS



- HTML+ SVG
- SVG + XSL (web)
- SVG + XSL-FO

INTÉGRATION SVG

Intégration SVG html

- Intégration dans une page HTML
 - On utilise pour intégrer un fichier SVG ***embed***

```
<embed type="image/svg+xml" src="file.svg">
```

- Ou ***object***

```
<object type="image/svg+xml" data="file.svg">
```

- Intégration dans des pages FO
 - On utilise la balise ***<fo:instream-foreign-object>***

* cf. chapitre FO

165

Exemple d'intégration avec <embed> :

```
<embed type="image/svg+xml" src="file.svg" width="50%" height="50%"  
pluginspage="http://www.adobe.com/svg/viewer/install/" align="right" />
```

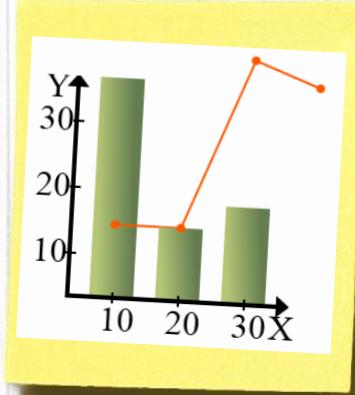
Exemple d'intégration avec <object> :

```
<object type="image/svg+xml" id="myfirstsvginhtml" data="file.svg" width="50%" height="50%"  
align="right" codebase="http://www.adobe.com/svg/viewer/install/"/>
```

Si vous ne disposez pas d'une visionneuse, cliquez [ICI](http://www.adobe.com/svg/viewer/install/)

Intégration SVG + XSL génération de graphiques

- Voici un exemple d'intégration svg
 - Création de reporting graphique
 - Conception graphique modulaire,...
 - Valeur provenant de XML
(et / ou json en version xsl3.0)
 - Exportation FO (pdf) / web
 - Mise a jour a chaque transformation
 - Anim-able (pour les fichiers compatibles)



166



Intégration SVG+XSL

Le langage SVG + XSL (web)

- Pour la génération html aucune balise n'est nécessaire pour l'import de SVG si le navigateur possède un interpréteur SVG intégré (MFF,Chrome,IE9)

Code html :

```
<div>
  <svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%" viewBox="0 0 580 310">[...]</svg>
</div>
```

- Pour les navigateurs nécessitant un plugin externe (ex: IE5/6/7/8) une balise est nécessaire pour que le navigateur ait connaissance qu'un interpréteur est nécessaire

– **<object ...><svg ...>[...]</svg></object>**

167



Usage XSL->html

```
<xsl:template match="info" >
  <div>
    <svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%" viewBox="0 0 580 310">
      <desc>Insertion de dessin</desc>
      <g>
        <rect x="0" y="10" width="10" height="10" fill="BLUE"/>
      </g>
    </svg>
  </div>
</xsl:template>
```

Intégration SVG+XSL

Le langage SVG + XSL-FO

- Pour la génération de documents FO une balise est nécessaire à la déclaration d'un bloc SVG :

```
<fo:instream-foreign-object ...>
  <svg .../>
</fo:instream-foreign-object>
```

- En plus des balises de mise en forme habituelles, cette balise propose les attributs suivants pour la mise en forme dans le bloc "instream" :

```
<fo:instream-foreign-object
  scaling="uniform | non-uniform | inherit"
  content-height="scale-to-fit | scale-down-to-fit | scale-up-to-fit"
  content-width="scale-to-fit | scale-down-to-fit | scale-up-to-fit" />
```

- (mêmes valeurs que height)

168

Usage XSL->pdf

```
<xsl:template match=<< info >> >
  <fo:block>
    <fo:instream-foreign-object margin-left="10%" scaling="uniform" content-height="scale-to-fit"
      content-width="scale-to-fit" width="80%">
      <svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%" viewBox="0 0 580
      310">
        <desc>Insertion de dessin</desc>
        <g>
          <rect x="0" y="10" width="10" height="10" fill="BLUE"/>
        </g>
      </svg>
    </fo:instream-foreign-object>
  </fo:block>
</xsl:template>
```

Conclusion

❑ Dans cette partie vous avez vu comment mettre en œuvre des svg statique ou dynamisé avec xsl :

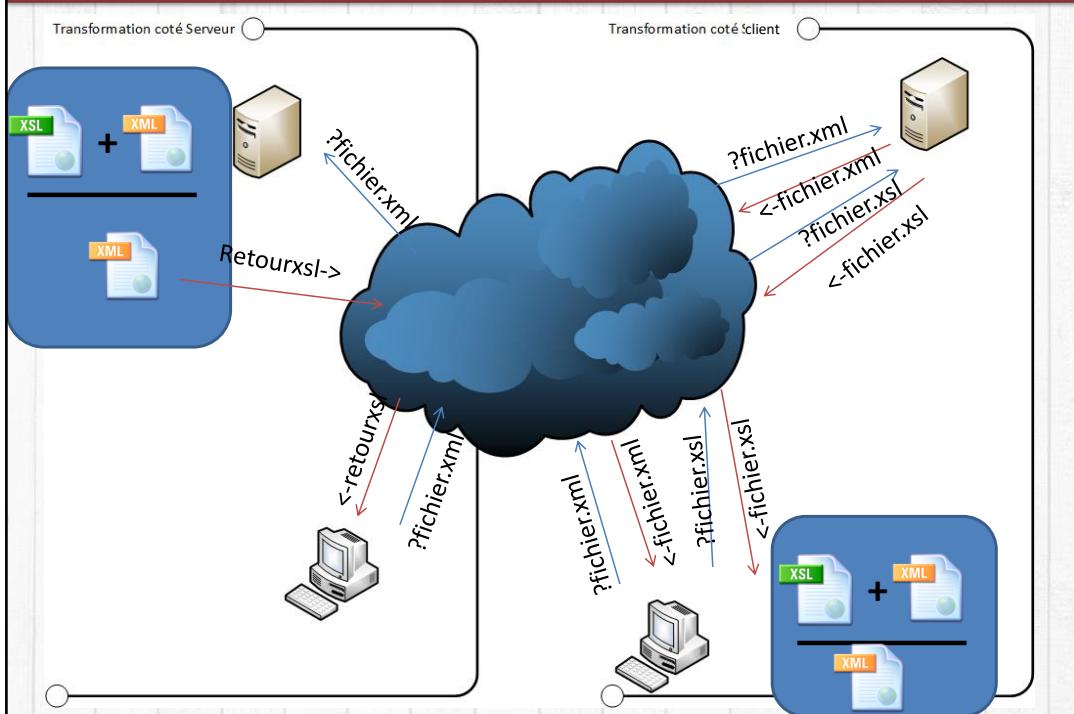
- ✓ HTML+ SVG pour l'intégration statique pour le web
- ✓ SVG + XSL pour la dynamisation du svg pour le web
- ✓ SVG+FO + XSL pour la dynamisation de svg pour FO



- Côté client
- Côté Serveur
- Coté excel
- Coté Application

EXPLOITATION XSL

Serveur ou Client



Exploitation XML

Côté client

- Visualisation dynamique côté client
 - Enfin, on peut lier directement 2 fichiers (XML + XSL) dynamiquement via le DOM (normalement document.implementation) mais MSIE utilise son ActiveX
 - Pour créer une interface DOM comprenant XML on utilise donc soit :

```
xml = new ActiveXObject( 'Microsoft.XMLDOM');
// sous MSIE, soit :
xml = document.implementation.createDocument("",",null);
//natif et on charge le fichier XML (puis XSL) avec la méthode « xml.load(file_name) »
```

- On passe ensuite le xml à la moulinette xsl via un processeur XSLT
xml.transformNode(xsl)

- et on affecte le DOM avec

```
//document.body.innerHTML ou
xsltProcessor = new XSLTProcessor(); xsltProcessor.importStylesheet(xsl);
xsltProcessor.transformToFragment(xml, document); //et...
body.appendChild
```

172

Exemple de traitement XSL-T avec liaison XML dynamique en JS :

```
function loadXMLDoc(fname) {
    var xmlDoc;
    if( window.ActiveXObject) { // pour MSIE
        xmlDoc = new ActiveXObject( 'Microsoft.XMLDOM');
    }
    else if( document.implementation && document.implementation.createDocument) { // code pour MFF
        xmlDoc = document.implementation.createDocument("",",null);
    } else { /* ne rien faire */ }
    xmlDoc.async = false;
    xmlDoc.load( fname);
    return( xmlDoc);
}

function displayResult() {
    xml = loadXMLDoc( 'data.xml');
    xsl = loadXMLDoc( 'transform.xsl');

    if( window.ActiveXObject) { // MSIE
        ex = xml.transformNode( xsl);
        document.body.innerHTML = ex;
    } else if( document.implementation && document.implementation.createDocument) { // pour MFF
        xsltProcessor = new XSLTProcessor();
        xsltProcessor.importStylesheet( xsl);
        resultDocument = xsltProcessor.transformToFragment( xml, document);
        document.body.appendChild( resultDocument);
    }
}
```

Exploitation XML Côté client

- Xsltjs (avec ou sans jQuery)
 - Facile à intégrer, java
 - Transformation dans le navigateur
 - Prise en charge :
 - Mozilla 0.9.4+, Ms IE 5+(6+ si jQuery plugin), Opera 9+, Safari 3+

Code html / jQuery :

```
<div id="transformResult"></div>
<script type="text/javascript">
$(function() {
    $('#transformResult').xslt("document.xml", "document.xsl");
});
</script>
```

Xsltjs: <http://johannburkard.de/software/xsltjs/>

Exploitation XML

Côté serveur

- Visualisation dynamique côté serveur
 - on peut lier directement 2 fichiers (XML + XSL) dynamiquement via le DOM (normalement *document.implementation*) mais MSIE utilise son ActiveX
 - Pour créer une interface DOM comprenant XML on utilise donc : **Server.CreateObject("Microsoft.XMLDOM")** puis on lui charge le fichier xml avec :
load(Server.MapPath(<< fichier.xml >>)). L'opération est la même pour les fichiers xml et xsl
 - On passe ensuite le xml à la moulinette xsl via un processeur XSLT
xml.transformNode(xsl) et on écrit le DOM avec
Reponse.write(RetourTransformNode)

174

Exemple de traitement XSL-T avec liaison XML dynamique côté serveur avec ASP:<%

```
'Load XML
set xml = Server.CreateObject("Microsoft.XMLDOM")
xml.async = false
xml.load(Server.MapPath("cdcatalog.xml"))
```

```
'Load XSL
set xsl = Server.CreateObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load(Server.MapPath("cdcatalog.xsl"))
```

```
'Transform file
Response.Write(xml.transformNode(xsl))
%>
```

Un équivalent est bien sûr faisable en php

Exploitation XML

Côté Excel

- Visualisation dynamique à l'ouverture d'un flux xml

- on peut lier directement 2 fichiers (XML + XSL) dynamiquement via une **processing-instruction stylesheet**

```
<xmlstylesheet type="text/xsl"  
    href="URLfichierXSL.xsl"?>
```

- Le fichier généré par le xsl devra être sous la forme **"feuille de calcul excel 2003 xml"**
- La moulinette xsl sera automatiquement faîs à l'ouverture



175

Exemple de traitement XSL-T avec liaison XML dynamique côté serveur avec ASP:<%

```
'Load XML  
set xml = Server.CreateObject("Microsoft.XMLDOM")  
xml.async = false  
xml.load(Server.MapPath("cdcatalog.xml"))
```

```
'Load XSL  
set xsl = Server.CreateObject("Microsoft.XMLDOM")  
xsl.async = false  
xsl.load(Server.MapPath("cdcatalog.xsl"))
```

```
'Transform file  
Response.Write(xml.transformNode(xsl))  
%>
```

Un équivalent est bien sûr faisable en php

Exploitation XML

Côté Appli

- Dans tous les langage il existe un outil de transformation xsl open et ou payant

- php

- [exemple de code](#) →

- c

- include libxml2 & [libxslt](#)

- c#

- [using System.Xml.Xsl;](#)

- apache ant, java ,

- google go, vb.net, ...

```
<?php  
// CHargeement du source XML  
$xml = new DOMDocument;  
$xml->load('collection.xml');
```



```
$xsl = new DOMDocument;  
$xsl->load('collection.xsl');
```



```
// Configuration du transformateur  
$proc = new XSLTProcessor;  
$proc->importStyleSheet($xsl);  
// attachement des règles xsl  
echo $proc->transformToXML($xml);
```

```
?>
```

176

libxslt :<http://xmlsoft.org/XSLT/downloads.html> & API: <http://xmlsoft.org/XSLT/API.html>
php <http://php.net/manual/fr/book.xsl.php>
c# : <https://docs.microsoft.com/fr-fr/dotnet/api/system.xml.xsl.xsltransform.transform?view=netframework-4.7.2>

Apache ant

- Installer Ant

- Extraction du zip [disponible ici](#)
- Définition des variables d'environnement :
 - %ANT_HOME%
 - path =%ANT_HOME%\bin
- Jdk 1.8 pour la dernière version



<http://ant.apache.org/bindownload.cgi>

Ant simple xml+xsl

- Build.xml pour transformation simple
- ```
<project default="do-it">
 <target name="do-it">
 <xslt processor="trax" in="input.xml"
 style="transform.xsl" out="output.html"/>
 </target>
</project>
```



## Ant composition de build.xml

- <project [default="" ][basedir="" ]>
- C'est la balise root du fichier build.xml
  - Default : permet de définir la section target si aucune n'est définie lors de l'appel
- Elle contient les target
  - Les targets sont les container des différentes tâches
  - Acceptent les dépendances d'executions



## Ant target instruction xslt

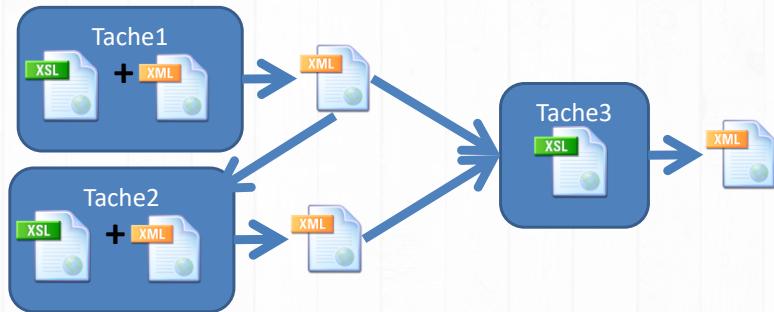
- C'est dans cette balise que les instructions et ou les tests se trouvent.
- ```
<target name="test" >
    <xslt in="in.xml" style="style.xsl"
          out="out.html" processor="trax" />
</target>
```
- Ici l' execution d'un tache xsl de transformation



<https://ant.apache.org/manual/Tasks/style.html>

Ant target dépendances

- Les target organisent l'ordre des instructions grâce au dépendances



- <target name="Tache2" depends="Tache1">
- <target name="Tache3" depends="Tache1,Tache2">

<https://ant.apache.org/manual/targets.html>

Ant echo

- Permet la sortie d' informations
- <echo [message="""] [append output]
[level]>[Message de sortie]</echo>
 - Message: remplace le contenu de la balise
 - file/output : fichier / ressource à écrire avec le message
 - append: *true* / *false* pour le fichier / ressource



<https://ant.apache.org/manual/Tasks/echo.html>

Ant get

- <get src="http://127.0.0.1/ant/facture3.xml" dest="fichier3.xml"/>
 - Permet l'accès à une url en déposant le contenu provenant de *src* vers *dest*
 - Authentification html possible
 - Compression Gzip possible



<https://ant.apache.org/manual/Tasks/get.html>

Ant condition

- Tache conditionnel
 - Property porte les retour de la condition *true / false*
- ```
<condition property="test.isTrue">
 <and>
 <available classname="javax.*"/>
 <available file="f.xml"/>
 <not>
 <os name="SunOS" arch="sparc"/>
 </not>
 </and>
</condition>
```



<https://ant.apache.org/manual/Tasks/condition.html>

## Ant ftp

- Grace a l'instruction <ftp il est possible d' effectué les actions courantes sur un ftp en sélectionnant une action
  - mkdir
  - rmdir
  - List
  - Del
  - Get
- ```
<ftp action="get" server="ftpperso.free.fr" userid="anonymous"
password="me@t.com">
    <fileset dir="htdocs/manual">
        <include name="**/*.html"/>
    </fileset>
</ftp>
```



Ant xmlValidate

- <xmlvalidate file="file.xml" />
 - Les attributs *true / false*
 - lenient = wellFormdness
 - failonerror = arret sur erreur
 - Les attributs pour schema
 - <attribute name="http://apache.org/xml/features/validation/schema" value="true"/>
 - <attribute name="http://xml.org/sax/features/namespaces" value="true"/>
 - <property name="http://apache.org/xml/properties/schema/external-noNamespaceSchemaLocation" value="\${xsd.file}"/>
 - L'attributs pour les doctype
 - <dtd publicId="-//DTD1.0//EN" location="dtdfile.dtd"/>

<https://ant.apache.org/manual/Tasks/xmlvalidate.html>

Ant task

- Voici la liste des task courantes :
 - telnet
 - Sql
 - echoXML
 - Zip
 - ...
- Liste exhaustive :
 - <https://ant.apache.org/manual/tasklist.html>



<https://ant.apache.org/manual/tasklist.html>

Conclusion

- ❑ Dans cette partie vous avez vu comment mettre en œuvre des transformations xsl
- ✓ Côté client
 - ✓ Côté Serveur
 - ✓ Coté excel
 - ✓ Coté Application



XSL 2.0, 3.0, LES ÉVOLUTIONS

XSL 2.0, 3.0, les évolutions XSL3.0 Nouveautés

- XSL Transformations (XSLT) Version 2.0
 - Recommandation du W3C du 23 janvier 2007
- XML Path Language (XPath) 3.0
 - recommandation du W3C
du 23 janvier 2007
 - 2.0 seconde édition du W3C du
14 Décembre 2010



Recommandation xsl : <https://www.w3.org/TR/xslt20/>
RC Xpath 2.0 :<https://www.w3.org/TR/2007/REC-xpath20-20070123/>
RC Xpath 2.0 seconde édition :<https://www.w3.org/TR/xpath-20/>
<https://www.w3.org/TR/xslt20/#changes>

XSL 2.0, 3.0, les évolutions

XSL2.0 Nouveautés

- Conversion auto fragments d'arbres / node-sets
- Multiple <output />
- Grouping natif XSL
 - <xsl:for-each-group/>
- Définition de fonctions personnel
 - <xsl:function/>
- Xpath 2.0
 - Typage fort, lot de fonctions, gestion du temps, ...

191



<https://www.w3.org/TR/xslt20/#changes>

<http://www.xml.com/pub/a/2002/04/10/xslt2.html>

XSL 2.0, 3.0, les évolutions

XSL2.0 Nouveautés

- Definition de functions personnel
 - <xsl:function/>
- *Création de séquences*
 - <xsl:sequence select="" />
- De nouveaux attributs pour
 - xsl:template
 - xsl:variable
 - as="xs:type"
- **Encore plus**
 - **value-of** permet désormais de sélectionner plusieurs données (couplées par ,), et séparation grâce à l'attribut separator

192



<http://www.xml.com/pub/a/2002/04/10/xslt2.html>

<http://jean-luc.massat.perso.luminy.univ-amu.fr/ens/xml/09-xsl2.html>

XSL 2.0, 3.0, les évolutions

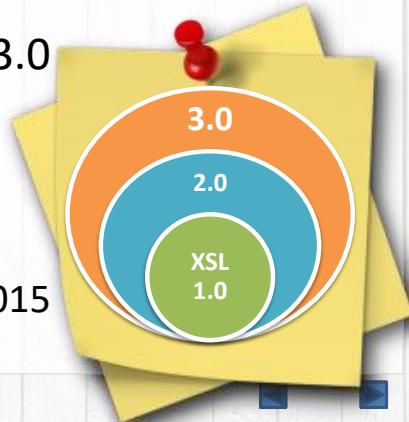
XSL2.0 Nouveautés

- descente de priorités
 - **xsl:next-match**
 - » Renvoie au template de priorité inférieur
- analyze de regex
 - **xsl:analyze-string**
 - **select=**" *XpathExpression* "
 - **regex**="[A-VX-Z1-2\-\]{1,}"
 - » déclenche la balise enfant
 - <xsl:matching-substring> si matching "ok"
 - <xsl:non-matching-substring>



XSL 2.0, 3.0, les évolutions XSL3.0 Nouveautés

- XSL Transformations (XSLT) Version 3.0
 - Candidat du W3C du 19 novembre 2015
- XML Path Language (XPath) 3.0
 - recommandation du W3C
du 8 avril 2014
 - 3.1 candidat du W3C du
17 Decembre 2015



Recomendation Xpath 3.0:<https://www.w3.org/TR/xpath-30/>
RC Xpath 3.1 :<https://www.w3.org/TR/xpath-31/>
<https://www.w3.org/TR/xslt30/#changes>

XSL 2.0, 3.0, les évolutions XSL3.0 Processeurs & éditeurs

- Editeurs
 - Altova XML Spy 2016 (300\$ → ∞\$ en pack)
 - Oxygen (300\$ → 2,500\$)
 - Editix 2016 (80\$ → 150\$)
 - Stylus studio (80\$ → 700\$)
 - XML BluePrint (40\$ → 115\$)
- Processeurs
 - **Saxon PE** → java based → propriétaire
 - **Exslt (*beta**)** → .NET based (F#) → free

195



Stylus studio:

<http://www.stylusstudio.com/buy/>

Oxygen:

<https://www.oxygenxml.com/buy.html>

Xml BluePrint:

<https://www.xmlblueprint.com/buy.htm>

XSL 2.0, 3.0, les évolutions

XSL3.0 Nouveautés

- supporte le XML et **JSON**
 - json-to-xml()
 - xml-to-json()
- Pré-évaluation d'expression
 - *xsl:evaluate*
- maps ou tableaux associatifs
 - *select="maps:new(...)"*
 - *map.get(...)*
- Gestion d'erreur
 - *<xsl:try> ... <xsl:try/>*
 - *<xsl:catch> ... <xsl:catch/>*

196



<http://fr.slideshare.net/jakubmaly/xslt-30-new-features>

<https://www.xml.com/articles/2017/02/14/why-you-should-be-using-xslt-30/>

XSL 2.0, 3.0, les évolutions

XSL3.0 xsl:stream

- Stream xml

- <xsl:stream />
 - *Defintion primaire*
 - *Definition de contexte en flux*

Code xsl 3.0 :

```
<xsl:stream
  href = { uri }
  use-accumulators? = tokens
  validation? = "strict" | "lax" | "preserve" | "strip"
  type? = eqname>
  <!-- Content: sequence-constructor -->
</xsl:stream>
```

- <xsl:fork/>
 - *Duplication processus*

Code xsl 3.0 :

```
<xsl:fork>
  <!-- Content: (xsl:fallback)*, ((xsl:sequence xsl:fallback*)*) | (xsl:for-each-group xsl:fallback*) -->
</xsl:fork>
```



XSL 2.0, 3.0, les évolutions

XSL3.0 xsl:iterate

- Itération

- **<xsl:iterate />**

- *Defintion primaire*
 - *Definition de contexte de noeud*
 - *Definition de le sequence d'execution*

Code xsl 3.0 :

```
<xsl:iterate>
  select = expression >
  <!-- Content: (xsl:param*, xsl:on-completion)?
    , sequence-constructor) -->
</xsl:iterate>
```

- **<xsl:next-iteration/>**

- *Gestion des itérations*

Code xsl 3.0 :

```
<xsl:next-iteration>
  <!-- Content: (xsl:with-param*) -->
</xsl:next-iteration>
```



XSL 2.0, 3.0, les évolutions

XSL3.0 xsl:iterate

- Itération

- **<xsl:break>**
 - Condition d'arrêt de *xsl:iterate*

Code xsl 3.0 :

```
<xsl:break
  select? = expression >
  <!-- Content: sequence-constructor -->
</xsl:break>
```

- **<xsl:on-completion>**
 - Gestion de sortie de boucle

Code xsl 3.0 :

```
<xsl:on-completion
  select? = expression >
  <!-- Content: sequence-constructor -->
</xsl:on-completion>
```



XSL 2.0, 3.0, les évolutions XSL3.0 xsl:accumulator

- Accumulateur

- <xsl:accumulator/>

- *Définition primaire*

Code xsl 3.0 :

```
<xsl:accumulator
  name = eqname
  initial-value = expression
  as? = sequence-type
  streamable? = boolean >
  <!-- Content: xsl:accumulator-rule+ -->
</xsl:accumulator>
```

- <xsl:accumulator-rule />

```
<xsl:accumulator-rule
  match = pattern
  phase? = "start" / "end"
  select? = expression >
  <!-- Content: sequence-constructor -->
</xsl:accumulator-rule>
```

200



Conclusion

- Dans cette partie vous avez pris connaissance des nouvelles possibilités XSL 3.0
- Vous avez appris les différences fondamentales entre le xsl 2.0 & xsl 3.0
- Vous avez acquis quelques techniques de mise en œuvre de transformations XML/XSL nouvelle génération



Conclusion finale

- Remerciements et encouragements

Pendant cette formation, nous espérons :

- que vous avez appris de nombreuses choses théoriques et pratiques
- que vous y avez trouvé plus que ce que vous étiez venu chercher
- et que vous avez passé un bon moment avec votre professeur

Merci d'avoir fait confiance à ORSYS, à bientôt pour d'autres formations !



Bibliographie

- Livres de référence
 - « XSLT and XPath 2.0 » de Michael Kay chez Wrox
 - « XML cours et exercices » de Alexandre Brillant
- Sites de référence
 - [http:// www.w3.org](http://www.w3.org)
(site du W3C - standards)
 - [http:// www.w3schools.com](http://www.w3schools.com)
(Web programming)
 - <http://www.oreillynet.com/>
(forum de développeurs)
 - <http://xmlfr.org>
(blog XML)
 - <http://www.zvon.org>
(guide de la galaxie XML)





ORSYS
formation

A BIENTÔT...



Xsl

Boucle récursive de parcours de branche:

```
<xsl:template name="boucleRecurcive">
<xsl:param name="Node"/>
...
<xsl:if test="$Node/following-sibling::*">
<xsl:call-template name="boucleRecurcive">
<xsl:with-param name="Node">
  select="$Node/following-sibling::*"/>
</xsl:call-template>
</xsl:if>
```

Xsl:number (value-of return like):

```
<xsl:number value="single où multiple"
 count="StartNodeName [UnionNodeName]" from="A.1" />
```

Fonctions XSL 2.0 déclaration:

```
<xsl:function name="Namespace:functionName">
<xsl:param name="param1"/>
<xsl:param name="param2">
  select="XpathExpression" />
<!-- Contenu de la fonction-->
<xsl:function>
  <!---retour de fonction par xsl:value-of-->
<xsl:value-of select="Namespace:functionName
  (param1,param2)" />
```

Fonctions 2.0 usage:



Html & CSS

dev.piwya.net

Intégration fichier css (dans <head>):

<link href="fichier.css" rel="stylesheet" type="text/css">

Références balises Html:

| | |
|--------------------------------|--|
| Attribut class | <balise class="class / id12...I"/> |
| Attribut id | <balise id="idName"/> |
| Elément bloc | <div> ... </div> |
| Elément enligne | ... |
| Tableau, ligne, cellule | <table><tr> <td>cellule</td> </tr></table> |
| Lien | visiter |
| image | |
| Liste de puces et puces | Une puce |
| Formulaire, entrée, soumission | <form method="POST GET" action="lienTraitemen"> <input type="text password checkbox hidden" name="Nom"> <input type="submit" value="Valider"/> </form> |

Le mémento XML, XPath, XSL, CSS est le rassemblement de tout ce qu'il est nécessaire de savoir sur ces technologies, dans un format de poche portable partout, et fait pour faciliter la vie du développeur XML et dérivés.

Les mémentos Programs Is What You Are, ou piwya, sont une suite d'outils d'aide au développement informatique.



MEMENTO

piwya

XML, XPath, Xsl, CSS

Structure minimum Html:

<html>

<head>

<title>Titre de votre page</title>

</head>

<body>

<!-- Contenu de page -->

</body>

</html>

Fonctions simple CSS:

#NomClass → Classe réutilisable dans la page
.NomClass → Identifiant à usage unique par page
NomDeBalise → Balise portant NomDeBalise

Sélecteur encapsulé:

→ SélecteurComeneur SélecteurContenu
SélecteurContenu → SélecteurComeneur SélecteurContenu

Sélecteur multiple:

→ Sélecteur1, Sélecteur2, ...

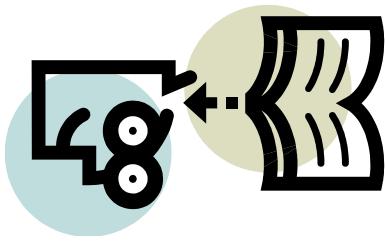
Liste de propriétés : margin-[left|right|top|bottom], padding-[left|right|top|bottom], left, right, top, bottom, text-align, color, background-[color|image|attachment|position|], opacity, [min|max]-width, [min|max]-height, display, visibility, cursor, z-index, float, overflow, border-[width|color|style], ...



piwya.net



w3schools.com



Avant toute chose, ce guide est un outil pédagogique pour progresser en Xsl dès les premiers pas, mais il convient tout a fait aux développeurs plus aguerris.



Section : Web XML développement

A.Desorbaix

| XML, XPath | Xpath, fonctions préfixe : fn:* | Xsl | Xsl |
|--|----------------------------------|-----|-----|
| Règles XML: | | | |
| - 1 seule balise Root | | | |
| - Pas de balise ouverte | | | |
| - Pas de balises croisées | | | |
| - nom d'attribut, nom de balises, valeurs d'id commencent par une lettre | | | |
| XPath | | | |
| Sélecteurs: | | | |
| nodeName → sélectionne tous les nœuds fils du nœud | | | |
| // → tous les nœuds depuis le contexte. | | | |
| .. → nœud racine | | | |
| */ → tous les nœuds depuis le contexte. | | | |
| @* → nœud parent du nœud courant. | | | |
| text() → attribut | | | |
| @Xml → n'importe quel nœud élément | | | |
| @XmlAttribute → n'importe quel attribut | | | |
| node() → n'importe quel nœud | | | |
| expr[exprBool] → expr : [n] à nième élément [exprBool] vérifiée | | | |
| Fonctions Math: | | | |
| floor(num) → Entier inf. ceiling(num) → Entier sup. | | | |
| round(num) → Arrondi entier plus proche | | | |
| Number(arg) → cast abs(num) → valeur absolue | | | |
| concat(arg1,arg2) → concaténation de arg1 et arg2 | | | |
| compare(str1,str2) → Compare 2 chaînes str1 et str2 | | | |
| substring(str ,L) → Coupe str de str sur L caractères | | | |
| string-length(str) → renvoie la taille de str | | | |
| contains(str,str2),match(str,str2) → Vrai si str2 dans str | | | |
| end-with(str,str2),start-with(str,str2) → Vrai si str débute ou finie par str2 | | | |
| Fonctions agrégat 1.0 & 2.0: | | | |
| last() → renvoient le dernier position() | → renvoient la position courante | | |
| (pipe) → UNION | →renvoient le dernier | | |
| avg(Node) → Moyenne des éléments Node | →renvoient la position courante | | |
| min(Node) → min | | | |
| max(Node) → faux | | | |
| count(Node) → Nombre d'éléments Node | | | |
| sum(Node) → Somme des éléments Node | | | |
| Boucles: | | | |
| xsl:for-each select="XpathExpression"> | | | |
| </xsl:for-each> | | | |
| <xsl:choose> | | | |
| xsl:when test="Bool Xpath exp"> ... </xsl:when> | | | |
| xsl:otherwise> ... </xsl:otherwise> | | | |
| </xsl:choose> | | | |
| Fonctions de nœuds: | | | |
| position() → Renvoie la position du contexte | | | |
| last() → position du dernière éléments du contexte | | | |
| Opérateurs: | | | |
| arithmétiques → + - div * mod | | | |
| logique → = != < > <= >= or and | | | |
| Axes usage: | | | |
| cheminXpath/axeName::nodeName/fils | | | |