XSLT et XSLT-FO
Eléments de base

Samir Bennani

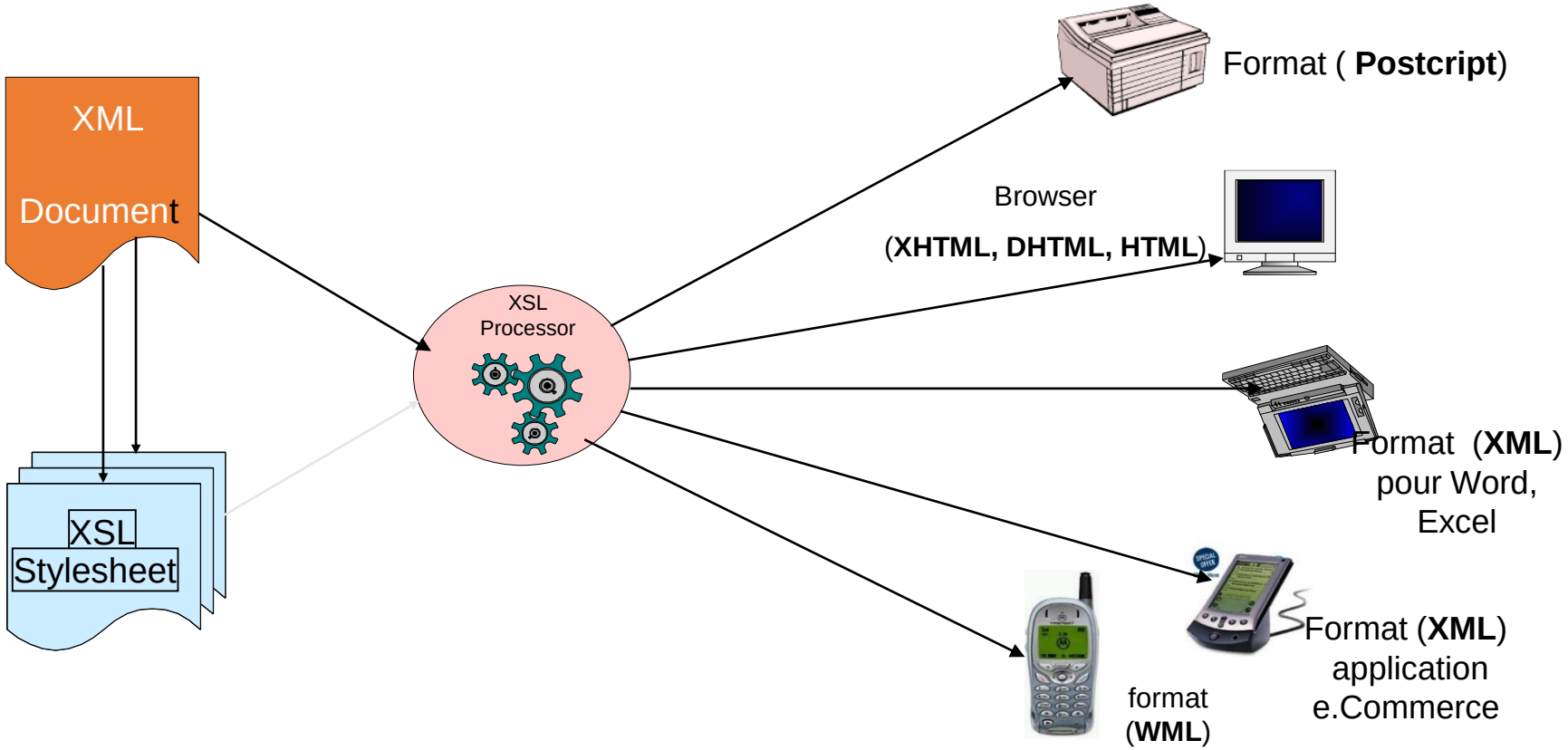
Asmae EL KASSIRI

(eXtensible Stylesheet Language Transformation)

- Pour transformer un document
  - XML vers XML,
  - XML vers une présentation (HTML, texte, rtf, pdf, etc.)
- Un document xml est un arbre, XSL:
  - parcourt l'arbre
  - applique les règles de transformations vérifiées (à condition vraie) aux nœuds sélectionnés
  - produit un document en sortie

Publications avec XSL

- Plusieurs formats de publication pour un contenu
-



- XSL permet la présentation sur des terminaux variés

Les feuilles de style

- Une feuille de style XSL
  - est un document XML de racine <xsl:stylesheet>
  - contient une liste de règles de transformation <xsl:template>
- Chaque règle (<xsl:template>) précise:
  - Une condition spécifiant le sous-arbre du document d'entrée auquel elle s'applique (match=)
  - Une production spécifiant le résultat de l'application de la règle (contenu)
- Il s'agit de règles de production classiques
  - If <condtition> then <production>
  - Codées en XML avec espace de nom xsl:

Côté document XML

- Référencer un document XSLT
  - Le référencement d'un document XSLT se fait au niveau du document XML dont les informations seront utilisées au cours de la transformation.
  - **<?xml-stylesheet type="text/xsl" href= "mon_document.xsl" ?>**
  - **L'attribut type:** Il permet de définir le *type* du document que nous souhaitons référencer. Dans notre cas, puisqu'il s'agit d'un document XSLT, il convient de renseigner la clef "**text/xsl**".
  - **L'attribut href:** Cet attribut, très connu de ceux qui manipulent régulièrement le HTML et ses variantes, permet d'indiquer l'*URI* du document que l'on souhaite référencer.

Côté Feuille de style XSL

- Le prologue
  - La première ligne d'un document XSLT est : <?xml version="1.0" encoding="UTF-8" ?>
- Le corps
  - Il est constitué d'un ensemble de **balises** dont l'**élément racine**. Comme c'était déjà le cas pour un schéma XML, l'élément racine d'un document XSLT est imposé.

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


</xsl:stylesheet>
```

Côté Feuille de style XSL

- La balise output: permet de décrire le document produit à l'issue des différentes transformations. Cet élément prend plusieurs attributs:
  - L'attribut method: permet de préciser le *type* du document produit à l'issue des transformations. 3 valeurs existent :
    - xml si le document à produire est un document XML.
    - html si le document à produire est un document HTML.
    - text si le document à produire est un document texte.
  - L'attribut encoding: permet de préciser l'*encodage* du document produit à l'issue des transformations. Un grand nombre de valeurs existent :
    - UTF-8.
    - ISO-8859-1.
    - etc.
  - L'attribut indent: permet d'indiquer si l'on souhaite que le document produit à l'issue des transformations soit *indenté* ou non. 2 valeurs sont possibles : yes et no
  - Les attributs doctype-public et doctype-system: pour associer un document DTD au document produit par la transformation XSLT.

<xsl:output method="html" encoding="UTF-8« doctype-public="-//W3C//DTD HTML 4.01//EN" doctype-system=http://www.w3.org/TR/html4/strict.dtd indent="yes" />

Les Templates

- Le corps d'un document XSLT est composé d'un ensemble de templates.
- Structure d'un template: Un template est défini par la balise <xsl:template /> à laquelle plusieurs attributs peuvent être associés:
  - L'attribut match permet de renseigner une expression XPath. Cette expression XPath permet alors de sélectionner les informations du document XML auxquelles le template s'applique.
  - L'attribut name est le nom donné au template, permettant de l'identifier de manière unique. Cet attribut est important puisque nous verrons par la suite qu'il ne soit pas rare qu'un template en appelle un autre.
  - Les attributs priority ou mode qui permettent respectivement de renseigner une priorité ou un mode de traitement.

Les Templates

- Contenu d'un template: permet de définir les transformations à appliquer à l'ensemble des données sélectionnées par l'expression XPath qui lui est attachée:
  - Attributs
    - *match*: condition de sélection des nœuds sur lesquels la règle s'applique.
    - *name*: nom de la règle, pour invocation explicite (en conjonction avec <apply-template>)
    - *mode*: permet d'appliquer à un même élément des règles différentes en fonction du contexte
    - *priority*: priorité, utilisé en cas de conflit entre deux règles ayant la même condition
  - Exemples
    - <xsl: template match="/">
    - <xsl: template match="auteur" name= " Hugo" priority="1">
    - <xsl: template match="auteur" name= " Hugo" priority="2">
    - <xsl: template match="auteur" name= "Hugo " mode="affichage1">
    - <xsl: template match="auteur" name= "Hugo " mode="affichage2">
    - <xsl:apply-template name="Hugo" mode="affichage1">

Les Fonctions

- **La fonction value-of:**
  - Elle permet d'extraire la *valeur d'un élément XML* ou la *valeur de ses attributs*
  - Elle possède un attribut **select** auquel il convient de renseigner une expression XPath permettant alors de sélectionner les informations à extraire
  - Syntaxe: <xsl:value-of select="expression XPath" />
  - Exemple:

```xml
<personne sexe="feminin">
  <nom>POPPINS</nom>
  <prenom>Marie</prenom>
  <adresse>
    <numero>28</numero>
    <voie type="avenue">avenue de la république</voie>
    <codePostal>13005</codePostal>
    <ville>MARSEILLE</ville>
    <pays>FRANCE</pays>
  </adresse>
  <telephones>
    <telephone type="professionnel">04 05 06 07 08</telephone
  </telephones>
  <emails>
    <email type="professionnel">contact@poppins.fr</email>
  </emails>
</personne>
```

```xml
<xsl:template match="/">
  <html>
    <head>
      <title>Test de la fonction value-of</title>
    </head>
    <body>
      <p>Type du numéro : <xsl:value-of
select="repertoire/personne[nom='POPPINS']/telephones/telephone/@type" /></p>
      <p>Numéro : <xsl:value-of select="repertoire/personne[nom='POPPINS']/telephones/telephone" /></p>
    </body>
  </html>
</xsl:template>
```

Les Fonctions

- **La fonction for-each:**
  - Elle permet de boucler sur un ensemble d'éléments
  - Par exemple, si l'on souhaite appliquer une transformation à l'ensemble des numéros de téléphone d'une personne, nous allons tous les sélectionner à l'aide d'une expression XPath, puis, grâce à la fonction <xsl:for-each />
  - Syntaxe: <xsl:for-each select="expression XPath" />

```
<xsl:template match="/">
  <html>
    <head>
      <title>Test de la fonction for-each</title>
    </head>
    <body>
      <xsl:for-each select="repertoire/personne[nom='DOE']/emails/email">
        <p>Type de l'adresse e-mail : <xsl:value-of select="@type" /></p>
        <p>adresse e-mail : <xsl:value-of select="." /></p>
      </xsl:for-each>
    </body>
  </html>
</xsl:template>
```

Les Fonctions

- **La fonction sort:**
  - Elle permet de trier un ensemble d'éléments par ordre croissant ou décroissant.
  - Elle est généralement utilisée au sein de la fonction <xsl:for-each />
  - Elle possède au moins un attribut **select** auquel il convient de renseigner une expression XPath permettant alors de sélectionner les informations à trier.
  - accepte également d'autres attributs qui sont cependant optionnels :
    - **Order** qui accepte les valeurs **ascending** (croissant) et **descending** (décroissant).
    - **Case-order** qui accepte les valeurs **upper-first** (les majuscules d'abord) et **lower-first** (les minuscules d'abord).
    - **Data-type** qui accepte les valeurs **text** (texte) et **number** (nombre) permet de préciser si les données à trier sont des nombres ou du texte.
    - **Lang** qui accepte pour valeur le code d'une langue (fr pour la langue française, es pour la langue espagnole, it pour l'italien, etc.)
  - Syntaxe: <xsl:sort select="expression XPath“

order="ascending|descending“

case-order="upper-first|lower-first"

data-type="text|number" lang="fr|es|it|..." />

```
<xsl:template match="/">
  <html>
    <head>
      <title>Test de la fonction sort</title>
    </head>
    <body>
      <xsl:for-each select="repertoire/personne">
        <xsl:sort  select="nom" />
        <xsl:sort  select="prenom" />
        <p><xsl:value-of select="nom" /> <xsl:value-of select="prenom" /></p>
      </xsl:for-each>
    </body>
  </html>
</xsl:template>
```

Les Fonctions

- **La fonction if:**
  - Elle permet de conditionner une transformation. Par exemple, grâce à cette fonction, il sera possible de n'appliquer une transformation qu'aux personnes de sexe masculin.
  - Elle possède un attribut test auquel il convient de renseigner la condition. Cette condition peut être la comparaison d'une chaîne de caractères ou de nombres.
  - Opérateurs arithmétiques

Condition	Explication
a = b	vérifie que la valeur de l'élément **a** est égale à la valeur de l'élément **b**.
not(a = b)	vérifie que la valeur de l'élément **a** n'est pas égale à la valeur de l'élément **b**.
a < b	Le symbole **<** (lower than) traduit en réalité le symbole
a <= b	Le symbole **<=** (lower or equal than) traduit en réalité le symbole **<=**.
a > b	Le symbole **>** (greater than) traduit en réalité le symbole **>**.
a >= b	Le symbole **>=** (greater or equal than) traduit en réalité le symbole **>=**.

  - Opérateurs logiques : AND / OR
  - Exemple:

```
<body>
  <xsl:for-each select="repertoire/personne">
    <xsl:if test="@sexe = 'masculin'">
      <p><xsl:value-of select="nom" /> <xsl:value-of select="prenom" /></p>
    </xsl:if>
  </xsl:for-each>
</body>
```

Les Fonctions

- **La fonction choose:**
  - Elle permet de conditionner une transformation.
  - Elle ne s'utilise pas toute seule. En effet, elle permet plusieurs conditions. Ainsi, dans le cas où la première condition n'est pas remplie, la seconde va être testée, puis la troisième, etc. Dans le cas où aucune condition n'est remplie, un cas par défaut peut être prévu.
  - Sous éléments:
    - <xsl:when />: pour exprimer les conditions
    - <xsl:otherwise />: pour exprimer le cas par défaut
  - Syntaxe:

```
<xsl:choose>
 <xsl:when test="test de comparaison">
  <!-- suite de la transformation -->
 </xsl:when>
 <xsl:when test="test de comparaison">
  <!-- suite de la transformation -->
 </xsl:when>
 <xsl:otherwise>
  <!-- suite de la transformation -->
 </xsl:otherwise>
</xsl:choose>
```

```
<body>
  <xsl:for-each select="repertoire/personne">
    <xsl:choose>
      <xsl:when test="nom = 'DOE'">
        <p>Bonjour John !</p>
      </xsl:when>
      <xsl:when test="nom = 'POPPINS'">
        <p>Quel beau sac !</p>
      </xsl:when>
      <xsl:otherwise>
        <p>Qui êtes-vous ?</p>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:for-each>
</body>
```

Les Fonctions

- **La fonction apply-templates:**
  - Elle permet de continuer la transformation des éléments enfants d'un template.

```
<xsl:template match="/">
  <html>
    <head>
      <title>Test de la fonction apply-templates</title>
    </head>
    <body>
      <xsl:apply-templates select="repertoire/personne[nom='POPPINS']" />
    </body>
  </html>
</xsl:template>

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

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

- A l'exécution de la transformation XSLT, le premier template à être appelé est le template dont l'expression XPath capture la racine de notre document XML.
- Dans ce template, la ligne<xsl:apply-templates select="repertoire/personne[nom='POPPINS']" /> permet d'indiquer que l'on souhaite continuer la transformation uniquement avec l'élément <personne /> correspondant à **POPPINS** ainsi que ses fils à savoir les éléments <nom /> et <prenom />.
- L'élément <nom /> va donc être transformé grâce au second template écrit dans notre document XSLT puisque son expression XPath le capture.
- Finalement, en suivant la même logique, l'élément <prenom /> va, quant à lui, être transformé grâce au dernier template de notre document XSLT.

XSL-FO

- XSL-FO est un *dialecte* de XML permettant de décrire le rendu de documents.
- Un document XSF-FO contient le contenu même du document ainsi que toutes les indications de rendu.
- Il s'apparente donc à un mélange de HTML et CSS avec une syntaxe XML mais il est plus destiné à l'impression qu'au rendu sur écran.
- Le langage XSL-FO est très verbeux et donc peu adapté à l'écriture directe de documents.
- Il est plutôt conçu pour des documents produits par des feuilles de style XSLT.

XSL-FO : le formatage

- Permet des mises en pages sophistiquées
- Objets de mise en forme applicables aux résultats avec XSLT
- Distinction
  - Formatage des pages
  - Formatage des objets à l'intérieur des pages
    - Statiques
    - Dynamiques

CEV* (Customer Economic Value) Comparison

International Model: 4300 SBA LP 4X2 Competitor Model: Freightliner FL700

Application: Dry Van - 5 years/25,000miles/yr

Expected Differential Savings for International Trucks and Tractors versus Competition

VALUE CATEGORY	ISB 5.9L	ISC 8.3L	MBE 900 4.3L	MBE 900 6.4L
Resale value	$ 800 - 1,200	$ 800 - 1,200	$ 1,637 - 2,456	$ 800 - 1,200
Engine overhaul cost	Not Available	Not Available	Not Available	Not Available
Preventive maintenance costs	$ 167 - 250	$ 288 - 432	$ (87) - (130)	$ 27 - 41
Repairability	Not Available	Not Available	Not Available	Not Available
TOTAL	$ 967 - 1,450	$ 1,088 - 1,632	$ 1,550 - 2,326	$ 827 - 1,241

VALUE NOTES

RESALE VALUE Consistent used vehicle pricing

ENGINE OVERHAUL COST Repair cost savings for engine rebuilds

PREVEATIVE MAINTENANCE COSTS Lower preventive maintenance costs due to longer service intervals

REPAIRABILITY Lower repair costs resulting from replacing individual parts instead of the entire section.

ADDITIONAL VALUE POINTS

VISIBILITY (PRODUCTIVITY) It's easier to see the gauges on our raised instrument panel. Improved visibility features like this one help drivers keep their eyes on the road.

VISIBILITY (PRODUCTIVITY) 1400 sq in. cooling package combines a 523 sq. in. radiator side-by-side with a 475 sq. in. charge air cooler, in highly efficient, light-weight, low profile design that enhances forward visibility.

VISIBILITY (PRODUCTIVITY) With up to 2074 sq. in. of Front windshield glass area, and 560 sq. in. of viewable glass area in each door the driver has a commanding view of the road. The repositioned A-pillar increases overall road view. A swept back angle helps deflect debris, thus minimizing glass damage. Standard tinted glass helps reduce glare.

RESALE VALUE (LIFECYCLE COST) International's galvanized steel cabs are constructed with welded-in reinforcements, a deep ribbed back panel, and a single piece steel door frame. It's built to withstand the toughest conditions - keeping you out of the shop and on the job.

RESALE VALUE (LIFECYCLE COST) Standard rear cab air suspension minimizes cab vibration, extending cab, and cab mounted component life and delivering an exceptional ride, resulting in lower driver turnover and associated costs.

RESALE VALUE (LIFECYCLE COST) Our redesigned all-aluminum cab resists corrosion, and it weighs less. A lighter truck means you can increase your payload and your profits.

RESALE VALUE (LIFECYCLE COST) Polished aluminum wheels can increase resale value, improve image, and reduce chassis weight, too. Add aluminum hubs to reduce chassis weight up to 108 pounds on the front axle.

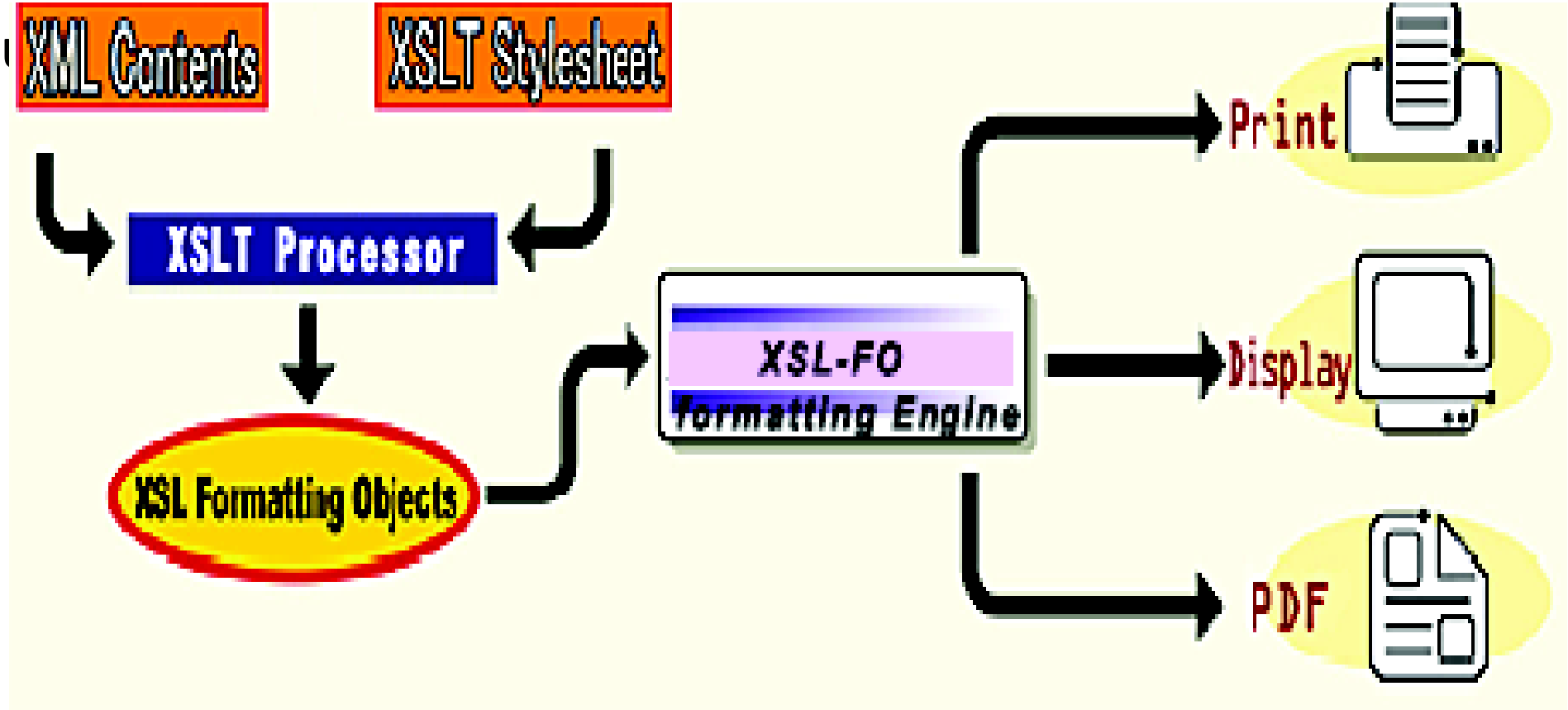
BODY AND CHASSIS INTEGRATION (OPERATING EFFICIENCY) Whether you're in the construction business, hauling garbage or plowing snow, the International 7000 Series has been designed to render the clean CA necessary to mount bodies and equipment with ease.

BODY AND CHASSIS INTEGRATION (OPERATING EFFICIENCY) Front frame extensions with integral reinforcements provide the strength required to mount plow hitches in combination with front PTO driven pumps and other heavy equipment.

CEV (Customer Economic Value) Comparisons are are intended for illustration purposes only. Individual results will vary based on truck owners specific application, loads cycle and miles per year.

Principes

- Pe



Organisation du document

- Un document FO est formé d'un élément fo:root qui comprend deux parties distinctes
  - une description des modèles de pages
    - fo:layout-master-set
  - une description du contenu
    - fo:page-sequence
- Le contenu comporte :
  - Des flux contenant les données mêmes du document
  - Des éléments statiques dont le contenu se répète sur les pages (en-têtes courants, no de page, etc.)

Objets de formatage

- Les objets de formatage sont multiples :
  - <fo:block>
    - utilisé pour les blocs de textes, paragraphes, titres…
  - <fo:display-rule>
    - ligne de séparation
  - <fo:external-graphic>
    - zone rectangulaire contenant un graphisme (SVG)
- Ils possèdent de nombreuses propriétés
  - Pour un block on peut définir
    - la marge gauche et droite
    - l'espace avant et après le paragraphe
    - la couleur du texte …….

Fonctionnalités

- Pages portrait ou paysage
- Pages recto-verso
- Page de tailles variées
- Marges multiples
- Colonnes multiples
- Entête et pieds de page
- Caractères unicode
- Formatage multi-langages
- Tables des matières générées
- Multiple directions d'écritures
- Numérotation des pages
- Graphiques et SVG
- Tables, avec entêtes,
  lignes et colonnes fusionnables
- Listes
- Zones flottantes
- Tris à l'édition
- Notes de bas de pages

XSL-FO: hello World

```
<?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">
        <fo:region-body margin="2 cm"/>
    </fo:simple-page-master>
 </fo:layout-master-set>
 <fo:page-sequence master-reference="my-page">
  <fo:flow flow-name="xsl-region-body">
    <fo:block>Hello, world!</fo:block>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
```

- Element Root
  - Permet de définir le namespace XSL-FO
- Layout master set
  - Permet de déclarer une ou plusieurs page masters (masque) et page sequence masters pour définir la structure des pages (ici une de 2 cm de marges)
- Page sequence
  - Les pages sont groupées en séquences et structurées selon la référence au masque.
- Flow
  - C'est le container du texte utilisateur dans le document. Le nom du flot lit le texte à une zone de la page définie dans le masque.
- Block
  - C'est le bloc de formatage qui inclut un paragraphe de texte pouvant être produit pas XSLT.

XSL-FO et XSLT : Exemple

- Définition de propriétés pour la racine
  - **<xsl:template match='/'>**

```
<fo-display-sequence
                font-style='italic'
                start-indent='4pt'
                end-indent='4pt'
                font-size='18pt'
        <xsl:apply-templates/>
        </fo-display-sequence>
        </xsl:template
```

- La définition d'une propriété locale est prioritaire devant l'héritage

Les processeurs XSL-FO

- Apache Group : FOP
  - Formating Object Processor
  - Génère du PDF http://www.apache.org
- JFOR (Open Source)
  - Génère du RTF http://www.jfor.org
- Antenna House
  - XSL Formatter http://www.antennahouse.com
- RenderX
  - Génère du PDF http://www.renderx.com
- Altova
  - StyleVision http://www.altova.com/products_xsl.html
- XML Mind FO Converter
  - Génère du RTF http://www.xmlmind.com/foconverter
- Autres
  - Arbortext, Adobe, Scriptura, XMLPDF, APOC, XSL-FO to TeX

XSL Conclusions

- XML = format pour la production de publications échangeables sur le web :
  - production d'une source unique en XML ;
  - XSLT = génération automatique de présentations multiples ;
  - XSL-FO = génération de présentations soignées avec pages maîtres et blocs formatés.