TUTORIAL XSL

Par Alain DESEINE

Ce document est inspiré des travaux de messieurs Paul Grosso et Norman Walsh

CEI Alain DESEINE
http://www.cabinfo.com
alain@cabinfo.com

Innovation Partners
http://www.innopart.com
innovationpartners@cabinfo.com

Sommaire du tutorial

Séparation du texte et de la présentation

- Langage de description de pages
- Feuilles de styles indépendantes des données
- Marquage des documents spécifique
- Marquage de présentation plutôt que logique
- Inclusion du style dans les marqueurs logiques

Avantage des feuilles de style indépendantes

- **Réutilisation possible des données**
- **Multiples possibilités de présentation du même jeu de données**
- **Permet l'utilisation de styles "utilisateurs"**
- **Permet une standardisation des styles**
- **Libère les auteurs de contenu des contraintes de style et de présentation**

XML ou HTML ?

- HTML possède un jeu fixe de marqueur (en dehors de ses évolutions)
- La méthode d'affichage des marqueurs HTML est définie basiquement dans la norme HTML
- Tout navigateur sait comment afficher un document HTML, même si celui-ci ne contient pas d'informations stylistiques
- XML, tout comme SGML, ne possède pas de jeux de marqueurs fixes, il founit au contraire un moyen simple de définir ses propres marqueurs.
- XML, comme SGML, ne décrit pas comment afficher le contenu d'un document
- Un mécanisme stylistique doit donc être associé à un document XML afin de

pouvoir l'afficher

Apporter du style aux données XML

Pour des questions d'interopérabilité, il est préférable d'utiliser une feuille de style conforme à un standard, apportant ainsi les avantages suivants :

- **Séparation du style et du contenu**
- **Permet des échanges de feuilles de styles entre utilisateurs, et entre logiciels (navigateur, éditeur, etc.)**
- **Permet à l'auteur de définir un style d'affichage que l'utilisateur peut utiliser ou non**
- **Permet de partager et de réutiliser très facilement les données**
- **Utilisation du même langage de feuille de style sur plusieurs applications**

Langage procédural ou déclaratif ?

- Les langages de feuille de style peuvent être déclaratifs ou procéduraux
- Ils décrivent des contraintes et des caractéristiques ou ils disent simplement comment afficher les données
- Marquage déclaratif, ou langage de script
- Gestion des erreurs, implémentations différentes
- Une feuille de style déclarative fournit des contraintes de formatage des données, alors qu'une feuille feuille de style procédurale implémente un processus de formatage à part entière

Conception de XSL

- **La conception de XSL est actuellement en cours**
- **Ce langage à été soumis au consortium W3C par ArborText, Inso, et Microsoft**
- **La soumission de ce projet à été accepté par le W3C le 10 Septembre 1997**
  http://www.w3.org/Submission/1997/13/ArborText
  http://www.w3.org/TR/NOTE-XSL-970910
- **Un groupe de travail à été désigné au sein du W3C pour la mise en place de la norme XSL**

XSL, perspectives d'avenir

XSL et mise en forme de texte

- Le style peut être inclu dans les données ou être extérieur
- Mise en place d'informations stylistiques sur des éléments isolés, ou au contraire sur des éléments génériques
- Sous classification des éléments en fonction de leur contexte (Sections, titre, paragraphes, etc.)
- Dépendance vis à vis de la structure des données
- Création de documents avec aucune, une ou plusieurs présentations possibles, au lieu de créer des documents avec une présentation statique et inamovible

XSL et XML

Les spécifications actuelles de XML comprennent les sub-divisions suivantes :

- **XML, qui est un sous-ensemble de SGML, et qui définit ce que doit être une application de XML**
- **XLL, apporte à XML des fonctionnalités d'adressage et de création de liens**
- **XSL, est un langage permettant d'associer des informations stylistiques pour la présentation d'un document XML**

Le point sur XSL

- XSL est accessible aux utilisateurs au niveau des marqueurs
- C'est une solution déclarative pour la plupart des cas
- Une extension vers des possibilités de "scripting" permet d'envisager des cas complexes
- Les possibilités déclaratives de XSL permettent :
  - Le formatage des éléments sources basé sur la position, la valeur des attributs, l'ascendance/descendance
  - La création de processus de formatage incluant des paragraphes des tables, des lignes des boîtes, etc.

- **Un sens d'écriture indépendant des données**
- **Des possibilités de suppression, duplication, génération, et ré-ordonnancement des données.**

XSL, perspectives d'avenir

Les racines de XSL

- XSL s'appuie sur DSSSL et CSS
- XSL s'adresse à un public d'utilisateurs différents
- XSL propose différentes syntaxes
- DSSSL apporte son modèle de formatage, le concept de flux d'objets et de représentation arborescente de ce flux, ainsi qu'un jeu de caractéristiques de base pour ce flux d'objets
- CSS apporte certaines de ses propriétés de formatage, une base solide pour la présentation des données "online", ainsi que l'apport du média audio

Liens entre XSL et DSSSL

XSL est basé sur DSSL et est ainsi compatible avec ses principes fondamentaux de conception, et son modèle de fonctionnement. Cependant, comme proposé, XSL diverge de DSSSL sur les points suivants :

- **XSL utilise une syntaxe XML (On peut dire que XSL est une application de XML)**
- **XSL suggère certaines extensions au langage de construction de l'arbre des flux d'objets**
- **XSL propose quelques nouvelles classes de flux objet afin de supporter des fonctionnalités de formatage nécessaire sur le web.**

Liens entre XSL et CSS

- XSL supporte toutes les fonctionnalités de la norme CSS1.
- XSL à été conçu dans la perspective de pouvoir facilement "traduire" une feuille de style CSS en feuille de style XSL, et vice-versa
- XSL apporte la ré-organisation des données, les flux objets, et certaines autres caractéristiques
- XSL apporte un mécanisme de sélection des cibles stylistiques plus performant que celui de CSS
- XSL s'enrichit des fonctionnalités audios, onlines, et interactives de CSS

Les précédentes tentatives de feuilles de style déclaratives

L'idée de feuille de style déclaratives basées sur SGML n'est pas neuve. En fait, plusieurs tentatives ont vues le jour par le passé :

- **FOSI, qui à été définit par le "CALS Output Specification" il y a une dizaine d'années, est aujourd'hui utilisé par les composeur "ADEPT" et "DL".**
- **DynaText**
- **Author/Editor**
- **Panorama**

Ces feuilles de styles sont tout a fait adaptées à leur contexte.

XSL bénéficie de l'expérience de ces précurseurs dans le domaine des feuilles de

styles déclaratives.

Les éléments clés de l'ajout de style

- Possibilité de transformer la structure originelle de la source de données
- Définition de la structure objet résultante
- Fourniture d'un mécanisme de localisation (motifs, contexte, sélecteurs, requêtes
- Une action exécuter sur le contenu traité qui inclu généralement :
  - Un jeu d'objet résultant du traitement comprenant des règles de formatage qui sont combinées en vue d'obtenir la structure finale du document
  - Un jeu de caractéristiques de formatage, ainsi qu'un moyen de les associer avec les divers objets résultants

- **La possibilité de traitements conditionnels, de processus et de requêtes**

Adaptabilité du processus de composition

- La feuille de style fournit des indications au processus de composition
- Un langage de feuilles de styles standardisé n'implique pas un "formateur" standardisé
- Le processus de composition utilise la structure objet résultante et créer ainsi la sortie formatée
- La feuille de style ne détermine pas obligatoirement tout

DSSSL

Document Style Semantics and Specification Language à été définit par le comité ISO.

- **DSSSL possède une phase de modification initiale, optionelle, de l'arbre objet**
- **DSSSL possède un langage de requêtes permettant de définir des cibles (DSSSL-o suggère juste un lien simple vers l'objet ancêtre**
- **DSSSL possède un jeu de caractéristiques de formatage associé au flux objet**
- **DSSSL possède un langage d'expression**
- **DSSSL possède un modèle pour la structure résultante qui est indépendant de la direction d'écriture du langage**

- **DSSSL possède un jeu d'objets suffisamment extensibles (qui ne sont pas orientés audio, ni vers une quelconque interactivité)**

FOSIs

Formatting Output Specification Instance à été définit par CALS

- **Output Specification (OS) definit FOSIs sans mécanisme de transformation, il permet seulement de générer du texte, d'en supprimer, et dans une certaine mesure de le ré-organiser**

- **FOSIs possède un concept de localisation d'un élément dans son contexte, qui permet de tester les ancêtres d'un élément, et qui peut mettre en jeu des "wild Cards"**

- **FOSIs possède un jeu de caractéristiques fixe, groupées dans un jeu de catégories lui aussi fixe**

- **FOSIs possède une structure de page résultante fixe, fournissant ainsi une mise en page sur une ou plusieurs colonnes de dimensions égales, avec des zones de pied de page et d'entête de page.**

Comparaison des différents langages de feuilles de style

CSS

Cascading Style Sheets à été définit par le W3C

- **CSS n'autorise pas la transformation des données. CSS2 autorisera une transformation simple de texte généré (avant ou après un élément), ainsi que la suppression**

- **CSS fournit un mécanisme de sélecteur pour son modèle de localisation des éléments (attributs CLASS et ID)**

- **CSS possède un jeu fixe de propriétés de formatage**

- **CSS possède, en mode page, une structure résultante qui n'autorise que des formatages mono-colonne avec CSS1, mais qui peuvent être multi-colonnes avec CSS2 (avec des notions d'entête et de pied de page)**

- **CSS à été originellement définit, et optimisé, pour être utilisé avec le langage HTML. Les travaux les plus récents sur cette norme visent à l'ouvrir à tout document dont la structure est basée sur un jeu d'éléments quel qu'il soit (soit n'importe quelle application xml).**

XSL, langage de transformation de document

- XSL peut être utiliser pour créer un processus de transformation de documents qui peut convertir un document XML, en un autre document XML.
- Cette transformation peut aussi conduire "à produire" des documents HTML
- La construction de l'arbre des flux d'objets, peut être vue comme une transformation en un type spécial de document XML appelé "Formating Markup Language" FML dans lequel le flux d'objets et ses caractéristiques sont vus comme un jeu de marqueurs.
- Il est possible d'utiliser une feuille de styles séparée pour engendrer un processus de transformation sur un document, puis de lui appliquer une autre feuille de styles pour lui attacher des informations stylistiques.

Comparaison des différents langages de feuilles de style

XSL, langage de formatage de document

- **L'utilisation de XSL afin de construire un arbre de flux d'objets (et une structure FML), n'entraîne pas de phase de transformation distincte. Ainsi, il est permis de générer du texte, d'en supprimer, ou d'en ré-organiser le contenu.**

- **XSL utilise une syntaxe XML pour définir des motifs de recherche pour son modèle de localisation d'éléments.**

- **XSL possède un jeu fixe de caractéristiques de formatage, associées à un jeu d'objets.**

- **XSL fournit un mécanisme permettant d'utiliser un langage de script, permettant d'étendre ses fonctionnalités de base.**

Un exemple concret : Les recettes

Tous les exemples de ce tutorial utilisent le DTD de l'application COOKML (COOKML est une application déposée par le CEI Alain DESEINE). La structure générale de ce type de document est la suivante :

```
<LISTERECETTES>

<RECETTE TEMPSPREPA="10" TEMPSCUISSON="10" DIFFICULTE="1"
NOMBREPARTS="4" TITRE="...">
<PHOTO URL="..."/>
<DESCRIPTION>...</DESCRIPTION>
<OPERATIONS>
<OPE>... </OPE>
<OPE>... </OPE>
</OPERATIONS>
<CONSEIL>...</CONSEIL>
<INGREDIENTS>
<INGREDIENT NOM="..." QTE="8" UNITE="" CALORIE="76"/>
```

```
<INGREDIENT NOM="..." QTE="200" UNITE="g" CALORIE="16"/>
</INGREDIENTS>
</RECETTE>

</LISTERECETTES>
```

Structure d'une feuille de styles XSL

Les feuilles de styles XSL sont des documents XML. Leurs éléments principaux sont :

L'élément racine de la feuille de styles : <xsl>

Il s'agit de l'élément contenant les différents élément de la feuille de styles (un peu comme le <HTML> du langage HTML.

Les règles de construction : <rule>

Permet de localiser les éléments cibles dans le source XML, et construit l'arbre des objets du document.

Les règles de style: <style-rule>

Permet de localiser les éléments cibles dans le source XML, et de leurs assigner des caractéristiques de formatage

Les styles nommés : <define-style>

Permet de définir un nom pour "logique" pour une série de règles

Les macros : <define-macro>

Permet de définir des macros commandes de traitement

Les scripts : <define-script>

Permet de définir des scripts

Les règles de construction

Les règles de construction sont toujours constituées de deux parties. La première définie un ou plusieurs motifs pour les éléments cibles du document source. La deuxième partie indique l'action à entreprendre, ou le résultat si les motifs de la première partie sont rencontrés dans le document source XML.

```
<rule>
  motif 1
  motif 2
  ...
  motif n
  résultat
</rule>
```

Les motifs simples

Le motif le plus simple est celui qui identifie un élément simple par son nom de manière inconditionnelle. Cet exemple sélectionne tous les éléments "OPE"

```
<target-element type="OPE"/>
```

Exemple 1

Les motifs permettent également d'identifier des éléments en fonction de leur contexte. Cet exemple sélectionne tous les éléments "OPE" qui sont contenus dans un élément "OPERATIONS"

```
<element type="OPERATIONS">
  <target-element type="OPE"/>
</element>
```

Exemple 2

Les jokers

La norme XSL comprend deux "jokers" <element> sans "type" et <any>. Le motif suivant permet de sélectionner tous les éléments "OPE" qui appartienent a n'importe quel élément à l'intérieur d'une "RECETTE".

```
<element type="RECETTE">
    <element>
        <target-element type="OPE"/>
    </element>
</element>
```

Le motif suivant permet de sélectionner tous les éléments "OPE" quelque soit son imbrication dans le document (sa profondeur d'imbrication) à l'intérieur d'une "RECETTE".

```
<element type="RECETTE">
    <any>
        <target-element type="OPE"/>
```

```
    </any>
</element>
```

Les attributs

Les motifs peuvent également baser leurs recherche sur les attributs des éléments du document source. L'exemple suivant recherche les éléments "INGREDIENTS" dont la valeur de l'attribut "NOM" est "Sel".

```
<target-element type="INGREDIENTS">
  <attribute name="NOM" value="Sel"/>
</target-element>
```

Exemple 3

Cet autre exemple recherche les éléments "INGREDIENTS" qui possèdent un attribut "QTE" et un attribut "UNITE" , quel que soit la valeur de ces deux attributs.

```
<target-element type="INGREDIENT">
  <attribute name="QTE" has-value="yes"/>
  <attribute name="UNITE" has-value="yes"/>
</target-element>
```

Exemple 4

Bien évidemment les notions de recherche de contexte, de jokers et de recherche d'attributs peuvent être conmbinées afin d'obtenir des mécanisme de sélection très complexes.

Les qualifiants

Les qualifiants permettent de baser le motif de recherche sur la position de l'élément recherché dans le document source. L'exemple de règle suivante ajouterais la phrase "Liste des ingrédients : " devant le premier ingrédient rencontrer dans le document source.

```
<rule>
  <target-element type="INGREDIENT" position="first-of-type"/>
  <sequence>
    <paragraph font-weight="bold" font-size="18pt"
               line-spacing="24pt" space-before="12pt">
       <literal>Liste des ingrédients : </literal>
    </paragraph>
    <paragraph>
      <children/>
    </paragraph>
  </sequence>
<rule>
```

Règles d'arbitrage entre les motifs

XSL définit un jeu complet de règles d'arbitrage afin de sélectionner la bonne règle quand plusieurs sont suceptibles de correspondrent. C'est en générale le motif le plus spécifique qui est utilisé au détriment du plus simple. Considérons les exemples suivants :

```
<target-element type="INGREDIENT"/>
```

et

```
<element type="INGREDIENTS">
  <target-element type="INGREDIENT"/>
</element>
```

Bien que les deux règles ci-dessus sélectionnent un élément "INGREDIENT", la seconde, qui est plus restrictive que la première, sera celle retenue, puisqu'elle sélectionne les éléments "INGREDIENT" appartenant à une liste d'ingrédients "INGREDIENTS".

A un instant donné une seule règle peut convenir à un élément donné.

Les actions simples

L'action la plus simple consiste à simplement entourer les éléments enfants d'un élément.

```
<paragraph>
  <children/>
</paragraph>
```

Il y a deux types d"éléments dans la partie "action" d'une règle de construction de XSL : Les objets générés, et les éléments de développement de XSL.

Les actions des règles de construction

Les objets de flux

XSL utilise un jeu d'objet de flux emprunté à DSSSL. Les objets de flux les plus couramment utilisés sont les suivants :

<paragraph>

Un bloc de texte

<sequence>

Texte inclu, ou sequence de bloc de texte

<display-group>

Un groupe de blocs

<table>

Une table

<external-graphic>

Un lien vers un graphique externe

<link>

Un lien HyperTexte

Caractéristiques des objets de flux

Les caractéristiques d'un objet de flux fournissent des informations complémentaires vis à vis de la présentation du document.

```
<paragraph space-before="12pt"
           space-after="36pt"
           font-weight="bold"
           font-size="24pt"
           line-spacing="36pt">
  <children/>
</paragraph>
```

Les classes de caractéristiques

Les classe de caratéristiques peuvent être subdivisée en deux catégories :

Les caractéristiques héritées

Elles s'appliquent aux objets auxquels elles sont destinées, ainsi qu'à tous leurs éléments enfants.

Les caractéristiques non héritées

Elles s'appliquent aux objets auxquels elles sont destinées.

Les éléments d'exécution

Il existe dans la norme XSL un certain nombre déléments qui sont utilisés pour contrôler l'analyse du document source. Les principaux sont :

<children/>

Demande l'analyse de tous les éléments enfants de l'élément courant.

<literal> ou <text>

Permet d'insérer du texte litéral.

<eval>

Permet d'insérer le résultat d'une expréssion ECMAScript.

<select-elements> ou <select>

Permet d'analyser des éléments arbitraires.

D'autres éléments d'execution devraient voir le jour afin de gérer les constructions standards, comme les listes par exemple.

Formater l'élément DESCRIPTION

Dans notre exemple la description est formatée comme un simple paragraphe :

```
<rule>
  <target-element type="DESCRIPTION"/>

  <paragraph space-before="8pt">
    <children/>
  </paragraph>
</rule>
```

Formater l'élément CONSEIL

Dans notre exemple le CONSEIL de la RECETTE est formaté en gras avec une fonte large :

```
<rule>
  <element type="RECETTE">
    <target-element type="CONSEIL"/>
  </element>

  <paragraph font-weight="bold"
             font-size="24pt"
             line-spacing="36pt">
    <children/>
  </paragraph>
</rule>
```

Formatage de l'élément NOTE 1

Dans l'exemple suivant l'élément NOTE est formaté différemment en fonction de la valeur de son attribut STATUS. Si celui-ci est égal à "Personnel" le contenu de l'élément NOTE ne doit pas être affiché. Si par contre l'attribut est égal à autre chose ou n'est pas présent l'élément NOTE doit être traité comme un paragraphe.

```
<rule>
  <target-element type="DESCRIPTION"/>
  <target-element type="NOTE"/>

  <paragraph space-before="8pt">
    <children/>
  </paragraph>
</rule>

<rule>
  <target-element type="note">
    <attribute name="STATUS"
```

```
                value="PERSONNEL"/>
  </target-element>

  <empty/>
</rule>
```

Formatage de l'élément NOTE 2

Si l'attribut STATUS de notre élément NOTE est égal a "CREDIT" alors on insére un texte litéral devant le contenu de l'élément NOTE.

```
<rule>
  <target-element type="NOTE">
    <attribute name="STATUS" value="CREDIT"/>
  </target-element>

  <paragraph space-before="8pt">
    <literal>Cette recette est de : </literal>
    <children/>
  </paragraph>
</rule>
```

Formatage de l'élément NOTE 3

Un élément NAME à l'intérieur d'un élément NOTE dont l'attribut STATUS est égal à CREDIT est lui aussi formaté d'une manière spécifique

```
<rule>
  <element type="NOTE">
    <attribute name="STATUS" value="CREDIT"/>
    <target-element type="NAME"/>
  </element>

  <sequence font-posture="italic">
    <children/>
  </sequence>
</rule>
```

Formatage de l'élément INGREDIENTS

Formater l'ensemble des ingrédients en une liste d'ingrédient constitue la règle la plus complexe de notre exemple.

```
<rule>
  <element type="INGREDIENTS">
    <target-element type="INGREDIENT">
      <attribute name="QTE"
                 has-value="yes"/>
      <attribute name="UNITE"
                 has-value="yes"/>
    >/target-element>
  </element>

  <paragraph>
    <eval>attributeString("QTE")</eval>
    <literal> </literal>
    <eval>attributeString("UNITE")</eval>
```

```
    <literal> </literal>
     <children/>
  </paragraph>
</rule>
```

Rendu "OnLine" avec HTML

Un exemple de transformation de document à l'aide de XSL est de transformer un document XML en document HTML. La règle suivante transforme par exemple l'élément DESCRIPTION en titre de niveau 1 (H1).

```
<rule>
  <element type="RECETTE">
    <target-element type="DESCRIPTION"/>
  </element>

  <H1><children/></H1>
</rule>
```

Le plugin pour Internet Explorer 4 MSXSL de chez MICROSOFT permet de transformer un document XML en document HTML à l'aide d'une feuille de style XSL.

Rendu "OnLine" avec ajout de style

Le rendu "online", comme le rendu à l'impression, peuvent être obtenu avec un ajout de style. Aujourd'hui ce processus nécessite plusieurs phases :

- **Formatage du document XML avec une feuille de style XSL. Le résultat est un document XML.**
- **Le document XML résultant est transformé en document HTML par une feuille de style XSL.**
- **Le navigateur client affiche le document HTML résultant.**

Dans le futur, les navigateurs devraient être capable de restituer directement des flux d'objets XML, simplifiant ainsi le processus.

Transformer la liste des INGREDIENTS en table CALS

XSL peut être utilisé pour des transformations d'ordre général. La règle ci-dessous transforme la liste des ingédients en une table CALS :

```
<rule>
  <target-element type="ingredient-list"/>
  <TITLE>Ingredients</TITLE>
  <TGROUP COLS="3">
  <THEAD>
  <ROW>
  <ENTRY>Quantity</ENTRY>
  <ENTRY>Units</ENTRY>
  <ENTRY>Ingredient</ENTRY>
  </ROW>
  </THEAD>
  <TBODY>
    <children/>
  </TBODY>
```

```
  </TGROUP>
</rule>
```

Cette possibilité est fondamentale lorsque l'on désire transformer la structure même d'un document.

La règle implicite

La règle implicite est utilisée par le moteur stylistique pour tout élément pour lequel on ne peut pas trouver de règle implicite. Il s'agit en fait d'une règle qui s'applique aux élément pour lesquels vous n'avez pas définit de règle. En général, la règle implicite se contente de déclencher l'analyse du contenu de l'élément auquel elle s'applique.

```
<rule>
  <target-element/>
</rule>
```

XSL autorise toutefois la surcharge de cette règle implicite. Dans le cas d'une feuille de style transformant un document XML en document HTML, cette nouvelle règle implicite pourrait être :

```
<rule>
```

```
  <target-element/>
  <SPAN STYLE="color: red"><children/></SPAN>
</rule>
```

Cette nouvelle règle implicite ferait apparaître dans le navigateur tous les éléments qui ne possède pas de règle explicite en rouge.

La règle "racine"

Cette règle permet de traiter tous les éléments contenus dans un document XML. Dans le cas de la transformation d'un document XML en HTML cette règle pourrait être :

```
<rule>
  <root/>
  <HTML><HEAD><TITLE>Recette</TITLE></HEAD><BODY>
    <children/>
  </BODY></HTML>
</rule>
```

Cette règle permet ainsi de générer toutes les entête HTML.

Filtrage

L'élément d'exécution *select* (ou *select-elements*) permet d'effectuer un filtrage. Ceci permet notament d'effectuer des réorganisation d'éléments dans un document.

```
<rule>
  <target-element type="RECETTE"/>

  <simple-page-sequence left-margin="1in"
                        right-margin="1in"
                        top-margin="1in"
                        bottom-margin="1in">
    <select from="children">
      <target-element type="DESCRIPTION"/>
    </select>
    <select from="children">
      <target-element type="INGREDIENTS"/>
    </select>
```

. . .

Filtrage (suite)

```
  ...

    <select from="children">
      <target-element type="OPERATIONS"/>
    </select>
    <select from="children">
      <target-element type="CONSEIL"/>
    </select>
    <select from="children">
      <target-element type="PHOTO"/>
    </select>
  </simple-page-sequence>
</rule>
```

Mode 1

Dans certains cas de mise en page complexe de document, il peut s'avérer utile de modifier plusieurs fois le même élément. Par exemple, une titre de chapitre doit apparaître en tête du chapitre, mais aussi dans une table des matières, et dans l'index. La norme XSL utilise les *modes* pour contrôler les règles de sélection.

- **L'attribut *mode* des éléments <select-elements> et <children> indique quel *mode* utiliser lors du traitement des éléments.**
- **L'attribut *mode* de l'élément <rule> identifie le mode auquel la règle s'applique.**
- **Si aucune règle du mode sélectionner ne correspond, c'est la règle du mode par défaut qui est utilisée, si il n'en existe pas c'est la règle implicite qui est employée.**

Mode 2

Cette règle formate une liste de recettes avec en tête de document la liste des recettes

```
<rule>
  <target-element type="LISTERECETTES"/>
  <sequence>
    <paragraph font-weight="bold" font-size="18pt"
               line-spacing="24pt">
      <score type="after">
        <literal>Liste des recettes</literal>
      </score>
    </paragraph>
    <select from="descendants" mode="toc">
      <element type="RECETTE">
        <target-element type="TITRE"/>
      </element>
    </select>
```

```
    <children/>
  </sequence>
</rule>
```

...

Mode 3

...

Voici maintenant les deux règles qui s'appliquent à l'élément RECETTE en fonction du contexte.

```
<rule mode="toc">
  <element type="RECETTE">
    <target-element type="TITRE"/>
  </element>
  <paragraph font-size="14pt"
             line-spacing="16pt">
    <children/>
  </paragraph>
</rule>

<rule>
  <element type="RECETTE">
```

```
    <target-element type="TITRE"/>
  </element>
  <paragraph font-weight="bold"
             font-size="24pt"
             line-spacing="36pt">
    <children/>
  </paragraph>
</rule>
```

Numérotage

Actuellement, numéroter des éléments recquiert d'utiliser un script. La règle suivante illustre ce que pourrait être la numérotation des opérationsde la recette.

```
<rule>
 <target-element type="OPE"/>
  <paragraph>
    <eval>formatNumber(childNumber(this),"1")</eval>
    <literal> - </literal>
    <children/>
  </paragraph>
</rule>
```

Les fonctions formatNumber() et childNumber() sont deux des nombreuses fonctions qui seront définies dans la version finale de la norme.

Création de liens

La création de lien est réalisée à l'aide de l'élément <link>. La destination du lien est un autre élément du document source.

La règle de construction suivante crée un lien entre le titre de la recette dans la table des matières et la recette par elle même :

```
<rule mode="toc">
  <element type="RECETE">
    <target-element type="TITRE"/>
  </element>

  <paragraph font-size="14pt"
                line-spacing="26pt">
    <eval>formatNumber(childNumber(parent(this)),"1")</eval>
    <literal> - </literal>
    <link destination="=nodeListAddress(this)">
      <children/>
```

```
    </link<>
  </paragraph>
</rule>
```

Les fonctions formatNumber() et childNumber() sont deux des nombreuses fonctions qui seront définies dans la version finale de la norme.

Règles stylistiques 1

Les règles stylistiques apportent le support des fonctionnalités de Cascading Style Sheet (CSS), permattant ainsi que plusieurs règles contribues aux caractéristiques de présentation d'un élément.

Les règles stylistiques sont définies à l'aide de l'élément <style-rule>.

Comme les règles de présentation, les règles stylistiques sont constituées de deux parties. Un motif pemettant d'identifier l'élément cible, et une action à entreprendre sur l'élément ciblé. La règle suivante présente toutes les opérations en rouge.

```
<style-rule>
  <target-element type="OPE"/>
  <apply color="=red"/>
</style-rule>
```

Cette autre règle formate les CONSEIL en italique

```
<style-rule>
  <element type="RECETTE"<
    <target-element type="CONSEIL"/>
  </element>
  <apply font-posture="italic"/>
</style-rule>
```

...

Règles stylistiques 2

...

Les règles de construction suivantes agissent elles aussi sur l'élément CONSEIL

```
<rule>
  <target-element type="CONSEIL"/>
  <children/>
</rule>

<rule>
  <element type="RECETTE">
    <target-element type="CONSEIL"/>
  </element>
  <paragraph font-size="24pt"
             line-spacing="36pt">
    <children/>
  </paragraph>
```

```
</rule>
```

En final l'élément CONSEIL héritera de la règle stylistique la couleur d'écriture rouge, et de la règle de construction une taille de fonte de 24 points et un espace inter-ligne de 36 points.

La première règle sera igorée dans le cas ou l'élément CONSEIL est contenu dans un élément RECETTE (ce qui est toujours le cas si l'on se réferre au DTD de cet exemple), puisqu'elle est moins précise que la seconde.

Les styles nommés

Les styles nommés permettent de référer à un style pré-définit par le biais d'un nom logique. Ceci permet de simplifier l'écriture de la feuille de style, et surtout sa relecture et sa maintenance.

```
<define-style name="heading-style"
              font-weight="bold"
              font-size="18pt"
              line-spacing="24pt"/>

<paragraph use="heading-style"
           space-before="12pt">
   <literal>Preparation</literal>
</paragraph><
```

Les macros

Les macros, définies à l'aide de l'élément <define-macro>, permettent de contruire de objets composites. Elle sont invoquées dans les règles de construction.

```
<define-macro name="heading">
  <arg name="title" default="you forgot the title!"/>
  <paragraph use="heading-style" space-before="12pt">
    <eval>title</eval>
  </paragraph>
</define-macro>

<invoke macro="heading">
  <arg name="title" value="Preparation"/>
</invoke>
```

Elle autorise un passage de paramètres par les attributs "name" et "value".

Définition d'un script

L'élément <define-script> permet de définir des expression calculées qui seront insérées dans le document final.

```
<define-script>
<![CDATA[
var FontSize="10pt";
var HeadSep=FontSize * 1.2
var ParaSep="8pt"

function fib(num) { num <= 0 ? 1 : num + fib(num-1); }
]]>
</define-script>
```

L'utilisation d'une section CDATA évite que les éventuels caractères de marquage ne soit interprêter par le parser XML.

Utiliser le résultat d'un script comme valeur d'une caractéristiques

Les scripts peuvent être invoqués de deux endroits dans une feuille de style XSL. Comme valeur d'une caractéristiques, ou dans un marqueur <eval>.

Si la valeur d'une caractéristique commence avec le signe "=", elle doit être interprétée comme le résultat d'un ECMAScript (Référence à la norme de script du comité ISO, ECMAScript).

```
<rule>
  <target-element type="description"/>

  <paragraph space-before="=ParaSep"
             color="=calcColor()">
    <hildren/>
  <paragraph>
</rule>
```

Ces expressions n'ont rien à voir avec les scripts exécuter par le navigateur qui visualise le document.

Utilisation des script dans l'élément <eval>

Le contenu d'un élément <eval> est toujours évalué comme une expression ECMAScript.

```
<rule>
  <element type="INGREDIENTS">
    <target-element type="INGREDIENT">
    </target-element>
  </element>
  <DT><eval>calcquantity(getAttribute("QTE"),
            parent.parent.getAttribute("UNITE"),
            parent.getAttribute("CALORIE"))</eval>
    <literal> </literal>
    <eval>getAttribute("NOM")</eval>
    <children/>
  </DT>
</rule>
```

Ces expressions n'ont rien à voir avec les scripts exécuter par le navigateur qui visualise le document.

Conclusion

Evévenements

- Co-sousmission par ArborText, Inso, et Microsoft en Septembre 1997
- Création du "XSL Working Group" par le W3C en Janvier 1998
- Premiére rencontre du groupe de travail XSL le 28 Janvier 1998 à SAN JOSE
- Création de la liste de diffusion xsl-list pour les discussions générales concernant la norme XSL
- Annonce d'outils prototypes pour le support de XSL
- Rencontre du groupe de travail XSL a SEATTLE le 28 Mars 1998

Calendrier

- **Cahier des charges de la norme XSL : *Avril 1998***
- **1er document de travail de la norme XSL 1.0 : *Juillet 1998***
- **2ème document de travail de la norme XSL 1.0 : *Novembre 1998***
- **3ème document de travail de la norme XSL 1.0 : *Février 1999***
- **Proposition de recommandation de la norme XSL 1.0 : *Mai 1999***

Application exemple

Vous pouvez consulter notre application exemple à l'adresse suivante

http://www.cabinfo.com/cookml.htm

Attention !!! Pour visualiser cette application exemple vous devez *impérativement* utiliser Internet Explorer 4 (ou supérieur).

Conclusion

Références

Ressources XSL :

- Le texte de la soumission originelle : http://www.w3.org/TR/NOTE-XSL-970910
- La page XSL de Robin Cover : http://www.sil.org/sgml/xsl.html
- La page sur le style au W3C : http://www.w3.org/Style/XSL/
- La liste de diffusion xsl-list : xsl-list-request@mulberrytech.com
- Ce tutorial : http://www.cabinfo.com/xsltut/xsl_tut.htm

Quelques outils XSL :

- XML Styler de ArborText : http://www.arbortext.com/xmlstyler/

- **Support de XSL dans IE4 de Microsoft : http://www.microsoft.com/xml/xsl/**
- **XSLJ de Henry Thompson : http://www.ltg.ed.ac.uk/~ht/xslj.html**
- **Jade de James Clark : htttp://www.jclark.com/jade/**

Exemple 1

```
<?XML VERSION="1.0"?>
<!DOCTYPE LISTERECETTES SYSTEM "xml_dtd.dtd">

<LISTERECETTES>

<RECETTE TEMPSPREPA="10" TEMPSCUISSON="10" DIFFICULTE="1" NOMBREPARTS="4"
TITRE="OMELETTE AUX CHAMPIGNONS">
<PHOTO URL="FACTORY.BMP"/>
<DESCRIPTION>
Des champignons de Paris revenus au beurre garnissent cette omelette toujours
appréciée.
</DESCRIPTION>
<OPERATIONS>
<OPE>Nettoyer les champignons. </OPE>
<OPE>Coupez les champignons en petites lamelles. </OPE>
<OPE>Faites fondre la moitié du beurre dans une poêle et mettez-y les champignons à
dorer. Retirer les au bout de 10 minutes et gardez les au chaud. </OPE>
<OPE>Mettez le reste du beurre dans la poêle. Battez vivement les oeufs avec le sel
et le poivre. Faites les cuire en omelette. </OPE>
<OPE>Lorsqu'elle est à point, parsemez avec les champignons et roulez la en la
faisant glisser sur un plat chaud. </OPE>
</OPERATIONS>
<CONSEIL>
Je décore le dessus de l'omelette avec une grosse tête de champignon coupée en
```

```
lamelles que je dispose en rosace.
</CONSEIL>
<INGREDIENTS>
<INGREDIENT NOM="oeufs" QTE="8" UNITE="" CALORIE="76"/>
<INGREDIENT NOM="de champignons de Paris" QTE="200" UNITE="g" CALORIE="16"/>
<INGREDIENT NOM="de beurre" QTE="40" UNITE="g" ALORIE="760"/>
<INGREDIENT NOM="Sel" QTE="" UNITE="" />
<INGREDIENT NOM="Poivre" QTE="" UNITE="" />
</INGREDIENTS>
</RECETTE>

</LISTERECETTES>
```

Exemple 2

```
<?XML VERSION="1.0"?>
<!DOCTYPE LISTERECETTES SYSTEM "xml_dtd.dtd">

<LISTERECETTES>

<RECETTE TEMPSPREPA="10" TEMPSCUISSON="10" DIFFICULTE="1" NOMBREPARTS="4"
TITRE="OMELETTE AUX CHAMPIGNONS">
<PHOTO URL="FACTORY.BMP"/>
<DESCRIPTION>
Des champignons de Paris revenus au beurre garnissent cette omelette toujours
appréciée.
</DESCRIPTION>
<OPERATIONS>
<OPE>Nettoyer les champignons. </OPE>
<OPE>Coupez les champignons en petites lamelles. </OPE>
<OPE>Faites fondre la moitié du beurre dans une poêle et mettez-y les champignons à
dorer. Retirer les au bout de 10 minutes et gardez les au chaud. </OPE>
<OPE>Mettez le reste du beurre dans la poêle. Battez vivement les oeufs avec le sel
et le poivre. Faites les cuire en omelette. </OPE>
<OPE>Lorsqu'elle est à point, parsemez avec les champignons et roulez la en la
faisant glisser sur un plat chaud. </OPE>
</OPERATIONS>
<CONSEIL>
Je décore le dessus de l'omelette avec une grosse tête de champignon coupée en
```

```
lamelles que je dispose en rosace.
</CONSEIL>
<INGREDIENTS>
<INGREDIENT NOM="oeufs" QTE="8" UNITE="" CALORIE="76"/>
<INGREDIENT NOM="de champignons de Paris" QTE="200" UNITE="g" CALORIE="16"/>
<INGREDIENT NOM="de beurre" QTE="40" UNITE="g" ALORIE="760"/>
<INGREDIENT NOM="Sel" QTE="" UNITE="" />
<INGREDIENT NOM="Poivre" QTE="" UNITE="" />
</INGREDIENTS>
</RECETTE>

</LISTERECETTES>
```

Exemple 3

```
<?XML VERSION="1.0"?>
<!DOCTYPE LISTERECETTES SYSTEM "xml_dtd.dtd">

<LISTERECETTES>

<RECETTE TEMPSPREPA="10" TEMPSCUISSON="10" DIFFICULTE="1" NOMBREPARTS="4"
TITRE="OMELETTE AUX CHAMPIGNONS">
<PHOTO URL="FACTORY.BMP"/>
<DESCRIPTION>
Des champignons de Paris revenus au beurre garnissent cette omelette toujours
appréciée.
</DESCRIPTION>
<OPERATIONS>
<OPE>Nettoyer les champignons. </OPE>
<OPE>Coupez les champignons en petites lamelles. </OPE>
<OPE>Faites fondre la moitié du beurre dans une poêle et mettez-y les champignons à
dorer. Retirer les au bout de 10 minutes et gardez les au chaud. </OPE>
<OPE>Mettez le reste du beurre dans la poêle. Battez vivement les oeufs avec le sel
et le poivre. Faites les cuire en omelette. </OPE>
<OPE>Lorsqu'elle est à point, parsemez avec les champignons et roulez la en la
faisant glisser sur un plat chaud. </OPE>
</OPERATIONS>
<CONSEIL>
Je décore le dessus de l'omelette avec une grosse tête de champignon coupée en
```

```
lamelles que je dispose en rosace.
</CONSEIL>
<INGREDIENTS>
<INGREDIENT NOM="oeufs" QTE="8" UNITE="" CALORIE="76"/>
<INGREDIENT NOM="de champignons de Paris" QTE="200" UNITE="g" CALORIE="16"/>
<INGREDIENT NOM="de beurre" QTE="40" UNITE="g" ALORIE="760"/>
<INGREDIENT NOM="Sel" QTE="" UNITE="" />
<INGREDIENT NOM="Poivre" QTE="" UNITE="" />
</INGREDIENTS>
</RECETTE>

</LISTERECETTES>
```

Exemple 4

```
<?XML VERSION="1.0"?>
<!DOCTYPE LISTERECETTES SYSTEM "xml_dtd.dtd">

<LISTERECETTES>

<RECETTE TEMPSPREPA="10" TEMPSCUISSON="10" DIFFICULTE="1" NOMBREPARTS="4"
TITRE="OMELETTE AUX CHAMPIGNONS">
<PHOTO URL="FACTORY.BMP"/>
<DESCRIPTION>
Des champignons de Paris revenus au beurre garnissent cette omelette toujours
appréciée.
</DESCRIPTION>
<OPERATIONS>
<OPE>Nettoyer les champignons. </OPE>
<OPE>Coupez les champignons en petites lamelles. </OPE>
<OPE>Faites fondre la moitié du beurre dans une poêle et mettez-y les champignons à
dorer. Retirer les au bout de 10 minutes et gardez les au chaud. </OPE>
<OPE>Mettez le reste du beurre dans la poêle. Battez vivement les oeufs avec le sel
et le poivre. Faites les cuire en omelette. </OPE>
<OPE>Lorsqu'elle est à point, parsemez avec les champignons et roulez la en la
faisant glisser sur un plat chaud. </OPE>
</OPERATIONS>
<CONSEIL>
Je décore le dessus de l'omelette avec une grosse tête de champignon coupée en
```

```
lamelles que je dispose en rosace.
</CONSEIL>
<INGREDIENTS>
<INGREDIENT NOM="oeufs" QTE="8" UNITE="" CALORIE="76"/>
<INGREDIENT NOM="de champignons de Paris" QTE="200" UNITE="g" CALORIE="16"/>
<INGREDIENT NOM="de beurre" QTE="40" UNITE="g" ALORIE="760"/>
<INGREDIENT NOM="Sel" QTE="" UNITE="" />
<INGREDIENT NOM="Poivre" QTE="" UNITE="" />
</INGREDIENTS>
</RECETTE>

</LISTERECETTES>
```
