XML

XPATH, XSLT

XPath

* Syntaxe non XML
* Langage de requête
* Basé sur des chemins d'accès
* Fonctionne sur des chemins relatifs ou absolus.
* S'applique à des nœuds, des booléens, des nombres et des chaînes de caractères.

XPath

* Chemin de localisation:
  * Un axe : direction dans l'arbre
  * Un type de nœud à localiser
  * Des conditions (0 à n)

XPath

* Les axes principaux :
  * child
  * ancestor
  * ancestor-or-self
  * descendant
  * descendant-or-self
  * parent

XPath

* Autres axes:
  * preceding
  * preceding-sibling
  * following
  * following-sibling

  * self
  * attribute
  * namespace

XPath

* Type de nœud:
  * Nom de l'élément. * : raccourcit pour tous les éléments ou attributs (selon axe).
* Fonction pour reconnaitre chaque type de nœud:
  * text() : uniquement nœud texte.
  * node() : tous les nœuds.
  * comment(): les commentaires.
  * processing-instruction() : instruction de traitement.

XPath

* Exemples requêtes:
  * child::para
  * child::*
  * descendant::*
  * attribute::titre

  * child::personne/child::email
  * /child::carnet/descendant::email

XPath

* Prédicat (condition) :
  * child::personne[child::nom='UnNom']
  * child::nom[self::nom='UnNom']
  * child::personne[attribute::numero='10']
  * child::personne[attribute::numero]
  * child::personne[attribute::nom or attribute::prenom]
  * child::personne[attribute::nom= 'UnNom' and attribute::prenom= 'UnPrenom']
  * child::personne[not(attribute::*)]

XPath

* Opérateur :

Opérateur	Description
+, -	Addition, soustraction
*,div	Multiplication, division
mod	Modulo
=, !=	Égal, différent
<, <=	Inférieur, inférieur ou égal
>, >=	Supérieur, supérieur ou égal
or, and	Ou, et (booléen)
not( )	Non(booléen)

XPath

* Fonctions :
  * string-length(chaine) : longueur d'une chaine
  * starts-with(chaine1,chaine2) : test si chaine1 commence par chaine2translate(chaine, caractère source, caractère dest)
  * floor(décimal) : arrondi inférieur.
  * count(nodeSet?) : nombre de nœuds.
  * Liste fonctions : http://www.w3.org/TR/xpath#corelib

XPath

* Exemples :
  * child::personne[3]
  * child::personne[last()]
  * child::personne[position()>1]
  * count(descendant::personne)

XPath

* Forme abrégé:
  * Axe child implicite
    * personne/nom
  * // désigne tout les descendants.
    * //personne
  * Attribut est préfixé par @
    * /contacts/personne/@num
  * Le texte est désigné par text()
    * /contacts/personne/adresse/text()
  * .. Désigne l'élément parent
    * //nom[name(..)='personne']
  * . Représente l'élément courant
    * //personne[./@num=10]

XSLT

* eXtensible Stylesheet Language Transformations
* Langage XML
* Transforme du XML en un autre format texte (xml, html, csv …)
* Utilise XPath

XSLT

* Squelette :

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.     <xsl:output method="xml" indent="yes"/>
4.         […]
5. </xsl:stylesheet>
```

XSLT

* output :
  * method : text, html, xml
  * encoding : jeux de caractères
  * indent = yes ou no
  * media-type : type MIME de sortie
  * doctype-public : DTD public
  * doctype-system : DTD system

XSLT

* XML :

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <?xml-stylesheet type="text/xsl" href="intro.xslt"?>
3. <document>
4.   <question>Que signifie XSL?</question>
5.   <reponse>eXtensible Stylesheet language.</reponse>
6. </document>
```

XSLT

* Fichier xslt:

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.      <xsl:output method="html" indent="yes"/>
4.    <xsl:template match="/">
5.      <html><head><title>QCM</title></head>
6.      <body>
7.        <h2><xsl:value-of select="document/question"/></h2>
8.        <h3><xsl:value-of select="document/reponse"/></h3>
9.      </body>
10.     </html>
11.   </xsl:template>
12. </xsl:stylesheet>
```

XSLT

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <?xml-stylesheet type="text/xsl" href="intro.xslt"?>
3.  <document>
4.    <questrep>
5.      <question>Que signifie XSL?</question>
6.      <reponse>eXtensible Stylesheet language.</reponse>
7.    </questrep>
8.    <questrep>
9.      <question>Que signifie XML?</question>
10.     <reponse>eXtensible Markup language.</reponse>
11.   </questrep>
12. </document>
```

XSLT

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.      <xsl:output method="html" indent="yes"/>
4.    <xsl:template match="/">
5.      <html><head><title>QCM</title></head>
6.      <body>
7.        <h2><xsl:value-of select="document/questrep/question"/></h2>
8.        <h3><xsl:value-of select="document/questrep/reponse"/></h3>
9.      </body>
10.     </html>
11.   </xsl:template>
12. </xsl:stylesheet>
```

XSLT: for-each

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.      <xsl:output method="html" indent="yes"/>
4.    <xsl:template match="/">
5.      <html><head><title>QCM</title></head>
6.      <body>
7.        <xsl:for-each select="document/questrep">
8.          <h2><xsl:value-of select="question"/></h2>
9.          <h3><xsl:value-of select="reponse"/></h3>
10.       </xsl:for-each>
11.     </body>
12.     </html>
13.   </xsl:template>
14. </xsl:stylesheet>
```

XSLT: sort

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <?xml-stylesheet type="text/xsl" href="intro.xslt"?>
3.  <document>
4.    <questrep id="2">
5.      <question>Que signifie DTD?</question>
6.      <reponse>Document Type Definition.</reponse>
7.    </questrep>
8.    <questrep id="3">
9.      <question>Que signifie XSL?</question>
10.     <reponse>eXtensible Stylesheet language.</reponse>
11.   </questrep>
12.   <questrep id="1">
13.     <question>Que signifie XML?</question>
14.     <reponse>eXtensible Markup language.</reponse>
15.   </questrep>
16. </document>
```

XSLT : sort

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.      <xsl:output method="html" indent="yes"/>
4.    <xsl:template match="/">
5.      <html><head><title>QCM</title></head>
6.      <body>
7.        <xsl:for-each select="document/questrep">
8.          <xsl:sort select="@id"/>
9.          <h2><xsl:value-of select="question"/></h2>
10.       <h3><xsl:value-of select="reponse"/></h3>
11.       </xsl:for-each>
12.     </body>
13.     </html>
14.   </xsl:template>
15. </xsl:stylesheet>
```

XSLT : sort

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.      <xsl:output method="html" indent="yes"/>
4.    <xsl:template match="/">
5.      <html><head><title>QCM</title></head>
6.      <body>
7.        <xsl:for-each select="document/questrep">
8.          <xsl:sort select="@id" order="descending"/>
9.          <h2><xsl:value-of select="question"/></h2>
10.       <h3><xsl:value-of select="reponse"/></h3>
11.       </xsl:for-each>
12.     </body>
13.     </html>
14.   </xsl:template>
15. </xsl:stylesheet>
```

XSLT : génération commentaire

* Génération de commentaire :

```
1.  <xsl:comment>ceci est un
    commentaire</xsl:comment>
```

* Résultat html :

```
1. <!--ceci est un commentaire-->
```

XSLT: template

* <xsl:template match="pattern"
* name="nom" mode="RenduPartiel"
priority="1" >
* </xsl:template>
* match: indique l'élément concerné par la règle de template
* mode: crée un mode de traitement du nœud
* priority: fixe le niveau de priorité (de -9 à +9)

XSLT: template

```
<xsl:template name="question" >
  <h2><xsl:value-of select="."/></h2>
</xsl:template>

<xsl:template match="reponse">
  <h3><xsl:value-of select="."/></h3>
</xsl:template>
```

XSLT: template

```
1.   <xsl:template match="/">
2.      <html>
3.        <head>
4.          <title>QCM</title>
5.        </head>
6.        <body>
7.          <xsl:for-each select="document/questrep">
8.            <xsl:call-template name="question" />
9.            <xsl:apply-templates select="reponse" />
10.         </xsl:for-each>
11.       </body>
12.     </html>
13.   </xsl:template>
```

XSLT: element, attribute

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <?xml-stylesheet type="text/xsl"
   href="intro.xslt"?>
3. <agenda>
4.   <contact>
5.     <nom>Michu</nom>
6.     <prenom>Michelle</prenom>
7.   </contact>
8.   <contact>
9.     <nom>Michu</nom>
10.    <prenom>Robert</prenom>
11.  </contact>
12.</agenda>
```

XSLT: element, attribute

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.      <xsl:output method="xml" indent="yes"/>
4.    <xsl:template match="/">
5.     <xsl:element name="agenda">
6.      <xsl:for-each select="agenda/contact">
7.        <xsl:element name="contact">
8.          <xsl:attribute name="nom">
9.            <xsl:value-of select="nom"/>
10.         </xsl:attribute>
11.         <xsl:attribute name="prenom">
12.           <xsl:value-of select="prenom"/>
13.         </xsl:attribute>
14.       </xsl:element>
15.     </xsl:for-each>
16.    </xsl:element>
17.   </xsl:template>
18. </xsl:stylesheet>
```

XSLT: attribute-set

```
1.  <xsl:attribute-set name="np">
2.     <xsl:attribute name="nom">
3.       <xsl:value-of select="nom"/>
4.     </xsl:attribute>
5.     <xsl:attribute name="prenom">
6.       <xsl:value-of select="prenom"/>
7.     </xsl:attribute>
8.   </xsl:attribute-set>

1.  <xsl:element name="contact" use-attribute-sets="np"/>
```

XSLT : if

```
1. <xsl:if test="position() = 1">
2.         […]
3. </xsl:if>
```

* Utilise des conditions XPath
* Pas de else

XSLT : choose

```
1.  <xsl:choose>
2.          <xsl:when test="test1">
3.            […]
4.          </xsl:when>
5.          <xsl:when test="test2">
6.            […]
7.          </xsl:when>
8.          <xsl:otherwise>
9.            […]
10.         </xsl:otherwise>
11.  </xsl:choose>
```

XSLT: paramètre de template

```
1.   <xsl:template match="/">
2.      <xsl:apply-templates select="contacts/personne">
3.        <xsl:with-param name="P1">10</xsl:with-param>
4.        <xsl:with-param name="P2">20</xsl:with-param>
5.      </xsl:apply-templates>
6.    </xsl:template>
```

```
1.    <xsl:template match="personne">
2.      <xsl:param name="P1"/>
3.      <xsl:param name="P2"/>
4.      <xsl:value-of select="$P1+$P2"/>
5.      […]
6.    </xsl:template>
```

XSLT : variables

```
1.   <xsl:for-each select="personne">
2.        <xsl:variable name="nbEnfant" select="count(enfant)"/>
3.        <xsl:choose>
4.          <xsl:when test="$nbEnfant &lt; 2">
5.            […]
6.          </xsl:when>
7.          <xsl:when test="$nbEnfant &gt; 4">
8.            […]
9.          </xsl:when>
10.         <xsl:otherwise>
11.           […]
12.         </xsl:otherwise>
13.       </xsl:choose>
```

XSLT : copy-of

* Effectue une copie complète de l'arbre selectionné

```
1.  <xsl:template match="/">
2.      <carnetBis>
3.        <xsl:copy-of select="//personne"/>
4.      </carnetBis>
5.    </xsl:template>
```

```
1.  <carnetBis>
2.      <personne nom="Michu" prenom="Michelle"/>
3.      <personne nom="Michu" prenom="Robert"/>
4.    </carnetBis>
```

XSLT : text

* `<xsl:text disable-output-escaping="yes"> […] </xsl:text>`
* disable-output-escaping : permet (ou non) l'utilisation des caractères d'échapement

* Conserve les espaces blancs
* Permet d'utiliser des caractère comme « & » ou « > »

XSLT : number

* `<xsl:number value="position()" format="a)" />`
* count: décompte les nœuds
* format: spécifie le format de sortie des nombres
* from: permet d'indiquer le nœud à partir duquel le décompte est effectué
* grouping-separator: caractère de séparation des groupes
* grouping-size: spécifie la taille des groupes

XSLT : decimal-format

* Définit la manière selon laquelle un nombre est convertit en texte.

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3.    <xsl:output method="xml" indent="yes"/>
4.    <xsl:decimal-format digit="D" decimal-separator="." infinity="INF"/>
```

XSLT : message

* Affiche un message, pour les erreurs.

```
1. […]
2. </xsl:when>
3. <xsl:otherwise>
4.   <xsl:message terminate="yes">
5.    cause erreur : valeur non prise en compte
6.   </xsl:message>
7. </xsl:otherwise>
```

XSLT

* Inclusion:
  * `<xsl:include href="inc.xsl"/>`
  * (équivalent à un copier-coller)

* Importation:
  * `<xsl:import href="imp.xsl"/>`
  * (priorité des templates inférieur au templates redéfinit)