

XSL

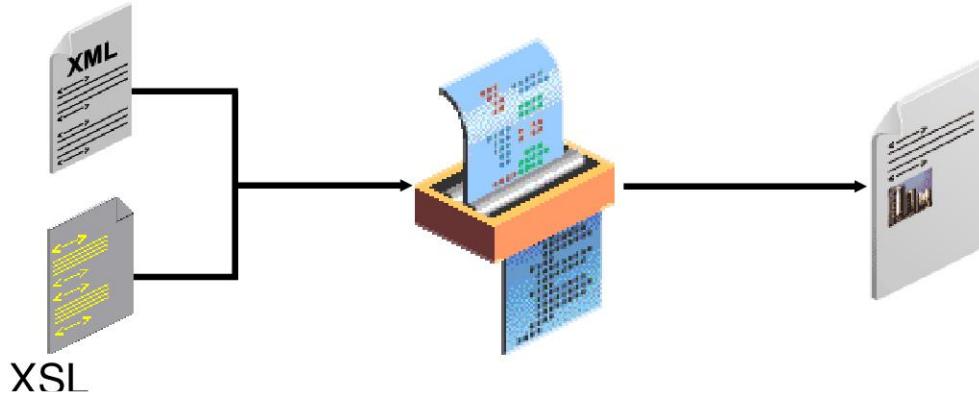
# EXtensible Stylesheet Language

## □ Qu'est-ce que XSL ?

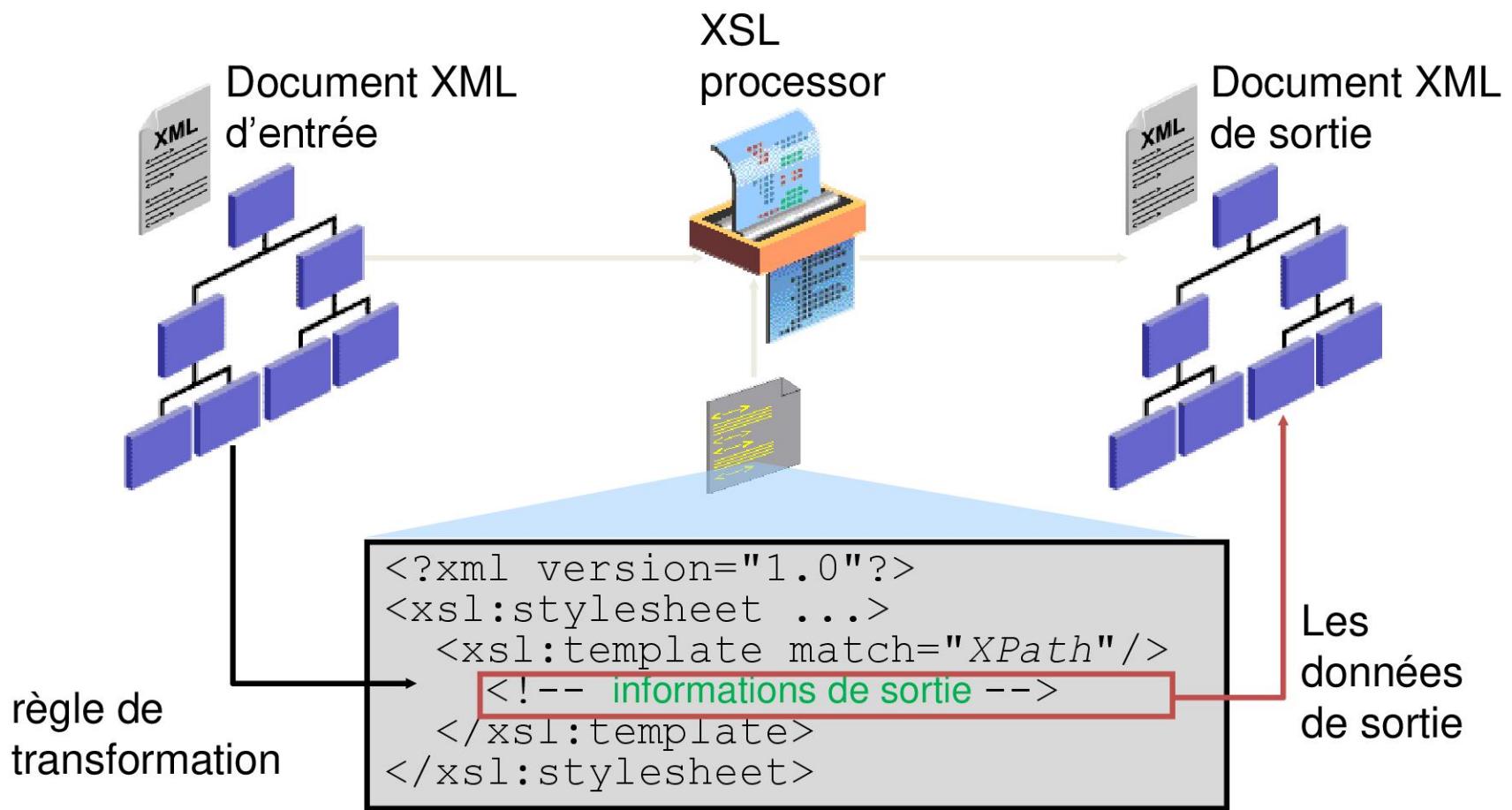
C'est un langage déclaratif (appelé **feuille de styles**).

Il est constitué de deux parties essentielles:

- XSL Transformations (**XSLT**)
- XSL Formatting Objects (**XSL-FO**)



# XSL Transformations



# XSL Transformations

## □ Definition:

Une feuille de styles XSLT est un document XML contenant:

- un élément racine **<xsl:stylesheet>** déclarant :
  - le préfixe d'espace de noms **xsl**
  - l'URI de l'espace <http://www.w3.org/1999/XSL/Transform>
- un ou plusieurs éléments **<xsl:template>** et d'autres éléments XSL définissant les règles de transformation.

# XSL Transformations

**NB:** l'indication de l'URI est **obligatoire**

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

L'exemple ci-dessous est une feuille de styles XSLT:

# XSL Transformations

```
<?xml version="1.0"?>
<xsl:stylesheet version ="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<table border="1">
<tr><th>Id</th><th>Name</th><th>Salary</th></tr>
<xsl:apply-templates/>-----<div style="border: 1px dashed black; padding: 5px; width: fit-content; margin-left: auto; margin-right: auto; background-color: #f0f0f0; border-radius: 10px; display: inline-block; text-align: center; font-size: 1em; font-weight: bold; color: black; text-decoration: none; text-underline-style: none; text-decoration-color: black; text-decoration-thickness: 1px; text-decoration-style: solid; text-decoration-width: 1px; text-decoration-position: under;">5
```

1

2

3

4

6

# XSL Transformations

## □ Utilisation d'une feuille de styles XSLT:

Un document XML utilise une instruction de traitement

<?xml-stylesheet ... ?>:

- après la déclaration XML et avant l'élément racine du document XML.
- contenant deux pseudo-attributs:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="employees.xsl"?>
<employees>
    ...
</employees>
```

# XSL Transformations

## □ Création des règles de transformation

Une règle de transformation (« **template** ») est:

- créée en utilisant des éléments **<xsl:template>** avec:
  - un attribut **match**
  - le modèle de sortie
- appliquée quand un nœud dans le document XML en entrée correspond à l'expression XPath dans la règle

```
<xsl:template match="XPath">  
    sortie-template  
</xsl:template>
```

**NB:** **match** détermine son nœud de contexte

# XSL Transformations

## □ L'élément <xsl:value-of>:

- possède l'attribut **select** contenant une expression **XPath**
- insère la valeur de l'expression **XPath**
- est utilisé dans un élément <xsl:template>

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

```
<region num="1">
    <id>1</id>
    <name>Europe</name>
</region>
```

Document XML

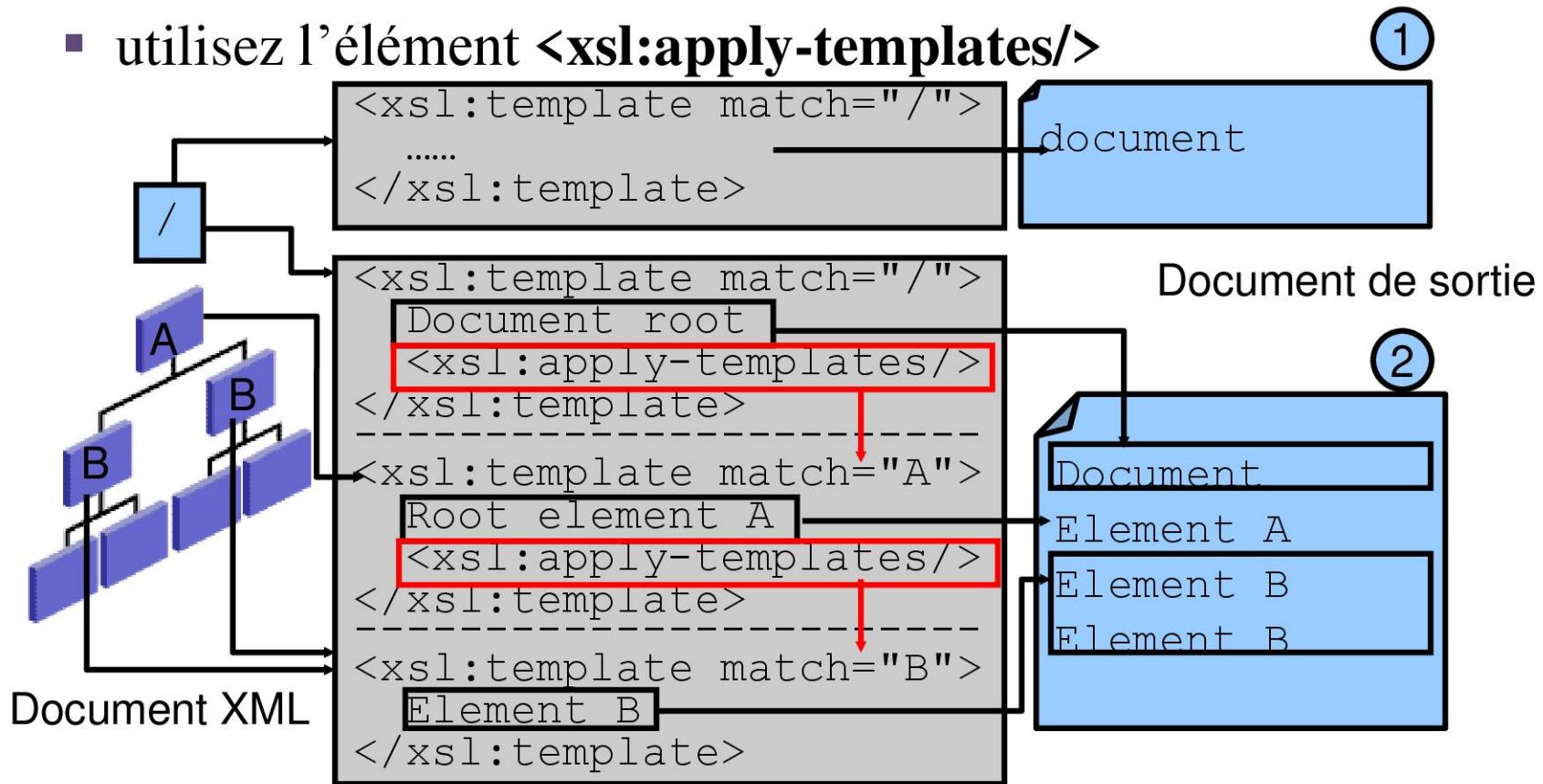
1 Europe

Document de sortie

# XSL Transformations

## □ Utilisation des règles de transformation

- utilisez l'élément **<xsl:apply-templates/>**



# XSL Transformations

## □ Utilisation des règles de transformation

- utilisez l'attribut **select** pour sélectionner la règle

```
<?xml version='1.0' encoding='utf-8'?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <xsl:apply-templates select="//region"/>
  </xsl:template>
  <xsl:template match="regions">
    <h1>Regions</h1><xsl:apply-templates/>
  </xsl:template>
  <xsl:template match="region">
    <p><xsl:value-of select="." /></p>
  </xsl:template>
</xsl:stylesheet>
```

# XSL Transformations

## □ L'élément <xsl:for-each> :

- Il permet de parcourir une liste de nœuds et leur appliquer une transformation.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="departments">
    <xsl:for-each select="department">
      <p><xsl:value-of select="."/></p>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

# XSL Transformations

## □ L'élément **<xsl:output>**:

- Il spécifie le format de sortie
- Il doit être un enfant de l'élément **<xsl:stylesheet>**
- Il possède des attributs notamment
  - **method** avec comme valeur (**xml**, **html**, **text**)
  - **encoding**
  - **indent** avec comme valeur **yes**

# XSL Transformations

## □ Remarque

- une expression **XPath**, entre accolades, peut être utilisé comme valeur d'attribut

```
<region>
  <region_id>1</region_id>
  <region_name>Europe</region_name>
</region>
```

```
<xsl:template match="/">
  <regions><xsl:apply-templates/></regions>
</xsl:template>
<xsl:template match="region">
  <region id="{region_id}" name="{region_name}" />
</xsl:template>
```

```
<regions><region id="1" name="Europe"/></regions>
```

# XSL Transformations

## □ Crédit d'éléments avec des attributs

- pour créer un élément, utilisez **<xsl:element>**:

```
<xsl:element name="region">Japan</xsl:element>  
          <region>Japan</region>
```

1

- pour créer un attribut, utilisez **<xsl:attribute>**:

✓ à l'intérieur d'un **<xsl:element>**:

```
<xsl:element name="region">  
  <xsl:attribute name="id">5</xsl:attribute>Japan  
</xsl:element>  
          <region id="5">Japan</region>
```

2

# XSL Transformations

## □ Crédit d'éléments avec des attributs

✓ à l'intérieur de l'élément **<xsl:element>** utilisez  
l'attribut **use-attribute-set** de **<xsl:element>**:

```
<xsl:attribute-set name="region-info">
  <xsl:attribute name="id">5</xsl:attribute>
  <xsl:attribute name="name">Japan</xsl:attribute>
</xsl:attribute-set>
<xsl:template match="region[1]">
  <xsl:element name="{local-name()}">
    use-attribute-sets="region-info">
  </xsl:element>
</xsl:template>          <region id="5" name="Japan"/>
```

3

# XSL Transformations

## □ L'élément <xsl:sort>:

- Il est utilisé pour trier les nœuds dans le document d'entrée avant que les nœuds soient traités, à l'intérieur des éléments

## <xsl:apply-templates> ou <xsl:for-each>

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/employees">
  <xsl:for-each select="employee">
    <xsl:sort select="last_name"/>
    <p><xsl:value-of select="."/></p>
  </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

# XSL Transformations

## □ L'élément <xsl:sort>:

- Il possède des attributs comme:
  - **data-type="text | number"**
  - **order="ascending | descending"**
  - **case-order="upper-first | lower-first"**

# XSL Transformations

## □ L'élément <xsl:if>:

- Il est traité lorsque l'attribut **test** est évalué à vrai.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <h2>Les ministères avec un gestionnaire </h2>
    <xsl:for-each select="//department">
      <xsl:if test="manager_id">
        <p><xsl:value-of select="." /></p>
      </xsl:if>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

# XSL Transformations

## □ L'élément **<xsl:choose>**:

- Il comporte un ou plusieurs éléments **<xsl:when>** et éventuellement un élément **<xsl:otherwise>**
- l'élément **<xsl:when>** permet de tester si une condition est réalisée.
- on a un nombre illimité d'éléments **<xsl:when>** et un élément **<xsl:otherwise>**.

# XSL Transformations

## □ L'élément <xsl:choose>:

```
<xsl:template match="/">
  <xsl:for-each select="//employee">
    <p><xsl:value-of select="last_name"/>,
      <xsl:choose>
        <xsl:when test="salary < 10000">
          <font color="red">
            <xsl:value-of select="salary"/>
          </font>
        </xsl:when>
        <xsl:otherwise>
          <font color="blue">
            <xsl:value-of select="salary"/>
          </font>
        </xsl:otherwise>
      </xsl:choose></p>
    </xsl:for-each>
  </xsl:template>
```

# XSL Transformations

## □ La notion de mode:

- définir le mode dans `<xsl:template>`

```
<xsl:template match="department" mode="toc"> ...
<xsl:template match="department" mode="body"> ...
<xsl:template match="department "> ...
```

- le référencer dans `<xsl:apply-templates>`:

```
<xsl:apply-templates
    select="department" mode="toc"/>
```

# XSL Transformations

## Exemple d'utilisation des modes

```
...
<xsl:template match="/departments">
  <html>
    <body>
      <h1>Rapport du département</h1>
      <ol><xsl:apply-templates
            select="department" mode="toc"/></ol>
      <xsl:apply-templates
            select="department" mode="body"/>
    </body>
  </html>
</xsl:template>
<xsl:template match="department" mode="toc">
  <a href="#id_{department id}">
    <li><xsl:value-of select="department_name"/></li>
  </a>
</xsl:template>
<xsl:template match="department" mode="body">
  <a name="#id_{department id}">
    <h2><xsl:value-of select="department name"/></h2>
  </a>...<xsl:apply-templates mode="body"/>...
</xsl:template>
```

# XSL Transformations

## □ Appel d'un template par son nom:

Créer un template en spécifiant son nom par l'attribut name

```
...
<xsl:template name="deptlist">
    <!-- instructions à traiter -->
</xsl:template>
...
```

Appeler le template par son nom

```
...
<xsl:template match="/departments">
    <html>
        <body>
            <h1>Rapport du département</h1>
            <ol><xsl:call-template name="deptlist"/></ol>
            <xsl:apply-templates mode="body"/>
        </body>
    ...

```

# XSL Transformations

## □ Création et utilisation des paramètres

Définir les paramètres du template avec **<xsl:param>**:

```
<xsl:template name="deptlist">
    <xsl:param name="loc_id"/> -----+
    <xsl:for-each
        select="//department [location_id=$loc_id]">
        <a href="#id_{department_id}><li><xsl:value-of
            select="department_name"/></li></a>
    </xsl:for-each>
</xsl:template>
```

Passer les paramètres avec **<xsl:with-param>** dans un  
**<xsl:apply-templates>** ou **<xsl:call-template>**:

```
<xsl:template match="/departments">
    <ol><xsl:call-template name="deptlist">
        <xsl:with-param name="loc id"
            select="department[1]/@location_id"/>
    </xsl:call-template></ol>
</xsl:template>
```

# XSL Transformations

## □Création et utilisation des variables

- Ils sont défini avec l'élément **<xsl:variable>** tout comme les parametres.
- l'appel des variables dans les attributs de balises se fait en mettant un \$ devant le nom.
- Il est aussi possible de les appeler à l'extérieur en l'enfermant dans des accolades : **{\$phrase}**.

# XSL Transformations

## □ Remarque:

Un template peut se définir dans un autre document en faisant appel aux balises **<xsl:include>** et **<xsl:import>**