NOM : Khedim **PRENOM :** Youcef **GROUP:** 5

XSLT Template, `apply-templates`, and `call-template` - A Summary

1. **Template (`<xsl:template>`):**
   - **general use case**
   - **Purpose:** Defines a transformation rule in XSLT. Templates match specific parts of an XML document and specify how those parts should be processed.
   - **Example:**

     ```
     <xsl:template match="book">
         <p><xsl:value-of select="title"/></p>
     </xsl:template>
     ```

     This template matches `<book>` elements and outputs their `<title>` in a paragraph.

2. **`xsl:apply-templates`:**
   - **Purpose:** This element is used to process child nodes of the current node. It applies templates to those child nodes. It is commonly used to apply templates to specific sets of nodes.
   - **Example:**

     ```
     <xsl:template match="library">
         <xsl:apply-templates select="book"/>
     </xsl:template>
     ```

     This applies the templates for `<book>` elements inside the `<library>` element. It processes each `<book>` according to the templates defined for it.

3. **`xsl:call-template`:**
   - **Purpose:** This explicitly calls a named template to apply it at a specific point in the XSLT transformation. It is useful when you want to reuse templates or perform specific processing in a controlled manner.
   - **Example:**

     ```
     <xsl:template name="showBook">
         <p><xsl:value-of select="title"/></p>
     </xsl:template>

     <xsl:template match="library">
         <xsl:call-template name="showBook"/>
     </xsl:template>
     ```

Here, the `showBook` template is called from within another template (like `library`), allowing reuse of the logic to output the book's title.

Key Differences:

- **`xsl:apply-templates`** is used for applying templates to child nodes of the current context (recursive processing).
- **`xsl:call-template`** is used to invoke a **named** template explicitly from anywhere in the stylesheet.

Example in Context:

```
<xsl:template match="library">
    <h2>Library Books:</h2>
    <xsl:apply-templates select="book"/>
</xsl:template>

<xsl:template match="book">
    <div>
        <xsl:value-of select="title"/>
        <xsl:value-of select="author"/>
    </div>
</xsl:template>


### XSLT Template, `apply-templates`, and `call-template` - Using the
exercice number 3 in the technical sheet 5 (TD5)

1. **Template (`<xsl:template>`):**
    - **specific use case **

   - **Purpose:** Defines a transformation rule in XSLT. Templates match
specific parts of an XML document and specify how those parts should be
processed.
   - **Example from exercise:**
   - for example we can use template to define custom functions even
recursive ones as in this example exercice
     ```xslt
     <xsl:template match="/concours">
         <!-- Generates the main HTML output -->
         <html>
             <head>
                 <title>Rssultats des Candidats</title>
             </head>
             <body>
                 <h1>Resultats des Candidats</h1>
                 <!-- More content here -->
             </body>
         </html>
     </xsl:template>
     ```
     This template matches the root `<concours>` and generates an HTML
structure.
```

````
2. **`xsl:apply-templates`:**
   - **Purpose:** Processes child nodes of the current node. It applies
templates to those child nodes, allowing recursive handling of XML
structures.
   - **Example:**
     ```xslt
     <xsl:template match="/concours">
         <body>
             <xsl:for-each select="grade">
                 <!-- Instead of for-each, you can also use apply-templates
here -->
                 <xsl:apply-templates select="etablissement"/>
             </xsl:for-each>
         </body>
     </xsl:template>
     ```
     This would apply the template that matches `<etablissement>` elements
under `<grade>`.

3. **`xsl:call-template`:**
   - **Purpose:** Explicitly calls a named template, often used for
functions or reusable operations.
   - **Example from exercise:**
     ```xslt
     <xsl:call-template name="sumCoefficients">
         <xsl:with-param name="nodes" select="matieres/matiere"/>
     </xsl:call-template>
     ```
     This explicitly calls the `sumCoefficients` template, passing a
parameter to perform recursive summing of coefficients.

### **Key Differences:**
- **`xsl:apply-templates`** is for matching and processing XML nodes based
on matching templates.
- **`xsl:call-template`** is used for calling predefined logic (like a
function), not based on matching.

### **Contextual Example from the Exercise:**
```xslt
<xsl:template name="sumCoefficients">
    <xsl:param name="nodes"/>
    <xsl:param name="sum" select="0"/>
    <xsl:choose>
        <xsl:when test="count($nodes) > 0">
            <xsl:variable name="first" select="$nodes[1]"/>
            <xsl:variable name="rest" select="$nodes[position() > 1]"/>
            <xsl:call-template name="sumCoefficients">
                <xsl:with-param name="nodes" select="$rest"/>
                <xsl:with-param name="sum" select="$sum +
number($first/@coefficient)"/>
            </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
````
````

```
            <xsl:value-of select="$sum"/>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
```

This is a recursive template function that calculates the total coefficient of all `<matiere>` nodes passed in as a parameter.

XSLT Basic Functions with Examples

1. **Selection and Extraction of Data**
   - **Function:** Selects specific elements or attributes from the XML to use in the output.
   - **Example:**

     ```
     <person>
         <name>John</name>
         <age>30</age>
     </person>
     ```

     ```
     <xsl:value-of select="person/name"/>
     ```

     Result: `John`

2. **Generation of Content**
   - **Function:** Generates new content in the output that doesn't exist in the original XML.
   - **Example:**

     ```
     <xsl:template match="/">
         <message>Hello, <xsl:value-of select="person/name"/>!
     </message>
     </xsl:template>
     ```

     Result:

     ```
     <message>Hello, John!</message>
     ```

3. **Deletion of Content**
   - **Function:** Removes elements from the output by not including them in the template.
   - **Example:**

```
<xsl:template match="age"/>
```

This will omit the `<age>` element from the output.

4. **Movement of Content**
   - **Function:** Moves content from one structure to another in the output.
   - **Example:**

```
<person>
    <name>John</name>
    <age>30</age>
</person>
```

```
<xsl:template match="person">
    <full-name>
        <xsl:value-of select="name"/>
    </full-name>
</xsl:template>
```

Result:

```
<full-name>John</full-name>
```

5. **Duplication of Content**
   - **Function:** Outputs the same content multiple times.
   - **Example:**

```
<xsl:template match="/">
    <xsl:value-of select="person/name"/>
    <xsl:value-of select="person/name"/>
</xsl:template>
```

Result: `JohnJohn`

6. **Data Filtering**
   - **Function:** Outputs only elements that match certain conditions.
   - **Example:**

```

```

```
<people>
    <person>
        <name>John</name>
        <age>30</age>
    </person>
    <person>
        <name>Jane</name>
        <age>17</age>
    </person>
</people>
```

```
<xsl:for-each select="people/person[age > 18]">
    <xsl:value-of select="name"/>
</xsl:for-each>
```

Result: `John`

7. **Sorting of Data**

   - **Function:** Sorts data in ascending or descending order.
   - **Example:**

```
<people>
    <person>
        <name>John</name>
        <age>30</age>
    </person>
    <person>
        <name>Jane</name>
        <age>17</age>
    </person>
</people>
```

```
<xsl:for-each select="people/person">
    <xsl:sort select="age" order="ascending"/>
    <xsl:value-of select="name"/>
</xsl:for-each>
```

Result: `JaneJohn`

8. **Other Functions (Etc.)**

   - **Conditional Processing:** Use of `<xsl:if>`, `<xsl:choose>`, etc.
   - **Template Reuse and Inheritance:** Using `<xsl:apply-templates>` and named templates.
   - **Variables and Parameters:** Use of `<xsl:variable>` and `<xsl:param>`.