Урок 6

План заняття:

1. Модульна організація приведення XML документа в інший формат

1.1. Введення

Як і в будь-якій мові програмування, мова XSLT дозволяє розділити код таблиці стилів на кілька документів XSLT, тобто організувати модульну структуру. Існує **два основних способи** організації модульної структури, тобто підключення інших модулів до поточного XSLT документу:

1. шляхом включення документа, яке здійснюється за допомогою елемента **xsl:include**;
2. шляхом імпортування документа за допомогою елемента **xsl:import**.

Ці елементи являються елементами верхнього рівня, тому їх оголошення повинно здійснюватись на самому початку опису XSLT документа: перед всіма іншими шаблонами і перед елементом xsl:output. Та в виняткових ситуаціях елемент xsl:include може бути також розміщений і в інших місцях таблиці стилів, на відміну від xsl:import.

На місці їх виклику, вміст включаємого або імпортованого файла підставляється повністю. **Основна відмінність** між цими елементами полягає в тому, що послідовність імпортування зовнішніх модулів (порядок імпорта) впливає на пріоритет виконання шаблонів, їх оголошення та всього, що з цим пов'язано. Отже, елемент xsl:import являється більш складнішим по своїй суті і по роботі, ніж елемент xsl:include, який здійснює просте включення одного документа в інший.

Розглянемо особливості використання всіх цих способів організації модульної структури трансформації більш детальніше.

1.2. Елемент xsl:include

Елемент **xsl:include** використовується для включення документів трансформації в інші документи XSLT. По своїй суті, він дуже схожий з директивої **#include** в мові C/C++. Але на відміну від останньої, елемент xsl:include не підтримує умовне підключення XSLT-документів.

Синтаксис даного елемента наступний:

```
<xsl:include href="URL_включаємого_файла" />
```

Єдиний і обов'язковий атрибут даного елемента містить URL зовнішнього включаємого документа XSLT. Використання відносних шляхів також допускається.

Принцип роботи даного елемента наступний: процесор XSLT знаходить таблицю стилів по вказаній адресі і підставляє її на місці елемента xsl:include перед безпосередньою трасформацією. Але варто відмітити, що у випадку прямого або неявного включення XSLT документа самого в себе викличе помилку, точніше кажучи призведе до безкінечного циклу включень.

Крім того, слід зважати, що повторне включення одного і того ж XSLT документу також може згенерувати помилку через повторне оголошення одних і тих же елементів.

Розглянемо для початку простий приклад використання даного елемента. Припустимо, в нас існує дві таблиці стилів: simple.xslt та main.xslt. В першому документі міститься оголошення змінної color, яке необхідно включити в документ main.xslt.

simple.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:variable name="color" select="red" />
</xsl:stylesheet>
```

main.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:include href="simple.xslt" />
      <xsl:template match="/">
            <xsl:text>Color is </xsl:text>
            <xsl:value-of select="$color" />
```

```
      </xsl:template>
</xsl:stylesheet>
```

Основна таблиця стилів в результаті такого включення набуде наступного вигляду:

main.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:variable name="color" select="red" />

      <xsl:template match="/">
            <xsl:text>Color is </xsl:text>
            <xsl:value-of select="$color" />
      </xsl:template>
</xsl:stylesheet>
```

Доречі, всі посилання і відносні адреси, які використовуються у включеному документі (simple.xslt), будуть обраховуватись відносно його базової адреси.

Елемент xsl:include можна використовувати і для включення шаблонів трансформації з спрощеним синтаксисом. Такі шаблони будуть включатись і вважатись шаблонами, які стосуються кореневого вузла. Наступний приклад демонструє винесення окремого спрощеного шаблону трансформації в інший документ:

simple.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:apply-templates />
</xsl:stylesheet>
```

main.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:include href="simple.xslt" />

      <xsl:template match="root">
            <xsl:text>Hello students!!!</xsl:text>
      </xsl:template>
</xsl:stylesheet>
```

Основна таблиця стилів тепер буде виглядати так:

main.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:template match="/">
            <html>
                  <xsl:apply-templates />
            </html>
      </xsl:template>

      <xsl:template match="root">
            <xsl:text>Hello students!!!</xsl:text>
      </xsl:template>
</xsl:stylesheet>
```

Варто відмітити, що різні процесори XSLT по-різному обробляють включення спрощених шаблонів. Крім того, не всі з них взагалі таке включення підтримують, хоча згідно специфікації такі включення дозволені. В зв'язку з цим, включення спрощених шаблонів варто уникати.

Ну і насамкінець більш складніший та повніший приклад. Припустимо, що в нас існує XML-документ, для якого необхідно написати таблицю стилів. Для демонстрації включення документів, розіб'ємо необхідну таблицю стилів на два XSLT документи.

books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
```

```
  <book>
     <title>Lover Birds</title>
     <author>Cynthia Randall</author>
     <price>151,20</price>
  </book>
  <book>
     <title>The Sundered Grail</title>
     <author>Eva Corets</author>
     <price>254,20</price>
  </book>
  <book>
     <title>Splish Splash</title>
     <author>Paula Thurman</author>
     <price>450,89</price>
  </book>
</books>
```

simple.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:template match="title">
        Title: <xsl:value-of select="."/><br />
      </xsl:template>

      <xsl:template match="author">
        Author: <xsl:value-of select="."/><br />
      </xsl:template>

      <xsl:template match="price">
        Price: <xsl:value-of select="."/><br />
      </xsl:template>

</xsl:stylesheet>
```

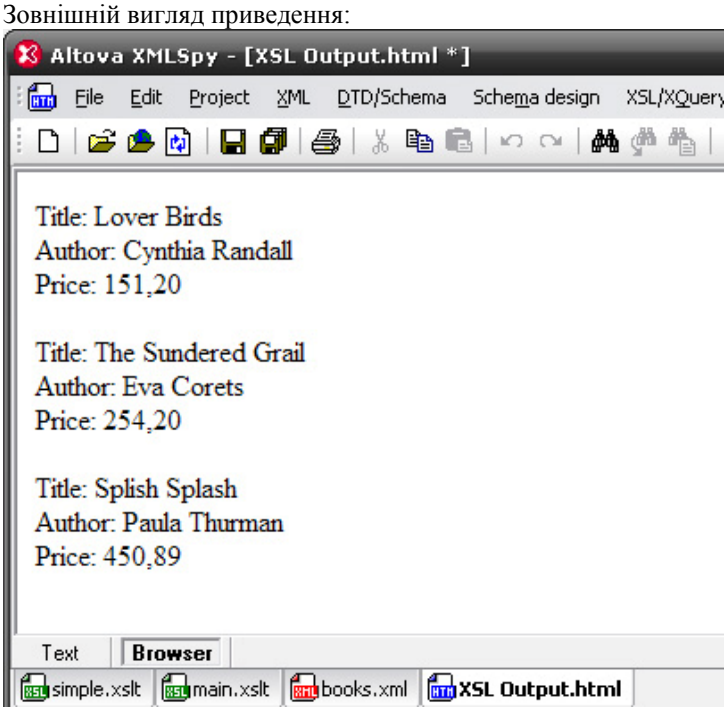
main.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="xml" version="1.0" omit-xml-declaration="yes" />

      <xsl:template match="/">
        <xsl:for-each select="books/book">
             <xsl:apply-templates select="title" />
             <xsl:apply-templates select="author" />
             <xsl:apply-templates select="price" />
             <br />
        </xsl:for-each>
      </xsl:template>

      <xsl:include href="simple.xslt" />
</xsl:stylesheet>
```

Результатом трансформації беде новий XML-документ:

```
Title: Lover Birds<br/>
Author: Cynthia Randall<br/>
Price: 151,20<br/><br/>
Title: The Sundered Grail<br/>
Author: Eva Corets<br/>
Price: 254,20<br/><br/>
Title: Splish Splash<br/>
Author: Paula Thurman<br/>
Price: 450,89<br/><br/>
```

Зовнішній вигляд приведення:



1.3. Елемент xsl:import

Елемент **xsl:import** дозволяє здійснити імпорт XSLT документів в інші. Імпорт таблиць стилів, як було вже вище сказано, являється складнішим, ніж включення, оскільки послідовність імпортування зовнішніх модулів (порядок імпорту) впливає на пріоритет виконання та оголошення шаблонів. При цьому,.

Синтаксис і принцип дії елемента xsl:import аналогічний вищеописаному елементу xsl:include:

```
<xsl:import href="URL_імпортуємого_файла" />
```

Не варто забувати, що елементи xsl:import повинні завжди бути першими дочірними елементами, тобто оголошуватись раніше всіх шаблонів і навіть елемента xsl:output.

Правила, по яким здійснюється імпортування зовнішніх модулів та зміна пріоритетів виконання шаблонів:

1. Порядок імпорту основного XSLT документа завжди вищий порядку імпорта зовнішнього документу, тобто описаних в ньому шаблонів.
2. У випадку, якщо імпортується кілька зовнішніх документів, порядок імпорту шаблонів, які імпортуються раніше, молодше порядку імпорта шаблонів документів, які імпортуються наступними. Як наслідок, якщо шаблонів в результаті імпорту на певний елемент виявилось кілька, то застосовуватись будуть правила, які імпортовані останніми.
3. Порядок імпорту шаблонів, які включені в основний, рівний порядку імпорту основного документа.

Для демонстрації цих правил, розглянемо маленький приклад. Припустимо, існує три XSLT документа (part1.xslt, part2.xslt та part3.xslt), які необхідно включити в основний:

main.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:import href="part1.xslt" />
      <xsl:import href="part2.xslt" />
      <xsl:import href="part3.xslt" />
      <!-- тіло таблиці стилів -->
</xsl:stylesheet>
```

В результаті такого запису, порядок імпорту основної таблиці стилів (main.xslt) вищий, ніж імпортованих. При цьому, порядок імпорту таблиці стилів part1.xslt меньший порядку part2.xslt, а шаблони таблиці стилів part3.xslt будуть мати порядок імпорту рівний документу main.xslt. Тобто порядок імпорту буде наступним:

```
part1.xslt            <!-- самий нижчий пріоритет -->
part2.xslt
main.xslt part3.xslt  <!-- найвищий пріоритет -->
```

2. XSLT трансформація з XML в XML. Основи, необхідність та приклад використання

На попередній парі, ми розібрали як здійснювати трансформацію з XML формату в HTML. Розглянули ряд елементів та прикладів використання. Наступним видом трансформації, який також вартий уваги, буде трансформація типу XML-XML. Як вже було неодноразово сказано, цей вид трансформації в основному призначений для бізнес-систем, коли на вході та на виході існують дані в форматі XML, які мають різну структуру. Наприклад, елементи на вході і на виході по різному називаються, додаються нові чи вилучаються існуючі елементи або атрибути. Оскільки на практиці зустрічаються всі ці випадки, то варто розглянути кожен з них окремо і на прикладах.

Перший приклад буде стосуватись випадку, коли на вході і на виході повинні знаходитись XML-документи, які мають ідентичну структуру, але назви елементі і атрибутів відрізняються.

Input.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<group>
	<student>Oleg Jakobchuk</student>
	<student>Dima Kuznecov</student>
</group>
```

InputToOutput.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" version="1.0" indent="yes" />

	<xsl:template match="/">
		<STUDENTS>
			<xsl:apply-templates />
		</STUDENTS>
	</xsl:template>

	<xsl:template match="student">
		<STUDENT>
			<xsl:value-of select="." />
		</STUDENT>
	</xsl:template>

</xsl:stylesheet>
```

В результаті ми отримаємо XML-документ, який має наступний вигляд:

Output.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<STUDENTS>
	<STUDENT>Oleg Jakobchuk</STUDENT>
	<STUDENT>Dima Kuznecov</STUDENT>
</STUDENTS>
```

Наступним прикладом ускладнимо дещо завдання, структура вихідного XML документа повинна трішки змінитись, тобто дані певних атрибутів повинні бути переміщені в нові елементи тощо.

Input.xml

```
<?xml version="1.0" encoding="windows-1251"?>
<persons>
 <person username="MP123456">
   <name>Иван</name>
   <surname>Иванов</surname>
 </person>
 <person username="PK123456">
   <name>Пётр</name>
   <surname>Петров</surname>
 </person>
</persons>
```

InputToOutput.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" version="1.0" encoding="windows-1251" indent="yes"/>

	<xsl:template match="/">
		<transform>
		  <xsl:apply-templates/>
		</transform>
	</xsl:template>

	<xsl:template match="person">
		<record>
			<id>
			  <xsl:value-of select="@username" />
			</id>
			<fullname>
			  <xsl:value-of select="name" /> <xsl:value-of select="surname" />
			</fullname>
		</record>
	</xsl:template>
</xsl:stylesheet>
```

Output.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<transform>
	<record>
		<id>MP123456</id>
		<fullname>Иван Иванов</fullname>
	</record>
	<record>
		<id>PK123456</id>
		<fullname>Пётр Петров</fullname>
	</record>
</transform>
```

Третій приклад продемонструє те, як додати новий елемент, наприклад, інструкцію по обробці, у вихідний документ не порушуючи при цьому основну структуру вхідного XML документа. Для демонстрації візьмемо XML документ з попереднього прикладу.

InputToOutput.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

	<xsl:template match="/">
		<xsl:processing-instruction name="xml-stylesheet">
			<xsl:text>type="text/xsl" href="style.xsl"</xsl:text>
		</xsl:processing-instruction>
		<xsl:apply-templates />
	</xsl:template>

	<xsl:template match="@*|*">
		<xsl:copy>
			<xsl:apply-templates select="@* | node()"/>
		</xsl:copy>
	</xsl:template>
</xsl:stylesheet>
```

Output.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
<persons>
```

```
      <person username="MP123456">
            <name>Иван</name>
            <surname>Иванов</surname>
      </person>
      <person username="PK123456">
            <name>Пётр</name>
            <surname>Петров</surname>
      </person>
</persons>
```

Варто відмітити, що подвійні лапки в значеннях інструкції по обробці являються обов'язковими.

Останній приклад найскладніший, оскільки в ньому нам необхідно на виході додати нові елементи та атрибути, а також вилучити зайві.

Response.xml

```
<?xml version="1.0"?>
<MyXML>
 <MyXMLMsgsRs>
   <VendorQueryRs requestID="0" statusCode="0" statusSeverity="Info"
                  statusMessage="Status OK">
     <VendorRet>
       <ItemID>680000-1046215164</ItemID>
       <TimeCreated>2003-02-26T01:19:24+02:00</TimeCreated>
       <FullName>Castuciano, James</FullName>
       <CompanyName>Castuciano, James</CompanyName>
       <Salutation>Mr</Salutation>
       <FirstName>James</FirstName>
       <LastName>Castuciano</LastName>
       <VendorAddress>
         <Addr>12 Van Horne Cres</Addr>
         <City>Toronto</City>
       </VendorAddress>
       <Phone>316-1116</Phone>
       <Fax>316-1116</Fax>
       <Contact>James Castuciano</Contact>
       <TermsRef>
         <ItemID>20000-1046201791</ItemID>
         <FullName>Net 30</FullName>
       </TermsRef>
       <Balance>10,00</Balance>
     </VendorRet>

     <VendorRet>
       <ItemID>6E0000-1046215165</ItemID>
       <TimeCreated>2003-02-26T01:19:25+02:00</TimeCreated>
       <FullName>Chai, Stephen</FullName>
       <CompanyName>Stephen Chai</CompanyName>
       <Salutation>Mr</Salutation>
       <FirstName>Stephen</FirstName>
       <LastName>Chai</LastName>
       <VendorAddress>
         <Addr>1333 Hornby Street</Addr>
         <City>Middlefield</City>
       </VendorAddress>
       <Phone>555-1313</Phone>
       <Fax>555-1222</Fax>
       <Contact>Stephen Chai</Contact>
       <TermsRef>
         <ItemID>20000-1046201791</ItemID>
         <FullName>Net 30</FullName>
       </TermsRef>
       <Balance>250,00</Balance>
     </VendorRet>
    </VendorQueryRs>
```

```
   <CustomerQueryRs requestID="1" statusCode="0" statusSeverity="Info"
                    statusMessage="Status OK">
     <CustomerRet>
       <ItemID>110000-3246215164</ItemID>
       <TimeCreated>2004-02-26T01:19:24+02:00</TimeCreated>
       <FullName>Pupkin, Ivan</FullName>
       <Salutation>Mr</Salutation>
       <FirstName>Ivan</FirstName>
       <LastName>Pupkin</LastName>
       <CustomerAddress>
         <Addr>Chreschatyk, 12</Addr>
         <City>Kiev</City>
       </CustomerAddress>
       <Phone>22-22-222</Phone>
     </CustomerRet>
    </CustomerQueryRs>
 </MyXMLMsgsRs>
</MyXML>
```

Перед тим як написати таблицю стилів для даного документа слід розглянути, що нам необхідно мати на виході, тобто оцінити ситуацію. А на виході необхідно мати XML документ наступної структури:

Request.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?myxml version="2.0"?>
<MyXML>
      <MyXMLMsgsRq onError="stopOnError">
            <VendorAddRq requestID="0">
                  <VendorAdd>
                        <FullName>Castuciano, James</FullName>
                        <CompanyName>Castuciano, James</CompanyName>
                        <Salutation>Mr</Salutation>
                        <FirstName>James</FirstName>
                        <LastName>Castuciano</LastName>
                        <VendorAddress>
                              <Addr>12 Van Horne Cres</Addr>
                              <City>Toronto</City>
                        </VendorAddress>
                        <Phone>316-1116</Phone>
                        <Fax>316-1116</Fax>
                        <Contact>James Castuciano</Contact>
                        <TermsRef>
                              <FullName>Net 30</FullName>
                        </TermsRef>
                  </VendorAdd>
            </VendorAddRq>
            <VendorAddRq requestID="0">
                  <VendorAdd>
                        <FullName>Chai, Stephen</FullName>
                        <CompanyName>Stephen Chai</CompanyName>
                        <Salutation>Mr</Salutation>
                        <FirstName>Stephen</FirstName>
                        <LastName>Chai</LastName>
                        <VendorAddress>
                              <Addr>1333 Hornby Street</Addr>
                              <City>Middlefield</City>
                        </VendorAddress>
                        <Phone>555-1313</Phone>
                        <Fax>555-1222</Fax>
                        <Contact>Stephen Chai</Contact>
                        <TermsRef>
                              <FullName>Net 30</FullName>
                        </TermsRef>
                  </VendorAdd>
```

```
            </VendorAddRq>
            <CustomerAddRq requestID="1">
                  <CustomerAdd>
                        <FullName>Pupkin, Ivan</FullName>
                        <Salutation>Mr</Salutation>
                        <FirstName>Ivan</FirstName>
                        <LastName>Pupkin</LastName>
                        <CustomerAddress>
                              <Addr>Chreschatyk, 12</Addr>
                              <City>Kiev</City>
                        </CustomerAddress>
                        <Phone>22-22-222</Phone>
                  </CustomerAdd>
            </CustomerAddRq>
      </MyXMLMsgsRq>
</MyXML>
```

Отже, на виході:

1. присутня нова інструкція по обробці;
2. ряд елементів переіменовані, в основному це стосується суфікса елементів, який з Rs (response) замінений на Rq (request);
3. деякі атрибути зникли (statusCode, statusSeverity, statusMessage тощо). Це пов'язано з тим, що вони стосуються запиту, а у відповіді вони втрачають сенс. Їм на заміну приходять інші, які безпосередньо стосуються відповіді (onError);
4. структура вкладених елементів та атрибутів також відрізняється.

Зробити таку трансформацію не так то і легко. В першу чергу необхідно обов'язково створити зовнішню таблицю стилів з спрощеном шаблоном трансформації, який має найнижчий пріоритет. Призначенням цього XSLT документа буде копіювання всіх вузлів. Після цього, основна таблиця стилів буде додавати і віднімати елементи та атрибути поверх зкопійованого. Шаблон, що здійснює таке копіювання повинен обов'язково мати найнижчий пріоритет в основній таблиці стилів і щоб бути в цьому впевненим на 100%, як правило, виставляється від'ємний пріоритет.

Механізм нескладний, але вартий уваги, адже по іншому вищеописану трансформацію здійснити навряд чи вдасться.

copynode.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
   <xsl:template match="*" priority="-100">
     <xsl:copy>
        <xsl:apply-templates />
     </xsl:copy>
 </xsl:template>
</xsl:stylesheet>
```

ResponseToRequest.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="copynode.xslt" />
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes" />

<!-- стаємо в корінь документа-->
  <xsl:template match="/">
  <!-- створюємо інструкцію по обробці -->
   <xsl:processing-instruction name="myxml">
     <xsl:text>version="2.0"</xsl:text>
   </xsl:processing-instruction>

   <xsl:apply-templates /> <!-- обробляємо всі інші вузли рекурсивно  -->
 </xsl:template>

 <xsl:template match="MyXML">
   <MyXML>
     <xsl:apply-templates /><!-- обробляємо всі дочірні вузли MyXML рекурсивно  -->
   </MyXML>
 </xsl:template>
```

```
  <xsl:template match="MyXMLMsgsRs">
   <MyXMLMsgsRq>
     <xsl:attribute name="onError">stopOnError</xsl:attribute>

     <!-- <xsl:call-template name="TransformObject" /> -->

     <!-- обробляємо всі дочірні вузли MyXMLMsgsRs рекурсивно  -->
     <xsl:call-template name="TransformObject0">
       <xsl:with-param name="entryPoint" select="VendorQueryRs" />
     </xsl:call-template>

     <xsl:call-template name="TransformObject1">
       <xsl:with-param name="entryPoint" select="CustomerQueryRs" />
     </xsl:call-template>
   </MyXMLMsgsRq>
 </xsl:template>

<!-- стандартний шаблон для трансформації замовників та клієнтів
  <xsl:template name="TransformObject">
   <xsl:for-each select="VendorQueryRs//VendorRet">
     <VendorAddRq>
       <xsl:attribute name="requestID">
         <xsl:value-of select="VendorQueryRs/@requestID" />
       </xsl:attribute>
       <VendorAdd>
         <xsl:apply-templates />
       </VendorAdd>
     </VendorAddRq>
   </xsl:for-each>
 </xsl:template>  -->

 <!-- конкретизуємо зразок для кожного випадку, оскільки результат дещо відрізняється-->
 <xsl:template name="TransformObject0">
   <xsl:param name="entryPoint" />
   <xsl:for-each select="$entryPoint//VendorRet">
     <VendorAddRq>
       <xsl:attribute name="requestID">
         <xsl:value-of select="$entryPoint/@requestID" />
       </xsl:attribute>
       <VendorAdd>
         <xsl:apply-templates />
       </VendorAdd>
     </VendorAddRq>
   </xsl:for-each>
 </xsl:template>

 <xsl:template name="TransformObject1">
   <xsl:param name="entryPoint" />
   <xsl:for-each select="$entryPoint//CustomerRet">
     <CustomerAddRq>
       <xsl:attribute name="requestID">
         <xsl:value-of select="$entryPoint/@requestID" />
       </xsl:attribute>
       <CustomerAdd>
         <xsl:apply-templates />
       </CustomerAdd>
     </CustomerAddRq>
   </xsl:for-each>
 </xsl:template>

  <!—вилучаємо елементи, які нам непотрібні  -->
  <xsl:template match="TimeCreated | ItemID | VendorRet/Balance">
  </xsl:template>
```

```
</xsl:stylesheet>
```

Закоментований код являє собою шаблон, який необхідно застосувати для того, щоб здійснити трасформацію елементів, які містять основну інформацію. На основі цього зразка ми створимо окремі іменовані шаблони для всіх випадків (для елементів, які містять інформацію про постачальників і замовників: VendorXXX і CustomerXXX). Але, якщо ви розробляєте прикладний додаток або інтерфейс, через який будуть проходити XML-документи, то ці шаблони можна буде генерувати програмно.

Останній шаблон дозволяє з основного XML-документа, структура якого була скопійована зовнішньою таблицею стилів, вилучити непотрібні нам елементи.

3. Домашнє завдання

Написати таблицю стилів, яка дозволить трансформувати XML документ company.xml в XML документ вигляду companyOutput.xml.

Ці документи мають наступний вигляд:

company.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<магазин>
	<товари>
		<товар>
			<код>ТП-123</код>
			<назва>Молоко</назва>
			<повна_назва>Молоко звичайне</повна_назва>
			<категорія>
				<код>КТ-111</код>
				<назва>Молочні вироби</назва>
			</категорія>
			<виробник>
				<код>В-421</код>
				<назва>Наш молочник</назва>
			</виробник>
			<кількість одиниці_виміру="шт.">120</кількість>
			<ціна одиниці="грн.">5,20</ціна>
		</товар>
		<товар>
			<код>ТП-124</код>
			<назва>Цукерки</назва>
			<повна_назва>Цукерки "Радість програміста"</повна_назва>
			<категорія>
				<код>КТ-112</код>
				<назва>Кондитерські вироби</назва>
			</категорія>
			<виробник>
				<код>В-789</код>
				<назва>Світоч</назва>
			</виробник>
			<кількість одиниці_виміру="кг">7</кількість>
			<ціна одиниці="грн.">1,50</ціна>
		</товар>
	</товари>
</магазин>
```

companyOutput.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?magazine version="1.1" parser="SOAP"?>
<magazine>
	<!-- опис продуктів -->
	<products>
		<product>
			<fullname>Молоко звичайне</fullname>
			<category>
				<name>Молочні вироби</name>
```

```
                </category>
                <producer>
                      <name>Наш молочник</name>
                </producer>
                <quantity measurement="шт.">120</quantity>
                <price unit="грн.">5,20</price>
          </product>
          <product>
                <fullname>Цукерки "Радість програміста"</fullname>
                <category>
                      <name>Кондитерські вироби</name>
                </category>
                <producer>
                      <name>Світоч</name>
                </producer>
                <quantity measurement="кг">7</quantity>
                <price unit="грн.">1,50</price>
          </product>
     </products>
</magazine>
```