Для запуска приложения разбора документа с помощью StAX ниже приведен достаточно простой код:

```
/* пример # 13 : запуск приложения : StreamOutputExample.java*/
package chapt16;
import java.io.FileInputStream;
import java.io.InputStream;

public class StreamOutputExample {
  public static void main(String[] args) throws Exception {
      ProductParser parser = new StAXProductParser();
      // создание входного потока данных из xml-файла
      InputStream input =
            new FileInputStream("chapt16\\mediatech.xml");
      // разбор файла с выводом результата на консоль
      parser.parse(input);
  }
}
```

XSL

Документ XML используется для представления информации в виде некоторой структуры, но он никоим образом не указывает, как его отображать. Для того чтобы просмотреть XML-документ, нужно его каким-то образом отформатировать. Инструкции форматирования XML-документов формируются в так называемые таблицы стилей, и для просмотра документа нужно обработать XML-файл согласно этим инструкциям.

Существует два стандарта стилевых таблиц, опубликованных W3C. Это CSS (Cascading Stylesheet) и XSL (XML Stylesheet Language).

CSS изначально разрабатывался для HTML и представляет из себя набор инструкций, которые указывают браузеру, какой шрифт, размер, цвет использовать для отображения элементов HTML-документа.

XSL более современен, чем CSS, потому что используется для преобразования XML-документа перед отображением. Так, используя XSL, можно построить оглавление для XML-документа, представляющего книгу.

Вообще XSL можно разделить на три части: XSLT (XSL Transformation), XPath и XSLFO (XSL Formatting Objects).

XSL Processor необходим для преобразования XML-документа согласно инструкциям, находящимся в файле таблицы стилей.

XSLT

Этот язык для описания преобразований XML-документа применяется не только для приведения XML-документов к некоторому “читаемому” виду, но и для изменения структуры XML-документа.

К примеру, XSLT можно использовать для:

- удаления существующих или добавления новых элементов в XML-документ;
- создания нового XML-документа на основании заданного;
- извлечения информации из XML-документа с разной степенью детализации;

- преобразования XML-документа в документ HTML или документ другого типа.

Пусть требуется построить новый XML-файл на основе файла **students.xml**, у которого будет удален атрибут **login**. Элементы **country**, **city**, **street** станут атрибутами элемента **address** и элемент **telephone** станет дочерним элементом элемента **address**. Следует воспользоваться XSLT для решения данной задачи. В следующем коде приведено содержимое файла таблицы стилей **students.xsl**, решающее поставленную задачу.

```
<?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" />

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

   <xsl:template match="student">
      <xsl:element name="student">
         <xsl:attribute name="faculty">
            <xsl:value-of select="@faculty"/>
         </xsl:attribute>
         <name><xsl:value-of select="name"/></name>
         <xsl:element name="address">
            <xsl:attribute name="country">
               <xsl:value-of select="address/country"/>
            </xsl:attribute>
            <xsl:attribute name="city">
               <xsl:value-of select="address/city"/>
            </xsl:attribute>
            <xsl:attribute name="street">
               <xsl:value-of select="address/street"/>
            </xsl:attribute>
            <xsl:element name="telephone">
               <xsl:attribute name="number">
                  <xsl:value-of select="telephone"/>
               </xsl:attribute>
            </xsl:element>
         </xsl:element>
      </xsl:element>
   </xsl:template>
</xsl:stylesheet>
```

Преобразование XSL лучше сделать более коротким, используя ATV (attribute template value), т.е «**{}**»

```
<?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" />
   <xsl:template match="/">
      <students>
         <xsl:apply-templates />
</students>
   </xsl:template>
   <xsl:template match="student">
<student faculty="{@faculty}">
         <name><xsl:value-of select="name"/></name>
      <address country="{address/country}"
               city="{address/city}"
               street="{address/street}">
               <telephone number="{telephone}"/>
      </address>
</student>
   </xsl:template>
</xsl:stylesheet>
```

Для трансформации одного документа в другой можно использовать, например, следующий код.

```
/* пример # 14 : трансформация XML : SimpleTransform.java */
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

public class SimpleTransform {
   public static void main(String[] args) {
      try {
         TransformerFactory tf =
            TransformerFactory.newInstance();

         //установка используемого XSL-преобразования
         Transformer transformer =
tf.newTransformer(new StreamSource("students.xsl"));

         //установка исходного XML-документа и конечного XML-файла
         transformer.transform(
            new StreamSource("students.xml"),
               new StreamResult("newstudents.xml"));

            System.out.print("complete");
      } catch(TransformerException e) {
         e.printStackTrace();
      }
   }
}
```

В результате получится XML-документ **newstudents.xml** следующего вида:

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
   <student faculty="mmf">
      <name>Mitar Alex</name>
      <address country="Belarus" city="Minsk"
            street="Kalinovsky 45">
         <telephone number="3462356"/>
      </address>
   </student>
   <student faculty="mmf">
      <name>Pashkun Alex</name>
      <address country="Belarus" city="Brest"
            street="Knorina 56">
         <telephone number="4582356"/>
      </address>
   </student>
</students>
```

Элементы таблицы стилей

Таблица стилей представляет собой well-formed XML-документ. Эта таблица описывает изначальный документ, конечный документ и то, как трансформировать один документ в другой.

Какие же элементы используются в данном листинге?

```
<xsl:output method="xml" indent="yes"/>
```

Данная инструкция говорит о том, что конечный документ, который получится после преобразования, будет являться XML-документом.

```
<xsl:template match="student">
   <lastname>
      <xsl:apply-templates/>
   </lastname>
</xsl:template>
```

Инструкция **<xsl:template...>** задает шаблон преобразования. Набор шаблонов преобразования составляет основную часть таблицы стилей. В предыдущем примере приводится шаблон, который преобразует элемент **student** в элемент **lastname**.

Шаблон состоит из двух частей:

1. параметр **match**, который задает элемент или множество элементов в исходном дереве, где будет применяться данный шаблон;
2. содержимое шаблона, которое будет вставлено в конечный документ.

Нужно отметить, что содержимое параметра **math** может быть довольно сложным. В предыдущем примере просто ограничились именем элемента. Но, к примеру, следующее содержимое параметра **math** указывает на то, что шаблон должен применяться к элементу **url**, содержащему атрибут **protocol** со значением **mailto**:

```
<xsl:template match="url[@protocol='mailto']">
```