XSLT

- Назначение
- Основные элементы
- Типы XSLT преобразований
- Использование переменных и параметров
- Сортировка
- Автоматическая нумерация
- Ключи и группировка
- Пространста имен

Что такое XSLT?

XSLT (eXtensible Stylesheet Language: Transformations) – это язык для преобразования структуры XML документа.

Стандарт XSL и XSLT

XSL состоит из:

- **XSLT – язык трансформации документов**
- **XPath – язык навигации по документу**
- **XSL-FO (Formatting Objects) – язык форматирования документов**

*** XSL версия 1.0 - W3C Recommendation 16 November 1999**

XSLT - язык программирования

Язык программирования для обработки XML данных и преобразования XML документов:

- **Набор поддерживаемых типов данных:**
  - **boolean, number, string, node set and external objects**
- **Набор операций:**
  - **<xsl:apply-templates>, <xsl:sort>, <xsl:output> и другие**
- **Команды управления:**
  - **<xsl:if>, <xsl:for-each>, <xsl:choose> и другие**

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

XSLT - язык программирования

- Использование языка XPath
- Использование встроенных функций и custom extension функций:
  - для построения запросов и преобразование XML данных
- XSLT документ – XML документ
  - XSLT инструкции – это XML элементы
  - XSLT таблицу стилей можно применять к XSLT документу и даже самому себе

Hello, World! (XSLT) исходный XML документ

XML (hello.xml)

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

<hello-world>
  <greeter>An XSLT Programmer</greeter>
   <greeting>Hello, World!</greeting>
</hello-world>
```

Hello, World! (XSLT) таблица стилей

XSLT (hello.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="1.0">
<xsl:template match="/hello-world">
        <HTML>
        <BODY>
                <H1><xsl:value-of select="greeting"/></H1>
                <xsl:apply-templates select="greeter"/>
        </BODY>
        </HTML>
</xsl:template>
<xsl:template match="greeter">
        <DIV>from
                <I><xsl:value-of select="."/></I>
        </DIV>
</xsl:template>
</xsl:stylesheet>
```

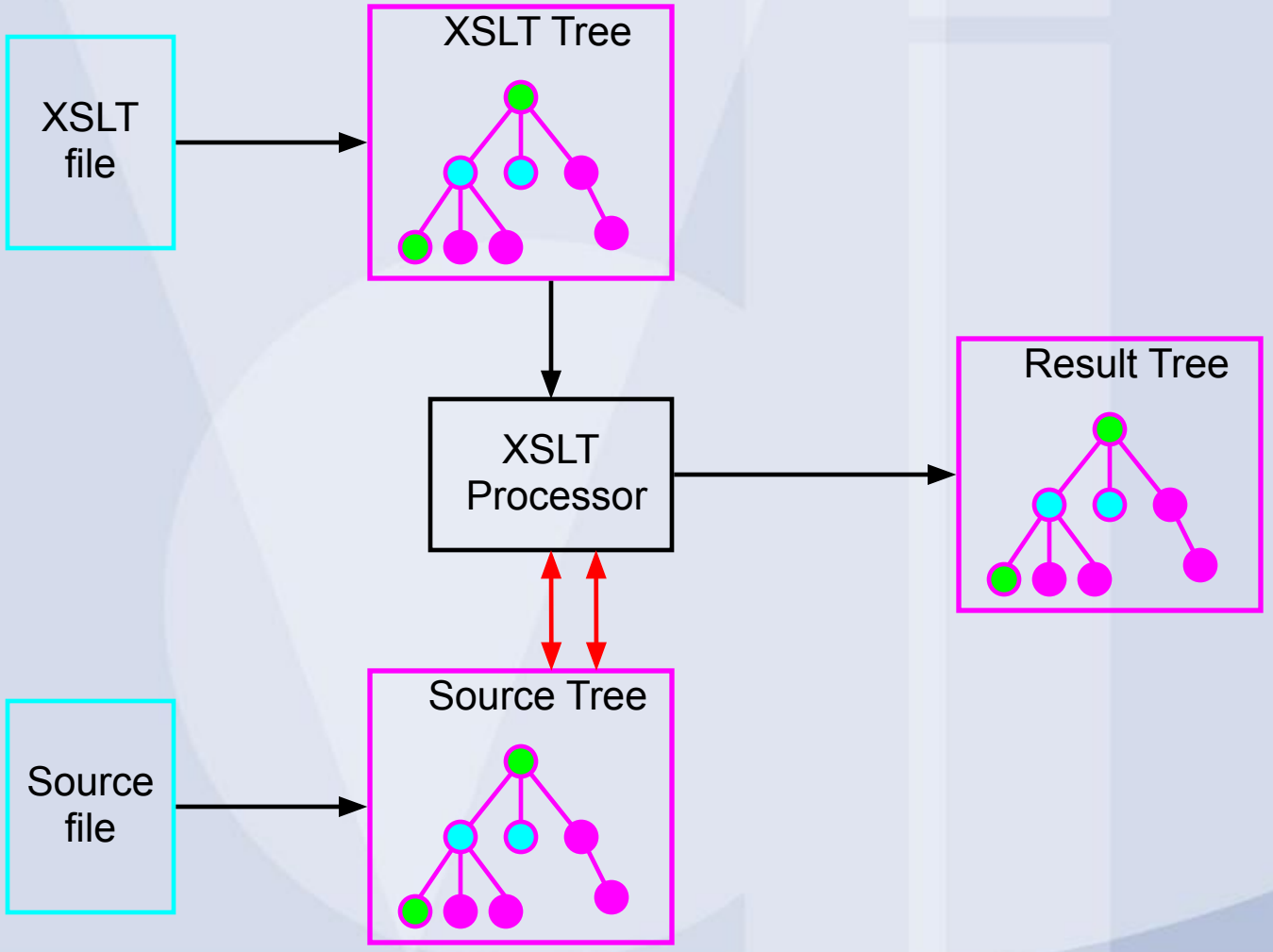
Hello, World!
Результат

XSLT – декларативный язык основанный на правилах

- Правило (template rule)
- Для обработки различных элементов используется набор правил
- Правила используются независимо друг от друга

Как это работает

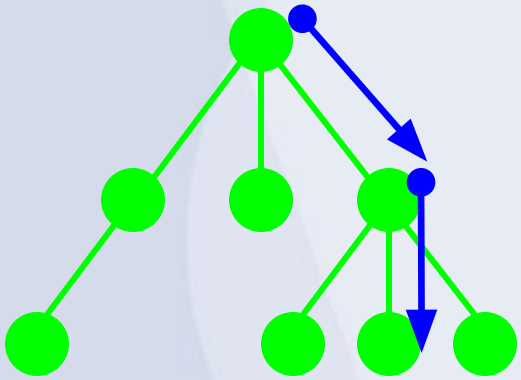
XSLT processing



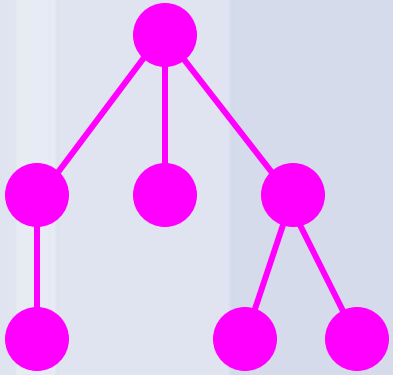
Как это работает

Представление документа в виде дерева и язык XPath

Source Tree



XSLT Tree



Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Основные элементы XSLT документа

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Объявление XML документа

XSLT документ как и все XML документы должен содержать объявление:

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

Объявление XSLT документа таблицы стилей

Корневым элементом любого XSLT документа является заголовок:

<xsl:stylesheet version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

Объявление формата для выходного документа

После того как XSLT процессор сгенерирует выходное дерево, он использует элемент <xsl:output> для создания текстового представления. Например:

<xsl:output method="xml"/>

<xsl:output method="html"/>

<xsl:output method="text"/>

Шаблон

Каждый шаблон состоит из двух частей: из образца (match pattern) и тела шаблона

<xsl:template match="region">

[Тело шаблона]

</xsl:template>

Шаблон – Образцы (match)

Образец задается в виде значения атрибута ‘match’ на языке XML Path Language (XPath)

Он определяет, каким узлам в исходном дереве соответствует шаблон:

<xsl:template match="region">

<xsl:template match="@books_sold">

Шаблон - тело шаблона

```
<xsl:template match="catalog">
    <h2>Titles Grouped by Author</h2>
    <xsl:for-each select="book[generate-id() =
generate-id(key('author_key', author)[1])]">
        <xsl:sort select="author"/>
        <h4> <xsl:value-of select="author"/></h4>
        <xsl:for-each select="key('author_key', author)">
            <xsl:sort select="title"/>
             <xsl:value-of select="title"/><br/>
        </xsl:for-each>
    </xsl:for-each>
</xsl:template>
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Конечные литеральные элементы

```
<xsl:template match="catalog" mode="grp_author">
    <h2>Titles Grouped by Author</h2>
    <xsl:for-each select="book[generate-id() =
     generate-id(key('author_key', author)[1])]">
        <xsl:sort select="author"/>
        <h4> <xsl:value-of select="author"/></h4>
        <xsl:for-each select="key('author_key', author)">
            <xsl:sort select="title"/>
             <xsl:value-of select="title"/><br/>
        </xsl:for-each>
    </xsl:for-each>
</xsl:template>
```

Комментарии

```
<h1>  Special Exhibit Catalog</h1>
    <!-- Group by catalog in first pass, then by author. -->
    <xsl:apply-templates select="catalog"
    mode="grp_categ"/>
    <xsl:apply-templates select="catalog"
    mode="grp_author"/>
    <xsl:apply-templates <!– Invalid comment -->/>
```

Переменные и параметры

```
<xsl:variable name="recentdate">2005-04-01</xsl:variable>
<xsl:variable name="ancillary" select="document('ancillary.xml')"/>

<xsl:template name="numbered-block">
    <xsl:param name="format">1. </xsl:param>
    <fo:block>
      <xsl:number format="{$format}"/>
      <xsl:apply-templates/>
    </fo:block>
</xsl:template>

<xsl:template match="ol//ol/li">
    <xsl:call-template name="numbered-block">
      <xsl:with-param name="format">a. </xsl:with-param>
    </xsl:call-template>
</xsl:template>
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Циклы

```
<xsl:for-each select="//book">
    <tr>
      <td> <xsl:value-of select="author"/> </td>
      <td> <xsl:value-of select="title"/> </td>
    </tr>
</xsl:for-each>
```

Условные выражения

```
<xsl:if test="price &gt; 10">
    <xsl:attribute name="bgcolor">lightgreen</xsl:attribute>
</xsl:if>

<xsl:choose>
    <xsl:when test="self::*[genre = 'Romance']">
      <xsl:attribute name="style">background-color: pink</xsl:attribute>
    </xsl:when>
    <xsl:when test="self::*[genre = 'Fantasy']">
      <xsl:attribute name="style">background-color: lightblue</xsl:attribute>
    </xsl:when>
    <xsl:otherwise>
      <xsl:attribute name="style">background-color: lightgreen</xsl:attribute>
    </xsl:otherwise>
</xsl:choose>
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Основные элементы XSLT документа - заключение

- Выражения на языке XPath
- XPath и XSLT функции
- Ссылки на внешние документы

XSLT processor – работа с исходным XML документом

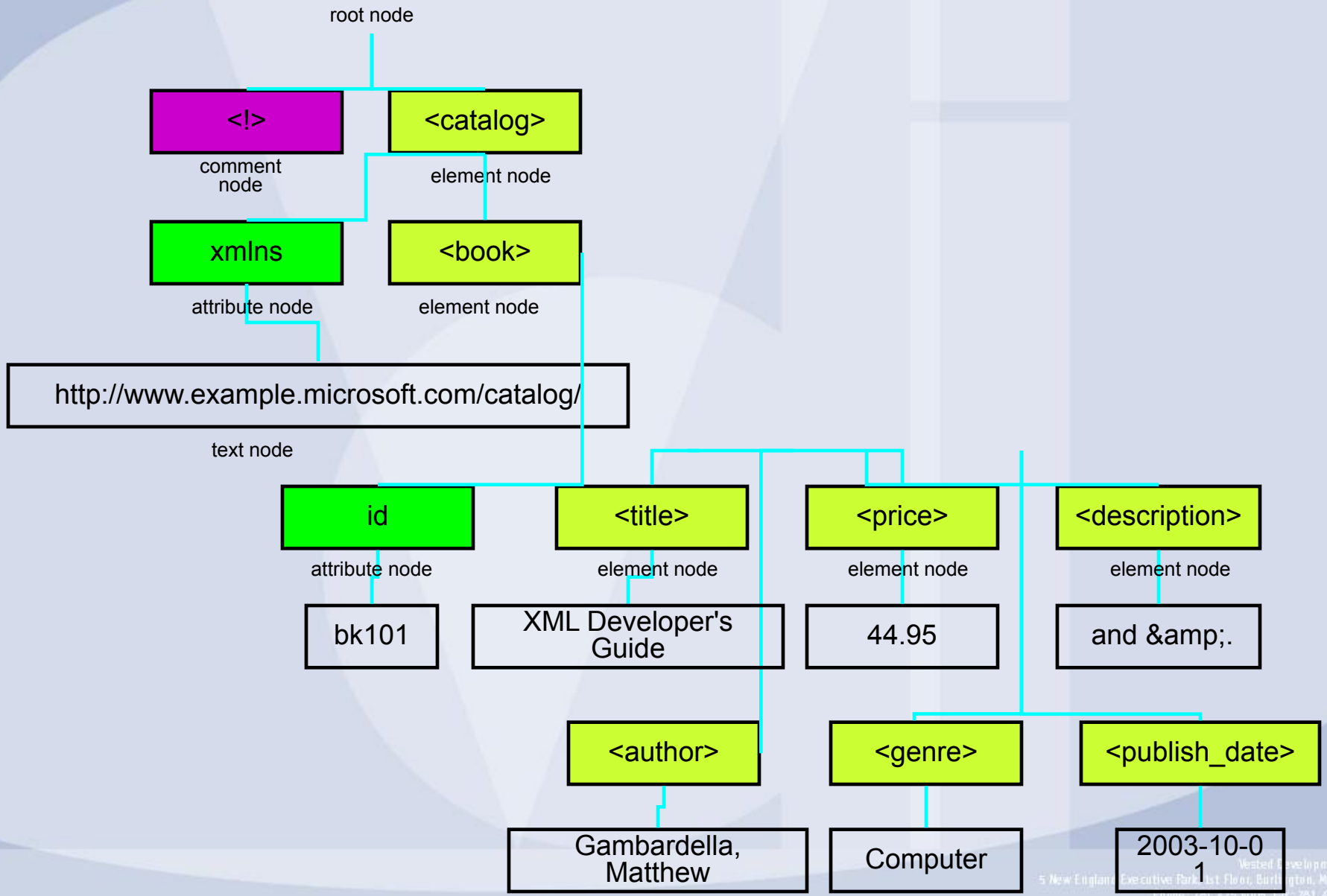
1. Поиск информации о преобразовании в XSLT дереве – поиск шаблонных правил (template rule)
2. Обработка/применение шаблонных правил к исходному дереву документа
3. Вывод преобразованных данных в виде выходного дерева

Исходный файл

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="show_book.xsl"?>
<!DOCTYPE catalog SYSTEM "catalog.dtd">
<!--catalog last updated 2005-11-01-->
<catalog xmlns="http://www.example.microsoft.com/catalog/">
    <book id="bk101">
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
      <price>44.95</price>
      <publish_date>2003-10-01</publish_date>
      <description><![CDATA[An in-depth look at creating applications
    with XML, using <, >,]]> and &amp;amp;.</description>
    </book>
</catalog>
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Исходное дерево документа



Имена и значения для вершин на XPath

Тип	Имя	Значение
Root	Пустая строка	Объединенное текстовое значение всех подчиненных текстовых вершин и вершин элементов
Элемент	Имя элемента*	См. выше
Атрибут	Имя атрибута*	Значение аттрибута
Текст	Пустая строка	Весь текст этой вершины
Processing-instruction	Цель (target) инструкции	Весь текст в инструкции, кроме target
Комментарий	Пустая строка	Весь текст между <!-- -->
Namespace	Префикс пространства имен или пустая строка, если префикса нет	URI пространства имен

* - расширенная форма включает URI пространства имен

Примеры имен и значений для вершин в документе

Тип	Имя	Значение
Root	Пустая строка	Gambardella, MatthewXML Developer's GuideComputer44.952003-10-01An in-depth look at creating applications with XML, using <, >, and &.
Элемент <catalog>	catalog	Gambardella, MatthewXML Developer's GuideComputer44.952003-10-01An in-depth look at creating applications with XML, using <, >, and &.
Элемент <book>	Book	Gambardella, MatthewXML Developer's GuideComputer44.952003-10-01An in-depth look at creating applications with XML, using <, >, and &.
Атрибут id (элемента book)	Id	Bk101

Обработка шаблонных правил

1. Поиск шаблона для корневой вершины документа

<xsl:template match="/*firstelement*">

2. Применение найденного шаблона
    1. Если нет условных выражений, то процессор применяет шаблон и далее обычно ищет ближайшую наивысшую вершину в исходном дереве и для нее находит подходящий шаблон и применяет его и так далее.
    2. Обрабатывает условное выражение и при его соответствии выполняет указанное в нем действие. Обычно условное выражение указывает на выполнение последовательности определенных шаблонных правил.

Управление обработкой шаблонных правил

1. Использование условных выражений

   Элементы <xsl:if> и <xsl:choose> - позволяют указать набор шаблонных правил, которые надо выполнять.

2. Импорт XSLT файлов

   Порядок перечисления элементов <xsl:import> влияет на приоритет выполнения шаблонных правил из этих файлов.

3. Разрешение конфликтов при выборе шаблонного правила для применения

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Встроенные шаблоны (Built-in Template Rules)

- **Шаблон для корневой вершины документа и элемента**

```
<xsl:template match="*|/">
    <xsl:apply-templates/>
</xsl:template>
```

- **Шаблон для аттрибуа и текста**

```
<xsl:template match="text()|@*">
    <xsl:value-of select="."/>
</xsl:template>
```

- **Шаблон для комментариев и инструкций**

```
<xsl:template match="processing-instruction()|comment()" />
```

Типы XSLT преобразований

- Преобразования (template-driven) управляемые шаблонами
- Преобразования (data-driven) управляемые данными

Template-driven преобразования – source file

```
<?xml version="1.0"?>
<books>
    <book>
     <title>Synchronized Jamming</title>                    <author>Kari Hensien</author>
     <abstract> A post modern flight of fancy. </abstract>
    </book>
</books>
```

Template-driven преобразования – XSL file

```
<?xml version="1.0"?> <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <HTML>
     <HEAD> </HEAD>
     <BODY>
          <H1><xsl:value-of select="/books/book/title"/></H1>
          <H3><xsl:value-of select="/books/book/author"/></H3>
          <P>
               <SPAN>Abstract:</SPAN>
               <SPAN><xsl:value-of select="/books/book/abstract"/></SPAN>

          </P>
     </BODY>
    </HTML>
</xsl:template>
</xsl:stylesheet>
```

Data-driven преобразования – source file

```
<?xml version='1.0'?>
<?xml-stylesheet type="text/xsl" href="book-review.xsl"?>
<book-review>
    <title>A Good Book</title> by <author>The Good
    Writer</author>, published by <publisher>The
    Publisher</publisher> on <date>A Good Day</date>, is
    indeed a good book. However, the one titled <title> A
    Bad Book</title> by the same publisher is very bad. This
    reviewer is left to wonder whether this is because
    <title>A Bad Book</title> was written by <author>A Bad
    Author</author>, or because it was published on
    <date>A Bad Date</date>.
</book-review>
```

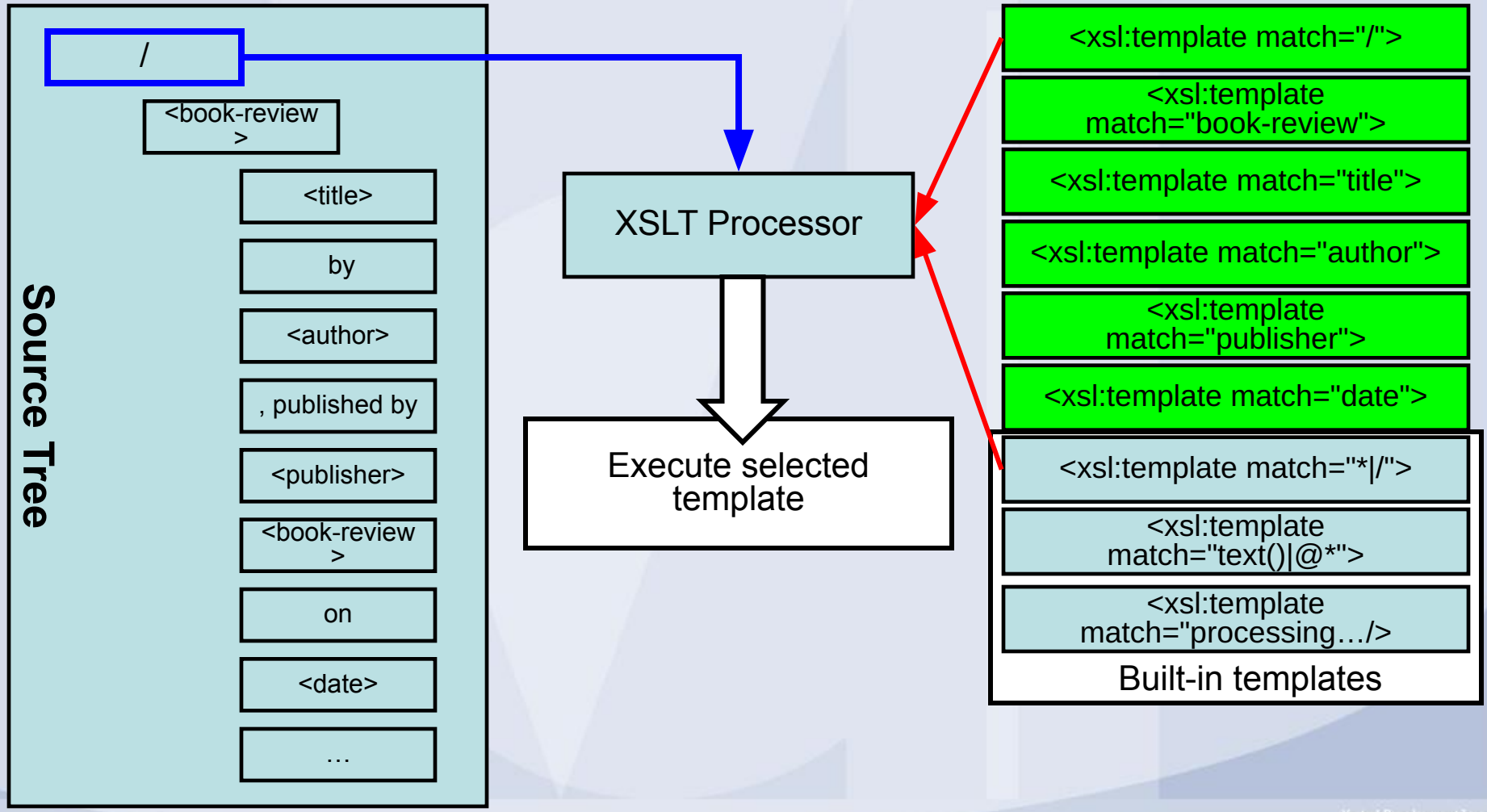
Data-driven преобразования – XSL file

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0“
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:template match="/">
          <HTML>
            <HEAD> </HEAD>
            <BODY> <xsl:apply-templates /> </BODY>
          </HTML>
      </xsl:template>
      <xsl:template match="book-review">
          <P><xsl:apply-templates /></P>
      </xsl:template>
      <xsl:template match="title">
          <SPAN style="font-weight:bold"><xsl:value-of select="."/></SPAN>
      </xsl:template>
```

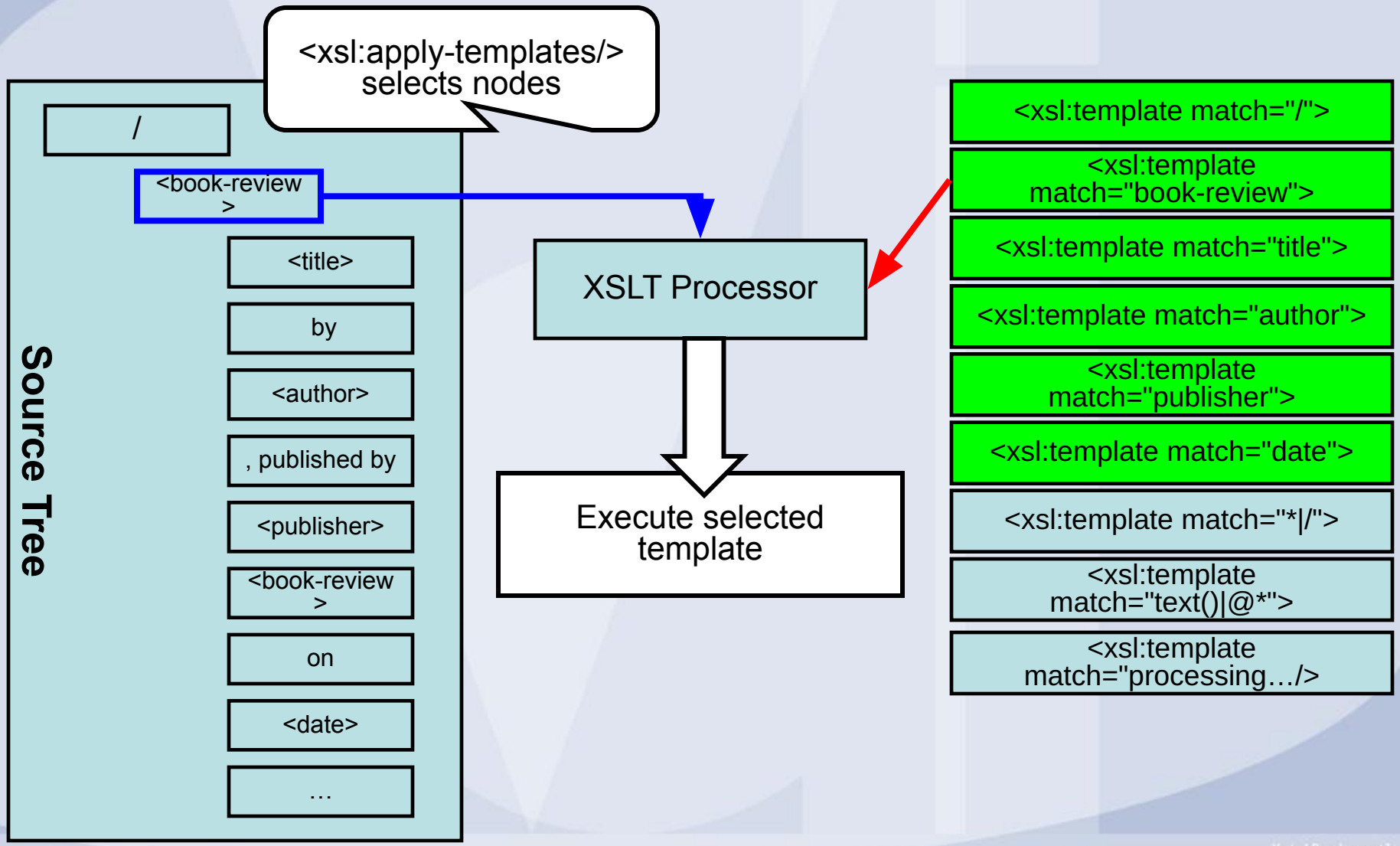
Data-driven преобразования – XSL file (продолжение)

```
    <xsl:template match="author">
        <SPAN style="font-style:italic"><xsl:value-of
        select="."/></SPAN>
    </xsl:template>
    <xsl:template match="publisher">
        <SPAN style="color:blue"><xsl:value-of select="."/></SPAN>
    </xsl:template>
    <xsl:template match="date">
        <SPAN style="font-family:courier"><xsl:value-of
        select="."/></SPAN>
    </xsl:template>
</xsl:stylesheet>
```

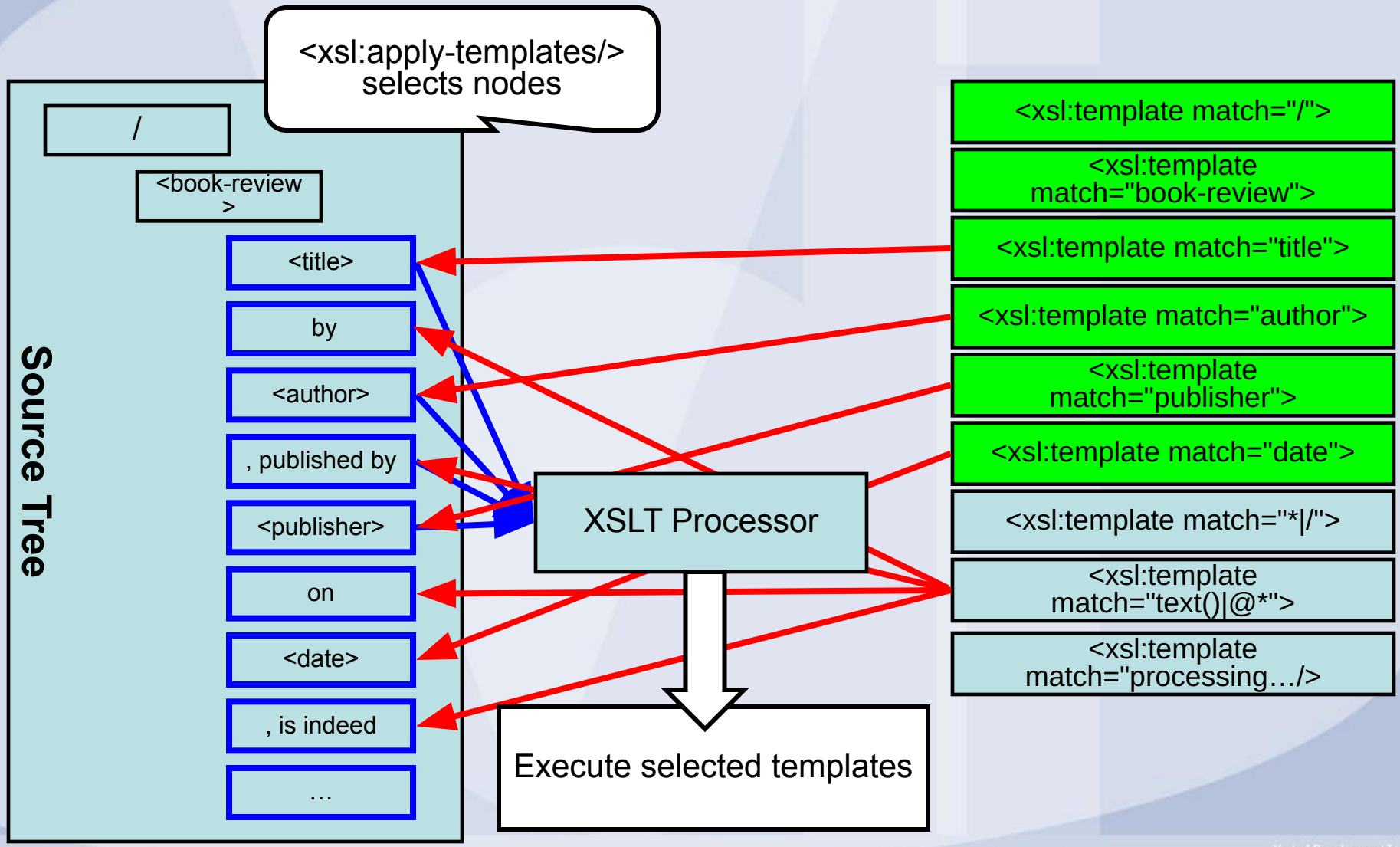
Пример работы XSLT процессора с шаблонами



Пример работы XSLT процессора с шаблонами продолжение



Пример работы XSLT процессора с шаблонами продолжение



Работа с образцами (match patterns) в элементах xsl:template

Аттрибут match содержит XPath выражение, подобное select в элементах <xsl:for-each>, <xsl:value-of> и <xsl:apply-templates>, но используются они по разному.

Select	Match
Стартует с текущей вершины (контекста) и возвращает вершины	Стартует с целевоей (target) вершины и возвращает true/false в случае совпадения образца
Стартует с текущего контекста и “подбирает” данные	Стартует сравнение с целевой вершины и далее может проверять текущий контекст, включая вершины-предки и вершины-наследники
Определяет запрос на данные в XML документе	Определяет описание шаблона-образца с которым будут сравниваться вершины в XML документе

Использование переменных (xsl:variable)

- Переменные в XSLT функциях – аналог именованных констант в традиционных языках программирования
  - Удобно использовать для хранения данных, к которым часто обращаются
  - Используется для хранения контекстно-зависимых данных и временного дерева
- Значение переменной не может быть изменено пока переменная не выйдет из области видимости
- Обращение к переменной осуществляется при помощи префикса $ перед названием переменной (аттрибут name элемента xsl:variable)
- Глобальные переменные определяются как непосредственный подчиненный элемент <xsl:stylesheet> - она может использоваться в любом месте таблицы стилей. Локальные переменные определяются в шаблонах и область их действия – контекст, в котором они определены

Использование переменных как констант

```
<xsl:for-each select="locale">
    <xsl:variable name="placename">
     <xsl:choose>
          <xsl:when test="@place='location1'">Midtown</xsl:when>
          <xsl:when test="@place='location2'">Northeast</xsl:when>
          <xsl:when test="@place='location3'">Airport</xsl:when>
    <xsl:otherwise>[Unknown Locale]</xsl:otherwise>
     </xsl:choose>
    </xsl:variable>
    <H3><xsl:value-of select="$placename"/></H3>
</xsl:for-each>
```

Использование переменных для упрощения описания

```
<xsl:for-each select="day">
    <xsl:variable name="average_temp" select="format-number(
    sum(locale/temp/high) div count(locale), '##0.00')"/>
    <H2>As of <xsl:value-of select="@date"/></H2>
    <P> Average Temperature: <xsl:value-of
    select="$average_temp"/>&#176;F </P>
```

Использование параметров (xsl:parameter)

- Параметр в XSLT функциях – аналог переменной в традиционных языках программирования
- Обращение к параметру осуществляется при помощи префикса $ перед именем параметра (аттрибут name элемента xsl:parameter)
- Глобальный параметр определяется как непосредственный подчиненный элемент элемента <xsl:stylesheet>. Он может использоваться в любом месте таблицы стилей. Локальный параметр определяется в шаблоне и область его действия – контекст, в котором он определен
- Параметр можно рассматривать как параметризованную переменную. После ее определения/инициализации значение ее не изменяется. Но параметр может передавать значение, которое присваивается ему вне области его действия. Т.е. глобальный параметр может хранить значение переданное ему из скрипта на HTML странице, а локальный параметр хранит значение назначенное при вызове шаблона, в который передается значение параметра.

Пример использования локального параметра

```
<xsl:template name="placename">
    <xsl:param name="location" select="'[Unknown Locale]'"/>
    <xsl:choose>
      <xsl:when test="$location='location1'">Midtown</xsl:when>
      <xsl:when test="$location='location2'">Northeast</xsl:when>
      <xsl:when test="$location='location3'">Airport</xsl:when>
      <xsl:otherwise>[Unknown Locale]</xsl:otherwise> </xsl:choose>
</xsl:template>

...
      <xsl:call-template name="placename">
         <xsl:with-param name="location" select="@place"/>
      </xsl:call-template>
```

Сортировка в XSLT

- Сортировка – очень простая операция, выполняемая при помощи элемента <xsl:sort>
- Элемент <xsl:sort> указывает ключ, по которому должна производится сортировка. Этот элемент должен быть указан в <xsl:for-each> или <xsl:apply-templates>

Сортировка в XSLT – исходный XML документ

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="prodsort.xsl" ?>
<products>
    <product prodID="AX5608">
      <name>FooBar</name>
      <version>1.5</version>
      <descr>Processes foo objects using standard FB API</descr>
      <categ>Software</categ>
      <price curr="USD">149.99</price>
      <units>each</units>
    </product>
    <product prodID="CB3241">
      <name>TrixelMaker</name>
      <version>3.0</version>
      <descr>Burns multiple trixels from single master (requires Wooden Rings 1.6 or
    greater)</descr>
      <categ>Hardware</categ>
      <price curr="EU">178.49</price>
      <units>each</units>
    </product> ...
```

Сортировка в XSLT – исходный XSL документ

```
<xsl:template match="products">
    <TABLE width="75%">
      <tr> <th>Category</th> <th>Prod ID</th> <th>Name/Version</th>
    <th>Description</th> <th>Price/Units</th> </tr>
      <xsl:apply-templates/>
    </TABLE>
</xsl:template>

<xsl:template match="product">
    <tr>
      <td valign="top"><xsl:value-of select="categ"/></td>
      <td valign="top"><xsl:value-of select="@prodID"/></td>
      <td valign="top"><xsl:value-of select="concat(name, '/', version)"/></td>
      <td valign="top"><xsl:value-of select="descr"/></td>
      <td valign="top" align="center"><xsl:value-of select="concat(price, ' (', price/@curr,
    ')')"/></td>
      <td valign="top" align="right"><xsl:value-of select="usd_equiv"/></td>
    </tr>
</xsl:template>
```

Сортировка в XSLT – сортировка по одному ключу

```
<xsl:template match="products">
    <TABLE width="75%">
     <tr> <th>Category</th> <th>Prod ID</th> <th>Name/Version</th>
    <th>Description</th> <th>Price/Units</th> </tr>
     <xsl:apply-templates>
          <xsl:sort select="categ"/>
     <xsl:apply-templates/>
    </TABLE>
</xsl:template>
```

Сортировка в XSLT – сортировка по двум ключам

```
<xsl:template match="products">
    <TABLE width="75%">
     <tr> <th>Category</th> <th>Prod ID</th> <th>Name/Version</th>
    <th>Description</th> <th>Price/Units</th> </tr>
     <xsl:apply-templates>
          <xsl:sort select="categ"/>
          <xsl:sort select="name"/>
     <xsl:apply-templates/>
    </TABLE>
</xsl:template>
```

Сортировка в XSLT – сортировка по числовым значениям

```
<xsl:template match="products">
    <TABLE width="75%">
     <tr> <th>Category</th> <th>Prod ID</th> <th>Name/Version</th>
    <th>Description</th> <th>Price/Units</th> </tr>
     <xsl:apply-templates>
          <xsl:sort select="categ"/>
          <xsl:sort select="name"/>
          <xsl:sort select="version" data-type="number"/>
     <xsl:apply-templates/>
    </TABLE>
</xsl:template>
```

Применение автоматической нумерации в XSLT

- Использование XPath функции position()
- Использование элемента <xsl:number>

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Применение автоматической нумерации в XSLT – исходный XML документ для примера

```
<?xml version="1.0" encoding="UTF-8"?>
<cardgame>
  <hand>
      <player name="Jack">
        <card dealtID="card01">
          <rank>Q</rank><suit>Clubs</suit>
        </card>
        …
      <player name="Jill">
        …
  </hand>
  <hand>
  …
</cardgame>
```

Применение автоматической нумерации в XSLT – использование функции position()

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns="http://www.w3.org/TR/REC-html40"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
   <HTML>
      <HEAD>
         <TITLE>Card Game</TITLE>
      </HEAD>
      <BODY>
            <xsl:apply-templates/>
      </BODY>
   </HTML>
</xsl:template>
```

Применение автоматической нумерации в XSLT – использование функции position() (продолжение)

```
<xsl:template match="hand">
    <h1>Hand #<xsl:value-of select="position()"/></h1>
    <xsl:apply-templates/>
</xsl:template>
</xsl:stylesheet>
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Применение автоматической нумерации в XSLT – использование функции position() (результат работы)

Hand #1

QClubsASpades10Hearts8Clubs4Spades5Diamonds5Spades6Clubs6Spades10Diamonds

Hand #2

4DiamondsKSpades7Spades6ClubsKSpadesKDiamonds7Diamonds9HeartsJSpades4Clubs

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Применение автоматической нумерации в XSLT – использование <xsl:number>

Специально используется для генерации числовых строк для иерархий:

1. Item
    1.1. Sub-item
        1.1.A. First sub-item
    1.2. Sub-item

Применение автоматической нумерации в XSLT – использование <xsl:number>

```
<xsl:template match="player">
    <h2>
        Player <xsl:number/>:
        <xsl:value-of select="@name"/>
    </h2>
    <xsl:apply-templates/>
</xsl:template>
```

Использование <xsl:number> - результат работы

Hand #1
Player 1: Jack
QClubsASpades10Hearts8Clubs4Spades
Player 2: Jill
5Diamonds5Spades6Clubs6Spades10Diamonds
Hand #2
Player 1: Jack
4DiamondsKSpades7Spades6ClubsKSpades
Player 2: Jill
KDiamonds7Diamonds9HeartsJSpades4Clubs

Использование <xsl:number> - атрибуты level, format, number

```
<xsl:template match="player">
   <h2>
      Player
      <xsl:number
         level="multiple"
         format="1.1"
         count="hand | player"/>:
      <xsl:value-of select="@name"/>
   </h2>
   <xsl:apply-templates/>
</xsl:template>
```

Использование <xsl:number> - атрибуты level, format, number – результат работы

Hand #1

Player 1.1: Jack

QClubsASpades10Hearts8Clubs4Spades

Player 1.2: Jill

5Diamonds5Spades6Clubs6Spades10Diamonds

Hand #2

Player 2.1: Jack

4DiamondsKSpades7Spades6ClubsKSpades

Player 2.2: Jill

KDiamonds7Diamonds9HeartsJSpades4Clubs

Использование именованных шаблонов (named template rules)

```
<xsl:template name="display_price">
   <xsl:value-of select="@price"/>
</xsl:template>

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

Использование именованных шаблонов – упрощение описания

```
<h3>Company-Wide Results</h3>
<table width="100%">
   <tr>
    <th width="33%"># Units</th>
    <th width="33%">Qtr Sales Amount</th>
    <th>Amt/Unit</th>
   </tr>
   <tr> [cells for company-wide data] </tr>
</table>
```

Использование именованных шаблонов – упрощение описания (продолжение)

```
<xsl:template name="table_hdgs">
    <tr>
      <th width="33%"># Units</th>
      <th width="33%">Qtr Sales Amount</th>
      <th>Amt/Unit</th>
    </tr>
</xsl:template>
...
<h3>Company-Wide Results</h3>
<table width="100%">
      <xsl:call-template name="table_hdgs"/>
      <tr> [cells for company-wide data] </tr>
</table>
```

Использование именованных шаблонов – передача параметров

```
<sales quarter="2001-01-01">
   <region>
    <name>Northeast</name>
    <manager>Kim Abercrombie</manager>
    <units>9881</units>
    <sales_amt curr="">150680.89</sales_amt>
   </region>
   <region>
    <name>Southeast</name>
    …
```

Использование именованных шаблонов – передача параметров (продолжение)

```
<tr>
    <td width="33%" align="right">
     <xsl:value-of select="sum(region/units)"/>
    </td>
    <td width="33%" align="right">
     <xsl:value-of select="sum(region/sales_amt)"/> </td>
    <td align="right">
     <xsl:value-of select="sum(region/sales_amt) div
    sum(region/units)"/>
    </td>
</tr>
```

Использование именованных шаблонов – передача параметров (продолжение)

```
<xsl:template name="common_table">
    <xsl:param name="table_hdg"
    select="'Default Heading'"/>
    <xsl:param name="units_param"
    select="'Default Units'"/>
    <xsl:param name="amt_param"
    select="'Default Amount'"/>
...
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Использование именованных шаблонов – передача параметров (продолжение)

```
<tr>
    <td width="33%" align="right">
     <xsl:value-of select="$units_param"/>
    </td>
    <td width="33%" align="right">
     <xsl:value-of select="$amt_param"/> </td>
    <td align="right">
     <xsl:value-of select="$amt_param div $units_param"/>
    </td>
</tr>
```

Использование именованных шаблонов – передача параметров (продолжение)

```
<xsl:template match="sales">
    <h1>Quarterly Sales by Region</h1>
    <h2>Quarter Beginning <xsl:value-of select="@quarter"/></h2>
    <xsl:apply-templates/>
    <xsl:call-template name="common_table">
     <xsl:with-param name="table_hdg" select="'Company-Wide Results'"/>
     <xsl:with-param name="units_param" select="sum(region/units)"/>
     <xsl:with-param name="amt_param" select="sum(region/sales_amt)"/>
    </xsl:call-template>
</xsl:template>
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Использование именованных шаблонов – передача параметров (продолжение)

```
<xsl:template match="region">
    <xsl:call-template name="common_table">
     <xsl:with-param name="table_hdg">
          <xsl:value-of select="name"/> Region (Manager:
    <xsl:value-of select="manager"/>)
     </xsl:with-param>
     <xsl:with-param name="units_param"
    select="units"/>
     <xsl:with-param name="amt_param"
    select="sales_amt"/>
    </xsl:call-template>
</xsl:template>
```

Использование XSLT ключей для увеличения производительности

- Решение проблемы многократного и быстрого обращения к одним и тем же вершинам исходного документа
  - Ключ – уникально идентифицирует вершину или класс вершин
  - Ключ используется для построения индекса, к которому XSLT процессор обращается при указании определенного значения для него
  - Индекс дает возможность обратиться к “ключевой” вершине без выполнения операции поиска
  - Ключ значительно упрощает кодирование XPath выражений

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Использование элемента <xsl:key> - объявление ключа

- <xsl:key> - элемент верхнего уровня, непосредственный подчиненный элемента <xsl:stylesheet>, количество обявлений неограничено
- Синтаксис

  <xsl:key name="*name*" match="*node*" use="*expression*" />

  – *name* – имя ключа, используется для обращения к нему
  – *node* – имя вершины документа, с которой связан ключ
  – *expression* - XPath выражение, которое объявляет значение ключа

Примеры объявления ключа <xsl:key>

1. <xsl:key name="isbn_key" match="book" use="@isbn" />
   1. Имя ключа - isbn_key
   2. Ключ назначается для вершин-элементов book
   3. В качестве значения ключа используется аттрибут вершины book - isbn

2. <xsl:key name="categ_key" match="book" use="category" />

3. <xsl:key name="book_for_categ" match="category" **use="../title"** />

Обращение к ключу – использование XSLT функции key()

- Функция key() служит для получения вершины или набора вершин указанных ключом и его значением
- Синтаксис

  key(*name*, *value*)

  name – имя ключа, объявленного элементом <xsl:key>

  value - значение ключа (может быть любого типа, даже набором вершин), может использоваться переменная

Примеры использования XSLT функции key()

Объявление ключа	Вызов функции key()	Результат
<xsl:key name="isbn_key" match="book" use="@isbn"/>	key("isbn_key", "991100-102")	См. следующий слайд
<xsl:key name="categ_key" match="book" use="category"/>	key("categ_key", "Fiction")	
<xsl:key name="author_id" match="book" use="author"/>	key("author_id", contains("Culbert"))	

Примеры использования XSLT функции key() - продолжение

```
<catalog>
  <book isbn="991100-001" catnum="101.3490">
    <title>Jambing on the Trixels</title>
    <author>Tristan Randall</author>
    <publisher>Scootney Publishing</publisher>
    <category>Fiction</category>
  </book>
  <book isbn="991100-008" catnum="562.1093">
    <title>Fretting Over Your Guitar</title>
    <author>Becki Culbert</author>
    <publisher>ScootMusic (a Division of Scootney Publishing)</publisher>
    <category>Non-fiction</category>
  </book>
  <book isbn="991100-102" catnum="450.9043">
    <title>Quantum Superstring Electrodynamics for Newbies</title>
    <author>Becki Culbert</author>
    <publisher>Scootney Publishing</publisher>
    <category>Non-fiction</category>
  </book>
</catalog>
```

key("categ_key", "Fiction")

key("author_id", contains("Culbert"))

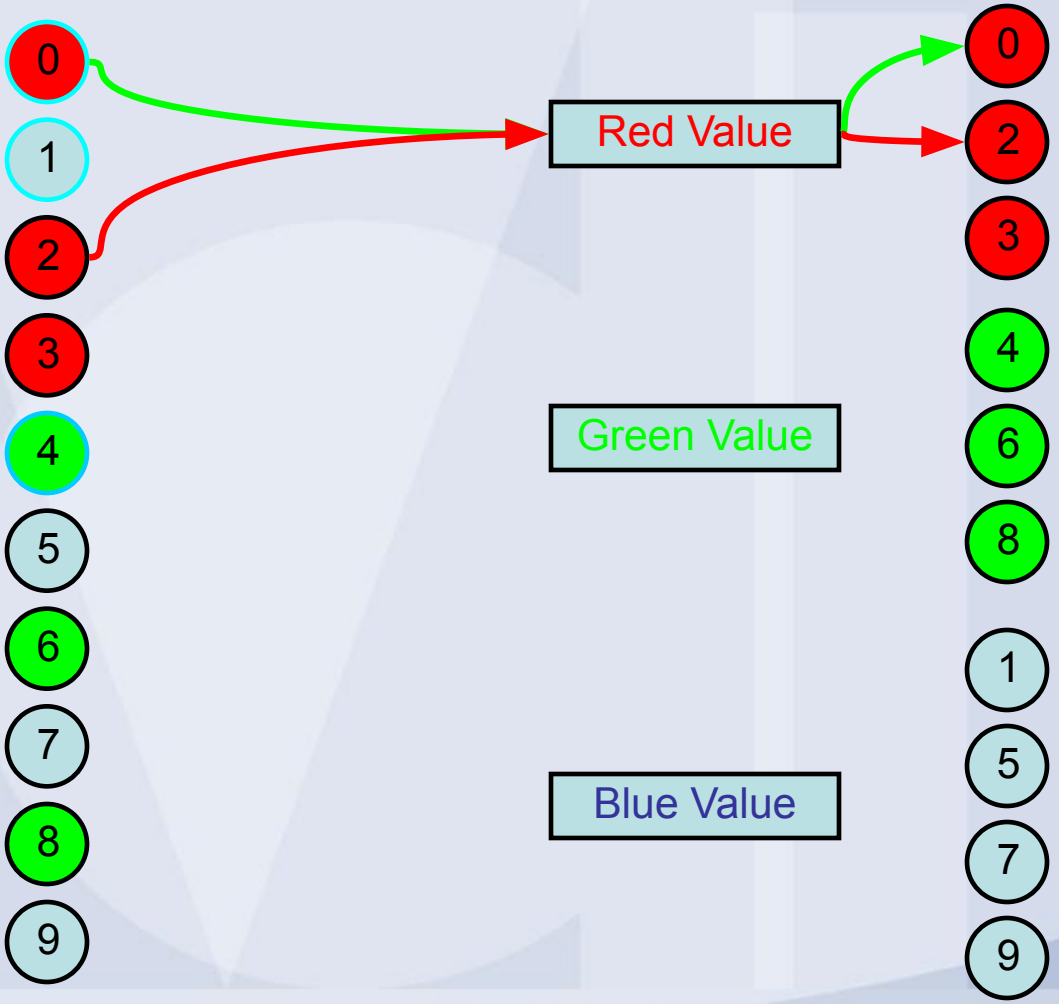
key("isbn_key", "991100-102")

Использование XSLT ключей для группировки данных

Два способа решения задачи группировки:

1. Использование функции generate-id()
2. Использование сравнений с наборами вершин, возвращаемых ключом

Алгоритм решения задачи группировки данных



Группировка с использованием функции generate-id()

```
<xsl:key name="author_key" match="book" use="author"/>
...
<xsl:template match="catalog" mode="grp_author">
        <h2>Titles Grouped by Author</h2>
        <xsl:for-each select="book[generate-id() =
        generate-id(key('author_key', author)[1])]">
            <xsl:sort select="author"/>
            <h3><xsl:value-of select="author" /></h3>
            <xsl:for-each select="key('author_key', author)">
                <xsl:sort select="title"/>
                    ...
                </xsl:for-each>
        </xsl:for-each>
</xsl:template>
```

Группировка с использованием сравнений с наборами вершин, возвращаемых ключом

```
<xsl:key name="author_key" match="book" use="author"/>
...
<xsl:template match="catalog" mode="grp_author">
    <h2>Titles Grouped by Author</h2>
    <xsl:for-each select="book[count(. | key('author_key', author)[1])=1]">
        <xsl:sort select="author"/>
        <h3><xsl:value-of select="author" /></h3>
        <xsl:for-each select="key('author_key', author)">
            <xsl:sort select="title"/>
                ...
            </xsl:for-each>
    </xsl:for-each>
</xsl:template>
```

Работа с пространством имен (namespace) и с их префиксами

Что такое пространство имен?

- Пространство имен определяет класс или набор элементов и атрибутов предназначенных для использования в определенной области:
  - HTML - <HTML>, <BODY>, <TABLE>
  - XSLT - <xsl:stylesheet>, <xsl:template>, <xsl:if>
- Пространство имен идентифицируется при помощи URI (пример: http://www.w3.org/1999/XSL/Transform) и для сокращенного указания используются префикс (пример: xsl)
- Принадлежность элемента пространству имен описывается в формате: namespace-prefix:element-name. (пример: xsl:sort)

Объявление пространства имен для XSLT

- Объявление пространства имен выполняется через атрибут *xmlns* в элементе:

```
<xsl:stylesheet version="1.0
xmlns:xsl="http://www.w3.org/1999/XSL/Transform“
xmlns="http://www.w3.org/TR/REC-html40“ >
```

- Выбор префикса и URI при объявлении
  - префикс – требование такое же как к названию XML элемента, удобочитаемое для человека
  - URI – выбор критичен (при совпадении с общеизвестными URI будет ассоциироваться с элементами из этого пространства имен)

Работа с Result Tree Fragments

- Result Tree Fragment – это часть выходного дерева (output tree), которая не является набором вершин.

The <color hue="red">red</color> color is <value brightness="bright">bright</value>.

Result Tree Fragments – подробное описание

1. Result Tree Fragment (RTF) – это тип данных указанных в спецификации XSLT, *неизвестный XPath*
2. RTF – часть результирующего дерева XSLT трансформации, “образец” для вывода в результирующее дерево, который хранится в <xsl:variable> или <xsl:parameter>
3. RTF – может быть не well-formed, но правильно сбалансированнным (наличие начального и конечного тегов и их правильная вложенность)
4. RTF может не иметь корневого элемента

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Result Tree Fragments – пример

Пример

```
<xsl:variable name="tree">
    <em>Emphasized and <b>bold</b></em>
    words
</xsl:variable>
```

Результат обращения к переменной – в выходное дерево попадет:

Emphasized and bold words

- Элемент - Emphasized and bold
- Текст – “ words”

Result Tree Fragments – попытка работы с ним

```
<xsl:template match="/">
    <table>
      <tr><th>Node name</th><th>Node
    type</th></tr>
      <xsl:for-each
    select="$tree/descendant-or-self::*">
        <tr><td>
          ...
```

ОШИБКА:

Expression must evaluate to a node-set.

-->$tree<--/descendant-or-self::*

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Result Tree Fragments – решение задачи

Использование функции msxsl:node-set():

1. Объявление пространства имен с префиксом msxsl

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt">
```

2. Вызов функции msxsl:node-set()

```
<xsl:for-each
    select="msxsl:node-set($tree)/descendant-or-self::*
    ">
```

Результат: Вместо строки или фрагмента для копирования в выходное дерево получаем полноценное дерево вершин.

Пример использования Result Tree Fragments – XML документ

Фрагмент исходного XML документа:

```
<products>
    <product prodID="AX5608">
        <name>FooBar</name>
        <descr>Processes foo objects using standard FB API</descr>
        <categ>Software</categ>
        <price curr="USD">149.99</price>
    </product>
    <product prodID="CB3241">
        <name>TrixelMaker</name>
        <descr>Burns multiple trixels from single master</descr>
        <categ>Hardware</categ>
        <price curr="EU">178.49</price>
    </product>
```

Пример использования Result Tree Fragments – решение

Задача: получить список продуктов, отсортированных по цене (проблема в разных валютах)

1. Попытка решения без RTF:
   1) Завести переменную – эквавалент доллара, которая пересчитывает цену в долларах
   2) Сделать сортировку, используя эту переменную, а не значение элемента <price>

   Ошибка: нельзя использовать переменную для атрибута select элемента <sort>. Это должно быть XPath выражение.
2. Решение с использованием RTF:
   1) Скопировать входное дерево документа в RTF
   2) Добавить вершину “цена в долларах” и добавить ее в RTF
   3) Выбрать и отсортировать нужные вершины из RTF

Пример использования Result Tree Fragments – шаг 1.1

```
<xsl:template match="products">
    <xsl:variable name="prods_with_usd">
      <xsl:apply-templates select="product" mode="calc_usd" />
    </xsl:variable>
    <TABLE width="75%">
      <tr>
        <th>Category</th>
        <th>Prod ID</th>
        <th>Name</th>
        <th>Description</th>
        <th>Price (Currency)</th>
        <th>Price (USD)</th>
      </tr>
    </TABLE>
</xsl:template>
```

Пример использования Result Tree Fragments – шаг 1.2

```
<xsl:template match="product" mode="calc_usd">
    <xsl:copy>
        <xsl:copy-of select="@*" />
        <xsl:copy-of select="*" />
    </xsl:copy>
</xsl:template>
```

Пример использования Result Tree Fragments – шаг 2

```
<xsl:template match="product" mode="calc_usd">
        <xsl:copy>
            <xsl:copy-of select="@*" />
            <xsl:copy-of select="*" />
        </xsl:copy>
        <xsl:element name="usd_equiv">
            <xsl:choose>
                <xsl:when test="price/@curr='USD'">
                    <xsl:value-of select="format-number(price, '#,##0.00')"/>
                </xsl:when>
                <xsl:when test="price/@curr='GBP'">
                    <xsl:value-of select="format-number(price * 1.87056, '#,##0.00')"/>
                </xsl:when>
                <xsl:when test="price/@curr='EU'">
                    <xsl:value-of select="format-number(price * 1.27845, '#,##0.00')"/>
                </xsl:when>
                <xsl:otherwise>Unknown Currency</xsl:otherwise>
            </xsl:choose>
        </xsl:element>
</xsl:template>
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Пример использования Result Tree Fragments – шаг 3

1. Подключаем пространство имен xmlns:msxsl="urn:schemas-microsoft-com:xslt" в элементе <xml:stylesheet> для использования функции msxsl:node-set()
2. Делаем изменение в шаблоне products – добавляем выражение ():

```
...
            <th>Price (USD)</th>
        </tr>
        <xsl:apply-templates select="msxsl:node-set($prods_with_usd)/product">
            <xsl:sort select="usd_equiv" data-type="number" />
        </xsl:apply-templates>
</TABLE>
```

3. Добавляем шаблон для обработки вершин product из RTF:

```
<xsl:template match="product">
        <tr>
            <td valign="top"><xsl:value-of select="categ"/></td>
            ...
            <td valign="top" align="right"><xsl:value-of select="usd_equiv"/></td>
        </tr>
</xsl:template>
```

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Обработка символов white space в XSLT

- Символы white space в XML документе
- Обработка символов white space XML процессором
- Возможности работы с символами white space в XSLT

Символы white space в XML документе

- Символы white space – это набор символов, которые “не высвечиваются” в текстовых редакторах

Символ	Entity Reference
Space	(hexadecimal)
Carriage return
 (
) |
| Tab | 	 (	) |
| Line feed (newline) |
 (
) |

Представление и операции с white space символами в XML

Фрагмент в документе:

<books>
    <book>

Реально представляет:

<books>
	 <book>

Единственная *white space* вершина содержит:

Операция нормализации:

<books> <book>
<books> <book>

Операция очистки (strip):

<books><book>

Сохранение white space вершин в XML и XSLT

1. **Non-breaking space   ( )**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xsl:stylesheet [<!ENTITY nbsp " ">]>
...
        <td> </td>
```

2. **<xsl:preserve-space>**

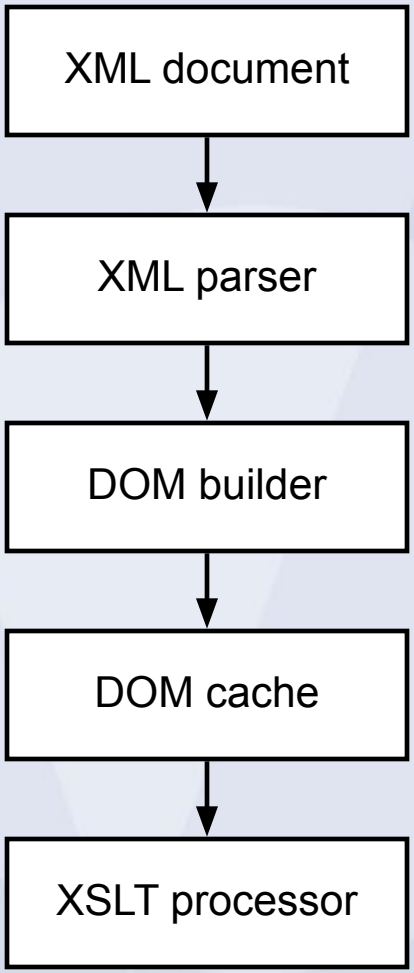
Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Правила работы с white space символами XML парсера

1. **В XML документе можно использовать xml:space атрибут (“default”, “preserve”)**
2. **Парсер нормализует символы, описывающе новую строку и специфичные для ОС, к символу 0xA (Line Feed)**
3. **Парсер нормализует значение атрибутов**
4. **Если явные установки для XSLT процессора конфликтуют со значением в атрибуте xml:space XML или XSLT документа, то преимущество имеет значение атрибутв xml:space**
5. **XSLT процессор объединяет две соседние текстовые вершины в одну текстовую. Если текстовая вершина содержит только white space, то ее элементы сравниваются с элементами в <xsl:strip-space>. Если совпадение есть, то вершина удаляется из результирующего дерева. Это для несущественных white space вершин между элементами. Внутри элементов это контролируется normalize-space() функцией**

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Схема работы с XML документом



Проверка установленного флага по работе с white space

Использование <xsl:text> для вывода символов white space

```
<body>
    <xsl:value-of select="'White space stripped:  '>
    <xsl:value-of select="'&lt;-here'"/><br/>
    <xsl:value-of select="' White space preserved:'"/>
    <xsl:text>  </xsl:text>
    <xsl:value-of select="'&lt;-here'"/>
</body>
```

Выходной документ:

White space stripped:<-here

White space preserved: <-here

Использование <xsl:preserve-space> и <xsl:strip-space>

<?xml-stylesheet type="text/xsl" ?>
<document>

<block> </block>	<code> </code>
<block>	<code>
TABSome textTAB	TABSome textTAB
TAB	TAB
TAB</block>	TAB</code>
<block>	<code>
TAB	TAB
TAB	TAB
TAB</block>	TAB</code>

</document>

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Использование <xsl:preserve-space> и <xsl:strip-space> - XSLT шаблон

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:preserve-space elements="code"/>
<xsl:strip-space elements="block"/>
…
<xsl:template match="/">
…
<xsl:for-each select="//code">
"Code" #<xsl:value-of select="position()"/>: [<xsl:value-of
    select="translate(.,'&#10;&#13;&#9; ','NRT-')"/>]<br/>
</xsl:for-each>
```

Результат работы

"Code" #1: [-] "Code" #2: [NTNSome-textTNTNT] "Code" #3: [-]	"Block" #1: [] "Block" #2: [NTNSome-textTNTNT] "Block" #3: []

Использование атрибута xml:space в XML документе

```
<block xml:space="preserve"> </block>
...
<block xml:space="preserve">
TAB

TAB
TAB</block>
```

Результат работы

"Code" #1: [-] "Code" #2: [NTNSome-textTNTNT] "Code" #3: [NTNNTNT]	"Block" #1: [-] "Block" #2: [NTNSome-textTNTNT] "Block" #3: [NTNNTNT]

Использование CDATA секций – сохранение формата текста

```
<?xml version='1.0'?>
<?xml-stylesheet type="text/xsl" href="cdata.xsl"
   ?>
<document>
<scriptcode language="JScript">
function message(msg){
alert(msg);
}
</scriptcode>
</document>
```

Использование CDATA секций – сохранение формата текста – XSLT документ

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output cdata-section-elements="script"/>
<xsl:template match="scriptcode">
<document>
<script language="JScript">
<xsl:value-of select="."/>
</script>
</document>
</xsl:template>
</xsl:stylesheet>
```

Использование CDATA секций – сохранение формата текста – выходной документ

```
<?xml version="1.0" encoding="UTF-16"?>
<document>
<script language="JScript"><![CDATA[
function message(msg){
alert(msg);
}
]]>
</script>
</document>
```

Использование XSLT функции normalize-space()

<generouswhitespace>
There is a l o t of white space here!
</generouswhitespace>

<xsl:template match="generouswhitespace">
[<xsl:value-of select="normalize-space(.)" />]
</xsl:template>

Результат:

[There is a l o t of white space here!]

Обработка white space символов в XSLT документе

Исходный текст в XSLT документе:

<td> </td>

В выходном документе:

<td></td>

Исходный текст в XSLT документе – решение проблемы:

<td><xsl:text> </xsl:text></td>

XSLT elements

- xsl:apply-imports Выполнение перегруженных шаблонных правил.
- xsl:apply-templates Применение шаблонов.
- xsl:attribute Создание нового атрибута.
- xsl:attribute-set Объявление набора атрибутов.
- xsl:call-template Вызов шаблона по имени.
- xsl:choose Множественный условный выбор.
- xsl:comment Создание комментария.
- xsl:copy Копирование текущего узла.
- xsl:copy-of Вставка фрагментов дерева.
- xsl:decimal-format Объявление десятичного формата.

XSLT elements

- xsl:element Создание элемента.
- xsl:for-each Цикл по узлам.
- xsl:if Условная операция.
- xsl:import Импорт XSLT файла.
- xsl:include Включение XSLT файла.
- xsl:key Объявление ключа, для использования функцией key().
- xsl:message Посылка текстового сообщения.
- xsl:namespace-alias Замена префикса пространства имен.
- xsl:number Вставка форматированного числа.
- xsl:otherwise Множественный выбор с <xsl:choose>.
- xsl:output Опции вывода.
- xsl:param Объявление параметра шаблона.

XSLT elements

- xsl:preserve-space Сохранение пробелов.
- xsl:processing-instruction Создание инструкции по обработке.
- xsl:sort Критерий сортировки для <xsl:for-each> или <xsl:apply-templates>.
- xsl:strip-space Удаление пробелов.
- xsl:stylesheet Корневой элемент XSLT файла.
- xsl:template Объявление шаблона.
- xsl:text Создание текста.
- xsl:value-of Вставка значения узла в виде текста.
- xsl:variable Определение значения выражения.
- xsl:when Множественный выбор с <xsl:choose>.
- xsl:with-param Передача параметра шаблону

Элемент <xsl:attribute>

- <xsl:attribute name = "*attribute-name*" namespace = "*uri-reference*"> </xsl:attribute>
- **Количество вхождений –** Не ограничено
- **Родительские элементы -** xsl:copyxsl:copy, xsl:elementxsl:copy, xsl:element, xsl:fallbackxsl:copy, xsl:element, xsl:fallback, xsl:for-eachxsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:ifxsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:messagexsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwisexsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:paramxsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:templatexsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:template, xsl:variablexsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:template, xsl:variable, xsl:whenxsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:template, xsl:variable, xsl:when, xsl:with-param
- **Дочерние элементы -** xsl:apply-importsxsl:apply-imports, xsl:apply-templatesxsl:apply-imports, xsl:apply-templates, xsl:call-templatexsl:apply-imports, xsl:apply-templates, xsl:call-template, xsl:choosexsl:apply-imports, xsl:apply-templates, xsl:call-template, xsl:choose, xsl:copyxsl:apply-imports, xsl:apply-templates, xsl:call-template, xsl:choose, xsl:copy, xsl:copy-ofxsl:apply-imports, xsl:apply-templates,

Элемент <xsl:copy>

- <xsl:copy use-attribute-sets = “QNames” > </xsl:copy>
- **Количество вхождений –** Не ограничено
- **Родительские элементы -** xsl:attributexsl:attribute, xsl:commentxsl:attribute, xsl:comment, xsl:copy, xsl:elementxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallbackxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-eachxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:ifxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:messagexsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwisexsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:paramxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing-instructionxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing-instruction, xsl:templatexsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing-instruction, xsl:template,

Элемент <xsl:copy-of>

- <xsl:copy-of select = Expression/>
- **Количество вхождений –** Не ограничено
- **Родительские элементы** xsl:attributexsl:attribute, xsl:commentxsl:attribute, xsl:comment, xsl:copyxsl:attribute, xsl:comment, xsl:copy, xsl:elementxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallbackxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-eachxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:ifxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:messagexsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwisexsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:paramxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing-instructionxsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing-instruction, xsl:templatexsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing-instruction, xsl:template,

Элемент <xsl:element>

- <xsl:element name = "*element-name*" namespace = "*uri-reference*" use-attribute-sets = QNames> </xsl:element>
- **Родительские элементы** xsl:copyxsl:copy, xsl:element, xsl:fallbackxsl:copy, xsl:element, xsl:fallback, xsl:for-eachxsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:ifxsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:messagexsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwisexsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:paramxsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:templatexsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:template, xsl:variablexsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:template, xsl:variable, xsl:whenxsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:template, xsl:variable, xsl:when, xsl:with-param, output elements
- **Дочерние элементы** xsl:apply-templatesxsl:apply-templates, xsl:attributexsl:apply-templates, xsl:attribute, xsl:call-templatexsl:apply-templates, xsl:attribute, xsl:call-template, xsl:choosexsl:apply-templates, xsl:attribute, xsl:call-template,

Функция current()

- Возвращает набор узлов в составе текущего узла.
- *node-set* current()

- <xsl:value-of select="current()"/>
- <xsl:value-of select="."/>

- <xsl:apply-templates select="//glossary/item[@name=current()/@ref]"/>
- Элементы <item>, у которых атрибут name имеет значение равное значению атрибута ref текущего узла.
- <xsl:apply-templates select="//glossary/item[@name=./@ref]"/>
- Элементы <item>, у которых атрибуты name и ref имеют одинаковое значение.

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Функция document()

- Возможность использовать другие XML ресурсы.
- *node-set* document(*object*, *node-set*?)
- Если только один строковый аргумент, document() считает строку URL и возвращает документ, как набор узлов.
- Если только один аргумент типа node set, тогда каждый узел считается URL и функция возвращает объединение всех ссылаемых документов.
- Если два аргумента, причем первый строка или node set, а второй - node set. Второй аргумент при использовании служит для указания базового URL, тогда первые аргументы относительны этого URL.
- Если функции передается пустая строка, то результат - исходный XML, если не задан не нулевой второй аргумент.
- И в последнем случае URL документа является базовым для узлов, содержащихся во втором аргументе.
- **Пример**
- <xsl:template match="groupRef">
- <xsl:copy-of select="**document(@href)//group**"/>
- </xsl:template>

Vested Development Inc.
5 New England Executive Park, 1st Floor, Burlington, MA 01803
Phone: 781.425.6005, Fax: 781.425.6333
www.vdiweb.com

Типы данных

- Строковый тип (String) – любая последовательность символов Unicode, разрешенных в XML
- Числовой тип (Number) – число с плавающей точкой двойной точности, как определено в IEEE 754
- Логический тип (Boolean) – принимает значение истина или ложь
- Набор узлов (Node-set) – набор узлов в исходном дереве
- Внешний объект (External object) – объект возвращаемый функцией расширения
- Фрагмент конечного дерева (Result Tree Fragment)

Новые возможности XSLT 2.0

- Система типов основанная на последовательностях узлов.
- Поддержка типов данных XML схем.
- Поддержка группировки узлов
- Агрегатные функции
- Циклы "For"
- Поддержка условных выражений
- Поддержка регулярных выражений
- Множественные выходные документы
- XHTML вывод
- Пользовательские функции
- Новые функции и операции