Курс «Языки разметки»

Лекция 3.
Язык XSL

Содержание

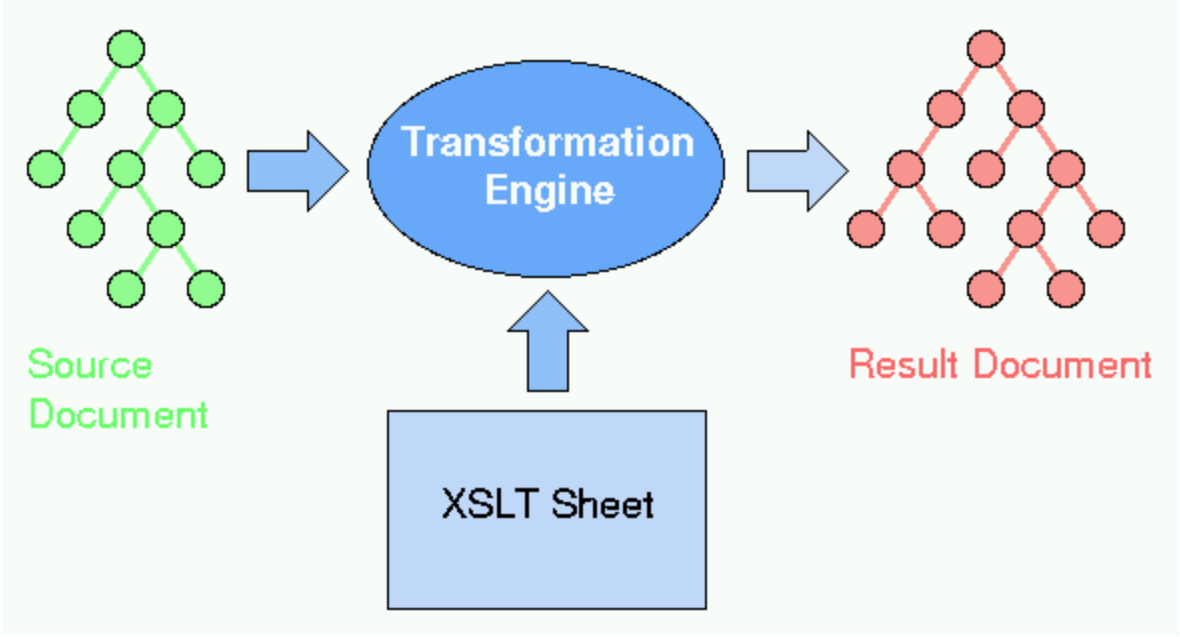
- Навигация по документу. Язык XPath
- **Введение в XSL**

XSL (eXtensible Stylesheet Language)

- **XSL** — семейство рекомендаций консорциума W3C, описывающее языки преобразования и визуализации XML-документов.
- Состоит из трех частей:
  - **XSL Transformations (XSLT)** — язык преобразований XML-документов.
  - **XSL Formatting Objects (XSL-FO)** — язык разметки типографских макетов и иных предпечатных материалов.
  - **XPath**
- XSLT и XPath были предложены в виде двух отдельных официальных рекомендаций W3C в1999 г. Для языка XSL-FO нет отдельной рекомендации.

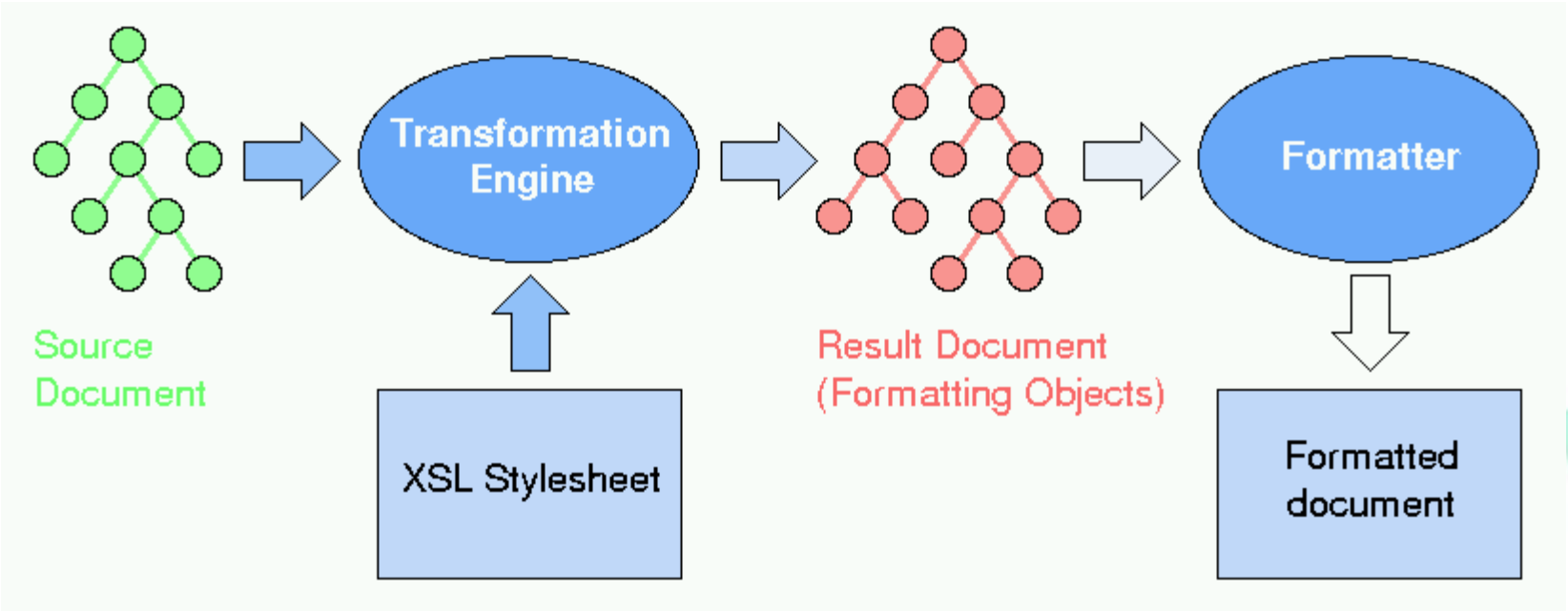
XSL-преобразования (XSLT)

- XSL использует XSLT для преобразования XML-дерева-оригинала в XML-дерево-результат.

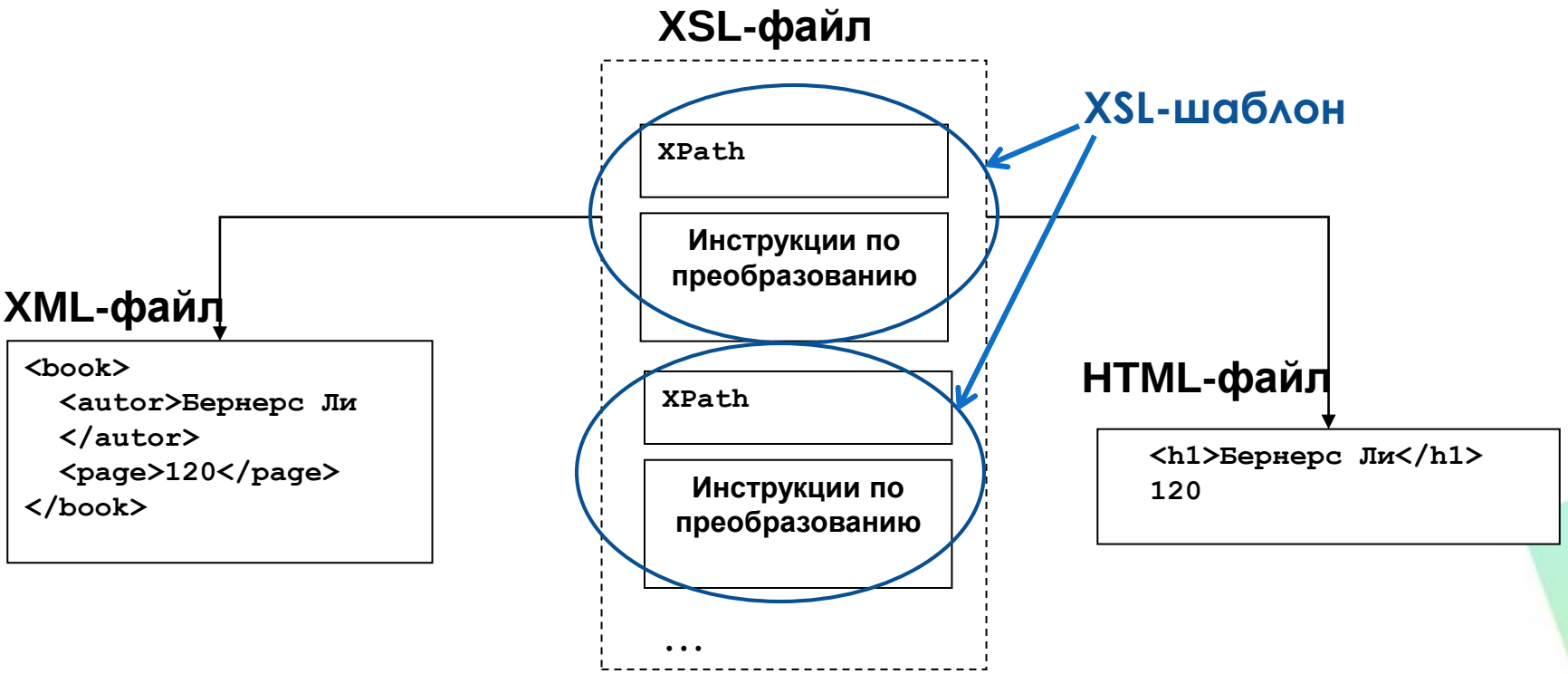


XSL-форматирование (XSL-FO)

- XSL использует XSLT для преобразования XML-дерева-оригинала в XML-дерево-результат.



Как происходит XSL-трансформация?



XSL-трансформация

XML-файл: cd.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"
   href="cd.xsl"?>
<catalog>
  <cd>
    <title>empire burlesque</title>
    <artist>bob dylan</artist>
</cd>
  ...
</catalog>
```

XSL-файл: cd.xsl

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

XSL-шаблоны

- Задаются элементами **xsl:template**
- Каждый шаблон содержит:
  - Адрес узлов (на XPath), на которые действует данный шаблон, записывается в атрибут **match** элемента xsl:template
  - Инструкции по преобразованию
- Пример:

```
<xsl:template match="путь">
  инструкции по преобразованию
</xsl:template>
```

XSL-шаблоны. Пример

```
<xsl:template match="title">
<h1>Это заголовок</h1>
</xsl:template>


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

Обход XML-дерева

- Начинается с шаблона, описывающего корень документа

XML

```
<?xml version="1.0"?>
<?xml-stylesheet ... ?>
<catalog>
<cd>
  <title>Empire Burlesgue
  </title>
  <artist>Bob Dylan</artist>
</cd>
<cd>
  <title>Live A Paris</title>
  <artist>Celin Dion</artist>
</cd>
</catalog>
```

XSL

```
...
<xsl:template match="/">
<html>
<body>
</body>
</html>
</xsl:template>
```

HTML

```
<html>
<body>
</body>
</html>
```

Обход XML-дерева

- Чтобы зайти в дерево глубже его корня, нужно дать указание преобразователю обрабатывать дочерние узлы корневого узла. Элемент, предназначенный для этого, **xsl:apply-templates**.
- **xsl:apply-templates** выбирает все дочерние элементы текущего узла.

XML

```
<?xml version="1.0"?>
<?xml-stylesheet ... ?>
<catalog>
<cd>
  <title>Empire Burlesgue
  </title>
  <artist>Bob Dylan</artist>
</cd>
<cd>
  <title>Live A Paris</title>
  <artist>Celin Dion</artist>
</cd>
</catalog>
```

XSL

```
<xsl:template match="/">
<html><body>
<xsl:apply-templates />
</body></html>
</xsl:template>
```

HTML

```
<html>
<body>
Empire Burlesgue Bob
Dylan Live A Paris
Celin Dion
</body>
</html>
```

Обход XML-дерева

- Далее мы можем писать шаблон для любого другого элемента, для обхода дочерних узлов которого также можно использовать xsl:apply-templates.

XML

```
<?xml version="1.0"?>
<?xml-stylesheet ... ?>
<catalog>
<cd>
  <title>Empire Burlesgue
  </title>
  <artist>Bob Dylan</artist>
</cd>
<cd>
  <title>Live A Paris</title>
  <artist>Celin Dion</artist>
</cd>
</catalog>
```

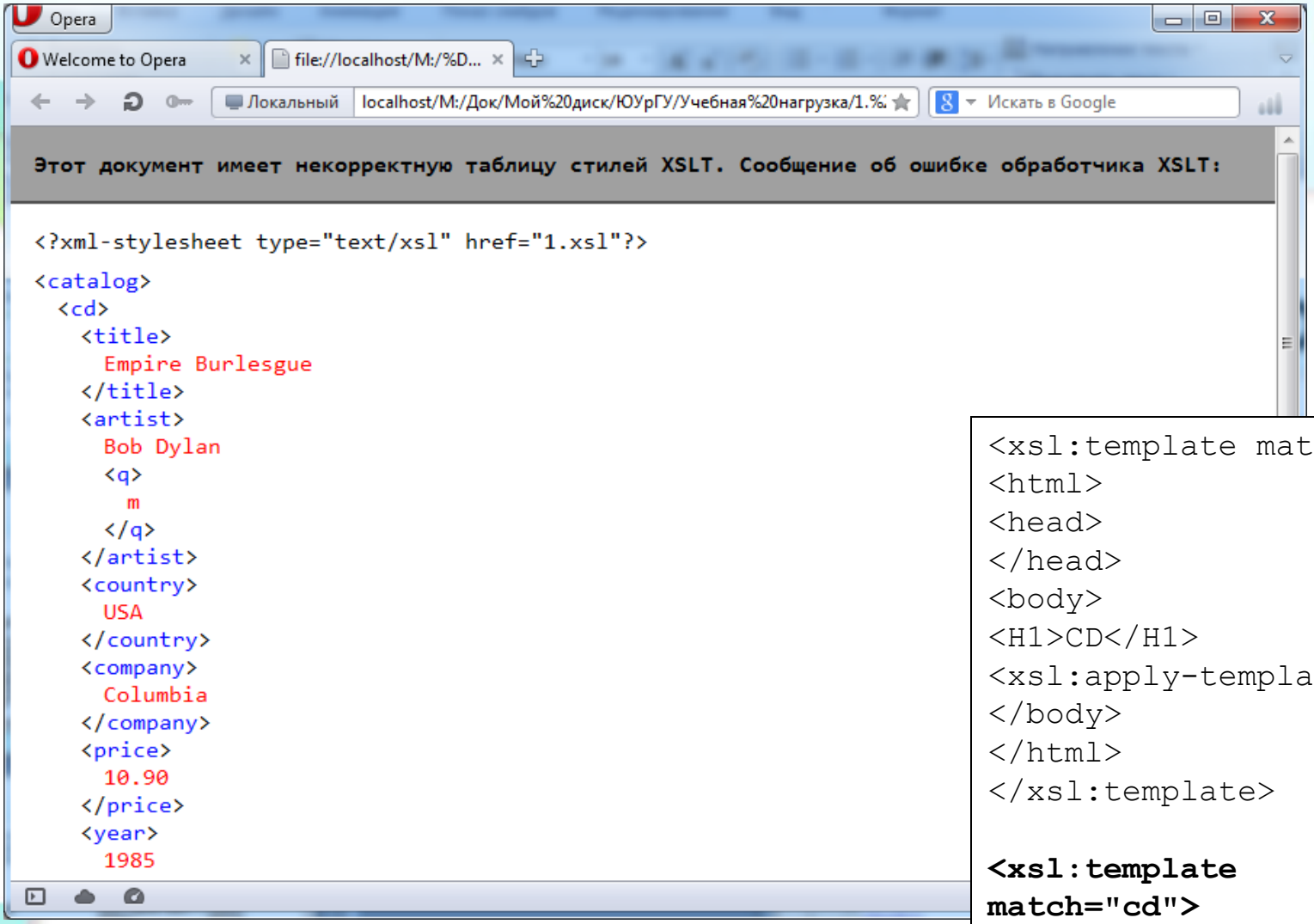
XSL

```
<xsl:template match="/">
<html><body>
<h1>CD</h1>
<xsl:apply-templates />
</body></html>
</xsl:template>

<xsl:template
match="cd">
Это элемент cd.
</xsl:template>
```

HTML

```
<html>
<body>
<h1>CD</h1>
Это элемент cd. Это
элемент cd.
</body>
</html>
```




```
<xsl:template match="/">
<html>
<head>
</head>
<body>
<H1>CD</H1>
<xsl:apply-templates />
</body>
</html>
</xsl:template>

<xsl:template
match="cd">
Это элемент cd.<br>
</xsl:template>
```

Обход XML-дерева

1. Корневой узел сравнивается со всеми правилами-шаблонами таблицы стилей. Находится соответствующий шаблон. На выход передается **<html>**
2. Элемент xsl:apply-templates указывает преобразователю обрабатывать дочерние узлы корневого узла. Первый дочерний узел корня документа инструкция xml-stylesheet.
3. Второй дочерний узел корневого узла – корневой элемент catalog. На выход передается **<body>**
4. xsl:apply-templates внутри body заставляет преобразователь обрабатывать дочерние узлы элемента catalog.
5. Первый дочерний элемент элемента catalog, элемент cd, описывающий альбом Empire Burlesgue, сравнивается с правилами-шаблонами. На выход передается текст **Это элемент cd.
**.
6. Второй дочерний элемент элемента catalog элемент cd сравнивается с правилами-шаблонами. На выход передается текст **Это элемент cd.
**.
7. На выход передается тэг **</body>**.
8. На выход передается тэг **</html>**.
9. Обработка завершена.

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

<catalog>
<cd>
  <title>Empire Burlesgue
  </title>
  <artist>Bob Dylan</artist>
</cd>
<cd>
  <title>Live A Paris</title>
  <artist>Celin Dion</artist>
</cd>
</catalog>
```

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

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

<xsl:template match="cd">
Это элемент cd.<br/>
</xsl:template>
```

xsl:apply-templates. Атрибут select

- Чтобы отобрать не все дочерние узлы данного узла, а лишь некоторый их набор, нужно снабдить xsl:apply-templates атрибутом select, выделяющим нужные дочерние узлы

XML

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

<catalog>
<cd>
  <title>Empire Burlesgue
  </title>
  <artist>Bob Dylan</artist>
</cd>
<cd>
  <title>Live A Paris</title>
  <artist>Celin Dion</artist>
</cd>
</catalog>
```

XSL

```
<xsl:template
match="/catalog">
<html><body>
<xsl:apply-templates />
</body></html>
</xsl:template>

<xsl:template
match="cd">
<h1>CD</h1>
<xsl:apply-templates
select="title"/>
</xsl:template>
```

HTML

```
<html>
<body>
<h1>CD</h1>
Empire Burlesgue
<h1>CD</h1>
Live A Paris
</body>
</html>
```

Элемент xsl:value-of

- Элемент xsl:value-of вычисляет некоторое значение (обычно, хотя и не всегда, на основе данных исходного документа) и копирует полученный результат в выходной документ. Атрибут select элемента xsl:value-of определяет, какое именно значение вычисляется.

XML

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

<catalog>
<cd>
  <title>Empire Burlesgue
  </title>
  <artist>Bob Dylan</artist>
</cd>
<cd>
  <title>Live A Paris</title>
  <artist>Celin Dion</artist>
</cd>
</catalog>
```

XSL

```
<xsl:template
match="/catalog">
<html><body>
<xsl:apply-templates />
</body></html>
</xsl:template>

<xsl:template
match="cd">
<h1>CD</h1>
<xsl:value-of
select="title"/>
</xsl:template>
```

HTML

```
<html>
<body>
<h1>CD</h1>
Empire Burlesgue
<h1>CD</h1>
Live A Paris
</body>
</html>
```

Обработка множественных элементов с помощью xsl:for-each

- Элемент xsl:value-of следует использовать только в тех случаях, когда очевидно, значение какого именно узла берется. Если может быть выбрано множество пунктов, тогда будет выбран лишь первый из них.
- Существует два способа по очереди обработать множественные элементы:
  - xsl:apply-templates с атрибутом select
  - использование xsl:for-each
- Элемент xsl:for-each обрабатывает по очереди каждый элемент, выбранный атрибутом select. Дополнительные шаблоны в данном случае не требуются. Например:

```
<xsl:template match="/catalog">
  <xsl:for-each select="cd">
    <xsl:value-of select="year"/>
  </xsl:for-each>
</xsl:template>
```

Сортировка выходных элементов

- Элемент xsl:sort сортирует выходные элементы.
- Элемент xsl:sort должен быть дочерним элементом элементов xsl:apply-templates или xsl:for-each.
- Атрибут select элемента xsl:sort задает ключ, используемый для сортировки элементов.

```
<xsl:template match="/catalog">
  <xsl:for-each select="cd">
    <xsl:sort  select="title”/>
</xsl:for-each>
</xsl:template>
```

- Если в данном элементе xsl:apply-templates или xsl:for-each присутствует несколько элементов xsl:sort, тогда элементы сортируются сначала по первому ключу, затем по второму ключу, и так далее.

Сортировка выходных элементов

- По умолчанию сортировка ведется в алфавитном порядке ключей.
- Обратная сортировка:

```
<xsl:sort order="descending" data-type="number"
select="title"/>
```

Дефолтные шаблонные правила

- **Первое дефолтное** правило:

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

- **Второе дефолтное правило:**

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

- **Дефолтное правило для инструкций по обработке и комментариев:**

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

- Задача этих правила - обеспечить рекурсивную обработку всех элементов, даже если к ним не применяется ни одно другое явно заданное правило
- Имеют низкий приоритет, заменяются авторскими правилами

Второе дефолтное правило

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

XML

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

<catalog>
<cd id="1">
  <title>Empire Burlesgue
  </title>
  <artist>Bob Dylan</artist>
</cd>
<cd id="2">
  <title>Live A Paris</title>
  <artist>Celin Dion</artist>
</cd>
</catalog>
```

XSL

```
<xsl:template match="/">
<html><body>
<xsl:apply-templates />
</body></html>
</xsl:template>
```

HTML

```
<html>
<body>
Empire Burlesgue Bob
Dylan Live A Paris
Celin Dion
</body>
</html>
```

Значения атрибутов не выведены! Почему?

Второе дефолтное правило

- Дополнительное правило для вывода значений атрибутов:

```
<xsl:template match="*[@*]">
  <xsl:value-of select="."/>
  <xsl:apply-templates select="@*"/>
</xsl:template>
```

Атрибуты

- Хотим получить код:

```
<img src="1.jpg">
```

- НЕВЕРНЫЙ:

```
<xsl:template match="cd">
  <img src="<xsl:value-of select="logo"/>" />
</xsl:template>
```

- ВЕРНЫЙ:

1.
```
<img>
   <xsl:attribute name="src">
         <xsl:value-of select="logo" />
   </xsl:attribute>
</img>
```
2.
```
<xsl:template match="cd">
  <img src="{logo}" />
</xsl:template>
```

Создание текста с помощью xsl:text

- Элемент xsl:text вставляет в выходной поток свое содержимое в качестве текста:

```
<xsl:template match="cd">
  <xsl:text>Это элемент cd.</xsl:text>
</xsl:template>
```

- Преимущество xsl:text:
  - сохраняет пробелы, переносы и табуляцию

Организация ветвления

- Элемент xsl:if позволяет выводить или не выводить определенный фрагмент XML
- Элемент xsl:choose берет один из нескольких возможных XML-фрагментов
- Пример:

```
<xsl:template match="cd">
  <xsl:value-of select="title"/>
  <xsl:if test="position()!=last()">,
  </xsl:if>
</xsl:template>
```

Именованный шаблон

- Шаблону можно назначить имя:

```
<xsl:template name="имя">
...
</xsl:template>
```

- Вызов именованного шаблона:

```
<xsl:call-template name="имя">
...
</xsl:call-template>
```

- Вызов именованного шаблона происходит быстрее, чем обычного

Вычисление выражений

- Именованный шаблон можно вызывать вне зависимости от того, какая часть документа обрабатывается в данный момент.
- Именованному шаблону можно передавать параметры.
- Результат выполнения именованного шаблона можно присваивать переменной.

```
<xsl:template name="sqr">
<xsl:param name="x"/>
<xsl:value-of select="$x * $x"/>
</xsl:template>

<xsl:variable name="y">
<xsl:call-template name="sqr">
<xsl:with-param name="x" select="6"/>
</xsl:call-template>
</xsl:variable>
```

```
<xsl:value-of select="$y"/>
// выведет 36
```