# СТАНДАРТ XML

E<u>X</u>tensible <u>M</u>arkup <u>L</u>anguage

# ВОПРОСЫ

Определение XML
Применение XML

Пример XML-документа

Правила создания XML-документа

DOM XML-документа, виды узлов

Валидация XML-документа, способы контроля.

DTD- описание, пример.

XML и CSS.

XML и XLS (дополнительно).

# Расширяемый язык разметки XML

- Это производный язык разметки документов, позволяющий структурировать информацию разного типа.
- XML является стандартом создания собственного языка разметки, позволяет создавать собственные теги и использовать их на странице
- XML можно определить как систему, управляющую данными
- XML — это текстовый формат, удобный для хранения и передачи структурированных данных.

# ИСТОРИЯ XML

- 1969 год - разработан GML (Generalized Markup Language), который состоял из слов, описывающих части официального документа
- 1974 год - GML был преобразован в новый универсальный язык разметки SGML, который был принят как стандарт в электронном издательстве (ISO 8879). SGML был очень сложным, после его упрощения был создан HTML.
- 1998 год - консорциум Всемирной сети (World Wide Web Consortium) принял стандарт XML

# ПРИМЕНЕНИЕ ЯЗЫКА XML

1. для разработчиков сложных **информационных систем**, с большим количеством приложений, связанных потоками информации самой различной структурой. В этом случае XML - документы выполняют роль универсального формата для обмена информацией между отдельными компонентами большой программы.

# ПРИМЕНЕНИЕ ЯЗЫКА XML

2. XML является базовым стандартом для **языка описания ресурсов**, RDF, позволяющего упростить многие проблемы в Web, связанные с поиском нужной информации, обеспечением контроля за содержимым сетевых ресурсов, создания электронных библиотек и т.д.

# ПРИМЕНЕНИЕ ЯЗЫКА XML

3. Язык XML позволяет описывать данные произвольного типа и используется для представления специализированной информации, например химических, математических, физических формул, медицинских рецептов, нотных записей, и т.д. Это означает, что XML может служить мощным дополнением к HTML для распространения в Web "нестандартной" информации. Возможно, в самом ближайшем будущем XML полностью заменит собой HTML, по крайней мере, первые попытки интеграции этих двух языков уже делаются (спецификация XHTML).

# ПРИМЕНЕНИЕ ЯЗЫКА XML

4. XML-документы могут использоваться в качестве **промежуточного формата данных в трехзвенных системах.** Обычно схема взаимодействия между серверами приложений и баз данных зависит от конкретной СУБД и диалекта SQL, используемого для доступа к данным. Если же результаты запроса будут представлены в некотором универсальном текстовом формате, то звено СУБД, как таковое, станет "прозрачным" для приложения.

# ПРИМЕНЕНИЕ ЯЗЫКА XML

5. Информация, содержащаяся в XML-документах, может изменяться, передаваться на машину клиента и обновляться по частям. Разрабатываемые спецификации XLink и Xpointer поволят ссылаться на отдельные элементы документа, с учетом их вложенности и значений атрибутов.

# ПРИМЕНЕНИЕ ЯЗЫКА XML

6. Использование **стилевых таблиц (XSL)** позволяет обеспечить независимое от конкретного устройства вывода отображение XML- документов.

# ПРИМЕНЕНИЕ ЯЗЫКА XML

7. XML может использоваться в **обычных приложениях** для хранения и обработки структурированных данных в едином формате.

# ЧТО ТАКОЕ XML?

- XML-документ представляет собой обычный текстовый файл, в котором при помощи специальных маркеров создаются элементы данных, последовательность и вложенность которых определяет структуру документа и его содержание.

# ЭЛЕМЕНТ

Имя элемента
Имя элемента
HTML-элемент
<H2>Web Site Contents</H2>
Начальный тег
Содержание
Конечный тег

# XML И HTML

- XML, безусловно, сильно отличается по своим возможностям и предназначению от языка гипертекстовой разметки, оба эти языка являются подмножествами SGML, и, следовательно, наследуют его базовые принципы.

# СОЗДАНИЕ XML ДОКУМЕНТА

```
<?xml version="1.0"?>
<INVENTORY>
  <BOOK>
    <TITLE>Мастер и маргарита</TITLE>
    <AUTHOR>Михаил Булгаков</AUTHOR>
    <BINDING>Ленинградское издательство (Лениздат)</BINDING>
    <PAGES>512</PAGES>
    <PRICE>74р.</PRICE>
  </BOOK>
  <BOOK>
    <TITLE>Ревизор</TITLE>
    <AUTHOR>Николай Гоголь</AUTHOR>
    <BINDING>Искатель</BINDING>
    <PAGES>80</PAGES>
    <PRICE>37р.</PRICE>
  </BOOK>
</INVENTORY>
```

# ТЕЛО ДОКУМЕНТА XML

- состоит из
  - элементов разметки (markup);
  - содержимого документа - данных (content).
- XML - тэги предназначены для определения элементов документа, их атрибутов и других конструкций языка.

# ИНСТРУКЦИЯ <?XML?>

- Любой XML- документ должен всегда начинаться с инструкции <?xml?>, внутри которой также можно задавать номер версии языка, номер кодовой страницы и другие параметры, необходимые программе-анализатору в процессе разбора документа
- Первая строка XML-документа называется объявлением XML (англ. XML declaration) — это необязательная строка, указывающая версию стандарта XML, также здесь может быть указана кодировка символов и внешние зависимости.

# ПРАВИЛА СОЗДАНИЯ XML- ДОКУМЕНТА

- **Документ должен иметь только один элемент верхнего уровня** (элемент Документ или корневой элемент). Все другие элементы должны быть вложены в элемент верхнего уровня.
- **Элементы должны быть вложены упорядоченным образом.** То есть, если элемент начинается внутри другого элемента, он должен и заканчиваться внутри этого элемента.
- **Каждый элемент должен иметь начальный и конечный тег.** В отличие от HTML, в XML не разрешается опускать конечный тег – даже в том случае, когда браузер в состоянии определить, где заканчивается элемент.
- **Имя типа элемента в начальном теге должно в точности соответствовать имени в соответствующем конечном теге.**
- **Имена типов элементов чувствительны к регистру, в котором они набраны.** В действительности весь текст внутри XML-разметки является чувствительным к регистру.

# КОНСТРУКЦИИ ЯЗЫКА

- Содержимое XML- документа представляет собой набор элементов, секций CDATA, директив анализатора, комментариев, спецсимволов, текстовых данных.

# XML-ДОКУМЕНТ СОСТОИТ ИЗ ДВУХ ОСНОВНЫХ ЧАСТЕЙ:

- пролога и элемента Документ

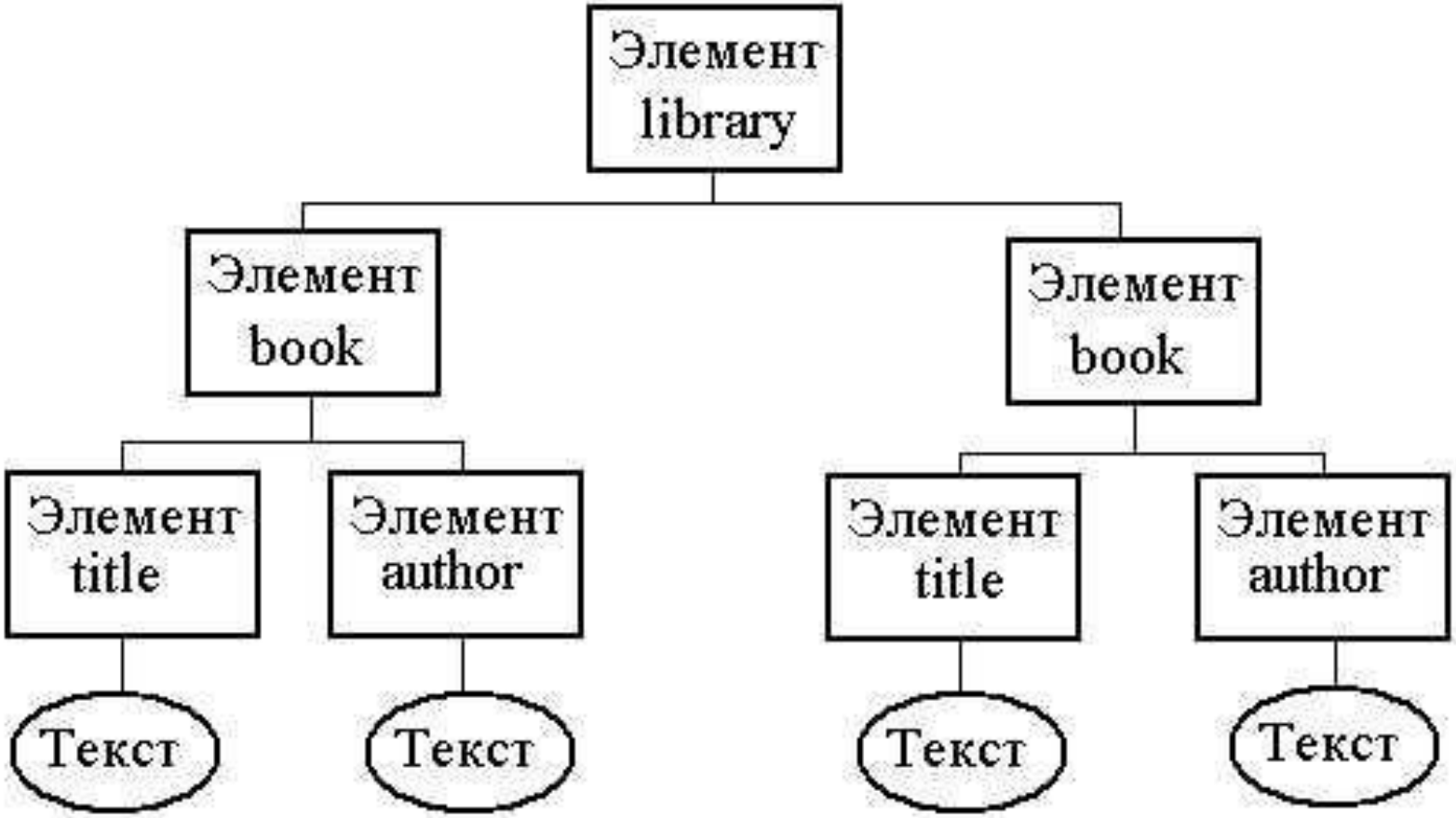
# ОБЪЕКТНАЯ МОДЕЛЬ XML

- Объектная модель документа (DOM - Document Object Model) является независимым от платформы реализации языком.
- С точки зрения программиста он представляется в виде Интерфейса Прикладного Программирования (API - Application Programming Interface), который предоставляет программам доступ и манипулирование содержанием и структурой XML.

# ПРИМЕР

```
<library>
  <book>
      <title>Программирование</title>
      <author>Иванов И.И.</author>
  </book>
  <book>
      <title>Информатика</title>
      <author>Петров П.П.</author>
  </book>
</library>
```

# УЗЛЫ ДЕРЕВА ПРИМЕРА



# ВИДЫ УЗЛОВ

- *Корневой узел* – самый верхний узел дерева, соответствующий корневому элементу XML документа.
- *Родительский узел (parent node)* – узел из которого наследуются узлы более низкого уровня дерева. Соответствует понятию контейнера в структуре XML документа.
- *Дочерний узел (child node)* – узел, который наследуется из узла более высокого уровня. Соответствует, в структуре XML документа, понятию вложенного в контейнер элемента.
- *Узлы братья (siblings)* – одноуровневые узлы, принадлежащие одному родителю. В структуре XML документа это узлы непосредственно вложенные в контейнер.

- Для того чтобы построить дерево XML документа, он должен быть обработан анализатором

# ЭЛЕМЕНТЫ ДАННЫХ

- **Элемент** - это структурная единица XML-документа.

В общем случае в качестве содержимого элементов могут выступать как просто какой-то текст, так и другие, вложенные, элементы документа, секции CDATA, инструкции по обработке, комментарии, - т.е. практически любые части XML- документа.

Набором всех элементов, содержащихся в документе, задается его **структура** и определяются все иерархическое соотношения.

# СПЕЦИАЛЬНЫЕ СИМВОЛЫ

- Для того, чтобы включить в документ символ, используемый для определения каких-либо конструкций языка (например, символ угловой скобки) и не вызвать при этом ошибок в процессе разбора такого документа, нужно использовать его специальный символьный либо числовой идентификатор.
- Например, < , > " или $(десятичная форма записи), &#x1a (шестнадцатеричная) и т.д.
- Строковые обозначения спецсиволов могут определяться в XML документе при помощи компонентов (entity).

# Альтернативные сочетания символов

| замена | символ |
|---|---|
| < | < |
| > | > |
| & | & |
| ' | ' |
| " | " |

# КОММЕНТАРИИ

- Комментариями является любая область данных, заключенная между последовательностями символов <!-- и -->
- Комментарии пропускаются анализатором и поэтому при разборе структуры документа в качестве значащей информации не рассматриваются.

# ОТЛИЧИЕ ОТ HTML

- В отличие от HTML, который игнорирует повторы пробелов, XML сохраняет полную длину строки из пробелов.

# АТРИБУТЫ

- Атрибут - это пара "название" = "значение", которую надо задавать при определении элемента в начальном тэге.

Пример:

- <color **RGB**="true">#ff08ff</color>
  <color **RGB**="false">white</color>

или

- <author id=0>Ivan Petrov</author>

# ДИРЕКТИВЫ АНАЛИЗАТОРА

- Инструкции, предназначенные для анализаторов языка, описываются в XML документе при помощи специальных тэгов - <? и ?>;.
- Программа клиента использует эти инструкции для управления процессом разбора документа.
- Наиболее часто инструкции используются при определении типа документа (например, <?xml version=”1.1”?>) или создании пространства имен.

# CDATA

- Внутри этого блока можно помещать любую информацию, которая может понадобится программе- клиенту для выполнения каких-либо действий (в область CDATA, можно помещать, например, инструкции JavaScript).
- Необходимо следить за тем, чтобы в области, ограниченной этими тэгами не было последовательности символов ]].

# ВАЛИДАЦИЯ XML-ДОКУМЕНТА

- Если XML- документ не нарушает правила построения, то он называется *формально-правильным* и все анализаторы, предназначенные для разбора XML-документов, смогут работать с ним корректно.

# Способы контроля правильности XML-документа

- DTD – определения (Document Type Definition)
- Схемы данных(Semantic Schema)

# ДОБАВЛЕНИЕ DTD

- Объявление типа документа представляет собой блок XML-разметки, который вы должны добавить в пролог валидного XML-документа. Он может располагаться в любом месте пролога – вне другой разметки – после XML-объявления
- DTD состоит из символа левой квадратной скобки ([), после которой следует ряд объявлений разметки, заканчивающихся правой квадратной скобкой (]).

# DTD МОЖЕТ СОДЕРЖАТЬ СЛЕДУЮЩИЕ ТИПЫ ОБЪЯВЛЕНИЙ РАЗМЕТКИ

- **Объявления типов элементов.** Они определяют типы элементов, которые может содержать документ, а также содержимое и порядок следования элементов.
- **Объявления списков атрибутов.** Каждое объявление списков атрибутов задает имена атрибутов, которые могут быть использованы с определенным типом элемента, а также типы данных и устанавливаемые по умолчанию значения этих атрибутов.
- **Объявления примитивов.** Вы можете использовать примитивы для хранения часто используемых фрагментов текста или для встраивания не относящихся к XML данных в ваш документ.
- **Объявления нотаций.** Нотация описывает формат данных или идентифицирует программу, используемую для обработки определенного формата.
- **Инструкции по обработке.**
- **Комментарии.**
- **Ссылки на параметрические примитивы.** Любой из приведенных выше компонентов может содержаться внутри параметрического примитива и добавляться путем ссылки на параметрический примитив.

Объявление разметки определяет тип элемента

```
<?xml version="1.0"?>

<!DOCTYPE SIMPLE
  [
  <!ELEMENT SIMPLE ANY>
  ]
>

<SIMPLE> This is an extremely simplistic XML document. </SIMPLE>
```

DTD

Объявление типа документа

# БЛОКИ DTD

- *Элементы* (Elements). Элементы могут содержать текст, другие элементы или быть пустыми.
- *Атрибуты* (Attributes). Атрибуты всегда размещаются внутри открывающего тэга элемента. Атрибуты всегда записываются в виде пары *имя/значение*.
- *Entities*.
- *PCDATA*. Означает структурированные символьные данные. Подлежит анализу с помощью парсера на наличие специальных символов (entities) и элементов разметки.
- *CDATA*. Означает текст, содержимое которого не рассматривается анализатором

# Операторы, задающие множественность вхождения

- ‘**+**’: один и более раз, например, *<!ELEMENT note (message+)>*
- ‘*****’: ноль и более раз, например, *<!ELEMENT note (message*)>*
- ‘**?**’: ноль и один раз, например, *<!ELEMENT note (message?)>*

# ОБЪЯВЛЕНИЕ ВИДА <!ELEMENT NOTE (MESSAGE|BODY)>

- указывает на возможность вхождения в элемент *note* другого элемента: *message* ИЛИ *body*.

# ОПИСАНИЕ АТРИБУТОВ

- <!ATTLIST element-name attribute-name attribute-type default-value>
- например:

<!ATTLIST payment type CDATA "check">

- Соответствует XML коду:

<payment type="check" />.

# ТИП АТРИБУТА

| Тип | Описание |
|---|---|
| CDATA | Символьные данные |
| (*en1*\|*en2*\|..) | Значение из списка |
| ID | Уникальный *id* |
| IDREF | *id* другого элемента |
| IDREFS | Список других *id* |
| NMTOKEN | Допустимое XML имя |
| NMTOKENS | Список допустимых XML имен |
| ENTITY | Специальные символы |
| ENTITIES | Список специальных символов |
| NOTATION | Имя нотации |
| xml: | Предопределенное XML имя |

# Значение по умолчанию может быть одним из следующих

| Значение | Интерпретация |
|---|---|
| #REQUIRED | Обязательно |
| #IMPLIED | Не обязательно |
| #FIXED *value* | Значение фиксировано |

# Стандартом определены два уровня правильности документа XML:

- **Правильно построенный** (Well-formed). Правильно построенный документ соответствует всем правилам синтаксиса XML.
- **Действительный** (Valid). Действительный документ дополнительно соответствует некоторым семантическим правилам. Это более строгая дополнительная проверка корректности документа на соответствие заранее определённым, но уже внешним правилам, в целях минимизации количества ошибок, например, структуры и состава данного, конкретного документа или семейства документов.

# Действительные XML документы содержат ссылку на DTD файл

- *<!DOCTYPE note SYSTEM "mail.dtd">*

# DTD CXEMA

```
<!DOCTYPE mail [
  <!ELEMENT note (to,from,subject,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT subject (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
```

# ПРОВЕРКА ВАЛИДНОСТИ

- использовать специальные валидаторы, например *W3C валидатор* (http://validator.w3.org/).
- Для проверки схем также существуют специальные валидаторы
- Согласно спецификации W3C XML программа должна прекратить обработку XML документа, как только будет обнаружена ошибка в этом документе

# DTD СХЕМА

- DTD (Document Type Definition) определяет допустимые строительные блоки XML документа, путем указания списка допустимых элементов и атрибутов.
- DTD может описываться как внутри XML документа, так и с помощью внешней ссылки.

# ПРИМЕР ВНУТРЕННЕГО ОПИСАНИЯ

```
<?xml version="1.1"?>
<!DOCTYPE note [
  <!ELEMENT mail (to,from,subject,body)>
  <!ELEMENT to      (#PCDATA)>
  <!ELEMENT from    (#PCDATA)>
  <!ELEMENT subject (#PCDATA)>
  <!ELEMENT body    (#PCDATA)>
]>
<note>
<mail>
  <to>user1@domain.ru</to>
  <from>user2@domain.ru</from>
  < subject >Встреча</subject>
  <body> Позвони мне завтра утром </body>
</mail>
</note>
```

# В ДАННОМ ПРИМЕРЕ:

- **!DOCTYPE mail** определяет корневой элемент документа **mail.**
- **!ELEMENT note** определяет элемент **note,** который содержит четыре элемента: "*to*, *from*, *subject*, *body*".
- **!ELEMENT to** определяет элемент **to** типа "#PCDATA".
- **!ELEMENT from** определяет элемент **from** типа "#PCDATA".
- **!ELEMENT subject** определяет элемент **subject** типа "#PCDATA".
- **!ELEMENT body** определяет элемент **body** типа "#PCDATA"

# ПРИМЕР ВНЕШНЕГО ОПИСАНИЯ

```
<?xml version="1.1"?>
<!DOCTYPE note SYSTEM "mail.dtd">
<note>
<mail>
  <to>user1@domain.ru</to>
  <from>user2@domain.ru</from>
  <subject>Встреча</subject>
  <body> Позвони мне завтра утром </body>
</mail>
</note>
```

# Для чего необходимо использование DTD

- С помощью DTD XML файлы могут содержать описание собственного формата.
- Независимые группы людей могут обмениваться данными.
- DTD-схема может быть использована для проверки действительности, как документов получаемых извне, так и собственных документов

# ПРОСМОТР XML - ДОКУМЕНТОВ

- В отличии от HTML, XML никак не определяет способ отображения и использования описываемых с его помощью элементов документа, т.е. программе-анализатору предоставляется возможность самой выбирать нужное оформление.

# ОТОБРАЖЕНИЕ XML-ДОКУМЕНТОВ

- **Таблица стилей.** С помощью данного метода вы связываете таблицу стилей с XML-документом. Таблица стилей представляет собой отдельный файл, содержащий инструкции для форматирования индивидуальных XML-элементов.
  - Каскадная таблица стилей (Cascading Style Sheet – CSS);
  - Расширяемая таблица в формате языка стилевых таблиц (Extensible Stylesheet Language – XSL.

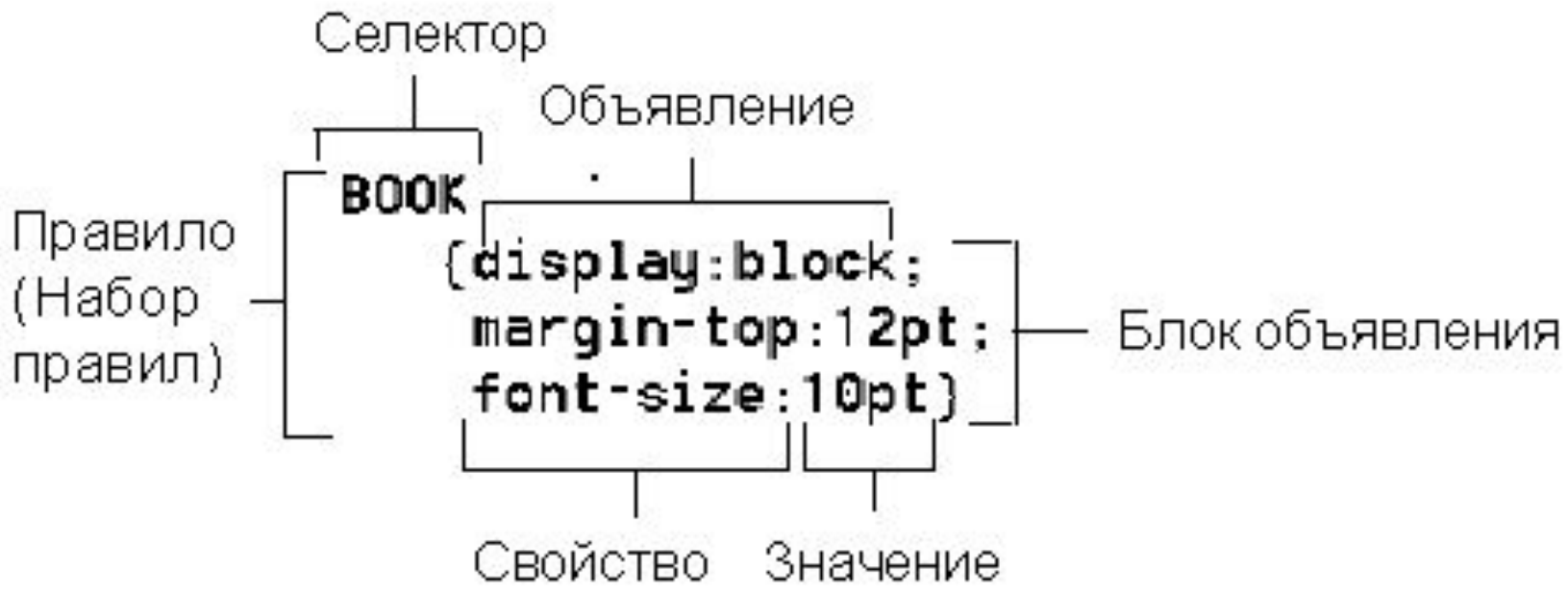
# ОТОБРАЖЕНИЕ XML-ДОКУМЕНТОВ С ИСПОЛЬЗОВАНИЕМ ТАБЛИЦ КАСКАДНЫХ СТИЛЕЙ

- Создание файла таблицы стилей
- Связывание таблицы стилей с XML-документом

- Таблица стилей состоит из одного или нескольких правил (иногда их называют набором правил). Правило содержит информацию по отображению определенного типа элемента в XML-документе

# СЕЛЕКТОР

Селектор представляет собой имя типа элемента, к которому относится информация по отображению



# ПРИМЕР

```
<?xml-stylesheet type="text/css" href="2.css"?>
<!-- File Name: 2.xml -->
<!DOCTYPE INVENTORY[
<!ELEMENT INVENTORY (BOOK*)>
<!ELEMENT BOOK
(TITLE,AUTHOR,BINDING,PAGES,PRICE)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT AUTHOR (#PCDATA)>
<!ELEMENT BINDING (#PCDATA)>
<!ELEMENT PAGES (#PCDATA)>
<!ELEMENT PRICE (#PCDATA)>
]>
```

# ПРИМЕР (ПРОДОЛЖЕНИЕ)

```
<INVENTORY>
  <BOOK>
    <TITLE>Мастер и маргарита</TITLE>
    <AUTHOR>Михаил Булгаков</AUTHOR>
    <BINDING>Ленинградское издательство (Лениздат)</BINDING>
    <PAGES>512</PAGES>
    <PRICE>74р.</PRICE>
  </BOOK>
  <BOOK>
    <TITLE>Ревизор</TITLE>
    <AUTHOR>Николай Гоголь</AUTHOR>
    <BINDING>Искатель</BINDING>
    <PAGES>80</PAGES>
    <PRICE>37р.</PRICE>
  </BOOK>
  <BOOK>
    <TITLE>Дети капитана Гранта</TITLE>
    <AUTHOR>Жюль Верн</AUTHOR>
    <BINDING>Азбука (Азбука-классика)</BINDING>
    <PAGES>640</PAGES>
    <PRICE>88р.</PRICE>
  </BOOK>
</INVENTORY>
```

# ПРИМЕР (РЕЗУЛЬТАТ)

***Мастер и маргарита***

**Михаил Булгаков**

Ленинградское издательство (Лениздат)

74р.

***Ревизор***

**Николай Гоголь**

Искатель

37р.

***Дети капитана Гранта***

**Жюль Верн**

Азбука (Азбука-классика)

88р.

# XSL-ТАБЛИЦА СТИЛЕЙ

- XSL-таблица стилей (eXtensible Stylesheet Language - расширяемый язык таблиц стилей) связывается с XML-документом и сообщает браузеру, как отображать данные XML.

# ОТОБРАЖЕНИЕ НА СТРАНИЦЕ

- Создание файла XSL-таблицы стилей. XSL является приложением XML, т.е. XSL-таблица представляет собой корректно сформированный XML-документ, который отвечает правилам XSL.
- Связывание XSL-таблицы стилей с XML-документом. В XML-документ включается инструкция по обработке xml-stylesheet, которая имеет следующую форму записи:

**<?xml-stylesheet type="text/xsl" href=xslFileURL?>**

# XSL-ТАБЛИЦА ВКЛЮЧАЕТ ОДИН ИЛИ НЕСКОЛЬКО *ШАБЛОНОВ*

```
<xsl:stylesheet xmlns:xsl="1.xml">
<!-- один или несколько элементов шаблонов... -->
</xsl:stylesheet>
```

# ЭЛЕМЕНТ XSL:STYLESHEET

- Служит не только хранилищем всех других элементов, но также идентифицирует документ как XSL-таблицу стилей.

# Шаблоны имеют следующую форму

```
<xsl:template match="/">
<!-- дочерние элементы... -->
</xsl:template>
```

Атрибут match шаблона указывает на определённую ветвь и аналогичен селектору в правиле CSS.

Значение атрибута match называется *образцом* (pattern).

# Шаблон может содержать два вида XML-элементов

- **XML-элементы**, представляющие HTML-разметку, например: **<h2>Каталог товаров</h2>.** Браузер просто скопирует каждый HTML-элемент непосредственно на выход HTML. Каждый из элементов, представляющих HTML-разметку, должен быть корректно сформированным XML-элементом. Например, чтобы задать перевод строки в HTML, вы должны использовать тэг пустого элемента <BR />.
- Собственно **XSL-элементы**, например: **<xsl:value-of select="PRODUCT/TITLE" />** Браузер отличает XML-элемент от HTML-элемента, поскольку первый имеет префикс xsl.

# XSL-ЭЛЕМЕНТ *VALUE-OF*

- добавляет текстовое содержимое определённого XML-элемента и всех его дочерних элементов в выходной модуль HTML.
- Порядок следования элементов value-of в шаблоне определяет порядок вывода информации.
- XSL-таблица стилей имеет преимущество перед CSS-таблицей стилей, которая выводит данные всегда только в том порядке, в котором они следуют в XML-документе.

# ЭЛЕМЕНТ *FOR-EACH*

```
<xsl:template match="/">
<H2>Заголовок</H2>
<xsl:for-each select="PRODUCTS/PRODUCT">
<SPAN>Наименование:</SPAN>
<xsl:value-of select="TITLE" />
<!-- другие элементы шаблона... -->
</xsl:for-each>
</xsl:template>
```

# АТРИБУТ SELECT ЭЛЕМЕНТА FOR-EACH

- задаёт текущий элемент, поэтому внутри элемента for-each все образцы (пути к элементам в атрибутах select) задаются уже относительно этого текущего элемента

# Использование XSL-элемента *apply-templates*

```
<xsl:template match="/">
<H2>Заголовок</H2>
<xsl:apply-templates
select="PRODUCTS/PRODUCT" />
</xsl:template>
<xsl:template match="PRODUCT">
<SPAN>Наименование:</SPAN>
<xsl:value-of select="TITLE" /> <BR />
<!-- другие элементы шаблона... -->
</xsl:template>
```

# ПРИМЕР (SAMPLE.XSL)

```
<?xml version="1.1" encoding="windows-1251"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
    <H1>Каталог товаров</H1>
    <xsl:apply-templates
select="PRODUCTS/PRODUCT" />
</xsl:template>
```

# ПРИМЕР (ПРОДОЛЖЕНИЕ)

```
<xsl:template match="PRODUCT">
    <SPAN
style="font-style:italic">Наименование:</SPAN>
    <xsl:value-of select="TITLE" /> <BR />
    <SPAN
style="font-style:italic">Импортный:</SPAN>
    <xsl:value-of select="@import" /> <BR />
    <TABLE border="1" width="100%"
cellspacing="0">
    <xsl:apply-templates select="SORT" />
    </TABLE>
    <BR />
</xsl:template>
```

# ПРИМЕР (ПРОДОЛЖЕНИЕ)

```
<xsl:template match="SORT">
    <TR>
    <TD><xsl:value-of select="COLOR" /></TD>
    <TD><xsl:value-of select="PRICE" /></TD>
    </TR>
</xsl:template>
</xsl:stylesheet>
```

# ПРИМЕР (SAMPLE.XLS)

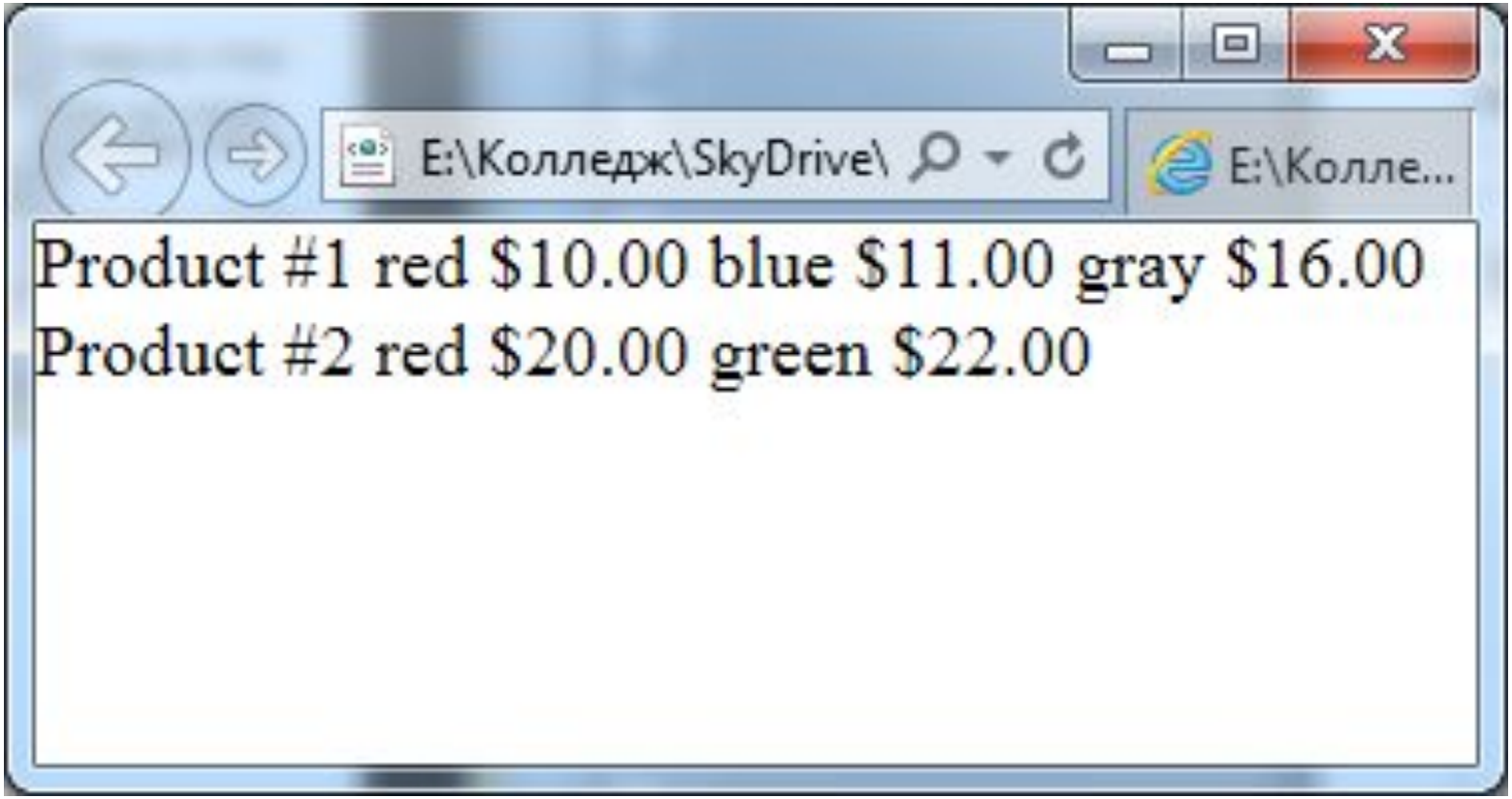
```
<?xml version="1.1"?>
<?xml-stylesheet type="text/xsl" href=Sample.xsl?>
<!DOCTYPE PRODUCTS
    [
    <!ELEMENT PRODUCTS (PRODUCT)*>
    <!ELEMENT PRODUCT (TITLE, SORT+)>
    <!ELEMENT TITLE (#PCDATA)>
    <!ELEMENT COLOR (#PCDATA)>
    <!ELEMENT PRICE (#PCDATA)>
    <!ELEMENT SORT (COLOR, PRICE)>
    <!ATTLIST PRODUCT import (yes | no) "no">
    ]
>
```

# ПРИМЕР (SAMPLE.XLS)

```
<PRODUCTS>
   <PRODUCT import="yes">
      <TITLE> Product #1 </TITLE>
      <SORT>
         <COLOR> red </COLOR>
         <PRICE> $10.00 </PRICE>
      </SORT>
      <SORT>
         <COLOR> blue </COLOR>
         <PRICE> $11.00 </PRICE>
      </SORT>
      <SORT>
         <COLOR> gray </COLOR>
         <PRICE> $16.00 </PRICE>
      </SORT>
   </PRODUCT>
```

```
<PRODUCT>
      <TITLE> Product #2 </TITLE>
      <SORT>
         <COLOR> red </COLOR>
         <PRICE> $20.00 </PRICE>
      </SORT>
      <SORT>
         <COLOR> green </COLOR>
         <PRICE> $22.00 </PRICE>
      </SORT>
   </PRODUCT>
</PRODUCTS>
```

# РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ



# Фильтрация и сортировка данных

- Можно ограничить количество элементов, отвечающих шаблону, введя фильтр - выражение, заключённое в квадратные скобки и следующее непосредственно за оператором пути.

# <XSL:APPLY-TEMPLATES SELECT="PRODUCTS/PRODUCT[SORT/COLOR='GRAY']" />

- обрабатывать надо только те элементы PRODUCT, у которых есть сорт серого цвета (элемент SORT имеет дочерний элемент COLOR, который содержит текст "gray"

# <XSL:APPLY-TEMPLATES SELECT="PRODUCTS/PRODUCT[@IMPORT='YES']" />

- обрабатывать надо только импортные товары (т.е. только те элементы PRODUCT, у которых атрибут import равен "yes"

- Если в фильтр включено только имя элемента (без знака равенства и "контрольного" значения), проверяется только наличие этого дочернего элемента.
- Если элемент имеет более одного дочернего элемента с именем, указанным в условии фильтрации, проверяется только первый дочерний элемент

# АТРИБУТ ORDER-BY

- Можно использовать атрибут order-by для сортировки данных XML при выводе.
- Можно назначить атрибуту order-by один или несколько образцов, разделяя их точкой с запятой.
- Браузер будет сортировать элементы с использованием образцов в том порядке, в котором они перечислены.
- Для указания направления сортировки (по возрастанию или убыванию) следует предварять образец префиксом + или -.

# <XSL:APPLY-TEMPLATES SELECT="PRODUCTS/PRODUCT" ORDER-BY="+@IMPORT; -TITLE" />

- сортировка товаров по возрастанию по признаку импорта, а для товаров с одинаковым признаком импорта - сортировка (по убыванию) по наименованию

# ПРОСТОЙ ПРИМЕР

- Файл 1.xml

- <?xml version="1.1" encoding="WINDOWS-1251"?>
  <tutorial>
  <title>"Заметки об XSL"</title>
  <author>Леонов Игорь Васильевич</author>
  </tutorial>

# ОТОБРАЖЕНИЕ В БРАУЗЕРЕ

С этим XML-файлом не связана ни одна таблица стилей. Ниже показано дерево элементов.

```
– <tutorial>
    <title>"Заметки об XSL"</title>
    <author>Леонов Игорь Васильевич</author>
  </tutorial>
```

# ДОБАВЛЕНИЕ ССЫЛКИ НА XSL ФАЙЛ

- <?xml version="1.1" encoding="WINDOWS-1251"?>
  **<?xml-stylesheet type='text/xsl' href='1.xsl'?>**
  <tutorial>
  <title>"Заметки об XSL"</title>
  <author>Леонов Игорь Васильевич</author>
  </tutorial>

# XSL-ФАЙЛ 1.XSL

```
<xsl:stylesheet version="1.1"
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<p><strong>
<xsl:value-of select="//title"/>
</strong></p>
<p><xsl:value-of select="//author"/></p>
</xsl:template>
</xsl:stylesheet>
```

# РЕЗУЛЬТАТ

**"Заметки об XSL"**

Леонов Игорь Васильевич

# ИЗМЕНЕНИЕ ПОРЯДКА СТРОК

```
<xsl:stylesheet version="1.1"
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<p><strong>
<xsl:value-of select="//author"/>
</strong></p>
<p><xsl:value-of select="//title"/></p>
</xsl:template>
</xsl:stylesheet>
```

# ПРИМЕР 2

```
<?xml version="1.1"
encoding="WINDOWS-1251"?>
<?xml-stylesheet type='text/xsl'
href='ex02-1.xsl'?>
<tutorial>
<dog caption="Собака: " name="Шарик">
<dogInfo weight="18 кг" color="рыжий с
черными подпалинами"/>
</dog>
</tutorial>
```

```
<xsl:stylesheet version="1.1"
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<P><B><xsl:value-of
select="//dog/@caption"/></B>
<xsl:value-of select="//dog/@name"/>.
<xsl:value-of select="//dogInfo/@weight"/>,
<xsl:value-of select="//dogInfo/@color"/>.</P>
</xsl:template>
</xsl:stylesheet>
```

- Результат имеет следующий вид:
- Собака: Шарик. 18 кг, рыжий с черными подпалинами.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<P><B><xsl:value-of
select="//animals/dog/@caption"/></B>
<xsl:value-of
select="//animals/dog/@name"/>.
<xsl:value-of
select="//animals/dog/dogInfo/@weight"/>,
<xsl:value-of select="//dogInfo/@color"/>.</P>
</xsl:template>
</xsl:stylesheet>
```

# ПРОСТАЯ ТАБЛИЦА

```
<xsl:template match="/">
<table border="1"> <tr bgcolor="#CCCCCC">
<td align="center"><strong>Кличка</strong></td>
<td align="center"><strong>Вес</strong></td>
<td align="center"><strong>Цвет</strong></td>
</tr>
<xsl:for-each select="tutorial/animals/dogs/dog">
<tr bgcolor="#F5F5F5">
<td><xsl:value-of select="dogName"/></td>
<td align="right"><xsl:value-of select="dogWeight"/>
<xsl:value-of select="dogWeight/@caption"/></td>
<td><xsl:value-of select="dogColor"/></td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
```

# РЕЗУЛЬТАТ

| Кличка | Вес | Цвет |
|---|---|---|
| Шарик | 18 кг | рыжий с черными подпалинами |
| Тузик | 10 кг | белый с черными пятнами |
| Бобик | 2 кг | бело-серый |
| Трезор | 25 кг | черный |

# СОРТИРОВКА АТРИБУТ ORDER-BY

- <xsl:for-each select="tutorial/animals/dogs/dog" order-by="dogName">

| Кличка | Вес | Цвет |
|---|---|---|
| Бобик | 2 кг | бело-серый |
| Трезор | 25 кг | черный |
| Тузик | 10 кг | белый с черными пятнами |
| Шарик | 18 кг | рыжий с черными подпалинами |

# СОРТИРОВКА ЧИСЛОВЫХ ЗНАЧЕНИЙ

- order-by="dogName" заменим на order-by="number(dogWeight)".

| Кличка | Вес | Цвет |
|---|---|---|
| Бобик | 2 кг | бело-серый |
| Тузик | 10 кг | белый с черными пятнами |
| Шарик | 18 кг | рыжий с черными подпалинами |
| Трезор | 25 кг | черный |

# СОРТИРОВКА ПО НЕСКОЛЬКИМ СТОЛБЦАМ

- order-by="number(dogWeight); dogName"

| Кличка | Вес | Цвет |
|---|---|---|
| Трезор | 10 кг | черный |
| Тузик | 10 кг | белый с черными пятнами |
| Бобик | 18 кг | бело-серый |
| Шарик | 18 кг | рыжий с черными подпалинами |

# СОРТИРОВКА ПО УБЫВАНИЮ

- значение атрибут order - значение ascending заменено на descending.

| Кличка | Вес | Цвет |
|---|---|---|
| Шарик | 18 кг | рыжий с черными подпалинами |
| Тузик | 10 кг | белый с черными пятнами |
| Трезор | 25 кг | черный |
| Бобик | 2 кг | бело-серый |

# ЭЛЕМЕНТ XSL:IF - ФИЛЬТР

- xsl:for-each select="tutorial/animals/dogs/dog[dogWeight$gt$10] " order-by="number(dogWeight); dogName;">

| Кличка | Вес | Цвет |
|---|---|---|
| Шарик | 18 кг | рыжий с черными подпалинами |
| Трезор | 25 кг | черный |

# ДРУГИЕ СПОСОБЫ СОРТИРОВКИ И ФИЛЬТРАЦИИ ДАННЫХ

- <xsl:sort order="ascending" select="number(dogWeight)"/> <xsl:sort order="ascending" select="dogName"/>

- условие фильтра у нас вынесено в отдельный элемент xsl:if.
  - <xsl:if test="dogWeight>10">
  - конечный тег элемента xsl:if.

# ПРИМЕР

```
<xsl:if test="dogWeight&gt;10">
<tr bgcolor="#F5F5F5">
<td><xsl:value-of select="dogName"/></td>
<td align="right"><xsl:value-of
select="dogWeight"/>
<xsl:value-of select="dogWeight/@caption"/>
</td>
<td>
<xsl:value-of select="dogColor"/>
</td>
</tr>
</xsl:if>
```

| Кличка | Вес | Цвет |
|---|---|---|
| Тузик | 10 кг | белый с черными пятнами |
| Трезор | 25 кг | черный |

- Функция start-with(string,startSubstring) проверяет, начинается ли строка string с подстроки startSubstring.

- Синтаксис элемента xsl:if.
- <xsl:if test="starts-with($varDogName,$varStartWith)">

- Значения переменных были инициализированы ранее
- <xsl:variable name="varStartWith">T</xsl:variable> <xsl:for-each select="tutorial/enimals/dogs/dog"> <xsl:variable name="varDogName"><xsl:value-of select="dogName"/></xsl:variable>

# СИНТАКСИС ЭЛЕМЕНТА XSL:IF.

- <xsl:if test="contains($varDogName,$varStartWith)">

- Два элемента xsl:if, вложенные друг в друга, дают нам эффект оператора AND
- <xsl:if test="dogWeight>10">
  <xsl:if test="dogWeight<20">
  ...
  </xsl:if>
  </xsl:if>
- Можно добиться и эффекта оператора OR. Для этого нам нужно включить два цикла, в каждом из которых формируется своя выборка

# Элемент xsl:if - улучшение внешнего вида таблиц

- Элемент xsl:if можно применять не только для фильтрации строк выборки.
- Можно использовать эту функцию для того, чтобы чередовать цвет четных и нечетных строк таблицы

- <tr>
  <xsl:if test="position() mod 2 = 0">
  <xsl:attribute
  name="bgcolor">#CCCCCC</xsl:attribute>
  </xsl:if>

# ДИНАМИЧЕСКОЕ ФОРМИРОВАНИЕ АТРИБУТОВ НА ПРИМЕРЕ ПАРАМЕТРОВ ССЫЛКИ В ТЕГЕ <A>

- Предположим теперь, что в каждой строке таблицы нам нужно сделать ссылку на некоторую страницу и передать на эту страницу два параметра - кличку и вес собаки.
- xsl:attribute

- <a target="_blank">
  <xsl:attribute
  name="href">DisplayDetails.html?dogName=<
  xsl:value-of
  select="dogName"/>&dogWeight=<xsl:va
  lue-of select="dogWeight"/></xsl:attribute>
  <xsl:attribute name="title">To view some
  more details about <xsl:value-of
  select="dogName"/> click to dog
  name</xsl:attribute>
  <xsl:value-of select="dogName"/>
  </a>