4. Интерфейсы и XML

Информационные технологии в корпоративных сетях

- Раздел 4

Интерфейсы и XML

Информационные технологии в корпоративных сетях

- Широкому использованию Интернет для организации взаимодействия между системами категории B2B и е-коммерции мешали две проблемы:
- Невозможность выполнения полноценных бизнес-транзакций;
- Разнородность интерфейсов RPC и необходимость переформатирования передаваемых данных

Интерфейсы и XML

- B2B-стиль интегрирует бизнес-процессы компаний, он требует передачи коммерческой информации между различными элементами бизнеса. Но способ ведения бизнеса, идентификация и использование данных существенно отличаются от предприятия к предприятию.
- Попытки использовать для передачи данных между предприятиями страницы HTML натолкнулись на непреодолимые трудности

Интерфейсы и XML

- Для решения этой проблемы был разработан новый стандарт языка разметки, называемый extensible Markup Language, или XML
- Язык XML разработан для облегчения обмена структурированными документами, например, заказами или счетами по Интернету и представляет собой метаязык, предназначенный для представления и манипулировании элементами данных.

Интерфейсы и XML

- Возможны несколько уровней или аспектов определения XML:
- На первом уровне XML является протоколом хранения и управления информацией.
- На следующем уровне – это семейство технологий, с помощью которых можно осуществлять все – от оформления документов до фильтрации данных.
- И, наконец, на самом высоком уровне – это философия обработки информации, которая призвана обеспечить максимальную полезность и гибкость данных путем придания им наиболее чистой и структурированной формы

Интерфейсы и XML

- Технология XML предназначена для транспортировки и хранения структурированных данных взамен существующих файлов баз данных, для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки, (например, XHTML).
- XML-данные хранятся в **текстовом формате**. Это обеспечивает программно- и аппаратно-независимый способ хранения данных.

Интерфейсы и XML

- XML - это метаязык.
- Он не является языком разметки, а определяет набор правил для создания языков разметки.
- Язык XML не имеет предопределенных элементов разметки (тегов), а указывает, как создавать собственные .
- XML позволяет автору определить свои собственные теги и структуру документа.

Интерфейсы и XML

- [Определение:
- Объект данных становится **XML документом** если, в соответствии с определениями спецификации, он является корректным. Корректный XML документ также может стать действительным, если отвечает некоторым дополнительным ограничениям.]

Интерфейсы и XML

- Каждый XML документ имеет логическую и физическую структуру. Физически документ состоит из элементов, называемых сущностями.
- Любая сущность может ссылаться на другие сущности, обеспечивая их включение в данный документ. Документ начинается с "корня" или сущности документа. С логической точки зрения, документ строится из деклараций, элементов, комментариев, ссылок на символ и инструкций обработки. Все они размечаются в документе явным образом. Логические и физические структуры должны иметь корректную вложенность

Интерфейсы и XML

- Определение: Текстовый объект становится **корректным** (well-formed) XML документом, если:
- как единое целое, он соответствует сценарию document.
- отвечает всем ограничениям корректности, представленным в спецификации XML.
- Все разобранные сущности, на которые в данном документе прямо или косвенно делается ссылка, являются корректными (well-formed).

Интерфейсы и XML

- Соответствие сценарию document подразумевает следующее:
- В данном объекте содержится один или несколько элементов.
- [Определение: В объекте имеется в точности один элемент, называемый **корневым** или элементом документа, ни одна из частей которого не попадает в содержимое какого-либо еще элемента.] Для всех остальных элементов действует правило, что если начальный тэг находится в содержимом некого элемента, то и конечный тэг должен находиться среди содержимого того же элемента. Проще говоря, элементы, маркируемые начальными и конечными тэгами, должны быть вложены друг в друга правильным образом.

Интерфейсы и XML

- [Определение: Из вышесказанного следует что в документе для любого некорневого элемента **C** имеется другой элемент **P** из этого же документа, такой что **C** находится в содержимом **P**, но при этом не попадает в содержимое какого-либо третьего элемента, также находящегося в содержимом элемента **P**. В таком случае об элементе **P** говорят как о **родителе** элемента **C**, а элемент **C** называют **непосредственным потомком** элемента **P**.]

Интерфейсы и XML

- **Символы**
- [Определение: Разобранная сущность (parsed entity) содержит **текст** - последовательности символов, образующие разметку и символьные данные.]
- [Определение:**символ** - это элементарная единица текста, описанная в ISO/IEC 10646 [ISO/IEC 10646] (см. также [ISO/IEC 10646-2000]). Допустимы символы табуляции, возврата каретки, конца строки, а также разрешенные символы из наборов Unicode и ISO/IEC 10646.. Перечисленные стандарты могут быть дополнены новыми символами в ходе обновления или при написании для них новых редакций. Соответственно, XML процессоры должны принимать любой символ из диапазона, указанного для Char..]

Интерфейсы и XML

- **Символьные данные и разметка**
- Текст документа образуется сочетанием символьных данных и разметки.
- [Определение: **Разметка** принимает форму начальных тэгов, конечных тэгов, тэгов пустых элементов, ссылок на сущности, ссылок на символы, комментариев, разделителей секций CDATA, объявлений типов документов, инструкций обработки, деклараций XML,деклараций текста и любых пробельных символов, которые располагаются на верхнем уровне сущности документа (то есть, вне элемента document и за пределами иных элементов разметки).]
- [Определение: Текст, который не относится к разметке, формирует **символьные данные** документа (character data).]

Интерфейсы и XML

- Определение: **Комментарий** может размещаться в любом месте документа при условии, что он не попадает в границы какого-либо элемента разметки. Комментаций может также появляться в тех местах декларации типа документа, где это разрешено грамматикой. Комментарии не относятся к символьным данным документа, однако XML процессоры могут (но не обязаны) передавать приложению текст полученных комментариев. Пример комментария:
- <!-- declarations for <head> & <body> -->

Интерфейсы и XML

- Определение: **Инструкции обработки**
- (processing instruction, PI) позволяют размещать в документе инструкции для приложений.
- Инструкция PI начинается с указания адреса (PITarget), используемого для идентификации приложения, которому предназначается эта инструкция
- Для формального декларирования адресата инструкции PI может использоваться механизм нотацийXML

Декларирование нотаций

- [Определение: **Нотация** идентифицирует по имени формат неразобранных сущностей, формат элементов, обеспечивающих атрибут нотации, или же приложение, которому адресуется инструкция обработки.]
- [Определение: **Декларация нотации** дает этой нотации название, используемое при декларировании сущности, списка атрибутов или в спецификациях атрибутов, а также внешний идентификатор этой нотации, который может позволить XML процессору или его клиентскому приложению найти вспомогательную программу, способную обработать данные, представленные в этой нотации.]
- **Декларации нотации**
- [82] **NotationDecl ::= '<!NOTATION' S Name S (ExternalID | PublicID) S? '>'**[VC: Уникальность имени нотации]
  [83] **PublicID ::= 'PUBLIC' S PubidLiteral**

Декларирование нотаций

- **Ограничение действительности: Уникальность имени нотации**
- Любое Name может быть использовано только в одной декларации.
- XML процессор должен передать приложению название и внешний идентификатор(ы) всех нотаций, которые были декларированы и на которые имеется ссылка в значениях атрибутов, определениях атрибутов, либо декларациях сущностей. Кроме того, процессор может преобразовывать внешний идентификатор в системный идентификатор, имя файла или иную информацию, необходимую приложению чтобы вызвать процессор для обработки данных в описываемой нотации. (Впрочем, ситуация, когда XML документ декларирует и ссылается на нотацию, для которой не имеется соответствующего процессора обработки в системе, где работают XML процессор или приложение ошибочной не будет.)

Интерфейсы и XML

- Определение: **Секция CDATA** может находиться повсюду, где могут размещаться символьные данные. Использование секции CDATA позволяет избежать обработки блока текста, содержащего символы, которые в других случаях распознавались бы как разметка. Секция CDATA начинается со строки "<![CDATA[" и заканчивается строкой "]]>":]

Интерфейсы и XML

- В секции CDATA распознается только один элемент разметки - строка CDEnd. Поэтому все символы левой угловой скобки и амперсанта могут предстать здесь в своем обычном текстовом виде. Эти символы не нужно (да и невозможно) маскировать с помощью комбинаций "<" и "&". Секции CDATA не могут быть вложенными.
- Пример секции CDATA, в которой строки "<greeting>" и "</greeting>" будут распознаваться не как разметка, а как обычные символьные данные:
- <![CDATA[<greeting>Hello, world!</greeting>]]>
-

Интерфейсы и XML

- Определение: Документ XML должен начинаться с **декларации XML**, указывающей версию используемого языка XML
- Задачей разметки XML документа должно быть описание схемы его размещения и логической структуры, а также связывание пар атрибут-значение с их логической структурой.
- XML предоставляет механизм для определения логических ограничений для логической структуры и формирования предопределенных единиц размещения -декларацию типа документа.
- [Определение: XML документ является **действительным**, если с ним связана декларация типа документа и если этот документ отвечает представленным в ней ограничениям.]
- Декларация типа должна располагаться в документе до первого элемента.

XML:особенности синтаксиса

- Все XML элементы наряду с открывающим должны иметь закрывающий тег:
- Параграф
- Следующий параграф
- В тегах нужно учитывать регистр:
- XML-теги чувствительны к регистру. Так <Letter> отличается от тега <letter>.
- Открывающий и закрывающий теги должно быть написаны с тем же регистром:
- <Message>Неверно</message>
- <message>Верно</message>

XML:особенности синтаксиса

- XML элементы должны быть правильно вложенными друг в друга:
- This text is bold and italic
- Значения атрибута должно быть заключено в кавычки:
- <note date="12/11/2007">
-

XML:особенности синтаксиса

- для добавления комментариев в XML-документ используются символы < -- к —>;
- префиксы «XML» и «xml» зарезервированы только для тегов XML .
- В XML определены два метода записи специальных символов: ссылка на сущность и ссылка по номеру символа.
- Ссылка на сущность (англ. entity references) указывается в том месте, где должна быть сущность и состоит из амперсанда (&), имени сущности и точки с запятой (;).

Особенности синтаксиса XML

- В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация;
- Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки;
- Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов;

Особенности синтаксиса XML

- В XML есть несколько предопределённых сущностей, но возможно также определять собственные.
- Полный список предопределённых сущностей состоит из &amp (&), &lt (<), &gt (>), &apos (') и &quot (") — последние две полезны для записи разделителей внутри значений атрибутов. Определить свои сущности можно в DTD-документе.

Особенности синтаксиса XML

- Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные, и поэтому учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML).
- Если XML-документ не нарушает приведенные правила, то он называется ***формально-правильным*** и все анализаторы, предназначенные для разбора XML-документов, смогут работать с ним корректно

Особенности синтаксиса XML

- Угловые скобки (< >) и заключенные в них имена называются тегами (tags). Теги разграничивают и помечают части документа, а также добавляют другую информацию, которая помогает определить структуру. Текст между тегами является содержимым документа, необработанной информацией, которая может быть телом сообщения, заголовком или полем данных. Разметка и содержимое дополняют друг друга, создавая информационный объект, в котором поделенные на части и помеченные данные заключены в удобный пакет.

Средства проверки правильности XML документа

- Кроме проверки на формальное соответствие грамматике языка, в документе могут присутствовать средства контроля над содержанием документа,
- Существует два способа контроля правильности XML-документа: **DTD - определения(Document Type Definition)** и **схемы данных(Semantic Schema)**.
- Они содержат правила, определяющие необходимые соотношений между элементами и формирующие **структуру документа**.

Проверяющие и непроверяющие процессоры

- XML процессоры, отвечающие требованиям спецификации, делятся на два класса: проверяющие и непроверяющие.
- И проверяющие, и непроверяющие процессоры должны докладывать о нарушениях правил корректности данной спецификации, выявленных в содержимом сущности документа и содержимом других читаемых разобранных сущностей.

Проверяющие процессоры

- [Определение: Проверяющие процессоры должны сообщить (по выбору пользователя) о нарушении ограничений, сформулированных в декларациях DTD, а также невозможности соответствовать критериям действительности, представленным в данной спецификации.] Чтобы выполнить это требование, проверяющий XML процессор должен прочесть и обработать весь DTD и все внешние разобранные сущности, на которые в данном документе делается ссылка.

Непроверяющий процессор

- Для проверки корректности от непроверяющего процессора требется проанализировать лишь сущность документа, включая полный внутренний набор DTD. [Определение: Хотя непроверяющий процессор и не обязан проверять действительность документа, он должен **обработать** все декларации, найденные во внутреннем наборе DTD, а также во всех прочитанных им сущностях параметров, но только до первой ссылки на сущность параметра, которую он уже *не* должен читать. Иными словами, он должен использовать сведения из этих деклараций для *нормализации* значений атрибутов, *подстановки* текста замены для внутренних сущностей и предоставления *значений по умолчанию для атрибутов*.] За исключением случая **standalone="yes"**, процессорам запрещается обрабатыватьдекларации сущностей и списков атрибутов, расположенные после ссылки на сущность параметра, последняя не читается, поскольку может содержать переопределяющие декларации.
-

Определение типа документа (DTD) и XML-схемы

- Решения B2B-стиля требуют высокой степени бизнес-интеграции компаний. Компании, использующие транзакции B2B-стиля, должны иметь способ понимать и проверять любые другие теги. Один из методов выполнения этой задачи состоит в использовании определения типа документа (Document Type Definition, DTD).

Определение типа документа (DTD)

- DTD представляет собой файл с расширением dtd, который описывает XML-элементы, DTD-файл обеспечивает построение логической модели базы данных и определяет синтаксические правила или теги проверки для каждого типа XM L-документов.
- Компании класса B2B и е-коммерции должны разработать DTD и предоставить его в совместное использование

Определение типа документа (DTD)

- В XML-документах DTD определяет набор действительных элементов, идентифицирует элементы, которые могут находиться в других элементах, и определяет действительные атрибуты для каждого из них .
- В DTD для XML используются следующие типы правил: правила для элементов и их атрибутов, описания категорий(макроопределений), описание форматов бинарных данных

Определение типа документа (DTD)

- Для того, чтобы использовать DTD в документе, можно или описать его во внешнем файле и при описании DTD просто указать ссылку на этот файл или же непосредственно внутри самого документа выделить область, в которой определить нужные правила.

Определение типа документа (DTD)

- В первом случае в документе указывается имя файла, содержащего DTD-описания:
- **<?xml version="1.0" standalone=“no" ?>**
- **<! DOCTYPE team SYSTEM "team.dtd">**
- **...**
- Внутри же документа DTD- декларации включаются следующим образом:
- **<?xml version="1.0" ?>**
- **<! DOCTYPE team [**
- **<! ELEMENT team (coach, player, assistant)>**
- **...**
- **]>**

Определение типа документа (DTD)

- В том случае, если используются одновременно внутренние и внешние описания, то программой-анализатором будут сначала рассматриваться внутренние, т.е. их приоритет выше.
- При проверке документа XML-процессор в первую очередь ищет DTD внутри документа.
- Если правила внутри документа не определены и не задан атрибут ***standalone ="yes"*** , то программа загрузит указанный внешний файл и правила, находящиеся в нем, будут считаны оттуда.
- Если же атрибут ***standalone*** имеет значение *"yes"*, то использование внешних DTD описаний будет запрещено

Определение типа документа (DTD)

Определение элемента

- Элемент в DTD определяется с помощью дескриптора **!ELEMENT**, в котором указывается название элемента и структура его содержимого. Например, для элемента **<coach>** можно определить следующее правило:
- **<!ELEMENT coach (#PCDATA)>**

PCDATA - parseable character data - любая информация, с которой может работать программа-анализатор) В тексте как правило символ P опускается. В данной секции размещается «чистый» текст.

Определение типа документа (DTD)

- Существует еще две инструкции, определяющие тип содержимого:

EMPTY,ANY.

- **EMPTY** указывает на то, что элемент должен быть пустым(например, <red/>),
- **ANY** - на то, что содержимое элемента специально не описывается.

Определение типа документа (DTD)

- Для текущего элемента может быть задано несколько дочерних объектов в виде списка разделенных запятыми названий элементов.
- Для того, чтобы указать количество повторений включений этих элементов могут использоваться символы или индикаторы повторяемости +,*, ? :

<!ELEMENT team(coach+, player*, assistant?)>

индикаторы повторяемости

Индикатор	Смысл
?	Содержимое встречается не более одного раза
*	Содержимое встречается любое количество раз
+	Содержимое встречается не менее 1 раза
[ничего не указывается]	Содержимое встечается ровно 1 раз

Пример DTD для XML документа

- <?xml version="1.0"?>
- <! DOCTYPE team [
- <!ELEMENT team (title,coach+, player*, assistant?)>
- <!ELEMENT coach (name|PCDATA)>
- <!ELEMENT name (#CDATA)>
- <!ELEMENT player (name, nationality)>
- <!ELEMENT nationality (#CDATA)>
- <!ELEMENT l_name (#CDATA)>
- <!ELEMENT assistant (#CDATA)>]>

Задание списков атрибутов элемента

- Списки атрибутов элемента определяются с помощью ключевого слова **!ATTLIST**. Внутри него задаются названия атрибутов, типы их значений и дополнительные параметры. Например, для элемента <player> могут быть определены следующие атрибуты:
- **<!ATTLIST playernumber ID #REQUIRED**

  type (goalkeeper | back | halfback | forward) #IMPLIED

Типы значений атрибутов (1)

- CDATA - содержимым документа могут быть любые символьные данные
- ID - определяет уникальный идентификатор элемента в документе
- IDREF(IDREFS) - указывает, что значением атрибута должно выступать название (или несколько таких названий, разделенных пробелами во втором случае) уникального идентификатора, определенного в этом документе элемента

Типы значений атрибутов (2)

- ENTITY(ENTITIES - значение атрибута должно быть названием(или списком названий, если используется ENTITIES) компонента (макроопределения), определенного в документе
- NMTOKEN (NMTOKENS) - содержимым элемента может быть только одно отдельное слово(т.е. этот параметр является ограниченным вариантом CDATA)
- Список допустимых значений - определяется список значений, которые может иметь данный атрибут

Типы значений атрибутов (3)

- Так же в определении атрибута можно использовать следующие параметры:
- #REQUIRED - определяет обязательный атрибут, который должен быть задан во всех элементах данного типа
- #IMPLIED - атрибут не является обязательным
- #FIXED "значение" - указывает, что атрибут должен иметь только указанное значение, однако само определение атрибута не является обязательным, но в процессе разбора его значение в любом случае будет передано программе-анализатору
- Значение - задает значение атрибута по умолчанию

Определение компонентов (макроопределений)

- Компонент (entity) представляет собой определения, содержимое которых может быть повторно использовано в документе. В других языках программирования подобные элементы называются макроопределениями. Создаются DTD-компоненты при помощи инструкции !ENTITY:
- **<!ENTITY hello ' Мы рады приветствовать Вас!' >**
- Программа-анализатор, просматривая в первую очередь содержимое области DTD- определений, обработает эту инструкцию и при дальнейшем разборе документа будет использовать содержимое DTD-компонента в том месте, где будет встречаться его название. Теперь в документе можно использовать выражение &hello; , которое будет заменено на строчку *"Мы рады приветствовать Вас"*

Определение компонентов (1)

- Внутри DTD можно задать три типа макроопределений:
- **внутренние макроопределения** - предназначены для определения строковой константы, с их помощью можно организовывать ссылки на часто изменяемую информацию, делая документ более читабельным. Внутренние компоненты включаются в документ при помощи амперсанта &

Определение компонентов (2)

- В XML существует пять предустановленных внутренних символьных констант:
- < - символ "<"
- > - символ ">"
- & - символ "&"
- ' - символ апострофа "'"
- " - символ двойной кавычки """"

Определение компонентов (3)

- **Внешние макроопределения** - указывают на содержимое внешнего файла, причем этим содержимым могут быть как текстовые, так и двоичные данные. В первом случае в месте использования макроса будут вставлены текстовые строки, во втором - бинарные данные, которые анализатором не рассматриваются и используются внешними программами
- <!ENTITY logotype SYSTEM "/image.gif" NDATA GIF87A>

Определение компонентов (4)

- **Макроопределения правил** - макроопределения параметров могут использоваться только внутри области DTD и обозначаются специальным символом %, вставляемым перед названием макроса. При этом содержимое компонента будет помещено непосредственно в текст DTD-правила
- Рассмотрим пример:

Определение компонентов (5)

- для следующего фрагмента документа:
- **<!ELEMENT title (PCDATA)> <!ELEMENT name (PCDATA)> <!ELEMENT nationality (PCDATA)> <!ELEMENT coach (PCDATA | name)> <!ELEMENT player ((PCDATA | name), nationality)>**
- **<!ELEMENT team (title,coach, player*)>**

Определение компонентов (6)

- можно использовать более короткую форму записи:
- **<!ELEMENT name (PCDATA)>**
- **<! ENTITY %names 'PCDATA | name'> <!ELEMENT coach (%names;)> <!ELEMENT player (%names, nationality)>**
- **<!ENTITY %content 'coach | (player*)'>**
- **<!ELEMENT team (title,%content;)>**

Типизация данных

- Довольно часто при создании XML-элемента разработчику требуется определить, данные какого типа могут использоваться в качестве его содержимого ;
- Используя типизацию данных, можно создавать элементы, значения которых могут использоваться, например, в качестве параметров SQL-запросов.
- Если в качестве программы на стороне клиента используется верифицирующий XML-процессор, то информацию о типе можно передавать при помощи специально созданного для этого атрибута элемента, имеющего соответствующее DTD-определение.

DTD-определение типа данных

- **Пример**: чтобы указать, что содержимое элемента должно быть длинным целым, можно использовать следующее DTD- определение:
- **<!ELEMENT counter (PCDATA)>**
- **<!ATTLIST counter data_long CDATA #FIXED "LONG">**
- Задав атрибуту значение по умолчанию LONG и определив его как FIXED, программа-клиент получит необходимую информацию о типе содержимого данного элемента, и сможет самостоятельно определить соответствие типа этого содержимого указанному в DTD-определении

DTD-определение типа данных

- <!ELEMENT price (PCDATA)>
- <!ATTLIST price data_currency CDATA #FIXED "CURRENCY">
- <!ELEMENT rooms_num (PCDATA)>
- <!ATTLIST rooms_num data_byte CDATA #FIXED "BYTE">
- <!ELEMENT floor (PCDATA)>
- <!ATTLIST floor data_byte CDATA #FIXED "INTEGER">
- <!ELEMENT living_space (PCDATA)>
- <!ATTLIST living_space data_float CDATA #FIXED "FLOAT">
- <!ELEMENT counter (PCDATA)>
- <!ATTLIST counter data_long CDATA #FIXED "LONG">
- <!ELEMENT is_tel (PCDATA)>
- <!ATTLIST is_tel data_bool CDATA #FIXED "BOOL">
- <!ELEMENT house (rooms_num, floor,living_space, is_tel, counter, price)>
- <!ATTLIST house id ID #REQUIED>
- ……

DTD-определение типа данных

- Продолжение: фрагмент локумента XML
- ……..
- <house id="0">
- <rooms_num>5</rooms_num>
- <floor>2</floor>
- <living_space>32.5</living_space>
- <is_tel>true</is_tel>
- <counter>18346</counter>
- <price>100 р. 00 к.</price>
- </house>

Схемы данных

- Схемы данных (Schemas) являются альтернативным способом создания правил построения XML-документов. По сравнению с DTD, схемы обладают более мощными средствами для определения сложных структур данных, обеспечивают более понятный способ описания грамматики языка, способны легко модернизироваться и расширяться. Безусловным достоинством схем является также то, что они позволяют описывать правила для XML-документа средствами самого же XML

Схемы данных

- схемы не могут полностью заменить DTD-описания - этот способ определения грамматики языка используется сейчас практическими всеми верифицирующими анализаторами XML и, более того, сами схемы, как обычные XML-элементы, тоже описываются DTD.
- Но серьезные возможности нового языка и его относительная простота, безусловно, дают основания утверждать, что данный стандарт имеет широкое применение в качестве удобного и эффективного средства проверки корректности составления документов

Схемы данных

- Документ размечается при помощи специальных элементов, выполняющих в схемах роль инструкций. Эти инструкции составляют набор правил, используя которые, программа-клиент будет делать вывод о том, корректен документ или нет. Схема данных, например, может выглядеть следующем образом:

Схемы данных. Пример (1)

- <schema id="TeamSchema">
- <elementType id="#namee">
- <string/>
- </elementType>
- <elementType id="player">
- <element type="#name"/>
- <attribute name="number"/>
- <attribute name="type"/>
- </elementType>

Схемы данных. Пример (2)

- <elementType id="team">
- <element type="#player"/>
- <attribute name=”title”/>
- </elementType>
- </schema

XML документ, соответствующий данной схеме данных

- **<team title=”Celtics”>**
- **<player number="1" type=”goalkeeper”> <name>John Ree</ name></ player> <player number="2" type=”back”> <name>Peter Loyd</ name></ player> <player number="2" type=”forward”> <name>Emil McGeer</ name></ player> </team>**
- Все конструкции языка схем описываются правилами "XML DTD for XML-Data-Schema".

Схемы данных

- Если использовать отдельное пространство имен, то полный XML-документ, содержащий в себе схему данных, будет выглядеть следующим образом:
- **<?XML version='1.0' ?>**
- **<?xml:namespace href= "http://www.mrcpk.nstu.ru/schemas/" as="s"/?> <s:schema id="OurSchema">**
- **<!-- последовательность инструкций --> </s:schema>**

Описание элементов

- Для определения класса элемента, к которому в дальнейшем будут применяться инструкции, описывающие его содержимое и структуру, предназначен специальный элемент схемы **elementType.**
- Название элемента задается атрибутом id . Все дальнейшие инструкции, которые относятся к описываемому классу, определяют его внутреннюю структуру и набор допустимых данных, содержатся внутри блока, заданного тэгами <elementType> и </elementType>.
- При определении класса элемента, можно также использовать комментарии к нему, которые заключаются в тэги <**descript**></**descript**>

Атрибуты элемента

- Для того, чтобы в описании элемента определить его атрибуты и описать свойства этих атрибутов нужно использовать элемент **attribute**:
- **<elementType id="player">**
- **<attribute name="number"/>…**
- **</elementType>**
- В данном примере элементу *<player>* определяется атрибут *number*, значением которого может быть любая последовательность разрешенных символов:
- **<player number="0"/>**
- **<player number="some text"/>**

Атрибуты элемента

- Подобно DTD, схемы данных позволяют устанавливать ограничения на значения и способ использования атрибутов. Для этого в дескрипторе <attribute> необходимо использовать параметр **atttype**. Например, если мы хотим указать, что значение атрибута должно использоваться программой-анализатором как уникальный идентификатор, то нам необходимо создать следующее правило:
- **<elementType id="player">**
- **<attribute name="number" atttype="ID"/>**
- **</elementType>**

Атрибуты элемента

Если же требуется задать список возможных значений атрибута, то пример будет выглядеть следующим образом:

- **<attribute name="type" atttype="ENUMERATION«**

values="goalkeeper, back, halfback, forward">

Модель содержимого элемента

- Под моделью содержимого в схеме данных понимают описание всех допустимых объектов XML-документа, использование которых внутри данного элемента является корректным.
- Модель содержимого определяется инструкциями, расположенными внутри блока **<elementType>.** Вложенные элементы описываются при помощи инструкции **element**, в которой параметром type указывается класс объекта - ссылка на его определение:
- **<elementType id="player">**
- **<element Type="#name"/>**
- **<element Type="#nationality"/>**
- **</elementType>**

Атрибуты элемента

- Если требуется указать режим использования вложенного элемента, то надо определить параметр **occurs**:
- **<elementType id="player">**
- **<elementType="#name" occurs="REQUIRED"/>**
- **<elementType="#nationality" occurs="OPTIONAL"/>**
- **<elementType="#clubs" occurs="ONEORMORE"/>**
- **</elementType>**

Атрибуты элемента

- Возможные значения этого параметра таковы:
- REQUIRED - элемент должен быть обязательно определен
- OPTIONAL - использование элемента не является обязательным
- ZEROORMORE - вложенный элемент может встречаться несколько раз или ни разу
- ONEORMORE - элемент должен встречаться хотя бы один раз

Примеры правильных XML-документов

- **<player><name>John Ree</name> <nationality>English</ nationality> <clubs>Celtics</clubs> <clubs>Portsmut</clubs>**

</article>

или

- **<player><name>John Ree</name> <clubs>Celtics</clubs> <clubs>Portsmut</clubs>**
- **</article>**

Модель содержимого XML документа

- Кроме элементов, содержимым XML-документа могут также являться обычный текст и области CDATA. Для обозначения типов содержимого текущего элемента в схемах используются следующие инструкции:
- **<string/>** - указывает на то, что содержимым элемента является только свободная текстовая информация(секция PCDATA) :

```
<elementType id="name">
<string/>
</elementType>
```

Модель содержимого XML документа

- **<any/>** - указывает на то, что содержимым элемента должны являться только элементы, без текста, не заключенного ни в один элемент:

  <elementType id="coach">

  <any/>

  </elementType>
- **<mixed>** - любое сочетание элементов и текста

  <elementType id="player">

  <mixed/>

  </elementType>
- **<empty>** - пустой элемент.

Группировка элементов

- Элемент **group** используется для того, чтобы задать некоторую последовательность вложенных объектов:

```
<elementType id="team">
   <element type="#title" occurs="REQUIRED"/>
   <group occurs="OPTIONAL">
    <element type="#player">
    <element type="#assistant">
   </group>
</elementType>
```

Преобразование XML документов. XSLT и XPATH

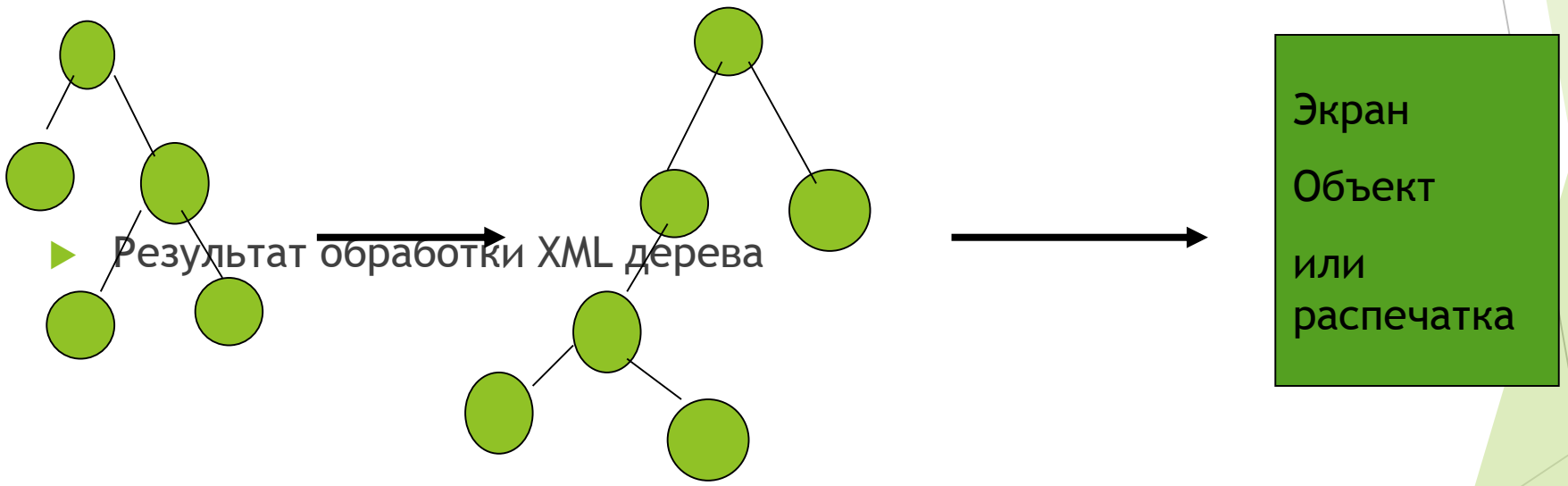
- Процесс преобразования и форматирования информации в готовый результат называется **стилизацией**. Чтобы стилизацию сделать возможной, существуют две рекомендации от W3C: **XSL преобразования** (XSLT), которые позволяют реорганизовывать информацию, а также сам язык **XSL**, который определяет как форматировать реорганизуемую информацию.

Этапы преобразования XML документов

- XSL stylesheet процессор производит представление исходного содержания XML к виду, определенному проектировщиком в stylesheet.
- Представление производится в 2 этапа:
- Первое – XML дерево (source tree) трансформируется в дерево преобразования (result tree);
- Второе – форматирование в представление пользователя. Этот процесс выполняется форматтером.

Этапы преобразования XML документов

- Данный процесс иллюстрируется следующей схемой
- XSL XSL
- трансформ форматтер



- Результат обработки XML дерева

Преобразование XML документов.

- Когда помещают XML и XSL данные в XSL-процессор, получают не просто улучшенный вариант XML кода. Получают результат в виде дерева, которое может быть расширено, изменено, либо реорганизовано.
- XSL процессор использует так называемый **стиль**, состоящий из набора XSL команд, который затем он преобразует, используя XML документ,

XSLT. Назначение и общее описание

- XSLT – язык преобразования XML документов в другие документы.
- XSLT является составной частью XSL, но может использоваться самостоятельно.
- XSL определяет стиль XML документа, используя XSLT.
- Помимо XSLT XSL содержит словарь XML для описания форматирования.

XSLT. Назначение и общее описание

- XSLT представляет собой правильно составленный документ XML и обладает определенной структурой (набором тэгов и атрибутов).
- Преобразование в языке XSLT предстает в виде корректного (правильного) XML документа, соответствующего требованиям для пространства имен XML, которое в спецификации XSLT называется пространством имен XSLT.

Описание стилей

- Преобразование, выраженное через XSLT , называется стилем (ctylesheet).
- Стили определяются набором команд XSL. Они создают корректные XML документы.
- Стили используют механизм совпадения паттернов (образцов) для нахождения элементов и атрибутов.

Описание стилей

- Описание стиля состоит из версии и пространства имен. Пространство имен объявляет префикс для тэгов, которые будут использованы в стиле и местонахождение описания тэгов:
- <xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

- . . .
- </xsl:stylesheet>
- Если присутствуют ссылки на какие-либо расширения, то должно быть указано и пространство имен. Например, если нужно использовать язык Java, то необходимо указать это пространсво имен:
- xmlns:java=http://xml.apache.org/xslt/java

Структура таблицы стилей

- *В этом примере показана структура таблицы стилей. Многоточие (...) обозначает место, где значения атрибутов или содержимое опущены. Хотя в примере показан каждый допустимый элемент, таблицы стилей могут содержать ноль или более каждого из этих элементов.*
- *<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:import href="..."/> <xsl:include href="..."/> <xsl:strip-space elements="..."/> <xsl:preserve-space elements="..."/> <xsl:output method="..."/> <xsl:key name="..." match="..." use="..."/> <xsl:decimal-format name="..."/> <xsl:namespace-alias stylesheet-prefix="..." result-prefix="..."/>*

Структура таблицы стилей (продолжение)

- *<xsl:attribute-set name="..."> ... </xsl:attribute-set>*
  <xsl:variable name="...">...</xsl:variable>
- *<xsl:param name="...">...</xsl:param>*
- *<xsl:template match="..."> ... </xsl:template>*
- *<xsl:template name="..."> ... </xsl:template>*
  </xsl:stylesheet>
- *Порядок, в котором появляются потомки элемента xsl:stylesheet, не имеет значения, за исключением элементов xsl:import и обработки ошибок. Пользователи могут произвольно упорядочивать элементы, и утилиты создания таблиц стилей не обязаны предоставлять контроль над порядком появления элементов*

Преобразование XML документов.

Обратимся к примеру с использованием стиля stylesheet:

- <xsl:stylesheet xmlns:xsl="">
- <xsl:template match="/">
- <html> <body>
- <xsl:value-of select="employee/name"/> <xsl:text>: </xsl:text>
- <xsl:value-of select="employee/title"/> </body>
- </html>
- </xsl:template>
- </stylesheet>

Стиль может находиться либо в файле, либо в базе данных:

Описание стилей

- Стиль содержит набор правил шаблона.
- Правило шаблона состоит из двух частей: образца (pattern) и шаблона (template).
- Образец сопоставляется с узлами в исходном дереве (resource tree).
- Шаблон обрабатывается для формирования фрагментов в конечном дереве (result tree). Такая схема позволяет использовать единый стиль для большого числа документов с одинаковой структурой исходного дерева

Применение шаблонов

- Примеры образцов выбора (math pattern):
- “/” -выбирает корневой узел
- “*” -выбирает элементы узлов (но не всех узлов);
- “library” -выбирает элементы library;
- “library/book”- выбирает все элементы book, дочерние для элемента library;
- “//library” выбирает все элементы library, производные от корневого узла;
- “.”- выбирает текущий узел но это не образец выбора, а выражение XPath/

Создание шаблонов

- Когда процессор XSLT находит узел, удовлетворяющий образцу, этот узел становится контекстным узлом шаблона и все операции производятся над этим узлом.
- На текущий узел можно ссылаться при помощи выражения XPath “.”
- В соответствии с правилами шаблона по умолчанию для обработки дочерних узлов необходимо использовать элемент
- “xsl:apple-templates”.

Синтаксис шаблона

- <xsl:template math= “nodename”>
- ……
- </xsl:template>
- Шаблон основан на имени ветви, поэтому все команды стилей применимы в этом шаблоне.
- Шаблон вызывается из используемого стиля командой apply-templates
- <xsl:apply-templates select=“nodename”/>

Значения типов элементов

- Команда <xsl:template match=“/”> означает выбор корневой ветви;
- Команда <xsl:apply-template> позволяет произвести новый поиск шаблонов для «текущего» узла;
- Команда <xsl:value of select..> дает возможность выбрать определенный элемент для анализируемого конечного XML документа.

Значения типов элементов

- Элемент <xsl:value of> записывает в результирующий документ строковое значение выражения. В частности, с его помощью можно возвратить значение узла, которым для элемента будет заключенный в него текст. Атрибуту выбора элемента <xsl:value of> можно присвоить выражение XPath, задающее узел или набор узлов

Значения типов элементов

- Паттерны позволяют выбирать определенные элементы из XML документа. Команда <xsl:value-of select... дает возможность выбрать требуемый элемент для конечного XML документа, как показано в таблице:

команда	результат
<xsl:value-of select= "employee/name"/> <xsl:value-of select= "employee/@id"/>	Joe Shmo 03432

Преобразование XML документов. пример

- Фрагмент XML документа имеет вид
- <employee id="03432">
- <name>Joe Shmo</name> <title>Manager</title>
- </employee>
- Если мы хотим чтобы наш HTML выглядел следующим образом:
- <html> <body> Joe Shmo: Manager </body> </html>

Преобразование XML документов.

Нужно использовать стиль stylesheet:

- <xsl:stylesheet xmlns:xsl="">
- <xsl:template match="/">
- <html> <body>
- <xsl:value-of select="employee/name"/> <xsl:text>: </xsl:text>
- <xsl:value-of select="employee/title"/> </body>
- </html>
- </xsl:template>
- </stylesheet>

Работа с XSLT (1)

- Версия planets.xml для Intertnet Explorer.6.0:
- <? Xml version=“1.0”?>
- <? Xsl: stylesheet type=“text/xsl” href =
  =“planet.xsl”?>
- <planets>
  - <planet>
    - <NAME>Venus </NAME>
    - <MASS UNITS=“(Earth=1)”>.815</MASS UNITS>
    - <DAY UNITS=“days”>58.65</DAY>
    - <RADIUS UNITS=“miles”>3716</RADIUS>
    - …….

  </planet> и т.д.

Работа с XSLT (2)

- <planet>
  - <NAME>Mercury </NAME>
  - <MASS UNITS="(Earth=1)">.0553</MASS UNITS>
  - ……
  - </planet>
- </planets>
- Версия **planets.xsl** для того же IE6.0 будет выглядеть следующим образом:

Работа с XSLT (3)

- <? xml version=“1.0”?>
- <? xsl: stylesheet xmlns: xsl=http://www.w3.org/TR/WD-xsl

  - <xsl: template math=“/”>
    - <HTML>
      - <HEAD>
        - <TITLE>The Planets Table </TITLE>

        </HEAD>

        <BODY>

        Прододжение на следующем слайде

Работа с XSLT (4)

```
<H11>The Planets Table</H1>
  <TABLE BORDER="2">
  <TR>
    <TD> Name</TD>
    <TD> Mass</TD>
    <TD> Radius</TD>
    <TD> Day</TD>
  </TR>
    <xsl: apply-templates/>
 </Table>
</Body>
</HTML>
</xsl: template>
```

Работа с XSLT (5)

- <xsl: template math="PLANETS">
- <xsl: apply-templates/>
- </xsl: template>
- <xsl: template math="PLANET">

```
<TR>
  <TD>xsl: value of select="NAME"/></TD>
  <TD> xsl: value of templates select ="Mass"/></TD>
  <TD> xsl: value of templates select =" Radius"/></TD>
  <TD> xsl: value of templates select =" Day"/></TD>
</TR>
</xsl: template>
```

Работа с XSLT

- Чтобы получить доступ к значениям атрибута при помощи выражений XPath, нужно добавить к имени атрибута префикс@. например “@MASS, “@Day” и т.д.Для выбора любого атрибута можно применить “@*”. Для выбора конкретного атрибута UNITS (ед. измерения) нужно в каждом элементе “MASS”,:RADIUS:, “DAY” использовать выражение “@UNITS”.

Работа с XSLT (6)

- <xsl: template math=“MASS”>
- <xsl: value of select=“.”/>

< xsl: value of select =“@UNITS”/>
</xsl: template>

- <xsl: template math=“RADIUS”>
- <xsl: value of select=“.”/>

< xsl: value of select =“@UNITS”/>
</xsl: template>
<xsl: template math=“DAY”>
<xsl: value of select=“.”/>
< xsl: value of select =“@UNITS”/>
</xsl: template>
</xsl: stylesheet>

Вид XMLдокумента в IE

NAME	MASS	RADIUS	DAY
VENUS	.815 (Earth=1)	3716	116
MERCURY	.0553 (Earth=1)	1516	58.65
EARTH	1 (Earth=1)	2107	1

Элемент “xsl:text”

- Текстовые узлы создаются при помощи элемента “xsl:text”, позволяющего по ходу дела замещать элементы целиком на текст. Одной из целей применения элемента “xsl:text”является сохранен6ие символов – разделителей; вставить единственный пробел можно с помощью атрибута «disable-output-escaping». Устанавливается в yes для того, чтобы такие символы как”<“ и “>”выводились буквально, а не как &lt и соответственно &gt/ По умолчанию устанавливается в no.

Элемент “xsl:text”

- Пример:
- <?xml vtrsion=“1.0”?>
- <?xsl: stylesheet version=“1.0” <xmlns: xsl=“http:// www.w3.org/1999/XSL/Transformxsl”?>
- <xsl:template match”/PLANETS”>
- <HTML>
  - <HEAD>
  - <TITLE>
    thse Planets Table
  - </TITLE>

</HEAD>

Элемент “xsl:text”

- <body>
  - <H1>
    Thse Planets Table
    </H1>
    <Table>
    <TD>NAME</TD>
    <TD>Mass </TD>
    <TD>Radius </TD>
    <TD>Day</TD>
  <xsl:apple tamplates/>
  </Table>
  </BODY>
  </HTML>
  <xsl:template>

Элемент “xsl:text”

- <xsl:template match=“PLANET”>
  - <TR>
    - <TD><xsl:value-of select=“NAME/>”><TD>
    - <TD><xsl:value-of select=“MASS/>”><TD>
    - <TD><xsl:value-of select=“RADIUS/>”><TD>

    </TR>

    </xsl:template>
- <xsl:template match=“MASS”>

  <xsl:value of select=“.”/>

  <xsl:text> </xsl:text>

  <xsl: value of select=“@UNITS”/>

  </xsl:template>

Элемент “xsl:text”

- <xsl:template match=“RADIUS”>
  <xsl:value of select=“.”/>
  <xsl:text> <xsl:text>
  <xsl: value of select=“@UNITS”/>
  </xsl:template>
- <xsl:template match=“Day”>
  <xsl:value of select=“.”/>
  <xsl:text> <xsl:text>
  <xsl: value of select=“@UNITS”/>
  </xsl:template>

</xsl: stylesheet>

Элемент “xsl:text”

NAME	MASS	RADIUS	DAY
MERCURY	0533 (Eath=1)	1.516 miles	58.65 days
VENUS	615 (Eath=1)	3716 miles	116.75 days
Eath	1 (Eath=1)	2107 miles	1 days

XSLT и XPath

- С точки зрения XSLT документы представляют собой образованные из узлов деревья.
- XSLT распознает 7 типов узлов:
- Корневой узел. Представляет для процессора XSLT весь документ. Следует различать корневой узел и корневой элемент;
- Узел атрибута;

XSLT и XPath (2)

- Узел комментариев;
- Узел элемента;
- Узел пространства имен;
- Узел инструкции обработки;
- Текстовый узел

XSLT и XPAth

- XSLT использует язык выражений XPATH для:
- Выбора узлов для обработки;
- Формулирования условий для разных вариантов обработки узла;
- Генерации текста для подстановки в конечное дерево.
- Выражение должно соответствовать сценарию Expr из Xpath.

Язык выражений XPATH

- Выражения используются:
- В значении определенных атрибутов у элементов, описанных в XSLT
- В фигурных скобках – в шаблоне для значений атрибута
- Внешние выражения – не являются частью других выражений; они получают свой контекст следующим образом:
- Узел контекста получается из текущего узла;

Язык выражений XPath

- Положение в контексте определяется положением текущего узла в текущем наборе узлов, первая позиция имеет индекс 1;
- Размер контекста определяется размером текущего набора узлов;
- Выражение XPath возвращает единственный удовлетворяющий выражению узел или множество узлов, если таких узлов несколько.

XSLT и XPath

- Основная задача XPath – адресовать части документа XML. Для реализации этой цели он предоставляет основные средства оперирования: строками, числами и логическими значениями.
- Синтаксис XPath отличен от синтаксиса XML, что облегчает его применение в идентификаторах URI и значениях атрибутов XML. Для навигации по иерархической структуре документа XML в нем используется нотация пути (patch) , как в идентификаторах URI.

Язык выражений XPath

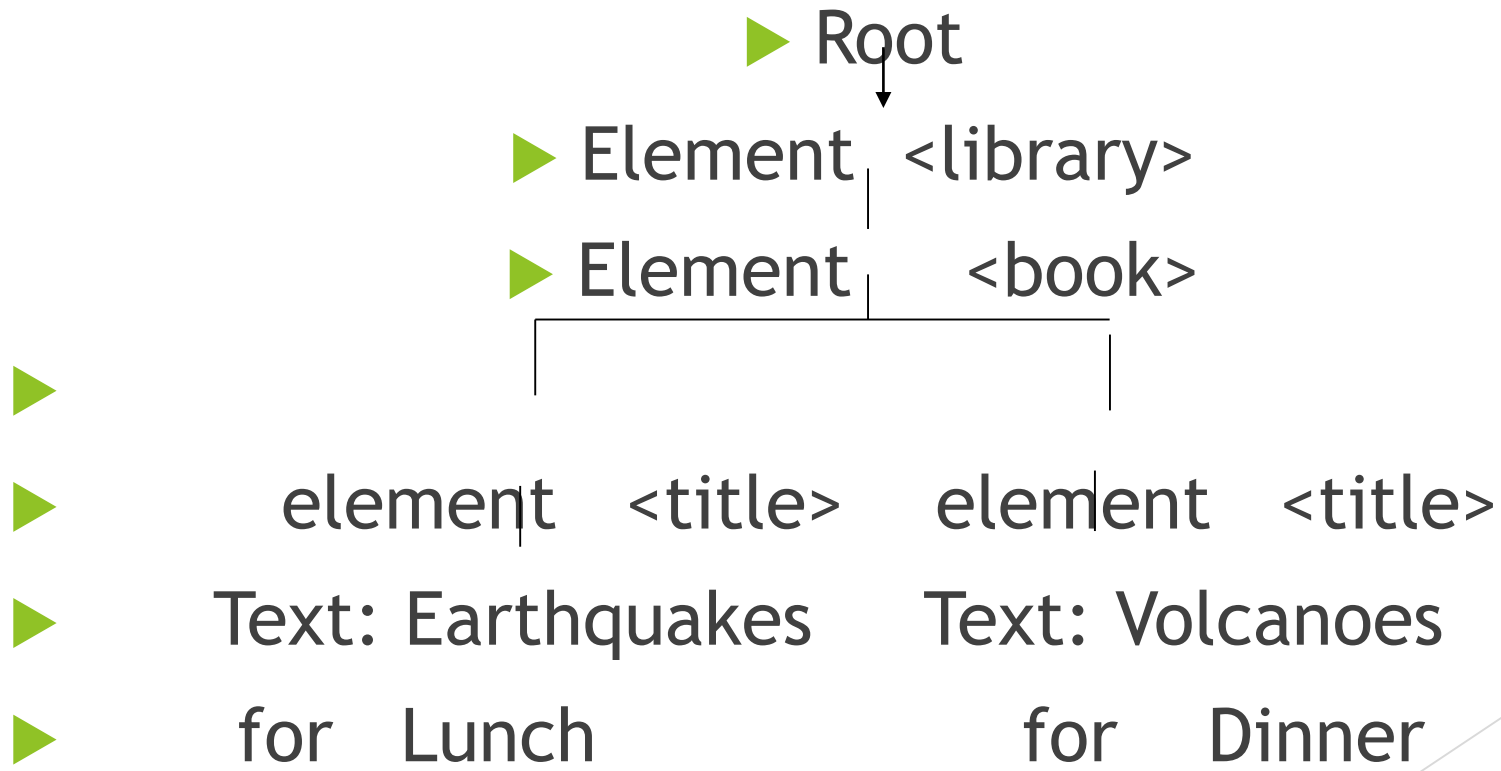
- XPath предоставляет средства для выбора набора узлов (node set), а также чисел, логических значений и строки.
- Полный синтаксис выражений XPatch приведен в спецификации XPatch

Язык выражений XPath

- Рассмотрим пример:
- <?xml version=“1.0”?>
- <library>
  - <book>
    - <title> Earthquakes for Lunch </title>
    - <title>Volcanoes for Dinner </title>
  - </book>
  - </library>

Язык выражений XPath

- Данный XML будет выглядеть в процессоре XSLT в виде дерева, состоящего из узлов:



Язык выражений XPath

- В этом дереве
- Root --Корневой узел;
- Library --Узел корневого элемента;
- Узел book имеет два дочерних узла title;
- Они же внуки для узла library;
- Родители и их родители – предки, а производные от них узлы - потомки .

Язык выражений XPath

- Выражения XPath применимы в XSLT не только в образцах выбора, но и в других приложениях: в атрибуте select элементов «xsl: apple_template», «xsl: value_of», «xsl:for-each», «xsl: param».«xsl: variable», в шаблонах значений атрибутов, в атрибуте test элементов «xsl: if», «xsl: when», атрибуте value элемента «xsl: number» и в предикатах образцов выбора.