22. Използване на XML за структуриране, валидация, обработка и представяне на документно съдържание. (XML)

Анотация: Изложението по въпроса трябва да включва следните по-съществени елементи:
1. Добре структуриран XML – основни концепции, XML йерархии, синтактични правила. XML пространства от имена.
2. XML валидация чрез Document Type Definition (DTD) – цели на валидирането, DTD структура, синтаксис.
3. XML валидация чрез XML Schema – спецификации, типове данни, фасети, структури. Сравнение с DTD.
4. Използване на XSLT (eXtensible StyleSheet Language Transformations) и XPath за алокиране, манипулиране и представяне на XML съдържание.
5. Използване на DOM (Document Object Model) и SAX (Simple API for XML) за обработка на XML документи – основни интерфейси на DOM и SAX и начини за използването им. Сравнение между DOM и SAX.

1. Добре структуриран XML

1.1. Основни концепции

Един добре структуриран XML има следните характеристики:
- Съдържание – информация, съдържаща текст, графика, аудио или видео;
- Структура – подялба и последователност на отделните тагове;
- Оформление – онагледяване на съдържанието и структурата на документ;
- Метаданни – описание на семантиката на съдържанието.

Един XML документ също така има две представяния на неговата структура. От едната страна стои логическата структура. Под логическа структура, имаме предвид логическата последователност и структура на елементите в XML документа. Например, логично би било един таг <father> да има под-тагове <son>, <daughter>, <child>, а не те да бъдат извън него. От друга страна, имаме физическа структура. Физическата структура на XML документ представлява единиците, от които е създаден документа. Един XML документ може да е създаден от множество XML под-документи или той самият съдържа други единици като например изображения.

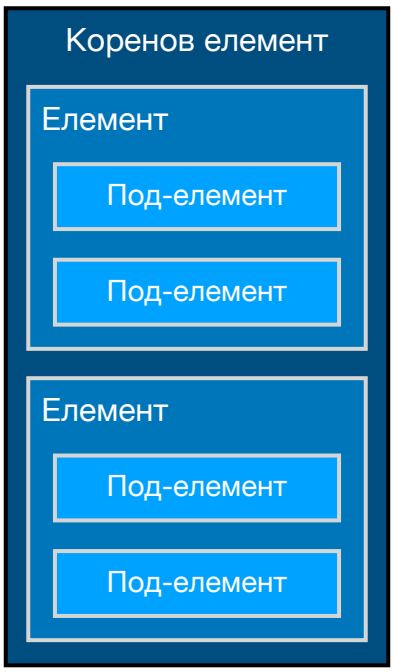
Всеки XML документ е изграден от следните компоненти:
- Маркер (таг) – име на елемент, заградено със символите „<“ и „>“ от двете страни. Например <parent>
  - Отварящ таг – започва дефиниция на елемент – <tag>
  - Затварящ таг – завършва дефиниция на елемент – </tag>
- Елемент – отварящ и затварящ таг, като между тях има някакво съдържание (текстово или други XML елементи). Елементите могат да бъдат влагани, като единствено един елемент може да бъде корен.
- Атрибут – допълнителна информация за елемент. Задава се като наредена двойка име-стойност. Стойността на атрибута се огражда от еднинични или двойни кавички.

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

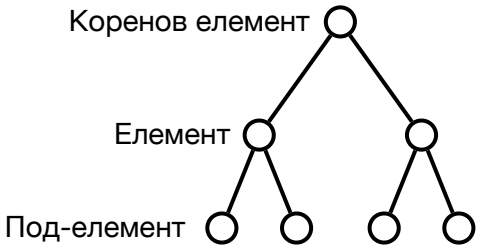
1.2. XML йерархии

Елементната структура на един XML документ представлява йерархия. Тя може да се представи по два начина: планарно и дървовидно.

Планарната йерархия на един XML документ представлява вертикално представяне на подредбата и релациите на елементите. Тя използва влагане за представяне на това, че вложеният елемент е дете на елемента, в който е вложен, Едно такова представяне е следното:



Дървовидната йерархия на един XML документ използва дървовидната структура от дискретните структури за по-добро представяне на релацията „родител-дете“, както и за по-добрата визуализация на онези елементи, които се намират на едно и също ниво спрямо влагането. Корен на дървото е един елемент, след което се разделя на елементите, които се намират в корена. Следващото ниво се състои от елементите, които се намират в елементите под корена и т.н. до листата, които представляват най-вложените елементи в дадена йерархия и в тях няма други вложени елементи.



В една XML йерархия, всеки отварящ таг трябва да има съответстващ затварящ таг, тъй като в противен случай йерархоята на отворения елемент няма да има дъно. Освен това, два или повече елемента не бива да се припокриват – тоест един елемент да започва в друг и да завършва след завършването на елемента, в който е започнал (аналогично на балансиран израз от скоби).

1.3. Синтактични правила

Елементите имат следните синтактични правила:

- Имената в XML правят разлика между големи и малки букви, т.е. <person> и <Person> са два различни елемента;
- Имената трябва да започват с малка или главна буква или „_“;
- След началната буква, могат да бъдат използвани букви, цифри, „_“, „–“ и „.“;
- Имената не могат да започват с думите XML, xml, Xml, xmL и т.н.;
- Празни пространства се допускат само след имената на елементите, т.е. преди знака „>“;
- Използването на символите „<“ и „>“ е резервирано за разделители;
- Символите „&“ и „;“ се използват за реферираме към съдържание на предефинирана единица (entity).

Атрибутите имат следните синтактични правила:
- Всеки атрибут трябва да има стойност, дори и тя да бъде празен стринг.
- За разделител между два и повече атрибута се използва празно пространство.
- Възможно е едновременно използване на кавички и апострофи за ограждане на стойностите на атрибутите на един елемент. Не е допустимо обаче, ограждането на стойност на един атрибут с кавички и апостроф или обратното.
- Дублиращи се атрибути в един и същ елемент не са разрешени.
- Атрибутите могат да следват произволен ред на подреждане в елемента.

1.4. XML пространства от имена

Използването на еднакви имена на XML елементи в различен контекст, който не е явно указан, може да доведе до проблеми с дублирани имена. Такива колизии могат да възникнат при:
- Смесване в един XML документ на маркирано съдържание от източници с различен контекст;
- Работа с един и същ XML документ в различни обкръжения, където имената на елементите се интерпретират по различен начин.

Това налага употребата на пространства от имена за елементите и атрибутите. Пространство от имена е абстрактна нотация за група от имена. В рамките на един XML документ дадено пространство от имена може да се означава с префикс (освен, ако не е пространството по подразбиране). За осигуряване на единствеността на префиксите, означаващи пространства от имена, се използва само URL. Чрез URL се задава уникален адрес на Уеб ресурс и начин за достъп до него. Не е задължително условие означеният чрез URL ресурс да съществува в интернет.

2. XML валидация чрез Document Definitions (DTD)

2.1. Цели на валидирането

Елементите и атрибутите на XML документите определят неговия речник. Използвайки DTD, лесно може да се валидират XML документи спрямо дефиниран речник от елементи и атрибути. За да се провери дали съдържанието на даден XML документ е валидно според декларациите в съответния DTD документ, може да се използва XML парсер. DTD задава правила, съгласно които се определят имената на елементите и атрибутите, тяхната последователност, честота на срещане и т.н.

Предимствата на валидацията са:
- осигуряване на начин за проверка за коректност на дикументната структура или съдържание;
- многократно използване на едно описание на документна структура за множество XML документи (инстанция на даден документен тип);
- Описание на правилата за структуриране на документа, изграждащите го елементи, възможните типове и атрибути и стойностите им по подразбиране.

2.2. DTD структура

DTD се разполагат във външни файлове, в текущия XML файл или и в комбинация от двете. Състоят се от:
- Document Type Definition (DOCTYPE), който информира parser-а, че с документа има асоциирано DTD. Задължително се поставя в началото на документа;
- Тяло – съдържа декларациите на елементи, атрибути, entities и нотации;
- Блок на декларациите на DTD заграден с [ ].

Когато DTD документа се намира вътре в XML документа, тоест е част от него, се добавя следното:

- **Елемент – <!ELEMENT ... >**
  - Използва се за декларация на нови елементи и модела на тяхното съдържание;
  - Състои се от три части – ключовата дума ELEMENT, името на елемента и модела на съдържанието;
  - Съдържанието може да бъде:
    - Елементно – тоест съдържа други елементи, които трябва да имат дефиниция в DTD. Например: <!ELEMENT person (father, mother, children)>;
    - Смесено – в тази декларация ключовият елемент #PCDATA трябва да предхожда всички други елементи. Различните възможности за съдържание се отделят чрез знака „|“. Например <!ELEMENT person (#PCDATA | a | b)>;
    - Празно – указва се с ключовата дума EMPTY;
    - Произволно – указва се с ключовата дума ANY.
  - В дефинициите на съдържанието може да използваме и индикатори за брой срещания:
    - + указва, че даденият елемент трябва да се съдържа 1 или повече пъти: [1, ...];
    - * указва, че даденият елемент трябва да се съдържа 0 или повече пъти: [0, ...];
    - ? указва, че даденият елемент трябва да се съдържа точно 1 път или да не се съдържа: [0, 1].
  - **Attribute – <!ATTLIST ...>**
    - Атрибутите могат да бъдат прикрепени към декларацията на елементите или да бъдат в отделна декларация;
    - Декларацията се състои от три части:
      - Ключовата дума ATTLIST;
      - Името на елемента, на когото тези атрибути принадлежат;
      - Списък с декларациите на атрибутите – за всеки атрибут се дават име, тип и декларация на стойността.

        Декларацията на стойността може да бъде – стойност по подразбиране, фиксирана стойност, изискваща се (#REQUIRED) или опционална (#IMPLIED).

      - Някои възможни типове на атрибутите:
        - CDATA – указва, че атрибутът има стойност от символи (Character Data), които няма да се интерпретират от парсера;
        - ID – указва, че стойността на атрибута служи като ключ на елемента и той може да се идентифицира еднозначно чрез нея;
        - IDREFS – стойността на атрибута е списък от IDREF стойности, които са разделени с празно място;
        - ENTITY – стойността на атрибута е референция към външна единица, съдържаща не-XML съдържание;
        - ENTITIES – списък от ENTITY стойности, разделени с празно място.
  - **Единици – <!ENTITY ...>**
    - Една единица се дефинира чрез специална декларация (ENTITY) и се използва през референция, която я указва;
    - В един XML документ, ENTITY референциите се използват като указател към текст или външен ресурс;
      - Типове единици:
        - Общи (General) – декларирани в DTD, реферирани в XML документите;
        - Параметрични – декларирани в DTD, реферирани единствено в декларациите на DTD;
        - Вътрешни (Internal) – съхраняват се в XML документа и представляват текстово съдържание. Те реферират текст, който е дефиниран в DTD документа;
        - Външни (External) – съхраняват се извън XML документа и представляват или текстово, или двоично съдържание.
        - Външни единици, съдържащи не-XML съдържание. Те служат за вграждане на не-XML данни (като например графика). Използваме външно unparsed entity. Те реферират към данни, които XML процесора не бива да обработва.
  - **Нотация – <!NOTATION ...>**
    - Нотацията се използва, за да специфицира данни като например файлове от тип image/gif, image/jpeg и други

2.3. Синтаксис

```
<?xml version="1.0" ... ?>
<DOCTYPE root-name [
        <!ELEMENT newspaper (article+)>
        <!ELEMENT article (heading, body, notes)>
        <!ELEMENT heading (#PCDATA)>
        <!ELEMENT body (#PCDATA)>
        <!ELEMENT notes (#PCDATA)>

        <!ATTLIST article author CDATA #REQUIRED>
        <!ATTLIST article editor CDATA #IMPLIED>
        <!ATTLIST article date CDATA #IMPLIED>

        <!ENTITY newspaper "Vervet Logic Times">
        <!ENTITY publisher "Vervet Logic Press">
        <!ENTITY copyright "Copyright 1988 Vervet Logic Press">

        <!NOTATION jpg PUBLIC "JPG 1.0">
]>
```

3. XML валидация чрез XML схема (XML Schema)

XML Schema е XML документ, който описва структурата на друг XML документ. Тя е наследник на DTD, но предоставя много по-богати възможности за по-прецизна спецификация на XML структура. Най-разпространеният език за дефиниране на XML схема носи същото име – XML Schema. XML Schema използва XML-базиран синтаксис, което улеснява работенето с него.

3.1. Спецификации

Спецификацията на XML Schema е разделена на три части:

- Част 0: Въведение – включва въведение в основните концепции и не е със задължителен характер.
- Част 1: Структури – дефинира структурите, които се позлват в XML Schema.
- Част 2: Типове данни – специфицира какви типове могат да се използват за елементи и атрибути.

Структури
Част 1 на спецификацията на XML Schema дефинира следното:

- Дефиниции на типове – <simpleType>, <complexType>
- Декларации на атрибути – <attribute>, <simpleType>
- Декларации на елементи – <element>, <simpleType>, <complexType>
- Декларации на групи от атрибути – <attributeGroup>, <attribute>
- Дефиниции на моделиращи групи – <group>, <element>, <all>, <choice>, <sequence>
- Декларации на нотации – <notation>
- Анотации – <annotation>, <appinfo>, <documentation>

Декларациите се използват за онези компоненти, които ще се използват в рамките на XML документа, а дефинициите – за онези компоненти, които ще се използват само в рамките на схемата (например се дефинира нов тип).

Съществуват два вида декларации: глобални и локални. Глобалните са тези, които са директни деца на <xs:schema> елемента, а локалните имат друг родителски елемент.

Сега нека разгледаме и гореизброените елементи, части от спецификацията на XML Schema:

- <xs:schema> – коренът на всеки схема документ.
  - targetNamespace – това е атрибут на този елемент. Указва пространството от имена, за което тази схема е предназначена.
- <xs:element> – чрез него се декларират елементи, които ще бъдат използвани в XML документа.
- <xs:attribute> – използва се за декларация на атрибути. Най-честите му употреби са в рамките на <attributeGroup> и <complexType>.
- <xs:simpleType> – дефинира елемент от прост тип. Елементите от прост тип в една XML Schema са елементите, които са от даден предефиниран тип и нямат атрибути и под-елементи. Самото създаване на нов прост тип е на базата на съществуващите, като се обединяват различни типове или се добавят ограничения върху тях.
- <xs:complexType> – дефинира елемент от сложен тип. За целта съответният сложен тип трябва да бъде дефиниран и именован извън елемента или да бъде специфициран като под-елемент. Сложните елементи могат да бъдат както празни елементи (без съдържание и под-елементи), така и елементи само с атрибути, само с под-елементи или с текстово съдържание и атрибути.
- <xs:attributeGroup> – позволява декларацията на група от атрибути. Ако се декларира на глобално ниво – трябва да има име.
- <xs:group> – позволява да се групират елементи и така получената група да се преизползва на различни места. В една група могат да присъстват единствено елементи. Редът на елементите в групата се задава с помощта на долните три декларации.
  - <xs:all> – съдържа декларации на елементи, които трябва да присъстват в XML документа, независимо в какъв ред;
  - <xs:choice> – съдържа декларации на елементи, от които точно един може да присъства в XML документа;
  - <xs:sequence> – съдържа декларации на елементи, които трябва да присъстват в XML документа в точно указания ред (последователно).

3.2. Типове данни

Има две групи типове данни:
- Примитивни (основни) – служат за основа при дефинирането на всички останали типове, например:
  - string
  - boolean
  - float
  - double
  - decimal
  - NOTATION
  - ID
  - duration
  - dateTime
  - time
  - date
  - и други
- Вградени производни – използвани са други примитивни и производни типове, за да бъдат дефинирани. Те се изграждат на базата на <xs:simpleType> и <xs:complexType>.

3.3. Фасети

Фасет е определящ аспект на пространството от стойности за даден тип. Фасетите на даден тип данни служат да се разграничат онези аспекти на този тип, които се различават от други типове данни. Един фасет е свойство или характеристика на <xs:simpleComplex>.

Съществуват два вида фасети:

- Фундаментални фасети – дефинират типа
  - Equal – важи за всички типове данни;
  - Order – за някои типове;

    - Bounds – за долна и горна граница;
    - Cardinality – крайност или безкрайност;
    - Numeric – дали е число или не.
- Ограничаващи фасети – ограничават стойностите на типа
    - minInclusive/minExclusive – минимална допустима стойност, включително или не и самата стойност;
    - maxInclusive/maxExclusive – максимална допустима стойност, включително или не и самата стойност;
    - length – определя броят елементи/символи в даден елемент;
    - minLength/maxLength – определя минималният/максималният брой елементи/символи в даден елемент;
    - pattern – ограничава низов тип чрез регулярни изрази;
    - enumeration – изброяване на допустими стойности;
    - precision – броят на цифрите от дясно на десетичната запетая в едно число с плаваща запетая;
    - scale – броят на цифрите от ляво на десетичната запетая в едно число с плаваща запетая.

3.4. Структури

```
<!-- Type definitions : <simpleType>, <complexType> -->
<!-- Attribute declarations : <attribute> -->
<!-- Element declarations : <element> -->
<!-- Attribute group definitions : <attributeGroup> -->
<!-- Notation declarations : <notation> -->
```

3.5. Сравнение с DTD

Чрез XML Schema може да зададем много по-стриктни правила спрямо максимално постижимото чрез DTD. Тъй като XML е по-близък до XML като синтаксис, той е по-интуитивен от DTD.

- DTD и XML Schema (или XML Schema Definition (XSD)) задават правила за структуриране на XML документи;
- DTD използва по-стегнат и кратък синтаксис от XSD;
- XSD се дефинира на чист XML;
- XSD предоставя по-богат набор от възможности за по-строго дефиниране на XML схемата.

4. Алокиране, манипулиране и представяне на XML съдържание

4.1. Използване на XSLT (eXtensible StyleSheet Language Transformations)

XML документите обикновено са предназначени за компютърна обработка, а не за да бъдат четени от хора. Често се налага XML съдържанието да се представи в различен вид в зависимост от това за кого или какво е предназначено. За тази цел се появява XSL, които се състои от две части: XSL-FO и XSLT. Първият служи за описание на форматирането на данните в XML документи, а вторият – за трансформиране на XML документи с помощта на различни стилове и функции. Най-честата употреба на XSLT е за конвертиране на документ от XML формат към документ в HTML формат, обикновен текстов файл или пък друг XML файл. Той е полезен, когато искаме да разделим презентационния слой на едно приложение от модела на данните му.

В XSLT има дефинирани 34 елемента, част от които ще представим тук:

- <xsl:stylesheet> – задава обхват на XSLT документа и осигурява конфигуриране на параметри;
- <xsl:template> – специфицира кои елементи трябва да бъдат обхванати от обработката. Дефинира какво ще бъде добавено в изходния документ;
- <xsl:apply-templates> – взима решение кои елементи ще бъдат обработени;
- <xsl:value-of> – изчислява израз и добавя резултата в изхода;

- <xsl:for-each> – осигурява групова обработка на елементи по еднотипен начин;
- <xsl:output> – позволява контрол над изходния документ. Разполага се веднага след кореновия елемент <xsl:stylesheet>;
- <xsl:number> – автоматично номериране;
- <xsl:variable> – за деклариране на променлива. Достъпва се чрез символа „$“, последван от името на променливата. Стойността на променливата може да се използва и в елементи от резултатното дърво. По този начин могат да се добавят прости константи;
- <xsl:element> – използва се за създаване на нови елементи и позволява динамично създаване на елементи;
- <xsl:attribute> – за създаване на атрибути;
- <xsl:copy> – копира съдържанието на контекстния възел без вложените елементи и атрибутите;
- <xsl:copy-of> – може да копира фрагменти от входното дърво без да се загубват атрибутите и вложените елементи.

В един XML документ, XSLT се задава посредством инструкцията:
<?xml-stylesheet type="text/xsl" href="stylesheet.xsl"

XSLT предоставя следните възможности:
- Структурни промени на входното съдържание;
- Многократно преизползване на елементно или атрибутно съдържание на друго място в документа;
- Трансформиране на данни между XML формати;
- Определяне на XSL форматиращи обекти и на други средства за представяне на съдържанието в дадена медия (например CSS).

XSLT може да се имплементира в клиентската част на едно приложение, Ако се иполза браузър XML документът се транслира от XHTML.

4.2. XPath

Xpath служи за адресиране и манипулиране на секции от XML документ. Той се използва от други XML спецификации като XPointer и XSLT. Локацията на дадена секция се задава чрез URL. XPath работи с интернет, интранет и файловата система.

В XPath 1.0 модела, повечето съставни части от XML документа са представени като възли на дърво, свързани с определени отношения. Коренът на XPath 1.0 дървото представлява самия документ и се бележи с „/“.

Един XPath израз е текстов низ, чрез който се избира множерство от възли в дървото.

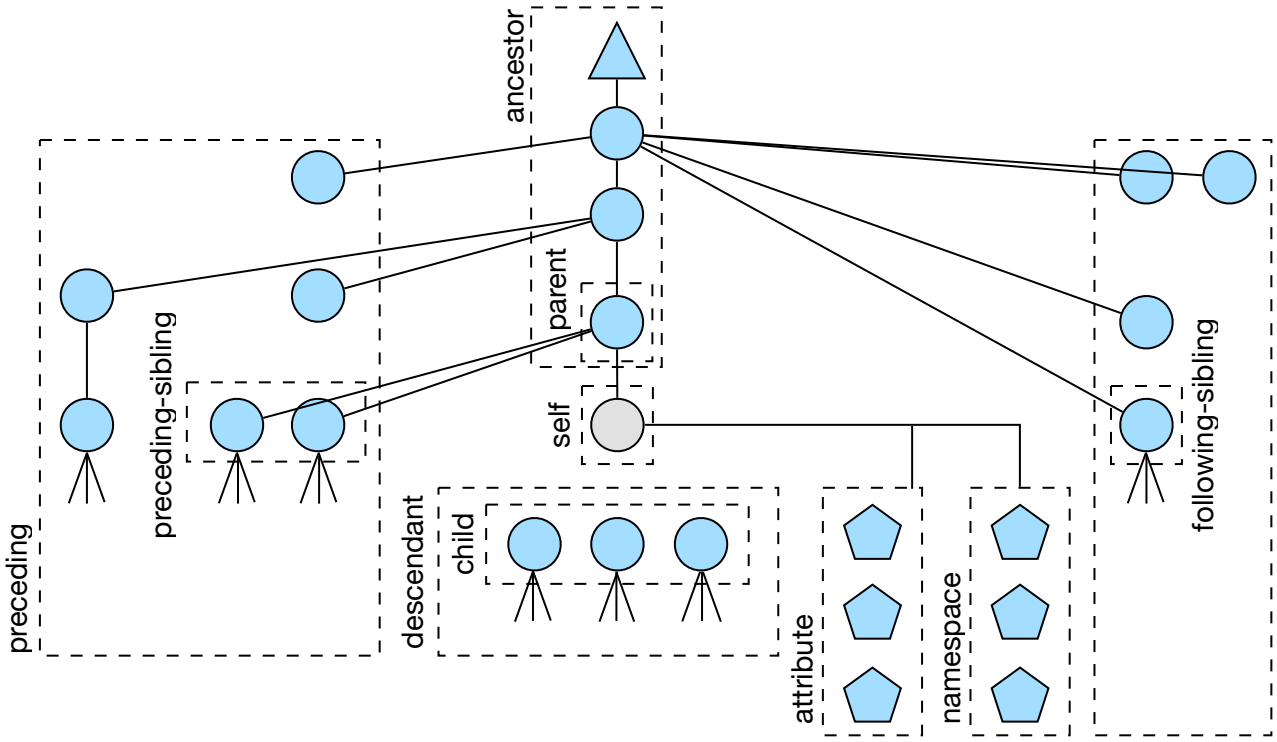
XPath процесорът изчислява XPath израз, който задава път от дадена начална точка в дървото. Чрез изчислението на този път като последователност от свързани възли, процесорът се „придвижва“ по дървото, моделиращо документа, до указания от израза възел. Този възел, в който за текущия момент се намира XPath процесорът, се нарича контекстен възел и се бележи с „.“.

Видове XPath възли:
- корен – сочи към корена на документа;
- елемент – представя елемент;
- текст – представя текстово съдържание на елемент;
- атрибут – представя атрибут;
- инструкция за обработка;
- декларация на CDATA секция (област от данни);
- коментар

Повечето XPath изрази връщат множество от върхове. Тези изрази се наричат местоположения. В XPath съществуват два вида пътища: относителни и абсолютни. Един относителен път на местоположение се състои от една или повече стъпки на местоположение, разделени със знака „/“. Те могат да започват с низа „./“, който ако е

пропуснат се подразбира. Тоест един относителен път започва от контекстния възел. Един абсолютен път на местоположение се състои от знака „/“, последван от относителен път. Тоест винаги започва от корена на дървото.

Нека разгледаме някои оси в XPath:



- Ос на предшествениците (ancestor axis) – избира предшествениците на контекстния възел в обратен ред на срещането им;
- Ос на директните наследнити (child axes) – избира преките наследници на контекстния възел;
- Ос на атрибутите (attribute axis) – задава всички атрибути на контекстния възел;
- Ос на наследниците (descendant axis) – връща всички наследници (преки и непреки) на контекстния възел по ред на появата им в документа;
- Ос на следващите възли (following axis) – селектира всички следващи възли след контекстния възел;
- Ос на следващите възли и на самия възел (following-sibling axis) – избира братя и сестри на контекстния възел, следващи в дясно от него;
- Ос на родителя (parent axis) – връща родителя на контекстния възел;
- Ос на предходните възли (preceding axis) – селектира всички предходни възли спрямо контекстния възел, без наследниците и атрибутите му и без пространствата от имена;
- Ос на предходните възли и на самия възел (preceding-sibling axis) – избира всички предшественици братя и сестри на контекстния възел вляво от него;
- Ос на самия възел (self axis) – връща контекстния възел.

За работата с относителни и абсолютни пътища съществуват два вида синтаксис – разширен и кратък. Разширеният синтаксис не е удобен за работа и затова XPath използва краткия синтаксис, който наподобява адресирането на директории на файлови системи. Краткият синтаксис използва означения като „@“ за атрибут и „*“ за избор на всички възли. Родителят на контекстния възел се адресира чрез „..“, а текущия – чрез „.“ – точно както в пътищата при файловите системи.

5. Обработка на XML документи

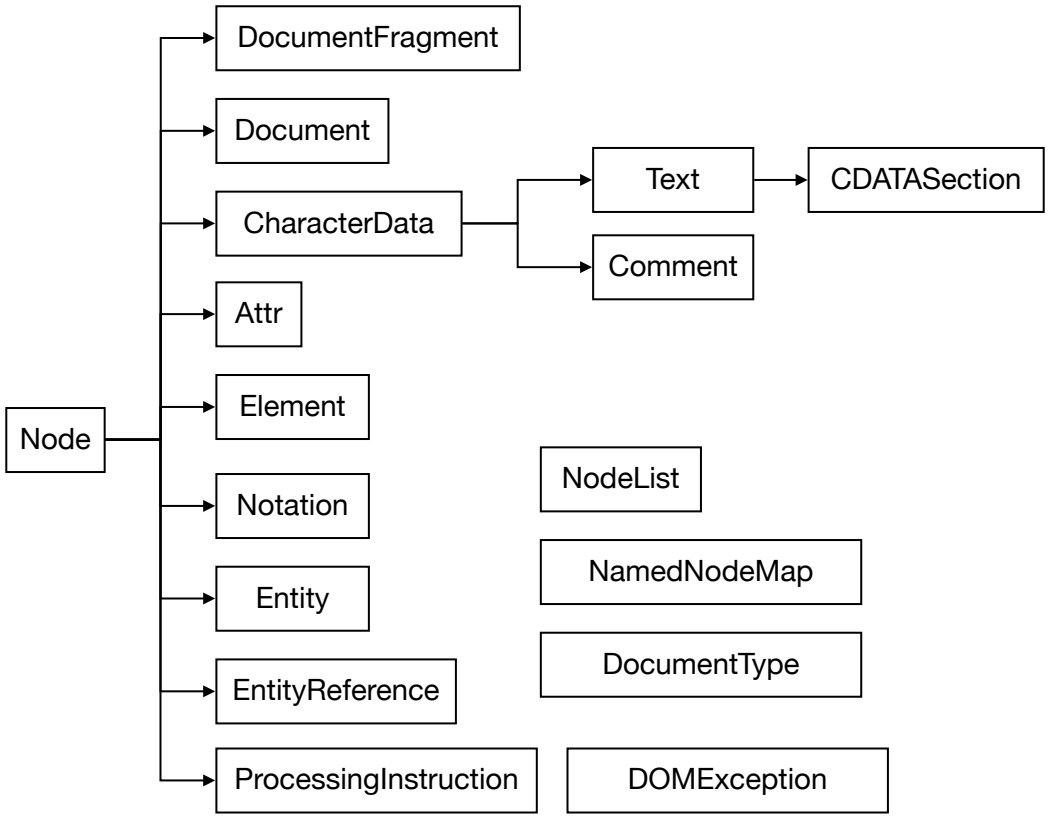
5.1. Основни интерфейси и начини за използване на DOM (Document Object Model)

DOM в W3C стандарт, който дефинира стандартен начин за работа с документи като XML и HTML. XML DOM дефинира обекти и свойства за всички XML елементи, както и интерфейс за достъп до тях. Най-честата употреба на XML DOM е за манипулирането на съдържанието на XML документи – може да се създава ново съдържание, да се навигира, да се модифицира или пък да се добавят или изтриват части от XML документи. В DOM един документ се представя като йерархия (дърво) от Node обекти.

Спецификацията на DOM се състои от 4 нива, като всяко надгражда предходното. Спецификацията включва основна задължителна част от интерфейси, която трябва да бъде налична във всички имплементации. Тя се нарича Core DOM и се състои от Ниво 2 и Ниво 3.

- Основни типове DOM интерфейси – състоят се както от базови интерфейси, задължителни за всички имплементации на DOM, така и от разширени интерфейси, необходими за DOM имплементации, работещи с XML документи.

- Базови интерфейси
  - Node – представлява един връх в документното дърво;
  - Document – представлява целия XML документ (корена на дървото). Съдържа и методи фабрики за създаване на останалите обекти;
  - DocumentFragment – фрагмент от Document може временно да бъде съхранен в DocumentFragment възел. Този възел се разрушава, когато се прикрепи към Node възел;
  - Element – представлява един елемент в рамките на даден XML документ. Докато Node е възел, който може да бъде елемент, текстово съдържание и т.н.. Element може да бъде само елемент.
  - Attr – интерфейс към обекти, съдържащи атрибутни данни:
    - Свойства name и value – връщат съответно името на атрибута и неговата стойност;
    - Атрибутите не са пряка част от дървовидната структура на документа.
  - CharacterData – съдържа полезни общи методи за текстообработка, но не се използват директно, а през класовете наследници Text или Comment;
  - Text – представя текстово съдържание на Елемент или Attr. Тези интерфейси винаги са възли-листа;
  - Comment – представя коментари;
  - NodeList – съдържа колекция от наредени Node обекти;
  - NamedNodeMap – дефинира неподредена колекциа от възли (Node обекти).
- Разширени интерфейси
  - CDATASection – представя CDATA секция;
  - DocumentType – информация за съдържанието на DTD;
  - ProcessingInstruction – представя декларация за инструкция за работа;
  - EntityReference – интерфейс за работа с референции към единици;
  - Entity – представя единица;
  - Notation – всяка декларация на нотация в DTD е представена чрез възел Notation.



5.2. Основни интерфейси и начини за използване на SAX (Simple API for XML)

SAX е event-driven API, който дефинира манипулатори (handlers), съдържащи методи за разбор (push) на XML документа. Разработен е за по-ефикасен анализ на големи XML документи (не се налага всеки път да прочита целите документи). Целта му е да реши проблема на DOM – създаването на масивно дърво на документа в паметта, преди да започнем работа с него. По този начин се пести както памет, така и време. Той осъществява това подобрение с управлението на събития. XML документът се изпраща до SAX парсера, след което документът се прочита ред по ред, като парсерът известява за грешки. След това имплементациите на интерфейсните методи обработват събитията. Самият SAX парсер използва callback механизъм за известяване на приложението. По този начин може да напишем код за обработката на всяко събитие.

- Основни SAX интерфейси:
  - DocumentHandler – за управление на събитията за съдържанието на документа в SAX1. Заместен от ContentHandler в SAX2;
  - ContentHandler – използва се за получаване на основни събития от парсера. Надгражда DocumentHandler, като добавя поддръжка и за пространства от имена. SAX предоставя имплементация по подразбиране наречена DefaultHandler. В нея може да пренапишем онези методи, които обработват важни за нас събития;
  - XMLReader – интерфейс за четене на XML документ с използване на callbacks. Той разрешава приложението да зададе и провери свойства за парсъра, да регистрира обработчици на събития за обработка на документа и да инициира самото парсване;
  - XMLFilterImpIt – имплементация на ContentHandler, която стои между XMLReader и обработчиците на събития в приложението. Предава събитията на обработчиците без промяна. Неговите подкласове могат да пренапишат специфични методи;
  - XMLReaderAdapter – обвива XMLReader и го прави като Parser от SAX1;
  - ErrorHandler – осигурява информация за възникнали грешки.
    - Три нива на изключения:
      - warning – известява приложението с предупреждение;
      - error – известява приложението за грешка;
      - fatalError – известява приложението за фатална грешка.
  - DTDHandler – предоставя методи за известяване за DTD събития;

- EntityResolver – определя поведението на SAX парсера при преобразуване на външни референции в рамките на DTD дефиниция.

5.3. Сравнение между DOM и SAX

- DOM разглежда XML като логическо дърво, докато SAX е базиран на събития;
- DOM е добър при обработка на много елементи и структурирани промени на документа;
- DOM е добър при многократен достъп до XML документа, тъй като не се налага да го прочита целия всеки път;
- SAX е по-ефикасен за анализ на цели големи документи, тъй като не се налага да ги прочита целите.