24. Използване на XML за структуриране, валидация, обработка и представяне на документно съдържание.

1. Добре-структуриран XML – основни концепции, XML йерархии, синтактични правила. XML пространства от имена.
2. XML валидация чрез Document Type Definitions (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 – основни концепции, XML йерархии, синтактични правила. XML пространства от имена.

XML (eXtensible Markup Language) е език за маркиране за създаване на структурирани документи, които могат да се четат от машини и хора. Един XML документ се състои от:

- **Таг**: Име на елемент, заградено между „<“ и „>“ (напр. <parent>).
  - **Отварящ таг** (<tag>): Започва дефиницията на елемент.
  - **Затварящ таг** (</tag>): Завършва дефиницията на елемент.
- **Елемент**: Отварящ и затварящ таг със съдържание (текст или други XML документи) между тях. Елементите могат да бъдат вложени, но само един може да бъде **коренов**.
- **Атрибут**: Допълнителна информация за елемент, дефинирана като **двойка име-стойност** със стойността, заградена в единични или двойни кавички.
- **XML декларация:** - <?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>

Пример:

```
<person gender=”male”>
        <firstName>Боян</firstName>
        <lastName>Покебончев</lastName>
</person>
```

В една XML йерархия **всеки отварящ таг трябва да им съответстващ затварящ таг.** Освен това, **два или повече елемента не могат да се припокриват,** т.е. Това не може <a>

Правила за синтаксис на XML елементи:

- Имената правят разлика между главни и малки букви (напр. `<person>` и `<Person>` са различни).
- Имената започват с буква или `_` и могат да съдържат букви, цифри, `_`, `-`, `.`.
- Имената не могат да започват с "XML" (без значение на case-а).
- Интервали са разрешени само след имената на елементите, преди `>`.
- Символите `<` и `>` са запазени за разделители.
- Символите `&` и `;` се използват за реферираме към предефинирана единица.

Правила за синтаксис на XML атрибути:

- Всеки атрибут трябва да има стойност (дори празен низ).
- За разделител между атрибути се използва интервал.
- Разрешено е едновременно използване на кавички и апострофи за ограждане на стойности, но не смесването им за един атрибут.
- Дублиращи се атрибути в един елемент не са разрешени.
- Редът на атрибутите е произволен.

XML пространства от имена - използва се за **разграничаване на елементи и атрибути**, които могат да имат еднакви имена, но различно значение. Синтаксис: Пространства от имена се дефинират с атрибута ***xmlns***:
<book xmlns="http://example.com/books">.

2. XML валидация чрез Document Type Definitions (DTD) – цели на валидирането, DTD структура, синтаксис.
Цели на валидирането

- **Верификация на структурата, проверка за коректност** - Уверява, че XML документите следват предварително дефинирана структура и правила.
- **Съответствие**: Гарантира, че документът е съвместим с определени стандарти;

Структура на DTD дефинира правила за имена, последователност и честота на срещане на елементи/атрибути.

Видове:

- **Вътрешен (вграден) DTD:** Включен в XML документа.
- **Външен DTD:** Отделен файл, посочен в XML.
    - Документите могат да използват и двете.

Деклариране на елемент: `<!ELEMENT element_name (content_model)>`

- **Модели на съдържание:**
    - **Елементарен:** Съдържа други дефинирани в DTD елементи (напр. `<!ELEMENT person (father, mother, children)>`).
    - **Смесен:** `#PCDATA` предхожда други елементи, разделени с `|` (напр. `<!ELEMENT person (#PCDATA | a | b)>`).
    - **Празен:** Използва `EMPTY`.

    - **Всякакъв:** Използва `ANY`.
- **Индикатори за съвпадение:**
    - `+`: Поне веднъж.
    - `*`: Нула или повече пъти.
    - `?`: Нула или един път.

Деклариране на атрибут:

- Може да бъде прикрепен към декларации на елементи или отделен.
- **Структура:** `ATTLIST` + име на елемент + списък с декларации на атрибути (име, тип, декларация на стойност).
- **Декларации на стойности:** По подразбиране, фиксирани, `#REQUIRED` или `#IMPLIED`.
- **Типове атрибути:**
    - **CDATA:** Неинтерпретирани от парсера символни данни.
    - **ID:** Уникален ключ на елемент.
    - **IDREFS:** Списък от стойности `IDREF`, разделени с интервали.
    - **ENTITY:** Препратка към не-XML външна същност.
    - **ENTITIES:** Списък от стойности `ENTITY`, разделени с интервали.
- **Пример:**
  <!ATTLIST book
  id ID #REQUIRED
  genre CDATA #IMPLIED
  >

Декларация на единици: <!ENTITY …> - използва се като указател към текст или към външен ресурс
Декларация на нотация: <!NOTATION …> - използва се за описване на данни, които не са XML, като например: снимки, анимации и др.
Пример:

```
<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 “Financial times”>
    <!ENTITY publisher “Sofia press”>

    <!NOTATION jpeg PUBLIC “image/jpeg”>
    <!ENTITY logo SYSTEM “logo.jpg” NDATA jpeg>
]>
```

3. Валидация на XML с XML Schema: Спецификации, типове данни, аспекти, структури. Сравнение с DTD.

XML Schema е XML документ, който **описва структурата на друг XML документ**. Той е наследник на DTD, **предлагайки по-богати възможности за прецизно специфициране** на XML структура.

Често срещани елементи на XML Schema:

- `<xs:schema>`: Корен на всеки документ на схема.
- `<xs:element>`: Дефинира елементи.
- `<xs:attribute>`: Дефинира атрибути.
- `<xs:simpleType>`: Дефинира елементи от прост тип.
- `<xs:complexType>`: Дефинира елементи от сложен тип.
- `<xs:attributeGroup>`: Позволява дефинирането на група от атрибути.
- `<xs:group>`: Позволява групирането на елементи за повторна употреба. Само елементи; редът се дефинира от:
    - `<xs:all>`: Елементите трябва да присъстват, в произволен ред.
    - `<xs:choice>`: Трябва да присъства точно един дефиниран елемент.
    - `<xs:sequence>`: Елементите трябва да присъстват в указания ред.

Типове данни

- **Прости типове**: integer, boolean, float, double, ID, date и т.н.
- **Сложни типове**: Дефинирани с помощта на други прости и сложни типове, изградени с `<xs:simpleType>` и `<xs:complexType>`.

Аспекти (Facets): Дефинират **разрешените стойности** за елементи и атрибути (напр. дължина, диапазон на стойностите, формат).

- **Фундаментални аспекти**: *Equal, Order, Bounds, Cardinality, Numeric*.
- **Ограничаващи аспекти**: minInclusive/minExclusive; length; minLength/maxLength; pattern (регулярен израз); enumeration; precision; scale.

Пример:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="author" type="xs:string"/>
        <xs:element name="year" type="xs:integer"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="language" type="xs:string" use="optional"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4. Използване на XSLT (eXtensible StyleSheet Language Transformations) и XPath за алокиране, манипулиране и представяне на XML съдържание
XSLT е език за **трансформация** на XML документи в други формати, като HTML, текст или друг XML. XSLT позволява на потребителите да **дефинират шаблони и правила за трансформацията**, базирани на структурата и съдържанието на оригиналния XML документ.

В **XSLT** има дефинирани 34 елемента като: <xsl:stylesheet>, <xsl:template>, <xsl:apply-templates>, <xsl:value-of>, <xsl:for-each>, <xsl:output>,, <xsl:number>, <xsl:variable>, <xsl:element>, <xsl:attribute>, <xsl:copy>, <xsl:copy-of> и др. Като например с помощта на <xsl:variable> можем да дефинираме променлива, а с помощта на <xsl:element> може да създадем нов елемент и позволява динамично създаване на елементи.

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

XPath е език за навигиране и селектиране на части от XML документи, представяйки компонентите на документа като възли на дърво. Коренът на документа се обозначава с „/“.

Изразът **XPath** е низ, който избира един или повече възли в дървото.

Типовете възли в XPath включват:
- root (корен)
- element (елемент)
- attribute (атрибут)
- text (текст)
- processing instruction (инструкция за обработка)
- CDATA declaration (декларация CDATA)
- comment (коментар)

XPath използва два типа пътища: относителни (започващи от контекстния възел, потенциално с „./“) и абсолютни (започващи от корена на дървото, с „/“).

XPath позволява избор на:
- елементи със специфична стойност на атрибут (напр. `library/book[@id='my_book']`)
- всички елементи от определен тип (напр. `library/book`)
- След като изберете елемент, можете да извлечете неговите преки наследници, родител, атрибути или следващи възли.

Синтаксисът на **XPath** е подобен на адресирането на директории:
- „..“ за родителския възел
- „.“ за текущия възел
- „*“ за избор на всичко
- „@“ за атрибут (напр. `library/book[@genre=”fiction”]`)"

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.

DOM представя XML документите като дървовидна структура, което позволява лесна навигация и модификация чрез програмни интерфейси. Въпреки че консумира повече памет (особено за по-големи документи), той е неразделна част от съвременните браузъри и динамичните уеб страници.

Основни интерфейси: Node (връх на дърво), Document (цял XML документ/корен), Element (специфичен XML елемент), Attr (име/стойност на атрибут), CharacterData (методи за обработка на текст), Text (листа на дърво), Comment, NodeList (колекция от възли), NamedNodeMap, DocumentFragment.

Разширени интерфейси: CDATASection, DocumentType (информация за DTD), ProcessingInstruction, EntityReference, Entity, Notation.

SAX (Simple API for XML) обработва XML документи въз основа на събития, което го прави ефективен за големи документи, тъй като не чете целия документ всеки път и консумира по-малко памет.

Основни SAX интерфейси: DocumentHandler, ContentHandler, XMLReader, XMLFilterImplt, XMLReaderAdapter, ErrorHandler (предупреждение, грешка, фатална грешка), DTDHandler, EntityResolver.

Сравнение между DOM и SAX:

- SAX е идеален за четене на големи XML документи и обработка в реално време.
- DOM разглежда XML като логическо дърво; SAX е базиран на събития.
- DOM е отличен за обработка на много елементи и извършване на промени в структурирани документи.