ЛЕКЦИЯ 9 – част 1

СЪДЪРЖАНИЕ

I. Работа с XML.

1. Общи сведения за XML
2. Прилики и разлики между XML и HTML
3. Дефиниране на пространства от имена
4. Схеми и валидация
5. XML и .NET Framework

I. РАБОТА С XML

I.1. Какво е XML ?

XML (Extensible Markup Language)

XML първоначално е замислен като език за дефиниране на нови доку-ментни формати за World Wide Web. XML произлиза от SGML (Standard Generalized Markup Language) и на практика е негово подмножество със значително опростен синтаксис, което прави внедряването му много по-лесно. С течение на времето XML се налага като markup език за структурирана информация.

XML е метаезик - за описание на markup (маркиращи) езици. Той няма собствена семантика и не определя фиксирано множество от тагове. Разработчикът на едно XML приложение има свободата да дефинира подходящо за конкретната ситуация множество от XML елементи и евентуални структурни връзки между тях.

XML е независим - Езикът XML е независим от платформата, езиците за програмиране и операционната система. Тази необвързаност го прави много полезен при нужда от взаимодействие между различни програмни платформи и/или операционни системи.

I. РАБОТА С XML

При електронните документи терминът **markup** описва специфичното обозначаване на части от документите с тагове. Таговете имат две основни предназначения – те описват изгледа и форматирането на текста или определят структурата и значението му (метаинформация).

Днес се използват два основни класа markup езици – специализирани и с общо предназначение (generalized) markup езици. Първата група езици служат за генериране на код, който е специфичен за определено приложение или устройство и адресира точно определена необходимост. Общите markup езици описват структурата и значението на документа, без да налагат условия по какъв начин ще се използва това описание. Пример за специализиран markup език е HTML, докато SGML и неговото функционално подмножество XML са типични markup езици с общо предназначение.

XML – пример 1

Следният пример демонстрира концепцията на markup езиците с общо предназначение. При тях структурата на документа е ясно определена, таговете описват съдържанието си, а форматирането и представянето на документа не е засегнато – всяко приложение може да визуализира и обработва XML данните по подходящ за него начин.

```
<?xml version="1.0" encoding="windows-1251"?>
<messages>
   <message>XML markup описва структура и съдържание</message>
   <message>XML markup не описва форматиране</message>
</messages>
```

XML – пример 2

Демонстрира един възможен начин за описание на книгите в една библиотека със средствата на XML. Информацията е лесно четима и разбираема, самодокументираща се и технологично независима.

```
<?xml version="1.0"?>
<library name=".NET Developer's Library">
   <book>
      <title>Programming Microsoft .NET</title>
      <author>Jeff Prosise</author>
      <isbn>0-7356-1376-1</isbn>
   </book>
   <book>
      <title>Microsoft .NET for Programmers</title>
      <author>Fergal Grimes</author>
      <isbn>1-930110-19-7</isbn>
   </book>
</library>
```

I.2. XML и HTML (прилики и разлики)

Външно езикът XML прилича на езика HTML, но между двата езика има и сериозни различия.

Прилики:

- ***и двата езика са текстово-базирани*** - XML и HTML са текстово-базирани езици и това осигурява прозрачност на информационния формат. При нужда такива документи могат да се отварят и редактират с помощта на обикновен текстов редактор.
- ***и двата езика използват тагове и атрибути*** - Двата езика използват елементи, всеки от които се състои от отварящ и затварящ таг (например <book> и </book>) и информация между тях (представяща съдържанието на елемента). Всеки елемент може да дефинира свои атрибути, които съдържат метаданни за съдържанието му.

Разлики:

- ***HTML е език, а XML – метаезик***. Въпреки че и двата езика произлизат от SGML, на практика HTML е негово специализирано приложение, докато XML е функционално подмножество на SGML. HTML елементите и атрибутите са предефинирани и с ясно определен смисъл. XML от своя страна запазва гъвкавостта и разширяе-мостта на SGML - той не дефинира собствена семантика и набор от тагове, а предоставя синтаксис за описание на други езици.

- ***HTML описва форматиране, а XML – структурирана информация*** - HTML е проектиран с единствената цел да осигури начин за форматиране на документи в World Wide Web. За разлика от него XML предоставя средства за дефиниране на произволни тагове и структурни връзки между тях. HTML описва как да се представи информацията, докато XML описва самата информация, като я структурира по стандартен начин, разбираем за различни приложения.

- ***HTML, XML и добре дефинираните документи***. Въпреки че XML и HTML документите си приличат на външен вид (с тази разлика, че таговете на единия език са предефинирани, а на другия – не), XML синтаксисът е много по-строг и не допуска отклонения за разлика от HTML. В един HTML документ е допустима употребата на некоректно зададени тагове и те се игнорират впоследствие от браузъра, ако той не намери начин как да ги обработи. В XML спецификацията изрично се забранява на приложенията, обработващи XML документи, да гадаят смисъла на синтактично некоректен файл. Ако XML документът не е добре дефиниран, обработката му трябва да се прекрати и да се докладва за грешка.

Какво означава терминът „Добре дефинирани документи“ ?

Някои основни правила, който определят един XML документ като добре дефиниран, са следните:

- документът да има само един основен документен елемент;
- таговете винаги да се затварят и то в правилен ред (да не се застъпват);
- атрибутите винаги да се затварят по правилен начин;
- имената на таговете и атрибутите да отговарят на някои ограничения.

```
<library name=".NET Developer's Library">
```

Документен елемент

Пример за лошо дефиниран XML документ

Използване на специален символ за име на атрибут

Некоректно затворен стринг

```
<xml>
    <button bug! value="OK name="b1">
        <animation source="demo1.avi"> 1 < 2 < 3
    </click-button>
< / xml >
```

Затварящият таг не трябва да има бели полета

Грешен затварящ таг

Кога се използва XML?

Езикът XML има изключително широка употреба в съвременните софту-ерни технологии, защото предоставя универсален формат за съхранение и обмен на информация, а от това имат нужда болшинството от съвре-менните софтуерни системи.

XML се използва при нужда от:

1) *Обмяна на информация* - Обмяната на информация между системи, които боравят с несъвместими формати, е сериозно предизвикателство в съвременното информационно общество. Много системи работят с нестандартизирани, собствени формати и при нужда от взаимодействие разработчиците трябва да полагат много усилия, за да осигурят съвместимост на обменяните данни при комуникацията. XML е едно възможно решение на този проблем, тъй като позволява дефинирането на специфичен за приложението, прозрачен формат за обмяна на информация.

2) *Съхранение на структурирани данни* - Почти всяко приложение има нужда от съхранение на данни. В много случаи XML е подходящ за тази задача, тъй като разделя структурираната информация от нейното визуално представяне. XML е подходящ формат за съхранение най-вече на малки информационни файлове или на данни, които не се очаква да поддържат произволно търсене (достъп). XML markup описва структурата на данните наред с тяхното съдържание. Това позволява да се дефинират схеми за валидация на XML документи, чрез които да се установява валидността на XML структурата.

Недостатъци на XML

- Обемисти данни - ML е текстово-базиран формат, който използва тагове като ограничители (и описатели) на съдържаната в документа информация. Самата му при-рода (текстов формат с чести и повтарящи се етикети) е предпоставка за увеличен размер на файловете (съответно и увеличен мрежов трафик). Големината на един XML файл винаги е по-голяма от тази на файл със същата информация, записана в сравним двоичен формат;
- Повишена необходимост от физическа памет - Един XML документ може да бъде голям по размер по два критерия – в статичния си файлов формат (нужда от повече дисково пространство за съхранение) или в заредената в динамичната памет форма (нужда от повече изчислителни ресурси и RAM памет). Като пряко следствие от това, че XML данните са значителни по обем, идва повишената необходимост от физическа памет за съхраняването им;
- Намалена производителност

I.3. Дефиниране на пространства от имена

Имената на елементите и атрибутите се състоят от две части – име на пространството, на което принадлежат, и локално име. Това съставно име е известно като квалифицирано име (qualified name, Qname). Идентификаторите на пространствата от имена в XML трябва да се придържат към специфичен URI (Uniform Resource Identifier) синтаксис. URI спецификацията дефинира две основни URI форми:

URL (Uniform Resource Locators) например **http://www.evropa.com/town**

URN (Uniform Resource Names) например **urn:america-com:country**.

URI идентификаторите обикновено са доста дълги и вместо тях в XML документите се използва префикс за асоцииране на локалните елементи и атрибути с определено пространство от имена. Префиксът е просто съкратен псевдоним за един URI идентификатор, който се свързва с него при дефинирането на пространство от имена:

```
xmlns:<префикс>="<идентификатор на пространство от имена>"
```

Използване на тагове с еднакви имена – пример:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <country:towns  xmlns:country="urn:evropa-com:country"
3                  xmlns:town="http://www.evropa.com/town">
4      <town:town>
5          <town:name>Sofia</town:name>
6          <town:population>1 200 000</town:population>
7          <country:name>Bulgaria</country:name>
8      </town:town>
9      <town:town>
10         <town:name>Plovdiv</town:name>
11         <town:population>700 000</town:population>
12         <country:name>Bulgaria</country:name>
13     </town:town>
14 </country:towns>
```

Пространства по подразбиране

Използването на префиксно-ориентиран синтаксис е сравнително интуи-тивен процес за повечето софтуерни разработчици. Съществува обаче и друг начин за асоцииране на XML елементите с пространствата от имена – дефинирането на пространства по подразбиране. Използва се следният синтаксис:

```
xmlns="<идентификатор на пространство от имена>"
```

Пространство по подразбиране

Пример:

```
1  <?xml version="1.0" encoding="windows-1251"?>
2  <order xmlns="http://www.hranitelni-stoki.com/orders">
3      <item>
4          <name>бира "Загорка"</name>
5          <amount>8</amount>
6          <measure>бутилка</measure>
7          <price>3.76</price>
8      </item>
9      <item>
10         <name>кебапчета</name>
11         <amount>12</amount>
12         <measure>брой</measure>
13         <price>4.20</price>
14     </item>
15 </order>
```

Елементът **<item> не е** изрично асоцииран с пространство от имена, затова той автоматично се свързва с пространството по подразбиране. **Пълното име** на елемента <item> **е http://www.hranitelni-stoki.com/orders:item.**

I.4. Схеми и валидация

В смисъла на XML, схема е формално описание на формата на XML документи.

Документ, който издържа успешно теста, описан от съответната XML схема, се определя като валиден (съобразяващ се със схемата). Процесът на тестване на документ спрямо зададена схема се нарича валидация.

Схемата гарантира, че документът изпълнява определени изисквания. Тя открива грешки в документа, които в последствие могат да доведат до неправилната му обработка. Схемите лесно се публикуват в Интернет и могат да служат като общодостъпен начин за описание на синтаксиса на дадено XML приложение.

XML схеми – защо са необходими?

Съдържанието на XML документите се контролира чрез дефиниране на схеми. Схемите контролират структурата на XML документите и дефинират необходимия синтаксис за целта.

Схемите описват:

- допустими тагове, които могат да присъстват в един XML документ;
- допустими атрибути за тези тагове;
- допустими стойности за елементите и атрибутите в документа;
- ред на поставянето на таговете в XML документа;
- дефинират стойности по подразбиране.

XML схеми – видове:

- DTD (Document Type Definition);
- XSD (XML Schema Definition language);
- XDR (XML-Data Reduced).

DTD (Document Type Definition)

- DTD съдържа правила за таговете и атрибутите в документа

- DTD контролира структурата на един XML документ, като дефинира множество от разрешени за използване елементи. Други елементи извън описаните не могат да присъстват в документа.

- DTD декларира множество от позволени атрибути за всеки елемент. Декларация на атрибут определя името, типа данни, стойностите по подразбиране (ако има такива) и указва дали атрибутът задължително трябва да присъства в документа или не.

- DTD е текстово-базиран език.

Пример: **library.dtd**

```
<!-- contents of library.dtd -->
<!ELEMENT library (book+)>
<!ATTLIST library name CDATA #REQUIRED>
<!ELEMENT book (title, author, isbn)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
```

Дефиниран е елемент с име **library**, който съдържа една или повече (но поне една) инстанция на елемента book. За елемента library е дефиниран списък от атрибути – library задължително трябва да притежава атрибут с име name от тип CDATA (character data).

Използване на DTD – пример

Документен елемент

Работим с публична външна DTD декларация

Относителен път до DTD декларация

Вмъкване на DTD документа

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <!DOCTYPE library PUBLIC "library.dtd">
3  <library name=".NET Developer's Library">
4      <book>
5          542 Програмиране за .NET Framework
6          <title>Programming Microsoft .NET</title>
7          <author>Jeff Prosise</author>
8          <isbn>0-7356-1376-1</isbn>
9      </book>
10     <book>
11         <title>Microsoft .NET for Programmers</title>
12         <author>Fergal Grimes</author>
13         <isbn>1-930110-19-7</isbn>
14     </book>
15 </library>
```

XSD (XML Schema Definition language)

- по-мощен инструмент за описание на XML документите;

- XML-базиран език за описание на XML структурата;

- предоставя набор от вградени типове данни, които разработчиците могат да използват, за да ограничават съдържанието на текста;

- позволява употребата и на потребителски типове - XSD поддържа дефинирането на **два основни потребителски класа – прости типове** (чрез **таг xs:simpleType**, където xs е префикс за системното пространство от имена http://www.w3.org/2001/XMLSchema) и **комплексни типове** (чрез таг **xs:complexType**)

- простите типове не задават структура, а само стойностно поле, и могат да бъдат задавани само на текстови елементи (без наследници) и атрибути.

- Елементите, притежаващи допълнителна структура – например с дефини-рани атрибути или наследници - трябва да бъдат описани с комплексен тип.

library.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="library">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="book" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:string"
            use="optional" />
    </xs:complexType>
  </xs:element>
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title" />
        <xs:element ref="author" />
        <xs:element ref="isbn" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="title" type="xs:string" />
  <xs:element name="author" type="xs:string" />
  <xs:element name="isbn" type="xs:string" />
</xs:schema>
```

Схемата дефинира пет **глобални елемента – library, book, title, author** и **isbn** – като три от тях са дефинирани от тип string (**xs:string**, където xs отново е префикс за пространството от имена на XMLSchema), а **library** и **book** са дефинирани като комплексни типове. Всеки един от глобалните елементи може да бъде използван като документен елемент в XML файл. Структурата на library определя, че този елемент съдържа неограничен брой елементи book и има незадължителен атрибут name от тип string. Елементът book е съставен от title, author и isbn (точно в тази последователност) и не дефинира атрибути.

Използване на XSD схеми – пример

Гореописаната XSD схема лесно може да се асоциира с даден XML документи и да се използва за неговата валидация. Това става най-лесно с помощта на дефинирания в пространството от имена **http://www.w3.org/2001/XMLSchema-instance** и атрибут **noNamespaceSchemaLocation**, който указва относителния път до XSD документа, съдържащ съответната валидираща схема:

```
<?xml version="1.0" encoding="utf-8" ?>
<library name=".NET Developer's Library"
     xsi:noNamespaceSchemaLocation="library.xsd"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <book>
     <title>Programming Microsoft .NET</title>
     <author>Jeff Prosise</author>
     <isbn>0-7356-1376-1</isbn>
   </book>
   <book>
     <title>Microsoft .NET for Programmers</title>
     <author>Fergal Grimes</author>
     <isbn>1-930110-19-7</isbn>
   </book>
</library>
```

XDR (XML-Data Reduced) library.xdr

```
<?xml version="1.0" encoding="UTF-8"?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data"
     xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="author" model="closed"
    content="textOnly" dt:type="string"/>
  <ElementType name="title" model="closed"
    content="textOnly" dt:type="string"/>
  <ElementType name="isbn" model="closed"
    content="textOnly" dt:type="string"/>
  <ElementType name="book" model="closed"
    content="eltOnly" order="seq">
    <element type="title" minOccurs="1" maxOccurs="1"/>
    <element type="author" minOccurs="1" maxOccurs="1"/>
    <element type="isbn" minOccurs="1" maxOccurs="1"/>
  </ElementType>
  <ElementType name="library" model="closed"
    content="eltOnly" order="seq">
    <AttributeType name="name" dt:type="string"
      required="yes"/>
      <attribute type="name"/>
    <AttributeType name="xmlns" dt:type="string"/>
      <attribute type="xmlns"/>
    <element type="book" minOccurs="1" maxOccurs="*"/>
  </ElementType>
</Schema>
```

Схемата определя, че съдържанието на елементите **author**, **title** и **isbn** може да бъде единствено текст, но не и други елементи

Стойността на атрибута **model (closed)** показва, че тези елементи не могат да съдържат елементи и атрибути, освен изрично споменатите в модела на съдържанието

Елементът **book** от своя страна не може да съдържа свободен текст, а само елементите, описани в неговия модел на съдържанието (**content= "eltOnly"**), като те трябва да спазват точната последователност (**order="seq"**) – точно по един елемент в реда title, author и isbn.

Използване на XDR схеми – пример


```
<?xml version="1.0"?>
<library name=".NET Developer's Library"
      xmlns="x-schema:http://url-of-schema/library.xdr">
   <book>
      <title>Programming Microsoft .NET</title>
      <author>Jeff Prosise</author>
      <isbn>0-7356-1376-1</isbn>
   </book>
   <book>
      <title>Microsoft .NET for Programmers</title>
      <author>Fergal Grimes</author>
      <isbn>1-930110-19-7</isbn>
   </book>
</library>
```

Необходимо е в документния елемент да се съдържа специално форматиран атрибут за включване на пространство от имена

Когато XDR-съвместим парсер срещне пространство от имена, започващо с **x-schema**, той изтегля схемата от зададения URL адрес и извършва необходимата валидация.

I.5. XML и .NET Framework

.NET Framework е проектиран от самото начало с идеята за силно интегрирана XML поддръжка.

Имплементациите на основните XML технологии се съдържат в асемблито System.Xml, където са дефинирани следните главни пространства от имена:

- **System.Xml** – осигурява основните входно-изходни операции с XML (XmlReader и XmlWriter) и други XML помощни класове.
- **System.Xml.Schema** – осигурява поддръжка на валидация на XML съдържание чрез XML Schema (XmlSchemaObject и наследниците му).
- **System.Xml.XPath** – реализира функционалност за XPath търсене на информация и навигация в XML документ (класовете XPathDocument, XPathNavigator и XPathExpression).