Въведение в...

XML

Съдържание

За какво служи XML?

XML е съкращение от eXtensible Markup Language. Това е език, с помощта на който по универсален начин може да се описва дървовидно структурирана информация. XML е независим от платформа, езици за програмиране и операционна система. Тази необвързаност го прави много полезен при нужда от взаимодействие между хетерогенни програмни платформи и/или операционни системи. Без XML, форматът на обменяната информация трябва да бъде договарян при всеки отделен случай, което значително затруднява комуникацията между системите.

Резюме

В тази статия ще ви запознаем със синтаксиса на XML и ще разгледаме примери на прости XML документи. Ще разберем как се създават и използват XML документите, какво е добре дефиниран и валиден XML. Накрая накратко ще разгледаме примери за широката му употреба.

История

Разработката на XML започва през 1996г. Тогава W3C (World Wide Web Consortium) свикват работна група, чиято цел е да създаде език за описание на данни, който да има широко приложение, а документите, писани на този език, да бъдат лесни за създаване и четене. Първата XML спецификация е готова в края на 1997г., а февруари 1998г. официално излиза като W3C препоръка. През 2000г. и 2004г. документът е доусъвършенстван, за да се стигне до W3C XML 1.1 в края на 2004г.

Основата на XML

XML е markup език

Самото съкращение XML подсказва, че XML е markup език. Това означава, че в XML си служим с тагове (подобно на HTML). Днес се използват два основни класа markup езици – специализирани и с общо предназначение. Първата група езици служат за генериране на код, който е специфичен за определено приложение или устройство и адресира точно определена необходимост – такъв език е HTML. Общите markup езици описват структурата и значението на документа, без да налагат условия по какъв начин ще се използва това описание. XML е типичен markup език с общо предназначение.

XML е разширяем

За разлика от HTML, където имената на таговете са определени в спецификацията на езика, в XML предварително зададени тагове няма. Това дава изключително голяма гъвкавост на езика и обяснява до голяма степен думата extensible в името му. Този, който пише XML документът, решава какви тагове да използва, за да опише най-добре данните в него.

Освен това програма, която стандартно "разчита" даден XML документ, трябва безпроблемно да извлича същата информация, дори и ако се добавят нови полета и данни в него. Така XML е разширяем, защото от това не възникват проблеми.

XML определя структура

Важно свойство на XML е способността му да описва дадена структура. Чрез влагане на тагове едни в други се създава дървовидна йерархия, като по този начин могат да се описват сложни структури от данни. Докато HTML визуализира данните и

е концентриран над техния външен вид, XML е създаден да ги описва и е концентриран над тяхната същност.

Какви знания ми трябват?

За да разберете напълно тази статия, не са ви нужни почти никакви предварителни познания. Все пак ще ви е от полза да имате основни познания по HTML, което ще ви помогне да разграничите по-добре специфичните особености на XML.

Създаване на XML документ

Първият ни XML документ

Преди да се запознаем с правилата за създаване на XML документи ще направим едно примерно XML описание на бележка:

Отворете някакъв текстов редактор, например Notepad. Напишете следното:

```
<?xml version="1.0" encoding="windows-1251"?>
<note>
  <to>Ани</to>
  <from>Никола</from>
  <heading>Въпрос</heading>
  <body>Свободна ли си довечера?</body>
</note>
```

Запазете този текст във файл с разширение xml. Така вече създадохте първия си XML документ! Сега можете да го отворите с вашия web браузър и, ако не сте объркали нещо, съдържанието му ще се визуализира на екрана.

XML документите са замислени сами да описват себе си и да са максимално разбираеми. Такъв би трябвало да е и този документ, който описва една бележка от Никола за Ани със заглавие "Въпрос" и текст "Свободна ли си довечера?". За човек, който не разбира XML, този запис би изглеждал странен, но пак има шанс да разбере описанието на бележката.

Синтаксис

Всеки един XML документ трябва да отговаря на синтаксиса на XML. По-долу са изредени и обяснени правилата, които са залегнали в документите на W3C:

Тагове

Таговете в XML почти винаги вървят по двойки – отварящ и затварящ. Една такава двойка определя един елемент от структурата, която XML документът описва.

Примерна двойка тагове са тези:

```
<note></note>
```

• Отварящият таг започва със символа '<'. Веднага след това е името на елемента, което може да съдържа всякакви букви (не символи) в Unicode формат, цифри, знакът точка ('.'), тире ('-') и долна черта ('_'). Името не може да започва с цифра, тире ('-') или точка ('.'). Имена, които започват с "xml" (без значение малки или големи букви), са резервирани за специфични конструкции дефинирани от езика и не могат да се ползват за общи цели. Между символа '<' и името на тага не може да има други символи – например интервали. Отварящият таг завършва със символа '>'.

Правилни отварящи тагове са:

```
<ime>, <име_на_кирилица>, <_.>, <a.tag>, <Я-w.9_СМЕСИЦА>
```

Неправилни отварящи тагове са:

< ime>, <име на кирилица>, <.NET>, <user@server>, <xMLtag> (последният е правилен, но името започва с "xML" и не трябва да се използва)

• Затварящият таг в двойката е същият като отварящия, с единствената разлика, че след символа '<' има наклонена черта ('/'). Името на затварящия таг трябва да е същото като на отварящия. Таговете <note> и </endnote> не са двойка, защото са с различни имена.

Правилни затварящи тагове са:

</ime>, </име_на_кирилица>, </_.>, </a.tag>, </Я-w.9_СМЕСИЦА>

Неправилни затварящи тагове са:

</ ime>, </име на кирилица>, </.NET>, </user@server>

Идеята на XML е да бъде лесен за разбиране и XML документите сами да обясняват съдържанието си. Затова тагове от вида <_.> са силно нежелани, били те и правилни. Имената най-често се състоят само от букви, като ако съдържат няколко думи, за разделител се използва долната черта: <име_на_кирилица>.

Обикновено описанието на един елемент съдържа не само отварящ и затварящ таг, а и нещо друго (текст, други елементи) между таговете. Когато обаче елементът е празен, може да се запише съкратено:

Вместо <atom></atom>

се използва <atom/> или с допълнителен интервал: <atom />

Case sensitive

В XML се прави разлика между малки и големи букви. Таговете <atag> и <aTag> са различни, защото буквата 't' е малка в единия и голяма в другия. По същата причина таговете <note> и </Note> не са двойка, защото имената, които съдържат, са различни.

Влагане

Един елемент в XML може да съдържа други елементи, но не може два елемента да се "припокриват".

Тази последователност на таговете не е правилна:

<tasks> task1 <current_tasks> task2 </tasks> task3 </current_tasks>

Елементите трябва да бъдат затваряни в обратен ред на тяхното отваряне.

Например:

This text is bold and italic

Root елемент

XML документите трябва да имат един елемент, който съдържа всички останали. Този елемент се нарича Root елемент.

Ето документ с две бележки:

```
<?xml version="1.0" encoding="windows-1251"?>
<conversation>
  <note>
    <to>Ани</to>
    <from>Никола</from>
    <heading>Въпрос</heading>
    <body>Свободна ли си довечера? </body>
  </note>
  <note>
    <to>Никола</to>
    <from>Ани</from>
    <heading>Отговор</heading>
    <body>Ако разбираш от XML, да! </body>
  </note>
</conversation>
```

Тук Root елементът е с име "**conversation**". Ако двата тага `<conversation>` и `</conversation>` се премахнат, документът вече няма да отговаря на изискванията.

Атрибути

В отварящия таг, освен име на елемента, който той определя, могат да се дефинират и т.нар. атрибути. Всеки атрибут си има име и стойност и носи някаква съпътстваща информация за елемента. Атрибутите се изреждат след името на елемента, отделени с празно място. Форматът им е *име=стойност*, като името трябва да отговаря на същите изисквания, както и името на елемента, а стойността трябва задължително да е заградена в единични ('...') или двойни ("...") кавички. Няма значение дали използваме единични или двойни кавички, но тази свобода е полезна, ако вътре в стойността трябва нещо да е също заградено в кавички – тогава цялата стойност се огражда в едните, а вътре използваме другите. Ето един пример за елемент с атрибут:

```
<file filename="data.txt">
  <size>100</size>
  <type>text</type>
  ...
</file>
```

В случая този, който ще чете документа, ще се интересува от размера, вида на файловете и т.н., а името е само някаква служебна информация, която помага да бъдат обработени. Съвсем валиден е и този запис:

```
<file>
  <name>data.txt</name>
  <size>100</size>
  <type>text</type>
  ...
</file>
```

Дали дадена информация ще е под формата на атрибут или елемент зависи от този, който създава документа. Все пак е препоръчително да избягваме атрибутите, защото са по-трудни за обработка, не могат да съдържат съставни данни и информацията, която носят, не може лесно да се обогатява (разширява). Следното нещо със сигурност не трябва да се случва:

```
<file name="data.txt" size="100" type="text"></file>
```

Атрибутите все пак са полезни, когато трябва да предоставим допълнителна информация на програмата, която ще обработва XML документа:

```
<note id='1'>
  <to>Ани</to>
  ...
</note>
<note id='2'>
  <to>Никола</to>
  ...
</note>
```

Пространства от имена

Тъй като в XML значението на таговете не е предварително дефинирано, понякога може да се получи неяснота в какъв контекст да се разбира името на даден таг. Например, таг с име table може да обозначава таблица в смисъла на HTML, таблица в база от данни или пък не таблица, а маса. Това може да е проблем (или поне води до неудобства), ако два елемента с еднакво име, но различен смисъл, трябва да се използват в един XML документ или в два, които ще се използват заедно. Разрешението е в използването на пространства от имена (namespaces). Чрез тях за всеки елемент се декларира в какъв контекст да се разглежда той. Така вече може да се разграничи table в смисъла на HTML от table в смисъла на мебели за дома:

```
<?xml version="1.0"?>
<furniture xmlns="furniture"
           xmlns:code="http://www.w3.org/1999/xhtml"
           xmlns:html="http://www.w3.org/1999/xhtml">
```

```
        <html:table>
          <html:tr>
            <code:td>
              <table>
                <pirce>100$</pirce>
                <diameter unit="meters">1.5</diameter>
              </table>
            </code:td>
          </html:tr>
        </html:table>
      </furniture>
```

В този пример са дефинирани две пространства от имена - "furniture" и "http://www.w3.org/1999/xhtml". Няма ограничение за името, но доста често то е Интернет адрес, на който има някакво описание на пространството. Употребата на адрес е подходящо, защото това дава известна гаранция, че няма да има друго пространство със същото име, а целта на пространствата от имена е еднозначно да определят елементите, които са свързани с тях.

Дефинирането на пространство от имена става в отварящ таг, като това пространство е валидно за всички елементи вложени в дадения. Дефинирането става чрез дефиниране на атрибута "xmlns" (от XML NameSpace) или на атрибута "xmlns:[префикс]", като стойността на атрибута е името на пространството от имена, а [префикс] определя с какъв префикс да се свърже даденото пространство. В примера два различни префикса са свързани с едно и също пространство. Даден елемент се свързва с определено пространство, когато името му се разшири, като в началото му се постави префикс и двоеточие (например: `<html:table></html:table>`). В случая елементите "html:tr" и "code:td" са от едно и също пространство от имена, защото и двата префикса "сочат" едно и също място. Ако някой елемент не е свързан чрез префикс, то той се причислява към най-близкото обкръжаващо пространство, което е дефинирано без префикс, ако има такова. В примера елементите "table", "price" и "diameter" са от пространството "furniture".

Стриктно и пълно описание на това, какво е namespace и как се прилага в XML документите, можете да намерите на сайта на w3c.

Добре дефинирани документи

Един документ е добре дефиниран, ако спазва гореописаните правила. Програмите, които четат XML документи, разчитат, че тези документи са добре дефинирани. В противен случай, при първото несъответствие със стандарта, четенето на документа приключва и не може да бъде довършено. Тези програми са изключително малки в сравнение с тези, които разпознават HTML, защото при HTML се допускат неточности (например липсващи затварящи тагове) и браузърите трябва да могат да се справят с всяка една от тях. За да се нарече даден документ "XML документ", той задължително трябва да е добре дефиниран.

Текст и коментари

Нека си представим, че искаме да напишем следния елемент с код:

```
      <code>
```

```
  int matchwo(a,b){
     if (a < b && a < 0){
        return 1;
     } else {
        return 0;
     }
  }
</code>
```

Програмите, които четат XML документи третират целия текст като XML код. Това води до някои неудобства. При срещане на символа '<' в if конструкцията, той ще бъде интерпретиран като начало на таг и това ще доведе до грешка. За да се избегнат такива ситуации може да се подходи по различни начини:

Escape символи

Вместо '<' можем да използваме escape последователността '<'. Тази последователност от символи ще се интерпретира като знакът за по-малко (и не като начало на таг). Символът '&' обозначава началото на такава последователност, като освен '<' има още четири: '>' за '>', ''' за апостроф, '"' за знака за кавички и '&' за '&'. Символите '<' и '&' задължително трябва да се заместват с техните escape последователности, докато за другите три това може и да не се прави. За правилно интерпретиране на текста, третият ред трябва да изглежда така:

```
     if (a &lt; b &amp;&amp; a &lt; 0){
```

CDATA таг

Можем да заградим текст, който не трябва да се интерпретира като XML код в специален CDATA таг. Кодът ще изглежда така:

```
<code>
 <![CDATA[int matchwo(a,b){
  if (a < b && a < 0){
    return 1;
  } else {
    return 0;
  }
 }]]>
</code>
```

Този таг е със специален синтаксис и няма нужда от затварящ таг. След срещането на последователността '<!CDATA[' всички следващи символи се интерпретират като обикновен текст, а не като XML, до първото срещане на ']]>'. Следователно CDATA тагът може да съдържа всякаква последователност от символи, с изключение на ']]>'.

Коментари

Още един начин дадена част от документ да не се интерпретира като XML е слагането на коментар:

```
<!-- <tag not parsed> </no errors> -->
```

Коментарът започва с '<!--' и след това всичко се пропуска до срещането на '-->'. Следователно коментарите могат да съдържат всякаква последователност от символи, с изключение на '-->'. За разлика от escape символите и CDATA-данните, коментарът не се счита за част от XML документа. Коментарите обикновено съдържат информация полезна за човека, който ще чете и евентуално ще променя документа:

```
<collection>
  <!-- All the books should be added here -->
</collection>
```

Processing instructions

В XML документите могат да се пишат специални тагове, които съдържат информация (инструкции) за правилното интерпретиране на данните. Тези тагове се наричат Processing Instructions (PI). Те, както и коментарите, не се считат за част от XML текста. За разлика от коментарите, PI съдържат информация, полезна за програмата, която ще чете и интерпретира данните. Форматът им е следният - <?target PIdata?> - на мястото на target стои името на програмата, за която е предназначена информацията, а PIdata е самата информация:

```
<collection>
  <?databaseEngine primary key = [attribute, id]?>
  <?libraryCatalog identification:attribute=id?>
  <book id="1">...</book>
  <book id="2">...</book>
  ...
</collection>
```

Двете инструкции в примера носят една и съща информация – че атрибутът id еднозначно определя дадена книга, но едната е предназначена за някаква програма, която ще работи с база от данни, а другата – за библиотечния каталог. Всяка програма търси и "разбира" само тези инструкции, които са създадени точно за нея.

Една специална PI е "xml декларация":

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

Тази декларация е специален случай на PI и трябва да присъства в началото на всеки XML документ. В нея атрибутът version задължително трябва да присъства. След това може да се дефинират и атрибутите encoding (определя символната таблица на документа) и standalone (може да е със стойност "yes" или "no" и определя дали документът зависи от някакви други документи). encoding и standalone могат да се пропускат.

Използване на XML документ

XML парсъри

XML парсъри се наричат програмите, които могат да проверяват дали даден текст е добре дефиниран XML документ и могат да извличат информацията в него. Съществуват множество XML парсъри, не малка част от които са с отворен код. XML парсър можете да си напишете и сами като част от някое приложение, което работи с XML, но по-лесно е да си намерите готова реализация на съответния език. Все пак, ако сами си реализирате XML парсър, това ще означава, че много добре разбирате спецификацията и занапред няма да имате проблеми с нея.

Най-широко разпространените XML парсъри са Интернет браузърите. Всеки съвременен браузър може да отваря XML документи и да ги визуализира според структурата, която те определят.

Валидни документи

В практиката обикновено за един XML документ трябва да се знае нещо повече от това, че е добре дефиниран. Например, ако една компютърна система очаква да обработва данни, свързани с online поръчка на книги, няма да знае какво да прави с документ, който съдържа данни за здравни осигуровки. Нещо повече – ако очаква елементи с имена catalog и book, а намери Catalog и Book, това не прави документа по-малко разбираем за човека, но го прави съвсем друг от гледна точка на тази система. Затова са създадени документи, които дефинират вида на информацията в даден XML документ – конкретната му структура, възможните атрибути, техните стойности и т.н. Тук накратко ще спомена двата най-известни вида документи Document Type Definition (DTD) и XML Schema Definition (XSD).

Валидирането на XML документи улеснява четенето им не само от хора, а и от програми. Това е нещото, което прави XML изключително полезен в разработването на съвременен софтуер. Преди да стигнем до приложенията на XML, ще дадем два примера за документи за валидация:

DTD

Нека разгледаме следния XML:

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to      (#CDATA)>
  <!ELEMENT from    (#CDATA)>
  <!ELEMENT heading (#CDATA)>
  <!ELEMENT body    (#CDATA)>
]>
<note>
  <to>Ани</to>
  <from>Никола</from>
  <heading>Въпрос</heading>
  <body>Свободна ли си довечера? </body>
</note>
```

Тагът DOCTYPE казва, че в документа ще има root елемент note, той може да съдържа елементите to, from, heading и body в този ред (последните съдържат текст – Character DATA). По подобен начин могат да се опишат и по-сложни документи.

Декларацията на съдържанието на документа (таговете <!ELEMENT>) може да се запише и в отделен текстов файл. Тогава, ако файлът е например definition.dtd, в XML документа трябва да присъства следното:

```
<!DOCTYPE note SYSTEM "definition.dtd">
```

XSD

Нека сега разгледаме този XML:

```
<?xml version="1.0" encoding="windows-1251"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string" />
        <xs:element name="body" type="xs:string" />
        <xs:element name="from" type="xs:string" />
        <xs:element name="heading" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Този XML е всъщност дефиниция на XML-а от предишния пример. XSD служи за същото, както и DTD, но е на основата на XML и затова предлага по-големи възможности за бъдещо развитие и подобрения.

За да се валидира документът според тази XSD схема, тагът <note> трябва да изглежда така:

```
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="schema.xsd">
```

Приложения на XML

XSD

Едно от най-важните приложения на XML документите е да описват XML документи. Голямата гъвкавост и свобода, която дава XML, прави XML документите трудни за описание по някакъв друг начин. Ако не съществуваше нещо като XSD, всеки XML щеше да е сам за себе си и евентуално разбираем само от човека. XSD прави XML наистина полезен от софтуерна гледна точка. XSD е нещото, на което най-много може да се разчита при валидирането на XML, защото е също XML и адекватно ще отговаря на всички промени в спецификацията на езика.

XSL

XML описва структурата на данните, а не как те изглеждат. Затова възниква нужда от средство, което да описва външния вид на информацията, точно както например CSS описват как да се визуализира един HTML документ. Това средство е XSL (XML Stylesheet Language). Това е език, който се състои всъщност от два независими езика – XSL Transformations (XSLT) и XSL Formatting Objects (XSL-FO). XSLT служи за преобразуване от един XML в друг документ с произволна структура (например HTML). По този начин данните могат да се преструктурират така, че да са по-удобни за визуализация. XSL-FO се грижи за визуализацията, като с него могат да се описват страници, полета, списъци, обекти, шрифтове и др. По този начин един XML може да се преобразува в PDF например.

SOAP

XML е полезен с универсалността си и с това, че чрез него различни компютърни системи могат да споделят данни. Simple Object Access Protocol (SOAP) е XML базиран протокол, който предоставя достъп до отдалечени обекти (и техните методи), Web services и сървъри. Чрез SOAP е възможно вашето приложение да извиква метод, който да се изпълни на отдалечен сървър, и да получи резултата, като информацията (във вид на XML) се предава по HTTP или друг протокол. Предимството е, че това са отворени и свободни за използване технологии. Простотата и липсата на ограничения правят SOAP предпочитан и полезен. С него все по-широка е и употребата на Web services. Така, чрез SOAP, XML осигурява по-лесен обмен на данни и споделяне на ресурси и ни спасява от нуждата да преоткриваме колелото, само защото няма общодостъпен интерфейс, по който да го използваме.

Други приложения

Тук само ще изредим някои от приложенията на XML, за да обърнем още един път внимание на неговата универсалност и широка приложимост. На основата на XML са например Chemical Markup Language – език за описание на химически съединения, Mathematical Markup Language (MathML) – за описание и вграждане на математически символи и изрази в HTML, RSS – използва се при публикуване на данни, които ежедневно се допълват (статии и новини, софтуерни ъпдейти, новооткрити проблеми със сигурността, решения на правителството или съда, обяви за публични събития и т.н.), в класическа литература – пиесите на Шекспир вече са преписани в XML формат, за да могат, например, актьорите по-лесно да извличат нужната им информация, Synchronized Multimedia Integration Language (SMIL) – език за мултимедийни презентации в Web пространството, Open Software Description (OSD) – формат за описание на софтуерни ъпдейти, Scalable Vector Graphics (SVG) – за описание на векторни изображения, MusicXML – за описание на ноти, петолиния, тактове, паузи и т.н., Open Financial Exchange – за обмен на финансови отчети.

Това са примери как XML може да се използва за наистина всякаква информация и не рядко това се оказва най-приемливият вариант за реална работа с данните.

Бъдещето на XML

XML е все по-често и широко използван. И това не е заради някоя голяма компания, която го "бута" напред, а заради това, че е прост и в същото време много гъвкав и полезен. XML е така замислен, че да търпи бъдещи промени и развитие и лесно да се адаптира към нуждите на тези, които го използват. Тези негови качества, както и широката му употреба в днешно време, обещават дълъг живот на XML. Това не е убягнало на разработчиците и като резултат XML е все по-интегриран и лесен за употреба. Затова, например, разполагаме с готови класове за работа с XML на различни езици за програмиране, съществуват множество готови програми за работа с XML документи, а в базите от данни има нов тип освен стандартните int, float и т.н. – xml.

Полезни връзки

http://www.w3schools.com/w3c/w3c_xml.asp - тук можете да намерите пълната XML спецификация.

http://www.w3schools.com/xml/ - това е сайт, на който достъпно и с много примери е описан синтаксиса на XML, както и някои негови прости употреби.