آشنایی با XML

منبع

Learning programming Visual Basic.NET

McGraw-Hill

مترجم

امیر مرادآبادی

مقدمه

زبان علامت گذاری قابل توسعه (eXtensible Markup Language) **یا همان** XML **زبانی است برای نمایش داده ها به فرمی که بر هیچ تکنولوژی اختصاصی خاصی مستند نیست.** این زبان خنثی (neutral) توسط کنسرسیوم شبکه ارتباطی جهانی (www.w3c.org) استاندارد شد و به سرعت استانداردی برای تبادل اطلاعات در اینترنت شد. نه تنها XML یک تکنولوژی خنثی است بلکه بر پایه قالب بندی کاراکتری (شبیه HTML برای انسان خواناست) است ، همچنین با پروتکل های شبکه (TCP/IP) که مدیریت انتقال اطلاعات در اینترنت را بر عهده دارد سازگار است.

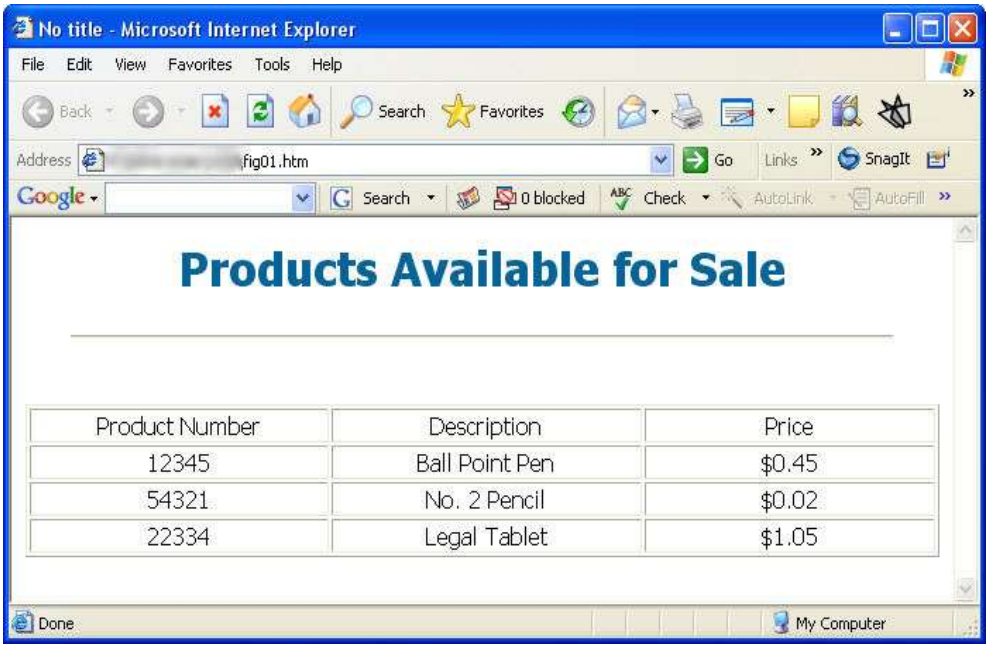
به خاطر XML یکسری از تکنولوژی های وابسته به آن نمو پیدا کردند که کار با آن را راحت تر کردند. یکی از این ها زبان شمای XML یا XML schema می باشد. این زبان روشی را برای تعریف این که چگونه یک مستند XML باید تشکیل شود فراهم می آورد. از آنجائی که XML می تواند به راحتی برای برنامه خاصی تغییر یابد ، **شمای XML روشی را برای تعریف و معتبر سازی مستندات XML به طوری که مطمئن شویم از قوانین شما پیروی کرده اند فراهم می آورد.** شرکای تجاری می توانند بر روی یک شمای XML مشخص برای مستندات XML ای که می خواهند مبادله کنند به توافق برسند. در این روش آنها مطمئن هستند که مستنداتی که انتقال می دهند تمام اطلاعاتی که انتظار دارند را با قالب بندی خاصی که خودشان طراحی کرده اند می بینند.

تکنولوژی کاربردی دیگری که وابسته به XML است XSL (eXtensible Stylesheet Language) و XSLT (eXtensible Stylesheet Language Transforms) می باشد.

XSLT روشی برای تبادل یک مستند XML به مستند دیگری را فراهم می آورد. این قابلیت کاربردی است زیرا می توان یک مستند XML ساخت و آن را به نسخه های مختلف (از جمله HTML) که برای کارهای خاصی درست شده اند انتقال داد.

XML چیست؟

از وقتی که اینترنت مشهورتر و تجارت با آن گسترش یافت و تجارت های " سازمان به مشتر ی" (B2C) و "تجاری – تجاری" (B2B) رواج یافت کمبود و کاستی های زبان HTML بیشتر واضح شد. HTML زبانی است که بر روی نمایش اطلاعات برای مصرف انسان ها متمرکز است. به عبارت دیگر برای تبدیل کردن اطلاعات به فرمی که برای انسان قابل فهم باشد طراحی شده است.



تقریبا مفهوم و معنی این صفحه به وضوح روشن است. ما اطلاعاتی را برای سه محصول که شامل product number, Description, price می باشد می بینیم. اگر بخواهید قیمت محصولی با شماره 54321 را پیدا کنید مشکلی برای این کار نخواهید داشت. در حقیقت این کار برای شما آنقدر راحت است که آن را بدون اینکه فکر کنید انجام می دهید. و این به خاطر این است که شما باهوش هستید و می توانید اطلاعات تصویری را به راحتی پردازش کنید. اما ماشین آن طوری که

شما این صفحه را پردازش کردید نمی بیند. بلکه آنها را به فرم اصلی شان که همان HTML است می بیند. آن HTML شبیه چیست؟ شکل زیر کد HTML ای است که به وسیله جستجوگر اینترنت در شکل قبلی نمایش یافت.

قسمتی از کد HTML شکل قبل

```
<html>
<head>
<title>No title</title>
</head>
<body bgcolor="white" text="black" link="blue" vlink="purple" alink="red">
...
<table border="1">
  <tr>
    <td width="299">
      <p align="center"><font face="tahoma">Product Number</font></p>
    </td>
    <td width="299">
      <p align="center"><font face="tahoma">Description</font></p>
    </td>
    <td width="299">
      <p align="center"><font face="tahoma">Price</font></p>
    </td>
  </tr>
  <tr>
    <td width="299">
      <p align="center"><font face="tahoma">12345</font></p>
    </td>
    <td width="299">
      <p align="center"><font face="tahoma">Ball Point Pen</font></p>
    </td>
    <td width="299">
      <p align="center"><font face="tahoma">$0.45</font></p>
    </td>
  </tr>
...
</table>
</body>
</html>
```

ممکن است شما HTML را بلد نباشید اما نماد <tr> که به عنوان یک تگ شناخته می شود شروع یک سطر از جدول و تگ </tr> پایان یک سطر از جدول را نمایش می دهد. همچنین درون یک سطر از جدول می توان چندین ستون را با تگ های <td> و </td> ایجاد نمود. شکل زیر تعریف یک سطر از جدول را نمایش می دهد.

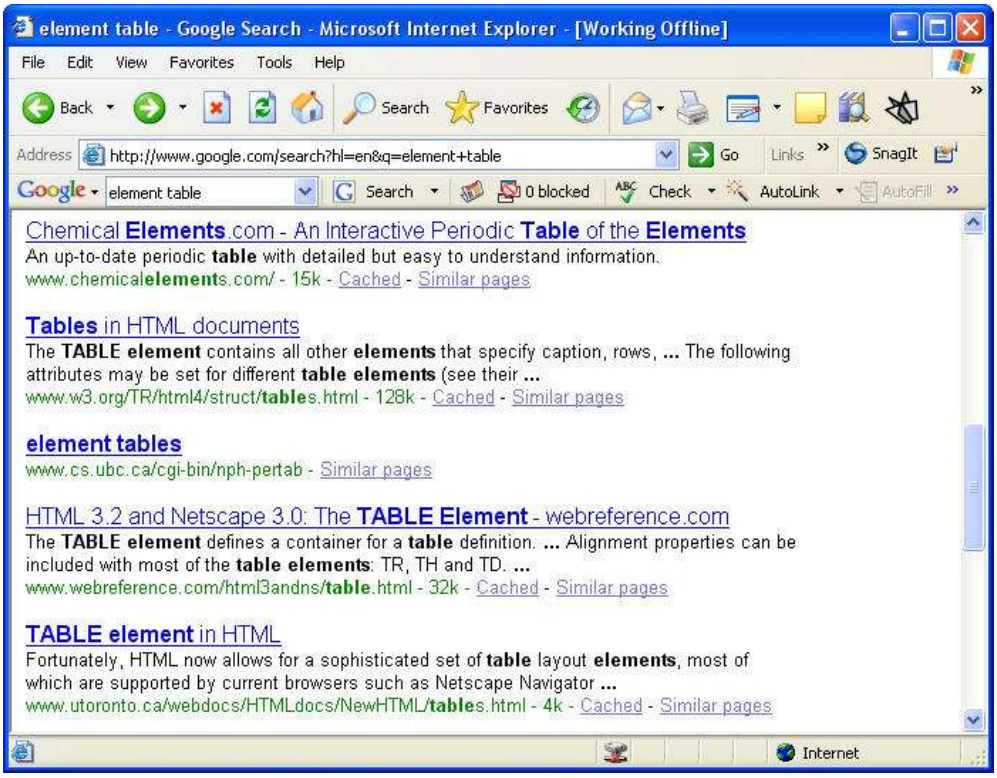
```
  <td width="299">
    <p align="center"><font face="tahoma">12345</font></p>
  </td>
  <td width="299">
    <p align="center"><font face="tahoma">Ball Point Pen</font></p>
  </td>
  <td width="299">
    <p align="center"><font face="tahoma">$0.45</font></p>
  </td>
</tr>
```

به خط سوم شکل که تعریف یک ستون از جدول می باشد بیشتر دقت کنید:

```
<p align=”center”>
    <font face=”tahoma”>Product number</font>
</p>
```

این به چه معنی است؟ تگ ابتدای یک پاراگ و انتهای آن را نشان می دهد. درون تگ p خاصیتی به نام align دیدیم که مقداری برابر center داشت. و به این است که محتویات پاراگراف در مرکز قرار خواهند گرفت. تگ <font> برای تعیین نوع فونت متن (اینجا Tahoma) به کار می رود. سرانجام بین تگ های پاراگراف و بین تگهای فونت اعداد 12345 را می بینیم. این محتوایی است که با فونت Tahoma در مرکز یک پاراگراف قرار می گیرد. همان طور که می بینید HTML برای نمایش اطلاعات به کار می رود.

اکنون یک سوال می پرسیم ، مندرجات 12345 به چه معنی است؟ در HTML ما می دانیم شبیه چیست ولی نمی توانیم معنی آن را بفهمیم. با رجوع به شکل اول چون آن زیر ستون شماره کالا قرار دارد می فهمیم که آن چیست ولی از HTML سخت است که به چنین نتیجه ای برسیم. این بزرگترین نقص HTML است. یک روش عالی برای نمایش داده هاست اما درک مفهوم مندرجات آن بسیار ضعیف است. این نقص چه طور به برنامه های شما ضربه وارد می سازد؟ پیش از هر کار دیگر برنامه هایی شبیه موتور های جستجو ممکن است اطلاعات بی فایده ای را به شما بدهند زیرا آنها فقط مندرجات را بدون توجه به معنای کاربردی آنها نشان می دهند. فرض کنید می خواهید به دنبال جدول عناصر شیمی بگردید. Element table را جستجو می کنید. شکل زیر چنین جستجویی را نشان می دهد.



می توانید برخی موفقیت ها و برخی خطا ها را ببینید زیرا جدول عناصر معانی مختلف زیادی دارد.

در چنین مواقعی دوباره این انسان است که باید بفهمد کدام گزینه برای مناسب تر است. اکثرتان

قادر به انجام چنین کاری هستید ولی این کار برای ماشین سخت است.

اگر HTML شکل گفته شده را با اطلاعات نشان داده شده در شکل زیر جایگزین کنیم چه می

شود؟

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet type="text/xsl" href="prodDesc.xslt" ?>
<ProductList>
    <Product>
        <ProductNumber>12345</ProductNumber>
        <Description>Ball Point Pen</Description>
        <Price>0.45</Price>
    </Product>
    <Product>
        <ProductNumber>54321</ProductNumber>
        <Description>No. 2 Pencil</Description>
        <Price>0.02</Price>
    </Product>
    <Product>
        <ProductNumber>22334</ProductNumber>
        <Description>Legal Tablet</Description>
        <Price>1.05</Price>
    </Product>
</ProductList>
```

توجه داشته باشید که تگ ها با ترمینولوژی ای است که مستقیما به آن داده ای که ذخیره می شود مربوط است. اگر پرسیده شود که "مندرجات 'legal tablet' به چه معناست؟" به راحتی پاسخ می دهیم که توصیف یک کالا می باشد (چون در تگ description که درون تگ product می باشد قرار دارد). در شکل بالا نمایش لیست محصولات را با استفاده از زبان علامت گذاری قابل توسعه (XML) می بینید. تگ های <product> و <price> از کجا آمده اند. بله نویسنده آنها را نوشته است. و این معنی extensible در این زبان می باشد. هر کسی می تواند این زبان را به هر اندازه که می خواهد با رعایت کردن موارد خاصی توسعه دهد.

اگرچه ممکن است بگوئید این روش نمایش به خوبی نمایش HTML نیست. بله درست می گوئید. اما باید بدانید که ما با XML مضمون داده ها را از نمایش آنها جدا می کنیم. البته که با چندین روش مختلف می توانیم XML را نمایش دهیم. شکل زیر دو گونه نمایش XML را نشان می دهد. هر دو نمایش شکل زیر از یک فایل XML واحد استفاده کرده اند. و این قدرت XML است که می تواند همان اطلاعات را به طرزی که می خواهیم نشان دهیم.

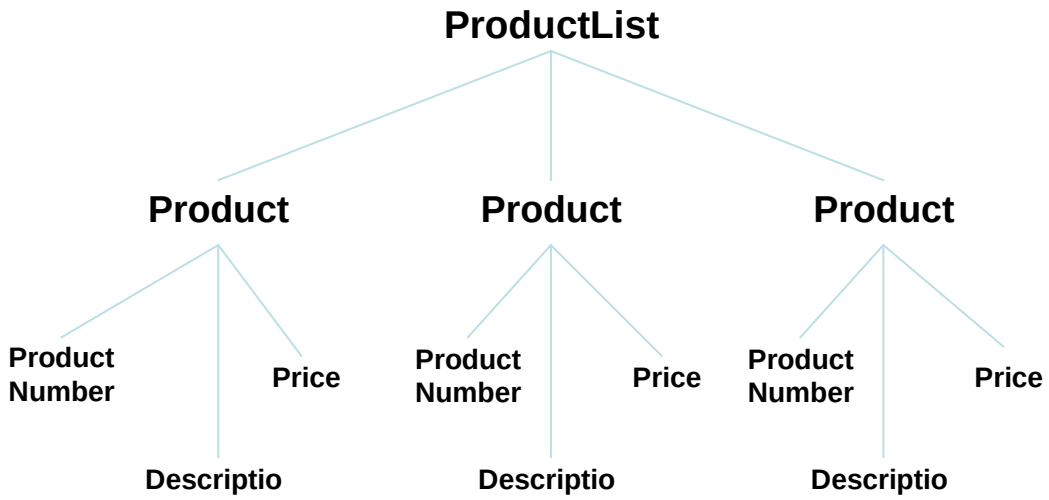
برای مثال یک طرز نمایش ممکن است شبیه HTML برای نمایش های خانگی (desktop browsers) شبیه شکل بالا باشد و شکل دیگر برای WML (Wireless Markup Language) که دستگاه های بی سیم از آن اسفاده می کنند باشد و شکل سوم ممکن است

XML اصلی برای یک کاتالوگ الکترونیکی یک فروشنده باشد. هیچ محدودیتی برای نمایش یک فایل XML وجود ندارد. یک مثال مهم استفاده XML در صنعت روزنامه نگاری است. بسیاری از روزنامه نگار های بزرگ مقاله را هم به صورت پرینت شده هم به صورت نسخه وب دارند. با ذخیره اخبار در فایل های XML آنها می توانند به هر منظوری از آن استفاده کنند.

XML به سرعت استانداردی برای تبادل اطلاعات در اینترنت شد. انتقال داده ها بین کامپیوتر ها ممکن است. زیرا کامپیوتر ها می توانند طوری برنامه ریزی شوند که تگ های خاصی را پیدا کنند و از محتویات آنها استفاده کنند.

استاندارد هایی شبیه ebXML (electronic business XML) برای فعالیت های business to business استفاده می شود.

داده های XML را می توان با درخت نمایش داد. برای مثال XML ای که پیشتر دیدیم را می توان با درخت نشان داد.



در شکل بالا "productList" گره ریشه است. Product شاخه و productNumber Description و price گره های برگ هستند.

قوانین تعریف XML

- ✓ زبان حساس به بزرگی و کوچکی حروف یا case sensitive می باشد. بنابراین <price> و <Price> با هم برابر نیستند.
- ✓ فقط و فقط دارای یک گره ریشه می باشد.
- ✓ تمامی عناصر باید دارای یک تگ شروع و پایان باشند. به این معنی که اگر مثلا تگ شروع <price> دارید حتما باید تگ متناظری مثل </price> داشته باشید. یک روش برای نمایش برگ هائی که محتویات خالی دارند وجود دارد. برای مثال اگر عنصر <price> هیچ مقداری نداشت باید آن را به شکل <price/> نشان دهید. و این به خوبی جایگزین تگ های باز و بسته می شود. بنابراین <price></price> و <price/> به یک معنی هستند.
- ✓ تگ ها باید به درستی تو در تو باشند. به عبارت دیگر یک تگ ممکن است درون تگ دیگری باشد اما تگ های شروع و پایان باید درون تگ های شروع و پایان محاصره کننده باشند. مثال زیر قانونی است:

```
<product>
        <price> 1.25 </price>
</product>
```

در این مثال price به صورت تو در تو داخل product قرار دارد. یک مثال غیر قانونی در شکل زیر آمده است:

```
<product>
        <price> 1.25 </product >
</price >
```

این قانونی نیست زیرا تگ های شروع و پایانی عنصر price هر دو داخل تگ های شروع و پایانی عنصر product قرار ندارند.

- ✓ اگر یک عنصر یک خاصیت داشت ، باید درون quotation marks قرار گیرد. خواص مقادیری هستند که که به یک گره وابسته (مربوط) هستند. برای مثال:

```
<product hazardous=”true”>
 <name> nitroglycerin </name>
</product>
```

در این مثال عنصر product خاصیتی با نام hazardous (پر خطر) دارد. هر عنصر می تواند دارای صفر یا n خاصیت باشد. بنابراین مقادیر صفات بایستی درون دو ' یا دو " قرار گیرند.

- ✓ عناصر و خواص باید با یک حرف یا underline شروع شوند و می توانند به هر تعداد حرف ، عدد ، خط فاصله ، نقطه و underline که می خواهید داشته باشند. به هر حال اسامی را با معنی و کوتاه انتخاب کنید (همان طور که با متغییر رفتار می کردید)

اگر از این قوانین پیروی کنید مستند XML شما را خوش ترکیب یا well formed گویند. به این معنی که در مستند ما هیچ خطای نحوی (دستوری) وجود ندارد. یک راه دیگر برای بررسی مستندات XML وجود دارد که معتبر یا valid نام دارد. این را بعدا بحث خواهیم کرد ولی فعلا قبول کنید که یک مستند می تواند خوش ترکیب باشد ولی معتبر نباشد.

فضا های نام (Namespaces)

اگر یک مستند XML را از یک فروشنده لوازم خانگی (furniture supplier) که تگی به نام <table> دارد و همچنین مستند XML دیگری را از یک مشاور مالیاتی (tax advisor) که آن نیز تگی با نام <table> دارد دریافت کنید چه می شود؟

در مستند اول منظور "میز" می باشد در حالی که در مستند دومی به یک جدولی از نتایج مالیاتی اشاره می کند. چه طور می خواهید این مشکل را حل کنید؟

بله ممکن است بتوانید با نگاه به این دو آنها را تشخیص دهید ولی این کار درست نیست.

XML راه حلی برای این وضعیت ارائه می دهد. این راه حل استفاده از فضاهای نام می باشد (www.w3.org/TR/REC-xml-names/)

یک فضای نام به راحتی به یک مرجع اشاره می کند. برای مثال در فضای نام فروشنده لوازم خانگی می دانیم <table> به چه معنی است. بنابراین از این فضای نام در مستنداتمان استفاده می کنیم. به راحتی نام عنصر را با اضافه کردن یک پیشوند و یک colon (:) تغییر می دهیم. بنابراین بر اساس آنچه گفته شد نام عناصر را به <furn:table> و <tax:table> تغییر می دهیم. این را بدانید که اگر این تگ ها در یک مستند نیایند ممکن است مشکلی برای تفکیک آنها نداشته باشید چون خود مستندات گواهی از محتویاتشان می دهند.

فضا های نام کار دیگری را برای تفکیک تگ ها انجام می دهند. این کار به پارسر (parser) این اجازه را می دهد که مفهوم تگی را بفهمد اگرچه آن مبهم باشد. (پارسر برنامه ایست که XML را برای تعیین محتویاتش پردازش می کند.)

برای مثال ممکن است چنین چیزی را ببینید:

<xsl:sort select=”ProductNumber”>

در اینجا فضای نام " xsl" به معنی eXtensible Stylesheet Language می باشد. بنابراین هنگامی که پارسر این فضای نام را می بیند می داند که باید تابع sort را که در XSL تعریف شده انجام دهد.

برای تعریف یک فضای نام باید یک xmlns یا همان XML NameSpace را در هر عنصری به عنوان یک خاصیت تعریف کنید. تمامی فرزندان درون آن عنصر می توانند از آن استفاده کنند. اگر

می خواهید فضای نام در تمام مستند XML تان استفاده شود می توانید xmlns را به عنوان یک خاصیت در گره ریشه تعریف کنید. این تعریف به صورت زیر است:

Xmlns:name = “uri”

که name را شما تعریف می کنید. uri یا Uniform Resource Identifier شناسه واحدی است که اغلب (ولی نه لزوما) یک url یا Uniform Resource Locator می باشد. درباره uri بعدا بیشتر صحبت می کنیم.

```
1  <?xml version="1.0" ?>
2  <ipo:purchaseOrder xmlns:ipo="http://www.altova.com/IPO">
3  ...
4    <Items>
5      <item partNum="833-AA">
6        <productName> Lapis necklace </productName>
7        <quantity> 1 </quantity>
8        <price> 99.95 </price>
9        <ipo:comment> Want this for holidays </ipo:comment>
10       <shipDate> 1999-12-05 </shipDate>
11     </item>
12   </Items>
13 </ipo:purchaseOrder>
```

شکل12

شکل بالا مثالی را که یک فضای نام تعریف می شود و در ادامه به عنوان بخشی از تعریف یک گره به کار می رود را نشان می دهد. در این مثال گره ریشه <purchaseOrder> می باشد و یک خاصیت (xmlns:ipo="http://www.altova.com/IPO ") به تعریف این گره اضافه شده است. ipo می تواند هر نامی باشد. توجه داشته باشید که ipo به هر دوی تگ های شروع و پایان اضافه شده است. ipo:purchaseOrder و ipo:comment

uri در این مثال (که در واقع یک url است) این آدرس http://www.altova.com/IPO است. اگر با یک مرورگر اینترنت به این آدرس رجوع کنید یک پیغام خطا دریافت خواهید کرد چون آنجا هیچ فایل HTML ای وجود ندارد. تنها هدف این uri این است که آن یکتاست. از آنجائی که شرکت Altova صاحب آدرس www.altova.com می باشد تمام کنترل آن دست این شرکت

می باشد و هیچ شرکت دیگری نمی تواند از آن استفاده کند. بدین گونه برای این که خیالمان راحت باشد ، از آنجائی که uri باید یکتا باشد از url استفاده می کنیم.

گاهی اوقات uri بر پایه url بنا نمی شود. ولی باید در یک مستند XML حتما این uri یکتا باشد.

Document Prolog

document prolog نشان می دهد که یک مستند XML است. همین طور نشان دهنده نوع مستند ، تعاریف موجودیت و دیگر دستور العمل های پردازشی می باشد. در اینجا به اعلان XML می پردازیم.

خط اول در هر مستند XML نشان می دهد که این یک فایل XML است و نسخه XML ای را که از آن استفاده می کنید نشان می دهد.

<?xml version=”1.0”?>

خواص دیگری نیز وجود دارند که می توانید کنار version ار آنها استفاده کنید. اینها شامل مستند چه طور کد گذاری شده (character set) و نام فایل های دیگری که این مستند برای بارگذاری درست لازم دارد می باشد. به تگ های مخصوص شروع و پایان (<? و ?>) این موجودیت توجه کنید. این تگ ها برای تعریف یک دستورالعمل پردازشی (processing instruction) می باشند. PI اطلاعاتی است که پارسر XML از آنها استفاده می کند و از داده های واقعی جدا هستند.

شِمای XML

فرض کنید دو شریک تجاری تصمیم می گیرند اطلاعاتشان را با مستندات XML مبادله کنند. چطور می توانند بفهمند مستندی که دریافت کرده اند از قوانین پیروی کرده است؟

تصور کنید هیچ قانونی وجود نداشت ، هر کدام می توانستند تگی را برای خود تعریف کنند که ممکن بود دیگری حتی معنی آن را هم نفهمد. همچنین می توانستند از دو تگ مختلف برای یک داده یکسان استفاده کنند. مثلا ممکن است یکی از <po> و دیگری از <purchOrder> برای purchase order استفاده کنند. بنابراین می بینید که ارتباطات چقدر سخت می شد اگر یک قانون مدون برای مستندات معتبر وجود نداشت.

XML دو روش برای تعریف قوانین برای مستندات XML دارد:

- ✓ تعریف نوع مستند یا DTD
- ✓ شمای XML یا www.w3c.org/XML/Schema

DTD ابزار اصلی برای این هدف بود. اما در حال جایگزین شدن با نوع جدید و بهتر که همان Schema می باشد هست. بنابراین از DTD می گذریم و بر روی schema که استاندارد Microsoft در visual basic.NET می باشد متمرکز می شویم.

به XML شکل زیر که از روی شکل های گفته شده نوشته شده است دقت کنید.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet type="text/xsl" href="prodDesc.xslt" ?>
<ProductList>
    <Product>
        <ProductNumber>12345</ProductNumber>
        <Description>Ball Point Pen</Description>
        <Price>0.45</Price>
    </Product>
    <Product>
        <ProductNumber>54321</ProductNumber>
        <Description>No. 2 Pencil</Description>
        <Price>0.02</Price>
    </Product>
    <Product>
        <ProductNumber>22334</ProductNumber>
        <Description>Legal Tablet</Description>
        <Price>1.05</Price>
    </Product>
</ProductList>
```

شکل زیر یک شمای XML که مستند شکل بالا را پشتیبانی می کند نشان می دهد. برای درک بیشترتان از شما درباره آن بیشتر صحبت می کنیم.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="ProductList" targetNamespace="http://tempuri.org/ProdList.xsd"
    xmlns:mstns="http://tempuri.org/ProdList.xsd"
    xmlns="http://tempuri.org/ProdList.xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
    attributeFormDefault="qualified" elementFormDefault="qualified">
  <xs:element name="ProductList" msdata:IsDataSet="true"
    msdata:EnforceConstraints="False">
    <xs:complexType>
     <xs:choice maxOccurs="unbounded">
      <xs:element name="Product">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="ProductNumber" type="xs:string" minOccurs="0" />
         <xs:element name="Description" type="xs:string" minOccurs="0" />
         <xs:element name="Price" type="xs:float" minOccurs="0" />
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

در ابتدا توجه کنید که شما در حقیقت یک مستند XML است. (این را می توانید از خط اول مستند دریابید.) این به این معنی است که شما از پیشقوانین دستوری نوشتن Schema را می دانید. خطوط دو تا شش عنصر ریشه هستند که شما را تعریف می کنند. <xs:schema ... >

همچنین در تعریف گره ریشه این جمله خواص id ، targetNameSpace ، attributeFormDefault و elementFormDefault را نیز تعریف می کند.

سرانجام چهار فضای نام را که شامل xs می باشد در ادامه تعریف می کند. شما همچنین عنصری با نام ProductList از نوع complex (پیچیده ، مجموعه) تعریف می کند. نوع complex نوعی است که شامل عناصر اضافی می باشد. عنصر ProductList از تعداد بی پایانی از عناصر که از لیستی از عناصر محصولات انتخاب شده ساخته شده است. این سه عنصر پایانی انواع رشته ، رشته و اعشار را ذخیره می کنند.

یک بار که یک شمای XML برای مستند XML ساخته شد مستند باید در برابر شما معتبرسازی شود. به عبارت دیگر مستند XML با قوانین تعریف شده در شما مقایسه می شود و چنانچه یک قانون نقض شده باشد مستند XML نامعتبر نامیده می شود. توجه کنید که همچنان می تواند خوش ترکیب باشد به این معنی که از قوانین تعریف XML تجاوز نمی کند اما همچنان نا معتبر باشد. خوش ترکیب بودن به قوانین نحوی بر می گردد در حالی که معتبر بودن به مستند خوش ترکیبی بر می گردد که از بعضی از مجموعه قوانین اضافی کنترل کننده این که عناصر چه طور در مستند XML مرتب شوند پیروی کند.

سرانجام ، تعدادی از گروه های استاندارد ساز در صنعت در حال ساخت شما های XML ای هستند که برای صنایع مختلف قابل استفاده باشند. برای مثال ebXML یا تجارت الکترونیکی XML مجموعه ای از شما ها را برای کارهای (transaction) مختلف و مستندات مشترک دیگری در تجارت الکترونیک ارائه می دهد. یک مثال دیگر XBRL یا Extensible Business Reporting Language (زبان گزارش گیری تجاری قابل توسعه) می باشد.

Styling XML

در شکل های قبل ما دو تعبیر مختلف از یک مستند یکسان را دیدیم. این به کمک XSLT یا همان Extensible Stylesheet Language امکان پذیر است.

(www.w3.org/style/XSL)

XSLT وسیله ای برای تبدیل یک مستند XML به مستند دیگر می باشد. شکل 9.12 چگونگی کار این تبدیل را نشان می دهد.

همان طور که میدانید ، یک مستند XML و دستور العمل های XSLT برای ایجاد یک مستند جدید بوسیله یک پردازشگر XSLT پردازش می شود. در مثال ما مستند XML جدید هم یک مستند HTML ایست. توجه کنید که HTML تا وقتی که از قوانین گفته شده پیروی کند XML است. این یک پردازش تبدیل مشترکی است که می توانیم یک مستند XML را به هر مستند معتبری تبدیل کنیم. مستندات دیگری که XML معتبر هستند شامل زبان علامت گذاری بی سیم (WML) و تگ های اساسی HTML که از دستگاه های بی سیم پشتیبانی می کنند می باشند. پردازشگر های XSLT در محصولات نرم افزاری گوناگونی وجود دارند. برای مثال Internet Explorer یک پردازشگر XSLT دارد که می تواند مستندات XML را برای نمایش به HTML تبدیل کند. vb.NET شامل متد ها و کلاس هایی است که همچنین توانائی پردازش XML را با استفاده از یک پردازشگر XSLT دارند. همچنین تعدادی از پردازشگر های XSLT به صورت open source و مجانی وجود دارند. بنابراین XSLT وسیله ایست برای گرفتن مستند XML اصلی (به یاد داشته باشید که ساخار درختی دارد) و انتخاب تمام درخت یا زیر شاخه های آن و قالب بندی و مرتب سازی گره ها برای تبدیل به یک مستند جدید.

در شکل های قبلی مستند اصلی XML و نمایش تبدیل شده آن به HTML با استفاده از XSLT را در I.e. نشان می دهد. توجه کنید که در خط دوم مستند XML دستورات زیر را می بینید:

<?xml-stylesheet type=”text/xsl” href=”prodDesc.xslt” ?>

این جمله یک رهنمود می باشد که به مرورگر می گوید stylesheet تعریف شده برای مستند در چه فایلی است.

شکل های زیر محتویات مستند XSLT فایل prodDesc.xslt را نشان می دهند. اجازه دهید نگاهی به XSLT بیاندازیم.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

<!-- Template #1 -->
<xsl:template match="ProductList">
 <html>
  <body>
    <div style="font-family:Tahoma,Arial,sans-serif;
          font-size:20pt; color:red;
          text-align:center; letter-spacing:8px;
          font-weight:bold">
       Another List of Products
    </div>
    <hr />
    <table width="100%" cellpadding="5" border="0"
          style="font-family:Tahoma,Arial,sans-serif;
          font-size:16pt; color:black;
          text-align:center">
```

```
        <tr>
          <td>Description</td>
          <td>Price</td>
          <td>Product Number</td>
        </tr>
      <xsl:for-each select="Product">
      <xsl:sort select="Description"/>
        <tr>
          <xsl:apply-templates select="Description" />
          <xsl:apply-templates select="Price" />
          <xsl:apply-templates select="ProductNumber" />
        </tr>
      </xsl:for-each>
    </table>
    <div style="font-family:Arial,sans-serif;
          font-size:8pt; margin-left:10px">
    </div>
  </body>
 </html>
</xsl:template>
```

```
<!-- Template #2 -->
<xsl:template match="Description">
 <td style="font-family:Comic Sans MS, Arial,sans-serif;
        color:darkblue; font-size:16pt;
        font-weight:bold">
  <b><xsl:value-of select="."/></b>
 </td>
</xsl:template>

<!-- Template #3 -->
<xsl:template match="ProductNumber">
 <td style="font-family:Tahoma,Arial,sans-serif;
        font-size:10pt">
  <xsl:value-of select="."/>
 </td>
</xsl:template>

<!-- Template #4 -->
<xsl:template match="Price">
 <td style="font-family:Tahoma,Arial,sans-serif;
        font-size:10pt">
  $<xsl:value-of select="."/>
 </td>
</xsl:template>
```

ابتدا با نگاه به prolog مستند (اولین خط) می بینیم که درست شبیه شما XSLT نیز یک مستند XML می باشد. خط دوم عنصر stylesheet را تعریف می کند. و این شامل تعریف فضای نام مناسب (xsl) و تعریف نسخه آن می باشد. خط سوم عنصر خروجی است که پردازشگر XSLT برای تعیین نوع خروجی استفاده می کند. در اینجا مقدار آن نشان می دهد که خروجی XML است.

خط ششم <xsl:template match=”productList” > شروع تعریف یک قالب (template) می باشد. template ها روش های اصلی برای اینکه تبدیلات چه طور انجام شوند می باشند. ممکن است قالب هایی برای طراحی استفاده کرده باشید. template های طراحی شکل های ساده

ای را تعریف کرده اند که شما می توانید با استفاده از مداد یا خودکار آنها را بر روی کاغذ به راحتی طراحی کنید. قالب های XSLT نیز به این صورت عمل می کنند.

برای درک XSLT لازم است ساختار درختی مستند XML اصلی را مرور کنیم.

حال به XSLT توجه کنید. چهار template می بینید (xsl:template) هر کدام از این قالب ها یک خاصیت به نام match دارند. قالب اول چون در ریشه قرار دارد template ایست که مربوط به تمام درخت است. هر سه قالب دیگر به یک node خاص مربوط اند. قالبی که در ریشه قرار دارد قالب master نامیده می شود. همان طور که می بینید چند خط ابتدایی در قالب اول حاوی HTML است. این HTML قسمتی از درخت جدید می باشد. (به یاد داشته باشید که HTML ای که از قوانین XML ما تبعیت کند نیز XML است)

قالب اول XSLT را نگاه کنید

خط اول آن (xsl:for-each) بسیار شبیه حلقه تکرار For each … Next در ویژوال بیسیک می باشد. در اینجا در حال ساخت حلقه ای می باشیم که برای هر محصول در مستند XML تکرار شود. مستند ما سه محصول دارد (شماره کالاهای 12345 ، 54321 و 22334) بنابراین حلقه سه بار تکرار می شود. جمله بعد (xsl:sort) گره های درخت جدید را با توجه به فیلد Description مرتب می کند. در ادامه ایجاد یک سطر را با HTML می بینید. (<tr> … </tr>) این به این معنی است که سه دستور xsl:apply-templates محتویات را درون این سطر از جدول قرار می دهند. سه جمله xsl:apply-template هر کدام با مقادیر مختلف برای خاصیت select شان به دنبال قالب های متناظر برای ایجاد آنها می گردند. اولین جمله xsl:apply-template گره Description را انتخاب می کند. این قالب در شکل زیر نشان داده شده است. در آنجا می بینیم که تعریف یک ستون از جدول HTML ایجاد می شود (<td> … </td>)

```
<!-- Template #2 -->
<xsl:template match="Description">
 <td style="font-family:Comic Sans MS, Arial,sans-serif;
        color:darkblue; font-size:16pt;
        font-weight:bold">
  <b><xsl:value-of select="."/></b>
 </td>
</xsl:template>
```

در جمله <xsl: value-of select=”.” /> خاصیت select به معنی انتخاب محتویات گره جاری می باشد. از آنجایی که گره جاری گره Description می باشد محتویات می تواند هر چیزی مثل “Ball Point Pen” باشد.

ترتیب xsl:apply-templates در حلقه ترتیب ستون ها درون سطر های جدولی که پر می شود را مشخص می کند. (اول Description سپس price و سرانجام ProductNumber)

البته درختان XML می توانند بسیار پیچیده تر از مثال های ما باشند. همچنین XSLT قابلیت های اضافی دیگری دارند. ما شما را به تحقیق روی این قضیه پیشنهاد می کنیم ولی تمامی نکاتی را که برای تبدیل یک مستند به مستند دیگری وجود دارد واضح شد.

پایان