Pengantar XML

Moh Junaedi
mjunaedi@neptunesolution.net

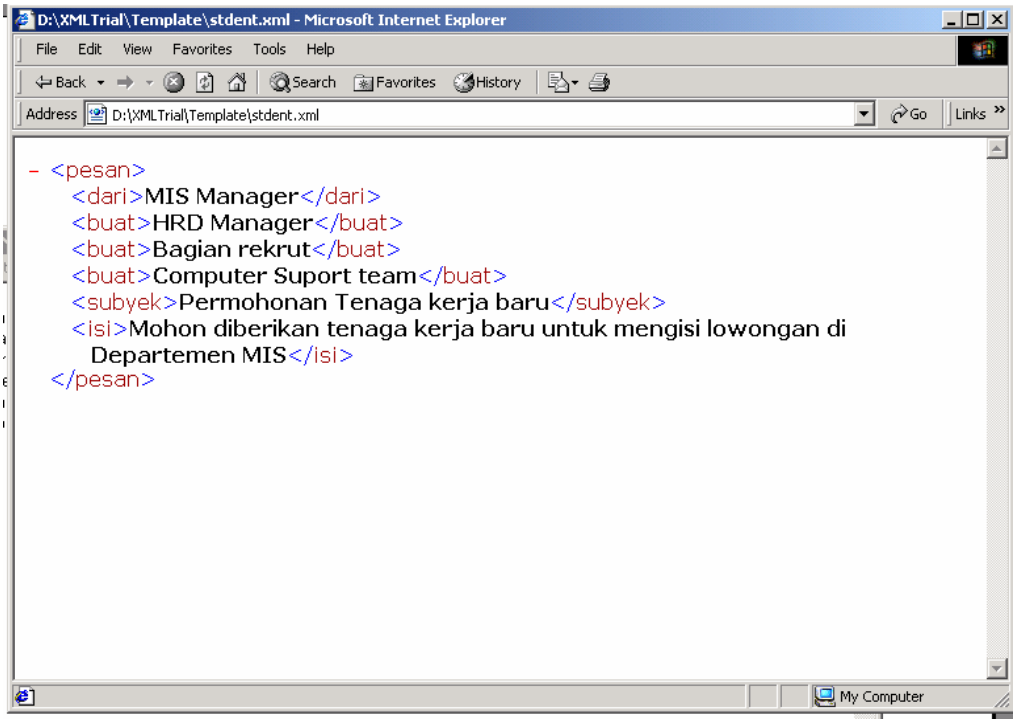
Apakah XML Itu ?

XML kependekan dari *eXtensible Markup Language*, dikembangkan mulai tahun 1996 dan mendapatkan pengakuan dari W3C pada bulan Februari 1998. Teknologi yang digunakan pada XML sebenarnya bukan teknologi baru, tapi merupakan turunan dari SGML yang telah dikembangkan pada awal 80-an dan telah banyak digunakan pada dokumentasi teknis proyek-proyek berskala besar. Ketika HTML dikembangkan pada tahun 1990, para penggagas XML mengadopsi bagian paling penting pada SGML dan dengan berpedoman pada pengembangan HTML menghasilkan markup language yang tidak kalah hebatnya dengan SGML.

Seperti halnya HTML, XML juga menggunakan *elemen* yang ditandai dengan tag pembuka (diawali dengan '<' dan diakhiri dengan '>'), tag penutup(diawali dengan '</ 'diakhiri '>') dan atribut elemen(parameter yang dinyatakan dalam tag pembuka misal <form name="isidata">). Hanya bedanya, HTML medefinisikan dari awal tag dan atribut yang dipakai didalamnya, sedangkan pada XML kita bisa menggunakan tag dan atribut sesuai kehendak kita. Untuk lebih jelasnya lihat contah dibawah:

```
<pesan>
<dari>MIS Manager</dari>
<buat>HRD Manager</buat>
<buat>Bagian rekrut</buat>
<buat>Computer Suport team</buat>
<subyek>Permohonan Tenaga kerja baru</subyek>
<isi>Mohon diberikan tenaga kerja baru untuk mengisi lowongan di
Departemen MIS</isi>
</pesan>
```

pada contoh diatas `<pesan>, <dari> <buat>,` dan `<isi>` bukanlah tag standard yang telah di tetapkan dalam XML. Tag-tag itu kita buat sendiri sesuai keinginan kita. Sampai di sini XML tidak melakukan apapun. Yang ada hanyalah informasi yang di kemas dengan tag-tag XML. Kita harus membuat software lagi untuk untuk mengirim, menerima atau menampilkan informasi di dalamnya.



Gambar 1.1 Tampilan Dokumen XML pada Browser

Kenapa Harus Menggunakan XML?

XML untuk saat ini bukan merupakan pengganti HTML. Masing-masing dikembangkan untuk tujuan yang berbeda. Kalau HTML digunakan untuk menampilkan informasi dan berfokus pada bagaimana informasi terlihat, XML mendeskripsikan susunan informasi dan berfokus pada informasi itu sendiri. XML terutama dibutuhkan untuk menyusun dan menyajikan informasi dengan format yang tidak mengandung format standard layaknya heading, paragraph, table dan lain sebagainya. Sebagai contoh apa bila kita ingin menyimpan dan menyajikan informasi notasi musik pada lagu "Indonesia raya", kita bisa menyimpannya dengan xml seperti contoh dibawah ini.

```
<lagu judul="Indonesia raya" nadadasar="G" Birama="4/4">
<bar nomor="1">
<nada not="B" ketukan="1/2"/>
<nada not="C" ketukan="1/2"/>
<nada not="D" ketukan="1"/>
<nada not="B" ketukan="2"/>
...
</bar>
<bar nomor="2">
...
</bar>
</lagu>
```

Kemudian dengan bantuan software lain misalnya MIDI generator kita bisa mendengarkan musiknya atau kita juga bisa membuat software sendiri yang menampilkan informasi ini dalam bentuk not balok.

Sama dengan HTML, File XML berbentuk teks sehingga bila diperlukan kita bisa membacanya tanpa memerlukan bantuan software khusus. Hal ini memudahkan pengembang aplikasi yang menggunakan XML untuk mendebug programnya. XML lebih fleksible dibanding HTML dalam hal kemampuannya menyimpan informasi dan data. Pada XML kita bisa menyimpan data baik dalam atribut maupun sebagai isi elemen yang diletakkan diantara tag pembuka dan tag penutup.

Kelebihan lain yang dimiliki XML adalah bahwa informasi bisa di pertukarkan dari satu system ke system lain yang berbeda platform. Misalnya dari Windows ke Unix, atau dari PC ke Machintosh bahkan dari internet ke handphone dengan teknologi WAP.

Apakah XSLT Itu?

XSLT adalah kependekan dari *eXtensible StyleSheet Language:Transformation*, adalah bagian dari XSL yang dikembangkan sebelumnya. XSL adalah Stylesheet yang khusus dikembangkan sebagai komplemen XML, untuk merubah informasi pada XML ke dalam bentuk lain agar bisa ditampilkan di layar, dicetak di kertas atau didengarkan telinga. Pada dasarnya proses ini di bagi menjadi dua bagian proses yakni pertama Transformasi Struktural yang meliputi pengumpulan, pengelompokan dan pengurutan data maupun penyusunan ulang, penambahan dan penghapusan tag dan atribut, dan yang kedua adalah proses merubah format menjadi pixel dilayar, nohtah tinta di kertas atau nada di speaker. Proses yang pertama itulah yang kemudian disebut XSLT, sedangkan yang kedua biasa disebut XSLFO (*eXtensible Stylesheet Language:Formatting Object*).
Hasil Keluaran XSLT bisa berupa HTML, Text file atau XML dengan format yang baru. Sebenarnya untuk menampilkan dokumen XML agar lebih menarik dilihat di browser bisa dilakukan oleh Cascade StyleSheet. CSS yang sering digunakan untuk memformat HTML bisa juga dipakai untuk XML. Akan tetapi CSS tidak mampu melakukan tugas tugas yang rumit seperti memformat angka desimal, menjumlah, menghitung rata-rata, menampilkan gambar, dan lain-lain. Dan untuk melakukan tugas-tugas itulah kita memerlukan XSLT

XSLT Processor

XSLT Processor atau yang biasa disebut *Parser* adalah software bantu yang tugasnya menerapkan perintah-perintah dalam XSLT pada dokumen sumber XML, dan menghasilkan dokumen keluaran baik berupa HTML, Text file ataupun XML.

XSLT Processor yang digunakan pada pembahasan-pembahasan dan contoh-contoh dalam buku ini adalah MSXML3 buatan Micosoft. Bila browser kita adalah Internet Explorer versi 5.5 ke bawah, secara default menggunakan MSXML atau MSXML2 sebagai Processor. Sebagian besar contoh dalam buku ini tidak bekerja dengan MSXML dan MSXML2. Untuk mendapatkan MSXML3 anda bisa download secara gratis dari websitenya Microsoft. Anda perlu men-download dua paket, yaitu MSXML itu sendiri beserta SDK-nya dan program XMLinst.exe. Yang disebutkan terakhir adalah utility untuk mengganti default XSL Processor pada internet Explorer dengan MSXML3. Setelah anda mendownloadnya, lakukan hal dibawah ini.

- Buka Command prompt
- Ketik `xmlinst /u` lalu tekan enter – perintah ini digunakan untuk melepaskan default XSL Processor yang digunakan sebelumnya.

- Ketik `regsvr32 msxml3.dll` lalu tekan enter – Perintah untuk meregister file dll ¥ Ketik `xmlinst` lalu tekan enter – menggunakan MSXML versi terbaru sebagai default XSL Processor pada Internet Explorer

Untuk lebih jelasnya anda bisa membacanya pada dokumentasi MSXML3 yang disertakan saat anda mendownloadnya.

Versi terakhir dari MSXML yang dikeluarkan Microsoft saat buku ini ditulis adalah versi 4. Pada MSXML4 ini, XSL Processor tidak terinstall secara default pada Internet Explorer menggantikan versi sebelumnya, tetapi bekerja paralel dengan MSXML versi sebelumnya. Diperlukan script khusus pada file HTML untuk memanggilnya. Bila anda hendak menggunakannya sebagai XSLT Processor di komputer anda, baca baik-baik Microsoft MSXML4 Parser SDK yang disertakan pada saat anda mendownloadnya.

Selain dari Microsoft, anda juga bisa menggunakan parser dari tempat lain misalnya *Saxon* yang dikembangkan oleh Michael Kay dapat anda peroleh secara gratis dari site http://users.iclway.co.uk/mhkay/saxon/instant.html. Satu lagi parser yang bisa digunakan adalah *xt* yang bisa anda peroleh di http://www.jclark.com/xml/xt.html. Cara mempergunakan kedua parser tersebut bisa anda baca di website masing-masing. Tidak menutup kemungkinan anda juga mendapatkan parser lain yang banyak bertebaran di internet. Sekali lagi, baca baik-baik manual user yang disertakan setiap parser yang akan anda gunakan.

Contoh Sederhana

Untuk memahami bagaimana XML dan XSLT bekerja, perhatikan contoh berikut ini:

```
<?xml version=”1.0” encoding=”iso-8859-1”?>
<berita>Saya sedang belajar XML</berita>
```

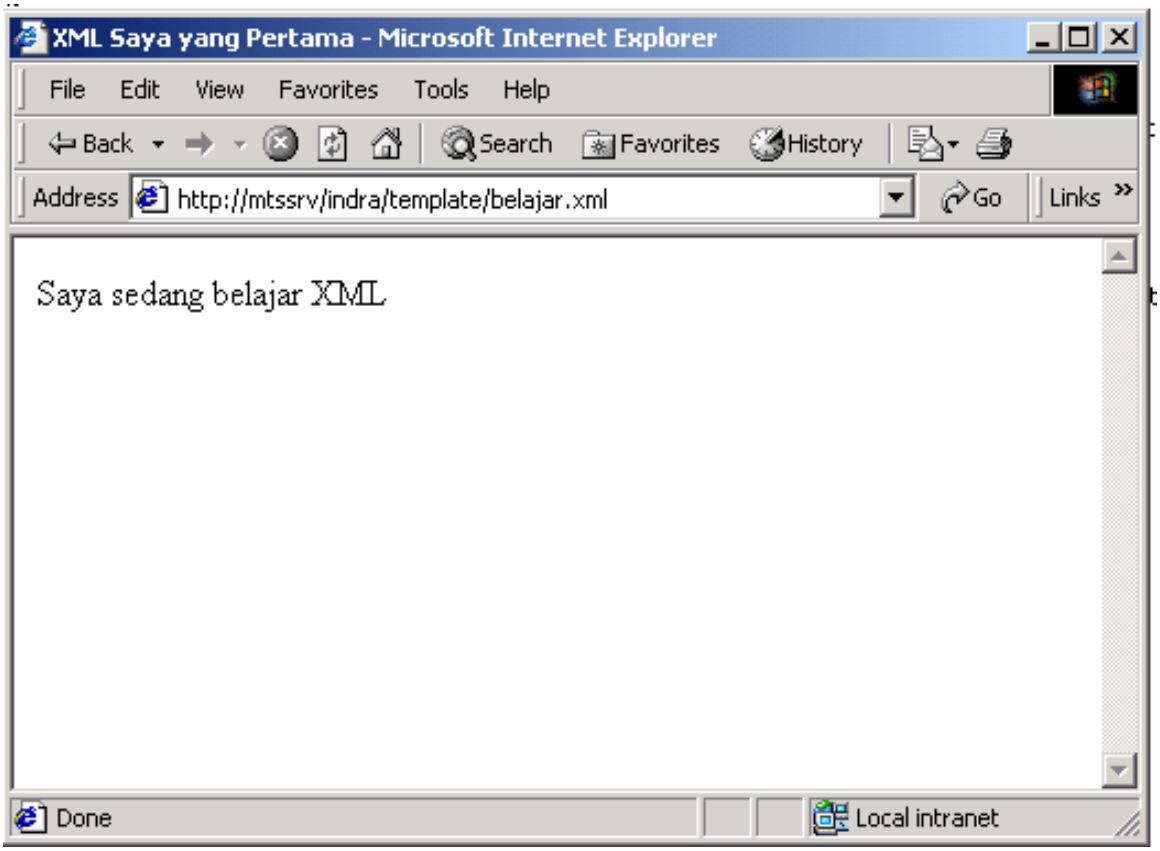
Simpanlah document diatas dengan nama belajar.xml
XSLT yang diperlukan adalah sebagai berikut

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match ='/'>
<html>
<head>
  <title>XML Saya yang Pertama</title>
</head>
<body>
  <p><xsl:value-of select="berita"/></p>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

simpanlah dokumen XSLT ini dengan nama belajar.xsl. Bila anda menggunakan MSXML3 sebagai parser, anda harus menambahkan satu baris heading standard sehingga dokumen XML Anda akan menjadi seperti berikut:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="belajar.xsl"
<berita>Saya sedang belajar XML</berita>
```

Setelah itu anda bisa langsung menjalankannya di browser anda. Anda akan melihat tulisan "Saya sedang belajar XML" di jendela browser anda dan tulisan "XML Saya yang Pertama" sebagai title halaman web anda.



Gambar 1.2 Tampilan Dokumen XML yang sudah diformat dengan XSLT

Sedangkan untuk parser lain anda perlu jalankan parser tersebut sesuai petunjuk menjalankannya.

Hasil yang anda dapatkan sama dengan apabila anda menulis HTML seperti dibawah:

```
<html>
<head>
<title>XML Saya yang Pertama</title>
</head>
<body><p>Saya sedang belajar XML</p></body>
</html>
```

Bila anda tidak melihat apapun di jendela browser anda, berarti Internet Explorer yang anda gunakan belum menggunakan MSXML3 sebagai default parser. Install MSXML3 seperti yang saya jelaskan sebelumnya.

Marilah kita bahas baris perbaris syntax XSLT diatas.

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

Baris pertama adalah heading standard pada XML. Versi yang digunakan dan satu-satunya versi yang ada saat buku ini ditulis adalah versi 1.0. encoding iso-8859-1 adalah nama resmi dari character encoding yang biasa disebut ANSI. iso-8859-1 banyak digunakan di Eropa dan Amerika. Bila anda menggunakan text editor yang mendukung penggunaan character encoding yang lainnya, anda bisa menggunakannya juga. Anda akan mendapatkan pembahasan yang lebih detil pada bab berikutnya.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
```

syntax diatas adalah heading standard untuk XLST. Atribut xmlns:xsl maksudnya adalah mendeklarasikan namespace XML bahwa awalan xsl akan digunakan untuk elemen yang didefinisikan pada W3C XSLT Specification. Tentang Namespace ini akan dibahas pada bab lain.
Selanjutnya,

```
<xsl:template match ='/'>
```

tag diatas memberitahukan aturan yang dipicu ketika bagian tertentu dari dokumen sumber diproses. Dalam hal ini atribut match='/' menunjukkan aturan dipicu dari *root* dokumen.

```
<html>
<head>
  <title>XML Saya yang Pertama</title>
</head>
<body>
  <p><xsl:value-of select="berita"/></p>
</body>
</html>
```

Ini adalah aturan yang dimaksud. Yaitu bagaimana output akan dihasilkan. Dan selanjutnya dua baris terakhir adalah tag penutup untuk dua tag pembuka pertama. Perlu diingat baris pertama bukan tag pembuka tetapi adalah heading standard XML, jadi tidak perlu dibuat tag penutupnya.

Bagian-Bagian dari Dokumen XML

Sebuah dokumen XML terdiri dari bagian bagian yang disebut dengan node. Node-node itu adalah:

- **Root node** yaitu node yang melingkupi keseluruhan dokumen. Dalam satu dokumen XML hanya ada satu root node. Node-node yang lainnya berada di dalam root node.
- **Element node** yaitu bagian dari dokumen XML yang ditandai dengan tag pembuka dan tag penutup, atau bisa juga sebuah tag tunggal elemen kosong seperti `<anggota nama="budi"/>` . Root node biasa juga disebut root element
- **Attribute note** termasuk nama dan nilai atribut ditulis pada tag awal sebuah elemen atau pada tag tunggal.

- **Text node**, adalah text yang merupakan isi dari sebuah elemen, ditulis diantara tag pembuka dan tag penutup
- **Comment node** adalah baris yang tidak dieksekusi oleh parser
- **Processing Instruction node**, adalah perintah pengolahan dalam dokumen XML. Node ini ditandai awali dengan karakter <? Dan diakhiri dengan ?>. Tapi perlu diingat bahwa header standard XML `<?xml version="1.0" encoding="iso-8859-1"?>` bukanlah processing instruction node. Header standard bukanlah bagian dari hirarki pohon dokumen XML.
- **NameSpace Node**, node ini mewakili deklarasi namespace

Sintaks XML

Dibandingkan dengan HTML, XML lebih *cerewet.* Kalau kita menulis sebuah dokumen HTML, beberapa kesalahan penulisan masih ditolerir. Misalnya kita menempatkan tag bersilangan seperti `<p><b>Huruf Tebal</p></b>` meskipun tidak dianjurkan, HTML masih bisa bekerja dan

menampilkan hasil seperti yang kita inginkan. Tidak demikian dengan XML. Lebih jelasnya kita akan bahas di bawah bagaimana membuat dokumen XML yang baik.

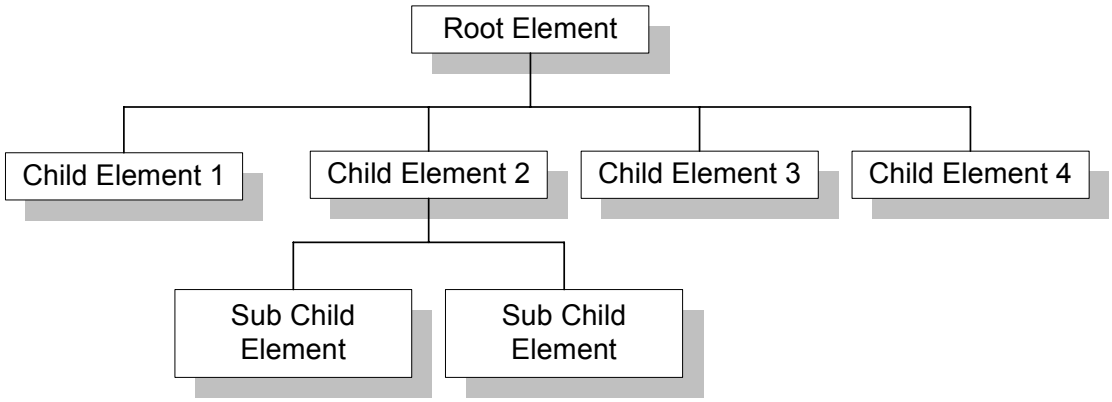
Heading standard untuk Document XML
Biasakanlah setiap membuat dokumen XML diawali dengan heading standard XML. Formatnya adalah sebagai berikut:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

Dokumen XML harus memiliki Root tag
Sebuah dokumen XML yang baik harus memiliki root tag. Yaitu tag yang melingkupi keseluruhan dari dokumen. Tag-tag yang lain, disebut child tag, berada didalam root membentuk hirarki seperti gambar 2.1

Contoh:

```
<root>
        <child>
                <subchild></subchild>
        </child>
</root>
```



Gambar 2.1 Diagram Hirarki XML

Tag pada XML harus lengkap berpasangan
Pada HTML beberapa elemen tidak harus berpasangan. Contoh berikut ini diperbolehkan dalam penulisan HTML.

```
<p>paragraph pertama
<p>paragarap kedua
```

yang demikian tidak berlaku pada XML. Kita harus menulis pula tag penutup untuk setiap tag yang kita buat. Penulisannya harus seperti ini

```
<p>paragraph pertama</p>
<p>paragarap kedua</p>
```

Tag tunggal hanya diperbolehkan untuk elemen kosong. Contoh penulisannya sebagai berikut:

```
<anggota nama="budi"/>
```

XML membedakan huruf besar dengan huruf kecil

Pada XML, <tanggal> berbeda dengan <Tanggal>. Tag pembuka dan tag penutup harus sama susunan huruf besar dan kecilnya.

```
<contoh>ini penulisan yang salah</Contoh>
<contoh>ini baru betul</contoh>
```

Penyarangan tag harus benar.

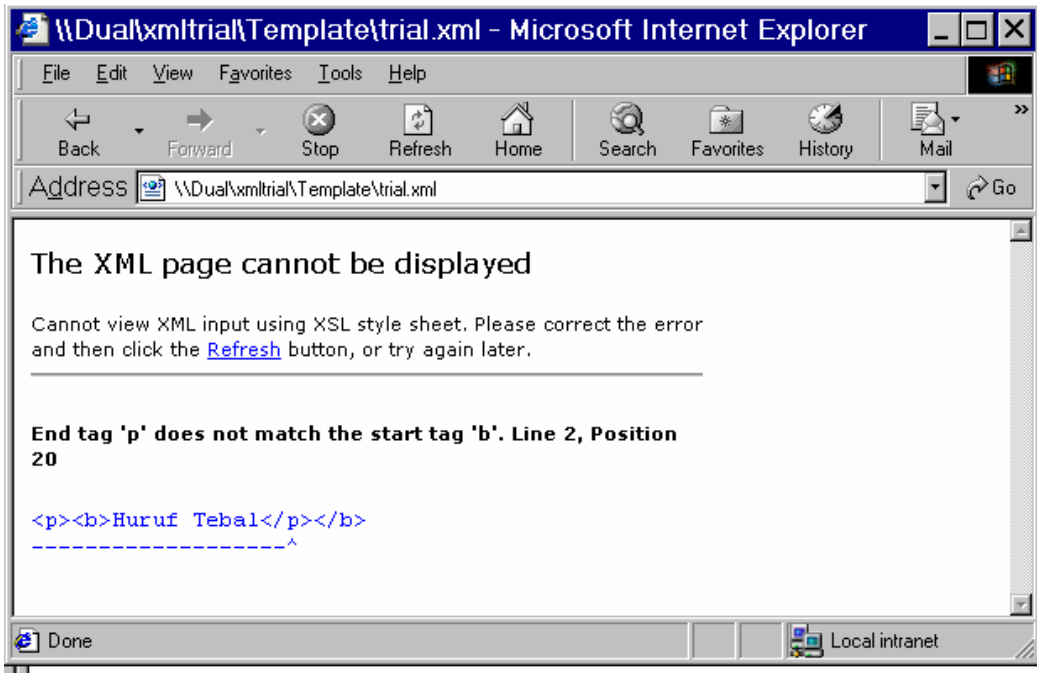
Penulisan tag pada XML harus mengikuti aturan Last In First Out (LIFO). seperti yang kita bahas terdahulu, pada XML kita tidak bisa membuat tag yang saling bersilang seperti dibawahini

```
<p><b>Huruf Tebal</p></b>
```

tapi harus disusun seperti ini

```
<p><b>Huruf tebal</b></p>
```

bila dipaksakan juga, browser akan menampilkan pesan error seperti gambar dibawah:



Gambar 2.2 Pesan Kesalahan yang ditampilkan browser

XML mempertahankan spasi seperti apa adanya

Berbeda dengan HTML, XML menampilkan spasi persis bagaimana data ditulis. Lebih jelasnya perhatikan contoh berikut ini:

Pada HTML kalimat

```
 Kami                    pergi bersama
```

akan ditampilkan sebagai:

```
Kami pergi bersama
```

Sedangkan pada XML akan ditampilkan sama persis dengan kalimat asalnya.

Nilai atribut harus diletakkan diantara tanda petik

Seperti HTML, XML memiliki atribut. Nilai atribut harus diletakkan diantara dua tanda petik. Tidak masalah apakah tanda petik tunggal atau tanda petik ganda. Contoh dibawah ini dua-duanya benar

```
<pesan dari=”lusy”>   atau
<pesan dari='lusy'>
```

Penamaan tag dan atribut

Nama tag bisa terdiri dari huruf, angka dan underscore("_"). Karakter awal nama tag harus berupa huruf atau underscore ("_"), tidak diawali dengan kata xml atau XML, (misal:<xmlstring>), dan tidak mengandung spasi. Aturan penamaan atribut sama dengan aturan penamaan tag.

Menyisipkan komentar
Pada bahasa pemrograman atau scripting kita mengenal adanya komentar (comment). Komentar adalah kalimat/baris yang tidak dieksekusi oleh compiler, browser atau parser. Untuk menyisipkan komentar pada dokumen XML caranya adalah sebagai berikut:

```
<!—Baris ini tidak di eksekusi oleh parser -->
```

Menggunakan Karakter Illegal pada XML
Sama seperti pada HTML, anda tidak bisa menggunakan karakter seperti kurung siku (< atau >), petik tunggal (‘), dan petik ganda (“) .
Contoh dibawah ini akan menghasilkan error kalau di eksekusi oleh browser.

```
<syarat>jika jumlah < 1000 maka</syarat>
```

Untuk menghindarinya, kita harus menggantikannya dengan entity reference seperti di bawah ini:

```
<syarat>jika jumlah &lt; 1000 maka</syarat>
```

Selengkapnya perhatikan tabel dibawah ini:

Entity references	Character	Character Name
<	<	Less Than
>	>	Greater then
&	&	Ampersand
'	‘	Apostrophe
"	“	Quotation mark

Perhatikan bahwa entity refence selalu diawali oleh & dan diakhiri ;
Karakter yang benar-benar illegal pada XML sebenarnya hanyalah < dan &. Yang lain bisa digunakan, akan tetapi lebih bijak kalo kita menghindarinya untuk menghindari kebingungan pada saat kita mendebugnya.
Kita bisa membuat sendiri entity reference untuk hal-hal lainnya. Pembahasan selengkapnya bisa anda baca pada bagian lain bab ini yang membahas Document Type Definition (DTD).

Namespace XML
Dari pembahasan terdahulu kita mengetahui bahwa tag-tag pada XML tidak didefinisikan secara baku tetapi kita buat sendiri sesuai keinginan kita. Karena itu akan sering terjadi konflik pada dua dokumen yang menggunakan nama tag yang sama tetapi mewakili dua hal yang berbeda. Contoh: bila ada dokumen yang mendiskripsikan tentang kebutuhan material pembuatan gardu jaga dari bambu

```
<bambu>
<jenis>Jawa</jenis>
<panjang>2</panjang>
</bambu>
```

dengan dokumen yang mendiskripsikan “bambu” sebagai merek produk.

```
<bambu>
<jumlah>246</jumlah>
<hargasatuan>200</hargasatuan>
</bambu>
```

untuk mengatasi hal ini, Namespace menyediakan metode dengan menggunakan awalan yang berbeda

```
<a:bambu>
<a:jenis>Jawa</a:jenis>
<a:panjang>2</a:panjang>
</a:bambu>
```

dokumen kedua menjadi seperti berikut

```
<b:bambu>
<b:jumlah>246</b:jumlah>
<b:hargasatuan>200</b:hargasatuan>
<b:/bambu>
```

Dengan cara demikian konflik penamaan tag tidak terjadi lagi. Aturan penggunaan namespace adalah sebagai berikut:

```
<a:bambu xmlns:a="http:/www.somewhere.com/gardu">
```

a adalah prefix yang dipakai, sedangkan atribut namespace di tambahkan pada tag. Syntax untuk atribut namespace adalah sebagai berikut:

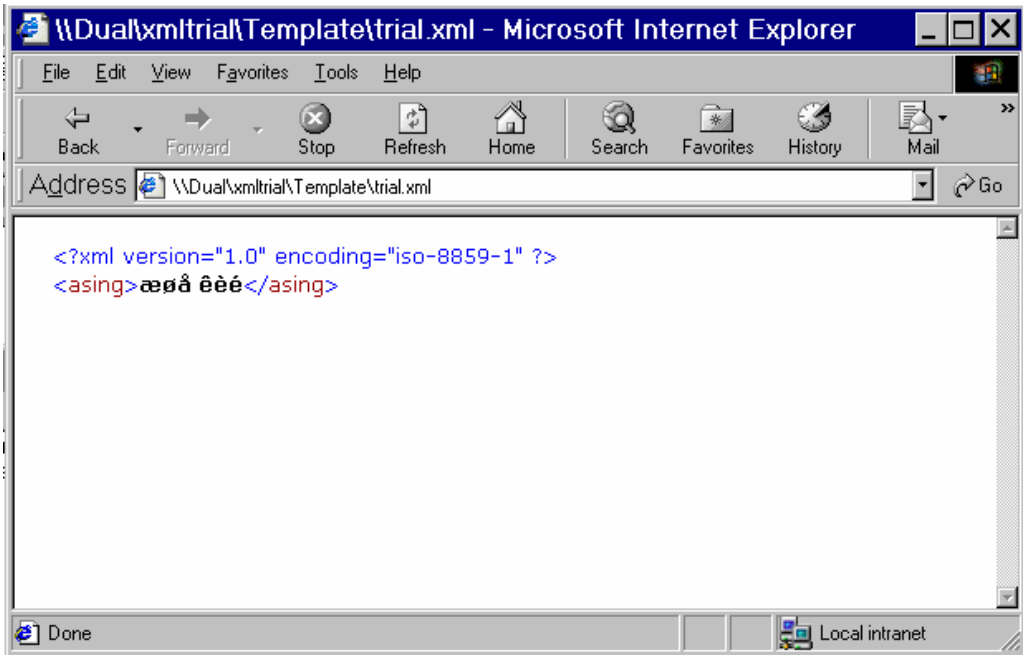
```
xmlns:a="namespace"
```

pada contoh diatas namespace-nya menggunakan alamat internet. W3C namespace spesification menyatakan bahwa namespace haruslah merupakan Uniform Resource Identifier (URI). Perlu di perhatikan bahwa alamat internet tersebut tidak digunakan untuk mendapatkan informasi, tapi hanya untuk memberikan nama yang unik bagi namespace. Tetapi seringkali perusahaan menggunakan namespace yang menunjuk pada webpage yang berisi informasi tentang namespace yang digunakan. Contoh yang disajikan pada Bab 1 menunjukkan penggunakan namespace xsl.

Text Encoding pada XML

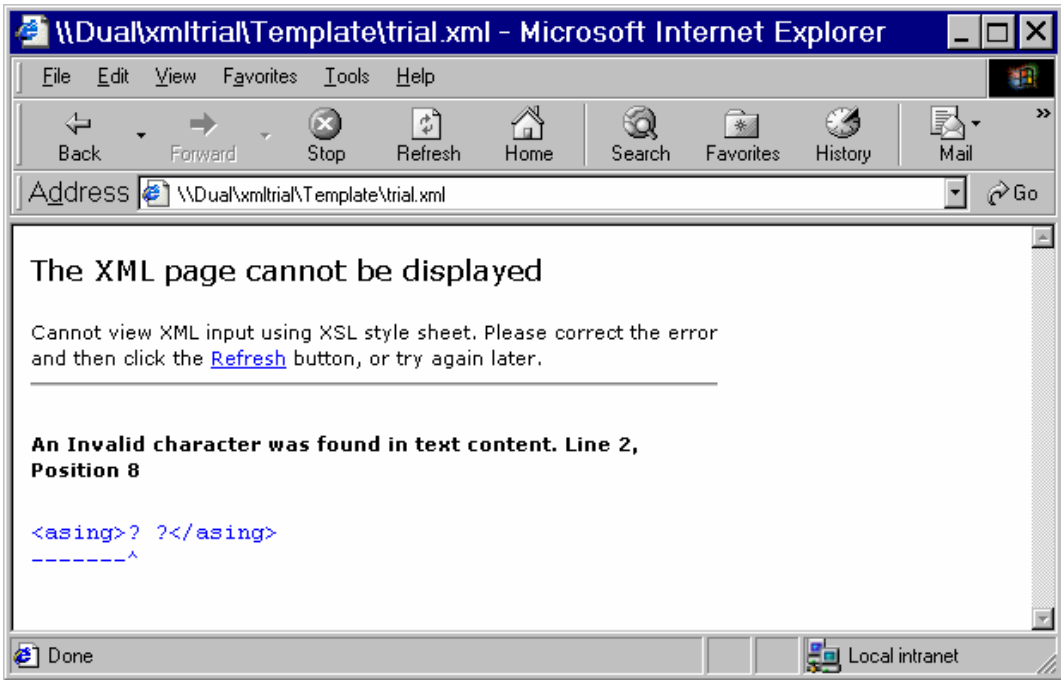
Pada bab 1 kita telah menyinggung sepintas tentang atribut encoding pada deklarasi XML. Contoh yang diberikan menggunakan encoding iso-8859-1 yaitu nama resmi dari text encoding yang oleh Microsoft disebut sebagai ANSI. Atribut encoding ini digunakan untuk memberitahukan pada XML Parser, text encoding apa yang digunakan.

Misalnya kita ingin menggunakan karakter asing seperti æøå êèé. Untuk menampilkannya dengan benar pada browser, kita harus menyimpan file XML kita dengan format unicode. Sayangnya notepad pada windows 95/98 tidak mendukung format unicode. Untuk itu kita harus menambahkan atribut encoding pada deklarasi XML kita. Kita bisa menggunakan encoding windows-1252 atau iso-8859-1. Kita tidak bisa menggunakan UTF-8 dan UTF-16 bila kita simpan file kita dengan notepad pada Windows 95/98.



Gambar 2.3 Karakter asing yang ditampilkan dengan benar

Notepad pada Windows 2000 bisa menyimpan dokumen dengan format unicode. Semua jenis encoding yang saya sebutkan diatas bisa digunakan. Tanpa menyertakan atribut encoding secara eksplisit pun dokumen yang disimpan dengan format unicode bisa menampilkan karakter asing dengan benar bila di buka dengan Internet Explorer 5.0 tapi bila dibuka dengan Netscape atribut encoding mutlak harus disertakan.



Gambar 2.3 Pesan kesalahan karena encoding tidak mendukung karakter asing

Sampai disini pengetahuan anda tentang XML sudah cukup untuk membuat sebuah dokumen xml yang baik (Well-formed Document) tetapi belum cukup untuk membuat dokumen XML yang valid (Valid Document). Karena untuk membuat document XML yang valid anda juga harus mengenal apa yang disebut Document Type Definition (DTD).

Document Type Definition (DTD)

Sesuai namanya DTD berfungsi untuk mendefinisikan tipe documen XML. Pada saat mempelajari salah satu bahasa pemrograman atau scripting anda diperkenalkan dengan deklarasi variable, deklarasi fungsi dan deklarasi tipe data. Serupa dengan itu, DTD mendefinisikan struktur dokumen XML dengan daftar element yang digunakan.
Contoh dibawah ini akan memperjelas pengertian DTD

```
<?xml version ='1.0' encoding='utf-8'?>
<!DOCTYPE organisasi [
<!ELEMENT organisasi (anggota)>
<!ELEMENT anggota (nama,alamat,kelamin,jabatan)>
<!ELEMENT nama (#PCDATA)>
<!ELEMENT alamat (#PCDATA)>
<!ELEMENT kelamin (#PCDATA)>
<!ELEMENT jabatan (#PCDATA)>

<!ENTITY org "Forum Komunikasi Remaja Masjid se-Jabotabek">
]>
<organisasi nama = “&org;”>
<anggota>
<nama>Budi Hermanto</nama>
<alamat>Kebayoran</alamat>
<kelamin>laki-laki</kelamin>
<jabatan>ketua</jabatan>
</anggota>
</organisasi>
```

Contoh diatas adalah DTD yang menjadi satu dalam satu dokumen dengan XML. Kita bisa memisahkan DTD pada file tersendiri, terpisah dari dokumen XML-nya. Caranya, perhatikan contoh berikut:

```
<?xml version ='1.0' encoding='utf-8'?>
<!DOCTYPE organisasi SYSTEM “organisasi.dtd”>
<organisasi nama = “&org;”>
<anggota>
<nama>Budi Hermanto</nama>
<alamat>Kebayoran</alamat>
<kelamin>laki-laki</kelamin>
<jabatan>ketua</jabatan>
</anggota>
</organisasi>
```

Lalu anda membuat file baru dengan nama “organisasi.dtd” yang berisi deklarasi DTD seperti contoh berikut

```
<!ELEMENT organisasi (anggota)>
<!ELEMENT anggota (nama,alamat,kelamin,jabatan)>
<!ELEMENT nama (#PCDATA)>
<!ELEMENT alamat (#PCDATA)>
<!ELEMENT kelamin (#PCDATA)>
<!ELEMENT jabatan (#PCDATA)>
```

```
<!ENTITY org "Forum Komunikasi Remaja Masjid se-Jabotabek">
```

DTD memungkinkan format yang unik untuk setiap file xml. DTD akan sangat berguna bila kita membuat aplikasi dalam Visual Basic, ASP atau bahasa pemrograman lain yang mendukung XML, yaitu untuk memastikan bahwa data yang diterima aplikasi itu adalah data yang valid. Atau bermanfaat juga digunakan bila satu organisasi menyepakati penggunaan satu DTD untuk tukar menukar data dan informasi.

Unsur-unsur yang dideklarasikan dalam DTD adalah semua unsur yang membentuk suatu dokumen XML yaitu

- **Element**, satu blok data yang diawali tag pembuka dan tag penutup.
- **Attribute**, informasi pendukung element yang disertakan pada tag pembuka
- **Entity**, karakter pengganti untuk sekumpulan informasi yang didefinisikan

Bagaimana mendeklarasikan masing-masing unsur diatas, kita akan bahas satu persatu:

Element

Bila sebuah element mengandung beberapa child element, maka kita perlu mendeklarasikan child element apa saja yang dipunyai element tersebut. Pada contoh diatas kita melihat deklarasi element

```
<!ELEMENT organisasi (anggota)>
<!ELEMENT anggota (nama,alamat,kelamin,jabatan)>
```

Maksudnya adalah element bernama organisasi memiliki satu child element bernama anggota. Lalu element bernama anggota itu sendiri mempunyai empat child element yang bernama nama, alamat, kelamin dan jabatan.
Setelah itu kita perlu medeklarasikan juga type dari element-element diatas.

```
<!ELEMENT nama (#PCDATA)>
<!ELEMENT alamat (#PCDATA)>
<!ELEMENT kelamin (#PCDATA)>
<!ELEMENT jabatan (#PCDATA)>
```

contoh diatas adalah cara untuk mendeklarasikan type elemen, dimana element nama, alamat, kelamin dan jabatan semuanya bertipe (#PCDATA)

Attribute

Agar dokumen XML kita valid, kita juga perlu mendefinisikan semua attribut yang akan kita gunakan dalam dokumen kita. Untuk mendefinisikannya kita akan menggunakan Attribute list declaration. Caranya seperti berikut:

```
<!ATTLIST namaelemen spesifikasiattribut>
```

namaelemen adalah nama elemen dimana attribute itu digunakan. Sedangkan spesifikasiattribute adalah serangkaian informasi tentang attribute itu. Unsur yang membentuknya antara lain nama attribut, type attribute, nilai awal (default value), dan sifat attribute”
perhatikan contoh dibawah:

```
<!ATTLIST ORGANISASI Nama CDATA #FIXED  “HMTE”>
```

maksud dari contoh diatas adalah elemen ORGANISASI memiliki Attribute *Nama* yang bertype CDATA. Sifat attribute dalam hal ini adalah #FIXED, yaitu nilai dari attribute nama harus seperti yang dideklarasikan ("HMTE" adalah nilai default yang diberikan bila kita tidak menyebutkannya). Sifat-sifat attribute selengkapnya adalah:

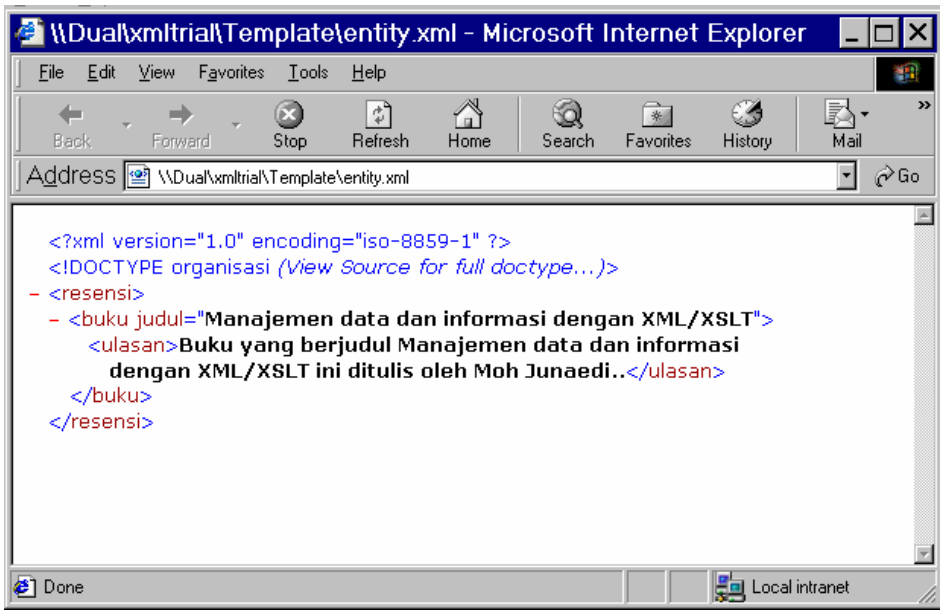
- #FIXED, bila attribute kita deklarasikan dengan ini, kita bisa mencantumkan atau menghilangkan attribute pada element bersangkutan. Bila kita menghilangkannya, secara otomatis parser akan mencantumkan attribute dengan nilai sesuai dengan nilai awal yang diberikan.
- #REQUIRED, ini menandakan bahwa attribut yang bersangkutan tidak bisa dihilangkan. Pendeklarasiannya tidak membutuhkan nilai awal. Bila anda paksakan, browser akan menampilkan pesan error.
- #IMPLIED, bila kita menggunakannya berarti kita bisa membuang atau mencantumkan attribute. Bila kita tidak mencantumkannya, prosesor tidak akan memberikan nilai default.

Entity

Dengan menggunakan entity XML kita bisa menggantikan kalimat yang panjang atau satu blok elemen yang sering kita gunakan dengan sebuah pengenal singkat. Misalnya kita ingin menggantikan kalimat "Manajemen Data dan Informasi dengan XML/XSL" dengan entity &judul;.(kita telah menyinggung pada pembahasan terdahulu bahwa entity diawali dengan & dan diakhiri dengan ;).
Sekali entity didefinisikan di dalam DTD, kita bisa menggunakannya dimana saja pada seluruh dokumen XML.

```
<?xml version="1.0" encoding="iso-8859-1">
<!DOCTYPE organisasi [
<!ENTITY judul "Manajemen data dan informasi dengan XML/XSLT">
]>
<resensi>
<buku judul="&judul;">
<ulasan>Buku yang berjudul &judul; ini ditulis oleh Moh
Junaedi..</ulasan>
</buku>
</resensi>
```

Bila dijalankan, pada browser akan tampil seperti gambar dibawah:



Gambar 2.4 Tampilan Dokumen XML dengan entity