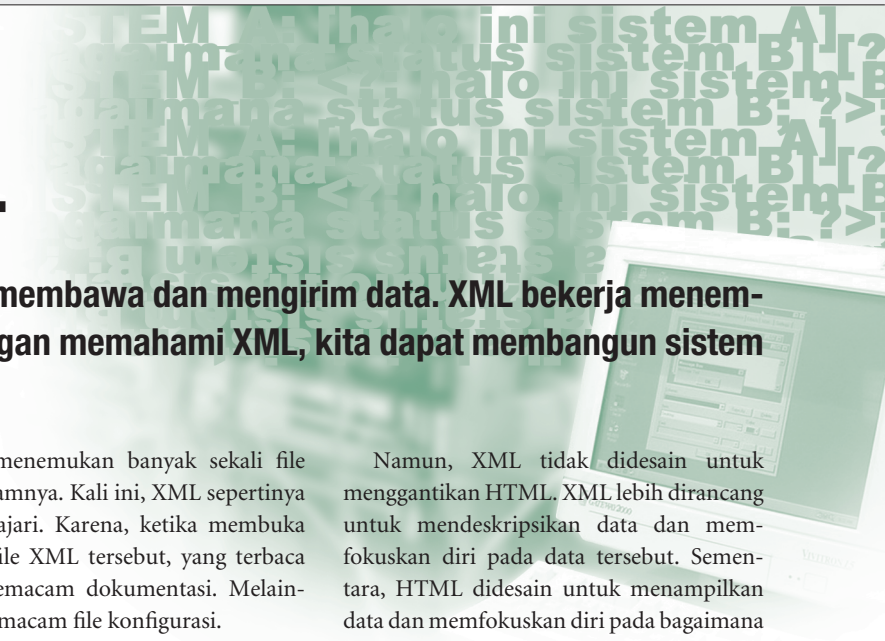
Mengenal XML



XML adalah teknologi universal untuk membawa dan mengirim data. XML bekerja menembus batas software dan hardware. Dengan memahami XML, kita dapat membangun sistem yang lebih universal.

Beberapa tahun yang lalu, saat penulis bekerja sebagai staf laboratorium software di sekolah, penulis mengenal seorang senior yang luar biasa unik. Beliau sangat mendalami algoritma dan sangat mampu mewujudkannya dengan bahasa C. *Scripting* **Perl** juga dikuasai dengan luar biasa. Begitupun dengan Linux. Setiap ada permasalahan dengan Linux, penulis hampir selalu menanyakan kepada beliau.

Beliau jarang sekali membeli buku yang umum ditemukan di pasaran Indonesia. Suatu hari, kami bertemu di salah satu ruangan lab, dan beliau membawa satu buku baru (yang kalau tidak salah dibeli di Mangga Dua) berjudul Mastering XML. Saat ini, penulis sesekali mendengar tentang XML dan tidak memiliki gambaran bagaimana XML akan mendorong perkembangan teknlogi komputer di masa depan.

Saat itu, penulis berpikiran bahwa XML mungkin adalah pengembangan lebih lanjut dari HTML dan, suatu hari akan menggantikan HTML dalam pembuatan halaman web. Karena kurang begitu tertarik akan web, maka penulis tidak peduli sama sekali akan XML, yang menurut penulis secara teknis mungkin HTML versi 5 yang namanya diubah. Suatu pemikiran yang rupanya salah.

Setelah lebih dalam berkenalan dengan dunia Linux dan *open source*, konsep penulis akan XML betul-betul menjadi berantakan. Bagaimana mungkin, sebuah teknologi pembuatan halaman web ditemukan pada paket-paket program seperti GNOME? Apakah mereka juga membangun *website* atau file-file XML tersebut adalah dokumentasi dalam halaman web?

Waktu berlalu dan seiring dengan kesibukan, XML pun terlupakan. Setelah bertemu dengan Ariya Hidayat (ariya@kde.org) yang sangat senang mempromosikan KDE, penulis pun sekali lagi berkenalan dengan XML. Kali ini karena ketika mempelajari KDE dan menemukan banyak sekali file XML di dalamnya. Kali ini, XML sepertinya harus dipelajari. Karena, ketika membuka salah satu file XML tersebut, yang terbaca bukanlah semacam dokumentasi. Melainkan lebih semacam file konfigurasi.

Dengar sana dengar sini, penulis mengetahui bahwa XML digunakan untuk menyimpan dan mendeskripsikan data. Masalahnya. Kenapa konfigurasi harus XML untuk sebuah file konfigurasi? Bukankah format berikut ini jauh lebih *readable*?

```
VARIABEL=NILAI
```

atau

```
VARIABEL:NILAI
```

Sejak itu, apalagi setelah berkenalan dengan seseorang rekan penulis, yang sangat mengagumi XML (baru pertama melihat penggemar XML seperti ini), penulis akhirnya menyadari bahwa XML adalah sesuatu yang luar biasa. Setiap developer, baik yang bekerja dengan sekedar file konfigurasi, konten, membangun format file, ataupun yang berhubungan dengan transmisi data, sangat dianjurkan bekerja dengan XML.

Apakah XML itu?

XML adalah sebuah teknologi *cross platform*, dan merupakan tool untuk melakukan transmisi informasi. XML bukanlah program, atau pustaka. XML adalah sebuah teknologi, sebuah standar dengan berbagai aturan tertentu.

Dalam pengertian yang sederhana, sebuah dokumen XML hanyalah sebuah file teks biasa yang berisikan berbagai tag yang didefinisikan sendiri oleh pembuat dokumen XML tersebut. Sesuai dengan namanya, *eXtensible Markup Language*, sebuah dokumen XML adalah sebuah dokumen dengan markup, sama seperti halnya dengan HTML.

Namun, XML tidak didesain untuk menggantikan HTML. XML lebih dirancang untuk mendeskripsikan data dan memfokuskan diri pada data tersebut. Sementara, HTML didesain untuk menampilkan data dan memfokuskan diri pada bagaimana data ditampilkan. Secara desain, hal ini sudah jauh berbeda. Dengan demikian, XML bukanlah pengganti HTML karena memang dirancang berbeda. Hubungan antara XML dan HTML lebih ke arah pelengkap. Anda dapat menyimpan data dalam sebuah dokumen XML dan mempergunakan HTML untuk menampilkan data tersebut.

Dokumen XML juga terdiri dari berbagai *tag*. Hanya, bedanya, tag-tag tersebut tidak memiliki standar khusus. Berbeda dengan tag <b> pada HTML yang berarti pemformatan teks tebal. Kita, sebagai pembuat dokumen lah yang harus menentukan tag dan artinya. Untuk menjaga agar tag-tag tersebut tetap berada di dalam lingkup jalan yang benar, maka keseluruhan aturan tag kita disimpan di dalam Document Type Definition (DTD) atau XML *Schema*. Dengan adanya aturan tersimpan di DTD atau XML Schema, maka sebuah dokumen XML diharapkan akan mampu mendeskripsikan diri sendiri (*self descriptive*). Boleh disamakan dengan tag HTML yang telah memiliki standar. Walaupun, pada XML, artinya akan lebih luas lagi.

Berikut ini adalah sebuah dokumen XML sederhana (1.xml):

```
<distro>
<os>Linux</os>
<name>SUSE</name>
<version>9.1</version>
<vendor>SUSE LINUX AG</vendor>
</distro>
```

Bisa kita lihat, sebenarnya 1.xml tersebut hanyalah sebuah file yang tidak berarti banyak. Tag-tag yang kita gunakan, <distro>,

<os>, <name> dan <vendor> juga tag yang digunakan oleh kita dengan maksud <os> adalah sistem operasi, <name> adalah nama distro dan <vendor> adalah vendor distro tersebut. Semuanya menerangkan distro.

Sebuah file XML tidak dapat berbuat apa-apa. File 1.xml di sini dibuat untuk membuat struktur data, menyimpan data dan mengandung nilai informasi. Struktur data diawali dengan menentukan tag-tag apa saja dan bagaimana aturannya. File tersebut juga menyimpan data distro Linux dan memiliki nilai informasi di dalamnya.

Bagaimanakah file XML tersebut bisa berguna? Ketika terdapat aplikasi yang dapat memproses data tersebut untuk tujuan tertentu, untuk ditampilkan misalnya. Bayangkanlah sebuah tampilan yang informasinya diambil dari file XML tersebut:

Nama Distro	OS	Versi	Vendor
SUSE	Linux	9.1	SUSE LINUX AG

Dengan menyimpan informasi tersebut dalam file XML, informasi dapat sampai ke siapa saja, dengan perantaraan apapun juga. Artinya, selama sistem tujuan bisa memproses XML tersebut, maka informasinya juga akan sampai kepada sistem tujuan.

Saat ini, untuk memroses tag-tag XML tersebut, tentu saja kita tidak perlu melakukannya sendiri secara manual. Terdapat banyak parser untuk melakukannya. Algoritma yang digunakan juga berbeda-beda. Bayangkan jika Anda harus membaca satu per satu karakter dengan bahasa C, menentukan yang mana tag dan yang kapan tag tersebut ditutup dan menyimpannya ke dalam variabel. Tidak, tidak perlu.

Sebenarnya, itulah yang sempat penulis pikirkan ketika mengetahui XML dapat digunakan sebagai file konfigurasi. Penulis berpikir, tentunya akan repot sekali memparsing file konfigurasi tersebut. Seperti pada awal tulisan, bukanlah dengan mudah dengan format VARIABEL=NILAI?

Setiap bahasa pemrograman umumnya telah melengkapi dirinya dengan kemampuan bekerja dengan XML. Perl, Python, Java, C dan apa saja.

Kita akan melihat contoh penerapan XML sebelum kita melanjutkan lebih dalam tentang XML.

Contoh penerapan XML di open source

XML sendiri adalah teknologi bebas dan dapat dikembangkan. Sangat sesuai dengan dunia open source. Namun, nilai praktisnya telah digunakan di mana-mana. Kita akan melihat contoh-contoh penerapan XML di dunia open source.

XML digunakan sebagai file konfigurasi. Coba saja amati file konfigurasi salah satu proyek open source yang paling Anda rasa populer. Berani bertaruh, file konfigurasi tersebut disimpan sebagai dokumen XML. Dengan demikian, pengambilan variabel dan nilai dapat dilakukan dengan cara yang praktis, apalagi jika aturan dan tagnya terdefinisi dengan baik. Apabila kita menulis file konfigurasi dengan cara masing-masing, bisa Anda bayangkan beberapa bentuk berikut:

```
VARIABEL=NILAI

[VARIABEL] NILAI

VARIABEL NILAI
```

Cara apapun yang kita inginkan, bisa saja. Tidak ada yang akan melarang. Namun, bayangkan jika Anda menggunakan yang bagus bagi Anda, namun aneh bagi yang lain. Tentu saja, pengembang aplikasi lain harus mencari tahu bagaimana memparsing konfigurasi Anda. Belum lagi jika ada aplikasi lain yang ingin ikut memparsing pula. Semua kondisi tersebut akan berujung pada banyak parsing untuk banyak aturan file konfigurasi. Tidak ada yang universal. Dengan XML, semuanya menjadi universal.

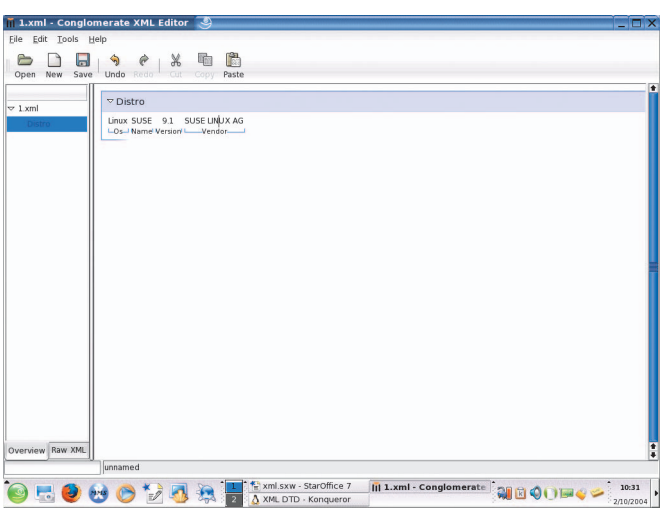
Bayangkan jika ditulis dalam format XML berikut:

```
<konfigurasi>
<variabel>nilai</variabel>
</konfigurasi>
```

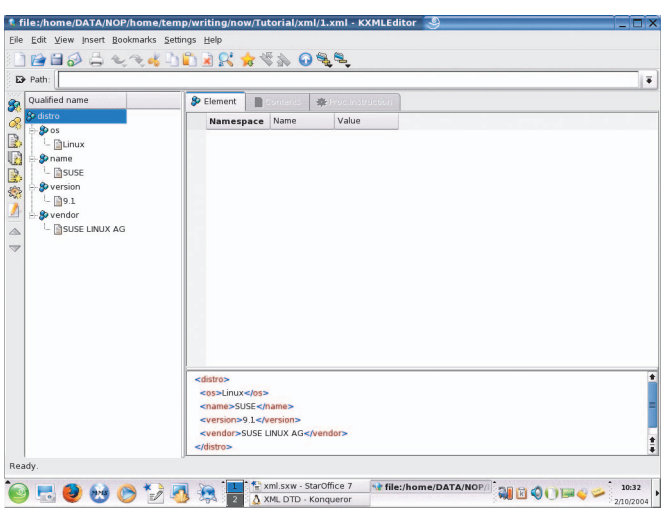
Hanya ada satu sistem yang diperlukan untuk memparsing konfigurasi tersebut. Yaitu, XML.

Penerapan kedua adalah untuk membangun *user interface*. Sudah bukan zamannya lagi untuk membekukan komponen control program ke dalam programnya itu sendiri. Tidak akan ada gunanya membangun aplikasi yang menu dan menu itemnya ditulis mati dalam program utama. Cepat, pintas, namun tidak ekstensibel.

Pernahkan Anda memperhatikan Kpart KDE? Penulis memberikan penghargaan



Conglomerate XML Editor.



Mengedit XML dengan KxmlEditor.

dan kekaguman yang sebesar-besarnya, setinggi-tingginya, sedalam-dalamnya kepada Kpart milik KDE. Anda pernah menggunakan Kontact? Itulah salah satu contoh kehebatan Kpart milik KDE. Semuanya bisa berupa komponen. Komponen-komponen Kontact sendiri, seperti halnya Kmail, KaddressBook adalah aplikasi terpisah. Namun, bisa diembed begitu saja ke dalam Kontact.

Memperhatikan Mac dengan global menunya (dulu)? Setiap aplikasi yang dijalankan akan mendaftarkan dirinya ke menu global Mac sehingga hanya diperlukan satu toolbar untuk segalanya. Menunya bisa dinamis.

Kedua hal tersebut bisa didekati dengan penggunaan XML. Anda mendefinisikan komponen menu dengan XML, sementara, kontainer akan mengartikan XML tersebut dengan aturan tertentu, dan kemudian dapat menempatkan menu sesuai konteks aplikasi. Tentu saja, part sistem tidak sesederhana ini. Namun, pendekatan XML akan sangat membantu, terutama dalam user interface design.

Penerapan ketiga adalah format dokumen, seperti yang digunakan oleh OpenOffice.org. Paket office yang saat ini sangat populer ini menggunakan file teks XML sebagai format filenya (http://xml.openoffice.org). Hal ini berlawanan sekali dengan Microsoft yang menggunakan format yang hanya diketahui oleh Microsoft tersebut.

Dengan menggunakan XML, kita tidak perlu repot-repot memikirkan format file. Untuk merancang suatu format file bukanlah perkara mudah. Kita harus benar-benar memikirkan format file kita bisa ekstensibel dan mudah ditelusuri. Merancang format file bisa memakan waktu tahunan. Dengan menggunakan XML, kita hanya cukup menentukan skema XML kita dan dapat segera memiliki format file sendiri.

Apabila Anda tidak suka dengan penggunaan satu file, Anda juga bisa memilih pendekatan OpenOffice.org. Beberapa file Anda kumpulkan, berikan metafile, lalu kompreslah semuanya menjadi satu. Berikan ekstensi yang Anda rasa keren, dan jadilah format file baru. Keren, bukan?

Kehadiran XML harus diakui telah memicu lahirnya banyak format dokumen baru. Sebut saja format dokumen paket office di dunia open source. Koffice, Abiword, Gnumeric, semuanya XML.

Dengan prinsip ini, pertukaran informasi akan menjadi lebih mudah. Katakanlah sebuah sistem menggunakan XML sebagai *format* filenya. Ketika akan dikirimkan ke sistem lain yang benar-benar tidak kompatibel (misal dari *Mainframe* ke PDA), format XML ini tetap bisa diandalkan. Informasi tetap terjaga. Namun, hal sebaliknya akan terjadi apabila kita membuat format file sendiri. Apabila kita tidak mampu menyediakan dukungan, pemindahan informasi dari satu sistem ke sistem lain akan menjadi masalah besar.

Karena sifatnya inilah, banyak yang mengatakan bahwa XML adalah format dokumen masa depan. Selama ada XML parser dan aturan XML, maka semua format dokumen akan sangat mudah dimengerti. Diharapkan, ketergantungan terhadap pihak tertentu bisa diminimasi.

Penerapan berikutnya adalah pengiriman informasi secara langsung. Bayangkan kedua sistem yang berbeda tersebut ini ketika ingin berkomunikasi:

```
SISTEM_A: [halo ini sistem A]
[bagaimana status sistem B] [?]
SISTEM_B: <?; halo ini sistem B;
bagaimana status sistem B; ?>;
```

Ketika keduanya harus saling berkomunikasi, bagaimanakah caranya agar SISTEM_A dan SISTEM_B bisa berkomunikasi dengan benar? Buat protokol baru? Suatu pekerjaan yang sangat boros *resource*, dengan hasil tidak sebesar pengorbanannya.

Bayangkan jika SISTEM_A dan SISTEM_B bekerja dengan cara berikut:

```
<message>
<from>SISTEM_A</from>
<to>SISTEM_B</to>
<subject>System status</subject>
<body>Bagaimana status sistem
B?</body>
</message>
```

Tentu saja, apabila SISTEM_A dan SISTEM_B memahami aturan XML, maka percakapan bisa dilakukan, walaupun berbeda SISTEM.

Hal inilah yang dilakukan Jabber. Berkomunikasi dengan bantuan XML. Dengan penggunaan XML di sini, tidak banyak pengorbanan yang harus dilakukan apabila suatu sistem harus berkomunikasi dengan sistem lain yang tidak kompatibel.

Masih banyak lagi penerapan XML pada komunitas Open Source. Hal ini wajar saja. Di mana lagi bisa mendapatkan begitu banyak manfaat dengan begitu sedikit pengorbanan? Luar biasa sekali bukan, XML ini?

XML sendiri bukanlah sistem yang sederhana. Betul bahwa XML adalah *markup based document*. Namun, kita perlu memahami beberapa hal ketika bekerja dengan XML untuk mendapatkan manfaat yang sebesar-besarnya. Termasuk menghindari kesalahan yang tidak perlu.

Sintaks XML

Sama seperti bahasa pemrograman misalnya, XML juga memiliki sintaks (walaupun tidak akan terlalu kaku). Kita harus memenuhi aturan-aturan ini agar sistem tetap sesuai standar. Menggunakan hal standar secara tidak standar adalah hal yang jelek sekali.

Aplikasi Anda, ketika melakukan parsing XML haruslah berhati-hati. Tidak semua XML valid. Apabila hal yang salah ditemukan, maka pemrosesan sangat disarankan untuk dihentikan.

Untuk lebih baiknya, kita akan segera membahas sintaks-sintaks XML.

1. Semua elemen XML harus memiliki *closing* tag. Hal ini berbeda dengan HTML. Perhatikan contoh HTML dan XML berikut ini:

   ```
   <p>ini adalah paragraph
   <hr>garis pemisah horizontal
   ```

   Pada XML, apabila menggunakan tag <p> dan <hr>, maka harus dituliskan seperti ini:

   ```
   <p>ini adalah paragraph</p>
   <hr>garis pemisah horizontal</
   hr>
   ```

2. tag adalah *case sensitive*. Hal ini berbeda dengan HTML yang case insensitive. Harap perhatikan betul masalah yang satu ini. Adalah jamak kalau kita menuliskan tag <message> dan lalai ditutup dengan </Message>.
3. Semua tag XML bersarang harus ditulis dengan benar. Perhatikan contoh HTML dan XML berikut:

```
<b><i>teks tebal dan miring</
b></i>
```

Di XML, tag <i> harus ditutup dahulu, barulah <b>. Seperti Dancow, ini dulu, baru itu.

```
<b><i>teks tebal dan miring</
i></b>
```

4. Semua elemen harus memiliki elemen root. Pada contoh pesan berikut ini, elemen <message> adalah elemen root.

```
<message>
<from>SISTEM_A</from>
<to>SISTEM_B</to>
<subject>System status</
subject>
<body>Bagaimana status sistem
B?</body>
</message>
```

5. Penggunaan atribut harus selalu dikutip. Tag pada XML bisa memiliki atribut. Perhatikan beda HTML dan XML berikut:

```
<img src=a.png>
```

pada XML, harus dituliskan sebagai:

```
<img src="a.png"></img>
```

Pembahasan mengenai atribut akan kita bahas setelah bagian ini.

6. White space akan diperhatikan di XML. Berbeda dengan HTML, white space pada XML akan tetap diperhatikan. Bedakan dengan HTML berikut ini:

```
<b>halo          apa kabar</b>
```

pada HTML, akan dituliskan sebagai:

```
hao apa  kabar
```

Sementara, pada XML akan dituliskan apa adanya.

7. Karakter *newline* XML adalah LF (sama seperti Unix dan Linux). Hal ini berbeda dengan Windows (CR dan LF) atau Mac (CR).
8. Untuk komentar, sama seperti HTML, tuliskan dalam <!-- dan -->. Contoh:

```
<!-- ini adalah komentar -->
```

Aturan-aturan tersebut cukup untuk pengunaan XML secara mendasar. Lebih lanjut, kita akan melanjutkan pada elemen-elemen dalam XML.

Elemen-elemen XML

Kita telah melihat bahwa secara sederhana, XML hanyalah terdiri dari TAG dengan definisi tag dan sejumlah aturan. Kita akan melihat lebih dalam sekarang. Perhatikanlah contoh pertama tadi:

```
<distro>
<os>Linux</os>
<name>SUSE</name>
<version>9.1</version>
<vendor>SUSE LINUX AG</vendor>
</distro>
```

XML tersebut bisa ditulis ulang sebagai berikut:

```
<distro os="Linux">
<name>SUSE</name>
<version>9.1</version>
<vendor>SUSE LINUX AG</vendor>
</distro>
```

Atau:

```
<distro>
<os>Linux</os>
<name>SUSE</name>
<version>
  <major>9</major>
  <minor>1</minor>
</version>
<vendor>SUSE LINUX AG</vendor>
</distro>
```

Dari contoh-contoh tersebut, kita melihat bahwa XML bisa dituliskan dengan berbagai cara. Perhaikanlah elemen tag tersebut. Sebuah elemen dapat mengandung sesuatu. Pada contoh-contoh tersebut, kita melihat bahwa :

- elemen dapat mengandung *element content* seperti pada:

```
<version>
  <major>9</major>
  <minor>1</minor>
</version>
```

- elemen dapat mengandung *simple* content atau text content seperti pada:

```
<name>SUSE</name>
```

- Elemen dapat mengandung *mixed* content apabila menggabungkan element content dan simple content.
- Elemen dapat memiliki *empty* content apabila tidak memiliki nilai informasi apapun secara eksplisit. Umumnya, di sini, atribut digunakan. Contoh:

```
<version major="9"
minor="1"></version>
```

- Elemen juga memiliki atribut seperti:

```
<distro os="Linux">
```

Seperti yang kita lihat, atribut cukup banyak digunakan. Lantas, apakah pendekatan yang lebih baik digunakan? Penggunaan atribut atau tidak? Menurut beberapa pembahasan, penggunaan atribut agak sedikit merepotkan dan sebaiknya dihindari. Berikut ini adalah beberapa permasalahan sehubungan dengan penggunaan atribut.

- atribut tidak dapat mengandung banyak nilai. Apabila diinginkan, maka beberapa atribut harus digunakan.
- Atribut tidak mudah untuk dikembangkan lebih lanjut
- atribut tidak dapat digunakan untuk mendeskripsikan struktur.
- Atribut lebih susah dimanipulasi oleh program
- nilai atribut tidak mudah diuji dengan penggunaan DTD.

Bagaimanapun, terkadang atribut tidak selalu jelek dan merepotkan. Kita dapat memberikan nilai id pada suatu elemen, dan berfungsi sebagaimana halnya name pada HTML. Pendekatan ini juga digunakan oleh Jabber. Seperti contoh berikut:

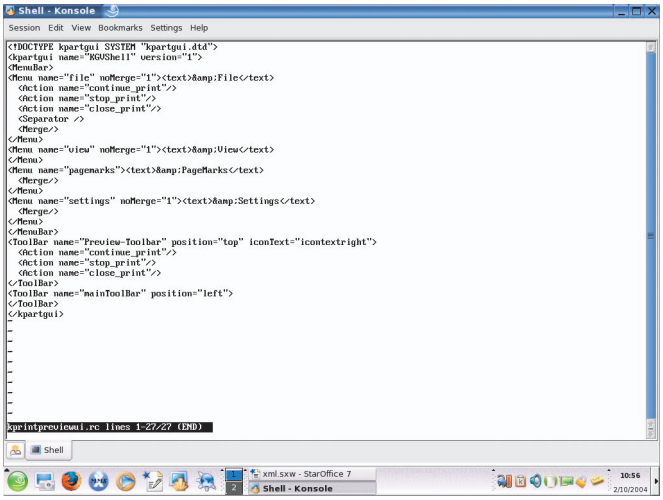
```
<message id="10112">
...
...
</message>
```

Kita lanjutkan. Apabila Anda memilih untuk menggunakan child element sebisa mungkin, maka tulislah sebaik mungkin. Rancanglah strukturnya agar lebih mudah untuk pengembangan lebih lanjut. Seperti pada contoh sebelumnya, kita bisa menuliskan <version> seperti ini:

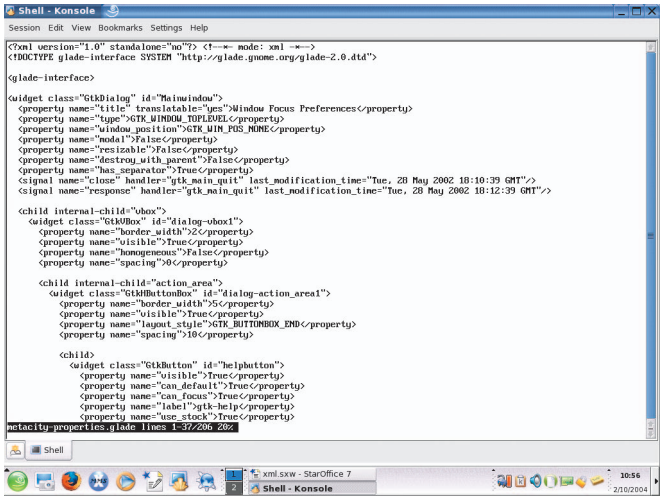
```
<version>9.1</version>
```

Namun, dapat juga dituliskan lebih baik dengan cara:

```
<version>
  <major>9</major>
  <minor>1</minor>
</version>
```



Konfigurasi KDE berbasis XML.



Konfigurasi GNOME berbasis XML.

Sementara, apabila Anda memutuskan untuk banyak menggunakan atribut, maka berhati-hatilah dalam merancang struktur XML sehubungan dengan banyaknya penggunaan atribut. Tentu saja, ini bukanlah sebuah XML yang sesungguhnya:

```
<distro os="Linux" name="SUSE"
version="9.1" vendor="SUSE LINUX
AG"></distro>
```

Penulisan elemen XML

Anda bebas dalam merancang tag yang digunakan, selama memenuhi aturan berikut:

- nama elemen dapat terdiri dari huruf, bilangan dan karakter lainnya.
- Nama elemen tidak boleh diawali dengan bilangan ataupun tanda baca.
- Nama tidak boleh diawali dengan XML (dan variasinya seperti xml, Xml...)
- nama tidak boleh berisi spasi.

Karena Anda mendefinisikan tag sendiri, tidak akan ada yang akan melarang tag yang Anda gunakan selama telah memenuhi beberapa aturan sebelumnya. Hanya, perhatikan betul agar umum digunakan dan tidak bermasalah di sistem lain. Perhatikan contoh berikut.

Lebih baik untuk tidak menggunakan –, :, dan . dalam tag. Karakter – bisa dianggap pengurangan dan karakter . dapat dianggap pemanggilan atribut kelas. Karakter : yang akan digunakan dalam *namespace* juga tidak boleh digunakan.

Nama haruslah sependek dan seefisien mungkin. Sebagai contoh:

```
<version>9.1</version>
```

Lebih baik digunakan daripada bentuk berikut:

```
<distro_linux_version>9.1</
distro_linux_version>
```

Karena tag <version> adalah anak dari <distro> dan kita memiliki tag <os> yang menginformasikan sistem operasi (dalam hal ini Linux).

Validasi XML

Kebebasan selalu merupakan pisau bermata dua. Apabila digunakan dengan baik, maka hasilnya akan berguna. Namun, apabila disalah gunakan, bisa-bisa terjadi masalah besar.

Di XML, bagaimanakah Anda tahu sebuah dokumen XML valid atau tidak secara sintaksis? Kita bisa membagi XML yang benar dalam dua bagian: XML yang secara struktural benar dan XML yang valid.

Contoh XML yang secara struktural benar:

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<distro>
<os>Linux</os>
<name>SUSE</name>
<version>
  <major>9</major>
  <minor>1</minor>
</version>
<vendor>SUSE LINUX AG</vendor>
</distro>
```

Namun, aturan XML tersebut tidak didefinisikan. Oleh karena itu, kita tidak bisa mengatakan XML tersebut valid. Perhatikan contoh berikut:

```
<?xml version="1.0"
encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "distro.
dtd">
<distro>
<os>Linux</os>
<name>SUSE</name>
<version>
  <major>9</major>
  <minor>1</minor>
</version>
<vendor>SUSE LINUX AG</vendor>
</distro>
```

Kali ini, kita memiliki sebuah file dengan nama distro.dtd yang merupakan DTD untuk XML kita. Apabila semua bagian XML sesuai dengan aturan dalam DTD, maka XML dikatakan valid.

Yang terakhir, sebagai penutup, sebaiknya, dalam menulis XML, kita menggunakan XML editor, jangan Vim, pico, fte atau editor teks biasa. Penggunaan editor khusus XML tidak hanya memberikan syntax highlight, namun juga lengkap dengan berbagai fitur khusus XML. Contoh XML editor untuk GNOME dan KDE adalah Conglomerate (GNOME) dan KxmlEditor (KDE).

Masih banyak lagi hal seputar XML yang menggambarkan betapa hebatnya XML. Pembahasan kita kali ini hanya menyentuh dasar-dasar XML. Sampai ketemu di pembahasan berikutnya!

Noprianto *(noprianto@infolinux.co.id)*