

MODUL PRAKTIKUM

PENGANTAR SISTEM INFORMASI TERDISTRIBUSI

D3 KOMPUTER DAN SISTEM INFORMASI



Penyusun:

Kabul Kurniawan, S.Kom

LABORATORIUM KOMPUTER

D3 KOMPUTER DAN SISTEM INFORMASI

SEKOLAH VOKASI UNIVERSITAS GADJAH MADA

Daftar Isi

BAB 1 Pengantar XML

Pengenalan dan Penggunaan XML	4
-------------------------------------	---

BAB 2 XML Fundamental

Dokumen XML dan File XML	7
Elemen, Tag, dan Data karakter	7
Atribut	10
XML Name	10
Entity Reference.....	11
CDATA Section.....	11
Komentar.....	12
Processing Instruction	13
Deklarasi XML	14
Pembuatan File XML.....	15

BAB 3 Document Type Definition (DTD)

Validation.....	19
Element Declarations	21
Deklarasi Atribut.....	23
General Entity Declaration.....	25
External Parsed General Entities	25
External Unparsed Entities and Notations.....	26
Parameter Entities	27
Conditional Inclusion	28
Dua Contoh DTD	28
Elemen dan Atribut DTD	28

BAB 4 *EXtensible Stylesheet Language Transform (XSLT)*

Pengenalan XSLT	29
<i>An Example Input Document</i>	29
<i>xsl:stylesheet</i> dan <i>xsl:transform</i>	30

<i>Template and Template Rules</i>	30
<i>Calculating the Value of an Element with <code>xsl:value-of</code>.....</i>	31
<i>Applying Templates with <code>xsl:apply-templates</code></i>	32
<i>The Built-in Template Rules.....</i>	32
<i>Modes.....</i>	33
<i>Attribute Value Templates.....</i>	34
<i>XSLT and Namespaces.....</i>	34
Menjalankan File XML Dalam Bentuk HTML Menggunakan XSL.....	35
Parsing data XML menggunakan PHP.....	36
Menampilkan Isi Database ke Dalam Bentuk HTML	41
Menambah Data ke Database Menggunakan XML.....	44
BAB 5 WEB SERVICE DESCRIPTION LANGUAGE (WSDL)	
Pengantar Web Service	47
Arsitektur Web Service	48
WSDL	49
<i>The WSDL Document Structure</i>	49
<i>The Full WSDL Syntax.....</i>	51
BAB 6 Simple Object Access Protocol (SOAP)	
Pengantar SOAP	53
SOAP Syntax	53
Contoh SOAP.....	55
Element Fault	56
Memproses SOAP Message	56
BAB 7 UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION (UDDI)	
Pengantar UDDI.....	58
Arsitektur Teknik	59
BAB 8 IMPLEMENTASI WEB SERVICE	
Toolkit NuSOAP.....	63
Implementasi Web Service menggunakan Toolkit NuSOAP	64

BAB 1

PENGANTAR XML

Tujuan :

- Praktikan dapat mengenal dasar-dasar XML dan penggunaan XML
- Praktikan mengerti pohon dan sintaks dan struktur data XML
- Praktikan mengerti tentang XML Namespace dan Skema pada dokumen XML

1.1 Pengenalan XML dan Penggunaan XML

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">). Berikut Perbedaan XML dengan HTML:

- a. XML didesain untuk mengangkut data dan menyimpan data, sehingga berfokus pada apa data tersebut.
- b. HTML didesain untuk menampilkan data, sehingga berfokus pada bagaimana data tersebut dilihat.
- c. XML merupakan pembawa informasi, sedangkan HTML merupakan penampil informasi

Mungkin akan sedikit sulit dipahami, namun XML tidak melakukan apapun. XML dibuat untuk menstrukturisasi, menyimpan dan membawa informasi. Contoh dibawah ini merupakan sebuah catatan dari Rudi kepada Budi, disimpan sebagai XML :

```
<catatan>
<kepada>Budi</kepada>
<dari>Rudi</dari>
<judul>Pengingat</judul>
<isi>Jangan lupa nanti sore futsal ya!</isi>
</catatan>
```

catatan diatas didiskripsikan sendiri, catatan tersebut memiliki informasi pengirim dan penerima, selain itu juga memiliki judul dan isi pesan. Tetapi dokumen XML ini tidak melakukan apapun. XML tersebut hanya membungkus informasi dalam *tag-tag*. Seseorang harus menulis sebuah *software* perangkat lunak untuk mengirim, menerima atau menampilkan informasi tersebut.

Tag-tag pada contoh diatas (seperti <kepada> dan <dari >) tidak didefinisikan pada standar XML manapun, tag-tag tersebut “ditemukan” dan “dibuat” sendiri oleh penulis dari dokumen XML tersebut. Hal ini karena bahasa XML tidak memiliki tag-tag yang sudah terdefinisi (predefined). Tag-tag yang digunakan pada dokumen HTML telah *predefined*. Dokumen HTML hanya dapat menggunakan tag-tag yang didefinisikan pada standar HTML (seperti <p>, <h1>, dsb.). XML membolehkan penulis untuk mendefinisika sendiri tag-tag dan struktur dokumen miliknya.

XML bukan merupakan pengganti HTML. Pada hampir semua aplikasi berbasis WEB, XML digunakan untuk mengangkut data sedangkan HTML digunakan untuk memformat dan menampilkan data. XML ada dimana-mana, XML sekarang sangat penting untuk Web seperti HTML yang merupakan pondasi dari sebuah Web. XML merupakan tool yang paling umum untuk mentransmisikan data diatara semua aplikasi-aplikasi.

Kemudian bagaimana XML digunakan?, XML digunakan pada semua aspek pengembangan web, seringnya untuk menyederhanakan penyimpanan dan sharing data . XML digunakan antara lain untuk :

- *Memisahkan data dari HTML*

Jika kita membutuhkan untuk menampilkan data dinamis pada dokumen HTML, ini akan menghabiskan banyak waktu dan tenaga untuk mengedit HTML setiap perubahan data. Dengan XML, data dapat disimpan pada file XML. Dengan sedikit kode JavaScript, kita dapat membaca file XML ekstternal dan mengupdate konten data yang ada dihalaman web.

- *Menyederhanakan sharing data*

Pada dunia nyata, sistem komputer dan database terdiri dari data pada format yang tidak kompatibel. Data XML disimpan pada sebuah format *plain text*. Hal ini untuk menyediakan sebuah independen software dan hardware sebuah cara untuk menyimpan data. Hal ini membuat data lebih mudah untuk membuat data yang dapat di-*share* oleh aplikasi yang berbeda-beda.

- *Menyederhanakan Transport Data.*

Satu dari sekian banyak tantangan konsumsi waktu bagi para developer adalah untuk mempertukarkan data dari berbagai sistem yang tidak kompatibel.

- *Menyederhanakan Perubahan Platform*

Mengupgrade ke sistem yang baru (platform hardware atau software) selalu memakan banyak waktu. Sejumlah banyak data yang harus dikonversi dan data yang tidak kompatibel seringnya hilang.

- Membuat data lebih tersedia (*available*)

Aplikasi yang berbeda dapat mengakses data kita, tidak hanya pada halaman HTML tetapi juga dari sumber data XML. Dengan XML, data kita dapat tersedia untuk semua jenis “mesin pembaca” dan membuatnya lebih berguna untuk orang buta dan orang cacat.

- *Digunakan untuk membuat bahasa internet baru*

Banyak dari bahasa baru di internet dibuat menggunakan XML., berikut contohnya:

- ✓ XHTML
- ✓ WSDL (untuk mendeskripsikan web service yang tersedia).
- ✓ WAP dan WML sebagai bahasa *mark-up* untuk handle peralatan.
- ✓ Bahasa RSS untuk *feeds* berita
- ✓ RDF dan OWL untuk mendeskripsikan sumber-sumber dan ontology
- ✓ SMIL untuk mendeskripsikan multimedia untuk web

BAB II

XML FUNDAMENTAL

Tujuan :

- Praktikan dapat membuat dokumen XML dan file XML
- Praktikan dapat membedakan antara tag, elemen, atribut, Cdata, entity serta penggunaannya

2.1 Dokumen XML dan File XML

Dokumen XML mengandung data text bukan data *binary*. Sehingga dokumen tersebut dapat dibuka atau diubah menggunakan aplikasi text editor seperti motepad, wordpad, notepad++, dreamweaver atau kita dapat langsung menggunakan aplikasi XML editor. Berikut adalah contoh penulisan dokumen XML sederhana, dokumen yang well-formed XML sehingga XML parser dapat membaca dan memahaminya.

```
<person>
  Alan Turing
</person>
```

Pada dokumen diatas, terlihat bahwa penulisan dokumen XML mengandung informasi yang diapit oleh tag. Dalam standar penulisan yang paling umum, file diatas akan disimpan dengan nama *person.xml*, atau *223.xml* bahkan *person.txt* sekalipun karena tidaklah mengutamakan nama filenya, tetapi isi dari file/dokumen tersebut. Bahkan dokumen XML pun dapat disimpan tidak didalam file. Kita dapat melakukan pencatatannya dengan menggunakan *database* yang dapat dihasilkan dengan cepat menggunakan program CGI untuk merespon perintah dari browser. Dokumen XML bahkan dapat disimpan lebih dari satu file bahkan file tersebut berada pada server yang berbeda lokasi, namun hal itu bukanlah untuk penanganan dokumen yang sederhana seperti diatas. Hal tersebut dapat dilakukan jika dokumen yang ditulis sangatlah kompleks.

2.2 Elemen, Tag, dan Data karakter

Pada contoh diatas, dokumen hanya mengandung satu elemen nama person. Elemen tersebut diapit oleh tag awal `<person>` dan tag akhir `</person>`. Semua yang berada pada tag awal dan tag akhir disebut elemen konten dan isi dari elemen tersebut adalah string. Misalnya pada contoh diatas, isi elemennya adalah :

Alan Turing

spasi juga termasuk bagian dari elemen konten, meskipun pada kebanyakan aplikasi mengabaikan hal ini. `<person>` dan `</person>` adalah *markup*. sedangkan “Alan Turing” dan spasi disekitarnya adalah elemen konten. Tag adalah hal yang paling umum untuk *markup* didalam dokumen XML.

Sintaks Tag

Tag XML secara sekilas tampak sama dengan tag HTML. Awal tag diawali dengan “`<`” sedangkan akhir tag diawali dengan “`</`” dan keduanya diakhiri dengan “`>`” sedangkan diantaranya merupakan nama elemen. Namun tag XML tidaklah seperti tag HTML, kita diizinkan untuk membuat tag baru sesuai dengan keinginan kita yang dapat mendeskripsikan elemen data. Misalnya untuk menjelaskan elemen tersebut adalah seseorang, maka tag-nya menggunakan `<person></person>`, untuk menjelaskan sebuah alamat dapat menggunakan tag `<address></address>`. Meskipun kita dapat memberikan nama tag secara bebas, namun pada umumnya nama tag tersebut mencerminkan atau mendeskripsikan isi elemen konten sehingga akan memudahkan kita dalam penulisan dokumen.

Elemen Kosong

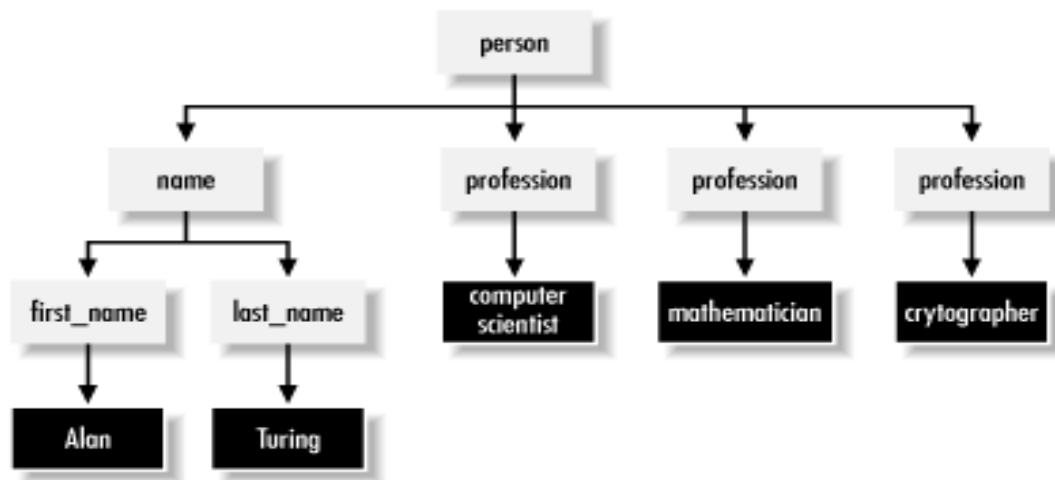
untuk elemen kosong yaitu elemen yang tidak memiliki konten, XML juga menyediakan tag khusus, dimana penulisannya dimulai dengan tag “`<`” dan diakhiri dengan “`>`”. misalnya pada HTML penulisan yang dilakukan adalah `
`, `<hr />`, `<input />` bukan dengan `
`, `<hr>`, `<input>`. Namun penulisan tersebut sama artinya dengan `
</br>`, `<hr></hr>`, `<input></input>`.

Case sensitif

Tidak seperti HTML, penulisan tag XML merupakan case sensitif. Penulisan `<Person>` tidak sama dengan `<PERSEON>` dan tidak sama pula dengan `<person>`. Jika kita membuka tag dengan `<person>`, maka kita harus menutupnya dengan `</person>`. Jika kita menutupnya menggunakan `</Person>` atau `</PERSON>` maka dokumen kita salah. Dalam penulisan tag kita bebas menggunakan huruf besar atau huruf kecil atau keduanya asalkan kita konsisten dengan apa yang kita gunakan.

Pohon XML

Mari kita coba memahami dokumen XML agak sedikit rumit. Pada contoh berikut dimana elemen *person* memiliki informasi lain yang memiliki art tersendiri :



```

<person>
  <name>
    <first_name>Alan</first_name>
    <last_name>Turing</last_name>
  </name>
  <profession>computer scientist</profession>
  <profession>mathematician</profession>
  <profession>cryptographer</profession>
</person>

```

Parents and children

Pada contoh dokumen diatas elemen name tidak hanya memiliki karakter data, tetapi elemen tersebut memiliki elemen lain. Dokumen tersebut memiliki empat *child* yaitu name, dan tiga elemen profession. Elemen name berisi dia elemen lagi yaitu first_name dan last_name.

Elemen person disebut *parent* dari elemen name dan tiga elemen profession. Elemen name merupakan *parent* dari elemen first_name dan elemen last_name. Sedangkan elemen first_name dan elemen last_name merupakan *child* dari elemen name dan elemen name dan tiga elemen profession merupakan *child* dari elemen person.

Elemen root

Elemen root merupakan suatu elemen yang tidak memiliki *parent* dan elemen ini merupakan elemen pertama. Pada contoh diatas, kita dapat merepresentasikan dokumen tersebut menjadi diagram tree seperti pada gambar dibawah ini.

Mixed Content

Maksud Mixed Content adalah dimana pada dalam elemen mengandung elemen dan data text. Misalnya contoh dibawah ini :

```

<biography>
  <name>
    <first_name>Alan</first_name>
    <last_name>Turing</last_name>
  </name>
  was one of the first people to truly deserve the
  name
  .....
</biography>

```

2.3 Atribut

Dalam penulisan dokumen XML, elemen dapat juga memiliki atribut layaknya HTML. Atribut merupakan pasangan *name-value* yang menempel pada elemen tag awal. Penulisan nama atribut dipisahkan dengan spasi dan nilainya diapit dengan petik satu atau petik dua. Sebagai contoh person memiliki atribut born dan memiliki nilai 1912-06-23, maka penulisannya :

```

<person born="1912-06-23" died="1954-06-07"> Alan Turing
</person>

```

Berikut ini penulisan yang juga benar dimana nilainya diapit dengan tanda kutip satu dan menempatkan beberapa spasi diantaranya.

```

<person died = '1954-06-07' born = '1912-06-23' > Alan Turing
</person>

```

Berikut ini adalah contoh penempatan atribut dimana dokumen yang ditempati adalah elemen kosong.

```

<person>
  <name first="Alan" last="Turing"/>
  <profession value="computer scientist"/>
  <profession value="mathematician"/>
  <profession value="cryptographer"/>
</person>

```

Sekarang yang menjadi pertanyaan kita kapa kita menggunakan atribut. Jawabannya tidak jelas penggunaan penggunaannya, namun penggunaannya tergantung pada dimana informasi tersebut akan disimpan.

2.4 XML Name

Spesifikasi XML dapat memungkinkan untuk disederhanakan yaitu dengan cara menggunakan kembali rule untuk item yang berbeda jika memungkinkan. Misalnya rule

XML untuk elemen name dan juga rule XML untuk nama atribut dan untuk beberapa nama lainnya.

Elemen dan XML name lainnya harus mengandung karakter *alphanumeric*. Ini termasuk karakter a-z, A-Z, 0-9, *_(under score)*, *-(hyphen)*.

XML name tidak boleh mengandung tanda baca selain karakter seperti tanda kutip, apostrophes, tanda-tanda dolar, carets, simbol persen, dan *semicolon*. Name XML tidak boleh juga mengandung spasi, *carriage return*, dan sebagainya. Selain itu kita juga tidak boleh menggunakan kata xml diawal name atau kombinasinya. Berikut adalah contoh penulisan yang benar :

```
<Drivers_License_Number>98 NY 32
  </Drivers_License_Number>
  <month-day-year>7/23/2001</month-day-year>
  <first_name>Alan</first_name>
  <_4-lane>I-610</_4-lane>
  <téléphone>011 33 91 55 27 55 27</téléphone>
```

Sedangkan berikut adalah contoh penulisan yang salah

```
<Driver's_License_Number>98 NY 32
</Driver's_License_Number>
<month/day/year>7/23/2001</month/day/year>
<first name>Alan</first name>
<4-lane>I-610</4-lane>
```

2.5 Entity References

Dalam penulisan karakter data kita tidak boleh menggunakan tanda “<” atau “>”. Lantas bagaimana jika kita ingin menuliskan karakter tersebut. Jalan keluarnya adalah kita dapat menggunakan entity reference **<** atau **>**. Ketika parser membaca dokumen, ia akan mengganti katakter **<** dengan string yang sebenarnya. Contohnya :

```
<SCRIPT LANGUAGE="JavaScript">
if (location.host.toLowerCase(). indexOf("cafeconleche")
    &lt; 0) {
    location.href="http://www.cafeconleche.org/";
}
</SCRIPT>
```

Didalam karakter tidak boleh mengandung ampersand (&), tetapi untuk penulisan itu dapat menggunakan entity reference **&**. Misalnya terlihat pada contoh berikut.

```
<publisher>O'Reilly &amp; Associates</publisher>
```

2.6 CDATA Sections

Pada bahasan sebelumnya kita telah membahas apa yang namanya *entity*

references dimana jika kita akan menuliskan string “<” maka kita tidak boleh menuliskannya secara langsung, namun kita harus meng-*encode* string tersebut menjadi “<”, begitu juga dengan string “>” akan di-*encode* menjadi “>”. Namun, jika kita menggunakan CData, kita dapat menuliskan string atau karakter reference secara alami, misalnya karakter “<” atau “>” dapat ditulis seperti itu juga sehingga elemen yang mengandung karakter “<” dan “>” bukan dianggap sebuah tag. Cara penulisannya adalah dengan diawali string “<![CDATA[” dan diakhiri dengan “]]>”. Contoh :

```
<p>You can use a default <code>xmlns</code> attribute to avoid having to
add the svg prefix to all your elements:</p>
<![CDATA[
<svg xmlns="http://www.w3.org/2000/svg"
    width="12cm" height="10cm">
    <ellipse rx="110" ry="130" />
    <rect x="4cm" y="1cm" width="3cm"
        height="6cm" />
</svg>
]]>
```

Pada contoh diatas yang dituliskan dalam XHTML file tanpa menggunakan karakter *reference* “<” dan “>”.

2.7 Komentar

Komentar merupakan kode atau string yang ditulis tetapi kode tersebut tidak akan dieksekusi. Dokumen XML juga mengizinkan kita untuk menuliskan komentar. Pada kebanyakan bahasa pemrograman, komentar biasanya ditulis untuk menjelaskan kode yang ditulis agar untuk kemudian hari jika kita ingin melihat kembali kode kita, kita akan dimudahkan dengan melihat komentar yang kita tulis. Penulisan komentar pada XML sama seperti pada HTML yaitu dengan diawali dengan “<!--” dan diakhiri dengan “-->”.

Contoh :

```
<!-- I need to verify and update these links when I get a chance. -->
```

Untuk penulisan double hyphen -- sebaiknya tidak diizinkan hingga sampai pada penutup komentar -->. selain itu tiga hyphen ---> juga tidak dibolehkan.

Pada komentar penulisan string apapun dibolehkan. Komentar boleh dituliskan sebelum root elemen atau setelah root elemen. Namun, tidak seperti pada HTML, komentar pada XML tidak dibolehkan untuk dituliskan didalam tag dan didalam komentar yang lain. Aplikasi yang membaca dan yang memproses dokumen XML akan melewati informasi yang terdapat didalam komentar.

2.8 Processing Instruction

Pada HTML, comment biasanya disalahgunakan untuk mendukung ekstensi yang tidak standart (*nonstandart extensions*). Contohnya, konten-konten dari elemen *script* biasanya dilampirkan pada sebuah comment untuk menjaganya dari penampakan dengan nonscript-aware browser. Apache web server memparser comment-comment dalam bentuk file .shtml untuk mengenal bagian yang mengandung server. Sayangnya, dokumen-dokumen ini tidak mampu bertahan dalam melewati berbagai macam editor HTML (HTML editor) dan *prosesor* dengan komentar dan *associated semantics intact*. Yang lebih buruk, dimungkinkan untuk sebuah comment menjadi salah arti, sehingga menjadi sebuah input pada aplikasi.

Oleh karenanya XML memberikan instruksi pemrosesan sebagai alternatif parsing informasi tertentu melalui aplikasi yang dapat membaca dokumen. *Processing Instruction* Merupakan perintah pengolahan dalam dokumen XML. Node ini ditandai awali dengan karakter `<?` Dan diakhiri dengan `?>`. `<?` merupakan XML name yang disebut target, mungkin nama aplikasi untuk *processing instruction* ini yang dimaksudkan atau mungkin hanya sebuah identifier untuk *processing instruction* secara khusus. Sisa pemrosesan instruksi mengandung teks dalam format yang sesuai dengan aplikasi yang akan digunakan. Tapi perlu diingat bahwa header standard XML `<?xml version="1.0" encoding="iso-8859-1"?>` bukanlah *processing instruction*. Header standard bukanlah bagian dari hirarki pohon dokumen XML.

Sebagai contoh, didalam HTML sebuah robots META tag digunakan untuk memberitahukan kepada *search-engine* dan pencarian lainnya dan bagaimana mereka harus meng-*index* sebuah halaman web :

```
<?robots index="yes" follow="no"?>
```

Target pemrosesan instruksi ini adalah robots. Sintaks dari instruksi ini adalah proses dua atribut palsu, satu nama dan satu nama indeks, yang nilai-nilainya *yes* atau *no*. Semantik dari pemrosesan instruksi ini adalah jika indeks atribut memiliki nilai *yes*, kemudian *search-engine robots* harus mengiindek halaman tersebut. Jika indeks memiliki nilai *no*, maka halaman tersebut tidak akan di- *index*. sebaliknya, jika memiliki nilai *yes*, maka link dari dokumen ini akan diikuti.

Pemrosesan instruksi lainnya mungkin saja dapat berbeda antara semantik dan sintaksisnya. Contohnya, pemrosesan instruksi dapat mengandung jumlah text yang

unlimited. PHP include program yang besar dalam pemrosesan instruksi. Sebagai contoh :

```
<?php
mysql_connect("database", "clerk", "password");
$result = mysql("HR", "SELECT LastName, FirstName
    FROM Employees ORDER BY LastName, FirstName");
$i = 0;
while ($i < mysql_numrows ($result)) {
    $fields = mysql_fetch_row($result);
    echo "<person>$fields[1] $fields[0] </person>
        \r\n";
    $i++;
}
mysql_close( );
?>
```

Pemrosesan instruksi adalah markup tampak seperti elemen, tetapi bukan elemen. Instruksi pemrosesan dapat muncul di manapun dalam dokumen XML di luar tag, termasuk sebelum atau sesudah elemen *root*. Yang paling umum pemrosesan instruksi, *xml-stylesheet*, digunakan untuk melampirkan *stylesheets* ke dokumen. *Stylesheets* selalu muncul sebelum elemen *root*, pada contoh berikut, pemrosesan instruksi *xml-stylesheet* memberitahu browser untuk menerapkan CSS stylesheet *person.css* ke dokumen ini sebelum tampil

```
<?xml-stylesheet href="person.css" type="text/css"?>
<person>
    Alan Turing
</person>
```

2.9 Deklarasi XML

Sebuah dokumen XML boleh dideklarasikan boleh juga tidak. Pendeklarasian XML mengandung *name* dan *version*, *standalone*, dan *encoding* atribut. Sebagai Contoh :

```
<?xml version="1.0" encoding="ASCII" standalone="yes"?>
<person>
    Alan Turing
</person>
```

pendeklarasian tidak perlu ditulis pada dokumen XML. Namun, jika pada dokumen ada pendeklarasian maka deklarasi harus berada paling atas, tidak boleh didahului sintaks apapun seperti komentar, spasi, dll. Penjelasan atau arti dari bagiannya :

1. Version

Merupakan versi penulisan dokumen XML yang digunakan

2. Encoding

Ini merupakan type encoding dari dokumen XML tersebut, misalnya

UTF-8, UTF-16, ISO-8859-1, ASCII, dll

3. Standalone

Apakah dokumen ini berdiri sendiri atau merupakan penggabungan dari dokumen lain.

2.10 Checking Documents for Well-Formedness

Setiap dokumen XML harus well-formed. Ini berarti harus sesuai dengan aturan yang ada misalnya :

1. Setiap awal tag harus diakhiri dengan tag yang sama.
2. Elemen boleh bersarang, tetapi tidak boleh saling tumpang tindih
3. Harus memiliki tepat satu elemen root.
4. Nilai atribut harus diapit oleh tanda petik.
5. Satu elemen tidak boleh memiliki dua atribut yang sama
6. komentar pemrosesan intruksi tidak muncul didalam tag
7. Tidak ada unescaped "<" atau "&" atau tanda-tanda lainnya

2.11 Pembuatan File XML

Instalasi

Beberapa aplikasi dan tool yang dibutuhkan untuk implementasi antara lain :

1. XAMPP (Web Server untuk Windows), isinya Apache, PHP dan MYSQL.

Silakan didownload disini: <http://www.apachefriends.org/en/xampp.html>

2. Notepad++, text editor untuk menuliskan bahasa pemrograman. Silakan didownload disini: <http://notepad-plus-plus.org/download>

1. Instalasi XAMPP

XAMPP merupakan paket aplikasi yang terdiri dari Web Server Apache, database MySQL dan PHP. Dengan adanya XAMPP kita tidak perlu menginstal satu persatu aplikasi-aplikasi tersebut. Cukup dengan mengeksekusi file instalernya maka ketiga aplikasi tadi terinstal. Berikut langkah-langkah instalasinya:

1. Jalankan file xampp-win32-1.7.7-installer.exe (atau versi lainnya)
2. Kemudian akan tampil pilihan untuk memilih bahasa ketika proses instalasi berjalan. Silakan pilih bahasa Indonesian atau English. (Misal kita memilih bahasa Indonesia)
3. Klik Maju untuk memulainya.
4. Klik Saya Setuju untuk melanjutkan.

5. Selanjutnya silakan anda pilih lokasi install untuk XAMPP. Kemudian klik install
6. Tunggu beberapa saat sampai proses instalasi selesai.
7. Instalasi selesai
8. Jalankan XAMPP Control Panel yang ada di Desktop. Atau anda juga dapat menjalankan XAMPP Control Panel dari menu Start.
9. Nyalakan Apache dan Mysql dengan mengklik tombol Start. Buka web browser anda, lalu ketikkan `http://localhost`. Jika tampilannya seperti di bawah ini, maka apache sudah terinstall dengan benar.

2. Instalasi Notepad ++

1. Siapkan software notepad++ nya.
2. Tampilan pertama saat ingin menginstal Notepad++ adalah memilih bahasa, pilih English lalu ok.
3. Selanjutnya klik next saja
4. Selanjutnya adalah agreement, klik I agree
5. Selanjutnya adalah menempatkan instalasi dari notepad, defaultnya terletak di `c:\program files\Notepad++`. Lokasi penempatan dapat disesuaikan dengan keinginan
6. Lalu memilih komponen yang akan di install, pilih default saja lalu klik ok
7. Selanjutnya klik install saja jika ingin langsung menginstall, terdapat pilihan untuk membuat shortcut pada desktop juga
8. Jika instalasi telah selesai, maka notepad++ sudah siap dijalankan

3. Konfigurasi Curl di XAMPP

Apa itu CURL? CURL adalah singkatan dari Client URL yang berfungsi sebagai alat bantu transfer file dengan sintaks URL melalui bermacam-macam protokol (FTP, SFTP, HTTP, HTTPS, SCP, TELNET, LDAP, dsb). Sedangkan libcurl adalah library portable yang menyediakan interface (untuk berbagai bahasa pemrograman, seperti Perl, Python, PHP, dsb) terhadap fungsionalitas CURL.

Salah satu manfaat CURL adalah untuk mengambil sebagian atau seluruh isi dari suatu website atau blog untuk ditampilkan di website milik kita. Tujuannya mungkin untuk menambah fungsionalitas dari website Anda sendiri. Atau mungkin agar website Anda terlihat lebih menarik di mata pengunjung. Untuk penggunaannya, Anda harus menguasai salah satu dari beberapa bahasa pemrograman web misalkan PHP.

Berikut langkah-langkah mengkonfigurasi curl di xampp:

1. Lakukan konfigurasi dengan mengedit file `php.ini` yang ada di dalam direktori

PHP. (C:\xampp\php\php.ini)

2. Buka file php.ini, lalu cari baris perintah: **extension=php_curl.dll**
3. Kemudian hilangkan tanda titik koma (;) didepan baris tersebut. Hasilnya menjadi seperti ini: **extension=php_curl.dll**
4. Simpan perubahan, lalu restart Apache Web Servernya.
5. Ketikkan "http://localhost" pada web browser. Kemudian cek php_info().
Jika anda berhasil maka anda akan menemukan CURL telah aktif pada server localhost anda. CURL pada Localhost Windows seperti gambar di bawah

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 mendefinisikan dari awal tag dan atribut yang dipakai didalamnya, sedangkan pada XML kita bisa menggunakan tag dan atribut sesuai kehendak kita. Untuk lebih jelasnya lihat contoh dibawah:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<data_mahasiswa>
  <mhs category="KOMSI">
    <nama>Fajar Barrun</nama>
    <nim>03672</nim>
    <alamat>cilacap</alamat>
  </mhs>
  <mhs category="ELINS">
    <nama>Farisa</nama>
    <nim>03628</nim>
    <alamat>Madiun</alamat>
  </mhs>
  <mhs category="REKMED">
    <nama>Redha</nama>
    <nim>03693</nim>
    <alamat>Malang</alamat>
  </mhs>
</data_mahasiswa>
```

Simpan dengan nama latihan1.xml

Pada contoh diatas <data_mahasiswa>, <mhs> <nama>,<nim>, dan <alamat> 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. Jika file XML tadi dibuka di browser dan benar, maka tampilannya seperti berikut

This XML file does not appear to have any

```
▼<data_mahasiswa>
  ▼<mhs category="KOMSI">
    <nama>Fajar Barrun</nama>
    <nim>03672</nim>
    <alamat>cilacap</alamat>
  </mhs>
  ▼<mhs category="ELINS">
    <nama>Farisa</nama>
    <nim>03628</nim>
    <alamat>Madiun</alamat>
  </mhs>
  ▼<mhs category="REKMED">
    <nama>Redha</nama>
    <nim>03693</nim>
    <alamat>Malang</alamat>
  </mhs>
</data_mahasiswa>
```

BAB III

DOCUMENT TYPE DEFINITIONS (DTD)

Tujuan :

- Mahasiswa memahami apa yang dimaksud dengan DTD
- Mahasiswa dapat menggunakan DTD untuk elemen dan atribut.
- Mahasiswa dapat membuat dokumen XML berdasarkan DTD yang sudah ada.

Meskipun XML sangatlah fleksibel, namun tidak semua program dapat membaca dokumennya secara baik. Banyak program yang dapat bekerja dengan beberapa aplikasi XML tetapi terkadang tidak untuk yang lain. Aplikasi XML harus memastikan bahwa dokumen yang diberikan adalah benar-banar mematuhi aturan. Misalnya didalam XHTML, pada elemen li merupakan child dari elemen ol. Dalam hal ini, apakah li atau ul tersebut merupakan bagian dari tag XML atautkah elemen isi. Solusi untuk permasalahan seperti itu adalah dengan menggunakan DTD. DTD berfungsi untuk mendefinisikan tipe dokumen XML. DTD ditulis untuk menjelaskan elemen dan entitas yang mungkin muncul di dalam dokumen dan elemen isi serta atributnya. Sehingga kita tahu bahwa seperti apa struktur dokumen dan dapat membedakan yang mana tag dan yang mana elemen.

3.1 Validation

Dokumen yang valid termasuk jenis dokumen mendeklarasikan DTD. DTD yang mengandung semua elemen, atribut, dan entitas yang akan digunakan didalam dokumen. Validitas operasinya pada prinsipnya bahwa semua yang tidak diperbolehkan adalah dilarang. Semua dokumen yang akan ditulis harus dideklarasikan didalam DTD. Jika pada suatu dokumen terdapat deklarasi DTD, maka mau tidak mau isi dari dokumen yang akan ditulis harus sesuai dengan DTDnya. Jika dokumen yang ditulis tersebut telah sesuai, maka dokumen tersebut dapat dikatakan valid, jika tidak maka dokumen tersebut tidak valid.

Selain itu, banyak hal-hal yang tidak diulas DTD, diantaranya :

1. Apa yang menjadi root element dari dokumen.
2. Berapa kasus dari tiap-tiap elemen yang muncul di dalam dokumen.
3. Karakter data apa yang tampak didalam elemen.

4. Semanting yang berarti dari sebuah elemen, apakah mengandung data tanggal atau nama orang.

DTD memungkinkan untuk penempatan beberapa kostrain pada form sebuah dokumen XML. Parser membaca apakah dokumen tersebut valid atau tidak. Jika dokumen tersebut valid, maka program akan menerima data dari parser. Dalam beberapa kasus, seperti memasukan *record* kedalam database, validation error mungkin cukup serius, sehingga field yang dibutuhkan hilang. Contoh sederhana penggunaan DTD :

```
<!DOCTYPE person [  
  <!ELEMENT person (name, profession*)>  
  <!ELEMENT name (first_name, last_name)>  
  <!ELEMENT first_name (#PCDATA)>  
  <!ELEMENT last_name (#PCDATA)>  
  <!ELEMENT profession (#PCDATA)>  
<person>  
  <name>  
    <first_name>Alan</first_name>  
    <last_name>Turing</last_name>  
  </name>  
  <profession>mathematician</profession>  
  <profession>cryptographer</profession>
```

Pada contoh diatas maksudnya adalah person menjadi elemen root dimana didalam elmen person harus ditulis satu kali element name dan elemen profession boleh muncul atau boleh tidak. Sedangkan pada elemen name harus memiliki dua elemen *child* yaitu first_name, last_name dan kemunculannya msing-masing satu kali. Sedangkan (#PCDATA)menunjukan tipe isi dari elemen yaitu data karakter.

Contoh:

```
<!DOCTYPE person [  
  <!ELEMENT person (student+)>  
  <!ELEMENT student (name, hobbi | favorite)>  
  <!ELEMENT name (#PCDATA)>  
  <!ELEMENT hobbi (#PCDATA)>  
  <!ELEMENT favorite (#PCDATA)>  
<person>  
  <student>  
    <name>Alan Turing</name>  
    <hobbi>Tenis</hobbi>  
  </student>  
  <student>  
    <name>Johan</name>  
    <favorite>Arktis</favorite>  
  </student>  
  <student>  
    <name>Marie</name>  
    <hobbi>Tenis</hobbi>  
    <favorite>Arktis</favorite>  
  </student>  
</person>
```

Pada contoh diatas, terdapat tanda “ | ” diantara hobbi dan faforite. Maksudnya adalah didalam elemen studentwajib mengandung satu elemen nama dan boleh salah satu dari elemen hobbiatau faforiteyang muncul atau keduanya muncul.

Selain itu, dokumen yang valid juga memasukan sebuah reference kedalam DTD yang harus dibandingkan disajikan dalam dokumen dari satu jenis dokumen deklarasi. Ini berguna jika kita menuliskan DTDnya didalam file yang terpisah dengan dokumen. Contoh deklarasi dokument type adalah :

```
<!DOCTYPE person SYSTEM
"http://www.politekniktelkom.ac.id/person.dtd">
<person>
    <name>
        <first_name>Alan</first_name>
        <last_name>Turing</last_name>
    </name>
    <profession>mathematician</profession>
    <profession>cryptographer</profession>
</person>
```

sintaks tersebut dituliskan didalam dokumen XML dan DTD-nya berada diluar serve.

3.2 Element Declarations

Setiap elemen yang digunakan dalam dokumen yang valid harus dinyatakan dalam dokumen DTD dengan elemen deklarasi. Deklarasi elemen memiliki bentuk dasar :

```
<!ELEMENT element_name content_specification>
```

adapun beberapa hal yang merupakan bagian dari elemen declaration adalah :

- **#PCDATA**

Konten yang terkandung didalam elemen yang dimaksud adalah text. Meskipun data yang ada dielemen tersebut berupa angka, tetap dianggap sebagai text.

Contohnya :

```
<!ELEMENT phone_number (#PCDATA)>
```

- **Elemen anak**

Penspesifikasi konten sederhana adalah salah satu elemen yang harus memiliki satu anak dari type yang sesuai. Pada kasus ini, nama anak ditulis didalam kurung. Contohnya :

```
<!ELEMENT fax (phone_number)>
```

Maksudnya adalah elemen phone_number merupakan anak dari elemen fax.

- **Sequence**

Kita dapat menuliskan elemen anak lebih dari satu pada deklarasinya.

Dengan penulisan ini dimaksudkan untuk menuliskan elemen anak lebih dari satu elemen. Misalnya :

```
<!ELEMENT name (first_name, last_name)>
```

- Jumlah anak

Penulisan jumlah anak dari suatu elemen dapat kita tuliskan dengan cara menambahkan tanda berikut pada akhir nama elemen.

Karakter	Arti
+	Muncul saatu kali atau lebih. Minimal muncul
*	Muncul 0 kali atau lebih
?	Boleh tidak muncul, tapi jika muncul maksimal
	Fungsi atau

Contoh penggunaannya adalah sebagai berikut :

```
<!ELEMENT people (phone_number*)>
```

artinya elemen people boleh memiliki satu atau lebih elemen phone_number atau bisa juga kosong.

- Pilihan

Terkadang pada kasus tertentu ada suatu elemen yang memiliki anak tetapi berbeda-beda. Cara pendeklarasiannya dapat dilakukan dengan :

```
<!ELEMENT methodResponse (params | fault)>
```

penulisannya tidak hanya dibatasi dua elemen saja, tetapi kita dapat menuliskan lebih dari itu.

- Parentheses

Dari yang telah kita bahas choices, sequences, dan suffixes sangat terbatas penggunaannya, oleh karena itu kita dapat pula menggabungkannya dari masing-masing bagian tadi. Misalnya :

```
<!ELEMENT circle (center, (radius | diameter))>
```

Atau

```
<!ELEMENT center ((x, y) | (y, x) | (r, θ) | (θ, r) )>
```

- Mixed Content

Dimana kita memungkinkan untuk menuliskan text dan elemen dalam suatu elemen. Contoh :

```
<definition>
```

```
The<term>Turing Machine</term> is an abstract finite state automaton  
with infinite memory that can be proven equivalent to any any other  
finite state automaton with arbitrarily large memory. Thus what is
```

```
true for a Turing machine is true for all equivalent machines no  
matter how implemented.  
</definition>
```

dimana deklarasi elemen adalah :

```
<!ELEMENT definition (#PCDATA | term)*>
```

- **Elemen Kosong**

Elemen kosong adalah sebuah tag yang tidak memiliki elemen nilai. Penulisannya dapat berupa :

```
<image source="bus.jpg" width="152" height="345" alt="Alan Turing  
standing in front of bus"/>
```

dengan deklarasi elemennya :

```
<!ELEMENT image EMPTY>
```

- **ANY**

Kita dapat mengizinkan apapun dapat berada pada suatu elemen.

Penulisan deklarasinya adalah :

```
<!ELEMENT page ANY>
```

3.3 Deklarasi Atribut

Seperti halnya elemen, dokumen dikatakan valid jika dapat menyatakan semua atribut elemen jika elemennya memiliki atribut. Yaitu sintaks ATTLIST dapat mendeklarasikan beberapa atribut dalam suatu elemen. Contoh berikut merupakan pendeklarasian ATTLIST pada elemen image, dimana elemen image memiliki atribut source :

```
<!ATTLIST image source CDATA #REQUIRED>
```

atau kita dapat menuliskan lebih dari satu atribut didalam suatu elemen :

```
<!ATTLIST image source CDATA #REQUIRED width CDATA  
#REQUIRED height CDATA #REQUIRED alt CDATA #IMPLIED>
```

maksud dari deklarasi diatas adalah elemen image memiliki atribut source, width, height dan alt. Dengan type CDATA.

Type Atribut

Type atribut yang ada didalam XML adalah :

- **CDATA**

Atribut yang hanya memiliki nilai text string.

- **NMTOKEN**

Type atribut yang mengandung token XML name yang dipisahkan dengan spasi

```
<performances dates="08-21-2001"> Kat and the Kings  
</performances>
```

- **NMTOKENS**

Type atribut yang mengandung satu atau lebih token XML name yang dipisahkan dengan spasi

```
<performances dates="08-21-2001 08-23-2001 08-  
27-2001"> Kat and the Kings </performances>
```

- **Enumeration**

Atribut yang bukan merupakan keyword XML.

```
<!ATTLIST date month (January | February | March  
| April | May | June | July | August |  
September | October | November | December)  
#REQUIRED >
```

- **ID**

Sebuah type atribut ID harus berisi XML name yang unik didalam dokumen.

- **IDREF**

Mengacu pada type atribut ID dari beberapa elemen.

- **IDREFS**

Mengandung spasi sebagai pemisah dan masing-masing harus mengacu pada ID dari elemen didalam dokumen itu sendiri.

- **ENTITY**

Berisi name dari sebuah deklarasi entity unparsed di DTD

- **ENTITIES**

Berisi name dari satu atau lebih deklarasi entity unparsed di DTD

- **NOTATION**

Mengandung deklarasi notasi didalam DTD

Atribut Default

- **#IMPLIED**

Tidak disediakan nilai default, jika kita mengisi nilai tersebut, maka nilai atribut elemen menjadi sesuai yang dituliskan, jika kita tidak memberikan nilai atributnya, maka nilai atributnya menjadi kosong.

- **#REQUIRED**

Merupakan default atribut. Artinya tidak disediakan nilai default untuk atribut, jadi nilainya harus diisi, jika tidak maka parser akan menampilkan pesan *error*.

- **#FIXED**

Disediakan nilai default dan jika kita mengisikan nilai atribut pada elemen, maka nilainya akan samadengan yang diisikan, sedangkan jika dokosongkan maka parser akan mengenali nilai atributnya sesuai dengan nilai default yang diberikan.

3.4 *General Entity Declarations*

Referensi entitas didefinisikan dengan pendeklarasian sebuah ENTITY didalam DTD. Hal ini diberikan guna memberikan nama pada entiti. Dengan kata lain, kita bisa menggantikan kalimat yang panjang atau satu blok elemen yang sering kita gunakan dengan sebuah pengenal singkat. Dengan adanya pendeklarasian entiti yang sesuai, kita dapat menuliskan text atau string apapun hanya dengan menuliskan entity referencenya saja yang telah dideklarasikan tanpa harus menulis text tersebut secara utuh. Misalnya deklarasi entity mendefinisikan **&email;** sebagai singkatan dari “admin@somethingweb.com”, maka contoh penulisannya :

```
<!ENTITY email "admin@somethingweb.com">
<person>
  <name>Peter Kok kok</name>
  <email>&email;</email>
</person>
```

3.5 *External Parsed General Entities*

Pada umumnya situs-situs web yang beredar lebih memilih menyimpan konten yang akan tampil berulang kedalam file external yang kemudian akan ditampilkan kehalaman webnya menggunakan PHP atau bahasa pemrograman web, termasuk server-site atau mekanisme yang lainnya. Dalam hal ini, XML mendukung teknik tersebut melalui *external general entity references* meskipun dalam hal ini client bukanlah sebagai server. *External general entity references* dideklarasikan didalam DTD yang menggunakan pendeklarasian ENTITY. Contohnya:

```
<!ENTITY footer SYSTEM
"http://www.oreilly.com/boilerplate/footer.xml">
```

atau kita juga dapat memasukan relative URL. Misalnya :

```
<!ENTITY footer SYSTEM "/boilerplate/footer.xml">
```

3.6 External Unparsed Entities and Notations

Tidak semua data itu XML. Terdapat banyak file ASCII didunia ini yang tidak memberikan arti tentang "<" sebagai < atau kendala lainnya yang pada dokumen XML yang terbatas. Pada dokumen JPEG photographs, GIF line art, QuickTime movies, MIDI sound files, dan sebagainya yang merupakan komponen penting dari berbagai macam komponen namun tidak dapat diimplementasikan menjadi dokumen XML. Namun mekanisme embeded XML menyarankan untuk menggunakan External Unparsed Entities yang dapat didefinisikan kedalam DTD sebagai entitas yang bukan dokumen XML. Misalnya pendeklarasian ENTITY yang mengasiasikan nama turing_getting_off_bus dengan gambar JPEG pada <http://www.turing.org.uk/turing/pi1/bus.jpg> :

```
<!ENTITY turing_getting_off_bus SYSTEM
    "http://www.turing.org.uk/turing/pi1/bus.jpg" NDATA jpeg>
```

- **Notifikasi**

Ini digunakan agar dokumen XML dapat mengenali file yang akan dimasukan, misalnya dengan memberikan penamaan jpeg pada file JPEG image. Contoh :

```
<!NOTATION jpeg SYSTEM "image/jpeg">
```

- **Meletakkan *Unparsed Entities* didalam dokumen**

Namun kita tidak bisa menempatkan entity reference, entitas reference hanya dapat merujuk ke parsed entities. Misalnya elemen image dan deklarasi atribut seperti :

```
<!ELEMENT image EMPTY>
<!ATTLIST image source ENTITY #REQUIRED>
```

kemudian elemen gambar ini akan merujuk pada
<http://www.turing.org.uk/turing/pi1/bus.jpg>
<image source="turing_getting_off_bus"/>

- **Notifikasi untuk target pemrosesan instruksi**

Notasi dapat juga digunakan untuk mengidentifikasi target yang tepat dari instruksi proses. Notasi dapat mengidentifikasi nama singkat XML dengan spesifikasi lebih lengkap. Misalnya targer path :

```
<!NOTATION tex SYSTEM "/usr/local/bin/tex">
```

3.7 Parameter Entities

tidak dapat digunakan pada multiple elemen untuk men-share semua atau sebagian untuk atribut yang sama. Misalnya adda elemen simpel XLink yang dijadikan xlink:type and atribut xlink:href, dan mungkin xlink:show and atribut xlink:actuate.

Misalnya, aplikasi XML untuk perumahan daftar *real-estate* yang menyediakan elemen terpisah untuk apartemen, sublets, coops untuk penjualan, Condos untuk dijual, dan rumah untuk dijual. Elemen deklarasi mungkin terlihat seperti ini:

```
<!ELEMENT apartment (address, rooms, baths, rent)>
<!ELEMENT sublet (address, rooms, baths, rent)>
<!ELEMENT coop (address, rooms, baths, price)>
<!ELEMENT condo (address, rooms, baths, price)>
<!ELEMENT house (address, rooms, baths, price)>
```

- **Parameter Entity Syntax**

parameter entity reference banyak dinyatakan pada general entitas reference. Namun, tanda persen (%) ditempatkan diantara <!ENTITY dan nama entitas.

```
<!ENTITY % residential_content "address, footage, rooms, baths">
<!ENTITY % rental_content "rent">
<!ENTITY % purchase_content "price">
```

Parameter entity yang ditulis kembali sebagai general reference, hanya dengan tanda persen.

```
<!ELEMENT apartment (%residential_content;,
    %rental_content;)>
<!ELEMENT sublet (%residential_content;,
    %rental_content;)>
<!ELEMENT coop (%residential_content;,
    %purchase_content;)>
<!ELEMENT condo (%residential_content;,
    %purchase_content;)>
<!ELEMENT house (%residential_content;,
    %purchase_content;)>
```

- **Redefining Parameter Entities**

Jika dokumen menggunakan *both* internal maupun eksternal DTD subset, maka subset internal DTD dapat menentukan teks untuk entity. Jika ELEMENT dan ATTLIST dideklarasikan di luar subset DTD ditulis langsung dengan parameter entitas reference.

```
<!ENTITY % residential_content "address, footage, rooms, bedrooms,
baths, available_date">
```

- **External DTD Subsets**

Entitas parameter external dideklarasikan menggunakan normal ENTITY dengan tanda %. Contoh, deklarasi ENTITY mendefinisikan sebuah konten eksternal entitas yang disebut *names* yang kontennya diambil dari file *names.dtd*. Kemudian entitas parameter referensi %name; memasukkan isi file ke dalam DTD.

```
<!ENTITY % names SYSTEM "names.dtd"> %names;
```

3.8 Conditional Inclusion

XML memberikan IGNORE direktif untuk tujuan "*commenting out*" sebuah selection deklarasi. Misalnya parser akan mengabaikan deklarasi dari elemen *production_note*:

```
<![ IGNORE[<!ELEMENT
production_note
(PCDATA)>
]]>
```

Sintaks diatas sepertinya tidak berguna, karena kita bisa saja menggunakan komentar. Namun tujuan ini adalah untuk menunjukkan bahwa deklarasi yang diberikan benar-benar digunakan didalam DTD. Contoh :

```
<![ INCLUDE[
<!ELEMENT production_note (PCDATA)>
]]>
```

3.9 Dua contoh DTD

- **Data-Oriented DTDs**

Membuat penggunaan sequences menjadi lebih berat. Kadang penggunaan pilihan dan hampir tidak ada penggunaan *mixed content*.

- **Narrative-Oriented DTDs**

Cenderung bebas dan membuat lebih banyak menggunakan mixed conten menjadi DTD yang mendeskripsikan dokumen database-like lainnya.

3.10 Menempatkan Standar DTD

DTD dan validity merupakan hal yang sangat penting ketika kita hendak mengganti data dengan yang lainnya. Hal ini untuk memastikan kita apakah data yang kita kirim diterima dengan baik

BAB IV

XSL TRANSFORMATIONS (XSLT)

1. Mahasiswa dapat memahami XSLT
2. Mahasiswa dapat menspesifikasikan XSLT pada dokumen XML
3. Mahasiswa dapat membuat XSLT pada dokumen XML
4. Mahasiswa bisa membedakan “perintah” XSLT dengan Xpath

4.1 Pengenalan XSLT

XSLT merupakan aplikasi XML untuk menspesifikasikan aturan antara dokumen XML yang satu ditransformasikan menjadi dokumen XML lainnya. Dokumen XSLT yaitu, sebuah XSLT stylesheet berisi template-rule. Setiap template memiliki aturan dan pola tersendiri. Sebuah XSL processor akan membaca dokumen XML dan template (dokumen XSLT). Berdasarkan instruksi yang ditemukan program pada XSLT, maka program akan menghasilkan dokumen XML yang baru.

4.2 An Example Input Document

```
<?xml version="1.0"?>
<people>
  <person born="1912" died="1954">
    <name>
      <first_name>Alan</first_name>
      <last_name>Turing</last_name>
    </name>
    <profession>computer scientist</profession>
    <profession>mathematician</profession>
    <profession>cryptographer</profession>
  </person>
  <person born="1918" died="1988">
    <name>
      <first_name>Richard</first_name>
      <middle_initial>P</middle_initial>
      <last_name>Feynman</last_name>
    </name>
    <profession>physicist</profession>
    <hobby>Playing the bongoes</hobby>
  </person>
</people>
```

Dokumen diatas disimpan kedalam file dengan nama people.xml. XSLT dapat bekerja dengan dokumen yang valid maupun tidak valid asalkan *well-formed*. Dokumen ini juga tidak menggunakan namespace meskipun bisa juga dengan menggunakan *namespace*.

XSLT hanya dapat berjalan dengan baik jika menggunakan *namespace*. Tidak seperti DTD, XSLT lebih memperhatikan namespace URIs dari pada prefiks.

4.3 *xsl:stylesheet* dan *xsl:transform*

XSLT stylesheet merupakan dokumen XML, dan umumnya harus menggunakan deklarasi XML atau paling tidak stylesheets. Root elemen dokumen ini adalah stylesheet atau transform. Selain itu kita juga dapat menggunakan stylesheet atau transform. Contoh minimal dokumen XSLT :

```
<?xml version="1.0"?>
<xsl:stylesheet          version="1.0"          xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

4.4 *Stylesheet Processors*

XSLT processor sebuah perangkat lunak yang membaca XSLT stylesheet, membaca dokumen input XML, dan membuat dokumen output sesuai dengan petunjuk dalam stylesheet menjadi informasi didalam dokumen masukan. XSLT processor dapat diterapkan pada web browser, seperti MSXML pada Internet Explorer 6.

The xml-stylesheet Processing Instruction

XML dokumen yang ditampilkan ke web browser memiliki *xml-stylesheet processing instruction* pada prolog yang memberitahukan browser dimana lokasi stylesheet pada dokumen tersebut. Jika stylesheet-nya merupakan XSLT stylesheet, maka tipe pseudoattribute harus memiliki nilai `application/xml`. Contoh, pemrosesan instruksi `xml-stylesheet` menunjukan browsers harus memakai stylesheet pada URL absolut `http://www.oreilly.com/styles/people.xsl`. Namun, Relativ URLs juga bisa digunakan.

```
<?xml version="1.0"?>
<?xml-stylesheet type="application/xml"
      href="http://www.oreilly.com/styles/people.xsl"?>
<people>
...
</people>
```

4.5 *Templates and Template Rules*

Untuk mengontrol hasil outputan, kita dapat menambahkan *template rule* kedalam XSLT stylesheet. Setiap template direpresentasikan dengan elemen `xsl:template`. Elemen

ini harus memiliki kesesuaian atribut dengan XPath *pattern* dan mengandung template yang di instansiasi dan output jika polanya cocok. Namun ada sedikit terminologi trik; elemen `xsl:template` merupakan *template rule* yang mengandung template. Elemen `xsl:template` bukan template untuk dirinya sendiri.

Pencocokan pola secara sederhana adalah pada elemen `name`. Dengan demikian, template rule ini menunjukkan elemen peson. stylesheet processor harus mengeluarkan text “A person”. Contoh Templatenya :

```
<xsl:template match="person">A Person</xsl:template>
```

Template rule :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl= "http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="person">A Person</xsl:template>
</xsl:stylesheet>
```

Output proses :

```
<?xml version="1.0" encoding="utf-8"?> A Person
      A Person
```

4.6 Calculating the Value of an Element with *xsl:value-of*

Elemen XSLT lainnya dapat memilih konten tertentu dari dokumen input dan memasukkannya ke dalam dokumen output. Elemen `xsl:value-of` akan mengolah nilai string dari ekspresi XPath dan memasukannya kedalam output. Nilai dari sebuah elemen merupakan text elemen setelah semua tag dihapus serta entitas dan karakter *reference* telah berubah. Elemen yang nilainya diambil diidentifikasi dengan atribut `select` berisi ekspresi XPath. Contoh :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
  "http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="person">
    <p>
      <xsl:value-of select="name"/>
    </p>
  </xsl:template>
</xsl:stylesheet>
```

Ketika XLS diproses, maka output dari hasil tersebut adalah :

```
<?xml version="1.0" encoding="utf-8"?>
  <p>Alan Turing</p>
  <p>Richard P Feynman</p>
```

4.7 Applying Templates with *xsl:apply-templates*

xsl:apply-templates dapat meng-eksplisitkan pilihan pemrosesan. *xsl:apply-templates* digunakan untuk membuat XLS processor untuk mengaplikasikan template yang digunakan.

Misalnya kita ingin menampilkan list people pada input dokumen, namun pada output dokumen kita akan menampilkan nama belakang ditulis didepan. Penulisan XSL nya adalah :

```
<xsl:template match="name">
  <xsl:value-of select="last_name"/>,
  <xsl:value-of select="first_name"/>
</xsl:template>
```

atau kita juga dapat menggunakan :

```
<xsl:template match="person">
  <xsl:apply-templates select="name"/>
</xsl:template>
```

4.8 The Built-in Template Rules

Terdapat tujuh jenis node dalam sebuah dokumen XML yaitu : *root node*, *element nodes*, *attribute nodes*, *text nodes*, *comment nodes*, *processing instruction nodes*, and *namespace nodes*. XSLT menyediakan default template built-in untuk ketujuh node tersebut. Namun template tersebut akan aktif jika :

The Default Template Rule for Text and Attribute Nodes

Yang paling mendasar dari *built-in template rule* adalah menyalin nilai text dan atribut node kedalam dokumen output. Seperti terlihat pada contoh berikut :

```
<xsl:template match="text( )|@*">
  <xsl:value-of select="."/>
</xsl:template>
```


node text () merupakan pattern matching untuk semua node. Misalnya first_name merupakan pattern matching untuk semua elemen node first_name. Sedangkan @* merupakan pattern matching untuk semua node atribut.

The Default Template Rule for Element and Root Nodes

Yang paling penting adalah bagaimana menjamin bahwa elemen child juga akan diproses. Aturan penulisannya adalah :

```
<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

The Default Template Rule for Comment and Processing Instruction Nodes

```
<xsl:template match="processing- instruction()|comment( )"/>
```

rule ini cocok untuk semua command dan pemrosesan instruksi.

4.9 Modes

Elemen xsl:apply-templates dan xsl:template dapat memiliki pilihan atribut mode yang menghubungkan template rule yang berbeda ke penggunaan yang berbeda. Atribut mode pada elemen xsl:template mengidentifikasi pada mode yang template rule-nya harus aktif. Elemen xsl:apply-templates dengan atribut mode hanya berjalan dengan template rule dengan atribut mode yang cocok. Contoh :

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="people">
    <html>
      <head><title>Famous Scientists</title></head>
      <body>
        <ul><xsl:apply-templates select="person" mode="toc"/></ul>
        <xsl:apply-templates select="person"/>
      </body>
    </html>
  </xsl:template>
  <!-- Table of Contents Mode Templates -->
  <xsl:template match="person" mode="toc">
    <xsl:apply-templates select="name" mode="toc"/>
  </xsl:template>
```

```

<xsl:template match="name" mode="toc">
  <li><xsl:value-of select="last_name"/>,
  <xsl:value-of select="first_name"/></li>
</xsl:template>
<!-- Normal Mode Templates -->
<xsl:template match="person">
  <p><xsl:apply-templates/></p>
</xsl:template>
</xsl:stylesheet>

```

4.9 Attribute Value Templates

Template rule membungkus elemen input person didalam elemen span HTML yang memiliki atribut class dengan nilai person.

```

<xsl:template match="person">
  <span class="person"><xsl:apply-templates/></span>
</xsl:template>

```

Namun, permasalahannya adalah jika nilai atribut tidak diketahui ketika stylesheet ditulis, tapi harus dibaca dari dokumen inputan. Solusinya dengan menggunakan nilai atribut template. Nilai atribut template merupakan ekspresi XPath. Misalnya, kita ingin menulis nama template yang mengubah input elemen name untuk elemen kosong dengan first_name, middle_initial, last_name dan atribut seperti :

```
<name first="Richard" initial="P" last="Feynman"/>
```

Template ini menyelesaikan task berikut :

```

<xsl:template match="name">
  <name first="{first_name}" initial="{middle_initial}"
    last="{last_name}" />
</xsl:template>

```

4.10 XSLT and Namespaces

XPath *pattern*, serta ekspresi yang sesuai dengan elemen yang dipilih, mengidentifikasi elemen berdasarkan lokal path dan *namespace* URI tidak mempertimbangkan awalan namespace. Yang paling umum dilakukan adalah awalan namespace yang sama dipetakan ke URI yang sama pula pada dokumen XML dan stylesheet. Namun hal itu tidaklah diperlukan.

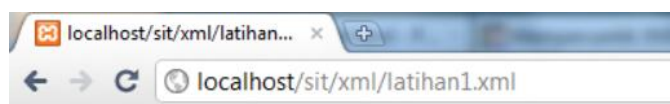
4.11. Menjalankan File XML Dalam Bentuk HTML Menggunakan XSL

Untuk dapat menampilkan isi file XML ke browser, maka harus dibawa dulu ke dalam bentuk HTML. Untuk itu diperlukan XSL untuk menerjemahkan XML ke HTML. XSL ini sejenis CSS jika dalam bahasa HTML. Langkah-langkah yang digunakan adalah sebagai berikut:

1. Siapkan file XML yang akan ditampilkan ke browser. Misal file xml pada sub bab 2.1 (latihan1.xml).
2. Buat file XSL (misal: tabel.xsl) dan ketikkan script sebagai berikut:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>Tabel Mahasiswa</h2>
    <table border="1">
      <tr>
        <td>Nama</td>
        <td>NIM</td>
        <td>Alamat</td>
      </tr>
      <xsl:for-each select="data_mahasiswa/mhs">
        <tr>
          <td><xsl:value-of select="nama"/></td>
          <td><xsl:value-of select="nim"/></td>
          <td><xsl:value-of select="alamat"/></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

3. Jalankan file latihan1.xml dengan browser melalui XAMPP. Maka hasilnya akan terlihat



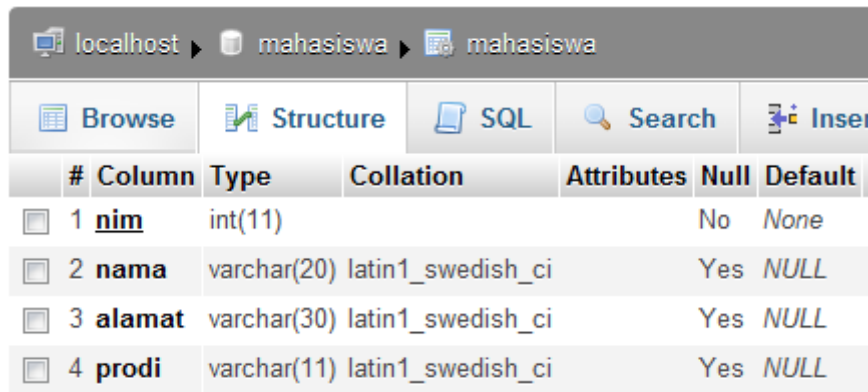
Tabel Mahasiswa

Nama	NIM	Alamat
Fajar Barrun	03672	cilacap
Farisa	03628	Madiun
Redha	03693	Malang

4.12 Parsing data XML menggunakan PHP

Buatlah sebuah file php yang berisi coding dibawah ini.

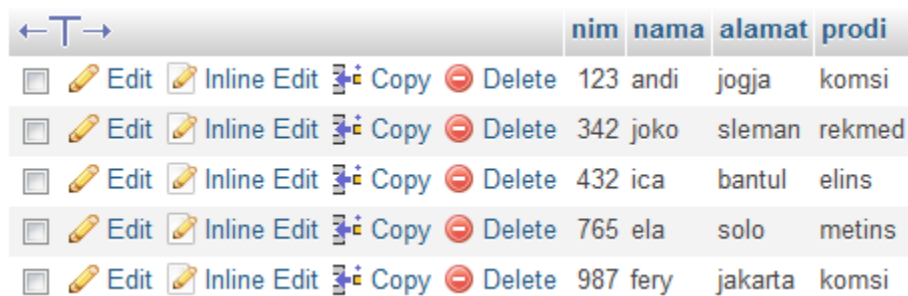
1. Sebelumnya buat sebuah database terlebih dahulu, misalnya mahasiswa dan tabelnya juga bernama mahasiswa yang berisi 4 kolom yaitu nim, nama, alamat dan prodi.



The screenshot shows the MySQL Structure view for a table named 'mahasiswa'. The table has four columns: 'nim' (int(11)), 'nama' (varchar(20) latin1_swedish_ci), 'alamat' (varchar(30) latin1_swedish_ci), and 'prodi' (varchar(11) latin1_swedish_ci). The 'nim' column is the primary key. The 'nama', 'alamat', and 'prodi' columns are nullable.

#	Column	Type	Collation	Attributes	Null	Default
1	<u>nim</u>	int(11)			No	None
2	nama	varchar(20) latin1_swedish_ci			Yes	NULL
3	alamat	varchar(30) latin1_swedish_ci			Yes	NULL
4	prodi	varchar(11) latin1_swedish_ci			Yes	NULL

2. Isi table tersebut dengan beberapa data.



The screenshot shows the MySQL Table view for the 'mahasiswa' table. It displays five rows of data with columns 'nim', 'nama', 'alamat', and 'prodi'. Each row has edit, inline edit, copy, and delete icons.

	nim	nama	alamat	prodi
	123	andi	jogja	komsu
	342	joko	sleman	rekmed
	432	ica	bantul	elins
	765	ela	solo	metins
	987	fery	jakarta	komsu

3. Kemudian buat coding seperti dibawah ini secara berurutan dan simpan pada root server (htdocs) dengan nama mahasiswa.php.

```
<?php

//1.koneksi database
$konek = mysql_connect("localhost", "root", "");
$db = mysql_select_db("mahasiswa");
if ($konek)
{
    echo("sip <br>");
}
else
{
    echo("ora sip <br>");
}
```

```

if ($db)
{
    echo("db ada <br>");
}
else
{
    echo("db null <br>");
}

```

Buatlah koneksinya, jika nanti saat dijalankan pada browser terdapat keterangan sip maka sudah terkoneksi dan terdapat pula keterangan bahwa db ada. Sebaliknya jika tidak terkoneksi maka terdapat keterangan ora sip dan db null. Pastikan nama databasenya sesuai dengan yang sudah anda buat pada server.

```

//2.query database
$query = "select * from mahasiswa";
$hasil = mysql_query($query);

$datamahasiswa = array();

while ($data = mysql_fetch_array($hasil))
{
    $datamahasiswa [] = array('nim' => $data['nim'],
    'nama' => $data['nama'],
    'alamat' => $data['alamat'],
    'prodi' => $data['prodi']);
    //echo $data['nim'];
}

```

Langkah kedua yaitu query database, memilih kolom dari table pada database. Kolom-kolom tersebut dibuat menjadi sebuah array agar dapat memilih semua baris data yang ada pada table.

```

//3.parsing data XML
$document = new DOMDocument();
$document->formatOutput = true;
$root = $document->createElement("data");
$document->appendChild($root);
foreach($datamahasiswa as $mahasiswa)
{
    $block = $document->createElement("mahasiswa");

    //create element nim
    $nim = $document->createElement("nim");
    //createElement untuk membuat element baru
    $nim->appendChild($document->createTextNode($mahasiswa['nim']));
    //createTextNode untuk menampilkan isi/value
    $block->appendChild($nim);
    //appendChild untuk mempersiapkan nilai dari element diatasnya

    //create element nama
    $nama = $document->createElement("nama");
    $nama->appendChild($document->createTextNode($mahasiswa['nama']));
    $block->appendChild($nama);

    //create element alamat
    $alamat = $document->createElement("alamat");
    $alamat->appendChild($document->createTextNode($mahasiswa['alamat']));
    $block->appendChild($alamat);

    //create element prodi
    $prodi = $document->createElement("prodi");
    $prodi->appendChild($document->createTextNode($mahasiswa['prodi']));
    $block->appendChild($prodi);

    $root->appendChild($block);
}

```

Dalam tahap ini, singkatnya adalah membuat file xml nya dengan kata lain parsing. Setelah file mahasiswa.php nya di jalankan pada browser, nantinya akan terbuat file xml dengan tag-tag seperti coding diatas.

```
//4.Menyimpan data dalam bentuk file XML
$generateXML = $document->save("mahasiswa.xml");
if($generateXML)
{
    echo("berhasil di generate <br>");
}
else
{
    echo("gagal <br>");
}
```

Tahap ke empat yaitu menyimpan data mahasiswa.php menjadi mahasiswa.xml setelah mahasiswa.php dijalankan di browser. Jika parsing berhasil dan penyimpanan file xml juga berhasil maka akan ada keterangan berhasil di generate jika tidak terdapat keterangan gagal.

```
//5.Membaca file XML
//membuka file
$url = "http://localhost/parsing/mahasiswa.xml";
$client = curl_init($url);
curl_setopt($client, CURLOPT_RETURNTRANSFER, 1);
$response = curl_exec($client);
curl_close($client);
//membaca file
```

Setelah berhasil menyimpan file xml pada root server, selanjutnya adalah membaca file xml nya. Pada tahap ini terdapat tempat dimana akan menyimpan file xml setelah di parsing. Dan curl harus aktif dan sudah di bahas dalam bab sebelumnya tentang bagaimana mengaktifkan curl.

```
//6.Ditampilkan dalam bentuk HTML
$datamahasiswaxml = simplexml_load_string($response);
//print_r($datamahasiswaxml);
//perulangan
echo "
    <table border='1'>
        <tr>
            <td>NIM</td>
            <td>Nama</td>
            <td>Alamat</td>
            <td>Prodi</td>
        </tr>";
foreach($datamahasiswaxml->mahasiswa as $mahasiswa)
{
    echo "
        <tr>
            <td>".$mahasiswa->nim."</td>
            <td>".$mahasiswa->nama."</td>
            <td>".$mahasiswa->alamat."</td>
            <td>".$mahasiswa->prodi."</td>
        </tr>";
}
echo "</table>"
?>
```

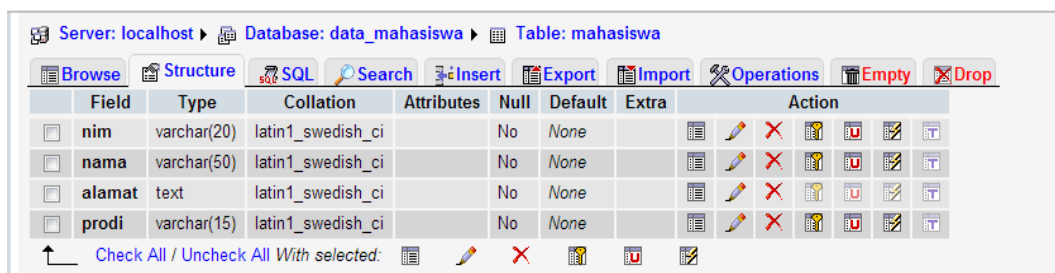
Tahap terakhir yaitu menampilkannya dalam bentuk HTML. Print_r(\$datamahasiswaxml) berfungsi untuk mengecek apakah data mahasiswaxml sudah berisi data yang diambil dari table mahasiswa ataukah belum.

NIM	Nama	Alamat	Prodi
123	andi	jogja	komsu
342	joko	sleman	rekmed
432	ica	bantul	elins
765	ela	solo	metins
987	fery	jakarta	komsu

Gambar diatas adalah hasil yang ditampilkan dalam bentuk html pada browser kita setelah melewati beberapa proses dari file php ke file xml dan ditampilkan dalam bentuk html.

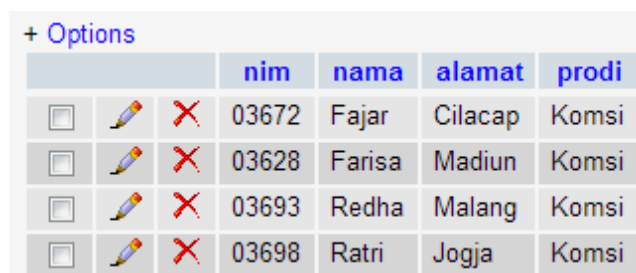
4.13 Menampilkan Isi Database ke Dalam Bentuk HTML

Pertama yang dilakukan menyiapkan data base seperti dibawah ini, dengan nama data base “data_mahasiswa” dan nama table “mahasiswa” yang berisi nim, nama, alamat, prodi mirip dengan database sebelumnya pada bab parsing data XML



Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/> nim	varchar(20)	latin1_swedish_ci		No	None		
<input type="checkbox"/> nama	varchar(50)	latin1_swedish_ci		No	None		
<input type="checkbox"/> alamat	text	latin1_swedish_ci		No	None		
<input type="checkbox"/> prodi	varchar(15)	latin1_swedish_ci		No	None		

Isi tabel tersebut dengan beberapa data



	nim	nama	alamat	prodi
<input type="checkbox"/>	03672	Fajar	Cilacap	Komsi
<input type="checkbox"/>	03628	Farisa	Madiun	Komsi
<input type="checkbox"/>	03693	Redha	Malang	Komsi
<input type="checkbox"/>	03698	Ratri	Jogja	Komsi

Agar database tersebut dapat digunakan untuk aplikasi yang berbeda, maka data-data tersebut harus diubah ke dalam format data web service, dalam hal ini menggunakan format XML. Berikut script PHP untuk melakukan metode-metode menampilkan data yang tersimpan dalam database mysql ke dalam bentuk XML, file ini disimpan dengan nama *mhs.php*. Setting koneksi ke database diatur di dalam script *koneksi.php*.

Script *koneksi.php*.

```
<?php
// koneksi database
$link = mysql_connect('localhost','root','')
or die('Could not connect: ' . mysql_error());
mysql_select_db('data_mahasiswa') or die('Could not select database');
```

Script *mhs.php*.

```
<?php
// header untuk format xml, jika dihilangkan maka akan berbentk data string
header('Content-Type: text/xml; charset=ISO-8859-1');
include "koneksi.php";

//check for the path elements
$spath = $_SERVER[PATH_INFO];
if ($spath != null) {
    $spath_params = spliti ("/", $spath);
}

//metode request untuk GET
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    if ($spath_params[1] != null) {
        $query = "select
            nim,
            nama,
            alamat,
            prodi
            from mahasiswa where nim = $spath_params[1]";
    } else {
        $query = "select
            nim,
            nama,
            alamat,
            prodi
            from mahasiswa";
    }
    $result = mysql_query($query) or die('Query failed: ' . mysql_error());

    echo "<data>";
    while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
        echo "<mahasiswa>";
        foreach ($line as $key => $col_value) {
            echo "</data>";

            mysql_free_result($result);
        }
    }

    //metode request untuk delete
}
mysql_close($link);
?>
```

Untuk mengecek apakah script *books.php* sudah dapat berkomunikasi dengan database yang ada, coba panggil script *mhs.php* yang ada di server masing-masing melalui browser. Misalnya <http://localhost/xml/mahasiswa/mhs.php>. Jika hasilnya berupa data XML seperti tampak pada gambar berikut, maka script *mhs.php* sudah berjalan dengan benar.

```
← → ↻ localhost/xml/mahasiswa/mhs.php

This XML file does not appear to have any style information a

▼<data>
  ▼<mahasiswa>
    <nim>03672</nim>
    <nama>Fajar</nama>
    <alamat>Cilacap</alamat>
    <prodi>Komsis</prodi>
  </mahasiswa>
  ▼<mahasiswa>
    <nim>03628</nim>
    <nama>Farisa</nama>
    <alamat>Madiun</alamat>
    <prodi>Komsis</prodi>
  </mahasiswa>
  ▼<mahasiswa>
    <nim>03693</nim>
    <nama>Redha</nama>
    <alamat>Malang</alamat>
    <prodi>Komsis</prodi>
  </mahasiswa>
  ▼<mahasiswa>
    <nim>03698</nim>
    <nama>Ratri</nama>
    <alamat>Jogja</alamat>
    <prodi>Komsis</prodi>
  </mahasiswa>
</data>
```

Untuk menampilkan data XML ke dalam HTML (web browser), maka diperlukan tag-tag HTML dikombinasikan dengan script *mhs.php*. Perhatikan script *view_mahasiswa.php* berikut:

```
<html>
  <head>
    <title> View Data Bukti</title>
  </head>
  <body>
    <?php
      //include "koneksi.php";
      //Resources
      $url = 'http://localhost/xml/mahasiswa/mhs.php';

      //MENGAMBIL Data String dari resources
      $client = curl_init($url);
      curl_setopt($client, CURLOPT_RETURNTRANSFER, 1);
      $response = curl_exec($client);
      curl_close($client);

      $datamahasiswaxml = simplexml_load_string($response);

      echo "<table border='1'>
        <tr>
          <td>NIM</td>
          <td>Nama</td>
          <td>Alamat</td>
          <td>Prodi</td>
        </tr>";

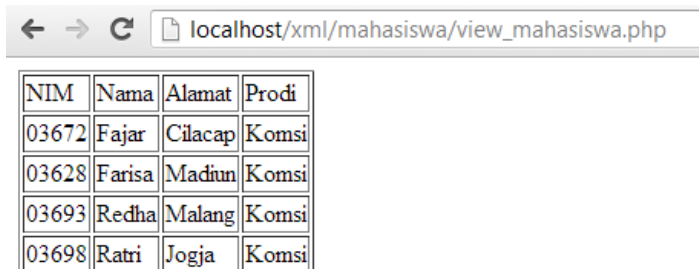
      foreach ($datamahasiswaxml->mahasiswa as $mahasiswa)
      {
        echo "
          <tr>
            <td>".$mahasiswa->nim."</td>
```

```

        <td>".$mahasiswa->nama."</td>
        <td>".$mahasiswa->alamat."</td>
        <td>".$mahasiswa->prodi."</td>
    </tr>
    ";
}
echo "</table>";
?>
</body>
</html>

```

Hasil tampilan *view_mahasiswa.php* di browser:



NIM	Nama	Alamat	Prodi
03672	Fajar	Cilacap	Komsi
03628	Farisa	Madiun	Komsi
03693	Redha	Malang	Komsi
03698	Ratri	Jogja	Komsi

4.14 Menambah Data ke Database Menggunakan XML

Untuk menambah data menggunakan XML kita tambahkan saja pada script yang sudah kita buat sebelumnya yaitu pada script *mhs.php*. kita tambahkan fungsi metode request untuk POST di bawah metode request untuk GET. Berikut script lengkapnya dari *mhs.php* :

```

<?php
// header untuk format xml, jika dihilangkan maka akan berbentuk data string
header('Content-Type: text/xml; charset=ISO-8859-1');
include "koneksi.php";

//check for the path elements
$spath = $_SERVER[PATH_INFO];
if ($spath != null) {
    $spath_params = spliti ("/", $spath);
}

//metode request untuk GET
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    if ($spath_params[1] != null) {
        $query = "select
            nim,
            nama,
            alamat,
            prodi
            from mahasiswa where nim = $spath_params[1]";
    } else {
        $query = "select
            nim,
            nama,
            alamat,
            prodi
            from mahasiswa";
    }
    $result = mysql_query($query) or die('Query failed: ' . mysql_error());

    echo "<data>";
    while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
        echo "<mahasiswa>";
        foreach ($line as $key => $col_value) {

```

```

//metode request untuk post
else if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $input = file_get_contents("php://input");
    $xml = simplexml_load_string($input);
    foreach ($xml->mahasiswa as $mahasiswa) {
        $querycek = "select * from mahasiswa where nim='".$mahasiswa->nim'";
        $num_rows = mysql_num_rows($querycek);
        if ($num_rows == 0)
        {
            $query = "insert into mahasiswa (
                nim,
                nama,
                alamat,
                prodi)
                values (
                '".$mahasiswa->nim',
                '".$mahasiswa->nama',
                '".$mahasiswa->alamat',
                '".$mahasiswa->prodi'
            )";
        }
        else if ($num_rows == 1)
        {
            $query = "update mahasiswa set
                nim = '".$mahasiswa->nim',
                nama = '".$mahasiswa->nama',
                alamat = '".$mahasiswa->alamat',
                prodi = '".$mahasiswa->prodi'
                where nim = '".$mahasiswa->nim'";
        }
        $result = mysql_query($query) or die('Query failed: ' . mysql_error());
    }

    //metode request untuk delete
}
mysql_close($link);
?>

```

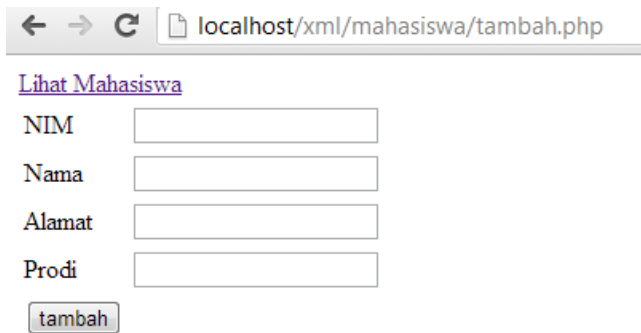
Buat file dengan *tambah.php*. berikut script lengkapnya :

```

<td>NIM</td>
<td><input type="text" name="nim" id="nim"></td>
</tr>
<tr>
<td>Nama</td>
<td><input type="text" name="nama" id="nama"></td>
</tr>
<tr>
<td>Alamat</td>
<td><input type="text" name="alamat" id="alamat"></td>
</tr>
<tr>
<td>Prodi</td>
<td><input type="text" name="prodi" id="prodi"></td>
</tr>
<tr>
<td><input type="submit" name="submit" value="tambah"></td>
</tr>
</table>
</form>
</body>
</html>

```

Hasil tampilan *tambah.php* :



[Lihat Mahasiswa](#)

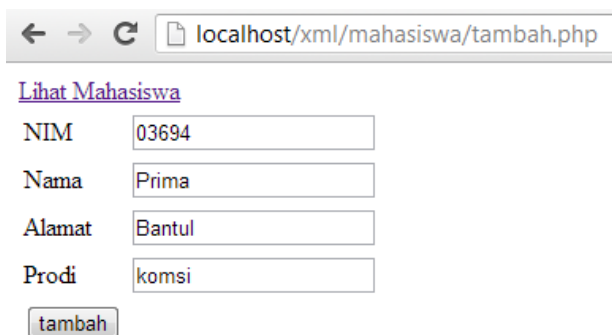
NIM

Nama

Alamat

Prodi

Coba kita isikan dengan contoh data berikut :



[Lihat Mahasiswa](#)

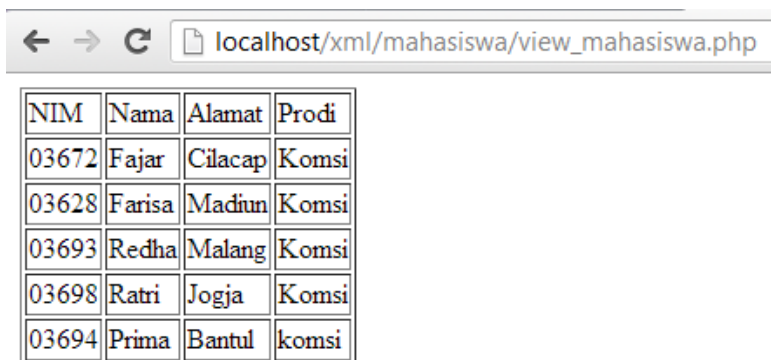
NIM

Nama

Alamat

Prodi

Setelah klik tambah maka kita klik pada link Lihat Mahasiswa :



NIM	Nama	Alamat	Prodi
03672	Fajar	Cilacap	Komsu
03628	Farisa	Madiun	Komsu
03693	Redha	Malang	Komsu
03698	Ratri	Jogja	Komsu
03694	Prima	Bantul	komsu

Data yang sudah di inputkan akan tampil seluruhnya di *view_mahasiswa.php*

BAB V

WEB SERVICE DESCRIPTION LANGUAGE (WSDL)

Tujuan :

1. Menenal Web Services dan arsitekturnya
2. Menenal WSDL dan komponennya

Bagaimana jika akses web site tidak hanya file html? Atau upload dan download file? Akses internet akan lebih berdaya guna jika antar program aplikasi dapat berkomunikasi. Komponen program yang ada di suatu web site yang dapat diakses dari web site lain dikenal dengan istilah Web Service.

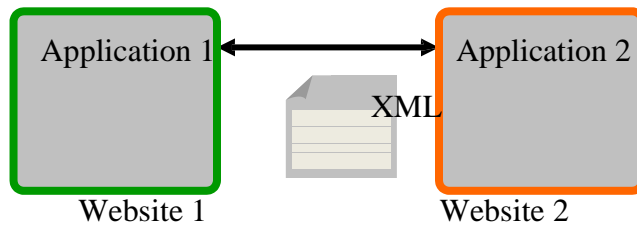
XML, yang tag-tagnya bebas didefinisikan oleh pengguna, dapat digunakan untuk menggambarkan Web Service. WSDL (Web Service Description Language) dibuat berbasis XML untuk menggambarkan Web Service dan bagaimana mengaksesnya.

5.1 Pengantar Web Service

Konsep komunikasi antar program aplikasi sesungguhnya bukan konsep baru. Beberapa terapannya sudah dilakukan sebelumnya :

- Sun RPC (1985)
- CORBA (1992)
- DCE / RPC (1993)
- Microsoft COM (1993)
- Microsoft DCOM (1996)
- Java RMI (1996)

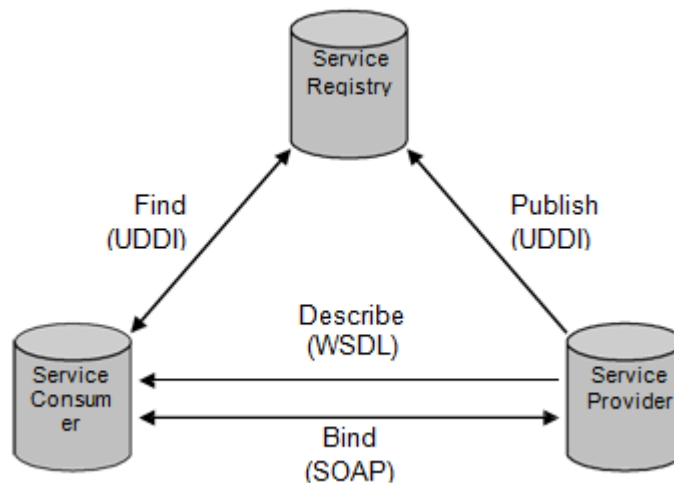
Method yang dimiliki suatu class yang ada di suatu computer dicoba untuk dapat diakses oleh class di computer lain. Konsep ini melahirkan ide pembuatan Web Service, aplikasi di suatu web site dapat diakses dari web site lain. Gambar berikut mungkin lebih menjelaskan konsep Web Service.



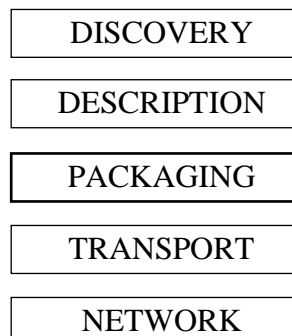
Web Service itu dapat dibuat dalam suatu bahasa pemrograman, dan dikenali oleh bahasa pemrograman lain. Interfacenya didefinisikan dengan XML. Antar aplikasi berinteraksi menggunakan pesan-pesan berbasis XML yang dibawa oleh protokol Internet. Web Service diidentifikasi dengan URI (universal resource identifier).

5.2 Arsitektur Web Service

Web Services dibangun pada landasan protokol XML. Untuk pertukaran pesan mungkin digunakan SOAP, XML-RPC. Untuk menggambarkan service digunakan WSDL. Untuk pencarian service digunakan UDDI. SOAP, WSDL dan UDDI adalah standard de facto implementasi Web Service. Gambar berikut diharapkan lebih menjelaskan peranan masing-masing.



Arsitektur Web Service diimplementasikan dalam lima layer berikut:



Layer teratas, “Discovery”, menyediakan mekanisme standar untuk mendaftarkan dan menemukan Web service, UDDI. Direktori UDDI berisi pointer ke Web service dan dokumen WSDLnya. Direktori UDDI digunakan aplikasi client untuk menemukan Web service.

Layer bawahnya, “Description”, mendeskripsikan Web service, menggunakan standar WSDL. Layer bawahnya, “Packaging”, menyediakan mekanisme pemaketan informasi yang akan dikirimkan atau diterima. Format paket yang sering digunakan adalah SOAP, yang dibangun di atas XML. Dengan SOAP, aplikasi dan komponen dapat mengkomunikasikan pesan call dan return-nya. Layer bawahnya, “Transport” dan “Network” disediakan protocol TCP/IP untuk komunikasi pesan.

5.3 WSDL

WSDL singkatan dari Web Services Description Language, dibuat untuk menggambarkan Web Service. WSDL ditulis dalam format XML, sehingga WSDL adalah dokumen XML. WSDL juga dimanfaatkan untuk mencari Web Service. WSDL sejak 2007 direkomendasikan oleh W3C.

5.4 The WSDL Document Structure

Dokumen WSDL berisi informasi berikut: daftar web services yang tersedia, fungsi Web service, apa parameternya, apa outputnya, format cara pakai, alamat URLnya. Tag-tag yang digunakan dalam dokumen WSDL ditunjukkan dalam tabel berikut.

Tabel Tag-tag dokumen WSDL

Element	Penjelasan
<types>	tipe data yang akan digunakan web service
<message>	messages yang akan dikirimkan web service
<portType>	operasi (function) yang disediakan the web service.
<binding>	Protokol komunikasi yang digunakan web service <ul style="list-style-type: none"> - Bagaimana message dikirim di kabel? - Apa rincian spesifik protokol message protocol disana
<service>	Dimana service disimpan

Struktur utama dokumen WSDL adalah sebagai berikut:

```
<definitions>
  <types>
    definition of types.....
  </types>

  <message>
    definition of a message....
  </message>

  <portType>
    definition of a port.....
  </portType>

  <binding>
    definition of a binding....
  </binding>

</definitions>
```

Elemen **<message>**, mirip parameter, mendefinisikan elemen- elemen data operasi. Setiap message dapat terdiri dari satu atau beberapa bagian. Elemen **<types>** mendefinisikan tipe data yang digunakan web service. Untuk netralitas maksimum, WSDL menggunakan skema XML untuk mendefinisikan tipe data. Elemen **<binding>** mendefinisikan format message dan rincian protokol untuk setiap port. Berikut adalah contoh dokumen WSDL yang disederhanakan:

```
<message name="getBookResponse">
  <part name="resp" element="book"/>
</message>

<portType name="bookPortType">
  <operation name="getBook">
    <input message="getBookRequest"/>
    <output message="getBookResponse"/>
  </operation>
</portType>
<binding type="bookPortType" name="bookBind">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation>
    <soap:operation soapAction="getBook"/>
    <input> <soap:body use="literal"/> </input>
    <output> <soap:body use="literal"/> </output>
  </operation>
</binding>
<service name="Hello_Service">
  <port binding="bookBind" name="bookPort">
    <soap:address location="http://localhost/bookservice"/>
  </port>
</service>
```

Di contoh ini elemen **<portType>** mendefinisikan "bookPortType" sebagai nama **port**, dan "getBook" sebagai nama **operation**. Operasi "getTerm" memiliki **input message** yang disebut "getBookRequest" dan **output message** yang disebut "getBookResponse".

5.5 The Full WSDL Syntax

Syntax WSDL 1.2 yang digambarkan oleh Draft Kerja W3C adalah sebagai berikut.

```
<wsdl:definitions name="nmtoken"?
    targetNamespace="uri">
    <import namespace="uri" location="uri"/> *
    <wsdl:documentation .... /> ?
    <wsdl:types> ?
        <wsdl:documentation .... /> ?
        <xsd:schema .... /> *
    </wsdl:types>
    <wsdl:message name="ncname"> *
        <wsdl:documentation .... /> ?
        <part name="ncname" element="qname"?
            type="qname"?/> *
    </wsdl:message>
    <wsdl:portType name="ncname"> *
        <wsdl:documentation .... /> ?
        <wsdl:operation name="ncname"> *
            <wsdl:documentation .... /> ?
            <wsdl:input message="qname"> ?
                <wsdl:documentation .... /> ?
            </wsdl:input>
            <wsdl:output message="qname"> ?
                <wsdl:documentation .... /> ?
            </wsdl:output>
            <wsdl:fault name="ncname" message="qname"> *
                <wsdl:documentation .... /> ?
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:serviceType name="ncname"> *
        <wsdl:portType name="qname"/> +
    </wsdl:serviceType>
    <wsdl:binding name="ncname" type="qname"> *
        <wsdl:documentation .... /> ?
        <!-- binding details --> *
        <wsdl:operation name="ncname"> *
            <wsdl:documentation .... /> ?
            <!-- binding details --> *
            <wsdl:input> ?
                <wsdl:documentation .... /> ?
            <!-- binding details -->
        </wsdl:input>
        <wsdl:output> ?
            <wsdl:documentation .... /> ?
            <!-- binding details --> *
        </wsdl:output>
        <wsdl:fault name="ncname"> *
            <wsdl:documentation .... /> ?
```

```

        <!-- binding details --> *
    </wsdl:fault>
</wsdl:operation>
</wsdl:binding>
    <wsdl:service name="ncname" serviceType="qname">
*
    <wsdl:documentation .... /> ?
    <wsdl:port name="ncname" binding="qname"> *
        <wsdl:documentation .... /> ?
        <!-- address details -->
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

BAB VI

SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

Tujuan :

1. mengenal apakah SOAP itu dan apa manfaatnya
2. mengetahui bagaimana menggunakannya

SOAP adalah protokol berbasis XML yang memungkinkan aplikasi bertukar informasi melalui HTTP. Bab ini akan menjelaskan apa itu SOAP, dan bagaimana menggunakan XML untuk bertukar informasi di antara aplikasi.

6.1 Pengantar SOAP

Banyak aplikasi saat ini berkomunikasi menggunakan Remote Procedure Calls (RPC) antar objek seperti DCOM dan CORBA, tetapi HTTP tidak dirancang untuk ini. SOAP, singkatan Simple Object Access Protocol, adalah protokol berbasis XML yang memungkinkan aplikasi bertukar informasi melalui HTTP. Dalam SOAP diatur format pengiriman pesan.

Selain menggunakan HTTP, mungkin juga menggunakan FTP atau SMTP. SOAP bebas platform, tidak seperti RMI yang harus Java atau DCOM. SOAP efektif untuk membangun Web services. SOAP sederhana dan ringan untuk pengiriman data dibanding CORBA, RMI, DCOM. SOAP mudah dipelajari dan diimplementasikan. SOAP didukung IBM, Microsoft, Apache, direkomendasikan W3C, sejak 2003.

Output Web service akan dipakai aplikasi client yang mungkin beda platform. Format XML berbasis plain-text, dapat digunakan untuk pertukaran data. XML digunakan untuk menggambarkan data dalam pesan SOAP.

6.2 SOAP Syntax

Pesan SOAP digunakan untuk bertukar data. Pesan berisi : data yang dikirim dan diterima, skema pengkodean dan alamat HTTP tujuan. Message SOAP adalah dokumen XML berisi elemen berikut:

- Elemen Envelope yang menunjukkan dokumen XML sebagai pesan SOAP.

Elemen ini merupakan elemen tertinggi, dan merupakan container seluruh elemen SOAP

Message.

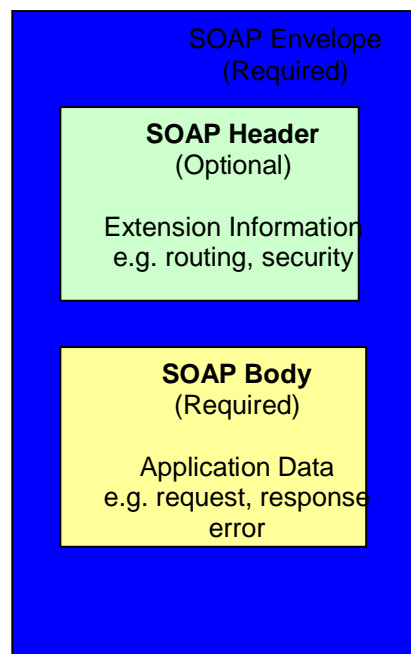
- Elemen Header yang berisi informasi tambahan, memungkinkan otentikasi data atau untuk manajemen transaksi, informasi routing, informasi sumber. elemen Body yang berisi data yang dikirimkan. elemen Fault berisi informasi error dan status. Seluruh elemen diatas dideklarasikan dalam namespace default untuk SOAP envelope:

<http://www.w3.org/2001/12/soap-envelope>

dan namespace default untuk kode SOAP dan tipe datanya di:

<http://www.w3.org/2001/12/soap-encoding>

Gambar berikut menggambarkan keterkaitan antar elemen. Gambar 10.2 menggambarkan tag-tagnya.



Beberapa aturan penting pembuatan SOAP Message:

- harus menggunakan SOAP Envelope namespace
- harus menggunakan SOAP Encoding
- namespace harus tidak berisi rujukan DTD
- harus tidak mengandung "XML Processing Instructions"

```

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
    ...
    ...
</soap:Header>
<soap:Body>
    ...
    ...
    <soap:Fault>
        ...
        ...
    </soap:Fault>
</soap:Body>
</soap:Envelope>

```

Gambar 10.2 Tag-tag SOAP Message

6.3 Contoh SOAP

Berikut adalah contoh SOAP Message yang dikirim dan yang diterima.

SOAP Message yang dikirim.

POST /InStock HTTP/1.1

Host: www.stock.org

Content-Type: application/soap+xml; charset=utf-8

Content-Length: nnn

```

<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header>
    ... (optional header information)
  </soap:Header>
  <soap:Body xmlns:m="http://www.stock.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>

```

SOAP Message yang diterima

HTTP/1.1 200 OK

Content-Type: application/soap; charset=utf-8

Content-Length: nnn

```

<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"

```

```

    soap:encodingStyle ="http://www.w3.org/2001/12/soapencoding">

    <soap:Header>
        ... (optional header information)
    </soap:Header>

    <soap:Body xmlns:m ="http://www.stock.org/stock">
        <m:GetStockPriceResponse>
            <m:Price>34.5</m:Price>
        </m:GetStockPriceResponse>
    </soap:Body>
</soap:Envelope>

```

6.4 Elemen “Fault”

Elemen “Fault”, bersifat optional, menyimpan informasi error dan status untuk suatu SOAP message. Jika ada, ia harus menjadi elemen anak dari elemen “Body”, dan hanya boleh muncul satu kali. Elemen Fault berisi sub elemen berikut:

Sub Elemen	Penjelasan
<faultcode>	Kode kesalahan
<faultstring>	Penjelasan error
<faultactor>	Informasi siapa penyebab kesalahan
<detail>	Berisi rincian kesalahan terkait elemen “Body”

Kode berikut menunjukkan contoh pesan SOAP yang berisi error.

```

<Body xmlns=http://www.w3.org/2001/12/soap-envelope>
<Fault>
    <faultcode>Client</faultcode>
    <faultstring>Something went wrong</faultstring>
    <detail>Application specific error information</detail>
</Fault>
</Body>

```

Kode kesalahan berikut digunakan untuk menggambarkan kesalahan yang terjadi.

Error	Penjelasan
VersionMismatch	namespace tidak sesuai
MustUnderstand	Elemen anak dari Header bertanda
Client	Kesalahan dari client
Server	Kesalahan dari server

6.5 Memproses SOAP Message

Aplikasi dan komponen yang memproses SOAP message disebut SOAP node. SOAP node mungkin berperan sebagai : Sender, Receiver, atau Intermediary. Perannya disimpan di Header

SOAP message. Sender mengirim SOAP message ke receiver. Pengirim awal SOAP disebut initial SOAP sender. Node perantara menerima dari SOAP sender dan mengirim ulang ke SOAP node lain.

SOAP Header dapat berisi dua atribut global:

- SOAP role : menyatakan peran SOAP node ketika pesan diterima.
- SOAP mustUnderstand : menyatakan apakah block Header diproses oleh SOAP node.

BAB VII

UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION (UDDI)

Tujuan :

1. Apakah UDDI itu?
2. Apa yang dimaksud dengan :
 - Entitas bisnis
 - Bisnis service
 - Binding template
 - tModels
 - UDDI query

UDDI, singkatan dari Universal Description, Discovery and Integration, adalah standar berbasis XML untuk menggambarkan, mempublikasikan, dan menemukan Web services. Di bab ini akan dipelajari apa itu UDDI, mengapa dan bagaimana kita menggunakannya.

7.1 Pengantar UDDI

UDDI, singkatan dari Universal Description, Discovery and Integration (UDDI) Protocol, adalah standar berbasis XML untuk menggambarkan, mempublikasikan dan menemukan Web service. UDDI merupakan insiatif industri terbuka, oleh Microsoft, IBM, Ariba, dan sebagainya, yang memungkinkan pelaku bisnis saling menemukan satu dengan yang lain dan mendefinisikan bagaimana mereka berinteraksi melalui internet.

UDDI bebas platform, merupakan kerangka kerja terbuka, dapat berkomunikasi melalui SOAP, CORBA, dan protocol RMI Java. UDDI menggunakan WSDL untuk menggambarkan interface ke web service. Bersama SOAP dan WSDL merupakan tiga fondasi standar web service.

UDDI terdiri dari dua bagian:

- direktori seluruh metadata web service termasuk pointer ke WSDL-nya
- direktori yang digunakan aplikasi client untuk menemukan Web service.

7.2 Tiga tipe informasi UDDI

Perusahaan dapat mendaftarkan dirinya ke dalam tiga tipe informasi dalam UDDI registry. Informasi ini dimasukkan dalam tiga elemen UDDI. Tiga elemen itu adalah :

- white page
- yellow page
- green page.

Kategori “white page” berisi : informasi dasar tentang perusahaan dan bisnisnya, informasi kontak bisnis : nama bisnis, alamat, nomor telpon, NPWP, dsb. Informasi ini penting bagi pihak lain yang ingin tahu bisnis perusahaan tersebut.

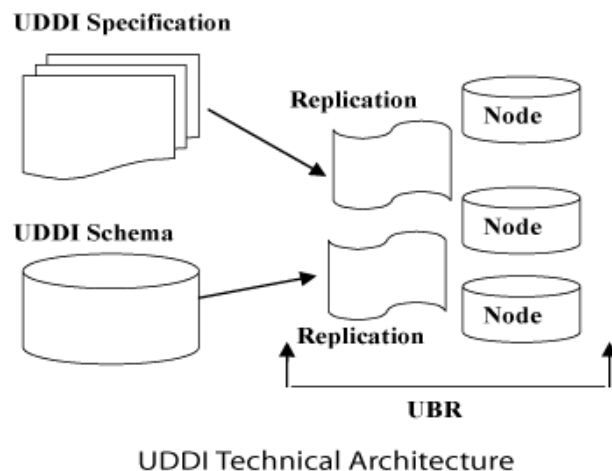
Kategori “yellow page” berisi : rincian lebih lanjut tentang perusahaan, termasuk kemampuan e-commerce. Di sini digunakan skema kategorisasi industri, kode industri, kode produk, kode identifikasi bisnis. Informasi ini memudahkan customer mencari produk yang mereka inginkan.

Kategori “green pages” berisi informasi teknik tentang web service: lokasi URL, informasi “discovery” dan data terkait untuk menemukan dan menjalankan Web Service. Jadi, UDDI tidak hanya dibatasi untuk menggambarkan web service berdasarkan SOAP. Lebih daripada itu UDDI dapat digunakan untuk menggambarkan beragam service, dari sebuah halaman web atau alamat email, hingga SOAP, CORBA, dan servis Java RMI.

7.3 Arsitektur Teknik

Arsitektur Teknik UDDI terdiri dari tiga bagian :

- *UDDI data model* : skema XML untuk menggambarkan bisnis dan web service.
- *UDDI API specification* : spesifikasi API untuk mencari dan mempublikasikan data UDDI
- *UDDI cloud services* : ini adalah situs operator yang menyediakan implementasi spesifikasi UDDI dan mensinkronkan secara periodik.



UDDI Business Registry (UBR), adalah sistem konseptual tunggal dari beberapa node yang memiliki data tersinkronisasi melalui replikasi. UDDI berisi skema XML yang menggambarkan lima struktur data:

1. Entitas bisnis

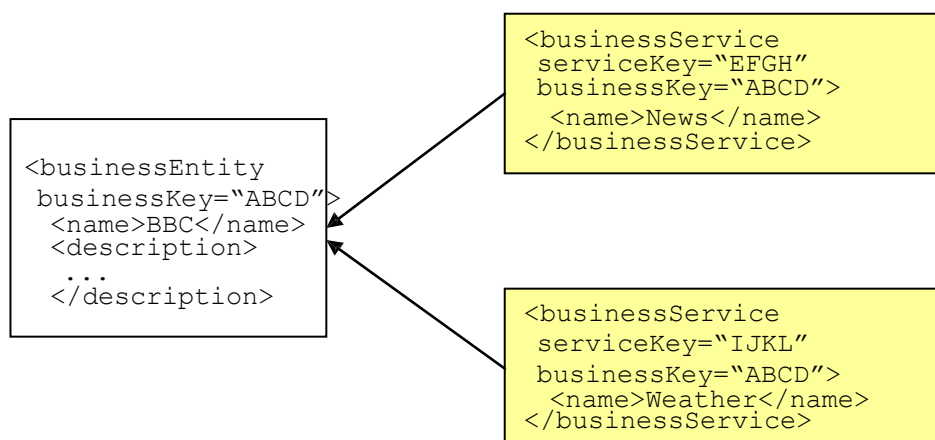
Entitas bisnis menyatakan penyedia web services dalam UDDI registry. Informasinya meliputi : nama perusahaan, deskripsi bisnis, informasi kontak, informasi lain. Gambar 11.2 menunjukkan contoh kasus.

```
<businessEntity
    businessKey="ABCD">
    <name>BBC</name>
    <description>
    ...
    </description>
</businessEntity>
```

Gambar 11.2 Entitas bisnis dalam UDDI

2. Business Service

Disini disimpan semua web service yang disediakan entitas bisnis. Berisi informasi : bagaimana menemukan web service, apa tipe web servicenya, termasuk kategori taxonomi yang mana. Gambar 11.3 menunjukkan contoh kasus ini.

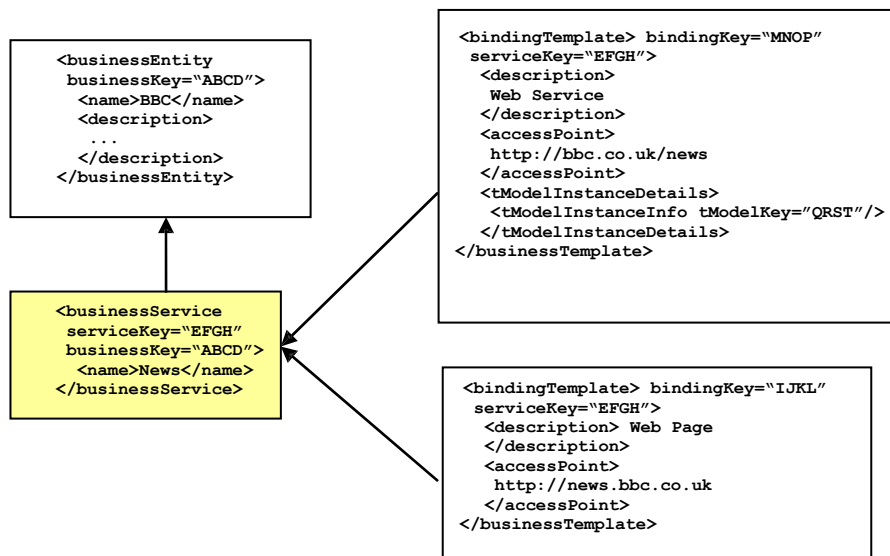


Perhatikan penggunaan Universally Unique Identifiers (UUIDs) dalam atribut *businessKey* dan *serviceKey*. Setiap entitas bisnis dan business service diidentifikasi unik dalam

seluruh registry UDDI melalui UUID yang diberikan registry ketika informasi pertama kali dimasukkan.

3. Binding Template

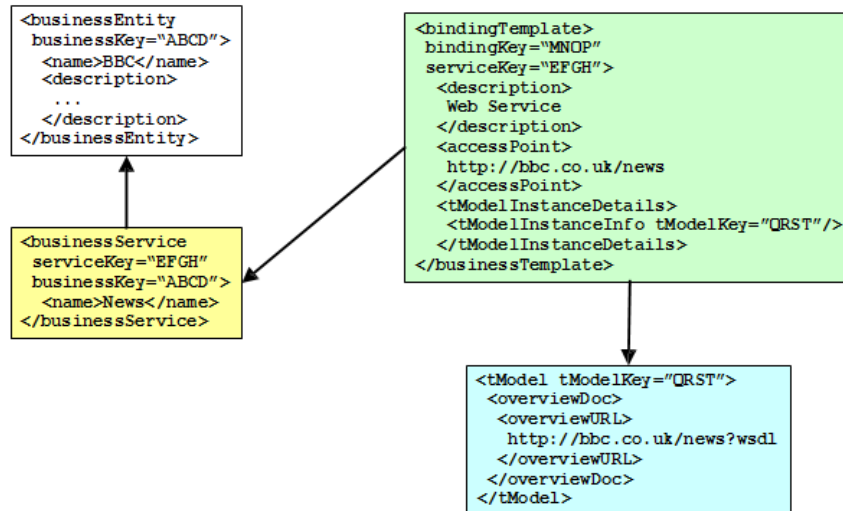
Pada bagian ini disimpan informasi teknis bagaimana service dapat diakses. Terkait Web Service ada alamat Web Service dan parameter. Tidak terkait Web Service ada E-mail, FTP, CORBA, Telephone. Suatu service mungkin memiliki beberapa binding, kaitan (misalnya web service binding, telephone binding). Gambar 11.4 diharapkan memperjelas konsep ini.



Business service boleh memiliki beberapa binding templates. Service boleh menyatakan implementasi berbeda dari service yang sama, masing-masing dikaitkan dengan sekumpulan protokol atau alamat jaringan yang berbeda.

4. tModel

Tidak ada hubungan eksplisit antara UDDI dan WSDL. Binding template berisi access point tetapi belum ada informasi bagaimana cara memakainya (misalnya tipe data yang diharapkan). tModel (Technical Model) menghubungkan deskripsi interface dengan suatu binding. Beberapa binding boleh merujuk interface yang sama. Mirip dengan industri penerbangan mendefinisikan interface standar pemesanan tiket, maskapai penerbangan menerapkan interface ini.



5. Publisher Assertion

Bagian ini menjelaskan hubungan antara dua atau lebih entitas bisnis : bagian, atau departemen. Struktur publisherAssertion terdiri dari tiga elemen: fromKey (businessKey pertama), toKey (businessKey kedua) dan keyedReference. Tag keyedReference berisi hubungannya. Gambar 11.6 menunjukkan contohnya.

```

<element name="publisherAssertion" type="uddi:publisherAssertion"
/>

<complexType name="publisherAssertion">
  <sequence>
    <element ref="uddi:fromKey" />
    <element ref="uddi:toKey" />
    <element ref="uddi:keyedReference" />
  </sequence>
</complexType>

```

BAB VIII

IMPLEMENTASI WEB SERVICE

Tujuan :

1. Mengetahui Web Service Toolkit NuSOAP
2. Mencoba salah satu Web Service Toolkit untuk mengimplementasikan Web Service.

Di bagian terdahulu sudah dibahas Web Service dibangun dari tiga komponen: SOAP, UDDI dan WSDL. Pada bab ini dibahas bagaimana mengimplementasikan Web Service

8.1 Toolkit NuSOAP

NuSOAP adalah sebuah kumpulan class-class PHP yang memungkinkan user untuk mengirim dan menerima pesan SOAP melalui protokol HTTP. NuSOAP didistribusikan oleh NuSphere Corporation (<http://www.nusphere.com>) sebagai open source toolkit di bawah lisensi GNU LGPL.

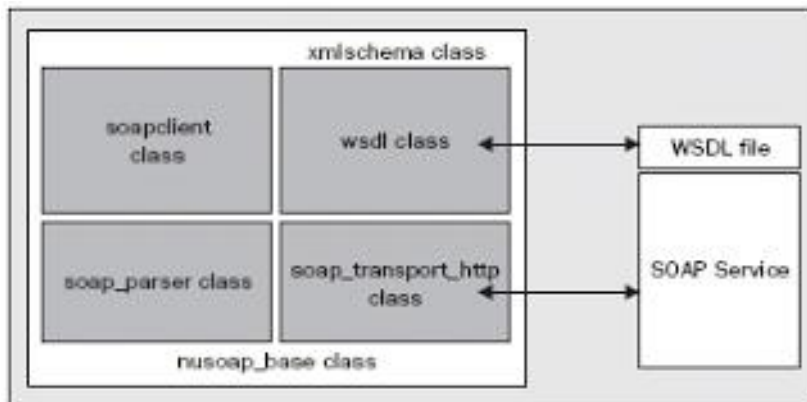


Diagram Web Service dngn Nusoap

Salah satu keuntungan dari NuSOAP adalah bahwa NuSOAP bukan merupakan PHP extension, sehingga penggunaannya tidak membutuhkan registrasi khusus ke Sistem Operasi maupun web server. NuSOAP ditulis dalam kode PHP murni sehingga semua developer web dapat menggunakan tool ini tanpa tergantung pada jenis web server yang digunakan.

NuSOAP merupakan toolkit web service berbasis komponen. NuSOAP memiliki sebuah class dasar yang menyediakan method seperti serialisasi variabel dan pemaketan SOAP-Envelope.

Interaksi web service dilakukan dengan class client yang disebut dengan class “soapclient” dan class server yang disebut dengan class “soap_server”. Class-class ini mengizinkan user untuk melakukan proses pengiriman dan penerimaan pesan-pesan SOAP dengan bantuan beberapa class-class pendukung lainnya untuk melengkapi proses tersebut.

Operasi-operasi pengiriman pesan SOAP dijalankan dengan melibatkan paramater nama operasi yang diinginkan melalui method call(). Jika web service yang dituju menyediakan sebuah file WSDL, maka class “soapclient” akan mengacu langsung pada URL file WSDL tersebut dan menggunakan class “wsdl” untuk mem-parsing file WSDL dan mengekstrak seluruh datanya. Class “wsdl” menyediakan method-method untuk mengekstrak data per-operasi dan per-binding.

Class “soapclient” menggunakan data dari file WSDL untuk menerjemahkan parameter-parameternya sekaligus menyusun SOAP envelope ketika user mengeksekusi suatu pemanggilan service. Ketika pemanggilan ini dieksekusi, class “soapclient” menggunakan “soap_transport_http” untuk mengirim pesan SOAP request dan menerima pesan SOAP response. Selanjutnya pesan SOAP response yang diterima di-parsing dengan menggunakan class “soap_parser” .

8.1 Implementasi Web Service menggunakan Toolkit NuSOAP

1. Buatlah folder “Login” pada *c://xampp/htdocs/* yang akan digunakan untuk menyimpan file-file untuk membuat web service
2. Copy dan paste folder “nusoap” dan “adodb” ke dalam folder “Login” tersebut yang masing-masing folder telah berisi file-file nusoap (sebagai middleware) dan adodb (koneksi ke database).
3. Buatlah file server.php terlebih dahulu yang isinya seperti dibawah ini, lalu simpan

```
<?php
//call library
require_once('nusoap/lib/nusoap.php');
require_once('adodb/adodb/adodb.inc.php');
$server = new nusoap_server;
$server->configureWSDL('server', 'urn:server');
$server->wsdl->schemaTargetNamespace = 'urn:server';

//register a function that works on server
$server->register('login_ws',
array('username' => 'xsd:string',
'password'=>'xsd:string'), //parameters
array('return' => 'xsd:string'), //output
'urn:server', //namespace
'urn:server#loginServer', //soapaction
```



```

        'rpc', // style
        'encoded', // use
        'login'); //description

//create function

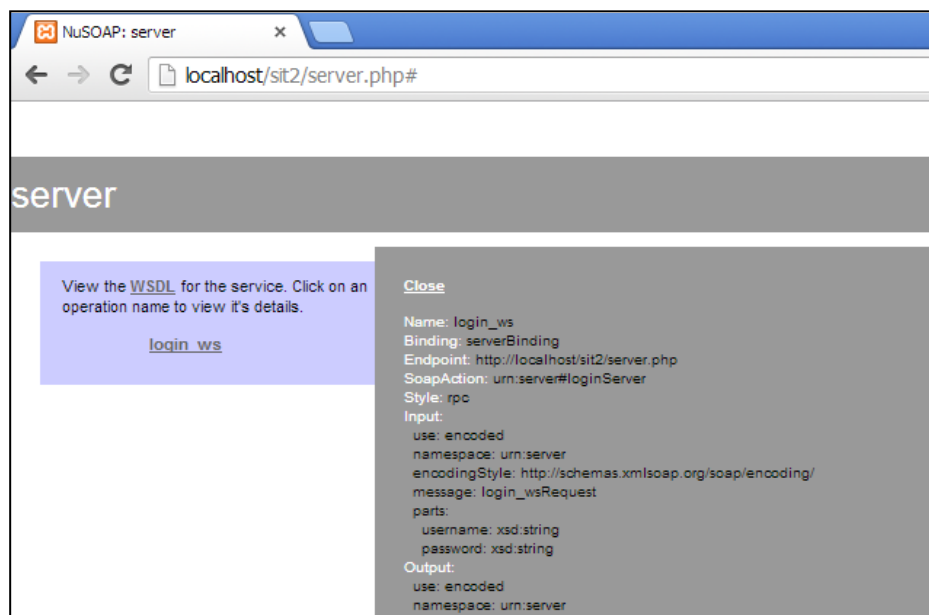
function login_ws($username, $password) {
    //enkripsi password dengan md5
    $password = md5($password);
    //buat koneksi
    $db = NewADOConnection('mysql');
    $db -> Connect('localhost','root','','data_mahasiswa');
    //cek username dan password dari database
    $sql = $db -> Execute("SELECT * FROM user where
    username='$username' AND
    password='$password'");

    //Cek adanya username dan password di database
    if ($sql->RecordCount() >= 1) //sama dengan mysql_num_rows pada php
    biasa
    {
        return "Login Berhasil";
    } else {
        return "Login gagal";
    }
}

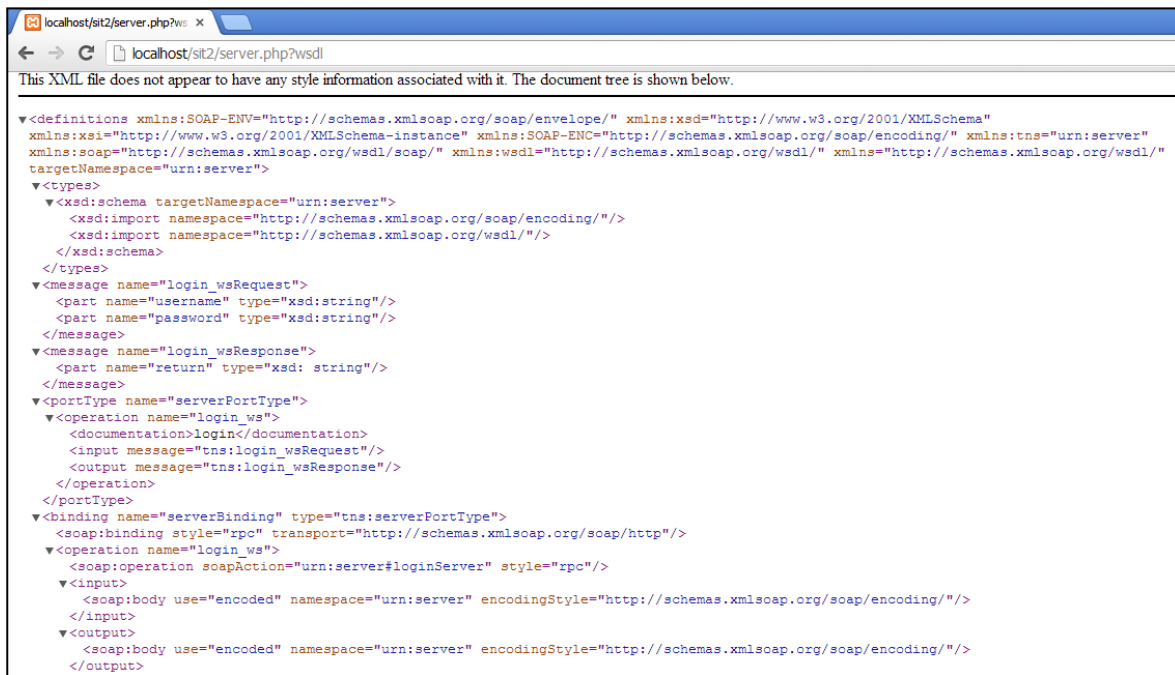
//create HTTP listener
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA : '';
$server->service($HTTP_RAW_POST_DATA);
?>

```

4. Setelah file server.php dan client.php selesai coba cek pada browser dengan menuliskan urlnya: “localhost/login/server.php” maka hasilnya akan seperti berikut.



Jika di klik tulisan “WSDL” pada tampilan server, maka akan muncul tulisan seperti ini :



5. Buatlah file client.php terlebih dahulu yang isinya seperti dibawah ini, lalu simpan.
Jangan lupa untuk membuat session untuk menentukan halaman yang pertama kali diakses

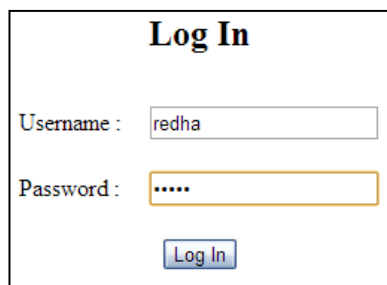
```
<?php
    //memulai
    session
    session_start(
    );
    //panggil library
    require_once('nusoap/lib/nusoa
    p.php');
    //mendefinisikan alamat url service yang disediakan oleh client
    $client = new
    soapclient('http://localhost/login/server.php?wsdl',true);
    $username = $_POST["username"];
    $password = $_POST["password"];
    $result = $client->call('login_ws',
        array('username'=>$username,
        'password'=>$password));
    if($result == "Login Berhasil"){
        $_SESSION['username'] = $username;
        header ("location:index.php");
    } else{
        header ("location:login.php");
    }
}

?>
```

6. Buat file “login.php” sebagai di bawah ini untuk membuat form loginnya, lalu simpan

```
<center><h2><b>Log In</b></h2></center>
<?php
    //memulai
    session
    session_st
    art();
    //cek adanya session, jika session sudah ada maka
    diarahkan ke index.php if
    (ISSET($_SESSION['username'])) {
        header("location:index.php");
    }
?>
<form method="post" action="client.php">
    <table border="0" align="center" cellpadding="5"
    cellspacing="8">
        <tr>
            <td>Username : </td>
            <td><input name="username" type="text"></td>
        </tr>
        <tr>
            <td>Password : </td>
            <td><input name="password" type="password"></td>
        </tr>
        <tr>
            <td colspan="2" align="center" height="10">
                <input name="submit" type="submit" value="Log In">
            </td>
        </tr>
    </table>
</form>
```

Hasil tampilannya adalah sebagai berikut:



7. Buat file “index.php” sebagai di bawah ini untuk membuat form loginnya, lalu simpan.

Pada tampilan index saya ini, saya menggunakan style template dari bootstrap.

```
<?php
    //memulai
    session
    session_st
    art();
    //cek adanya session
```

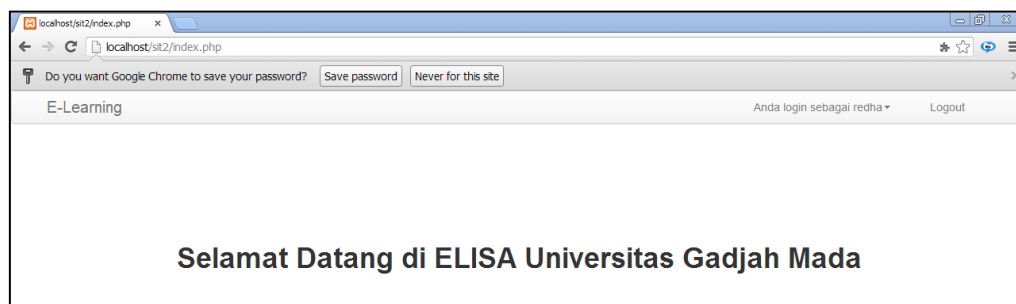
```

        if (ISSET($_SESSION['username']))
        {
            //jika tidak ada session
        } else
            header("location:login.php");
    ?>

    <link href="asset/css/bootstrap.css" rel="stylesheet"
    type="text/css">
    <script type="text/javascript" src="asset/js/bootstrap-
    dropdown.js"> </script>
    <body>
    <div class="navbar navbar fixed-top">
    <div class="navbar-inner">
    <div class="row">
    <div class="span2">
    <a class="brand" href="#">&nbsp;   &nbsp;   E-Learning</a>
    </div>
    <div class="nav-
    collapse">
    <ul class="nav">
    <li class="divider"> </li>
    <li class="dropdown
    offset7">
    <a href="#" class="dropdown-toggle"
    data-toggle="dropdown"></i><?php echo
    "Anda login sebagai ";
    echo $_SESSION['username']; ?><b class="caret"></b></a>
    </li>
    <li><a href="logout.php"><i class="icon-off">
    </i>&nbsp;   &nbsp;  
    &nbsp;  Logout</a></li>
    </ul>
    </div>
    </div>
    </div>
    <br /> <br /> <br /> <br /> <br /> <br />
    <div class="container">
    <center><h1>Selamat Datang di ELISA Universitas Gadjah
    Mada</h1></center>
    </div>
    </body>

```

Tampilan index.php akan menjadi seperti in



8. Terakhir yang perlu di buat adalah file logout.php untuk mengembalikan kembali ke halaman login.php saat dipilih logout.

```
<?php
//memulai
session
session_s
tart();
//cek adanya session, jika session ada maka akan di unset
dan dilanjutkan dengan session_destroy
if (ISSET($_SESSION['u
sername'])) {
    UNSET($_SESSION['us
ername']);
}
header("location: index.php");
session_destroy();
?>
```

DAFTAR PUSTAKA

1. Ian Wang, "Deploying Web Service", users.cs.cf.ac.uk/
I.N.Wang/teaching/lecture a Deploying Web Services.ppt, Maret 2013
2. NIIT, XML Courseware, "02-Querying XML Documents",
NIIT Publishing, 2004
3. www.w3schools.com, Web Services Tutorial, Maret 2013
4. www.w3schools.com, WSDL Tutorial, Maret 2013
5. www.w3schools.com, SOAP Tutorial, Maret 2013
6. www.w3schools.com, UDDI Tutorial, Maret 2013