Generator Form HTML Berbasis Tabel Dengan Pemrograman Berorientasi Objek

Katon Wijana, S.Kom., M.T.
Program Studi Sistem Informasi, Universitas Kristen Duta Wacana
Jl. Dr. Wahidin Sudirohusodo 5-25, Yogyakarta
katony@staff.ukdw.ac.id

***Abstract*— To insert new data into a database table using a web-based application, a graphical user interface in the form of HTML Form is required. Each table field / attribute requires an appropriate form control in order to minimize data errors that will be entered. There is a relation between the data type of a field in the table with the type of form control to be used, therefore the graphical user interface in the form of HTML Form can be created automatically.**

There are various control forms of HTML in the form of tags, generally in the form of input tags. What distinguishes the form control from one to another is the attribute: type, size, value therefore to determine the type and content of form controls can be given through parameters.

HTML Form can be regarded as an object which has many other objects in the form of form controls. Object Oriented Programming (OOP) paradigm it can be implemented to build HTML Form.

Through meta data from a table, it will be able to obtain the appropriate HTML form control, but for each specific data type it can have the appropriate form control candidate, therefore before Form HTML is created by the generator, there should be a little user intervention to get the interface The desired HTML form.

Intisari—Untuk menambahkan data pada tabel memakai aplikasi berbasis web diperlukan antarmuka grafis yang berupa Form HTML. Tiap field/atribut tabel memerlukan kontrol form yang sesuai dalam rangka meminimalkan kesalahan data yang akan dimasukkan. Ada keterkaitan antara tipe data suatu field pada tabel dengan jenis kontrol form yang digunakan, sehingga antarmuka grafis yang berupa Form HTML dapat dibuat secara otomatis.

Ada bermacam-macam kontrol Form HTML yang berupa tag, pada umumnya berupa tag input. Yang membedakan antara kontrol form satu dengan yang lain adalah atribut : type, size, value sehingga untuk menentukan jenis dan isi kontrol form dapat diberikan melalui parameter-parameter.

Form dapat dipandang sebagai suatu objek yang di dalamnya mempunyai banyak objek yang lain yaitu berupa kontrol-kontrol form. Dengan paradigma Pemrograman Berorientasi Objek (PBO) generator Form HTML ini dapat diwujudkan.

Melalui meta data dari suatu tabel, akan dapat diperoleh jenis kontrol form HTML yang sesuai, namun untuk tiap tipe data terentu dapat mempunyai kandidat kontrol form yang sesuai, oleh karena itu sebelum Form HTML dibuat oleh generator, perlu ada sedikit campur tangan pemakai untuk mendapatkan antarmuka Form HTML yang dikehendaki.

***Kata Kunci*— form, html, kontrol form, oop, object, generator, tabel, antarmuka grafis, aplikasi web.**

I. PENDAHULUAN

Untuk menampilkan informasi dan mengambil data pada aplikasi berbasis web digunakan tag-tag HTML yang tersimpan dalam bentuk dokumen teks (*plain text*), sehingga dengan demikian berbagai macam perangkat dengan berbagai *platform* dapat mengkonsumsi dokumen tersebut untuk diproses menjadi hasil keluaran (informasi) maupun sarana untuk mengirimkan data (formulir masukan). Dengan memakai tag-tag HTML tersebut dokumen menjadi bersifat dapat dibuka pada *platform* apa saja namun dengan konsekuensi bahwa dokumen tersebut menjadi tampak kompleks karena merupakan gabungan antara informasi yang sebenarnya dengan tag-tag HTML untuk memformat bentuk tampilan.

Antarmuka grafis web, serupa dengan antarmuka grafis *desktop* maupun *mobile*, merupakan bentuk tampilan yang berupa *form* (formulir) dan kontrol-kontrol *form* yang ada di dalamnya, seperti misalnya *text box, combo box, radio button, date picker, spinner* dan lain sebagainya namun ditulis dalam bentuk teks polos dalam bentuk tag-tag HTML, sehingga bentuk grafis yang seharusnya terlihat diganti dengan notasi-notasi (tag-tag) HTML yang cukup rumit. Selain itu tata letak kontrol *form* tersebut juga harus dilakukan melalui bentuk tabel atau cara yang lain sehingga cukup rumit untuk membuat sebuah Form HTML yang sesuai dengan kebutuhan.

Form HTML dapat dipergunakan untuk banyak keperluan, diantaranya otentikasi pengguna (*user login*), menu pilihan, mengakses halaman web tertentu, dan lain-lain. Pada penelitian ini penulis fokus pada Form HTML yang digunakan untuk menambah (*insert*) baris dari suatu tabel yang dapat dibangun secara otomatis oleh sistem berdasarkan tabel yang akan diakses.

Tipe data dan ukuran data dari tiap atribut tabel dapat dipergunakan sebagai acuan untuk menentukan jenis kontrol *form* dan ukurannya, sehingga dengan demikian pembuatan Form HTML yang rumit dapat dibangun dalam waktu yang relatif singkat melalui program aplikasi Generator Form HTML berbasis tabel ini. Namun demikian, oleh karena suatu tipe data mempunyai lebih dari satu kemungkinan kontrol *form*, maka sebelum benar-benar dibuat secara otomatis diperlukan pengaturan-pengaturan manual terlebih dahulu.

Paradigma Pemrograman Berorientasi Objek menjadi

sarana yang memudahkan untuk terwujudnya generator Form HTML ini, dengan memandang form sebagai sebuah objek dan kontrol-kontrol form sebagai objek yang ada di dalamnya, maka berbagai macam form akan dapat dibuat.

II. LANDASAN TEORI

Untuk mewujudkan Generator Form HTML berbasis tabel ini diperlukan beberapa landasan teori, yaitu Form HTML, tabel, bahasa pemrograman PHP dan Pemrograman Berorientasi Objek.

A. *Form HTML*

- Kontrol Form[1]

Untuk membuat form HTML digunakan tag **<form>** elemen mendefinisikan bentuk yang digunakan untuk mengumpulkan input (kontrol form) seperti Gambar 1 ini:

```
<form>
.
form elements
.
</form>
```

Gambar 1. Tag form HTML

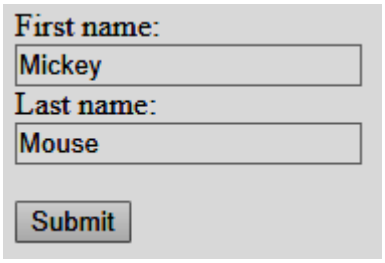
Unsur-unsur di dalam form tersebut bentuk berbagai jenis elemen input, seperti text box, checkbox, tombol radio, tombol submit, dan lain-lain.

<input> Elemen

<input> merupakan bentuk yang paling penting untuk membuat form HTML, **<input>** dapat ditampilkan dalam beberapa cara, tergantung pada jenis atribut yang dipakai. Berikut adalah beberapa contoh:

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>
```

Akan dihasilkan antarmuka grafis seperti Gambar 2 di bawah ini:



Gambar 2. Tampilan form HTML pada Browser

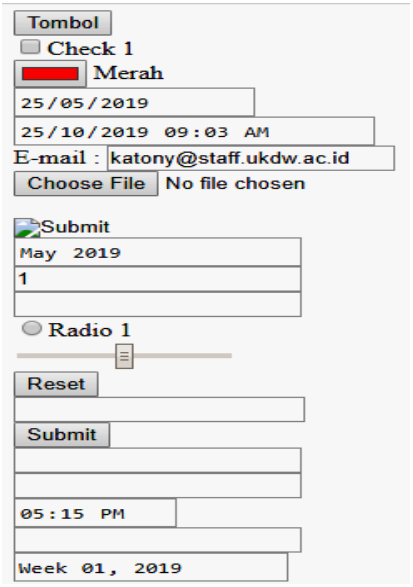
Pada dokumen HTML di atas tampak bahwa atribut pada tag **<form>** yaitu **action** menentukan tujuan ketika tombol Submit ditekan, yaitu ke dokumen bernama **"action_page.php"**.

Pada tag input atribut **type** menentukan jenis kontrol form yang akan ditampilkan, pada contoh di atas bernilai **text** artinya kontrol form akan berupa **text box** seperti pada Gambar 2 di atas. Atribut **name** digunakan sebagai pengenal variabel global **$_POST** ketika data form diterima pada dokumen penerima, pada contoh di atas teksbox pertama akan dikenal pada **$_POST["firstname"]** dan yang kedua pada **$_POST["lastname"]**.

Tag input selengkapnya adalah sebagai berikut:

```
<input type="button" value="Tombol"><br>
<input type="checkbox">Check 1<br>
<input type="color" value="#FF0000"> Merah<br>
<input type="date" value="2019-05-25"><br>
<input type="datetime-local"><br>
E-mail : <input type="email"><br>
<input type="file"><br>
<input type="hidden"><br>
<input type="image"><br>
<input type="month"><br>
<input type="number"><br>
<input type="password"><br>
<input type="radio">Radio 1<br>
<input type="range"><br>
<input type="reset"><br>
<input type="search"><br>
<input type="submit"><br>
<input type="tel"><br>
<input type="text"><br>
<input type="time"><br>
<input type="url"><br>
<input type="week"><br>
```

Bentuk tampilan yang dihasilkan adalah seperti Gambar 3 di bawah ini:



Gambar 3. Kontrol form dari tag-tag **input**

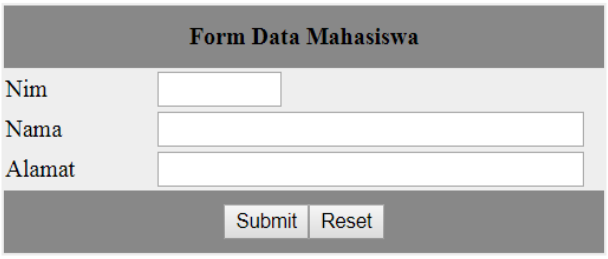
JUTEI Edisi Volume.3 No.1 April 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2018.31.147

- Mengatur Form[2]

Setiap elemen form yang ditampilkan akan menyambung dengan elemen sebelumnya sehingga tampilan form kurang rapi dan tidak mudah dipahami. Form dapat dikombinasikan dengan tag **<table>** untuk mengatur bentuk form agar rapi dan mudah dipahami. Seperti contoh berikut ini adalah form HTML yang dikombinasikan dengan **<table>** :

```
<TABLE Border=0 bgcolor="#EEEEEE">
<FORM METHOD=POST ACTION="Simpan.php">
  <TR bgcolor="#888888" height=40>
  <TH colspan=2>Form Data Mahasiswa</TH></TR>
  <TR><TD width=100>Nim</TD><TD width=300>
  <INPUT TYPE="text" NAME="nim" SIZE="8"></TD></TR>
  <TR><TD width=100>Nama</TD><TD width=300>
  <INPUT TYPE="text" NAME="nama" SIZE="40"></TD></TR>
  <TR><TD width=100>Alamat</TD><TD width=300>
  <INPUT TYPE="text" NAME="alamat" SIZE="40"></TD></TR>
  <TR bgcolor="#888888" height=40><TH colspan=2>
  <INPUT TYPE="submit"><INPUT TYPE="reset"></TH></TR>
</FORM></TABLE>
```

Form HTML tersebut akan menghasilkan tampilan pada Browser seperti Gambar 4 di bawah ini:



Gambar 4. Form HTML dikombinasikan dengan tabel

B. *Tabel* [3]

Tabel adalah tempat penyimpanan data berbentuk baris-baris dan kolom-kolom. Baris-baris menyimpan niai suatu entitas sedangkan kolom-kolom merupakan atribut yang menjelaskan identitas entitas tersebut.

Pada *Database Management System* (DBMS) kolom-kolom suatu tabel mempunyai tipe-tipe data terentu, pada RDBMS MySQL misalnya mempunyai tipe-tipe data kolom: TINYINT, SMALLINT, MEDIUMINT, INT, INTEGER, BIGINT, FLOAT, DOUBLE, DECIMAL, REAL, NUMERIC, DATE, DATETIME, TIMESTAMP, TIME, YEAR, VARCHAR dan lain-lain.

Untuk mendapatkan informasi mengenai kolom-kolom beserta tipe datanya digunakan perintah `DESCRIBE` atau `DESC` **<nama_tabel>**, sebagai contoh misalnya

```
MariaDB [formgenerator]> desc mahasiswa;
+-----------+-------------+------+-----+---------+
| Field     | Type        | Null | Key | Default |
+-----------+-------------+------+-----+---------+
| nim       | char(8)     | NO   | PRI | NULL    |
| nama      | varchar(40) | YES  |     | NULL    |
| alamat    | varchar(40) | YES  |     | NULL    |
| kota      | varchar(30) | YES  |     | NULL    |
| tgl_lahir | date        | YES  |     | NULL    |
| gender    | bit(1)      | YES  |     | NULL    |
| agama     | varchar(15) | YES  |     | NULL    |
+-----------+-------------+------+-----+---------+
7 rows in set (0.03 sec)
```

Daftar nama kolom dan tipe data inilah yang nanti akan dibutuhkan untuk membangkitkan Form HTML.

C. *Bahasa pemrograman PHP*

PHP merupakan bahasa pemrograman portabel yang sederhana, cepat, sangat sesuai untuk pengembangan aplikasi web yang melibatkan basis data.

- Struktur Kendali[4]

Struktur kendali pada PHP sama dengan struktur kendali pada bahasa C++, Java atau C#, pada penelitian ini memakai perintah percabangan dan perulangan yaitu :

- Percabangan if

  Bentuk umum :

```
if(kondisi){
   statment1;
} else {
   statments2;
}
```

- Perulangan while

  Bentuk umum :

```
while(kondisi){
   statments;
   increment/ decrement counter;
}
```

- Variabel[5]

Variabel pada PHP dapat mengenali tipe data sesuai dengan nilai yang disimpan pertama kali, setiap variabel PHP diawali dengan karakter dolar ($) tanpa perlu didekla-rasikan tipe datanya. Secara umum ada 3 asal usul variabel, yaitu : dari dalam *script* (kode program), diterima dari halaman HTML atau dari dalam lingkungan PHP.

- Variabel dalam kode program (*script*)

  Untuk membuat variabel ini cukup dengan menuliskan nama variabel yang diawali dengan karakter dolar ($) dan diberi nilai, secara otomatis variabel tersebut mengenal tipe data sesuai dengan nilai yang diberikan. Contoh:

```
$a = "ini adalah string"; //<-- data string
$b = 100; //<-- data integer
$c = 1.257; //<-- data floating point
```

  Tipe array juga dapat langsung diberikan pada variabel, misalnya :

```
$hari= array("Senin","Selasa","Rabu");
for($i=0 ; i<3 ; i++)
     echo "Hari : ".$hari[$i]."<br>";
```

- Variabel dari halaman HTML

  Untuk mengirimkan data dari halaman HTML dapat digunakan Form HTML, setiap tag <input> atau

kontrol form lain pasti mempunyai atribut "name" yang merupakan pengenal. Sebagai contoh misalnya Form HTML di bawah ini:

```
<form action="terima.php">
    <input type="text" name="kota">
    <input type="submit" value="Kirim">
</form>
```

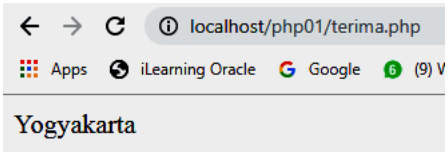
Akan menampilkan form pada browser seperti Gambar 5 di bawah ini:

Gambar 5. Form HTML dengan sebuah Text Box

Apabila pada text box diisi suatu nilai misalnya "`Yogyakarta`" kemudian ditekan tombol "`Kirim`", maka data akan diterima pada halaman web bernama "`terima.php`" dan cara menerimanya memakai variabel array global `$_POST` seperti berikut :

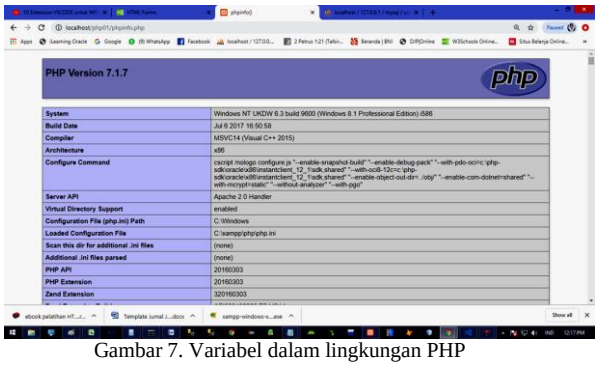
```
<?php
    echo $_POST["kota"];
?>
```

Maka pada browser akan ditampilkan isi dari text box yang ada pada form tadi, yaitu seperti Gambar 6 di bawah ini:



Gambar 6. Halaman web "`terima.php`" menampilkan isi variabel `$_POST` dengan id "`kota`"

- Variabel lingkungan PHP
  Ada berbagai macam variabel yang diatur oleh server dan lingkungan PHP, secara lengkap dapat dilihat melalui metode **`phpinfo()`**, seperti berikut:



Gambar 7. Variabel dalam lingkungan PHP

Pada penelitian ini digunakan variabel Apache untuk mengenali lokasi dokumen pada server, yaitu variabel Apache: **`$_SERVER["DOCUMENT_ROOT"]`**, sebagai contoh misalnya untuk menyertakan halaman web "`connect.php`" dapat digunakan perintah:

```
Require_once
$_SERVER["DOCUMENT_ROOT"]."/php01/connect.php";
```

D. Pemrograman Berorientasi Objek

Dalam membuat halaman web, khususnya form HTML bisa dilakukan dengan menyalin dan mengubah dokumen tersebut untuk disesuaikan dengan kebutuhan, namun cara tersebut tidak praktis dan kontraproduktif. Akan jauh lebih baik apabila menulis kode sekali saja dan menggunakan bahasa skrip sisi server tersebut untuk menyisipkan form HTML di mana pun dibutuhkan.

Dengan kata lain "sertakan dan gunakan kembali, jangan menulis ulang". Pemrograman berorientasi objek (OOP) merupakan pewujudan dari konsep ini. Objek menyederhanakan pengembangan web dengan menghindari pekerjaan memotong, menempel, dan mengadaptasi kode yang ada. [6]

Untuk mewujudkan paradigma Pemrograman Berorientasi Objek, diperlukan suatu bentuk yang dinamakan Kelas (*Class*) untuk membuat objek. Di dalam kelas terdapat konstruktor yang bermanfaat untuk melakukan inisialisasi data yang ada dalam kelas, selain itu program-program yang bekerja di dalam kelas ditulis di dalam suatu sub rutin yang disebut dengan metode. [6]

- Kelas (*Class*)
  Pada prinsipnya adalah suatu tipe data, tetapi berbeda dengan tipe data primitif seperti misalnya int, double, float dan lain-lain, karena di dalam kelas dapat mempunyai anggota banyak variabel dan banyak metode (sub rutin), jadi kelas adalah suatu bentuk yang serupa dengan rekaman (*record*) yang mengenkapsulasi karakteristik (menjaga kondisi) dari suatu objek. Pada PHP bentuk kelas adalah seperti berikut ini:

```
<?php
    class <nama_kelas>{
        <konstruktor>
        <variabel_anggota>
        <metode>
    }
?>
```

- Konstruktor
  Konstruktor merupakan bagian kelas yang dieksekusi ketika kelas tersebut diinstansiasi (menjadi objek), sehingga konstruktor sangat bermanfaat untuk memberikan nilai awal (inisialisasi) nilai-nilai dari member variable. Pada umumnya konstruktor mempunyai nama yang sama dengan nama kelasnya, namun PHP mengubahnya dari :

```
function <nama_kelas>(<parameter>){ ... }
```

  menjadi :

```
public function __construct(<parameter>){ ... }
```

- Metode
  Metode merupakan tempat untuk menuliskan program, melalui metode-metode inilah kelas melakukan enkapsulasi untuk menjaga kondisi objek (mengenkapsulasi karakteristik objek).

Berikut ini adalah contoh sebuah kelas untuk membuat Form dengan konstruktor yang digunakan untuk menentukan judul Form dan lokasi halaman web penerima Form.

JUTEI Edisi Volume.3 No.1 April 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2018.31.147

```
<?php
class Form{
  private $action;
  private $judul;
  private $kontrol=array();

  function __construct($act,$jud){
    $this->action = $act;
    $this->judul = $jud;
  }

function addTextBox($nama,$nilai,$lebar){
  $this->kontrol[$nama] =
      $this->textBox($nama,$nilai,$lebar);
}

function textBox($nama){
  $h="<INPUT TYPE=\"text\" NAME=\"$nama\">";
      return $h;
  }

function showForm(){
  echo "<FORM ACTION=\"$this->action\">\n";
  echo "<TABLE>\n";
  echo "<TR><TH colspan=2>";
  echo $this->judul."</TH></TR>\n";
  foreach($this->kontrol as $field => $fc){
    echo "<TR><TD>".($field);
    echo "</TD><TD>$fc</TD></TR>\n";
  }
  echo "<TR><TH colspan=2>";
  echo "<INPUT TYPE=\"submit\">
        <INPUT TYPE=\"reset\"></TH></TR>\n";
  echo "</FORM>\n";
  }
}
?>
```

Program 1. Contoh kelas Form sederhana

Untuk memakai kelas tersebut dapat dibuat program, misalnya dibuat 3 buah text box seperti berikut:

```
<?php
    Require_once "Form.php";
    $Form = new Form("Simpan.php","Data Teman");
    $Form->addTextBox("Nama");
    $Form->addTextBox("Alamat");
    $Form->addTextBox("Kota");
    $Form->showForm();
?>
```

Program 2. Contoh pemakaian kelas Form

Program itu akan menghasilkan Form HTML dalam bentuk tag HTML seperti di bawah ini:

```
<FORM ACTION="Simpan.php">
<TABLE Border=0 bgcolor="#EEEEEE">
<TR><TH colspan=2>Data Teman</TH></TR>
<TR><TD>Nama</TD>
    <TD><INPUT TYPE="text" NAME="Nama"></TD></TR>
<TR><TD>Alamat</TD>
    <TD><INPUT TYPE="text" NAME="Alamat"></TD></TR>
<TR><TD>Kota</TD><TD>
    <INPUT TYPE="text" NAME="Kota"></TD></TR>
<TR><TH colspan=2><INPUT TYPE="submit">
    <INPUT TYPE="reset"></TH></TR>
</FORM>
```

Untuk dapat bekerja dengan OOP harus ada tiga karakteristik objek, yaitu :

- Perilaku objek (*behaviour*) - apa yang bisa dilakukan objek atau metode apa yang bisa dilakukan terhadap objek.
- Status objek (*state*) - bagaimana reaksi objek ketika suatu metode dikerjakan.
- Identitas objek (*identity*) - bagaimana satu objek dapat dibedakan dengan objek yang lainnya.[7]

Objek adalah bentuk kongrit (*bulding blocks*) dari sebuah program Berorientasi Objek. Sebuah program yang memakai teknologi Berorientasi Objek pada umumnya terbentuk dari kumpulan objek-objek lainnya.[8]

Suatu pesan adalah sah (*valid*) jika penerimanya mempunyai metode yang berhubungan dengan nama metode dalam pesan tersebut dengan argumen (parameter) yang sesuai apabila ada. Hanya pesan yang sah yang akan dieksekusi oleh penerima pesan.[9]

Pemrograman berorientasi objek, terkadang disingkat OOP, bukan hanya masalah menggunakan sintaks yang berbeda. Ini cara berbeda untuk menganalisis masalah pemrograman. Aplikasi ini dirancang dengan memodelkan masalah pemrograman. Dalam pemrograman berorientasi objek, elemen skrip adalah objek. Objek mewakili elemen masalah yang ingin diselesaikan skrip.[10]

III. METODOLOGI PENELITIAN

Langkah pertama adalah mempelajari cara membangun Form HTML secara umum yaitu bentuk dasarnya maupun kontrol-kontrol form yang tersedia, setelah itu dilanjutkan dengan perancangan kelas untuk membuat form beserta kontrol-kontrol *form*nya. Dari sisi data yang akan disimpan melalui Form HTML dipelajari hubungan antara tipe data dengan kontrol form yang ada kemudian mengimplementasikan semuanya menjadi Generator Form HTML berbasis tabel.

A. Form HTML

From HTML terbentuk dari pasangan tag `<form>` dengan atribut-atribut penting : **`action`** dan **`method`**, komponen di dalam *form* dapat berupa tag-tag input untuk memasukkan data dan yang terakhir adalah tombol submit yang digunakan untuk mengirimkan semua data yang ada ke halaman web tertentu.

Dalam sebuah form terdapat banyak kontrol-kontrol *form* yaitu berupa: *text box*, *combo box*, *spinner*, *text area* dan lain-lain namun semua bentuk tersebut mempunyai tag yang sama yaitu `<input>`. Yang membedakan antara kontrol form satu dengan yang lain adalah nilai dari atribut **`type`**, sedangkan identitas yang digunakan untuk menyimpan data adalah atribut **`name`**.

Supaya form dapat dikirimkan ke tujuan, di dalam form harus ada tag `<input>` dengan `type="submit"`. Berikut ini contoh sebuah form dengan sebuah Text box yang dapat dikirimkan ke program bernama "`simpan.php`":

```
<form action="simpan.php" method="POST">
   <input type="text" name="nama_mahasiswa">
   <input type="submit">
</form>
```

Gambar 8. Tampilan Form pada Browser

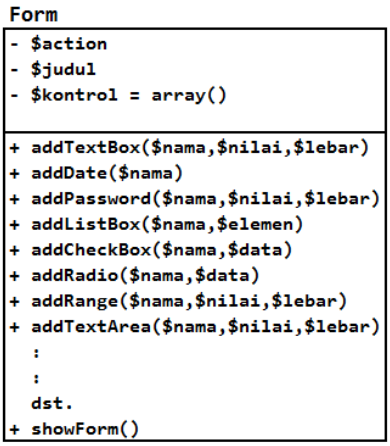
Karena di dalam sebuah form terdapat banyak kontrol-kontrol form, maka diperlukan variabel bertipe array untuk menyimpan tag-tag **<input>** tersebut.

Ada berbagai macam tag yang tersedia, pada prinsipnya sama-sama memakai tag <input> hanya saja mempuyai atribut type yang nilainya berbeda. Pada HTML 5 terdapat berbagai macam, diantaranya yaitu :

```
<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
<input type="text">
<input type="time">
<input type="url">
<input type="week">
```

B. Rancangan Kelas Form

Supaya form HTML bersifat fleksibel bisa terdiri dari berbagai jumlah atribut maka dalam rancangan kelas Form akan disediakan sebuah member variabel bertipe array. Isi dari array ini berupa teks sehingga dapat memuat berbagai tag **<input>** seperti contoh di atas. Berikut ini adalah diagram kelas Form :



Gambar 9. Diagram Kelas Form

Member variable **$action** digunakan untuk menentukan nama halaman web yang akan menerima data form, **$judul** adalah teks yang akan ditampilkan pada antarmuka grafis form di bagian atas supaya pemakai mengetahui manfaat antarmuka grafis ini, sedangkan **$kontrol** adalah berupa array karena akan digunakan untuk menyimpan banyak kontrol-kontrol form.

Metode AddTextBox, addDate, addPassword dan seterusnya digunakan untuk menambah elemen *array* **$kontrol** yang tiap elemennya bisa berisi salah satu dari tag <input> tipe tertentu, sehingga banyaknya kontrol form dan jenisnya fleksibel sesuai dengan metode yang dipanggil.

Metode showForm digunakan untuk menampilkan Form HTML, yaitu kerangka form <form> dan di dalamnya kontrol-kontrol form yang sudah ditambahkan.

C. Tipe data atribut tabel dan jenis kontrol form

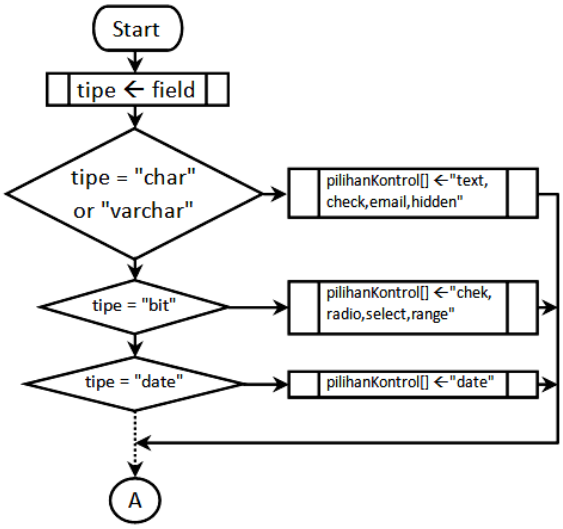
MySQL mempunyai berbagai macam tipe data untuk field-fieldnya, tiap tipe data bisa mempunyai kemungkinan beberapa bentuk kontrol form, diantaranya misalnya :

Tipe Data	Kontrol Form
Char	<input type="checkbox">
Varchar	<input type="email">
	<input type="hidden">
Bit	<input type="checkbox">
	<input type="radio">
	<select>
	<input type="range">
Date	<input type="date">
Text	<textarea>

Gambar 10. Tipe data dan jenis kontrol form

D. Implmenentasi

Agar form dapat dibuat berdasarkan tabel tertentu, pertama kali harus baca struktur data dari tabel yang dimaksud, kemudian sesuai dengan tipe datanya akan ditambahkan kontrol form yang sesuai dengan cara memanggil metode yang sesuai misalnya addTextBox, addRadio, dan sebagainya. Namun karena untuk satu tipe data ada beberapa kemungkinan kontrol form, maka sebelumnya akan disediakan menu pilihan seperti berikut:



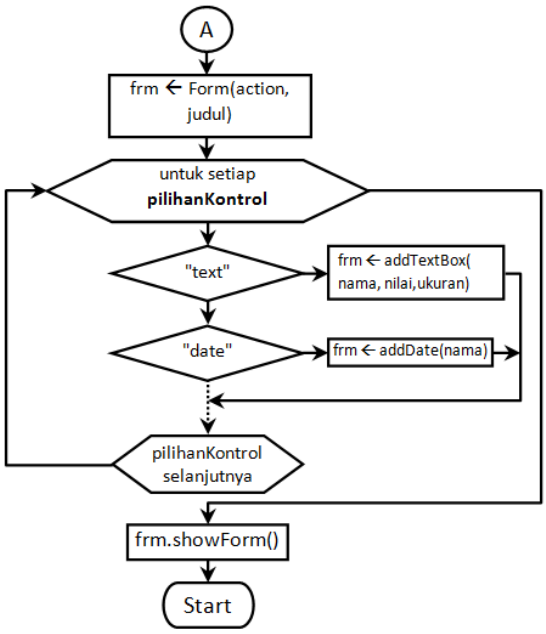
Gambar 11. Memetakan tipe data ke jenis kontrol form

Gambar 11 di atas adalah pemetaan jenis kontrol form berdasarkan tipe data yang ada, selanjutnya pemakai akan

JUTEI Edisi Volume.3 No.1 April 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2018.31.147

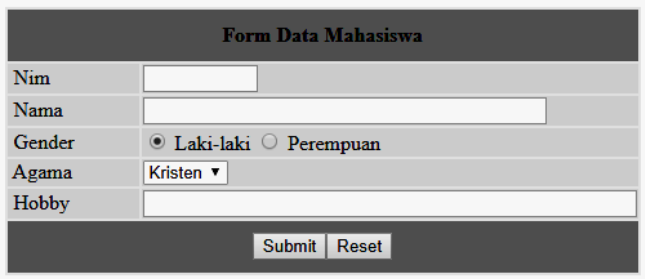
memilih kontrol form yang disarankan untuk setiap field dari tabel tersebut. Untuk menghemat ruang pada naskah ini tidak semua tipe data digambarkan dalam flowchart, garis putus-putus menggambarkan bahwa masih ada banyak proses yang serupa yang pada intinya bahwa untuk setiap tipe data tertentu disediakan pilihan kemungkinan kontrol form yang sesuai.

Selanjutnya setelah ditentukan jenis kontrol formnya, maka akan dipanggil metode-metode untuk menambahkan objek kontrol form pada objek form, misalnnya addTextBox, addRadio dan sebagainya seperti gambar flowchart di bawah ini :



Gambar 12. Membuat objek Form dan menambahkan kontrol form

Gambar 12 di atas merupakan lanjutan dari flowchart Gambar 11, setelah diperoleh data mengenai kontrol-kontrol form yang dibutuhkan maka dibuatlah objek Form dengan parameter **`action`** (halaman web penerima) dan **`judul`** (judul yang akan ditampilkan pada form HTML). Kemudian semua pilihan kontrol akan dibaca satu persatu untuk menentukan jenis kontrol yang akan ditambahkan sampai semua kontrol sudah ditambahkan. Setelah semua kontrol selesai ditambahkan kemudian form HTML akan ditampilkan, sebagai contoh misalnya hasil dari pembuatan form HTML dari tabel mahasiswa adalah seperti berikut:



Gambar 13. Form HTML yang dihasilkan

IV. HASIL DAN ANALISIS

Inti dari generator form HTML ini adalah kelas Form, yaitu kelas yang dapat menentukan judul form, tujuan form (***action***), menambahkan kontrol-kontrol dan menampilkan form. Untuk memakai kelas ini dibutuhkan serangkaian proses sebelumnya, yaitu memilih tabel, membaca tipe data, ukuran dan menyediakan alternatif pilihan kontrol form.

A. Kelas Form

Kelas ini mempunyai pengenal berupa judul dan halaman web yang dituju serta memiliki anggota-anggota berupa kontrol-kontrol form.

Member variable **`$judul`** merupakan teks biasa yang akan digunakan untuk keterangan pada tampilan form HTML, **`$action`** juga berupa teks yang isinya adalah nama file halaman web yang dituju, sedangkan *member variable* **`$kontrol`** bertipe array karena untuk setiap kontrol form akan disimpan dalam sebuah elemen array ini.

Berikut ini adalah *member variable* untuk kelas Form yang dapat dipergunakan untuk membangun form HTML.

```
class Form{
  private $action;
  private $judul;
  private $kontrol=array();

  function __construct($act,$jud){
    $this->action = $act;
    $this->judul = $jud;
    }
}
```

Program 3. Kelas Form dengan 3 *member variable*

Konstruktor kelas ini membutuhkan parameter untuk `action` dan `judul` supaya pemakai tidak lupa untuk memberi-tahukan nilai-nilainya.

Untuk menambahkan kontrol-kontrol form ke dalam kelas ini digunakan metode-metode yang diberi nama awalan "`add`", sebagai contoh misalnya `addTextBox`. Seperti kita ketahui bahwa setiap kontrol form mempunyai atribut-atribut penting agar tag **`<input>`** tersebut dapat dipergunakan sebagaimana mestinya, untuk itu pada tiap metode perlu disediakan parameter-parameter yang dibutuhkan. Sebagai contoh misalnya pada metode `addTextBox()` dibutuhkan parameter **`$nama`** untuk pengenal kontrol form, **`$nilai`** untuk nilai awal (*default value*) apabila dibutuhkan dan **`$lebar`** untuk menentukan lebar dari textbox yang akan dihasilkan seperti berikut:

```
function addTextBox($nama,$nilai,$lebar){
    $this->kontrol[$nama] =
      $this->textBox($nama,$nilai,$lebar);
}

private function textBox($nama,$nilai,$lebar){
    $h="<INPUT TYPE=\"text\" NAME=\"$nama\"
      VALUE=\"$nilai\" SIZE=\"$lebar\">";
    return $h;
}
```

Program 4. Metode **`addTextBox($nama,$nilai,$lebar)`**

Demikian juga untuk kontrol-kontrol form yang lain perlu disediakan metode untuk membuatnya sesuai dengan karakteristiknya masing-masing. Kontrol form ListBox membutuhkan lebih dari sebuah tag HTML, maka bentuk

metodenya akan sedikit berbeda dengan kontrol form TextBox yang hanya memerlukan sebuah tag **<input>**.

```
function addListBox($nama,$data){
    $this->kontrol[$nama] =
      $this->listBox($nama,$data);
}
private function listBox($nama,$elemen){
  $hasil = "<SELECT NAME=\"$nama\">\n";
  foreach($elemen as $lst)
    $hasil .= "<OPTION>$lst</OPTION>\n";
  $hasil .= "</SELECT>";
  return $hasil;
}
```

Program 5.a. Metode add yang membutuhkan banyak tag (bersambung)

```
function addCheckBox($nama,$data){
  $this->kontrol[$nama] =
    $this->checkBox($nama,$data);
}
private function checkBox($nama,$elemen){
  $hasil = "";
  foreach($elemen as $lst)
  $hasil .= "<INPUT TYPE=\"checkbox\"
    NAME=\"$nama"."[]\" VALUE=\"$lst\">$lst\n";
    return $hasil;
}
function addRadio($nama,$data){
  $this->kontrol[$nama] =
    $this->radio($nama,$data);
}
private function radio($nama,$elemen){
  $hasil ="";
  foreach($elemen as $lst)
    $hasil .= "<INPUT TYPE=\"radio\"
      NAME=\"$nama\" VALUE=\"$lst\"> $lst \n";
  return $hasil;
}
```

Program 5.b. Metode add yang membutuhkan banyak tag (lanjutan)

Metode yang terakhir adalah metode untuk menghasilkan form tersebut, yaitu dengan cara menuliskan tag **<form>** beserta atribut dan anggota-anggotanya dengan metode **showForm()** berikut ini:

```
function showForm(){
  echo "<CENTER><FORM METHOD=POST ACTION=
    \"$this->action\">\n";
  echo "<TABLE Border=0
    bgcolor=\"#EEEEEE\">\n";
  echo "<TR bgcolor=\"#888888\" height=40>
    <TH colspan=2>";
  echo ucwords(strtolower($this->judul)).
    "</TH></TR>\n";
  foreach($this->kontrol as $field => $fc){
    echo "<TR bgcolor=\"#E0E0E0\">
      <TD width=100> ".ucwords($field);
    echo "</TD><TD width=300>$fc</TD></TR>\n";
  }
  echo "<TR bgcolor=\"#888888\" height=40>
    <TH colspan=2>";
  echo "<INPUT TYPE=\"submit\">
    <INPUT TYPE=\"reset\"></TH></TR>\n";
  echo "</FORM></CENTER>\n";
}
```

Program 6. Metode **showForm()** untuk menampilkan

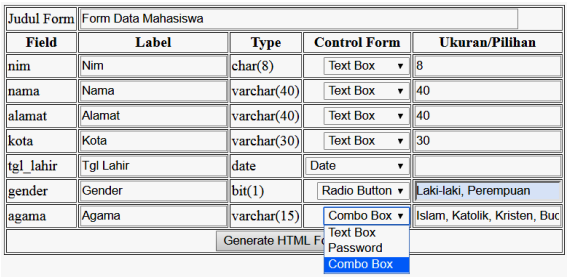
B. Pemetaan tipe data menjadi kontrol form

Setelah tabel dipilih, maka perlu ditentukan jenis-jenis kontrol form yang mungkin diperlukan, sebagai contoh misalnya tabel mahasiswa yang mempunyai deskripsi struktur seperti berikut:

#	Name	Type
1	nim	char(8)
2	nama	varchar(40)
3	alamat	varchar(40)
4	kota	varchar(30)
5	tgl_lahir	date
6	gender	bit(1)
7	agama	varchar(15)

Gambar 14. Deskripsi struktur tabel **mahasiswa**

Maka untuk menentukan kontrol formnya disediakan alternatif yang sesuai, misalnya untuk tipe data varchar kontrol form yang memungkinkan adalah **Text Box**, **Password** atau **Combo Box**, seperti misalnya field **agama** pada tabel **mahasiswa** berikut ini:



Gambar 15. Alternatif pilihan kontrol form untuk tiap field

Dengan bahasa pemrograman PHP alternatif dalam bentuk tampilan List Box seperti gambar tersebut dapat dibuat memakai fungsi di bawah ini:

```
function kontrol($type){
  $tipe = strtolower(substr($type,0,3));
  $kontrol = "<option>$tipe</option>";
  if(strpos(" chavar",$tipe) > 0){
    $kontrol = "<option value=text size=80>
      Text Box</option>\n";
    $kontrol .= "<option value=password size=80>
      Password</option>\n";
    $kontrol .= "<option value=select size=80>
      Combo Box</option>\n";
  }

  if(strpos(" dat",$tipe) > 0){
    $kontrol = "<option value=date>
      Date</option>\n";
    $kontrol .= "<option value=datetime-local>
      Datetime-Local</option>\n";
  }

  if(strpos(" bit",$tipe) > 0){
    $kontrol = "<option value=radio>
      Radio Button</option>\n";
    $kontrol .= "<option value=select>
      Combo Box</option>\n";
  }
  return $kontrol;
}
```

Program 7. Fungsi membuat ListBox pilihan kontrol form

JUTEI Edisi Volume.3 No.1 April 2019
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2018.31.147

Pemetaan ini sesuai dengan rancangan program yang ditampilkan dalam bentuk flowchart pada Gambar 5, namun untuk menghemat ruang tidak semua kemungkinan dibuat programnya, untuk kemungkinan-kemungkinan lain dapat ditambahkan pada fungsi **kontrol** tersebut.

C. Membangkitkan Form HTML

Berdasarkan pilihan kontrol tersebut maka dapat dibangkitkan form HTML berdasarkan pilihan-pilihan tersebut seperti rancangan Gambar 6. Membuat objek Form dan menambahkan kontrol form seperti berikut:

```
Require_once
$_SERVER["DOCUMENT_ROOT"]."/php01/Form.php";

$bentuk = array();
foreach ($_POST as $key => $value) {
    $bentuk[$key] = $value;
}
```

Program 8.a. Program membangkitkan form (bersambung)

```
foreach ($_POST as $key => $value) {
  if($key=="judul")
    $Form = new Form("Simpan.php","$value");
  else if ($value=="text") {
    $size = $bentuk["bentuk".$key];
    $Form->addTextBox("$key","",$size);
  } else if ($value=="radio") {
    $pilihan = explode(",",$bentuk["bentuk".$key]);
    $Form->addRadio("$key",$pilihan);
  } else if ($value=="range")
    $Form->addRange("$key","",15);
  else if ($value=="select") {
    $pilihan = explode(",",$bentuk["bentuk".$key]);
     $Form->addListBox("$key",$pilihan);
  } else if ($value=="date") {
    $Form->addDate("$key");
   }
}
$Form->showForm();
```

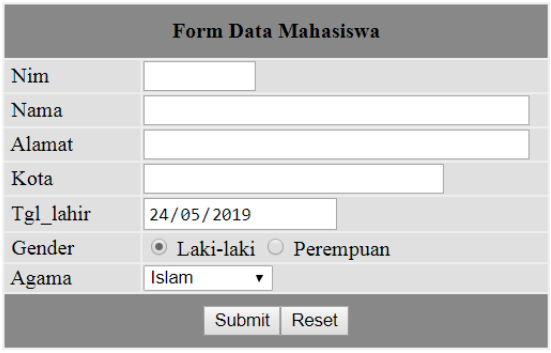
Program 8.b. Program membangkitkan form (lanjutan)

Untuk memudahkan pengambilan nilai ukuran/pilihan seperti tampak pada Gambar 15, maka dibuat variabel array **`$bentuk`** seperti tampak pada Program 8.a, dengan kunci nama field dari tiap tabel, tampak pada program di atas, perulangan `foreach` pertama adalah menyediakan variabel array tersebut.

Semua metode-metode untuk menambahkan kontrol form membutuhkan nama sebagai identitas kontrol form, yaitu untuk attribut **`name`** pada tag **`input`**, untuk keperluan ini dapat digunakan nama *field* dari tabel. Untuk atribut lainnya, misalnya **`size`** (untuk **text**) dan elemen pilihan pada Radio Button dan Check Box dapat nilai dari variabel array `$bentuk` dengan kunci memakai nama field tersbut. Sebagai contoh misalnya pada pemetaan form sesuai Gambar 15, variabel array **`$bentuk["nim"]`** akan mempunyai nilai **"8"**, sedangkan variabel array **`$bentuk["agama"]`** akan mempunyai nilai sebuah string **"Islam, Katolik, Kristen, Budha, Hindu, Kong Hu Cu"**.

Pada metode **addTextBox**, nilai variabel `$bentuk` tersebut akan digunakan sebagai ukuran (**size**) dari **TextBox**, sedangkan pada metode **addListBox** nilai variabel bentuk perlu di-explode untuk dijadikan array yang dikirim sebagai parameter yang menentukan pilihan yang ada pada **ListBox** tersebut.

Pada contoh kasus ini form HTML yang dibangkitkan berdasarkan parameter yang dipilih seperti Gambar 9 adalah seperti berikut:



Gambar 16. Form HTML yang dihasilkan oleh generator

Adapun kode sumber dari form HTML di atas dihasilkan oleh program PHP khususnya pada metode `showForm()`. Form HTML yang dihasilkan dapat dilihat dengan melihat kode sumber yang ada pada browser dengan cara klik kanan pada browser kemudian pilih menu *View Page Source* yang hasilnya adalah seperti tampak pada Gambar 17 berikut ini:

```
<CENTER><FORM METHOD=POST ACTION="Simpan.php">
<TABLE Border=0>
<TR height=40><TH colspan=2>Form Data Mahasiswa</TH></TR>
<TR ><TD width=100>Nim</TD><TD width=300><INPUT TYPE="text" NAME="nim" VALUE="" SIZE="8"></TD></TR>
<TR ><TD width=100>Nama</TD><TD width=300><INPUT TYPE="text" NAME="nama" VALUE="" SIZE="40"></TD></TR>
<TR ><TD width=100>Alamat</TD><TD width=300><INPUT TYPE="text" NAME="alamat" VALUE="" SIZE="40"></TD></TR>
<TR ><TD width=100>Kota</TD><TD width=300><INPUT TYPE="text" NAME="kota" VALUE="" SIZE="30"></TD></TR>
<TR ><TD width=100>Tgl_lahir</TD><TD width=300><INPUT TYPE="date" NAME="tgl_lahir" value="2019-05-24"></TD></TR>
<TR ><TD width=100>Gender</TD><TD width=300><INPUT TYPE="radio" NAME="gender" VALUE="Laki-laki"> Laki-laki
<INPUT TYPE="radio" NAME="gender" VALUE=" Perempuan"> Perempuan
</TD></TR>
<TR ><TD width=100>Agama</TD><TD width=300><SELECT NAME="agama">
<OPTION>Islam</OPTION>
<OPTION> Katolik</OPTION>
<OPTION> Kristen</OPTION>
<OPTION> Budha</OPTION>
<OPTION> Hindu</OPTION>
<OPTION> Kong Hu Cu</OPTION>
</SELECT></TD></TR>
<TR bgcolor="#888888" height=40><TH colspan=2><INPUT TYPE="submit"><INPUT TYPE="reset"></TH></TR>
</FORM></CENTER>
```

Gambar 17. Tag Form HTML yang dihasilkan

V. KESIMPULAN

Form HTML dapat dipandang sebagai sebuah objek dengan anggota-anggotanya berupa kontrol-kontrol form halaman web tujuan form dan judul form merupakan field/ variabel anggota dari kelas Form.

Kontrol-kontrol form HTML berupa tag-tag input dapat disimpan sebagai array of string, dalam hal ini variabel anggota **`$kontrol`**, yang elemen-elemennya ditambahkan melalui metode-metode.

Dalam menciptakan kontrol form, tiap kontrol form mempunyai parameter yang berbeda-beda sesuai dengan atribut kontrol form tersebut, hal ini dapat diatur dengan adanya parameter-parameter pada metode untuk menambah kotrol form.

Metode **`showForm()`** dapat membangun form HTML dengan cara membuat bentuk form HTML dan di dalamnya tag-tag input yang tersimpan dalam variabel array $kontrol.

Dengan Object Oriented Programming pembuatan form HTML menjadi lebih mudah karena sudah disediakan metode-metode untuk menambahkan kontrol form.

Dengan membaca struktur dan tipe data suatu tabel dan kelas Form dapat dibangun form HTML secara otomatis namun perlu adanya sedikit campur tangan pengguna mengingat suatu tipe data tertentu dapat dibuat beberapa pilihan kontrol form.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih yang sebesar-bearnya kepada Universitas Kristen Duta Wacana Yogyakarta, khususnya Program Studi Sistem Informasi Fakultas Teknologi Informasi atas fasilitas-fasilitas yang disediakan sehingga dapat terselesaikannya penelitian ini.

Penulis juga sangat berterima kasih kepada rekan-rekan dosen yang tidak jemu-jemu untuk membantu memberikan petunjuk dalam menyelesaiakan masalah penelitian ini.

Tidak lupa tentu saja saya sangat berterima kasih kepada segenap staf redaksi maupun tim *reviewer* yang telah memeriksa naskah penelitian saya, kiranya Tuhan yang maha pengasih menyertai dan memberkati kita semua.

DAFTAR PUSTAKA

[1] (2019) w3schools.com. [Online]. Tersedia: https://www.w3schools.com/

[2] Surfas M and friends,"Combining Forms with Table", dalam *Using Intranet HTML, Indianapolis, Indiana, 1996*, hal 546-547.

[3] HB, Detron,"Column Type", dalam *MySQL Reference Manual, Finland*, 2000, hal 157-160.

[4] Quigley E. and Gargenta M., "About PHP," dalam *PHP and MYSQL By Example*, edisi ke-*1*, Soughton, Massachusetts, 2007, hal.4, 222, 232.

[5] *Greenspan J and Bulger B*, "Variables", dalam *MySQL/PHP Databbase Application, Indianapolis, New York, M&T Books, 2001*, hal 71,77,79, 87.

[6] Lavin P,"What a tangled web we weave", dalam *OBJECT-ORIENTED PHP Concepts, Techniques, and Code, No Starch, San Fransico, 2006*, hal 1-2, 6, 28

[7] Horstmann C S,"Objects", dalam *Core Java Volume I-Fundamentals, Prentice Hall, Crawfordsville, Indiana, 2015*, hal 132

[8] Weisfield M, "What exacly is an Object?", dalam The *Object-Oriented Thought Process, Addison-WESLEY, United States of America, 2008*, hal 10

[9] Poo D and friends, "Method", dalam *Object-Oriented Programming and Java, Second edition, Springer, Verlag London, 2008*, hal 10

[10] Lavin P, "Introducing Object-Oriented Programming ", dalam *PHP, MySQL®, JavaScript® & HTML5 All-in-One For Dummies, John Wiley & Sons, Inc., New Jersey, 2013*, hal 397