Modul 7 _ Form

1.1 Waktu Pelaksanaan Praktikum

Durasi kegiatan praktikum adalah **170 menit,** dengan rincian sebagai berikut.

a. 15 menit untuk pengerjaan Tes Awal atau wawancara Tugas Pendahuluan
b. 60 menit untuk penyampaian materi
c. 45 menit untuk pengerjaan jurnal, tes akhir atau tugas
d. 50 menit **pengayaan**

1.2 Tujuan

Setelah mengikuti praktikum ini, mahasiswa diharapkan dapat:

1. Memahami mekanisme *form* di HTML.
2. Membuat *form* menggunakan HTML.
3. Mengolah data yang terkirim dari *form* HTML menggunakan PHP.
4. Menentukan metode *form* yang cocok untuk berbagai kasus.

1.3 Dasar Teori

a) Form

Form dalam HTML adalah suatu bagian yang berfungsi sebagai input atau masukan dari pengguna yang kemudian akan diproses atau diolah untuk dapat digunakan sesuai dengan kebutuhan. *Form* adalah antarmuka pengguna (*user interface*) agar pengguna dapat berkomunikasi dengan sistem aplikasi. Setiap permintaan dari pengguna disebut *request* dan setiap hasil yang dari pemrosesan oleh *server* disebut *response*.

Form pada HTML dibuat dengan menggunakan elemen *form*. Elemen *form* harus membungkus seluruh elemen-elemen masukan form lain untuk memastikan masukan informasi pengguna dapat dibaca oleh aplikasi *web*.

```
1  <form action="process.php" method="post">
2      ....
3  </form>
```

Pada *form* terdapat dua atribut yang wajib dimiliki oleh elemen *form*. Atribut pertama adalah "**action"** yang berfungsi untuk memberitahukan *browser* alamat pengiriman dari data yang diisikan pengguna di dalam *form*. Parameter ini diisikan dengan sebuah URL pada *server* yang melakukan pemrosesan data. Atribut kedua yang wajib diisikan adalah atribut "**method"**. Atribut ini memberitahukan *browser* dengan metode apa data akan dikirimkan ke *server* yang alamatnya diisikan pada atribut *action*. Terdapat dua nilai yang dapat diisikan pada atribut ini, yaitu "**GET"** dan "**POST"**.

a) ***Form Control***

Sebuah form berisi satu atau beberapa *form control*, antara lain *text field*, *text area*, *checkbox*, dan lain-lain. Setiap *form control* tersebut harus diberi nama agar dapat dibaca di sisi *server*. Tanpa nama, data yang dikirim tidak dapat karena setiap *form control* akan dibaca sebagai variabel yang berbeda di sisi *server*.

Pemberian nama tersebut dilakukan menggunakan atribut "name" pada masing-masing *form control*. Berikut contoh pemberian nama pada sebuah *text field*.

```
1  <input type="text" name="username">
```

Ada beberapa jenis *form control*, yaitu:

- Input teks
  Kategori input ini terdiri dari beberapa jenis, yaitu:
  - Input teks satu baris

```
1  <input type="text" name="...">
```

  - Input teks banyak baris (*text area*)

```
1  <textarea name="..."></textarea>
```

  - Input teks *password*
    Dengan form control ini, setiap karakter yang diketikkan diganti dengan karakter bintang (*) atau *bullet* (•) sehingga tidak terbaca oleh pengguna.

```
1  <input type="password" name="...">
```

  - Input teks pencarian (HTML5)

```
1  <input type="search" name="...">
```

  - Input teks *email* (HTML5)

```
1  <input type="email" name="...">
```

  - Input teks telepon (HTML5)

```
1  <input type="tel" name="...">
```

- Input teks URL (HTML5)

```
1   <input type="url" name="...">
```

Input teks pencarian, *email*, telefon, dan URL memiliki tampilan yang sama dengan input teks satu baris. Perbedaannya adalah pada tampilan *keyboard* saat dibuka di perangkat *mobile*. Saat menampilkan input teks pencarian, keyboard pada perangkat mobile akan menampilkan tombol bertuliskan "Search". Saat menampilkan input teks *email*, tombol karakter "@" akan ditampilkan. Saat menampilkan input telepon, keyboard hanya akan memberikan tampilan angka. Saat menampilkan input teks URL, akan tampil tombol bertuliskan ".com". Hal tersebut dilakukan untuk mempermudah pengguna memasukkan data pada form.

Selain itu, *browser* juga dapat melakukan validasi terhadap format teks yang diisikan. Sebagai contoh, jika pada input teks email, dimasukkan alamat *email* yang tidak valid, maka *browser* akan memberikan pesan kesalahan pada saat *form* di-*submit* dan mencegah data dikirim sebelum semua data yang dimasukkan memiliki format yang valid.

- Tombol *submit* dan *reset*

  Tombol *submit* digunakan untuk mengirim data yang telah diisikan di *form*. Tombol *reset* digunakan untuk mengosongkan isian pada setiap *form control* yang ada di *form* tersebut. Perlu diperhatikan bahwa saat ini tombol reset ini jarang sekali digunakan karena alasan pengalaman pengguna (*user experience*, UX). Pengguna yang telah mengisikan data pada suatu *form* seringkali secara tidak sengaja menekan tombol *reset* padahal sebenarnya ingin menekan tombol *submit* karena kedua tombol tersebut biasanya diletakkan berdekatan.

  Teks pada kedua tombol tersebut dapat diganti dengan menambahkan atribut "value". Contoh:

```
1   <input type="submit" value="Simpan">
2   <input type="reset" value="Kosongi">
```

- Tombol radio dan *checkbox*

  Tombol radio bertujuan agar pengguna dapat memilih paling banyak satu pilihan dari beberapa pilihan yang tersedia. Sedangkan *checkbox* bertujuan agar pengguna dapat memilih lebih dari satu pilihan dari beberapa pilihan yang tersedia. Beberapa tombol radio dan beberapa *checkbox* dapat dikelompokkan dengan memberi nilai yang sama pada atribut `name`.

  Berikut contoh pembuatan tombol radio pada form pemilihan usia di mana pengguna hanya dapat memilih satu jenis usia.

```
1   <input type="radio" name="usia" value="17">Di bawah 17
2   <input type="radio" name="usia" value="25">17-25
3   <input type="radio" name="usia" value="35">25-35
4   <input type="radio" name="usia" value="50">35-50
5   <input type="radio" name="usia" value="17">Di atas 50
```

Berikut contoh pembuatan *checkbox* pada *form* pemilihan hobi pengguna di mana pengguna dapat membuat beberapa opsi.

```
<input type=”checkbox” name=”hobi” value=”membaca”>Membaca
<input type=”checkbox” name=”hobi” value=”menulis”>Menulis
<input type=”checkbox” name=”hobi” value=”olahraga”>Olah raga
<input type=”checkbox” name=”hobi” value=”berkebun”>Berkebun
```

- *Dropdown menu*
  Dropdown menu memiliki fungsi yang sama dengan tombol radio, yaitu memungkinkan pengguna untuk memilih satu dari beberapa pilihan yang tersedia. Namun, *dropdown menu* lebih cocok digunakan apabila pilihan yang tersedia berjumlah banyak. Form control ini dibuat menggunakan tag `<select>` dan `<option>`. Berikut contoh pembuatan *dropdown menu*.

```
<select name=”band”>
  <option value=”dmasiv”>
  <option value=”dewa”>
  <option value=”thegroove”>
  <option value=”hivi”>
  <option value=”nidji”>
</select>
```

- *File upload*
  Form control ini memungkinkan pengguna dapat memilih satu file untuk kemudian di-*upload*. Contoh:

```
<input type=”file” name=”profilepic”>
```

  Agar dapat melakukan proses *upload*, pada tag `<form>`, harus ditambahkan atribut `enctype` seperti berikut.

```
<form action=”...” method=”post” enctype="multipart/form-data">
  ...
</form>
```

- Input tersembunyi
  Form control ini bertujuan untuk mengirimkan data yang tidak berasal dari input oleh penguna. Salah satu contoh kasus penggunaan *form control* ini adalah untuk mengirimkan URL yang ingin dituju (*redirect*) saat user berhasil melakukan proses *login*. Contoh:

```
<input type=”hidden” name=”redirect” value=http://filkom.ub.ac.id”>
```

- Tanggal dan waktu (HTML5)

Ini termasuk fitur baru yang ditambahkan secara *native* di *browser* di HTML5. Sebelum HTML5, *form control* ini hanya dapat dibuat menggunakan JavaScript. Ada beberapa jenis dari input teks ini, yaitu:

- Tanggal

```
1 <input type="date">
```

- Waktu

```
1 <input type="time">
```

- Tanggal dan waktu

```
1 <input type="datetime">
```

- Tanggal dan waktu lokal

```
1 <input type="datetime-local">
```

- Bulan

```
1 <input type="month">
```

- Pekan

```
1 <input type="week">
```

- Input numerik (HTML5)
  Ada dua jenis dari input numerik, yaitu:
  - Angka

```
1 <input type="number">
```

  - Rentang

```
1 <input type="range">
```

- Input warna (HTML5)
  Dengan *form control* ini, *browser* dapat menampilkan *pop-up* untuk memilih warna. Contoh:

```
1  <input type="color" name="favorit">
```

b) Metode GET dan POST

Data dari *form* yang menggunakan metode get dikirimkan ke *server* melalui URL pada bagian ***query string***. Pada URL, nama *domain* dan *query string* dipisahkan oleh karakter tanda tanda (?). *Query string* memiliki format:

```
field1=nilai1&field2=nilai2&...
```

Sebagai contoh, perhatikan data berikut.

Field	Nilai
nama	Joko
usia	56
alamat	Jakarta

Data tersebut dapat dituliskan dalam format *query string* seperti berikut.

```
nama=Joko&usia=56&alamat=Jakarta
```

Sehingga bentuk *query string* tersebut pada URL dengan nama *domain* "domain.com" adalah:

```
http://domain.com?nama=Joko&usia=56&alamat=Jakarta
```

Karena data dari *form* yang menggunakan metode get dikirimkan melalui URL, maka pengguna dapat membaca data tersebut melalui *address bar* pada *browser*. Oleh karena itu, untuk alasan keamanan, metode ini tidak baik untuk digunakan untuk mengirimkan data yang bersifat sensitif. Form dengan data yang sensitif, seperti *password*, harus dikirimkan menggunakan metode POST. Selain itu, metode GET memiliki keterbatasan pada kapasitas jika dibandingkan dengan metode POST.

Salah satu contoh kasus penggunaan metode GET dalam sebuah *form* adalah pada *form* pencarian. *Keyword* yang dikirimkan dari sebuah *form* pencarian umumnya tidak berisi informasi yang sensitif. Selain itu, keuntungan penggunaan metode GET pada *form* pencarian adalah URL dari hasil pencarian tersebut dapat dibagi (*share*) ke orang lain dan juga dapat di-*bookmark* di *browser* pengguna.

Berikut contoh *form* dalam HTML dengan metode GET.

```
1  <form action="search.php" method="get">
2    <input type="search" name="keyword">
3    <input type="submit" value="Search">
4  </form>
```

Data dari *form* yang menggunakan metode POST dikirimkan ke *server* melalui HTTP *request header* tanpa mengubah URL sehingga URL hanya berisi nama *domain* saja sehingga data yang dikirimkan tidak terbaca oleh pengguna.

Berikut contoh form dalam HTML dengan metode POST.

```
<form action=”proses.php” method=”post”>
  <input type=”text” name=”nama”>
  <input type=”submit” value=”Go”>
</form>
```

Untuk mengakses data yang telah diisi oleh pengguna melalui sebuah *form*, PHP menyediakan tiga *array* secara built-in: `$_GET`, `$_POST`, dan `$_FILES`. Ketiga *array* tersebut merupakan *array* asosiatif dengan indeks sesuai dengan nilai pada atribut `name` di masing-masing *form control*. Sebagai contoh, perhatikan *form* berikut.

```
<form action=”login.php” method=”post”>
  <input type=”text” name=”username”>
  <input type=”password” name=”password”>
  <input type=”submit” value=”Login”>
</form>
```

Saat *form* di atas di-*submit*, data *username* dan *password* yang diisikan oleh pengguna dapat dibaca di kode PHP menggunakan kode PHP berikut.

```
<?php
  $username = $_POST[‘username’];
  $password = $_POST[‘password’];
```

1.4 Prosedur Praktikum

a) Metode GET dan POST

Tulis kode berikut.

```
<body>
  <form action=”proses.php” method=”post”>
    <input type=”text” name=”nama”>
    <input type=”submit” value=”Go”>
  </form>
</body>
```

Jalankan kode tersebut di *browser* lalu ganti metodenya dengan `GET`. Lihat perbedaannya pada URL di *browser*. Beri kesimpulan.

---

b) Menentukan Metode

Untuk beberapa kasus berikut, tentukan metode apakah yang cocok untuk digunakan dan beri alasan.

1. *Form* untuk *login*.
2. *Form* untuk mengirim gambar desain poster ke *website* percetakan.
3. *Form* untuk mencari suatu artikel di situs berita.
4. *Form* untuk *search engine*.

---

c) **Form**

Tulis kode di bawah ini.

```
1  <form action="proses02.php" method="post‟ name="input">
2    Nama Anda: <input type="text" name="nama" required>
3    <br>
4    <input type="submit" name="input" value="Input">
5  </form>
```

Kemudian tulis kode di bawah ini.

```
1  <?php
2    if (isset($_POST['input'])) {
3      $nama = $_POST['nama'];
4      echo "Nama Anda: <b>$nama</b>";
5   }
```

a. Jalankan kode di atas kemudian isikan data input dan tekan tombol input. Apa yang terjadi?
b. Ulangi dengan mengosongkan data input dan amati apa yang terjadi.
c. Jelaskan alur pengiriman data dari *form* yang ada di kode tersebut!

---

d) **Required**

Ubah kembali kode pada Latihan 3 dengan menghapus tulisan `required` pada baris 2. Jalankan kode dan kosongi data nama kemudian tekan tombol Input. Apa yang terjadi? Jelaskan apa fungsi dari kode `required`.

---

e) **Upload**

Tulis kode berikut.

```
1   <form enctype="multipart/form-data" action="upload.php" method="post">
2     Choose a file to upload:
3     <input name="uploadedfile" type="file" /> <br>
4     <input type="submit" value="Upload File" />
5   </form>
```

Tulis kode berikut.

```
1   <?php
2     $target_path = "uploads/";
3     $target_path = $target_path . basename($_FILES['uploadedfile']['name']);
4
5     if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path))
6   {
7       echo "The file ".  basename( $_FILES['uploadedfile']['name']). " has
8   been uploaded";
9     } else {
10      echo "There was an error uploading the file, please try again!";
11    }
```

Di manakah posisi file setelah proses upload berhasil?

Apakah hasil output dari kode tersebut setelah dijalankan?

