Gathering input from user (Laravel)

A. Form

- Form merupakan tempat penginputan data sebelum diproses oleh sistem. Salah satunya adalah form login, form comment, form data user, dan lain sebagainya. Tag yang digunakan untuk membuat form adalah <form> yang didalamnya bisa berupa <input>, <textarea>, <select> dan <option>.
- Dalam tag <form> ini dibutuhkan beberapa atribut untuk memproses data yang akan dikirim. dan biasanya nilai yang dikirim berupa alamat dari sebuah halaman untuk memproses data inputan, dan atribut yang kedua berupa method. method ini menjelaskan bagaimana data yang akan dikirim oleh web browser. nilai dari method ini biasanya terdiri dari get atau post. berikut contoh penulisan <form> pada HTML :

```
<form action="proses.php" method="post">
    <!-- Isi Form -->
</form>
```

- Tag input merupakan tag yang akan digunakan dalam form pengisian. Tag input ini memiliki banyak sekali bentuk yang bisa digunakan misalkan isian teks, password, checkbox, radiobutton, sampai dengan tombol submit berada dalam tag <input>. Berikut beberapa bentuk dalam keluarga tag <input> yang dikategorikan berdasarkan atribut :
    1. **<input type='text'>** textbox yang bisa menerima inputan text misalkan username atau berupa inputan text yang pendek.
    2. **<input type='password'>** textbox yang menerima inputan text, akan tetapi text yang diinput akan di tampilkan sebagai tanda bintang atau titik, textbox ini biasanya digunakan untuk inputan password.
    3. **<input type='submit'>** inputan ini berupa tombol (button) untuk memproses data inputan dari form.
    4. **<input type='checkbox'>** inputan berupa checkbox yang dapat di ceklis oleh user.
    5. **<input type='radio'>** inputan yang berupa radio grup, dimana user dapat memilih salah satu dari pilihan yang disediakan. salah satu contoh penggunaan dari radio ini adalah jenis kelamain.

6. **<textarea>** merupakan tag yang sama fungsinya dengan inputan text, hanya saja pada textarea ini dapat diisikan dengan lebih banyak teks didalamnya. Untuk mengatur panjang dan banyak baris pada textarea ini dapat menggunakan CSS.
7. **<select>** merupakan tag yang digunakan untuk user memilih data yang sudah disediakan. Dalam penggunakan <select> selelau diikuti oleh <option> yang digunakan untuk membuat pilihan.

B. Input Validation

- Input validasi digunakan untuk melakukan pengecekan terhadap inputan yang di input di form, contohnya salah satu field di form tersebut harus diisi dengan email, kita dapat mengatur validasi di field tersebut yang mengharuskan pengguna memasukkan alamat email dengan format yang valid, jika alamat email yang dimasukkan tidak sesuai format yang valid maka akan muncul error pada field di form tersebut, dan data tidak bisa di proses, jadi kurang lebih seperti itu fungsi dari form validasi.
- Input validation terdapat pada dua sisi yaitu client side dan server side.
- Pada client side, validasi input dapat menggunakan html5 atau javascript. Sedangkan pada server side, menggunakan input validation dari Laravel. Berikut ini beberapa input validation dari client side :

1. Validasi Inputan Tidak Boleh Kosong (required)

```
<!DOCTYPE html>
<html>
<head>
   <title>Belajar Form Validasi</title>
</head>
<body>
<form method="post" action="">
    <input type="text" name="username" required/>
    <input type="submit" name="tombol" value="Tombol"/>
</form>
</body>
</html>
```

2. Validasi Inputan Harus Berupa Angka

```
<!DOCTYPE html>
<html>
<head>
   <title>Belajar Form Validasi</title>
</head>
<body>
<form method="post" action="">
    <input type="number" name="umur"/>
    <input type="submit" name="tombol" value="Tombol"/>
</form>
</body>
</html>
```

3. Validasi Inputan harus berupa alamat URL

```
<!DOCTYPE html>
<html>
<head>
   <title>Belajar Form Validasi</title>
</head>
<body>
<form method="post" action="">
    <input type="url" name="alamat_web"/>
    <input type="submit" name="tombol" value="Tombol"/>
</form>
</body>
</html>
```

4. Validasi Inputan harus berupa Email

```
<!DOCTYPE html>
<html>
<head>
   <title>Belajar Form Validasi</title>
</head>
<body>
<form method="post" action="">
    <input type="email" name="alamat_email"/>
    <input type="submit" name="tombol" value="Tombol"/>
</form>
</body>
</html>
```

5. Validasi jumlah maximal karakter yang dapat diinput

```
<!DOCTYPE html>
<html>
<head>
   <title>Belajar Form Validasi</title>
</head>
<body>
<form method="post" action="">
    <input type="text" name="username" maxlength="5"/>
    <input type="submit" name="tombol" value="Tombol"/>
</form>
</body>
</html>
```

- Validasi input dari Laravel, menggunakan fungsi validate() pada method yang digunakan untuk memproses data, seperti berikut :

```
public function proses(Request $request)
{
    $this->validate($request,[
        'nama' => 'required|min:5|max:20',
        'pekerjaan' => 'required',
        'usia' => 'required|numeric'
    ]);

    return view('proses',['data' => $request]);
}
```

C. CRUD (Model Eloquent)

- Untuk bermain-main dengan database pada laravel, sebenarnya ada 2 cara yang umum digunakan, yaitu bisa menggunakan **Query Builder** dan **Eloquent**. Keduanya merupakan fitur yang sudah disediakan pada Laravel, sehingga kita bisa lebih mudah dalam membuat CRUD pada laravel.
- Seperti yang dijelaskan pada dokumentasi laravel, Eloquent adalah sebuah fitur untuk mengelola data yang ada pada database dengan sangat mudah. Eloquent ORM menyediakan fungsi-fungsi active record, atau fungsi-sungsi query sql untuk mengelola data pada database.
- Dan fungsi query nya semua sudah dibuat dan disediakan secara default dalam laravel. jadi kita tidak perlu lagi mengetik query sql yang panjang-panjang.
- Dengan Eloquent, kita bisa mengelola data yang ada pada database dari hanya satu buah model. misalnya kita punya table siswa, maka kita juga akan mempunyai sebuah model dengan nama siswa, nah dengan model siswa ini kita bisa mengelola data-data yang ada pada tabel siswa dengan mudah dan cepat.
- Kita bisa menginput data, mengedit, menampilkan, mengupdate, bahkan kita juga bisa menggunakan relasi tabel dengan sangat mudah. Struktur penulisan coding nya pun sangat singkat. Jika secara manual pada PHP native, jika kita ingin mengakses atau menampilkan data dari table siswa, biasanya kita menggunakan query "select * from siswa".
- Jika kita menggunakan eloquent laravel, kita cukup mendefinisikan nama modelnya, kemudian kita bisa langsung menggunakan fungsi all() untuk mengambil semua data pada table siswa.

D. Setting Database untuk koneksi

- Buat database dengan nama yang sesuai dengan nama database yang sudah ada pada file .env. pada contoh ini saya membuat database dengan nama “belajar_laravel“. Dan buat table dengan nama “pegawai”. di sini kita membuat table pegawai dengan 5 column. yaitu pegawai_id, pegawai_nama, pegawai_jabatan.

pegawai_id	int (auto increment)
pegawai_nama	varchar (50)
pegawai_jabatan	varchar (20)
pegawai_umur	int
pegawai_alamat	text

1. Cari dan buka file .env. tetapi sebelumnya, pastikan sudah memiliki database. Setting konfigurasinya seperti dibawah ini :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=belajar_laravel
DB_USERNAME=root
DB_PASSWORD=root
```

2. jika sudah, berarti konfigurasi databasenya selesai.

E. Menampilkan Data Dari Database Dengan Laravel

1. Sebelum melangkah lebih jauh untuk cara menampilkan data dari database dengan laravel. Pastikan sudah memiliki beberapa data di tabel pegawai.
2. Setelah kita memiliki beberapa data untuk ditampilkan. maka sekarang kita mulai dengan membuat route untuk menampilkan data pegawai.
3. Buat route baru dengan alamat '/pegawai'.

```
//route CRUD
Route::get('/pegawai','PegawaiController@index');
```

4. Di sini kita memerintahkan untuk menjalankan method index() pada controller PegawaiController pada saat route '/pegawai' di akses.
5. Selanjutnya buat controller PegawaiController.php, karena kita belum mempunyai controller dengan nama PegawaiController.php.

```
php artisan make:controller PegawaiController
```

6. Karena sebelumnya pada route '/pegawai' kita memerintahkan untuk menjalankan method index(). Maka pada PegawaiController.php ini kita akan membuat method

index().

7. Dan karena kita akan menggunakan query builder laravel, maka kita wajib menambahkan perintah berikut pada bagian paling atas.

```
use Illuminate\Support\Facades\DB;
```

8. Kemudian, berikut ini adalah isi dari controllernya :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;


class PegawaiController extends Controller
{
    public function index()
    {
        // mengambil data dari table pegawai
        $pegawai = DB::table('pegawai')->get();

        // mengirim data pegawai ke view index
        return view('index',['pegawai' => $pegawai]);

    }
}
```

9. Dengan fungsi table() kita menentukan nama table yang ingin di pilih. fungsi get() berguna untuk mengampil data dari table yang di pilih. Sehingga data yang diambil akan disimpan dalam variabel $pegawai. Jadi intinya fungsi di atas itu seperti kita menampilkan data dari database dengan query mysql biasa seperti ” SELECT * FROM pegawai “.
10. Karena kita belum punya view index. maka sekarang kita buat view nya dengan nama index.blade.php.

```
<!DOCTYPE html>
<html>
<head>
    <title>Belajar Laravel</title>
</head>
<body>
    <h3>Data Pegawai</h3>
    <a href="/pegawai/tambah"> + Tambah Pegawai Baru</a>
    <br/>
    <br/>

    <table border="1">
        <tr>
            <th>Nama</th>
            <th>Jabatan</th>
            <th>Umur</th>
            <th>Alamat</th>
            <th>Opsi</th>
        </tr>
        @foreach($pegawai as $p)
        <tr>
            <td>{{ $p->pegawai_nama }}</td>
            <td>{{ $p->pegawai_jabatan }}</td>
            <td>{{ $p->pegawai_umur }}</td>
            <td>{{ $p->pegawai_alamat }}</td>
            <td>
                <a href="/pegawai/edit/{{ $p->pegawai_id }}">Edit</a>
                |
                <a href="/pegawai/hapus/{{ $p->pegawai_id }}">Hapus</a>
            </td>
        </tr>
        @endforeach
    </table>
</body>
</html>
```

11. Sampai di sini kita telah berhasil menampilkan data pegawai dari database pada laravel.

F. Menginput Data Ke Database Dengan Laravel

1. Buat dulu route dengan nama ‘/pegawai/tambah’.

   Route::get('/pegawai/tambah','PegawaiController@tambah');
2. Disini kita membuat pengaturan route, jika di akses route ‘pegawai/tambah‘ , maka akan kita perintahkan untuk menjalankan method tambah yang ada dalam controller PegawaiController.
3. Sekarang kita buat dulu method tambah nya dalam controller PegawaiController.

```php
// method untuk menampilkan view form tambah pegawai
public function tambah()
{

    // memanggil view tambah
    return view('tambah');

}
```

4. Sekarang buat view baru dengan nama tambah.blade.php dalam folder views seperti biasa

```php
<!DOCTYPE html>
<html>
<head>
    <title>Belajar Laravel</title>
</head>
<body>
    <h3>Data Pegawai</h3>
    <a href="/pegawai"> Kembali</a>
    <br/>
    <br/>

    <form action="/pegawai/store" method="post">
        {{ csrf_field() }}
        Nama <input type="text" name="nama" required="required"> <br/>
        Jabatan <input type="text" name="jabatan" required="required"> <br/>
        Umur <input type="number" name="umur" required="required"> <br/>
        Alamat <textarea name="alamat" required="required"></textarea> <br/>
        <input type="submit" value="Simpan Data">
    </form>

</body>
</html>
```

5. Pehatikan lagi pada view tambah.blade.php yang di buat di atas, untuk action form nya di arahkan ke route '/pegawai/store'. karena kita ingin nanti route 'pegawai/store' yang akan menangani pemprosesan data yang di input. Agar bisa di olah oleh controller.
6. Di dalam form kita juga telah membuat beberapa buah form inputan, yaitu ada nama, jabatan, umur dan alamat. sesuai dengan format table pegawai yang sudah di buat sebelumnya. Disini ada satu lagi fitur unggulan dari laravel, yaitu csrf protection. csrf projection semacam fitur keamanan untuk pencegahan penginputan data dari luar aplikasi atau sistem kita.

7. Jika tombol simpan diklik kita belum bisa memasukkan data ke database. karena kita belum punya aksi untuk mengolah data yang di input agar di insert ke database.
8. Karena pada form tambah pegawai action nya kita buat '/pegawai/store', maka pada saat form di submit, akan di arahkan ke route '/pegawai/store' untuk di proses.
9. Sekarang buat route baru lagi. yaitu route '/pegawai/store'.

   Route::post('/pegawai/store','PegawaiController@store');
10. Pada route ini kita menggunakan method "post", tidak lagi menggunakan method "get" seperti pada route sebelumnya yang kita buat. Karena kita mengirimkan data melalui form ke route '/pegawai/store'. Oleh karena itu, diwajib menggunakan method post pada route nya. Route ini kita perintahkan untuk menjalankan method store dalam controller PegawaiController, maka sekarang kita buat lagi sebuah method baru dalam controller PegawaiController. yaitu method store.

```
// method untuk insert data ke table pegawai
public function store(Request $request)
{
    // insert data ke table pegawai
    DB::table('pegawai')->insert([
        'pegawai_nama' => $request->nama,
        'pegawai_jabatan' => $request->jabatan,
        'pegawai_umur' => $request->umur,
        'pegawai_alamat' => $request->alamat
    ]);
    // alihkan halaman ke halaman pegawai
    return redirect('/pegawai');

}
```

11. Fungsi table() untuk memberitahukan nama table, fungsi insert() bertujuan untuk menginsert data dan menetapkan data apa saja yang ingin di insert. Jadi query builder di atas sama saja artinya seperti : INSERT INTO pegawai ….
12. Jika sudah, silahkan coba jalankan kembali.

G. Update Data Pada Database Dengan Laravel

1. Pada view yang menampilkan data pegawai yang sudah dibuat sebelumnya. yaitu pada view index.blade.php. Di sana telah ada tombol atau link edit seperti berikut. pegawai_id }}">Edit
2. Buat route baru seperti berikut, agar data id yang dikirimkan sekalian kita kirim juga ke controller untuk kita ambil data pegawai yang ber id tersebut. **Route::get('/pegawai/edit/{id}','PegawaiController@edit');**
3. Buat method edit nya dalam controller PegawaiController.php.

```php
// method untuk edit data pegawai
public function edit($id)
{
    // mengambil data pegawai berdasarkan id yang dipilih
    $pegawai = DB::table('pegawai')->where('pegawai_id',$id)->get();
    // passing data pegawai yang didapat ke view edit.blade.php
    return view('edit',['pegawai' => $pegawai]);

}
```

4. Sekarang buat view baru dengan nama edit.blade.php. karena kita akan menampilkan data pegawai yang ingin di edit tadi di dalam form edit.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Belajar Laravel</title>
</head>
<body>
    <h3>Edit Pegawai</h3>
    <a href="/pegawai"> Kembali</a>
    <br/>
    <br/>

    @foreach($pegawai as $p)
    <form action="/pegawai/update" method="post">
        {{ csrf_field() }}
        <input type="hidden" name="id" value="{{ $p->pegawai_id }}"> <br/>
        Nama <input type="text" required="required" name="nama" value="{{ $p->pegawai_nama }}"> <br/>
        Jabatan <input type="text" required="required" name="jabatan" value="{{ $p->pegawai_jabatan }}"> <br/>
        Umur <input type="number" required="required" name="umur" value="{{ $p->pegawai_umur }}"> <br/>
        Alamat <textarea required="required" name="alamat">{{ $p->pegawai_alamat }}</textarea> <br/>
        <input type="submit" value="Simpan Data">
    </form>
    @endforeach
</body>
</html>
```

5. buat route ‘/pegawai/update‘ dulu untuk meng-handle data dari form edit pegawai ini. Route::post('/pegawai/update','PegawaiController@update');
6. Sekarang buat lagi sebuah method updatenya pada pegawaiController.php

```php
// update data pegawai
public function update(Request $request)
{
    // update data pegawai
    DB::table('pegawai')->where('pegawai_id',$request->id)->update([
        'pegawai_nama' => $request->nama,
        'pegawai_jabatan' => $request->jabatan,
        'pegawai_umur' => $request->umur,
        'pegawai_alamat' => $request->alamat
    ]);
    // alihkan halaman ke halaman pegawai
    return redirect('/pegawai');
}
```

7. Coba jalankan apakah fungsi editnya telah berjalan.

H. Menghapus Data Dari Database Dengan Laravel

1. Perhatikan pada view index.blade.php yang sudah kita buat sebelumnya. Pada masing-masing row data pegawai sudah kita buatkan link atau tombol hapus nya.
2. Sekarang buat route '/pegawai/hapus' nya untuk menangani pengiriman data id ini. buka file web.php seperti biasa. dan buat route berikut.

   Route::get('/pegawai/hapus/{id}','PegawaiController@hapus');
3. Buat method hapus dalam controller PegawaiController.php. untuk melakukan proses penghapusan data pegawai berdasarkan id yang di terima menggunakan query builder laravel.

```php
// method untuk hapus data pegawai
public function hapus($id)
{
    // menghapus data pegawai berdasarkan id yang dipilih
    DB::table('pegawai')->where('pegawai_id',$id)->delete();

    // alihkan halaman ke halaman pegawai
    return redirect('/pegawai');
}
```

4. Sekarang coba fungsi hapusnya.

I. Exercise: Lanjutkan sistem yang telah dibuat, tambahkan tabel baru yang berhubungan dengan pegawai, kemudian buat CRUD nya, untuk menambah data , menampilkan

data, menghapus data barang, dan mengubah data barang. (Tabel dan fieldnya bebas, sesuai dengan sistem penjualan)