LAPORAN BAB 6

PEMROGRAMAN WEB LANJUT

Dosen Pengampu :

Lintang Setyo Kurniawati S.Kom, M.Pd

DISUSUN OLEH :

BAYU KRISNA (2010651095)

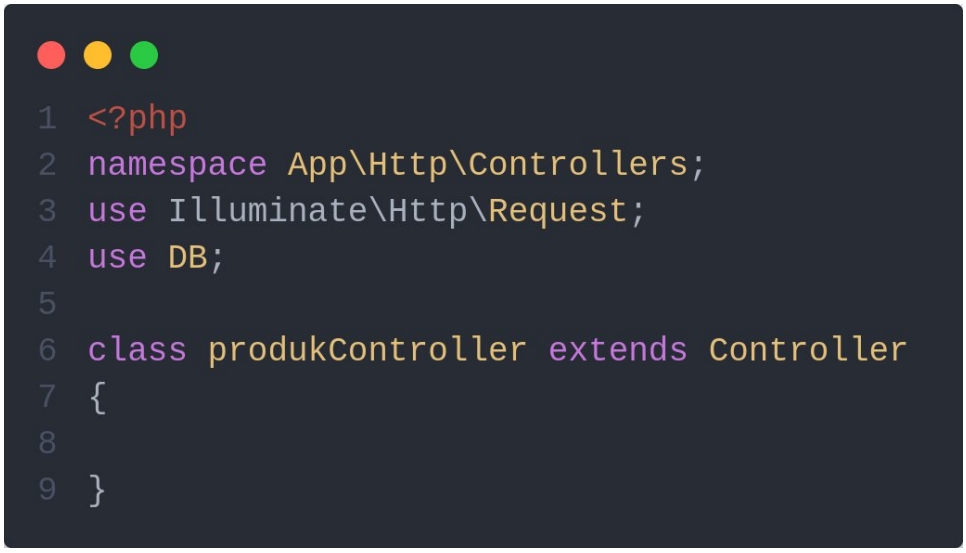
TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH JEMBER

2022

INTERAKSI DATABASE DENGAH QUERY BUILDER

Query Builder merupakan salah satu cara untuk menjalankan query database dengan lebih mudah, Query Builder juga telah dilengkapi dengan fitur keamanan untuk mencegah terjadinya SQL Injextion (adalah sebuah aksi hacking yang dilakukan di aplikasi client dengan cara memodifikasi perintah SQL yang ada di memori aplikasi client). Selain itu kita dapat menggunakan query builder tanpa harus membuat model terlebih dahulu. Sricpt query bulder cukup mudah dipahami, misalnya untuk menampilkan data dari tabel produk yang dibuat sebelumnya pertama-tama kita harus mendefinisikan penggunaan class dari QUERY BUILDER dengan cara menambahkan script use DB; pada bagian atas controller atau seperti contoh dibawah:



Dengan pendefinisian mengunakan script use DB; kita telah menambahkan class QUERY BUILDER ke dalam controller produkController. Lalu untuk menampilkan data dari tabel produks yang telah dibuat sebelumnya bisa menggunakan script berikut:

DB::table('produks')->get();

KEGIATAB PRAKTIKUM 1 MENAMPILKAN DATA DENGAN QUERY BUILDER

Untuk mempraktekan contoh dari latihan menampilkan data dengan QUERY BUILDER pertama kita harus memeriksa file .env apakah projek sudah terhubung dengan database atau belum. File ini terletak pada bagian luar projek laravel yang dibuat, dan pastikan pada bagian mysql sudah terkonfigurasi seperti pada gambar dibawah:

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=penjualan

DB_USERNAME=root

DB_PASSWORD=

Database yang akan kita gunakan adalah database dengan nama penjualan, yang telah kita buat pada latihan pembuatan migration sebelumnya. Kemudian pada file

produkController.php yang berada pada folder app\Http\Controller telah dibuat pada latihan sebelumnya buatlah satu fungsi yang bernama index() atau modifikasi jika sudah ada dan tambahkan script sebagai berikut:

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use DB;
class produkController extends Controller
{
 public function index()
 {
 $produk = DB::table('produks')->get();
 return view('produk/index',compact('produk'));
 }
}
```

Fungsi index() diatas akan mengambil semua data yang ada pada tabel produk dan kemudian akan mengirimkannya ke view index.blade.php yang ada di dalam folder resources/views/produk. Jadi pada view index.blade.php yang ada pada folder resourcess/views/produk (jika belum ada silahkan dibuat) modifikasi script nya menjadi seperti dibawah:

```blade
<!DOCTYPE html>
<html>

<head>
    <title>Laravel Saya</title>
</head>

<body>
    <h3><b>Data Produk</b></h3>
    <table border="1">
        <thead>
            <tr>
                <th>No</th>
                <th>Nama</th>
                <th>Kategori</th>
                <th>Qty</th>
                <th>Harga Beli</th>
                <th>Harga Jual</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                @foreach ($produk as $i => $p)
            <tr>
                <td>{{ $i+1 }}</td>
                <td>{{ $p->nama }}</td>
                <td>{{ $p->id_kategori }}</td>
                <td>{{ $p->qty }}</td>
                <td>{{ $p->harga_beli }}</td>
                <td>{{ $p->harga_jual }}</td>
            </tr>
            @endforeach
            </tr>
        </tbody>
    </table>
</body>

</html>
```

Sebelum mencoba menjalankan skript diatas pertama-tama cobalah untuk memeriksa route yang ada pada file web.php yang ada pada folder routes/web.php dan periksa apakah route Route::get('/produk', 'produkController@index'); sudah didefinisikan atau belum. Jika sudah maka aktifkan server artisan pada cmd lalu akses http://localhost:8000/produk maka hasilnya akan nampak seperti pada gambar dibawah:

Data Produk

No	Nama	Kategori	Qty	Harga Beli	Harga Jual
1	Meja	1	12	50000	540000
2	Kursi	1	12	40000	450000

KEGIATAN PRAKTIKUM 2 JOIN TABEL DENGAN QUERY BUILDER

Jika diperhatikan pada tampilan data produk pada tabel diatas pada kolom id_kateogri kita masih menampilkan angka atau bukan data dari suatu kategori produk. Untuk memperbaikinya buatlah satu tabel baru dengan nama kateogri menggunakan fasilitas migration pada Laravel dengan tahapan sebagai berikut:

1. Jalankan perintah artisan berikut pada cmd atau comand promt yang sudah terarah ke dalam directori projek laravel yang digunakan

2. Langkah kedua bukalah file 2018_09_14_015345_create_kategori_table.php yang ada folder database/migration dan modifikasi koding di dalamnya menjadi seperti dibawah:

```php
<?php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateKategoriTable extends Migration
{
 /**
 * Run the migrations.
 *
 * @return void
 */
 public function up()
 {
 Schema::create('kategori', function (Blueprint $table) {
 $table->increments('id');
 $table->string('kategori');
 $table->string('keterangan');
 $table->timestamps();
 });
 }
 /**
 * Reverse the migrations.
 *
 * @return void
 */
 public function down()
 {
 Schema::dropIfExists('kategori');
 }
}
```

4. Jalankan perintah *php artisan migrate* dan tabel kan beruba seperti ini

Options	id	kategori	keterangan	created_at	updated_at
Edit Copy Delete	1	Perlengkapan Rumah Tangga	0	NULL	NULL

Setelah membuat tabel kategori beserta dengan 1 contoh datanya sekarang bukalah class Controller produkController.php yang ada pada folder app\Http\Controller dan modifikasi fungsi index() yang ada di dalam controller tersebut menjadi seperti dibawah:

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use DB;
class produkController extends Controller
{
 public function index()
 {
 $produk = DB::table('produks')
 ->join('kategori','produks.id_kategori','=','kategori.id')
 ->get();

 return view('produk/index',compact('produk'));
 }
}
```

Kemudian pada view index.blade.php yang ada pada folder resources/views/produk/ modifikasi kodingnya menjadi seperti dibawah:

```blade
<!DOCTYPE html>
<html>

<head>
    <title>Laravel Saya</title>
</head>

<body>
    <h3><b>Data Produk</b></h3>
    <table border="1">
        <thead>
            <tr>
                <th>No</th>
                <th>Nama</th>
                <th>Kategori</th>
                <th>Qty</th>
                <th>Harga Beli</th>
                <th>Harga Jual</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                @foreach ($produk as $i => $p)
            <tr>
                <td>{{ $i+1 }}</td>
                <td>{{ $p->nama }}</td>
                <td>{{ $p->kategori }}</td>
                <td>{{ $p->qty }}</td>
                <td>{{ $p->harga_beli }}</td>
                <td>{{ $p->harga_jual }}</td>
            </tr>
            @endforeach
            </tr>
        </tbody>
    </table>
</body>
```

MENAMBAH DATA DENGAN QUERY BUILDER

Untuk menambahkan data menggunakan Query Builder buatlah satu route baru pada file web.php yang terletak pada folder routes/web.php dengan script sebagai berukut:

```
Route::get('/produk/store', 'produkController@store');
```

Pada route diatas akan menjalankan fungsi store yang ada di dalam class Controller produkController.php, untuk itu bukalah file produkController yang ada pada folder App/Http/Controller dan tambahkan satu fungsi yang bernama store() dan isikan script sebagai berikut

```php
<?php
public function store()
 {
 DB::table('produks')
 ->insert([
 'nama' => 'Lampu',
 'id_kategori' => 1,
 'qty' => 14,
 'harga_beli' => 40000,
 'harga_jual' => 60000,
 ]);
 echo "Data Berhasil Ditambah";
 }
```

Pada script diatas adalah script untuk menambah data ke tabel yang ada di dalam database, pada kasus diatas tabel yang akan ditambah datanya adalah tabel produks. Laravel sendiri menggunakan array assosiatif untuk parameter kolom dan nilai yang akan ditambah dengan ketentuan index dari array akan menjadi nama kolom dan nilai dari array akan menjadi nilai yang akan disimapn ke kolom. Jika kita menjalankan localhost:8000/produk/store pada browser akan muncul tampilan seperti berikut:

Data Berhasil Ditambah

Dan bila kita memerika tabel produk yang ada di dalam database penjualan maka akan ada satu data baru seperti pada gambar dibawah:

←T→	id	nama	id_kategori	qty	harga_beli	harga_jual	created_at	updated_at
Edit Copy Delete	3	Meja	1	12	50000	540000	NULL	NULL
Edit Copy Delete	4	Kursi	1	12	40000	450000	NULL	NULL
Edit Copy Delete	5	Lampu	1	14	40000	60000	NULL	NULL

MEMPERBAHRUI DATA QUERY BUILDER

Untuk memperbaharui (update) data, kita harus menentukan terlebih dahulu data mana yang akan di update. Pada contoh ini kita akan memperbaharui data yang baru saja ditambahkan yaitu data dengan id 5, pertama buatlah satu route baru di fiile web.php di folder routes/web.php dengan skrip sebagai berikut:

Route::get('/produk/update', 'produkController@update');

Pada route diatas jika kita membuka localhost:8000/produk/update maka kita akan dibawa ke function update() yang ada pada file produkController.php, maka dari itu buatlah satu function baru bernama update di file produkController.php yang ada pada folder App/Http/Controller dengan script sebagai berikut:

```php
<?php
public function store()
 {
 DB::table('produks')
 ->insert([
 'nama' => 'Lampu',
 'id_kategori' => 1,
 'qty' => 14,
 'harga_beli' => 40000,
 'harga_jual' => 60000,
 ]);
 echo "Data Berhasil Diperbarui";
 }
```

Pada script diatas merupakan script untuk memperbaharui data menggunakan Query Builder. Pertama kita harus menentukan data mana yang akan diperbaharui dengan menggunakan fungsi where() dan diikuti dengan fungsi update() untuk memperbaharui data yang ada pada kolom. Dan jika dilihat pada tabel produk yang ada pada database penjualan data dengan id 3 akan berubah

MENGHAPUS DATA QUERY BUILDER

Untuk menghapus data kita perlu menentukan terlebih dahulu data mana yang akan dihapus lalu diikuti dengan fungsi delete(). Pertama buatlah route seperti berikut pada file web.php yang ada pada folder routes:

Route::get('/produk/delete', 'produkController@delete');

Pada route diatas jika kita mengakses localhost:8000/produk/delete pada browser maka kita akan diarahkan ke fungsi delete yang ada pada file produkController, maka dari itu buatlah fungsi yang bernama delete pada file produkController.php dan isikan script sebagai berikut

```php
<?PHP
public function delete()
 {
 DB::table('produks')->where('id',3)->delete();
 echo "Data Berhasil Dihapus";
 }
```

Pada script diatas adalah script untuk menghapus data menggunakan Query Builder. Pertama kita harus menentukan terlebih dahulu data mana yang akan dihapus menggunakan fungsi where() untuk menghapus data dengan syarat tertentu. Lalu diikuti dengan fungsi delete() untuk menghapus data.