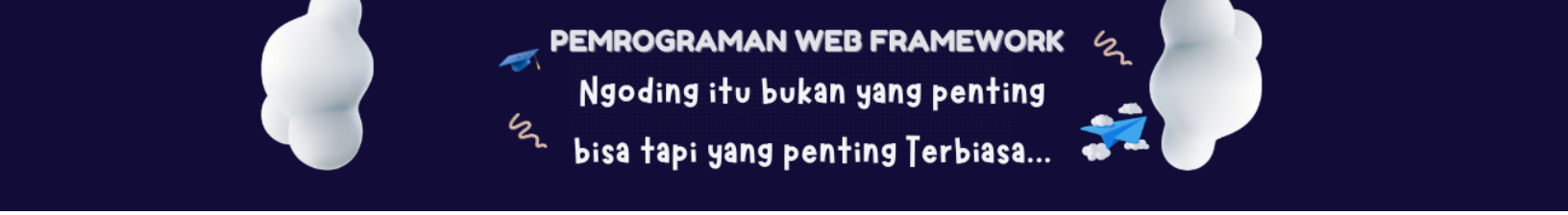
PENGEMBANGAN DAN PENYELENGGARAAN
PEMBELAJARAN DIGITAL (P3D)

Modul Pembelajaran

SISTEM INFORMASI

Pemrgoraman Web Framework

Disusun oleh : Wicaksono Yuli Sulistyo



Modul Ajar: Select, Insert, Update, dan Delete Database pada Laravel

Pengenalan

Laravel adalah framework PHP yang dirancang untuk mempermudah pengembangan aplikasi web dengan menyediakan alat yang kuat dan elegan. Salah satu fungsi utama dari Laravel adalah kemampuannya untuk berinteraksi dengan database melalui Eloquent ORM (Object-Relational Mapping). Dalam modul ini, kita akan membahas cara melakukan operasi dasar pada database, termasuk Select, Insert, Update, dan Delete, menggunakan Laravel.

Persiapan Awal

Sebelum kita mulai, pastikan Anda telah mengikuti langkah-langkah berikut:

1. **Instalasi Laravel**: Jika belum, buat proyek Laravel baru dengan perintah:

   ```
   bash

   composer create-project --prefer-dist laravel/laravel crud-example
   ```

2. **Koneksi Database**: Konfigurasikan koneksi database di file `.env`. Misalnya, jika menggunakan MySQL, tambahkan detail berikut:

   ```
   plaintext

   DB_CONNECTION=mysql
   DB_HOST=127.0.0.1
   DB_PORT=3306
   DB_DATABASE=nama_database
   DB_USERNAME=username_database
   DB_PASSWORD=password_database
   ```

3. **Migrasi Database**: Jalankan migrasi untuk membuat tabel `users` yang sudah ada dan siap digunakan.

   ```
   bash

   php artisan migrate
   ```

Membuat Tabel dengan Migrasi

Langkah pertama adalah membuat tabel yang akan kita gunakan untuk operasi database. Kita akan membuat tabel `products` yang berisi informasi tentang produk. Gunakan perintah berikut untuk membuat migrasi:

```
bash

php artisan make:migration create_products_table
```

Setelah itu, buka file migrasi yang baru saja dibuat di `database/migrations`, dan tambahkan kolom yang diperlukan, seperti nama produk, harga, dan deskripsi:

```
php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateProductsTable extends Migration
{
    public function up()
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->decimal('price', 8, 2);
            $table->text('description')->nullable();
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('products');
    }
}
```

Setelah mendefinisikan migrasi, jalankan perintah berikut untuk membuat tabel `products` di database:

```
bash

php artisan migrate
```

Membuat Model Eloquent

Selanjutnya, kita perlu membuat model Eloquent untuk tabel `products`. Model ini akan memudahkan kita dalam melakukan operasi database. Gunakan perintah berikut untuk membuat model:

```
bash

php artisan make:model Product
```

Model `Product` akan berada di `app/Models/Product.php`. Model ini secara otomatis terhubung dengan tabel `products` di database dan dapat digunakan untuk melakukan operasi CRUD.

Operasi Database dengan Eloquent

1. Insert (Menyisipkan Data)

Untuk menambahkan data baru ke dalam tabel, kita dapat menggunakan metode `create` atau `save`. Berikut adalah contoh cara menyisipkan data menggunakan metode `create`:

```
php

use App\Models\Product;

// Menyisipkan data produk baru
$product = Product::create([
    'name' => 'Produk 1',
    'price' => 100.00,
    'description' => 'Deskripsi Produk 1',
]);
```

Jika Anda ingin menggunakan metode `save`, Anda dapat melakukannya sebagai berikut:

```
php

$product = new Product();
$product->name = 'Produk 2';
$product->price = 150.00;
$product->description = 'Deskripsi Produk 2';
$product->save();
```

2. Select (Mengambil Data)

Untuk mengambil data dari tabel, kita dapat menggunakan berbagai metode yang disediakan oleh Eloquent. Berikut adalah beberapa contoh:

- **Mengambil semua data produk**:

```
php

$products = Product::all();
```

- **Mengambil produk berdasarkan ID**:

```
php

$product = Product::find(1);
```

- **Mengambil produk dengan kondisi tertentu**:

```
php

$expensiveProducts = Product::where('price', '>', 100)->get();
```

3. Update (Memperbarui Data)

Untuk memperbarui data yang sudah ada, kita dapat menggunakan metode `update`. Berikut adalah contoh cara memperbarui data produk:

```
php

$product = Product::find(1);
$product->price = 120.00;
$product->save();
```

Anda juga bisa menggunakan metode `update` dalam satu langkah:

```
php

Product::where('id', 1)->update(['price' => 130.00]);
```

4. Delete (Menghapus Data)

Untuk menghapus data dari tabel, kita dapat menggunakan metode `delete`. Berikut adalah contoh cara menghapus produk:

```
php

$product = Product::find(1);
$product->delete();
```

Anda juga bisa menghapus produk berdasarkan kondisi tertentu:

```
php

Product::where('price', '<', 50)->delete();
```

Menampilkan Data di Tampilan

Untuk menampilkan data yang telah diambil dari database, kita perlu membuat tampilan. Mari kita buat tampilan untuk menampilkan daftar produk. Buat file `resources/views/products/index.blade.php` dengan konten berikut:

```
blade

@extends('layouts.app')

@section('content')
    <div class="container">
        <h1>Daftar Produk</h1>
        <a href="{{ route('products.create') }}" class="btn btn-
primary">Tambah Produk</a>
        <table class="table">
            <thead>
```

```
                <tr>
                    <th>ID</th>
                    <th>Nama</th>
                    <th>Harga</th>
                    <th>Deskripsi</th>
                    <th>Aksi</th>
                </tr>
            </thead>
            <tbody>
                @foreach ($products as $product)
                    <tr>
                        <td>{{ $product->id }}</td>
                        <td>{{ $product->name }}</td>
                        <td>{{ $product->price }}</td>
                        <td>{{ $product->description }}</td>
                        <td>
                            <a href="{{ route('products.edit', $product->id)
}}" class="btn btn-warning">Edit</a>
                            <form action="{{ route('products.destroy',
$product->id) }}" method="POST" style="display:inline;">
                                @csrf
                                @method('DELETE')
                                <button type="submit" class="btn btn-
danger">Hapus</button>
                            </form>
                        </td>
                    </tr>
                @endforeach
            </tbody>
        </table>
    </div>
@endsection
```

Membuat Controller

Selanjutnya, kita perlu membuat controller untuk mengelola logika aplikasi. Gunakan perintah berikut untuk membuat controller:

```
bash

php artisan make:controller ProductController
```

Di dalam `ProductController`, kita akan mendefinisikan metode untuk setiap operasi CRUD:

```
php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\Http\Request;

class ProductController extends Controller
{
```

```
// Menampilkan semua produk
public function index()
{
    $products = Product::all();
    return view('products.index', compact('products'));
}

// Menampilkan form untuk menambah produk baru
public function create()
{
    return view('products.create');
}

// Menyimpan produk baru ke database
public function store(Request $request)
{
    $request->validate([
        'name' => 'required',
        'price' => 'required|numeric',
        'description' => 'nullable|string',
    ]);

    Product::create($request->all());

    return redirect()->route('products.index')
                     ->with('success', 'Produk berhasil ditambahkan.');
}

// Menampilkan form untuk mengedit produk
public function edit(Product $product)
{
    return view('products.edit', compact('product'));
}

// Memperbarui produk yang ada
public function update(Request $request, Product $product)
{
    $request->validate([
        'name' => 'required',
        'price' => 'required|numeric',
        'description' => 'nullable|string',
    ]);

    $product->update($request->all());

    return redirect()->route('products.index')
                     ->with('success', 'Produk berhasil diperbarui.');
}

// Menghapus produk
public function destroy(Product $product)
{
    $product->delete();

    return redirect()->route('products.index')
                     ->with('success', 'Produk berhasil dihapus.');
```

```
    }
}
```

Mengatur Rute

Selanjutnya, kita perlu mendefinisikan rute untuk aplikasi CRUD di file `routes/web.php`. Kita perlu membuat rute untuk setiap operasi CRUD yang telah kita buat di `ProductController`. Tambahkan kode berikut:

```
php

use App\Http\Controllers\ProductController;

Route::resource('products', ProductController::class);
```

Menambahkan Form Tambah dan Edit Produk

1. **Form Tambah Produk**: Buat file `resources/views/products/create.blade.php`:

```
blade

@extends('layouts.app')

@section('content')
    <div class="container">
        <h1>Tambah Produk</h1>
        <form method="POST" action="{{ route('products.store') }}">
            @csrf
            <div class="mb-3">
                <label for="name" class="form-label">Nama Produk</label>
                <input type="text" class="form-control" name="name" required>
            </div>
            <div class="mb-3">
                <label for="price" class="form-label">Harga Produk</label>
                <input type="number" step="0.01" class="form-control"
name="price" required>
            </div>
            <div class="mb-3">
                <label for="description" class="form-label">Deskripsi</label>
                <textarea class="form-control" name="description"></textarea>
            </div>
            <button type="submit" class="btn btn-primary">Tambah</button>
        </form>
    </div>
@endsection
```

2. **Form Edit Produk**: Buat file `resources/views/products/edit.blade.php`:

```
blade

@extends('layouts.app')
```

```
@section('content')
    <div class="container">
        <h1>Edit Produk</h1>
        <form method="POST" action="{{ route('products.update', $product->id)
}}">
            @csrf
            @method('PUT')
            <div class="mb-3">
                <label for="name" class="form-label">Nama Produk</label>
                <input type="text" class="form-control" name="name" value="{{
$product->name }}" required>
            </div>
            <div class="mb-3">
                <label for="price" class="form-label">Harga Produk</label>
                <input type="number" step="0.01" class="form-control"
name="price" value="{{ $product->price }}" required>
            </div>
            <div class="mb-3">
                <label for="description" class="form-label">Deskripsi</label>
                <textarea class="form-control" name="description">{{
$product->description }}</textarea>
            </div>
            <button type="submit" class="btn btn-primary">Update</button>
        </form>
    </div>
@endsection
```

Menjalankan Aplikasi

Setelah semua langkah di atas selesai, jalankan aplikasi Laravel Anda menggunakan perintah:

```
bash

php artisan serve
```

Akses aplikasi di browser melalui `http://localhost:8000/products`. Anda sekarang dapat menambahkan, melihat, mengedit, dan menghapus produk.

Kesimpulan

Dalam modul ini, Anda telah belajar cara melakukan operasi Select, Insert, Update, dan Delete (CRUD) pada database menggunakan Laravel. Anda telah membuat tabel, model Eloquent, controller, rute, dan tampilan yang diperlukan untuk menjalankan aplikasi CRUD. Laravel menyediakan alat yang kuat untuk bekerja dengan database dan menyederhanakan banyak tugas yang terkait dengan pengelolaan data.