ROUTES DAN PRINSIP ARSITEKTUR FRAMEWORK LARAVEL 11

Bagi sebagian orang, terutama bagi mereka yang baru memulai dalam pengembangan web dengan *framework*, Laravel bisa tampak seperti *tools programming* yang rumit, oleh karena itu, tujuan utama dari bab ini adalah untuk memberikan Anda gambaran umum yang komprehensif dan mendalam tentang cara kerja framework Laravel. Kami ingin membantu Anda memahami setiap elemen dan fitur dari Laravel, mulai dari yang paling dasar hingga konsep-konsep yang lebih tinggi.

Sering kali, ketika pertama kali diperkenalkan dengan bahasa pemrograman atau framework bahasa pemrograman baru seperti Laravel, Anda mungkin merasa perlu belajar lebih banyak dengan istilah teknis dan konsep yang belum familiar, namun, dengan memahami keseluruhan *framework* secara lebih baik, Anda dapat menghilangkan rasa kebingungan dan menggantikannya dengan pengetahuan baru yang cukup bermanfaat untuk karir Anda, memahami Laravel bukan hanya tentang mengetahui cara menggunakannya, tetapi juga tentang memahami prinsip-prinsip yang mendasari cara kerja *framework* ini. Ketika Anda mengerti bagaimana bagian-bagian berbeda dari Laravel bekerja bersamaan, Anda akan merasa lebih nyaman dan lebih percaya diri dalam membangun aplikasi yang efektif dan efisien. Misalnya, Laravel menggunakan arsitektur Model View Controller (MVC) yang memisahkan logika aplikasi menjadi tiga bagian utama seperti model, view, dan controller, seperti pada bab sebelumnya yang telah kita bahas, Dengan memahami konsep dasar ini, Anda akan dapat merancang aplikasi Anda dengan cara yang lebih terstruktur dan mudah dikelola. Anda akan tahu persis di mana menempatkan kode tertentu, bagaimana mengatur logika bisnis Anda, dan cara terbaik untuk menampilkan data kepada pengguna. Pengetahuan ini tidak hanya membuat proses pengembangan menjadi lebih lancar tetapi juga meningkatkan kualitas dan keandalan aplikasi Anda secara keseluruhan.

Sering kali, ketika pertama kali diperkenalkan dengan kerangka kerja baru seperti Laravel, Anda mungkin merasa kewalahan dengan banyaknya istilah teknis dan konsep yang belum familiar. Anda mungkin merasa seperti mempelajari bahasa baru atau masuk ke dunia yang sepenuhnya berbeda. Namun, dengan memahami keseluruhan kerangka kerja secara lebih baik, Anda dapat menghilangkan rasa kebingungan ini dan menggantikannya dengan kepercayaan diri yang lebih besar. Memahami Laravel bukan hanya tentang mengetahui cara menggunakannya, tetapi juga tentang memahami prinsip-prinsip yang mendasari cara kerja framework ini. Ketika Anda mengerti bagaimana bagian-bagian berbeda dari Laravel bekerja bersama-sama,

Anda akan merasa lebih nyaman dan lebih percaya diri dalam membangun aplikasi yang efektif dan efisien. Misalnya, Laravel menggunakan arsitektur Model View Controller (MVC) yang memisahkan logika aplikasi menjadi tiga bagian utama: model, view, dan controller. Dengan memahami konsep dasar ini, Anda akan dapat merancang aplikasi Anda dengan cara yang lebih terstruktur dan mudah dikelola. Anda akan tahu persis di mana menempatkan kode tertentu, bagaimana mengatur logika bisnis Anda, dan cara terbaik untuk menampilkan data kepada pengguna. Pengetahuan ini tidak hanya membuat proses pengembangan menjadi lebih lancar tetapi juga meningkatkan kualitas dan keandalan aplikasi Anda secara keseluruhan.

File index pada laravel

Setiap aplikasi web membutuhkan titik masuk atau sering disebut index.php di mana permintaan dari pengguna pertama kali diterima dan kemudian diproses. Dalam konteks framework Laravel, file public/index.php berfungsi sebagai titik masuk untuk semua permintaan yang masuk ke aplikasi. Ini berarti setiap permintaan HTTP yang dibuat oleh pengguna ke aplikasi Laravel akan diarahkan terlebih dahulu ke file index.php. Pengarahan ini diatur oleh konfigurasi server web yang digunakan, seperti Apache atau Nginx. Konfigurasi ini memastikan bahwa semua permintaan yang datang ke server diarahkan ke direktori public tempat file index.php berada, sehingga permintaan tidak langsung mengakses file lain dalam struktur aplikasi yang dapat menimbulkan risiko keamanan. File index.php tidak berisi banyak kode, tetapi memainkan peran penting dalam arsitektur Laravel. Peran utama file ini adalah bertindak sebagai titik awal untuk memuat seluruh framework. Ini memungkinkan Laravel untuk mengontrol sepenuhnya bagaimana setiap permintaan diproses, menyediakan lingkungan yang aman dan terstruktur untuk menjalankan aplikasi.

Di dalam file index.php, langkah pertama adalah memuat autoloader yang dihasilkan oleh Composer. Autoloader ini sangat penting karena memudahkan dalam memuat kelas-kelas yang diperlukan secara dinamis tanpa harus menulis banyak pernyataan require dan include secara manual. Dengan adanya autoloader ini, proses pemanggilan kelas menjadi lebih efisien dan terorganisir, serta mengurangi kemungkinan terjadinya kesalahan akibat file yang hilang atau salah dimuat.

Memahami Kernel HTTP dan Kernel Konsol pada Laravel

Dalam arsitektur aplikasi Laravel, terdapat dua jenis kernel utama yang mengelola aliran permintaan yaitu kernel HTTP dan kernel konsol. Kedua kernel ini memainkan peran penting dalam menangani permintaan yang masuk ke aplikasi. Kernel HTTP menangani permintaan HTTP (seperti permintaan GET, POST, PUT, DELETE), sementara kernel konsol menangani perintah baris perintah (CLI) seperti perintah Artisan. Ketika permintaan masuk ke aplikasi, Laravel menentukan jenis permintaan

tersebut dan kemudian menggunakan metode yang sesuai (handleRequest untuk permintaan HTTP atau handleCommand untuk permintaan CLI) pada *instance* aplikasi untuk mengarahkan permintaan tersebut ke kernel yang tepat. Ini berarti kernel bertindak sebagai pusat pemrosesan untuk semua permintaan yang diterima oleh aplikasi, mengarahkan mereka melalui berbagai komponen dan lapisan framework sebelum menghasilkan respons akhir.

Middleware dalam Laravel

Middleware adalah lapisan logika kode yang bertindak sebagai penjaga gerbang (*authentication* dan *authorization*) antara permintaan HTTP yang masuk dan respons aplikasi. Middleware ini mengontrol alur permintaan dan memastikan bahwa berbagai kondisi dan aturan terpenuhi sebelum permintaan mencapai *Controller* atau *routes*, secara default *middleware* berada didalam direktori app/http/middleware.

Beberapa fungsi penting dari middleware di Laravel termasuk:

- Pembacaan dan Penulisan Sesi HTTP: Middleware menangani pengelolaan sesi HTTP, seperti membaca data sesi yang ada atau menulis data baru ke sesi saat ini. Ini penting untuk pelacakan status pengguna, seperti saat mereka login atau mengunjungi halaman tertentu dalam aplikasi.
- Mode Pemeliharaan (Maintenance Mode): Middleware dapat menentukan apakah aplikasi saat ini dalam mode pemeliharaan. Jika aplikasi dalam mode ini, middleware akan memblokir semua permintaan masuk dan menampilkan pesan pemeliharaan kepada pengguna, memastikan bahwa pengguna tidak dapat mengakses aplikasi selama periode pemeliharaan atau pembaruan.
- Verifikasi Token Cross-Site Request Forgery (CSRF): Middleware VerifyCsrfToken adalah contoh lain dari middleware penting yang digunakan untuk melindungi aplikasi dari serangan CSRF dengan memverifikasi bahwa setiap permintaan POST yang masuk memiliki token CSRF yang valid.
- Middleware fungsi lainnya, dapat mencakup logika autentikasi pengguna, pembatasan laju (rate limiting), enkripsi dan dekripsi, serta banyak fungsi lainnya yang berguna untuk mengontrol akses dan perilaku aplikasi.

Middleware memungkinkan Laravel untuk tetap modular dan terstruktur, di mana logika yang menangani permintaan ditempatkan di lokasi yang dapat diatur dan dikelola dengan baik. Pengembang dapat dengan mudah menambahkan, menghapus, atau mengubah middleware untuk menyesuaikan bagaimana permintaan diproses dalam aplikasi, laravel memiliki perintah bawaan untuk membuat sistem otentikasi dasar menggunakan Auth Scaffolding, middleware dapat diakses di app/Http/Middleware.

Siklus Hidup Permintaan dan Respons dalam Kernel HTTP

Setelah permintaan HTTP melalui middleware dan setiap middleware menjalankan fungsinya, permintaan tersebut kemudian diteruskan ke *Routes* atau pengontrol yang sesuai untuk penanganan lebih lanjut. Di sinilah inti aplikasi berjalan, di mana logika bisnis yang sesungguhnya dieksekusi. Metode pengontrol atau *Routes* menangani permintaan, memproses data yang diperlukan, berinteraksi dengan database atau layanan lain, dan luarannya menghasilkan respons.

Routes dalam laravel

Routes dalam Laravel berfungsi sebagai peta untuk aplikasi yang dibuat, mengarahkan permintaan HTTP ke *Controller* yang sesuai atau logika penanganan lainnya. Dalam Laravel, Routes adalah salah satu aspek inti dari *framework* yang menentukan bagaimana aplikasi menanggapi berbagai permintaan HTTP yang masuk. Laravel menyediakan berbagai fitur Routes yang kuat dan fleksibel, termasuk Routes dasar, Routes dinamis, middleware, *Routes* berada pada folder root /routes seperti pada bab sebelumnya yang telah dibahas tentang direktori *Routes* ini. Berikut adalah penjelasan tentang penggunaan routes dalam Laravel beserta contohnya, dan kita akan mengedit pada file *Routes* web.php sebagai penerima *Request* http dari pengguna melalui browser. Berikut beberapa contoh dari penggunaan *Routes* :

a. Routes Dasar (Basic Routing)

Rute dasar dalam Laravel memungkinkan Anda menentukan URL sederhana yang memetakan permintaan HTTP ke suatu fungsi atau pengontrol tertentu. Anda dapat menggunakan berbagai metode HTTP seperti GET, POST, PUT, DELETE, dll., untuk menentukan jenis permintaan yang akan ditangani. Silahkan anda buka file routes di routes/web.php kemudian buatlah kode seperti dibawah ini, pada baris baru pada file tersebut.

```
// Rute GET sederhana
Route::get('/hello', function () {
    return 'Hello, World!';
});
```

Pada contoh di atas:

- *Routes* pertama menangani permintaan GET ke URL /hello dan mengembalikan teks "Hello, World!". Untuk mengaksesnya, silahkan buka di browser http://127.0.0.1:8000/hello

b. Rute dengan Parameter (Route Parameters)

Laravel menyediakan fitur untuk mendefinisikan rute dengan parameter dinamis, yang memungkinkan pengembang menangani berbagai pola URL secara fleksibel dan dinamis. Fitur ini sangat berguna ketika Anda perlu mengelola rute yang berisi variabel, seperti ID pengguna atau nama produk, yang berbeda-beda untuk setiap permintaan. Dengan menggunakan parameter dinamis, Anda dapat menangkap bagian dari URL sebagai variabel dan meneruskannya ke fungsi atau pengontrol untuk diproses lebih lanjut. Misalnya, Anda dapat mendefinisikan rute yang menangani URL seperti /user/{id} di mana {id} adalah parameter dinamis yang dapat diisi dengan nilai apapun. Saat pengguna mengakses URL seperti /user/1 atau /user/42, Laravel secara otomatis akan menangkap nilai 1 atau 42 dan meneruskannya ke fungsi yang sesuai. Fungsi tersebut kemudian dapat menggunakan nilai parameter ini untuk mengambil informasi spesifik dari database atau menjalankan logika lainnya.

Laravel juga mendukung parameter opsional dalam rute, yang berarti Anda dapat menentukan parameter yang mungkin tidak selalu ada dalam URL. Jika parameter tersebut tidak disertakan dalam URL, Anda dapat menetapkan nilai default untuk digunakan. Kemampuan untuk mendefinisikan rute dengan parameter dinamis dan opsional ini memberikan fleksibilitas yang besar dalam pengelolaan URL dan penanganan permintaan, memungkinkan aplikasi Laravel untuk merespons permintaan pengguna dengan lebih efisien dan efektif. Fitur ini membantu dalam mengembangkan aplikasi yang lebih dinamis dan responsif, dengan kemampuan untuk

menangani berbagai permintaan dari pengguna secara real-time. Silahkan kembali tambahkan kode pada file web.php

Contoh:

```
// Rute dengan parameter
Route::get('/user/{id}', function ($id) {
    return "User ID: " . $id;
});
```

Kemudian silahkan akses di browser dengan menambahkan parameter angka atau huruf seperti berikut http://127.0.0.1/user/1234

Penjelasan pada contoh ini:

- Rute menangani permintaan GET ke URL /user/{id}, di mana {id} adalah parameter dinamis. Misalnya, jika URL adalah /user/1, maka nilai 1 akan diteruskan ke fungsi callback dan ditampilkan sebagai "User ID: 1".

c. Rute dengan Parameter Opsional (Optional Parameters)

Laravel menyediakan dukungan untuk parameter opsional dalam rute, yang memungkinkan pengembang untuk menetapkan parameter yang tidak selalu harus ada di dalam URL. Fitur ini memberikan fleksibilitas lebih dalam penanganan rute karena memungkinkan aplikasi untuk merespons berbagai pola URL yang mungkin tidak selalu memiliki semua bagian yang ditentukan. Misalnya, Anda bisa mendefinisikan rute seperti /user/{name?} di mana {name?} adalah parameter opsional. Jika pengguna mengakses URL /user, Laravel akan menggunakan nilai default yang telah ditentukan, seperti "Guest", dan menampilkan pesan "Hello, Guest". Namun, jika pengguna mengakses URL /user/John, nilai "John" akan digunakan, dan pesan "Hello, John" akan ditampilkan. Penggunaan parameter opsional ini sangat berguna dalam berbagai skenario, terutama ketika aplikasi harus menangani URL yang memiliki struktur serupa tetapi tidak selalu memerlukan semua informasi. Misalnya, dalam aplikasi e-commerce, Anda mungkin memiliki rute yang menampilkan semua produk dalam kategori tertentu. Jika kategori tidak disebutkan dalam URL, Anda dapat menampilkan semua produk. Dengan menggunakan parameter opsional, Anda dapat dengan mudah mengatur

aplikasi untuk menangani kedua skenario tersebut tanpa perlu mendefinisikan dua rute terpisah.

Selain itu, kemampuan untuk menggunakan parameter opsional membantu meningkatkan kegunaan dan fleksibilitas aplikasi web dengan mengurangi kompleksitas logika routing dan membuat kode lebih bersih dan mudah dipelihara. Ini memungkinkan pengembang untuk mengoptimalkan pengalaman pengguna dengan menyediakan respons yang tepat berdasarkan ketersediaan parameter tertentu dalam URL. Dengan begitu, Laravel membuat pengelolaan permintaan menjadi lebih dinamis dan responsif, menyesuaikan dengan berbagai kebutuhan aplikasi modern.

Contoh:

```
// Rute dengan parameter opsional
Route::get('/user/{name?}', function ($name = 'Guest') {
    return "Hello, " . $name;
});
```

Di sini {name?} adalah parameter opsional. Jika tidak ada nilai yang diberikan dalam URL, maka defaultnya adalah 'Guest'. Misalnya, /user akan menampilkan "Hello, Guest", sedangkan /user/John akan menampilkan "Hello, John".

d. Rute dengan Penamaan (Named Routes)

Laravel menyediakan fitur untuk memberi nama pada rute, yang membuat proses pengelolaan rute dalam aplikasi menjadi lebih mudah dan terorganisir. Dengan memberikan nama pada rute, pengembang dapat merujuk rute tersebut dengan lebih mudah di berbagai bagian aplikasi, seperti saat melakukan pengalihan (redirect) atau membangun URL. Misalnya, jika Anda memiliki rute yang dinamai profile, Anda bisa menggunakan nama ini untuk mengarahkan pengguna ke halaman profil dengan metode route('profile') di dalam fungsi atau pengontrol. Ini lebih efisien dibandingkan harus mengingat atau menuliskan URL lengkapnya setiap kali ingin merujuk rute tersebut. Penggunaan rute bernama juga membantu meningkatkan kejelasan dan pemeliharaan kode. Saat URL berubah, Anda hanya perlu memperbarui definisi rutenya saja tanpa harus mencari dan mengganti semua referensi URL di seluruh aplikasi. Selain itu, rute bernama memungkinkan pengembang untuk lebih fokus pada logika aplikasi tanpa harus memikirkan detail URL yang spesifik. Ini sangat berguna dalam pengembangan aplikasi besar di mana rute bisa sangat banyak dan kompleks.

Contoh lain penggunaan rute bernama adalah saat membuat tautan dalam tampilan (view). Dengan menggunakan fungsi route('nama_rute'), Anda dapat dengan mudah menghasilkan URL yang konsisten dan bebas kesalahan. Misalnya, Profile akan menghasilkan tautan yang tepat ke halaman profil. Dengan demikian, Laravel membantu memudahkan pengelolaan dan pemeliharaan aplikasi dengan memberikan cara yang lebih efisien dan fleksibel untuk menangani rute. Fitur ini memperkuat kemampuan Laravel untuk membangun aplikasi web yang dinamis, fleksibel, dan mudah dikelola, terutama saat menangani perubahan dan pembaruan dalam struktur URL aplikasi.

Contoh:

```
// Rute dengan nama
Route::get('/profile', function () {
    return 'This is the profile page.';
})->name('profile');
```

Contoh routes untuk redirect ke routes lain

```
// Menggunakan rute bernama untuk pengalihan
Route::get('/redirect-to-profile', function () {
    return redirect()->route('profile');
});
```

Pada contoh di atas:

- Rute /profile dinamai profile.
- Rute /redirect-to-profile mengarahkan pengguna ke rute bernama profile menggunakan metode redirect()->route().

e. Rute Grup (Route Groups)

Laravel menyediakan fitur pengelompokan rute menggunakan metode Route::group(), yang memungkinkan pengembang untuk mengatur beberapa rute secara bersamaan dan menerapkan atribut bersama seperti middleware, namespace, atau prefix. Dengan mengelompokkan rute, Anda dapat mengurangi redundansi dan membuat kode lebih bersih serta mudah dikelola. Misalnya, jika ada beberapa rute yang memerlukan middleware autentikasi yang sama, Anda dapat memasukkan semua rute tersebut dalam satu grup dan menetapkan middleware pada grup tersebut, daripada harus menetapkan middleware secara individual untuk setiap rute.

Pengelompokan rute juga berguna ketika Anda ingin menerapkan prefix yang sama pada beberapa rute. Misalnya, jika Anda memiliki beberapa rute yang semuanya dimulai dengan /admin, Anda dapat menggunakan Route::group() untuk menambahkan prefix ini secara otomatis ke semua rute dalam grup tersebut. Ini tidak hanya menghemat waktu tetapi juga membuat kode lebih mudah dibaca dan dimengerti.

Contoh:

```
Route::prefix('admin')->group(function () {
    Route::get('/dashboard', function () {
        return 'Admin Dashboard';
    });

    Route::get('/profile', function () {
        return 'Admin Profile';
    });
});
```

Pada contoh ini:

- Kedua rute berada di dalam grup dengan prefix 'admin'. Jadi, rute /dashboard dan /profile sebenarnya dapat diakses melalui URL /admin/dashboard dan /admin/profile.

f. Middleware pada Rute (Route Middleware)

Laravel memungkinkan Anda untuk menetapkan middleware pada rute guna menangani berbagai tugas penting seperti autentikasi, validasi, dan fungsi lainnya. Middleware berfungsi sebagai lapisan perantara yang beroperasi sebelum rute sebenarnya diakses, memastikan bahwa permintaan HTTP memenuhi kriteria tertentu atau menjalani proses tertentu sebelum diteruskan ke logika aplikasi utama. Dengan menggunakan middleware, Anda dapat mengotomatiskan tugas-tugas seperti memeriksa apakah pengguna telah masuk sebelum mengizinkan akses ke halaman tertentu. Misalnya, middleware auth memastikan hanya pengguna yang terautentikasi yang dapat mengakses halaman dashboard atau profil. Ini sangat penting untuk mengamankan aplikasi dan menjaga data pengguna tetap aman. Middleware lain, seperti VerifyCsrfToken, melindungi aplikasi dari serangan CSRF dengan memverifikasi token yang ada di setiap permintaan POST. Selain autentikasi dan keamanan, middleware juga dapat digunakan untuk validasi data, memastikan bahwa input pengguna memenuhi persyaratan tertentu sebelum diproses lebih lanjut. Misalnya, jika ada form yang memerlukan data tertentu, middleware dapat memastikan bahwa data tersebut ada dan valid sebelum form diteruskan untuk diproses.

Laravel juga memungkinkan Anda menetapkan beberapa middleware pada satu rute atau grup rute, yang memberikan fleksibilitas besar dalam mengelola alur kerja aplikasi. Anda bisa, misalnya, menggabungkan middleware autentikasi dan validasi pada satu rute untuk memastikan pengguna terautentikasi dan input mereka valid. Dengan menetapkan middleware pada rute atau grup rute, Laravel membantu menjaga aplikasi tetap modular dan terstruktur, mengurangi risiko kesalahan, dan meningkatkan keamanan serta efisiensi aplikasi secara keseluruhan. Fitur ini membuat pengelolaan aplikasi menjadi lebih mudah dan efektif, terutama dalam proyek skala besar dengan kebutuhan kompleks.

Contoh:

```
Route::get('/dashboard', function () {
    return 'Welcome to your dashboard!';
})->middleware('auth');
```

Penjelasan contoh diatas:

- Middleware auth memastikan bahwa hanya pengguna yang terautentikasi yang dapat mengakses rute /dashboard.

g. Rute Sumber Daya (Resource Routes)

Laravel menyediakan cara cepat untuk mengatur rute yang menangani semua tindakan CRUD untuk sumber daya dengan menggunakan Route::resource().

Contoh:

```
Route::resource('posts', 'PostController');
```

Rute ini secara otomatis menghasilkan beberapa rute untuk mengelola sumber daya posts, seperti:

- GET /posts - Menampilkan daftar posting
- GET /posts/create - Menampilkan formulir untuk membuat posting baru
- POST /posts - Menyimpan posting baru
- GET /posts/{id} - Menampilkan posting tertentu
- GET /posts/{id}/edit - Menampilkan formulir untuk mengedit posting
- PUT/PATCH /posts/{id} - Memperbarui posting yang ada
- DELETE /posts/{id} - Menghapus posting