

Nama : Misyhel Oktavia Br Nababan

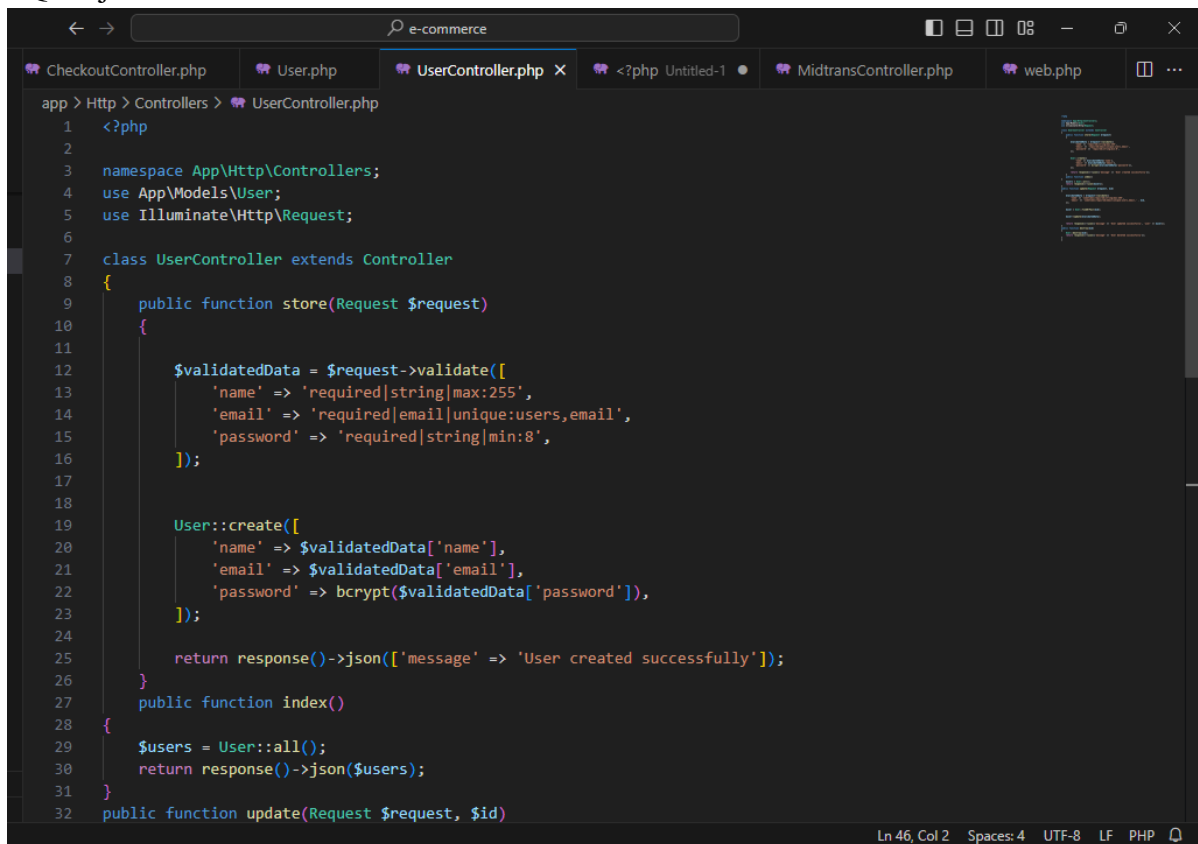
Kelas : PW 1

Kampus : Universitas Methodist Indonesia Medan

1. Jelaskan pentingnya menggunakan Query Builder dan Eloquent ORM untuk mencegah SQL Injection

Menggunakan Query Builder dan Eloquent ORM di Laravel penting untuk mencegah SQL Injection karena keduanya secara otomatis melindungi aplikasi dari potensi serangan. Mereka ini melakukan sanitasi input dari pengguna, sehingga nilai yang dimasukkan tidak dijalankan langsung sebagai bagian dari query SQL mentah. Dengan Query Builder, kita menggunakan metode yang memisahkan data input dari struktur query, misalnya dengan menggunakan `where()` daripada memasukkan nilai langsung ke dalam query. Sedangkan dengan Eloquent ORM, setiap interaksi dengan database dilakukan melalui model yang memanfaatkan binding parameter, jadi data input terproteksi secara otomatis. Kedua teknik ini jauh lebih aman dibandingkan menulis query SQL secara manual, terutama jika kita menerima input langsung dari pengguna.

2. Lampirkan screenshoot dari proyek laravel yang telah dikerjakan penerapan ORM dan SQL injection



```
app > Http > Controllers > UserController.php
1  <?php
2
3  namespace App\Http\Controllers;
4  use App\Models\User;
5  use Illuminate\Http\Request;
6
7  class UserController extends Controller
8  {
9      public function store(Request $request)
10     {
11
12         $validatedData = $request->validate([
13             'name' => 'required|string|max:255',
14             'email' => 'required|email|unique:users,email',
15             'password' => 'required|string|min:8',
16         ]);
17
18
19         User::create([
20             'name' => $validatedData['name'],
21             'email' => $validatedData['email'],
22             'password' => bcrypt($validatedData['password']),
23         ]);
24
25         return response()->json(['message' => 'User created successfully']);
26     }
27     public function index()
28     {
29         $users = User::all();
30         return response()->json($users);
31     }
32     public function update(Request $request, $id)
```

3. Jelaskan langkah-langkah dalam menggunakan parameter binding dalam raw query untuk mencegah SQL injection dalam proyek Laravel

Parameter binding adalah metode untuk mengikat nilai variabel ke dalam query SQL menggunakan placeholder, yang mencegah SQL Injection. Dengan teknik ini, data pengguna yang dimasukkan tidak langsung dimasukkan ke dalam query, melainkan diproses terpisah dan diikat ke query sebagai parameter. Hal ini memastikan bahwa input pengguna diperlakukan hanya sebagai data, bukan bagian dari kode SQL, sehingga mencegah potensi eksekusi query berbahaya yang dapat merusak database. Ini adalah cara yang lebih aman dibandingkan menyisipkan data langsung ke dalam query SQL.

4. Jelaskan langkah-langkah melakukan sanitasi data pada proyek laravel yang telah dikerjakan screenshootnya

Sanitasi data dalam Laravel adalah proses untuk membersihkan input pengguna agar aman sebelum diproses oleh aplikasi. Berikut adalah langkah-langkah :

1. **Validasi Data:** Memastikan data yang diterima sesuai dengan kriteria yang ditetapkan, seperti format yang benar atau panjang yang sesuai.
2. **Sanitasi Input Otomatis:** Laravel secara otomatis menghindari potensi ancaman seperti XSS dengan membersihkan data sebelum disimpan atau ditampilkan.
3. **Pembersihan Karakter Berbahaya:** Menghapus atau memfilter karakter berbahaya, seperti tag HTML, untuk mencegah eksekusi kode berbahaya.
4. **Middleware untuk Sanitasi:** Menggunakan middleware untuk membersihkan input sebelum mencapai controller, memastikan konsistensi sanitasi di seluruh aplikasi.
5. **Sanitasi Output:** Melakukan pembersihan pada data yang ditampilkan ke pengguna untuk mencegah **Cross-Site Scripting (XSS)**.
6. **Sanitasi File Upload:** Memvalidasi file yang diunggah pengguna untuk memastikan file tersebut aman dan tidak berbahaya.

Sanitasi data memastikan bahwa aplikasi aman dari serangan seperti SQL Injection dan XSS, dengan memproses hanya data yang aman.

5. Jelaskan langkah-langkah menggunakan fungsi `e()` atau Blade template engine untuk melakukan escaping output

Fungsi `e()` dan **Blade template engine** digunakan untuk **escaping output** di Laravel, yang bertujuan untuk mencegah serangan **Cross-Site Scripting (XSS)**. Berikut langkah-langkah “

1. **Menggunakan `e()`**: Fungsi ini secara otomatis meng-escape karakter-karakter khusus dalam output, sehingga teks yang dimasukkan oleh pengguna atau data yang tidak dipercaya tidak akan dieksekusi sebagai kode HTML atau JavaScript.
2. **Menggunakan Blade Template Engine**: Laravel secara otomatis melakukan escaping output yang ditampilkan menggunakan sintaks Blade (`{{ }}`). Ini memastikan bahwa segala data yang ditampilkan ke pengguna akan terproteksi dari potensi XSS, karena karakter-karakter khusus akan di-escape.

Dengan menggunakan kedua cara ini, Laravel membantu melindungi aplikasi dari serangan XSS dengan memastikan bahwa input atau data yang ditampilkan ke pengguna tidak bisa dieksekusi sebagai kode.

6. Jelaskan langkah-langkah melakukan validasi data pada proyek laravel yang telah dibuat dan berikan screenshootnya

Langkah-langkah melakukan validasi data di Laravel adalah sebagai berikut:

1. **Menerima Input**: Data dari pengguna diterima melalui request, baik itu melalui form atau API.
2. **Menentukan Aturan Validasi**: Tentukan aturan validasi yang sesuai dengan kebutuhan aplikasi, seperti tipe data, panjang karakter, format email, dll.
3. **Menggunakan Validasi Built-in**: Laravel menyediakan metode validasi built-in yang dapat diterapkan langsung pada request. Validasi ini akan memeriksa apakah data yang diterima memenuhi aturan yang telah ditentukan.
4. **Menangani Validasi**: Jika data tidak valid, Laravel secara otomatis akan mengembalikan pesan kesalahan yang bisa ditampilkan ke pengguna.
5. **Melanjutkan Proses jika Validasi Berhasil**: Jika validasi berhasil, data dapat diteruskan untuk diproses lebih lanjut, seperti disimpan ke database atau digunakan dalam logika aplikasi lainnya.
6. **Kustomisasi Pesan Validasi**: Anda dapat menyesuaikan pesan kesalahan agar lebih mudah dipahami oleh pengguna.

Dengan melakukan validasi, Anda memastikan bahwa data yang masuk ke aplikasi sudah sesuai dengan kriteria yang diinginkan, dan membantu mencegah kesalahan atau potensi masalah keamanan.

7. Jelaskan bagaimana Laravel secara otomatis menyertakan token CSRF di setiap form dan memvalidasinya dalam proyek laravel yang telah dikerjakan lengkap dengan screenshootnya.

Laravel secara otomatis mengelola **Cross-Site Request Forgery (CSRF)** dengan menyertakan token CSRF di setiap form yang dihasilkan oleh aplikasi. CSRF token digunakan untuk melindungi aplikasi dari serangan di mana penyerang mencoba untuk mengirimkan permintaan yang tidak sah atas nama pengguna yang sah. Laravel menangani ini dengan cara berikut:

Langkah-langkah Laravel Mengelola CSRF:

1. **Menyertakan Token CSRF dalam Form:** Setiap form di Laravel secara otomatis disertakan dengan token CSRF. Anda hanya perlu memastikan form menggunakan direktif `@csrf` di dalam Blade template.
 - **Token CSRF disertakan di setiap form:** Laravel menambahkan token CSRF ke dalam form menggunakan tag `@csrf`. Tag ini menghasilkan elemen input tersembunyi yang berisi token.
2. **Validasi CSRF Token:** Saat form dikirimkan, Laravel memvalidasi token CSRF. Token ini dibandingkan dengan token yang disimpan di session pengguna. Jika tidak cocok atau hilang, permintaan akan diblokir dan error CSRF akan dikembalikan.
 - Laravel secara otomatis memvalidasi token CSRF untuk setiap permintaan POST, PUT, DELETE, dan PATCH.
3. **Token CSRF pada Permintaan AJAX:** Untuk permintaan AJAX, Anda harus mengirimkan token CSRF dalam header permintaan. Laravel menyediakan cara untuk menambahkan token ke header secara otomatis.

Pengelolaan CSRF di Laravel:

- **Form Blade Template:** Dalam Blade, Anda cukup menambahkan direktif `@csrf` untuk memasukkan token CSRF ke dalam form.
- **Middleware CSRF:** Laravel memiliki middleware `VerifyCsrfToken` yang secara otomatis memeriksa validitas token CSRF untuk setiap permintaan yang membutuhkan.

Langkah-langkah:

1. **Menambahkan Token CSRF di Form:** Cukup tambahkan direktif `@csrf` dalam form di Blade.
2. **Mengirim Form:** Saat form dikirim, Laravel secara otomatis memvalidasi token.
3. **Menangani Kesalahan CSRF:** Jika token tidak cocok atau hilang, Laravel akan mengembalikan error CSRF dan mencegah form diproses.

8. Jelaskan bagaimana menambahkan token CSRF secara manual pada AJAX request dalam proyek laravel yang telah dikerjakan lengkap dengan screenshootnya.

Berikut adalah cara Laravel menambahkan dan memvalidasi **token CSRF** pada permintaan **AJAX**:

1. **Token CSRF di Halaman HTML:** Laravel secara otomatis menambahkan token CSRF di dalam halaman HTML yang dihasilkan. Token ini disertakan sebagai elemen input tersembunyi di form atau dalam tag meta pada bagian `<head>` dari halaman HTML.
2. **Menangkap Token CSRF di JavaScript:** JavaScript yang berjalan di halaman web dapat mengambil token CSRF ini dari elemen meta yang ada di dalam HTML, untuk kemudian digunakan dalam permintaan AJAX. Token ini penting agar server dapat memverifikasi bahwa permintaan berasal dari aplikasi yang sah.
3. **Menambahkan Token CSRF ke Header Permintaan AJAX:** Saat membuat permintaan AJAX menggunakan JavaScript, token CSRF yang telah diambil ditambahkan ke dalam header permintaan dengan nama `X-CSRF-TOKEN`. Hal ini memungkinkan server untuk memvalidasi permintaan dan memastikan bahwa permintaan tersebut sah dan tidak berasal dari sumber yang tidak diinginkan.
4. **Middleware CSRF di Laravel:** Laravel secara otomatis memvalidasi token CSRF yang dikirimkan melalui header permintaan AJAX dengan menggunakan middleware `VerifyCsrfToken`. Middleware ini membandingkan token yang diterima dengan token yang ada di session pengguna. Jika token valid, permintaan akan diproses; jika tidak, permintaan akan diblokir dan error CSRF akan dikembalikan.
5. **Penanganan Error CSRF:** Jika token CSRF tidak valid atau tidak ditemukan, Laravel akan mengembalikan error **419 (Page Expired)** atau **Token Mismatch**. Ini menandakan bahwa permintaan tidak sah atau telah dimanipulasi oleh pihak ketiga.

Dengan langkah-langkah ini, Laravel memastikan bahwa setiap permintaan yang membutuhkan perubahan data (seperti POST, PUT, DELETE) aman dari serangan **Cross-Site Request Forgery (CSRF)**.

9. Jelaskan langkah-langkah melakukan testing CSRF protection dalam proyek laravel yang telah dikerjakan lengkap dengan screenshootnya.

Berikut adalah langkah-langkah untuk menguji **CSRF protection** dalam proyek Laravel:

1. **Uji Form dengan Token CSRF:** Pastikan setiap form di aplikasi menyertakan token CSRF. Laravel secara otomatis menambahkannya jika menggunakan direktif `@csrf` dalam Blade template. Pengujian ini memastikan bahwa token CSRF disertakan dan dikirim saat form disubmit.
2. **Uji Permintaan AJAX:** Pastikan setiap permintaan AJAX mengirimkan token CSRF dalam header. Token ini diambil dari elemen meta di halaman HTML atau form yang ada. Uji untuk memastikan token CSRF terkirim dengan benar dan valid.
3. **Uji Permintaan Tanpa Token CSRF:** Coba kirim permintaan POST, PUT, atau DELETE tanpa token CSRF atau dengan token yang salah. Laravel seharusnya

- menanggapi dengan error **419 (Page Expired)** atau **Token Mismatch**, menunjukkan bahwa permintaan ditolak.
4. **Verifikasi Middleware CSRF:** Pastikan middleware `VerifyCsrfToken` aktif di aplikasi. Middleware ini bertanggung jawab untuk memverifikasi token CSRF yang dikirimkan dalam setiap permintaan yang memodifikasi data.
 5. **Menangani Error CSRF:** Uji apakah Laravel menangani error CSRF dengan benar, yaitu dengan memberikan respons yang sesuai (error **419**) jika token tidak valid atau hilang.
 6. **Pengujian Otomatis:** Lakukan pengujian otomatis menggunakan alat pengujian seperti PHPUnit untuk memastikan token CSRF dikirimkan dan diverifikasi dengan benar dalam permintaan API atau form.

Dengan langkah-langkah ini, Anda dapat memastikan bahwa aplikasi Laravel terlindungi dengan baik dari serangan **Cross-Site Request Forgery (CSRF)**.