Laravel 2 | Authenticatie & Packages | Nog in ontwikkeling

Les 1: Wat zijn Laravel packages?

Leerdoelen

- Je kunt uitleggen wat een package is.
- Je kunt Composer gebruiken om packages te installeren.
- Je kent het verschil tussen `require` en `require --dev`.
- Je kunt Laravel Debugbar installeren en gebruiken.

Uitleg

In de vorige Laravel module heb je code geschreven voor je Game Collection app. Soms wil je functionaliteit toevoegen aan je Laravel applicatie, die iemand anders al heeft gebouwd. Zoals een inlogsysteem, een betalingssysteem of hulpmiddelen voor debuggen. Hiervoor gebruik je packages.

Een package is een kant-en-klaar stuk code dat je kunt installeren in je Laravelproject. Je kunt het vergelijken met een plug-in in WordPress of een extensie in VS Code. Iemand anders heeft de code geschreven, getest en beschikbaar gemaakt. Jij hoeft het alleen te installeren en te gebruiken.

afbeelding.png

Composer is de package manager voor PHP. Net zoals npm dat is voor JavaScript, beheert Composer alle packages in je (PHP / Laravel) project. Wanneer je een package installeert, downloadt Composer de code naar Laravel en plaatst deze in de map `vendor/`.

composer.json

In het bestand `composer.json` staat een lijst van alle packages die je project gebruikt.

Soort	Commando	Wanneer beschikbaar?
require	composer require naam/package	Altijd (ook op de live website)
require-dev	composer require naam/package --dev	Alleen tijdens ontwikkelen

□□ **Let op:** Packages die alleen handig zijn tijdens het programmeren, zoals een debugger, installeer je met `--dev`. Packages die nodig zijn op de live website, zoals een inlogsysteem, installeer je zonder `--dev`.

□□ Opdracht – Laravel Debugbar installeren

Installeer je eerste package: Laravel Debugbar. Deze tool toont onderin je browser informatie over je applicatie, zoals databasequeries, laadtijd, routes en gebruikte views.

```
composer require barryvdh/laravel-debugbar --dev
```

□□ **Let op:** De Debugbar is alleen zichtbaar als `APP_DEBUG=true` in je `.env`-bestand staat. Op een live website zet je dit op `false`. Controleer dit eerst!

Start je server met `php artisan serve` en open je applicatie in de browser. Onderin zie je de Debugbar verschijnen.

Tabblad	Wat zie je?
Messages	Berichten die je zelf logt
Timeline	Hoe lang het duurde om de pagina te laden
Route	Welke route en controller gebruikt worden
Queries	Alle databasequeries die worden uitgevoerd
Views	Welke Blade-views worden geladen

□□ Commit met het bericht *Debugbar package geïnstalleerd* en Sync via Source Control.

□□ Inleveren

1. Screenshot van de Debugbar onderin je browser met de welkomstpagina zichtbaar.
2. Screenshot van het tabblad **Queries** in de Debugbar.

Les 2: Authenticatie met Laravel Breeze

□□ Leerdoelen

- Je kunt uitleggen wat authenticatie is.
- Je kunt Laravel Breeze installeren.
- Je kunt gebruikers registreren en controleren in de database.

□□ Uitleg

Authenticatie is controleren of jij echt bent wie je zegt dat je bent. Dit gebeurt meestal met een e-mailadres en wachtwoord. Hierdoor kan een applicatie bepalen welke gebruiker is ingelogd.

Laravel Breeze is een officiële package van Laravel die een compleet authenticatiesysteem toevoegt. Na installatie heb je pagina's voor registreren, inloggen, uitloggen, wachtwoord vergeten en een dashboard.

afbeelding.png

□□ Opdracht – Laravel Breeze installeren

Voer de volgende commando's uit in je terminal:

```
composer require laravel/breeze --dev
php artisan breeze:install blade
```

Kies bij eventuele opties voor de standaardopties door op Enter te drukken.

```
npm install
npm run build
php artisan migrate
```

□□ Let op: Breeze overschrijft je routes

Na het installeren van Laravel Breeze kan het gebeuren dat je bestaande bestand `routes/web.php` wordt aangepast of deels wordt overschreven. Hierdoor kunnen routes die je eerder hebt gemaakt, bijvoorbeeld voor je Game Collection CRUD, verdwijnen.

Gebruik daarom na de installatie van Breeze altijd de **Git Difference View** in Source Control. Hiermee kun je precies zien welke regels zijn toegevoegd, gewijzigd of verwijderd.

Zie je dat oude routes zijn verdwenen? Kopieer deze routes dan terug uit de oude versie en combineer ze met de nieuwe Breeze-routes. Zo behoud je je bestaande CRUD-routes én voeg je de authenticatie van Breeze toe.

Start je server met `php artisan serve` en open `http://localhost:8000`. Rechtsboven zie je de links **Log in** en **Register**.

Registreer twee gebruikers:

Gebruiker	E-mailadres	Rol (later)
Klant-account	<jouwnaam>@klant.nl	klant
Admin-account	<jouwnaam>@admin.nl	admin

Kies zelf een wachtwoord van minimaal 8 tekens en onthoud dit wachtwoord.

Opdracht - Commit

Commit met het bericht 'Breeze authenticatie geïnstalleerd' en Sync via Source Control.

Inleveren

1. Screenshot van je browser met de inlog- en registreerknoppen zichtbaar.
2. Screenshot van phpMyAdmin met de tabel **users**, waarin beide accounts zichtbaar zijn.

Les 3: Middleware: pagina’s beschermen

Leerdoelen

- Je kunt uitleggen wat middleware is.
- Je kunt routes beveiligen met `auth`-middleware.
- Je kunt testen of een pagina alleen voor ingelogde gebruikers zichtbaar is.

Uitleg

Middleware is een filter dat wordt uitgevoerd voordat een pagina wordt getoond. Je kunt het vergelijken met een beveiliger bij de ingang: voordat je binnenkomt, wordt gecontroleerd of je een geldig pasje hebt.

afbeelding.png

Laravel heeft standaard de `auth`-middleware. Deze controleert of een gebruiker is ingelogd. Is dat niet zo, dan wordt de gebruiker doorgestuurd naar het inlogscherm.

Dit regel je in je routes bestand. /routes/web.php

```
// Alleen ingelogde gebruikers mogen het dashboard zien
Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware('auth');
```

Je kunt ook meerdere routes tegelijk beschermen met een groep:

```
Route::middleware('auth')->group(function () {
    Route::get('/dashboard', function () {
        return view('dashboard');
    });

    Route::get('/profiel', function () {
        return view('profiel');
    });
});
```

□□ **Let op:** Test dit in een incognito-venster. Ga naar `/dashboard`, log in en probeer het opnieuw.

□□ Opdracht – Route /geheim beveiligen

1. Maak een nieuwe route `/geheim` met een simpele Blade-view en bescherm deze route met de `auth`-middleware.
2. Commit met het bericht *Middleware auth toegepast* en Sync via Source Control.

□□ Inleveren

1. Screenshot: bezoek `/geheim` zonder ingelogd te zijn en toon het inlogscherm.
2. Screenshot: bezoek `/geheim` terwijl je ingelogd bent en toon de pagina.

Les 4: Game Collection achter een login

□□ Leerdoelen

- Je kunt bestaande CRUD-routes beveiligen.

- Je kunt bepalen welke acties alleen voor ingelogde gebruikers zijn.
- Je kunt de beveiliging testen in de browser.

□□ Uitleg

Je gaat je bestaande Game Collection CRUD beveiligen met de `auth`-middleware. Iedereen mag het overzicht bekijken, maar alleen ingelogde gebruikers mogen games toevoegen, bewerken en verwijderen.

Beveiligingsregels:

- Iedereen mag de overzichtspagina met alle games bekijken.
- Alleen ingelogde gebruikers mogen games toevoegen, bewerken en verwijderen.

```
// Iedereen mag het overzicht zien
Route::get('/games', [GameController::class, 'index']);

// Alleen ingelogde gebruikers
Route::middleware('auth')->group(function () {
    Route::get('/games/create', [GameController::class, 'create']);
    Route::post('/games/store', [GameController::class, 'store']);
    Route::get('/games/edit/{id}', [GameController::class, 'edit']);
    Route::post('/games/update/{id}', [GameController::class, 'update']);
    Route::post('/games/destroy/{id}', [GameController::class, 'destroy']);
});
```

□□ **Let op:** Het is niet erg als niet-ingelogde bezoekers de knoppen **Edit** en **Delete** nog zien. Als ze klikken, worden ze doorgestuurd naar het inlogscherm.

□□ Opdracht – Game Collection beveiligen

1. Pas de routes van je Game Collection aan zodat create, update en delete beschermd zijn met `auth`-middleware.
2. Commit met het bericht *Games CRUD beveiligd met auth* en Sync via Source Control.

□□ Inleveren

1. Kort filmpje: laat het game-overzicht zien en toon dat je eerst moet inloggen om een game toe te voegen.

Les 5: Spatie Permission installeren

□□ Leerdoelen

- Je kent het verschil tussen authenticatie en autorisatie.
- Je kunt Spatie Laravel Permission installeren.
- Je kunt het User-model voorbereiden op rollen en permissies.

□□ Uitleg

Authenticatie betekent: wie ben jij? Autorisatie betekent: wat mag jij doen? Een admin mag bijvoorbeeld alles, terwijl een klant alleen mag bekijken.

Concept	Vraag	Voorbeeld
Authenticatie	Wie ben jij?	Inloggen met e-mail en wachtwoord
Autorisatie	Wat mag jij doen?	Admin mag alles, klant mag alleen bekijken

Rollen vertegenwoordigen een groep gebruikers, zoals **admin**, **klant** of **editor**. Permissies zijn acties die aan rollen worden toegewezen, zoals **product bekijken** of **product verwijderen**.

□□ Opdracht – Spatie Laravel Permission installeren

Voer de volgende commando's uit:

```
composer require spatie/laravel-permission
php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"
php artisan migrate
```

Open `app/Models/User.php` en voeg de trait toe:

```
// Bovenaan bij de andere use-statements:
use Spatie\Permission\Traits\HasRoles;

// In de class:
class User extends Authenticatable
{
```

```
    use HasFactory, Notifiable, HasRoles;
}
```

Na de migratie zijn er vijf belangrijke tabellen aangemaakt:

Tabel	Wat staat erin?	Voorbeeld
users	Gebruikersaccounts	Piet, Truus
roles	Soorten gebruikers	admin, klant
permissions	Mogelijke acties	product bekijken, product verwijderen
role_has_permissions	Koppeling: welke rol heeft welke permissie	admin → product verwijderen
model_has_roles	Koppeling: welke gebruiker heeft welke rol	Piet → klant

Commit met het bericht *Spatie Permission geïnstalleerd* **en Sync via Source Control.**

Inleveren

1. Screenshot van phpMyAdmin met de vijf tabellen zichtbaar: **users**, **roles**, **permissions**, **role_has_permissions** en **model_has_roles**.

Les 6: Rollen en permissies instellen

Leerdoelen

- Je kunt rollen en permissies handmatig aanmaken in phpMyAdmin.
- Je kunt permissies koppelen aan rollen.
- Je kunt rollen koppelen aan gebruikers.

Uitleg

In deze les stel je via phpMyAdmin de rollen en permissies in je database in. Later bouw je hiervoor een webinterface, maar eerst leer je hoe de tabellen samenwerken.

◫ Opdracht – Rollen en permissies koppelen

Stap 1: Rollen aanmaken

id	name	guard_name
1	klant	web
2	admin	web

⚠ **Let op:** Vul bij `guard_name` altijd `web` in.

Stap 2: Permissies aanmaken

id	name	guard_name
1	product bekijken	web
2	product bestellen	web
3	product invoeren	web
4	product aanpassen	web
5	product verwijderen	web
6	bestellingen bekijken	web

Stap 3: Permissies koppelen aan rollen

Klant (`role_id = 1`) mag producten bekijken en bestellen.

permission_id	role_id
1 (product bekijken)	1 (klant)
2 (product bestellen)	1 (klant)

Admin (`role_id = 2`) mag alles.

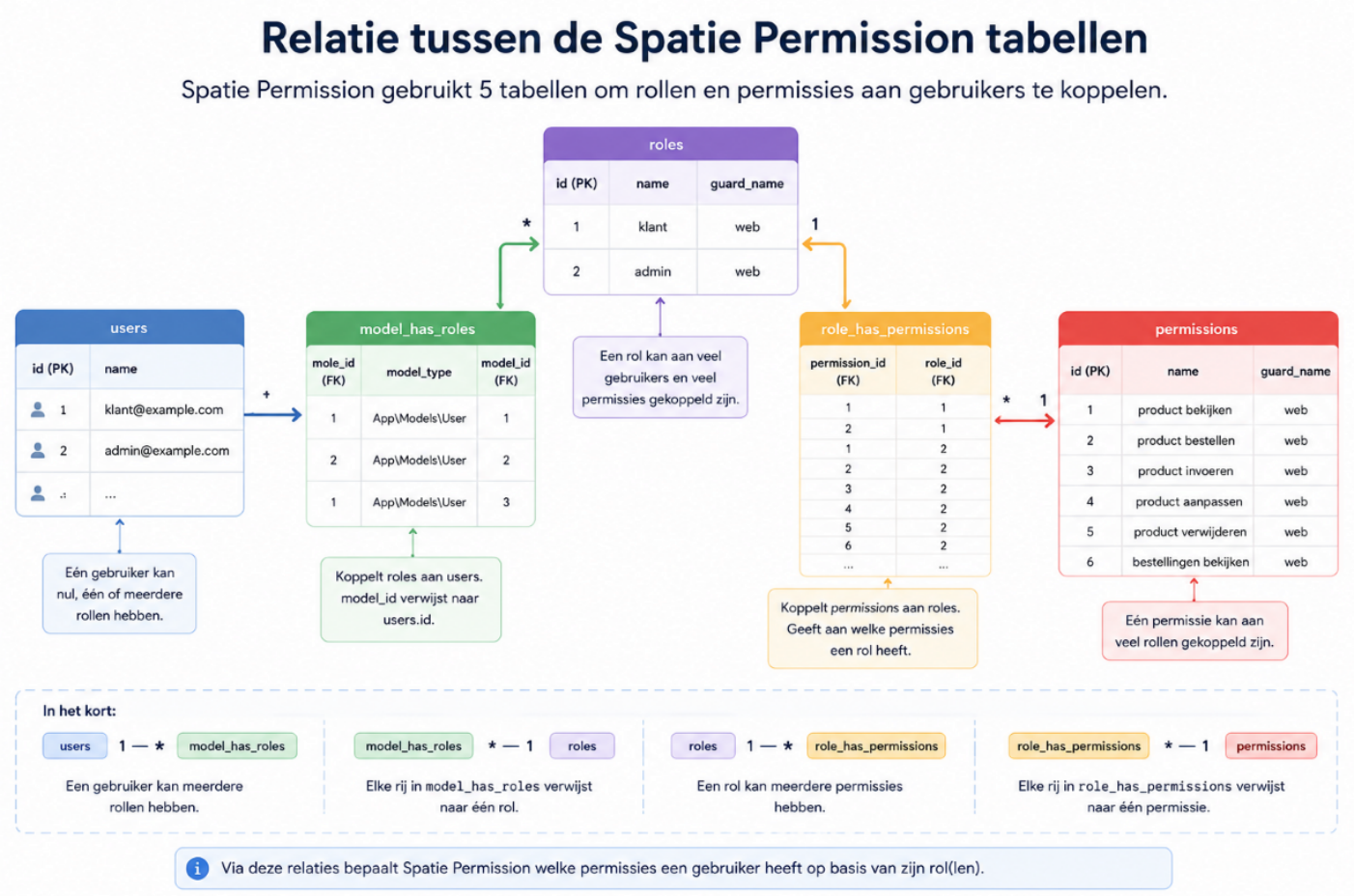
permission_id	role_id
1 (product bekijken)	2 (admin)
2 (product bestellen)	2 (admin)
3 (product invoeren)	2 (admin)
4 (product aanpassen)	2 (admin)
5 (product verwijderen)	2 (admin)
6 (bestellingen bekijken)	2 (admin)

Stap 4: Rollen koppelen aan gebruikers

role_id	model_type	model_id

1 (klant)	App\Models\User	<id van klant-account>
2 (admin)	App\Models\User	<id van admin-account>

Let op: Kijk in de tabel **users** naar de id van het klant-account en het admin-account. Deze id's gebruik je als `model_id`.



Opdracht - voer alle gegevens in de database, zoals die in bovenstaande tabellen staan.

Inleveren

1. Screenshot van de tabel **roles** met de ingevoerde rollen.
2. Screenshot van de tabel **permissions** met de ingevoerde permissies.
3. Screenshot van de tabel **role_has_permissions** met de koppelingen.
4. Screenshot van de tabel **model_has_roles** met de koppelingen.

Les 7: Blade directives: @role en @can

□□ Leerdoelen

- Je kunt Blade directives gebruiken voor rollen.
- Je kunt Blade directives gebruiken voor permissies.
- Je kunt knoppen tonen of verbergen op basis van rechten.

□□ Uitleg

Nu je rollen en permissies hebt ingesteld, kun je deze gebruiken in je Blade-views. Hiermee toon of verberg je onderdelen op basis van de rol of permissies van de ingelogde gebruiker.

Op basis van rollen: @role

```
@role('admin')
    <p>Welkom beheerder! Je hebt volledige toegang.</p>
@endrole

@role('klant')
    <p>Welkom klant! Je kunt producten bekijken en bestellen.</p>
@endrole
```

Op basis van permissies: @can en @cannot

```
@can('product aanpassen')
    <a href="/product/edit/{{ $product->id }}" class="btn btn-primary">
        Bewerken
    </a>
@endcan

@cannot('product aanpassen')
    <p>Je hebt geen rechten om dit product aan te passen.</p>
@endcannot
```

□□ **Let op:** Gebruik `@can` voor specifieke acties, zoals knoppen verbergen. Gebruik `@role` voor grotere secties, zoals een menu.

□□ Opdracht – Knoppen tonen per rol

1. Pas de overzichtspagina van je Game Collection aan.

2. Gebruikers met de rol **klant** zien geen knoppen voor toevoegen, bewerken of verwijderen.
3. Gebruikers met de rol **admin** zien alle knoppen.
4. Commit met het bericht *Blade directives role en can* en Sync via Source Control.

Inleveren

1. Screenshot: ingelogd als `<jouwnaam>@klant.nl` zonder bewerk- en verwijderknoppen.
2. Screenshot: ingelogd als `<jouwnaam>@admin.nl` met alle knoppen.

Les 8: Route middleware voor rollen

Leerdoelen

- Je kunt routes beveiligen op basis van rollen.
- Je kunt routes beveiligen op basis van permissies.
- Je kunt een 403 Forbidden-melding herkennen als geen-toegang-melding.

Uitleg

In de vorige les heb je onderdelen op een pagina verborgen met Blade directives. Maar een slimme gebruiker kan nog steeds een URL rechtstreeks intypen. Daarom beveilig je nu ook de routes op basis van rollen en permissies.

Role middleware in routes

```
// Alleen admins mogen naar het admin-dashboard
Route::get('/admin/dashboard', function () {
    return view('admin.dashboard');
})->middleware('role:admin');

// Zowel admins als klanten mogen deze pagina zien
Route::get('/catalogus', function () {
    return view('catalogus');
})->middleware('role:admin|klant');
```

Permission middleware in routes

```
Route::get('/product/create', [ProductController::class, 'create'])
    ->middleware('permission:product invoeren');

// Meerdere permissies toestaan
->middleware('permission:product bekijken|bestellingen bekijken');
```

Groepen van routes

```
Route::middleware(['auth', 'role:admin'])->group(function () {
    Route::get('/admin/dashboard', [AdminController::class, 'index']);
    Route::get('/admin/users', [AdminController::class, 'users']);
});

Route::middleware(['auth', 'role:klant'])->group(function () {
    Route::get('/bestellingen', [OrderController::class, 'index']);
});
```

⚠ **Let op:** Als een gebruiker niet de juiste rol of permissie heeft, toont Laravel een **403 Forbidden** -pagina. Dit is normaal gedrag.

□□ Opdracht – Game Collection beveiligen met rollen

1. Pas de routes van je Game Collection aan zodat klanten alleen het game-overzicht kunnen bekijken en admins op alle pagina's kunnen komen.
2. De klant mag alleen het game-overzicht bekijken.
3. De admin mag op alle pagina's komen.
4. Commit met het bericht *Route middleware rollen en permissies* en Sync via Source Control.

□□ Inleveren

1. Kort filmpje van maximaal 1 minuut waarin je laat zien dat de klant op sommige pagina's wordt geweigerd en de admin overal bij kan.

Les 9: Eindopdracht – Beheeromgeving

□□ Leerdoelen

- Je kunt een beheeromgeving bouwen voor rollen en permissies.
- Je kunt CRUD's maken voor Spatie-tabellen.
- Je kunt een beheeromgeving beveiligen met `role:admin`.

□□ Uitleg

Tot nu toe heb je rollen en permissies handmatig ingevoerd via phpMyAdmin. Dat is niet handig als je veel rollen, permissies en gebruikers hebt. Daarom bouw je als eindopdracht een beheeromgeving waar alleen admins bij kunnen.

afbeelding.png

De beheeromgeving bestaat uit vier CRUD's:

1. **Permissies beheren**: tabel `permissions`. Invoerveld: naam. `guard_name` wordt automatisch `web`.
2. **Rollen beheren**: tabel `roles`. Invoerveld: naam. `guard_name` wordt automatisch `web`.
3. **Permissie koppelen aan rol**: tabel `role_has_permissions`. Selecteer een permissie-ID en een rol-ID.
4. **Rol koppelen aan gebruiker**: tabel `model_has_roles`. Selecteer een rol-ID en een gebruiker-ID.

Extra eisen

- Maak een navigatiemenu met links naar de vier overzichtspagina's van de CRUD's.
- Beveilig alle pagina's van de beheeromgeving met route middleware: `role:admin`.
- Commit regelmatig na elk werkend onderdeel en sync met GitHub.

□□ **Let op:** Bouw de CRUD's één voor één. Begin met de eenvoudigste CRUD, bijvoorbeeld permissies beheren, en commit na elke werkende CRUD.

□□ Opdracht – Beheeromgeving bouwen

1. Bouw een beheeromgeving waarin de admin rollen, permissies en koppelingen kan beheren via de browser.
2. Bouw de vier CRUD's in de beheeromgeving.
3. Maak een navigatiemenu met links naar de vier overzichtspagina's.
4. Beveilig alle beheer-routes met `role:admin`-middleware.

Inleveren

1. Lever een tekstje in waarin je aangeeft dat je klaar bent voor een eindgesprek.
2. Bereid je voor op het eindgesprek.
3. Vraag een eindgesprek aan bij een docent.

Revision #4
Created 2026-06-15 10:07:59 UTC by Aron
Updated 2026-06-17 10:22:52 UTC by Aron