Database notification in laravel 11

Maitrik Thakkar · Follow

5 min read · Jul 5, 2024

What are Notifications in Laravel?

Notifications in Laravel provide a standardized way to send messages to users concerning events or actions that may require their attention or

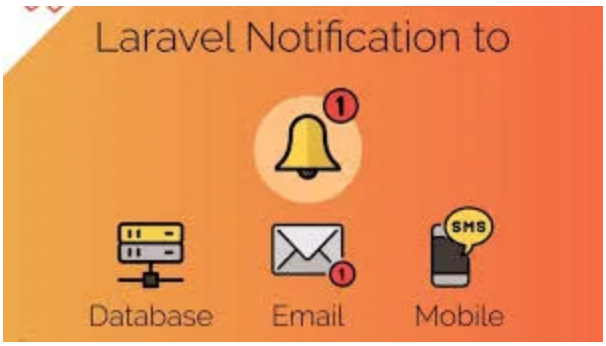
response. These messages can be delivered through multiple channels, including email, SMS, or even via the application's interface. Laravel's notification system simplifies the process of sending notifications by providing a unified API that abstracts the underlying communication mechanisms, making it easy for developers to implement and manage notifications in their applications.

Types of Notifications in Laravel:

Laravel supports various types of notifications, each tailored to specific communication channels:

1. **Mail Notifications:** Email notifications are the most common type and involve sending email messages to users. Laravel provides a convenient way to define and send mail notifications using its Mail facade.
2. **SMS Notifications:** SMS notifications allow you to send text messages to users' mobile phones. Laravel integrates with SMS service providers to facilitate the sending of SMS notifications.
3. **Database Notifications:** Database notifications are stored in the database and can be accessed through the application's interface. This type of notification is useful for displaying messages within the application itself, such as in a user's dashboard or notification center.

4. **Broadcast Notifications:** Introduced in Laravel 5.1, broadcast notifications enable real-time communication with users who are actively connected to the application. These notifications are sent through websockets and can be used to provide instant updates or alerts.
5. **Custom Notifications:** Laravel allows you to create custom notification classes to suit specific communication needs. This flexibility empowers developers to tailor notifications to their application's requirements.



In these article we Discuss about Database Notifications

When New user Register We store notification in database and display in admin side.

Step 1: Create New Project

```
composer create-project laravel/laravel --prefer-dist laravel-notification
```

step 2: Add column In user table

```
php artisan make:migration add_usertype_fields_to_users_table --table=users
```

Go inside the database\migrations\xxxxadd_usertype_fields_to_users_table.php

replace the code

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->string("usertype")->default('user');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn("usertype");
        });
    }
};
```

Run Migration

```
php artisan migrate
```

create admin record using seeder

database\seeders\UserSeeder.php

```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Str;
use Illuminate\Support\Facades\Hash;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        DB::table('users')->insert([
            'name' => 'admin',
            'email' => 'admin@gmail.com',
            'password' => Hash::make(12345678),
            'usertype' => 'admin'
        ]);
    }
}
```

Run seeder

```
php artisan db:seed --class=UserSeeder
```

Step 3: Create notification table using command

```
php artisan notifications:table
```

Implementing Notifications in Laravel:

Implementing notifications in Laravel is a straightforward process:

1. **Create Notification Class:** Define a notification class that extends Laravel's `Notification` class. In this class, specify the notification's content and the channels through which it should be delivered.
2. **Notify Users:** Use the `notify` method on the user model to send the notification. You can specify multiple users as recipients, and Laravel will handle the delivery process.
3. **Display Notifications:** Depending on the notification type, Laravel provides various ways to display notifications to users. For example, email notifications can be displayed in the user's email inbox, while database notifications can be displayed within the application's interface

Step 4: Create Notification

```
php artisan make:notification RegistrationSuccessful
```

Add Following code in app\Notifications\RegistrationSuccessful.php

```
<?php

namespace App\Notifications;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Notifications\Messages\MailMessage;
use Illuminate\Notifications\Notification;

class RegistrationSuccessful extends Notification
{
    use Queueable;
    public $data;

    /**
     * Create a new notification instance.
     */
    public function __construct($data)
    {
        $this->data = $data;
    }

    /**
     * Get the notification's delivery channels.
     *
     * @return array<int, string>
     */
    public function via(object $notifiable): array
    {
        return ['database'];
    }

    /**
     * Get the array representation of the notification.
     *
     * @return array<string, mixed>
     */
    public function toArray(object $notifiable): array
```

```
    {
        return [
            'data' => $this->data->name.' register successfully'
        ];
    }
}
```

Add Following code in AuthController.php

```
use App\Notifications\RegistrationSuccessful;

public function registerpost(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'name' => 'required | string | max:255',
            'email' => 'required | string | email | max:255 | unique:users',
            'password' => 'required | string | min:8 | max:15',
            'hobbies' => 'nullable | array',
        ]);

        if ($validator->fails()) {
            return redirect()->back()
                ->withErrors($validator)
                ->withInput();
        }

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
            'hobbies' => $request->hobbies,
        ]);
        if($user){

            $adminUsers = User::where('usertype', 'admin')->get();
            foreach ($adminUsers as $admin) {
                $admin->notify(new RegistrationSuccessful($user));
            }
            return redirect('/login')->with('success', 'Registration successful'
        }else{
            return redirect('/register')->with("failed","something went rong!");
        }
    }
```

```
    public function markAsRead()
    {
          Auth::user()->unreadNotifications->markAsRead();
          return redirect()->back();
     }
```

add following Route in web.php

```
et('/mark-as-read', [AuthController::class,'markAsRead'])->name('mark-as-read');
```

Add Following code in home.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>Document</title>
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awes
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.m

   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.b
</head>
<body>
   <li class="nav-item dropdown">
      <a id="navbarDropdown" class="nav-link " href="#" role="button" data-bs-to
          <i class="fa fa-bell"></i>
          <span class="badge badge-light bg-success badge-xs">{{auth()->user()->
      </a>
      <ul class="dropdown-menu">
                 @if (auth()->user()->unreadNotifications)
                 <li class="d-flex justify-content-end mx-1 my-2">
                     <a href="{{route('mark-as-read')}}" class="btn btn-success
```

```
                    </li>
                    @endif

                    @foreach (auth()->user()->unreadNotifications as $notification
                    <a href="#" class="text-success"><li class="p-1 text-success">
                    @endforeach
                    @foreach (auth()->user()->readNotifications as $notification)
                    <a href="#" class="text-secondary"><li class="p-1 text-seconda
                    @endforeach
        </ul>
    </li>

    <?php $name = Auth::user()->name ?>
    <div class="container">
        <h3> Hello {{$name}}</h3>
    </div>
</body>
</html>
```

Output:

I hope it helps you...

Laravel11 Laravel Notification Laraveldbnotification

Written by Maitrik Thakkar

28 Followers

Follow

More from Maitrik Thakkar

Maitrik Thakkar

Polymorphic Relationships In Laravel

In Laravel's Eloquent ORM, a morph relationship (or polymorphic relationship)...

Mar 23 1

Maitrik Thakkar

Email notification in laravel 11

What are Notifications in Laravel?

Jul 24 4

Maitrik Thakkar

Laravel 10 and React authentication Part 2

Hello Everyone, In these article We Will Work on User Registration and Login.

Jan 21 6

Maitrik Thakkar

Laravel one to one Relationship Full Guide

In Laravel, a one-to-one relationship is a type of Eloquent relationship where a single recor...

Jan 21 1

See all from Maitrik Thakkar

Recommended from Medium

Achmad Fatoni in Stackademic

Improve laravel performance using swoole: handle 1000 req/s

Hi everyone!!! In this article I will explain how to improve the performance of Laravel by...

Aug 18 16 1

Valerio Barbera

PHP Attributes: how to use PHP Attributes and create custom...

PHP attributes were introduced in PHP 8.0. This version marked a significant milestone...

Aug 23 78

Lists

Staff Picks

727 stories · 1277 saves

Stories to Help You Level-Up at Work

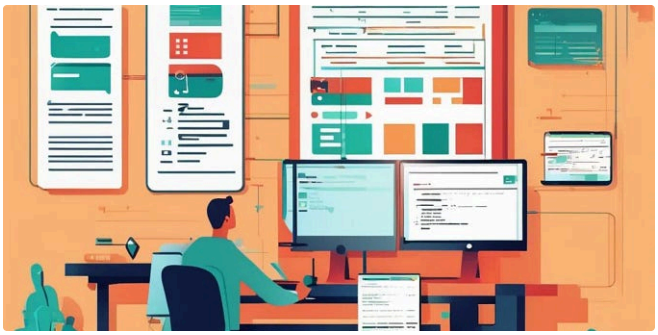
19 stories · 781 saves

Self-Improvement 101

20 stories · 2679 saves

Productivity 101

20 stories · 2298 saves



Olujimi Sanwo

Understanding Laravel Validation: sometimes vs nullable

Sometimes: In some cases, you may want to perform validation checks on a field only if it...

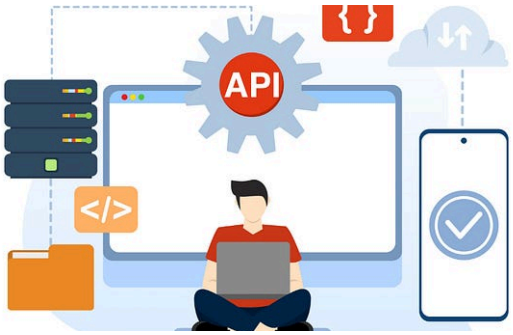
Aug 17

Kawtar mancouri

A Comprehensive Guide to Preventing SQL Injection in Larav...

As developers, it's our duty to ensure the security of our data and code. That's why PH...

Aug 19 15



Abishek in Work Done Right

Transform Your Laravel Application into a Progressive Web App (PWA)

In the ever-evolving world of web development, staying ahead of the curve is...

Apr 5 57

Jos Koomen

11 Ways to Elevate the Quality of Your Laravel Application

Apr 22 31

See more recommendations