🔍 Search Docs

Hashing

- # Introduction
- # Configuration
- # Basic Usage

Introduction

The Laravel `Hash` facade provides secure Bcrypt and Argon2 hashing for storing user passwords. If you are using the built-in `LoginController` and `RegisterController` classes that are included with your Laravel application, they will use Bcrypt for registration and authentication by default.

> 💡 Bcrypt is a great choice for hashing passwords because its "work factor" is adjustable, which means that the time it takes to generate a hash can be increased as hardware power increases.

Configuration

The default hashing driver for your application is configured in the `config/hashing.php` configuration file. There are currently three supported drivers: Bcrypt and Argon2 (Argon2i and Argon2id variants).

> ❗ The Argon2i driver requires PHP 7.2.0 or greater and the Argon2id driver requires PHP 7.3.0 or greater.

Basic Usage

You may hash a password by calling the `make` method on the `Hash` facade:

```php
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UpdatePasswordController extends Controller
{
    /**
     * Update the password for the user.
     *
     * @param  Request  $request
     * @return Response
     */
    public function update(Request $request)
    {
        // Validate the new password length...

        $request->user()->fill([
            'password' => Hash::make($request->newPassword)
        ])->save();
    }
}
```

Adjusting The Bcrypt Work Factor

If you are using the Bcrypt algorithm, the `make` method allows you to manage the work factor of the algorithm using the `rounds` option; however, the default is acceptable for most applications:

```php
$hashed = Hash::make('password', [
    'rounds' => 12
]);
```

Adjusting The Argon2 Work Factor

If you are using the Argon2 algorithm, the `make` method allows you to manage the

work factor of the algorithm using the `memory`, `time`, and `threads` options; however, the defaults are acceptable for most applications:

```php
$hashed = Hash::make('password', [
    'memory' => 1024,
    'time' => 2,
    'threads' => 2,
]);
```

For more information on these options, check out the [official PHP documentation](#).

Verifying A Password Against A Hash

The `check` method allows you to verify that a given plain-text string corresponds to a given hash. However, if you are using the `LoginController` included with Laravel, you will probably not need to use this directly, as this controller automatically calls this method:

```php
if (Hash::check('plain-text', $hashedPassword)) {
    // The passwords match...
}
```

Checking If A Password Needs To Be Rehashed

The `needsRehash` function allows you to determine if the work factor used by the hasher has changed since the password was hashed:

```php
if (Hash::needsRehash($hashed)) {
    $hashed = Hash::make('plain-text');
}
```

```php
$hashed = Hash::make('password', [
    'memory' => 1024,
    'time' => 2,
    'threads' => 2,
]);
```

Become a Laravel Partner

Laravel Partners are elite shops providing top-notch Laravel development and consulting. Each of our partners can help you craft a beautiful, well-architected project.

Our Partners

Laravel

Highlights	Resources	Partners	Ecosystem
Release Notes	Laracasts	Vehikl	Vapor
Getting Started	Laravel News	Tighten Co.	Forge
Routing	Laracon	Kirschbaum	Envoyer
Blade Templates	Laracon EU	Byte 5	Horizon
Authentication	Laracon AU	64Robots	Lumen
Authorization	Jobs	Cubet	Nova
Artisan Console	Certification	DevSquad	Echo
Database	Forums	Ideil	Valet
Eloquent ORM		Cyber-Duck	Mix
Testing		ABOUT YOU	Spark
		Become A Partner	Cashier
			Homestead
			Dusk
			Passport
			Scout
			Socialite

Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable and creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in most web projects.