Laravel : Générer un QR code avec simple-qrcode

Auteur	Domaine	Technologies
Wilo Ahadi	Développement web	Laravel, PHP

Un tutoriel pour créer un code QR (Quick Response) avec le package simplesoftwareio/simple-qrcode dans un projet Laravel.

Sommaire

Introduction

Un **QR code** (QR pour Quick Response) est un type de code-barres en deux dimensions (ou code matriciel) qui se présente généralement sous forme de petits carrés noirs disposés dans un carré à fond blanc.

Voici un code QR, il représente l'URL de ce cours :

Un QR code offre l'avantage de stocker plus d'informations qu'un code-barres classique. Il peut contenir du texte, une adresse web, une carte de visite virtuelle (vCard), les informations d'un réseau WIFI, … Ces données sont directement reconnues par des applications, permettant ainsi de déclencher une action tels que :

- Ouvrir le navigateur (browser) pour visiter une adresse web
- Déclencher un appel téléphone vers un numéro de téléphone ou envoyer un SMS
- Connecter à un réseau WIFI
- Envoyer un courriel
- …

Nous allons voir dans ce guide comment générer (créer) un **code QR** au format **SVG** avec le package **simplesoftareio/simple-qrcode** dans un projet **Laravel**.

Installer simple-qrcode

Le package **simplesofwareio/simple-qrcode** est un wrapper pour Laravel du package PHP **Bacon/BaconQrCode** qui génère les codes QR. Pour l'importer dans un projet Laravel, on exécute la commande *composer* suivante à la racine du projet :

```
composer require simplesoftwareio/simple-qrcode "~4"
```

Créer un QR code avec simple-qrcode

Une fois le package simplesofwareio/simple-qrcode importé dans un projet Laravel, nous pouvons l'utiliser pour générer un code QR. Pour ce faire, nous allons procéder de la manière suivante :

1. Définir la route « simple-qrcode » (GET) pour présenter la vue « simple-qrcode.blade.php » où nous allons afficher le code QR
2. Créer le contrôleur « SimpleQRcodeController » pour décrire l'action « generate » de la route « simple-qrcode » : générer le code QR et l'envoyer à la vue

1. Définissions la route « simple-qrcode » au fichier **routes/web.php** :

```
<?php
```

```
use Illuminate\Support\Facades\Route;

# la route "simple-qrcode"
Route::get("simple-qrcode", "SimpleQRcodeController@generate");
```

2. Générons le contrôleur **SimpleQRcodeController** en exécutant la commande *artisan* suivante :

```
php artisan make:controller SimpleQRcodeController
```

Cette commande crée le fichier **app/Http/Controllers/SimpleQRcodeController.php** que nous allons compléter avec l'action `generate()` de la route « simple-excel » en générant un code QR de 200px qui contient le texte « je suis un code QR ».

Pour générer un code QR, nous avons la façade **SimpleSoftwareIO\QrCode\Facades\QrCode** qui présente les méthodes `generate($texte)` où `$texte` est le contenu du code QR et `size($taille)` où `$taille` est la taille de l'image en pixel.

Le code source du contrôleur **SimpleQRcodeController.php** :

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

# 1. La facade QrCode
use SimpleSoftwareIO\QrCode\Facades\QrCode;

class SimpleQRcodeController extends Controller
{
    // L'action "generate" de la route "simple-qrcode" (GET)
    public function generate () {

     # 2. On génère un QR code de taille 200 x 200 px
     $qrcode = QrCode::size(200)->generate("Je suis un QR Code");

     # 3. On envoie le QR code généré à la vue "simple-qrcode"
     return view("simple-qrcode", compact('qrcode'));
    }
}
```

Affichons le code QR généré sur la vue (template Blade) **resources/views/simple-qrcode.blade.php** :

```
<!DOCTYPE html>
<html>
<head>
 <title>Simple Qrcode</title>
</head>
<body>
 <!-- On affiche le code QR au format SVG -->
 {{ $qrcode }}
```

```
</body>
</html>
```

J’obtiens le rendu suivant au navigateur :

Le code QR généré est au format SVG par défaut. Nous pouvons aussi directement l'enregistrer dans le répertoire **/public** du projet en indiquant le chemin `$chemin` en second paramètre de la méthode `generate($texte, $chemin)` :

```
$qrcode = QrCode::size(200)->generate("Je suis un QR Code", '../public/codes-qr/text
```

Les helpers

Le package simplesoftwareio/simple-qrcode fournit des helpers (fonctions) pour stocker d'autres types d'informations dans un code QR :

Un lien

```
$qrcode = Qrcode::size(200)->generate("https://www.akilischool.com");
```

Un courriel

L'helper `email($email, $subject, $content)` génère un code QR d'un courriel pour l'adresse email `$email` avec le sujet `$subject` et le message `$content` :

```
// Un e-mail avec destinataire, sujet et message
$qrcode = QrCode::size(200)->email("info@akilischool.com", "Salutations", "Bonjour A

// Un e-mail avec destinataire uniquement
$qrcode = QrCode::size(200)->email("info@akilischool.com");

// Un e-mail sans destinataire
$qrcode = QrCode::size(200)->email(null, "Salutations", "Bonjour Akili School ! Quoi
```

Avec le préfixe « mailto: » dans la méthode `generate()` :

```
$qrcode = QrCode::size(200)->generate("mailto:info@akilischool.com");
```

Une position géographique

L'helper `geo($latitude, $longitude)` génère un code QR d'une position en indiquant la latitude `$latitude` et la longitude `$longitude` :

```
$qrcode = QrCode::size(200)->geo(-4.308904, 15.302172);
```

Avec le préfixe « geo: » dans la méthode `generate()` :

```
$qrcode = QrCode::size(200)->generate("geo:-4.308904,15.302172");
```

Un numéro de téléphone

L'helper `phoneNumber($number)` génère un code QR d'un numéro de téléphone `$number` à composer ou ajouter aux contacts :

```
$qrcode = QrCode::size(200)->phoneNumber("+243811111111");

$qrcode = QrCode::size(200)->phoneNumber("666-666-6666");
```

Avec le préfixe « tel: » dans la méthode `generate()` :

```
$qrcode = QrCode::size(200)->generate("tel:+243811111111");
```

Un message texte SMS

L'helper `SMS($phone, $message)` génère un code QR d'un texte `$message` pour le numéro de téléphone `$phone` :

```
$qrcode = QrCode::size(200)->SMS("+243811111111", "Hello World !");
```

Avec le préfixe « sms: » dans la méthode `generate()` :

```
$qrcode = Qrcode::size(200)->generate("sms:+243811111111");
```

Connecter à un réseau Wifi

L'helper `wifi($options)` génère un code QR pour connecter un périphérique à un réseau WiFi :

```
$qrcode = QrCode::wiFi([
    'encryption' => 'WPA', // Cryptage "WPA" ou "WEP"
    'ssid' => 'Akili School', // Nom du réseau WiFi
    'password' => 'Emilie-9876543210.', // Clé de sécurité
    'hidden' => 'false' // Si le réseau WiFi est masqué "true" ou non "false"
]);
```

Seul le paramètre « ssid » est requis. On renseigne les paramètres « encryption » et « password» pour un réseau WiFi sécurisé (avec une clé de sécurité) et « hidden » pour un réseau masqué.

Personnaliser le QR code

Hormis définir la taille du code QR avec la méthode `size($taille)`, le package simple-qrcode fournit les méthodes :

- `format($format)` pour générer le code QR au format `$format`. « svg », « png » et « eps » sont supportés. « imagic » est requis pour générer une image .png
- `color(int $rouge, int $vert, int $bleu, int $alpha = null)` pour changer la couleur du code QR
- `backgroundColor(int $rouge, int $vert, int $bleu, int $alpha = null)` pour changer la couleur de fond du code QR
- `encoding(string $encoding)` pour changer l'encodage des caractères
- `style(string $style, float $size = 0.5)` pour changer le style du code QR en `$style` (square, dot, round)
- ....

Reportez-vous à la documentation pour plus d'informations.

Personnalisons le code QR de l'URL de ce cours :

```
$qrcode = Qrcode::encoding("UTF-8")
        ->color(8, 114, 145)
        ->backgroundColor(245, 234, 62)
        ->size(300)
```

```
        ->generate("https://akilischool.com/cours/laravel-generer-un-qr-code-avec-si
```

Notez bien : La méthode `generate()` vient après toutes les autres méthodes.

A vous les codes QR !