QR Code

By endroid

This library helps you generate QR codes in a jiffy. Makes use of bacon/bacon-qr-code to generate the matrix and khanamiryan/qrcode-detector-decoder for validating generated QR codes. Further extended with Twig extensions, generation routes, a factory and a Symfony bundle for easy installation and configuration. Different writers are provided to generate the QR code as PNG, SVG, EPS or in binary format.

Sponsored by

Installation

Use Composer to install the library. Also make sure you have enabled and configured the GD extension if you want to generate images.

```
$ composer require endroid/qr-code
```

Usage: using the builder

```php
use Endroid\QrCode\Builder\Builder;
use Endroid\QrCode\Encoding\Encoding;
use Endroid\QrCode\ErrorCorrectionLevel\ErrorCorrectionLevelHigh;
use Endroid\QrCode\Label\Alignment\LabelAlignmentCenter;
use Endroid\QrCode\Label\Font\NotoSans;
use Endroid\QrCode\RoundBlockSizeMode\RoundBlockSizeModeMargin;
use Endroid\QrCode\Writer\PngWriter;

$result = Builder::create()
    ->writer(new PngWriter())
    ->writerOptions([])
    ->data('Custom QR code contents')
    ->encoding(new Encoding('UTF-8'))
    ->errorCorrectionLevel(new ErrorCorrectionLevelHigh())
    ->size(300)
    ->margin(10)
    ->roundBlockSizeMode(new RoundBlockSizeModeMargin())
    ->logoPath(__DIR__.'/assets/symfony.png')
    ->labelText('This is the label')
    ->labelFont(new NotoSans(20))
    ->labelAlignment(new LabelAlignmentCenter())
    ->validateResult(false)
    ->build();
```

Usage: without using the builder

```php
use Endroid\QrCode\Color\Color;
use Endroid\QrCode\Encoding\Encoding;
use Endroid\QrCode\ErrorCorrectionLevel\ErrorCorrectionLevelLow;
use Endroid\QrCode\QrCode;
use Endroid\QrCode\Label\Label;
use Endroid\QrCode\Logo\Logo;
use Endroid\QrCode\RoundBlockSizeMode\RoundBlockSizeModeMargin;
use Endroid\QrCode\Writer\PngWriter;
use Endroid\QrCode\Writer\ValidationException;

$writer = new PngWriter();

// Create QR code
$qrCode = QrCode::create('Life is too short to be generating QR codes')
    ->setEncoding(new Encoding('UTF-8'))
    ->setErrorCorrectionLevel(new ErrorCorrectionLevelLow())
    ->setSize(300)
    ->setMargin(10)
    ->setRoundBlockSizeMode(new RoundBlockSizeModeMargin())
    ->setForegroundColor(new Color(0, 0, 0))
    ->setBackgroundColor(new Color(255, 255, 255));

// Create generic logo
$logo = Logo::create(__DIR__.'/assets/symfony.png')
    ->setResizeToWidth(50);

// Create generic label
$label = Label::create('Label')
    ->setTextColor(new Color(255, 0, 0));

$result = $writer->write($qrCode, $logo, $label);
```

```php
// Validate the result
$writer->validateResult($result, 'Life is too short to be generating QR codes');
```

Usage: working with results

```php
// Directly output the QR code
header('Content-Type: '.$result->getMimeType());
echo $result->getString();

// Save it to a file
$result->saveToFile(__DIR__.'/qrcode.png');

// Generate a data URI to include image data inline (i.e. inside an <img> tag)
$dataUri = $result->getDataUri();
```

Writer options

```php
use Endroid\QrCode\Writer\SvgWriter;

$builder->setWriterOptions([SvgWriter::WRITER_OPTION_EXCLUDE_XML_DECLARATION => true])
```

Encoding

If you use a barcode scanner you can have some troubles while reading the generated QR codes. Depending on the encoding you chose you will have an extra amount of data corresponding to the ECI block. Some barcode scanner are not programmed to interpret this block of information. To ensure a maximum compatibility you can use the `ISO-8859-1` encoding that is the default encoding used by barcode scanners (if your character set supports it, i.e. no Chinese characters are present).

Round block size mode

By default block sizes are rounded to guarantee sharp images and improve readability. However some other rounding variants are available.

- `margin (default)` : the size of the QR code is shrunk if necessary but the size of the final image remains unchanged due to additional margin being added.
- `enlarge` : the size of the QR code and the final image are enlarged when rounding differences occur.
- `shrink` : the size of the QR code and the final image are shrunk when rounding differences occur.
- `none` : No rounding. This mode can be used when blocks don't need to be rounded to pixels (for instance SVG).

Readability

The readability of a QR code is primarily determined by the size, the input length, the error correction level and any possible logo over the image so you can tweak these parameters if you are looking for optimal results. You can also check $qrCode->getRoundBlockSize() value to see if block dimensions are rounded so that the image is more sharp and readable. Please note that rounding block size can result in additional padding to compensate for the rounding difference. And finally the encoding (default UTF-8 to support large character sets) can be set to `ISO-8859-1` if possible to improve readability.

Validating the generated QR code

If you need to be extra sure the QR code you generated is readable and contains the exact data you requested you can enable the validation reader, which is disabled by default. You can do this either via the builder or directly on any writer that supports validation. See the examples above.

Please note that validation affects performance so only use it in case of problems.

Symfony integration

The endroid/qr-code-bundle integrates the QR code library in Symfony for an even better experience.

- Configure your defaults (like image size, default writer etc.)
- Support for multiple configurations and injection via aliases
- Generate QR codes for defined configurations via URL like /qr-code//Hello
- Generate QR codes or URLs directly from Twig using dedicated functions

Read the bundle documentation for more information.

Versioning

Version numbers follow the MAJOR.MINOR.PATCH scheme. Backwards compatibility breaking changes will be kept to a minimum but be aware that these can occur. Lock your dependencies for production and test your code when upgrading.

License