URL Generation

# Introduction

Laravel provides several helpers to assist you in generating URLs for your application. These are mainly helpful when building links in your templates and API responses, or when generating redirect responses to another part of your application.

The Basics

Generating Basic URLs

The `url` helper may be used to generate arbitrary URLs for your application. The generated URL will automatically use the scheme (HTTP or HTTPS) and host from the current request:

```
$post = App\Post::find(1);

echo url("/posts/{$post->id}");

// http://example.com/posts/1
```

Accessing The Current URL

If no path is provided to the `url` helper, a `Illuminate\Routing\UrlGenerator` instance is returned, allowing you to access information about the current URL:

```
// Get the current URL without the query string...
echo url()->current();

// Get the current URL including the query string...
echo url()->full();

// Get the full URL for the previous request...
echo url()->previous();
```

Each of these methods may also be accessed via the `URL` facade:

```
use Illuminate\Support\Facades\URL;

echo URL::current();
```

URLs For Named Routes

The `route` helper may be used to generate URLs to named routes. Named routes allow you to generate URLs without being coupled to the actual URL defined on the route. Therefore, if the route's URL changes, no changes need to be made to your `route` function calls. For example, imagine your application contains a route defined like the following:

```
Route::get('/post/{post}', function () {
    //
})->name('post.show');
```

To generate a URL to this route, you may use the `route` helper like so:

```
echo route('post.show', ['post' => 1]);

// http://example.com/post/1
```

You will often be generating URLs using the primary key of Eloquent models. For this reason, you may pass Eloquent models as parameter values. The `route` helper will automatically extract the model's primary key:

```
echo route('post.show', ['post' => $post]);
```

The `route` helper may also be used to generate URLs for routes with multiple parameters:

```
Route::get('/post/{post}/comment/{comment}', function () {
    //
})->name('comment.show');

echo route('comment.show', ['post' => 1, 'comment' => 3]);

// http://example.com/post/1/comment/3
```

Signed URLs

Laravel allows you to easily create "signed" URLs to named routes. These URLs have a "signature" hash appended to the query string which allows Laravel to verify that the URL has not been modified since it was created. Signed URLs are especially useful for routes that are publicly accessible yet need a layer of protection against URL manipulation.

For example, you might use signed URLs to implement a public "unsubscribe" link that is emailed to your customers. To create a signed URL to a named route, use the `signedRoute` method of the `URL` facade:

```
use Illuminate\Support\Facades\URL;

return URL::signedRoute('unsubscribe', ['user' => 1]);
```

If you would like to generate a temporary signed route URL that expires, you may use the `temporarySignedRoute` method:

```
use Illuminate\Support\Facades\URL;

return URL::temporarySignedRoute(
    'unsubscribe', now()->addMinutes(30), ['user' => 1]
);
```

Validating Signed Route Requests

To verify that an incoming request has a valid signature, you should call the `hasValidSignature` method on the incoming `Request`:

```
use Illuminate\Http\Request;

Route::get('/unsubscribe/{user}', function (Request $request) {
    if (! $request->hasValidSignature()) {
        abort(401);
    }

    // ...
})->name('unsubscribe');
```

Alternatively, you may assign the `Illuminate\Routing\Middleware\ValidateSignature` middleware to the route. If it is not already present, you should assign this middleware a key in your HTTP kernel's `routeMiddleware` array:

```
/**
 * The application's route middleware.
 *
 * These middleware may be assigned to groups or used individually.
 *
 * @var array
 */
protected $routeMiddleware = [
    'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
];
```

Once you have registered the middleware in your kernel, you may attach it to a route. If the incoming request does not have a valid signature, the middleware will automatically return a `403` error response:

```
Route::post('/unsubscribe/{user}', function (Request $request) {
    // ...
})->name('unsubscribe')->middleware('signed');
```

URLs For Controller Actions

The `action` function generates a URL for the given controller action. You do not need to pass the full namespace of the controller. Instead, pass the controller class name relative to the `App\Http\Controllers` namespace:

```
$url = action('HomeController@index');
```

You may also reference actions with a "callable" array syntax:

```
use App\Http\Controllers\HomeController;

$url = action([HomeController::class, 'index']);
```

If the controller method accepts route parameters, you may pass them as the second argument to the function:

```
$url = action('UserController@profile', ['id' => 1]);
```

Default Values

For some applications, you may wish to specify request-wide default values for certain URL parameters. For example, imagine many of your routes define a `{locale}` parameter:

```
Route::get('/{locale}/posts', function () {
    //
})->name('post.index');
```

It is cumbersome to always pass the `locale` every time you call the `route` helper. So, you may use the `URL::defaults` method to define a default value for this parameter that will always be applied during the current request. You may wish to call this method from a route middleware so that you have access to the current request:

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\URL;

class SetDefaultLocaleForUrls
{
    public function handle($request, Closure $next)
    {
        URL::defaults(['locale' => $request->user()->locale]);

        return $next($request);
    }
}
```

Once the default value for the `locale` parameter has been set, you are no longer required to pass its value when generating URLs via the `route` helper.

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.