URL generation

The essentials

The MUSTs about this topic, explained as clear as possible

Generating URLs

As a developer of web applications, you often need to add a URL to another route within your application. Typically in:

- Navigation menus or other navigation elements (`<a href="…">`
- Submitting a form (`<form action="…">`)
- Redirecting

a Proper web framework like Laravel should provide ways to help you with this task.

Manually using string concatenation

Most uninformed developers probably will create URL's 'manually', defining them just like fixed test strings, i.e.

```
<a href="/greeting">Hello</a>
```

When referring to routes with parameters, you might even do:

```
<a href="users/{{ $id }}">Link to user {{ $id }}</a>
```

Although this works fine, it has some drawbacks. You are forced to work with relative links (unless you want to introduce a major maintainability issue), which has its own drawbacks, among these are:

- **Potential for Error:** Mistakes in writing relative paths can occur, especially in larger projects with complex directory structures. A small error can result in broken links that are hard to debug.
- **Base URL Changes:** If your website's base URL changes (e.g., moving from HTTP to HTTPS), relative URLs might not adapt automatically, causing broken links or unexpected behavior.

In cases where you can justify that these drawback will not create issues in the future, you can stick to this method. However, Laravel can help you solve this, which is often a good thing to accept.

With the help of Laravel helper functions

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

These helpers are just functions that return a string that represents a valid URL. This URL depends mainly on the parameter value(s) you specified. The `url()` helper is the most basic of the provided methods, and perhaps the most easy to understand. When called, the return value will:

- start with `http` or `https`, pending on the current request that is handled
- not be a ‘relative link’, but uses the current request to find the host
- have the parameter value appended

For example:

```
// When current request was https://example.com/... url('/greeting'); //
returns: https://example.com/greeting
```

Which can be used in a view like:

```
<a href="{{ url('/greeting') }}">Hello</a>
```

Which renders to (when the current request was `https://example.com/...`):

```
<a href="https://example.com/greeting">Hello</a>
```

When the URL routes to a parameterized route (i.e. `/users/{id}`), you need to define it using string concatenation or template strings:

```
// When current request was https://example.com/... // and $post->id = 32
url('/posts/'.$post->id); // using string concatenation
url("/posts/{$post->id}"); // using template string // returns
http://example.com/posts/32
```

💡 The `url()` helper is easy to understand and can avoid many mistakes that novice developers tend to make. Therefore we advise you to use this helper anywhere you can, or consider more advanced helpers (see the Digging Deeper chapter)

Accessing the current URL

It is also useful to know the URL of the *current* request, the URI of the HTTP request that is currently processed. For instance when you want to add an `active` class to a currently rendered menu item, you can do:

```
<a href="..." class="navbar-item {{ url()->current() == url('/about') ?
'is-active' : '' }}"> ... </a>
```

Digging deeper

Named routes

If you understand the `url()` helper, you might consider using 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 calls to the `route` function. For example, imagine your application contains a route defined like the following:

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

Note the `name()` declaration, which defines the name for this route. To generate a URL to this route, you may use the `route` helper like so:

```
echo route('post.show', ['id' => 1]); // http://example.com/post/1
```

The `route` helper is able to read the required URI from the named route and can therefore replace its `id` parameter with the value given. For a better developer experience, Laravel will throw an error when you made a mistake in, for instance:

- The route name
- The parameter declarations

Which help building more robust applications.

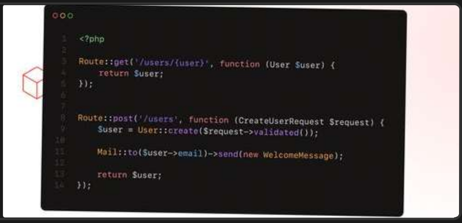
Further reading

You can read about the details of Laravel URL generation here. Specifically the topics listed below:

Laravel - The PHP Framework For Web Artisans

Laravel is a PHP web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you

https://laravel.com/docs/urls



```
<?php

Route::get('/users/{user}', function (User $user) {
    return $user;
});


Route::post('/users', function (CreateUserRequest $request) {
    $user = User::create($request->validated());

    Mail::to($user->email)->send(new WelcomeMessage);

    return $user;
});
```

URLs for controller actions

If you understand named routes, you might want to consider generating URLs based on controller actions.