**Laravel blade and Controller**

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding** |

|  |
| --- |
| **Regarding URL with parameter** |

**✅ All possible ways to build a URL like this in Laravel Blade**

**1. Using url() and string concatenation (your existing example)**

<a href="{{ url('billspendingforcollection?' . 'partnerId=' . $partnerId . '&&' . 'yearly=' . $yearly . '&&' . 'monthsdigit=' . $monthsdigit) }}">

**Notes:**

* Simple and straightforward.
* Works, but error-prone if variables are null or empty.
* Use when you're quickly prototyping or working with static variables.

**2. Using url() with http\_build\_query() (recommended)**

@php

$queryParams = [];

if (!empty($partnerId)) {

$queryParams['partnerId'] = $partnerId;

}

if (!empty($yearly)) {

$queryParams['yearly'] = $yearly;

}

if (!empty($monthsdigit)) {

$queryParams['monthsdigit'] = $monthsdigit;

}

$url = url('billspendingforcollection') . (!empty($queryParams) ? '?' . http\_build\_query($queryParams) : '');

@endphp

<a href="{{ $url }}">

**Notes:**

* Cleaner, more flexible.
* Automatically handles empty or missing parameters.
* Safer than manual concatenation.
* Recommended for maintainability.

**3. Using route() with named routes and array\_filter()**

Define your route in web.php:

Route::get('/billspendingforcollection', [DashboardReport::class, 'billspendingforcollection'])->name('billspendingforcollection');

In your Blade template:

@php

$queryParams = array\_filter([

'partnerId' => $partnerId,

'yearly' => $yearly,

'monthsdigit' => $monthsdigit

]);

@endphp

<a href="{{ route('billspendingforcollection', $queryParams) }}">

**Notes:**

* Best when you have named routes.
* array\_filter() removes empty or null values.
* Cleaner and more Laravel-like.
* Use this approach for larger apps with route caching and better structure.

**4. Using request()->fullUrlWithQuery()**

If you're adding or modifying query parameters on the current URL:

<a href="{{ request()->fullUrlWithQuery([

'partnerId' => $partnerId,

'yearly' => $yearly,

'monthsdigit' => $monthsdigit

]) }}">

**Notes:**

* Extends the current URL.
* Good if you're on a page that already has query parameters and want to add or override them.

**5. Using route() with query() in Laravel 9+**

Laravel provides helper methods like this:

<a href="{{ route('billspendingforcollection')->withQuery(['partnerId' => $partnerId, 'yearly' => $yearly, 'monthsdigit' => $monthsdigit]) }}">

*(Note: This is a pseudo-code; Laravel doesn't support this out-of-the-box but it's achievable with custom macros or extending URL generation.)*

**6. Using Blade components**

You can wrap URL logic inside a Blade component:

<x-link-button :href="route('billspendingforcollection', array\_filter([

'partnerId' => $partnerId,

'yearly' => $yearly,

'monthsdigit' => $monthsdigit

]))">

Click here

</x-link-button>

**Notes:**

* Use this when you want reusable buttons or links across your app.
* Improves readability and reusability.

**7. Using Laravel Collective's Form helpers (if installed)**

{!! link\_to\_route('billspendingforcollection', 'View Bills', [

'partnerId' => $partnerId,

'yearly' => $yearly,

'monthsdigit' => $monthsdigit

]) !!}

**Notes:**

* Requires Laravel Collective package.
* Simplifies link generation.

**8. Using JavaScript if needed**

If URL generation is complex and based on user interactions:

<a href="#" onclick="window.location='{{ url('billspendingforcollection') }}?partnerId={{ $partnerId }}&yearly={{ $yearly }}&monthsdigit={{ $monthsdigit }}'">

View Bills

</a>

**Notes:**

* Avoid unless necessary.
* Useful for dynamically updating URL based on form inputs or buttons.

**✅ Summary Table**

| **Method** | **Laravel-friendly?** | **Handles empty params?** | **Best for** |
| --- | --- | --- | --- |
| url() + concat | ✔ | ❌ | Quick prototypes |
| url() + http\_build\_query() | ✔✔ | ✔ | Most flexible for queries |
| route() + array\_filter() | ✔✔ | ✔ | Best structured, named routes |
| request()->fullUrlWithQuery() | ✔✔ | ✔ | Extend current URL |
| Blade components | ✔✔ | ✔ | Reusable links/buttons |
| Laravel Collective | ✔ | ✔ | If package installed |
| JS onclick | ❌ | ❌ | Interactive or dynamic changes |

**✅ Conclusion**

You can create URLs in multiple ways depending on the use case:

1. Use **url() with http\_build\_query()** for simple query building.
2. Prefer **named routes with route() and array\_filter()** for structured and maintainable code.
3. Use **request()->fullUrlWithQuery()** when working with existing URLs.
4. Consider **components or packages** for reusable or advanced solutions.

Pick the approach based on your project size, team practices, and readability!