Asynchronous Calls

In Laravel, you can make **asynchronous calls** in several ways depending on whether you want to run **PHP code asynchronously on the server** or **make async HTTP requests from the client** .

Here’s a breakdown with examples:

1. Asynchronous HTTP Requests (Server-Side)

If you want to call an API without blocking execution, you can use **Guzzle’s async methods**.

```
use GuzzleHttp\Client;
use GuzzleHttp\Promise;

Route::get('/async-api', function () {
    $client = new Client();

    // Create multiple async requests
    $promise1 = $client->getAsync('https://jsonplaceholder.typicode.com/posts/1');
    $promise2 = $client->getAsync('https://jsonplaceholder.typicode.com/posts/2');

    // Wait for all promises to complete
    $results = Promise\Utils::unwrap([$promise1, $promise2]);

    return [
        'post1' => json_decode($results[0]->getBody(), true),
        'post2' => json_decode($results[1]->getBody(), true),
    ];
});
```

 Non-blocking until you explicitly wait for results.

2. Asynchronous Jobs (Background Processing)

If you want to run heavy tasks without blocking the request, use **Laravel Queues**.

```
// Create a job
php artisan make:job ProcessReportJob
```

`app/Jobs/ProcessReportJob.php`

```
namespace App\Jobs;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Queue\SerializesModels;

class ProcessReportJob implements ShouldQueue
{
    use InteractsWithQueue, Queueable, SerializesModels;

    public function handle()
    {
        // Heavy processing logic here
        \Log::info("Report processed asynchronously.");
    }
}
```

Dispatch the job asynchronously:

Php

```
use App\Jobs\ProcessReportJob;

Route::get('/run-job', function () {
    ProcessReportJob::dispatch();
```

```
    return 'Job dispatched!';
});
```

Run the queue worker:

Bash

```
php artisan queue:work
```

 This is the **Laravel-recommended** way for async server tasks.

3. Laravel 10+ / 11+ Process Async Execution

Laravel's `Process` facade can run commands asynchronously.

```
use Illuminate\Support\Facades\Process;

Route::get('/async-process', function () {
    $process = Process::start('php artisan schedule:run');

    // Do other work here while process runs
    return 'Process started!';
});
```

 Useful for running shell commands without blocking.

4. Client-Side Async Calls (AJAX / Axios)

If you just want async calls from the browser, use **JavaScript**:

```
axios.get('/api/data')
    .then(response => {
```

```
        console.log(response.data);
    })
    .catch(error => {
        console.error(error);
    });
```

 This keeps the Laravel backend fast and responsive.

□□ Best Practice:

- For **long-running PHP tasks** → Use **Queues**.
- For **parallel API calls** → Use **Guzzle async**.
- For **non-blocking shell commands** → Use **Laravel Process**.
- For **frontend async** → Use **AJAX / Axios / Fetch API**.

Revision #1
Created 10 January 2026 15:20:02 by AI Channel
Updated 10 January 2026 15:21:26 by AI Channel