Laravel request lifecycle in simple terms

The Laravel request lifecycle **describes the series of steps the framework takes to handle an HTTP request and return a response**. This process ensures essential services are loaded and the correct application logic is executed in an organized manner
The key steps in the lifecycle are:
Entry Point Every request is directed by the web server (Apache or Nginx) to the single entry point: the `public/index.php` file. This script loads the Composer autoloader to make all necessary classes available and retrieves an instance of the Laravel application.

Kernel The incoming request is then sent to the HTTP kernel (`App\Http\Kernel` in `app/Http/Kernel.php`). The kernel acts as a central dispatcher and defines an array of "bootstrappers" that run first to configure error handling, logging, and the application environment.

Middleware The HTTP kernel is also responsible for routing the request through the application's global middleware stack. Middleware are filters that can inspect, modify, or even reject the request (e.g., for authentication, CSRF verification, or maintenance mode checks) before it reaches the core application logic.

Service Providers A crucial bootstrapping action is loading the application's service providers. Service providers are the "heart" of the framework; they register and configure all of Laravel's components, such as the database, queue, and routing services, making them available in the service container.

Routing Once the application is fully bootstrapped, the request is handed to the router. The router matches the incoming request's URL and HTTP method to a specific route defined in files like `routes/web.php` or `routes/api.php`.

Controller/Logic Execution If a matching route is found, the associated controller method or closure is executed. This is where your application's business logic runs, typically involving fetching data from the database (using Models) and preparing the data for the final output.

Response The controller returns a response (which could be an HTML view, JSON data, or a redirect). This response object then travels back through any route-specific middleware, allowing for final modifications.

Sending the Response Finally, the HTTP kernel's `handle` method returns the response, which is then sent back to the user's browser or client by the `public/index.php` file, completing the

lifecycle

Revision #1
Created 9 March 2026 04:44:18 by AI Channel
Updated 9 March 2026 04:45:01 by AI Channel