

Домашняя работа
по предмету:

Продвинутое программирование на PHP — Laravel

Выполнил: Байборodin Александр

Урок 8. Сервисы: создание и использование

Цели практической работы:

Научиться:

- создавать свои сервисы на Laravel;
- работать с логами Laravel и их обработкой.

Что нужно сделать:

В этой практической работе вы разработаете сервис логирования, который:

- фиксирует обращения к сайту;
- собирает их в базе данных с возможностью отключения системы логирования;
- отражает в реальном времени HTTP-запросы к приложению.

Создадим новый проект:

```
composer create-project laravel/laravel log-service
```

1. Для начала создадим модель логов. Для создания модели необходимо использовать **artisan** с параметром **make:model**.

В итоге наша команда будет выглядеть так:

```
php artisan make:model Log
```

По умолчанию модель создаётся в `./app/Models/Log.php`.

Модель создана, для избежания ошибок запросов SQL необходимо отключить автоматические метки времени.

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent
6
7  class Log extends Model
```

2. Теперь опишем миграцию для создания нашей таблицы логов:

```
php artisan make:migration create_logs_table
```

Напомним, что таблицы миграции создаются по умолчанию в
/database/migration/current_date_time_create_logs_table.php.

По умолчанию создаётся файл, содержимое которого выглядит так:

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('logs', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }
}

```

В этом файле нам нужно определить поля, которые будет собирать наш сервис логирования:

- time — время события;
- duration — длительность;
- IP — IP-адрес зашедшего пользователя;
- url — адрес, который запросил пользователь;
- method — HTTP-метод (GET, POST);
- input — передаваемые параметры.

В итоге файл должен приобрести такой вид:

```

<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('logs', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->dateTime('time');
            $table->integer('duration');
            $table->string('ip',100)->nullable();
            $table->string('url')->nullable();
            $table->string('method',10)->nullable();
            $table->string('input')->nullable();
        });
    }
};

```

3. Миграция создана, параметры описаны. Теперь создадим таблицу.

Напоминаем, что таблица создаётся также через **artisan** с параметром **migrate**
php artisan migrate.

4. База данных подготовлена, теперь нужно создать звено (middleware) для обработки HTTP-запросов. Напоминаем, что звенья создаются при помощи команды **php artisan make:middleware название модели**.

В нашем случае нам нужна команда:
php artisan make:middleware DataLogger

По умолчанию звено (посредник) создастся по пути
 ./app/Http/Middleware/DataLogger.php.

Теперь необходимо настроить middleware. Открываем Datalogger.php. Добавим использование созданной модели.

```

1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use App\Models\Log;
7 class DataLogger
8 {
9     private $start_time;
10    /**
11     * Handle an incoming request.
12     *
13     * @param \Illuminate\Http\Request $request
14     * @param \Closure(\Illuminate\Http\Request): (\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse) $next
15     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
16     */
17    public function handle($request, Closure $next)
18    {
19        $this->start_time = microtime(true);
20        return $next($request);
21    }
22    public function terminate($request, $response) //Функция, которая вызывается после завершения запроса
23    {
24        if ( env('API_DATA_LOGGER', true) ) { //Если в env файле прописана опция API_DATA_LOGGER
25            if ( env('API_DATA_LOGGER_USE_DB', true) ) { //Если в env файле прописана опция API_DATA_LOGGER_USE_DB
26                //Записываем данные в базу данных
27            }
28            else //На всякий случай, если опция записи в БД недоступна пишем в файл
29            {
30                $endTime = microtime(true);
31                $filename = 'api_data_logger_' . date('d-m-y') . '.log';
32                $dataToLog = "Time: " . gmdate("F j, Y, g:i a") . "\n";
33                $dataToLog .= "Duration: " . number_format($endTime - LAZARUS_START, 3) . "\n";
34                $dataToLog .= "IP Address: " . $request->ip() . "\n";
35                $dataToLog .= "URL: " . $request->fullUrl() . "\n";
36                $dataToLog .= "Method: " . $request->method() . "\n";
37                $dataToLog .= "Input: " . $request->getContent() . "\n";
38            }
39        }
40    }
41 }

```

Также нужно завершить создание middleware DataLogger, зарегистрировать его в ./app/Http/Kernel.php.

```

protected $middleware = [
    // \App\Http\Middleware\TrustHosts::class,
    \App\Http\Middleware\TrustProxies::class,
    \Illuminate\Http\Middleware\HandleCors::class,
    \App\Http\Middleware\PreventRequestsDuringMaintenance::class,
    \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
    \App\Http\Middleware\TrimStrings::class,
    \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,
];

```

5. Модель создана, посредник HTTP-запросов настроен и зарегистрирован как класс в Kernel.php. Если сейчас запустить Laravel командой `php artisan serv`, всё будет работать. Логи будут записываться в базу данных.

Но увидеть это можно только в самой базе SQL. Для получения более наглядных результатов необходимо создать в web.php эндпоинт.

```
Route::get('/logs',function()
{
    return view('logs');
});
```

Также для этого эндпоинта необходимо создать blade-шаблон:
./resource/view/logs.blade.php

В нём создать запрос к базе SQL и вывод логов в таблицу.

```
1 <!DOCTYPE html>
2 <!--html style="background: #2C282C;"-->
3 <head>
4     <meta charset="UTF-8">
5     <title>Лог</title>
6     <style>
7         td:nth-child(5),td:nth-child(6){text-align:center;}
8         table{position: absolute; border-spacing: 0;border-collapse: collapse;width: 70%;box-
9         td, th{padding: 10px;border: 1px solid #282828;}
10        tr:nth-child(odd) { background-color: #C18787; }
11    </style>
12 </head>
13 <body>
14 <?php
15 $db_server = "127.0.0.1";
16 $db_user = "root";
17 $db_password = "";
18 $db_name = "laravel";
19
20
21
22
23
24
25
26
27
28
29 $statement->execute();
30
31 $result_array = $statement->fetchAll();
32
33 echo "<div class='table'>";
34 echo "<table><tr><th>id</th><th>time</th><th>duration</th><th>ip</th><th>url</th><th>method</th><th>input</th></tr>";
35 foreach ($result_array as $result_row) {
36     echo "<tr>";
37     echo "<td align='center'>" . $result_row["id"] . "</td>";
38     echo "<td align='center'>" . $result_row["time"] . "</td>";
39     echo "<td align='center'>" . $result_row["duration"] . "</td>";
40     echo "<td align='center'>" . $result_row["ip"] . "</td>";
41     echo "<td align='center'>" . $result_row["url"] . "</td>";
42     echo "<td align='center'>" . $result_row["method"] . "</td>";
43     echo "<td align='center'>" . $result_row["input"] . "</td>";
44     echo "</tr>";
45 }
46 echo "</table>";
47 echo "</div>";
48
```

Запускаем приложение, при открытии вашего приложения <http://localhost:8000/logs> должна открываться таблица с логами обращения к сайту.

Что оценивается:

Принято:

- Выполнены все основные пункты работы.
- Сервис запускается без ошибок.
- Логи HTTP-запросов фиксируются в БД.
- Логи отображаются в веб-интерфейсе в реальном времени.

Решение:

Создадим новый проект Laravel:

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson8
$ composer create-project laravel/laravel log-ser
Creating a "laravel/laravel" project at "./log-se
Cannot use laravel/laravel's latest version v12.0
^8.2 which is not satisfied by your platform.
Installing laravel/laravel (v10.3.3)
- Installing laravel/laravel (v10.3.3): Extracti
Created project in C:\laravelapp\lesson8\log-ser
> @php -r "file_exists('.env') || copy('.env.exam
Loading composer repositories with package inform
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0
- Locking brick/math (0.12.3)
- Locking carbonphp/carbon-doctrine-types (2.1.
- Locking dflydev/dot-access-data (v3.0.3)
- Locking doctrine/inflector (2.0.10)
- Locking doctrine/lexer (3.0.1)
- Locking dragonmantank/cron-expression (v3.4.0)
- Locking egulias/email-validator (4.0.3)
- Locking fakerphp/faker (v1.24.1)
- Locking filp/whoops (2.17.0)
- Locking fruitcake/php-cors (v1.3.0)
- Locking graham-campbell/result-type (v1.1.3)
- Locking guzzlehttp/guzzle (7.9.2)
- Locking guzzlehttp/promises (2.0.4)
- Locking guzzlehttp/psr7 (2.7.0)
- Locking guzzlehttp/uri-template (v1.0.4)
- Locking hamcrest/hamcrest-php (v2.0.1)
- Locking laravel/framework (v10.48.28)
```


Generating optimized autoload files

```
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
```

INFO Discovering packages.

```
laravel/sail .....
laravel/sanctum .....
laravel/tinker .....
nesbot/carbon .....
nunomaduro/collision .....
nunomaduro/termwind .....
spatie/laravel-ignition .....
```

81 packages you are using are looking for funding. Use the `composer fund` command to find out more!

```
> @php artisan vendor:publish --tag=laravel-assets
```

INFO No publishable resources for tag [laravel-assets]

Настроим файл для соединения с базой .env:

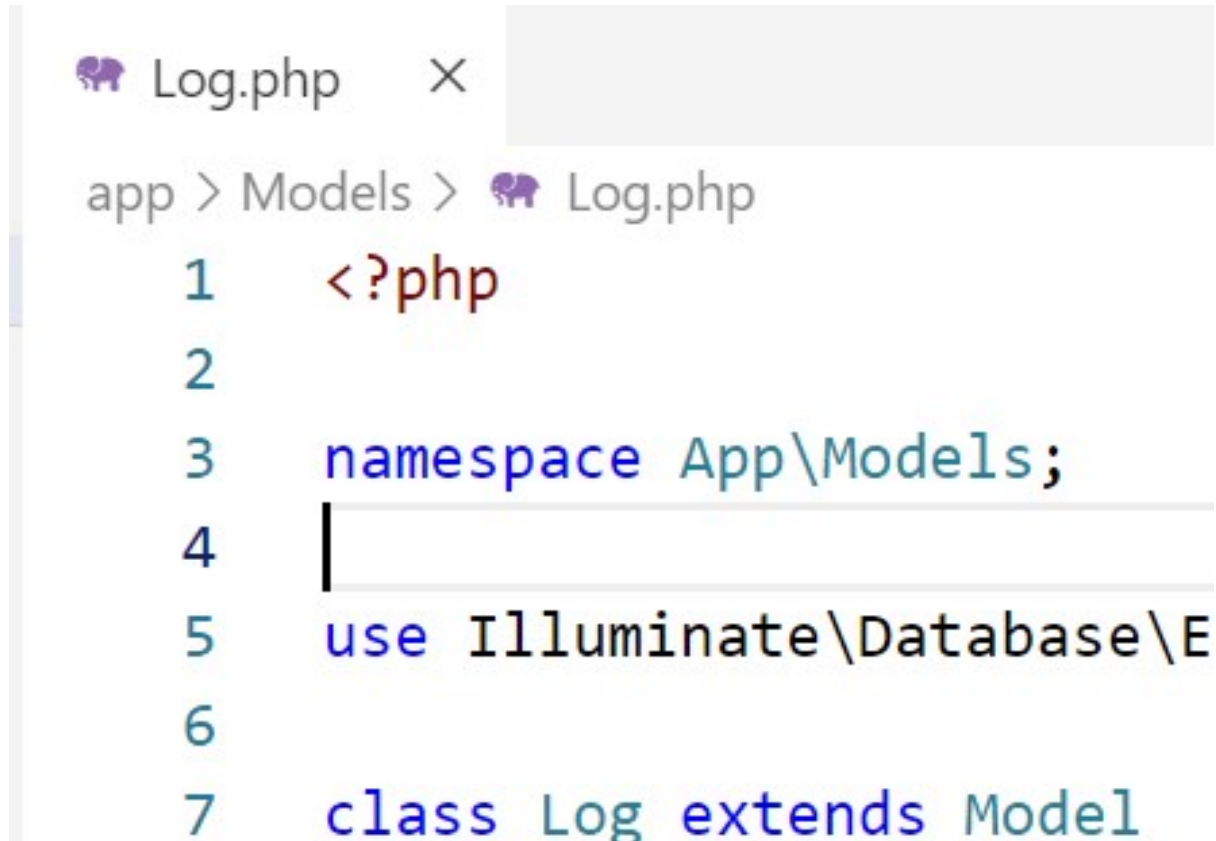
```
.env
C: > laravelapp > lesson8 > log-service > .env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:iEa15RwKpB7P5asSuRH6mgp
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
```

Создадим модель Log с помощью Artisan:


```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson8
$ php artisan make:model Log
```

INFO Model [C:\laravelapp\lesson8\log-service] created successfully.

В файле app/Models/Log.php отключим автоматические метки времени:



```
Log.php X
app > Models > Log.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent
6
7  class Log extends Model
```

Создадим миграцию для таблицы logs:

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson8
$ php artisan make:migration create_logs_table
```

INFO Migration [C:\laravelapp\lesson8\log-service] created successfully.

В файле миграции определим необходимые поля для таблицы:

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
```

```

public function up(): void
{
    Schema::create('logs', function (Blueprint $table) {
        $table->id();
        $table->dateTime('time');
        $table->integer('duration');
        $table->string('ip', 100)->nullable();
        $table->string('url')->nullable();
        $table->string('method', 10)->nullable();
        $table->text('input')->nullable();
    });
}

/**
 * Reverse the migrations.
 */
public function down(): void
{
    Schema::dropIfExists('logs');
}
};

```

Выполним миграцию:

```

A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson1:
$ php artisan migrate

```

```

INFO Running migrations.

```

```

2025_03_10_063251_create_logs_table .....

```



Создадим Middleware DataLogger:

```
A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/lesson8/log-service
$ php artisan make:middleware DataLogger
```

```
INFO Middleware [C:\laravelapp\lesson8\log-service\app\Http\Middleware\
```

В файле app/Http/Middleware/DataLogger.php добавим логику для записи логов:

```
<?php
```

```
namespace App\Http\Middleware;
```

```
use Closure;
```

```
use App\Models\Log;
```

```
class DataLogger
```

```
{
```

```
    private $start_time;
```

```
    public function handle($request, Closure $next)
```

```
    {
```

```
        $this->start_time = microtime(true);
```

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

```

    }

    public function terminate($request, $response)
    {
        if (env('API_DATALOGGER', true)) {
            if (env('API_DATALOGGER_USE_DB', true)) {
                $endTime = microtime(true);
                $log = new Log();
                $log->time = date("Y-m-d H:i:s");
                $log->duration = number_format($endTime -
LARAVEL_START, 3);
                $log->ip = $request->ip();
                $log->url = $request->fullUrl();
                $log->method = $request->method();
                $log->input = $request->getContent();
                $log->save();
            }
        }
    }
}

```

Зарегистрируем Middleware в app/Http/Kernel.php:

 Kernel.php X

app > Http >  Kernel.php

```

8      {
9          /**

```

Добавим маршрут `/logs` в routes/web.php:

```
<?php
```

```

use Illuminate\Support\Facades\Route;
use App\Models\Log;

```

```

Route::get('/', function () {
    return view('welcome');
});

```

```

Route::get('/logs', function () {
    $logs = Log::all();
    return view('logs', ['logs' => $logs]);
});

```

Создадим Blade-шаблон resources/views/logs.blade.php:

```

<!DOCTYPE html>
<html>
<head>
    <title>Logs</title>

```

```

</head>
<body>
    <table border="1">
        <thead>
            <tr>
                <th>ID</th>
                <th>Time</th>
                <th>Duration</th>
                <th>IP</th>
                <th>URL</th>
                <th>Method</th>
                <th>Input</th>
            </tr>
        </thead>
        <tbody>
            @foreach($logs as $log)
            <tr>
                <td>{{ $log->id }}</td>
                <td>{{ $log->time }}</td>
                <td>{{ $log->duration }}</td>
                <td>{{ $log->ip }}</td>
                <td>{{ $log->url }}</td>
                <td>{{ $log->method }}</td>
                <td>{{ $log->input }}</td>
            </tr>
            @endforeach
        </tbody>
    </table>
</body>
</html>

```

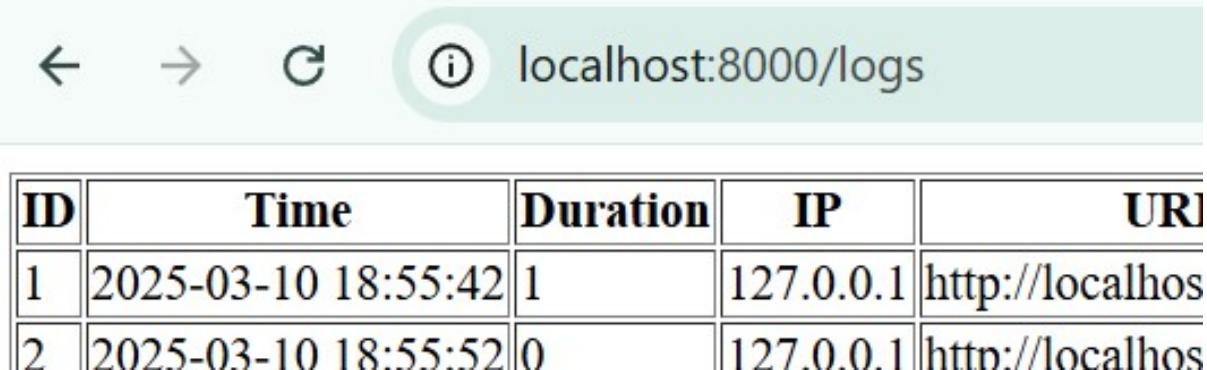
Запустим сервер:

```

A@LAPTOP-RHI13GV6 MINGW64 /c/laravelapp/
$ php artisan serve

```

Перейдем по адресу <http://localhost:8000/logs>, чтобы увидеть таблицу с логами:



ID	Time	Duration	IP	URL
1	2025-03-10 18:55:42	1	127.0.0.1	http://localhos
2	2025-03-10 18:55:52	0	127.0.0.1	http://localhos