Introduction

Standard-Compliant HTTP Library with Extended Features

Anonymous · 09/09/2026

Standard-Compliant HTTP Library with Extended Features

A PSR and RFC compliant HTTP library that provides elegant request/response handling, content negotiation and problem details for PHP applications.

Why Derafu\Http?

Simple, but not limited, HTTP Handling

Most HTTP libraries either do too much or too little. Derafu\Http provides:

- **Extended Request/Response**: Smart content type negotiation and safe data access.
- **Content Negotiation**: Intelligent format detection and response transformation.
- **Problem Details**: RFC 7807 implementation for structured error handling.
- **PSR Compliance**: Built on PSR-7 and PSR-15 standards.
- **Middleware Architecture**: Flexible request/response processing pipeline.

Key Features

- **Smart Request Handling**: Safe access to query, post, and JSON data.
- **Intelligent Responses**: Automatic content negotiation and format transformation.
- **Structured Errors**: Complete RFC 7807 Problem Details implementation.
- **Modular Design**: Core HTTP functionality with middleware support.
- **Type Safety**: Use of enums for HTTP status codes and content types.
- **Middleware Pipeline**: PSR-15 compliant middleware chain for request processing.

Installation

```
composer require derafu/http
```

Basic Usage

public/index.php

```
use Derafu\Http\Kernel;
use Derafu\Kernel\Environment;

require_once dirname(__DIR__) . '/app/bootstrap.php';

return fn (array $context): Kernel => new Kernel(new Environment(
    $context['APP_ENV'],
    (bool) $context['APP_DEBUG'],
    $context
));
```

Middleware Configuration

The library uses PSR-15 middlewares for request processing. Configure your middleware stack in the services file, for example `services.yaml`:

```
# Core middlewares in required order.
Psr\Http\Server\RequestHandlerInterface:
    class: Derafu\Http\Service\RequestHandler
    public: true
    arguments:
        $middlewares:
            - '@Derafu\Http\Middleware\RequestFactoryMiddleware'
            - '@Derafu\Http\Middleware\RouterMiddleware'
            - '@Derafu\Http\Middleware\DispatcherMiddleware'
            - '@Derafu\Http\Middleware\ResponseNormalizerMiddleware'

# Register individual middlewares.
Derafu\Http\Middleware\RequestFactoryMiddleware: ~
Derafu\Http\Middleware\RouterMiddleware: ~
Derafu\Http\Middleware\DispatcherMiddleware: ~
Derafu\Http\Middleware\ResponseNormalizerMiddleware: ~
```

Core Middlewares

The library includes four essential middlewares that must be configured in order:

1. **RequestFactoryMiddleware**: Converts PSR-7 requests to Derafu requests.
2. **RouterMiddleware**: Handles URL routing and route matching.
3. **DispatcherMiddleware**: Executes route handlers (controllers, closures or templates).

4. **ResponseNormalizerMiddleware**: Ensures PSR-7 compliant responses.

Custom Middlewares

Create custom middlewares by implementing PSR-15's `MiddlewareInterface`:

```
class CustomMiddleware implements MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ): ResponseInterface {
        // Process request.
        $response = $handler->handle($request);
        // Process response.
        return $response;
    }
}
```

Working with Requests

```
// Safe access to request data.
$id = $request->query('id', 0);
$data = $request->json();
$file = $request->file('document');

// Content negotiation.
$format = $request->getPreferredFormat();
```

Creating Responses

```
// Automatic content negotiation.
// Returns JSON for API requests in `/api` paths
return ['status' => 'ok'];

// Explicit responses.
return (new Response())->asJson($data)->withHttpStatus(HttpStatus::CREATED);

// Redirect.
return (new Response())->redirect('https://www.example.com');
```

Error Handling with Problem Details

Any exception will be treated as "Problem Detail" (RFC 7807). To have control over all fields, exceptions must implement `HttpExceptionInterface`.

```
{
    "type": "about:blank",
    "title": "Not Found",
    "status": 404,
    "detail": "No route found for \"/api/mustFail\".",
    "instance": "/api/mustFail",
    "extensions": {
        "timestamp": "2025-02-20T22:04:25+00:00",
        "environment": "dev",
        "debug": true,
        "context": [],
        "throwable": null
    }
}
```

Integration with Other Packages

Derafu\Http is designed to work with other Derafu packages:

Required

- **derafu/kernel**: The core of the application, with dependency injection.
- **derafu/renderer**: For template rendering, with dependency on **derafu/twig**.
- **derafu/routing**: For URL routing, with autodiscovery of templates.
- **derafu/translation**: For I18n, with a simple translator that supports ICU.

Optional

- **derafu/markdown**: For renderig templates in markdown format.

Suggested

- **derafu/data-processor**: For data processing: format, cast, sanitize and validate.