

Requests and Responses

Requests and Responses

Anonymous · 08/10/2026

Requests and Responses

`Derafu\Http\Request` and `Derafu\Http\Response` wrap PSR-7 messages and add shortcuts for the things an application does all the time: reading input, negotiating the format and building common responses.

They are mutable

Request

`Request` implements PSR-7's `ServerRequestInterface`, so everything from PSR-7 is available. It is created by `RequestFactoryMiddleware`, which means a handler or any middleware after it receives a `Derafu\Http\Request`.

Reading input

```
$id = $request->query('id', 0);           // From the query string.
$name = $request->post('name');           // From the parsed body (form).
$token = $request->header('X-Token', ''); // First value of a header.

if ($request->isJson()) {
    $data = $request->json();              // Decoded array, or null.
}

$all = $request->all();
$email = $request->input('email', '');
$some = $request->only(['name', 'email']);
```

Method	Returns
<code>query(\$key, \$default)</code>	A value from the query string
<code>post(\$key, \$default)</code>	A value from the parsed body, or the default when the body is not an array
<code>header(\$name, \$default)</code>	The first value of a header, or the default
<code>isJson()</code>	Whether <code>Content-Type</code> contains <code>application/json</code>

Method	Returns
<code>json()</code>	The decoded JSON body as an array, or <code>null</code> when the request is not JSON, the body is invalid, or it does not decode to an array
<code>all()</code>	Query, parsed body and JSON body merged, in that order: a later source wins on the same key
<code>input(\$key, \$default)</code>	A value from <code>all()</code>
<code>only(\$keys)</code>	The given keys from <code>all()</code>

Files

```
if ($request->hasFile('document')) {
    $file = $request->file('document');    // PSR-7 UploadedFileInterface.
}
$files = $request->files();
```

`hasFile()` is `true` only when the file was uploaded without error. `file()` returns `null` when there is no such file.

Application context

Method	Returns
<code>route()</code>	The <code>RouteMatchInterface</code> stored by <code>RouterMiddleware</code>
<code>session()</code>	The session, stored by the session middleware
<code>flash()</code>	The flash messages, stored by the flash middleware
<code>user()</code>	The authenticated user, or <code>null</code>

`route()`, `session()` and `flash()` throw a `LogicException` when the middleware that provides them did not run before. `user()` never throws.

Content negotiation

The request decides which format the client wants, and the response, errors included, follow it. `getPreferredContentType()` checks, in this order:

1. The extension in the URL: `/report.json` means JSON, `/page.html` means HTML.
2. Whether it is an API request, that is the path is `/api` or starts with `/api/`: JSON.
3. Whether it is an XHR request (`X-Requested-With: XMLHttpRequest`): JSON.
4. The `Accept` header, ordered by its `q` values. The first of `application/json`, `text/html` or `text/plain` that appears wins.
5. HTML, when there is no `Accept` header or none of those is in it.

`getPreferredFormat()` returns the same as a subtype string (`json`, `html`, ...). `isApiRequest()` and

`isXmlHttpRequest()` expose the individual checks.

Response

```
use Derafu\Http\Enum\HttpStatus;
use Derafu\Http\Response;

return (new Response())
    ->asJson(['id' => 42])
    ->withHttpStatus(HttpStatus::CREATED);
```

Method	What it does
<code>asJson(\$data, \$flags)</code>	JSON body with Content-Type: <code>application/json</code> . Throws a <code>JsonException</code> if the data can not be encoded. The default flags do not escape slashes or unicode
<code>asText(\$data, \$type)</code>	Text body; <code>\$type</code> is a <code>ContentType</code> , <code>ContentType::PLAIN</code> by default
<code>asHtml(\$html)</code>	HTML body
<code>redirect(\$url, \$status)</code>	Location header and empty body. <code>HttpStatus::FOUND</code> (302) by default
<code>withHttpStatus(\$status)</code>	Sets the status code and reason phrase from an <code>HttpStatus</code>
<code>withContentType(\$type)</code>	Sets Content-Type, adding <code>charset=UTF-8</code> to text types

Returning data instead of a response

A handler does not have to build a `Response`. If it returns an array, a string or an object, `ResponseNormalizerMiddleware` turns it into one using the preferred content type of the request. See [Middleware](#).

```
public function show(string $id): array
{
    // Sent as JSON to an API client.
    return ['id' => $id];
}
```

Build the `Response` yourself when you need control over the status or the headers.

Enums

ContentType

`Derafu\Http\Enum\ContentType` lists the media types the library knows. Besides its `value` (the media

type string) it provides:

Method	Purpose
<code>fromFilename(\$name), fromExtension(\$ext)</code>	Find the type from a file name or extension; <code>OCTET_STREAM</code> when unknown
<code>withCharset(\$charset)</code>	The media type with <code>; charset=...</code> appended
<code>getMainType(), getSubType()</code>	<code>application</code> and <code>json</code> for <code>application/json</code>
<code>isText(), isImage(), isAudio(), isVideo(), isFont()</code>	Group checks
<code>isStatic()</code>	Whether it is a known file type: everything except <code>OCTET_STREAM</code> . It is what <code>StaticFilesMiddleware</code> uses

HttpStatus

`Derafu\Http\Enum\HttpStatus` has a case for each status code, for example `HttpStatus::NOT_FOUND`. It provides `getReasonPhrase()` and the checks `isInformational()`, `isSuccessful()`, `isRedirection()`, `isClientError()`, `isServerError()` and `isError()`.

Last updated on 08/10/2026