Docs

Spreadsheet Project

Derafu Spreadsheet

Anonymous · 24/08/2026 · #php

Table of Contents

Spreadsheet Project

Derafu Spreadsheet

Anonymous · 24/08/2026 · #php

Derafu Spreadsheet

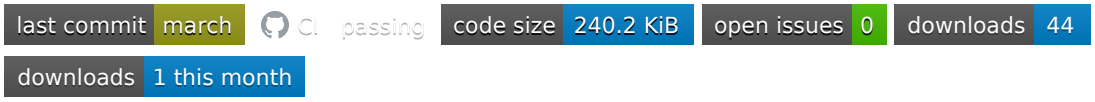
Last updated on 24/08/2026

Introduction

Unified Spreadsheet Processing for PHP

Anonymous · 24/08/2026

Unified Spreadsheet Processing for PHP



Derafu Spreadsheet is a modern PHP library that provides a **single, consistent API** for working with spreadsheet files in multiple formats (XLSX, CSV, ODS, JSON, XML, YAML, and more).

Features

- **Unified API** across all file formats.
- **Multiple format support**: XLSX, XLS, CSV, ODS, JSON, XML, YAML, HTML, PDF.
- **Smart type casting**: Automatically detects and converts data types (dates, numbers, booleans, JSON).
- **Format-agnostic data manipulation**: Work with your data consistently regardless of source format.
- **Minimal dependencies**: Use only what you need.
- **PSR-7 compatible**: Generate HTTP responses with downloadable spreadsheets.
- **Modern PHP**: Written for PHP 8 with strict typing.

Why Derafu Spreadsheet?

While powerful libraries like PhpSpreadsheet exist, and this library use it under de hood, they often require different approaches for different formats and have complex APIs. Derafu Spreadsheet offers several key advantages:

- **Simplified API**: Work with all spreadsheet formats through a consistent interface.
- **Intelligent type handling**: Focus on your data, not type conversion.
- **Format abstraction**: Write your code once, and it works with any format.
- **Flexible format handlers**: Easily switch between formats or implement custom handlers.
- **Minimal learning curve**: Clean, intuitive API with sensible defaults.

Installation

```
composer require derafu/spreadsheet
```

For specific format support, you may need additional dependencies:

```
# For CSV support with League CSV (recommended).
composer require league/csv

# For XLSX, XLS, ODS, HTML support.
composer require phpoffice/phpspreadsheet

# For YAML support.
composer require symfony/yaml

# For HTTP response generation.
composer require nyholm/psr7
```

Basic Usage

```
<?php

use Derafu\Spreadsheet\SpreadsheetLoader;
use Derafu\Spreadsheet\SpreadsheetDumper;

// Load a spreadsheet (format auto-detected from extension).
// Under the hood it will create default Factory and Caster instances. If you
// don't want that, inject your implementation.
$loader = new Loader();
$spreadsheet = $loader->loadFromFile('data.xlsx');

// Access data from sheets.
$sheet = $spreadsheet->getSheet('Sheet1');
$rows = $sheet->getRows();

// Modify data.
$sheet->setCell(0, 0, 'Updated value');
$spreadsheet->createSheet('NewSheet', [['Header1', 'Header2'], [1, 2]]);

// Save in different format.
// Under the hood it will create default Factory and Caster instances. If you
// don't want that, inject your implementation.
$dumper = new Dumper();
$dumper->dumpToFile($spreadsheet, 'output.csv');

// Or convert to string
$jsonString = $dumper->dumpToString($spreadsheet, 'json');
```

Working with Different Formats

Derafu Spreadsheet handles format conversion automatically:

```
// Load Excel file.
$spreadsheet = $loader->loadFromFile('data.xlsx');

// Save as CSV.
$dumper->dumpToFile($spreadsheet, 'data.csv');

// Save as JSON.
$dumper->dumpToFile($spreadsheet, 'data.json');

// Save as YAML.
$dumper->dumpToFile($spreadsheet, 'data.yaml');
```

Intelligent Type Casting

One of Derafu Spreadsheet's key features is automatic type casting for both reading and writing:

```
$spreadsheet = $loader->loadFromFile('data.csv');

// String '123' is automatically cast to integer 123.
// 'true' is cast to boolean true.
// '2025-03-12' is cast to DateTimeImmutable object.
// JSON strings are parsed to indexed arrays or associative arrays ("objects").

$cell = $sheet->getCell(0, 0); // Typed data, not just strings.

// When writing, types are automatically converted to appropriate formats.
$dumper->dumpToFile($spreadsheet, 'output.xlsx');
```

Last updated on 24/08/2026

Architecture

Architecture

Anonymous · 24/08/2026

Architecture

This document provides an overview of the architecture and design principles behind Derafu Spreadsheet.

Core Components

Derafu Spreadsheet is built around several key components that work together to provide a unified approach to spreadsheet processing:

Key Components

1. **Loader**
    - Handles loading spreadsheet data from files or strings.
    - Uses Factory to create appropriate Format Handlers.
    - Uses Caster to convert raw data to appropriate PHP types.

2. **Dumper**
    - Handles saving spreadsheet data to files or strings.
    - Uses Factory to create appropriate Format Handlers.
    - Uses Caster to convert PHP types to format-appropriate representations.
3. **Factory**
    - Creates and manages Format Handlers based on file extension or specified format.
    - Allows registering custom Format Handlers.
    - Provides format detection capabilities.
4. **Caster**
    - Converts raw data values to appropriate PHP types when reading.
    - Converts PHP types to appropriate string representations when writing.
    - Handles intelligent date, boolean, numeric, and JSON detection.
5. **Format Handlers**
    - Format-specific implementations that know how to read/write a particular format.
    - All implement a common FormatHandlerInterface.
    - Includes handlers for XLSX, XLS, CSV, ODS, JSON, XML, YAML, HTML, PDF.
6. **Data Model**
    - **Spreadsheet**: Top-level container for all data.
    - **Sheet**: Container for rows and cells with a name.
    - Both implement interfaces for consistent interaction.

Design Principles

Derafu Spreadsheet was designed with several key principles in mind:

1. **Unified API**
    - Consistent interface across all supported formats.
    - Same code works regardless of source or target format.
2. **Type Intelligence**
    - Automatic conversion between string values and appropriate PHP types.
    - No manual type casting required in application code.
3. **Separation of Concerns**

    - Format handling is separated from data model.
    - Type conversion is separated from data loading/saving.
    - Each component has a single, clear responsibility.

4. **Interface-Based Design**

    - All components define and implement interfaces.
    - Allows for custom implementations and extensions.

5. **Minimal Dependencies**

    - Core functionality has minimal dependencies.
    - Format-specific dependencies only required for formats you use.

Data Flow

When working with Derafu Spreadsheet, data flows through the components as follows:

Loading Process:

1. **Loader** receives a file or string.
2. **Factory** creates appropriate Format Handler based on format.
3. **Format Handler** reads raw data into internal Spreadsheet structure.
4. **Caster** converts raw values to appropriate PHP types.
5. Typed **Spreadsheet** object is returned to application.

Saving Process:

1. **Dumper** receives a Spreadsheet object and target format.
2. **Caster** converts PHP values to appropriate string representations.
3. **Factory** creates appropriate Format Handler for target format.
4. **Format Handler** writes structured data to file or string.
5. File path or string content is returned to application.

Extensibility

Derafu Spreadsheet is designed to be extensible:

- **Custom Format Handlers**: Create handlers for proprietary or custom formats.
- **Custom Casters**: Implement specialized type conversion logic.
- **HTTP Integration**: Generate responses with downloadable spreadsheets.
- **Framework Integration**: Easy to integrate with popular PHP frameworks.

Directory Structure

```
src/
├── Abstract/
│   └── AbstractPhpSpreadsheetFormatHandler.php
├── Contract/
│   ├── SpreadsheetCasterInterface.php
│   ├── SpreadsheetDumperInterface.php
│   ├── FactoryInterface.php
│   ├── FormatHandlerInterface.php
│   ├── SpreadsheetLoaderInterface.php
│   ├── SheetInterface.php
│   ├── SpreadsheetInterface.php
│   └── Http/
│       └── SpreadsheetHttpResponseGeneratorInterface.php
├── Exception/
│   ├── SpreadsheetDumpException.php
│   ├── SpreadsheetFileNotFoundException.php
│   ├── SpreadsheetFormatNotSupportedException.php
│   └── SpreadsheetLoadException.php
├── Format/
│   ├── CsvLeagueHandler.php
│   ├── CsvPhpSpreadsheetHandler.php
│   ├── HtmlHandler.php
│   ├── JsonHandler.php
│   ├── OdsHandler.php
│   ├── PdfHandler.php
│   ├── XlsHandler.php
│   ├── XlsxHandler.php
│   ├── XmlHandler.php
│   └── YamlHandler.php
├── Http/
│   └── NyholmSpreadsheetHttpResponseGenerator.php
├── Caster.php
├── Dumper.php
├── Factory.php
├── Loader.php
├── Sheet.php
└── Spreadsheet.php
```

Last updated on 24/08/2026

Install

Install the library

Anonymous · 24/08/2026

Install the library

This guide will walk you through installing Derafu Spreadsheet and its dependencies.

Basic Installation

The simplest way to install Derafu Spreadsheet is through Composer:

```
composer require derafu/spreadsheet
```

This installs the core package. Out of the box, only works with JSON and XML. Depending on which file formats you want to work with, you may need to install additional dependencies.

Format-Specific Dependencies

Derafu Spreadsheet supports multiple formats through different handlers. Each format may require additional dependencies:

CSV Support

For CSV support, you can choose between two handlers:

```
# Recommended: League CSV (faster, more memory efficient).
composer require league/csv

# Alternative: PhpSpreadsheet CSV handling.
composer require phpoffice/phpspreadsheet
```

Excel and OpenDocument Support

For XLSX, XLS, and ODS support:

```
composer require phpoffice/phpspreadsheet
```

YAML Support

For YAML support:

```
composer require symfony/yaml
```

PDF Support

For PDF export:

```
# Base requirement.
composer require phpoffice/phpspreadsheet

# Choose one PDF library:
composer require mpdf/mpdf        # For MpdfWriter (recommended).
# OR
composer require dompdf/dompdf    # For DompdfWriter.
# OR
composer require tecnickcom/tcpdf # For TcpdfWriter.
```

HTTP Response Support

For PSR-7 HTTP response integration:

```
composer require nyholm/psr7
```

If you don't want to use nyholm/psr7, you can use any PSR-7 compatible library implementing by your own `SpreadsheetHttpResponseGeneratorInterface`.

Complete Installation (All Formats)

If you want to support all formats, with the default handlers, you can install all dependencies at once:

```
composer require derafu/spreadsheet league/csv phpoffice/phpspreadsheet \
    symfony/yaml mpdf/mpdf nyholm/psr7
```

Installation in Frameworks

Symfony

To use Derafu Spreadsheet in Symfony, install the package:

```
composer require derafu/spreadsheet
```

Then define services in your `services.yaml`:

```
services:
    Derafu\Spreadsheet\Contract\SpreadsheetFactoryInterface:
        class: Derafu\Spreadsheet\Factory

    Derafu\Spreadsheet\Contract\SpreadsheetCasterInterface:
        class: Derafu\Spreadsheet\Caster

    Derafu\Spreadsheet\Contract\SpreadsheetLoaderInterface:
        class: Derafu\Spreadsheet\Loader
        arguments:
            - '@Derafu\Spreadsheet\Factory'
            - '@Derafu\Spreadsheet\Caster'

    Derafu\Spreadsheet\Contract\SpreadsheetDumperInterface:
        class: Derafu\Spreadsheet\Dumper
        arguments:
            - '@Derafu\Spreadsheet\Factory'
            - '@Derafu\Spreadsheet\Caster'
```

Check the constructors of the classes to see what arguments they expect. You can configure them as you want, for example change the delimiter for CSV.

Last updated on 24/08/2026

Basic Usage

Basic Usage

Anonymous · 24/08/2026

Basic Usage

This guide covers the fundamental operations with Derafu Spreadsheet: loading, manipulating, and saving spreadsheet data.

Loading a Spreadsheet

You can load a spreadsheet from a file or from a string:

```
<?php

use Derafu\Spreadsheet\SpreadsheetLoader;

// Create a loader.
$loader = new Loader();

// From a file (format auto-detected from extension).
$spreadsheet = $loader->loadFromFile('data.xlsx');

// From a string (must specify format).
$csvString = "header1,header2\nvalue1,value2";
$spreadsheet = $loader->loadFromString($csvString, 'csv');
```

Working with Sheets

A spreadsheet contains one or more sheets, which you can access and manipulate:

```
// Get all sheet names.
$sheetNames = $spreadsheet->getSheetNames();

// Get a specific sheet.
$sheet = $spreadsheet->getSheet('Sheet1');

// Create a new sheet with data.
$spreadsheet->createSheet('NewSheet', [
    ['Header1', 'Header2', 'Header3'],
    ['Value1', 'Value2', 'Value3'],
```

```
    ['Value4', 'Value5', 'Value6']
]);

// Set active sheet.
$spreadsheet->setActiveSheet('NewSheet');

// Get active sheet.
$activeSheet = $spreadsheet->getActiveSheet();

// Check if a sheet exists.
if ($spreadsheet->hasSheet('SomeSheet')) {
    // ...
}

// Remove a sheet.
$spreadsheet->removeSheet('SomeSheet');
```

Working with Rows and Cells

Once you have a sheet, you can access and modify its data:

```
// Get all rows.
$rows = $sheet->getRows();

// Get a specific row.
$firstRow = $sheet->getRow(0);

// Add a new row.
$sheet->addRow(['New', 'Row', 'Data']);

// Get a specific cell.
$value = $sheet->getCell(0, 0); // Row 0, Column 0.
// or with named columns (for associative sheets).
$value = $sheet->getCell(0, 'column_name');

// Update a cell.
$sheet->setCell(0, 0, 'New Value');

// Get header row.
$headers = $sheet->getHeaderRow();

// Get data rows (excludes header for indexed sheets).
$dataRows = $sheet->getDataRows();
```

Data Types

Thanks to the automatic type casting, you can work with native PHP types:

```
// Numbers are already cast to int/float.
$number = $sheet->getCell(0, 0); // e.g., 123 (int).

// Dates are cast to DateTimeImmutable.
$date = $sheet->getCell(0, 1); // e.g., DateTimeImmutable object.
echo $date->format('Y-m-d'); // "2025-03-12".

// Booleans are properly typed.
$bool = $sheet->getCell(0, 2); // e.g., true (bool).

// You can set any type and it will be properly stored.
$sheet->setCell(0, 3, new DateTimeImmutable());
$sheet->setCell(0, 4, ['array', 'values']);
$sheet->setCell(0, 5, ['nested' => ['object' => 'structure']]);
```

Associative vs. Indexed Sheets

Derafu Spreadsheet supports both associative (column names as keys) and indexed (numeric keys) sheets:

```
// Check if a sheet is associative.
if ($sheet->isAssociative()) {
    // Work with associative data.
    $rowData = $sheet->getRow(0);
    echo $rowData['column_name'];
} else {
    // Work with indexed data.
    $rowData = $sheet->getRow(0);
    echo $rowData[0]; // First column.
}

    // Convert between formats.
$associativeSheet = $sheet->toAssociative(); // First row becomes header.
$indexedSheet = $sheet->toIndexed(); // Keys become first row.
```

Saving a Spreadsheet

Once you’re done modifying the data, you can save it to a file or get it as a string:

```
use Derafu\Spreadsheet\SpreadsheetDumper;
```

```
// Create a dumper.
$dumper = new Dumper();

// Save to a file (format detected from extension).
$dumper->dumpToFile($spreadsheet, 'output.xlsx');

// Convert to a different format.
$dumper->dumpToFile($spreadsheet, 'output.csv');

// Get as a string.
$jsonString = $dumper->dumpToString($spreadsheet, 'json');
```

Creating a Spreadsheet from Scratch

You can also create a spreadsheet from scratch:

```
use Derafu\Spreadsheet\Spreadsheet;

// Create an empty spreadsheet.
$spreadsheet = new Spreadsheet();

// Create a sheet with data.
$spreadsheet->createSheet('Sheet1', [
    ['Name', 'Age', 'Email'],
    ['John Doe', 30, 'john@example.com'],
    ['Jane Smith', 25, 'jane@example.com']
]);

// Create from array.
$data = [
    'Users' => [
        ['Name', 'Age', 'Email'],
        ['John Doe', 30, 'john@example.com'],
        ['Jane Smith', 25, 'jane@example.com']
    ],
    'Products' => [
        ['ID', 'Name', 'Price'],
        [1, 'Product A', 29.99],
        [2, 'Product B', 49.99]
    ]
];

$spreadsheet = Spreadsheet::fromArray($data);

// Save it.
$dumper = new Dumper();
$dumper->dumpToFile($spreadsheet, 'new_spreadsheet.xlsx');
```

Last updated on 24/08/2026

Format Handlers

Format Handlers

Anonymous · 24/08/2026

Format Handlers

Format handlers are a key component of Derafu Spreadsheet that enable the library to work with different file formats. Each format handler implements the `FormatHandlerInterface` and knows how to read and write a specific format.

Supported Formats

Derafu Spreadsheet supports the following formats in the core package:

Format	Extension	Handler Class	Dependencies
Excel XLSX	.xlsx	`XlsxHandler`	PhpSpreadsheet
Excel XLS	.xls	`XlsHandler`	PhpSpreadsheet
OpenDocument	.ods	`OdsHandler`	PhpSpreadsheet
CSV (League)	.csv	`CsvLeagueHandler`	League/CSV
CSV (PhpSpreadsheet)	.csv	`CsvPhpSpreadsheetHandler`	PhpSpreadsheet
JSON	.json	`JsonHandler`	PHP built-in
XML	.xml	`XmlHandler`	PHP built-in
YAML	.yaml	`YamlHandler`	Symfony/Yaml
HTML	.html	`HtmlHandler`	PhpSpreadsheet
PDF	.pdf	`PdfHandler`	PhpSpreadsheet + PDF renderer

How Format Handlers Work

Each format handler is responsible for:

1. Loading data from a file into a `SpreadsheetInterface` object.
2. Dumping data from a `SpreadsheetInterface` object to a file or string (memory).
3. Providing metadata like the file extension and MIME type.

The `Factory` class manages the format handlers and automatically selects the appropriate handler based on the file extension.

Using Format Handlers

Most of the time, you don't need to interact with format handlers directly. The `Loader` and `Dumper` classes handle this for you:

```
$loader = new Loader();
$spreadsheet = $loader->loadFromFile('data.xlsx');

$dumper = new Dumper();
$dumper->dumpToFile($spreadsheet, 'output.csv');
```

However, if needed, you can access format handlers directly through the `Factory`:

```
$factory = new Factory();
$xlsxHandler = $factory->createFormatHandler('filepath.xlsx');

// or with explicit format
$csvHandler = $factory->createFormatHandler('filepath.file', 'csv');
```

Format-Specific Options

Some format handlers support additional options in the constructor.

CSV Options (CsvLeagueHandler)

```
$csvHandler = new CsvLeagueHandler(
    delimiter: ',',      // Column delimiter.
    enclosure: '"',      // Field enclosure character.
    escape: '\\',        // Escape character.
    sheetName: 'Sheet'   // Default sheet name when loading CSV.
);
```

OpenDocument/Excel Options (OdsHandler, XlsHandler, XlsxHandler)

```
$xlsxHandler = new XlsxHandler(
    readDataOnly: true   // Read values only, ignore formatting.
);
```

PDF Options (PdfHandler)

```
$pdfHandler = new PdfHandler(
    writerType: 'Mpdf'   // Can be 'Mpdf', 'Dompdf', or 'Tcpdf'.
);
```

JSON Options (JsonHandler)

```
$jsonHandler = new JsonHandler(
    encodeOptions: JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE,
    decodeOptions: JSON_OBJECT_AS_ARRAY,
    depth: 512,
    sheetName: 'Sheet'
);
```

YAML Options (YamlHandler)

```
use Symfony\Component\Yaml\Yaml;
```

```
$yamlHandler = new YamlHandler(
    parseFlags: Yaml::PARSE_EXCEPTION_ON_INVALID_TYPE,
    dumpFlags: Yaml::DUMP_MULTI_LINE_LITERAL_BLOCK,
    sheetName: 'Sheet'
);
```

Custom Format Handlers

You can create your own format handlers by implementing the `FormatHandlerInterface`:

```
use Derafu\Spreadsheet\Contract\SpreadsheetFormatHandlerInterface;

class MyCustomHandler implements FormatHandlerInterface
{
    // Implement required methods.
}

// Register with the factory.
$factory = new Factory();
$factory->registerFormatHandler('custom', MyCustomHandler::class);

// Now you can use it.
$loader = new Loader($factory);
$spreadsheet = $loader->loadFromFile('filepath.custom');
```

Format Detection

The `Factory` class detects the format based on the file extension. If the file doesn't have an extension or if you want to override it, you can explicitly specify the format:

```
$loader = new Loader();
$spreadsheet = $loader->loadFromFile('filepath', 'xlsx'); // Treat as XLSX.
```

You can also check which formats are supported:

```
$factory = new Factory();
$supportedFormats = $factory->getSupportedFormats();
// ['csv', 'xlsx', 'xls', 'ods', 'xml', 'json', 'yaml', 'html', 'pdf']
```

Alternative Format Handlers

For some formats like CSV, Derafu Spreadsheet provides multiple handlers. For example:

- `CsvLeagueHandler` - Uses League/CSV (recommended for most cases).
- `CsvPhpSpreadsheetHandler` - Uses PhpSpreadsheet.

You can choose which handler to use by registering it for the format:

```
$factory = new Factory();
$factory->registerFormatHandler('csv', CsvPhpSpreadsheetHandler::class);

$loader = new Loader($factory);
// Now CSV files will use PhpSpreadsheet handler.
```

Format Handling and Type Casting

When loading a file, the process works like this:

1. Format handler reads the raw data from the file.
2. Data is converted to a `SpreadsheetInterface` object.
3. `Caster` processes all values to convert them to the appropriate PHP types.

When saving a file:

1. `Caster` converts PHP types to appropriate string representations.
2. Format handler writes the data to the file in the correct format.

Last updated on 24/08/2026

Type Casting

Type Casting

Anonymous · 24/08/2026

Type Casting

One of the most powerful features of Derafu Spreadsheet is its intelligent type casting system. This enables seamless conversion between spreadsheet file data and native PHP data types.

The Type Casting Problem

Spreadsheet files typically store everything as text, but your application needs properly typed data to work effectively. Other libraries often leave the type conversion to you, resulting in code like:

```
// Without automatic type casting.
$value = $spreadsheet->getCell(0, 0);
if (is_numeric($value)) {
    $value = (int)$value;
} elseif ($value === 'true' || $value === 'false') {
    $value = $value === 'true';
} elseif (preg_match('/^\d{4}-\d{2}-\d{2}$/', $value)) {
    $value = new \DateTime($value);
}
// ... and so on for every cell.
```

Automatic Type Casting with Derafu Spreadsheet

Under the hood, the `Caster` class automatically handles all of these conversions for you:

```
// With Derafu Spreadsheet - all values are properly typed.
$spreadsheet = $loader->loadFromFile('data.csv');
$value = $spreadsheet->getSheet('Sheet1')->getCell(0, 0); // Already properly typed!
```

Supported Type Conversions

After doing a load (reading from file/data to PHP)

File Value	PHP Type	Example
Empty string	`null`	`""` → `null`
Integer	`int`	`"123"` → `123`
Decimal	`float`	`"123.45"` → `123.45`
Boolean text	`bool`	`"true"` → `true`
Date string	`DateTimeImmutable`	`"2025-03-12"` → `DateTimeImmutable`
ISO 8601 date	`DateTimeImmutable`	`"2025-03-12T14:30:00"` → `DateTimeImmutable`
JSON array	`array`	`"[1,2,3]"` → `[1, 2, 3]`
JSON object	`array` (associative)	`"{"key":"value"}"` → `["key" => "value"]`

Before doing a dump (writing from PHP to file/data)

PHP Type	File Value	Example
`null`	Empty string	`null` → `""`
`int`/`float`	Preserved as is	`123.45` → `123.45`
`bool`	String “true”/“false”	`true` → `"true"`
`DateTimeInterface`	ISO 8601 / Date	`new DateTime()` → `"2025-03-12T14:30:00"`
`array`/`object`	JSON string	`["a", "b"]` → `"["a","b"]"`
`Stringable` objects	String representation	`$stringable` → `(string)$stringable`

Date Format Detection

The library automatically detects various date formats:

- `Y-m-d` (2025-03-12).
- `d/m/Y` (12/03/2025).
- `Y-m-d H:i:s` (2025-03-12 14:30:45).
- `d/m/Y H:i:s` (12/03/2025 14:30:45).
- `Y-m-d\TH:i:s` (2025-03-12T14:30:45).
- `Y-m-d\TH:i:sP` (2025-03-12T14:30:45+00:00).

All dates are automatically converted to UTC for consistency.

JSON Detection and Parsing

With strings that appear to be JSON (starting with `{` or `[`), the caster automatically will try to parse them into PHP arrays or objects:

```
// Cell contains: {"name":"John","age":30}
$person = $sheet->getCell(0, 0);
// Returns associative array: ["name" => "John", "age" => 30]
```

Custom Type Casting

If you need different casting behavior, you can implement your own `SpreadsheetCasterInterface`:

```
use Derafu\Spreadsheet\Contract\SpreadsheetCasterInterface;

class MyCaster implements SpreadsheetCasterInterface
{
    // Your custom implementation.
}

// Then use it with your loader/dumper.
$loader = new Loader(new Factory(), new MyCaster());
```

How Type Casting Works Internally

1. When loading a file with `Loader`, after the raw data is read, `Caster::castAfterLoad()` is called.
2. When saving a file with `Dumper`, before writing data, `Caster::castBeforeDump()` is called.
3. The type casting is performed on every cell in every sheet, ensuring consistent typing.

Performance Considerations

Type casting is performed in-memory and typically adds minimal overhead. However, for extremely large spreadsheets with millions of cells, you might consider implementing a more selective casting strategy through a custom `SpreadsheetCasterInterface` implementation.

If you don’t want to use the Caster, you can use the Factory with the Format Handlers directly. If you bypass the Loader and Dumper, no type casting will be done.

HTTP Responses

Generating HTTP Responses

Anonymous · 24/08/2026

Generating HTTP Responses

Derafu Spreadsheet makes it easy to generate downloadable spreadsheet files directly from your web application. This guide shows how to use the library to create HTTP responses for spreadsheet downloads.

PSR-7 Response Integration

Derafu Spreadsheet includes a PSR-7 compatible response generator for easy integration with frameworks that support PSR-7 standards.

Requirements

To use the HTTP response features, you need to install Nyholm's PSR-7 implementation:

```
composer require nyholm/psr7
```

If you don't want to use nyholm/psr7, you can use any PSR-7 compatible library implementing by your own `SpreadsheetHttpResponseGeneratorInterface`.

Basic Usage

```
<?php

use Derafu\Spreadsheet\Http\NyholmSpreadsheetHttpResponseGenerator;
use Derafu\Spreadsheet\Spreadsheet;

// Create or load a spreadsheet.
$spreadsheet = new Spreadsheet();
$spreadsheet->createSheet('Sheet1', [
    ['Name', 'Email', 'Age'],
    ['John Doe', 'john@example.com', 30],
    ['Jane Smith', 'jane@example.com', 25]
]);

// Create response generator.
$responseGenerator = new NyholmSpreadsheetHttpResponseGenerator();
```

```
// Generate PSR-7 response with auto-detected format (xlsx).
$response = $responseGenerator->createResponse(
    $spreadsheet,
    'users-export.xlsx'
);

// The response can now be sent by any PSR-7 compatible framework.
```

Specifying Format

You can explicitly specify the format for the response:

```
$response = $responseGenerator->createResponse(
    $spreadsheet,
    'users-export.csv',
    'csv'  // Explicitly specify format.
);
```

PSR-7 Compatible Frameworks (Slim, Mezzio, etc.)

With PSR-7 compatible frameworks, you can directly use the `NyholmSpreadsheetHttpResponseGenerator`:

```
use Derafu\Spreadsheet\Http\NyholmSpreadsheetHttpResponseGenerator;
use Derafu\Spreadsheet\SpreadsheetLoader;
use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

class ExportAction
{
    public function __invoke(ServerRequestInterface $request): ResponseInterface
    {
        // Create or load your spreadsheet
        $loader = new Loader();
        $spreadsheet = $loader->loadFromFile('data.xlsx');

        // Modify it if needed
        // ...

        // Generate response
        $responseGenerator = new NyholmSpreadsheetHttpResponseGenerator();
        return $responseGenerator->createResponse(
            $spreadsheet,
            'exported-data.xlsx'
        );
    }
}
```

Custom Response Generation

If you need to customize the response generation or use a different PSR-7 implementation, you can implement your own `SpreadsheetHttpResponseGeneratorInterface`:

```
<?php

namespace App\Spreadsheet;

use Derafu\Spreadsheet\Contract\Http\SpreadsheetHttpResponseGeneratorInterface;
use Derafu\Spreadsheet\Contract\SpreadsheetInterface;
use Psr\Http\Message\ResponseInterface;

class CustomResponseGenerator implements SpreadsheetHttpResponseGeneratorInterface
{
    public function createResponse(
        SpreadsheetInterface $spreadsheet,
        ?string $filename = null,
        ?string $format = null
    ): ResponseInterface {
        // Your custom implementation
    }
}
```

MIME Types for Different Formats

When generating HTTP responses, it’s important to use the correct MIME type. Derafu Spreadsheet handles this for you, but here’s a reference of the MIME types used for each format:

Format	MIME Type
XLSX	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
XLS	application/vnd.ms-excel
CSV	text/csv
ODS	application/vnd.oasis.opendocument.spreadsheet
JSON	application/json
XML	application/xml
YAML	application/yaml
HTML	text/html
PDF	application/pdf

Security Considerations

When generating downloadable files from user data, be sure to:

1. **Sanitize data**: Ensure user-provided data doesn't contain malicious content.
2. **Validate filenames**: Clean and validate user-provided filenames.
3. **Set appropriate headers**: Ensure you're using the correct content type and disposition.
4. **Clean up temporary files**: Remove any temporary files after sending the response.

Performance Tips

For large spreadsheets, generating the response can be resource-intensive. Consider:

1. **Queuing exports**: For large exports, process them in a background job and notify the user when ready.
2. **Streaming responses**: Some frameworks support streaming responses to reduce memory usage.
3. **Pagination**: Consider exporting data in smaller batches if possible.

Last updated on 24/08/2026

Comparison

Comparison with Alternative Libraries

Anonymous · 24/08/2026

Comparison with Alternative Libraries

When choosing a spreadsheet library for PHP, it’s important to understand the differences between available options. This guide compares Derafu Spreadsheet with some popular alternatives.

Derafu Spreadsheet vs PhpSpreadsheet

PhpSpreadsheet is the most widely used PHP spreadsheet library and is actually used internally by Derafu Spreadsheet for some format handlers.

Advantages of Derafu Spreadsheet

- **Simplified API**: The API is more streamlined and consistent across formats.
- **Customizable type casting**: Has a custom and flexible type casting system.
- **Format-agnostic code**: Write code once that works with any spreadsheet format.
- **Easier data access**: Direct methods for working with rows and cells.
- **Native handling of JSON/YAML**: Built-in support for these data exchange formats.
- **JSON inside**: Allows you to easily store and retrieve JSON data inside spreadsheets.
- **Modular approach**: Use only the format handlers you need.
- **Less overhead**: Simpler in-memory structure for data representation.

When to use PhpSpreadsheet instead

- **Advanced Excel features**: If you need complex Excel features like conditional formatting, charts, etc.
- **Cell styling**: If you need detailed control over cell appearance.
- **Formula calculation**: If you need to evaluate Excel formulas.
- **Mature ecosystem**: PhpSpreadsheet has been around longer and has more examples and community code.

Derafu Spreadsheet vs league/csv

league/csv is a focused library for working with CSV files. Internally, Derafu Spreadsheet uses it, as default, for CSV format handling.

Advantages of Derafu Spreadsheet

- **Multiple format support**: Work with many formats using the same code.
- **Automatic type casting**: league/csv returns everything as strings.
- **Higher-level abstractions**: Work with the concepts of sheets, rows, and cells.
- **Easy format conversion**: Convert between CSV and other formats seamlessly.

When to use league/csv instead

- **CSV-only projects**: If you only need to work with CSV files.
- **Memory efficiency for large files**: league/csv has specialized streaming capabilities.
- **CSV-specific features**: If you need advanced CSV features like RFC compliance options.
- **Simplicity**: If you want a more focused, single-purpose library.

Code Comparison Examples

Basic Reading Example

Derafu Spreadsheet:

```
use Derafu\Spreadsheet\SpreadsheetLoader;

$loader = new Loader();
$sheet = $loader->loadFromFile('data.xlsx')->getActiveSheet();

foreach ($sheet->getDataRows() as $row) {
    $id = $row[0]; // Already cast to proper type (int).
    $date = $row[1]; // Already a DateTimeImmutable.
    // Process...
}
```

PhpSpreadsheet:

```
use DateTimeImmutable;
use PhpOffice\PhpSpreadsheet\IOFactory;

$spreadsheet = IOFactory::load('data.xlsx');
$sheet = $spreadsheet->getActiveSheet();

foreach ($sheet->getRowIterator(2) as $row) {
    $cellIterator = $row->getCellIterator();
    $rowData = [];
    foreach ($cellIterator as $cell) {
        $rowData[] = $cell->getValue();
    }
    $id = (int)$rowData[0]; // Depending on your app and code, manual casting needed.
```

```
    $date = new DateTimeImmutable($rowData[1]); // Manual conversion.
    // Process...
}
```

league/csv:

```
use DateTimeImmutable;
use League\Csv\Reader;

$reader = Reader::createFromPath('data.csv');
$reader->setHeaderOffset(0);

foreach ($reader->getRecords() as $record) {
    $id = (int)$record['id']; // Depending on your app and code, manual casting
needed.
    $date = new DateTimeImmutable($record['date']); // Manual conversion.
    // Process...
}
```

Format Conversion Example

Derafu Spreadsheet:

```
use Derafu\Spreadsheet\SpreadsheetLoader;
use Derafu\Spreadsheet\SpreadsheetDumper;

$loader = new Loader();
$dumper = new Dumper();

// Load XLSX and save as CSV.
$spreadsheet = $loader->loadFromFile('data.xlsx');
$dumper->dumpToFile($spreadsheet, 'data.csv');

// Load CSV and save as JSON.
$spreadsheet = $loader->loadFromFile('data.csv');
$dumper->dumpToFile($spreadsheet, 'data.json');
```

With alternative libraries:

```
use League\Csv\Reader as CsvReader;
use PhpOffice\PhpSpreadsheet\IOFactory as PhpSpreadsheetIOFactory;
use PhpOffice\PhpSpreadsheet\Writer\Csv as PhpSpreadsheetCsvWriter;

// Load with PhpSpreadsheet, save as CSV.
$spreadsheet = PhpSpreadsheetIOFactory::load('data.xlsx');
$writer = new PhpSpreadsheetCsvWriter($spreadsheet);
$writer->save('data.csv');
```

```
// To convert CSV to JSON, would require:
$csv = CsvReader::createFromPath('data.csv');
$csv->setHeaderOffset(0);
$records = iterator_to_array($csv->getRecords());
file_put_contents('data.json', json_encode($records, JSON_PRETTY_PRINT));
```

Feature Comparison Table

Feature	Derafu Spreadsheet	PhpSpreadsheet	league/csv
Formats Supported	XLSX, XLS, CSV, ODS, JSON, XML, YAML, HTML, PDF	XLSX, XLS, CSV, ODS, HTML, PDF	CSV only
API Consistency	Excellent	Complex	Simple
Memory Efficiency	Can be better with streaming support	Poor for large files	Excellent
Streaming Support	Soon	Yes	Excellent
Customizable Casting	✅	✅	✅
PSR-7 Integration	✅	✅	✅
JSON/YAML Support	✅	✅	✅
Formula Support	✅	✅	✅
Cell Styling	✅	✅	✅

Conclusion

Derafu Spreadsheet is focused on simplicity and ease of use for handling the data inside spreadsheets, not the styling. It excels when you need:

1. **Unified handling** of multiple file formats.
2. **Clean, intuitive API** for spreadsheet operations.
3. **Automatic type handling** to reduce boilerplate code.
4. **Format conversion** capabilities.
5. **Modern PHP architecture** with a focus on developer experience.

Other libraries may be better suited for specific use cases:

- **PhpSpreadsheet**: For complex Excel features and formatting.
- **league/csv**: For CSV-specific operations and streaming large files.

Choose the tool that best matches your specific requirements and constraints.

Last updated on 24/08/2026