



# MultiDocConfig

*Class representing the MultiDoc Configuration*

## Summary

### Public methods

- `__construct()`
- `getIncludeDirectories()`
- `setIncludeDirectories()`
- `addIncludeDirectory()`
- `getExcludeDirectories()`
- `setExcludeDirectories()`
- `addExcludeDirectory()`
- `getOutputTo()`
- `setOutputTo()`
- `getOutputFormat()`
- `setOutputFormat()`
- `getAssetDirectory()`
- `setAssetDirectory()`
- `getHtmlDirectory()`
- `setHtmlDirectory()`
- `getMarkdownDirectory()`
- `setMarkdownDirectory()`
- `getFontsDirectory()`
- `setFontsDirectory()`

## Properties

### **includeDirectories (protected)**

```
includeDirectories: string[]
```

Directories to include

### **excludeDirectories (protected)**

```
excludeDirectories: string[]
```

Directories to exclude

### **outputTo (protected)**

```
outputTo: string
```

Directory to which the docs should be published

## **outputFormat (protected)**

```
outputFormat: string
```

The output format

## **assetDirectory (protected)**

```
assetDirectory: string
```

The directory where the assets are stored

## **htmlDirectory (protected)**

```
htmlDirectory: string
```

The directory where the HTML markup files are stored

## **markdownDirectory (protected)**

```
markdownDirectory: string
```

The directory where the markdown markup files are stored

## **fontsDirectory (protected)**

```
fontsDirectory: string
```

The directory where the font files are stored

## **Methods**

---

### **\_\_construct (public)**

```
__construct()
```

Constructor

### **getIncludeDirectories (public)**

```
getIncludeDirectories()
```

Get the directories to search in

### **setIncludeDirectories (public)**

```
setIncludeDirectories(array $directories)
```

Set a bunch of directories to search in

## Parameters

array \$directories

## addIncludeDirectory (public)

```
addIncludeDirectory(string $directory)
```

Add a directory to search in

## Parameters

string \$directory

## getExcludeDirectories (public)

```
getExcludeDirectories()
```

Get the directories to exclude from search

## setExcludeDirectories (public)

```
setExcludeDirectories(array $directories)
```

Set a bunch of directories to exclude from search

## Parameters

array \$directories

## addExcludeDirectory (public)

```
addExcludeDirectory(string $directory)
```

Add a directory to exclude from search

## Parameters

string \$directory

## getOutputTo (public)

```
getOutputTo()
```

Get the directory to which the documentation is saved

## setOutputTo (public)

```
setOutputTo(string $outputTo)
```

Set the directory to which the documentation is saved

### Parameters

string \$outputTo

### getOutputFormat (public)

```
getOutputFormat()
```

Get the format in which the documentation is rendered

### setOutputFormat (public)

```
setOutputFormat(string $outputFormat)
```

Set the directory to which the documentation is saved

### Parameters

string \$outputFormat

### getAssetDirectory (public)

```
getAssetDirectory()
```

Get the directory where the assets are stored

### setAssetDirectory (public)

```
setAssetDirectory(string $assetDirectory)
```

Set the directory where the assets are stored

### Parameters

string \$assetDirectory

### getHtmlDirectory (public)

```
getHtmlDirectory()
```

Get the directory where the html markup files are stored

### setHtmlDirectory (public)

```
setHtmlDirectory(string $htmlDirectory)
```

Set the directory where the html markup files are stored

### Parameters

string \$htmlDirectory

## **getMarkdownDirectory (public)**

```
getMarkdownDirectory()
```

Get the directory where the markdown markup files are stored

## **setMarkdownDirectory (public)**

```
setMarkdownDirectory(string $markdownDirectory)
```

Set the directory where the markdown markup files are stored

### **Parameters**

string \$markdownDirectory

## **getFontsDirectory (public)**

```
getFontsDirectory()
```

Get the directory where the font files are stored

## **setFontsDirectory (public)**

```
setFontsDirectory(string $fontsDirectory)
```

Set the directory where the font files are stored

### **Parameters**

string \$fontsDirectory

# MultiDocApplication

*Class representing the MultiDoc Console Application*

## Summary

---

### Public methods

- `__construct()`

## Methods

---

### `__construct` (public)

```
__construct(string $name = 'UNKNOWN', string $version = 'UNKNOWN')
```

Constructor

### Parameters

string \$name = 'UNKNOWN'  
string \$version = 'UNKNOWN'

# MultiDocApplicationCreateCommand

*Class representing the MultiDoc Console Application "Create"-Commands*

## Summary

### Protected methods

- `configure()`
- `execute()`

## Methods

### **configure (protected)**

```
configure()
```

### **execute (protected)**

```
execute(\Symfony\Component\Console\Input\InputInterface $input,  
\Symfony\Component\Console\Output\OutputInterface $output)
```

### Parameters

`\Symfony\Component\Console\Input\InputInterface $input`

`\Symfony\Component\Console\Output\OutputInterface $output`



# MultiDocApplicationListRenderers

*Class representing the MultiDoc Console Application "Create"-Commands*

## Summary

### Protected methods

- `configure()`
- `execute()`

## Methods

### **configure (protected)**

```
configure()
```

### **execute (protected)**

```
execute(\Symfony\Component\Console\Input\InputInterface $input,  
\Symfony\Component\Console\Output\OutputInterface $output)
```

### Parameters

`\Symfony\Component\Console\Input\InputInterface $input`

`\Symfony\Component\Console\Output\OutputInterface $output`

# MultiDocCreatorServiceInterface

*Interface for a service class which will create the documentation*

## Summary

---

### Public methods

- renderDocumentation()

## Methods

---

### renderDocumentation (public)

```
renderDocumentation()
```

Starts the creation of the documentation

# MultiDocFinderServiceInterface

*Interface for a service class which will give us all files to handle*

## Summary

### Public methods

- `getAllFiles()`
- `getAllFilesAsPhpDocLocalFiles()`

## Methods

### **getAllFiles (public)**

```
getAllFiles()
```

Get all found files

### **getAllFilesAsPhpDocLocalFiles (public)**

```
getAllFilesAsPhpDocLocalFiles()
```

Get all files as a PHPdoc LocalFile

# MultiDocMarkupServiceInterface

*Interface for a service class which renders the markup*

## Summary

### Public methods

- initializeService()
- getMarkupTemplateDirectory()
- getMarkupOutput()
- addToMarkupOutput()
- render()
- renderAndAddToOutput()
- writeHeader()
- writeSummary()
- writeConstants()
- writeProperties()
- writeMethods()
- createFromClass()
- createFromInterface()
- createFromTrait()

## Methods

### initializeService (public)

```
initializeService()
```

Initialize (e.g. the internal markup Content Container)

### getMarkupTemplateDirectory (public)

```
getMarkupTemplateDirectory()
```

Get the directory where all the markup template files are located

### getMarkupOutput (public)

```
getMarkupOutput()
```

Return the created markup

### addToMarkupOutput (public)

```
addToMarkupOutput(string $add)
```

Add data to markup output

## Parameters

string \$add

## render (public)

```
render(string $name, array $data = array())
```

Render a markup

## Parameters

string \$name

array \$data = array()

## renderAndAddToOutput (public)

```
renderAndAddToOutput(string $name, array $data = array())
```

Render a markup and add the rendered output to internal markup storage

## Parameters

string \$name

array \$data = array()

## writeHeader (public)

```
writeHeader(string $name, string $summary, string $description)
```

Write Header

## Parameters

string \$name

string \$summary

string \$description

## writeSummary (public)

```
writeSummary(array $constants, array $properties, array $methods)
```

Write a summary

## Parameters

array \$constants

array \$properties

array \$methods

## writeConstants (public)

```
writeConstants(array $constants)
```

Write constants

### Parameters

array \$constants

### writeProperties (public)

```
writeProperties(array $properties)
```

Write properties

### Parameters

array \$properties

### writeMethods (public)

```
writeMethods(array $methods)
```

Write methods

### Parameters

array \$methods

### createFromClass (public)

```
createFromClass(\phpDocumentor\Reflection\Php\Class_ $class)
```

Generate class description

### Parameters

\phpDocumentor\Reflection\Php\Class\_ \$class

### createFromInterface (public)

```
createFromInterface(\phpDocumentor\Reflection\Php\Interface_ $interface)
```

Generate Interface description

### Parameters

\phpDocumentor\Reflection\Php\Interface\_ \$interface

### createFromTrait (public)

```
createFromTrait(\phpDocumentor\Reflection\Php\Trait_ $trait)
```

Generate Trait description

## Parameters

`\phpDocumentor\Reflection\Php\Trait_ $trait`

# MultiDocRenderServiceInterface

*Interface for a service class which will render the documentation*

## Summary

### Public methods

- `setLocalFiles()`
- `renderDocumentation()`

## Methods

### setLocalFiles (public)

```
setLocalFiles(array $files)
```

Set the files which are to handle

### Parameters

array \$files

### renderDocumentation (public)

```
renderDocumentation()
```

Render the documentation from files



# MultiDocRendererInterface

*Interface for a service class which renders the output documents*

## Summary

---

### Public methods

- setReflectedFiles()
- render()

## Methods

---

### setReflectedFiles (public)

```
setReflectedFiles(array $files)
```

Set the file to render

#### Parameters

array \$files

### render (public)

```
render()
```

Render the file

# MultiDocTwigServiceInterface

*Interface for service class which will render the twig templates*

## Summary

### Public methods

- addTemplateDirectory()
- render()

## Methods

### addTemplateDirectory (public)

```
addTemplateDirectory(string $directory)
```

Add a directory where to find the needed templates

#### Parameters

string \$directory

### render (public)

```
render(string $name, array $data)
```

Render a twig remplate

#### Parameters

string \$name

array \$data

# MultiDocRendererMultipleMarkdown

*service class which renders the output documents as an single markdown document*

## Summary

### Public methods

- `__construct()`
- `setReflectedFiles()`
- `render()`

### Private methods

- `renderSingleMarkdown()`
- `renderClass()`
- `renderInterface()`
- `renderTrait()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### markupService (protected)

```
markupService: \horstoeko\multidocumentor\Interfaces\MultiDocMarkupServiceInterface
```

The internal markup service

### reflectedFiles (protected)

```
reflectedFiles
```

Files to handle

## Methods

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

## Parameters

\horstoeko\multidocumentor\Config\MultiDocConfig \$config

### setReflectedFiles (public)

```
setReflectedFiles(array $files)
```

## Parameters

array \$files

### render (public)

```
render()
```

### renderSingleMarkdown (private)

```
renderSingleMarkDown(string $destinationFilename)
```

Render a single markdown file

## Parameters

string \$destinationFilename

### renderClass (private)

```
renderClass(mixed $class)
```

Render a class markdown file

## Parameters

mixed \$class

### renderInterface (private)

```
renderInterface(mixed $interface)
```

Render a interface markdown file

## Parameters

mixed \$interface

### renderTrait (private)

```
renderTrait(mixed $interface)
```

Render a interface markdown file

## Parameters

mixed \$interface

# MultiDocRendererSingleMarkdown

*service class which renders the output documents as an single markdown document*

## Summary

### Public methods

- `__construct()`
- `setReflectedFiles()`
- `render()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### markupService (protected)

```
markupService: \horstoeko\multidocumentor\Interfaces\MultiDocMarkupServiceInterface
```

The internal markup service

### reflectedFiles (protected)

```
reflectedFiles
```

Files to handle

## Methods

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

### setReflectedFiles (public)

```
setReflectedFiles(array $files)
```

## Parameters

array \$files

## render (public)

```
render()
```

# MultiDocRendererMultipleMarkdown

*service class which renders the output documents as an single markdown document*

## Summary

### Public methods

- `__construct()`
- `setReflectedFiles()`
- `render()`

### Private methods

- `renderSingleMarkdown()`
- `renderClass()`
- `renderInterface()`
- `renderTrait()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### markupService (protected)

```
markupService: \horstoeko\multidocumentor\Interfaces\MultiDocMarkupServiceInterface
```

### htmlConverter (protected)

```
htmlConverter: \League\HTMLToMarkdown\HtmlConverter
```

### reflectedFiles (protected)

```
reflectedFiles
```

Files to handle

## Methods

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor



## Parameters

\horstoeko\multidocumentor\Config\MultiDocConfig \$config

### setReflectedFiles (public)

```
setReflectedFiles(array $files)
```

## Parameters

array \$files

### render (public)

```
render()
```

### renderSingleMarkdown (private)

```
renderSingleMarkdown(string $destinationFilename)
```

Render a single markdown file

## Parameters

string \$destinationFilename

### renderClass (private)

```
renderClass(mixed $class)
```

Render a class markdown file

## Parameters

mixed \$class

### renderInterface (private)

```
renderInterface(mixed $interface)
```

Render a interface markdown file

## Parameters

mixed \$interface

### renderTrait (private)

```
renderTrait(mixed $interface)
```

Render a interface markdown file

## Parameters

mixed \$interface

# MultiDocRendererSingleMarkdown

*service class which renders the output documents as an single markdown document*

## Summary

### Public methods

- `__construct()`
- `setReflectedFiles()`
- `render()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### markupService (protected)

```
markupService: \horstoeko\multidocumentor\Interfaces\MultiDocMarkupServiceInterface
```

### htmlConverter (protected)

```
htmlConverter: \League\HTMLToMarkdown\HtmlConverter
```

### reflectedFiles (protected)

```
reflectedFiles
```

Files to handle

## Methods

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

## **setReflectedFiles (public)**

```
setReflectedFiles(array $files)
```

### **Parameters**

array \$files

## **render (public)**

```
render()
```

# MultiDocRendererFactory

*class which is a factory for a renderer*

## Summary

---

### Public methods

- createRenderer()

## Methods

---

### createRenderer (public)

```
createRenderer(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Create a renderer by format identifiziert

### Parameters

\horstoeko\multidocumentor\Config\MultiDocConfig \$config

# MultiDocRendererFactoryDefinition

*class which is a factory definition for a renderer*

## Summary

### Public methods

- `__construct()`
- `make()`
- `getName()`
- `getDescription()`
- `getClassName()`

## Properties

### name (protected)

```
name: string
```

A short name for a renderer

### description (protected)

```
description: string
```

A longer introduction for a renderer

### classname (protected)

```
classname: string
```

The classname of the renderer to use

## Methods

### \_\_construct (public)

```
__construct(string $name, string $description, string $classname)
```

Constructor

### Parameters

string \$name  
string \$description  
string \$classname

## **make (public)**

```
make(string $name, string $description, string $classname)
```

Create a new renderer definition

### **Parameters**

string \$name  
string \$description  
string \$classname

## **getName (public)**

```
getName()
```

Returns the name of the renderer

## **getDescription (public)**

```
getDescription()
```

Returns the description of the renderer

## **getClassname (public)**

```
getClassname()
```

Returns the class name of the renderer

# MultiDocRendererFactoryDefinitionList

*class which is a list of factory definitions for a renderer*

## Summary

### Public methods

- `__construct()`
- `findByIndex()`
- `findByName()`
- `existsByIndex()`
- `existsByName()`
- `getAllRegisteredRenderers()`

### Private methods

- `initDefaultRenderers()`
- `initCustomRenderers()`
- `addRendererDefinition()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### rendererDefinitions (protected)

```
rendererDefinitions:  
\horstoeko\multidocumentor\Renderer\MultiDocRendererFactoryDefinition[]
```

A List of defined renderers

## Methods

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`



## **findByIndex (public)**

```
findByIndex(int $index, bool $raiseExceptionIfNotFound = true)
```

Find a renderer by it's registerd \$index

### **Parameters**

int \$index

bool \$raiseExceptionIfNotFound = true

## **findByName (public)**

```
findByName(string $name, bool $raiseExceptionIfNotFound = true)
```

Find renderer by it's \$name

### **Parameters**

string \$name

bool \$raiseExceptionIfNotFound = true

## **existsByIndex (public)**

```
existsByIndex(int $index)
```

Check if a renderer has been registered on \$index

### **Parameters**

int \$index

## **existsByName (public)**

```
existsByName(string $name)
```

Check if a renderer has been registered with \$name

### **Parameters**

string \$name

## **getAllRegisteredRenderers (public)**

```
getAllRegisteredRenderers()
```

Returns a list of all registered renderers

## **initDefaultRenderers (private)**

```
initDefaultRenderers()
```

Initialize a list of default renderers

## **initCustomRenderers (private)**

```
initCustomRenderers()
```

Initialize custom renderers from config

## **addRendererDefinition (private)**

```
addRendererDefinition(\horstoeko\multidocumentor\Renderer\MultiDocRendererFactoryDef  
inition $rendererDefinition)
```

Add a renderer definition to list

### **Parameters**

\horstoeko\multidocumentor\Renderer\MultiDocRendererFactoryDefinition \$rendererDefinition

# MultiDocPdfFile

*Class which creates a pdf file*

## Summary

---

### Public methods

- `__construct()`

## Properties

---

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

## Methods

---

### `__construct` (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

# MultiDocRendererMultiplePdf

*service class which renders the output documents as an single PDF document*

## Summary

### Public methods

- `__construct()`
- `setReflectedFiles()`
- `render()`

### Private methods

- `renderSingleMarkDown()`
- `renderClass()`
- `renderInterface()`
- `renderTrait()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### markupService (protected)

```
markupService: \horstoeko\multidocumentor\Interfaces\MultiDocMarkupServiceInterface
```

### reflectedFiles (protected)

```
reflectedFiles
```

Files to handle

## Methods

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

## setReflectedFiles (public)

```
setReflectedFiles(array $files)
```

### Parameters

array \$files

## render (public)

```
render()
```

## renderSingleMarkdown (private)

```
renderSingleMarkdown(string $destinationFilename)
```

Render a single PDF file

### Parameters

string \$destinationFilename

## renderClass (private)

```
renderClass(mixed $class)
```

Render a class pdf file

### Parameters

mixed \$class

## renderInterface (private)

```
renderInterface(mixed $interface)
```

Render a interface pdf file

### Parameters

mixed \$interface

## renderTrait (private)

```
renderTrait(mixed $interface)
```

Render a interface pdf file

### Parameters

mixed \$interface

# MultiDocRendererSinglePdf

*service class which renders the output documents as an single PDF document*

## Summary

### Public methods

- `__construct()`
- `setReflectedFiles()`
- `render()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### markupService (protected)

```
markupService: \horstoeko\multidocumentor\Interfaces\MultiDocMarkupServiceInterface
```

### reflectedFiles (protected)

```
reflectedFiles
```

Files to handle

## Methods

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

### setReflectedFiles (public)

```
setReflectedFiles(array $files)
```

### Parameters

`array $files`

## render (public)

```
render()
```

# MultiDocAbstractMarkupService

*Service class which renders the markup*

## Summary

---

### Public methods

- `__construct()`
- `initializeService()`
- `getMarkupTemplateDirectory()`
- `getMarkupOutput()`
- `addToMarkupOutput()`
- `render()`
- `renderAndAddToOutput()`
- `writeHeader()`
- `writeSummary()`
- `writeConstants()`
- `writeProperties()`
- `writeMethods()`
- `createFromClass()`
- `createFromInterface()`
- `createFromTrait()`

## Properties

---

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### twigService (private)

```
twigService: \horstoeko\multidocumentor\Interfaces\MultiDocTwigServiceInterface
```

The HTML Engine

### markup (private)

```
markup: string
```

The internal markup container

## Methods

---



## **\_\_construct (public)**

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructur

### **Parameters**

\horstoeko\multidocumentor\Config\MultiDocConfig \$config

## **initializeService (public)**

```
initializeService()
```

## **getMarkupTemplateDirectory (public)**

```
getMarkupTemplateDirectory()
```

## **getMarkupOutput (public)**

```
getMarkupOutput()
```

## **addToMarkupOutput (public)**

```
addToMarkupOutput(string $add)
```

### **Parameters**

string \$add

## **render (public)**

```
render(string $name, array $data = array())
```

### **Parameters**

string \$name

array \$data = array()

## **renderAndAddToOutput (public)**

```
renderAndAddToOutput(string $name, array $data = array())
```

### **Parameters**

string \$name

array \$data = array()

## **writeHeader (public)**

```
writeHeader(string $name, string $summary, string $description)
```

## Parameters

string \$name  
string \$summary  
string \$description

## writeSummary (public)

```
writeSummary(array $constants, array $properties, array $methods)
```

## Parameters

array \$constants  
array \$properties  
array \$methods

## writeConstants (public)

```
writeConstants(array $constants)
```

## Parameters

array \$constants

## writeProperties (public)

```
writeProperties(array $properties)
```

## Parameters

array \$properties

## writeMethods (public)

```
writeMethods(array $methods)
```

## Parameters

array \$methods

## createFromClass (public)

```
createFromClass(\phpDocumentor\Reflection\Php\Class_ $class)
```

## Parameters

\phpDocumentor\Reflection\Php\Class\_ \$class

## createFromInterface (public)

```
createFromInterface(\phpDocumentor\Reflection\Php\Interface_ $interface)
```

## Parameters

\phpDocumentor\Reflection\Php\Interface\_ \$interface

## createFromTrait (public)

```
createFromTrait(\phpDocumentor\Reflection\Php\Trait_ $trait)
```

## Parameters

\phpDocumentor\Reflection\Php\Trait\_ \$trait

# MultiDocCreatorService

*Service class which will create the documentation*

## Summary

### Public methods

- `__construct()`
- `renderDocumentation()`

## Properties

### **config (protected)**

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### **finderService (protected)**

```
finderService: \horstoeko\multidocumentor\Interfaces\MultiDocFinderServiceInterface
```

Finder Service

### **renderService (protected)**

```
renderService: \horstoeko\multidocumentor\Interfaces\MultiDocRenderServiceInterface
```

Render Service

## Methods

### **\_\_construct (public)**

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

### **renderDocumentation (public)**

```
renderDocumentation()
```

# MultiDocFinderService

*Service class which will give us all files to handle*

## Summary

### Public methods

- `__construct()`
- `getAllFiles()`
- `getAllFilesAsPhpDocLocalFiles()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### finder (protected)

```
finder: \Symfony\Component\Finder\Finder
```

Internal finder component

## Methods

### `__construct` (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

### `getAllFiles` (public)

```
getAllFiles()
```

### `getAllFilesAsPhpDocLocalFiles` (public)

```
getAllFilesAsPhpDocLocalFiles()
```

# MultiDocMarkupHtmlService

*Service class which renders the markup in HTML format*

## Summary

### Public methods

- getMarkupTemplateDirectory()
- writeHeader()
- writeSummary()
- writeConstants()
- writeProperties()
- writeMethods()
- createFromClass()
- createFromInterface()
- createFromTrait()

## Methods

### getMarkupTemplateDirectory (public)

```
getMarkupTemplateDirectory()
```

### writeHeader (public)

```
writeHeader(string $name, string $summary, string $description)
```

#### Parameters

string \$name  
string \$summary  
string \$description

### writeSummary (public)

```
writeSummary(array $constants, array $properties, array $methods)
```

#### Parameters

array \$constants  
array \$properties  
array \$methods

### writeConstants (public)

```
writeConstants(array $constants)
```

## Parameters

array \$constants

## writeProperties (public)

```
writeProperties(array $properties)
```

## Parameters

array \$properties

## writeMethods (public)

```
writeMethods(array $methods)
```

## Parameters

array \$methods

## createFromClass (public)

```
createFromClass(\phpDocumentor\Reflection\Php\Class_ $class)
```

## Parameters

\phpDocumentor\Reflection\Php\Class\_ \$class

## createFromInterface (public)

```
createFromInterface(\phpDocumentor\Reflection\Php\Interface_ $interface)
```

## Parameters

\phpDocumentor\Reflection\Php\Interface\_ \$interface

## createFromTrait (public)

```
createFromTrait(\phpDocumentor\Reflection\Php\Trait_ $trait)
```

## Parameters

\phpDocumentor\Reflection\Php\Trait\_ \$trait

# MultiDocMarkupMarkdownService

*Service class which renders the markup in markdown format*

## Summary

### Public methods

- getMarkupTemplateDirectory()
- writeHeader()
- writeSummary()
- writeConstants()
- writeProperties()
- writeMethods()
- createFromClass()
- createFromInterface()
- createFromTrait()

## Methods

### getMarkupTemplateDirectory (public)

```
getMarkupTemplateDirectory()
```

### writeHeader (public)

```
writeHeader(string $name, string $summary, string $description)
```

#### Parameters

string \$name  
string \$summary  
string \$description

### writeSummary (public)

```
writeSummary(array $constants, array $properties, array $methods)
```

#### Parameters

array \$constants  
array \$properties  
array \$methods

### writeConstants (public)

```
writeConstants(array $constants)
```



## Parameters

array \$constants

## writeProperties (public)

```
writeProperties(array $properties)
```

## Parameters

array \$properties

## writeMethods (public)

```
writeMethods(array $methods)
```

## Parameters

array \$methods

## createFromClass (public)

```
createFromClass(\phpDocumentor\Reflection\Php\Class_ $class)
```

## Parameters

\phpDocumentor\Reflection\Php\Class\_ \$class

## createFromInterface (public)

```
createFromInterface(\phpDocumentor\Reflection\Php\Interface_ $interface)
```

## Parameters

\phpDocumentor\Reflection\Php\Interface\_ \$interface

## createFromTrait (public)

```
createFromTrait(\phpDocumentor\Reflection\Php\Trait_ $trait)
```

## Parameters

\phpDocumentor\Reflection\Php\Trait\_ \$trait

# MultiDocRenderService

*Service class which will render the documentation*

## Summary

### Public methods

- `__construct()`
- `setLocalFiles()`
- `renderDocumentation()`

## Properties

### config (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### localFiles (protected)

```
localFiles
```

Files to handle

## Methods

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

### setLocalFiles (public)

```
setLocalFiles(array $files)
```

### Parameters

`array $files`

### renderDocumentation (public)

```
renderDocumentation()
```

# MultiDocTwigService

*Service class which will render the twig templates*

## Summary

### Public methods

- `__construct()`
- `addTemplateDirectory()`
- `render()`

## Properties

### `config` (protected)

```
config: \horstoeko\multidocumentor\Config\MultiDocConfig
```

Configuration

### `twigEngine` (protected)

```
twigEngine: \horstoeko\multidocumentor\Twig\MultiDocTwigEngine
```

The Twig engine

## Methods

### `__construct` (public)

```
__construct(\horstoeko\multidocumentor\Config\MultiDocConfig $config)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Config\MultiDocConfig $config`

### `addTemplateDirectory` (public)

```
addTemplateDirectory(string $directory)
```

### Parameters

`string $directory`

### `render` (public)

```
render(string $name, array $data)
```

## Parameters

string \$name

array \$data

# MultiDocTwigEngine

*Class which wraps base Twig Engine*

## Summary

### Public methods

- `__construct()`
- `addTemplateDirectory()`
- `render()`

## Properties

### twigLoader (protected)

```
twigLoader: \Twig\Loader\FilesystemLoader
```

Twig template loader

### twigEnvironment (protected)

```
twigEnvironment: \Twig\Environment
```

Twig Environment

## Methods

### \_\_construct (public)

```
__construct()
```

Constructor

### addTemplateDirectory (public)

```
addTemplateDirectory(string $directory)
```

Add a folder where templates are stored

### Parameters

string \$directory

### render (public)

```
render(string $name, array $data)
```

Render a template

### **Parameters**

string \$name

array \$data

# MultiDocTwigExtension

*Class for multiDoc twig extensions*

## Summary

### Public methods

- `getFilters()`
- `removeInvisibleCharacters()`
- `parsedown()`

## Methods

### **getFilters (public)**

```
getFilters()
```

Returns a list of filters to add to the existing list.

### **removeInvisibleCharacters (public)**

```
removeInvisibleCharacters(mixed $string)
```

Removes invisble Characters

#### **Parameters**

mixed \$string

### **parsedown (public)**

```
parsedown(mixed $string)
```

Parse markdown to HTML

#### **Parameters**

mixed \$string