



# 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'

# MultiDocApplicationAbstractCommand

*Class representing the MultiDoc base class for MultiDoc's console command*

## Summary

---

### Public methods

- `__construct()`

### Protected methods

- `getValidationOptions()`
- `validateOptions()`
- `validateOptionsWithMessage()`
- `validatedOption()`
- `execute()`
- `handle()`

## Properties

---

### validator (protected)

```
validator: \Rakit\Validation\Validator
```

The internal validation class

### validation (protected)

```
validation: \Rakit\Validation\Validation
```

The internal validation class containing the validation result

## Methods

---

### \_\_construct (public)

```
__construct(string $name = null)
```

### Parameters

string \$name = null

## getValidationOptions (protected)

```
getValidationOptions()
```

Return a list of all validation options

## validateOptions (protected)

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

Perform option validation

### Parameters

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

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

## validateOptionsWithMessage (protected)

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

Perform option validation. If validation fails all error messages will be shown

### Parameters

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

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

## validatedOption (protected)

```
validatedOption(string $name)
```

Return the validated value of an option with name \$name

### Parameters

string \$name

## 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

## handle (protected)

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

Contains the business logic of the command

### Parameters

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

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

# MultiDocApplicationCreateCommand

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

## Summary

---

### Protected methods

- `configure()`
- `getValidationOptions()`
- `handle()`

## Methods

---

### `configure (protected)`

```
configure()
```

### `getValidationOptions (protected)`

```
getValidationOptions()
```

### `handle (protected)`

```
handle(\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`

# MultiDocApplicationCreateMultipleCommand

*Class representing the MultiDoc Console Application "CreateMultipleFormats"-Commands.*

You can specify multiple output formats

## Summary

---

### Protected methods

- `configure()`
- `getValidationOptions()`
- `handle()`
- `handleFormat()`

## Methods

---

### configure (protected)

```
configure()
```

### getValidationOptions (protected)

```
getValidationOptions()
```

### handle (protected)

```
handle(\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

### handleFormat (protected)

```
handleFormat(\Symfony\Component\Console\Input\InputInterface $input,  
\Symfony\Component\Console\Output\OutputInterface $output, string $format)
```

Handle an output in a defined format

#### Parameters

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

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

string \$format



# MultiDocApplicationListRenderers

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

## Summary

---

### Protected methods

- `configure()`
- `handle()`

## Methods

---

### **configure (protected)**

```
configure()
```

### **handle (protected)**

```
handle(\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`

# MultiDocValidationArrayOption

*Class representing the validation of a console array option*

## Summary

---

### Public methods

- `check()`
- `modifyValue()`

## Properties

---

### message (protected)

message: string

Message

### fillableParams (protected)

fillableParams: array

Parameters

## Methods

---

### check (public)

```
check(mixed $value)
```

Check the \$value is an array

#### Parameters

mixed \$value

### modifyValue (public)

```
modifyValue(mixed $value)
```

{@inheritDoc}

#### Parameters

mixed \$value

# MultiDocValidationStringOption

*Class representing the validation of a console string option*

## Summary

---

### Public methods

- `check()`
- `modifyValue()`

### Protected methods

- `isEmptyValue()`

## Properties

---

### message (protected)

message: string

Message

### fillableParams (protected)

fillableParams: array

Parameters

## Methods

---

### check (public)

```
check(mixed $value)
```

Check the \$value is an array

#### Parameters

mixed \$value

### modifyValue (public)

```
modifyValue(mixed $value)
```

{@inheritDoc}

#### Parameters

mixed \$value

### isEmptyValue (protected)

```
isEmptyValue(mixed $value)
```

Check \$value is empty value

#### Parameters

mixed \$value

# MultiDocContainer

*Class representing the MultiDoc Container*

## Summary

---

### Public methods

- `__construct()`
- `__get()`
- `__set()`
- `__call()`

## Properties

---

### container (public)

container: array

The container that holds all settings

# Methods

---

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

```
__construct()
```

Constructor

## **\_\_get (public)**

```
__get(mixed $name)
```

Magic getter

### **Parameters**

mixed \$name

## **\_\_set (public)**

```
__set(mixed $name, mixed $value)
```

Magic setter

### **Parameters**

mixed \$name

mixed \$value

## **\_\_call (public)**

```
__call(string $name, array $arguments)
```

Magic call

### **Parameters**

string \$name

array \$arguments

# MultiDocCreatorServiceInterface

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

## Summary

---

### Public methods

- `render()`

## Methods

---

### **render (public)**

```
render ( )
```

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()
- beforeGetMarkupOutput()
- 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

### beforeGetMarkupOutput (public)

```
beforeGetMarkupOutput()
```

Trigger which called before the getMarkupOutput method is run

## 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

- `setIncludedFiles()`
- `render()`

## Methods

---

### setIncludedFiles (public)

```
setIncludedFiles(array $files)
```

Set the files which are to handle

### Parameters

array \$files

### render (public)

```
render()
```

Render the documentation from files

# MultiDocRendererInterface

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

## Summary

### Public methods

- `getShortName()`
- `getDescription()`
- `setReflectedFiles()`
- `render()`

## Methods

### getShortName (public)

```
getShortName()
```

Get the shortname for this renderer

### getDescription (public)

```
getDescription()
```

Get a description for this renderer

### 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



# MultiDocRendererMultipleHtml

*service class which renders the output as multiple HTML documents*

## Summary

---

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `render()`
- `getCss()`

### Private methods

- `renderSingleHtml()`
- `renderClass()`
- `renderInterface()`
- `renderTrait()`

## Properties

---

### markupService (protected)

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

The internal markup service

## Methods

---

### `__construct` (public)

`__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)`

Constructor

### Parameters

`\horstoeko\multidocumentor\Container\MultiDocContainer $container`

## getShortName (public)

```
getShortName()
```

## getDescription (public)

```
getDescription()
```

## render (public)

```
render()
```

## getCss (public)

```
getCss()
```

Return CSS content

## renderSingleHtml (private)

```
renderSingleHtml(string $destinationFilename)
```

Render a single markdown file

### Parameters

string \$destinationFilename

## renderClass (private)

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

Render a class markdown file

### Parameters

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

## renderInterface (private)

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

Render a interface markdown file

### Parameters

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

## renderTrait (private)

```
renderTrait(\phpDocumentor\Reflection\Php\Trait_ $interface)
```

Render a interface markdown file

### Parameters

\phpDocumentor\Reflection\Php\Trait\_ \$interface

# MultiDocRendererSingleHtml

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

## Summary

---

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `render()`

## Properties

---

### markupService (protected)

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

The internal markup service

## Methods

---

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

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### **Parameters**

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

### **getShortName (public)**

```
getShortName()
```

### **getDescription (public)**

```
getDescription()
```

### **render (public)**

```
render()
```

# MultiDocRendererMultipleMarkDown

*service class which renders the output documents as multiple HTML documents*

## Summary

---

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `render()`

### Private methods

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

## Properties

---

### markupService (protected)

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

The internal markup service

## Methods

---

### `__construct` (public)

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Container\MultiDocContainer $container`

### `getShortName` (public)

```
getShortName()
```

## getDescription (public)

```
getDescription()
```

## render (public)

```
render()
```

## renderSingleMarkdown (private)

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

Render a single markdown file

### Parameters

string \$destinationFilename

## renderClass (private)

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

Render a class markdown file

### Parameters

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

## renderInterface (private)

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

Render a interface markdown file

### Parameters

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

## renderTrait (private)

```
renderTrait(\phpDocumentor\Reflection\Php\Trait_ $interface)
```

Render a interface markdown file

### Parameters

\phpDocumentor\Reflection\Php\Trait\_ \$interface

# MultiDocRendererSingleMarkdown

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

## Summary

---

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `render()`

## Properties

---

### markupService (protected)

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

The internal markup service



## Methods

---

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

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### **Parameters**

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

### **getShortName (public)**

```
getShortName()
```

### **getDescription (public)**

```
getDescription()
```

### **render (public)**

```
render()
```

# MultiDocRendererMultipleMarkdown

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

## Summary

---

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `render()`

### Private methods

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

## Properties

---

### markupService (protected)

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

### htmlConverter (protected)

`htmlConverter: \League\HTMLToMarkdown\HtmlConverter`

## Methods

---

### `__construct` (public)

`__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)`

Constructor

### Parameters

`\horstoeko\multidocumentor\Container\MultiDocContainer $container`

## getShortName (public)

```
getShortName()
```

## getDescription (public)

```
getDescription()
```

## render (public)

```
render()
```

## renderSingleMarkdown (private)

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

Render a single markdown file

### Parameters

string \$destinationFilename

## renderClass (private)

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

Render a class markdown file

### Parameters

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

## renderInterface (private)

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

Render a interface markdown file

### Parameters

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

## renderTrait (private)

```
renderTrait(\phpDocumentor\Reflection\Php\Trait_ $interface)
```

Render a interface markdown file

## Parameters

\phpDocumentor\Reflection\Php\Trait\_ \$interface

# MultiDocRendererSingleMarkdown

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

## Summary

---

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `render()`

## Properties

---

### markupService (protected)

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

### htmlConverter (protected)

htmlConverter: `\League\HTMLToMarkdown\HtmlConverter`

## Methods

---

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

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### **Parameters**

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

### **getShortName (public)**

```
getShortName()
```

### **getDescription (public)**

```
getDescription()
```

### **render (public)**

```
render()
```

# MultiDocAbstractRenderer

*Basic class which renders the document(s)*

## Summary

---

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `setReflectedFiles()`
- `render()`

## Properties

---

### container (protected)

```
container: \horstoeko\multidocumentor\Container\MultiDocContainer
```

Container (Settings)

### reflectedFiles (protected)

```
reflectedFiles
```

Files to handle

# Methods

---

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

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### **Parameters**

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

## **getShortName (public)**

```
getShortName()
```

Get the shortname for this renderer

## **getDescription (public)**

```
getDescription()
```

Get a description for this renderer

## **setReflectedFiles (public)**

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

### **Parameters**

array \$files

## **render (public)**

```
render()
```



# MultiDocRendererClassList

*Class which is a list of available renderer classes*

## Summary

---

### Public methods

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

### Private methods

- `initDefaultRenderers()`
- `initCustomRenderers()`
- `addRendererClasses()`
- `addRendererClass()`

## Properties

---

### container (protected)

```
container: \horstoeko\multidocumentor\Container\MultiDocContainer
```

Container (Settings)

### rendererClasses (protected)

```
rendererClasses: string[]
```

A List of defined renderer classes

## Methods

---

### `__construct` (public)

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### Parameters

\\horstoeko\\multidocumentor\\Container\\MultiDocContainer \$container

### 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 container

## **addRendererClasses (private)**

```
addRendererClasses(array $classNames)
```

Add multiple renderer classes

### **Parameters**

array \$classNames

## **addRendererClass (private)**

```
addRendererClass(string $className)
```

Add a renderer class

### **Parameters**

string \$className

# MultiDocRendererFactory

*class which is a factory for a renderer*

## Summary

---

### Public methods

- createRenderer()

## Methods

---

### createRenderer (public)

```
createRenderer(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Create a renderer by format identified

### Parameters

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

# MultiDocPdfFile

*Class which creates a pdf file*

## Summary

---

### Public methods

- `__construct()`

## Properties

---

### container (protected)

```
container: \horstoeko\multidocumentor\Container\MultiDocContainer
```

Container (Settings)

## Methods

---

### `__construct` (public)

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Container\MultiDocContainer $container`

# MultiDocRendererMultiplePdf

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

## Summary

---

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `render()`

### Private methods

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

## Properties

---

### markupService (protected)

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

## Methods

---

### `__construct` (public)

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### Parameters

`\horstoeko\multidocumentor\Container\MultiDocContainer $container`

### `getShortName` (public)

```
getShortName()
```

## getDescription (public)

```
getDescription()
```

## render (public)

```
render()
```

## renderSingleMarkDown (private)

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

Render a single PDF file

### Parameters

string \$destinationFilename

## renderClass (private)

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

Render a class pdf file

### Parameters

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

## renderInterface (private)

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

Render a interface pdf file

### Parameters

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

## renderTrait (private)

```
renderTrait(\phpDocumentor\Reflection\Php\Trait_ $interface)
```

Render a interface pdf file

### Parameters

\phpDocumentor\Reflection\Php\Trait\_ \$interface

# MultiDocRendererSinglePdf

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

## Summary

### Public methods

- `__construct()`
- `getShortName()`
- `getDescription()`
- `render()`

## Properties

### markupService (protected)

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

## Methods

### `__construct` (public)

`__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)`

Constructor

### Parameters

`\horstoeko\multidocumentor\Container\MultiDocContainer $container`

### `getShortName` (public)

`getShortName()`

### `getDescription` (public)

`getDescription()`

### `render` (public)

`render()`



# MultiDocAbstractMarkupService

*Basic Service class which renders the markup*

## Summary

---

### Public methods

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

## Properties

---

### container (protected)

```
container: \horstoeko\multidocumentor\Container\MultiDocContainer
```

Container (Settings)

### twigService (private)

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

The HTML Engine

### markup (protected)

```
markup: string
```

The internal markup container

## Methods

---

### \_\_construct (public)

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructur

### Parameters

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

### initializeService (public)

```
initializeService()
```

### getMarkupTemplateDirectory (public)

```
getMarkupTemplateDirectory()
```

### beforeGetMarkupOutput (public)

```
beforeGetMarkupOutput()
```

## 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()`
- `render()`

## Properties

---

### container (protected)

```
container: \horstoeko\multidocumentor\Container\MultiDocContainer
```

Container (Settings)

### finderService (protected)

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

Finder Service

### renderService (protected)

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

Render Service

## Methods

---

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

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### **Parameters**

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

### **render (public)**

```
render()
```

# MultiDocFinderService

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

## Summary

---

### Public methods

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

## Properties

---

### container (protected)

```
container: \horstoeko\multidocumentor\Container\MultiDocContainer
```

Container (Settings)

### finder (protected)

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

Internal finder component

## Methods

---

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

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

### **Parameters**

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

### **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

# MultiDocMarkupHtmlSkeletonService

*Service class which renders the markup in plain HTML format wrapped into a HTML skeleton*

## Summary

---

### Public methods

- `beforeGetMarkupOutput()`

### Private methods

- `createInlineFonts()`
- `createInlineCss()`

## Methods

---

### **beforeGetMarkupOutput (public)**

```
beforeGetMarkupOutput ( )
```

### **createInlineFonts (private)**

```
createInlineFonts ( )
```

Create inline fonts by font definition as an array evaluable in twig template

### **createInlineCss (private)**

```
createInlineCss ( )
```

Create the inline CSS

# 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()`
- `setIncludedFiles()`
- `render()`

## Properties

---

### container (protected)

```
container: \horstoeko\multidocumentor\Container\MultiDocContainer
```

Container (Settings)

### includedFiles (protected)

```
includedFiles
```

Files to handle

## Methods

---

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

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

#### **Parameters**

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

### **setIncludedFiles (public)**

```
setIncludedFiles(array $files)
```

#### **Parameters**

array \$files

### **render (public)**

```
render()
```

# MultiDocTwigService

*Service class which will render the twig templates*

## Summary

---

### Public methods

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

## Properties

---

### container (protected)

```
container: \horstoeko\multidocumentor\Container\MultiDocContainer
```

Container (Settings)

### twigEngine (protected)

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

The Twig engine



## Methods

---

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

```
__construct(\horstoeko\multidocumentor\Container\MultiDocContainer $container)
```

Constructor

#### **Parameters**

\horstoeko\multidocumentor\Container\MultiDocContainer \$container

### **addTemplateDirectory (public)**

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

#### **Parameters**

string \$directory

### **render (public)**

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

#### **Parameters**

string \$name

array \$data

# MultiDocTools

*Class representing a collection of tools*

## Summary

---

### Public methods

- `isArray()`
- `arrayKeyExists()`
- `arrayCollapse()`
- `value()`
- `dataGet()`
- `objectToArray()`
- `classImplementsInterface()`

## Methods

---

### **isArray (public)**

```
isArray(mixed $value)
```

Determine whether the given value is array accessible.

#### **Parameters**

mixed \$value

### **arrayKeyExists (public)**

```
arrayKeyExists(mixed $array, mixed $key)
```

Determine if the given key exists in the provided array.

#### **Parameters**

mixed \$array

mixed \$key

### **arrayCollapse (public)**

```
arrayCollapse(mixed $array)
```

Collapse an array of arrays into a single array.

## Parameters

mixed \$array

## value (public)

```
value(mixed $value, mixed $args)
```

Return the default value of the given value.

## Parameters

mixed \$value

mixed \$args

## dataGet (public)

```
dataGet(mixed $target, mixed $key, mixed $default = null)
```

Get an item from an array or object using "dot" notation.

## Parameters

mixed \$target

mixed \$key

mixed \$default = null

## objectToArray (public)

```
objectToArray(mixed $obj)
```

Convert an object to an associate array

## Parameters

mixed \$obj

## classImplementsInterface (public)

```
classImplementsInterface(string $className, string $interface)
```

Checks if a class implements an interface

## Parameters

string \$className

string \$interface

# 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