

Templates

The screenshot shows a dark-themed API documentation page for the `DotnetApiCatalog` class. At the top, there's a navigation bar with links for `Docs`, `API`, and `Extensions`. Below the navigation is a search bar. The main content area has a header for `Class DotnetApiCatalog`. It includes sections for `Inheritance`, `Methods`, and `Properties`. The `Methods` section contains a code snippet for `GenerateManagedReferenceYamlFiles` and its parameters (`configPath` and `options`). The `Properties` section lists inherited members like `object.Equals`, `object.GetHashCode`, etc. On the right side, there's a sidebar titled "IN THIS ARTICLE" with links to `GenerateManagedReferenceYamlFiles`, `GenerateManagedReferenceYamlFile`, and `GenerateManagedReferenceYamlFileString`.

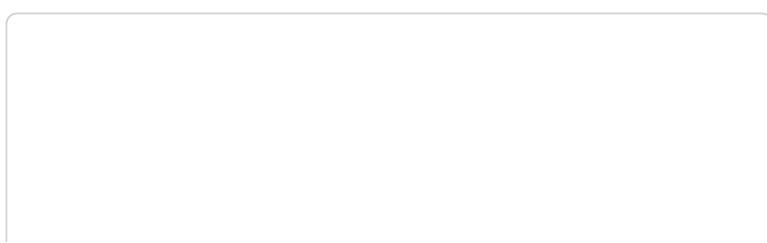
modern ↗

The modern template

The screenshot shows a dark-themed API documentation page for the `ExpandedDependencyMap` class. At the top, there's a navigation bar with links for `Tutorials`, `Guidelines`, `Specifications`, `API Documentation`, and `Themes And Templates`. Below the navigation is a search bar. The main content area has a header for `Class ExpandedDependencyMap`. It includes sections for `Inheritance`, `Methods`, and `Properties`. The `Properties` section lists inherited members like `Object.ToString`, `Object.Equals`, etc. The `Methods` section contains a code snippet for `ConstructFromDependencyGraph` and its parameters (`DependencyGraph dg`). On the right side, there's a sidebar titled "IN THIS ARTICLE" with links to `ConstructFromDependencyGraph`, `View Source`, and `Improve this Doc`.

default ↗

The default template



multiple templates must be separated by `,` with no spaces. The other way is to set key-value mapping in `docfx.json`:

```
{
  ...
  {
    "build": [
      ...
      "template": "custom",
      ...
    ]
  }
}

{
  ...
  {
    "build": [
      ...
      "template": ["default", "X:/template/custom"],
      ...
    ]
  }
}
```

NOTE
The template path could either be a zip file called `<template>.zip` or a folder called `<template>`.

WARNING
DocFX has embedded templates: `default`, `iframe.html`, `statistic` and `common`. Please avoid using them.

statictoc

The template similar to default template however with static toc. With static toc, the generated web pages can be previewed from local file system.

`docfx.json: "template": "statictoc"`

`docfx: -t statictoc`

Subtraction(Cat<T, K>, Int32) ^
Similar with operator `+`, refer to that topic.

Declaration

```
public static int operator -(Cat<T, K> lsr, int rsr)
```

Parameters

<code>lsr</code>	<code>Cat<T, K></code>
<code>rsr</code>	<code>System.Int32</code>

Returns

<code>System.Int32</code>

Explicit Interface Implementations

IAnimal.Eat() ^
Implementation of Eat(food)

Declaration

```
void IAnimal.Eat(string food)
```

mathew

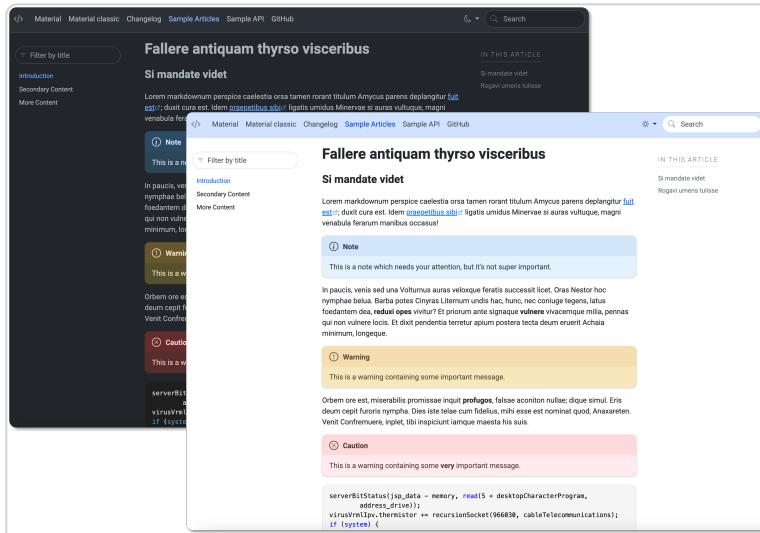
A simple template

`docfx.json: "template":`

`["default", "mathew/src"]`

`docfx: -t default,mathew/src`

docfx init: git clone
<https://github.com/MathewSachin/docfx-tmpl.git> mathew

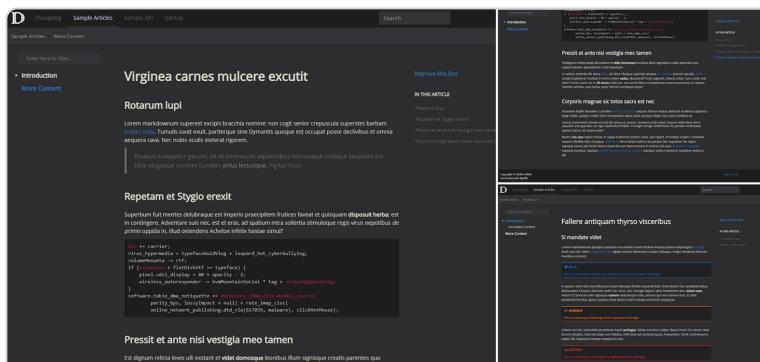


DocFX Material ↗

A simple material theme for DocFX

docfx.json: "template":

```
["default", "material/material"]
docfx: -t default,material/material
docfx init: git clone
https://github.com/ovasquez/docfx-material.git material
```



darkFX ↗

A dark theme for DocFX .

```

docfx.json: "template": 
["default","templates/darkfx"]
docfx: -t default,templates/darkfx
docfx init: git clone
https://github.com/steffen-
wilke/darkfx.git darkfx

```

The screenshot shows a detailed view of a class definition in a Unity-like documentation theme. The class is named `PlayerMovement`. It is described as a basic movement script using WASD/Arrow keyboard input. The code snippet shows the declaration:

```
public class PlayerMovement : MonoBehaviour
```

Below the code, there are sections for **Fields** and **Speed**, which is described as player speed in units per second. The field is declared as `public float Speed`. There is also a **Field Value** section for `Speed`, indicating it is of type `System.Single`.

UnityFX ↗

A theme for Unity-esque documentation

```

docfx.json: "template": 
["default","templates/unity"]
docfx: -t statictoc

```

The screenshot shows a page from the UnityFX theme. The top navigation bar includes links for ACME, Articles, API Documentation, and GitHub. The main content area has a breadcrumb trail: Articles / Introduction. The page title is `Fallere antiquam thyrso visceribus`. A sidebar on the left contains sections for Introduction, Secondary Content, More Content, and Even More Content.

The main content area features several blocks of secondary content with different styles:

- Si mandate videt**: A note block containing placeholder Latin text.
- This is a note**: A note block containing placeholder Latin text.
- This is a tip**: A note block containing placeholder Latin text.
- This is a warning containing some important message**: A note block containing placeholder Latin text.
- Orbem ore est, miserabilis promissae inquit profugus, falsae aconitum nullata; dique simul. Eris deum cepit furoris nymphs. Dies iste telle cum fidelus, mili esse est nominat quod, Anxeraten. Vomit Confluentre, inplet, ibi inspicunt tamque meista his suis.**: A note block containing placeholder Latin text.

At the bottom of the page, there is a code block containing C# code related to ZLIB decompression:

```
# Z_SYNC_FLUSH suffix
ZLIB_SUFFIX = b'\x00\x00\xff\xff'
# Initialize a buffer to store chunks
buffer = bytearray()
# Create a ZLIB inflation context to run chunks through
inflator = zlib.decompressobj()
```

The footer of the page indicates it was generated by DocFX.

DiscordFX ↗

DocFX template to create documentation similar to Discord

```
docfx.json: "template":  
["default", "templates/discordfx"]  
docfx: -t default,templates/discordfx
```

The screenshot shows a dark-themed API documentation page for the `Singulink.IO` library. On the left is a sidebar with a 'File System' icon, a search bar, and a navigation menu with links to 'Articles', 'API Documentation', and 'GitHub'. Below these are sections for 'Singulink.IO' and 'DirectoryPath', with 'Methods' expanded to show `GetAssemblyLocation`, `GetCurrent`, `GetMountingPoints`, `GetSpecialFolder`, `GetTemp`, `Parse`, `ParseAbsolute`, `ParseRelative`, and `SetCurrent`. Under `Parse`, `ParseAbsolute` is highlighted. The main content area has a header 'Method ParseAbsolute'. It contains two method signatures: `ParseAbsolute(ReadOnlySpan<Char>, PathOptions)` and `ParseAbsolute(ReadOnlySpan<Char>, PathFormat, PathOptions)`. Each signature includes a 'Declaration' section with the corresponding C# code, a 'Parameters' table, and a 'Returns' section. The 'Parameters' table for the first method has three columns: TYPE, NAME, and DESCRIPTION. The first parameter, `path`, is described as 'An absolute directory path'. The second parameter, `options`, is described as 'Specifies the path parsing options'. The 'Returns' section for both methods indicates they return `IAbsoluteDirectoryPath`.

SingulinkFX

Customizable responsive DocFX template designed with memberpage plugin compatibility to produce docs similar to Microsoft .NET docs.

```
docfx.json: "template":  
["default", "templates/singulinkfx"]  
docfx: -t default,templates/singulinkfx
```

Enter here to filter...[Installation](#)

DocFX Minimal Template

DocFX Minimal Template is a minimal theme derived from default template.

Features

- Full width (Container-fluid in Bootstrap)
- Minimal white pages
- Simple interface without a breadcrumb
- Table of contents aligned left

Installation

1. Download source files of DocFX minimal template as a zip file from [Here](#) or [GitHub](#).
2. Create `templates` folder in your docfx project folder.
3. Extract the zip file and copy `minimal` folder into the `templates` folder.
4. Apply minimal template by adding `minimal` in your `docfx.json`.

```
"build": {  
  "template": [  
    "default", "templates/minimal"  
  ]  
}
```

[Improve this Doc](#)[IN THIS ARTICLE](#)[Features](#)[Installation](#)Generated by **DocFX**[Back to top](#)

Minimal ↗

A minimal template.

```
docfx.json: "template":  
["default", "templates/minimal"]  
docfx: -t default,templates/minimal
```

Packages

The screenshot shows the Pet Store API documentation. On the left, there's a sidebar with a search bar and sections for Pet Store API (pet, store, user) and Contacts API. The main content area is titled "Pet" with a description "Description for pet tag". It shows an "AddPet" operation under "Request" with a POST /pet endpoint. Parameters include a required "body" parameter of type string with notes "Pet object that needs to be added to the store". Responses show a 405 status code with the note "Invalid input". A note at the bottom says "NOTE: Add pet only when you need it." Below this is an "UpdatePet" operation with a description "Update an existing pet".

rest.tagpage ↗

It splits the *REST* model into tag level model. With this plugin enabled, operations with the same tag are grouped into one page. If the operation is in multiple tags, it would be included in first tag level page.

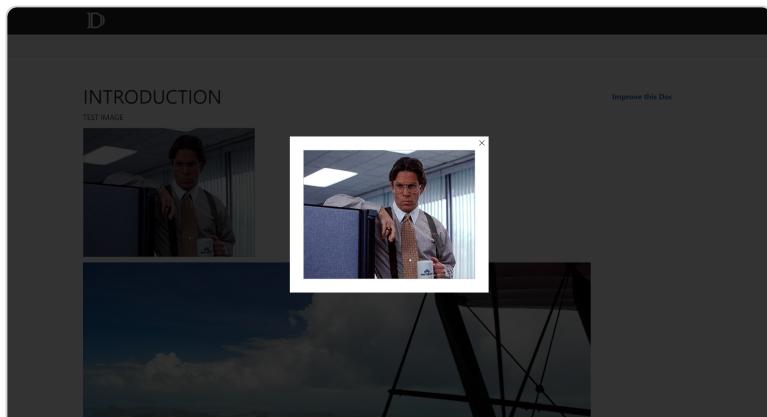
```
docfx.json: template: ["default", "  
<output>/rest.tagpage.<version>/content"]  
docfx: -t default,<output>/rest.tagpage.  
<version>/content  
docfx init: nuget install rest.tagpage -  
OutputDirectory <output>
```

The screenshot shows the Pet Store API documentation. The sidebar includes a search bar and sections for Pet Store API (addPet, createUser, etc.) and Contacts API. The main content area is titled "DeletePet" with a description "Deletes a pet". It shows a "DeletePet" operation under "Request" with a DELETE /pet/{petId} endpoint. Parameters include an "api_key" parameter of type string and a required "petId" parameter of type integer with notes "Pet id to delete". Responses show a 400 status code with the note "Invalid ID supplied" and a 404 status code with the note "Pet not found".

rest.operationpage ↗

It splits the *REST* model into operation level model. If it's enabled together with `rest.tagpage`, the *REST* model will split to tag level first, then split to operation level.

```
docfx.json: template: ["default", "  
<output>/rest.operationpage.  
<version>/content"]  
docfx: -t default,  
<output>/rest.operationpage.  
<version>/content  
docfx init: nuget install  
rest.operationpage -OutputDirectory  
<output>
```

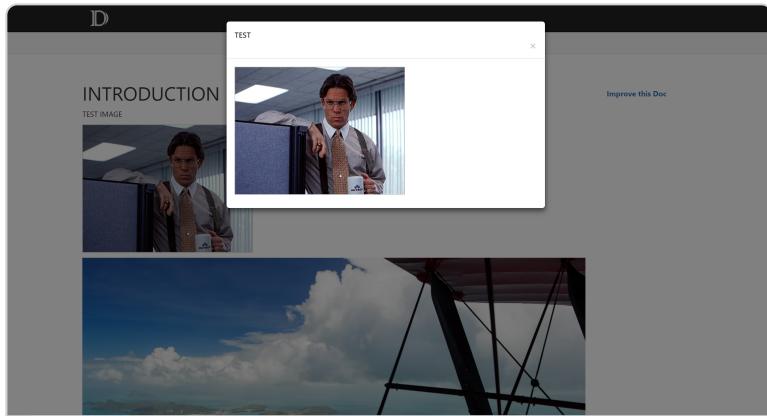


[docfx-lightbox-plugin](#) (Featherlight) ↗

A template which adds a lightbox to each image, using the jquery plugin Featherlight.

```
docfx.json: "template": ["default", "docfx-  
lightbox-plugin/templates/lightbox-  
featherlight"]  
docfx: -t default, docfx-lightbox-  
plugin/templates/lightbox-featherlight
```

```
docfx init: git clone  
https://github.com/roel4ez/docfx-  
lightbox-plugin.git docfx-lightbox-plugin
```



[docfx-lightbox-plugin \(Bootstrap Modal\)](#)

A template which adds a lightbox to each image, using the Modal window from Bootstrap.

```
docfx.json: "template": ["default", "docfx-  
lightbox-plugin/templates/bootstrap-  
modal"]  
docfx: -t default,docfx-lightbox-  
plugin/templates/bootstrap-modal  
docfx init: git clone  
https://github.com/roel4ez/docfx-  
lightbox-plugin.git docfx-lightbox-plugin
```

The screenshot shows the API Documentation for the 'CatLibrary' namespace. The 'ICat' interface is selected. The interface diagram shows 'ICat' extending from 'Animal'. Both have a 'Eat' method. Below the interface, a list of 'Inherited Members' includes 'Animal.Name', 'Animal.Eat<String>', 'Animal.Eat<Tool>', 'Animal.Eat<Tool>[Tool]', and 'Animal.Eat<String>'. The 'Name' member is highlighted with a red border.

[DocFx.Plugins.PlantUml](#)

A template to render PlantUml diagrams from markdown code blocks.

```
docfx.json: "template":  
["default", "DocFx.Plugins.PlantUml/template"]  
docfx: -t  
default,DocFx.Plugins.PlantUml/template  
docfx init: nuget install  
DocFx.Plugins.PlantUml -ExcludeVersion -  
OutputDirectory .
```

Tools

```
# This is an automatically generated file
items:
- name: Getting started
  href: getting-started/README.md
  items:
  - name: Using DocFx and Companion Tools to generate a Documentation website
    href: getting-started/README.md
  - name: Deploy the DocFx Documentation website to an Azure Website automatically
    href: getting-started/deploy-docfx-azure-website.md
  - name: Customize the Look and Feel
    href: getting-started/customize-look-and-feel.md
```

[DocFxTocGenerator](#)

Generate a Table of Contents (TOC) in YAML format for DocFX. It has features like the ability to configure the order of files and the names of documents and folders. This tool is part of the DocFx Companion Tools set that can be installed using Chocolatey.

```
docfx: TocGenerator -d <docs folder> [-o
<output folder>] [-vsi]
docfx init: git clone
https://github.com/Ellerbach/docfx-
companion-tools.git
```

```
cs\fr\plant-production\ex
/.. ./attachments/de.png
```

erenced:

```
ttachments\de.png
ttachments\en.jpg
urn code 1
```

[DocLinkChecker](#)

Validate links in documents and check for orphaned attachments in the .attachments folder. The tool indicates whether there are errors or warnings, so it can be used in a CI pipeline. It can also clean up orphaned attachments automatically. This tool is part of the DocFx Companion Tools set that can be installed using Chocolatey.

```
docfx: DocLinkChecker -d <docs folder> [-vac]
docfx init: git clone
https://github.com/Ellerbach/docfx-companion-tools.git
```

```
Translating C:\DMP\userdocs\de\data-size-estimation.md
Translating ...
Saving C:\DMP\userdocs\fr\data-size-estimation.md
Translating C:\DMP\userdocs\de\index.md
Translating ...
Saving C:\DMP\userdocs\fr\index.md
Translating C:\DMP\userdocs\de\plant-production\example.md
Translating ...
Saving C:\DMP\userdocs\fr\plant-production\example.md
Translating C:\DMP\userdocs\de\plant-production\second-file.md
Translating ...
Saving C:\DMP\userdocs\fr\plant-production\second-file.md
Translating C:\DMP\userdocs\de\plant-production\another-dir\another\and-another\something.md
Translating ...
Saving C:\DMP\userdocs\fr\plant-production\another-dir\another\and-another\something.md
Process finished. 5 translated and properly created. Please make sure to run the Markdown linter and also check the file links and images.
```

DocLanguageTranslator ↗

Allows to generate and translate automatically missing files or identify missing files in multi language pattern directories. This tool is part of the DocFx Companion Tools set that can be installed using Chocolatey.

```
docfx: DocLanguageTranslator -d <docs folder> [-k <key>] [-l <location>] [-cv]
docfx init: git clone
https://github.com/Ellerbach/docfx-companion-tools.git
```

The screenshot shows a browser window with the URL localhost:8080/docs/getting-started/README.html. The page title is "Using DocFx and Companion Tools to generate a Documentation website". The left sidebar contains a navigation menu with sections like "Documentation", "Getting started", "Reference", and "Quick Start". The "Getting started" section is expanded, showing sub-links such as "Using DocFx and Companion Tools to generate a Documentation website", "Deploy the DocFx Documentation website to an Azure Website automatically", "Guidelines", "Architecture decisions", "Tooling agreements", and "Templates". The main content area features a "Quick Start" section with a tip about generating a website from scratch using a QuickStart folder. It also includes a "TIP" section with steps for setting up a documentation website using Azure DevOps and Azure App Service.

DocFx Quick Start

A repo containing documentation, configuration and sample pipelines to help you get started quickly with DocFx. The Quick Start can be used as a reference or to copy elements from it to your own repo. The Quick Start itself can be generated to a website using DocFx as well. It uses the DocFx Companion Tools *DocFxTocGenerator* and *DocLinkChecker*.

```
docfx init: git clone  
https://github.com/mtirionMSFT/DocFxQuickStart.git
```



Addin for Cake Build System ↗

Cake AddIn that generates documentation for .Net API reference and markdown files using DocFx.