

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 light-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`). The `Properties` section lists inherited members like `Object.ToString`, `Object.Equals`, etc.

default ↗

The default template

The screenshot shows the DocFX User Manual interface. On the left, there's a navigation sidebar with sections like 'Getting Started', 'Content', 'Template', and 'Extensibility'. The main content area displays code snippets for docfx.json files, specifically focusing on the 'template' key. A note at the bottom says: 'The template path could either be a zip file called <template>.zip or a folder called <template>.' A warning below it states: 'DocFX has embedded templates: default, iframe.html, statictoc 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

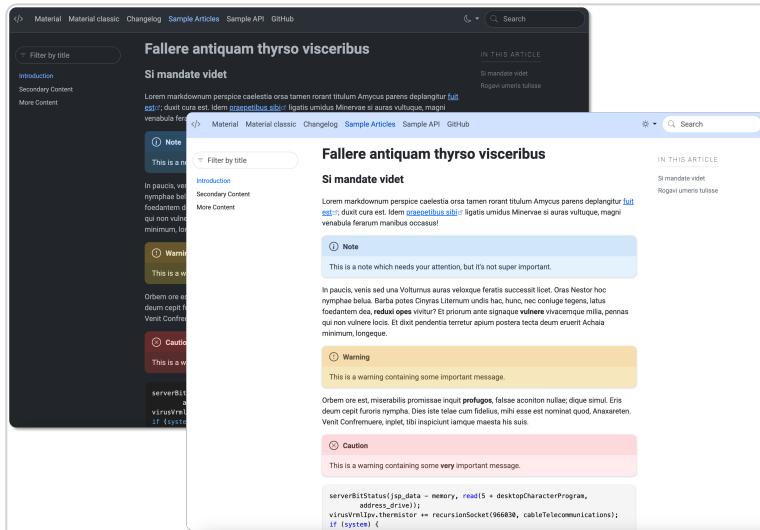
The screenshot shows the API Documentation interface. The left sidebar lists categories like 'CatLibrary', 'Microsoft.DevDiv', and 'MRef.Demo.Enumeration'. The main content area shows the 'Subtraction' operator for the 'Cat<T, K>' class. It displays the declaration: 'public static int operator -(Cat<T, K> lsr, int rsr)'. Below it, parameters 'lsr' and 'rsr' are shown with types 'Cat<T, K>' and 'System.Int32' respectively. The 'Operators' section is highlighted, showing other operators like Addition, Explicit, and Subtraction.

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

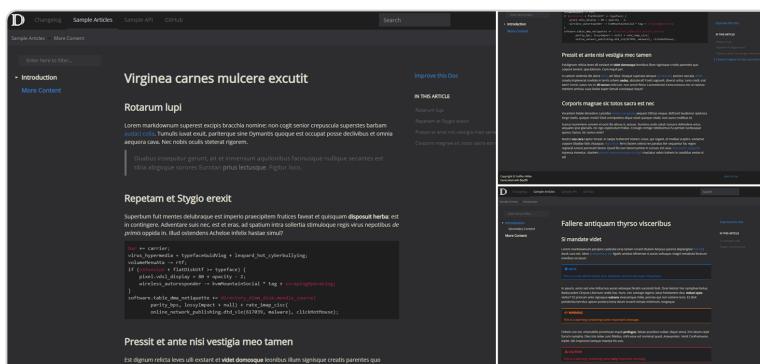


DocFX Material

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 documentation page for the Unity Asset Store. The top navigation bar includes links for Manual, Scripting API, Submitting API, Submitting API, Assets, and PlayerMovement. A search bar is present. The main content area is titled 'Class PlayerMovement' and describes it as a basic movement script using WASD/Arrow keyboard input. It shows the class hierarchy: System.Object > PlayerMovement, with a namespace of Assets.cs. The code block contains the following C# code:

```

public class PlayerMovement : MonoBehaviour

```

Below the code, there are sections for 'Fields' and 'Speed'. The Speed field is described as 'Player speed in units per second' and has a declaration of 'public float Speed'. A 'Field Value' section shows 'Type: System.Single'.

UnityFX ↗

A theme for Unity-esque documentation

```

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

```

The screenshot shows a documentation page for ACME. The left sidebar includes links for ACME, Articles, API Documentation, and GitHub. The main content area is titled 'Articles / Introduction' and features a section titled 'Fallere antiquam thyrso visceribus'. Below this is a 'Si mandate videt' section containing a note: 'This is a note which needs your attention, but it's not super important.' Another note follows: 'In paucis, venis set una Volturnus auris veloxque ferat successus licet. Oras Nestor hoc nymphae belua. Barba potes Cineras Liternum undis hac, hinc, nec conlige tegens, magni venabula ferum manibus occidit.' A 'This is a tip.' note is also present. The page concludes with a code block:

```

# 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()

```

In the bottom left corner of the sidebar, it says '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 interface. On the left is a sidebar with a 'File System' icon, a search bar, and sections for 'Articles', 'API Documentation', and 'GitHub'. Below these are navigation links for 'Singulink.IO' and a 'filter' input field. The main content area has a header 'Method ParseAbsolute' and two method definitions: 'ParseAbsolute(ReadOnlySpan<Char>, PathOptions)' and 'ParseAbsolute(ReadOnlySpan<Char>, PathFormat, PathOptions)'. Each method includes a 'Declaration' code snippet, 'Parameters' table, and 'Returns' type information. The 'Parameters' table for the first method is as follows:

TYPE	NAME	DESCRIPTION
ReadOnlySpan<Char>	path	An absolute directory path
PathOptions	options	Specifies the path parsing options.

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 for the Pet resource. 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 has a title "Pet" with a description "Description for pet tag". It includes an "AddPet" operation with a POST request to "/pet". Parameters for the body are shown as a table with columns Name, Type, Value, and Notes. A note says "Pet object that needs to be added to the store". Responses include a 405 status code with the message "Invalid input". Below this is a note: "NOTE: Add pet only when you need it." Another section "UpdatePet" is also present.

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.

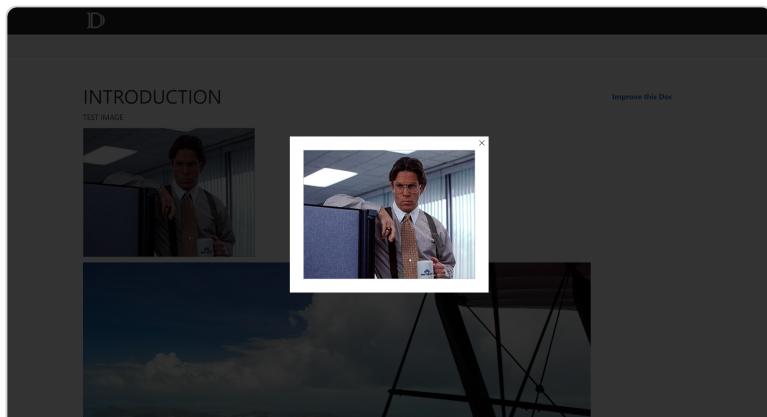
```
docfx.json: template: ["default", "  
<output>/rest.tagpage.<version>/content"]  
docfx: -t default,<output>/rest.tagpage.  
<version>/content  
docfx init: nuget install rest.tagpage -  
OutputDirectory <output>
```

The screenshot shows the Pet Store API documentation for the DeletePet operation. The sidebar is identical to the previous screenshot. The main content area has a title "DeletePet" with a description "Deletes a pet". It includes a "pet" section with a "DeletePet" operation using a DELETE request to "/pet/{petId}". Parameters for the path are shown as a table with columns Name, Type, Value, and Notes. A note says "Pet id to delete". Responses include a 400 status code with the message "Invalid ID supplied" and a 404 status code with the message "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>
```

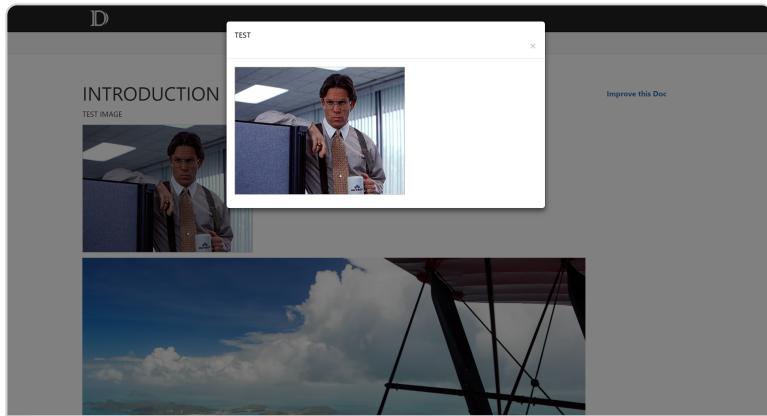


[docfx-lightbox-plugin](#) (Featherlight) ↗

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 'Interface ICat' page. The interface is defined with a single method 'Eat'. It inherits from the 'Animal' interface, which has methods 'Name', 'Eat<Tool>', and 'Eat<String>'. The 'CatLibrary' namespace is specified at the bottom.

[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
```

Using DocFx and Companion Tools to generate a Documentation website

Quick Start

TIP

TL;DR

If you want an even easier website with all your documentation coming from Markdown files and comments coming from code, you can use DocFx. The website generated by DocFx also includes fast search capabilities. There are some gaps in the DocFx solution, but we've provided companion tools that help you fill those gaps. Also see the blog post [Providing quality documentation in your project with DocFx and Companion Tools](#) for more explanation about the solution.

IN THIS ARTICLE

- Quick Start
- Documents and projects folder structure
- Reference documentation from source code
- 1. Install DocFx and markdownlint cli
- 2. Configure DocFx
- 3. Setup some basic documents
- 4. Compile the companion tools and run them
- 5. Run DocFx to generate the website
- Deploy to an Azure Website
- References

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.