

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`, showing its namespace (`Microsoft.DocAsCode.Dotnet`) and assembly (`Microsoft.DocAsCode.Dotnet.dll`). A note states it provides access to a .NET API definitions and their associated documentation. The `Inheritance` section shows `Object` as the base class. The `Methods` section contains one method, `GenerateManagedReferenceYamlFiles`, which generates metadata reference YAML files using `dotnet.json` config. The method signature is `public static Task GenerateManagedReferenceYamlFiles(string configPath, DotnetApiOptions options)`. Parameters include `configPath` (the path to `dotnet.json` config file) and `options` (a `DotnetApiOptions` object). The method returns a `Task` object.

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`, showing its inheritance from `Object` and its implementation of `IEquatable<Object>`. The `Methods` section contains one method, `ConstructFromDependencyGraph`, which constructs an expanded dependency map from a dependency graph. The method signature is `public static ExpandedDependencyMap ConstructFromDependencyGraph(DependencyGraph dg)`. Parameters include `dg` (a `DependencyGraph` object).

default ↗

The default template

The screenshot shows the DocFX User Manual interface. In the top navigation bar, there are links for Tutorials, Guidelines, Specifications, and API Documentation. A search bar is also present. The main content area is titled 'Getting Started' and contains sections like 'Quickstart', 'DocFX User Manual', 'HowTo: Filter Out Unwanted APIs', and 'Walkthrough Topics'. On the right side, there's a sidebar titled 'IN THIS ARTICLE' with links to 'Syntax', 'Commands', 'docfx.json Format', 'Properties for metadata', 'Properties for build', and 'Supported File Mapping Form'. Below the sidebar, there are two callout boxes: one blue box labeled 'NOTE' stating 'The template path could either be a zip file called <template>.zip or a folder called <template>.' and a yellow box labeled 'WARNING' stating '<DocFX> has embedded templates: default, iframe.html, statictoc and common. Please avoid using'.

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

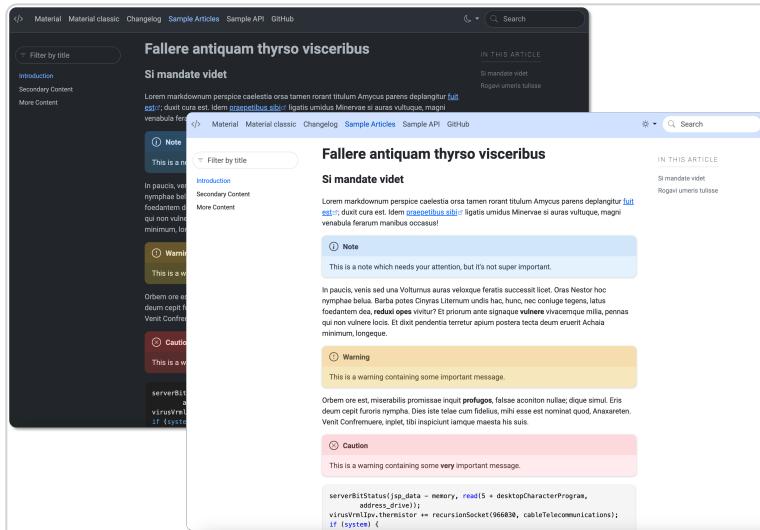
The screenshot shows the API Documentation for the 'CatLibrary'. The left sidebar lists classes like 'Cat<T, K>', 'CatException<T>', 'Complex<T, J>', 'FakeDelegate<T>', 'IAnimal', 'ICat', 'ICatExtension', 'MRefDelegate<K, T, L>', 'MRefNormalDelegate', 'Tom', and 'TomFromBaseClass'. The main content area shows the 'Subtraction(Cat<T, K>, Int32) ^' operator implementation. It includes parameters 'lsr' (Cat<T, K>) and 'rsr' (System.Int32), and a return type of System.Int32. Below this, it shows the 'Explicit Interface Implementations' for 'IAnimal.Eat()' with a 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

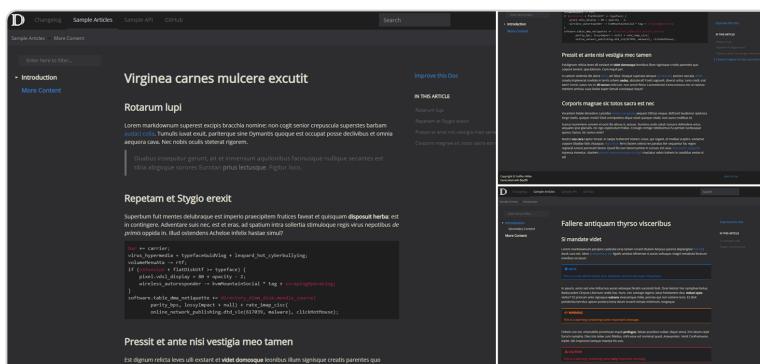


DocFX Material

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 documentation page for the 'PlayerMovement' class. At the top, there's a navigation bar with links for 'Manual', 'Scripting API', 'Submitting API', 'Assets Src', and 'PlayerMovement'. Below the navigation is a search bar with placeholder text 'Enter here to filter...'. A sidebar on the left lists 'Assets Src' and has a selected item 'PlayerMovement (typescript)'. The main content area is titled 'Class PlayerMovement' and describes it as 'Basic movement script using WASD/Arrow keyboard input'. It shows the class hierarchy: 'System.Object' → 'PlayerMovement'. The namespace is 'Assets.FX'. Below this is a 'Syntax' section containing the C# code:

```
public class PlayerMovement : MonoBehaviour
```

. The 'Fields' section contains a 'Speed' field, described as 'Player speed in units per second'. The 'Declaration' section shows the code:

```
public float Speed
```

. The 'Field Value' section shows the type as 'System.Single'. On the right side of the page, there's a sidebar with 'IN THIS ARTICLE' sections for 'Editor' and 'Inspector'.

UnityFX ↗

A theme for Unity-esque documentation

```

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

```

The screenshot shows a documentation page for the article 'Fallere antiquam thyrsō visceribus'. The top navigation bar includes links for 'ACME', 'Articles', 'API Documentation', and 'GitHub'. A sidebar on the left has a 'Introduction' section with 'Secondary Content', 'More Content', and 'Even More Content'. The main content area has a title 'Fallere antiquam thyrsō visceribus' and a subtitle 'Si mandate videt'. It contains several callout boxes: one for 'This is a note which needs your attention, but it's not super important.', another for 'This is a tip.', a warning box for 'This is a warning containing some important message.', and a note box for 'Orbem ore est, miserabilis promissae iniqui profugos, falsae aconitum nullata; dique simul. Eris deum cepit furoris nymphs. Dies iste tēlae cum fidelus, mihi esse est nominat quod. Anxerant. Vomit Confluentre, inplet, ibi inspicunt tamque meista his suis.' At the bottom, there's a code block with C# code:

```

# Z_SYNC_FLUSH suffix
ZLIB_SUFFIX = b'\x00\x00\x00\x00\x00\x00\x00\x00'
# 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, there's a footer: 'ACME Inc. 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. On the left is a sidebar titled 'File System' with sections for 'Articles', 'API Documentation' (which is currently selected), and 'GitHub'. Below these are search and filter fields. 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 has a 'Declaration' section with code snippets, 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 of type 'ReadOnlySpan<Char>' and is described as 'An absolute directory path'. The second parameter 'options' is of type 'PathOptions' and is described as 'Specifies the path parsing options'. The 'Returns' section for both methods indicates the return type is 'IAbsoluteDirectoryPath'. The bottom of the page includes navigation links for 'View Source' and 'Improve this Doc'.

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 a tree view of API tags: 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 the 'Pet' tag. The 'Request' section has a 'POST /pet' button. Below it, there's a table for 'Parameters' with columns 'Name', 'Type', 'Value', and 'Notes'. A note says 'Pet object that needs to be added to the store'. The 'Responses' section shows a table for 'Status Code' (405) and 'Description' (Invalid input). To the right, there are 'View Source' and 'Improve this Doc' links, and a 'IN THIS ARTICLE' sidebar with links like 'addPet', 'updatePet', etc.

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.

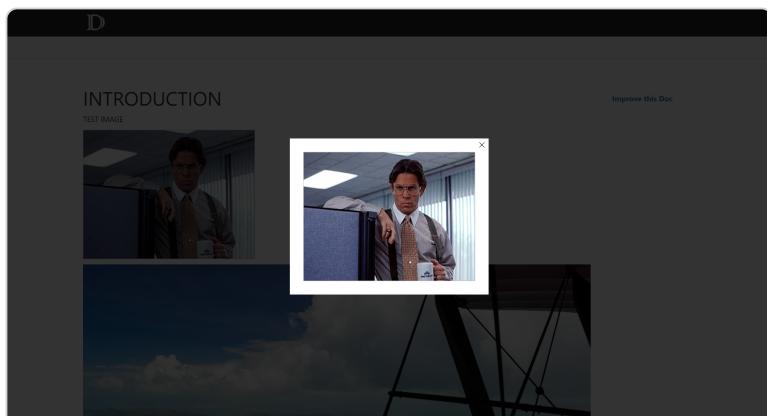
```
docfx.json: template: ["default", "  
<output>/rest.tagpage.<version>/content"]  
docfx: -t default,<output>/rest.tagpage.  
<version>/content  
docfx init: nuget install rest.tagpage -  
OutputDirectory <output>
```

The screenshot shows the Pet Store API documentation. The sidebar is identical to the previous screenshot. The main content area is titled 'DeletePet' with a description 'Deletes a pet'. It shows another 'DeletePet' operation under the 'Pet' tag. The 'Request' section has a 'DELETE /pet/{petId}' button. Below it, there's a table for 'Parameters' with columns 'Name', 'Type', 'Value', and 'Notes'. A note says 'Pet id to delete'. The 'Responses' section shows a table for 'Status Code' (400 and 404) and 'Description' (Invalid ID supplied and Pet not found). To the right, there are 'View Source' and 'Improve this Doc' links, and a 'IN THIS ARTICLE' sidebar with links like 'pet', '>deletePet', etc.

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>
```

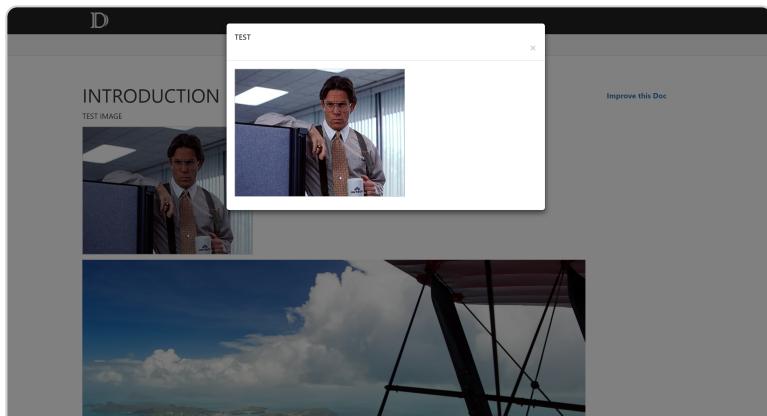


[docfx-lightbox-plugin](#) (Featherlight) ↗

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 interface for the 'ICat' interface. The left sidebar contains a navigation tree with categories like 'CatLibrary', 'Microsoft.DevDiv', 'MRef.Demo.Enumeration', and 'VBTestClass1'. The main content area displays the 'Interface ICat' page. It features a UML-like class hierarchy diagram where 'ICat' is the base interface and 'Animal' is a derived class. Both classes have a 'Eat' method. Below the diagram, a section titled 'Inherited Members' lists methods such as 'Animal.Name', 'Animal.Eat<String>', 'Animal.Eat<Tool>', and 'Animal.Eat<Tool>[Tool]'. On the right side, there are sections for 'Improve this Doc', 'View Source', and 'IN THIS ARTICLE' with links to 'Events' and 'Extension Methods'. The bottom of the page shows the namespace 'CatLibrary' and assembly 'CatLibrary.dll'.

[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
```

```
c:\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

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.